simex

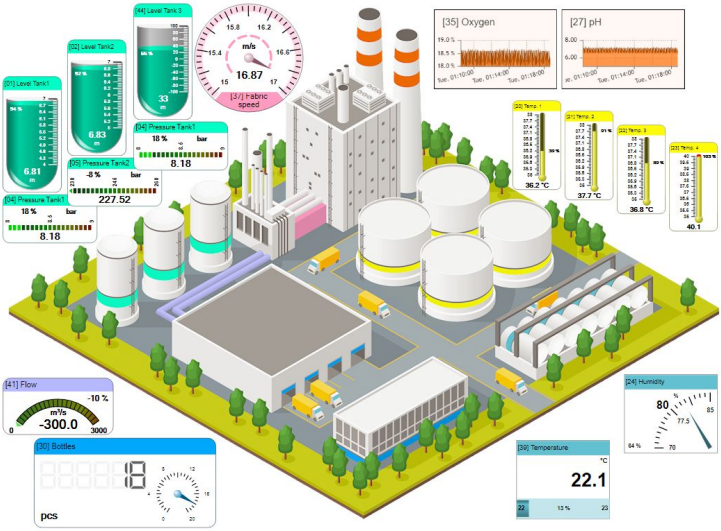
Assisting the automation industry since 1986

Visualization creation tutorial using

MultiCon Sidgets

- Firmware version of MultiCon: from **5.14.0**
- Designed for cooperation with MultiCon devices

MultiCon Sidgets



Please read this tutorial carefully before using the software.
The manufacturer reserves the right to make changes without prior notice.

CONTENTS

The meaning of the symbols used in this tutorial:

This symbol draws attention to particularly important descriptions.
It is recommended that you read the notes marked with this symbol carefully.

This symbol indicates additional information and explanations that may be useful to fully understand the issue.

1. BACKGROUND

MultiCon Sidgets technology gives the possibility of the easy creation of SCADA visualizations for real-time viewing of measurements provided by **MultiCon** devices. Each visualization is created by the user in the form of a web page, but in order to create a simple visualization, no programming expertise is required. To create simple visualizations, you only need to read this guide. Additionally, **MultiCon Sidgets** technology has been developed in such a way that more experienced users can use it as part of a larger IT system, e.g. with visualizations available in the area of user accounts.

The main assumption in the creation of this technology was to provide the minimum number of actions needed by the user to create a simple, dynamically changing visualization of the industrial process, which could be launched by any device with the ability to browse the Internet via a web browser.

As this technology works in accordance with the latest web development standards, such as HTML5+, CSS3+, JS6+, it requires the use of newer versions of web browsers[1], but the hardware platform or operating system used to display this visualization is irrelevant. Enabling access to **MultiCon** devices via the Internet and placing the prepared visualization on a publicly accessible server will additionally allow for surveillance of monitored facilities from any place in the world.

sidget – a graphic object placed on a web page, used to visualize in real-time the measurements changing in the **MultiCon** series devices. The graphic representation of this object may resemble a physical measuring device such as a pointer meter, diode, LED bargraph, chart recorder, etc. A sidget does not require setting the basic parameters of the visualized channel, as they are continuously downloaded directly from the **MultiCon** device.

[1] For a list of tested versions of web browsers, see chapter 5.

2. SIDGET OPERATION TEST

2.1. INTRODUCTION

In order to check the correct connection of the computer with the **MultiCon** device and the correct operation of the sidget engine, let us create a test visualization. It will be prepared locally (on your computer) as the simplest web page.

2.2. PREPARATION OF INFRASTRUCTURE

Sidget visualizations work by constantly downloading data from **MultiCon** devices via a web browser. Therefore, the following components will need to be prepared to test the operation of the sidgets:

- **MultiCon** device with firmware version, at least **5.07.0** connected to a LAN at a known IP address, e.g. 192.168.1.222
- user's PC connected to the same LAN as the above device (direct connection to the device is also possible)
- computer with any Windows operating system (tutorial applies to Windows 10)

2.3. PREPARATION OF THE TEST PAGE

In order to create on the user's computer the easiest web page using the sidget engine, it is recommended to create a special folder in which such visualizations will be placed. To do this, let's open **File Explorer** and create a folder with any name, e.g. "Sidgets", in the selected computer location. The way of creating such a folder on Windows 10 is shown in the figure below.

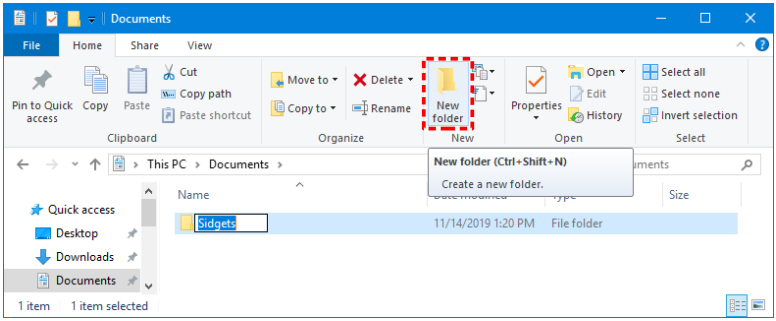


Fig. 2.1 Creating a folder to group different visualizations

It is good practice to create additionally in the above folder separate subfolders for visualizations using common resources, e.g. graphics, styles, etc. So let's create an additional subfolder called "Webpage1" (Figure 2.2).

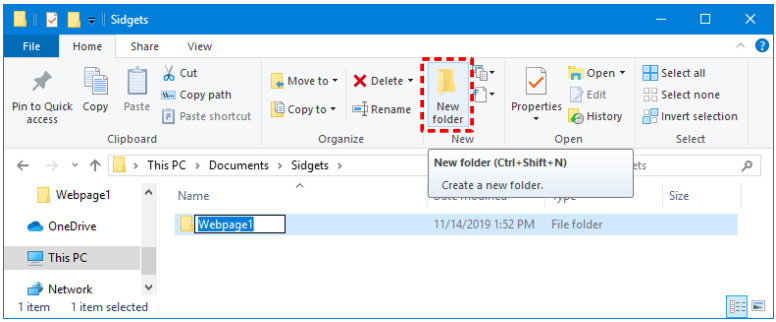


Fig. 2.2 Creating a folder for the first visualization

Before creating a file with the first visualization, make sure that the display of file name extensions is enabled (Figure 2.3).

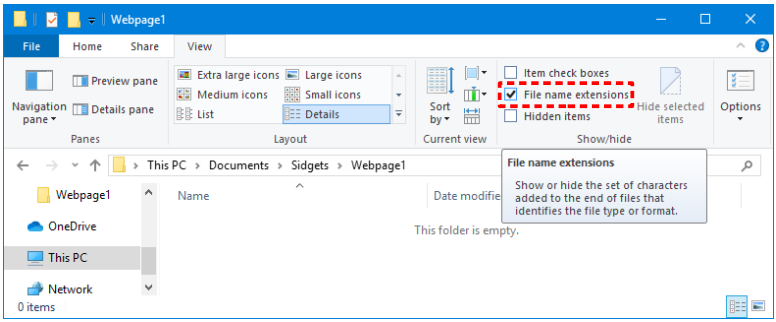


Fig. 2.3 "File name extensions" option enabled

To create a test visualization web page, create a new text file in the "Webpage1" folder using the appropriate option in the **File Explorer** menu or the mouse context menu (Figure 2.4).

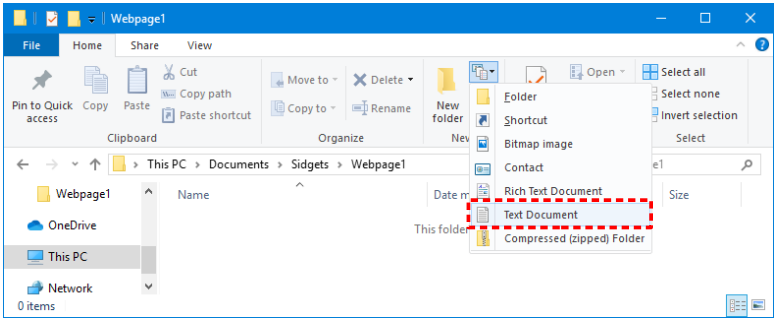


Fig. 2.4 Creating a new web page file

When creating a text file, you should give it a proper name by changing the default extension *.txt to *.html (Figure 2.5).

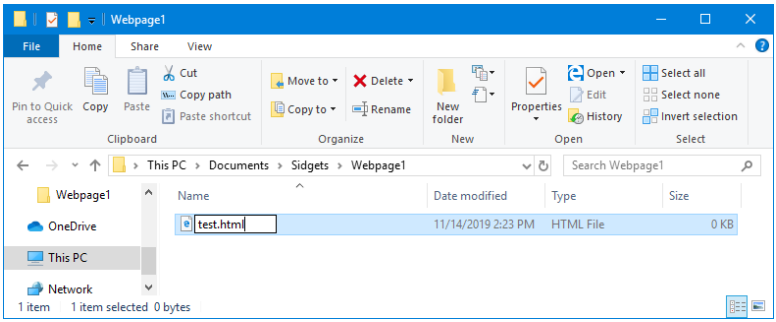


Fig. 2.5 Proper web page name

In this way, an empty web page file was created. It is now necessary to fill its content with the basic HTML code – essential for the correct launch of the visualization in a web browser.
You can use the system's **Notepad** tool for this purpose. To do this, open the context menu of the mouse on the previously created file and select the commands:

"Open with" > "Choose another app" > "More apps |" > "Notepad" > OK

After opening the "test.html" file in **Notepad**, you should enter or paste the following HTML code into it.

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://192.168.1.222/sidgets.js"></script>
</head>
<body>
  <div class="sidget"></div>
</body>
</html>
```

Listing 2.1. HTML code to test the sidget engine

As a result of the above actions, the **Notepad** window should look like in the figure below.

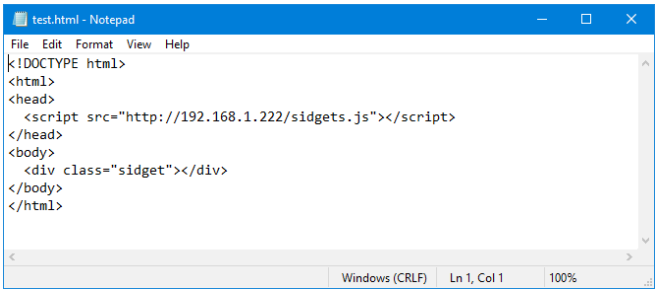


Fig. 2.6 ***Notepad*** *window with pasted HTML code*

After saving the changes in "test.html" file, you can test its operation. To do so, simply run this file by double-clicking from the **File Explorer**. The visualization will be launched using the default web browser assigned to the *.html file extension. If the connection with the device works properly, after a short while a graphic object should appear on the screen displaying not only the current measurement but also the current settings of one of the measurement channels with which the **MultiCon** device works (Figure 2.7).

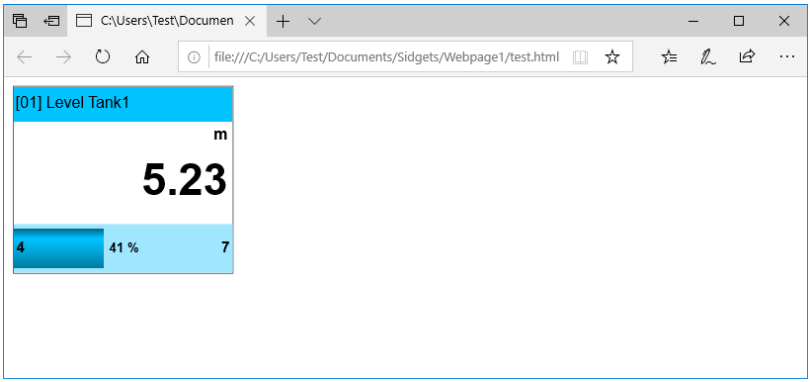


Fig. 2.7 Test visualization appearance in ***Microsoft Edge*** *browser*

The explanation of how the HTML code used works is described in chapter 4.

3. PREPARATION OF THE ENVIRONMENT FOR CREATING VISUALIZATION

3.1. INTRODUCTION

Although it is possible and sufficient to create and test web pages using the tools provided with the Windows operating system (**File Explorer, Notepad, Microsoft Edge** browser), it is recommended to install appropriate tools to create more complex visualizations that will naturally support this process.
This chapter will show you how to prepare a free editorial environment that allows you to create multiple visualization pages that use the sidget engine in a fast and user-friendly way. At the same time, it will not limit the possibility of editing the code of web pages for users who deal with these issues professionally on a daily basis.
One of the editors supporting code creation in HTML, CSS, JavaScript and many other web-related technologies, is a free (also for commercial purposes) Microsoft product – **Visual Studio Code**. Although this tutorial is about creating visualization pages on the Windows operating system, there is nothing to prevent you from applying the appropriate actions to other operating systems. The **Visual Studio Code** editor is also available for Linux and macOS platforms.

3.2. INSTALLATION OF VISUAL STUDIO CODE

After downloading the **Visual Studio Code** installer from https://code.visualstudio.com/, you need to install it. The installation process is typical for Windows programs, but it is worth noting the stage where additional actions are available. The most important of these are shown in the figure below.

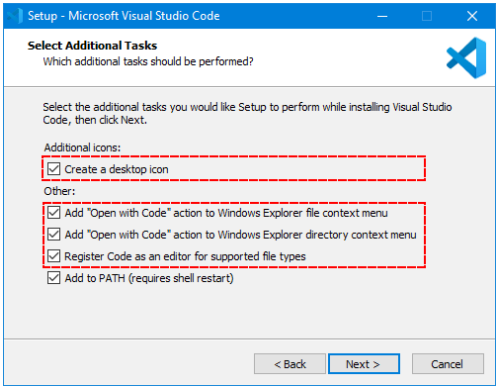


Fig. 3.1 Selection of additional installation options that will facilitate later work with the program

Installing the program according to the suggestions from the figure will make it easier for the user to follow the steps as presented in this tutorial.

3.3. LAUNCH OF VISUAL STUDIO CODE

Once **Visual Studio Code** is installed, it can be started using the shortcut created in the operating system. But having already prepared the "test.html" page (see Chapter 2), you can use the new **Open with Code** command that appears in the context menu of *.html files (Figure 3.2). This command will allow you to edit this file immediately with the new editor.

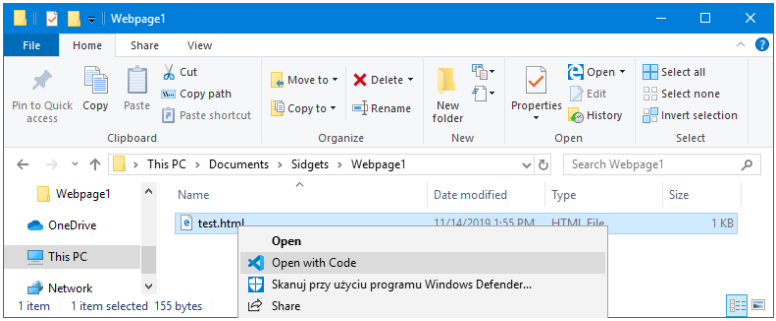


*Fig. 3.2 Editing *.html file using **Visual Studio Code***

The above method of opening files is similar to the one used when editing a file using the system **Notepad**, however, for more complex projects it is recommended to open whole folders directly from the program itself.

3.4. ADJUSTMENT OF VISUAL STUDIO CODE

The display color of the user interface and code in **Visual Studio Code** is set to dark mode by default. This tutorial uses screenshots and code syntax in light colors. For consistency, the user can also change this mode by selecting the appropriate command from the menu **File > Preferences > Color Theme** or by pressing the keyboard shortcut **Ctrl + (K > T)** and then selecting "Light+ (default light)" (Figure 3.3).

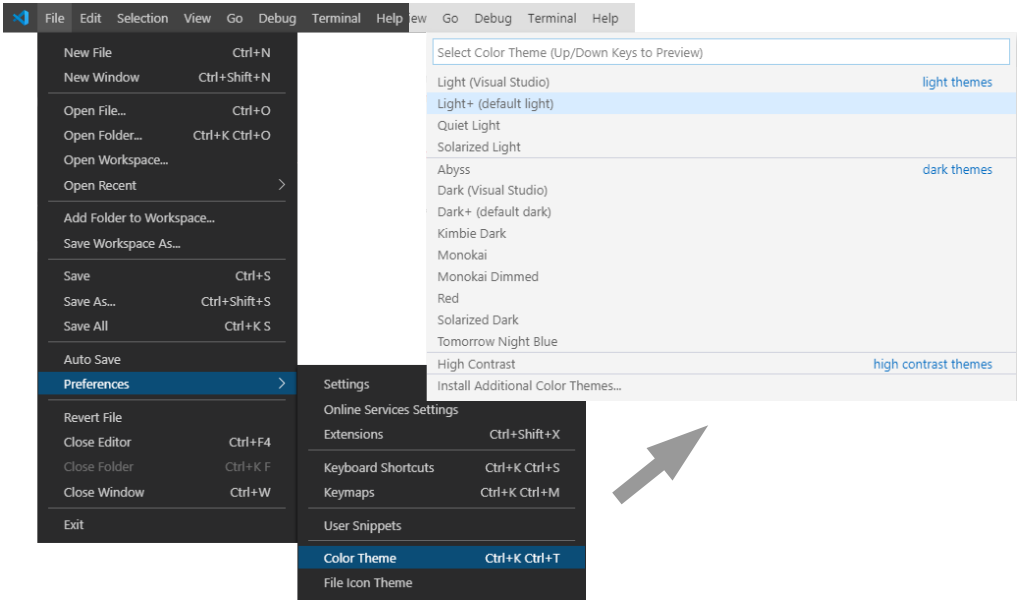


Fig. 3.3 Changing the color scheme to light

Another activity you can do in the Visual Studio Code environment for working with sidgets is to install extensions for a quick preview of the final effect of the edited page. To do this, open the extension manager by using the Extensions command from the View menu or by clicking on the corresponding icon in the side menu (Figure 3.4).

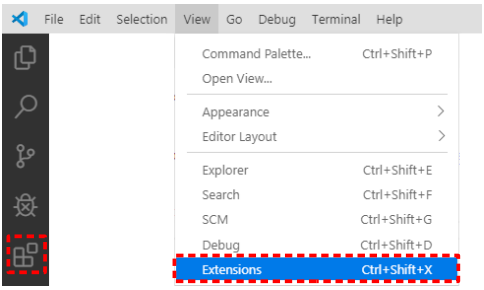


Fig. 3.4 Opening the extension manager

Then, enter the "live server" phrase in the search field. This will limit the list to such items, among which you will find our extensions of interest. Extensions you should install are **Live Server Preview** and **Live Server**. To do this, click on the **[Install]** button next to each of them (Figure 3.5).

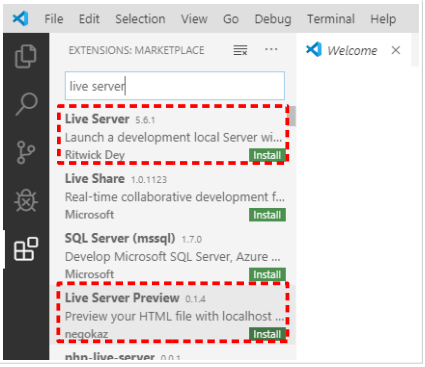


Fig. 3.5 Extensions recommended for installation

The **Live Server Preview** extension is used to quickly preview the end result of a created HTML page as it would be rendered in a web browser.

In order to test the operation of the extension, make sure that any *.html file is open for editing in the program, e.g. a previously created "test.html" file. Then select **Command Palette...** from the **View** menu or press the keyboard shortcut **F1** and after searching, run the "Show Live Server Preview" command (Figure 3.6).

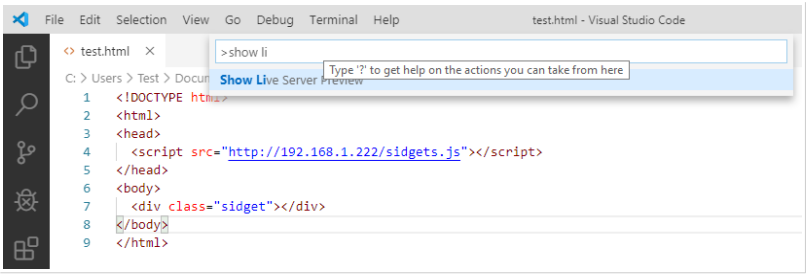


Fig. 3.6 Selecting the command to start previewing the current file

This will cause the program to run a separate tab, which will display a preview of the operation of the just modified *.html file (Figure 3.7).

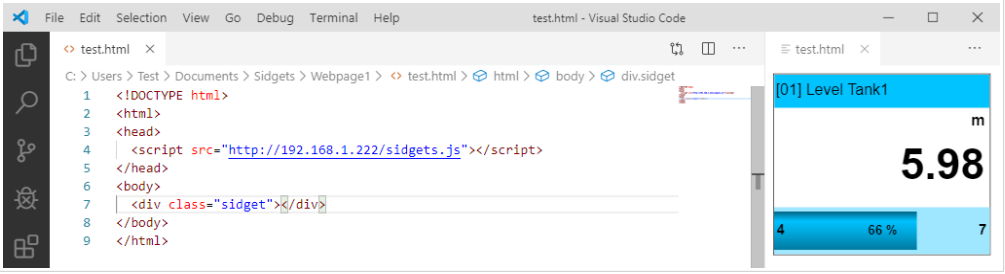


*Fig. 3.7 Preview of web page operation with **Live Server Preview** extension*

During further development of the web page with sidgets, any saving of changes to the *.html file will automatically refresh the preview tab. This solution is more practical than switching frequently between the **Visual Studio Code** environment and a web browser, whose content should be refreshed additionally.

For more complex websites, easy access to the whole area of their content or better control over the code execution using browser development tools, it is recommended to use the facilities offered by the **Live Server** extension. This extension, like **Live Server Preview**, keeps track of saving changes to the page file and automatically refreshes the preview. However, this already works in a separate web browser window where we want to test our web page.

In order to test the operation of the **Live Server** extension, it is no longer sufficient to open a single *.html file for editing, as in the case of the **Live Server Preview** extension, but it is required to open the entire project with the created web page. This usually comes down to opening the parent folder where our page is located. So let's open the previously created the "Webpage1" folder with the **Open Folder...** command from the **File** menu. This will cause the file structure of this folder to appear in the side menu. Then open the "test.html" file from the structure displayed in this way for editing. Clicking on the **Go Live** field in the status bar will launch a simplified version of the local web server and open the designed visualization page in the default web browser window. While the web server is active, the **Go Live** field should change its name to one that indicates the port number on which the server operates (Figure 3.8).

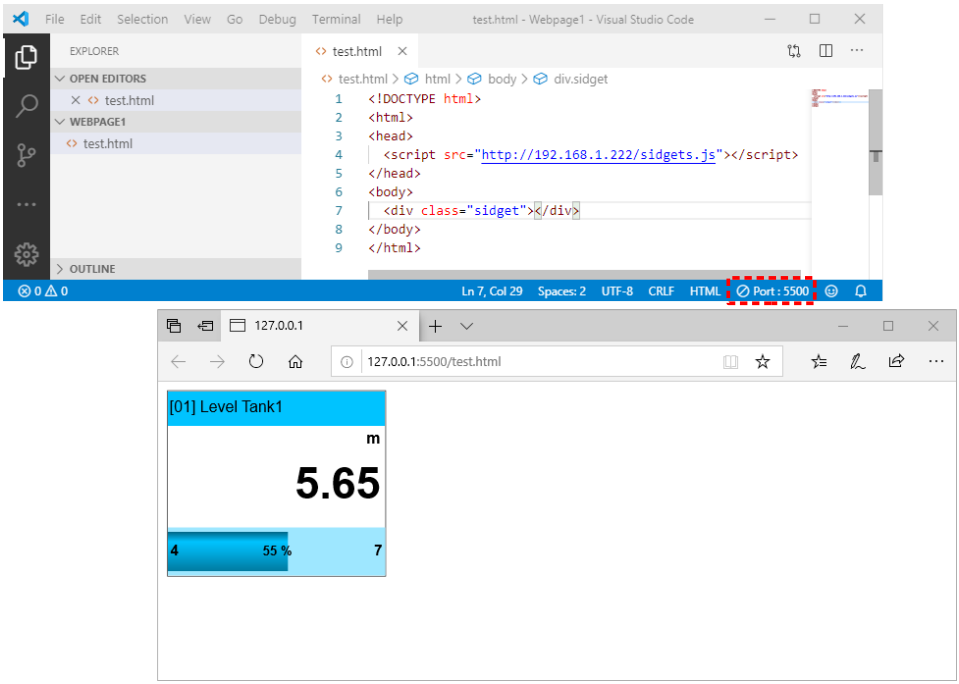


*Fig. 3.8 Preview of website operation with the **Live Server** extension*

As long as the web server is running, the URL from the default opened web browser can be copied and also run in another web browser installed on your computer.

Configuration of the **Visual Studio Code** environment according to the above guidelines is not necessary and does not have to be limited to these changes, but it is a starting point for the convenient creation of web pages with sidgets.

4. CREATING VISUALIZATIONS

4.1. INTRODUCTION

Creating visualization pages using sidgets consists of declaring guidelines for individual objects, which determine the appearance, behavior and data source. The whole logic that takes care of these declarations is done automatically. This chapter will discuss how all the declarations work that the user can define when creating a visualization.

After completing this chapter, you will have the skills to understand how to work and create simple visualizations using sidgets. Deepening the knowledge of website creation can further expand the possibilities that are not mentioned in this tutorial.

4.2. BASIC VISUALIZATION STRUCTURE

Each valid web page should contain a code with a structure at least as shown in listing 4.1.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 </head>
6 <body>
7
8 </body>
9 </html>
```

Listing 4.1. Basic structure of each web page

This is due to certain guidelines for building websites using HTML[2].

Section 2.3 shows how to create a simple test visualization that displays a single sidget. For this purpose, the most basic HTML code that should be included in any visualization using sidgets was used. It is shown again in the listing below.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <script src="http://192.168.1.222/sidgets.js"></script>
5 </head>
6 <body>
7   <div class="sidget"></div>
8 </body>
9 </html>
```

Listing 4.2. 4.2. HTML code to test the sidget engine

As you can see, in relation to the basic structure of an HTML document, additional code has been declared in lines 4 and 7. These are declarations related to sidget technology.

[2] More knowledge about building web pages in HTML technology can be obtained from many widely available books and tutorials

The line

```
4   <script src="http://192.168.1.222/sidgets.js"></script>
```

causes attach to the web page the engine responsible for automatic creation of sidgets in the user's declared places on the web page and for refreshing their measurements.

*The "sidgets.js" engine is located in every **MultiCon** device with a firmware version of at least **5.07**. If the device has an older firmware version, if you want to use the functionality of creating your own visualizations using this technology, you must first update the device software to the latest available version.*

The next line

```
7   <div class="sidget"></div>
```

informs the engine that the above `<div>` tag will be a container for the sidget. This is because a `sidget` class has been assigned to the item and all HTML tags acting as block containers, e.g. `<div>`, or as linear containers, e.g. `<span>`, that have this class assigned are the places where the sidgets will be automatically created. Such tags can be placed anywhere in the structure of an HTML document.

Several other valid tags inside which the sidgets will be created:

```
<span class="sidget"></span>

<p class="sidget"></p>

<object class="sidget"></object>

<table>
  <tr>
    <td class="sidget"></td>
    <td class="sidget"></td>
  </tr>
</table>
```

4.3. SIDGET ENGINE ATTRIBUTES

The sidget engine can be started with special parameters that change the way it works. These parameters can be useful during creating and testing visualization. They are declared as attributes with the `data-…` prefix in the `<script>` tag, e.g.

```
<script src="http://192.168.1.5/sidgets.js" data-mode=1 data-demo="random"></script>
```

The list of possible attributes of the sidget engine to declare is as follows:

`data-mode` – *defines the mode in which the engine will run the visualization*

Possible values
`0` **preview mode** (default) – it is the basic mode of operation of a completed visualization
`1` **edit mode** – unlocks the functionality of changing sidgets' positions and activates tools to identify their settings

It is not necessary to specify the engine mode to create a visualization, but running with active the **edit mode** will make the process easier. The way this mode works is described in chapter **4.6 CREATING VISUALIZATION IN PRACTICE**.

Remember to delete this parameter or set it to `0` before you put the visualization into use.

`data-demo` – *determines how to obtain the measurement value by sidgets*

Possible values
`"chan"` (default) – each sidget displays the changing **actual value of the measurement** taken from the selected channel of the device on a current basis
`"random"` – each sidget displays a changing **simulated value generated at random** from a range slightly exceeding the upper and lower limits set in the device channel
`"value"` – each sidget displays the same **fixed value** given in this attribute, e.g. `"-5.3"`, `"912"`, `"-inf"`, `"inf"`

The engine's ability to simulate changing measurement values can facilitate the creation of different visualizations, even without setting the target logic in the **MultiCon** device. However, please note that despite the simulation of the measurement itself, channel settings such as name, unit, display precision, chart limits, etc., are still taken directly from the device and displayed in sidgets.

Remember to delete this parameter before you put the visualization into use.

`data-hide-inactive` – *specifies how to control the visibility of containers with sidgets for inactive channels*

Possible values
`0` **don't hide** (default) – all sidget containers are visible
`1` **hide when the channel is inactive** – containers with sidgets in which the declared channel does not exist or is inactive are hidden

Hidden container with a sidget as a result of the `data-hide-inactive=1` attribute doesn't take up space on the page. So let's use this advantage to design more universal pages with an automatically adjusted layout depending on the enabled channels.

4.4. SIDGET PARAMETERS

Example from Listing 4.2 works correctly. However, it is difficult to find its practical application because, as you can see, the measurement from channel 1 is displayed in the form of a numerical indicator, although this has not explicitly specified. These are the default values of each sidget, which rarely correspond to real needs. Therefore, in each container, additional parameters should be provided specifying the source of data and the way it is presented.

The parameters that can be specified for each sidget or group of sidgets should be given as CSS styles. For example, if you want to specify that the measurement value is to be taken from channel 2, you should slightly modify the object declaration line to the form:

```
7   <div class="sidget" style="--sid-chan:2;"></div>
```

As you can see, the parameter starts with a prefix `--sid-...`. This is a common feature of all parameters specific to sidget objects and allows to easily distinguish them from style properties already covered by the CSS language standard itself.

The parameters defining sidget settings are as follows:

`--sid-host` – *specifies the source device from which the measurements will be read out*

Allowed values	Default value	Example values

IP number or host name of the **MultiCon** device (may include HTTP port number)	host specified in `src` in the `<script>` tag	`'192.168.1.151:80'`
		`'dev1.multicon.com'`
		`'192.168.100.5'`

Sidgets allow you to read measurement data from multiple devices simultaneously on one web page. If we specify with the parameter `--sid-host` another device than the one from which the "sidgets.js" engine is taken, then the sidget or their group affected by this parameter will take measurements from this data source.

The maximum number of possible data sources for one web page is limited by the web browser used. In order to maintain full compatibility between different browsers, it is recommended not to exceed 5 different hosts for one visualization.

`--sid-chan` – *specifies the channel number from which settings and measurements will be taken*

Allowed values	Default value	Example values
`1` … `60` (**MultiCon 99/N16**) or `1` … `90` (**MultiCon 141**)	`1`	`15` `3`

This parameter specifies the channel number of the device from which the settings and current measurements will be taken. The information obtained in this way will be automatically used by the selected sidget (Figure 4.1).

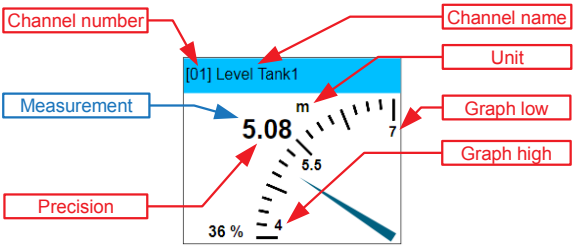


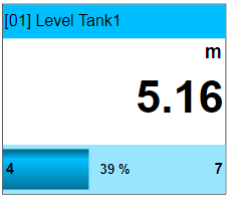
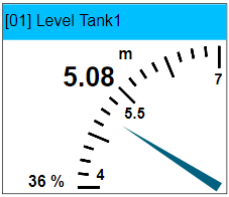
Fig. 4.1 Example of displaying by sidget the measurement and settings for a selected channel

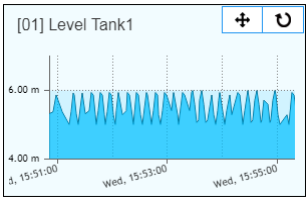
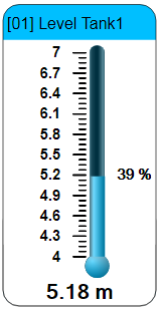
`--sid-type` – *specifies the type of sidget used*

Allowed values	Default value	Example values

`0` ... `13`
or
`'name'`
	`1`	`2`
		`'Analog Meter'`
		`'analogmeter'`

This parameter specifies the sidget type. It is associated with a different appearance and behavior of the object, compared to other types. The value of this parameter can be given as a **Number** or **Name** among the permissible values shown in the table below.

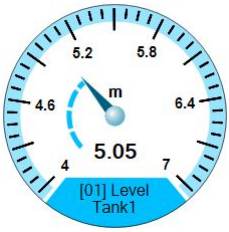
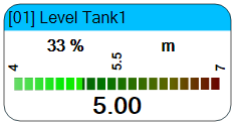
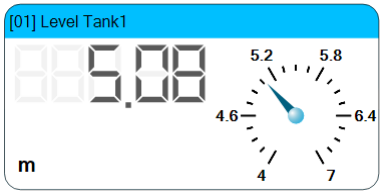
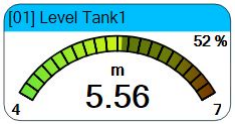
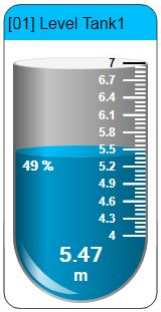
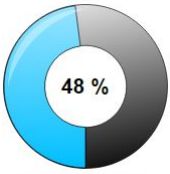
Number	Name	Appearance
`0`	`Text`	*The current value of the "Level Tank1" channel is 5.16 m and is 39% of the range*
`1` (default)	`Value`	
`2`	`Needle`	

<table>
<tr><th>Number</th><th>Name</th><th>Appearance</th></tr>
<tr><td>3</td><td>Graph</td><td> 
</td></tr>
<tr><td>4</td><td>Thermometer</td><td> 
</td></tr>
<tr><td>5</td><td>Two-state LED</td><td> 

<table>
<tr><th>Measurement</th><th>LED color</th></tr>
<tr><td>P ≤ 0</td><td>(transparent)</td></tr>
<tr><td>P > 0</td><td>--sid-color</td></tr>
</table></td></tr>
<tr><td>6</td><td>Three-state LED</td><td> 

<table>
<tr><th>Measurement</th><th>LED color</th></tr>
<tr><td>P > Graph high</td><td>--sid-hicolor</td></tr>
<tr><td>any</td><td>--sid-color</td></tr>
<tr><td>P < Graph low</td><td>--sid-locolor</td></tr>
</table></td></tr>
<tr><td>7</td><td>Three-state Rectangular LED</td><td> 

<table>
<tr><th>Measurement</th><th>LED color</th></tr>
<tr><td>P > Graph high</td><td>--sid-hicolor</td></tr>
<tr><td>any</td><td>--sid-color</td></tr>
<tr><td>P < Graph low</td><td>--sid-locolor</td></tr>
</table></td></tr>
</table>

Number	Name	Appearance
8	Analog Meter	
9	Horizontal LED Bar	
10	Digital & Analog	
11	Arc LED Bar	
12	Tank	
13	Pie Chart	

`--sid-interval` – *specifies in seconds the time between measurement refreshes*

Allowed values	Default value	Example values
`1` … *n*	`1`	`5` `300`

This parameter specifies the time interval that tells to the sidget how often it should refresh its measurement value. This time is expressed in full seconds.

The internal mechanism of the "sidgets.js" engine constantly polls all devices for new measurements with the frequency resulting from the user-set time intervals. If several sidgets that relate to the same device have different intervals set, the engine will optimize the polling of the device to provide the declared intervals of refresh values, while keeping the minimum load on the device. Channel settings are downloaded continuously every 15 seconds.

Setting the `--sid-interval` parameter is particularly important for sidgets using historical data, e.g. graphs (`Graph`) or text sidgets (`Text`). These type of sidgets have a buffer of **600 measurements**, so each graph can display at most as many points from the past. This means that by selecting the appropriate update interval, we can control the range of time that can be reached in the past to be able to read the recorded measurement. For example, specifying the update time as `--sid-interval:60;` means that every point will be added to the graph every minute. If the graph can accumulate 600 measurements, then 60s*600 = 36000s = 10h. In this case, the graph will present data up to 10 hours back, where each addition of a new point will delete the oldest one.

`--sid-color` – *defines the main color of the sidget*

this parameter doesn't apply to the following type of sidgets: **`0-Text`**

Allowed values	Default value	Example values

name or numerical value of the color	`DeepSkyBlue`	`Yellow`
		`#14ff01`
		`rgb(83, 83, 245)`

Sidgets have a main color styling mechanism. It is based on the fact that the sidget's design has one basic color, in relation to which important elements of the sidget automatically adjust, maintaining the designed brightness levels. This functionality allows you to easily adjust the colors of dynamic objects in relation to other graphic elements of the page. It also allows to distinguish selected groups of sidgets.
The `--sid-color` parameter is used to determine the main color of the sidget. As a value, the parameter takes the color given as a name, hexadecimal code or in RGB format – according to the CSS standard. A list of all color names can be found at https://html-color-codes.info/color-names/.

Sidgets displaying the measurement in a two-state manner, e.g. diodes, can interpret this parameter as a color indicating their active state.

`--sid-hicolor` *– determines the color of the sidget element, which informs the user that the measurement exceeds the **Graph high** set in the device channel*

this parameter doesn't apply to the following type of sidgets: **`0-Text`**

Allowed values	Default value	Example values

name or numerical value of the color	color depended on the sidget type, usually `Red`	`Magenta`
		`#14ff01`
		`rgb(83, 83, 245)`

`--sid-locolor` – *determines the color of the sidget element, which informs the user that the measurement exceeds the* ***Graph low*** *set in the device channel*

this parameter doesn't apply to the following type of sidgets: **`0-Text`**

Allowed values	Default value	Example values

name or numerical value of the color	color depended on the sidget type, usually `Blue`	`Green`
		`#14ff01`
		`rgb(83, 83, 245)`

The most sidgets have a mechanism to inform that the measurement exceeds set in the channel the range of displayed values by **Graph high** and **Graph low** settings.

If the measurement increases and exceeds the upper limit, the sidget will display the relevant graphical information in the pre-designed color for such an exceedance (usually red).

If the measurement decreases below the lower limit, the sidget will display the relevant graphical information in the pre-designed color for such an exceedance (usually blue).

Exemplary reactions of several sidgets to such events are shown in the figure 4.2.

Fig. 4.2 Exemplary reactions of selected sidgets to the value of the upper (left) or lower (right) display limit being exceeded

Parameters `--sid-hicolor` and `--sid-locolor` allow to change the default colors of graphical elements that inform about the exceedances.

The rules for the value format for this parameter are the same as for the `--sid-color` parameter.

`--sid-scaling` – *determines the scaling factor of the sidget as a percentage of the default dimensions*

this parameter doesn't apply to the following type of sidgets: **`0-Text`**

Allowed values	Default value	Example values
1, 2, …, *n*	100	80 150

Each sidget is designed to be as small as possible on a 1:1 scale, without sacrificing good readability. Depending on the needs, the user can reduce or enlarge such an object while maintaining its original proportions. The above parameter is used to change the scale of the object, while the number given as a value determines the percentage of the sidget's default dimensions.

Using this parameter overrides the ability to specify dimensions using the `width` and `height` properties in the sidget style.

4.5. WAYS TO DECLARE SIDGET PARAMETERS

Chapter 4.4 provides a way to declare the `--sid-chan` sidget parameter directly in the code line where this sidget was declared. This way of declaring the parameter is only one of many the user can use. As the sidget parameters are CSS (*Cascading Style Sheets*) properties, they can be combined with other known CSS language properties, e.g. the `width` property, ultimately building a style used by single or multiple HTML tags.

The CSS language defines three basic possibilities of style declaration:

A) **local style, called "inline style"** – consists of the direct assignment of a style to an HTML element through a `style` attribute, e.g.

```
<body>
  <div class="sidget" style="--sid-type:2; --sid-chan:3; width:300px;"></div>
  <div class="sidget" style="--sid-type:2; --sid-chan:4;"></div>
</body>
```

B) **internal style** – this is a sheet placed inside the `<style>` tag in the header part of the page, e.g.

```
<head>
  <style>
    .sidget {
       --sid-type: 2;
    }
  </style>
</head>
<body>
  <div class="sidget" style="--sid-chan:3; width:300px;"></div>
  <div class="sidget" style="--sid-chan:4;"></div>
</body>
```

C) **external style** – it consists of placing the definition of styles in a separate file with the *.css extension and placing the import instruction in the header part of the web page using it, e.g.

```
.sidget {
 --sid-type: 2;
}
```

mysidgets.css

```
<head>
  <link rel="stylesheet" type="text/css" href="mysidgets.css" />
</head>
<body>
  <div class="sidget" style="--sid-chan:3; width:300px;"></div>
  <div class="sidget" style="--sid-chan:4;"></div>
</body>
```

The way the styles are placed is associated with the concept of *cascading*. It defines the hierarchy of style sources. *Cascading* determines that styles from the external sheet are taken into account first. These, in turn, can be overwritten by the styles defined in the inner sheet. The styles defined locally ("inline styles") are located closest to the described element of the HTML page, so the properties defined there will overwrite those that have been set using other methods.

Using the advantages of cascading in the creation of web pages with sidgets is a very useful functionality, by which you can quickly define the common or default parameters of all sidgets using an internal or external style sheet, such as type, color, size, interval, and specific parameters for particular objects, such as channel number or other interval, can be defined using local styles.

The broader discussion of the CSS language goes beyond this tutorial. In order to take full advantage of the possibilities of creating visualizations using sidgets, it is recommended to learn at least in a basic way the syntax of defining styles and the most important features of this language.

4.6. CREATING VISUALIZATION IN PRACTICE

Knowledge of the visualization structure (chapter 4.2), the meaning of sidget parameters (chapter 4.4) and the ways of their declaration (chapter 4.5) allows to create the first web page for the real application.

Listing 4.3 shows the *.html file code, which takes the sidget engine from a device with IP address 192.168.1.100 and declares two sidgets. In addition, each indicator should display in yellow by default and take measurements and settings from a device with IP address 192.168.1.200 and channel number 3. In the declaration of the second sidget, the channel number has been changed to 5 by the local style using a cascade of styles.

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://192.168.1.100/sidgets.js"></script>
  <style>
    .sidget {
      --sid-host: '192.168.1.200';
      --sid-color: Yellow;
      --sid-chan: 3;
    }
  </style>
</head>
<body>
  <div class="sidget"></div>
  <div class="sidget" style="--sid-chan:5;"></div>
</body>
</html>
```

Listing 4.3. HTML code showing the use of the possibility to redefine parameters

In order to make it easier to distinguish one sidget from another, it is recommended to give them unique identifiers using the `id` attribute, e.g.

```
  <div class="sidget" id="sid1"></div>
  <div class="sidget" id="tank-level" style="--sid-chan:5;"></div>
```

This makes it easier to identify sidgets when further modifying the visualization, especially when using the **edit mode** of the page (see chapter 4.3). So let's enable this mode for editing time by adding the following parameter in the engine script declaration:

```
  <script src="http://192.168.1.100/sidgets.js" data-mode=1></script>
```

The page running in this mode will contain additional facilities and tools for editing it.

One such facility is to show an additional label in the top right-hand corner of each object, where is displayed the container identifier (`id` attribute) in which this sidget was created. There is also the number of this sidget. This number is related to the order in which its declaration is placed in the page code. This information allows the user to quickly find the appropriate entry in the code, seeing only the visualization itself (Figure 4.3).

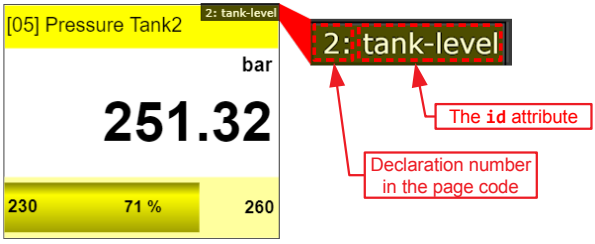


Fig. 4.3 Sidget in edit mode with an additional identification label

In **edit mode**, in addition to the identification label on each sidget, an information bar appears in the top left-hand corner of the page. This is the element that provides information about the currently pointed sidget. In addition to repeating the information from the identification label, it provides coordinates of the position of the upper left corner of the sidget. Clicking on the information bar will expand it and show the detailed settings of the pointed object (Figure 4.4).

Fig. 4.4 Displaying additional information about the sidget

Another facility that appears in the edit mode is assistance in positioning sidgets on the page with the mouse (Figure 4.5).

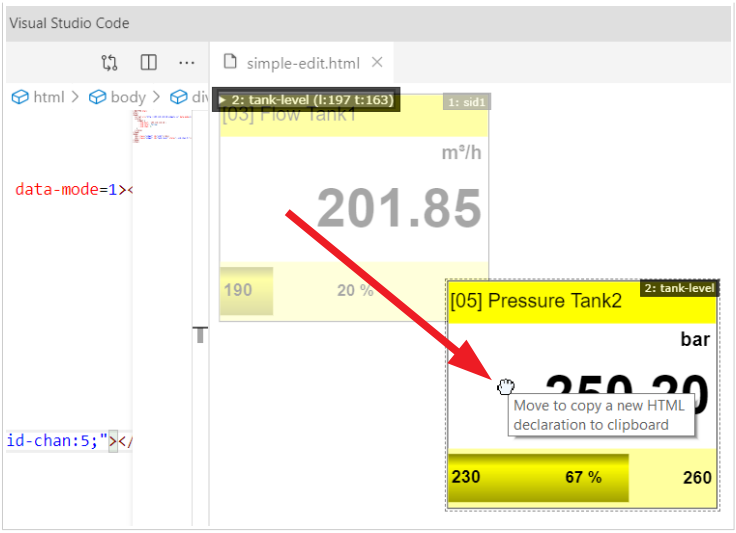


Fig. 4.5 Changing the position of the sidget in edit mode with the mouse

Once you have grabbed and moved the object with the mouse to another location, the HTML declaration appropriate for that sidget, but already modified with a new coordinates, will be copied to the clipboard. This modified line copied to the clipboard can be used to replace the original line in the page code.

```
15    <div class="sidget" id="tank-level" style="--sid-chan:5; left:197px; top:163px;"></div>
```

All elements of the `sidget` class have the `position:absolute;` attribute set by default, so their position is determined relative to the nearest HTML tag being its parent whose style has the property set to `position:relative;`. Therefore, it is recommended to place sidgets in an additional container. So let's create a container as a `<div>` tag and name it a `main-container`.

```
  <div id="main-container" style="position: relative;">
    <div class="sidget" id="sid1"></div>
    <div class="sidget" id="tank-level" style="--sid-chan:5; left:197px;
      top:163px;"></div>
  </div>
```

This way of grouping sidgets allows you to easily add other page elements to this group, in relation to which the sidget position must be given. For example, this could be a background image stored on your computer or placed on the Internet. To add such an image, just use the `<img>` tag and place it inside the top of the container.

```
  <div id="main-container" style="position: relative;">
    <img src="./res/Factory.png">
    <div class="sidget" id="sid1"></div>
    <div class="sidget" id="tank-level" style="--sid-chan:5; left:197px;
      top:163px;"></div>
  </div>
```

The above code shows how to refer to the image placed on the user's computer in the "res" subfolder located in the same folder as the *.html page.

The code of the entire visualization being result of above steps should be similar to the one presented in listing 4.4.

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://192.168.1.100/sidgets.js" data-mode=1></script>
  <style>
    .sidget {
      --sid-host: '192.168.1.200';
      --sid-color: Yellow;
      --sid-chan: 3;
    }
  </style>
</head>
<body>
  <div id="main-container" style="position: relative;">
    <img src="./res/Factory.png">
    <div class="sidget" id="sid1" style="left: 202px; top: 92px;"></div>
    <div class="sidget" id="tank-level" style="--sid-chan:5;
      left: 50px; top: 505px;"></div>
  </div>
</body>
</html>
```

Listing 4.4. HTML code for visualization during design

The above code generates a visualization, which view is shown in the figure 4.6.

The above visualization presents only a small range of possibilities offered by sidgets, but creating it allows you to acquire basic skills that can be used to build more advanced web pages based on this technology.

A target visualization allowing for real-time monitoring of the production process of the entire factory, where 4 **MultiCon** devices collect data from sensors located in different areas of the factory, can be similar to the one shown in the figure 4.7.

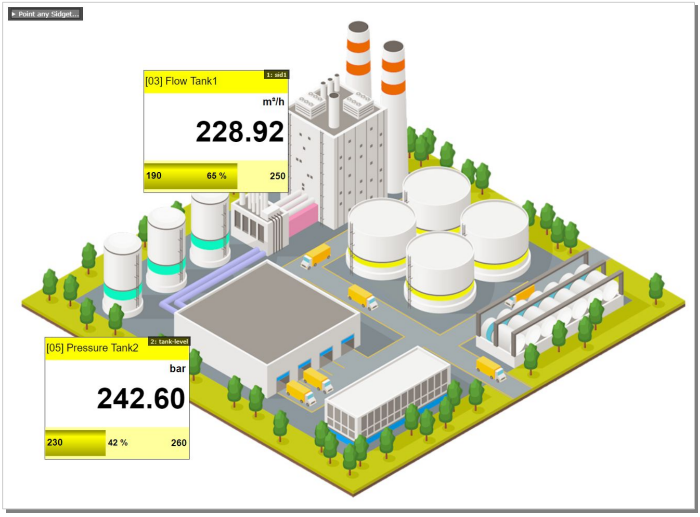


Fig. 4.6 Visualization view during design

The HTML code responsible for the above effect is presented in listing 4.5.

```
<!DOCTYPE html>
<html>
<head>
  <title>Factory</title>
  <script src="http://192.168.1.101/sidgets.js" data-mode=0></script>
  <style>
    body {
      background-color: #cccccc;
      margin: 0px; }
    .sidget {
      --sid-interval: 1;
      --sid-scaling: 60%; }
    .dev1 {
      --sid-color: #00fec3; }
    .dev2 {
      --sid-host: '192.168.1.102';
      --sid-color: #00a2fa; }
    .dev3 {
      --sid-host: '192.168.1.103';
      --sid-color: #fffc09; }
    .dev4 {
      --sid-host: '192.168.1.104';
      --sid-color: #5cbcce; }
    .chart {
      --sid-scaling: ''; /*reset scaling*/
      width: 200px;
      height: 130px; }
    #main-container {
      position: relative;
      border-width: 0px;
      margin: auto;
      width: 1050px; }
  </style>
</head>
<body>
  <div id="main-container">
    <img src="./res/Factory.png"/>
    <!-- Device 1 -->
    <div id="tank1-l" class="sidget dev1"       style="--sid-type:12; --sid-chan:1;                          --sid-scaling:55%;        left:7px;   top:115px;"></div>
    <div id="tank1-p" class="sidget dev1"       style="--sid-type:9;  --sid-chan:4;                                                    left:7px;   top:285px;"></div>
    <div id="tank2-l" class="sidget dev1"       style="--sid-type:12; --sid-chan:2;                          --sid-scaling:55%;        left:100px; top:62px; "></div>
    <div id="tank2-p" class="sidget dev1"       style="--sid-type:9;  --sid-chan:5;                                                    left:100px; top:232px;"></div>
    <div id="tank3-l" class="sidget dev1"       style="--sid-type:12; --sid-chan:44;                         --sid-scaling:55%;        left:195px; top:10px; "></div>
    <div id="tank3-p" class="sidget dev1"       style="--sid-type:9;  --sid-chan:4;                                                    left:195px; top:180px;"></div>
    <div id="speed"   class="sidget dev1"       style="--sid-type:8;  --sid-chan:37; --sid-color: #ff97be;    --sid-scaling:70%;        left:282px; top:10px; "></div>
    <!-- Device 2 -->
    <div id="flow"    class="sidget dev2"       style="--sid-type:11; --sid-chan:41; --sid-color: #b3b5ff;    --sid-scaling:70%;        left:7px;   top:550px;"></div>
    <div id="bootles" class="sidget dev2"       style="--sid-type:10; --sid-chan:30;                          --sid-scaling:70%;        left:50px;  top:635px;"></div>
    <!-- Device 3 -->
    <div id="oxygen"  class="sidget dev3 chart" style="--sid-type:3;  --sid-chan:35; --sid-color: #ff6e05;                              left:615px; top:10px; "></div>
    <div id="ph"      class="sidget dev3 chart" style="--sid-type:3;  --sid-chan:27; --sid-color: #ff6e05;                              left:825px; top:10px; "></div>
    <div id="temp1"   class="sidget dev3"       style="--sid-type:4;  --sid-chan:20;       --sid-scaling:''; width:70px; height:130px; left:740px; top:150px;"></div>
    <div id="temp2"   class="sidget dev3"       style="--sid-type:4;  --sid-chan:21;       --sid-scaling:''; width:70px; height:130px; left:815px; top:165px;"></div>
    <div id="temp3"   class="sidget dev3"       style="--sid-type:4;  --sid-chan:22;       --sid-scaling:''; width:70px; height:130px; left:890px; top:185px;"></div>
    <div id="temp4"   class="sidget dev3"       style="--sid-type:4;  --sid-chan:23;       --sid-scaling:''; width:70px; height:130px; left:965px; top:210px;"></div>
    <!-- Device 4 -->
    <div id="temp"    class="sidget dev4"       style="--sid-type:1;  --sid-chan:39;                                                   left:745px; top:640px;"></div>
    <div id="humidit" class="sidget dev4"       style="--sid-type:2;  --sid-chan:24;                                                   left:900px; top:545px;"></div>
  </div>
</body>
</html>
```

Listing 4.5. HTML code of the target visualization monitoring the production process

4.7. TEXT SIDGETS

4.7.1.Introduction

The main goal of Sidget objects is to simplify the process of creating visualizations by the user as much as possible. The library of graphic objects from which the user can choose is quite rich.

However, there are situations where the user may need to optimize the visualization page in order to display much more information, display them in a way more suited to the structure and color of the target web site or even design his own objects showing only specific data obtained from devices. In response to such needs, a special type of sidget has been designed – a text sidget.

*Support of text sidget has been added to the **MultiCon Sidgets** engine in the **5.14** firmware version of **MultiCon** devices. If the device has a lower firmware version, then if you want to use the above functionality, you must first update the device software to the latest available version.*

4.7.2.Declaration

A text sidget is specified as a type `0` named `Text` (see the `--sid-type` parameter in the chapter 4.4). This type of object follows the same parameter declaration rules as other sidget types, with a few exceptions for appearance-oriented parameters, which are not present in the text sidget. Information about the lack of support for a given parameter is placed in its description.

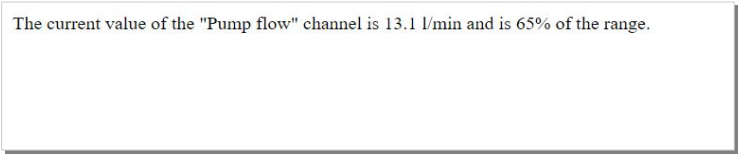
The main difference to the other - pre-designed sidgets - the text sidget requires specify additional HTML code in the body of the container tag with the variables selected by the user. An example declaration is shown on listing 4.6.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <script src="http://192.168.1.100/sidgets.js"></script>
5 </head>
6 <body>
7   <div class="sidget" style="--sid-type:0; --sid-chan:1;">The current value of
  the "{ChName}" channel is {Value} {ChUnit} and is {Value;%} of the range.</div>
8 </body>
9 </html>
```

Listing 4.6 HTML code showing how to declare a text sidget

In line 7 of above listing, inside the `<div>` container, you can notice the occurence of special tags like `{ChName}`, `{Value}`, `{ChUnit}`, `{Value;%}`. These are predefined variables where in place of their the sidget engine automatically inserts the appropriate result values.

The above code creates a visualization presented in Figure 4.8.



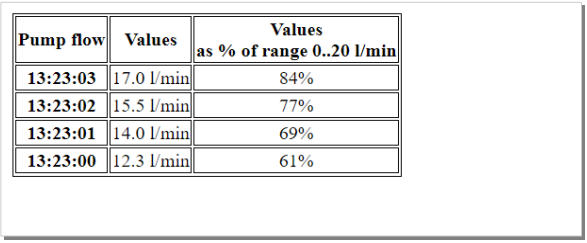
Rys 4.8 The appearance of the visualization using a text sidget

As can be seen, the text displayed in the web browser is default formatted. Nothing prevents the user from using any HTML code inside the sidget container that introduce additional formatting of the content. For example of making a text sidget more useful, see listing 4.7.

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://192.168.1.100/sidgets.js"></script>
  <style>
    table, th, td { border: 1px solid black; text-align: center;}
  </style>
</head>
<body>
  <table class="sidget" style="--sid-type:0; --sid-chan:1;">
    <tr> <th>{CN}</th><th>Values</th><th>Values<br>as % of range {CLL}{CHL} {CU}</th></tr>
    <tr> <th>{Time;HH:mm:ss}</th>   <td>{Value} {ChUnit}</td>   <td>{Value;%}</td> </tr>
    <tr> <th>{Time-1;HH:mm:ss}</th> <td>{Value-1} {ChUnit}</td> <td>{Value-1;%}</td> </tr>
    <tr> <th>{Time-2;HH:mm:ss}</th> <td>{Value-2} {ChUnit}</td> <td>{Value-2;%}</td> </tr>
    <tr> <th>{Time-3;HH:mm:ss}</th> <td>{Value-3} {ChUnit}</td> <td>{Value-3;%}</td> </tr>
  </table>
</body>
</html>
```

Listing 4.7 HTML code showing the ability to format a text sidget

The above code creates a visualization presented in Figure 4.9.



Pump flow	Values	Values as % of range 0..20 l/min
13:23:03	17.0 l/min	84%
13:23:02	15.5 l/min	77%
13:23:01	14.0 l/min	69%
13:23:00	12.3 l/min	61%

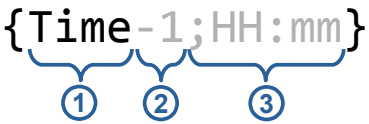
Rys 4.9 The appearance of the visualization using text sidget formatted by additional HTML

In above examples of text sidget declarations, many variables were used. Meaning of each of them is described in chapter 4.7.3.

4.7.3.Variables

Each variable that can be declared in the body of a text sidget consists of curly braces, its name and additional elements.

General structure of the variable:



(1) Variable name (required element)
(2) Position in the buffer of previous values (additional element)
(3) Variable parameter (additional element)

The variable name may be given in full or shortened form and it is a required element. The list of acceptable variable names, their meaning and examples of declaration are presented in the table below.

Variable name	Short name	Variable description	Buffer	Para-meters	Source of obtaining	Examples of declaration with the result
Host	**H**	*IP address or hostname of the device*	-	-	**--sid-host**	**{H}** → 192.168.3.97 **{H}** → dev1.multicon.com
Color	**C**	*The leading color of the sidget*	-	-	**--sid-color**	**{Color}** → DeepSkyBlue
HiColor	**HC**	*The color of the sidget fragment specified for the upper channel limit*	-	-	**--sid-hicolor**	**{HC}** → Red
LoColor	**LC**	*The color of the sidget fragment specified for the lower channel limit*	-	-	**--sid-locolor**	**{LoColor}** → #fffc09;
Interval	**I**	*The refresh interval of the values in the channels expressed in seconds*	-	-	**--sid-interval**	**{I}** → 300
ChNo	**CN**	*Channel number*	-	-	**--sid-chan**	**{ChNo}** → 1
ChName	**CNA**	*Channel name*	-	-	**device**	**{CNA}** → Pump flow
ChUnit	**CU**	*Unit of value in the channel*	-	-	**device**	**{ChUnit}** → l/min
ChHiLimit	**CHL**	*High limit of value in the channel*	-	-	**device**	**{CHL}** → 20
ChLoLimit	**CLL**	*Low limit of value in the channel*	-	-	**device**	**{ChLoLimit}** → 0
Value	**V**	*Channel value. This variable may show the last obtained or one of the earlier values. It can also be expressed as a percent of the range* **{ChLoLimit}**…**{ChHiLimit}**.	**YES**	**%**	**device**	**{Value}** → 15.6 **{V;%}** → 51% **{V-10}** → 12.3 **{Value-1;%}** → 46%
Time	**T**	*Occurrence time of the value in the channel. This variable may show the refresh time of the last or one of the previous values. The time format can be freely adjusted.*	**YES**	time format	**web**	**{Time}** → 2021-07-18 13:52:37 **{T-60}** → 2021-07-18 13:51:37 **{T-30;HH:mm:ss}** → 13:52:07
BuffPeriod	**BP**	*Information about the time interval for which the* **{ValueMin}**, **{ValueMax}**, **{ValueAvg}** *values are provided. This period is calculated based on the assumed* **{Interval}** *parameter and the internal buffer limit of 600 samples. The declaration of a variable with the information limiting the number of samples allows to calculate a different time period.*	**YES**	-	**web**	**{BP}** → 10m **{BP-10}** → 10s **{BuffPeriod-300}** → 5m
ValueMin	**VMI**	*The minimum value in the channel that occurred in the* **{BuffPeriod}** *period. It is possible to return the minimum value that occurred in a shorter period. It can also be expressed as a percent of the range* **{ChLoLimit}**…**{ChHiLimit}**.	**YES**	**%**	**web**	**{VMI}** → 5.0 **{VMI;%}** → 10% **{VMI-10}** → 5.3 **{VMI-100;%}** → 20%
ValueMinTime	**VMIT**	*Time of the* **{ValueMin}** *occurrence in the* **{BuffPeriod}** *period. It is possible to return the time of the minimum value in a shorter period. The time format can be freely adjusted.*	**YES**	time format	**web**	**{VMIT}** → 2021-07-18 13:50:25 **{VMIT-60}** → 2021-07-18 13:52:03 **{VMIT-30;HH:mm:ss}** → 13:52:16
ValueMax	**VMA**	*Maximum value on the channel that occurred during* **{BuffPeriod}**. *It is possible to return the maximum value that occurred in a shorter period. It can also be expressed as a percent of the range* **{ChLoLimit}**…**{ChHiLimit}**.	**YES**	**%**	**web**	**{ValueMax}** → 20.0 **{VMA;%}** → 110% **{VMA-10}** → 16.2 **{VMA-100;%}** → 95%
ValueMaxTime	**VMAT**	*Time of the* **{ValueMax}** *occurrence in the* **{BuffPeriod}** *period. It is possible to return the time of the maximum value in a shorter period. The time format can be freely adjusted.*	**YES**	time format	**web**	**{VMAT}** → 2021-07-18 13:50:27 **{VMAT-60}** → 2021-07-18 13:52:06 **{VMAT-30;HH:mm:ss}** → 13:52:27
ValueAvg	**VA**	*The mean value of the measurements that occurred during the* **{BuffPeriod}** *period. It is possible to return the average value calculated on the basis of a shorter period. It can also be expressed as a percent of the range* **{ChLoLimit}**…**{ChHiLimit}**.	**YES**	**%**	**web**	**{ValueAvg}** → 15.5 **{VA;%}** → 48% **{VA-10}** → 13.3 **{ValueAvg-300;%}** → 51%

For variables that represent time it is possible to pass as a parameter the string with the time format specifiers. These specifiers and their meanings are listed in the table below.

Format Specifier	Description
D	The day of the month, from 1 through 31
DD	The day of the month, from 01 through 31
DDD	The abbreviated name of the day of the week
DDDD	The full name of the day of the week
M	The month, from 1 through 12
MM	The month, from 01 through 12
MMM	The abbreviated name of the month
MMMM	The full name of the month
Y	The year, from 0 to 99
YY	The year, from 00 to 99
YYY	The year, with a minimum of three digits
YYYY	The year as a four-digit number
h	The hour, using a 12-hour clock from 1 to 12
hh	The hour, using a 12-hour clock from 01 to 12
H	The hour, using a 24-hour clock from 0 to 23
HH	The hour, using a 24-hour clock from 00 to 23
m	The minute, from 0 through 59
mm	The minute, from 00 through 59
s	The second, from 0 through 59
ss	The second, from 00 through 59
t	The first character of the am/pm designator (lower case)
tt	The am/pm designator (lower case)
T	The first character of the AM/PM designator (upper case)
TT	The AM/PM designator (upper case)
K	Time zone information
z	Hours offset from UTC, with no leading zeros
zz	Hours offset from UTC, with a leading zero for a single-digit value
zzz	Hours and minutes offset from UTC

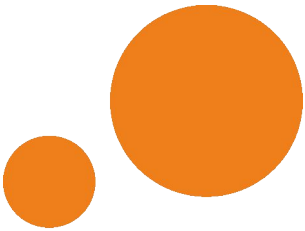
5. COMPATIBILITY

MultiCon Sidgets engine uses HTML 5, CSS 3 and JavaScript ES6 technologies. For the proper functioning of these technologies care good designed browser engines. For the purpose of ensuring the proper operation of the designed visualizations and the security of operating systems, it is recommended to use web browsers updated to the latest versions.

The correctness of sidget engine operation has been verified by the manufacturer on selected browsers in specific versions. The manufacturer guarantees its proper operation only on the versions of browsers verified by the manufacturer. However, there is a small risk that newer versions of these programs will not work or will not work properly with the ***MultiCon Sidgets*** *engine.*

Verified versions of web browsers:

Browser name	Browser version	Operating system	Does it work?
DAQ Manager	to 1.9	Windows XP-10	**NO**
DAQ Manager	from 1.10	Windows XP-10	**YES** (external browser)
Microsoft Internet Explorer	11.0	Windows XP-10	**NO**
Microsoft Edge	44.18362	Windows 10	**YES**
Google Chrome	79.0	Windows	**YES**
Mozilla Firefox	71.0	Windows	**YES**
Google Chrome	78.0	Android 9	**YES**
Google Chrome	87.0	Android 9	**YES**
Google Chrome	91.0	Android 9 i 10	**YES**
Mozilla Firefox	87.0	Android	**YES**
Mozilla Firefox	88.1	Android 9	**YES**
Apple Safari	12.1	iOS 12.4.1	**YES**



SIMEX Sp. z o.o.
ul. Wielopole 11
80-556 Gdańsk
Poland

tel.: (+48 58) 762-07-77
fax: (+48 58) 762-07-70

http://www.simex.pl
e-mail: info@simex.pl