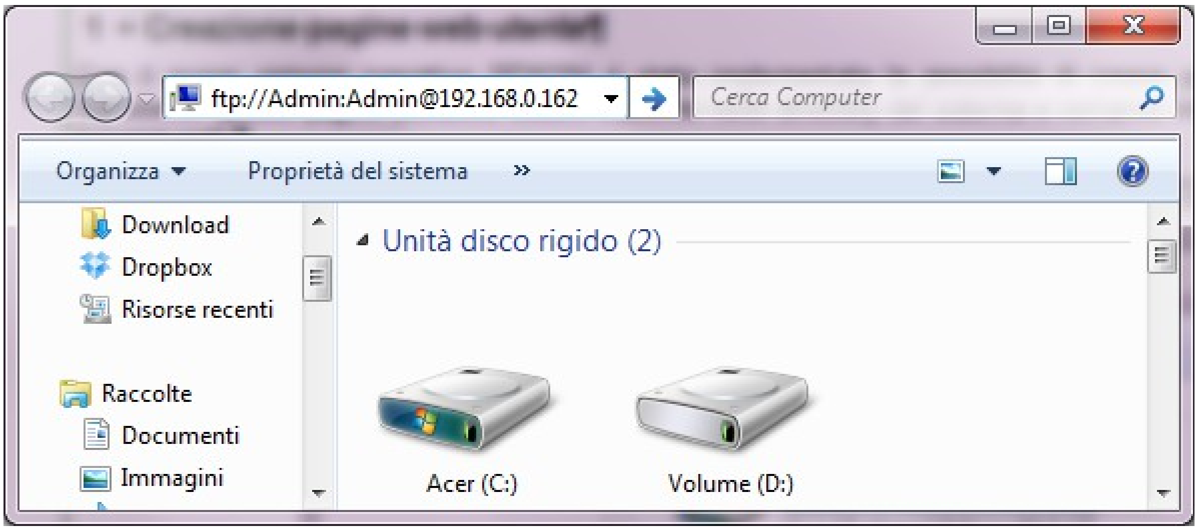
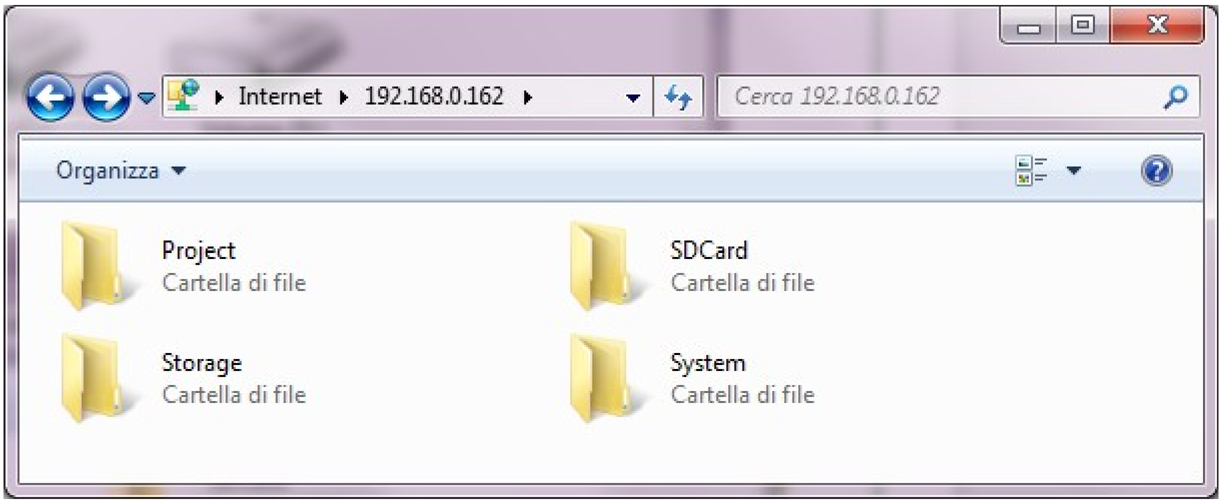
1 How to create user web pages

With the new SFW184 operating system has been implemented the possibility to create web pages directly by the user, these pages can be transferred in the system directory and will be displayed and accessed from web browsers.

To transfer the web pages created by the user in the SlimLine file system system, must be used an FTP client (Example FileZillla) but you can also simple use the Windows Explorer. as shown in the figure below, setting the credentials in the address bar and the IP address ***ftp://Admin:Admin@192.168.0.162***, you can connect and see the file system.



Here's the view of the file system at the connection. ***Project*** and ***System*** folders are reserved for system use and you should not change its contents. The files of the user pages can be transferred in the ***Storage*** and ***SDCard*** (if present) folders.



So the user can create his own web pages using any HTML editor but also simply by using a simple text editor such as Notepad, of course it's needed to know the HTML syntax. The created pages will be transferred in the desired directory and accessing to them by a browser the page will be displayed.

1.1 Web pages criteria

Of course, the SlimLine web server the has only a limited set of functions, and thus the creation of web pages must abide certain rules, let's see:

a) The page can not contain inclusion of other pages (Example pages of style or scripts).

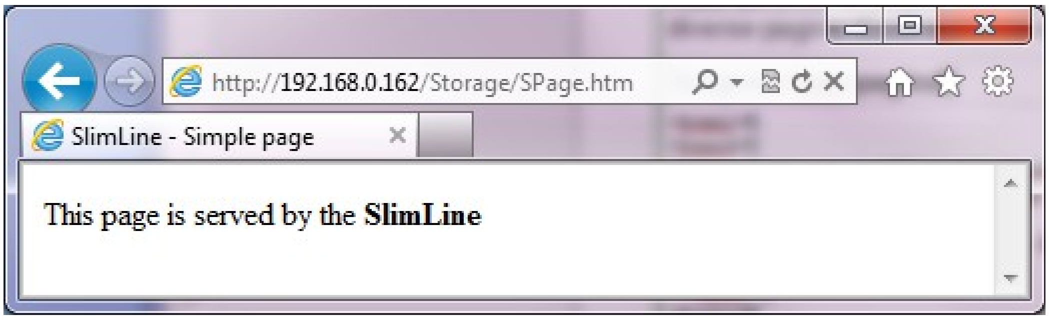
b) The page can not contain inclusion of images (jpg or gif file example), any images can be embedded in the page itself.

We see a simple page that displays a presentation message.

Html source page

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>SlimLine - Simple page</title>
</head>
<body>
This page is served by the <b>SlimLine</b>
</body>
</html>
```

Saving the above text in a file, such as ***SPage.htm***, and transferring it to the ***Storage*** directory of the SlimLine, you can see the resulting web page by simply typing in the browser the page address.



This page is served by the **SlimLine**

Of course, the page can contain links to other pages, it will be possible to achieve a personal navigation between different pages. Here is the same example as before with included the definition of a style.

Html source page

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>SlimLine - Simple page</title>
<style type="text/css">
.Bolded {font-family: Arial, Helvetica, sans-serif; font-size: 20px; font-style: normal;font-weight: bold;}
</style>
</head>
<body>
This page is served by the <span class="Bolded">SlimLine</span>
</body>
</html>
```

1.2 Dinamic web pages

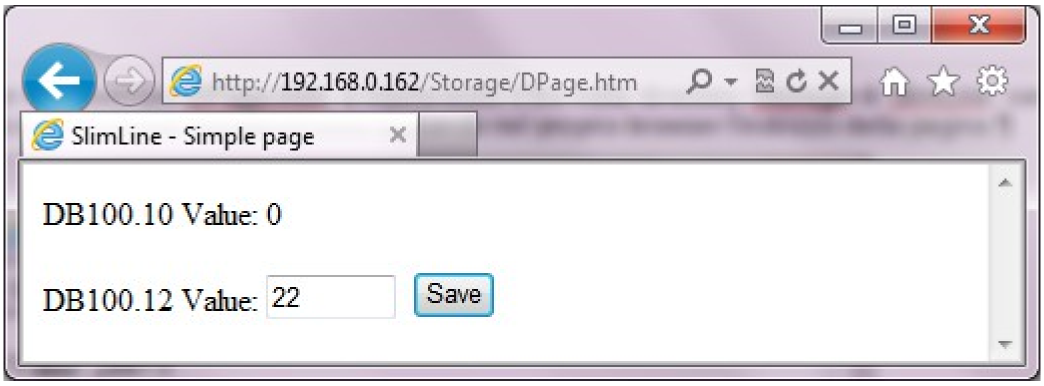
The most important feature of the SlimLine integrated web server is the ability to handle dynamic pages. A dynamic page is a page whose content, in whole or in part, is generated by the server on demand, and therefore can be different each time it is invoked, thus allowing interactivity with the user.

Hence it will be possible to create a page that lists the values of PLC variables and allow you to change the value of them. The following example report an html source of a simple page that displays the value of a PLC variable of type UINT allocated at DB100.10 and allows you to set the value of a PLC variable of type UINT allocated at DB100.12.

Html source page

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>SlimLine - Simple page</title>
</head>
<body>
DB100.10:&nbsp;<!--["%d", UINT, 10]--></br>
<form id="MyForm" name="MyForm" method="post" action="DPage.htm">
DB100.12:&nbsp;<input name="UINT 12" type="text" size="5" maxlength="10" value”<!--["%d", UINT, 12]-->”>
&nbsp;<input name="MyButton" type="submit" id="MyButton" value="Save"/>
</form>
</body>
</html>
```

Saving the above text in a file, such as ***DPage.htm***, and transferring it to the ***Storage*** directory of the SlimLine, you can see the resulting web page by simply typing in the browser the page address.



DB100.10 Value: 0

DB100.12 Value: 22 Save

As can be seen, the top row shows the value of the PLC DB100.10 while setting a value in the text box in the bottom row and then pressing the ***Save*** button, you can set the value of the DB100.12.

Of course in a web page can be displayed and can be set all the desired variables, it is advisable not to go overboard with the number of variables, it is preferable to split them into multiple pages.

1.3 TAGs format

As we have seen in a dynamic page the content is generated on the fly by Http server (SlimLine CPU module), we see what are the mechanisms for defining ***TAGs*** to be displayed. Within the HTML source of the page you can define a comment fields like <!--["%d", UINT, 10]-->.

The fields are interpreted as comments and therefore can be managed from any HTML editor (Example Macromedia), but the Http server when sending the page to the client (The browser that displays it) replace the field with the value of the variable. In TAG are contained all the necessary information according to the syntax

<!--[Format, Type, Address]-->

1.3.1 Format field

The ***Format*** string can contain elements of two types, the first consists of characters that are returned to the page unaffected. The second consists in the conversion directives that describe the way in which the topics are to be displayed. The directives conversion begin with a % sign followed by the directives in the format:

% [Flags] [Width] [.Precision] [Length] Conversion

Flags

+	The visualization of signed variables, will always start with the - or + signs.
space	The visualization of signed variables, will always start with the space or – sign.
x	Values other than 0 are prefixed with 0x.
0	At the displayed value are added 0 until the desired number of digits (Only for variables of type d, i, o, u, x, X, e, E, f, g, G).

Width: Defines the number of digits to be displayed.

Precision: Defines the number of decimal places to be displayed (Only for variables of type e, E, f).

Length

h	Prima di (d, i, u, x, X, o) denota una variabile short int o unsigned short int.
l (elle)	Prima di (d, i, u, x, X, o) denota una variabile long int o unsigned long int.
L	Prima di (e, E, f, g, G) denota una variabile long double.

Conversion

d	Decimal value with sign.
i	Decimal value with sign.
o	Octal value without sign.
u	Decimal value without sign.
x	Hexadecimal value is displayed using lowercase letters (From 0 to 9, from a to f).
X	Hexadecimal value is displayed using uppercase letters (From 0 to 9, from A to F).
e	Decimal floating-point value, displayed on exponential notation (Example: [-]d.ddde+dd).
E	Decimal floating-point value, displayed on exponential notation (Example: [-]d.dddE+dd).
f	Decimal floating-point value (Example: [-]d.ddd).
c	Single character.
s	String.

1.3.2 Type field

The ***Type*** field indicates the type of variable you want to display, are managed all types defined in IEC61131.

1.3.3 Address field

The ***Address*** field indicates the address of the variable, remember that you can specify only variables allocated in the DB 100.

1.3.4 Some TAGs example

To better understand the display format of the TAGs here some examples.

<!--["%d", UINT, 10]-->	Displays the value of a UINT variable allocated at DB 100.10 with a number of integer digits based on the variable value.
<!--["%04d", UINT, 10]-->	Displays the value of a UINT variable allocated at DB 100.10 always expressed with 4 digits.
<!--["%3.0f", REAL, 32]-->	Displays the value of a REAL variable allocated at DB 100.32 with 3 integers digits and no decimal.
<!--["%4.2f", REAL, 50]-->	Displays the value of a REAL variable allocated at DB 100.50 with 2 integers digits and 2 decimals.

1.4 ARGs format

The user, other than request to the web server a page can also specify certain parameters to be sent to the web server. To set a PLC variable value from a web page a POST request will be managed, a module is inserted in the web page. When an HTML page contains a ***<form>***, the data set on the various objects in the ***<form>*** are sent so as not to be directly visible to the user, through the HTTP request that the browser sends to the server.

If we refer to the previous example we can see that the part of the HTML page which allows the setting of the PLC variable UINT allocated at DB 100.12 is as follows..

Html source page

```
<form id="MyForm" name="MyForm" method="post" action="DPage.htm">
DB100.12:&nbsp;<input name="UINT 12" type="text" size="5" maxlength="10" value”<!--["%d", UINT, 12]-->”>
&nbsp;<input name="MyButton" type="submit" id="MyButton" value="Save"/>
</form>
```

Pratically a ***<form>*** field with id ***MyForm*** contains a text box of 5 characters with a maximum of 10 characters with id UINT12. In the form is also placed a button of submit type which pressure sends the whole form.

By defining in the browser the value of the text box and pressing the ***Save*** button, the defined data will be sent to the server that will display the page ***DPage.htm*** and at the same time it will write the defined value in the UINT DB100.12 variable.

TARG name

The ***name*** field of the argument is very important, it defines the type of PLC variable to be set (All types defined in IEC61131 are managed) and its address, the two fields must be separated by a space.

A name like UINT 12 will indicate a UINT variable allocated to address DB 100.12.
A name like REAL 128 will indicate a REAL variable allocated to address DB 100.128.
A name like STRING 1000 16 will indicate a STRING 16 chars length variable, allocated to address DB 100.1000.

1.4.1 ARG id

The ***id*** field of the argument is used to reference the object within the form by the SetValues() function. The choice to define it UINT12 used in the example is just an example, it would be better to use a definition that remind to the meaning (Example "SetPoint", "Preset", etc.).

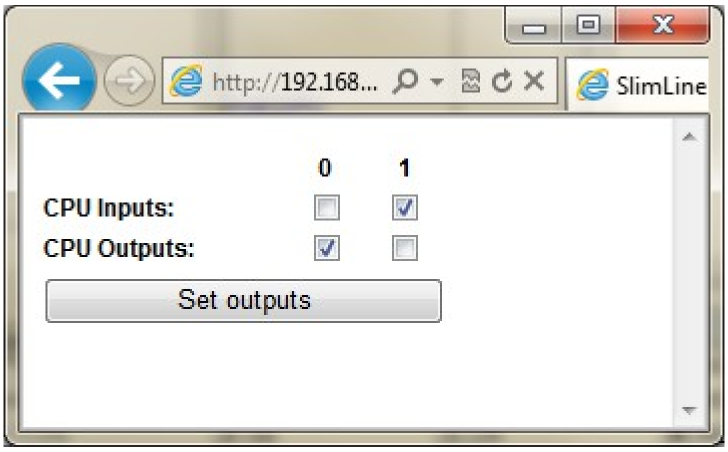
1.5 Some examples

Of course web pages to be created to reach your needs by entering the desired objects. To facilitate the development of own pages we give the PTP128*000 demonstration program which contains a number of SlimLine programs and related web pages.

To test the various examples, transfer the progrmam on the CPU module with LogicLab and with an FTP client transfer the htm page in the ***Storage*** directory. Now from a browser you type the IP address of the CPU module followed by the directory and the name of the page Example ***http://192.168.0.122/Storage/Page.htm***.

Below are details of some of the examples in the demonstration program..

1.6 LogicIO, logic I/O management



	0	1
CPU Inputs:	☐	✓
CPU Outputs:	✓	☐

Here's an example of logic I/O management from web page, to view the status of the inputs and set the outputs have been used checkbox objects. The state of active is indicated by the presence of the tick, to activate the outputs set the tick on the desired output and acts on the ***Set outputs*** button.

To view the actual status of the inputs the page is automatically reload every 10 seconds. To reload the page after the ***<head>*** directive is placed the statement:

<meta http-equiv="refresh" content="10">

To manage the page some javascript functions are used.

Check(Field, Value), Sets or removes the tick symbol on the checkbox object indicated in ***Field*** based on ***Value***.

SetValues(), Executed at page lod it updates all checkbox objects.

SubmitForm(Form), Executed on Set outputs button pressure, it checks if logic output checkbox are set and updates the value of hidden fields that write the PLC variables.

Javascript functions source

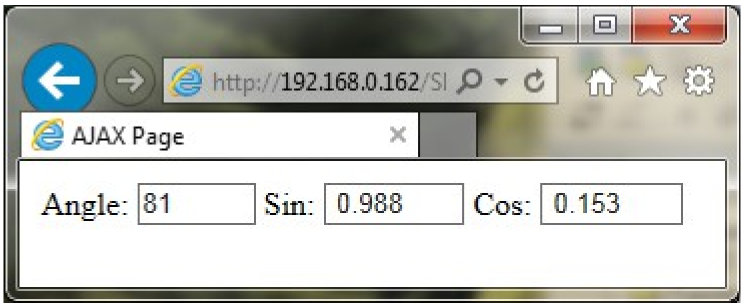
```
<script language="javascript">
function Check(Field, Value) {document.MyForm[Field].checked=(Value != 0);}
function SetValues()
{
    Check("Inp00", '<!--["%d", BOOL, 0]-->');
    Check("Inp01", '<!--["%d", BOOL, 1]-->');
    Check("Out00", '<!--["%d", BOOL, 3]-->');
    Check("Out01", '<!--["%d", BOOL, 4]-->');
}

function SubmitForm(Form)
{
    if (document.getElementById('Out00').checked) document.getElementById('BOOL3').value="1";
    if (document.getElementById('Out01').checked) document.getElementById('BOOL4').value="1";
    document.forms[Form].submit();
}
</script>
```

1.7 Updating pages with AJAX

AJAX , acronym for Asynchronous JavaScript and XML, is a software development technique for creating interactive web applications. The development of HTML applications with AJAX is based on an exchange of background data between server and web browser , which enables the dynamic update of a web page without explicitly reloading by the user.

This technique allows for an automatic update of data in a web page without reloading the page, allowing to view the PLC tags automatically. Let's see how this technique works, a java script must be inserted into the web page, it handles AJAX requests . A ready to use script is supplied (Our code SFW191*000).

Here a web page that displays the value of an angle and the respective values of sine and cosine . The values are managed by the SlimLine that executes the change automatically. The values are stored in 3 variables on the DB 100



Angle: 81 Sin: 0.988 Cos: 0.153

Angle UINT DB 100.0 It contains the angle

Sin REAL DB 100.4 It contains the sine

Cos REAL DB 100.8 It contains the cosine

html page source

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>AJAX Page</title>
<script type="text/javascript">

// ********************************************************************************
// "SFW191A000" FUNZIONI PER GESTIONE AJAX
// ********************************************************************************
// Le seguenti funzioni gestiscono lo standard AJAX "Asynchronous Java and XML",
// con esse viene gestito lo scambio dinamico di dati con le pagine web.

var XMLHttp=AJAXCreateReqObject();function AJAXCreateReqObject(){var b=null;var
a=navigator.userAgent.toUpperCase();if(window.XMLHttpRequest){b=new
XMLHttpRequest()}else{if(window.ActiveXObject&&(a.indexOf("MSIE 4")<0)){if(a.indexOf("MSIE 5")<0){b=new
ActiveXObject("Msxml2.XMLHTTP")}else{b=new ActiveXObject("Microsoft.XMLHTTP")}}}return(b)}function
AJAXSendRequest(b){var a=Math.random();if(XMLHttp!=null){XMLHttp.open("GET",b+"?
Rnd="+escape(a),true);XMLHttp.setRequestHeader("connection","close");XMLHttp.onreadystatechange=AJAXHandleR
sp;XMLHttp.send(null)}}function AJAXHandleRsp(){switch(XMLHttp.readyState){case 0:break;case 1:break;case
2:break;case 3:break;case 4:if(XMLHttp.status==200){SetupValues(XMLHttp.responseText)}break}};

// ********************************************************************************
// FUNZIONE "SetupValues(PContent)"
// ********************************************************************************
// Questa funzione viene eseguita su risposta Ajax, nella variabile "PContent"
// è presente tutto il contenuto della pagina richiesta.
// ----------------------------------------------------------------------------

function SetupValues(PContent)
{
   var Value=new Array(); //Array valori ricevuti da server

   // Eseguo separazione valori, sono separati dal simbolo "|".

   if (PContent.indexOf('|') != -1)
   {
      Value=PContent.split('|');
```

```
        document.getElementById("Angle").value=Value[0];
        document.getElementById("Sin").value=Value[1];
        document.getElementById("Cos").value=Value[2];
    }
}

</script>
</head>
<body onLoad="setInterval('AJAXSendRequest(\'Values.htm\')', 3000)">
<table border="0">
  <tr>
    <td>Angle:</td>
    <td><input type="text" id="Angle" size="4" maxlength="4"/></td>
    <td>Sin:</td>
    <td><input type="text" id="Sin" size="6" maxlength="6"/></td>
    <td>Cos:</td>
    <td><input type="text" id="Cos" size="6" maxlength="6"/></td>
  </tr>
</table>
</body>
</html>
```

On page load ***<body onLoad="setInterval('AJAXSendRequest(\'Values.htm\')', 3000)">*** the AJAX request of the page ***Values.htm*** is performed every 3 seconds. The return value of this page is automatically passed to the function ***SetupValues*** that it parse it and copy the values on the display objects. The page ***Values.htm*** returns the values of the 3 variables separated by the **|** symbol. Here's the listing for this page.

Values.htm source page

```
<!--['%d', UINT, 0]-->|<!--['%6.3f', REAL, 4]-->|<!--['%6.3f', REAL, 8]-->
```