Application Note

Anbindung von Kunden-Software an SpiderControl Web Visualisierung

Version

Version	Datum	Kommentar	Autor
0.1		Entwurf	br

Inhalt

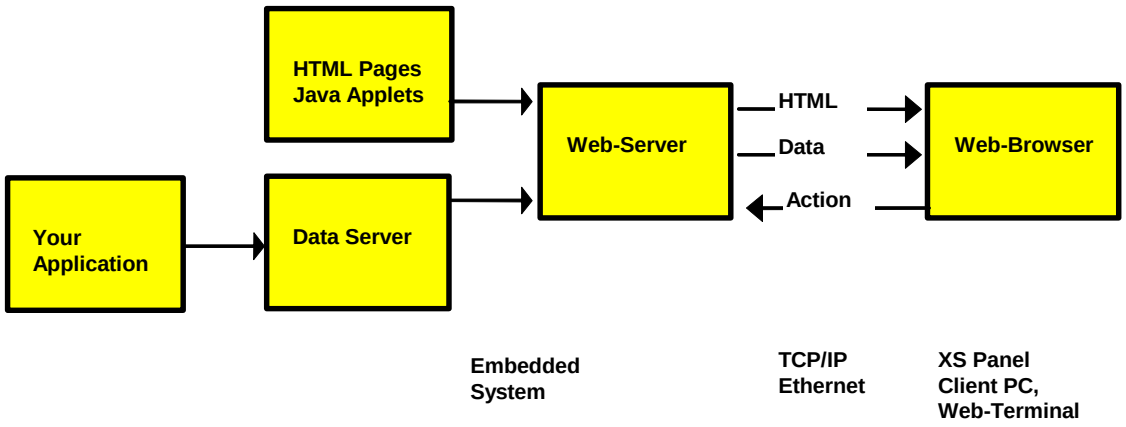
1. Grundsätzlicher Aufbau

Der MicroBrowser der XS-Serie ist in der Lage, Web-HMI's darzustellen, welche entweder mit dem SpiderControl HMI Editor, einem dazu kompatiblen OEM Produkt oder der CoDeSys Webvisu erstellt wurden. Alle diese Webvisualisierungen können normalerweise in einem Standardbrowser (z.B. als Java Applet) angezeigt werden. Der MicroBrowser ist eine in C implementierte Runtime, welche spezifisch diese Projekte auf demselben Weg anzeigen kann.

Der SpiderControl HMI Editor ist ein rein grafisches Entwicklungswerkzeug, mit welchem man auf einfachste Weise interaktive Bedieneroberflächen zeichnen kann. Es stehen umfangreiche Funktioselemente wie Text-, Editfelder, DropDown Listen, Buttons, Grafikelemente (Raster und Vektorgrafik) sowie viele weitere Objekte zur Verfügung, welche auch das Design von komplexen Bedienerseiten ermöglichen.

Dieser Editor erstellt eine HTML Seite sowie alle dazugehörigen Projektfiles, welche auf das Root-Verzeichnis des Webservers kopiert werden sollen. Dieser Webserver befindet sich in dem hier beschriebenen Fall auf dem PC, wo auch die SW des Kunden laufen soll. Auf der XS-Serie muss nun die IP Adresse des PC's sowie der Name der erzeugten HTML Seite angegeben werden, sodass sich dieses die Webvisualisierung laden und die Datenkommunikation mit dem Webserver aufnehmen kann. Die Datenkommunikation erfolgt dabei im Polling Betreib, sodass der MicroBrowser automatisch immer den letzten Stand der Variablen anzeigen kann.

Der Webserver ist dazu mit einem Datenserver verbunden, welcher als API über eine Funktion zum Lesen- bzw. Schreiben der Variablen verfügt. Die Variablen werden an dieser Schnittstelle über ihren symbolischen Variablennamen identifiziert, welcher auch bei der Projektierung des HMI's mit dem Grafikeditor verwendet wurde. Die ganze Datensynchronisation mit dem Webserver und den Web-Clients ist durch diese Komponenten bereits gelöst.



Um eine Verbindung mit der eigenen Software des Kunden zu bekommen, gibt es nun verschiedene Möglichkeiten.

2. Zugriff über URL Klasse

Es kann ein bestehender DataServer verwendet werden, welcher jede Variable, welche zum Lesen oder Schreiben angefragt wird, automatisch in einer internen verketteten Liste angelegt und mit dem Wert ‚0' inintialisiert wird. Diese Variablen sind für alle Clients als globale Variablen über Web-Services sichtbar. Der Kunde kann nun aus seiner Applikation heraus per http einen Webservice zum lesen/schreiben aufrufen, um so Daten mit dem MicroBrowser auszutauschen. Die Webservices heissen

RedaVal

WriteVal

The following commands are sent using the HTTP GET request:
http://host/dir/cgi-bin/readVal.exe?ppoVarName
Example reply (HTTP header not included):
92

http://host/dir/cgi-bin/writeVal.exe?ppoVarName+Value
These two commands are used to read or write directly a ppo variable.

In den meisten Entwicklungsumgebungen gibt es dafür vorgesehene Klassen, welche auf einfachste Weise so ein Kommando absetzen können.

Dazu sind auf Anfrage Sourcecodebeispiele in Java oder C# verfügbar.

SpiderLink DLL

Es auch eine DLL, welche diese Funktionen in einer C-Notation anbieten und welche in Kundenprojekte eingebunden werden kann.

3. DataServer DLL

Eine weitere Integrationsmöglichkeit besteht darin, den DataServer als DLL in eine Kundenapplikation zu integrieren. Dieselbe DLL wird somit durch den Webserver als auch durch die Kundenapplikation verwendet. Der Kunde kann die Funktionen zul Lesen und Schreiben der Variablen auscodieren und von dort eigene Funktionen aufrufen. Er erhält den Variablennamen als String und muss durch einen geeigneten Compare die von ihme gewünschte Variable kopieren oder eine Funktion aufrufen. Der Vorteil dieser Variante gegenüber der ersten besteht primär darin, dass eine Interaktion (wie z.B. drücken einer Taste oder Eingabe eines Wertes) auf der XS-Serie direkt auf eine eigene Funktion abgebildet werden kann, während bei der erstgenannten Variante eine solche Interaktion durch Lesen der entsprechenden Variablen gepollt werden müsste.

Diese DLL kann als VisualStudio Projekt im Source Code zur Verfügung gestellt werden. Die DLL ist in C/C++ implementiert.

Das Grundprinzip für den Aufbau des DataServers ist in den folgenden beiden Funktionen ersichtlich:

```
void DS_readVal(char *varName, char *value)
    {
    if (strcmp(varName, "version") == 0)    sprintf(value, "%i" , version);
    else if (strcmp(varName, "var1") == 0) sprintf(value, "%f" , var1);
    else if (strcmp(varName, "text_1") == 0) sprintf(value, "%s" , text_1);
    else sprintf(value, "Variable %s not found in Database.\n", varName);
    }

void DS_writeVal(char *varName, char *value)
    {
    if (strcmp(varName, "version") == 0)    sscanf(value, "%i" , &version);
    else if (strcmp(varName, "var1") == 0) sscanf(value, "%f" , &var1);
    else if (strcmp(varName, "text_1") == 0) sscanf(value, "%s" , text_1);
    else sprintf(value, "Variable %s not found in Database.\n", varName);
    }
```

4. Integration des Webservers in die Kundenapplikation

Als dritte Variante kann der gesamte Webserver im Sourcecode in ein Projekt des Kunden integriert werden. Die Schnittstelle für die Variablen ist dabei dieselbe (DataServer) wie in Variante 2. Der Vorteil der dritten Lösung liegt darin, dass alle Komponenten in die Kundenapplkation integriert sind und somit eine monolithische, einfach zu verteilende Lösung entsteht.

Der Webserver kann als VisualStudio Projekt im Source Code zur Verfügung gestellt werden. Der Webserver ist in C/C++ implementiert.