User's Manual

Lightware Advanced Room Automation

Software Platform

v1.1 03-10-2023

Symbol Legend

The following symbols and markings are used in the document:

WARNING! Safety-related information that is highly recommended to read and keep in every case!

ATTENTION! Useful information for performing a successful procedure; it is recommended to read.

DIFFERENCE: Feature or function that is available with a specific firmware/hardware version or product variant.

INFO: A notice, which may contain additional information. Procedure can be successful without reading it.

DEFINITION: The short description of a feature or a function.

TIPS AND TRICKS: Ideas that you may have not known yet, but can be useful.

Navigation Buttons

Go back to the previous page. If you clicked on a link previously, you can go back to the source page by pressing the button.

Navigate to the Table of Contents.

Step back one page.

Step forward to the next page.

Document Information

All presented functions refer to the indicated products. The descriptions have been made while testing these functions in accordance with the indicated Hardware/Firmware/Software environment:

Item	Version
LARA version	1.1.10b1

Document revision: **v1.1**

Release date: **03-10-2023**

Editor: Tamas Forgacs, Laszlo Zsedenyi

About Printing

Lightware Visual Engineering supports green technologies and eco-friendly mentality. Thus, this document is made primarily for digital usage. If you need to print out a few pages for any reason, follow the recommended printing settings:

- Page size: A4
- Output size: Fit to page or Match page size
- Orientation: Landscape

TIPS AND TRICKS: Thanks to the size of the original page, a border around the content (gray on the second picture below) makes it possible to organize the pages better. After punching holes in the printed pages, they can easily be placed into a ring folder.

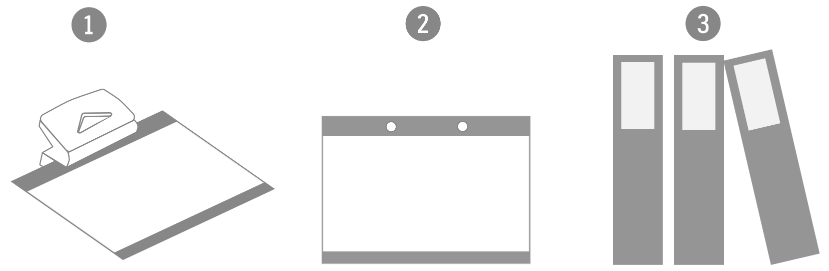


Table of Contents

1 Introduction

Thank You for choosing LARA. In the first chapter we would like to introduce the software, highlighting the most important features in the sections listed below:

1.1. Description

What is LARA?

Lightware Advanced Room Automation (LARA) is a future-proof room automation platform.

Where Can You Find It?

It runs integrated in Taurus UCX and MMX2 series devices, configurable via a browser. Meeting participants can control the devices in the room through a touch panel.

Solution for Meeting Rooms

LARA is a software platform that has been designed mainly for controlling meeting rooms. While developing LARA, the experiences of the popular Event Manager - which can be found in numerous Lightware devices - have been applied. The platform is designed and developed by Lightware Visual Engineering.

With LARA, you can automate your meeting rooms, create or re-use software modules by using the power of JavaScript, control the behavior of the Taurus and connect it to other third-party devices or services, or do virtually everything that is possible.

Supported Devices

LARA is available in the following devices:

TAURUS UCX		TAURUS TPX	MMX2
UCX-2x1-HC30	UCX-2x1-HC40	UCX-4x3-TPX-TX20	MMX2-4x3-H20
UCX-2x2-H30	UCX-2x2-H40		MMX2-4x1-H20
UCX-4x2-HC30	UCX-4x2-HC40		
UCX-4x2-HC30D	UCX-4x2-HC40D		
	UCX-4x3-HC40		

LARA Version

ATTENTION! The LARA versions may contain big differences. This User's Manual is based on a certain version of LARA that is displayed on the bottom of each page. Please check the LARA version of your device. LARA comes with the Taurus/MMX2 firmware package and we always recommend updating the firmware of your device to the latest version.

Key Benefits

Ready-to-use

LARA comes with factory driver modules that can be edited with wizards to eliminate the need for complex programming skills. These elements are re-usable with easy parameter-setup.

Open Integration

Controlling any device that supports open protocols (REST, TCP/IP, LW3, etc). No need for certificated training, as LARA is not based on our own programming language but JavaScript.

No External Controller

LARA is in Lightware products (for now in Taurus and MMX2) and can be used to connect and control third-party devices. No need to purchase an extra controller box.

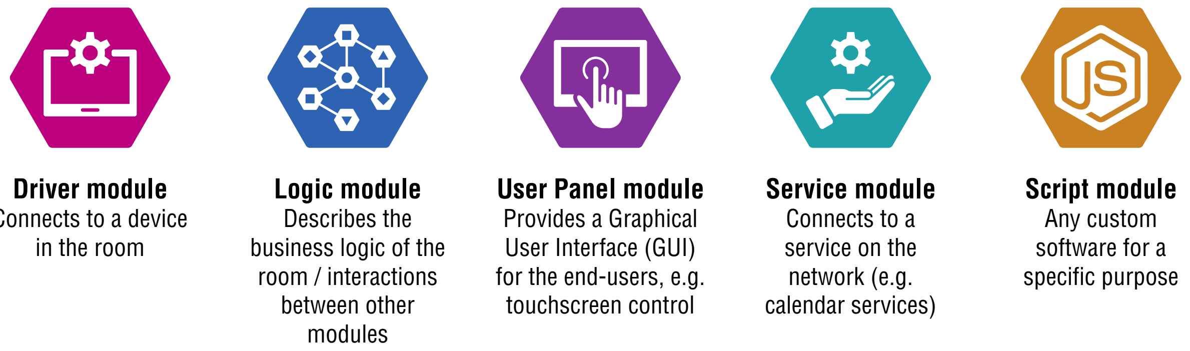
Up/downloading the Configuration

The full LARA configuration can be downloaded as a ZIP file. The file can be uploaded to the same device or to another device of the same type.

1.2. The Building Blocks of LARA

1.2.1. Modules

The module is the basic building block of LARA that contains a piece of software with optional parameters about a certain operation to be executed. It may contain JavaScript code based on Node.js® runtime environment. A module cannot be run directly, an **Instance** needs to be created from it. LARA comes with **built-in (factory) modules** but you can create your own modules as well.


Driver module
Connects to a device in the room
Logic module
Describes the business logic of the room / interactions between other modules
User Panel module
Provides a Graphical User Interface (GUI) for the end-users, e.g. touchscreen control
Service module
Connects to a service on the network (e.g. calendar services)
Script module
Any custom software for a specific purpose

The Module Types of LARA

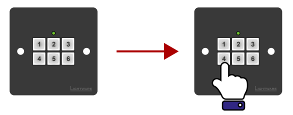
Driver Module

A driver module is an interface to a device. It can be used to connect to a device in a room. It has two types:

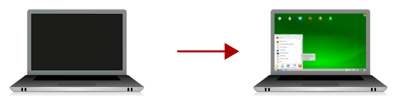
- **Device-specific driver**: e.g. Taurus / MMX2 Driver, Generic Lightware LW3 Driver
- **Communication driver:** e.g. Generic TCP Client Driver, Generic REST API Client Driver

Drivers can emit **Events** when something changes in the linked device. Examples:

- Signal appears on Input 1 (Signal Present)
- GPIO 1 input becomes Low



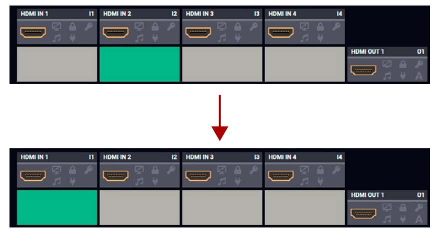
GPIO pin becomes low
(e.g. a button is pressed)



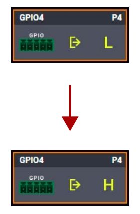
Signal is present
(e.g. a device is connected)

Methods are capabilities that can be invoked (called) to make a change in the linked device. Examples:

- Switch Input 1 to Output 1
- Set GPIO 4 output to High



Switching input1 to output1



GPIO4 output is switched from low level to high

Logic Module

Describes the **business logic** of the room and/or it can be used for interactions between other modules. A logic module creates connections between other modules with **Rules**. The rule will be triggered when the selected event from the selected source has been emitted.

Rules

A rule consists of the three main parts:

- **Event**: a specific change has happened.
- **Condition(s)***: after triggering the rule, LARA will check if all of the conditions apply.
- **Action step(s)**: execute one or more operations when the conditions are met.

* The condition management for now is only available with programming in the Action part. It will be available without programming in a future LARA release

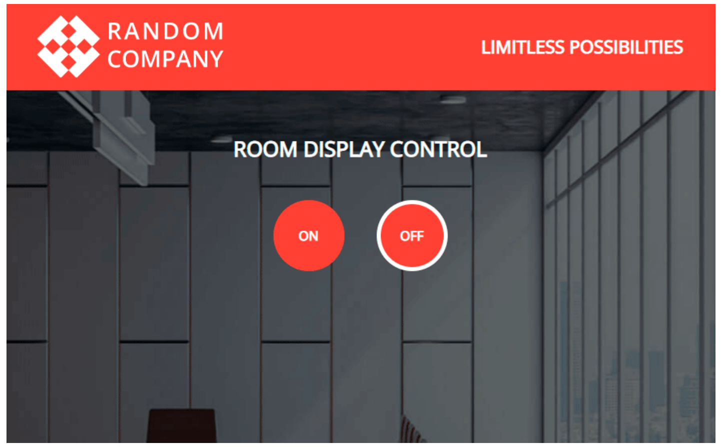
User Panel Module

Provides a GUI for the end-users, e.g. touch screen control.

HTML content is accepted that can be uploaded as a pre-formatted HTML file, and you can edit the content in the code editor later.

UserPanel Action Steps can modify the visible elements of the touch screen:

- The visibility,
- The text,
- A style property,
- The “Inner HTML” content (the content of the section: between `<div>` and `</div>`).



A Simple Touch Screen Interface (Example)

Service Module

Connects to a service on the network (e.g. calendar services).

Script Module

Any custom software for a specific purpose. e.g. **Cisco Webex module** (see the Most of the features are written in JavaScript and can be seen in this menu. section).

1.2.2. Instances

Modules cannot be run directly, an instance must be created from it. If you have a module you can define **parameters**. If you create an instance, you can set the **value** of the parameter in the instance. When creating **another instance**, you can define **another value** for the parameter in that instance.

With a simple analogy: if you have an executable file on your computer, that is a module. When you run it once or multiple times, the windows are the instances. The software pieces are identical behind the running instances, however, you can modify their behaviour by giving them different parameters at start.

Stored in the Module	Stored in the Instance
Properties of a parameter	Value of a parameter
Event	
Properties of an event parameter	Value of an event parameter
Method	
Properties of a method parameter	Value of a method parameter

1.3. The Intended Use of LARA

Please consider the following when using LARA:

- The Taurus/MMX2 configuration contains the whole LARA configuration as well. If the device configuration is handled with Bulk management or with the Backup/Restore feature, LARA settings can be preserved.
- If LARA is run and the Lightware device (that runs LARA) is restarted, LARA will **run automatically again**.
- A connected or an external device can be accessed via an instance.
- When LARA is running, all the **defined instances run together**.
- At most 20 instances can be run parallel if the complexity of the instances are at 'average' level (see the specification below of a complex sample configuration).
- The storage space is **128 MB** for the modules, instances, user module content and all codes.
- The available RAM is **128 MB**, LARA uses 56 MB in factory default state.
- Restoring the factory default settings in the UCX/MMX device will **delete the LARA configuration**. This also happens during a firmware update if the **firmware version is v2.0 or below**.

The Specifications of a Complex Sample Configuration

- It consists of 17 instances:
  - 14 from the Generic-tcp-ip-client module,
  - 1 from the Taurus-ucx-mmx2 module
  - 1 from the User panel module, and
  - 1 from the Logic module.
- This configuration needs cca. **92 MB RAM** space when running.

1.4. How to Access?

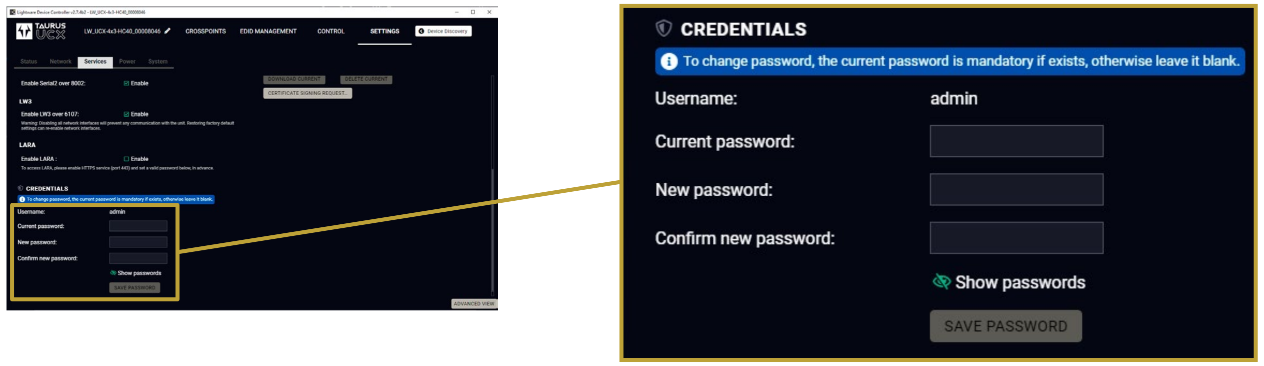
Short Steps

Step 1. Setting the password in the desired UCX/MMX2 device.
Step 2. Enabling port #443 (factory default state is enabled).
Step 3. Enabling LARA in the device.
Step 4. Opening LARA in a browser.

Detailed steps

Step 1 – Setting the Credentials in the Desired Taurus Device

- Start the **LDC software** and connect to the device or open the **web LDC** in a browser and type the IP address of the device.
- Navigate to the **Settings/Services** tab.
- **Set a password** for the user 'admin' (if not set previously).



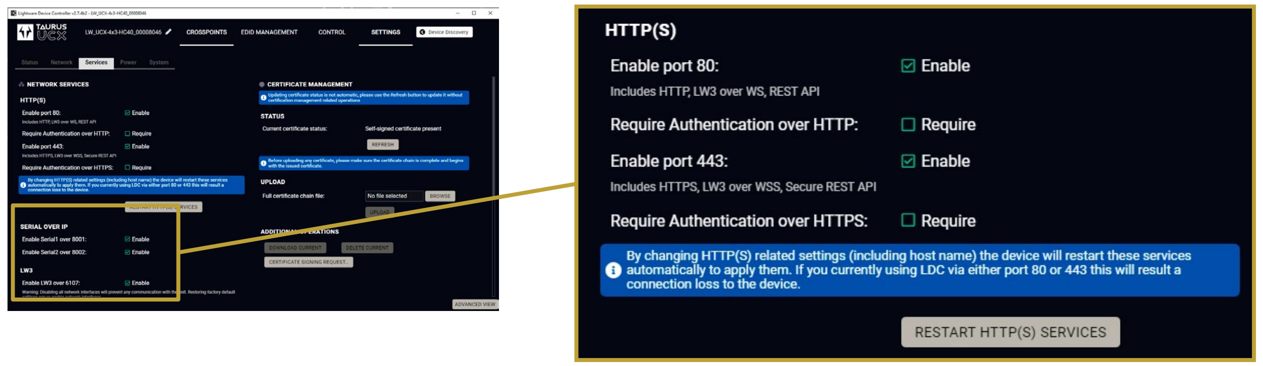
CREDENTIALS

To change password, the current password is mandatory if exists, otherwise leave it blank.

Username: admin
Current password:
New password:
Confirm new password:
Show passwords
SAVE PASSWORD

Step 2 – Enabling Port #443

- Navigate to the **Settings/Services** tab in LDC or in web LDC.
- See the Network services section and mark the **Enable port 443** setting (factory default state is **enabled**).



HTTP(S)

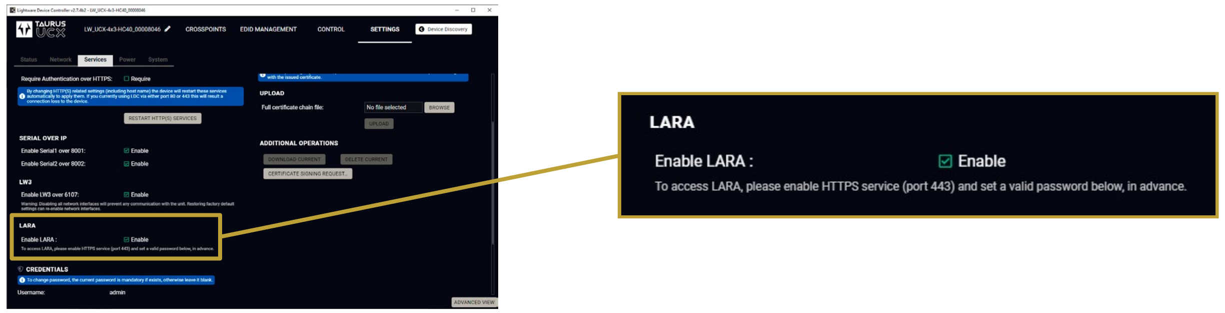
Enable port 80: ☑ Enable
Includes HTTP, LW3 over WS, REST API
Require Authentication over HTTP: ☐ Require
Enable port 443: ☑ Enable
Includes HTTPS, LW3 over WSS, Secure REST API
Require Authentication over HTTPS: ☐ Require

By changing HTTP(S) related settings (including host name) the device will restart these services automatically to apply them. If you currently using LDC via either port 80 or 443 this will result a connection loss to the device.

RESTART HTTP(S) SERVICES

Step 3 – Enabling LARA in the Device

- Navigate to the **Settings/Services** tab in LDC or web LDC.
- See the Network services section and mark the **Enable LARA** (factory default state is **disabled**).



LARA

Enable LARA : ☑ Enable

To access LARA, please enable HTTPS service (port 443) and set a valid password below, in advance.

Step 4 – Opening LARA in a Browser

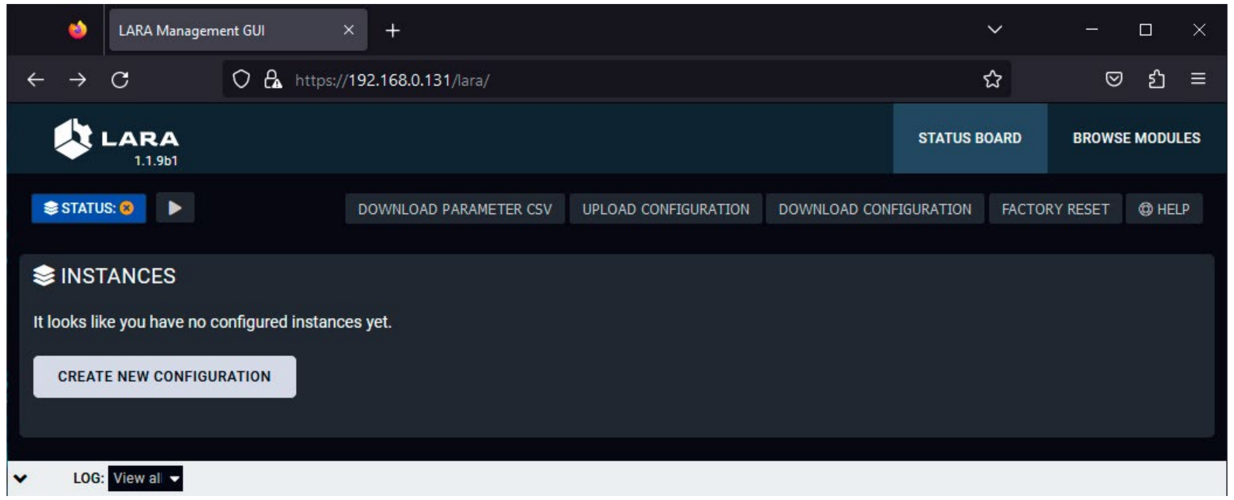
- Open a **web browser** and type the following in the address line: (make sure to type **https**)

```
https://<IP_address>/lara
```

TIPS AND TRICKS: If you select the **Settings/System** tab, you can open LARA with the **OPEN LARA** button.

After that you have to enter the user **admin** and the set **password**.

INFO: When opening LARA for the first time or after factory reset, you must accept the disclaimer statement.

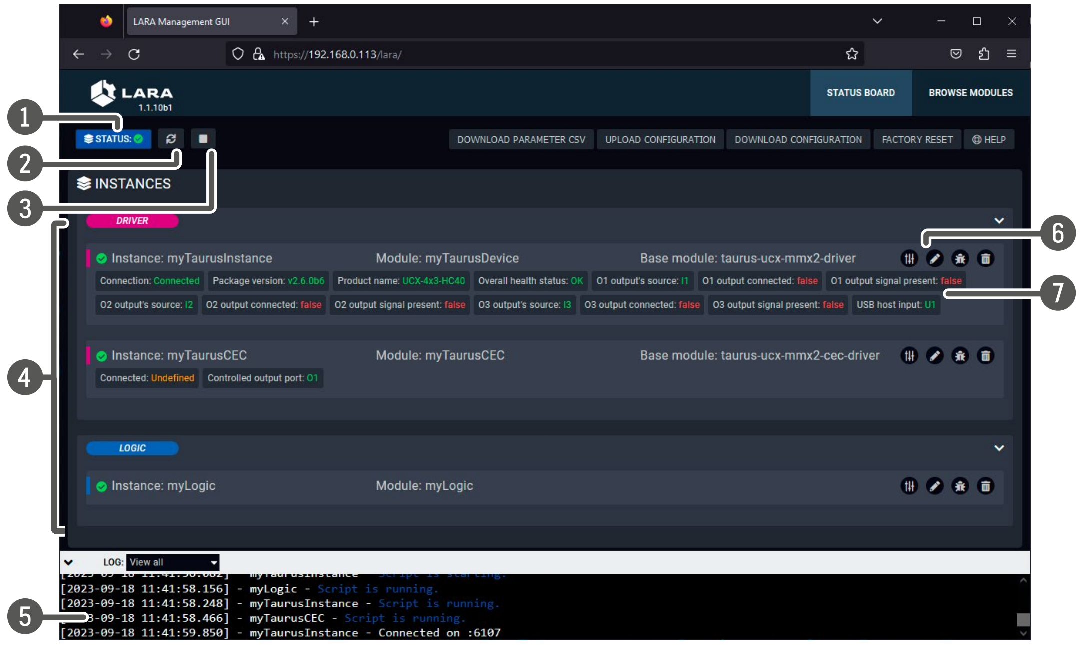


It looks like you have no configured instances yet.

The Status Board of LARA – in Factory Default State

1.5. The Main Screen

1.5.1. The Status Board Tab



Instance: myTaurusInstance Module: myTaurusDevice Base module: taurus-ucx-mmx2-driver

Connection: Connected | Package version: v2.6.0b6 | Product name: UCX-4x3-HC40 | Overall health status: OK | O1 output's source: I1 | O1 output connected: false | O1 output signal present: false | O2 output's source: I2 | O2 output connected: false | O2 output signal present: false | O3 output's source: I3 | O3 output connected: false | O3 output signal present: false | USB host input: U1

Instance: myTaurusCEC Module: myTaurusCEC Base module: taurus-ucx-mmx2-cec-driver

Connected: Undefined | Controlled output port: O1

Instance: myLogic Module: myLogic

```
[2023-09-18 11:41:58.156] - myLogic - Script is running.
[2023-09-18 11:41:58.248] - myTaurusInstance - Script is running.
[2023-09-18 11:41:58.466] - myTaurusCEC - Script is running.
[2023-09-18 11:41:59.850] - myTaurusInstance - Connected on :6107
```

1 **LARA status icon** — ✓ LARA configuration is **running**; ✗ LARA configuration is **not running**

2 **Restarting LARA** — : press to restart LARA

3 **Starting / Stopping LARA** — ▶ : press to start LARA configuration; ■ : press to stop LARA configuration

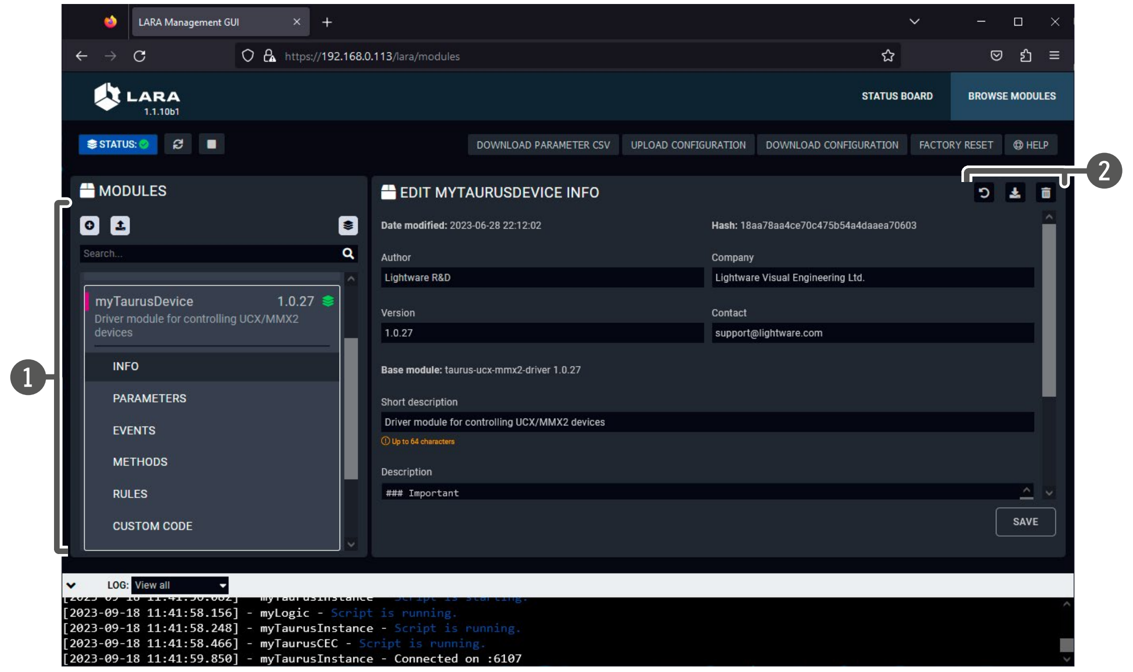
4 **Instances** — List of the currently defined instances of any modules with a status icon.

5 **Log window** — Displaying system messages but custom messages can be also displayed. The section is resizable or it can be hidden by the down arrow: ⌄

6 **Instance buttons**
- : Editing the **parameters** of the instance.
- : Editing the **parent** module.
- : Opening the **Methods and Events** testing page.
- : **Deleting** the instance.
- : QR code and link to the **Web page** of the module; available only in case of **User panel** modules.

7 **Status indicators** — Only displayed if LARA is currently running, in order to show valid information.

1.5.2. The Browse Modules Tab



myTaurusDevice 1.0.27
Driver module for controlling UCX/MMX2 devices

EDIT MYTAURUSDEVICE INFO

Date modified: 2023-06-28 22:12:02
Hash: 18aa78aa4ce70c475b54a4daaea70603

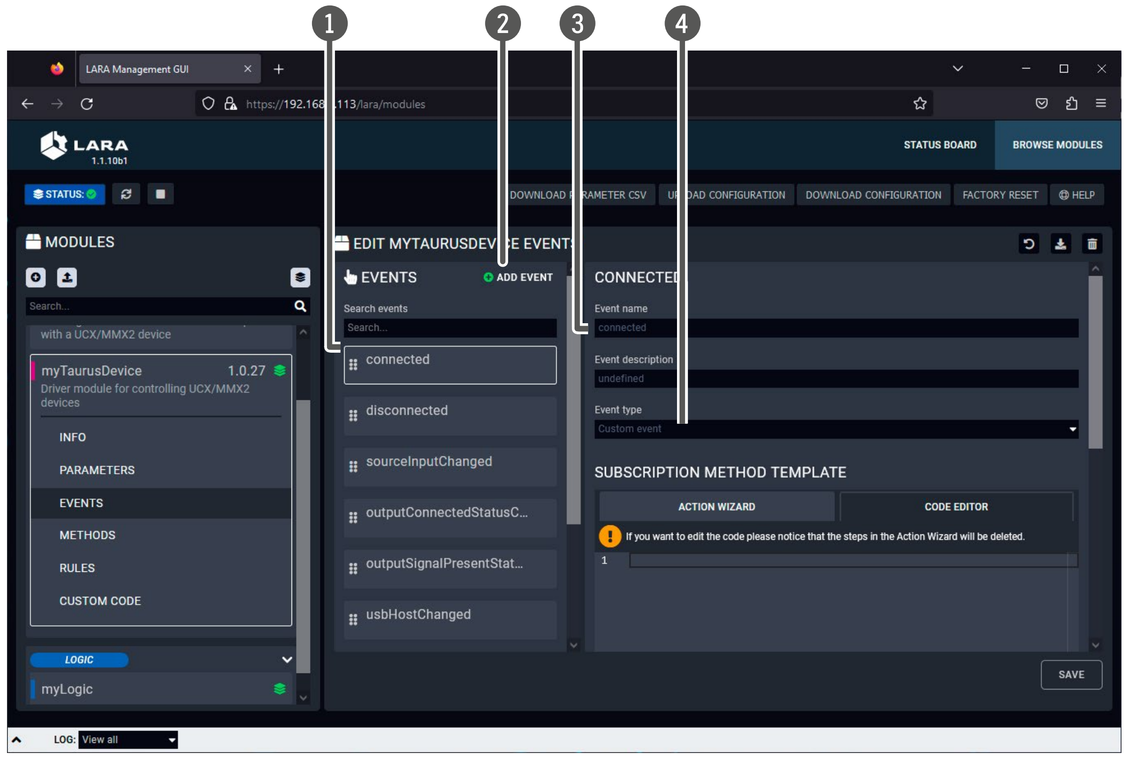
Author: Lightware R&D
Company: Lightware Visual Engineering Ltd.
Version: 1.0.27
Contact: support@lightware.com
Base module: taurus-ucx-mmx2-driver 1.0.27
Short description: Driver module for controlling UCX/MMX2 devices
Up to 64 characters
Description: ### Important

```
[2023-09-18 11:41:58.156] - myLogic - Script is running.
[2023-09-18 11:41:58.248] - myTaurusInstance - Script is running.
[2023-09-18 11:41:58.466] - myTaurusCEC - Script is running.
[2023-09-18 11:41:59.850] - myTaurusInstance - Connected on :6107
```

1 **Modules** — List of the available modules. Pressing the down arrow ⌄ the submenu is opened.
- : Creating a new module from scratch or from a base module template.
- : Uploading a new module from a ZIP file.
- : Creating a new instance from the selected module.
- : The module has no instances.
- : Instance is created from the module.

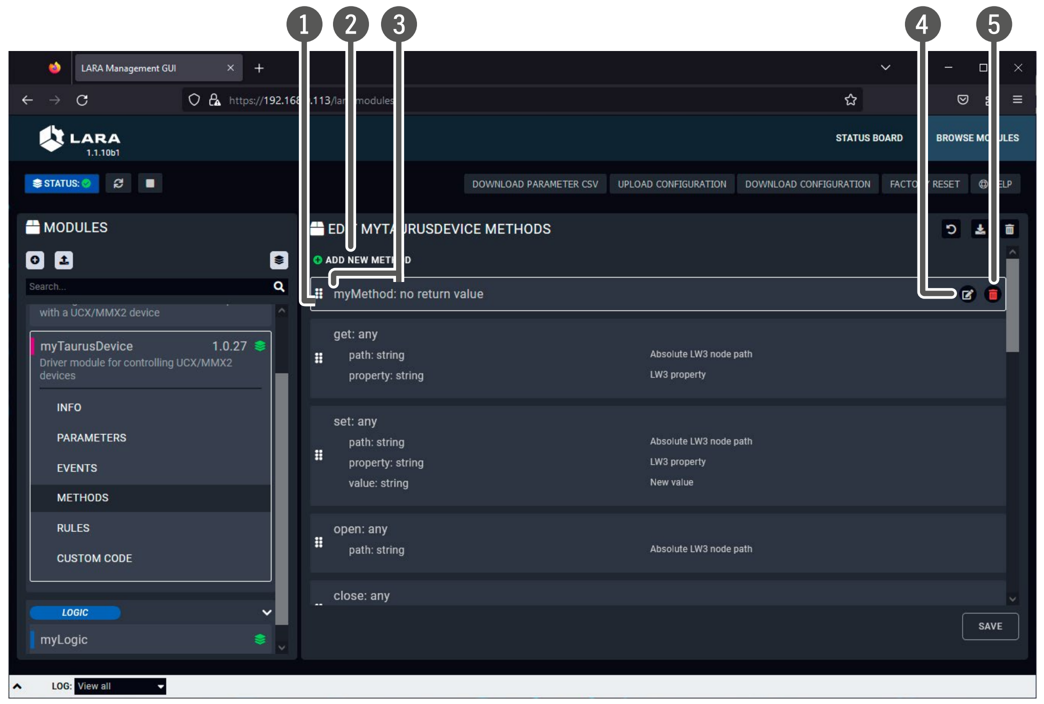
2 **Module Editing Buttons**
- : Updating the module. See more information in the Module Update section.
- : Resetting the module to the original state.
- : Downloading the module (ZIP file).
- : Deleting the module (and all of its previously created instances).

1.5.3. The Event Editor



1	**List of events**	The factory default and the user-made events are listed here. Factory events cannot be deleted.
2	**Add new event**	User-defined events can be added with this button.
3	**Name of the event**	
4	**Type of the event**	The list of the available types depends on the module. Factory default options: ▪ Custom event ▪ Periodically dispatch event ▪ Dispatch event at a specific time ▪ HTML element clicked

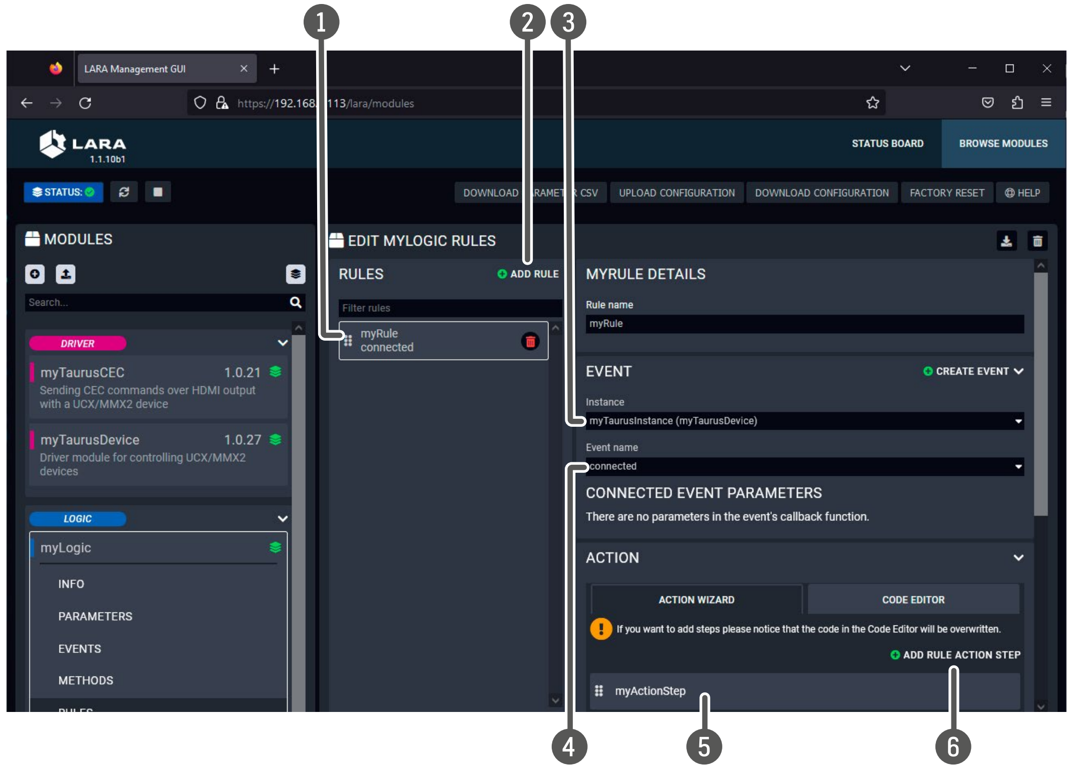
1.5.4. The Method Editor



1	**List of methods**	The factory default and the user-made methods are listed here. Factory methods cannot be deleted.
2	**Add new method**	User-defined methods can be added with this button.
3	**Name of the method**	
4	**Editing the method**	The icon is displayed if the mouse cursor is above the method.
5	**Deleting the method**	Factory methods cannot be deleted. The icon is displayed if the mouse cursor is above the method.

1.5.5. The Rule Editor

ATTENTION! A **rule** is recommended to create in the **Logic** module (except in case of User Panel module).



1	**List of rules**	The factory default and the user-made events are listed here. Factory events cannot be deleted.
2	**Add new rule**	User-defined events can be added with this button.
3	**Instance selector**	You can select an event from the current instance of the module or from another instance.
4	**Event selector**	The events of the selected instance are listed – select one of them.
5	**Action step(s)**	The defined action steps are listed here; they will be executed if above event occurs.
6	**Add new action step**	More action steps can be defined. The default action step types: **Invoke method**: You can select a pre-defined method from the desired instance. **Log message**: A message can be displayed in the log window below. **Dispatch event**: Another Event can be triggered. **Wait**: Insert a time delay between actions.

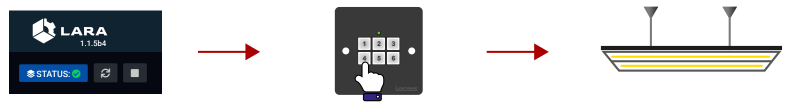
2 Sample Configuration

This chapter is about building up a whole configuration. The example starts from scratch as LARA contains no configuration.

2.1. Basic LED Toggle

What will happen?

A simple button (dry contact) is connected to GPIO PIN1 and an LED to PIN4 of the UCX Taurus device. When the button is pressed, the LED is toggled to be switched on or off.

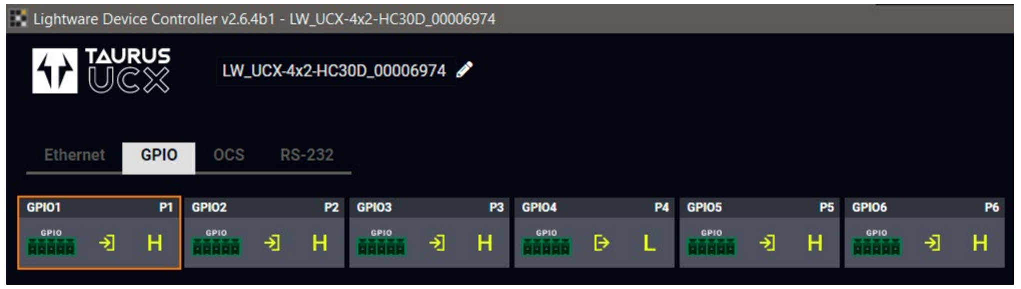


LARA starts
- Taurus is connected
- GPIO P4 set to **output**
- Log message sent

A button is pressed
- GPIO P1 becomes **low** for a moment

LED light is toggled
- GPIO P4 state is toggled from low to high or high to low
- Log message sent

TIPS AND TRICKS: If you do not have devices to connect to the GPIO, you can trigger and monitor the state changes in LDC:


Lightware Device Controller v2.6.4b1 - LW_UCX-4x2-HC30D_00006974

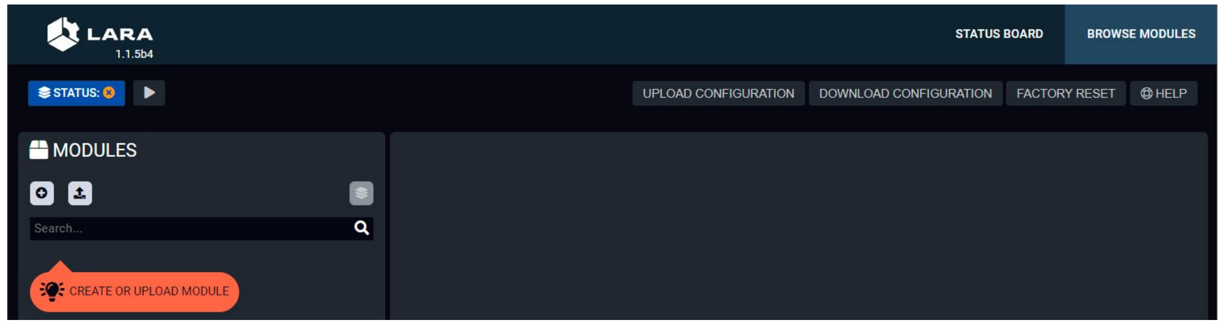
LW_UCX-4x2-HC30D_00006974

Configuration Steps

Step 1. Creating the **Driver** module(s).
Step 2. Creating the **Logic** module.
Step 3. Creating **Instances** from the Modules.
Step 4. Creating **Events** in the Drived module(s).
Step 5. Creating **Rules** in the modules.
Step 6. **Starting** the configuration.

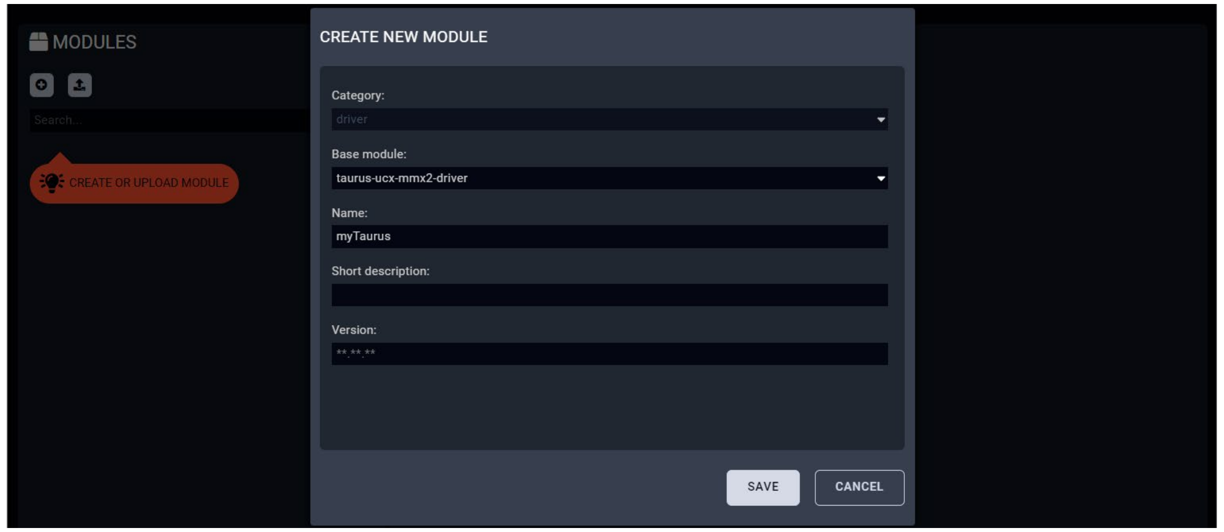
Step 1. Creating the Driver Module: *taurus-ucx-mmx2-driver*

Add new module



- Open LARA and navigate to the **Browse Modules** page.
- Create a **new module.**

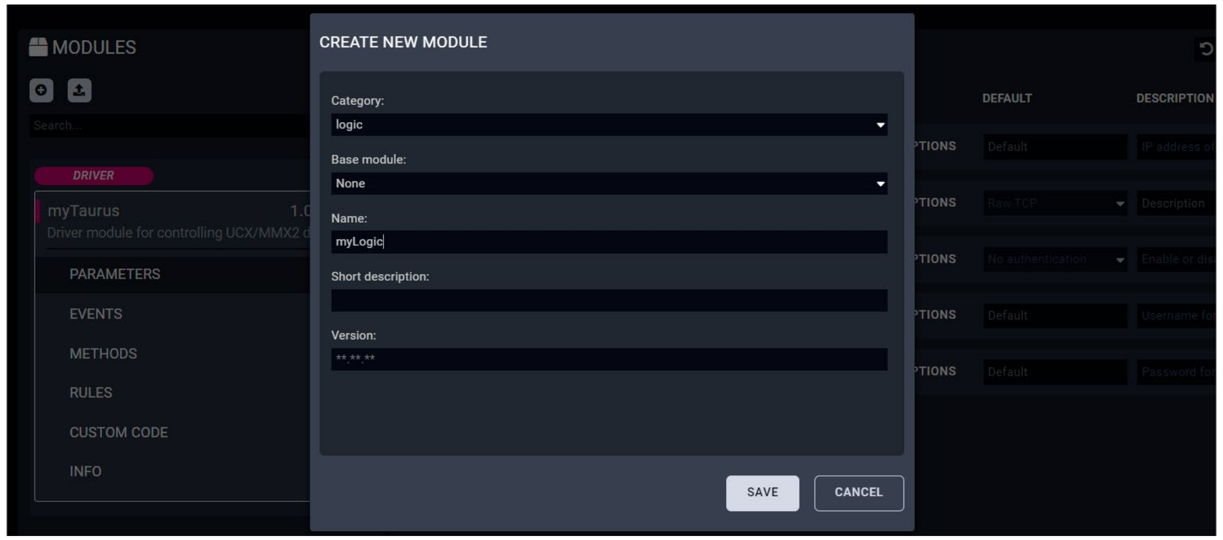
Create Taurus driver



Category: driver
Base module: taurus-ucx-mmx2-driver
Name: myTaurus
Short description:
Version: **.**.**

- Select the **Category**: driver.
- Select the **Base module**: taurus-ucx-mmx2-driver.
- Type a **name** for the module.
- Press the **Save** button.

Step 2. Creating the Logic module: *Logic*



CREATE NEW MODULE

Category: logic

Base module: None

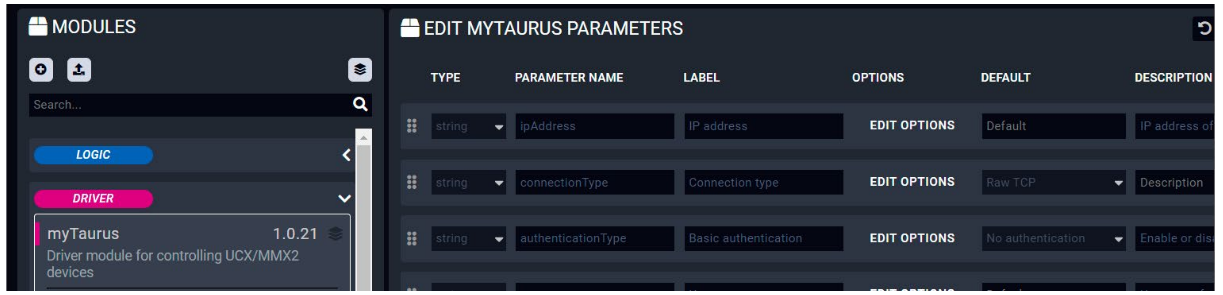
Name: myLogic

Short description:

Version: **.**.**

- Create a **new module** in the **Browse modules** page.
- Select the **Category**: logic.
- Type a **name** for the module.
- Press the **Save** button.

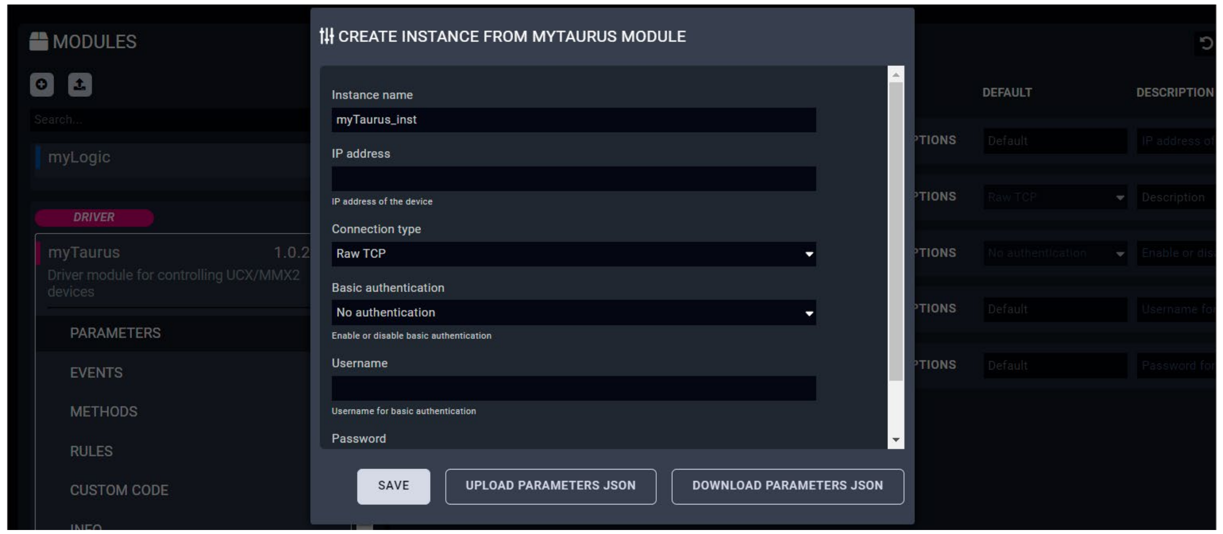
Step 3. Creating the Instances from the modules



TYPE	PARAMETER NAME	LABEL	OPTIONS	DEFAULT	DESCRIPTION
string	ipAddress	IP address	EDIT OPTIONS	Default	IP address of
string	connectionType	Connection type	EDIT OPTIONS	Raw TCP	Description
string	authenticationType	Basic authentication	EDIT OPTIONS	No authentication	Enable or dis

- Select the Taurus (myTaurus) driver in the **Browse modules** page.
- Press the icon.

Saving the *MyTaurus_inst* instance



Instance name: myTaurus_inst

IP address: (IP address of the device)

Connection type: Raw TCP

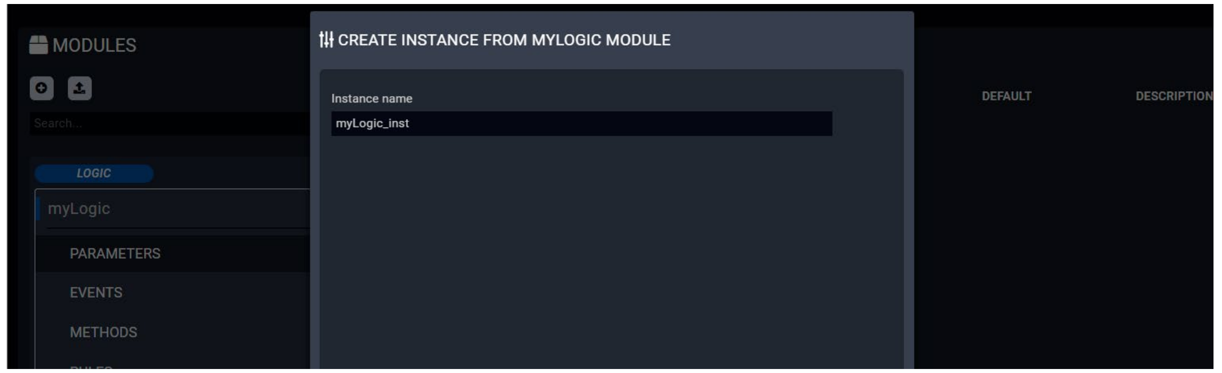
Basic authentication: No authentication (Enable or disable basic authentication)

Username: (Username for basic authentication)

Password

- Type a **name** for the instance.
- Leave the **IP address** box blank (LARA will save the 127.0.0.1 value which is the device that runs LARA).
- Press the **Save** button.

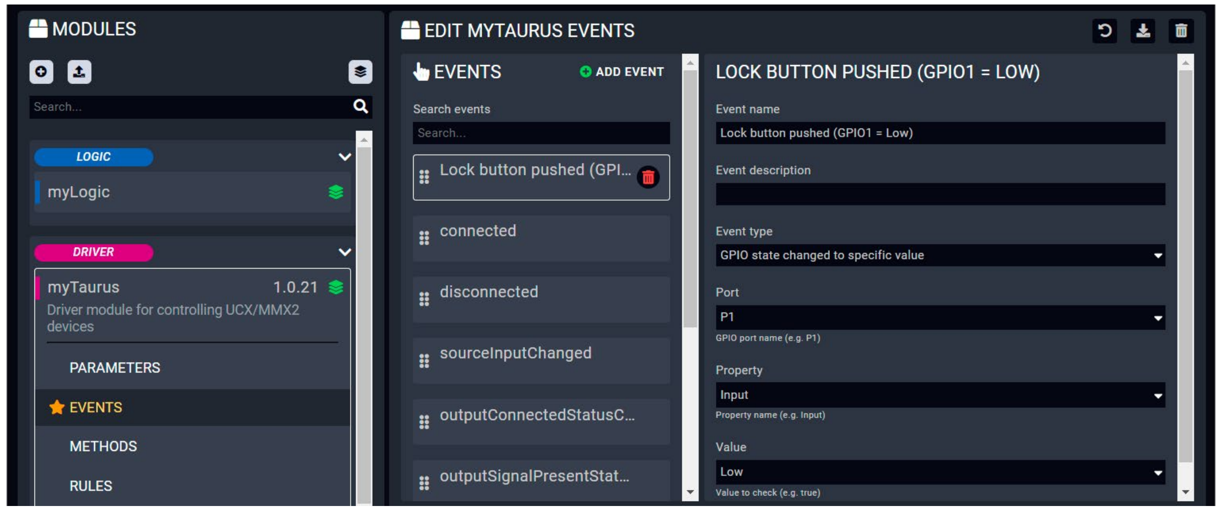
Saving the *myLogic_inst* instance



Instance name: myLogic_inst

- Select the logic module in the **Browse modules** page and press the icon.
- Type a **name** for the instance.
- Press the **Save** button.

Step 4. Creating event(s) in the Driver module



LOCK BUTTON PUSHED (GPIO1 = LOW)

Event name: Lock button pushed (GPIO1 = Low)

Event description:

Event type: GPIO state changed to specific value

Port: P1 (GPIO port name (e.g. P1))

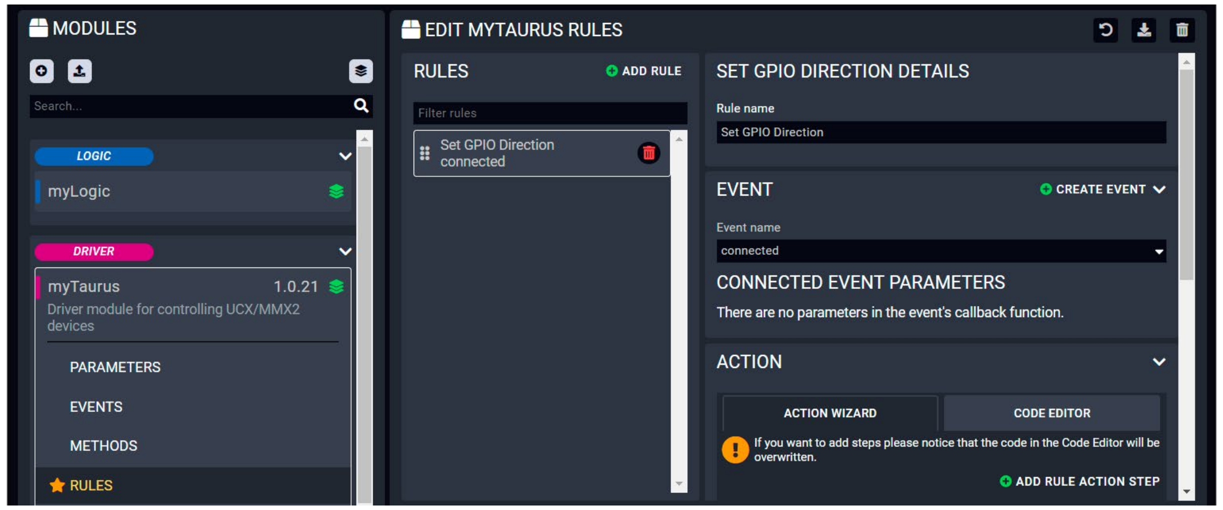
Property: Input (Property name (e.g. Input))

Value: Low (Value to check (e.g. true))

- Select the **Taurus driver (myTaurus)** in the **Browse modules** page, then navigate to the **Events** page.
- Press the **Add event** button.
- Type a **name** for the event: 'Lock button pushed (GPIO1 = Low)'.
- Select the **event type**: 'GPIO state changed to specific value'.
- Select the **port**: 'P1'.
- Select the **property**: 'Input'.
- Select the **value**: 'Low'.
- Press the **Save** button.

Step 5. Creating Rules in the modules

Initializing the GPIO pin in the Taurus when LARA starts – *Taurus* module



SET GPIO DIRECTION DETAILS

Rule name: Set GPIO Direction

EVENT

Event name: connected

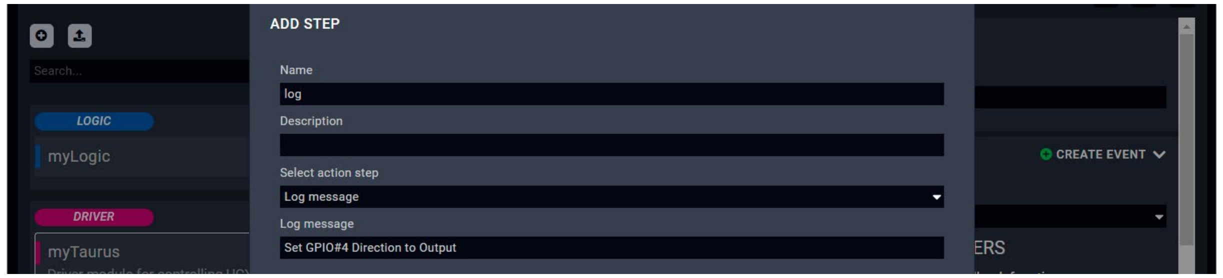
CONNECTED EVENT PARAMETERS

There are no parameters in the event's callback function.

If you want to add steps please notice that the code in the Code Editor will be overwritten.

- Select the **Taurus driver (myTaurus)** in the **Browse modules** page, then navigate to the **Rules** page.
- Press the **Add rule** button.
- Type a **name** for the rule: 'Set GPIO Direction'.
- Select the **event**: 'connected'.
- Press the **Save** button.

Adding the first action step: logging a short message



ADD STEP

Name: log

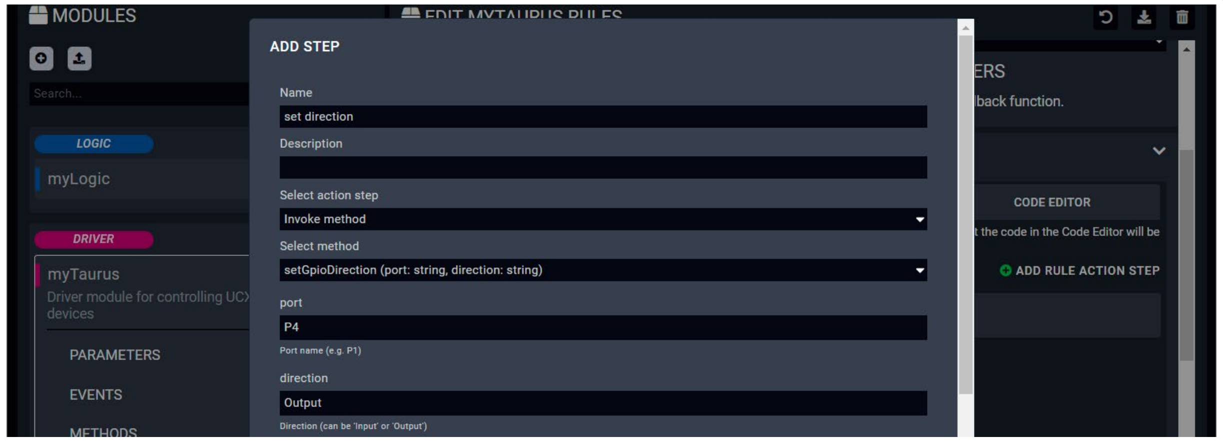
Description:

Select action step: Log message

Log message: Set GPIO#4 Direction to Output

- Press the **Add rule action step** button.
- Type a **name** for the step: 'log'.
- Select the **action step**: 'Log message'.
- Type the desired text, e.g.: 'Set GPIO4 Direction to Output'.
- Press the **Save** button.

Adding the second action step: setting the GPIO pin direction



ADD STEP

Name: set direction

Description:

Select action step: Invoke method

Select method: setGpioDirection (port: string, direction: string)

port: P4

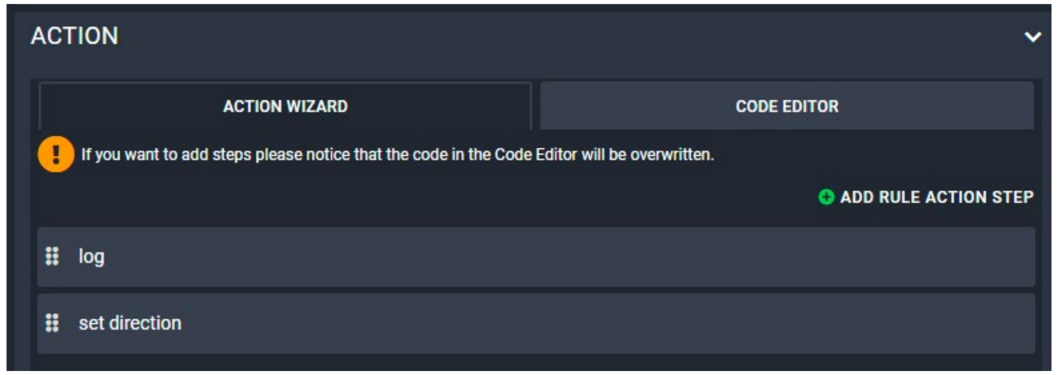
Port name (e.g. P1)

direction: Output

Direction (can be 'Input' or 'Output')

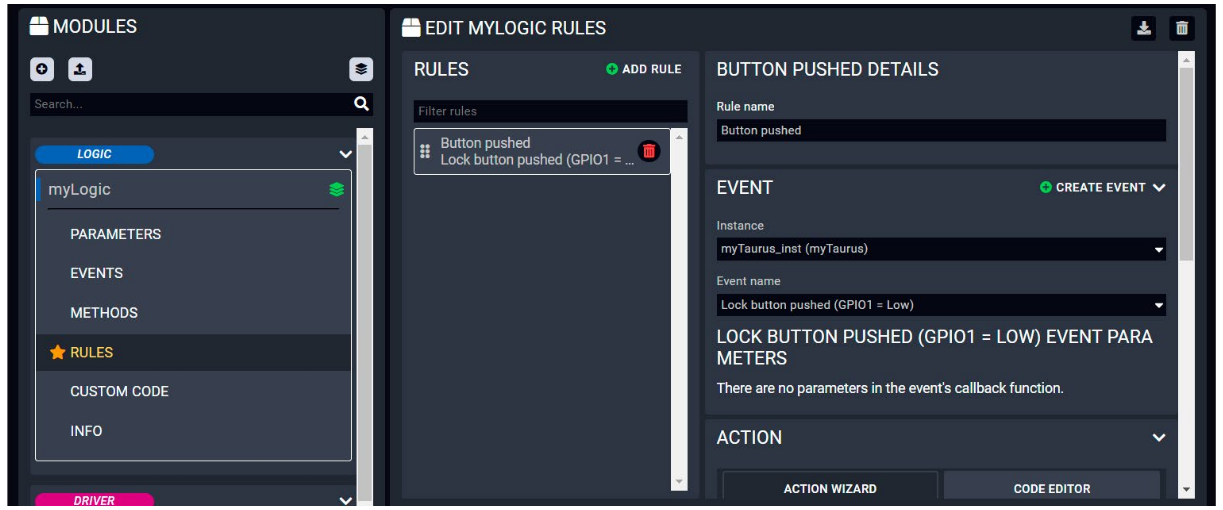
- Press the **Add rule action step** button.
- Type a **name** for the step: 'set direction'.
- Select the **action step**: 'Invoke method'.
- Select the **method**: 'setGpioDirection'.
- Type the **port** number: 'P4'.
- Type the port **direction**: 'Output'.
- Press the **Save** button in this window.
- Press the **Save** button in the main window, too.

After that, the action steps look like this:



If you want to add steps please notice that the code in the Code Editor will be overwritten.

Toggle the output level of P4 – *Logic* module



BUTTON PUSHED DETAILS

Rule name: Button pushed

EVENT

Instance: myTaurus_inst (myTaurus)

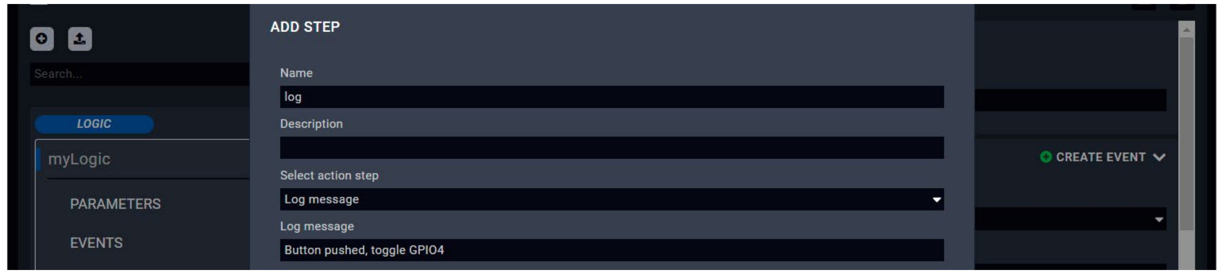
Event name: Lock button pushed (GPIO1 = Low)

LOCK BUTTON PUSHED (GPIO1 = LOW) EVENT PARAMETERS

There are no parameters in the event's callback function.

- Select the **Logic module** in the **Browse modules** page, then navigate to the **Rules** page.
- Press the **Add rule** button.
- Type a **name** for the rule: 'Button pushed'.
- Select the **instance**: 'myTaurus_inst'.
- Select the event name (that you have just created): 'Lock button pushed (GPIO1 = Low)'.
- Press the **Save** button.

Adding the first action step: logging a short message



ADD STEP

Name: log

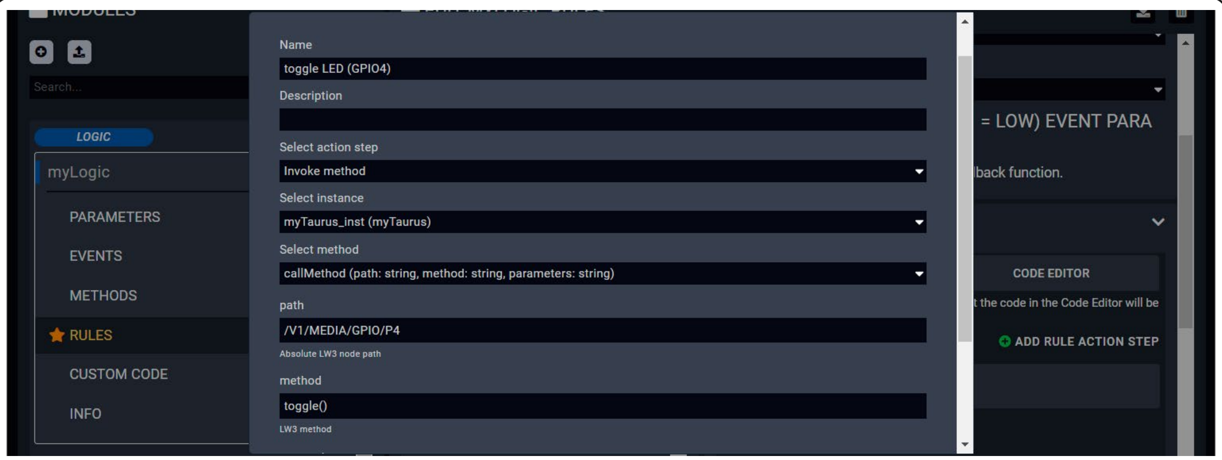
Description:

Select action step: Log message

Log message: Button pushed, toggle GPIO4

- Press the **Add rule action step** button.
- Type a **name** for the step: 'log'.
- Select the **action step**: 'Log message'.
- Type the desired **text**, e.g.: 'Button pushed, toggle GPIO4'.
- Press the **Save** button in this window.

Adding the second action step: toggle the GPIO pin output level



Name: toggle LED (GPIO4)
Description:
Select action step: Invoke method
Select instance: myTaurus_inst (myTaurus)
Select method: callMethod (path: string, method: string, parameters: string)
path: /V1/MEDIA/GPIO/P4
Absolute LW3 node path
method: toggle()
LW3 method

- Press the **Add rule action step** button.
- Type a **name** for the step: 'toggle LED (GPIO4)'.
- Select the **action step**: 'Invoke method'.
- Select the **instance**: 'myTaurus_inst'.
- Select the **method**: 'callMethod'.
- Type the **LW3 node path** of the method: '/V1/MEDIA/GPIO/P4'.
- Type the **method**: 'toggle()'.
- Press the **Save** button in this window.
- Press the **Save** button in the main window, too.

Step 6. Starting the configuration

The Configuration is ready to use, run LARA!

TIPS AND TRICKS: This configuration can be downloaded from Lightware's Cloud storage, so you can compare your work with the solution.

3 Factory Module Descriptions

This chapter is about the modules that have been developed by Lightware. The properties and the configuration steps are described in the coming sections.

3.1. Generic TCP/IP Device Module

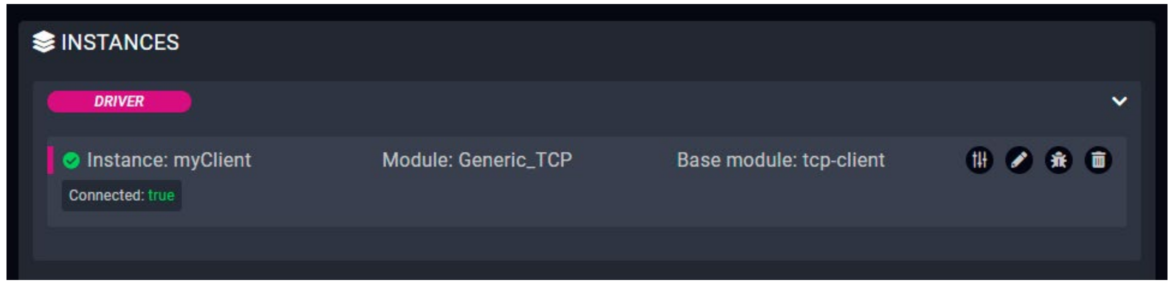
Introduction

This module is for controlling any third-party device that can be queried/controlled over TCP/IP protocol.

Dashbord Content

The following status indicator is displayed on the Status board:

- Connection state of the device



Instance: myClient Module: Generic_TCP Base module: tcp-client
Connected: true

Defined Parameters

- **ipAddressOrHost:** The IP address or the host name of the device.
- **portNumber**: Port to use when connecting.
- **permanent**: Boolean parameter to determine if the connection should be kept open and not only for sending messages (reciving data is not availabe if false).
- **autoConnect**: Boolean parameter to determine if client should automatically retry the connection if not connected.
- **autoConnectInterval**: Interval (in ms) for trying to connect.
- **frameDelimiter**: Delimiter to determine the end of a frame (e.g. `\r\n`).
- **frameDelimiterIsHex**: Specifying if the provided data is a hexadecimal string.
- **frameTimeout**: If the set time (in ms) is elapsed, the received data is considered as a frame.
- **keepAliveDelay**: Keep alive signal will be sent after the set interval time (in ms) is elapsed.

They can be referred in the JavaScript code as e.g. `params.portNumber`

Defined Events

- **connected**: The TCP/IP device specified with the ipAddress is connected.
- **disconnected**: The TCP/IP device specified with the ipAddress is disconnected.
- **error**: e.g. Problem in the data transmission, e.g. wrongly set frame delimiter. `'errormessage'` parameter is defined in this event for the error code.
- **frameReceived**: The received frame; `'frame parameter'` is defined in this event. It can be used to analyse the content of the received frame.

Defined Methods

- **send**: Sending a text message without a delimiter. `'message'` parameter is defined in this method.
- **sendFrame**: Sending a text message with a frame delimiter. `'message'` parameter is defined in this method.
- **sendHex**: Sending a message in hexadecimal format. `'message'` parameter is defined in this method.

Defined Rules

No rules are defined in this module.

Custom Code

Most of the features are written in JavaScript and can be seen in this menu.

3.2. Generic LW3 Device Module

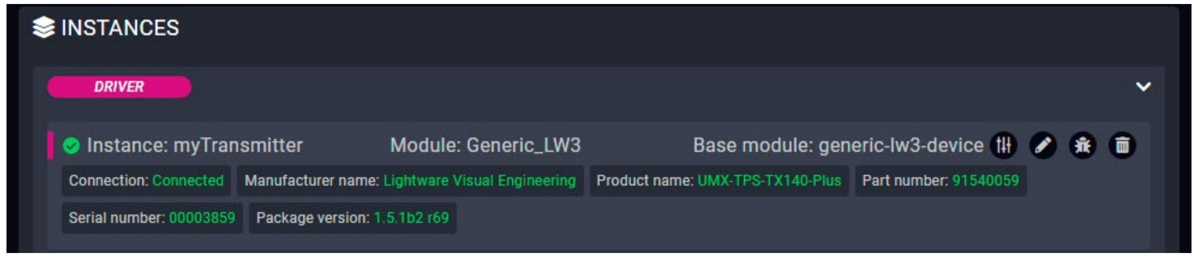
Introduction

This module is for controlling a Lightware device that supports LW3 protocol. Please refer to the User's Manual of the desired Lightware device for whether it can be controlled with LW3 commands.

Dashbord Content

The following status indicators are displayed on the Status board:

- Connection state of the device
- Manufacturer name
- Product name
- Part number
- Serial number
- FW package version


INSTANCES

DRIVER

Instance: myTransmitter Module: Generic_LW3 Base module: generic-lw3-device

Connection: Connected
Manufacturer name: Lightware Visual Engineering
Product name: UMX-TPS-TX140-Plus
Part number: 91540059
Serial number: 00003859
Package version: 1.5.1b2 r69

Defined Parameters

- **ipAddress**
- **portNumber**

They can be referred in the JavaScript code as e.g. `params.ipAddress`

Defined Events

- **connected**: The LW3 device specified with the ipAddress is connected.
- **disconnected**: The LW3 device specified with the ipAddress is disconnected.
- **LW3 property changed**: The value of a specific LW3 property is changed to the defined value. `'path'`, `'property'` and `'value'` parameters are defined in this event.

Defined Methods

- **get**: Querying the value of a specific property (with path). `'path'` and `'property'` parameters are defined in this method.
- **set**: Setting the value of a specific writable property (with path). `'path'`, `'property'` and `'value'` parameters are defined in this method.
- **open**: Subscribing to a specific node. It means that the user will get a notification if the property changes. `'path'` parameter is defined in this method.
- **close**: Unsubscribing from a specific node. `'path'` parameter is defined in this method.
- **callMethod**: Calling a specific LW3 method; does not work for an LW3 property. `'path'` and `'method'` parameters are defined in this method (parameters are optional). Please note that wildchar (*) cannot be used in case of the `'path'` parameter.

Defined Rules

No rules are defined in this module.

Custom Code

Most of the features are written in JavaScript and can be seen in this menu.

3.3. Taurus UCX/MMX2 Driver Module

Introduction

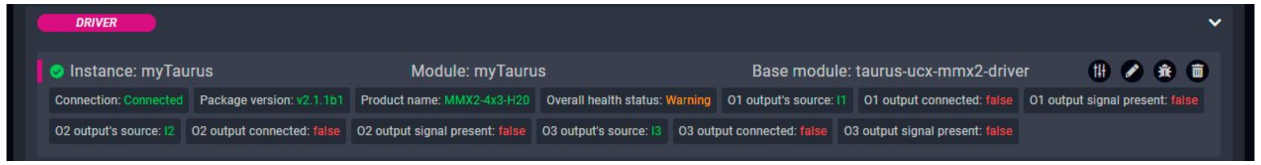
This module is for controlling Lightware's UCX and MMX2 devices with a few events and methods. The module covers the following models:

- UCX-2x1-HC30
- UCX-2x2-H30
- UCX-4x2-HC30
- UCX-4x2-HC30D
- UCX-4x3-HC40
- MMX2-4x3-H20
- MMX2-4x1-H20

Dashbord Content

The following status indicators are displayed on the Status board in the row of the instance:

- Connection state of the device
- FW package version
- Product name
- Overall health status
- Outputs' source
- Outputs' connection state
- Outputs' signal presence



Instance: myTaurus Module: myTaurus Base module: taurus-ucx-mmx2-driver

Connection: Connected | Package version: v2.1.1b1 | Product name: MMX2-4x3-H20 | Overall health status: Warning | O1 output's source: I1 | O1 output connected: false | O1 output signal present: false | O2 output's source: I2 | O2 output connected: false | O2 output signal present: false | O3 output's source: I3 | O3 output connected: false | O3 output signal present: false

Defined Parameters

- ipAddress
- connectionType
  - Raw TCP
  - Websocket
  - Secure Websocket
- authenticationType
  - no authentication
  - basic authentication
- username
- password

They can be referred in the JavaScript code as e.g. `params.ipAddress`

Defined Events

- **connected**: The UCX/MMX2 device specified with the ipAddress is connected.
- **disconnected**: The UCX/MMX2 device specified with the ipAddress is disconnected.
- **sourceInputChanged**: Another source is switched to the specified output. `'outoutPort'` and `'sourcePort'` parameters are defined in this event.
- **outputConnectedStatusChanged**: Sink device is connected/disconnected to/from the specified output. `'outputPort'` and `'status'` parameters are defined in this event.
- **outputSignalPresentStatusChanged**: The signal presence is changed on the specified output (e.g. signal is present/not present). `'outputPort'` and `'status'` parameters are defined in this event.
- **usbHostChanged**: Another USB host device is selected for the USB peripherals. `'sourcePort'` parameter is defined in this event to specify the USB input port.
- **overallHealthChanged**: The value of the /V1/MANAGEMENT/HEALTH/OverallHealthState property is changed. `'healthStatus'` parameter is defined in the event.
- **healthAlert**: The value of a specific parameter under /V1/MANAGEMENT/HEALTH node is changed to the defined value. `'property'` and `'value'` parameters are defined in this event.
- **LW3 property changed:** The value of a specific LW3 property is changed to the defined value. `'path'`, `'property'` and `'value'` parameters are defined in this event.

Defined Methods

- **get**: Querying the value of a specific property (with path). `'path'` and `'property'` parameters are defined in this method.
- **set**: Setting the value of a specific writable property (with path). `'path'`, `'property'`and `'value'` parameters are defined in this method.
- **open**: Subscribing to a specific node. It means that the user will get a notification if the property changes. `'path'` parameter is defined in this method.
- **close**: Unsubscribing from a specific node. `'path'` parameter is defined in this method.
- **callMethod**: Calling a specific LW3 method; does not work for an LW3 property. `'path'`, `'method'` and `'parameters'` are defined in this method (parameters are optional.)
- **switchCrosspoint**: Changing the crosspoint state of a specific layer (AUDIO, VIDEO or USB). Please note that USB is available only for specific models.
- **setGpioDirection**: Setting the direction (input/output) of a specific GPIO pin. `'port'` and `'direction'` parameters are defined in this method.
- **setGpioState**: Setting the state (low/high) of a specific GPIO pin. `'port'` and `'state'` parameters are defined in this method.
- **emulateEdid**: Setting an EDID (source) to emulate on a specific input port (destination). `'source'` and `'destination'` parameters are defined in this method. Source can be: factory, dynamic, user. Destination can be: emulated.
- **uploadEdid**: Uploading a HEX-format EDID to a User EDID slot. target and data parameters are defined in this method.
- **changeAnalogAudioVolumeDB**: Setting the volume level of a specific analog audio output in DB. `'port'` and `'value'` parameters are defined in this method.
- **changeAnalogAudioVolumePercent**: Setting the volume level of a specific analog audio output in percentage. `'port'`and `'value'` parameters are defined in this method.

For further details about the LW3 properties or methods, please see the User's Manual of the device.

Defined Rules

No rules are defined in this module.

Custom Code

Most of the features are written in JavaScript and can be seen in this menu.

3.4. Taurus CEC Driver Module

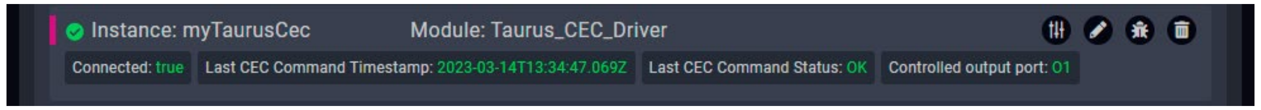
Introduction

This module is for sending CEC commands (with REST API) to a sink (display) device connected to Lightware's UCX or MMX2 devices over HDMI output.

Dashbord Content

The following status indicators are displayed on the Status board in the row of the instance:

- Connection state of the device
- Controlled output port
- Last CEC command timestamp (optional)
- Last CEC command status (optional)



Instance: myTaurusCec Module: Taurus_CEC_Driver

Connected: true | Last CEC Command Timestamp: 2023-03-14T13:34:47.069Z | Last CEC Command Status: OK | Controlled output port: O1

Defined Parameters

- **outputPort**: Video output port of the UCX/MMX2 device for CEC command requests.
- **idlePollingFrequency**: Time in seconds for CEC polling on output (0 = no idle polling).
- **ipAddressOrHost**: The IP address or the host name of the device.
- **protocol**: Protocol to be used for connection.
- **portNumber**: Port number (80 for HTTP, 443 for HTTPS) to be used for connection.
- **authenticationType**: Enable or disable basic authentication.
- **username**: User name for basic authentication.
- **password**: Password for basic authentication.
- **customHeaders**: Setting custom headers with the following syntax: header-name:value.
- **cecMessageStats**: Enable the last sent CEC message statistics on the Status Board.
- **debugMode**: Enables the debug mode for sent CEC messages. All of the details of the messages will be printed to the console.

They can be referred in the JavaScript code as e.g. `params.outputPort`.

Defined Events

- **error**: The CEC command sending was not successful.
- **availableOnCEC**: The device responds to CEC REST requests.
- **unavailableOnCEC**: Device do not respond to CEC REST requests.

Defined Methods

- **powerOn**: Sending the Power On CEC command.
- **imageViewOn**: Sending the Image View On CEC command (use this if the Power On command does not work).
- **powerOff**: Sending the Power Off CEC command.
- **standby**: Sending the Standby CEC command (use this if the Power Off command does not work).
- **mute**: Sending the Mute CEC command.
- **unmute**: Sending the Unmute CEC command.
- **volumeUp**: Sending the Volume Up CEC command.
- **volumeDown**: Sending the Volume Down CEC command.
- **customCommand**: Sending a custom command via CEC. The command parameter is defined for this purpose in this method. The custom command can be a HEX string. Only hexadecimal characters are allowed, spaces can be used as delimiters.

Defined Rules

No rules are defined in this module.

Custom Code

Most of the features are written in JavaScript and can be seen in this menu.

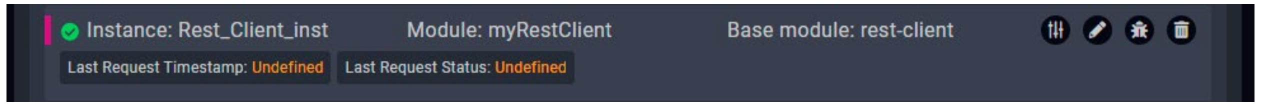
3.5. Generic Rest Client Driver Module

Introduction

This module is for handling REST API-based communication.

Dashbord Content

- **Last request timestamp**: Last request status:


Instance: Rest_Client_inst Module: myRestClient Base module: rest-client
Last Request Timestamp: Undefined Last Request Status: Undefined

Defined Parameters

- **certAuthEnable**: Enabling certificate-based authentication. Disable this when communicating with Lightware devices via HTTPS.
  - true
  - false
- **ipAddressOrHost**: IP address or host name of device.
- **protocol**: Protocol to be used for connection:
  - HTTP
  - HTTPS
- **portNumber**: 80 for HTTP, 443 for HTTPS connection.
- **authenticationType**:
  - No authentication
  - Basic authentication
- **username**: user name for basic authentication.
- **password**: password for basic authentication.
- **customHeaders**: Setting custom headers with the following syntax: header-name:value.

They can be referred in the JavaScript code as e.g. `params.username.`

Defined Events

- **error**
- **responseReceived**

Defined Methods

- **put**: PUT method. Defined parameters:
  - path: string type
  - parameters: JSON type (optional)
  - customHeaders: JSON type (optional)
- **post**: POST method. Defined parameters:
  - path: string type
  - parameters: JSON type (optional)
  - customHeaders: JSON type (optional)
- **get**: GET method. Defined parameters:
  - path: string type
  - parameters: JSON type (optional)
  - customHeaders: JSON type (optional)
- **del**: DEL method. Defined parameters:
  - path: string type
  - parameters: JSON type (optional)
  - customHeaders: JSON type (optional)

Defined Rules

No rules are defined in this module.

Custom Code

Most of the features are written in JavaScript and can be seen in this menu.

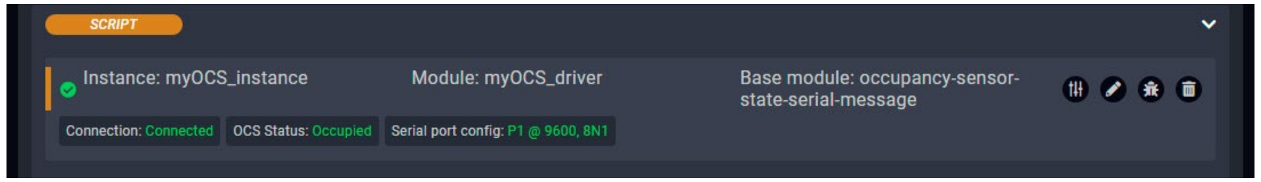
3.6. Occupancy Sensor and Serial Message Script Module

Introduction

This script module is for Lightware Taurus and MMX2 series devices for sending serial messages based on the occupancy sensor status.

Dashbord Content

- **Connection state**: UCX/MMX2 device connection state
- **OCS Status**: low/high level state
- **Serial Port config**: optional


SCRIPT
Instance: myOCS_instance Module: myOCS_driver Base module: occupancy-sensor-state-serial-message
Connection: Connected OCS Status: Occupied Serial port config: P1 @ 9600, 8N1

Defined Parameters

- **ipAddress**
- **connectionType**
  - Raw TCP
  - Websocket
  - Secure Websocket
- **authenticationType**
  - No authentication
  - Basic authentication
- **username**
- **password**
- **serialPort**: port number (e.g. P1)
- **baudRate**: Baud rate of the port (e.g. 9600)
- **stopBits**: stop bits (e.g. 1)
- **parity**: parity setting (e.g. None)
- **occupiedMessageType**:
  - string (it may contain unicode characters e.g. ¸\u000d for CR)
  - HEX
- **occupiedMessageInput**
- **freeMessageTimeout**: timeout in seconds.
- **freeMessageType**:
  - string (it may contain unicode characters e.g. ¸\u000d for CR)
  - HEX
- **freeMessageInput**: If type is HEX, these formats are accepted: A0B1... or A0 B1... or 0xA0 0xB1... or \xA0 \xB1...
- **showSerialPortConfig**: if enabled the serial port connection status icon is displayed in the dashboard.

They can be referred in the JavaScript code as e.g. `params.ipAddress`

Defined Events

- **connected**
- **disconnected**
- **error**
- **ocsStatusChanged**

Defined Methods

No methods are defined in this module.

Defined Rules

No rules are defined in this module.

Custom Code

Some features are written in JavaScript and can be seen in this menu.

3.7. Cisco Webex Module

Supported Cisco Devices

The following Cisco devices are supported by this module:

- Cisco Webex Room Kit
- Cisco Webex Room Kit Mini
- Cisco Webex Room Kit Plus
- Cisco Webex Room Kit Pro
- Cisco Webex Codec Plus
- Cisco Webex Codec Pro
- Cisco Webex Desk Pro
- Cisco Webex Room 70 G2
- Cisco Telepresence SX20
- Cisco Telepresence SX80
- Cisco DX70
- Cisco DX80
- Cisco MX700
- Cisco MX800
- Room Bar
- Room Kit EQ
- Board Pro

The Purpose of This Module

The following main benefits can be achieved by using Taurus/MMX2 devices and LARA with this module:

Bring Your Own Device (BYOD) mode: adding inputs to the Cisco Codec: thus more video sources (e.g. laptops) can be shared.

Camera share option: using the high quality camera and audio system of the Codec for other meeting purposes like a Zoom, Skype, etc. conference for a connected PC/laptop.

BYOD Mode

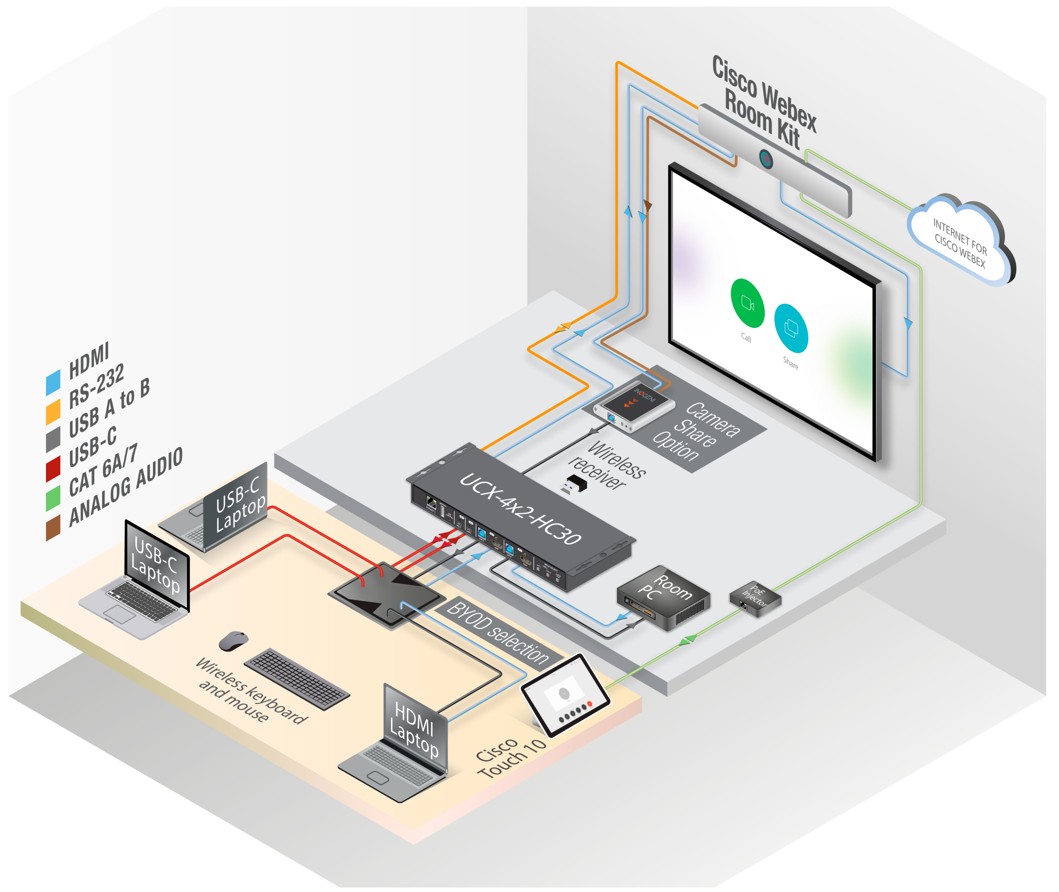
Cisco Codecs usually contain only one HDMI input. It can be increased by connecting a Taurus/MMX2 device to that input port, thus up to four devices can be connected. Furthermore, certain Taurus models contain USB-C ports, allowing another connection type. When video signal is detected on a video input port of the Taurus device, it will be visible on the UI of the Codec. After that the source can be selected as a BYOD source. Each input can be labelled with a unique name to display in the UI of the Codec.

Camera Share Option

It may happen that a video conference is taking place not via the Webex service, but with one of the particpant's laptop. The camera and the audio system of the Codec is much more suitable for this purpose than a laptop, so this feature means the camera and the audio devices can be shared to a PC/laptop connected over **HDMI and USB-B** or only **USB-C**.

3.7.1. Controlling Over Serial Connection

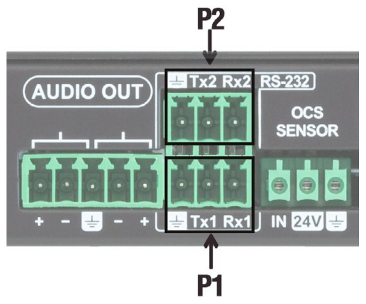
This chapter is about the settings when the Taurus and the Codec are connected via Serial port.



Checklist for the Taurus/MMX2 Device

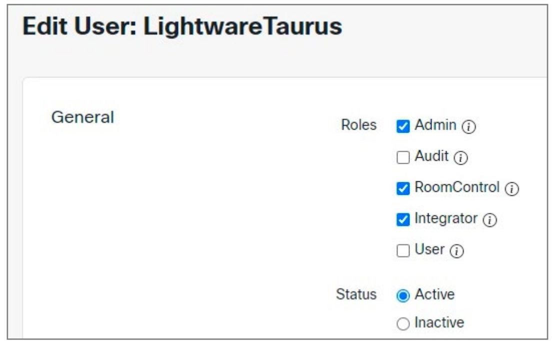
Before connecting and configuring the Taurus to a Cisco Codec, the following have to be checked:

- The **serial port settings must match** in the Taurus and in the Codec: **115200 Baud, 8 N, 1** (based on Cisco Codecs).
- Make sure the **autoselect is disabled** on the video output ports as well as on USB ports.
- Check if the **Serial over IP port** is enabled: **8001** for P1, **8002** for P2 (LDC/Settings/Network).
- Set the Video and USB-C ports to **umuted and unlocked** state.
- Pay attention to the **physical serial port**: use the port where the cable is connected to:



Checklist for the Codec

- The **serial port settings must match** in the Taurus and in the Codec.
- Create a **user name** and set a **password** in the Codec. (Note them, these should be entered when configuring the module in LARA.)
- Privileges should be **enabled** for:
  - **RoomControl**,
  - **Integrator**,
  - **Admin**.
- The following option shall be **disabled**:
  - **Require passphrase change on next user sign in.**
- If you install a **USB capture device** for camera share, e.g. Inogeni 4KXUSB3, connect it to the **last HDMI output** of the Codec.



Edit User: LightwareTaurus

General

Roles: [✓ Admin] Audit [✓ RoomControl] [✓ Integrator] User

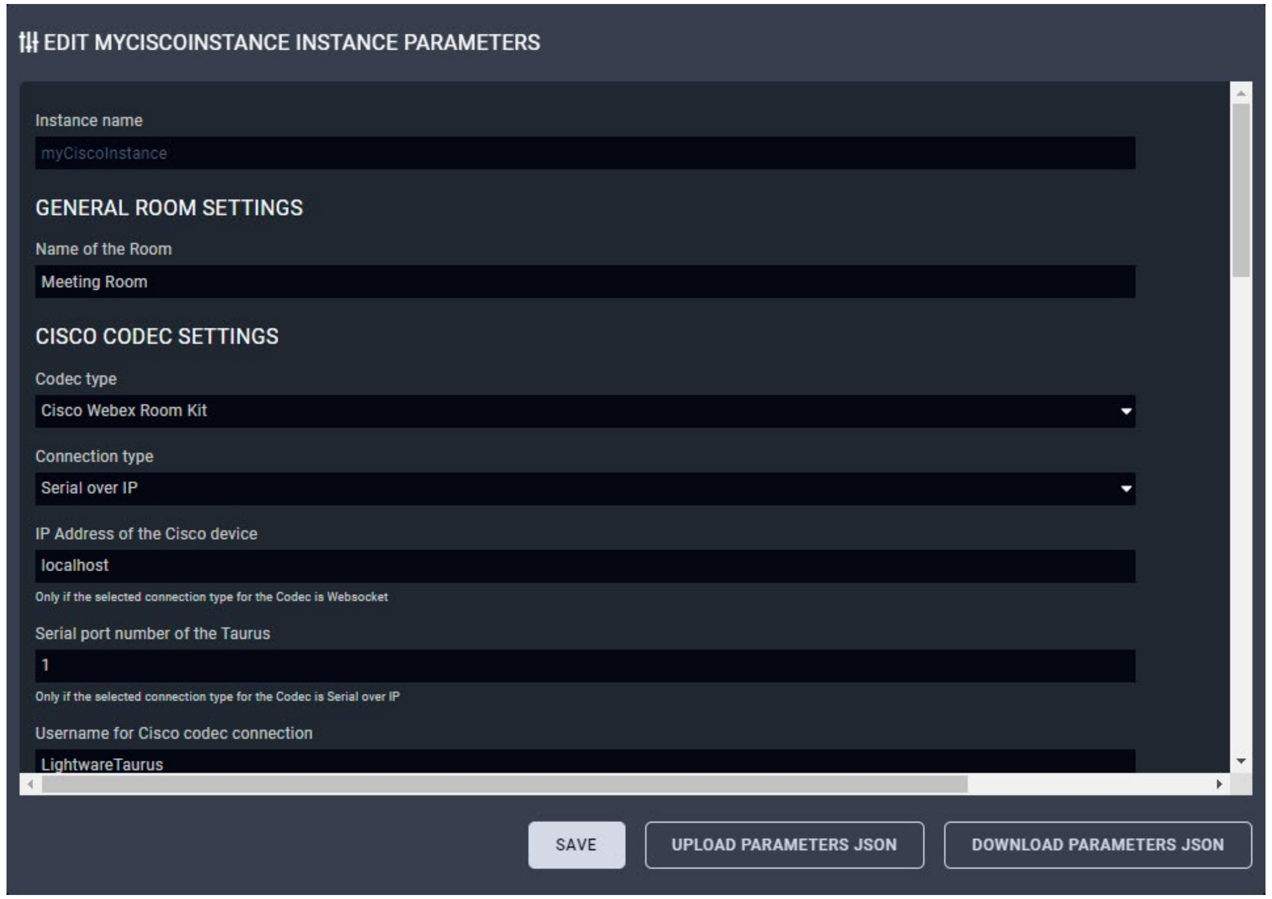
Status: [✓ Active] Inactive

INFO: Certain Cisco codecs do not have separate RS-232 port, but the **dedicated USB port** can be used for this purpose. In that case, connect a **USB-Serial (FTD) cable** between the Taurus and the Codec.

Instance Parameters

Create an Instance from the module '**ciscowebex**' and set the instance parameters as follows:

ATTENTION! Do not leave the value of the parameter in 'empty' state if it can be selected from a drop-down menu.



EDIT MYCISCOINSTANCE INSTANCE PARAMETERS

Instance name: myCiscoInstance

GENERAL ROOM SETTINGS

Name of the Room: Meeting Room

CISCO CODEC SETTINGS

Codec type: Cisco Webex Room Kit

Connection type: Serial over IP

IP Address of the Cisco device: localhost
Only if the selected connection type for the Codec is Websocket

Serial port number of the Taurus: 1
Only if the selected connection type for the Codec is Serial over IP

Username for Cisco codec connection: LightwareTaurus

Instance Name and Name of the Room

Set them as you wish.

Codec Type

Select the **Cisco model** you install.

Connection Type

Select **Serial over IP**.

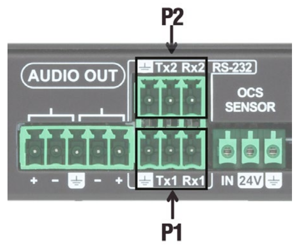
INFO: Ethernet connection is not required between the Taurus and the Codec. These packets are transferred internally, between LARA and the Taurus TCP Serial Gateway.

IP Address of the Cisco Device

Leave it as is: '**localhost**'.

Serial Port Number of the Taurus

Write the index number of the port without letter, e.g. **1** for P1 RS-232 port.



Username for Cisco Codec Connection

As set in the Codec previously.

Password for Cisco Codec Connection

As set in the Codec previously.

Cisco Codec Output Port Connection Settings (#1/#2/#3)

DIFFERENCE: The available port numbers are Codec-dependent.

- If you have a **display device** connected to a certain output, set it to '**Display**'.
- If you connect a **USB capture device** to the last HDMI output for camera share option, set the '**Last HDMI Output as USB Capture**' setting to '**true**'. In this case the Output connection setting on that output does not matter.
- Set '**no connection**' for the output that is not used.

Taurus Settings

The default settings work, no need to change.

Taurus Input Port Settings

Set the labels that will be visible on the Cisco UI (BYOD selection). If you leave a field empty, the input will not be placed to the UI as a source. ASCII characters (space also) are accepted.

ATTENTION! For Cisco-compatibility reasons the HDCP will be disabled on the inputs that are defined in the BYOD selection.

Taurus Output Port Connections

Select '**Cisco Device Input #1**' for **Output#1**. The other output selectors are reserved for future developments.

Enable USB Parallel Switching

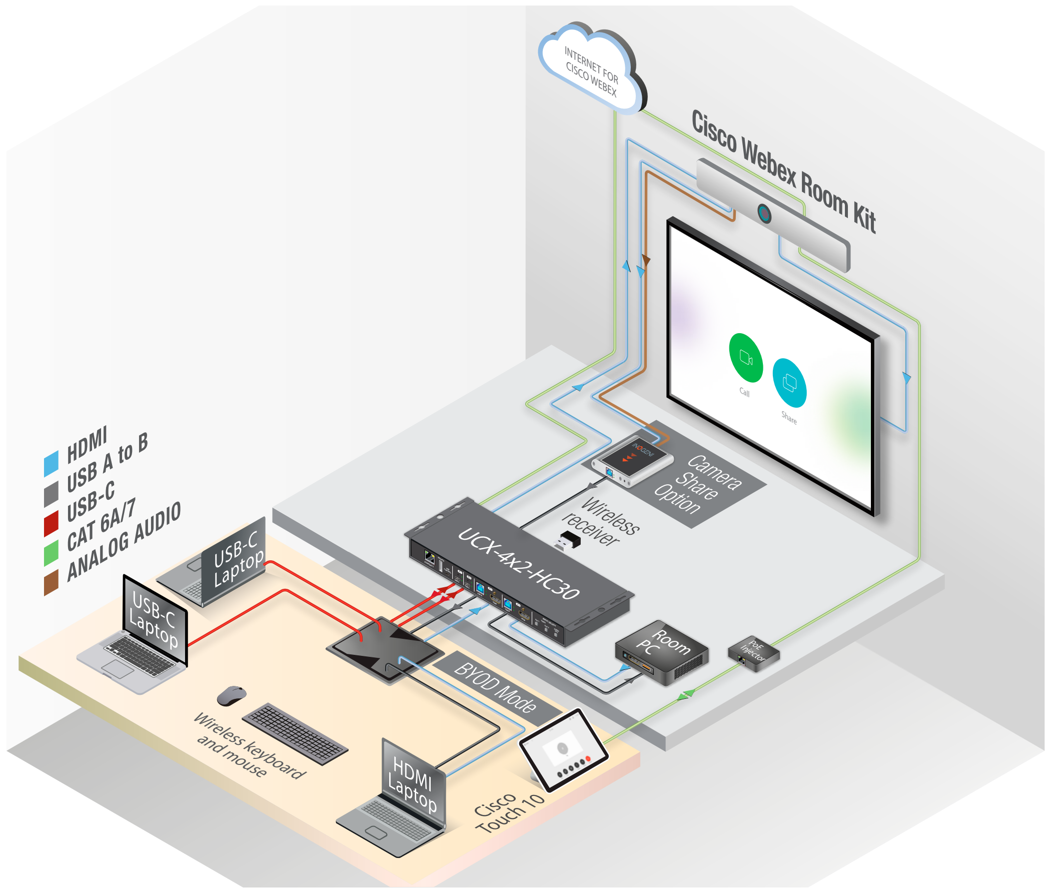
If set to 'true', the **USB** crosspoint will **follow the video** crosspoint.

Enable SignalPresent detection on inputs

If set to '**true**', the Taurus device will offer the **BYOD device selection** prompt when signal is present on the HDMI input.

3.7.2. Controlling Over Ethernet Connection

This chapter is about the settings when the Taurus and the Codec are connected via Ethernet.

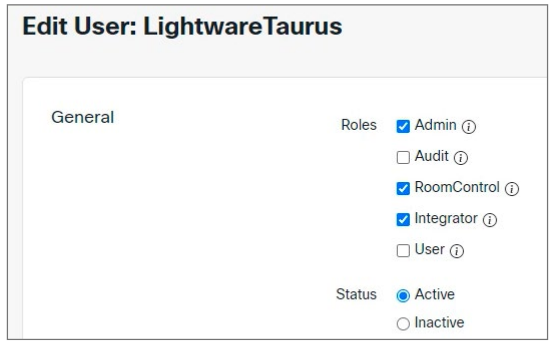


Checklist for the Taurus/MMX2 Device

- Make sure that the Taurus and the Codec can **communicate with each other over Ethernet**.
- Check the **VLAN preset settings** of the Taurus (Control/Ethernet page in LDC): the Codec and the MCU must be in the same VLAN.
- Make sure the **autoselect is disabled** on the video output ports as well as on USB ports.
- Set the Video and USB-C ports to **umuted and unlocked** state.

Checklist for the Codec

- **Websocket** has to be **enabled** (the integration works with wss (secured websocket) communication). Open the Codec **web interface:**
  - Settings → Configurations → NetworkServices → Websocket → **FollowHTTPService.**
  - Settings → Configurations → NetworkServices → HTTP → Mode → **HTTPS.**
  - **Save** the settings.
- Create a **user name** and set a **password** in the Codec. (Note them, these should be entered when configuring the module in LARA.)
- Privileges should be **enabled** for:
  - **RoomControl**,
  - **Integrator**,
  - **Admin**.
- The following option shall be **disabled**:
  - **Require passphrase change on next user sign in.**
- If you install a **USB capture device** for camera share, e.g. Inogeni 4XKUSB3, connect it to the **last HDMI output** of the Codec.



Edit User: LightwareTaurus

General

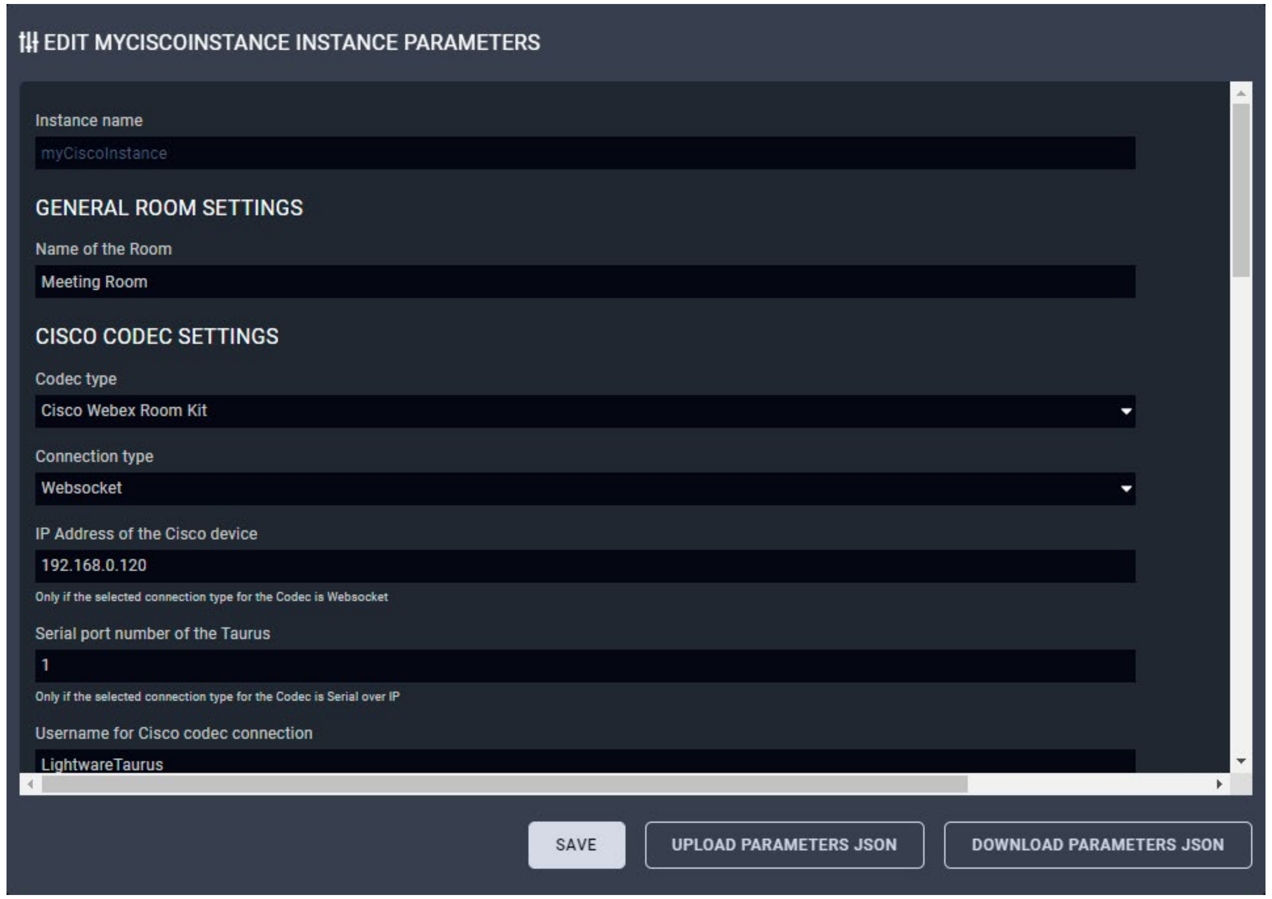
Roles: [✓ Admin] Audit [✓ RoomControl] [✓ Integrator] User

Status: [✓ Active] Inactive

Instance Parameters

Create an Instance from the module '**ciscowebex**' and set the instance parameters as follows:

ATTENTION! Do not leave the value of the parameter in 'empty' state if it can be selected from a drop-down menu.



EDIT MYCISCOINSTANCE INSTANCE PARAMETERS

Instance name: myCiscoInstance

GENERAL ROOM SETTINGS

Name of the Room: Meeting Room

CISCO CODEC SETTINGS

Codec type: Cisco Webex Room Kit

Connection type: Websocket

IP Address of the Cisco device: 192.168.0.120

Only if the selected connection type for the Codec is Websocket

Serial port number of the Taurus: 1

Only if the selected connection type for the Codec is Serial over IP

Username for Cisco codec connection: LightwareTaurus

Instance Name and Name of the Room

Set them as you wish.

Codec Type

Select the **Cisco model** you install.

Connection Type

Select **Websocket**.

IP Address of the Cisco Device

Type the IP address of the Codec.

Serial Port Number of the Taurus

N/A

Username for Cisco Codec Connection

As set in the Codec previously.

Password for Cisco Codec Connection

As set in the Codec previously.

Cisco Codec Output Port Connection Settings (#1/#2/#3)

DIFFERENCE: The available port number is Codec-dependent.

- If you have a **display device** connected to a certain output, set it to '**Display**'.
- If you connect a **USB capture device** to the last HDMI output for camera share option, set the '**Last HDMI Output as USB Capture**' setting to '**true**'. In this case the Output connection setting on that output does not matter.
- Set '**no connection**' for the output that is not used.

Taurus Settings

The default settings work, no need to change.

Taurus Input Port Settings

Set the labels that will be visible on the Cisco UI (BYOD selection). If you leave a field empty, that input will not be placed to the UI as a source. ASCII characters (space also) are accepted.

ATTENTION! For Cisco-compatibility reasons the HDCP will be disabled on the inputs that are defined in the BYOD selection.

Taurus Output Port Connections

Select '**Cisco Device Input #1**' for **Output#1**. The other output selectors are reserved for future developments.

Enable USB Parallel Switching

If set to true, the **USB** crosspoint will **follow the video** crosspoint.

Enable SignalPresent detection on inputs

If set to '**true**', the Taurus device will offer the **BYOD device selection** prompt when signal is present on the HDMI input.

3.7.3. Further Notices

Webex Mode

Webex mode is activated if:

- The Codec is started, or
- All devices are disconnected from the Codec.

INFO: **Half Wake** and **Standby** modes are active in Webex mode, but in BYOD mode these modes are inactive.

After Restart

If the Codec, the UCX/MMX2 or LARA is restarted for some reason, the system must recover in the same working state as it has been previously.

3.7.4. Known Issues

'Preset does not exist'

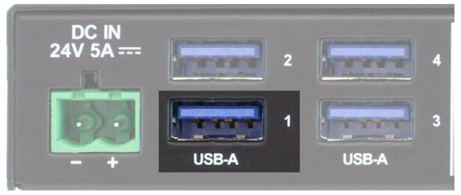
This message may appear in the log window when the Codec is a Cisco Room Kit Mini. This entry does not mean an error.

'Unknown widget: 'byodselector'

This message may appear in the log window when the Codec is a Cisco Room Kit Mini. This entry does not mean an error.

For camera sharing option

Please connect the USB capture device (or the RoomKit Mini USB-C cable) to the Taurus port USB-A #1:



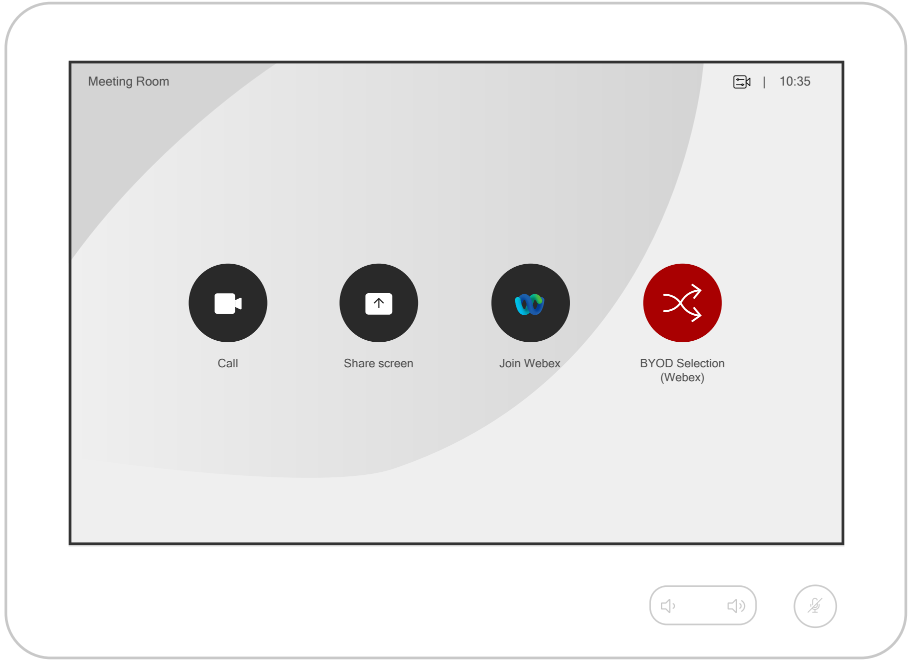
Useless Button

It may happen in case of Room Kit Mini and Room Kit Bar Codecs that the UI contains a button with the following label: 'Call from laptop'. This button has no function in this case.

3.7.5. Integration with the Cisco Codec

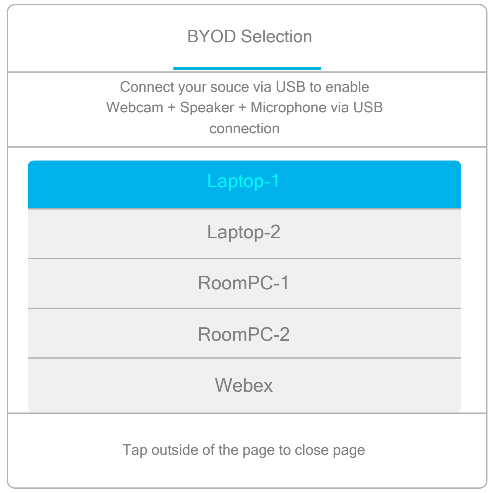
Main Menu

After a successful setup a new button will appear in the Main menu of the Codec: **BYOD Selection**.



BYOD Selection

After pressing the '**BYOD Selection**' button, a window appears on the UI of the Codec. On the bottom of the screen the defined and available inputs can be seen with the labels set in the instance *.


BYOD Selection

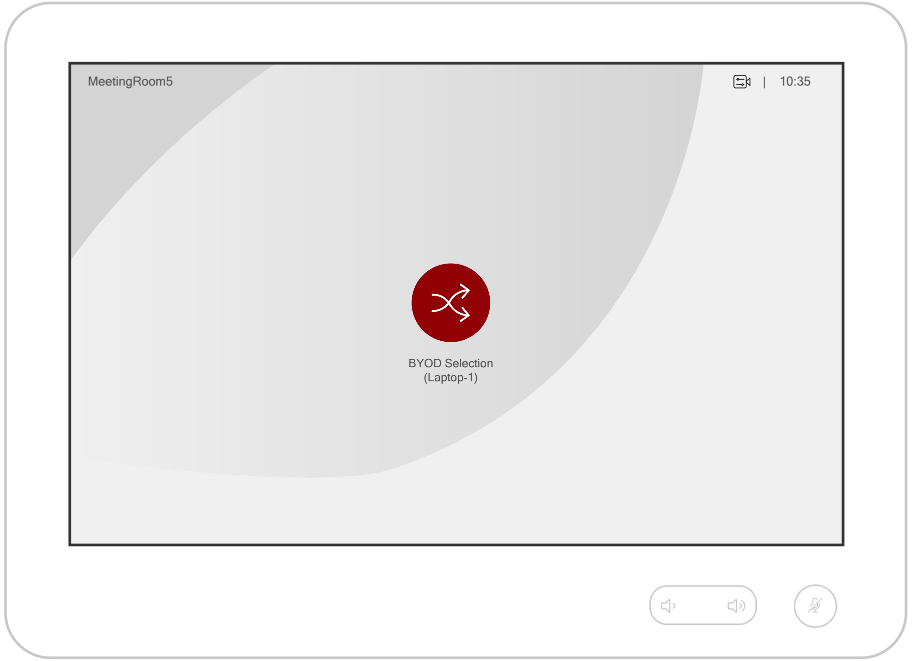
Connect your souce via USB to enable Webcam + Speaker + Microphone via USB connection

Laptop-1
Laptop-2
RoomPC-1
RoomPC-2
Webex

Tap outside of the page to close page

* If there is no USB layer in the Lightware device, the label below the 'BYOD selection' is: 'Connect your source via HDMI'.

When a source is selected:



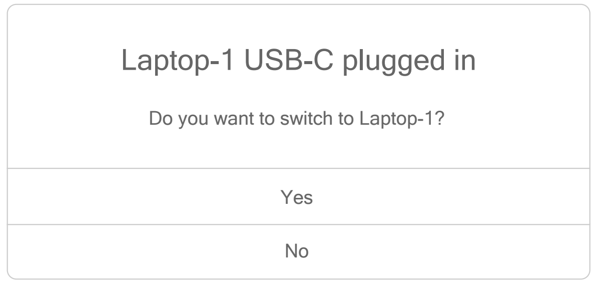
- The video signal of the selected source is **displayed on all outputs** of the Codec except the last HDMI output if a USB capture device is installed on that port.
- Not applicable buttons will be **hidden in the main menu**.
- The **USB crosspoint** of the Taurus UCX device **will follow** the video crosspoint state if the 'Enable USB parallel switching' option is 'true'.
- The '**SpeakerTrack**' feature of the camera is turned on – if it is supported.
- The **microphone signal** will be sent via the analog audio output of the Codec to the USB capture device. *
- The '**Do not disturb**' mode of the Codec will be switched on with default timeout (24 h).
- The automatic **standby mode** is deactivated in the Codec.

When selecting '**Webex**' again:

- The video output of the Codec will be the **default UI** content.
- Previously **hidden buttons will appear** again in the menu.
- The '**SpeakerTrack**' feature of the camera will be still on – if it is supported.
- The **analog audio output** of the Codec will be switched off. *
- The '**Do not disturb**' mode of the Codec will be switched off.
- The '**Halfwake**' mode must be activeted in the Codec after 2 minutes of idle time.

* Does not refer to Cisco Webex Room Kit Mini.

If a new source is connected to the UCX/MMX2 device, the following window appears:


Laptop-1 USB-C plugged in

Do you want to switch to Laptop-1?

4 User Module Descriptions

This chapter is about the modules that have been developed by Lightware. The properties and the configuration steps are described in the coming sections.

4.1. Touchscreen UI Module

Introduction

The module offers ready-to-use user interface templates for touchscreen devices, which provide controlling various type of meeting or huddle rooms.

4.1.1. Templates

The following templates are available in the software:

Devices

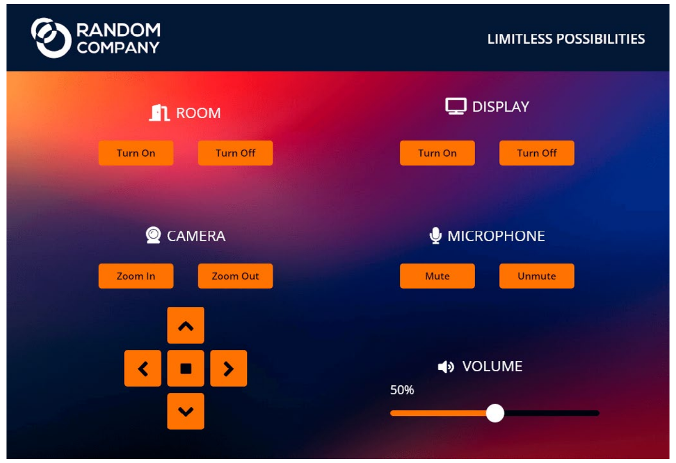
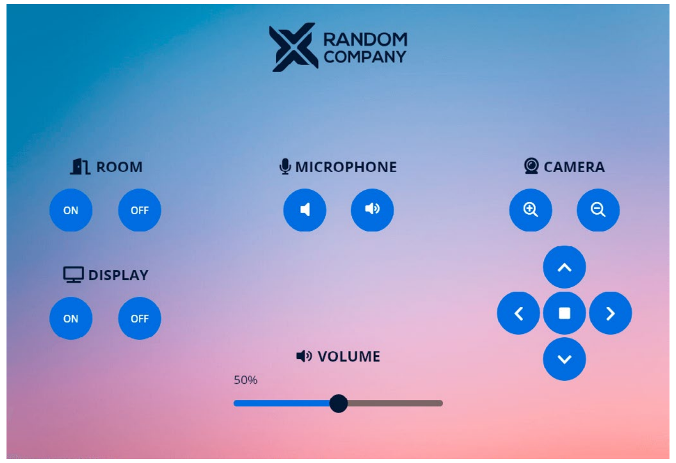
- **Room**: On / Off
- **Camera**: Move Up / Down / Right / Left / Zoom in / Zoom out
- **Microphone**: Mute / Unmute
- **Display**: Turn on / Turn off
- **Volume**: Increase / Decrease
- **Lights**: On / Off

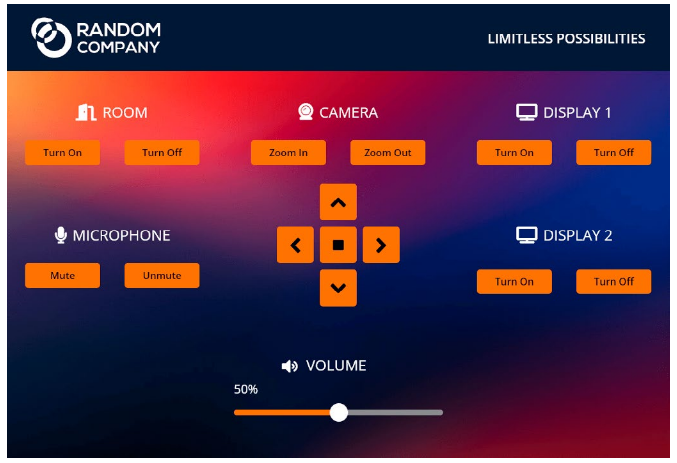
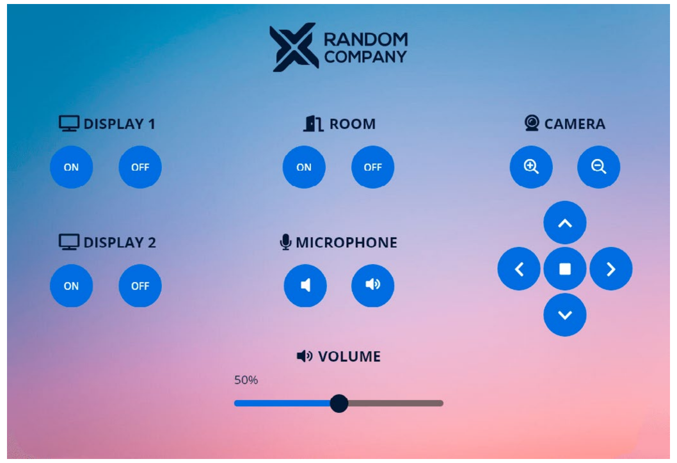
Designs

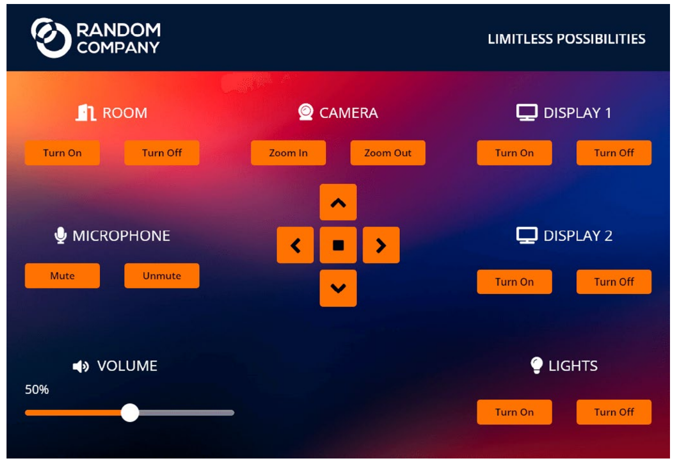
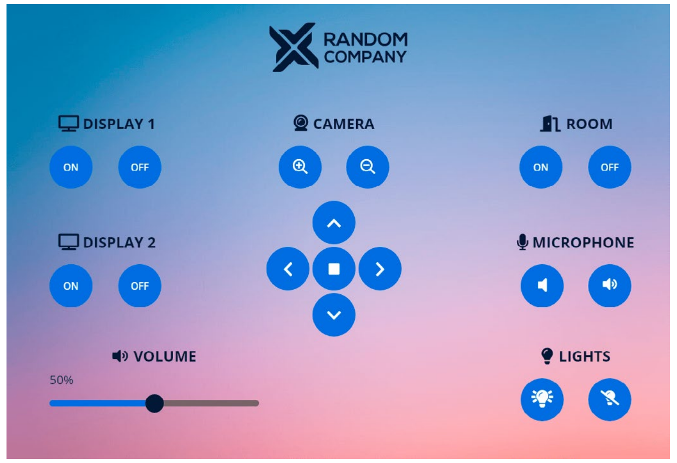
- 2 different design sets
- Apply each design set for all above templates

TIPS AND TRICKS: The built-in templates can be customized freely and users can add new buttons and features to the touchpanel as well. See more personalization samples in the Personalizing of the UI section.

User Panel Templates

	Devices in the Template	Touchscreen UI Preview
Huddle Room	**Dark theme** ▪ Room ▪ Camera ▪ Microphone ▪ Display ▪ Volume	 RANDOM COMPANY LIMITLESS POSSIBILITIES ROOM Turn On Turn Off DISPLAY Turn On Turn Off CAMERA Zoom In Zoom Out MICROPHONE Mute Unmute VOLUME 50%
	Light theme ▪ Room ▪ Camera ▪ Microphone ▪ Display ▪ Volume	 RANDOM COMPANY ROOM ON OFF MICROPHONE CAMERA DISPLAY ON OFF VOLUME 50%

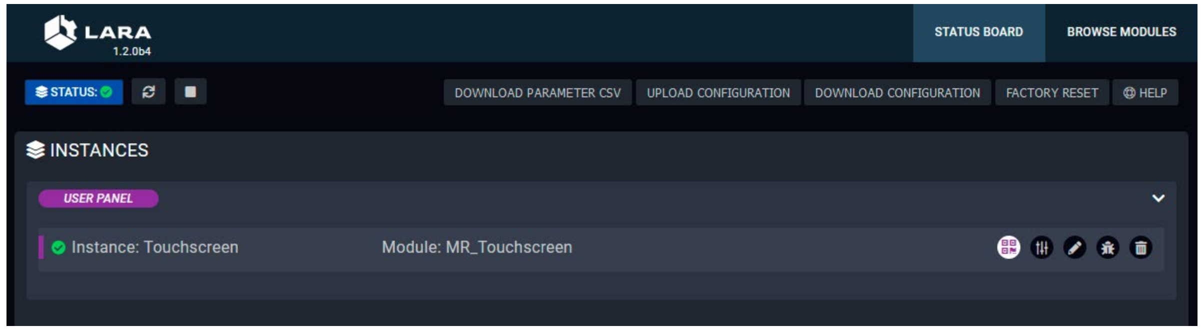
	Devices in the Template	Touchscreen UI Preview
Meeting Room	**Dark theme** ▪ Room ▪ Camera ▪ Microphone ▪ 2 Displays ▪ Volume	 RANDOM COMPANY LIMITLESS POSSIBILITIES ROOM Turn On Turn Off CAMERA Zoom In Zoom Out DISPLAY 1 Turn On Turn Off MICROPHONE Mute Unmute DISPLAY 2 Turn On Turn Off VOLUME 50%
	Light theme ▪ Room ▪ Camera ▪ Microphone ▪ 2 Displays ▪ Volume	 RANDOM COMPANY DISPLAY 1 ON OFF ROOM ON OFF CAMERA DISPLAY 2 ON OFF MICROPHONE VOLUME 50%

	Devices in the Template	Touchscreen UI Preview
Meeting Room with Light Control	**Dark theme** ▪ Room ▪ Camera ▪ Microphone ▪ 2 Displays ▪ Volume	 RANDOM COMPANY LIMITLESS POSSIBILITIES ROOM Turn On Turn Off CAMERA Zoom In Zoom Out DISPLAY 1 Turn On Turn Off MICROPHONE Mute Unmute DISPLAY 2 Turn On Turn Off VOLUME 50% LIGHTS Turn On Turn Off
	Light theme ▪ Room ▪ Camera ▪ Microphone ▪ 2 Displays ▪ Volume	 RANDOM COMPANY DISPLAY 1 ON OFF CAMERA ROOM ON OFF DISPLAY 2 ON OFF MICROPHONE VOLUME 50% LIGHTS

4.1.2. Dashboard Content

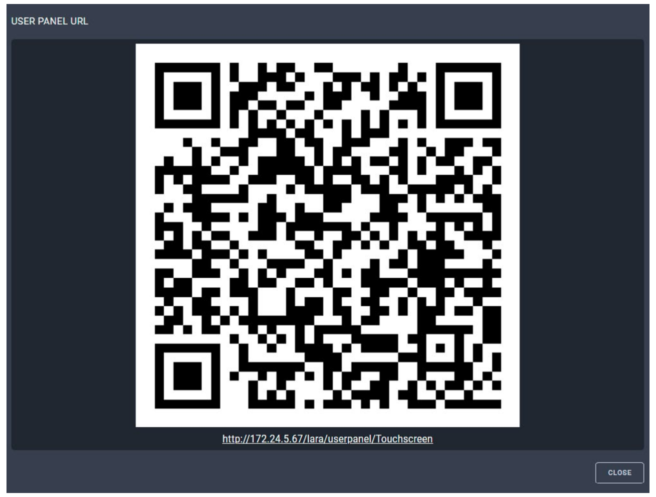
The following status indicator is displayed on the Status board:

- QR code and link to the Touchscreen UI



Instance: Touchscreen Module: MR_Touchscreen

Clicking on the icon pops up the scanable QR code and the link of the touchscreen UI.

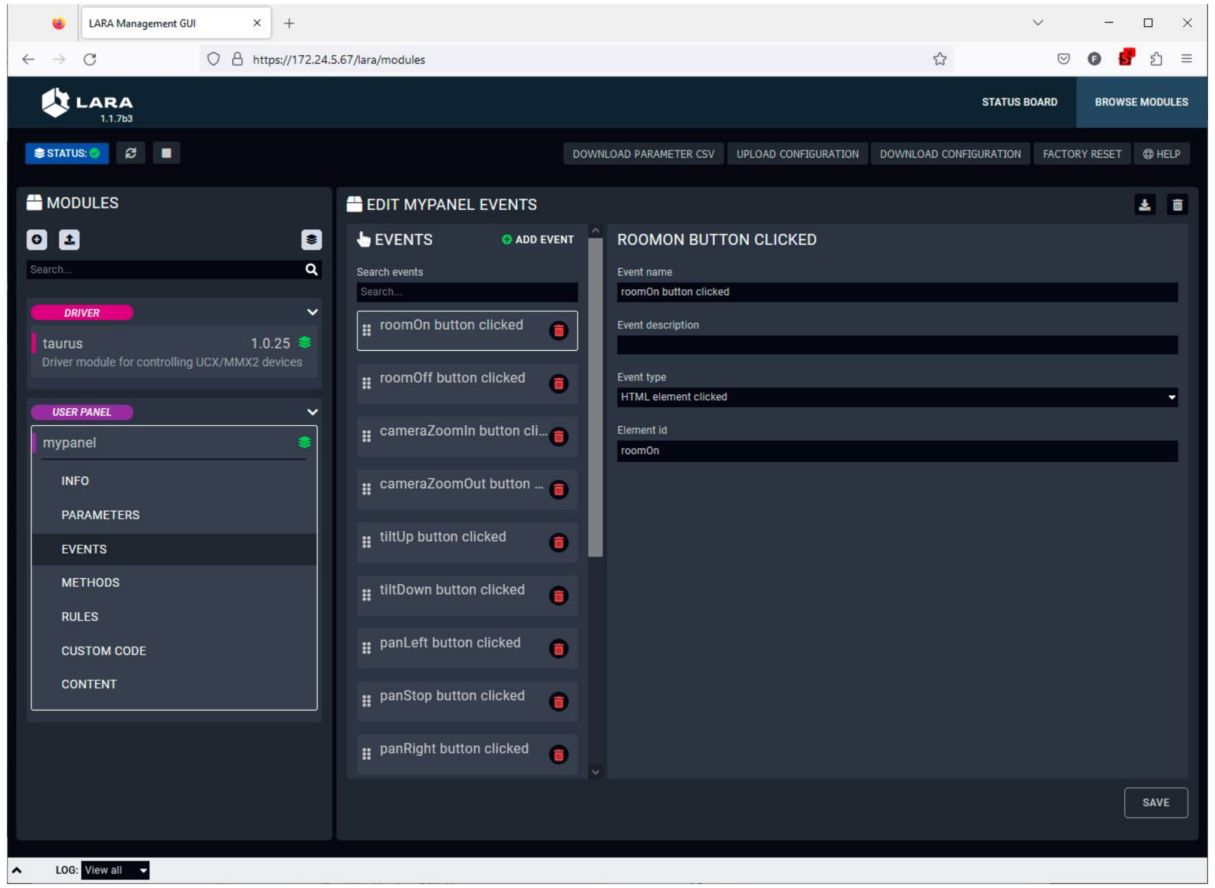


http://172.24.5.67/lara/userpanel/Touchscreen

4.1.3. Editing of the Module

Click on the icon to edit the parameters of the module and to get more information about it.

The following sections are available in the user panel module:



taurus 1.0.25
Driver module for controlling UCX/MMX2 devices

mypanel

Events:
- roomOn button clicked
- roomOff button clicked
- cameraZoomIn button cli...
- cameraZoomOut button ...
- tiltUp button clicked
- tiltDown button clicked
- panLeft button clicked
- panStop button clicked
- panRight button clicked

ROOMON BUTTON CLICKED

Event name: roomOn button clicked
Event description:
Event type: HTML element clicked
Element id: roomOn

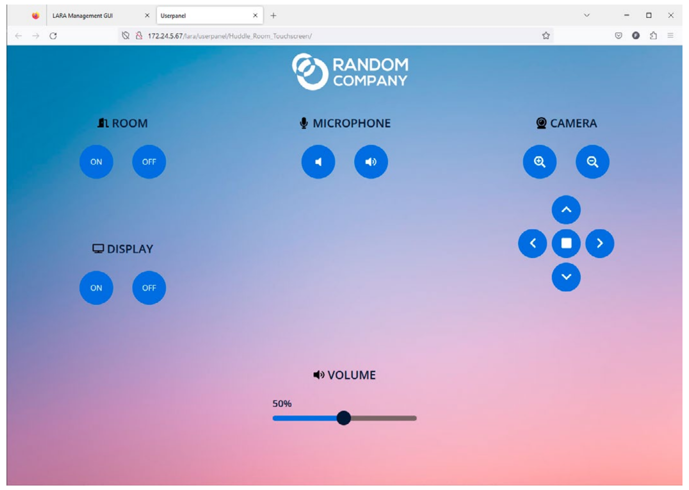
- **Info**: it can be filled with basic user information about the user panel like author, version, etc.
- **Parameters**: parameters can be added to module which can be used for the user panel. User defined parameters can be used in Javascript code only.
- **Events**: predefined events can be cusomized or new events can be added to the user panel interactions and a specific change (user interaction or parameter change) that can be used as a trigger.
- **Methods**: methods can be added to module and actions to run when an event is triggered.
- **Rules**: predefined rules can be cusomized or new rules can be added to the user panel interactions. In case of User Modules, the rule can be arranged in the module, Logic module is not necessary.
- **Custom code**: custom user code in javascript can be uploaded.
- **Content**: the original HTML and CSS code of the chosen template can be reviewed and customized here.

4.1.4. Personalizing of the UI

This section shows how to personalize the touchscreen UI templates through some simple examples. The **Huddle Room Light template** will be used in the following samples.

Changing of the Company Logo

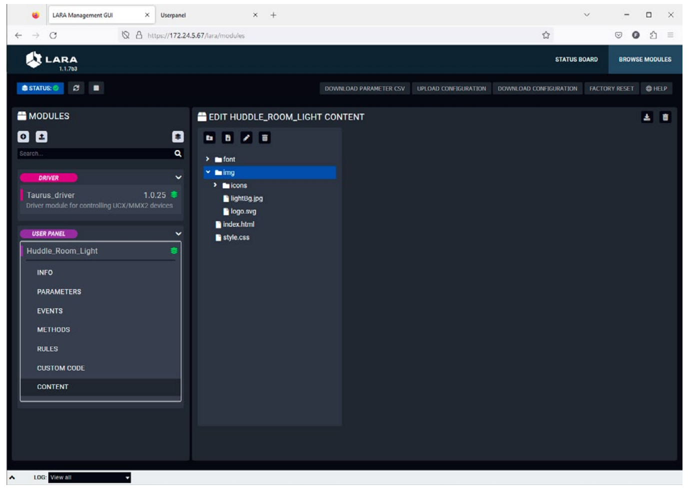
The template contains a logo which is displayed on the touchscreen UI by default. The logo can be changed easily in few simple steps.



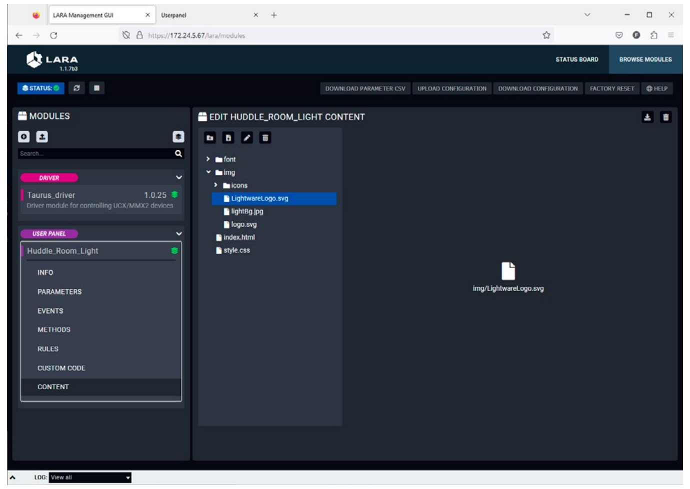
Huddle Room Light template with the default company logo

Uploading the New Logo

Step 1. Edit the user panel module by clicking on the icon. Go to the **Content** section and select the **img** folder.



Step 2. Upload the new company logo to the **img** folder. Click on the **Upload file** icon. Browse the new logo file and upload it. Recommended file extension is **.svg**. In our example it is the LightwareLogo.svg.

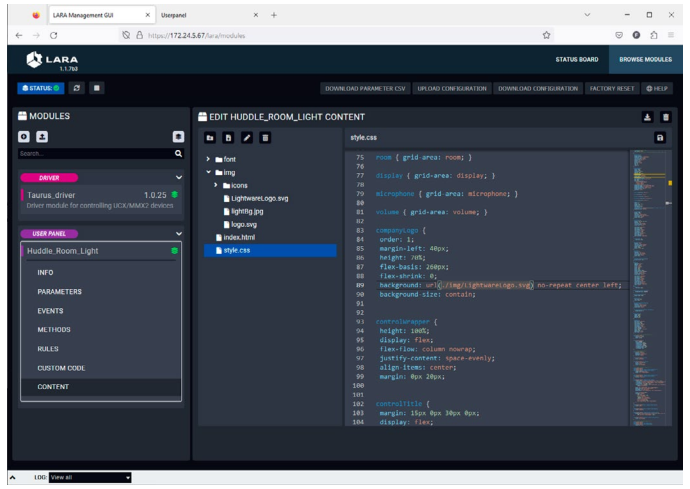


Editing of the CSS File

Step 3. Select the **style.css** and find the row in the code where the logo.svg is actually linked. In our example it is in the row 89. Edit the link of the logo file to the new one like in our example:

Original code: `background: url(./img/logo.svg) no-repeat center left;`

Edited code: `background: url(./img/LightwareLogo.svg) no-repeat center left;`

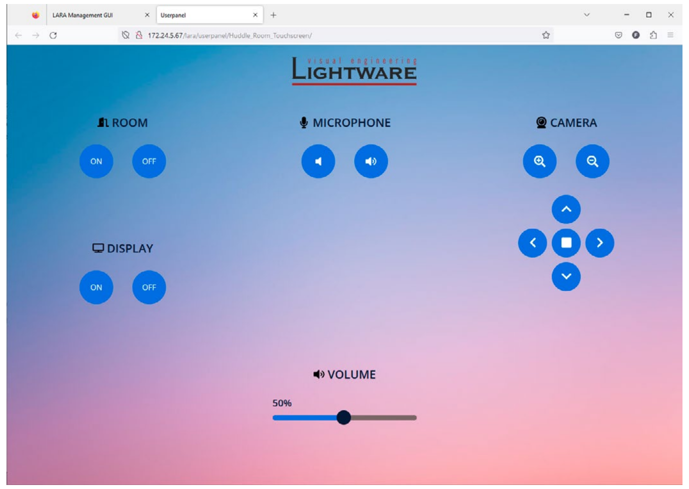


```
75  room { grid-area: room; }
76
77  display { grid-area: display; }
78
79  microphone { grid-area: microphone; }
80
81  volume { grid-area: volume; }
82
83  companyLogo {
84   order: 1;
85   margin-left: 40px;
86   height: 70%;
87   flex-basis: 260px;
88   flex-shrink: 0;
89   background: url(./img/LightwareLogo.svg) no-repeat center left;
90   background-size: contain;
91
92
93  controlWrapper {
94   height: 100%;
95   display: flex;
96   flex-flow: column nowrap;
97   justify-content: space-evenly;
98   align-items: center;
99   margin: 0px 20px;
100
101
102 controlTitle {
103  margin: 15px 0px 30px 0px;
104  display: flex;
```

Step 4. Save the style.css by clicking on the **Save file** icon.

Refreshing of the Touchscreen UI

Step 5. Go the touchscreen panel UI and refresh it in the web browser. The new company logo appears immediately.



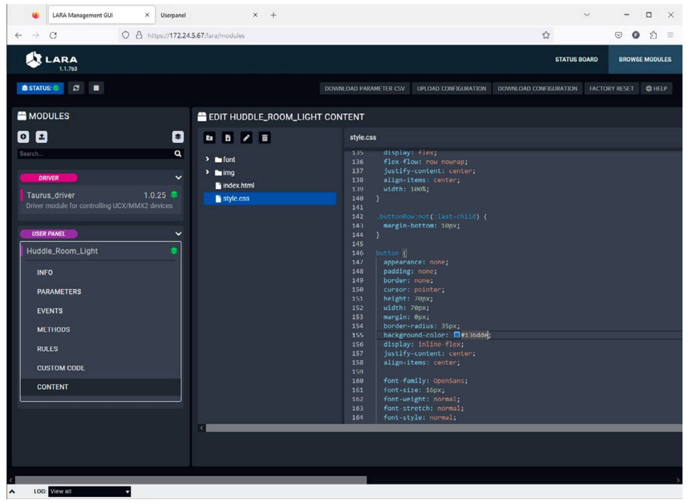
Touchscreen UI with the new company logo

Changing the Color of the Buttons

The next few steps show how to change the color of the buttons.

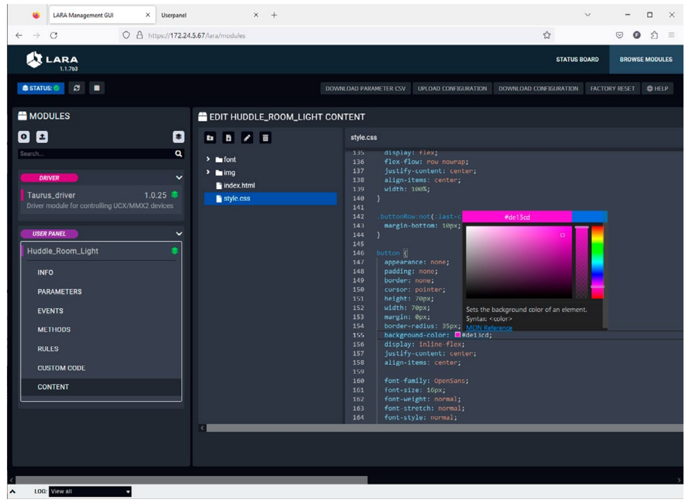
Editing of the CSS File

Step 1. Select the **style.css** and find the row in the code where the background color of the buttons are defined. In our example it is in the row 155.



```
135    display: flex;
136    flex-flow: row nowrap;
137    justify-content: center;
138    align-items: center;
139    width: 100%;
140  }
141
142  .buttonRow:not(:last-child) {
143    margin-bottom: 10px;
144  }
145
146  button {
147    appearance: none;
148    padding: none;
149    border: none;
150    cursor: pointer;
151    height: 70px;
152    width: 70px;
153    margin: 0px;
154    border-radius: 35px;
155    background-color: #136dde;
156    display: inline-flex;
157    justify-content: center;
158    align-items: center;
159
160    font-family: OpenSans;
161    font-size: 16px;
162    font-weight: normal;
163    font-stretch: normal;
164    font-style: normal;
```

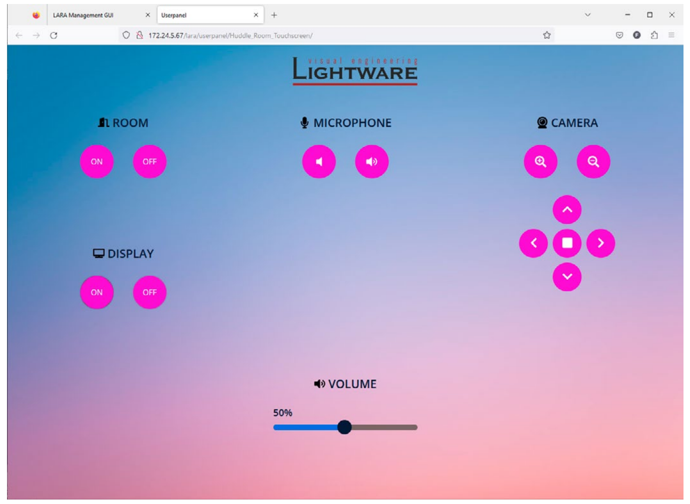
Step 2. Type the wished RGB code after the # character or select it in the color picker pop-up window.



```
155    background-color: #de13cd;
```

Step 3. Save the style.css by clicking on the **Save file** icon.

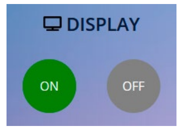
Step 4. Go the touchscreen panel UI and refresh it in the web browser. The new button color appears immediately.

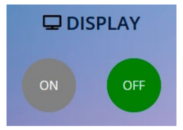


Adding Feedback Indicators to the Buttons

A feedback from the UI which makes visible that the interaction is actually happened is a valid user requirement. The following section shows how to do it in the touchscreen panel module.

Example Description



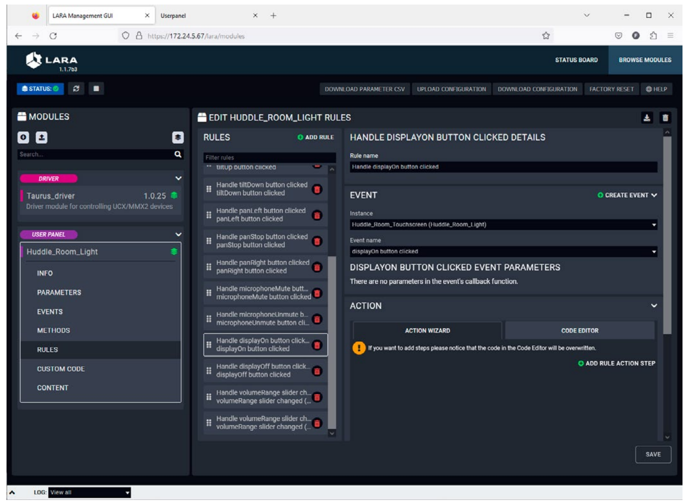


The Display On button shall change its color to green when it is clicked and parallel the Display Off button shall change its color to grey.

When Display Off button is clicked, its color shall change to green and paralel the Display On button shall change its color to grey.

Adding Rule Action Steps

Step 1. Edit the user panel module by clicking on the icon. Navigate to the **Rules** section and select the **Handle displayOn button clicked** rule.


LARA Management GUI
Userpanel
https://172.24.5.67/lara/modules
LARA 1.1.7b3
STATUS BOARD
BROWSE MODULES
STATUS:
DOWNLOAD PARAMETER CSV
UPLOAD CONFIGURATION
DOWNLOAD CONFIGURATION
FACTORY RESET
HELP
MODULES
Search...
DRIVER
Taurus_driver 1.0.25
Driver module for controlling UCX/MMX2 devices
USER PANEL
Huddle_Room_Light
INFO
PARAMETERS
EVENTS
METHODS
RULES
CUSTOM CODE
CONTENT
EDIT HUDDLE_ROOM_LIGHT RULES
RULES
ADD RULE
Filter rules
tiltUp button clicked
Handle tiltDown button clicked tiltDown button clicked
Handle panLeft button clicked panLeft button clicked
Handle panStop button clicked panStop button clicked
Handle panRight button clicked panRight button clicked
Handle microphoneMute butt... microphoneMute button clicked
Handle microphoneUnmute b... microphoneUnmute button cli...
Handle displayOn button click... displayOn button clicked
Handle displayOff button click... displayOff button clicked
Handle volumeRange slider ch... volumeRange slider changed (...
Handle volumeRange slider ch... volumeRange slider changed (...
HANDLE DISPLAYON BUTTON CLICKED DETAILS
EVENT
CREATE EVENT
DISPLAYON BUTTON CLICKED EVENT PARAMETERS
ACTION
ACTION WIZARD
CODE EDITOR
ADD RULE ACTION STEP
SAVE
LOG: View all

Rule name: Handle displayOn button clicked

Instance: Huddle_Room_Touchscreen (Huddle_Room_Light)

Event name: displayOn button clicked

There are no parameters in the event's callback function.

If you want to add steps please notice that the code in the Code Editor will be overwritten.

Step 2. Click on the **Add rule action step** button. Fill the fields with the following values. Click on the **Save** button when it is done.

Name	DisplayOn color green
Description	Color of the Display On button changes to green when it's clicked.
Select action step	Set HTML Element CSS Style
HTML Element ID	displayOn
CSS Property name	background-color
CSS Property value	green

Step 3. Click on the **Add rule action step** button again. Fill the fields with the following values. Click on the **Save** button when it is done.

Name	DisplayOff color grey
Description	Color of the Display Off button changes to grey when Display On is active..
Select action step	Set HTML Element CSS Style
HTML Element ID	displayOff
CSS Property name	background-color
CSS Property value	grey

Step 4. Navigate to the **Rules** section and select the **Handle displayOff button clicked** rule.

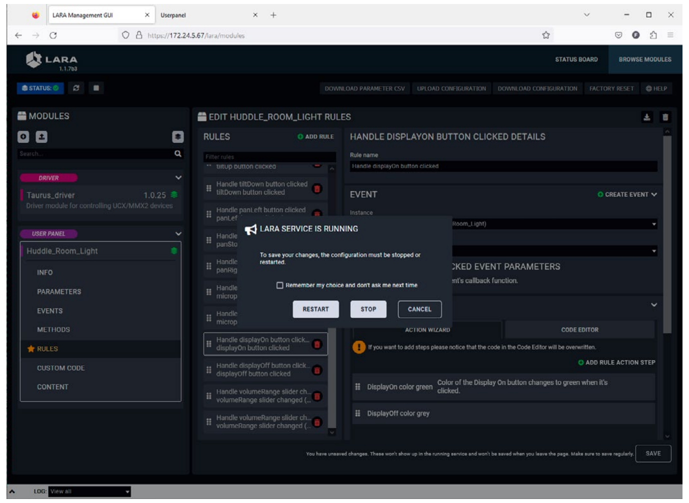
Step 5. Click on the **Add rule action step** button. Fill the fields with the following values. Click on the **Save** button when it is done.

Name	DisplayOff color green
Description	Color of the Display Off button changes to green when it's clicked.
Select action step	Set HTML Element CSS Style
HTML Element ID	displayOff
CSS Property name	background-color
CSS Property value	green

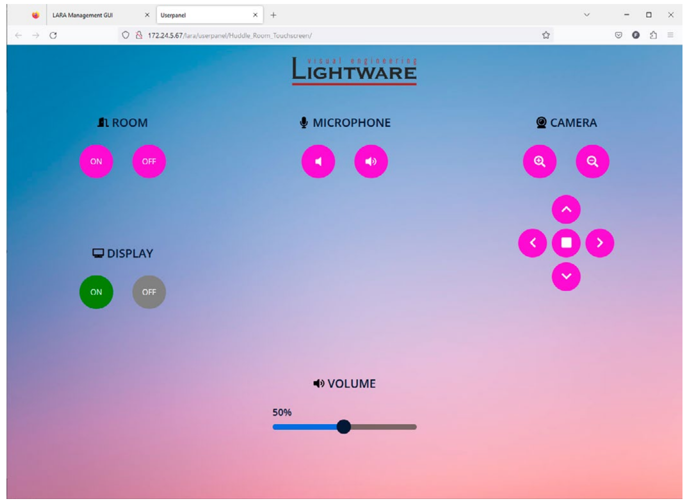
Step 6. Click on the **Add rule action step** button again. Fill the fields with the following values. Click on the **Save** button when it is done.

Name	DisplayOn color grey
Description	Color of the Display On button changes to grey when Display Off is active..
Select action step	Set HTML Element CSS Style
HTML Element ID	displayOn
CSS Property name	background-color
CSS Property value	grey

Step 7. Click on the **Save** button to apply the recent changes. Accept to **Restart** LARA service.


LARA SERVICE IS RUNNING
To save your changes, the configuration must be stopped or restarted.
Remember my choice and don't ask me next time
RESTART
STOP
CANCEL

Step 8. Go the touchscreen panel UI. After restarting of the LARA service the UI will be refreshed automatically and the new button styles appears.



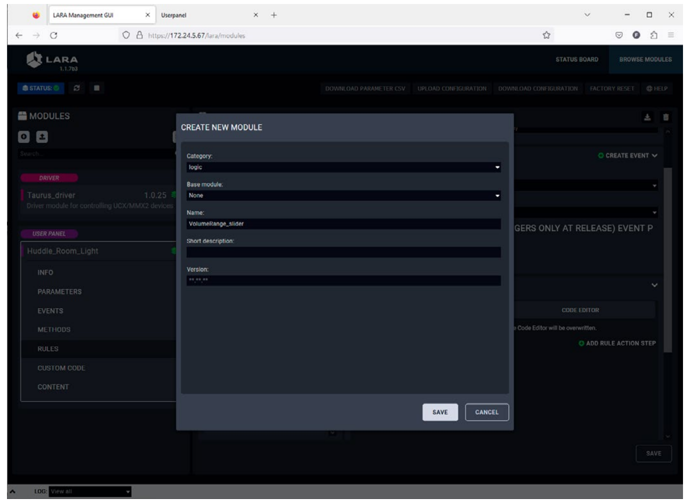
TIPS AND TRICKS: The example above is one possibility to customize the image of the touchscreen panel. Every element which has CSS property can be re-colored / resized / added new element (for example border around the buttons) etc. Only the imagination can set the limit

How to Use the Value of the VolumeRange Slider Changed Event

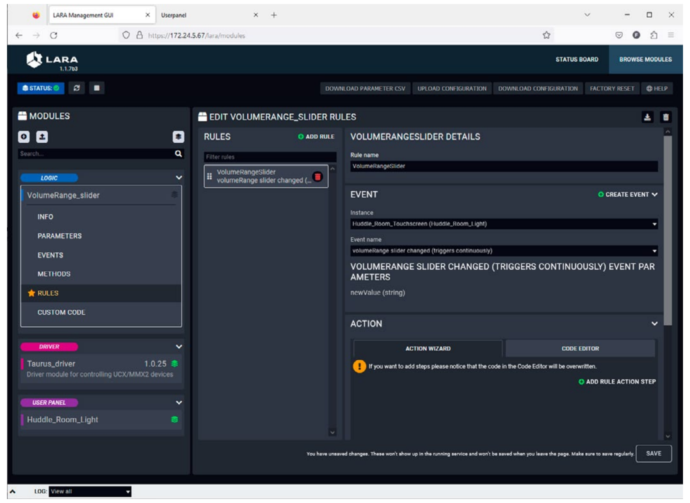
Disclaimer: due to a bug, the slider of the HTML templates do not work as it should. The following workaround helps to have a correct configuration – until the bug is corrected. The phenomenon is that the slider does not show the correct setting when the slider is moved. The root cause is in the rule editor's action wizard: the values written into the parameter input fields (for example a method's parameter input field) are handled as constant values, thus an event parameter cannot be passed to an invoked method via wizard. This section is about a workaround to solve that issue.

Creating of a Logic Module

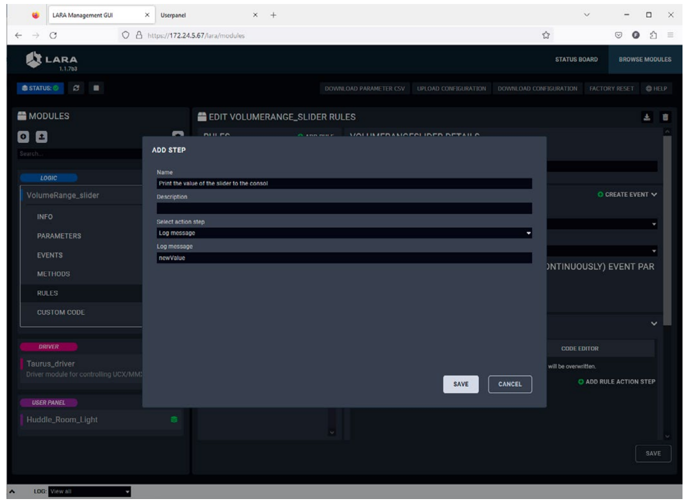
Step 1. Click on the ⊕ button under the Modules section to create a new logic module. Select the **logic** category and type a custom name. Finally click on the **Save** button.



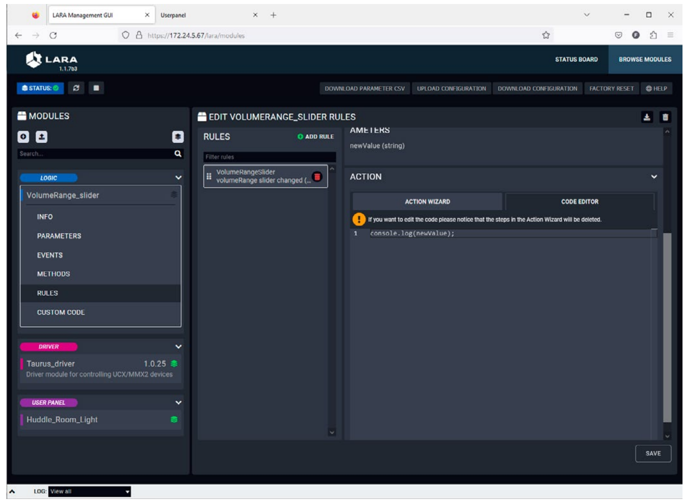
Step 2. Create a new rule in the logic module by clicking on the ⊕ **Add rule** button. Select the created userpanel instance and the **volumeRange slider changed (triggers continuously)** event in the Event section. Finally click on the **Save** button.


EDIT VOLUMERANGE_SLIDER RULES
RULES
VOLUMERANGESLIDER DETAILS
Rule name: volumeRangeSlider
EVENT
Instance: Huddle_Room_Touchscreen (Huddle_Room_Light)
Event name: volumeRange slider changed (triggers continuously)
VOLUMERANGE SLIDER CHANGED (TRIGGERS CONTINUOUSLY) EVENT PARAMETERS
newValue (string)
ACTION
ACTION WIZARD
CODE EDITOR
ADD RULE ACTION STEP
SAVE

Step 3. Add an action step to the rule by clicking on the **Add rule action step** button. Type a custom name and choose the **Log message** for example. In the "Log message" input, type in the parameter name of the previously selected event, which is **newValue**, and click on **Save** button.

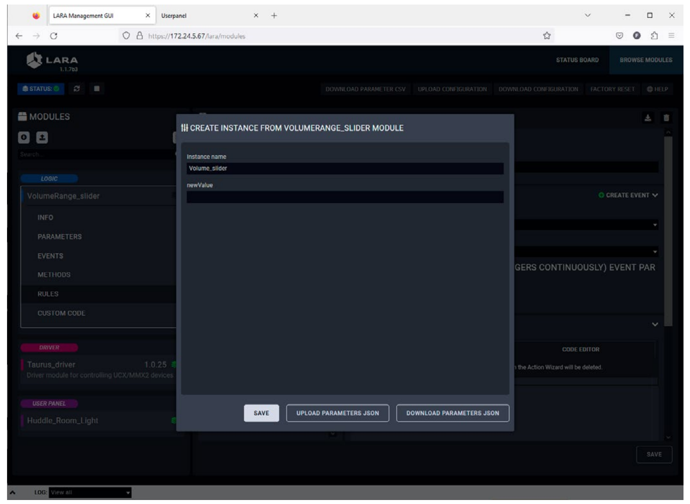

ADD STEP
Name: Print the value of the slider to the consol
Description:
Select action step: Log message
Log message: newValue

Step 4. Switch to **Code Editor** in the actions section. **Remove the apostrophes** from the console.log function call. Finally click on the **Save** button.

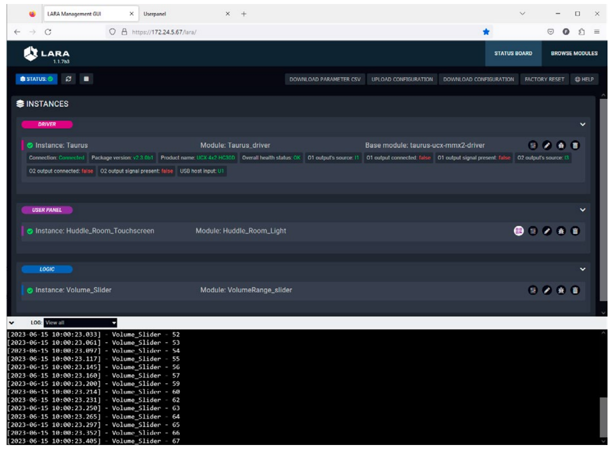

If you want to edit the code please notice that the steps in the Action Wizard will be deleted.

```
console.log(newValue);
```

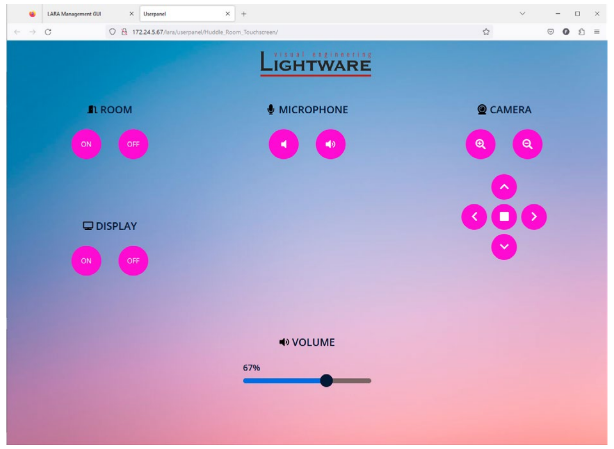
Step 5. Create an instance by clicking on the **Create new instance for selected module** icon. Type a custom name and click on **Save** button.


CREATE INSTANCE FROM VOLUMERANGE_SLIDER MODULE
Instance name: Volume_slider
newValue:

Step 6. It's working!


Instance: Volume_Slider Module: VolumeRange_slider

```
[2023-06-15 10:00:23.033] - Volume_Slider - 52
[2023-06-15 10:00:23.061] - Volume_Slider - 53
[2023-06-15 10:00:23.097] - Volume_Slider - 54
[2023-06-15 10:00:23.117] - Volume_Slider - 55
[2023-06-15 10:00:23.145] - Volume_Slider - 56
[2023-06-15 10:00:23.160] - Volume_Slider - 57
[2023-06-15 10:00:23.200] - Volume_Slider - 59
[2023-06-15 10:00:23.214] - Volume_Slider - 60
[2023-06-15 10:00:23.231] - Volume_Slider - 62
[2023-06-15 10:00:23.250] - Volume_Slider - 63
[2023-06-15 10:00:23.265] - Volume_Slider - 64
[2023-06-15 10:00:23.297] - Volume_Slider - 65
[2023-06-15 10:00:23.352] - Volume_Slider - 66
[2023-06-15 10:00:23.405] - Volume_Slider - 67
```



5 Additional Features

The factory modules and the pre-defined settings cover numerous cases in a room environment. But sometimes you have to go deeper and arrange unique settings – this chapter is about such cases with useful practices.

5.1. General Features

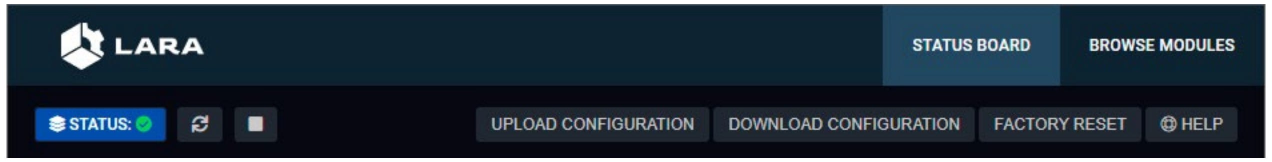
When the Firmware is Updated

If the firmware is updated with FW package v2.1 or newer version, the settings of the UCX/MMX2 device can be preserved, as well as the LARA configuration.

Configuration Download and Upload

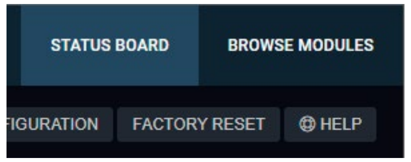
The full LARA configuration (including modules, instances and parameters with passwords as well) can be downloaded as a ZIP file. The file can be uploaded to the same device or another device of the same type. Use the **Upload Configuration/Download Configuration** buttons in the upper menu.

TIPS AND TRICKS: The uploading can be done by dragging and dropping the ZIP file.



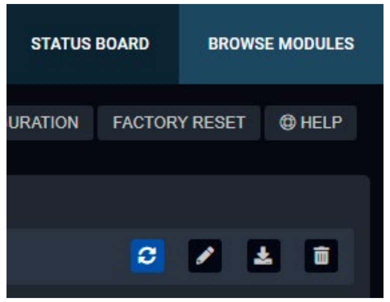
Factory Reset

All modules and instances can be deleted (after confirmation) by pressing the button.



Module Update

LARA contains factory modules that cannot be removed. If you upload a configuration/module, it may contain a newer version of a module than the existing one of your configuration. In that case, a blue icon is displayed in the **Browse modules** page:

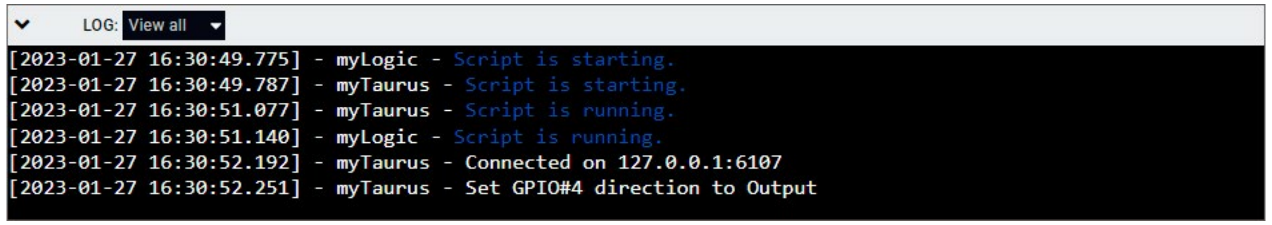


You can update the module by pressing the button and the new version will be available in your device.

INFO: Parameters, events, methods and rules defined by the user in a factory module are preserved when updating to a new version.

Log Window

The bottom part of the window can be toggled to show the log screen. The messages from each running instance can be viewed here in real time here, so you can follow the state of your system. You can view **all instances** at same time or filter the messages to only **one instance**. The section is resizable or it can be hidden by the down arrow:



```
[2023-01-27 16:30:49.775] - myLogic - Script is starting.
[2023-01-27 16:30:49.787] - myTaurus - Script is starting.
[2023-01-27 16:30:51.077] - myTaurus - Script is running.
[2023-01-27 16:30:51.140] - myLogic - Script is running.
[2023-01-27 16:30:52.192] - myTaurus - Connected on 127.0.0.1:6107
[2023-01-27 16:30:52.251] - myTaurus - Set GPIO#4 direction to Output
```

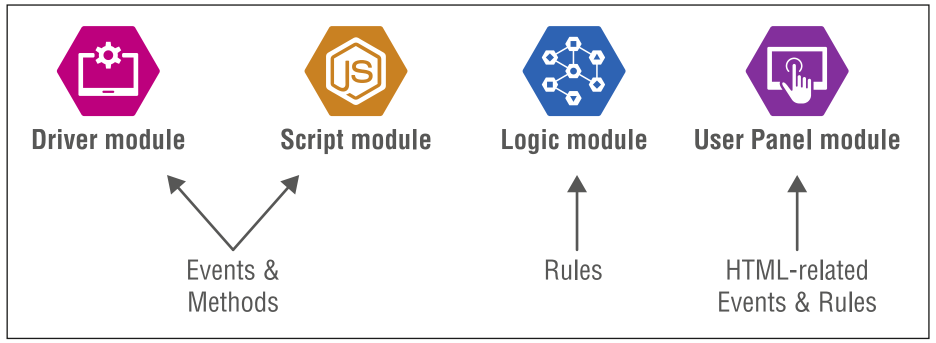
TIPS AND TRICKS: The content of the log window can be copied to the clipboard.

5.2. Best Practices

How to Build Your Configuration

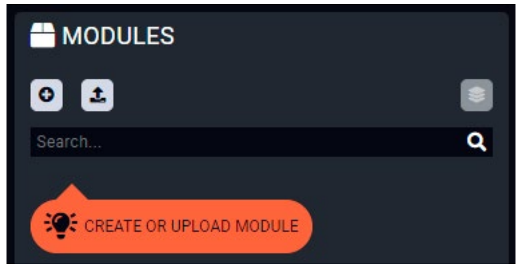
When building a LARA configuration, it is worth to pay attention for:

- Setting the **Events** and **Methods** in the **Driver** and **Script** modules.
- Setting the **Room automation related rules** in the **Logic** module.
- When using a UserPanel module, set the **HTML-related events** and **rules** in the **UserPanel** module.



These principles help create **re-usable modules**.

Adding a New Module



As mentioned previously, the driver module is the interface to a device. It can be used to connect to a device. A module can be added on the **Browse Modules** page with:

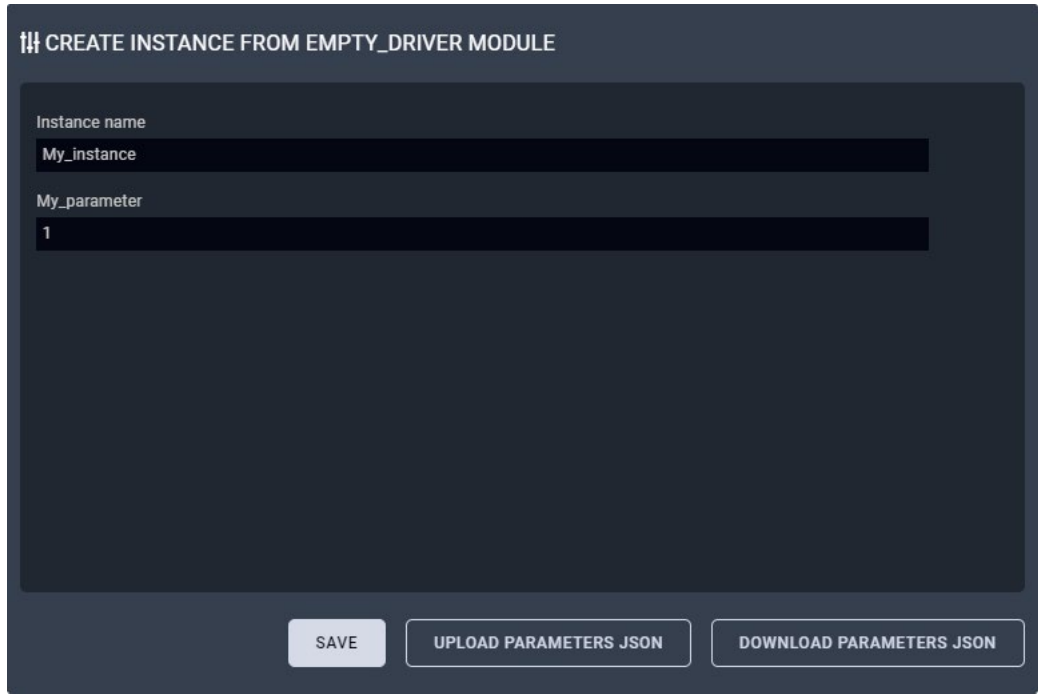
- Pressing the button and selecting the available list, or
- Pressing the button and add a previoulsy saved module (as a ZIP file).

ATTENTION! The default methods, parameters and codes of the factory modules cannot be deleted.

TIPS AND TRICKS: The modul (as a ZIP file) can be added with mouse drag&drop into this window.

Creating Instances

Step 1. Navigate to the **Browse modules** page.
Step 2. Press the icon in the line of the desired **module**.
Step 3. Type a name for the instance and set the value of the defined parameters (optional).
Step 4. Press the **Save** button.


CREATE INSTANCE FROM EMPTY_DRIVER MODULE

Instance name
My_instance

My_parameter
1

The value of the parameters can be set later as well:

Step 1. Navigate to the **Status Board**.
Step 2. Press the button in the end of the desired instance line to open the **Parameter editor** window.

TIPS AND TRICKS: The parameters and values of the instance (seen on the screen) can be saved as a JSON file by the **download** button. Previously downloaded parameters and values can be uploaded at this step by the **upload** button.

5.3. JavaScript Code Examples

The modules may contain custom codes based on nodeJS programming language. Please find below the following JavaScript example codes that could help you writing your custom codes.

Writing something to the log window (comments can be written after double slash)

```
console.log('Hello world!'); //this is a comment
```

Writing an error message to the output console (seen at the bottom part of the window, written in red)

```
console.error('Oh, no! Something terrible has happened!');
```

Accessing instance parameters

```
console.log('Device IP address is ', params.ip_address);
```

Waiting for 2000 milliseconds

```
await new Promise(r => setTimeout(r, '2000'));
```

Showing/updating information on status board with a given color

```
this.instance.updateStatus('Today weather', 'Sunny', {color: 'yellow'});
```

Emitting an event from an instance

```
this.instance.dispatchEvent('resulutionChanged', '1920x1080p60');
```

Running a custom code 10 seconds later

```
console.log('Enable the relay');
setTimeout(()=>{
  console.log('Disable the relay');
},10000);
```

Running a code every 5 seconds

```
setInterval(()=>{
  console.log('This is printed every 5 seconds');
},5000);
```

Running an action when our instance has fired an event

```
this.instance.on('messageReceived', (message) => {
  console.log('A message has just been received:', message)
});
this.instance.dispatchEvent('messageReceived', 'Meow');
```

Running an action when another instance has fired an event

```
this.getInstanceById('display').on('volumeChanged', (message) => {
  console.log('The display volume has just been changed:', message)
});
```

Invoking a method from our own instance

```
this.myFancyMethod(param1, param2);
```

Invoking a method from another instance

```
instanceApi.getInstanceById('display').send('powerOn');
```

Importing nodejs built-in modules (see this documentation about the available modules)

```
//import net module
var net = require('net');
```

Writing a file

```
this.fs.writeFileSync(this.instance.getLocalStoragePath()+'/data.txt', 'Hello world');
```

Reading a file

```
var data = this.fs.readFileSync(this.instance.getLocalStoragePath()+'/data.txt');
```

More information and examples about file operations: https://nodejs.dev/learn/the-nodejs-fs-module

Sending and receiving response to/from a client over TCP/IP

```
var net = require('net');
var client = new net.Socket();
client.connect(1337, '192.168.1.1', function() {
  console.log('Connected');
  client.write('Hello, server! Love, Client.');
});

client.on('data', function(data) {
  console.log('Received: ' + data);
  client.destroy(); // kill client after server's response
});

client.on('close', function() {   console.log('Connection closed');
});
```

Sending an HTTP GET message

```
const http = require('http');

http.get('http://192.168.0.1', (res) => {
  const { statusCode } = res;
  const contentType = res.headers['content-type'];
  if (statusCode !== 200) {
    console.log('Request Failed. Status Code: ', statusCode);
    res.resume();
    return;
  }
  res.setEncoding('utf8');
  let rawData = '';
  res.on('data', (chunk) => { rawData += chunk; });
  res.on('end', () => {
    console.log('Received: ', rawData);
  });
}).on('error', (e) => {
  console.error('Got error:',e);
});
```

5.4. External Links

The following documentations help becoming a more professional user of LARA:

The webpage of LARA:

https://lightware.com/lara

The nodeJS programming language – for creating custom codes:

https://nodejs.org/en/docs/

The markdown formatting – for editing a cool module description:

https://daringfireball.net/projects/markdown/

Reserved Words – good to know when defining parameters:

https://www.w3schools.com/js/js_reserved.asp

Document Revision History

Rev.	Release date	Changes	Editor
v1.0	22-03-2023	Initial version	Laszlo Zsedenyi
v1.1	03-10-2023	User module chapter (Touchscreen) added; minor corrections and updates.	Tamas Forgacs Laszlo Zsedenyi

Contact Us

sales@lightware.com

+36 1 255 3800

support@lightware.com

+36 1 255 3810

Lightware Visual Engineering PLC.

Peterdy 15, Budapest H-1071, Hungary

www.lightware.com