smartDEN IP-16R-XX

Web-enabled Ethernet-based 16 Relay Modules

User Manual

Date: 13 Apr 2021

Device	Short Name	Integration Protocol	Firmware version
	*smartDEN IP-16R*	*SNMPv2*	*v1.20 / May 2017*
	*smartDEN IP-16R-MT*	*Modbus TCP*	*v1.21 / Jul 2020*
	*smartDEN IP-16R-MQ*	*MQTT V3.1.1*	*v1.21 / Sep 2020*

Content

1. Features

smartDEN IP-16R-XX is a LAN relay module with 16 SPDT relays for remote control with integrated web server for set-up, API integration protocol (depending on the model) and http/xml/json support for embedding in other systems. The built-in real time clock allows organizing schedule stand-alone work without connection to computer. The module is able to act like network watch-dog in order to monitor and reboot network equipment automatically.

- 10 Mb Ethernet interface with Link/Activity Led;
- Auto-MDIX;
- 16 SPDT relays (with NO and NC contacts);
- Led for each relay;
- Pulse function (timer) for every relay;
- Real Time Clock (RTC) for schedule (calendar) stand-alone work;
- Web server with secure login authorization;
- Secure HTTP/XML/JSON API protocol support for read/write relays status;
- Integration protocol:
  - SNMPv2 - for smartDEN IP-16R;
  - ModBUS TCP - for smartDEN IP-16R-MT;
  - MQTT V3.1.1 - for smartDEN IP-16R-MQ;
- Supported by various Home Automation systems like Domoticz, OpenHab, Home Assistant, Node-Red;
- Supported protocols: ARP, IP, ICMP (ping), DHCP, DNS;
- Watch-Dog Auto-Reboot ICMP (outgoing) function;
- Access protection (by IP and MAC address);
- Option for relays states saving and loading on reset;

2. Ordering codes

Table 2.1. Ordering codes

Short Name	Ordering Codes	Description
smartDEN IP-16R	smartDEN IP-16R-12V-PCB	SNMP model, PCB version, 12VDC supply voltage
	smartDEN IP-16R-24V-PCB	SNMP model, PCB version, 24VDC supply voltage
	smartDEN IP-16R-12V-BOX	SNMP model, BOX version, 12VDC supply voltage
	smartDEN IP-16R-24V-BOX	SNMP model, BOX version, 24VDC supply voltage
smartDEN IP-16R-MT	smartDEN IP-16R-MT-12V-PCB	Modbus TCP model, PCB version, 12VDC supply voltage
	smartDEN IP-16R-MT-24V-PCB	Modbus TCP model, PCB version, 24VDC supply voltage
	smartDEN IP-16R-MT-12V-BOX	Modbus TCP model, BOX version, 12VDC supply voltage
	smartDEN IP-16R-MT-24V-BOX	Modbus TCP model, BOX version, 24VDC supply voltage
smartDEN IP-16R-MQ	smartDEN IP-16R-MQ-12V-PCB	MQTT model, PCB version, 12VDC supply voltage
	smartDEN IP-16R-MQ-24V-PCB	MQTT model, PCB version, 24VDC supply voltage
	smartDEN IP-16R-MQ-12V-BOX	MQTT model, BOX version, 12VDC supply voltage
	smartDEN IP-16R-MQ-24V-BOX	MQTT model, BOX version, 24VDC supply voltage

3. Application examples

- Remote control of electrical appliances
- Industrial automation
- Home automation
- Watchdog monitoring of network equipment and auto-reboot
- Internet of Things (IoT)

Figure 3.1. smartDEN IP-16R-XX supports up to 30 schedule events for controlling appliances without connection with computer.

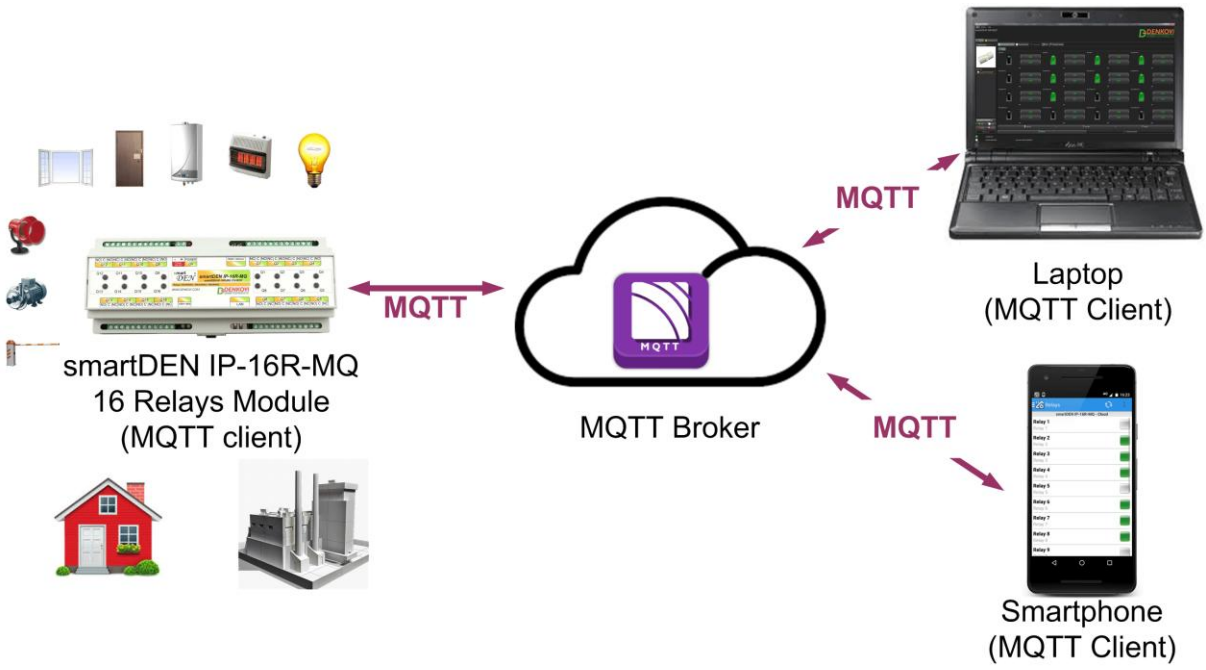

Figure 3.2. Cloud access **smartDEN IP-16R-MQ** via MQTT Broker for IoT, Home Automation and Industrial Automation applications.

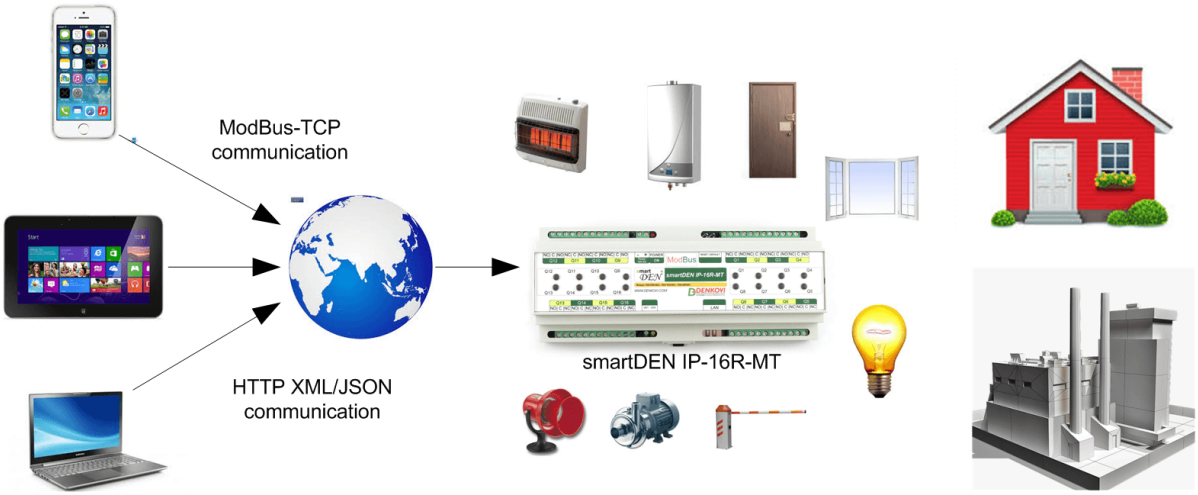

Figure 3.3. Easily access **smartDEN IP-16R-XX** via internet or LAN and control appliances remotely for Home Automation and Industrial Automation applications using some of the supported integration protocols.

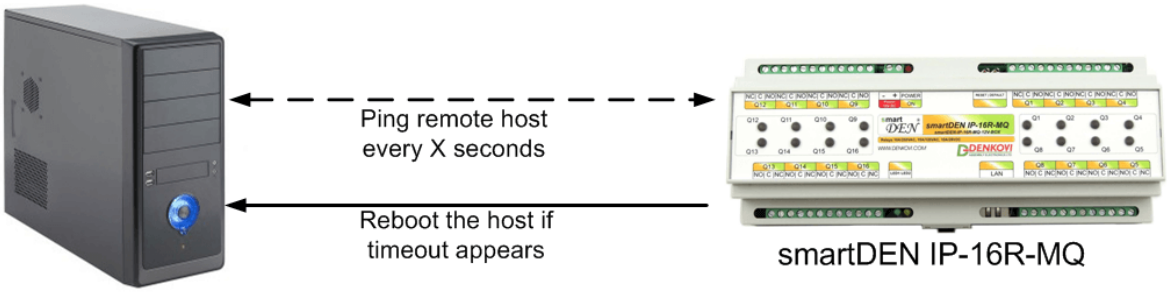

Figure 3.4. smartDEN IP-16R-XX supports periodically sending outgoing ping request to remote host (router, switch, PC, IP camera...) and upon timeout it will perform reboot of the device in order "to keep it alive".

4. Technical parameters

Table 4.1. Technical parameters

Parameter	Value
Box size, mm	210 x 85 x 58
PCB size, mm	203 x 82
Box weight, g	420
PCB weight, g	285
Power supply voltage,VDC	12 or 24 (depends on the model) ±2
Maximum current consumption at 12VDC (when all relays are ON), mA	600
Maximum current consumption at 24VDC (when all relays are ON), mA	400
Operating temperature, °C	0 to 70
Relays maximum switchable current / voltage	10A / 250VAC, 15A / 120VAC, 10A / 28VDC

5. Connectors, ports and led indicators

Bellow is shown a picture with the device connectors, ports and led indicators.

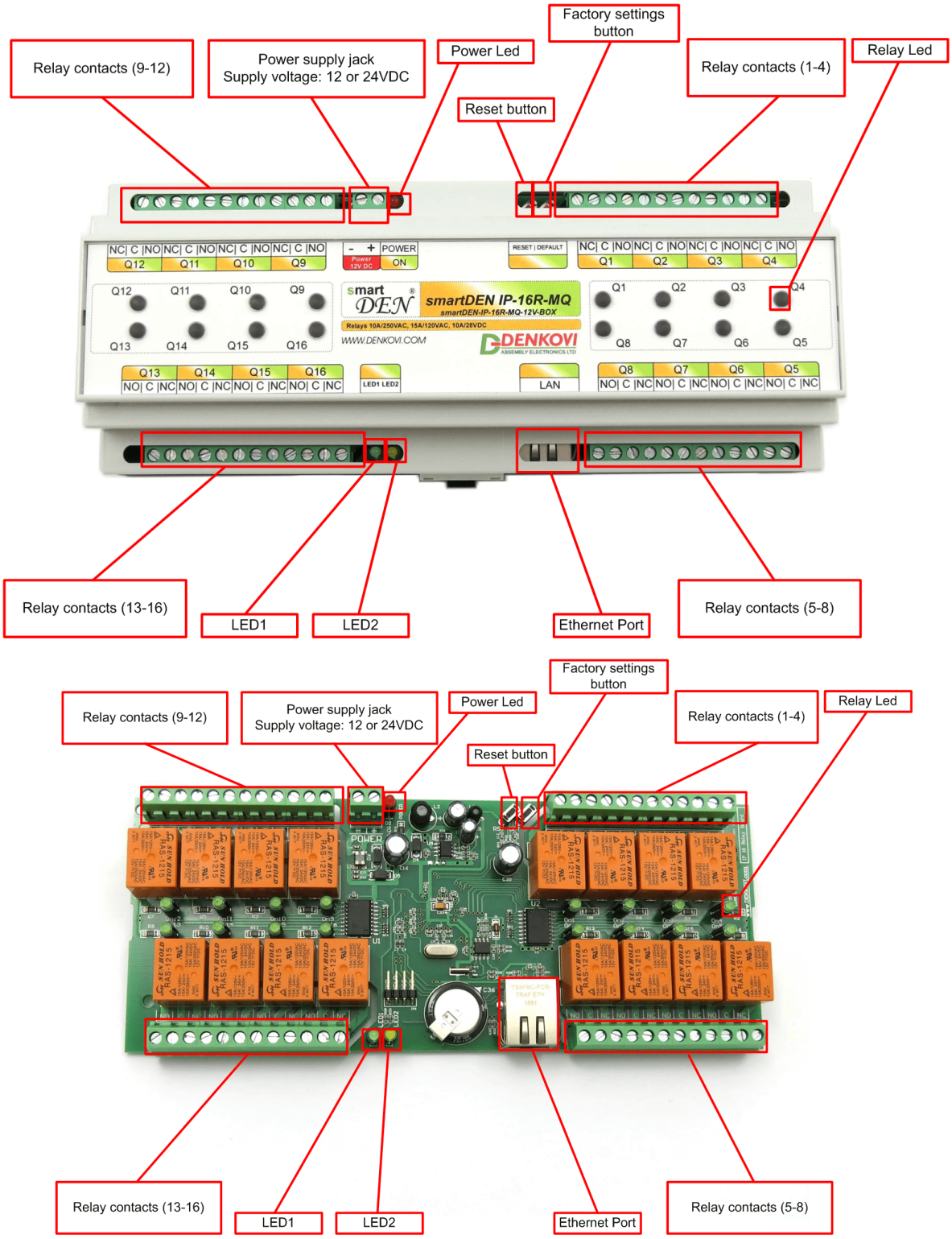

Figure 5.1. Device overview

6. Installation

- This device must be installed by qualified personnel;
- This device must not be installed directly outdoors;
- Installation consists of mounting the device, connecting to an IP network, connecting the relays, providing power and configuring via a web browser.

6.1. Box mounting

Figure 6.1. Mounting the device to DIN rail

smartDEN IP-16R-XX can be mounted to a standard (35mm by 7.55mm) DIN rail. Attach the module to the DIN rail by hooking the hook on the back of the enclosure to the DIN rail and then snap the bottom hook into place.

6.2. Power supply

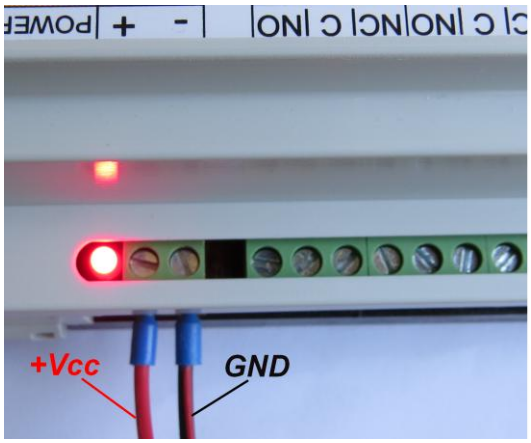

Figure 6.2. smartDEN IP-16R-XX power supply

Depending on the selected model during purchase the power supply source for **smartDEN IP-16R-XX** must be with voltage either **12VDC** or **24VDC** stabilized and filtered. After power on, the power led must be on and **Led1 indicator** must start blinking in 5 seconds which means the controller is running normally.

Figure 6.3. Connecting a LAN cable

- Please keep the polarity and supply voltage range!
- **smartDEN IP-16R-XX** does not accept AC supply voltage. It is highly recommended to check the power supply source parameters before supply the module.
- The power supply equipment shall be resistant to short circuit and overload in secondary circuit.
- When in use, do not place the equipment so that it is difficult to disconnect the device from the power supply.

6.3. Relay connection

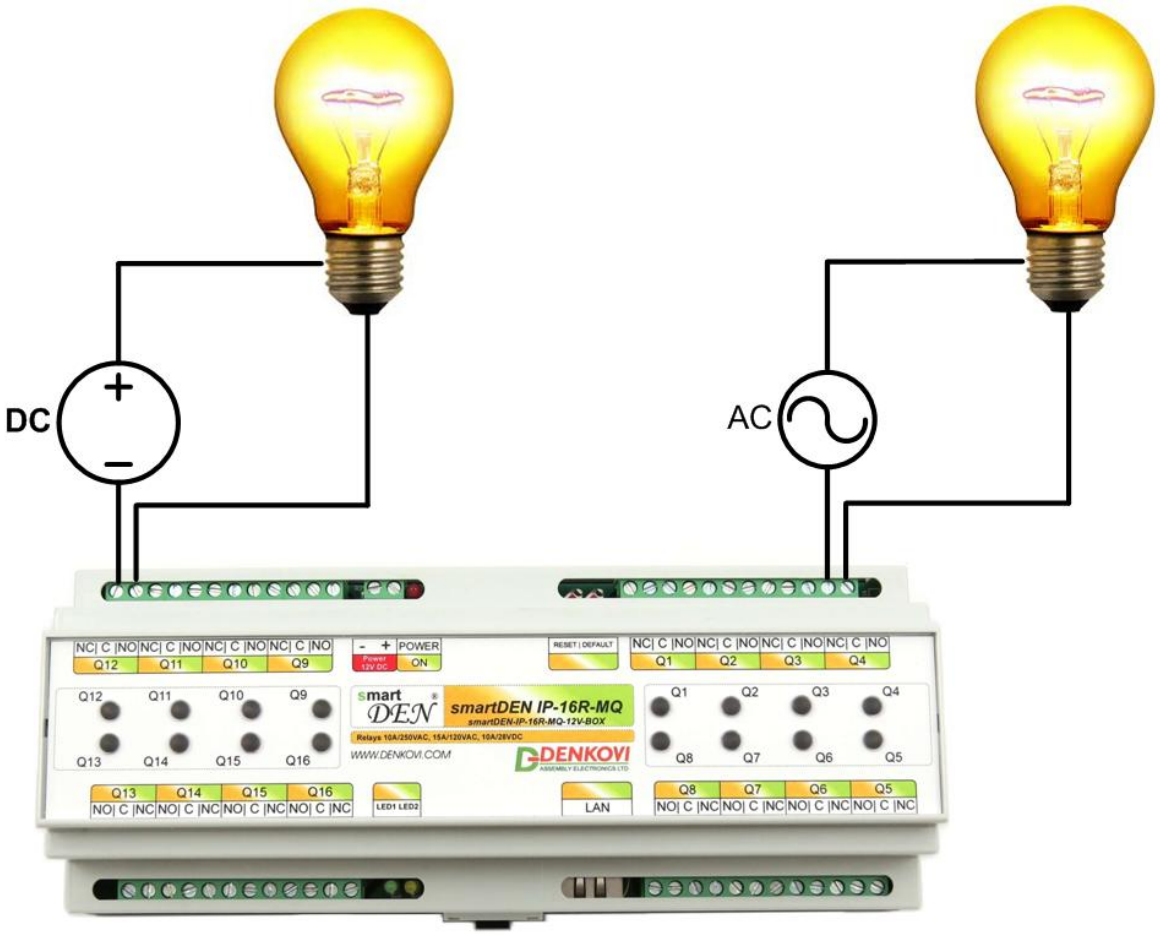

Figure 6.4. Connecting a lamp to relay

smartDEN IP-16R-XX has 16 SPDT relays with parameters specified in the technical parameters section. Every relay channel has normally open (NO) and normally closed (NC) contacts connected directly to the terminals.

If you are connecting inductive loads to the relays an extra measures must be taken in order to ensure the proper work of the device. For more information please refer to this link:
http://denkovi.com/controlling-inductive-devices

6.4. Network connection

smartDEN IP-16R-XX supports AUTO-MDIX so either "crossover" or "straight-through" network cable can be used.

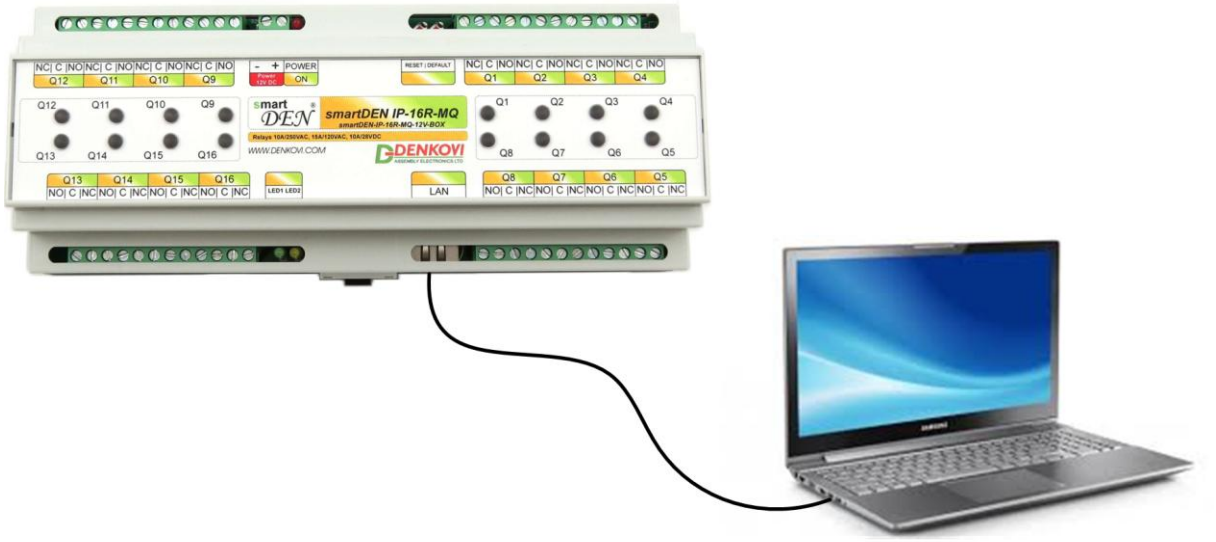

Figure 6.5. Connecting **smartDEN IP-16R-XX** to a computer directly. This is the recommend initial connection.

Figure 6.6. Connecting **smartDEN IP-16R-XX** to a wireless router.

6.5. Communication setup

smartDEN IP-16R-XX is shipped with the following default parameters:

- IP address: **192.168.1.100**
- Subnet mask: **255.255.255.0**
- Gateway: **192.168.1.1**
- Web password: **admin**

Initially it is recommended to connect the module directly to the computer.

Next you have to change your PC's IP address.

You can google how to change you computer IP settings or just visit this web page: http://www.howtochangeipaddress.com/changeip.php

For Windows 7 OS for example you can do that in the following way:
Navigate to *Control Panel -> Network and Internet -> View network and status tasks -> Change adapter settings*

Then just select the local area connection with right click and select *Properties*:

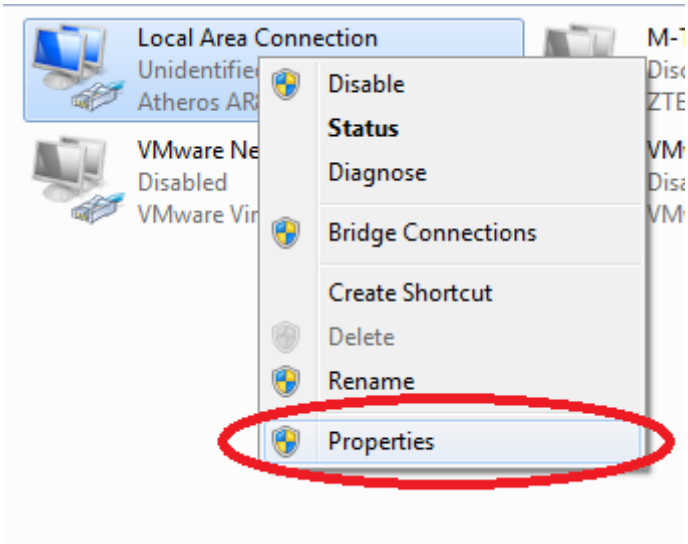

Figure 6.7. LAN card properties

The next step is to enter into IPv4 properties.

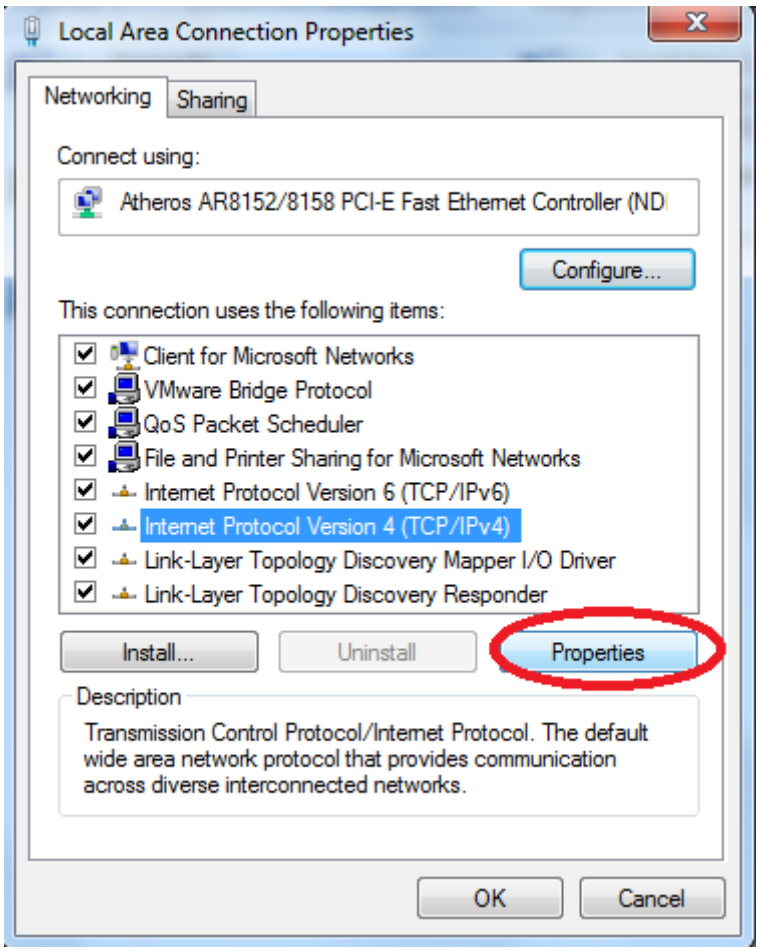

Figure 6.8. Enter in IPv4 properties section

Set the IP address of your PC to be in the same network.

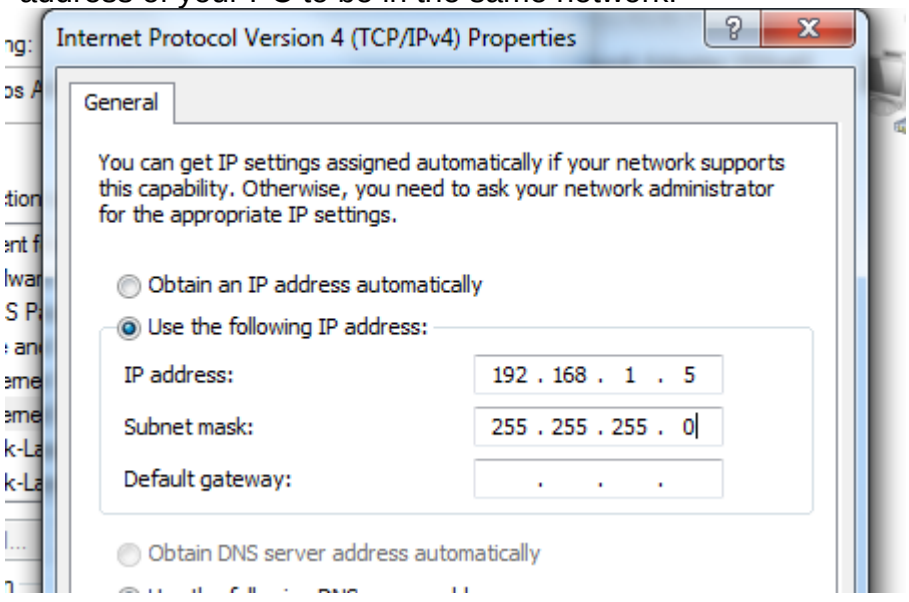

Figure 6.9. Set the IP address

Finally, in order to access **smartDEN IP-16R-XX** just type in your browser 192.168.1.100

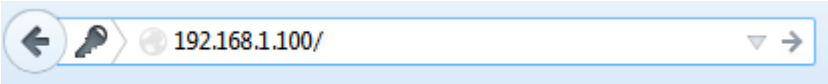

Figure 6.10. Open the device via browser

If the network settings are O'K, the log-in page should appear:

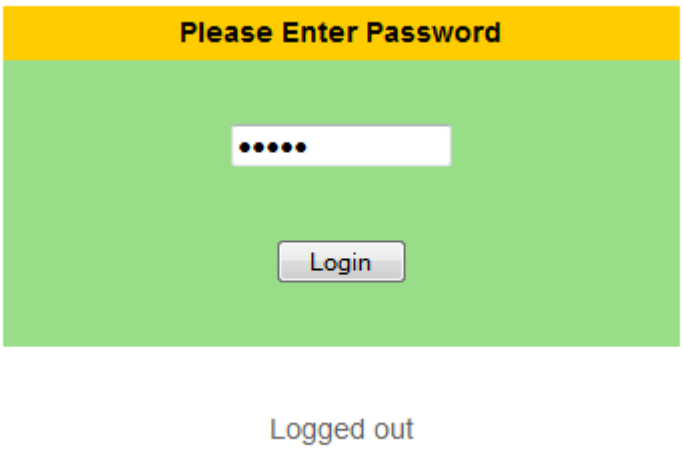

Figure 6.11. Login page

smartDEN IP-16R-XX modules connected locally can be easily scanned and found via the tool ***Denkovi Finder*** as well.

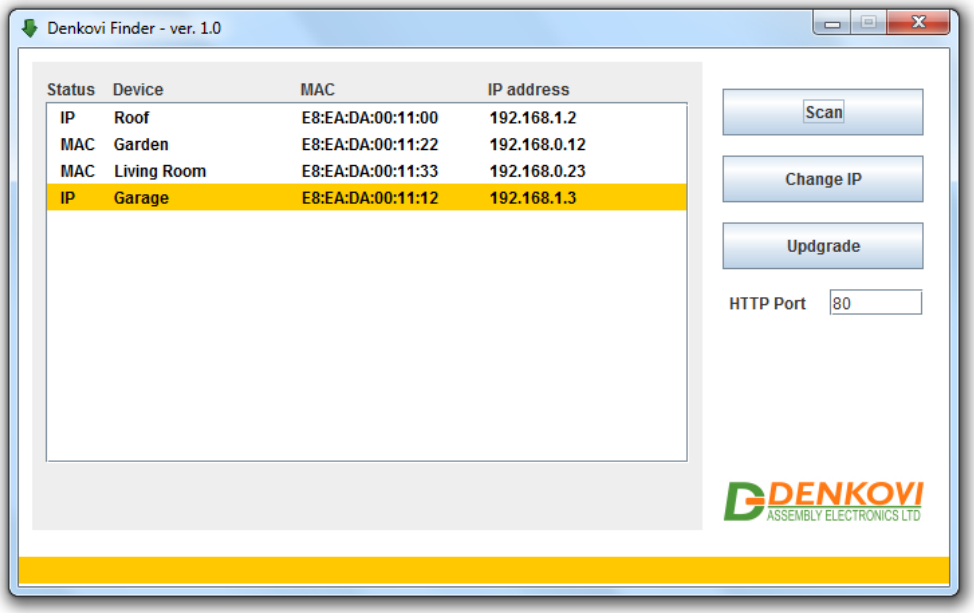

Figure 6.12. Denkovi Finder

7. Default Settings

7.1. Table with default settings

The **smartDEN IP-16R-XX** module is shipped with default (factory) settings shown in below table. The default settings can be reloaded, if necessary (see Steps for loading default settings).

Table 7.1. Default settings

Settings group	Parameter (according Web pages)	Value
Common Parameters		
General Settings	Save Outputs	No
	Password	admin
Network Settings	DHCP	Disabled
	IP Address	192.168.1.100
	Gateway	192.168.1.1
	Subnet Mask	255.255.255.0
	Primary DNS	192.168.1.1
	Secondary DNS	0.0.0.0
HTTP/XML/JSON Access	HTTP Port	80
	Access IP Address	192.168.1.0
	Access Mask	0.0.0.0
	Access MAC Address	00:00:00:00:00:00
	Session Timeout, min	3
	Enable Access	Yes
	Encrypt Password	No
	Multiple Access	Yes
For smartDEN IP-16R only		
SNMP Agent	Enable SNMP	Yes
	SNMP Port	161
	Read-only Community1	public
	Read-only Community2	read
	Read-write Community1	private
	Read-write Community2	write
For smartDEN IP-16R-MT only		
Modbus-TCP	Enable Modbus-TCP	Yes
	Modbus-TCP Port	502
	Idle Timeout, min	5
For smartDEN IP-16R-MQ only		
MQTT Settings	Enable	No

7.2. Steps for loading default settings

When necessary, the factory (default settings) may be applied so the module parameters will be returned back as those in **point 6.1** from the current document.

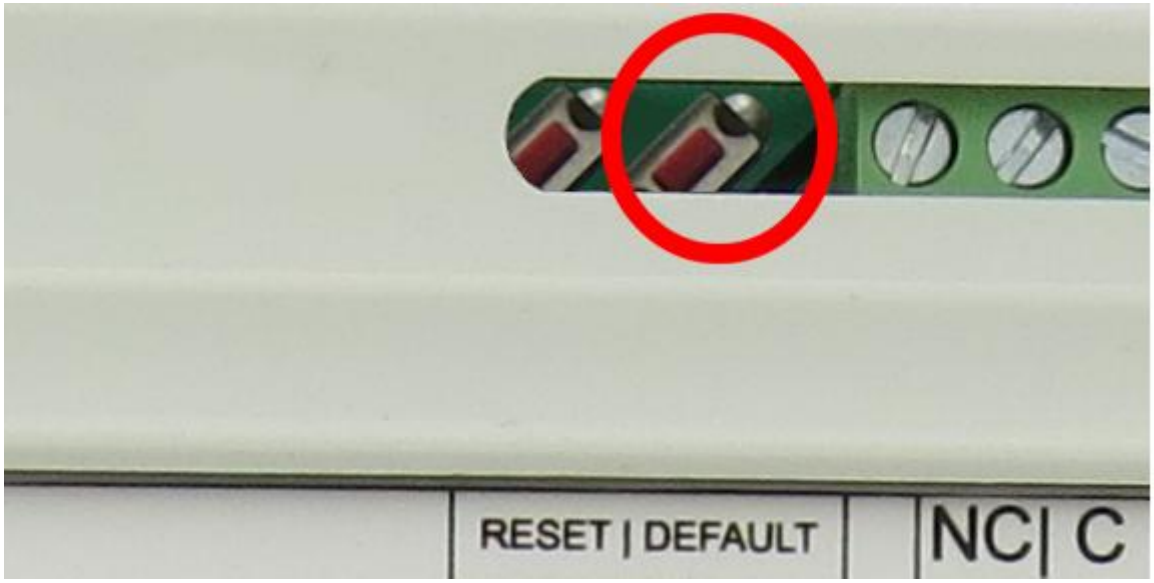

Figure 7.1. Loading the default settings

1. Turn off the power supply of the device;
2. Press and hold the default button;
3. Turn on the power supply of the device;
4. Wait for until both led indicators (led1 and led2) become ON (approximately 10 sec);
5. Release the default button;
6. The module is configured with default settings.

8. Web access

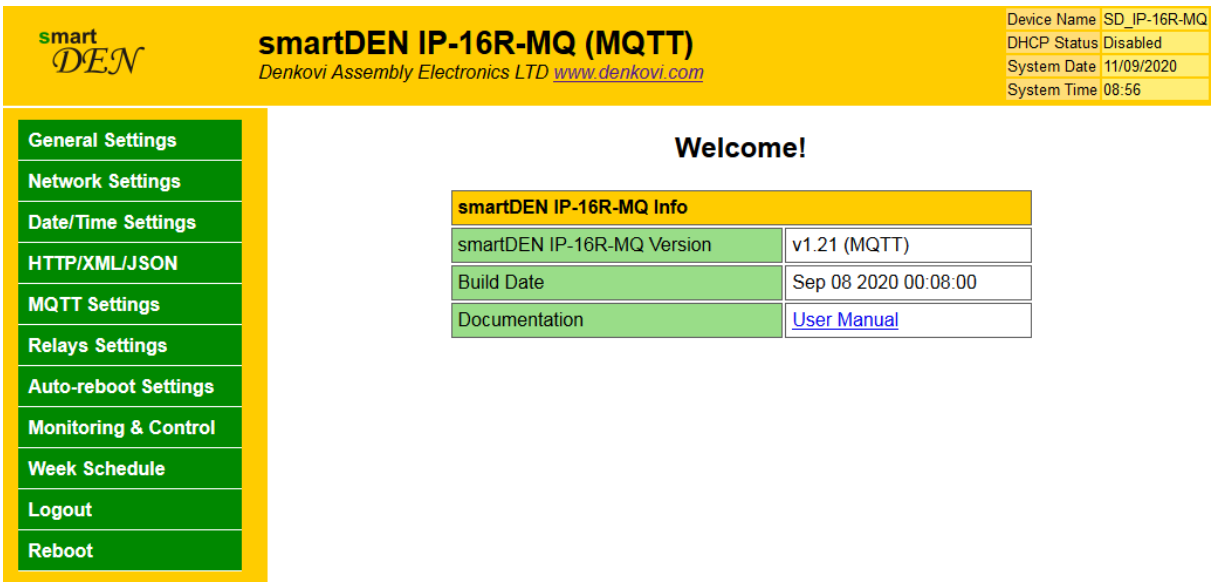

Figure 8.1. Web access

To access the setup pages, run a web browser (Internet Explorer, Mozilla Firefox or similar), and enter the **smartDEN IP-16R-XX** IP address, for example: http://192.168.1.100

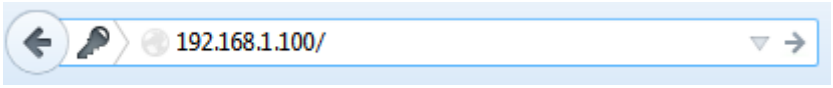

Figure 8.2. Open via browser

Note: You will need to have JavaScript enabled in your browser.

8.1. Login

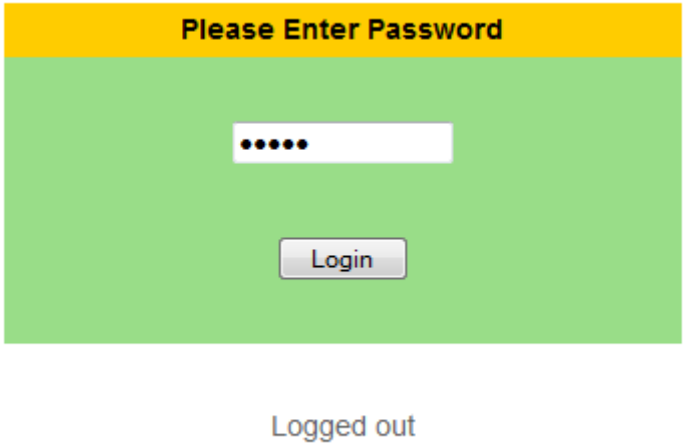

Figure 8.3. Login page

Enter the password and click "Login" button. This will bring you to the **smartDEN IP-16R-XX** main configuration page which contains details for the current firmware version and build date and provides buttons and links to obtain further details.

Note: The default password is admin (passwords are case sensitive).

Note: When the password is entered, it is transmitted across the network in encrypted form, so eavesdropping on the data transmission will not reveal the password.

Note: In order to prevent setup/control conflicts, at any given moment, only one user can be logged in.

Note: If there is no data traffic between the Web-browser and the **smartDEN IP-16R-XX** for time, specified by ***Session Timeout*** parameter, the session "times out" and a new login is required.

8.2. General Settings

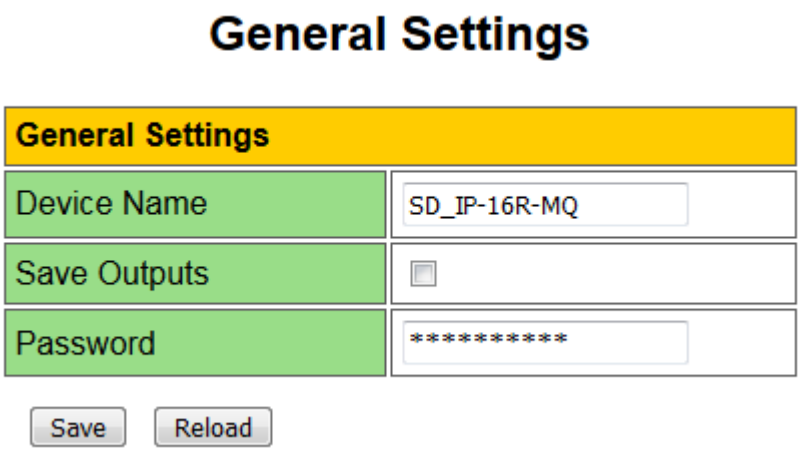

Figure 8.4. General settings

- **Device Name:** The name of the module (max 15 symbols). Every module can have different name in your network so they can be distinguished;
- **Save Outputs:** When checked, each time the relays state is changed, it will be saved in non-volatile memory (EEPROM), so after reboot/restart it will be restored;

This option should be used with care in dynamic systems because of restriction in maximum write cycles of the EEPROM (usually 100 000 write/erase cycles).

- **Password:** The password used for logging into the web admin and XML operation (max. 10 chars);

When typed. the password in this screen is not hidden. Only in this case, when the password is being changed, it is transmitted across the network "in the open". Therefore, set passwords in a secure environment where you can make sure that no one is "eavesdropping". Subsequent transmissions of the password to "login" onto the device are encrypted and "safe".

- **Save button:** Once you have changed the settings as required, click this button.

8.3. Network settings

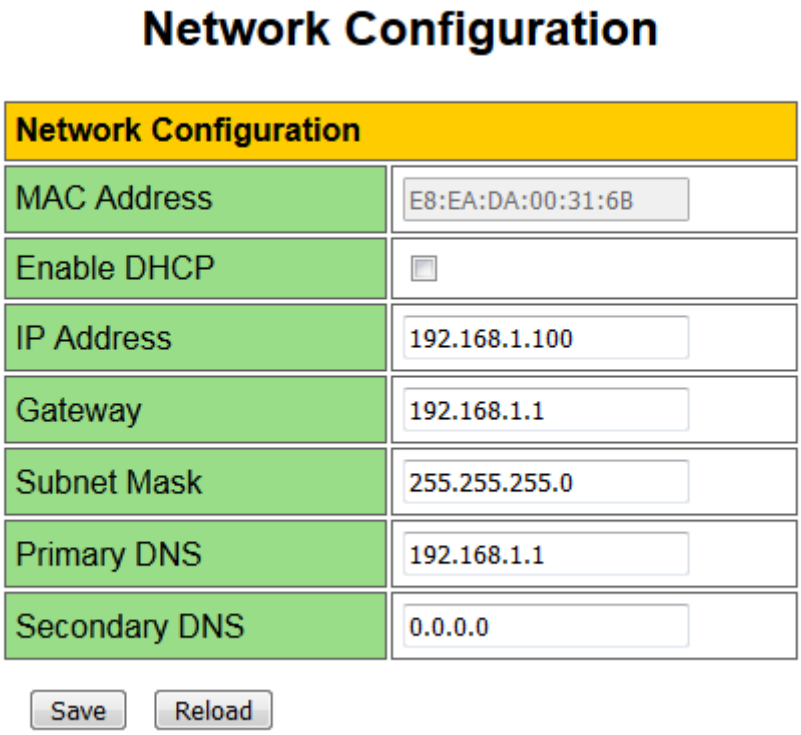

Figure 8.5. Network settings

This menu lets you configure the network settings of **smartDEN IP-16R-XX** relay module:

- **Enable DHCP:** This option allows DHCP to be enabled or disabled. If DHCP is set to Enabled, the Network page must be saved and **smartDEN IP-16R-XX** must be rebooted before obtaining an IP address;
- **IP address:** This is the IP address of the **smartDEN IP-16R-XX**. It needs to be manually assigned only if DHCP is disabled. With DHCP enabled, this field displays the currently assigned address;
- **Gateway:** This specifies the IP address of the gateway router. It is used for accessing public time servers for automatic time synchronization;
- **Subnet Mask:** This is the subnet mask for the network on which the **smartDEN IP-16R-XX** is installed;
- **Primary DNS:** Primary DNS (Domain Name Service) address;
- **Secondary DNS:** Secondary DNS address;
- **Save button:** Once you have changed the settings as required, click this button.

You have to reboot the device for these settings to apply.

8.4. Date and Time Settings

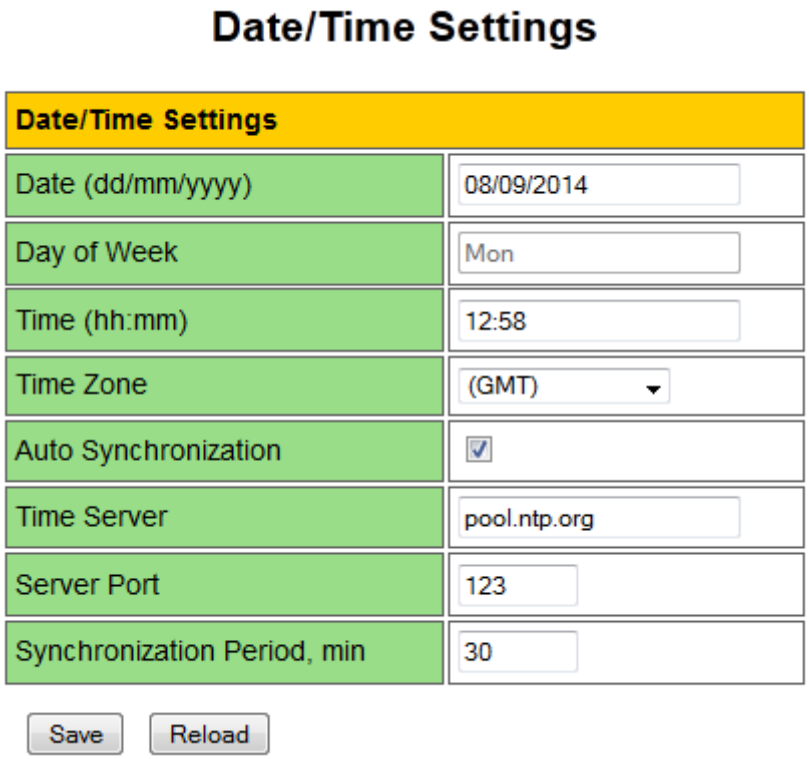

Figure 8.6. Date/Time settings

This page lets you configure the following parameters related with the real time clock built-in the module:

- **Date (dd/mm/yyyy):** Enter the current date here in specified format;
- **Time (hh:mm):** Enter the current time here in 24-hour format;
- **Time Zone:** Select the time zone for your geographic location.
- **Auto Synchronization:** This option enables or disables automatic synchronization with the SNTP (Simple Network Time Protocol) server with period specified by **Synchronization Period**;
- **Time Sever:** This is the SNTP server, used for synchronizing the time automatically;
- **Server Port:** SNTP server port;
- **Synchronization Period, min:** This option sets the period in which automatic synchronization will take place, if enabled;
- **Save button:** Once you have changed the settings as needed, click "**Save**". These settings apply immediately and do not require a reboot.

8.5. HTTP/XML/JSON Settings

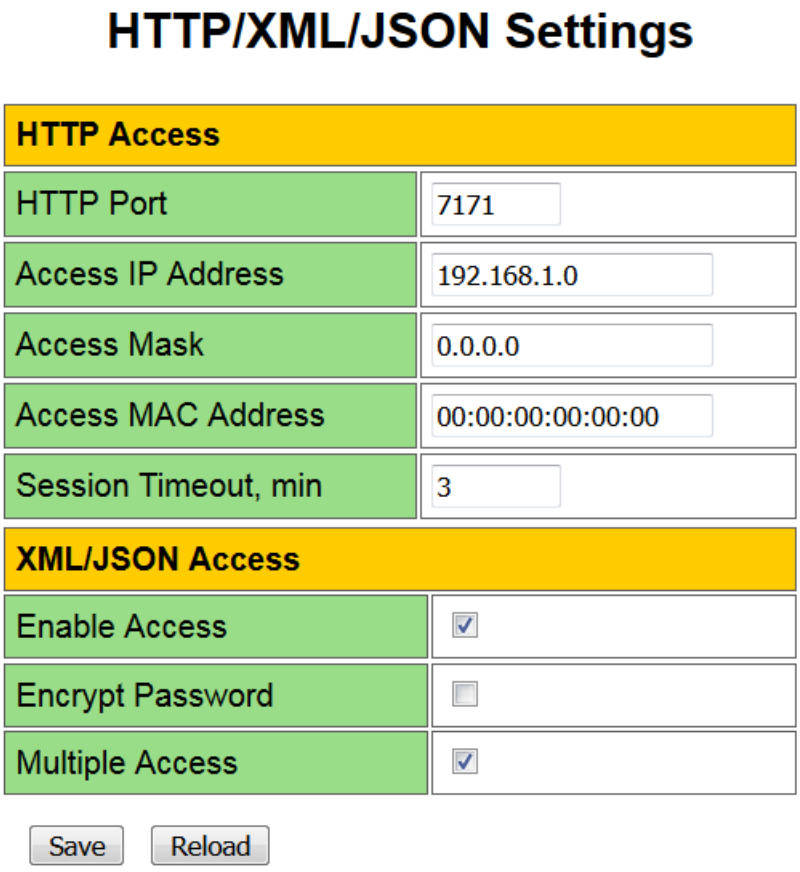

Figure 8.7. HTTP/XML/JSON Settings

These settings let you configure the HTTP, XML and JSON access parameters of **smartDEN IP-16R-XX:**

- **HTTP Port:** Port that the Web server listens for HTTP requests (default port is 80). You have to reboot the device for a new port setting to apply;
- **Access IP Address/Access Mask:** These fields can be used to restrict the HTTP/XML/JSON access by specifying the IP address and subnet mask of the HTTP client;
- **Access MAC Address:** This field can be used to restrict the HTTP/XML/JSON access by specifying the MAC address of the HTTP client;
- **Session Timeout, min:** Specifies the timeout period for HTTP, XML and JSON sessions in minutes;
- **Enable Access:** This option enables or disables XML/JSON access to the **smartDEN IP-16R-XX**;
- **Encrypt Password:** When XML/JSON access is enabled, this option adds additional security level by encrypting the login password;
- **Multiple Access:** This option enables simultaneous access from several HTTP/JSON clients;
- **Save button:** Once you have changed the settings as required, click this button.

Note: When **Encrypt Password** mode is enabled, the **Multiple Access** option is not taken into account and, at any given moment, only one user can be logged-in.

Note: When **Multiple Access** mode is enabled, any XML/JSON request will always reset the current HTTP session.

Note: When **Multiple Access** mode is disabled, whether **Encrypt Password** is enabled or not, it is possible to access the module via XML/JSON only after login for the specified session timeout.


You have to reboot the device for these settings to apply.


It is highly recommended to log out from the web server after finishing the parameters setup.


If you don’t want to restrict the HTTP/XML/JSON access by IP address, set the **Access Mask** to 0.0.0.0.


If you don’t want to restrict the HTTP/XML/JSON access by MAC address, set the MAC **Address** to 00:00:00:00:00:00.


Setting the **Access Mask** to 255.255.255.255 allows the HTTP/XML/JSON access only from the exactly specified **Access IP Address**.


You can allow the HTTP/XML/JSON access to a range of IP addresses by setting an appropriate value for **Access Mask**. For example setting the **Access IP Address** to 192.168.1.0 and **Access Mask** to 255.255.255.0 allows the access from IP addresses in range from 192.168.1.0 to 192.168.1.255.

8.6. Relays Settings

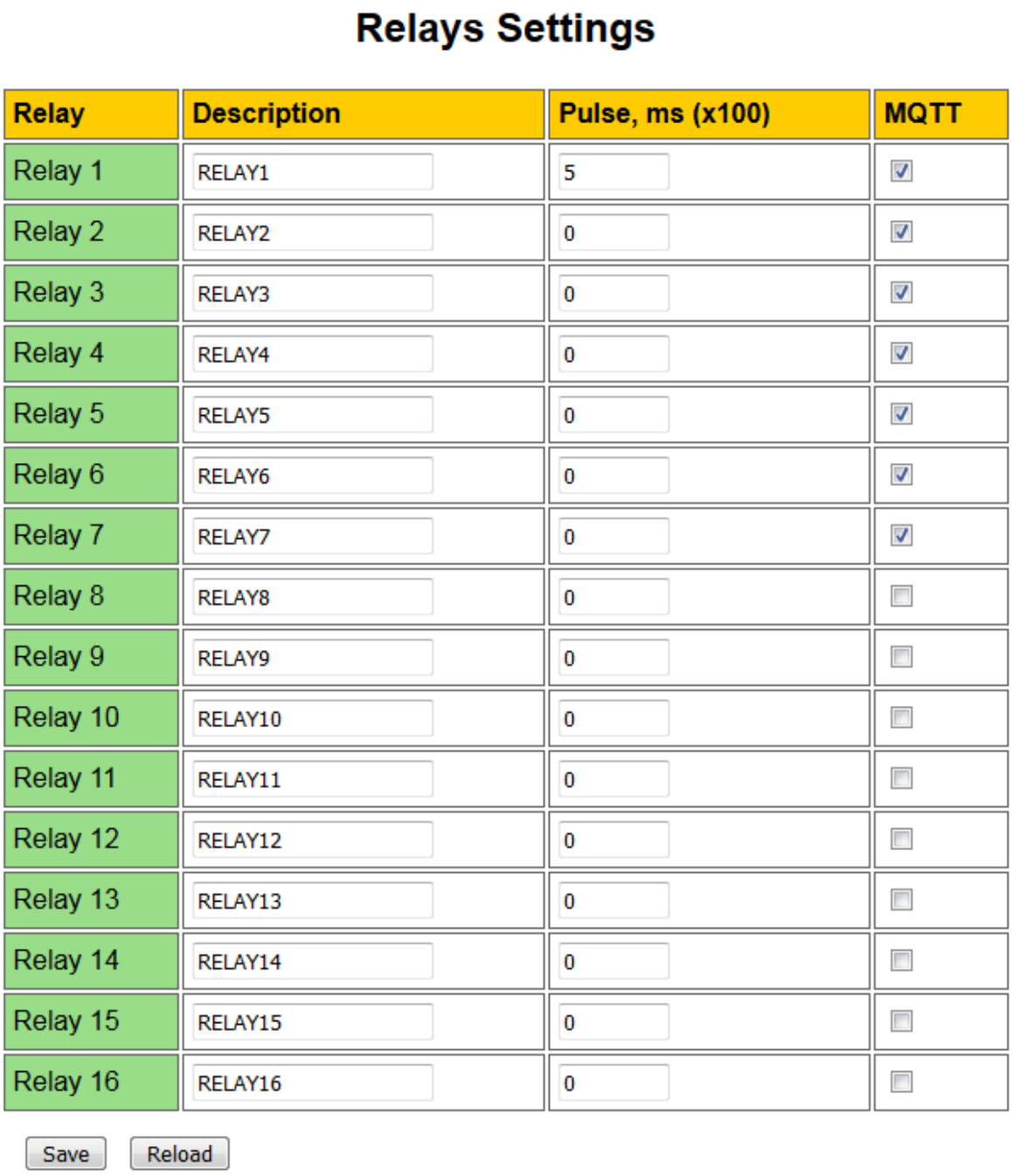

Figure 8.8. Relays names

This page configures the following parameters for the relays:

- **Description:** Relay identification string (max 7 chars).
- **Pulse, ms (x100)**: Determines if the relay works in pulse mode and if so what is the duration of the single pulse (in milliseconds x 100 - for example value of 10 means 1 second). This parameter can accept values between 0 and 65535. If it is 0, then the pulse mode is disabled. If it is between 1 and 65535, then the pulse mode for this relay is activated and it will be hold in high level

(ON) for the specified time by this parameter. During this time, the relay can be set in low level (OFF) via week schedule, via some of the integration protocols, web browser control, HTTP/XML/JSON:

- If the relay is set to high level (ON) via browser manually, via some of the integration protocols or HTTP/XML/JSON it will be in high level (ON) for the determined time by this parameter and then set back to low level;
- If the output is controlled by weekly schedule, then when the output becomes in high level, it will be set to high level for time, specified by this parameter and then will be set to low;

- **MQTT:** Enable/disable MQTT notification on relay state change (for **smartDEN IP-16R-MQ** only).

8.7. Auto-reboot Settings

smartDEN IP-16R-XX can be used for Auto-rebooting of IP devices (servers, PC hosts, switches, cameras etc.). In this mode **smartDEN IP-16R-XX** checks the working state of the device at regular intervals by sending ping requests. After preset number of successive ping failures **smartDEN IP-16R-XX** performs a predefined reset/reboot sequence of the monitored device connected to one of the 16 relays.

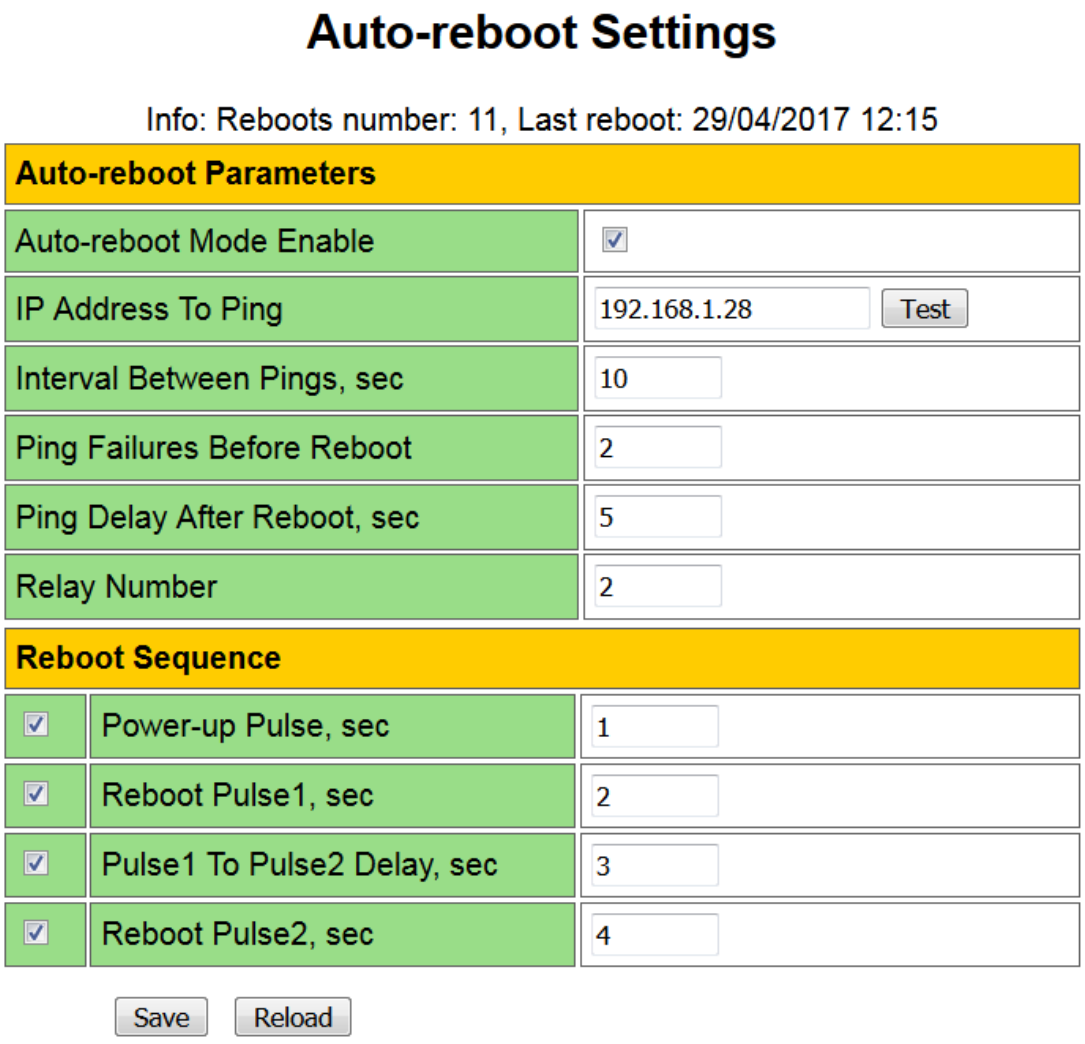

Figure 8.9. Auto-reboot configuration page

One of **smartDEN IP-16R-XX** channels can be configured to work in Auto-reboot mode:

- **Auto-reboot Mode Enable** - activate this mode;
- **IP Address To Ping** - the IP address of the device to be rebooted when pings will fail;
- **Interval Between Pings, sec** – the time interval between two sequential ping requests sent to IP address of the monitored device (from 1 to 3600 seconds);

- **Ping Failures Before Reboot** - the number of successive failed pings before the device is rebooted (from 1 to 100 pings);
- **Ping Delay After Reboot, sec** – the waiting period after reboot that should pass before the device is checked again (from 1 to 3600 seconds);
- **Relay Number** – relay used for rebooting;

The reboot circuit can be wired to common (C) and normally open (NO) or normally closed (NC) contacts. Relay state "ON" means that the NO contacts are closed and the NC contacts are open. Relay state "OFF" means that the NO contacts are open and the NC contacts are closed.

When the relay is wired in series with the power circuit of the device, the C and NC contacts should be used.

When the relay is wired in parallel with the "reset" or "power on/off" button the device, the C and NO contacts should be used.

When the associated relay is working in auto-reboot mode, when it is set ON via web browser, XML, JSON or Integration Protocol it will perform the reboot sequence.

- **Power-up Pulse, sec** – if checked, the **smartDEN IP-16R-XX** will generate a pulse at power-up. This, for example can be used to switch on the device. This parameter can be set from 1 to 3600 seconds;
- **Reboot Pulse1, sec** - if checked, the **smartDEN IP-16R-XX** will generate a pulse when the reboot condition is detected. This can be used to switch off or reset the device. Range is from 1 to 3600 seconds;
- **Pulse1 To Pulse2 Delay, sec** - if the device is switched off by Pulse1, the **smartDEN IP-16R-XX** will wait before generating Pulse2 to switch it on. This delay can be set from 1 to 3600 seconds;
- **Reboot Pulse2, sec** - if checked, the **smartDEN IP-16R-XX** will generate a second pulse to switch it on the device.

"Pulse" means that the relay switches ON for defined time and then switches OFF.

If the reboot circuit is wired in parallel to "reset" button of the device, only Reboot Pulse1 option can be checked.

If the relay is wired in series with the power circuit of the device, only Reboot Pulse1 option can be checked (the device is switched off when the relay is ON).

If the reboot circuit is wired in parallel to "power on/off" button of the device, Reboot Pulse1, Pulse1 To Pulse2 Delay and Reboot Pulse2, options can be checked.

8.8. Monitoring and control

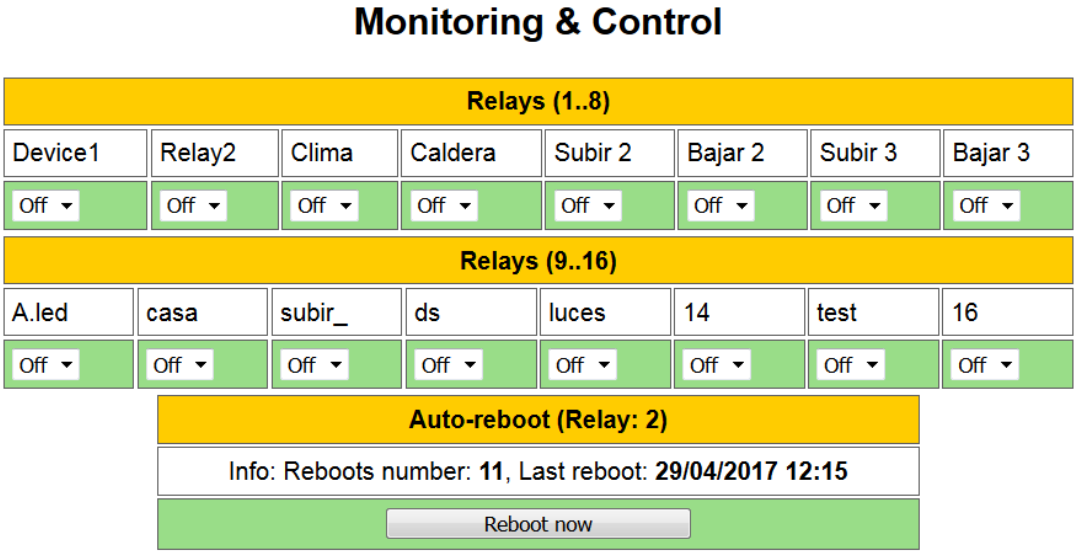

Figure 8.10. Monitoring and control

This page provides monitoring and control of the **smartDEN IP-16R-XX** relays. From here you can control/monitor the relays.

There is also provided information about how many reboots are performed and when was the last reboot and button for immediate reboot.

8.9. Week Schedule

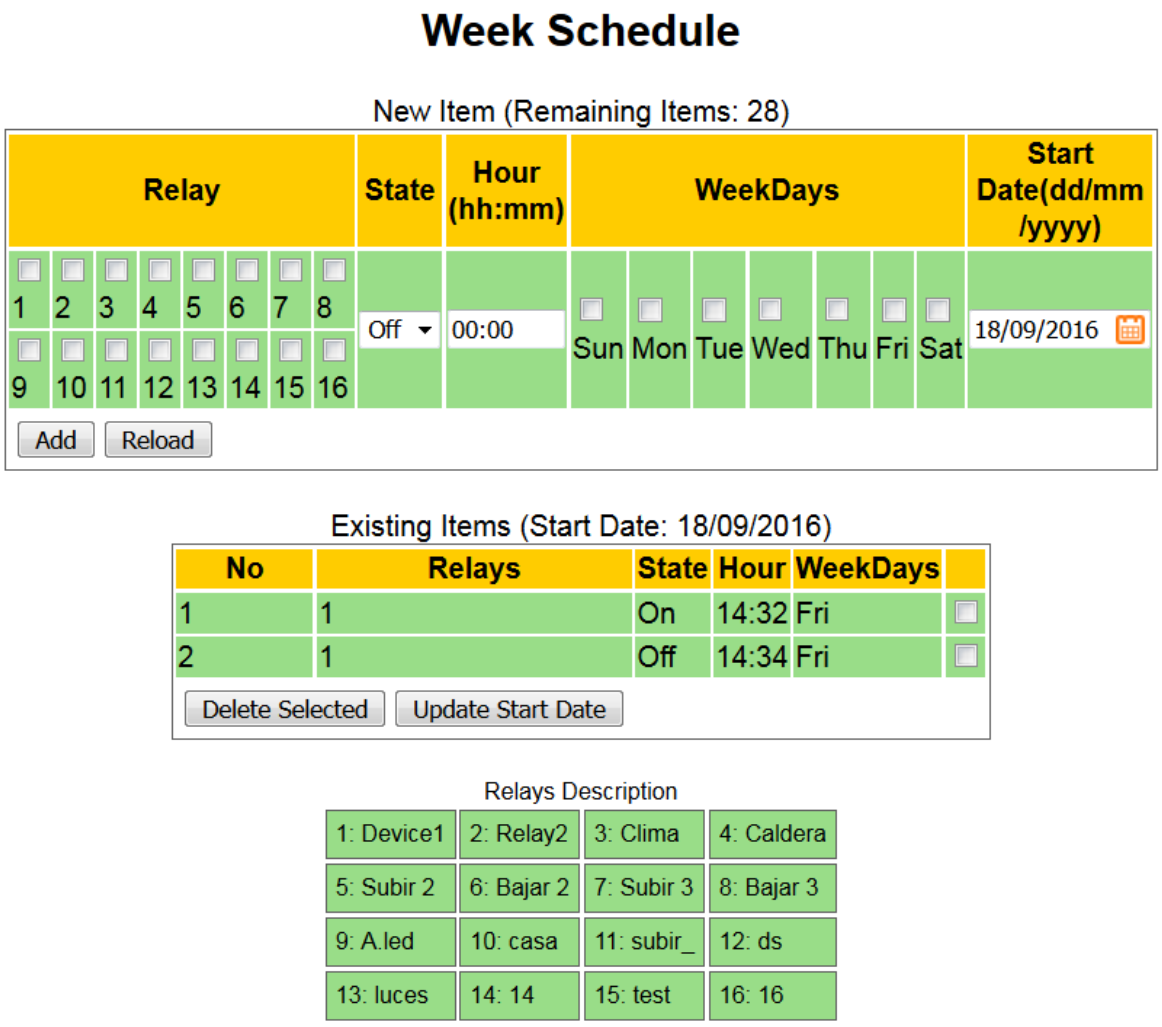

Figure 8.11. Week schedule

This page configures the **Week Schedule** table for switching **Relays** ON or OFF at specific times. You can add up to 30 items to the list. The top table of this page allows you to define a new item, while the bottom table shows the already defined list:

- **Outputs:** Select a group of relays that should be switched;
- **State:** Defines the state (ON/OFF) for the selected group of relays;
- **Hour:** Time the group of relays will be switched at;
- **WeekDays:** Select the days the defined switching should take place;
- **Start Date (dd/mm/yyyy):** The start date for the **Week Schedule** table.

Once you have defined a new item, click "**Add**". This item will be added as a new row in a **Week Schedule** table.

- This feature allows you to turn on/off specific relays upon certain date and time or weekday without the need of LAN connection between the computer and the module.

- To delete an item, select it in **Existing Items** table and click on "**Delete Selected**" button.

- To set a new start date, click on "Update Start Date" button.

- The module has back-up supply source for the RTC in order to keep the current date/time for several days during power off.

8.10. Logout

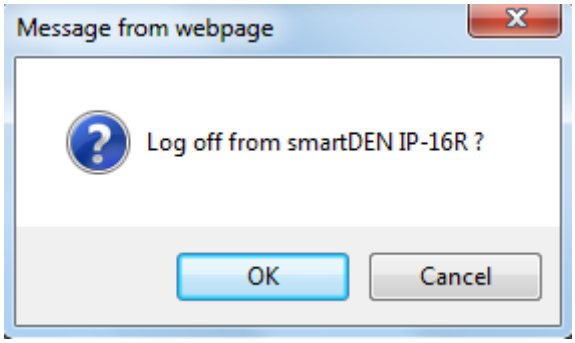

Figure 8.12. Log off

8.11. Reboot

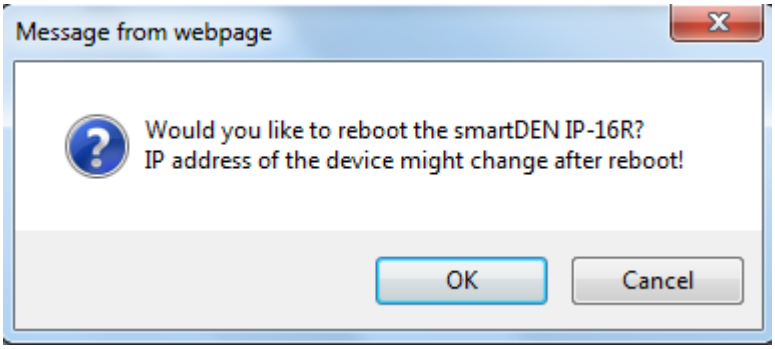

Figure 8.13. Reboot

8.12. SNMP Agent Configuration (for smartDEN IP-16R only)

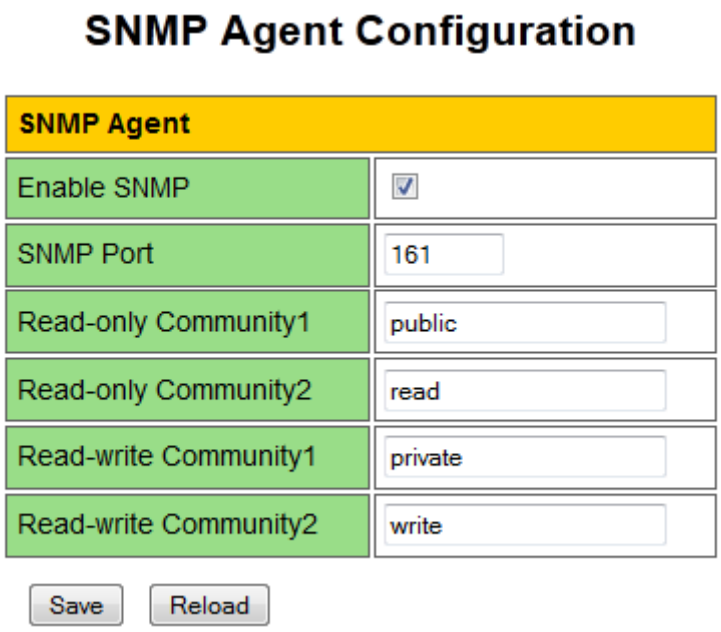

Figure 8.14. SNMP settings

These settings let you configure the SNMPv1 (Simple Network Management Protocol Version 1) access to the module**:**

- **Enable SNMP:** This option enables or disables SNMP access to the module;
- **SNMP Port:** UDP port number the SNMP agent receives requests on (default port is 161);
- **Read-only Community1/2:** Community string for client's authentication, used in read operations;
- **Read-write Community1/2:** Community string for client's authentication, used in read/write operations;
- **Save button:** Once you have changed the settings as required, click this button.

You have to reboot the device for these settings to apply.

8.13. Modbus-TCP Settings (for smartDEN IP-16R-MT only)

These settings let you configure the Modbus-TCP parameters of **smartDEN IP-16R-MT**

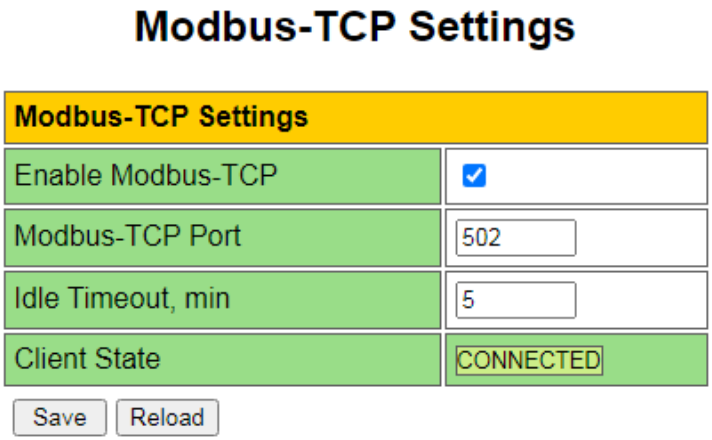

Figure 8.15. Modbus-TCP Settings page

- **Enable Modbus-TCP** - this option enables or disables the Modbus-TCP communication;
- **Modbus-TCP Port** - port on which the Modbus-TCP server listens for requests (default port is 502);
- **Idle Timeout, min** - the length of time that a connection is idle before the connection is closed by the Modbus-TCP server. The timeout is disabled when its value is set to 0;
- **Client State** - Modbus-TCP client state;
- **Save** button - once you have changed the settings as required, click this button.

You have to reboot the device for these settings to apply.

Please note that only one TCP socket connection is supported at a time. Multiple concurrent TCP connections are not supported.

To refresh the Modbus-TCP client state, click the Reload button.

8.14. MQTT Settings (for smartDEN IP-16R-MQ only)

These settings let you configure the MQTT communication of **smartDEN IP-16R-MQ.**

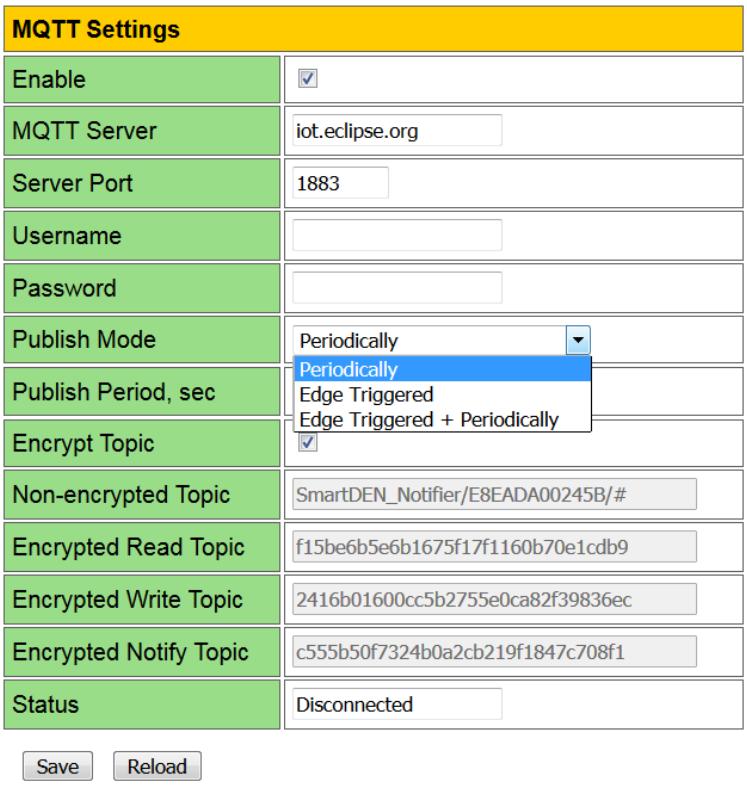

Figure 8.16. MQTT settings

- **Enable** – enable/disable the MQTT protocol;
- **MQTT Server** - the MQTT server (broker) IP address or domain name (max. 22 chars);
- **Server Port** – the MQTT server listening port (the default port is 1883);
- **Username** – username, if used (max. 10 chars), used for encrypted mode only;
- **Password** – password, if used (max. 10 chars), used for encrypted mode only;
- **Publish Mode**:
  - **Periodically** – notifications are send periodically (the period is defined by the **Publish Period** value);
  - **Edge Triggered** – notifications are triggered by events (change of digital inputs state or crossing of analog value/temperature over or below the predefined threshold values);
  - **Edge Triggered + Periodically** - combination of the above two modes. In that mode notifications are send just like the mode **Edge Triggered** but periodically (the period is defined by the **Publish Period** value);
- **Publish Period, sec** – the time interval between two successive notifications (from 5 to 250 seconds);
- **Encrypt Topic** – when enabled, the MQTT topic is encrypted;

- **Non-encrypted Topic** – plain (non-encrypted) topic the clients can subscribe to receive notifications from **smartDEN IP-16R-MQ**;
- **Encrypted Read Topic**, **Encrypted Write Topic**, **Encrypted Notify Topic** - encrypted topics for communication with the Android DAE-aModules application or customized MQTT libraries.
- **Save button** - once you have changed the settings as required, click this button.

A list of sample publically-accessible MQTT servers (brokers): iot.eclipse.org, test.mosquitto.org, mqtt.fluux.io…

Two configuration options are used by default: Publish QoS (Quality of Service) = 0; Keep Alive value = 120 seconds.

The plain topics are supposed to be used with general MQTT clients.
The encrypted topics are designed for communication with the Android DAE-aModules application or customized MQTT libraries.

The communication protocol, used in working with encrypted topics, is available upon request.

You have to reboot the device for these settings to apply.

9. HTTP/XML/JSON access

This operation mode allows custom applications to control the **smartDEN IP-16R-XX** without using a Web-browser. The custom application acts as a HTTP client, sending HTTP GET requests to the **smartDEN IP-16R-XX**.

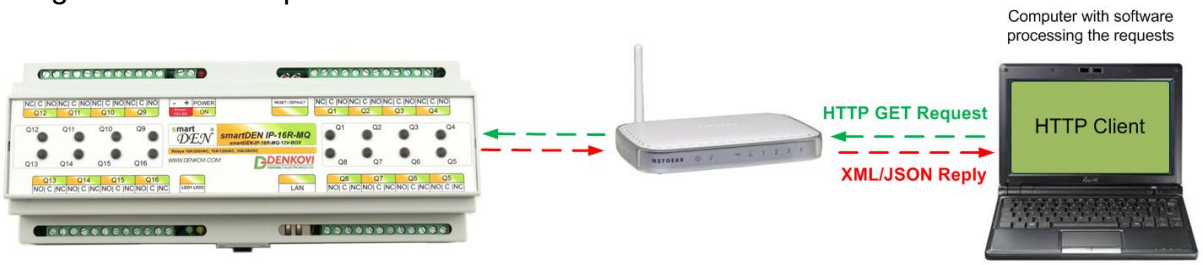

Figure 9.1. smartDEN IP-16R-XX working as a HTTP server

To receive the current state of the **smartDEN IP-16R-XX,** the application requests the page *current_state.xml (current_state.json)*, for example:

http://192.168.1.100/current_state.xml

http://192.168.1.100/current_state.json

The custom application can also control the **smartDEN IP-16R-XX** by sending parameters (name/value pairs) with the HTTP/XML/JSON GET request, for example:

http://192.168.1.100/current_state.xml?Relay=1&Relay2=0&Relay3=1

http://192.168.1.100/current_state.json?Relay=1&Relay2=0&Relay3=1

The XML/JSON login process differs depending on the selected **Encrypt Password** option.

9.1. Login (Encrypted Password)

In this mode a two-step login sequence is provided as a protection against unauthorized access. The first time the custom application requests the page *current_state.xml / current_state.json*, a random login key is issued in the reply. Next the custom application uses this key to encrypt the password. The encrypted password is sent as a parameter with the next request to the page *current_state.xml / current_state.json*.

Bellow is an example of login process:

Step 1:
Request
http://192.168.1.100/current_state.xml
Reply (login required):

<CurrentState>
<LoginKey>65156</LoginKey>
</CurrentState>

http://192.168.1.100/current_state.json
Reply (login required):
{"CurrentState": {"LoginKey": "65156"}}

Step 2:
Request (password is sent as a parameter)
http://192.168.1.100/current_state.xml?pw=28237099263eabfd88626124a822c64c
or
http://192.168.1.100/current_state.json?pw=28237099263eabfd88626124a822c64c

Reply (password is O'K, login accepted): See: *Appendix 2. Application reply formats*

Password encryption algorithm to be implemented in custom application is available upon request.

9.2. Login (Non-Encrypted Password)

In this mode the password is passed as non-encrypted parameter with the request:

http://192.168.1.100/current_state.xml?pw=admin

http://192.168.1.100/current_state.json?pw=admin

Getting the <LoginKey> in the answer in this mode means only that the provided password is wrong or the login session has been expired.

If there is no data traffic between the custom application and the **smartDEN IP-16R-XX** for time, specified by **Session Timeout** parameter, the session "times out" and a new login is required.

9.3. Getting the current state

After a login the custom application can obtain the **smartDEN IP-16R-XX** current state by a request to the page *current_state.xml / current_state.json* :

http://192.168.1.100/current_state.xml

The reply contains page in XML format, see: *Appendix 2. Application reply formats*

http://192.168.1.100/current_state.json

The reply contains page in JSON format, see: *Appendix 2. Application reply formats*

9.4. Multiple XML Access

In this mode the password should be passed as non-encrypted parameter with each request:

http://192.168.1.100/current_state.xml?pw=admin&Relay1=1

http://192.168.1.100/current_state.json?pw=admin&Relay1=1

Multiple XML/JSON Access is not allowed when **Encrypt Password** option is enabled.

9.5. Parameters

After a login the custom application can also control the **smartDEN IP-16R-XX** by sending parameters (name/value pairs) with the HTTP GET request.

Valid parameters and values are shown in the bellow tables.

9.5.1. smartDEN IP-16R

Table 9.1. Valid **smartDEN IP-16R** HTTP parameters

Name	Value	Description
Relayi	0...1	Relay**i** value (**i**=1...16)
SetAll	0...65535	Set all the relays with single command
Pulsei	1...65535	Generate pulse to an output (**i**=1...16)
pw	password	Required at login

9.5.2. smartDEN IP-16R-MT

Table 9.2. Valid **smartDEN IP-16R-MT** HTTP parameters

Name	Value	Description
Relayi	0...1	Relay**i** value (**i**=1...16)
SetAll	0...65535	Set all the relays with single command
Pulsei	1...65535	Generate pulse to an output (**i**=1...16)
Date	dd/mm/yyyy	Set date
Time	hh:mm	Set time
pw	password	Required at login

9.5.3. smartDEN IP-16R-MQ

Table 9.3. Valid **smartDEN IP-16R-MQ** HTTP parameters

Name	Value	Description
Relayi	0...1	Relayi value (i=1...16)
SetAll	0...65535	Set all the relays with single command
PulseOni	1...65535	Generate a positive (ON) pulse to relay (i=1...16)
PulseOffi	1...65535	Generate a negative (OFF) pulse to relay (i=1...16)
Descriptioni	string (max. 7 chars)	Set relay identification string
Date	dd/mm/yyyy	Set date
Time	hh:mm	Set time
pw	password	Required at login

10. Integration protocols

10.1. SNMP (for smartDEN IP-16R only)

smartDEN IP-16R supports SNMPv1 protocol – snmpget and snmpset. Most of the parameters can be configured/read via these commands. Read-only community string is used for reading and Read-Write Community String is used for changing the parameters. Parameters that can be changed, are grouped according to their functions in the tables below. To obtain a valid OID number it is necessary to replace the "x" symbol with the prefix "**.1.3.6.1.4.1.42505**". Also all the snmp commands are described in the MIB file**.**

During SNMP access, it must be used snmpget and snmpset only to one OID and not to group of OIDs.

10.1.1. Product

Table 10.1. Product parameters

OID	Name	Access	Description	Syntax
x.6.1.1.0	Name	read-only	Description of the module	DISPLAYSTRING
x.6.1.2.0	Version	read-only	Current firmware version	DISPLAYSTRING
x.6.1.3.0	Date	read-only	Current firmware version build date	DISPLAYSTRING

10.1.2. Setup

Table 10.2. Setup

Start OID	End OID	Name	Access	Description	Syntax
x.6.2.1.0	**...**	SystemDate	read-write	System Date (dd/mm/yyyy)	DISPLAYSTRING
x.6.2.2.0	**...**	SystemTime	read-write	System Time (hh:mm)	DISPLAYSTRING
x.6.2.3.1.2.0	**x.6.2.3.1.2.15**	RelayName	read-write	Relay Name (maxlen=7)	DISPLAYSTRING (SIZE (0..7))
x.6.2.3.1.3.0	**x.6.2.3.1.3.15**	RelayState	read-write	Relay State (off-0, on-1)	INTEGER {off(0), on(1) }
x.6.2.3.1.4.0	**x.6.2.3.1.4.15**	RelaySetPulsePeriod	read-write	Relay Set Pulse Period (0..65535), ms(x100)	INTEGER32 (0..65535)
x.6.2.3.1.5.0	**x.6.2.3.1.5.15**	RelayStartPulse	read-write	Relay Start Pulse (0..65535), ms(x100)	INTEGER32 (0..65535)

10.1.3. Control

Table 10.3. Control

OID	Name	Access	Description	Syntax
x.6.3.1.0	RelaysState	read-write	Access all the relays with single command	INTEGER32 (0..65535)
x.6.3.2.0	Reboot	read-write	Reboot the device	INTEGER (0..255)
x.6.3.3.0	sysUpTime	read-only	The time (in hundredths of a second) since the device was last re-initialized.	TIMETICKS

10.1.4. Week Schedule

Table 10.4. Week schedule parameters

Start OID	End OID	Name	Access	Description	Syntax
x.6.4.1.0	**...**	WeekScheduleStartDate	read-write	Week Schedule Start Date (dd/mm/yyyy)	DISPLAYSTRING
x.6.4.2.1.2.0	**x.6.4.2.1.2.29**	Enabled	read-write	Week Schedule Row Enable Flag (Disabled-0, Enabled-1)	INTEGER { no(0),yes(1) }
x.6.4.2.1.3.0	**x.6.4.2.1.3.29**	Outputs	read-write	Outputs Code (0..65535), Output1 - bit 0, ..., Output16 - bit 15	INTEGER32 (0..65535)
x.6.4.2.1.4.0	**x.6.4.2.1.4.29**	OutputsState	read-write	Outputs Code (0..65535), Output1 - bit 0, ..., Output16 - bit 15	Outputs State (off-0, on-1)
x.6.4.2.1.5.0	**x.6.4.2.1.5.29**	Hour	read-write	Hour (hh:mm)	DISPLAYSTRING
x.6.4.2.1.6.0	**x.6.4.2.1.6.29**	WeekDays	read-write	WeekDays Code (0..127), Sunday - bit 0, ..., Saturday - bit 6	INTEGER (0..127)

To reboot the device via SNMP, set the Reboot value to the ASCII code of the first char of your Web password. For example, if this is the char 'a', code in decimal is 97.

10.1.5. Auto-reboot

Table 10.5. Auto-reboot settings

Start OID	Name	Access	Description	Syntax
x.6.5.1.0	AutorebootModeEnable	read-write	Auto-reboot Mode Enable (Disabled-0, Enabled-1)	INTEGER {no(0),yes(1) }
x.6.5.2.0	IPAddressToPing	read-write	IP Address To Ping	DISPLAYSTRING (SIZE (0..23))
x.6.5.3.0	IntervalBetweenPings	read-write	Interval Between Pings, sec (1..3600)	INTEGER32 (1..3600)
x.6.5.4.0	PingFailuresBeforeReboot	read-write	Ping Failures Before Reboot (1..100)	INTEGER32 (1..100)
x.6.5.5.0	PingDelayAfterReboot	read-write	Ping Delay After Reboot, sec (1..3600)	INTEGER32 (1..3600)
x.6.5.6.0	RelayNumber	read-write	Relay Number (1..16)	INTEGER32 (1..16)
x.6.5.7.0	PowerUpPulseEnable	read-write	Power-up Pulse Enable (Disabled-0, Enabled-1)	INTEGER {no(0),yes(1) }
x.6.5.8.0	PowerUpPulse	read-write	Power-up Pulse, sec (1..3600)	INTEGER32 (1..3600)
x.6.5.9.0	RebootPulse1Enable	read-write	Reboot Pulse1 Enable (Disabled-0, Enabled-1)	INTEGER {no(0),yes(1) }
x.6.5.10.0	RebootPulse1	read-write	INTEGER32 (1..3600)	Reboot Pulse1, sec (1..3600)
x.6.5.11.0	Pulse1ToPulse2DelayEnable	read-write	Pulse1 To Pulse2 Delay Enable (Disabled-0, Enabled-1)	INTEGER {no(0),yes(1) }
x.6.5.12.0	Pulse1ToPulse2Delay	read-write	Pulse1 To Pulse2 Delay, sec (1..3600)	INTEGER32 (1..3600)
x.6.5.13.0	RebootPulse2Enable	read-write	Reboot Pulse2 Enable (Disabled-0, Enabled-1)	INTEGER {no(0),yes(1) }
x.6.5.14.0	RebootPulse2	read-write	Reboot Pulse2, sec (1..3600)	INTEGER32 (1..3600)
x.6.5.15.0	AutoRebootsNumber	read-only	Auto-reboots Number	INTEGER32
x.6.5.16.0	LastAutoRebootTime	read-only	Last Auto-reboot Time	DISPLAYSTRING

10.2. Modbus TCP (for smartDEN IP-16R-MT only)

Modbus-TCP is an application layer messaging protocol, which provides master/slave (client/server) communication between devices connected on Ethernet networks. A Modbus-TCP message consists of a header (7 bytes) and the protocol data unit, which is encapsulated by the transmitting device into a standard TCP frame (Figure 10.1).

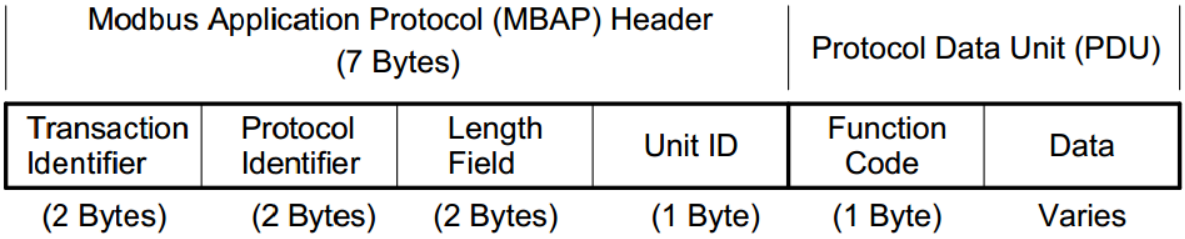

Figure 10.1. Modbus-TCP message format

The MBAP header includes the following fields:

- **Transaction Identifier** - used for transaction pairing when multiple messages are sent along the same TCP connection by a client without waiting for a prior response;
- **Protocol Identifier** - this field is always set to 0 for Modbus-TCP services;
- **Length** – number of bytes in the remaining fields (unit identifier byte, function code byte, and data fields);
- **Unit Identifier** - used to identify a remote server located on a non TCP/IP network (for serial bridging). In a typical Modbus-TCP server application, the unit ID is set to 0;

The function code field of the message contains one byte that specifies what kind of action the slave needs to take. When the server responds to the client, it echoes the same function code to indicate a normal (error-free) response. If the server cannot process a request, it will instead return an error function code (exception response) that is the original function code plus 80h (i.e. with its most significant bit set to 1).

Modbus-TCP uses a 'big-Endian' representation for addresses and data fields (when a numerical quantity larger than a single byte is transmitted, the most significant byte is sent first).

smartDEN IP-16R-MT acts as a Modbus-TCP slave/server device, while a typical master device is a host computer running appropriate application software (for example a Raspberry Pi board running Home Assistant automation platform).

10.2.1. MODBUS Commands

This following table shows the standard Modbus-TCP functions supported by the **smartDEN IP-16R-MT** module:

Table 10.6. Modbus commands

Function	Code	Resources	Access
Read Coil Status	01 (0x01)	Relays 1..16	bit
Write Single Coil	05 (0x05)	Relays 1..16	bit
Write Multiple Coils	15 (0x0F)	Relays 1..16	bit
Read Holding Registers	03 (0x03)	Configuration parameters	16-bit
Write Single Register	06 (0x06)	Generate pulses, Configuration parameters,	16-bit
Write Multiple Registers	16 (0x10)	Generate pulses, Configuration parameters	16-bit

smartDEN IP-16R-MT uses the following registers to represent the resources accessed by a Modbus command:

Table 10.7. Modbus registers

Resources	Start Address	End Address	Value Range
Generate Pulses on Relays 1..16 (write only)	0x0000	0x000F	1..65535
Set Pulse Width for Relays 1..16	0x6000	0x600F	0..65535
Week Schedule Start Date (Day)	0x6100	0x6100	1..31
Week Schedule Start Date (Month)	0x6101	0x6101	1..12
Week Schedule Start Date (Year)	0x6102	0x6102	2000..2099
Week Schedule Row Enable Flag	0x6200	0x621D	0..1
Week Schedule Row Outputs Code	0x6300	0x631D	0..65535
Week Schedule Row Outputs State	0x6400	0x641D	0..1
Week Schedule Row Hour	0x6500	0x651D	0..23
Week Schedule Row Minute	0x6600	0x661D	0..59
Week Schedule Row WeekDays Code	0x6700	0x671D	0..127
Save Outputs Option	0x6800	0x6800	0..1
System Date (Day)	0x6900	0x6900	1..31
System Date (Month)	0x6901	0x6901	1..12
System Date (Year)	0x6902	0x6902	2000..2099
System Time (Hour)	0x6903	0x6903	0..23
System Time (Minutes)	0x6904	0x6904	0..59
Firmware Version (read only)	0x6A00	0x6A00	

Pulse width is specified in milliseconds x 100. For example, a value of 30 will generate a pulse with 3 seconds duration.

Week schedule table has 30 rows. The row number (starting from zero) is defined by the low significant byte of the address.

The least significant bit (LSB) of the Outputs Code corresponds to Relay 1, the most significant bit (MSB) – to Relay 16.

The least significant bit (LSB) of the WeekDays Code corresponds to Sunday, the most significant bit (MSB) – to Saturday.

10.2.1.1. Read Coil Status

This command is used to read the ON/OFF status of the output relays (coils).

Request

The Read Coil Status request specifies the starting address and quantity of relays to be read:

- Start Address: 0x0000 (Relay 1) to 0x000F (Relay 16)
- Coil Quantity: 0x0001 (1 Relay) to 0x0010 (16 Relays)

Relays are addressed starting from zero (relays 1–16 are addressed as 0–15).

Note: If the sum of the start address and coil quantity exceeds 16, an error response will be returned.

Request example 1: Read Coil Status: Relays 1..3:

Table 10.8. Read Coils Request Example 1

Field	Length	Data
Transaction Identifier	2 Bytes	0x0001
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0006
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x01
Start Address	2 Bytes	0x0000
Quantity of Coils	2 Bytes	0x0003

Request example 2: Read Coil Status: Relays 4..16:

Table 10.9. Read Coils Request Example 2

Field	Length	Data
Transaction Identifier	2 Bytes	0x0001
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0006
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x01
Start Address	2 Bytes	0x0003
Quantity of Coils	2 Bytes	0x000D

Response

The coils in the response message are packed as one coil per bit of the data field. Status is indicated as 1 for ON and 0 for OFF. The least significant bit (LSB) of the first data byte contains the relay addressed in the query. The other relays follow toward the high order end of this byte, and from low order to high order in subsequent bytes. If the returned output quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeros. The Byte Count field specifies the quantity of complete bytes of data.

Response example 1: Read Coil Status: Relays 1..3:

Table 10.10. Read Coils response example 1

Field	Length	Data
Transaction Identifier	2 Bytes	0x0001
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0004
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x01
Byte Count	1 Byte	0x01
Coil Status	1 Byte	0x05

Coils Status							
bit 7 (MSB)	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0 (LSB)
0	0	0	0	0	1 (Relay 3)	0 (Relay 2)	1 (Relay 1)

In this example relays states are:

- Relay 1: ON
- Relay 2: OFF
- Relay 3: ON

Response example 2: Read Coil Status: Relays 4..16:

Table 10.11. Read Coils response example 2

Field	**Length**	**Data**
Transaction Identifier	2 Bytes	0x0001
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0004
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x01
Byte Count	1 Byte	0x02
Coil Status	2 Bytes	0x590B

Byte	**Coils Status**							
	bit 7 (MSB)	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0 (LSB)
1	0 (Relay 11)	1 (Relay 10)	0 (Relay 9)	1 (Relay 8)	1 (Relay 7)	0 (Relay 6)	0 (Relay 5)	1 (Relay 4)
2	0	0	0	0 (Relay 16)	1 (Relay 15)	0 (Relay 14)	1 (Relay 13)	1 (Relay 12)

Error

The possible error responses for function code 0x01 are:

- Function Code (1 byte): 0x81 (0x80 + 0x01)
- Exception Codes (1 byte):
  - 0x01 – Function code not supported.
  - 0x02 – Incorrect combination of start address and coil quantity

Error response example:

Table 10.12. Read Coils error response

Field	**Length**	**Data**
Transaction Identifier	2 Bytes	0x0001
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0003
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x81
Exception Codes	1 Byte	0x01 or 0x02

10.2.1.1. Write Single Coil

This command will turn single relay output to ON or OFF.

Request

The Write Single Coil request specifies the address of the coil to be forced:

- Address: 0x0000 (Relay 1) to 0x000F (Relay 16)

A value of 0xFF00 requests the coil to be ON, a value of 0x0000 requests the coil to be OFF, and a value of 0xFF02 toggles the coil state. All other values are illegal and will not affect the coil state.

Note: If the address exceeds 16, an error response will be returned.

Request example: Write Single Coil 2 (Relay 2) to ON:

Table 10.13. Write Single Coil request

Field	Length	Data
Transaction Identifier	2 Bytes	0x0002
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0006
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x05
Output Address	2 Bytes	0x0001
Output Value	2 Bytes	0xFF00

Response

The response indicates the new state of the relay.

Response: Write Single Coil 2 (Relay 2) to ON:

Table 10.14. Write Single Coil reply

Field	Length	Data
Transaction Identifier	2 Bytes	0x0002
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0006
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x05
Output Address	2 Bytes	0x0001
Output Value	2 Bytes	0xFF00

Error

The possible error responses for function code 0x05 are:

- Function Code (1 byte): 0x85 (0x80 + 0x05)
- Exception Codes (1 byte):
  - 0x01 – Function code not supported
  - 0x02 – Incorrect relay address
  - 0x03 – Illegal relay value

Error response example:

Table 10.15. Write Single Coil error response

Field	Length	Data
Transaction Identifier	2 Bytes	0x0002
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0003
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x85
Exception Codes	1 Byte	0x01 or 0x02 or 0x03

10.2.1.2. Write Multiple Coils

This command will force each coil in a sequence of coils to either ON or OFF.

Request

The Write Multiple Coils request specifies the starting address and quantity of relays to be forced:

- Start Address: 0x0000 (Relay 1) to 0x000F (Relay 16)
- Coil Quantity: 0x0001 (1 Relay) to 0x0010 (16 Relays)

Note: If the sum of the start address and coil quantity exceeds 16, an error response will be returned.

The requested ON/OFF states are specified by contents of the request data field. A logical '1' in a bit position of the field requests the corresponding output to be ON. A logical '0' requests it to be OFF.

Request example 1: Set relays 1 and 3 to ON, and relay 2 to OFF:

Table 10.16. Write Multiple Coils request example 1

Field	Length	Data
Transaction Identifier	2 Bytes	0x0003
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0008
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x0F
Start Address	2 Bytes	0x0000
Quantity of Outputs	2 Bytes	0x0003
Byte Count	1 Byte	0x01
Outputs Value	1 Byte	0x05

Request example 2: Set relay 1 to OFF and relays 2..16 to ON:

Table 10.17. Write Multiple Coils request example 2

Field	Length	Data
Transaction Identifier	2 Bytes	0x0003
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0009
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x0F
Start Address	2 Bytes	0x0000
Quantity of Outputs	2 Bytes	0x0010

Byte Count	1 Byte	0x02
Outputs Value	2 Bytes	0xFEFF

In this example the first byte of the Outputs Value field corresponds to relays 8 to 1, and the second byte – to relays 16 to 9.

Response

The normal response returns the function code, starting address, and quantity of coils forced.

Response example 1: Set relays 1 and 3 to ON, and relay 2 to OFF:

Table 10.18. Write Multiple Coils response example 1

Field	Length	Data
Transaction Identifier	2 Bytes	0x0003
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0006
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x0F
Start Address	2 Bytes	0x0000
Quantity of Outputs	2 Bytes	0x0003

Response example 2: Set relay 1 to OFF and relays 2..16 to ON:

Table 10.19. Write Multiple Coils response example 2

Field	Length	Data
Transaction Identifier	2 Bytes	0x0003
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0006
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x0F
Start Address	2 Bytes	0x0000
Quantity of Outputs	2 Bytes	0x0010

Error

The possible error responses for function code 0x0F are:

- Function Code (1 byte): 0x8F (0x80 + 0x0F)
- Exception Codes (1 byte):
  - 0x01 – Function code not supported
  - 0x02 – Incorrect combination of start address and coil quantity

Error response example:

Table 10.20. Write Multiple Coils error response

Field	Length	Data
Transaction Identifier	2 Bytes	0x0003
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0003
Unit Identifier	1 Byte	0x00

Function Code	1 Byte	0x8F
Exception Codes	1 Byte	0x01 or 0x02

10.2.1.3. Read Holding Registers

This command is used to read the contents of a contiguous block of registers.

Request

The Read Holding Registers request specifies the starting register address and the number of registers to be read.

Note: If the sum of the start address and number of registers exceeds the size of the accessed block of registers, an error response will be returned.

Request example: Read Week Schedule Start Date fields:

Table 10.21. Read Holding Registers request

Field	Length	Data
Transaction Identifier	2 Bytes	0x0004
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0006
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x03
Starting Register Address	2 Bytes	0x6100
Quantity of Registers	2 Bytes	0x0003

Response

The Read Holding Registers response returns the function code, byte count, and register's values packed as two bytes per register.

Response example: Read Week Schedule Start Date fields:

Table 10.22. Read Holding Registers response

Field	Length	Data
Transaction Identifier	2 Bytes	0x0004
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0009
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x03
Byte Count	1 Byte	0x06
Start Date (Day)	2 Bytes	0x000A
Start Date (Month)	2 Bytes	0x0009
Start Date (Year)	2 Bytes	0x07E4

Error

The possible error responses for function code 0x03 are:

- Function Code (1 byte): 0x83 (0x80 + 0x03)
- Exception Codes (1 byte):
  - 0x01 – Function code not supported
  - 0x02 – Incorrect combination of start address and number of registers

Error response example:

Table 10.23. Read Holding Registers error response

Field	Length	Data
Transaction Identifier	2 Bytes	0x0004
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0003
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x83
Exception Codes	1 Byte	0x01 or 0x02

10.2.1.4. Write Single Register

This command is used to write a single register.

Request

The Write Single Register request specifies the register address to be written, and the register value.

Request example: Generate pulse on Relay 10, pulse width 3 seconds:

Table 10.24. Write Single Register request

Field	Length	Data
Transaction Identifier	2 Bytes	0x0005
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0006
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x06
Register Address	2 Bytes	0x0009
Register Value	2 Bytes	0x001E

Note: Pulse duration is given in milliseconds x 100.

Response

The normal response returns the function code, register address, and register value (echo of the query).

Response example: Generate pulse on Relay 10, pulse width 3 seconds:

Table 10.25. Write Single Register response

Field	Length	Data
Transaction Identifier	2 Bytes	0x0005
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0006
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x06
Register Address	2 Bytes	0x0009
Register Value	2 Bytes	0x001E

Error

The possible error responses for function code 0x06 are:

- Function Code (1 byte): 0x86 (0x80 + 0x06)
- Exception Codes (1 byte):
  - 0x01 – Function code not supported
  - 0x02 – Invalid register address
  - 0x03 – Invalid register value

Error response example:

Table 10.26. Write Single Register error response

Field	Length	Data
Transaction Identifier	2 Bytes	0x0005
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0003
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x86
Exception Codes	1 Byte	0x01 or 0x02 or 0x03

10.2.1.5. Write Multiple Registers

This command sets a block of holding registers to specific values.

Request

The Write Multiple Registers request specifies the starting register address, the number of registers, byte count, and the values to be written in ascending order. Values are packed as two bytes per register.

Note: If the sum of the start address and number of registers exceeds the size of the accessed block of registers, an error response will be returned.

Request example: Set Week Schedule Start Date to 10/09/2020:

Table 10.27. Write Multiple Registers request

Field	Length	Data
Transaction Identifier	2 Bytes	0x0006
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x000D
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x10
Starting Register Address	2 Bytes	0x6100
Quantity of Registers	2 Bytes	0x0003
Byte Count	1 Byte	0x06
Start Date (Day)	2 Bytes	0x000A
Start Date (Month)	2 Bytes	0x0009
Start Date (Year)	2 Bytes	0x07E4

Response

The normal response returns the function code, starting register address, and quantity of registers written.

Response example: Set Week Schedule Start Date to 10/09/2020:

Table 10.28. Write Multiple Registers reply

Field	Length	Data
Transaction Identifier	2 Bytes	0x0006
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0006
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x10
Starting Register Address	2 Bytes	0x6100
Quantity of Registers	2 Bytes	0x0003

Error

The possible error responses for function code 0x0F are:

- Function Code (1 byte): 0x90 (0x80 + 0x10)
- Exception Codes (1 byte):
  - 0x01 – Function code not supported
  - 0x02 – Invalid register address
  - 0x03 – Invalid register value

Error response example:

Table 10.29. Write Multiple Registers error response

Field	Length	Data
Transaction Identifier	2 Bytes	0x0003
Protocol Identifier	2 Bytes	0x0000
Length	2 Bytes	0x0003
Unit Identifier	1 Byte	0x00
Function Code	1 Byte	0x90
Exception Codes	1 Byte	0x01 or 0x02 or 0x03

10.3. MQTT (for smartDEN IP-16R-MQ only)

The **smartDEN IP-16R-MQ** supports MQTT protocol V3.1.1. The module runs a MQTT client that publishes the relays states, and subscribes to messages to switch the relays On/Off.

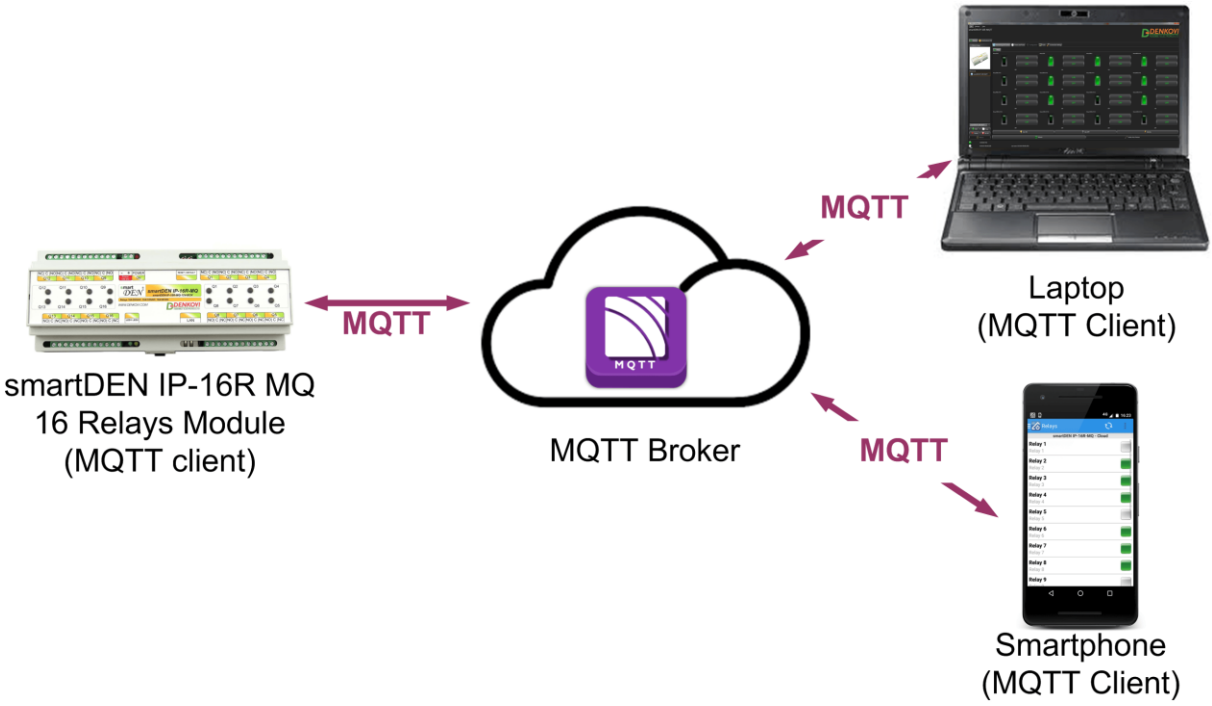

Figure 10.2. smartDEN IP-16R-MQ working as MQTT client

Two types of publish/subscribe topics modes are supported: plain (non-encrypted) and encrypted.

10.3.1. Plain (non-encrypted) mode

In this mode, **smartDEN IP-16R-MQ** uses non-encrypted topics to get relays states and switch relays On/Off.

Encrypt option in *MQTT Settings* page must be turned off in order to work in this mode.

Below are described all the available topics in order to communicate with smartDEN IP-16R-MQ via MQTT protocol. All topics begin with the following prefix:

SmartDEN_MQTT16R/<MAC identifier>/

Where, <MAC identifier> is the MAC address of the board written without the colons. For example:

SmartDEN_MQTT16R/E8EADA123456/Get/#
SmartDEN_MQTT16R/E8EADA123456/Set/All

10.3.1.1. Get topics

The **smartDEN IP-16R-MQ** publishes data up-on certain conditions. Another MQTT client (Laptop, Smartphone) is subscribed to these topics and receives the data.

Table 10.30. MQTT get topics

Topic	Application Message	Description
Get/#		all messages published by **smartDEN IP-16R-MQ**
Get/Status	Rebooted, Connected, Disconnected	The module status related to the MQTT broker connection
Get/Ri	On, Off	The relays (**i**=1...16) states. Can be published: • periodically; • in case of relay state change (see the Publish Mode option from MQTT settings page); • upon receiving command with value of 0 in topic **Set/GetStatus**
Get/All	Relays state in XML or JSON format.	**smartDEN IP-16R-MQ** publishes the relays states upon receiving a command under the **Set/GetStatus** topic: • 1 - XML format; • 2 - JSON format. See: *Appendix 2. Application reply formats*
Get/Auto-reboot	RebootsNumber: **i**, LastReboot: **date time**	A notification when a monitored device connected to one of the relays is rebooted. **i** is the reboots number, **date** is in format dd/mm/yyyy, **time** is in format hh:mm

10.3.1.2. Set topics

An MQTT Client (Laptop, Smartphone) publishes to these topics the request data. The **smartDEN IP-16R-MQ** subscribes to them and receives the request data and takes some actions (for example turn relays) and/or send reply to the get topic.

Table 10.31. MQTT set topics

Topic	Application Message	Description
Set/RSi	0,1,2	Turns relays (**i**=1...16) OFF or ON (0 - OFF, 1 - ON, 2 – TOGGLE)
Set/All	0 to 65535	Turns all the relays at once. The most significant bit of the value specifies the state of relay 16, and the least significant bit – the state of relay 1.
Set/RNi	1 to 65535	Generates a positive (ON) pulse on a single relay (**i**=1...16). The pulse

		duration is in ms x 100.
Set/RFi	1 to 65535	Generates a negative (OFF) pulse on a single relay (**i**=1...16). The pulse duration is in ms x 100.
Set/Date	Date in format **dd/mm/yyyy**	Changes the date of **smartDEN IP-16R-MQ**
Set/Time	Time in format **hh:mm**	Changes the time of **smartDEN IP-16R-MQ**
Set/RDi	Relay description string (max 7 chars)	Changes the relay description string
Set/GetStatus	0,1,2	• 0 - the relays states are published under the **Get/Ri** topic (**i**=1...16). • 1 - the relays states are published in XML format under the **Get/All** topic. • 2 - the relays states are published in JSON format under the **Get/All** topic. See: *Appendix 2. Application reply formats*

Notification MQTT messages are sent to **Get/Ri** or/and **Get/All** topics only for relays with checked "MQTT" option in the Relays Settings page.

10.3.2. Encrypted topics and DAE-aModules Android app

10.3.2.1. Commands

In this mode, there are used the following topics (encrypted topics):

- **Admin Read Topic** - MQTT client (Laptop, smartphone, DAE-aModules) publishes requests to this topic. The **smartDEN IP-16R-MQ** is subscribed to this topic and replies back to Admin Write Topic;
- **Admin Write Topic** - the **smartDEN IP-16R-MQ** publishes replies to this topic. Another MQTT client (Laptop, Smartphone) is subscribed to this topic and receives the data.
- **Notification Topic** - for notifications sent by the smartDEN IP-16R-MQ module upon event.

The encrypted topics are designed mainly for communication with the Android DAE-aModules application or customized MQTT libraries.

The topics generating process is available upon request.

The below commands in Table 10.32 are published by the MQTT client (Laptop, smartphone, DAE-aModules) to the **Admin Read Topic** and the **smartDEN IP-16R-MQ** publishes a reply in XML format to the **Admin Write Topic** (see *Appendix 2. Application reply formats*)

Table 10.32. Encrypted topics and reply in XML

Topic	Application Message	Description
Admin Read Topic	MQTT_COMMAND?GETSTATUS;	Get the relays states
	MQTT_COMMAND?RSX**i**=**x**;	Turns relay (**i**=1…16) ON/OFF. **x**: 0 - OFF, 1 - ON, 2 – TOGGLE
	MQTT_COMMAND?ALLX=**x**;	Turns all the relays at once. The most significant bit of the value specifies the state of relay 16, and the least significant bit – the state of relay 1. **x** represents the relays states - from 0 to 65535
	MQTT_COMMAND?RNX**i**=**x**;	Generates a positive (ON) pulse on a single relay (**i**=1...16). The pulse duration **x** is in ms x 100.
	MQTT_COMMAND?RFX**i**=**x**;	Generates a negative (OFF) pulse on a single relay (**i**=1...16). The pulse duration **x** is in ms x 100.
	MQTT_COMMAND?DONAME**i**=**x**;	Sets the name of a relay (**i**=1...16). The length of **x** is 1 to 7 symbols.
	MQTT_COMMAND?DATE=**x**;	Sets the date, **x** is format dd/mm/yyyy
	MQTT_COMMAND?TIME=**x**;	Sets the time, **x** is format hh:mm

The below commands in Table 10.33 are published by the MQTT client (Laptop, smartphone, DAE-aModules) to the **Admin Read Topic** and the **smartDEN IP-16R-MQ** publishes a reply in JSON format to the **Admin Write Topic** (see *Appendix 2. Application reply formats*)

Table 10.33. Encrypted topics and reply in JSON

Topic	Application Message	Description
Admin Read Topic	MQTT_COMMAND?GETSTATUS2;	Get the relays states
	MQTT_COMMAND?RSJ**i**=**x**;	Turns relay (**i**=1…16) ON/OFF. **x**: 0 - OFF, 1 - ON, 2 – TOGGLE
	MQTT_COMMAND?ALLJ=**x**;	Turns all the relays at once. The most significant bit of the value specifies the state of relay 16, and the least significant bit – the state of relay 1. **x** represents the

		relays states - from 0 to 65535
	MQTT_COMMAND?RNJ**i**=**x**;	Generates a positive (ON) pulse on a single relay (**i**=1...16). The pulse duration **x** is in ms x 100.
	MQTT_COMMAND?RFJ**i**=**x**;	Generates a negative (OFF) pulse on a single relay (**i**=1...16). The pulse duration **x** is in ms x 100.

10.3.2.2. Android app DAE-aModules

In order to control/monitor the **smartDEN IP-16R-MQ** by Android device the module should be added to the list of boards, controlled by the DAE-aModules application:

Figure 10.3. Add a new **smartDEN IP-16R-MQ** module

Next, the added module must be configured in the **Device Settings** screen of DAE-aModules:

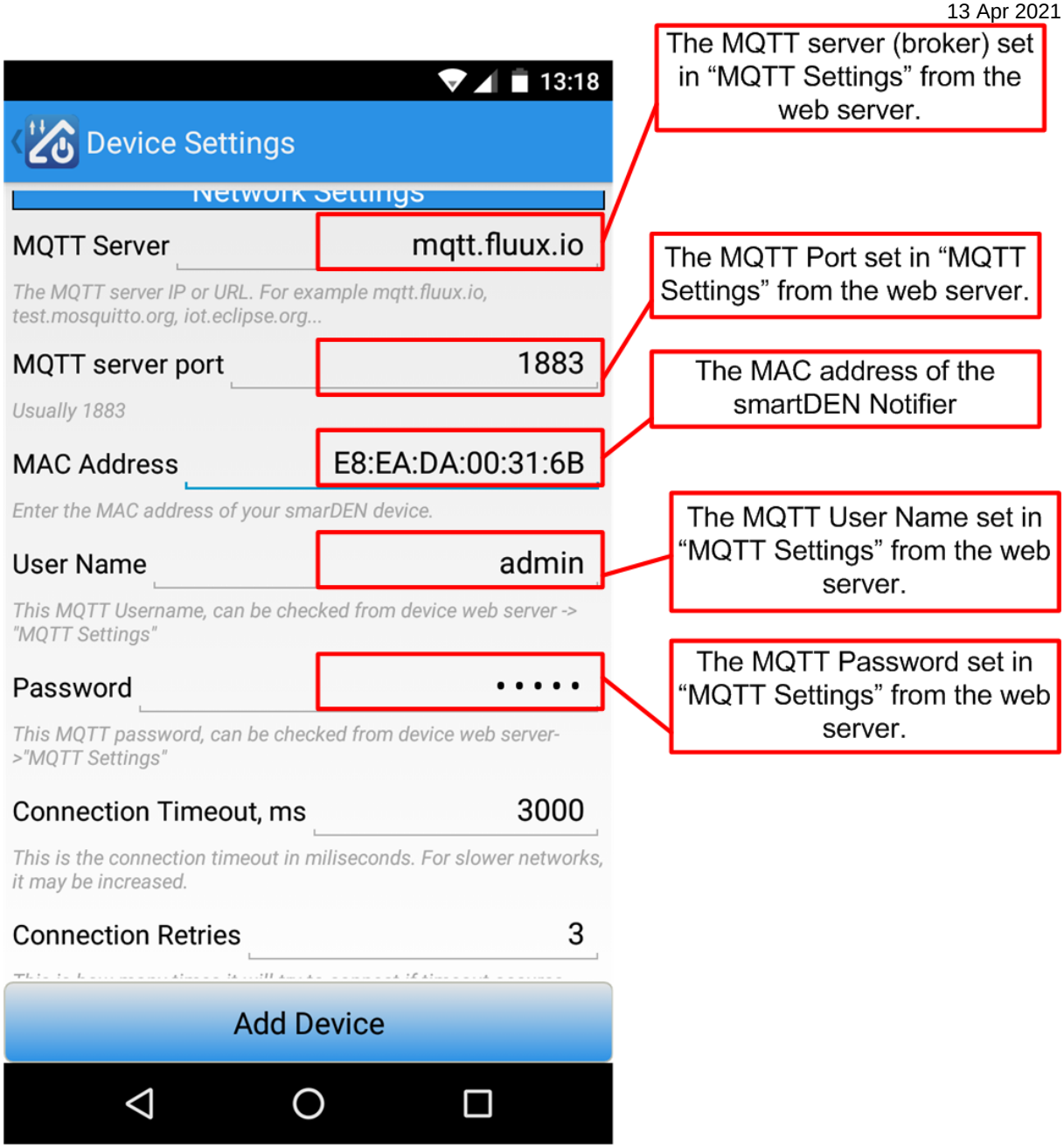

Figure 10.4. MQTT settings

Last, the notifications from the DAE-aModules must be allowed in the Android *"Notifications management"* screen:

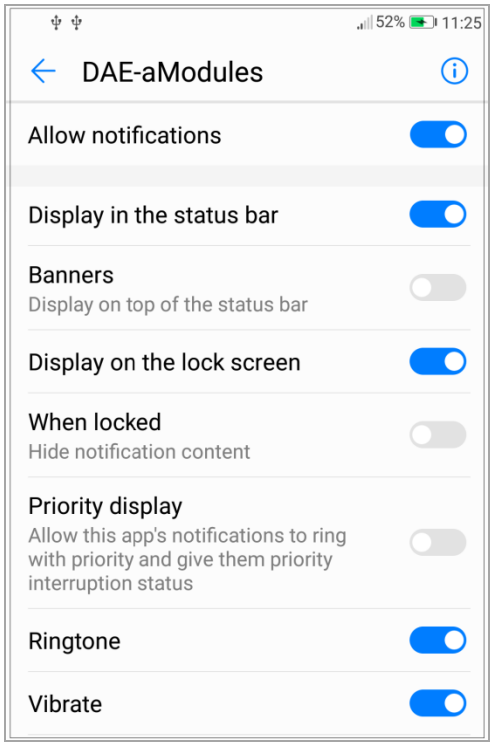

Figure 10.5. Android *"Notifications management"* settings for DAE-aModules

When configured, the inputs to be monitored can be selected from the navigation menu:

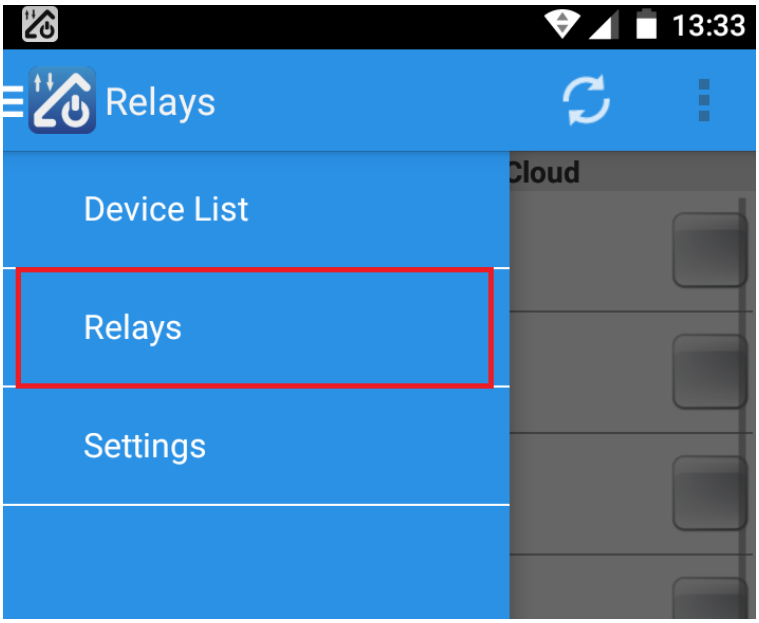

Figure 10.6. Navigation menu

Example monitoring screens for digital, analog and temperature inputs are shown below:

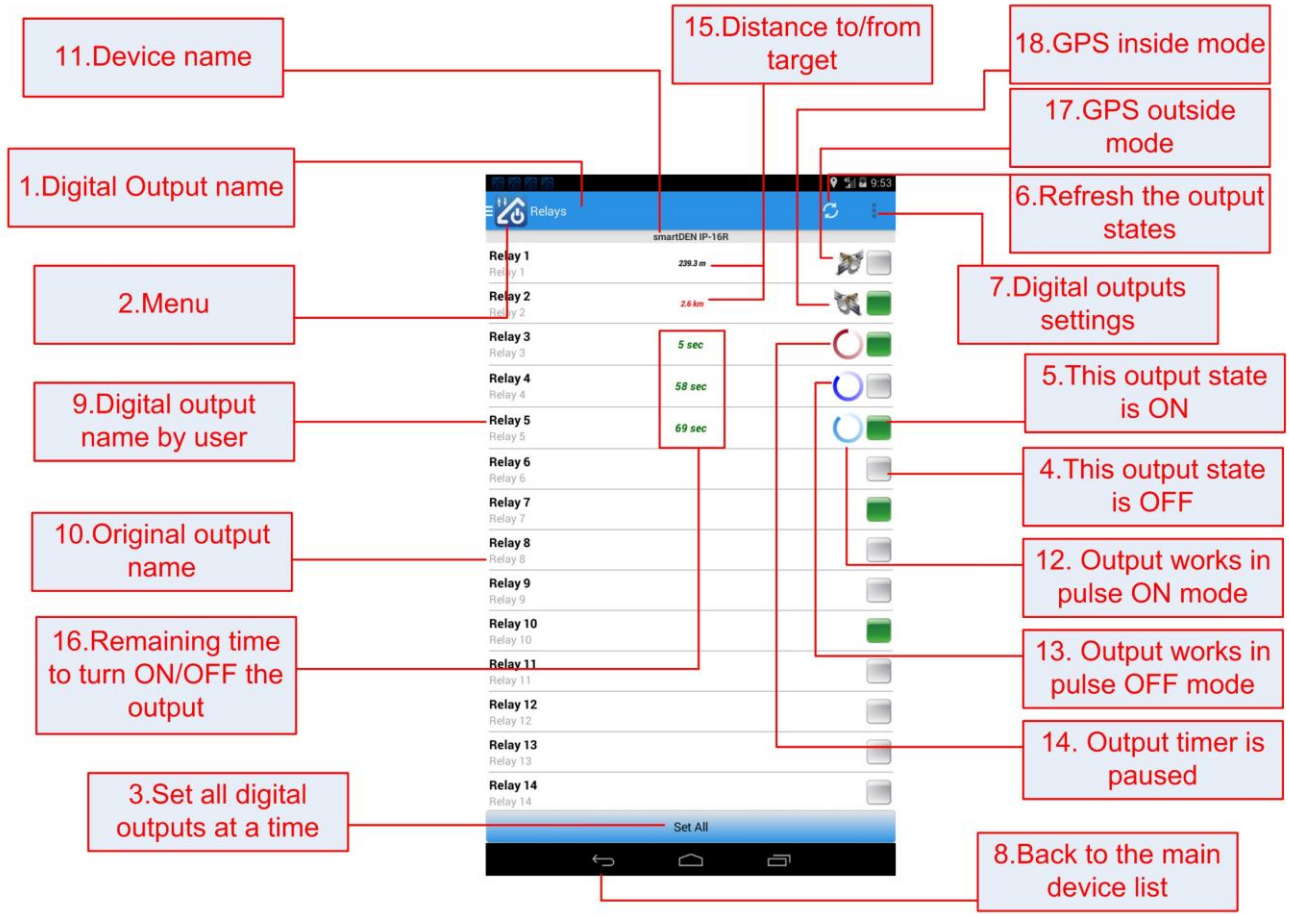

Figure 10.7. Relays in DAE-aModules

Examples of status and relays state notifications, published by **smartDEN IP-16R-MQ** are shown below.

Figure 10.8. Status notifications

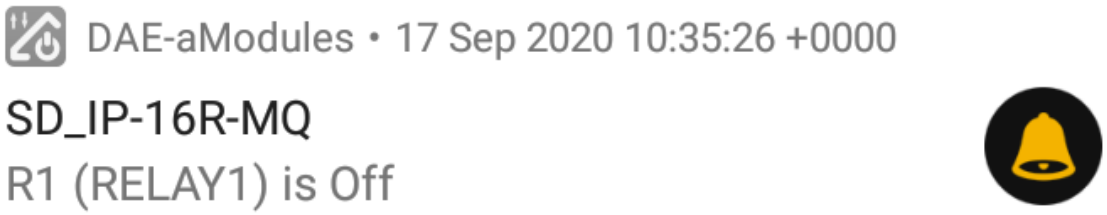

Figure 10.9. Inputs state notifications

11. Security considerations

The **smartDEN IP-16R-XX** runs a special firmware and do not have a general-purpose operating system. There are no extraneous IP services found on general-purpose operating systems (e.g. fingerd, tcp_wrapper, etc.) that can possibly be exploited by an unauthorized agent. In particular, the **smartDEN IP-16R-XX** does not run protocols such as Telnet and FTP which may have the potential for security breech. The only exception from this is the integration protocol, that can be disabled.

Web-browser access
A challenge-response authentication is used in login process. When the password is entered, it is transmitted across the network in encrypted form, so eavesdropping on the data transmission will not reveal the password. Subsequent transmissions of the password to "login" onto the device are encrypted and "safe". The only case when the password is transmitted across the network "in the open", is when it is being changed and submitted in **General Settings** form. Therefore, you must set passwords in the secure environment where you can make sure that no one is "eavesdropping".

SNMP communication (for smartDEN IP-16R only)
SNMPv1 does not implement encryption. Authentication of clients is performed only by a "community string", which is transmitted in clear text. SNMP communication should be used in trusted networks and disabled if not used.

Modbus-TCP communication (for smartDEN IP-16R-MT only)
Modbus-TCP does not implement encryption. Modbus-TCP communication should be used in trusted networks and disabled if not used.

MQTT communication (for smartDEN IP-16R-MQ only)
Within the current module implementation the MQTT does not implement any encryption. This communication should be used in trusted networks and disabled if not used.

XML/JSON operation
A challenge-response authentication can be used in login process. The password can be transmitted by custom application across the network in encrypted form.

Web and XML/JSON access can be restricted by IP Address (range of IP Addresses) or by MAC Address.

12. Appendix 1. Mechanical dimensions

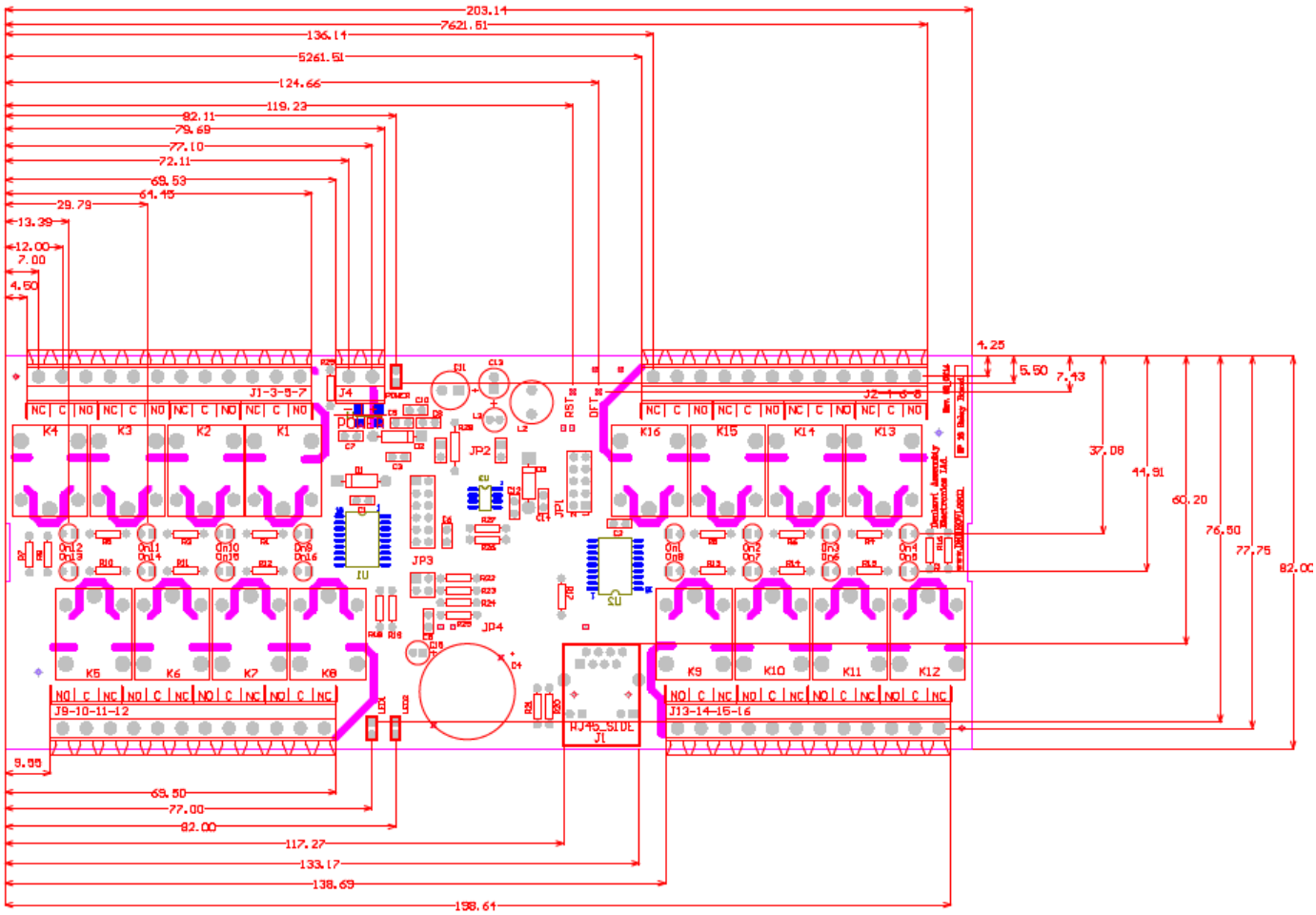

Figure 12.1. PCB dimensions

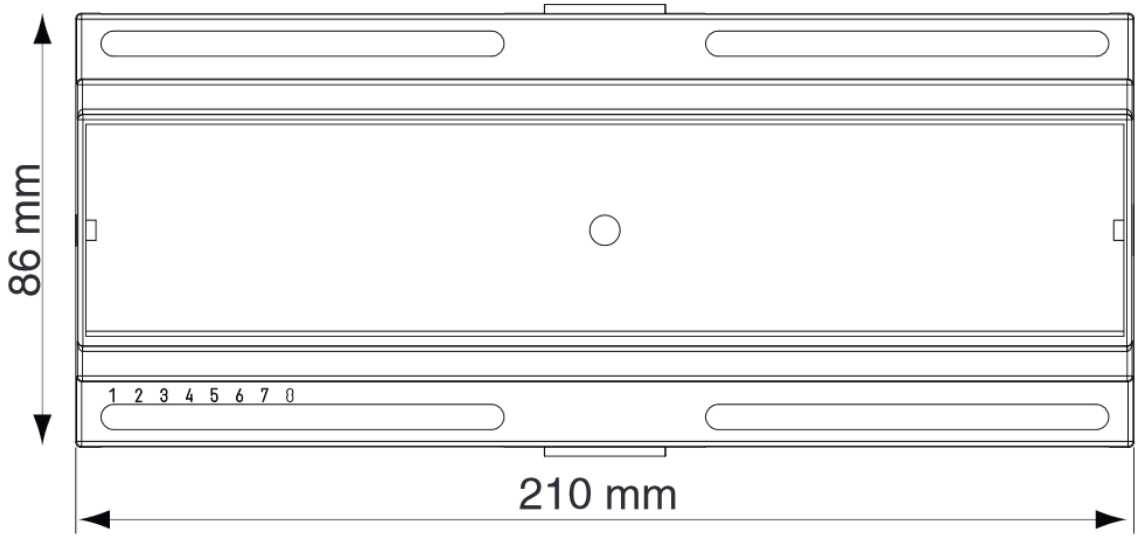

Figure 12.2. Box dimensions

13. Appendix 2. Application reply formats

13.1. XML reply


```
−<CurrentState>
  −<Relay1>
      <Name>RELAY1</Name>
      <State>0</State>
    </Relay1>
  +<Relay2></Relay2>
  +<Relay3></Relay3>
  +<Relay4></Relay4>
  +<Relay5></Relay5>
  +<Relay6></Relay6>
  +<Relay7></Relay7>
  +<Relay8></Relay8>
  +<Relay9></Relay9>
  +<Relay10></Relay10>
  +<Relay11></Relay11>
  +<Relay12></Relay12>
  +<Relay13></Relay13>
  +<Relay14></Relay14>
  +<Relay15></Relay15>
  +<Relay16></Relay16>
  −<Auto-reboot>
      <RebootsNumber>0</RebootsNumber>
      <LastReboot>-</LastReboot>
    </Auto-reboot>
  −<Device>
      <Name>SD_IP-16R-MQ</Name>
      <MAC>E8:EA:DA:00:31:6B</MAC>
      <Date>11/09/2020</Date>
      <Time>09:07</Time>
    </Device>
</CurrentState>
```

13.2. JSON reply


```
{
  "CurrentState": {
    "Output": [
      {"Name": "RELAY1", "Value": "0"},
      {"Name": "RELAY2", "Value": "0"},
      {"Name": "RELAY3", "Value": "0"},
      {"Name": "RELAY4", "Value": "0"},
      {"Name": "RELAY5", "Value": "0"},
      {"Name": "RELAY6", "Value": "0"},
      {"Name": "RELAY7", "Value": "0"},
      {"Name": "RELAY8", "Value": "0"},
      {"Name": "RELAY9", "Value": "0"},
      {"Name": "RELAY10", "Value": "0"},
      {"Name": "RELAY11", "Value": "0"},
      {"Name": "RELAY12", "Value": "0"},
      {"Name": "RELAY13", "Value": "0"},
      {"Name": "RELAY14", "Value": "0"},
      {"Name": "RELAY15", "Value": "0"},
      {"Name": "RELAY16", "Value": "0"}
    ],
    "Auto-reboot": {
      "RebootsNumber": "0",
      "LastReboot": "-"
    },
    "Device": {
      "Name": "SD_IP-16R-MQ",
      "MAC": "E8:EA:DA:00:31:6B",
      "Date": "11/09/2020",
      "Time": "09:08"
    }
  }
}
```