Tutorial zum Web-IO Digital:

Web-IO Digital aus eigener Webseite mit PHP steuern

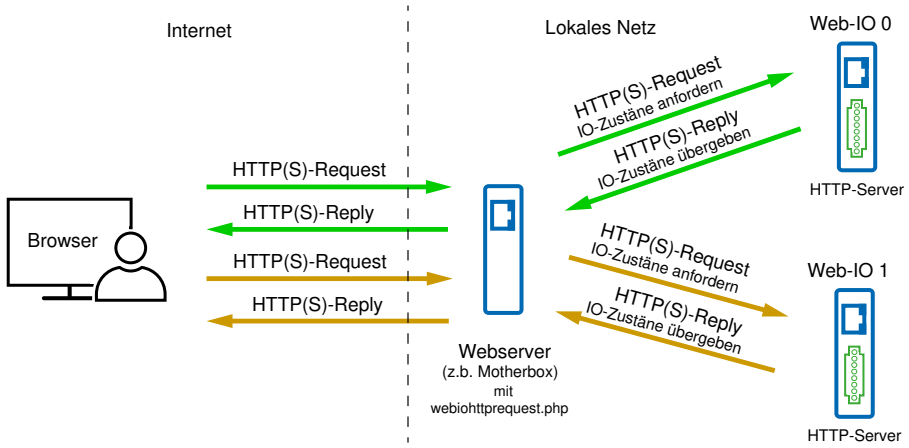
Produktübersicht
Applikationsübersicht

Die Web-IO Digital 4.0-Modelle können über HTTP-Requests angesteuert werden. Die Verwendung von HTTP-Requests zum Datenaustausch wird von den meisten Script- und Hochsprachen unterstützt. PHP steht als Script-Sprache auf fast jedem Webserver zur Verfügung und bietet sich perfekt an, die Kommunikation mit dem Web-IO abzuwickeln.

Das hier gezeigte Beispiel arbeitet zweigeteilt. Es trennt die dynamische Anzeige der IO-Zustände mittels HTML, JavaScript bzw. AJAX (Webseite) und die Kommunikation mit dem Web-IO mittels PHP(-Script).

Kernstück ist das PHP-Script *webiohttprequest.php*, das ohne Anzeigeelemente auskommt und ausschließlich den Datenaustausch übernimmt. Die Webseite, auf der die IO-Zustände angezeigt werden, ruft mittels AJAX das PHP-Script auf und fordert darüber den Status des Web-IO an.

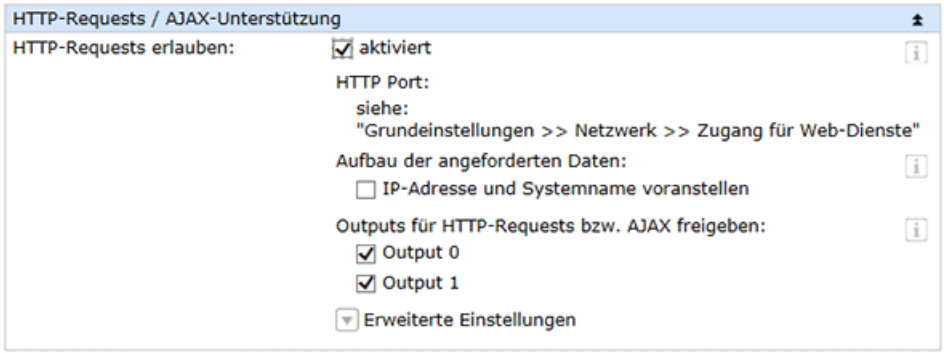
Ein Vorteil dieser Technik besteht darin, dass über dasselbe PHP-Script mehrere Web-IO angesprochen werden können, wobei mitgesendete Parameter das Ziel-Web-IO bestimmen. Liegen die Web-IOs und der PHP-Server hinter einer Firewall, muss nur der Zugang zum PHP-Server nach außen freigegeben werden.



Das folgende Beispiel zeigt, wie das PHP-Script aufgebaut ist und wie es mit einer einfachen Webseite und den notwendigen Funktionen angesteuert wird. Die Anzeige ist für das Web-IO Digital 4.0 2xIn, 2xOut vorbereitet.

Vorbereitungen

- Web-IO mit Spannung versorgen und IOs verdrahten
- Web-IO mit dem Netzwerk verbinden
- IP-Adressen vergeben
- Beim Web-IO im Bereich *Kommunikationswege >> Web-API* den Punkt *HTTP-Request erlauben* aktivieren und die *Outputs zum Schalten freigeben*



Das PHP-Script

Dem PHP-Script *webiohttprequest.php* müssen beim Aufruf diverse Parameter übergeben werden, die als Bestandteil des HTTP-Request übergeben werden.

- **HTTPS**
  HTTPS=0 - der Request wird als HTTP-Request verschickt
  HTTPS=1 - der Request wird verschlüsselt als HTTPS-Request verschickt

- **IP**
  IP=<IP-Adresse des Web-IO>

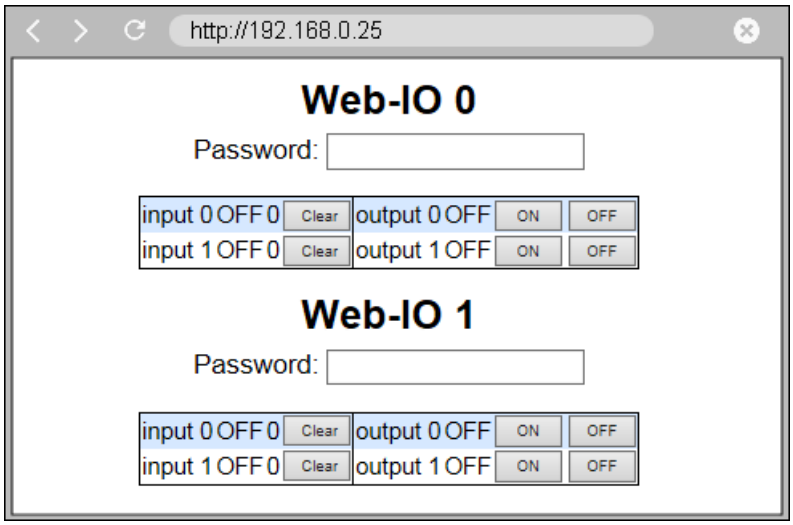
- **PORT**
  PORT=<HTTP(s)-Port des Web-IO>
- **PW**
  PW=<Paswort des Web-IO>
- **COMMAND**
  COMMAND=<Kommando, das an das Web-IO gesendet werden soll>
  mögliche Kommandos: *input, output, counter, outputaccess, single*
- **STATE** (nur bei Kommando *outputaccess*)
  STATE=<Bit-Muster der Schaltzustände, in die die Outputs geschaltet werden sollen, als Hexadezimalzahl>
- **MASK** (nur bei Kommando *outputaccess*
  MASK=<Bitmuster der Outputs, die geschaltet werden sollen, als Hexadezimalzahl>

Die unter COMMAND zu übergebenden Kommandos und Parameter entsprechen den Grundkommandos der HTTP-Requests.

Für das Versenden der HTTP-Requests nutzt das PHP-Script die PHP-Funktionsbibliothek *CURL*.

```
<?php
 $PROTO="http://";
 $PORT="80";
 if (!isset($_GET['COMMAND'])) {die( 'ERROR command' );} else {$COMMAND=$_GET['COMMAND'];}
 if (!isset($_GET['IP'])) {die( 'ERROR ip' );} else {$IP=$_GET['IP'];}
 if (!isset($_GET['HTTPS'])) {$PROTO='http://';}
 else
 { if ($_GET['HTTPS'] == 1)
   { $PROTO='https://';
     $PORT='443';
   }
 }
 if (isset($_GET['PORT'])) {$PORT=$_GET['PORT'];}
 if (!isset($_GET['PW'])) {$PW="";} else {$PW=$_GET['PW'];}
 if (!isset($_GET['MASK'])) {$MASK="0FFF";} else {$MASK=$_GET['MASK'];}
 if (!isset($_GET['STATE']))
 { $STATE="0000";
   $MASK="0000";
 }
 else $STATE=$_GET['STATE'];
 { if ($COMMAND == "outputaccess")
   { $URL=$PROTO.$IP.":".$PORT."/".$COMMAND."?PW=".$PW."&Mask=".$MASK."&State=".$STATE."&";
   }
   else
   { if (substr($COMMAND,0,6) == "single")
     { $URL=$PROTO.$IP.":".$PORT."/".$COMMAND;
     }
     else
     { $URL=$PROTO.$IP.":".$PORT."/".$COMMAND."?PW=".$PW."&";
     }
   }
   $ch = curl_init();
   $options = array(
     CURLOPT_URL            => $URL,
     CURLOPT_RETURNTRANSFER => true,     // return web page
     CURLOPT_HEADER         => false,    // don't return headers
     CURLOPT_ENCODING       => "",       // handle all encodings
     CURLOPT_USERAGENT      => "webio",  // who am i
     CURLOPT_CONNECTTIMEOUT => 20,      // timeout on connect
     CURLOPT_TIMEOUT        => 20,      // timeout on response
     CURLOPT_MAXREDIRS      => 1,       // stop after 10 redirects
     CURLOPT_SSL_VERIFYHOST => false,
     CURLOPT_SSL_VERIFYPEER => false,
   );
   curl_setopt_array( $ch, $options );
   $data = curl_exec($ch);
   if ($data==false)
   { die('ERROR '.$URL);
   }
   else
   { $http_status = curl_getinfo($ch, CURLINFO_HTTP_CODE);
     if ($http_status==200)
     { echo $data;
     }
     else
     { die('ERROR '.$URL.' status '.$http_status);
     }
   }
   curl_close($ch);
 }
?>
```

Grundaufbau der Webseite



Das HTML-Grundgerüst dieser Seite sieht so aus:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
<title>Web-IO Digital, Cross Origin</title>
<style type="text/css">
* { font-family:arial; }
table { font-size:14px; text-align:center; padding-left:5px; padding-right:5px;}
.borderLeft { border-left:1px solid #000000; }
.button { font-size:9px; width:40px; }
.name { font-size:20px; font-weight:bold; text-align:center }
.table { background-color:#d6e8ff; border-collapse:collapse; border:1px solid #000000; }
.whiteBack { background-color:#ffffff; }
</style>
<script language="JavaScript" type="text/javascript">

  ........
  ........

</script>
</head>
<body onload="CommandLoop()">
  <form>
    <div align="center">
      <h2>Web-IO 0</h2>
      <span>Password: </span>
      <input id="webio0pw" type="password" maxlength="31" size="20"">
    </div>
  </form>
  <table align="center" class="table">
    <tr>
      <td>input 0</td>
      <td id="webio0input0">OFF</td>
      <td id="webio0counter0">0</td>
      <td>
        <input class="button" onclick="clearCounter(0,0);" type="button" value="Clear">
      </td>
      <td class="borderLeft">output 0</td>
      <td id="webio0output0">OFF</td>
      <td>
        <input class="button" onclick="setOutput(0,0,1);" type="button" value="ON">
        <input class="button" onclick="setOutput(0,0,0);" type="button" value="OFF">
      </td>
    </tr>
    <tr class="whiteBack">
      <td>input 1</td>
      <td id="webio0input1">OFF</td>
      <td id="webio0counter1">0</td>
      <td>
        <input class="button" onclick="clearCounter(0,1);" type="button" value="Clear">
      </td>
      <td class="borderLeft">output 1</td>
      <td id="webio0output1">OFF</td>
      <td>
        <input class="button" onclick="setOutput(0,1,1);" type="button" value="ON">
        <input class="button" onclick="setOutput(0,1,0);" type="button" value="OFF">
      </td>
    </tr>
  </table>
  <br />
  <form>
    <div align="center">
      <h2>Web-IO 1</h2>
      <span>Password: </span>
      <input id="webio1pw" type="password" maxlength="31" size="20"">
    </div>
  </form>
  <table align="center" class="table">
```

```
    <tr>
      <td>input 0</td>
      <td id="webio1input0">OFF</td>
      <td id="webio1counter0">0</td>
      <td>
        <input class="button" onclick="clearCounter(1,0);" type="button" value="Clear">
      </td>
      <td class="borderLeft">output 0</td>
      <td id="webio1output0">OFF</td>
      <td>
        <input class="button" onclick="setOutput(1,0,1);" type="button" value="ON">
        <input class="button" onclick="setOutput(1,0,0);" type="button" value="OFF">
      </td>
    </tr>
    <tr class="whiteBack">
      <td>input 1</td>
      <td id="webio1input1">OFF</td>
      <td id="webio1counter1">0</td>
      <td>
        <input class="button" onclick="clearCounter(1,1);" type="button" value="Clear">
      </td>
      <td class="borderLeft">output 1</td>
      <td id="webio1output1">OFF</td>
      <td>
        <input class="button" onclick="setOutput(1,1,1);" type="button" value="ON">
        <input class="button" onclick="setOutput(1,1,0);" type="button" value="OFF">
      </td>
    </tr>
  </table>
</body>
</html>
```

Auf den *head*-Bereich und die Style-Informationen wollen wir hier nicht näher eingehen. Die Inhalte des *script*-Bereichs werden im weiteren Verlauf beschrieben. Wichtig sind im *body*-Bereich *id*-Benennungen, über die in den JavaScript-Funktionen die Objekte bzw. Tabellenzellen angesprochen werden. Den Buttons wird über *onclick* die aufzurufende Funktion zugeteilt. Als Parameter wird die Nummer des Web-IO gefolgt von der Nummer der Counter bzw. der Outputs übergeben. Bei den Outputs gibt der zweite Parameter den Status (0=OFF, 1=ON) an.

Bei Einhaltung dieser Systematik kann das Beispiel ohne Änderung der JavaScript-Funktionen durch Hinzufügen weiterer Tabellenzeilen an weitere Web-IO-Modelle angepasst werden.

Globale Variablen und Funktionen im JavaScript

Zunächst müssen einige globale Variablen deklariert werden.

```
var MaxI = 2;
var MaxO = 2;
var WebioIP = ['192.168.0.25', '192.168.0.26'];
var WebioPort = ['80', '80'];
var ApplicationStep = 0;
var Interval = 250;

// Converting hexadecimal string to Integer
function HexToInt(HexStr)
{ var TempVal;
  var HexVal=0;
  for( var i=0; i<HexStr.length;i++)
  { if (HexStr.charCodeAt(i) > 57)
    { TempVal = HexStr.charCodeAt(i) - 55;
    }
    else
    { TempVal = HexStr.charCodeAt(i) - 48;
    }
    HexVal=HexVal+TempVal*Math.pow(16, HexStr.length-i-1);
  }
  return HexVal;
}
```

HTTP-Requests versenden und HTTP-Replys empfangen

```
// Sending command lines to Web-IO and receiving IO State
function DataRequest(WebioNo, SendString)
{ var xmlHttp;
  if( window.ActiveXObject )       // Internet Explorer
  { xmlHttp = new ActiveXObject( "Microsoft.XMLHTTP" );
  }
  else if(window.XMLHttpRequest )  // Mozilla, Opera und Safari
  { xmlHttp = new XMLHttpRequest();
  }
  if (xmlHttp)
  { xmlHttp.onreadystatechange = function()
    { if (xmlHttp.readyState == 4)
      { if (xmlHttp.status == 200)
        { if (xmlHttp.responseText.length > 0)
          { updateDisplay(WebioNo, xmlHttp.responseText);
          }
          xmlHttp=null;
        }
      }
    }
    xmlHttp.open("GET", SendString, true);
    xmlHttp.setRequestHeader("If-Modified-Since", "Thu, 1 Jan 1970 00:00:00 GMT");
    xmlHttp.send(null);
  }
}
```

Die Abwicklung von HTTP-Request und HTTP-Reply übernimmt die Funktion DataRequest. Leider nutzt der Internet Explorer für die Verarbeitung von HTTP-Requests andere Mechanismen als die anderen Browser. Deshalb prüft die Funktion zunächst, in welchem Browser die Webseite ausgeführt wird. Der als SendString übergebene HTTP-Request wird dann an den Server bzw. das Web-IO gesendet, von dem auch die Webseite geladen wurde. Die Antwort, also der HTTP-Reply, wird dann an die Funktion UpdateDisplay übergeben.

Zyklische Abfrage von Inputs, Outputs und Countern

```
// Preparing command lines for cycle sending
function CommandLoop()
{ var Command ='';
  var CommandString ='';
  var WebioNo = 0;
  ApplicationStep++;
  switch(ApplicationStep)
  { case 1:
      Command = 'input';
    break;
    case 2:
      Command = 'input';
      WebioNo = 1;
    break;
    case 3:
      Command = 'output';
    break;
    case 4:
      Command = 'output';
      WebioNo = 1;
    break;
    case 5:
      Command = 'counter';
    break;
    case 6:
      Command = 'counter';
      WebioNo = 1;
      ApplicationStep = 0;
    break;
  }
  var IOPassword = document.getElementById('webio'+WebioNo+'pw').value;
  CommandString = 'webiohttprequest.php?IP='+WebioIP[WebioNo]+'&PORT='+WebioPort[WebioNo]+'&COMMAND='+Command+'&
  if (WebioPort[WebioNo] == '443')
  { CommandString = CommandSting + '&HTTPS=1'
  }
  DataRequest(WebioNo, CommandString);
  maintimer = setTimeout("CommandLoop()", Interval);
}
```

Die Funktion CommandLoop wird über den Parameter *onload* im *body*-Tag gestartet, sobald die Webseite vollständig geladen ist. Die Funktion schickt über die Funktion DataRequest abwechselnd die HTTP-Requests für die Abfrage von Inputs, Outputs und Countern an die Web-IOs und ruft sich mit zeitlicher Verzögerung wieder selbst auf. So werden kontinuierlich im festen Intervall die IO-Zustände aktualisiert.

Schalten der Outputs

```
// Set Output to ON (requested from User)
function setOutput(WebioNo, OutNo, OutVal)
{ var IOPassword = document.getElementById('webio'+WebioNo+'pw').value;
  var CommandString = 'webiohttprequest.php?IP='+WebioIP[WebioNo]+'&PORT='+WebioPort[WebioNo]+'&COMMAND=outputacces
  if(OutVal>0)
  { CommandString = CommandString + '&STATE=0fff';
  }
  else
  { CommandString = CommandString + '&STATE=0';
  }
  if (WebioPort[WebioNo] == '443')
  { CommandString = CommandSting + '&HTTPS=1'
  }
  DataRequest(WebioNo, CommandString);
}
```

Durch Klick auf die entsprechenden Buttons wird die Funktion setOutput aufgerufen und die Nummer des Web-IOs, des Outputs sowie der zu schaltende Zustand übergeben. Über die Funktion DataRequest wird der notwendige HTTP-Request an das Web-IO geschickt.

Löschen der Counter

```
// Set Counter to 0 (requested from display)
function clearCounter(WebioNo, CounterNo)
{ var IOPassword = document.getElementById('webio'+WebioNo+'pw').value;
  var CommandString = 'webiohttprequest.php?IP='+WebioIP[WebioNo]+'&PORT='+WebioPort[WebioNo]+'&COMMAND=counterclea
  if (WebioPort[WebioNo] == '443')
  { CommandString = CommandSting + '&HTTPS=1'
  }
 DataRequest(WebioNo, CommandString);
}
```

Durch Klick auf die entsprechenden Buttons wird die Funktion clearCounter aufgerufen und die Nummer des Web-IOs und des Inputs übergeben.

Aktualisieren der Webseiten Inhalte

```
// Dynamic update of the display depending on IO state
function updateDisplay(WebioNo, ReceiveStr)
{ var HexVal;
  var state;
  var ReceiveData = ReceiveStr.split(';')
  // Display Intput state
  if (ReceiveData[ReceiveData.length - 2].substring(0, 1) == 'i')
  { HexVal = HexToInt(ReceiveData[ReceiveData.length - 1]);
    for (var i = 0; i < MaxI; i++)
    { state = false;
      if ((HexVal & Math.pow(2, i)) == Math.pow(2, i))
      { state = true;
      }
      document.getElementById('webio'+WebioNo+'input'+i).firstChild.data = ( !state ) ? 'OFF' : 'ON';
    }
  }
  // Display Output state
  if (ReceiveData[ReceiveData.length - 2].substring(0, 1) == 'o')
  { HexVal = HexToInt(ReceiveData[ReceiveData.length - 1]);
    for (var i = 0; i < MaxO; i++)
    { state = false;
      if ((HexVal & Math.pow(2, i)) == Math.pow(2, i))
      { state = true;
      }
      document.getElementById('webio'+WebioNo+'output'+i).firstChild.data = ( !state ) ? 'OFF' : 'ON';
    }
  }
  //Display Counter
  if (ReceiveData.length - MaxI - 1 >= 0)
  { if (ReceiveData[ReceiveData.length - MaxI - 1].substring(0, 1) == 'c')
    { for (var i = 0; i < MaxI; i++)
      { document.getElementById('webio'+WebioNo+'counter' + i).innerHTML = ReceiveData[ReceiveData.length - MaxI + i]
      }
    }
  }
  //Display cleared Counter
  if (ReceiveData[ReceiveData.length - 2].substring(0, 1) == 'c')
  { document.getElementById('webio'+WebioNo+'counter'+ ReceiveData[ReceiveData.length - 2].substring(7, ReceiveData[ReceiveDa
  }
}
```

Als Antwort auf einen HTTP-Request sendet das Web-IO immer das eigentliche Kommando und getrennt mit Semikolon einen oder mehrere Parameter. Die Antwort des Web-IO gibt das PHP-Script 1:1 weiter.

Für eine Input Abfrage z.B. input;1

Die Funktion UpdateDisplay zerlegt mittels split den übergebenen String in einen String-Array. Für das Input-Beispiel ergibt sich ein Array mit der ersten Feldvariablen input und der zweiten 1. Über den ersten Buchstaben der ersten Feldvariablen wird festgestellt, ob

die Antwort einen Input, Output oder die Counter betrifft.

Die zweite Feldvariable beinhaltet für Input bzw. Output das Bit-Muster für den Schaltzustand der IOs (Bit0=Input0, Bit1= Input1, ...). Da der Wert in hexadezimaler Schreibweise übergeben wird, muss er zunächst über die Funktion `HexToInt` in eine Dezimalzahl gewandelt werden.

Über eine UND-Verknüpfung des Input-Wertes mit der Zweierpotenz, die dem Bit-Wert der einzelnen Inputs bzw. Outputs entspricht, wird festgestellt, ob der betreffende Input (Output) gleich 0 oder 1 also OFF oder ON ist. Das erfolgt in einer Schleife, die entsprechend der Anzahl der IOs durchlaufen wird.

Mittels der JavaScript-Funktion `document.getElementById` wird das zu aktualisierende Anzeigeobjekt identifiziert und entsprechend dem aktuellen Schaltzustand angepasst.

Die gleichen Mechanismen werden für die Aktualisierung der Counter verwendet, wobei die Umrechnung hexadezimal in dezimal entfällt, da die Zählerstände der Counter dezimal übergeben werden.

Das Beispiel unterstützt die wichtigsten Funktionen des Web-IO, die über HTTP-Requests verfügbar sind, optimiert für das Web-IO Digital 4.0 2xIn, 2xOut. Für die anderen Web-IO-Modelle müssen ggf. Anpassung am Aufbau der Webseite und am JavaScript-Teil vorgenommen werden. Eine detaillierte Beschreibung zur Nutzung von HTTP-Requests für die Web-IO Digital-Modelle finden Sie in der Request-Kommandoübersicht oder im Programmierhandbuch zum Web-IO.

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

W&T
www.WuT.de

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