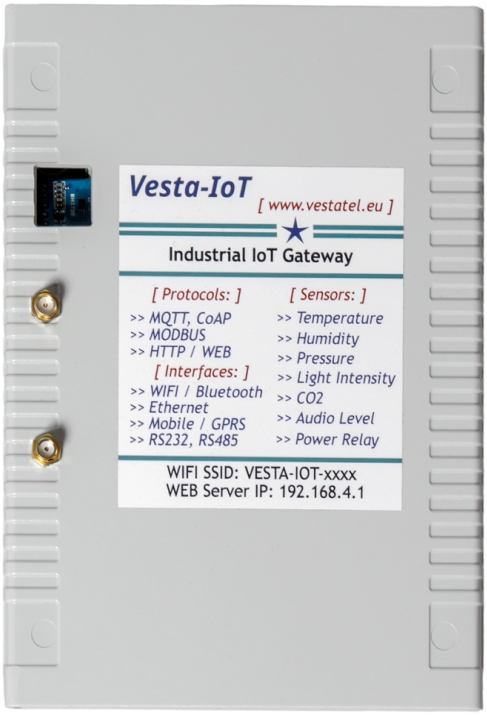


Vesta-IoT is an Industrial Internet of Things (IoT) Gateway. It continuously monitors multiple on-board sensors - Temperature, Humidity, Pressure, Light Intensity, CO2 concentration, Air Resistance, Audio / Noise Level.

On each sensor poll cycle Vesta-IoT re-calculates Air Quality Index (IAQ) given the relative humidity and air resistance and exposes IAQ as another sensor value.

Vesta-IoT can periodically send the sensor readings using MQTT and also makes the data available using MODBUS, DNP3, IEC-60870-5-104, CoAP and HTTP / SOAP protocols.

Additionally Vesta-IoT can be configured to log all sensor readings into non-volatile memory (flash) thus acting as a data logger. The data can be downloaded as a CSV file or can be viewed using the web interface.

The device also features an on-board AC power relay and various communication interfaces: WIFI, wired Ethernet, GSM/LTE/GPRS, RS232, RS485.

Key Features:

- Multiple Environmental Sensors: Temperature, Humidity, Pressure, Light Intensity, CO2 concentration, Air Resistance, Audio / Noise Level
- Integrated AC Power Relay: Allows switching on or off AC (up to 250 AC) or DC power loads with multiple ways to control the relay: Via Web UI, Modbus, DNP3, IEC 60870-5-104, CoAP, MQTT, HTTP / SOAP via internal time scheduler or based on internal sensors crossing defined threshold values
- Real-Time Clock: With an internal battery backup, ensures accurate time-stamping and event scheduling, with NTP time synchronization
- Multiple Communication Interfaces: WiFi with external antenna in AP or Station Mode, Wired Ethernet, embedded LTE/GSM/GPRS module, RS232 / RS485 interfaces
- Multiple Communication Protocols: DHCP, TCP/IP, UDP/IP, WEB SERVER, MQTT CLIENT, CoAP Server, Modbus TCP Server, DNP3 TCP server, IEC 60870-5-104 Server, HTTP/ SOAP Server, RS232 Server, NTP
- Built-In Web Server: For easy configuration and status monitoring, HTTP / SOAP API Server
- TCP Diagnostic / CLI Console: TCP console server can be used for essencial maintenance and status checking

Technical Specifications

TEMPERATURE SENSOR	
Maximum Range	-40 C .. +85 C
Accuracy	From +/- 0.5C to +/-1.0C
Resolution	0.01 C
RELATIVE HUMIDITY SENSOR	
Accuracy Tolerance	+- 3 %RH
Resolution	0.008 %RH
PRESSURE SENSOR	
Pressure Range	300 .. 1100 hPa
Accuracy	+- 0.6 hPa
Resolution	0.18 hPa
GAS RESISTANCE SENSOR	
Resolution	0.08%
CO2 SENSOR	
Operating Range	0 ppm to 32000 ppm
Accuracy	+- 30 ppm (+3%)
LIGHT INTENSITY SENSOR	
Operating Range	1 .. 65525 Lux
Accuracy	1.2 Lux
AUDIO POWER LEVEL SENSOR	
Operating Range	0 to 100% Audio Volume Power
POWER RELAY	
Operating Range	Max AC Voltage: 250VA, MAx current: 5A (resistive load), 2.5A (inductive load)
WIFI	
Standards & Protocols	IEEE 802.11 b/g/n, 2.4 GHz, AP/STA modes
Communication Range	30 – 50 meters indoor / up to 200m meters outdoor in direct line of sight
LTE/GPRS	
LTE Cat-1	Frequency band: LTE-FDD: B1, B3, B5, B7, B8, B20
2G	GSM/GPRS/EDGE: 900/1800 MHz
WIRED ETHERNET (LAN)	
Standards & Protocols	10BaseT/100BaseTX, Auto Negotiation (Full/half duplex
RS232 / RS485	
Features	Transient Voltage Suppressor, DB9 socket for RS232, RX, TX GND terminals for RS485, Configurable RS485 termination resistor
SOFTWARE	
Supported Protocols / Features	DHCP, TCP/IP, UDP/IP, WEB / HTTP / SOAP Server, MQTT CLIENT, CoAP SERVER, MODBUS TCP SERVER, DNP3 Server, IEC 60870-5-104 Server, RS232 SERVER, NTP, Data Logger (to non-volatile memory / flash)

Hardware Models Cross Reference

Feature / Model	Model 100	Model 200	Model 300	Model 400M	Model 400S	Model 500
COMMUNICATIION INTERFACES						
WIFI	Yes	Yes	Yes	Yes	Yes	Yes
Ethernet	--	Yes	Yes	--	--	Yes
LTE / Mobile	--	--	--	Yes	--	Yes
Serial RS232 + RS485	--	--	--	--	Yes	Yes
ON – BOARD SENSORS						
Temperature Sensor	Yes	Yes	Yes	Yes	Yes	Yes
Humidity Sensor	Yes	Yes	Yes	Yes	Yes	Yes
Pressure Sensor	Yes	Yes	Yes	Yes	Yes	Yes
Air Resistance Sensor	Yes	Yes	Yes	Yes	Yes	Yes
IAQ – Index of Air Quality	Yes	Yes	Yes	Yes	Yes	Yes
CO2 Sensor	--	--	Yes	Yes	Yes	Yes
Audio Level Sensor	--	--	Yes	Yes	Yes	Yes
Light Intensity Sensor	--	--	Yes	Yes	Yes	Yes
OTHER HARDWARE						
Digital Output (Power Relay)	--	--	Yes	Yes	Yes	Yes
SOFTWARE FEATURES						
Web Inteface for Configuration, Control and Monitoring	Yes	Yes	Yes	Yes	Yes	Yes
HTTP / SOAP API Server	Yes	Yes	Yes	Yes	Yes	Yes
NTP synchronization of real time clock	Yes	Yes	Yes	Yes	Yes	Yes
MQTT Client	Yes	Yes	Yes	Yes	Yes	Yes
MODBUS TCP Server	Yes	Yes	Yes	Yes	Yes	Yes
MODBUS RTU Client over RS232 / RS485	--	--	--	--	Yes	Yes
CoAP Server	Yes	Yes	Yes	Yes	Yes	Yes
DNP3 TCP Server	Yes	Yes	Yes	Yes	Yes	Yes
IEC 60870-5-104 Server	Yes	Yes	Yes	Yes	Yes	Yes
UDP Sender	Yes	Yes	Yes	Yes	Yes	Yes
TCP Sender (TCP client and TCP Server)	Yes	Yes	Yes	Yes	Yes	Yes
TCP Console Server	Yes	Yes	Yes	Yes	Yes	Yes
TCP Serial Server over RS232	--	--	--	--	Yes	Yes
Output Power Realy Control (Time Scheduled, sensor threshold based, Manual from the web UI)	--	--	Yes	Yes	Yes	Yes
Data Logger (to non-volatile memory / flash	Yes	Yes	Yes	Yes	Yes	Yes

Table of Contents

1 Vesta-IoT Operation

Sensor Readings:

Vesta-IoT takes the sensor readings at regular 5 second intervals. These sensors include light intensity, temperature, humidity, pressure, air quality, CO2 concentration, Air resistance, audio noise level, calculated IAQ (Index of Air Quality)

Data Handling:

After each reading, if configured to do so, the data can be pushed over MQTT to an MQTT broker.

Additionally the sensor readings can be sent over TCP or UDP (TCP Sender and UDP Sender features).

The data is also made continuously available to be presented to a connecting Modbus TCP client, a DNP3 Client, an IEC 60870-5-104 a CoAP client over UDP or to an HTTP / Web client.

The device also runs a SOAP API server and an external SOAP client can poll the sensor data by sending properly formatted SOAP queries.

Additionally Vesta IOT can be configured to act as a data logger where all sensor readings are pediodically saved into a non-volatile memory / flash. The data can be downloaded as a CSV file or viewed via web interface.

Communication Interfaces:

The data can be transmitted / made available over WIFI in AP or Station mode, over wired Ethernet (LAN) or over LTE/GSM/GPRS.

Serial Interfaces:

The RS232 interface can be used as part of Serial Server feature which provides access to any device connected to the RS232 serial interface over the TCP. Additionally a Modus RTU client / master can be configured to run on over RS232 or RS485 serial interface.

On-board Power Relay:

The on-board power relay can be activated in persistent or Pulse On mode with configurable ON time directly from the Web UI or using configured scheduled time intervals. Additionally the relay can be configured to activate and de-activate based on any sensor value crossing the configurable high and low thresholds. The relay can also be controlled over Modbus, CoAP, MQTT, DNP3, IEC 60870-5-104 or HTTP / SOAP protocols

Status LED:

Staus LED blinks green at every sensor polling interval.

Buttons:

The device features two push buttons:

- Hardware Reset (Reboot)
- Load Factory Setings

2 Configuring Vesta-IoT

Default Web Configuration Access:

By default the built-in web server runs on both the wired ethernet (LAN) and WiFi interfaces.

The WiFi interface in AP mode, advertizes SSID in the form: VESTA-IOT-N (N differs per device). Default WIFI password: 123456789

A connecting WiFi client gets IP address assigned over DHCP. The built-in web server runs on IP address 192.168.4.1

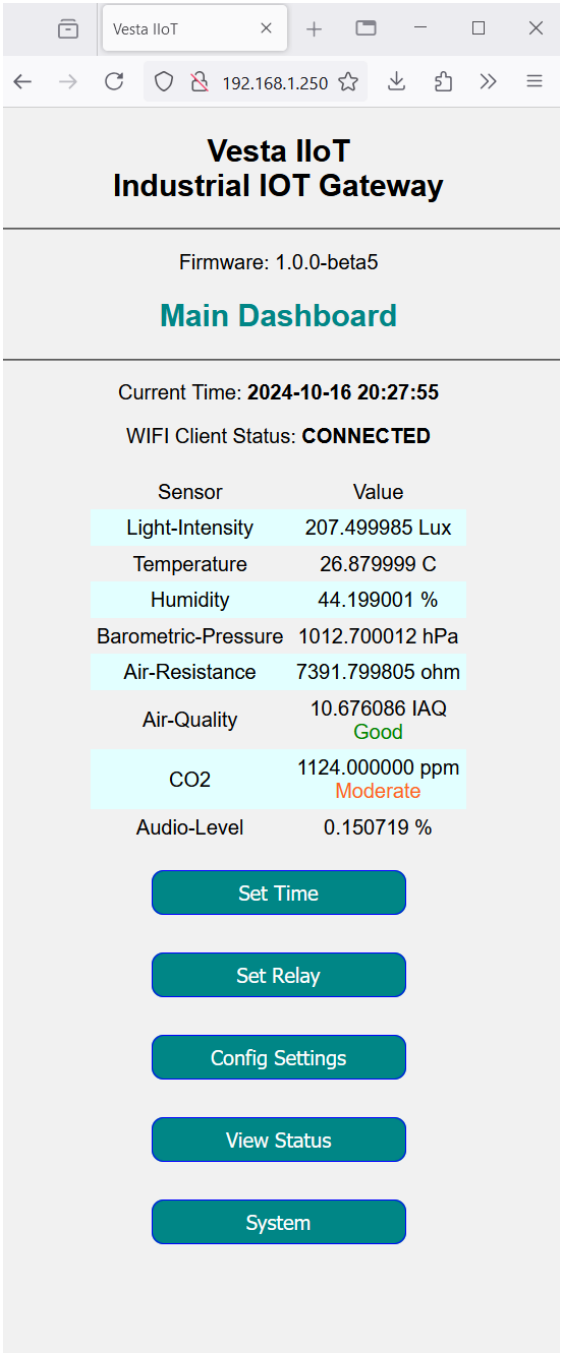
To access the web server, type in the browser: http://192.1.4.1/

Wired Ethernet has statically configured IP address: 192.168.1.250

To access web server, type in the browser: http://192.168.1.250/

You will be presented with the Main Dashboard screen where you can

- View current time
- View current sensor readings
- Go to Set Time page
- Go to Relay Control page
- Go to Configure Settings page
- Go to View Status
- Or go to System page



3 Setting Time

Vesta-IoT features on-board real time clock (RTC) which is battery backed – continues operating even when device is powered down.

The device is shipped with the internal clock set to CEST (Central Eastern European Time)

To set the time click “Set Time” button and enter the necessary values for year, month, day, hour, minute and second and click “Set TIME” button.

4 NTP Time Synchronization

To configure periodic time synchronization with internet Time server set the necessary options on the same page and click “Save / Apply”

NTP Sync Enabled field enables or disables NTP synchronization

NTP Server IP: sets the IP address of time server

Sync Interval (Secs): sets the period of syncronization in seconds.

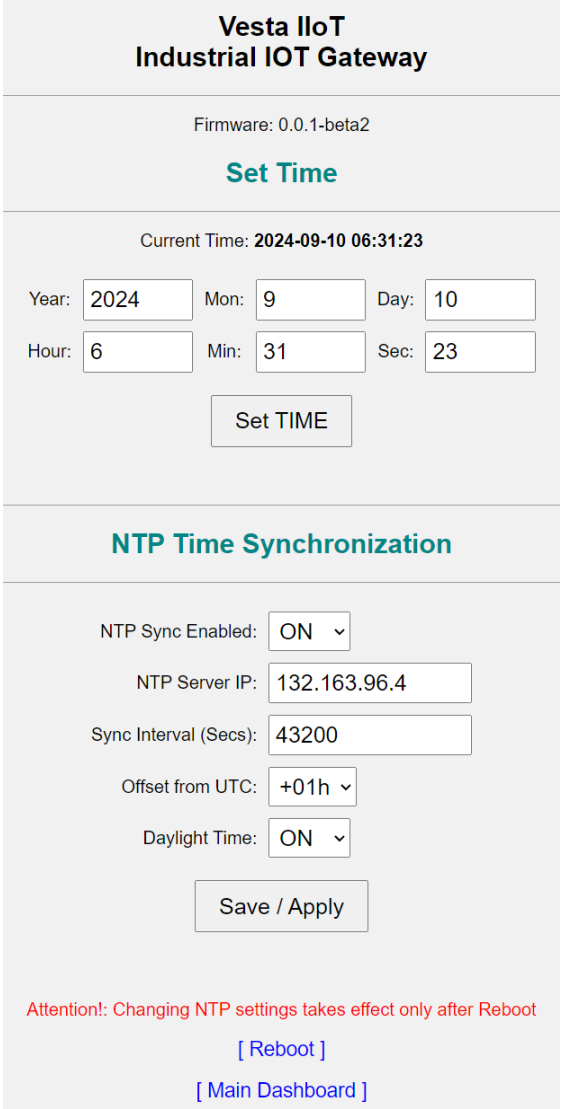
Offset from UTC: Set this to the timezone in which your time is set (see “Setting Time” above)

Daylight Time: When in summer savings time, set this to ON.

NOTE: NTP operates with global time in UTC format. Setting offset from UTC and Daylight time fields correcly are necessary to calculate adjustments to on-board real time clock.

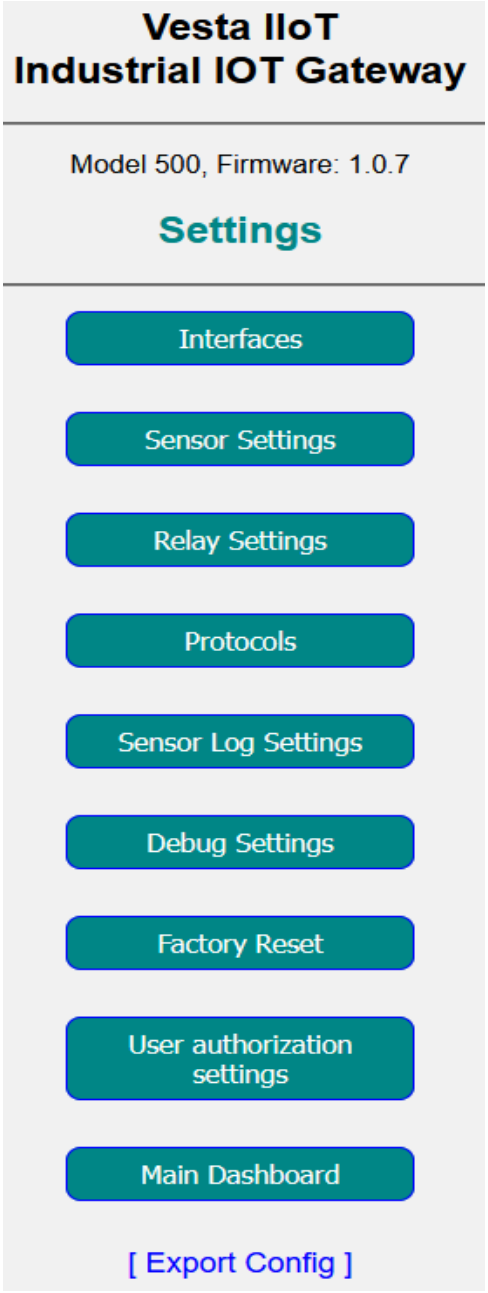
When changing any NTP settings, the changes take effect on next reboot. To proceed with reboot, click [Reboot]

To return to main screen, click [Main Dashboard]



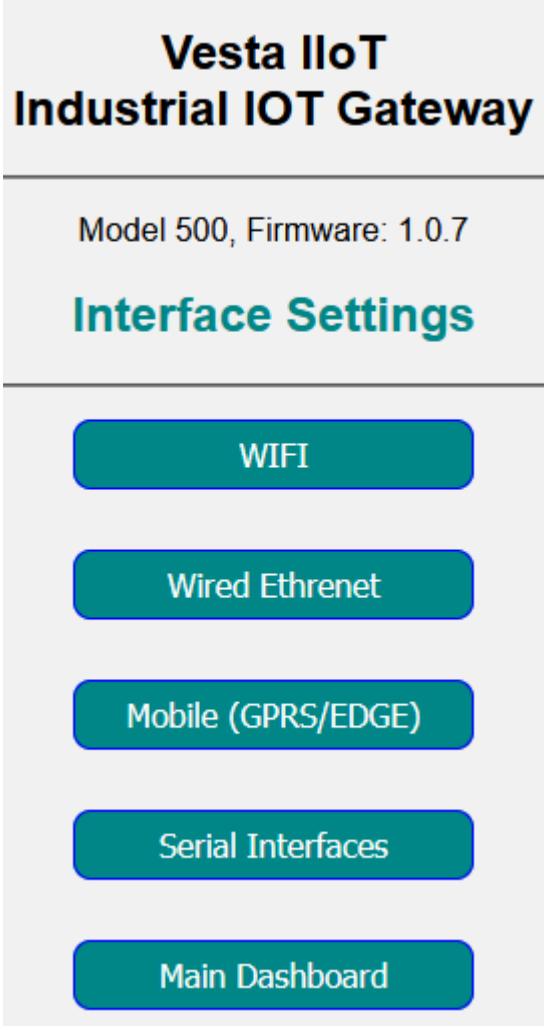
5 Configuration Menu

To configure various aspects of the device operation click "Config Settings" button. You will be presented with the following screen:



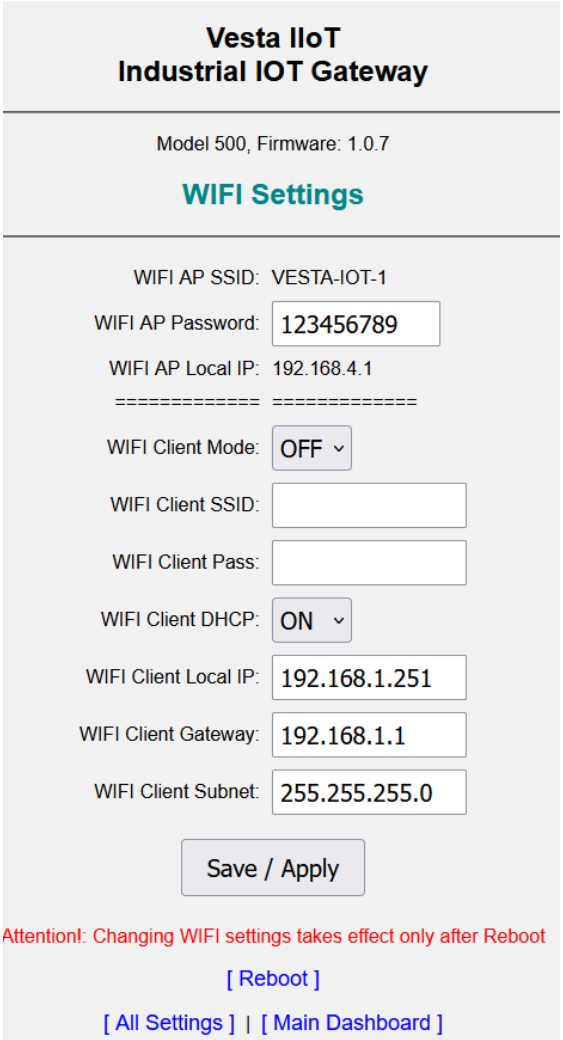
6 Configuration of Interfaces

By clicking the relevant button you can access the required part of configuration:



7 WiFi Configuration

To Configure WiFi, click "WIFI" button in the Interfaces menu.



WiFi AP Password configures the password for the WIFI Access Point mode

WIFI Client SSID, Wifi Client Pass and WIFI Client Local IP, gateway and subnet configure corresponding parameter for the WIFI in Station (Client) mode.

WIFI AP and Station mode co-exist simultaneously on the device.

WIFI Client DHCP: enables or disables using built in DHCP client to get an IP address from the access point in WIFI client mode

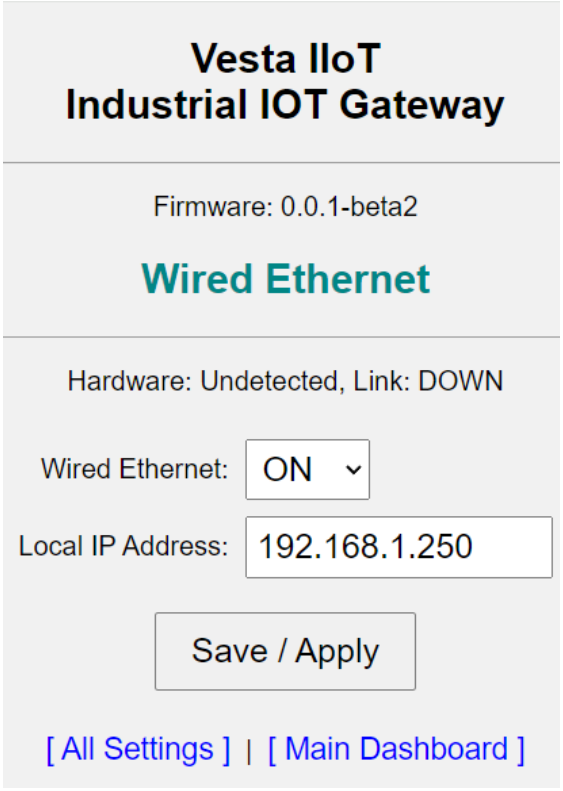
Local IP, Gateway and Subnet are used to configure local IP settings for the WIFI in client mode when DHCP is OFF.

When finished configuration, click "Save / Apply"

Please note that the WIFI settings changes take effect on next reboot. To reboot to to Main Dashboard →System →Reboot

8 Wired Ethernet Configuration

To Configure Wired Ethernet, click "Wired Ethernet" button in the Interfaces menu.



Wired Ethernet <ON | OFF> field enables or disables the wired ethernet

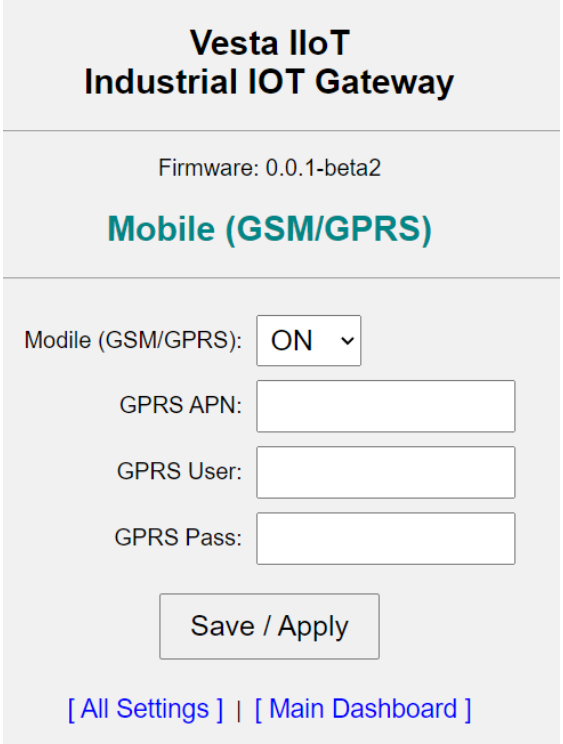
Local IP Address sets the static IPv4 address assigned to Wired Etherent.

When finished configuration, click “Save / Apply”

Please note that the Wired Ethernet settings changes take effect on next reboot. To reboot to to Main Dashboard →System →Reboot

9 Mobile (GSM/GPRS) Configuration

To Configure Mobile interface, click “Mobile GSM/GPRS” button in the Interfaces menu.

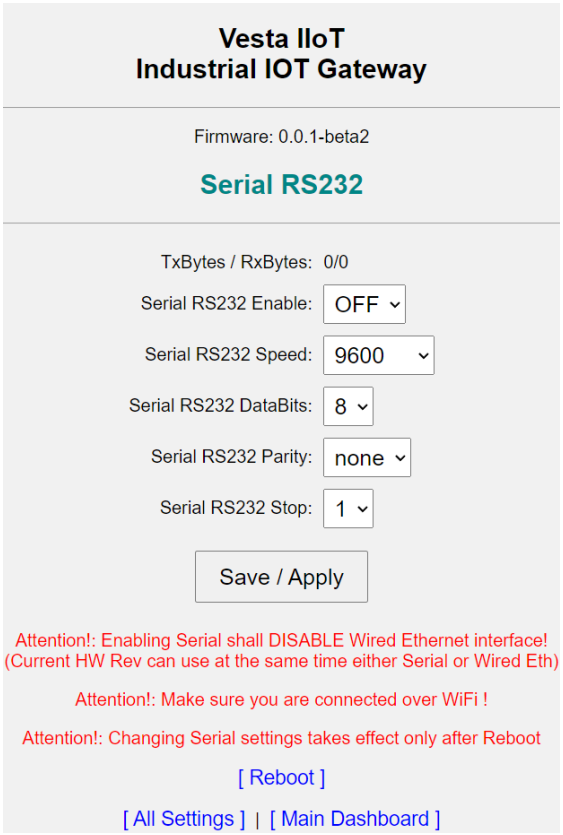


Here you can enable or disable Mobile interface and et the GPRS parameters (APN, username and password.

Please note that the Mobile settings changes take effect on next reboot. To reboot to to Main Dashboard →System →Reboot

10 Serial RS232 Configuration

To configure serial RS232 interface, click “Serial RS232” button in the Interfaces menu.

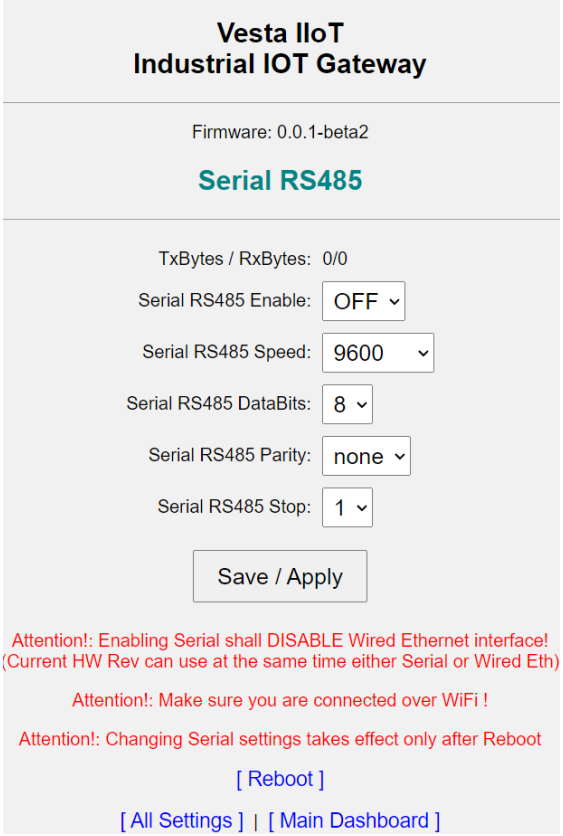


Here you can enable or disable Serial RS232 interface and set serial parameters: baud rate, number of data bits, number of stop bits, parity type.

Please note that the Serial settings changes take effect on next reboot. To reboot to to Main Dashboard →System →Reboot or click “Reboot” link at the bottom of the page

11 Serial RS485 Configuration

To configure serial RS485 interface, click “Serial RS485” button in the Interfaces menu.

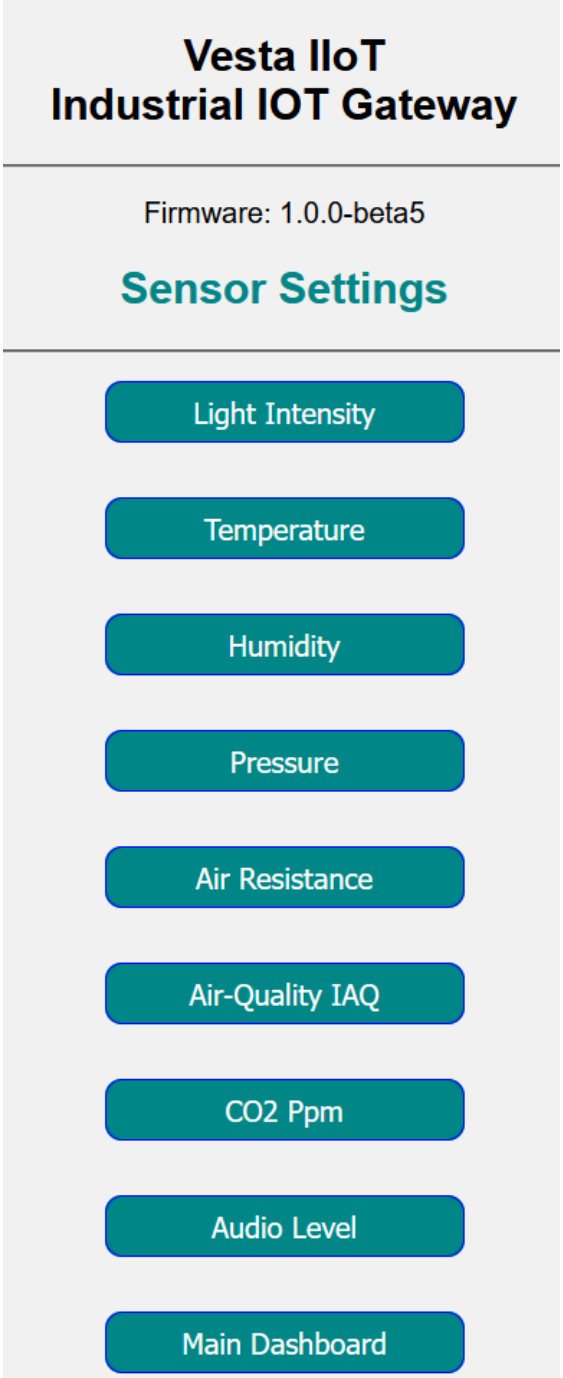


Here you can enable or disable Serial RS485 interface and set serial parameters: baud rate, number of data bits, number of stop bits, parity type.

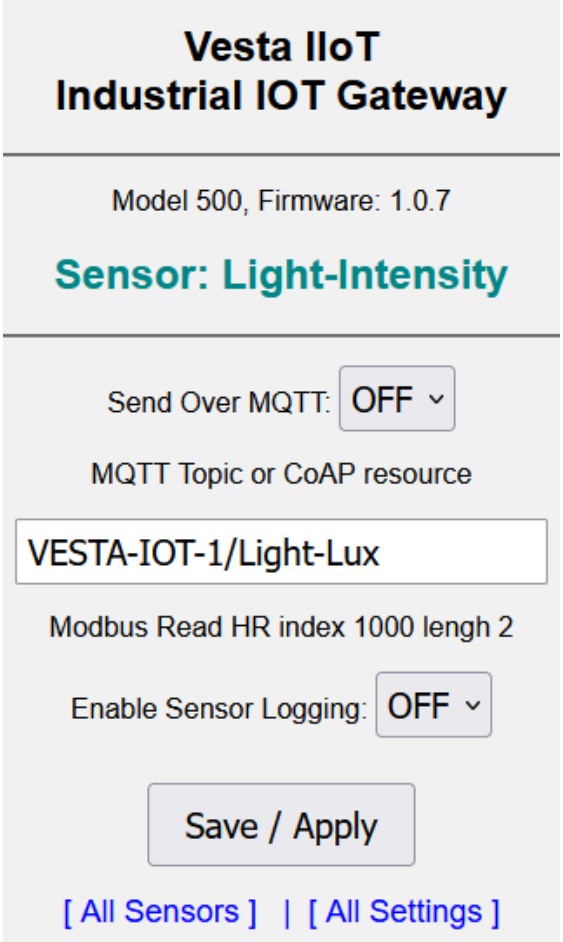
Please note that the Serial settings changes take effect on next reboot. To reboot to to Main Dashboard →System →Reboot or click “Reboot” link at the bottom of the page

12 Configuring Sensors

To configure on-board sensors, click “Sensor Settings” from “Settings” sub-menu



All sensors will have a similar configuration layout as shown below in the example of Light Intensity sensor (Click on “Light Intensity” in the sensors menu):



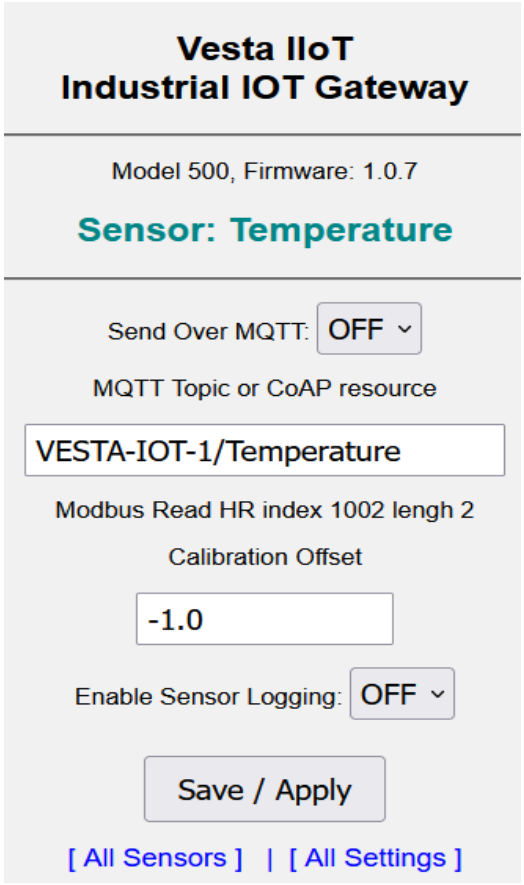
For each sensor it is possible to enable sending of sensor readings over MQTT (ON/OFF). The indivudual MQTT topic is specified in “MQTT Topic or CoAP resource”. The same field is used to configure the CoAP resource name when CoAP server is enabled

For the MODBUS server the type of register (HR – Holding register) and the index is pre-assigned for each sensor and it can be viewed on this page.

To include the sensor readings into Data Logger's sensor log (stored in non-volatile memory) set Enable Sensor Loggin: ON.

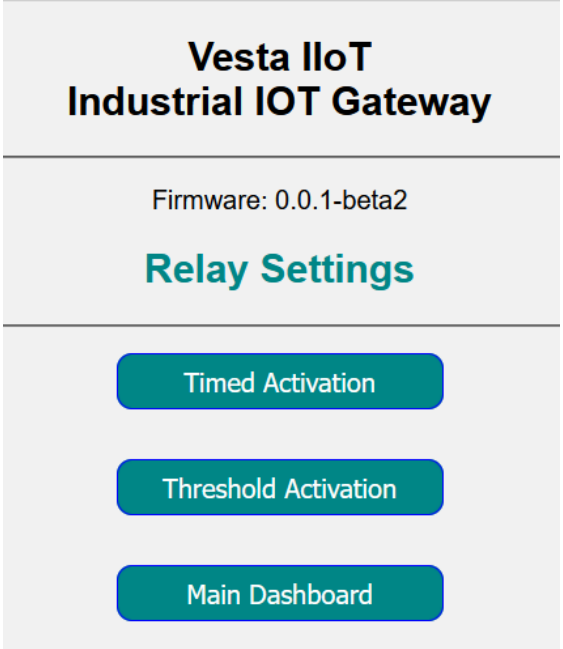
When finished with configuration, click “Save / Apply”

The Temperature sensor has an additional field – calibration offset – this is a signed integer field which values is added to the result of reading the sensor. For example if the reading is 27C and the offset is -1.0. the result reported shall be 26C



13 Relay Settings

To configure various aspects of power relay operation click “Relay Settings” from “Settings” sub menu.



To configure time scheduled relay activation / deactivation, click “Timed Activation” menu item.

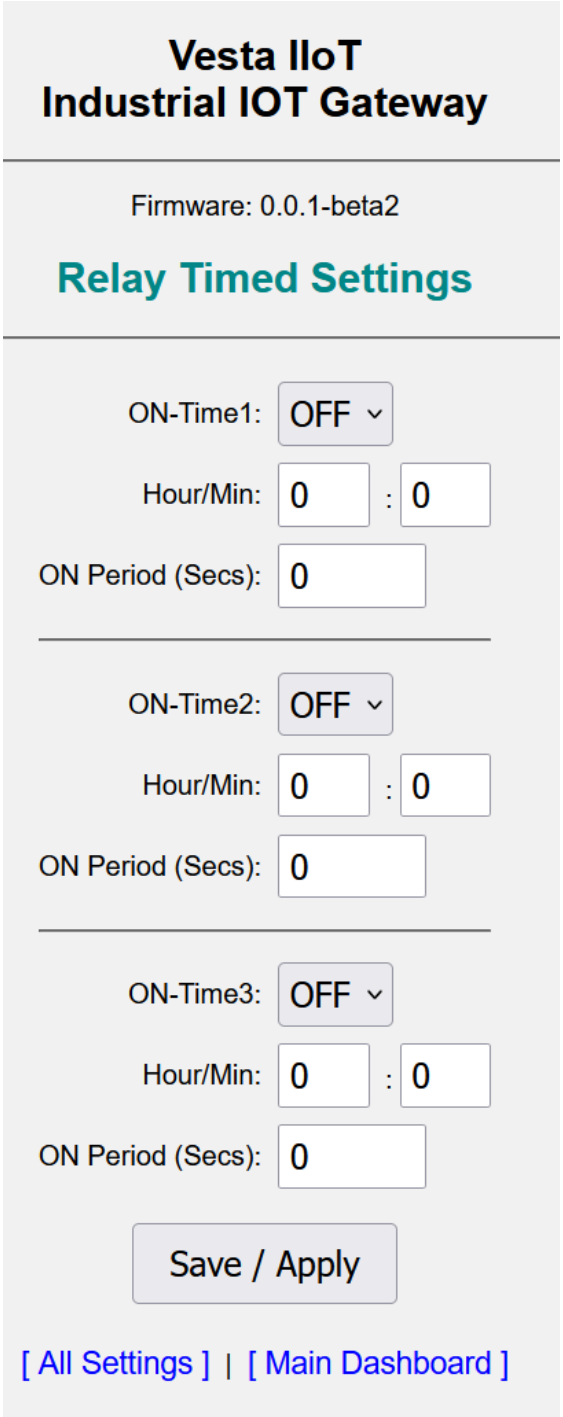
There are three relay activation time slots that can be indivually enabled (Set ON-TimeX to ON)

For each time slot you can set hour and minute of relay activaton and the ON Period in seconds (defines the duration the relay remains switched ON)

When a time slot is configured in activates every day at the set time.

For example one might configure three slots to turn on the relay at 8h 00m, at 12h 0m and at 23h 30m

14 Relay Timed Settings



15 Relay Threshold Settings

To configure relay activation / deactivation based on sensor reading values crossing the configured hi and / or low thresholds click "Threshold Activation" menu item

There are four threshold based relay action items available for configuration (NOTE: only first two shown in the screenshot on the right).

To enable a threshold item, set Thres-X to ON.

Thres-Type sets the threshold type: LOW or HIGH

Sensor: Selects the sensor

Thres-Value configures the threshold value.

Relay-Action configures the relay action that executes when sensor read value crosses the threshold – ON – relay switches ON, OFF – relay switches OFF

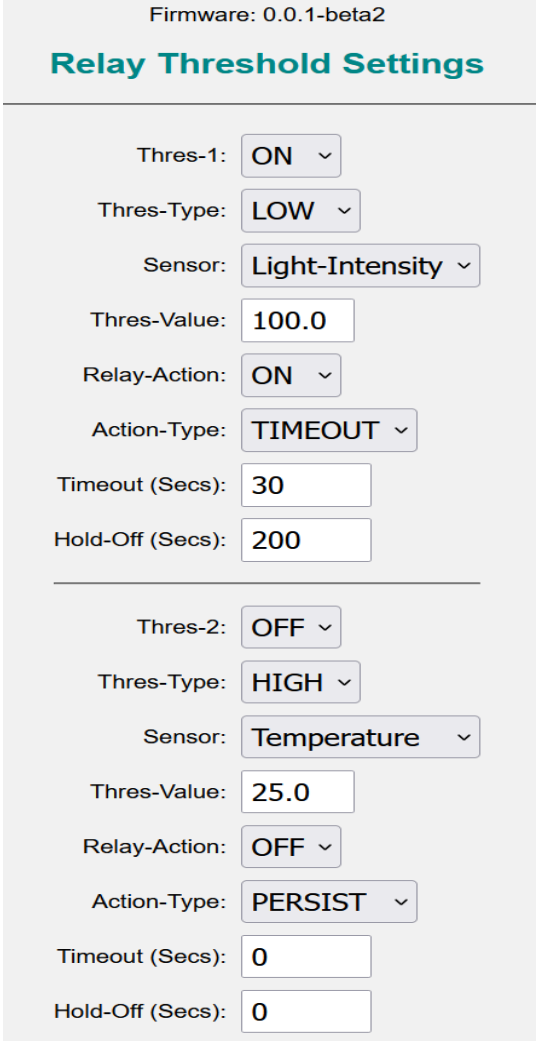
Action-Type selects the Relay action type:

TIMEOUT – After action is done, the relay returns to previous state on configured timeout.

PERSIST – After action is done, relay state persists.

Timeout (Secs) – sets the timeout in seconds after which relay returns to previous state if Action-Type = TIMEOUT.

Hold-Off (Secs) – sets the hold off period in seconds after acton was executed during which the threshold is not checked – if Action-Type = TIMEOUT.



16 Protocol Settings

From the "Settings →""Protocols" page you can configure various communication protocols supported by Vesta-IOT.

MQTT Client, pushes the sensor data after each sensor poll cycle to an external MQTT broker.

Modbus TCP server is a TCP server running Modbus protocol and serving the sensor data in a form of modbus registers.

Modbus RTU client is a serial based protocol which runs over RS232 or RS485 interface. It can read externally connected serial device. This can be used to read additional Modbus sensor(s). The modbus data are represented as data points which can be pushed to MQTT or retrieved by other means – including by externally connected Modbus TCP client.

DNP3 Server is a TCP server running DNP3 protocol and serves the sensor data as DNP3 points.

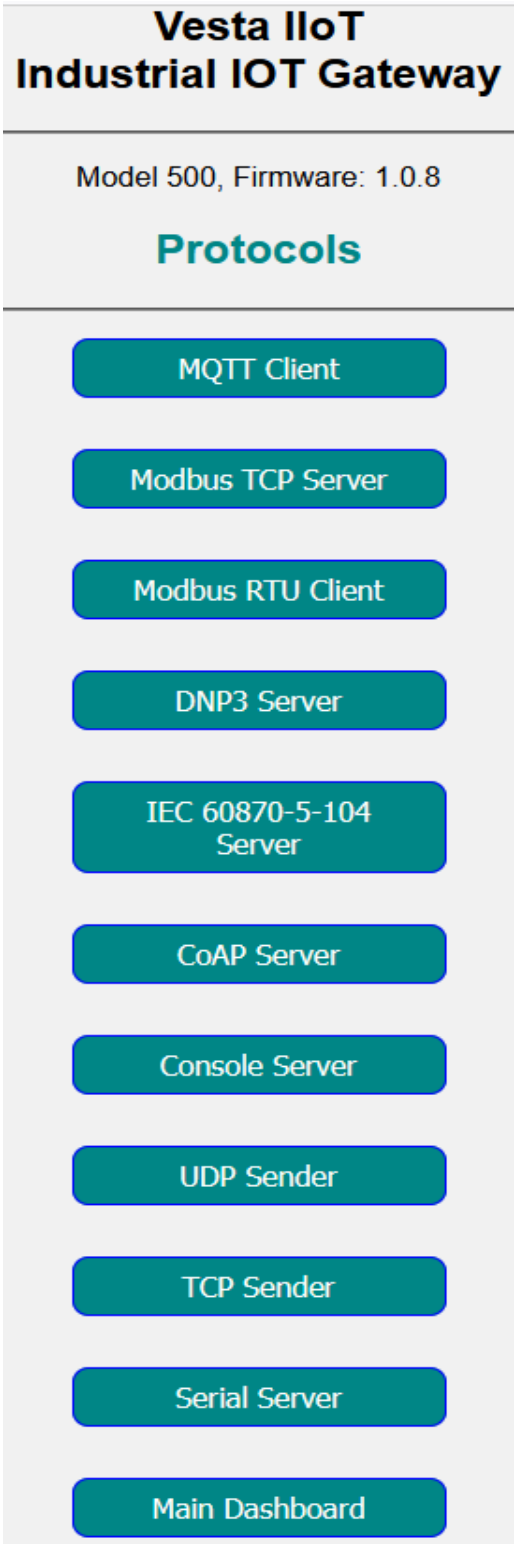
IEC 60870-5-104 is a TCP server that runs IEC 104 protocol serving the sensor data points and allowin the output relay control.

CoAP sever runs over UDP and allows reading all sensor data and to control the output relay.

Console Server is a simple TCP based console service which presents a connecting TCP client with a basic command line interface into the Vesta-IoT device.

UDP Sender and TCP Sender both push the sensor data to external UDP or TCP endpoint.

Serial Server acts as a Terminal server which on one side accepts a TCP connection from an external client and forwards the data between the connection and the serial RS232 interface



17 MQTT Client Settings

Message Queueing Telemetry Transport protocol client can be optionally enabled by clicking the "MQTT Client" button from the Settings menu.

MQTT Net Interface option is used to select the network interface through which the MQTT client shall connect, you can choose "Wired Ethernet", "WIFI" or "Mobile".

In order to send sensor data over MQTT, each in each sensors configuration it is necessary to enable the sensor for sending over MQTT. To do that go to "Settings" →"Sensor Settings" →click on a sensor, and change "Send Over MQTT " to ON and click "Save / Apply".

MQTT Update Intvl – sets the interval in seconds for sending MQTT publish messages containing the sensor data to MQTT broker.

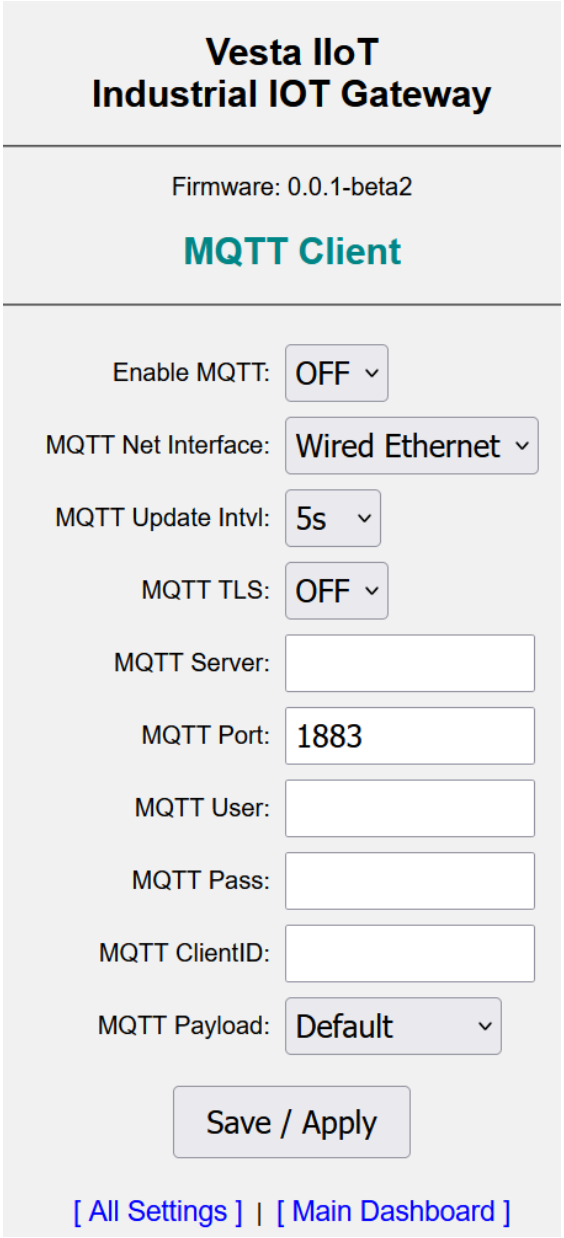
f TLS requires to be enabled by your MQTT server, set MQTT TLS to "ON". (dafault = OFF).
Set the MQTT server's IP address or DNS name in the MQTT server field. The maximum size of this field is 63 characters.

Set the MQTT server's port in the MQTT port field. Normally port 1883 is used for plain non-TLS connection and port 8883 is used for TLS connection.

If your MQTT server requires it, set the MQTT User and Pass fields. The maximum size of these fields is 63 characters.

If your MQTT server requires it, set the MQTT ClientID field. The maximum size of this field is 63 characters.

To set the payload format of MQTT publish messages set the value of MQTT payload field to "Default" or "Thingsboard".



18 UDP Sender Settings

UDP Sender feature forwards sensor readings periodically at configured intervals over UDP to a configured remote UDP peer.

To configure UDP Sender click “Settings” → “UDP Sender”

Send over Interface – sets the network interface over which to send UDP packets – Wired Ethernet or WIFI

Enable UDP Sender – enables or disables UDP Sender feature

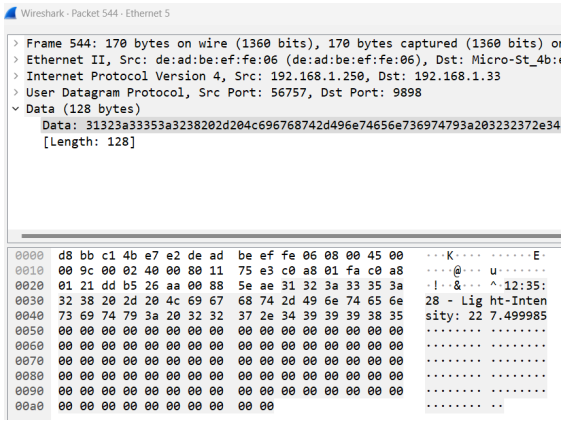
UDP Remote IP address – sets the destination IP address where to send UDP packets

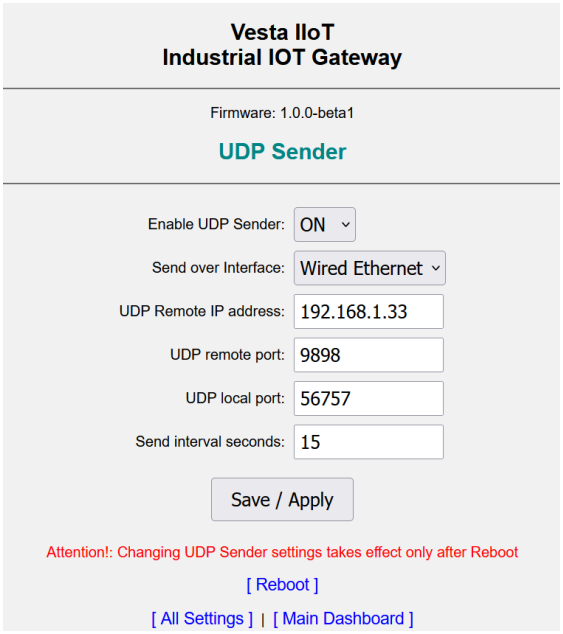
UDP remote port – sets the destination port number where to send UDP packets

UDP local port – sets the local UDP port number to eb used by UDP sender

Send Interval seconds – configures UDP sending interval in seconds

Each sensor reading is sent in a UDP paylod of fixed size – 128 bytes, the content is ASCII zero terminated string with timestamp, sensor name followed by sensor value.





19 TCP Sender Settings

TCP Sender feature forwards sensor readings periodically at configured intervals over a TCP connection.

To configure TCP Sender click “Settings” → “TCP Sender”

Enable TCP Sender – enables or disables TCP Sender feature

Send over Interface – sets the network interface over which TCP sender runs – Wired Ethernet or WIFI

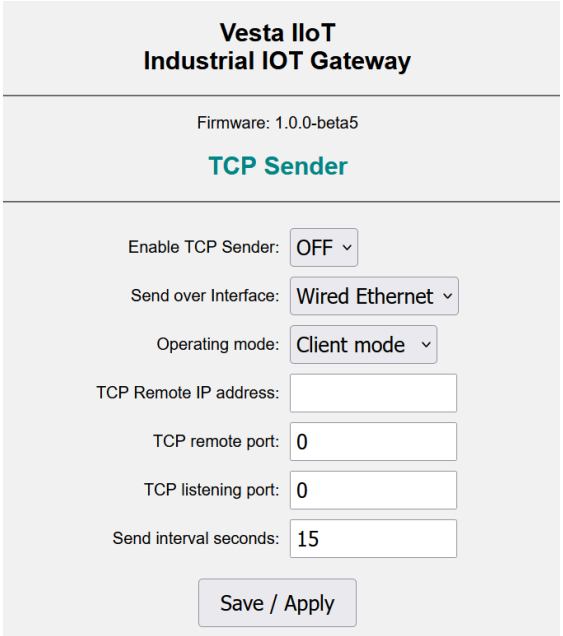
Operating mode: Client mode or Server mode – configure TCP sender to run in TCP client or TCP server mode respectively

TCP Remote IP address – sets the destination IP address of TCP server (in Client mode)

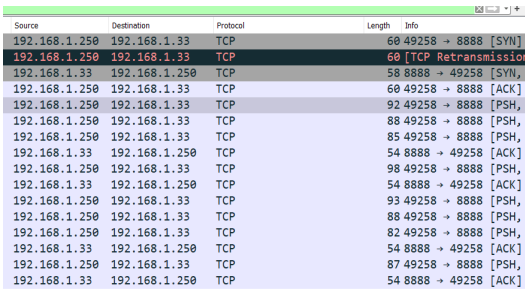
TCP remote port – sets the TCP port of server to connect to (in Client mode)

TCP listening port – sets the port on which to listen for an incoming TCP connection (in Server mode)

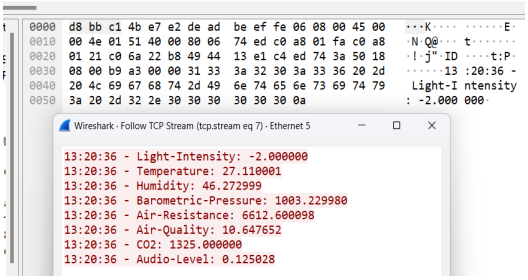
Send Interval seconds – configures data sending interval in seconds



The individual sensor readings are separated with the CR LF in the TCP stream:



Source	Destination	Protocol	Length	Info
192.168.1.250	192.168.1.33	TCP	60	49258 → 8888 [SYN]
192.168.1.250	192.168.1.33	TCP	60	[TCP Retransmissio
192.168.1.33	192.168.1.250	TCP	58	8888 → 49258 [SYN,
192.168.1.250	192.168.1.33	TCP	60	49258 → 8888 [ACK]
192.168.1.250	192.168.1.33	TCP	92	49258 → 8888 [PSH,
192.168.1.250	192.168.1.33	TCP	88	49258 → 8888 [PSH,
192.168.1.250	192.168.1.33	TCP	85	49258 → 8888 [PSH,
192.168.1.33	192.168.1.250	TCP	54	8888 → 49258 [ACK]
192.168.1.250	192.168.1.33	TCP	98	49258 → 8888 [PSH,
192.168.1.33	192.168.1.250	TCP	54	8888 → 49258 [ACK]
192.168.1.250	192.168.1.33	TCP	93	49258 → 8888 [PSH,
192.168.1.250	192.168.1.33	TCP	88	49258 → 8888 [PSH,
192.168.1.250	192.168.1.33	TCP	82	49258 → 8888 [PSH,
192.168.1.33	192.168.1.250	TCP	54	8888 → 49258 [ACK]
192.168.1.250	192.168.1.33	TCP	87	49258 → 8888 [PSH,
192.168.1.33	192.168.1.250	TCP	54	8888 → 49258 [ACK]



20 MODBUS TCP Server Settings

To configure MODBUS TCP server click “MODBUS Server” button from the “Settings” sub-menu.

Enable Modbus Server – Enables or Disables the MODBUS server

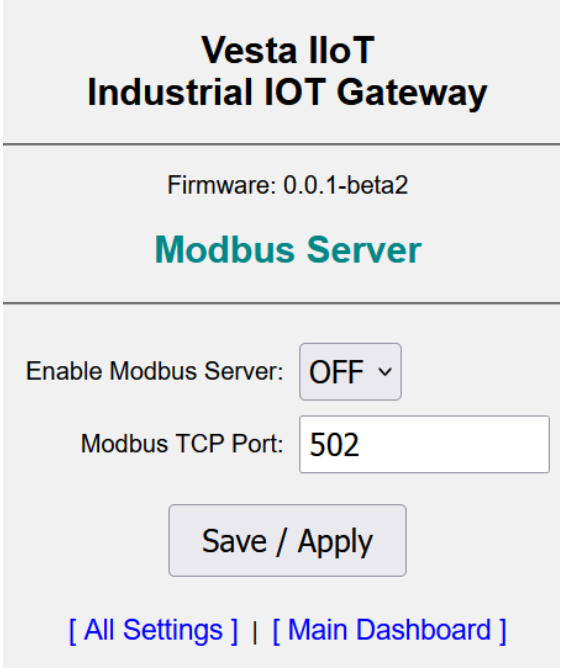
Modbus TCP port – sets the TCP port number on which the Modbus server listens for a connection from a Modbus TCP client.

Modbus TCP server has a memory map consisting of eight HR registers (Holding Registers) and one CO (Coil) Output register

Each of the on-board sensors has associated unique HR register index.

Each HR register has length of 32 bits (two adjacent 16 bit HR registers). The data storage format in these 32 bits is IEEE Std 754 Float number that holds the sensor value.

Reg. Type	Reg. Index	Size	Format	Sensor
HR (16bit)	1000	2	Float	Light Intensity (Lux)
HR (16bit)	1002	2	Float	Temperature C
HR (16bit)	1004	2	Float	Humidity % rH
HR (16bit)	1006	2	Float	Pressure hPA
HR (16bit)	1008	2	Float	Air Resistance
HR (16bit)	1010	2	Float	Air Quality (IAQ)
HR (16bit)	1012	2	Float	CO2 ppm
HR (16bit)	1014	2	Float	Audio Level (%)
CO (1 bit)	3000	1	Single bit	Power Relay



NOTE: Modbus server runs simultaneously on Wired Ethernet and WIFI interface

21 DNP3 TCP Server Settings

To configure DNP3 TCP server click "DNP3 Server" button from the "Settings" sub-menu.

Enable DNP3 Server – Enables or Disables the DNP3 server

TCP Listening port – sets the TCP port number on which the DNP3 server listens for a connection from a DNP3 TCP client.

DNP3 source and destination address configure the data link layer addresses of the DNP3 link.

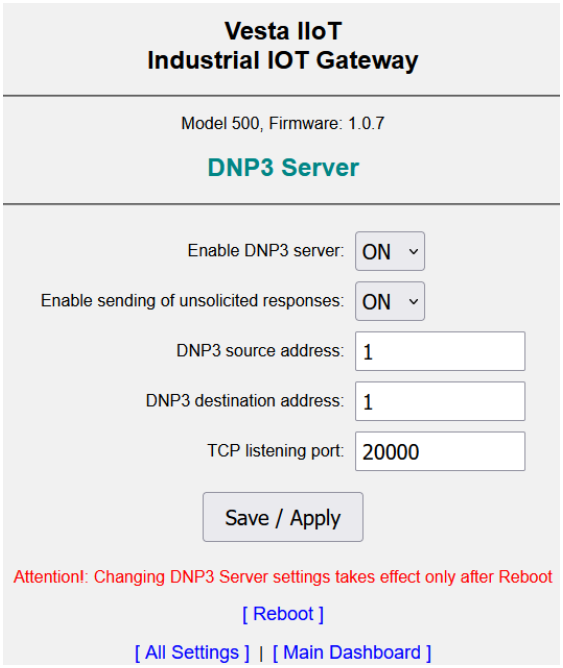
Modbus TCP server has a memory map consisting of eight HR registers (Holding Registers) and one CO (Coil) Output register

Each of the on-board sensors has an associated unique DNP3 Object

Sensor data is reported as Group 30 (Analog Inputs), Variation 5 - IEEE Std 754 Float (32 bit), object indexes 0 - 7.

The output power relay can be controlled by sending Group 12v1 command to index 0.

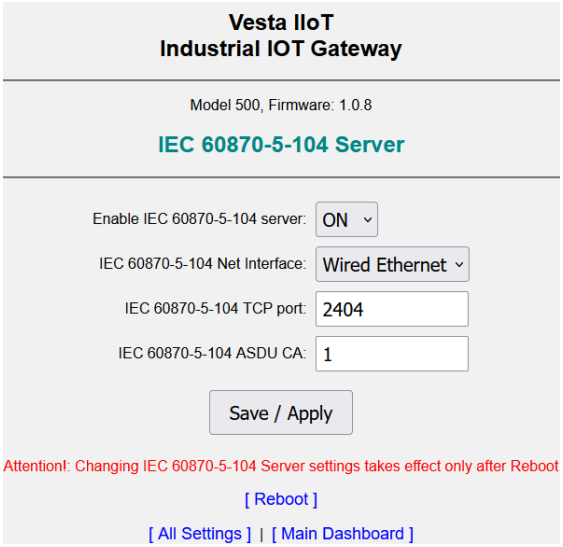
The following table shows DNP3 Group / Variation and object index mapping to on-board sensors and the power relay



DNP3 Group	DNP3 Index	Var	Format	Sensor
30	0	5	Float	Light Intensity (Lux)
30	1	5	Float	Temperature C
30	2	5	Float	Humidity % rH
30	3	5	Float	Pressure hPA
30	4	5	Float	Air Resistance
30	5	5	Float	Air Quality (IAQ)
30	6	5	Float	CO2 ppm
30	7	5	Float	Audio Level (%)
12	0	1	CROB	Power Relay

22 IEC 60870-5-104 Server Settings

To configure IEC 60870-5-104 server click " IEC 60870-5-104 Server" button from the "Settings" → "Protocols" sub-menu.



Enable IEC 60870-5-104 Server – Enables or Disables the IEC 60870-5-104 server.

IEC 60870-5-104 Net Interface: Selects interface on which the server listens for client connections: Wired Ethernet or WIFI.

IEC 60870-5-104 TCP port: sets port number on which the server listens for client connections.

IEC 60870-5-104 ASDU CA: sets the ASDU CA address of the IEC 60870-5-104 session.

The folliowing table shows the mapping of supported IEC 60870-5-104 IOAs (Information Object Addresses) to reported inputs (sensors) and outputs (Power Relay):

Type ID	IOA	Format	Resource
1-Single Pt. M_SP_NA_1	201	Single bit	Relay output state
45-Single Cmd C_SC_NA_1	200	Single bit	Command sets the relay output state.
13-Measured F M_ME_NC_1	100	32bit Float	Light Intensity (Lux)
13-Measured F M_ME_NC_1	101	32bit Float	Temperature C
13-Measured F M_ME_NC_1	102	32bit Float	Humidity % rH
13-Measured F M_ME_NC_1	103	32bit Float	Pressure hPA
13-Measured F M_ME_NC_1	104	32bit Float	Air Resistance
13-Measured F M_ME_NC_1	105	32bit Float	Air Quality (IAQ)
13-Measured F M_ME_NC_1	106	32bit Float	CO2 ppm
13-Measured F M_ME_NC_1	107	32bit Float	Audio Level (%)

23 Modbus RTU Client Settings

To configure Modbus RTU client click “Modbus RTU Client” button from the “Settings” →“Protocols” sub-menu.

Enable Modbus RTU client – Enables or Disables the Modbus RTU client server

Serial Interface Channel – selects the serial interface on which the Modbus RTU Client runs – RS232 or RS485

You can configure specific serial settings, such as baud rate by clicking [Configure Serial] link

Modbus Slave Address: Sets the Modbus slave address to which the Modbus RTU client shall communicate. The valid value range is from 1 to 254

Poll Interval (Secs) – configures the slave polling interval.

Reply Timeout (Secs) – configures the reply wait timeout.

To successfully setup the Modbus RTU client it is necessary to confiure the data points
– click [Setup Data Points] link to do it.

When finished configuration, Click “Save / Apply”

Note that you will also need to reboot the device for the settings to take effect, you can do it by following the [Reboot] link.

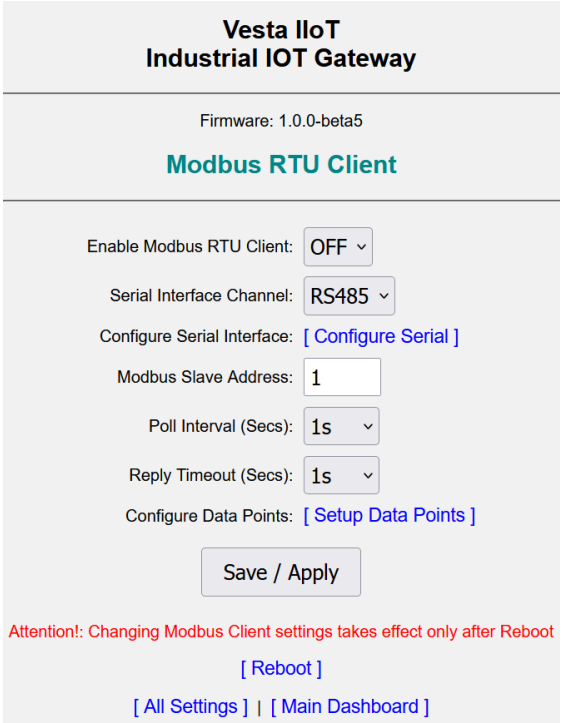
On the Modbus RTU Client Points page:
Point X Enable – enables reading of the data point from the slave device

Point Name – user configurable text name of the point

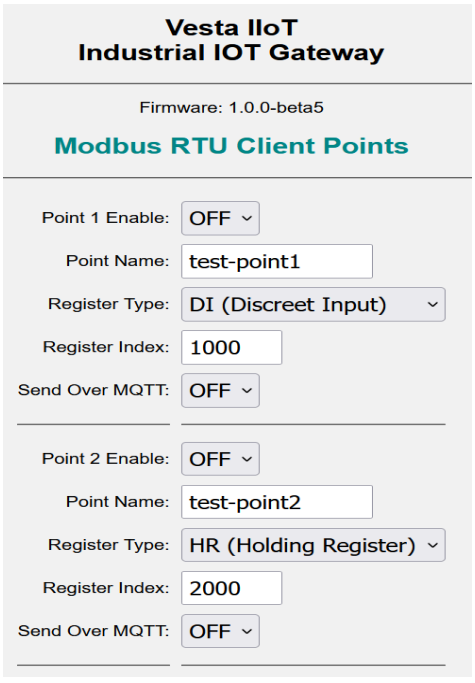
Register Type – selects the Modbus register type

Register Index – selects the register index

Send Over MQTT – enables sending of the polled register value over MQTT



There are total of 8 data point entries supported for configuration, only first two show in the screenshot



24 Modbus RTU Client Usage

When Modbus RTU client is configured and running the data points read from the Modbus slave connected over the serial interface can be used in different ways:

- Can be viewed via "View Status" →"Modbus RTU Client" menu – see screenshot on the right

- Can be sent / pushed over MQTT to an MQTT broker

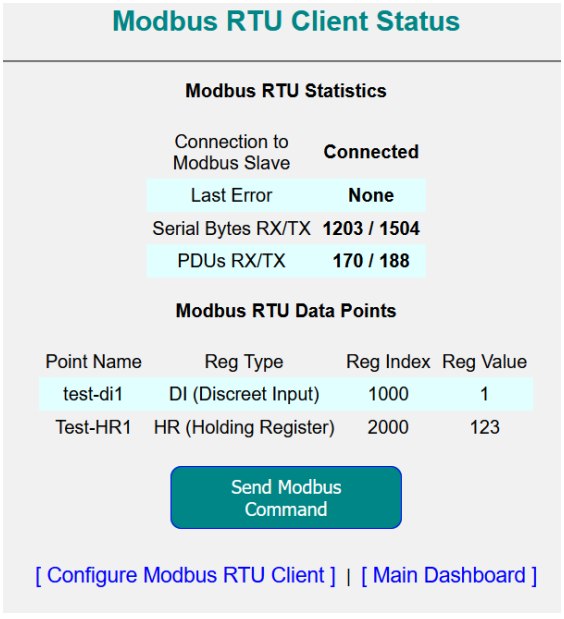
- Can be sent / pushed as part of UDP Sender or TCP Sender featutes

- Can be exposed as Modbus TCP server side points when Modbus TCP Server is enabled

- Can be served by the CoAP server

Viewing Modbus RTU Client Status

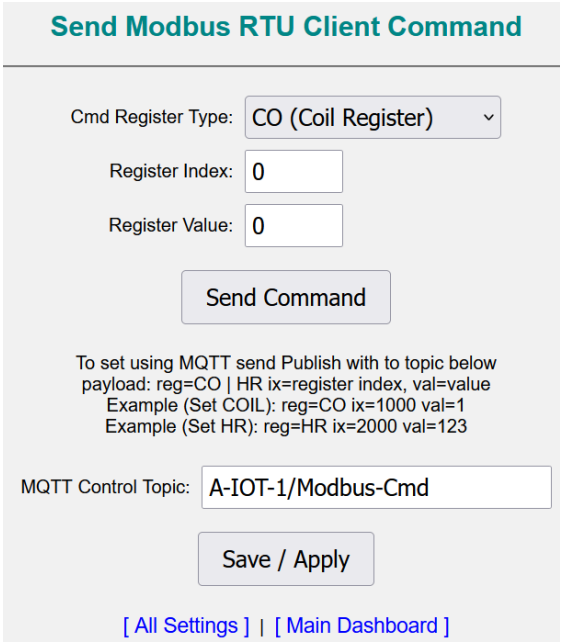
To view Modbus RTU client status click "View Status" →"Modbus RTU Client"



Modbus RTU Statistics show the count of PDUs transferred and any errors. Modbus RTU data points shows the values received for data points

25 Sending Modbus RTU Command

To send the command from the web UI, click "Send Modbus Command" button from "Modbus RTU Client Status" page.



On the command page, select the command register type (Coil or Holding Register), set register index and register value and click "Send Command"

A Modbus command can be sent from an MQTT client by sending a publish message to the configured MQTT Control Topic.

The payload should be as shown in the screenshot:

E.g.:
reg=CO ix=1000 val=1

26 CoAP Server Settings

To configure CoAP server click “CoAP Server” button from the “Settings” sub-menu.

CoAP Net Interface – selects the network interface on which the CoAP server runs – Wired Ethernet or WIFI

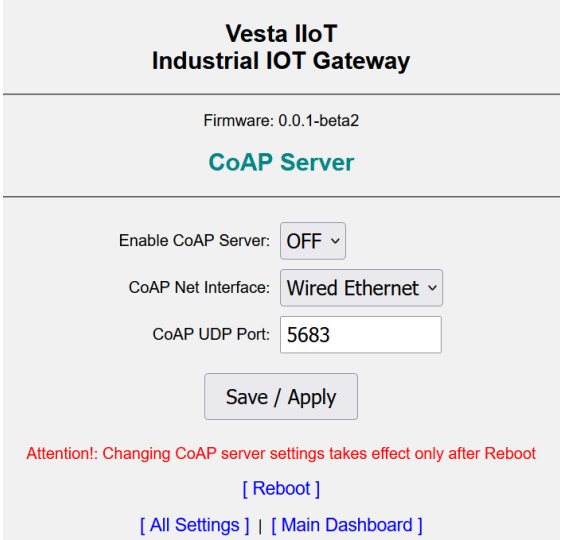
Enable CoAP Server – Enables or Disables the CoAP server

CoAP UDP port – sets the UDP port number on which the CoAP server receives requests from a CoAP client.

CoAP protocol server responds to client requests by sending sensor data in response.

Each sensor has an associated CoAP resource URI:

In the table below VESTA-IOT-x – x is the unique device ID. (1, 2 … etc). The table shows the resource names for device 1.



CoAP resource	Sensor
VESTA-IOT-1/Light-Lux	Light Intensity (Lux)
VESTA-IOT-1/Temperature	Temperature C
VESTA-IOT-1/Humidity	Humidity % rH
VESTA-IOT-1/Pressure	Pressure hPA
VESTA-IOT-1/Air-Resistance	Air Resistance
VESTA-IOT-1/Air-Quality	Ait Quality IAQ
VESTA-IOT-1/CO2	CO2 ppm
VESTA-IOT-1/Audio-Level	Audio Level (%)
/relay-on /relay-off	Power Relay

27 Console Server Settings

To configure Console TCP server click “Console Server” button from the “Settings” sub-menu.

Enable Console Server – Enables or Disables the Console TCP server

Console Server TCP port – sets the TCP port number on which the console server listens for a connection from a TCP client.

Set Console Password – when set the console server demaps the user to enter the matching password before it will allow access to the functions availabe through console

Echo On – enables or disables echoing the received characters back to the client

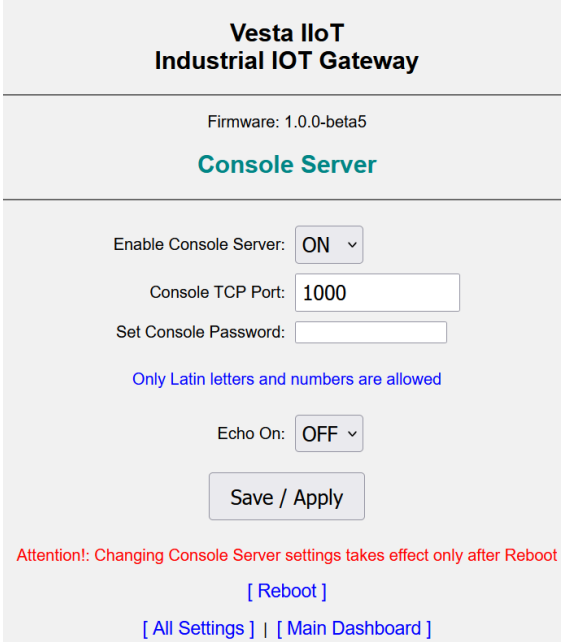
Console Server is a simple raw TCP server. A user can open a TCP connection to it using for example Putty session using TCP “Raw” or “Other” mode and specifying TCP port that is set in this page.

Once TCP connection is established, press single <ENTER> and you will be presented with Vesta-IoT command prompt:

```
VESTA-IOT-1 >

VESTA-IOT-1 >?
=== Vesta-IIOT Gateway. Command syntax: ===

show version - show version and build date
logread - [-f] shows log, -f -monitor log
show sensors - Shows values of on-board sensors
reboot - Reboots the device
tftp update - <ip=server ip> Starts firmware update
? - Prints this help
```



NOTE: Console Server runs simultaneously on WIFI and Wired Ethernet.

28 Serial Server Settings

To configure Serial Server click “Serial Server” button from the “Settings” sub-menu.

Enable Serial Server – Enables or Disables the Serial Server

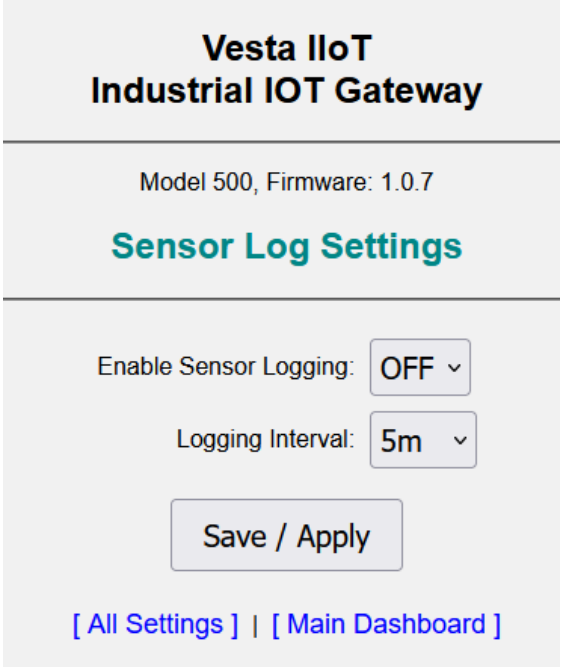
Serial Server Listening TCP port – sets the TCP port number on which the serial server listens for a connection from a TCP client.

Serial Server is a TCP server which when the TCP connection is established forwards the data transparently between the TCP connection and the Serial RS232 interface.

NOTE: To run the serial server you will also need to enable and configure RS232 serial interface in “Interfaces” →“Serial-RS232”

29 Sensor Log Settings

To configure the data logger feature (also called in this product "Sensor Log", goto "Config Settings" → "Sensor Log Settings"



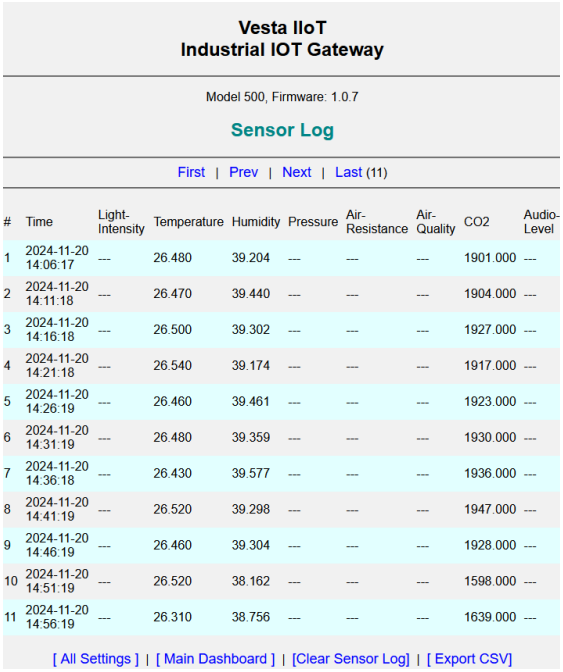
Enable Sensor Log – enables or disables the data logger feature.

Logging interval – selects the interval at which the sensor readings are saved into non-volatile memory of of the sensor log.

When finished changing the settings,
click "Save / Apply"

30 Viewing Sensor Log

To view the sensor log records stored in non-volatile memory, from the main page, go to "View Status" →"Sensor Log"



Vesta IIoT
Industrial IOT Gateway

Model 500, Firmware: 1.0.7

Sensor Log

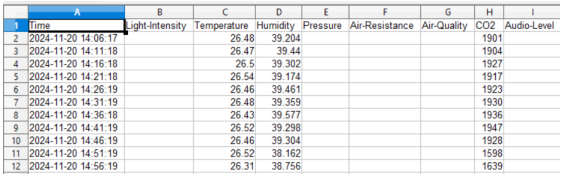
First | Prev | Next | Last (11)

#	Time	Light-Intensity	Temperature	Humidity	Pressure	Air-Resistance	Air-Quality	CO2	Audio-Level
1	2024-11-20 14:06:17	---	26.480	39.204	---	---	---	1901.000	---
2	2024-11-20 14:11:18	---	26.470	39.440	---	---	---	1904.000	---
3	2024-11-20 14:16:18	---	26.500	39.302	---	---	---	1927.000	---
4	2024-11-20 14:21:18	---	26.540	39.174	---	---	---	1917.000	---
5	2024-11-20 14:26:19	---	26.460	39.461	---	---	---	1923.000	---
6	2024-11-20 14:31:19	---	26.480	39.359	---	---	---	1930.000	---
7	2024-11-20 14:36:18	---	26.430	39.577	---	---	---	1936.000	---
8	2024-11-20 14:41:19	---	26.520	39.298	---	---	---	1947.000	---
9	2024-11-20 14:46:19	---	26.460	39.304	---	---	---	1928.000	---
10	2024-11-20 14:51:19	---	26.520	38.162	---	---	---	1598.000	---
11	2024-11-20 14:56:19	---	26.310	38.756	---	---	---	1639.000	---

[All Settings] | [Main Dashboard] | [Clear Sensor Log] | [Export CSV]

From this page you can browse through the sensor log entries, clear the sensor log or export it to CSV file by clicking the corresponding links at the bottom of the screen.

Clicking "Export CSV" downloads the current the sensor log file in CSV format which can be opened from an Excel style application, a screenshot shown below:



	A	B	C	D	E	F	G	H	I
1	Time	Light-Intensity	Temperature	Humidity	Pressure	Air-Resistance	Air-Quality	CO2	Audio-Level
2	2024-11-20 14:06:17		26.48	39.204				1901	
3	2024-11-20 14:11:18		26.47	39.44				1904	
4	2024-11-20 14:16:18		26.5	39.302				1927	
5	2024-11-20 14:21:18		26.54	39.174				1917	
6	2024-11-20 14:26:19		26.46	39.461				1923	
7	2024-11-20 14:31:19		26.48	39.359				1930	
8	2024-11-20 14:36:18		26.43	39.577				1936	
9	2024-11-20 14:41:19		26.52	39.298				1947	
10	2024-11-20 14:46:19		26.46	39.304				1928	
11	2024-11-20 14:51:19		26.52	38.162				1598	
12	2024-11-20 14:56:19		26.31	38.756				1639	

31 Direct Relay Control

On-board power relay can be directly controlled from the web UI by clicking “Set Relay” button from the main dashboard screen.

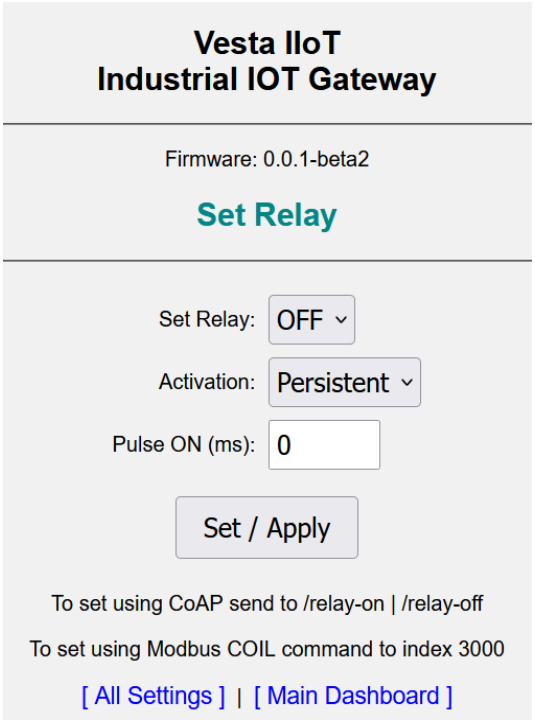
To set the relay, change “Set Relay” selector to “ON” or “OFF”,

Activation option sets the type of relay action:

“Persistent” - relay state shall persist
“Pulse ON” - When “Set Relay” = ON, a pulse from OFF to ON shall be performed with the ON period set in “Pulse ON (ms)” field.

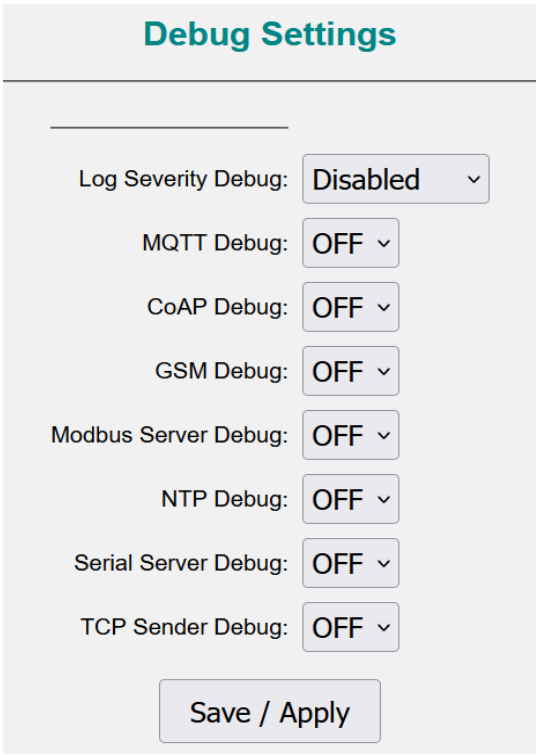
To execute the relay action click

“Set / Apply” button



32 Debug Settings

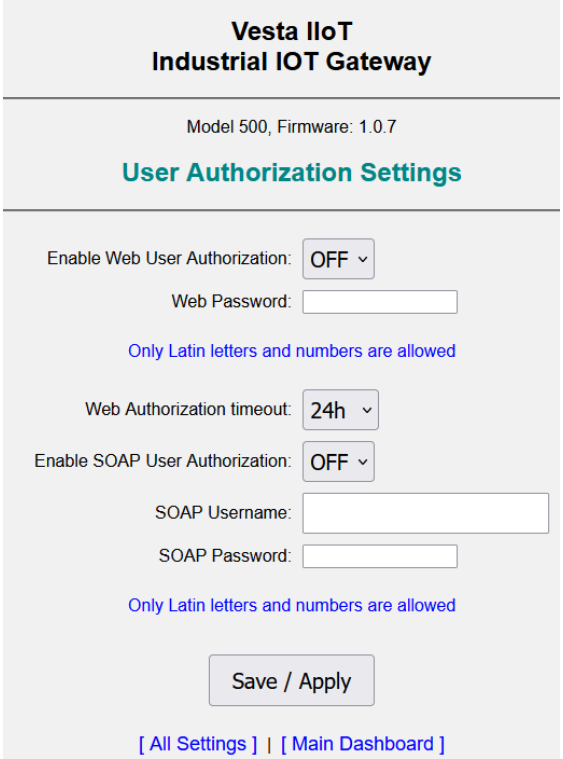
Debug settings screen accessible from “Settings” → “Debug Settings” control various debug options:



33 User Authorization Settings

On this page, accessible from "Settings" →"User Authorization Settings" it is possible to enable password authentication for Web UI clients. By default web password authorization is OFF, here you can change it to ON and set the password and the authorization timeout.

This screen is also used to configure SOAP Username and Password



34 Loading Factory Settings

To reset the configuration to default "factory" settings click "Settings" →"Factory Reset"

To proceed with loading the default settings click "Apply"

35 Factory Reset via Physical Button

Press and hold "Factory Reset" button for >10 seconds to cause loading of the factory reset configuration

36 Rebooting Vesta-IoT

To reboot the device, clikc “Main Dashboard” → “System” →“Reboot”

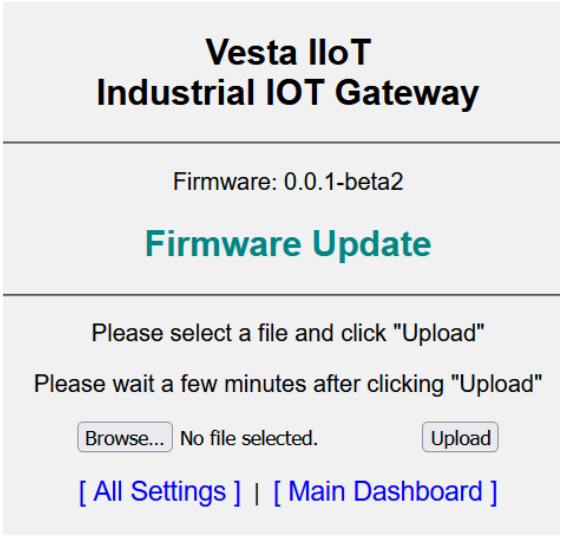
37 Rebooting by Physical Button

You can cause the Vesta-IoT device to reset by pressing and releasing the “HW RESET” button shown in the picture on the left

38 Firmware Update (via Web UI)

There are two methods to update the firmware on Vesta-IoT device.

To update via the web ui, click “Main Dashboards” →“System” →“Firmware Update”.



Select the firmware image file by clicking “Browse” button.

Once selected, click “Upload” button.

Firmware upgrade procedure shall take a few minutes. At the end of the procedure the result of operation shall be displayed

At the end of the process click “Reboot” link and proceed with the device reboot.

39 Firmware Update (via TFTP)

1. Run TFTP server on your PC and make sure the working directory contains the firmware file vesta_iot_gw.ino.bin
2. Make sure “Console Server” is ON in “Settings” →“Console Server”
3. Connect a TCP client (for example Putty to port number confiugred on the router and IP address of wired eth interface (192.168.1.250 by default) or to WIFI AP IP address 192.168.4.1
4. Hit <ENTER>
5. type command:
   tftp update ip=ipaddress_of_your_pc
6. Hit <ENTER>
7. TFTP transfer of the firmware file should start
8. The result of operation shall be shown in the console server session
9. Once firmware update is complete you can reboot the box by going to “System” →“Reboot”

40 VENDOR INFORMATION

Developed by VestaTel
www.vestatel.eu

info@vestatel.eu

Registered in Poland

NIP: 6040233038

REGON: 522919182

41 SOAP API Specification

Vesta IOT device runs an HTTP / SOAP API server on a fixed TCP port 80 – shared with the regular web server.

SOAP API can be used by an externally connecting HTTP / SOAP client to retrieve information from the device – such as sensor readings and to control the device – for example to turn the power relay ON or OFF.

41.1 Introduction

41.1.1 Security Credentials

All the SOAP requests decribed below contain the security block containing the SOAP user name and password. The use and the presence of this block in the request is mandatory when it is enabled in the SOAP server configuration on the device.

To set or unset the SOAP user / password credentials from the web UI's main screen (Dashboard), go to "Config Settings" → "User Authorization Settings".

If SOAP user / password authentication is disabled (SOAP Username and SOAP Password are unset) the `<wsse:Security>` block shall be ignored by the server and all requests shall be served.

41.1.2 General Request – Response Structure

- Each request and response consists of HTTP header and HTTP body.
- Each line terminates with <CR><LF> characters (0x0d 0x0a)
- The Header and Body are separated with <CR><LF><CR><LF> characters (0x0d 0x0a 0x0d 0x0a)

41.1.3 Header Content-Length

All the requests must contain `Content-Length: <length>` tag value pair, where the length = integer number of bytes that comprises the request or response body - starting from `<?xml version="1.0"?>` and ending with `</soap:Envelope>`

41.2 Product Information

This request is used to read device's hardware model and / or firmware version.

Request

```
POST /soap_api HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "/product_info"
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>myUsername</wsse:Username>
        <wsse:Password>myPassword</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <GetProductInfoRequest xmlns="/">
      <ProdInfoKind>HwModel</ProdInfoKind>
      <ProdInfoKind>FirmwareVer</ProdInfoKind>
    </GetProductInfoRequest>
  </soap:Body>
</soap:Envelope>
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Body>
    <GetProductInfoResponse xmlns="/">
      <HwModel>Vesta-IoT</HwModel>
      <FirmwareVer>1.0.0</FirmwareVer>
    </GetProductInfoResponse>
  </soap:Body>
</soap:Envelope>
```

41.3 Reading On-Board Sensors

This request is used to read one or more of the on-board sensors.

Request

In the following example, the information about all 2 out of of 8 on-board sensors is requested, but the request may contain any number of <SensorName></SensorName> tag elements (more than zero) to read the required sensors.

```
POST /soap_api HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "/sensor_info"
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>myUsername</wsse:Username>
        <wsse:Password>myPassword</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <GetSensorInfoRequest xmlns="/">
      <SensorName>Light-Intensity</SensorName>
      <SensorName>Temperature</SensorName>
    </GetSensorInfoRequest>
  </soap:Body>
</soap:Envelope>
```

Response

Note: the example response contains information about all sensors, however the response can contain only a few or just one sensor, depending on the request.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Body>
    <GetSensorInfoResponse xmlns="/">
      <Sensor>
          <SensorName>Light-Intensity</SensorName>
          <SensorUnits>Lux</SensorUnits>
          <SensorValue>123.5</SensorValue>
      </Sensor>
      <Sensor>
          <SensorName>Temperature</SensorName>
          <SensorUnits>C</SensorUnits>
          <SensorValue>123.5</SensorValue>
      </Sensor>
      <Sensor>
          <SensorName>Humidity</SensorName>
          <SensorUnits>%</SensorUnits>
          <SensorValue>123.5</SensorValue>
      </Sensor>
       <Sensor>
          <SensorName>Pressure</SensorName>
          <SensorUnits>hPa</SensorUnits>
          <SensorValue>123.5</SensorValue>
      </Sensor>
      <Sensor>
          <SensorName>Air-Quality</SensorName>
          <SensorUnits>IAQ</SensorUnits>
          <SensorValue>123.5</SensorValue>
      </Sensor>
      <Sensor>
          <SensorName>Air-Resistance</SensorName>
          <SensorUnits>ohm</SensorUnits>
          <SensorValue>123.5</SensorValue>
      </Sensor>
      <Sensor>
          <SensorName>CO2</SensorName>
          <SensorUnits>ppm</SensorUnits>s
          <SensorValue>123.5</SensorValue>
```

```
      </Sensor>
      <Sensor>
          <SensorName>Audio-Level</SensorName>
          <SensorUnits>%</SensorUnits>
          <SensorValue>123.5</SensorValue>
      </Sensor>
      </GetSensorInfoResponse>
  </soap:Body>
</soap:Envelope>
```

41.4 Reading Relay State

Request

Note1: Possible values of State: (tag <RelayState>): ON | OFF

```
POST /soap_api HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "/relay_state"
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>myUsername</wsse:Username>
        <wsse:Password>myPassword</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <GetRelayStateRequest xmlns="/">
    </GetRelayStateRequest>
  </soap:Body>
</soap:Envelope>
```

Response

Note1: Possible values of State: (tag <RelayState>): ON | OFF

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Body>
    <GetRelayStateResponse xmlns="/">
```

```
      <RelayState>State</RelayState>
    </GetRelayStateResponse>
  </soap:Body>
</soap:Envelope>
```

41.5 Setting Relay State

This request is used to set relay “state” ON/OFF.

Note1: Possible values of State: (tag <RelayState>): ON | OFF.
Note2: Possible values of Mode: (tag <RelayAction>) Persistent | Pulse.
Note3: tag value pair `<PulseOnMs>PulseTime</PulseOnMs>` applies in the case where RelayAction = Pulse
PulseTime = ON pulse duration in milliseconds

Request

```
POST /soap_api HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "/set_relay"
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>myUsername</wsse:Username>
        <wsse:Password>myPassword</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <SetRelayStateRequest xmlns="/">
      <RelayState>state</RelayState>
      <RelayAction>Mode</RelayAction>
      <PulseOnMs>PulseTime</PulseOnMs>
    </SetRelayStateRequest>
  </soap:Body>
</soap:Envelope>
```

Response

Note1: State = Relay state after operation started executing, possible values: ON | OFF

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: <length>

<?xml version="1.0"?>
```

```
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Body>
    <SetRelayStateResponse xmlns="/">
      <RelayState>State</RelayState>
    </SetRelayStateResponse>
  </soap:Body>
</soap:Envelope>
```

41.6 Reading Time

The response to this request will show the current time set in the device

Request

```
POST /soap_api HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "/read_time"
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>myUsername</wsse:Username>
        <wsse:Password>myPassword</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <GetTimeRequest xmlns="/"></GetTimeRequest>
  </soap:Body>
</soap:Envelope>
```

Response

Note1: TimeSpec has the following syntax:
YYYY-MM-DD HH:MM:SS, example: `2024-10-14 11:19:27`

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Body>
    <GetTimeResponse xmlns="/">
      <CurrentTime>TimeSpec</CurrentTime>
    </GetTimeResponse>
```

```
  </soap:Body>
</soap:Envelope>
```

41.7 Setting Time

Request

Note1: TimeSpec has the following syntax:
YYYY-MM-DD HH:MM:SS, example: `2024-10-14 11:19:27`

```
POST /soap_api HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "/set_time"
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>myUsername</wsse:Username>
        <wsse:Password>myPassword</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <SetTimeRequest xmlns="/">
      <CurrentTime>TimeSpec</CurrentTime>
    </SetTimeRequest>
  </soap:Body>
</soap:Envelope>
```

Response

Note1: TimeSpec has the following syntax:
YYYY-MM-DD HH:MM:SS, example: `2024-10-14 11:19:27`

Note2: TimeSpec in this response contains the time read from RTC (real time clock) after it has been set.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Body>
    <SetTimeResponse xmlns="/">
      <CurrentTime>TimeSpec</CurrentTime>
    </SetTimeResponse>
```

```
  </soap:Body>
</soap:Envelope>
```

41.8 Reset to Factory Default Settings

Request

```
POST /soap_api HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "/load_factory_settings"
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>myUsername</wsse:Username>
        <wsse:Password>myPassword</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <LoadFactorySettingsRequest xmlns="/">
    </LoadFactorySettingsRequest>
  </soap:Body>
</soap:Envelope>
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Body>
    <LoadFactorySettingsResponse xmlns="/">
       <Status>Success</Status>
    </LoadFactorySettingsResponse>
  </soap:Body>
</soap:Envelope>
```

41.9 Initiate Device Reload

Request

```
POST /soap_api HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "/reload_device"
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>myUsername</wsse:Username>
        <wsse:Password>myPassword</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <ReloadDeviceRequest xmlns="/">
    </ReloadDeviceRequest>
  </soap:Body>
</soap:Envelope>
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Body>
    <ReloadDeviceResponse xmlns="/">
        <Status>Success</Status>
    </ReloadDeviceResponse>
  </soap:Body>
</soap:Envelope>
```

41.10 Error Responses

Error response may be rest in reply to any request. All error responses has the following general format:

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset=utf-8
Content-Length: <length>

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="<http://www.w3.org/2003/05/soap-envelope>">
  <soap:Body>
    <soap:Fault>
      <faultstring>Error text specific to api request</faultstring>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

<lengh> = number of bytes of the response body starting from first byte of `<?xml version="1.0"?>` and endinf with the last byte of `</soap:Envelope>`