LAN GATEWAY INSTALLATION AND CONFIGURATION V1.21

This document is in draft stage. Subject to change without prior notice.

Version assignment		
b2l_Gateway_Demo_Server	Document	Firmware on Gateway
1.15	1.13	
1.16	1.14	1.21
2.0	1.18	1.45
2.0	1.19	1.45
2.0	1.20	1.45
2.0	1.21	1.45

Changelog	
Document	Changes
1.14	- Version assignment, changelog and API incompatibility list added - Chapter 8.9 „Scan nodes“ added. - "Scan nodes” added to the API in Chapter 11.1.13 - Chapter 9.3 “Database page of the server UI” added - “Enable Beaconing” added in Chapter 10.1, some minor corrections.
1.18	- Added description of DHCP and Bootloader-Mode. - Added description if “Gateway time”. - Some smaller corrections which are related to V 2.0. - Windows version temporarily not available in version 2.0.
1.19	- Added API note.
1.20	- some smaller error corrections in the API documentation and in the document.
1.21	- Improved the description of the update process. - Corrections in the description of the API.

API incompatibility list	
Firmware	API command changes

CONTENT

1 WARRANTY, WARNINGS, LIMITATIONS

VS does not provide any warranty. Also, the document may change without prior notice. Vossloh-Schwabe is not responsible for any kind of usage of the software, especially Vossloh-Schwabe is/does not:

- Provide warranty on data loss
- Care about necessary network security
- Provide data protection
- Provide bug-free code

The user of the documentation and the user of the final installation are responsible that all legal requirements in the country where this is used are fulfilled. Especially requirements on data protection, security must be fulfilled.

1.1 LICENCE OF THE DEMO SOFTWARE

The demo software is provided according to the MIT-License. This is the granted license therefore:

2 THERMES USED IN THIS DOCUMENT

VS	Vossloh-Schwabe Deutschland GmbH
PSK	Pre-shared key

3 INTRODUCTION

VS provides a Gateway (187055) to connect a Blu2Light system to ethernet. This enables the integration of a Blu2Light system in various kinds of applications.

To enable the integration Vossloh-Schwabe provides a demonstration written in Python™ running for example on a Raspberry Pi.

The demonstration is easy to setup and builds a basic for all different variants of implementation.

3.1 WHY PYTHON™?

The LAN Gateway software is written in Python™ because it's a common high-level interpreted programming language. One of its advantages is, that it runs on every common OS. It's also easy to read and it is open source.

4 INSTALLATION AND CONFIGURATION OF THE VS DEMO SOFTWARE

All required installation files can be found here:

https://www.vossloh-schwabe.com/en/products/light-management-indoor/blu2light-iot-devices/blu2light-gateway

The VS demo software is intended to be installed on the stock Raspberry Pi OS. Please follow the instructions provided at https://www.raspberrypi.com/software/ how to basically set up a new Raspberry Pi OS and gain SSH login.

The demo software will show:

- How to establish a connection between the server and the Gateway
- How to receive and decode incoming mesh events
- Store the incoming events in a database.
- Visualize incoming data in Grafana.
- Provide an easy Web-based UI to control functional groups in a Blu2Light system.
- Read out the systems PMD data (if provided by the DALI drivers)

4.1 INSTALLATION ON RASPBERRY PI

1. Download the latest version of "Raspberry Pi OS" or "Raspberry Pi OS lite" (32 Bit or 64 Bit) on a SD card. The instructions therefore and all required downloads can be found here: https://www.raspberrypi.com/software/.
   Tip: for installation with "Raspberry Pi Imager" press **CTRL + SHIFT + X** for advanced settings like time zone, SSH or WIFI. It's important to set the time zone on the Raspberry Pi correctly, because the server time will be sent into the B2L system.
2. Copy the provided .zip file to the Raspberry Pi and to the desired installation path.
   - Method 1 (recommended): 'wget https://www.vossloh-schwabe.com/fileadmin/perfion/files/b2l_gateway_Demo_server_2.0.zip'
   - Method 2: if you use "Raspberry Pi OS" with a graphical interface, you can download the zip file directly.
   - Method 3: use an USB drive.
   - Method 4: use other file transfer methods, like Samba or SCP.
3. Extract the contents of the file with 'sudo *unzip <filename>*'.
4. Go into the extracted folder and run '*sudo chmod* +x *setup.sh*' to make setup.sh executable.
5. Run '*sudo ./setup.sh*' to start the setup routine.
   While the setup is running, you will be asked which database system you want to use (MariaDB or InfluxDB) and for the credentials and the database name, so that the database and the server can be configured. *Attention:* If you use MariaDB-database, please do not use the user "root" as a username for the MariaDB-database. This will cause a malfunction of the VS LAN Gateway software after every start and the device will not work properly. Please do as well only use letters and numbers and no special characters like "space" or "brackets".
   The setup script updates all packages on the Raspberry Pi and installs the necessary packages for the server.
6. After the setup has been run successfully, it will show you the hostname / IP of your devices. Note down this information carefully.
7. After the setup has finished, you can now enable the server service to start on every reboot of the Raspberry Pi by the following command '*sudo systemctl enable lan-gateway*'
8. After that, start the server service the first time with '*sudo systemctl start lan-gateway*'.

4.2 INSTALLATION ON WINDOWS SERVER

The Windows-server version is temporarily not available in version 2.0.

1. Download and unzip the installation files.
2. Start PowerShell 7 (or higher) as Administrator.
   Type in '& "*[path]**setup.ps1*"' f.e.: '& "*T:**media converter**setup.ps1*"'

4.3 UPDATE OF THE VS DEMO SOFTWARE ON RASPBERRY PI

1. Copy the provided .zip file to the Raspberry Pi and to the desired installation path.
   - Method 1: 'wget https://www.vossloh-schwabe.com/fileadmin/perfion/files/b2l_gateway_Demo_server_2.0.zip' (recommended).
   - Method 2: If you use "Raspberry Pi OS" with a graphical interface, you can download the zip file directly.
   - Method 3: Use an USB drive.
   - Method 4: Use another file transfer methods, like Samba or SCP.
2. Extract the contents of the file with '*unzip <filename>*'.
3. Go into the extracted folder and run '*sudo chmod +x update.sh*' to make update.sh executable.
4. Run '*sudo ./update.sh*' to start the update routine.
   The update script updates all packages on the Raspberry Pi and updates the necessary packages for the server.
5. After the update has been finished successfully, it will show you the hostname / IP of your devices. Note down this information carefully.
6. After the successful update, start the server service with '*sudo systemctl start lan-gateway*'.

4.4 CONFIGURATION

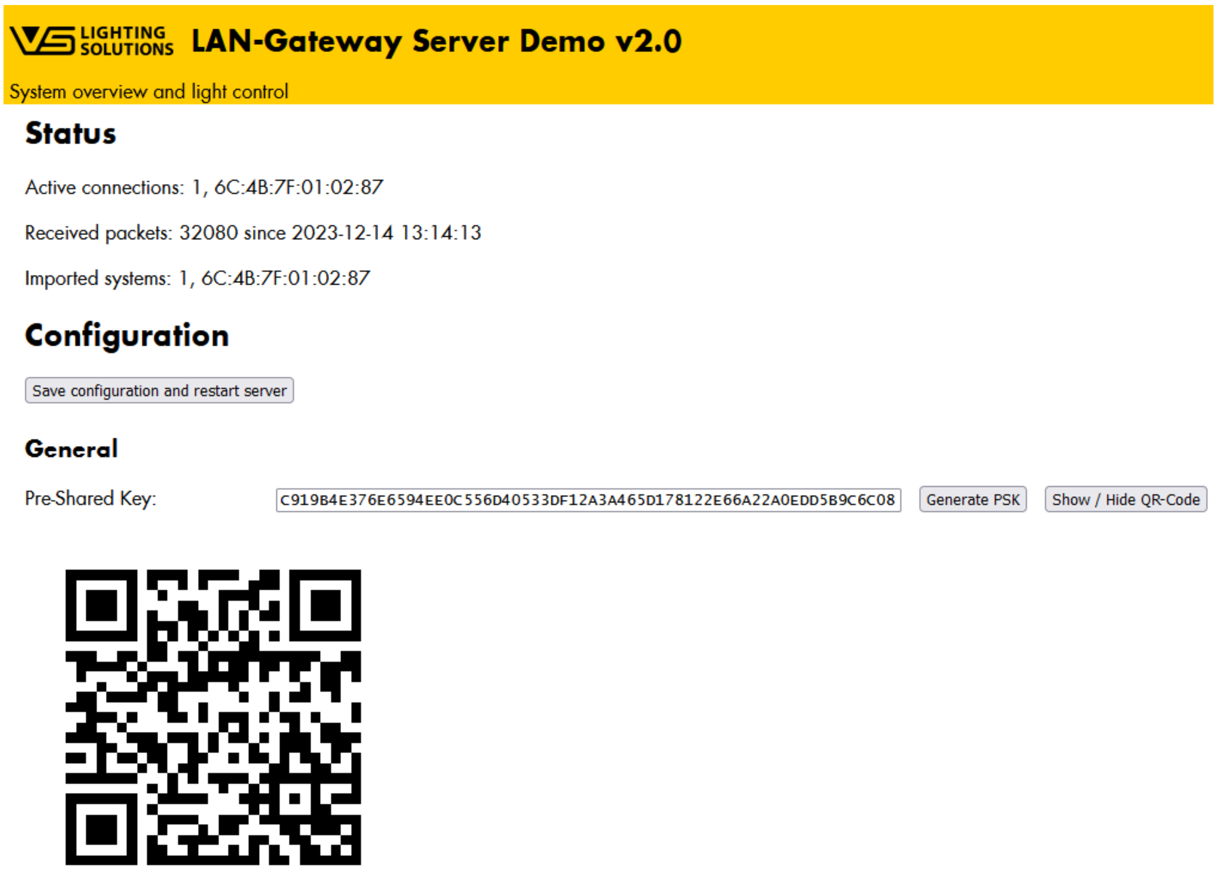
1. All configuration of the server can be done via the web interface, which is accessible at 'http://<IP or domain of the device>:31460/'.
2. Before the LAN Gateway can connect to the server, a pre-shared key (PSK) must be generated via the web interface. For simplicity a QR code with the PSK can be created, which can be scanned and copied (f.e. with the iPad camera app). Afterwards, the IP address, port and pre-shared key must be configured to the LAN Gateway via the LiNA Connect app. Therefore, the Gateway must have been added to a system in the LiNA Connect app and the system must be in expert mode. When everything was configured correctly, LED1 on the LAN Gateway while shine green.

The pre-shared key is generated in the web interface of the raspberry pi.

(http://<IP or domain of the device>:31460).

The pre-shared key is used to encrypt the communication between the Raspberry Pi and the Blu2Light LAN Gateway.

The following screenshot shows the generating of the key. By clicking on the field "Generate PSK" a pre-shared key is generated randomly.


LAN-Gateway Server Demo v2.0

System overview and light control

Status

Active connections: 1, 6C:4B:7F:01:02:87

Received packets: 32080 since 2023-12-14 13:14:13

Imported systems: 1, 6C:4B:7F:01:02:87

Configuration

General

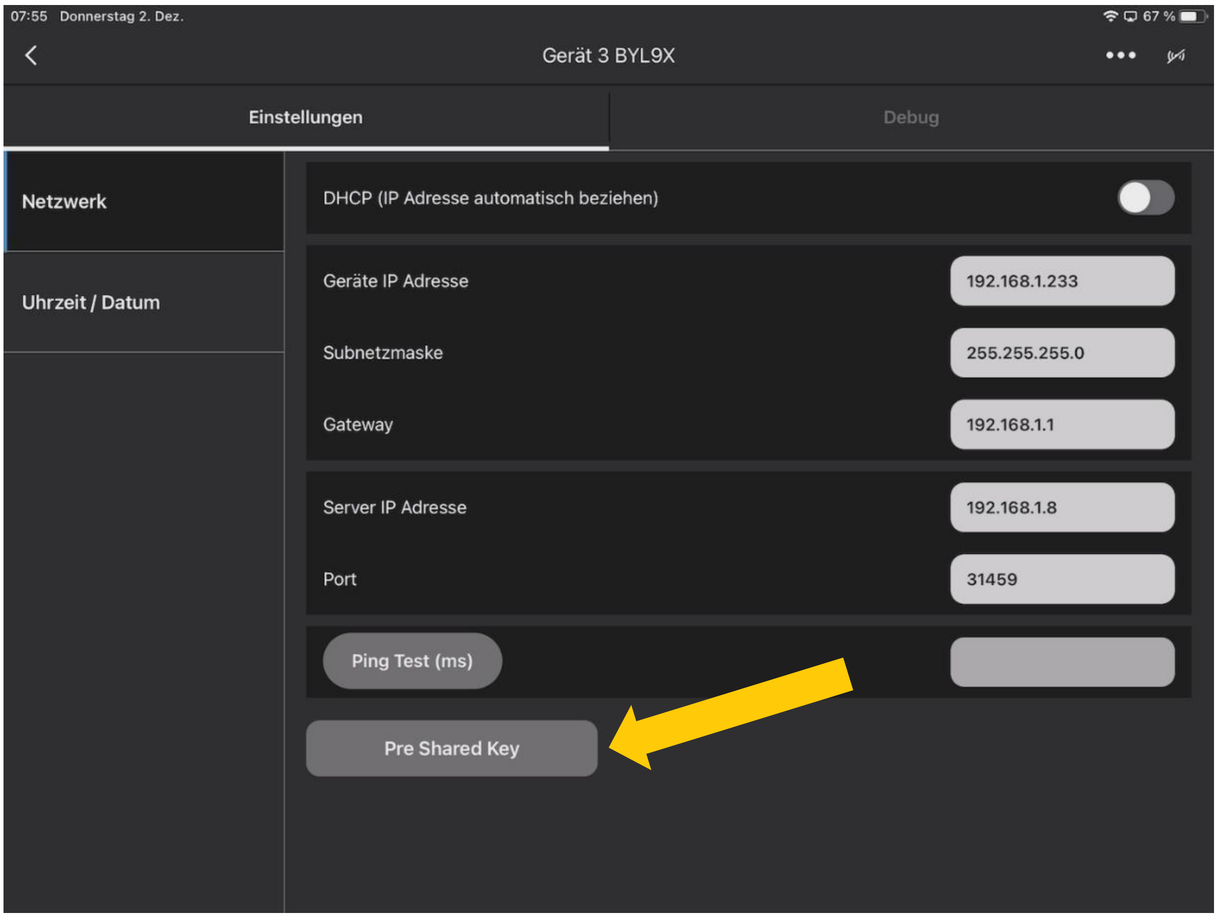
Pre-Shared Key: C919B4E376E6594EE0C556D40533DF12A3A465D178122E66A22A0EDD5B9C6C08

Figure 1: Generating the pre-shared key in the web interface of the Raspberry PI

The key must now be copied into the appropriate field in the LiNA Connect app. For that, go into the network settings of the Gateway and choose "pre-shared key".

If an additional Blu2Light LAN Gateway will be used, a click on "Show as QR code" will open the suitable QR code for the current server, which has already been generated before. The code can then be assigned to the new Blu2Light LAN Gateway that shall be added to this server.

It is important to know, that only one "pre-shared key" exists or can exist for a server. At the end, all connected Blu2Light LAN Gateways connected to the same server must have the same "pre-shared key" assigned.


07:55 Donnerstag 2. Dez.

Gerät 3 BYL9X

Einstellungen | Debug

Netzwerk

Uhrzeit / Datum

DHCP (IP Adresse automatisch beziehen)

Geräte IP Adresse: 192.168.1.233

Subnetzmaske: 255.255.255.0

Gateway: 192.168.1.1

Server IP Adresse: 192.168.1.8

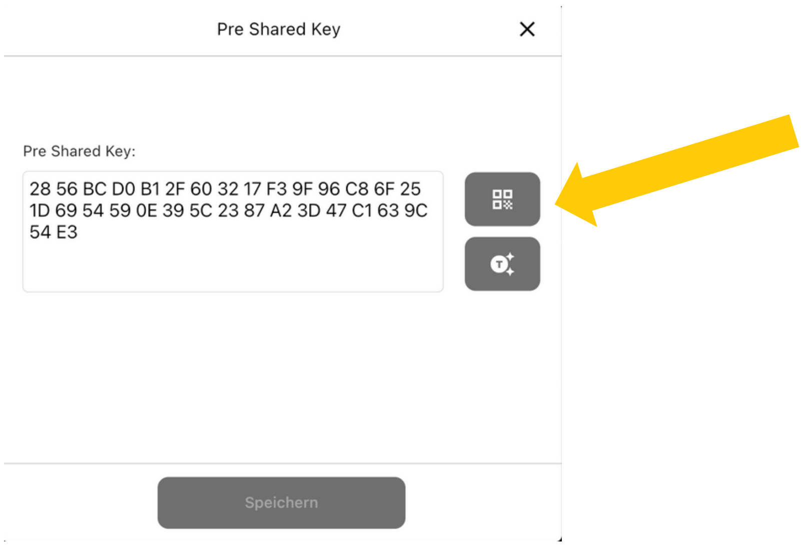
Port: 31459

Ping Test (ms)

Pre Shared Key

Figure 2: "pre-shared key" button for adding the generated pre-shared key from the raspberry pi.

Now select the QR symbol to scan the PSK from your screen and then press "save".


Pre Shared Key

Pre Shared Key:

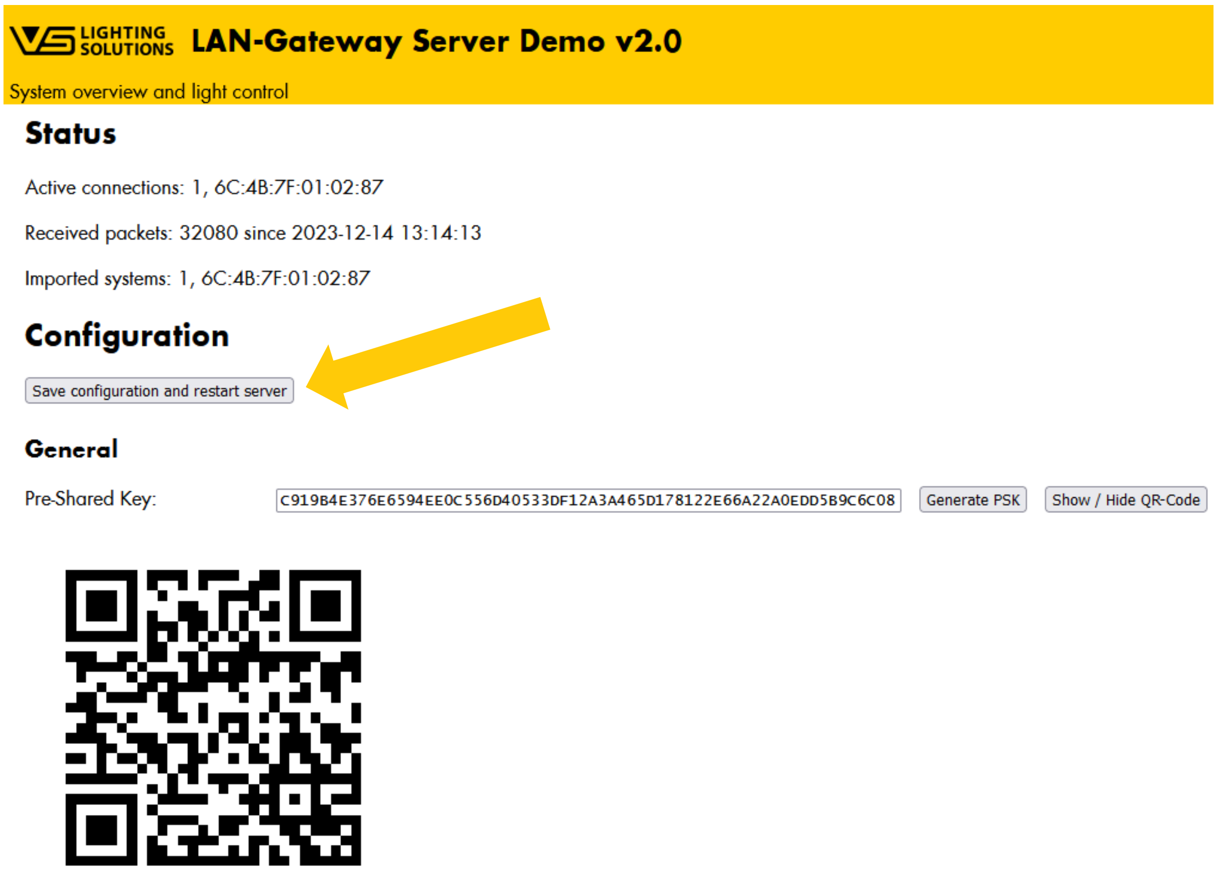
28 56 BC D0 B1 2F 60 32 17 F3 9F 96 C8 6F 25 1D 69 54 59 0E 39 5C 23 87 A2 3D 47 C1 63 9C 54 E3

Speichern

Figure 3: Add the PSK to the App

By clicking on the symbol pointed with the yellow arrow in figure 3, the code can be scanned as a QR code as well.

After the pre-shared key has been copied to LiNA Connect, it shall be saved in the web interface of the Raspberry Pi:


LAN-Gateway Server Demo v2.0

System overview and light control

Status

Active connections: 1, 6C:4B:7F:01:02:87

Received packets: 32080 since 2023-12-14 13:14:13

Imported systems: 1, 6C:4B:7F:01:02:87

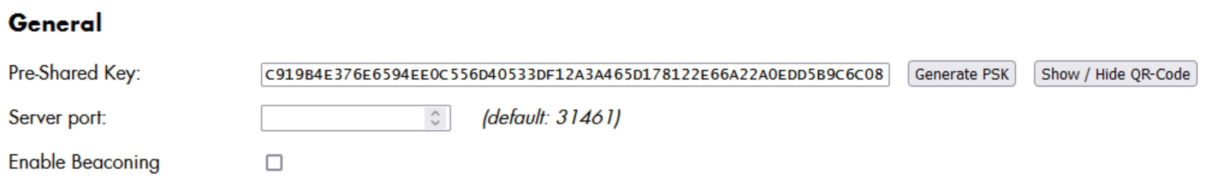
Configuration

General

Pre-Shared Key: C919B4E376E6594EE0C556D40533DF12A3A465D178122E66A22A0EDD5B9C6C08

After this step has been done successfully, the LED 1 on the Blu2Light LAN Gateway shall be green.

The pre-shared key will be visible in the web configuration. If needed at a later point it can be taken out of the following field:


General

Pre-Shared Key: C919B4E376E6594EE0C556D40533DF12A3A465D178122E66A22A0EDD5B9C6C08

Server port: (default: 31461)

Enable Beaconing

Figure 4: pre-shared key – visible in the web configuration

Additional Information: The used encryption method for the communication between server and LAN Gateway is TLSv1.2 ECDHE-PSK-CHACHA20-POLY1305.

4.5 CONFIGURATION PAGE

The demo provides a configuration and control interface via WEB browser. Navigate to ‘http://<IP or domain of the device>:31460’ to access the configuration.

4.6 GRAFANA LOGIN

The demo software will create a template of Grafana screens depending on your system. To access Grafana, go to ‘**http://<IP or domain of the device>:3000**’ and login with the default credentials admin / admin. You will be forced to change your password at the first login.

4.7 DATA STORAGE IN THE DEMO

The configuration of the demo software is stored in the following location:

Raspberry Pi OS:

    /var/vs_lan_gateway_server/config.json

Windows server:

    C:/Users/All Users/.vs_lan_gateway_server/config.json

If you have selected MariaDB, the database is located here:

Raspberry Pi OS:

    /var/lib/mysql

Windows server:

    C:/Users/All Users/scoop/persist/mariadb/data

For more information about MariaDB: https://mariadb.org/

The pre-defined templates for Grafana are located here:

Raspberry Pi OS:

    /var/vs_lan_gateway_server/dashboards

Windows server:

    C:/Users/All Users/.vs_lan_gateway_server/dashboards

Some additional data for Grafana are located here:

Raspberry Pi OS:

    /etc/grafana/provisioning/datasources
    /etc/grafana/provisioning/dashboards

Windows server:

    C:/Users/All Users/scoop/persist/grafana/conf/provisioning/datasources
    C:/Users/All Users/scoop/persist/grafana/conf/provisioning/dashboards

For more information about Grafana, and how graphical panels are created:
https://grafana.com/

4.8 EXPECTED DATABASE SIZE

For every B2L device added to the system, the user should expect about 2-10 Mbyte of data per month.

Make sure that you have enough memory or add a cleaning function.

4.9 BACKUP

To prevent data loss, it is generally recommended to frequently backup your data. Beside the basic data from the OS, make sure the data locations mentioned in paragraph 4.7 are also backed up.

4.10 SAMPLE CONFIGURATION OF THE ACTUAL WORKING SYSTEM

To get the sample configuration running, a backup of the B2L system must have been created in the LiNA Connect app. For that, go to the "…" menu in the system overview.

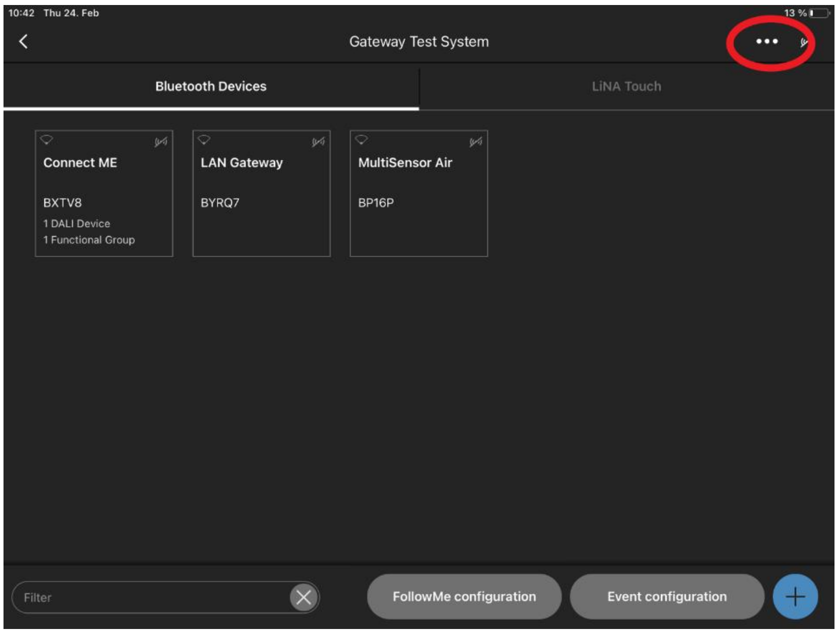


Figure 5: **Option-***menu in LiNA Connect*

Select "Backup/Restore":

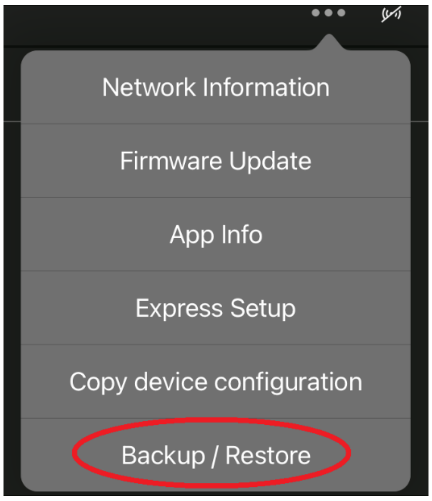
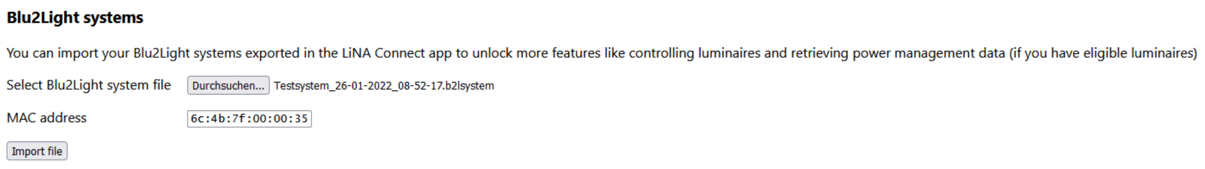


Figure 6: Backup / Restore -selection in LiNA Connect

Now go to "Create new backup/Export current configuration". If desired, give the system a new name. Now, a new backup on the tablet can be created, by tapping on "Create new backup", but to import the system to the Gateway web interface the system needs get exported to the device, where the web interface is opened. Therefore, go to "Export of system configuration" and then "Share". Now the backup can be uploaded to the cloud or sent to an email address.

Now go back to the web interface and upload the backup with the MAC address of the Blu2Light LAN Gateway.



Blu2Light systems

You can import your Blu2Light systems exported in the LiNA Connect app to unlock more features like controlling luminaires and retrieving power management data (if you have eligible luminaires)

Select Blu2Light system file Durchsuchen... Testsystem_26-01-2022_08-52-17.b2lsystem

MAC address 6c:4b:7f:00:00:35

Import file

Figure 7: Import of the recorded backup-file in the web configuration.

After clicking on "import file" a message shall appear, that the configuration has been imported successfully. If the message has been appeared, a login to Grafana (http:// < IP or domain>: 3000) can be done. Please keep in mind that the first login to Grafana is only successful with "admin" as user and as well as password.

You will be forced to change the password directly after the first login has been successful. After the password has been changed, you will be forwarded to the welcome page.

The “imported system” shall be visible in the Grafana environment. When you login the first time you will have to click on the upper left corner on “manage”. The system shall become visible:

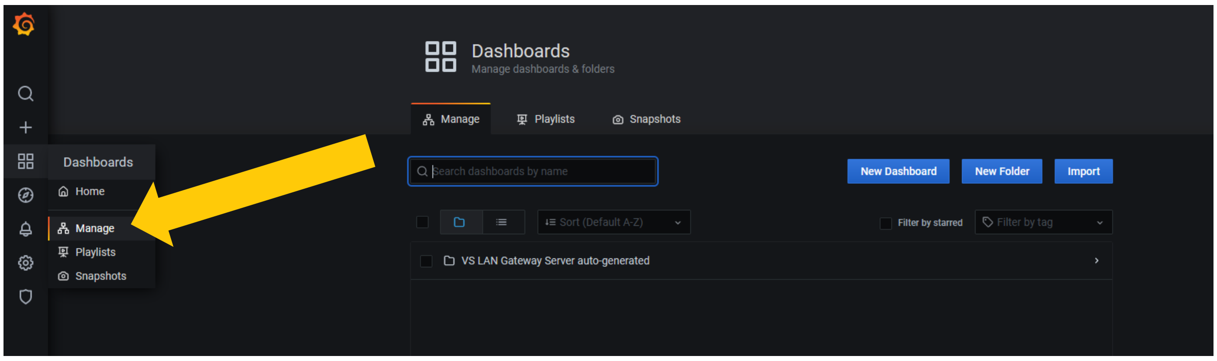
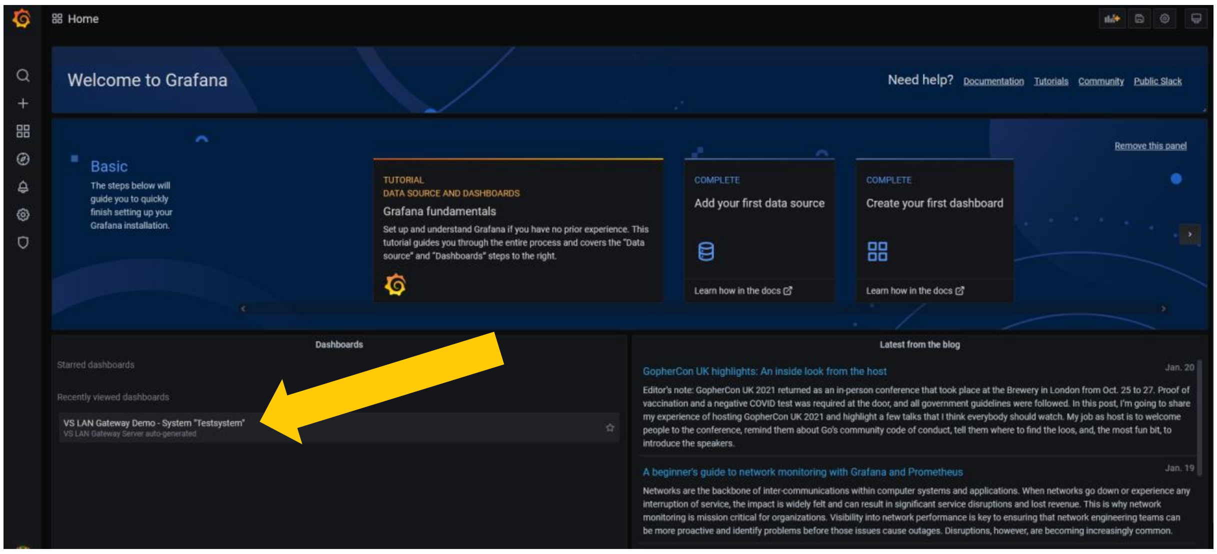


Figure 8: Dashboard-settings – the automatic generated dashboard shall be visible in the middle after a click on “Dashboards”

After the 2nd login, the dashboard should appear on the starting page in Grafana:


Home
Welcome to Grafana
Need help? Documentation Tutorials Community Public Slack
Remove this panel
Basic
The steps below will guide you to quickly finish setting up your Grafana installation.
TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals
Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the “Data source” and “Dashboards” steps to the right.
COMPLETE
Add your first data source
Learn how in the docs
COMPLETE
Create your first dashboard
Learn how in the docs
Dashboards
Starred dashboards
Recently viewed dashboards
VS LAN Gateway Demo - System "Testsystem"
VS LAN Gateway Server auto-generated
Latest from the blog

GopherCon UK highlights: An inside look from the host — Jan. 20

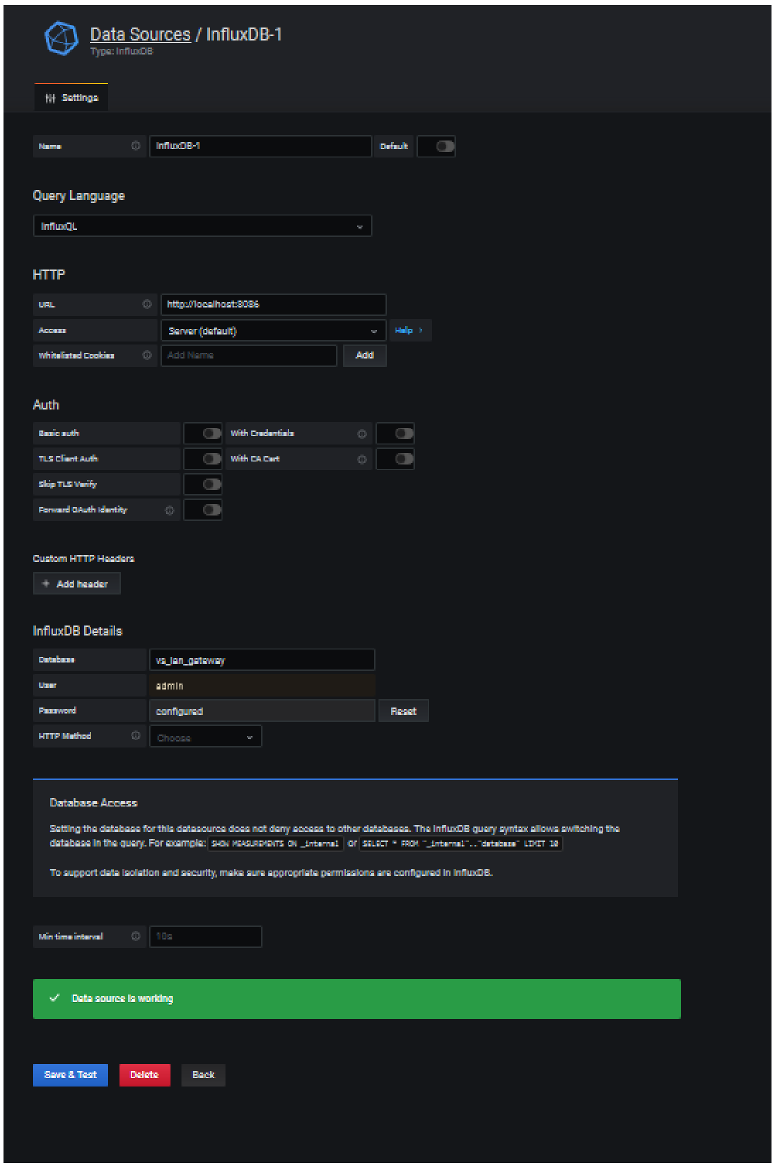
Editor's note: GopherCon UK 2021 returned as an in-person conference that took place at the Brewery in London from Oct. 25 to 27. Proof of vaccination and a negative COVID test was required at the door, and all government guidelines were followed. In this post, I'm going to share my experience of hosting GopherCon UK 2021 and highlight a few talks that I think everybody should watch. My job as host is to welcome people to the conference, remind them about Go's community code of conduct, tell them where to find the loos, and, the most fun bit, to introduce the speakers.

A beginner's guide to network monitoring with Grafana and Prometheus — Jan. 19

Networks are the backbone of inter-communications within computer systems and applications. When networks go down or experience any interruption of service, the impact is widely felt and can result in significant service disruptions and lost revenue. This is why network monitoring is mission critical for organizations. Visibility into network performance is key to ensuring that network engineering teams can be more proactive and identify problems before those issues cause outages. Disruptions, however, are becoming increasingly common.

Figure 9: The imported „system” shall be visible in Grafana after a successful import.

When the InfluxDB is installed, the following settings must be applied:



Data Sources / InfluxDB-1
Type: InfluxDB

Settings

Name: InfluxDB-1
Default: off

Query Language
InfluxQL

HTTP
URL: http://localhost:8086
Access: Server (default)
Whitelisted Cookies: Add Name

Auth
Basic auth: off
With Credentials: off
TLS Client Auth: off
With CA Cert: off
Skip TLS Verify: off
Forward OAuth Identity: off

Custom HTTP Headers

InfluxDB Details
Database: vs_lan_gateway
User: admin
Password: configured
HTTP Method: Choose

Database Access

Setting the database for this datasource does not deny access to other databases. The InfluxDB query syntax allows switching the database in the query. For example: `SHOW MEASUREMENTS ON _internal` or `SELECT * FROM "_internal".."database" LIMIT 10`

To support data isolation and security, make sure appropriate permissions are configured in InfluxDB.

Min time interval: 10s

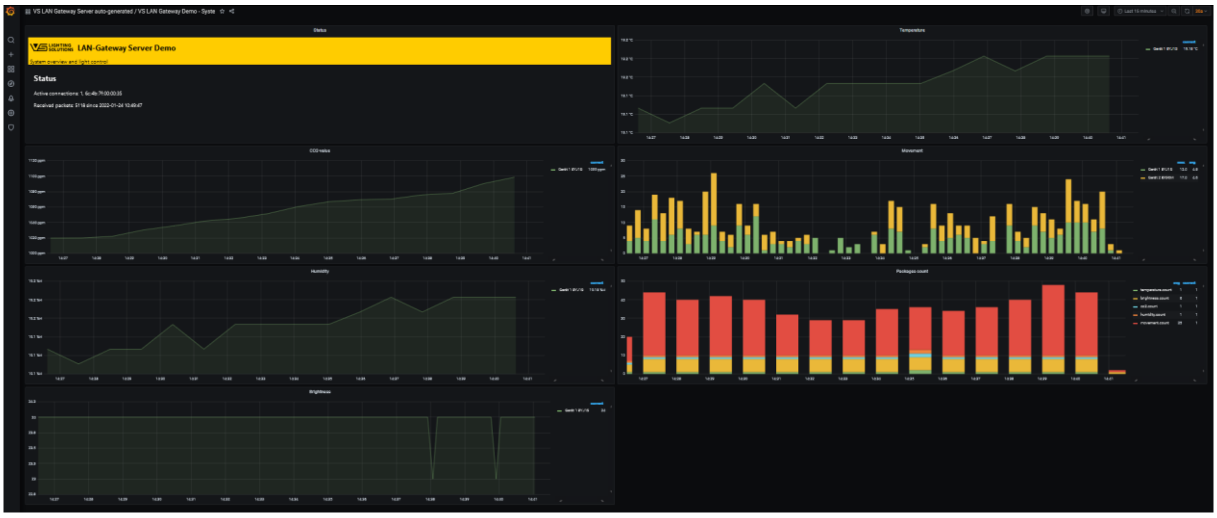
✓ Data source is working

Figure 10: Settings when using influxDB

The database name must be applied. It is “vs_lan_gateway”. As well fill in the “URL”: http://localhost:8086.

The database is functional and after clicking on “Save & Test”, the information, which is given out, shall be: “Data source is working”.

If you click on the dashboard, the demo site shall be visible where the incoming data can be viewed:


VS LAN Gateway Server auto-generated / VS LAN Gateway Demo - Syste
Status
LAN-Gateway Server Demo
System overview and light control
Temperature
CO2 value
Movement
Humidity
Packages count
Brightness

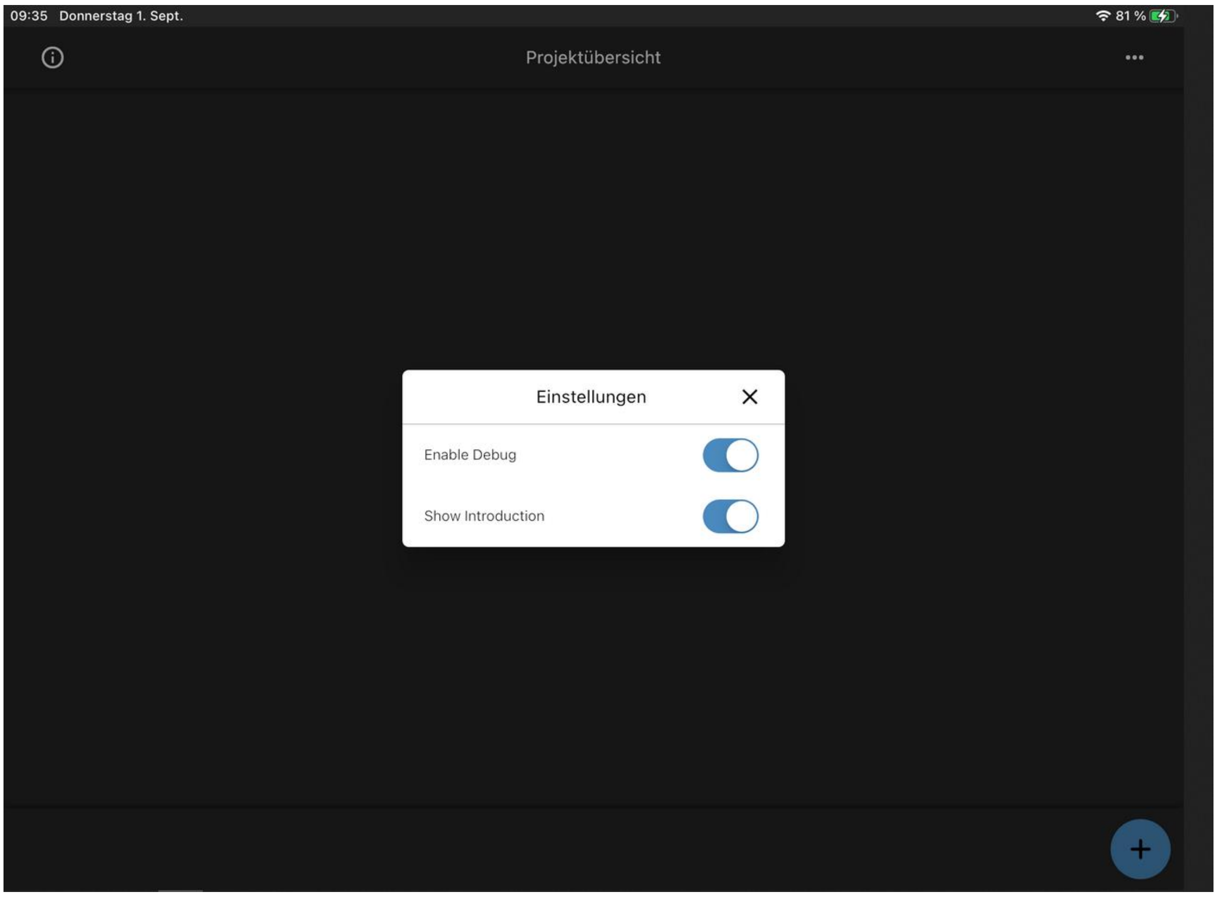
Status

Active connections: 1, 6c:4b:7f:00:00:35

Received packets: 5118 since 2022-01-24 10:49:47

Figure 11: Example site of the imported system.

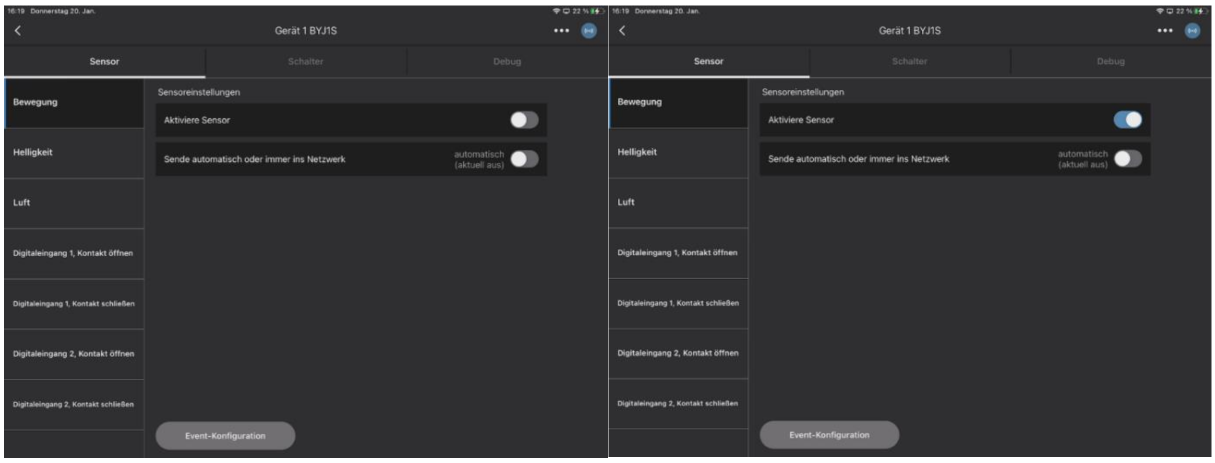
You can adjust all settings, add, or delete a dashboard and its content as you wish and desire. To this end the “debug-mode” in LiNA Connect must be enabled to access this feature in the settings of the Blu2Light Multisensor AIR:



Enable Debug
Show Introduction

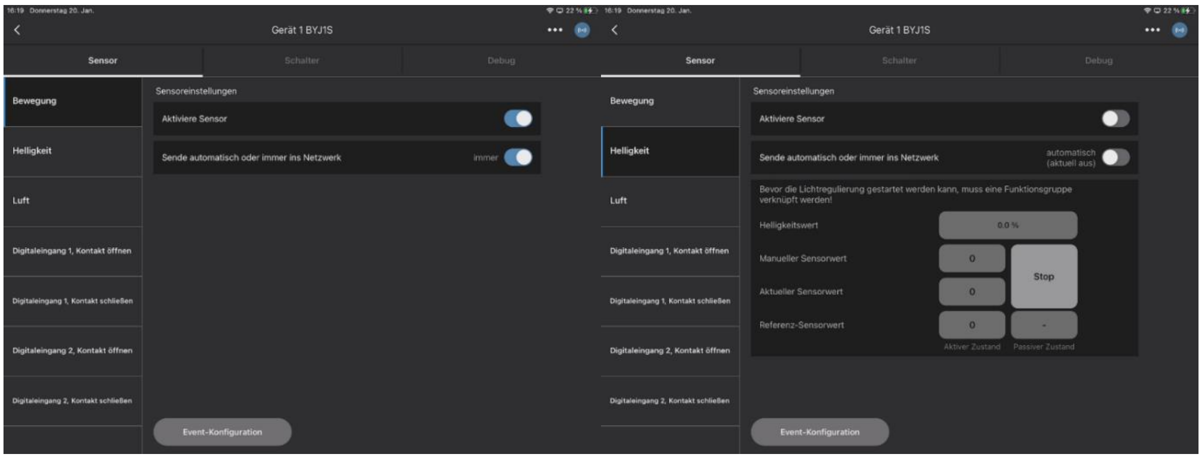
Figure 12: Enabling the „debug-mode" in LiNA Connect

There is one important thing to prepare before you can see data like shown in the figure above. The interval of the incoming data must be set up at the corresponding device. The following pictures show the necessary steps with the LiNA Connect App for a Blu2Light Multisensor AIR.


16:19 Donnerstag 20. Jan.
Gerät 1 BYJ1S
Sensor
Schalter
Debug
Bewegung
Helligkeit
Luft
Digitaleingang 1, Kontakt öffnen
Digitaleingang 1, Kontakt schließen
Digitaleingang 2, Kontakt öffnen
Digitaleingang 2, Kontakt schließen
Event-Konfiguration

Sensoreinstellungen
Aktiviere Sensor
Sende automatisch oder immer ins Netzwerk — automatisch (aktuell aus)

Figure 13: Step 1 (left) and step 2 (right) – activating the motion sensor


Gerät 1 BYJ1S

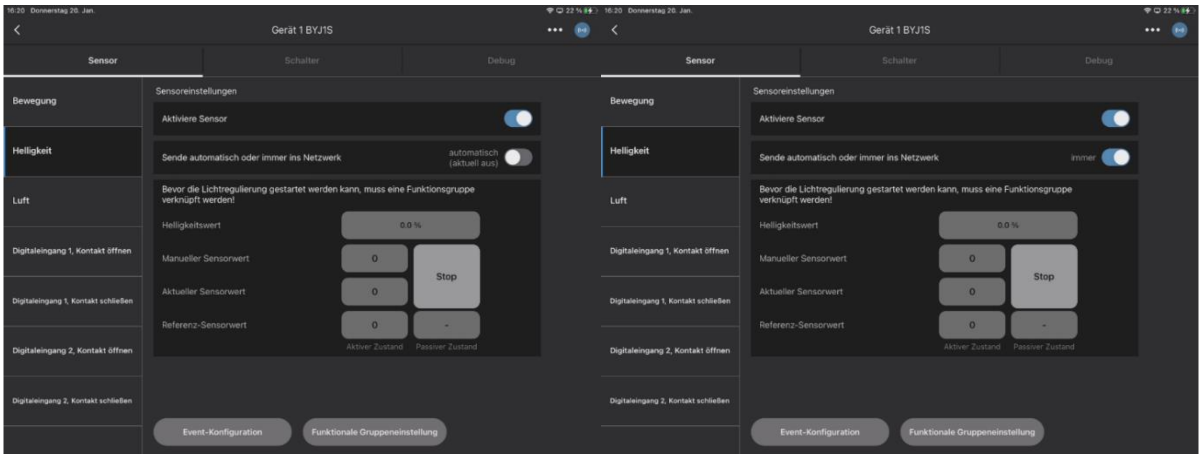
Sensor | Schalter | Debug

Bewegung
Helligkeit
Luft
Digitaleingang 1, Kontakt öffnen
Digitaleingang 1, Kontakt schließen
Digitaleingang 2, Kontakt öffnen
Digitaleingang 2, Kontakt schließen

Sensoreinstellungen
Aktiviere Sensor
Sende automatisch oder immer ins Netzwerk — immer
Event-Konfiguration

Sensoreinstellungen
Aktiviere Sensor
Sende automatisch oder immer ins Netzwerk — automatisch (aktuell aus)
Bevor die Lichtregulierung gestartet werden kann, muss eine Funktionsgruppe verknüpft werden!
Helligkeitswert — 0.0 %
Manueller Sensorwert — 0
Aktueller Sensorwert — 0
Referenz-Sensorwert — 0 — -
Stop
Aktiver Zustand — Passiver Zustand
Event-Konfiguration

Figure 14: Step 3 (left) – activating “send always to mesh” of the motion sensor and step 4 (right) – brightness menu.


Gerät 1 BYJ1S

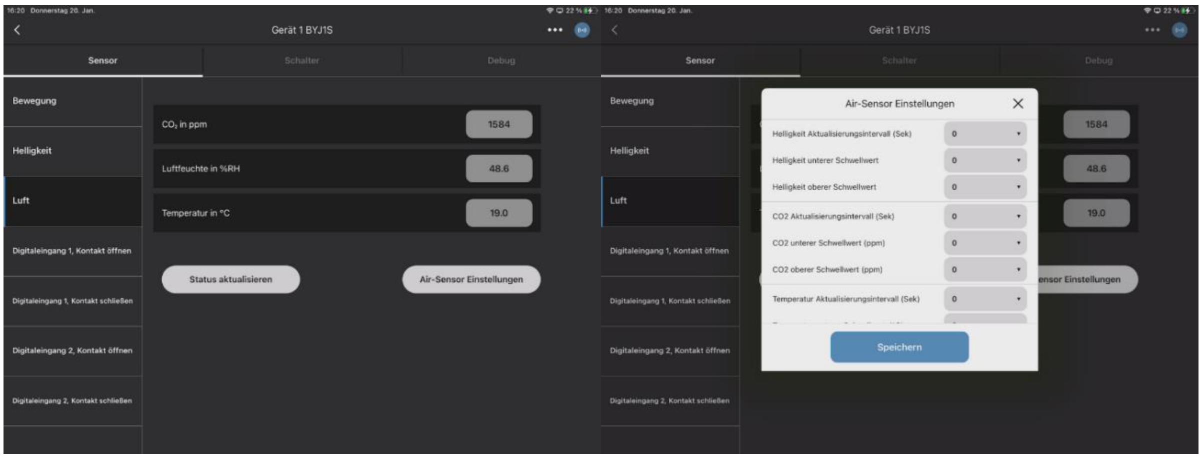
Sensor | Schalter | Debug

Bewegung
Helligkeit
Luft
Digitaleingang 1, Kontakt öffnen
Digitaleingang 1, Kontakt schließen
Digitaleingang 2, Kontakt öffnen
Digitaleingang 2, Kontakt schließen

Sensoreinstellungen
Aktiviere Sensor
Sende automatisch oder immer ins Netzwerk — automatisch (aktuell aus)
Bevor die Lichtregulierung gestartet werden kann, muss eine Funktionsgruppe verknüpft werden!
Helligkeitswert — 0.0 %
Manueller Sensorwert — 0
Aktueller Sensorwert — 0
Referenz-Sensorwert — 0 — -
Stop
Aktiver Zustand — Passiver Zustand
Event-Konfiguration — Funktionale Gruppeneinstellung

Sensoreinstellungen
Aktiviere Sensor
Sende automatisch oder immer ins Netzwerk — immer
Bevor die Lichtregulierung gestartet werden kann, muss eine Funktionsgruppe verknüpft werden!
Helligkeitswert — 0.0 %
Manueller Sensorwert — 0
Aktueller Sensorwert — 0
Referenz-Sensorwert — 0 — -
Stop
Aktiver Zustand — Passiver Zustand
Event-Konfiguration — Funktionale Gruppeneinstellung

Figure 15: Step 5 (left) – activating the brightness sensor and step 6 (right) – activating “send always to mesh” of the brightness sensor.


Gerät 1 BYJ1S

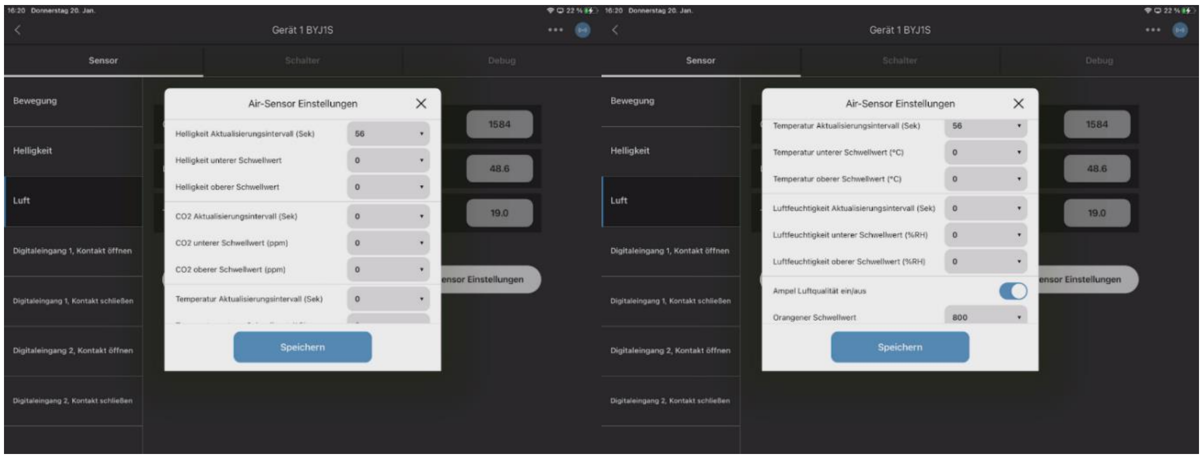
Sensor | Schalter | Debug

Bewegung
Helligkeit
Luft
Digitaleingang 1, Kontakt öffnen
Digitaleingang 1, Kontakt schließen
Digitaleingang 2, Kontakt öffnen
Digitaleingang 2, Kontakt schließen

CO₂ in ppm — 1584
Luftfeuchte in %RH — 48.6
Temperatur in °C — 19.0
Status aktualisieren — Air-Sensor Einstellungen

Air-Sensor Einstellungen
Helligkeit Aktualisierungsintervall (Sek) — 0
Helligkeit unterer Schwellwert — 0
Helligkeit oberer Schwellwert — 0
CO2 Aktualisierungsintervall (Sek) — 0
CO2 unterer Schwellwert (ppm) — 0
CO2 oberer Schwellwert (ppm) — 0
Temperatur Aktualisierungsintervall (Sek) — 0
Speichern

Figure 16: Step 6 (left) –and step 7 (right) – setting the data interval for data from Blu2Light Multisensor AIR.


16:20 Donnerstag 20. Jan.
Gerät 1 BYJ1S
Sensor
Schalter
Debug
Bewegung
Helligkeit
Luft
Digitaleingang 1, Kontakt öffnen
Digitaleingang 1, Kontakt schließen
Digitaleingang 2, Kontakt öffnen
Digitaleingang 2, Kontakt schließen
1584
48.6
19.0
ensor Einstellungen
Speichern

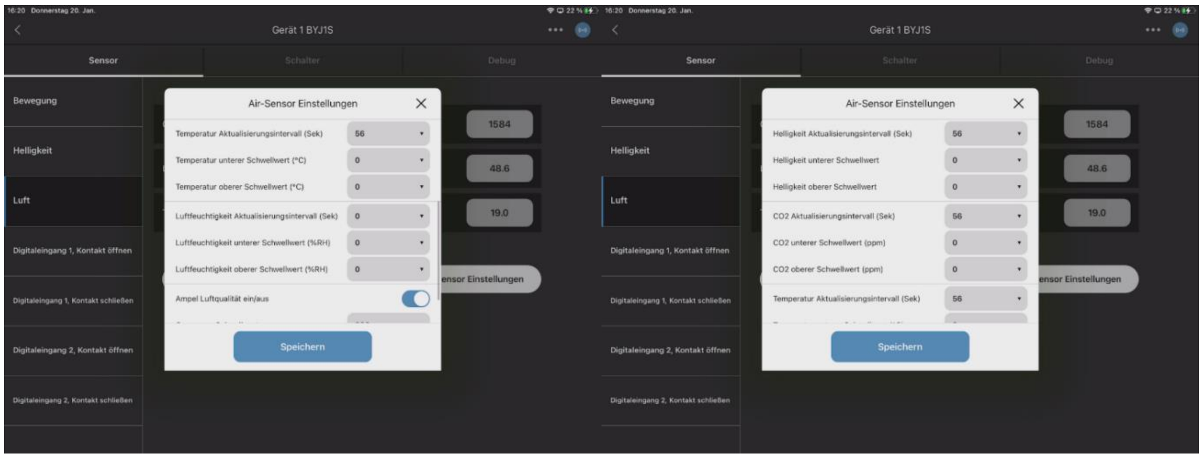
Air-Sensor Einstellungen

Helligkeit Aktualisierungsintervall (Sek) 56
Helligkeit unterer Schwellwert 0
Helligkeit oberer Schwellwert 0
CO2 Aktualisierungsintervall (Sek) 0
CO2 unterer Schwellwert (ppm) 0
CO2 oberer Schwellwert (ppm) 0
Temperatur Aktualisierungsintervall (Sek) 0

Air-Sensor Einstellungen

Temperatur Aktualisierungsintervall (Sek) 56
Temperatur unterer Schwellwert (°C) 0
Temperatur oberer Schwellwert (°C) 0
Luftfeuchtigkeit Aktualisierungsintervall (Sek) 0
Luftfeuchtigkeit unterer Schwellwert (%RH) 0
Luftfeuchtigkeit oberer Schwellwert (%RH) 0
Ampel Luftqualität ein/aus
Orangener Schwellwert 800

Figure 17: Step 7 (left) –and step 8 (right) – setting the data interval for data from Blu2Light Multisensor AIR.


16:20 Donnerstag 20. Jan.
Gerät 1 BYJ1S
Sensor
Schalter
Debug
Bewegung
Helligkeit
Luft
Digitaleingang 1, Kontakt öffnen
Digitaleingang 1, Kontakt schließen
Digitaleingang 2, Kontakt öffnen
Digitaleingang 2, Kontakt schließen
1584
48.6
19.0
ensor Einstellungen
Speichern

Air-Sensor Einstellungen

Temperatur Aktualisierungsintervall (Sek) 56
Temperatur unterer Schwellwert (°C) 0
Temperatur oberer Schwellwert (°C) 0
Luftfeuchtigkeit Aktualisierungsintervall (Sek) 0
Luftfeuchtigkeit unterer Schwellwert (%RH) 0
Luftfeuchtigkeit oberer Schwellwert (%RH) 0
Ampel Luftqualität ein/aus

Air-Sensor Einstellungen

Helligkeit Aktualisierungsintervall (Sek) 56
Helligkeit unterer Schwellwert 0
Helligkeit oberer Schwellwert 0
CO2 Aktualisierungsintervall (Sek) 56
CO2 unterer Schwellwert (ppm) 0
CO2 oberer Schwellwert (ppm) 0
Temperatur Aktualisierungsintervall (Sek) 56

Figure 18: Step 9 (left) –and step 10 (right) – setting the data interval for data from Blu2Light Multisensor AIR.

If the steps above have been configured, the data shall appear in Grafana within the configured time intervals. For a Blu2Light Multisensor XS, the brightness and motion sensor has only to be activated in the menu. As well as "send always to mesh". The interval is fixed and cannot be changed.

5 UPDATING THE VS LAN GATEWAY FIRMWARE

The VS LAN Gateway has a firmware update function. You can use either the VS LAN Gateway Demo software or you can use the available update tool program for Windows. The Gateway can be updated from within the web configuration site. In the following, the general limitations of the update process are being described.

5.1 GENERAL LIMITATIONS OF THE UPDATE PROCESS

- The device must be in the same subnet of your LAN network for the update process.
- If the device is powered via PoE and is supplying other devices in a daisy chain, note that the chained devices will be disconnected from PoE during the update process.
- If you use the Windows update Tool, you must provide a DHCP server in your network.
- Attention: Allow the Windows firewall to release the port for an update.
- Please make sure that you click on “Network information” and “Refresh version info” in LiNA Connect after a successful update so that the app knows the new firmware version of your LAN Gateway.

5.2 UPDATE PROCESS WITH VS UPDATE TOOL FOR WINDOWS

For the update with the “VSMediaConverterUpdateTool”, the push button S_{401} on the PCB must be pushed while powering the device on. For the update with the UpdateTool, it is necessary, that the PC from which the update is being performed and the Blu2Light LAN Gateway are in the same network segment. It is necessary to have a DHCP-Server which provides an IP-address to the gateway. The firmware update will not be possible without an assigned IP address from a DHCP-Server. If the bootloader-mode has been entered successfully, the following LED indication will be visible:

LED 1	LED 2
○	🟢

If the LAN Gateway doesn’t get an IP address, this state will hold until a power on reset occurs.

If it gets an IP address successfully from the DHCP server, the LED 2 will stay green while the LED 1 will start to blink in blue with 3 Hz for 30 seconds:

LED 1	LED 2
🔵	🟢

If no firmware update has been made or the firmware update hasn't been done successfully, the LED 2 will go off and the LED 1 will start to shine constantly for about 5 seconds.

LED 1	LED 2
🔵	○

When the device has been restarted successfully the LED 1 will be shining red constantly:

LED 1	LED 2
🔴	○

If a firmware update is in progress the LED 1 (blue) and LED 2 (green) will shine constantly:

LED 1	LED 2
🔵	🟢

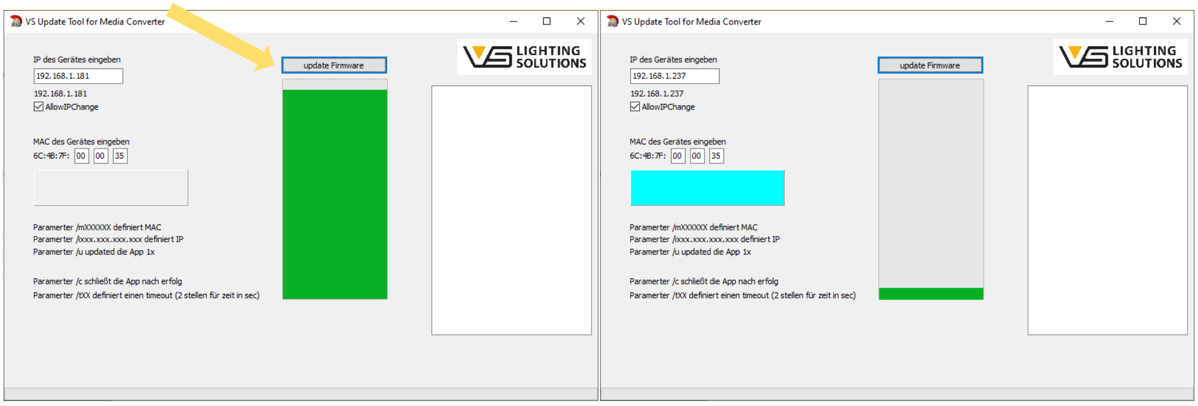
When the firmware update has been performed successfully, the LED 1 will shine constantly blue for 5 seconds until it will turn red in the same process as described before when no firmware has been put on the LAN Gateway.

If the Blu2Light LAN Gateway is in "DHCP-Mode" and no IP can be assigned from the DHCP server, the LED 1 will blink with 1 Hz until it has a valid IP:

LED 1	LED 2
▥	○

The indication of the LEDs is as well shown by the horizontal and vertical bar in the software. The MAC address can be found on the device label.

The update can be started by clicking on the button “update Firmware” (yellow arrow):


VS Update Tool for Media Converter

IP des Gerätes eingeben
192.168.1.181
192.168.1.181
☑ AllowIPChange

MAC des Gerätes eingeben
6C:4B:7F: 00 00 35

Paramerter /mXXXXXX definiert MAC
Paramerter /xxx.xxx.xxx.xxx definiert IP
Paramerter /u updated die App 1x

Paramerter /c schließt die App nach erfolg
Paramerter /tXX definiert einen timeout (2 stellen für zeit in sec)

VS Update Tool for Media Converter

IP des Gerätes eingeben
192.168.1.237
192.168.1.237
☑ AllowIPChange

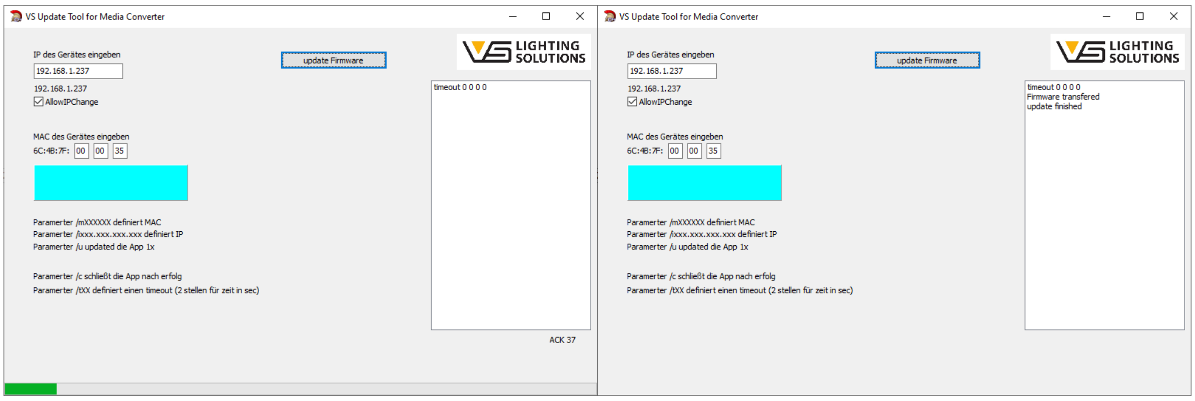
MAC des Gerätes eingeben
6C:4B:7F: 00 00 35

Paramerter /mXXXXXX definiert MAC
Paramerter /xxx.xxx.xxx.xxx definiert IP
Paramerter /u updated die App 1x

Paramerter /c schließt die App nach erfolg
Paramerter /tXX definiert einen timeout (2 stellen für zeit in sec)

Figure 19: Update with the VS Update Tool for the Blu2Light LAN Gateway #1

If a timeout occurs somehow, the update tool will automatically make another try to load the firmware onto the device.


VS Update Tool for Media Converter

IP des Gerätes eingeben
192.168.1.237
192.168.1.237
☑ AllowIPChange

MAC des Gerätes eingeben
6C:4B:7F: 00 00 35

Paramerter /mXXXXXX definiert MAC
Paramerter /xxx.xxx.xxx.xxx definiert IP
Paramerter /u updated die App 1x

Paramerter /c schließt die App nach erfolg
Paramerter /tXX definiert einen timeout (2 stellen für zeit in sec)

```
timeout 0 0 0 0
```

ACK 37

VS Update Tool for Media Converter

IP des Gerätes eingeben
192.168.1.237
192.168.1.237
☑ AllowIPChange

MAC des Gerätes eingeben
6C:4B:7F: 00 00 35

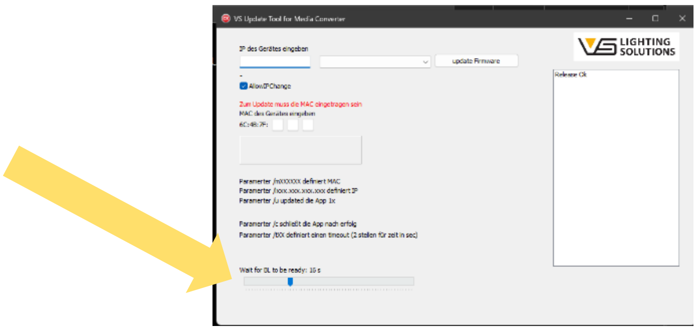
Paramerter /mXXXXXX definiert MAC
Paramerter /xxx.xxx.xxx.xxx definiert IP
Paramerter /u updated die App 1x

Paramerter /c schließt die App nach erfolg
Paramerter /tXX definiert einen timeout (2 stellen für zeit in sec)

```
timeout 0 0 0 0
Firmware transfered
update finished
```

Figure 20: Update with the VS Update Tool for the Blu2Light LAN Gateway #2

Since version 1.31, a slider has been added for the case that the assessment of the IP from the DHCP-Server to the Blu2Light LAN Gateway needs a few seconds. This time can be adjusted so that the update progress won’t fail. The following screenshot shows the slider with a yellow arrow:


VS Update Tool for Media Converter

IP des Gerätes eingeben
-
☑ AllowIPChange

Zum Update muss die MAC eingetragen sein
MAC des Gerätes eingeben
6C:4B:7F:

Paramerter /mXXXXXX definiert MAC
Paramerter /xxx.xxx.xxx.xxx definiert IP
Paramerter /u updated die App 1x

Paramerter /c schließt die App nach erfolg
Paramerter /tXX definiert einen timeout (2 stellen für zeit in sec)

Wait for BL to be ready: 15 s

```
Release Ok
```

Figure 21: adjustable wait-time for the firmware-update to be started

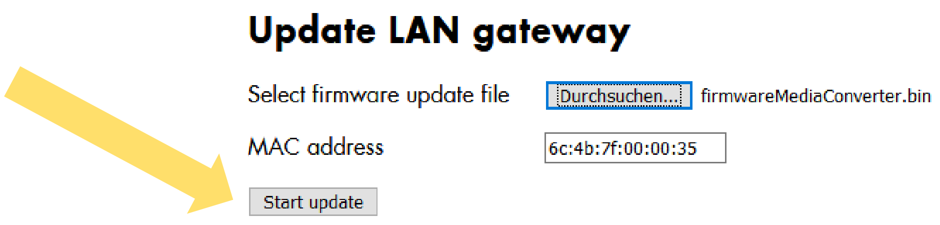
5.3 UPDATE VIA WEB CONFIGURATION PAGE OF VS DEMO SERVER

The device can as well be updated via the web configuration. The following steps describe the update process and show the LED indication in each corresponding mode.

The LED 1 and LED 2 of the Blu2Light LAN Gateway will show the following color while the LAN Gateway is in standard mode and when it is receiving data from the mesh, where it is commissioned to. LED 1 shines constantly green. The LED 2 flashes green while data from the mesh is incoming:

LED 1	LED 2
🟢	▥

By selecting the by VS provided, "bin-File" and entering the MAC address of the Blu2Light LAN-Gateway in the field "MAC address", the update process will start by clicking on "Start update" (see the yellow arrow in the next picture):



Update LAN gateway

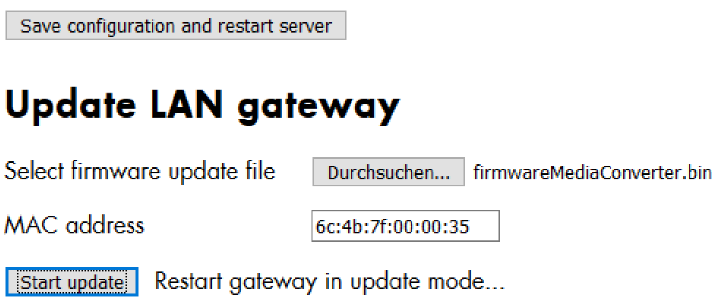
Select firmware update file [Durchsuchen...] firmwareMediaConverter.bin

MAC address 6c:4b:7f:00:00:35

[Start update]

Figure 22: "Update LAN Gateway" mask in the web management of the Raspberry Pi

After a click on "Start update", the Gateway will switch in update mode. LED 1 will shine blue, while LED 2 will be off. The message "Restart Gateway in update mode" will be shown as well:



[Save configuration and restart server]

Update LAN gateway

Select firmware update file [Durchsuchen...] firmwareMediaConverter.bin

MAC address 6c:4b:7f:00:00:35

[Start update] Restart gateway in update mode...

Figure 23: The Blu2Light LAN Gateway is put into "update mode"

LED 1 and LED 2 will indicate the following status:

LED 1	LED 2
🟢	○

LED 1 shines green and LED 2 will be dark for about 15 seconds.

LED 1	LED 2
○	🟢

LED 1 will be dark and LED2 shines green for about 10 seconds.

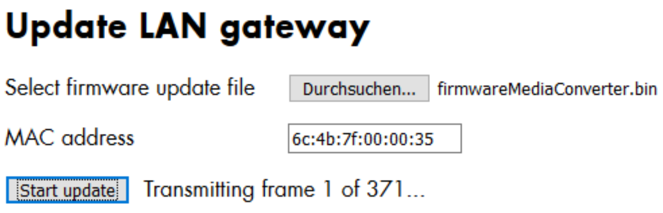
LED 1	LED 2
◍	○

LED 1 will blink blue (1 Hz) and LED 2 will be dark for about 2 seconds.

LED 1	LED 2
🔵	○

LED will shine constantly blue, and LED 2 will be dark for about 1 second.

The update process starts when LED 1 becomes dark and when LED 2 will constantly shine green. The whole progress will be shown down in the update window in the web configuration:



Update LAN gateway

Select firmware update file [Durchsuchen...] firmwareMediaConverter.bin

MAC address 6c:4b:7f:00:00:35

[Start update] Transmitting frame 1 of 371...

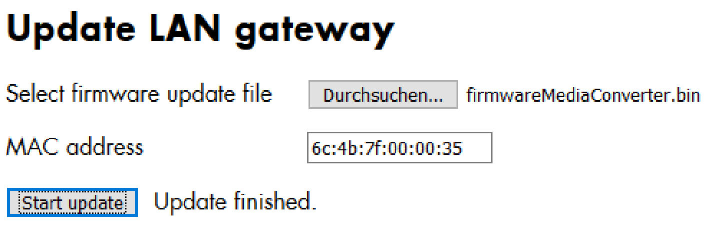
Figure 24: Frame 1 of 371 is being transmitted.

While the update process takes place, the LED 2 on the Blu2Light LAN Gateway will shine green. LED 1 will stay dark:

LED 1	LED 2
⚪	🟢

The update process will take approximately 60 seconds – during this time, LED 1 will be dark and LED 2 will be constantly green, depending on the firmware size, that is being flashed.

When the update process has been successfully finished, the message "Update finished" will appear next to the button "Start update":



Update LAN gateway

Select firmware update file [Durchsuchen...] firmwareMediaConverter.bin

MAC address 6c:4b:7f:00:00:35

[Start update] Update finished.

Figure 25: The Blu2Light LAN Gateway is put into bootloader-mode.

After a successful update, LED 1 will be constantly blue, and LED 2 will be off. The Blu2Light LAN Gateway will restart and after a short yellow phase of LED 1 (Blu2light LAN Gateway has a "IP address"), the LED 1 will shine constantly green, while LED 2 will flash when data is incoming:

LED 1	LED 2
🔵	⚪

LED 1 will be constantly blue, and LED 2 will be off for about 8 seconds.

LED 1	LED 2
🟡	⚪

LED 1 constantly yellow and LED 2 will be off for about 0,5 seconds.

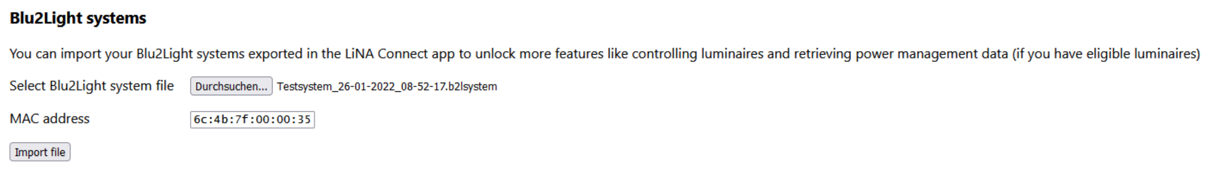
LED 1	LED 2
●	▥

The Blu2Light LAN Gateway will be back in normal operation mode (LED 1 constantly green and LED 2 flashing when data from the mesh arrives).

6 SUBSTITUTION OR REPLACEMENT OF A BLU2LIGHT LAN GATEWAY

If it is necessary to replace an already full configured Blu2Light LAN Gateway, the following steps shall be done:

1. Delete the "old" Blu2Light LAN Gateway in the corresponding system with LiNA Connect App.
2. Add the new Blu2Light LAN Gateway to the corresponding system (Commissioning).
3. Configure the new Blu2Light LAN Gateway with the same parameters as IP-Address, Subnet-Mask, server IP Address and Port of the "old" Blu2Light LAN Gateway.
4. Create a new PSK in server Demo or use the old PSK and put it into the corresponding field in the App LiNA Connect at the Blu2Light LAN Gateway options.
5. Make a new copy of the current system in the option menu of Blu2Light LiNA Connect ("Backup and Restore).
6. Export the "Backup and Restore" configuration-file from LiNA connect app and import this file on the configuration web page of the LAN Gateway Demo. Therefore, navigate to http://<IP or domain of the device>:31460 to access the configuration page like shown in the following screenshot:



Blu2Light systems

You can import your Blu2Light systems exported in the LiNA Connect app to unlock more features like controlling luminaires and retrieving power management data (if you have eligible luminaires)

Select Blu2Light system file: Durchsuchen... Testsystem_26-01-2022_08-52-17.b2lsystem

MAC address: 6c:4b:7f:00:00:35

Import file

Figure 26: Import of the configuration file of the corresponding system in the web configuration

7. The LED1 on the Blu2light LAN Gateway shall be green then and the configuration of the replaced or substituted Blu2light LAN Gateway has been successfully completed.

7 COMMUNICATION BETWEEN THE LAN GATEWAY AND THE SERVER

The communication is based on an SSL/PSK encrypted connection. The server acts as a socket for several Gateways to connect to. All incoming traffic on the Bluetooth side is forwarded to the ethernet socket.

7.1 THE BLU2LIGHT PROTOCOL

The communication frames follow the Blu2Light protocol.

Group	Offset	Size	Field	Value	Description
-	0	1	Sync Byte	0x42 = 66 = 'B'	Serial sync byte; always 0x42
Header	1	1	Address Length = AL	0xnn	
	2	2	Block Count = BC	0xnnnn	Data Length (DL) is BC * 8 – PL – 2
	4	2	Packet Type	0xnnnn	See Appendix F
	6	1	Packet Version	0xnn	
	7	2	Manufacturer	0xnnnn	0 = according open standard 1 = VS
	9	1	CRC Header	0xnn	
Address	10	1	Address Status	0xnn = 0b0f0edcba	Bit field for present Address data (1 = present): a = Source ID field b = Destination ID field c = Destination Type field d = Packet ID field e = Encryption Type field

Group	Offset	Size	Field	Value	Description
					(0=unencrypted) f = Timestamp field
		2	Source ID	0xnnnn	ID of the source node
		2	Destination ID	0xnnnn	ID of destination node
		1	Destination Type	0xnn	0 = Message between BT-/MWAYfirmware 1 = Mesh-Message 2 = Message between Server / Gateway
		1	Packet ID	0xnn	Necessary to identify response frame
		1	Encryption Type	0xnn	0 = NodekeyID 1 = SystemkeyID N= UserkeyID#N-2; (N=2..251)
		4	Timestamp	0xnnnnnnnn	
		1	CRC Address	0xnn	
Data (encrypted)		1	PL	0xnn	Length of Padding Data
		DL	Data	…	DL = BC *–8 - –L - 2
		1	CRC of Data	0xnn	CRC of decrypted Data
		PL	Padding data 0x00	…	Add Padding to reach DL + PL + 2 = BC * 8
		1	CRC Data	0xnn	
		1	CRC of CRCs	0xnn	CRC of CRC

Group	Offset	Size	Field	Value	Description
					Header & CRC Address & CRC Data

7.2 AVAILABLE EVENTS FROM SENSORS

ET_SENSE_MOVEMENT (Movement events; MultiSensor XS, XL, XXL, Industrial, Air)

EventNumber	Input number	Count of movement events in measurement interval
6 (1 Byte)	(1 Byte)	(1 Byte)

ET_SENSE_BRIGHTNESS (Brightness events; MultiSensor XS, XL, XXL, Industrial, Air)

EventNumber	Input number	Current brightness value	Target brightness value
7 (1 Byte)	(1 Byte)	(2 Byte, big endian)	(2 Byte, big endian)

ET_SENSE_AIR (Brightness, CO2, temperature & humidity events; MultiSensor Air)

EventNumber	Sensor ID	Measurement value	Alarm

251 (1 Byte)	(1 Byte)
	SENSOR_ID_LIGHT = 0,
	SENSOR_ID_CO2 = 1,
	SENSOR_ID_TEMPERATURE = 2,
	SENSOR_ID_HUMIDITY = 3
		(4 Byte), float, little endian	(1 byte)

7.3 USED CIPHER SUITE

IANA name:	TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256
OpenSSL name:	ECDHE-PSK-CHACHA20-POLY1305
GnuTLS name:	TLS_ECDHE_PSK_CHACHA20_POLY1305
Hex code:	0xCC, 0xAC
TLS Version(s):	TLS1.2
Protocol:	Transport Layer Security (TLS)
Key Exchange:	Elliptic Curve Diffie-Hellman Ephemeral (ECDHE)
Authentication:	Pre-shared key (PSK)
Encryption:	ChaCha stream cipher and Poly1305 authenticator (CHACHA20 POLY1305)
Hash:	Secure Hash Algorithm 256 (SHA256)
Included in RFC:	RFC 7905

7.4 PORTS

TCP 31460	used for web interface (http)
TCP 31461	used for LAN Gateway connection (configurable)
UDP 31462	used for LAN Gateway Firmwareupdate
TCP 3000	used by Grafana.

8 COMMUNICATION BETWEEN THE LAN GATEWAY AND THE SERVER

8.1 PING (GATEWAY ONLY)

The ping command is used to check the communication if the LAN Gateway is connected and if the keys are correct. To ping the Gateway, simply use the command *ping* without any parameters.

Example:

```
ping()
```

Example description:

Ping command will be sent, and Gateway will answer.

8.2 HOW TO ENCODE A B2L COMMAND

To send commands to nodes in mesh, the command must be encoded with Systemkey.

To encode a B2L command, we use the function *build_enc_frame*.

Command parameters:

```
build_enc_frame(        data: bytearray,        #command with parameters to be sent
                        source_id: int,         #who sent the package.
                        destination_id: int,    #package destination
                        packet_id: int,         #ID to identify response frame.
                        key: UUID)              # the key to encrypt with
```

return value is a bytearray of the Blu2Light frame.

Range of values:

data:	command to be encoded (example: *set_fg_state* function)
source_id:	0 to 0xFFFF
destination_id:	0 to 0xFFFF
	0 = broadcast
packet_id:	0 to 0xFF
key:	key as UUID

Example:

build_enc_frame(set_fg_state(…), 0, da‘a['targe’Id'], 99, sys[1]. net_key)))

Example description:

A frame with the command set_fg_state will be created. **Source ID is 0** and **target ID** (nodeId of the target node, will **be added from an array**. **Packet ID** is **99** and the **encryption key** will also be added **from an array**.

8.3 HOW TO SEND A B2L FRAME

Use function *config.send_queue.put* to send out a B2L frame.

config.send_queue is a Queue object, for sending data over ethernet, to the LAN Gateway.

Command parameters:

config.send_queue.put((socket_id, data))

Range of values:

socket_id:	The socket ID of the gateway to which the data should be sent.
data:	The data to send.

Example:

config.send_queue.put((sys[0], build_enc_frame(X))

Example description:

The **encoded B2L frame** will be added to the **send queue**, with **socket ID** added from **an array**.

8.4 HOW TO CREATE A LIGHT CONTROL COMMAND

To create a light control command, the function *set_fg_state* is used. It will create the payload part of the B2L command frame. With ***build_enc_frame***, the B2L command gets encrypted into a complete B2L frame.

The function *config.send_queue.put* sends out the frame to LAN Gateway and further to Mesh.

Command parameters:

```
set_fg_state(FGNumber: int,          #function group number
             newState: FGStates,     #state of the function group (FG)
             sceneNum: int,          #scene number
             lightLevel: int,        #Light level in %
             param: int = 0)         #additional parameter, defaults to 0
```

Range of values:

FGNumber:	0 to 15
FGStates:	STATE_MANUAL = 0
	STATE_AUTO_ACTIVE = 1
	STATE_AUTO_PASSIVE = 2
	STATE_AUTO_BASIC = 3
	STATE_AUTO_OFF = 4
	STATE_SEQUENCE = 5
	STATE_KEEP_CURRENT = 255
sceneNum:	0 to 63
lightLevel:	0 to 100

Example:

```
set_fg_state(1, STATE_MANUAL, 2, 42)
```

Example description:

This light command will set **function group 1** to FG state to Manual, switch to **scene 2 at 42%.**

8.5 HOW TO READ THE FUNCTION GROUP STATES

To read the function group state, the function *get_fg_state* is used. With *build_enc_frame*, the B2L command gets encrypted into a complete B2L frame.

The function *config.send_queue.put* sends out the frame to LAN Gateway and further to Mesh.

Command parameters:

get_fg_state(FGNumber: int)   #function group number (FG)

Range of values:

FGNumber: 0 to 15

Example:

get_fg_state(0)

Example description:

Function group 0 will be read.

8.6 HOW TO CREATE A DALI TUNNEL

The DALI tunnel is used to send DALI commands through a B2L device directly to a DALI driver (for example to read out parameters or memory banks). To create a DALI tunnel command, use the function *dali_tunnel*. With *build_enc_frame*, the B2L command gets encrypted into a complete B2L frame.

The function *config.send_queue.put* sends out the frame to LAN Gateway and further to Mesh.

WARNING: Do not use the DALI tunnel to change direct any DALI parameters, because this may cause malfunctions of the B2L system.

Command parameters:

```
dali_tunnel( dali_cmd: int,         #DALI command (DALI standard)
             dtr0: int = None,      #dtr0 register
             dtr1: int = None,      #dtr1 register
             dtr2: int = None,      #dtr2 register
             edtx: int = None,      #enable device type
             dri_addr: int = 0xFE,  #DALI address byte (DALI standard)
             repetition: int = 0,   #send DALI command multiple times
             prio: int = 0,         #priority
             answer: int = 0,       #answer required
             repeat: int = 0)       #command must be sent twice (according to
                                    the DALI standard)
```

Range of values:

dali_cmd:	0 to 255 (0xFF)
dtr0:	0 to 255 (0xFF)
dtr1:	0 to 255 (0xFF)
dtr2:	0 to 255 (0xFF)
edtx:	0 to 255 (0xFF)
dri_addr:	0 to 255 (0xFF) (YAAA AAAS) Y: short or group, A: address bits, S: standard or DAPC
repetition:	0 to 64
prio:	0: high 1: low
answer:	0: no 1: yes
repeat:	0: no repetition (default) 1: repetition

Example:

```
dali_tunnel(dali_cmd=0xC5, dtr0=0, dtr1=202, dri_addr=0b0000 0011,
repetition=1, answer=1)
```

Example description:

The **DALI command 0xC5** (READ MEMORY LOCATION) is send **twice (repetition = 1)**, with the precommands **SetDTR0(0)** and **SetDTR1(202)**, addressed with **short address 1.** An **answer is required,** and **no device type** is set. This example will read byte 0 and 1 of DALI memory bank 202 of the addressed DALI controlgear with shortaddress 1.

8.7 PMD COMMANDS

PMD (Power Metering and Monitoring Device) commands are used to monitor different parameters of DALI devices, like power or energy consumption. Therefore, the DALI device must support device type 49, 50, 51 and/or 52.

To get the PMD readings, we use the dali_tunnel functionality of our B2L devices with DALI interface.

8.7.1 PMD INITIALIZATION

The function *pmd_init_start* is used to search for all nodes and DALI devices, which are compatible with PMD. Therefore, a list of all DALI devices available in the B2L system must be added by importing a Blu2light system file (chapter 4.9).

This function will activate the *pmd_init_handler*, which will receive and collect the answers.

Command parameters:

pmd_init_start() -> None

Example:

pmd_init_start()

Example description:

PMD will be initialized. This is necessary after every new import of a Blu2Light system file to the Server.

It will check all configured DALI devices on all nodes with DALI devices for supported PMD functionality.

8.7.2 PMD RETRIEVE

The function *pmd_retrieve_start* is used to read out the PMD parameter **Actual power** of every DALI device on every B2L node, which support PMD functionality and is initialized.

This function will activate the *pmd_retrieve_handler*, which will receive/collect the answer of the devices.

Command parameters:

pmd_retrieve_start() → None

Example:

pmd_retrieve_start()

Example description:

It will read out the PMD parameter **Actual power** of every DALI device on every B2L node, which support PMD functionality and is initialized.

PMD quit

The function *pmd_init_quit* is used, to abort a PMD search after timeout.

Command parameters:

pmd_init_quit() → None

Example:

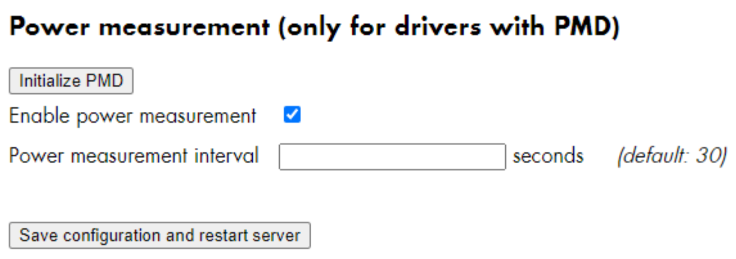
pmd_init_quit()

Example description:

The running PMD search will be aborted.

8.7.3 USING PMD WITH THE SERVER DEMO

In the LAN Gateway server demo you can find a section named "Power measurement (only for drivers with PMD)".


Power measurement (only for drivers with PMD)

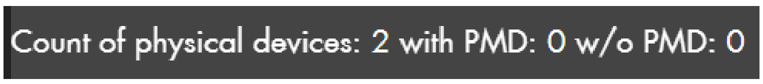
Enable power measurement ✓

Power measurement interval seconds *(default: 30)*

By pressing the button "Initialize PMD" first the pmd_collect_devices function must be called. After that the pmd_init_start function gets triggered. As mentioned before, therefore a B2L system must be added. While the PMD initialization is running "pmd_init: PMD initialization running..." is displayed in the status section. When the initialization is done, the check mark for "Enable power measurement" can be set. By enabling the power measurement, every configured interval, default is 30 seconds, the current PMD parameter 'Actual power' will be read from the compatible drivers. This interval can be changed in the "Power measurement interval" text field. After changing the configuration, click "Save configuration and restart server" button to save it.

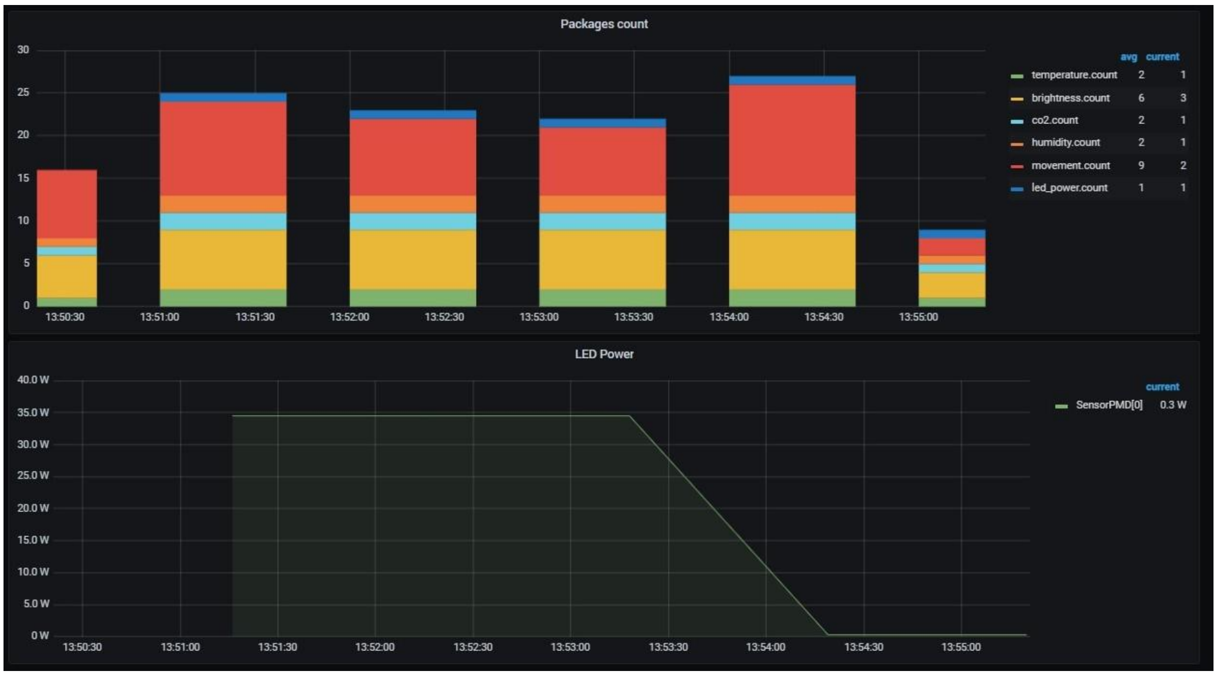
Important: As soon as a new B2L system gets uploaded to the server, PMD must be initialized again.

In "System overview and light control" for every node with physical devices, the number of devices with and without PMD is shown. If the count of physical devices and the sum of those with and without PMD doesn't match up, like in the following picture, this means there wasn't a PMD search yet or some devices weren't able to respond (f.e. turned off or disconnected from the DALI bus). In this case, PMD must be initialized again, with all physical devices available.


Count of physical devices: 2 with PMD: 0 w/o PMD: 0

8.7.4 ADDING THE POWER MEASUREMENT TO GRAFANA

On every new start of server demo, it will be checked, if any PMD devices are initialized and the dashboard template will be updated. If there any DALI devices active (initialized), the LED power graph will automatically be shown, in the Grafana dashboard.



	avg	current
temperature.count	2	1
brightness.count	6	3
co2.count	2	1
humidity.count	2	1
movement.count	9	2
led_power.count	1	1

	current
SensorPMD[0]	0.3 W

Figure 27: Grafana view of the power-measurement

8.8 B2L ENCRYPTION METHOD

The data transfer in the B2L mesh is additional encrypted with XTEA (e**X**tended **T**iny **E**ncryption **A**lgorithm). To ensure, that the B2L network is safe, the commands for the B2L network must be sent XTEA encrypted, from the server.

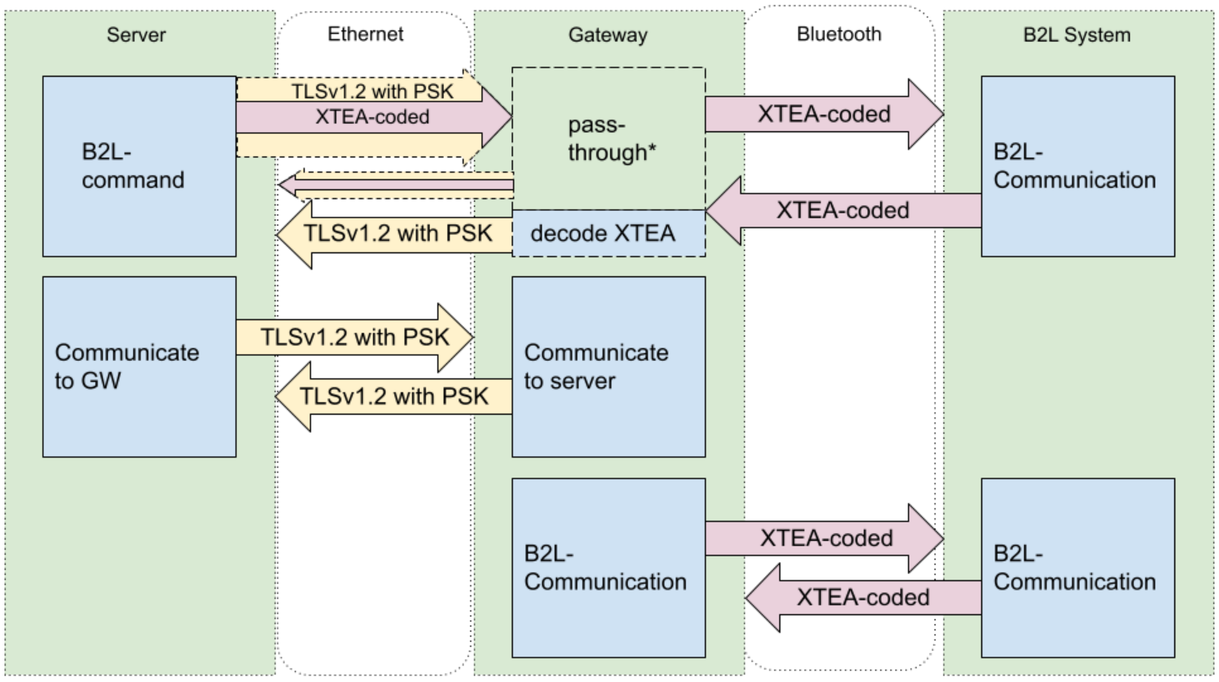
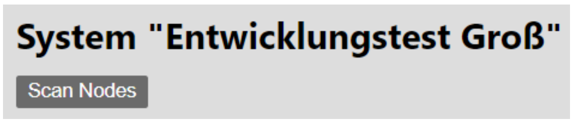


Figure 28: Visualization of the XTEA-algorithm in B2L

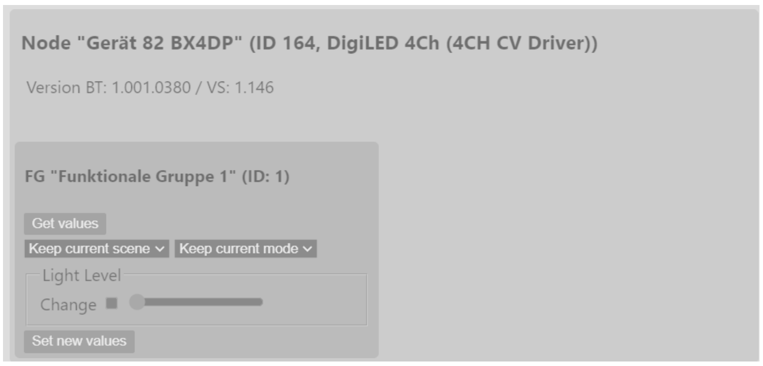
*Answers to server commands will get passed through to the server. B2L-events will get decoded in the Gateway.

8.9 SCAN NODES

"Scan Nodes" was added with Server Demo v1.16. With this command all nodes of a system will be checked if they are reachable. This function can be found in "System overview and light control".



As soon "Scan Nodes" is clicked on, the system will check for every device. After this the page must be refreshed. Inactive nodes will be grayed out.

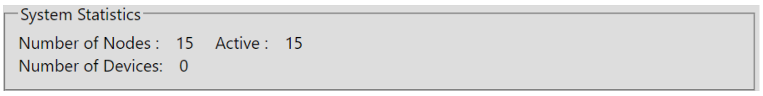

Node "Gerät 82 BX4DP" (ID 164, DigiLED 4Ch (4CH CV Driver))

Version BT: 1.001.0380 / VS: 1.146

FG "Funktionale Gruppe 1" (ID: 1)

*Figure 29: **Node Overview in the web-configuration***

Under "System Statistics" the number of nodes, active nodes and lighting devices can be found:


System Statistics

Number of Nodes : 15 Active : 15

Number of Devices: 0

Figure 30: Overall-statistic in the web-configuration

9 WHERE TO PULL DATA

If you need access to data from your B2L system, there are several different ways to get them:

9.1 READ FROM DATABASE (RECOMMENDED)

Movement events from the different Blu2Light MultiSensors, as well as measurements from the "Blu2Light MultiSensor AIR" are directly stored into the database (MariaDB or InfluxDB) and can be accessed from there.

If PMD is initialized (chapter 8.7), this data is also be stored in the database.

9.2 WHEN WRITTEN TO DATABASE

Another way to get data is adding an additional handler function, that writes the incoming events to another database or location (file / server). For example, a modified DBHelper, based on the functions *MariaDBHelper* (maria_db.py) or *InfluxDBHelper* (influx_db.py), can be created for that. Which DBHelper is used, can be modified in _main_.py.

9.3 DATABASE PAGE OF THE SERVER UI

The database page can be found under *http://<IP or domain of the device>:31460/database*. The last ten entries for CO2, light level, LED power (PMD), movement, humidity, temperature, and brightness can be found here.

Additional: The light level is the combination of Master dimmer and Channel brightness combined. So, this is the actual DAPC value of the functional group.

9.4 RETRIEVE EVENTS

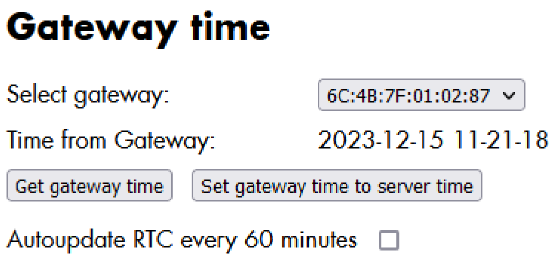
The third possible way to get data, is in the event parser. Therefore, the function parse_event (b2l_parser.py) can be modified to get the data and handle them differently.

9.5 USING B2L COMMANDS

Another possibility is by using B2L commands. How to create B2L commands can be found in chapter 8.2.

9.6 GATEWAY TIME

The time from the Gateway can be retrieved as well in the web-configuration by clicking on "Get Gateway time":



Select gateway: 6C:4B:7F:01:02:87
Time from Gateway: 2023-12-15 11-21-18
Autoupdate RTC every 60 minutes ☐

Figure 31: Gateway-time menu in the web-configuration

The time can as well be set to the Gateway by clicking “Set gateway time to server time”. This can be done automatically every 60 seconds as well by activating the checkbox next to “Autoupdate RTC every 60 minutes”.

10 BEACONING

With the Beaconing functionality of all Blu2Light nodes, including LAN Gateway, the user can configure advertising messages, up to 31 Bytes, which will be send out periodically. This can be an URL for example. Common profiles for Bluetooth Beacons are iBeacon by Apple or Eddystone by Google. But also, custom advertising messages are supported by Blu2Light, as long these are below 31 Bytes long.

For more information, read these documentations provided by our Partner M-Way:

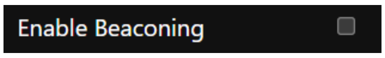
https://www.bluerange.io/docs/fruitymesh/BeaconingModule.html

https://www.bluerange.io/docs-commercial/bluerange-manual/Beaconing/Beaconing.html

10.1 HOW TO SET UP A BEACON MESSAGE IN OUR WEB UI

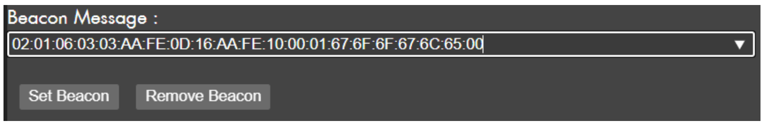
The easiest way to set up a beaconing/advertising message is the Gateway demo software.

To set it up it first has to be enabled. This can be found on the main page under 'Configuration'.



To use beaconing the checkbox "Enable Beaconing" must be activated.

When beaconing is enabled beacon messages can be set in "System overview and light control":


Beacon Message :

02:01:06:03:03:AA:FE:0D:16:AA:FE:10:00:01:67:6F:6F:67:6C:65:00

Insert the Beacon Message and the click "Set Beacon", to write the message to the node.

Click "Remove Beacon", if you want to delete the Beacon Message from the node.

The Beacon Message can only be set or deleted, but not read.

There are 2 example Beacon messages selectable by pressing the triangle sign at the text box.

10.2 HOW TO SET UP A BEACON MESSAGE USING THE WEB API

A beacon can be set or removed via Web-API by using */api/light_control* and the commands “sb” (set beacon) or “rb” (remove beacon). More information can be found in chapter 11.

11 ADDITIONAL INFORMATION

11.1 WEB-API CALLS

All API calls are handled in webconfig.py. In this file additional calls should be added.

NOTE:

Calling an API function on a Blu2Light node requires keeping the limit of a wireless system into account. Per Blu2Light node only one request should be sent out at a time. Also note, to each system, also only a limited number of calls to each system should run at a time.

This rate limit is not part of the DEMO software in all cases but could be added by the user at any time.

There is a packet-ID (1 byte), which can be used to send several requests and keep tracking of the corresponding answer.

All examples are shown using "curl". It is a command line tool which offers data transferring with URLs.

Some commands and replies have been colored to get a better understanding of the command or output.

Variable outputs are printed within apostrophes: 'apostrophe'.

Fixed output strings are printed within quotation marks: "quotation mark".

11.1.1 CSRF TOKEN

Read out the current valid token, which is necessary to use the commands that require the CSRF-token (where “login required” is necessary):

1. Get http://<IP or domain of the device>:31460/
2. Search for entry name = “csrf-token” and take the ‘value’ which contains the csrf-token (the token has 64 Byte length).
3. Take the csrf-token and save it for further internal use.
4. After a start or a restart of the server the csrf-token is being changed and the procedure must be done again.

Output: complete HTML-reply of the server.

http://<IP or domain of the device>:31460/api/light_control?csfr=csfr token

Example: If you like to use “CURL” you can use the following command:

```
curl -u user:password http://<IP or domain of the device>:31460/
```

You can take the “csrf-token” out of the response and take it for your application.

11.1.2 /API/GET_STATUS

Read out active connections, how many packages were received, MAC-Address of active system and the state of the "pmd init" request.

Parameter: none

Output: JSON: {"connections": ["XX:XX:XX:XX:XX:XX"], "received_packets": "N since year-month-day hour:minute:seconds", "systems": ["XX:XX:XX:XX:XX:XX"], "pmd_init_state": "state"}

N: count of packets.

state: Status of pmd-init-request (0 = not running, 1= running).

Example 1:

Command to Server:

```
curl -u admin:admin http://10.254.105.67:31460/api/get_status
```

Response from Server:

```
{"connections": ["6C:4B:7F:01:02:67"], "received_packets": "925 since 2024-03-08 09:39:18", "systems": ["6C:4B:7F:01:02:67"], "pmd_init_state": "0"}
```

If there are no active connections, the following output may occur:

Example 2:

Command to Server:

```
curl -u admin:admin http://10.254.105.67:31460/api/get_status
```

Response from Server:

{"connections": [], "received_packets": "0 since 2024-03-08 11:14:34", "systems": ["6C:4B:7F:00:01:01"], "pmd_init_state": "0"}

11.1.3 /API/GET_LC_STATUS

Read a list of active nodes of all systems.

Parameter: none

Output: JSON: {“Systems”: [“mac”: “XX:XX:XX:XX:XX:XX”, “nodelist”: [NodeID, Timestamp of last connection, Type],};

Systems: “mac”: ‘mac-Address of active system’”.

nodelist: ‘NodeID‘ (Node ID of the Node).

Timestamp of last connection = Timestamp in the Unix-format.

Type: 0 = Node, 1 = LAN Gateway.

Example:

Command to Server:

```
curl -u admin:admin http://10.254.105.67:31460/api/get_lc_status
```

Response from Server:

```
{"Systems": ["mac”:6C:4B:7F:00:01:01"], “1”: [1, 1710143572, 0],};
```

11.1.4 /API/PMD_INIT

(login required)

Start PMD Initializing run (see chapter 8.7).

Parameter: none

Output: none (or error)

Example:

Command to Server:

```
curl -u admin:admin http://10.254.105.67:31460/api/pmd_init
```

Response from Server:

```
{“pmd_init”:started…”}
```

11.1.5 /API/GET_CONFIG

Read out config data.

Parameter: none

Output: JSON: {"mariadb": {"enabled": true, "port": "", "username": "name_of_the_user", "password": "pw", "dbname": " databasename","host": ""}, "web": {"username": "name_of_the_webuser", "password": "pbkdf2:sha256:passwordhash"}, "influxdb": {"enabled": false, "port": "", "username": "", "password": "", "dbname": "", " host": ""}, " beaconing ": {"enabled": true }, "psk": "key", "pmd": {"enabled": false, "interval": "seconds"}, "rtcupdate": {"enabled": true }, "system": ["***XX:XX:XX:XX:XX:XX***"]}

For the pre-shared key for connection between server and Gateway (see chapter 4.4).

- Config of **web interface**: ("username": 'name_of_the_webuser' , "password": 'passwordhash').

- Config of **Maria database**: ("mariadb": {"enabled": 'true' or 'false', "port", "username", "password", "dbname", "host"}).

- Config of **Influx database**: ("influxdb": {"enabled": 'true' or 'false', "port", "username", "password", "dbname", "host"}).

- Config of pmd ("enabled": 'true' or 'false', "interval": "seconds").

- beaconing ("enabled": 'true' or 'false').

- **rtcupdate**: ("enabled: 'true' or 'false').

- **system**: mac addresses of the imported systems in the

format (["XX:XX:XX:XX:XX:XX"]).

Example:

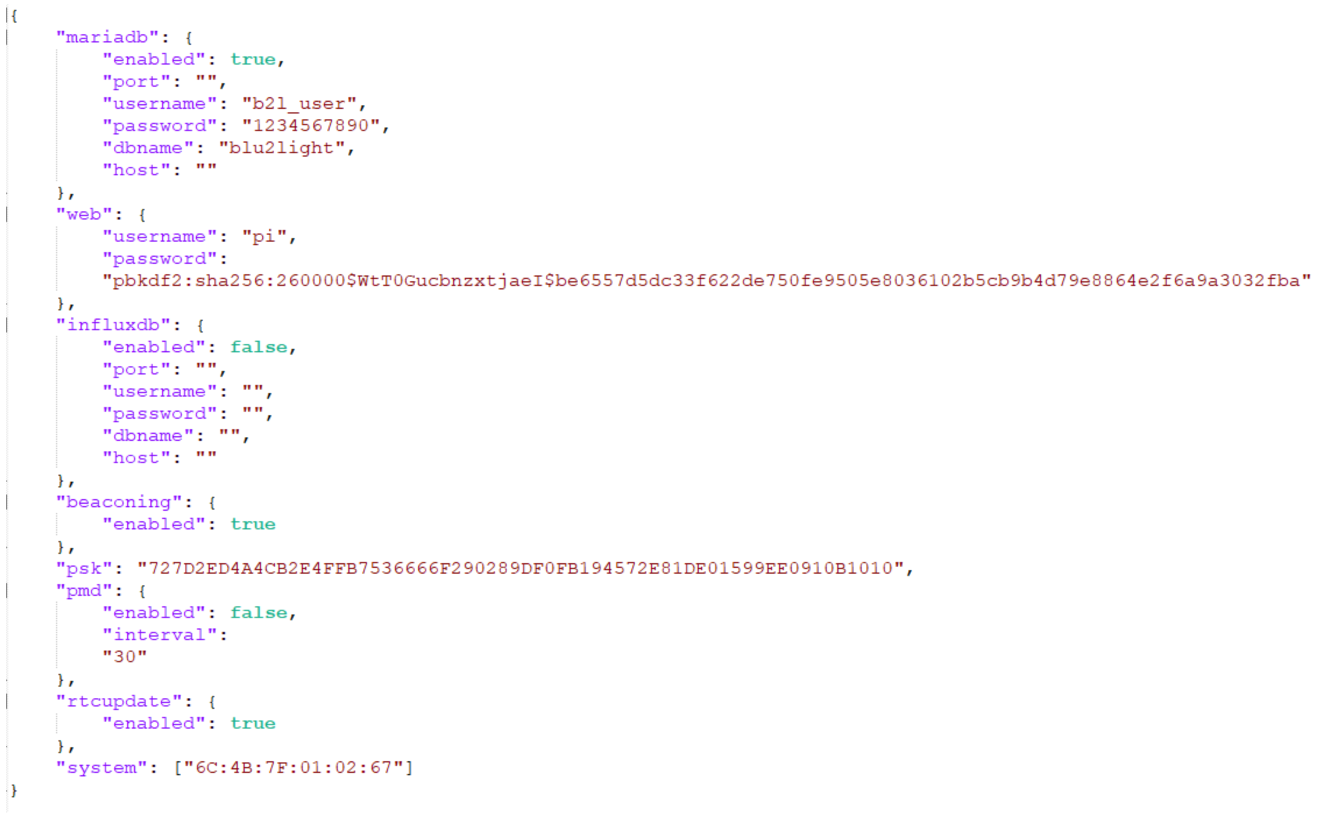
<u>Command to Server</u>:

```
curl -u admin:admin http://10.254.105.67:31460/api/get_config
```

<u>Response from Server</u>:

{"mariadb": {"enabled": true, "port": "", "username": "b2l_user", "password": "1234567890", "dbname": "blu2light", "host": ""}, "web": {"username": "pi", "password": "pbkdf2:sha256:pw"}, "influxdb": {"enabled": false, "port": "", "username": "", "password": "", "dbname": "", "host": ""}, "beaconing": {"enabled": true}, "psk": "key", "pmd": {"enabled": false, "interval": "30"}, "rtcupdate": {"enabled": true}, "system": ["6C:4B:7F:01:02:67"]}

The output can be shown in a more appropriate version as Json output:



```
{
    "mariadb": {
        "enabled": true,
        "port": "",
        "username": "b2l_user",
        "password": "1234567890",
        "dbname": "blu2light",
        "host": ""
    },
    "web": {
        "username": "pi",
        "password":
        "pbkdf2:sha256:260000$WtT0GucbnzxtjaeI$be6557d5dc33f622de750fe9505e8036102b5cb9b4d79e8864e2f6a9a3032fba"
    },
    "influxdb": {
        "enabled": false,
        "port": "",
        "username": "",
        "password": "",
        "dbname": "",
        "host": ""
    },
    "beaconing": {
        "enabled": true
    },
    "psk": "727D2ED4A4CB2E4FFB7536666F290289DF0FB194572E81DE01599EE0910B1010",
    "pmd": {
        "enabled": false,
        "interval":
        "30"
    },
    "rtcupdate": {
        "enabled": true
    },
    "system": ["6C:4B:7F:01:02:67"]
}
```

11.1.6 /API/GET_RTC_TIME

Read out the RTC time/date of the LAN Gateway.

Parameter: system: system mac address f.e.: /api/get_rtc_time?system= XX:XX:XX:XX:XX:XX

Output: RTC Time

JSON:{"msgType":"GetRTCTime","time":"YEAR-MONTH-DAY HOUR:MINUTES:SECONDS"}

Example:

Command to Server:

```
curl -u admin:admin http://10.254.105.67:31460/api/get_rtc_time?

system=6C:4B:7F:01:02:67
```

Response from Server:

```
{"msgType":"GetRTCTime","time":"2022-07-27 16:56:26"}
```

11.1.7 /API/SET_RTC_TIME

Sets RTC Time of the LAN Gateway. It uses the local time of the server.

Parameter: system: system mac address f.e.: /api/set_rtc_time?system= XX:XX:XX:XX:XX:XX

Output: HTML Response code: 204

Example:

Command to server:

```
curl -u admin:admin http://10.254.105.67:31460/api/get_rtc_time?
system=6C:4B:7F:01:02:67
```

Response from server:

HTML response code: 204

11.1.8 /API/SET_CONFIG

(login required)

Sets all configurations (f.e. PSK, server communication port and database) and restarts the server software.

Methods: POST

Parameter:	csrf	- CSRF-Token
	psk	- Pre-shared key (PSK) for connection between server and Gateway (see chapter 4.4)
	port	- Server communication port (default: 31461)
	mariadb	- Config Maria database (enabled, host (default: localhost), port (default: 3306), username, password, dbname)
	influxdb	- Config Influx database (enabled, host (default: localhost), port (default: 8086), username, password, dbname)
	pmd	- Config PMD (enabled, interval (default: 30))
	beaconing	- Config Beaconing (enabled)
	rtcupdate	- Config Automatic RTCupdate (enabled)

Example: Javascript: data={“port” :31461, ”psk”: “A5A387DADD6C319E51EC

F65B1973A8295F9430C186C9EB006375C5656CE32701,

“mariadb”: {

“enabled”: true,

“port”: “3306”,

“username”: “pi”,

```
        “password":        “pwd123”,
        “dbname”: “maria”,
        “host":       “127.0.0.1”
        },
        “influxdb”: {
        “enabled”: false,
        “port":       “8086”,
        “username”: “”,
        “password”: “”,
        “dbname”: “”,
        “host”: “”
        },
        pmd: {
        enabled: true,
        interval: “30”
        },
        beaconing: {
        enabled: false,
        },
        rtcupdate: {
        enabled: false,
        }
};
fetch('/api/light_control?csrf='+document.getElementById('csrf-
token').value, {
```

```
                method: 'POST', headers: {'Content-Type': 'application/json',
                },
                body: JSON.stringify(data),
        };
```

Output: If OK - HTML Response code: 204

If NOT OK - Bad CSRF token: 400

11.1.9 /API/DELETE_SYSTEM

(login required)

Deletes a B2L system from the Server.

Methods: GET

Parameter: csrf,

System: setting systems macaddress

Output: If OK - HTML Response code: 204

If NOT OK - Bad CSRF token: 400

Example:

Command to Server:

```
curl -u admin:admin http://10.254.105.67:31460/api/delete_system?csrf=
A5A387DADD6C319E51ECF65B1973A8295F9430C186C9EB006375C5656CE3270
1&system= 6C:4B:7F:01:02:67
```

Response from server:

If OK - HTML Response code: 204

If NOT OK - Bad CSRF token: 400

11.1.10 /API/GET_ERRORS

Gets errors from Server.

Parameter: none

Output: no error or a list of errors

Example:

Command to server:

```
curl -u admin:admin http://10.254.105.67:31460/api/get_errors
```

Response from server:

```
{"mariadb": null, "server": null}
```

11.1.11 /API/GET_MESSAGES

(login required)

Get the list of messages for the web interface.

Methods: GET

Parameter: none

Output: list of messages if no message available:

JSON:{ [ ] }

Example:

Command to server:

```
curl -u admin:admin http://10.254.105.67:31460/api/get_messages
```

Response from Server:

```
[]
```

11.1.12 /API/GET_UPDATE_STATUS

Get the status of a running firmware update.

Parameter: none

Output: update status or none if no update running f.e.: Transmitting frame 42 of 69...

Example:

Command to server:

```
curl -u admin:admin http://10.254.105.67:31460/api/update_status
```

Response from server:

If no update in progress: █

If update in progress: Transmitting frame 42 of 69...

11.1.13 /API/UPLOAD

(login required)

Upload / Imports a *.b2lsystem file of the connected B2L system to get infos about the connected devices.

It also generates the Grafana dashboard templates for this system.

Methods: POST

Parameter: csrf,

file (*.b2lsystem),

macaddr

Example:

```
let b2lsystem = document.getElementById("b2lsystem-file").files[0];

let formData = new FormData();

formData.append("file", b2lsystem);

formData.append("csrf",

document.getElementById('csrf-token').value);

formData.append("macaddr",
document.querySelector('macaddress').value);

fetch('/api/upload', {method: "POST", body: formData});
```

Response: Bad CSRF token: 400

No file: 400

Incorrect format

OK: 200

11.1.14 /API/UPDATE

(login required)

Upload an update file and starts the firmware update for the Gateway (.bin).

Methods: POST

Parameter: csrf,

file (*.bin),

macaddr

Example:

```
let update = document.getElementById("update-file").files[0];

let formData = new FormData();

formData.append(“file”, update);

formData.append(“csrf”,

document.getElementById('csrf-token').value);

formData.append(“macaddr”,
document.querySelector('macaddress').value);

fetch('/api/upload', {method: "POST", body: formData});
```

Output: none (or error)

Response: Bad CSRF token: 400

No file: 400

Incorrect format

OK: 200

11.1.15 /API/LIGHT_CONTROL

(login required)

Light control API

Methods:	GET	- Get values (default)
	POST	- Set values

Parameter (GET):	csrf	- CSRF-Token
	syst	- System (str)
	node	- Selected node Id (int)
	lum	- Selected functional group (FG) of the node (int)
	cmd	- Command (int) (see below)

Commands (GET):	41	- get_version_init
	80	- get_fg_state_init
	89	- get_gps_position_init

Example 1 (GET):
```
fetch('/api/light_control?csrf='+
document.getElementById('csrf-token').value+
'&system='XX:XX:XX:XX:XX'+'&node=+ids[2]+'&lum='+
ids[3]+'&cmd=80')
```

Example 2 (GET):
```
fetch('/api/light_control?csrf='+
document.getElementById('csrf-token').value+
'&system='XX:XX:XX:XX:XX'+'&node=+ids[2]+'&cmd=89')
```

Response: Bad CSRF token: 400

System not connected: 400

Incorrect format: 400

OK: 200

Parameter (POST): csrf - CSRF-Token

Data - Command (sfgs, eu, sb, rb, sn) and parameter (see below)

Data (sfgs): command - sfgs (set FG state), eu (energy update), sb (set beacon), (see chapter 8.2 and 8.4)

rb (remove beacon), sn (scan nodes)

lightLevel - DALI value (int)

FGNumber - Selected function group (int)

newState - New FG-state (manual: 0, auto (active: 1/passive: 2/basic: 3/off: 4), keep current state: 255) (int)

sceneNum - Selected Scene Number

targetId - ID of the Target Node (also named destination_id) (int)

system - Selected System (str)

Example (sfgs):

```
data =      {
command‘    'sfgs',
system:     ids[1],
targetId:   Number(ids[2]),
FGNumber:   Number(ids[3]),
newState:   Number(state),
sceneNum:   Number(scene),
lightLevel: Number(ll),
```

};

fetch('/api/light_control?csrf='+ document.getElementById('csrf-token').value,{method: 'POST', headers:'{'Content-Type': 'application/json',}, body: JSON.stringify(data),};

Data (eu):	command	- eu (Energy Update / Set Emergency)
	energyUpdate	- Energy Update Value (signed 8-bit value, range -100 to 100, unit: %, negative value will increase brightness, positive will reduce)
	system	- Selected System (str)

Example (eu):

```
data = {
command‘ 'eu',
system: ids[1],
energyUpdate:    Number(eu)
};
fetch('/api/light_control?csrf='+document.getElementById('
csrf-token').value, {method: 'POST',headers:'{'Content-Type':
'application/json',},body: JSON.stringify(data),};
```

Data (sb):	command	- sb (set beacon)
	Slot	- slot is a parameter to send more than one advertising message per Node. Currently only one slot (0) is available.
	beacon_msg	- The message that will be sent by the Node (see chapter 10)
	targetId	- ID of the Target Node (also named destination_id) (int)

system	- Selected System (str)

Example (sb):

```
data = {
command‘  ’sb',
system:      ids[1],
targetId:    Number(ids[2]),
slot:   0,
beacon_ms‘:
'02:01:06:03:03:AA:FE:0D:16:AA:FE:10:00:01:67:6F:6F:67:6C:65:
00
'};
fetch('/api/light_control?csrf='+document.getElementById('
csrf-token').value, {method: 'POST',headers:‘{'Content-Type‘:
'application/json',}, body: JSON.stringify(data),),};
```

Data (rb):

command	- rb (remove beacon)
Slot	- slot is a parameter for more than one advertising message per Node. Currently only one slot (0) is available.
targetId	- ID of the Target Node (also named destination_id) (int)
system	- Selected System (str)

Example (rb):

```
data =       {
command‘  ’rb',
system:      ids[1],
targetId:    Number(ids[2]),
slot:        0
```

};

fetch('/api/light_control?csrf='+ document.getElementById('csrf-token').value,{method: 'POST',

headers:‘{'Content-Type‘: 'application/json',},

body: JSON.stringify(data),),};

Data (sn): command - sn (scan nodes, requires server software 1.16 or newer)

system - Selected System (str)

Example (sn): data = {

command: 'sn',

system: ids[1]};

fetch('/api/light_control?csrf='+ document.getElementById('csrf-token').value, {

method: 'POST', headers: {'Content-Type': 'application/json',},

body: JSON.stringify(data),),};

Response:	Bad CSRF token: 400
	System not connected: 400
	Incorrect format: 400
	OK: 200

12 TROUBLESHOOTING

12.1 BAD CSRF TOKEN

The CSRF token is stored in the client browser and on the server. It's needed to authorize the communication between the web interface and the server. "Bad CSRF token" means that the token stored in the client browser is outdated.

How to fix: Refresh the Webpage (press "F5"-key in the webbrowser).

12.2 LED 1 RED

The red LED shows, that the Gateway doesn't have an IP address, and something is wrong with the network connection.

How to fix: Check the ethernet cable and check the router settings.

12.3 NO RESPONSE OF COMMAND SEND TO NODES IN A BLU2LIGHT SYSTEM

If there is no response of commands like 'GetFGState' which were sent over the Lightcontrol page, please check the following:

- Make Firmware update with LiNA Connect App for your system.
- Check if the node is switched on and active in the system with LiNA Connect App.
- Check if you had the latest system configuration exported from LiNA Connect App and imported in the VS LAN Gateway server demo.