Inveo Sp. z o.o.

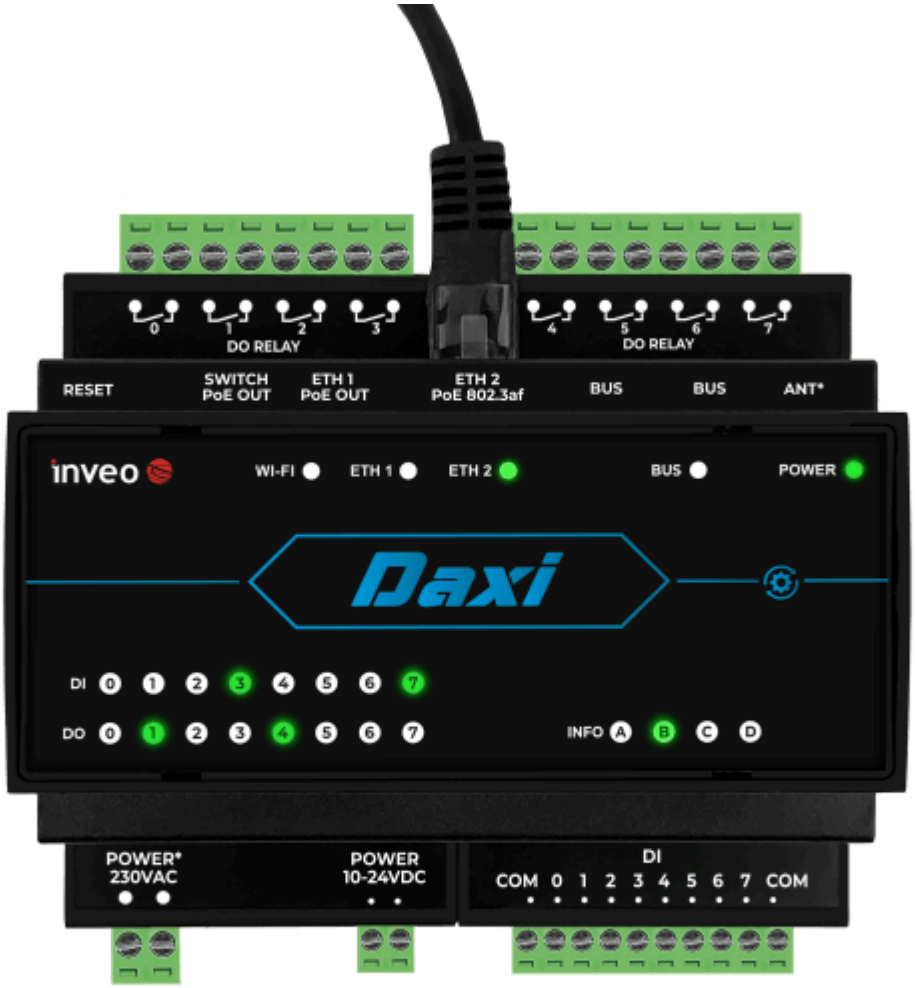
Daxi

Instruction Manual

Overview

Warning

This manual applies only to the firmware version v0.37.0 onwards. Inveo does not guarantee that the information contained in this document applies to previous firmware revisions.



Purpose of the device

The Daxi device provides an advanced solution for controlling electrical circuits. It is equipped with independent outputs that allow the control of a variety of system components, from simple circuits to more complex modules. With its ability to handle multiple sensors (e.g.: temperature, humidity, etc.), Daxi is not only a tool for control, but also for monitoring various parameters in the system.

One of Daxi's key strengths is its ability to automatically respond to signals and readings from connected sensors and digital inputs. The user can program the device to perform specific actions (actions), such as controlling a specific output or sending notifications in response to a specific signal, etc.

Daxi can be used as a tool for scheduling timed tasks, working on the basis of the Cron syntax characteristic of Linux systems. In this syntax, each line represents a single time criterion which, when satisfied, initiates the designated command.

Daxi has built-in logic capabilities - Logic, Lua. With logic functions and the Lua language, the device is capable of performing complex logic operations without the need for external controllers. This makes Daxi not only an effective, but also a flexible tool for any automation system.

Finally, Daxi supports a range of popular communication protocols such as HTTP, HTTPS, MQTT, SNMP v2/v3, SMSApi, Modbus TCP and Modbus RTU, making it compatible with many other devices and systems. The ability to integrate with external Modbus sensors further extends its range of applications.

Table of contents

- Overview
- Purpose of the device
- Device construction
  - Technical Characteristics
  - Connections schematics
  - LED indicators
- Device description
  - Sensors
  - Actions
  - Inputs
  - Outputs
  - Logic module
  - Integration with smart building systems
  - Integration with other devices
  - Cloud integration
  - Modbus client and server
  - Integration with Inveo devices
  - Control of network device state
- Network configuration
  - Network settings configuration
- Device configuration
  - Module's www interface
  - Device status preview
    - Sensors window
    - Outputs window
    - Inputs window
    - Device status preview via a JSON resource
  - Inputs/outputs configuration
    - Outputs
      - Last output state or counter retention
    - Shutters
    - Inputs
    - LED
  - Sensors configuration
    - All

        - Alarms configuration
    - History
    - Chart
    - Sensor handling
        - Sensor readings logging and graph preview
        - Download of stored sensor readings
- Notifications configuration
    - Sensors
    - Inputs
    - Outputs
    - Configuration
- Work matrix
    - KNX outputs
    - KNX/TCP/UDP Inputs
- Binding
    - iCluster
        - Lantick modules configuration
    - Outputs
    - Inputs
    - mbAgent
- Schedule
    - Weekly
    - Cron action
    - Cron standard
    - Configuration
    - Twilight
- Remote hosts ping (Watchdog)
- Logic functions
    - Conditions
    - Variables
    - Lua Script
        - Available system functions
        - Lua libraries

- Defining actions
    - All

- Protocol configuration
  - KNX
  - UDP
  - TCP
  - HTTP
  - MQTT
  - IO
  - Internal log
  - E-mail
  - SMS
  - SNMP Trap
- Inputs
- KNX
- System
- Periodic

- Communication protocols
  - Web
  - HTTPc
  - MQTT
  - E-mail
  - SMS API
  - Modbus
    - Coils addressing
    - Discrete Inputs addressing
    - Holding Registers addressing
    - Input Registers addressing
  - SNMP
    - SNMP v2c
    - SNMP v3
  - Syslog
  - TCP/UDP
  - iCluster
  - IO commands
  - Built-in variables
  - REST API

- Factory settings, backup

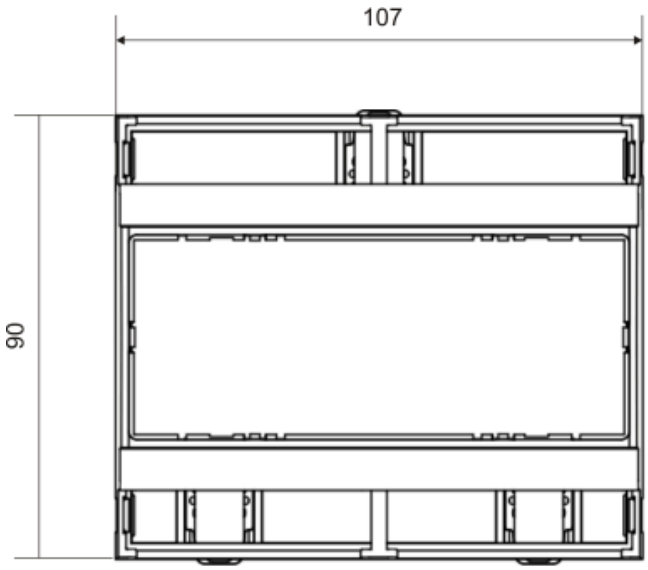
- Emergency firmware restore / restoring factory settings
- Warranty and manufacturers liability
  - Storage, operation and transport conditions
  - Disposal and decommissioning

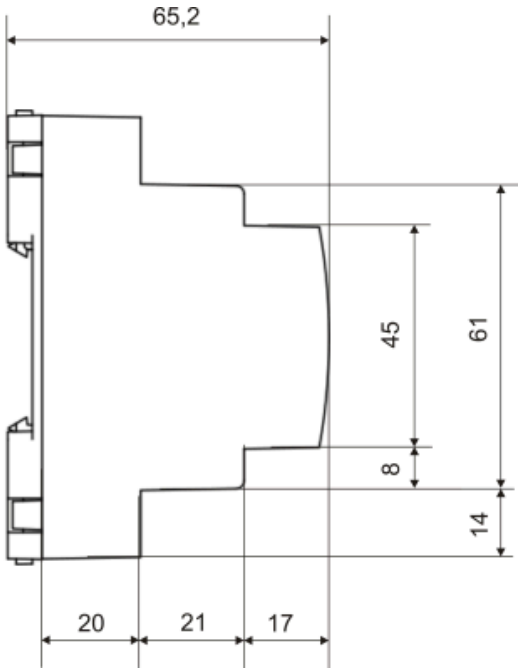
Device construction

Technical Characteristics

Parameter	Description
Power supply	**10-24V DC** with a 3,5mm pluggable terminal block **PoE:** 33-57V PoE IEEE 802.3af **85-264V AC** *(Daxi 230V)*
Power consumption	1,2 - 3,5W depending on the amount of activated outputs
Bus	**1-wire** – flat RJ12 cable Supports up to 16 sensors Dedicated sensors: temperature, humidity, water pressure, air pressure, analogue voltage, analogue current, gases: CO, CO_2, NO_x
Inputs	8 digital inputs **Input type:** digital, normally open, pre-polarized, galvanically isolated
Outputs	**Output type:** potential-free NO relay output **Maximum operating voltage:** 230V AC **Maximum relay load:** 2A NO output (normally open) **Activation time:** 1ms **Deactivation time:** 5ms **Operating mode:** bistable, astable, monostable, timed
Communication	2x Ethernet 10/100Mbps Full Duplex, RJ45 port, Wi-Fi 2,4GHz *(Daxi Wi-Fi)*
Enclosure ingress protection	IP30
Weight	ca. 320g (11,3oz) (Daxi Wi-Fi 230V)

Dimensions





Information

All dimensions are in millimeters.

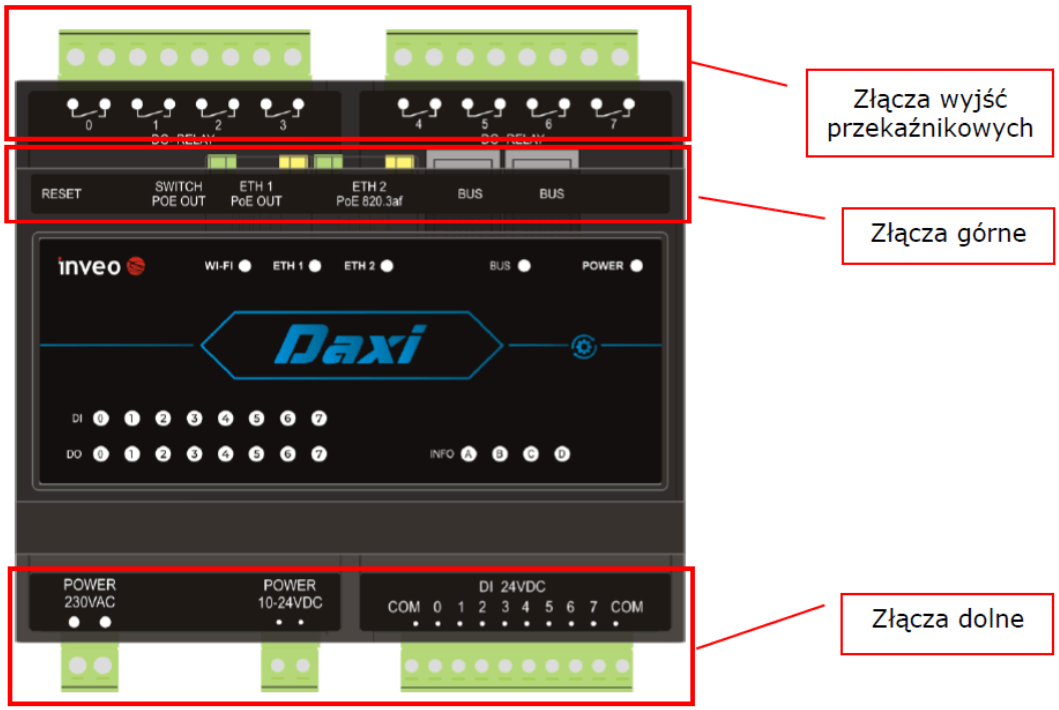
General view



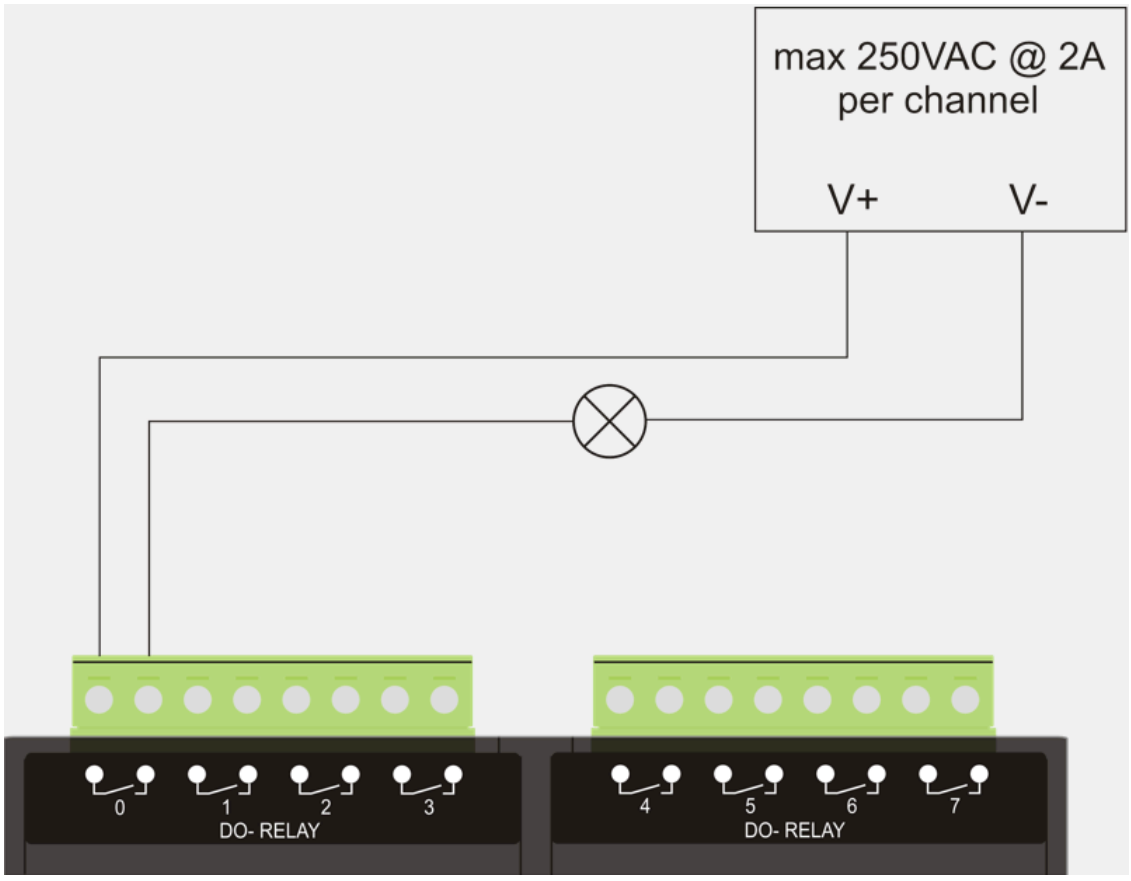


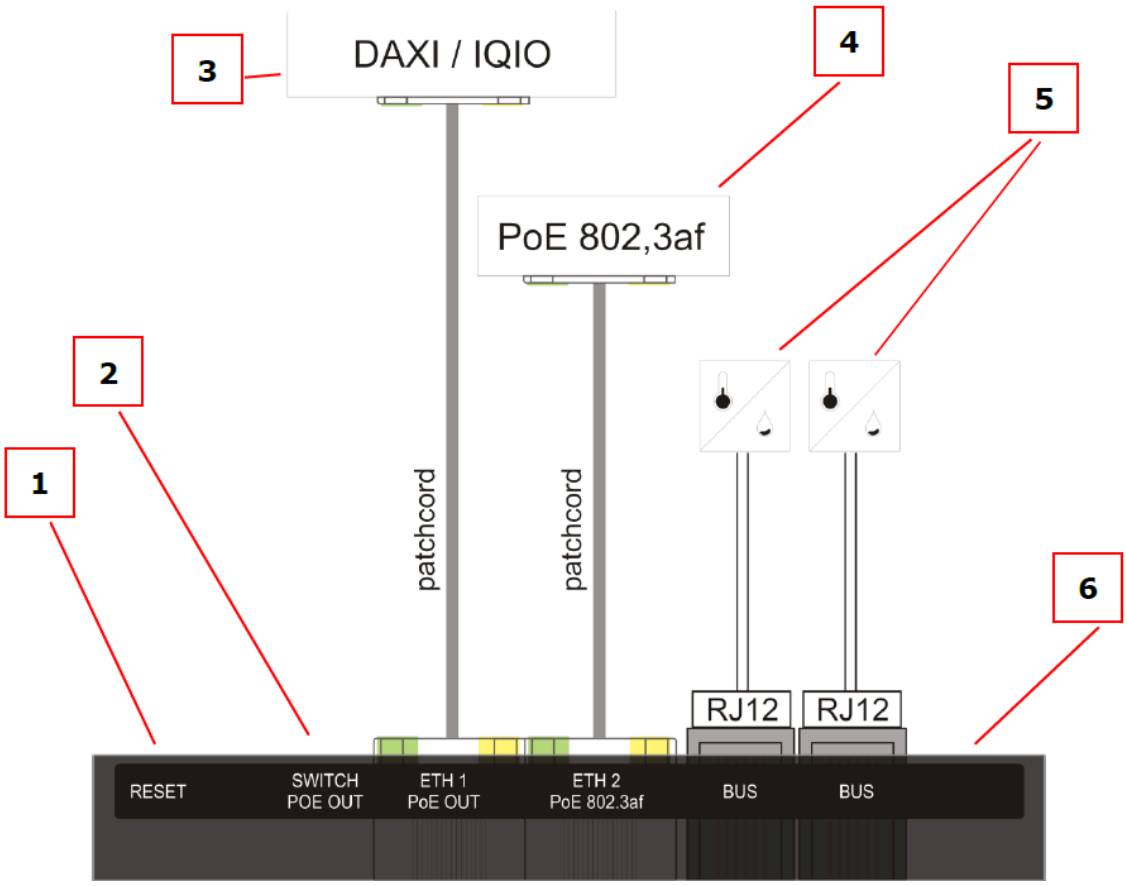


Connections schematics



Relay output connectors





Top connectors - level 2

1. **RESET** button,
2. PoE power supply splitter switch:
   - **up position** – No power to **ETH1 PoE Out**,
   - **down position** – Power is supplied to **ETH1 PoE Out**.
3. **ETH1 – PoE OUT*** switch port,
4. **ETH2 PoE 802.3 af*** switch port,
5. Supported sensors – Connected via a RJ12 cable, up to 16 sensors (daisy-chained together or with a BUS EXPANDER),
6. WiFi antenna (option).

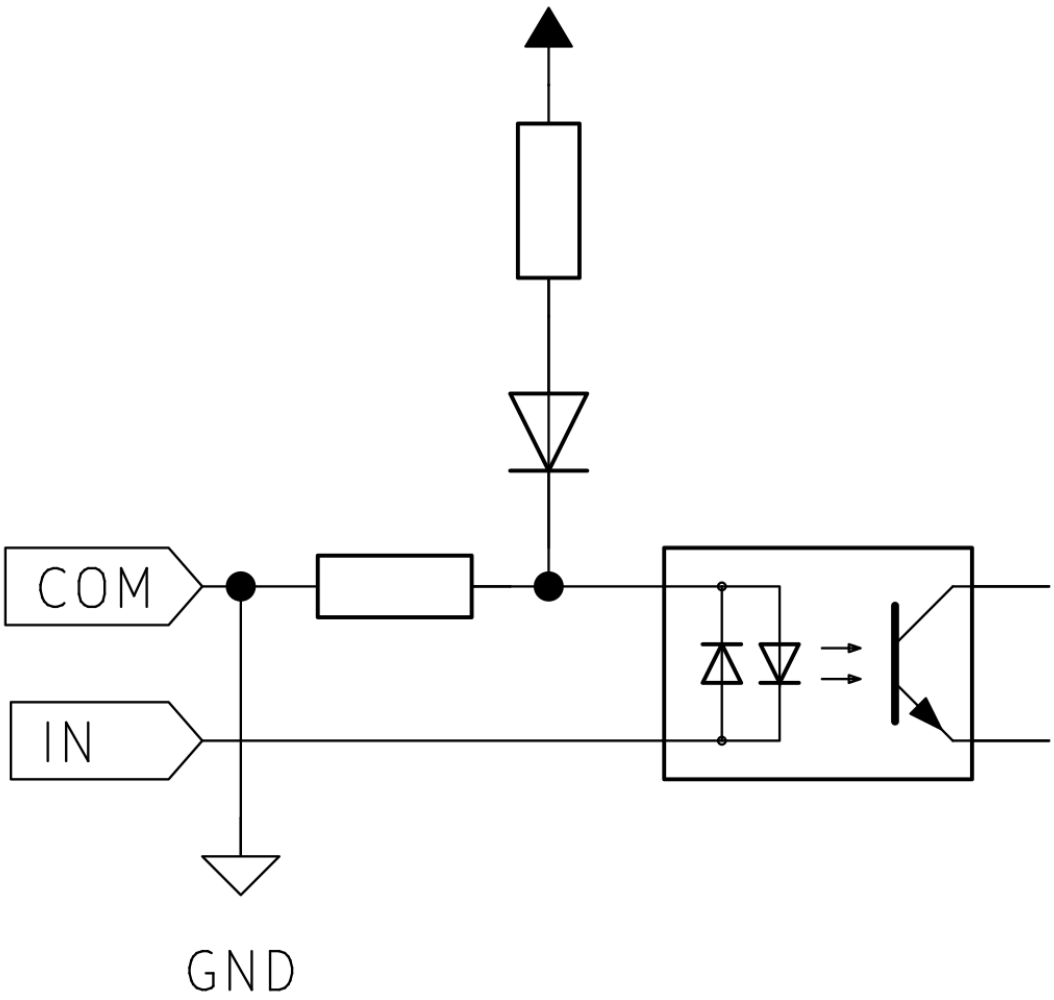
* **ETH1** and **ETH2** are the ports of a switch. **ETH2** allows to power Daxi via PoE 802.3af. If using **ETH2** to supply Daxi with power, **ETH1** can be used to power another device. Up to two devices can be powered this way.

Bottom connectors – Designed to connect supported inputs and power (10-24V DC and optionally 230V AC):

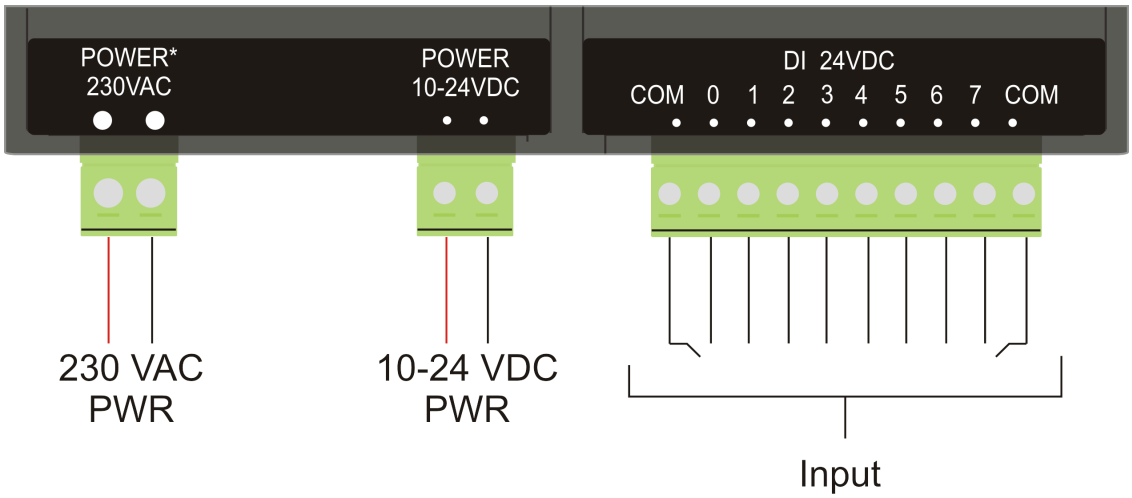
Information

The power connectors are indifferent to polarity – swapping + and - won't damage the module.

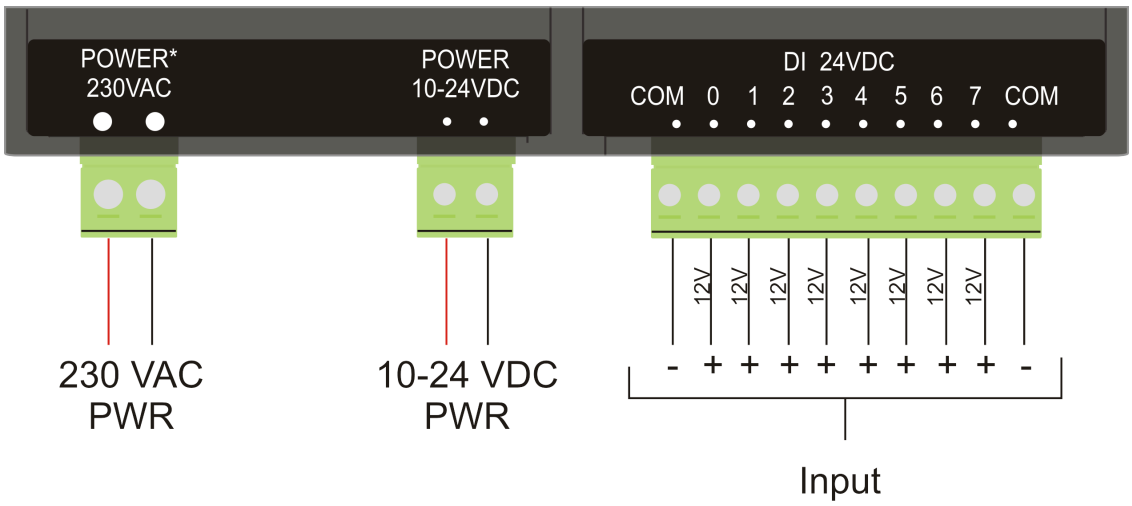
Outputs can be connected both in digital and voltage mode



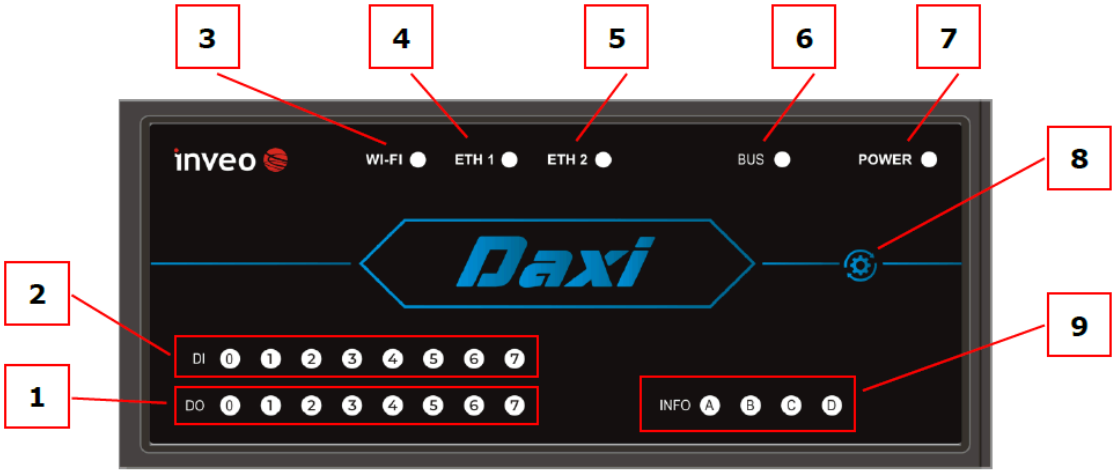
Connections diagram when using digital inputs (pre-polarized):



Connections diagram when using opto-isolated inputs (10-24V DC voltage input):


POWER*
230VAC
POWER
10-24VDC
DI 24VDC
COM 0 1 2 3 4 5 6 7 COM
12V
12V
12V
12V
12V
12V
12V
12V
- + + + + + + + + -
230 VAC
PWR
10-24 VDC
PWR
Input

LED indicators

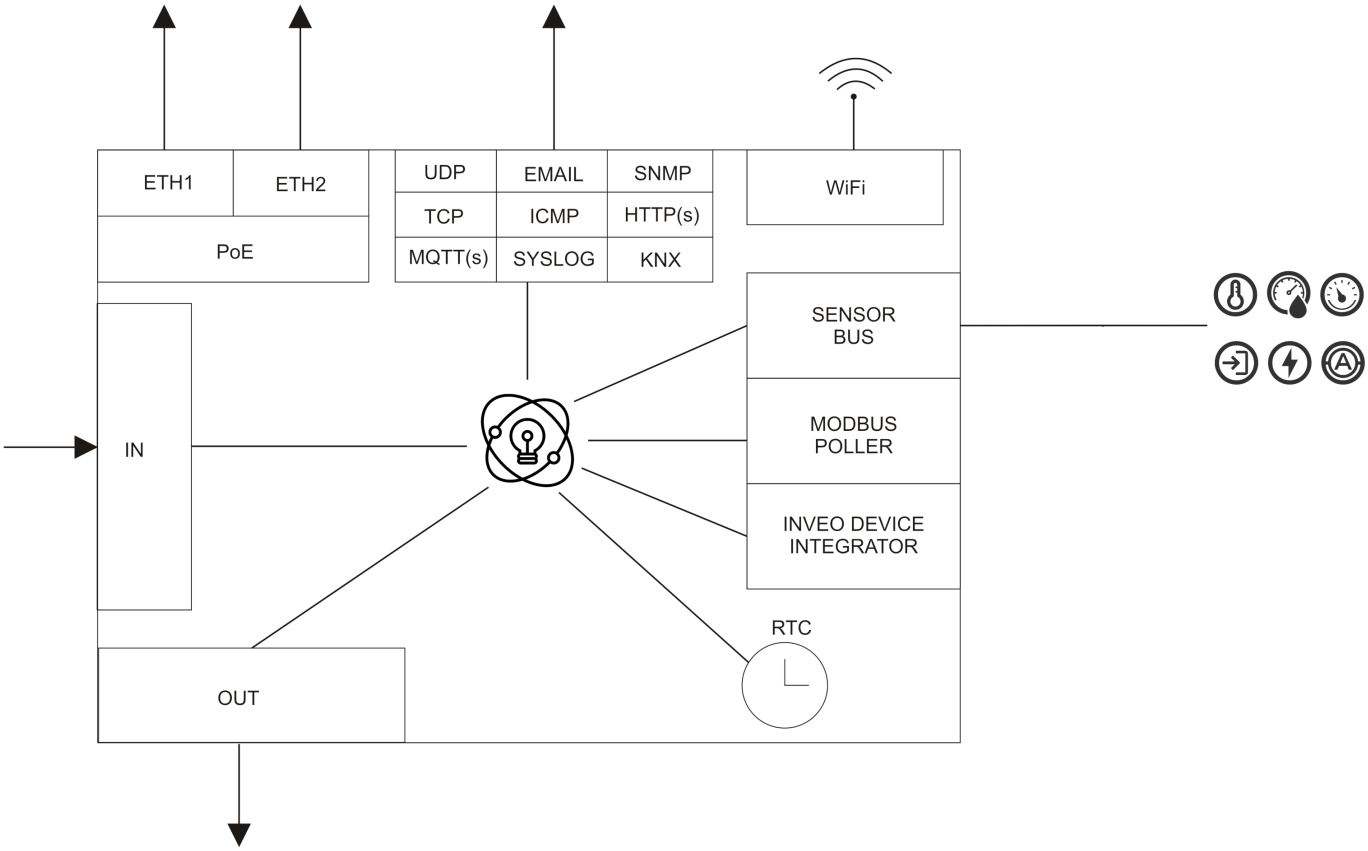


1. **Outputs state**
2. **Inputs state**
3. **Active Wi-Fi connection:**
   - Off: WiFi inactive,
   - On: WiFi active,
   - Blinks: WiFi connection error.
4. **Link Ethernet 1**
5. **Link Ethernet 2**
6. **Sensor bus status:**
   - Off: Sensor bus inactive,
   - On: Sensor bus active,
   - Blinks: Sensor bus error.
7. **Device power**
8. **Function button**
9. **Additional LED indicators – their assignment is defined by the user**

Device description

Daxi combines a programmable I/O controller and a sensor controller. It's main purpose is to collect data from sensors and inputs, process them and perform appropriate functions (actions). The device is configured via a web browser.

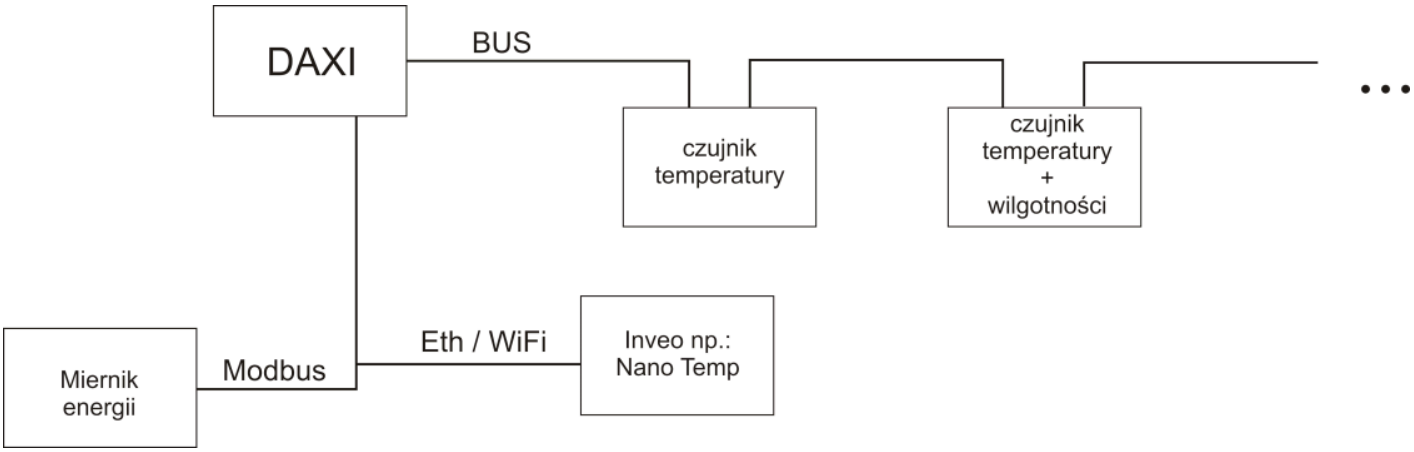
Block diagram:

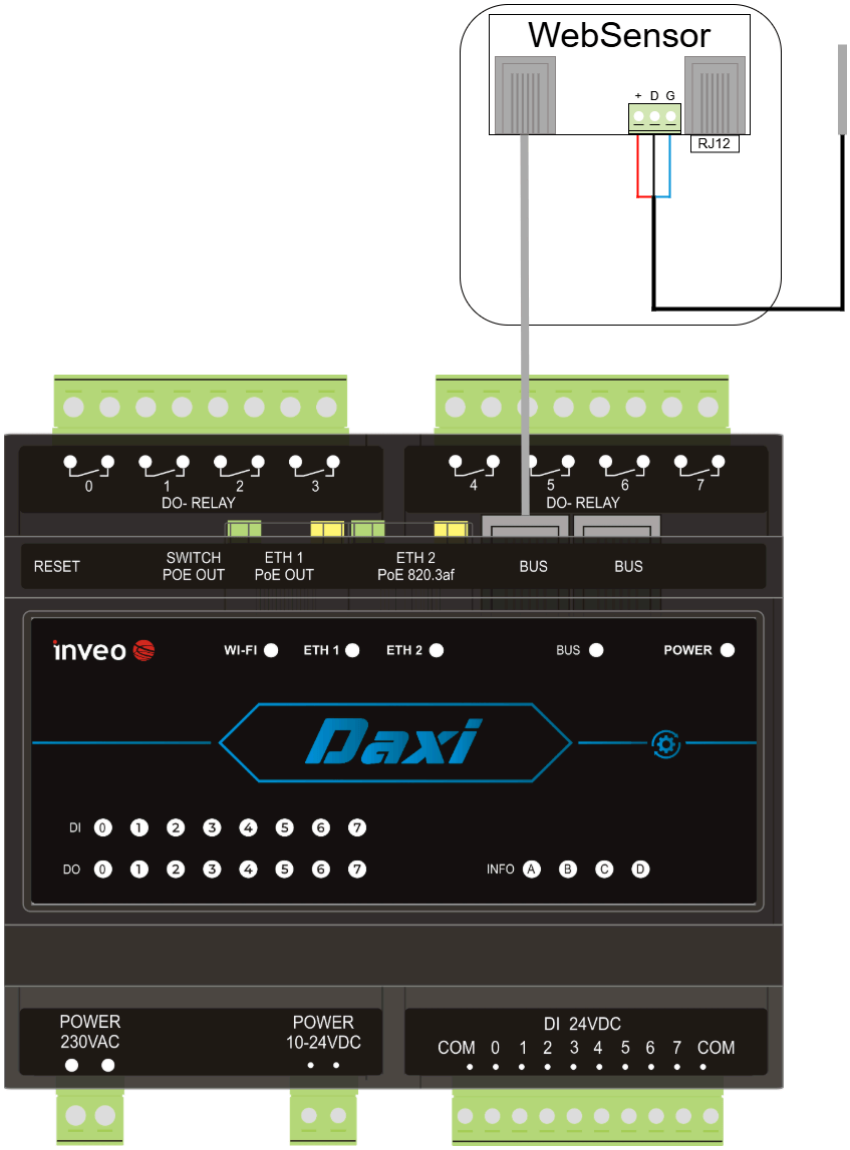


The module is equipped with a Real Time Clock with battery backup, allowing it to perform time-related functions, enabling the use of periodic actions performed at set times. Additionally, it's equipped with an astronomic clock, allowing for controlling outputs depending on the sunrise and sunset hours. The clock can be updated with NTP server time via Internet.

Sensors

Environmental factor sensors are connected to the device using a 3-wire bus in the form of a flat RJ12 cable, with support of up to 16 sensors. To make installation easier, the sensors are equipped with an additional connector used to connect another sensor.





Daxi supports the following sensors:

- Temperature,
- Humidity,
- Flooding detector,
- Phase/electricity loss detector,
- CO/CO_2 gases,
- Reed switch,
- Others by special order.

In the form of a single sensor, multiple component measurements are supported. The sensor data can be saved in internal memory and viewed in later time. The data can be downloaded in various formats. For user convenience, data can be viewed in the form of graphs. Each sensor can be assigned with specific thresholds, exceeding of which will trigger the device to complete a specific action.

The source of the sensor can be a 3-wire bus, a device interrogated by Modbus TCP/RTU protocols (eg. meters, counters, thermometers) or another Inveo device that supports sensors. Every external device readout can be assigned to Daxi as a sensor and perform all functions of sensors.

Actions

Functions performed by the module are available in the form of blocks defined by the user. An action is a set of functions executed by the module during it's recalled. An action can be recalled eg. when temperature exceeds a thresholds, input state changes, the device looses connection etc. Each of those events can be assigned with a different action or share an action with other events.

An action can contain the following operations (single or multiple):

- Control an output,
- Send a HTTP request,
- Send an E-Mail message,
- Send an SNMP Trap,
- Send an SMS,
- Send a MQTT message,
- Send data via TCP/UDP,
- Send a KNX notification,
- Save to an internal variable,
- Save to log memory.

Example

After the temperature is exceeded, send an E-mail after 1 minute and repeat 6 times every 10 minutes.

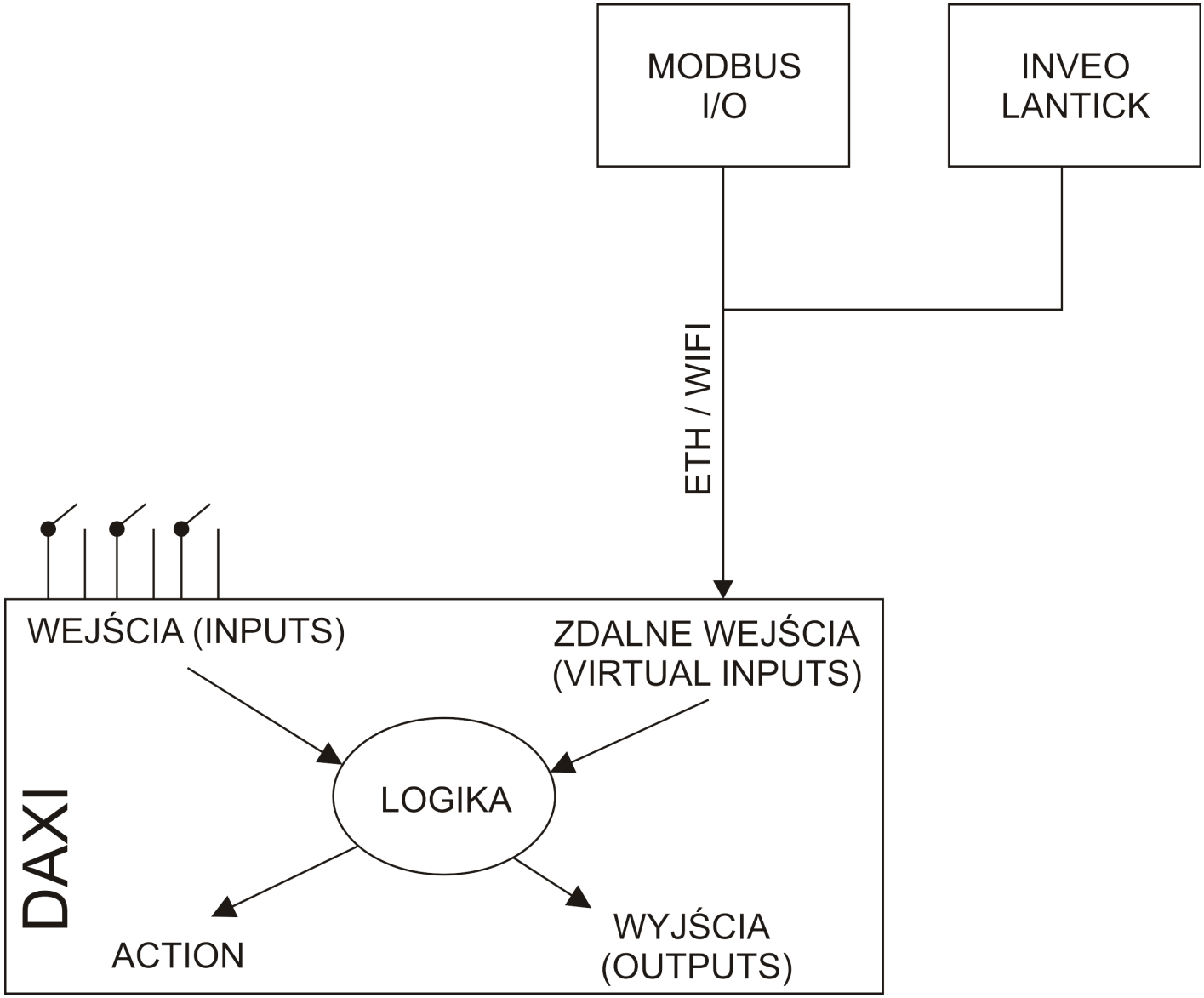
Inputs

Every input is equipped with a pulse counter, interference filter, ability to invert its state.

Built-in algorithms enable execution of multiple functions such as:

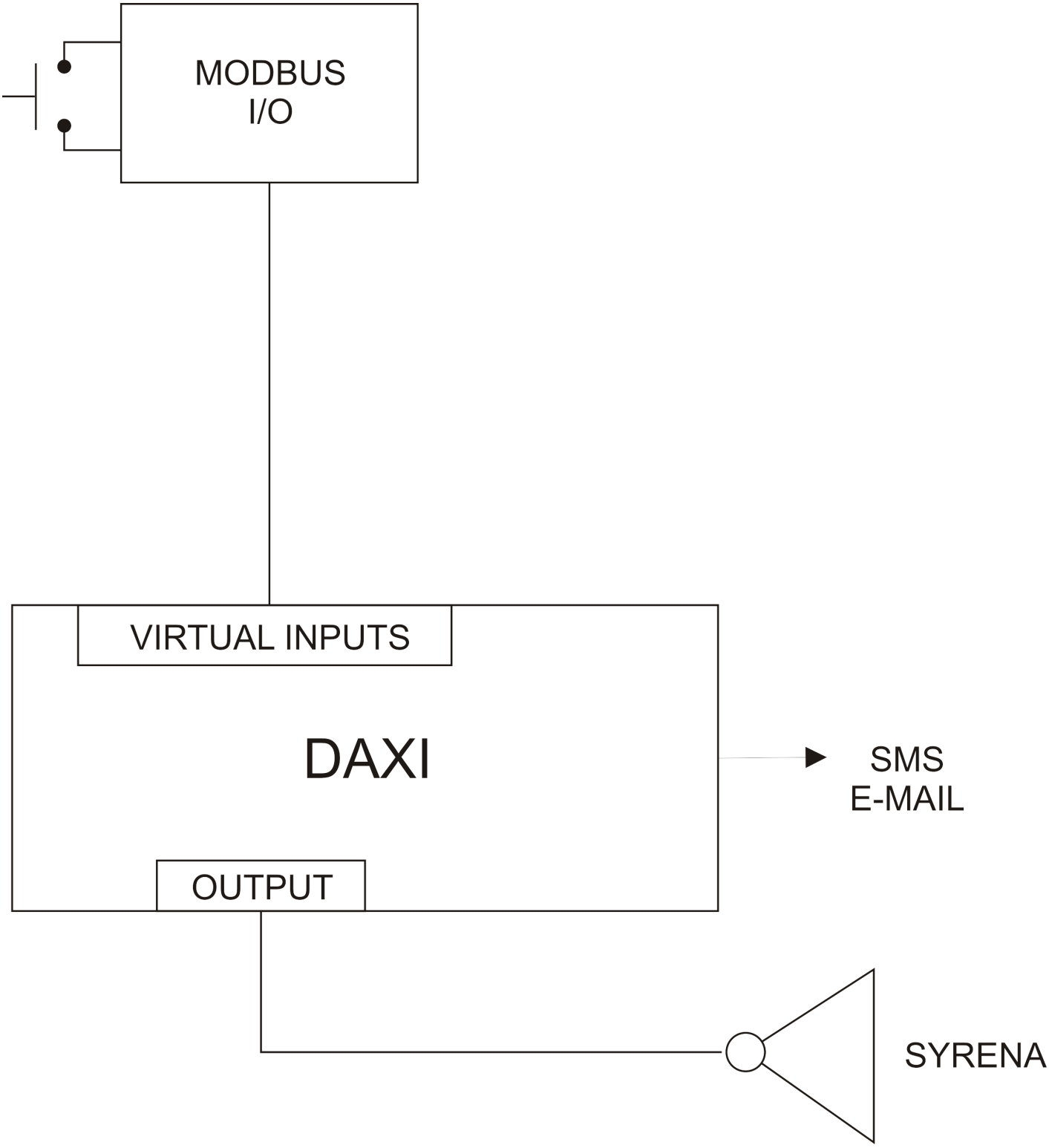
- Simple on-off relations,
- Longer press detection - allows 2 actions to be controlled with a single button,
- Counting the number of actuations and only after a preset time has elapsed recalling the corresponding function (up to 4 different functions) - allowing up to 4 separate circuits/devices to be controlled from a single pushbutton,
- Comparing pulse frequencies to a preset value and execution of the corresponding action on a change,
- Standard input pulse counting and sending this data to a server.

Input flow chart:



In addition to physical inputs, the module is equipped with virtual inputs. Data can be routed to them from other sources such as:

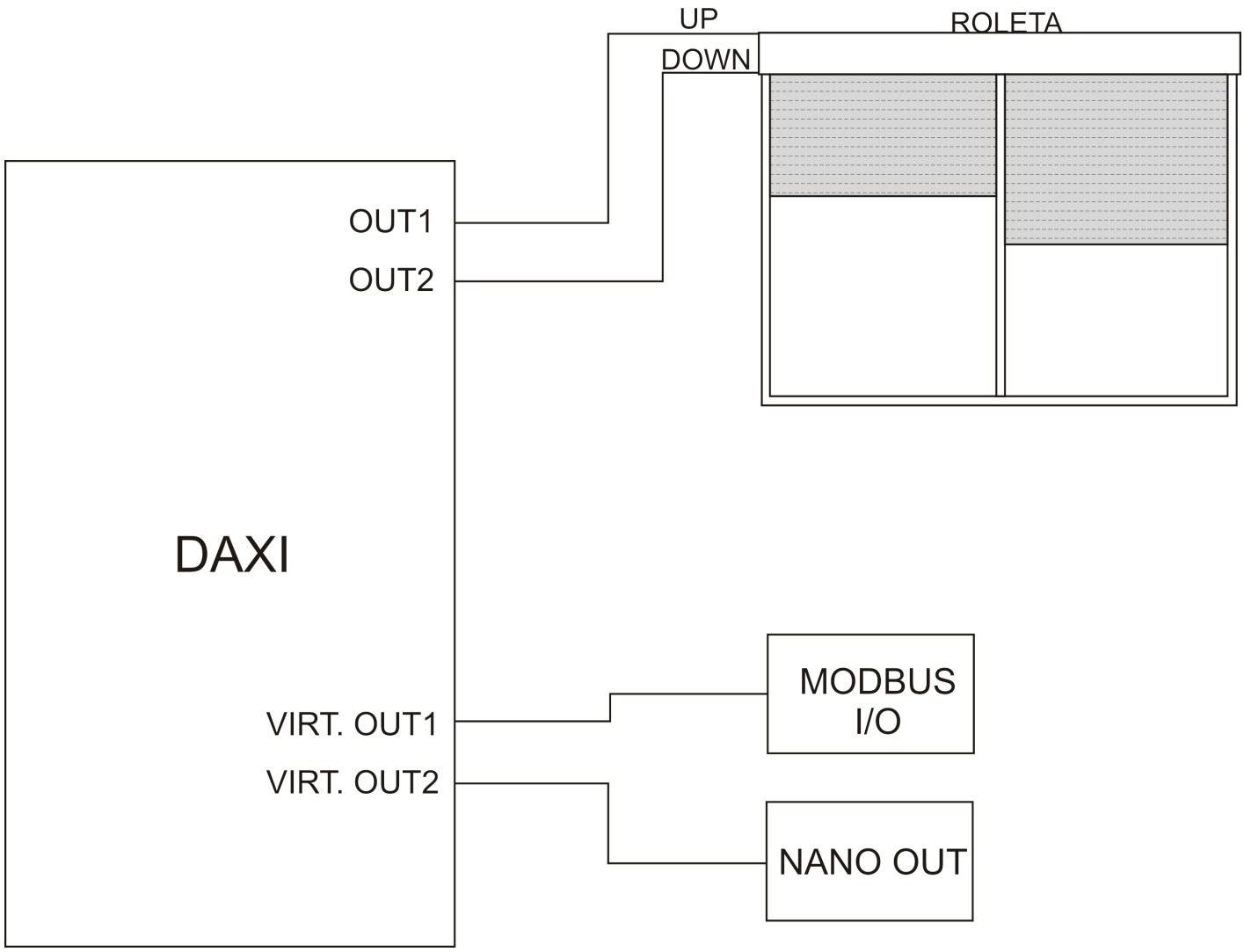
- Modbus devices,
- Other Inveo devices,
- Internal device states i.e. sensor alarm state, MQTT broker connection state or an internal variable - based on those values, a function can be assigned that causes eg. sending an E-mail message informing about the fact that a sensor is damaged or doesn't have a connection with the MQTT broker.



Outputs

The device's outputs can operate in the following modes:

- **Bistable** – Enable/disable,
- **Astable** – Continuous activation/deactivation every specified time,
- **Timed** – Activating an input activates it for a set time,
- **Shutter** – Groups 2 outputs for working with blinds/shutters.



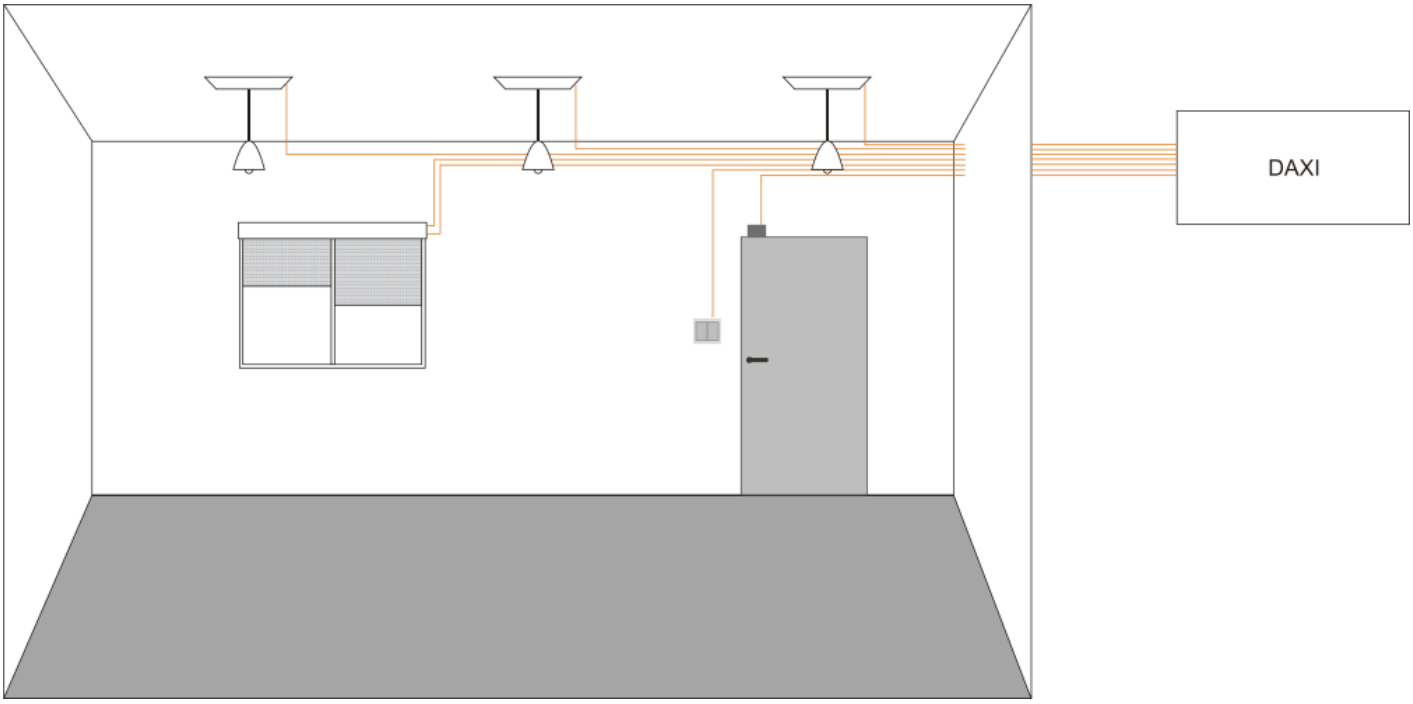
Outputs can be assigned with routes from other sources (eg. inputs, external devices channel states, connection states to external modules, internal variables, sensor alarm states). Inputs can be assigned to KNX groups and react to incoming notifications. Additionally, virtual outputs are provided that allow the output to be routed to external Modbus modules, Inveo modules or internal variables eg. alarm states, MQTT broker connection status.

Logic module

The user can connect the device states with any logical function (eg. input, output, sensor, variable state etc.) and control outputs or recall actions depending on the results. A Lua interpreter has been developed for advanced users, offering unlimited control options over device functions.

Integration with smart building systems

Built-in support for KNX-IP facilitates integration with building automation systems. Using logical and timer functions, it is possible to create any relation that can be used to control a building.



Integration with other devices

The device supports various network protocols, allowing for data exchange such as: HTTP(S), MQTT(S), MODBUS, SNMP, KNX, TCP/UDP, Email, SMS.

Cloud integration

Access to device data from any place is possible with cloud applications such as **CEAppMonitor** https://cemonitorapp.com/free/inveo.

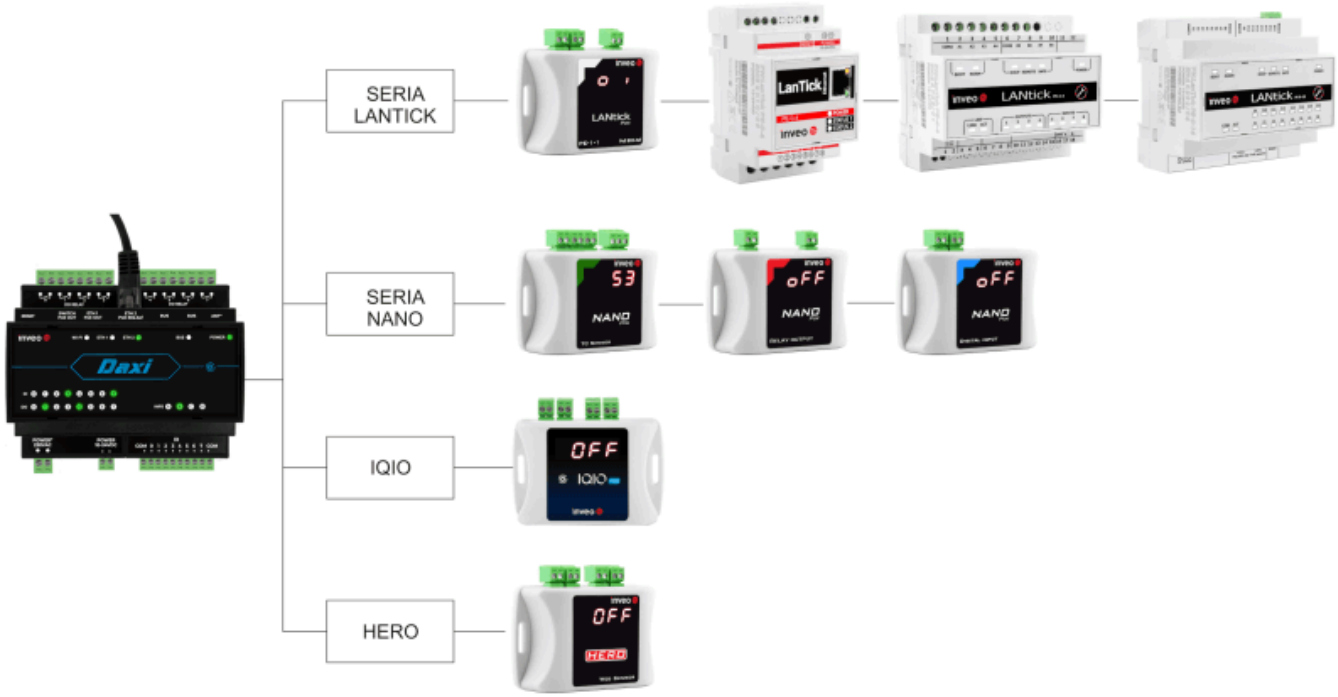


Modbus client and server

Built-in Modbus interrogation module reads data from other devices and shares it as a source of data for inputs, outputs and sensors. The device can be interrogated by other Modbus devices, making integration with input/output modules, sensors, PLC controllers etc. possible.

Integration with Inveo devices

The module recognizes other Inveo devices in the network and allows access to them from the level of the device (integrator). This allows for eg. configuration of a temperature sensor that will download data from a distant point or connection of an input or output to a remote device. The device cooperates with Nano, Hero, Lantick and IqIO modules.



Control of network device state

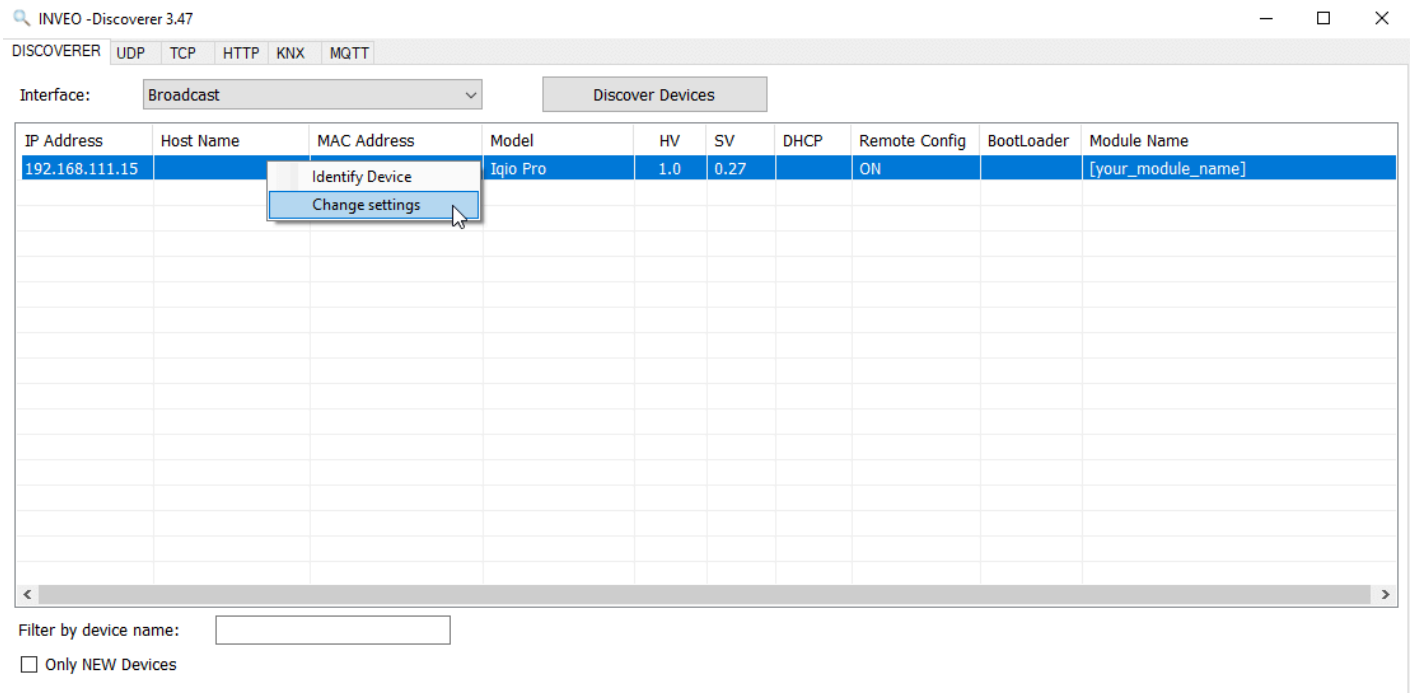
The module can check the status of other devices connected to the LAN/WiFi network by periodically sending ICMP (Ping) packets. Depending on the status of these devices, the module can send a notification or reset the device.



Network configuration

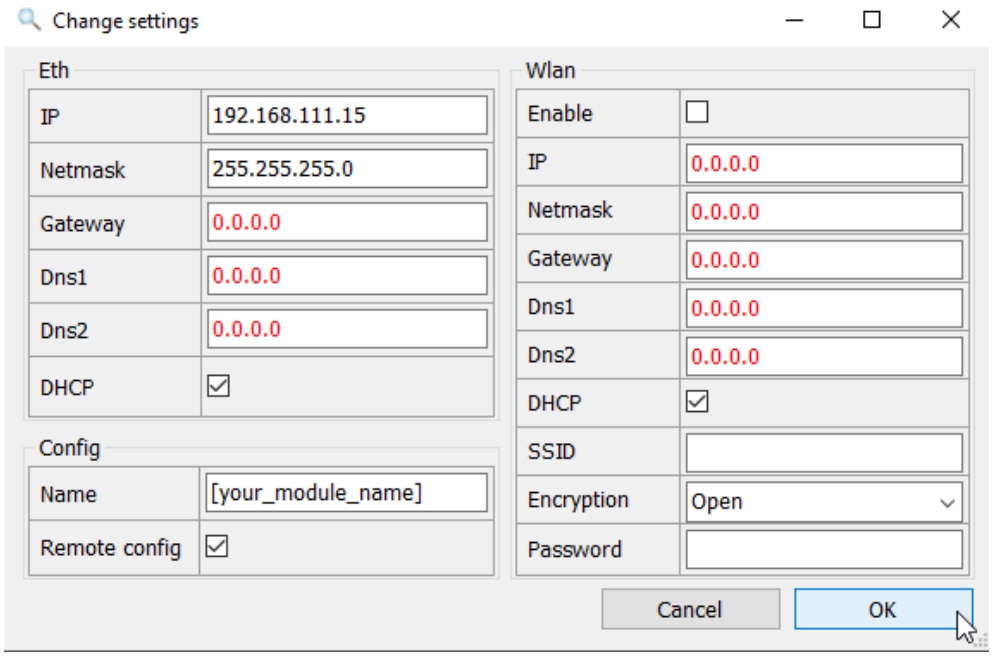
> **Warning**
>
> **After the initial power-up, it's needed to configure the device. It can be done in two ways, the simplest method is to use the Inveo Discoverer programme.**

Changing the device's IP address - Discoverer programme



After running the **Discoverer** programme and searching for the appropriate device:

1. Right-click the device,
2. Click the **Change** button.


Change settings
Eth
IP
192.168.111.15
Netmask
255.255.255.0
Gateway
0.0.0.0
Dns1
0.0.0.0
Dns2
0.0.0.0
DHCP
Config
Name
[your_module_name]
Remote config
Wlan
Enable
IP
0.0.0.0
Netmask
0.0.0.0
Gateway
0.0.0.0
Dns1
0.0.0.0
Dns2
0.0.0.0
DHCP
SSID
Encryption
Open
Password
Cancel
OK

After the dialogue window is opened you can:

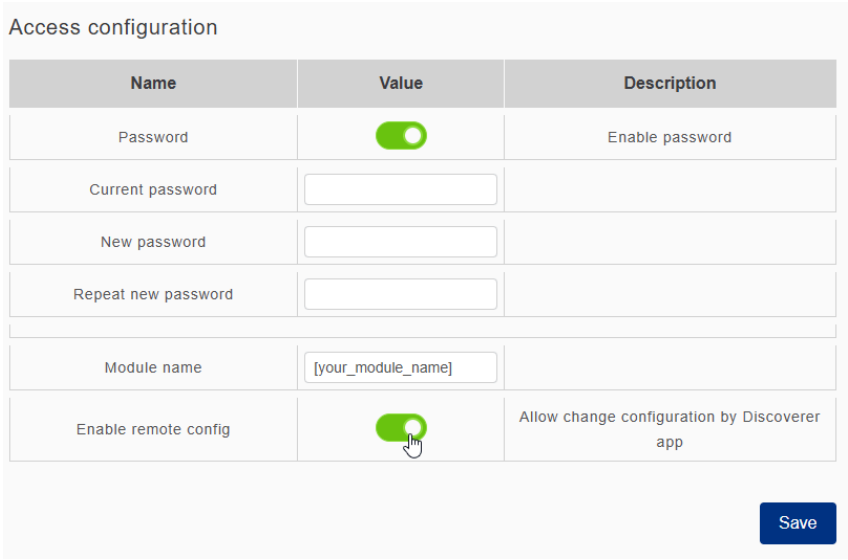
- Set the IP address, mask, gateway, DNS1/DNS2.
- Change the Host name.

> **Note**
>
> **The device will be configured after the Change button is pressed.**

If **Remote Config** is disabled (enabled by default), the device needs to be configured by changing the computer's subnet:

To enable remote configuration:

1. Go to the **Administration** tab,
2. In the **Access configuration** window, select **Enable Remote Config**.



> **Tip**
>
> **After making changes, apply them using the Save button**

Changing the computer's subnet address for configuration

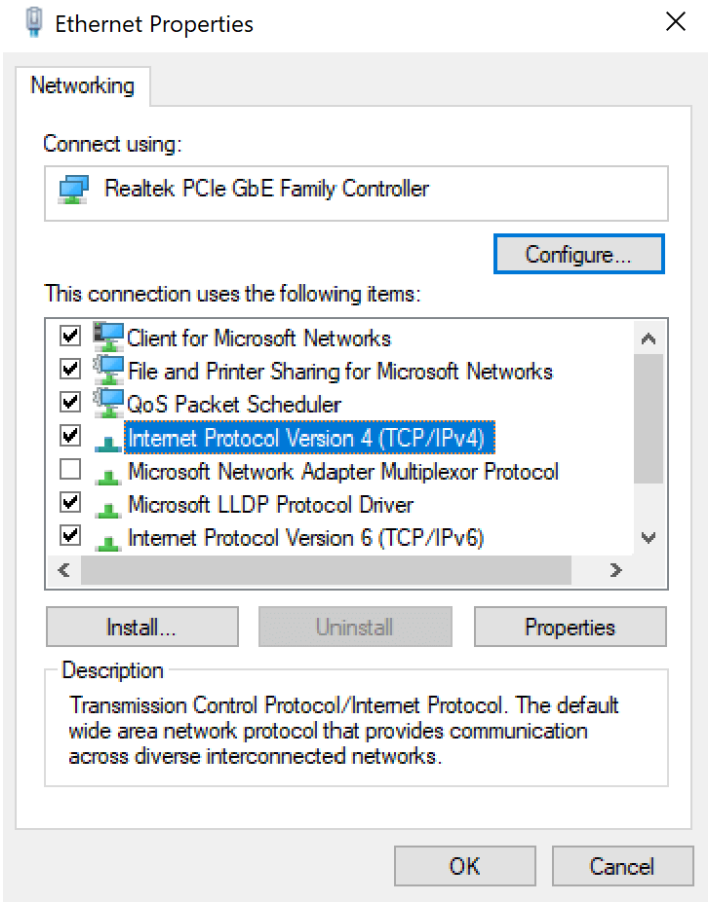
During configuration without **Discoverer** programme, it's needed to change the subnet address of the computer on the same network.

To access the computer's network configuration, do one of the following:

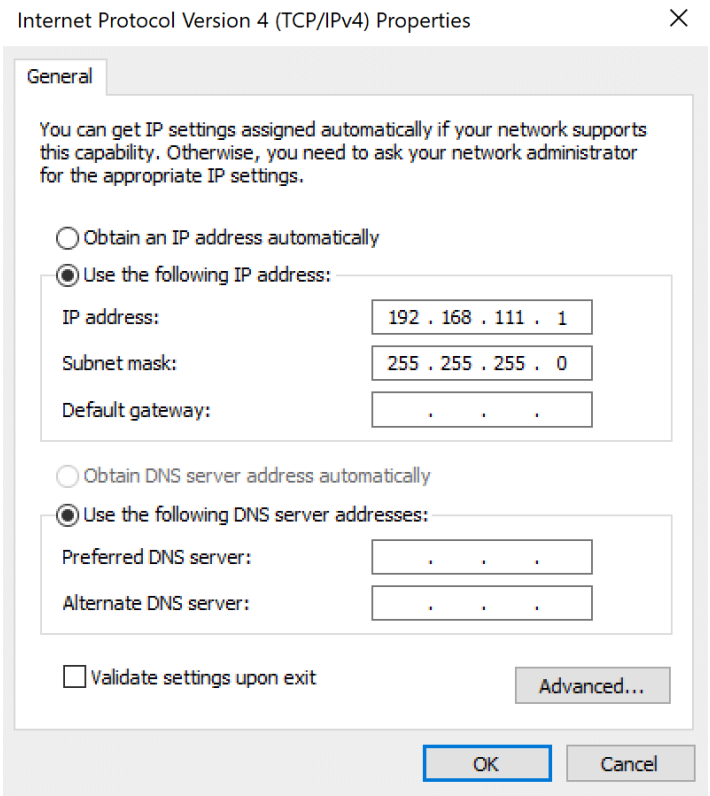
- Press **Win + R**, type in `ncpa.cpl`, and then press **Enter**.
- Alternatively, go to:
  Start → Control Panel → Network and Internet → Network and sharing center → Change adapter settings.

Then:

1. Select the network connection.
2. Right-click it and select **Properties**.
3. After selecting this option, network connection configuration screen will appear.



Select **Internet protocol version 4 (TCP/IPv4)** and type in the following parameters:

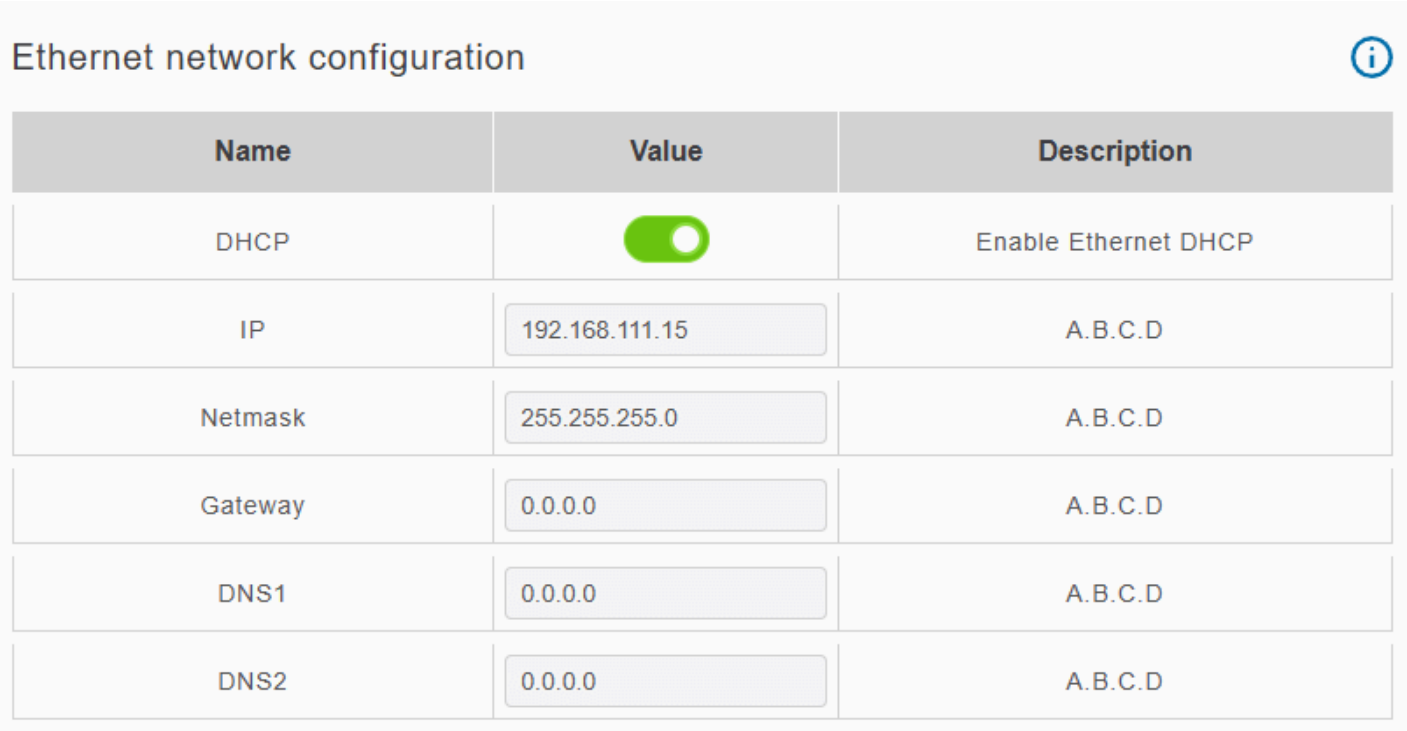


Note

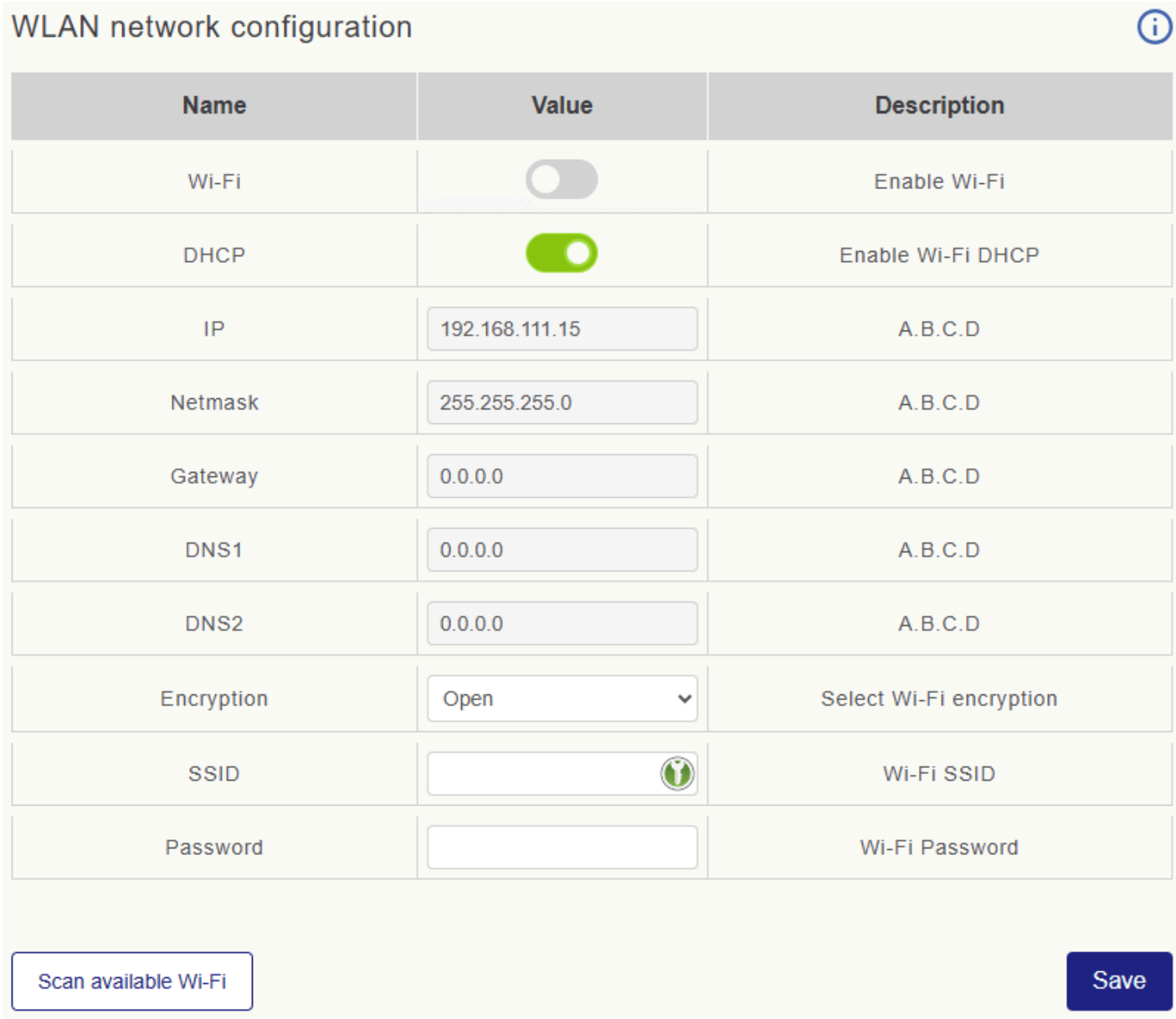
After applying the configuration with OK, start a web browser and type in the following address: 192.168.111.15. (Default user and password: admin/admin)

Network settings configuration

To customise the device's network settings, go to the **Administration / Network** tab. Parameters such as the IP address, subnet mask, gateway, DNS, and other network-specific options can be configured here. This tab allows the user to configure both the wired (**Ethernet network configuration**) and wireless (**WLAN network configuration**) connections.



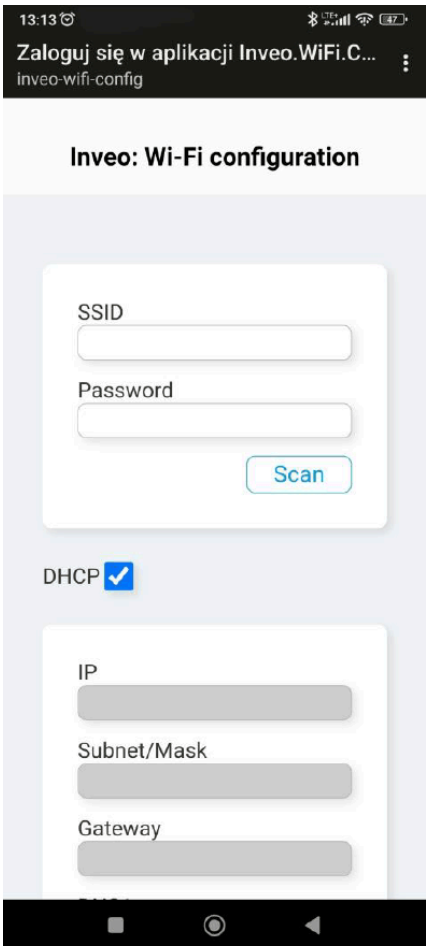
- **DHCP** – Enable/disable DHCP server,
- **IP** – Device IP address,
- **Netmask** – IP subnet mask,
- **Gateway** – Network gateway,
- **DNS1, DNS2** – DNS server addresses.



- **Wi-Fi** – Enable/disable Wi-Fi service,
- **DHCP** – Enable/disable DHCP server in Wi-Fi,
- **IP** – Device IP address,
- **Netmask** – IP subnet mask,
- **Gateway** – Network gateway,
- **DNS1, DNS2** – DNS server addresses,
- **Encryption** – Wi-Fi encryption type:
  - Open,
  - WEP,
  - WPA-PSK,
  - WPA2_PSK,
  - WPA_WPA2_PSK,
  - WPA3_PSK.

- **SSID** – Network name,
- **Password** – Wi-Fi access password.

Wi-Fi connection configuration



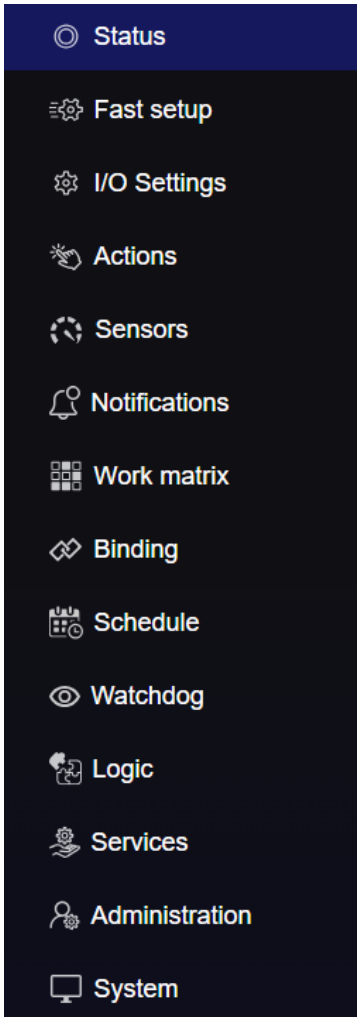
1. For 3 minutes after the power has been applied to the device (during **Configuration mode**), Wi-Fi connection can be configured. To do that, press and hold the **RESET** button until **AP** is displayed.
2. Activate Wi-Fi network search on your smartphone or other device. A network called "Inveo-wifi-config" should appear on the list.
3. Connect with the network.
4. After connection is established, click **scan** in the configuration interface or enter the Wi-Fi SSID into the correct field.
5. Select the network the device should connect to from the list of available options.
6. Enter the correct password to the selected network.
7. If the DHCP server is unavailable, manual configuration can be conducted after de-selecting **DHCP**.
8. **SUCCESS** will appear if the settings are saved successfully. The device will restart and resume normal operation.

Device configuration

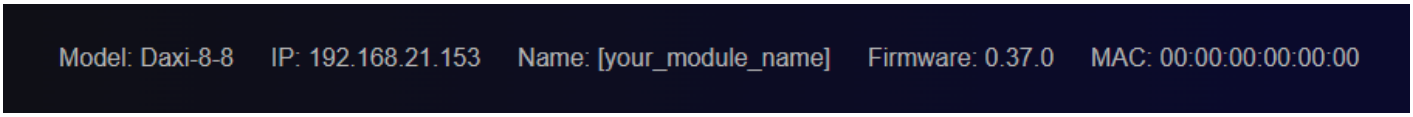
Module's www interface

The web interface allows for intuitive and advanced device management. After typing in the device's IP address into the browser's search bar, a webpage opens allowing full configuration and customisation of the device's operating parameters according to the individual user needs. On the left-hand side of the screen, a list of tabs is located, allowing for quick access to various functions and settings. Available tabs:

> **Note**
>
> **Some tabs are only visible after activationg the Advanced view.**



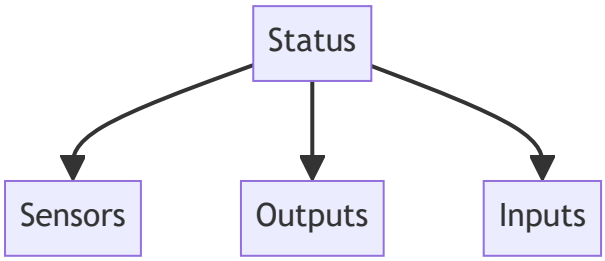
In the upper portion of the website, an information ribbon is located, informing the user about key device data, such as: model, IP address, unique user-assigned name, firmware version, and MAC address.



Thanks to the website, the user can modify settings, configure parameters, and monitor device operation in real-time. The webpage is the central control point, allowing for effective management and customization according to the changing user needs.

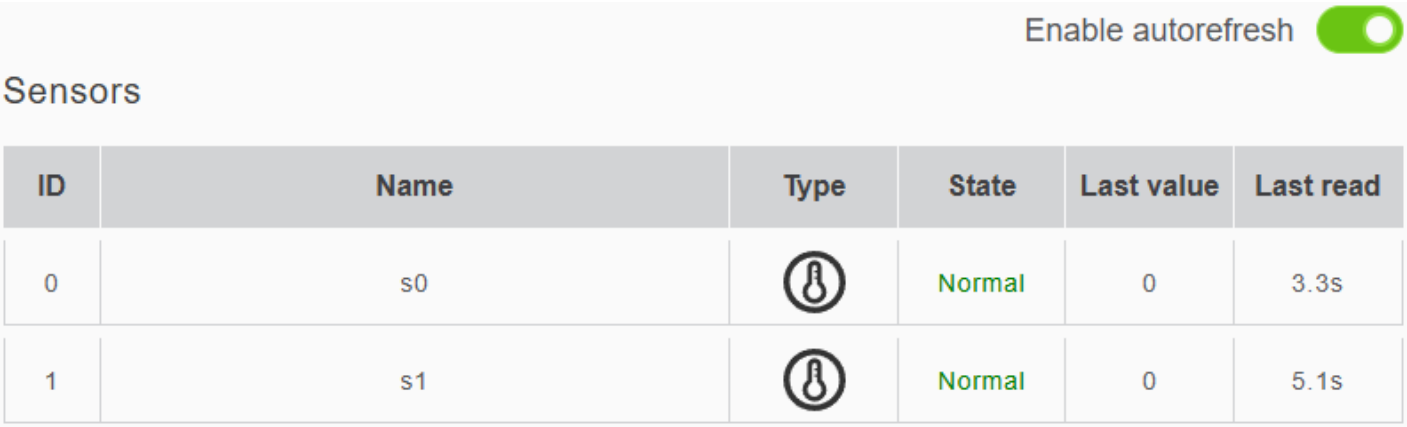
Device status preview

In the **Status** tab, all information about currently used outputs, inputs and sensor readings can be found.



Sensors window

In this window, current sensor readings are displayed. Sensors can be configured in the **Sensors** tab.



Using the **Enable autorefresh** button, the user can enable or disable automatic readout refreshing.

Individual table columns contain the following data:

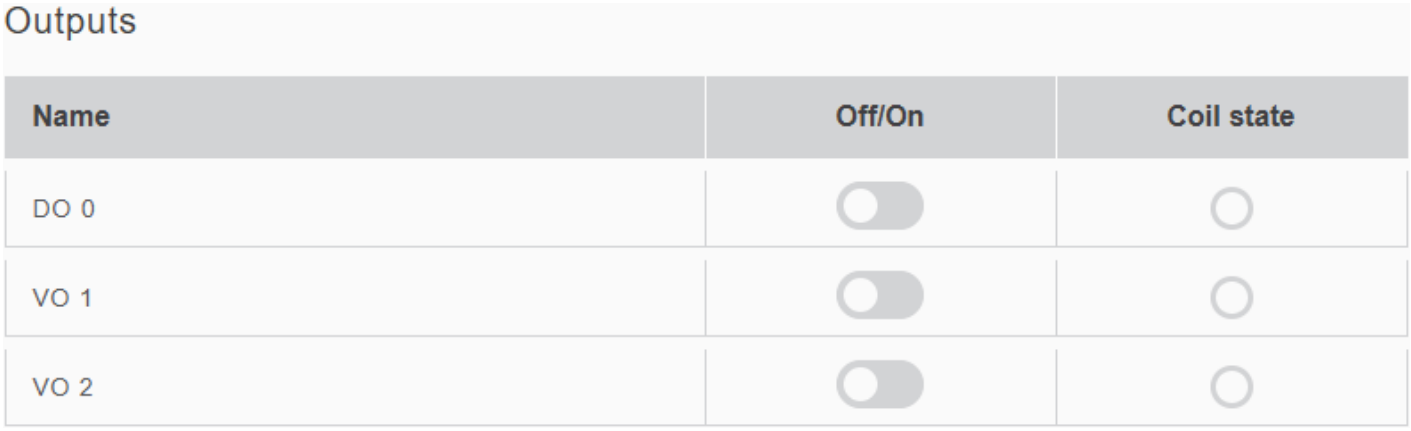
- **Name** – Sensor name defined in the **Sensors** tab,
- **Type** – Graphic representation of the sensor type:
  - – Temperature sensor,
  - – Humidity sensor,
  - – Input,
  - – Analogue current sensor 4-20mA,
  - – Pressure sensor,
  - – Analogue voltage sensor 0-10V DC,

-  – Carbon monoxide presence sensor (CO),
-  – Carbon dioxide presence sensor (CO2).

- **State** – Sensor state:
  - **Error** – Reading error (sensor damaged or connected incorrectly),
  - **Normal** – Sensor gives correct readings that are within normal limits,
  - **Warn L** – Lower threshold warning,
  - **Warn H** – Upper threshold warning,
  - **Alert L** – Lower alarm threshold,
  - **Alert H** – Upper alarm threshold.
- **Last value** – Last read value,
- **Last read** – Time that has passed from the last reading (value is refreshed constantly when autorefresh is on).

Outputs window

In this window, current output state is displayed. Outputs can be defined in the **I/O Settings** tab.



Individual table columns contain the following data:

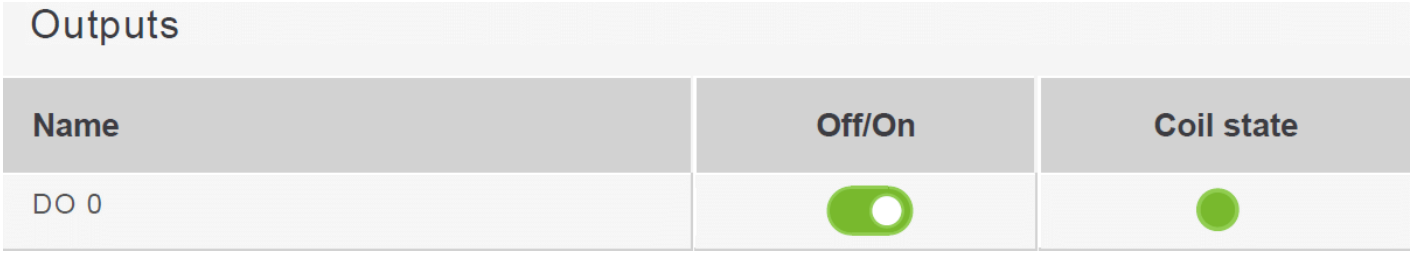
- **Name** – Output name defined in the **I/O Settings** tab.

The following input types are available:

- **Digital – DO** – Physical module outputs,
- **Virtual – VO** – Virtual outputs used for binding and relations configuration.

If the output state is dependent upon different factors, appropriate information is shown under its name:

- **output unavailable – assigned to the shutter**,
- **output unavailable – output is routed** – See section **Binding**.
- **Off/On** – Current output state, clicking the left mouse button in this area will change the output state - this option enables manual output control,
- **Coil state** – Current relay coil state - green colour means that the relay is activated.



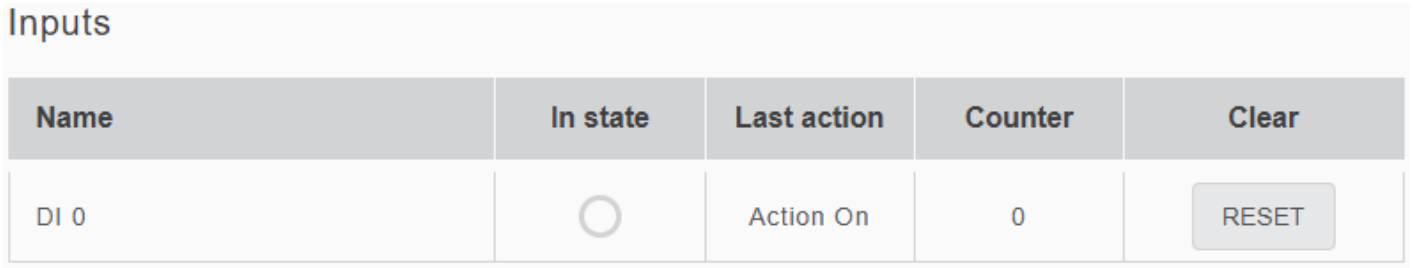
Output activation (**Off/On** column in the table) is not always identical as the **Coil state**.

> **Example**
>
> **If the output is configured in astable mode, with Time on and Time off parameters defined, enabling the output in the Status tab changes the state in the Off/On column. In this case we can observe the coil state is alternately activated/deactivated, according to the parameters defined in Time on / Time off.**

> **Tip**
>
> **If the output is configured for shutter control or other input/output state has been bound to it, testing its operation in the Off/On column will not be possible.**

Inputs window

In this window, current input state is displayed. Inputs can be defined in the **I/O Settings** tab.



Individual table columns contain the following data:

- **Name** – Input name defined in the **I/O Settings** tab,
- **In state** – Input state,
- **Last action** – Last registered action (eg. Action On),
- **Counter** – Displays the amount of times the input has been activated since last reset,
- **Clear** – Resets the counter.

Device status preview via a JSON resource

The device's state can be previewed by recalling the **JSON** resource: `http://<IP_address>/data/status.json`

Example response

▼ Details

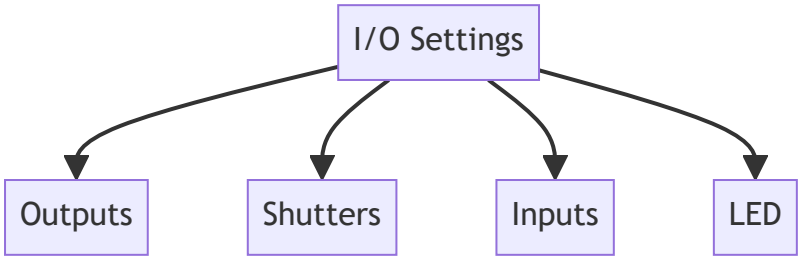

```
{
"status": {
    "model": "Daxi-8-8",
    "name": "[your_module_name]",
    "mac": "00:00:00:00:00:00",
    "output": [
    {
        "name": "DO 0",
        "on": 0,
        "coil": 0
    },
    {
        "name": "VO 8",
        "on": 0,
        "coil": 0
    }
    ],
    "input": [
    {
        "name": "DI 0",
        "state": 0,
        "action": 1,
        "cnt": 0
    }
    ],
    "shutter": [
    {
        "name": "Shutter 0",
        "isTop": 0,
        "isBottom": 0,
        "currentTask": "idle",
        "currentPos": -1
    }
    ],
    "repeat_next": [2424, 2424, 2424, 2424],
    "sensor": [
    {
        "id": 0,
        "name": "Sens 0",
        "value": 0,
        "unit": "",
        "state": 0,
        "last": 1663
    }
    ],
    "logic": 0,
    "virtstate": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    "build": "Aug 11 2026 08:42:32",
    "coredump": 0,
    "uptime": "2424",
    "sntp_update": 0,
    "mqtt_conn": 0,
    "fw": "0.37.0",
    "sunrise": "05:44",
    "sunset": "17:48",
    "time": "09:22:29",
    "date": "12-10-2026",
    "gsm": {
    }
}
}
```

- **model** – Device model,
- **name** – User-assigned module name,
- **mac** – Module's MAC address,
- **output**
  - **name**,
  - **on** – Output state,
  - **coil** – Relay coil state.
- **input**
  - **name**,
  - **state** – Input state – 0 – input inactive, 1 – input active,
  - **action** – Last input action – 1 – Action On, 2 – Action Off,
  - **cnt** – Input counter value.
- **shutter**
  - **name**,
  - **isTop** – Shutter is in the upper position,
  - **isBottom** – Shutter is in the lower position,
  - **currentTask**,
  - **currentPos** – Current shutter position.
- **repeat_next** – Time until the next periodic action is executed (if defined),
- **sensor**
  - **id** – Sensor's ID number,
  - **name**,
  - **value** – Measured value,
  - **unit** – Measurement unit,
  - **state**,
  - **last** – Time since last readout (in seconds).
- **logic** – Bitmask based on the indexes of satisfied logic conditions,
- **virtstate** – Internal variable values,
- **build** – Firmware build date,
- **coredump** – Critical program error indicator,
- **uptime** – Time in seconds since the last device restart,
- **sntp_update** – Last time downloaded from the SNTP server in the form of a Unix timestamp,
- **mqtt_conn** – MQTT broker connection status,
- **fw** – Firmware version,
- **sunrise** – Calculated sunrise time,

- **sunset** – Calculated sunset time,
- **time**,
- **date**,
- **gsm** – Function unused in Daxi.

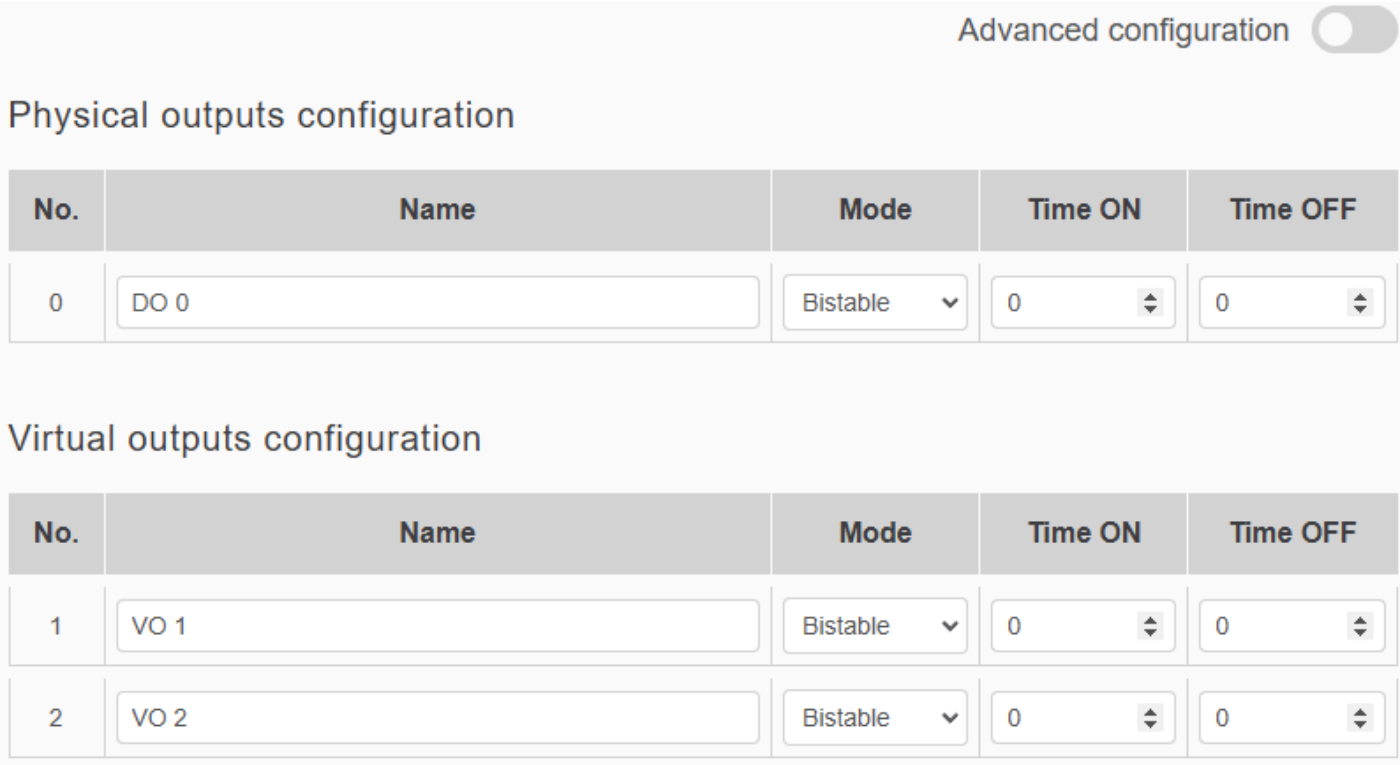
Inputs/outputs configuration

In the **I/O Settings** tab, advanced configuration options can be accessed, allowing the user to define device operation. Here, behaviour of inputs and outputs can be configured. Additionally, this tab gives users the option to configure the display.

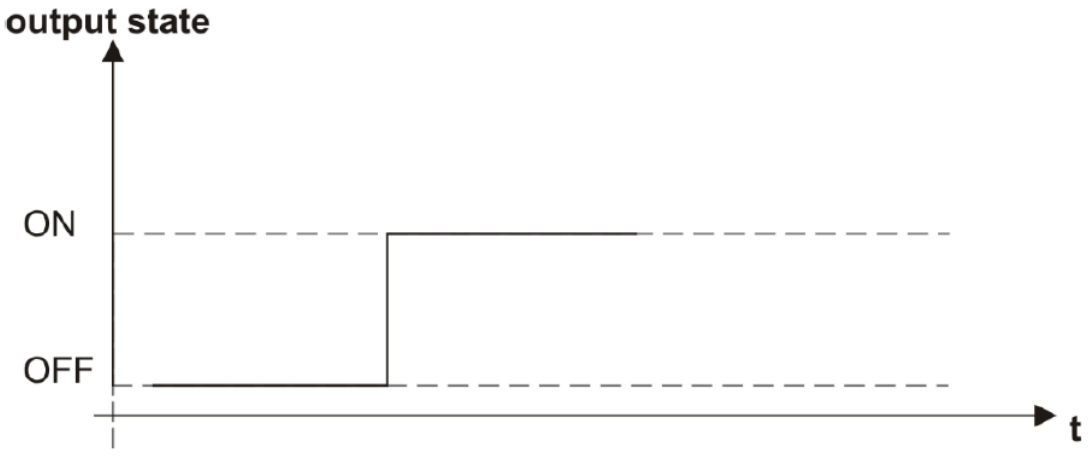


Outputs

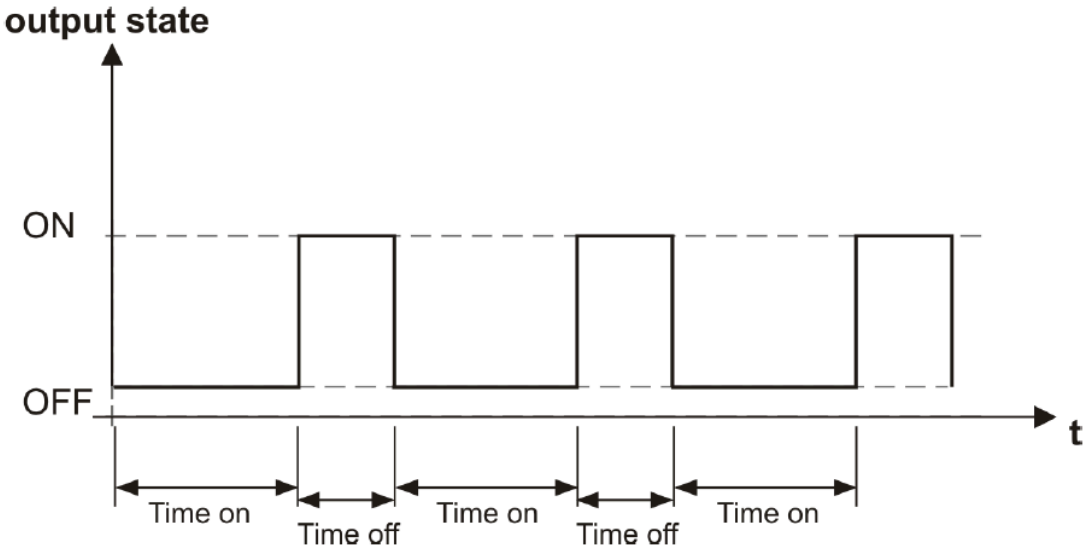
This tab allows configuration of outputs – both physical and virtual *(option)*. The following settings can be changed in individual columns:



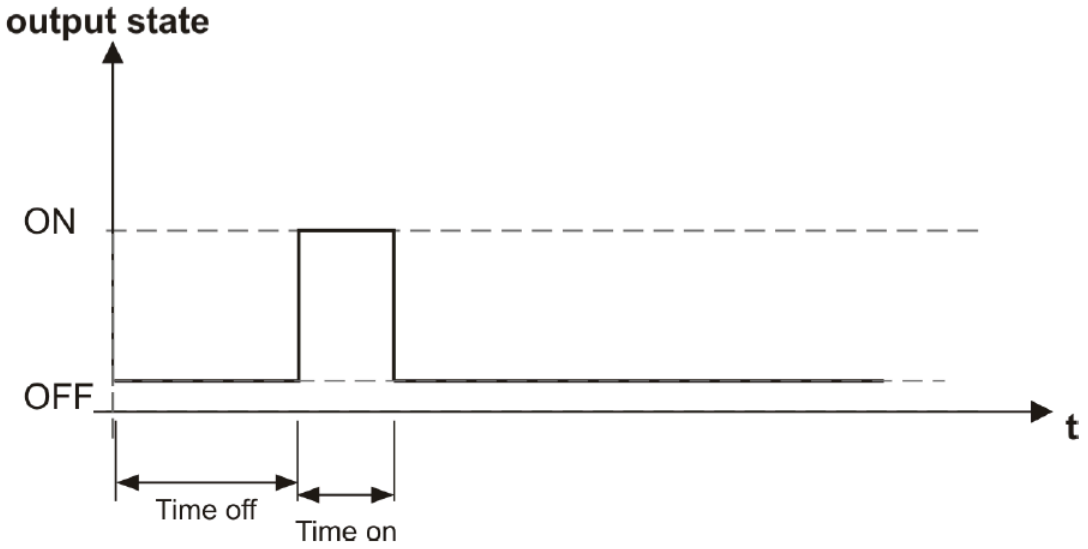
- **Name** – This field allows the output name to be changed,
- **Mode** – Output operation mode:
  - **Disable**,
  - **Bistable**,



  - **Astable** – Periodic mode - output is activated and deactivated periodically. The activation and deactivation times are defined by the parameters **Time on** and **Time off**,

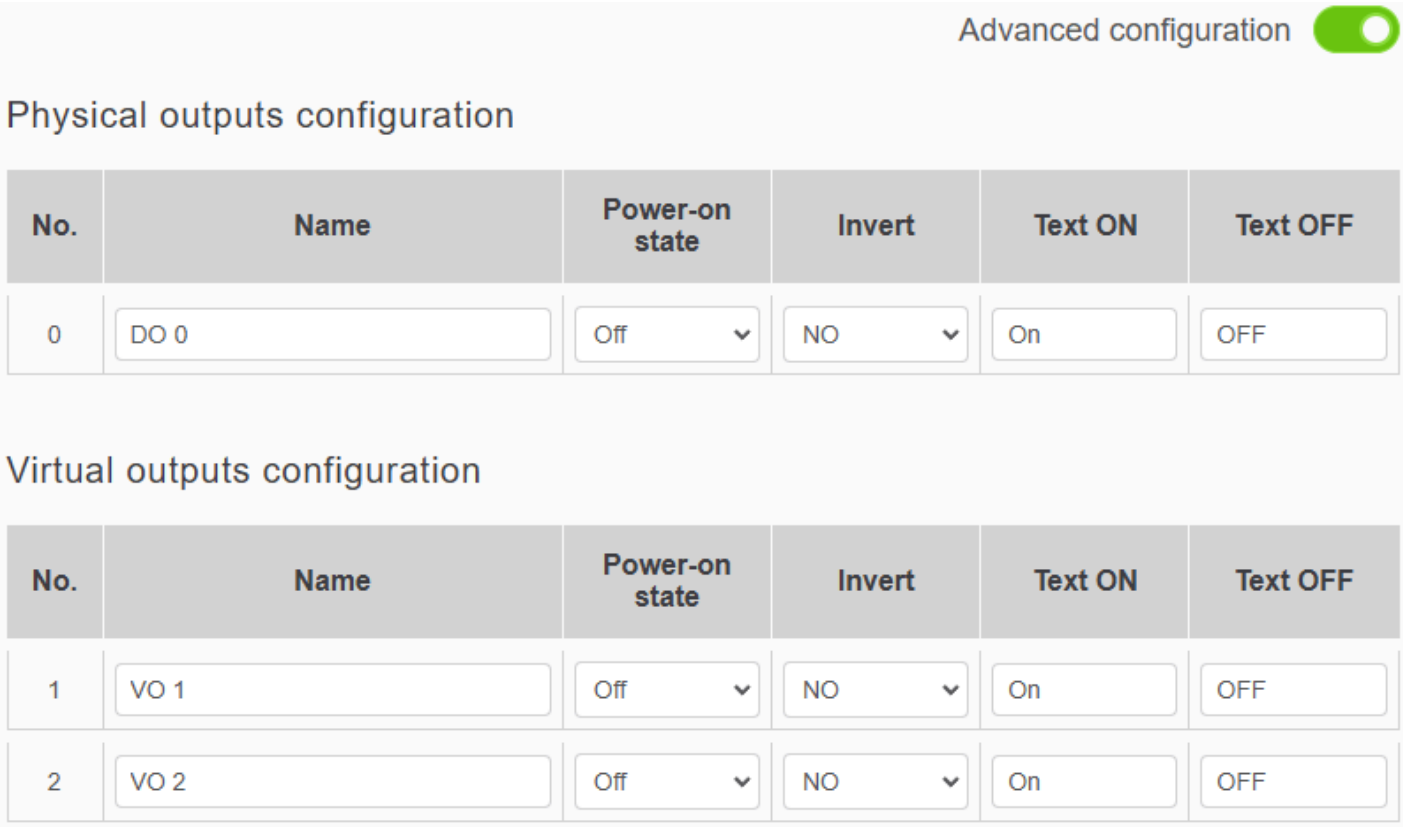


- **One-pulse** – Single output trigger mode - output is activated for a time defined in **Time on** after time defined in **Time off** passes,



- **Time on** – (parameter used in **Astable** and **One-pulse** mode),
- **Time off** – (parameter used in **Astable** and **One-pulse** mode).

Clicking **Advanced configuration** enables changing of advanced output settings:

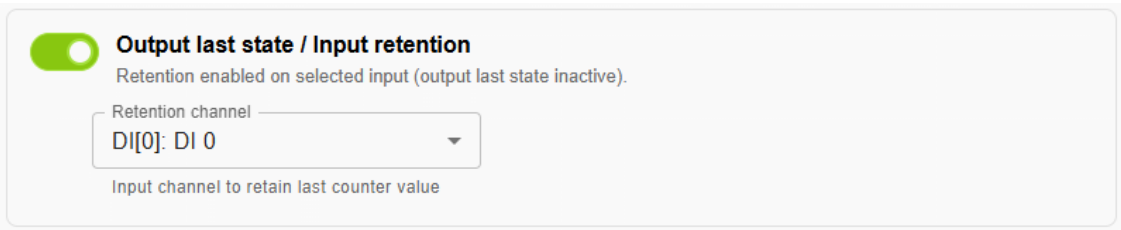


- **Name** – This field allows the output name to be changed,
- **Power-on state** – State that the device should reach after power is connected to the module:
  - **Off**,
  - **On**,
  - **Last** – State before the power was disconnected.
- **Invert** – Output channel inversion:
  - **NO** – Normally open,
  - **NC** – Normally closed.
- **Text ON** – Text dispayed on the LED display when the output is activated,
- **Text OFF** – Text dispayed on the LED display when the output is deactivated.

LAST OUTPUT STATE OR COUNTER RETENTION

The module offers the possibility to store the last output or specified input counter state in non-volatile memory.

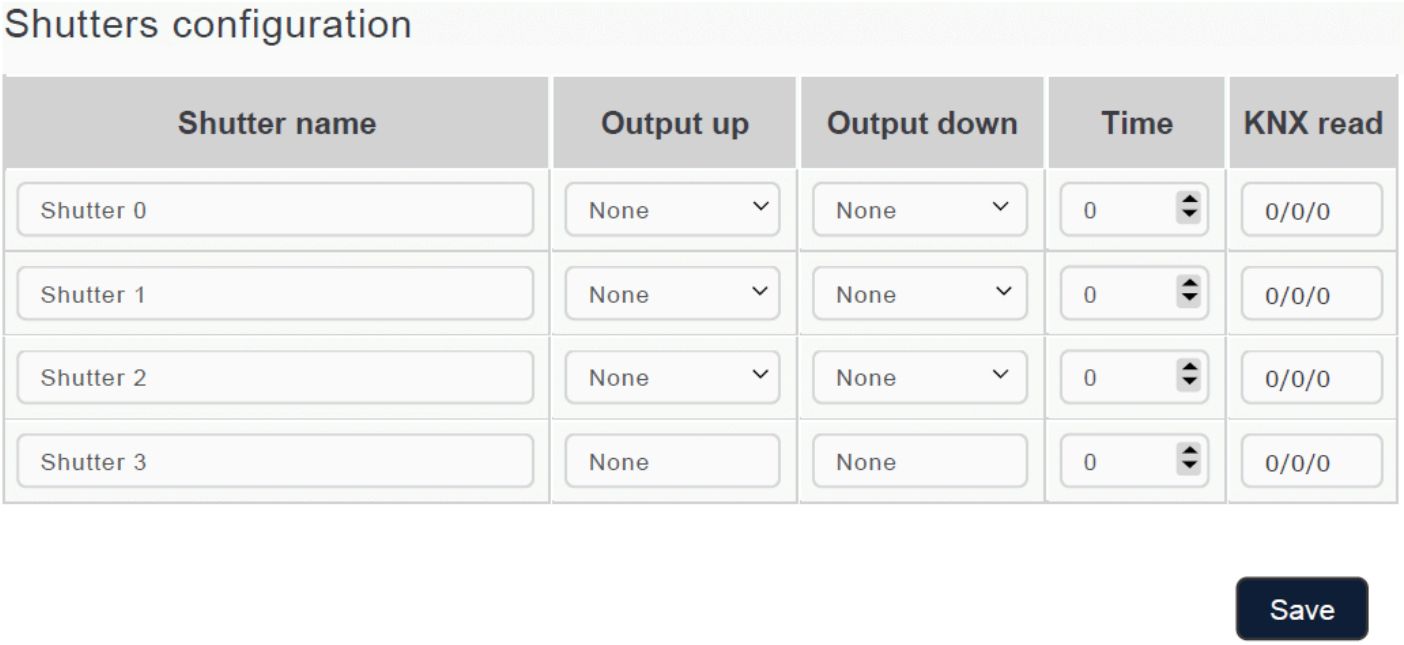
The **Ouptut last state / Input retention** slider enables selection of the desired setting.



Shutters

This tab allows configuration of shutters. In shutter control mode, Daxi automatically switches one of two relays, which results in the opening or closing of the shutter.

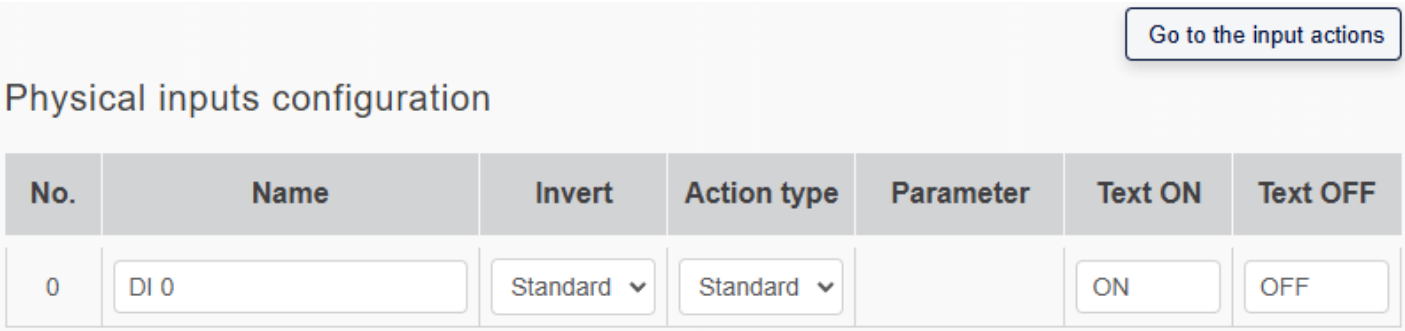
The following settings can be changed in individual columns:



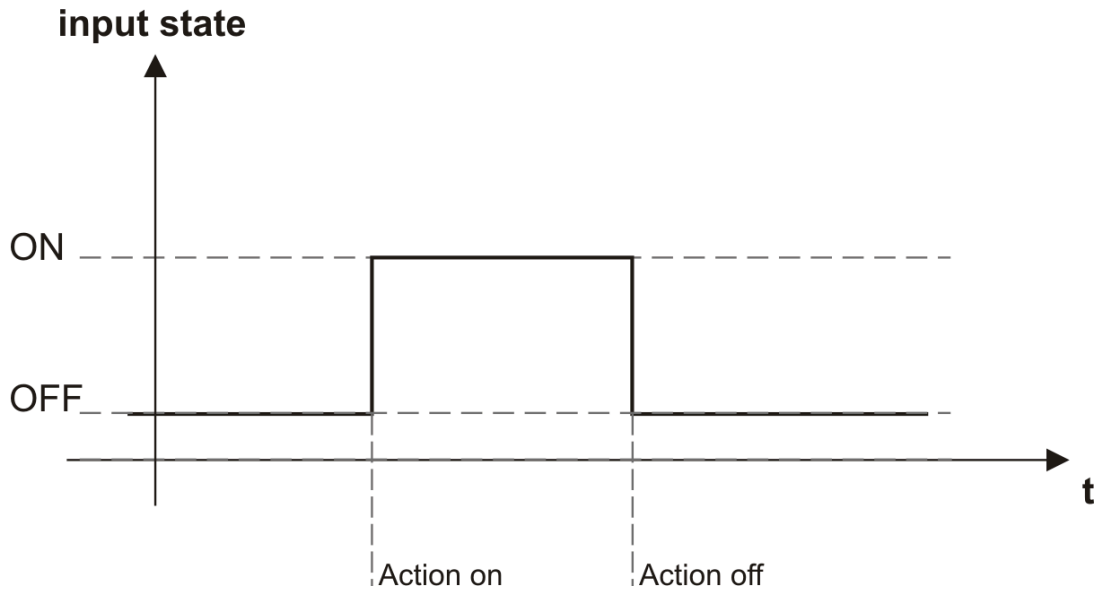
- **Shutter name** – Defined by the user,
- **Output up** – Allows selection of the relay that will be used to open the shutter,
- **Output down** – Allows selection of the relay that will be used to close the shutter,
- **Time** – Time required for the shutter to open/close entirely (in seconds),
- **KNX read** – KNX group assignment.

Inputs

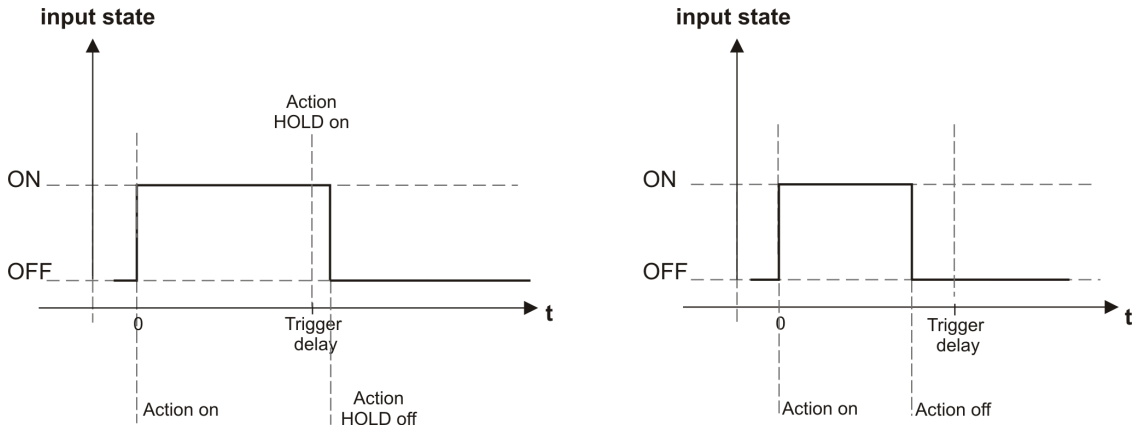
This tab allows configuration of inputs supported by the device. The following settings can be changed in individual columns:



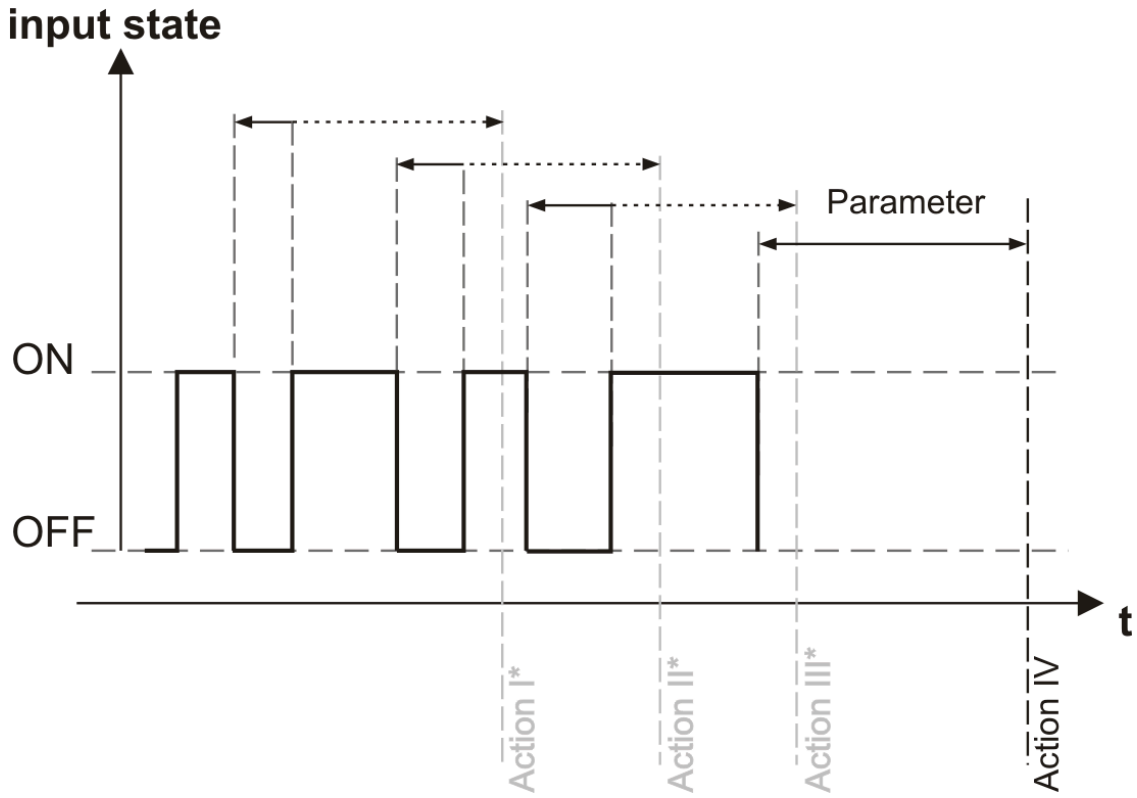
- **Name** – This field allows the input name to be changed,
- **Invert** – Input channel inversion,
- **Action type**:
  - **Standard** – Action is triggered by input activation/deactivation,



- **Hold** – Triggering of a defined action is dependent upon the pulse length on the input. The pulse recalls the action defined as **Action on**. If the pulse persists and exceeds the time in the **Parameter** field - action defined as **Action hold on** is recalled. If the pulse ceases before the time defined in **Parameter** - **Action off** is recalled. If the pulse ceases after the time defined in **Parameter** field - **Action hold off** is recalled,

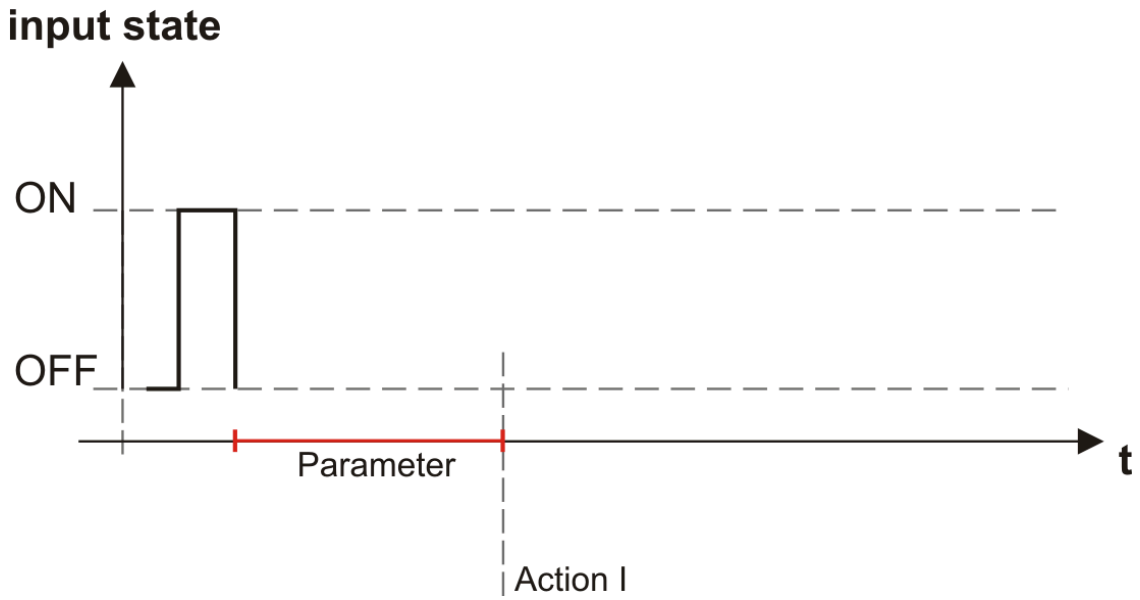


- **Cnt** – Pulse counter during specified time period - the action is recalled by the defined amount of activations in the time period specified by **Parameter**,

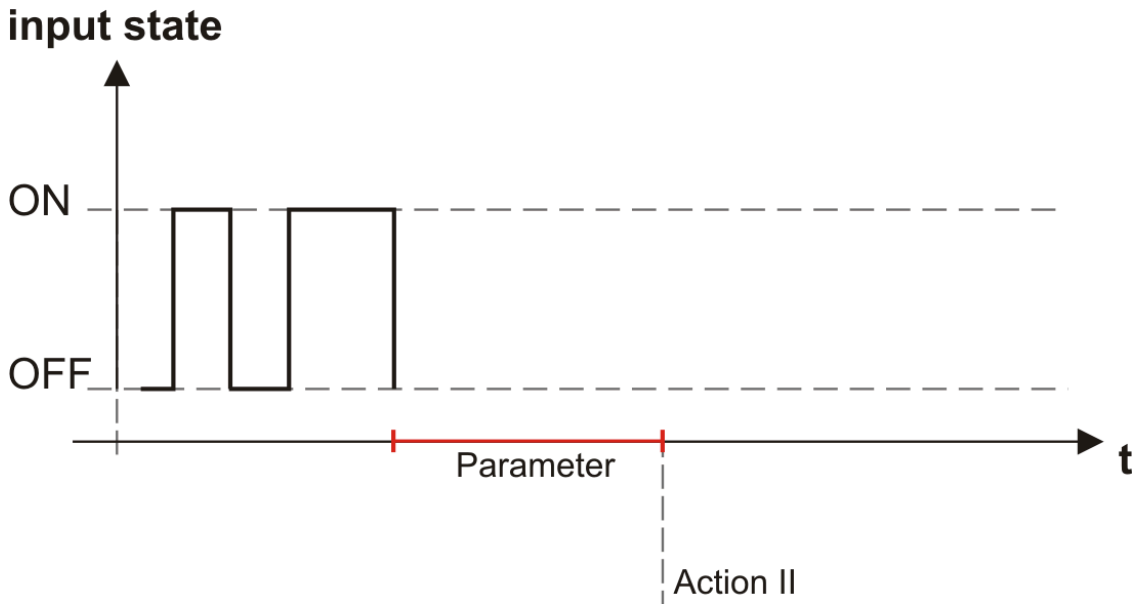


Action will be recalled only when no subsequent pulse occurs (input is not activated again). The defined action is recalled after the pulse absence delay time passes (i.e. a button is released). If another pulse occurs during this delay time, the countdown is stopped.

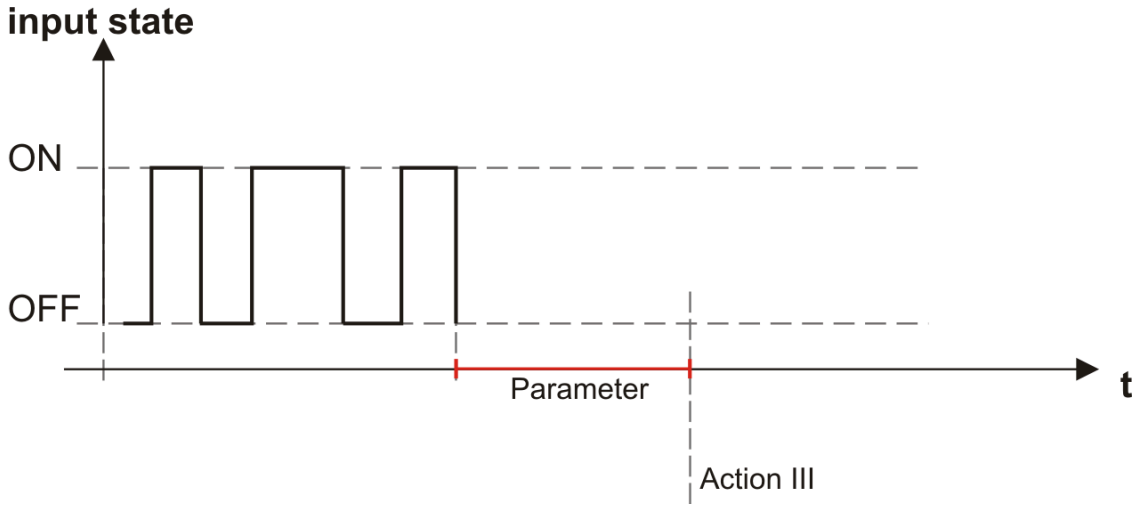
Recalling action I:



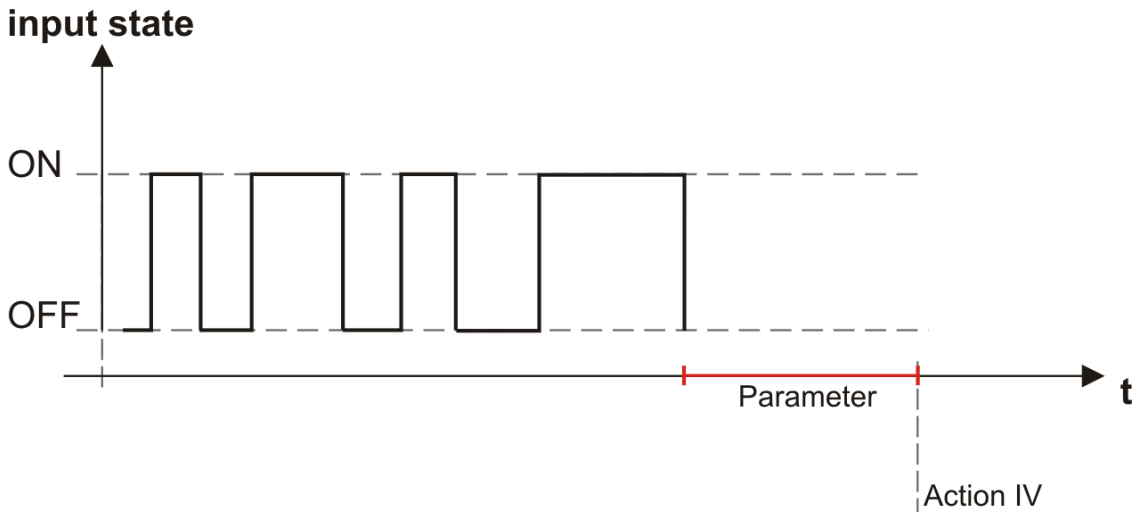
Recalling action II:



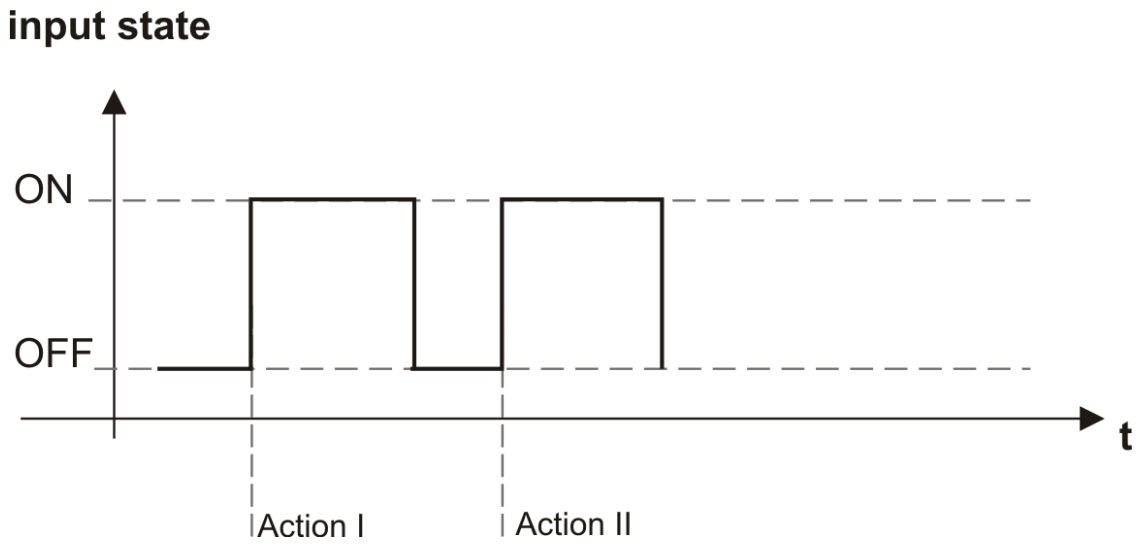
Recalling action III:



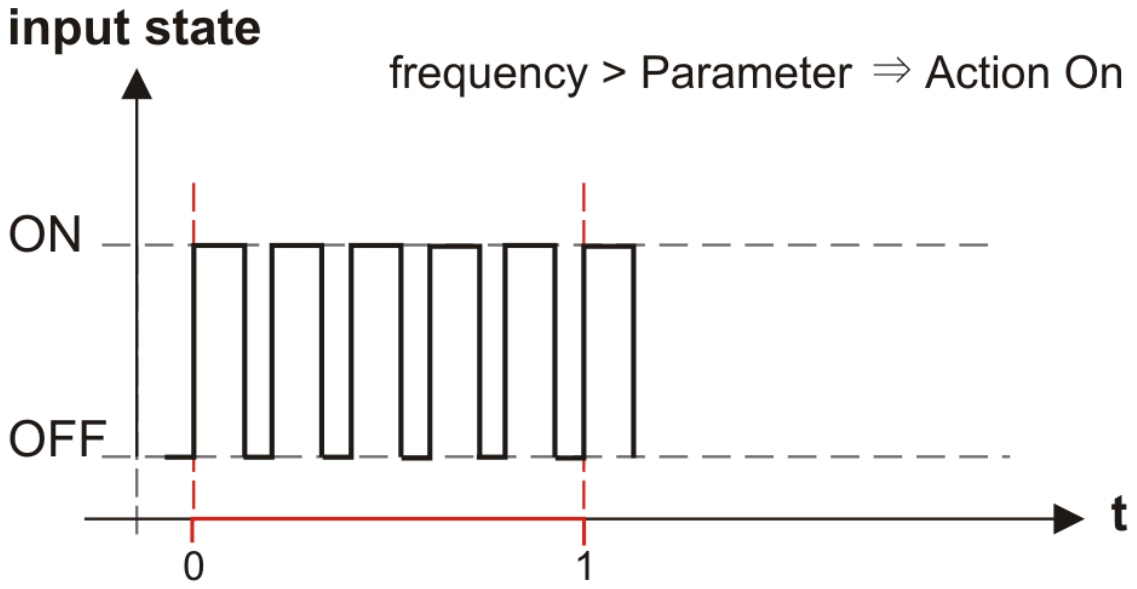
Recalling action IV:

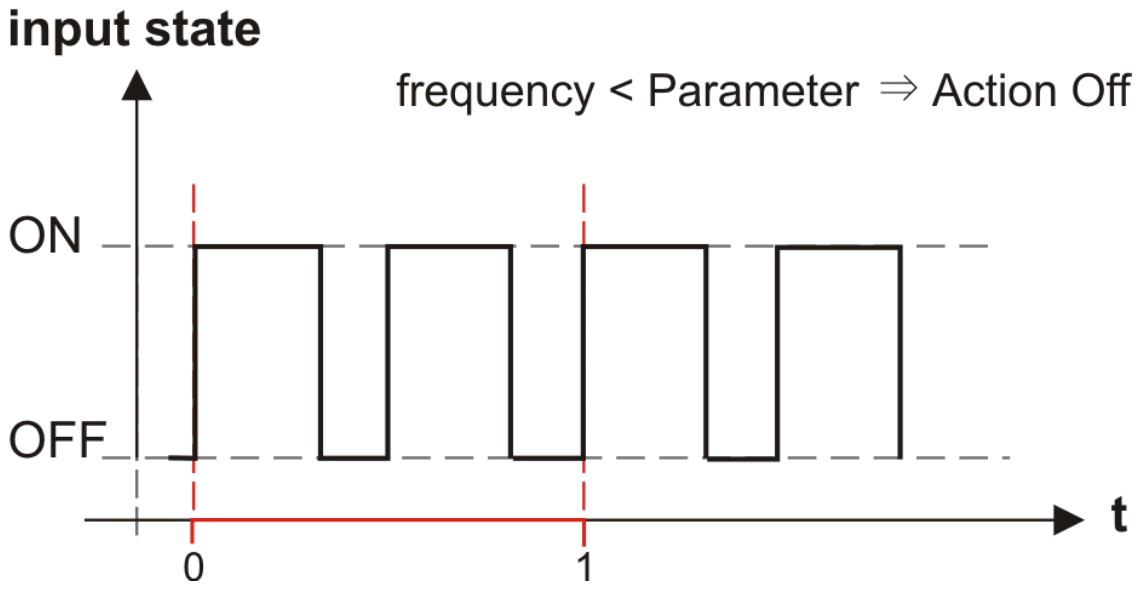


- **Toggle** – Subsequent pulses on the input recall **Action I** / **Action II** alternately,



- **Freq** – Action recall is dependent upon the frequency of pulses on the input. The desired frequency is inputted into the **Parameter** field in **Hz**,



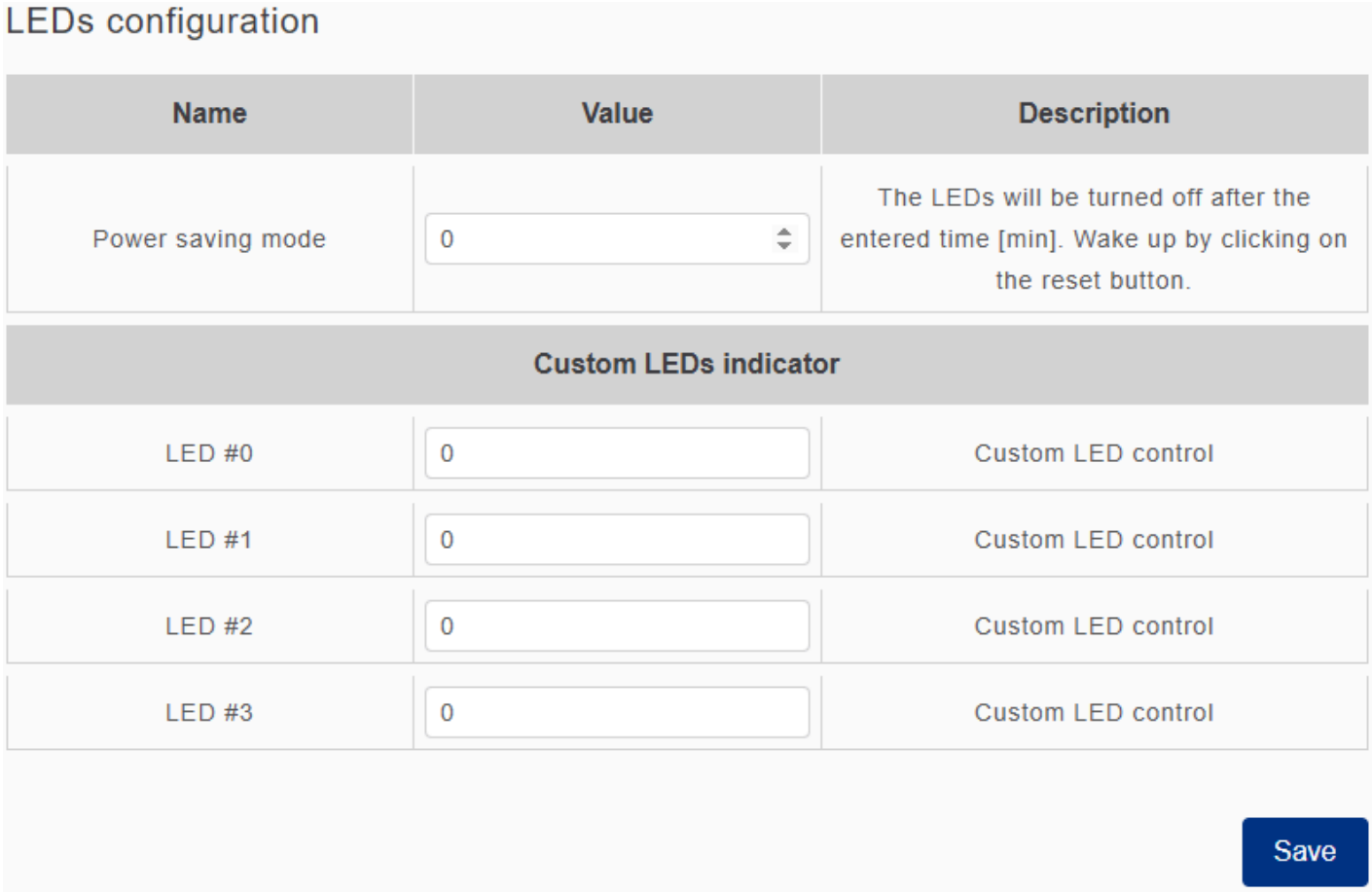


- **Parameter** – Value used in various action activation methods. When the cursor is placed over the input field, a parameter unit is displayed - eg. **Hz** for **Freq**,
- **Text ON** – Text dispayed on the LED display when the input is activated,
- **Text OFF** – Text dispayed on the LED display when the input is deactivated.

The **Go to the input actions** button located in the upper-right corner allows quick access to the **Action/Inputs** tab.

LED

LED signaling settings can be configured in this tab:



- **Power saving mode** – Parameter allows activation of power saving mode – after the defined time (in seconds) passes, the device's LED indicators are turned off. Clicking the **RESET** button activates the indicators,
- **LED #0 .. LED #3** – Parameter allows customization of additional LED's - INFO A/B/C/D to react in accordance to the preferences set.

> **Tip**
>
> **Apply changes using the Save button.**

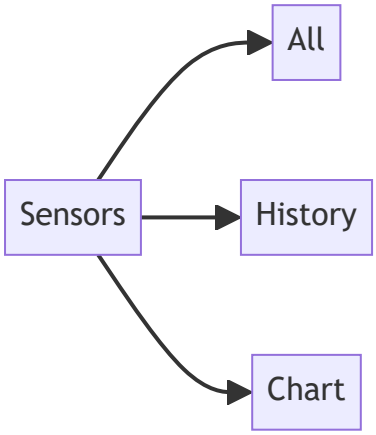
The following commands can be used to program the additional LED's (**x** indicates the input/output/sensor channel):

- `i[x]` – Input channel state,
- `o[x]` – Output channel coil state,
- `o[x].on` – Output channel state,
- `v[x]` – Virtual variable state,
- `s[x].aHi` – Sensor high level alarm,
- `s[x].aLo` – Sensor low level alarm,
- `s[x].wHi` – Sensor high level warning,

- `s[x].wLo` – Sensor low level warning,
- `s[x].err` – Sensor error,
- `s[x].ok` – Sensor operates correctly,
- `ping[x]` – Ping status: `0` - error, `1` - success,
- `poll[x].y` – Poller value,
- External device:
  - `dev[x].err` – Connection lost,
  - `dev[x].o[y]` – External output,
  - `dev[x].i[y]` – External input,
  - `dev[x].s[y].err` – External sensor error.
- Negation and logical operations (|/&/^):
  - `!s[0].err`
  - `!o[0]&v[1]`

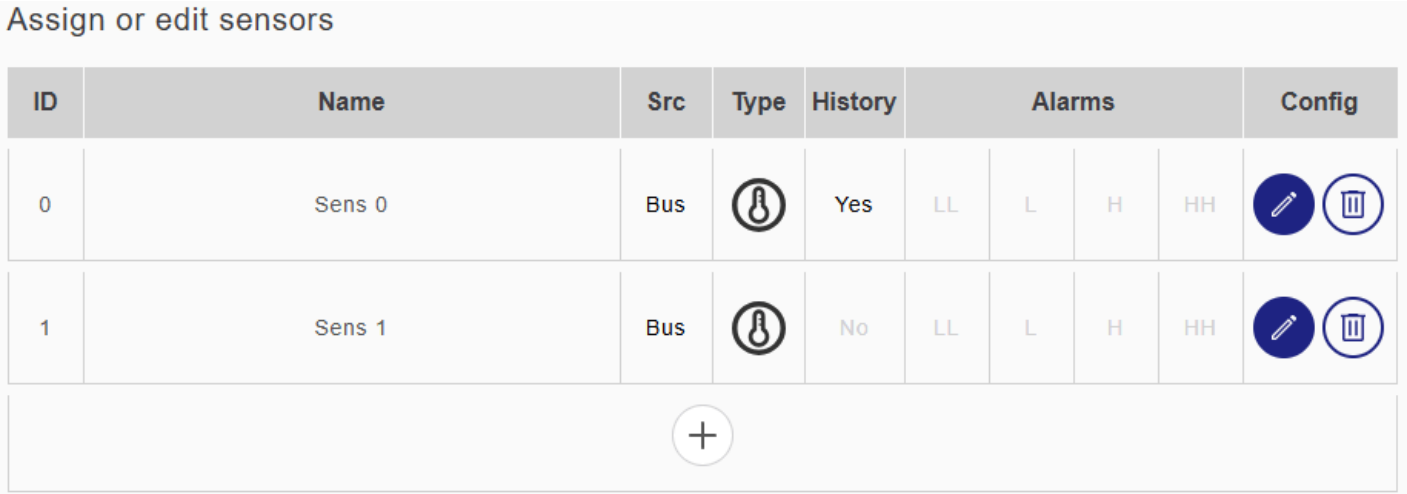
Sensors configuration

This tab allows the user to assign specific sensors to dedicated memory slots and to configure their parameters. It allows individual control over each sensor, setting specific parameters and modes of operation. Additionally, it enables the preview of historical readings, which can be downloaded in the form of JSON or CSV files.



All

Thanks to this tab, the user has full control over sensors configuration, readings correction and notifications. New sensors that integrate with the system automatically can be added. The user can define parameters for each sensor, customizing them to own needs and operation conditions. Additionally, this tab shares the tools to edit existing sensors, allowing them to be constantly adaptable to the changing conditions or user needs.



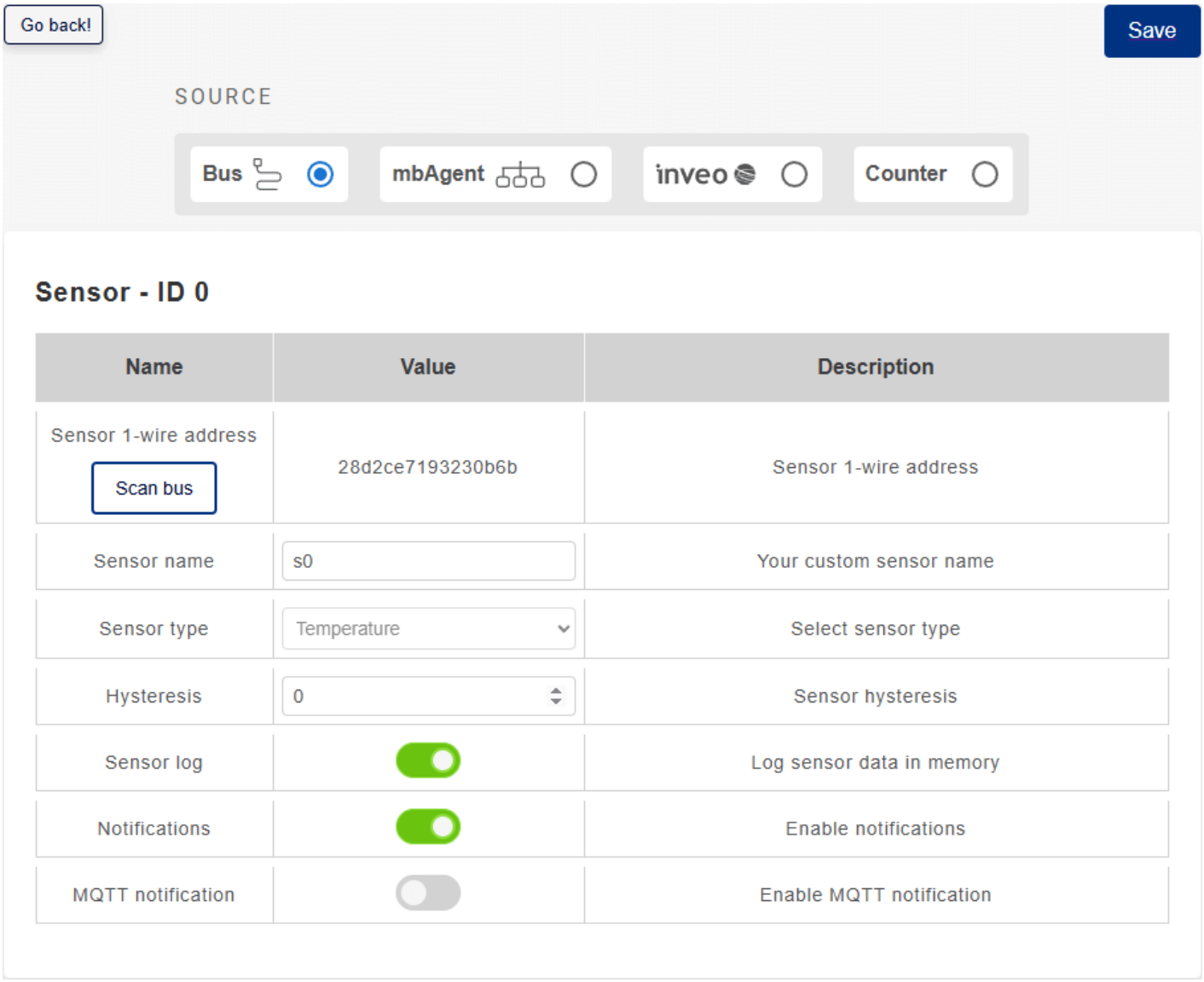
Each sensor table column contains the following information:

- **ID** – Sensor identification number,
- **Name** – Sensor name,
- **Src** – Source, from which the sensor readings are taken (1-Wire or counter),
- **Type** – Connected sensor type:
  - – Temperature sensor,

-  – Humidity sensor,
-  – Input,
-  – Pressure sensor,
-  – Analogue 0-10V DC voltage sensor,
-  – Analogue 4-20mA current sensor,
-  – Carbon monoxide presence sensor (CO),
-  – Carbon dioxide presence sensor (CO2).

- **Log** – Informs if the sensor has an active (Yes) or inactive (No) readout logging function,
- **Alarms** – Activated warnings/alarms:
  - **LL** – Lower alarm threshold,
  - **L** – Lower warning threshold,
  - **H** – Upper warning threshold,
  - **HH** – Upper alarm threshold.
- **Config** – Sensor configuration buttons:  – Edit sensor parameters,  – Delete sensor.

The following dialogue window is displayed afer clicking the edit button:



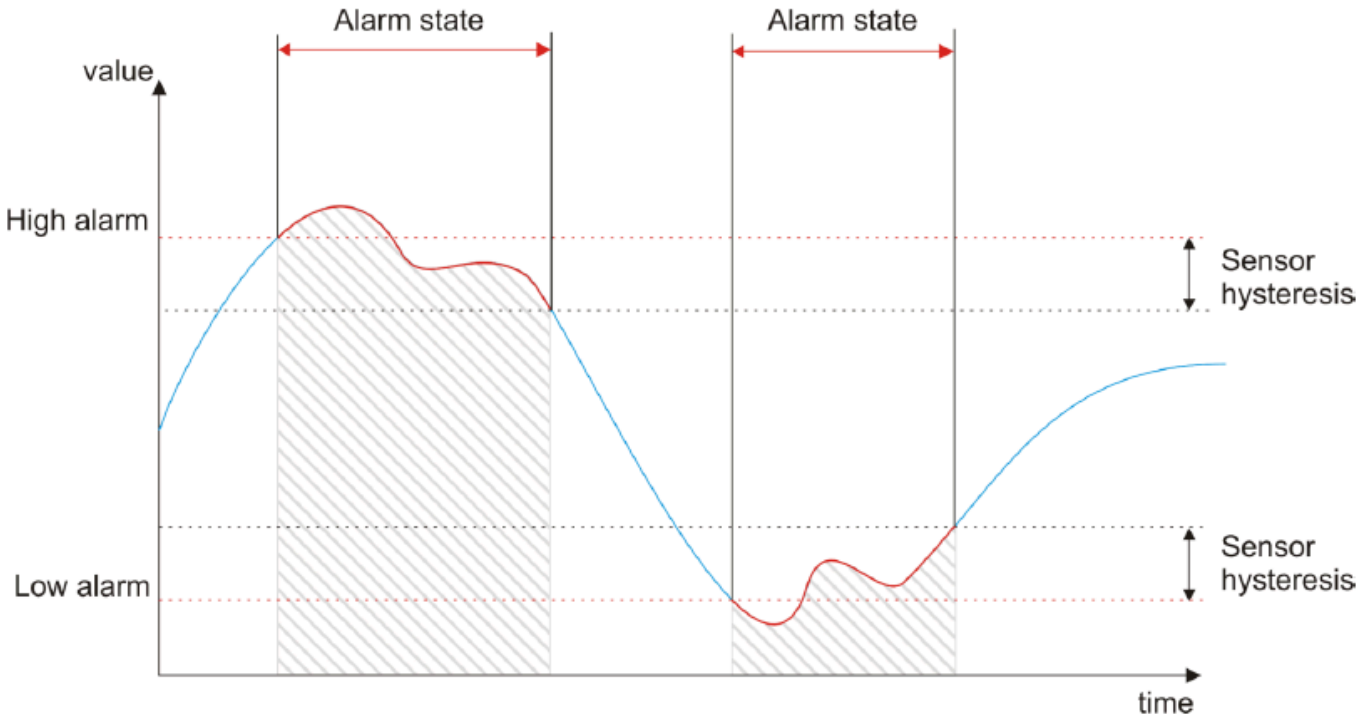
- **Source** – Sensor source *(version dependent)*:
  - **Bus** – Sensor bus,
  - **mbAgent** – Data read off Modbus,
  - **Inveo** – Data received from another device via iCluster,
  - **Counter** – Counter input.
- **Sensor bus address** – **Scan bus** makes it possible to locate and assign a sensor connected to the device,
- **Sensor name**,
- **Sensor type**:
  - **Temperature**,
  - **Humidity**,
  - **Input**,
  - **Current**,
  - **Pressure**,

- **Voltage**,
- **CO** – Carbon monoxide,
- **CO2** – Carbon dioxide.

- **Hysteresis** – Sensor hysteresis (inactive if the sensor type is set to **Input**) – applies to the warning and alarm states. The hysteresis defines the maximum acceptable difference between the warning/alarm threshold and normalization value.

Example

High warning is set to 30 degrees, hysteresis is 2 degrees. After the sensor reaches 30°C, the device will reach the alarm state that will presist until the sensor value drops to 28°C (30-2=28). Hysteresis is the offset between warning/alarm activation or deactivation, preventing the alarm states from being triggered too often during small measurement fluctuations.

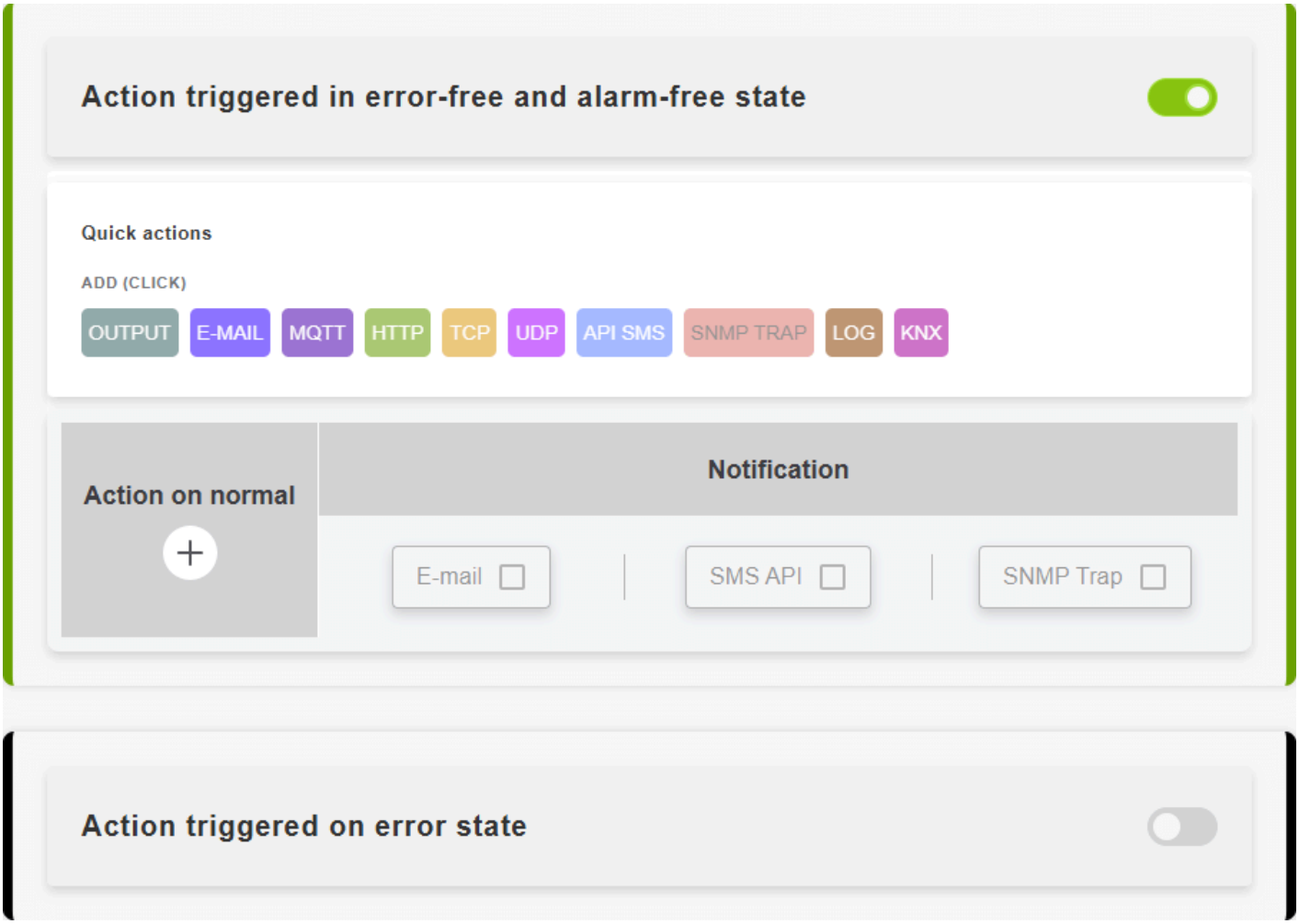


- **Channel** – Channel selection – active only if the sensor type is set to **Input**,
- **Sensor log** – Enable/disable sensor data logging to device memory *(version dependent)*,
- **On display** – Enable/disable display of measured value on the LED display *(IQIO series only)*,
- **Notifications** – Enable/disable notifications,
  - **MQTT notification** – Enable/disable MQTT notifications.

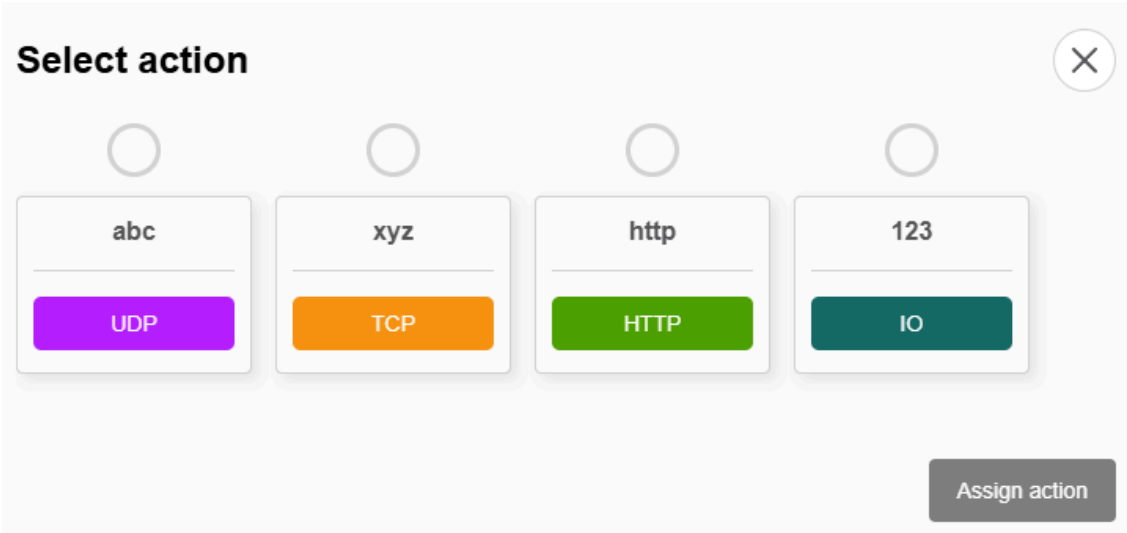
Enabling **Notifications** allows to change the device's reaction to:

- Transition of sensor to an alarm- and error-free state,
- Transition of sensor to the error state.

The user can select the type of a notification that will be sent in response to the above events. Appropriate settings in the **Services** tab must be configured first for the notifications to be sent.

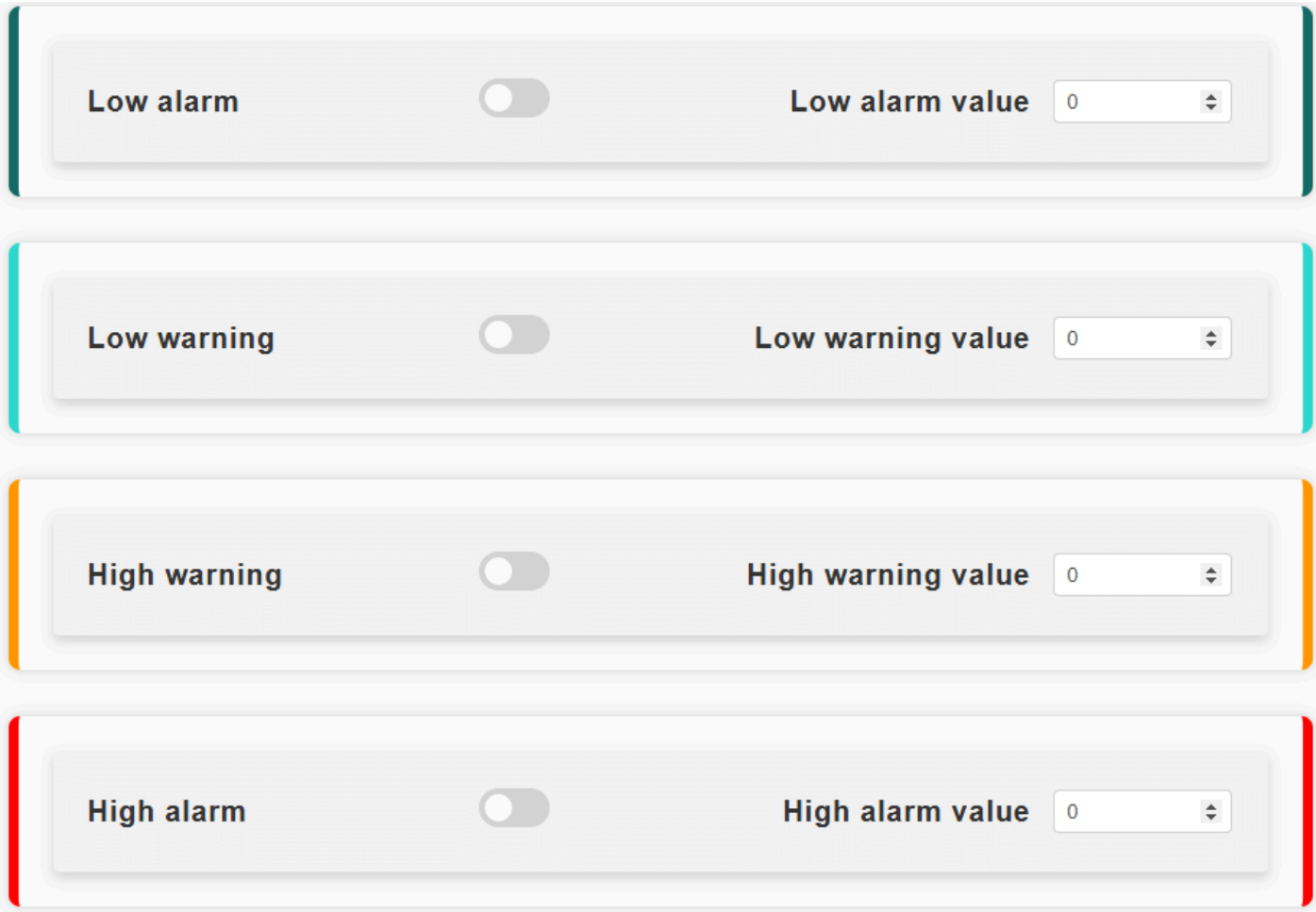


To assign an action, click **+**. A dialogue window will be displayed, where the desired action (defined previously in **All**) can be selected.



ALARMS CONFIGURATION

In the settings section dedicated to sensor alarm configuration, the user gains full control over customization of alarm parameters to his own needs. This allows configuration of alerts and reactions to important sensor operation events.



- **Low alarm** – Enable low level alarm,
  - **Low alarm value** – Value that will make the sensor reach the state of low level alarm.
- **Low warning** – Enable low level warning,
  - **Low warning value** – Value that will make the sensor reach the state of low level warning.
- **High warning** – Enable high level warning,
  - **High warning value** – Value that will make the sensor reach the state of high level warning.
- **High alarm** – Enable high level alarm,
  - **High alarm value** – Value that will make the sensor reach the state of high level alarm.

After the alarm is activated, additional window that allows customization of device reaction for alarm situation appears. Here, the user can configure notifications and assign specific actions.

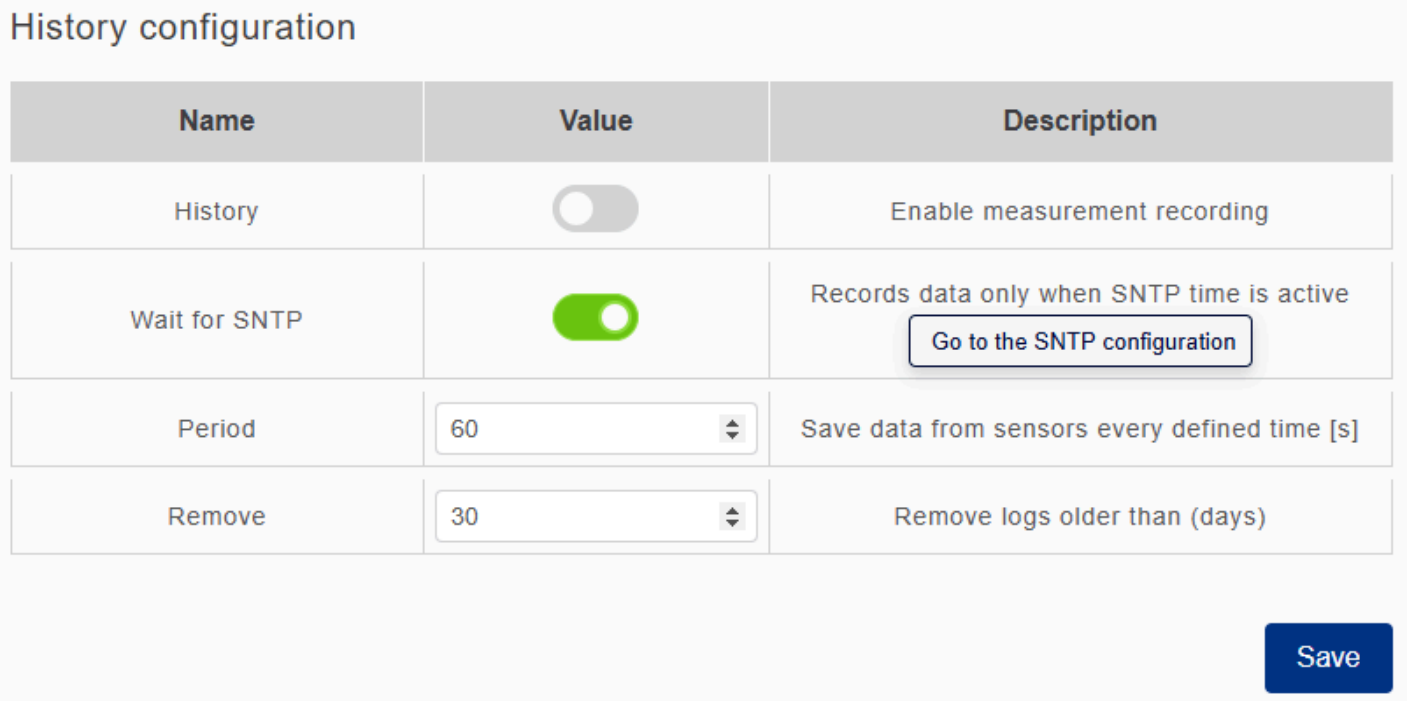
Sensor corrections final = a * (x + preoffset) + b		
Sensor preoffset	0	Sensor preoffset correction
Sensor multiplication: a	1	Multiplying sensor value
Sensor offset: b	0	Fixed correction value

- **Sensor preoffset** – Sensor correction field, according to the linear function formula f(x)=ax*b,

- **Sensor multiplication 'a'**,
- **Sensor offset 'b'** – Constant value correction.

History

This tab allows for activation and configuration of historical sensor readout logging:

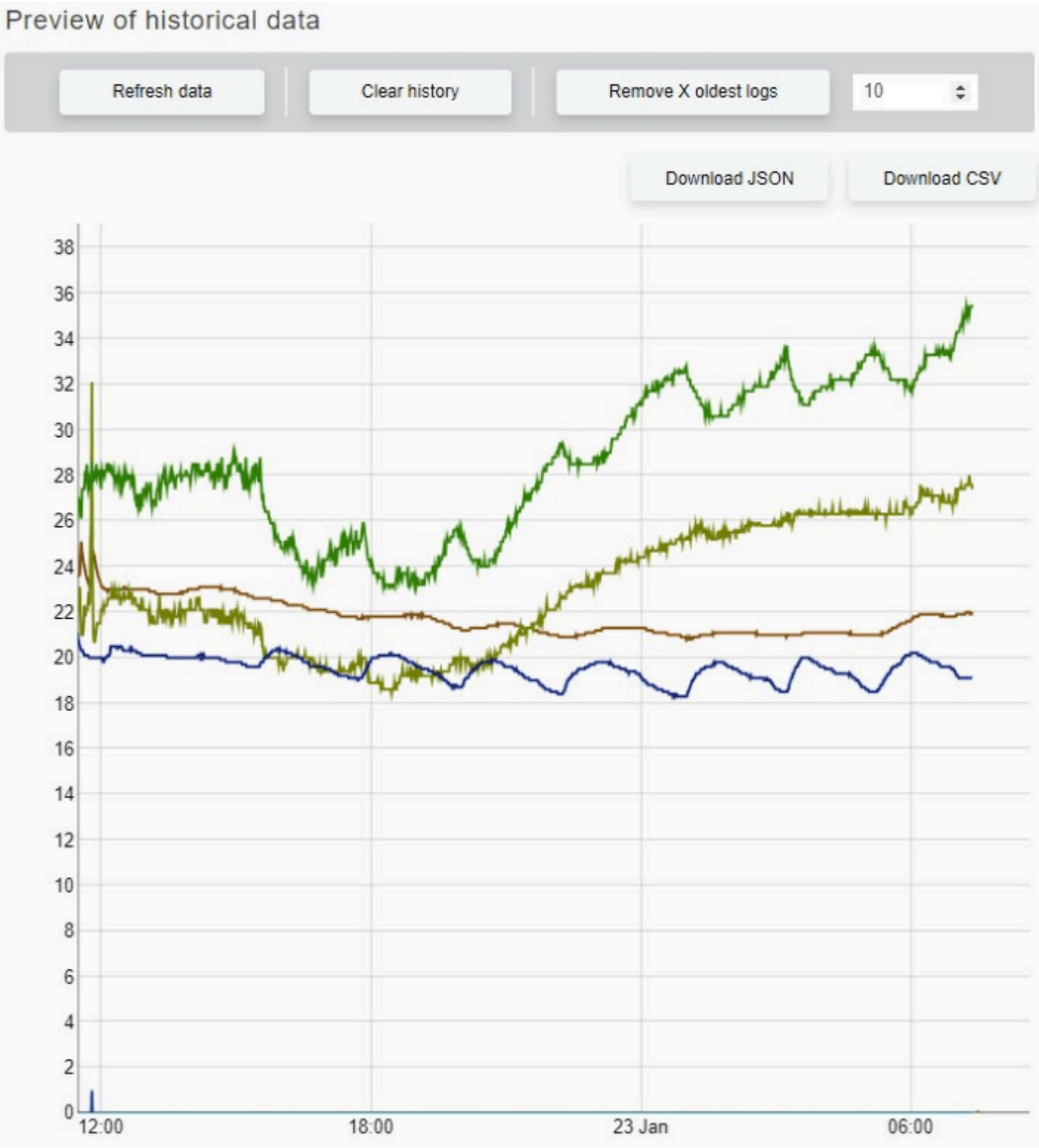


- **History** – Enable/disable sensor readout logging,
- **Wait for SNTP** – When enabled, data from the sensors will be logged only when the device downloads time from the SNTP server after a restart,
- **Go to SNTP configuration button** – Enables quick access to the SNTP configuration tab,
- **Remove** – Allows clearing memory of records older than the specified number of days.

> **Tip**
>
> **In order to log the sensor data into the device's memory, the Sensor log function must be activated in Sensors / All (available in sensor configuration mode).**

Chart

In this tab, all sensors (with **Sensor log** enabled in **Sensors / All**) readings are presented on graphs.



- **Refresh data**,
- **Download JSON**,
- **Download CSV**,
- **Remove X oldest logs** – Remove x oldest sensor readings from memory,
- **Clear history** – Remove all readings from device memory.

Sensor handling

To make sure that the sensor readings are precise and reliable, it is recommended to follow the configuration steps below:

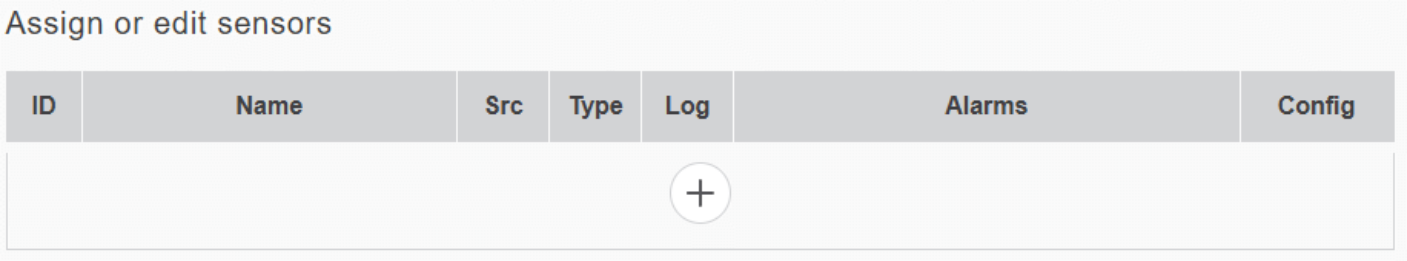
Sensor assignment

First, connect the sensor to the device – consult **Sensor connection diagram**.

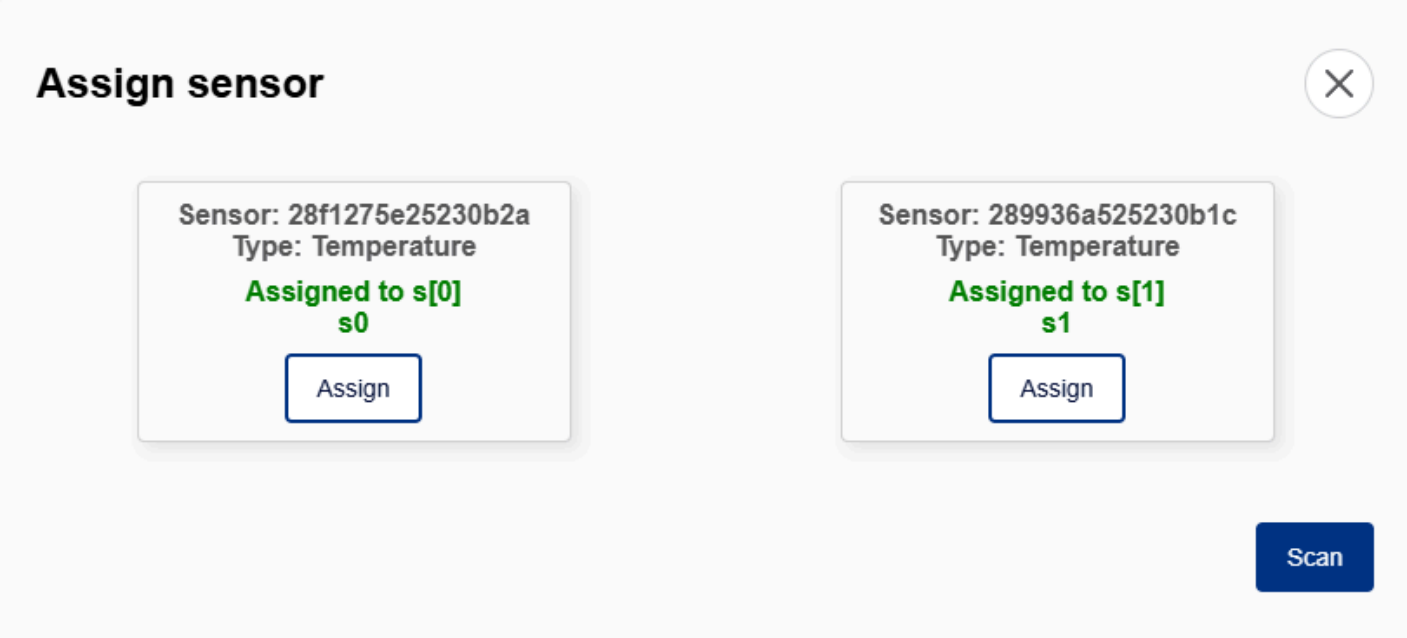
After that, using the device's webpage, locate and configure the sensor's basic parameters. If no sensors were connected to the device previously, the user can use the automatic sensor assignment option. Reset the device after the sensors have been connected – after the device restarts, it will automatically recognize and assign available sensors, specifying their type too.

Manual sensor configurations steps are described below.

Step 1: In **Sensors / All**, click **+**:



In the displayed dialogue window, select the sensor **Source** first. In this case, select **One-wire** (sensor physically connected to the device's bus).
Sensor assignment should begin with clicking **Scan bus**, that will display a window with detected sensors:



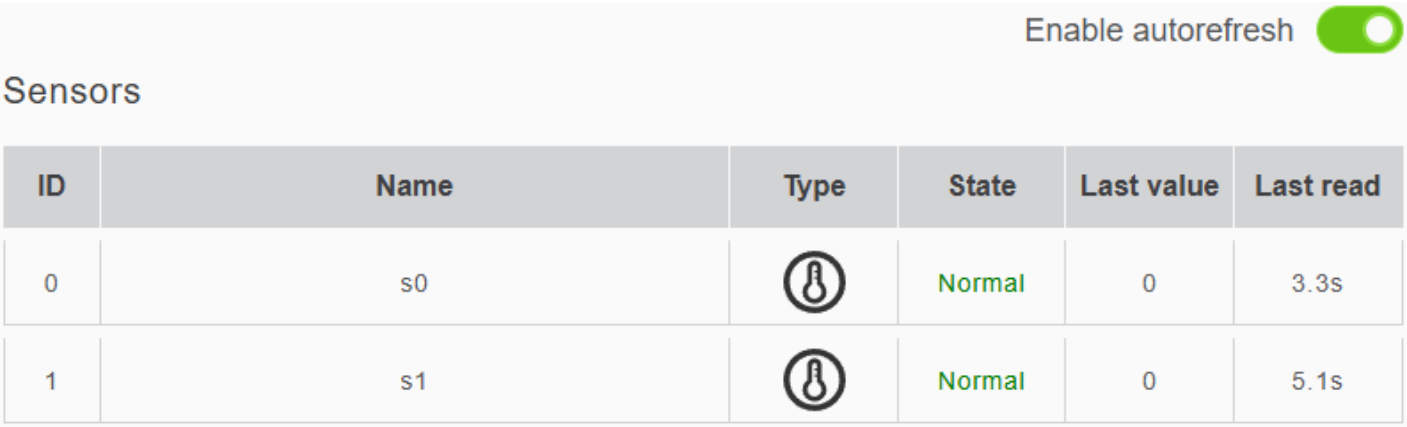
Assign the selected sensor.

Step 2: Basic parameters configuration:

In the sensor's configuration window, set the correct sensor type, assign a name etc. All settings should be applied using the **Save** button.

Step 3: Status tab preview:

A sensor that is configured correctly will display its readings in **Status / Sensors**:



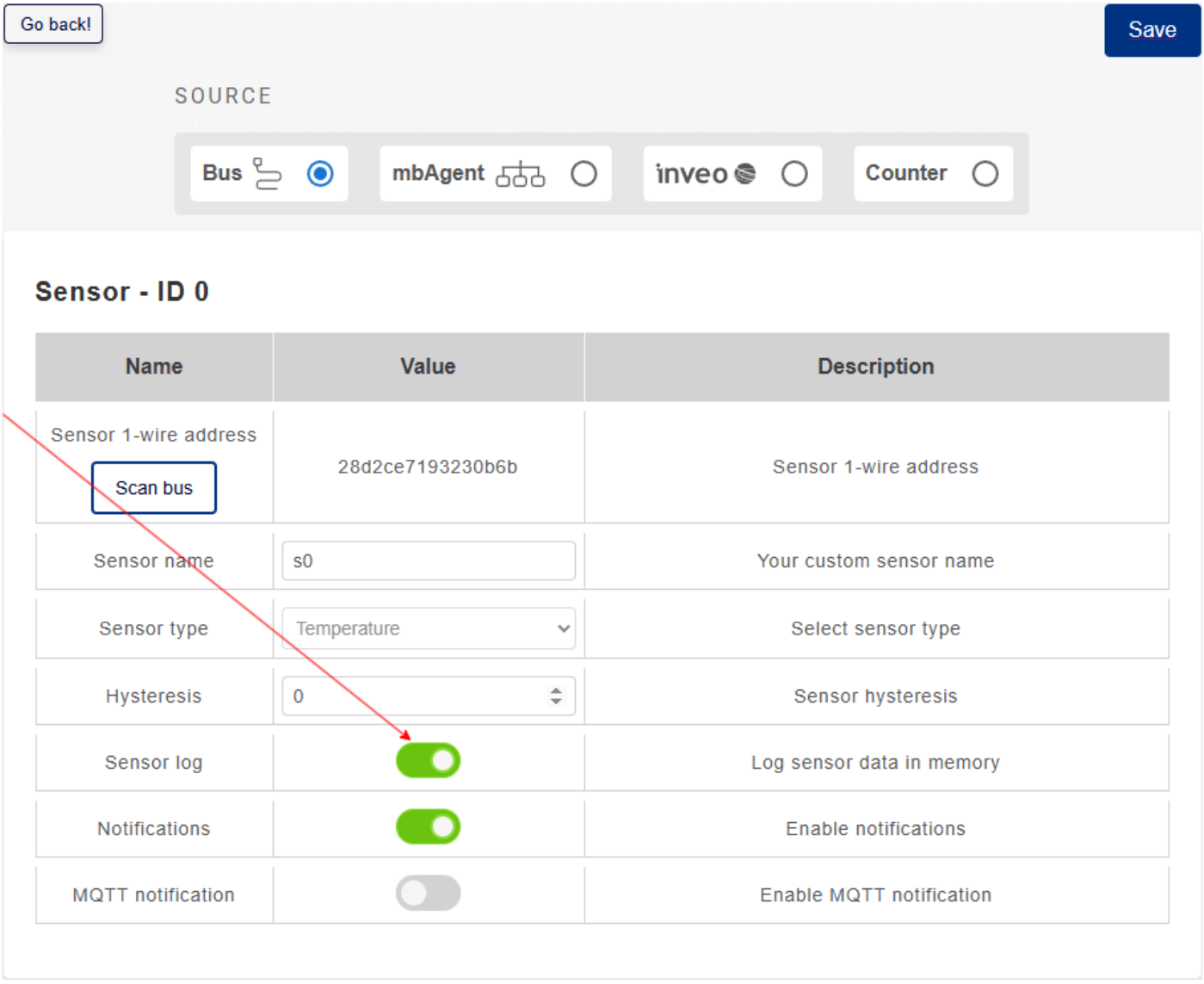
SENSOR READINGS LOGGING AND GRAPH PREVIEW

Conducting the configuration as per above instructions does not guarantee automatic sensor readings logging to the device memory. It only allows current value preview. To log the sensor data periodically, follow those directions:

Step 1: Enable automatic sensor readings logging:

In **Sensors / All**, click 

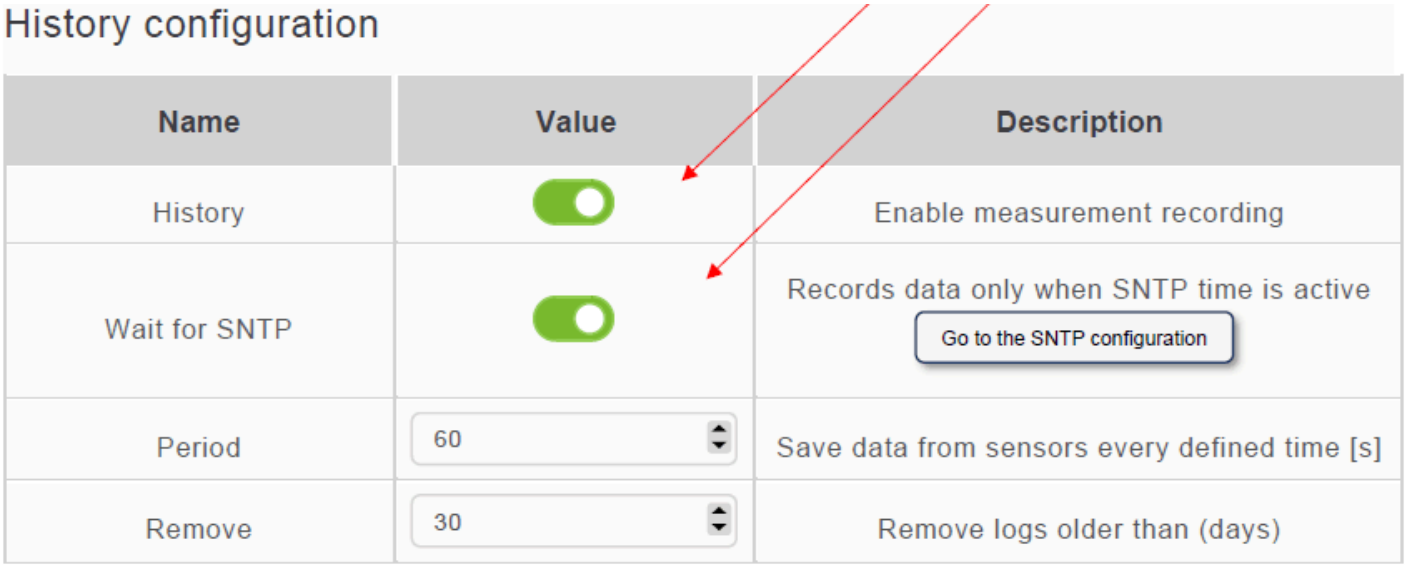
A sensor configuration window will be displayed:



Go to **Sensor log** and **Enable** it. All settings should be applied using the **Save** button.

Step 2: Enable automatic reading logging:

Sensors / History tab:

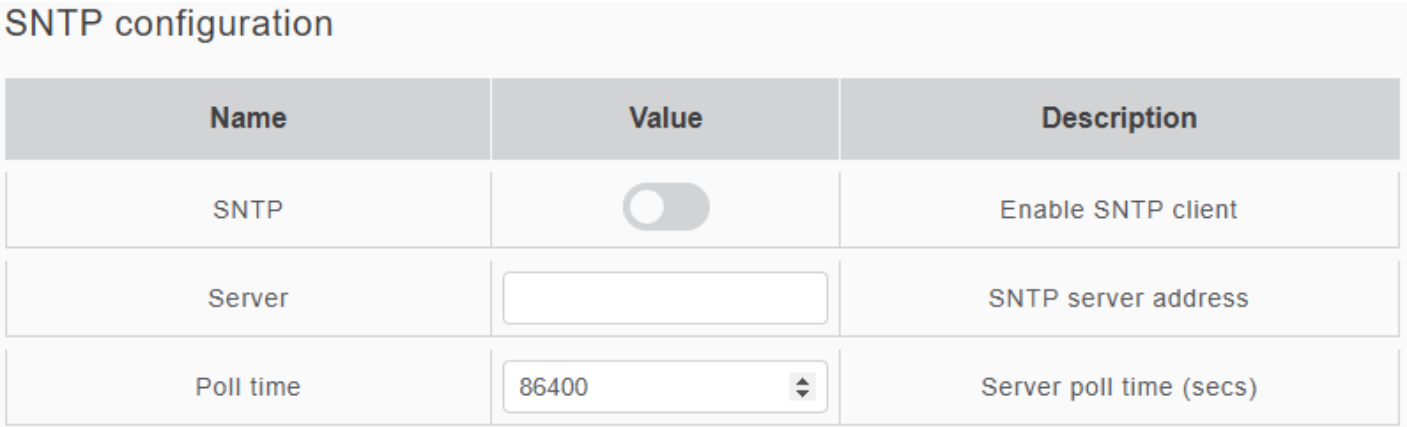


Go to **History** and select it using the slider.
After the device restarts or an Internet connection error occurs, it will automatically switch to logging using the built-in Real Time Clock (RTC). **Wait for SNTP** allows the device to wait for SNTP time before it begins logging data. Thanks to this, reading time precision is provided, even when no Internet connection is available.

> **Tip**
>
> **After making changes, apply them using the Save button.**

Step 3: Set the current time:

The device is equipped with an internal RTC with battery backup. If the device has constant Internet access, **SNTP** can be enabled.
Administration / Time - SNTP allows for SNTP configuration – guaranteeing that the current time will be kept even when no power is connected to the device.



> **Tip**
>
> **After making changes, apply them using the Save button.**

Once the unit has been configured according to the above instructions, the device will start saving sensor data to internal memory. The module also allows preview of graph of sensor readings over time.

DOWNLOAD OF STORED SENSOR READINGS

Saved sensor readings can be downloaded in the form of JSON or CSV files by recalling the internal memory resource. To do this, use the following command: `http://<IP_address>/data/log.json`.

Sensor readings can also be downloaded from the device's webpage. To do that, in **Sensors / Chart**, click the appropriate button: **Download JSON** or **Download CSV**.

Notifications configuration

This tab enables configuration of various notifications – activation, deactivation and notifications assignment involving E-mail, SMS, SNMP Trap, MQTT, related to sensors, inputs and outputs *(version dependent)*.

For the notifications to be effective:

Step 1: Enable notifications in the selected system elements tab: sensors, inputs or outputs, define the type of notifications,

Step 2: Depending on the notification type selected - SMS, e-mail, SNMP Trap, MQTT – configure the settings in **Services** – consult **Network services**

Step 3: Enable notifications in the **Configuration** tab.

Sensors

In this tab, notifications settings regarding operation of particular sensors can be configured. Notifications for the selected sensor can be enabled in two ways: in the **Sensors** tab during sensor configuration or by clicking an icon in **Notifications / Sensors**.

The configuration window that appears after notifications are activated, allows for full customization:

s0 (s0)				
State	E-mail	SMS	SNMP Trap	MQTT
Info	☐	☐		
OK	☐	☐	☐	
Error	☐	☐	☐	
Alarm low (disabled)	☐	☐	☐	☐
Warning low (disabled)	☐	☐	☐	
Warning high (disabled)	☐	☐	☐	
Alarm high (disabled)	☐	☐	☐	

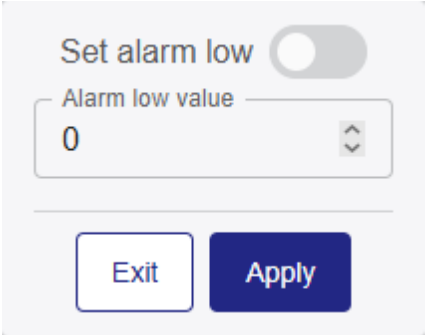
Tip

In order for the E-mail, SMS, SNMP Trap and MQTT notifications to operate correctly, configure them in the Services tab – consult Network services.

In the table, the user has the ability to select the type of notifications that are to be sent in response to defined events:

- **Info** – Periodic sensor state information,
- **OK** – Sensor normalization state after a previous error or alarm state,
- **Error** – Sensor error state,
- **Alarm low** – Low level alarm state,
- **Warning low** – Low level warning state,
- **Warning high** – High level warning state,
- **Alarm high** – High level alarm state.

If no alarm values have been previously defined, it can be done using the icon which opens a window to input the required value:



After inputting the required value, enable the function by clicking .
The settings will also be visible in the **Sensors** tab.

> **Tip**
>
> **To activate the notifications function, it is important to enable it in the Configuration tab, in addition to the settings configured here.**

Inputs

In this tab, notifications settings regarding operation of device inputs can be configured. After clicking , notifications for the selected input will be activated, leaving only their details to be configured. The configuration window that appears after the notifications have been activated, enables full customization.

DI 0 (DI0)				
State	E-mail	SMS	SNMP Trap	MQTT
On change action	☐	☐	☐	☐
Info	☐	☐		☐

It's possible to activate MQTT and SMS, SNMP Trap and notifications.

> **Tip**
>
> **In order for the E-mail, SMS, SNMP Trap and MQTT notifications to operate correctly, configure them in the Services tab – consult Network services.**

In the table, the user can select the type of notifications that are to be sent in response to defined events:

- **On change action** – Input state change (activated – deactivated),
- **Info** – Input state information.

> **Tip**
>
> **To activate the notifications function, it is important to enable it in the Configuration tab, in addition to the settings configured here.**

Outputs

In this tab, notifications settings regarding operation of devices connected to the device's outputs can be configured.

After clicking , notifications for the selected output will be activated, leaving only their details to be configured. The configuration window that appears after the notifications have been activated, enables full customization.

DO 0 (DO0)				
State	E-mail	SMS	SNMP Trap	MQTT
On change action	☐	☐	☐	☐
Info	☐	☐		☐

It's possible to activate E-mail, SMS, SNMP Trap and MQTT notifications.

> **Tip**
>
> **In order for the E-mail, SMS, SNMP Trap and MQTT notifications to operate correctly, configure them in the** Services **tab - consult** Network services**.**

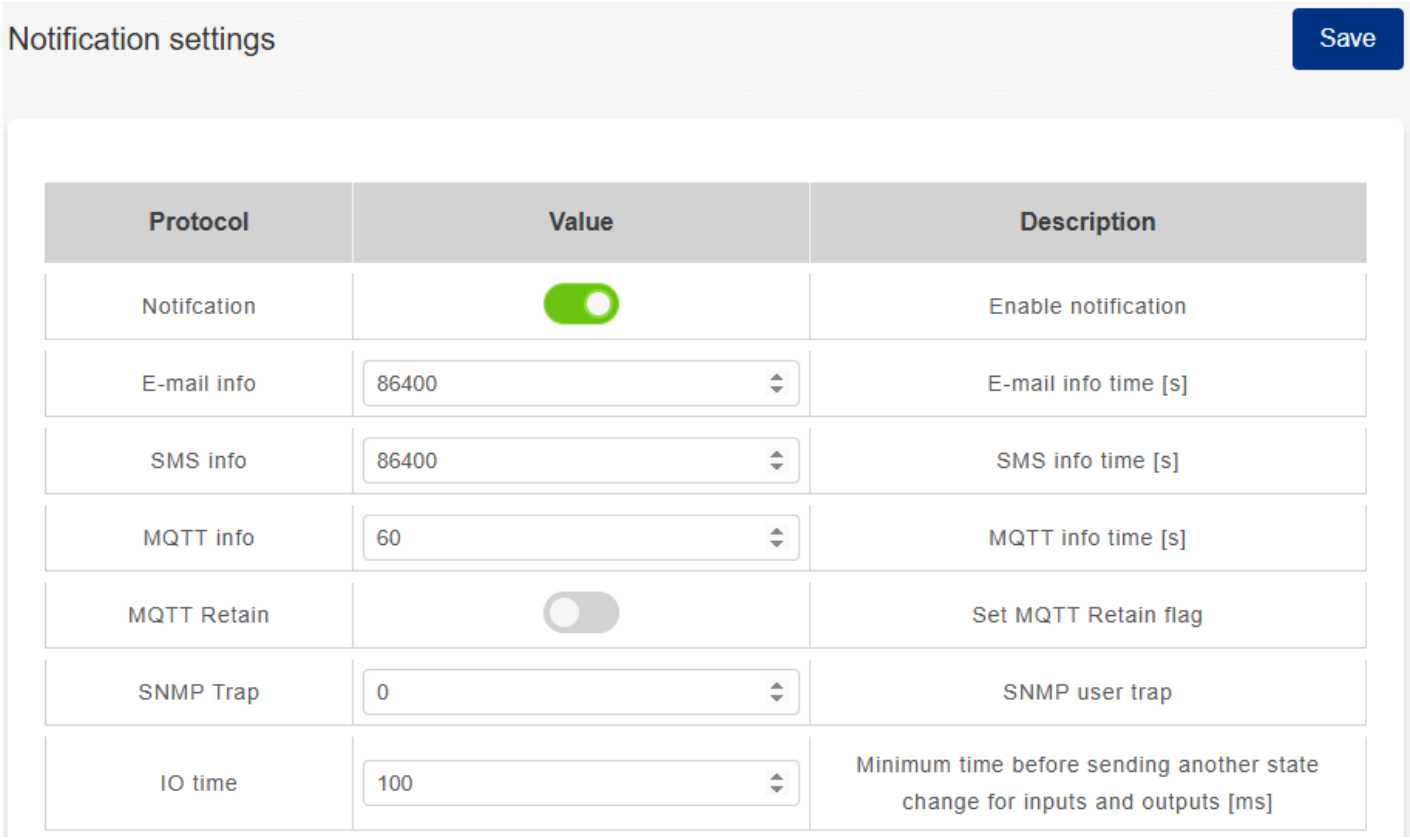
In the table, the user has the ability to select the type of notifications that are to be sent in response to defined events:

- **On change action** – Output state change (activated – deactivated),
- **Info** – Output state information.

> **Tip**
>
> **To activate the notifications function, it is important to enable it in the** Configuration **tab, in addition to the settings configured here.**

Configuration

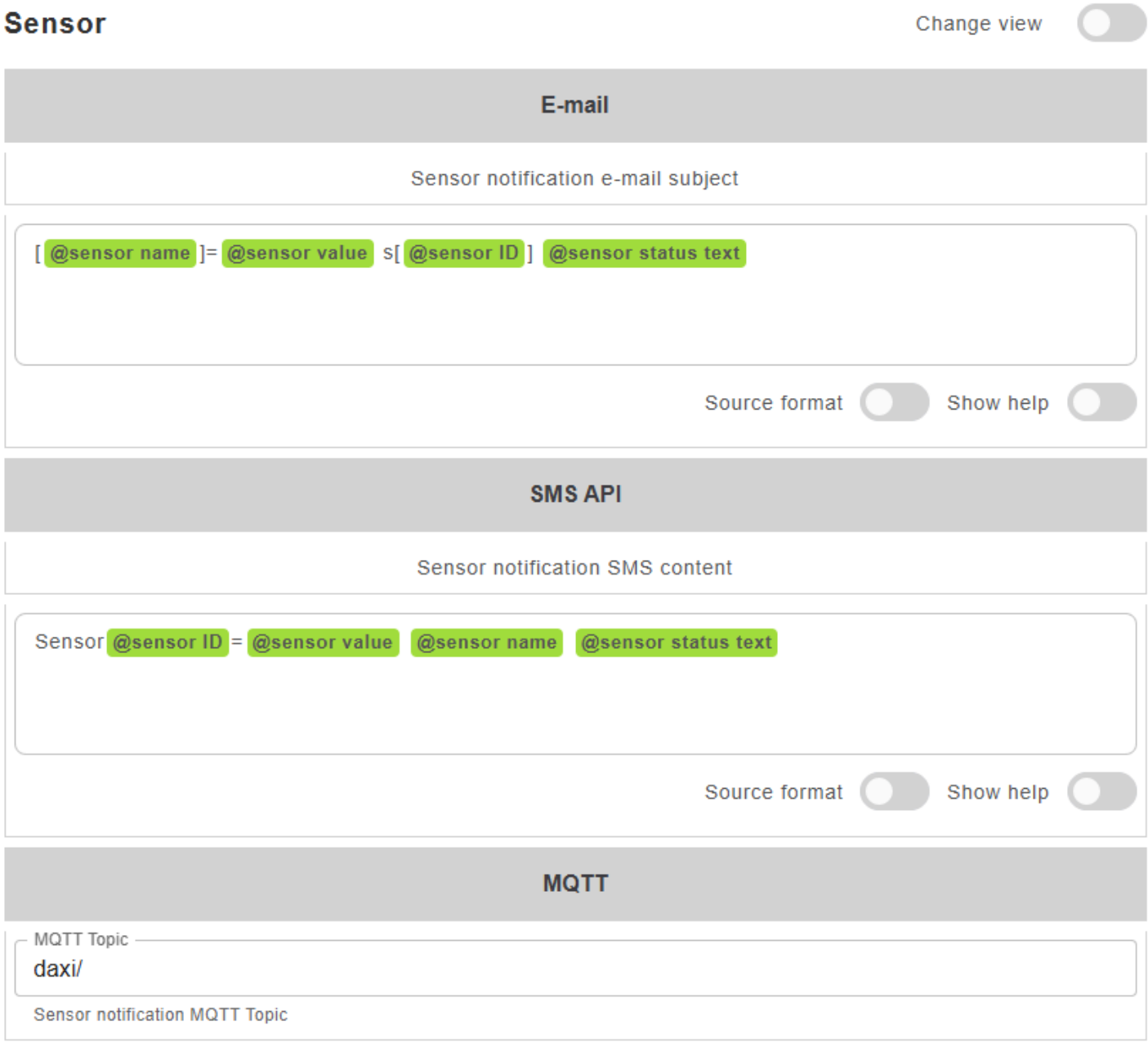
In this section, there is an option to activate the notification functions required to send notifications. Additionally, the user has the ability to adjust general parameters related to notifications.



- **Notification** – Enable/disable notifications,
- **E-mail info** – Frequency of e-mail messages containing the sensor/input/output state sending *(only modules with SMTP support)*,

- **SMS info** – Frequency of SMS messages containing the sensor/input/output state sending *(only modules with SMS API support)*,
- **MQTT info** – Frequency of MQTT messages containing the sensor/input/output state sending,
- **MQTT Retain** – Activated option causes the broker to retain last messages for topics the device sends data to,
- **SNMP Trap** – Selected SNMP Trap,
- **IO time** – Minimum time that must pass between subsequent input/output state changes to prevent multiple notifications to be sent, especially during testing or experiments with device's inputs/outputs.

The tables **Sensor**, **Input** and **Output** are located in this tab.
Each table contains predefined commands that allow sending of e-mail and SMS messages containing the current device states. Additionally, the user has the ability to edit those commands, which makes customization to own preferences possible, eg. by assigning a name to the device. Each table contains a topic, use of which is required if using MQTT to send notifications.



Input

Change view

E-mail

Input notification e-mail subject

[@input name]= @input state i[@input ID]

Source format Show help

SMS API

Input notification SMS content

Input @input ID = @input state @input name

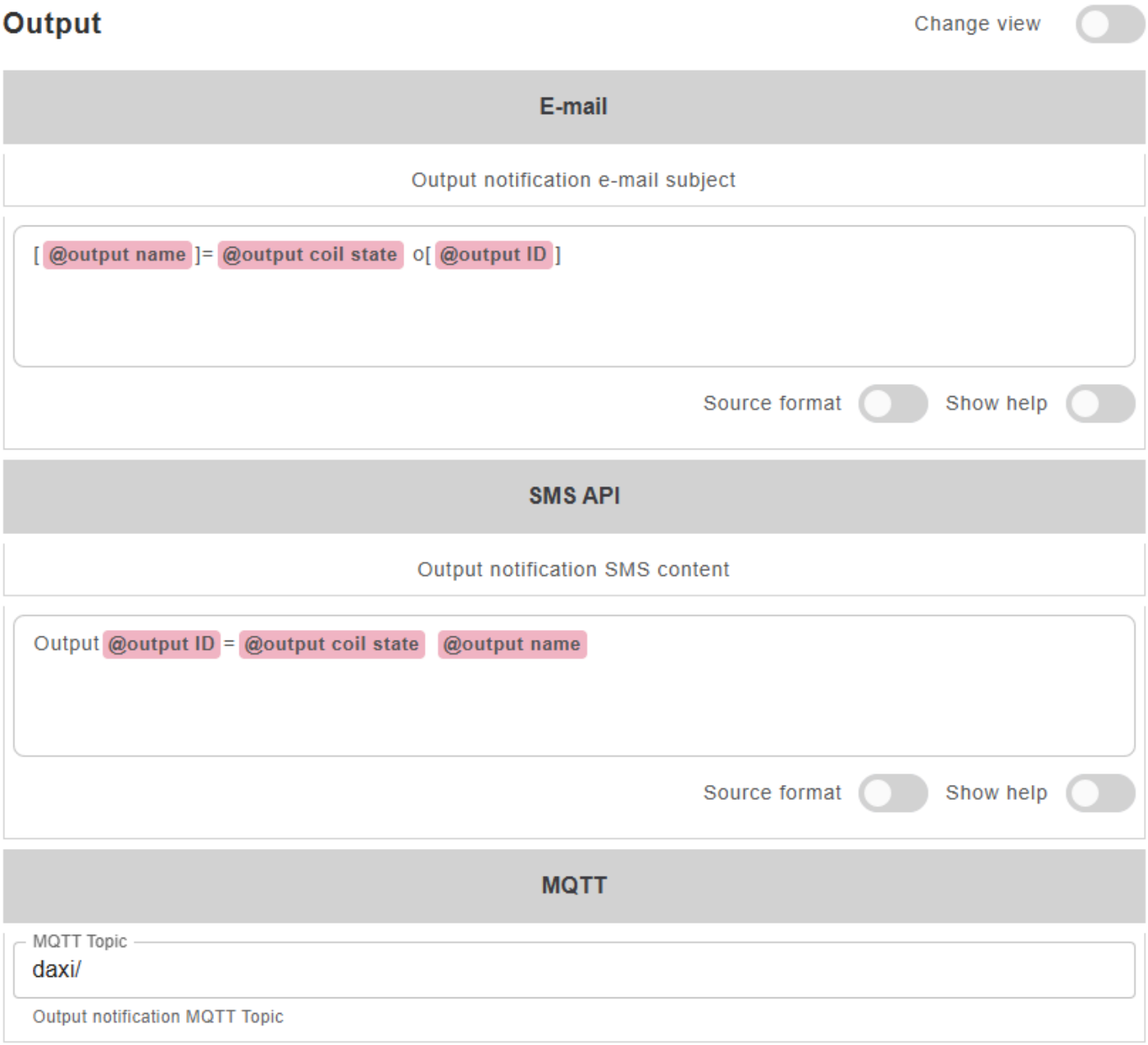
Source format Show help

MQTT

MQTT Topic

daxi/

Input notification MQTT Topic

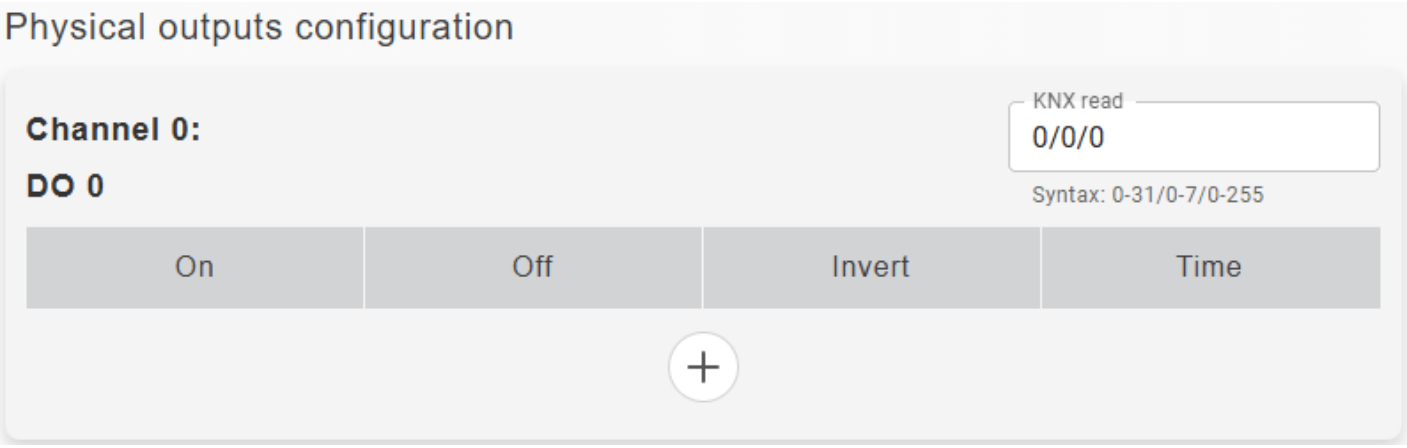


Work matrix

KNX outputs

In the **Work matrix/KNX outputs** tab, it's possible to configure the behaviour of an output in response to data frames sent to specified KNX groups.

When Daxi receives a data frame on a specified KNX group (eg. 1/1/1), it will automatically execute the defined actions such as output activation, deactivation, state inversion or momentary deactivation. Thanks to this, the user can precisely control the device's behaviour depending on the KNX signals.



Clicking **Advanced configuration** will change the configuration view, which makes copying the command path easy:

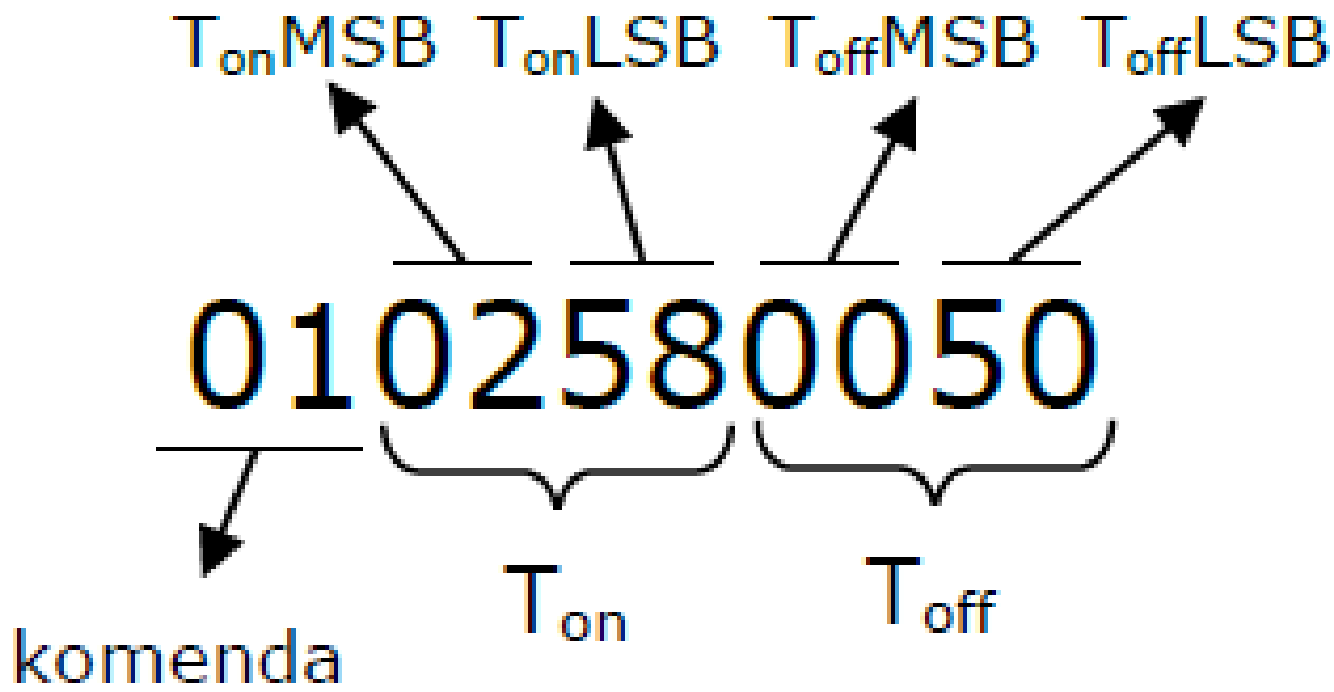


KNX commands recognized by Daxi:

- **On** – Output activation – can assume 2 values:
  - 0 – Sending this command will deactivate the output,
  - 1 – Sending this command will activate the output.
- **Off** – Output deactivation – can assume 2 values:
  - 0 – Sending this command will activate the output,
  - 1 – Sending this command will deactivate the output.
- **Invert** – Output state inversion:
  - 0 – This command will not affect the output state,
  - 1 – Sending this command will invert the output state.
- **Time** – Timed output activation/deactivation:
  - 00 – Relay deactivation,
  - 01 TonMSB TonLSB ToffMSB ToffLSB – Relay activation for the defined time after the specified delay. The time is defined in 0,1 second units stored in HEX.

Example

Activate the relay for 60 seconds (value of 600) after a 8 second delay (value of 80): - dec 600 hex 0x02 0x58 - dec 80 hex 0x00 0x50 Syntax:

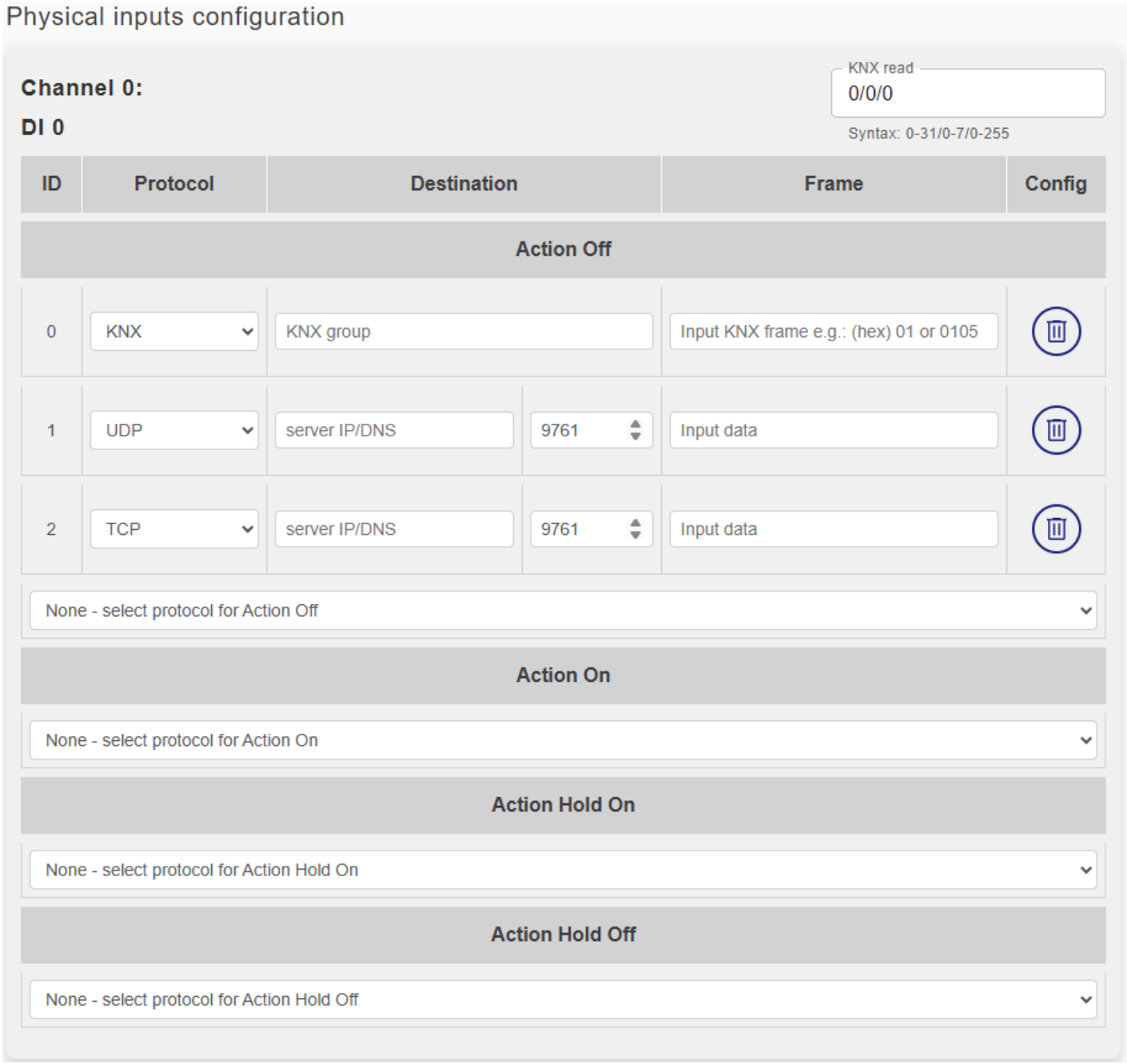


Tip

To program relay activation without a delay, don't enter the Toff value.

KNX/TCP/UDP Inputs

This tab allows the user to program the device's reaction to a signal received on a selected input. To achieve this, KNX, UDP, and TCP protocols can be used. Multiple reactions can be assigned to a single input.



After selecting the protocol, input additional information depending on the selected setting:

KNX:

- **KNX group** – Target KNX group, eg. 1/1/1,
- **Frame** – Syntax of the command sent to the specified KNX group in hexadecimal (HEX) format. Example syntax: 0105.

TCP/UDP:

- **Server IP/DNS** – Target IP address for the TCP/UDP frame,
- **Port** – Default 9761,

- **Frame** – TCP/UDP frame contents. For example `input %i[0]% triggered`.

Clicking the icon deletes the action.

The **Go to the inputs configuration** button located in the upper-right corner allows quick access to **IO Settings/Inputs**.

Binding

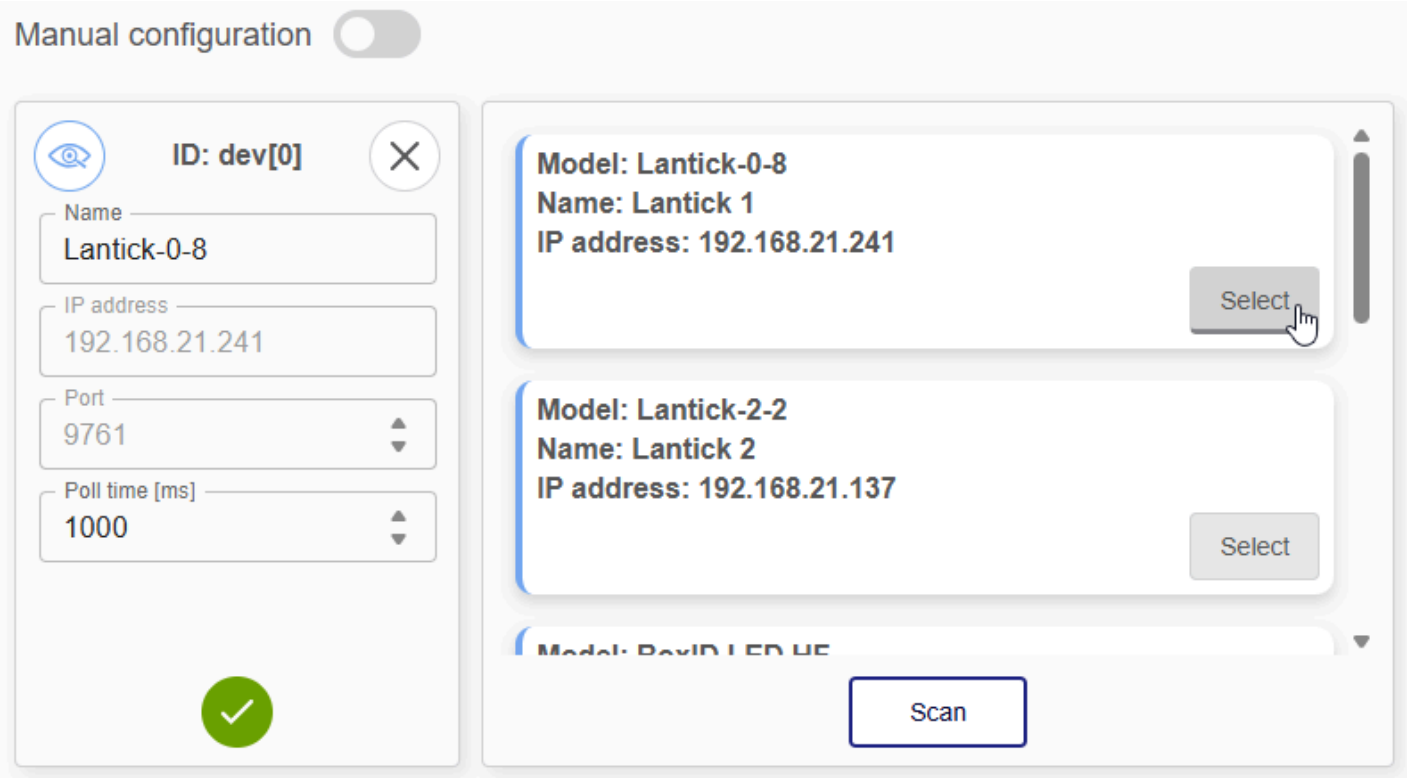
This tab enables configuration of relations between inputs and outputs.

iCluster

iCluster allows multiple Inveo devices to interconnect within a single LAN. This enables preview of device state, measured values, and output control. It is also possible to transfer the input state of an external device to the selected Daxi output. The following Inveo modules can be interfaced with Daxi:

- IQIO series,
- Nano Relay Output,
- Nano Digital Input,
- Nano Temperature Sensor,
- Hero Web Sensor,
- Lantick series.

To connect Daxi with another Inveo module, click - a dialogue window will appear with the option to find the device in a LAN. After clicking **Scan**, Daxi will search for all available devices:



After **Selecting** the device, apply the settings using .

By clicking the  icon, details of the connected devices can be viewed, output control tested, external sensor redout routed to Daxi etc.

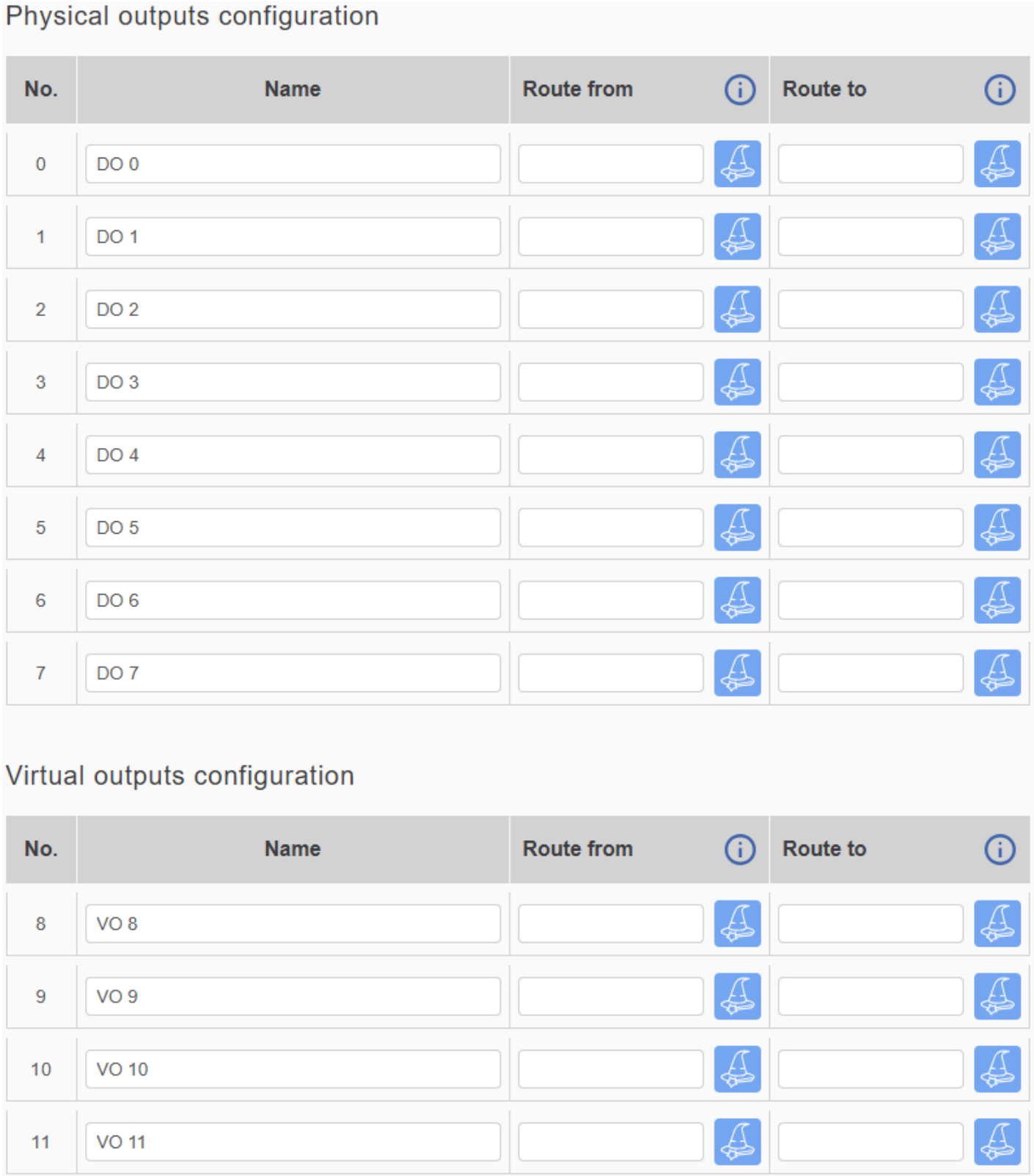
LANTICK MODULES CONFIGURATION

In order to configure Lantick series modules to cooperate with Daxi, select **Enable Program Access** in the **Administration** tab.

Both modules must be located within the same subnet.

Outputs

This tab allows routing of state from other components:



- **Name**,
- **Route from** – Source that will have influence over the output state – change of state of the channel in this field will cause the output state to change accordingly.

Available output state sources:

- `i[x]` – Input channel state,
- `i[x].act` – Last input action,

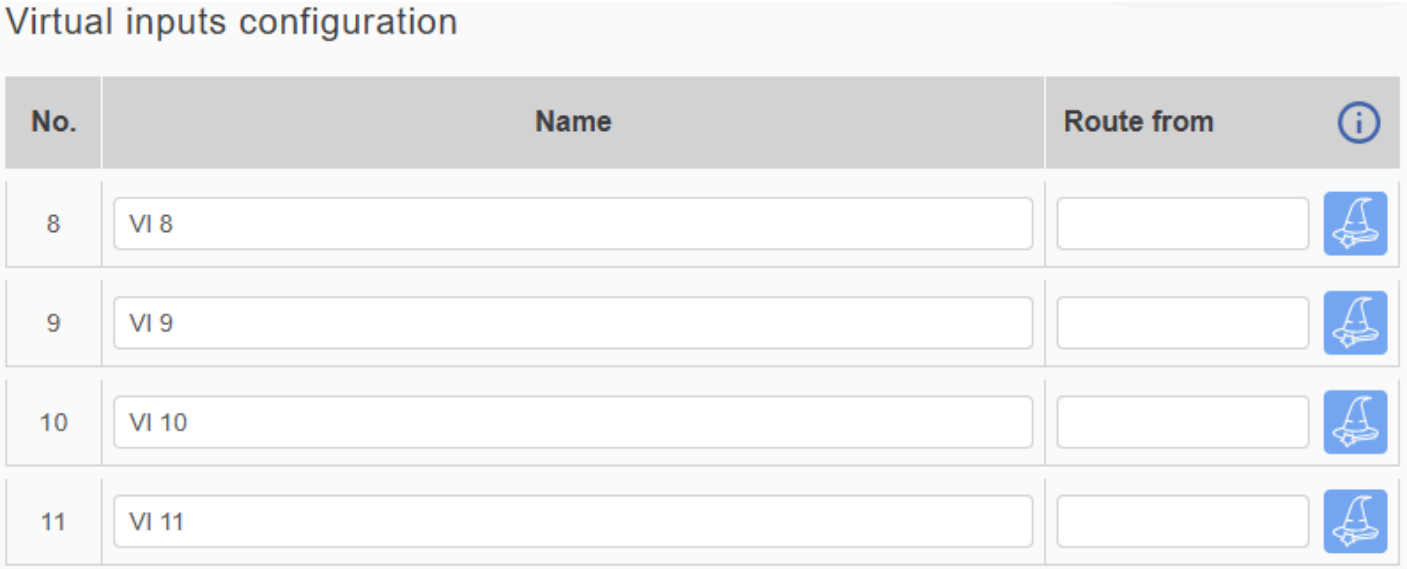
- `o[x]` – Output channel coil state,
- `o[x].on` – Output channel state,
- `v[x]` – Virtual variable state,
- `s[x].aHi` – Sensor high level alarm,
- `s[x].aLo` – Sensor low level alarm,
- `s[x].wHi` – Sensor high level warning,
- `s[x].wLo` – Sensor low level warning,
- `s[x].err` – Sensor error,
- `s[x].ok` – Sensor operates correctly,
- `ping[x]` – Ping status: `0` - error, `1` - success,
- `mb[x].y` – mbAgent value,
- External device:
  - `dev[x].err` – Connection lost,
  - `dev[x].o[y]` – External output,
  - `dev[x].i[y]` – External input,
  - `dev[x].s[y].err` – External sensor error.
- Negation and logic operations (|/&/^) e.g.:
  - `!s[0].err`
  - `!o[0]&v[1]`
- **Route to** – Variable, the output state will have influence on. Available variables:
- `v[x]` – Internal variable,
- `mb[x].y` – mbAgent value,
- External device:
  - `dev[x].o[y]` – External output.
- Negation and logic operations (|/&/^) e.g.:
  - `!s[0].err`
  - `!o[0]&v[1]`

> **Tip**
>
> **If the source is preceeded by "!i", the coil state will be opposite to the state of source channel. For example: !io[3] means that the output will have the state opposite to input 3.**

The  icon opens the step-by-step configuration window.

Inputs

This tab allows to set the virtual inputs in relation to external sources.



- **Name**,
- **Route from** – Source that will have influence over the output state - change of state of the channel in this field will cause the output state to change accordingly.

The following commands can be used (the lis is available by clicking **i**):

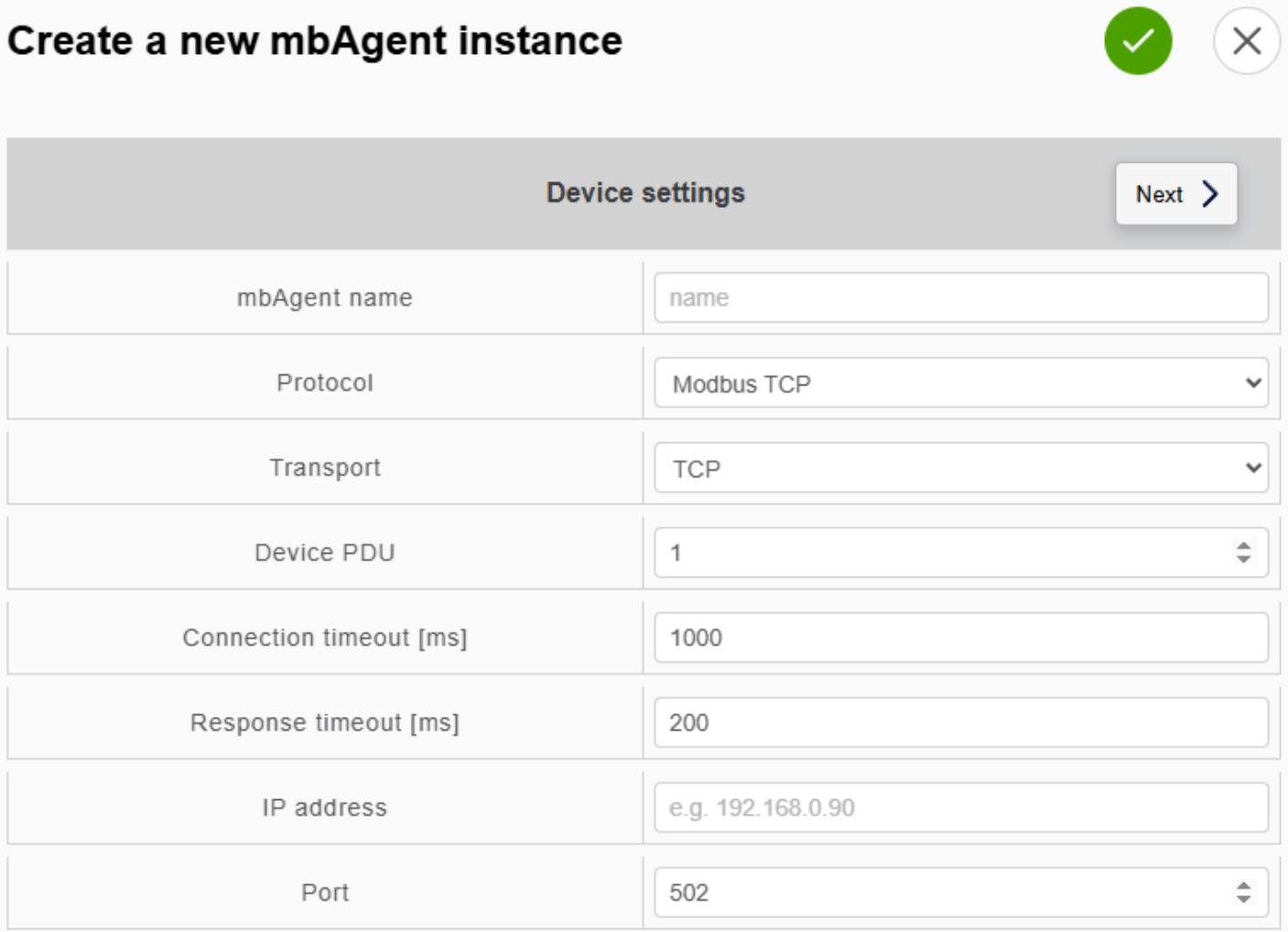
- `i[x]` – Input channel state,
- `o[x]` – Output channel coil state,
- `o[x].on` – Output channel state,
- `v[x]` – Virtual variable state,
- `s[x].aHi` – Sensor high level alarm,
- `s[x].aLo` – Sensor low level alarm,
- `s[x].wHi` – Sensor high level warning,
- `s[x].wLo` – Sensor low level warning,
- `s[x].err` – Sensor error,
- `s[x].ok` – Sensor operates correctly,
- `ping[x]` – Ping status: `0` - error, `1` - success,
- `poll[x].y` – Poller value,
- External device:
  - `dev[x].err` – Connection lost,
  - `dev[x].o[y]` – External output,
  - `dev[x].i[y]` – External input,
  - `dev[x].s[y].err` – External sensor error.

The  icon activates the step-by-step configuration.

mbAgent

Poller is a function of the program that regularly downloads data from other system, device or application using the Modbus TCP/Modbus RTU protocol. The poller operates by sending inquiries to a specified target and awaiting a reply. Clicking **Create a mbAgent instance** will display subsequent dialogue windows with detailed configuration options:

Device settings:

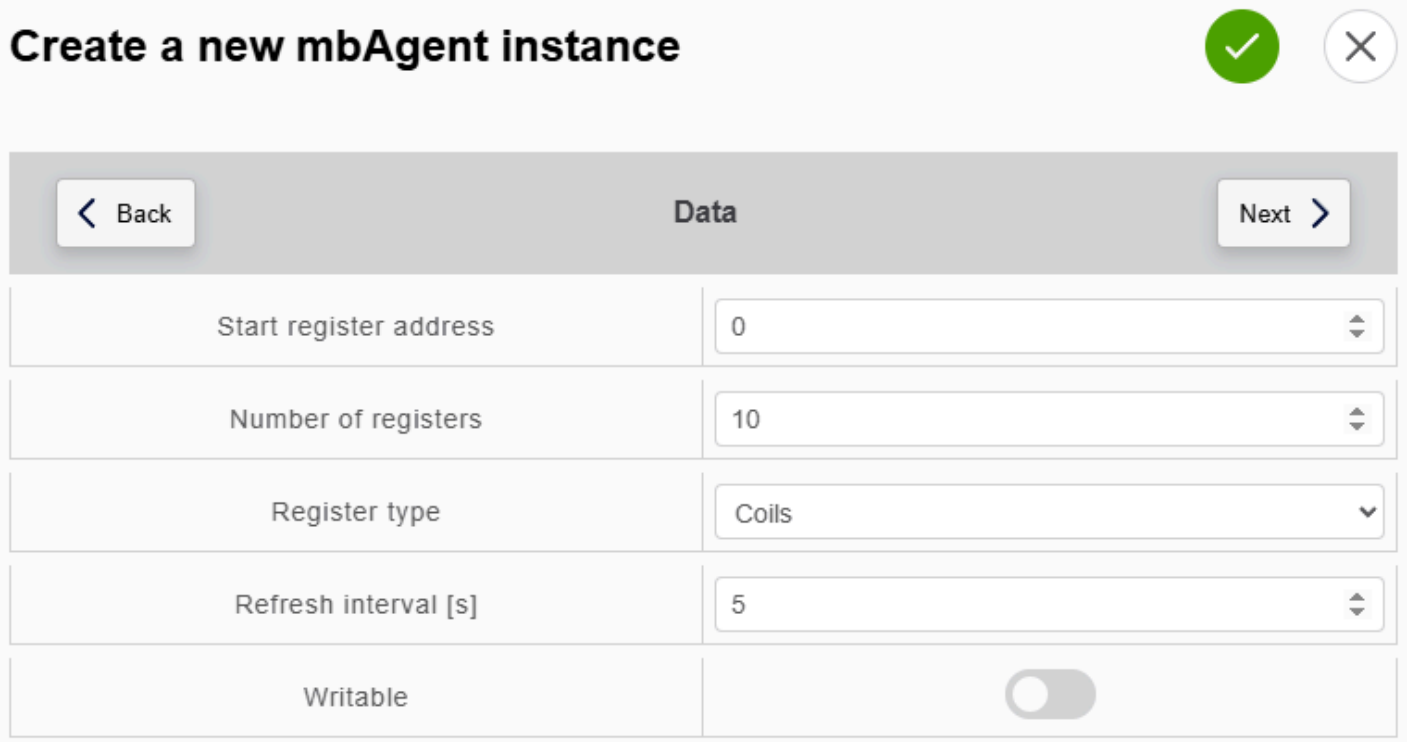


- **mbAgent name** – Poller name,
- **Protocol** – Poller communication protocol:
  - Modbus TCP,
  - Modbus RTU.
- **Transport**
  - TCP (option for Modbus TCP),
  - UDP (option for Modbus TCP),
  - RS485/RS232 *(Modbus RTU - Virtus)*.
- **Device PDU** – Device's Modbus address,
- **Connection timeout [ms]** (only for Modbus TCP),
- **Response timeout [ms]**,

- **IP address (Modbus TCP/RTU via TCP only)** of the monitored device,
- **Port (Modbus TCP/RTU via TCP only)** – Communication port for the monitored device.

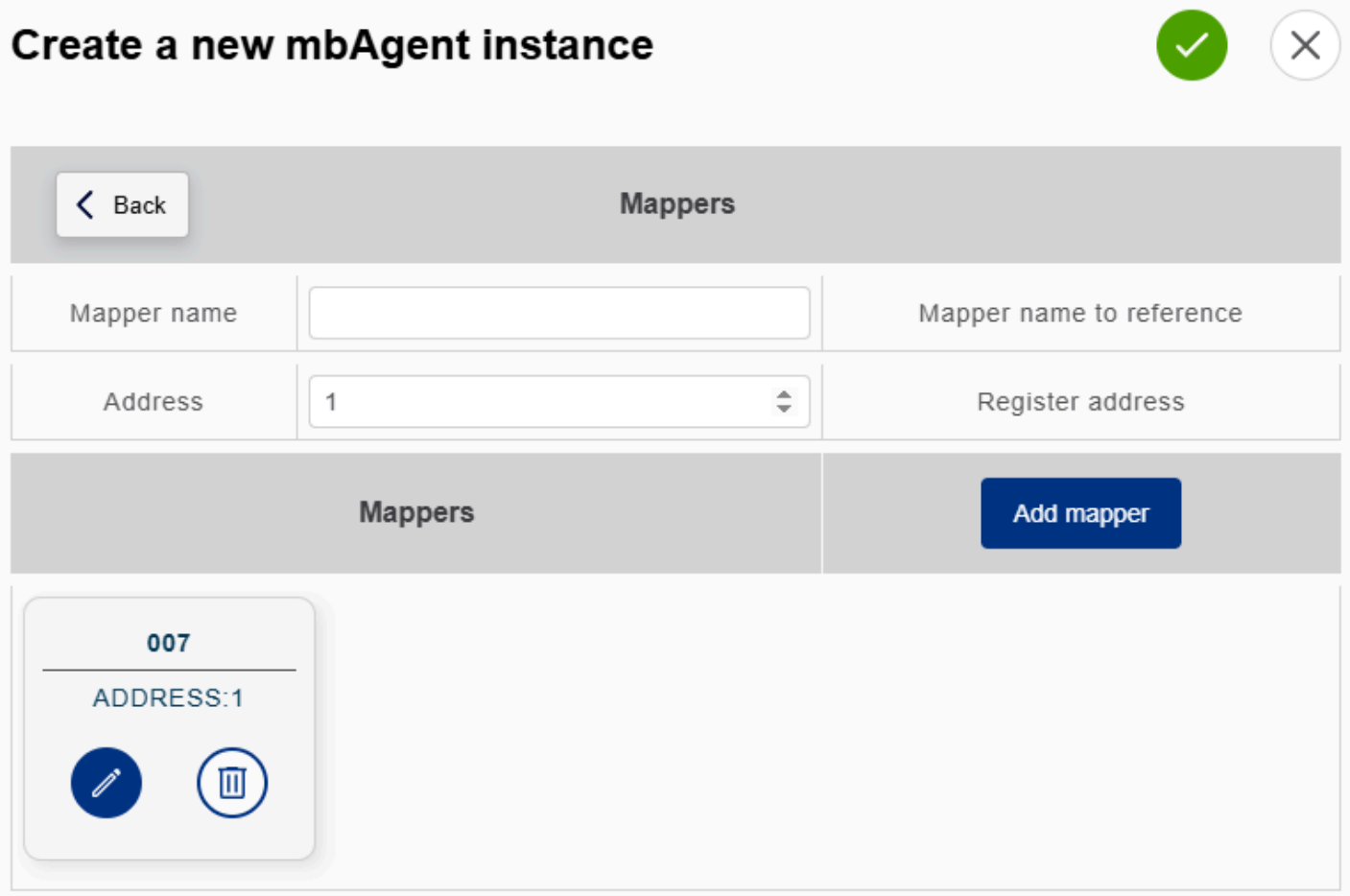
Clicking **Next** will go to the next parameter window.

Data:



- **Start register address**,
- **Number of registers**,
- **Register type:**
  - Coils,
  - Input Register 16-bits,
  - Holding Register 16-bits,
  - Discrete Inputs.
- **Refresh interval [s]**,
- **Writable**.

Mappers:



- **Mapper name**,
- **Address** – Register address,
- **Format** – Variable format selection:
  - Convert to signed integer 16 bits
  - Convert to unsigned integer 16 bits
  - Convert to signed integer 32 bits
  - Convert to unsigned integer 32 bits
  - Swap registers and convert to signed integer 32 bits
  - Swap registers and convert to unsigned integer 32 bits
  - Convert to floating-point 32 bits
  - Swap registers and convert to floating-point 32 bits
- **Multiplication** – Sensor value multiplication.

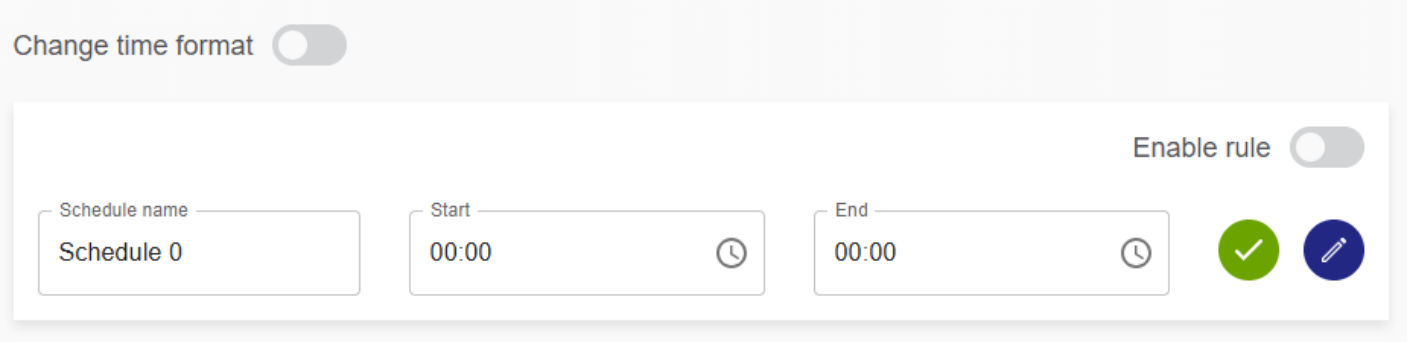
The **Go to the sensor configuration** button enables quick access to the **Sensors** tab.

Schedule

This tab allows the user to control system operation using defined schedules or cron expressions. This advanced function allows for flexible programming of various actions, such as turning devices on and off, defining latitude and longitude for precise synchronization with sunrises and sunsets. Additionally, it's possible to assign specified actions to cron expressions, creating a personalized system operation schedule.

Weekly

It's possible to define up to four weekly schedules – activation of device outputs during specified hours of a weekday.



Change time format – Change time format from 24h to 12h.

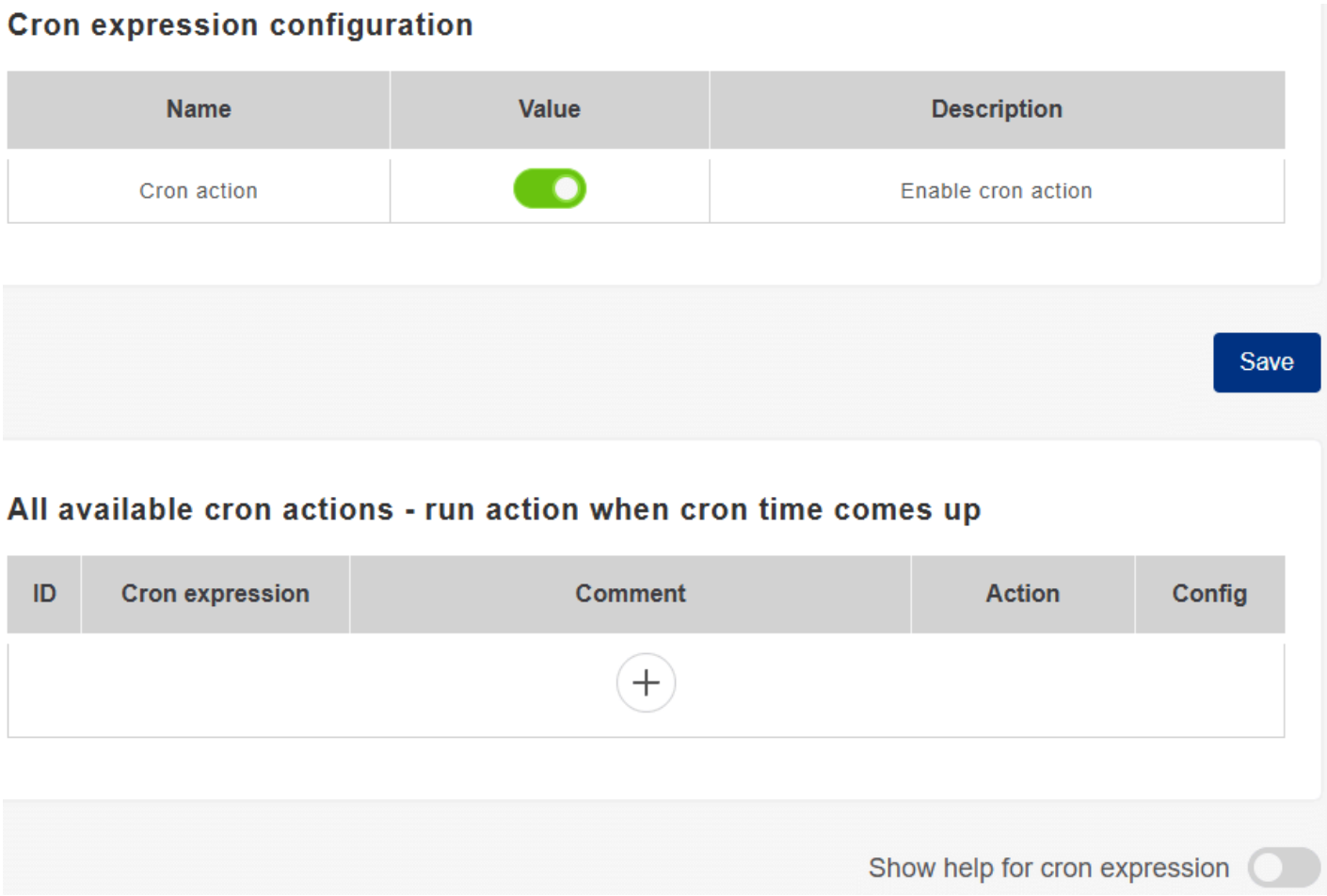
Programming of the schedule should begin with defining the start and end times. Next, using the button, go to weekday selection, and define the outputs that the schedule will aplly to. Save all changes using .

Enable rule – Enables/disables the selected schedule.

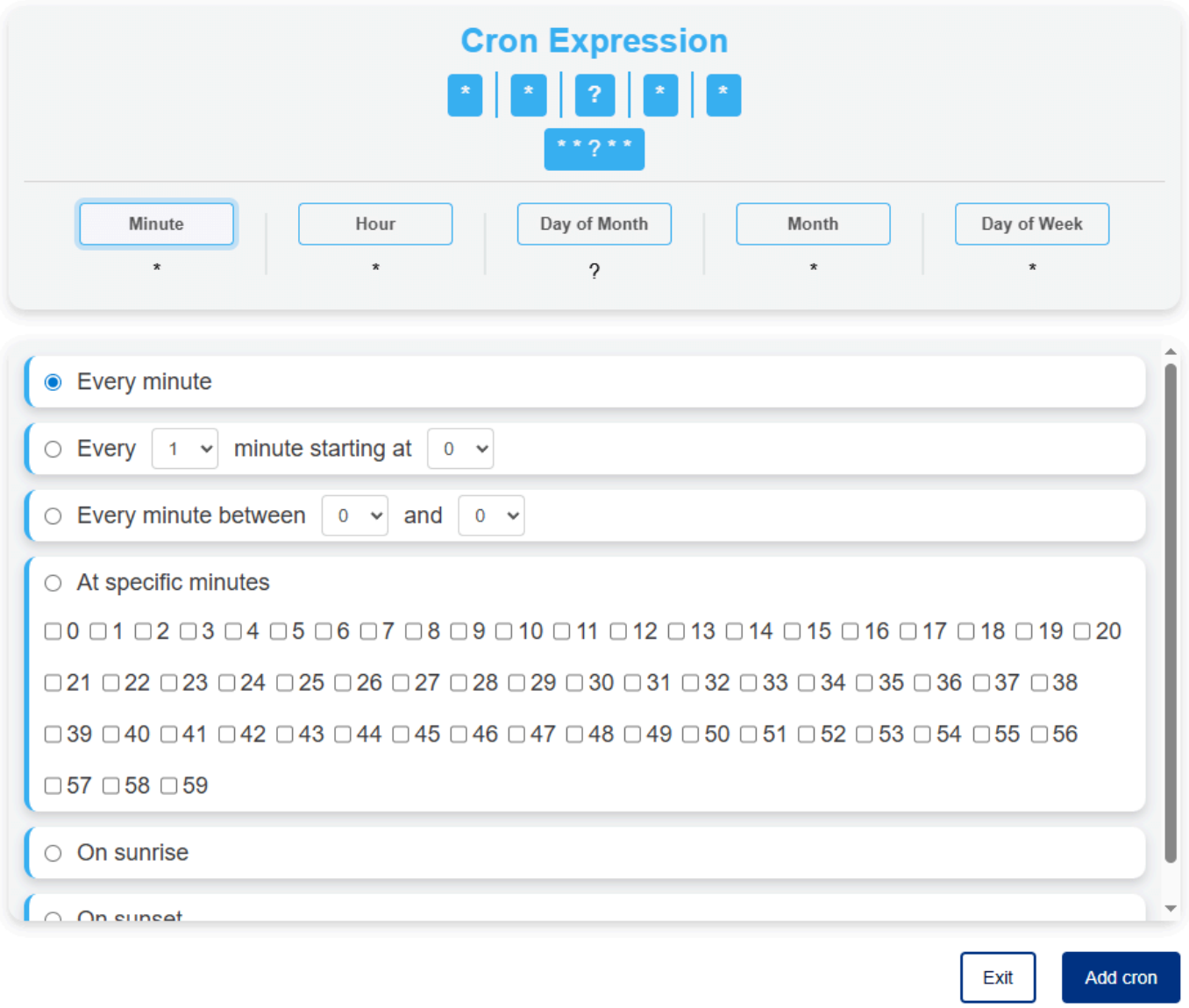
Cron action

The tab allows the user to activate previously defined actions, according to a schedule utilizing a cron expression.

Firstly, enable cron operation:



Clicking **+** will start a step-by-step cron configuration.



After applying the cron time settings, go to the next configuration step:

All available cron actions - run action when cron time comes up

ID	Cron expression	Comment	Action	Config
0	Add cron Cron expression 5 15 4 * ?	Your notes	+	

- **Cron expression** – Defined cron expression.  returns to cron configuration,
- **Comment** – User comments,
- **Action** – Assign actions defined previously – consult **Defining actions**,
- **Config** – Allows the user to accept the configuration or leave the editor.

Cron standard

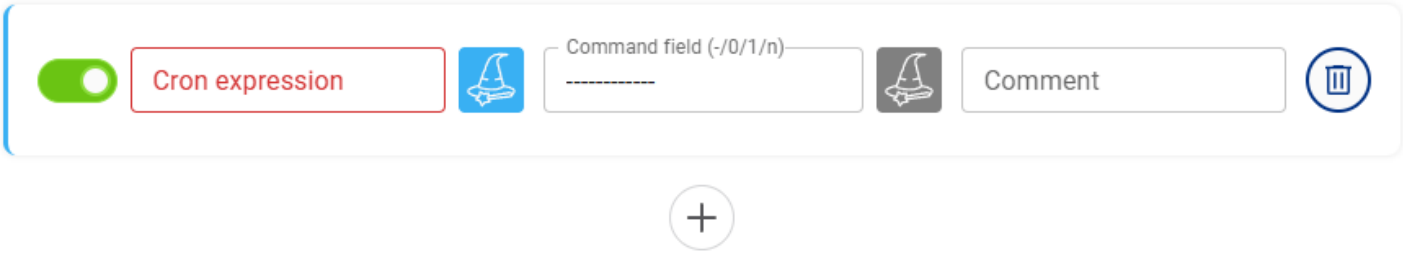
This tab allows for programming of output operation according to the cron expressions:

Cron expression

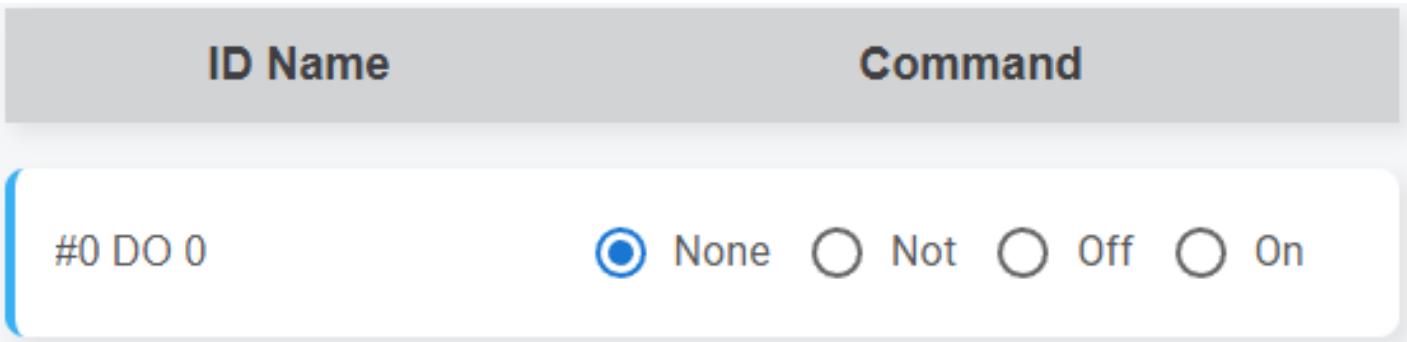
Name	Value	Description
Cron standard		Enable cron standard
Missed Cron Replay		Execute cron commands that were missed due to power loss Max recovery time: 7 days of power loss

- **Cron standard** – Enable/disable Cron,
- **Missed Cron Replay** – Execution of cron orders that have been missed because of a power failure. Maximum restore time: 7 days since power loss.

Step-by-step cron configuration:



Clicking will go to the cron expression configuration. After a cron expression is added, command configuration is possible - program how particular outputs should operate during the set cron expression. The following options are available to each output:



- **None** – No state change,
- **Not** – State inversion,
- **Off**,
- **On**.

Apply the configuration by clicking **Add command**.

Click **+** to add additional commands.

Advanced cron configuration:

Advanced configuration allows the user to access advanced cron configuration.

The drawing describes the meaning of columns:

```
* * * * * Control command
| | | | |
| | | | ---------- Day of the week (0-7) (0 and 7 sunday)
| | | -------------- Month (1-12)
| | ----------------- Day of the month (1-31)
| --------------------- Hour (0-23)
------------------------ Minute (0-59)
```

The control command consists of 8 fields representing the state of outputs.

Each field can assume the values:
- 1 – Output activation,
- 0 – Output deactivation,

- n – Negation (state inversion),
- - – Leave the output with no changes.

Example

Wyjście	1	2	3	4	5	6	7	8
Stan	1	1	0	1	n	1	-	0

- Activate outputs 1, 2, 4, 6,
- Deactivate outputs 3, 8,
- Invert output 5,
- Output 7 - no changes.

The controller recognizes the following entries:

- Sunday can be represented by 0 or 7
- The day of command execution can be inputted in two ways: day of the month or day of the week. If two fields are filled, the command will be executed both on the day of the month and day of the week!
- Numerical values can be saved in various formats:
  - 1-3 – Values 1, 2, 3,
  - 0-10/2 – Values 0, 2, 4, 6, 8 i 10 (every 2nd value in the range from 0 to 10),
  - 1,2,5 – Values 1, 2, 5,
  - */2 – Every 2nd allowed value (np. in the first collumn it will be 0, 2, 4, 6...56, 58),
  - 1-3,5,6 – 1, 2, 3 and 5, 6,
- SR – time of the sunrise,
- SS – time of the sunset.

Tip

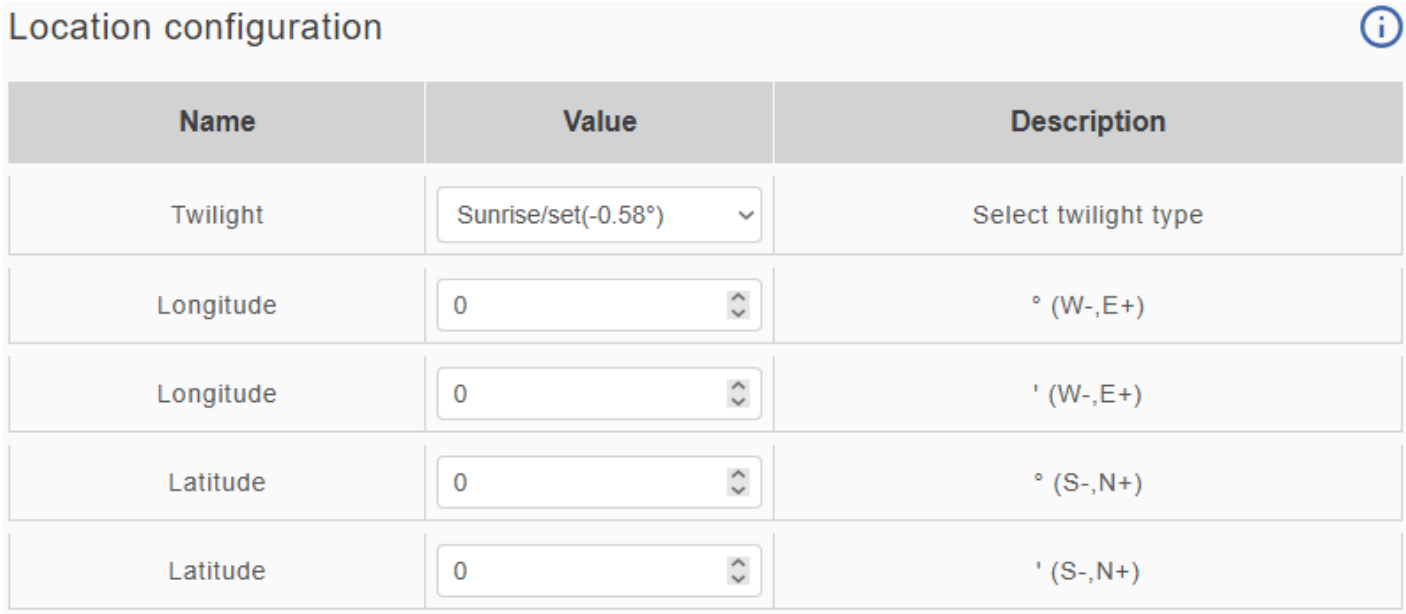
Because of Daylight Savings Time, avoid tasks that should happen between 2:00 and 3:00 on a Sunday. During automatic change from winter to summer time, the action will not complete, while during change from summer to winter time, it will be completed twice.

Example

- Activate relay No. 3 at 12:45: `45 12 * * * --1------`
- Activate relay No. 5 at 17:30 on the 16th of October: `30 17 16 10 * ----1---`
- Activate relay No. 2 at 8:00 form Monday til Friday: `00 8 * * 1-5 -1--------`
- Activate relay No. 1 and 5 from Monday til Friday at sunrise: `SR SR * * 1-5 1---0---`
- Activate relay No. 1 at dusk and deactivate at dawn: `SS SS * * * 1---------  SR SR * * * 0---------`
- Activate relay No. 3 10 minutes after dawn: `SR+10 SR * * * --1-----`

Configuration

This tab allows output control configuration using cron expressions. The device can calculate the sunrise and sunset times automatically depending on the specified location – to do that, geographical coordinates should be entered in the degree and minute format.



- **Twilight** – Twilight type selection – see **Twilight**,
- **Longitude** – Degrees (positive value for the eastern longitude, negative for western longitude),
- **Longitude** – Minutes (positive value for the eastern longitude, negative for western longitude),
- **Latitude** – Degrees (positive value for the northern latitude, negative for southern latitude),
- **Latitude** – Minutes (positive value for the northern latitude, negative for southern latitude).

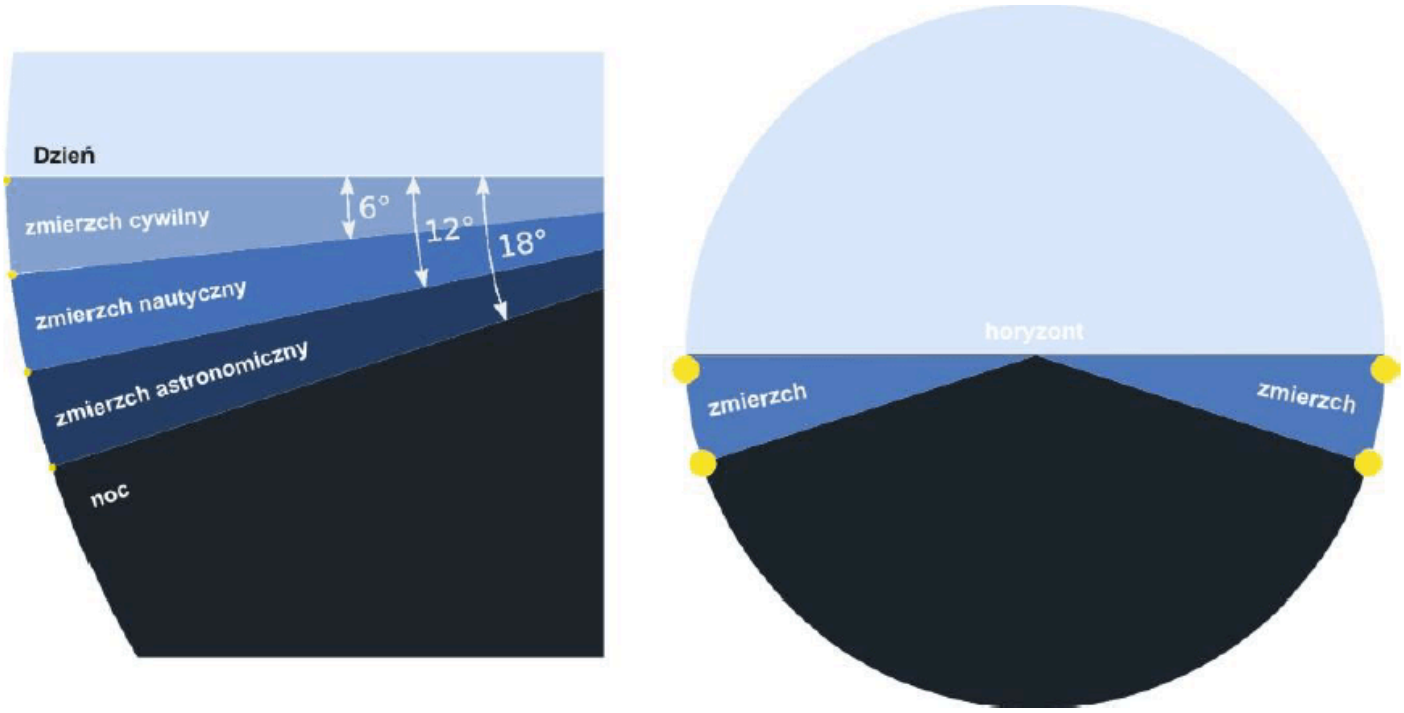
Tip

The blue "i" icon in the upper-right corner of the tab enables preview of the calculated sunrise and sunset times.

Twilight

- **Civil (-6º)** – Civil twilight - the phase of sunset when the centre of the solar disc is no more than 6 angular degrees below the horizon. During this period, the brightest stars and planets appear in the sky, retaining enough natural light for normal outdoor activities.
- **Nautical (-12º)** – Nautical twilight (also known as dusk) - the time after sunset when the centre of the solar disc is less than 6 angular degrees below the horizon, but still above 12°. This phase allows the horizon line to be clearly observed, which is important in astronavigation, enabling simultaneous navigation based on terrestrial and celestial objects.
- **Astromical (-18º)** – Astronomical twilight - the period when the centre of the solar disc is between 12 and 18 angular degrees below the horizon. It ends with the beginning of astronomical night. Astronomical dawn - the time before sunrise when the sun's rays light up the sky, making the faintest stars no longer visible. Astronomical dawn lasts from the time the centre of the Sun's disc is more than 18° below the horizon line until it is more than 12° below the horizon line. The astronomical dawn ends the astronomical night and then the nautical dawn begins.
- **Nautical dawn** – The time before sunrise when the horizon and landscape become visible, allowing navigation based on illuminated objects on land or sea. During this time, the brightest stars are still visible in the sky. Nautical dawn lasts from the moment when the centre of the Sun's disc is more than 12° below the horizon line, but still less than 6° below the horizon line.

- **Civil dawn** – The time before sunrise when the centre of the sun's disc is already more than 6° below the horizon line.

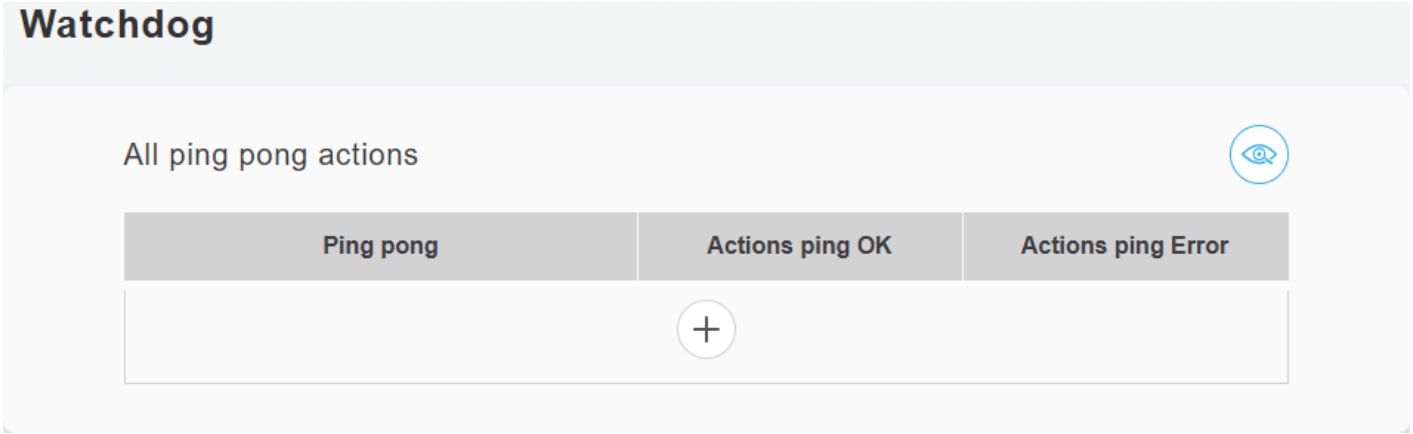


Virtus calculates the hour of the sunrise and sunset - to do that, input of geographical coordinates is needed (longitude and latitude in the degrees/minutes format).

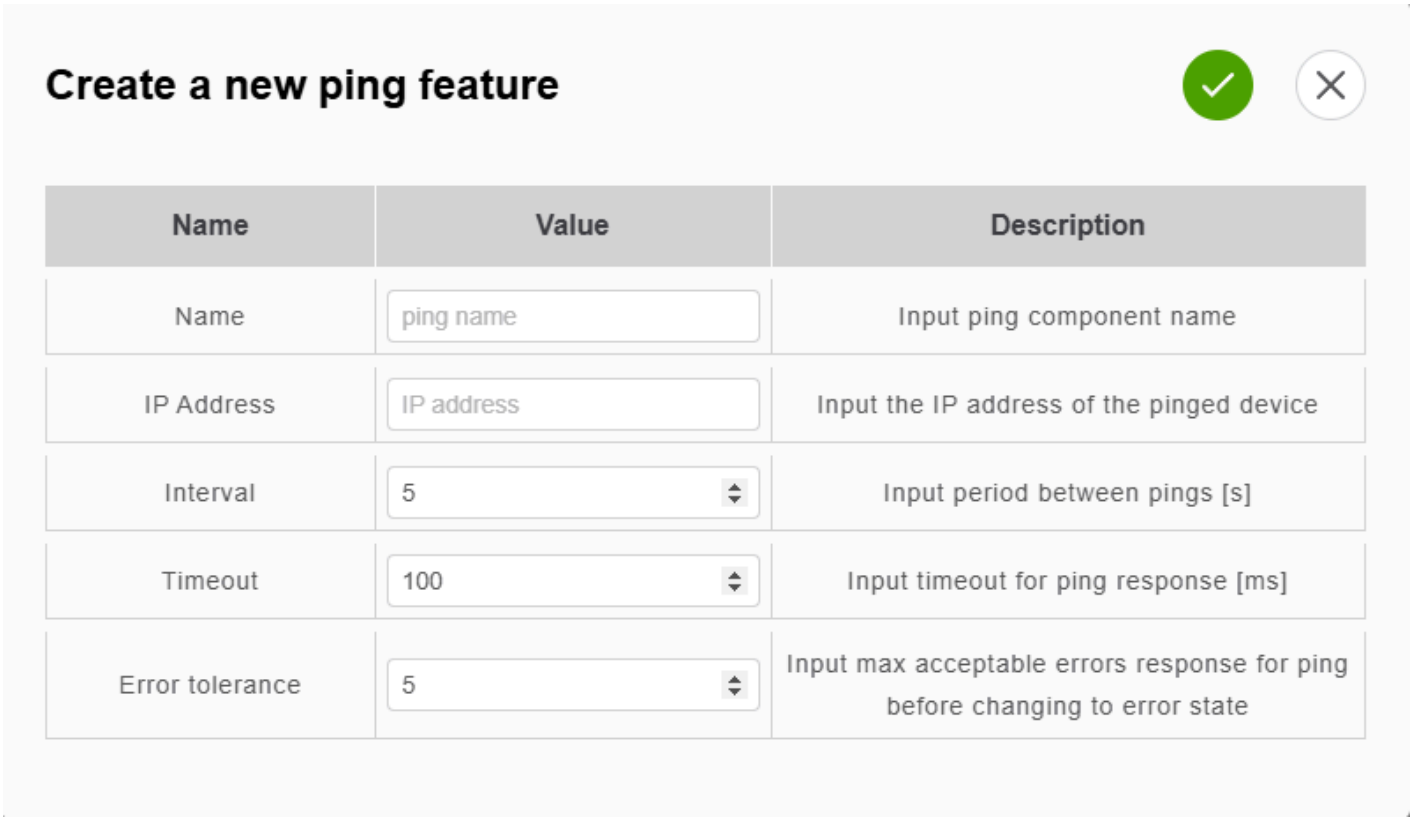
- **Twilight** – Select the astronomical clock mode:
  - **Sunrise/set (-0,58°)**,
  - **Civil (-6°)**,
  - **Nautical (-12°)**,
  - **Astronomical (-18°)**.

Remote hosts ping (Watchdog)

The device is equipped with the function of "pinging" – it can interrogate remote hosts (other network devices, servers etc.) and react to their availability.



To add a new device to check the availability of, click **+** in the **All ping pong actions** table – a **Create a new ping feature** dialogue window will appear:



- **Name** – Ping name,
- **IP address** – IP address of the pinged device,
- **Interval** – Ping interval (in seconds),
- **Timeout** – Ping response time limit,
- **Error tolerance** – Maximum permissible amount of ping response errors before error state occurs.

Tip

All changes should be saved with

A new ping will show up in the table, **Actions ping OK** and **Actions ping Error** columns will be populated with **+** buttons, that allow assignment of selected actions that are to be executed in the event of correct and erroneous ping response. The process of assigning actions is described in the **Defining actions** chapter.

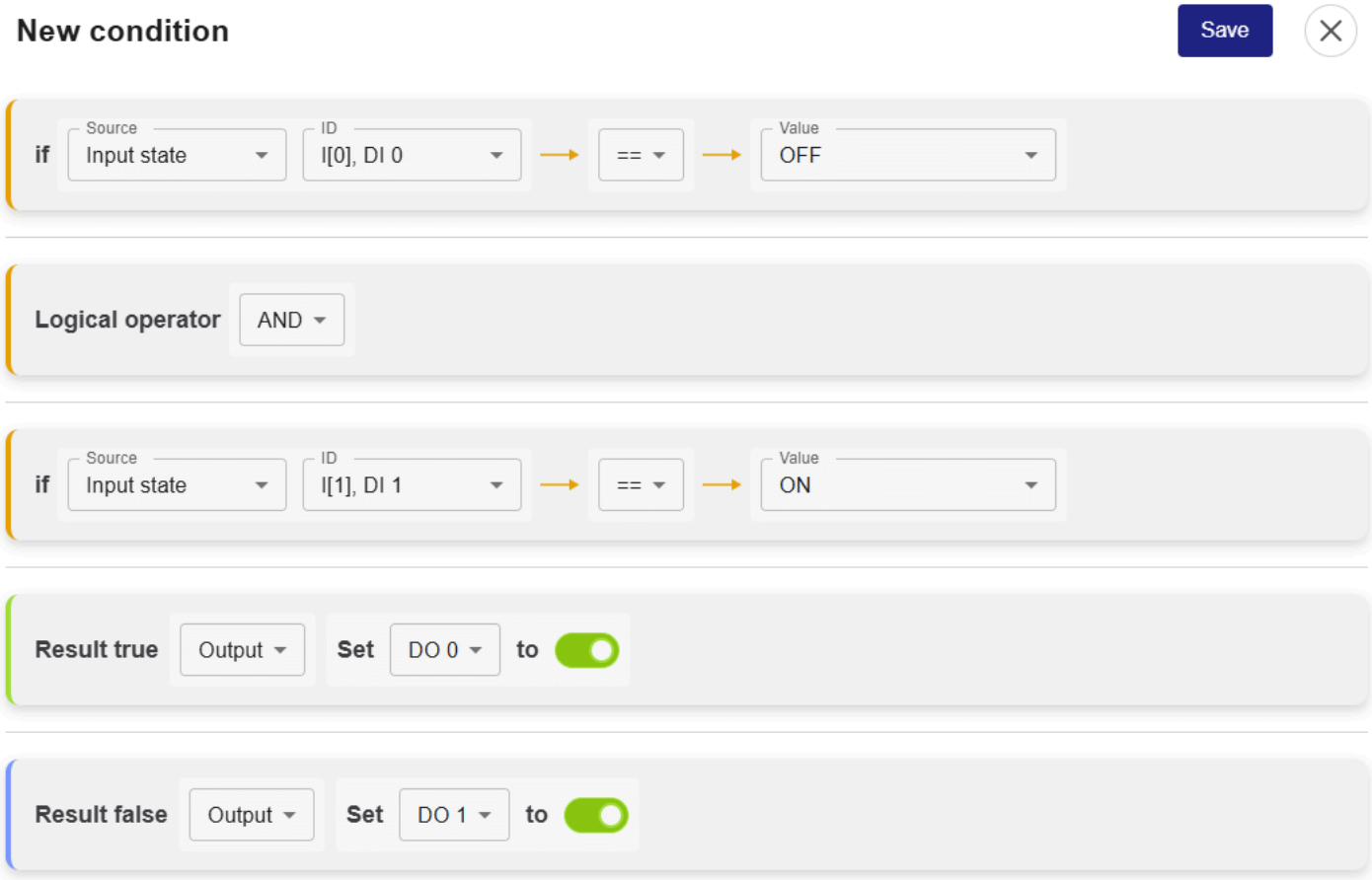
Clicking will activate an additional indicator next to the request name - assuming **green** colour for a correct ping response or **red** if a response error occurs.

Logic functions

This tab allows configuration of logic functions – afer certain conditions are met, a defined action is executed.

Conditions

To configure a new logical funtion, click **Add a condition**.



Available logical variables *(availability depends on the device version)*:

- **input** – Physical or virtual input state,
- **input cnt** – Input counter,
- **output** – Physical or virtual output state,
- **sensor** – Sensor reading,
- **sensor state**:
  - **0** – Sensor error,
  - **1** – Normal state,
  - **2** – Sensor low level warning,
  - **3** – Sensor high level warning,
  - **4** – Sensor low level alarm,
  - **5** – Sensor high level alarm.
- **constant**,
- **variable**,
- **active input action** – In **cnt** input mode,
- **schedule**.

Available logic operators:

- **AND** – Conjunction,

- **OR** – Disjunction (logical sum),
- **XOR** – Exclusive disjunction,
- **NAND** – Negation of conjunction,
- **NOR** – Negation of disjunction,
- **XNOR** – Negation of exclusive disjunction (logical equivalence).

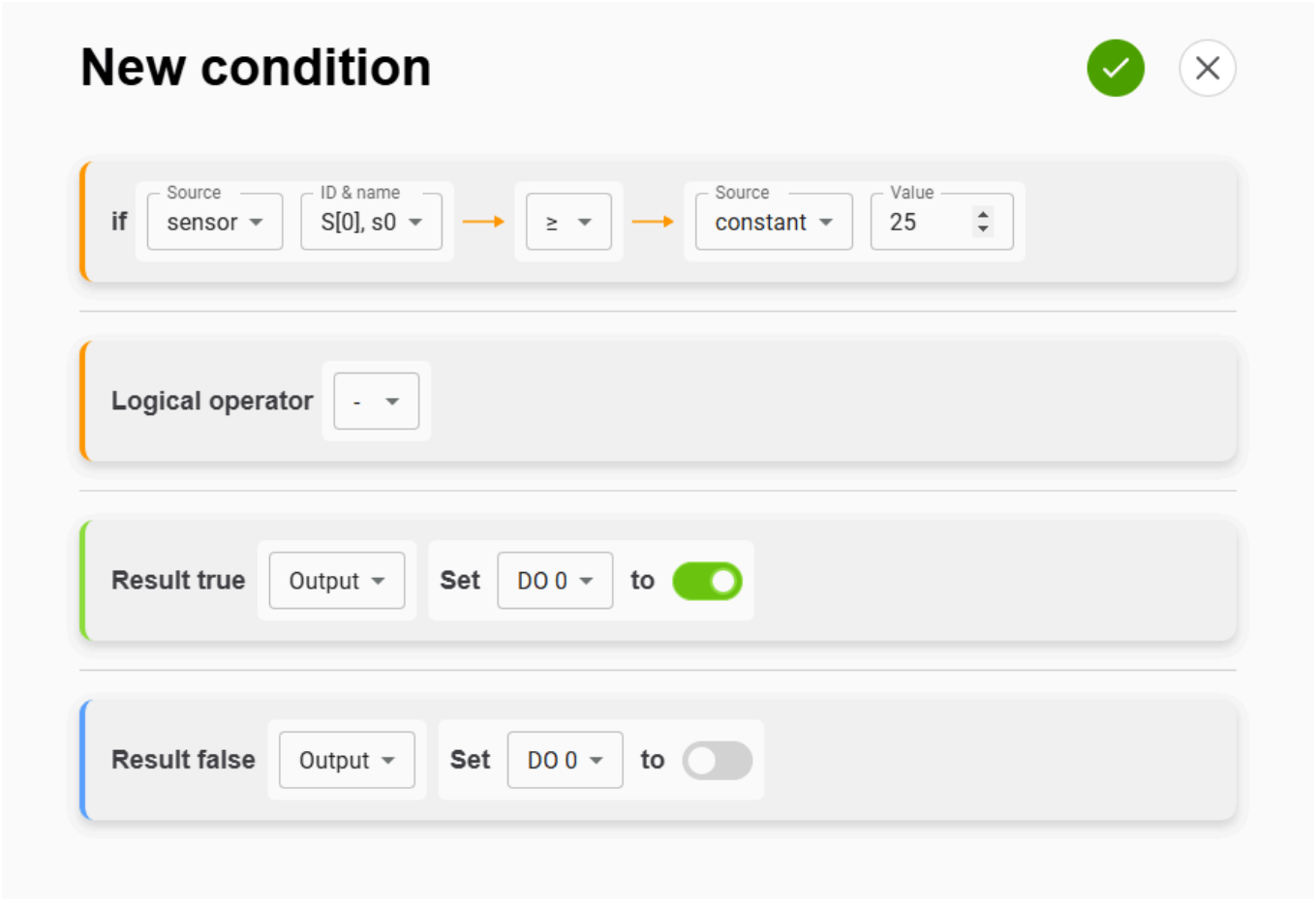
Result True / Result False – Reaction for meeting/not meeting the condition:

- **None**,
- **Output** – Output control,
- **Action** – Recall defined action,
- **Variable** – Virtual input/output control.

Tip

All changes should be saved with

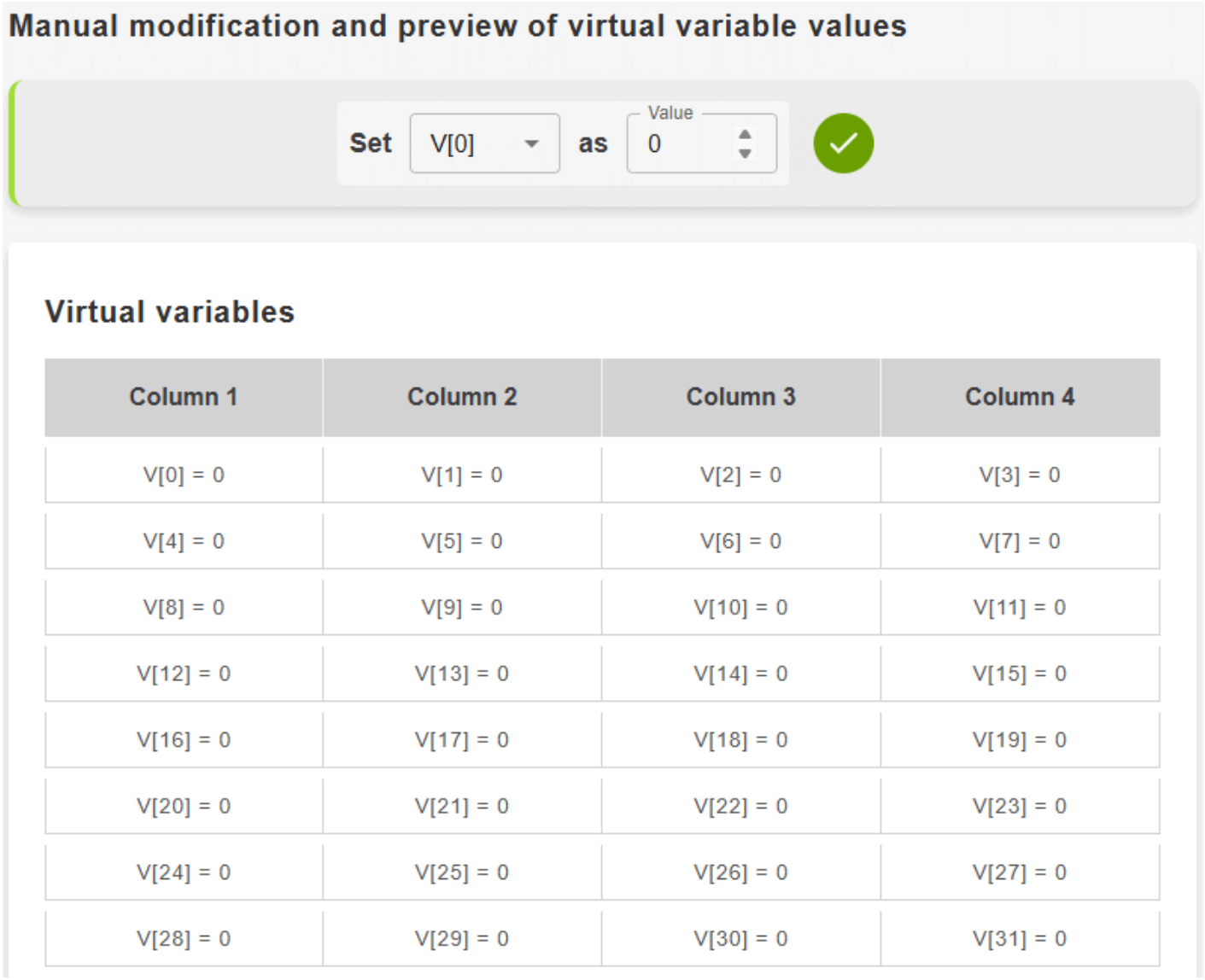
Example



Output 1 will be activated as long as the sensor 0 temperature is greater than or equal to 25. If the temperature drops below this value, output 1 will be deactivated.

Variables

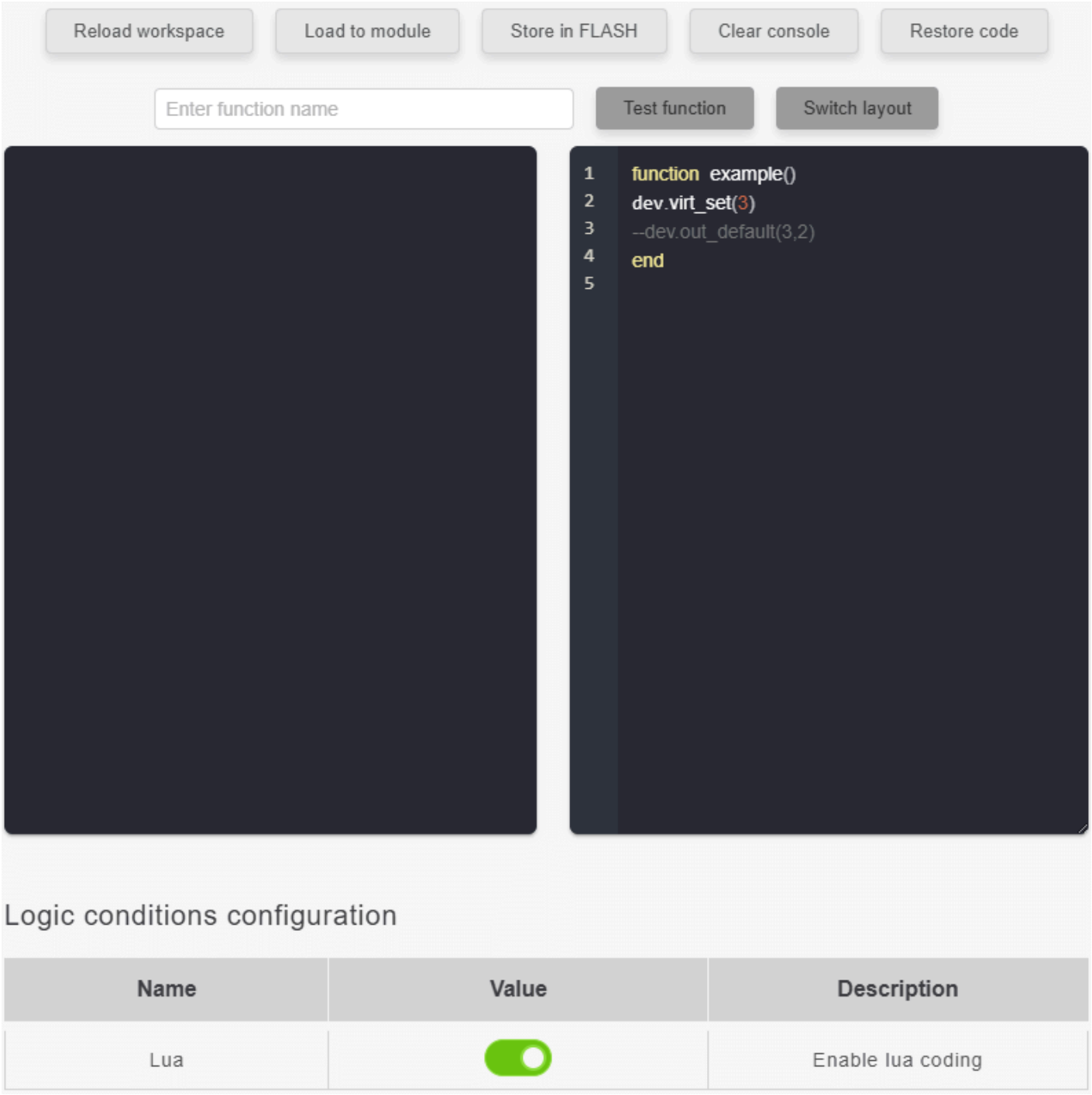
The **Variables** tab allows the user to define the values of internal variables.



Lua Script

The **Lua Script** tab allows the user to define custom actions and input/output control logic using the Lua script language.

The main tab section contains a code editor that can be used to create and edit Lua scripts. Those scripts allow customization of device operation depending on different events.



- **Reload workspace** – Reload the editor,
- **Load to module** – Upload the Lua script to the device (without saving to persistent memory),
- **Store in FLASH** – Save the Lua script to device memory (the script will remain active after a restart),
- **Clear console** – Clear the output console,
- **Restore code** – Restore the default Lua code.

Tip

The "Lua" option located at the bottom of the tab allows the user to enable or disable Lua script support in the device.

AVAILABLE SYSTEM FUNCTIONS

- `start()` – Recalled after the system start-up,
- `dev.timeout()` – Recalled after `dev.set_timeout(ms)` period has passed,
- `httpdscript(name,args)` – Recalled when a HTTP request begins with `/data/script/`,
- `set_timeout(ms)` – Sets the internal timer and recalls the `timeout()` function,
- `clear_timeout()` – Clears and disables the timer.

LUA LIBRARIES

The system offers several libraries enabling interaction with the device. The `dev` library is specific to Inveo devices. All other libraries are standard.

> **Tip**
>
> **All channel and sensor indexes start from `0`.**

`dev` library

- `dev.out_default(ch, fun)` – Output No. `ch` control in configured mode:
  - `0` – Disable,
  - `1` – Enable,
  - `2` – Invert state.
- `dev.out_static(ch)` – Enable output `ch` in static mode,
- `dev.out_off(channel)` – Disable the output on a given channel `ch`,
- `dev.out_single(ch, ton, toff)` – Sets a single pulse, `ton` long after a delay of `toff` (in 0,1s) on the output channel `ch`,
- `dev.out_blink(ch, ton, toff, cnt)` – Blinks the output `ch` `cnt` - times with pulses of `ton` length separeted by `toff` (in 0,1s),
- `dev.out_get(ch)` – Returns the `ch` output state,
- `dev.in_get(ch)` – Returns the `ch` input state,
- `dev.in_get_name(ch)` – Returns the `ch` input name,
- `dev.in_get_cnt(ch)` – Returns the `ch` input counter,
- `dev.in_set_cnt(ch, val)` – Set the `ch` input counter value to `val`,
- `dev.virt_set(id, val)` – Set virtual variable `id` to `val`,
- `dev.virt_get(id)` – Returns the state of the virtual variable `id`,
- `dev.sensor_get(id)` – Returns the sensor `id` state and value `(state,value)`:
  - `0` – Sensor error (damaged/incorrectly connected etc.),
  - `1` – The sensor returns correct readings,
  - `2` – Sensor low level warning,
  - `3` – Sensor low level alarm,

    - `4` – Sensor high level warning,
    - `5` – Sensor high level alarm.
- `dev.action(name)` – Recalls the action referenced by its `name`.

Communication

E-mail

- `dev.email(recipient, subject, text)` – Sends an e-mail:
    - `recipient` – Recipient address,
    - `subject` – Message topic,
    - `text` – Message content.

HTTP

- `dev.http_send(url, data, data_length, method, content_type)` – Sends an HTTP request:
    - `url` – Address, the device will send the request to. Entering `null` will send the request to the address specified in the **HTTP Client** tab,
    - `data` – Data that will be sent,
    - `data_length`,
    - `method` – HTTP method:
        - `0` – GET,
        - `1` – POST,
        - `2` – PUT,
        - `3` – DELETE.
    - `content_type`:
        - `0` – PLAIN,
        - `1` – JSON,
        - `2` – XML.

MQTT

- `dev.mqtt(topic, data, length)` – Sends an MQTT notification:
    - `topic` – Topic, the notification will be sent to,
    - `data` – Data that will be sent,
    - `length` – Data length.

Logging and debugging

- `dev.log(message)` – Saves message to flash memory,
- `dev.console(message)` – Dispays the message in the Lua console.

`httpdscript()` request response

- `dev.httpd_resp()` – Sets the HTTP response.

`table` library

- `table.insert(t, [pos], value)` – Inserts `value` to array `t`. If `pos` is specified, inserts the element to a specified position, shifting other elements,
- `table.remove(t, [pos])` – Removes the element from `pos` in `t` array. If `pos` is not specified, removes the last added element,
- `table.sort(t, [, comp])` – Sorts the `t` array in ascending order. Optionally, the `comp(a, b)` comparison function can be added; it should return `true` if `a` should be before `b`,
- `table.concat(t [, sep [, i [, j]]])` – Combines the `t` array contents into a single string, separating them with `sep` from index `i` to `j`,
- `table.pack(...)` – Returns an array containing all function arguments. Remembers the amount of arguments as `n`,
- `table.unpack(t [, i[, j]])` – Returns the contents of `t` array as separate values for indexes `i` to `j`. Default: `i = 1`, `j = #t`.

`string` library

- `string.bytebyte(s [, i [, j]])` – Returns the byte code for `s` characters, from `i` to `j`,
- `string.char(...)` – Creates a string from the specified values,
- `string.len(s)` – Returns the character string length,
- `string.lower(s)` – Returns `s` with upper-case letters converted to lower-case,
- `string.upper(s)` – Returns `s` with lower-case letters converted do upper-case,
- `string.sub(s, i [, j])` – Returns the `s` substring from `i` to `j`,
- `string.rep(s, n)` – Returns `s` repeated `n` times,
- `string.reverse(s)` – Returns the inverted text of `s`,
- `string.find(s, pattern [, i [, plain]])` – Searches for `pattern` in `s`, returns the indexes or `nil`. Setting `plain` to `true` searches with no patterns,
- `string.match(s, pattern [, i])` – Returns the first pattern match,
- `string.gsub(s, pattern, repl [, n])` – Substitutes `pattern` with the `repl` string a maximum of `n` times. Returns the new string and the amount of substitutions.

`os` library

- `os.time([table])` – Returns current time in UNIX format or converts `table` to seconds,
- `os.date([format [, time]])` – Returns the date/time in the form of formatted text or a table,
- `os.clock()` – Returns the time spent for program exacution by the processor,
- `os.difftime(t1, t2)` – Returs the difference in seconds between `t1` and `t2`,

`math` library

- `math.abs(x)` – Absolute value of `x`,

- `math.ceil(x)` – Round up to the nearest whole number,
- `math.floor(x)` – Round down,
- `math.fmod(x, y)` – Remainder of the `x` by `y` division operation,
- `math.max(x, ...)` – The biggest number specified,
- `math.min(x, ...)` – The smallest number specified,
- `math.sqrt(x)` – Square root of `x`,
- `math.exp(x)` – `e` to the power of `x`,
- `math.log(x [, base])` – Natural logarithm `ln(x)` or `log_b(x)` (if `base` is specified),
- `math.pow(x, y)` – Exponentiation of `x^y`,
- `math.modf(x)` – Splits `x` into integer and fractional parts `int, frac`,
- `math.random([m [, n]])` – Random number between 0 and 1,
- `math.randomseed(seed)` – Sets a seed of the random number generator,
- `math.deg(x)` – Converts radians to degrees,
- `math.rad(x)` – Converts degrees to radiants,
- `math.sin(x)` – Sinus (`x` in radians),
- `math.cos(x)` – Cosinus (`x` in radians),
- `math.tan(x)` – Tangens (`x` in radians),
- `math.asin(x)` – Arcus sinus,
- `math.acos(x)` – Arcus cosinus,
- `math.atan(x)` – Arcus tangens,
- `math.atan2(y, x)` – Arcus tangens `y/x` including the quadrant.

`utf8` library

- `utf8.len(s [, i [, j]])` – Returns the number of UTF-8 characters in the substring `s` from `i` to `j`,
- `utf8.offset(s, n [, i])` – Returns the byte index in `s` of the beggining of the nth character (if `n` positive) or the nth from the end (if `n` negative),
- `utf8.codepoint(s [, i [, j]])` – Returns at least one Unicode point code of characters from `s` between `i` and `j`,
- `utf8.char(...)` – Creates an UTF-8 string from the given Unicode point codes,
- `utf8.codes(s)` – Returns the iterator, that returns subsequent byte indexes and Unicode point codes in `s`,
- `utf8.charpattern` – Lua pattern for matching a single UTF-8 character.

`pack` library

- `pack.pack` – Packs data to a binary string,
- `pack.unpack` – Unpacks data from a binary string,
- `pack.packsize` – Returns the size of a packet.

Example 1

```
function example()
--download sensor ID=0 state
    state,value = dev.sensor_get(0)
    if(value>20) then
        dev.console("Highest value" .. value)
    --set channel ID 3 to:
    --0-off,1-on,2-inv
        dev.out_default(3,2)
    elseif (value<10) then
        dev.console("Lowest value" .. value)
    else
        dev.console("Mean value" .. value)
    end
end

--recall function when the device starts
function start()
    dev.set_timeout(100);
end

--recall the timeout function
function timeout()
    example()
    dev.set_timeout(100);
end
```

Example 2

The following example ilustrates the functionality of sending sensor readings to a remote LED display via HTTP.
4 sensors are connected to Daxi:
- 0, 1 – temperature,
- 2, 3 – humidity.

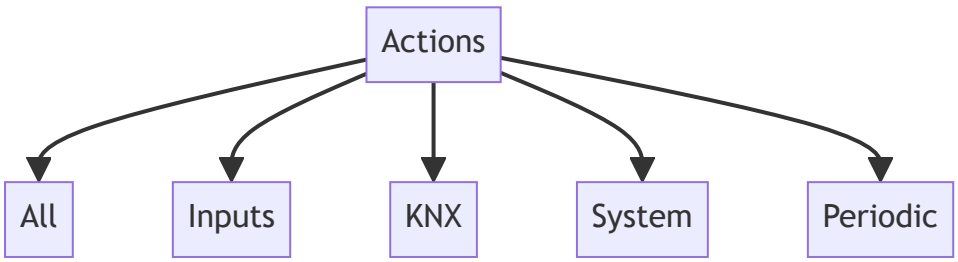
```
n_cnt = 0 --set the code execution counter to 0

function timeout()
    dev.set_timeout(5000) --delay between subsequent notifications, in ms

    text,len = get_data(n_cnt)
    dev.http("http://192.168.11.116/status.xml", text, len, 0, 0) --display's address and resource,
text to be sent, data string length, method (GET), content type (PLAIN)
    n_cnt = n_cnt + 1 --counter incrementation
    if n_cnt>4 then --max. No. of sensors (4)
        n_cnt=0 --set the counter to 0 after the last sensor's data is sent
    end
end

function get_data(index)
    state,value = dev.sensor_get(index)
    local label
    if index <= 1 then --index of the last temperature sensor
        label = "TMP" .. index --adds the prefix TMP for sensors 0 and 1
    else
        label = "HUM" .. index --adds the prefix HUM for other sensors
    end

    text = "texta=1" .. label .. "%20" .. value --request syntax required by the display (e.g.
texta=1HUM%2070.0)
    len = string.len(text) --calculate the data string length
    return text, len --return the data (text) and string length (len)
end

--call the function when the device starts
function start()
    dev.set_timeout(5000); --initialization time
end
```

Defining actions

The device undertakes user-defined actions in response to specific signals or sensor readings. Those may include:

- **Output control:** Activation or deactivation of a selected output based on sensor readings. (eg. turning a fan on after temperature exceeds a certain threshold),
- **Sending notifications in the form of SMS, e-mail, MQTT frame, HTTP, TCP, UDP, SNMP trap, and other:** Automatic sending of alert/notification to the user or another system in response to defined conditions,
- **Other actions defined by the user:** Actions specific to a particular system or needs such as saving data to a database, alarm activation, change to other device settings etc.

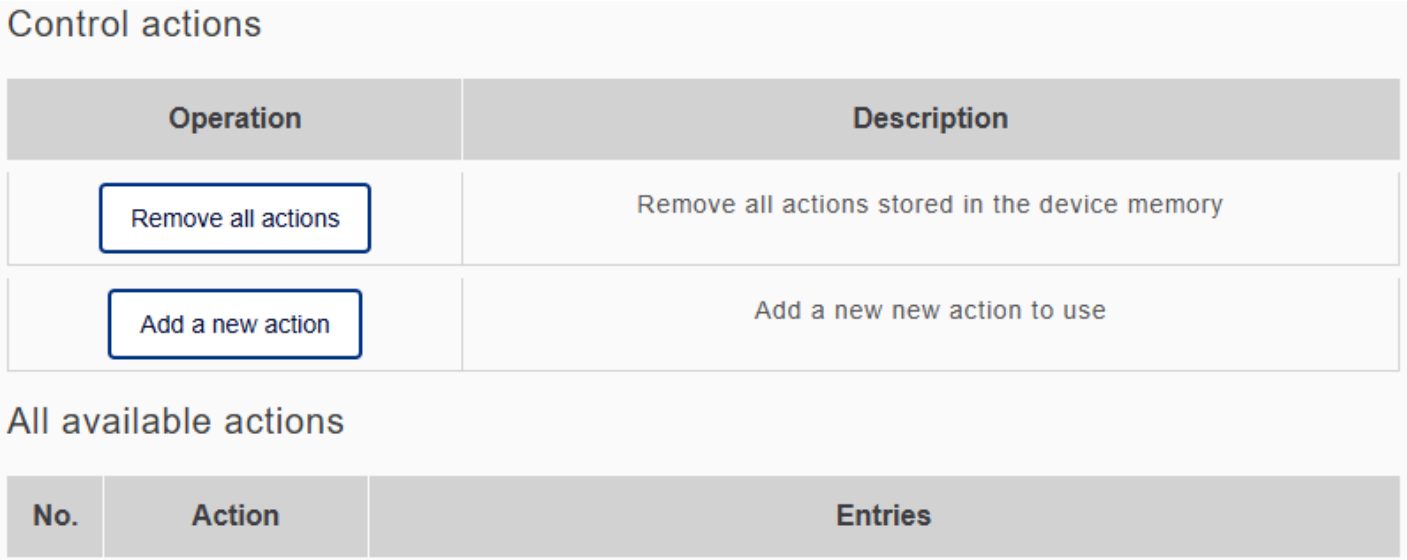
The actions are specific reactions of the module to received signals and input data, in accordance to the instructions set by the user. Many functions can be conducted by multiple methods depending on preferences and needs.



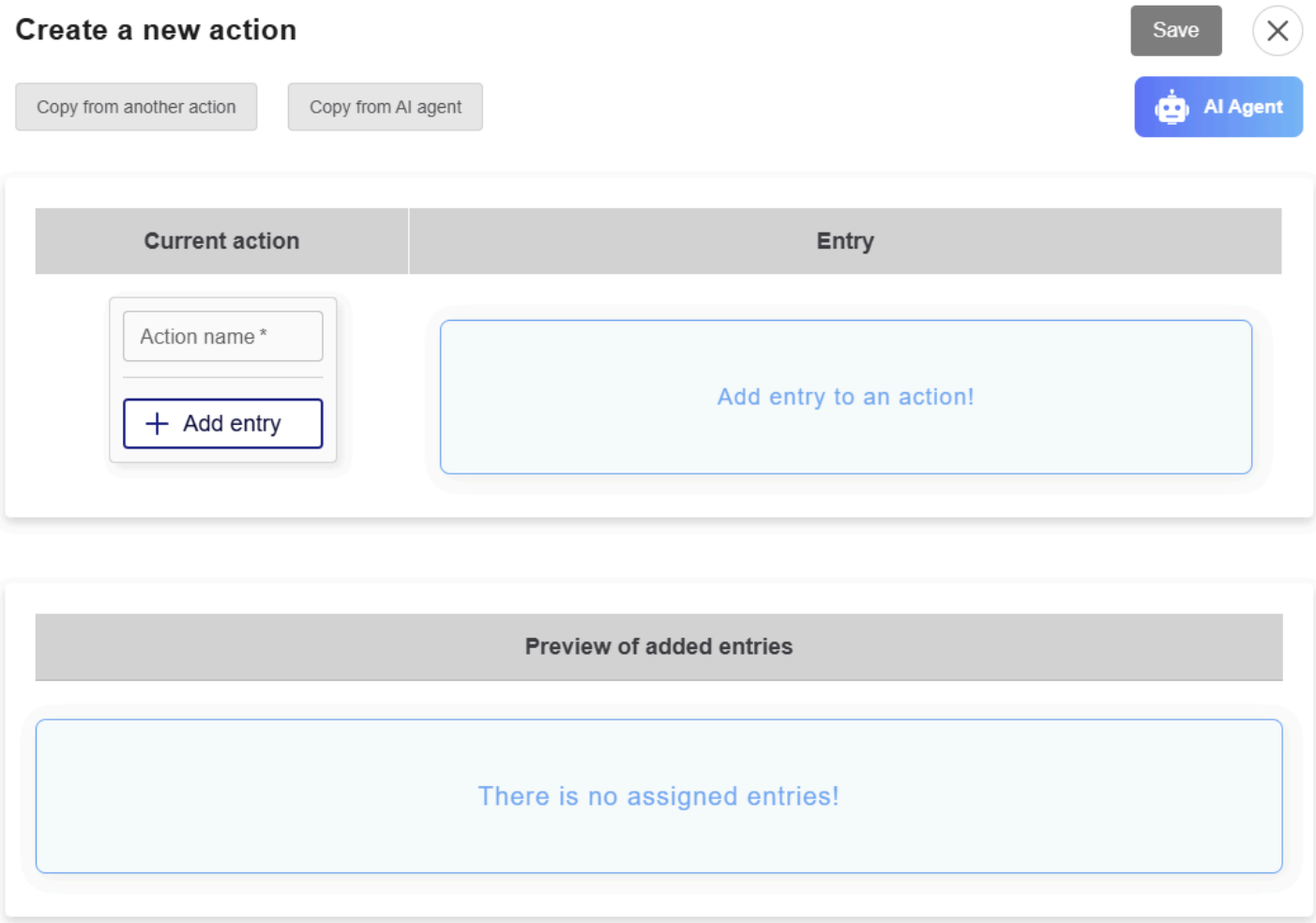
All

This tab allows preview and management of defined actions supported by the device.

Control Actions window



- **Remove all actions** – Removes all actions defined in the device,
- **Add a new action** – A window displayed after clicking this button allows the user to define each parameter of the added action:



- **Copy from another action**,
- **Copy from AI agent** – Opens a new window, to which the Inveo AI - generated string can be entered,
- **Action name**.

Clicking **Add entry** allows for communication protocol selection and further configuration.

After the programmed action's details are configured, click **Add**. It's possible to configure multiple actions for one event. After defining all required settings, save the changes using .

All available actions window

All defined and system actions are visible in this window. Each of them can be:

- Amended –
- Tested – Try
- Deleted –

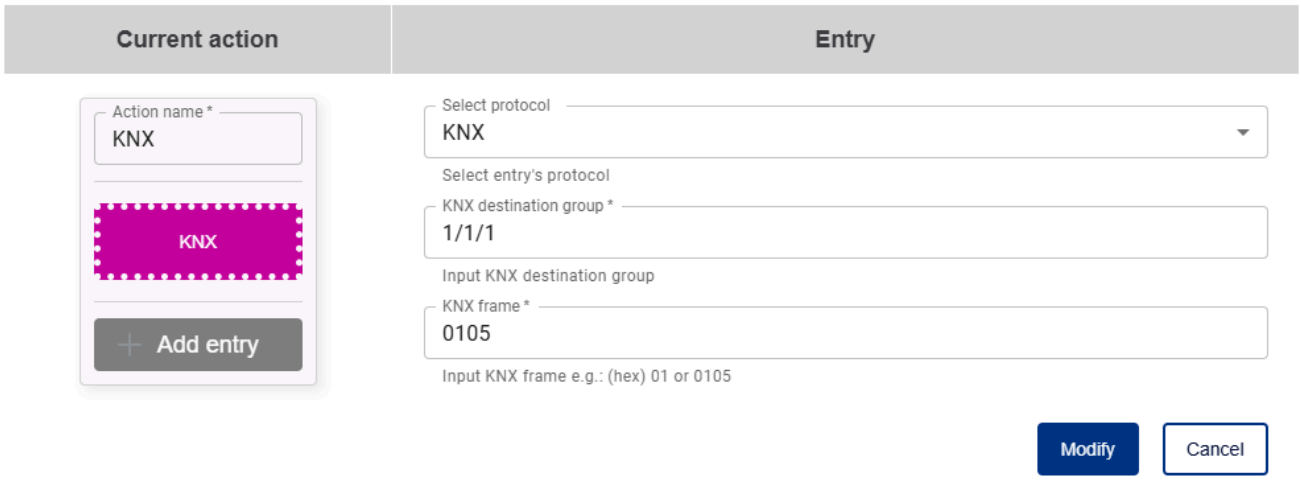
Protocol configuration

KNX

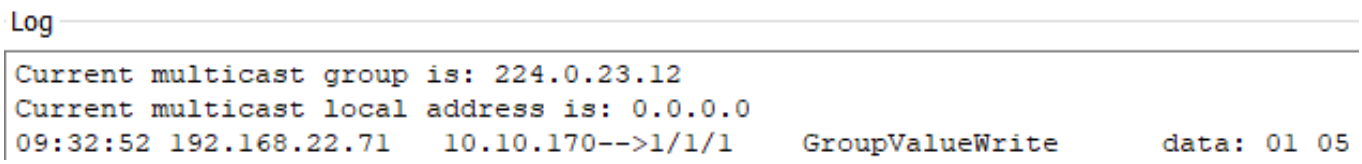
- **KNX destination group**,
- **KNX frame** – KNX frame syntax in HEX format.

Example

- **KNX destination group: 1/1/1**
- **KNX frame: 0105**



Sent data:

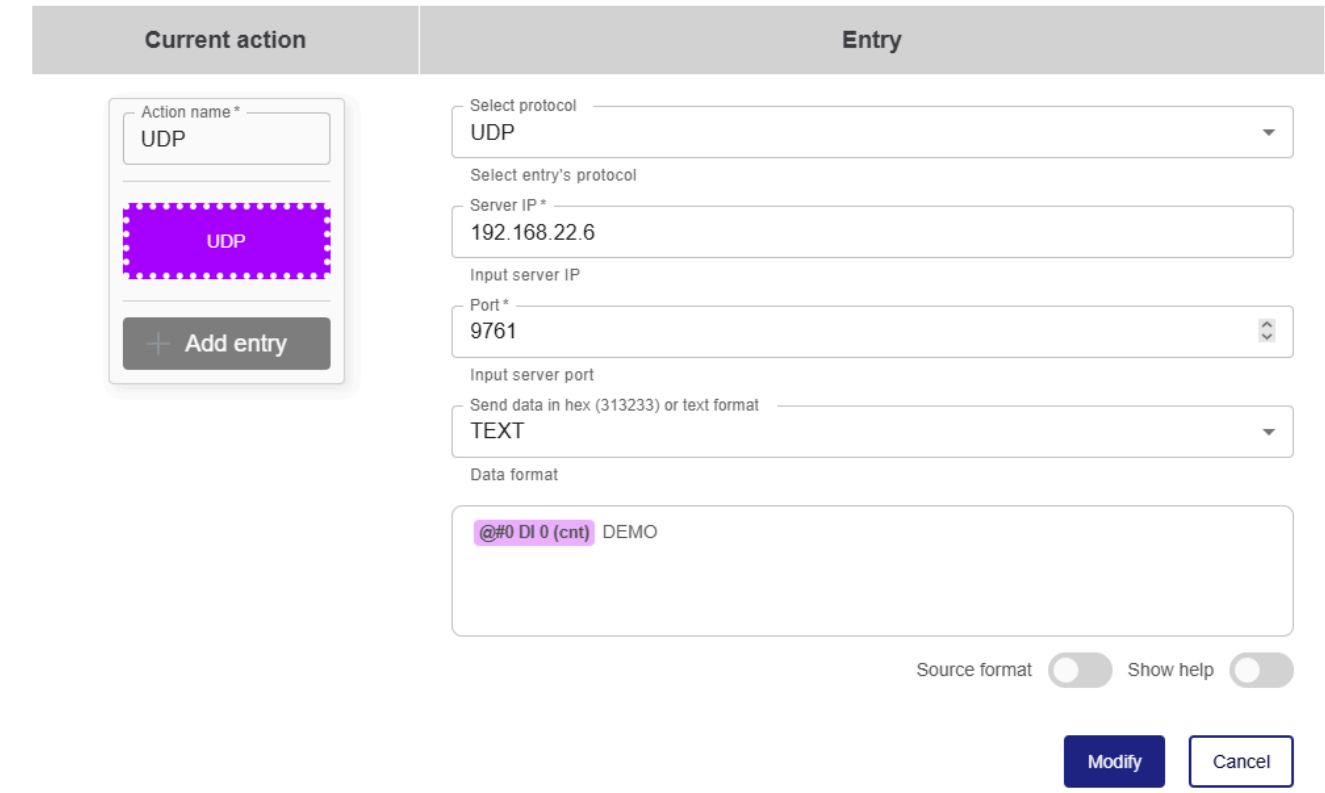


UDP

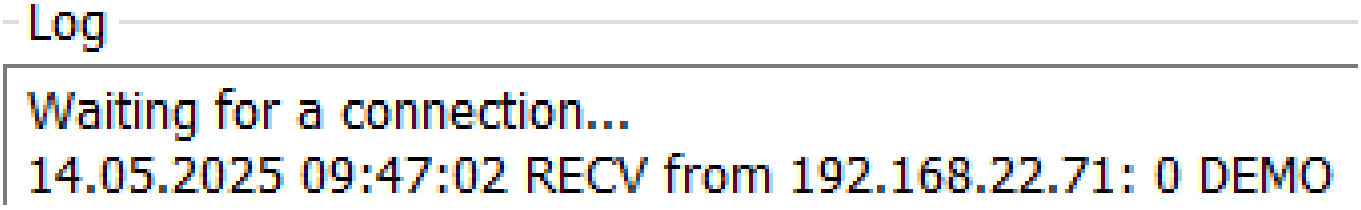
- **Server IP** – Target IP address,
- **Port** – Port, on which the target device is listening,
- **Format** – Data format: **TEXT** or **HEX**,
- **Input data** – Command sent to the target device.

Example

- **Server IP: 192.168.22.6**
- **Port: 9761**
- **Format: TEXT**
- **Input data:** `%i[0].cnt% DEMO`



Sent data:

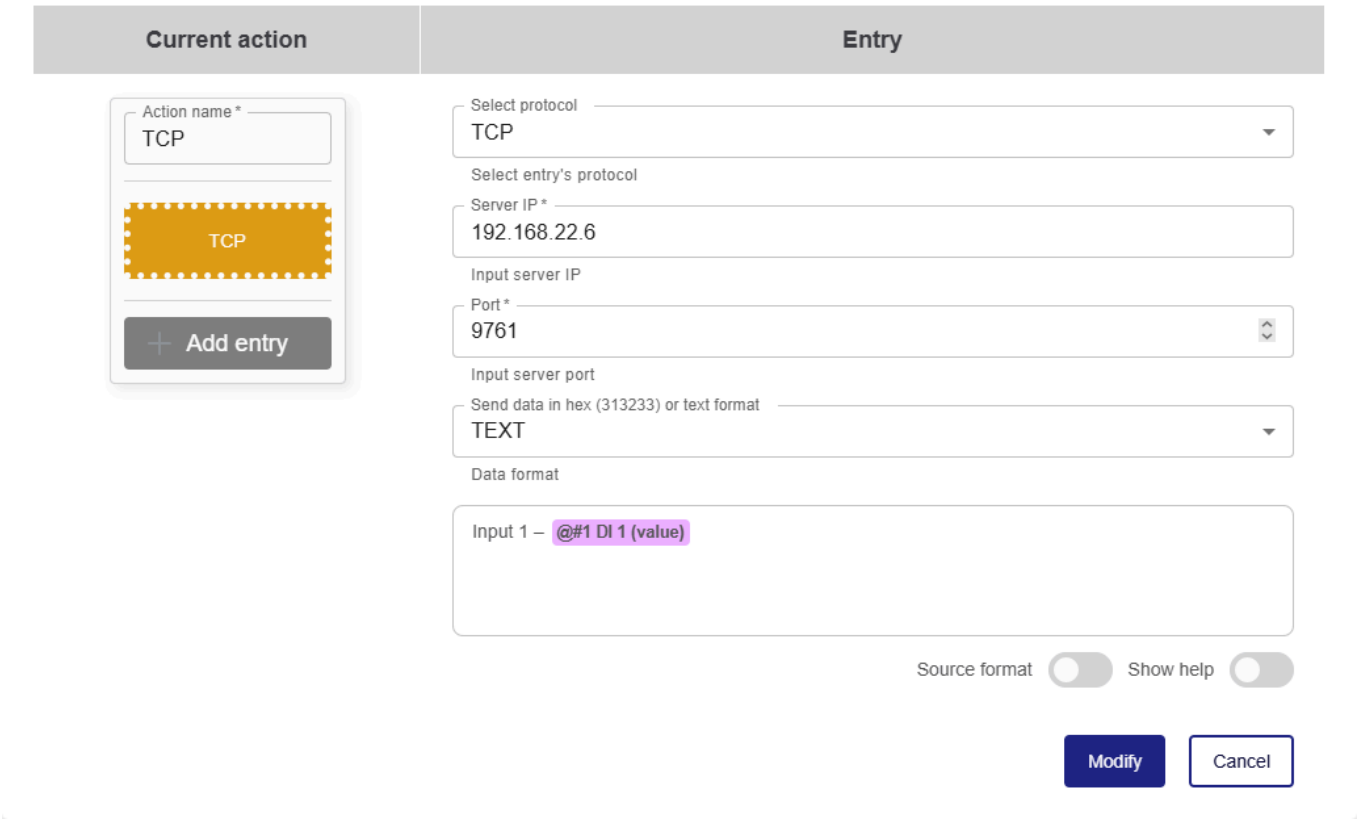


TCP

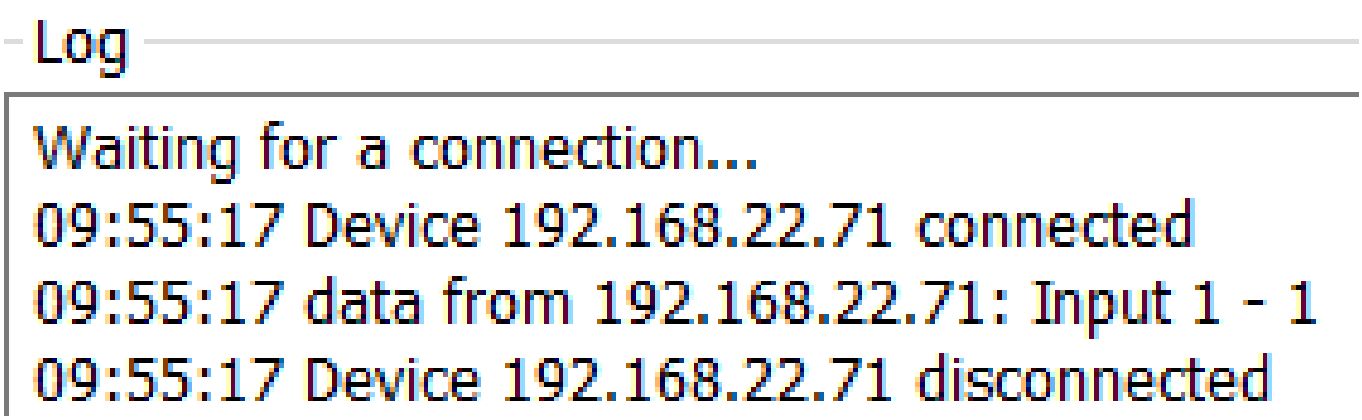
- **Server IP** – Target IP address,
- **Port** – Port, on which the target device is listening,
- **Format** – Data format: **TEXT** or **HEX**,
- **Input data** – Command sent to the target device.

Example

- **Server IP: 192.168.22.6**
- **Port: 9761**
- **Format: TEXT**
- **Input data:** `Input 1 – %i[1]%`



Sent data:



HTTP

Detailed protocol configuration is located in the **Services** tab.

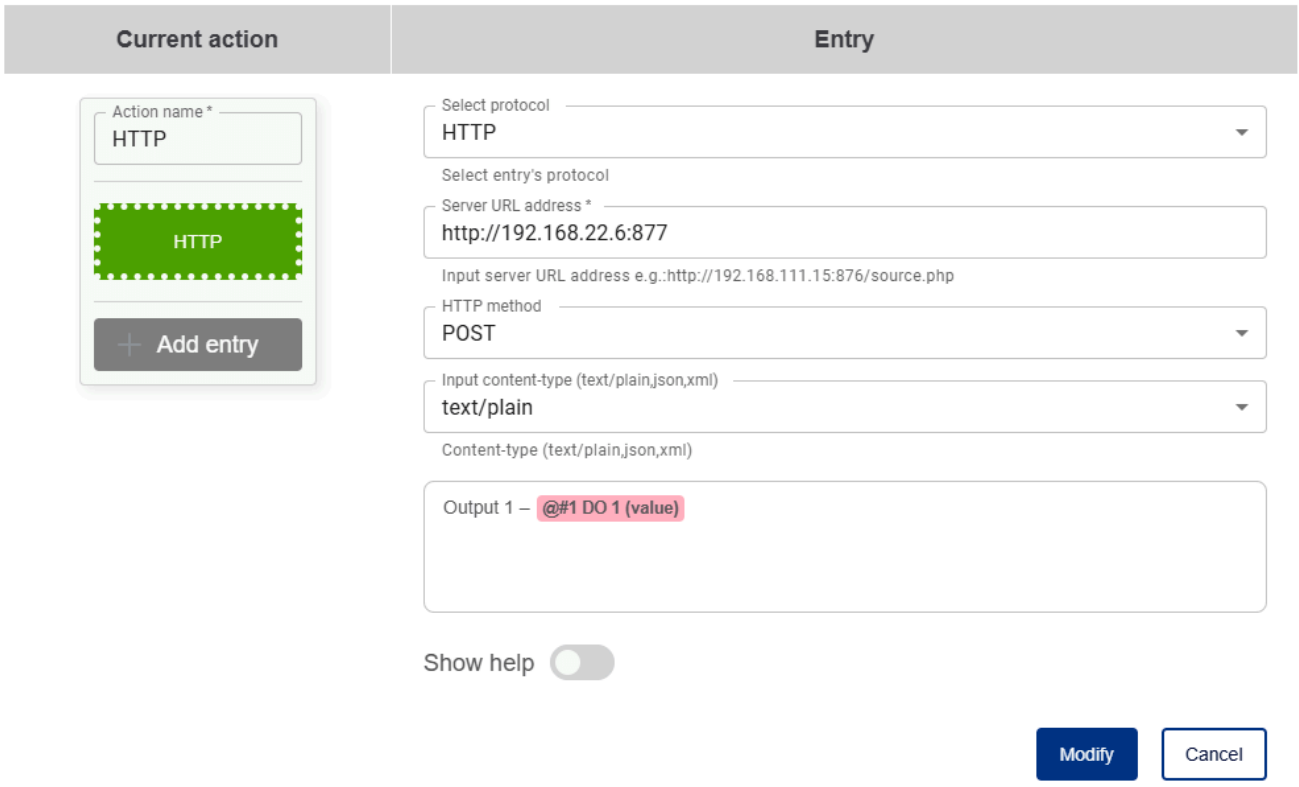
- **Server URL address** – Target URL address,

> **Warning**
>
> **The target URL must match the following convention:** `http://<IP_address>:<port>/<resource>`

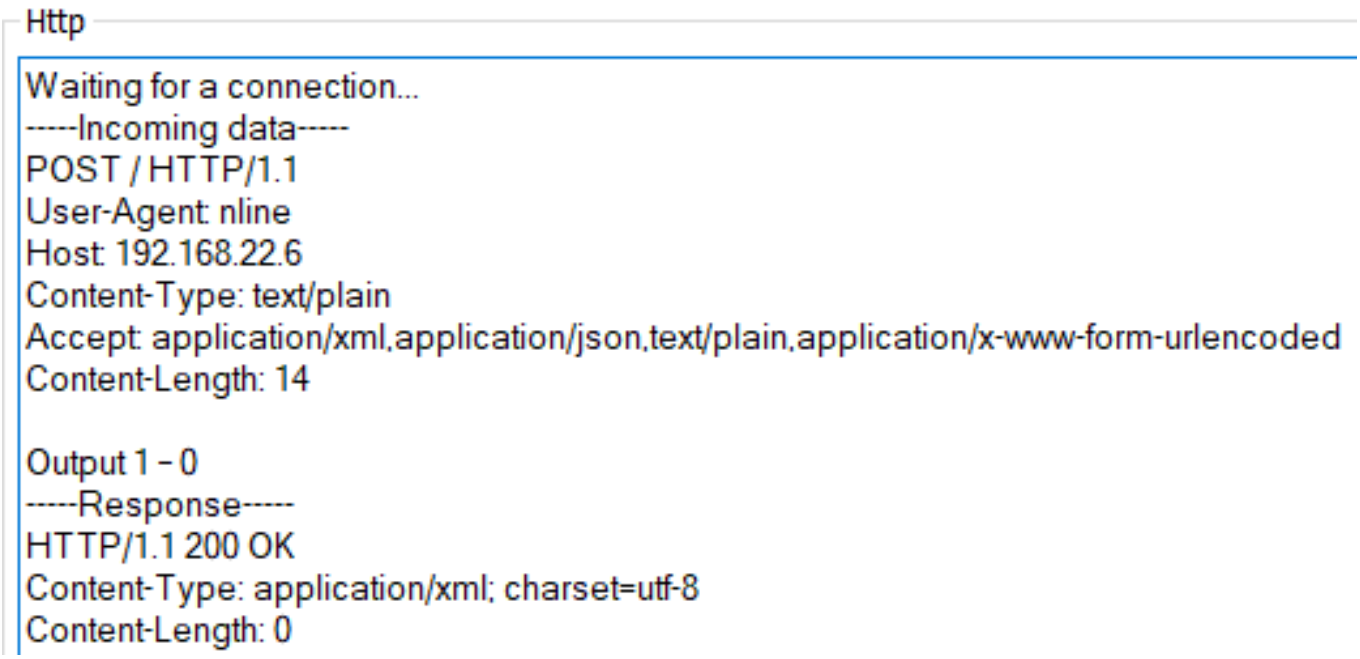
- **HTTP method:**
  - GET
  - POST
  - PUT
  - DELETE
- **Content-type:**
  - text/plain
  - application/x-www-form-urlencoded
  - application/json
  - application/xml
- **Input data** – Command sent to the target device.

Example 1

- **Server URL address:** `http://192.168.22.6:877`
- **HTTP method: POST**
- **Content-type: text/plain**
- **Input data:** `Output 1 – %o[1]%`



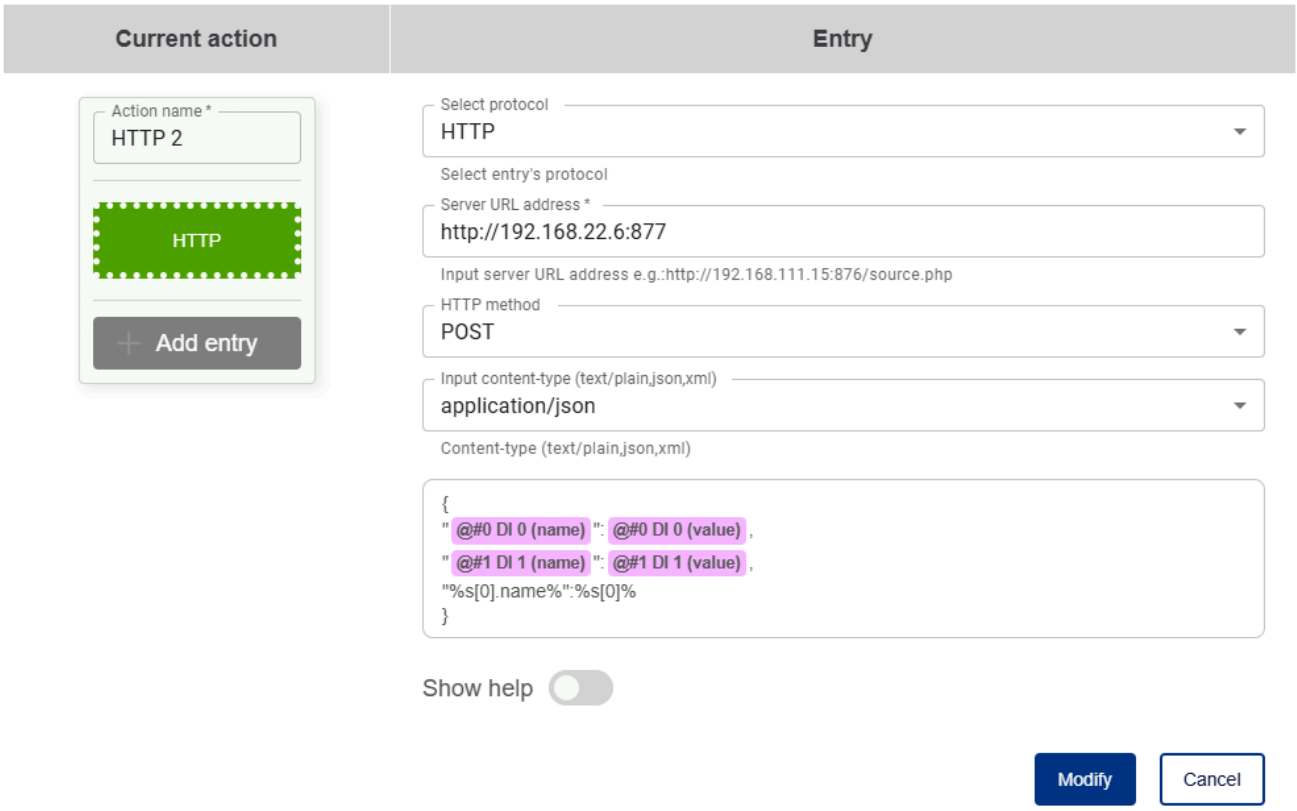
Sent data:



Example 2

- **Server URL address:** `http://192.168.22.6:877`
- **HTTP method: POST**
- **Content-type: application/json**
- **Input data:**

```
{
"%i[0].name%":%i[0]%,
"%i[1].name%":%i[1]%,
"%s[0].name%":%s[0]%
}
```



Sent data:

Http

```
Waiting for a connection...
-----Incoming data-----
POST / HTTP/1.1
User-Agent: nline
Host: 192.168.22.6
Content-Type: application/json
Accept: application/xml,application/json,text/plain,application/x-www-form-urlencoded
Content-Length: 30

{"DI 0":0,"DI 1":1,"s0":0}
-----Response-----
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 0
```

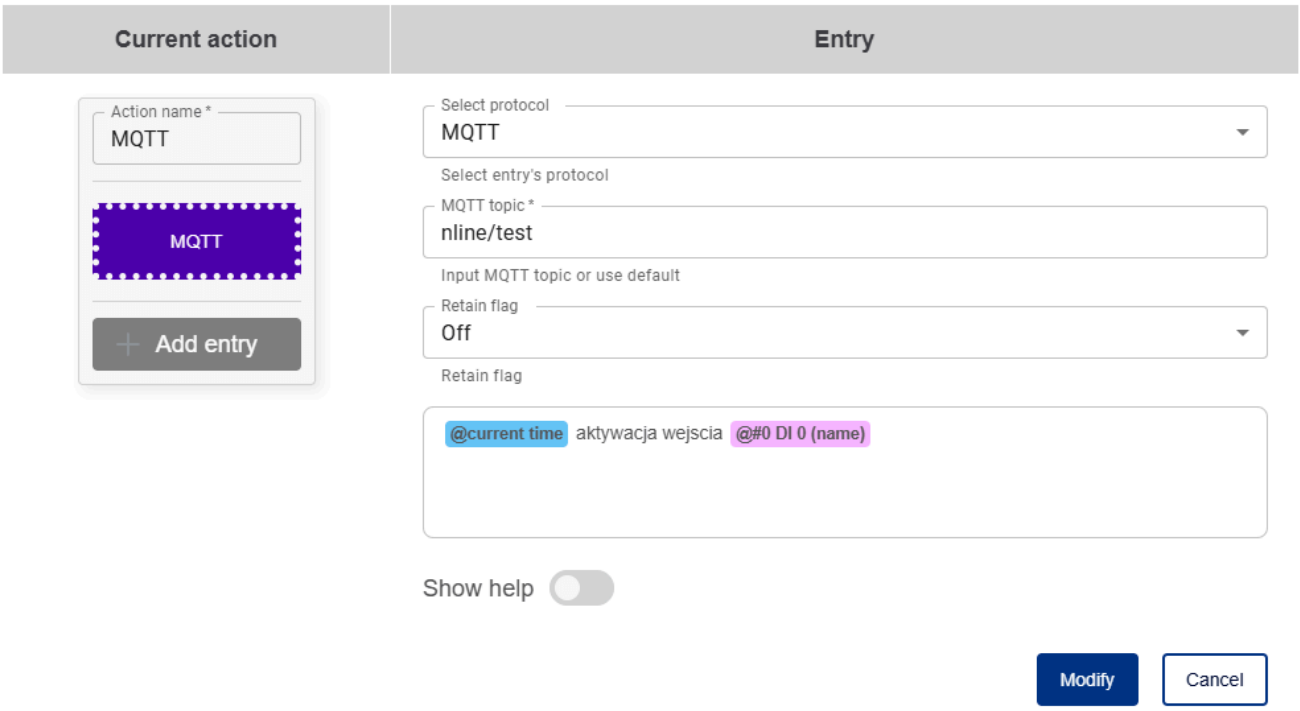
MQTT

Detailed protocol configuration is located in the **Services** tab.

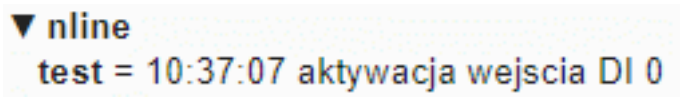
- **MQTT topic** – Topic, the device sends data to,
- **Retain flag** – Activated option causes the broker to retain last messages for topics the device sends data to,
- **Input data** – Message payload.

Example

- **MQTT topic:** `nline/test`
- **Retain flag: Off**
- **Input data:** `%time% input %i[0].name% activation`



Sent data:

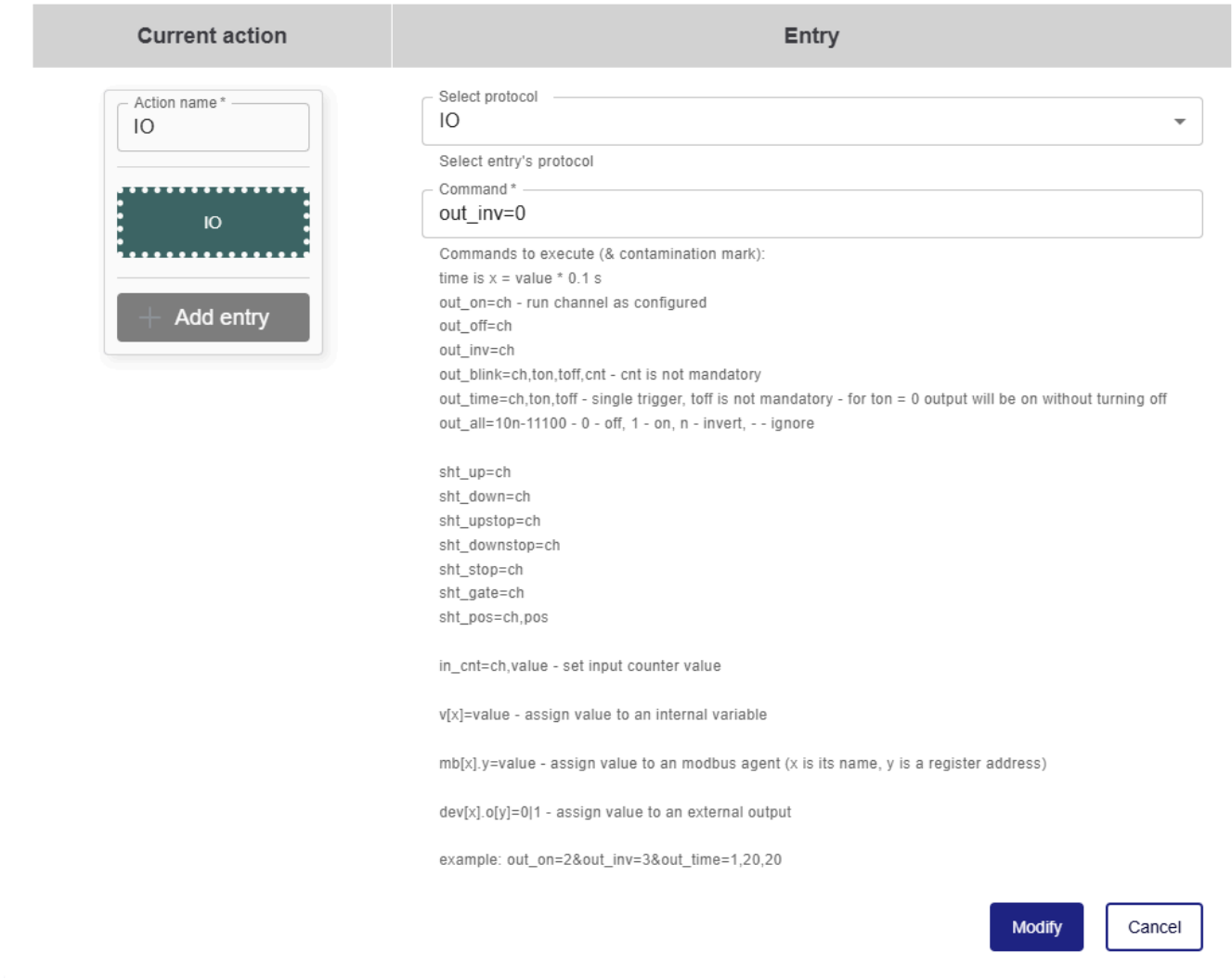


IO

- **Input command** – Command field:
  - `in_cnt=ch,val` – Set the `ch` channel counter value to `val`,
  - `out_on=ch` – Activate output channel `ch`,
  - `out_off=ch` – Deactivate output `ch`,
  - `out_inv=ch` – Output `ch` state invert,
  - `out_blink=ch,ton,toff,cnt` – Periodic output `ch` control. Parameters:
    - `ton` – Activation time (in seconds),
    - `toff` – Deactivation time (in seconds),
    - `cnt` – Number of activation cycles (not required).
  - `out_time=ch,ton,toff` – Activate output `ch` for the time defined in `ton`, after `toff` time passes. The `toff` parameter is not required – omitting it will activate the output without a delay,
  - `out_all=10n-11100` – Command defining the state of all available outputs. Each digit represents a subsequent output:
    - `1` – Activated,
    - `0` – Deactivated,
    - `n` – State invert,
    - `-` – No state change.
  - `sht_up=ch` – Open the shutter,
  - `sht_down=ch` – Close the shutter,
  - `sht_upstop=ch` – Open the shutter and stop,
  - `sht_downstop=ch` – Close the shutter and stop,
  - `sht_stop=ch` – Shutter stop,
  - `sht_gate=ch` – ,
  - `sht_pos=ch,pos` – Move shutter to the specified position,
  - `v[x]=value` – Assign a `value` to the variable `x`,
  - `mb[x].y=value` – Write the `value` to the Modbus register `y`, of the agent `x`,
  - `dev[x].o[y]=0|1` – Assign 0 or 1 to the external output `y` in device `x`.

Example

- **Input command:** `out_inv=0`



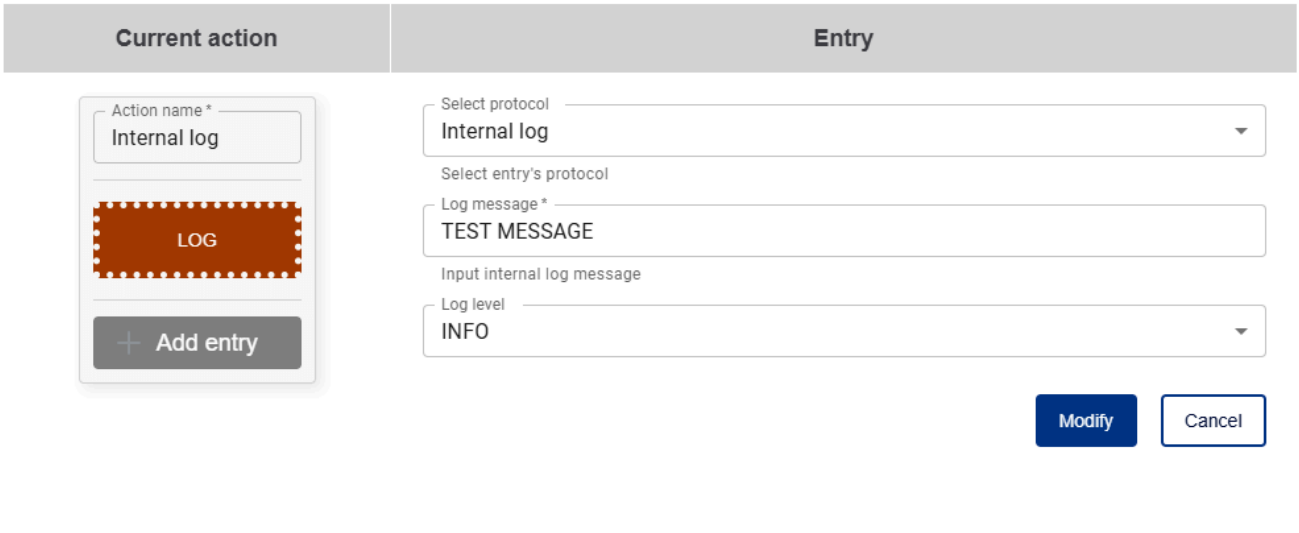
This setting will invert the state of output 0.

Internal log

- **Log message** – Message syntax,
- **Log level:**
  - DEBUG
  - INFO
  - WARNING
  - ERROR

Example

- **Log message:** `TEST MESSAGE`
- **Log level: INFO**



Sent data:

```
[2025-05-14 10:45:25] INFO LOG - TEST MESSAGE
```

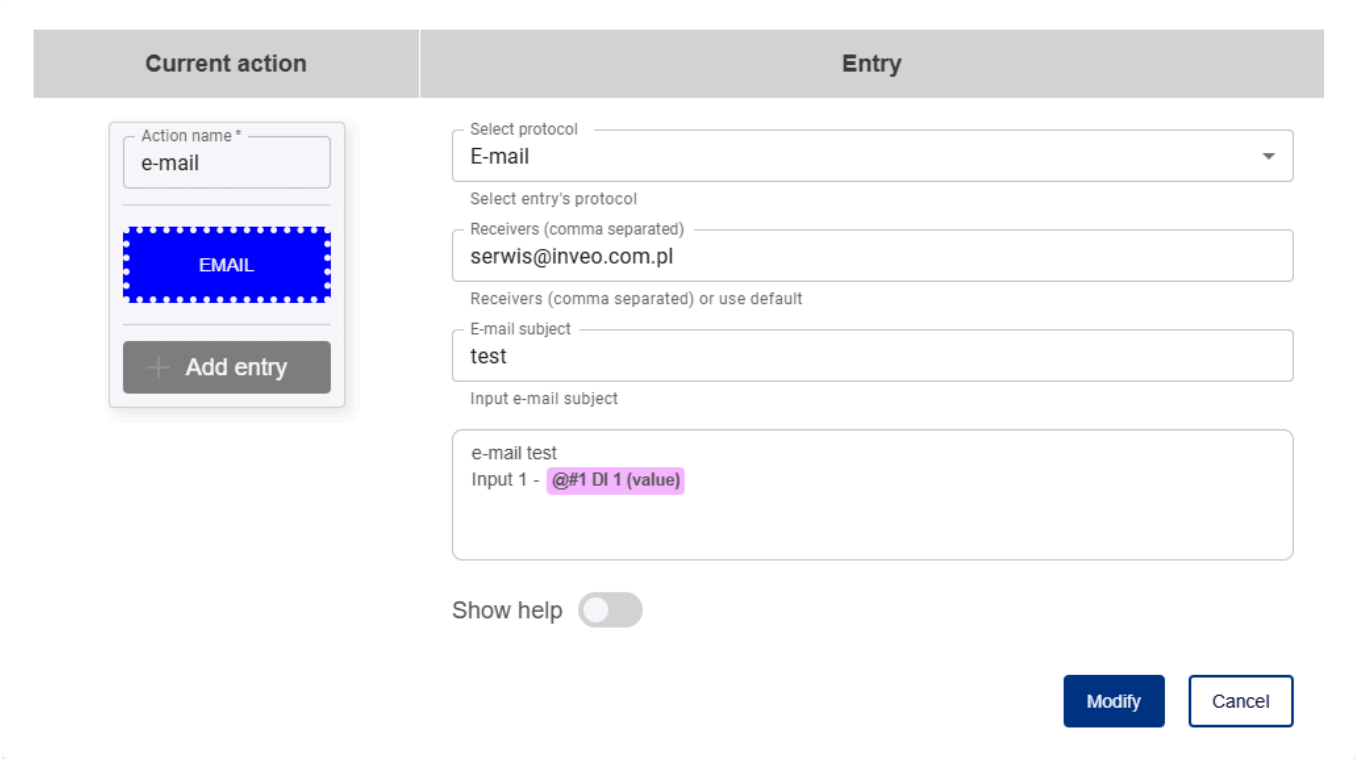
E-mail

Detailed protocol configuration is located in the **Services** tab.

- **Receivers (comma separated)** – E-mail recipients (comma separated),
- **E-mail subject** – E-mail message subject,
- **E-mail message** – E-mail message contents.

Example

- **Receivers (comma separated):** `test1@test.com`
- **E-mail subject:** `test`
- **E-mail message:** `e-mail test Input 1 - %i[0]%`



Those settings will cause an e-mail with the topic of `test`, with the contents of `e-mail test Input 1 -` and current input state to be sent to test1@test.com.

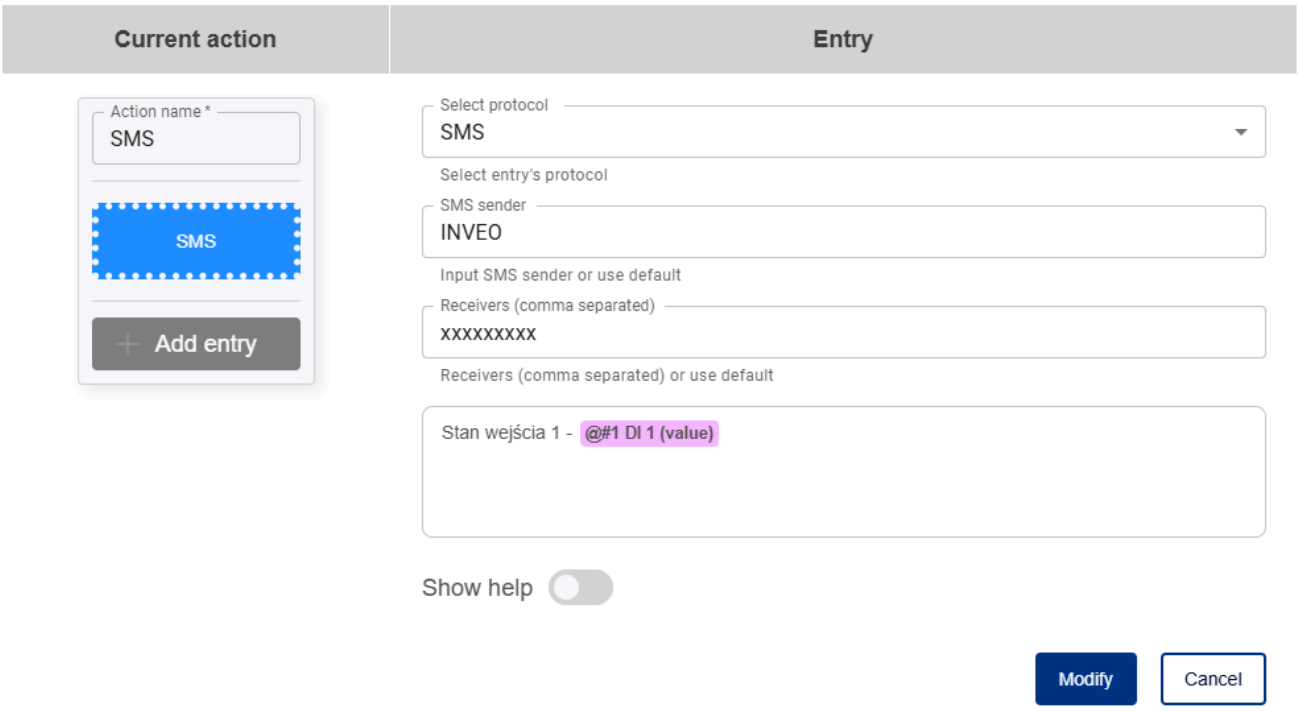
SMS

Detailed protocol configuration is located in the **Services** tab.

- **SMS sender**,
- **Receivers (comma separated)** – SMS recipients (comma separated),
- **SMS message** – SMS message contents.

Example

- **SMS sender: INVEO**
- **Receivers (comma separated): xxx xxx xxx**
- **SMS message:** `Output 1 state - %i[1]%`



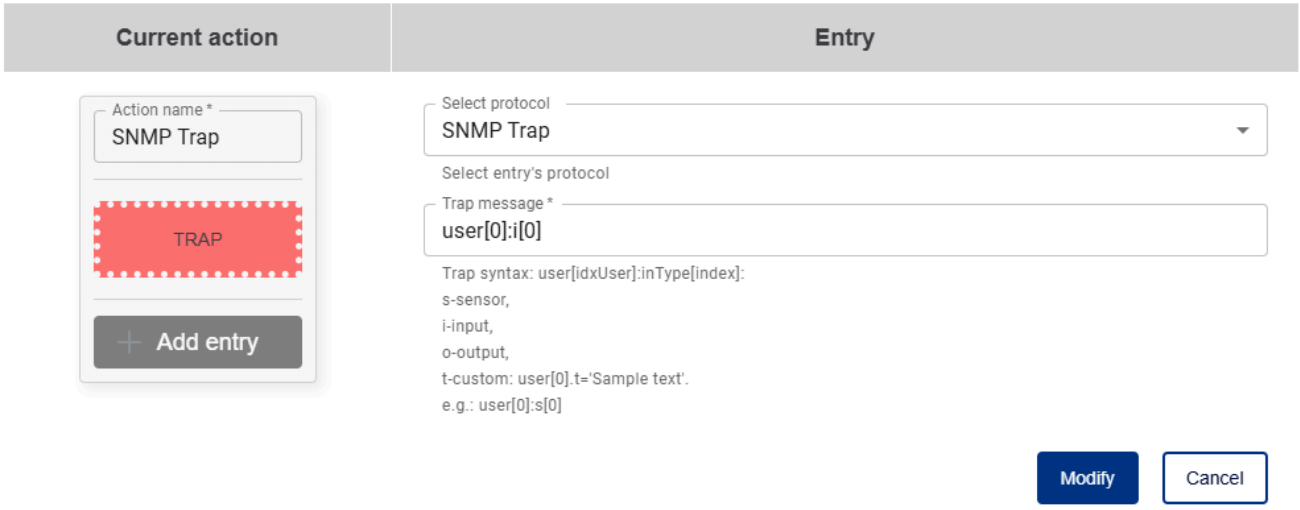
Those settings will cause an SMS from INVEO**, with the contents of** `Output 1 state - (output state)` **to be sent to the phone number xxx xxx xxx.**

SNMP Trap

- **Trap message** – Message syntax.

Example

- **Trap message:** `user[0]:i[0]`

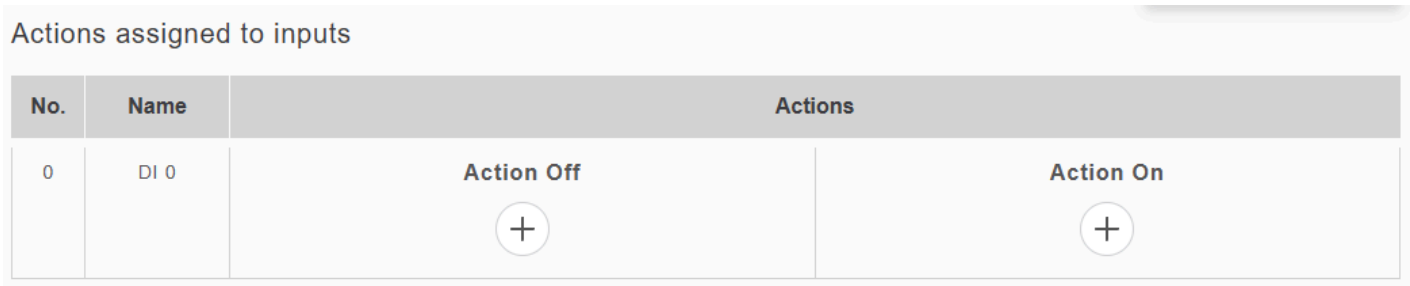


Sent data:

Description	Source	Time	Severity
.1.3.6.1.4.1.42814.10.1.1.100.0.5	192.168.22.71	2025-05-14 10:41:34	

Inputs

Defined actions can be configured and assigned to specific inputs in this tab.



Depending on the input operation mode (defined in **I/O Settings / Inputs**), various forms of actions triggering the assigned action are available.

Selected action mode	Available action forms	Action that will trigger the assigned action
Standard	Action Off	Action is triggered by input deactivation
	Action On	Action is triggered by input activation
Hold	Action Off	Action is triggered by input deactivation - input was activated for a time shorter than defined by the user in **Parameter**
	Action On	Action is triggered by input activation
	Action Hold On	Action is triggered by input activation for a time longer than defined by the user in **Parameter**
	Action Hold Off	Action is triggered by input deactivation – input was activated for a time longer than defined by the user in **Parameter**
Cnt	Action I	Action is triggered by the specified number of activations in a time period defined by the user in **Parameter**
	Action II	Action is triggered by the specified number of activations in a time period defined by the user in **Parameter**
	Action III	Action is triggered by the specified number of activations in a time period defined by the user in **Parameter**
	Action IV	Action is triggered by the specified number of activations in a time period defined by the user in **Parameter**
Toggle	Action I	Subsequent pulses on the input recall **Action I** and **Action II** alternately

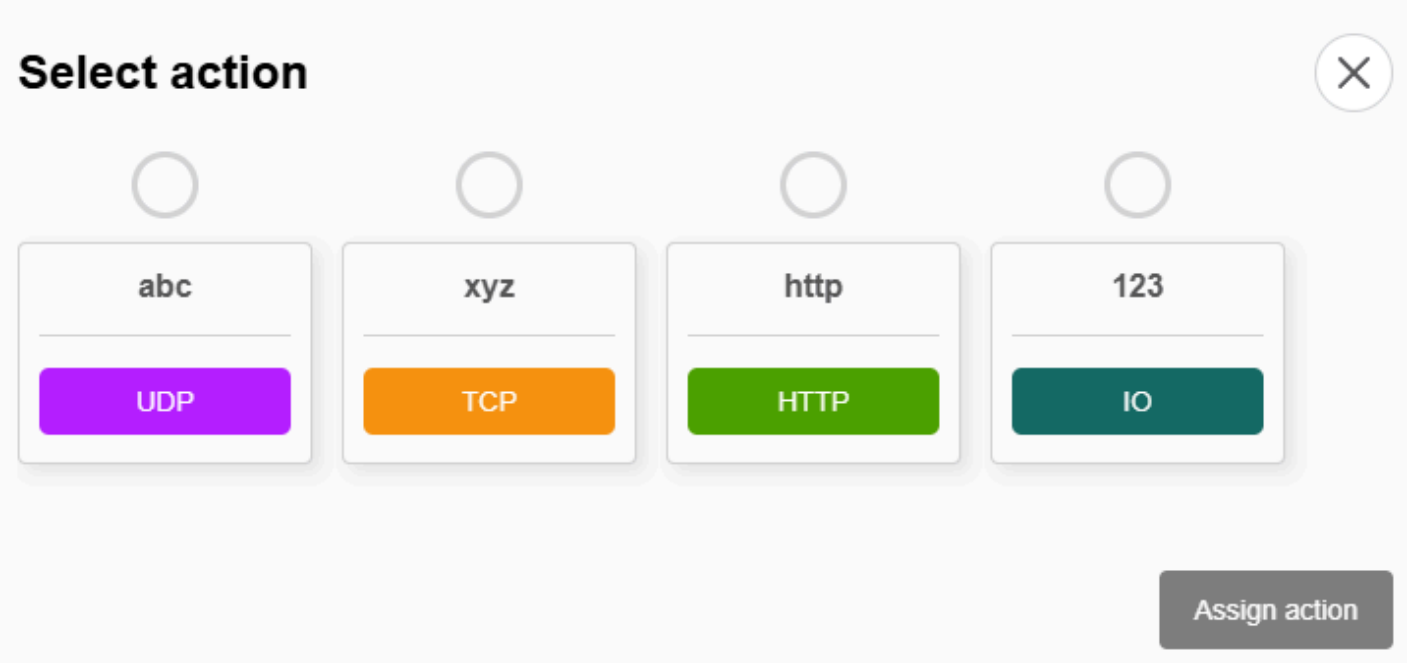
	Action II	
Freq	Action Off	Action is triggered by input activation with a frequency lower than defined by the user in **Parameter**
	Action On	Action is triggered by input activation with a frequency higher than defined by the user in **Parameter**

The **Go to the input actions** allows quick access to the **I/O Settings / Inputs** tab.

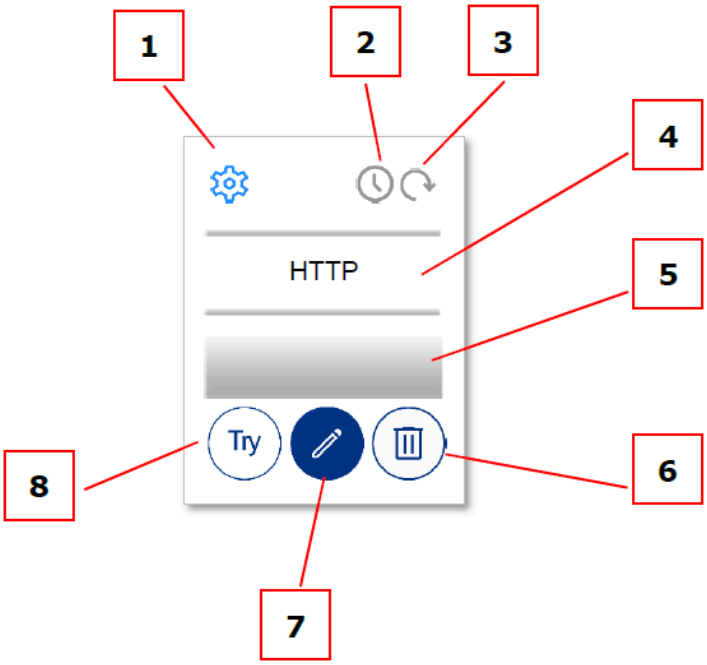
The  icon allows for step-by-step configuration.

Assigning actions

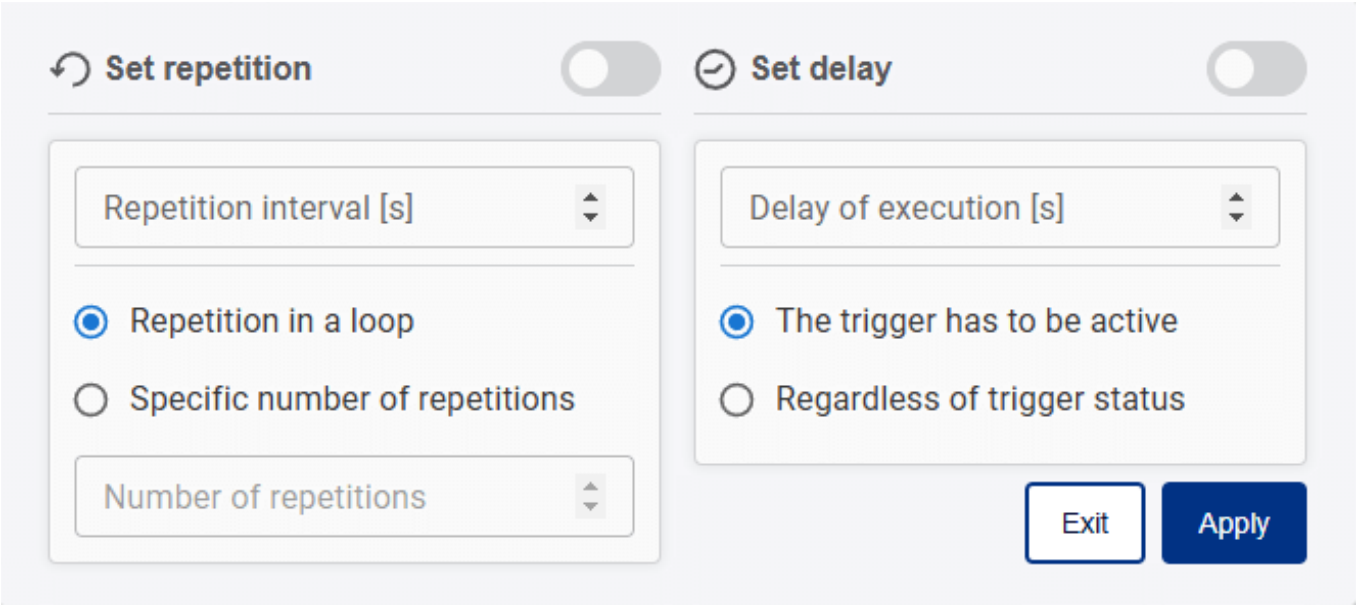
To assign an action to a chosen event, click **+**. A dialogue window will be displayed, where the desired action (defined previously in **All**) can be selected.



After an action is assigned, a window appears in the table:



1. Additional settings (repetition and delay) icon:



- **Repetition interval [s]** – Interval between executed actions. If this field is empty, the action will be executed only once,
  - **Repetiton in a loop** – Actions will be executed in a loop,
  - **Specific number of repetitions** – Actions will be repeated for a specified number of times,
    - **Number of repetitions**,
- **Delay of action execution [s]**:
  - The trigger has to be active – The action will be delayed only when the trigger is active,
  - Regardless of trigger status – The action will be delayed regardless of the trigger state, Save the changes using **Apply**.

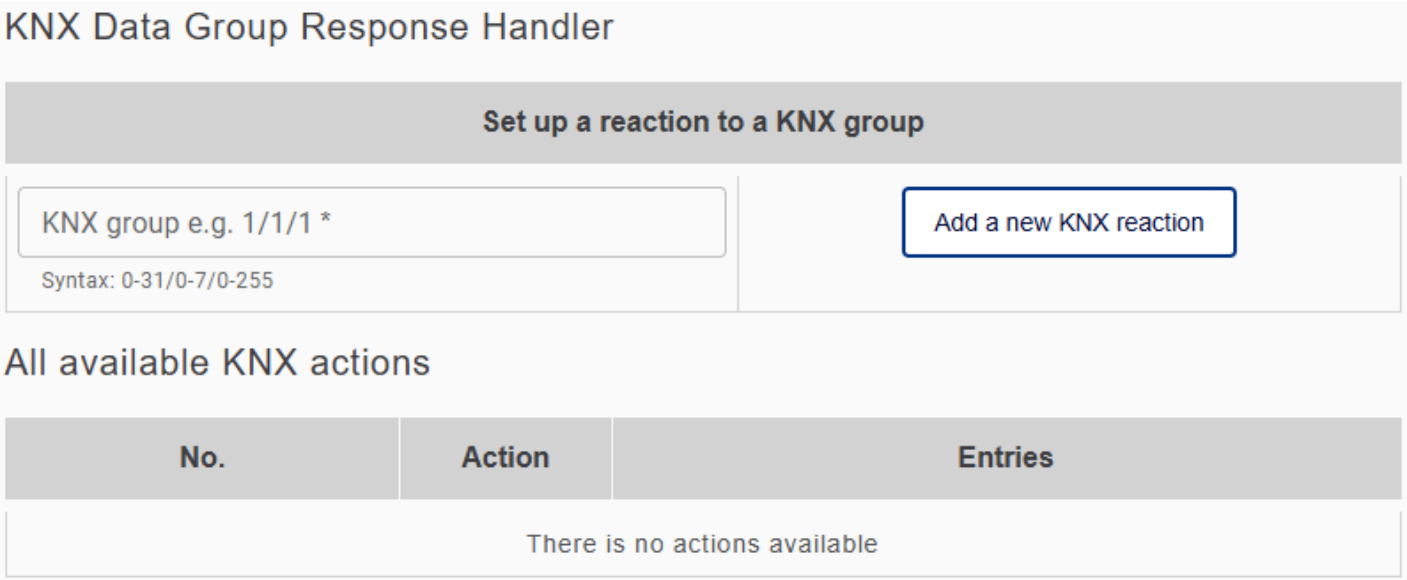
2. Action delay icon: **grey** – delay disabled, **green** – delay enabled,

3. Action repetition icon: **grey** – repetition disabled, **green** – repetition enabled,
4. Action name – defined by the user during addition or amendment of action settings,
5. Utilized communication protocol,
6. Bin icon – clicking it will remove action assignment,
7. Edit icon – clicking it will edit the action settings,
8. Try icon – clicking it will execute the action.

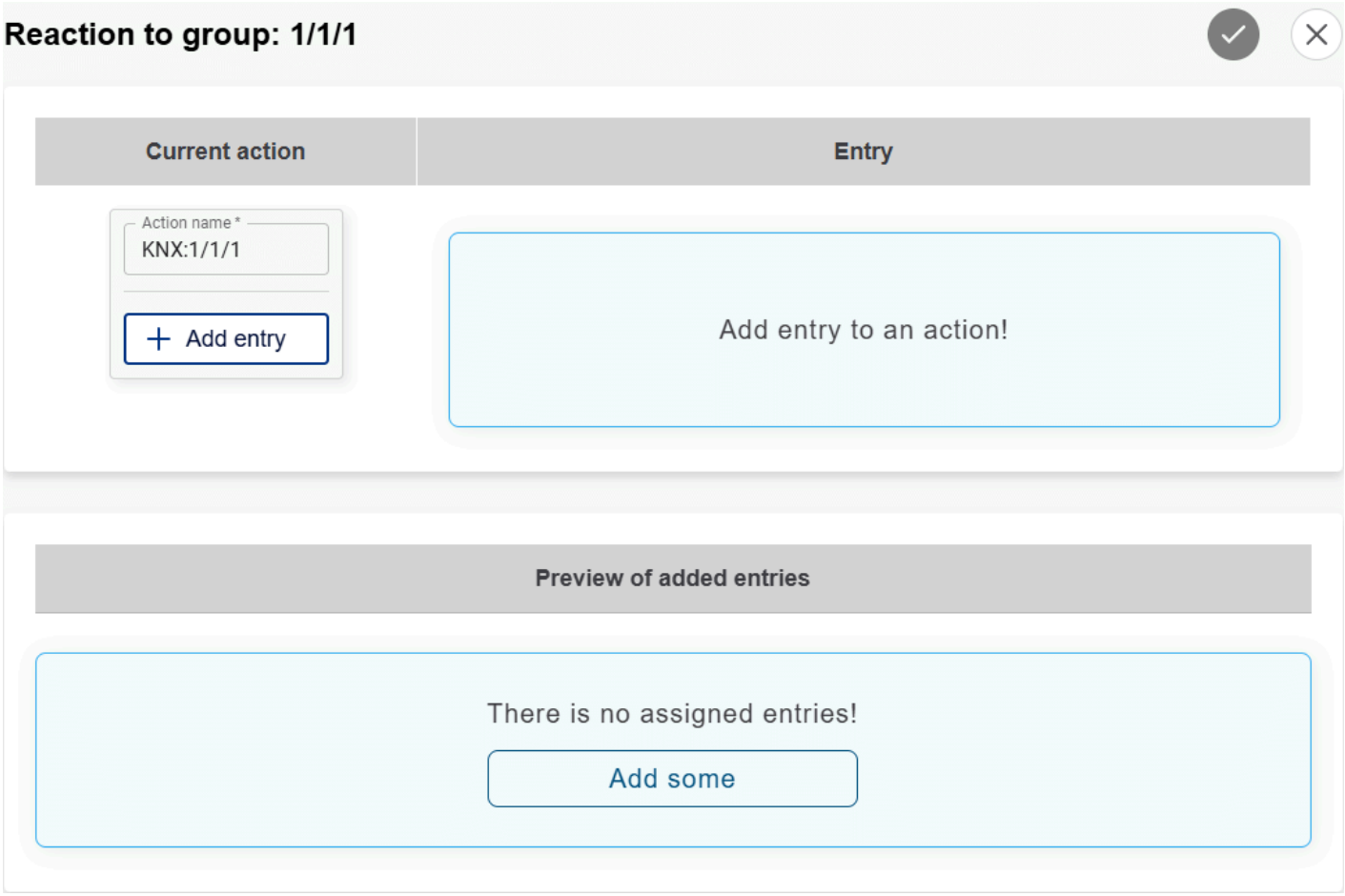
KNX

In the **Actions/KNX** tab, actions that are to be performed in response to data frames received by specified KNX groups can be defined.

When the module receives a data frame on a specified KNX group (eg. 1/1/1), it will automatically execute the defined actions. For example, depending on the configuration, it could be sending an e-mail to a specified address, running a predefined procedure or notifying the users about an event.



Firstly, the KNX group number should be entered. This will cause a dialogue window to appear, containing detailed settings. The next step is to select the communication protocol:



Clicking **Add entry** will enable communication protocol selection and further configuration.
- **Select protocol** – Protocol selection field – the parameteres of particular protocols have been described in detail in the **Control Actions Window** chapter.

After the details have been set, click **Add**. It's possible to configure several actions for a single event.

After all settings have been cofigured, apply the changes using .

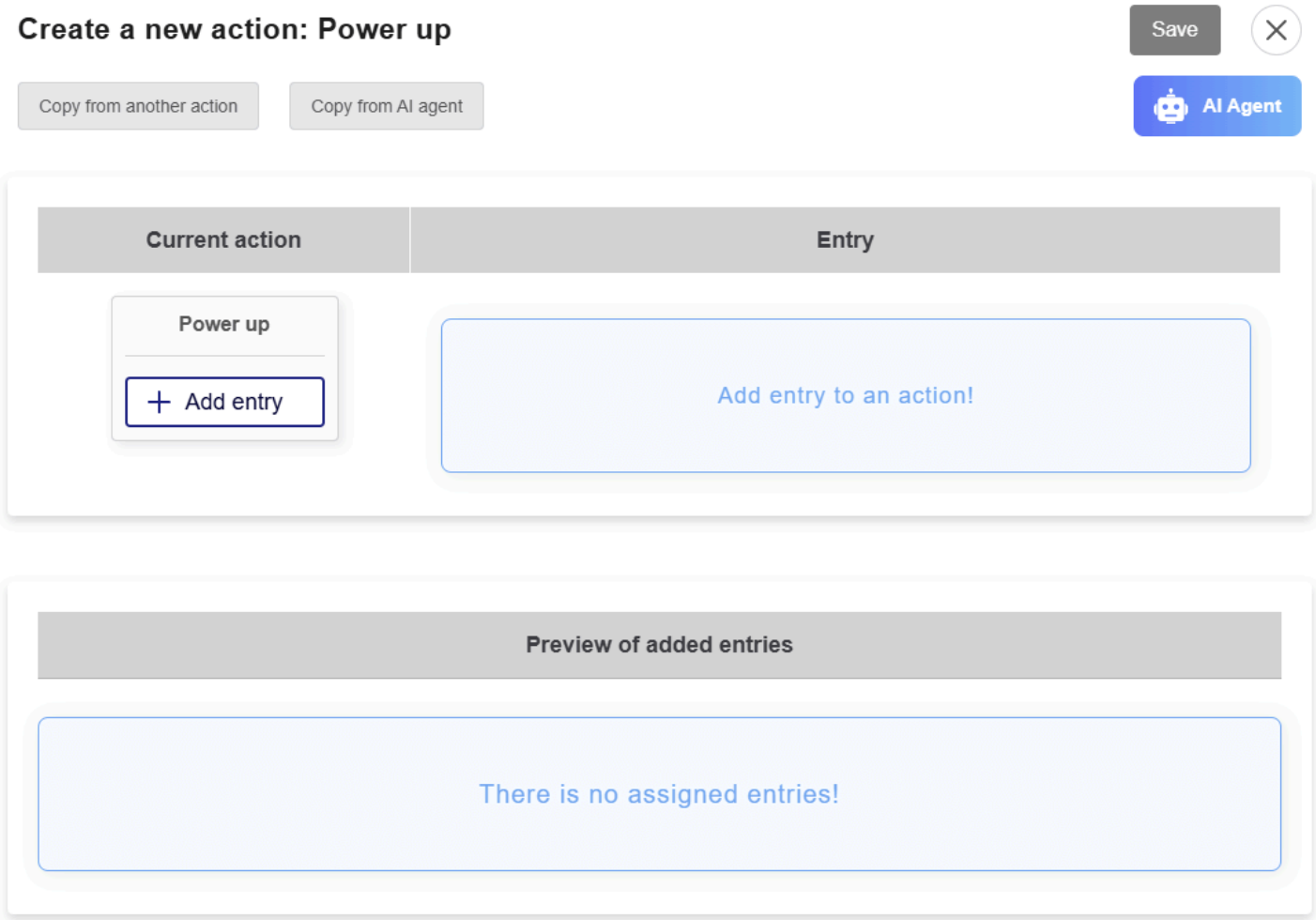
System

This tab allows the user to define system actions which the device will perform in the following events:



- **Power up** – Device power supply restore,
- **Ethernet up** – Ethernet access gained,
- **Ethernet down** – Ethernet access lost,
- **Wi-Fi up** – Wi-Fi access gained,
- **Wi-Fi down** – Wi-Fi access lost,
- **Modbus safe mode**.

Click **+** to assign the action to a desired event. A new dialogue window will appear:



Clicking **Add entry** will enable communication protocol selection and further configuration.

- **Select protocol** – The parameters of each protocol have been described in detail in the **Control Actions window** section.

After the programmed action's details are configured, click **Add**. It's possible to configure multiple actions for one event.

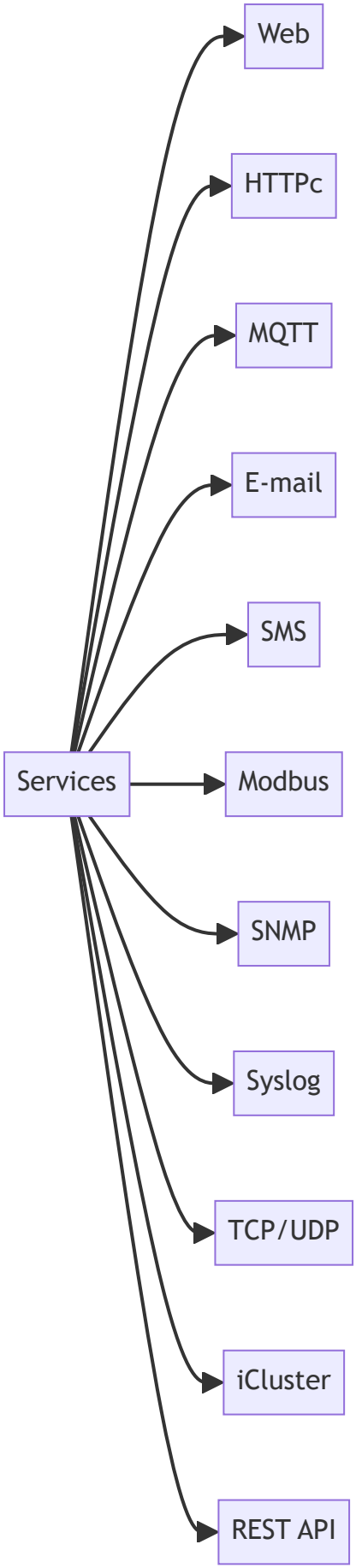
After defining all required settings, save the changes using .

Periodic

This tab allows the user to define periodic actions - performed in specified time intervals.

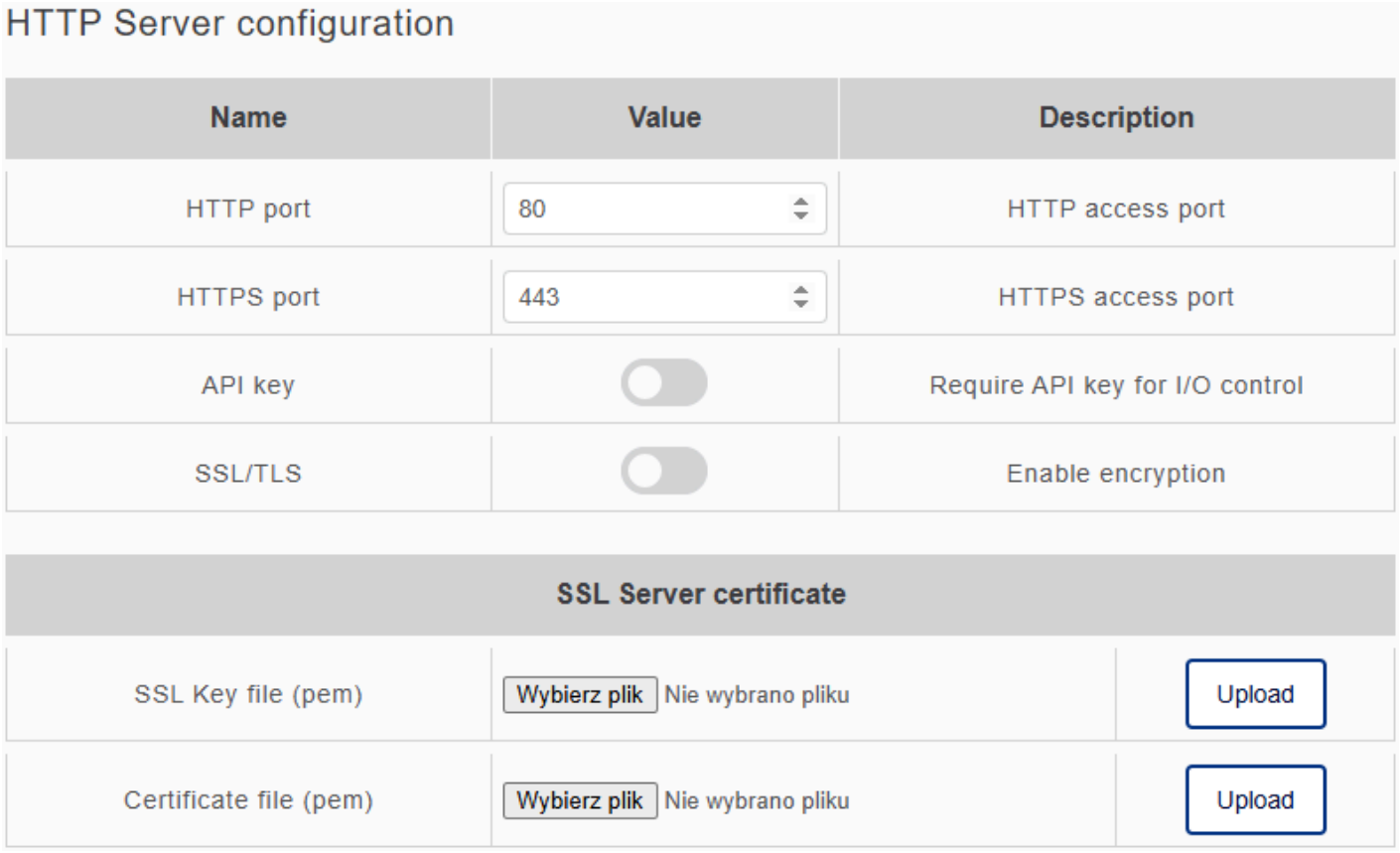
Communication protocols

In the **Services** tab, options enabling detailed communication protocol configuration options are located:


Web
HTTPc
MQTT
E-mail
SMS
Services
Modbus
SNMP
Syslog
TCP/UDP
iCluster
REST API

Web

In this section, the user can customize the device's network interface settings, manage resource access or modify network connection parameters.



- **HTTP Port**,
- **HTTPS port**,
- **API key** – Enable/disable API key requirement for I/O control,
- **SSL/TLS** – Enable/disable encryption,
- **Select Key file (pem)** – Allows upload of SSL server key (pem format),
- **Select CSR file (pem)** – Allows upload of CSR server key (pem format).

HTTPc

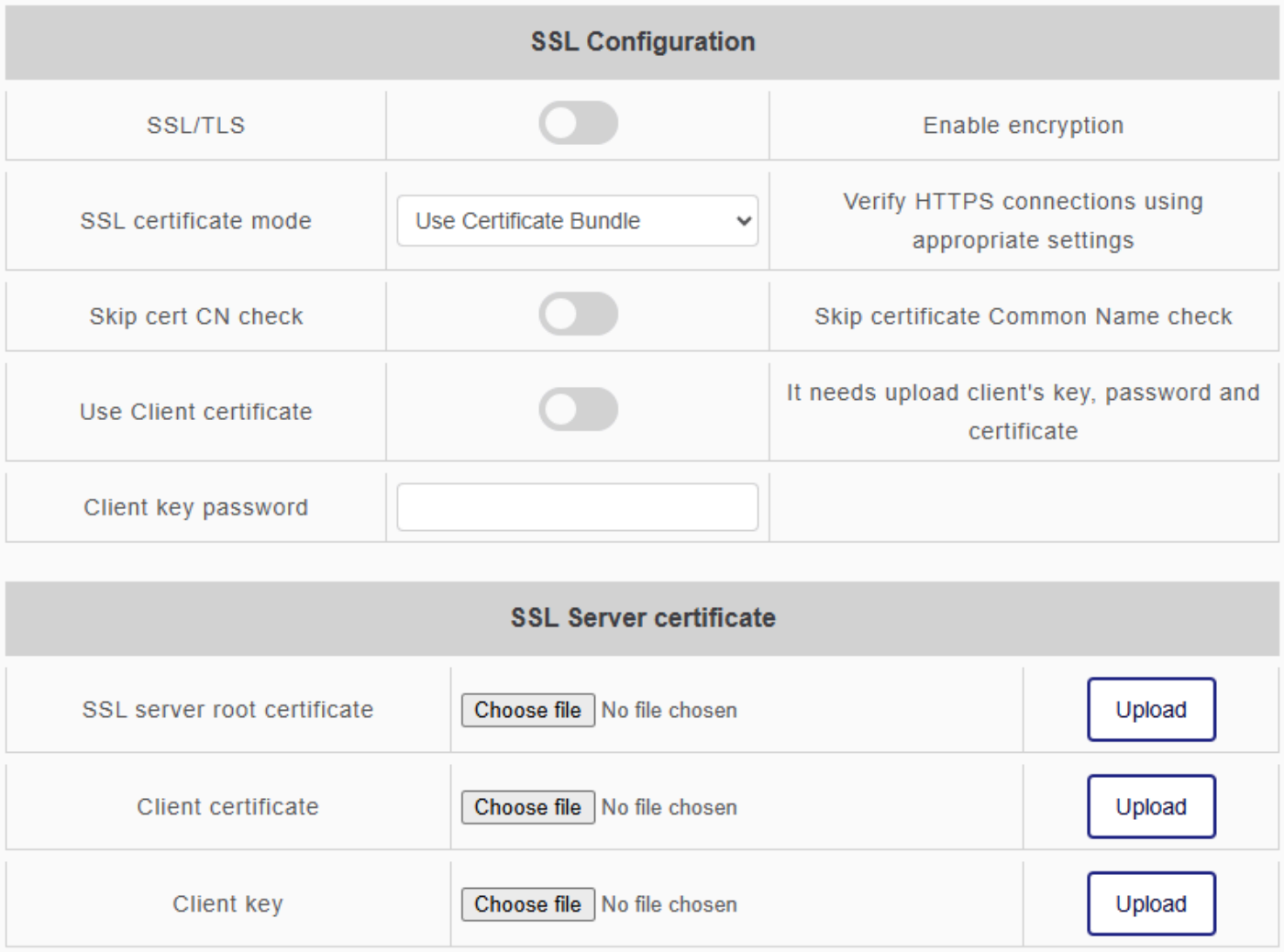
In this section, the device can be configured so it initiates HTTP connections with defined servers or services. URL addresses, request parameters and other connection details can be defined here. The device can send event data via HTTP/HTTPS, using GET or POST.

HTTP Client configuration

Name	Value	Description
HTTP Client		Enable HTTP Client
Server		Remote server address: IP or DNS-registered name
HTTP port	0	HTTP access port
HTTP Method	GET	HTTP request default method
Content type	text/plain	Content-type header
Resource	/	HTTP resource i.e. /resource.php
API key		Require API key for I/O control
Authentication to Server		
Authentication type	None	Select the authentication method for server communication
HTTP ping request		
HTTP ping request interval	0	Ping time in secs. 0-disable
HTTP ping request	HTTP ping requests to test the availability of web servers and APIs. These can be used in health testing or monitoring systems.	

Type @ and select variable...
You can mention variables like current time or module name

Source format Show help



- **HTTP Client** – Enable **HTTP Client** service,
- **Server** – HTTP server address, to which information will be sent,
- **HTTP Port** – Port, on which the HTTP server listens,
- **HTTP Method** – Message sending method: GET / POST / PUT / DELETE,
- **Content type**:
  - text/plain,
  - json,
  - xml,
  - form.
- **Resource** – Resource the module will refer to,
- **API key** – Enable/disable API key requirement for I/O control,
- **Authentication type** – Server authentication type,
  - **None**,
  - **Basic Authentication** – Username and password authorization:
    - **User**,
    - **Password**.

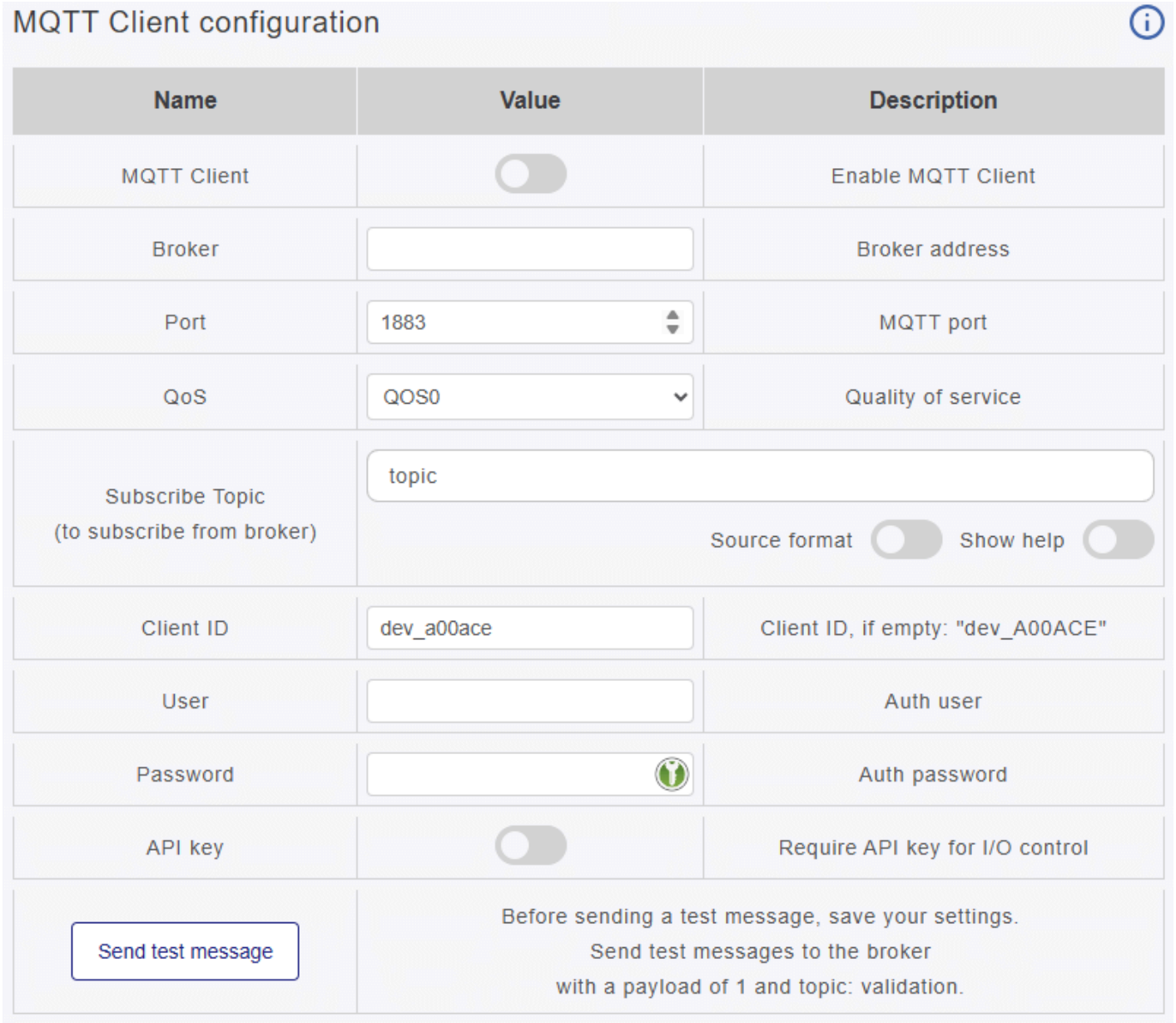
    - **X-API-KEY** – API key authorization:
        - **API Key**.
    - **Bearer Token**
        - **API Key**.
- **HTTP ping request interval**,
- **HTTP ping request** – Ping request syntax,
- **SSL/TLS** – Enable/disable encryption,
- **SSL certificate mode**:
    - **Use Certificate Bundle**,
    - **Use Uploaded Certificate**,
    - **Insecure! Disable SSL verification**,
- **Skip cert CN check** – Skip certificate common name check,
- **Use Client certificate**,
- **Client key password**,
- **SSL server root certificate**,
- **Client certificate**,
- **Client key**.

MQTT

This tab is used to configure MQTT broker communication parameters, allowing for data exchange with the publish-subscribe model. This allows the user to define key aspects such as topics, server address, port, and other connection parameters.
The device sends data to the server every minute and every time a value change occurs.

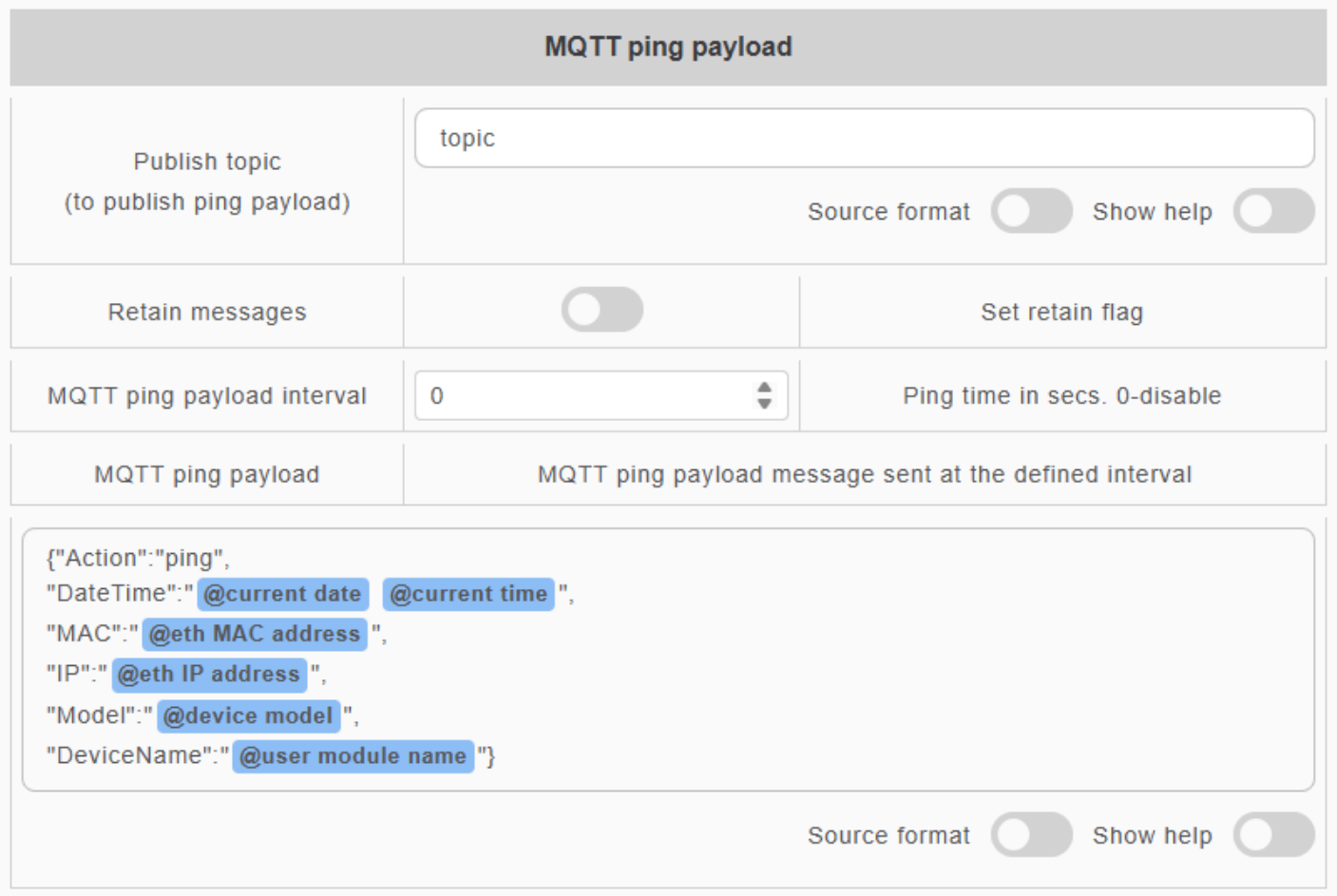
The data stream can be encrypted. After a connection with the MQTT broker is established, the users can subscribe to the data coming from the device.

There is no limit in the amount of subscribers that can receive data from a single device.



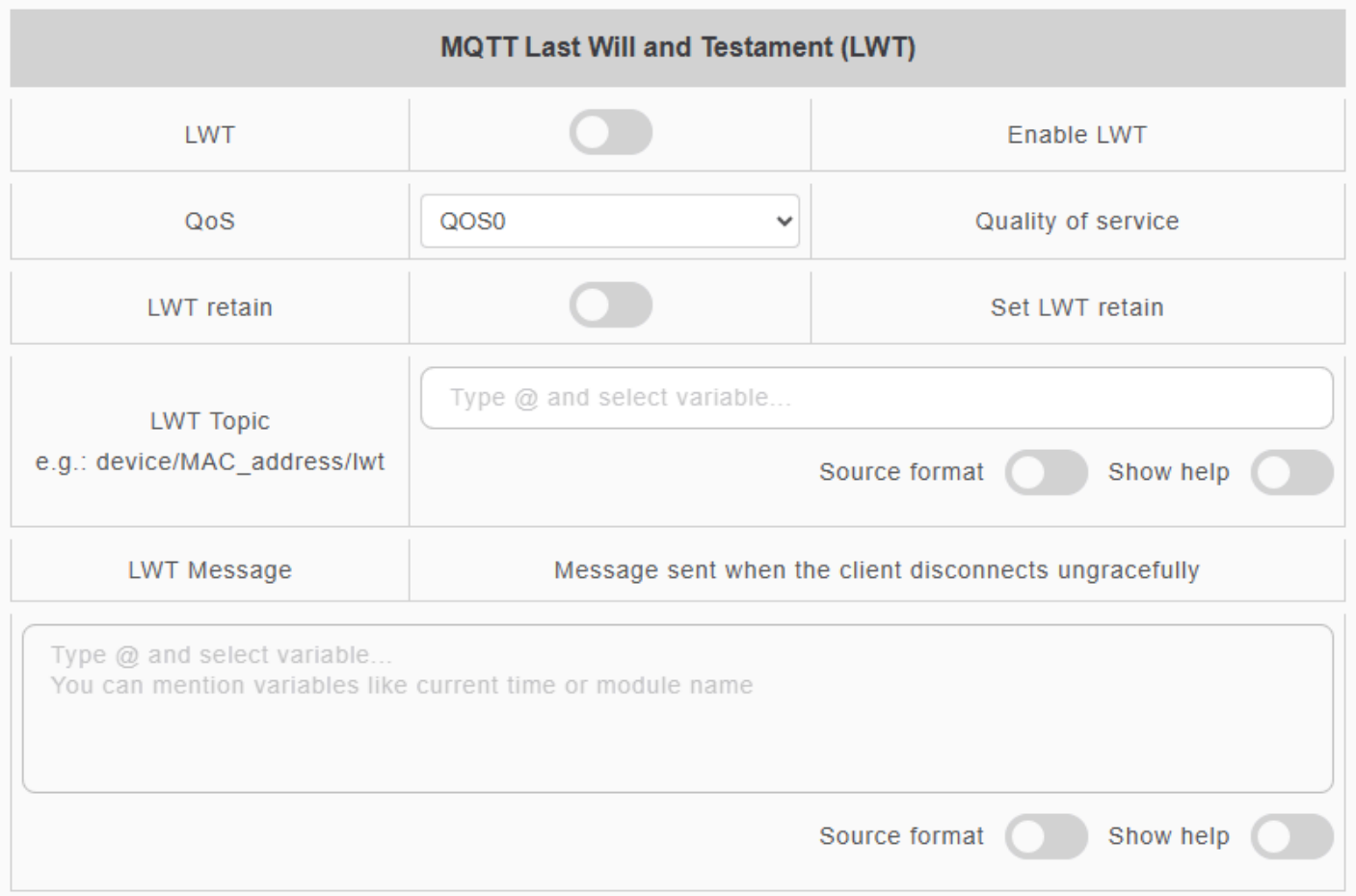
- **MQTT Client** – Enable MQTT service,
- **Broker** – MQTT broker address,
- **Port** – Port, on which the server listens (commonly 1883),
- **QoS** – Quality of Service level. Can assume one of three values: 0 (At most once), 1 (At least once), 2 (Exactly once),
- **Subsribe topic** – The topic must be in the format (eg. /sensor/home – without „/” at the end),
- **Client ID**,
- **User** – (optional) MQTT user name,
- **Password** – (optional),
- **API key** – Enable/disable API key requirement for I/O control,
- **Send test message** - Send a test message to the broker - topic **validation**, payload **1**.

The module can periodically send its data to the Broker. This functionality can be used to verify the integrity of notifications or for periodic data reading.

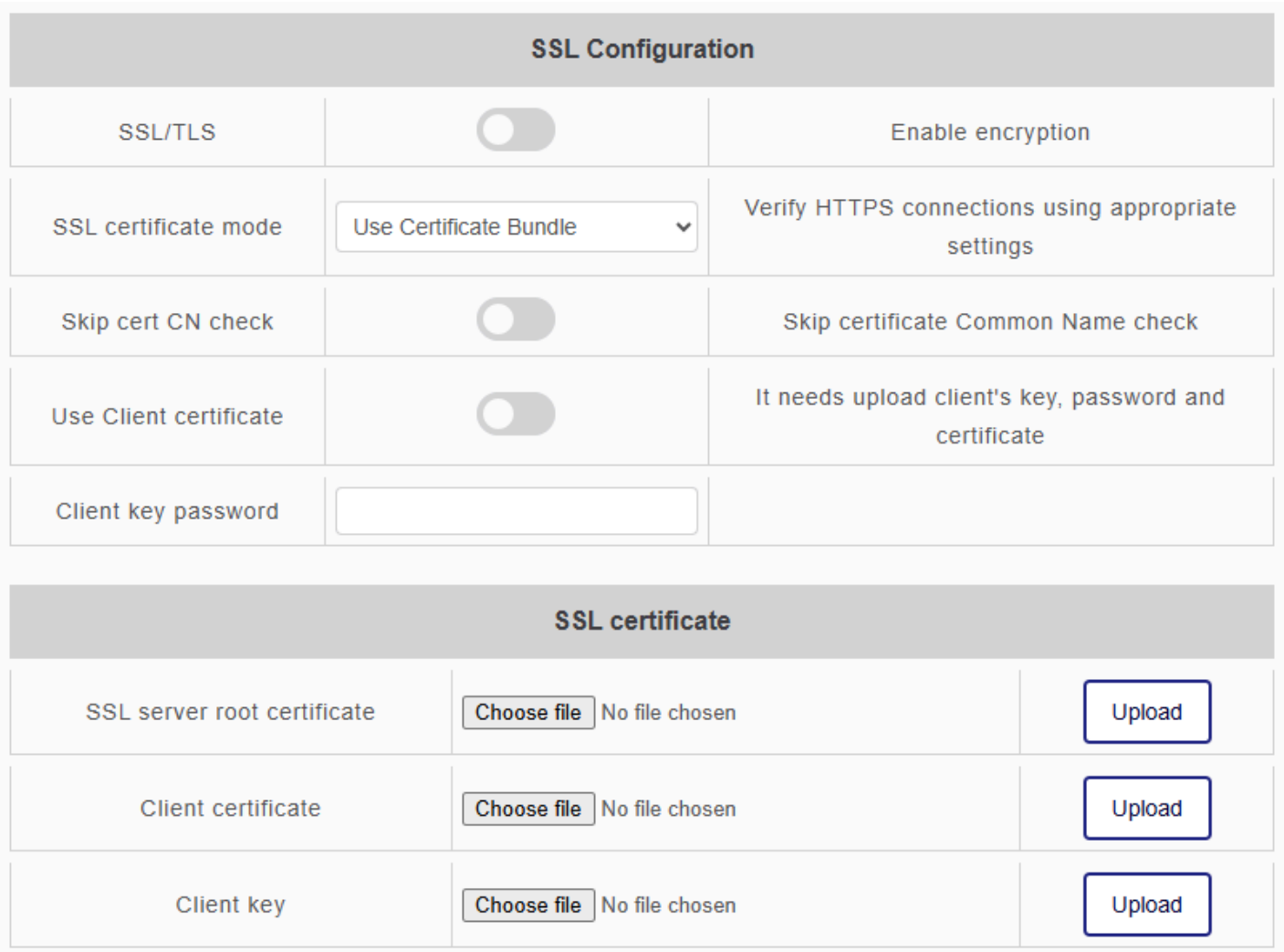


- **Publish topic** – Topic, to which data will be sent periodically,
- **Retain messages** – If enabled, the message will be saved on the broker an will be available for new topic subscribers,
- **MQTT ping payload interval** – Time interval between the data packets. Setting `0` disables the function,
- **MQTT ping paylaod**.

The device is equipped with the LWT mechanism, meaning **Last Will and Testament**. LWT is a mechanism that allows the MQTT client to send a message automatically in the event of a client error or MQTT broker connection loss.
The LWT mechanism allows the user to define a topic and message that will be published if the client looses connection.



- **LWT** – Enable/disable LWT,
- **QoS** – Quality of Service level. Can assume one of three values: 0 (At most once), 1 (At least once), 2 (Exactly once),
- **LWT retain** – If enabled, the LWT message will be saved on the broker an will be available for new topic subscribers,
- **LWT Topic** – Topic that the LWT message will be published at,
- **LWT Message** – LWT message contents.



- **SSL/TLS** – Enable/disable encryption,
- **SSL certificate mode**:
  - **Use Certificate Bundle**,
  - **Use Uploaded Certificate**,
  - **Insecure! Disable SSL verification**,
- **Skip cert CN check** – Skip certificate common name check,
- **Use Client certificate**,
- **Client key password**.
- **SSL server root certificate**,
- **Client certificate**,
- **Client key**.

Tip

After making changes, apply them using the Save button.

Tip

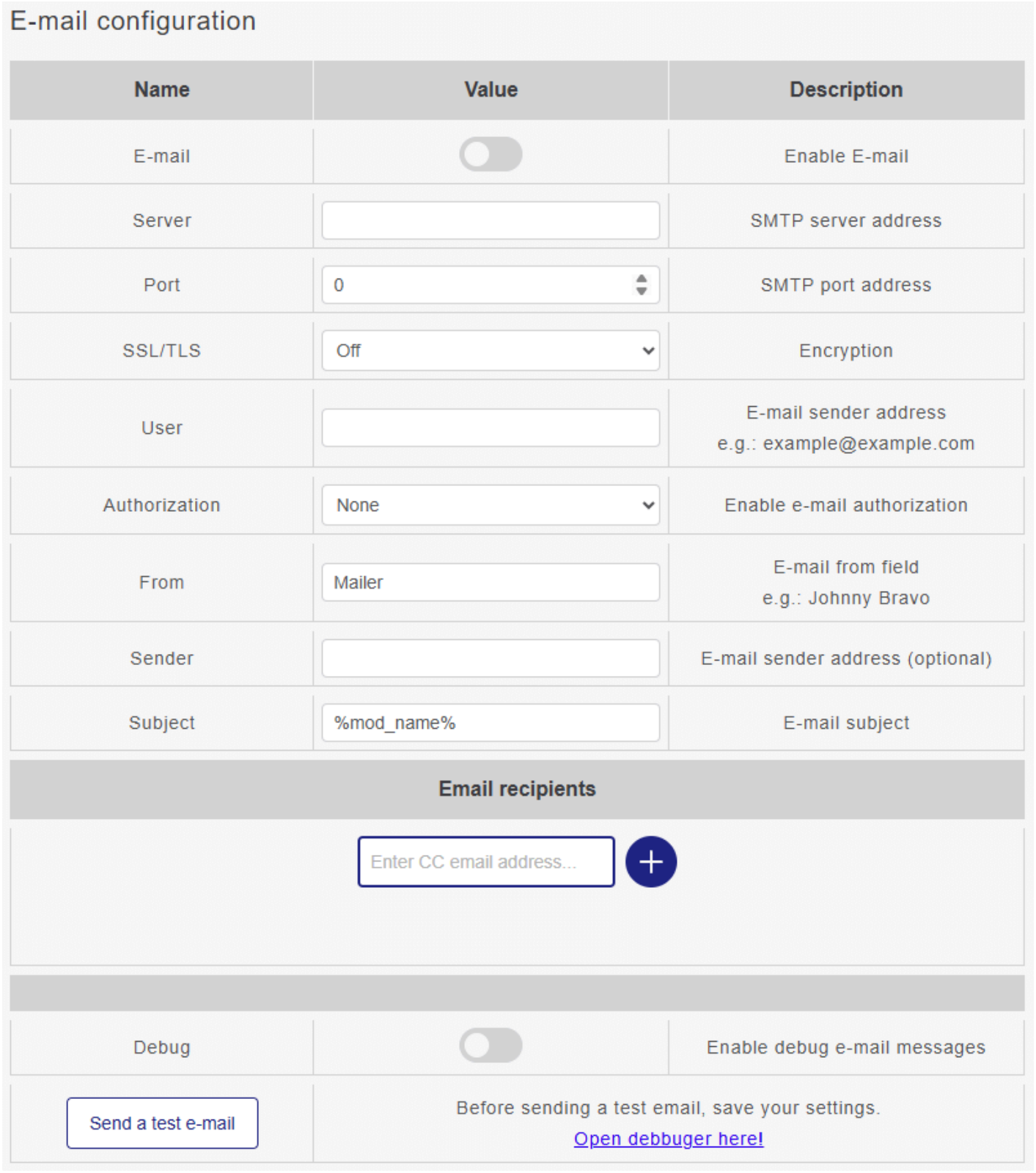
For the Inveo broker, the settings should be as follows:
- MQTT Address**: mqtt.inveo.com.pl**
- MQTT Port**: 1883**
A computer with the Inveo Monitoring application installed can be used as a broker. To do that, type the PC's IP address into the MQTT Address field.

Tip

Make sure that the selected Topic is unique, eg. /daxi/MAC address.

E-mail

In this section, e-mail server connection parameters can be configured. This makes it possible to send e-mails automatically in response to defined actions or alarms.

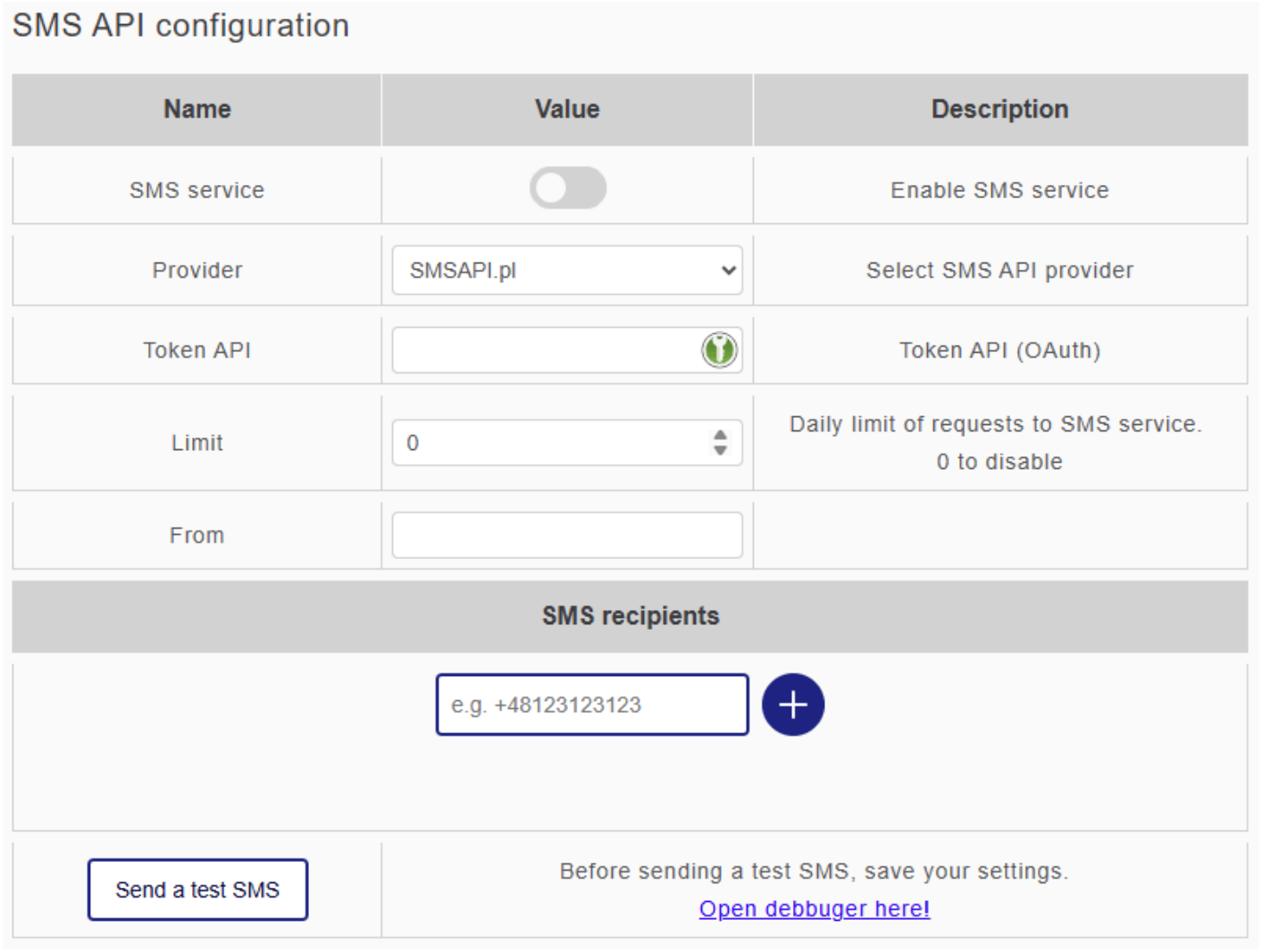


- **E-mail** – Enable/disable e-mail service,
- **Server** – SMTP server address,
- **Port**,
- **SSL/TLS** – Encryption:
  - **Off**,
  - **SSL**,
  - **STARTTLS**.

- **User** – E-mail address,
- **Authorization** – Authorization method:
  - **None**,
  - **Plain** – Password:
    - **Password**.
- **From** – Message sender,
- **Sender** – Sender's e-mail address (if different from the user),
- **Subject**,
- **Recipients**,
- **Debug** – Message debugging function,
- **Send a test e-mail**.

SMS API

The module can be configured to send SMS messages in respondse to defined events or with an information eg. about an alarm.

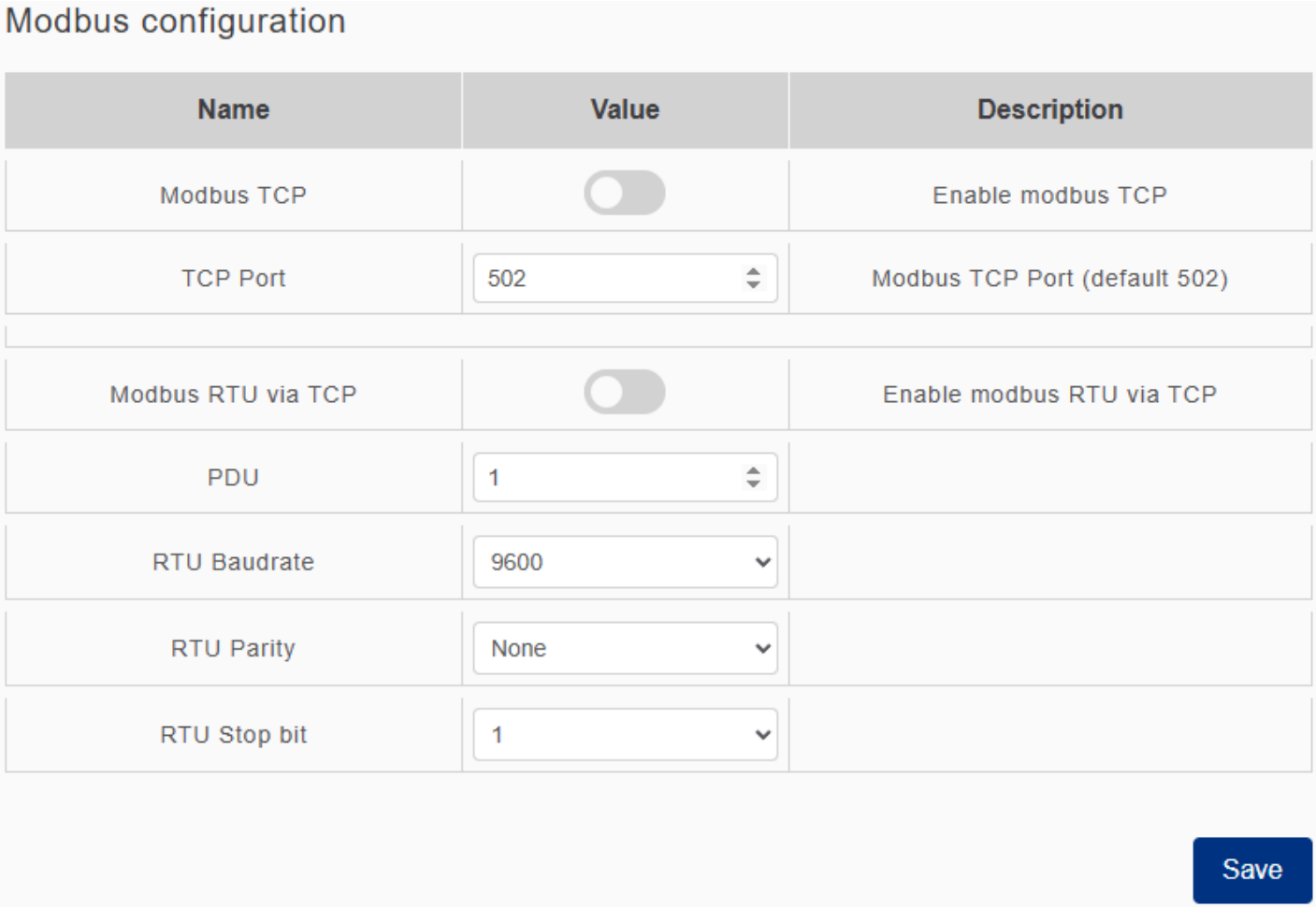


- **SMS service** – Enable/disable SMS service,
- **Provider** – Defines the SMS service provider, with whom the API is integrated:
  - **SMSAPI.pl**
- **Token API** – Unique authorization key used for access authentication and communication with the SMS service provider's API,
- **Limit** – Daily limit of SMS messages, value of 0 will disable the SMS sending service,
- **From** – Defines the message sender: this can be a predefined name, phone number or other identifier that will be visible to the message's recipient,
- **Recipients** – List of recipients phone numbers,
- **Send a test SMS**.

Modbus

Device data can be read and written using Modbus TCP. The device supports the following Modbus functions:

- 0x01 Read Coils
- 0x02 Read Discrete Inputs
- 0x03 Read Holding Register
- 0x04 Read Input Register
- 0x05 Write Single Coil
- 0x06 Write Single Register
- 0x0F Write Multiple Coils
- 0x10 Write Multiple Registers



- **Modbus TCP** – Enable/disable Modbus TCP service,
- **TCP Port** – Modbus TCP port (default 502),
- **Modbus RTU via TCP** – Enable/disable Modbus RTU via TCP service,
- **PDU**,
- **RTU Baudrate**,
- **RTU Parity**,

- **RTU Stop bit**.

Content of the registers is shown in the following tables:

Coils addressing

Address	R/W	Description
1	R/W	Output 0 state, enable/disable output 0: **0** – OFF, **1** – ON
2	R	Output 0 state
3		As shown above – applies to subsequent outputs

Discrete Inputs addressing

Address	R/W	Description
1	R	Input 0 state
2...	R	State of subsequent inputs

Holding Registers addressing

Address	R/W	Description
1	R	Input 0 counter – MSB
2	R	Input 0 counter – LSB
3-200	R	As shown above – applies to subsequent inputs
201	R/W	Output 0 state, enable/disable output 0: **0** – OFF, **1** – ON
202	R/W	Output 0 operating mode: **0** – disable **1** – bistabile **2** – astabile **3** – one pulse
203	R/W	**Time On** value
204	R/W	**Time off** value

205...		As shown above – applies to subsequent outputs
1001		Sensor 0 Low Alarm: **0** – OFF, **1** – ON
1002-1003	R/W	Sensor 0 Low Alarm value (FLOAT32, little endian)
1004	R/W	Sensor 0 Low Warning: **0** – OFF, **1** – ON
1005-1006	R/W	Sensor 0 Low Warning value (FLOAT32, little endian)
1007	R/W	Sensor 0 High Warning: **0** – OFF, **1** – ON
1008-1009	R/W	Sensor 0 High Warning value (FLOAT32, little endian)
1010	R/W	Sensor 0 High Alarm: **0** – OFF, **1** – ON
1011-1012	R/W	Sensor 0 High Alarm value (FLOAT32, little endian)
1013...	R/W	As shown above – applies to subsequent sensors

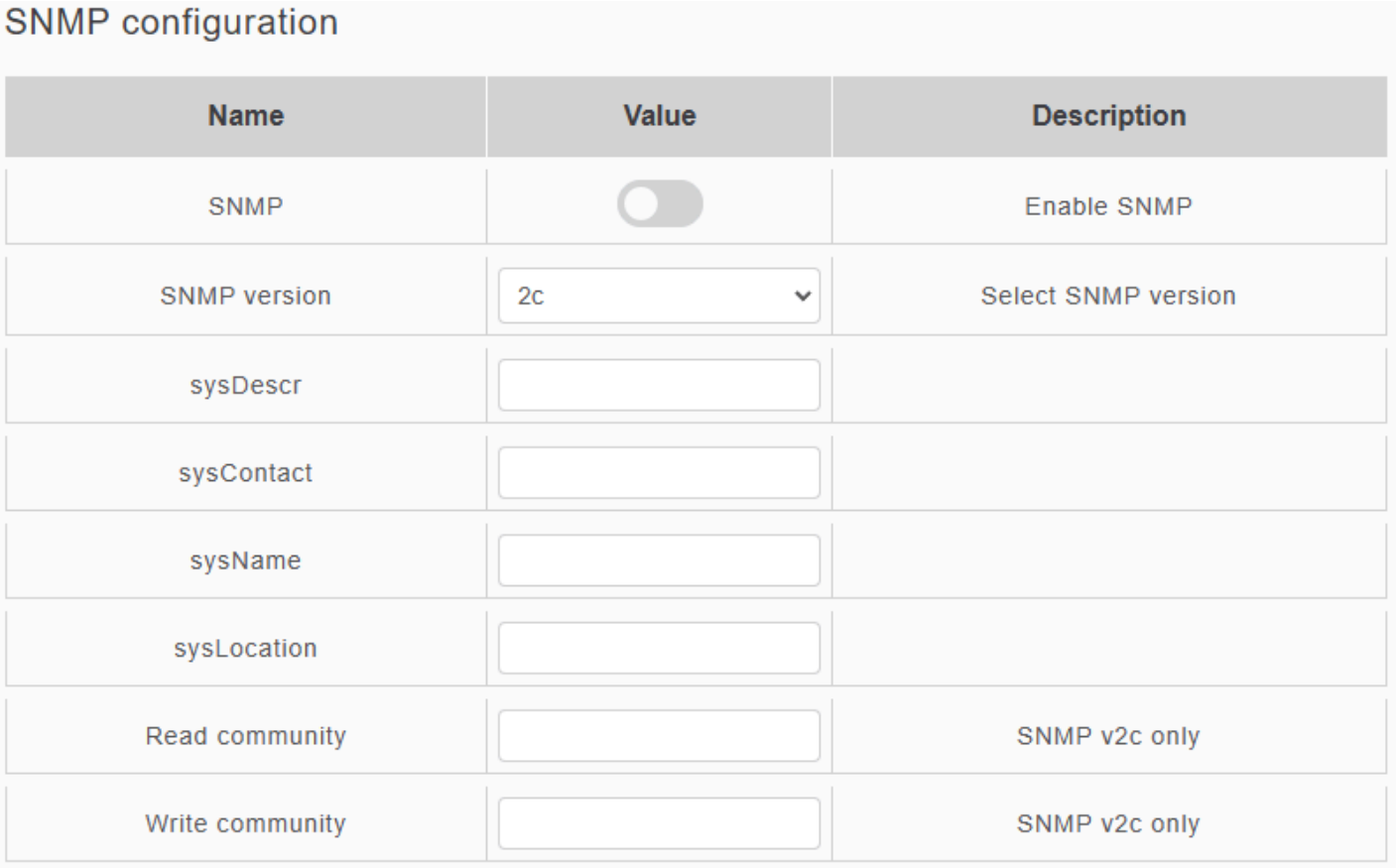
Input Registers addressing

Address	R/W	Description
1	R	**0** – read error (sensor damaged or miswired), **1** – the sensor is giving readings that are within set limits, **2** – low level warning, **3** – high level warning, **4** – low level alarm, **5** – high level alarm, **6** – sensor initializing.
2	R	Sensor value x10 (SINT16)
3-4	R	Sensor value (FLOAT32, little endian)
5	R	Sensor's last read time – MSB
6	R	Sensor's last read time – LSB
7...	R	As shown above – applies to subsequent sensors

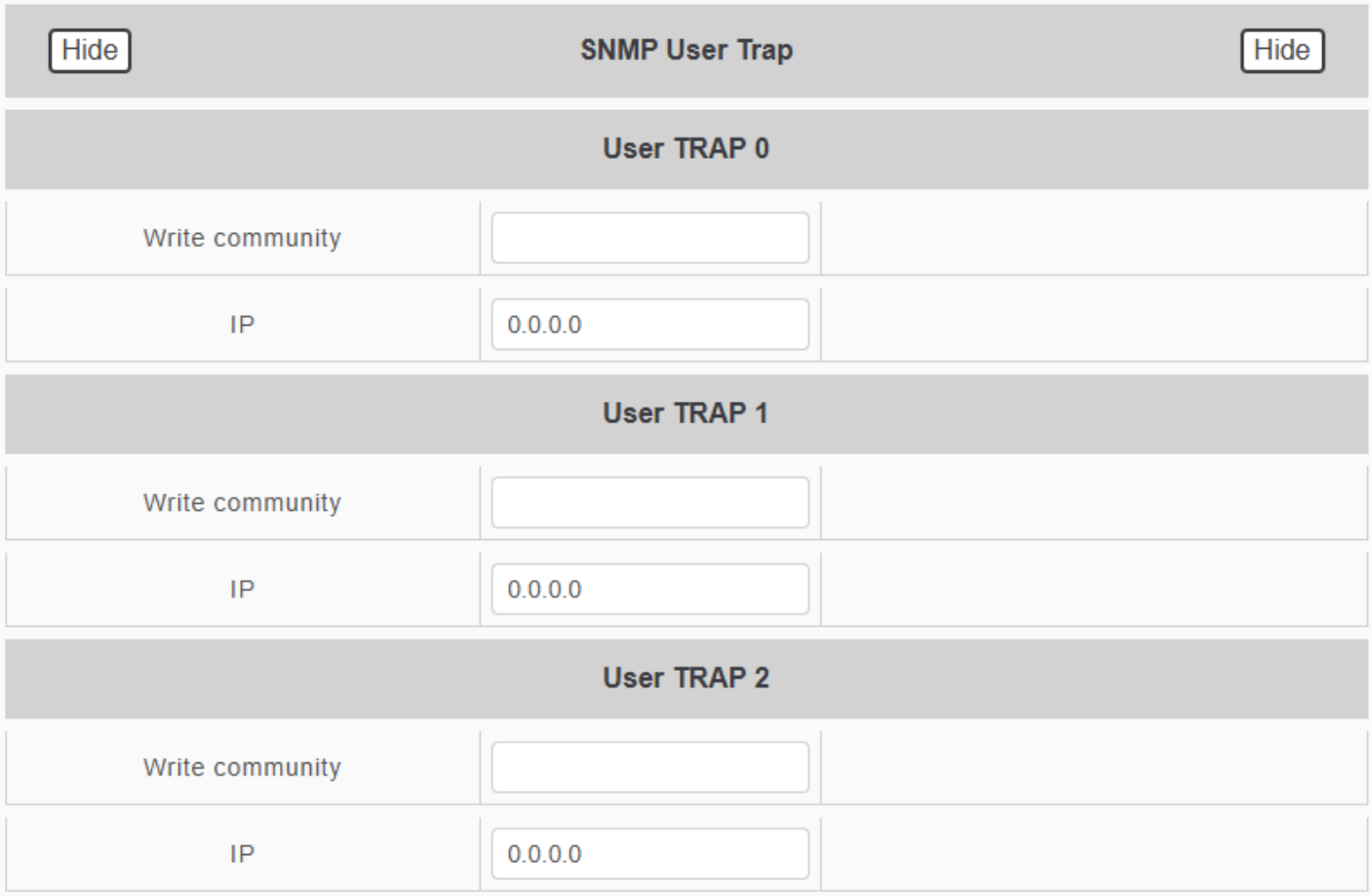
SNMP

This section allows configuration of SNMP protocol settings, used for monitoring and network device management. The module is equipped with the v2c or v2c and v3 *(version dependent)* SNMP servers. Depending on the SNMP version selected, different parameters are available.

SNMP v2c

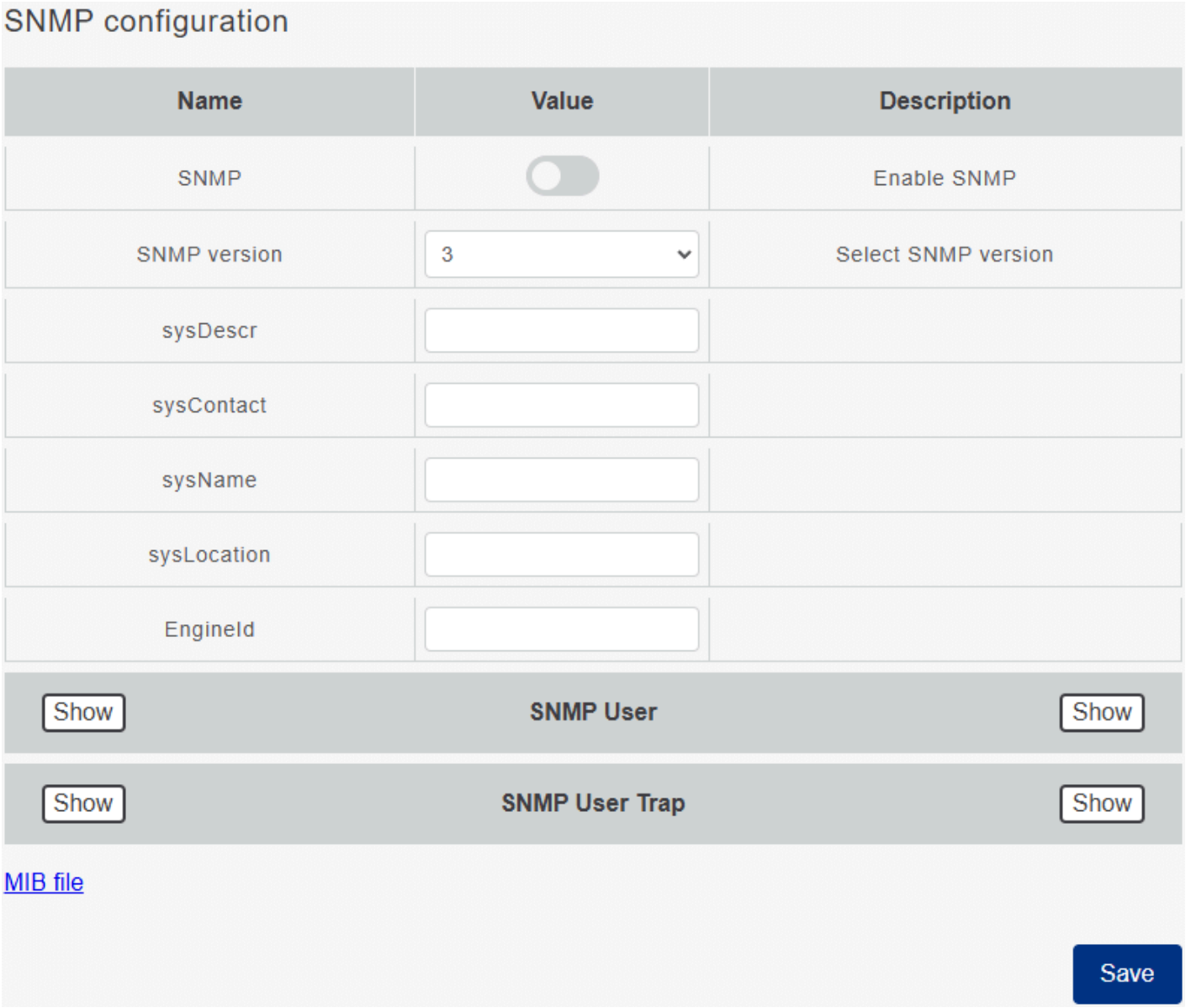


- **SNMP** – Enable/disable SNMP service,
- **SNMP version** – **v2c** or **v3**,
- **sysDescr**, **sysContact**, **sysName**, **sysLocation** – Optional system identification fields,
- **Read community** – Data read password,
- **Write community** – Data write password.

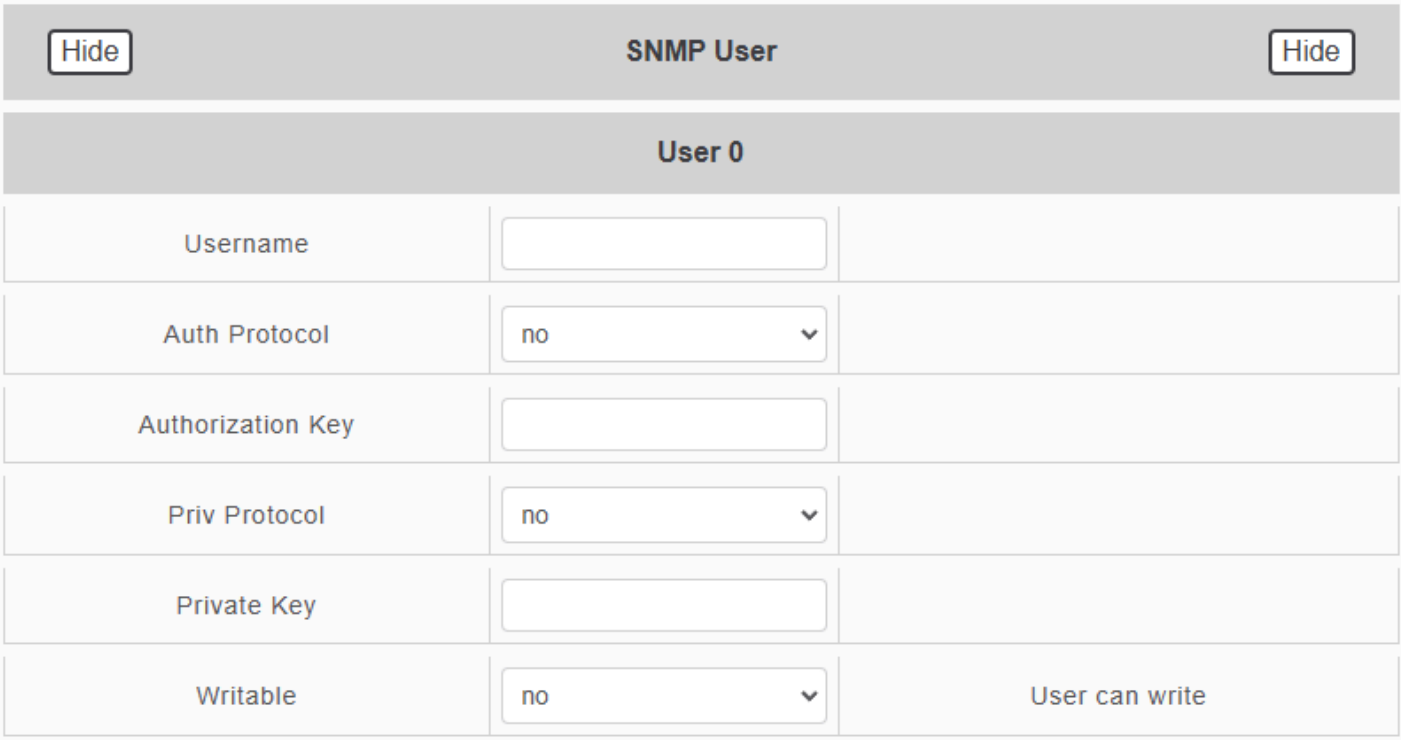


- **Write community** – Data write password,
- **Trap IP** – Address, to which trap messages will be sent.

SNMP v3

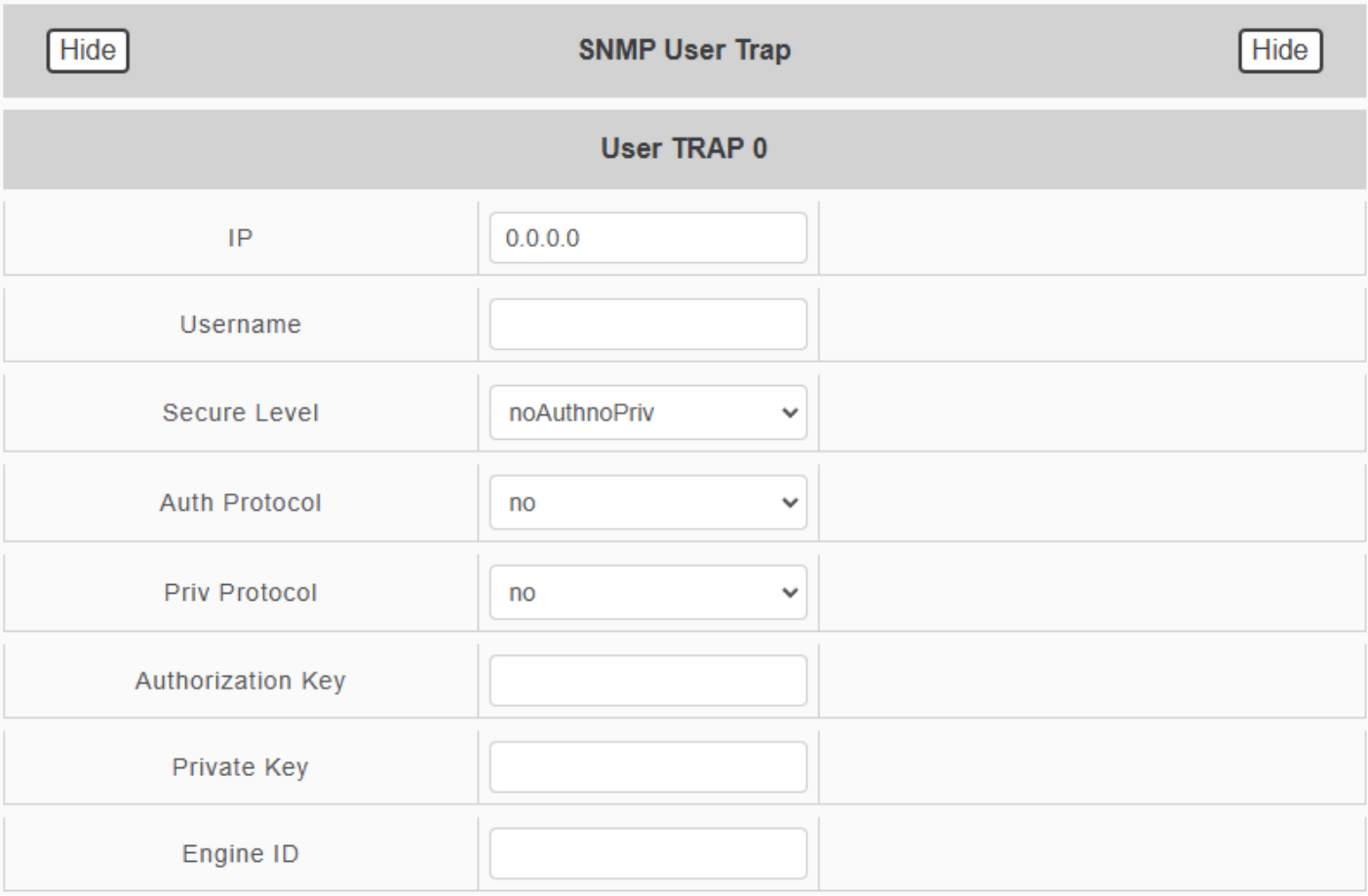


- **Enable** – Enable/disable SNMP service,
- **SNMP version** – **v2c** or **v3**,
- **sysDescr**, **sysContact**, **sysName**, **sysLocation** – Optional system identification fields,
- **EngineId** – Unique device identifier.



Parameters that can be set in this window allow defining of authentication and privacy mechanism for different users.

- **Username**,
- **Secure level** – SNMP v3 security level:
  - **noAuthnoPriv** – SNMPv3 communication is carried out with no security,
  - **authNoPriv** – SNMPv3 communication is authenticated but not encrypted,
  - **authPriv** – SNMPv3 communication is authenticated and encrypted,
- **Auth Protocol**:
  - MD5,
  - SHA.
- **Authorization Key**,
- **Priv Protocol**:
  - DES,
  - AES.
- **Private Key**,
- **Writable** – Assign the user with privilege to send messages to the device.



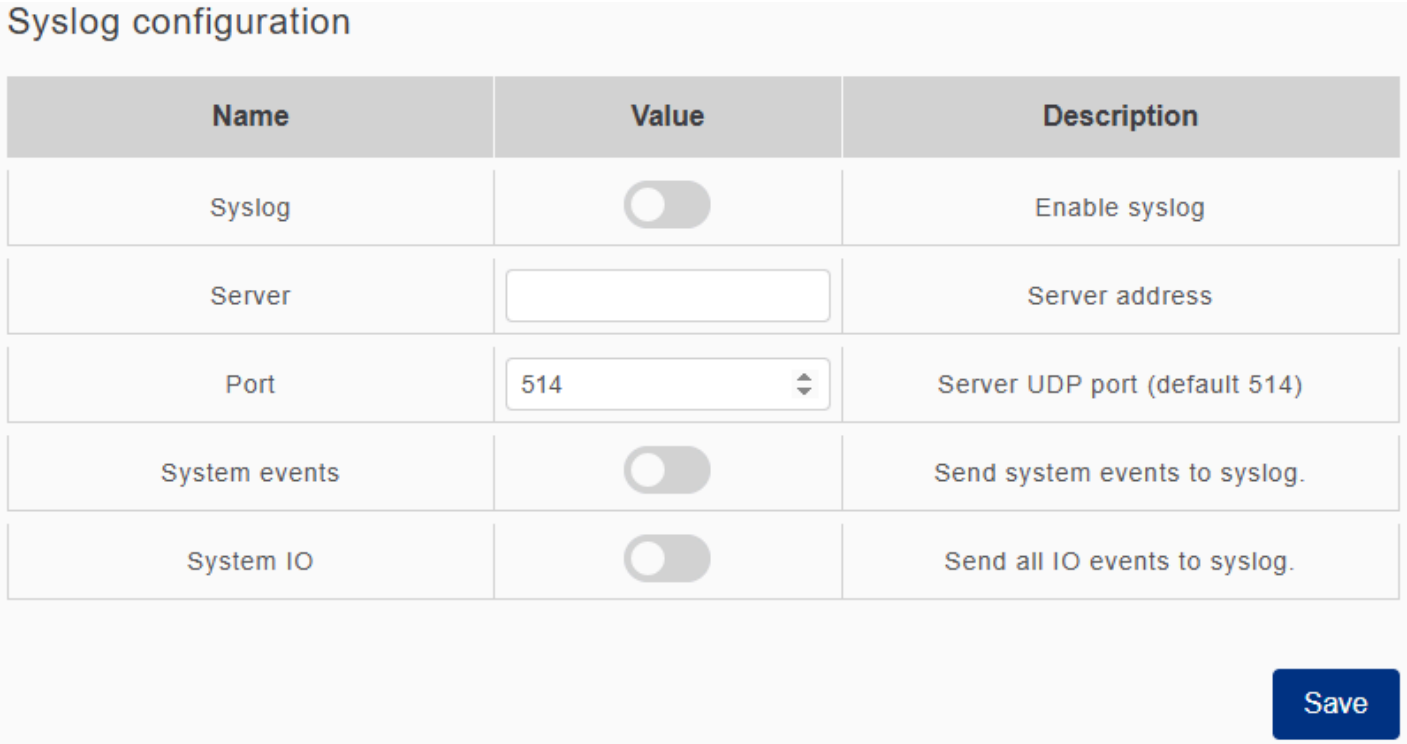
Parameters that can be set in this window apply to specific trap notifications generated for or by the user concerned.

- **IP** – IP address of the device or system generating the trap notification,
- **Username** – SNMPv3 user name,
- **Secure Level** used with SNMPv3:
  - **noAuthnoPriv** – SNMPv3 communication is carried out with no security,
  - **authNoPriv** – SNMPv3 communication is authenticated but not encrypted,
  - **authPriv** – SNMPv3 communication is authenticated and encrypted,
- **Auth Protocol**:
  - MD5,
  - SHA.
- **Authorization Key**,
- **Priv Protocol**:
  - MD5,
  - SHA.
- **Private Key**,
- **Engine ID** – Unique identifier used to represent the SNMPw engine in the device.

Download MIB file – Link to download the MIB file.

Syslog

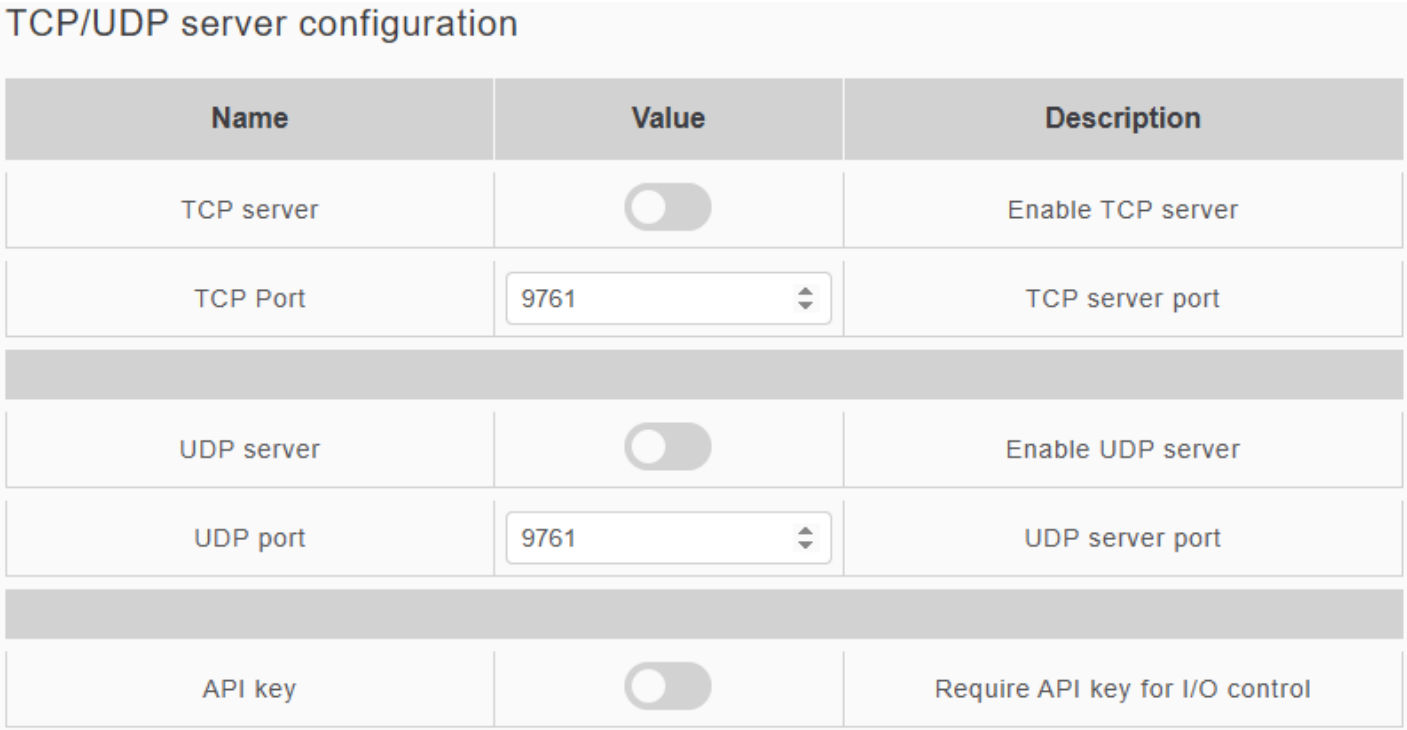
In this section, sending of system logs to a remote syslog server can be configured, making remote diagnostics and device monitoring easier.



- **Enable** – Enable syslog client service,
- **Server** – Syslog server address in the form of IP address or domain name (maximum of 24 characters),
- **Port** – Port number, the syslog server is listening to,
- **System events** – Events related to system operation,
- **Syslog IO**.

TCP/UDP

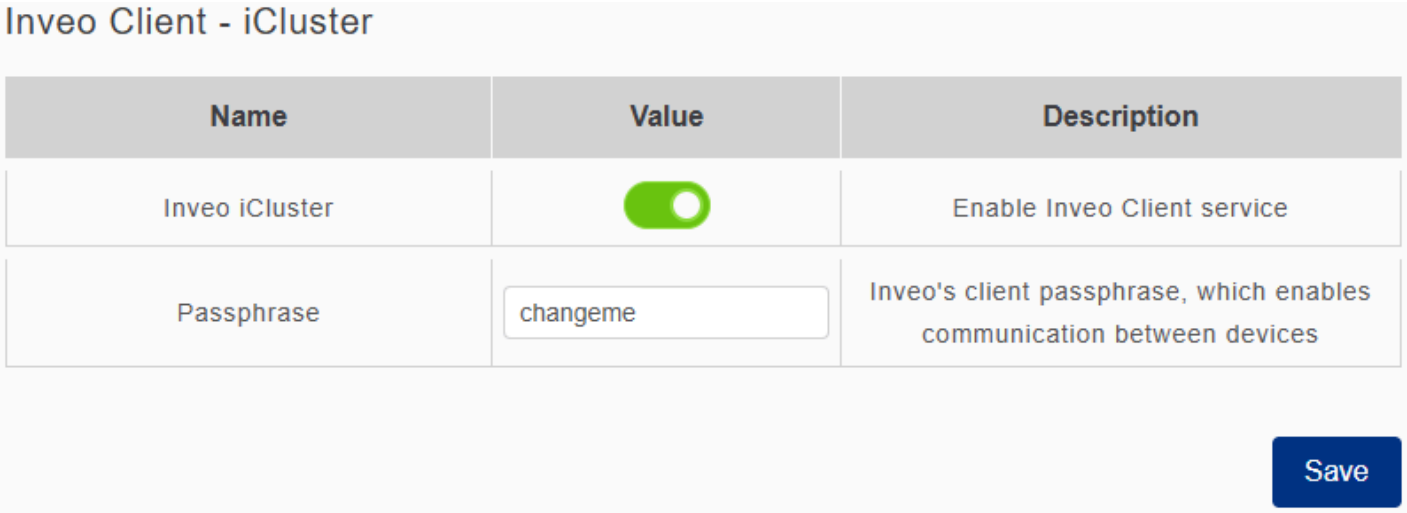
This tab allows the user to enable and configure TCP and UDP protocols support. The user can customize settings such as ports and communication parameters, providing flexibility in device configuration in compliance with network requirements. By sending commands to the device's IP address and port using an appropriate protocol, the user can control the outputs state. Available commands – see **IO commands**.



- **TCP server**,
- **TCP port**,
- **UDP server**,
- **UDP port**,
- **API key** – Enable/disable API key requirement for I/O control.

iCluster

iCluster allows multiple Inveo devices to interconnect within a single LAN. IQIO can operate as an iCluster client, allowing the device to oversee its parameters such as: sensor readings, input/output state, output control etc.



- **Inveo iCluster** – Enable/disable **iCluster** service,
- **Passphrase** – Password used by the Inveo client that allows communication between devices. The password set here will be required in the device that would like to establish a connection with a Daxi device.

IO commands

Shown below is the summary of commands used to create actions based upon IO commands. It is worth noting that those commands are useful when using different protocols such as HTTP, MQTT, UDP and TCP too.

- `in_cnt=ch,val` – set the `ch` channel counter value to `val`,
- `out_on=ch` – activate output channel `ch`,
- `out_off=ch` – deactivate output `ch`,
- `out_inv=ch` – output `ch` state invert,
- `out_blink=ch,ton,toff,cnt` – periodic output `ch` control. Parameters:
  - `ton` – activation time (in seconds),
  - `toff` – deactivation time (in seconds),
  - `cnt` – number of activation cycles (not required).
- `out_time=ch,ton,toff` – activate output `ch` for the time defined in `ton`, after `toff` time passes. The `toff` parameter is not required – omitting it will activate the output without a delay,
- `out_all=10n-11100` – command defining the state of all available outputs. Each digit represents a subsequent output:
  - `1` – activated,
  - `0` – deactivated,
  - `n` – state invert,
  - `-` – no state change.

Example

`out_all=10n-1110` will activate outputs No. 0, 4, 5, 6; deactivate outputs 1 and 7; invert the state of 2; leave output No. 3 with no changes.

The commands can be joined together using **&**.

Example

```
out_on=2&out_inv=3&out_time=1,20,20
```

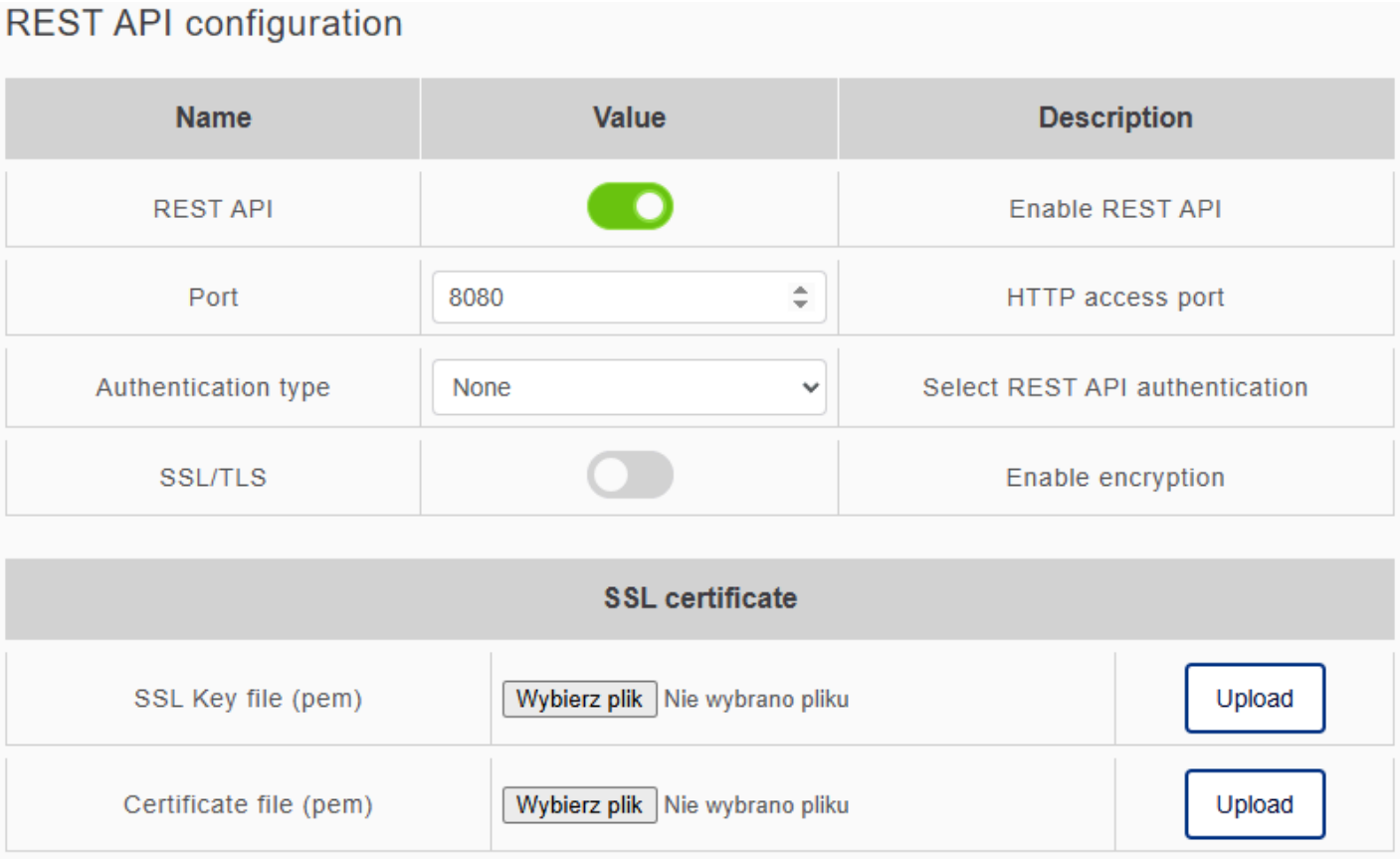
Built-in variables

The following table contains example internal variables that allow for precise transfer of module operation data. Those variables are key elements in configuration, e-mail and SMS notifications, HTTP Client etc.

Syntax	Example	Description
`%out[range],[off],[on]%`	`%out[0-5],0,1%`	Outputs state [range] is the range of outputs to be displayed [off] is the value for inactive state [on] is the value for active state Example: state of OUT 0-5 will be displayed. 0 - inactive 1 - active
`%in[range],[off],[on]%`	`%in[0-7],i,I%`	Inputs state [range] is the range of inputs to be displayed [off] is the value for inactive state [on] is the value for active state Example: state of IN 0-7 will be displayed. 0 - inactive 1 - active
`%cnt[number]%`	`%cnt5%`	Input counter value [number] is the input number Example: counter value of input No. 5 will be displayed
`%sens[number]%`	`%sens10%`	Sensor value [number] is the sensor number Example: sensor No. 10 value will be displayed
`%sunrise%`	`%sunrise%`	Sunrise hour
`%sunset%`	`%sunset%`	Sunset hour
`%time%`	`%time%`	Current time
`%date%`	`%date%`	Current date
`%timedate%`	`%timedate%`	Current time and date
`%ts%`		Current timestamp – number of seconds since the specified date: 1st of January 1970
`%mod_name%`		Module name defined by the user
`%mod_model%`		Device model
`%eip%`		Device's IP address
`%emac%`		Device's MAC address

`%s[x]%`	`%s[3]%`	Sensor value Example: sensor No. 3 value will be displayed
`%s[x].statTxt`	`%s[2].statTxt`	Sensor status Example: sensor No. 2 status will be displayed
`%o[x]%`	`%o[4]%`	Output state Example: state of output No. 4 will be displayed
`%i[x]%`	`%i[1]%`	Input state Example: state of input No. 1 will be displayed
`%v[x]%`	`%v1%`	Virtual variable value
`%cnt[x]%`	`%cnt1%`	Input counter value

REST API



- **REST API** – Enable REST API,
- **Port**,
- **Authentication type**:
  - **None**,
  - **Basic authentication** – User name and password authentication,

    - **API KEY** – API key authentication, utilizing a key generated in **Administration / Access**. The key should be entered as a parameter, preceded with `apikey=`.
- **User** – User name used with Basic authentication,
- **Password** – Password used with Basic authentication,
- **SSL/TLS** – Enable/disable encryption.
- **SSL Key file (pem)** – Allows upload of SSL key (pem format),
- **Certificate file (pem)** – Allows upload of certificate file (pem format).

Example

API key value: `12345678901234567890`
REST API request syntax: `http://<IP_address>:<port>/io/outputs/1?apikey=12345678901234567890`

Factory settings, backup

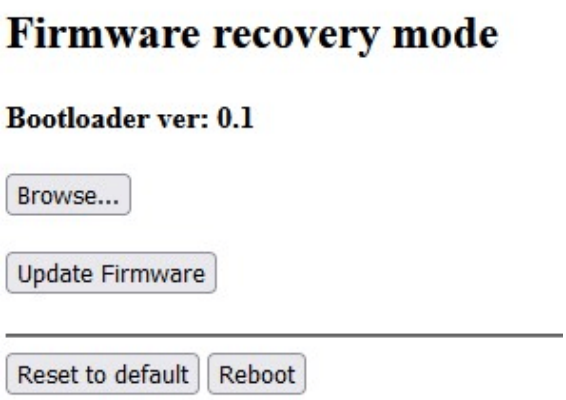
Emergency firmware restore / restoring factory settings

If device failure occurs, preventing normal access to the webpage, the emergency procedure should be put into effect:

1. Disconnect the power supply,
2. Depress the **RESET** button,
3. Power up the device and connect it to LAN,
4. Without releasing the **RESET** button, open the device's webpage:
    - **IP address:** 192.168.111.15
    - **IP mask:** 255.255.255.0

> **Tip**
>
> **In order to access the 192.168.111.15 address, computer's IP address must be in the same subnet (example computer address: 192.168.111.1).**

Referring to the specified IP address will give access to the device's bootloader. The **RESET** button can be released after the following webpage opens:



In this window, the following options are available:

- Firmware upload,
- Default settings reset,
- Device restart.

Warranty and manufacturers liability

> **Warning**
>
> **The manufacturer provides a two-year warranty for the device and post-warranty service for a period of 10 years from the date of introduction of the device to the market. The warranty covers all material and production defects.**

The manufacturer undertakes to respect the warranty agreement, if the following conditions are met:

- All repairs, changes, expansions and device calibrations are carried out by the manufacturer or an authorized service center,
- The power supply system meets the applicable standards,
- The device is operated in accordance with the suggestions presented in this manual,
- The device is operated in accordance with its intended purpose.

The manufacturer assumes no responsibility for consequences resulting from improper installation, improper use of the device, failure to comply with the instruction manual, and repairs made by unauthorized personnel.

> **Warning**
>
> **The device contains no user serviceable parts inside.**

Storage, operation and transport conditions

The device should be stored in enclosed rooms, where the atmosphere is free from vapours and corrosive substances:

- Environment temperature from -30°C to +60°C (-22°F - 140°F),
- Humidity from 25% to 90% (condensation unacceptable),
- Atmospheric pressure from 700 to 1060 hPa.

The device is intended to operate in the following conditions:

- Environment temperature from -10°C do +55°C (14°F - 131°F),
- Humidity from 30% to 75%,
- Atmospheric pressure from 700 to 1060 hPa.

Recommended transport conditions:

- Environment temperature from -40°C do +85°C (-40°F - 185°F),
- Humidity from 5% to 95%,
- Atmospheric pressure from 700 to 1060 hPa.

Installation and device operation:

- The module should be operated in accordance with recommendations provided later in this manual.

Disposal and decommissioning

In an event the device needs to be decommissioned (eg. after its intended life period is surpassed), it is recommended to contact the manufacturer or his representative, who are responsible to respond appropriately, i.e., to collect the device from the user. The user can alternatively contact companies specializing in electronic device or computer equipment disposal and/or decommissioning. Under no condition should the device be placed with other waste.