Sensor Beschreibung
SMARTI

Autor: **Dipl. Ing. (FH) Benjamin Cermak**
Version: 02.00
Datum: November 2024

Inhaltsverzeichnis

1 HARDWARE

SMARTI LED Status

- LED grün: POWER ON
- LED orange: Smart Meter Daten Empfang
- LED rot: Dauer leuchten -> Device booting
  Blinken 1sek. -> keine aktive WLAN Verbindung, WLAN Verbindungsversuch
  AUS: aktive Netzwerkverbindung

Übermittelte Daten

OBIS Codes:
0.1.0.32.7.255: Momentanspannung L1 [V]
0.1.0.52.7.255: Momentanspannung L2 [V]
0.1.0.72.7.255: Momentanspannung L3 [V]
0.1.0.31.7.255: Momentanstrom L1 [A]
0.1.0.52.7.255: Momentanstrom L2 [A]
0.1.0.72.7.255: Momentanstrom L3 [A]
0.1.0.1.7.255: Momentaner Leistungsbezug (gesamt) [W]
0.1.0.2.7.255: Momentane Leistungseinspeisung (gesamt) [W]
0.1.0.1.8.255: Zählerstand Bezug Wirkleistung [Wh]
0.1.0.2.8.255: Zählerstand Einspeisung Wirkleistung [Wh]
0.1.0.3.8.255: Zählerstand Bezug Blindleistung [Varh]
0.1.0.4.8.255: Zählerstand Einspeisung Blindleistung [Varh]
0.1.0.13.7: Leistungsfaktor

2 INBETRIEBNAME

Android APP:

IOS:

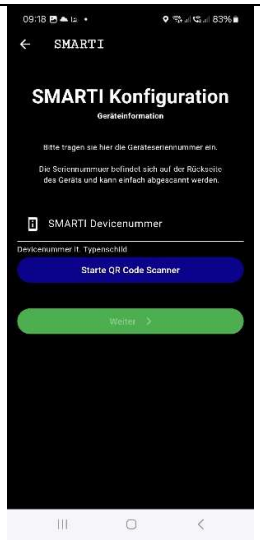
2.1 Neues Gerät einrichten

2.1.1 Konfiguration über Smart Konfigurator

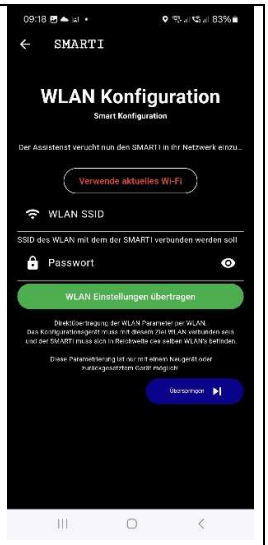
- SMARTPHONE mit selben WLAN verbinden mit welchem auch der SMARTI verbunden werden soll
- SMARTI Config APP starten

1. **Neues Gerät einrichten über Smart Konfigurator START**

2. **Geräte auswählen**



3. **Geräte Seriennummer Barcode auf Rückseite scannen**

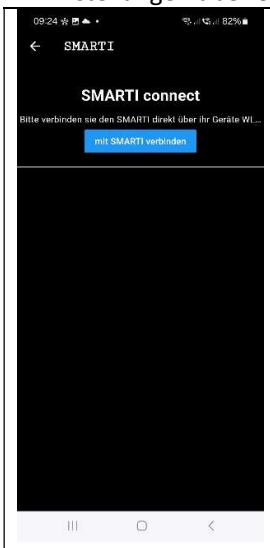
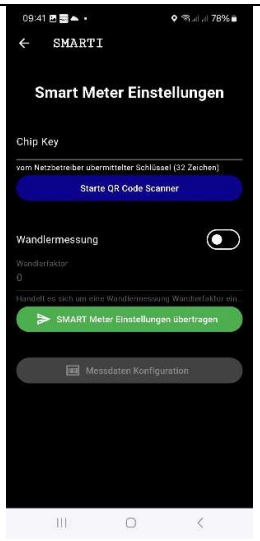
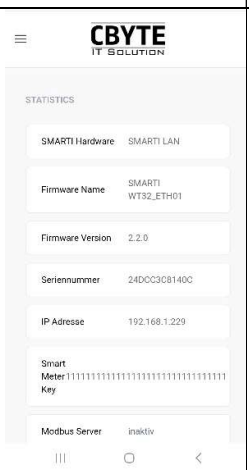


4. **WLAN Daten eintragen und übertragen**

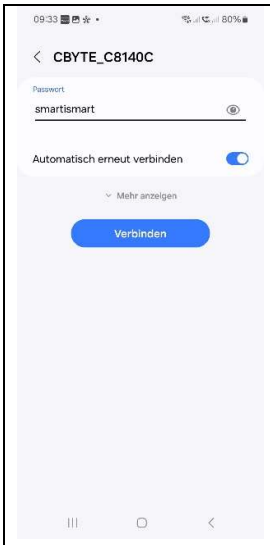
5. SMARTI einstecken

Konnte die WLAN Konfiguration erfolgreich übermittelt werden, wird das Gerät über die Rückgemeldete IP Adresse automatisch verbunden und mit Schritt 7 fortgesetzt. Andererseits wird zur Konfiguration über eine direkte Verbindung gewechselt

Konnte die Verbindung zum WLAN nicht automatisch konfiguriert werden, müssen die WLAN Einstellungen über eine direkte Verbindung zum SMARTI Konfiguriert werden

			
6. Konfigurationsgerät (Smartphone) direkt mit dem WLAN des SMARTI verbinden.	Hierzu in die WLAN Einstellungen wechseln und mit der SSID CBYTE_***** verbinden. **SSID: CBYTE_xxxxxx** **PW: smartismart**	7. Ist eine Verbindung vorhanden können nun die WLAN Zugangsdaten eingetragen werden	**8. 32 Stelliger SMART Meter Schlüssel vom Netzbetreiber eingetragen** Tipp: QR Code über https://goqr.me/de/# erstellen und einfach abscannen
			
9. SMARTI ist nun noch über die eigene WLAN Verbindung und die IP 192.168.4.1 verbunden. Weitere Einstellungen können nun über die Menüpunkte konfiguriert werden.	**10. Über die Statusseite -> Statistics kann die IP des SMARTI abgerufen werden.**		

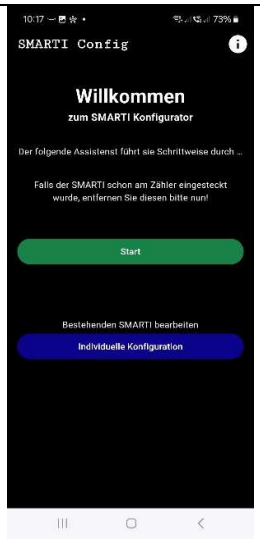
2.1.2 Konfiguration über Individuelle Konfiguration



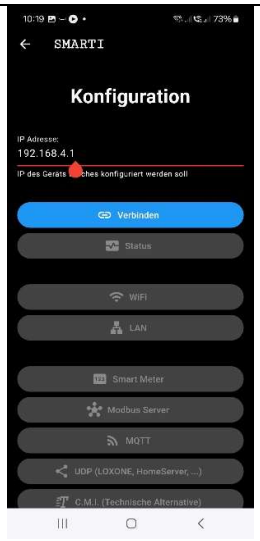
1. SMARTI einstecken und über direkte WLAN Verbindung verbinden

Hierzu in die WLAN Einstellungen wechseln und mit der SSID CBYTE_***** verbinden.

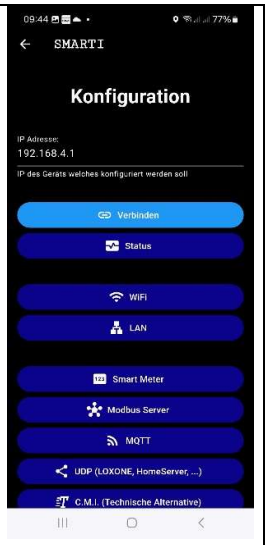
SSID: CBYTE_xxxxxx
PW: smartismart



2. Individualle Konfiguration starten



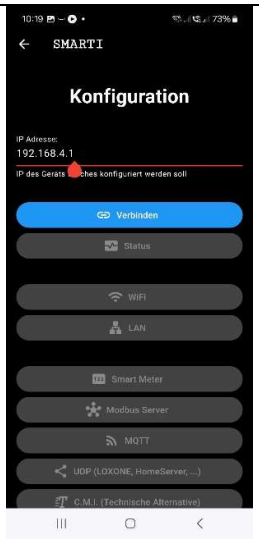
3. Über IP 192.168.4.1 verbinden



4. SMARTI Konfigurieren

2.2 Bestehendes Gerät konfigurieren / ändern

5. Individualle Konfiguration starten



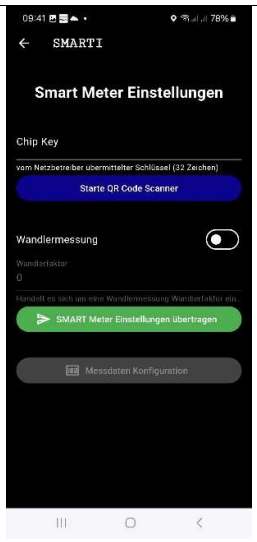
6. Über IP über welche der SMARTI in Netzwerk erreichbar ist verbinden

7. SMARTI Konfigurieren

2.3 SMARTI Konfiguration

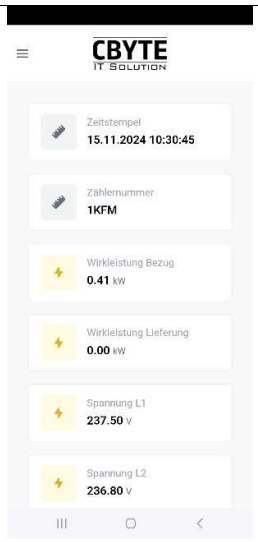
2.3.1 Smart Meter Konfiguration

Zum einstellen oder ändern des SMART Meter KEY. Wurde im Normalfall schon im Onboardingprozess konfiguriert.



32 Stelliger SMART Meter Schlüssel vom Netzbetreiber eingetragen

Tipp: QR Code über https://goqr.me/de/# erstellen und einfach abscannen



Smart Meter Daten prüfen über Statusseite.
Es kann ein paar Minuten dauern bis die ersten Daten Decodiert werden und erscheinen

2.3.2 MQTT

MQTT kann für die Einbindung in HomeAssistant oder iQBroker verwendet werden.

Hierzu sendet der SMARTI seine Daten an einen MQTT Brocker welcher z.b: auf dem HomeAssistant läuft.

TOPIC: CBYTE/<ZÄHLERNUMMER>/sensor

HomeAssistant MQTT Einbindungsbeispiel: HomeAssistent MQTT.yaml

2.3.3 UDP

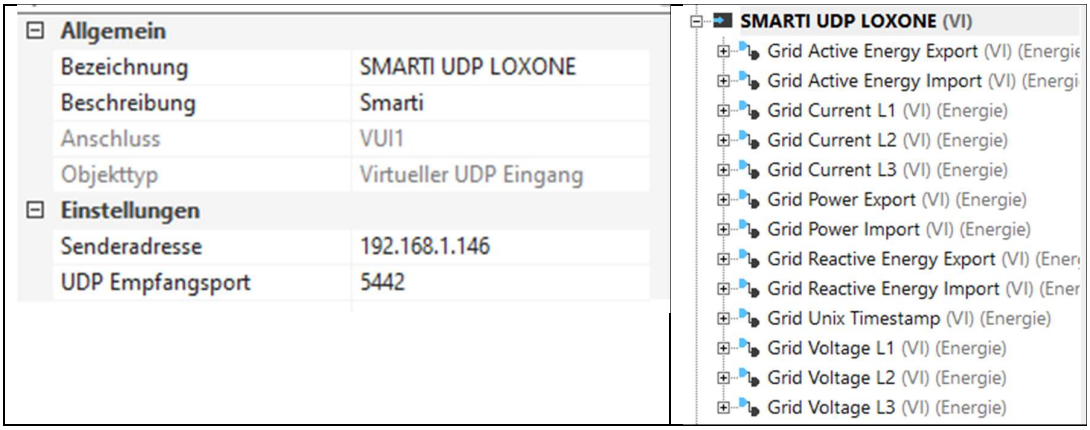
UDP Protokoll für LOXONE oder GiraHomeserver

Protokoll:

Byte 1-5 -> OBIS Code in HEX
Byte 6-8 -> Value in HEX
BSP: 01 00 20 07 ff 18 5b
-> OBIS Code: 1.0.32.7.0.255 (GRID Voltage L1)
-> VALUE: 5b18 -> 23320 -> 233,20V

LOXONE:
Für Loxone steht eine eigene Library zur Verfügung welche Import werden kann.
Für die Kommunikation muss die IP des SMARTI und der PORT welcher in der SAMRTI Config APP eingestellt wurde in der LOXONE Config angepasst werden. In Der SMARTI Config APP ist die IP des LOXONE Server als Gegenstelle einzutragen.

Download LOXONE Library



2.3.4 C.M.I. Technische Alternative

Für eine Anbindung eines C.M.I. der Technischen Alternative steht ein eigenes Protokoll zur Verfügung.

2.3.5 Modbus Server

Protocol SMARTI Smart Meter Interface

Communication protocol	ModbusTCP
Port	502
Data encryption	Big Endian
Communication mode	DEVICE is Modbus Master
Adress Slave (ID)	1

Input registers - Read data – Analog IN

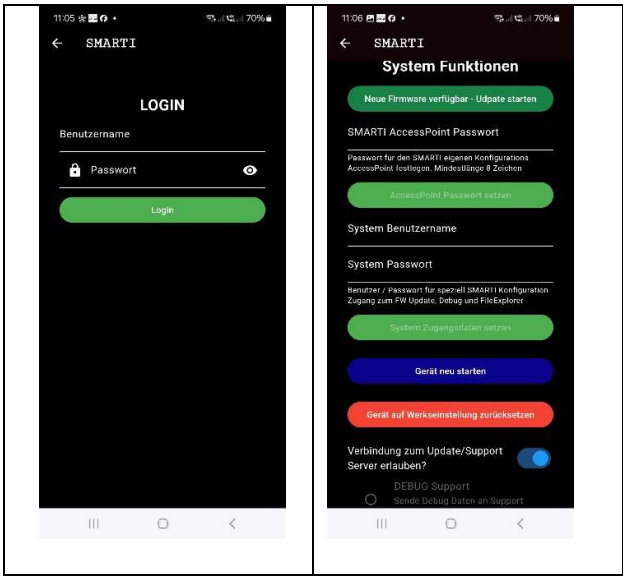
Modbus Adress	Command	Data Type	Name	Unit	Description OBIS Code
1001	3 – Read Holding Register (4x)	16-bit unsigned integer	Grid voltage L1	mV	1.0.32.7
1002	3 – Read Holding Register (4x)	16-bit unsigned integer	Grid voltage L2	mV	1.0.52.7
1003	3 – Read Holding Register (4x)	16-bit unsigned integer	Grid voltage L3	mV	1.0.72.7
1004	3 – Read Holding Register (4x)	16-bit unsigned integer	Grid current L1	mA	1.0.31.7
1005	3 – Read Holding Register (4x)	16-bit unsigned integer	Grid current L2	mA	1.0.51.7
1006	3 – Read Holding Register (4x)	16-bit unsigned integer	Grid current L3	mA	1.0.71.7
1007	3 – Read Holding Register (4x)	16-bit unsigned integer	Grid power import	W	1.0.1.7
1008	3 – Read Holding Register (4x)	16-bit unsigned integer	Grid power export	W	1.0.2.7
1009	3 – Read Holding Register (4x)	32-bit Unsigned Integer	Grid active energy import	Wh	1.0.1.8
1011	3 – Read Holding Register (4x)	32-bit Unsigned Integer	Grid active energy export	Wh	1.0.2.8
1013	3 – Read Holding Register (4x)	32-bit Unsigned Integer	Grid reactive energy import	Wh	1.0.3.8
1015	3 – Read Holding Register (4x)	32-bit Unsigned Integer	Grid reactive energy export	Wh	1.0.4.8
1017	3 – Read Holding Register (4x)	32-bit Unsigned Integer	TimeStamp	-	0.0.1.0
1019	3 – Read Holding Register (4x)	32-bit Unsigned Integer	Grid powerfactor	-	0.1.13.7

2.3.6 System

Unter System können Systemparameter geändert werden.

Default Zugangsdaten (je nach FW stand unterschiedlich):

User: admin (oder leere lassen)
Passwort: cbsmart (oder leere lassen)



SMARTI AccessPoint Passwort:

Passwort für die direkte Verbindung zum SMARTI.
Der Interne AccessPoint für eine direkte Verbindung zum SMARTI wir nach Neustart des Geräts für 10 Minuten aktiviert und danach aus Sicherheitsgründen deaktiviert.

Default Passwort: smartismart

System Benutzername und Passwort:

Zugangsdaten zu diesem System Menü und Zugriff für lokales Firmwareupdate, Zertifikats Manager und Debugger.

Default Zugangsdaten (je nach FW stand unterschiedlich):

User: admin (oder leere lassen)
Passwort: cbsmart (oder leere lassen)

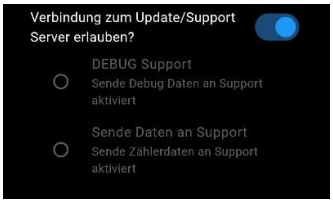
Gerät neu starte:

Für einen Neustart des SMARTI durch. Internen AP ist für 10 Minuten aktiv.

Gerät auf Werkseinstellungen zurücksetzen:

Alle Einstellungen werden gelöscht und Gerät befindet sich im Auslieferungszustand.

Verbindung zum Update/Support Server erlauben:



Ist diese Funktion kann das Gerät ggf. über einen Update Server mit einem neuem Firmware geupdatet werde so das immer die neusten Funktionen zur Verfügung stehen. Des weiteren ist es möglich generierte Rohdaten der Zählerschnittstelle (DebugDaten) oder auch die Zählerdaten an den SupportServer zu übermitteln.

Welche Funktion vom Support aktiviert sind ist hier ersichtlich.

Verbindung zum Update/Support Server erlauben:

aktiv: SMARTI kann über das Device Management von CBYTE IT Solution mit neuen Updates versorgt werden.

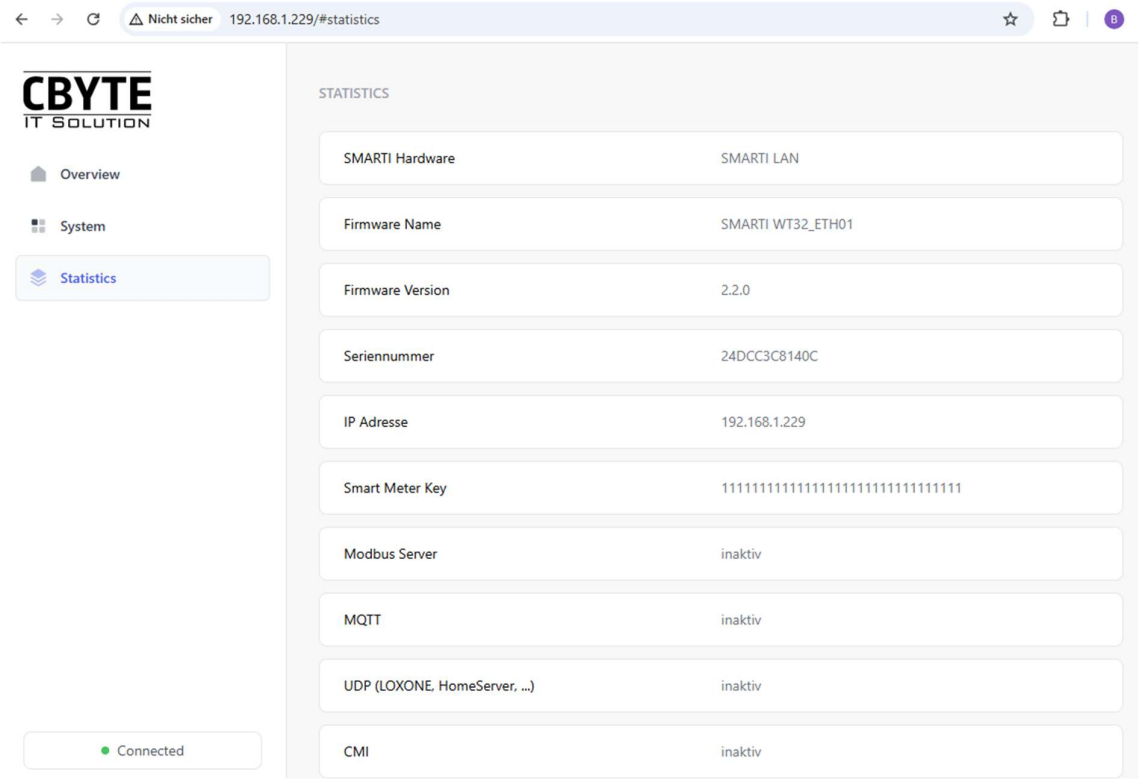
Debug Support: Rohdaten werden an Supportserver übermittelt. Verbindung zum Update/Support Server muss aktiviert sein!

Sende Daten an Support: Zählerdaten werden an Supportserver übermittelt Verbindung zum Update/Support Server muss aktiviert sein!

3 Dashbaord

Über die IP Adresse des SMARTI kann auf ein Dashbaord zugegriffen werden auf dem die aktuelle Zählerdaten dargestellt werden.

Informationen über die Hardware und Softwarestand sind unter Statistics abrufbar.

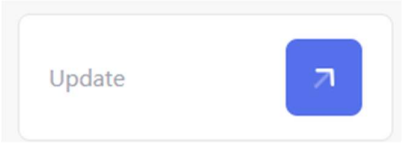


4 Firmwareupdate

Ein Firmwareupdate kann Online oder Offline erfolgen.

Online: wenn der Sensor mit dem Internet verbunden ist, prüft der Sensor 1x täglich ob es für ihn auf einem Updateserver ein Update gibt und lädt dieses automatisch herunter und installiert es.

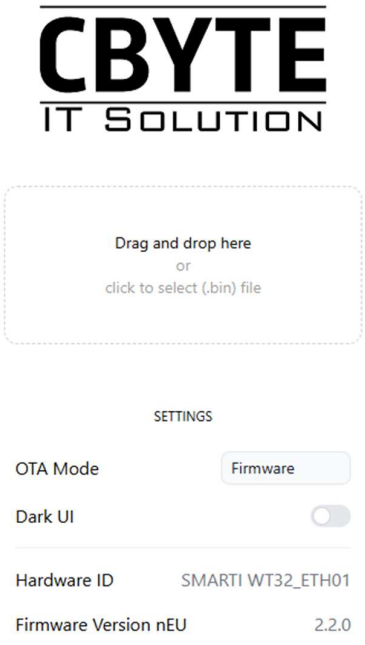
Offline: über das Dashbaord unter System -> Update



kann die Update Seite abgerufen werde und ein bereitgestelltes Firmwareimage lokal eingespielt werden.

Default Zugangsdaten:

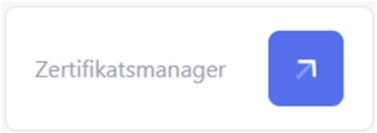
User: admin
Passwort: cbsmart



5 Zertifikatsmanager

Der Zertifikats Manger dient dazu ein Zertifikat für die sichere Verbindung für MQTTS zu hinterlegen.

Dashbaord -> System -> Zertifikats Manger



Default Zugangsdaten:

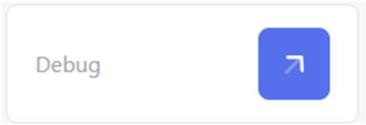
User: admin
Passwort: cbsmart

Das Zertifikat für den MQTT Brocker muss als **ca.crt** hochgeladen werden

6 Debugger

Für Supportzwecke kann ein Debugger gestartet werden um Protokolle aufzuzeichnen und an den Support für Analysezwecke zu übermitteln.

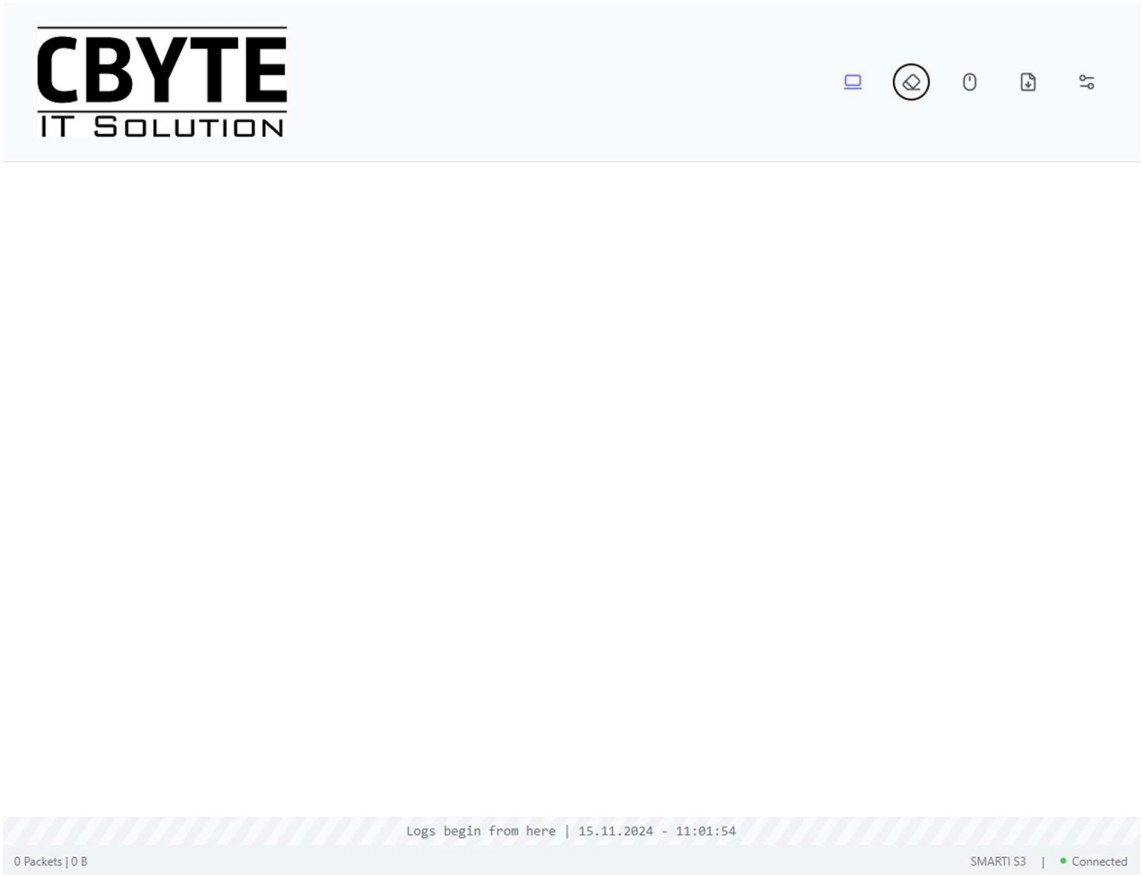
Dashbaord -> System -> Debug



Das Logging läuft so lange die Seite aktiv ist.

Default Zugangsdaten:

User: admin
Passwort: cbsmart


CBYTE
IT SOLUTION
Logs begin from here | 15.11.2024 - 11:01:54
0 Packets | 0 B
SMARTI S3 | Connected

7 REST API

Der SMARTI verfügt über eine REST-API Schnittstelle, über welche eine direkte Parametrierung oder Kommunikation mit dem Sensor erfolgen kann. Voraussetzung ist, dass der Sensor über das Netzwerk angesprochen werden kann, oder eine direkte Verbindung mit dem Sensor eigenen AP besteht.

GET: http://<ip-address>:8080/
POST: http://<ip-address>:8080/api

7.1 WiFi-Konfiguration

7.1.1 WiFi Einstellungen abfragen

Command Typ: GET http://<ip-address>:8080/getwifi_settings

Response:
```
{"type": "active", "value": bool},
{"type": "ssid", "value": ""},
{"type": "wifistaticip_active", "value": bool},
{"type": "wifistaticip", "value": ""},
{"type": "wifistaticsub", "value": ""},
{"type": "wifistaticgw", "value": ""},
{"type": "wifistaticdns","value": ""}
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/getwifi_settings'
```

7.1.2 Wifi Einstellungen setzen

Command Typ: POST

Command: {"cmd": "set_WiFi", "ssid": "", "password": "", "active": true}

Response:
```
{"status": "ok"}
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/api'
--header 'Content-Type: text/plain' --data '{
"cmd": "set_WiFi", "ssid": "", "password": "", active": bool}'
```

7.1.3 Wifi statische IP setzen

Command Typ: POST

Command: {"cmd": "set_WiFi_staticip", "wifistaticip_active": bool, "wifistaticip": "0.0.0.0", "wifistaticsub": "0.0.0.0", "wifistaticgw": "0.0.0.0", "wifistaticdns": "0.0.0.0"}

Response:
```
{"status": "ok"}
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/api'
--header 'Content-Type: text/plain' --data '{
"cmd": "set_WiFi_staticip", "wifistaticip_active": true,
"wifistaticip": "192.168.1.10", "wifistaticsub":
"255.255.255.0", "wifistaticgw": "192.168.1.254",
"wifistaticdns": "192.168.1.254"}'
```

7.1.4 Wifi AP Passwort setzen

Command Typ: POST

Command: {"cmd": " set_WiFi_AP_password", ap_password": ""}

Response:
```
{"status": "ok"}
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/api'
--header 'Content-Type: text/plain' --data '{
"cmd": " set_WiFi_AP_password", "ap_password": "smartismart"}'
```

7.2 SmartMeter Konfiguration

7.2.1 SmartMeter Einstellungen abfragen

Command Typ: GET http://<ip-address>:8080/getsmartmeter_settings

Response:
```
{"type": "chipkey", "value": ""},
{"type": "powerconverteraktiv", "value": bool},
{"type": "owerconverterfactor","value": 0},
{"type": "selfreg", "value": bool}
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/getsmartmeter_settings'
```

7.2.2 SmartMeter letzte Messdaten abfragen

Command Typ: GET http://<ip-address>:8080/getmeterdata

Response:
```
{"OBIS": "0.0.1.0.0.255", "value": ""},
{"OBIS": "0.0.96.1.0.255", "value": ""},
{"OBIS": "0.0.42.0.0.255", "value": ""},
{"OBIS": "1.0.32.7.0.255", "value": 0,"unit": "V"},
{"OBIS": "1.0.52.7.0.255", "value": 0, "unit": "V"},
{"OBIS": "1.0.72.7.0.255", "value": 0, "unit": "V"},
{"OBIS": "1.0.31.7.0.255", "value": 0, "unit": "A"},
{"OBIS": "1.0.51.7.0.255", "value": 0, "unit": "A"},
{"OBIS": "1.0.71.7.0.255", "value": 0, "unit": "A"},
{"OBIS": "1.0.1.7.0.255", "value": 0, "unit": "kW"},
{"OBIS": "1.0.2.7.0.255", "value": 0, "unit": "kW"},
{"OBIS": "1.0.1.8.0.255", "value": 0, "unit": "kWh"},
….
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/getmeterdata'
```

7.2.3 SmartMeter Einstellungen setzen

Command Typ: POST

Command: {"cmd": " set_MeterData", "chipkey": "", "powerconverteraktiv": bool, "powerconverterfactor": integer}

Response:
```
{"status": "ok"}
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/api'
--header 'Content-Type: text/plain' --data '{
"cmd": "set_MeterData",
"chipkey": "0047886c34434b4e7136703643577a77",
"powerconverteraktiv": false, "powerconverterfactor": 0}'
```

7.3 MQTT-Konfiguration

7.3.1 MQTT Einstellungen abfragen

Command Typ: GET http://<ip-address>:8080/getmqtt_settings

Response:
```
{"type": "mqtt_broker_aktiv", "value": bool},
{"type": "mqtt_broker_server", "value": ""},
{"type": "mqtt_broker_user","value": ""},
{"type": "mqtt_broker_password", "value": ""},
{"type": "mqtt_broker_port", "value": integer}
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/getmqtt_settings'
```

7.3.2 MQTT Einstellungen setzen

Command Typ: POST

Command: {"cmd": "set_MQTT_BR", "mqtt_broker_aktiv": bool, "mqtt_broker_server": "0.0.0.0", "mqtt_broker_user": "", "mqtt_broker_pw": "", "mqtt_broker_port": integer}

Response:
```
{"status": "ok"}
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/api'
--header 'Content-Type: text/plain' --data '{
"cmd": "set_MQTT_BR", "mqtt_broker_aktiv": false,
"mqtt_broker_server": "192.168.0.10",
"mqtt_broker_user": "Username", "mqtt_broker_pw": "pw",
"mqtt_broker_port": 1883}'
```

7.4 UDP-Konfiguration

7.4.1 UDP Einstellungen abfragen

Command Typ: GET http://<ip-address>:8080/getloxoneudp_settings

Response:
```
{"type": "aktiv", "value": bool},
{"type": "udp_ip", "value": ""},
{"type": "udp_port", "value": integer}
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/getloxoneudp_setting'
```

7.4.2 UDP Einstellungen setzen

Command Typ: POST

Command: {"cmd": "set_LOXONEUDP", "udp_active": bool, "udp_ip": "0.0.0.0", "udp_port": integer}

Response:
```
{"status": "ok"}
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/api'
--header 'Content-Type: text/plain' --data '{
"cmd": "set_LOXONEUDP", "udp_active": true,
"udp_ip": "191.191.191.191", "udp_port": 8100}'
```

7.5 C.M.I. (Technische Alternative) Konfiguration

7.5.1 UDP C.M.I. Einstellungen abfragen

Command Typ: GET http://<ip-address>:8080/getcmi_settings

Response:
```
{"type": "aktiv", "value": bool},
{"type": "cmi_ip", "value": ""},
{"type": "cmi_knoten", "value": integer},
{"type": "cmi_shortpayload", "value": bool},
{"type": "cmi_mA", "value": bool}
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/getcmi_settings'
```

7.5.2 C.M.I. Einstellungen setzen

Command Typ: POST

Command: {"cmd": " set_CMI", "cmi_active": bool, "cmi_ip": "0.0.0.0", "cmi_knoten": integer, "cmi_shortpayload": bool, "cmi_mA": bool }

Response:
```
{"status": "ok"}
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/api'
--header 'Content-Type: text/plain' --data '{
"cmd": "set_CMI", "cmi_active": false,
"cmi_ip": "191.191.191.191", "cmi_knoten": 11,
"cmi_shortpayload": true, "cmi_mA": true}'
```

7.6 Send Data Filter

7.6.1 Send Data Filter Einstellungen abfragen

Command Typ: GET http://<ip-address>:8080/getsenddatafilter_settings

Response:
```
{"type": "0.0.1.0", "value": bool},
{"type": "0.0.96.1", "value": bool},
{"type": "0.0.42.0", "value": bool},
{"type": "1.0.32.7", "value": bool},
{"type": "1.0.52.7", "value": bool},
{"type": "1.0.72.7", "value": bool},
{"type": "1.0.31.7", "value": bool},
{"type": "1.0.51.7", "value": bool},
{"type": "1.0.71.7", "value": bool},
{"type": "1.0.1.7", "value": bool},
{"type": "1.0.2.7", "value": bool},
```

```
{"type": "1.0.1.8", "value": bool},
{"type": "1.0.2.8", "value": bool},
{"type": "1.0.3.8", "value": bool},
{"type": "1.0.4.8", "value": bool},
{"type": "1.0.13.7", "value": bool}
```

Bsp. curl: `curl --location 'http://<ip-address>:8080/getsenddatafilter_settings'`

7.6.2 Send Data Filter Einstellungen setzen

Command Typ: POST

Command: {"cmd": "set_SendDataFilter"", "0.0.1.0": bool, "0.0.96.1": bool, "0.0.42.0": bool, "1.0.32.7": bool, "1.0.52.7": bool, "1.0.72.7": bool, "1.0.31.7": bool, "1.0.51.7": bool, "1.0.71.7": bool, "1.0.1.7": bool, "1.0.2.7": bool, "1.0.1.8": bool, "1.0.2.8": bool, "1.0.3.8": bool, "1.0.4.8": bool, "1.0.13.7": bool}

Response: `{"status": "ok"}`

Bsp. curl:

```
curl --location 'http://<ip-address>:8080/api'
--header 'Content-Type: text/plain' --data '{
"cmd": " set_SendDataFilter", "0.0.1.0": false, "0.0.96.1": false,
"0.0.46.0": false, "1.0.32.7": false, "1.0.52.7": false, "1.0.72.7":
false, "1.0.31.7": false, "1.0.51.7": false, "1.0.71.7": false,
"1.0.1.7": false, "1.0.2.7": false, "1.0.1.8": false, "1.0.2.8":
false, "1.0.3.8": false, "1.0.4.8": false, "1.0.13.7": false}'
```

7.7 System Funktionen

7.7.1 AccessPoint Passwort setzen

Command Typ: POST

Command: {"cmd": "set_WiFi_AP_password", "ap_password": string }

Response: `{"status": "ok"}`

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/api'
--header 'Content-Type: text/plain' --data '{
"cmd": "set_WiFi_AP_password", "ap_password": "smartismart"}'
```

7.7.2 Restart

Command Typ: POST

Command: {"cmd": "restart"}

Response: `{"status": "ok"}`

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/api'
--header 'Content-Type: text/plain' --data '{
"cmd": "restart"}'
```

7.7.3 Factory reset

SMARTI auf Werkseinstellungen zurücksetzen. Alle Parameter werden gelöscht!!

Command Typ: POST

Command: {"cmd":"factoryreset"}

Response: `{"status": "ok"}`

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/api'
--header 'Content-Type: text/plain' --data '{
"cmd":"factoryreset"}'
```

7.7.4 Device Typ abfragen

Command Typ: GET http://<ip-address>:8080/getdevicetyp

Response:
```
[
 {"type": "devicetyp", "value": "SMARTI S3"},
 {"type": "firmwareversion", "value": "2.0.0 d"},
 {"type": "serialnumber", "value": "348518BD280C"}
]
```

Bsp. curl:
```
curl --location 'http://<ip-address>:8080/getdevicetyp
```