Signal Processor

(SW version 3.0.0.0)

USER GUIDE

Updated: February 2021
www.safibra.cz

TABLE OF CONTENTS

1 BEFORE YOU BEGIN

1.1 About the Guide

This guide is intended for anyone who is responsible for using the “Signal Processor”, the universal data analyzing and processing software for SAFIBRA’s sensing and monitoring systems. This person must be aware of the risks associated with using this software and must be qualified and trained.

1.2 Disclaimer

The information in this guide has been carefully checked and is believed to be accurate. However, SAFIBRA, s.r.o., assumes no responsibility for any inaccuracies that may be contained in this guide. In no event will SAFIBRA, s.r.o., be liable for direct, indirect, special, incidental, or consequential damages resulting from any defect or omission in this guide, even if advised of the possibility of such damages. In the interest of continued product development, SAFIBRA, s.r.o., reserves the right to make modifications to this guide and the products it describes at any time, without notice or obligation.

1.3 Warranty

All SAFIBRA´s systems are warranted against defective materials and workmanship for one year from the date of shipment. Our obligation is limited to repairing or replacing parts and components which prove to be defective during the warranty period and which are returned to the factory, transportation charges prepaid. We are not liable for any consequential damages and costs.

Repairs inside the instrument must be performed by SAFIBRA, s.r.o., or its representatives only. The above warranty may, therefore, be rendered null and void in the event of an unauthorized opening. If you need to claim against the warranty, please contact SAFIBRA and we’ll guide you through the process.

We reserve the right to make improvements to our products at any time without incurring any liability to purchasers of earlier models.

2 PRODUCT OVERVIEW

2.1 Product Description

The “Signal Processor” (hereinafter referred to as “SigProc”) is a universal data analyzing and processing software developed for SAFIBRA’s sensing and monitoring systems. This powerful tool is designed for acquisition of raw data and consequent data processing.

It allows the use of filtering, Fast Fourier Transformation (FFT), signal visualization, definition of alarm statuses, signal logic, alarm logic, signal sources pooling, data storage in databases, data streaming, data messages sending and third party integration. The "SigProc" also enables preparing data for additional processing using AI, data visualization and storage in databases or cloud.

Working with "SigProc" is intuitive and simple. Individual tasks are solved in configurations that consist of predefined modules. Each module represents computational tasks for processing input data. The total list counts 64 modules.

Please, be aware that ”SigProc" in the standard version is not equipped with pre-installed configurations. The user creates all configurations himself, from the modules the ”SigProc" contains.

”SigProc" is always delivered together with the “Specification sheet” which contains important information about system and login details. We highly recommend keeping this document safe and sharing it only with people involved.

Features:

- web-based interface to manage configurations and modules
- incorporated in SAFIBRA's monitoring systems, such as “FBGuard” or “PeriGuard”
- data processing and analysis
- graphical interpretations
- signal thresholding
- third-party integration
- sophisticated algorithms, artificial intelligence (AI)
- data streaming, real-time output

2.2 HW and SW Requirements

Use of the “ProcessGuard” workstation is required to ensure a trouble-free installation, commissioning and data flow. This powerful machine comes with the pre-installed ”SigProc" software. The ”SigProc" is also a part of each SAFIBRA’s “Measurement System”.

To ensure a smooth working experience when using ”SigProc" on your own “ProcessGuard” workstation, please see the minimal HW requirements.

The minimum HW requirements on the “ProcessGuard” workstation are:

- **CPU:** Intel 4 Core Xeon (3.4 GHz multi core)
- **RAM:** 16 GB
- **HDD:** 1 000 GB (2x in RAID - recommended)
- **Network Interface Controller (NIC):** 1 GbE port LOM
- **Compatibility:** Linux CentOS

The “SigProc" has a web-based user interface, making it available remotely over the network. There is no need to install specialized software on client computers, just an internet browser is necessary (“Mozilla Firefox” is recommended).

2.3 Terminology Used

While it is not essential to read this section before proceeding to other sections of the guide, it may help to clarify the meanings of the most common terminology frequently used in this guide.

- **Module:** The basic building block of which the modular ”SigProc" system is composed. Module represents a single step of a configuration. Modules can be connected together - output from module1 is being input to module2.
- **Module group:** It is, in fact, a folder which groups several modules together. Modules can be connected with other modules inside of the module group as well as with modules outside of the module group. Each group is represented by the “folder” icon before the module group name together with a “+” or “-” sign that allows collapsing/expanding the whole group.

Figure 2.1 Module group

- **How to select module, module group or configuration:** Before you run any action with module, module group or configuration, the preferred element has to be selected. To make a selection, just click on the element by the “left mouse” button. In case of modules and module groups it is also possible to select more than one element using Ctrl or Shift key.
- **Configuration:** It is created by connecting two or more modules together. Each configuration is basically a simple program which says how the modules are combined or arranged one after the other.
- **Configuration string:** It displays the configuration or part of the configuration written as a code. It is used when copying or exporting/importing configuration.
- **Input:** Represents data that are sent to a module for processing.
- **Output:** Represents data that are processed by and sent out from a module.
- **Parameter:** It defines properties of a module. Every module has common (type, name, inputs, enabled) and specific parameters that define how the module processes data.

- **Data file:** The file that contains data set in a structured manner. The file can be used as a source of readouts or it can be used as a storage for processed readouts. For further information see chapter "Data File Example".
- **Readout:** The readout is a pair that consists of a timestamp and a value. The readouts are produced by sensors in the “MeSyCo” or by modules in the “SigProc”.
- **Vector:** The vector is a readout that consists of a timestamp and more than one value.
- **Drag and drop:** User selects a virtual object by "grabbing" it and dragging it to a different location.

3 LOGIN

3.1 Login Page

To start ”SigProc", open a web browser (“Mozilla Firefox” is recommended) and enter the IP address into the address bar including the protocol (https://) and the port (8888). For example https://10.23.23.117:8888.

If connecting for the first time, adding a security exception is required. Click the "Advanced..." button and then "Accept the Risk and Continue".

Note: SSL stands for “Secure Sockets Layer”, a global standard security technology that enables encrypted communication between a web browser and a web server. It is utilized by millions of online businesses and individuals to decrease the risk of sensitive information (e.g., credit card numbers, usernames, passwords, emails, etc.) being stolen or tampered with by hackers and identity thieves. In essence, SSL allows for a private “conversation” just between the two intended parties.

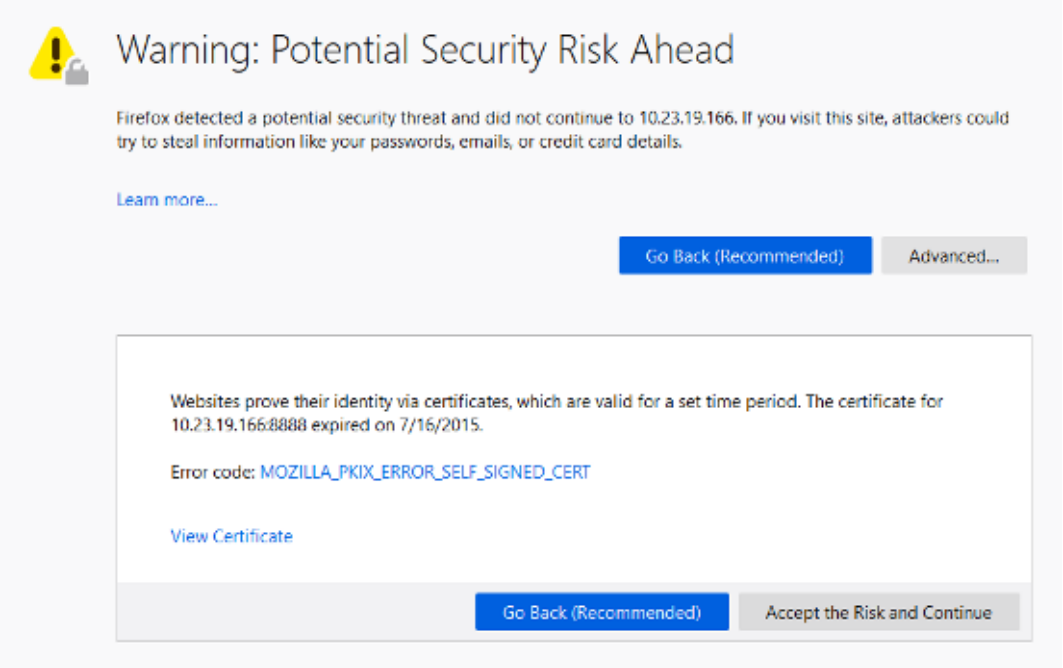


Figure 3.1 Adding a security exception when logging in for the first time

On the “Login” page, the ID of the measuring unit where the ”SigProc" is running, e.g. ProcessGuard_14 (see Figure 3.2), is always indicated. Enter your credentials in the “Login” form and then click the "Login" button.

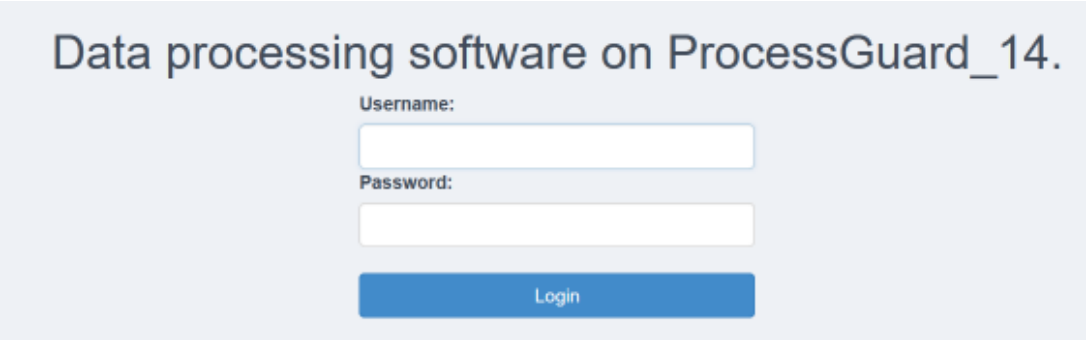


Figure 3.2 The “Login” page

3.2 Main Panel

When logging into the ”SigProc" there are three sections available on the “Main panel” (dark blue buttons on top of the screen):

- **System:** This section provides a summary of various system information (disk usage, git and licence information etc.) and enables system control (user management, file browser, access to system and debug logs, etc.).
- **Configurations:** This section allows users to create configurations from modules that are available in their ”SigProc" version (depending on the license type).
- **Running Configurations:** This section shows all configurations currently running. When you click on the “Running configurations” button, the dropdown menu with all running configurations is displayed. When you select the particular configuration from the dropdown menu, the “Dashboard” page is displayed.

Below the “Main panel” the ID of the measurement unit where the ”SigProc" is running is displayed. Next to the ID is a visible headline of the page where you are actually working, e.g. “Configurations”.

Figure 3.3 The “Main panel”

4 SYSTEM

After logging in, click on the "System" button on the top left corner of the screen.

On this page there is the "System menu" allowing global controls and the "System summary" displaying various information.

Figure 4.1 The “System” page

4.1 System Summary

Version

It shows the current version of the "SigProc" software and the release date of this version.

Disk Usage

This category is displayed only if the “Automatic file delete parameters” option is enabled (see Figure 4.1 above). It displays current disc usage.

Automatic File Delete Parameters

If enabled, the file eraser automatically erases old data and log files when the disk usage exceeds a specified threshold.

Files are deleted in the following directories:

- */home/sigproc/data*
- */home/sigproc/log*
- */home/sigproc/log_debug*

Other categories, specifically "Database message logging", "Git info" and "License information", contain information about the system version and licence. Those are useful mainly for debugging the system.

4.2 System Menu

4.2.1 Data File Browser

The "Data file browser" allows viewing .csv and .bin data. Data files created by the "File writer" module are stored in the directory named */home/sigproc/data/*. Each configuration has its own subdirectory named after the name of the configuration, i.e. */home/sigproc/data/<configuration_name>/<module_name>/<date>/<module_name>-<date_and_time>.<csv_or_bin>.*

Note: File browser cannot open other formats than mentioned in the paragraph above.

The "Left menu" buttons provide actions you can perform for each file (see Figure 4.2):

Show file: Displays selected file contents in a plot on the right side of the file tree. To display multiple files in a single plot use the "Show file" button. Up to four files can be displayed in one plot. In order to display more files, select another file and add this file by clicking on the "Add file" button below the plot.

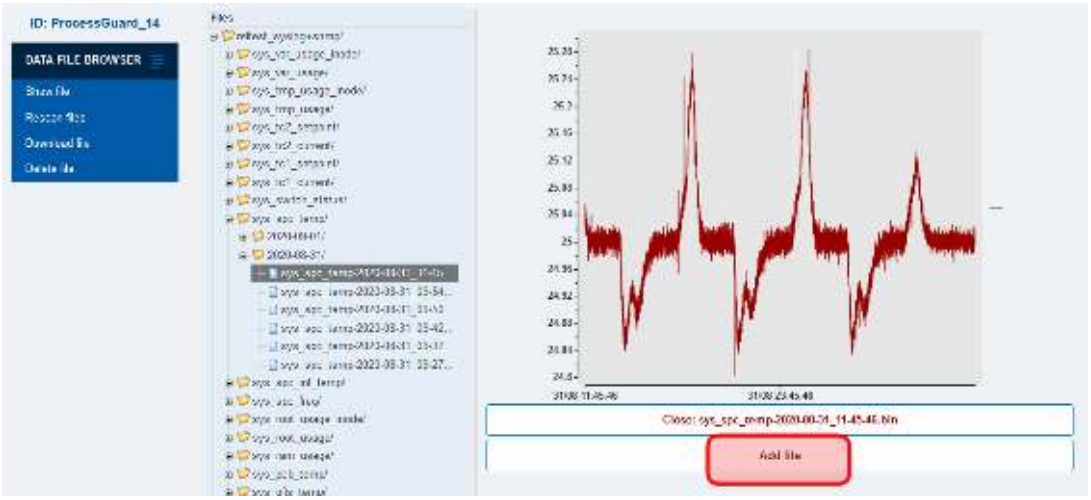


Figure 4.2 File browser - the "Show file" function

Rescan files: Refreshes all files and updates the file tree with the most up to date data files.

Download file: Opens a dialog box to download the selected file.

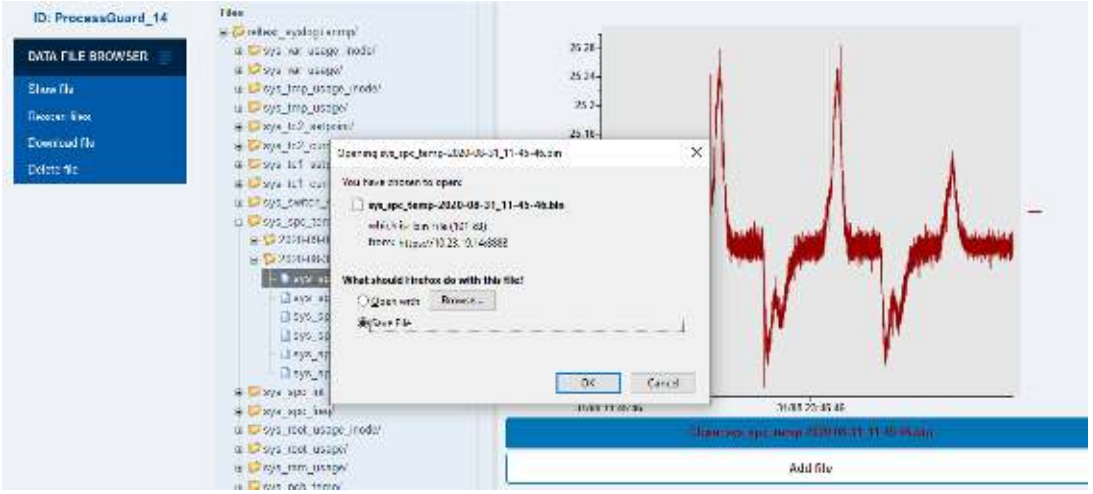


Figure 4.3. File browser - the ”Download file” function

Delete file: Opens a dialog box to confirm the permanent deletion of the selected file.

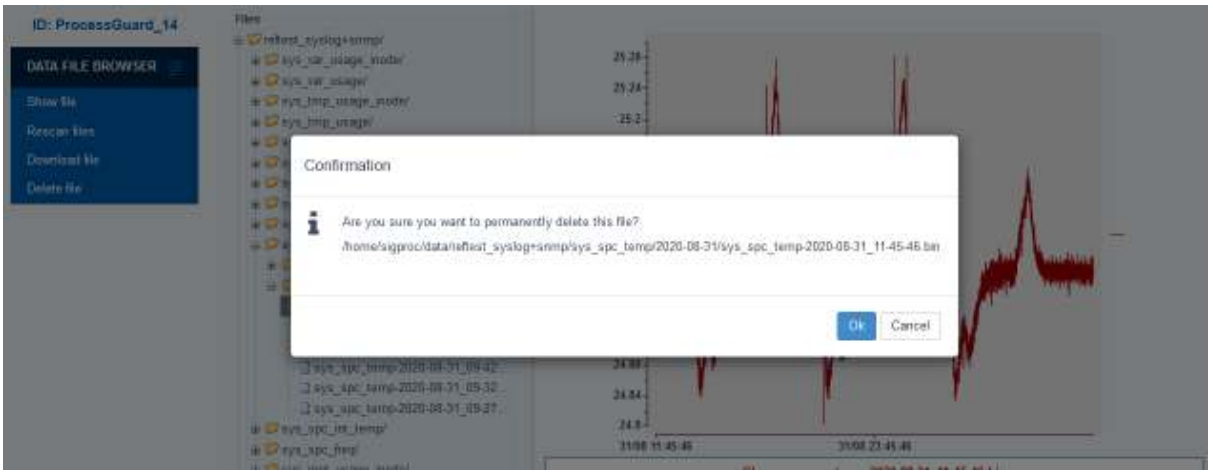


Figure 4.4. File browser - the “Delete file” function

4.2.2 System Log & Debug Log Browsers

The “System log browser” and “Debug log browser” allow the User to open log files generated by "SigProc". The “System log” contains information for Users whereas the “Debug log” contains information that is useful for developers. Log files in the file tree are sorted from the newest on top.

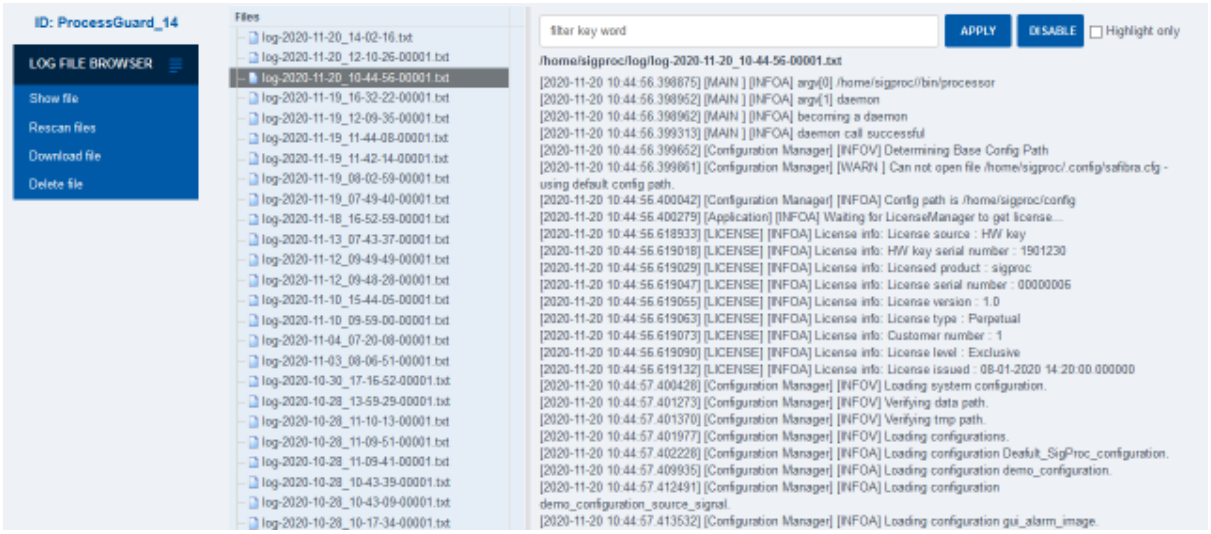


Figure 4.5 Log browser

The left menu buttons provide the actions you can perform for each file (see Figure 4.5):

Show file: Opens the selected file (also double-clicking a file has the same result).

Rescan files: Rescans folder and updates list of files.

Download file: Downloads the selected file.

Delete file: Deletes the selected file.

It is possible to search by keywords using the filter box above the log file that is currently shown. Filtering options are:

- **Text field:** Enters filtered keyword (case sensitive, regular expressions not accepted).
- **Apply:** Applies filter.
- **Disable:** Removes current filter.
- **Highlight only:** If selected lines containing filtered keywords are highlighted.

4.2.3 Users

The “Users” section allows viewing and managing user accounts. Every user has an assigned role which determines his/her permissions.

There are three default user accounts in "SigProc" (see Figure 4.6). Key differences for each user type are explained below:

- **Admin:** Admin can see and edit everything.
- **User:** User has read and write rights. User can see or edit only the configurations defined by Admin (see Figure 4.7). User has limited access to the “System” section (“System summary” page, both “Log browsers” and “Database manager”) and cannot change the settings.
- **Guest:** Guest can only see the running configurations defined by admin. Guest can access the “System” section to see the “System summary” page and the “Data file browser” only.

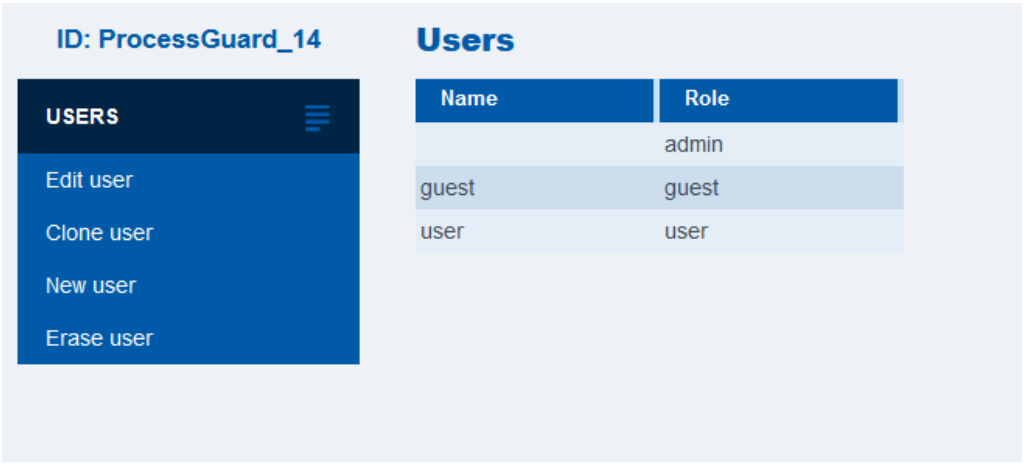


Figure 4.6 Users

Edit user: "Edit user" section allows changing his/her password, role and level of access to each configuration. The name of the user has to be set when the user is being created. In order to save or cancel all changes, click on the "Save" or "Cancel" button. Afterwards, you'll be able to access the top menu bar again.

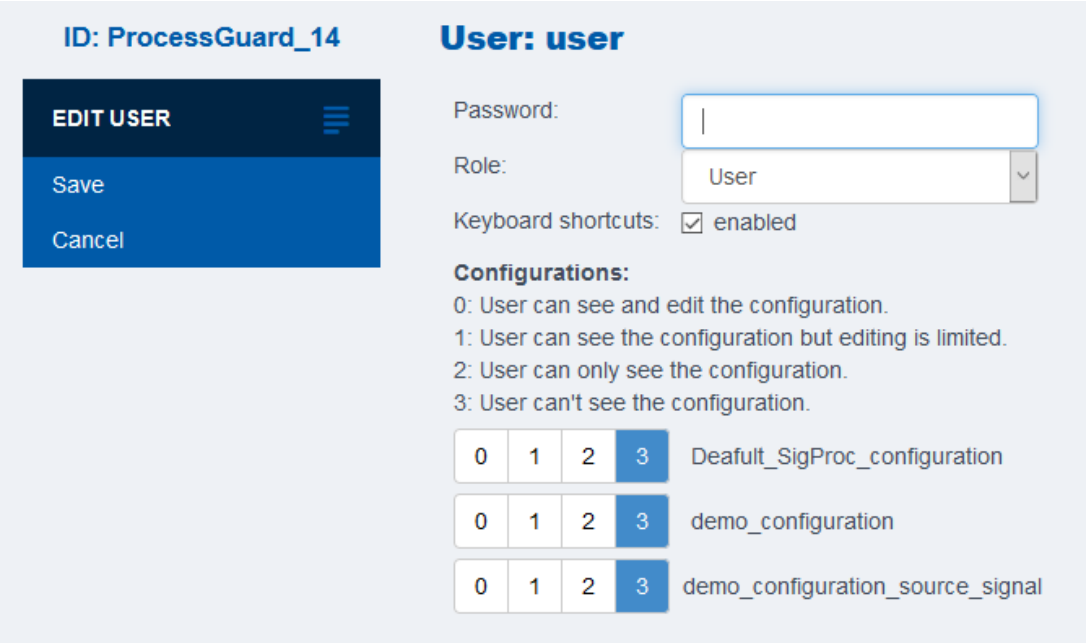


Figure 4.7 Edit user

Setting access levels for every user enables you to choose permissions for what they can see and do in "SigProc". Access level is set up for every configuration independently. The following access levels are defined (see Figure 4.7):

- **0:** User can see and edit the configuration.
- **1:** User can see the configuration but editing options are limited.
- **2:** User or Guest can only see the configuration.
- **3:** User or Guest can't see the configuration.

Note: Admin can always see and edit all configurations (without setting up any access level). Users with the "User" role can hypothetically see and edit all configurations (in case their

profile is set this way). Otherwise, the user profile can be set up to see only some configurations and to edit other configurations. Furthermore, some configurations can even be completely inaccessible (not to see, nor edit) for particular users.

The “Keyboard shortcuts” tick box allows or denies using keyboard shortcuts while setting up the "SigProc" configuration for each user. If this feature is enabled, user will see a tooltip. For more information about keyboard shortcuts see the “Keyboard shortcuts” section in this guide.

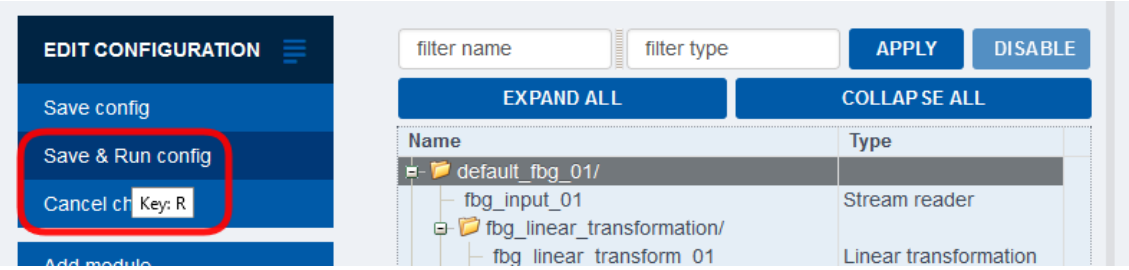


Figure 4.8 “Keyboard shortcuts” feature enabled

Clone user: In order to create a similar user profile as an already existing user, select the particular user that should be cloned and use the “Clone user” option from the menu on the left side. Enter the user’s name and in the next step set up his/her password. Afterwards, the new user will appear on the list of all users.

New user: In order to create a new user click on the “New user” button from the menu on the left side. Enter the user’s name and in the next step set up a password. New users are always created with the default “Guest” role. Changing the role and defining access levels can be done afterwards in the “Edit user” section.

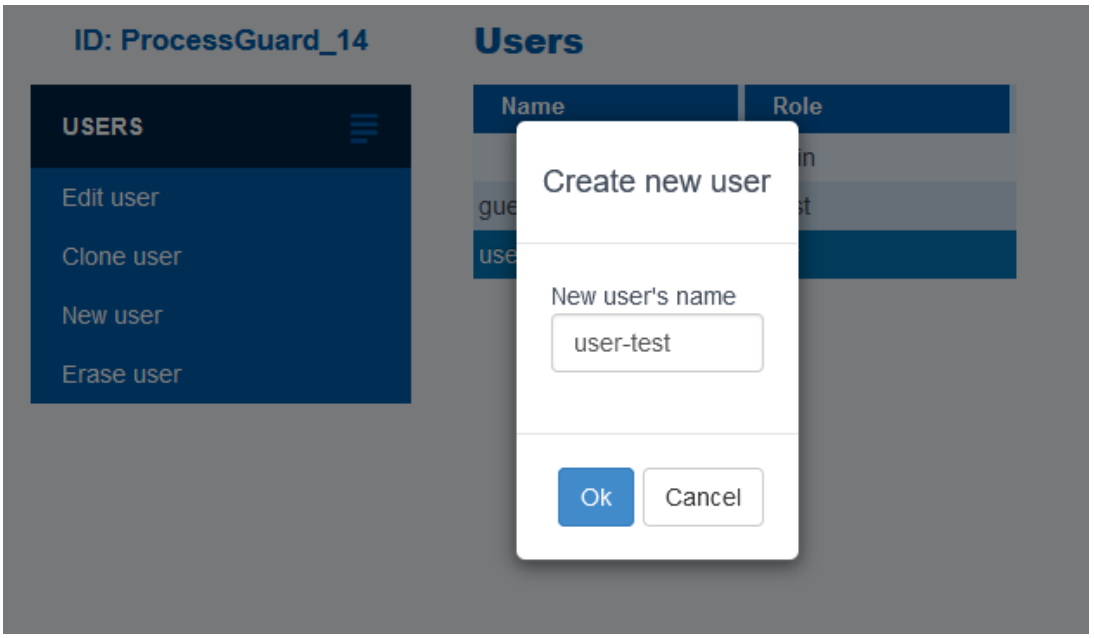


Figure 4.9 Create a new user

Erase user: To remove the user profile, select the particular user and use the “Erase user” option from the menu on the left side. This selection has to be confirmed afterwards, as deleting the user is permanent and cannot be undone.

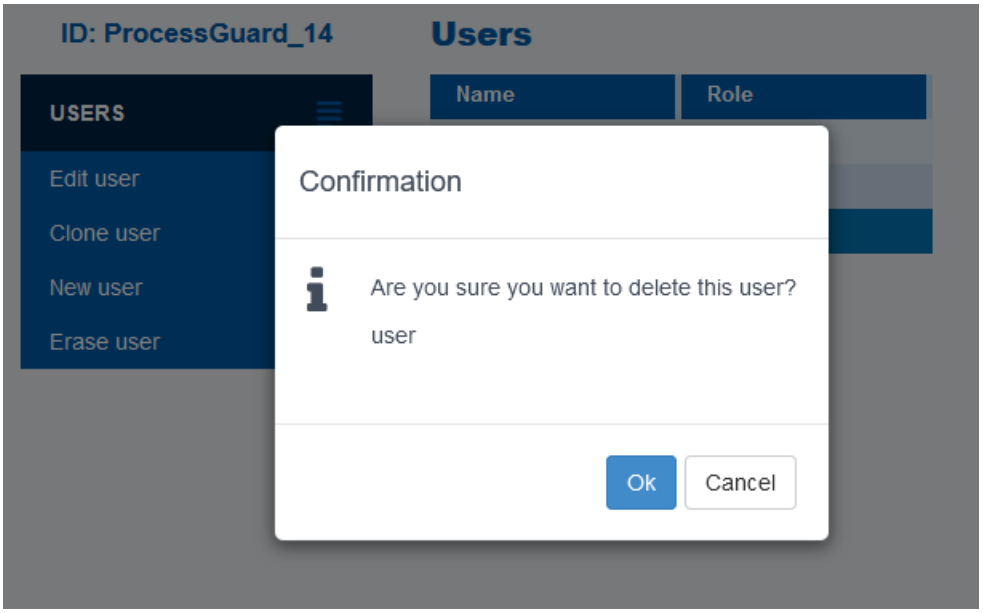


Figure 4.10 Erase user

4.2.4 Database Manager

Database manager allows to create database configuration if streaming data into the database is needed. "SigProc" supports either “InfluxDB” or "PostgreSQL" databases.

Database connection shows the list of already existing database configurations. It is possible to edit existing database configuration in the “Database connection edit” window on the right hand side of the list. To open the database configuration for editing just select it from the list.

Note: All fields are editable at any time except for the *connection_id* field. The *connection_id* has to be unique and cannot be changed later.

To create a new database configuration select the “Add InfluxDB” or “Add PostgreSQL” button from the left hand side menu. Then in a new window define the name of the database configuration and hit “Ok”. Afterwards, this new configuration will appear on the list of all configurations and it is possible to set up all necessary fields.

Figure 4.11 Database manager

Explanatory notes to Database manager:

- **db_type**: Type of database connection.
- **connection_id**: Connection identifier. It is set up when a new database connection is being created. It is not possible to change it later. The *connection_id* is used in all modules to define which DB connection should be used.
- **server_address**: Server URL or IP address. It is recommended to contact your DB manager to receive exact information about the database you want to use.
- **server_port:** TCP port on which the server listens.
- **user**: Database user name.
- **password**: Database password.
- **db_name**: Name of the database where data should be sent to.
- **sending_period**: Period of sending all buffered data in sec.
- **testing**: Verbose logging. When enabled it sends more information to the log. Useful when debugging.
- **backup_buffer_enabled**: Saves queries in the backup buffer on the disk when the database is not available.

Note: More information on how to set up and use the database connection can be found in the "InfluxDB writer module" section.

4.2.5 Network Manager

This section allows you to change the network settings of the ethernet interface (the RJ-45 connector, internet cable). Be aware that change of the network settings has a direct impact on data streaming and the entire network. Make sure the configuration is correctly set up otherwise the "SigProc" will not be accessible remotely.

Note: This topic assumes that you are familiar with network configuration and that you already know which settings to use. If not, you may need to consult your IT department or network administrator for this information.

Explanatory notes to Network manager:

- **Method**: Static IP (Network settings are defined by the parameters below.) or DHCP (Network settings are obtained automatically from the DHCP server.).
- **IP address**: Internet Protocol (IPv4) address assigned to this device.
- **Subnet prefix:** Defines a range of IP addresses available within a network (24 is a subnet prefix for subnet mask 255.255.255.0).
- **Gateway:** IP address of the gateway.
- **DNS 1:** IP address of the Domain Name System (DNS) server.
- **DNS 2:** IP address of an alternative DNS server. Optional, can be left blank.

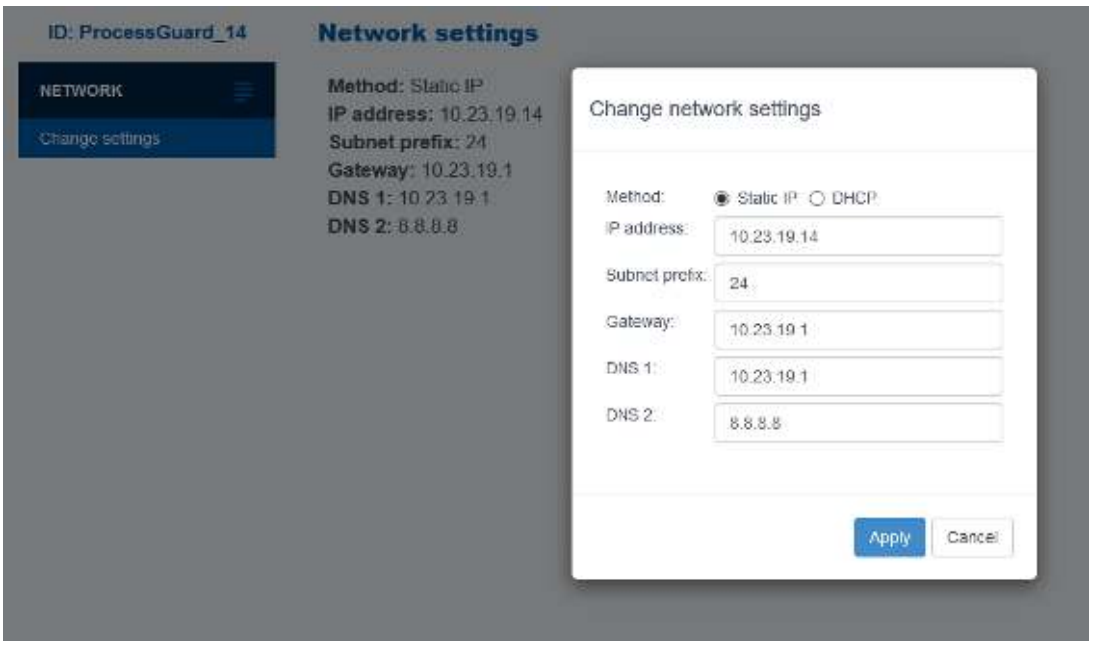


Figure 4.12 Network manager

4.2.6 Logout

The “Logout” button allows the user to log out of the "SigProc" in order to switch to another user or to simply leave. This button won’t quit the application and stop it from running.

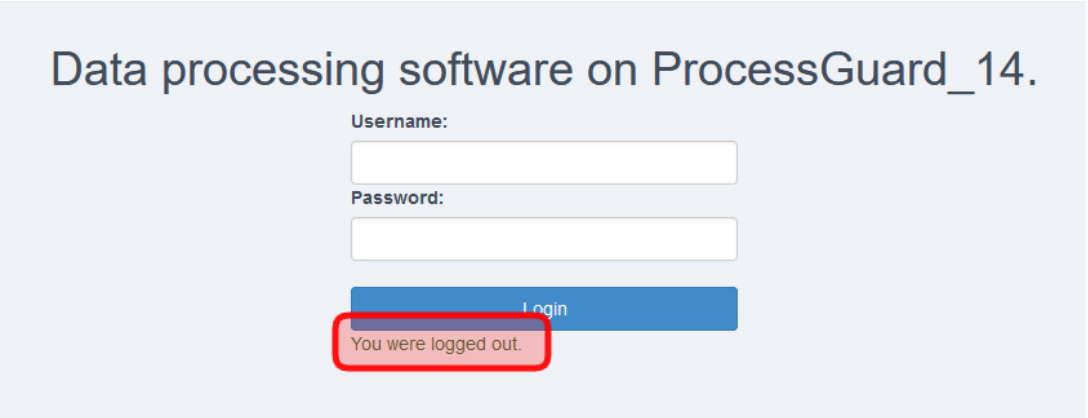


Figure 4.13 Log out confirmation

5 CONFIGURATIONS

After logging in, click on the "Configurations" button in the middle of the top of the screen. This page offers configuration overview and controls. Visibility and options depend on user privileges (for more information see the "Users" section of this guide).

Running configurations are displayed on the "Main panel" in the list of configurations and are highlighted in bold font. All configurations are saved directly to the HDD of the "SigProc" workstation (or the "Measurement System" if running on the same device) in the directory named */home/sigproc/config/configs/*.

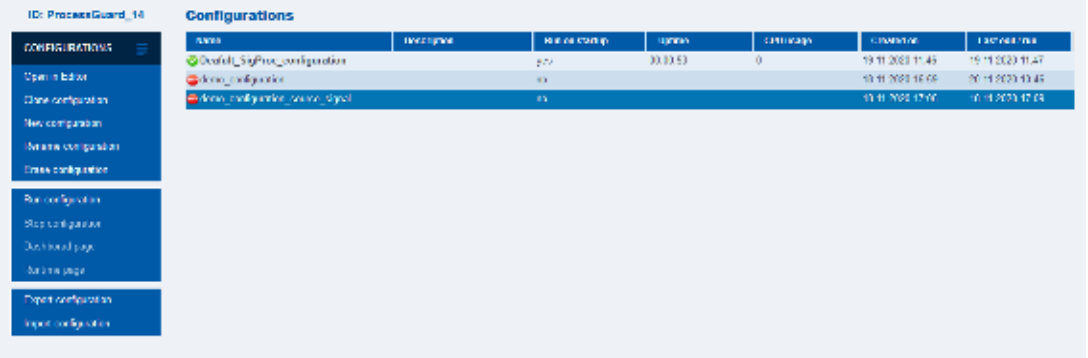


Figure 5.1 Configurations table

Explanatory notes to configurations table:

- **Indicator:** It is a red or green icon before the name of the configuration. It indicates whether the configuration contains modules which aren't included in the current "SigProc" licence. Even if there is a red sign it is possible to run the configuration but the modules without licence will be disabled.
- **Configuration:** Configuration name. It can be set up when creating or cloning the configuration. It is possible to change the name using the "Rename configuration" function.
- **Description:** Configuration description (editable field). Double click on the appropriate line in the "Description" column and start writing into the white box. Then click on the "Enter" key.
- **Run on startup:** If set up as "Yes" the configuration will start when the "SigProc" application is restarted (or the "Measurement System" is rebooted).
- **Uptime:** Actual session uptime for the running configuration since the last start.
- **CPU usage:** Current CPU usage for the running configuration.
- **Created on:** Displays date and time of creation of the current configuration.
- **Last edit / run:** Shows date and time when the configuration was run last time.

Note: Names of all running configurations are shown on the "Main panel" in the dropdown menu named "Running Configurations". It is possible to display the "Dashboard" page with the dashboard of the running configuration by clicking on its name in the dropdown menu.

The left menu buttons provide the actions you can perform for each configuration (see Figure 5.1). Some buttons can only be used when the configuration is in a certain status

(e.g. “Run configuration”, “Stop configuration” and others). Inactive buttons for currently selected configuration are displayed in grey color.

5.1 Open in Editor

The “Configuration editor” section allows you to edit the selected configuration by adding or removing new modules or module groups. In order to select the configuration for editing, click on the configuration in the list and select “Open in editor” in the left menu (see Figure 5.2) or double click on the particular configuration.

There are three sections on the “Configuration editor” page - the left menu with actions, “Configuration tree” in the central part and “Module parameters edit” on the right side.

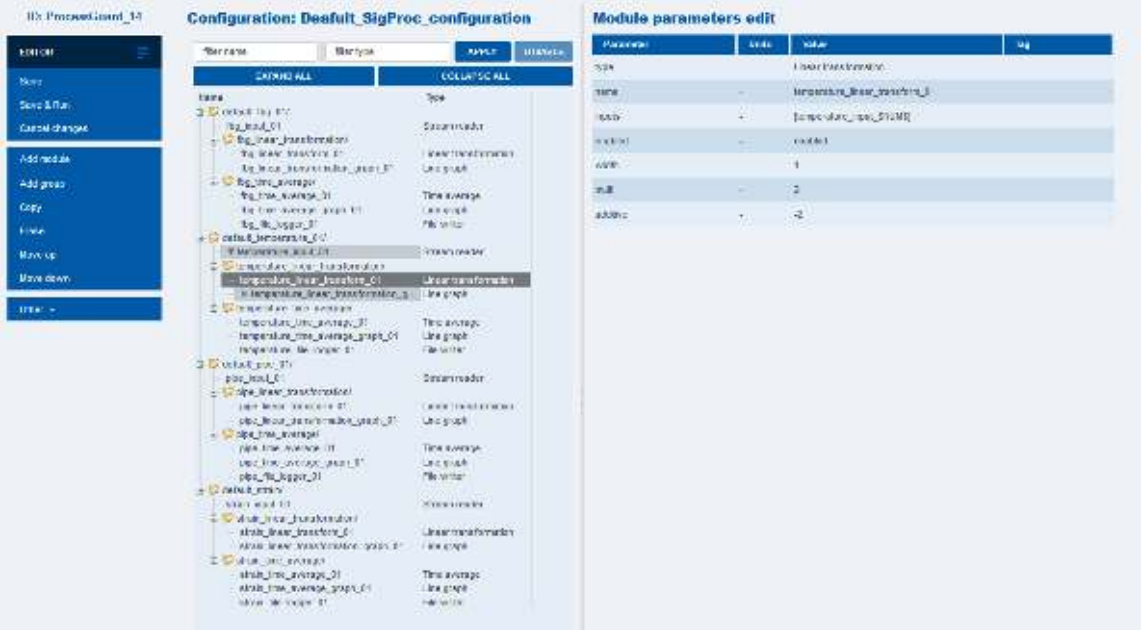


Figure 5.2. The “Configuration editor” section

5.1.1 Left Menu

The left menu buttons provide several actions that can be performed with configuration:

Save: Saves configuration changes to the file. This action needs to be confirmed in a dialog window.

Save & Run: Saves configuration changes to the file and immediately runs the edited configuration. This action needs to be confirmed in a dialog window.

Cancel changes: Discards all changes and loads the unchanged configuration from the saved file. This action needs to be confirmed in a dialog window.

Add module: To add a new module into the configuration, it first needs to be selected from the list of all modules. The dialog window allows you to filter across all licenced and unlicensed modules using a tick box, whereas modules not supported by the current

"SigProc" licence version are marked by [X] in the “Module type” column. There are three options how to filter and select the requested module (see Figure 5.3):

- Select the requested module manually from the “Module type” list.
- Select one of the categories in the “Module Category” list to specify which type of module you are searching for. Then manually select the requested module from the narrowed “Module type” list.
- Filter the requested module by using the “Module type filter” to enter a keyword that you are looking for. Afterwards, hit enter and both lists “Module category” and “Module type” will be filtered. Then select the requested module.

The “Module name” field is optional. It allows you to add the name of the module, which is then visible in the “Configuration tree” section and which can be edited in the “Module parameters edit” section. If this field remains empty, the added module will be populated to the “Configuration tree” section as a “new_item”.

After selecting the requested module, hit one of the buttons in the bottom left corner (“Insert below/above/inside”). As the names of the buttons indicate, it is possible to add the module below or above the module or module group you have selected in the “Configuration tree” section or to add it inside of the selected module group. This module or module group are selected prior to clicking the “Add module” button and are marked by dark grey color (see figure 5.2).

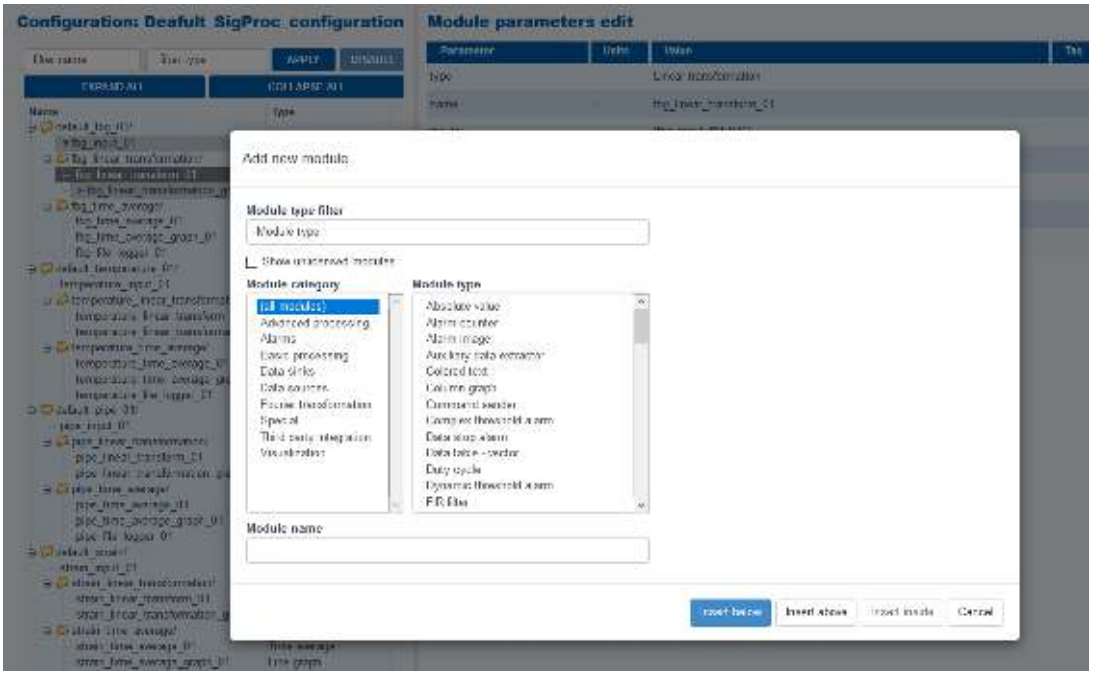


Figure 5.3 Add new module

Add group: Opens a dialog window with the option to define the name of the module group. The “Insert below/above/inside” buttons add a new group below or above the module or module group you have selected in the “Configuration tree” section or inside of the selected module group (see Figure 5.4). This module or module group are selected prior to clicking the “Add group” button and are marked by dark grey color (see Figure 5.2).

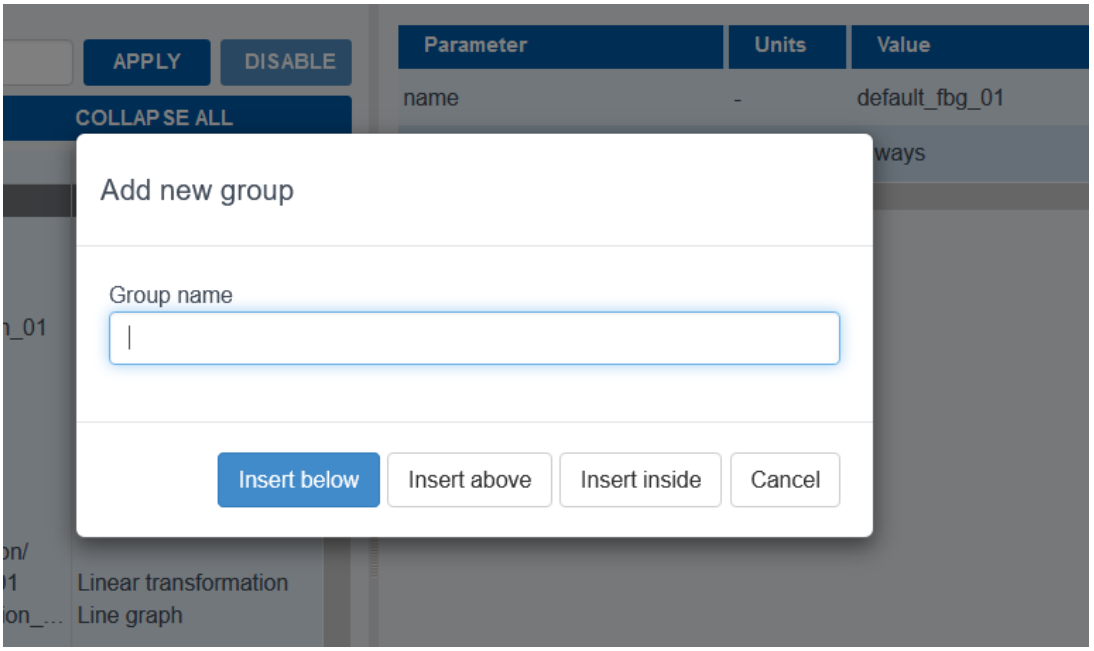


Figure 5.4 Add new group

Copy: Copies the selected module or module group and inserts it below the selected module group.

Erase: Deletes all selected modules or module groups.

Move up: Moves up the selected module or module group within the “Configuration tree” section.

Move down: Moves down the selected module or module group within the “Configuration tree” section.

Other: Shows/hides the dropdown menu.

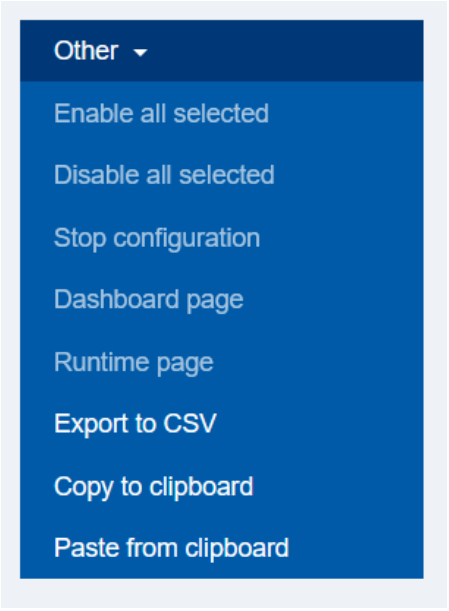


Figure 5.5 Other - dropdown menu

The dropdown menu provides additional actions you can perform with each configuration (see Figure 5.5):

Enable all selected: This button allows you to bulk enable all selected modules. Disabled module is striked out. Simply select all that needs to be enabled again and hit the “Enable all selected” (see Figure 5.6).

Disable all selected: This button allows you to bulk disable all selected modules. Select all modules that aren’t supposed to be (temporarily) active in the configuration and hit the “Disable all selected” button (see Figure 5.6).

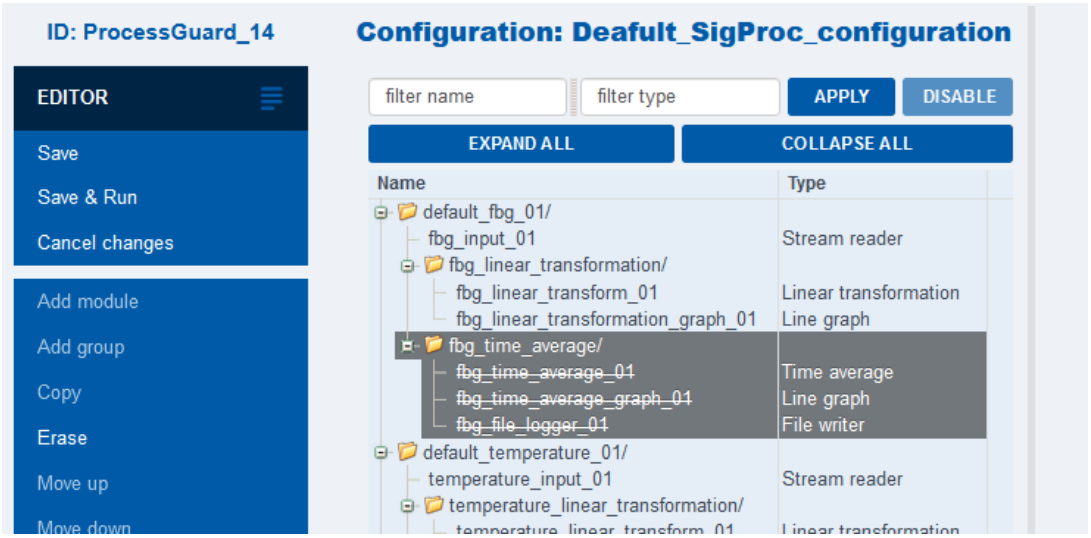


Figure 5.6 Enable/Disable all selected

Stop configuration: Stops the configuration if it is currently running.

Dashboard page: Opens the “Dashboard” page. This button is available only if the configuration is running.

Runtime page: Opens the “Runtime” page. This button is available only when the configuration is running.

Export to CSV: Exports parameters of all modules to the CSV format. This allows a visual verification for large configurations.

Copy to clipboard: Opens a dialog window that contains the appropriate part of the “Configuration string” of the selected part of the configuration (the selection must be done prior to clicking on the “Copy to clipboard” button). This configuration string can be then simply copied (Ctrl+C) and pasted (Ctrl+V) into different parts of the configuration tree or into different configurations using the “Paste from clipboard” button. This feature works only for a single module or a single module group.

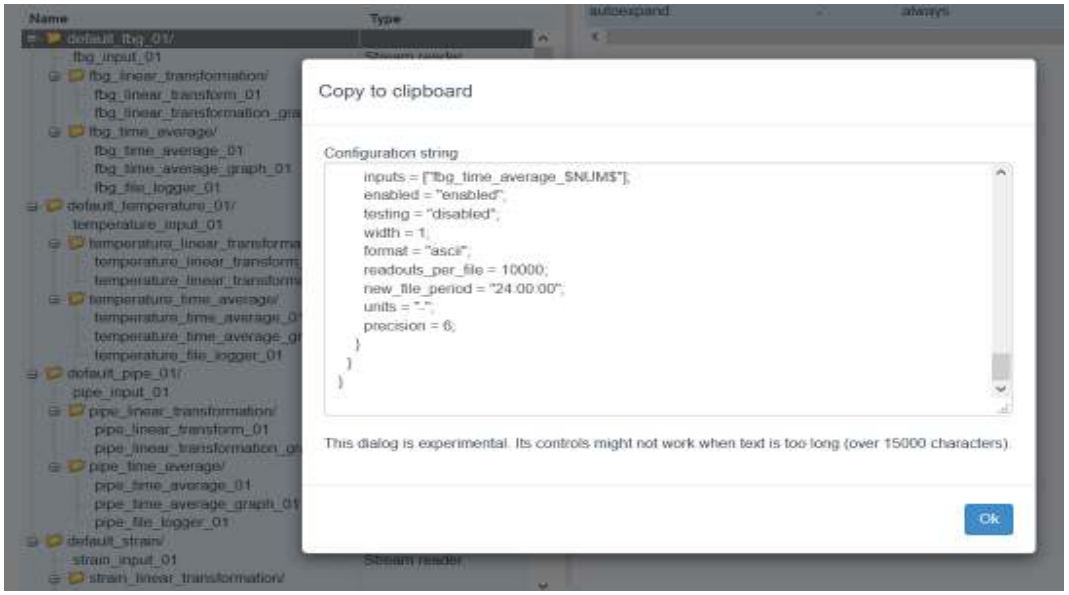


Figure 5.7 Copy to clipboard

Paste from clipboard: Opens a dialog window that allows you to paste the module or module group to the previously selected location in the configuration tree. Simply paste (Ctrl+V) the “Configuration string”, which was copied to clipboard in the previous step, into the dialog window and choose where the string is supposed to be added (e.g. below, above or inside the module group). This feature works only for a single module or a single module group.

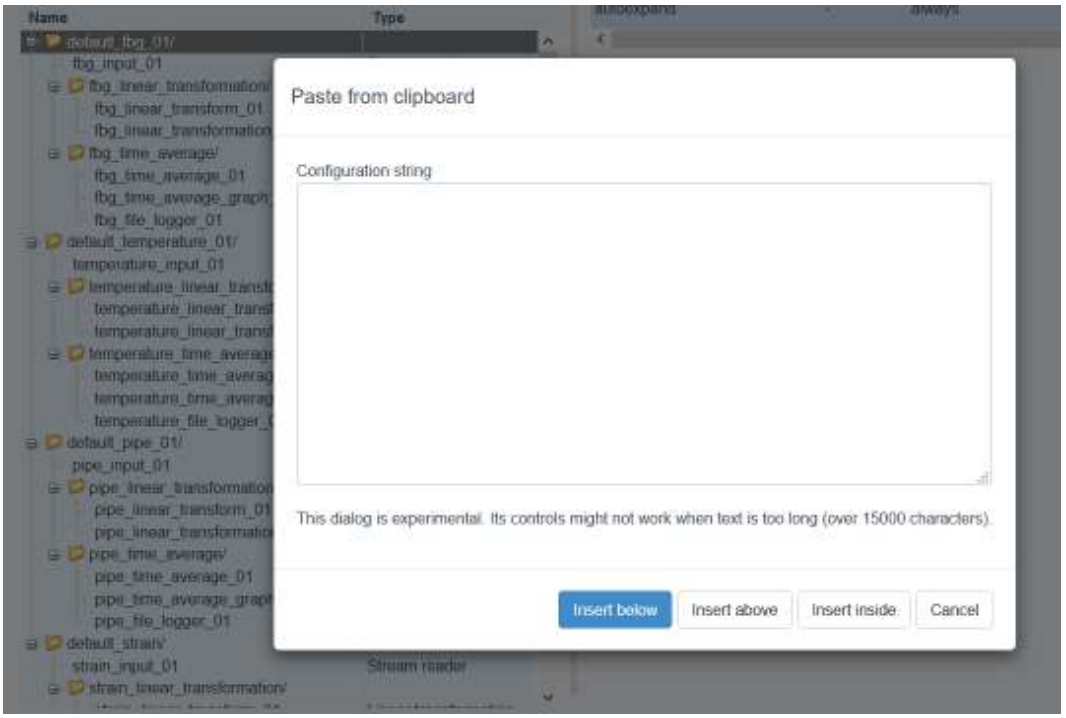


Figure 5.8 Paste from clipboard

5.1.2 Configuration Tree

The “Configuration tree” section displays configuration structure and allows reorganization of modules or module groups by the “Drag & Drop” feature or by using the “Move up/down” buttons. Clicking any item in the configuration tree opens the “Module parameters edit” section on the right hand side.

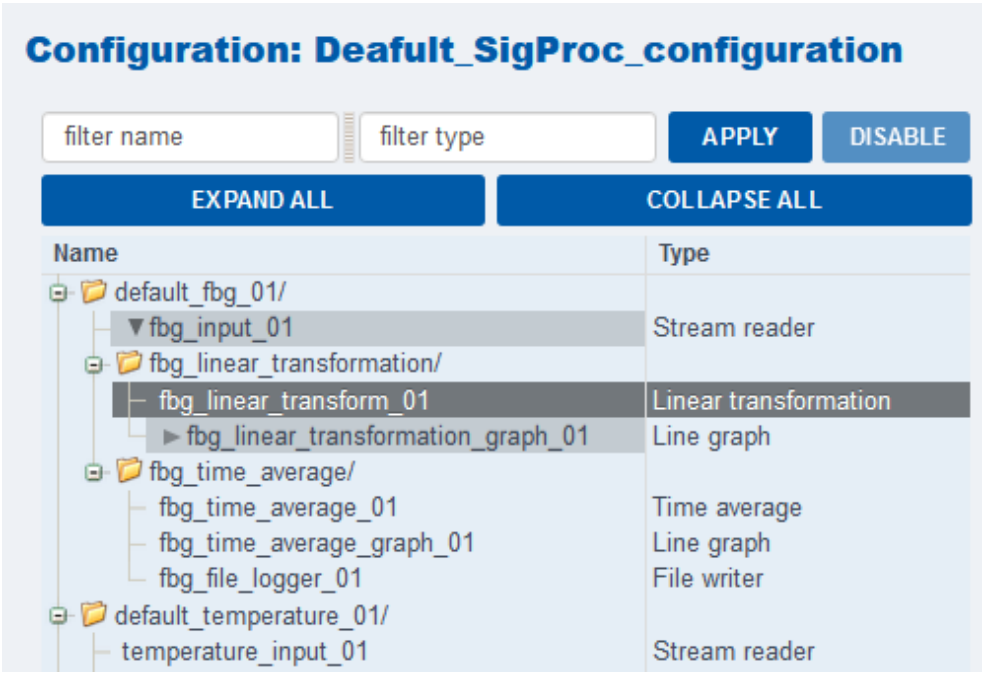


Figure 5.9 Configuration Tree

There are two filters that enable you to display only specific modules. Filtering is based on either module name or module type or both. Filter accepts keywords and limited regular expressions. Invalid regular expressions result in text inputs turning red.

Note: When filter is enabled some of the left menu buttons are disabled because their usage could lead to breaking the configuration.

Explanatory notes to Configuration Tree:

- **Filter name:** Filter based on the module name.
- **Filter type:** Filter based on the module type.
- **Apply:** Applies filter.
- **Disable:** Removes current filter.

Example: Filter examples

1. filter name: **alarm**
   filter type: **<empty>**

   Displays modules that contain the word “**alarm**”.

2. filter name: **alarm.*01**
   filter type: **<empty>**

Displays modules that contain the word "**alarm**" followed by "**01**". Words have to be in a given order.

3. filter name: **alarm.*01**
   filter type: **graph**

   Same as above. In addition this *module_type* parameter has to contain the word "**graph**".

4. filter name: **alarm.*0((1|2)|3)**
   filter type: **<empty>**

   Displays modules that contain the word "**alarm**" followed by numbers "**01**", "**02**" or "**03**". Words have to be in a given order.

There are two buttons below the filter: "Expand all" and "Collapse all". These buttons allow you to expand or collapse all module groups quickly.

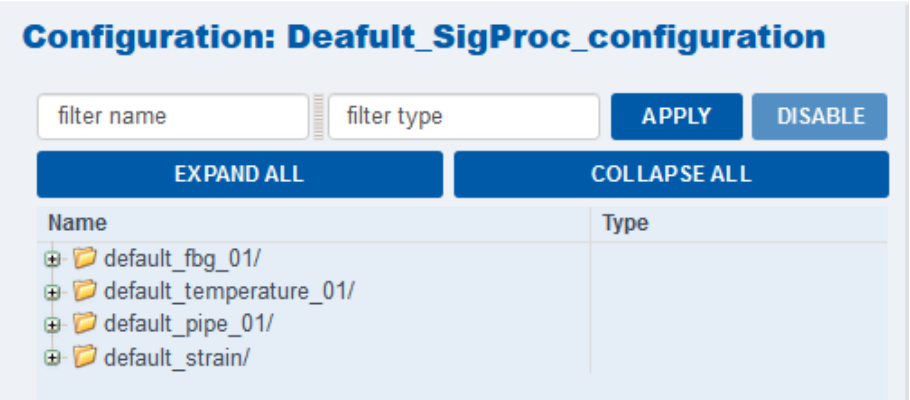


Figure 5.10 Collapse all function

If there are more modules selected it is possible to use the buttons for bulk editing, namely "Erase modules", "Disable all selected" and "Enable all selected".

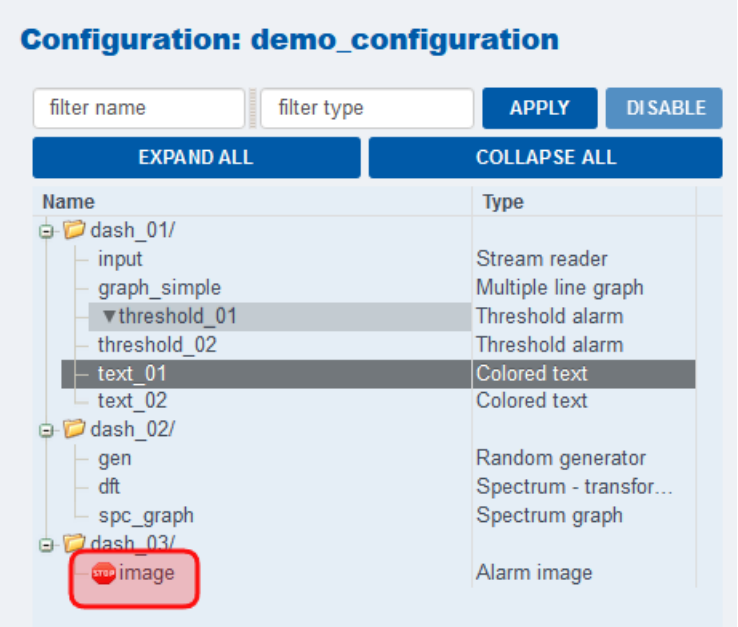
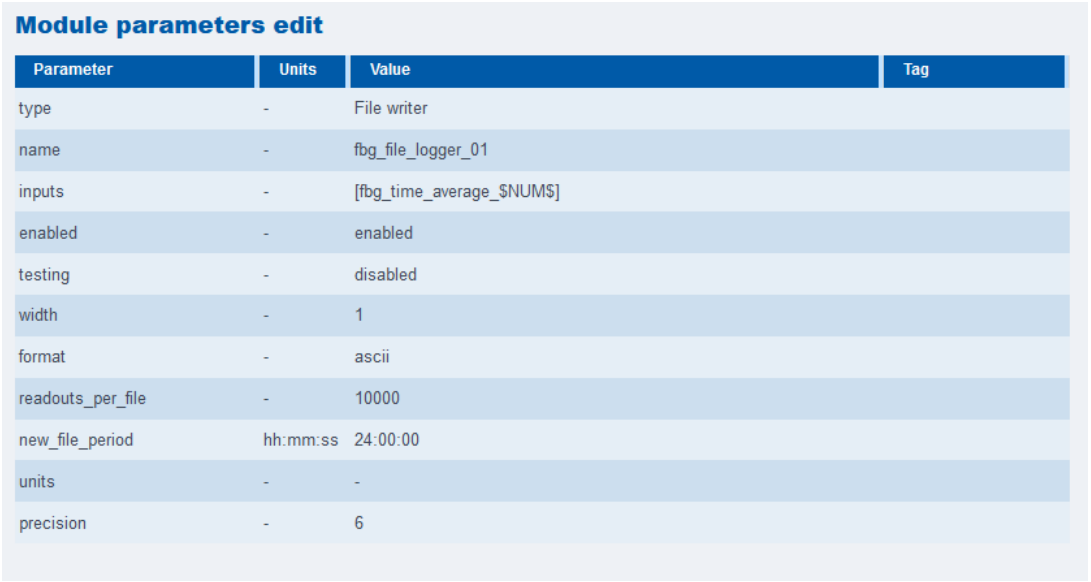


Figure 5.11 Module not supported by the current licence version

Note: If there is a module with a red stop icon before its name it means that this module isn't included in the current "SigProc" licence. In that case this module isn't supported and won't run in the current configuration. Even in this case it is possible to run the configuration, but all modules with this red icon will be disabled. They will run only in the "SigProc" with a higher license, which supports them.

5.1.3 Module Parameters Edit

The "Module Parameters Edit" section displays all parameters of the particular module and allows changing them. To change the parameter value type new text into the "Value" field and use the "Enter" key.



Module parameters edit

Parameter	Units	Value	Tag
type	-	File writer	
name	-	fbg_file_logger_01	
inputs	-	[fbg_time_average_NUM]	
enabled	-	enabled	
testing	-	disabled	
width	-	1	
format	-	ascii	
readouts_per_file	-	10000	
new_file_period	hh:mm:ss	24:00:00	
units	-	-	
precision	-	6	

Figure 5.12 Module parameters edit

Explanatory notes to Module Parameters Edit:

- **Parameter:** Parameter name.
- **Units:** Parameter units when any are applicable.
- **Value:** Parameter value.
- **Tag:** Parameters that share the "Tag name" change value whenever one of these tagged values is changed. For more information see the "Tags" section.

Some parameters are common for all modules, but their values are filled in individually for each module and therefore can differ across all modules. For example, the *name* parameter can be found by each module in the "Module parameters edit" section, but for each module this parameter acquires a different value.

Common parameters across all modules:

- **type:** Module type is filled in automatically when the module is created and it cannot be changed.
- **name:** Module name. This unique string is inserted as an input parameter in case this module serves as a signal source for another module. For naming rules see the note under the Figure 5.13.

- **inputs:** List of module inputs. Enter a module name of the signal source module into the square brackets. In case there are more inputs each module name has to be separated by a comma.
- **enabled:** Allows or disables the whole module within the configuration. This parameter can be modified in several modules at once using the "Enable/Disable all selected" buttons in the left menu.

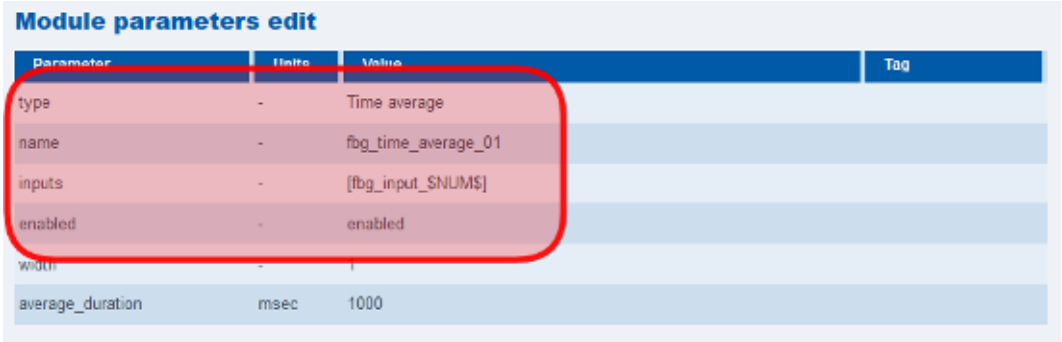


Figure 5.13 Shared parameters

Note: It is necessary to follow the naming rules for module and module group name. The name has to be unique across the whole configuration and must be only one string. It must not start with a number. Allowed characters are letters (lowercase and uppercase), numbers and underscore. If the chosen name for the module or module group already exists it will be automatically saved with the “index number” suffix at the end of the name followed by a warning notice. This ensures that the name is unique. Similarly, in case that not supported character is used it will be automatically changed to underscore.

5.2 Clone Configuration

It allows you to copy an existing configuration. It opens a dialog window where you can enter a new name for the cloned configuration.

Note: The configuration name must follow the naming rules similar as to module or module group names. The configuration name has to be unique and must be only one string. It must not start with a number. Allowed characters are letters (lowercase and uppercase), numbers, underscore, plus sign and dash.

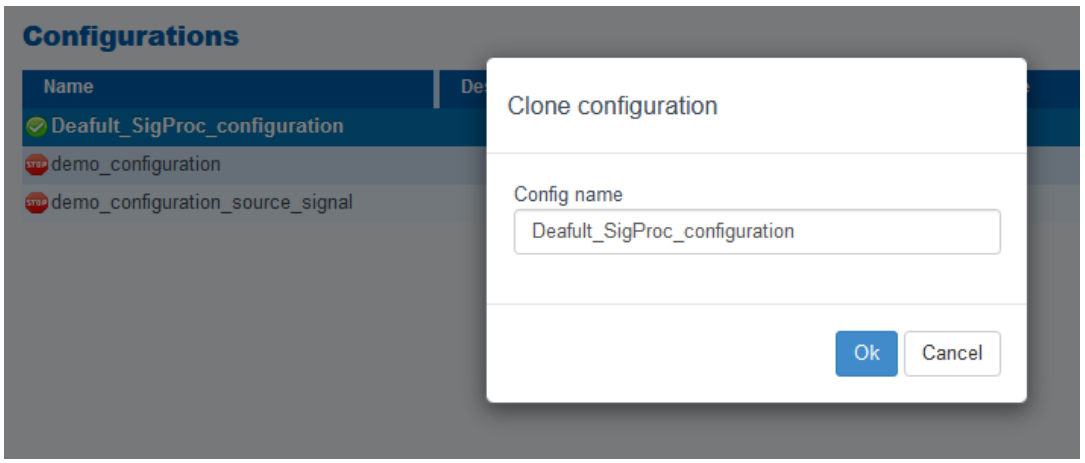


Figure 5.14 Clone configuration

5.3 New Configuration

It opens a dialog window where the name of the new configuration can be inserted. When the name is confirmed, a new empty configuration will be added to the list of configurations.

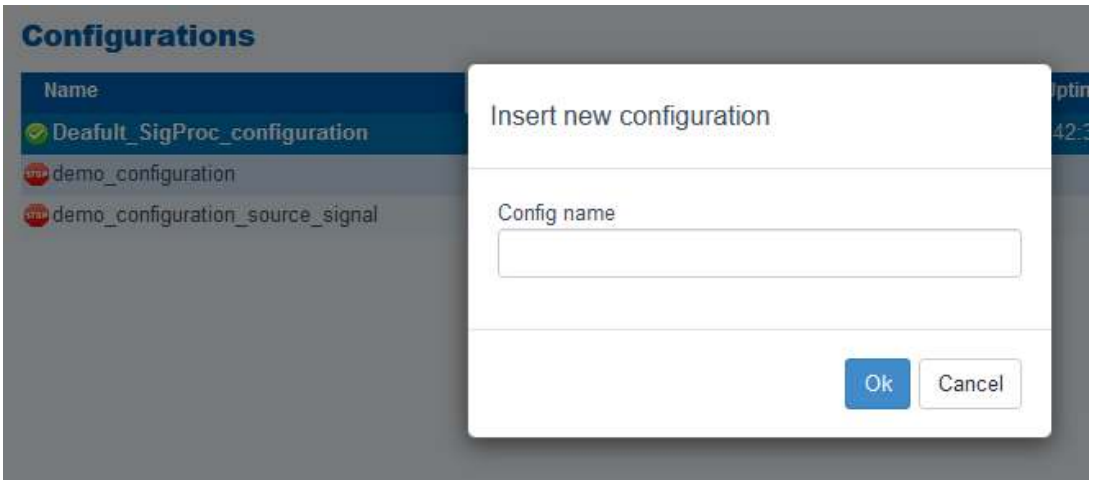


Figure 5.15 New configuration

5.4 Rename Configuration

It is possible to change the name of an existing configuration except for the running one.

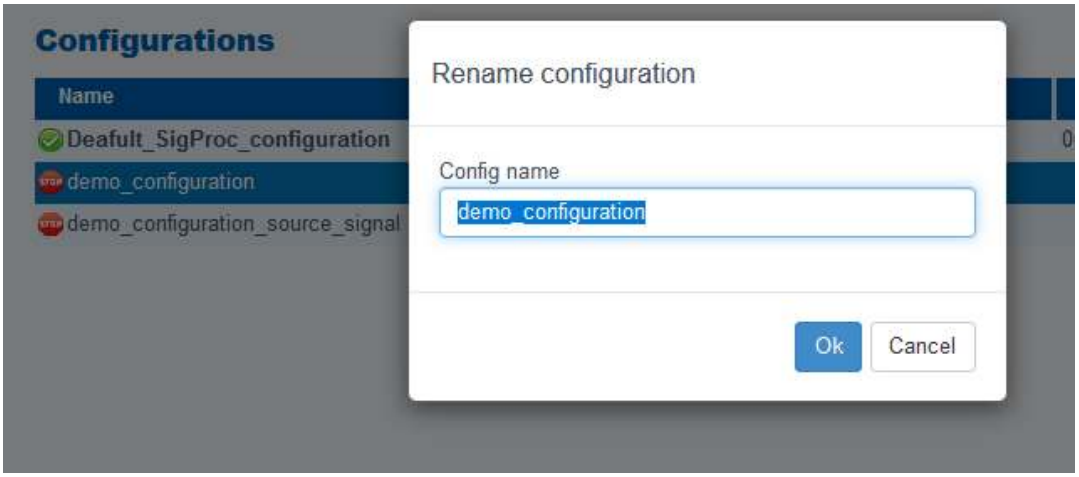


Figure 5.16 Rename configuration

5.5 Erase Configuration

It opens a dialog window to confirm the deletion of the current configuration.

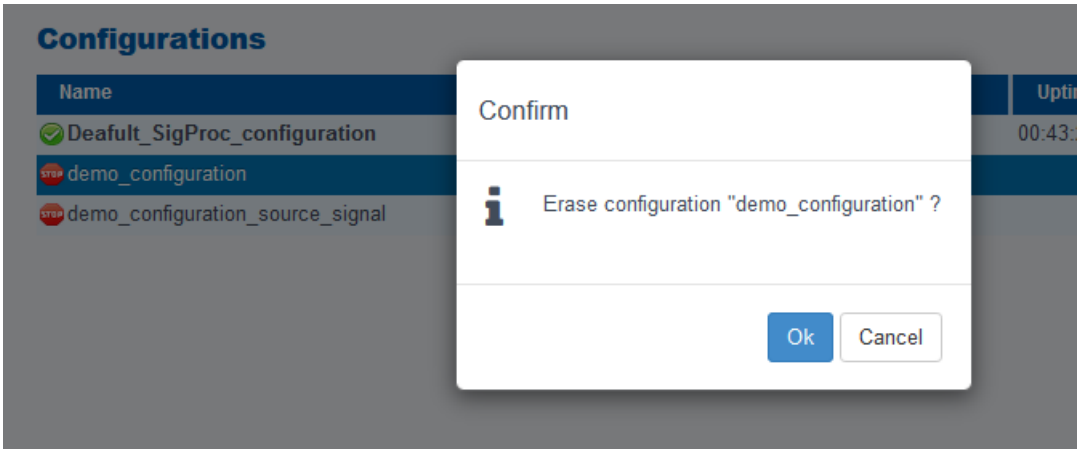


Figure 5.17 Erase configuration

5.6 Run Configuration

It opens a dialog window to confirm the start of the configuration. If the configuration is already running, the dialog window is asking if the configuration should be restarted.

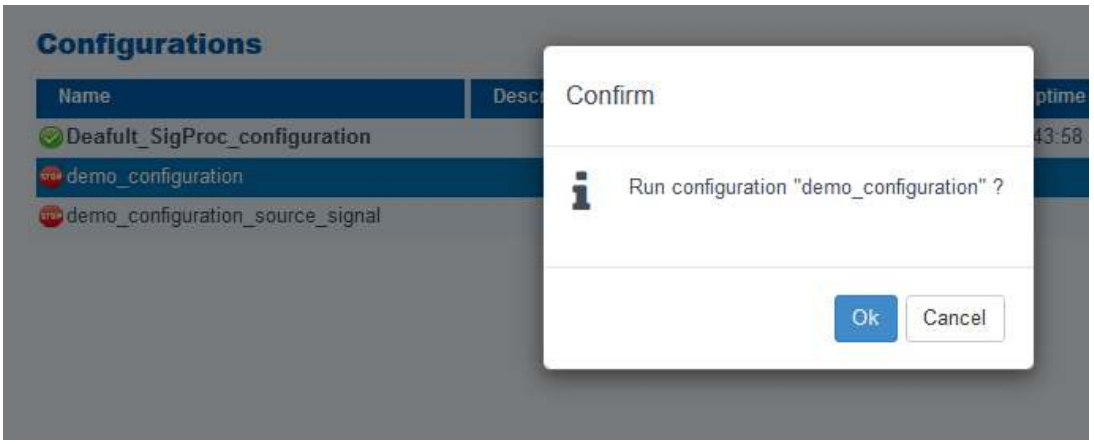


Figure 5.18 Run configuration

5.7 Stop Configuration

It opens a dialog window to confirm stopping the configuration.

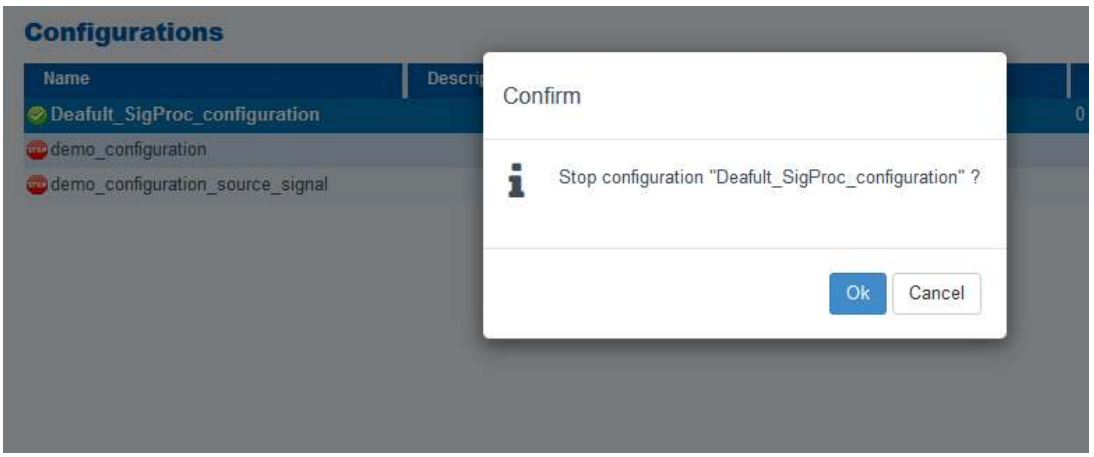


Figure 5.19 Stop configuration

5.8 Dashboard Page

The “Dashboard” page displays only modules with graphical output. The graphical modules are those from the module category called “Visualization”. If the configuration does not contain any visualization module, the ”Dashboard” page is empty. To display the “Dashboard” page, the configuration must be currently running.

Figure 5.20 Graphical output - the “Dashboard” page

The “Dashboard” page is divided into two parts. Central part displays outputs of visualisation modules and the name of the configuration that the “Dashboard” page belongs to. Left menu allows users to switch between dashboards in case that the configuration has more dashboards set up to navigate to other pages.

The left menu buttons provide the actions you can perform:

Dashboards: This section shows all dashboards available in the current configuration. Dashboard name is defined for every graphical module. Every graphical module has the parameter called *gui_page* that specifies its dashboard name. This parameter is always set up as “Dashboard” by default. Furthermore, any graphical module cannot be displayed on more “Dashboards” at the same time.

Note: If you want to place more graphs into one dashboard enter the same *gui_page* parameter. It is also possible to use "Tags" for easier editing.

Editor page: Opens the "Edit configuration" page.

Runtime page: Opens the configuration runtime view.

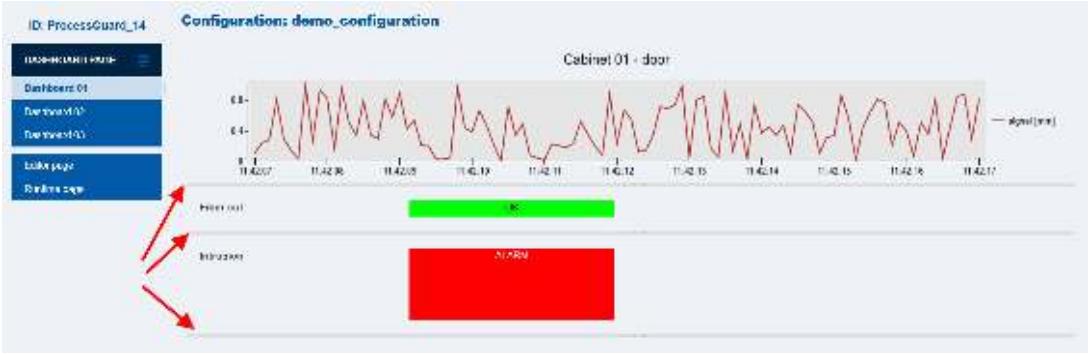


Figure 5.21 Graphical output - adjusting line

The size of each element on the "Dashboard" page can be adjusted using the gray line below each graph (highlighted by red arrows in Figure 5.21).

For more information about how to set up the "Dashboard" page reach out to the "Modules" section.

5.9 Runtime Page

The "Runtime" page shows detailed information and uptime of each module in the running configuration. This page has the same layout as the "Edit configuration" page. However, editing is not enabled and only information about modules can be viewed.

Note: This functionality is accessible only for the running configuration. It is useful when debugging or when searching for modules with high CPU usage.

It is possible to jump into the "Configuration editor" page using the "Open in editor" button from the left side menu.

Figure 5.22 View runtime

Explanatory notes to Module runtime parameters view:

- **Invocation count:** How many times the module has been run.
- **Run time:** How much time the particular module consumed to process its function.
- **Up time:** How long the whole configuration is running.
- **CPU usage:** CPU usage of the whole module.
- **Start time:** The time when the configuration started.
- **Log:** Displays log messages for selected module only. Shows last 20 logs.

5.10 Export Configuration

This function allows you to export configuration into the .cfg file which can be later imported into other "SigProc" software. It is a smooth and fast way how to copy configurations between “SigProces” on different devices.

Select the particular configuration that should be exported and click on the “Export configuration” button. In the dialog window confirm the file name of the selected configuration (see Figure 5.23) and click on the “Download” button. Afterwards choose the location where the exported file will be saved.

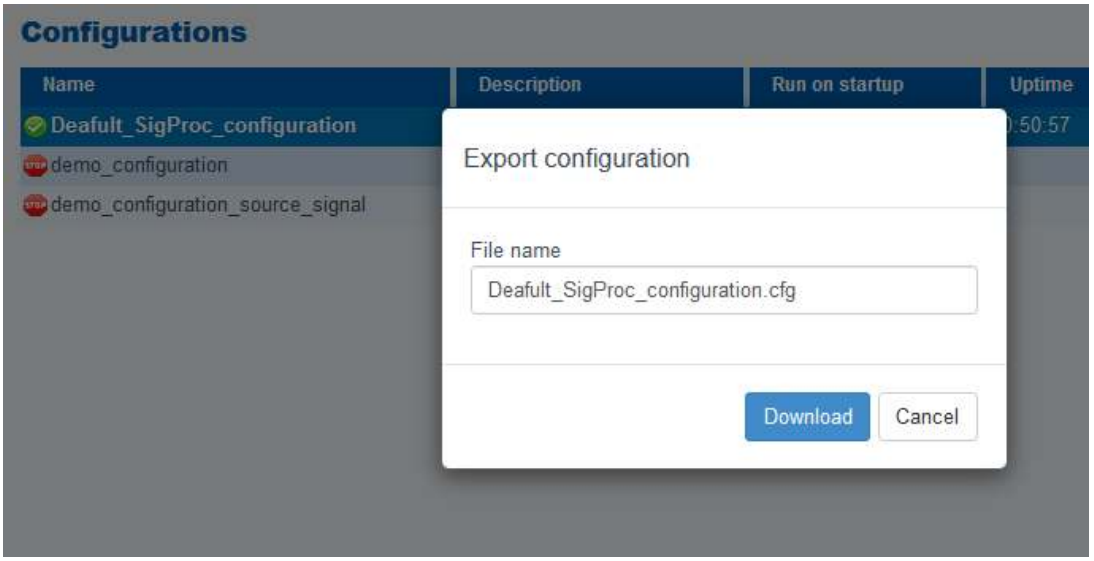


Figure 5.23 Export configuration

5.11 Import Configuration

This function allows you to import configuration in .cfg format. To import a new configuration click on the “Import configuration” button. In the dialog window select the particular .cfg file, define the name of the new configuration (see Figure 5.24) and click on the “Import” button. Afterwards, this new configuration will appear on the list of all configurations.

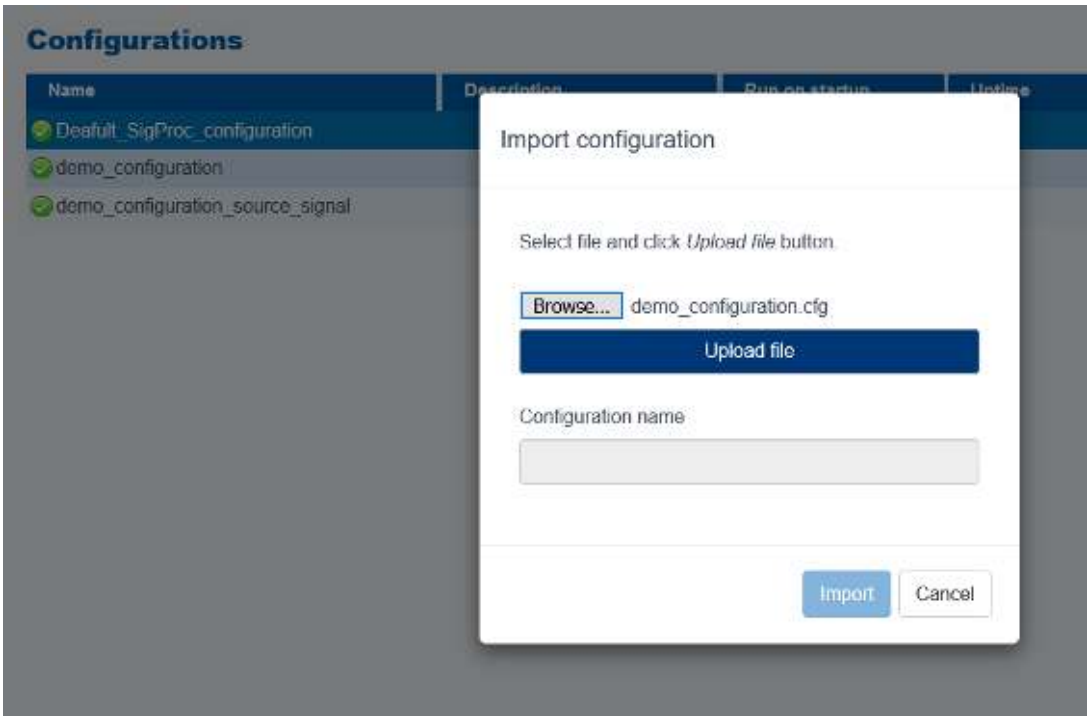


Figure 5.24 Import configuration

6 SIGPROC LICENCES

6.1 Licence Versions

The ”SigProc" licenses are divided according to the areas and tasks they are focused on. All licenses are permanent and have no expiration date. It is possible to upgrade to another level licence anytime (contact SAFIBRA for more information).

Fundamental Licence

Basic modules required for work with signal, allowing to process data from the external sources in a real-time stream. Allows basic Visualization and test modules for the configurations.

Module name	Category
Absolute value	Basic processing
Colored text	Visualization
File reader	Data sources
File writer	Data sinks
Line graph	Visualization
Linear transformation	Basic processing
Multiple line graph	Visualization
Random generator	Data sources
RelayUnit controller	Data sinks
Spectrum graph	Visualization
Spectrum - transformation	Fourier transformation
Stream reader	Data sources
Stream writer	Data sinks
Sum	Basic processing
Threshold alarm	Alarms
Time average	Basic processing
Value table	Visualization

Fundamental Plus Licence

Extensive work in the frequency domain as well as an intermediate signal processing and thresholding. Enables sending the stream output into the additional software. Includes bonus Visualisation modules.

Module name	Category
Complex threshold alarm	Alarms
FIR filter	Advanced processing
Hysteresis threshold alarm	Alarms
Line graph - vector	Visualization
Linear regression	Advanced processing
Moving average	Basic processing
Polynomial transformation	Advanced processing
Signal modifier	Special
Signals to vector convertor	Special
Spectrum - band filter	Fourier transformation
Spectrum - band sum	Fourier transformation
Stream writer - vector	Data sinks
Sum - vector	Basic processing
User input	Visualization

Advanced Licence

Special modules designed for more difficult configuration schemes. Alarm handling modules and Visualization color outputs for observing the status of the signal and alarms. Output to basic database systems, such as "PostgreSQL".

Module name	Category
Column graph	Visualization
Command sender	Special
Data stop alarm	Alarms
Data table - vector	Visualization
Fourier filter	Fourier transformation

Min/max holder	Advanced processing
PostgreSQL writer	Data sinks
Sampling speed	Advanced processing
Spectrum - binary	Fourier transformation
Spectrum - max amplitude	Fourier transformation
Spectrum - max frequency	Fourier transformation
Vector min/max	Advanced processing
Zabbix	Third party integration

Expert Licence

Higher mathematical modules, such as the dynamic threshold. Alarm visualization in “Google Maps” or into an image. Real-time output to "InfluxDB" or "PostgreSQL".

Module name	Category
Alarm counter	Alarms
Alarm image	Visualization
Auxiliary data extractor	Special
Duty cycle	Advanced processing
Dynamic threshold alarm	Alarms
Google Maps	Visualization
Inclination profile	Special
InfluxDB writer	Data sinks
Interferometer deconvolution	Special
Modbus slave	Data sinks
ObjectGuard	Special
Sequence generator	Data sources
Sine wave generator	Data sources
Time maximum	Advanced processing
Time minimum	Advanced processing

Exclusive Licence

Third party integration modules, JSON messages and HTTP requests. Fingerprint pattern based alarms.

Module name	Category
HTTP request	Third party integration
JSON message event	Third party integration
JSON message service	Third party integration
Milestone string event	Third party integration
Wavelet fingerprints	Special

Note: For more information about the different modules in each licence version, please, see the link under the name of each module.

7 MODULES

7.1 Working with Modules

Working with modules means creating configurations by connecting two or more modules. In order for a module to work properly it needs some input data that are processed to produce the requested output. And that is the basic principle of connecting modules. Output from module 1 (i.e. input module) is being used as an input to module 2 (i.e. output module). Here are some basic steps how you can proceed when connecting modules:

Open the requested configuration for editing: Each configuration can be edited in the "Configuration editor" (see "Open in editor" section). In order to select the configuration for editing in the "Configuration editor", click on the particular configuration from the list of all configurations and then click the "Open in editor" button in the left menu. Same can be also achieved by double clicking on the particular configuration.

Distinguish between an input and an output module: For better orientation in the configuration tree, there are all input and output modules graphically marked (see Figure 7.1):

- **Light grey color with arrow pointing down:** The input module is marked by light grey color with an arrow next to the module name, which is pointing down.
- **Light grey color with arrow pointing right:** The output module is marked by light grey color as well but the arrow next to the module name is pointing right.
- **Dark grey color:** The module that is being edited is marked by dark grey color.

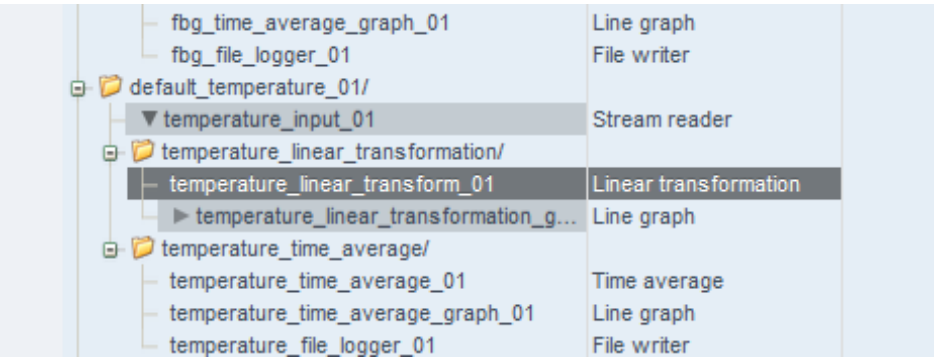


Figure 7.1 Graphical marking of modules (input, output and edited module)

Note: Detailed information about inputs and outputs of each module can be found in the section dedicated to the particular module in the "Modules" section.

Select the output module for editing: To select the preferred output module from the list of all modules just click on the element by the left mouse button. Then go to the "Module parameters edit" section of the selected module, which is located on the right side.

Fill in the *inputs* parameter of the output module: The input for each module is set in the *inputs* parameter in the "Module parameters edit" section (see Figure 7.2). Just enter the "name" of the input module in square brackets into the *inputs* parameter of the output module. If you want to use more than one module as an input module, then enter the exact names of all input modules separated by a comma.

Note: Connecting modules always has to be done manually for each module.

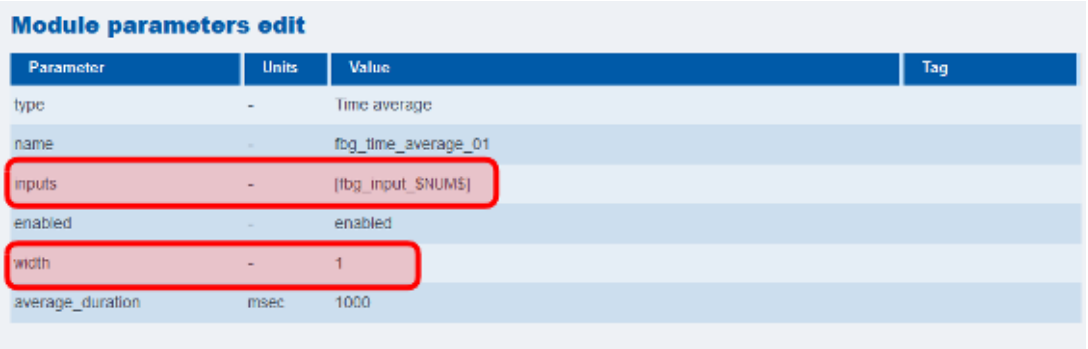


Figure 7.2 - “Inputs” parameter of the module

Check the *width* parameter of the input and output module: If you work with modules that work with vectors or when you are connecting more modules into one input it is important to make sure that all concerned input and output modules are set to the same width.

7.1.1 NUM Construct – Automatic Number Replacement

To copy configurations without having to change every parameter, there is a syntax construct NUM that copies the number defined in the module name. In this way, if the “inputs” parameter contains “_NUM” at the end, then it automatically reflects the number from the “name” parameter.

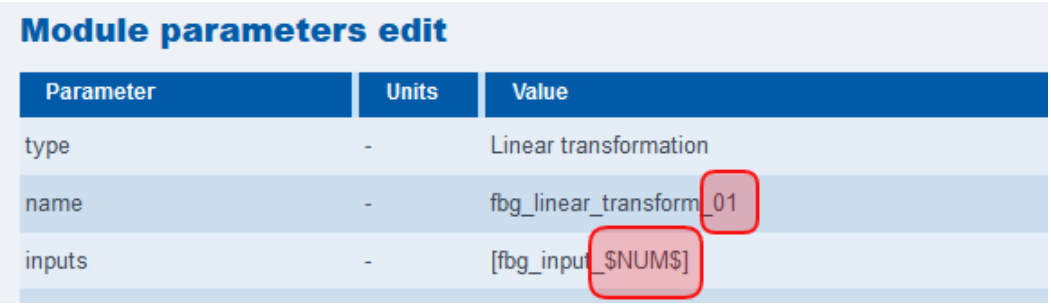


Figure 7.3 $NUM construct

Note: Name modules in a way that allows the use of NUM construct. For example, if the *name* parameter is “ch_01” while *inputs* parameter is “input_NUM”, then when the module is copied, the *name* parameter changes to “ch_02” meanwhile *inputs* parameter stays “input_NUM”.

7.1.2 Tags

Tags allow you to change the parameters across the whole configuration on one click. Every parameter can have a tag assigned. When a parameter value has a tag assigned and it is changed, it adjusts instantly every parameter value with the same tag assigned.

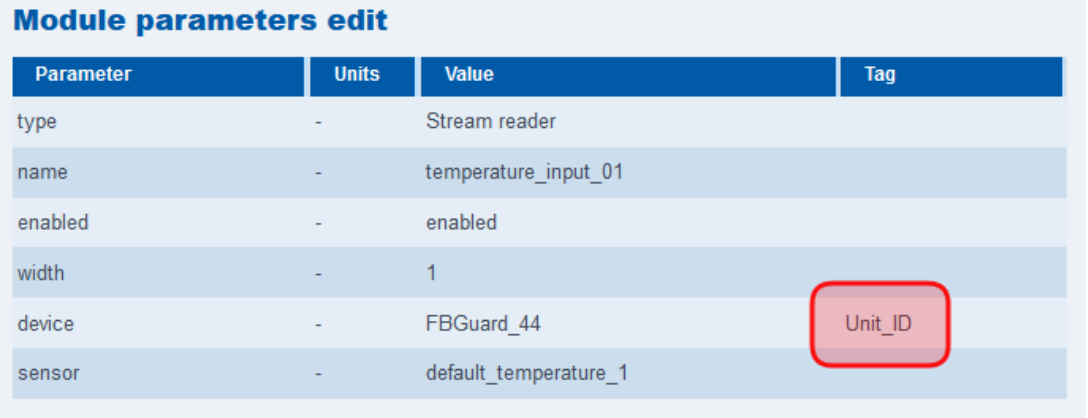


Figure 7.4 Tag

Note: When you copy a module where tags contain numbers, for example Tag01, the number is not changing. It stays the same as in the original module tag. The NUM construct is not supported in case of Tags.

7.2 Modules under Fundamental Licence

7.2.1 Absolute Value Module

Description

The module computes an absolute value.

- **License**: Fundamental.
- **Category**: Basic processing.
- **Input**: Single value, single vector.
- **Output**: Single value, single vector.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.

7.2.2 Colored Text Module

Description

The module displays text with a colored background. The text and the color change dynamically based on the input value.

- **License**: Fundamental.
- **Category**: Visualization.
- **Input**: Single value (index).
- **Output**: None.

Parameters

Name	Units	Type	Description
gui_name	-	string	Input signal name displayed on the “Dashboard” page.
item_strings	-	string array	Array of displayed texts.
colors	-	integer array	Array of background colors.

Documentation

Module rounds the input value to the nearest integer and then uses this integer as an index to the *item_strings* array and the *colors* array. The text and the color at that index is displayed. Please, note that the index starts from zero.

The color in *colors* parameter is represented as a 9 digit number (trailing zeros can be omitted). The first triplet is the red component, the second triplet is the green component and the last triplet is the blue component. The range of each color component is 0 - 255.

The module is typically used for alarm visualization.

Module parameters edit

Parameter	Units	Value
type	-	Colored text
name	-	intrusion_alarm
inputs	-	[alarm_01]
enabled	-	enabled
refresh	msec	1000
gui_page	-	Dashboard
gui_name	-	Intrusion
item_strings	-	[OK, Alarm]
colors	-	[255000, 255000000]

Figure 7.5 The “Colored text” module settings

Figure 7.6 Displayed text if the input is 0

Figure 7.7 Displayed text if the input is 1

7.2.3 File Reader Module

Description

The module reads data files from the disk.

- **License**: Fundamental.
- **Category**: Data sources.
- **Input**: None.
- **Output**: Single value, single vector.

Parameters

Name	Units	Type	Description
width	-	integer	Number of columns loaded from the files.
testing	-	option	(**disabled**) Log only basic messages. (**enabled**) Log verbose messages.
lines	-	integer	Total number of loaded lines (0 = read everything).
base_location	-	string	File/directory path - the first part.
file	-	string	File/directory path - the second part.
mode	-	option	(**fast**) Reading is as fast as possible. (**real time**) Reading is timed by timestamps in the files.
reading_speed	-	integer	Multiplier of the original speed. Valid only when the mode is set to real time.
repeat	-	option	**(disabled**) Read the files only once. (**enabled**) Repeat the reading.
time_start	-	string	Time range limit, older readouts will be skipped. Time format: yyyy-MM-ddTHH:mm:ss. Optional parameter.
time_end	-	string	Time range limit, newer readouts will be skipped. Time format: yyyy-MM-ddTHH:mm:ss. Optional parameter.

Documentation

The module loads readouts from data files created by the "File writer" module or the "MeSyCo". The structure of these files is described in the "Data file example" section.

The *base_location* and *file* parameters are concatenated into one string that is used as a path to data files. The path can be a single file or a directory. If the path is a directory then the module reads all files in the directory and subdirectories. Files are read one after another in alphabetical order.

If the *testing* parameter is enabled the module logs more information about the loaded files.

The time range of the loaded readouts can be set by *time_start* and *time_end* parameters. Both parameters can be set independently.

If the *mode* is set to "fast" the readouts are loaded as fast as possible. Please, note that this option is CPU intensive. If the *mode* is set to real time the readouts are loaded at the same speed as they were sampled. The speed can be accelerated using the *reading_speed* parameter.

Module parameters edit

Parameter	Units	Value
type	-	File reader
name	-	read_file
enabled	-	enabled
testing	-	enabled
width	-	1
lines	-	0
base_location	-	/home/sigproc/data/demo_source_data/file_writer/
file	-	2017-06-01/save-2017-06-01_23-48-53.csv
mode	-	fast
reading_speed	-	1
repeat	-	disabled
time_start	-	
time_end	-	

Figure 7.8 The "File reader" module settings - single file

Example: Logged message from the "File reader" module - single file

```
Loaded data from file
"/home/sigproc/data/demo_source_data/file_writer/2017-06-01/save-2017-06-01_23-48-5
3.csv". Readout count 8640.
```

Module parameters edit

Parameter	Units	Value
type	-	File reader
name	-	read_directory
enabled	-	enabled
testing	-	enabled
width	-	1
lines	-	0
base_location	-	/home/sigproc/data/demo_source_data/file_writer/
file	-	
mode	-	fast
reading_speed	-	1
repeat	-	disabled
time_start	-	2017-06-01T00:00:00
time_end	-	

Figure 7.9 Whole folder and limited time range

Example: Logged messages from the “File reader” module - whole folder:

```
Loaded data from file
"/home/sigproc/data/demo_source_data/file_writer/2017-05-28/save-2017-05-28_23-48-5
3.csv". Readout count 0.

Loaded data from file
"/home/sigproc/data/demo_source_data/file_writer/2017-05-29/save-2017-05-29_23-48-5
3.csv". Readout count 0.

Loaded data from file
"/home/sigproc/data/demo_source_data/file_writer/2017-05-30/save-2017-05-30_23-48-5
3.csv". Readout count 0.

Loaded data from file
"/home/sigproc/data/demo_source_data/file_writer/2017-05-31/save-2017-05-31_23-48-5
3.csv". Readout count 8573.

Loaded data from file
"/home/sigproc/data/demo_source_data/file_writer/2017-06-01/save-2017-06-01_23-48-5
3.csv". Readout count 8640.

Loaded data from file
"/home/sigproc/data/demo_source_data/file_writer/2017-06-02/save-2017-06-02_23-48-5
3.csv". Readout count 8640.
```

```
Loaded data from file
"/home/sigproc/data/demo_source_data/file_writer/2017-06-03/save-2017-06-03_23-48-5
3.csv". Readout count 21651.
```

Some files were not loaded (Readout count 0.) because they were outside of the defined time range.

7.2.4 File Writer Module

Description

The module writes data into files on a disk.

- **License**: Fundamental.
- **Category**: Data sinks.
- **Input**: Single value, single vector.
- **Output**: None.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
testing	-	option	(**disabled**) Log only basic messages. (**enabled**) Log verbose messages.
format	-	option	(**ascii**) File format is CSV. (**binary**) File format is BIN.
readouts_per_file	-	integer	Maximal number of readouts saved in one file. A new file is created when the limit is exceeded.
new_file_period	-	string	Maximal time interval saved in one file. A new file is created when the time limit is exceeded. Format: HH:mm:ss.
units	-		Units saved at the beginning of each file. Only for ascii format.
precision	-		Number of decimal places. Only for ASCII format.

Documentation

Data files are created in the */home/sigproc/data/<configuration_name>/<module_name>/<date>/<module_name>-<date_and_time>.<csv_or_bin>* directory.

Example: File created by “module logger_01” in the “DemoFileLogger” configuration

/home/sigproc/data/DemoFileLogger/logger_01/2020-09-02/logger_01-2020-09-20_16-40-21.csv

Created files are accessible from the “Data file browser”, see the “Data file browser” section.

The structure of these files is described in the “Data file example” section.

7.2.5 Line Graph Module

Description

The module displays input values in a time dependent line graph.

- **License**: Fundamental.
- **Category**: Visualization.
- **Input**: Single value.
- **Output**: None.

Parameters

Name	Units	Type	Description
gui_description	-	string	Title displayed on the “Dashboard” page.
gui_name	-	string	Input signal name displayed on the “Dashboard” page.
gui_units	-	string	Units displayed on the “Dashboard” page.
grid_lines	-	option	(**enabled**) Horizontal grid lines are enabled. (**disabled**) Horizontal grid lines are disabled.
duration	s	integer	Time interval displayed in the graph.
minmax_waveform	-	option	(**enabled**) Show min and max waveforms. (**disabled**) Hide min and max waveforms.
default_min	-	decimal	Default fixed minimum.
default_max	-	decimal	Default fixed maximum.

Documentation

The module manages to process large amounts of measured values (long time interval, high sampling frequency), but in order to be able to display these values in a real time graph, it averages them to reduce their quantity.

Because of the averaging, the short-term extremes can be lost from the signal, and therefore the graph shows not just the averaged values but also the minimums and the maximums from the averaged intervals. The minimums and the maximums represent the signal envelope.

The number of averaged readouts and calculated sampling frequency is displayed under the graph.

The graph can be controlled using the graphical control elements located under the graph:

- **Stop time**: Stop refreshing the graph.
- **Autoscale**: The Y axis scale is adjusted automatically.
- **Hold min/max**: The Y axis scale is adjusted automatically and it can only be expanded.
- **Fixed**: The Y axis scale is set to the user defined values entered in the *Fixed min* and *Fixed max* text fields.

Module parameters edit

Parameter	Units	Value
type	-	Line graph
name	-	graph
inputs	-	[sensor_01]
enabled	-	enabled
refresh	msec	1000
gui_page	-	Dashboard
gui_description	-	Title
gui_name	-	length
gui_units	-	mm
grid_lines		enabled
duration	s	10
minmax_waveform	-	enabled
default_min		0
default_max		0

Figure 7.10 The “Line graph” module settings

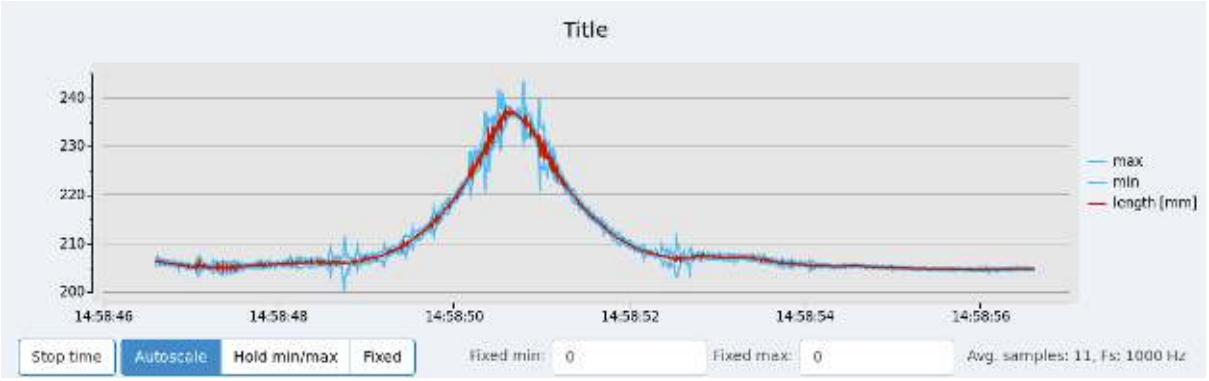


Figure 7.11 The line graph on the “Dashboard” page (red line - average, blue lines - minimum and maximum)

7.2.6 Linear Transformation Module

Description

The module computes a linear transformation.

- **License**: Fundamental.
- **Category**: Basic processing.
- **Input**: Single value, single vector.
- **Output**: Single value, single vector.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
multi	-	decimal	Multiplicative coefficient.
additive	-	decimal	Additive coefficient.

Documentation

If the input is a single value the module computes the output value using the following formula.

$$y = multi \cdot x + additive$$

Where y is output value and x is input value.

If the input is a single vector the module computes the output using the following formula.

$$y_i = multi \cdot x_i + additive$$

Where y_i is an element of the output vector and x_i is an element of the input vector.

7.2.7 Multiple Line Graph Module

Description

The module displays input values in a time dependent line graph. The graph has multiple lines if the module has multiple inputs.

- **License**: Fundamental.
- **Category**: Visualization.
- **Input**: Single value, multiple values.
- **Output**: None.

Parameters

Name	Units	Type	Description
gui_description	-	string	Title displayed on the “Dashboard” age.
gui_names	-	string array	Input signal names displayed on the “Dashboard” page. Leave empty if you want to use the default names.
gui_units	-	string	Units displayed on the “Dashboard” page.
grid_lines	-	option	(**enabled**) Horizontal grid lines are enabled. (**disabled**) Horizontal grid lines are disabled.
duration	s	integer	Time interval displayed in the graph.

Module parameters edit

Parameter	Units	Value
type	-	Multiple line graph
name	-	graph_multi
inputs	-	[sensor_01, sensor_02, sensor_03]
enabled	-	enabled
refresh	msec	1000
gui_page	-	Dashboard
gui_description	-	Title
gui_names	-	[sensor A, sensor B, sensor C]
gui_units	-	
grid_lines		enabled
duration	s	10

Figure 7.12 The “Multiple line graph” module settings

Figure 7.13 Multiple line graph on the “Dashboard” page

7.2.8 Random Generator Module

Description

The module generates uniformly distributed random numbers.

- **License**: Fundamental.
- **Category**: Data sources.
- **Input**: None.
- **Output**: Single value, single vector.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the output vector.

period	msec	integer	Sampling period of the generated values.
speed	-	decimal	Generating period (divisor of the period).
min	-	integer	Upper range of the generated values.
max	-	integer	Lower range of the generated values.
offset_increment	-	decimal	Offset cumulatively added to the generated values.
start	-	string	Start time of the generated values. Time format: yyyy-MM-ddTHH:mm:ss. Leave empty if you want to use the current time.

7.2.9 RelayUnit Controller Module

Description

The module controls relays in the RelayUnit.

- **License**: Fundamental.
- **Category**: Data sinks.
- **Input**: Single value (boolean).
- **Output**: None.

Parameters

Name	Units	Type	Description
external_relay_index	-	integer	Index of the controlled relay.
external_relay_timeout	sec	integer	Timeout for automatic deactivation (0 = disabled).

Documentation

Relay at the index *external_relay_index* is activated when the input value is nonzero and deactivated after the specified timeout. When the relay is already activated and the input is nonzero, then the timeout is extended. If the timeout is disabled the relay is deactivated immediately when the input is zero.

For the module to work properly, the “RelayUnit driver” must be configured in the “System configuration file”.

7.2.10 Spectrum Graph Module

Description

The module displays frequency spectrum in a column graph. The input is a spectrum, which is generated by modules from the Fourier transformation category.

- **License**: Fundamental.
- **Category**: Visualization.
- **Input**: Single vector (spectrum).
- **Output**: None.

Parameters

Name	Units	Type	Description
gui_description	-	string	Title displayed on the “Dashboard” page.
grid_lines	-	option	(**enabled**) Horizontal grid lines are enabled. (**disabled**) Horizontal grid lines are disabled.
transformation_length	-	integer	Number of readouts used for computing one spectrum.
sampling_period	usec	integer	Signal sampling period.
band_low	Hz	decimal	Lower range of displayed frequency band.
band_high	Hz	decimal	Upper range of displayed frequency band.
show_max		option	(**enabled**) Show amplitude maximums. (**disabled**) Do not show amplitude maximums.

Documentation

The graph displays the latest spectrum and optionally (if enabled) displays historically maximal amplitudes. The range of visible frequencies can be adjusted by parameters or by using input fields under the graph.

The graph can be controlled using the graphical control elements located under the graph:

- **Stop time**: Stop refreshing the graph.
- **Autoscale**: The Y axis scale is adjusted automatically.
- **Hold max**: The Y axis scale is adjusted automatically and can only be expanded.
- **Keep current:** Keep the current Y axis scale.

- **Clear history**: Reset historically maximal amplitudes to zero.
- **Band [Hz]**: Set the range of the visible frequencies.

Module parameters edit

Parameter	Units	Value
type	-	Spectrum graph
name	-	dft_graph
inputs	-	[dft]
enabled	-	enabled
refresh	msec	1000
gui_page	-	Dasboard
gui_description	-	Title
grid_lines		enabled
transformation_length	-	100
sampling_period	usec	1000
band_low	Hz	0
band_high	Hz	0
show_max	-	enabled

Figure 7.14 The “Spectrum graph” module settings

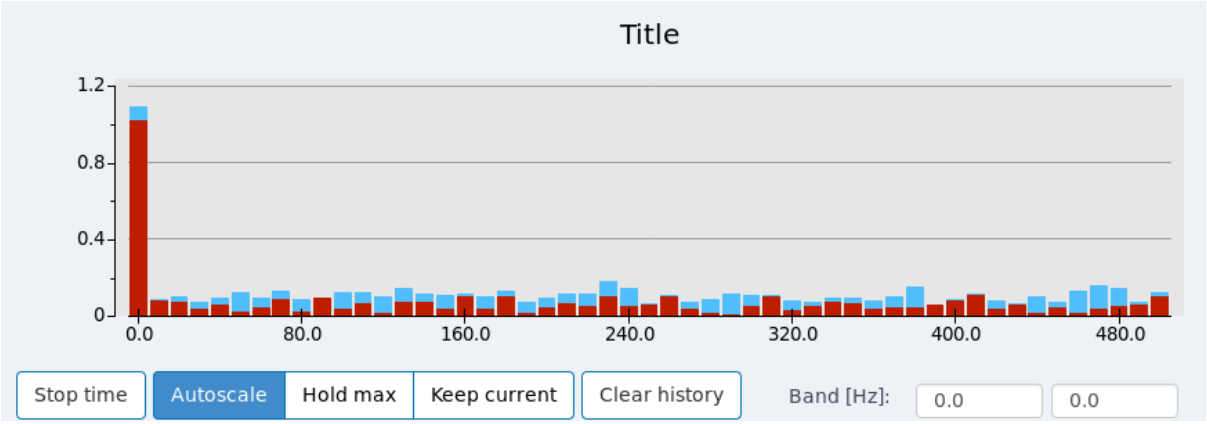


Figure 7.15 The spectrum graph on the “Dashboard” page (red columns - current spectrum, blue columns - maximal amplitudes)

7.2.11 Spectrum - Transformation Module

Description

The module computes frequency spectrum using the Fast Fourier Transformation (FFT) algorithm.

- **License**: Fundamental.
- **Category**: Fourier transformation.
- **Input**: Single value.
- **Output**: Single vector (spectrum).

Parameters

Name	Units	Type	Description
transformation_length	-	integer	Number of readouts used for computing one spectrum.
overlap	-	integer	Number of readouts from previous computation used in the new computation.
window_function	-	option	(**rectangular**) Rectangular window function. (**hamming**) Hamming window function.

7.2.12 Stream Reader Module

Description

The module reads data from a network stream.

- **License**: Fundamental.
- **Category**: Data sources.
- **Input**: None.
- **Output**: Single value, single vector.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
device	-	string	System ID.
sensor	-	string	Sensor ID.

Documentation

The module is used to receive measured data from the “Measurement System Configurator” (“MeSyCo”).

The data transfer uses the same streaming protocol as the “Stream writer“ module.

7.2.13 Stream Writer Module

Description

The module streams data over the network.

- **License**: Fundamental.
- **Category**: Data sinks.
- **Input**: Single value.
- **Output**: None.

Parameters

Name	Units	Type	Description
device	-	string	System ID.
sensor	-	string	Sensor ID.
ip_address	-	string	Network address of the target server. The address must have the *X.X.X.X* format (the only exception is localhost).
port	-	integer	Port number of the target server.
readouts_per_packet	-	integer	Number of readouts per packet.

Documentation

The module sends readouts to another "SigProc" or to other configurations in the same "SigProc".

The data transfer uses the same streaming protocol as the ”Stream reader“ module.

7.2.14 Sum Module

Description

The module adds up all inputs.

- **License**: Fundamental.
- **Category**: Basic processing.
- **Input**: Multiple values.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
multiplicative_coeficients	-	decimal array	Multiplicative coefficients of each input.
additive_coeficients	-	decimal array	Additive coefficients of each input.
output_multiplicative	-	decimal	Multiplicative coefficient of the final sum.
output_additive	-	decimal	Additive coefficient of the final sum.

Documentation

Before computation all inputs are aligned according to the timestamp. Make sure that all inputs have the same sampling frequency otherwise the alignment won't be possible and the output of the module won't be reliable.

The module computes output according to the following formula:

$$y = \sum_{i=1}^{N}\left(a_i \cdot x_i + b_i\right) \cdot A + B$$

Where y is output, N is number of inputs, a_i are *multiplicative_coeficients*, b_i are *additive_coeficients*, x_i are inputs, A is *output_multiplicative* and B is *output_additive*.

7.2.15 Threshold Alarm Module

Description

The module activates an alarm when the input value crosses the defined threshold.

- **License**: Fundamental.
- **Category**: Alarms.
- **Input**: Single value.

- **Output**: Single value (boolean).

Parameters

Name	Units	Type	Description
alarm_continual_output	-	option	(**enabled**) Always output value for every input value. (**disabled**) Output value only if the alarm state changes.
alarm_mode	-	option	(**alarm above**) Alarm is activated above the threshold value. (**alarm bellow**) Alarm is activated below the threshold value. (**alarm outside**) Alarm is activated above the high threshold or below the low threshold. (**alarm inside**) Alarm is activated between the high and low thresholds.
alarm_threshold	-	decimal	Threshold value. For alarm above and alarm below.
alarm_threshold_high	-	decimal	High level threshold value. For alarm outside and alarm inside.
alarm_threshold_low	-	decimal	Low level threshold value. For alarm outside and alarm inside.
alarm_trig_count	-	integer	Minimal number of consecutive values for alarm activation.
alarm_auto_reset_timeout	msec	integer	Timeout for automatic alarm deactivation (0 = disabled).

Documentation

The module activates an alarm when the input crosses the threshold for at least *alarm_trig_count* consecutive values.

If the auto-reset function is enabled (*alarm_auto_reset_timeout* ≠ 0), the alarm is not deactivated immediately, but it remains activated for another *alarm_auto_reset_timeout* milliseconds before being deactivated. This function basically extends the duration of the activated alarm.

7.2.16 Time Average Module

Description

The module computes average over the specified time interval.

- **License**: Fundamental.
- **Category**: Basic processing.
- **Input**: Single value, single vector.
- **Output**: Single value, single vector.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
average_duration	msec	integer	Time interval for averaging.

7.2.17 Value Table Module

Description

The module displays values in a table.

- **License**: Fundamental.
- **Category**: Visualization.
- **Input**: Single value, single vector, multiple values.
- **Output**: None.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
gui_description	-	string	Title displayed on the “Dashboard” page.
gui_names	-	string array	Input signal names displayed on the “Dashboard” page. Leave empty if you want to use the default names.
gui_units	-	string	Units displayed on the “Dashboard” page.

Documentation

The *width* parameter is used only for “single vector” input type. If the input is single value or multiple values then the *width* parameter must be set to 1.

Module parameters edit

Parameter	Units	Value
type	-	Value table
name	-	value_table
inputs	-	[sensor_01, sensor_02, sensor_03]
enabled	-	enabled
width	-	1
refresh	msec	1000
gui_page	-	Dashboard
gui_description	-	Temperature sensors
gui_names	-	[sensor 1, sensor 2, sensor 3]
gui_units	-	degC

Figure 7.16 The “Value table” module settings

Temperature sensors

sensor 1 [degC]	sensor 2 [degC]	sensor 3 [degC]
1.0947750172091...	8.9387513130845...	5.2453497111902...

Figure 7.17 Table on the “Dashboard” page

7.3 Modules under Fundamental Plus Licence

7.3.1 Complex Threshold Alarm Module

Description

The module detects incidents in the input signal and activates an alarm when the configured conditions are met.

- **License**: Fundamental Plus.
- **Category**: Alarms.
- **Input**: Single value.
- **Output**: Single value (boolean).

Parameters

Name	Units	Type	Description
alarm_continual_output	-	option	(**enabled**) Always output value for every input value. (**disabled**) Output value only if the alarm state changes.
alarm_mode	-	option	(**alarm above**) Alarm is raised when the input value is above the threshold. (**alarm bellow**) Alarm is raised when the input value is below the threshold.
alarm_threshold	-	decimal	Threshold value.
alarm_trig_count	-	integer	Minimal number of incidents for raising an alarm.
alarm_max_spacing	msec	integer	Maximal time interval between incidents.
alarm_recovery_time	msec	integer	Time duration with no sensitivity after an incident.
alarm_min_duration	msec	integer	Minimal time duration of an incident.
alarm_max_duration	msec	integer	Maximal time duration of an incident.
alarm_max_duration_enabled	-	option	(**disabled**) Accept all incidents. (**enabled**) Accept only incidents shorter than the alarm_max_duration.
alarm_auto_reset_timeout	msec	integer	Timeout for automatic alarm reset (0 = disabled).

7.3.2 FIR Filter Module

Description

The module implements a finite impulse response filter (FIR).

- **License**: Fundamental Plus.
- **Category**: Advanced processing.
- **Input**: Single value.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
impulse_response	-	decimal array	Filter coefficients.

7.3.3 Hysteresis Threshold Alarm Module

Description

The module activates an alarm when the input value crosses the defined threshold.

- **License**: Fundamental Plus.
- **Category**: Alarms.
- **Input**: Single value.
- **Output**: Single value (boolean).

Parameters

Name	Units	Type	Description
alarm_continual_output	-	option	(**enabled**) Always output value for every input value. (**disabled**) Output value only if the alarm state changes.
alarm_mode	-	option	(**alarm above**) Alarm is raised when the input value is above the threshold. (**alarm bellow**) Alarm is raised when the input value is below the threshold.
high	-	decimal	High level threshold value.
low	-	decimal	Low level threshold value.

Documentation

If the *alarm_mode* is set to “alarm above” then the alarm is activated when the input value rises above the high threshold and deactivated when the input value drops below the low threshold.

If the *alarm_mode* is set to “alarm below” then the alarm is activated when the input value drops below the low threshold and deactivated when the input value rises above the high threshold.

7.3.4 Line Graph - Vector Module

Description

The module displays the input vectors in a multiple line graph.

- **License**: Fundamental Plus.
- **Category**: Visualization.
- **Input**: Single vector.
- **Output**: None.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
gui_description	-	string	Title displayed on the “Dashboard” page.
gui_name	-	string	Input signal name displayed on the “Dashboard” page.
gui_units	-	string	Units displayed on the “Dashboard” page.
grid_lines	-	option	(**enabled**) Horizontal grid lines are enabled. (**disabled**) Horizontal grid lines are disabled.
buffer_size	-	integer	Number of vectors in the graph.

Documentation

The graph shows the time evolution of the input vector. The timeline is displayed in color. The newest vector is red and the oldest one is blue.

Module parameters edit

Parameter	Units	Value
type	-	Line graph - vector
name	-	graph_vector
inputs	-	[sensor_vector]
enabled	-	enabled
width	-	4
refresh	msec	1000
gui_page	-	Line graph
gui_description	-	Title
gui_name	-	profile
gui_units	-	mm
grid_lines		enabled
buffer_size	-	10

Figure 7.18 The “Line graph - vector” module settings

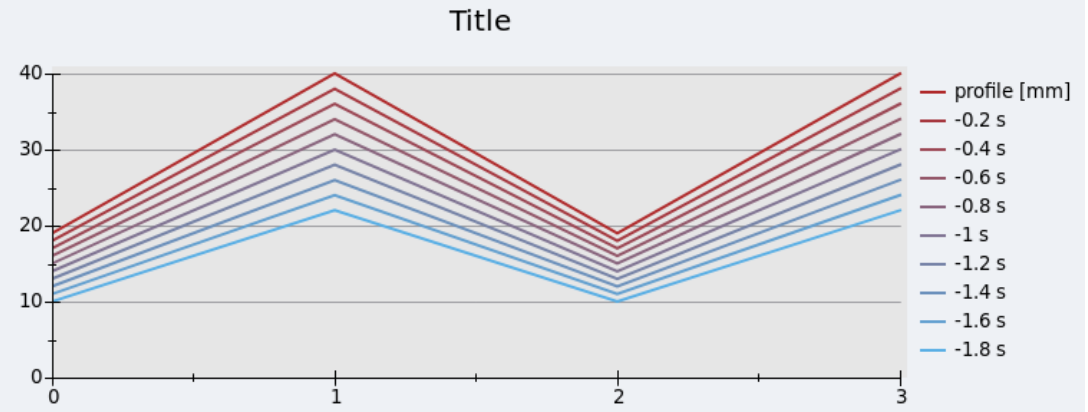


Figure 7.19 Line graph on the “Dashboard” page

7.3.5 Linear Regression Module

Description

The module fits a linear function ($y = A{\cdot}x + B$) on a series of a consecutive readouts. The output is a coefficient A.

- **License**: Fundamental Plus.
- **Category**: Advanced processing.
- **Input**: Single value.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
length	-	integer	Number of readouts in a series. One series is used for one regression analysis.
overlap	-	integer	Number of readouts shared by two consecutive series.

7.3.6 Moving Average Module

Description

The module computes a moving average.

- **License**: Fundamental Plus.
- **Category**: Basic processing.
- **Input**: Single value, single vector.
- **Output**: Single value, single vector.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
average_length	-	integer	Number of averaged values.

7.3.7 Polynomial Transformation Module

Description

The module computes polynomial transformation.

- **License**: Fundamental Plus.
- **Category**: Advanced processing.
- **Input**: Single value.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
coefficients	-	decimal array	Coefficients ordered by descending powers. Example: [5, 3, 2.4] -> 5x^2 + 3x + 2.4

Documentation

The module computes output according to the following formula:

$$y = a_1 \cdot x^{N-1} + a_2 \cdot x^{N-2} + \ldots + a_N$$

Where y is output, $x$is input, N is degree of the polynomial and ai are *coefficients*.

7.3.8 Signal Modifier Module

Description

The module modifies one signal by another signal. The first input is the controlling signal and the second one is the modified signal.

- **License**: Fundamental Plus.
- **Category**: Special.
- **Input**: 2x single value.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
process_period	msec	integer	Time period of processing inputs.
control_type	-	option	(**add**) Add value. (**multiply**) Multiply by value. (**mask**) Let pass through. (**event**)

			Create an event.
time_before_event	msec	integer	Time interval before the event starts. Used only for the event option.
time_after_event	msec	integer	Time interval after the event ends. Used only for the event option.

Documentation

The modified signal is changed according to the value of the controlling signal. The timestamp of the modified signal remains unchanged, only the numeric value changes.

There are four types of modification available:

- **Add**: The value of the controlling signal is added to the modified signal.
- **Multiply**: The modified signal is multiplied by the controlling signal.
- **Mask**: The modified signal passes through the module only when the controlling signal is nonzero.
- **Event**: An event is captured on a modified signal when the controlling signal is nonzero.

Event Description

The event modification is similar to the mask modification. The signal passes through the module when the controlling signal is nonzero. The difference is that the time interval after which the modified signal is passed is at the beginning extended by the *time_before_event* parameter and at the end extended by the *time_after_event* parameter. The purpose of extending this interval is to capture not only the part of the signal that preceded this event but also the part of the signal that followed the end of this event.

The first output value of each event is assigned a special tag (“not visible for a user”) that the “File writer” module can detect in order to create a new file. Each event is stored in a separate file.

Note: The event option introduces a propagation delay of *time_before_event* milliseconds.

Each input signal may have a different sampling frequency and therefore the input values are processed periodically in batches. The processing period is set by the *process_period* parameter.

7.3.9 Signals to Vector Convertor Module

Description

The module aligns data from multiple inputs and creates one single vector.

- **License**: Fundamental Plus.
- **Category**: Special.
- **Input**: Multiple values.

- **Output**: Single vector.

Parameters

Name	Units	Type	Description
testing	-	option	(**disabled**) Log only basic messages. (**enabled**) Log verbose messages.
buffer_size	-	integer	Buffer size for each input.
max_time_diff	usec	integer	Maximum allowed difference in the timestamp.
schedule_policy	-	option	(**all_data**) Inputs are processed when all inputs have at least one value. (**partia_data**) Inputs are processed when at least one input has a value.
output_policy	-	option	Policy for a situation when the inputs cannot be aligned. (**only valid**) Discard values. (**fill zeros**) Fill zeros. (**repeat last**) Repeat last valid output value.

Documentation

Input values are internally buffered before processing. The size of the buffer is defined by the *buffer_size* parameter. Larger buffer size usually allows for better alignment but at the expense of the higher propagation delay. If the input values came from a stream, then the *buffer_size* should be bigger than the number of readouts in one packet (*readouts_per_packet*).

The alignment is performed according to the timestamps. The readouts are aligned together if their timestamps differ by less then the *max_time_diff*. A half of the sampling period is usually a good value for the *max_time_diff*.

7.3.10 Spectrum - Band Filter Module

Description

The module works as a band-pass filter. It zeros out all frequencies outside of the given frequency band.

- **License**: Fundamental Plus.
- **Category**: Fourier transformation.
- **Input**: Single vector (spectrum).
- **Output**: Single vector (spectrum).

Parameters

Name	Units	Type	Description
transformation_length	-	integer	Number of readouts used for computing one spectrum.
sampling_period	usec	integer	Signal sampling period.
band_low	Hz	decimal	Lower range of the frequency band.
band_high	Hz	decimal	Upper range of the frequency band.

7.3.11 Spectrum - Band Sum Module

Description

The module computes the sum of amplitudes in the given frequency band.

- **License**: Fundamental Plus.
- **Category**: Fourier transformation.
- **Input**: Single vector (spectrum).
- **Output**: Single value.

Parameters

Name	Units	Type	Description
testing	-	option	(**disabled**) Log only basic messages. (**enabled**) Log verbose messages.
transformation_length	-	integer	Number of readouts used for computing one spectrum.
sampling_period	usec	integer	Signal sampling period.
band_low	Hz	decimal	Lower range of the frequency band.
band_high	Hz	decimal	Upper range of the frequency band.

7.3.12 Stream Writer - Vector Module

Description

The module streams data over the network.

- **License**: Fundamental Plus.
- **Category**: Data sinks.
- **Input**: Single vector.
- **Output**: None.

Parameters

Name	Units	Type	Description
device	-	string	System ID.
sensor	-	string	Sensor ID.
ip_address	-	string	Network address of the target server. The address must have the *X.X.X.X* format (the only exception is localhost).
port	-	integer	Port number of the target server.

Documentation

The module sends readouts to another "SigProc" or to other configurations in the same "SigProc".

The data transfer uses the same streaming protocol as the “Stream reader” module.

7.3.13 Sum - Vector Module

Description

The module adds up all elements of the input vector.

- **License**: Fundamental Plus.
- **Category**: Basic processing.
- **Input**: Single vector.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
multiplicative_coeficients	-	decimal array	Multiplicative coefficients of each element.
additive_coeficients	-	decimal array	Additive coefficients of each element.
output_multiplicative	-	decimal	Multiplicative coefficient of the final sum.
output_additive	-	decimal	Additive coefficient of the final sum.

Documentation

The module computes output according to the following formula.

$$y = \sum_{i=1}^{N}\left(a_i \cdot x_i + b_i\right) \cdot A + B$$

Where y is output, N is size of input vector, a_i are *multiplicative_coeficients*, b_i are *additive_coeficients*, x_i are elements of the input vector, A is *output_multiplicative* and B is *output_additive*.

7.3.14 User Input Module

Description

The module creates graphical control elements on the “Dashboard” page. The controls read input from the user and generate value on the output of the module.

- **License**: Fundamental Plus.
- **Category**: Visualization.
- **Input**: None.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
gui_description	-	string	Title displayed on the “Dashboard” page.
gui_name	-	string	Input signal name displayed on the “Dashboard” page.

input_type	-	option	(**toggle switch**) Toggle switch. (**number**) Input for a number. (**button**) Button.

Documentation

"Toggle switch" generates 1 when toggled and 0 when untoggled. "Number" generates a number entered in the input field. "Button" generates 1 when pressed. If the parameter *gui_description* is not empty then the button asks for confirmation before generating the value.

Module parameters edit

Parameter	Units	Value
type	-	User input
name	-	user_input_01
enabled	-	enabled
refresh	msec	1000
gui_page	-	Other GUI
gui_description	-	
gui_name	-	Simulate alarm
input_type		toggle switch

Figure 7.20 The "User input" module settings - toggle switch

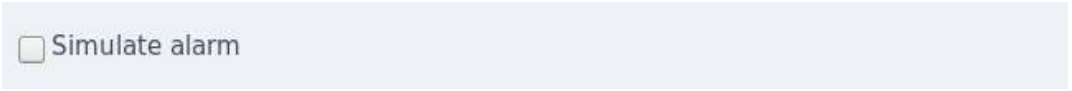


Figure 7.21 Checkbox on the "Dashboard" page

Module parameters edit

Parameter	Units	Value
type	-	User input
name	-	user_input_02
enabled	-	enabled
refresh	msec	1000
gui_page	-	Other GUI
gui_description	-	
gui_name	-	Insert number:
input_type		number

Figure 7.22 The "User input" module settings - number

Figure 7.23 Input field on the “Dashboard” page

Module parameters edit

Parameter	Units	Value
type	-	User input
name	-	user_input_03
enabled	-	enabled
refresh	msec	1000
gui_page	-	Other GUI
gui_description	-	Do you realy want to reset the counter?
gui_name	-	Reset counter
input_type		button

Figure 7.24 The “User input” module settings - button

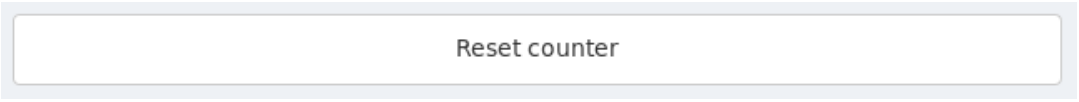


Figure 7.25 Button on the “Dashboard” page

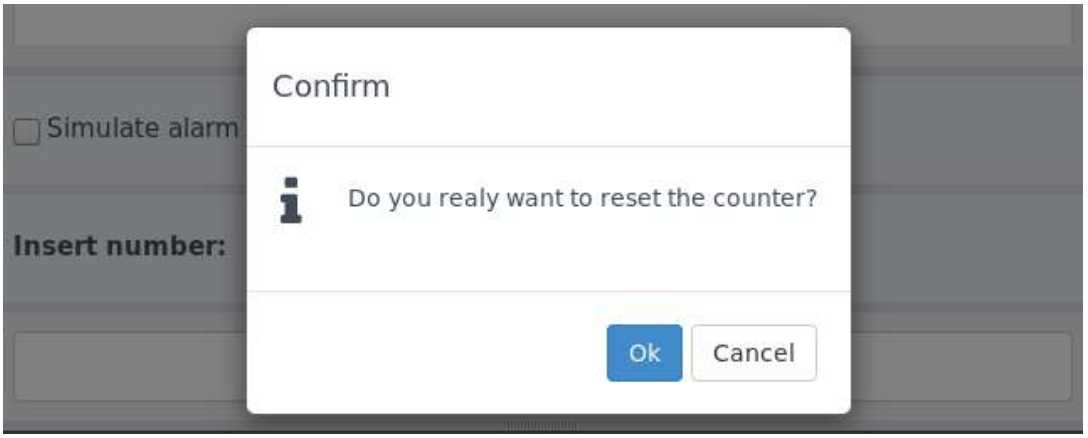


Figure 7.26 Confirmation dialog box

7.4 Modules under Advanced Licence

7.4.1 Column Graph Module

Description

The module displays inputs in a column graph.

- **License**: Advanced.
- **Category**: Visualization.
- **Input**: Single value, multiple values, single vector.
- **Output**: None.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
gui_description	-	string	Title displayed on the “Dashboard” page.
gui_names	-	string array	Input signal names displayed on the “Dashboard” page. Leave empty if you want to use the default names.
gui_units	-	string	Units displayed on the “Dashboard” page.
grid_lines	-	option	(**enabled**) Horizontal grid lines are enabled. (**disabled**) Horizontal grid lines are disabled.
y_axis_scale	-	option	(**linear**) Linear scale for Y axis. (**logarithmic**) Logarithmic scale for Y axis.
offset	-	decimal	The value is added to all inputs before displaying.

Documentation

The graph can be controlled using the graphical control elements located under the graph:

- **Stop time**: Stop refreshing the graph.
- **Autoscale**: The scale is adjusted automatically.
- **Hold min/max**: The scale is adjusted automatically but it can only be expanded.
- **Keep current**: Keep the current scale.

If the module has only one input or if the input vector has width 1, then the module displays a horizontal bar graph with one bar.

Module parameters edit

Parameter	Units	Value
type	-	Column graph
name	-	column_graph
inputs	-	[sensor_01, sensor_02, sensor_03]
enabled	-	enabled
width	-	1
refresh	msec	1000
gui_page	-	Dashboard
gui_description	-	Title
gui_names	-	[length_01, length_02, length_03]
gui_units	-	mm
grid_lines		enabled
y_axis_scale	-	linear
offset	-	0

Figure 7.27 The “Column graph” module settings - multiple columns

Figure 7.28 Autoscale - multiple columns

Module parameters edit

Parameter	Units	Value
type	-	Column graph
name	-	bar_graph
inputs	-	[sensor_01]
enabled	-	enabled
width	-	1
refresh	msec	1000
gui_page	-	Dashboard
gui_description	-	Title
gui_names	-	[]
gui_units	-	
grid_lines		enabled
y_axis_scale	-	linear
offset	-	0

Figure 7.29 The “Column graph” module settings - one bar

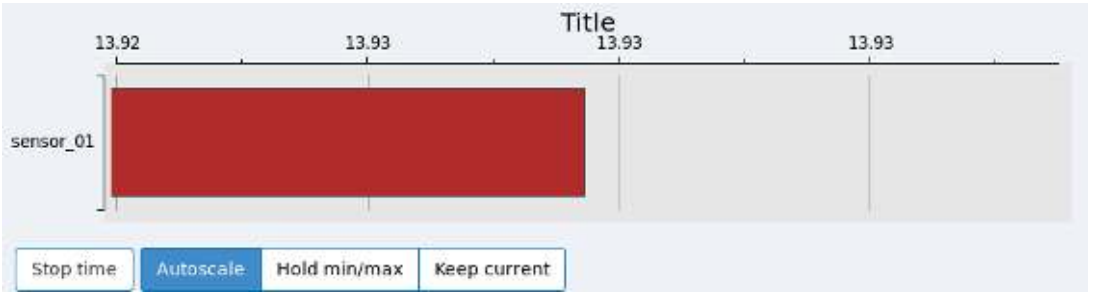


Figure 7.30 Autoscale - one bar

7.4.2 Command Sender Module

Description

The module sends commands to the other modules.

- **License**: Advanced.
- **Category**: Special.
- **Input**: Single value.
- **Output**: None.

Parameters

Name	Units	Type	Description
target_modules	-	string array	List of the target modules which will receive the command.
click_event_ids	-	integer array	Command IDs sent to the target modules when input value is 2 (click).
check_event_ids	-	integer array	Command IDs sent to the target modules when input value is 1 (check).
uncheck_event_ids	-	integer array	Command IDs sent to the target modules when input value is 0 (uncheck).

Documentation

The module sends command IDs to all modules listed in the *target_modules*. If the input value is 0 the module sends command IDs defined in the *click_event_ids*. If the input is 1 the module sends command IDs defined in the *check_event_ids*. If the input is 2 the module sends command IDs defined in the *uncheck_event_ids*.

This module was created to be used in combination with the "User input" module, hence the expected input values. The input value 0 represents a click on a button, value 1 represents a check of a checkbox and value 2 represents an uncheck of a checkbox.

Please note that only some modules are able to receive a command:

- "Inclination profile" module
- "Alarm counter" module
- "Min/max holder" module

7.4.3 Data Stop Alarm Module

Description

The module checks the sampling period of the input signal and activates an alarm if the interval between the readouts is longer than a predefined value.

- **License**: Advanced.
- **Category**: Alarms.
- **Input**: Single value.
- **Output**: Single value (boolean).

Parameters

Name	Units	Type	Description
alarm_continual_output	-	option	(**enabled**) Always output value for every input value. (**disabled**) Output value only if the alarm state changes.
time_period	-	sec, msec	Time interval for activating the alarm.
units	-	option	Units of the parameter time_period. (**s**) Seconds. (**ms**) Milliseconds.

7.4.4 Data Table - Vector Module

Description

The module displays the input vectors in a table.

- **License**: Advanced.
- **Category**: Visualization.
- **Input**: Singe value, single vector.
- **Output**: None.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
gui_description	-	string	Title displayed on the “Dashboard” page.
buffer_size	-	integer	Number of vectors displayed in the table.

Module parameters edit

Parameter	Units	Value
type	-	Data table - vector
name	-	table_vector
inputs	-	[sensor_vector_01]
enabled	-	enabled
width	-	3
refresh	msec	1000
gui_page	-	Dashboard
gui_description	-	Title
buffer_size	-	10

Figure 7.31 The “Data table - vector” module settings

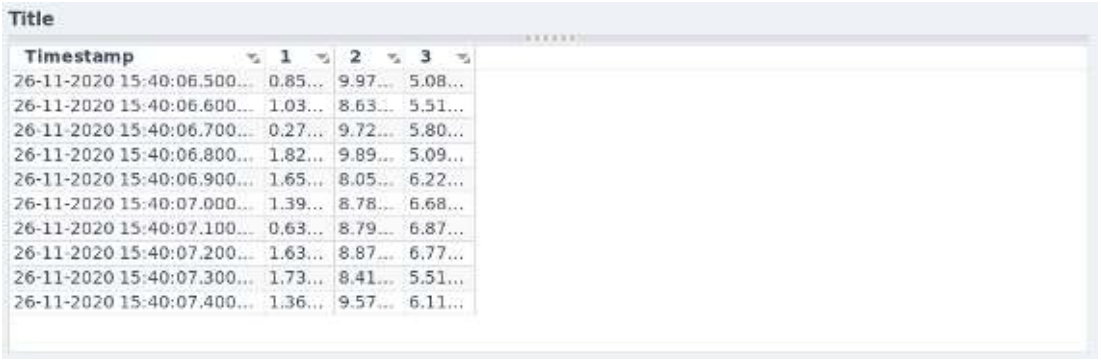


Figure 7.32 Table on the “Dashboard” page

7.4.5 Fourier Filter Module

Description

The module works as a band-pass filter. The filtering is performed in the frequency spectrum.

- **License**: Advanced.
- **Category**: Fourier transformation.
- **Input**: Single value.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
transformation_length	-	integer	Number of readouts used for computing one spectrum.
sampling_period	usec	integer	Signal sampling period.
overlap	-	integer	Number of readouts from previous computation used in the new computation.
window_function	-	option	(**rectangular**) Rectangular window function. (**hamming**) Hamming window function.
band_low	Hz	decimal	Low frequency of the band.
band_high	Hz	decimal	High frequency of the band.

7.4.6 Min/Max Holder Module

Description

The module holds the extreme value of the input signal and pushes the value to the output.

- **License**: Advanced.
- **Category**: Advanced processing.
- **Input**: Single value.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
continual_output	-	option	(**enabled**) Always output value for every input value. (**disabled**) Output value only if the minimum/maximum changes.
hold_type	-	option	(**min**) Hold minimum. (**max**) Hold maximum.
default_value	-	decimal	Default starting value.

Documentation

The output is the minimum/maximum of the input since the configuration started or since the last reset. The output starts at the *default_value* and it is updated only when the input gets lower/higher than the previous minimum/maximum.

The module can receive commands from the "Command sender" module.

List of command IDs:

- **1**: Reset the current minimum/maximum to the *default_value* and push the value to the output.

7.4.7 PostgreSQL Writer Module

Description

The module writes data to the "PostgreSQL" database.

- **License**: Advanced.
- **Category**: Data sinks.
- **Input**: Single vector.
- **Output**: None.

Parameters

Name	Units	Type	Description
testing	-	option	(**disabled**) Log only basic messages. (**enabled**) Log verbose messages.
width	-	integer	Width of the input vector.
connection_id	-	string	Database Connection ID.
measurement	-	string	Table name (mandatory).
serial_number	-	string	Primary key value (mandatory).
sensor_id	-	string	Primary key value (mandatory).
units	-	string	Column value (optional).
note	-	string	Column value (optional).

Documentation

The *connection_id* parameter is an unique identifier of a database connection created and configured on the "Database manager" page.

Before using the module, it is necessary to set up the database connection correctly and have the "PostgreSQL table" already created in the database. "SigProc" cannot create this table automatically.

Example: Creating the "PostgreSQL table" on the "Linux" server

1. Switch user to postgres.

```
[user@localhost ~]$ su postgres
```

2. Open the psql terminal.

```
bash-4.2$ psql
```

3. Connect to a database (In this example the name of the database is "demo".).

```
postgres=# \c demo
```

4. Create a new "PostgreSQL table" (In this example the name of the table is "test_measurement".).

```
create table test_measurement (
"serial_number" text,
"sensor_id" text,
"time" timestamp,
"value" double precision [],
"units" text,
"note" text,
primary key ("serial_number", "sensor_id", "time"));
```

5. Done! The following steps are optional.

6. List all tables in the database.

```
demo=# \d
          List of relations
 Schema |   Name   | Type  |  Owner
--------+------------------+-------+----------
 public | test_measurement | table | postgres
(1 row)
```

7. Show the content of the table.

```
demo=# select * from test_measurement;
 serial_number | sensor_id | time | value | units | note
---------------+-----------+------+-------+-------+------
(0 rows)
```

Database connection edit

Parameter	Units	Value
db_type		PostgreSQL
connection_id		demo_connection
server_address		10.23.23.10
server_port		5432
user		postgres
password		
db_name		demo
sending_period	sec	1
testing		enabled
backup_buffer_enabled		enabled
backup_buffer_path		/home/sigproc/gdb_data/
backup_buffer_sync_count		10
backup_buffer_load_count		10

Figure 7.33 The "PostgreSQL" database connection settings

Module parameters edit

Parameter	Units	Value
type	-	PostgreSQL writer
name	-	postgresql
inputs	-	[sensor_01]
enabled	-	enabled
testing	-	disabled
width	-	1
connection_id	-	demo_connection
measurement	-	test_measurement
serial_number	-	ID123465
sensor_id	-	Length_01
units	-	mm
note	-	bridge

Figure 7.34 The “PostgreSQL writer” module settings

Example: The contents of the table in the database

```
demo=# select * from test_measurement;
 serial_number | sensor_id |         time        |         value        | units |  note
---------------+-----------+---------------------+----------------------+-------+--------
ID123465       | Length_01 | 2020-11-06 16:38:38 | {1.79906045628118}   | mm    | bridge
ID123465       | Length_01 | 2020-11-06 16:38:39 | {0.376679114793636}  | mm    | bridge
ID123465       | Length_01 | 2020-11-06 16:38:40 | {0.851534045557006}  | mm    | bridge
(3 rows)
```

7.4.8 Sampling Speed Module

Description

The module computes the sampling speed between two consecutive readouts. The output can be a frequency or a period.

- **License**: Advanced.
- **Category**: Advanced processing.
- **Input**: Single value.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
sampling_speed_type	-	option	(**frequency [Hz]**) The output is frequency. (**period [sec]**) The output is period.

7.4.9 Spectrum - Binary Module

Description

The module converts a frequency spectrum to a binary form.

- **License**: Advanced.
- **Category**: Fourier transformation.
- **Input**: Single vector (spectrum).
- **Output**: Single value, single vector.

Parameters

Name	Units	Type	Description
transformation_length	-	integer	Number of readouts used for computing one spectrum.
window_count	-	integer	Number of windows.
thresholds	-	decimal array	List of thresholds for each window. If the size of the list is not equal to the window_count then the first value in the list is used for all windows.
comparison_mode	-	option	(**maximum**) The maximum in a window is compared to the threshold. (**sum**) Sum of all values in a window is compared to the threshold.
output	-	option	(**number**) Spectrum in a binary form is converted into a single number. (**vector**) Spectrum in a binary form remains as a vector.

Documentation

The spectrum is converted to binary form as follows:

1. The DC component is removed.
2. The rest of the spectrum is divided into equally sized windows (frequency bands). Make sure that the *transformation_length* is completely divisible by (2 * *window_count*). Otherwise the windows won't have the same size.
3. For each window find the characteristic value. The value can be a sum of all amplitudes in the window or the maximum amplitude in the window.
4. For each window compare the characteristic value with the threshold. If the value is above the threshold the binary representation of the window is 1. If the value is below the threshold the binary representation is 0.
5. The sequence of all binary representations is the spectrum in the binary form.

7.4.10 Spectrum - Max Amplitude Module

Description

The module finds the highest amplitude in the frequency spectrum.

- **License**: Advanced.
- **Category**: Fourier transformation.
- **Input**: Single vector (spectrum).
- **Output**: Single value.

Parameters

Name	Units	Type	Description
transformation_length	-	integer	Number of readouts used for computing one spectrum.
sampling_period	usec	integer	Signal sampling period.
min_amplitude	-	decimal	Minimum allowed amplitude. Smaller is ignored.
max_amplitude	-	decimal	Maximum allowed amplitude. Bigger is ignored.

Documentation

If the highest amplitude is outside of the allowed range then the module produces no output.

7.4.11 Spectrum - Max Frequency Module

Description

The module finds a frequency with the highest amplitude.

- **License**: Advanced.
- **Category**: Fourier transformation.
- **Input**: Single vector (spectrum).
- **Output**: Single value.

Parameters

Name	Units	Type	Description
transformation_length	-	integer	Number of readouts used for computing one spectrum.
sampling_period	usec	integer	Signal sampling period.
min_amplitude	-	decimal	Minimum allowed amplitude. Smaller is ignored.
max_amplitude	-	decimal	Maximum allowed amplitude. Bigger is ignored.

Documentation

If the highest amplitude is outside of the allowed range then the module produces no output.

7.4.12 Vector Min/Max Module

Description

The module finds the minimum or the maximum in the input vector.

- **License**: Advanced.
- **Category**: Advanced processing.
- **Input**: Single vector.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
mode	-	option	(**maximum**) Find the maximum. (**minimum**) Find the minimum. (**absolute_maximum**) Calculate the absolute value of each element of the vector and then find the maximum. (**absolute_minimum**) Calculate the absolute value of each element of the vector and then find the minimum.

7.4.13 Zabbix Module

Description

The module sends data to the "Zabbix" system.

- **License**: Advanced.
- **Category**: Third party integration.
- **Input**: Single value.
- **Output**: None.

Parameters

Name	Units	Type	Description
zabbix_channel_id	-	string	Channel identifier. Must match with the "Zabbix" configuration.
zabbix_unit_id	-	string	Unit identifier. Must match with the "Zabbix" configuration.

Documentation

For the module to work properly, the “Zabbix driver” must be configured in the “System configuration file”.

7.5 Modules under Expert Licence

7.5.1 Alarm Counter Module

Description

The module counts alarms. The counter is incremented on every transition from zero to nonzero value.

- **License**: Expert.
- **Category**: Alarms.
- **Input**: Single value (boolean).
- **Output**: Single value.

Parameters

Name	Units	Type	Description
alarm_continual_output	-	option	(**enabled**) Always output value for every input value. (**disabled**) Output value only if the counter changes.

Documentation

The module can receive commands from the "Command sender" module.

List of command IDs:

- **1**: Reset the counter and push zero value to the output.

7.5.2 Alarm Image Module

Description

The module displays numeric values and alarms in an image.

- **License**: Expert.
- **Category**: Visualization.
- **Input**: Multiple values.
- **Output**: None.

Parameters

Name	Units	Type	Description
pairs_value		string array	Numeric inputs. They are paired with the “pairs_alarms” parameter.
pairs_alarm		string array	Alarm inputs. They are paired with the “pairs_value” parameter.
signal_names		string array	Displayed signal names.
signal_units		string array	Displayed signal units.
coordinates_x	px	integer array	Box center coordinates. The number of pixels from the left edge.
coordinates_y	px	integer array	Box center coordinates. The number of pixels from the top.
box_width	px	integer	Box width.
box_height	px	integer	Box height.
font_size	-	option	(**small**) Small font size. (**big**) Big font size.
image_path	-	string	Path to the image.

Documentation

The module can display the alarm state, the numerical value or both together in the image. The module has only one parameter to define its inputs, therefore both the alarm inputs and the numeric inputs must be defined in the *inputs* parameter. Pairing the alarm inputs with the numeric inputs is done using the *pairs_value* and *pairs_alarms* parameters. Both parameters must have the same length, so if an input is not in a pair then the array contains an empty string.

Activated alarms are displayed in red, deactivated alarms in green. If the alarm input hasn't received any value yet, it is displayed in light blue color. The units displayed next to the numeric value can contain special characters, but they have to be encoded in a special way using dollar symbols. Conversion table for special characters:

Encoded character	Displayed result
mu	µ
$epsilon$	ε
$degree$	°
$squote$	'
$dquote$	"

Module parameters edit

Parameter	Units	Value
type	-	Alarm image
name	-	image
inputs	-	[alarm_01, alarm_02, sensor_01, sensor_02]
enabled	-	enabled
refresh	msec	1000
gui_page	-	Image
pairs_value	-	[sensor_01, sensor_02,]
pairs_alarm	-	[, alarm_01, alarm_02]
signal_names	-	[Temperature, Dilatation, Fiber cut]
signal_units	-	[$degree$C, mum,]
coordinates_x	px	[216, 635, 1055]
coordinates_y	px	[382, 382, 382]
box_width	px	170
box_height	px	30
font_size	-	big
image_path	-	/home/sigproc/config/bridge.png

Figure 7.35 The “Alarm image” module settings

The following table shows pairs and their names defined by the *pair_value*, *pair_alarm* and *signal_names* parameters.

	pair_value	***pair_alarm***	***signal_names***	**Content of the box**
1	sensor_01	---	Temperature	Only numeric value.
2	sensor_02	alarm_01	Dilatation	Numeric value and alarm state.
3	---	alarm_02	Fiber cut	Only alarm state.

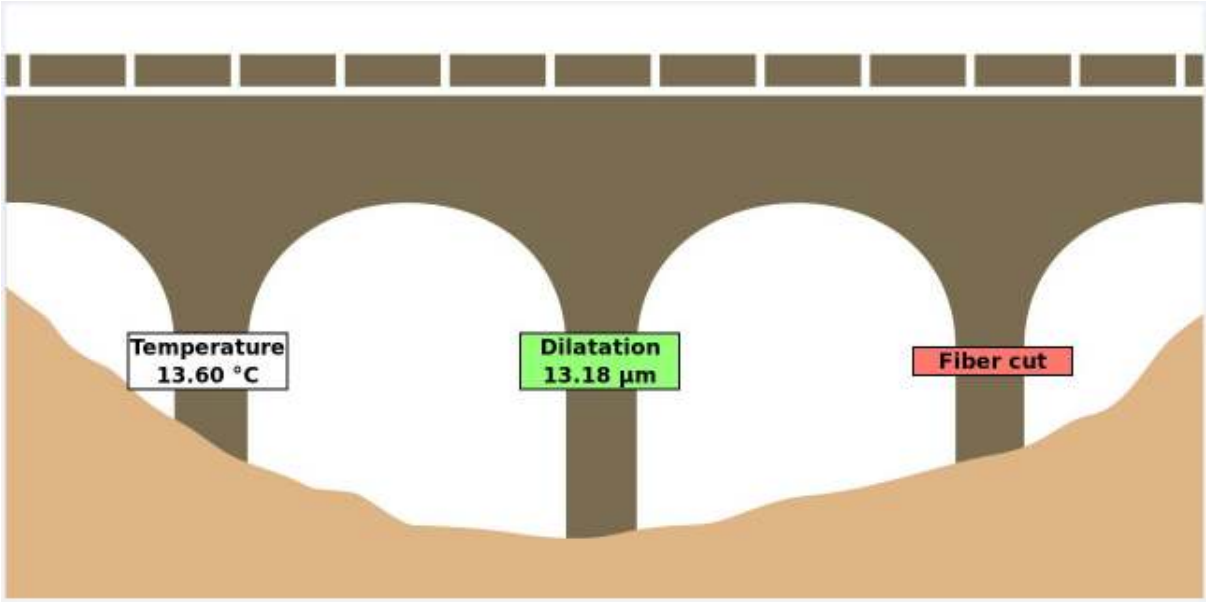


Figure 7.36 Image on the “Dashboard” page

7.5.3 Auxiliary Data Extractor Module

Description

The module extracts auxiliary data from the input and pushes the result to the output.

- **License**: Expert.
- **Category**: Special.
- **Input**: Single value, single vector.
- **Output**: Single value, single vector.

Parameters

Name	Units	Type	Description
testing	-	option	(**disabled**) Log only basic messages. (**enabled**) Log verbose messages.
width	-	integer	Width of the input vector.
output_width	-	integer	Width of the output vector.
aux_data_index	-	integer	Index of the auxiliary value. Starts from zero.
aux_data_ident	-	string	Name of the auxiliary value.

Documentation

Some modules that perform complex computation provide auxiliary data in their outputs. The auxiliary data contain intermediate results of the computation.

The auxiliary data may contain multiple values. In order to extract only a specific value, the values are named and placed in an indexed array. The selection of the extracted value is done by the *aux_data_index* or *aux_data_ident* parameters. Only one of these two parameters needs to be set.

Standard modules send only the calculated result to the output. Unfortunately, this is not always enough for the modules with complex calculations and sometimes it is necessary to have intermediate results. Therefore, some modules provide together with the output value also intermediate results, usually referred to as auxiliary data. As there can be several auxiliary values for one output value, in order to distinguish them from each other, they are named and placed in an indexed list.

A specific auxiliary value (intermediate result) can be extracted either by its name or by its index. The name is set in the *aux_data_ident* and the index is set in *aux_data_index* parameter. Only one of these two parameters needs to be set.

List of modules that generate auxiliary data:

- "Dynamic threshold alarm" module

7.5.4 Duty Cycle Module

Description

The module computes a duty cycle over a specified time period.

- **License**: Expert.
- **Category**: Advanced processing.
- **Input**: Single value.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
cycle_duration	msec	integer	Time period of one cycle.
threshold	-	decimal	Threshold value.
max_duty_value	-	decimal	Multiplicative coefficient of the computed duty cycle.

Documentation

Duty cycle is the ratio between the time when the signal is above the threshold and the time when the signal is below the threshold. If the signal is above the threshold for the whole period then the ratio is 1.

7.5.5 Dynamic Threshold Alarm Module

Description

The module detects alarm based on an internally computed threshold values. The module is also able to adapt to slow changes in the input signal.

- **License**: Expert.
- **Category**: Alarms.
- **Input**: Single value.
- **Output**: Single value (boolean).

Parameters

Name	Units	Type	Description
testing	-	option	(**disabled**) Log only basic messages. (**enabled**) Log verbose messages.
block_length	msec	integer	Time interval for finding extremal values (no overlap).
block_count	-	integer	Number of blocks for distribution.
pot_buffer_size	-	integer	Size of circular buffer for POT distribution.
percentil_bd	%	decimal	Percentil for computing threshold for declustering.
alpha	-	decimal	Coefficient for calculating the threshold, ThLow = p60 + alpha * (p60 - p30).
beta	-	decimal	Coefficient for calculating the threshold, ThHigh = p60 + beta * (p60 - p30).
parameter_file_path	-	string	Path to the file with parameters. Leave empty if the file is not used.
aux_data_output	-	option	(**disabled**) Auxiliary data is disabled. (**enabled**) Add auxiliary data to the output. [input, delta, th_low, th_high, alarm_count].

7.5.6 Google Maps Module

Description

The module shows alarms on Google Maps.

- **License**: Expert.
- **Category**: Visualization.
- **Input**: Multiple values (boolean).
- **Output**: None.

Parameters

Name	Units	Type	Description
alarm_names	-	string array	Displayed alarm names (optional).
alarm_latitude	deg	decimal array	Alarm coordinates.
alarm_longtitude	deg	decimal array	Alarm coordinates.
focus_zoom_lvl	-	integer	Zoom level when focusing alarms.
map_width	px	integer	Map width.
map_height	px	integer	Map height.

Documentation

Module expects the input to be a value 0 or 1. Value 0 stands for an inactive alarm displayed in green color and value 1 stands for an active alarm displayed in red. For an input that hasn't received any value yet, the alarm is displayed in light blue color.

The map can be controlled using the graphical control elements located next to the map:

- **Default position**: Move and zoom the map so that all alarms are visible.
- **Previous position**: Go to the previous position.
- **A, B, C… buttons**: Focus the alarm. These buttons are colored according to the alarm state (green, red, blue). Next to these buttons is displayed the time and date of the last alarm activation.

Double-clicking on the map will display a pop up with geographical coordinates.

For the module to work properly, the Google Maps API key must be entered in *wt_config.xml* configuration file. Contact SAFIBRA for more information on how to acquire the API key and how to edit the configuration file.

Module parameters edit

Parameter	Units	Value
type	-	Google Maps
name	-	google_maps
inputs	-	[alarm_01, alarm_02]
enabled	-	enabled
refresh	msec	1000
gui_page	-	Maps
alarm_names	-	[sensor 01, sensor 02]
alarm_latitude	deg	[50.006573, 50.006187]
alarm_longtitude	deg	[14.651757, 14.652562]
focus_zoom_lvl	-	16
map_width	px	600
map_height	px	400

Figure 7.37 The “Google Maps” module settings

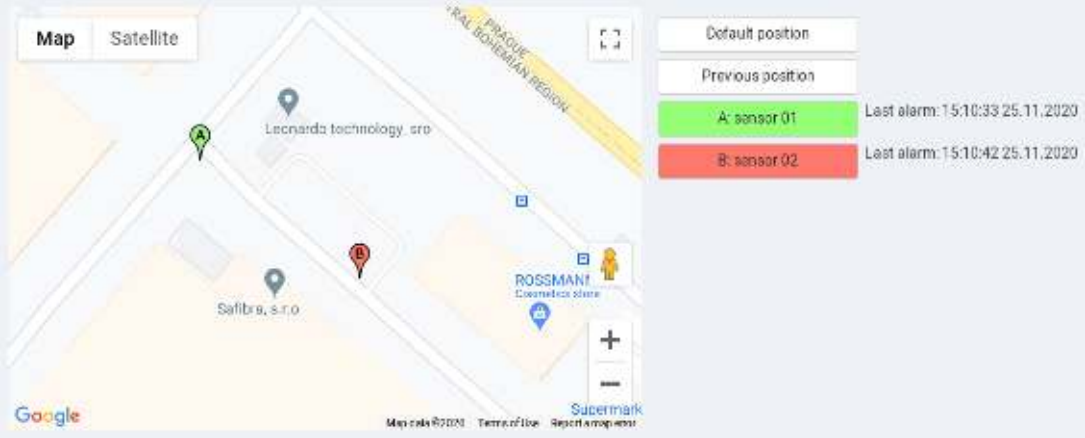


Figure 7.38 Map on the “Dashboard” page

7.5.7 Inclination Profile Module

Description

The module computes an inclination profile from a set of strain sensors.

- **License**: Expert.
- **Category**: Special.
- **Input**: Single vector.
- **Output**: Single vector.

Parameters

Name	Units	Type	Description
testing	-	option	(**disabled**) Log only basic messages. (**enabled**) Log verbose messages.
width	-	integer	Width of the input vector.
bendline_length	m	decimal	Length of the profile.
bendline_height	m	decimal	Height of the profile.
bendline_fbg_positions	m	decimal array	Positions of FBG sensors.
output_width	-	integer	Width of the output vector.
polynomial_order	-	integer	Order of polynomials used to fit a profile curve.
compensation_sensor	-	integer	Index of the compensating sensor.
bendline_position	-	option	(**normal**) The first sensor is on top. (**inverted**) The first sensor is on the bottom.
bendline_start	-	option	(**zero_left**) Zero position is on the left. (**zero_right**) Zero position is on the right.
y_output_units	-	option	(**meter**) Output is in meters. (**millimeter**) Output is in millimeters.
x_output_units	-	option	(**meter**) Output is in meters. (**millimeter**) Output is in millimeters.
async_temp_compensation	-	option	(**disabled**) Static asymmetrical temperature compensation is disabled. (**enabled**) Static asymmetrical temperature compensation is enabled.
async_temp_compensation_pair	-	integer	Index of a pair of sensors used for temperature compensation. Index starts from zero.
output_data_type	-	option	(**deflection**) Output is inclination. (**strain_diff**) Output is differences of pairs of sensors.
incremental_compensation	-	option	(**disabled**) Incremental asymmetrical temperature compensation is disabled.

			(**enabled**) Incremental asymmetrical temperature compensation is enabled.
temperature_compensation_factor	-	decimal	Incremental compensation temperature factor.
bend_compensation_factor	-	decimal	Incremental compensation bending factor.
temperature_dumping_factor			Incremental compensation dumping factor.
strain_diff_average_maximum_count			Maximum length of strain difference average for compensation.

7.5.8 InfluxDB Writer Module

Description

The module writes data to "InfluxDB".

- **License**: Expert.
- **Category**: Data sinks.
- **Input**: Single value, single vector.
- **Output**: None.

Parameters

Name	Units	Type	Description
testing	-	option	(**disabled**) Log only basic messages. (**enabled**) Log verbose messages.
width	-	integer	Width of the input vector.
connection_id	-	string	Database connection ID.
measurement	-	string	Measurement value (mandatory).
serial_number	-	string	Tag set value (optional).
sensor_id	-	string	Tag set value (optional).
units	-	string	Tag set value (optional).
note	-	string	Tag set value (optional).

Documentation

The *connection_id* parameter is an unique identifier of the database connection created and configured on the "Database manager" page.

The "SigProc" uses the /write HTTP endpoint with basic authentication to write to the pre-existing databases. More information on the endpoint is available in the official documentation of InfluxDB.

The data are transferred using the line protocol, which is a text-based format that provides the measurement, tag set, field set, and timestamp of a readout.
More information on the line protocol is available in the official documentation of InfluxDB.

The "SigProc" uses the line protocol with the following elements:

- For single values:
  - **measurement** (string): User-defined, mandatory parameter.
  - **tag set** (strings)**:**
    - serial_number**:** User-defined, optional parameter.
    - sensor_id**:** User-defined, optional parameter.
    - units**:** User-defined, optional parameter.
    - note**:** User-defined, optional parameter.
  - **field set** (float): Numeric value of the readout.
  - **timestamp** (nanoseconds): Timestamp of the readout, the time is in UTC.
- For vectors:
  - **measurement** (string): User-defined, mandatory parameter.
  - **tag set** (strings)**:**
    - serial_number**:** User-defined, optional parameter.
    - sensor_id**:** User-defined, optional parameter.
    - units**:** User-defined, optional parameter.
    - note**:** User-defined, optional parameter.
    - vector_id: Index in the vector. The index starts from one.
  - **field set** (float): Numeric value of the readout at the index *vector_id*.
  - **timestamp** (nanoseconds): Timestamp of the readout, the time is in UTC.

Database connection edit

Parameter	Units	Value
db_type		InfluxDB
connection_id		connection_internal
server_address		10.11.12.13
server_port		8086
user		admin
password		**********
db_name		internal
sending_period	sec	1
testing		disabled
backup_buffer_enabled		disabled
backup_buffer_path		/home/sigproc/gdb_data/
backup_buffer_sync_count		10
backup_buffer_load_count		10

Figure 7.39 The "InfluxDB" database connection settings

Module parameters edit

Parameter	Units	Value
type	-	InfluxDB writer
name	-	influx
inputs	-	[sensor_01]
enabled	-	enabled
testing	-	disabled
width	-	1
connection_id	-	connection_internal
measurement	-	test_measurement
serial_number	-	AB1234
sensor_id	-	sensor_01
units	-	mm
note	-	artificial_sensor

Figure 7.40 The “InfluxDB writer” module settings

Example: Target URL of the HTTP requests

http://10.11.12.13:8086/write?db=internal

Example: Readout converted into the line protocol

```
test_measurement, note=artificial_sensor, sensor_id=sensor_01, serial_number=AB1234,
units=mm value=4.5 1465839830100400200
```

7.5.9 Interferometer Deconvolution Module

Description

The module deconvolutes signals from the optical fiber interferometer.

- **License**: Expert.
- **Category**: Special.
- **Input**: Multiple values.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
channel_count	-	option	(**3**) Interferometer has 3 channels.
angle_coefficients	rad	decimal array	Optical coupler angles.
time_coefficient	sec	integer	Time interval at the beginning in which a minimum and a maximum of input signals are found.

7.5.10 Modbus Slave Module

Description

The module writes the input value into the “Modbus register”. The values in the “Modbus register” are accessible using the Modbus/TCP protocol.

- **License**: Expert.
- **Category**: Special.
- **Input**: Single value.
- **Output**: None.

Parameters

Name	Units	Type	Description
testing	-	option	(**disabled**) Log only basic messages. (**enabled**) Log verbose messages.
width	-	integer	Width of the input vector.
register_address	-	integer	Modbus register address that stores the value.

Documentation

The values in the “Modbus register” are formatted as 32bit single precision floating point numbers.

The value is stored at the *register_address* address increased by the *data_offset*. The *data_offset* parameter is defined in the *system.cfg* file.

For the module to work properly, the “Modbus Slave driver” must be configured in the “System configuration file”.

Modpoll

“Modpoll” is "Windows" and "Linux" command line utility for reading values from the “Modbus register”.

Module parameters edit

Parameter	Units	Value
type	-	Modbus Slave
name	-	modbus_01
inputs	-	[sensor_01]
enabled	-	enabled
testing	-	disabled
width	-	1
register_address	-	0

Module parameters edit

Parameter	Units	Value
type	-	Modbus Slave
name	-	modbus_02
inputs	-	[sensor_02]
enabled	-	enabled
testing	-	disabled
width	-	1
register_address	-	1

Figure 7.41 The “Modbus slave” module settings for two modules

The “Modbus slave driver” is configured as follows: port = 1502, data_offset = 1000, float_little_endian = 0.

The value from the modbus_01 module is stored on address 1000 and the value from the modbus_02 module is stored on address 1001.

Example: Reading the values using modpoll

```
[user@localhost ~]$ ./modpoll -m tcp -t 4:float -i -0 -r 1000 -c 2 -p 1502
10.11.12.1
modpoll 3.6 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright (c) 2002-2018 proconX Pty Ltd
Visit https://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: MODBUS/TCP
Slave configuration...: address = 1, start reference = 1000 (PDU), count = 2
Communication.........: 127.0.0.1, port 1502, t/o 1.00 s, poll rate 1000 ms
Data type.............: 32-bit float, output (holding) register table
Word swapping.........: Slave configured as big-endian word machine

-- Polling slave…
[1000]: 8.000000
[1002]: 1.366009
```

7.5.11 ObjectGuard Module

Description

The module processes signals from the “ObjectGuard” measurement system and computes a position deviation.

- **License**: Expert.
- **Category**: Special.
- **Input**: Multiple values.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
position_offset	mm	decimal	Offset added to the computed position.
calibration_spacing	mm	decimal	Spacing of calibration file readouts.
maximum_error	-	decimal	Maximum allowed difference from calibration values.
calibration_files	-	string array	Calibration files. One for each input.
multiplicative_coeficients	-	decimal array	Multiplicative correction coefficients.
additive_coeficients	-	decimal array	Additive correction coefficients.
no_signal_output_value	mm	decimal	Error value sent to the output when the input signals cannot be processed.

configuration_path	-	string	Path to the directory with configuration files.
signal_threshold	-	decimal	Minimal signal intensity considered useful for computation.
comparison_mode	-	option	Comparison mode used in computation is (**all valid**), (**maximum only**) or (**smart**).
auto_normalizing_mode	-	option	(**disabled**) Automatic normalization for equal intensity is disabled. (**enabled**) Automatic normalization for equal intensity is enabled.

7.5.12 Sequence Generator Module

Description

The module generates an ascending sequence of consecutive integers. The sequence starts at zero and ends at a defined value.

- **License**: Expert.
- **Category**: Data sources.
- **Input**: None.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
count	-		The final sequence value.
repeat	-		(**disabled**) Repeat the sequence. (**enabled**) Generate the sequence only once.
period	msec		Sampling period of the generated values.

7.5.13 Sine Wave Generator Module

Description

The module generates a sum of sine waves.

- **License**: Expert.
- **Category**: Data sources.
- **Input**: None.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
period	msec	integer	Period in which the values are generated.
sine_amplitudes	-	decimal array	Amplitudes of generated sine waves.
sine_periods	msec	integer array	Periods of generated sine waves.
sine_phases	deg	integer array	Phase shifts of generated sine waves.
offset	-	decimal	Added offset.

7.5.14 Time Maximum Module

Description

The module finds the maximum in the specified time period.

- **License**: Expert.
- **Category**: Advanced processing.
- **Input**: Single vector.
- **Output**: Single vector.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
time_slot_duration	msec	integer	Time interval for finding the maximum.
min_input_limit	-	decimal	Minimal valid input value. Lower values are ignored.

max_input_limit	-	decimal	Maximal valid input value. Lower values are ignored.
no_signal_output_value	-	decimal	Output value when the input value is ignored.

7.5.15 Time Minimum Module

Description

The module finds the minimum in the specified time period.

- **License**: Expert.
- **Category**: Advanced processing.
- **Input**: Single vector.
- **Output**: Single vector.

Parameters

Name	Units	Type	Description
width	-	integer	Width of the input vector.
time_slot_duration	msec	integer	Time interval for finding the minimum.
min_input_limit	-	decimal	Minimal valid input value. Lower values are ignored.
max_input_limit	-	decimal	Maximal valid input value. Lower values are ignored.
no_signal_output_value	-	decimal	Output value when the input value is ignored.

7.6 Modules under Exclusive Licence

7.6.1 HTTP Request Module

Description

The module sends the HTTP requests.

- **License**: Expert.
- **Category**: Third party integration.
- **Input**: Single value (index).
- **Output**: None.

Parameters

Name	Units	Type	Description
method	-	option	(**POST**) Use POST method. (**GET**) Use GET method.
url	-	string	URL of the target server.
strings	-	string array	Array of text strings.

Documentation

The module rounds the input value to the nearest integer and then uses the integer as an index to the strings array. The text string at that index is sent in the HTTP request. Note that the index starts from zero.

The request is sent only when the index changes from one value to another. For example from 0 to 1 or from 2 to 0.

For the module to work properly, the “HTTP request driver” must be configured in the “System configuration file”.

Module parameters edit

Parameter	Units	Value
type	-	HTTP request
name	-	http_get
inputs	-	[alarm_01]
enabled	-	enabled
method	-	GET
url	-	https://httpbin.org/get
strings	-	[x=zero, x=active]

Figure 7.42 The “HTTP request” module settings - method GET

Example: Messages printed by the “SigProc”, if the verbose logging is enabled (verbose_log_enabled = 1 in the “System configuration file”) - method GET

```
[2018-10-11 14:42:10.941143]D[HTTP_REQ] [INFOA] GET:
URL='https://httpbin.org/get?x=active'
[2018-10-11 14:42:11.591056]D[HTTP_REQ] [INFOA] reply:
{
  "args": {
    "x": "active"
  },
  "headers": {
    "Accept": "*/*",
    "Connection": "close",
    "Host": "httpbin.org"
  },
  "origin": "183.21.212.42",
  "url": "https://httpbin.org/get?x=active"
}
```

Module parameters edit

Parameter	Units	Value
type	-	HTTP request
name	-	http_post
inputs	-	[alarm_01]
enabled	-	enabled
method	-	POST
url	-	https://httpbin.org/post
strings	-	[x=zero, x=active]

Figure 7.43 The “HTTP request” module settings - method POST

Example: Messages printed by the "SigProc, if the verbose logging is enabled (verbose_log_enabled = 1 in the "System configuration file") - method POST

```
[2018-10-11 14:42:11.591463]D[HTTP_REQ] [INFOA] POST:
URL='https://httpbin.org/post', data='x=active'
[2018-10-11 14:42:12.126796]D[HTTP_REQ] [INFOA] reply:
{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "x": "active"
  },
  "headers": {
    "Accept": "*/*",
    "Connection": "close",
    "Content-Length": "8",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org"
  },
  "json": null,
  "origin": "183.21.212.42",
  "url": "https://httpbin.org/post"
}
```

7.6.2 JSON Message Event Module

Description

The module sends the JSON formatted event messages.

- **License**: Expert.
- **Category**: Third party integration.
- **Input**: Single value (boolean).
- **Output**: None.

Parameters

Name	Units	Type	Description
resend_interval	s	decimal	Interval in which the message is automatically resend if the input is nonzero (0 = disabled).
device_id	-	string	Device identifier.
msg_type	-	option	Type of the message is (**status**), (**notification**), (**alarm**), (**alert**) or (**fault**).
classification	-	string	Classification. Single quotes are internally converted to double quotes. The parameter is not sent if the text is

			empty.
description	-	string	Description. Single quotes are internally converted to double quotes. The parameter is not sent if the text is empty.
fault_code	-	integer	Fault code. The parameter is not sent if the text is empty.
fiber_distance	-	string	Fiber distance. The parameter is not sent if the text is empty.
feature_distance	-	string	Feature distance. The parameter is not sent if the text is empty.
is_primary	-	string	Is primary. The parameter is not sent if the text is empty.
weight	-	string	Weight. The parameter is not sent if the text is empty.
location_table_id	-	string	Location table ID. The parameter is not sent if the text is empty.

Documentation

The module sends a message when the input changes from zero to a nonzero number. If the parameter *resend_interval* is not zero then the message is resent periodically as long as the input is nonzero. The resending is stopped when the last input value is older than the *resend_interval*.

The following table defines the relation between message type (*msg_type*) and parameters that are included in the message.

	status	**notification**	**alarm**	**alert**	**fault**
version	✓	✓	✓	✓	✓
device_id	✓	✓	✓	✓	✓
message_id	✓	✓	✓	✓	✓
time	✓	✓	✓	✓	✓
type	✓	✓	✓	✓	✓
classification			✓	✓	
description	✓	✓			
fault_code	✓				✓

fiber_distance			✓		
feature_distance			✓		
is_primary			✓		
weight			✓		
location_table_id			✓		

For the module to work properly, the “JSON message driver” must be configured in the “System configuration file”.

Module parameters edit

Parameter	Units	Value
type	-	JSON message event
name	-	json_event
inputs	-	[alarm_intrusion_01]
enabled	-	enabled
resend_interval	s	0
device_id	-	south_gate
msg_type	-	notification
classification	-	
description	-	The operational status for the device at '/fibers...
fault_code	-	0
fiber_distance	-	
feature_distance	-	
is_primary	-	
weight	-	
location_table_id	-	

Figure 7.44 The “JSON message event” module settings - notification

Example: The JSON message - notification

```
{
"Version":2.1,
"Device_id":"south_gate",
"Message_id":4,
"time":"2018-04-20T09:18:38.256743Z ",
"Type":"notification",
"description":"The operational status for the device at
```

```
"/fibers/commander/device/10.10.10.3" changed to "offline""
}
```

Module parameters edit

Parameter	Units	Value
type	-	JSON message event
name	-	json_event_alarm
inputs	-	[alarm_intrusion_01]
enabled	-	enabled
resend_interval	s	0
device_id	-	south_gate
msg_type	-	alarm
classification	-	WARNING
description	-	
fault_code	-	0
fiber_distance	-	100
feature_distance	-	150
is_primary	-	false
weight	-	2
location_table_id	-	64868

Figure 7.45 The “JSON message event” module settings - alarm

Example: The JSON message - alarm

```
{
"version":2.1,
"device_id":"south_gate",
"message_id":17,
"time":"2018-04-23T16:19:31.256743Z ",
"type":"alarm",
"classification":"WARNING",
"fiber_distance":100,
"feature_distance":150,
"is_primary":"false",
"weight":2,
"location_table_id":64868
}
```

7.6.3 JSON Message Service Module

Description

The module sends the JSON formatted service messages.

- **License**: Expert.
- **Category**: Third party integration.
- **Input**: None.
- **Output**: None.

Parameters

Name	Units	Type	Description
device_id	-	string	Device identifier.
device_path	-	string	IP address of the Command Server.
device_ip	-	string	IP address of the Command Server.
heartbeat_period	sec	integer	Time period of heartbeat messages (0 = disabled).

Documentation

The module sends 4 messages when it starts and 3 messages when it ends. The module also sends heartbeat messages periodically.

For the module to work properly, the "JSON message driver" must be configured in the "System configuration file".

Module parameters edit

Parameter	Units	Value
type	-	JSON message service
name	-	json_service
enabled	-	enabled
device_id	-	south_gate
device_path	-	/fibers/commander/device/10.0.0.3
device_ip	-	10.0.0.10
heartbeat_period	s	60

Figure 7.46 The "JSON message service" module settings

Example: The JSON messages at the beginning


```
{
"version":2.1,
"device_id":"south_gate",
"message_id":0,
"time":"2018-04-23T16:07:26.256743Z",
"type":"notification",
"description":"connection established (10.0.0.10)"
}
{
"version":2.1,
"device_id":"south_gate",
"message_id":1,
"time":"2018-04-23T16:07:26.256743Z ",
"type":"status",
"fault_code":-1,
"description":"Initializing..."
}
{
"version":2.1,
"device_id":"south_gate",
"message_id":2,
"time":"2018-04-23T16:07:26.256743Z ",
"type":"notification",
"description":"started"
}
{
"version":2.1,
"device_id":"south_gate",
"message_id":3,
"time":"2018-04-23T16:07:26.256743Z ",
"type":"notification",
"description":"The connection status for the device at
"/fibers/commander/device/10.0.0.3" changed to "connected""
}
```


Example: The JSON heartbeat message


```
{
"version":2.1,
"device_id":"south_gate",
"message_id":5,
"time":"2018-04-23T16:50:50.256743Z ",
"type":"status",
"fault_code":0,
"description":"OK"
}
```


Example: The JSON messages at the end


```
{
"version":2.1,
"device_id":"south_gate",
"message_id":32,
"time":"2018-04-23T16:34:08.256743Z ",
"type":"notification",
"description":"The connection status for the device at
"/fibers/commander/device/10.0.0.3" changed to "disconnected""
```

```
}
{
"version":2.1,
"device_id":"south_gate",
"message_id":33,
"time":"2018-04-23T16:34:08.256743Z ",
"type":"status",
"fault_code":-2,
"description":"Stopping server..."
}
{
"version":2.1,
"device_id":"south_gate",
"message_id":34,
"time":"2018-04-23T16:34:08.256743Z ",
"type":"notification",
"description":"stopping"
}
```


7.6.4 Milestone String Event Module

Description

The module sends messages to the Milestone system.

- **License**: Expert.
- **Category**: Third party integration.
- **Input**: Single value (boolean).
- **Output**: None.

Parameters

Name	Units	Type	Description
message_string	-	string	Message text.

Documentation

The message with an unformatted text defined in message_string is sent when the input value changes from zero to a nonzero value. The message is sent over the TCP/IP.

For the module to work properly, the “Milestone string driver” must be configured in the “System configuration file”.

7.6.5 Wavelet Fingerprints Module

Description

The module uses wavelets and pattern matching to activate an alarm.

- **License**: Exclusive.
- **Category**: Special.
- **Input**: Single value.
- **Output**: Single value.

Parameters

Name	Units	Type	Description
print_help	-	integer	Print help and exit (not implemented).
sample_each	-	integer	Process every n-th value only.
initial_avg_diff	-	decimal	Initial average noise related difference.
n_amend_avgdiff	-	integer	Number of diff values used to compute average diff.
number_of_points_to_alarm	-	integer	Number of consecutive events to activate an alarm.
multiplicator_to_detect	-	decimal	Multiplicator for average noise level to get an alarm threshold.
wait_state_usec	usec	integer	Waitstate duration after an alarm is activated. No new alarm is activated in this period.
distance_calculation_type	-	integer	How distance between vectors is calculated.
fingerprint_length	-	integer	Length of fingerprint vector - number of points.
fingerprint_match_positives_from	-	integer	First fingerprint vector index to start calculating the distance for positive patterns. Starts from 0.
fingerprint_match_negatives_from	-	integer	First fingerprint vector index to start calculating the distance for negative patterns. Starts from 0.
fingerprint_match_positives_to	-	integer	Last fingerprint vector index to start calculating the distance for positive patterns. Last value is (fingerprint_length - 1).

fingerprint_match_negatives_to	-	integer	Last fingerprint vector index to start calculating the distance for negative patterns. Last value is (fingerprint_length - 1).
wavelet_function	-	integer	Wavelet function type.
generate_fingerprints	-	integer	How to generate fingerprints (0 = disabled).
matching_distance_positives_max	-	decimal	Individual match threshold for positives.
matching_distance_negatives_max	-	decimal	Individual match threshold for negatives.
matches_evaluation_logic	-	integer	Logic for generating the final match.
use_diff_value	-	integer	Whether difference values should be used instead of direct measurement.
fingerprints_directory	-	integer	Fingerprints bank directory.
debug_level	-	integer	Level of debug info.
matchdistance_to_output	-	integer	If match_distance_should be outputted instead of standard output value.
genpatterns_hour_limit	-	integer	Max number of patterns to be generated within one hour (0 = unlimited).

Documentation

Documentation is available upon request.

8 SUPPLEMENTARY INFORMATION

8.1 Keyboard Shortcuts

List of all keyboard shortcuts that can be used in the "Configurations" section.

Key	Description
Up/Down	Move up/down in the table
D/Delete	Delete the configuration
C	Clone the configuration
E/Enter	Open the configuration in the editor
R	Run the configuration
S	Stop the configuration
A	Create a new configuration
W	Go to "GUI page" (dashboard)

List of all keyboard shortcuts that can be used in the "Configuration tree" section.

Key	Description
Arrow Up/Down	Move up/down within the tree
Shift + arrow Up/Down	Move module up/down the tree
Left	Go to a higher level module group/collapse module group
Right	Expand module group
Enter	Go to module editor
D/Delete	Delete module/module group
S	Save the configuration
R	Save and run the configuration
A	Add module
Shift + A	Add module group
C	Copy module/module group

Shift + C	Copy module/module group to clipboard
Shift + V	Paste module/module group from clipboard
F	Enable/disable filter
X	Expand/collapse all module groups
W	Go to “GUI page” (dashboard)
Q	Go to list of configurations
T	Test module connection

List of all keyboard shortcuts that can be used in the “Module parameters edit” section.

Key	Description
Up/Down	Move up/down between the lines
Enter	Confirm changes and close the editor/open the editor
Shift + Enter	Confirm changes and go to configuration tree
Esc	Confirm changes and close the editor/go to configuration tree
Tab	Go from "Value" to "Tag" column
Shift + Tab	Go from "Tag" to "Value" column

List of all keyboard shortcuts that can be used in all dialog windows of the “Open in editor” section.

Key	Description
Up/Down	Move up/down the dialog window options
Tab	Go to the next option/dialog window
Shift + Tab	Go to the previous option/dialog window
Enter	Confirm filter/confirm the whole dialog window
Esc	Close the dialog window

8.2 Remote Access

All data files and configurations are accessible through “Secure Shell” (SSH), a cryptographic network protocol for operating network services securely over an unsecured network.

It is possible to use various SSH clients but for easier support we recommend to use “MobaXterm”.

To connect to the ”SigProc", click on the “Sessions” button, then select “SSH”. Insert the IP address of the ”SigProc" into the “Remote host” field. Tick “Specify username”, fill in “**sigproc**” and click “OK”. New session will open asking for your SSH password (password can be found in the “Specification sheet” delivered together with your ”SigProc").

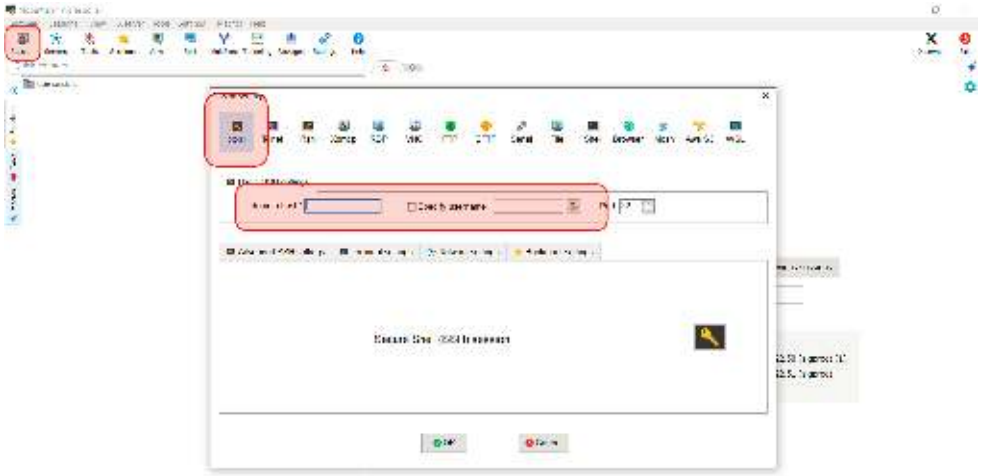


Figure 8.1 MobaXterm SSH connection

After logging in, the Graphical SFTP browser will be displayed on the left side. Data from the measurement are available in the “Data” folder (the usual path is */home/sigproc/data/*). Configurations are available in the folder named *config*. The usual path is */home/sigproc/config/configs/*.

To download this data, just drag the folder and drop it to the file browser on your computer.

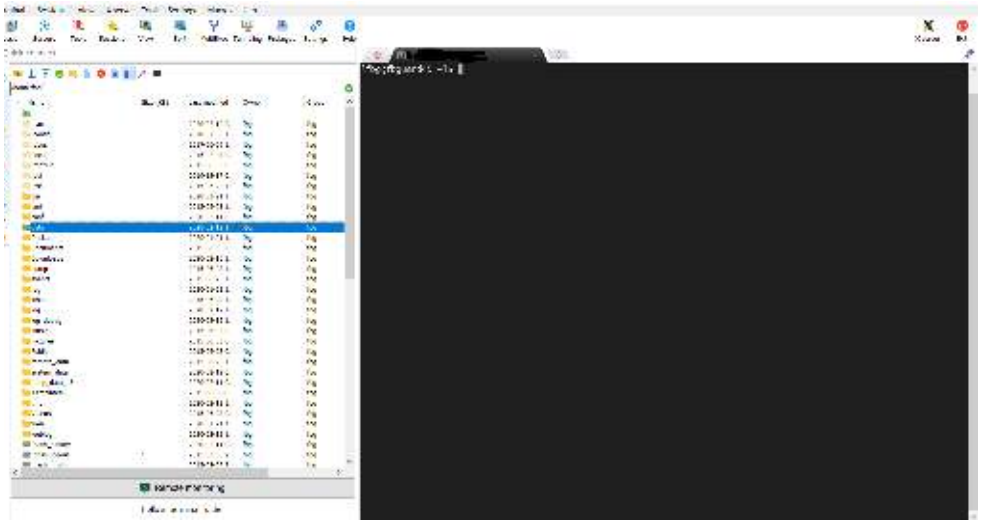


Figure 8.2 SSH - the Graphical SFTP browser

8.3 System Configuration File

The “System Configuration File” is used to set the ”SigProc" parameters. This file is present at the default factory settings and should be edited only in exceptional cases.

The “System Configuration File” is located at */home/sigproc/config/system.cfg*. When editing the file, it is necessary to keep the text structure (parentheses, equals, semicolons, quotation marks, etc.). After editing the file, it is necessary to restart the ”SigProc" in order to load the new settings. This should be carried out by restarting the entire device or by using the procctl stop and procctl start commands.

It is strongly advised to make a backup copy before making any modifications.

8.3.1 Automatic File Eraser

```
file_eraser = {
       enabled = 1;
       erase_level = 90;
       warning_level = 80;
       megabytes_to_delete = 1000;
       check_period = 50;
       disk_quota = 850;
};
```

- **enabled**: The eraser is enabled (1) or disabled (0).
- **warning_level**: Threshold level for issuing a warning (percentage).
- **erase_level**: Threshold level for erasing old data and log files (percentage).
- **megabytes_to_delete**: Amount of deleted files when the disk usage exceeds the threshold.
- **check_period**: Period for checking the disk usage (seconds).
- **disk_quota**: Available space on the disk (GB). **Do not edit!**

8.3.2 HTTP Request Driver

```
http_request_sender = {
       enabled = 0;
       vervose_log_enabled = 0;
};
```

- **enabled**: Driver for sending HTTP requests wil (1) or won’t (0) be created.
- **verbose_log_enabled**: Print verbose logs. Enabled (1) or disabled (0).

8.3.3 JSON Message Driver

```
json_message_sender = {
       enabled = 0;
       server_ip_address = "10.23.23.117";
       server_port = 11111;
```

```
        protocol = "UDP";
        log_enabled = 1;
}
```

- **enabled**: Driver for sending JSON messages will (1) or won't (0) be created.
- **server_ip_address**: IP address of the target server.
- **server_port**: Port of the target server.
- **protocol**: Type of the communication protocol: "TCP" or "UDP".
- **log_enabled**: Sent messages are also saved locally on the disk. Enabled (1) or disabled (0). Text files with messages are located in the folder */home/sigproc/data/json_log*.

8.3.4 Log Messages Uploader [Experimental]

```
db_message_log = {
        enabled = 0;
        connection_id = "Influx_logs";
        measurement = "logs";
        serial_number = "ProcessGuard_00";
        min_severity = 0;
};
```

- **enabled**: Uploading is enabled (1) or disabled (0).
- **connection_id**: Connection ID of the target database. The connection is defined in the "Database manager".
- **measurement**: Measurement name (if the connection is to the "InfluxDB"). Table name (if the connection is to the "PostgreSQL").
- **serial_number**: Serial number or name of the "SigProc". User-defined text string.
- **min_severity**: Minimal message severity that will be uploaded into the database. Info (0), Warning (3), Error (4).

8.3.5 Milestone String Driver

```
milestone_string_sender = {
        enabled = 0;
        server_ip_address = "10.23.23.129";
        server_port = 11111;
}
```

- **enable**: Driver for sending "Milestone strings" will (1) or won't (0) be created.
- **server_ip_address**: IP address of the target server.
- **server_port**: Port of the target server.

8.3.6 Modbus Slave Driver

```
modbus_slave = {
	ip_address = "127.0.0.1";
	port = 502;
	float_little_endian = 0;
	data_offset = 1000;
	testing = 0;
};
```

- **ip_address**: Set to "127.0.0.1". **Do not edit!**
- **port**: TCP port on which the driver listens for incoming connections.
- **float_little_endian**: Float value endianness. Little endian (1) or big endian (0).
- **data_offset**: Offset of the data holding register.
- **testing**: Verbose logging is enabled (1) or disabled (0).

8.3.7 RelayUnit Driver

```
external_relay = {
	dev_present = 0;
	dev_file = "/dev/ttyS0";
	hard_restart = 1;
};
```

- **dev_present**: The “RelayUnit” is connected (1) or disconnected (0).
- **dev_file**: Path to the serial port file descriptor. **Do not edit!**
- **hard_restart**: The RelayUnit is powered off during the restart. Enabled (1) or disabled (0).

8.3.8 Zabbix Driver

```
zabbix = {
	remote_log_global_enable = 0;
	remote_log_internal_values_log_interval = 0;
	remote_log_max_processes = 1;
	remote_log_process_timeout = 120;
	remote_log_interval = 5000;
};
```

- **remote_log_global_enable**: Driver for sending data into the "Zabbix" tool will (1) or won’t (0) be created.
- **remote_log_internal_values_log_interval**: Not available. Set to zero.
- **remote_log_max_processes**: Maximum number of parallel data transfer.
- **remote_log_process_timeout**: Timeout for finishing one data transfer (seconds).
- **remote_log_interval**: Period for executing the data transfer (milliseconds).

8.3.9 Other Parameters

```
threads = 2;
graph_background_color = [220, 220, 220];
```

- **threads**: Number of threads. **Do not edit!**
- **graph_background_color**: Background color of graphs ([red, green, blue]).

```
application_name = {
	login_prompt_string = "Data Processing Software (SigProc) on
ProcessGuard_00"
	browser_application_title = "ProcessGuard_00 SigProc"
	page_application_title = "ProcessGuard_00"
};
```

- **login_prompt_string**: Text displayed on the login screen.
- **browser_application_title**: Page title displayed in the browser tab.
- **page_application_title**: Text displayed in the left menu.

```
network_manager = {
	enabled = 0;
	config_file = "/etc/sysconfig/network-scripts/ifcfg-enp2s0";
};
```

- **enabled**: Driver for changing the network settings is enabled (1) or disabled (0). **Do not edit!**
- **config_file**: Path to the network configuration file. **Do not edit!**

8.4 Data File Example

Ascii format:

- CSV format with semicolon delimiter.
- The first two rows contain additional information.
- The file has two columns - timestamps and values.
- The approximate size is 37 bytes per readout.

Example: File structure in ascii format

```
time; value;
[dd-mm-yyyy hh:mm:ss.usec];[mm];
14-10-2020 07:30:00.119195 ; 1.04257;
14-10-2020 07:30:01.119195 ; 1.04257;
14-10-2020 07:30:02.119195 ; 0.936086;
```

Binary format:

- The file is not human readable.
- Faster writing and reading.
- Smaller size.
- The size is exactly 12 bytes per readout.

Example: File structure in binary format

	Readout #1			Readout #2			...
Address	0	4	8	12	16	20	...
Size	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	...
Content	time sec	time usec	value	time sec	time usec	value	...

The *time sec* is standard Unix timestamp (seconds from epoch beginning).

The *time usec* is a microsecond part of the timestamp.

The *value* is a single precision floating point number

9 SUPPORT

Once you experience any unexpected behavior of “SigProc”, feel free to contact SAFIBRA support for assistance at:

SAFIBRA, s.r.o.
U Sanitasu 1621
251 01 Říčany
Czech Republic

+420 323 601 615
support@safibra.cz