# periSNOOP 4-20mA

## Single Pair Ethernet sensor monitoring device

Datasheet

Perinet
Seamless IoT Connectivity

**Abstract**

This datasheet describes the *periSNOOP 4-20mA*, a sensor monitoring device designed for industrial use. This device allows easy remote access and control through a network using Single Pair Ethernet (SPE), a modern network standard. It is designed to be integrated into existing installations without needing to alter those systems.

The periSNOOP 4-20mA is equipped with advanced communication features and state-of-the-art security measures. These features ensure that it can reliably transmit sensor data to digital systems.

Doc-No.: PRN.000.069 rev: 3 2024-05-13

# Document Information

| | |
|---|---|
| **Title** | periSNOOP 4-20mA |
| **Subtitle** | Single Pair Ethernet sensor monitoring device |
| **Type** | Datasheet |
| **Status** | Release |
| **Version** | 3 |
| **Date** | 2024-05-13 |
| **Disclosure Restriction** | |

Intellectual property rights in the products, names, logos and designs included in this document may be held by *Perinet* or third parties. Copying, reproduction, modification or disclosure to third parties of this document or any part thereof is only permitted with the express written permission of *Perinet*.

The information contained herein is provided “as is” and *Perinet* assumes no liability for its use. No warranty, either express or implied, is given, including but not limited to, with respect to the accuracy, correctness, reliability and fitness for a particular purpose of the information. This document may be revised by *Perinet* at any time without notice. For the most recent documents, visit https://perinet.io.

Copyright © Perinet GmbH.

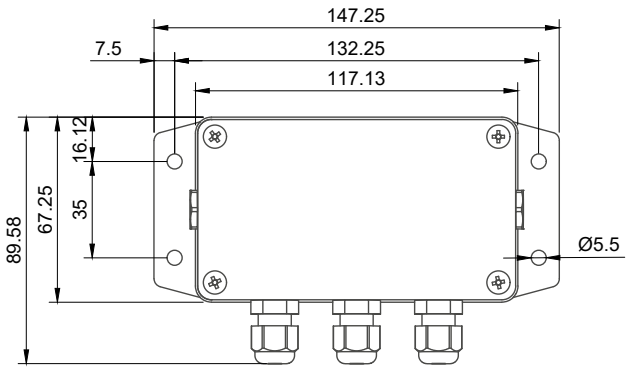
# Contents

# 1 Overview

The *periSNOOP 4-20mA* easily integrates into 4-20mA sensor to PLC loops, digitizing sensor readings without the need for modifying existing installations. This digitization makes sensor data more accessible through Single Pair Ethernet, utilizing the periCORE SPE communication module. Key features of the periSNOOP 4-20mA include state-of-the-art security measures, the ability to update firmware remotely, and TCP/IPv6-based communication capabilities and a RESTful API. Additionally, its hardware is specifically designed for safety, providing protection against overcurrent and similar malfunctions.

## Key Features

- Integrable into existing applications
- No influence on sensor current loop
- Converts 4-20mA analog signals to digital format
- 16-bit ADC resolution
- Supports Daisy Chain topology
- Supports remote firmware updates
- Operates on 24V power supplied via hybrid network cable
- Two 100BASE-T1 Single Pair Ethernet interfaces
- Integrated Software Stacks: TCP/IPv6, mTLS, MQTTs, HTTPs, RESTful API
- End-to-end encryption
- Integrated WebUI & RESTful API
- No-Code Firmware, adoptable to multiple sensors




periSNOOP 4-20mA's dimensions in mm.

## Operational Parameters

- Operating voltage: 24VDC input
- Supply Voltage: 24V output (2A max)
- Temperature range: -40°C to +85°C
- Power consumption: 1.1W
- Accuracy: 0.2 %

## Interfaces

- 2 x 100BASE-T1 Phy (IEEE 802.3bw)
- 1 x 24VDC Power input
- 1 x 24VDC Power output
- 1 x 4-20mA input
- 1 x 4-20mA output
- HTTPs RESTful API, for configuration
- MQTTClient, for data flow
- C++20 library for custom firmware development

## Compliance

- CRA ready
- RoHS 3 (EU 2015/863), WEEE (2012/19/EU), REACH, CE

## Dimensions

**Typical(L/W/H)** 147.25 x 89.58 x 52.02 mm

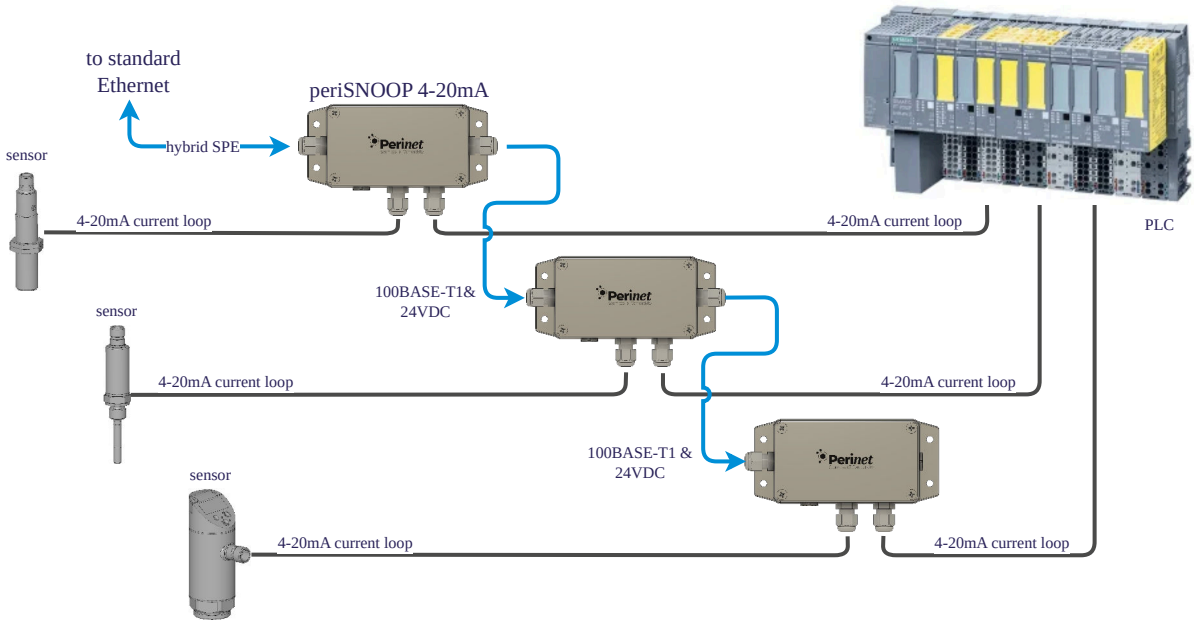
## Security

- NIST compliant TLS implementation
- Role Based Access Control (RBAC)
- Certificate based client authentication
- AES encryption algorithm
- X.509 certificates and PKIX path validation
- Elliptic Curve Cryptography (ECC)

## Software Deliverables

- Integrated HTTP server with TLS
- Mutual TLS (mTLS) based Authentication
- Role Based Access Control (RBAC)
- Integrated MQTT-client, with mTLS
- Zero-configuration (mDNS and DNS-SD)
- Full TCP/IPv6 communication stack
- LifeCycle Management, remote firmware update
- Re-branding support
- No-Code Firmware, compatible with different sensors via configuration changes

# Typical Application




Typical installation of the periSNOOP 4-20mA

# 2 Single Pair Ethernet

Single Pair Ethernet (SPE) describes a single twisted pair of wires that is one aspect of the physical layer for an Ethernet based communication. The periSNOOP 4-20mA uses a point-to-point full-duplex communication scheme with 100Mbits/s throughput on the physical layer as specified by the IEEE [18]. Figure 1 shows conventional Ethernet compared to SPE on a simplified overview.

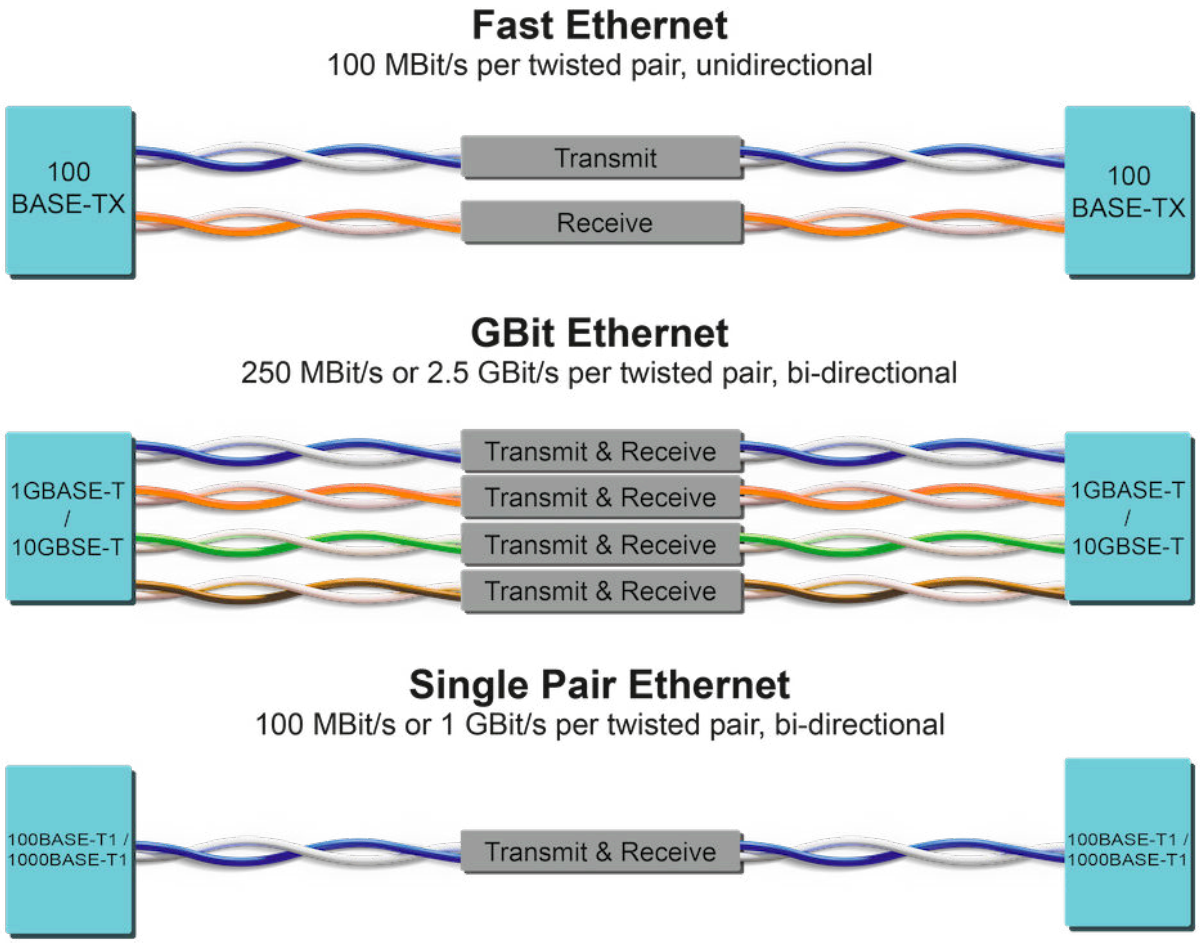



Figure 1: Single Pair Ethernet compared to other Ethernet technologies.

# 3 Architecture

The periSNOOP 4-20mA's structure is depicted in the block diagram shown in Figure 2. It comprises two main sections that are galvanically isolated from each other: the Network side and the Sensor side. This isolation is crucial as it separates the existing setup on the Sensor side from the newly added components on the Network side. The Network side contains the periCORE, which handles all communications over the SPE network and controls the operations of the periSNOOP 4-20mA. On the Sensor side, the device includes an Analog-to-Digital Converter (ADC) and a normally closed relay (N.C.). The device measures the 4-20mA signal current by detecting the voltage across a 100 Ω shunt resistor, with the voltage being directly proportional to the signal current. An isolation block supplies power to the sensor side and enables the transfer of communication and control signals between the two isolated domains. The relay provides a low-resistance path for the signal current, ensuring continued operation in case of power loss or any malfunction in the periSNOOP 4-20mA.

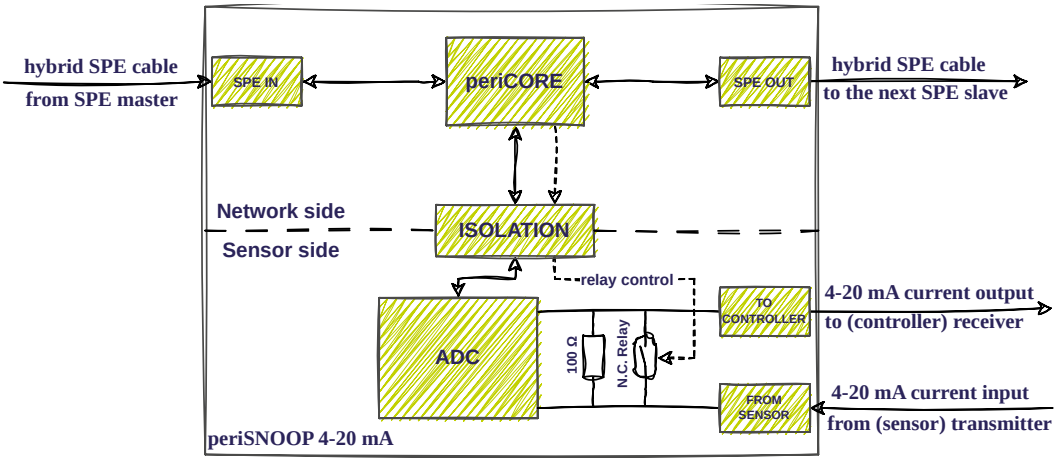



Figure 2: periSNOOP 4-20mA hardware block diagram

periSNOOP 4-20mA blocks description:

`SPE IN` is a 4-pole clamp connector for connecting periSNOOP 4-20mA with the 100BASE-T1 master and the power supply using a hybrid SPE cable. The hybrid SPE cable has 4 wires. 2 wires are used for SPE, and the other 2 wires are for supplying power.

The network interface of the periCORE module associated with this connector implements (100BASE-T1) and is configured in *slave* mode (connects to 100BASE-T1 device with *master* configuration). For further details, see [4].

`SPE OUT` is a 4-pole clamp connector for connecting periSNOOP 4-20mA to a possible remote device on the daisy chain topology with a hybrid SPE cable. This can be used to deliver power and SPE connection to another periSNOOP 4-20mA for example.

The network interface of the periCORE module associated with this connector implements (100BASE-T1) and is configured in *master* mode (connects to 100BASE-T1 device with *slave* configuration). For further details, see [4].

`FROM SENSOR` is a 4-pole clamp connector to connect the signal cable from the sensor (4-20mA transmitter) to periSNOOP 4-20mA. One pole is for 4-20mA signal and the other 3 are fed to the TO CONTROLLER connector through. The must flows into periSNOOP 4-20mA at this point.

`TO CONTROLLER` is a 4-pole clamp connector to connect periSNOOP 4-20mA to the existing controller (4-20mA receiver). One pole is for 4-20mA signal and the other 3 are only fed through. The current must flow out of periSNOOP 4-20mA at this point.

`periCORE` is the Single Pair Ethernet communication module. It implements the SPE communication interfaces and provides a µController, which implements the software stack for communication, security as well as functionality. For further details, see [4].

`ADC` is a 16-bit Analog-to-Digital Converter.

## 3.1 Typical Application

periSNOOP 4-20mA finds its place in the existing control loops based on 4-20mA current signals. Figure 3 shows a typical periSNOOP 4-20mA application.

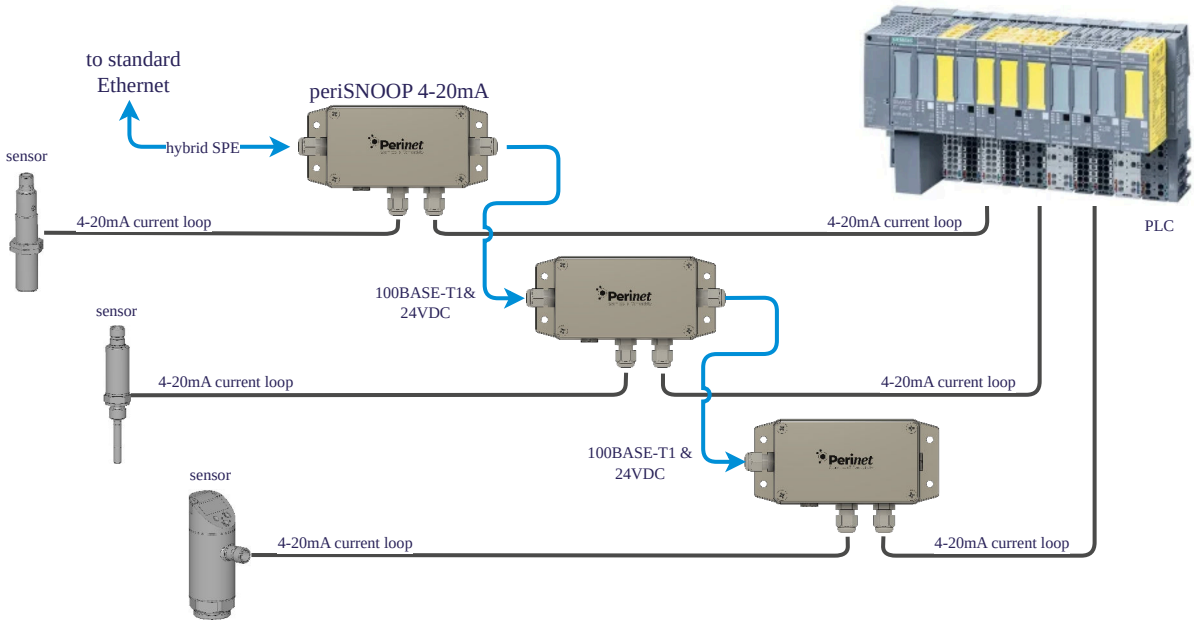



Figure 3: Peripheral connectivity for the periSNOOP 4-20mA with daisy chain network topology.

In the setup depicted in Figure 3, a control loop consists of a Programmable Logic Controller (PLC) and three sensors generating 4-20mA current signals. The periSNOOP 4-20mA, positioned between the sensor and the PLC, digitizes these current signal values, making them accessible over the network without needing any changes to the existing installation.

Further details about the wiring process are available in Section 6.

## 3.2 Theory of Operation

Upon powering on or restarting, the periSNOOP 4-20mA begins with an initialization phase. Initially, the 4-20mA current signal passes through the closed relay. Once initialization completes, the relay opens, redirecting the current signal through a shunt resistor. The voltage drop across this resistor is measured by a 16-bit ADC, transforming it into digital data. This data is then processed and made available on the network. In case of errors, like overcurrent, the relay closes, disconnecting the shunt resistor from the current loop. Manual intervention (possible through the network) is required to reset the relay.

## 3.3 Typical Network Topology

The periSNOOP 4-20mA is designed for field implementation using SPE (Single Pair Ethernet), featuring a multiport switch compatible with 100BASE-T1. This allows for the support of daisy chain network topology. As shown in Figure 3, the periSNOOP 4-20mA is deployed within a daisy chain network configuration.

**Note:** The 100BASE-T1 physical layer establishes a link between master and slave as per [18]. This means, the periSNOOP 4-20mA must be connected from `SPE_out` to `SPE_in`.

# 4 Electrical Connectivity

## 4.1 Signal Types

| Type Name | Description |
|---|---|
| DC_PWR | Direct current supply |
| DC_GND | Common Ground supply |
| SPE_DATA | Digital differential communication |
| AnalogSense | 4-20mA current signal |
| AnalogMisc | General purpose, fed through analog signal |

Table 1: Signal Type Definitions

## 4.2 Connectors

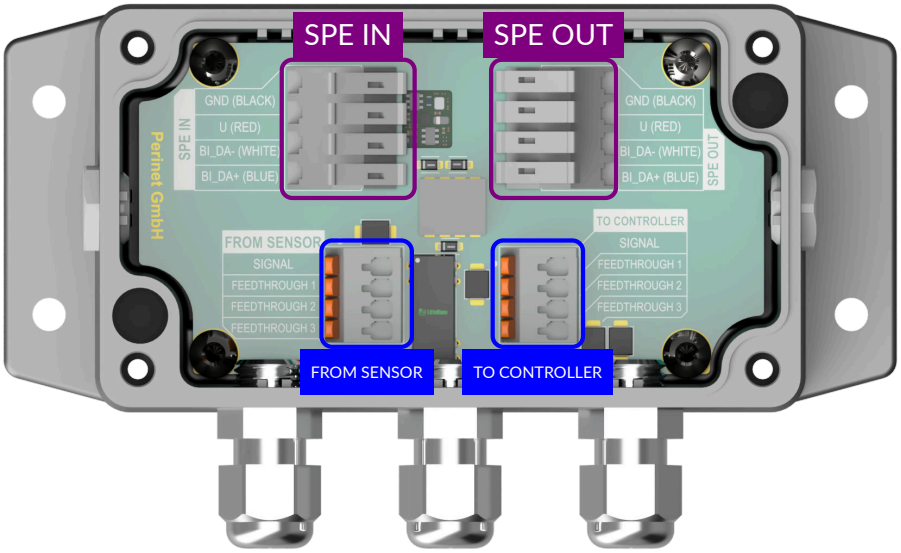



Figure 4: periSNOOP 4-20mA's electrical connectivity.

| Pin | Sig | Type | Descr. |
|---|---|---|---|
| 1 | GND | DC_GND | Power supply (0V) direct current input |
| 2 | U | DC_PWR | Power supply (24V) direct current input |
| 3 | BI_DA- | SPE_DATA | white, 100BASE-T1 slave mode [18] data pair - |
| 4 | BI_DA+ | SPE_DATA | blue, 100BASE-T1 slave mode [18] data pair + |

Table 2: periSNOOP 4-20mA's electrical connectivity for connector SPE IN.

| Pin | Sig | Type | Descr. |
|---|---|---|---|
| 1 | GND | DC_GND | Power supply (0V) direct current output |
| 2 | 24VDC | DC_PWR | Power supply (24V) direct current output |
| 3 | BI_DA- | SPE_DATA | white, 100BASE-T1 master mode [18] data pair - |
| 4 | BI_DA+ | SPE_DATA | blue, 100BASE-T1 master mode [18] data pair + |

Table 3: periSNOOP 4-20mA's electrical connectivity for connector SPE OUT.

| Pin | Sig | Type | Descr. |
|---|---|---|---|
| 1 | SIGNAL | AnalogSense | 4-20mA current signal input from the sensor. The current must flow into periSNOOP 4-20mA at this point |
| 2 | FEEDTHROUGH 1 | AnalogMisc | Analog input/output, short circuit to FEEDTHROUGH 1 of the TO CONTROLLER connector |
| 3 | FEEDTHROUGH 2 | AnalogMisc | Analog input/output, short circuit to FEEDTHROUGH 2 of the TO CONTROLLER connector |
| 4 | FEEDTHROUGH 3 | AnalogMisc | Analog input/output, short circuit to FEEDTHROUGH 3 of the TO CONTROLLER connector |

Table 4: periSNOOP 4-20mA's electrical connectivity for connector FROM SENSOR.

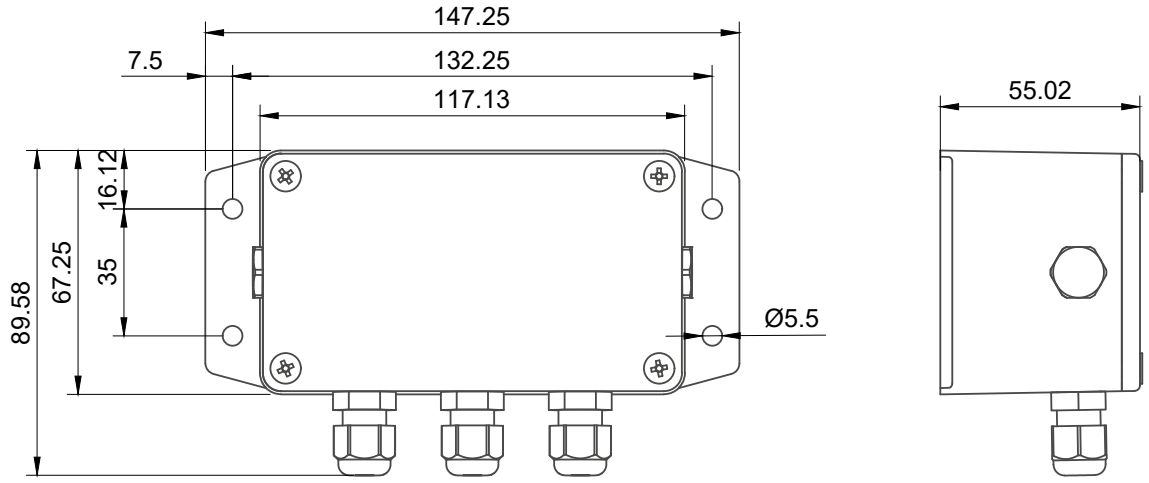
| Pin | Sig | Type | Descr. |
|---|---|---|---|
| 1 | SIGNAL | AnalogSense | 4-20mA current signal output to the controller. The current must flow out of periSNOOP 4-20mA at this point |
| 2 | FEEDTHROUGH 1 | AnalogMisc | Analog input/output, short circuit to FEEDTHROUGH 1 of the FROM SENSOR connector |
| 3 | FEEDTHROUGH 2 | AnalogMisc | Analog input/output, short circuit to FEEDTHROUGH 2 of the FROM SENSOR connector |
| 4 | FEEDTHROUGH 3 | AnalogMisc | Analog input/output, short circuit to FEEDTHROUGH 3 of the FROM SENSOR connector |

Table 5: periSNOOP 4-20mA's electrical connectivity for connector TO CONTROLLER.

# 5 Specifications

## 5.1 Mechanical Specifications

periSNOOP 4-20mA physical dimensions are shown in Figure 5.




(a) Top view (b) Side view

Figure 5: periSNOOP 4-20mA mechanical dimensions drawing.

periSNOOP 4-20mA handling involves tightening of 5 different types of screws: lid screws, cable gland body, cable gland sealing nut and blind gland. For achieving the optimal mechanical connections those screws need to be fastened using a specific torque. The specific torque is given in Table 6.

| Screw type | Torque | Unit |
|---|---|---|
| Lid screw (M 4) | 0.85 | Nm |
| Cable gland body (M 12) | 8.5 | Nm |
| Cable gland nut (M 12) | 8.5 | Nm |
| Blind cable gland (M 12) | 2 | Nm |

Table 6: Torque for screws used in periSNOOP 4-20mA

More information about the installation and wiring of periSNOOP 4-20mA can be found in Section 6.

## 5.2 Environmental Specifications

| Parameter | Min | Max | Unit |
|---|---|---|---|
| Operating temperature | -40 | +70 | °C |
| Storage temperature | -40 | +85 | °C |
| Relative storage humidity (non-cond.) | 5 | 90 | % |
| Protection degree | | IP65/IP67 | |

Table 7: Environmental specification

## 5.3 Electrical specifications

### 5.3.1 Absolute Maximum Ratings

| Parameter | Min | Max | Unit |
|---|---|---|---|
| DC_PWR | 0 | 27 | V |
| DC_GND | 0 | 0 | V |
| SPE_DATA | -30 | 30 | V |
| AnalogSense | -1.5 | 1.5 | A |
| AnalogSense | -36 | 36 | V |
| AnalogMisc | -2 | 2 | A |
| AnalogMisc | -36 | 36 | V |

Table 8: Absolute maximum ratings of the periSNOOP 4-20mA

**Warning:** Exceeding the specified absolute maximum ratings may damage the periSNOOP 4-20mA.

### 5.3.2 Common specifications

| Parameter | Condition | Min | Typ. | Max | Unit |
|---|---|---|---|---|---|
| Supply voltage (U) | | 21.6 | 24 | 26.4 | V |
| Power consumption | Supply voltage U = 24V | - | 1.1 | - | W |
| Isolation voltage | 1 minute per UL 1577 | - | 2500 | - | V rms |

Table 9: Operational conditions

### 5.3.3 Input specifications

| Parameter | Condition | Min | Typ. | Max | Unit |
|---|---|---|---|---|---|
| Shunt resistor | Nominal operation, relay opened | - | 100 | - | Ω |
| Relay resistance | Initial value | - | - | 0.15 | Ω |
| Current measuring range | | 0 | - | 25 | mA |
| Surge protection | EN 61000-4-5 | - | - | 1000 | V |

Table 10: Input specifications

### 5.3.4 Output specifications

| Parameter | Condition | Min | Typ. | Max | Unit |
|---|---|---|---|---|---|
| Conversion resolution | | - | 16 | - | bit |
| Conversion accuracy | | - | 0.2 | - | % |
| Conversion rate | | - | - | 0.1 | s |

Table 11: Output specifications

# 6 Installation Instructions

For the installation process, the following periSNOOP 4-20mA parts are relevant (Figure 6 and Figure 7):

1. Lid
2. Lid screws
3. Cable glands
4. Blind cable glands
5. Mounting holes (also used as terminals for the protective earth (P.E.) connection)
6. `SPE IN` and `SPE OUT` connectors
7. `FROM SENSOR` and `TO CONTROLLER` connectors

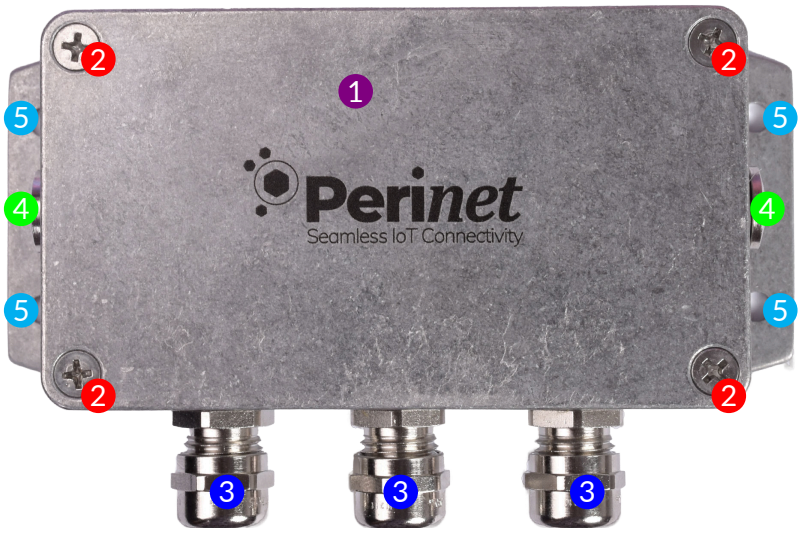



Figure 6: Installation parts of periSNOOP 4-20mA

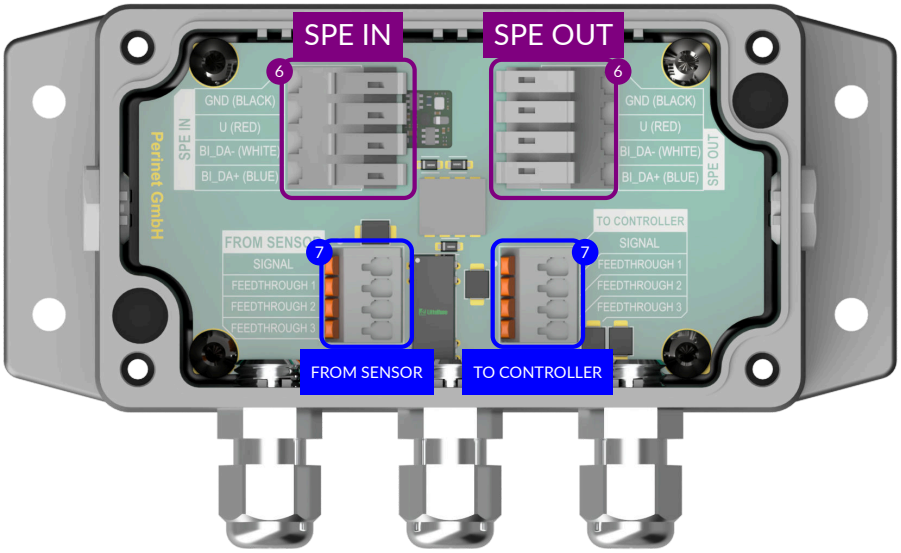



Figure 7: periSNOOP 4-20mA connectors.

The periSNOOP 4-20mA installation process consists of the following steps:

1. Opening the lid
2. Wiring
3. Closing the lid
4. Mounting and earthing

For the installation, the following tools are needed:

- Screw driver, PH-2
- Torque wrench size 14

## 6.1 Lid Opening/Closing

To open the lid of the periSNOOP 4-20mA, remove the four screws as illustrated in Figure 6. Inside the lid, there's a groove designed for a rubber seal. This seal helps to prevent fluids from entering the device's interior. When reattaching the lid, ensure the screws are tightened to a torque of 0.85 Nm, in accordance with the specifications detailed in Section 5.1.

## 6.2 Wiring

Wiring the periSNOOP 4-20mA involves two key areas:

- On the Network side, there are two wiring options: one with Daisy Chain and another without it.
- On the Sensor side, it supports various 4-20mA current loop configurations, including 2-wire, 3-wire, and 4-wire topologies.

### 6.2.1 Network Side Wiring - With Daisy Chain

To set up a Daisy Chain wiring configuration, two hybrid SPE cables are required. This setup necessitates an extra cable gland, as depicted in Figure 8. You'll need to remove the left cable gland located on the broader side of the housing and seal the opening with a blind cable gland.

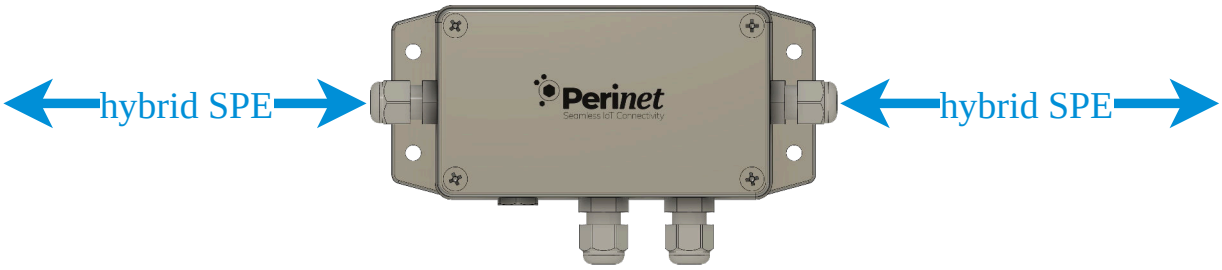



Figure 8: Wiring scheme with Daisy Chain - cable position

The hybrid SPE cables are connected to the `SPE IN` and `SPE OUT` ports on the periSNOOP 4-20mA. Use the `SPE IN` port to attach the cable coming from the 100BASE-T1 master device. The `SPE OUT` port is for the cable leading to the 100BASE-T1 slave device. The wires in the hybrid SPE cable are color-coded as follows:

- Black - Power supply negative terminal
- Red - Power supply positive terminal
- White - [18] data pair negative
- Blue - [18] data pair positive

The wiring is shown below.

### 6.2.2 Network side wiring - without Daisy Chain

In this wiring scheme only one hybrid SPE cable enters the periSNOOP 4-20mA housing. periSNOOP 4-20mA comes with the cable glands already installed to support this use case, which is shown in Figure 9.

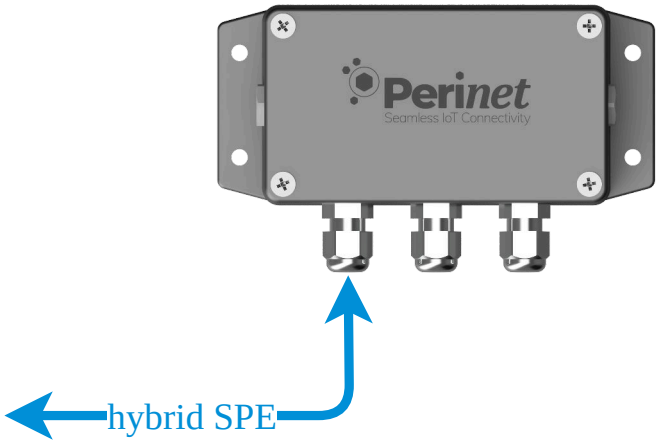



Figure 9: Wiring scheme without Daisy Chain - cable position

The hybrid SPE cable is connected to the `SPE IN` connector. The `SPE IN` connector is used to connect the hybrid SPE cable from the 100BASE-T1 master. The wire colors of the hybrid SPE cable are defined in the following way:

- Black - Power supply negative terminal
- Red - Power supply positive terminal
- White - [18] data pair -
- Blue - [18] data pair +

The wiring is shown in Figure 10.

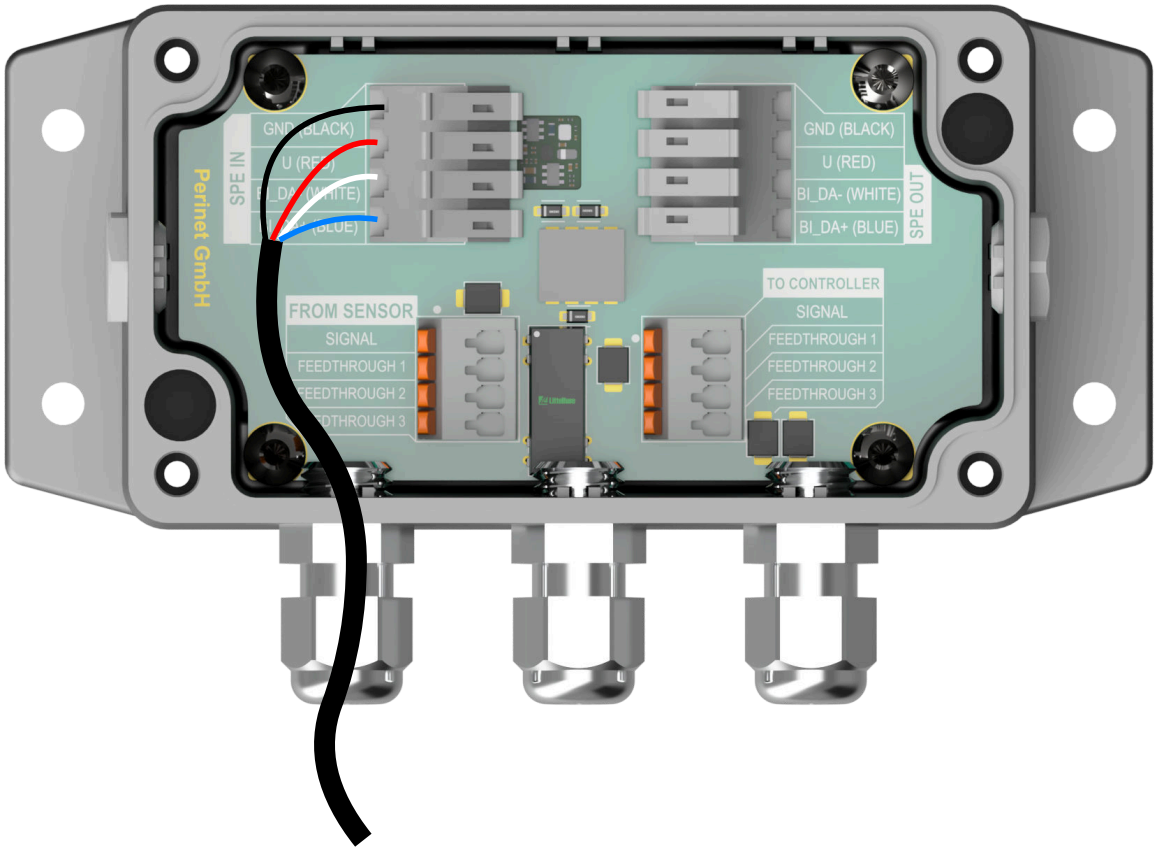



Figure 10: Line topology wiring

### 6.2.3 Sensor Wiring

periSNOOP 4-20mA supports 4-20mA sensors with 2, 3, and 4 wires. The wiring procedure is identical for all three cases: the existing sensor cable is first cut, and the both newly produced cable ends should be put through the cable glands into the periSNOOP 4-20mA housing. It is important to know the direction of the signal current before clamping the wires into the `FROM SENSOR` and `TO CONTROLLER` connectors. The names of the connectors imply that: the current flows from the sensor into the `FROM SENSOR` connector, and from the `TO CONTROLLER` connector out. Both `FROM SENSOR` and `TO CONTROLLER` sensors have 4 clamps. Only the clamp named `SIGNAL` is relevant for the current signal, the other clamps are only used as feedthroughs. The examples of 2-, 3-, and 4-wire sensor wiring are shown in Figure 11, Figure 12, and Figure 13, respectively.

### 6.2.4 Sensor Wiring

The periSNOOP 4-20mA is compatible with 4-20mA sensors using 2, 3, or 4 wires. To wire these sensors correctly, follow these steps:

1. Cut the existing sensor cable.
2. Pass the two resulting cable ends through the cable glands into the housing of the periSNOOP 4-20mA.

It's crucial to understand the direction of the signal current before attaching the wires to the connectors. The connectors are named based on the current flow:

- `FROM SENSOR`: Connect the cable coming from the sensor here.
- `TO CONTROLLER`: Connect the cable going to the controller here.

Each of these connectors, `FROM SENSOR` and `TO CONTROLLER`, has four clamps. For the current signal, only the clamp labeled `SIGNAL` is used; the other clamps serve merely as pass-throughs. Visual examples for wiring 2-wire, 3-wire, and 4-wire sensors can be found in Figure 11, Figure 12, and Figure 13, respectively.

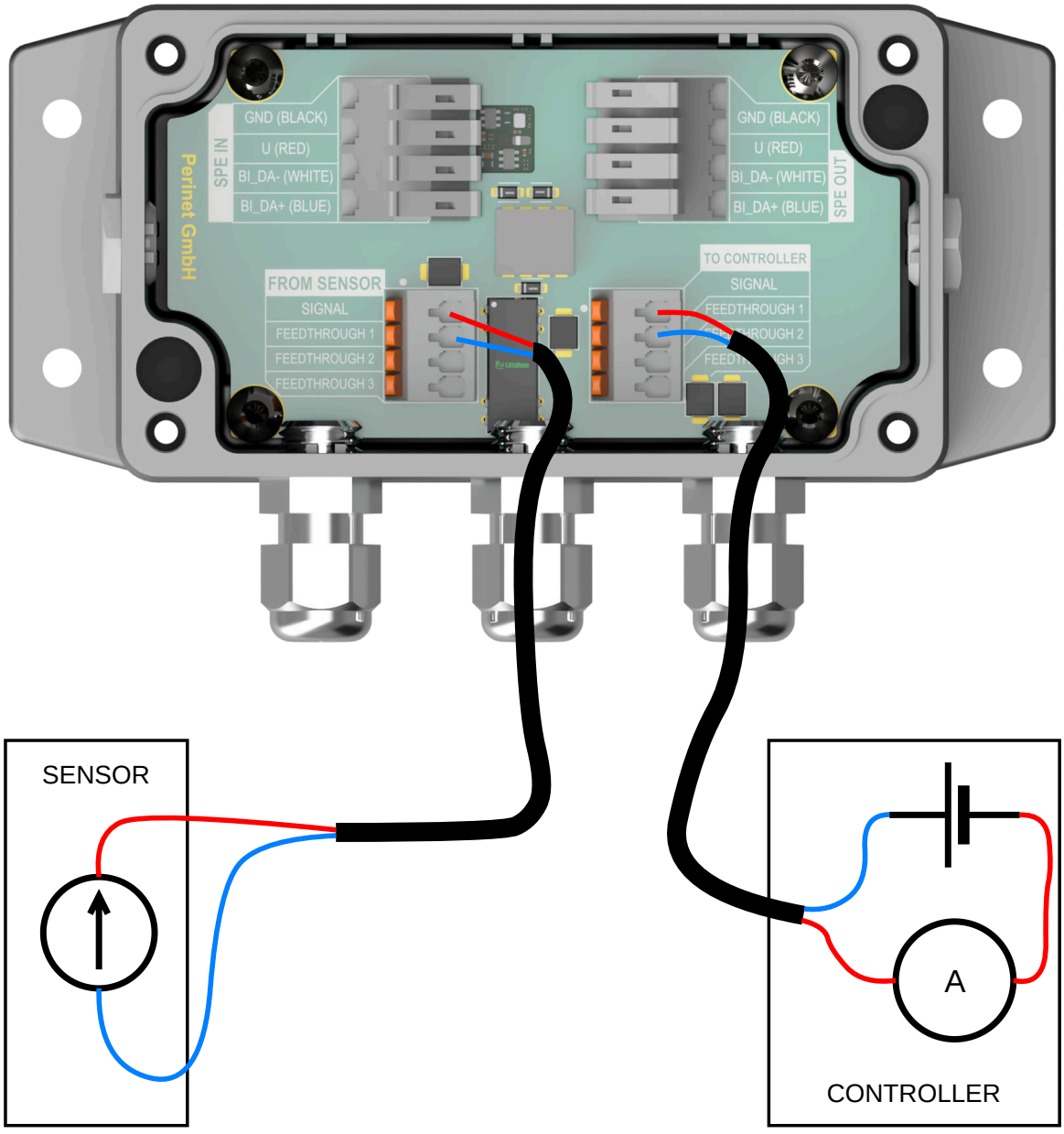



Figure 11: 2-wire sensor wiring

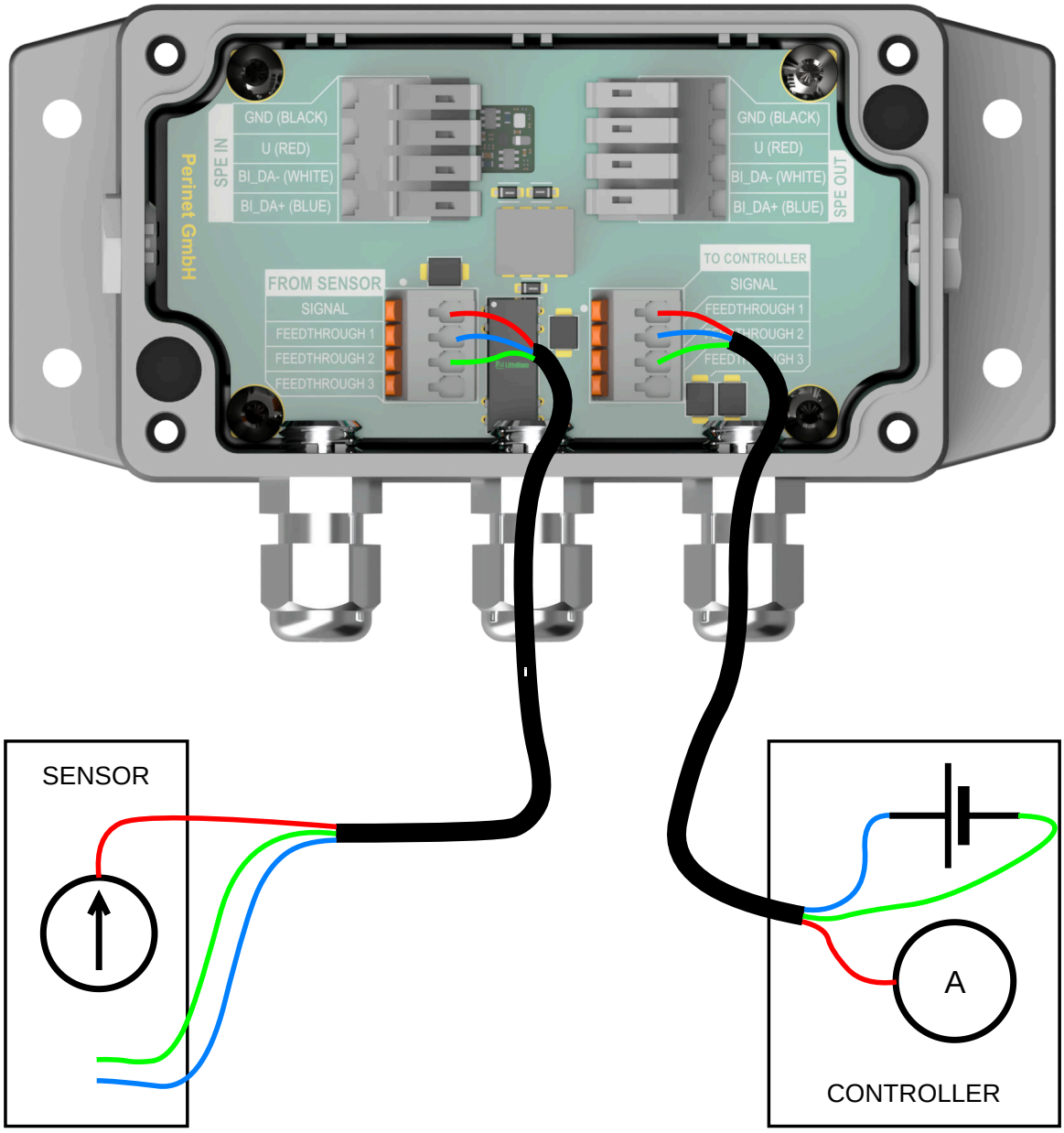



Figure 12: 3-wire sensor wiring

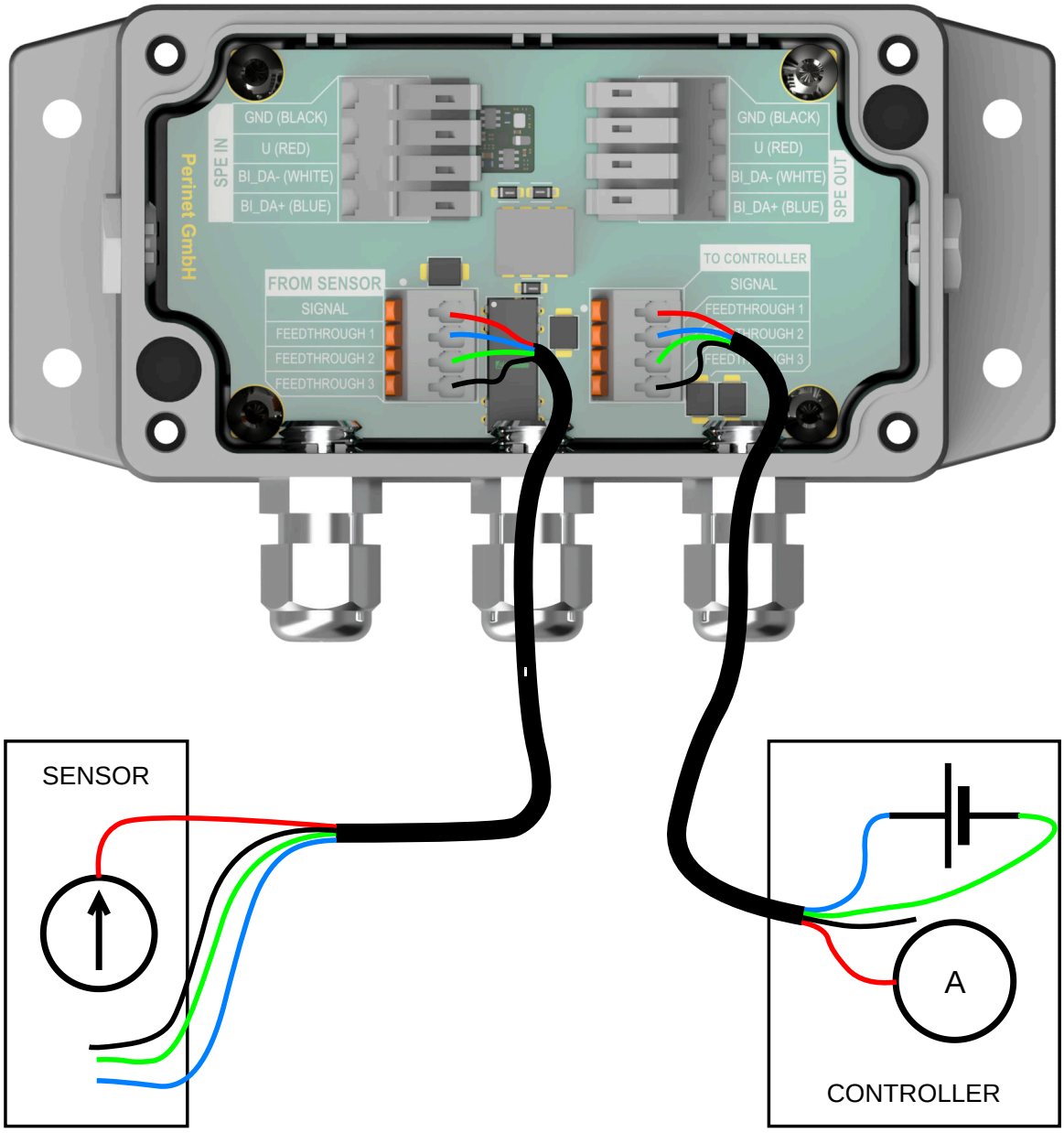



Figure 13: 4-wire sensor wiring

## 6.3 Mounting and Earthing

The periSNOOP 4-20mA features four holes specifically designed for mounting. When installing the device, it is important to use one of these mounting screws to connect the protective earth (P.E.) conductor. This ensures proper earthing of the periSNOOP 4-20mA for safety and functionality.

# 7 Factory Reset

A periCORE based product can be reset to factory settings using two different methods. One method is through the RESTful API, as detailed in **??**. The second method is a physical reset, which is necessary if RESTful API access is lost, such as in cases where the admin mTLS certificate is unavailable. The foundational *libperiCORE* library facilitates the factory reset process. To physically initiate a factory reset, ensure that the periCORE based product is powered on (using `24VDC_IN`) but is not connected to any network through its network ports (Port 0, Port 1, and Port 4). The factory reset will automatically occur 20 seconds after the device is powered up, returning the device to its original factory settings. For more details on the Factory Reset process, refer to **??**.

# 8 Integrated WebUI

periSNOOP smart components offer two main interfaces: a web-based user interface for humans and a RESTful API for machines. These interfaces are available through an integrated HTTP server, which requires TLS for secure access.

**Note:** Access to the integrated HTTP server is only possible with TLS enabled.

Each periSNOOP smart component has a unique hostname, formed by combining the host prefix `periSNOOP` with its unique serial number. Example: `periSNOOP-serno`, where `serno` is the serial number. This unique identifier is also present on the product's label (refer to **??** for details).

By default, periSNOOP use the Zeroconf protocol multicast DNS (mDNS) and Link Local Multicast Name Resolution (LLMNR) to enable easy network discovery.

To access the web-based user interface, you can use mDNS-based name resolution. To access a periSNOOP device, you would use its mDNS name as follows:

```
https://periSNOOP-serno.local/
```

or via an LLMNR based name resolution:

```
https://periSNOOP-serno/
```

A RESTful API specification is available on each periSNOOP via the URI:

```
https://periNODE-serno.local/doc/api.proto
```

## 8.1 Authenticity and Security Warnings

Users may encounter a security warning in environments where the **Perinet ECC Root CA** trust anchor is not installed. This certificate is essential for verifying periSNOOP Smart Components. It confirms that the component is genuinely made by an authorized manufacturer.

However, the **Perinet ECC Root CA** certificate is not automatically added to systems and requires manual installation. If this certificate is not present in the system, the authenticity of the Smart Component cannot be verified.

This situation occurs when a web client, like a standard web browser, attempts to establish a secure connection with the Smart Component. The web client first tries to authenticate the device. If the **Perinet ECC Root CA** certificate is missing, this authentication will fail, resulting in security warnings, as illustrated in Figures 14 (Edge browser) and 15 (Safari browser).

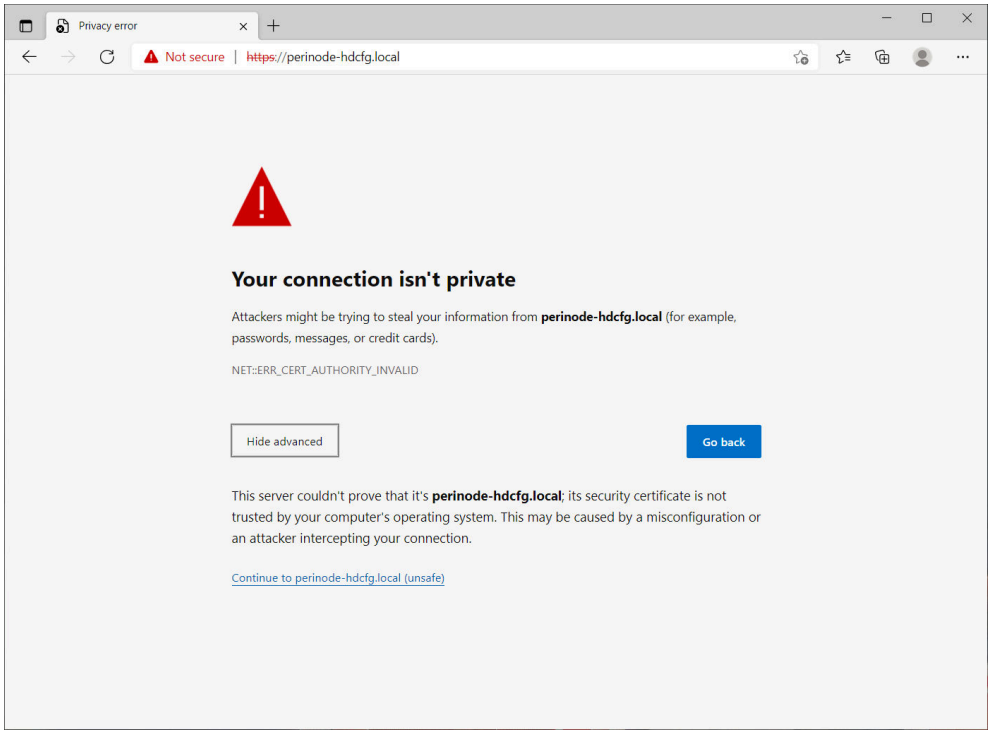



Figure 14: Edge browser security warning example

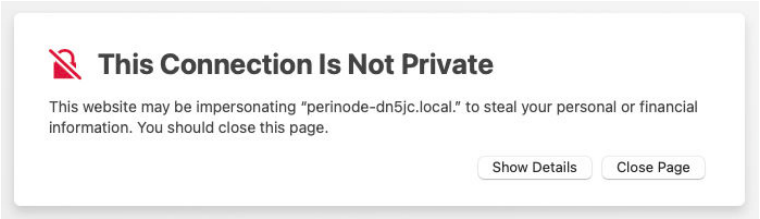



Figure 15: Safari browser security warning example

To address this issue, there are two options: either install the **Perinet ECC Root CA** trust anchor on the system, or choose to ignore the warning. However, bypassing the warning compromises system security.

**Note:** *Perinet* strongly advises installing the **Perinet ECC Root CA** in the system's list of trusted entities, instead of disregarding the security warning.

### 8.1.1 Add Perinet as a Trust Anchor

To install the **Perinet ECC Root CA** certificate, download it from https://docs.perinet.io. The installation process is straightforward: double-click the downloaded file and follow the on-screen instructions.

For comprehensive instructions on installing certificates across various operating systems and browsers, refer to the *Security Certificates Installation Guide* [16]. This guide assists in installing the necessary certificates for all *Perinet* Smart Components.

Once the **Perinet ECC Root CA** trust anchor is installed, you should be able to access the web user interface of any smart component without encountering security warnings. If issues persist, please reach out to our support team.

You can verify the installation by clicking on the *locker* icon in your web browser's URL bar. The subsequent figures demonstrate the expected certificate chain for reference.

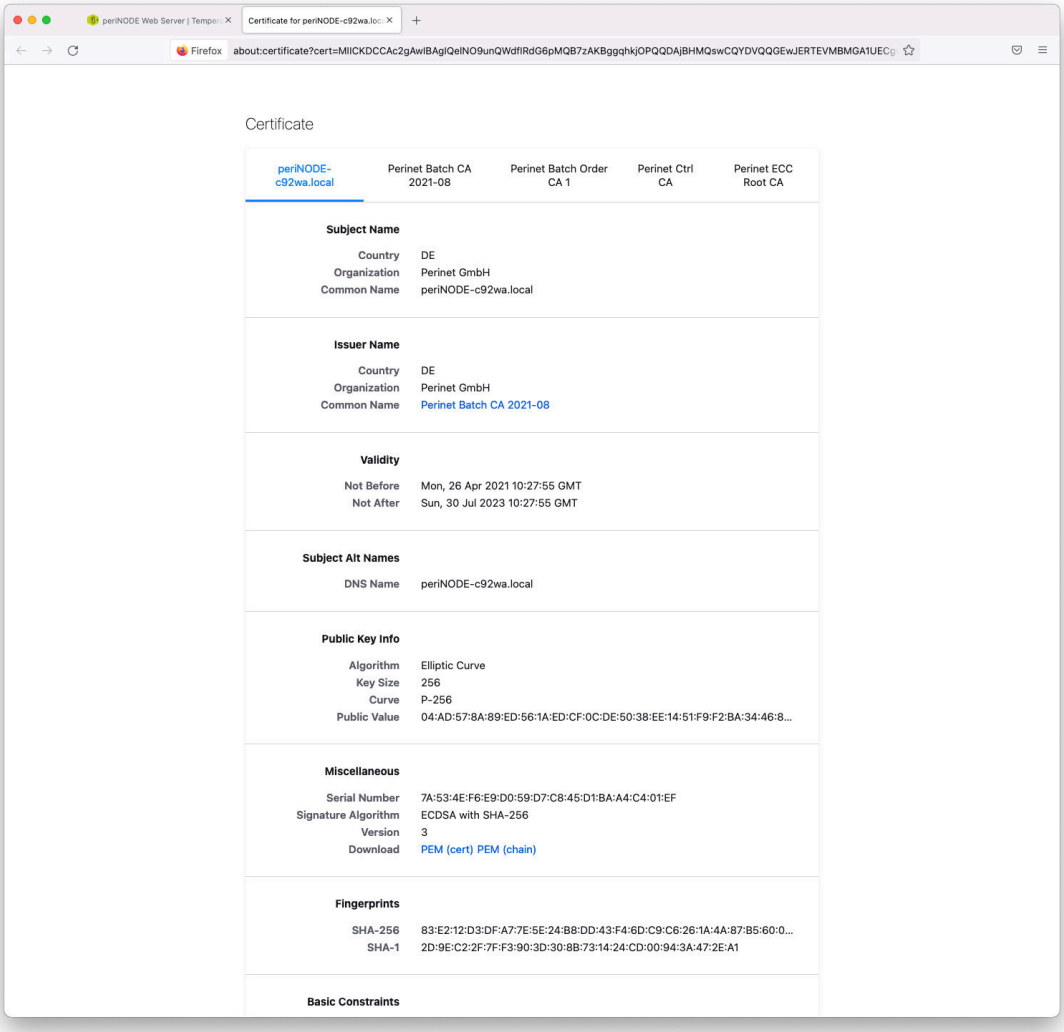



Figure 16: Firefox example security trusted chain

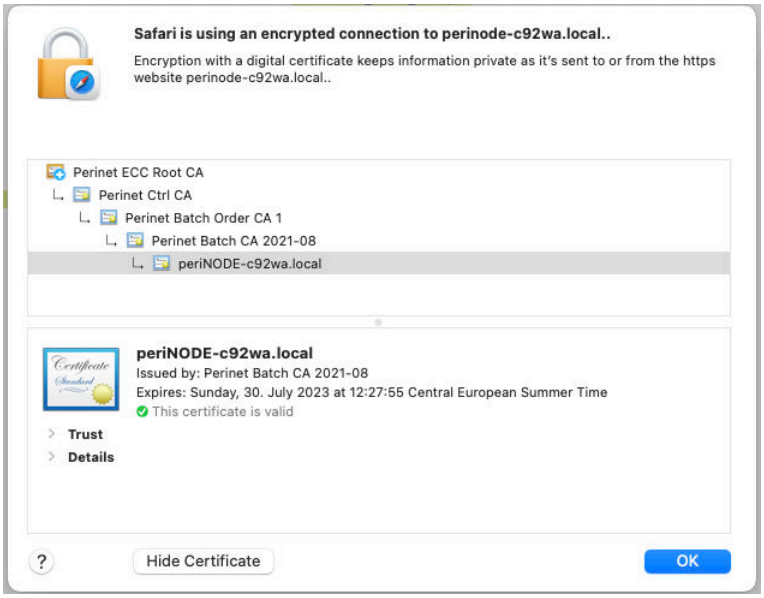



Figure 17: Safari example security trusted chain

**Note:** When using the RESTful API to access the Smart Component, the absence of the **Perinet ECC Root CA** trust anchor may impact command line tools. In this scenario, it's necessary to install the **Perinet ECC Root CA** trust anchor system-wide, rather than only in the web client (standard browser).

### 8.1.2 Proceed with Exception

As mentioned earlier, it's possible to bypass the security warnings if you choose not to add *Perinet* as a trusted entity. To do this, click on "Advanced" and confirm your intention to proceed. This action allows encrypted but unauthenticated communication.

When using the RESTful API to access a Smart Component, the absence of the **Perinet ECC Root CA** trust anchor can affect command line tools. However, there are ways to circumvent this. For instance, when using the **curl** tool, the `-k`, `--insecure` option allows proceeding without server authentication.

**Note:** *Perinet* strongly advises against ignoring security warnings. Without the certificate, the system is vulnerable to certain attacks, such as man-in-the-middle attacks.

## 8.2 Configuration

The firmware of periSNOOP offers two configuration methods: a machine-friendly RESTful API and an HTML-based graphical user interface (GUI). The GUI is designed for interactive use, while the RESTful API is better suited for automated configurations.

All configuration settings are saved persistently and will be reinstated during the startup process.

### 8.2.1 periSNOOP Home

The periSNOOP Smart Component features a user-friendly interface for monitoring sensor measurements. The *Home screen* serves as the main page of the web user interface. An example is illustrated in Figure 18. To access the *Home screen*, use the following URL format: `https://periSNOOP-serno.local/`.

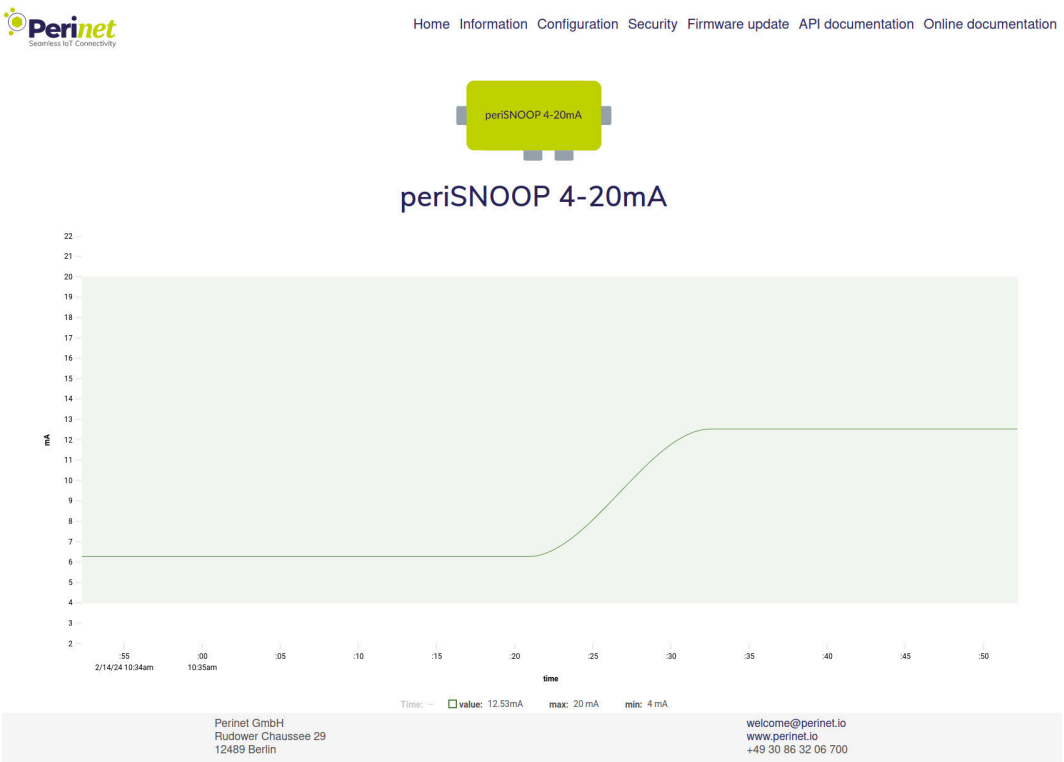



Figure 18: The Web Home Page of periSNOOP

### 8.2.2 periSNOOP GUI Configuration

The GUI (Graphical User Interface) of the periSNOOP enables configuration of several parameters. It is split into several sections:

### Node


Figure 19: periSNOOP Node section of Configuration Web Page

The Node section consists of two parameters:

- *Application name*: The (***application_name***) is used to group IoT devices and software components, that together represent an application. It also sets the prefix for MQTT topics.
- *Element name*: The (***element_name***) allows you to label things within applications understandably, getting rid of clumsy structural labeling like IP-addresses or hostnames.

### Peripheral

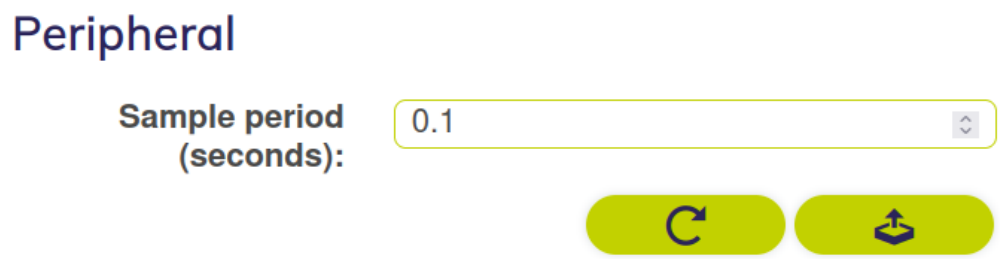



Figure 20: periSNOOP Peripheral section of Configuration Web Page

The Peripheral section groups parameters of the sensor side. It consists of the following parameter:

- *Sample period (seconds)*: The (***period_seconds***) setting specifies the interval, in seconds, at which the system samples data. This parameter is crucial for controlling the data collection frequency, allowing for adjustments based on the application's performance and data requirements.

### Transformation

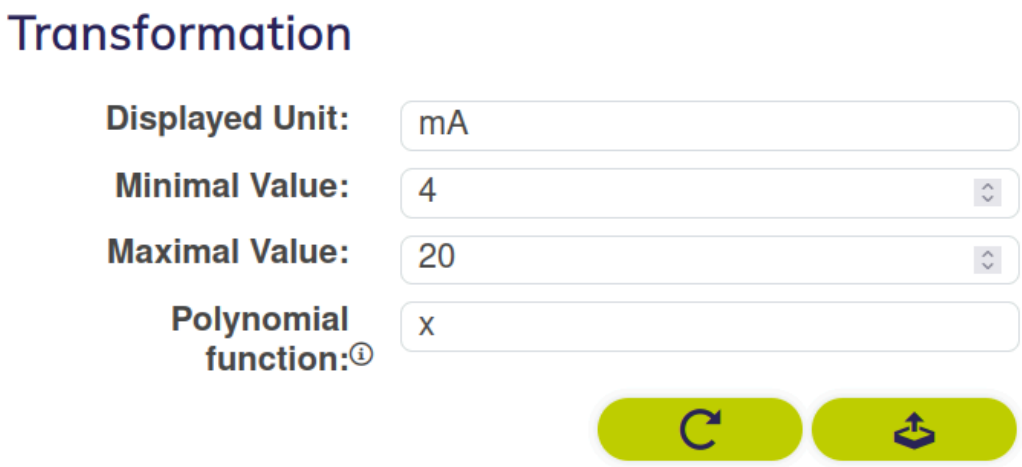



Figure 21: periSNOOP Transformation section of Configuration Web Page

The Transformation section consists of parameters necessary for the polynomial transformation of the sensor data. For details see PRN100764 - periCORE Telemetry Data Ingress and Egress Application Note [**perinet:periCORE:ingressNegress:applnote**].

- *Displayed Unit*: The unit of the transformed sensor data.
- *Minimal Value*: Lowest value of valid transformed sensor data.
- *Maximal Value*: Highest value of valid transformed sensor data.
- *Polynomial function*: The transformation formula.

### MQTT Client

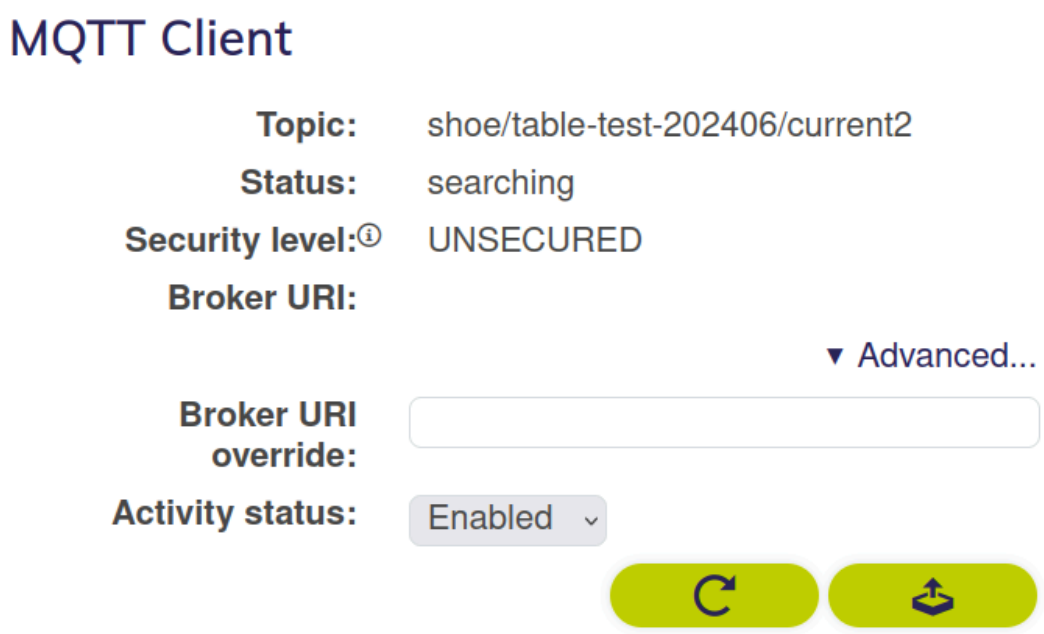



Figure 22: periSNOOP MQTT Client section of Configuration Web Page

The MQTT Client section includes information about the MQTT client. It consists of the following parameters:

- *Topic*: The MQTT topic used for transmitting the transformed sensor data. It is a combination of *Application name* and *Element name* of section Node (8.2.2).
- *Status*: The status of the connection to the MQTT broker.
- *Security level*: The security level at which the MQTTs Client (with TLS, Transport Layer Security) is connected:
  - UNSECURED (Level-0): No encryption or authentication of the remote entity for the connection is active. Note: This level will never be reached on an active connection.
  - ENCRYPTED (Level-1): TLS has been activated, but no verification of the authenticity of the client is done. For example, it allows connection to an MQTTs broker without verifying the server host certificate.
  - AUTHENTICATED (Level-2): TLS has been activated with the verification of the authenticity of the remote entity. For example, it refuses connection to an MQTTs broker where the verification of the server host certificate has failed.
  - AUTHORIZED (Level-3): mTLS (mutual TLS), which is the same as AUTHENTICATED but also authenticates a client by sending the client TLS certificate.
- *Broker URI*: the MQTT broker the periSNOOP is associated with (The periSNOOP auto-connects to an MQTT broker configured with the same application name).

- *Broker URI override*: Enter the Broker URI manually here when using another broker than the autoconfigured one.
- *Activity status*: Enable/Disable the MQTT client.

### HTTP Server

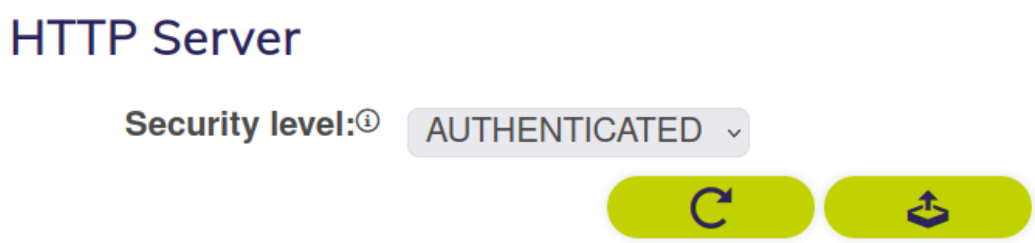



Figure 23: periSNOOP HTTP Server section of Configuration Web Page

The HTTP Server section stores parameter belonging to periSNOOPs HTTP Server. It consists of the following parameter:

- *Security level*: The security level at which the HTTP Server (with TLS, Transport Layer Security) is accepting connections:
  - UNSECURED (Level-0): No encryption or authentication of the connection is active. Note: This level will never be reached on an active connection.
  - ENCRYPTED (Level-1): TLS has been activated, but no verification of the authenticity of the client is done. For example, a HTTPs server sends a fallback certificate which cannot be used to authenticate the HTTPs server.
  - AUTHENTICATED (Level-2): TLS has been activated and the client can authenticate the server. For example, the HTTPs server sends a valid certificate that authenticates the server.
  - AUTHORIZED (Level-3): includes AUTHENTICATED (Level-2) and enables mTLS (mutual TLS), which authenticates the client and verifies its access (RBAC role based access control). The roles are USER, SUPER and ADMIN.

### Default Settings

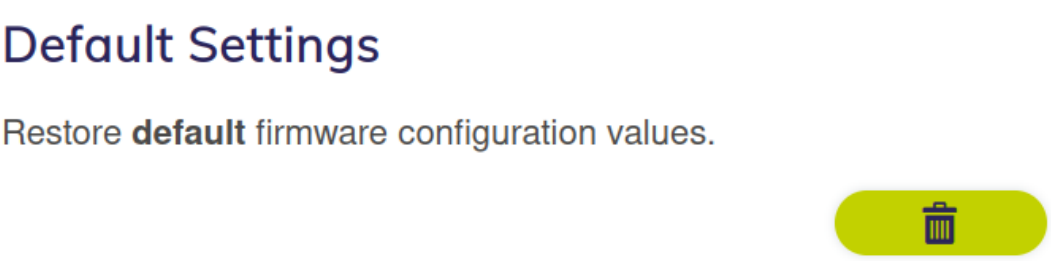



Figure 24: periSNOOP Default Settings section of Configuration Web Page

By clicking the wastebasket button all customized parameters are overwritten by its default values.

### Reboot Firmware

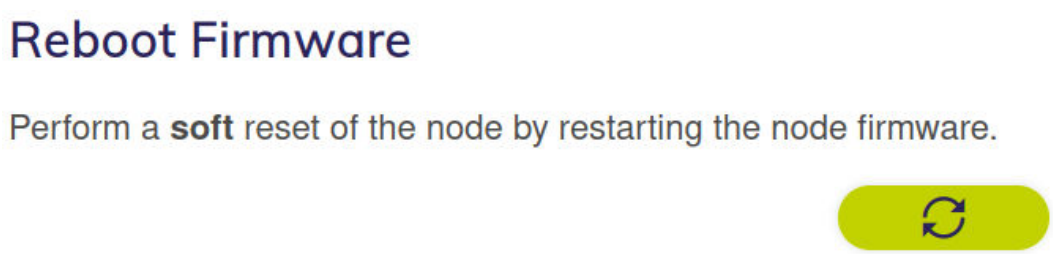



Figure 25: periSNOOP Reboot Firmware section of Configuration Web Page

By clicking the Reboot Firmware button a manual reboot of the periSNOOP is performed.

### 8.2.3 Security Configuration

Security settings for a periNODE can be modified using the RESTful API at `/security` or through a web interface accessible at *https://<hostname>.local/security.html*.

There are three certificates for configuring a periNODE's security:

- ***Host Certificate***: A unique certificate that validates the periNODE's identity to external clients like web browsers. It includes the hostname and must be signed by a trusted root CA, ensuring the periNODE's authenticity.
- ***Root Certificate***: Represents the trusted authorities in the periNODE system. It is essential for verifying external clients (e.g. web browsers) during connections, especially when mutual TLS (mTLS) is enabled.
- ***Client Certificate***: Utilized by the periNODE when connecting as a client to external servers, like MQTT brokers. This certificate authenticates the periNODE **and defines its access level in these interactions, under the Role Based Access Control (RBAC) system**.

≥ FW 22

For more detailed information on security concepts used for Perinet products, please refer to https://docs.perinet.io.

#### Mutual TLS and Authorization

A periNODE can be set up to use mutual TLS (mTLS) and Role Based Access Control (RBAC). With mTLS enabled, any remote client must authenticate itself using a client certificate to establish a connection. This certificate is checked against the periNODE's *Root Certificate*.

**Note:** Remote clients can be authenticated by the periNODE only if the *Root Certificate* is stored in the periNODE and the *Client Certificate* is signed by this same *Root Certificate*.

A user role must be included in the client certificate for the *periNODE*. This role is crucial for the Role Based Access Control (RBAC) system. The *periNODE* supports three roles: ***admin***, ***super***, and ***user***:

- ***admin***: This user has complete read/write access to the *periNODE*, with no limitations.
- ***super***: This user can access and modify all resources, except for */security* and */update*.
- ***user***: This user can only view resources and cannot modify any.

**Note:** The ***user*** role is not permitted to modify actuator states via the RESTful API.

Mutual TLS (mTLS) is not active by default. It will also be deactivated following a security or factory reset.

**Note:** By default, role-based access control for both the RESTful API and web user interface is *deactivated*.

For more detailed information on the client certificate, please refer to https://docs.perinet.io.

### Web User Interface

The web user interface includes sections for inputting all three types of certificates: the *Host Certificate*, *Root Certificate*, and *Client Certificate*. For illustration, Figure 26 displays the input section specifically for the *Host Certificate*.

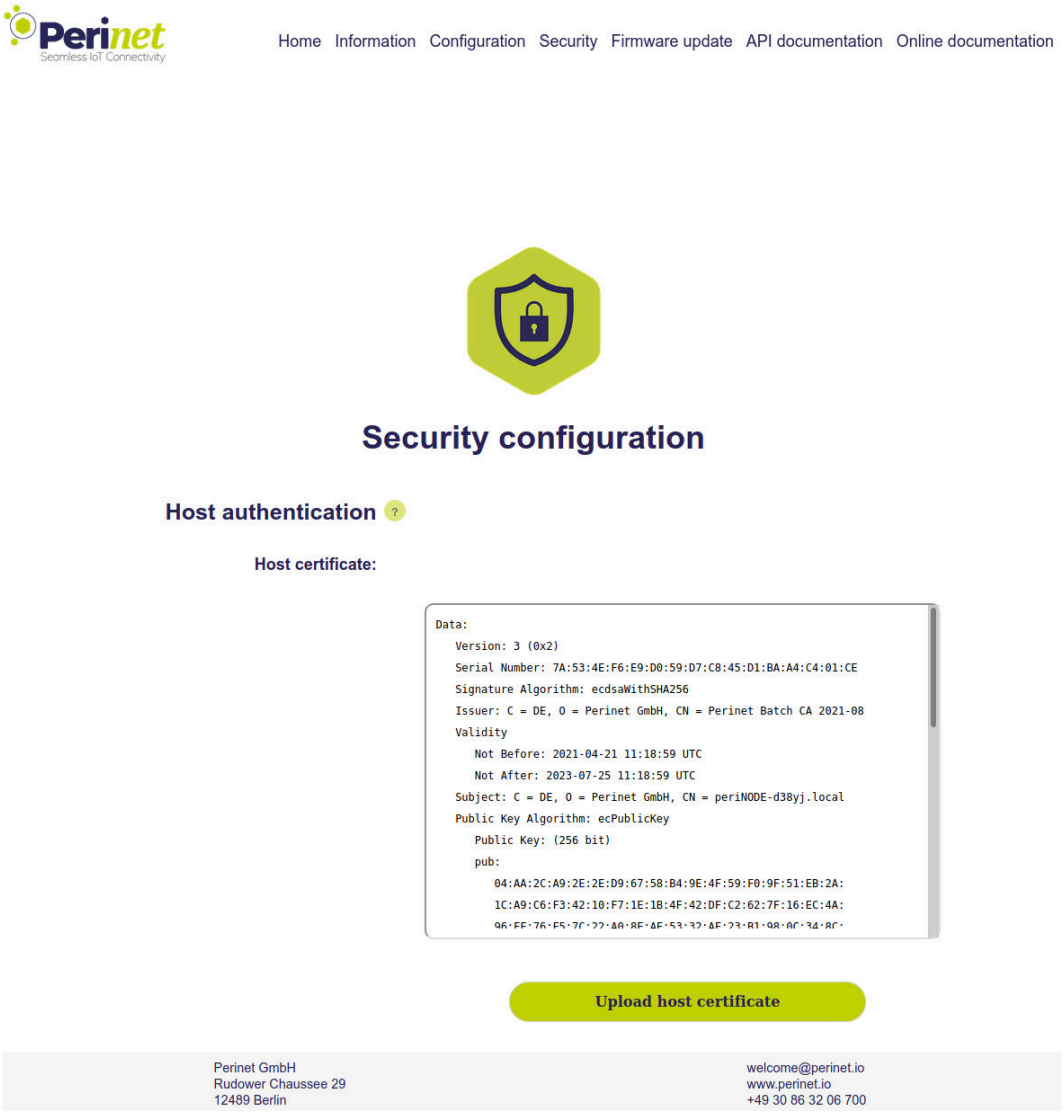



Figure 26: Security web user interface

The certificate displayed in the text area is the current one stored in the *periNODE*. This device accepts X.509 certificates encoded in PEM format (Base64 ASCII). Commonly, files with the extension `.pem` use this encoding, but it's also seen in `.crt`, `.cer`, and `.key` files.

For the *Host Certificate* and the *Client Certificate*, they should be uploaded along with their respective private keys concatenated at the end. Conversely, the *Root Certificate* should be uploaded without its private key.

For detailed guidance on generating certificates, visit https://docs.perinet.io.

*Enforce mTLS Access*

Activating the mTLS feature mandates that all remote clients authenticate themselves to the *periNODE* using a valid *Client Certificate*. This *Client Certificate* is verified against the *Root Certificate* already stored in the system.

**Note:** Before activating 'Enforce mTLS Access', ensure that a valid *Root Certificate* is already in place.

### RESTful API Certificate Deployment

The RESTful API facilitates the management of all three types of certificates through dedicated resources. These resources are modifiable exclusively by the *admin* role and must be updated using an HTTP `PATCH` request. The specific endpoints for each certificate are as follows:

- *Host Certificate*: Accessed via `/security/host-cert`.
- *Root Certificate*: Accessed via `/security/root-cert`.
- *Client Certificate*: Accessed via `/security/client-cert`.

The subsequent section provides an example of updating the *Root Certificate* using the **curl** command-line tool.

```
curl --data-binary @<certificate-file> \
 -H "Content-Type: application/octet-stream" \
 -X PATCH https://periNODE-serno.local/security/root-cert
# reboot device
curl -X PATCH https://periNODE-serno.local/reboot
```

'Enforce mTLS Access' can be activated through the RESTful API, as demonstrated in the upcoming excerpt:

```
curl --data='{"enable_user_role":true}' \
  -X PATCH https://periNODE-serno.local/security
# reboot device
curl -X PATCH https://periNODE-serno.local/reboot
```

### Reset Security Configuration

The **Reset security** button enables users to revert the security settings to their original factory defaults. This action will replace any user-configured Host and *Root Certificates* with the default ***Perinet ECC Root CA*** certificates. Additionally, any existing *Client Certificate* will be removed.

The procedure to reset the security configuration via the **curl** command-line tool is shown in the following code snippet:

```
curl -X PATCH https://periNODE-serno.local/security/reset
```

If the *periNODE* becomes inaccessible, such as in cases where the *Client Certificate* is lost, please consult the Factory Reset procedure detailed in Section Section 8.2.7.

### 8.2.4 MQTT

The application firmware of periSNOOP supports the MQTT protocol to publish sensor values and/or subscribe to an information stream, which can be used to control actuators. periSNOOP is compatible to MQTT protocol version 3.1.1.

The periSNOOP firmware is restricted to connect to MQTT brokers using TLS. It is capable of detecting MQTT brokers through an MDNS based service discovery protocol (DNS-SD). Brokers are discovered by browsing for the ***service_type*** `\_secure-mqtt.\_tcp` for MQTT brokers using TLS and the hostname of the broker e.g. ”mqtt-perimica-serno”. The periSNOOP autoconnects to an MQTT broker configured with the same application name. MQTT brokers which do not have the ability to advertise themselves via the mDNS based service discovery protocol can be configured manually via the web user interface or RESTful API.

As the periSNOOP firmware only supports IPv6, the local network and the broker must also be capable of communicating via IPv6.
The MQTT topic under which the periSNOOP smart adapter publishes messages or subscribes to is based on the ***application_name*** and ***element_name*** provided by the configuration parameters:

```
topic : "<application_name>/<element_name>"
```

The MQTT message payload is in a JSON encoded format. After configuring the periSNOOP with the desired broker, it is possible to subscribe to periSNOOP sensor data using any MQTT client, as the client that is provided by the ***mosquitto*** library.

#### *MQTT Client - Command Line Examples*

Since periSNOOP firmware only supports secured MQTT connections the ***mosquitto*** needs the server certificate of the broker. When using Perinets MQTT broker container and the configured MQTT broker is ***mqtt.local*** its server certificate can be obtained by

```
curl "https://mqtt.local/security/certificates/host_cert" --insecure > host.crt
```

If the configured ***application_name*** of the device is ***measure*** and the ***element_name*** is ***current***, one can subscribe to the periSNOOP messages (e.g. using the ***mosquitto*** library):

```
mosquitto_sub -h mqtt.local -p 8883 -t measure/current" --cafile host.crt
```

which prints sample JSON messages similar to the following:

```
{ "incarnation": 388, "sequence_number": 102, "data": { "unit": "mA", "value":
12.5268, "state": { "valid": true, "reason": "None" } } }
```

### 8.2.5 Secure MQTT

The communication between periSNOOP and MQTT broker also needs to be protected through TLSv1.2. The periSNOOP accesses the MQTT broker, verifying the broker certificate against the root certificate configured in the security page. Therefore, the periSNOOP MQTT client uses the same root certificate as the periSNOOP HTTPS server to authenticate the remote entity.

Mutual TLS (mTLS), where the remote MQTT broker authenticates the periSNOOP before continuing with the connection request, is supported as well. The periSNOOP uses the configured *client certificate* (see Section 8.2.3) in order to authenticate itself towards the MQTT broker.

*Mosquitto Examples using mTLS:*

Subscribing to periNODE sensor data:

```
mosquitto\_sub -h mqtt-secure.local -t test/topic -p 8883
---cafile root-ca.crt ---cert client.pem ---key client.key
```

Publishing messages:

```
mosquitto\_pub -h mqtt-secure.local -t test/topic -p 8883 -m testmessage
---cafile root-ca.crt ---cert client.pem ---key client.key
```

### 8.2.6 Firmware Update

Perinet may provide security and/or feature updates for periSNOOP over time. A firmware update shall be performed with the proper firmware update image artifact (e.g. matching variant).

**Note:** Inter-variant firmware updates are not supported by Perinet and may result in damaging the product.

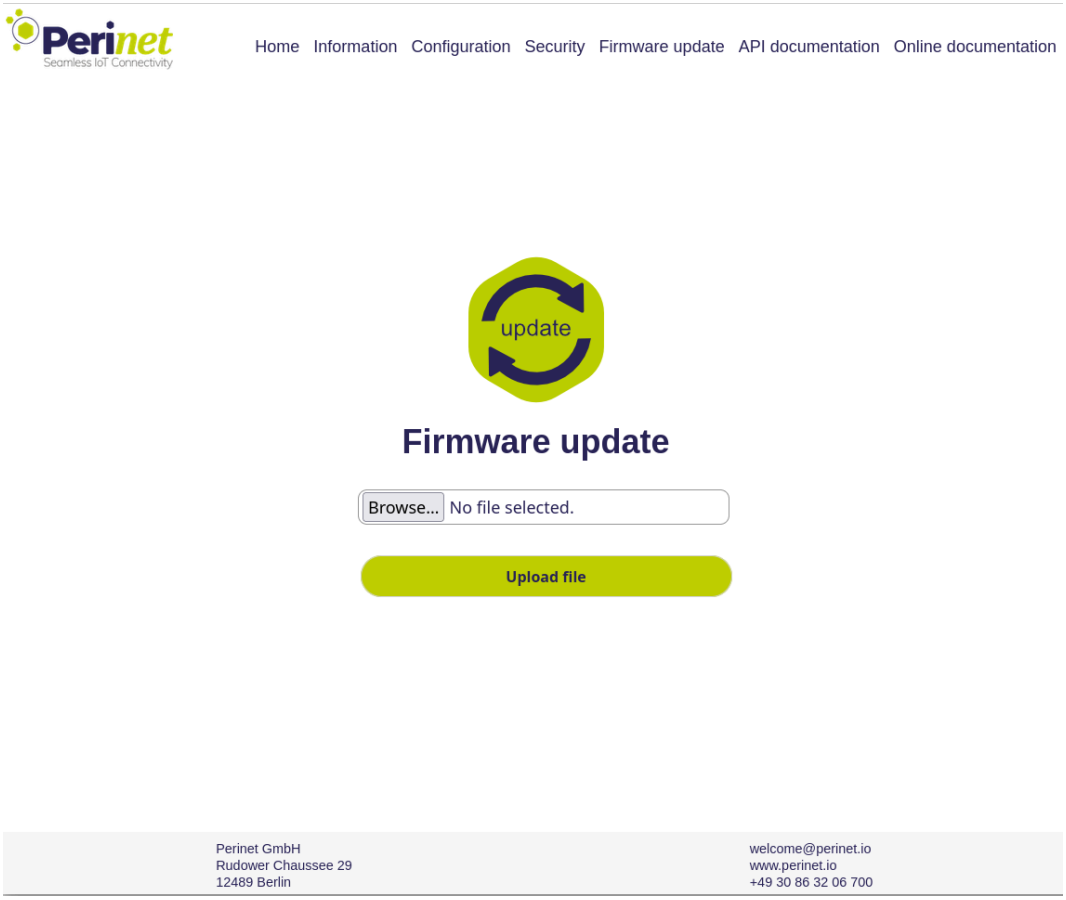



Figure 27: Firmware update

In order to keep the firmware of the device up to date, follow these steps:

1. Download the newest version of periSNOOP firmware https://downloads.perinet.io
2. Access the web page of the component through your browser: `https://hostname.local/`
3. Click on the **Firmware update** link located in the header of the page.
4. On the update page, click on **Browse...** and select the new firmware image to be uploaded.

5. Finally click on the **Upload file** button to install the new firmware.

**Note:** After a successful firmware update, the reboot of the component to the newly installed firmware will only be automatically triggered when using the web user interface.

### RESTful API

The RESTful API accepts an update image via a *PUT* request on the resource */update*. The following command line shows how to perform an update with the **curl** tool:

```
curl --data-binary @<filename> \
    -H "Content-Type: application/octet-stream" \
    -X PUT https://periNODE-serno.local/update
# reboot into new firmware
curl -X PATCH https://periNODE-serno.local/reboot
```

### 8.2.7 Factory Reset

In order to bring periSNOOP back into fresh production state a factory reset performs the following steps:

- Erase host certificate
- Reset all configuration values back to production state
- Reset firmware version to produced firmware version

A factory reset of a periSNOOP can be performed via the RESTful API remotely or physically by using a dedicated reset cable. The former solution might not be possible in some cases, e.g. where the client certificate has been lost and the access to the device is not possible anymore.

#### *Factory Reset via Reset Cable*

Connect the periSNOOP via the reset cable (see Figure 28) for at least 20 seconds to a powered network, e.g. a powered periSTART. Please ensure that the power supply of the network is active before connecting the product.


Figure 28: Perinet Reset Cable for Factory Reset

#### *Factory Reset via RESTful API*

When performing the factory reset via the RESTful API, a config as well as a security reset needs to be performed:

```
curl -X PATCH https://periSNOOP-serno.local/config/reset
curl -X PATCH https://periSNOOP-serno.local/security/reset
```

# 9 RESTful API

A periCORE based product features a RESTful [3] API for user access. To connect, use the hostname `periCORE-sernm.local`, where *sernm* is a unique identifier based on the product's serial number as explained in Section 10. The API provides various routes, which are listed in Table 12. These routes and their functions will be detailed in the subsequent subsections.

| URL Resource | Description |
|---|---|
| **/info** | See Section 9.1 |
| **/config** | |
| **/config/reset** | |
| **/update** | See Section 9.2 |
| **/reboot** | |
| **/reset** | |
| **/production/oem-firmware** | |
| **/production/reset** | |
| **/security** | See Section 9.3 |
| **/security/host-cert** | |
| **/security/root-cert** | |
| **/security/client-cert** | |
| **/security/reset** | |

Table 12: RESTful API routes overview

## 9.1 Info Service

**/info** periCORE based product information object (see Section 9.1.1).

`GET` expects empty body.

`200` Return `NodeInfo` object.

`204` Return empty body. No data is available.

`401` Unauthorized access, returns empty body

`500` Internal server error on unexpected error, returns empty body.

**/config** periCORE based product configuration object (see Section 9.1.2).

`GET` expects an empty body.

`200` Return `NodeConfig`.

`204` Return empty body. No data is available.

`401` Unauthorized access, returns empty body

`500` Internal server error on unexpected error, returns empty body.

`PATCH` expects a complete or partial `NodeConfig` object.

`204` The Request has been accepted but was not processed yet. The object will be deserialized and merged. Given *keys* will be overwritten. The object will be stored persistently.

`401` Unauthorized access, returns empty body

`500` Internal server error on unexpected error, returns empty body.

**/config/reset** periCORE based product configuration object reset (see Section 9.1.2).

`PATCH` expects an empty body.

`204` The Request has been accepted but was not processed yet. The default content of the object will be restored and the object will be stored persistently.

`401` Unauthorized access, returns empty body

`500` Internal server error on unexpected error, returns empty body.

### 9.1.1 Node Info

```
syntax="proto3";
package perinet.api;

message SwVersion {
  uint32 api = 1; // API compatibility incarnation
  uint32 build = 2; // build iteration
  uint32 version_number = 3; // firmware feature level incarnation
}

message NodeInfo {
    VersionInfo version_info = 1; // firmware version information
    string manufacturer = 2; // manufacturer identifier
    string hostname = 3; // host network identification name, e.g. periNODE-<id>.local
    periCORE-<id>.local
    string mac_address = 4; // unique mac address
    string product_charge = 5; // production batch identifier
    string product_part_number = 6; // product part identifier
    string product_serial = 7; // serial number of the product
    string product_name = 8; // calling name of the product
    string product_version = 9; // version of the product at production time
    string pericore_charge = 10; // batch identifier of the included periCORE
    string pericore_part_number = 11; // periCORE part identifier
    string pericore_serial = 12; // periCORE serial number
    string pericore_version = 13; // periCORE version identifier
}
```

Listing 1: periCOREs *NodeInfo* object definition

### 9.1.2 Node Config

```
syntax="proto3";
package perinet.api;
option go_package = "perinet/api";

message NodeConfig {
    message Interface {
        google.protobuf.Any type = 1; //the type of interface, the particular interface
    has been configured to. Implementation specific.
        string element_name = 2; // element name of a particular interface, like an
    sensor source or an actuator sink.
        float period_seconds = 3; // in seconds, 0 means only event based (triggered)
    publishing
        uint32 samples_per_period = 4;  //defines how many values shall be sampled
    within a period,
                                        //the published value will be the rounded
    average
        // repeated Trigger trigger = 5;
    }
    string application_name = 1; // identification of the application the periCORE based
     node is assigned to
    string mqtt_broker_name = 2; // URI of the MQTT broker, the periCORE based node
    shall be connected to
    repeated Interface configs = 3;
}
```

Listing 2: periCOREs *NodeConfig* object definition

## 9.2 Life Cycle Service

**/update** periCORE based product *Update Firmware Image* (see Section 9.2.1).

`PUT` expects octet-stream body.

`204` Returns empty body. The Request has been received and is being processed.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

`404` Resource is not available in this life state, e.g for *periCORE Production State* (see **??**).

**/reboot** periCORE based product configuration object.

`PATCH` expects an empty body.

`204` Returns an empty body. The request is being processed and the device is performing a software reboot.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

**/reset** periCORE based product remote factory reset (see **??**).

`PATCH` expects an empty body.

`204` The Request has been accepted and is being processed. All persistent data is reset to its default state. (see **??**)

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

**/production/oem-firmware** periCORE based product *OEM Firmware Image* (see **??**).

`PUT` expects octet-stream body (see Section 9.2.1).

`204` Returns empty body. The Request has been received and is being processed.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

`404` Resource is not available in this life state, e.g for *OEM Production State* (see **??**).

**/production/reset** periCORE based product *Production Reset* (see **??**).

`PATCH` expects an empty body.

`204` The Request has been accepted and is being processed. All persistent data is reset to its default state. (see **??**)

`404` Internal server error on unexpected error, returns empty body.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

### 9.2.1 Firmware Image

The Life Cycle Service of a periCORE based product requires the *Firmware Image* to be in a signed binary format. For more comprehensive details on this format and related procedures, refer to the periCORE Firmware Development Application Note [8].

## 9.3 Security Service

**/security/host-cert** periCORE based product host certificate object (see **??**).

`GET` expects an empty body.

`200` Return the host public certificate.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

`PATCH` expects a text/plain-encoded body containing the certificate.

`204` The request has been accepted and processed. Returns empty bod

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

**/security/root-cert** periCORE based product root certificate object (see **??**).

`GET` expects an empty body.

`200` Return the root public certificate.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

`PATCH` expects a text/plain-encoded body containing the root certificate.

`204` The request has been accepted and processed. Returns empty body.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

**/security/client-cert** periCORE based product client certificate object (see **??**).

`GET` expects an empty body.

`200` Return the client public certificate.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

`PATCH` expects a text/plain-encoded body containing the certificate.

`204` The request has been accepted and processed. Returns empty body.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

**/security/reset** periCORE based product security configuration reset object. (see **??**)

`PATCH` expects an empty body.

`204` Security configuration is reset to factory defaults: root and host certificates are replaced with the factory certificates, the client certificate is deleted, and the mTLS feature is disabled.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

**/security** periCORE based product mTLS configuration object (see **??**).

`GET` expects an empty body.

`200` Returns a JSON-encoded object containing the key `enable_user_role` and it's value.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

`PATCH` Expects a JSON-encoded object containing the key `enable_user_role` and it's value.

`204` The request has been accepted and processed. Returns empty body.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

# 10 Product Marking

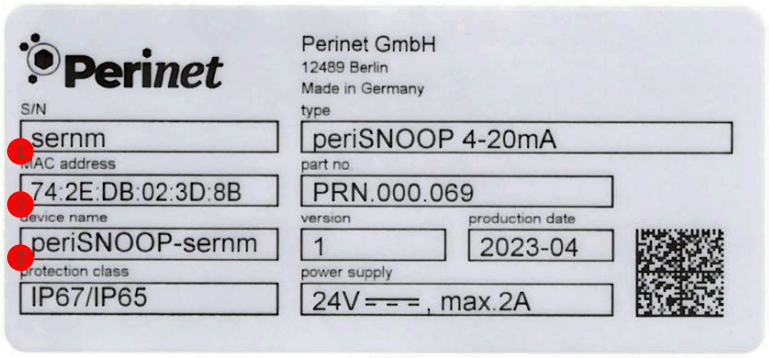



Figure 29: periSNOOP 4-20mA's electrical connectivity.

| Pos. | Name | Desciption |
|---|---|---|
| 1 | `serial number` | unique serial number of the product |
| 2 | `MAC address` | unique network address |
| 3 | `hostname` | unique DNS name of the device |

Table 13: periSNOOP 4-20mA's product marking layout.

# 11 Further Documentation

| Document Name | Description |
|---|---|
| periCORE Product Summary [9] | A product features summary documentation for the product *periCORE*. |
| periCORE Datasheet [4] | A detailed reference documentation of the product *periCORE*. |
| periCORE Development Kit Product Summary [5] | A product features summary documentation for the product *periCORE Development Kit*. |
| periCORE Development Kit Setup Application Note [6] | A setup guide for the product *periCORE Development Kit*. The starting point when you are new to the product which describes how to quickly set up a development environment for firmware development for a periCORE based product. |
| periCORE Development Kit User Guide [7] | A guide and reference documentation for the product *periCORE Development Kit*. |

| Document Name | Description |
|---|---|
| periLINE Product Summary [10] | A product features summary documentation for the product *periLINE*. |
| periNODE 0-10V Product Summary [11] | A product features summary documentation for the product *periNODE 0-10V*. |
| periNODE Pt100 Product Summary [13] | A product features summary documentation for the product *periNODE Pt100*. |
| periNODE GPIO Product Summary [12] | A product features summary documentation for the product *periNODE GPIO*. |
| periSWITCH 3-port Product Summary [15] | A product features summary documentation for the product *periSWITCH 3-port*. |
| periSTART Standard Product Summary [14] | A product features summary documentation for the product *periSTART standard*. |
| Smart Components Datasheet [17] | A detailed reference documentation of the Smart Components products (*periLINE*, *periNODE 0-10V*, *periNODE Pt100*, *periNODE GPIO*, *periSWITCH 3-port*, *periSTART standard*). |

# 12 Ordering Information

| Ordering Code | Product Name | Description |
|---|---|---|
| **PRN.000.069** | periSNOOP 4-20mA | Single Pair Ethernet 4-20mA based sensor monitor |

| Ordering Code | Product Name | Description |
|---|---|---|
| **PRN.000.008** | periSWITCH 3-port | 3-port hybrid SPE switch |
| **PRN.000.003** | periSTART standard | Media converter from SPE to standard Ethernet |
| **PRN.000.017** | periLINE 0.2m | Hybrid SPE cable with M8 connectors |
| **PRN.000.025** | periMICA | Open modular edge computer |
| **PRN.000.020** | periCORE Development Kit | Full featured firmware development setup |

# 13 Contact & Support

For customer support, please call us at **+49 30 863 206 701** or send an e-mail to *support@perinet.io*.

For complete contact information visit us at www.perinet.io

# A List of Figures

# B List of Listings

# C List of Tables

# D Glossary

**100BASE-T1** A Ethernet Standard where two endpoints are connected by a single twisted pair cable. It is one of the so-called Single Pair Ethernet (SPE) standards. It operates in full-duplex with a data rate of 100 MBit per second. Furthermore, it uses PAM-3 modulation with a voltage level from -1 to +1V, differentially on the two wires. 8, 10, 18, 19

**API** Application Programming Interface. 25, 45

**DNS-SD** DNS Service Discovery [1] is a way of using standard DNS programming interfaces, servers and packet formats to browse the network for services. 6, 39

**full-duplex** In a full-duplex system, both parties can communicate with each other simultaneously. Transmitting or receiving of information is operated independently contrary to a half-duplex scheme.. 7

**IoT** Internet of Things. 30

**JSON** JavaScript Object Notation is standard text-based format for representing structured data based on JavaScript object syntax. 39

**LLMNR** The Link-Local Multicast Name Resolution is a protocol based on the Domain Name System packet format that allows IPv6 hosts to perform name resolution for hosts on the same local link. 25

**mDNS** multicast Domain Name Service [2], a protocol that implements a local distributed name resolving mechanism. 6, 25, 39

**MQTT** Message Queuing Telemetry Transport is a lightweight, publish-subscribe based network protocol that transports messages between devices. 30, 39, 40

**mTLS** Mutual TLS extends the TLS protocol by requiring clients to pass certificates, allowing to provide authorization mechanisms of Application services. 6, 24, 40, 49

**RBAC** Role Based Access Control. 6

**REST** REpresentational State Transfer, a web API style. 25, 28, 29, 39, 43, 44

**SPE** Single Pair Ethernet. 1, 7–10

**TLS** Transport Layer Security Protocol. Used by Application Protocols like MQTT or HTTP to allow secure data transfer. 25, 39

# E References


[1] S. Cheshire and M. Krochmal. *DNS-Based Service Discovery*. RFC 6763. http://www.rfc-editor.org/rfc/rfc6763.txt. RFC Editor, Feb. 2013. URL: http://www.rfc-editor.org/rfc/rfc6763.txt.

[2] S. Cheshire and M. Krochmal. *Multicast DNS*. RFC 6762. http://www.rfc-editor.org/rfc/rfc6762.txt. RFC Editor, Feb. 2013. URL: http://www.rfc-editor.org/rfc/rfc6762.txt.

[3] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. RFC 7231. http://www.rfc-editor.org/rfc/rfc7231.txt. RFC Editor, June 2014. URL: http://www.rfc-editor.org/rfc/rfc7231.txt.

[4] Perinet GmbH. periCORE Datasheet. PRN.100.375. https://docs.perinet.io/PRN100375-periCOREDatasheet.pdf.

[5] Perinet GmbH. periCORE Development Kit Product Summary. PRN.100.546. https://docs.perinet.io/PRN100546-periCOREDevelopmentKitProductSummary.pdf.

[6] Perinet GmbH. periCORE Development Kit Setup Application Note. PRN.100.376. https://docs.perinet.io/PRN100376-periCOREDevelopmentKitSetupApplicationNote.pdf.

[7] Perinet GmbH. periCORE Development Kit User Guide. PRN.100.378. https://docs.perinet.io/PRN100378-periCOREDevelopmentKitUserGuide.pdf.

[8] Perinet GmbH. periCORE Firmware Development Application Note. PRN.100.379. https://docs.perinet.io/.

[9] Perinet GmbH. periCORE Product Summary. PRN.100.301. https://docs.perinet.io/PRN100301-periCOREProductSummary.pdf.

[10] Perinet GmbH. periLINE Product Summary. PRN.100.386. https://docs.perinet.io/PRN100386-periLINEProductSummary.pdf.

[11] Perinet GmbH. periNODE 0-10V Product Summary. PRN.100.380. https://docs.perinet.io/PRN100380-periNODE0-10VProductSummary.pdf.

[12] Perinet GmbH. periNODE GPIO Product Summary. PRN.100.382. https://docs.perinet.io/PRN100382-periNODEGPIOProductSummary.pdf.

[13] Perinet GmbH. periNODE Pt100 Product Summary. PRN.100.381. https://docs.perinet.io/PRN100381-periNODEPt100ProductSummary.pdf.

[14] Perinet GmbH. periSTART Standard Product Summary. PRN.100.383. https://docs.perinet.io/PRN100383-periSTARTstandardProductSummary.pdf.

[15] Perinet GmbH. periSWITCH 3-port Product Summary. PRN.100.385. https://docs.perinet.io/PRN100385-periSWITCH3-portProductSummary.pdf.

[16] Perinet GmbH. Security Certificates Installation Guide. PRN.100.447. https://docs.perinet.io/PRN100447-SecurityCertificatesInstallationGuide.pdf.

[17] Perinet GmbH. Smart Components Datasheet. PRN.100.387. https://docs.perinet.io/PRN100387-SmartComponentsDatasheet.pdf.

[18] “ISO/IEC/IEEE International Standard - Part 3: Standard for Ethernet - Amendment 1: Physical Layer Specifications and Management Parameters for 100 Mb/s Operation over a Single Balanced Twisted Pair Cable (100BASE-T1)”. In: *ISO/IEC/IEEE 8802-3:2017/Amd 1:2017(E)* (2018), pp. 1–92. DOI: 10.1109/IEEESTD.2018.8310988.

# F Revision History

| Revision | Date | Author(s) | Description |
|---|---|---|---|
| 1 | 2023-09-19 | shoe | initial release |
| 2 | 2023-11-15 | nsav | engineering samples |
| 3 | 2024-05-13 | clip,shoe,tkla | editorial review Fix extra characters for listings to allow copy and paste. |