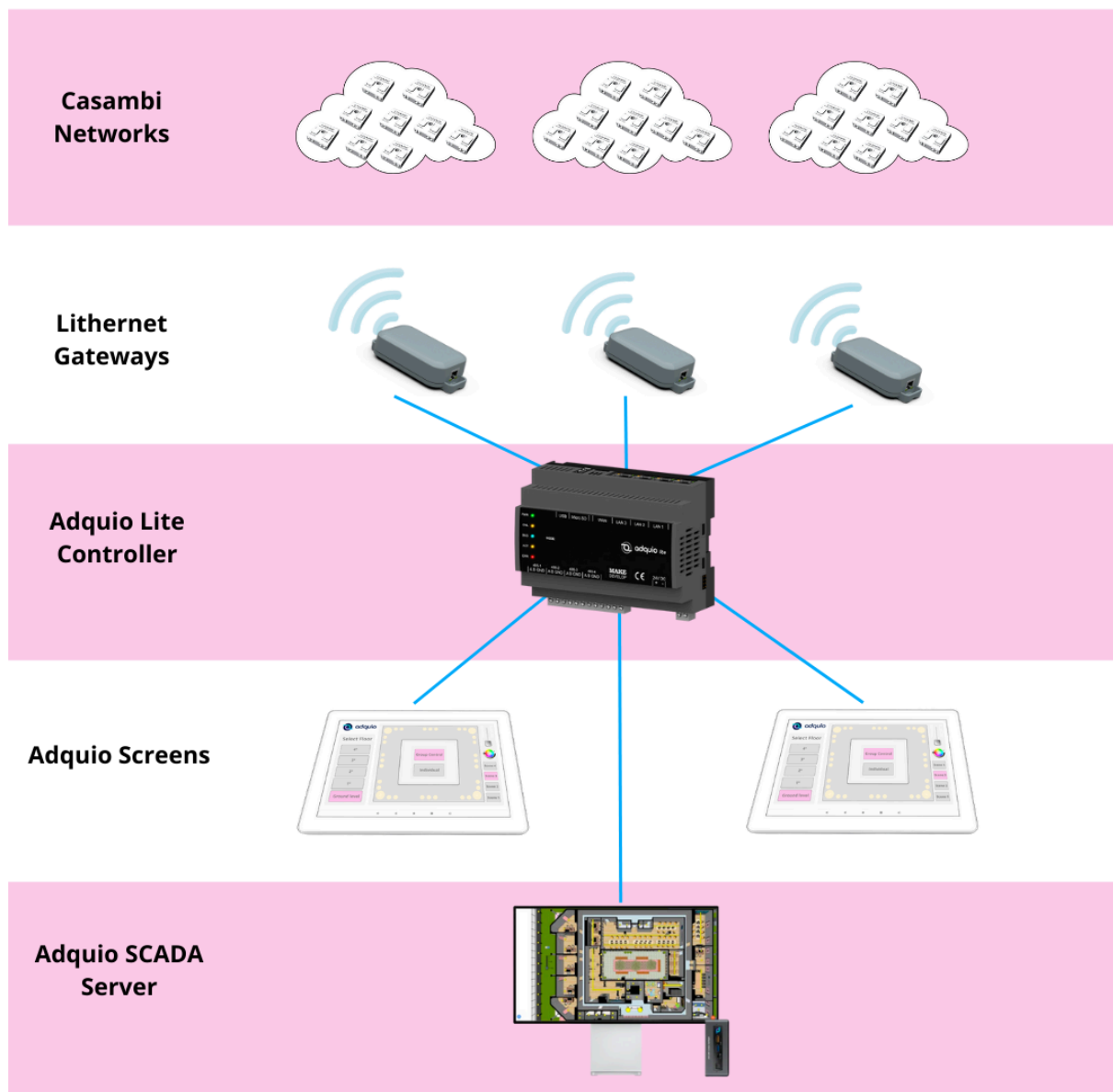




From your Casambi networks to total control with SCADA or custom screens.



From your Casambi networks to total control with SCADA or custom screens with Adquio.

Guillermo Cuervo López: CTO of Adquio.

November 1, 2023

Updated January 26, 2024

Index

Index.....	3
Objective of this white paper:.....	4
General scheme.....	5
Adopting Lithernet Gateway in your Casambi network.....	6
Configuring Lithernet Gateway.....	11
Getting list of variables with Yabe.....	20
Converting the CSV file.....	25
Importing file in Adquio.....	30
Connection with Adquio via Wi-Fi:.....	33
Connection with Adquio by cable:.....	35
How to change language.....	37
Creating a module.....	38
Creating a device.....	41
Importing the device.....	42
Processing information in Adquio.....	45
Synchronize scenes between different networks.....	47
Creating a module.....	48
Creating a device and its variables.....	50
Programming the connection with scenes.....	54
Managing errors and statuses in Adquio.....	60
Launching Alarms in Adquio.....	62
Exporting values in Adquio.....	65
Working with Adquio screens.....	71
Set up the connection.....	73
Creating an Interface.....	75
How to create objects.....	75
How to add a background image.....	80
How to export your HMI.....	82
How to import your HMI.....	84
HMI auto start.....	87
Converting variables for SCADA.....	89
Importing and using variables in Adquio SCADA Server/Cloud.....	92
Importation process.....	92
Using variables in SCADA.....	97
Final conclusion.....	106

Objective of this white paper:

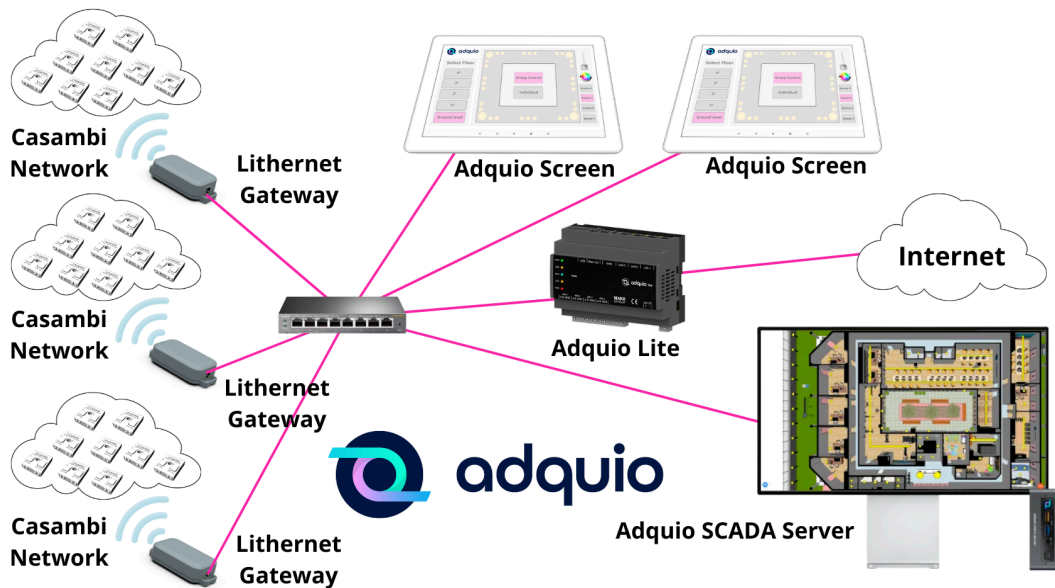
Welcome to an exciting system's integration journey! This document has a clear purpose: to help you master the integration of Casambi networks with our powerful Adquio programmable controllers. Wondering how to easily make that data flow to your local or remote control systems? As [Adquio control screens](#) or the robust systems [Adquio SCADA Server](#) or [Adquio SCADA Cloud](#); you are in the right place!

Imagine the satisfaction of completing this reading with the knowledge to integrate any size installation with Casambi networks into your local control system, displays or SCADA, whether in the cloud, or in the comfort of your local environment. But before we dive into this exciting journey, let's clarify an important point: we will focus on local integration, the strongest and safest option that currently offers exceptional results.

Have you ever wondered how complicated it can be to integrate different systems? In this case, we will work with four elements: three, from our Adquio brand, and one external from the manufacturer, [Light Manufactory](#). Don't worry, though, our Adquio ecosystem is designed to simplify every step of this integration process. You will learn to use all the tools and programs that facilitate the flow of data between these various platforms.

When you complete this document, you will be a master at integrating Casambi networks with Adquio control systems, and you will be ready to face any challenge that arises. So get ready to soak up the knowledge and take a qualitative leap in your integration projects. Let's start!

General scheme



To follow this document, we will use the previously established scheme! Before we dive into the details, we'll make a few crucial observations to get things crystal clear.

- Take a look at the scheme, where three Casambi networks have been integrated. And best of all, this can be applied to any number of networks you need!
- In our example, we have opted for the powerful Adquio Lite, capable of controlling six Casambi networks with dexterity. If your requirement is even greater, don't worry, we have at your disposal [a suitable controller to fit any size](#).
- And that's not all, two control screens also shine in the diagram. Our controllers are capable of handling up to 30 screens each!
- The choice to use an Adquio Lite is no coincidence, its presence not only guarantees control, but also provides solid protection to your control installation thanks to its internal firewall.

In the pages that follow, we will discover together and in detail all the steps to travel from Casambi networks to control through personalized screens and supervision through an SCADA, both locally and in the cloud!

Let's start this exciting adventure!

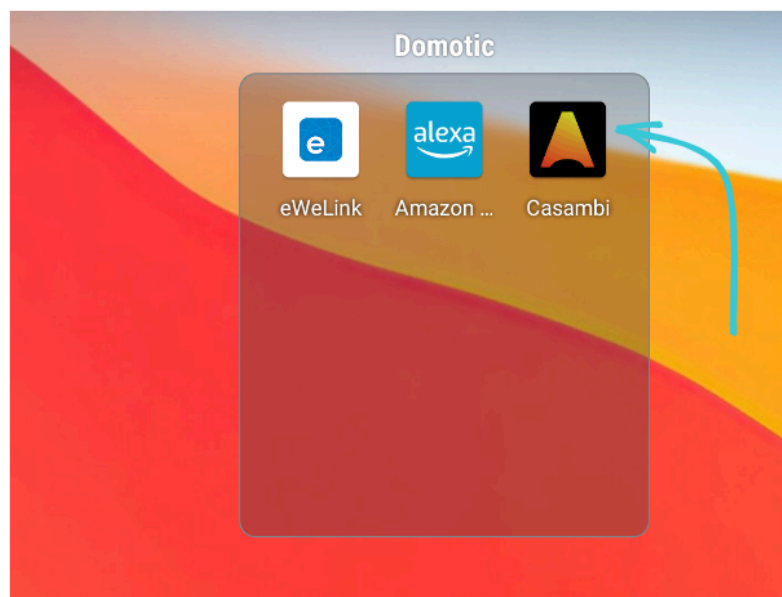
Adopting Lithernet Gateway in your Casambi network



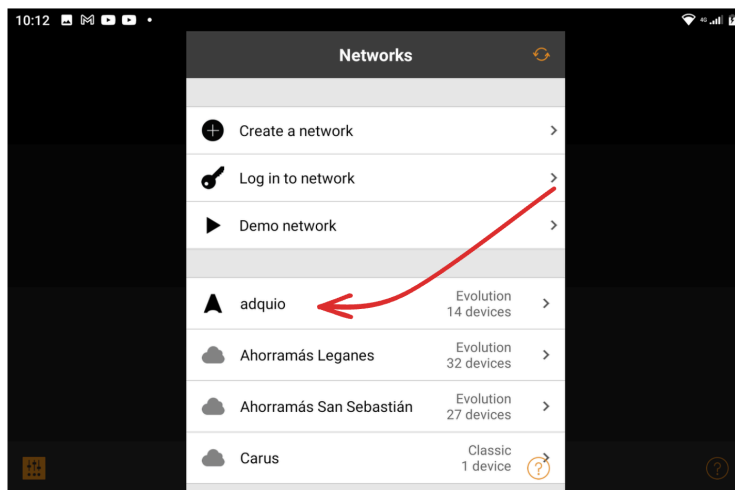
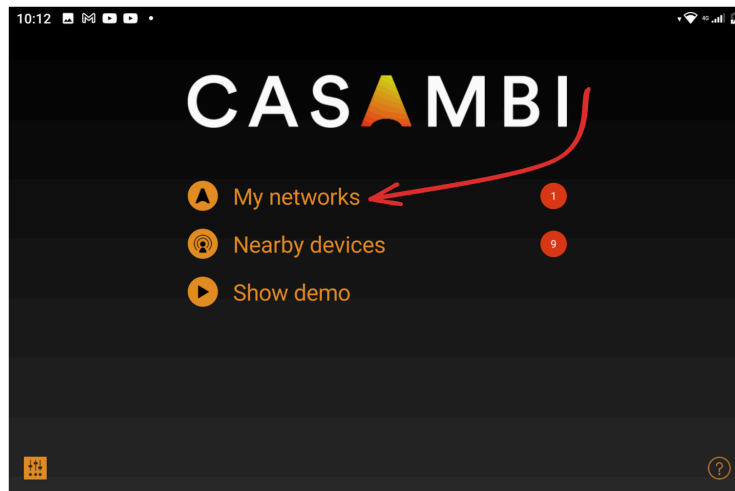
We begin this exciting journey by learning how to include the Lithernet Gateway in your Casambi network. This is the first step to start controlling your Casambi networks at another level.

The first thing we need to know, is that we will need the standard Casambi application. We assume that you already have it installed, otherwise, you can do it from the Android or Apple application stores, in any case they are completely free.

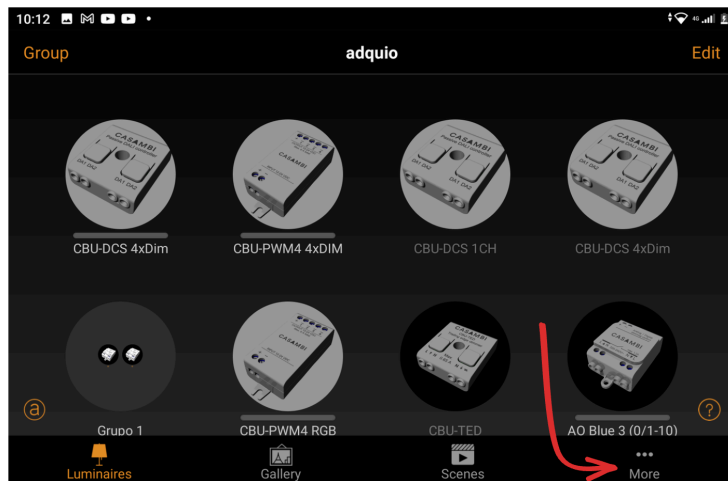
If you already have it installed, you just have to click on its icon to enter.



Next, you will see this screen, where you must enter and select your Casambi network, in which you are going to integrate your Lihernet. In our case, the network is called 'Adquio', original, right 😊?



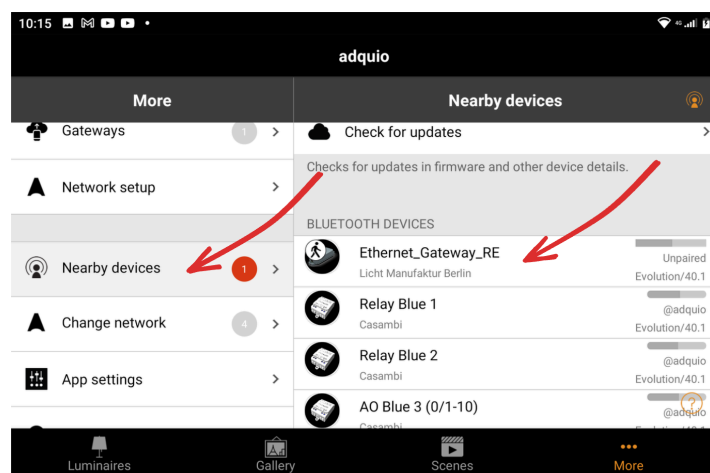
Very good, we have already entered the main Casambi screen. Now we will go to the bottom right 'More' button.



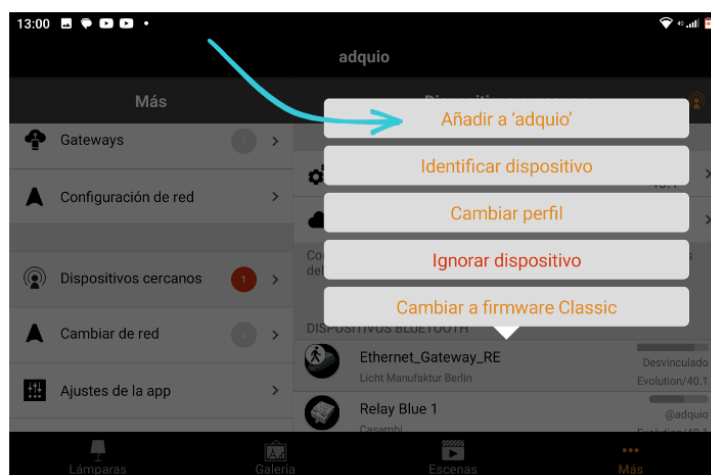
Before continuing, you must power your Lihernet Gateway. You can see how to do it at point '**Configuring Lihernet Gateway**' which you will find later.

Upon entering, you will see a column on the left with the heading name '**Further**', swipe up until you see an option called '**Nearby devices**', click on it.

Now, on the right, you will see a list of devices, among them, you will find one called '**Ethernet_Gateway_RE**', that is your Lihernet, click on it.

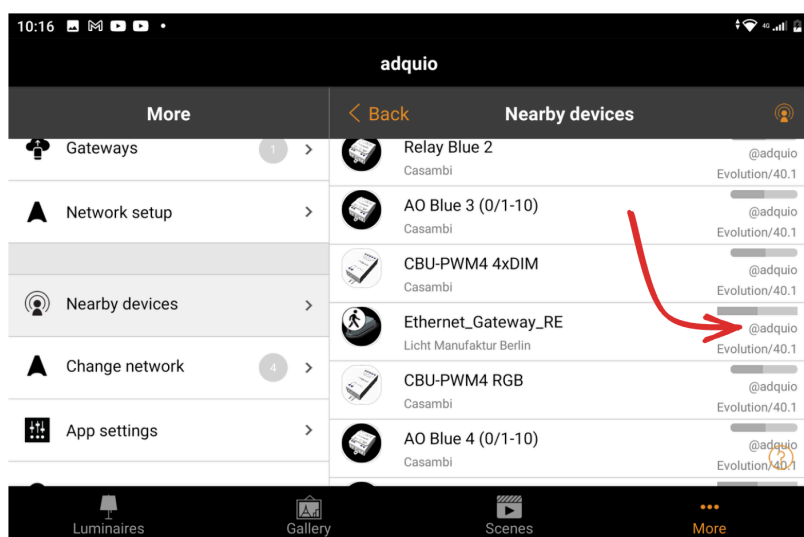


Doing so will open a menu in which you will see an option '**Add to**' and your network, click on it.

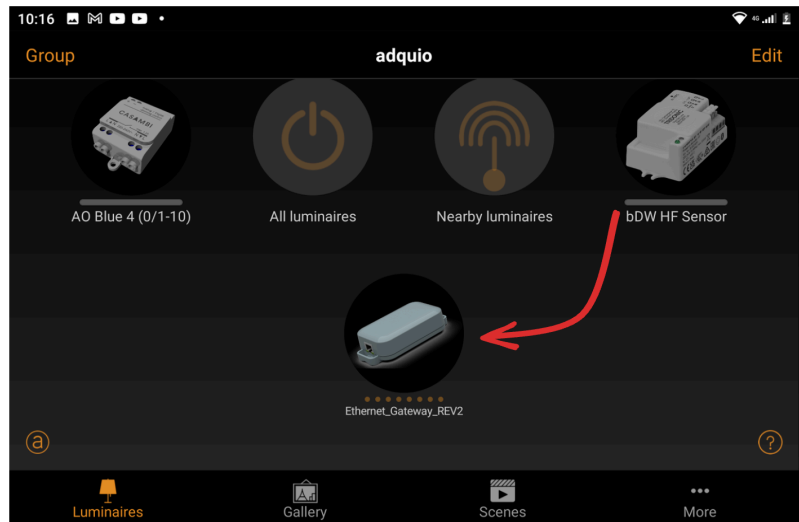


Remember, During this adoption process it is very important that your devices, tablet, or phone and your Lighthouse Gateway are very close, this will facilitate good communication and faster adoption.

Once the process is finished, you will see the device as another member of your Casambi network.

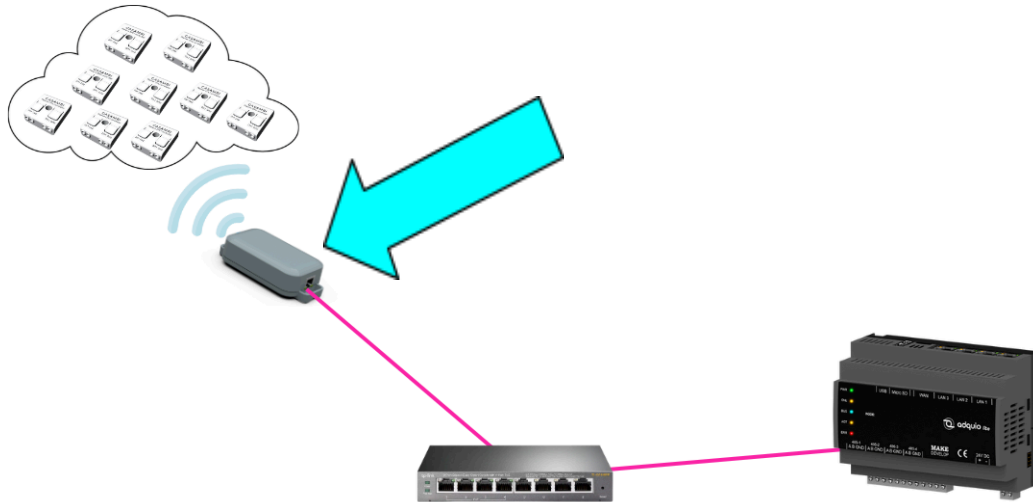


And you can see it on the main screen of your Casambi devices as one more.



We are done, we are ready to move on to the next step, it has been easy, right?
Congratulations!

Configuring Lithernet Gateway



Before starting, it is important to note that for correct operation you must only connect your computer to the Lithernet device, there should be no other device on the network interrogating it.

Remember that the Lithernet gateway is a POE device, therefore, you must power it through the network cable. You can do this in two ways.

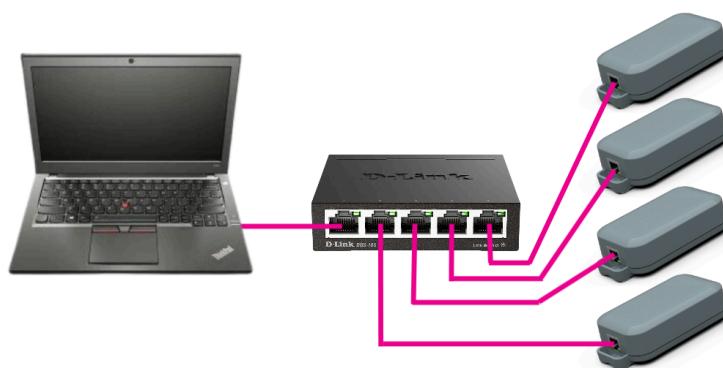
With a POE power supply. It is the easiest way if you are going to use a single Lithernet device. We leave you a small diagram of how to connect it.



With a POE switch. This option has the advantage that you can now connect several Casambi networks, since by having several ports you can connect several Lithernet interfaces.



This is the correct scheme for configuring each Lithernet using a POE switch.



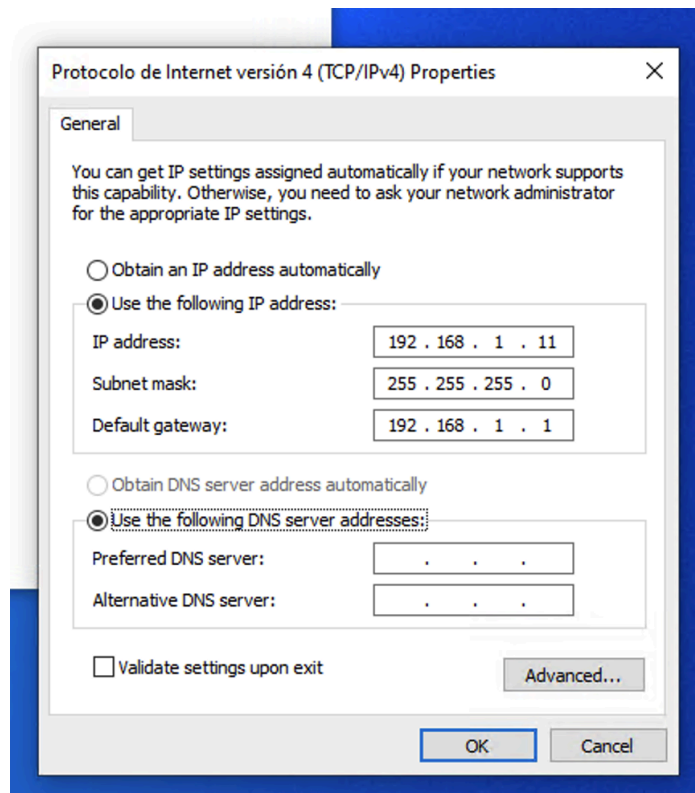
Be careful, The diagram shown above is only for final operation, not for commissioning. Each Lithernet device comes from the factory with a default IP, which is the same in all of them.



Therefore, if you connect them as you see in the image, you will have an IP address conflict, and you will not be able to access the devices.

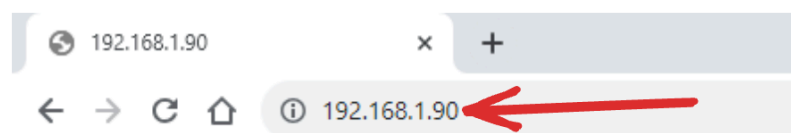
This scheme is valid for the moment in which you have associated with each Lithernet its distinct and definitive IP address.

In order to connect to the Lithernet device, you must give your computer a fixed IP address in the same range. Therefore, if the default Lithernet IP address is 192.168.1.90, your computer must have one that begins with 192.168.1, and the last digits a number between 1 and 254, but it must not be 90. For example, 11, consequently, your IP address will be 192.168.1.11.



We are not going to explain how to set an IP address on your computer, because [you can easily find this on the internet](#).

Ok, once we have everything prepared, we are going to connect to our Lithernet. To do this, we open a browser and enter the address of our Lithernet:



We confirm, and we will see, that the Lithernet screen appears in the browser, to select a username and password. This will only happen the first time, the following times it will only ask us for the data that we are going to enter now.

Casambi Ethernet Gateway - 3.30_beta - REV1

Set Login Credentials

Welcome

Welcome to the Lithernet Casambi Gateway.
Please take a note of the login credentials you set here and the mac adress you see below.
Store these information in an safe place.
We cant recover your login details. If you forget the password you can do an full reset with the mac address.
For more details, take a look in the manual.

MAC Address

Mac EC:62:60:1A:F5:EB

User Management

Username

Password

Continue

© Licht Manufaktur Berlin GmbH 2023

Once the username and password have been selected, we will click on 'Continue', at the bottom, and this will take us to the main page of the Lithernet interface.

Casambi Ethernet Gateway - 3.30_beta - REV1

Generell Settings

To Casambi

From Casambi

Console

General Setting

Network

System

Memory

At the top we can see a menu, within 'General Setting', we will click on 'System'. This takes us to the next page, where we will click on 'Wizard'.

Casambi Ethernet Gateway - 3.26 - REV1

Generell Settings
To Casambi
From Casambi
Console

General Setting

Network
System
Memory

Control System

Type	BacNet/IP
Local Device ID	400001
UDP-Port	47808
Use BBMD/FDT	inactive
BBMD Server IP	192.168.1.255
Use Broadcast	false
Devices	1 / 30
Use Ungrouped	false
Groups	1 / 10
Scenes	1 / 10
State	BacNet/IP Listening on IP: 192.168.1.90 Port: 47808

Wizard

This takes us to a page where we can select the integration protocol we want. We must choose 'BacnetIP' in the drop-down menu.

Casambi Ethernet Gateway - 3.30_beta - REV1

Generell Settings
To Casambi
From Casambi
Console

General Settings

Network
System
Memory

Control System Wizard

Type	<div style="border: 1px solid #ccc; padding: 5px;"> <div style="background-color: #cccccc; padding: 2px 5px;">BacnetIP ▼</div> <div style="padding: 5px;"> none Netcomposer Artnet (Input Only) UDP Free Messages HelvarNet (TCP) TCP Free Messages UDP Casambi Command UDP Casambi Bridge BacnetIP MQTT TCP Casambi Command WebApi (alpha) </div> </div>
------	--

Casambi Ethernet Gateway - 3.26 - REV1			
Generell Settings	To Casambi	From Casambi	Console
General Settings			
Network	System	Memory	
Control System Wizard			
Type	BacnetIP		
		next step	

We click on 'Next Step', and this takes us to the following configuration page:

Casambi Ethernet Gateway - 3.30_beta - REV1	
Generell Settings	To Casambi
General Settings	
Network	System
Control System Wizard	
Type	BacnetIP
Local Device ID	400001
UDP-Port	47808
Use BBMD Server / FDT	false
BBMD Server IP	192.168.1.255
Use Broadcast	false
Start Device	1
Device Count	30
Use Ungrouped	false
Start Group	1
Group Count	10
Start Scene	1
Scene Count	10
step back	
reboot..	

© Licht Manufaktur Berlin GmbH 2023

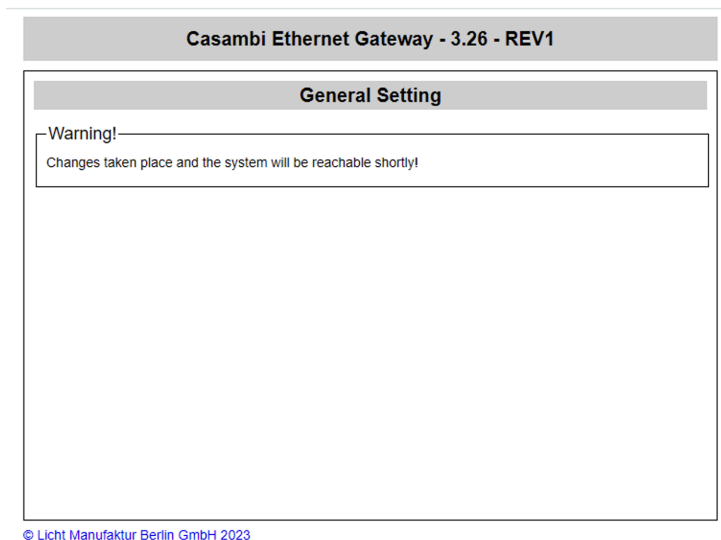
On this page, we must comment on several points that we indicate with red arrows in the image:

- **Local Device ID:** This is the unique identifier of this BACnet device on the network, if you have several, all the addresses in this position must be different. If it is the first one you configure, you can leave it like this, but if you configure several, you must enter consecutive values here, that is, the second will be 400002, and so on.
- **Use Broadcast:** With this, we indicate whether we are going to use broadcast in our network or not. Broadcast is a technology that allows all devices to receive and execute a command by sending it to the network. In some cases, it is used for speed to turn on or off all devices on the network. You can execute any action on all devices, not just turning on and off, but dimming, changing RGB, etc. If you activate it, you will see that Lihernet includes a new control block with all the options for this type of control. By default, it will be disabled.
- **Device Count:** It is the number of devices you want to export. It does not have to match the actual number of devices on your network. You should only enter the actual number if you need the data for each and every device on your network individually. This is not typical, but the normal thing is to control the lighting through groups or scenes. It is important to adjust to the real needs of your network, so if you do not need individual control you can put 0 here.
- **Group Count:** If you have decided to control your lighting by groups, then here you must indicate their number, but if you are going to do it by scenes you can leave this value at 0.
- **Scene Count:** You must indicate the number of scenes that you have configured on your Casambi network. This is only necessary if you are going to control your lighting by scenes, if you don't use scenes, you can leave it at 0.

In general, it is crucial to adjust to the requirements as much as possible, since the ideal is to have as few control variables as possible. In this way, the data exchange between the Adquio controller and the Casambi network will be more efficient and faster.

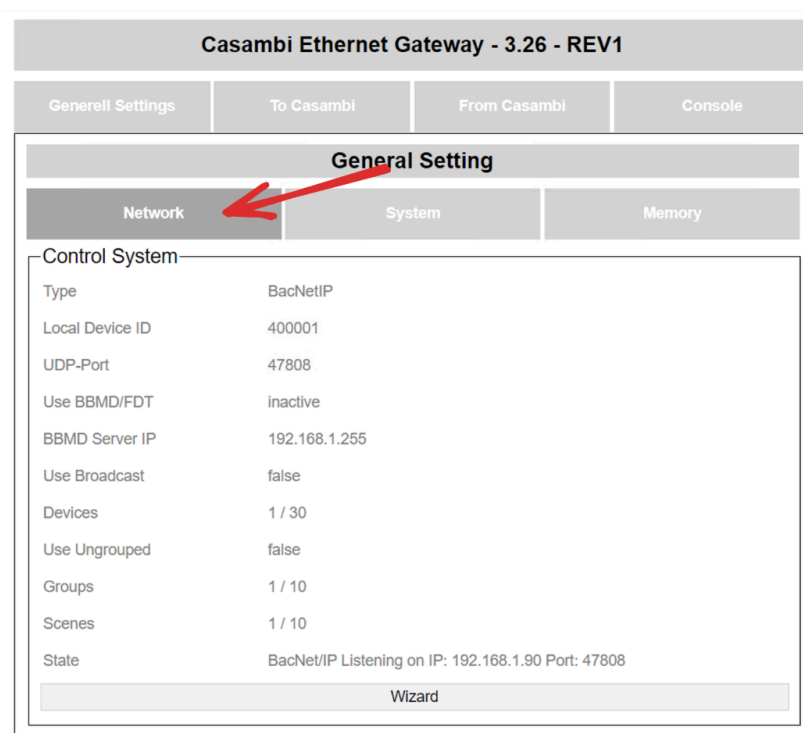
When you have finished and reviewed all the values, click on the 'Reboot' button below.

While the system reboots, you will see this page.



When your Lithernet device restarts, it will already be exporting values in BACnet as we have configured.

Once the reboot is complete, the system will take you to this new page:



The last step you will need is to configure the network to suit your needs. To do this you will press the 'Network' button as you see in the image above:

Casambi Ethernet Gateway - 3.26 - REV1			
Generell Settings	To Casambi	From Casambi	Console
General Setting			
Network	System	Memory	
IP-Settings			
<div> DHCP <div>inactive</div> </div> <div> Hostname <div>casambi-gateway-B8D61AA3E38F</div> </div> <div> IP-Address <div>192.168.1.90</div> </div> <div> Subnet <div>255.255.255.0</div> </div> <div> Gateway <div>192.168.1.1</div> </div> <div> Nameserver 1 <div>8.8.8.8</div> </div> <div> Nameserver 2 <div>0.0.0.0</div> </div> <div>Save</div>			

On this page, we are going to configure the indicated values. Let's go with it:

- **DHCP:** It is a technology that allows the network to automatically assign IP addresses to each device. The problem is that we cannot predict what address it will assign us, and additionally the IPs provided automatically are temporary, so they can change over time. To read this device stably in the long term, we need this IP to be fixed forever, so we will disable this option.
- **IP-Address:** This field is one of the most important to guarantee access to this device. We must select one [IP](#) in the range in which we are working, and we must give a different IP to each LitherNet if we work with more than one. If you are working with an Adquio Lite, its LAN range is 172.20.20.x. The Adquio Lite controller will be at IP address 172.20.20.1, so you can use any IP in that range except this one. If you use Adquio Mini, then your client must provide you with the IP set to use.
- **Subnet:** Unless your client tells you otherwise, you can leave this field at the default value.
- **Gateway:** This is the IP address that your LitherNet device will use to connect to other systems, it is its '[Exit door](#)'. It is only necessary if you are going to activate services, such as connecting to the time-server. If you use Adquio Lite, the value will be 172.20.20.1. If, however, you use Adquio Mini, this value must be provided by your client.
- **Nameserver 1 y 2:** This is the service that will provide your LitherNet with the [converting domain names to IP addresses](#). As in the previous case, it is only necessary if you need your LitherNet to access external services, such as the time-server.

Restart Gateway
Reboot

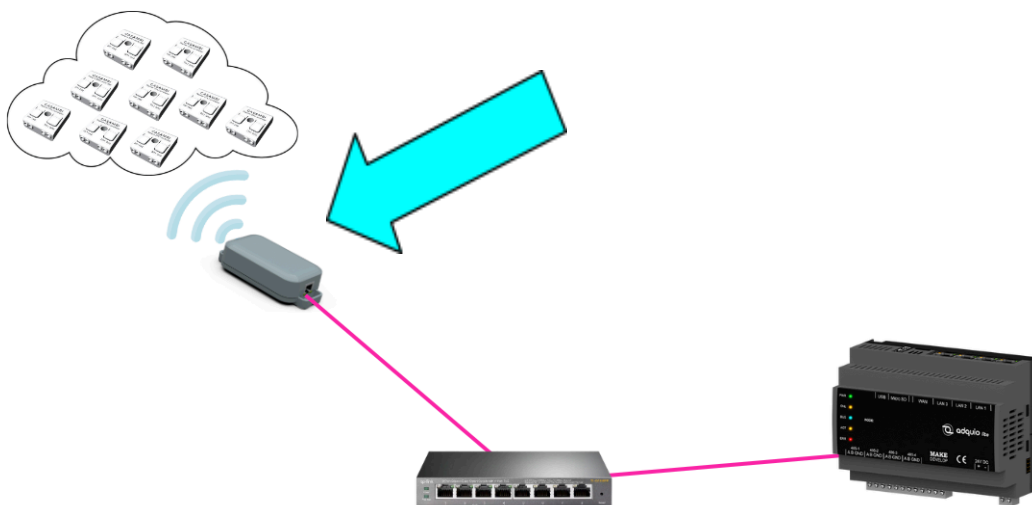
Once all the values are configured, we will click on 'Save', and at the bottom of this page on the 'Reboot' button



Remember that if you have changed the IP, when you restart you will no longer be able to access your Lithernet with the current configuration of your device. This is normal, as at the beginning of this section, if you want to access it again, you must put your computer in the same range that you have put your Lithernet.

We are done, you must repeat this process with each of your Lithernets. Remember that you can configure them all the same, except the IP address and the BACnet address, which must be different.

Getting list of variables with Yabe.



Yabe, they are the initials of 'Yet Another BACnet Explorer'. We are going to use it because it is free and very versatile software that allows us to explore our Lithernet, and obtain all the variables that it exports. You can download it here in its version for Windows:

<https://sourceforge.net/projects/yetanotherbacnetexplorer/>

You also have versions for Linux and Android available at this link:

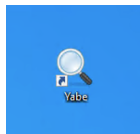
<https://sourceforge.net/projects/yetanotherbacnetexplorer/files/>

In its version for Windows it does not require installation, you simply have to download it and run it.

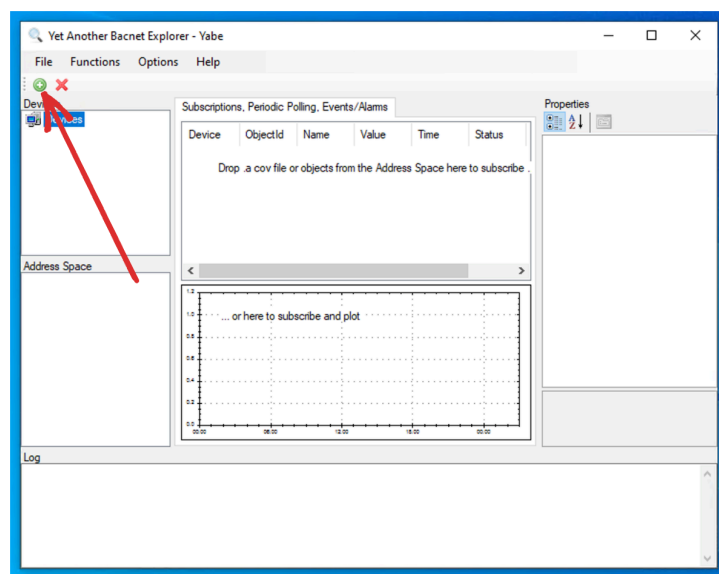


Before starting, it is essential that you make sure that there is no other device accessing your Lithernet. If there is, the entire process that we will explain to you below may present random errors.

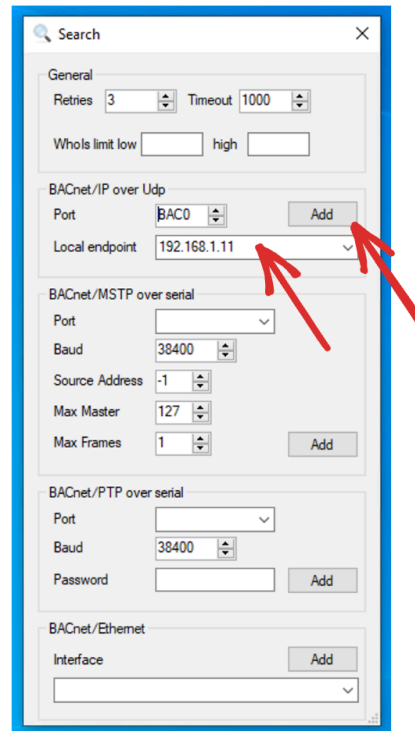
Look on your desktop or downloads folder for this icon:



By double-clicking on it, this window will open:



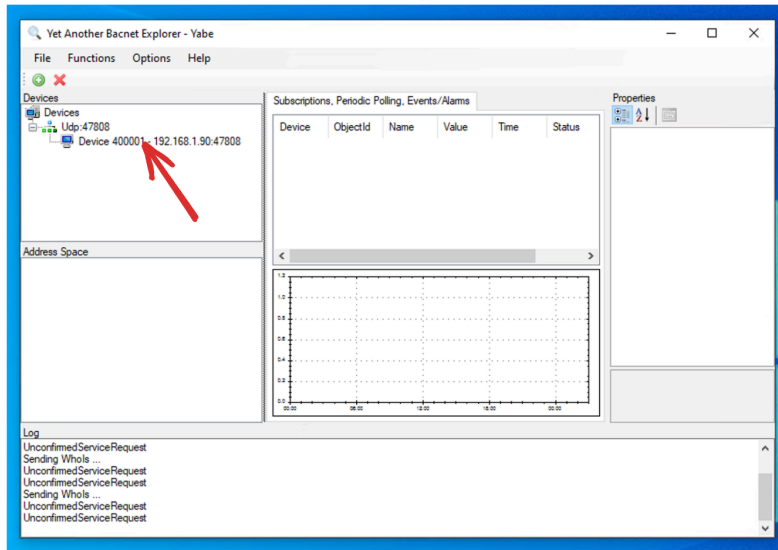
To begin, we are going to press the small green button at the top left, this will open the following window:



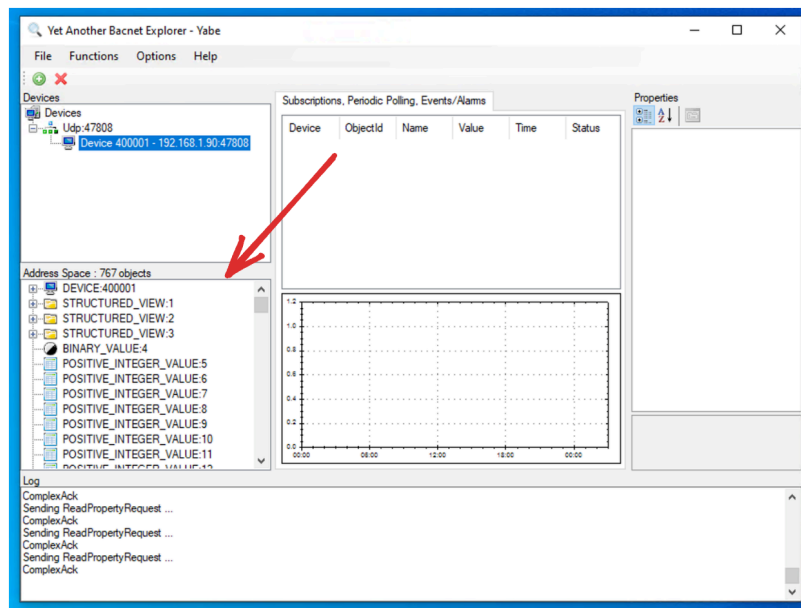
In this window there are many values, don't worry, we are going to use only two:

- **Local endpoint:** The first thing is to display the 'Local endpoint' field, you **must select the IP address of your computer, that** of the network adapter that you have connected to your Lithernet. It is very important that you select the correct one, otherwise Yabe will not find your device.
- **'Add' button:** This will add an item to the main window tree.

We are done 😊. Actually yes and no, with this, Yabe can now read all the variables of your Lithernet. Let's see them and how to act with them. By pressing the Add button, the program will return to the main screen like this:

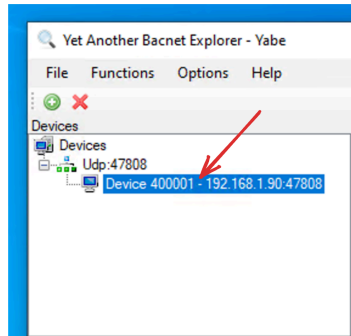


By clicking on the element we indicated, Yabe will begin scanning your Lithernet. You can see this because the item tree begins to grow, this process may take some time.

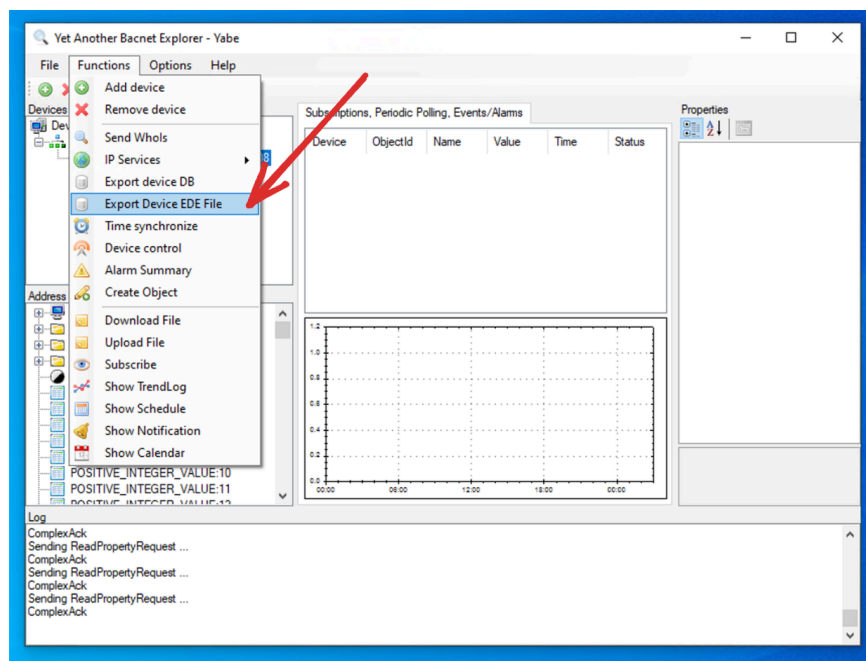


We must wait for it to finish loading all the elements before moving to the next step.

Once finished, we only need to export this information. First, make sure that you have your device node selected in the tree, as you see in the image.



We will go to the 'Functions' menu, and within it, to the option: 'Export Device EDE File'.



This will open a new window that allows us to give a name and location to our CSV file. In our case, we have called it 'Export' and the system will save it in our downloads' folder with the extension CSV (Comma Separated Values).

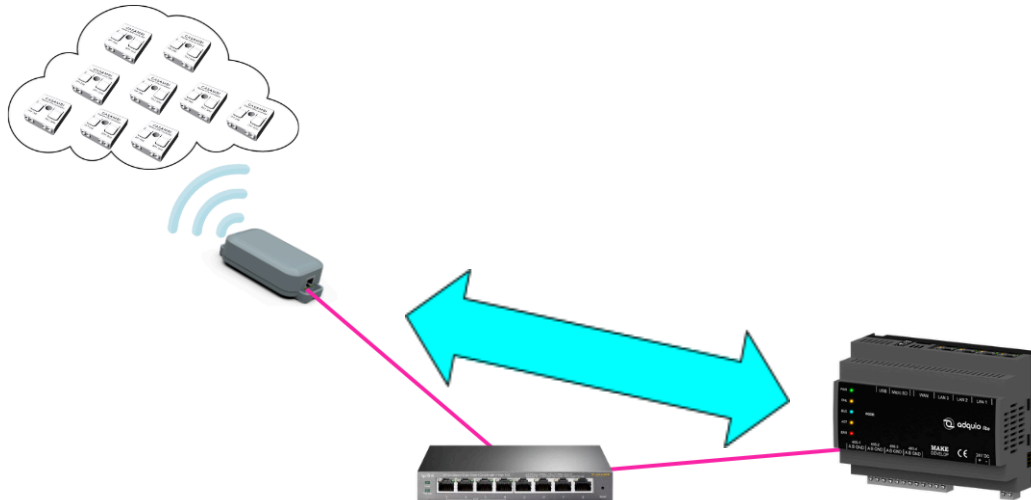
Now that we are done, remember the location of your file, since you will need it in the next step.



If you do this process with each of your Lihthernets, you must save the export files with different names that clearly identify each of them.

You can close the Yabe application, you will not use it again.

Converting the CSV file.



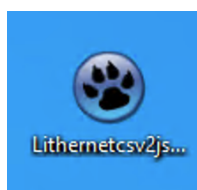
At Adquio, we always try to make the work of our partners easier. In this case, we have created a special program that allows you to automatically convert the file downloaded from Yabe to a format directly compatible with Adquio.

You can download this program from here:

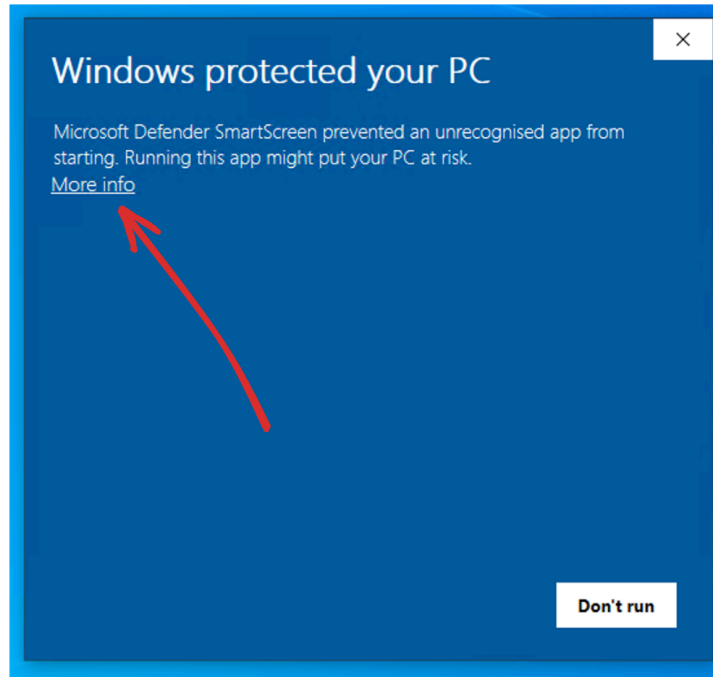
<https://adquio.com/wp-content/uploads/2023/09/Lithernetcsv2json.zip>

This program is currently available for Windows ©, and will soon be available for Mac © and GNU/Linux as well.

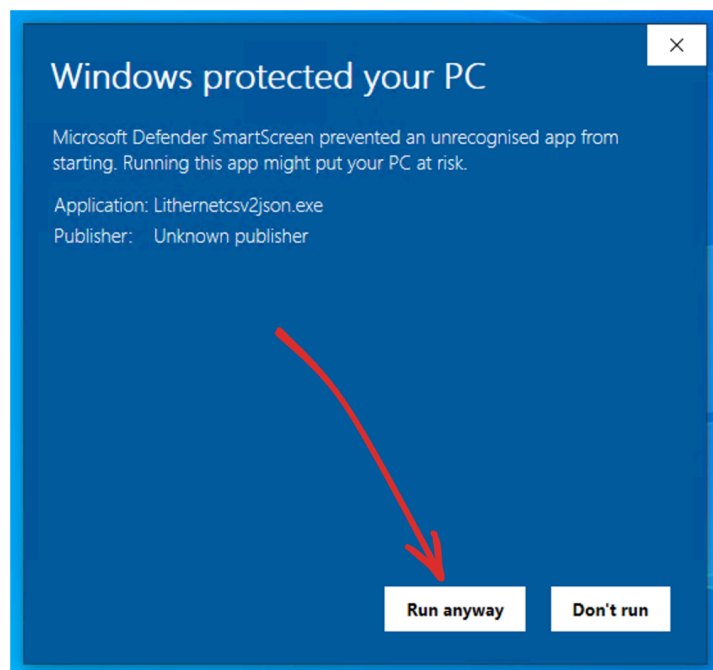
You will see that it is a compressed file, when you decompress it you will get an executable, you can put it directly on your desktop.



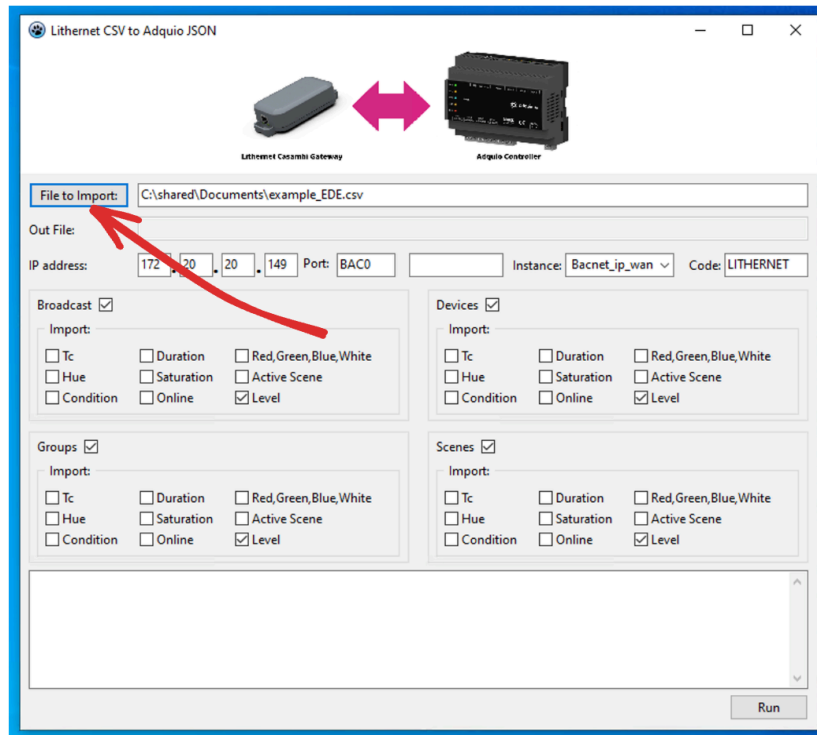
On Windows, executable programs downloaded from the Internet are always suspected of containing viruses, in this case we have downloaded it from a trusted website <https://adquio.com> 😊, so we can open it without problems. To do this, you must double-click on it, and you will get the following notice:



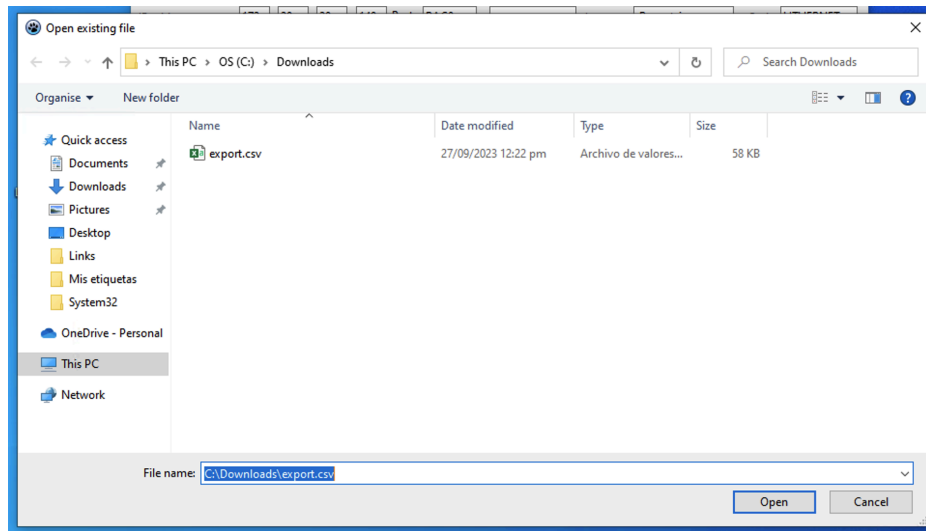
We will click on more information and...



We press the 'Run anyway' button, and finally we have our program ready to run.



The first thing will be to import the file from the previous step. To do this, press the button we indicated and select it.



Next, the program will load the file and automatically tell you what your output file will be. It will create a name with the current date and time so that they never conflict with others already created on your drive. It will leave it in the same folder as the source file with the name 'conversion', and then the current date and time.



At the moment, the program has not done anything, first we must select and indicate what we need for our specific case, let's start:

IP address: . . . Port: Instance: Code:

Remember that we are generating a file so that Adquio can automatically read all the Lithernet values, therefore, all the data that we are going to indicate will be used to access the Current device:

- **IP address:** It is the address of your Lithernet device. If you have left it at the default address you will see it as in the image, if not, you must go to the second chapter of this document where we set it.
- **Port:** It is always BAC0, it will be loaded by default.
- **Instance:** This value depends on what Adquio model you have, and how you have connected the Lithernet device, the possible values are:

- **Bacnet_ip_wan:** It is the correct option for Adquio Mini, Adquio Micro Server and Adquio Server. In Adquio Lite and Pro, it is valid if the controller can reach the Lithernet device through its WAN port, that is, if it is on the customer network and not on the internal network.
 - **Bacnet_ip:** Valid for cases in which you have an Adquio Lite or Pro, and you have connected your Lithernet to its LAN ports.
- **Code:** By default, it suggests 'LITHERNET'. It is the device name that you are going to use within Adquio as a code. If you have more than one you must put different codes here to be able to identify them, in our case we will leave it like this, but in lower case and we will put an example number:

IP address: 192 . 168 . 1 . 90 Port: BAC0 Instance: Bacnet_ip_wan Code: Lithernet01

Next, you can select what data you want Adquio to manage. We have previously done this in Lithernet when selecting what to export, but now we have the opportunity to detail our selection.

The screenshot shows four configuration sections, each with a checkbox and a list of data points to import:

- Broadcast:** ☒. Import: ☐ Tc, ☐ Duration, ☐ Red, Green, Blue, White, ☐ Hue, ☐ Saturation, ☐ Active Scene, ☐ Condition, ☐ Online, ☒ Level.
- Devices:** ☒. Import: ☐ Tc, ☐ Duration, ☐ Red, Green, Blue, White, ☐ Hue, ☐ Saturation, ☐ Active Scene, ☐ Condition, ☐ Online, ☒ Level.
- Groups:** ☒. Import: ☐ Tc, ☐ Duration, ☐ Red, Green, Blue, White, ☐ Hue, ☐ Saturation, ☐ Active Scene, ☐ Condition, ☐ Online, ☒ Level.
- Scenes:** ☒. Import: ☐ Tc, ☐ Duration, ☐ Red, Green, Blue, White, ☐ Hue, ☐ Saturation, ☐ Active Scene, ☐ Condition, ☐ Online, ☒ Level.

You will see that we have 4 main options indicated: Broadcast, Devices, Groups, and Scenes.

Here you must decide exactly what information you need your Adquio controller or your Screens or SCADA's to manage. Let's see what each section entails:

- **Broadcast:** As we explained previously, if you have selected this option in Lithernet, now here you can also select it or not. If you select it, Adquio will be able to send messages to the entire Casambi network, turning it on, off, regulating, changing color, etc. If you require this, leave this option checked, otherwise uncheck it.
- **Devices:** If you are going to need to read the values of each device individually, leave this option checked, if you are going to work only with scenes or groups, uncheck it. To manage the status of the devices and create alarms, (as we will see in later chapters), you need to select this option and check 'Online' and 'Condition'.
- **Groups:** Leave it checked only if you are going to work with groups, in many cases it is not used, since everything is managed by scenes, you decide.

- **Scenes:** Leave it checked only if you are going to work with scenes, in many cases it is not used, since everything is managed by groups, as before, you decide.



We remind you that we must select our needs as specifically as possible.
Fewer variables = Better response speed.

Let's now see what we have within each option, we start with the most important ones:

- **Level:** This variable allows you to turn on, turn off or regulate this luminaire, scene, etc. Its values range from 0 (off) to 255 (100% on) and any value in between is regulation.
- **Online:** It allows Adquio to know if each device is connected, it is not very useful in scenes or groups, but it is useful in devices. If you are going to need alarms, it is a good idea to select this option in devices.
- **Condition:** This is one of the most important variables in the devices if you are going to do individual monitoring of them. This variable provides the status of the drivers and luminaires, with a series of values such as temperature, lamp error, and a long list of other states. You can consult the complete list in this link: [Get the most out of your Casambi devices](#)

The rest of the variables from here are related to lighting, color, saturation, tone of white, etc. Select only the ones you require, for example, if your installation does not have RGB, never select this option, as it adds many variables to the communication.

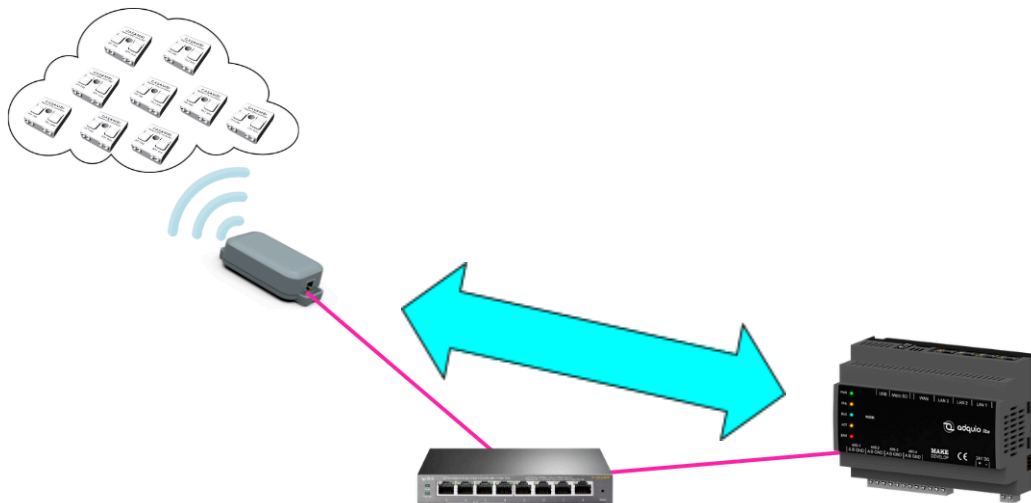
When you have finished your selection, you just have to press the 'Run' button, and the process of creating the output file will end.

Now you have two options:

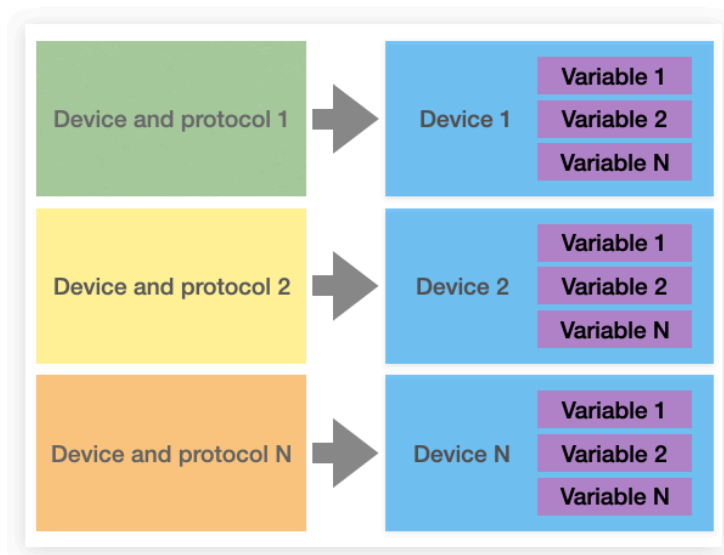
- Either use the file that was created for you, and you can see its location at the top (Out file).
- Or right-click on the bottom part, where you see a lot of text, select everything and copy it. In the next chapter, we will tell you where to paste it.

You are done in this part.

Importing file in Adquio



Before starting, we assume that you will have an Adquio controller at your disposal, to start it you will need a 24 volt DC power supply.



Adquio controllers work internally with an architecture of devices and variables. For Adquio, any other element it connects to (other controllers, expansions, gateways, etc.) is a device, and this device has variables. This standardizes any external device that provides data to Adquio into this simple architecture.

Additionally, Adquio also has the figure of modules, which are nothing more than drivers to connect with different types of protocols or physical media. All devices that Adquio manages always have a module associated with them. Examples of modules are: BACnet IP, Modbus RTU, etc. Additionally, in these modules, you can select the associated Adquio port on which it will work.

Adquio saves all configuration data for these devices in JSON format. Therefore, every device you configure manually will eventually be saved in this format. This represents an advantage for you, since it allows you to automate the creation of new and complex devices. Additionally, it allows you to copy one device from another very easily.

Well, we are going to base ourselves on this quality to send to Adquio all the values exported in the previous chapter.

But first, let's see how we connect to the Adquio controller and how we get to its web interface. To make the connection with the Adquio controller you have two main options, by cable or by Wi-Fi (valid for Adquio Pro, Lite, and Mini).

Connection with Adquio via Wi-Fi:



Step 1:

Be sure to **not having the WAN port connected**, (in Adquio Mini, simply that the network cable of its only port is not connected).



Step 2:

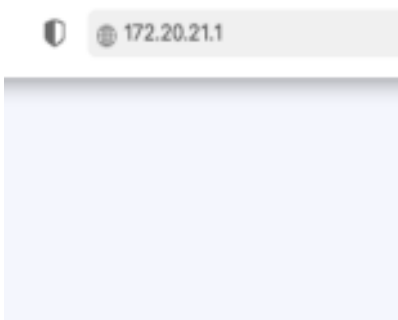
Feed the Acquisition, and wait a few seconds



Step 3:

On a laptop or tablet, in the list of available Wi-Fi, you will see one named adquio-XXXXXX

Connect to that Wi-Fi with the password: oiugda-0m3d



Step 4:

Once connected, open a browser and **see the address: <http://172.20.21.1>**



Step 5:

You will see the Adquio user input screen. Use the username and password that goes on all Adquio boxes.



Step 6:

It will ask you to change the password, **We strongly request that you do so.** And write it down in a safe place.



Step 7:

Now you can **enjoy all its functions!**

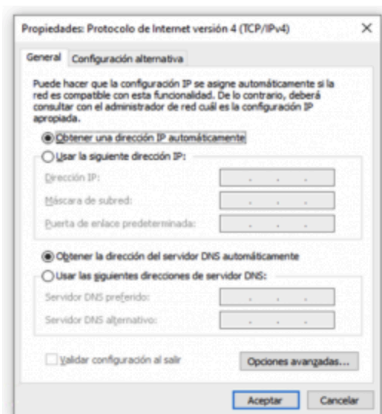
Connection with Adquio by cable:



Step 1:

Power the Adquio with a 24 V source, and wait a few seconds.

Plug a network cable into one of the LAN ports.



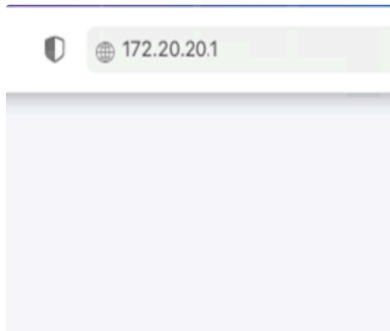
Step 2:

We connect the other end of the network cable to a computer and make sure that its network settings are set to automatic. (You need Gigabit Ethernet port)



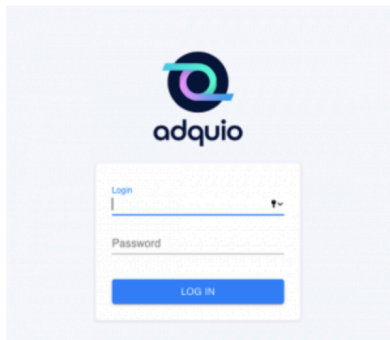
Step 3

We wait for the computer to obtain an IP address, normally the system will notify us of this.



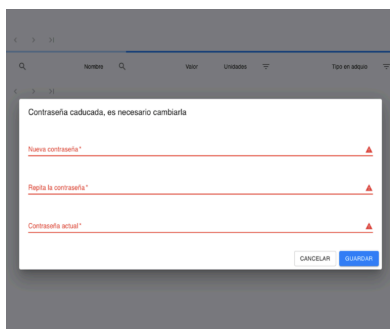
Step 4

Once connected, we open our favorite Web browser (We recommend Chrome) and go to the address: <http://172.20.20.1>



Step 5

We will see this user entry page, we must use the username and password that goes in all Adquio boxes.



Step 6

It will invite us to change the password for a personalized one, **we must do it**, since otherwise our Acquio will be vulnerable.



Step 7

We are now inside the Adquio Web Interface, we can now enjoy all its fantastic features!

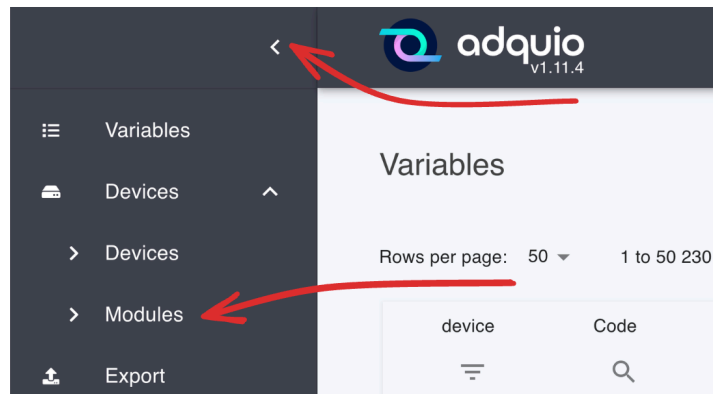
How to change language



At the top right you have available a button that allows you to switch between English, Spanish, or French. Select the one that best suits your preferences.

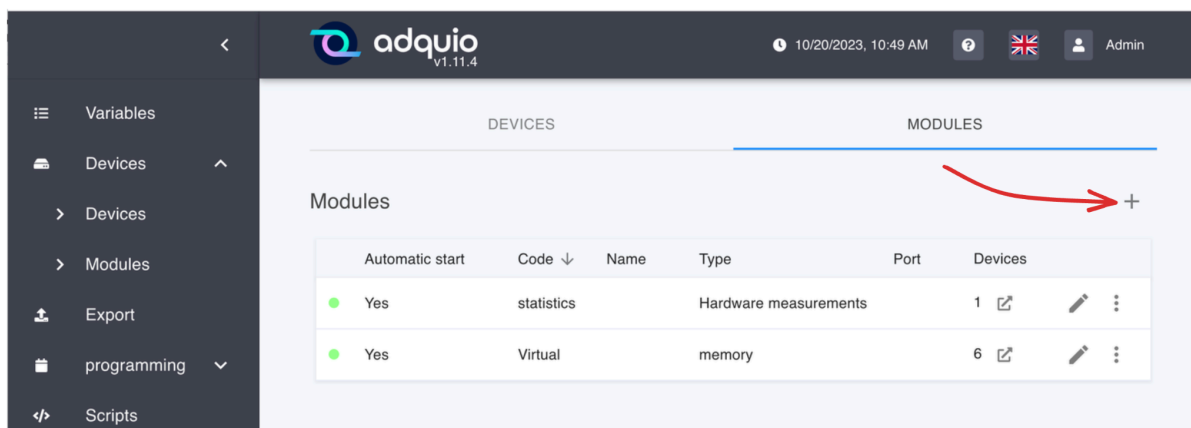
Creating a module

Ok, if you have followed the steps, you will already be inside the Adquio interface, well, let's go from here.



On the Adquio main page, to the left of the logo, you will see a little arrow, if you press it, the menu legends will open.

Click on 'Devices', and then on 'Modules', this will take you to this page:



Click on the '+' button to add a new module.

The screenshot shows a configuration page for a module. At the top, there is a toggle switch for 'Automatic start' which is turned on. Below this, there are two columns of settings. The first column has 'Type' set to 'BACnet IP' and 'Module type' which is empty. The second column has 'Code*' set to 'Lithernets' and 'Unique identifier' which is empty. An information icon (i) is next to the 'Code*' field. Below these settings is a section titled 'Connection data'. Inside this section, there is a 'Port' dropdown menu set to 'wan (br-wan)', a 'Physical connection port' field which is empty, a 'BACnet instance number*' dropdown menu set to '1', and a 'BACnet network port' field set to '0xBAC0'. At the bottom of the form, there are three buttons: 'SAVE' (blue), 'SAVE AND RETURN' (grey), and 'BACK' (grey).

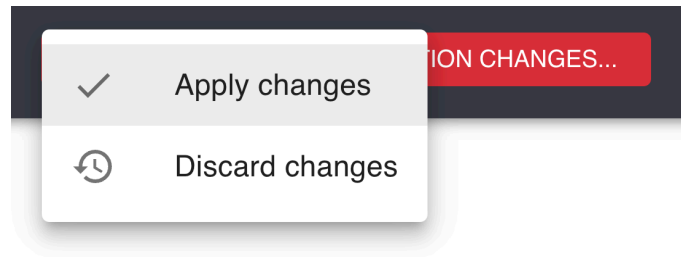
On the page that appears, we only need to cover three fields:

- **Type:** Display and select BACnet IP, since it is the protocol that we have configured in Lithernet.
- **Code:** A code that tells you that you are going to connect with this module. In our case, since we are going to connect Lithernets, we have named it Lithernets.
- **Port:** On an Adquio Mini, micro server, server, or Link, it will always be 'wan'. On the other hand, on an Adquio Pro or Lite it can be 'lan' or 'wan' depending on where you have connected your Lithernet, on the LAN or WAN ports of your Adquio.
- **BACnet instance number.** It is the number that you associate with Adquio on the BACnet network, normally 1.

We are done, click on 'SAVE' and this will cause a red button to light up at the top indicating that you have unconfirmed changes.



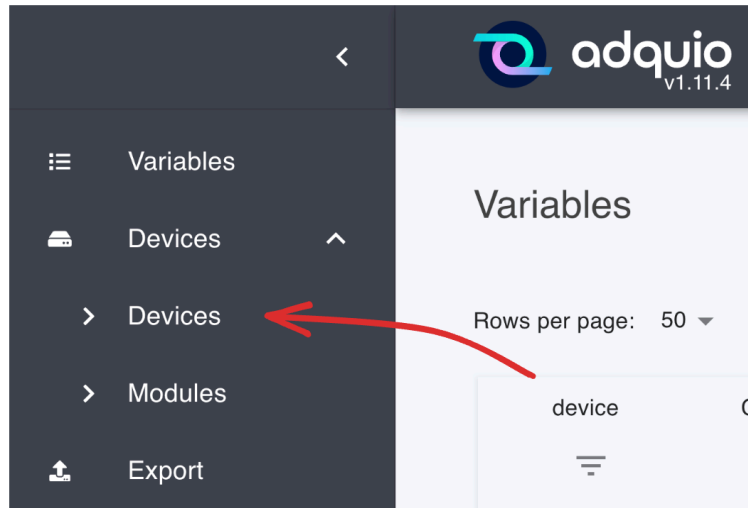
Click on it, and confirm the changes by clicking on 'Apply changes'.



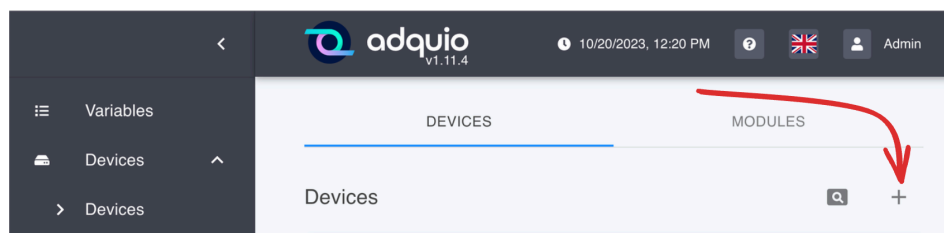
You already have the module created, **you only need to do this once**, since as you will see below, all Lithenet devices will share it. You should only generate more than one module if you have devices of this type on both the LAN and WAN ports.

Creating a device

Very good, we now have everything ready to import our device. To do this, we go to 'Devices', and again 'Devices'.



This opens the following page:



As in the previous case, click on the '+' button at the top right to create a new device:

Create device

Device identification

☒ Active

Type

Remote Adquio ▾

Code *

Name

Type of device

Unique identifier

Remote device code

Device code on remote system. It is only necessary to cover it if it is different from the one used in this Adquio.

Description

Importing the device

You won't do anything, just scroll down to the bottom of this page until you find this button:

SHOW JSON (ADVANCED)

Press it, and with this you will open a window where you will see some text.

HIDE JSON

JSON representation

```
{
  "schemaVersion": 5,
  "type": "adquio_client"
}
```

Simply delete all the text you see, and paste the one we copied in the previous chapter.

Adquio will read the pasted text and create the device for you, you can check it at the top where you will see that all the fields are now covered.

Create device

Device identification

☒ Active

Type	Code *	<input type="text" value="Lithernet01"/> <input type="button" value="i"/>	<input type="text" value="Name"/>
BACnet IP ▾	Lithernet01		
Type of device	Unique identifier		

You see that it has respected the code that we had indicated 'Lithernet01'

Connection data

Module

Multiple read variables

The maximum number of variables in each multi-read request, the maximum value is automatically calculated

Device instance *

MAC BACnet

We continue down and see that it has also brought us the device instance. What remains to be indicated is the module, since this cannot come from the import because we defined it in Adquio, click on the 'Module' drop-down menu and select the that we created in the previous case.

Connection data

Module

We click on the button on the right that we indicate:



And we will see that it has also correctly brought us the indicated IP address.

MAC Bacnet Generation Assistant

IP Address *

192.168.1.90

Port *

0xBAC0

CANCEL

CONVERT

We leave here with 'Cancel'.

Variables

☐

Code

🔍

Name

🔍

Supports
reading

Supports
writing

Type

≡

<input type="checkbox"/>	Scene_1_Level	Scene 1 Level	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	generic
<input type="checkbox"/>	Scene_2_Level	Scene 2 Level	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	generic
<input type="checkbox"/>	Scene_3_Level	Scene 3 Level	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	generic

Virtual variables

SAVE

SAVE AND RETURN

And finally, we check that it has correctly brought us all the variables that we had selected. In this case, the 'Level' variable for all scenes and groups.

We press the 'SAVE' button. As in the previous case, you must confirm the changes in the red button above.

You have finished.

With this, Adquio will now have full access to the Lithernet device and all the variables of the Casambi network that we have selected.

Processing information in Adquio



As you know, Adquio controllers process information with the JavaScript programming language. Therefore, everything we will see below will be based on this language.

The most normal information processing that is usually done in Adquio are, for example, the following:

- Synchronize scenes between different networks.
- Manage states of all network devices.
- Raise alarms so that those responsible for the installation are notified.

Let's look at the first example mentioned, since we will dedicate a specific chapter to the following ones.

Synchronize scenes between different networks

Within Adquio we have two different options to carry out this work.

- The first is to manage the scenes and send commands to all Casambi networks to turn them on, off or regulate them.
- And the second is to use one of the networks as 'Master' and have the others copy the changes from the first.

In this case, we are going to see the first point, since the second has a small delay until the Lithernet gateway notifies a change in a scene.

That we within Adquio manage the scenes means that we will not take a Casambi network as 'Master', but rather that we ourselves will take the initiative to send commands to all networks.

You can use this system when Adquio receives orders from outside to manage the lighting, and this can happen if it receives orders from:

- Another PLC.
- An external BMS.
- An Adquio control screen.
- An SCADA.

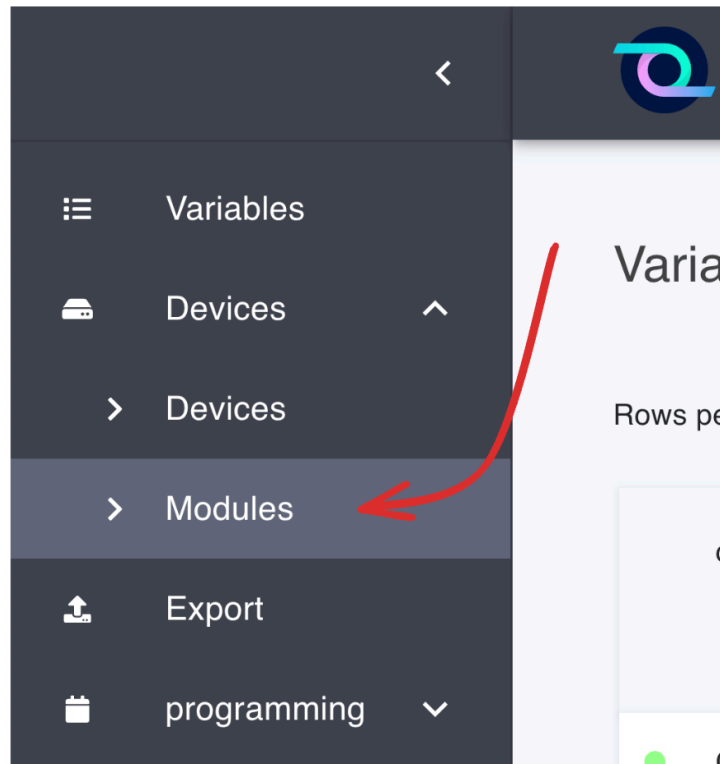
It is the most effective and fastest way to send commands to the Casambi network, since it is immediate and does not need to wait for any refreshment to work.

Let's see then how to perform this task in Adquio. The first thing we need to do is create a variable for each scene. This allows us to export them so that a BMS, SCADA or screen can use them, and, on the other hand, associate the sending of information to the Casambi networks in the state change events of our variables.

Creating a module

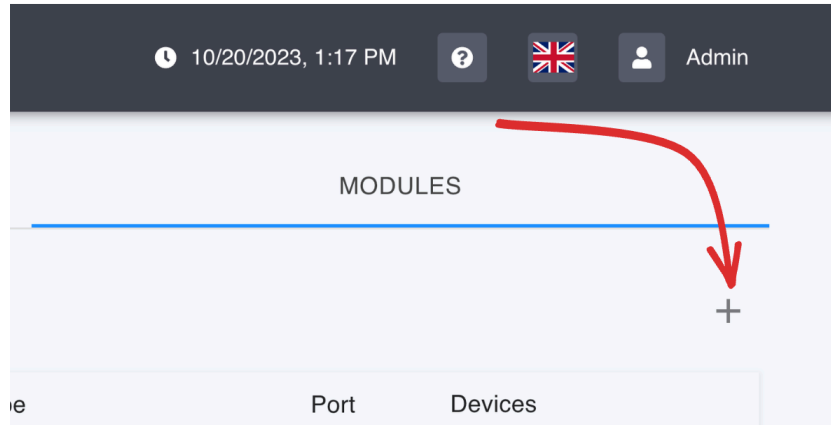
Let's see then how to generate these variables in Adquio.

The first thing, is to explain that in Adquio, in addition to having devices and variables (that correspond to physical machines), you also have virtual devices and variables. You can generate it for internal use, (and that do not belong to anything real or physical). In this case, we are going to create a device called Scenes and inside we will create a variable for each scene, let's go with it:



As you already know from previous steps, in Adquio each device needs a module, therefore, we must start here, by generating a module for our virtual variables.

To do this, in the left menu of Adquio, you must go to Devices, Modules, as you see in the image.



Once inside, press the '+' button at the top right. This will take you to this page, where you must select the Type: 'Memory' and the code you want, in the example we have put 'Virtual'.

Instantiate the module

Module instance identifier

☒ Automatic start

Type Code*

memory ▼ Virtual i

Module type Unique identifier

SHOW JSON (ADVANCED)

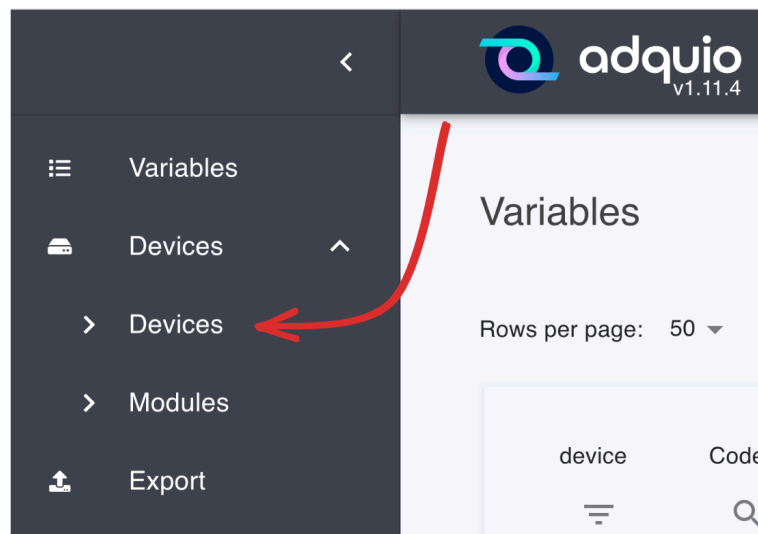
SAVE

SAVE AND RETURN

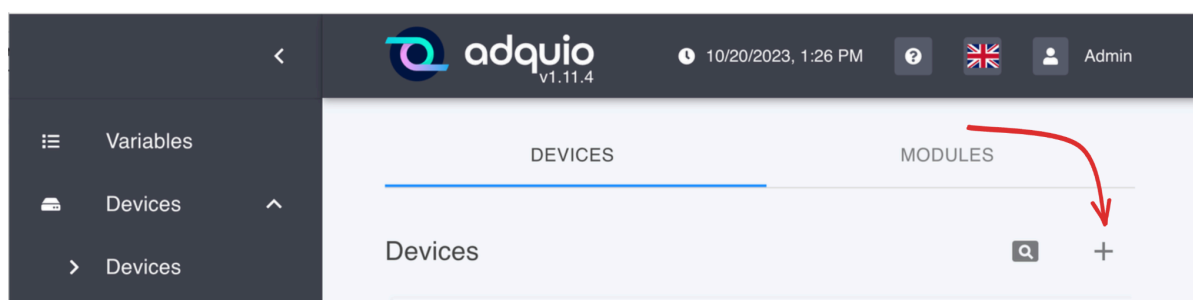
BACK

We save and exit. Now we can create our virtual device, let's go with it.

Creating a device and its variables

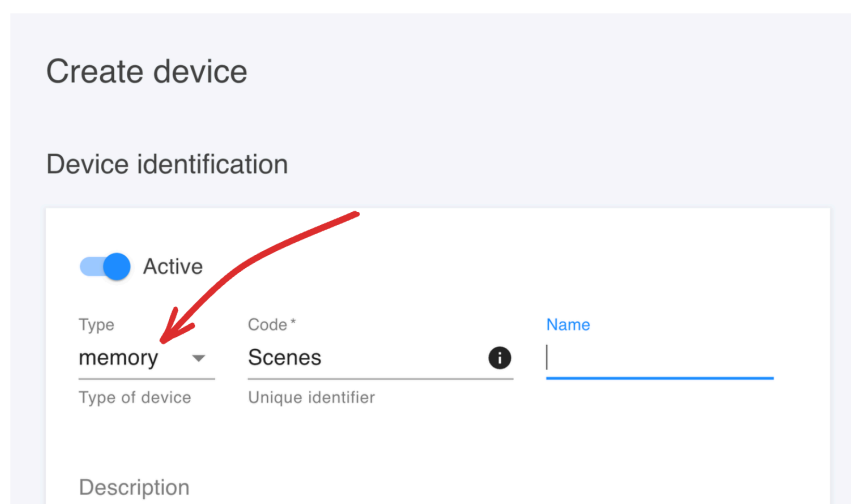


In the left menu, we now enter the 'Devices' option



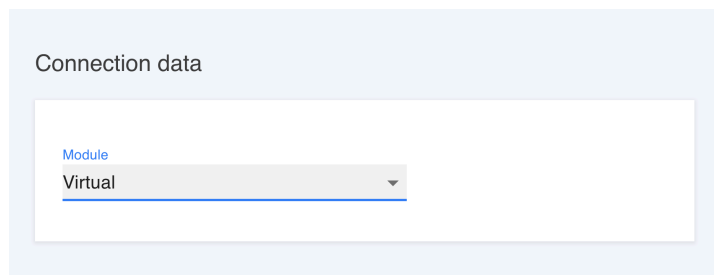
Click on the '+' button at the top right to create a new device:

The page that appears now is where we are going to create our device.



The most important thing here is that in type you select Memory, in code you can put whatever you want, as long as it reminds you of what it really is and that you put it without spaces or symbols. In our case, we have put 'Scenes'.

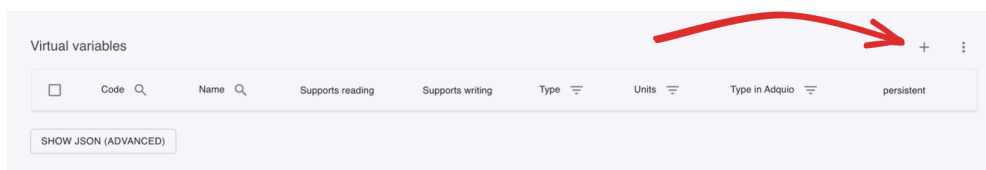
At the bottom is where we have the possibility of selecting the module that we generated in the previous point.



Connection data

Module
Virtual

Now we are going to generate the virtual variables that will launch the Casambi scenes. To do this, at the bottom right, we click on the '+' button.



Virtual variables

	Code	Name	Supports reading	Supports writing	Type	Units	Type in Adquio	persistent
<input type="checkbox"/>								

SHOW JSON (ADVANCED)

On the page that appears, you only have to pay attention to three fields:

- **Code:** It must be unique to this device, in our case we are going to call it 'Scene', and then a counter. 'Scene01'
- **Name:** You can put whatever you want, you really have to indicate here the explanation of what you are creating.
- **Type in Adquio:** Since in this case we know that we are going to work with values between 0 and 255, a 16-bit integer is more than enough. Then, we select '16-bit integer'

Finally, at the bottom, it is very important that we check the 'Persistent' option.

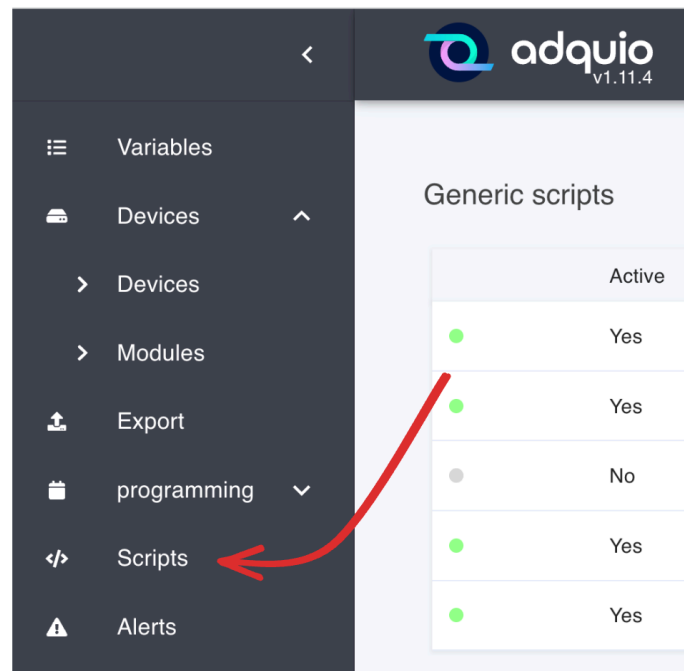
We save and repeat this process as many times as you have scenes. Each one, with its code numbered consecutively.

Perfect, you have now finished preparing everything you need to start scheduling the launch of scenes.

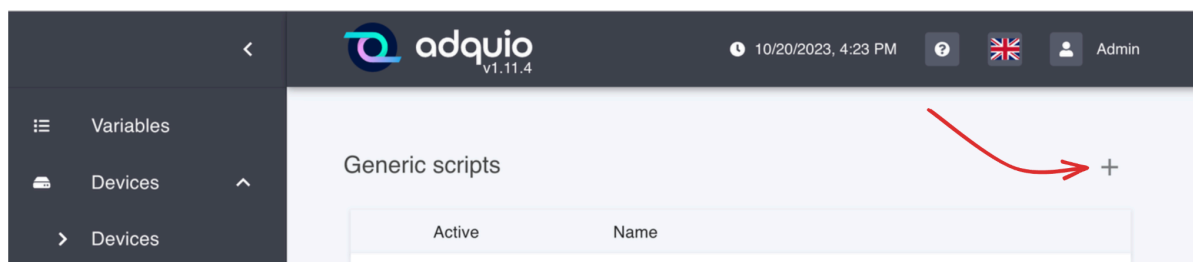
Programming the connection with scenes

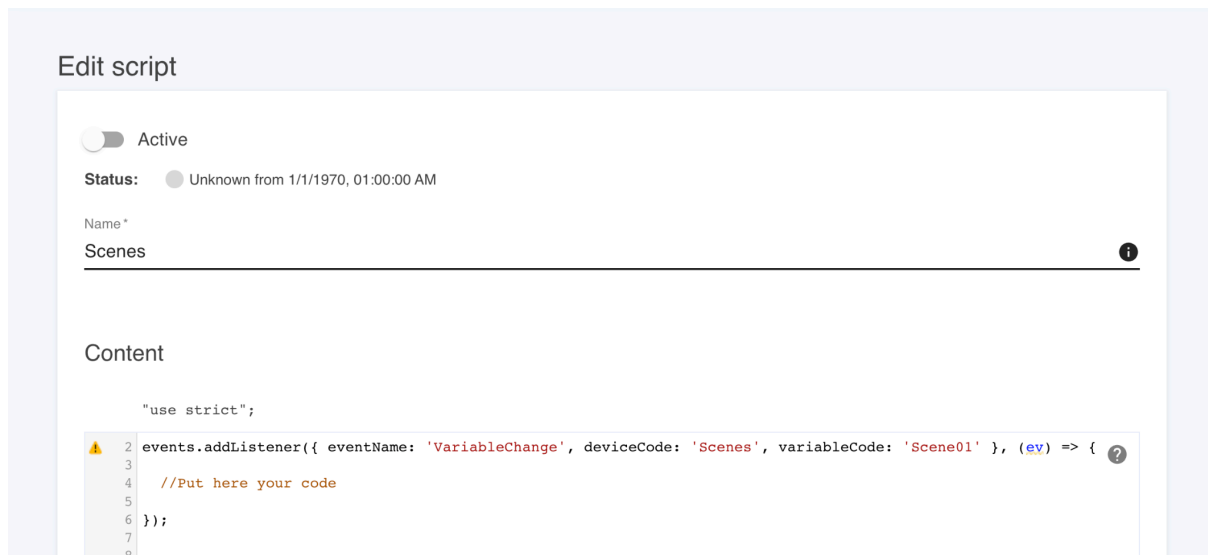
The most anticipated moment has arrived, where we will finally see how to program communication with Casambi scenes in JavaScript through the Lighthouse gateway.

First, we must go to the Adquio option that allows us to create scripts. So, in the main menu, you will go to 'Scripts'.



Once inside, you will see the list of scripts that you already have to generate a new one, click on the '+' symbol that you see at the top right.





As you see, when entering the option, the first thing we must do is give our Script a name, in this case, we have called it “Scenes”. At the bottom, called 'Content', is where you are going to program.

Now, we are ready to start programming. Previously, we mentioned that we will use **events** of our virtual variables to launch the Casambi scenes. Well, in Adquio the way to react to events is as follows:

We leave you the code here in case you need to copy and paste it:

```
events.addListener({eventName: 'VariableChange', deviceCode: 'Scenes',  
variableCode: 'Scene01' }, (this) => {  
  
//Here your code  
})
```

Actually, you can copy and paste this code, since it is always the same, you simply have to replace the device and the variable with yours, and you will have it.

Let's look at it carefully:

- “`events.addListener({eventName: 'VariableChange'}`,”: This part is fixed and tells Adquio that we want to schedule an event that reacts when the value of a variable changes.
- “`deviceCode: 'Scenes'`,”: Now, you just have to indicate the code of your device, the one that contains the variable you want to monitor. In our case, we have indicated the one we had created in the previous step 'Scenes'.

- “variableCode: 'Scene01' },”: As you will already assume, now we only have to indicate the code of our variable. In this case, we have used the first one that we have generated.
- “(this) => {” 'ev' contains the value of the variable that comes to us from the event, and that made it jump, it is a very useful value, since it allows us to use it directly within the event. They are the initials of 'Event Value'.

Well, now we will see how to send information to the Casambi network, we already know that we must react for scene 1.

In Adquio to write to a device you must use the command:

Content

```
"use strict";
2 events.addListener({ eventName: 'VariableChange', deviceCode: 'Scenes', variableCode: 'Scene01' }, (ev) => {
3
4   devices.write('Lithernet01', 'Scene_1_Level', ev.newValue)
5
6 })
7
8
```

As you can see, it is very easy, the command is 'devices.write', and then we will pass three values as parameters:

- First, **the device** we want to write about, in our case, 'Lithernet01'.
- Next, on which **variable** we are going to do it, in this case 'Scene_1_Level'.
- And finally, what **value do we want to give to this scene**, in this case, this value is given to us by the event in the variable 'ev', and the way to use it is this: 'ev.newValue'.

As in the previous case, we leave you the code in case you need to copy it.

```
events.addListener({ eventName: 'VariableChange', deviceCode: 'Scenes',
variableCode: 'Scene01' }, (ev) => {
  devices.write('Lithernet01', 'Scene_1_Level', ev.newValue)
});
```

Let's review and summarize what we have done:

- We decided that Adquio will be the one to launch the scenes.
- We create a device and virtual variables to do this.
- Furthermore, we have programmed an event in one of those virtual variables, so that when it changes, it sends the value to the Casambi scene through the 'Lithernet01' device.

With this, we now have control of a scene from Adquio. Obviously, if you have more than one, you just have to repeat the writing instructions, look at this example:

```

events.addListener({ eventName: 'VariableChange', deviceCode: 'Scenes',
variableCode: 'Scene01' }, (ev) => {

devices.write('Lithernet01','Scene_1_Level',ev.newValue)
devices.write('Lithernet02','Scene_1_Level',ev.newValue)
devices.write('Lithernet03','Scene_1_Level',ev.newValue)
devices.write('Lithernet04','Scene_1_Level',ev.newValue)
devices.write('Lithernet05','Scene_1_Level',ev.newValue)
devices.write('Lithernet06','Scene_1_Level',ev.newValue)
});

```

With this, you are commanding scene 1 to be activated in six Casambi networks.

Let's see how to ensure this. In the example above, if writing to a Lithernet device fails, the program breaks at that position and will not execute what is below, therefore scene 1 will be activated on Casambi networks, only up to the one given the mistake.

To avoid this, in JavaScript we have the 'try' statement that allows us to manage errors and keep the program working.

```

3
4   try {
5       // Here your code
6   } catch(err) {
7       //Here your code to manage the error
8   }
9

```

Consequently, if we apply this to our code, it looks like this:

```

events.addListener({ eventName: 'VariableChange', deviceCode: 'Scenes',
variableCode: 'Scene01' }, (this)=> {

try {devices.write('Lithernet01','Scene_1_Level',ev.newValue)
} catch(err) {
logger.error('Error writing on device Lithernet01 '+err);
}

});

```

As you see, the 'write' instruction is the same, but now protected within a structure that manages its possible errors.

In this example we have introduced the 'Logger' figure, which allows you to display messages at the bottom of the code editor, very useful for debugging your code. And finally, if you apply this to all 6 networks, you will have this:

```
events.addListener({ eventName: 'VariableChange', deviceCode: 'Scenes',
variableCode: 'Scene01' }, (ev) => {
  try {
    devices.write('Lithernet01', 'Scene_1_Level', ev.newValue)
  } catch(err) {
    logger.error('Error writing on device Lithernet01 '+err);
  }

  try {
    devices.write('Lithernet02', 'Scene_1_Level', ev.newValue)
  } catch(err) {
    logger.error('Error writing on device Lithernet02 '+err);
  }

  try {
    devices.write('Lithernet03', 'Scene_1_Level', ev.newValue)
  } catch(err) {
    logger.error('Error writing on device Lithernet03 '+err);
  }

  try {
    devices.write('Lithernet04', 'Scene_1_Level', ev.newValue)
  } catch(err) {
    logger.error('Error writing on device Lithernet04 '+err);
  }

  try {
    devices.write('Lithernet05', 'Scene_1_Level', ev.newValue)
  } catch(err) {
    logger.error('Error writing on device Lithernet05 '+err);
  }

  try {
    devices.write('Lithernet06', 'Scene_1_Level', ev.newValue)
  } catch(err) {
    logger.error('Error writing on device Lithernet06 '+err);
  }

});
```

You can repeat this in all your scenes and on as many networks as you need.



Remember that the variables that are going to trigger these actions are virtual, therefore, something has to modify them so that their values change. Consequently originate the event, this 'something' will be a BMS from the outside, an SCADA or by example an Adquio screen. We will see in [next chapters](#) how to export them to make them available to these external systems.

Managing errors and statuses in Adquio



When connecting to a Casambi network through Lihernet, Adquio has two variables on each device that inform it of its status:

- **Online:** Indicates whether a device that had been adopted into the Casambi network is currently connected or not, its value should be 1 if everything goes well.
- **Condition:** It tells us a code that tells us if everything is going well, and if not, it tells us what is happening. We have codes to warn us of excessive driver heating, excess current, lamp failure, etc. We will see them in detail later.

As we have seen in the previous chapter, Adquio works event-oriented, and in this case, this fits us perfectly, since we only have to monitor two events and we will have total error control of the Casambi network. Before we had connected to an event of a virtual variable, now we are going to use the events of the variables of the Casambi network.



In the example we have been following, we had only chosen to import the 'Level' variables of the scenes into Adquio. In order to perform this error checking, that we are looking at now, you will need to select all the devices and the 'Online' and 'Condition' variables in the ['Converting the CSV file'](#). And previously when you export the variables in Lihernet as we explain in the chapter ['Configuring Lihernet Gateway'](#)

```
events.addListener((ev) =>
  ev.eventName === 'VariableChange' &&
  ev.deviceCode.startsWith('acquio-unit_') &&
  ev.variableCode === 'online', (ev) => {
//Put your code here
});
```

This event will be executed when any device on our Casambi network changes its Online variable from 1 to 0, or vice versa. It will give us all the necessary data to be able to report it, let's see how it works:

Capture all events and leave the value in ev (Event Value).

- Next, we check that the event is of the type we are interested in, `VariableChange` (Adquio manages more types of events, such as device state change events).
- Then the condition is added that the device name must begin with 'adquio-unit_' (In this installation, the devices go from 001 to 250).
- And finally, it is indicated that we only want the events of the 'online' variable.

As you can see, all conditions must be met for the event to finally jump and execute our code. With these simple lines (which you can copy and paste into your Adquio), you are managing the status of all your devices connected to Casambi. Of course, inside this event and accessing the 'ev' variable you will have access to all the values, already mentioned above.

Some of the interesting properties that 'ev' provides us are:

- `ev.deviceCode`
- `ev.variableCode`
- `ev.newValue`

As you can see, with Adquio you have a simple and fast way to manage all the events of your Casambi network. And not only this, but (as you will see in [later chapters](#)), you also have a way to notify incidents, so you have a complete tool, also to manage incidents on your networks.

This entire example is also applicable to the 'Condition' variable, you just have to copy the example and change 'Online' to 'Condition'.



Keep in mind how you write the device and variable codes in Adquio, these are case-sensitive, therefore, always observe how you have called your variables and devices and always use them in the same way.

The values that you can receive through this 'Condition' variable are the following:

- **0: Ok** Everything is going well
- **1: Over Heating** Excess heat in the driver
- **9: Over Load** Excess luminaires in the driver
- **128: Ok** Everything is going well
- **129: Thermal Overload** Excess heat in the driver
- **130: Lamp Failure** The lamp connected to the driver is failing
- **131: Driver Failure** The driver is failing
- **133: Incompatible Hardware** There is some incompatibility in that driver or lamp
- **134: Hardware not Found** The driver does not detect the lamps
- **135: Configuration Fail** The driver is incorrectly configured
- **160: Ok** Everything is going well

Launching Alarms in Adquio



The previous chapter has been left a little incomplete, since we have been able to receive errors from any device on the Casambi network, but we have not been able to disseminate them to notify the correct people.

The combination of Adquio and Adquio Cloud(BMS) allows us to do exactly this; to make the alarm messages reach the right people, classifying them in different departments and selecting in our alarm which department it should reach, depending on the type of alarm we are managing.

Let's see how to do this by completing some previous examples. First, with the 'Online' variable.

```
events.addListener((ev) =>
  this.eventName === 'VariableChange' &&
  ev.deviceCode.startsWith('acquio-unit_') &&
  ev.variableCode === 'online', (this)=> {

  if (this.newValue ==0) {
    equus.alert('alert5', 1, 'Device error'+ev.deviceCode + ' is no longer online',
    0, 3600, ['maintenance']);
  }
});
```

First, we use an 'if' to check if the 'Online' status has gone to 0, since if it has gone to 1 we should not raise any alarm because it means that the device is available again.

As you see, we use the 'equus.alert' command, let's see what parameters we should pass to it:

- Alert code, if we want to type them here, we can indicate a free code to do so. In the example, we have used 'alert5'.
- Priority from 0 to 5, 0 is the highest. In the example, we have put 1.

- Text of the message, in the example: 'Device error '+ev.deviceCode +'is no longer online'. As you can see, we have integrated the device code into the message to identify which one is failing.
- Value or variable that we want to appear in the alarm, you can indicate a constant, or a variable so that its value is reflected with the alarm message.
- Expiration time: in this case 36000, it is measured in seconds, therefore, in this example we are indicating that it expires in one hour if the alarm does not repeat.
- Finally, in brackets is the department or departments to which this alarm is intended, they can be one, or several separated by commas, and must be previously defined in the BMS Adquio Cloud platform.

Let's now look at a similar example, but with the 'condition' variable.

```
events.addListener((ev) =>
  ev.eventName === 'VariableChange' &&
  ev.deviceCode.startsWith('acquio-unit_') &&
  ev.variableCode === 'Condition', (this)=> {

    let msg = ''

    switch (ev.newValue) {
      case 0: msg = 'Ok';break;
      case 1: msg = 'Over Heating';break;
      case 9: msg = 'Over Load';break;
      case 128: msg = 'Ok';break;
      case 129: msg = 'Thermal Overload';break;
      case 130: msg = 'Lamp Failure';break;
      case 131: msg = 'Driver Failure';break;
      case 133: msg = 'Incompatible Hardware';break;
      case 134: msg = 'Hardware not Found';break;
      case 135: msg = 'Configuration Fail';break;
      case 160: msg = 'Ok';break;
    }

    if (msg !== 'Ok') {
      equs.alert('alert5', 1, 'Device error on '+ev.deviceCode +' '+msg, 0, 3600,
['maintenance']);
    }
  });
```

In this new example, we evaluate the number that we received from the 'condition' variable and convert it to its equivalent message.

Below, we send the alert, only if the message is not 'Ok'.



The Adquio Cloud service is contracted independently and includes many more features than alarm management. We invite you to inform yourself by clicking [here](#).

Exporting values in Adquio

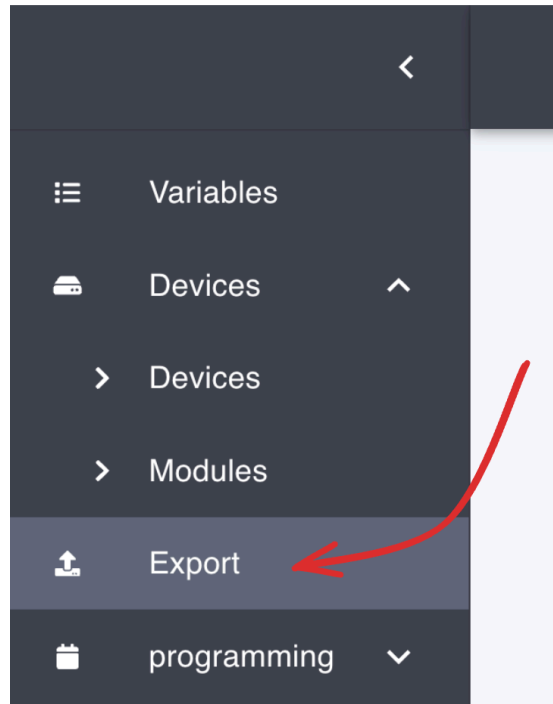


Now, that we know how to send commands to the Casambi network. And we can also manage its status and possible incidents. We are going to see how to export our virtual variables so that any other device, from outside our Adquio, can read and write to them. These devices can be the ones you already know, SCADA, BMS or control screens, among others.

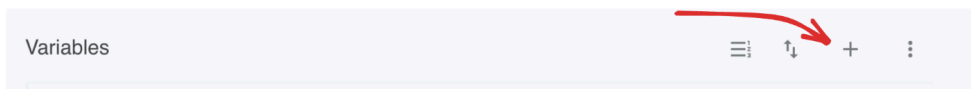


If you do not want to do any processing on the variables, you can directly export the variables you receive from your Luthernet here. In this way, you are giving direct control of the Casambi network to any external device.

So let's see how to do it. First, in the Adquio interface, we will go to the 'Export:' option.

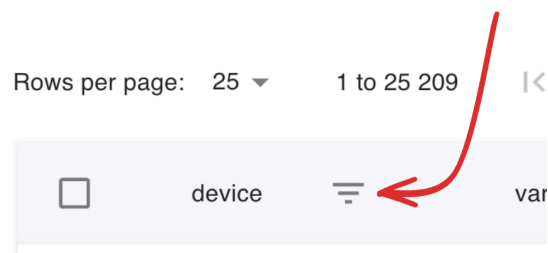


Once inside, you can leave everything as is and press the '+' button that you will see at the top right:



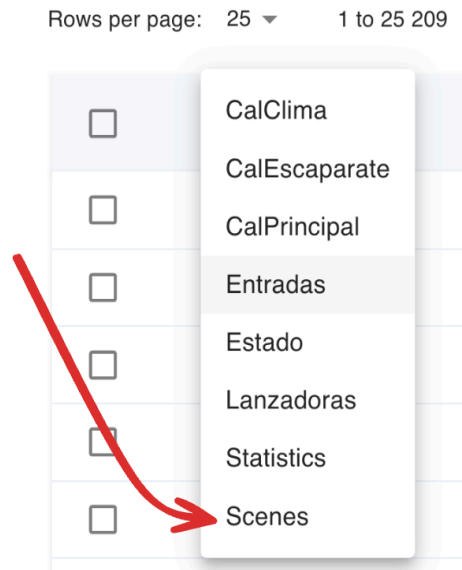
By doing this, a selection window will open where we can select our scenes.

Variable selector

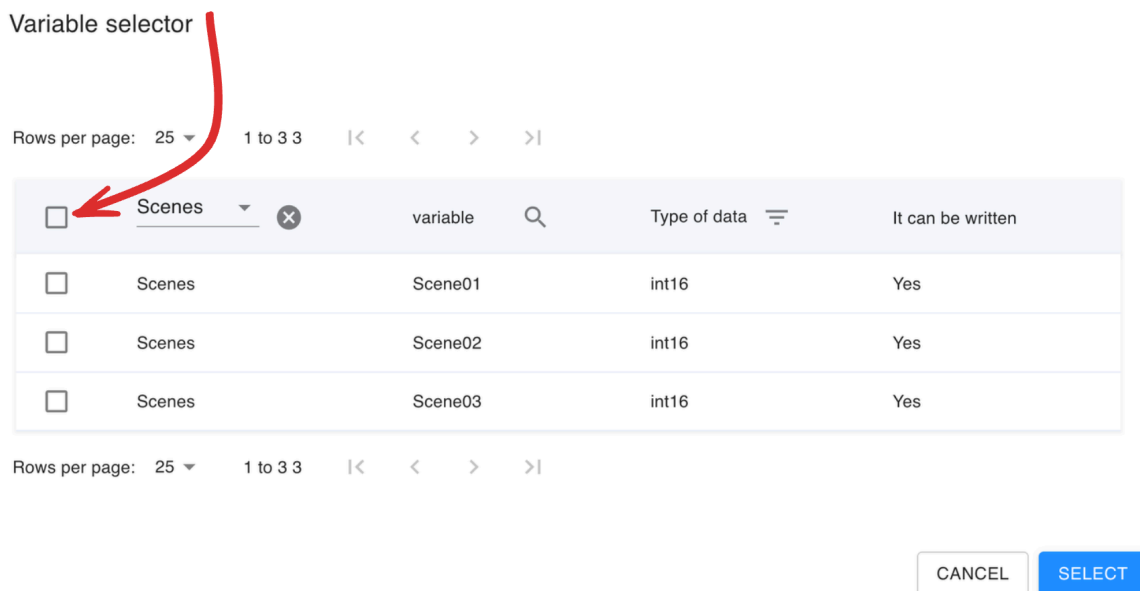


Within this screen, all the variables that Adquio manages will appear. To make their selection faster and more comfortable, we have this button that if pressed allows us to filter that list of variables by device.

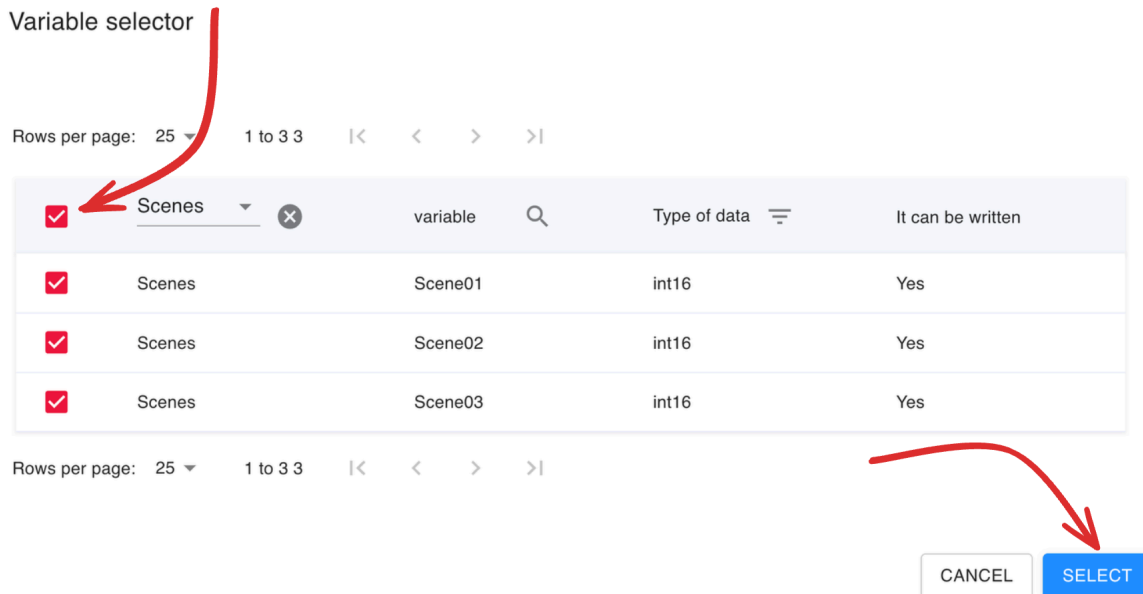
Variable selector



On the list of devices we are going to select the one that we have created to store our virtual variables corresponding to the scenes, in this case, 'Scenes'.



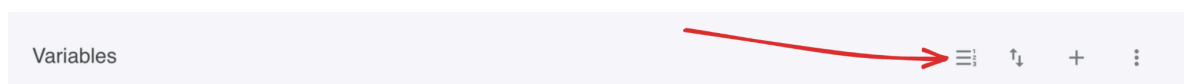
Now, in the list, you will only see the variables of the previously selected device; if you click on the grid at the top left, you will select all of them.



Ok, click on the 'Select' button at the bottom right, and we will exit the window. This will add all those that you just selected to the variables that you already had in the export list, if you go to the end of this, you will be able to see them.

<input type="checkbox"/>	Estado#Escaparat	<input checked="" type="checkbox"/>	Integer (16b)	0 items	Integer (16b)	Holding register	240		
<input type="checkbox"/>	Entradas#CodigoEncargadaP	<input checked="" type="checkbox"/>	Integer (16b)	0 items	Integer (16b)	Holding register	241		
<input type="checkbox"/>	Scenes#Scene01	<input checked="" type="checkbox"/>	Integer (16b)	0 items	Integer (16b)	Holding register			
<input type="checkbox"/>	Scenes#Scene02	<input checked="" type="checkbox"/>	Integer (16b)	0 items	Integer (16b)	Holding register			
<input type="checkbox"/>	Scenes#Scene03	<input checked="" type="checkbox"/>	Integer (16b)	0 items	Integer (16b)	Holding register			

Very good, we already have them on the list. We see that in the right column, they still do not have an assigned address. Let's put it on. You can do it manually by clicking on the pencil that you see in each row on the left. If you have many variables, this process becomes very laborious. To do it automatically, you have a button at the top of the list of variables. In this screenshot we show you which one it is:



When you press it, a window will open for you to indicate in which direction you want to start, normally you can enter 1, since the system will automatically skip the ones that are already in use. In any case, you can indicate any value, if for example you want your variables to start in a round address. For example 300.

Automatic Modbus address assignment

Initial address (holding register) *

1

CANCEL
SAVE

In our case, we have indicated 1 so that the system assigns the following ones to those already in use. When you click 'Save', Adquio will do the work. If you now go down to the end of the list, you will see that they have already been assigned their address, and that they go two by two, since the variables we export are 16 Bits, 2 bytes.

<input type="checkbox"/>	Scenes#Scene01	<input checked="" type="checkbox"/>	Integer (16b)	0 items	Integer (16b)	Holding register	242		
<input type="checkbox"/>	Scenes#Scene02	<input checked="" type="checkbox"/>	Integer (16b)	0 items	Integer (16b)	Holding register	243		
<input type="checkbox"/>	Scenes#Scene03	<input checked="" type="checkbox"/>	Integer (16b)	0 items	Integer (16b)	Holding register	244		

We just need to save all the changes, and we will be done. To do this, you must press the 'Save' button at the bottom left.

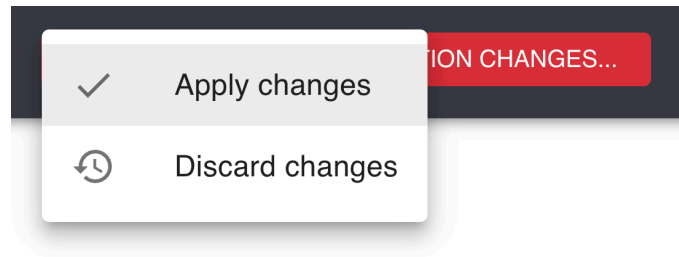
<input type="checkbox"/>	Scenes#Scene01	<input checked="" type="checkbox"/>	Integer (16b)	0 items	Integer (16b)	Holding register	242		
<input type="checkbox"/>	Scenes#Scene02	<input checked="" type="checkbox"/>	Integer (16b)	0 items	Integer (16b)	Holding register	243		
<input type="checkbox"/>	Scenes#Scene03	<input checked="" type="checkbox"/>	Integer (16b)	0 items	Integer (16b)	Holding register	244		

SHOW JSON (ADVANCED)

SAVE
BACK

This will save the changes, now you just have to confirm it on the red button above:



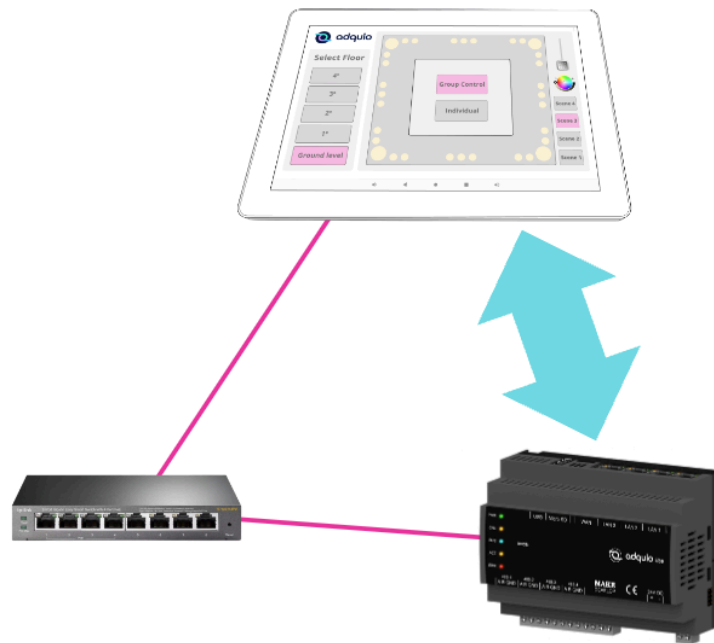


Press it and confirm the changes, and we go to the bottom left, where you will see this button, press it.



Copy all the content of the window that opens, you can right-click on it and select the 'Select all' option. Using your favorite plain text editor (for example, notepad), paste the copied text and save it with the name you like best and with a '.json' extension. You will use this file to import it into your SCADA in the next chapters. Now, we are done.

Working with Adquio screens



As we mentioned previously, once the data has been exported, there are many possible 'Consumers' of the data, in this document we will see two of them.

We start with the [control screens](#), all models of Adquio control screens, from the smallest of 10.1", to the largest of 32", communicate with Adquio using the Modbus TCP protocol.

In this chapter, we are going to see step by step how to configure an interface for our client, with the necessary controls, and how to connect it to Adquio so that they work together harmoniously.

Before starting, we must explain that Adquio screens work with Android, and that there are two different applications: one in which we will design the scenarios (which we will use), and another that simply visualizes them (for the end customer), and does not have the ability to modify them.

As you will see throughout this text, we will use the Android downloads folder as an exchange site between the editor and the "Viewer". In this way, by loading both programs in the same file, you can connect remotely and make changes very easily.


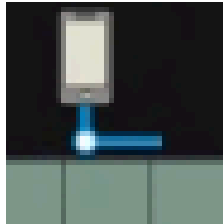
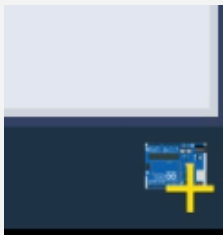
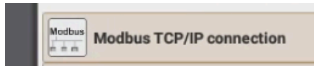



We recommend that you configure an email on the screen to be able to send files to it in the simplest way possible.

You should know that this editor allows you total freedom to create your own buttons and interface, simply with this, and designing a good background [you can create impressive control environments.](#)

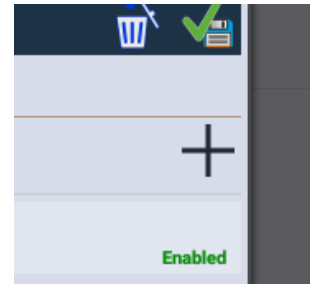
We will start by looking at the software that allows us to generate and edit the scenarios.

Set up the connection

Steps to follow	Description	Image
1	The Adquio Screen screens have the Virtuino 6 software pre-installed, locate this icon on your desktop and click on it.	
2	Look for this icon at the top left and tap it.	
3	Then in the lower right icon to add a new server.	
4	Press on Modbus TCP/IP connection.	
5	Now you must name the server, in the example we put "adquio" and the IP address of the Adquio to which it is going to connect. If it is on the local network (LAN) of an Adquio lite or pro, it will be 172.20.20.1, otherwise you will have to find out what IP it has in the System/General menu option of your Adquio.	

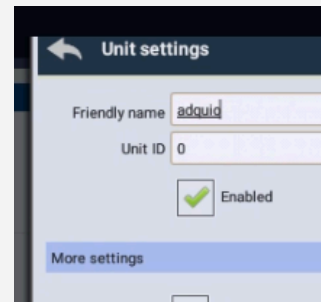
6

Click on the 'Modules' tab and create a new module by clicking on the large + symbol on the right.



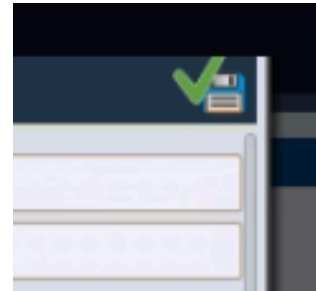
7

Give it a name and the address of the Modbus device you are connecting to. You must enter the same one that you configured in Adquio when configuring the Modbus server.



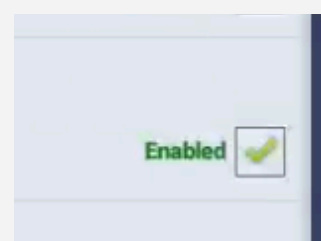
8

Then press the top right icon to save.



9

Click on the box on the right to put it 'Enabled'.


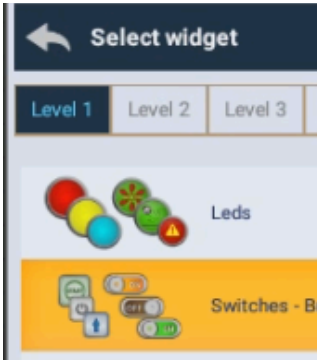
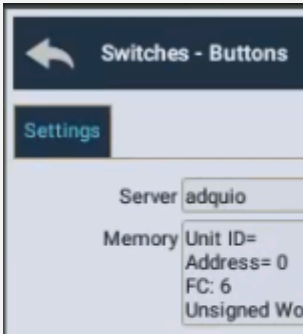


We have finished the connection configuration.

Creating an Interface

With the screen already connected to Adquio, we can start:

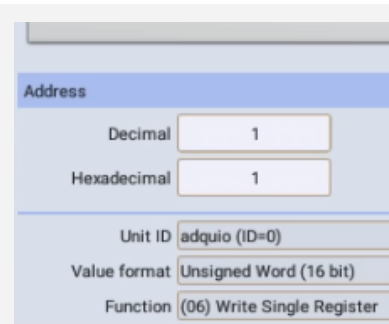
How to create objects

Steps to follow	Description	Image
1	We continue working on the same program from the previous chapter. Let's start by creating a switch, to do this simply click on the + box at the bottom right.	
2	And select Switches - Buttons	
3	In Server select "adquio", or the name you have given it in your case.	

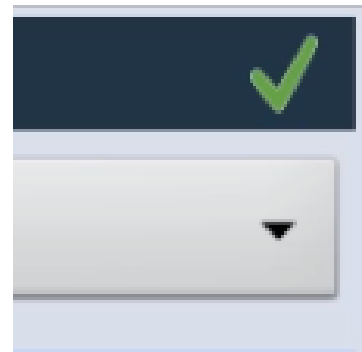
4 Click on the “Memory” box



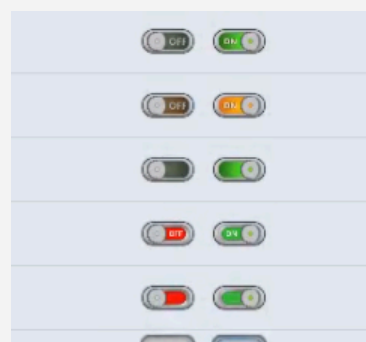
5 A small wizard opens to help you configure the address where it should read and write. You must select the address you want, just as you have configured it on your Modbus server in Adquio.



6 Always, when leaving each menu it is important that you press the upper right option to save.



7 You can choose the switch that best suits you.



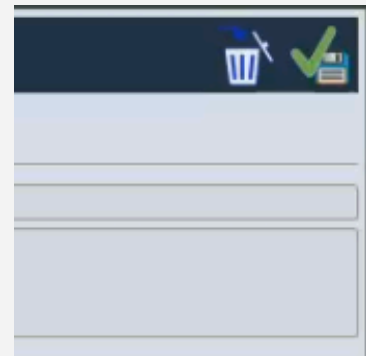
8

Although you can later move the component by clicking on it and dragging, here you can indicate its size and position manually, which is very convenient if you want to align components precisely.



9

You would simply have to hit save.



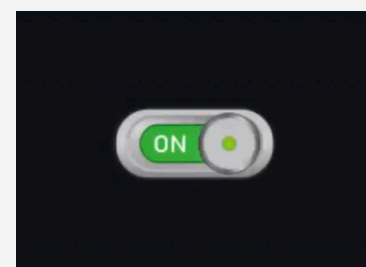
10

Press the green Check with the padlock, bottom right.

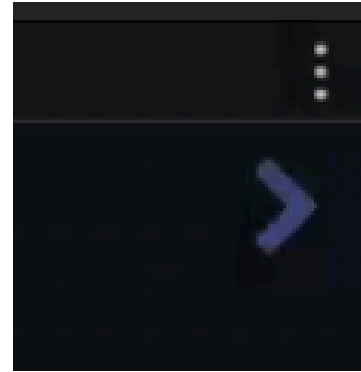


11

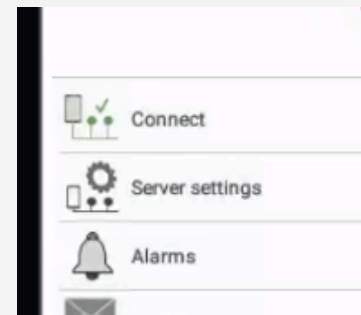
This will take you to "Preview" mode.



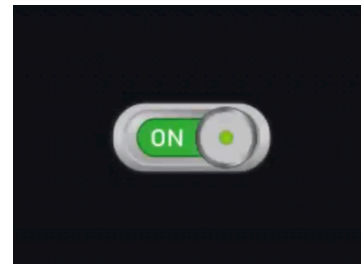
12 Press the 3 dots in the upper right corner...



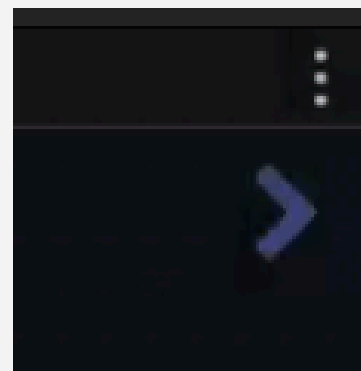
13 Press the “Connect” option and you will have everything connected with Adquio.



14 Now, you will see that pressing the button changes the variable in Adquio.

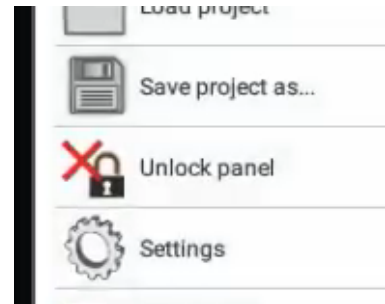


15 If you want to return to design mode, you must click on the three dots in the upper right corner.

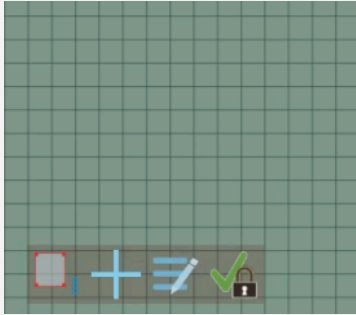
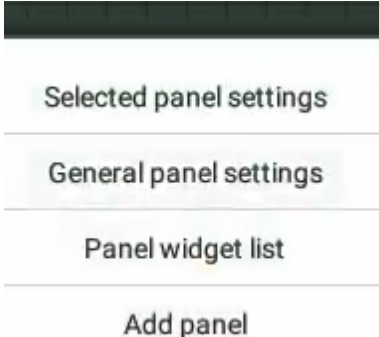
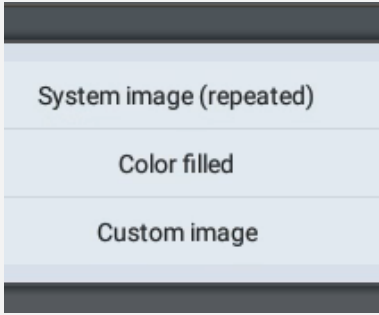
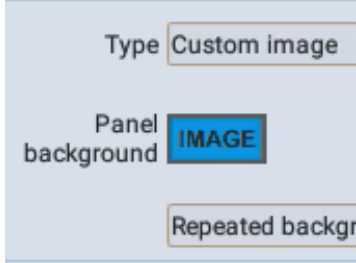


16

And click on the “Unlock panel” option.
With this you will return to the editor
where you can continue adding objects.
Go back to step 1.



How to add a background image

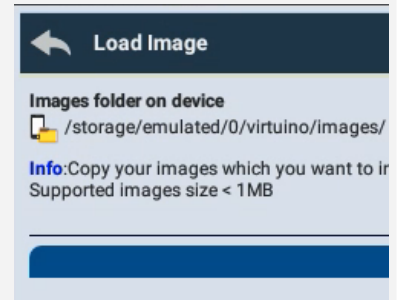
Steps to follow	Description	Image
1	To add a background image, you must click on the icon with a pencil on three horizontal lines in the lower right corner.	
2	Click on the “Selected panel settings” option	
3	Click on the “Type” drop-down menu, and select “Custom image”.	
4	Click on the Blue “IMAGE” button of “Panel background”.	

5

It tells you that the image you want to place in the background must be placed in the folder: **/storage/emulated/0/virtuino/images/**

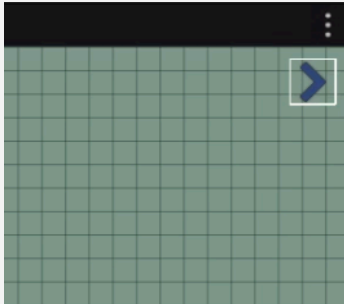
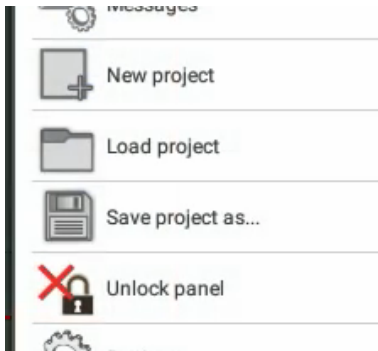
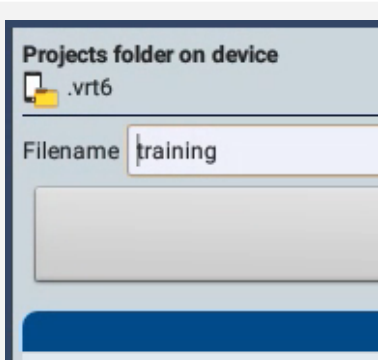
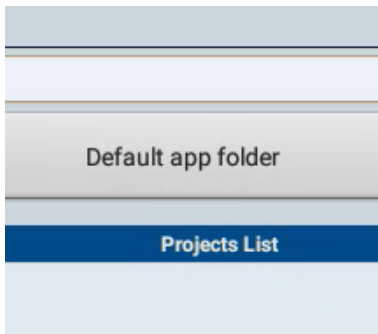
You must send the image that you want to place in the background via E-mail to the screen, and then using the file manager, copy it to this folder.

Once this is done, you can use it on this screen by returning to step 1.



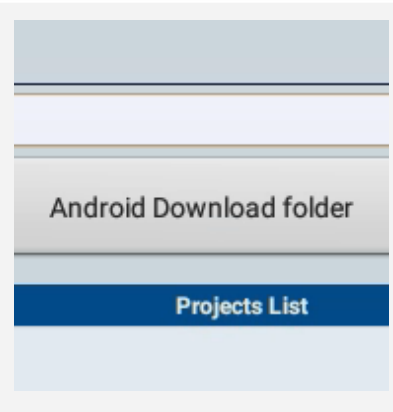
How to export your HMI

In order to use the HMI that you have created, you must make it reach the viewer program, to do this you must export it, let's see how to do it.

Steps to follow	Description	Image
1	To save or export your HMI you must go to the three dots at the top right.	
2	A menu will open, click on the option "Save project as..."	
3	Indicates the name for the file, in this example we have put "training"	
4	Click on the large button below that indicates "Default app folder"	

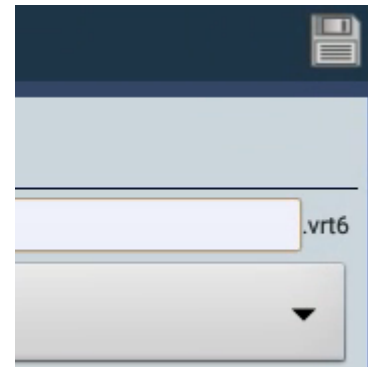
5

And select "Android download folder"





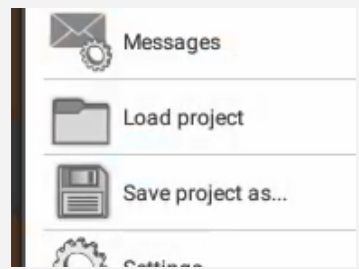
6

And to confirm, click on the save option at the top right, with this you already have the HMI saved in an easily accessible folder.

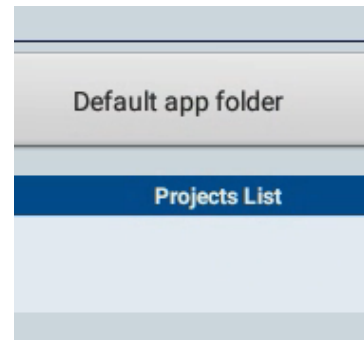


How to import your HMI

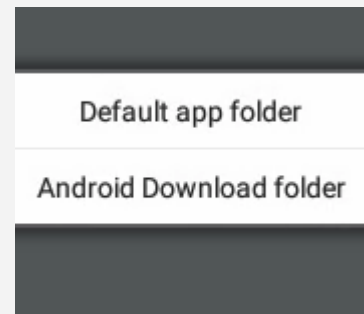
You have finished the HMI design for your client, and have saved it. We will now proceed to import it into the APP that will view it and allow your end user to use it, but will not allow them to modify it. Let's see the steps to follow:

Steps to follow	Description	Image
1	Look on your desktop or in the application drawer for the Virtuino 6 viewer application, and click on it to open it.	
2	Click on the three dots at the top right	
3	Select the "Load project" option	

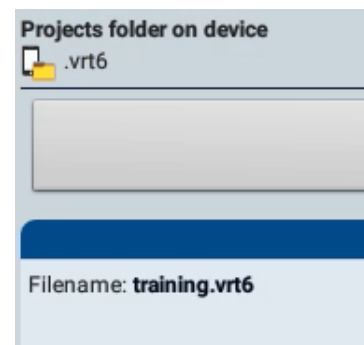
4 Click on “Default app folder”



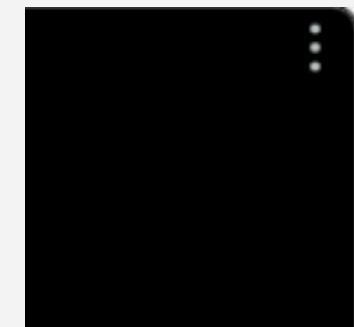
5 And change it to “Android Download folder”



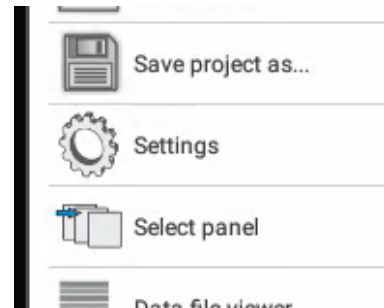
6 Now the file that you saved in the previous chapter appears in the list. Click on it, in our example “training.vrt6”.



7 Now you are going to configure it so that when it starts up it automatically connects to its data source (Adquio), to do this click again on the three dots in the upper right corner.



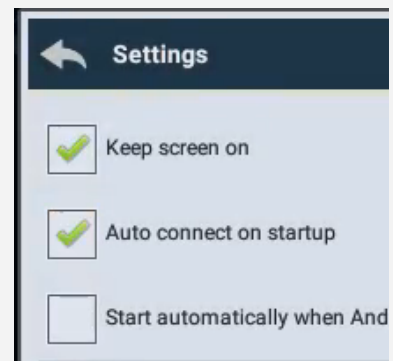
8 Click on “Settings”



9

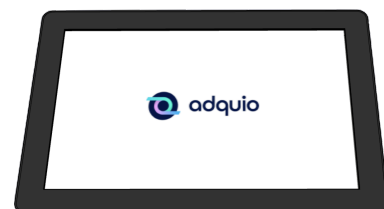
Make sure all three options are checked.

- The first keeps the screen always on.
- The second connects automatically when starting.
- The third starts this app automatically when Android starts, but the latter in some versions of Android does not work correctly, so we will indicate later how to solve it.



10


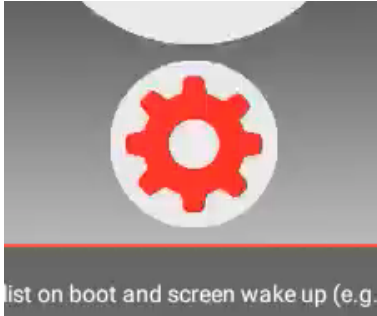
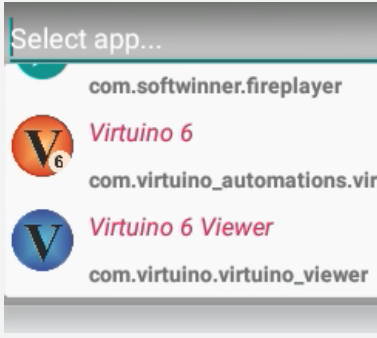
You have finished. Adquio Screen is ready to install on your client and have it working continuously for years.



We recommend that you only leave this App (Virtuino 6 viewer) on the desktop, to discourage the client from possible problems due to improper use.

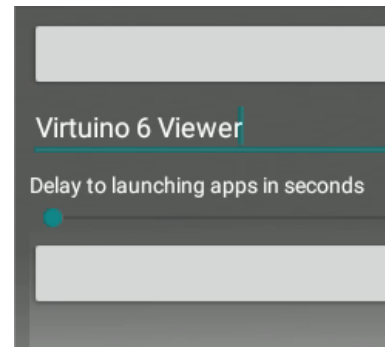
HMI auto start

By default, the display App has the option to “Start automatically when Android finishes starting”, but this option does not work correctly in all versions of Android. So, if this happens in your case, we have a solution planned, and We explain it to you below:

Steps to follow	Description	Image
1	To force the automatic start of our HMI display application, we will use the App: Boot Control, which is installed by default on your Adquio Screen.	
2	Once started, click on the gear button	
3	Click on “Select app...”, scroll down until you find “Virtuino 6 viewer” and select it.	

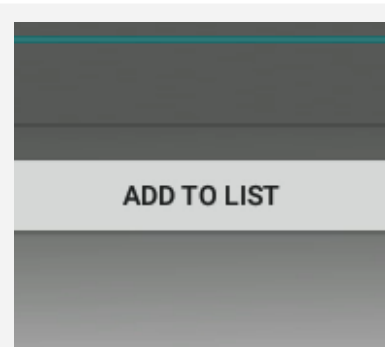
4

Now you must select what delay you want for the start of the app after Android starts, you can leave it at 0 if you are going to start only one App as is this case.



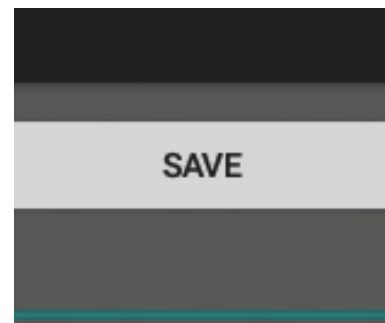
5

Click on the "ADD TO LIST" button.



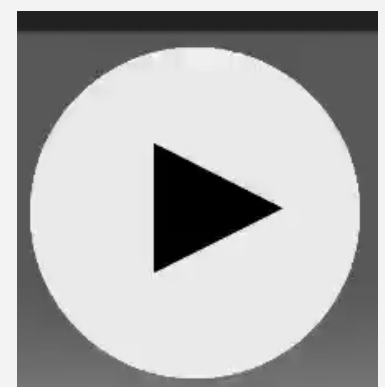
6

Click on the "SAVE" button

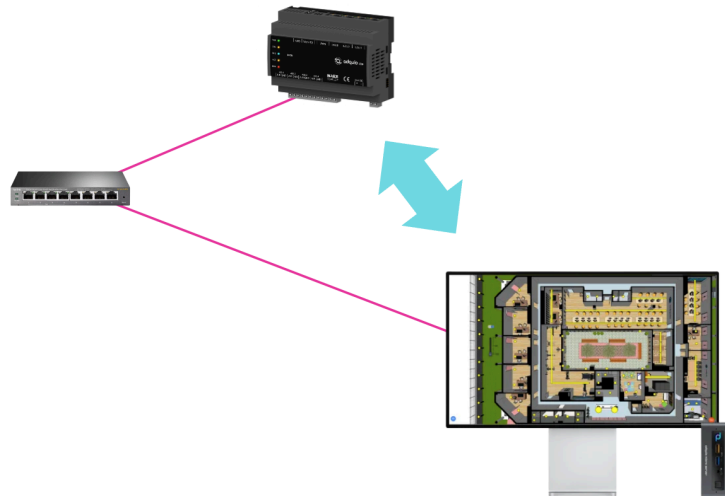


7

Click on the big play button and this will launch the service, you are done.



Converting variables for SCADA



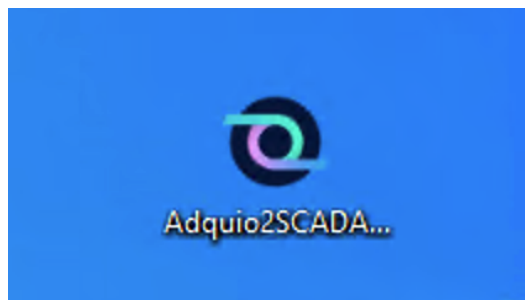
We have already seen how to use the Adquio screens with the controllers. Now we will see how to send all the variables exported in Adquio to your Adquio [SCADA](#), and this is valid for both the Local version and the Cloud version.

Adquio's SCADA solutions allow you to have local and remote control of all your Casambi installations, and as you have already seen, regardless of the number of networks or their geographical location.

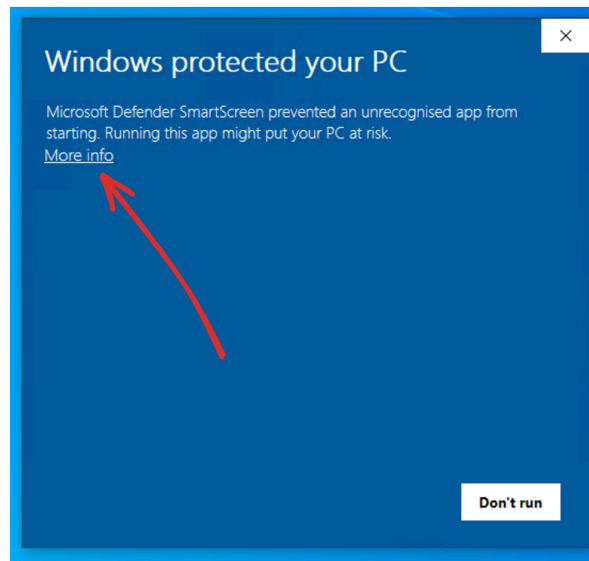
At Adquio we have created another specific program so that the data transition between our controllers and our SCADA is completely automatic and fast for you.

This program is now available for Windows, and in the coming weeks it will also be available for Mac and GNU/Linux.

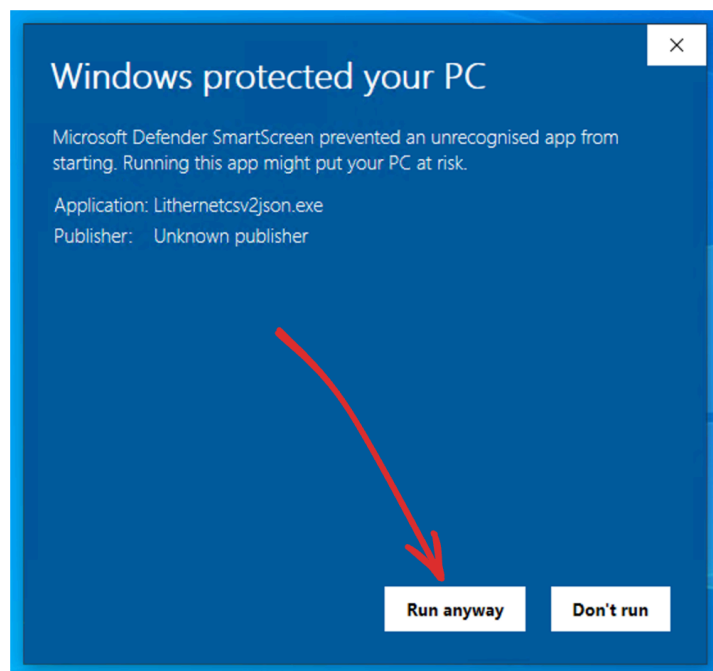
In its version for Windows, you can download it from [here](#). It is a compressed executable, download it and unzip it. With this you will get this icon in the same folder, in our case on the desktop:



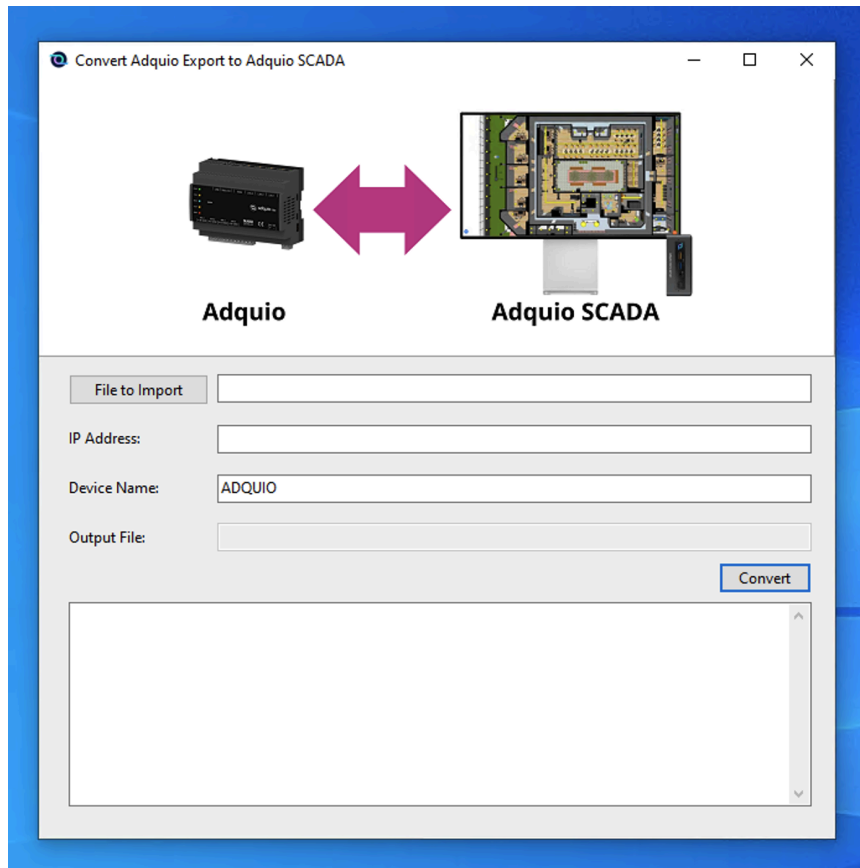
By double-clicking on it, you will get this Windows security message:



We will click on more information and...



We press the 'Run anyway' button, and finally we have our program ready to run.



Let's look at each of its fields to know how it works. This program will convert the file that we have previously exported in Adquio (see chapter: [Exporting variables in Adquio](#)) in a format directly compatible with your Adquio SCADA.

- **File to import:** You must select the file that you had exported in Adquio in JSON format.
- **IP Address:** Here you must specify the IP address of your Adquio controller, from where your SCADA will take all the information to function. If you don't know it, you can consult it in the system section of your Adquio menu in the Network option.
- **Device Name:** It is the name that you will see within your SCADA in the data source. If you are going to work with more than one, it is a good idea to place a name here with a counter, such as: Adquio01, Adquio02, etc. It should not have spaces or special symbols.
- **Output File:** This value will be generated automatically when you fill out the 'File to import' field. The name of the output file will be generated automatically, in the same folder from which you took the input file.

When you have finished covering this data, you just have to press the 'Convert' button, and you will be done. You will see that the output file has been generated in the folder that the program indicates. We will use this file in the next chapter to send all the information to Adquio SCADA.

Importing and using variables in Adquio SCADA Server/Cloud



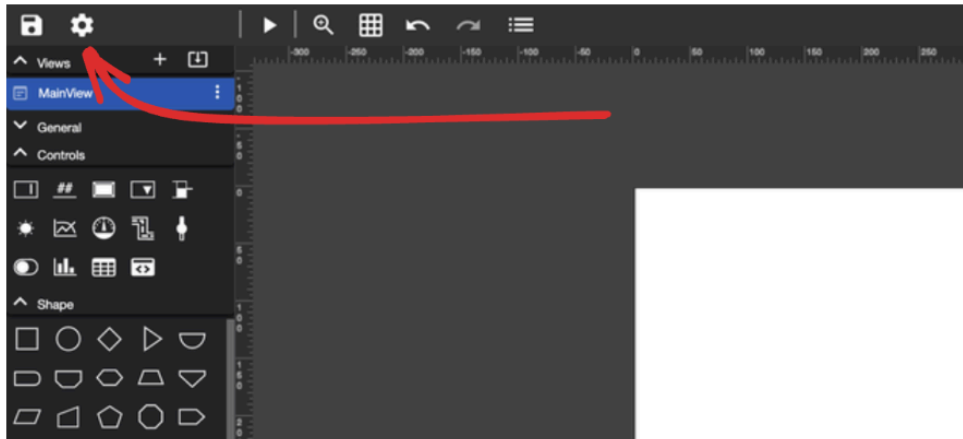
Importation process

In this example, we have only worked with 3 Casambi scenes, but the entire process is the same for any number of Casambi scenes and networks.

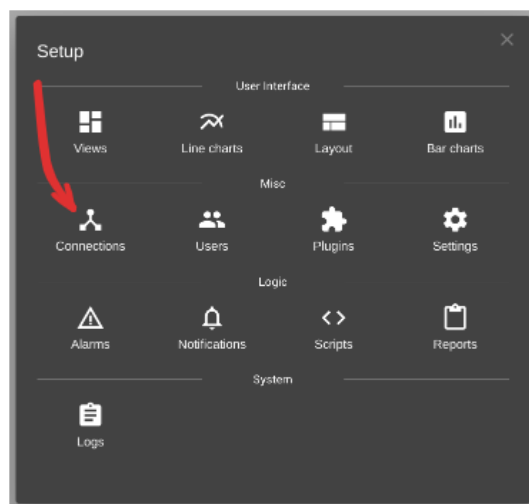


We are going to see the import process on an Adquio SCADA Server, but if you are working on an Adquio SCADA Cloud, the steps are exactly the same, since we always work from a browser.

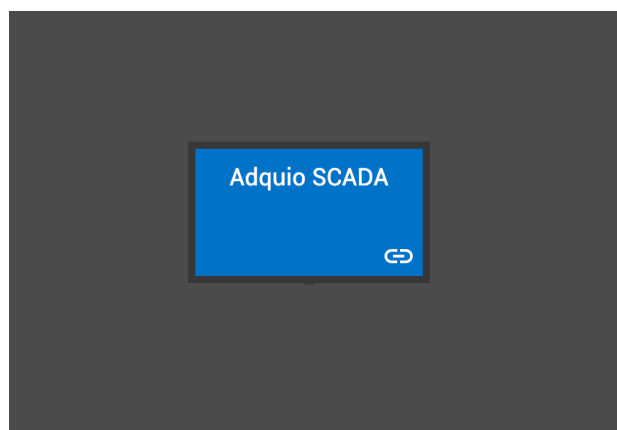
We assume that you are already connected to your Adquio SCADA Server, therefore, you will have something like this:



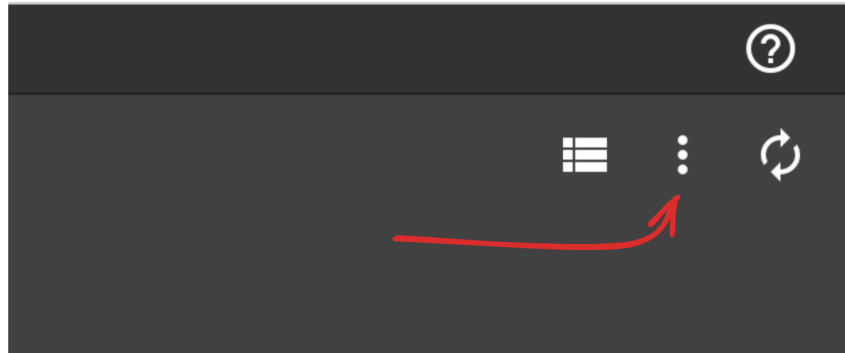
To import our newly created file, we will go to the icon that we indicated in the upper left.



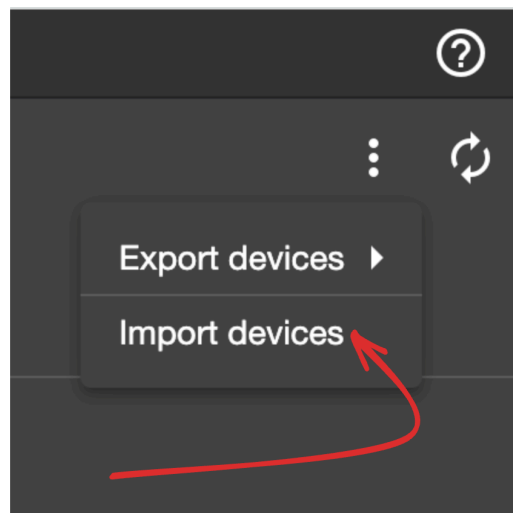
Next, we will go to the 'Connections' button to generate a new connection with our Adquio SCADA.



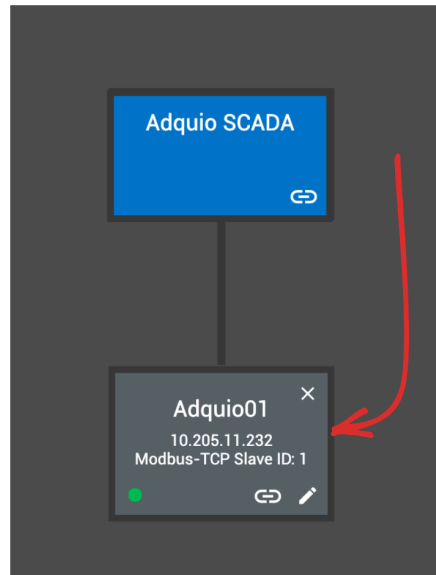
You will see a screen similar to this, at the moment we do not have any connection, we are going to import our device, and you will see that it appears here.



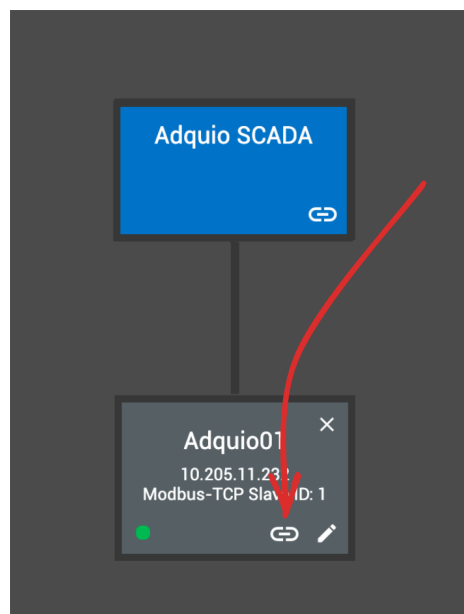
To do this, you will go to the button that you see at the top right and press it.



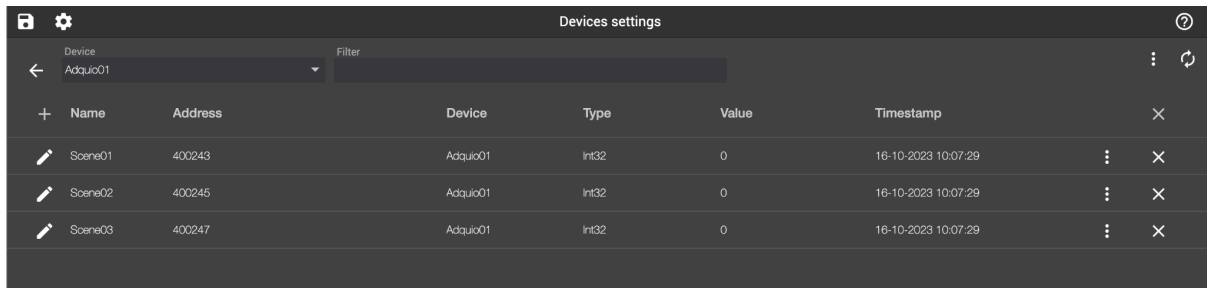
With this, you will open this menu, in which you will click on 'Import device'. Now you must select the file that you had exported in the previous chapter with a .JSON extension.



When you do this, you will see that a new device appears in the window. This device corresponds exactly to the one we created in the previous chapter.



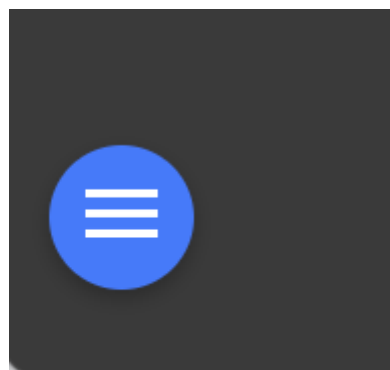
We go to see its variables, clicking on the clip button.



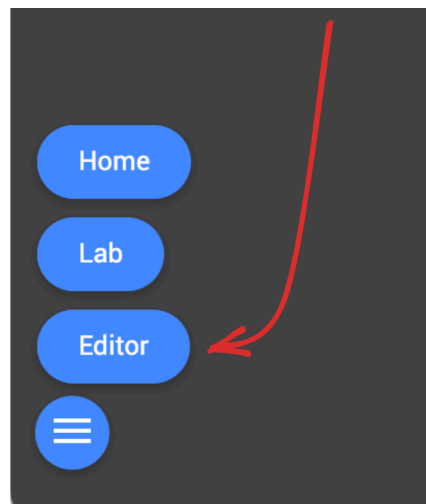
The screenshot shows a 'Devices settings' window with a dropdown menu set to 'Adquio01'. Below the dropdown is a table with columns: Name, Address, Device, Type, Value, and Timestamp. The table contains three rows of data for 'Scene01', 'Scene02', and 'Scene03', all with a value of 0 and a timestamp of 16-10-2023 10:07:29. Each row has a delete icon (X) and a menu icon (three dots) on the right.

+	Name	Address	Device	Type	Value	Timestamp		X
✎	Scene01	400243	Adquio01	Int32	0	16-10-2023 10:07:29	⋮	X
✎	Scene02	400245	Adquio01	Int32	0	16-10-2023 10:07:29	⋮	X
✎	Scene03	400247	Adquio01	Int32	0	16-10-2023 10:07:29	⋮	X

And you will see that, finally, it has brought us all the variables that we had exported in Adquio, without having to do any work manually. We can now begin to design an SCADA using these variables, to change the values in the Adquio controller from outside, and consequently in the Casambi network.

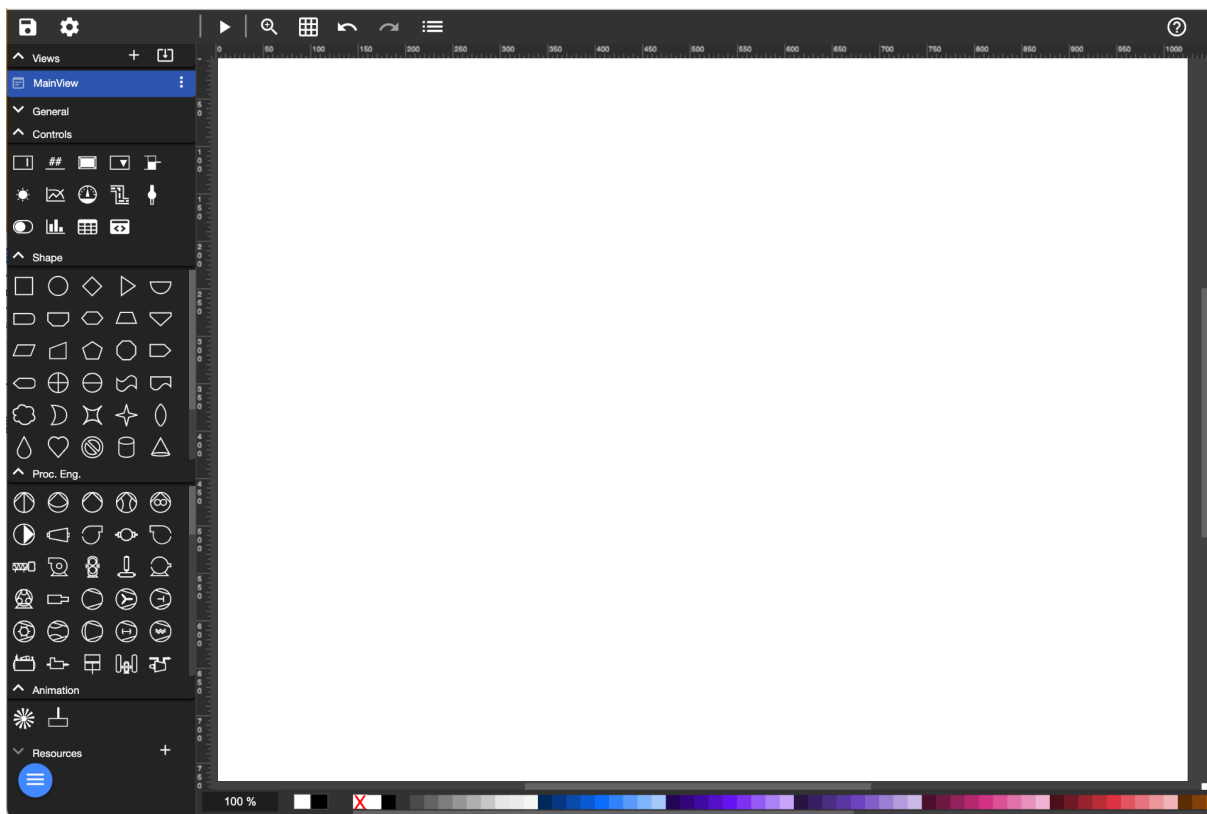


Now, to leave here, click on the blue button in the lower left corner. When you press it, this small menu appears.

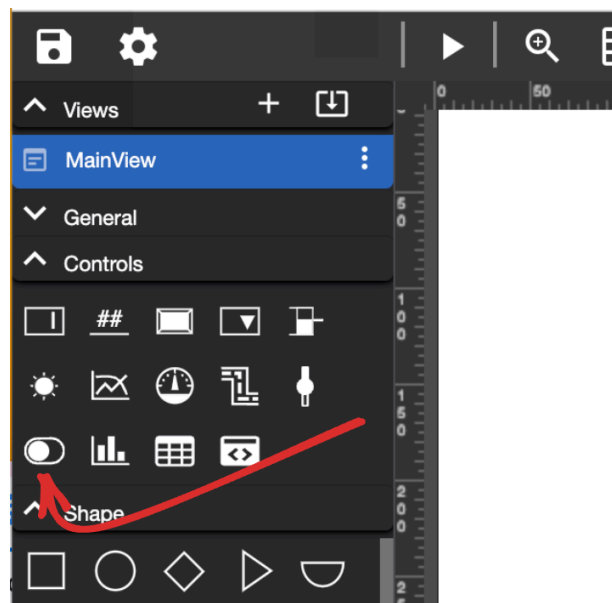


Click on 'Editor' and this will take you to the editor.

Using variables in SCADA



You will see that you have hundreds of options, in this document we are going to see how to connect three elements to the variables that we have brought from Casambi. With this, we hope to make your work easier when you have to do it yourself, in your client's SCADA.

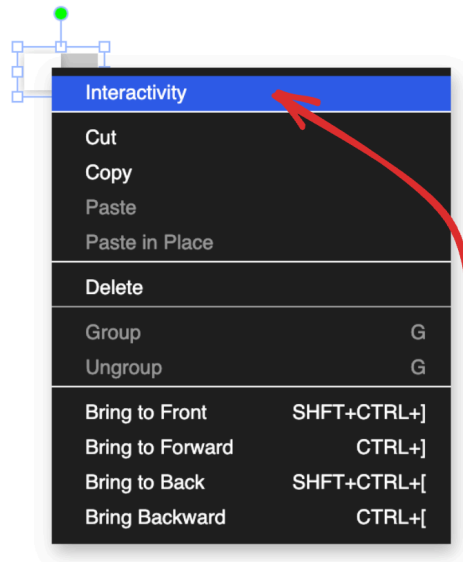


Perfect, let's start with the easiest thing, inserting a switch. To do this, click on the icon that we indicate. This will select it, then click on the work area (the large white part).

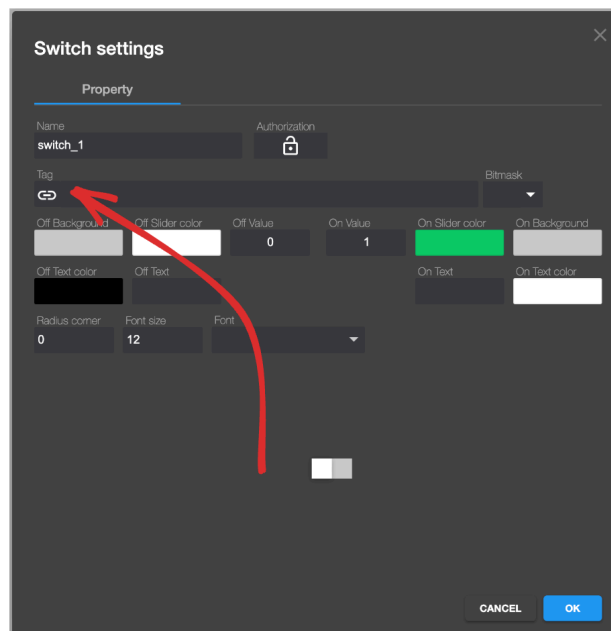
This will create, right in that position, a small switch.



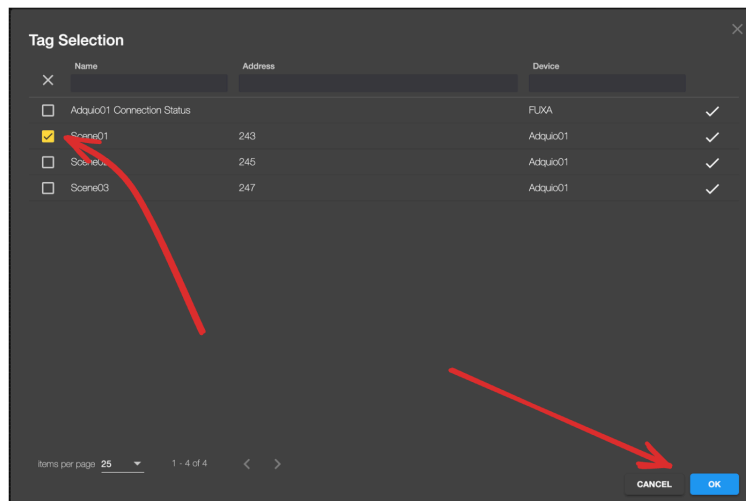
And push the right button over it; you will get this menu.



Click on 'Interactivity', this will open the following window

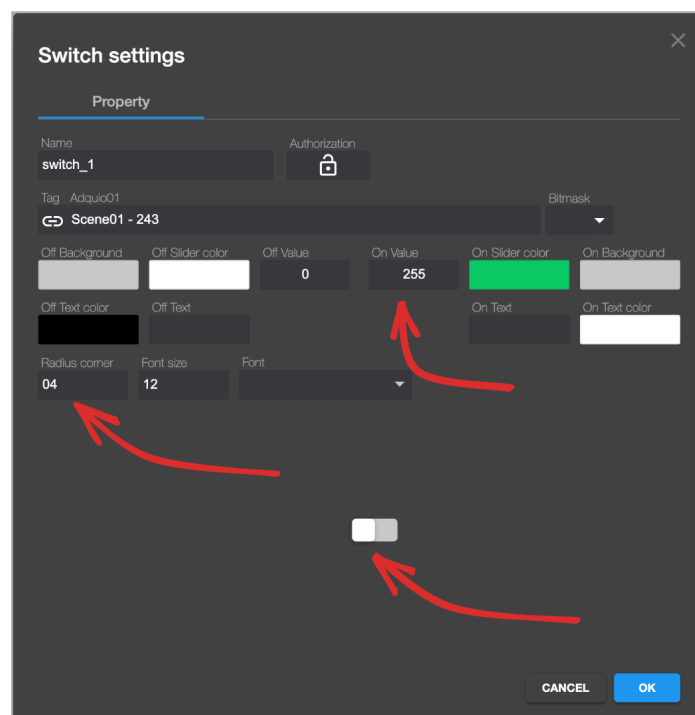


In this window we are going to touch some parameters, we will start with the 'Tag' field, which is what allows us to associate this object with one of our imported variables. Click on it to select.



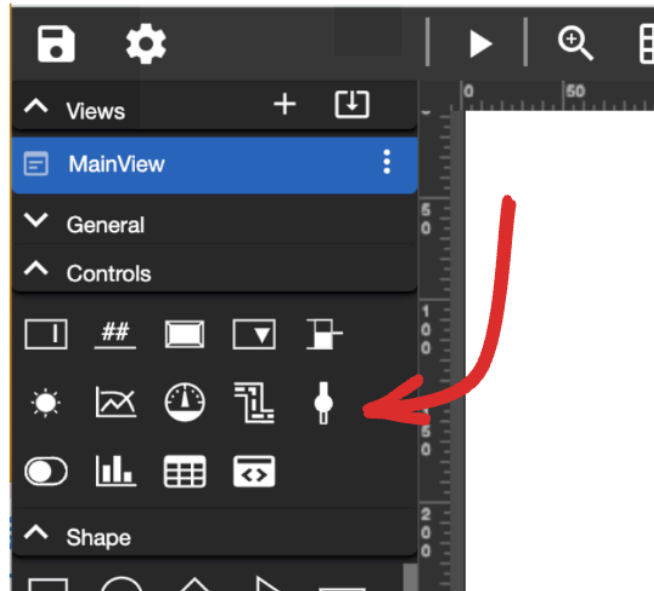
Let's select, for example, the first variable. To do this, simply click on the small checkbox to the left. Next, simply click on 'OK'.

With this, we now have the variable from scene 1 (remember that this contains the value of 'level') associated with our object.



A very important piece of information is the range of values that this object will manage, as it is a switch, it only has two values, on and off. In this case, the shutdown already corresponds to the default value, but the ignition must be modified, in this case when working with Lithernet, the maximum brightness value is 255, we set it to 'On value'.

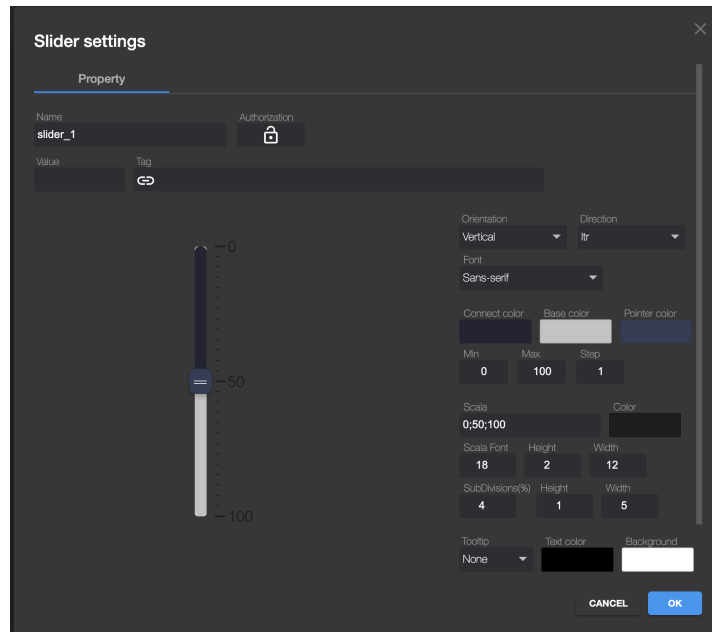
One more detail, we like rounded objects, so we can set the value 4 in the 'Radius corner' field, and we will see that now our switch has slightly rounded corners. We click on the 'OK' button, and you are done with this object. If you wish, you can move it to the position that best suits you.



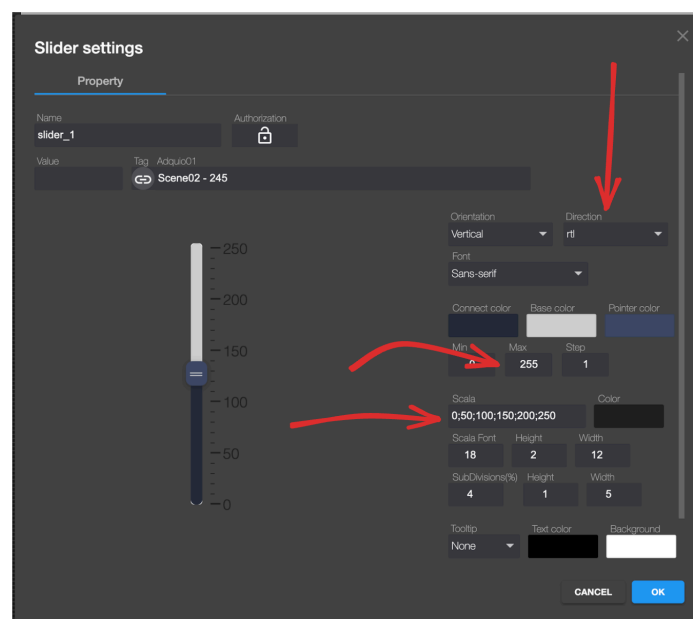
Now, we now go with a slider to control the regulation of another of our scenes. To do this, we will click on the linear regulator icon and again on the white area to locate it.



As in the previous case, we right-click on it, and we select 'Interactivity' to go to your properties.



As you can see, each object has different properties, adapted to its capabilities, therefore, in this case we will have some common things and some special things about this object. As in the previous case, we will have the 'Tag' field to associate with our variables, but let's focus on the differences.

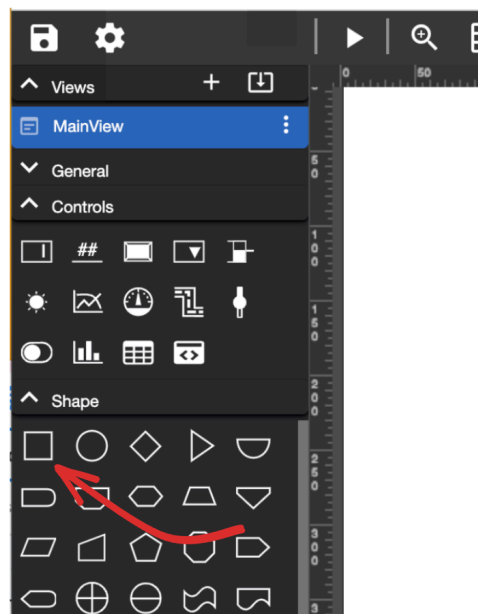


- **Direction:** It allows us to decide if the maximum of the current control will be up or down, we are going to change it to 'rtl' (Right to left) so that the values grow upwards.
- **Max:** As in the previous case we must indicate 255, the maximum for a luminaire in Lithernet is 255.
- **Scala:** Here we must write the scale that we want to see on the object, in this example we have set a scale of 50 by 50.

With this we now have our 'slider' ready to work, press 'OK'. You will now see your object in the design area.



Finally, we are going to create a luminaire to be able to see the response in our SCADA.

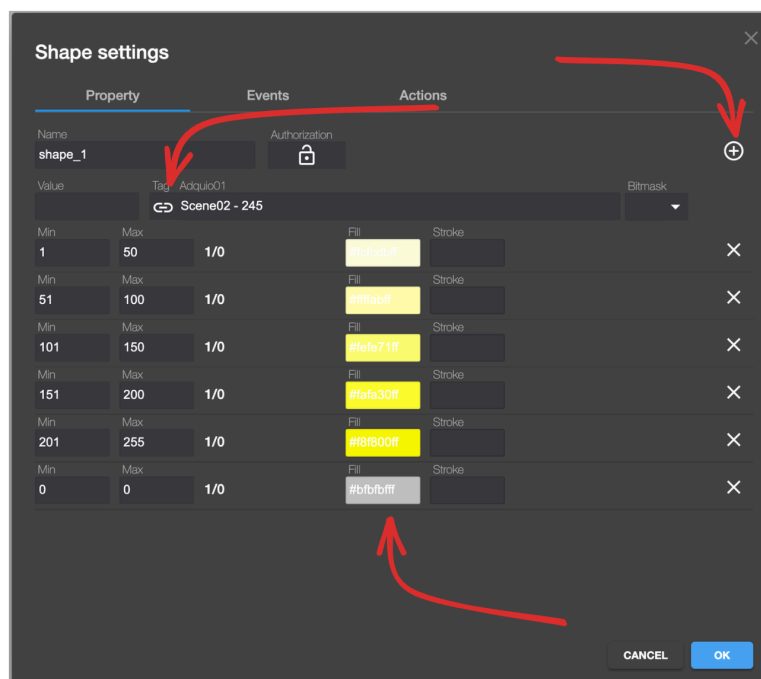


For this, we are going to use a simple polygon. We select the indicated icon and on the work area, click and drag, this will create a square.

Then, by clicking and stretching on one of its sides, you can create a rectangle similar to the one you see.

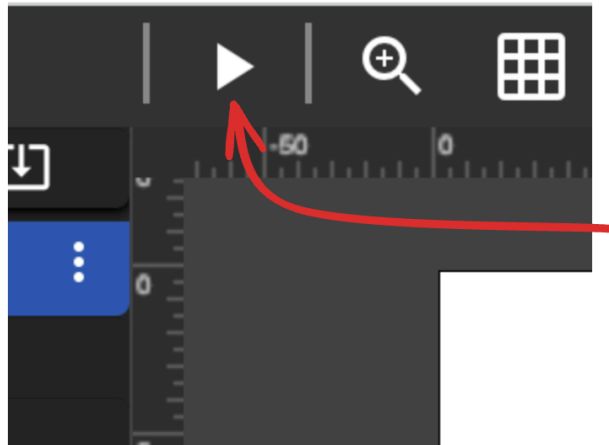


We do the same as always, right click on it, and go to 'Interactivity'.



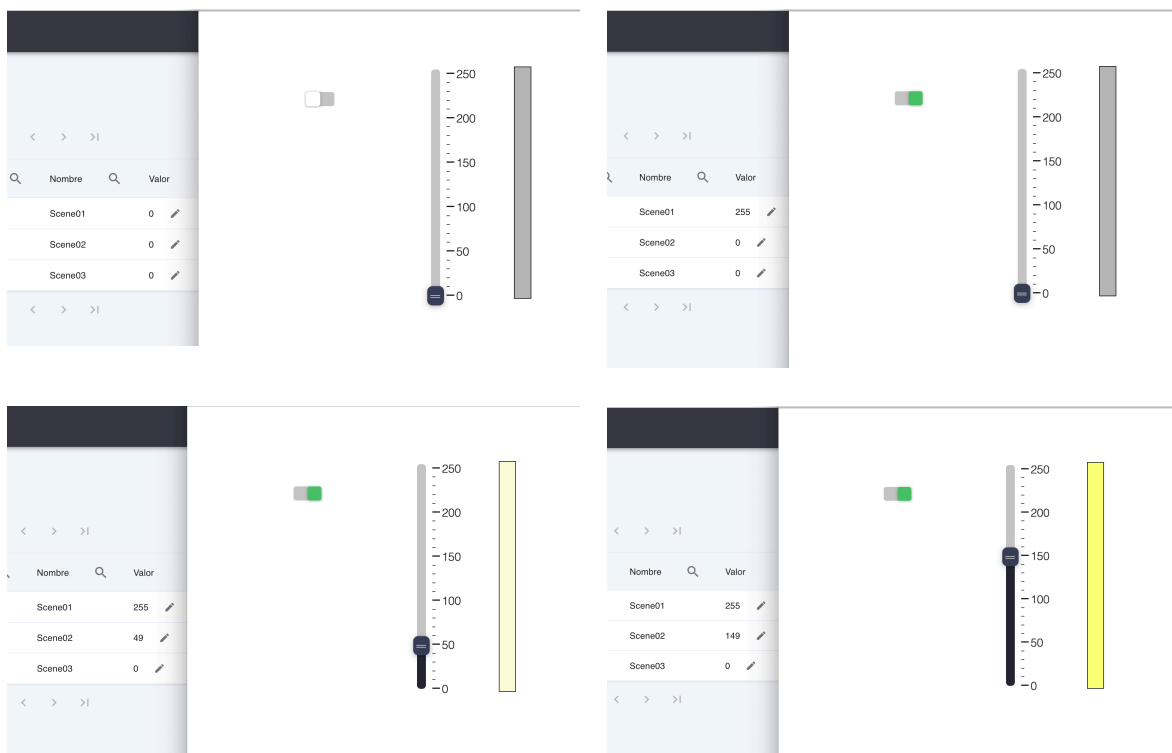
You already know how to select the 'Tag' in this case we are going to use the same one that we put in the previous object to be able to see its changes.

In the '+' symbol, you add colors to your object based on the value of the variable to which it is associated. If you press it several times you can add, as you see in the image, different colors that help you identify the regulation level. Press 'OK' to finish and save this object.



Now, in the main editor window, press the 'Play' symbol at the top left.

This will take you to your running SCADA, where you can finally test all the controls you just created and also see how the variables change in your Adquo and consequently in the Casambi network.



In these screenshots we have put the Adquo interface in a window and above it the window of our SCADA. We have started testing the objects, as you can see, it works perfectly and any changes you make in the SCADA are instantly reflected in your Adquo.

Very well, with this we finish this document, where we have given you all the information so that you can carry out all the steps, from connecting with the Casambi networks, to controlling them from an SCADA or control screen. We hope that all this information is very useful to you and that you find our products suitable for many of your installations.

Final conclusion

On this exciting system's integration journey, we've explored the intricate details of how to pair powerful Casambi networks with our versatile Adquio programmable controllers. Throughout this document, we've broken down in detail the steps needed to make data flow seamlessly to your control systems, whether local or remote, and equipped you with the tools necessary to meet any integration challenges.

We hope this document has served its purpose and provided you with a deep understanding of how to successfully integrate diverse systems. Now, you can look to the future with confidence, knowing that you have the ability to marry Casambi networks with Adquio control systems effectively.

We have focused our work on local integration, which is the most solid and safe option to achieve exceptional results. As you move forward on your integration journey, keep in mind that technology and best practices can evolve over time. Always keep an eye on the latest trends and advancements to ensure your integration remains efficient.

In summary, this document has been an invaluable tool for those seeking to master the integration of Casambi networks with Adquio control systems. You have acquired the knowledge necessary to face challenges with confidence, and we are sure that your integration projects will be even more successful from now on.

So, as we close this chapter of our journey together, we encourage you to continue learning, experimenting, and continually improving. Systems integration is a constantly evolving field, and your ability to adapt and grow will make the difference in your future projects.

Thank you for joining us on this journey of knowledge and learning. We wish you every success in your future integration projects!

Goodbye and until the next technological adventure!

Copyright © 2024 Neuronal Innovation Control S.L. All rights reserved. No part of this document may be reproduced, distributed or transmitted in any form or by any means, including hard copy, recording or electronic media, without the prior written permission of the copyright holder, except in the case of brief quotations and with proper attribution. Failure to comply with this restriction is subject to penalties provided by law. This document is provided for informational purposes only and does not constitute professional advice.