Visualisierung mit WAGO Controllern auf einem Smartfon

Sie besitzen ein Smartfon, mit dem Sie auch surfen können und möchten zuhause schnell mal das Licht der Einfahrt einschalten. Hier wird beschrieben wie Sie mit JavaScript auf Werte im Controller zugreifen können und auch beliebige Werte zum Controller schicken können.

Das Prinzip

Neben der Web-Visualisierung von CoDeSys, die ein JAVA-Applet erzeugt, gibt es auch die Möglichkeit die Programmiersprache JavaScript einzusetzen. JavaScript ist sehr schnell geladen und wird von allen Pads und Smartfons (egal ob das Betriebssystem von Apple, Google oder Microsoft kommt) ausgeführt. Allerdings gibt es keine fertigen Bausteine mit der man sich die Visu zusammenklickt, sondern Sie müssen „richtig" Programmieren und alles selber machen. Im Gegenzug gibt es dafür keine Grenzen mehr (außer Ihrer Fantasie oder Ihren Fähigkeiten).

Das Handwerkszeug

JavaScript besteht, wie HTML auch, aus einfachen Textdateien. Um die Dateien zu schreiben reicht ein kleiner ASCII-Editor wie Notepad. Aber bequemer wird es mit einem leistungsfähigeren Editor wie z.B. dem kostenlosen Notepad++.
(http://notepad-plus-plus.org/download/v6.1.2.html).
Um Textdateien auf den Controller zu transportieren brauchen Sie einen FTP-Klienten. Das kann der Windows-Explorer sein oder ein spezialisiertes Programm wie z.B. das OpenSource Programm Filezilla. (http://www.filezilla.de/download.htm)
Beide Programme gibt es auch als Portable Applikationen, die ohne Installation auskommen.

Der erste Schritt

Tippen Sie folgende Zeilen in den Editor und speichern sie als „Test01.ssi" ab.
Siehe Abbildung 1.

```
<!-- Ein einfaches Beispiel. Zeigt wie es im Prinzip funktioniert  -->
<html>
<body>
      <H2>Analogwert auf Adresse 0: </H2>
      <H3>Drücken Sie F5 um den Wert zu aktualisieren </H3>
        <!--#READPI ADR=IW0&FORMAT=%d-->
</body>
</html>
```

Die Dateiendung ssi (für server side includes) braucht der Webserver auf dem Controller damit er eine statische Anweisung durch lebende Werte ersetzt. Dazu muss der Lesebefehl im body-Bereich der Datei stehen und das **#READPI ADR=IW0&FORMAT=%d** in Kommentarzeichen (das „<!—„ markiert den Beginn und das „-->" markiert das Ende des Kommentars) eingehüllt werden.

<H2> (oder <h2> Gross- und Kleinschrift wird nicht unterschieden) bedeutet Überschrift 2. Ordnung (englisch „heading"). Eine schöne Übersicht über alle HTML Auszeichnungen gibt es hier:
http://de.selfhtml.org/ oder auch bei Wikipedia: http://de.wikipedia.org/wiki/Html

READPI steht für read process image. Die Adresse (hier **ADR=IW0**) des zu messenden Wertes lässt sich mit der CoDeSys in der Karteikarte „Ressourcen" im Punkt Steuerungskonfiguration ermitteln.
Siehe auch Abbildung 2.
FORMAT=%d bestimmt, das der Wert dezimal angezeigt wird (**=%x** bedeutet he**x**adezimale Anzeige)

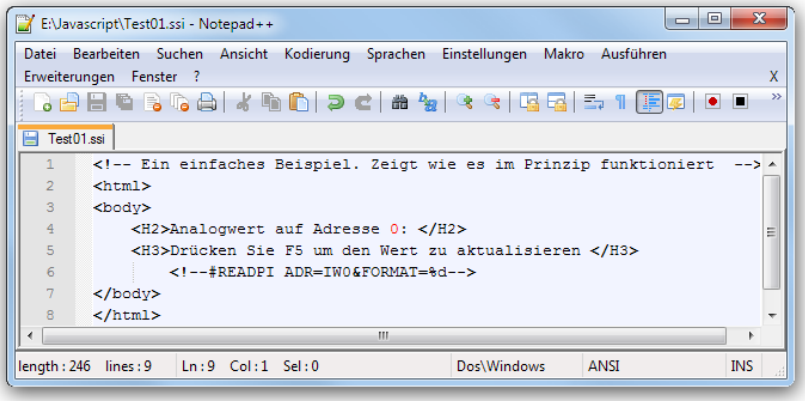


Abbildung 1 Das erste Beispiel im Editor Notepad++

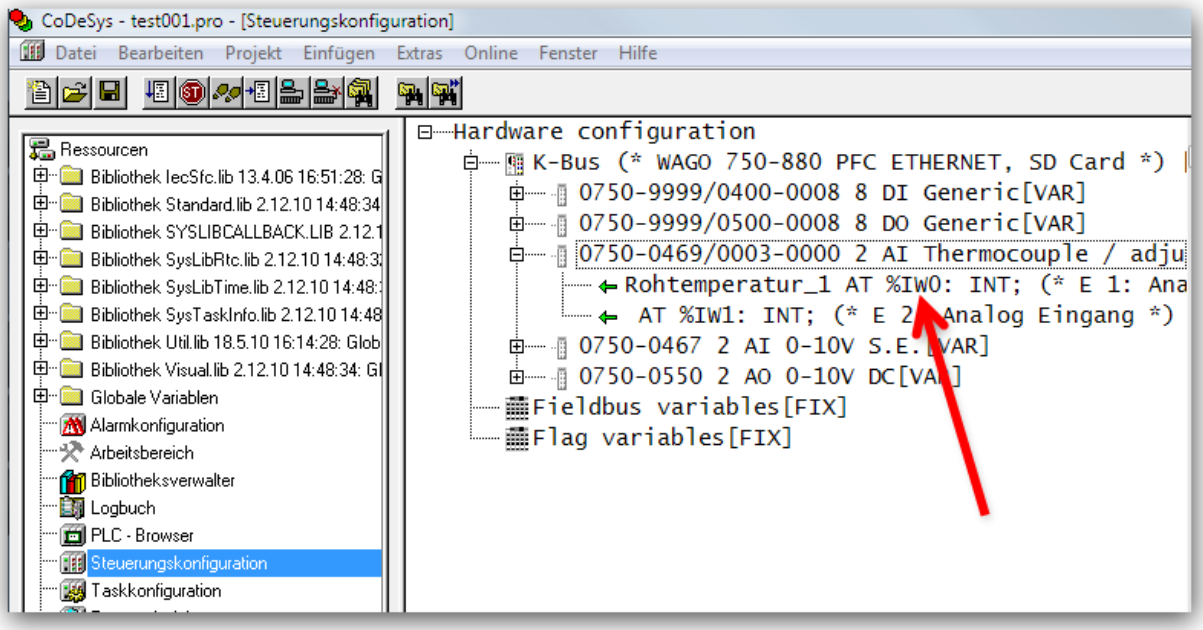


Abbildung 2 Die Adressen lassen sich in der Steuerungskonfiguration ermitteln

Diese Datei muss nun auf den Controller kopiert werden. In Abbildung 4 wurde die Datei mit Filezilla übertragen. Es bietet sich das Verzeichnis „webserv“ an, da es standardmäßig aufgerufen wird und man nur einen Teil des Namens ändern muss, wenn man sich das Ergebnis im Browser anschauen möchte. Tippen Sie nun in Ihrem Lieblingsbrowser dort die IP-Adresse des Controllers ein und ändern Sie den Eintrag index durch test01.
Der angezeigte Wert sieht so aus wie in Abbildung 3 und kommt in diesem Beispiel von einer Temperaturklemme, muss also noch durch 10 geteilt werden um die Raumtemperatur (hier genau 20.0 °C) zu erhalten.
Da HTML nur statische Seiten anzeigt, müssen Sie auf einem PC die Funktionstaste F5 (für aktualisieren) drücken, um neue Werte zu erhalten. In JavaScript geht das auch automatisch.

Beim ersten Einloggen auf die frisch erstellte Seite werden Sie nach Name und Passwort gefragt.
In einem späteren Kapitel machen wir es uns bequemer.



Abbildung 3 Der erste Test im Browser

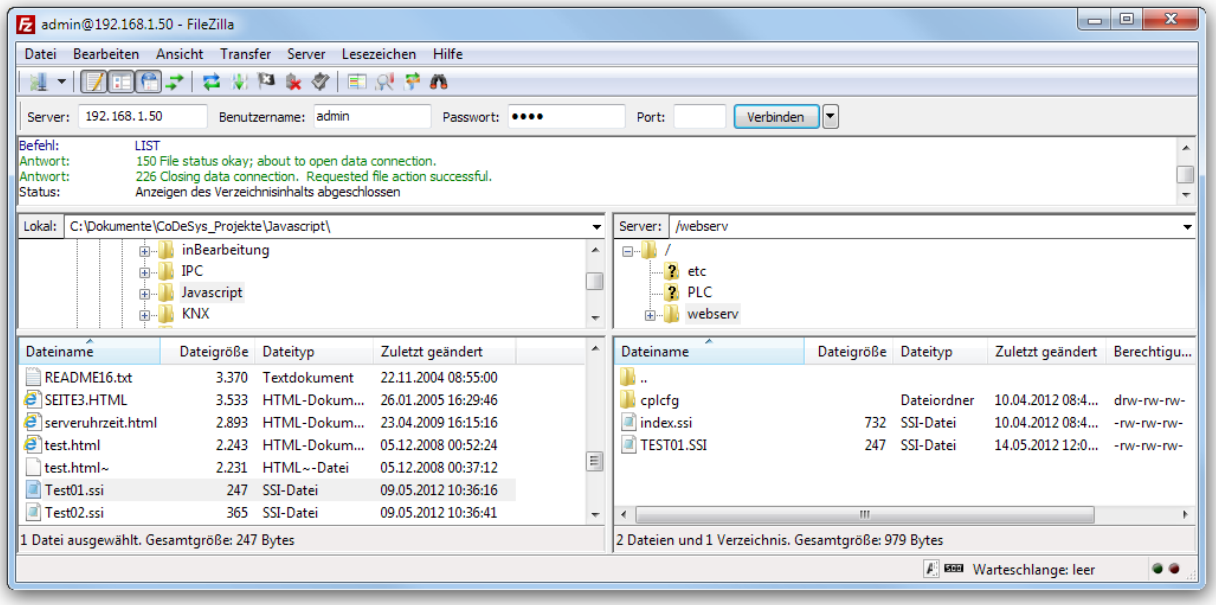


Abbildung 4 Filezilla kann Dateien auch per Doppelklick kopieren

Ein Analogwert und ein Digitalwert

Es lassen sich auf diese Weise auch digitale Werte messen. Die folgende Datei habe ich Test02.ssi genannt. Kopieren Sie die Datei auch auf den Controller:

```
<!-- Wieder ein einfaches Beispiel. Aber mit einem Analogwert und einem
Digitalwert -->
<html>
<body>
      <H2>Analogwert des ersten Einganges: </H2>
      <H3>Drücken Sie F5 um den Wert zu aktualisieren </H3>
      <p><!--#READPI ADR=IW0&FORMAT=%d--></p>
      <H2>Digitalwert des ersten Schalters: </H2>
      <p><!--#READPI ADR=IX4.0&FORMAT=%d--></p>
</body>
</html>
```

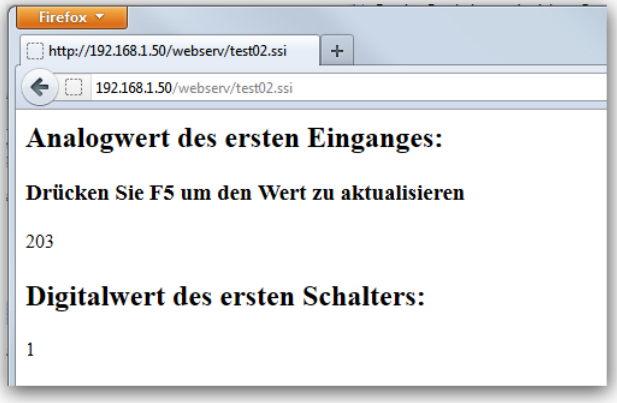


Abbildung 5 Test002 im Browser

Eine kleine Variation

Die angezeigten Werte müssen nicht zwangsweise in eine neue Zeile. Das Auslesen kann an beliebiger Stelle erfolgen.

```
<!-- Fast wie Test02, aber diesmal mit zwei Digitalwerten -->
<html>
<body>
      <h2> Drücken Sie F5 um den Wert zu aktualisieren </h2>
      <h3> Analogwert des ersten Einganges:</h3>
      <p><!--#READPI ADR=IW0&FORMAT=%d--></p>
      <h3> Digitalwert des ersten Schalters: </h3>
     <p><!--#READPI ADR=IX4.0&FORMAT=%d--></p>
      <h3>Schalter 2:  <!--#READPI ADR=IX4.1&FORMAT=%d-->   </h3>
</body>
</html>
```

Jetzt kommt JavaScript

Reines HTML hat den Nachteil, dass man nur statische Inhalte darstellen und nichts berechnen kann. Hier springt JavaScript in die Bresche.
Im Text wird der JavaScript-Teil durch **„<script language="JavaScript">“** und am Ende durch **„</script>“** eingerahmt.

```
<!-- Das Beispiel aus Test02, aber diesmal mit Javascript realisiert -->
<html>
<body>
 <h1>  Clientseitiges Java-Scripting     </h1>
 <script language="JavaScript">
    var Analogwert
    var Binaerwert
      Analogwert=<!--#READPI ADR=IW0&FORMAT=%d-->/10.0;
      Binaerwert=<!--#READPI ADR=IX4.0&FORMAT=%d-->;
      document.write("<h2>Analogwert des ersten Einganges:  <h2>");
      document.write("<p>Analogwert:  " + Analogwert + " °C </p>");
      document.write("<h2>Digitalwert des ersten Schalters: </h2>");
      document.write("<p>Binärwert:  " + Binaerwert + "</p>");
 </script>
</body>
</html>
```

Will man innerhalb von JavaScript wieder HTML Elemente ausgeben, benutzt man die Funktion document.write(““).



Abbildung 6 So zeigt der Browser es an

Automatisches Aktualisieren

Es ist möglich das sich die Seite regelmäßig selber aktualisiert:

```
<!-- Verbesserung zu HTML: automatisches Aktualisieren der Seite -->
<html>
<head>
<!-- Die Zahl hinter content gibt das Aktualisierungsintervall in Sekunden an
-->
<meta http-equiv="refresh" content="2">
</head>
<body>
 <h2>  Clientseitiges Java-Scripting  </h2>
  <script language="JavaScript">
    var Analogwert
    var Binaerwert
    Analogwert=<!--#READPI ADR=IW0&FORMAT=%d-->/10.0;
    Binaerwert=<!--#READPI ADR=IX4.0&FORMAT=%x-->;
    document.write("<p>Analogwert:  " + Analogwert + " °C</p>");
    document.write("<p>Binärwert:  "  + Binaerwert + "</p>");
 </script
</body>
</html>
```

Das Aktualisieren des Bildschirmes lässt sich mit einem Druck auf die Esc-Taste unterbrechen.

Werte zur SPS schicken

Bisher wurden nur Werte aus dem Controller abgeholt, jetzt wollen wir ein Bit in Richtung SPS schicken. Dazu brauchen wir eine Variable, in das wir schreiben dürfen. Das kann, wie hier, eine Feldbusvariable sein. D.h. Sie müssen im CoDeSys eine Empfangsvariable anlegen. (Es geht auch über Merkerworte %MW100 usw. aber das hebe ich mir noch auf ;-)

Das muss in der Karteikarte Ressourcen gemacht werden:

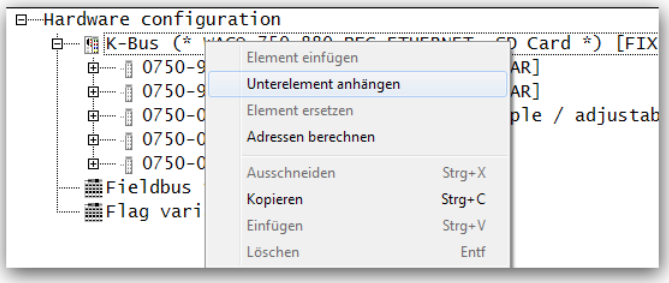


Abbildung 7 Kartei Ressourcen -> Steuerungskonfiguration Rechte Maus über dem K-Bus

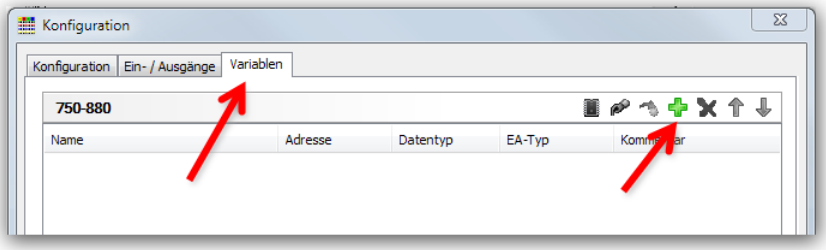


Abbildung 8 Dann in die Variablen und auf das "+" klicken

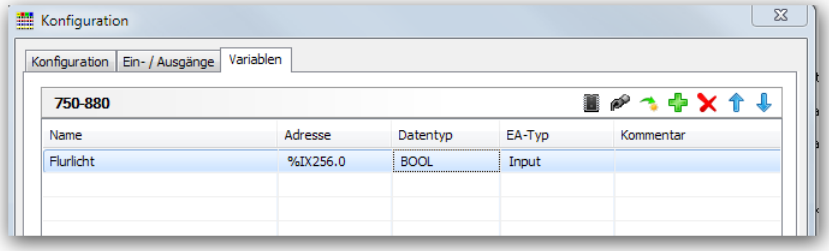


Abbildung 9 Über eine Dropdown-Box wählen Sie den Datentyp BOOL aus

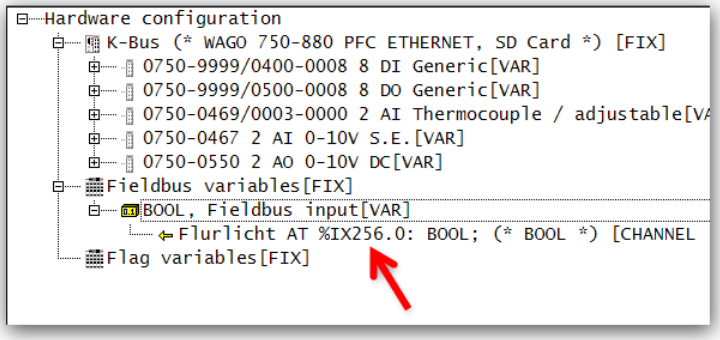


Abbildung 10 Wenn alles fertig ist, sieht es so aus (der Pfeil zeigt auf die Adresse)

Die neu erzeugte Variable ist aus der Sicht des Controllers ein Eingang. Die Hardware-Eingänge werden von den Klemmen beschrieben. Die virtuellen Eingänge in unserem Fall von einem Smartfon mit Hilfe von JavaScript.
Neben der Funktion **READPI** gibt es noch **WRITEPI**, die max 24 Datensätze pro Aufruf schreiben kann. Ein Datensatz besteht aus den Parametern: ADR(i), VALUE(i) und FORMAT(i), wobei i für den Index von 1 bis 24 steht. In folgendem Beispiel wird nur ein Datensatz, je nach geklicktem Button, geschrieben, dh. wird brauchen nur ADR1. Übertragen werden die Datensätze in einem Formular mit der Methode POST. Das Zusammenbauen des **WRITEPI** Befehls wurde in der selbstdefinierten "**WriteData**" Funktion realisiert

```
<!-- Beispiel einer digitalen Eingabe über zwei Schaltflächen -->
<html>
<body>
 <h2>  Test einer digitalen Eingabe  </h2>
<script language="JavaScript">
  function WriteData(MyAddress, MyValue)
{
   var format1 = "";
   document.write("<form action=\"/WRITEPI\" method=\"POST\" name=\"MySubmitForm\">");
   document.write("<input type=\"hidden\" name=\"ADR1\" value=\"" + MyAddress + "\"></input>");
   document.write("<input type=\"hidden\" name=\"VALUE1\" value=\"" + MyValue + "\"></input>");
   document.write("<input type=\"hidden\" name=\"FORMAT1\" value=\"%d" + "\"></input>");
   document.write("</form>");
   window.setTimeout("document.MySubmitForm.submit()", 1);
}
</script>
  <form name="FlurlichtForm">
       <input type=button value="Flurlicht EIN" onClick="WriteData('IX256.0', 1)"></input>
       <input type=button value="Flurlicht AUS" onClick="WriteData('IX256.0', 0)"></input>
  </form>
</body>
</html>
```



Abbildung 11 Die Eingabebuttons im Browser

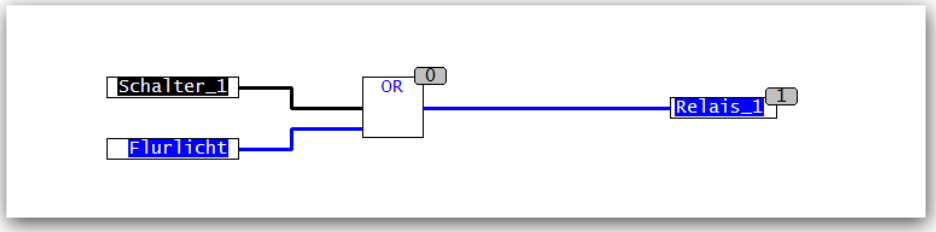


Abbildung 12 Die Feldbusvariablen können normal in CoDeSys verwendet werden

Einige Smartfones haben eine hohe Auflösung und die Pixel sind dementsprechend klein. Damit man einen Button besser mit dem Finger trifft lässt sich vorschreiben welche Größe er haben soll.
Wenn man hinter type=button folgendes einfügt **`size="30" style="font-family: Verdana; font-size: 22px"`** ergibt sich folgender Text:

```
form name="FlurlichtForm">
       <input type=button size="30" style="font-family: Verdana; font-size: 22px" value="Flurlicht
EIN" onClick="WriteData('IX256.0', 1)"></input>

       <input type=button size="30" style="font-family: Verdana; font-size: 22px" value="Flurlicht
AUS" onClick="WriteData('IX256.0', 0)"></input>
</form>
```



Abbildung 13 Die Buttons dürfen auch größer ausfallen

Mit Rückmeldung ob der Wert angekommen ist

```
<html>
<head>
<title>Beispiel mit Rueckmeldung</title>
</head>
<body>
 <H2>  Test einer digitalen Eingabe mit Rückmeldung </H2>
<script language="JavaScript">
function WriteData(MyAddress, MyValue)
{
   var format1 = "";
   document.write("<form action=\"/WRITEPI\" method=\"POST\" name=\"MySubmitForm\">");
   document.write("<input type=\"hidden\" name=\"ADR1\" value=\"" + MyAddress + "\"></input>");
   document.write("<input type=\"hidden\" name=\"VALUE1\" value=\"" + MyValue + "\"></input>");
   document.write("<input type=\"hidden\" name=\"FORMAT1\" value=\"%d" + "\"></input>");
   document.write("</form>");
   window.setTimeout("document.MySubmitForm.submit()", 1);
}
</script>
<form name="FlurlichtForm">
  <input type=button style="font-family: Verdana; font-size: 22px;" value="Flurlicht EIN"
onClick="WriteData('IX256.0', 1)"></input>
  <input type=button style="font-family: Verdana; font-size: 22px;" value="Flurlicht AUS"
onClick="WriteData('IX256.0', 0)"></input>
</form>

<script language="JavaScript">
  /* anzeigen lassen, ob das Auslesen geklappt hat  */
   if (<!--#READPI ADR=IX256.0&FORMAT=%x--> == 1)
       {document.write("Flurlicht ist EIN  ");}
      else
       {document.write("Flurlicht ist AUS  ");}
</script>
</body>
</html>
```



Abbildung 14 Eine einfache Statusrückmeldung

Mit Rückmeldung, die den Button einfärbt

Das vorherige Beispiel hat je eine Schaltfläche für den Zustand (hier EIN und AUS) verwendet.
Es lässt sich auch (wie bei einem Stromstossschalter) nur mit einem Button schalten. Um diesen Button je nach Schaltzustand einzufärben braucht man eine if-Abfrage in JavaScript.
Das JavaScript erzeugt dann je nach gelesener Rückmeldung
(hier direkt mit **`if (<!--#READPI ADR=IX256.0&FORMAT=%d--> == 1)`** eingelesen) entweder einen dunklen oder hellen Button.
JavaScript selbst kennt keinen Button, HTML schon. Daher muss man mit JavaScript folgenden HTML-Code erzeugen:

```
<form name="FlurlichtForm">
       <input type=button value="Flurlicht EIN" onClick="WriteData('IX256.0', 1)"></input>
       <input type=button value="Flurlicht AUS" onClick="WriteData('IX256.0', 0)"></input>
</form>
```

Das geht mit der schon bekannten Funktion **`document.write("HTML-Text").`** Unglücklicherweise ist der auszugebende Code selber von Anführungszeichen umrahmt. Wenn man also

```
document.write("<form name="FlurlichtForm">")
```

eintippen würde, wäre nach **`name=`** Schluss, da das Ende-Anführungszeichen gelesen wird. Möchte man nun Anführungszeichen in den auszugebenden HTML-Code integrieren muss man ein Backslash \ vor jedes Anführungszeichen setzen. Damit ergibt sich für die erste Zeile:

```
document.write("<form name=\"FlurlichtForm\">")
```

Achtung: Kommentare innerhalb des JavaScript Blockes werden mit /* */ umrahmt, statt mit <!-- --> bei HTML.
Es lassen sich innerhalb des bodys auch mehrere JavaScript Bereiche definieren.

Auf der nächsten Seite sehen Sie den kompletten Code und wie es der Browser anzeigt.

```
<!-- Nun sollen die Buttons je nach Status eingefärbt werden -->
<html>
<head>
<title>Beispiel dynamische Buttons</title>
</head>
<body>
<script language="JavaScript">
function WriteData(MyAddress, MyValue)
{
   var format1 = "";
   document.write("<form action=\"/WRITEPI\" method=\"POST\" name=\"MySubmitForm\">");
   document.write("<input type=\"hidden\" name=\"ADR1\" value=\"" + MyAddress + "\"></input>");
   document.write("<input type=\"hidden\" name=\"VALUE1\" value=\"" + MyValue + "\"></input>");
   document.write("<input type=\"hidden\" name=\"FORMAT1\" value=\"%d" + "\"></input>");
   document.write("</form>");
   window.setTimeout("document.MySubmitForm.submit()", 1);
}
</script>
<!--  Zwischen den script-tags muss irgendwas html-mäßiges stehen.  -->
 <H2>  Test einer digitalen Eingabe mit Rückmeldung </H2>
<script language="JavaScript">
       /* Ruft den Button auf und beschriftet ihn je nach Zustand  */
 if (<!--#READPI ADR=IX256.0&FORMAT=%d--> == 1)
       {       document.write("<form name=\"FlurlichtForm\">");
               document.write("<input type=button style=\"font-size: 28px; background-color:
#ffffff; color: #000; \" value=\"Flurlicht ist EIN\" onClick=\"WriteData('IX256.0',
0)\"></input>");
               document.write("</form>");
       }
       else
       {       document.write("<form name=\"FlurlichtForm\">");
               document.write("<input type=button style=\"font-size: 28px; background-color:
#000000; color: #F0f0f0; \" value=\"Flurlicht ist AUS\" onClick=\"WriteData('IX256.0',
1)\"></input>");
               document.write("</form>");
       }
</script
</body>
</html>
```

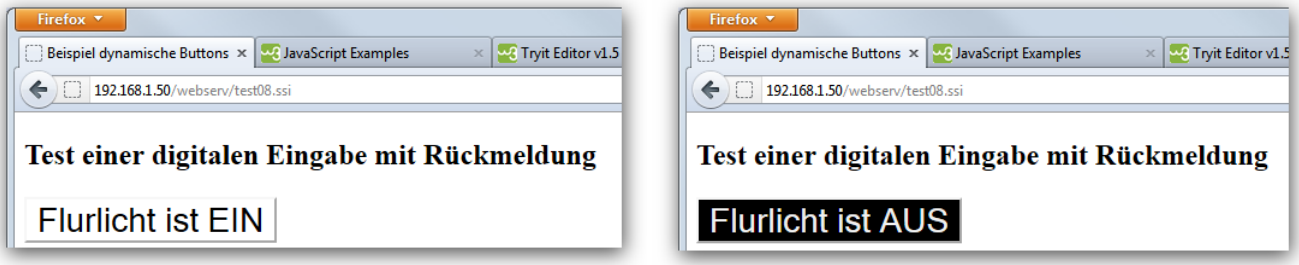


Abbildung 15 Die Farbe des Buttons zeigt seinen Zustand an

Befehlsvorrat:

Uhrzeit des Controllers lessen:

<!--#GETTIMEDATE FORMAT=%d.%m.%y %H:%M:%S -->

Optimiertes Auslesen mehrerer Bits
Liest ein Word am Stück aus und separiert die Bits in einer FOR Schleife mit JavaScript

```
<script language="JavaScript">
     var WordData
     var Rest
     var i

      WordData=<!--#READPI ADR=IW0&FORMAT=%d-->;
      document.write("<BR> <BR>");
      document.write("Zustand des Eingangswordes: " + WordData);
      document.write("<BR> <BR>");
      document.write("Bit      Value");
      document.write("<BR>");

      for (i=1; i<=16; ++i)
      {     Rest = WordData % (Math.pow(2, i));
            if (Rest > 0)
          { document.write("  " + i + "   ON" );
          }
           else
          { document.write("  " + i + "   OFF");
          }
         WordData -= Rest;
         document.write("<BR>");
      }
  </script
```