

# FactoryTalk Optix First steps

## Contents

1.	Using web Help .....	4
2.	Using integrated help.....	4
3.	Configuring Tags from PLC access.....	5
4.	Displaying a PLC Tag .....	9
5.	Web access .....	13
6.	Navigation Panels .....	14
7.	PopUp Windows.....	20
7.1.	PopUp from Button.....	20
7.2.	PopUp from variable.....	26
7.3.	PopUp from PLC value.....	31
8.	Reusable Objects. Working with Alias.....	31
9.	Communications with Micro8XX PLC's .....	41
10.	Configuring an application as an MQTT client.....	45
11.	PLC data and MQTT .....	56
11.1.	Asynchronous publish.....	76
12.	OPC UA Client.....	79
13.	Writing to an SQL database.....	86
13.1.	Datalogger. Working with an embedded database .....	86
13.2.	Datalogger Trigger on Tag Transition .....	87
13.3.	Let's write to an external SQL database from Optix .....	87
14.	Creating a OPC UA to SQL converter .....	109
15.	Subscribing to an MQTT broker with user and password. TTN example .....	114
16.	Creating an Http REST API client .....	119
16.1.	GET request .....	119
16.2.	Installing InfluxDB on windows.....	130
16.3.	Inject your first data with node-red.....	132
16.4.	Injecting to InfluxDB from command line .....	134
16.5.	POST request to local InfluxDB .....	137
16.6.	POST request to remote InfluxDB.....	140
17.	Using Github to store and share your repositories on the cloud.....	141
18.	Pushing your local Optix projects to Github .....	152

18.1.	Using Optix .....	152
18.2.	initUsing Git.....	170
18.3.	Remote repository creation with FT Optix version 1.3 .....	180
19.	Sending Telegram messages.....	184
20.	Modbus TCP to OPC UA converter .....	185
21.	Modbus TCP to EtherNet/IP Gateway .....	185
22.	Modbus to MQTT data aggregator .....	186
23.	Creating, importing C# .Net libraries to your environment .....	187
23.1.	Design time libraries. Importing Modbus Tags from a CSV File.....	187
23.2.	Runtime libraries.....	196
23.3.	How to use .NET third party libraries in our projects.....	202
23.3.1.	Installing Nuget packages in Visual Studio Code.....	208
23.3.2.	With Visual Studio 2022 community edition .....	215
23.3.3.	Complete your MQTTnet application.....	220
24.	First steps with OptixPanel.....	222
25.	First steps with ASEM6300B PC and runtime .....	228
26.	NetLogic and C# tutorial .....	233
27.	Understanding compilation of NetLogic code.....	234
28.	Understanding Classes (Objects), methods and data .....	237
29.	Creating a Method .....	239
30.	Linking events with Methods without input parameters .....	243
31.	Linking events with Methods with input parameters .....	246
32.	Asynchronous tasks, creating callback functions .....	248
33.	Periodic Tasks.....	253
33.1.	Native Periodic Tasks .....	253
33.2.	Netlogic Periodic Tasks .....	257
34.	Log info to emulator output.....	258
35.	Accessing variables .....	258
36.	Accessing objects .....	265
37.	Using Owner.....	272
37.1.	Accessing object using owner.....	272
37.2.	Detecting change on object using Owner .....	275
38.	Accessing PLC Tags.....	277
38.1.	Method0 .....	278

38.2. Method1 .....	280
38.3. Method 2 .....	283
38.4. Writing to an array of PLC values .....	286
39. Object events .....	289
40. Open a cmd terminal to run a command or program.....	295
41. Socket TCP using System.Net.Sockets without external libraries .....	300
42. How to define NetLogic Parameters .....	333
43. Using third party dll libraries in FactoryTalk Optix.....	336
43.1. Using third party libraries in Visual Studio 2022 windows Forms Framework .....	336
43.2. Thirt party libraries (dll) Socket Server application with Optix .....	352
43.3. Thirt party libraries (dll) Socket Client application with Optix.....	363
43.3.1. Installing external references (dll libraries) into the project Visual Studio 2022 .....	368
43.3.2. Installing external references (dll libraries) into the project with Visual Studio code .....	376
44. Using NuGet libraries (HiveMQ .Net example).....	378
45. Modbus server with FTOptix .....	390
46. Native Modbus TCP client with FactoryTalk Optix .....	394
47. DotNet ModbusTCP client with C# and FactoryTalk Optix .....	395
48. Modifying Existing projects for page navigation .....	405
49. Generating a pdf file from C#.....	418
49.1. Using Foxit Pdf .....	420
50. Using BoilerDemo .....	421
51. Users in Optix.....	425
51.1. Creating users and Groups manually .....	425
51.2. Create a Login form .....	427
51.3. Writing the logged user to the PLC .....	429
51.4. Creating Groups in design time .....	433
51.5. Creating Users in Design time .....	440
51.6. User batch creation in design time (100 users) .....	447
51.7. Manage user rights to Panels in a Native way .....	447
51.8. Making Login Form invisible when a user is Logged in .....	455
51.9. Make Panels visible if user belongs to one of a set of Roles .....	461
51.10. User management with landing panel.....	468
51.11. Manage user rights to Panels on a NavigationPanel in C# .....	469
51.12. Registering logged user on a database .....	472

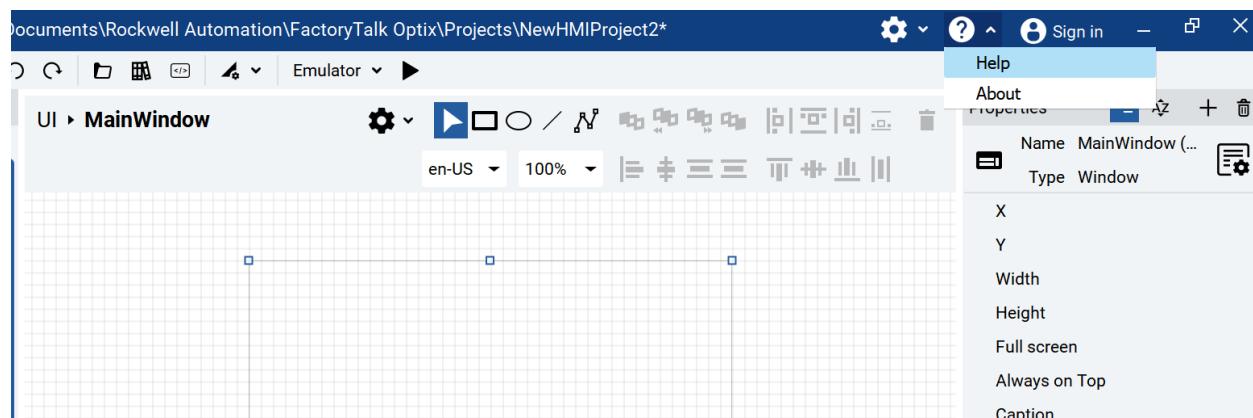
51.12.1. Adding the Logged user to a variable .....	472
51.13. Alphabetical order on the Users ComboBox .....	479

## First steps with FT Optix

### 1. Using web Help

<https://www.rockwellautomation.com/docs/en/factorytalk-optix/1-00/contents-ditamap.html>

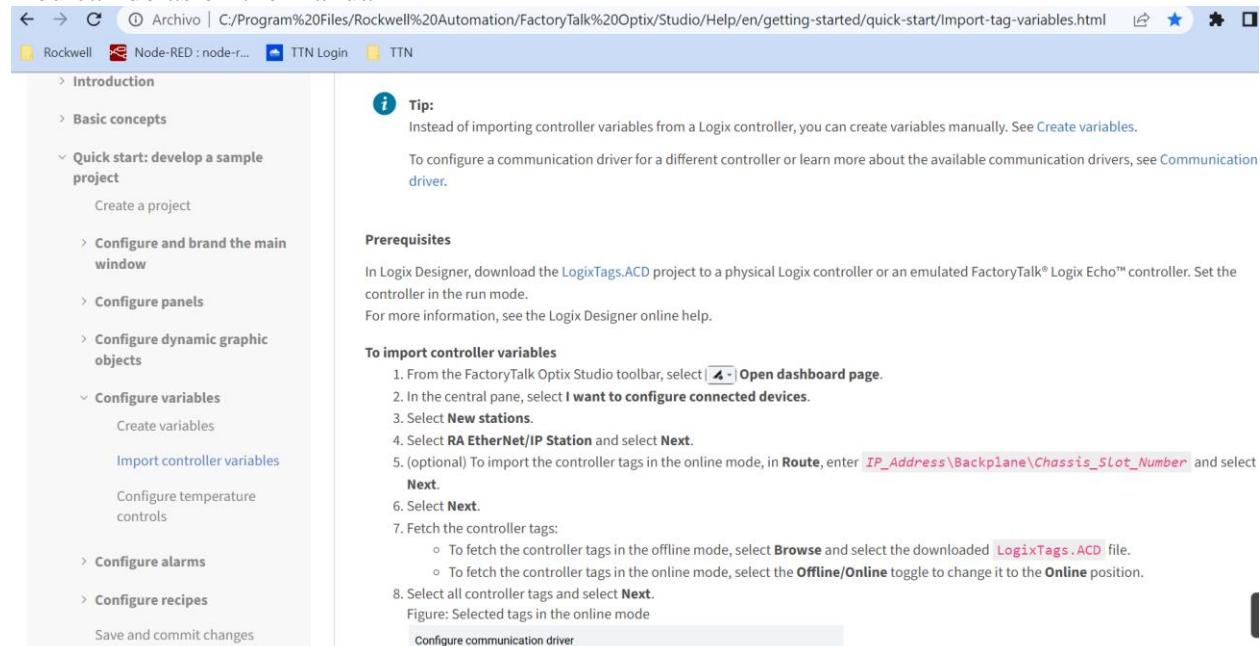
### 2. Using integrated help



<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/getting-started/quick-start/Configure-the-main-window.html>

### 3. Configuring Tags from PLC access

You can do as on the manual



The screenshot shows the FactoryTalk Optix Studio toolbar with several icons. The 'Import Tag Variables' icon (a document with a gear) is highlighted with a yellow box.

**Tip:**  
Instead of importing controller variables from a Logix controller, you can create variables manually. See [Create variables](#).  
To configure a communication driver for a different controller or learn more about the available communication drivers, see [Communication driver](#).

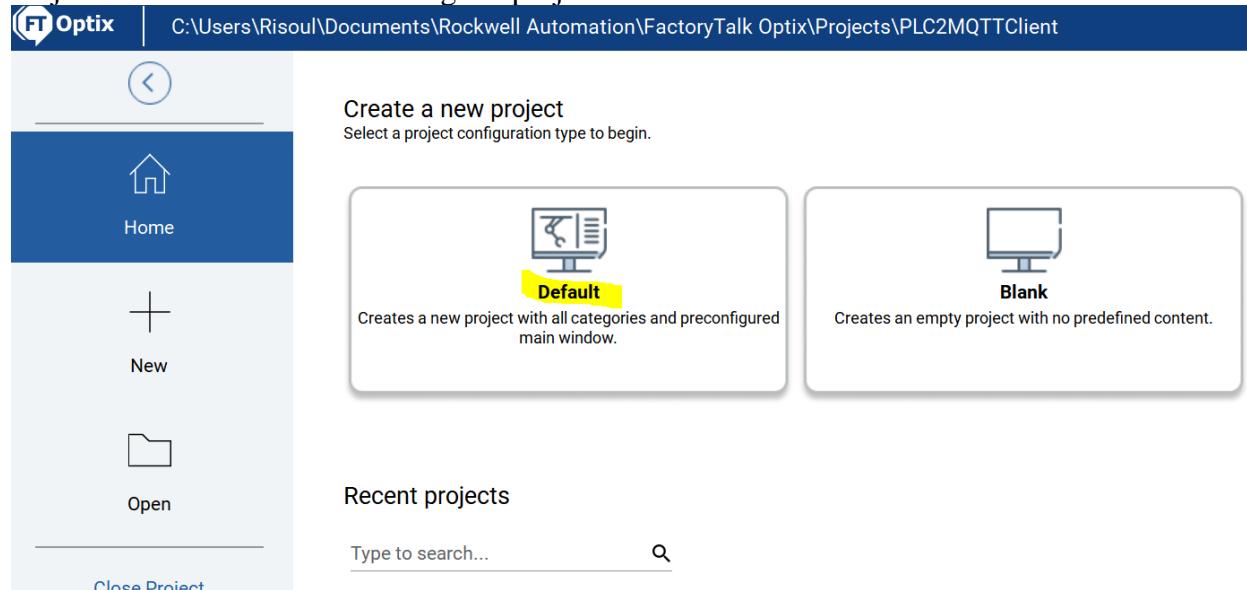
**Prerequisites**  
In Logix Designer, download the `LogixTags.ACD` project to a physical Logix controller or an emulated FactoryTalk® Logix Echo™ controller. Set the controller in the run mode.  
For more information, see the Logix Designer online help.

**To import controller variables**

- From the FactoryTalk Optix Studio toolbar, select **Import Tag Variables**.
- In the central pane, select **I want to configure connected devices**.
- Select **New stations**.
- Select **RA EtherNet/IP Station** and select **Next**.
- (optional) To import the controller tags in the online mode, in **Route**, enter `IP_Address\Backplane\Chassis_Slot_Number` and select **Next**.
- Select **Next**.
- Fetch the controller tags:
  - To fetch the controller tags in the offline mode, select **Browse** and select the downloaded `LogixTags.ACD` file.
  - To fetch the controller tags in the online mode, select the **Offline/Online** toggle to change it to the **Online** position.
- Select all controller tags and select **Next**.

Figure: Selected tags in the online mode  
Configure communication driver

Or just with the wizard on creating the project



The screenshot shows the 'Create a new project' wizard in FactoryTalk Optix Studio. The title bar says 'C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\PLC2MQTTClient'. The left sidebar has 'Home', 'New', and 'Open' buttons. The main area shows two options: 'Default' (selected) and 'Blank'. A search bar at the bottom says 'Type to search...'. The 'Recent projects' section is empty.

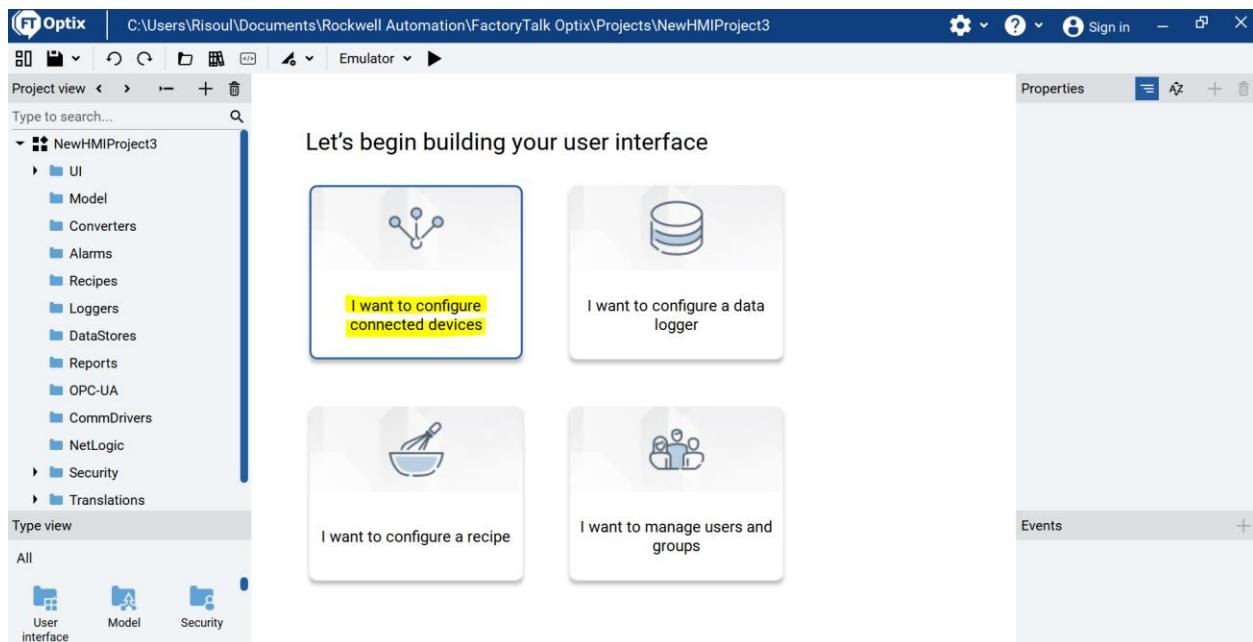
**Create a new project**  
Select a project configuration type to begin.

**Default**  
Creates a new project with all categories and preconfigured main window.

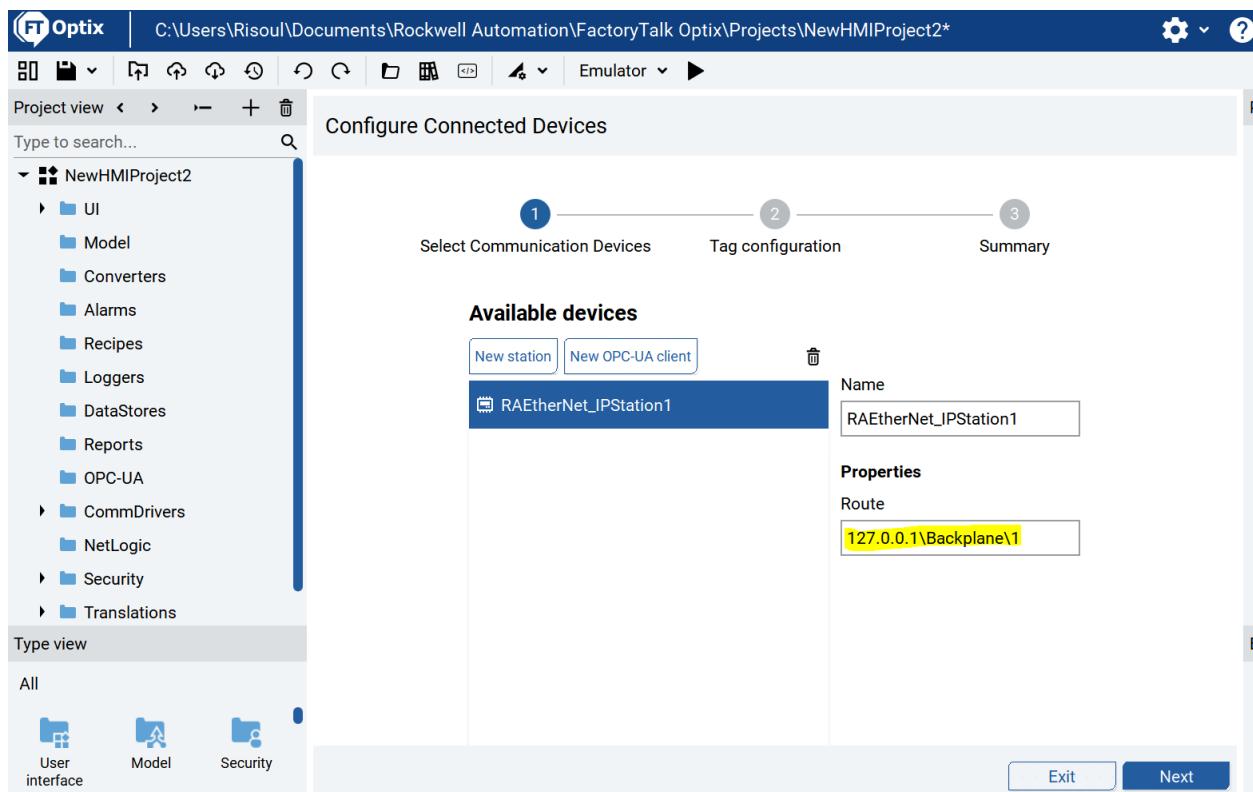
**Blank**  
Creates an empty project with no predefined content.

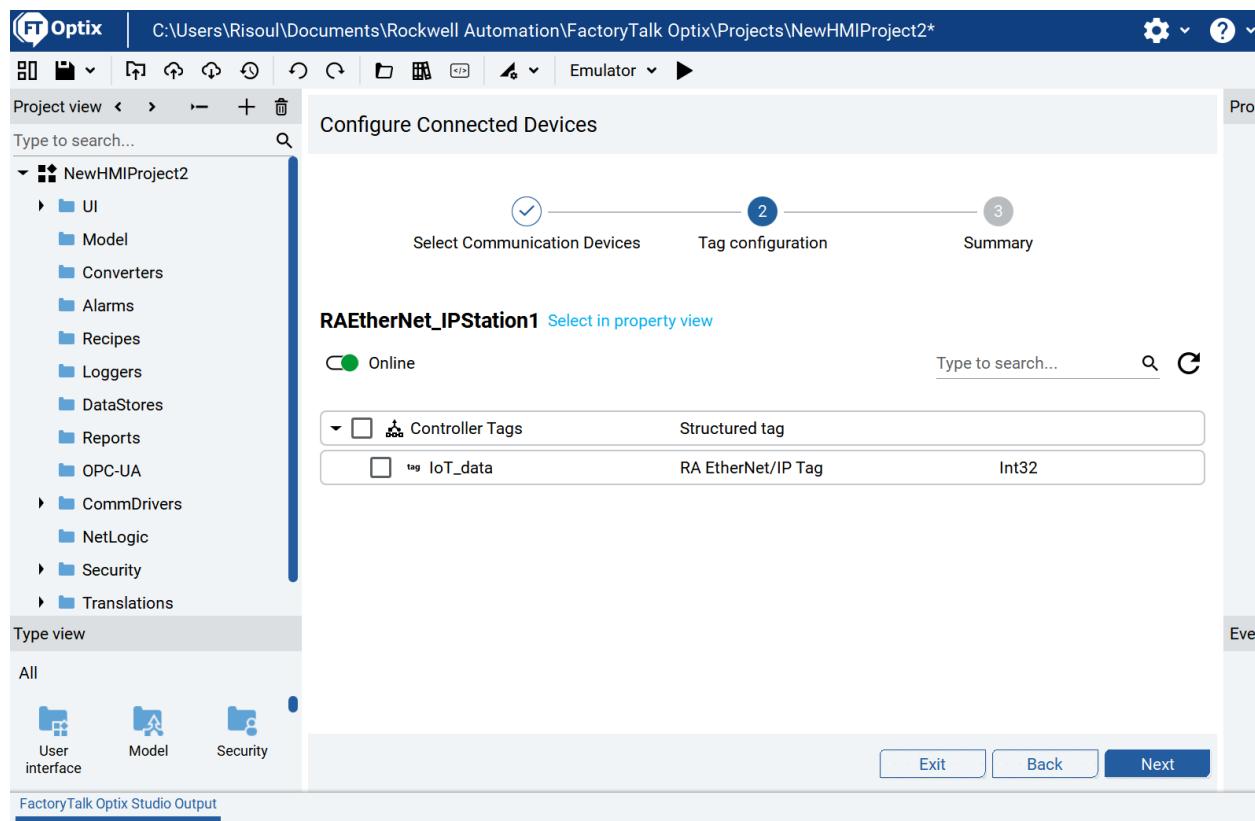
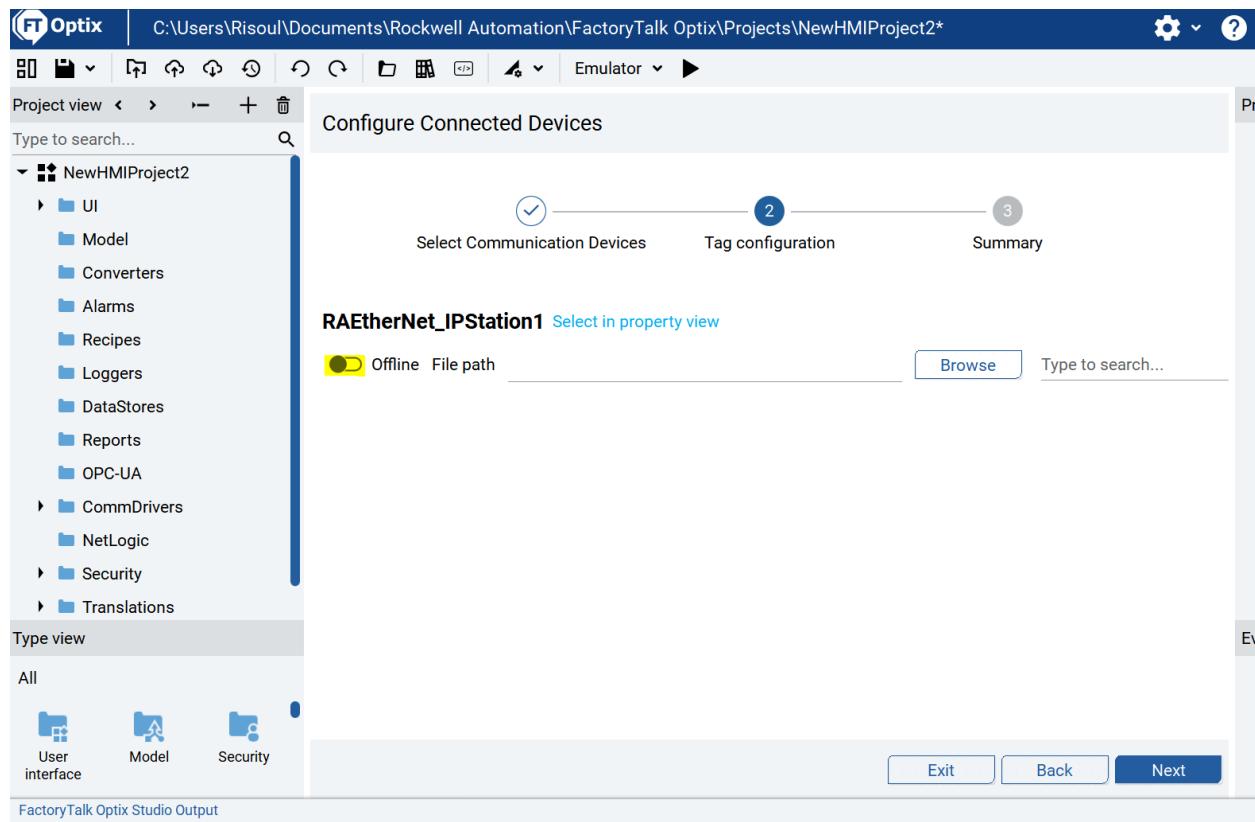
Recent projects

Type to search...



On both cases you will arrive to this point





## Configure Connected Devices

1 Select Communication Devices      2 Tag configuration      3 Summary

### RAEtherNet\_IPStation1 [Select in property view](#)

 Online

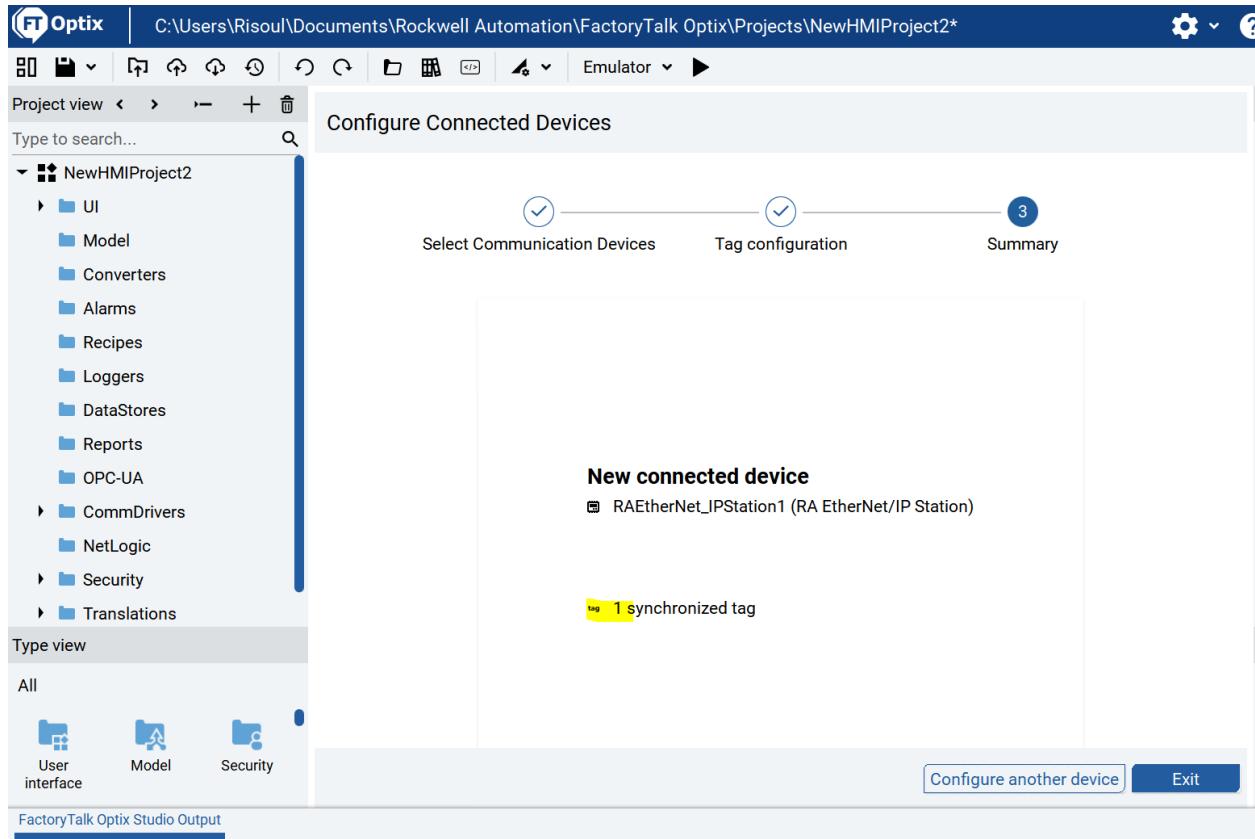
Type to search...  

<input checked="" type="checkbox"/>  Controller Tags	Structured tag
<input checked="" type="checkbox"/>  IoT_data	RA EtherNet/IP Tag

Exit

Back

Next



## 4. Displaying a PLC Tag

<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/getting-started/quick-start/Configure-the-main-window.html>

The screenshot shows a web browser displaying a help page titled "Configure the main window". The page is part of the "Getting started" section of the "FactoryTalk Optix Studio Help Center". It contains instructions for configuring the main window dimensions and a tip about screen resolution. The browser's address bar shows the full URL of the help page.

**Configure the main window**

The main window contains all graphical elements displayed at design time in FactoryTalk Optix Studio and at runtime in your FactoryTalk Optix Application.

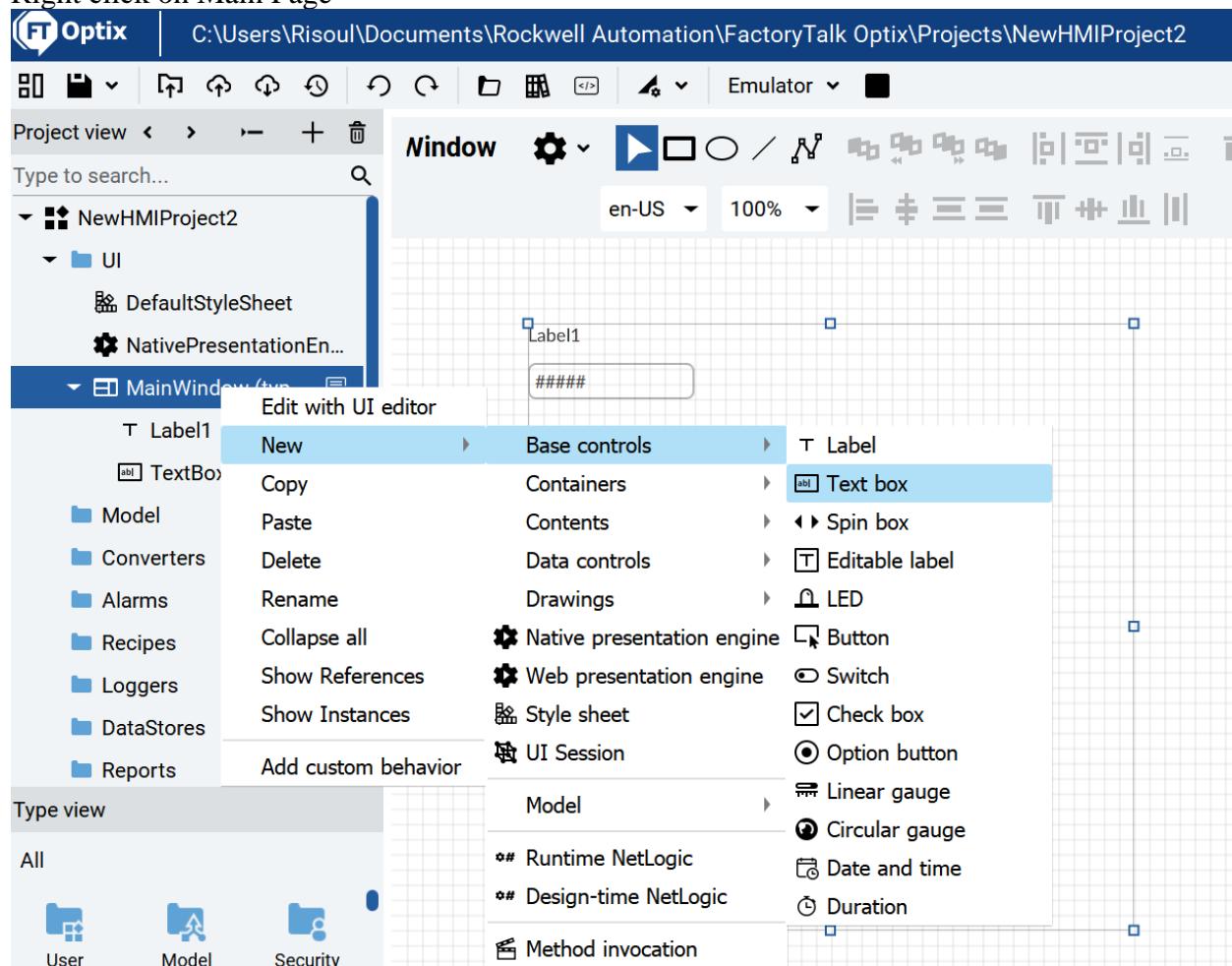
- In **Project view**, expand **UI** and double-click **MainWindow (type)**. The main window area displays in the editor. The main window does not contain any graphical elements.
- In **Properties**, set **Width** to **1080** and **Height** to **600**. The initial dimensions of the main window are now **1080 x 600**. At runtime, you can resize the main window at any time. For more information about sizing, see **Size conventions**.

**Tip:** To make the application fit your screen resolution, you can set different **Width** and **Height** values. Alternatively, you can set **Full screen** to **True**, however; it is easier to develop and preview applications with the default **False** settings.

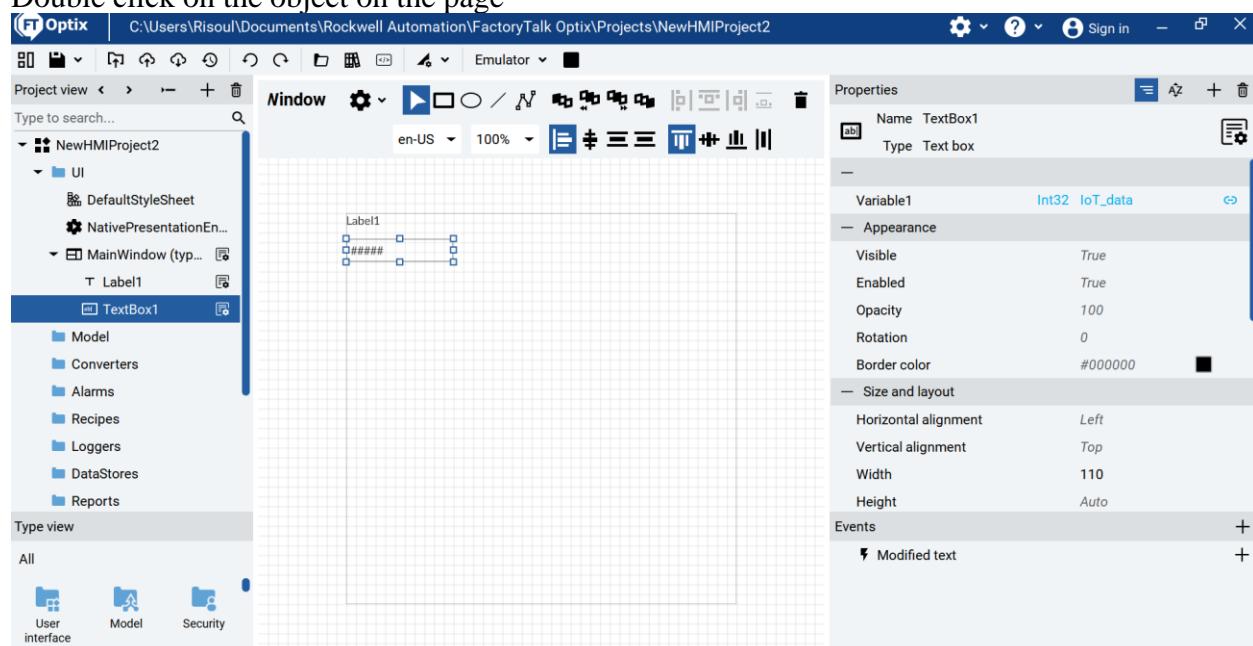
Figure: Blank resized main window in the editor

Let's prepare the page with some controls

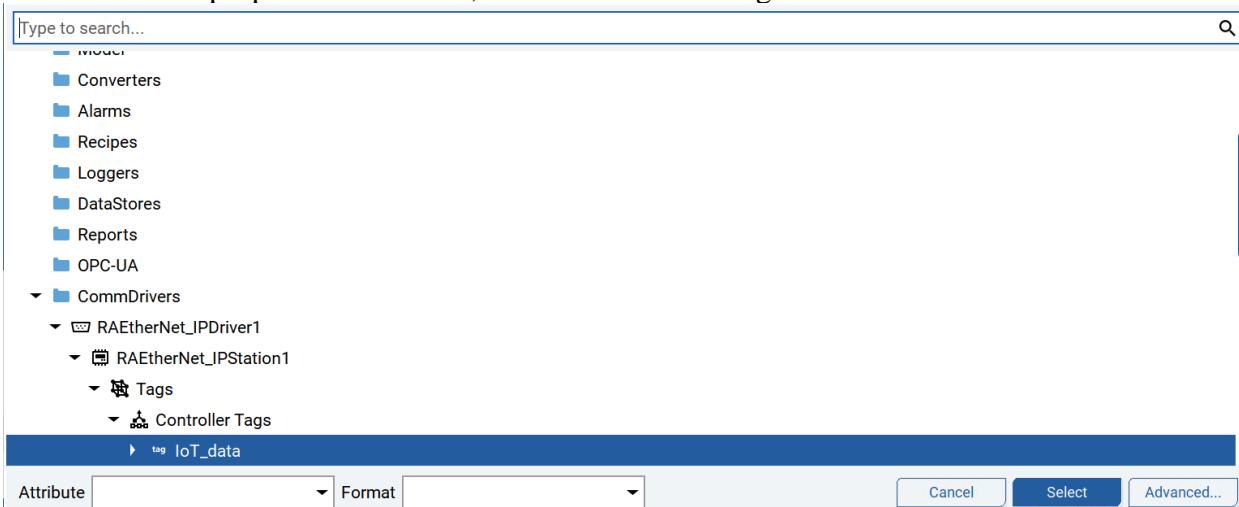
Right click on Main Page



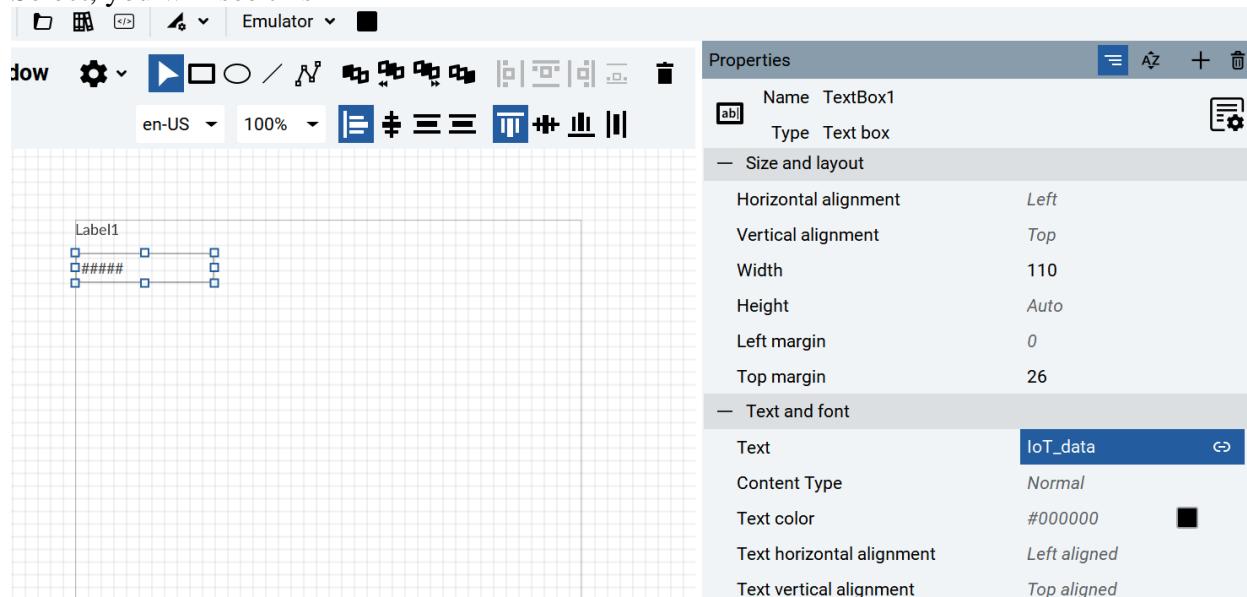
Double click on the object on the page



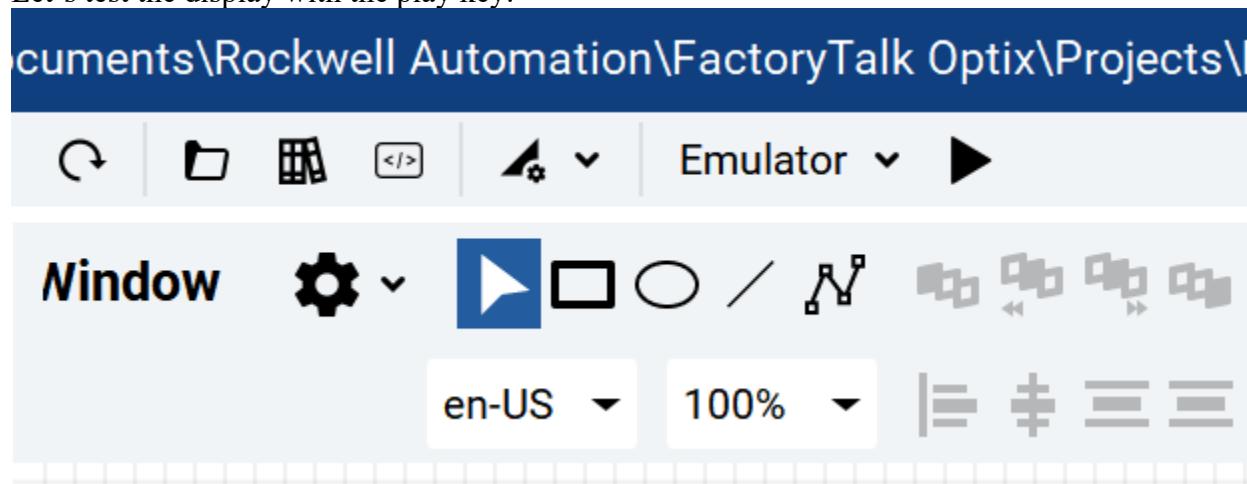
Go down in the properties until Text, and search for the Tag



Select, you will see this



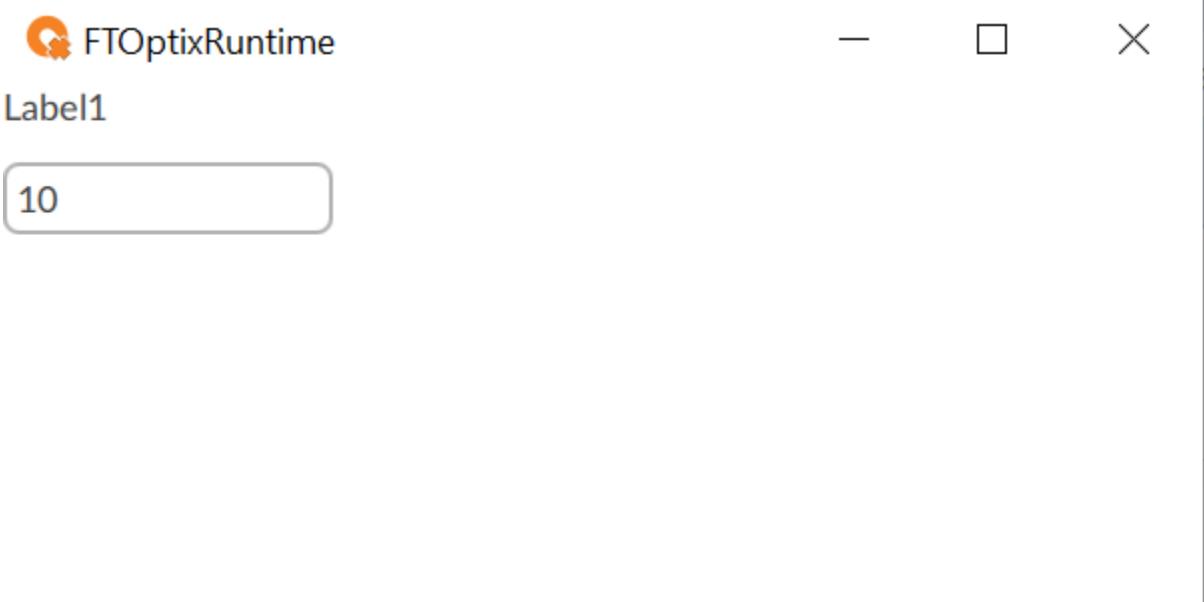
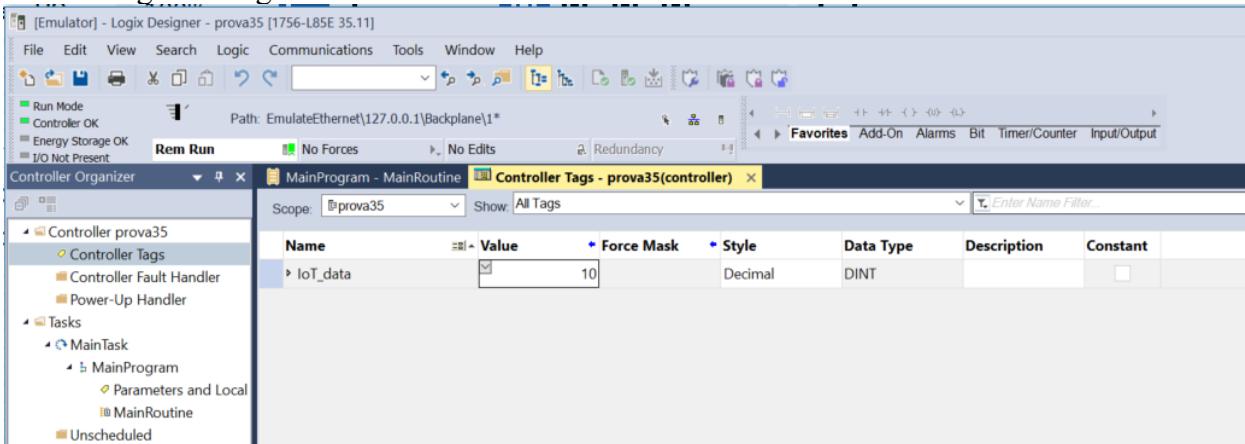
Let's test the display with the play key:



You will see this

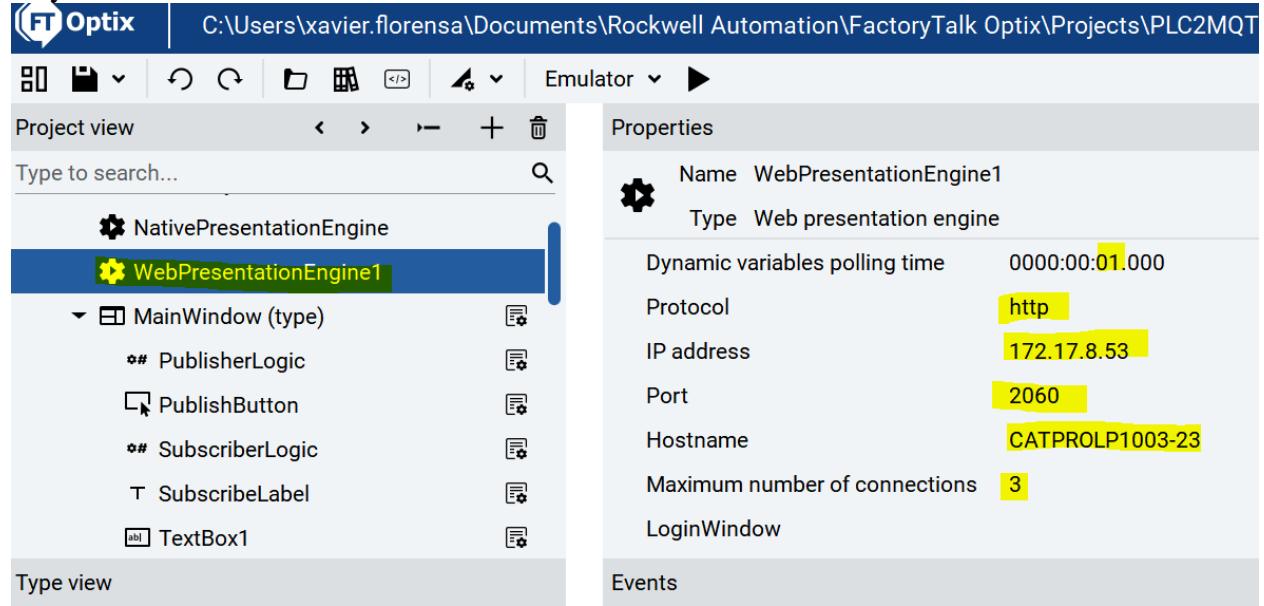


Let's change the Tag value on Studio5000

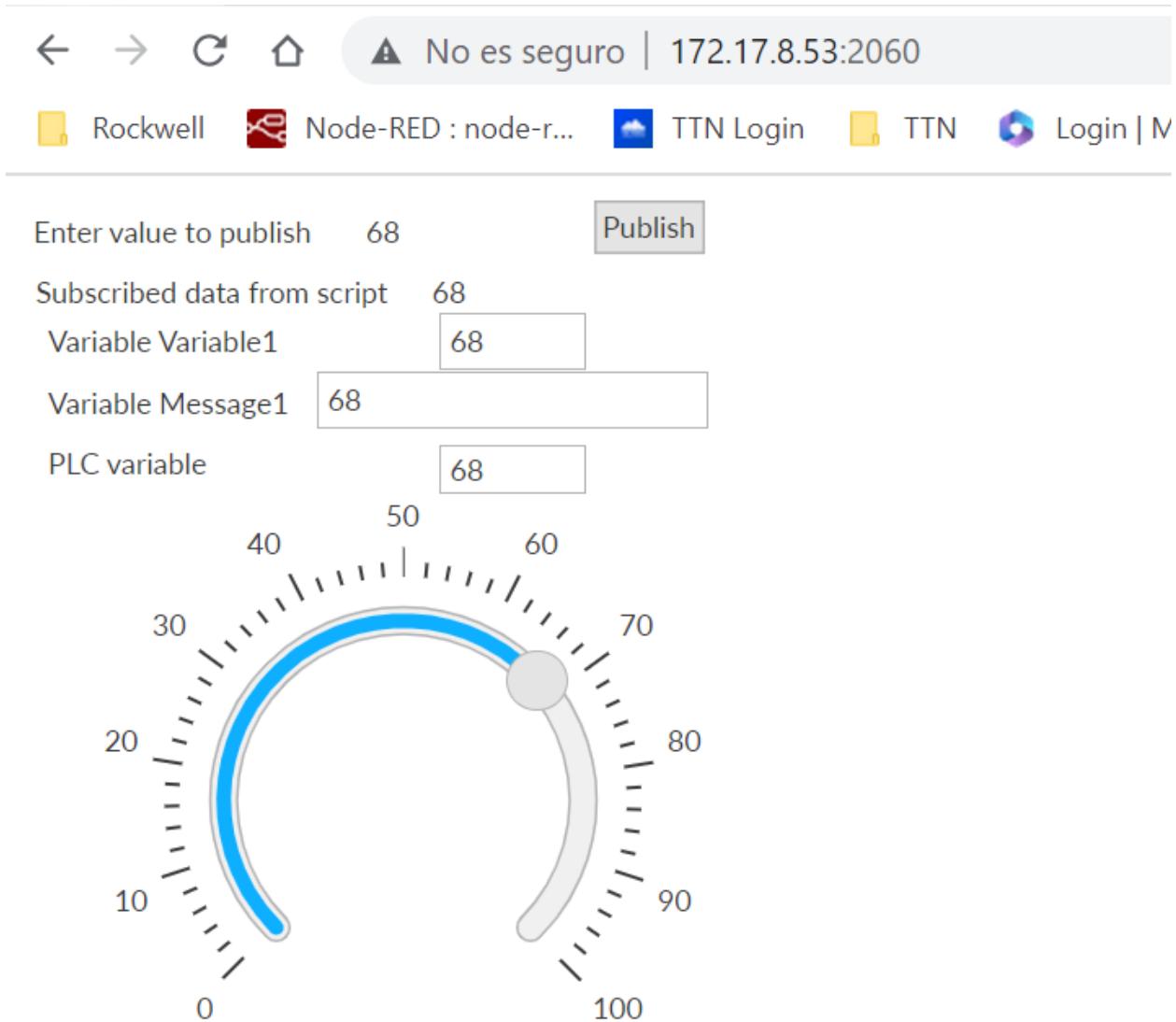


## 5. Web access

Do add a WebPresentationEngine (as new object), place before the MainWindow (type) and set it up with the IP address you have on your PC. Choose a port, for instance 2060, but any port may work.



Open a explorer with the IP address:port  
That's all



On Chrome works fine

By default you can only monitor, not control like the value of the slider.

On Apple Safari, there were some differences on the display.

You can see a demo on web access on this video

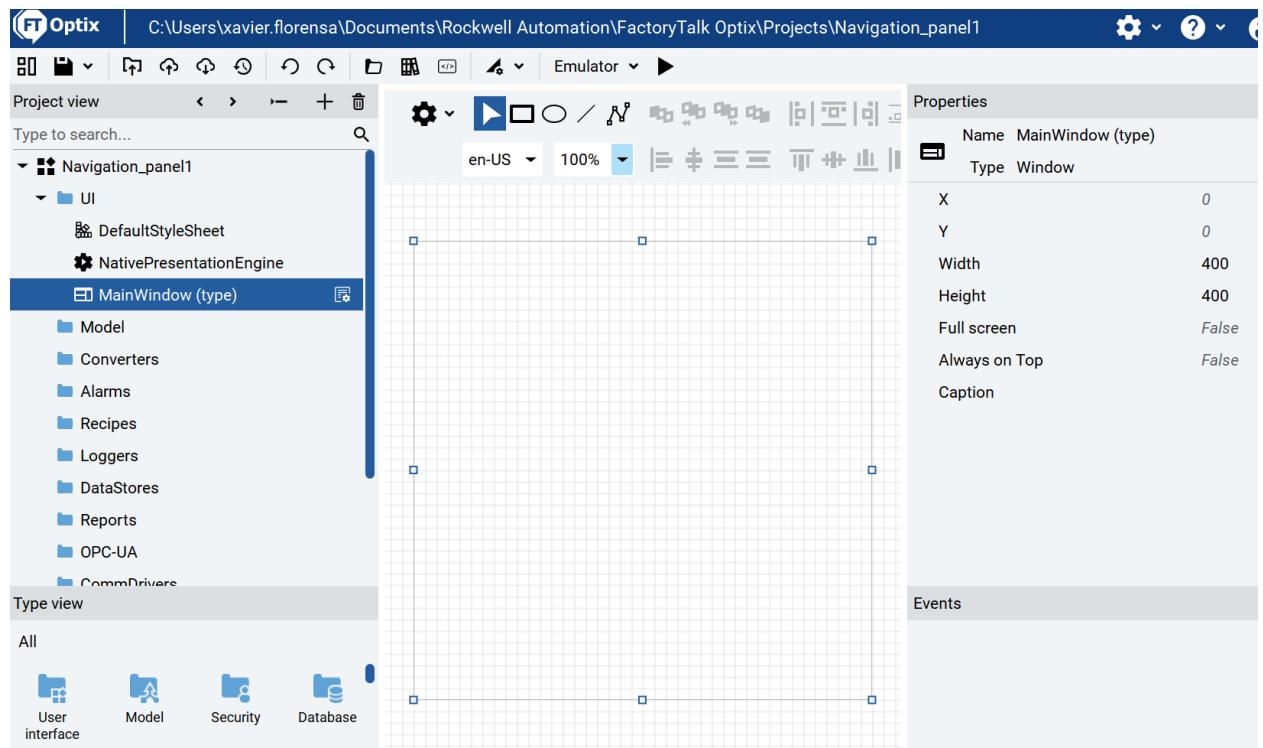
[https://www.youtube.com/watch?v=78\\_cwp0iSJA](https://www.youtube.com/watch?v=78_cwp0iSJA)

## 6. Navigation Panels

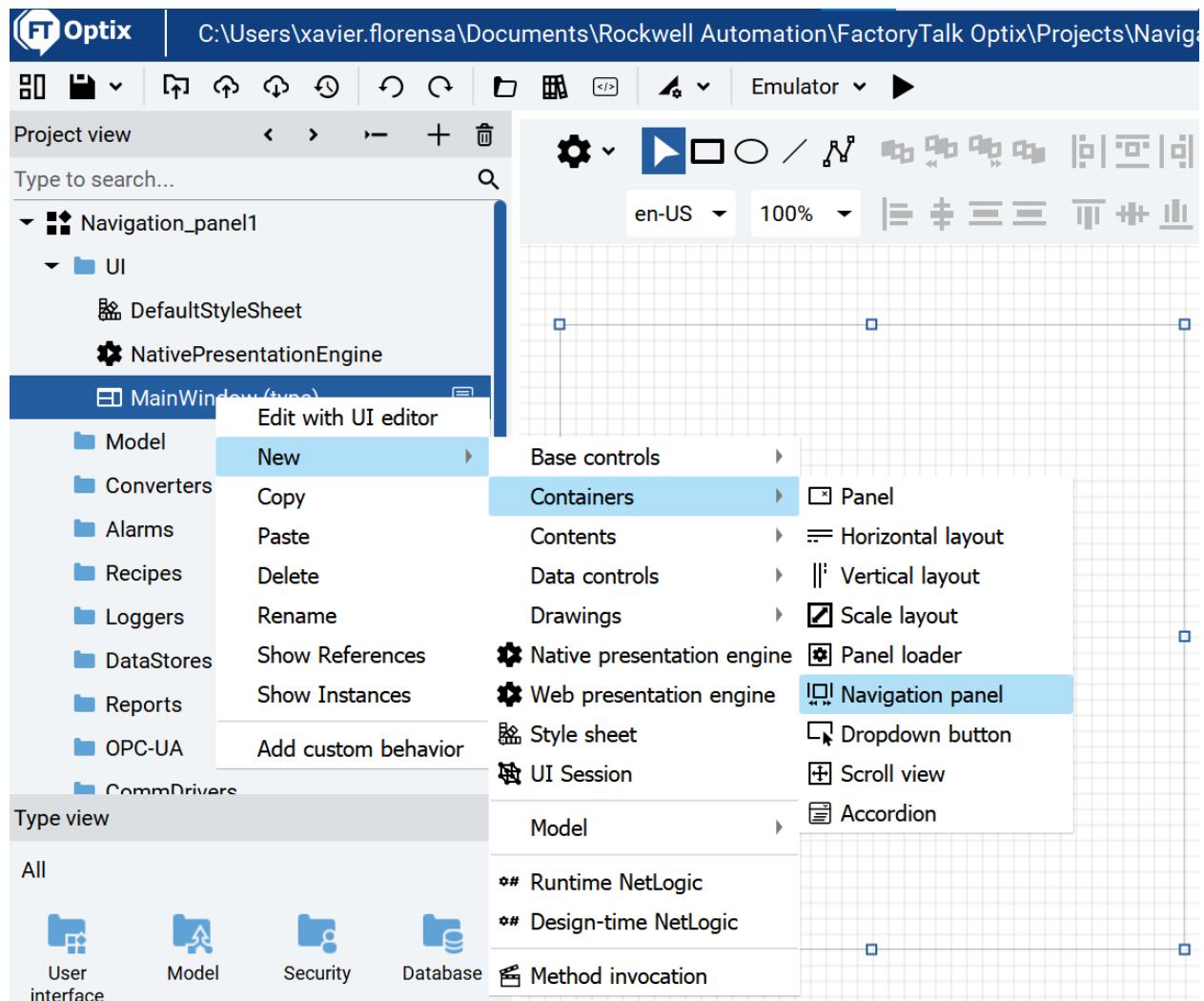
Start with this if you want to have different windows or panels visible on your project.

Create a blank new project

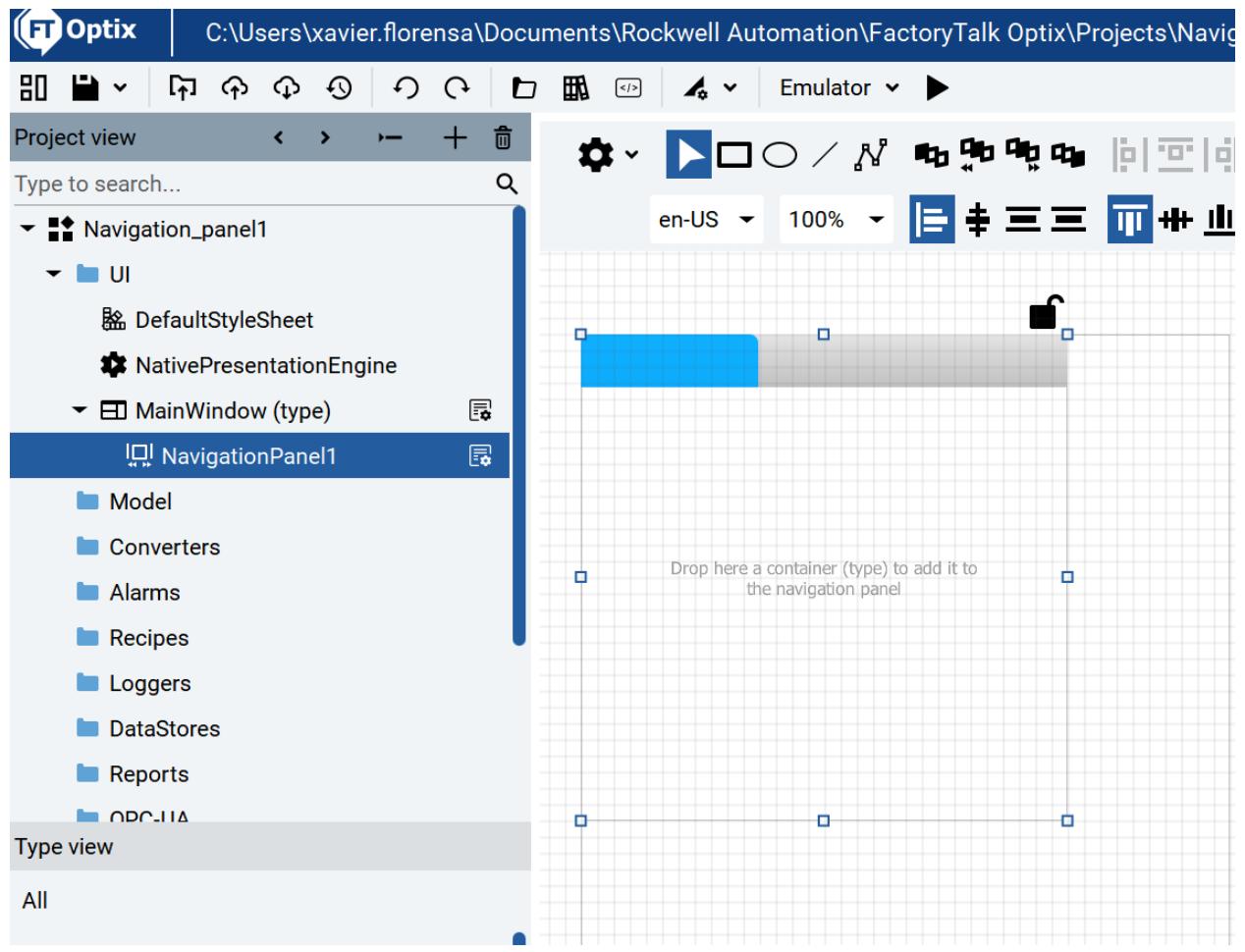
Go to Main window



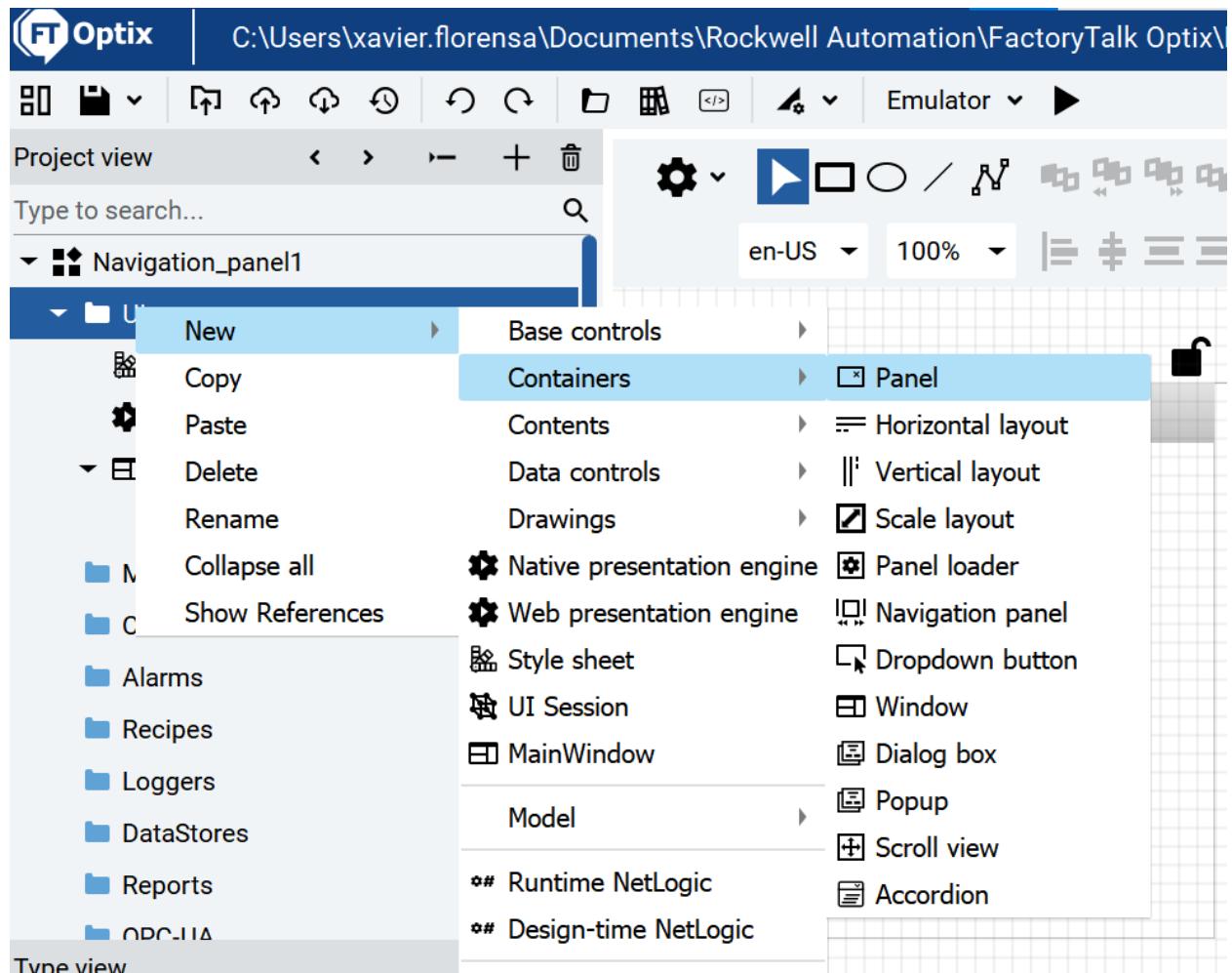
Create a Navigation Pannel with right button click



You will see this

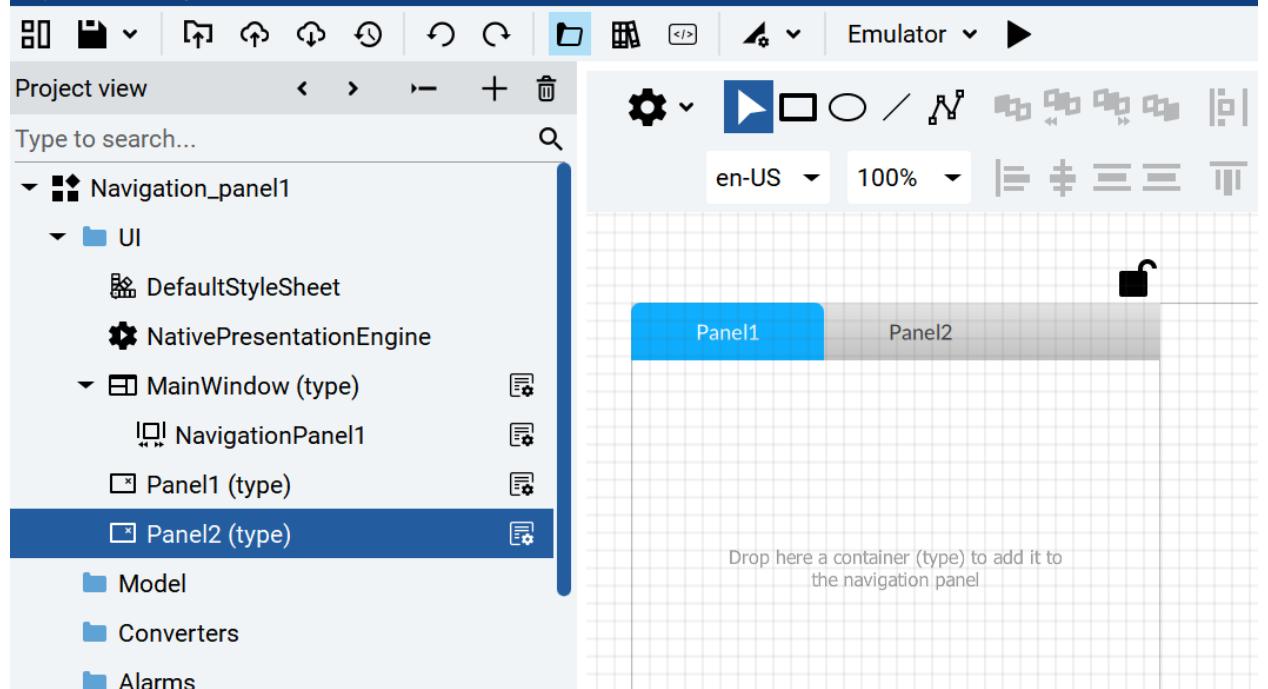
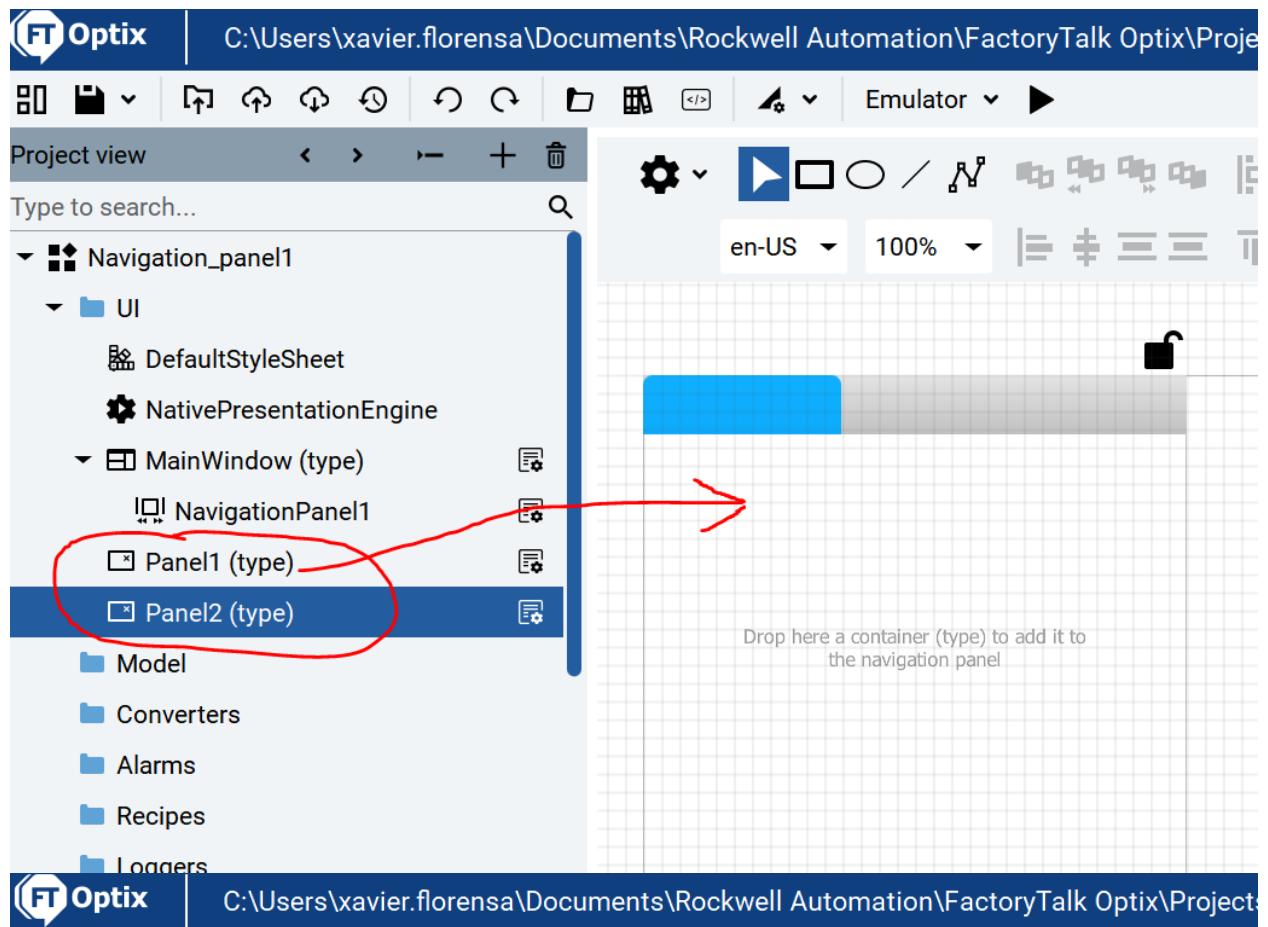


Then create two (or more) new panels with the right mouse button.

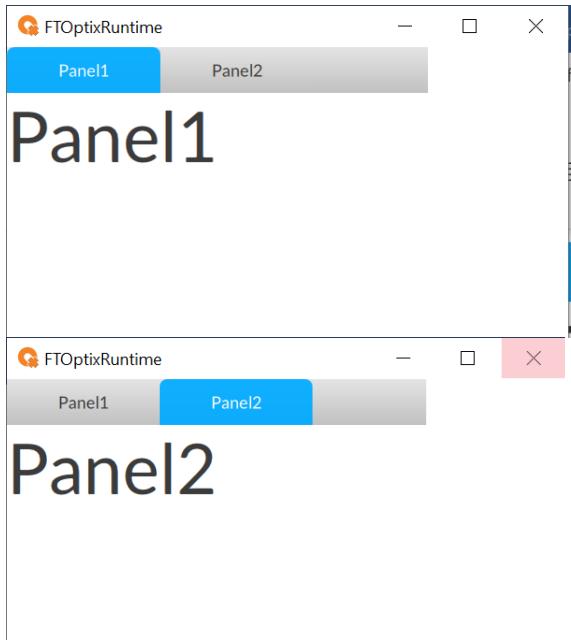


Type view

Then Drag and drop the two panels into the navigation panel



You can add labels to each panel to see on which panel you are



That's all

If you introduce a remote web navigation as on previous chapter, you will be able to have two operators working with different panels each.

You can see a demo on Navigation panels and web access on this video

[https://www.youtube.com/watch?v=78\\_cwp0iSJA](https://www.youtube.com/watch?v=78_cwp0iSJA)

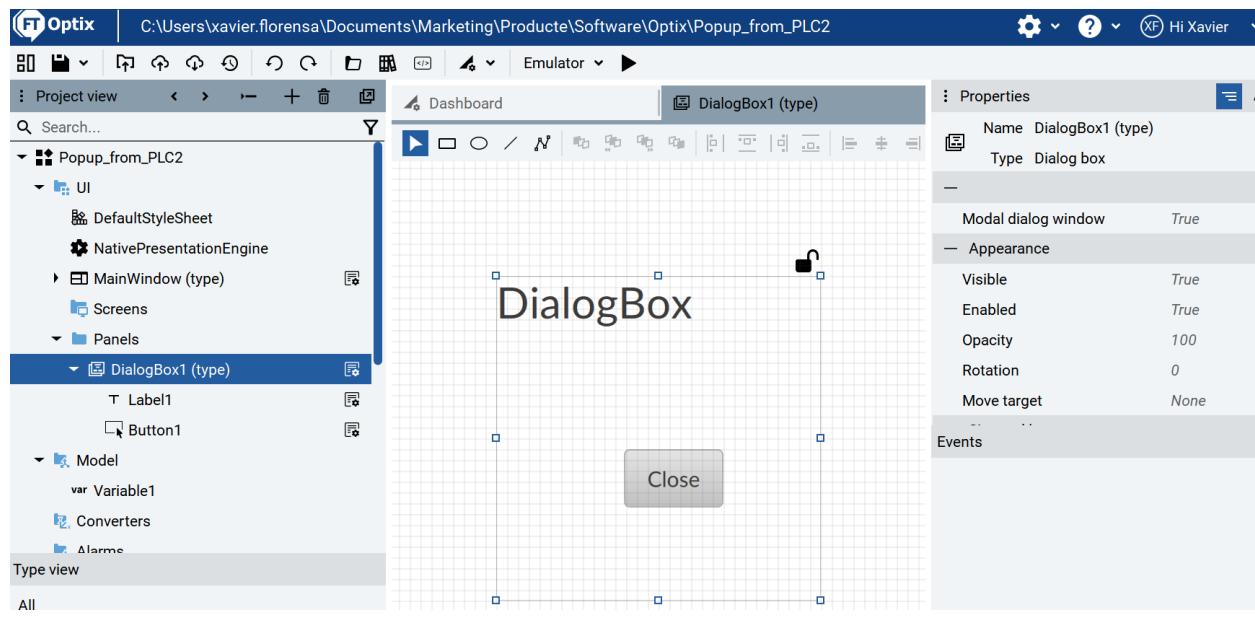
## 7. PopUp Windows

The PopUp Window is called DialogBox in Optix.

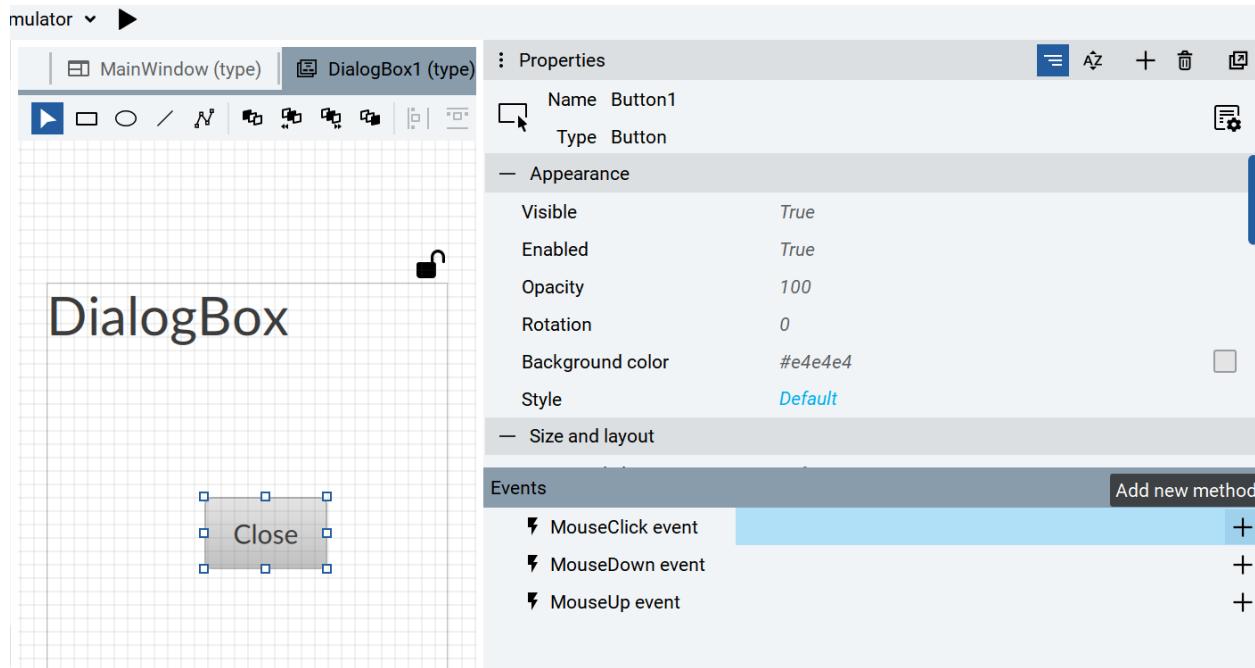
It's easy to create a button to open a DialogBox.

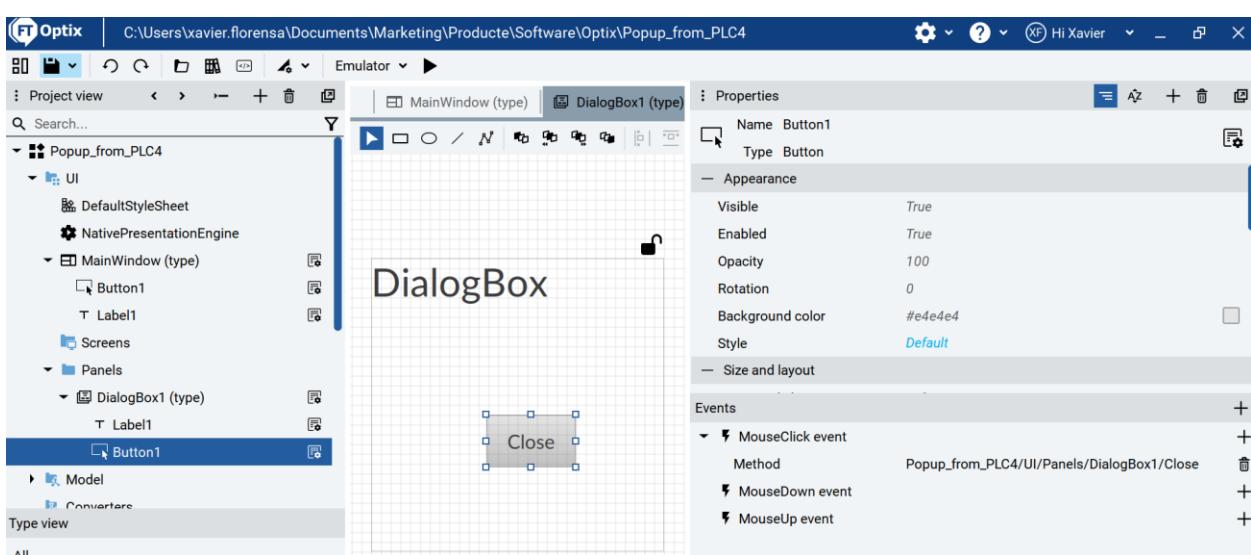
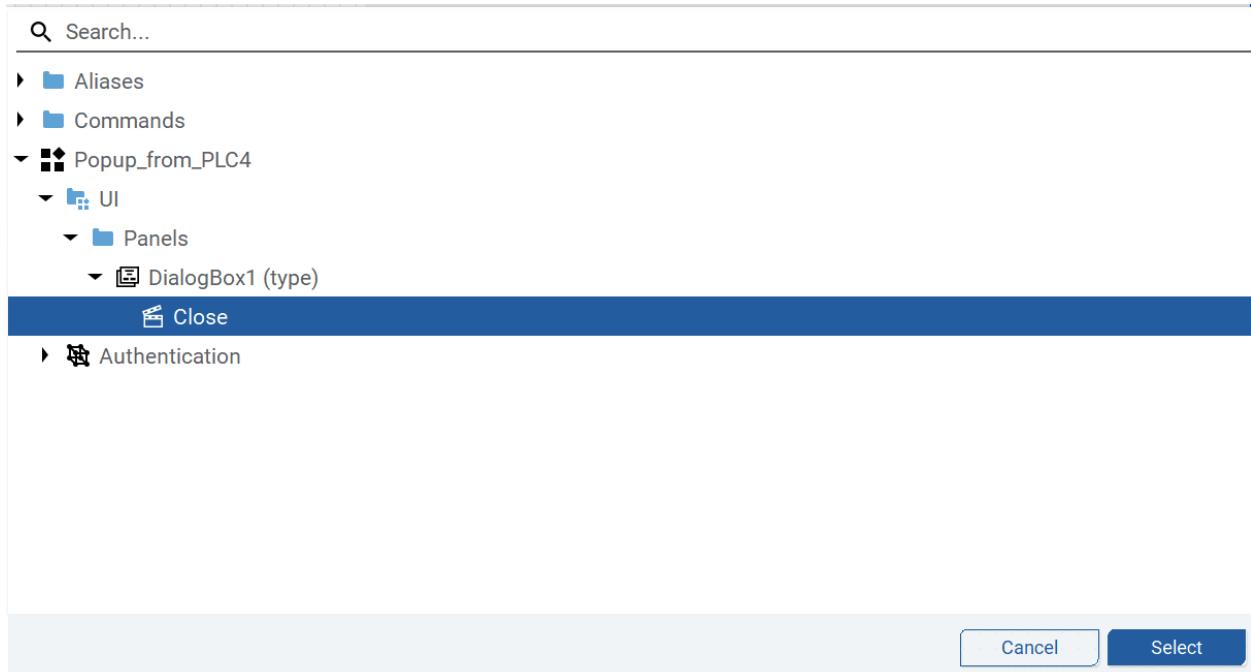
### 7.1. PopUp from Button

Let's create a new DialogBox

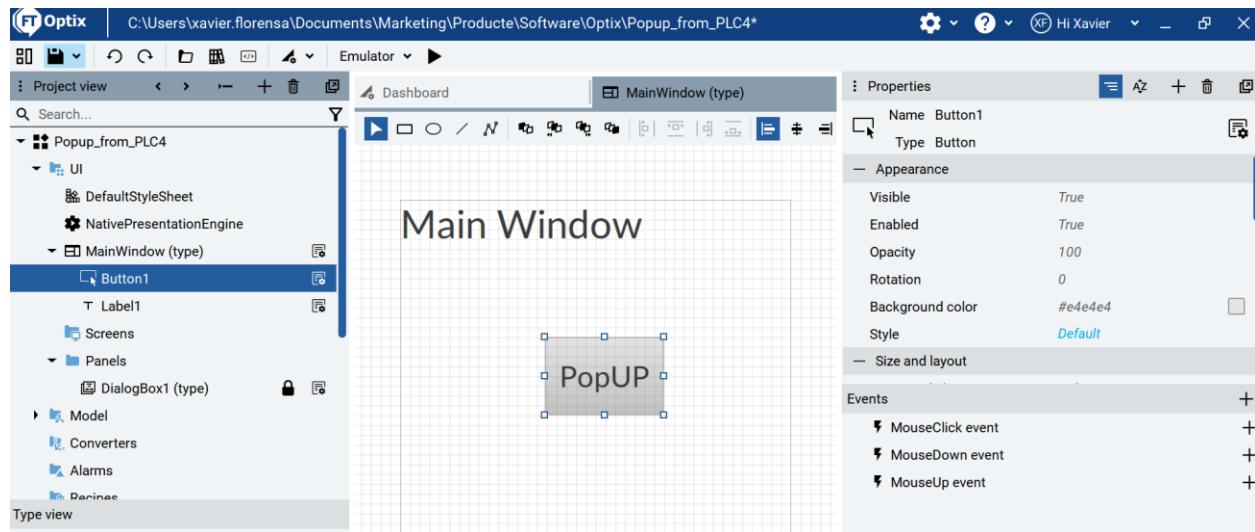


Add also a close Button

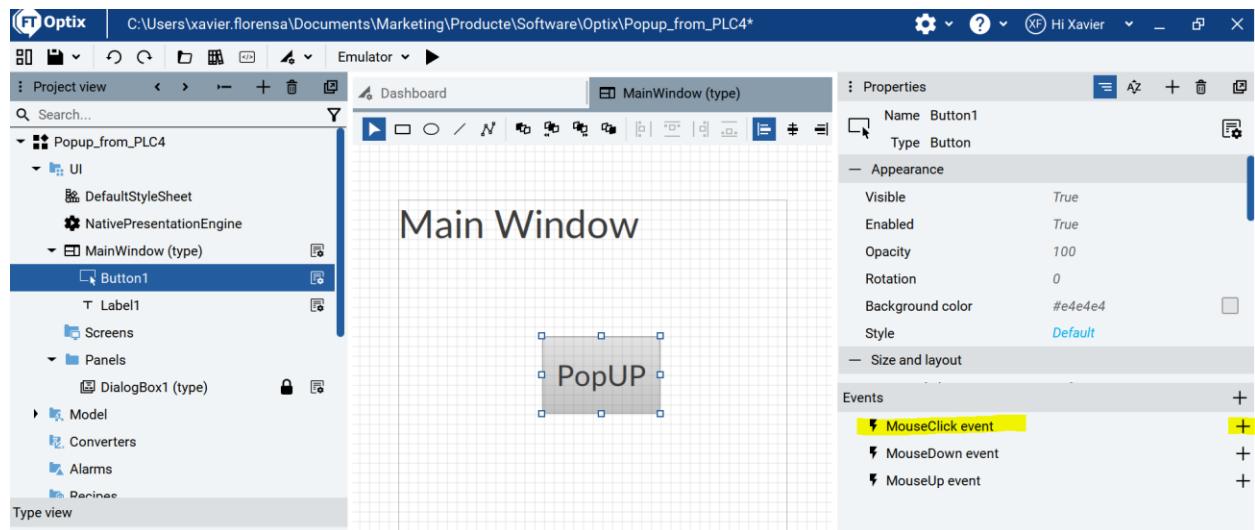




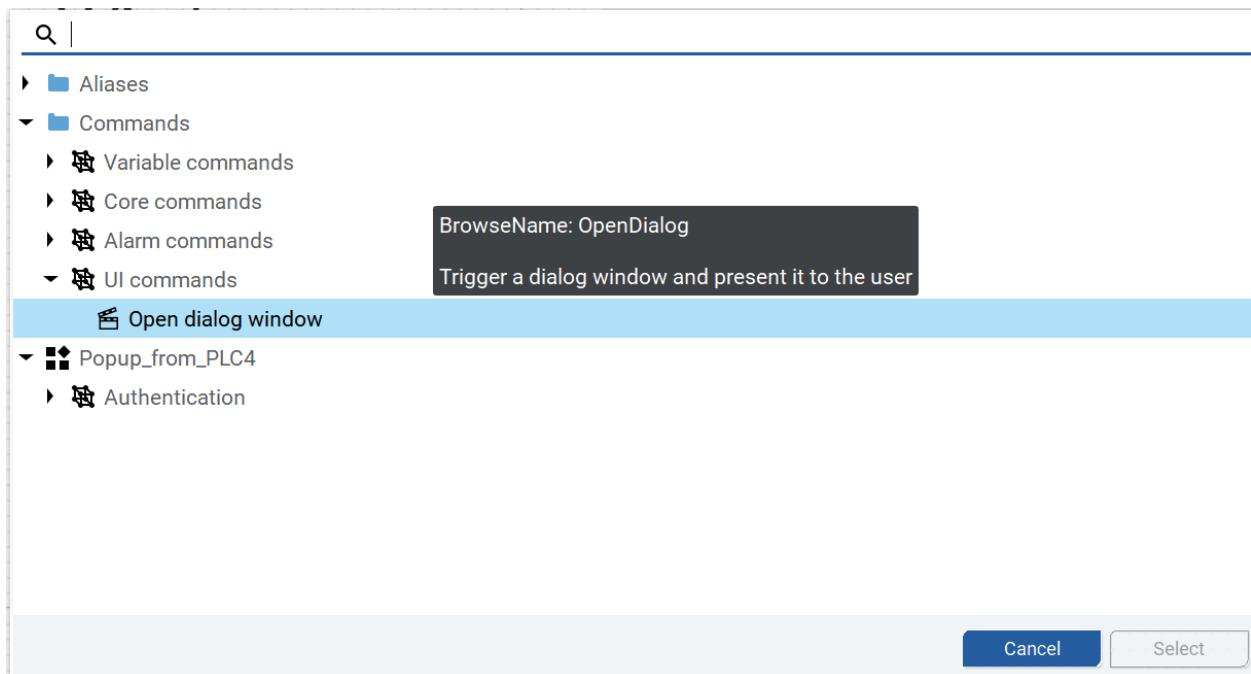
Now create a Button for instance on MainWindow



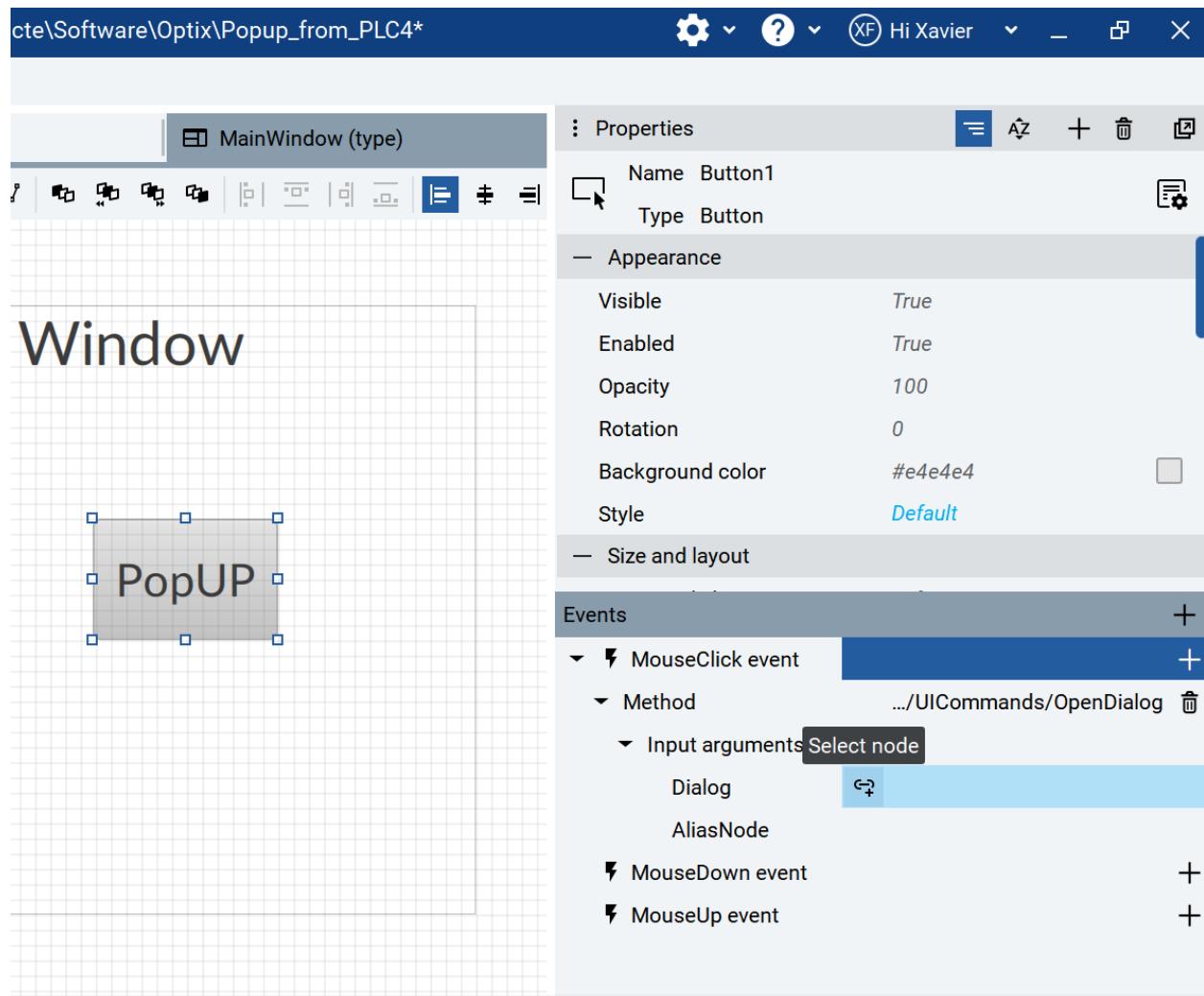
Create a new MouseClick event with +



Navigate to



And select the DialogBox



## Select

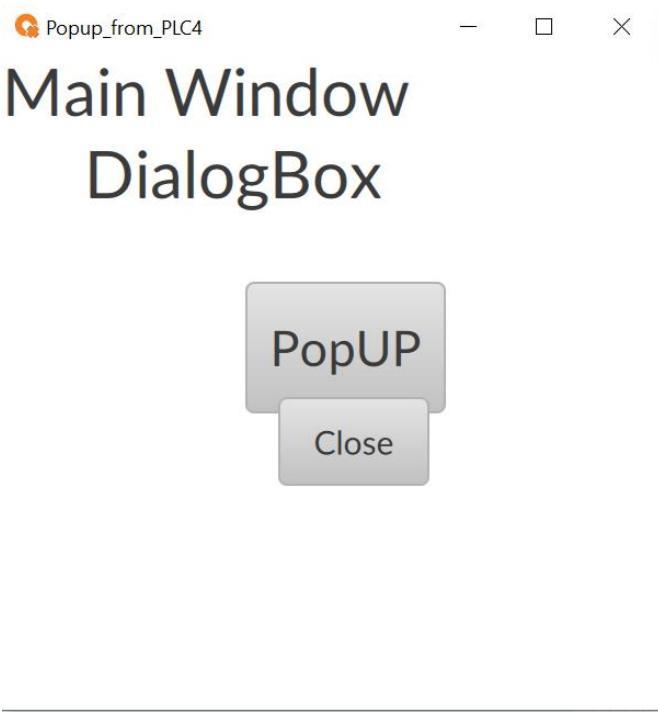
Basic   Advanced   JI ▶ MainWindow ▶ Button1 ▶ MouseClickEventHandler1 ▶ MethodsToCall ▶ MethodContainer1 ▶ InputArguments ▶ Dialog

Search... Y

Attribute ⓘ ▼

Retained alarms  
AllAlarms  
CommController  
OPCUAClientController  
Popup\_from\_PLC4  
UI  
DefaultStyleSheet  
NativePresentationEngine  
MainWindow (type)  
Screens  
Panels  
DialogBox1 (type)  
Model  
Converters

Cancel Select Remove link



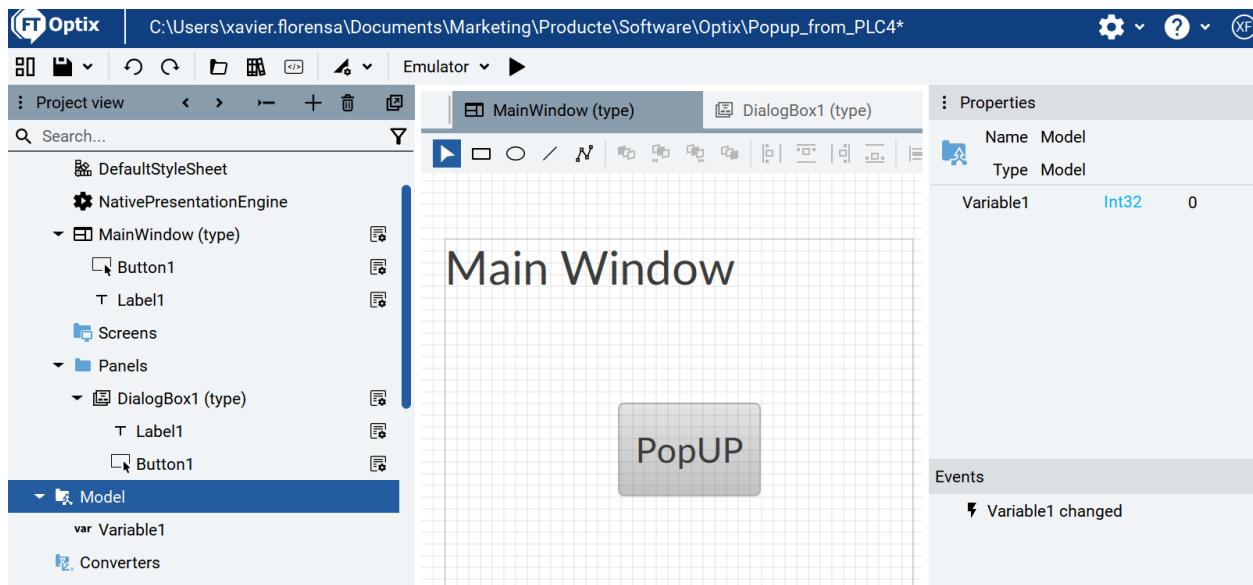
You can add a colored rectangle to overlap and do not see what's behind the Dialog Box

## 7.2. PopUp from variable

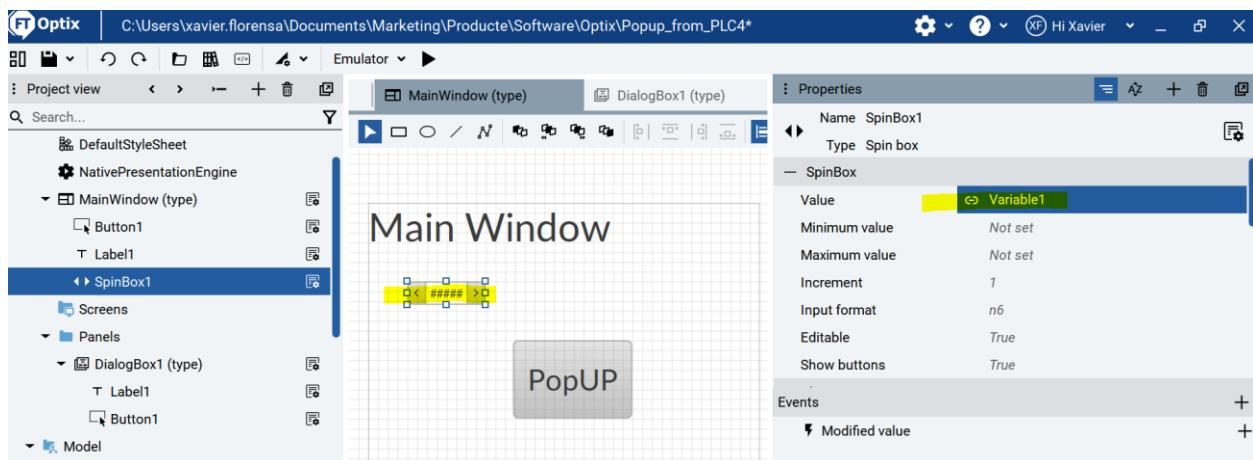
You can get the code from here

[https://github.com/xavierflorensa/Popup\\_from\\_PLC2](https://github.com/xavierflorensa/Popup_from_PLC2)

Create a new variable on Model, this will be used to trigger the DialogBox if changes

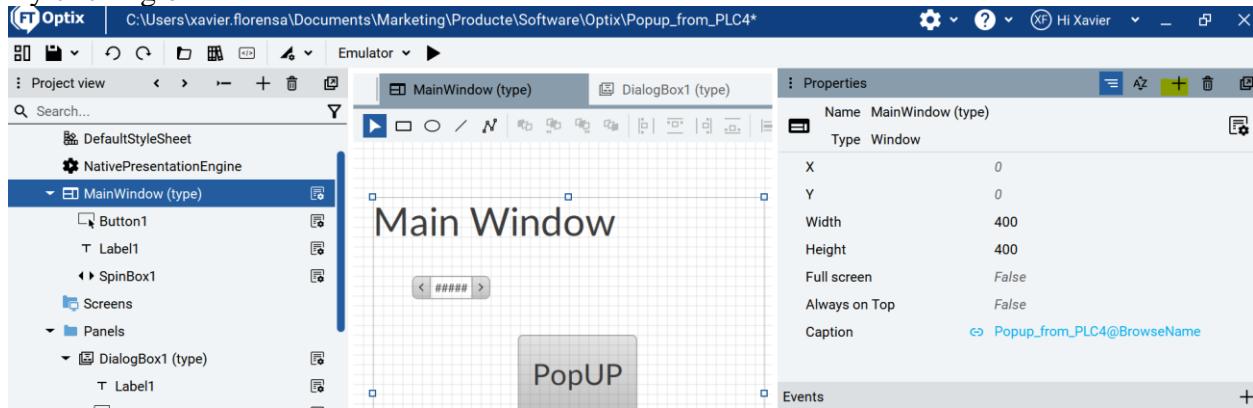


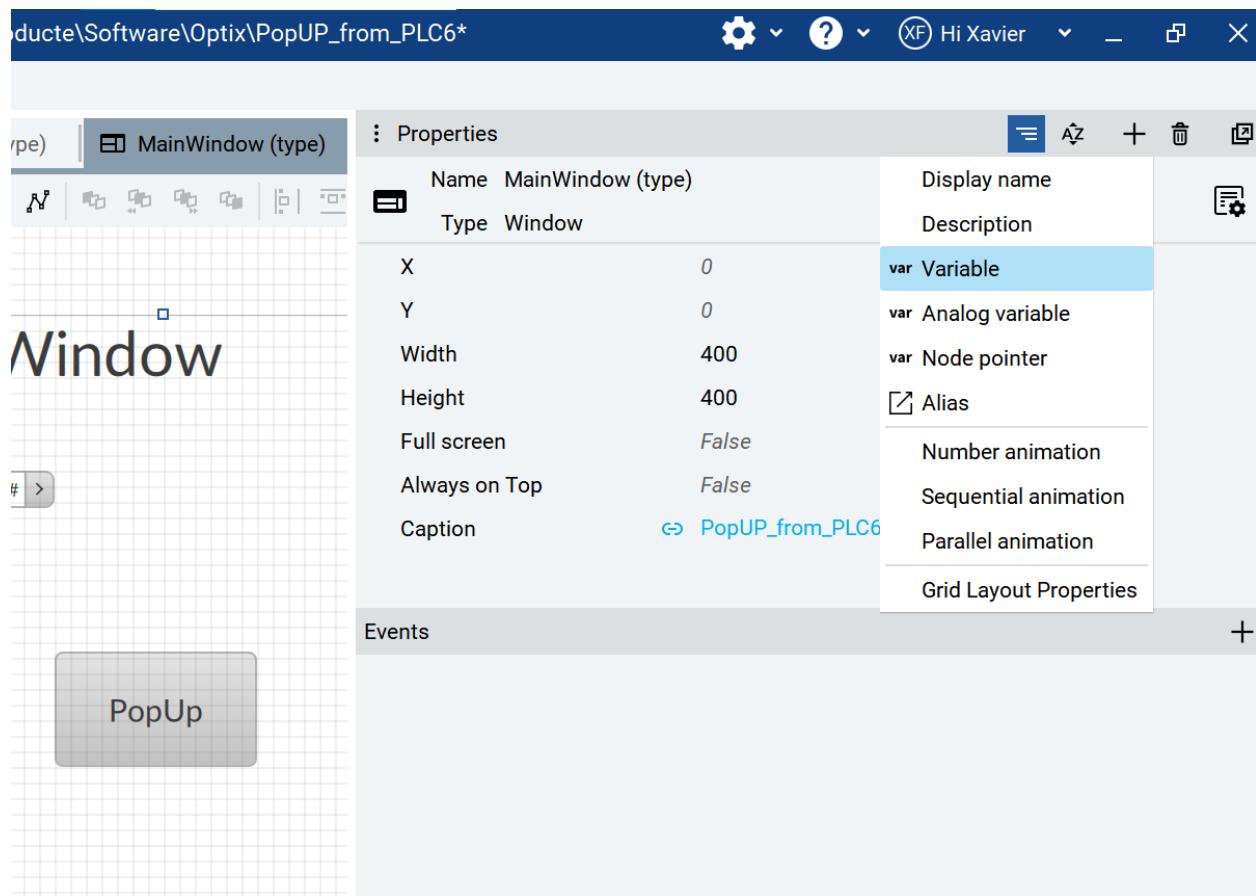
Let's create a SpinBox to be vinculated with this variable



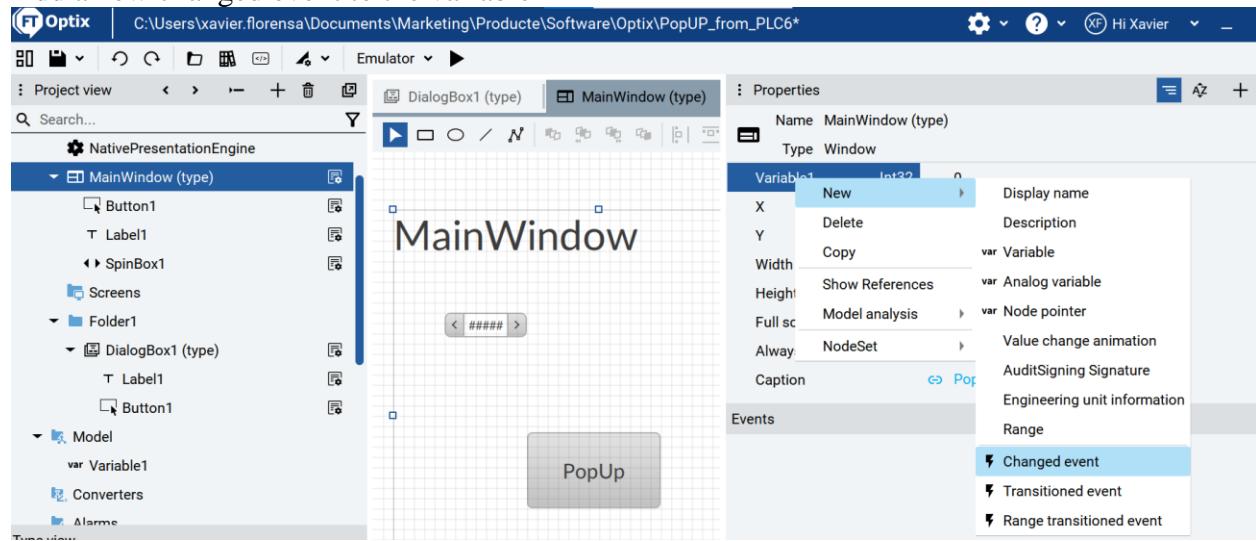
Now let's create a new variable on the Main Window (Not on Model)

By clicking on +

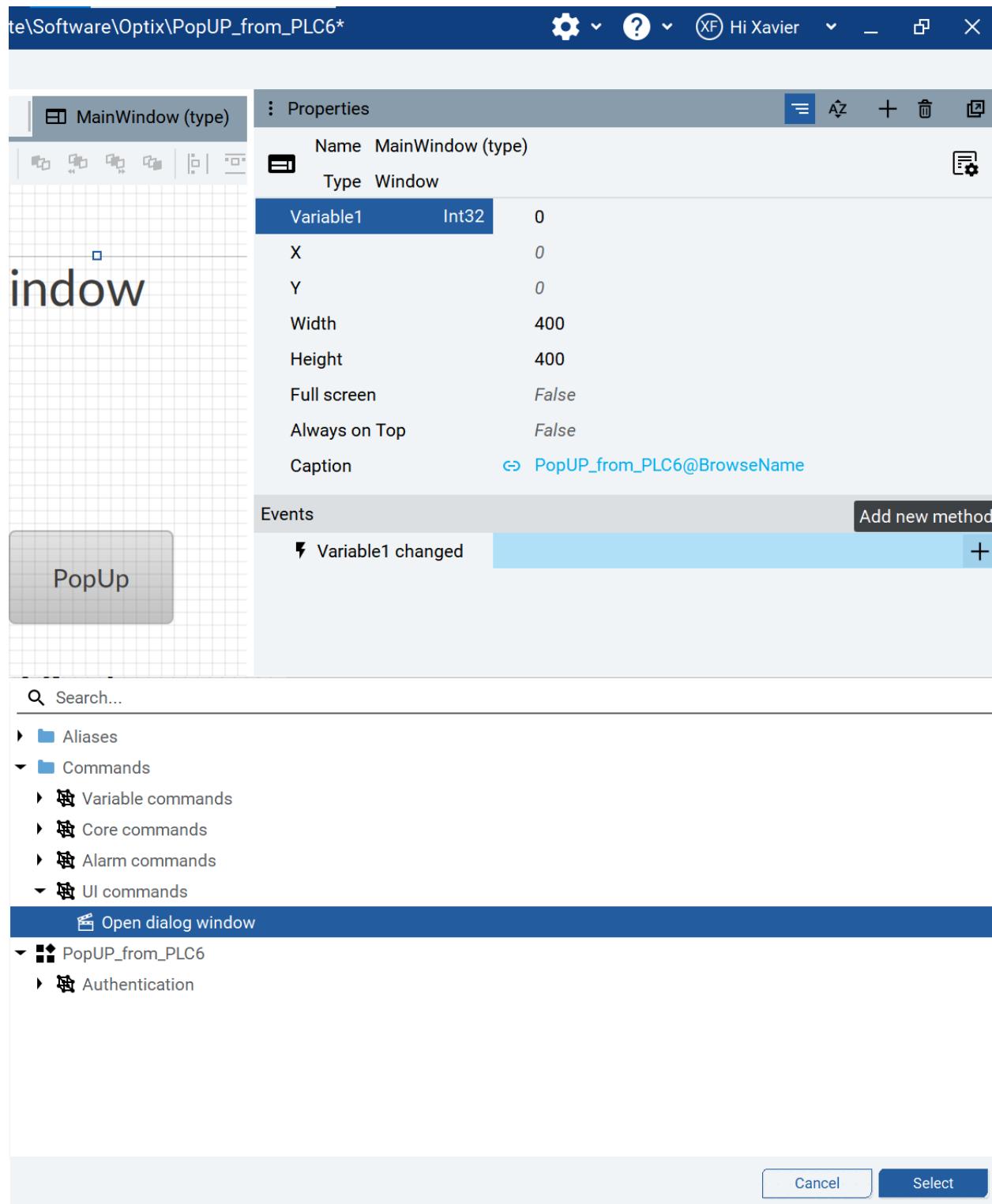




Add a new changed event to the variable



Select the method



and the Dialog to open

ucte\Software\Optix\PopUP\_from\_PLC6\*

**Properties**

Name	Type	Value
Variable1	Int32	0
X		0
Y		0
Width		400
Height		400
Full screen		False
Always on Top		False
Caption		PopUP_from_PLC6@BrowseName

**Events**

- Variable1 changed
- Method /Root/Objects/Commands/UICommands/OpenDialog
- Input arguments Select node

  - Dialog
  - AliasNode

**Basic**    **Advanced**    Variable1Changed ▶ VariableValueChangedEventHandler1 ▶ MethodsToCall ▶ MethodContainer1 ▶ InputArguments ▶ Dialog

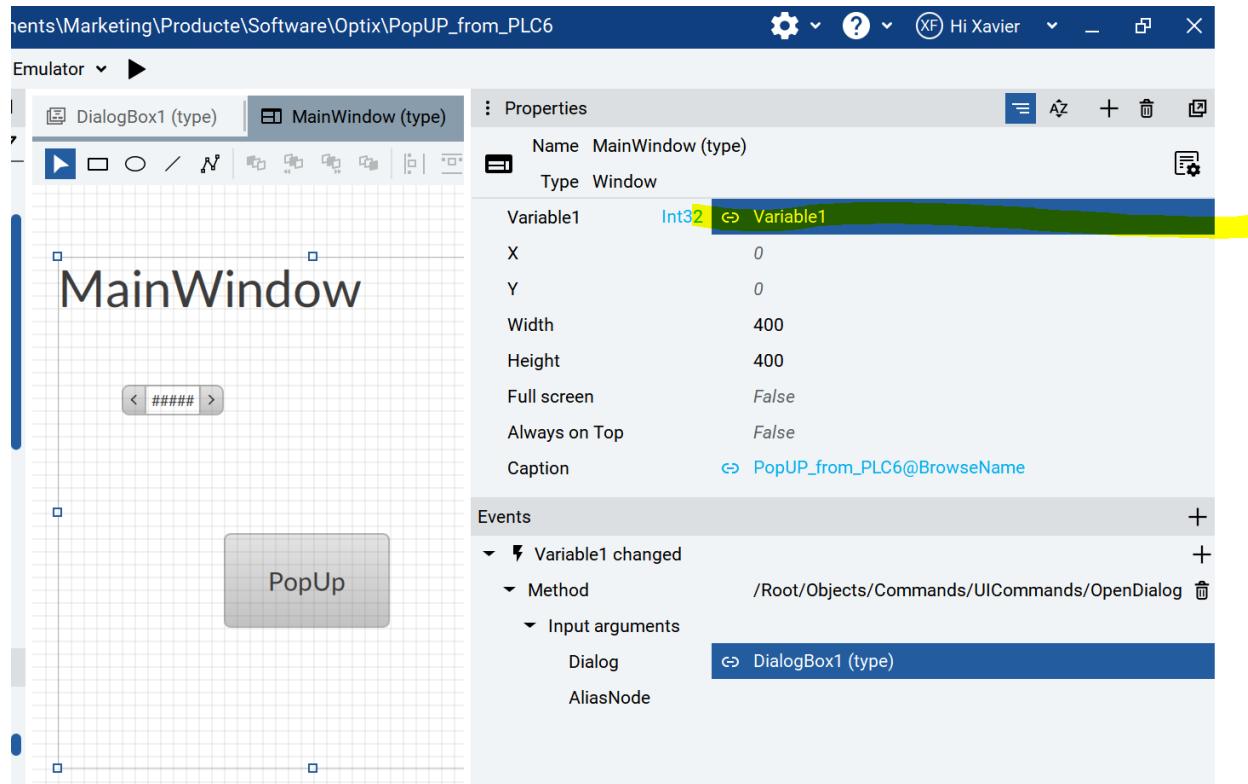
Search... Y

Attribute ⓘ

Retained alarms  
AllAlarms  
CommController  
OPCUAClientController  
PopUP\_from\_PLC6  
UI  
DefaultStyleSheet  
NativePresentationEngine  
MainWindow (type)  
Screens  
Folder1  
DialogBox1 (type)  
Model  
Converters

Cancel Select Remove link

Finally vinculate Variable1 from MainWindow to the variable from Model



That's all

### 7.3. PopUp from PLC value

You have an example here

Only one DialogBox

[https://github.com/xavierflorensa/FTOptix\\_Popup\\_from\\_PLC\\_CSHARP](https://github.com/xavierflorensa/FTOptix_Popup_from_PLC_CSHARP)

Multiple Dialogboxes

<https://github.com/FactoryTalk-Optix/DialogByPlcValue.git>

## 8. Reusable Objects. Working with Alias

You can learn to work with Alias in those series of videos

**FactoryTalk Optix Reusable Graphics Part 1 - Create Reusable Graphics**

<https://www.youtube.com/watch?v=nVEAtHQeFu4&list=PLctxwem-UjOVrNpq7SqwHjy0mYjqHwqj&index=7>

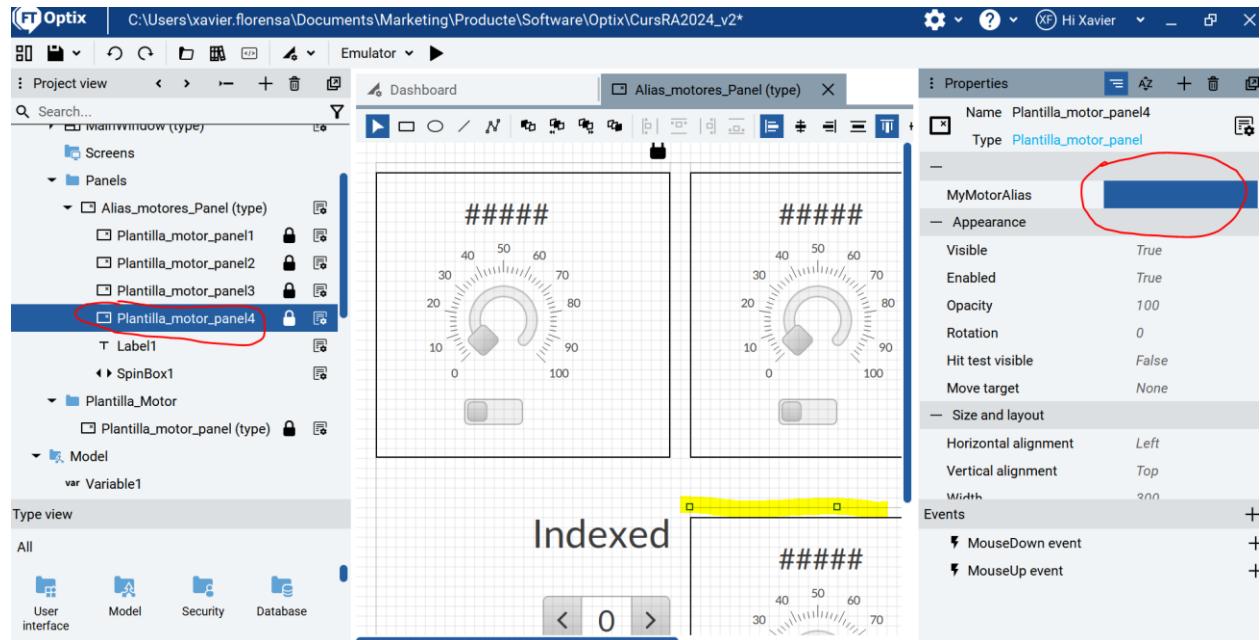
## FactoryTalk Optix Reusable Graphics Part 2 - Object Model Alias

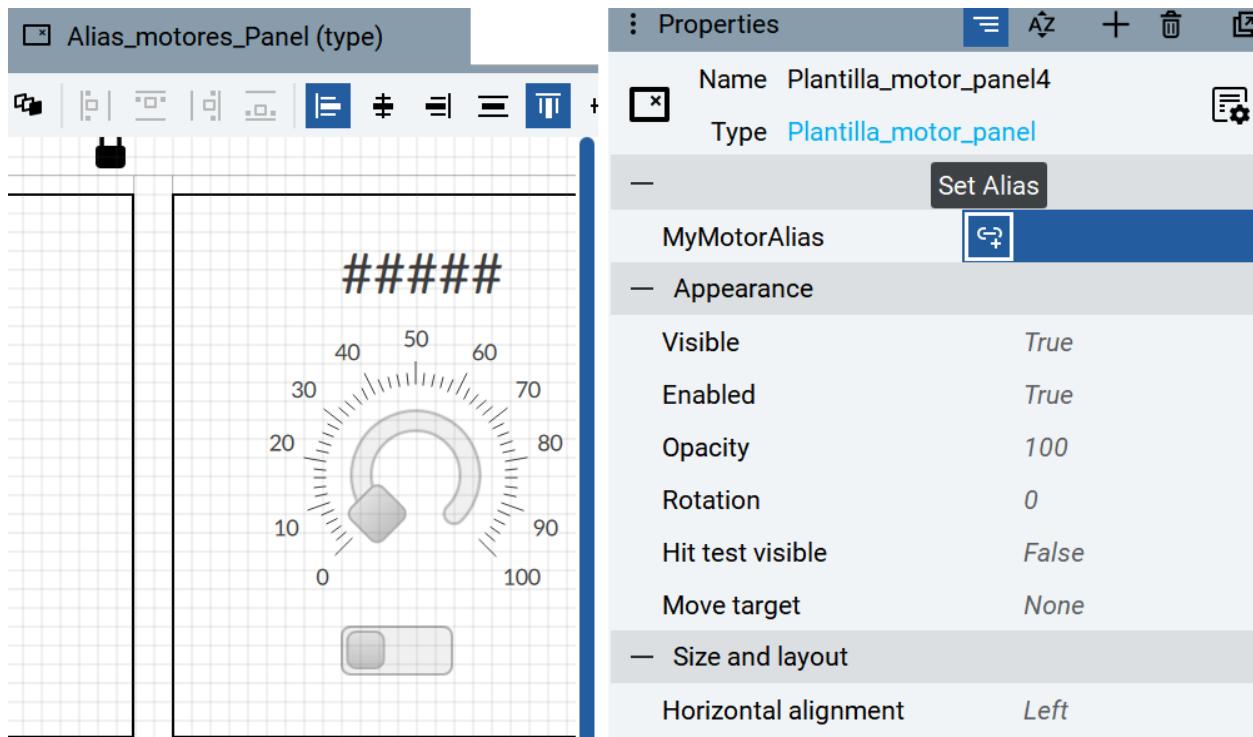
[https://www.youtube.com/watch?v=O\\_royUATME&list=PLctxwem-UjOVrNpq7SqwHjy0mYjqHwqj&index=8](https://www.youtube.com/watch?v=O_royUATME&list=PLctxwem-UjOVrNpq7SqwHjy0mYjqHwqj&index=8)

With this you can use several Widgets instances pointing to different UDT's

But what if you have lots of motors

On the same project defined on the videos you can create an indexed widget  
But instead of pointing directly to the UDT instances let's point this way





Basic Advanced

UI > Panels > Alias\_motores\_Panel > Plantilla\_motor\_panel4 > MyMotorAlias

Search: |

- Aliases
- Panel Loaders
- Types
- Commands
- Retained alarms
- AllAlarms
- CommController
- OPCUAClientController
- CursRA2024\_v2
  - UI
    - DefaultStyleSheet
    - NativePresentationEngine
    - MainWindow (type)
    - Screens

Cancel Select Remove link

Basic Advanced

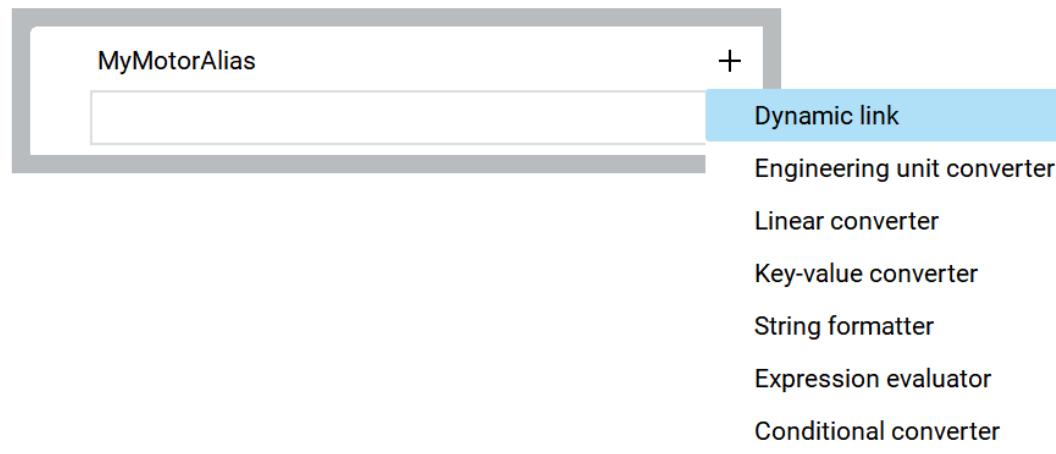
UI ▶ Panels ▶ Alias\_motores\_Panel ▶ Plantilla\_motor\_panel4 ▶ MyMotorAlias



[Close](#) [Remove link](#) [Open in tab](#)

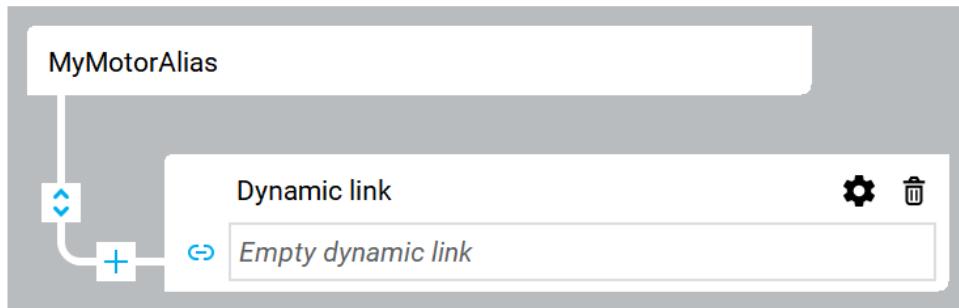
Basic Advanced

UI ▶ Panels ▶ Alias\_moto

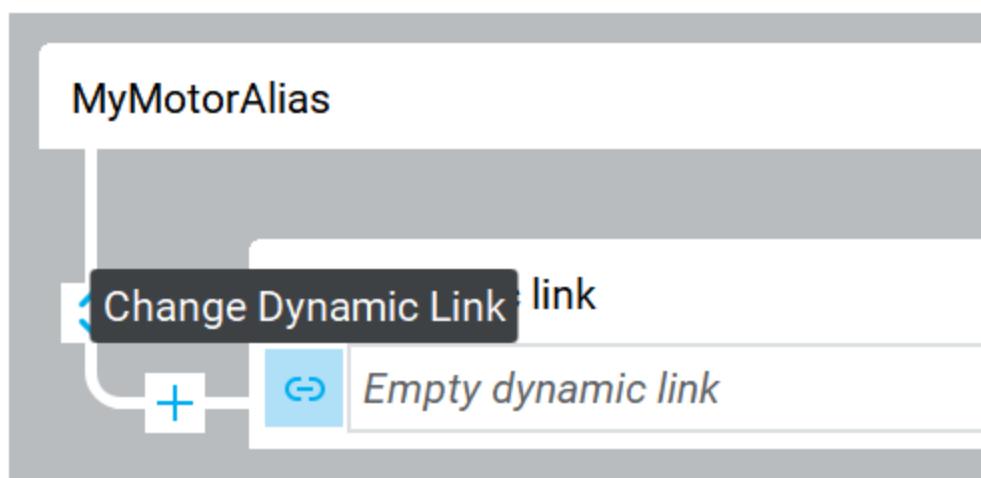


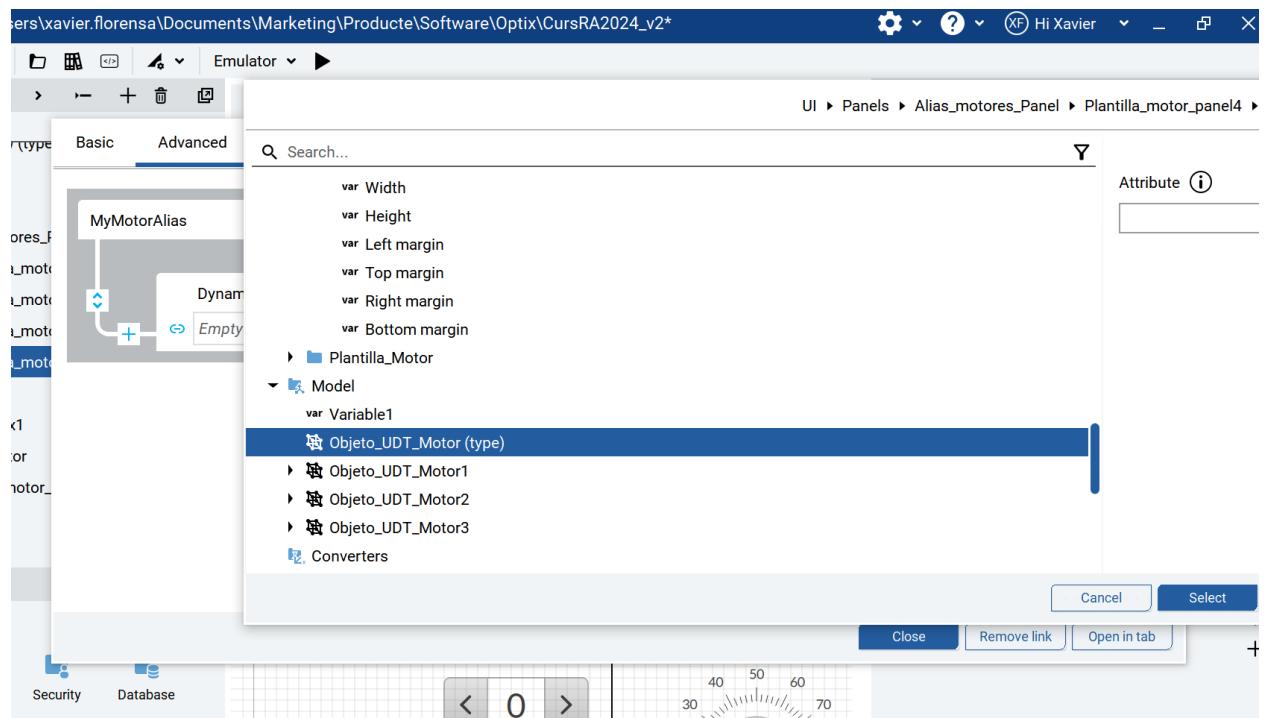
Basic Advanced

UI ▶ Panels ▶ Alias\_

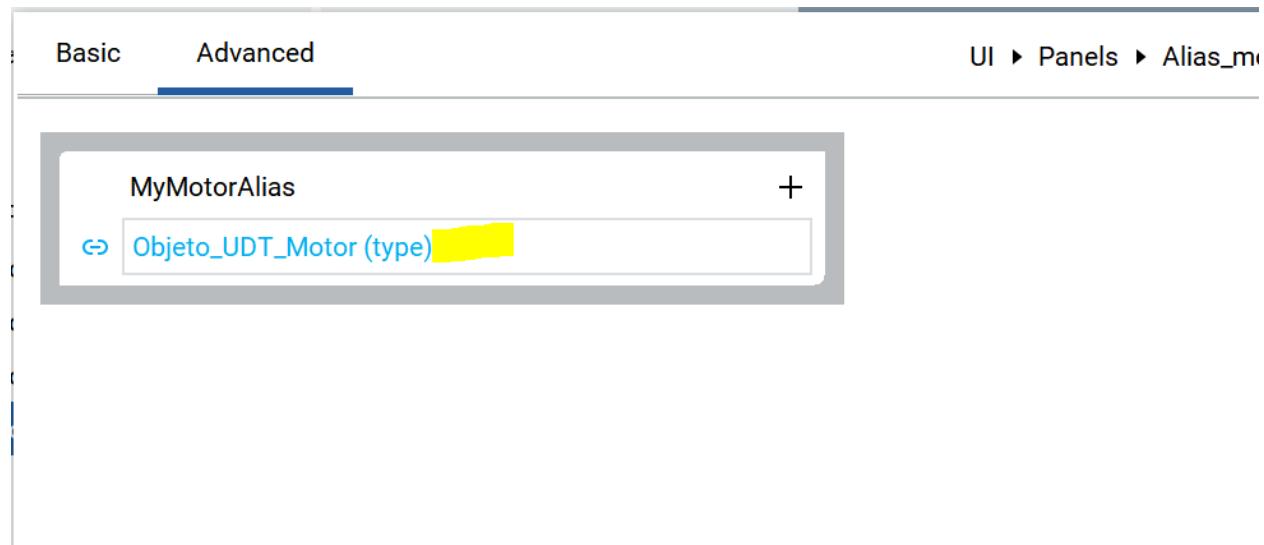


Basic Advanced





Add {0} at the end and click enter



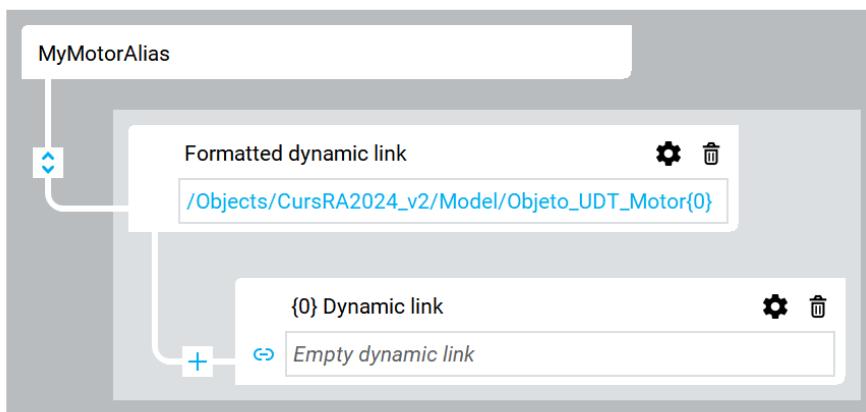
Basic Advanced

UI ▶ Panels ▶ Ali



Basic Advanced

UI ▶ Panels ▶ Alias\_motores\_Panel ▶ Plantilla\_m

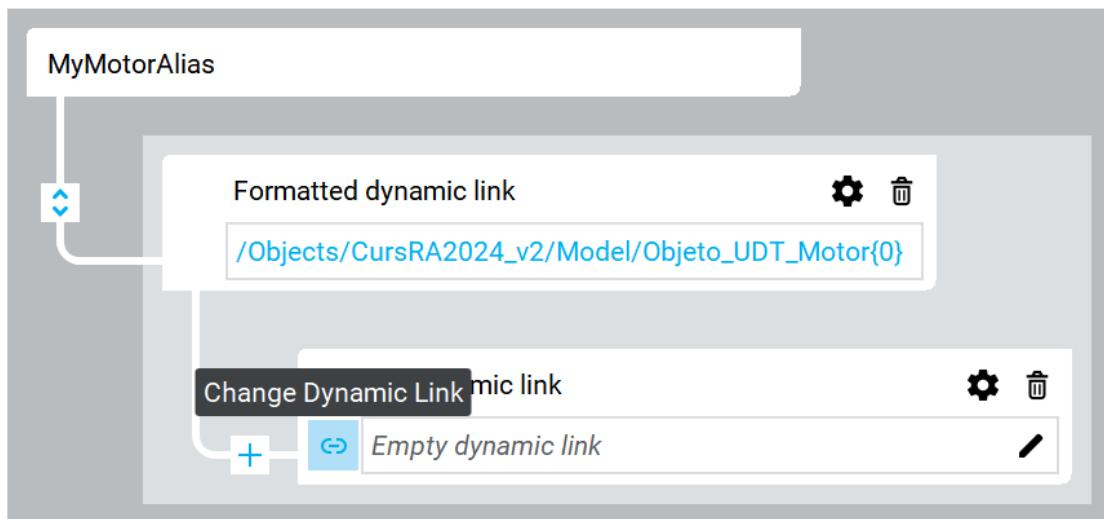


Select the dynamic link

Close

Basic Advanced

UI ▶ Panels ▶ Alias\_m



UI ▶ Panels ▶ Alias\_motores\_Panel ▶ Plantilla\_motor\_panel4 ▶ MyMotorAlias ▶ DynamicLink ▶ DynamicLinkFormatter ▶ Source0

Search...

- var Left margin
- var Top margin
- var Right margin
- var Bottom margin
- ↳ T Label1
- ↳ □ SpinBox1
  - var Value
  - var Minimum value
  - var Maximum value
  - var Increment
  - var Input format
  - var Editable
  - var Show buttons
  - var Visible

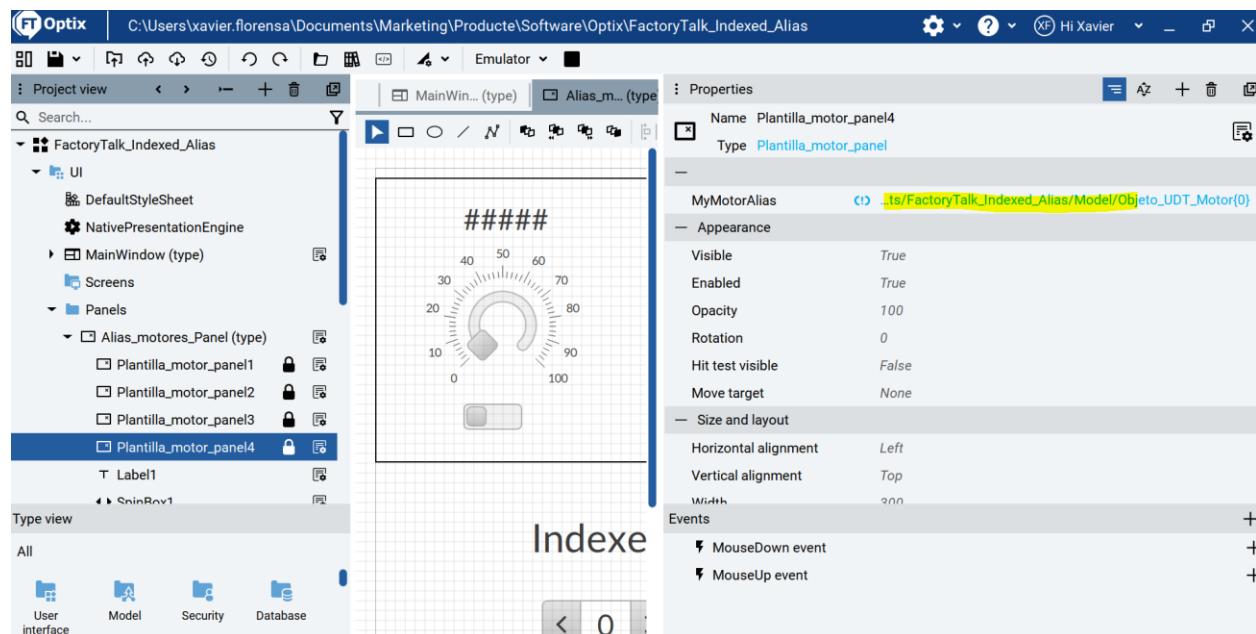
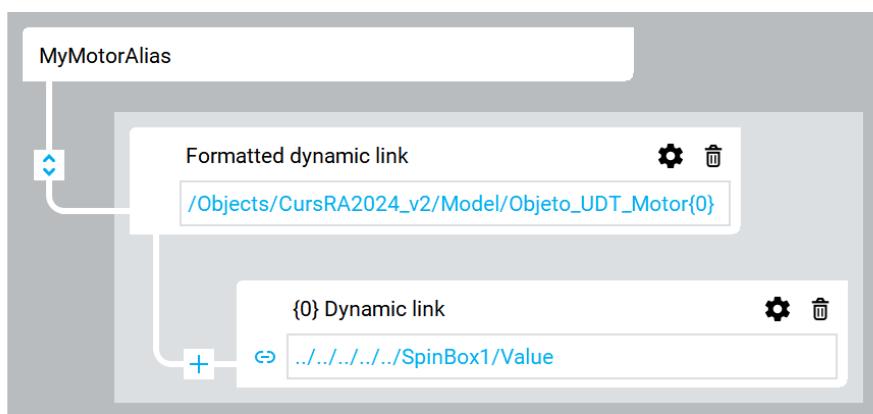
Attribute ⓘ

Format ⓘ

Cancel Select

Basic Advanced

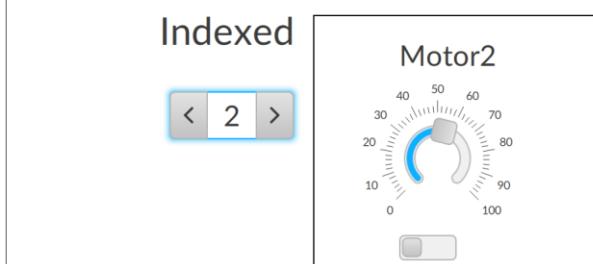
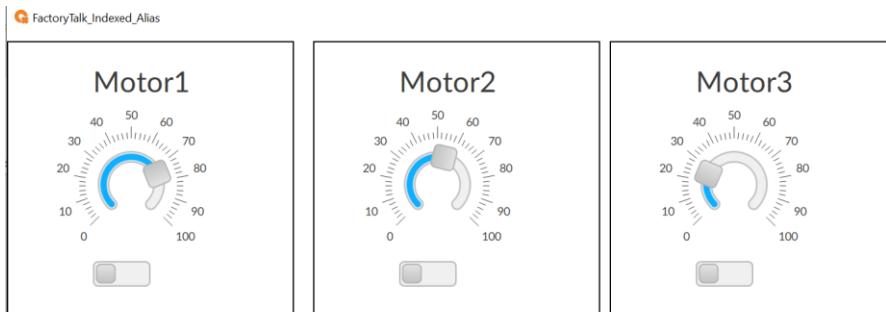
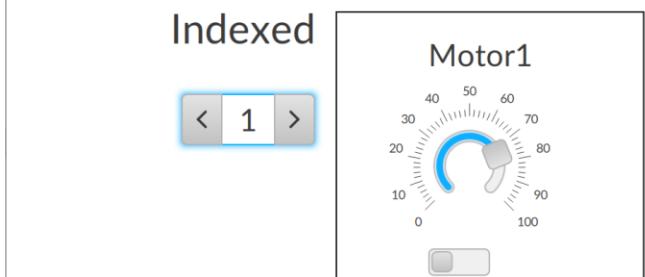
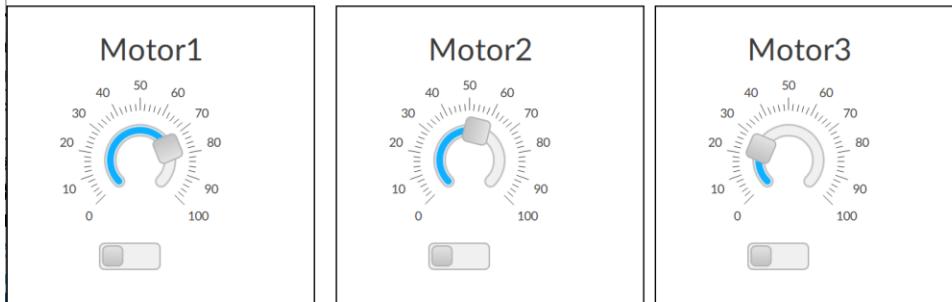
UI ▶ Panels ▶ Alias\_motores\_Panel ▶ Plantilla\_m

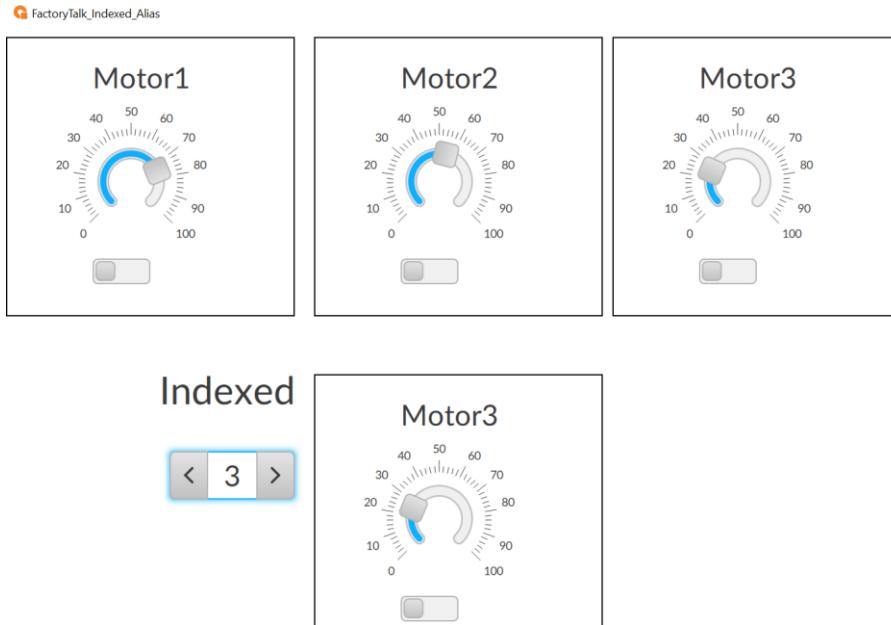


Voilà

Then you will have an indexed Alias

FactoryTalk\_Indexed\_Alias





You can find the code here

[https://github.com/xavierflorensa/FactoryTalk\\_Indexed\\_Alias.git](https://github.com/xavierflorensa/FactoryTalk_Indexed_Alias.git)

## 9. Communications with Micro8XX PLC's

From Optix version 1.03 and so on, the driver is enabled if you activate the features option on your FT Optix configuration, and restart FTOptix. So if you have this version, you can skip this section.

You can see it on this video

<https://www.youtube.com/watch?v=UKUBQ2-sX24>

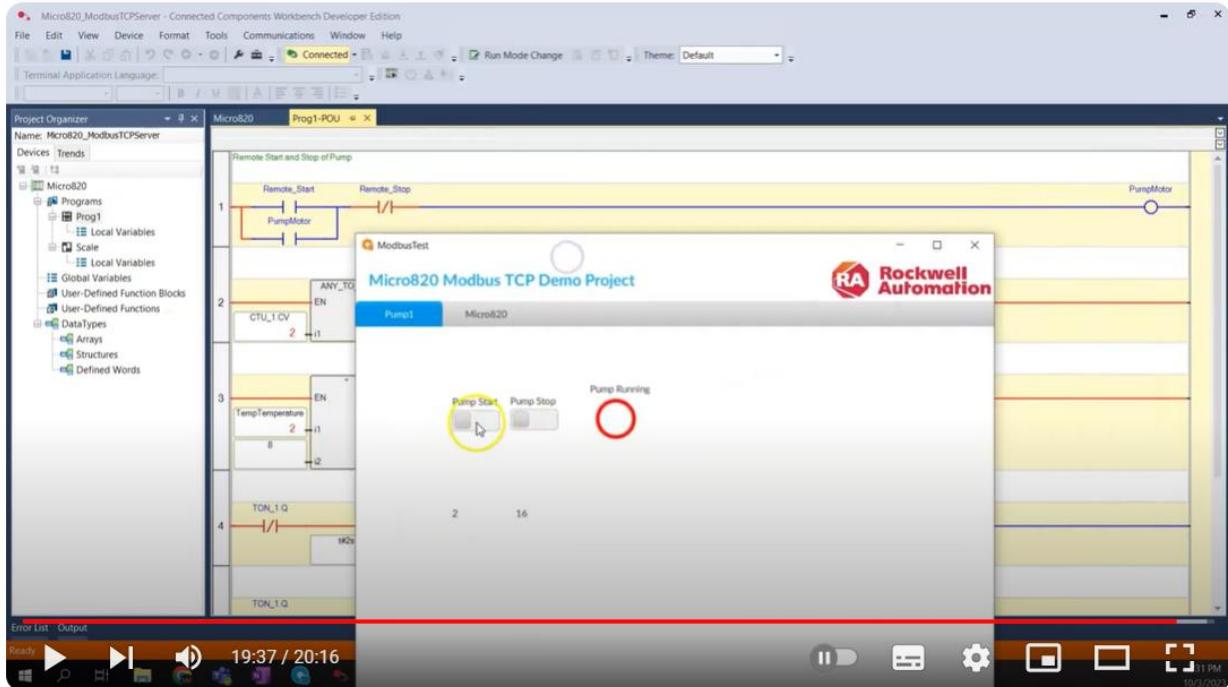
Before version 1.03

You can follow this video from Wayne Welk

<https://www.youtube.com/watch?v=k4QOC0uCeG0&t=848s>

And you can download the code from here

<https://github.com/wawelk/ModbusTest>



## Configuring FactoryTalk Optix Modbus TCP Communications with a Micro820 PLC



Document ID QA64805

Published Date 05/11/2023

FactoryTalk Optix: Modbus TCP communication with Micro850

Xavier Florencia

Automation Specialist

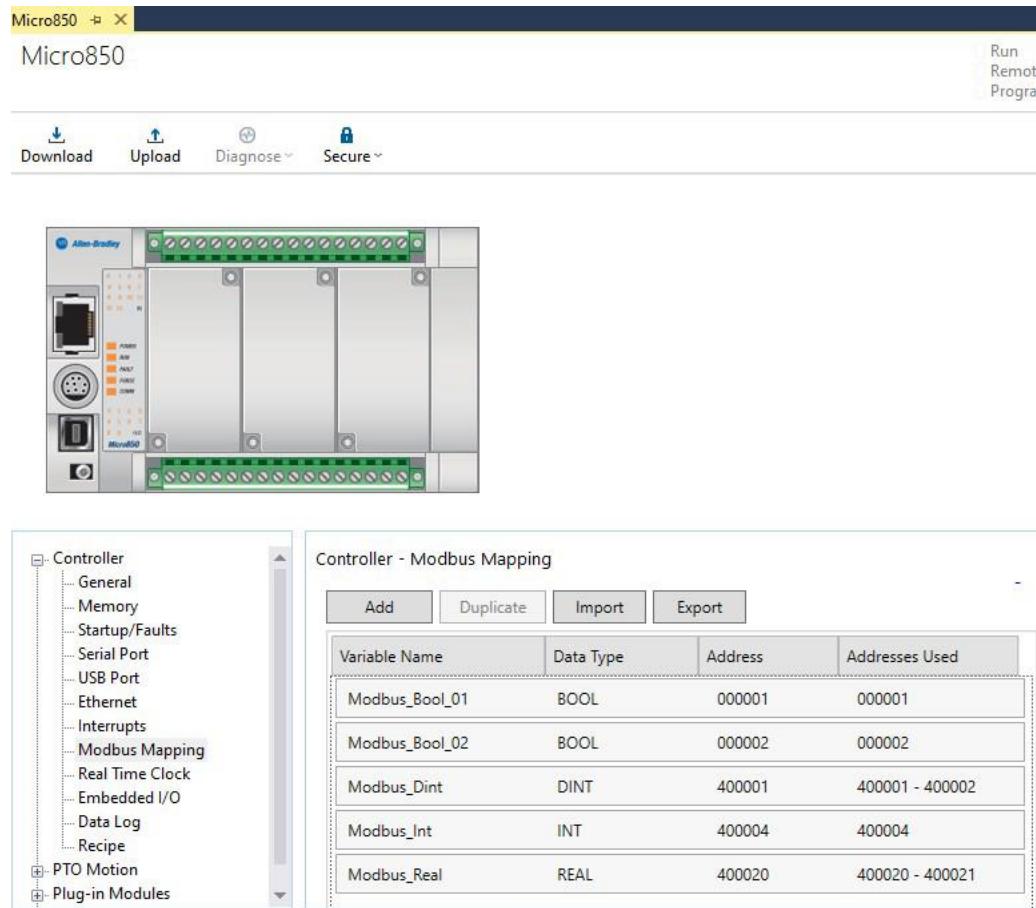
Risoul Ibérica SL

Micro800 is not compatible with RA EtherNet/IP Driver. But FactoryTalk Optix has ModbusTCP Driver that can be used to communicate with Micro800 controllers over Ethernet/IP. To configure Modbus TCP on the Micro800 controller:

In Project Organizer or Controller Organizer, double-click the controller to open the controller workspace.

Check if Modbus TCP is enabled in the controller properties, like in this technote: QA14133 - How to configure the Ethernet port for Modbus TCP in the Micro850

In the Controller tree, click Modbus Mapping --> Add to add tags that will be shared with FactoryTalk Optix.



Note: See this technote for Micro800 Modbus mapping limitations: QA49552 -Micro800: Modbus mapping limitation

Important: A valid numeric address is five or six characters in length with the first digit representing the register type. 0xxxxx (coils)

1xxxxx (discrete inputs)

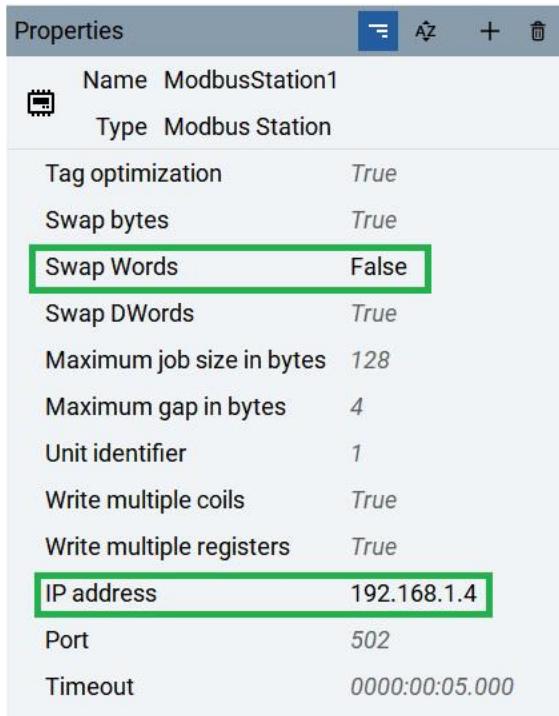
3xxxxx (input registers)

4xxxxx (holding registers)

To configure Modbus TCP in the FactoryTalk Optix:

Navigate to Project view and right-click CommDrivers. Select New to get list of supported communication protocols. Select Modbus Driver. The driver will be added and named automatically within your Project view.

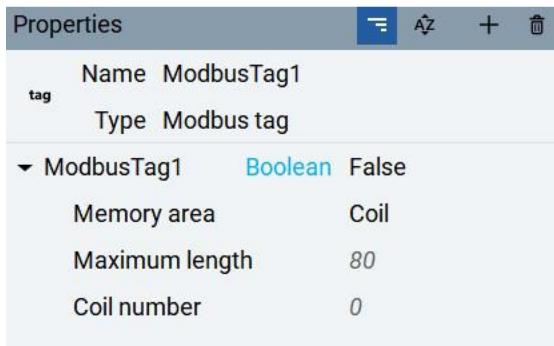
Right click Modbus Driver1 --> New --> Modbus Station to open Modbus Station1 properties. Change Swap Words to False and in the IP address field set the IP address of the controller.



Expand Modbus Station1, right-click Tags --> New --> Modbus Tag to add and configure tags mapped in the controller. Click on the blue tag data type to match it with the controller's mapped tag. Important: Some tag data types are different in Connected Component Workbench and FactoryTalk Optix. For example INT is Int16, DINT is Int32, REAL isFloat.

Memory area - select the correct register type of the mapped tag.

Coil number - set address of the mapped tag. Important: Connected Component Workbench is starting addressing at 1, while FactoryTalk Optix is starting addressing at 0. For example if tag is mapped in Connected Component Workbench to 400001, in FactoryTalk Optix you need to enter Holding Register in the Memory area, and set Register number to 0



Red X wireframe on the object using modbus tag can be caused by:  
FactoryTalk Optix has not established communication with the controller,  
Tag data type in FactoryTalk Optix do not match the tag data type in the controller,Important: Some tag data types are different in Connected Component Workbench and FactoryTalk Optix. For example Int16 is INT, Int32 is DINT, Float is REAL.  
FactoryTalk Optix is checking different register type or different address than it is mapped to,Important: Connected Component Workbench is starting addressing at 1, while FactoryTalk Optix is starting addressing at 0. For example if tag is mapped in Connected Component Workbench to 400001, in FactoryTalk Optix you need to enter Holding Register in the Memory area, and set Register number to 0.

#### FTOptixRuntime



## 10. Configuring an application as an MQTT client

This example is using a cloud public broker like test.mosquitto.org  
The Optix application is both subscribed and publishing to the same topic  
Go to help

The screenshot shows a Windows desktop with a browser window open to the Rockwell Automation FactoryTalk Optix Studio Help page. The title bar includes icons for settings, help, user profile (Hi Xavier), and standard window controls. A dropdown menu labeled 'Help' is open, showing options like 'Properties' and 'About'. The main content area displays several sections: 'Getting started', 'Creating projects' (which is highlighted in yellow), 'Extending projects', 'Deploying projects', and 'FactoryTalk Optix Release Notes'. Each section has a brief description and a link. The browser's address bar shows the URL: C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/Index.html.

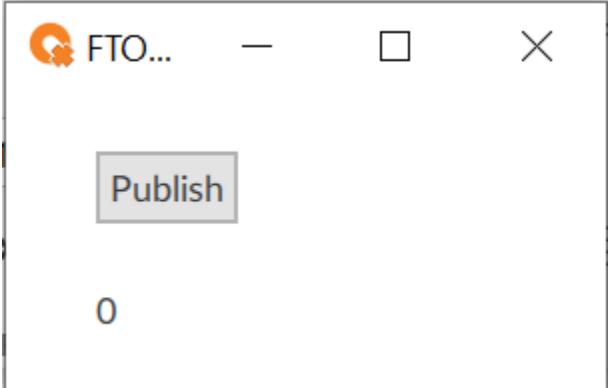
This screenshot shows the same help page as above, but the 'Creating projects' section is expanded. It contains a sub-section titled 'Configure an application as an MQTT client' which is also highlighted in yellow. This section provides a step-by-step guide with three sub-steps: 'Configure the message broker IP', 'Develop the publisher NetLogic and interface', and 'Develop the subscriber NetLogic and interface'. The browser's address bar shows the URL: C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/creating-projects/iot/mqtt-client/Configure-an-application-as-an-mqtt-client.html.

This help file is old and you will not find unless you use an old version of FT Optix like version 1.2 or similar.

<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/creating-projects/iot/mqtt-client/Configure-an-application-as-an-mqtt-client.html>

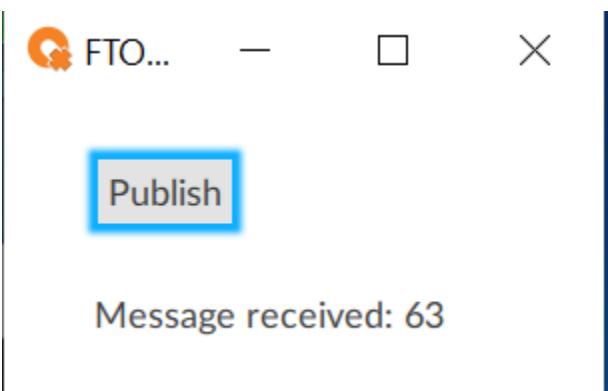
**Tip:** You can download a sample project from:  
[https://github.com/xavierflorensa/MQTTCClient\\_FTOptix\\_help](https://github.com/xavierflorensa/MQTTCClient_FTOptix_help)

Let's try opening the example



Click on "Publish"

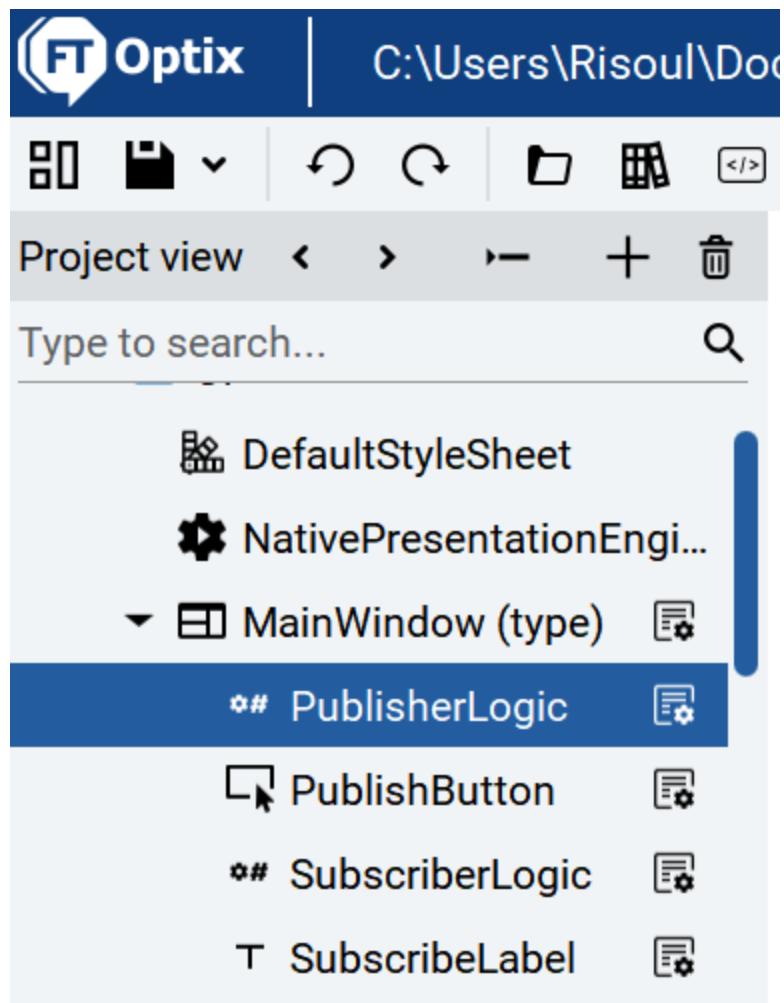
It works



But let's look at the details

Like PublisherLogic

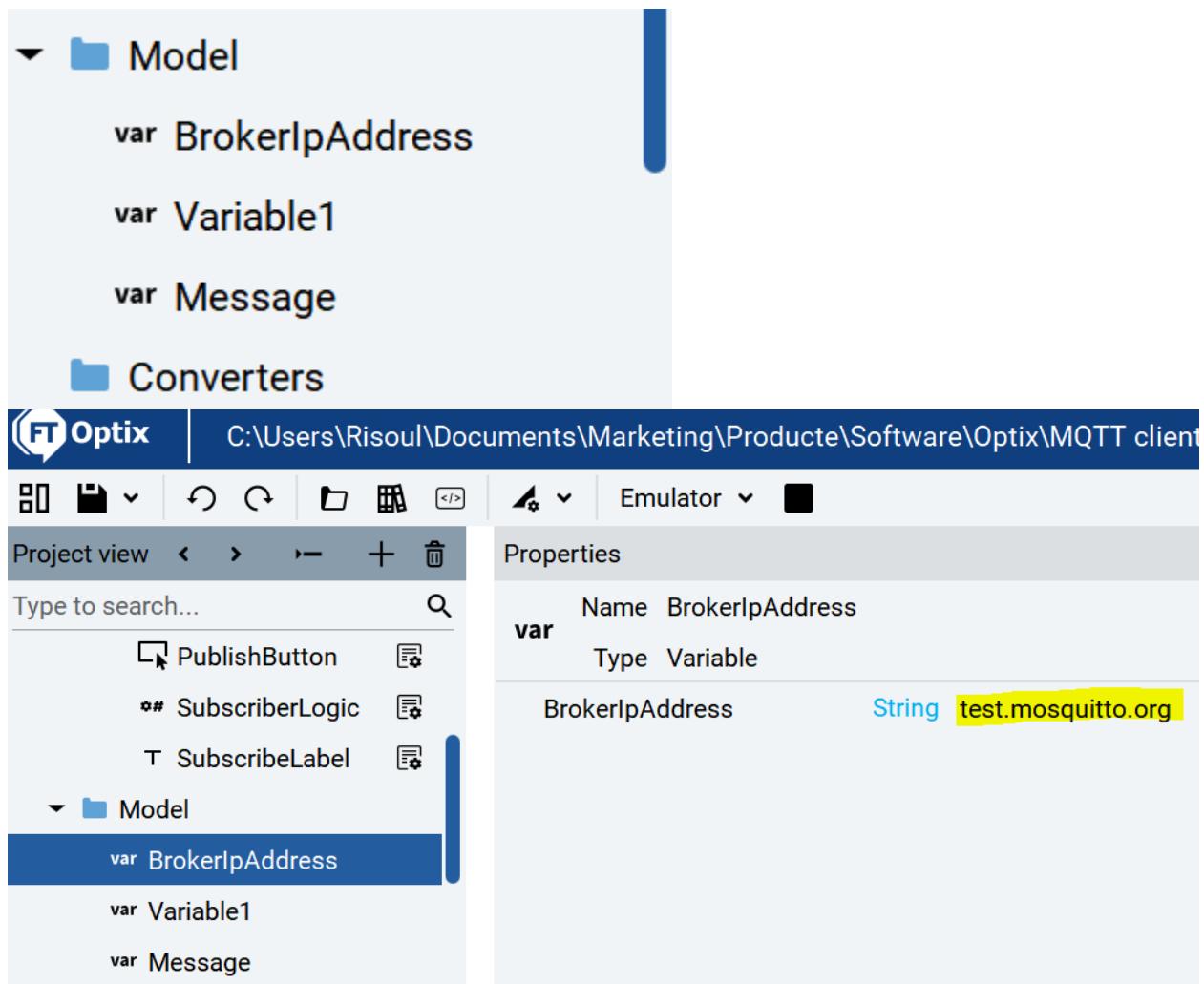
Doubleclick on PublisherLogic

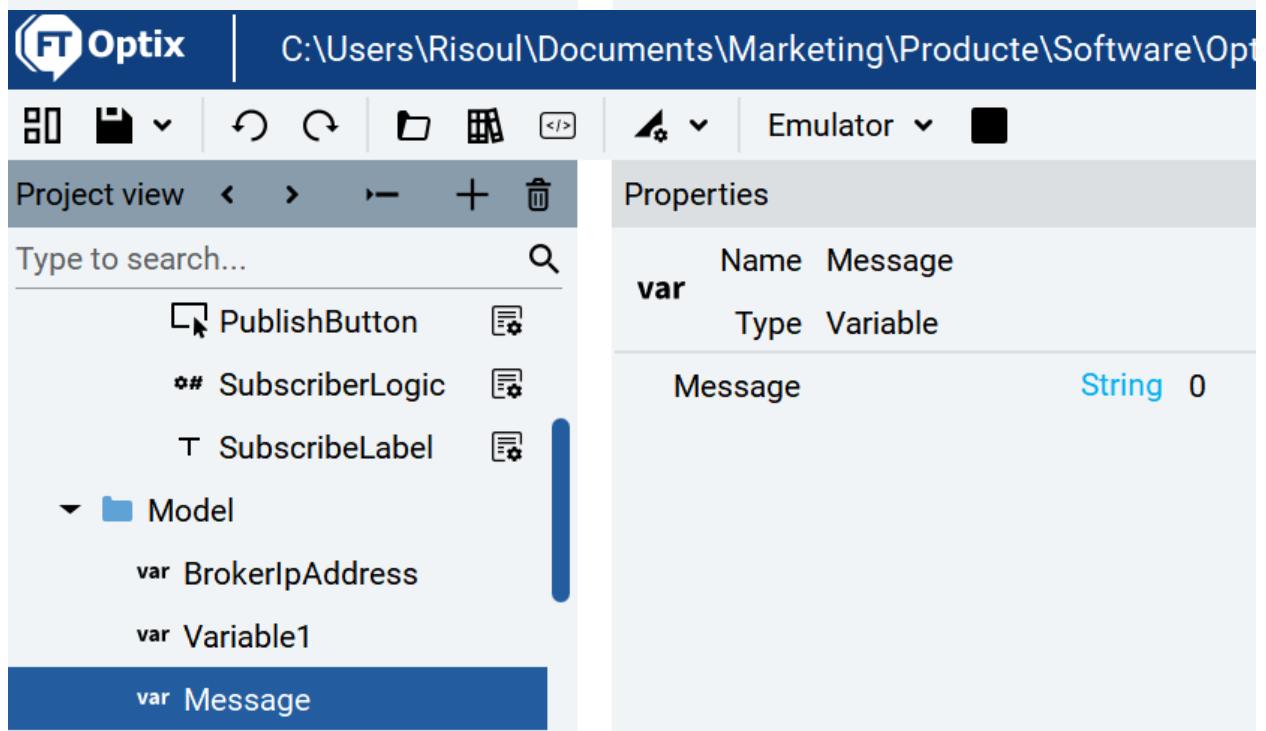
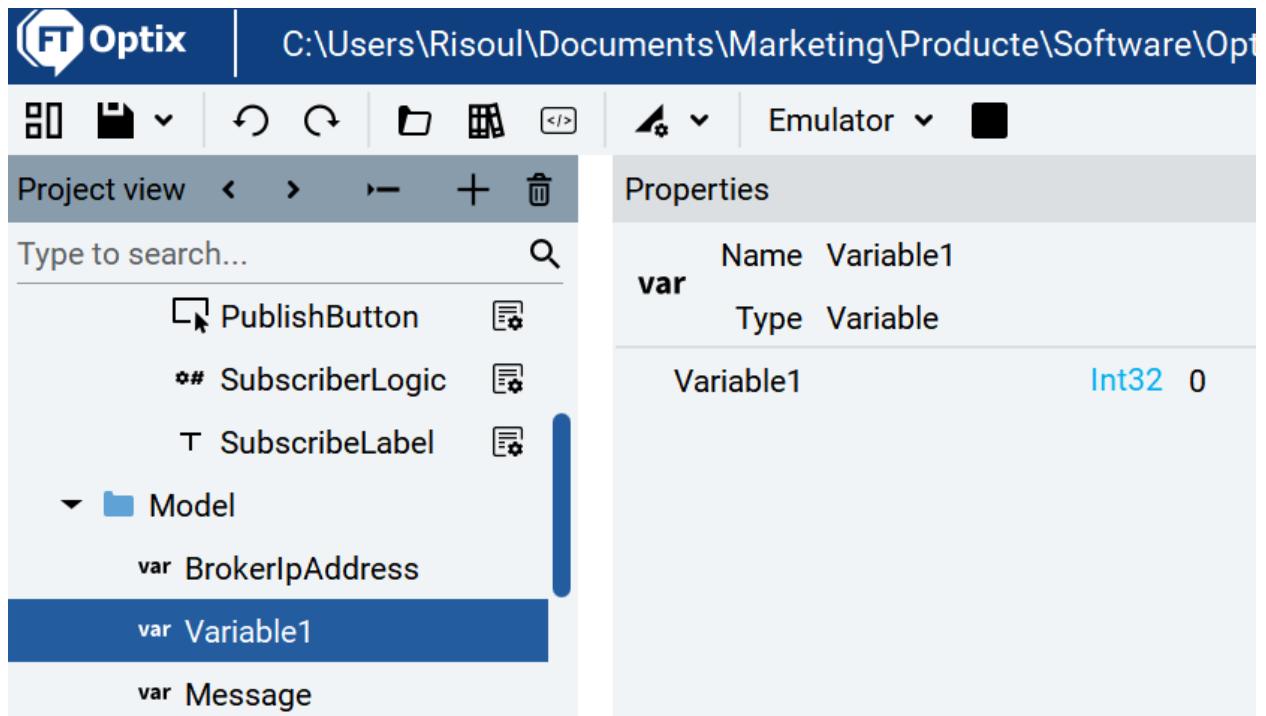


Visual Studio code opens

```
NetSolution > C# PublisherLogic.cs
1  #region StandardUsing
2  using System;
3  using FTOptix.CoreBase;
4  using FTOptix.HMIPProject;
5  using UAManagedCore;
6  using OpcUa = UAManagedCore.OpcUa;
7  using FTOptix.NetLogic;
8  using FTOptix.UI;
9  using FTOptix.OPCUAServer;
10 #endregion
11 using uPLibrary.Networking.M2Mqtt;
12 using uPLibrary.Networking.M2Mqtt.Messages;
13
14 public class PublisherLogic : BaseNetLogic
15 {
16     public override void Start()
17     {
18         var brokerIpAddressVariable = Project.Current.GetVariable("Model/BrokerIpAddress");
19
20         // Create a client connecting to the broker (default port is 1883)
21         publishClient = new MqttClient(brokerIpAddressVariable.Value);
22         // Connect to the broker
23         publishClient.Connect("FTOptixPublishClient");
24         // Assign a callback to be executed when a message is published to the broker
25         publishClient.MqttMsgPublished += PublishClientMqttMsgPublished;
26     }
27
28     public override void Stop()
29     {
30         publishClient.Disconnect();
31         publishClient.MqttMsgPublished -= PublishClientMqttMsgPublished;
32     }
33
34     private void PublishClientMqttMsgPublished(object sender, MqttMsgPublishedEventArgs e)
35     {
36         Log.Info("Message " + e.MessageId + " - published = " + e.IsPublished);
37     }
38
39     [ExportMethod]
40     public void PublishMessage()
41     {
42         var variable1 = Project.Current.GetVariable("Model/Variable1");
43         variable1.Value = new Random().Next(0, 101);
44
45         // Publish a message
46         ushort msgId = publishClient.Publish("/my_topic", // topic
47                                         System.Text.Encoding.UTF8.GetBytes(((int)variable1.Value).ToString()), // message body
48                                         MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, // QoS level
49                                         false); // retained
50     }
51
52     private MqttClient publishClient;
53 }
```

Look at BrokerIpAddress under UI





So we are publishing a random value to mosquito on the cloud

```

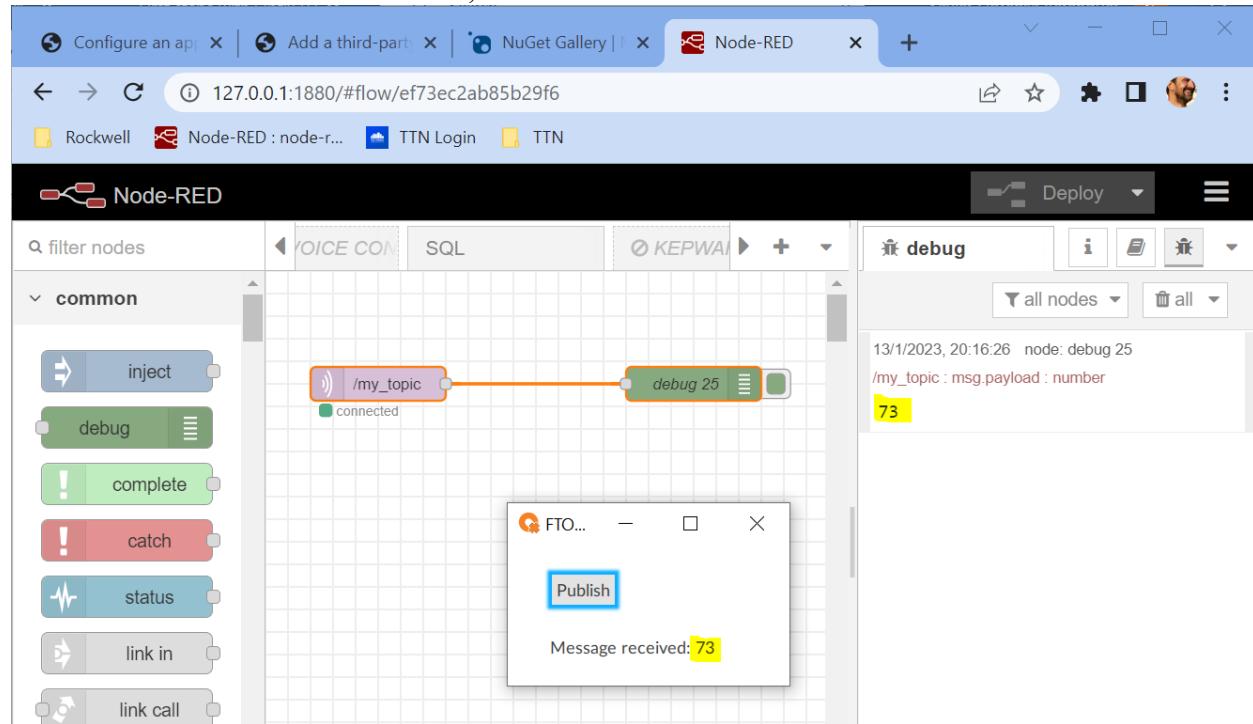
[ExportMethod]
public void PublishMessage()
{
    var variable1 = Project.Current.GetVariable("Model/Variable1");
    variable1.Value = new Random().Next(0, 101);

    // Publish a message
    ushort msgId = publishClient.Publish("/my_topic", // topic
        System.Text.Encoding.UTF8.GetBytes((int)variable1.Value).ToString(), // message body
        MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, // QoS level
        false); // retained
}

```

We are publishing to Topic /my\_topic

Let's check it with Node-RED. Yes, if we click on "Publish" the we receive the data



## Edit mqtt in node

DeleteCancelDone

### Properties



Server



Action



Topic

QoS

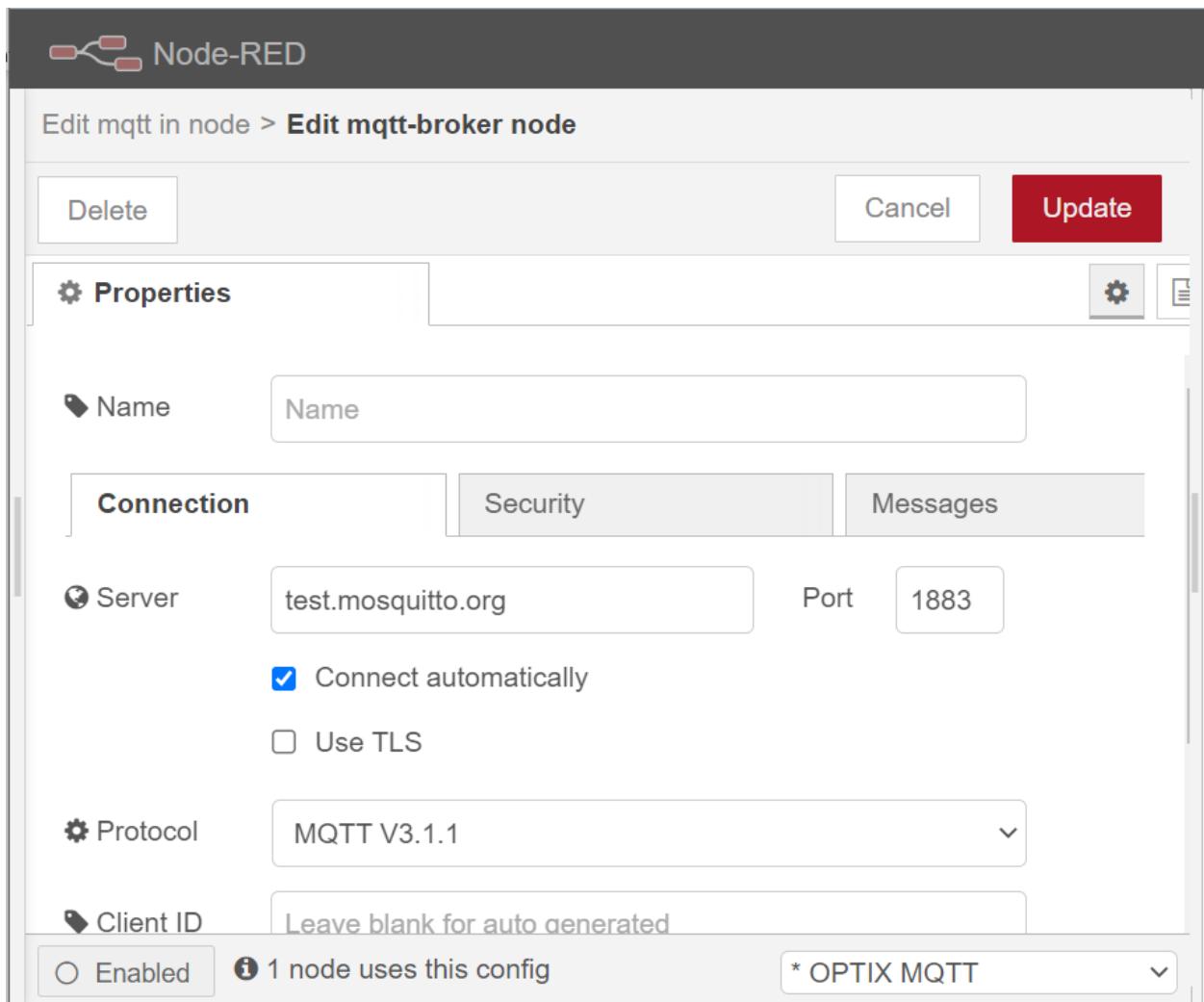


Output

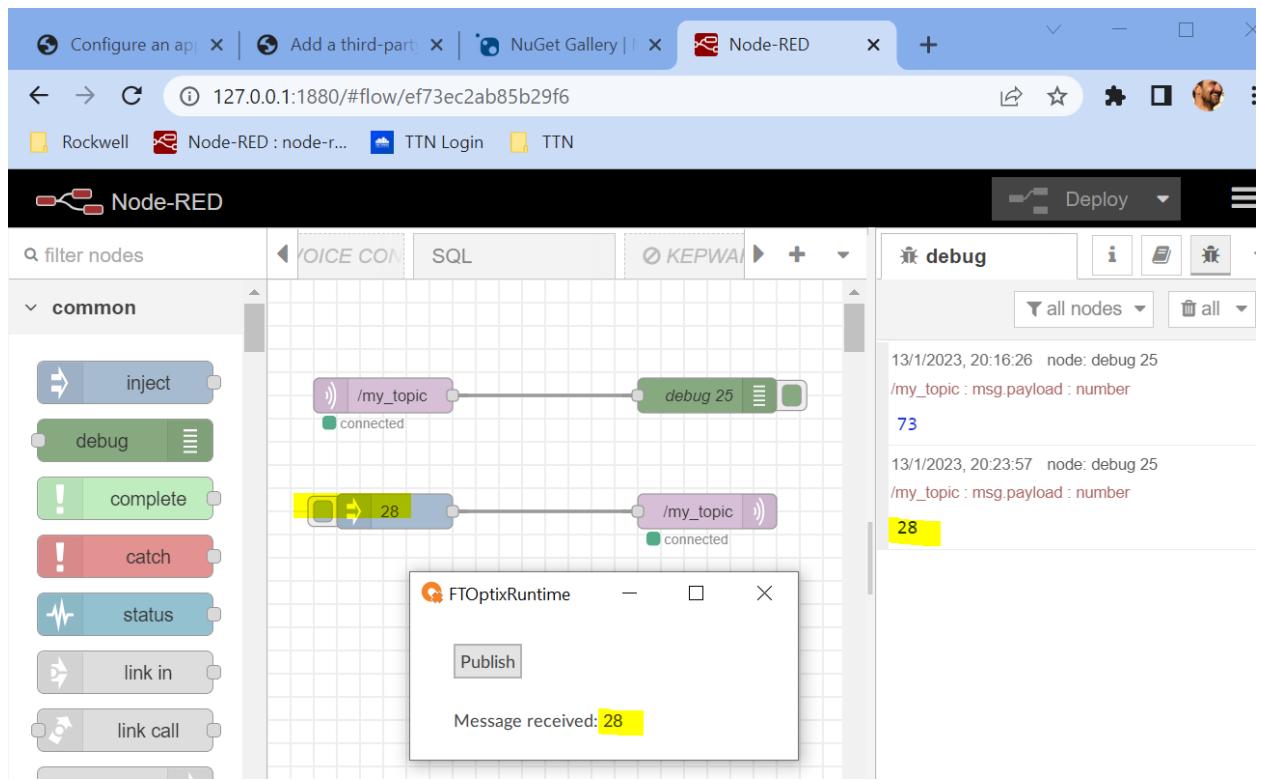


Name

Enabled



Let's test the subscription from Optix application  
Yes, if we inject on Node-RED, the Optix Application gets the value



**Edit mqtt out node**

Delete      Cancel      Done

**Properties**

Server: test.mosquitto.org:1883      Edit icon

Topic: /my\_topic

QoS:      Retain:      Edit icon

Name: Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

Enabled

FTO...   -     

  (highlighted)

Message received: 49

49

Next step is to use PLC data

The screenshot shows a configuration dialog for an MQTT output node. At the top, there are buttons for Delete, Cancel, and Done. Below that is a section titled "Properties" with icons for gear, file, and edit. The configuration fields are: Server (test.mosquitto.org:1883), Topic (/my\_topic), QoS (dropdown), Retain (dropdown), and Name (Name). A tip message at the bottom says: "Tip: Leave topic, qos or retain blank if you want to set them via msg properties." Below the configuration is a preview window titled "FTO..." with a "Publish" button highlighted in blue. It shows a message received with the value "49".

## 11. PLC data and MQTT

This is what we will do on this example



You can find the code here

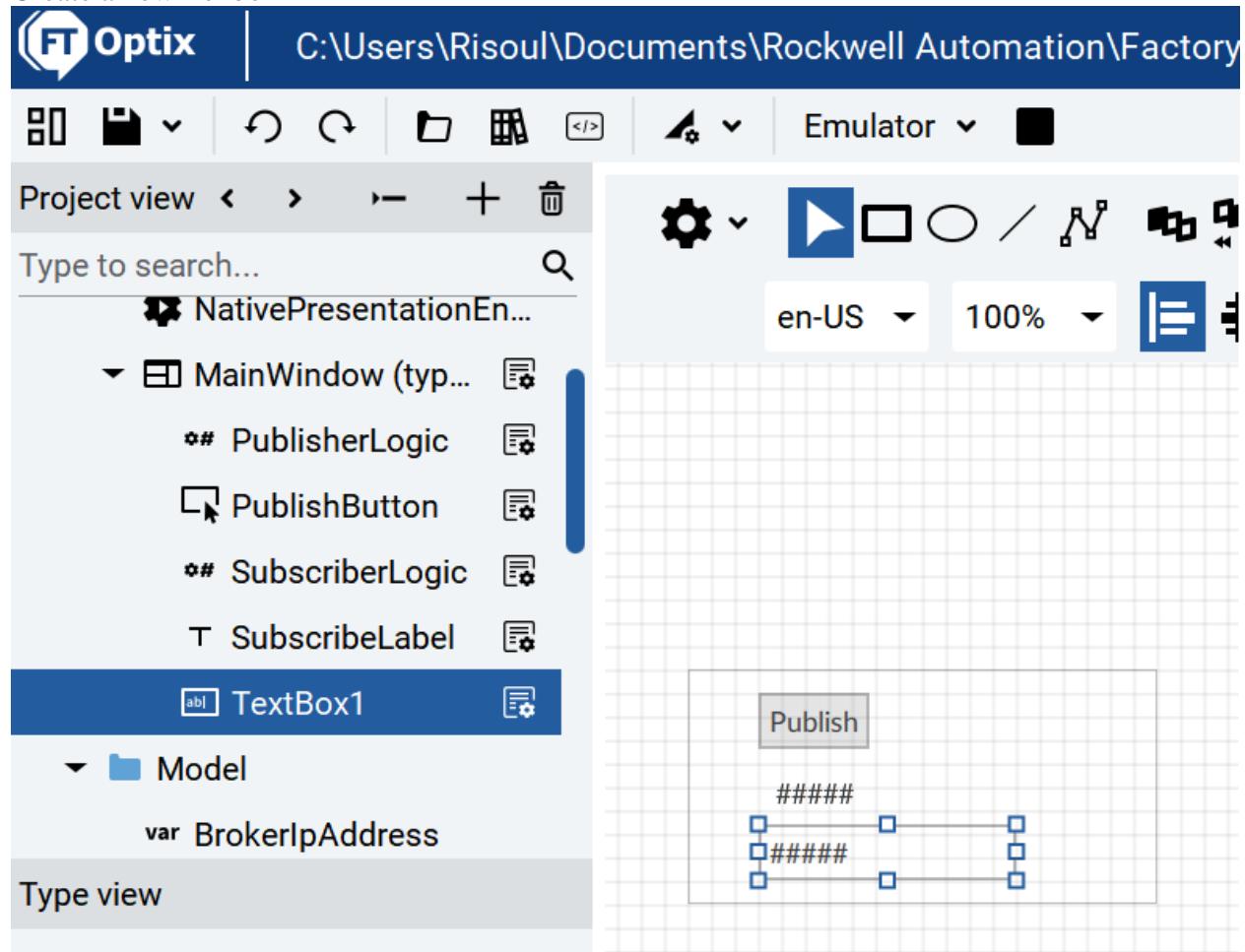
[https://github.com/xavierflorensa/PLC2MQTTClient\\_HiveMQ\\_v1\\_Git](https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git)

We will create an EtherNet/IP driver on same MQTT sample project

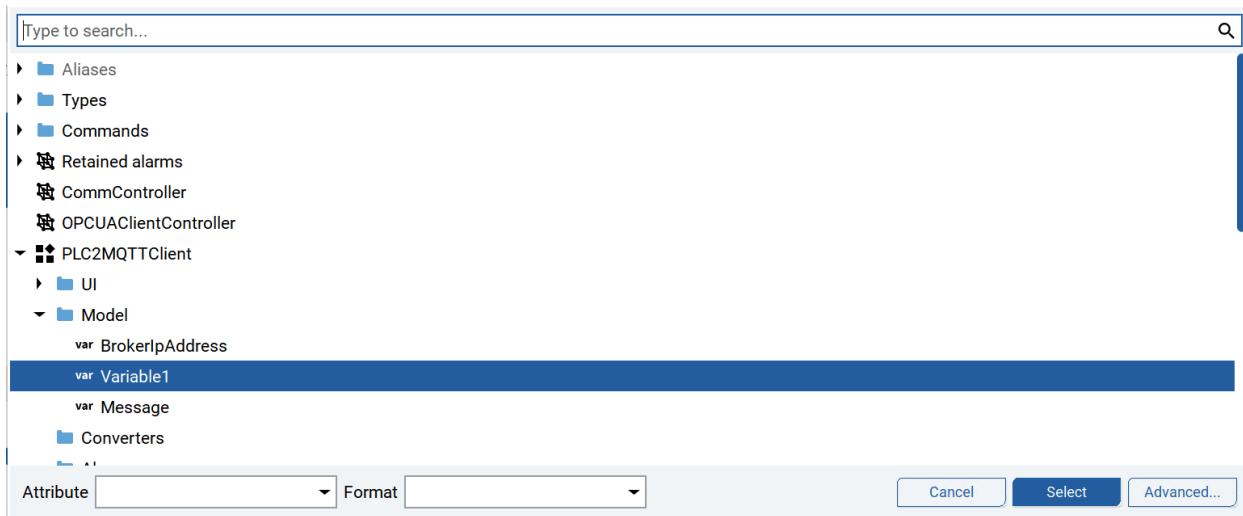
Let's take a look on how to work in our project with the variables provided by MQTT

Reading the subscribed variable

Create a new Texbox



On properties point to Model.variable1



Like this

The screenshot shows the 'Properties' panel of the FactoryTalk Studio interface. At the top, there are navigation icons: gear, question mark, sign in, minimize, maximize, and close. Below that is a toolbar with icons for list view, sort, add, and delete.

**Properties**

Name	Type	Value	Editor
TextBox1	Text box		
Height		Auto	
Left margin		30	
Top margin		66	
— Text and font			
Text		Variable1	
Content Type		Normal	
Text color		#000000	
Text horizontal alignment		Left aligned	
Events			
Modified text			

And on the other hand let's try to write to MQTT from PLC data

First let's get PLC data

<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/getting-started/quick-start/Import-tag-variables.html>

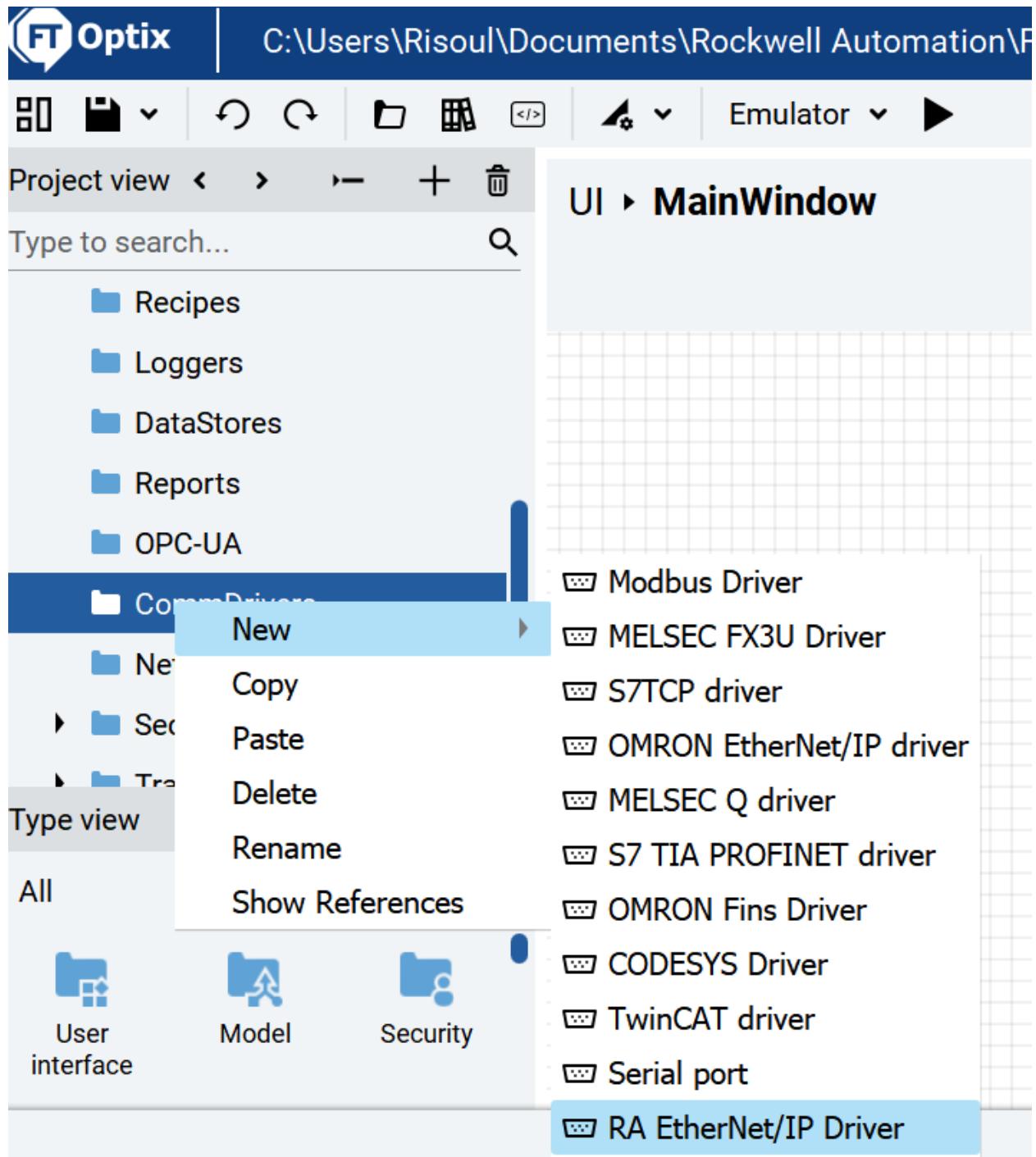
The screenshot shows a web browser window with the following details:

- Address Bar:** Archivo | C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/getting-started/quick-start/import-tag-variables.html
- Tab Bar:** Rockwell, Node-RED : node-r..., TTN Login, TTN
- Content Area:**
  - Left Sidebar:** A navigation tree with sections like Introduction, Basic concepts, Quick start: develop a sample project (selected), Configure and brand the main window, Configure panels, Configure dynamic graphic objects, Configure variables (selected), Configure alarms, Configure recipes, and Save and commit changes.
  - Right Content:**
    - Tip:** Instead of importing controller variables from a Logix controller, you can create variables manually. See Create variables.
    - To configure a communication driver for a different controller or learn more about the available communication drivers, see Communication driver.
    - Prerequisites:** In Logix Designer, download the LogixTags.ACD project to a physical Logix controller or an emulated FactoryTalk® Logix Echo™ controller. Set the controller in the run mode. For more information, see the Logix Designer online help.
    - To import controller variables:**
      - From the FactoryTalk Optix Studio toolbar, select **Open dashboard page**.
      - In the central pane, select **I want to configure connected devices**.
      - Select **New stations**.
      - Select **RA EtherNet/IP Station** and select **Next**.
      - (optional) To import the controller tags in the online mode, in **Route**, enter **IP\_Address\Backplane\Chassis\_Slot\_Number** and select **Next**.
      - Select **Next**.
      - Fetch the controller tags:
        - To fetch the controller tags in the offline mode, select **Browse** and select the downloaded **LogixTags.ACD** file.
        - To fetch the controller tags in the online mode, select the **Offline/Online** toggle to change it to the **Online** position.
      - Select all controller tags and select **Next**.

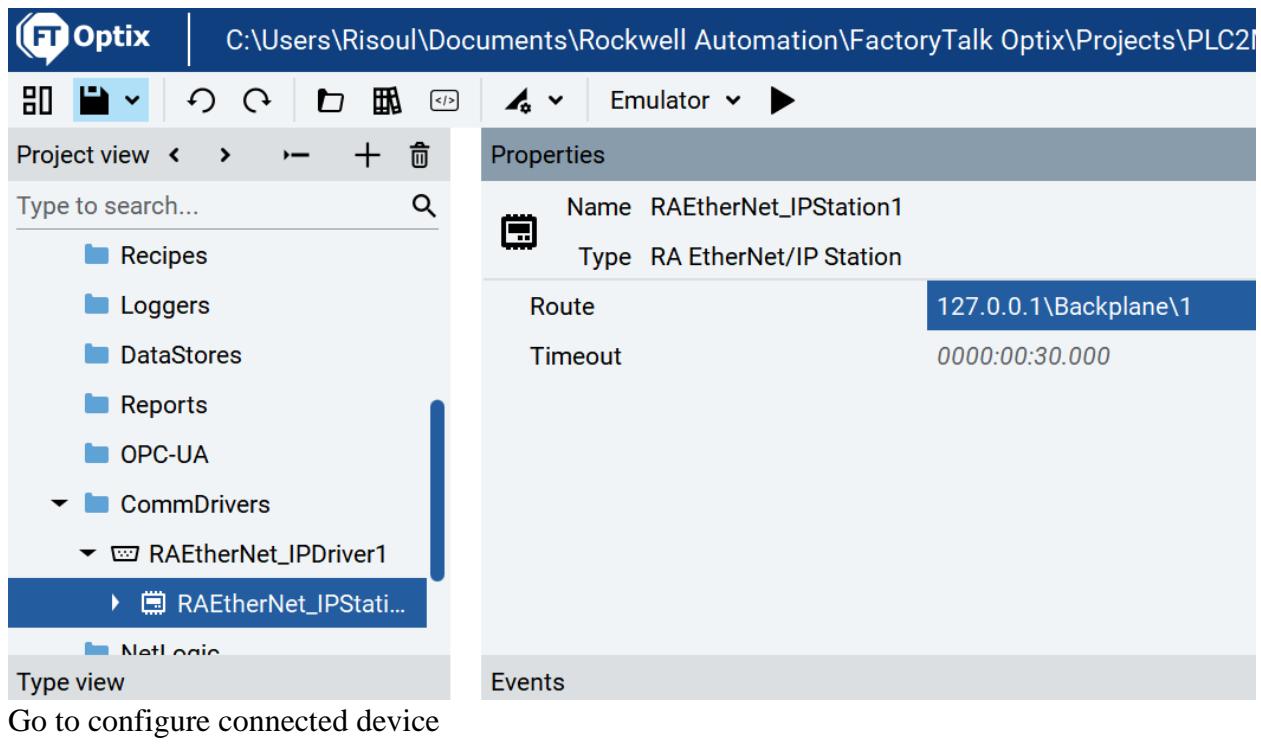
Figure: Selected tags in the online mode

Configure communication driver

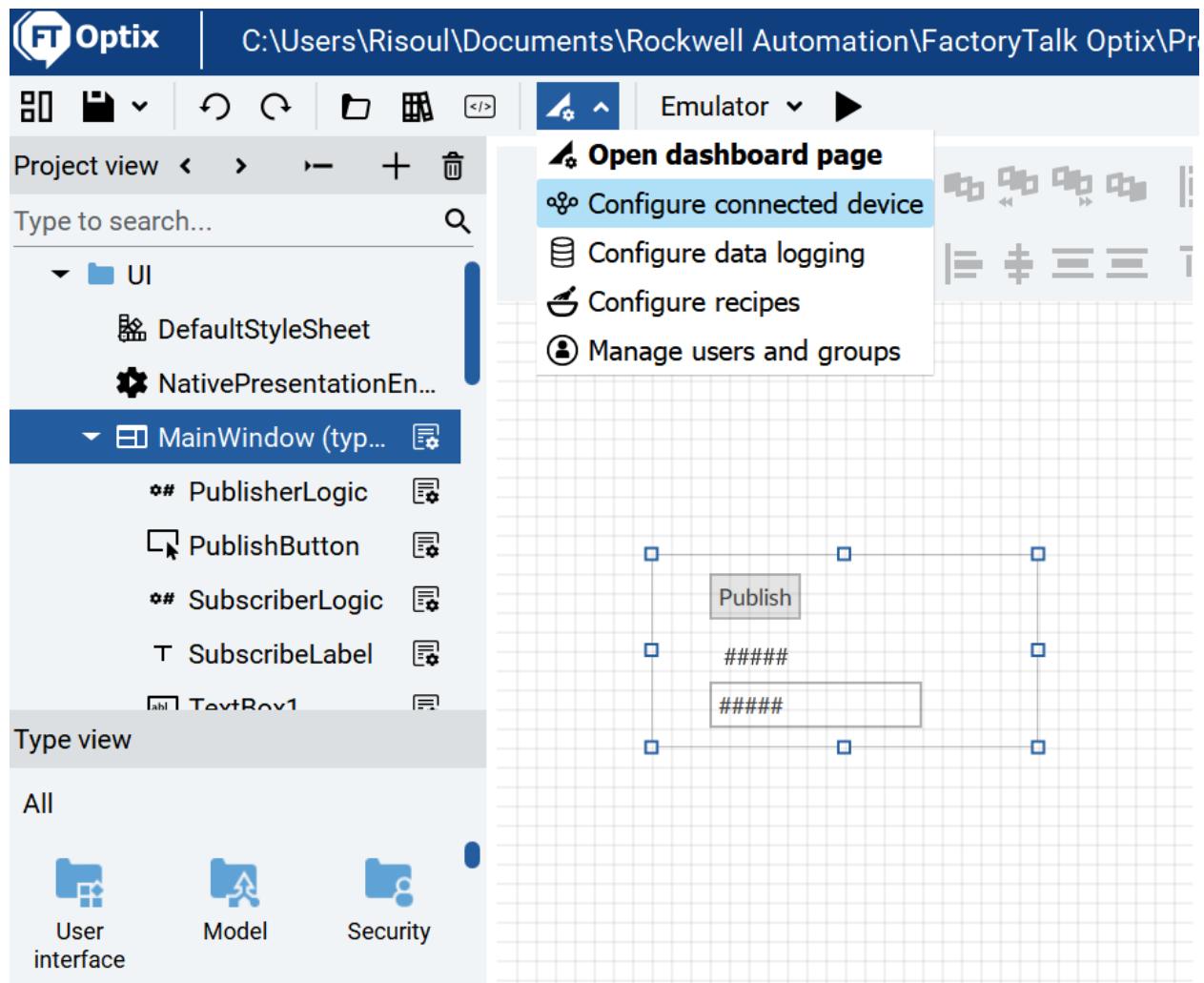
First of all create a communications driver

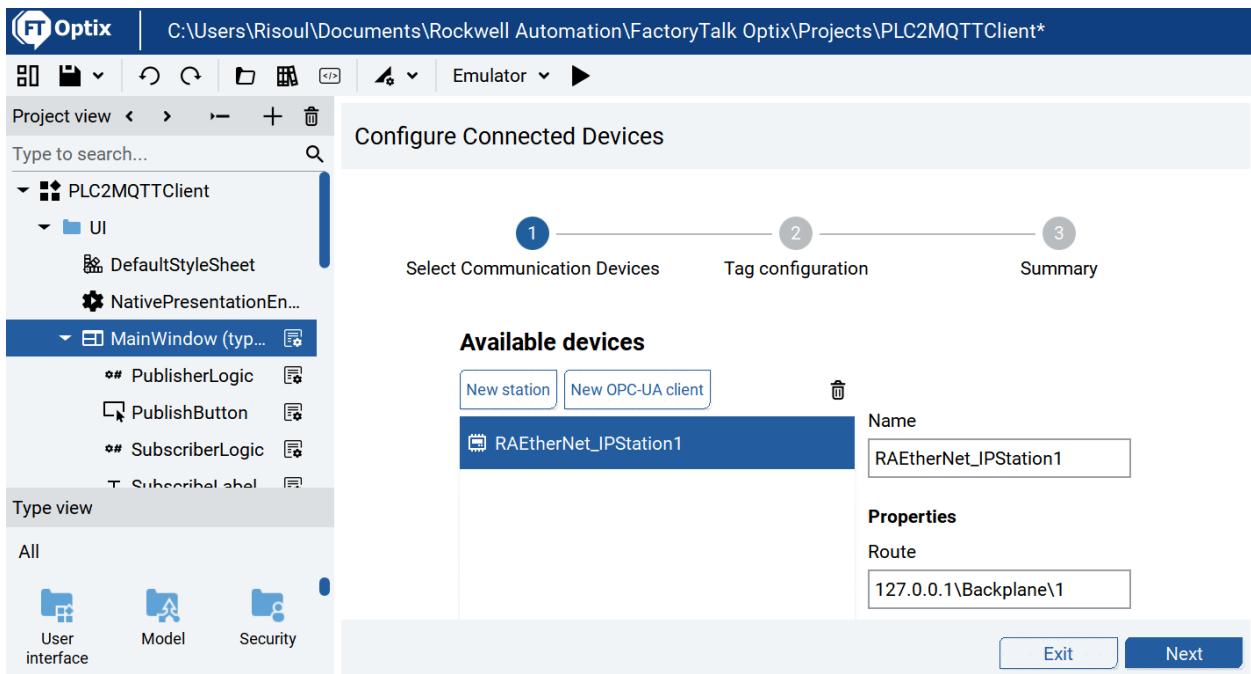


Add a new station

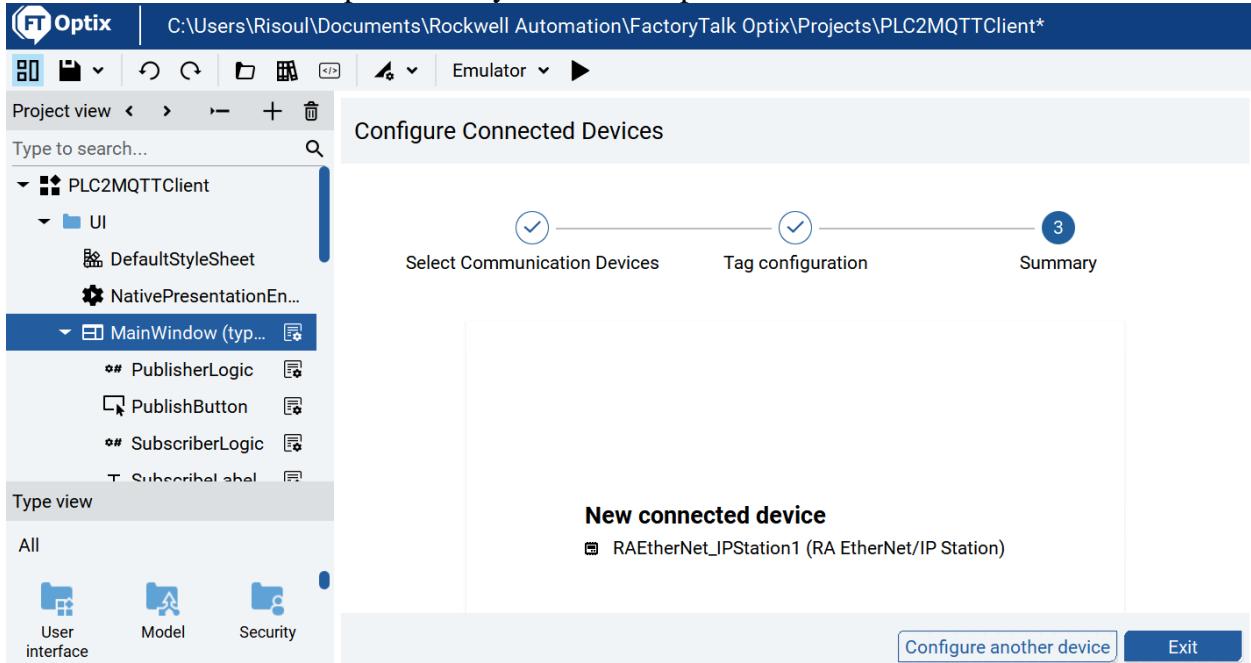


Go to configure connected device

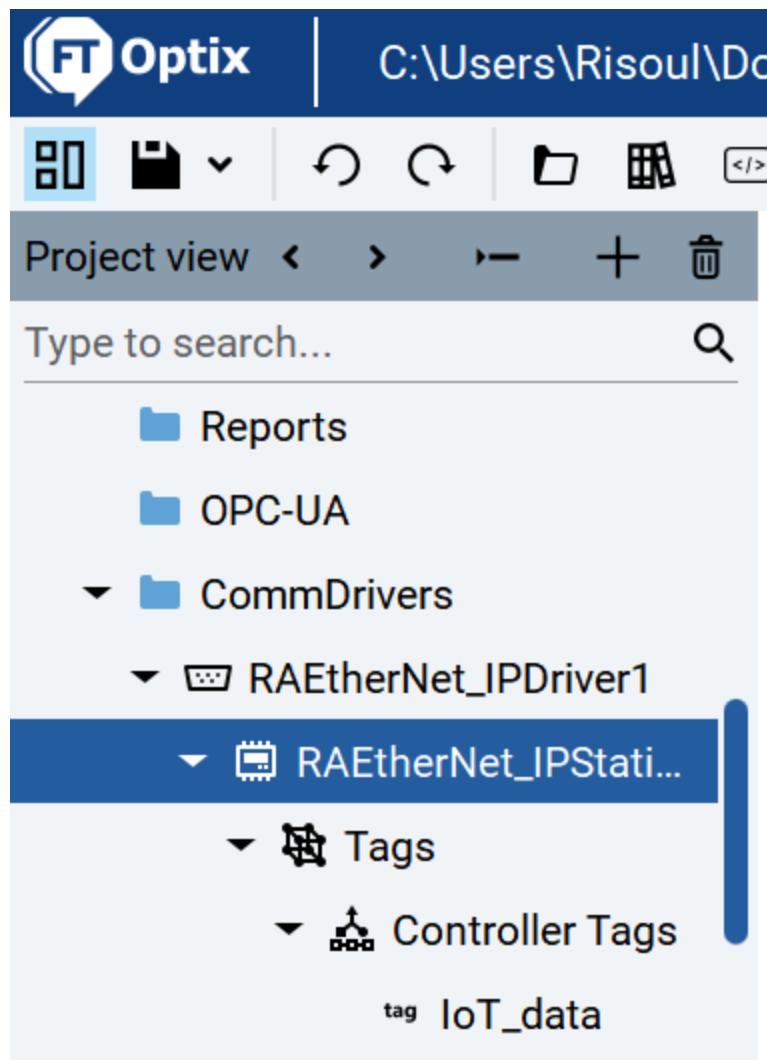




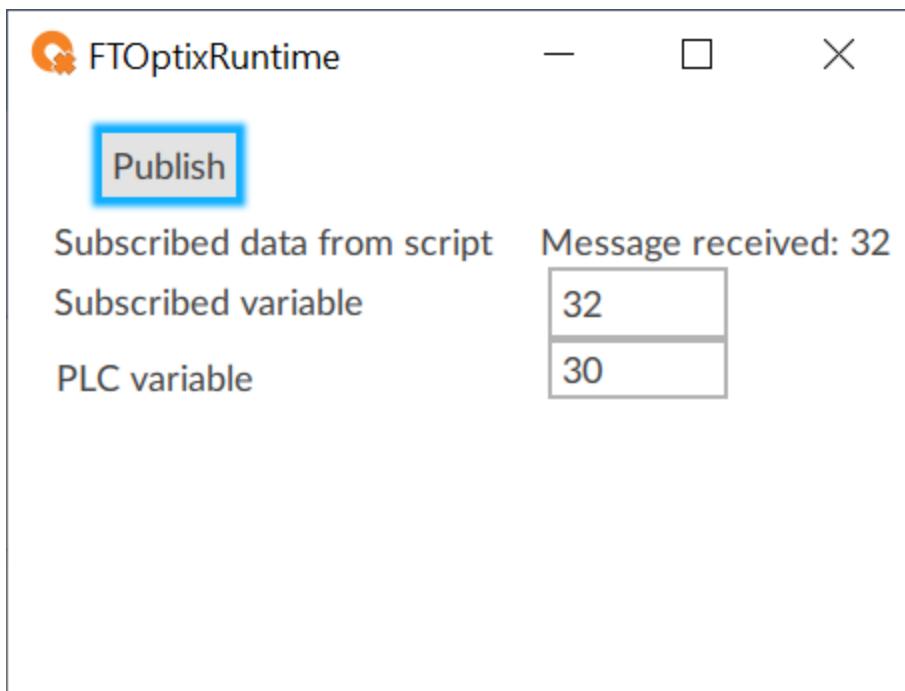
Continue as we did on chapter 2 until you reach this point



Verify that the Tag is there



Now let's display the PLC data on a new text label



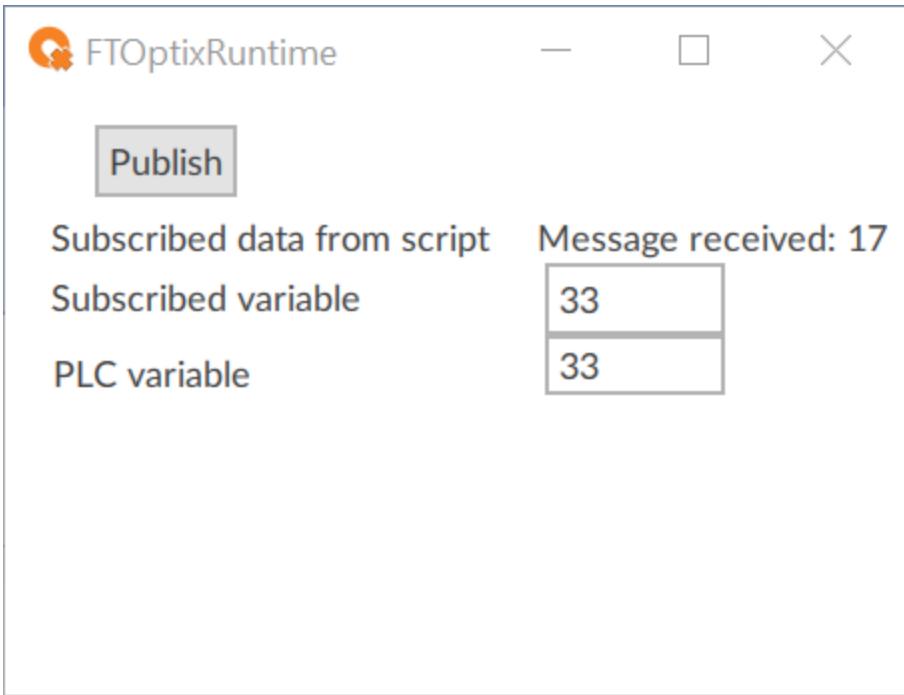
Now let's try to publish the data from PLC variable

We did a try that is publishing PLC data to Mosquitto, without writing any script, just with dynamic links:

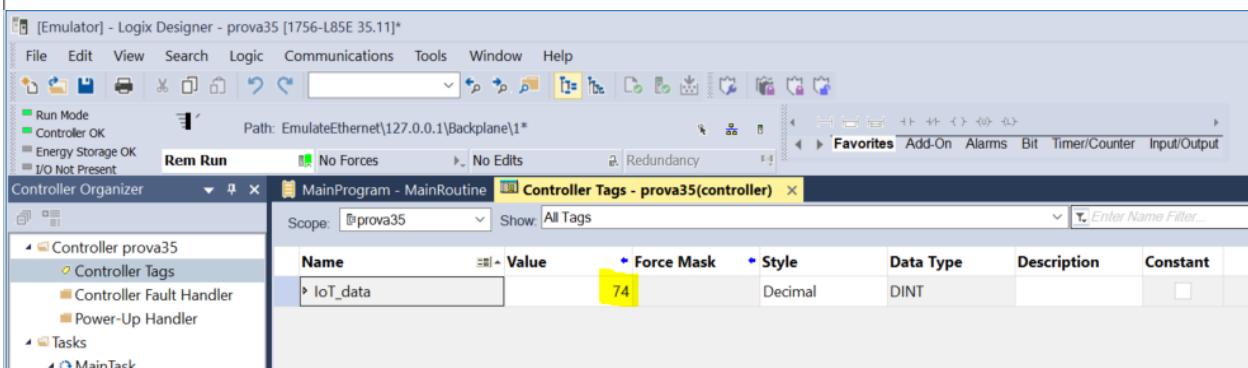
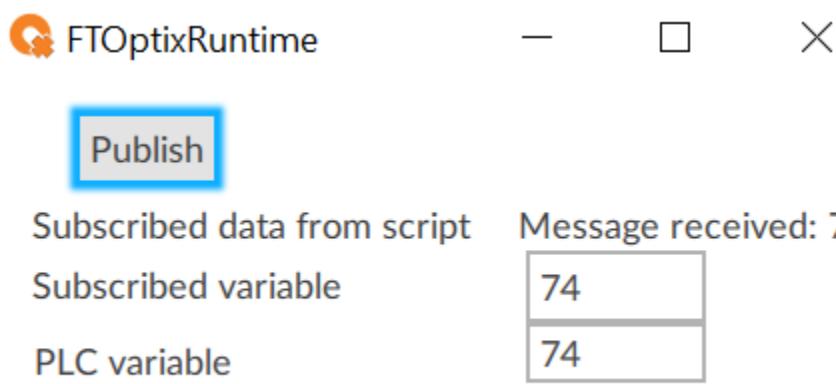
#### Modify the value on the PLC

Name	Value	Force Mask	Style	Data Type
IoT_data	33		Decimal	DINT

Then look at the Optix application



We wanted just to write on the PLC, but on the other hand if we click the button publish, then we are writing a random value on the PLC!!!



This is how to make this link

Just go to the Variable1 properties under model

Project view

Type to search...

- PLC2MQTTClient
  - UI
  - Model
    - var BrokerIpAddress
    - var Variable1**
    - var Message

Properties

	Name	Variable1
var	Name	Variable1
	Type	Variable
Variable1	Int32	

Then edit the link

Like this

Type to search...

- Model
  - Converters
  - Alarms
  - Recipes
  - Loggers
  - DataStores
  - Reports
  - OPC-UA
  - CommDrivers
    - RAEtherNet\_IPDriver1
      - RAEtherNet\_IPStation1
        - Tags
        - Controller Tags
          - IOT\_data

Attribute

Select Advanced...

You will get this

Project view

Type to search...

- PLC2MQTTClient
  - UI
  - Model
    - BrokerIpAddress
    - Variable1**
    - Message
  - Converters

Properties

	Name	Variable1
var	Name	Variable1
	Type	Variable
Variable1	Int32	../../../../CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/Tags/Controller Tags/IOT_data

And that's all, you have the variables linked in both directions

But the problem when writing to the PLC is that the variable is random. Let's try to be our desired value entered by the operator on Optix.

Now we are able to write the desired value on the PLC as soon as we hit enter

FTOptixRuntime

Enter value to publish 15 Publish

Subscribed data from script Subscribed variable PLC variable

Message received: 93

[Emulator] - Logix Designer - prova35 [1756-L85E 35.11]\*

File Edit View Search Logic Communications Tools Window Help

Run Mode Controller OK Energy Storage OK I/O Not Present Rem Run No Forces No Edits Redundancy Favorites Add-O

Path: EmulateEthernet\127.0.0.1\Backplane\1\*

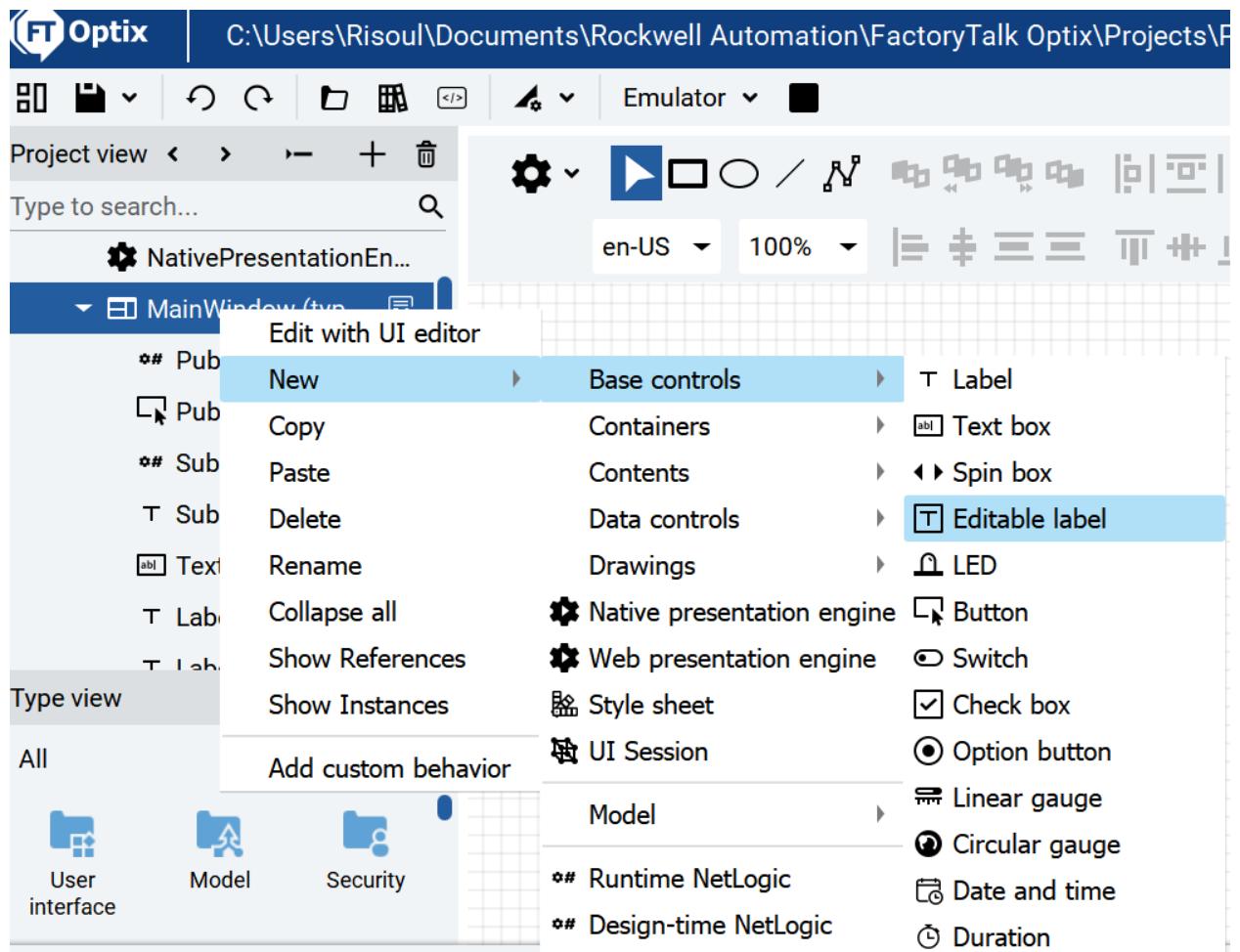
Controller Organizer MainProgram - MainRoutine Controller Tags - prova35(controller)

Scope: prova35 Show: All Tags

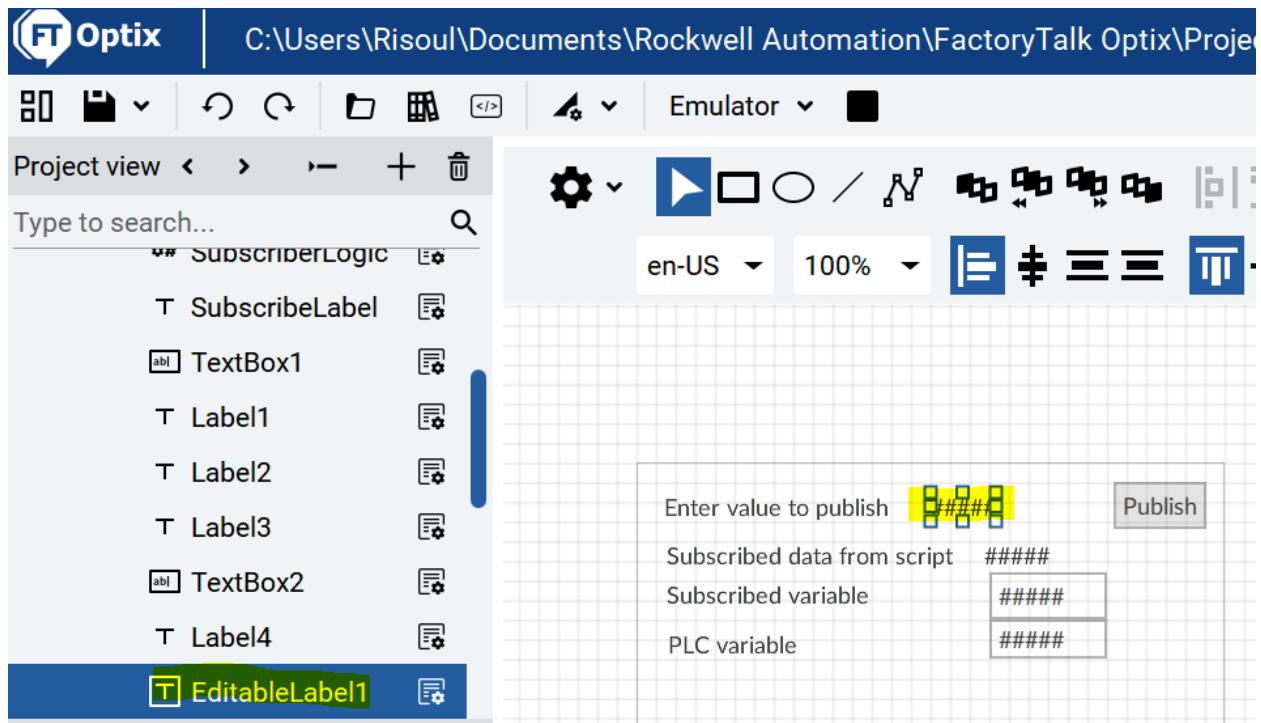
Name	Value	Force Mask	Style	Data Ty
IoT_data	15		Decimal	DINT

Let's see how to do it

Let's create an editable Label



Like this



And on the text property, link to the variable

Type to search...

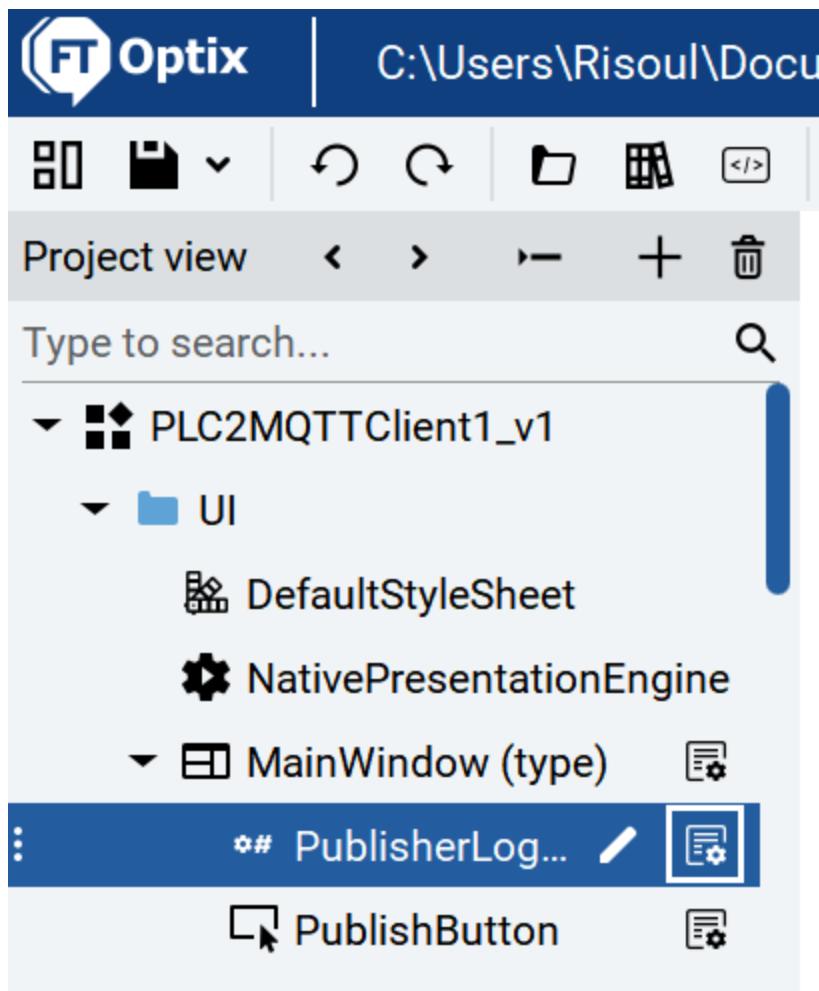
- ▶ Aliases
- ▶ Types
- ▶ Commands
- ▶ Retained alarms
- ▶ CommController
- ▶ OPCUAController
- ▶ PLC2MQTTClient1\_v1
  - ▶ UI
  - ▶ Model
    - var BrokerIpAddress
    - var Variable1**
    - var Message
    - Converters
    - ...

Attribute  Format  Cancel Select Advanced...

We are still writing a random variable with the publish button.

In order to cancel this we can try to comment this line on the code

Click on the code icon



And comment this line

Then save on Visual Studio Code

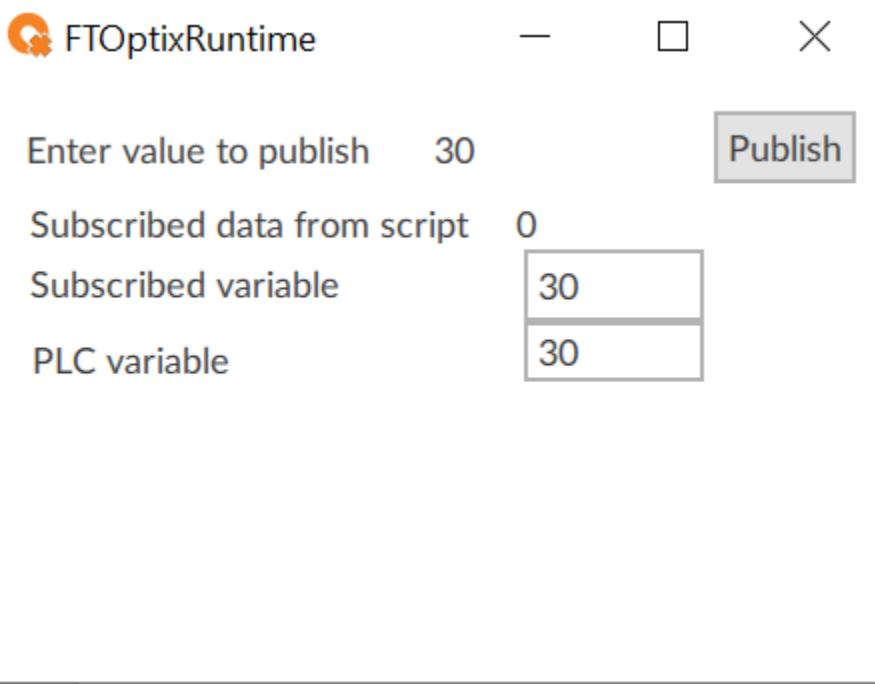
```

36     private void PublishClientMqttMsgPublished(object sender, MqttMsgPublishedEventArgs e)
37     {
38         Log.Info("Message " + e.MessageId + " - published = " + e.IsPublished);
39     }
40
41     [ExportMethod]
42     public void PublishMessage()
43     {
44         var variable1 = Project.Current.GetVariable("Model/Variable1");
45         //variable1.Value = new Random().Next(0, 101);
46
47         // Publish a message
48         ushort msgId = publishClient.Publish("/my_topic", // topic
49                                         System.Text.Encoding.UTF8.GetBytes(((int)variable1.Value).ToString()), // message body
50                                         MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, // QoS level
51                                         false); // retained
52     }
53
54     private MqttClient publishClient;
55
56 }
57

```

Now click on play and try  
No more random publishing.

But you do not need to click the publish button to write on the PLC.  
But you have to click on publish in order to send per MQTT



But we are not yet able to write on the PLC from an MQTT client like a mobile phone.  
We have to do two steps

First of all, modify on the subscriber code this line

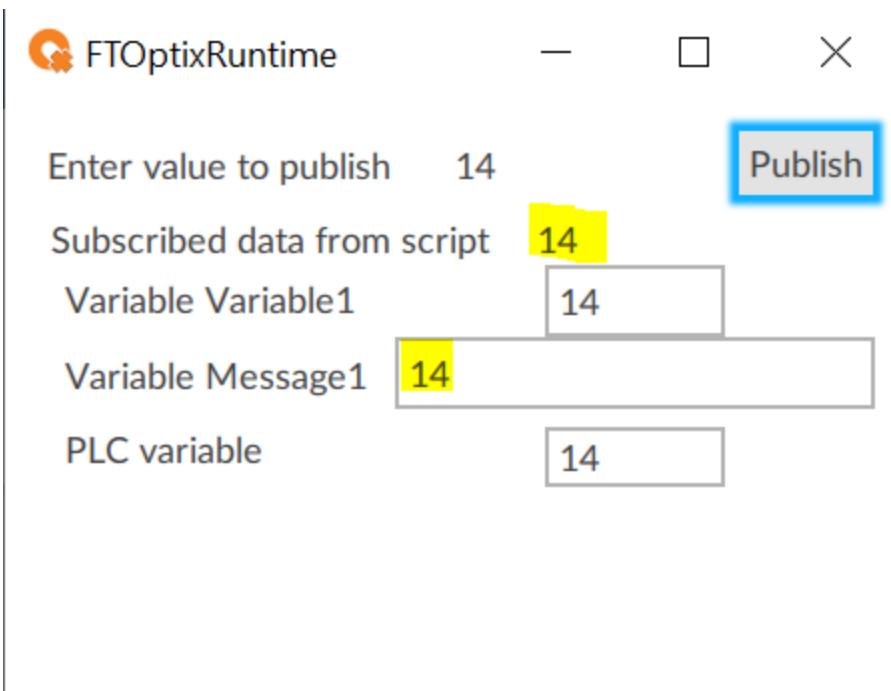
```
43     messageVariable.Value = "Message received: " + System.Text.Encoding.UTF8.GetString(e.Message);
```

And leaving like this

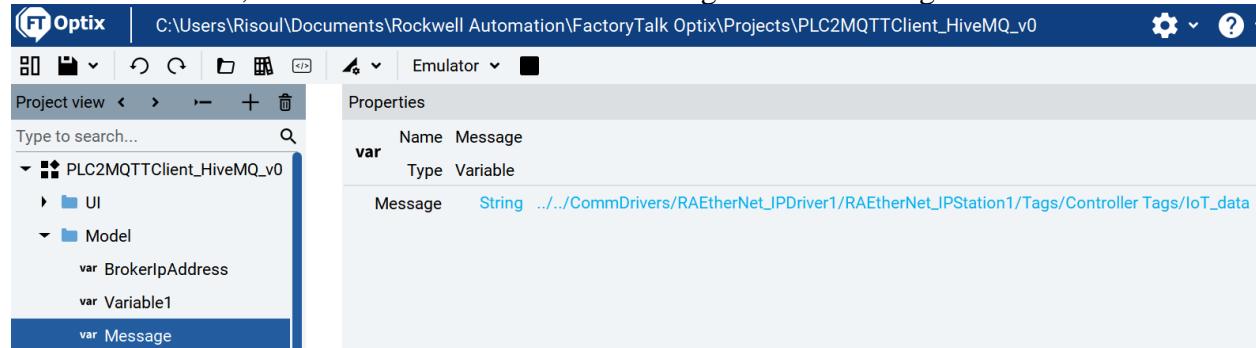
```
44     messageVariable.Value = System.Text.Encoding.UTF8.GetString(e.Message);
```

Then save on Visual Studio Code

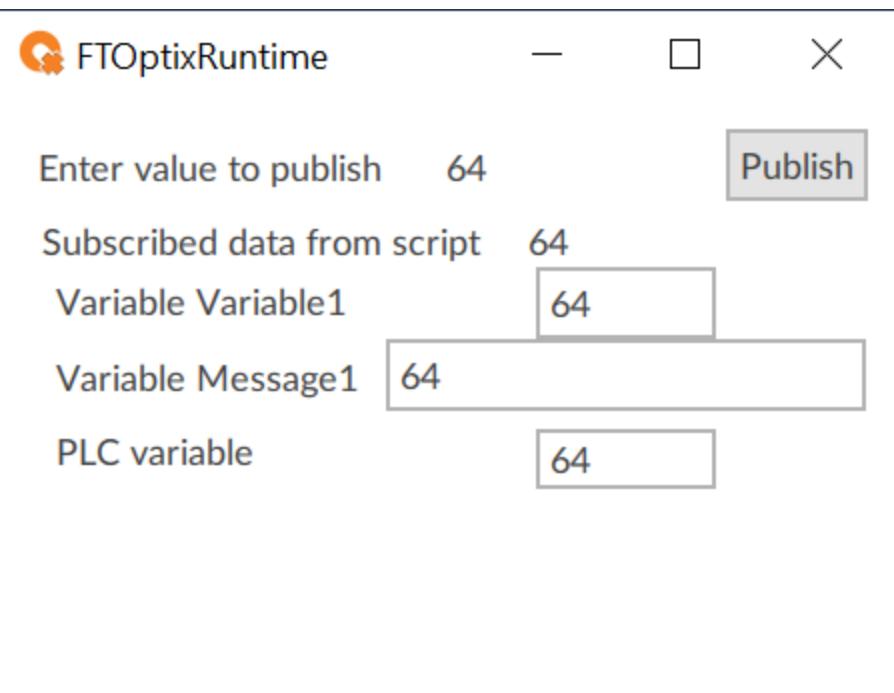
Now we do not have the string like before



On the other hand, we have to link the variable Message to the PLC Tag like this



Now we are writing on the PLC from a MQTT client like a Mobile Phone



[Emulator] - Logix Designer - prova35 [1756-L85E 35.11]

File Edit View Search Logic Communications Tools Window Help

Run Mode Controller OK Energy Storage OK I/O Not Present

Path: EmulateEthernet\127.0.0.1\Backplane\1\*

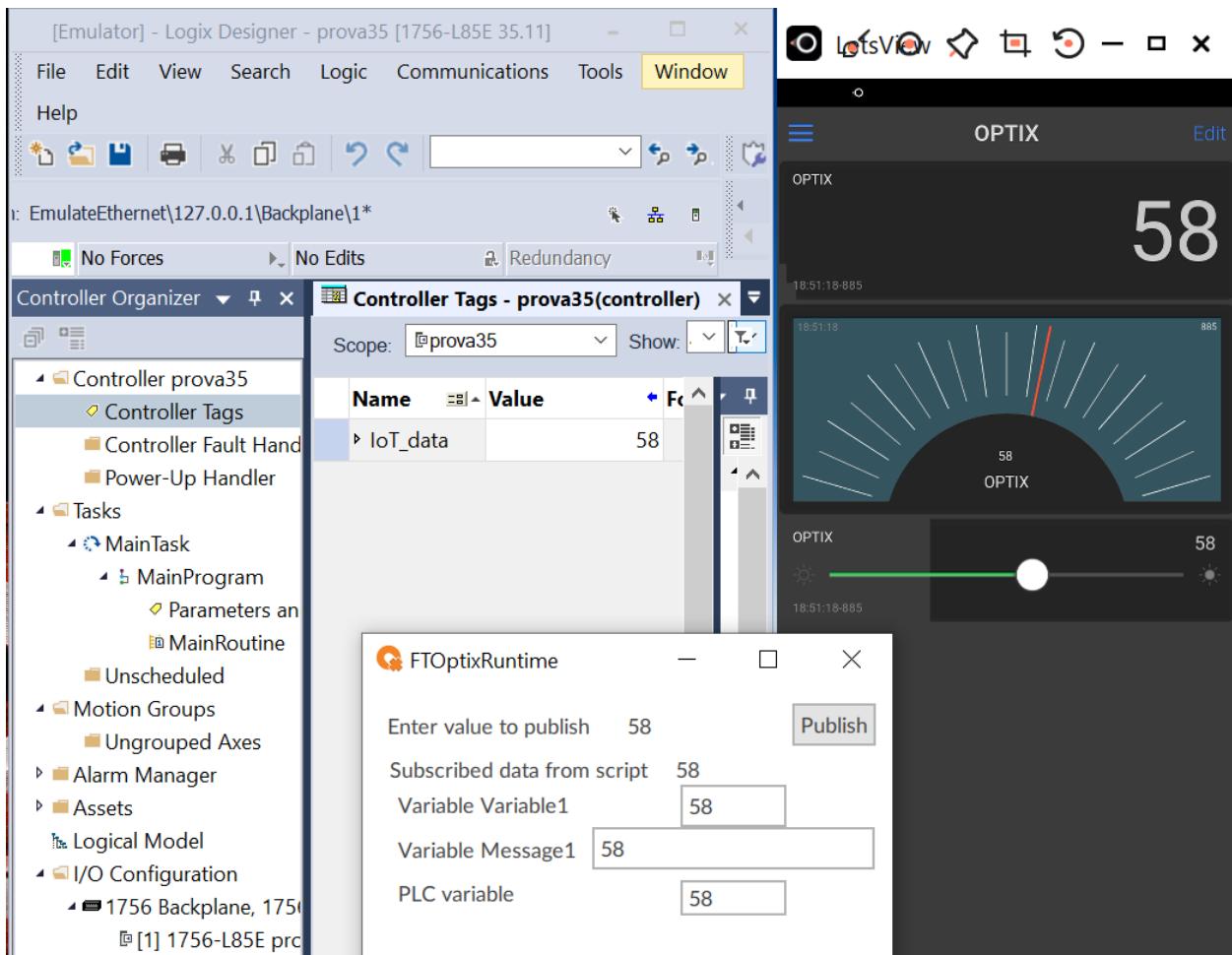
Rem Run No Forces No Edits Redundancy

Controller Organizer

Controller Tags - prova35(controller)

Scope: prova35 Show: All Tags

Name	Value
IoT_data	64



As you can see on this video

<https://youtu.be/u6EEDMmJJBU>

You can find the code here

[https://github.com/xavierflorensa/PLC2MQTTClient\\_HiveMQ\\_v1\\_Git](https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git)

### 11.1. Asynchronous publish

You may want to publish to MQTT broker each time you modify the PLC variable

You can see the result here

<https://youtu.be/4C3tAHfN1I>

You can do it with a callback function that executes a publish when Variable1 is modified.

This way

On same project delete the publish button

Copy the publish method code and insert it on a publish function

Also change the topic name from /my\_topic to /my\_new\_topic

So use this code on the publisher side

```
#region StandardUsing
using System;
using FTOptix.CoreBase;
using FTOptix.HMIProject;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
```

```

using FTOptix.NetLogic;
using FTOptix.UI;
using FTOptix.OPCUAServer;
#endregion
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
using FTOptix.RAEtherNetIP;
using FTOptix.CommunicationDriver;

public class PublisherLogic : BaseNetLogic
{
    private IUAVariable variable1;
    public override void Start()
    {
        var brokerIpAddressVariable =
Project.Current.GetVariable("Model/BrokerIpAddress");

        // Create a client connecting to the broker (default port is 1883)
        publishClient = new MqttClient(brokerIpAddressVariable.Value);
        // Connect to the broker
        publishClient.Connect("FTOptixPublishClient");
        // Assign a callback to be executed when a message is published to the
broker
        publishClient.MqttMsgPublished += PublishClientMqttMsgPublished;
        variable1 = Project.Current.GetVariable("Model/Variable1");
        //assing a callback function to be executed when the variable changes
        variable1.VariableChange += variable1_VariableChange;
    }

    public override void Stop()
    {
        publishClient.Disconnect();
        publishClient.MqttMsgPublished -= PublishClientMqttMsgPublished;
        variable1.VariableChange -= variable1_VariableChange;
    }

    private void PublishClientMqttMsgPublished(object sender,
MqttMsgPublishedEventArgs e)
    {
        Log.Info("Message " + e.MessageId + " - published = " + e.IsPublished);
    }

    private void variable1_VariableChange(object sender, VariableChangeEventArgs
e)

```

```

    {
        Log.Info("Variable has changed");
        var variable1 = Project.Current.GetVariable("Model/Variable1");
        //variable1.Value = new Random().Next(0, 101);

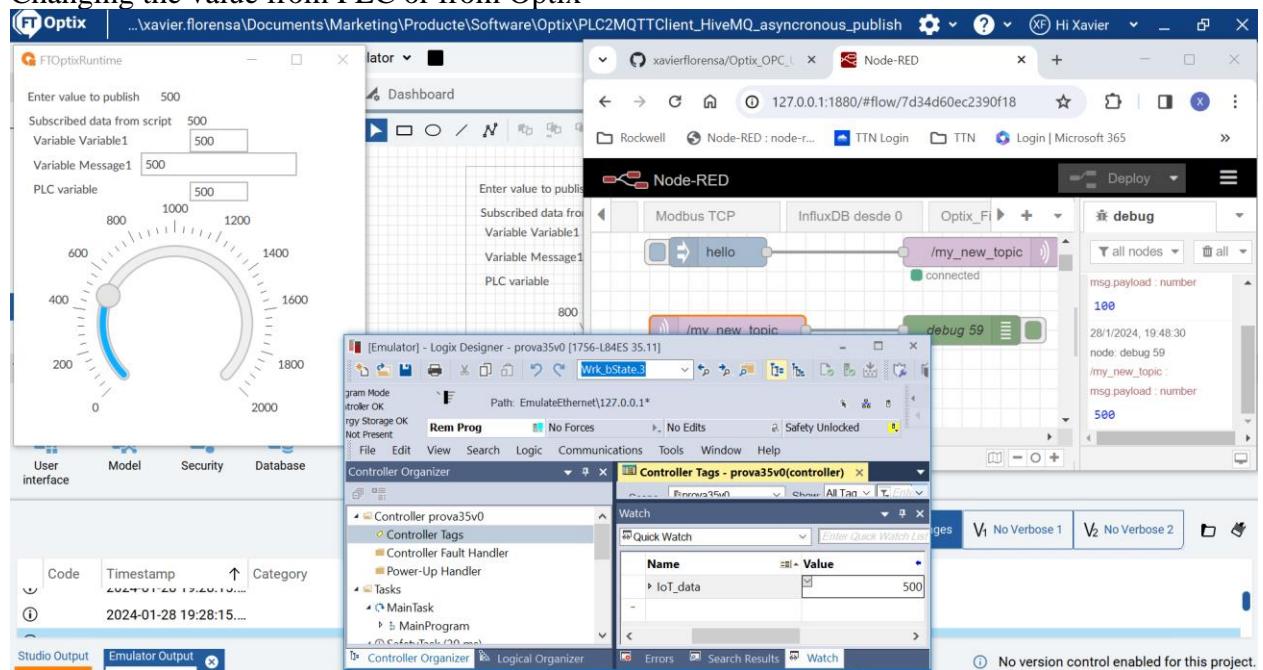
        // Publish a message
        ushort msgId = publishClient.Publish("/my_new_topic", // topic
            System.Text.Encoding.UTF8.GetBytes(((int)variable1.Value).ToString())
        , // message body
            MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, // QoS level
            false); // retained
    }

    private MqttClient publishClient;
}

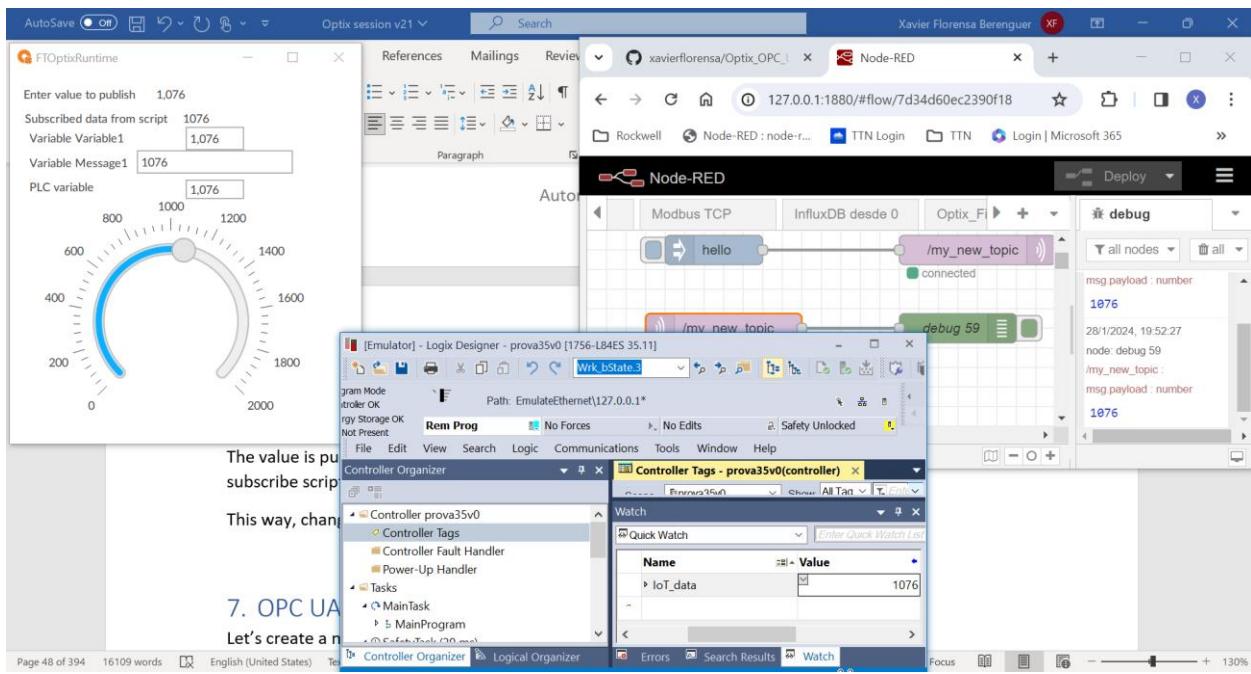
```

## Test the application

### Changing the value from PLC or from Optix



But what happens when you change the PLC value from your MQTT client as a mobile phone, The value is published twice, first time from mobile phone and second time from Optix, since Optix subscribe script will publish everything that is received thru MQTT This way, change the value from mobile phone MQTT client app



This may be corrected from C#, but this will be done some other day...

You can find the code here

[https://github.com/xavierflorensa/Optix\\_PLC\\_to\\_MQTT\\_asynchronous\\_publish](https://github.com/xavierflorensa/Optix_PLC_to_MQTT_asynchronous_publish)

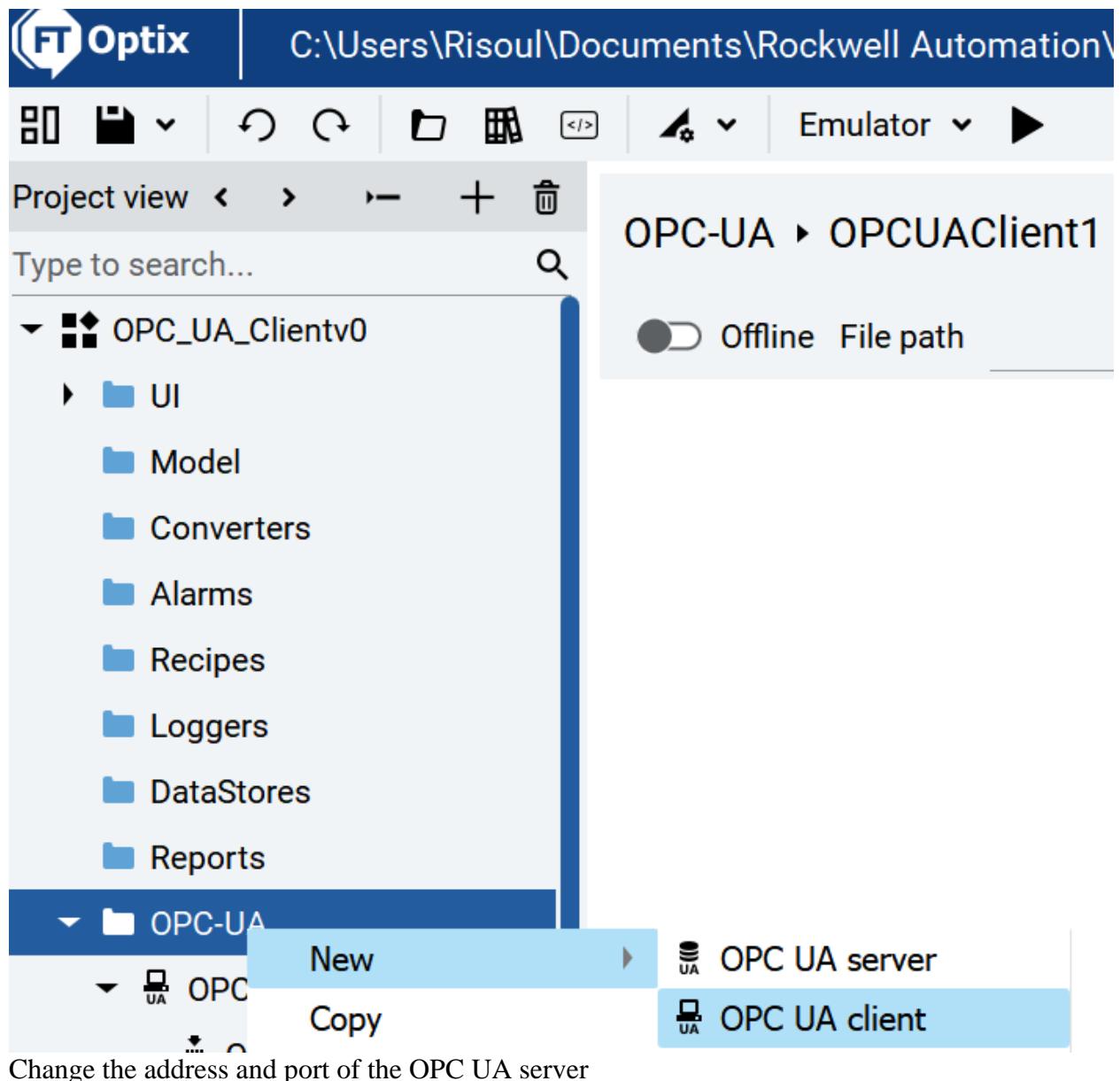
You can see the result here

<https://youtu.be/4C3tAHiFN1I>

## 12. OPC UA Client

Let's create a new OPC Client

<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/using-the-software/opcua/Add-an-OPC-UA-client-object.html>



The screenshot shows the Rockwell Automation FactoryTalk Optix OPC UA Client configuration interface. The left pane displays a project tree under 'OPC\_UA\_Clientv0' with nodes like UI, Model, Converters, Alarms, Recipes, Loggers, DataStores, Reports, and OPC-UA. Under OPC-UA, 'OPCUAClient1' is selected, showing its properties. The right pane details the client configuration:

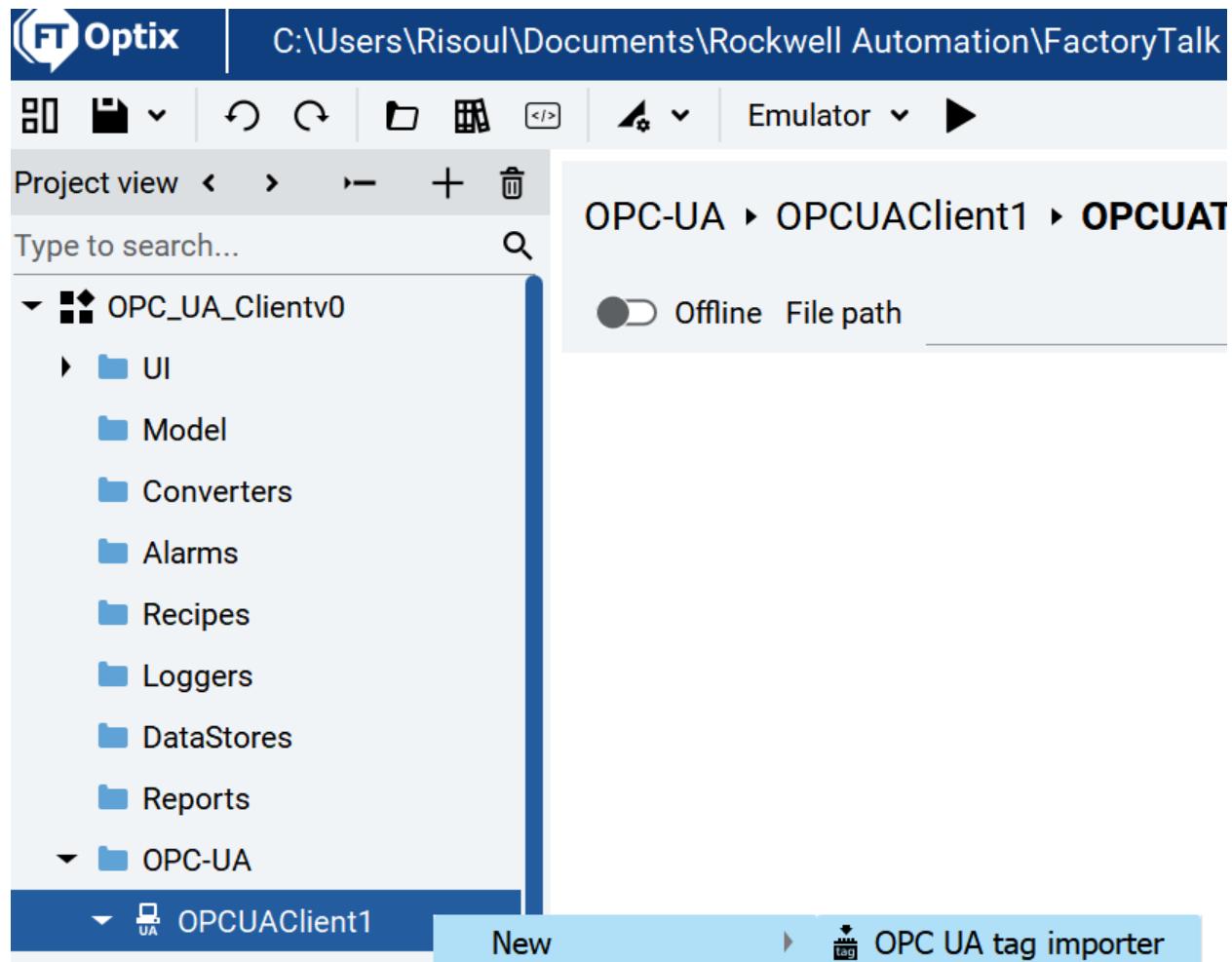
- Name:** OPCUAClient1
- Type:** OPC UA client
- Client** settings include:
  - Events: All
  - Synchronize the Node IDs on start: False
- Server Connection** settings include:
  - Server endpoint URL: opc.tcp://localhost:49370
  - Verify server identity: True
  - Requested publishing interval: 0000:00:00.000
- Runtime configurations**
- Security** settings include:
  - Minimum message security mode: None
  - Minimum security policy: None
  - Client certificate file
  - Client private key file

For instance with FT Kepserver Enterprise, port number is 49370

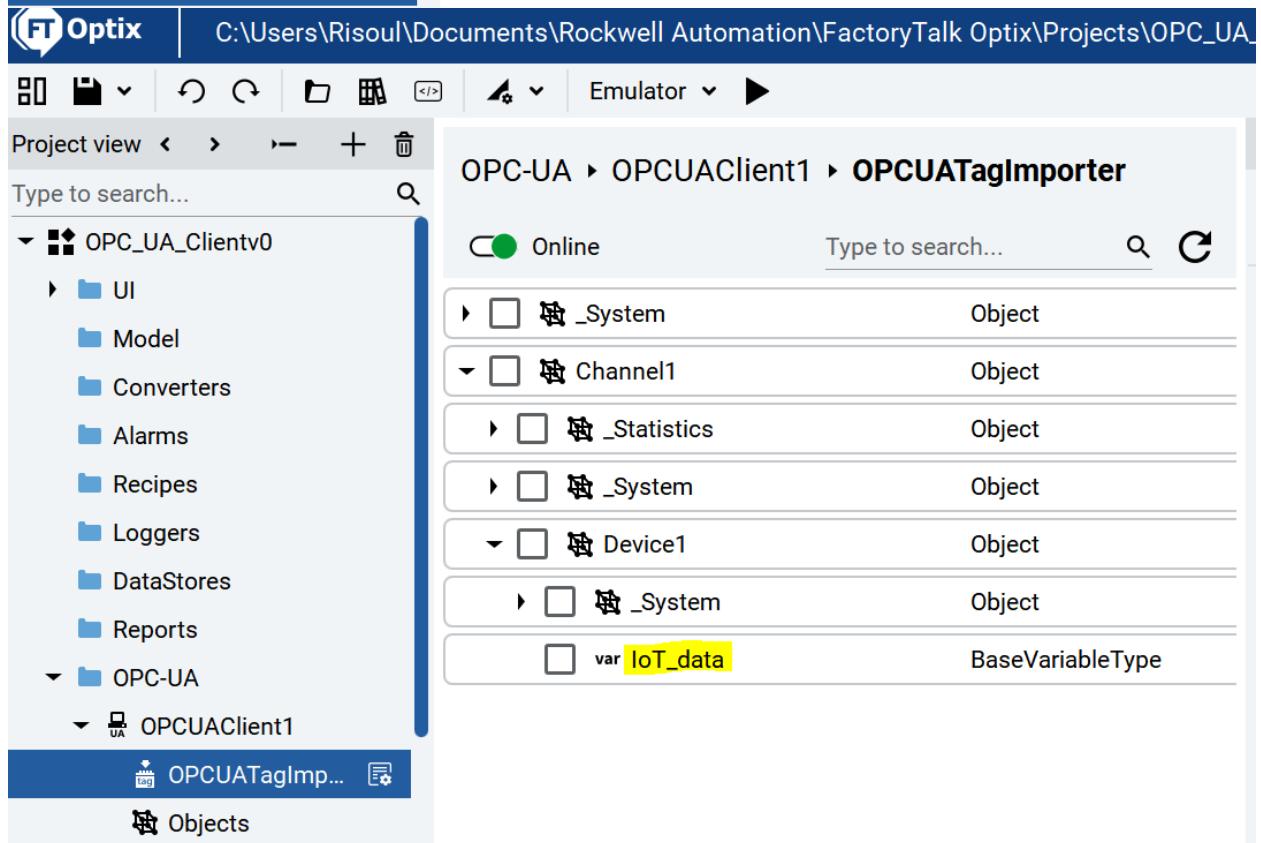
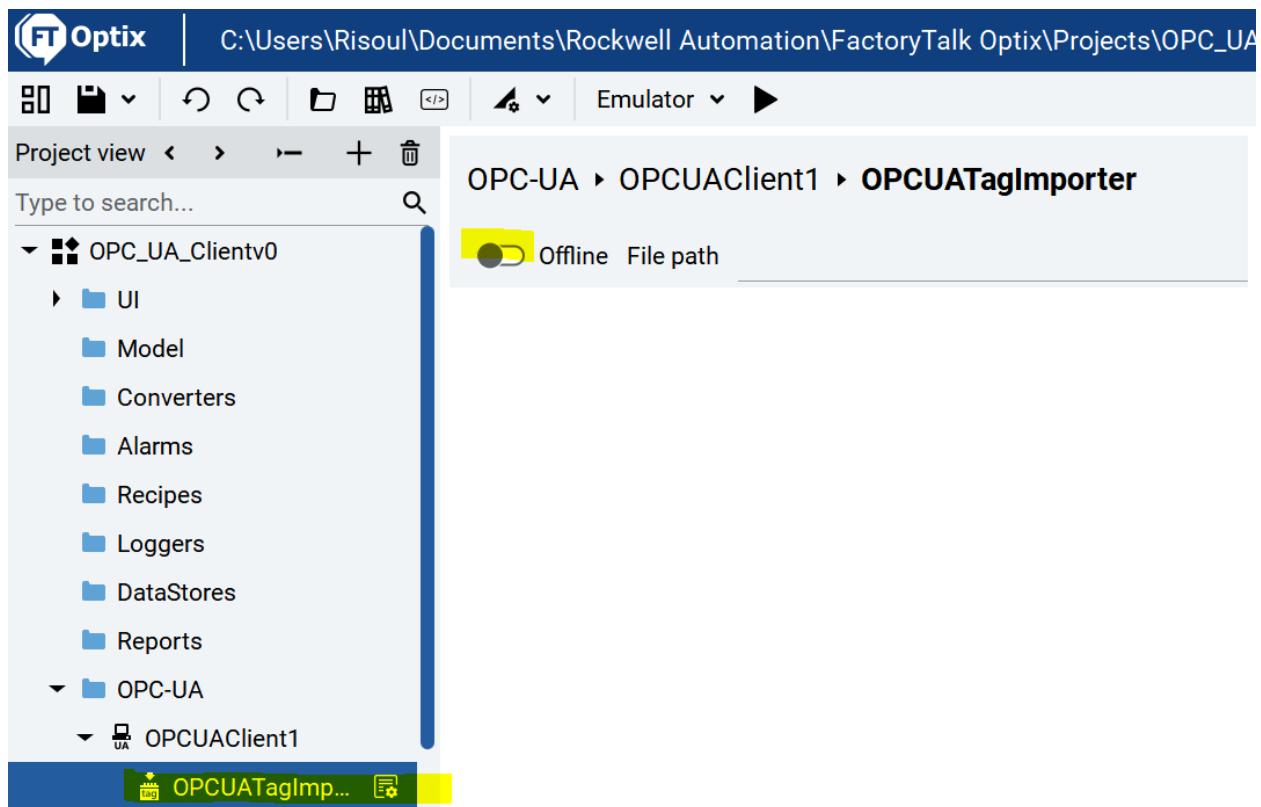
The screenshot shows the OPC UA Configuration Manager with the 'Server Endpoints' tab selected. It lists two server endpoints:

URL	Security
opc.tcp://127.0.0.1:49370	None, Basic128Rsa15 (S,SE), Basic256 (S,SE)
opc.tcp://LAPTOP-RJD8T884:49370	None, Basic128Rsa15 (S,SE), Basic256 (S,SE)

Right click to add a new OPCUA Tag Importer  
Then double click on it



Click on the Button to go online with the OPC server



Click on apply

Project view

Type to search...

OPC-UA > OPCUAClient1 > OPCUATagImporter

Online

Type to search...

<input type="checkbox"/>	_System	Object
<input type="checkbox"/>	Channel1	Object
<input type="checkbox"/>	_Statistics	Object
<input type="checkbox"/>	_System	Object
<input type="checkbox"/>	Device1	Object
<input type="checkbox"/>	_System	Object
<input checked="" type="checkbox"/>	var IoT_data	BaseVariableType UInt16

OPCUATagImp... Objects CommDrivers

Type view

All

User interface Model Security

Apply

On Objects you will see this

- OPC-UA
- OPCUAClient1
  - OPCUATagImp...
  - Objects
    - Channel1
    - Device1

Now let's create a textbox that points to the data

Type to search... 🔍

- OPCUAClient1
  - Objects
    - Channel1
      - Device1
        - var IoT\_data

BrowseName: IoT\_data  
DataType: UInt16

```

var IoT_data
OPCUATagImporter
var Connection status
var Server URI
var Enabled
var Events
var Synchronize the Node IDs on start
var Server endpoint URL
var Verify server identity

```

Attribute   Cancel Select Advanced...

**Optix** | C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\OPC\_UA\_Clientv0\*

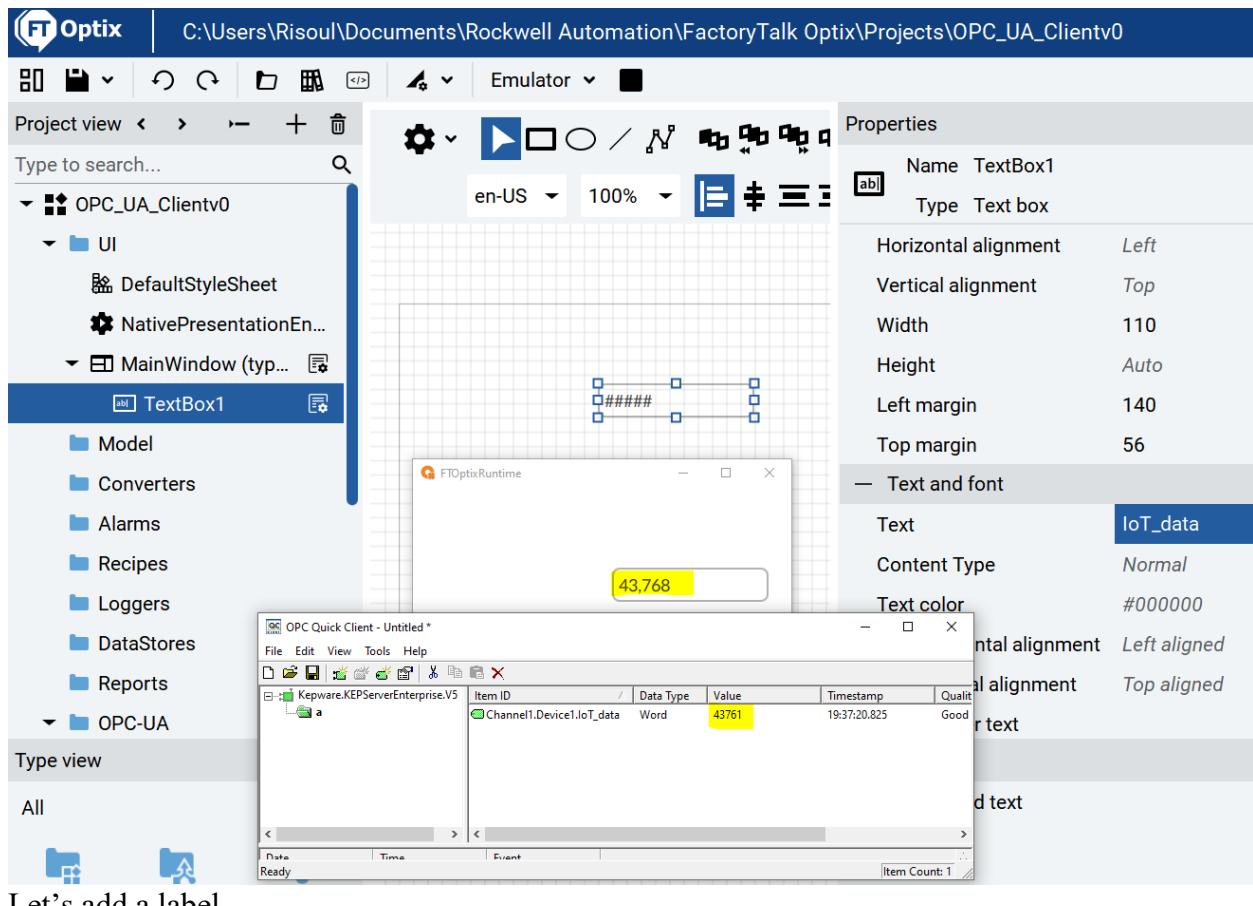
Project view Type to search... Emulator ▶

UI DefaultStyleSheet NativePresentationEn... MainWindow (typ... TextBox1 Model Converters Alarms Recipes Loggers DataStores Reports OPC-UA

Properties

Name	TextBox1
Type	Text box
Horizontal alignment	Left
Vertical alignment	Top
Width	110
Height	Auto
Left margin	140
Top margin	56
Text and font	
Text	IOT_data
Content Type	Normal
Text color	#000000
Text horizontal alignment	Left aligned
Text vertical alignment	Top aligned
Placeholder text	
Events	
Modified text	

That's all



Let's add a label



## 13. Writing to an SQL database

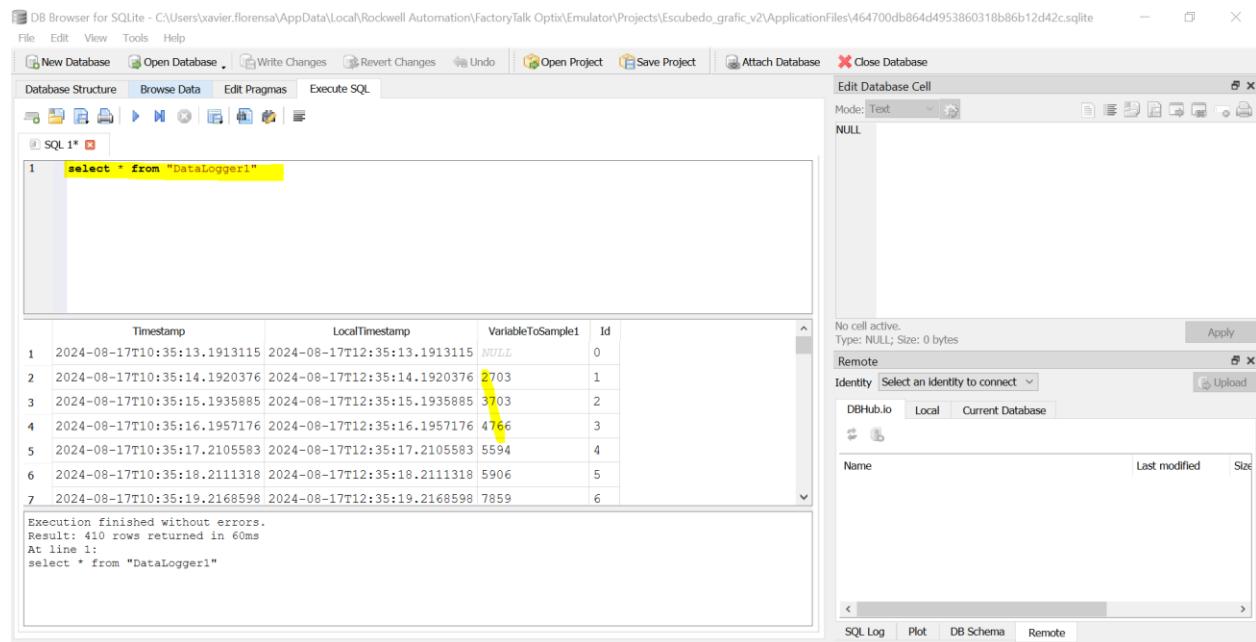
13.1. Datalogger. Working with an embedded database  
You have the step by step process on this video

[https://www.youtube.com/watch?v=Kiu5fdDzC7k&list=PL3K\\_BigUXJ1M1-JpRiwIIhzJUbhwtk3yy&index=7](https://www.youtube.com/watch?v=Kiu5fdDzC7k&list=PL3K_BigUXJ1M1-JpRiwIIhzJUbhwtk3yy&index=7)

What if you want to access the embedded database from other applications.  
Each project stores de database with a number (or name) and extension \*.sqlite on this folder

C:\Users\<yourName>\AppData\Local\Rockwell Automation\FactoryTalk Optix\Emulator\Projects\<ProjectName>\ApplicationFiles

You can open the database for instance with SQLite Browser  
[sqlitebrowser.org](http://sqlitebrowser.org)



### 13.2. Datalogger Trigger on Tag Transition

Imagine you want to store data on each machine cycle, or on each good part is produced.  
You can do it this way

<https://www.youtube.com/watch?v=IhsMH06kN0w>

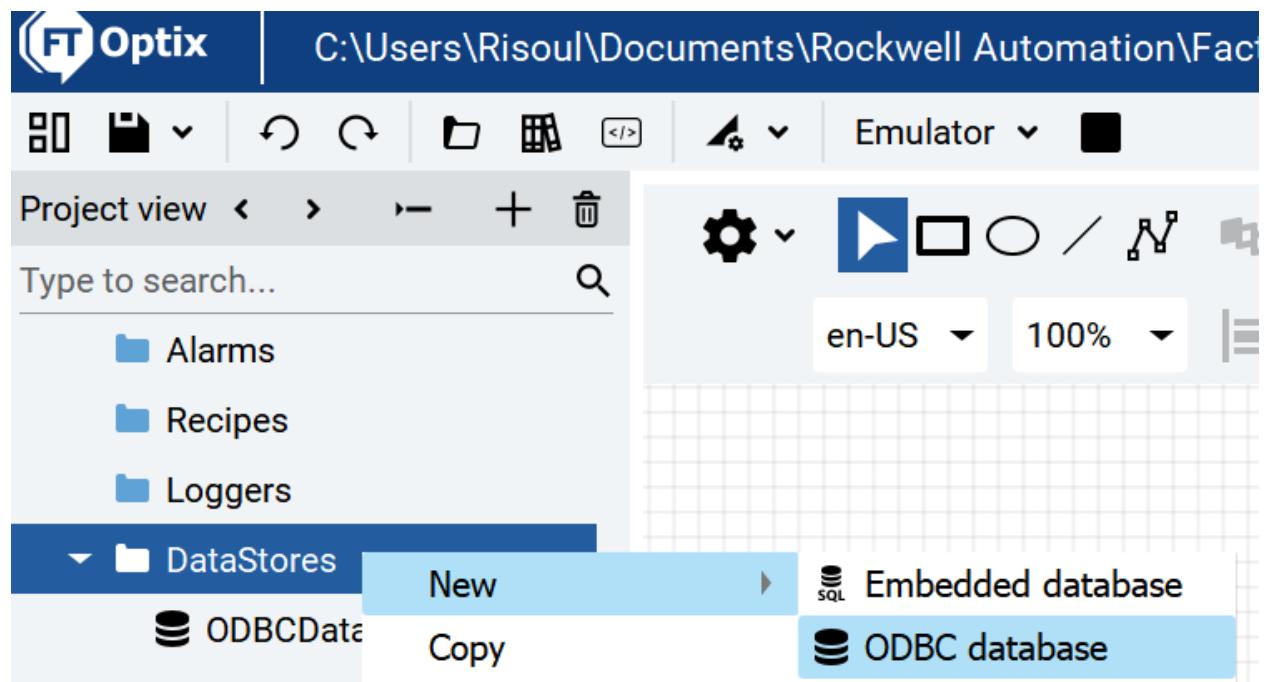
[https://github.com/xavierflorensa/FactoryTalk\\_Optix\\_Datalog\\_Trigger\\_PLC](https://github.com/xavierflorensa/FactoryTalk_Optix_Datalog_Trigger_PLC)

### 13.3. Let's write to an external SQL database from Optix

First just read from a Database

<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/using-the-software/datastore/Create-a-database.html>

Add a Database object



Be sure to have an SQL working database, with user, password, access thru TCP/IP, etc.

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The title bar indicates the session is connected to LAPTOP-RJD8T884\SQLEXPRESS under user xavier. The menu bar includes File, Edit, View, Project, Tools, Window, and Help. The toolbar contains various icons for file operations like New Query, MDX, DMX, XMLA, and DAX. The Object Explorer on the left shows the database structure: LAPTOP-RJD8T884\SQLEXPRESS (SQL Server) -> Databases -> System Databases, Database Snapshots, xavier -> Database Diagrams, Tables -> System Tables, FileTables, External Tables, Graph Tables, dbo.Table1 -> Columns (highlighted), Keys, Constraints, Triggers. The main pane displays a query window with the command "select \* from Table1;" and a results grid showing four rows of data for the column PLC\_data, all containing the value 25.

	PLC_data
1	25
2	25
3	25
4	25

Complete database properties click on Table + and Column + to match our Database

**Properties**

Name ODBCDatabase1  
Type ODBC database

— Configuration

DBMS type	SQL Server
DSN	
Server	127.0.0.1
TCP Port	1433
Database	xavier
Username	sa
Password	*****
Password	****

— Structure

▼ Tables + -

  ▼ Table1

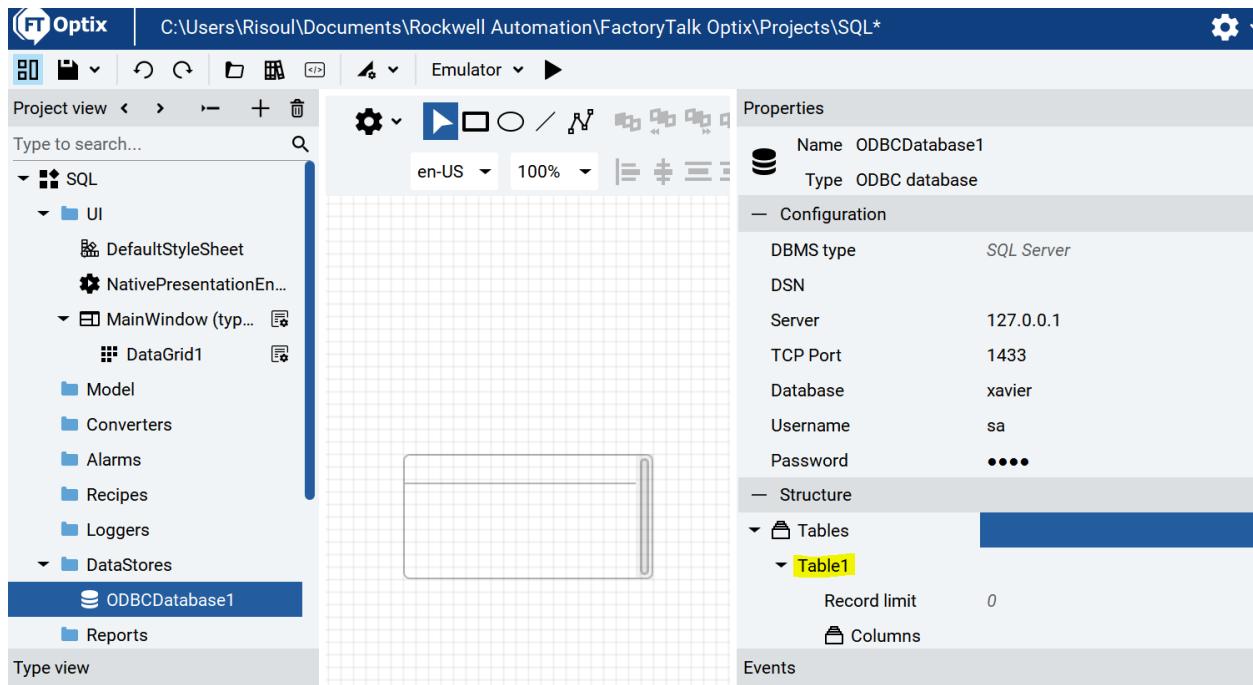
Record limit	0
▼  Columns  +	
PLC_data	Decimal 0

If you do not add a column here you will have errors when drag and drop to the Table1 to the data grid (Unable to insert a empty table to the datagrid)

Display Database data

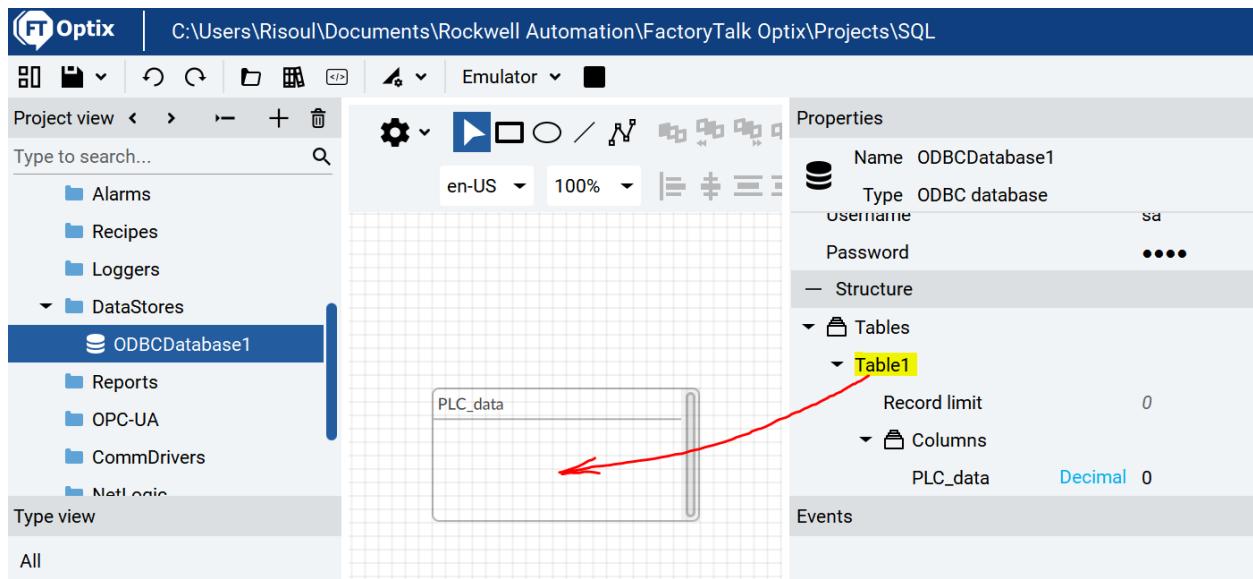
<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/using-the-software/datastore/Display-database-table-data.html>

Just click on the + on Tables to create a new table (even thought it already exists on the MS database)

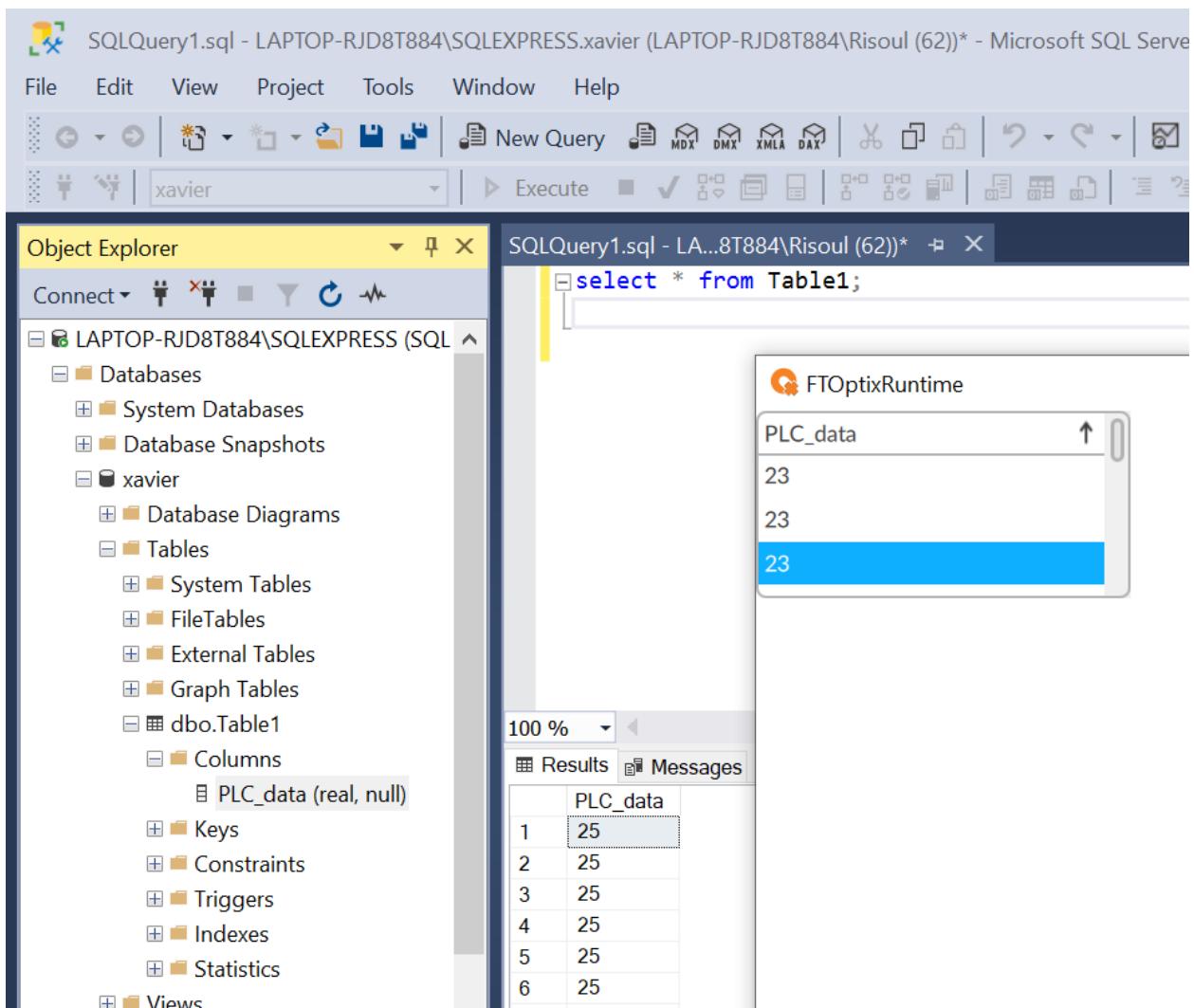


Do not forget to add a column minimum

Drag and Drop



It works!!



Now let's try to write a variable on the database

Using this example

<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/developing-solutions/app-ex/logger/log-to-odbc/Develop-a-data-logger-with-odbc-data-store.html>

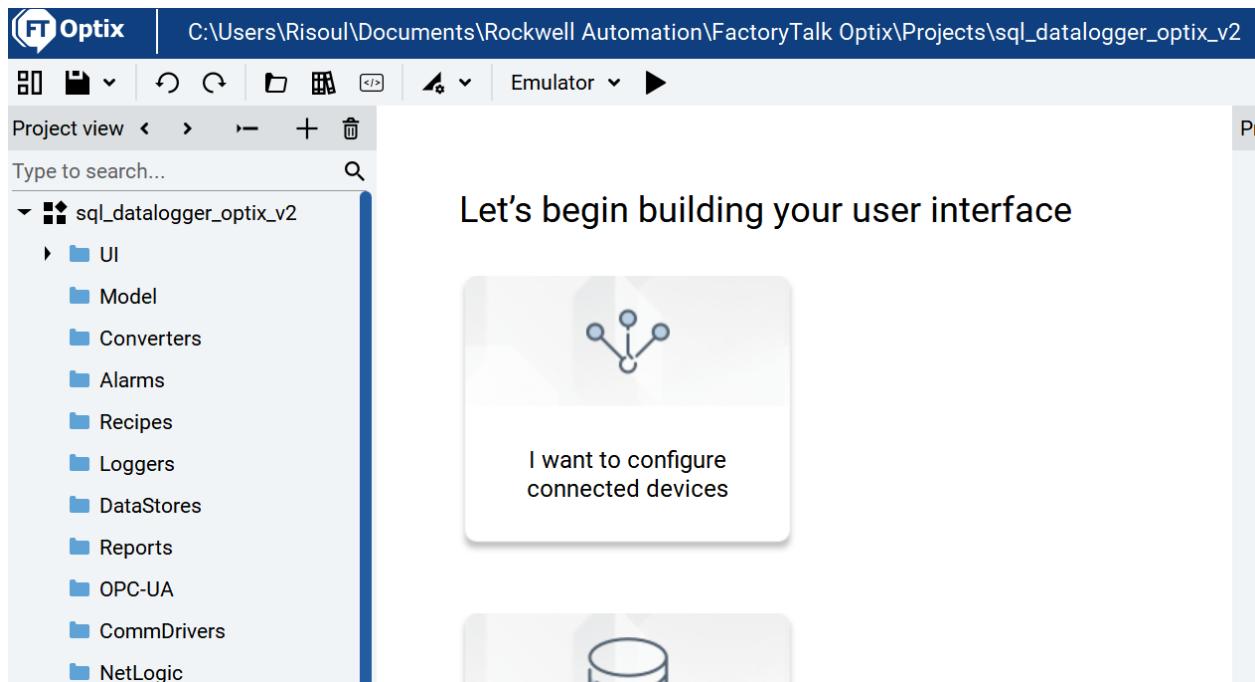
Download a sample project: [DataLoggerODBC.zip](#)

<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/downloads/DataLoggerODBC.zip>

To see how it is made.

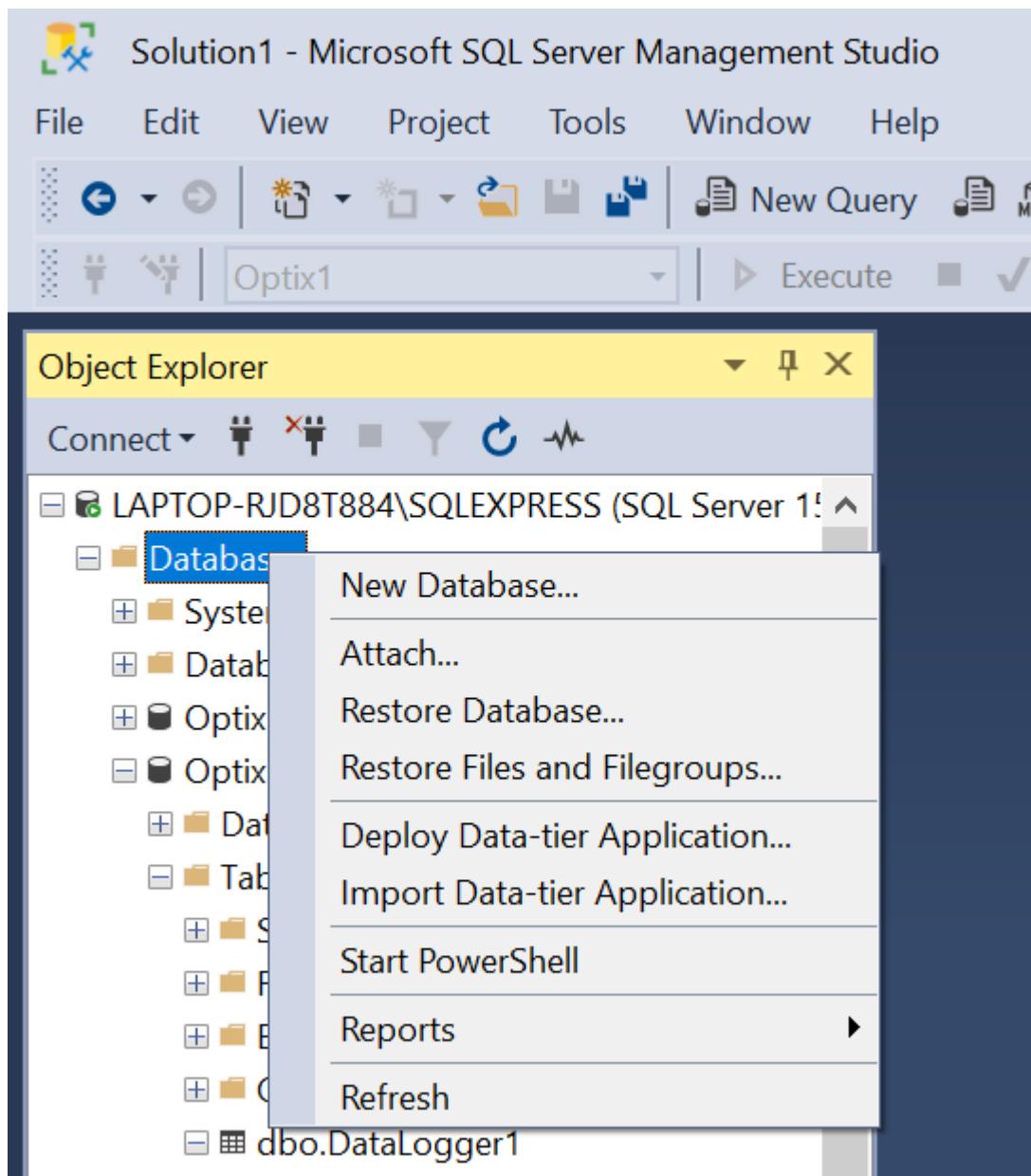
But let's start creating a complete new project

For instance sql\_datalogger\_optix\_v2

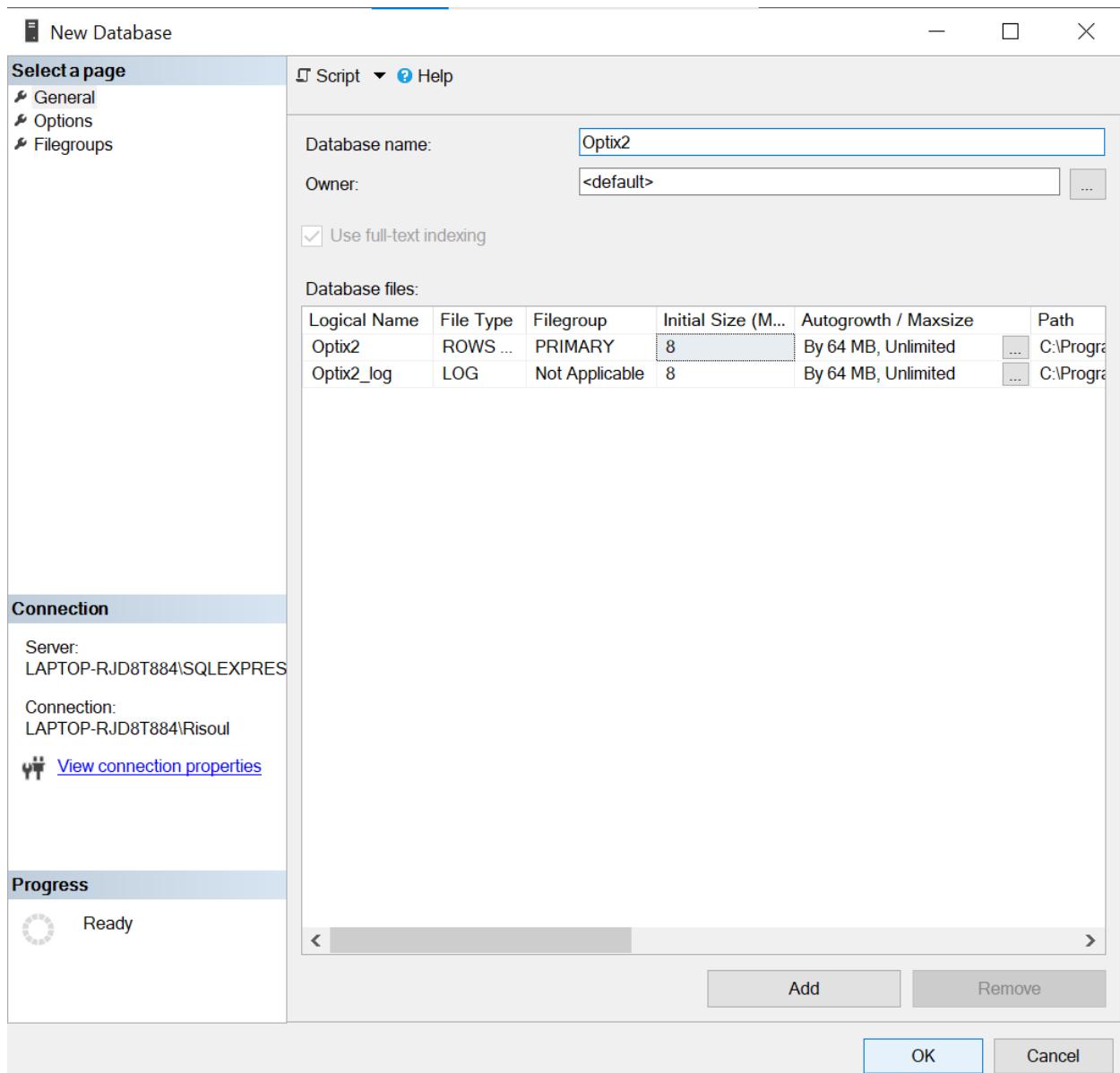


But first let's create a database without a Table (Optix and The Datalogger function will create the Table)

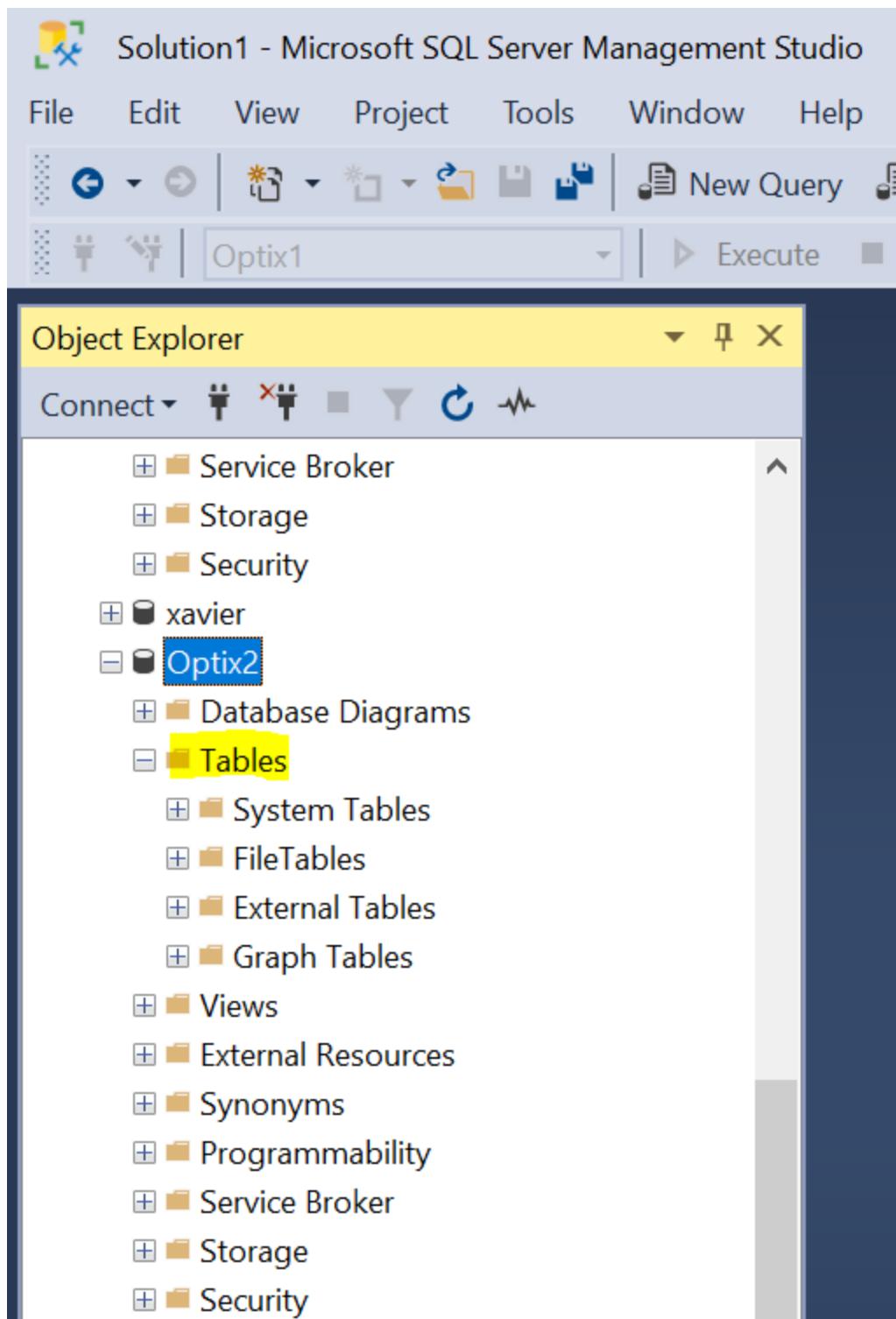
For instance let's create a new database Optix2



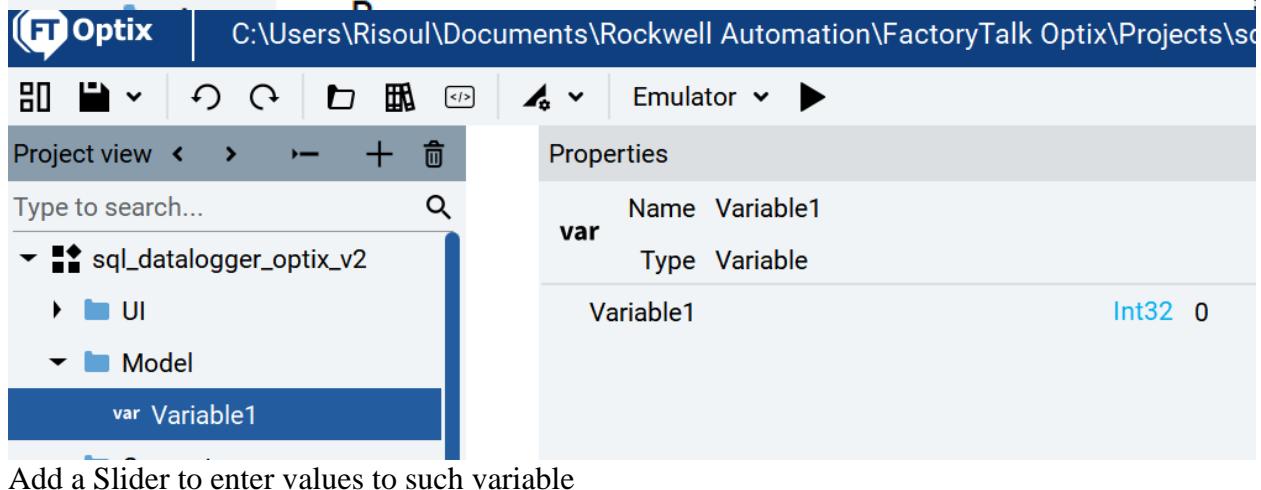
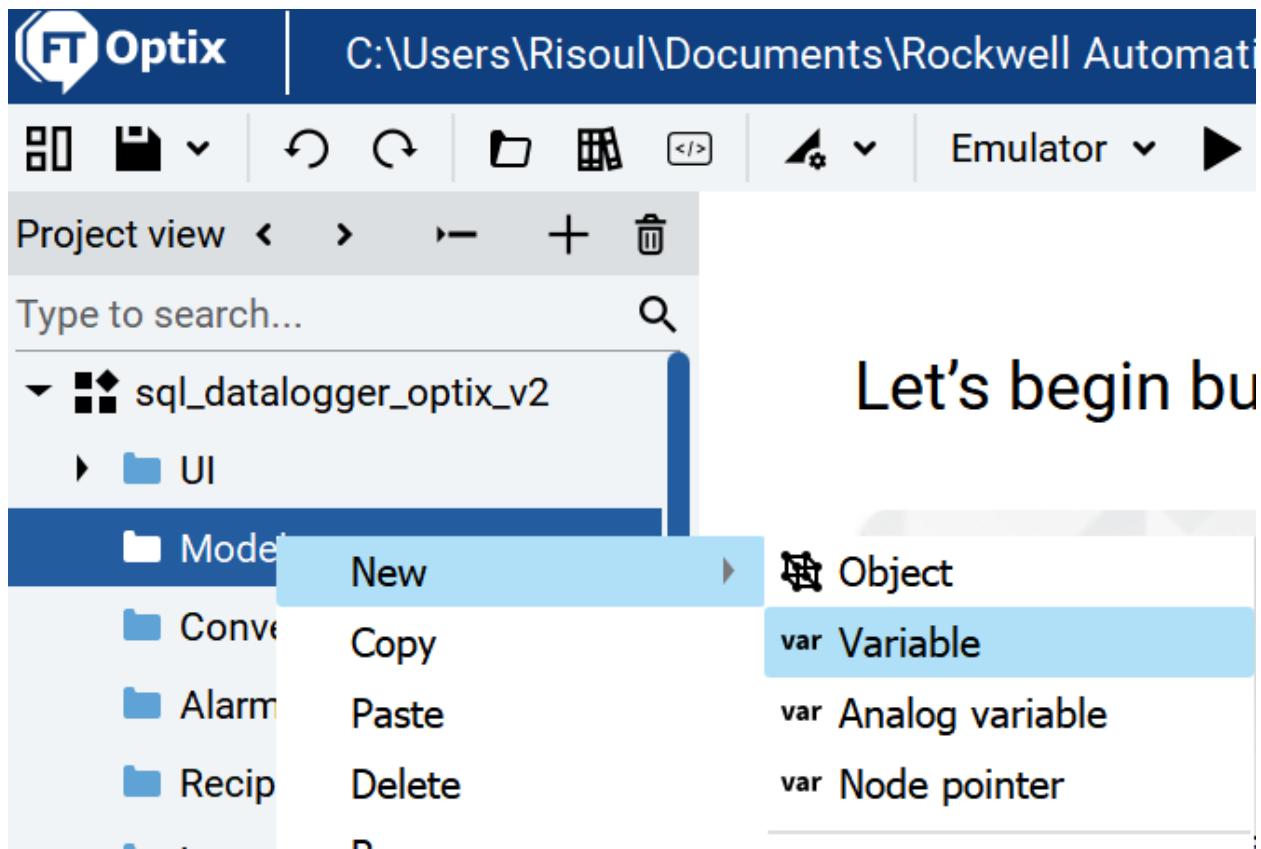
Just give a name, that's all

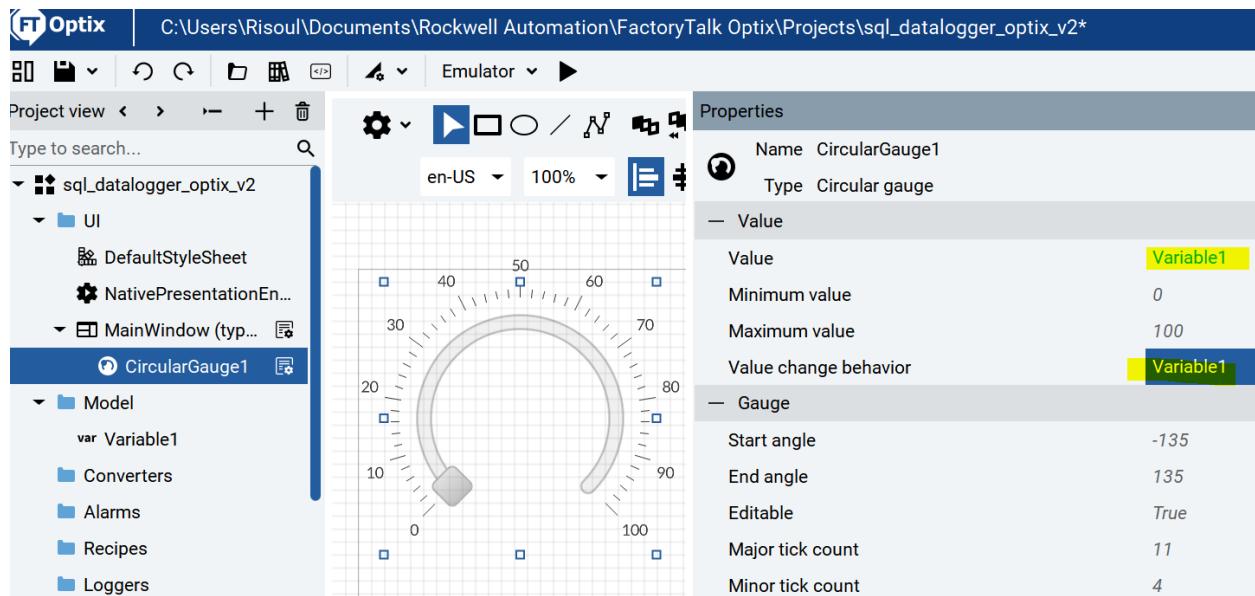


There are no user tables

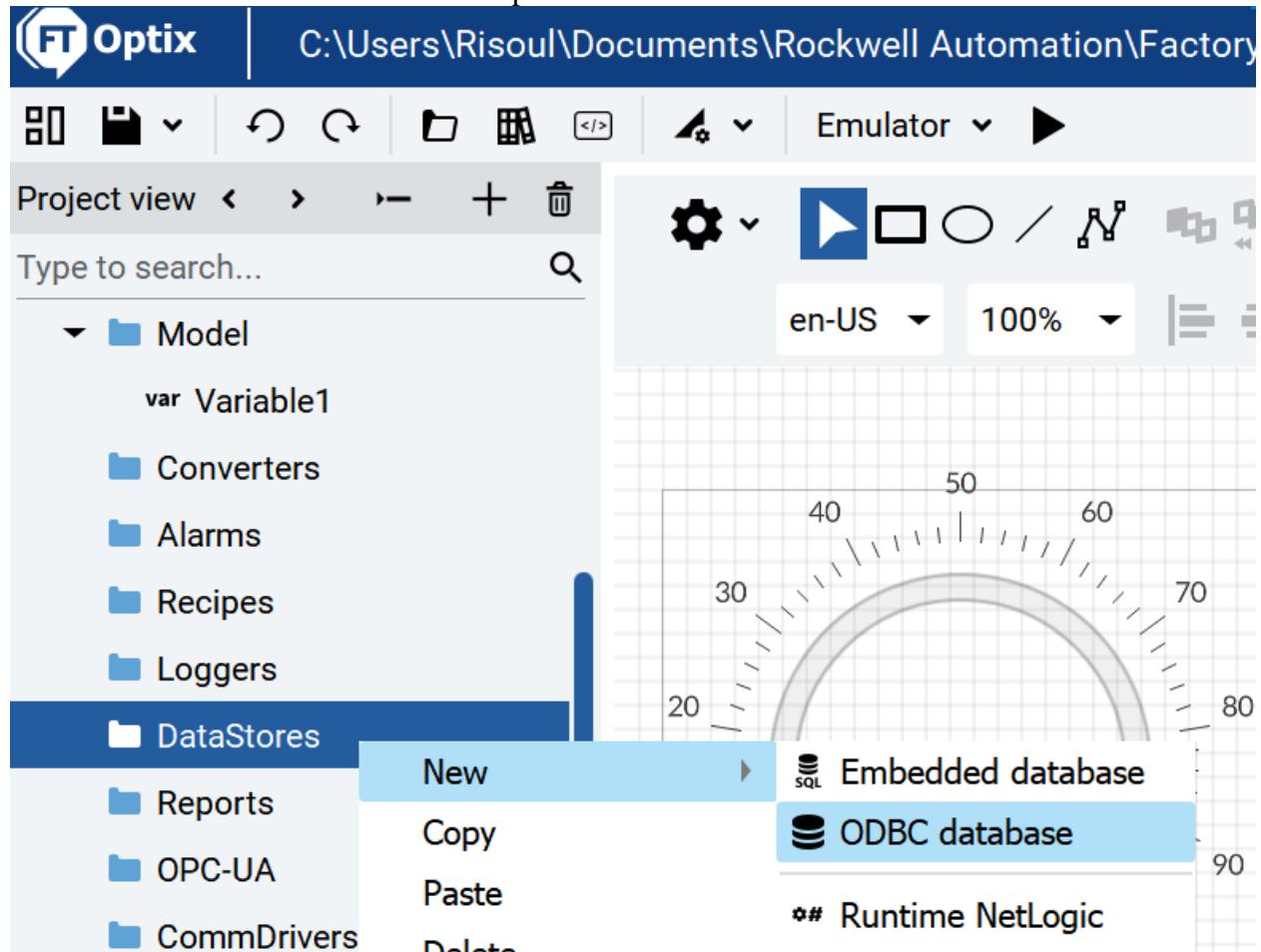


Now Let's go to Optix  
Add a variable to work with

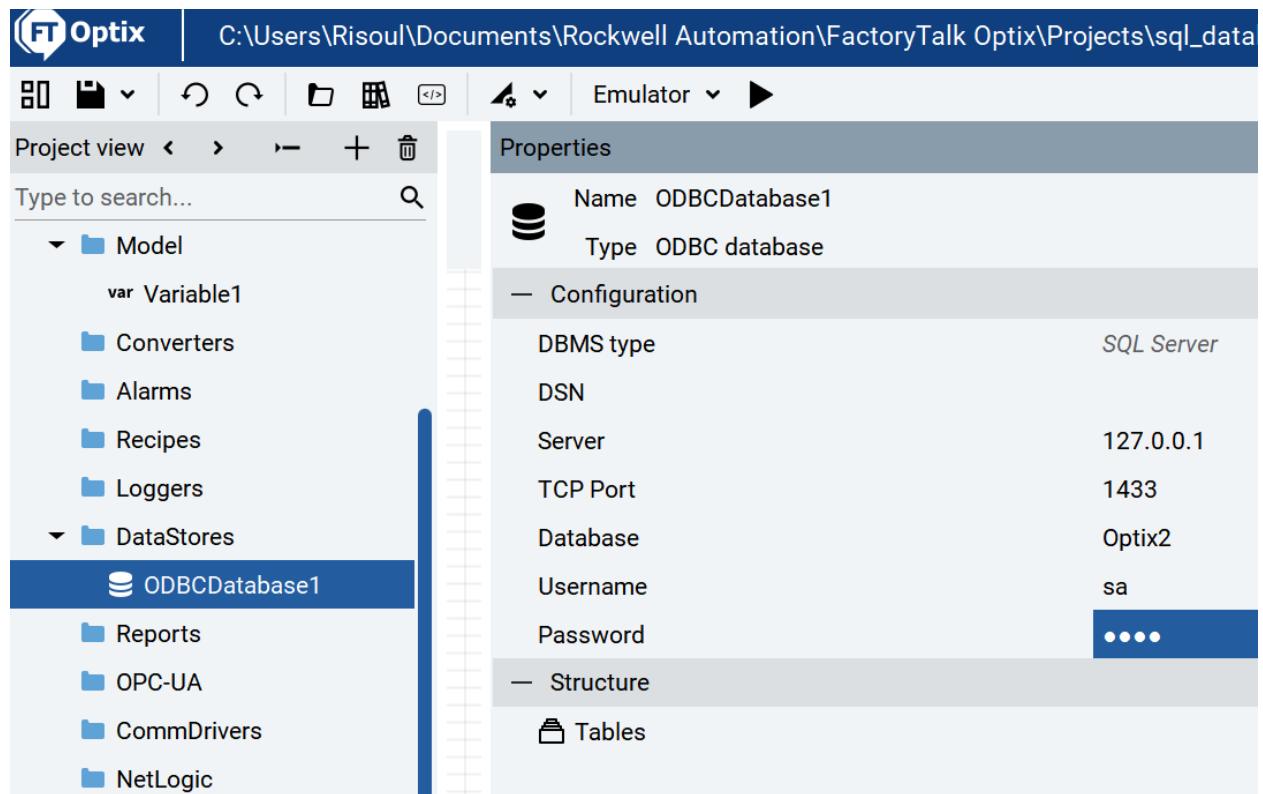




Next create the Database instance on Optix



Point to the existing database, but do not create any Table on Optix



Then create a new DataLogger

FT Optix | C:\Users\Risoul\Documents\Rockwell Automation

The screenshot shows the FT Optix interface with the title bar "C:\Users\Risoul\Documents\Rockwell Automation". The main area has a toolbar with icons for file operations, search, and navigation. Below the toolbar is a "Project view" section with a search bar and a tree view of project components. A context menu is open over the "Loggers" folder, listing options: New, Copy, Paste, and Delete. The "New" option is highlighted, and its submenu shows "Data logger" selected. Other options in the submenu include "Event logger" and "Runtime NetLogic".

And select the variable to log and the destination, just the database

FT Optix | C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\sql\_datalogger\_optix

The screenshot shows the FT Optix interface with the title bar "C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\sql\_datalogger\_optix". The main area has a toolbar with icons for file operations, search, and navigation. Below the toolbar is a "Project view" section with a search bar and a tree view of project components. A context menu is open over the "DataLogger1" object in the tree view, showing properties for the "DataLogger1" object. The properties listed are:

Name	DataLogger1
Type	Data logger
Sampling mode	Periodic
Sampling period	0000:00:01.000
Log operation code of each variable	False
Log timestamp of each variable	False
Log local time	True
Table name	
Variables to log	
VariableToSample1	BaseDataType ..../..../Model/Variable1
Store	ODBCDatabase1

If we now open the ODBC object, surprise, a new Table an columns have been created for us

The screenshot shows the Rockwell Automation FactoryTalk Optix software interface. The left pane is the 'Project view' showing a tree structure of project components:

- Model
  - var Variable1
- Converters
- Alarms
- Recipes
- Loggers
  - DataLogger1
- DataStores
  - ODBCDatabase1
  - Reports
  - OPC-UA
  - CommDrivers
  - NetLogic
  - Security

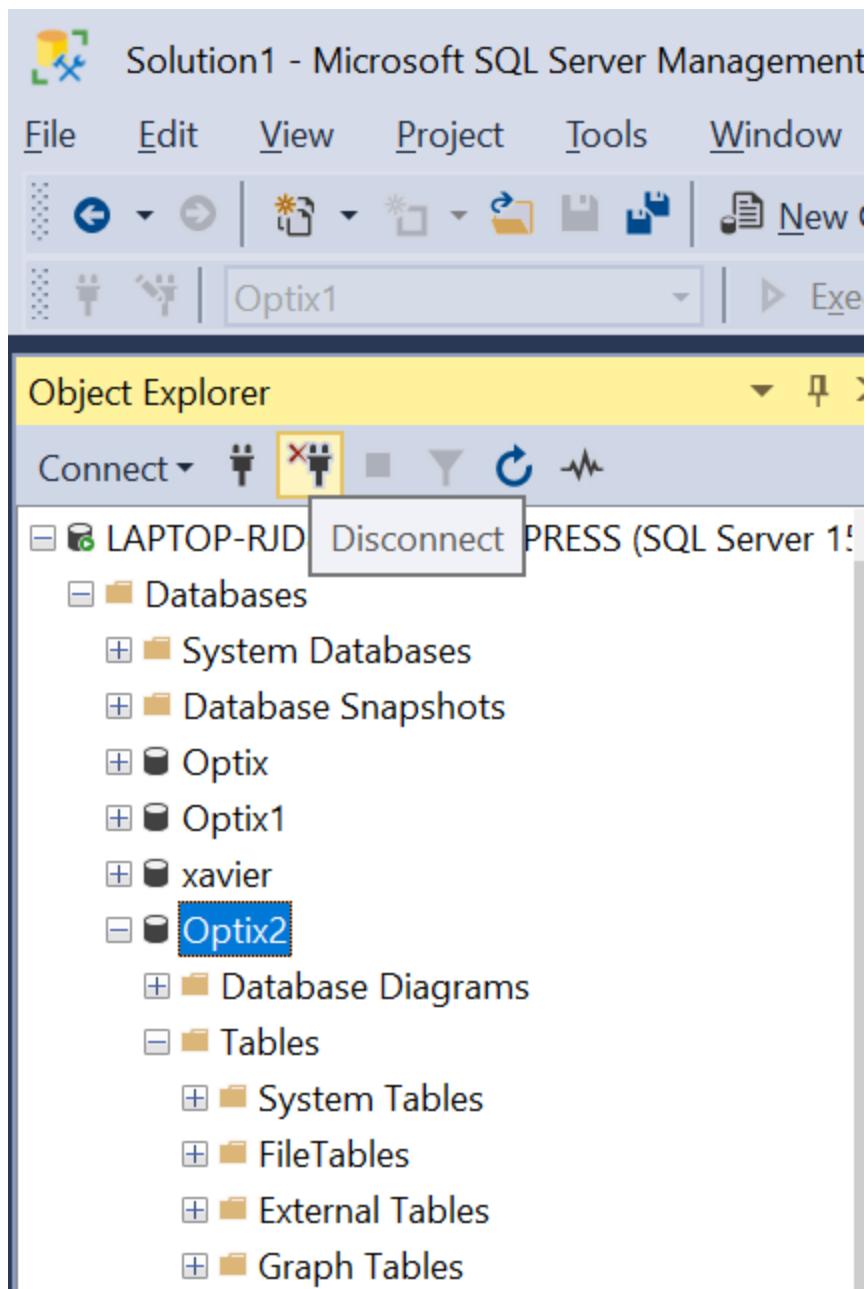
The 'ODBCDatabase1' node is selected and highlighted in blue.

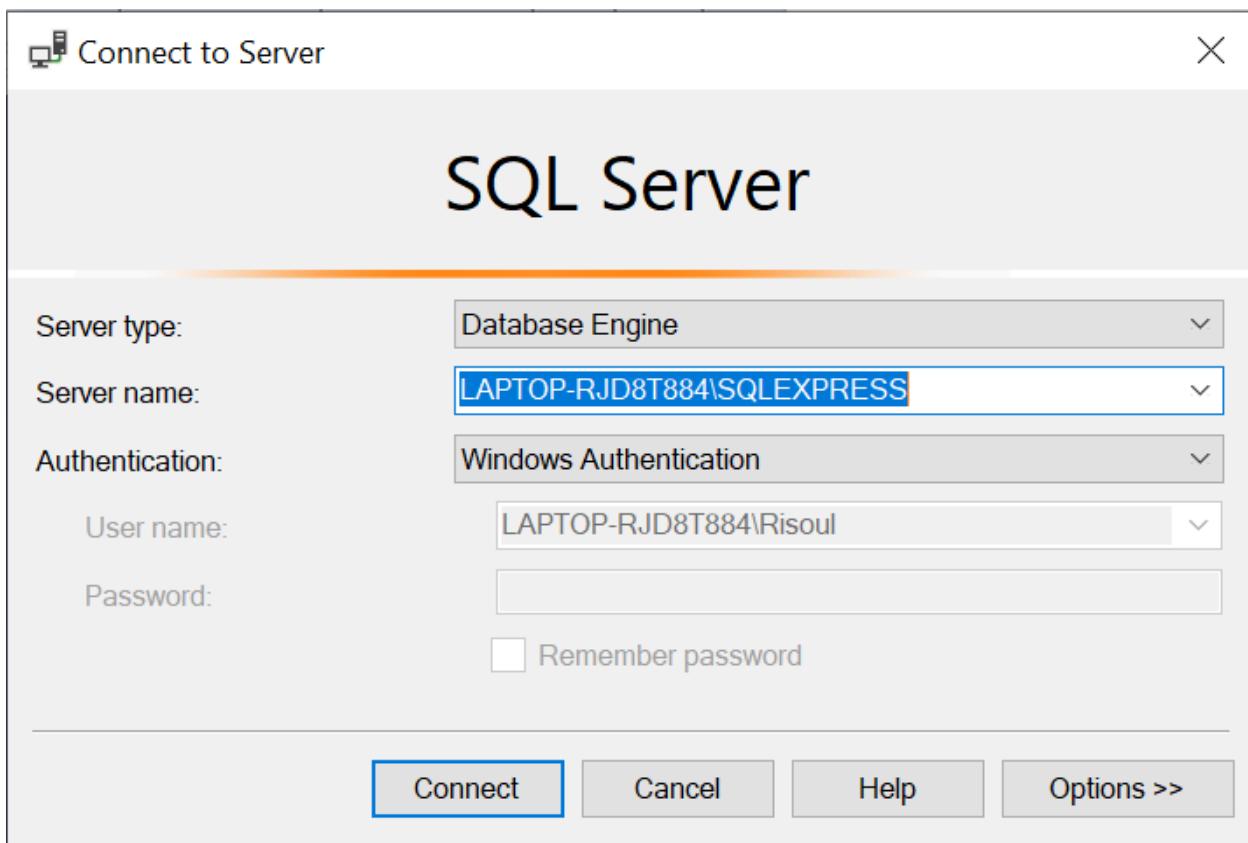
The right pane is the 'Properties' panel, displaying configuration settings for the selected ODBC database:

Setting	Value
Name	ODBCDatabase1
Type	ODBC database
Server	127.0.0.1
TCP Port	1433
Database	Optix2
Username	sa
Password	••••
Structure	
Tables	
DataLogger1	
Record limit	0
Columns	
Timestamp	UtcTime Not set
LocalTimestamp	DateTime Not set
VariableToSample1	Int32 0

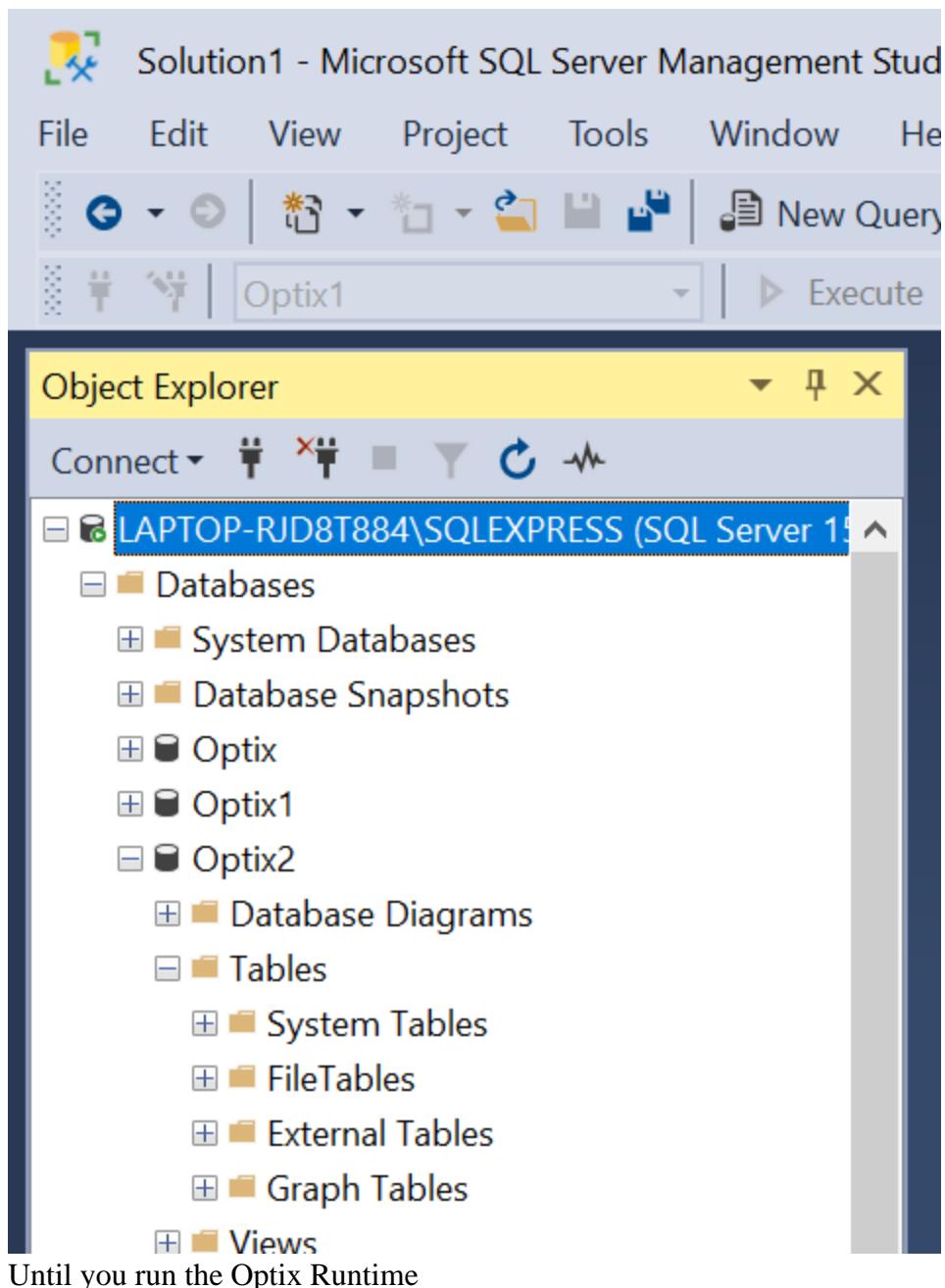
If we take a look at the database:

You have to disconnect and connect again to see the changes



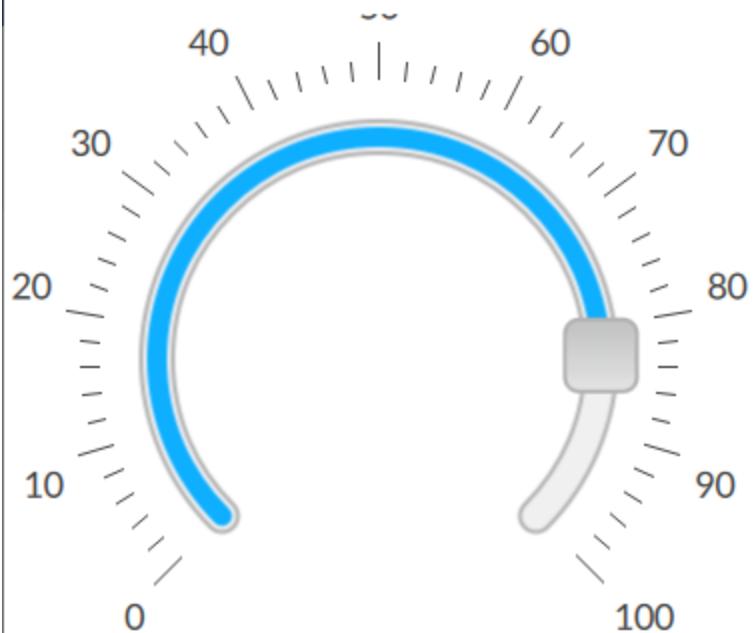


Still nothing

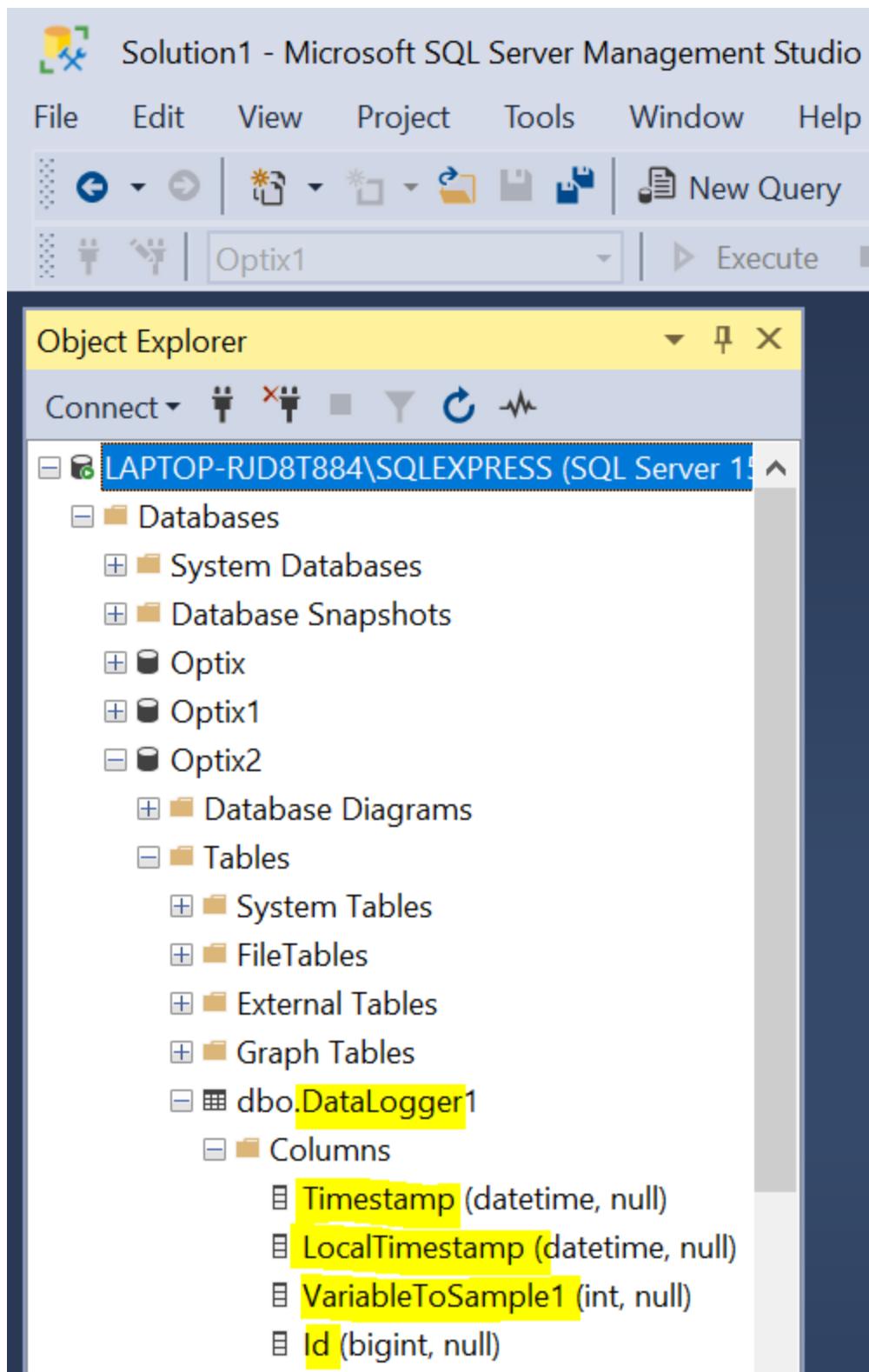




FTOptixRuntime



Give some values to the gauge dropping the index  
Then go to the Database, and ...voilà



Now let's look at the data, this is working fine

SQLQuery5.sql - LAPTOP-RJD8T884\SQLEXPRESS.Optix2 (LAPTOP-RJD8T884\Risoul (63))\* - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Optix2 Execute

Object Explorer

LAPTOP-RJD8T884\SQLEXPRESS (SQL Server 1!)

- Databases
  - System Databases
  - Database Snapshots
  - Optix
  - Optix1
  - Optix2
- Tables
  - System Tables
  - FileTables
  - External Tables
  - Graph Tables
- dbo.DataLogger1
  - Columns
    - Timestamp (datetime, null)
    - LocalTimestamp (datetime, null)
    - VariableToSample1 (int, null)
    - Id (bigint, null)
  - Keys
  - Constraints

SQLQuery5.sql - LA...8T884\Risoul (63)\*

```
use Optix2;
select * from DataLogger1;
```

Results

	Timestamp	LocalTimestamp	VariableToSample1	Id
1	2023-01-21 12:12:51.063	2023-01-21 13:12:51.063	0	0
2	2023-01-21 12:12:52.063	2023-01-21 13:12:52.063	0	1
3	2023-01-21 12:12:53.063	2023-01-21 13:12:53.063	22	2
4	2023-01-21 12:12:54.063	2023-01-21 13:12:54.063	47	3
5	2023-01-21 12:12:55.063	2023-01-21 13:12:55.063	72	4
6	2023-01-21 12:12:56.063	2023-01-21 13:12:56.063	83	5
7	2023-01-21 12:12:57.063	2023-01-21 13:12:57.063	83	6
8	2023-01-21 12:12:58.063	2023-01-21 13:12:58.063	83	7

Now let's add a Datagrid on the Optix Project  
 Drag and drop the table to the Datagrid

FT Optix C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\sql\_datalogger\_optix\_v2\*

MainWindow

Properties

- Name: ODBCDatabase1
- Type: ODBC database
- Server
- TCP Port
- Database
- Username
- Password
- Structure
- Tables
  - DataLogger1
- Record limit
- Columns
  - Timestamp
  - LocalTimestamp
  - VariableToSample1
- Events

FT Optix C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\sql\_datalogger\_optix\_v2\*

Project view Emulator

Type to search... CircularGauge1 DataGrid1

Model

- var Variable1
- Converters
- Alarms
- Recipes
- Loggers
- DataLogger1
- DataStores
- ODBCDatabase1
- Reports
- OPC-UA
- CommDrivers

Properties

Name ODBCDatabase  
Type ODBC database  
Server  
TCP Port  
Database  
Username  
Password  
Structure  
Tables

DataLogger1

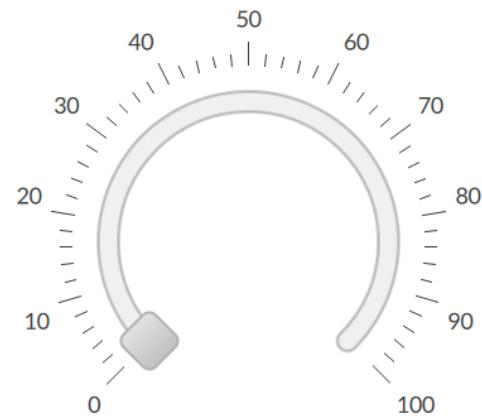
- Record limit
- Columns
- Timestamp
- LocalTimestamp
- VariableToSample1

Events

Let's test it. It works! That's all

FTOptixRuntime

Timestamp	↑ LocalTimestamp	VariableToSample1
Jan 21, 2023, 12:19:09 PM	Jan 21, 2023, 1:19:09 PM	54
Jan 21, 2023, 12:19:10 PM	Jan 21, 2023, 1:19:10 PM	100
Jan 21, 2023, 12:19:11 PM	Jan 21, 2023, 1:19:11 PM	100



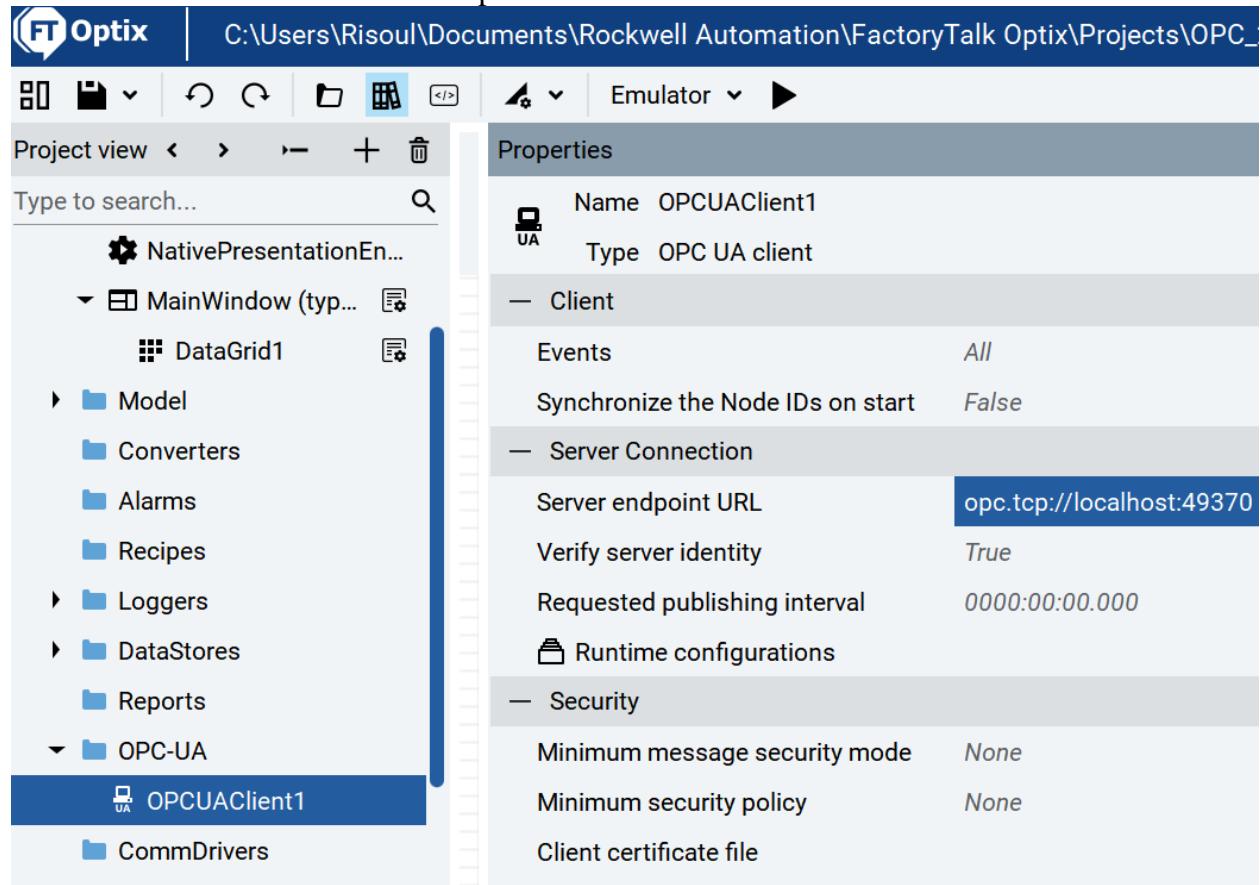
## 14. Creating a OPC UA to SQL converter

You can see the results here

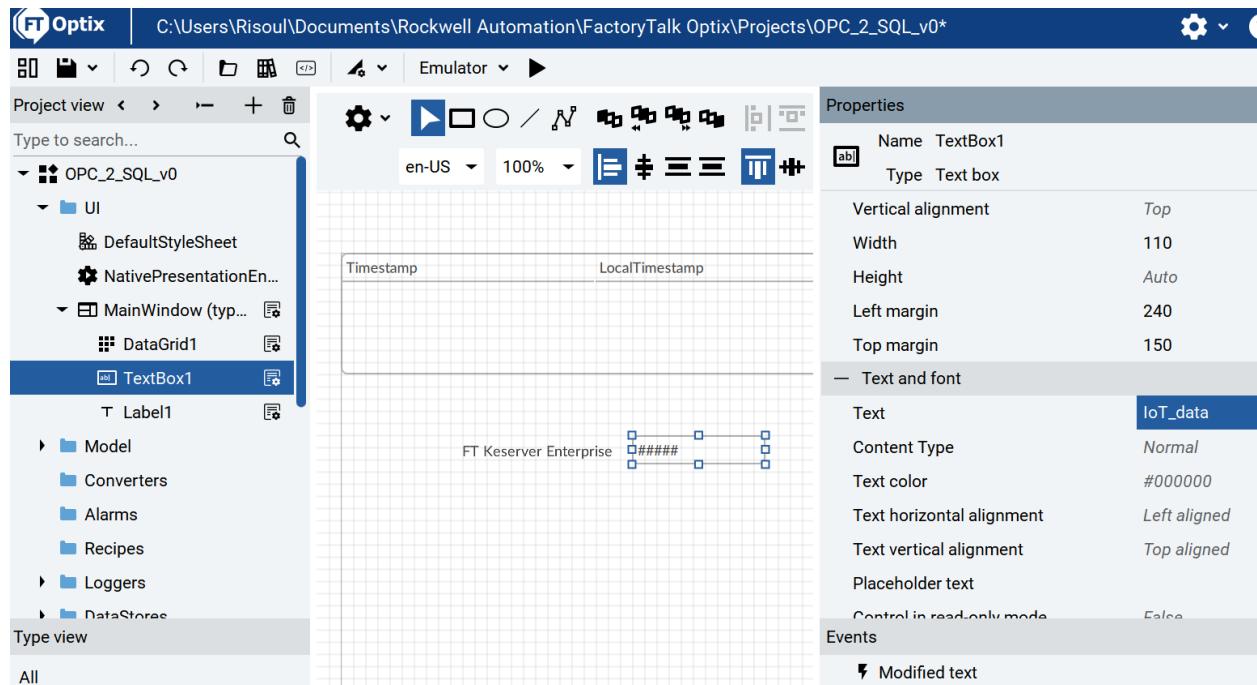
<https://youtu.be/Bcs8rT0ytUQ>

Take last SQL project and delete the gauge.

Then add the OPC UA client like in previous section.



Now we have access to IoT\_data Tag



First of all test that we are reading OPC UA data

Yes

FTOptixRuntime		
Timestamp	LocalTimestamp	VariableToSample1
Jan 21, 2023, 12:12:51 PM	Jan 21, 2023, 1:12:51 PM	0
Jan 21, 2023, 12:12:52 PM	Jan 21, 2023, 1:12:52 PM	0
Jan 21, 2023, 12:12:53 PM	Jan 21, 2023, 1:12:53 PM	22

FT Keserver Enterprise 44,946

Then let's erase the data on the database from previous chapter.

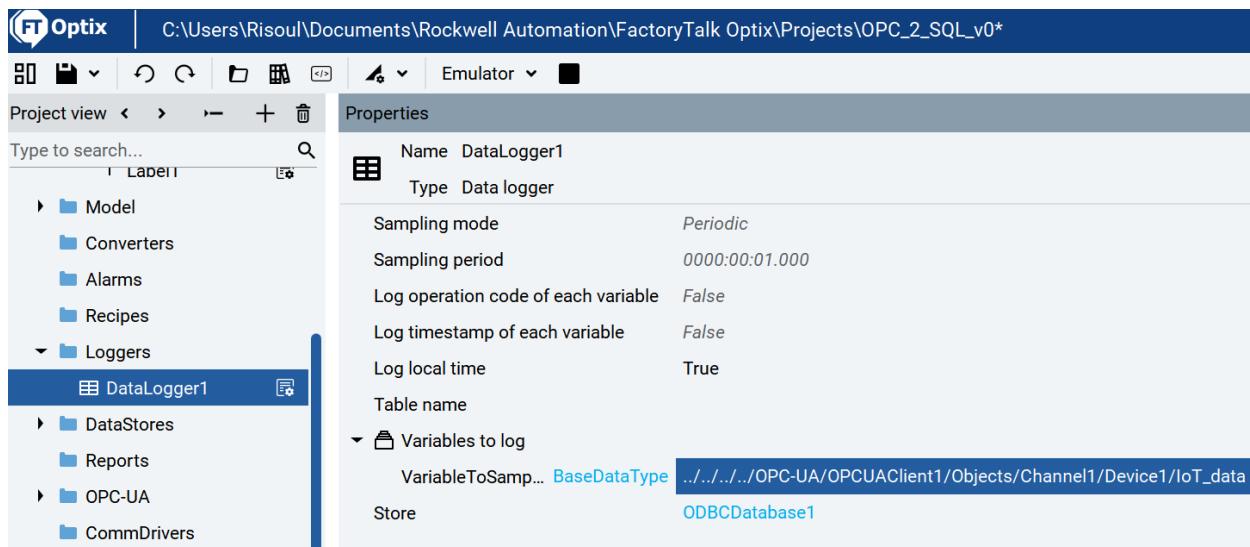
The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left displays the database structure for 'LAPTOP-RJD8T884\SQLEXPRESS (SQL Server 1)'. It shows databases like System Databases, Database Snapshots, Optix, Optix1, and Optix2. Under Optix2, there is a 'Tables' node expanded, showing 'dbo.DataLogger1' with columns: Timestamp, LocalTimestamp, VariableToSample1, and Id. The right pane contains a query window titled 'SQLQuery1.sql - LAPTOP-RJD8T884\Risoul (55)\*'. The query is:

```
use Optix2;
delete from DataLogger1;
select * from DataLogger1;
```

Verify that the data is erased from Optix

The screenshot shows the 'FTOptixRuntime' application window. It has three columns: 'Timestamp', 'LocalTimestamp', and 'VariableToSample1'. Below the table, there is a status bar with the text 'FT Kserver Enterprise' and a value '45,137'.

Now just change the source of data on the datalogger Object



## Let's test The Optix Runtime

It takes a while until the values are logged. I have also restarted the OPC UA server.

	Timestamp	LocalTimestamp	VariableToSample1	Id
147	2023-01-21 19:15:31.540	2023-01-21 20:15:31.540	0	1...
148	2023-01-21 19:15:32.540	2023-01-21 20:15:32.540	0	1...
149	2023-01-21 19:15:33.540	2023-01-21 20:15:33.540	0	1...
150	2023-01-21 19:15:50.237	2023-01-21 20:15:50.237	46471	1...
151	2023-01-21 19:15:51.237	2023-01-21 20:15:51.237	46481	1...
152	2023-01-21 19:15:52.237	2023-01-21 20:15:52.237	46491	1...

Clear the Table again

First time we see nothing on the Optix runtime datagrid window

Let's close the runtime and open it again

Now it works

FTOptixRuntime

Timestamp	LocalTimestamp	VariableToSample1
Jan 21, 2023, 7:20:32 PM	Jan 21, 2023, 8:20:32 PM	46,666
Jan 21, 2023, 7:20:33 PM	Jan 21, 2023, 8:20:33 PM	46,676
Jan 21, 2023, 7:20:34 PM	Jan 21, 2023, 8:20:34 PM	46,686

FT Keserver Enterprise 46,959

Let's put an autorefresh to the datagrid and sort ascending, descending to see it live!

FT Optix C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\OPC\_2\_SQL\_v0\*

Properties

- Name: DataGrid1
- Type: Data grid
- Model: ODBCDatabase1
- Query: SELECT \* FROM "DataLogger1"
- Auto refresh time: 0000:00:01.000

FT OptixRuntime

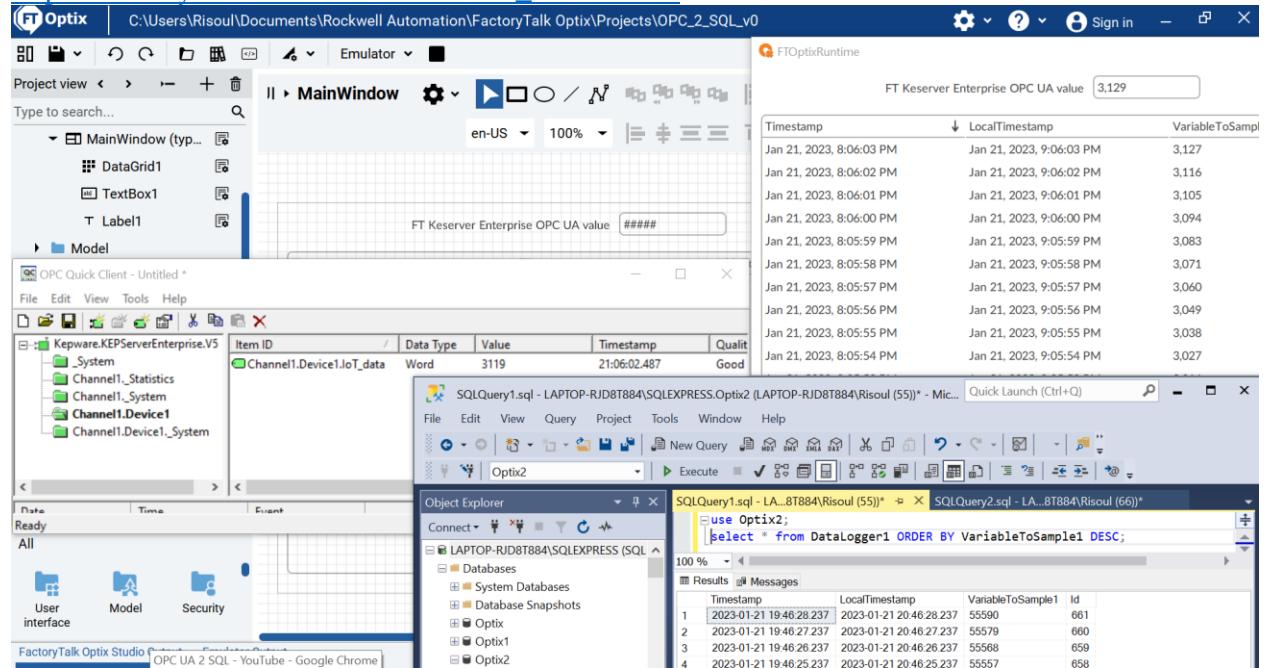
FT Keserver Enterprise 48,296

Timestamp	LocalTimestamp	VariableToSample1
Jan 21, 2023, 7:28:52 PM	Jan 21, 2023, 8:28:52 PM	48,290
Jan 21, 2023, 7:28:51 PM	Jan 21, 2023, 8:28:51 PM	48,280
Jan 21, 2023, 7:28:50 PM	Jan 21, 2023, 8:28:50 PM	48,270
Jan 21, 2023, 7:28:49 PM	Jan 21, 2023, 8:28:49 PM	48,260
Jan 21, 2023, 7:28:48 PM	Jan 21, 2023, 8:28:48 PM	48,250
Jan 21, 2023, 7:28:47 PM	Jan 21, 2023, 8:28:47 PM	48,239
Jan 21, 2023, 7:28:46 PM	Jan 21, 2023, 8:28:46 PM	48,229
Jan 21, 2023, 7:28:45 PM	Jan 21, 2023, 8:28:45 PM	48,219
Jan 21, 2023, 7:28:44 PM	Jan 21, 2023, 8:28:44 PM	48,209
Jan 21, 2023, 7:28:43 PM	Jan 21, 2023, 8:28:43 PM	48,199
Jan 21, 2023, 7:28:42 PM	Jan 21, 2023, 8:28:42 PM	48,189
Jan 21, 2023, 7:28:41 PM	Jan 21, 2023, 8:28:41 PM	48,179
Jan 21, 2023, 7:28:40 PM	Jan 21, 2023, 8:28:40 PM	48,169

As you can see on this video

<https://youtu.be/Bcs8rT0ytUQ>

[https://www.youtube.com/watch?v=irL\\_CHUZaHY](https://www.youtube.com/watch?v=irL_CHUZaHY)



## 15. Subscribing to an MQTT broker with user and password. TTN example

Just use the example from chapter 12  
But change some lines on subcribelogic  
From

```
// Connect to the broker
subscribeClient.Connect("FTOptixSubscribeClient");
```

to

```
// Connect to the broker
subscribeClient.Connect("FTOptixSubscribeClient", "username", "your_password");
```

That's all  
For example suscribing to TTN broker (eu1.cloud.thethings.network)  
Then simulate an uplink

eu1.cloud.thethings.network/console/applications/smartbridge-sinci-demo/devices/smartbridge-mod... Node-RED : node-r... TTN Login TTN Login | Microsoft 365

THE THINGS STACK Community Edition

Applications > smartbridge-sinci-demo > End devices > smartbridge-modbus-demo

**smartbridge-modbus-demo**  
ID: smartbridge-modbus-demo

↑ 997 ↓ 20 • Last activity 1 minute ago

Overview Live data **Messaging** Location Payload formatters Claiming General settings

Uplink Downlink

**Simulate uplink**

FPort \*

**Payload**

00 01 93 13 88 00 00 00 00 00 00  
The desired payload bytes of the uplink message

**Simulate uplink**

eu1.cloud.thethings.network/console/applications/smartbridge-sinci-demo/devices/smartbridge-modbus-demo/data

Rockwell Node-RED : node-r... TTN Login TTN Login | Microsoft 365

Overview Applications Gateways Organizations

Applications > smartbridge-sinci-demo > End devices > smartbridge-modbus-demo > Live data

**smartbridge-modbus-demo**  
ID: smartbridge-modbus-demo

↑ 998 ↓ 20 • Last activity 1 minute ago

Overview Live data **Messaging** Location Payload formatters Claiming General settings

Time Type Data preview

↑ 20:41:17 Forward uplink data message Payload: { Speed\_Hz: 50 } 00 01 93 13 88 00 00 00 ... FPort: 1 Data rate: SF7BW125 SNR: 4.2 RSSI: 42

Uplink



This is how the subscriber logic looks like for this use case:

```
// Create a client connecting to the broker (default port is 1883)
    subscribeClient = new MqttClient(brokerIpAddressVariable.Value);
    // Connect to the broker
    subscribeClient.Connect("FTOptixSubscribeClient", "smartbridge-sinci-
demo@ttn", "this_is_my_TTN_tocken");

    // Subscribe to the "my_topic" topic with QoS 2
    ushort msgId = subscribeClient.Subscribe(new string[] { "v3/smartbridge-
sinci-demo@ttn/devices/smartbridge-modbus-demo/up" }, // topic

public override void Stop()
{
    subscribeClient.Unsubscribe(new string[] { "v3/smartbridge-sinci-
demo@ttn/devices/smartbridge-modbus-demo/up" });
    subscribeClient.Disconnect();
}

private void SubscribeClientMqttMsgPublishReceived(object sender,
MqttMsgPublishEventArgs e)
{
    //messageVariable.Value = "Message received: " +
System.Text.Encoding.UTF8.GetString(e.Message);
    messageVariable.Value = "Message received: " +
System.Text.Encoding.Default.GetString(e.Message);

}
```

But what if we want to get only the number after “Speed\_Hz:” then we use the substring to cut and get only the number

This is how the subscriber code looks like

```
#region StandardUsing
using System;
```

```

using FTOptix.CoreBase;
using FTOptix.HMIProject;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.NetLogic;
using FTOptix.UI;
using FTOptix.OPCUAServer;
#endregion
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
using FTOptix.RAEtherNetIP;
using FTOptix.CommunicationDriver;

public class SubscriberLogic : BaseNetLogic
{
    public override void Start()
    {
        var brokerIpAddressVariable =
Project.Current.GetVariable("Model/BrokerIpAddress");

        // Create a client connecting to the broker (default port is 1883)
        subscribeClient = new MqttClient(brokerIpAddressVariable.Value);
        // Connect to the broker
        subscribeClient.Connect("FTOptixSubscribeClient","smartbridge-sinci-
demo@ttn","mytocken");
        // Assign a callback to be executed when a message is received from the
        broker
        subscribeClient.MqttMsgPublishReceived +=
SubscribeClientMqttMsgPublishReceived;
        // Subscribe to the "my_topic" topic with QoS 2
        ushort msgId = subscribeClient.Subscribe(new string[] { "v3/smartbridge-
sinci-demo@ttn/devices/smartbridge-modbus-demo/up" }, // topic
            //new byte[] { MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE }); // QoS level
            new byte[] { MqttMsgBase.QOS_LEVEL_AT_MOST_ONCE }); // QoS level
        messageVariable = Project.Current.GetVariable("Model/Message");
    }

    public override void Stop()
    {
        subscribeClient.Unsubscribe(new string[] { "v3/smartbridge-sinci-
demo@ttn/devices/smartbridge-modbus-demo/up" });
        subscribeClient.Disconnect();
    }
}

```

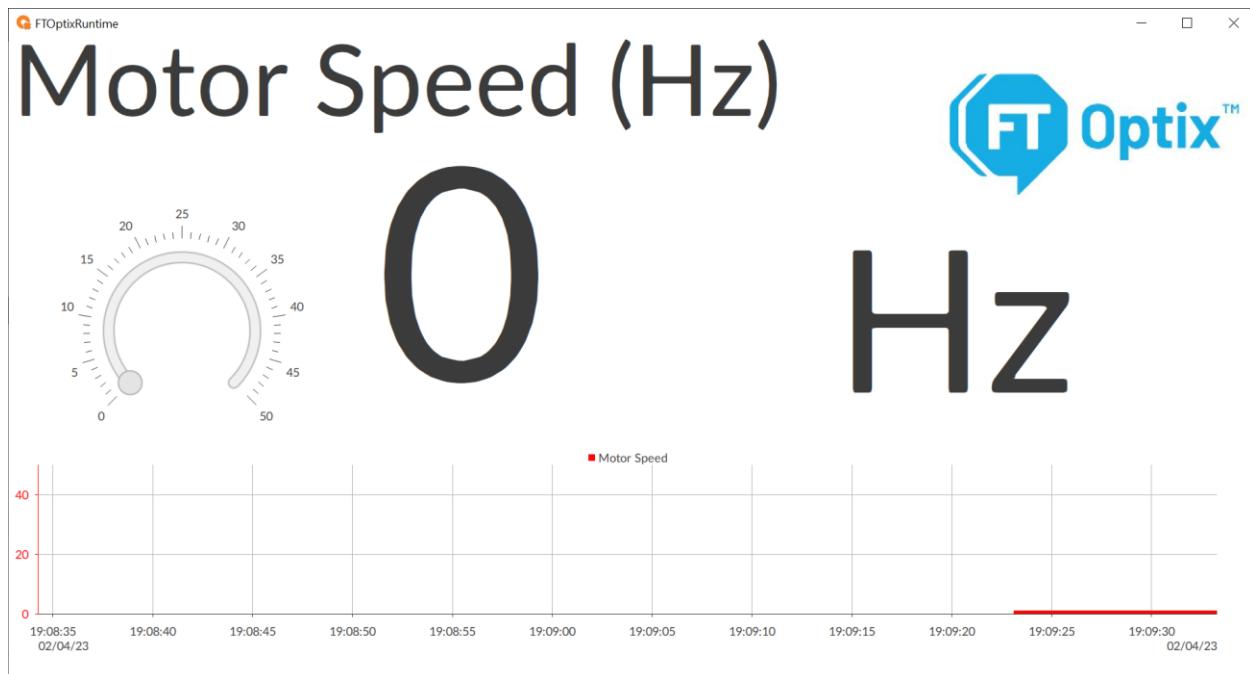
```

private void SubscribeClientMqttMsgPublishReceived(object sender,
MqttMsgPublishEventArgs e)
{
    //messageVariable.Value = "Message received: " +
System.Text.Encoding.UTF8.GetString(e.Message);
    string toBeSearched = "Speed_Hz";
    string myString="Message received: " +
System.Text.Encoding.Default.GetString(e.Message);
    //string code = myString.Substring(myString.IndexOf(toBeSearched) +
toBeSearched.Length);
    //string code = myString.Substring(myString.IndexOf(toBeSearched)+10,2);
    string code = myString.Substring(myString.IndexOf(toBeSearched)+10,1);
    if (code=="0")
    {
        code="00";
    }
    else
    {
        code = myString.Substring(myString.IndexOf(toBeSearched)+10,2);
    }
    messageVariable.Value = code;
}

private MqttClient subscribeClient;
private IUAVariable messageVariable;
}

```

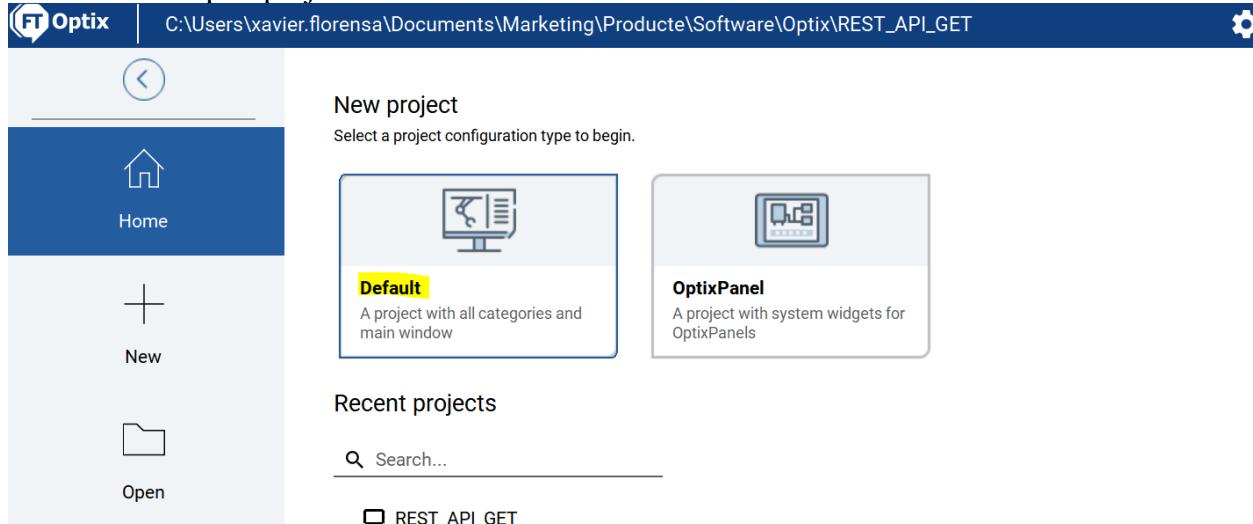
And this is the runtime screenshot



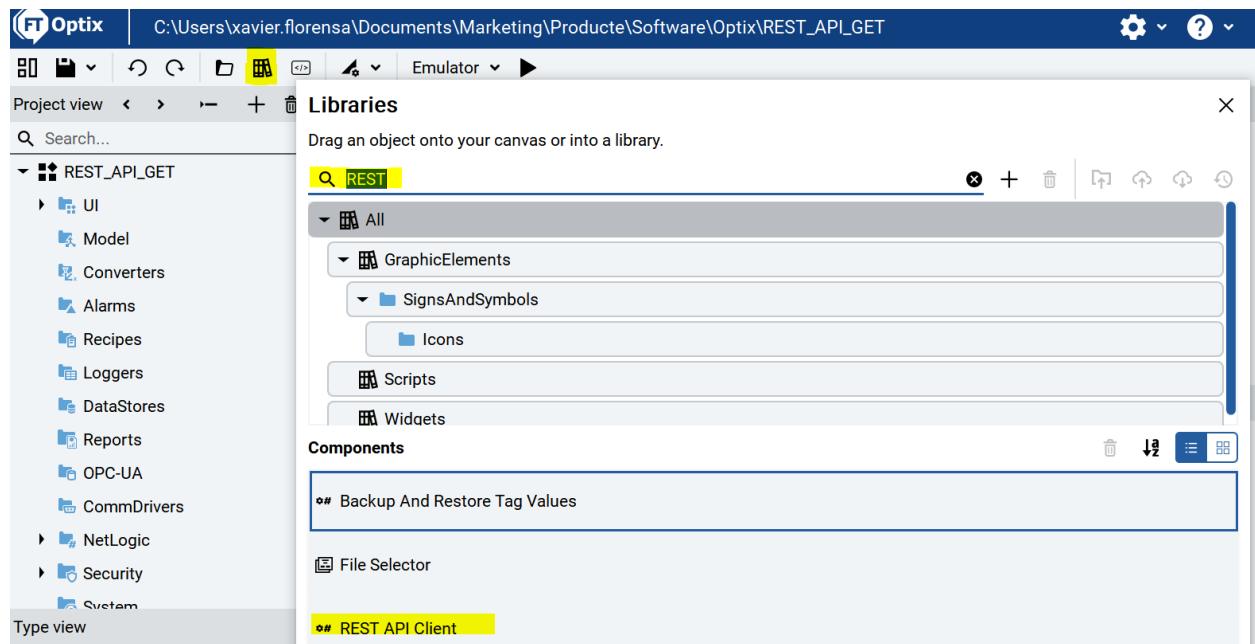
## 16. Creating an Http REST API client

### 16.1. GET request

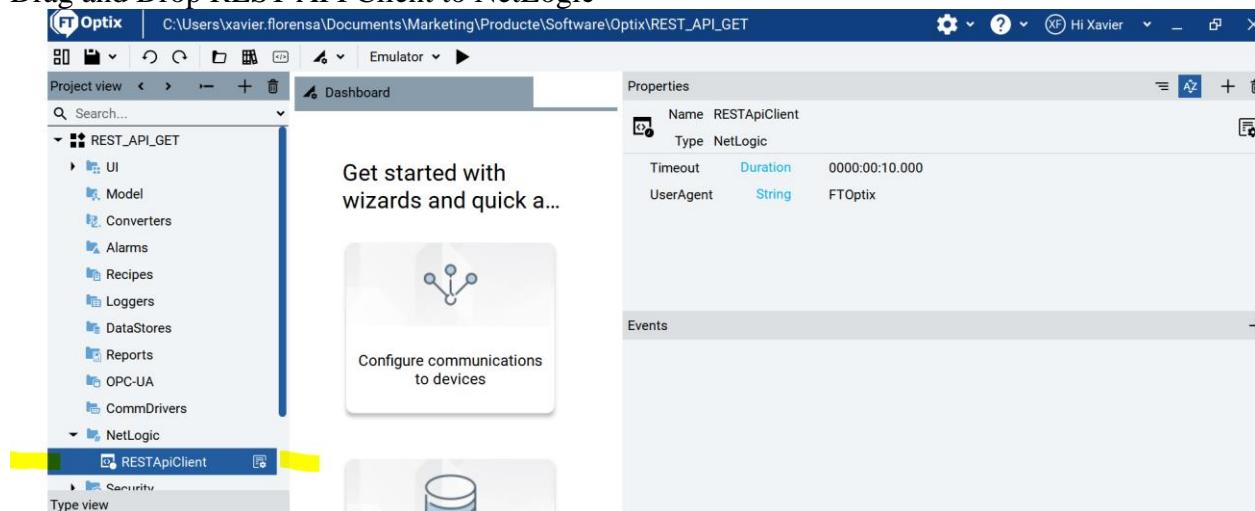
Create a new Optix project



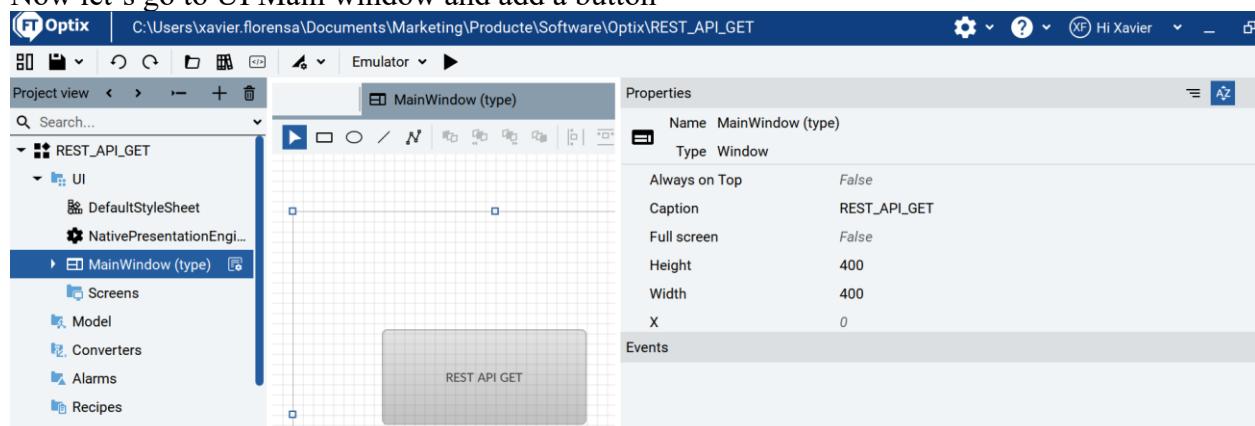
Go to libraries



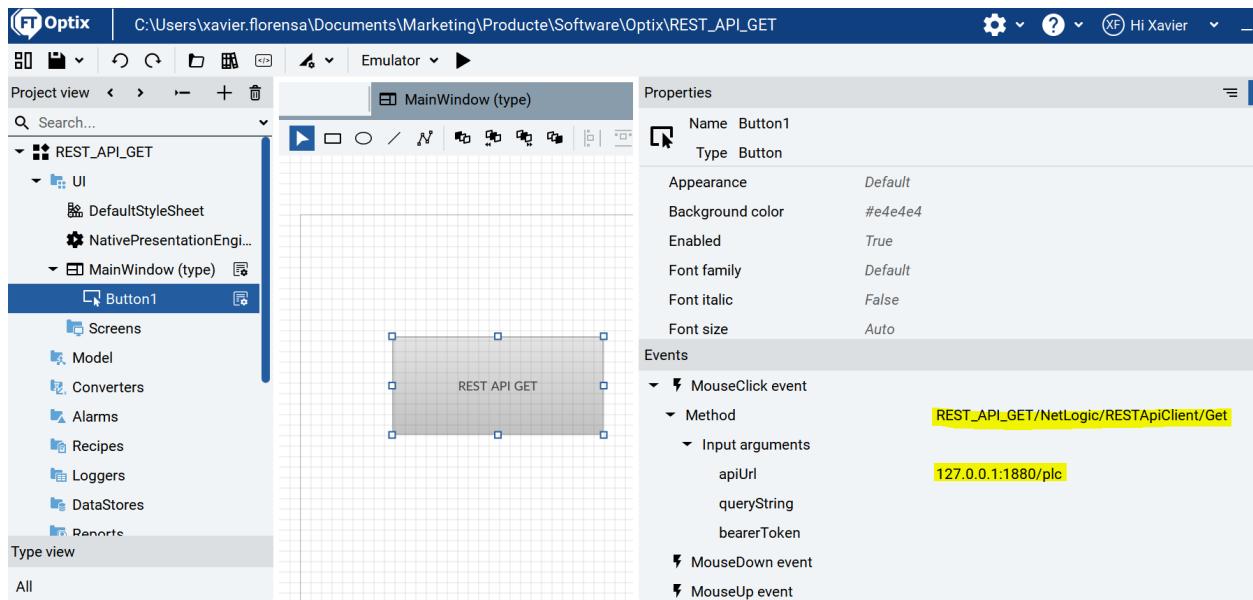
### Drag and Drop REST API Client to NetLogic



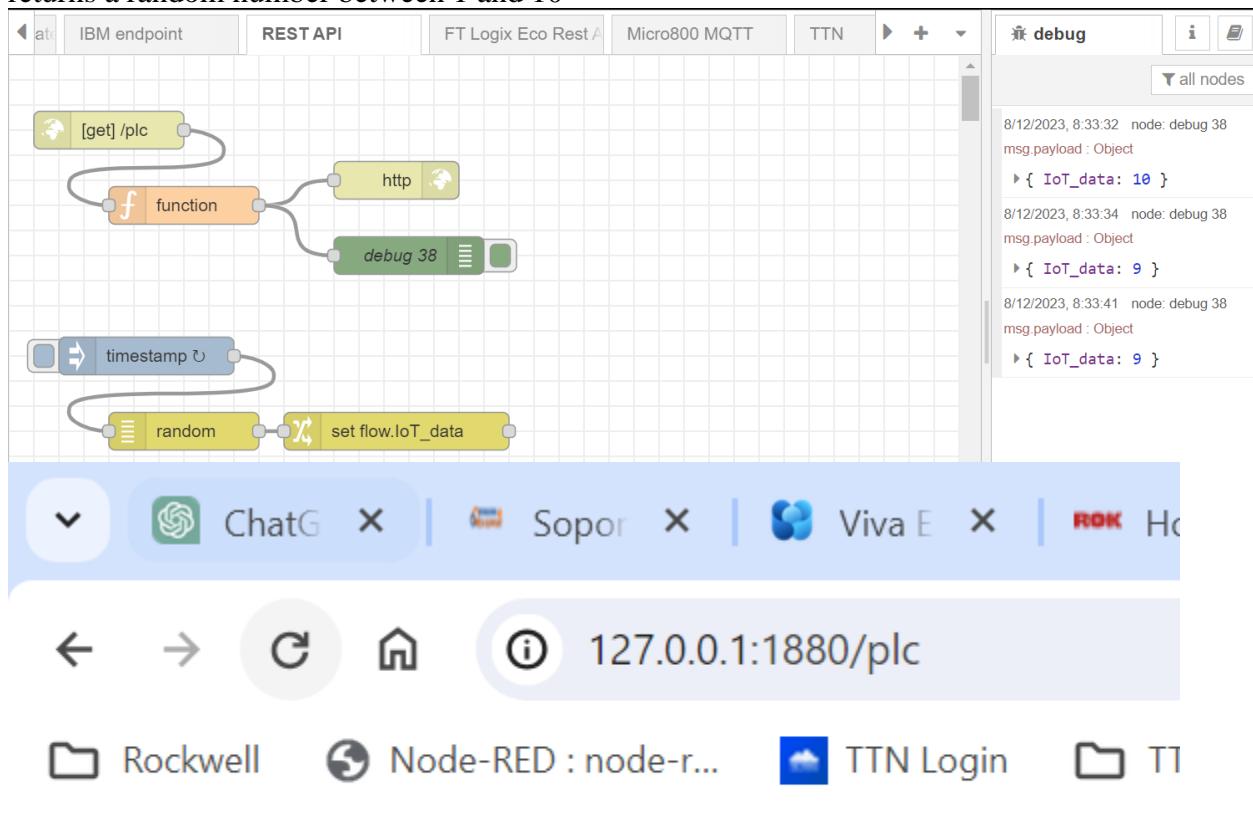
Now let's go to UI Main window and add a button



Click on the button to see the properties and add a Mouse Click event with the URL of the API



First of all we have to test these API URL, we are using node red here, to create an API that returns a random number between 1 and 10



```
{"IoT_data":6}
```

The screenshot shows a web browser window with the URL `127.0.0.1:1880/plc`. The page displays the result of a Node-RED flow execution, specifically the output of a function node which returned the JSON object `{"IoT_data":7}`.

Below the browser, the Node-RED interface is visible, showing the flow structure and the debug log on the right side.

Now let's test the Optix runtime  
This is working fine

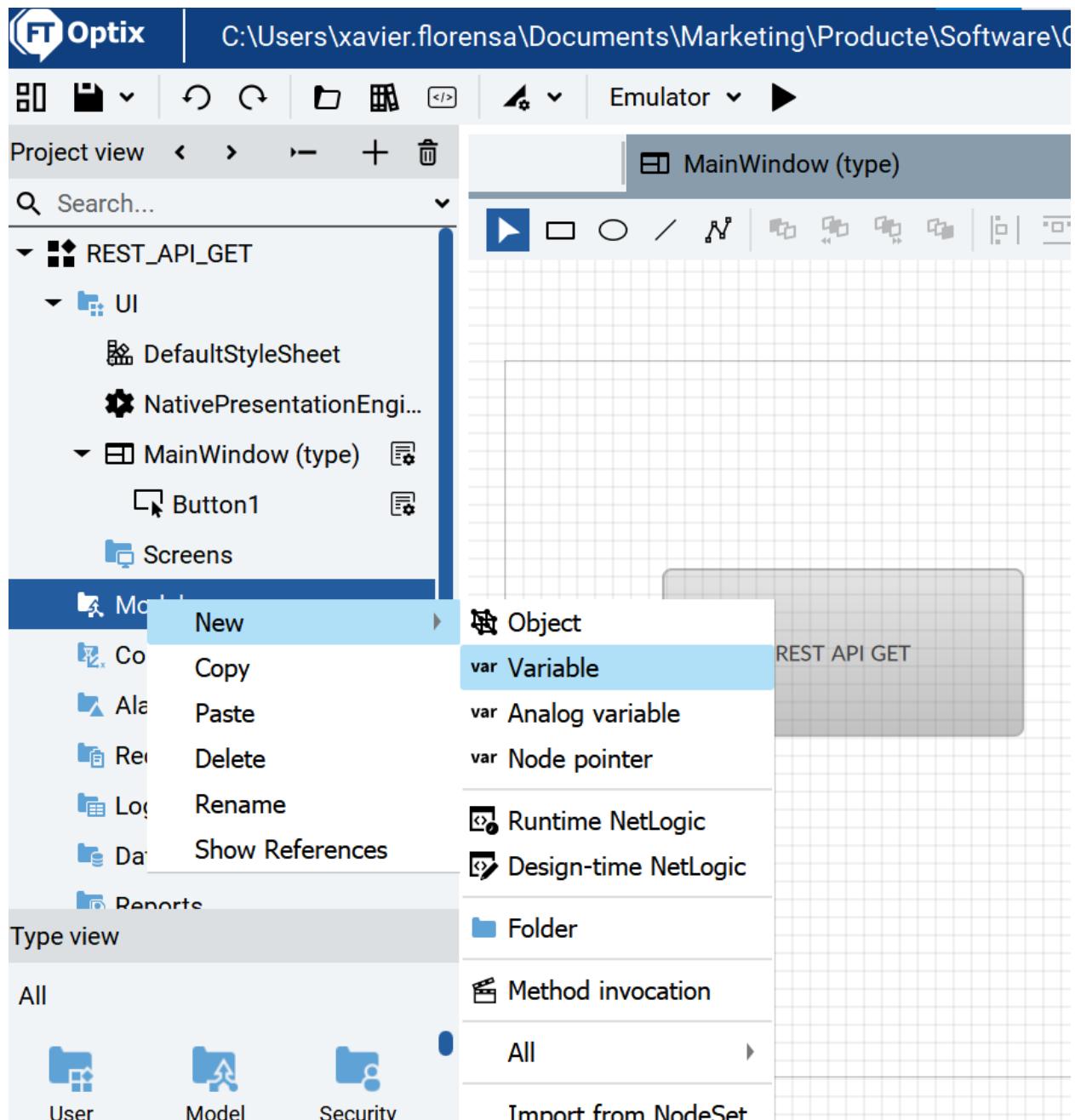
The screenshot shows the Optix runtime interface with a successful REST API GET request. The response body contains the JSON object `{"IoT_data":7}`.

On the left, there is a sidebar with options like Loggers, DataStores, and Reports. On the right, the Node-RED interface is shown again, with the flow running and the debug log displaying the same JSON object.

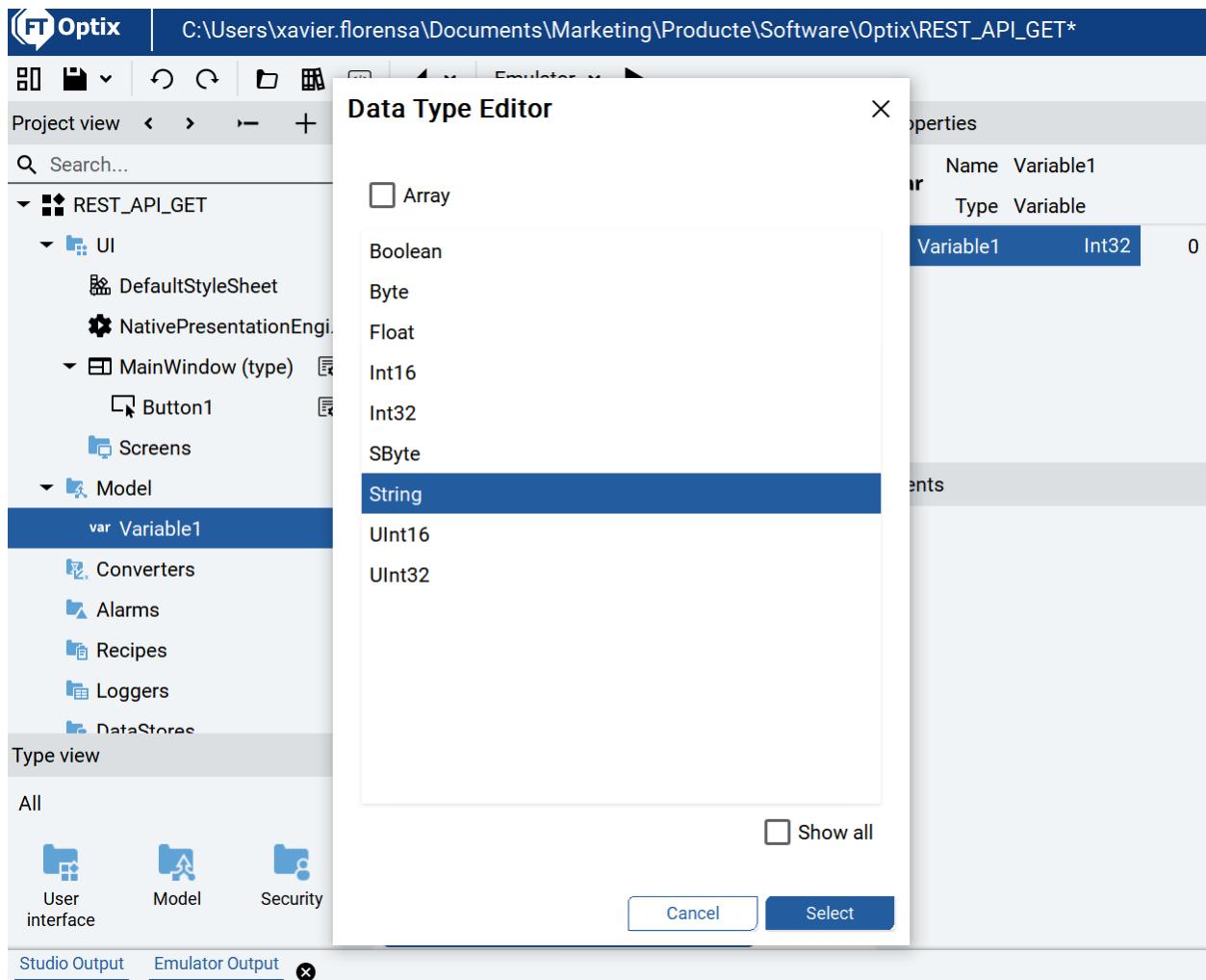
But we need to get the data on Optix  
Let's open the C# script

```
1  #region Using directives
2  using System;
3  using UAManagedCore;
4  using OpcUa = UAManagedCore.OpcUa;
5  using FTOptix.HMIProject;
6  using FTOptix.NetLogic;
7  using FTOptix.UI;
8  using FTOptix.Retentivity;
9  using FTOptix.NativeUI;
10 using FTOptix.CoreBase;
11 using FTOptix.Core;
12 using System.Net.Http.Headers;
13 using System.Net.Http;
14 using System.Threading.Tasks;
15 #endregion
16
17 public class RESTApiClient : BaseNetLogic
18 {
19     readonly struct HTTPResponse
20     {
21         public HTTPResponse(string payload, int code)
22         {
23             Payload = payload;
24             Code = code;
25         }
26
27         public string Payload { get; }
28         public int Code { get; }
29     };
30 }
```

Let's create a new variable to store the response data



And change variable type



Now introduce the variable somehow on the script

```
var Variable1 = Project.Current.GetVariable("Model/Variable1");
```

So you can update the variable value like this

```
Variable1.Value = System.Text.Encoding.UTF8.GetString(Response);
```

Where response is the result of the Http request

We can introduce our variable for instance here

```

[ExportMethod]
public void Get(string apiUrl, string queryString, string bearerToken, out string response, out int
{
    TimeSpan timeout = TimeSpan.FromMilliseconds(GetTimeout());
    UriBuilder uriBuilder = new UriBuilder(apiUrl);
    uriBuilder.Query = queryString;

    var requestMessage = BuildMessage(HttpMethod.Get, uriBuilder.Uri, "", bearerToken, "");
    var requestTask = PerformRequest(requestMessage, timeout);
    var httpResponse = requestTask.Result;

    (response, code) = (httpResponse.Payload, httpResponse.Code);
}

```

Like this

```

[ExportMethod]
public void Get(string apiUrl, string queryString, string bearerToken, out string response, out int
{
    //
    var Variable1 = Project.Current.GetVariable("Model/Variable1");
    Variable1.Value = "Hello World"
    //

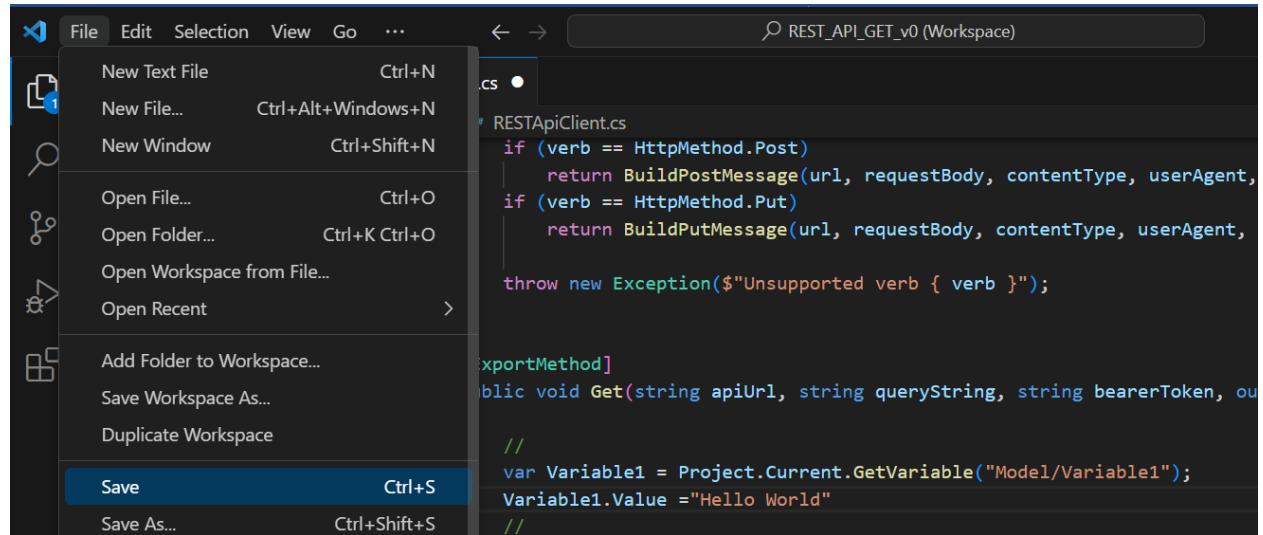
    TimeSpan timeout = TimeSpan.FromMilliseconds(GetTimeout());
    UriBuilder uriBuilder = new UriBuilder(apiUrl);
    uriBuilder.Query = queryString;

    var requestMessage = BuildMessage(HttpMethod.Get, uriBuilder.Uri, "", bearerToken, "");
    var requestTask = PerformRequest(requestMessage, timeout);
    var httpResponse = requestTask.Result;

    (response, code) = (httpResponse.Payload, httpResponse.Code);
}

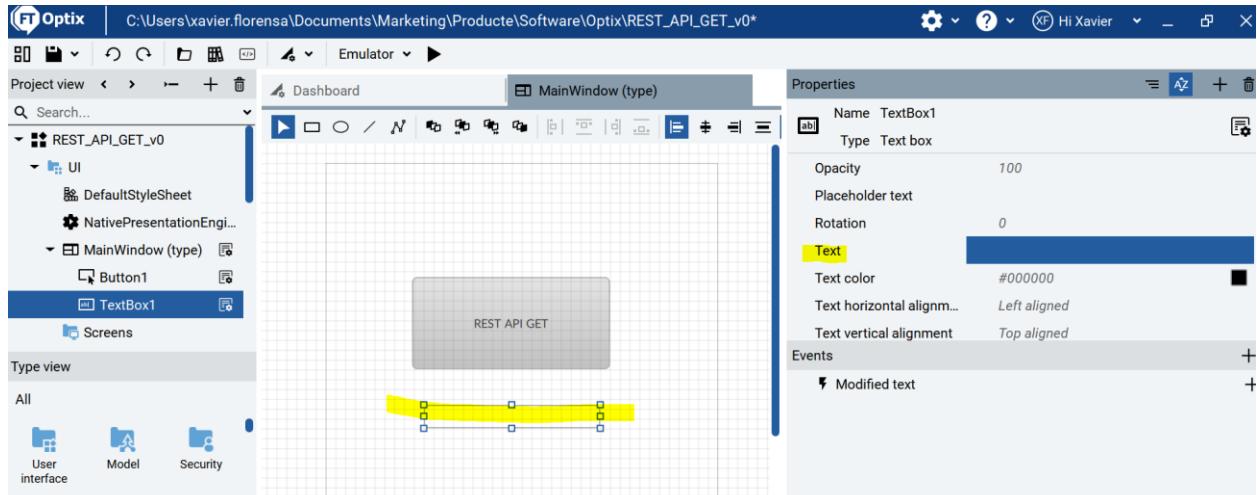
```

Save the file

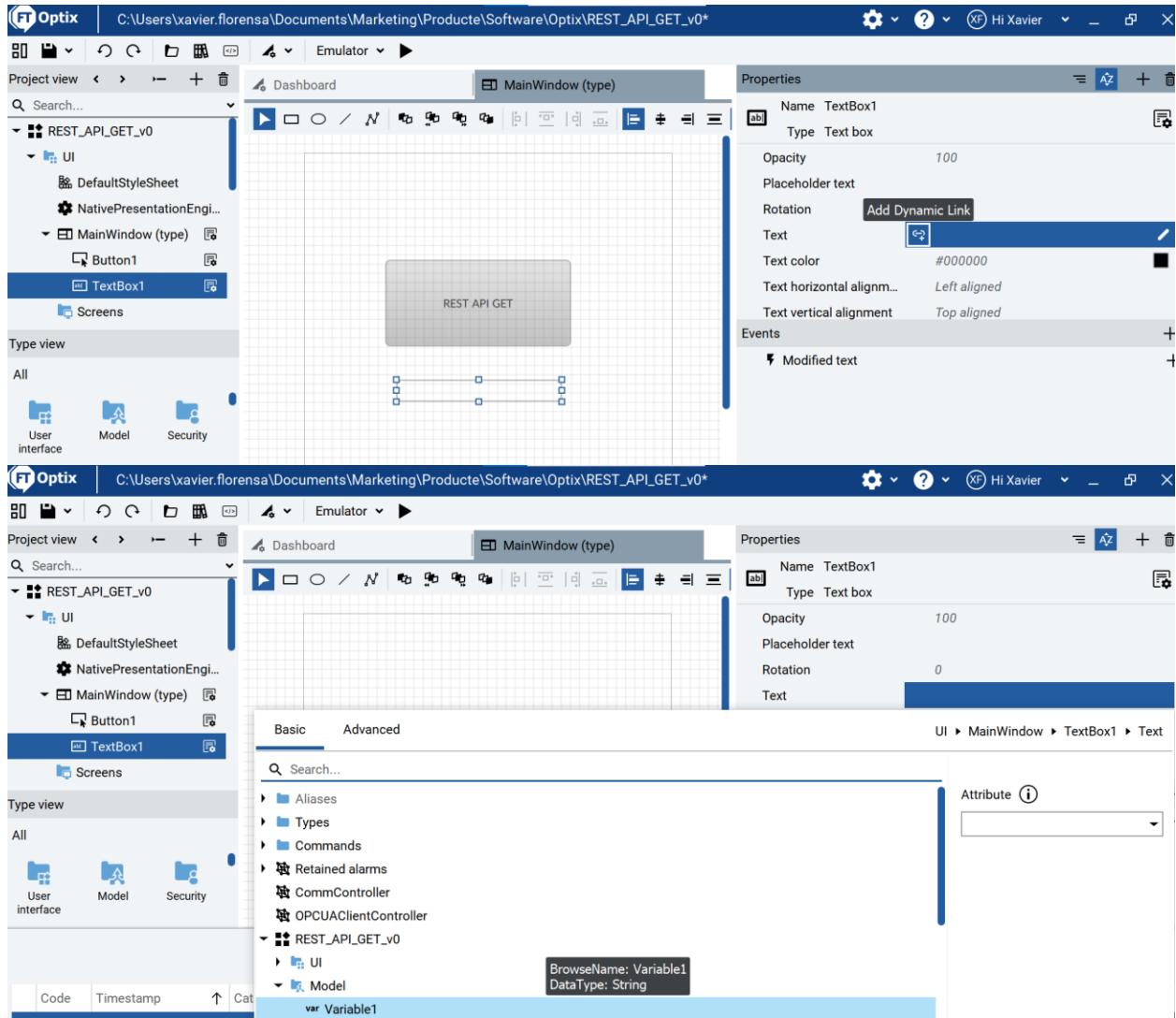


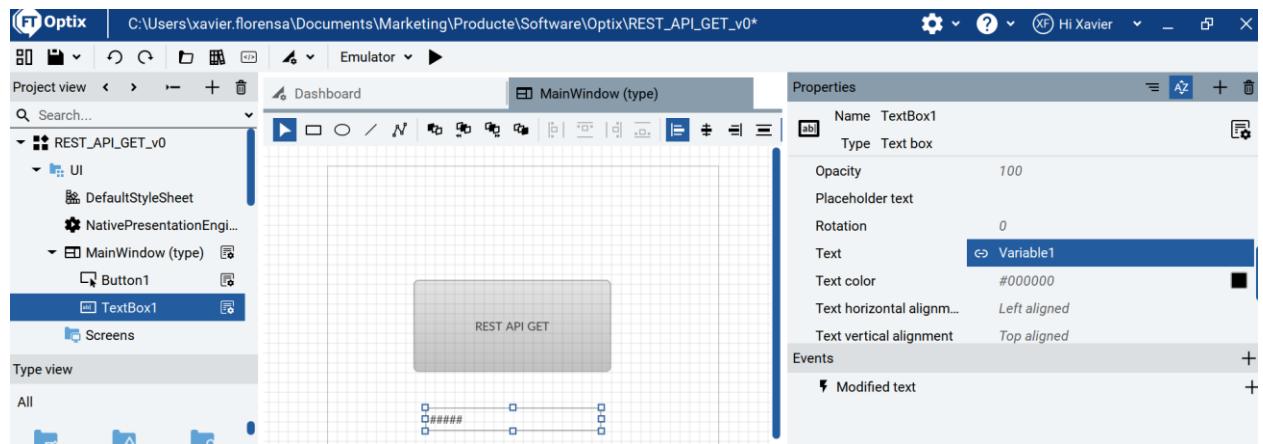
2023-12-08 09:17:07.807;NetHelper;222;Warning;16;User .NET solution failed to build:  
C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\REST\_API\_GET\_v0\ProjectFiles\NetSolution\RESTApiClient.cs(152,39): error CS1002: ; expected  
[C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\REST\_API\_GET\_v0\ProjectFiles\NetSolution\REST\_API\_GET\_v0.csproj]  
Let's correct this

## Now introduce a text box

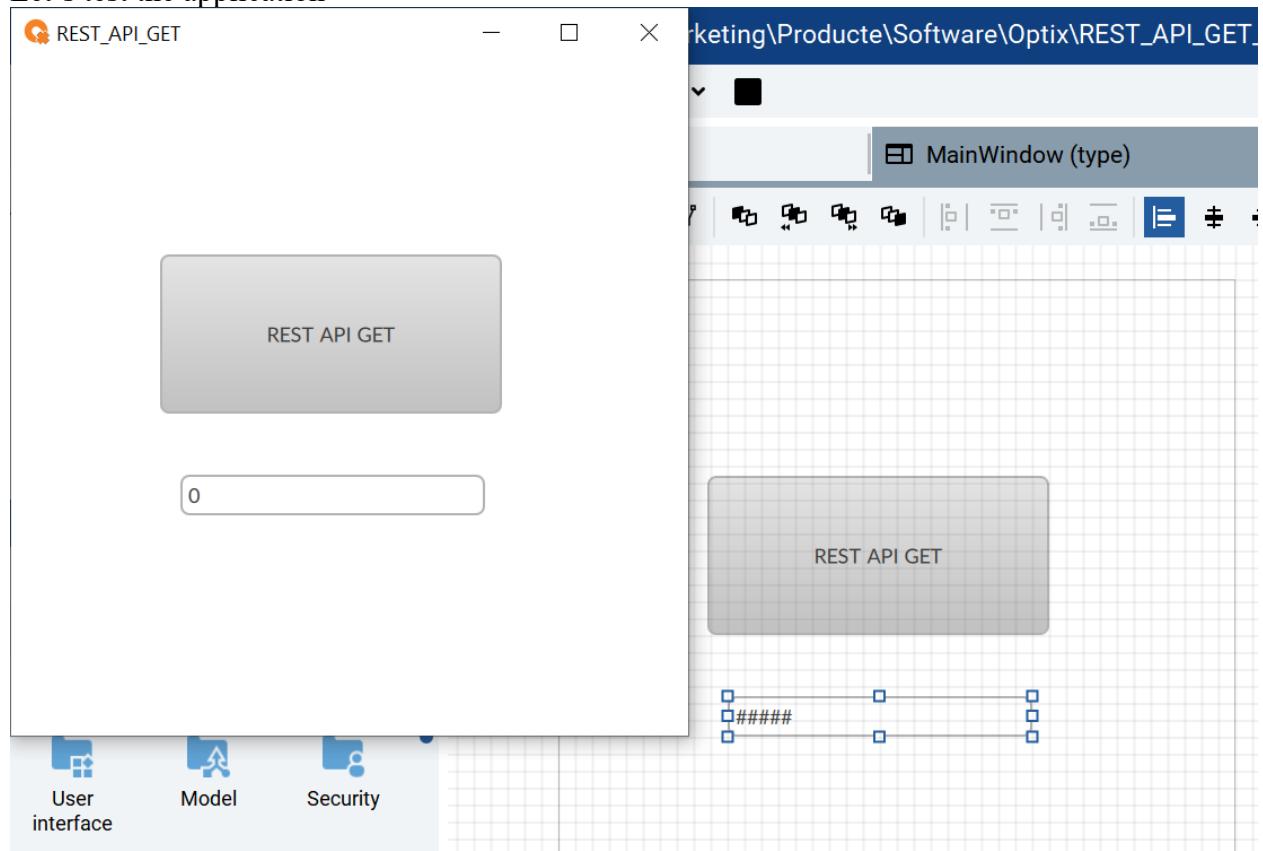


## And link to Variable1

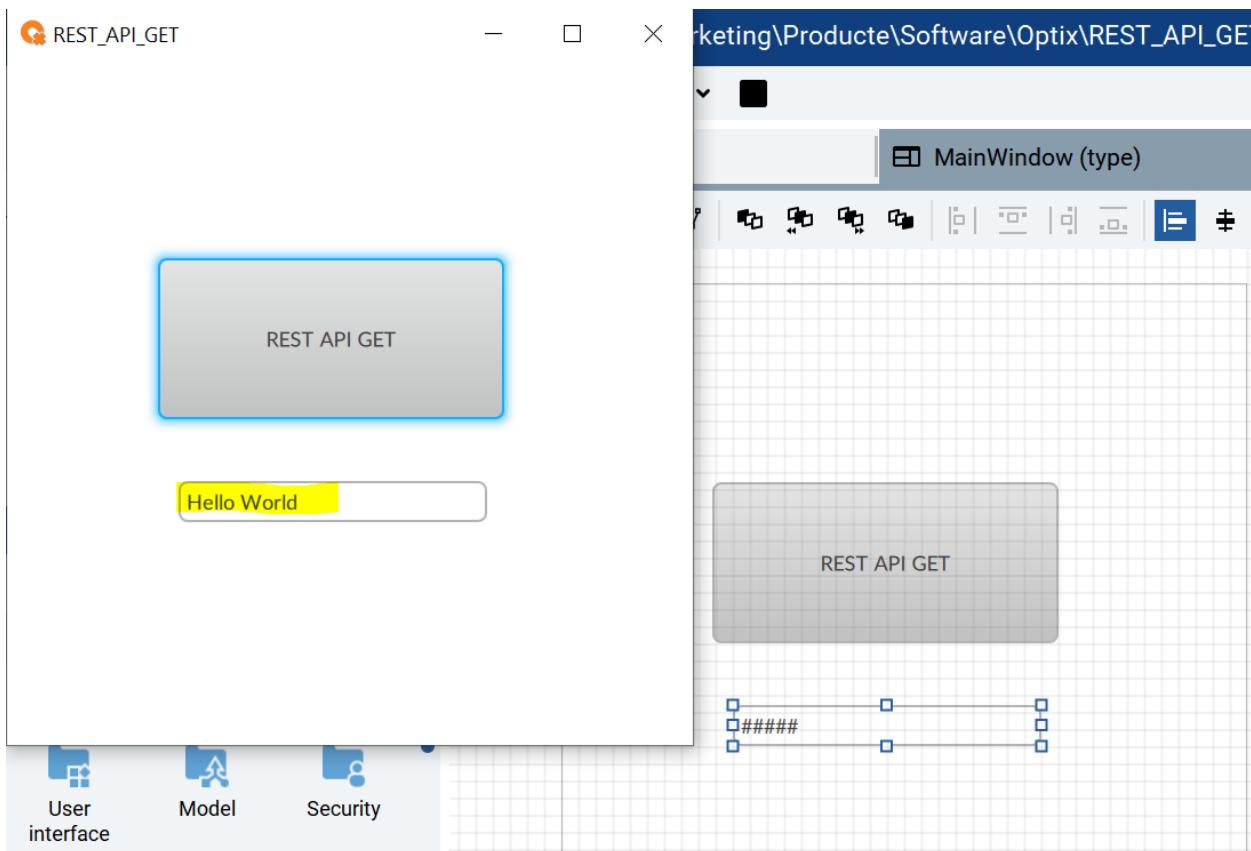




Let's test the application



Voilà

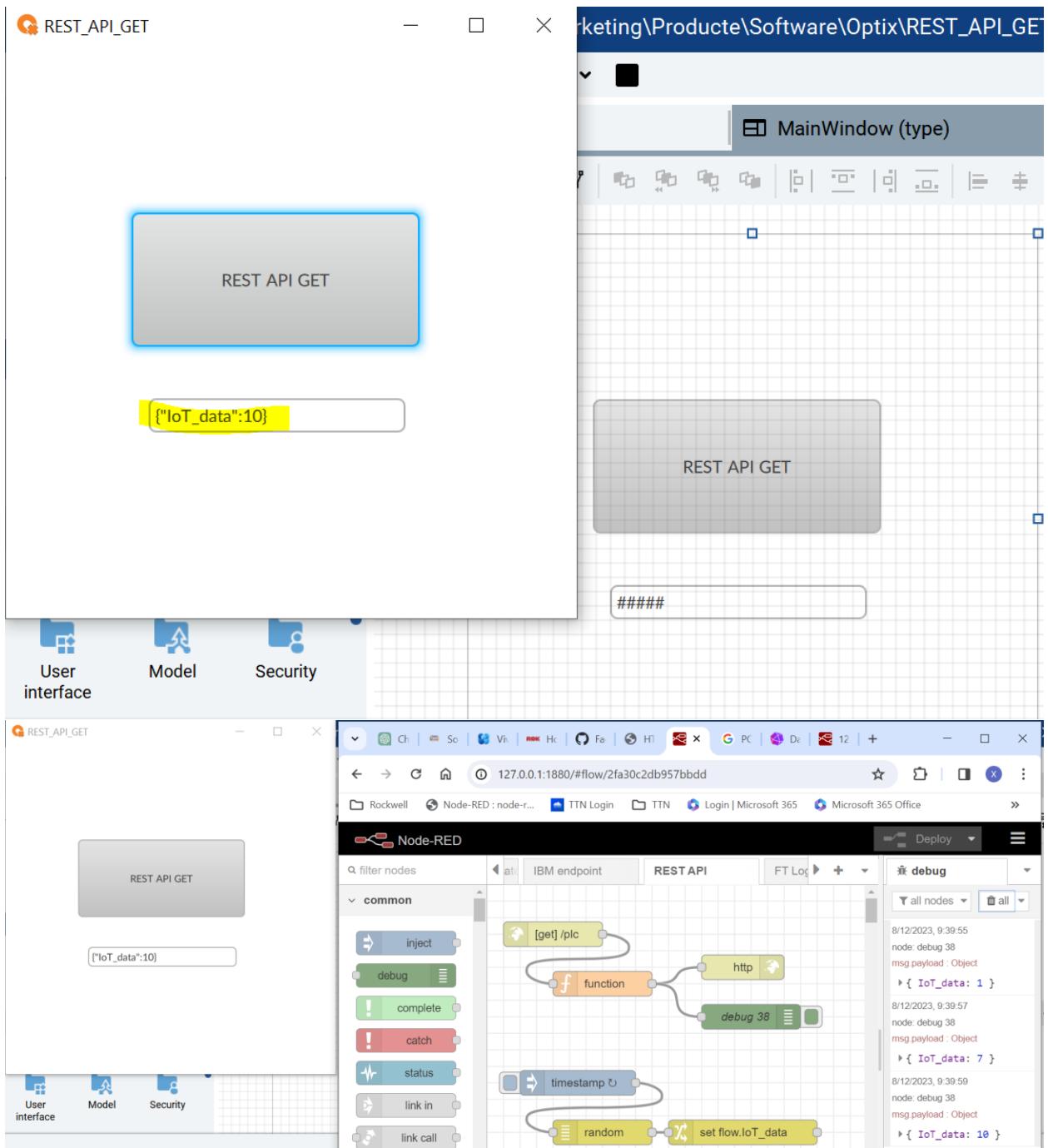


Not let's try to get the http response on that Textbox

```

147 [ExportMethod]
148 public void Get(string apiUrl, string queryString, string bearerToken, out string
149 [
150     //var Variable1 = Project.Current.GetVariable("Model/Variable1");
151     //Variable1.Value ="Hello World";
152     //
153     TimeSpan timeout = TimeSpan.FromMilliseconds(GetTimeout());
154     UriBuilder uriBuilder = new UriBuilder(apiUrl);
155     uriBuilder.Query = queryString;
156
157     var requestMessage = BuildMessage(HttpMethod.Get, uriBuilder.Uri, "", bearerTo
158     var requestTask = PerformRequest(requestMessage, timeout);
159     var httpResponse = requestTask.Result;
160     //
161     Variable1.Value =httpResponse.Payload;
162     //
163     (response, code) = (httpResponse.Payload, httpResponse.Code);
164 ]
165 
```

Voilà, it works



We have created a API Rest client to perform a GET Http request  
As you can see on this video  
<https://youtu.be/-jf5SwSq8Us>

## 16.2. Installing InfluxDB on windows

Using PowerShell in Administrator mode  
Download with this command

```
wget https://dl.influxdata.com/influxdb/releases/influxdb2-2.7.3-windows.zip -UseBasicParsing -OutFile influxdb2-2.7.3-windows.zip  
Expand-Archive .\influxdb2-2.7.3-windows.zip -DestinationPath 'C:\Program Files\InfluxData\influxdb\'
```

From this guide

<https://portal.influxdata.com/downloads/>

## Start InfluxDB

In **Powershell**, navigate into `C:\Program Files\InfluxData\influxdb` and start InfluxDB by running the `influxd` daemon:

```
> cd -Path 'C:\Program Files\InfluxData\influxdb'  
> ./influxd  
  
> cd -Path 'C:\Program Files\InfluxData\influxdb'  
> ./influxd
```

`cd -Path 'C:\Program Files\InfluxData\influxdb'`

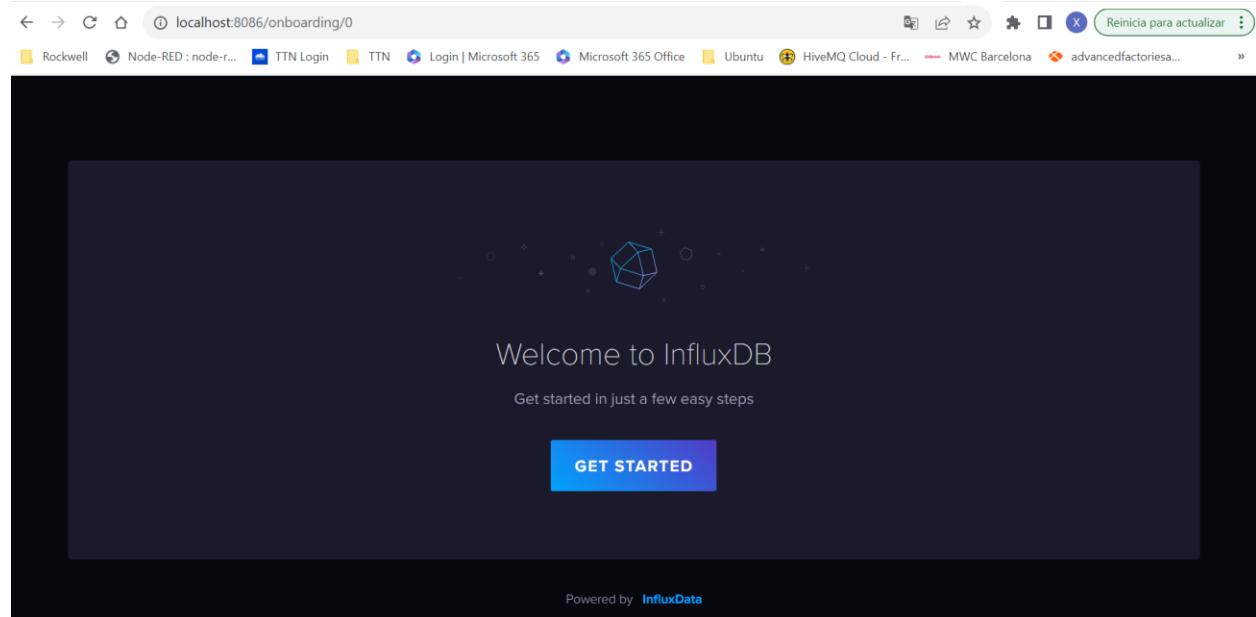
`./influxd`

Open an explorer with localhost and port 8086

Username

xavier

Password



Start building your influxdb use cases

localhost:8086/onboarding/1

Welcome — Initial User Setup — Complete

### Setup Initial User

You will be able to create additional Users, Buckets and Organizations later

Username: xavier

Password: ..... Confirm Password: .....

Initial Organization Name: Risoul

Initial Bucket Name: PLC

**CONTINUE**

API token

MwQvFkNF8uI\_Yx8327ohwgDG2qHBhO9ZbAqbpFPcFRX6amE9SooSyxAiA9zofuxj8c\_C26  
cf-zGmMeLyGYKgHA==

Select Quickstart

localhost:8086/orgs/45bd74e06c16b567

## Get Started

Write and query data using the programming language of your choice

- Python
- Node.js
- Go
- Arduino

InfluxDB CLI

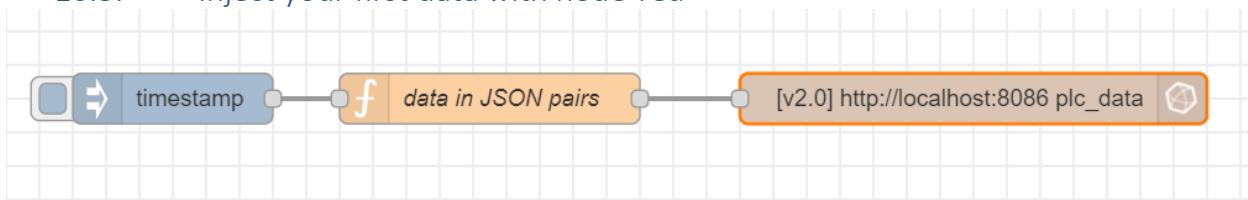
Search Documentation

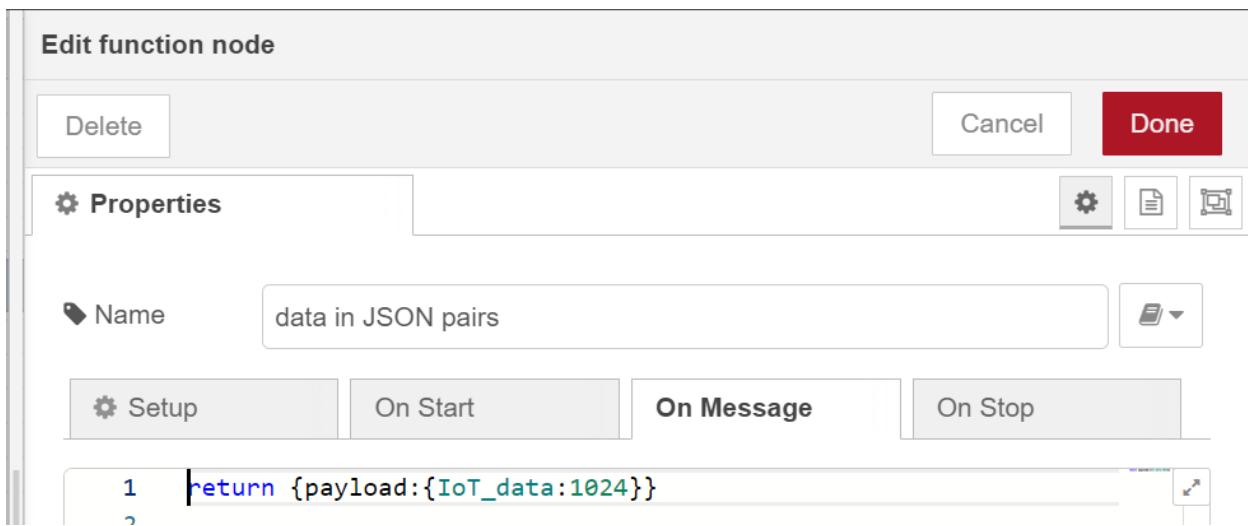
Press CTRL + M on any page to search

### USEFUL LINKS

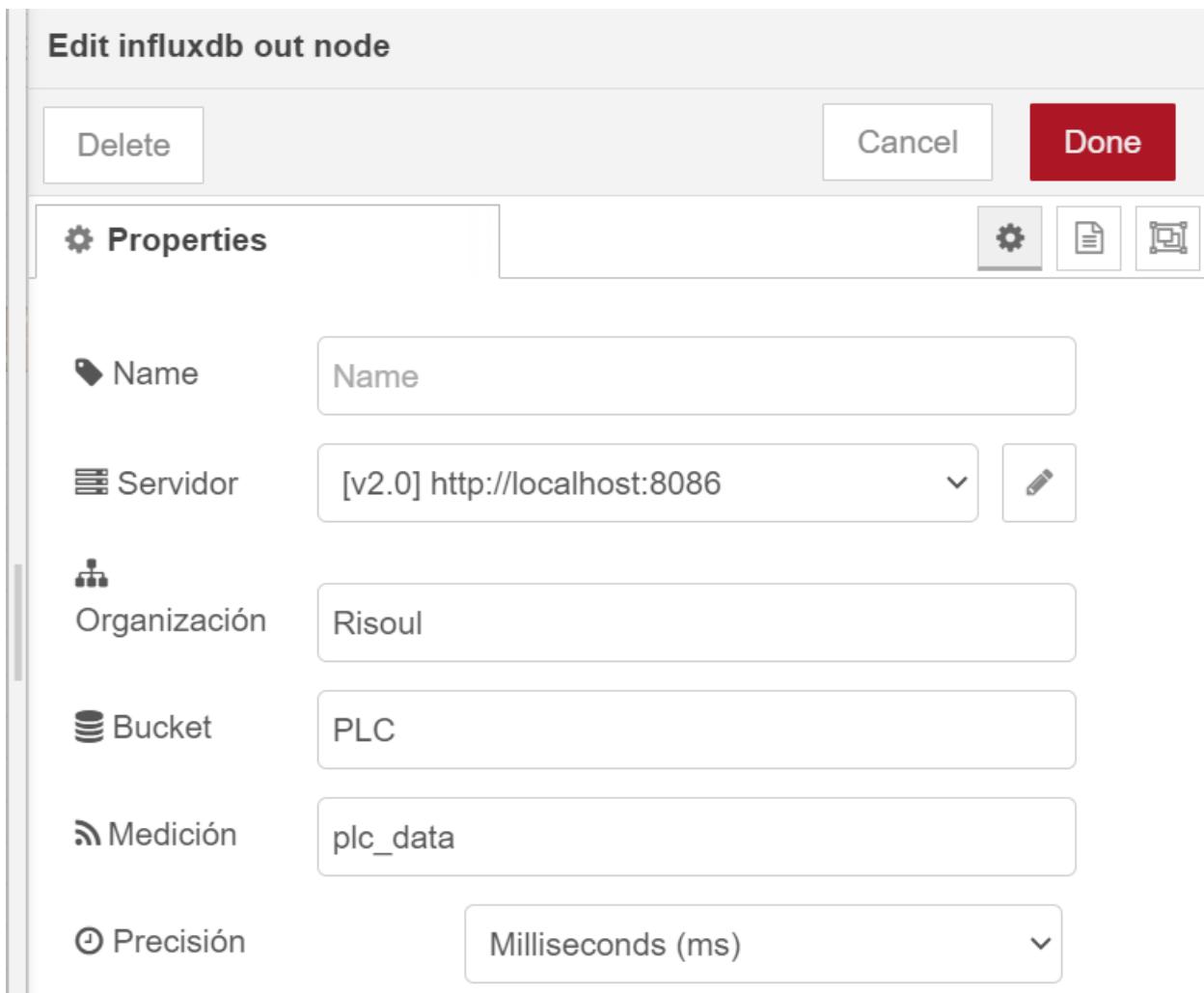
- InfluxDB University
- Get Started with Flux
- Explore Metrics
- Build a Dashboard
- Write a Task
- Report a bug
- Community Forum
- Feature Requests

### 16.3. Inject your first data with node-red





Let's try to inject to INfluxDB using an Http request



Edit influxdb out node > **Edit influxdb node**

[Delete](#) [Cancel](#) [Update](#)

**Properties**

Name	Name
Versión	2.0
URL	http://localhost:8086
Token	.....

Verificar el certificado del servidor

localhost:8086/orgs/45bd74e06c16b567/data-explorer?fluxScriptEditor

Rockwell Node-RED : node-red... TTN Login TTN Login | Microsoft 365 Microsoft 365 Office Ubuntu HiveMQ Cloud - Fr... MWC Barcelona advancedfactories...

### Data Explorer

Single Stat CUSTOMIZE Local SAVE AS

1024,00

Query 1 (0.01s) +

FROM Filter Filter

Search buckets  \_measurement  \_field

No tag keys found In the current time range

View Raw Data CSV Past 1h SCRIPT EDITOR SUBMIT

WINDOW PERIOD

CUSTOM AUTO

auto (10s)

Fill missing values

AGGREGATE FUNCTION

CUSTOM AUTO

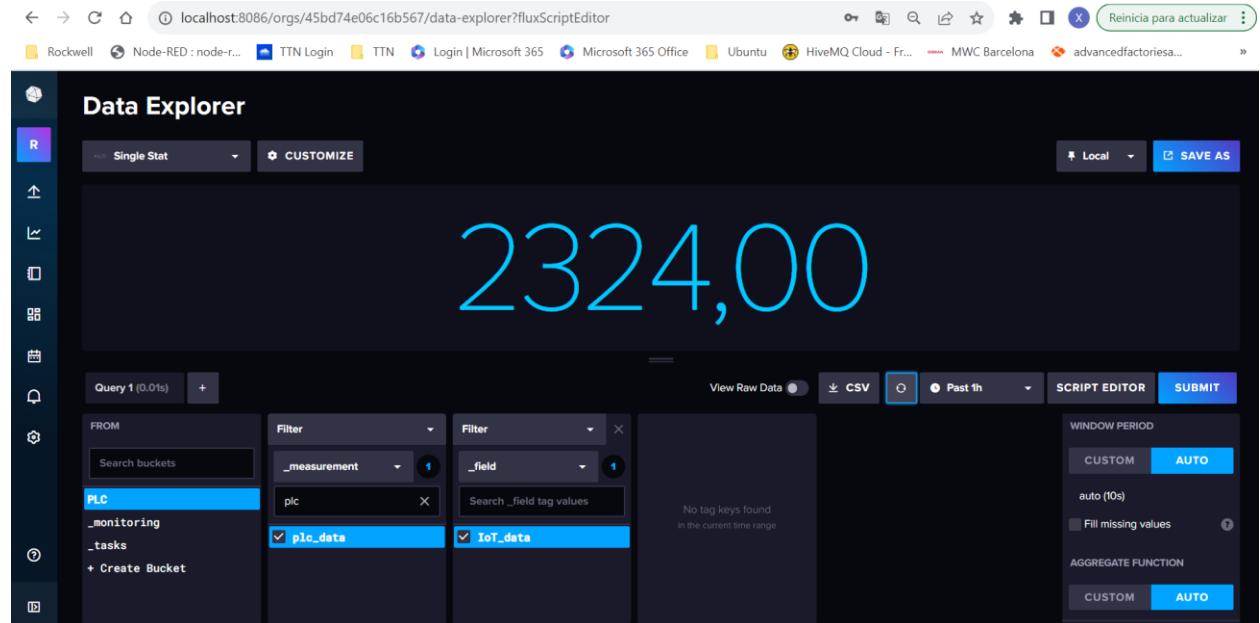
#### 16.4. Injecting to InfluxDB from command line

Now let's try to build the http request ourselves. From windows command line

```
curl -POST "http://127.0.0.1:8086/api/v2/write?org=Risoul&bucket=PLC&precision=s" --header "Authorization: Token"
```

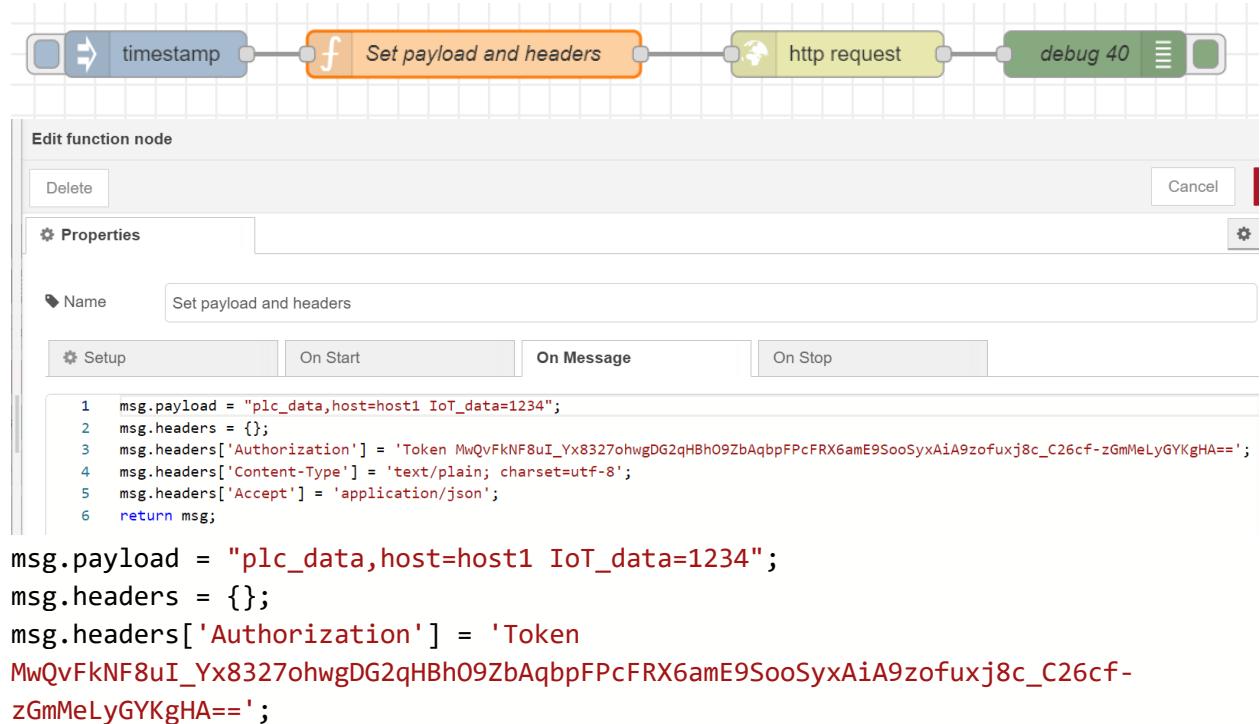
```
MwQvFkNF8uI_Yx8327ohwgDG2qHBhO9ZbAqbpFPcFRX6amE9SooSyxAiA9zofuxj8c_C26  
cf-zGmMeLyGYKgHA==" --data-raw "plc_data,host=host1 IoT_data=2324"
```

```
Command Prompt  
Microsoft Windows [Version 10.0.19045.3570]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\xavier.florensa>curl -POST "http://127.0.0.1:8086/api/v2/write?org=Risoul&bucket=PLC&precision=s" --header "Authorization: Token MwQvFkNF8uI_Yx8327ohwgDG2qHBhO9ZbAqbpFPcFRX6amE9SooSyxAiA9zofuxj8c_C26cf-zGmMeLyGYKgHA==" --data-raw "plc_data,host=host1 IoT_data=2324"  
  
C:\Users\xavier.florensa>
```



The screenshot shows the Data Explorer interface from a web browser. A large blue digital gauge displays the value '2324,00'. Below it, a 'Single Stat' card is visible. On the left, there's a sidebar with various icons and a search bar. The main area contains a 'Query 1' section with a 'FROM' dropdown set to 'PLC' and a 'Filter' section where 'plc' is selected. To the right, there are buttons for 'View Raw Data', 'CSV', 'Past 1h', 'SCRIPT EDITOR', and 'SUBMIT'. Further right are 'WINDOW PERIOD' and 'AGGREGATE FUNCTION' settings.

Let's do this from Node-RED



The screenshot shows a Node-RED flow. It starts with a 'timestamp' node, followed by a 'Set payload and headers' node, an 'http request' node, and finally a 'debug 40' node. An inset window titled 'Edit function node' shows the properties of the 'Set payload and headers' node. Under the 'Properties' tab, the 'Name' is 'Set payload and headers'. The 'On Message' tab contains the following JavaScript code:

```
msg.payload = "plc_data,host=host1 IoT_data=1234";  
msg.headers = {};  
msg.headers['Authorization'] = 'Token MwQvFkNF8uI_Yx8327ohwgDG2qHBhO9ZbAqbpFPcFRX6amE9SooSyxAiA9zofuxj8c_C26cf-zGmMeLyGYKgHA==';  
msg.headers['Content-Type'] = 'text/plain; charset=utf-8';  
msg.headers['Accept'] = 'application/json';  
return msg;
```

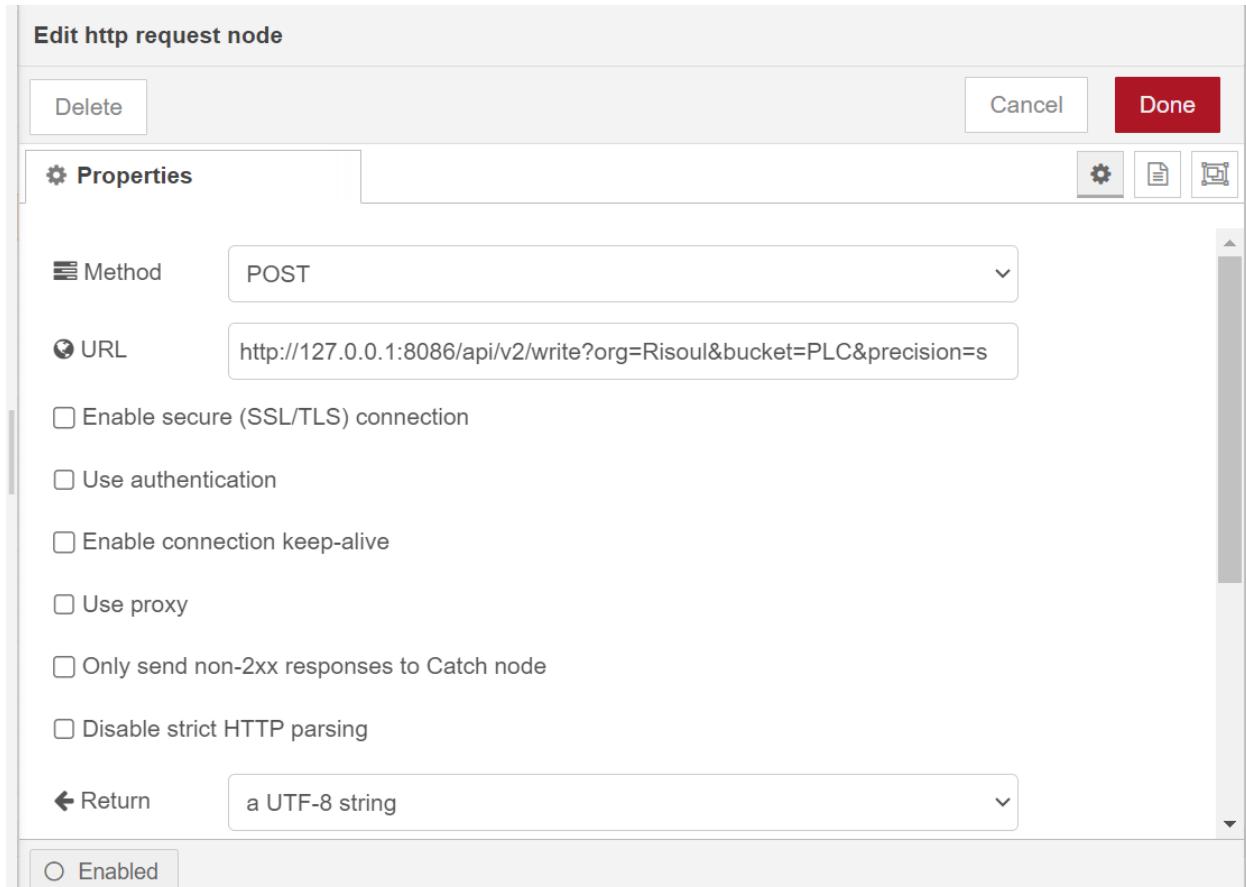
Below this, the 'On Start' tab contains the code:

```
msg.payload = "plc_data,host=host1 IoT_data=1234";  
msg.headers = {};  
msg.headers['Authorization'] = 'Token  
MwQvFkNF8uI_Yx8327ohwgDG2qHBhO9ZbAqbpFPcFRX6amE9SooSyxAiA9zofuxj8c_C26cf-  
zGmMeLyGYKgHA==' ;
```

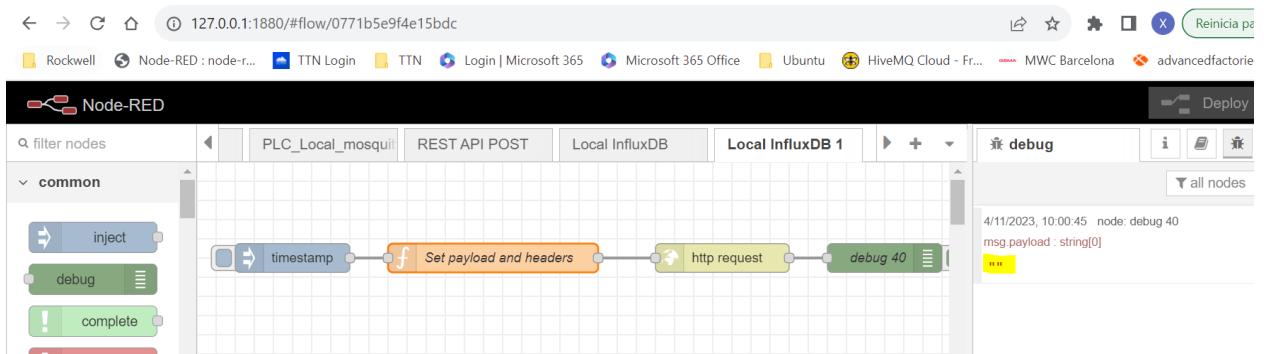
```

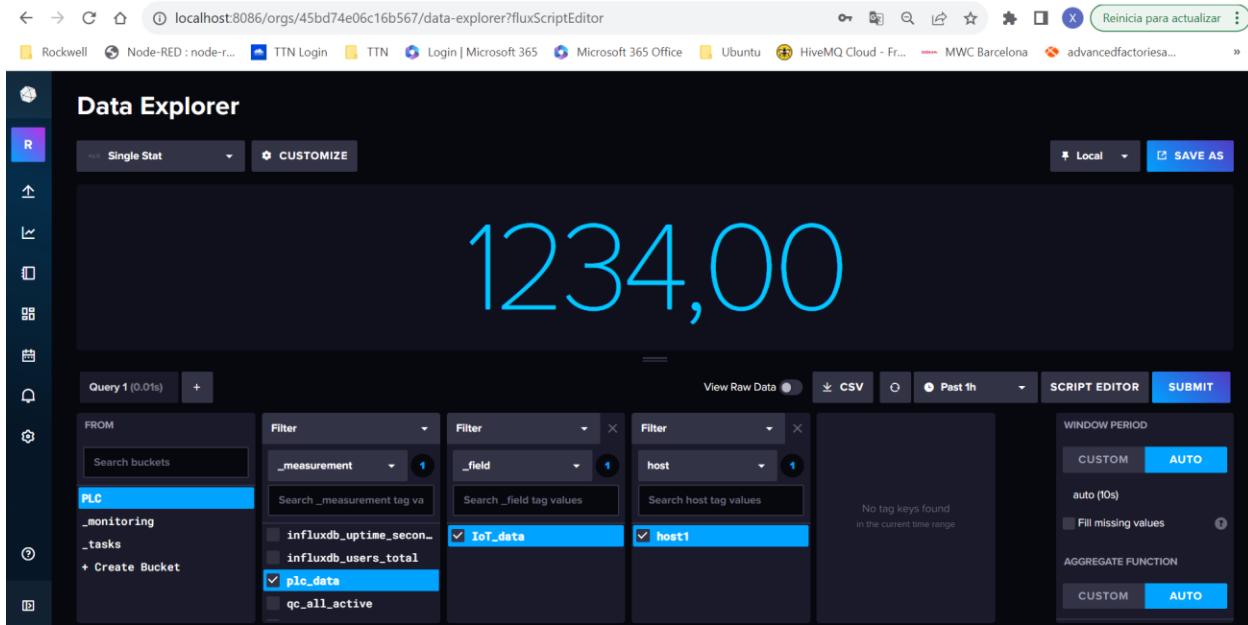
msg.headers['Content-Type'] = 'text/plain; charset=utf-8';
msg.headers['Accept'] = 'application/json';
return msg;

```



### Success





Now, let's try to do this from Factory Talk Optix

### 16.5. POST request to local InfluxDB

First let's try the right http request

API token example

```
c4K4HCsdkgH_9Vv1zBDLJ2ay8QR9ORkYTIPTclHo7PI8--
```

```
BPcuhEBQkb0sl5QCbAEVozZgzuA9vWk2iHnYxijg==
```

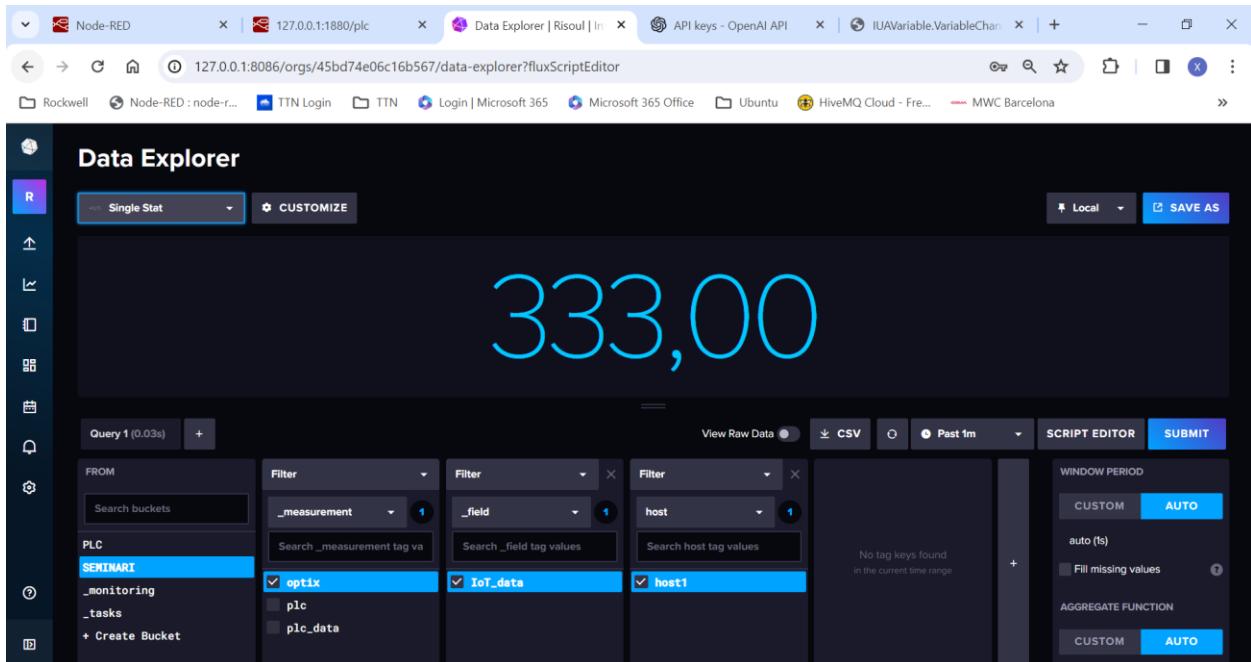
```
C:\Users\xavier.florensa>curl -POST "http://127.0.0.1:8086/api/v2/write?org=Risoul&bucket=SEMINARI&precision=s" --header "Authorization: Token 3fIg3FPeLRzm3nALax4VZdBLW4wGqUUbFhrHsSzBG9wSiiDyMVuuwj_9hKwb0v2xEi_r7K_VhndhDEdwQTiR9g==" --data-raw "optix,host=host1 IoT_data=333"
```

```
C:\Users\xavier.florensa>
```

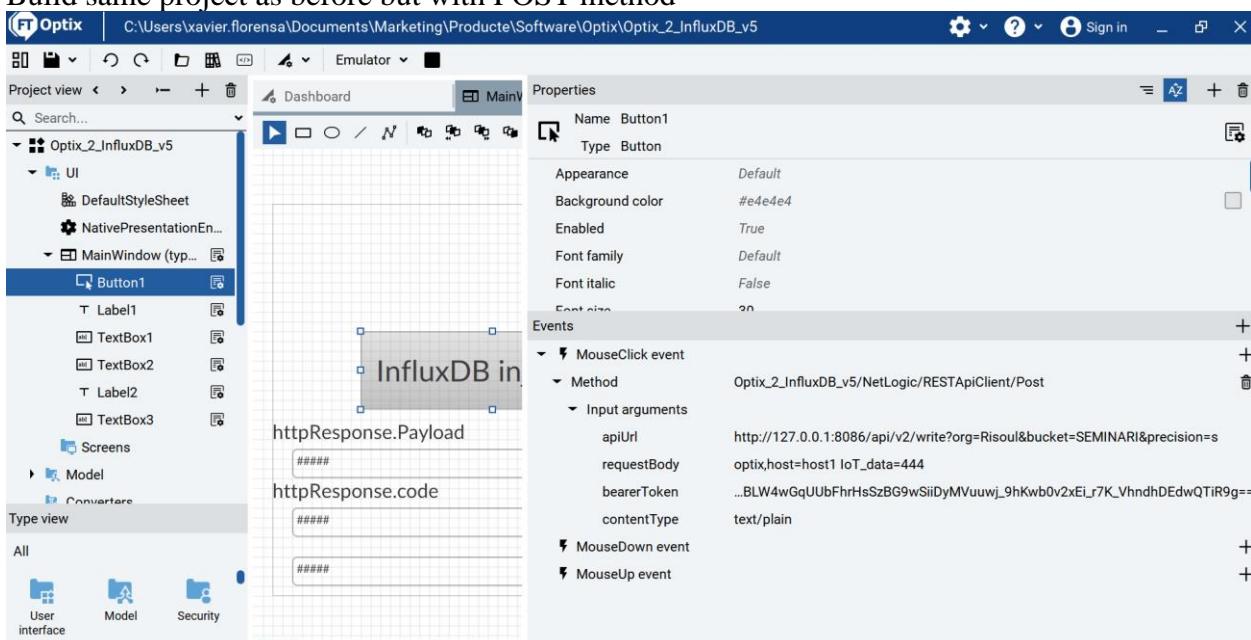
```
curl -POST "http://127.0.0.1:8086/api/v2/write?org=Risoul&bucket=SEMINARI&precision=s" -  
-header "Authorization: Token
```

```
3fIg3FPeLRzm3nALax4VZdBLW4wGqUUbFhrHsSzBG9wSiiDyMVuuwj_9hKwb0v2xEi_r7K_VhndhDEdwQTiR9g==" --data-raw "optix,host=host1 IoT_data=333"
```

With this result



Build same project as before but with POST method



## apiUrl

<http://127.0.0.1:8086/api/v2/write?org=Risoul&bucket=SEMINARI&precision=s>

## Request Body

plc\_data,host=host1 IoT\_data=232

## bearer Token

3fIg3FPeLRzm3nALax4VZdBLW4wGqUUbFhrHsSzBG9wSiiDyMVuuwj\_9hKwb0v2xEi\_r7K\_VhndhDEdwQTiR9g==

## Content type

text/plain

Optix\_2\_InfluxDB

# InfluxDB injection

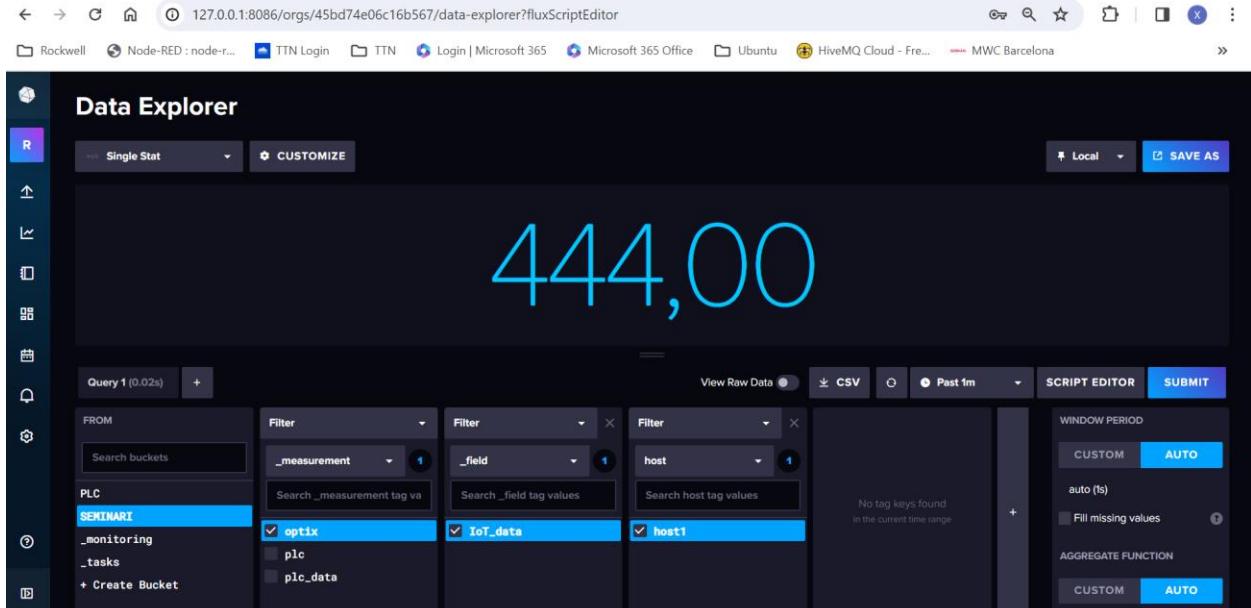
httpResponse.Payload

httpResponse.code

204

optix,host=host1 IoT\_data=444

The screenshot shows a web page with a title 'InfluxDB injection'. Below it, there are two sections: 'httpResponse.Payload' containing a large empty text area, and 'httpResponse.code' containing the value '204'. At the bottom, there is a text area with the content 'optix,host=host1 IoT\_data=444'. The browser's address bar shows the URL '127.0.0.1:8086/orgs/45bd74e06c16b567/data-explorer?fluxScriptEditor'. The page is styled with a light gray background and blue outlines for the input fields.



As you can see here  
<https://youtu.be/nqSctOoieOk>

## 16.6. POST request to remote InfluxDB

You only have to change the URL address, organization, etc, and pay attention to this little change

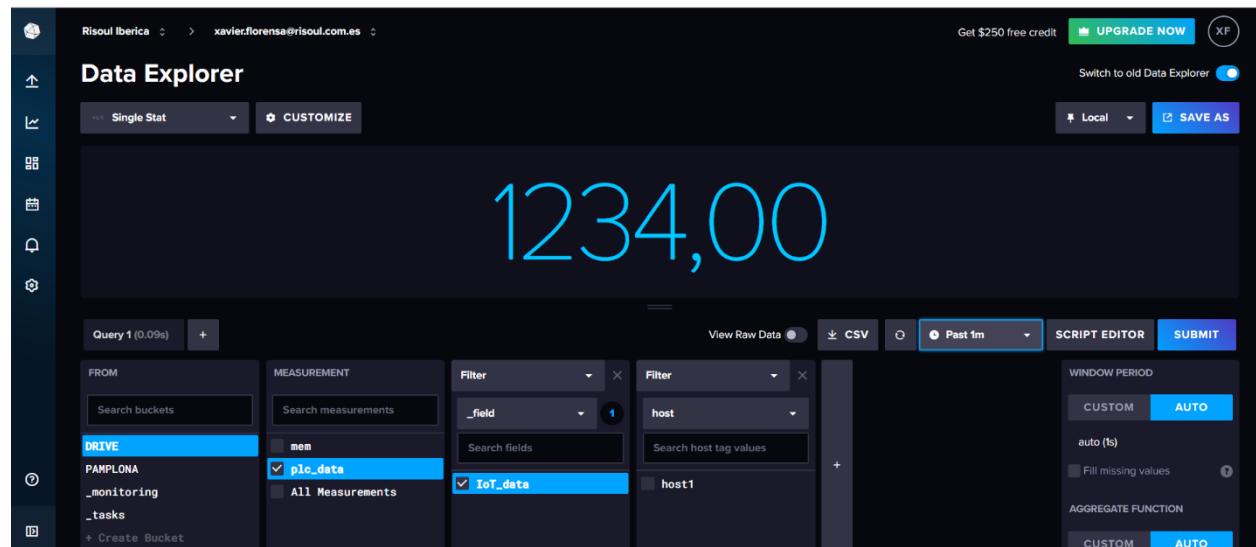
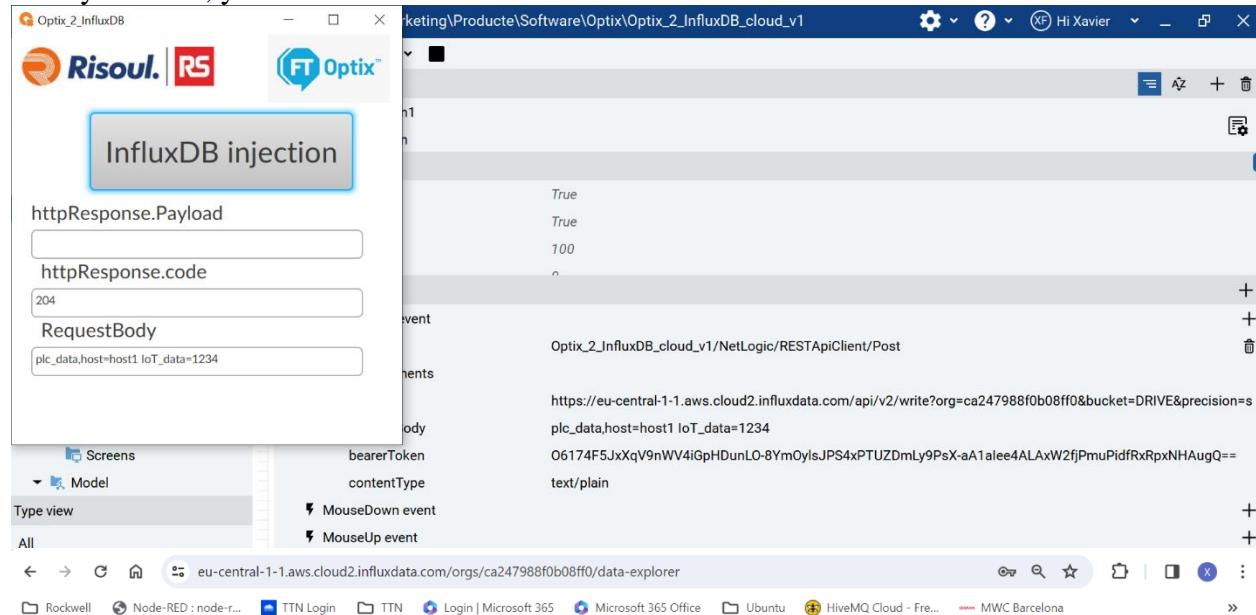
Local injection

```
91     request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", bearerToken);
```

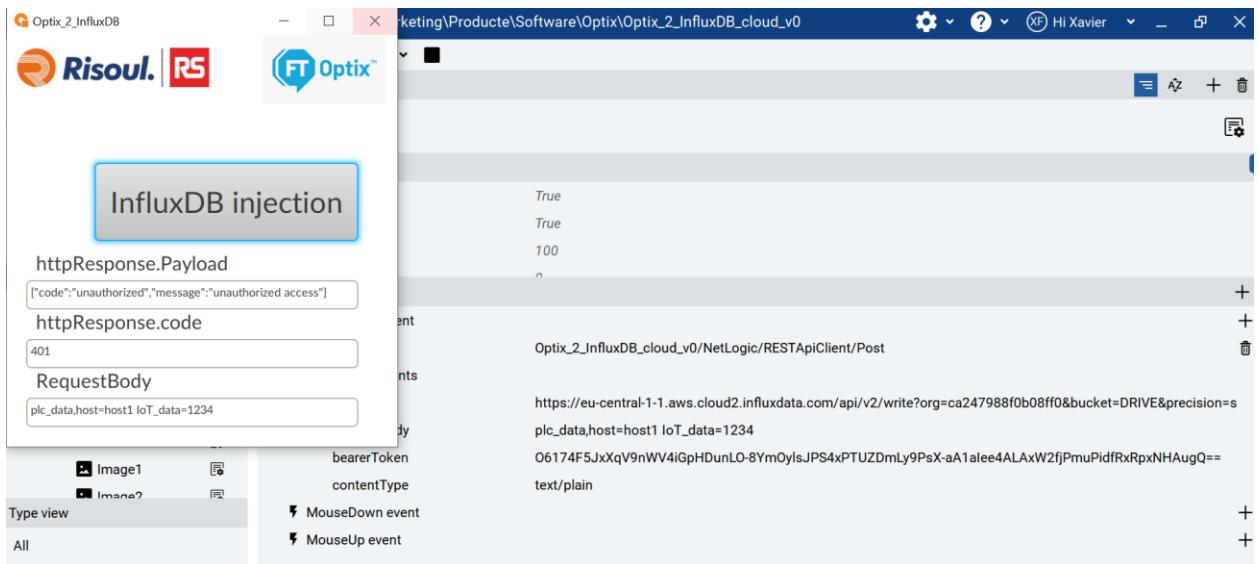
Remote injection

```
91     //request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", bearerToken);
92     request.Headers.Authorization = new AuthenticationHeaderValue("Token", bearerToken);
```

So if you do so, you will have success



But if you do not make the change from Bearer to Token, you will get this error



## 17. Using Github to store and share your repositories on the cloud

Around 100 million developers across the globe use GitHub. The majority of them are based in the US, India, and China. In 2022, there were over 413 million open-source contributions on the platform. Over 90 percent of Fortune 100 companies use GitHub. 2 d'oct. 2023



You need the Professional edition of FactoryTalk Optix and a Pro license  
The Pro license runs only on the cloud  
This is explained here



Industries Capa

ID: QA65456 | Access Levels: Everyone

## FactoryTalk Optix and GitHub

READ LATER:  Email this page  Print

To find an answer using a previous Answer ID, click here

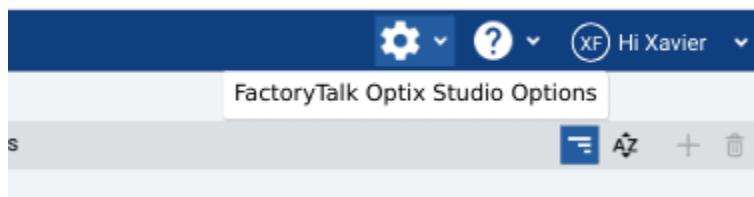
Search Knowledgebase...

Document ID QA65456

Published Date 01/27/2023

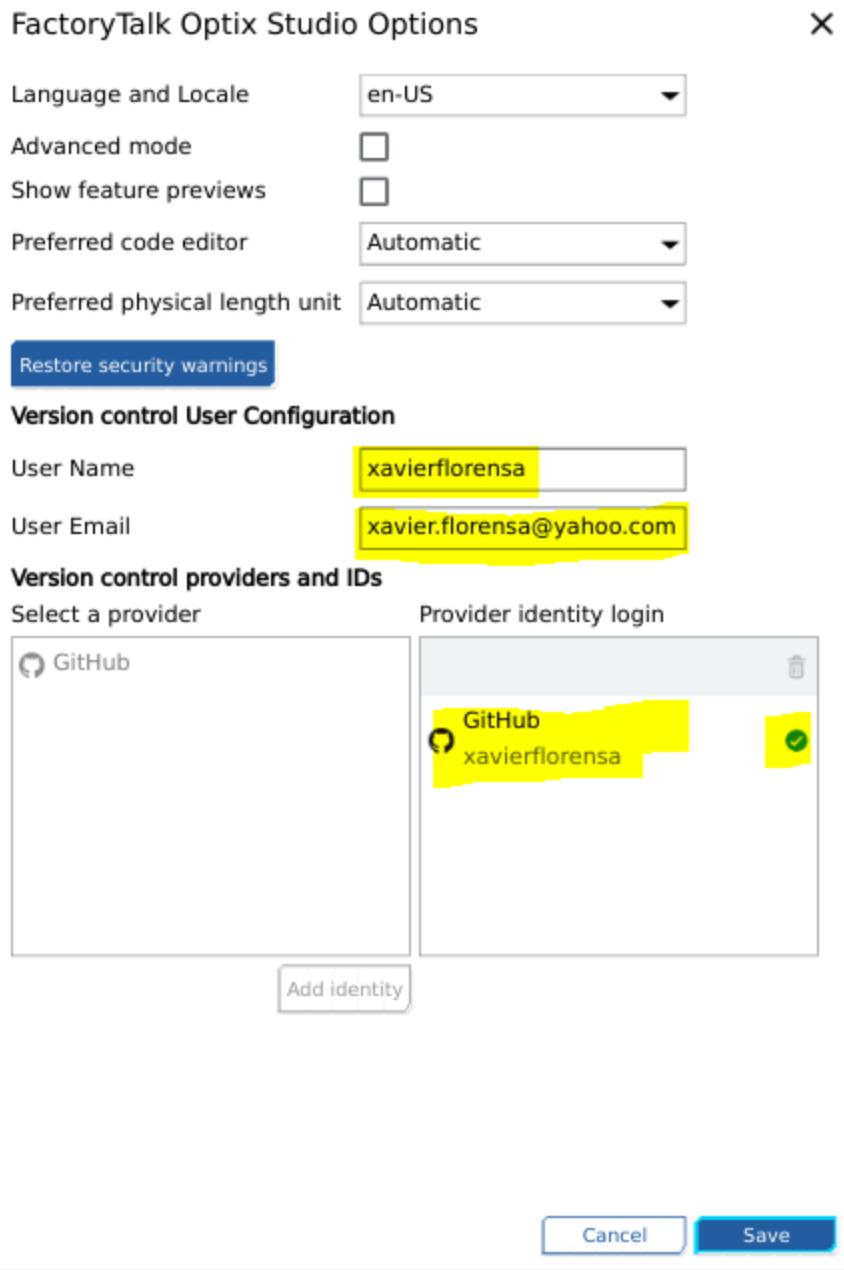
[https://rockwellautomation.custhelp.com/app/answers/answer\\_view/a\\_id/1138150/loc/en\\_US#highlight](https://rockwellautomation.custhelp.com/app/answers/answer_view/a_id/1138150/loc/en_US#highlight)

First of all modify your Optix Pro configuration on the toothwheel



Use your github user name and the associated email

You should authorize ASEM to access your github account when prompted  
Then you will see a green checkbox right to your account



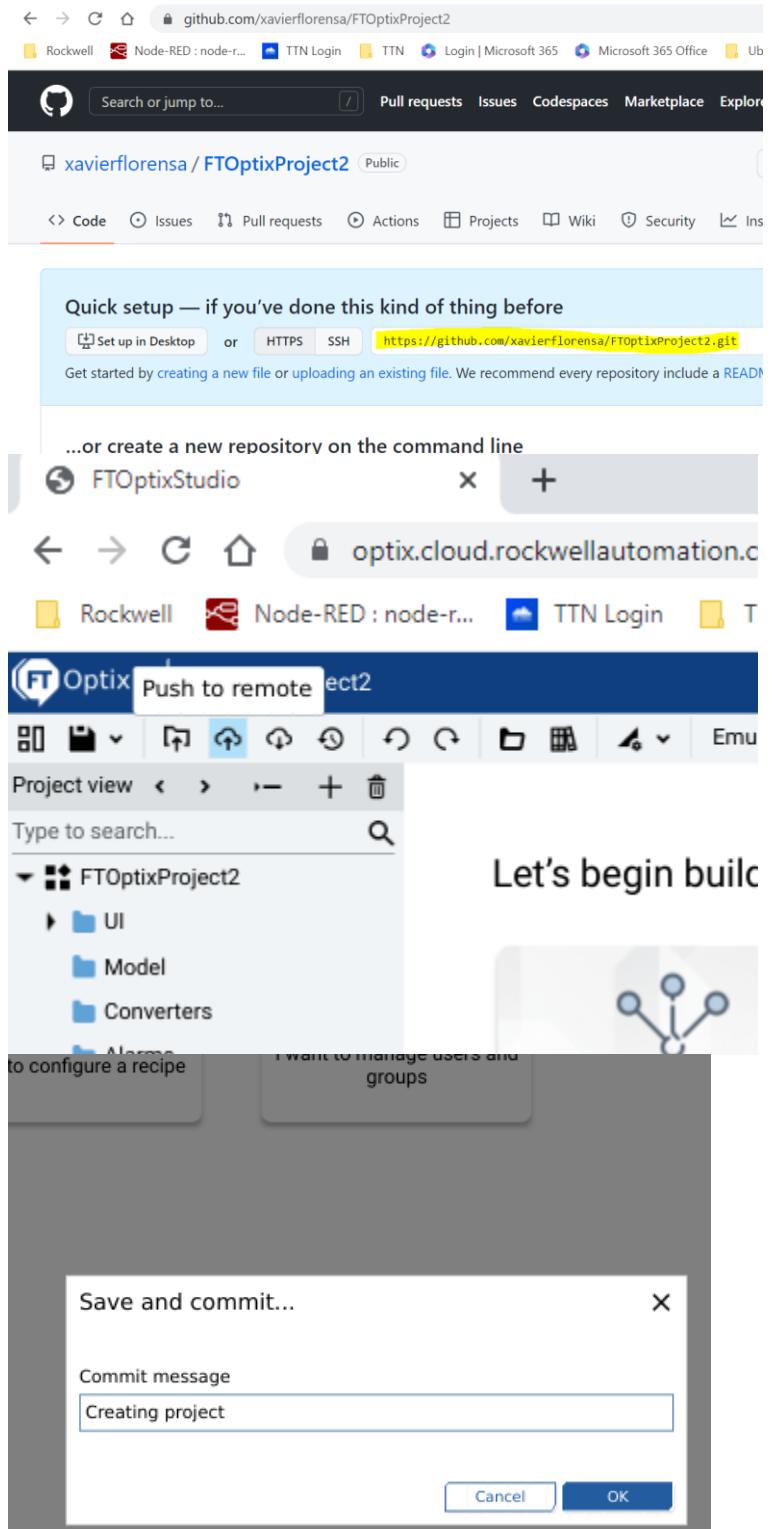
Click on save.

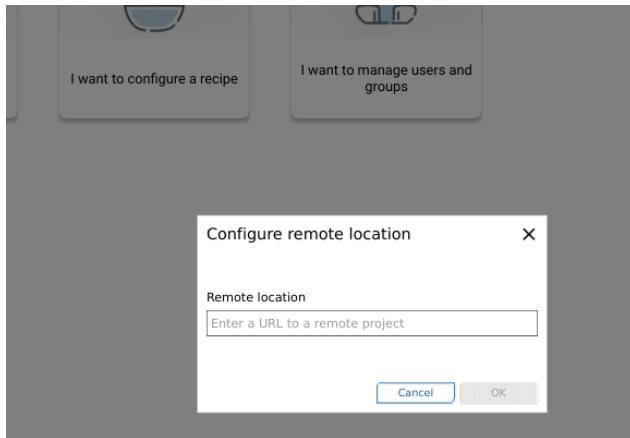
Create a project with a given name on your local instance of FT Optix  
Check version control if you want that project to be committed to Github.

Go to Github

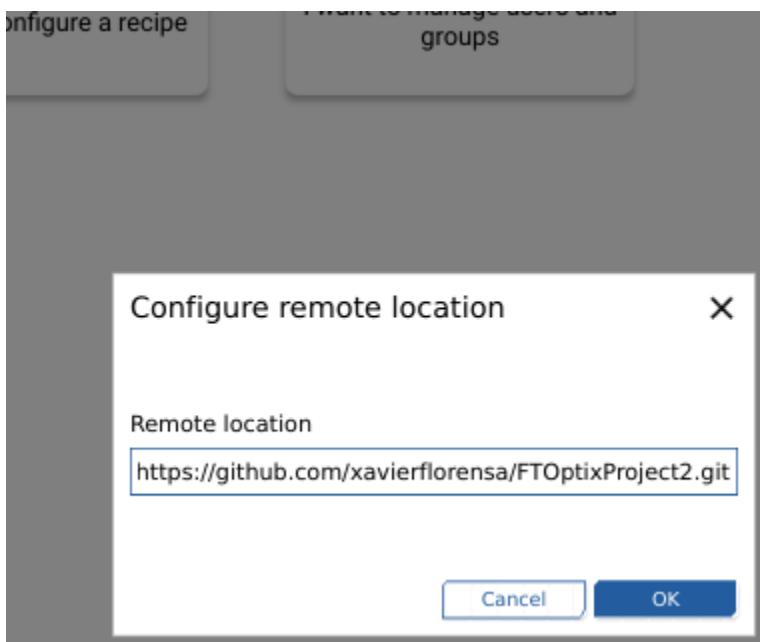
Create a new project on Github with the same name you use on Optix on Github  
That's all.

When you save and commit you will be prompted to give a comment and a destination url.  
The destination url is this one

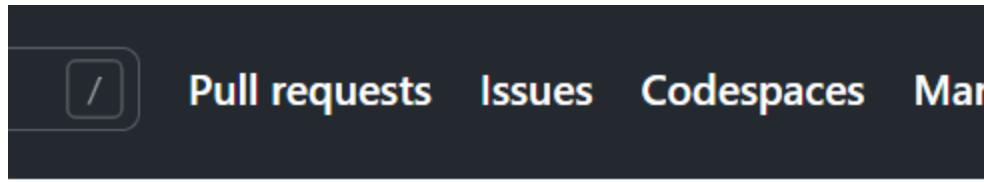




<https://github.com/xavierflorensa/FTOptixProject2.git>



Let's go to github



Find a repository...

# FTOptixProject2

Updated now

Search or jump to... / Pull requests Issues Codespaces Marketplace Explore

xavierflorensa / FTOptixProject2 Public

Pin Unwatch 1

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file ▾ Code ▾

xavierflorensa	Creating project	655fb22 2 minutes ago	1 commit
Nodes	Creating project	2 minutes ago	
.gitattributes	Creating project	2 minutes ago	
.gitignore	Creating project	2 minutes ago	
FTOptixProject2.optix	Creating project	2 minutes ago	
FTOptixProject2.optix.design	Creating project	2 minutes ago	

Screenshot of a GitHub repository page for "xavierflorensa / FTOptixProject2". The repository is public. The navigation bar includes links for Pull requests, Issues, Codespaces, Marketplace, and Explore. Below the navigation bar, there are links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. A search bar for branches is present, and the "Overview" tab is selected.

Default branch

main Updated 4 minutes ago by xavierflorensa Default

Now let's go to Optix and make some changes

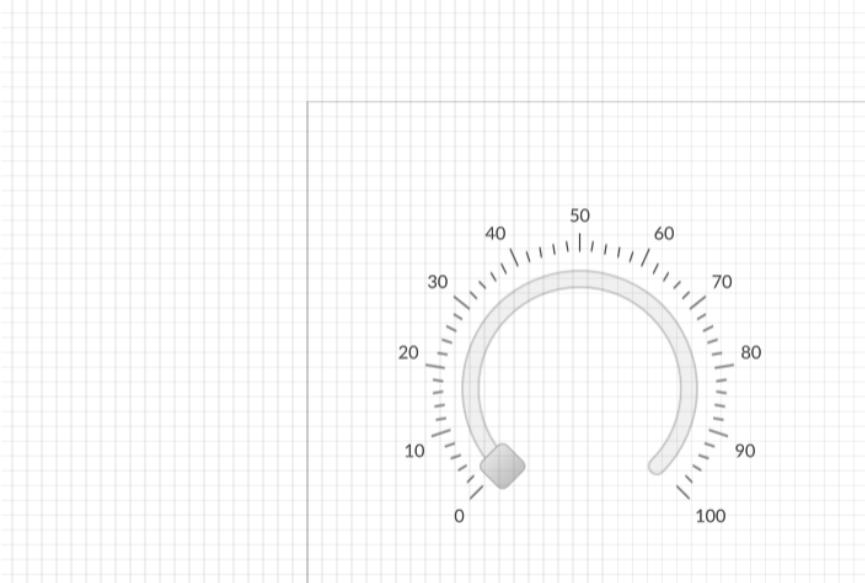
FTOptix FTOptixProject2\*

Project view UI > MainWindow

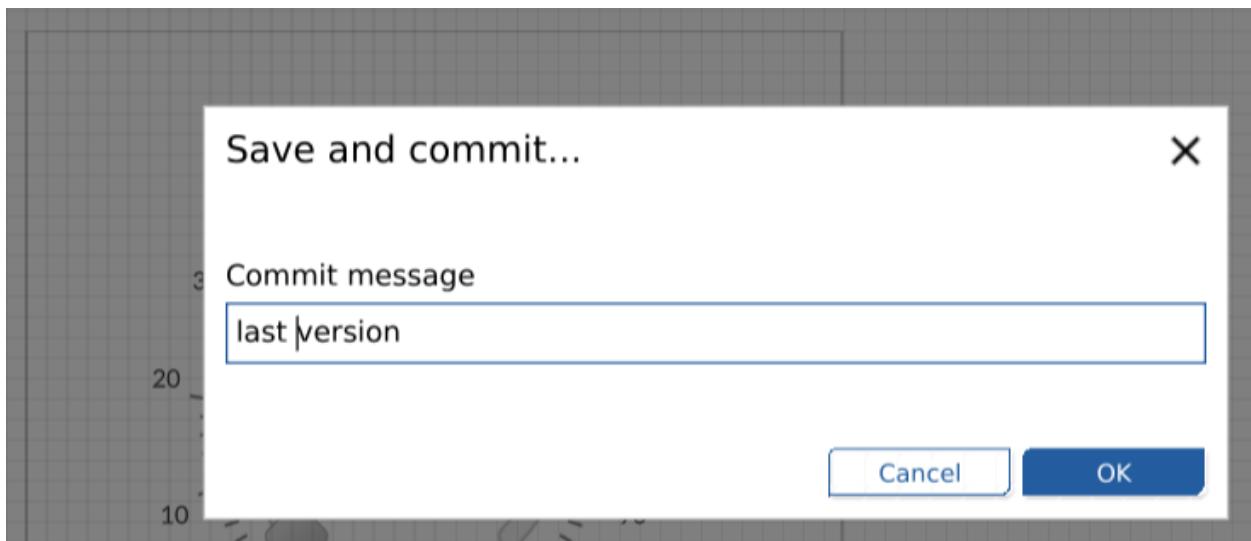
Type to search... en-US 100%

FTOptixProject2

- UI
  - DefaultStyleSheet
  - NativePresentationEng...
- MainWindow (type)
  - CircularGauge1
- Model
- Converters
- Alarms
- Recipes
- Loggers
- DataStores
- Reports
- OPC-UA
- CommDrivers



Now save and commit



But this is not already on Github until you hit Push to remote

A screenshot of the FTOptixStudio interface. The top bar shows the project name 'FTOptixStudio'. Below it is a toolbar with various icons. The main area is titled 'FTOptixProject2' and shows a 'Project view' with a search bar and a tree view of files. One file under 'UI' is selected. A status bar at the bottom shows 'UI &gt; MainWindow' and indicates there are 2 commits.

Now we have two commits. If you click on 2 commits:

A screenshot of a GitHub repository page. The URL in the address bar is `github.com/xavierflorensa/FTOptixProject2`. The repository name is `xavierflorensa / FTOptixProject2` (Public). The main navigation tabs are Pull requests, Issues, Codespaces, Marketplace, and Explore. Below the tabs, there are links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. A dropdown menu shows the current branch is `main` (1 branch, 0 tags). Buttons for Go to file, Add file, and Code are present. A list of recent commits by `xavierflorensa` is shown, all dated 4 minutes ago:

Commit	Message	Date
ff563b7	last version	4 minutes ago
Nodes	last version	4 minutes ago
.gitattributes	Creating project	12 minutes ago
.gitignore	Creating project	12 minutes ago
FTOptixProject2.optix	Creating project	12 minutes ago
FTOptixProject2.optix.design	Creating project	12 minutes ago

You will see this

A screenshot of a GitHub repository page for the same repository. The main navigation tabs are Pull requests, Issues, Codespaces, Marketplace, and Explore. Below the tabs, there are links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and Insights. A dropdown menu shows the current branch is `main`. A list of commits is shown, starting with "Commits on Mar 25, 2023":

- last version**  
xavierflorensa committed 6 minutes ago
- Creating project**  
xavierflorensa committed 13 minutes ago

At the bottom right, there are buttons for Newer and Older.

That's all, so you have a record on all the changes and who made it  
You can view commit details

The screenshot shows a GitHub repository page for 'xavierflorensa / FTOptixProject2'. The main navigation bar includes 'Pull requests', 'Issues', 'Codespaces', 'Marketplace', and 'Explore'. Below the bar, there are buttons for 'Pin', 'Unwatch 1', 'Fork 0', and 'Star 0'. The repository name 'xavierflorensa / FTOptixProject2' is displayed, along with a 'Code' tab which is currently selected. Other tabs include 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. A dropdown menu for the 'main' branch is open, showing two commits:

- last version** by xavierflorensa committed 7 minutes ago
- Creating project** by xavierflorensa committed 14 minutes ago

Each commit has a 'View commit details' button, a copy icon, and a diff icon. At the bottom of the commit list are 'Newer' and 'Older' buttons.

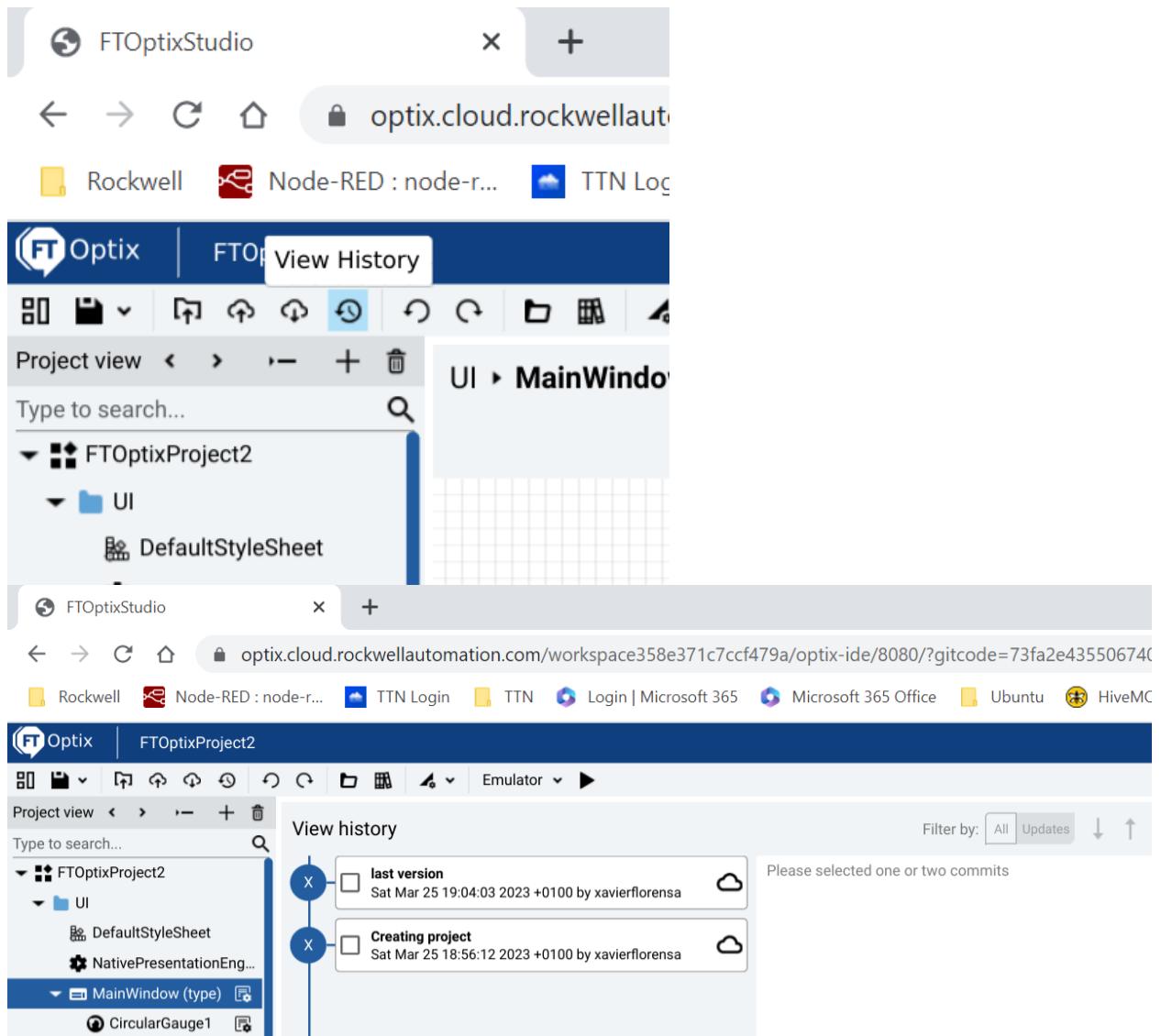
And you can see additional comments

The screenshot shows a GitHub commit page for the repository `xavierflorensa / FTOptixProject2`. The commit is titled "last version" and was made to the `main` branch by `xavierflorensa` 8 minutes ago. The commit message is "Showing 1 changed file with 37 additions and 0 deletions." The diff view shows 37 additions to the file `Nodes/UI/UI.yaml`. The additions are highlighted in green. The code changes are as follows:

```
diff --git a/Nodes/UI/UI.yaml b/Nodes/UI/UI.yaml
@@ -82,3 +82,40 @@ Children:
E2  E2   DataType: Size
E3  E3   ModellingRule: Optional
E4  E4   Value: 400.0
E5  +   - Name: CircularGauge
E6  +   Type: CircularGauge
E7  +   Children:
E8  +     - Name: Value
E9  +       Type: BaseDataVariableType
E10 +      DataType: Float
E11 +      Value: 0.0
E12 +     - Name: MinValue
E13 +       Type: BaseDataVariableType
E14 +      DataType: Float
E15 +      Value: 0.0
E16 +     - Name: MaxValue
E17 +       Type: BaseDataVariableType
E18 +      DataType: Float
E19 +      Value: 100.0
E20 +     - Name: WarningZones
E21 +       Type: BaseObjectType
E22 +     - Name: Width
E23 +       Type: BaseVariableType
E24 +      DataType: Size
E25 +      ModellingRule: Optional
E26 +      Value: 210.0
E27 +     - Name: Height
E28 +       Type: BaseVariableType
E29 +      DataType: Size
E30 +      ModellingRule: Optional
E31 +      Value: 210.0
E32 +     - Name: TopMargin
E33 +       Type: BaseVariableType
E34 +      DataType: Size
E35 +      ModellingRule: Optional
E36 +      Value: 90.0
E37 +     - Name: LeftMargin
E38 +       Type: BaseVariableType
E39 +      DataType: Size
E40 +      ModellingRule: Optional
E41 +      Value: 90.0
```

Below the diff view, there is a comment section with a "Write" button and a yellow-highlighted "Leave a comment" input field. The placeholder text in the input field is "Leave a comment".

You can also view history on Optix



## 18. Pushing your local Optix projects to Github

### 18.1. Using Optix

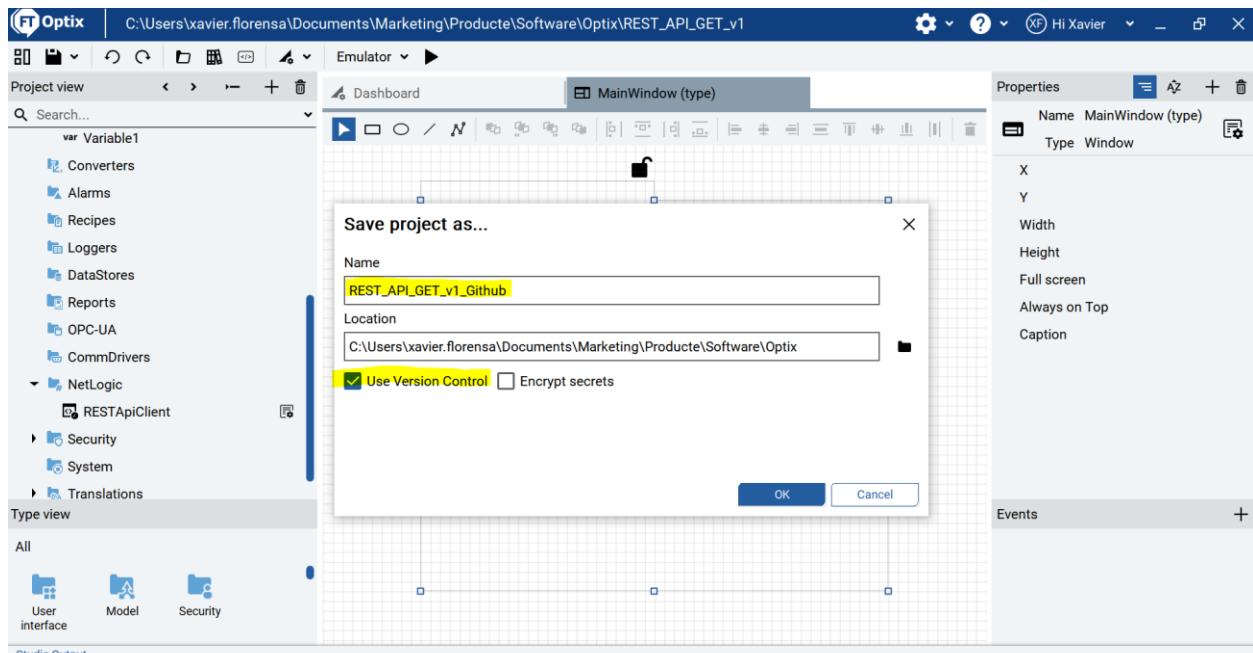
Probably you have started with Optix locally and created some projects that you want to store later on on Github without having to create them again.

Save your project with a different name and copy the project name

For instance

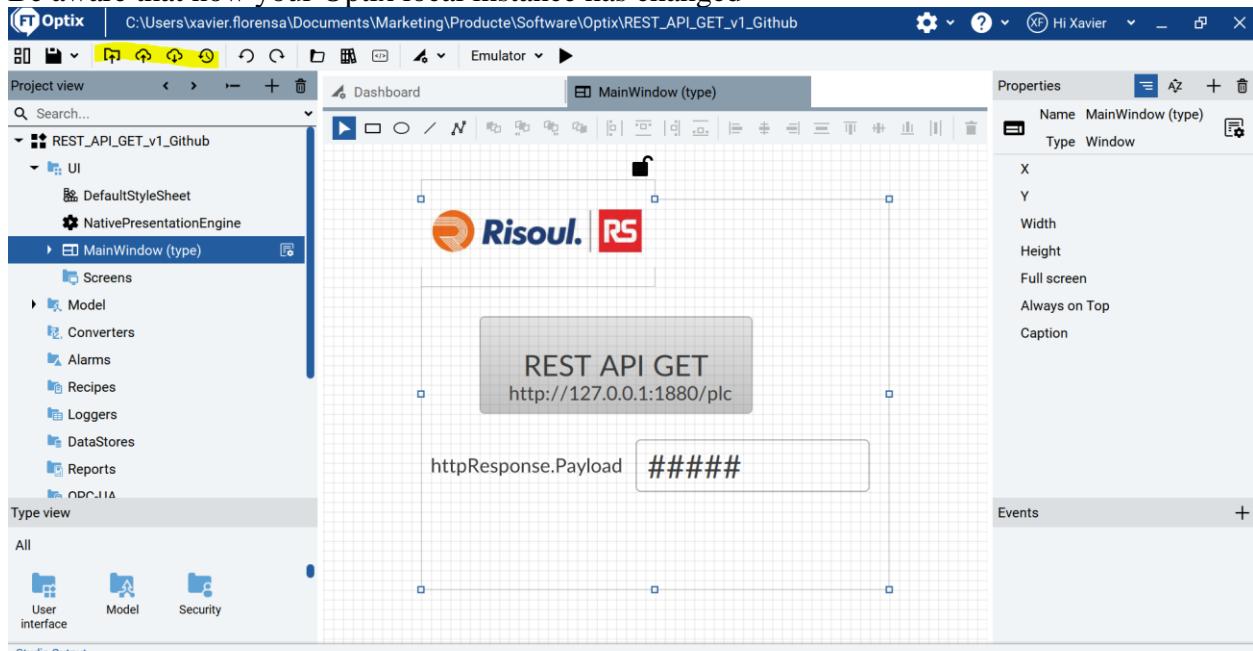
REST\_API\_GET\_v1\_Github

Mark “Use version control”



Click OK

Be aware that now your Optix local instance has changed



Also the files and directory has changed

Before

Before

Name	Date modified	Type	Size
Nodes	12/9/2023 6:32 PM	File folder	
ProjectFiles	12/9/2023 2:55 PM	File folder	
IDEVersion	12/4/2023 7:55 PM	Text Document	1 KB
Logo	11/16/2023 8:55 PM	JPG File	15 KB
REST_API_GET_v1	12/9/2023 6:32 PM	FactoryTalk Optix Stu...	2 KB
REST_API_GET_v1.optix.design	12/9/2023 6:32 PM	DESIGN File	1 KB

After

Name	Date modified	Type	Size
Nodes	12/17/2023 6:43 PM	File folder	
ProjectFiles	12/17/2023 6:43 PM	File folder	
.gitattributes	7/28/2023 3:39 PM	Text Document	13 KB
.gitignore	7/28/2023 3:39 PM	Text Document	1 KB
IDEVersion	12/4/2023 7:55 PM	Text Document	1 KB
REST_API_GET_v1_Github	12/17/2023 6:43 PM	FactoryTalk Optix Stu...	2 KB
REST_API_GET_v1_Github.optix.design	12/17/2023 6:43 PM	DESIGN File	1 KB

Now let's create a repository with same name on Github

So

REST\_API\_GET\_v1\_Github

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \*      Repository name \*

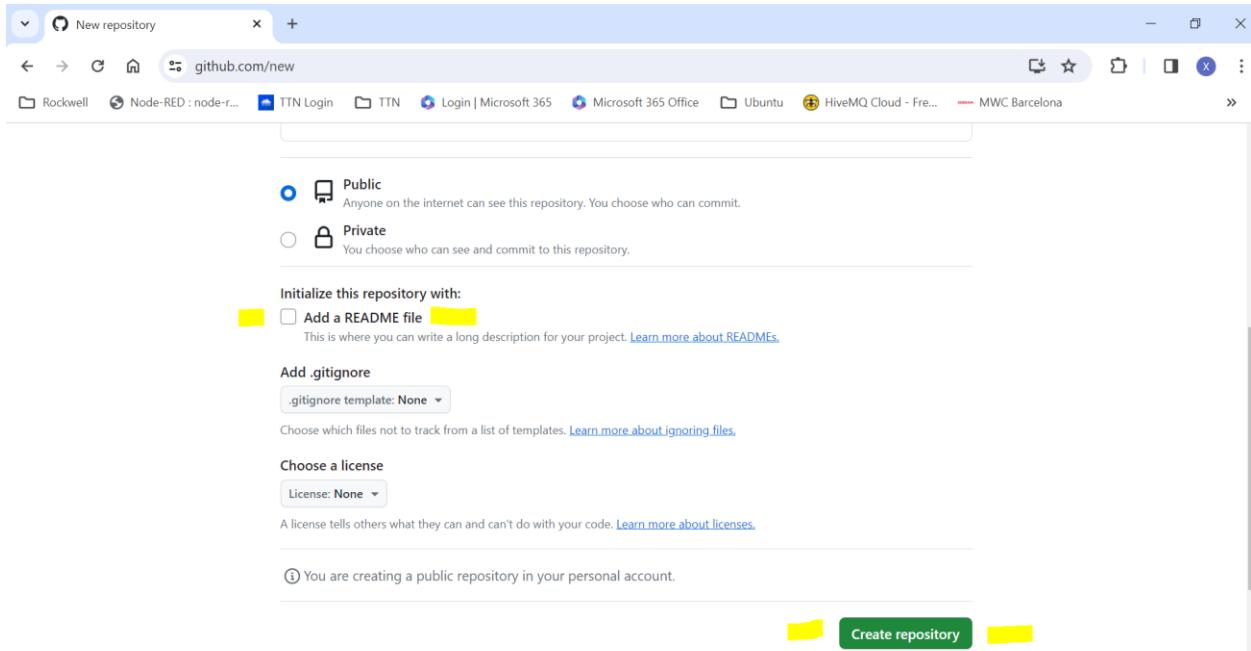
xavierflorena / REST\_API\_GET\_v1\_Github  
REST\_API\_GET\_v1\_Github is available.

Great repository names are short and memorable. Need inspiration? How about [cautious-spoon](#) ?

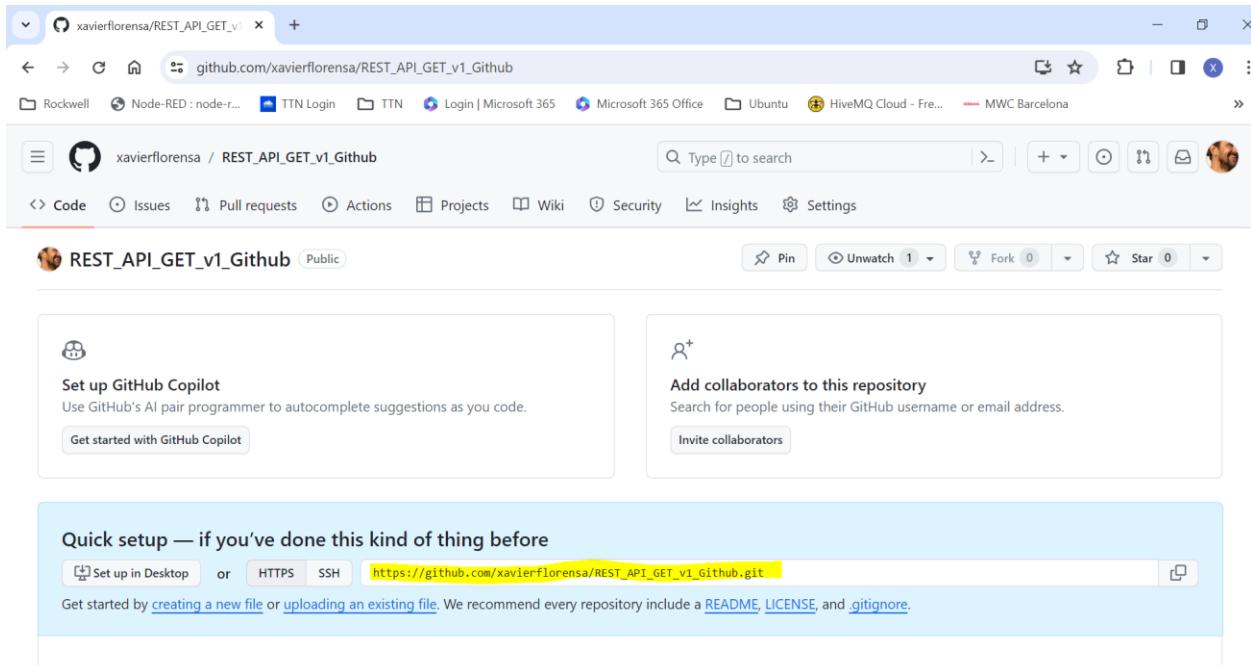
Description (optional)

Public      Anyone on the internet can see this repository. You choose who can commit.  
 Private      You choose who can see and commit to this repository.

Do not check create a readme file



Click on create

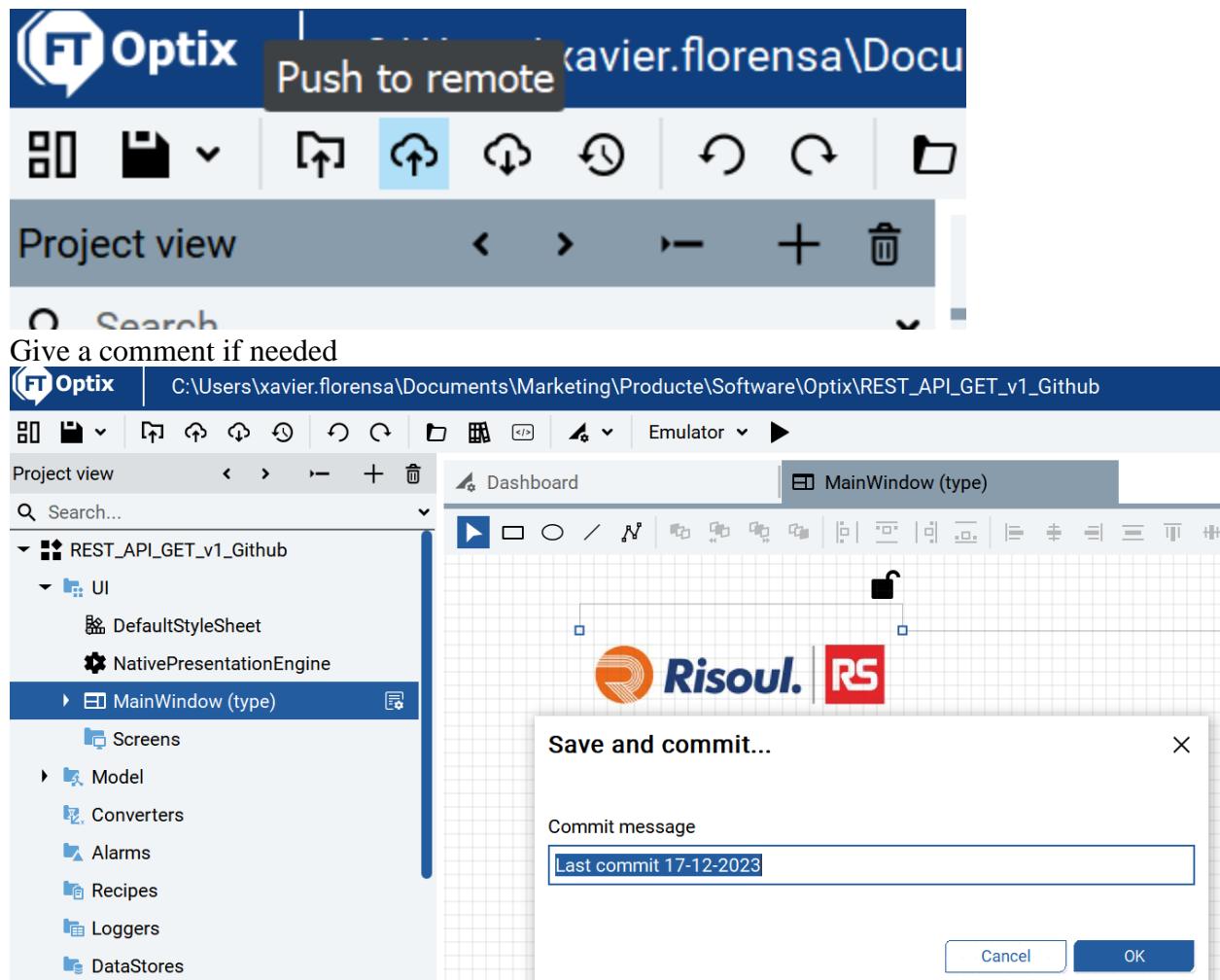


Copy this address

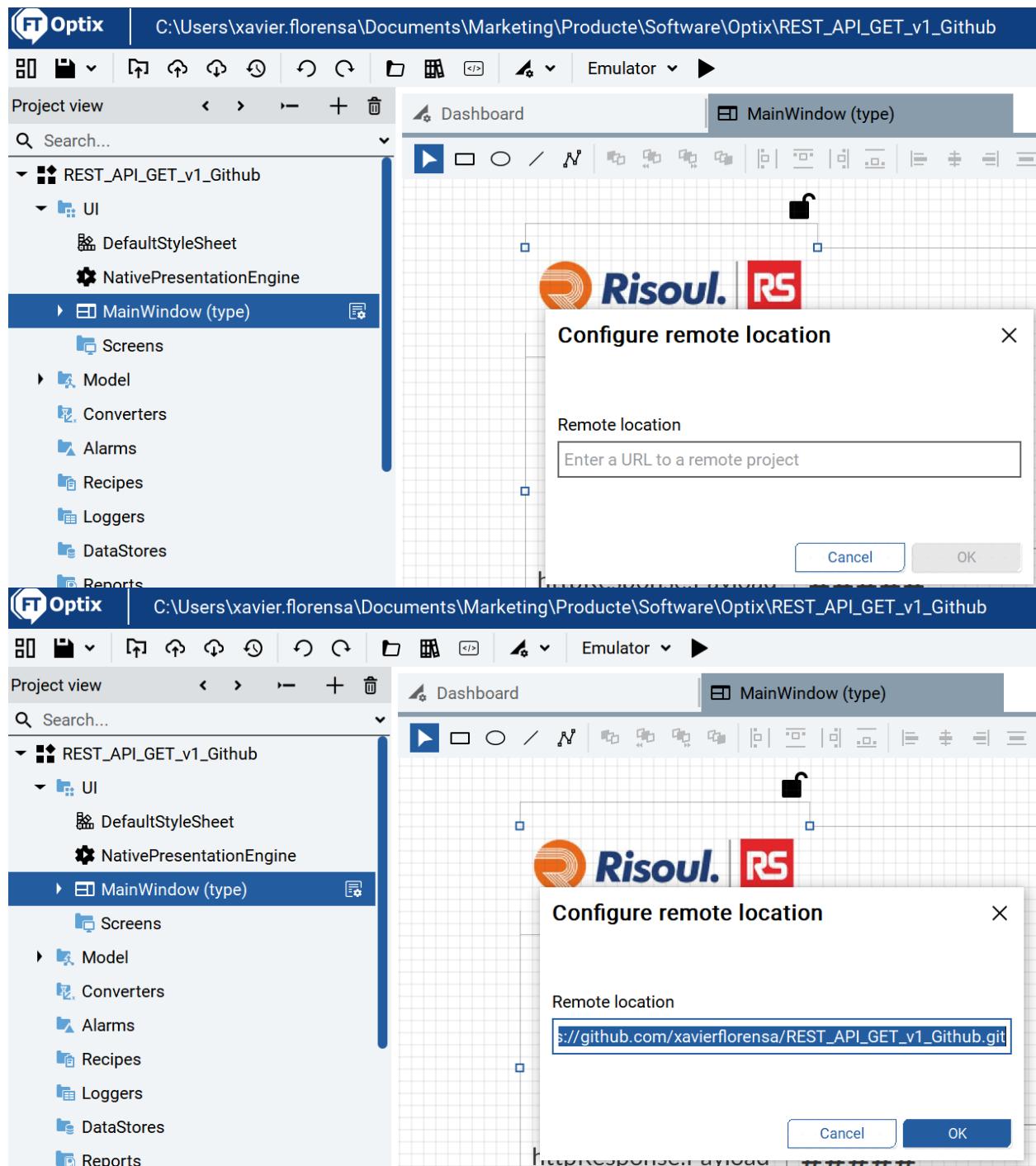
[https://github.com/xavierflorensa/REST\\_API\\_GET\\_v1\\_Github.git](https://github.com/xavierflorensa/REST_API_GET_v1_Github.git)

Your new repository is empty.

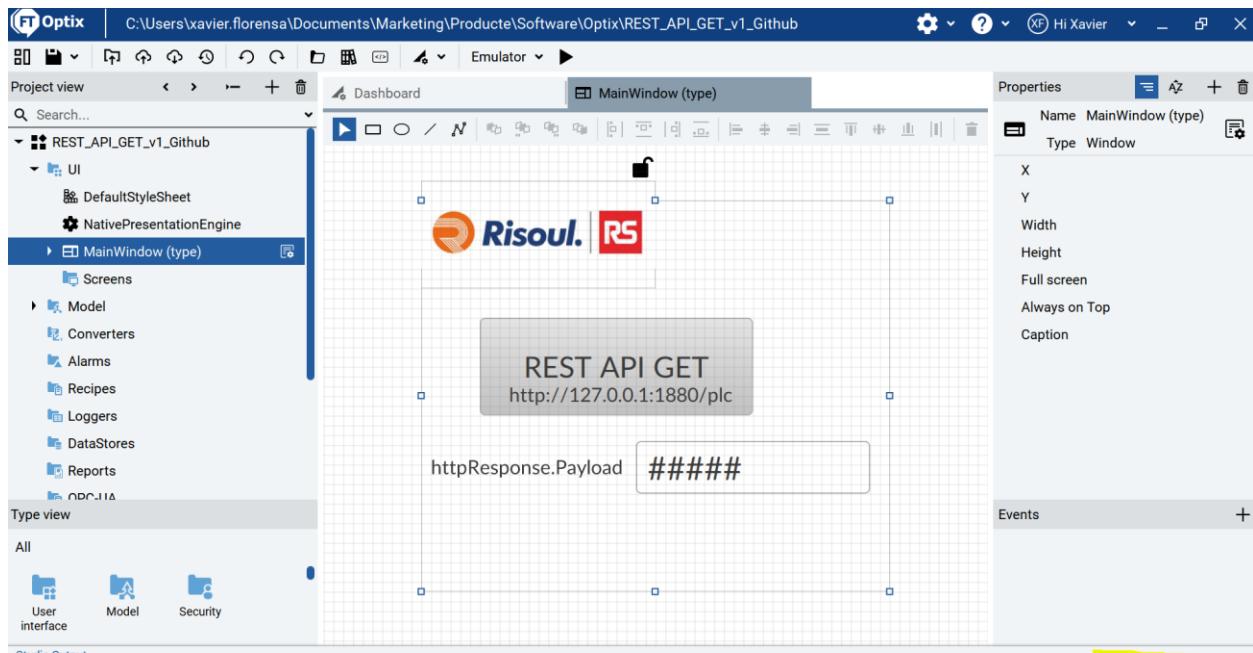
Now go to Optix Studio local  
And hit on “Push to remote”



Give here the copied url address



You get no errors and project is synchronized



Now look at your project in Github

**Code**

**REST\_API\_GET\_v1\_Github** Public

main · 1 Branch · 0 Tags

Go to file Add file Code About

xavierflorensa Last commit 17-12-2023 58d1e73 · 3 minutes ago 1 Commits

- Nodes Last commit 17-12-2023 3 minutes ago
- ProjectFiles Last commit 17-12-2023 3 minutes ago
- .gitattributes Last commit 17-12-2023 3 minutes ago
- .gitignore Last commit 17-12-2023 3 minutes ago
- REST\_API\_GET\_v1\_Github.optix Last commit 17-12-2023 3 minutes ago
- REST\_API\_GET\_v1\_Github.optix.design Last commit 17-12-2023 3 minutes ago

No description, website, or topics provided.

Activity 0 stars 1 watching 0 forks

Releases No releases published Create a new release

Packages

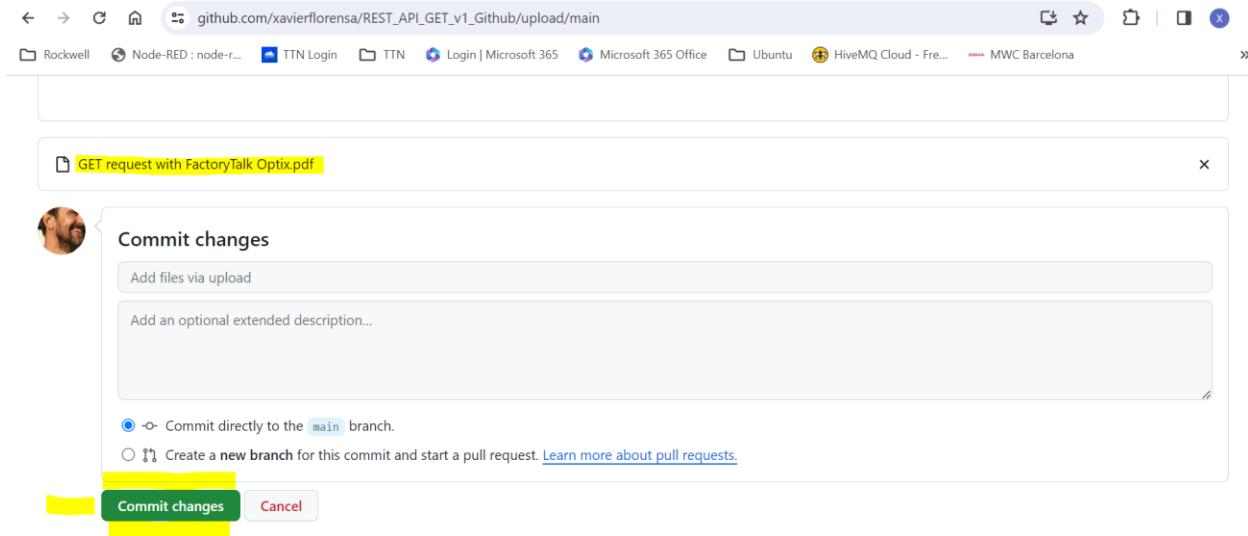
Your project is now public (or private) and you can share now this project with others.  
Next step will be how to include a document with info since readme file is only text.

Let's try to do so from Github.

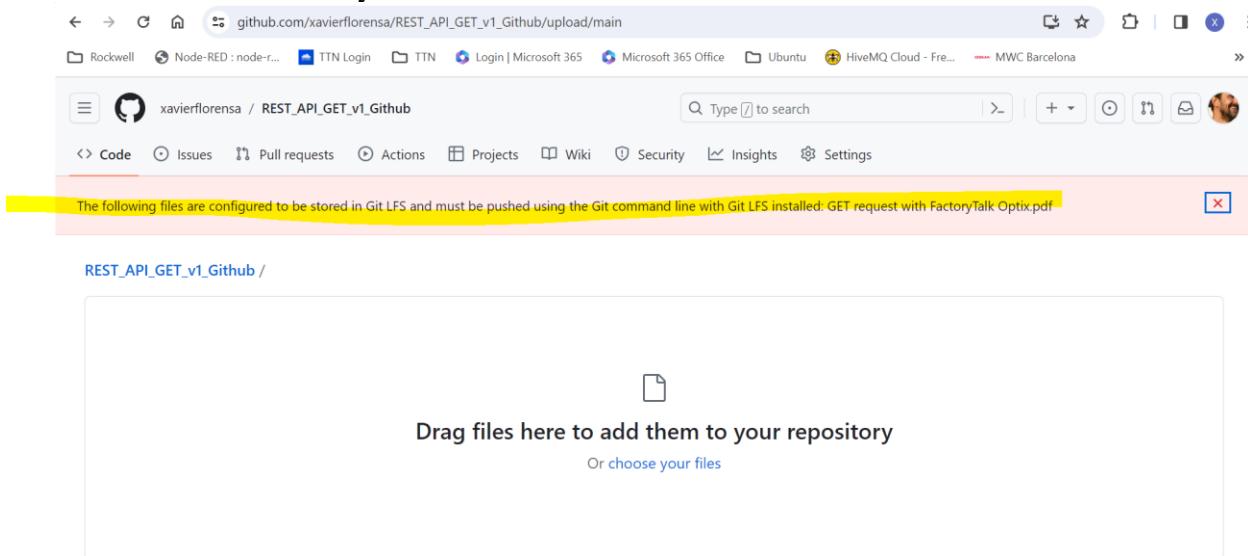
Add file / upload file

The screenshot shows a GitHub repository page for 'REST\_API\_GET\_v1\_Github'. The repository is public and has 1 branch and 0 tags. The last commit was made by 'xavierflorensa' on 17-12-2023. The repository contains several files: 'Nodes', 'ProjectFiles', '.gitattributes', '.gitignore', 'REST\_API\_GET\_v1\_Github.optix', and 'REST\_API\_GET\_v1\_Github.optix.design'. All files were last committed on 17-12-2023 at 20 hours ago. A context menu is open over the repository name, showing options like 'Create new file' and 'Upload files'.

The screenshot shows the GitHub 'upload/main' page for the 'REST\_API\_GET\_v1\_Github' repository. It displays a large text input field with the URL 'https://github.com/xavierflorensa/REST\_API\_GET\_v1\_Github/upload/main'. Below the input field is a placeholder text 'Drag files here to add them to your repository' and a 'Or choose your files' button. At the bottom, there is a 'Commit changes' section where the user can add files via upload.

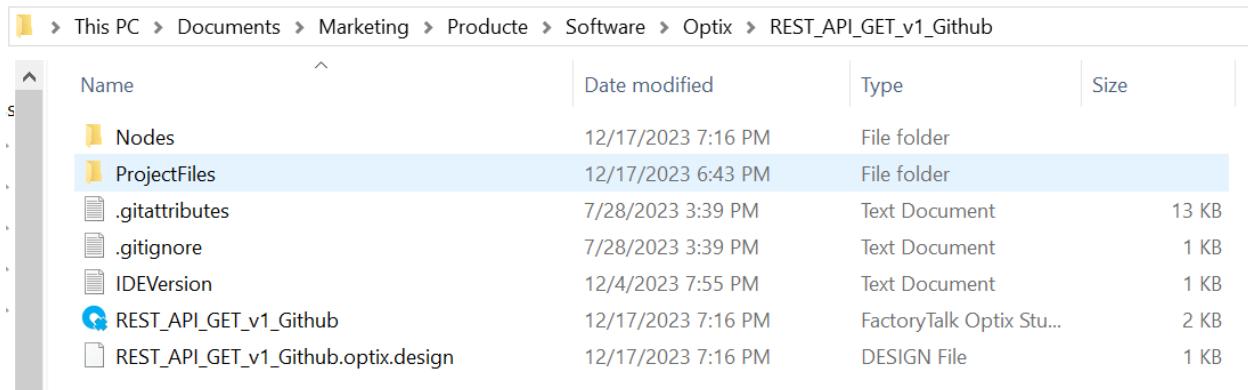


You can not do that this way



So you have to do it with GIT.

Unless you go back to your local Optix project directory and add a file there.  
Like this

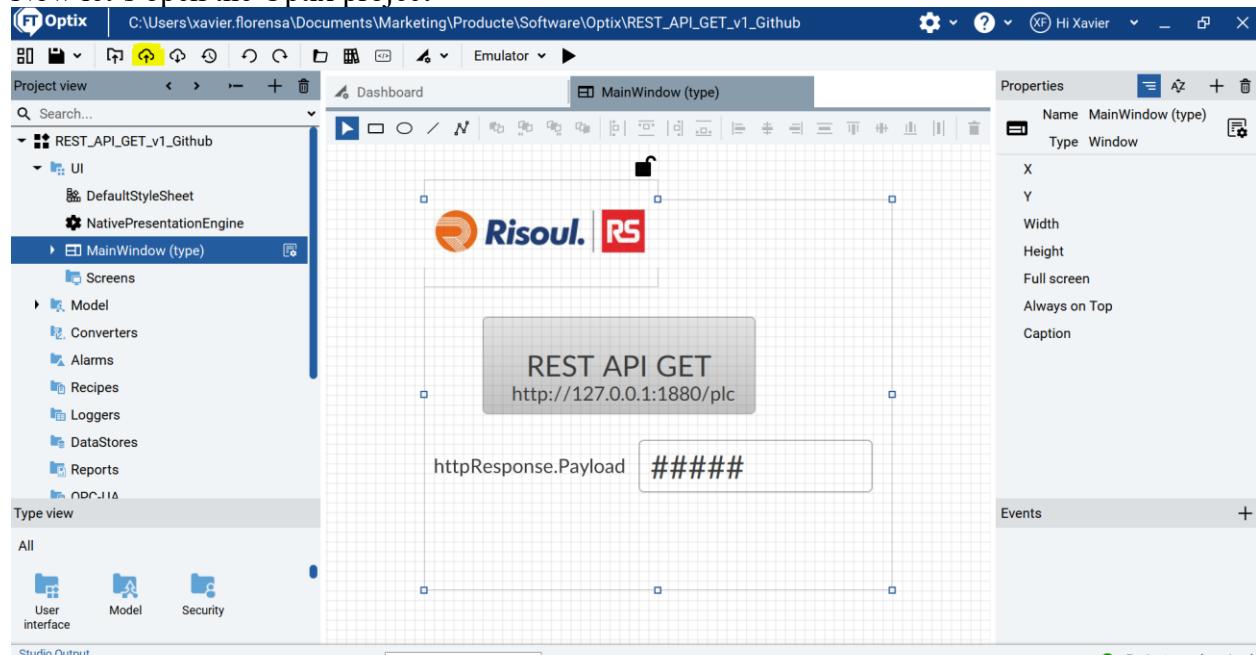


I add the documentation file I have forgot

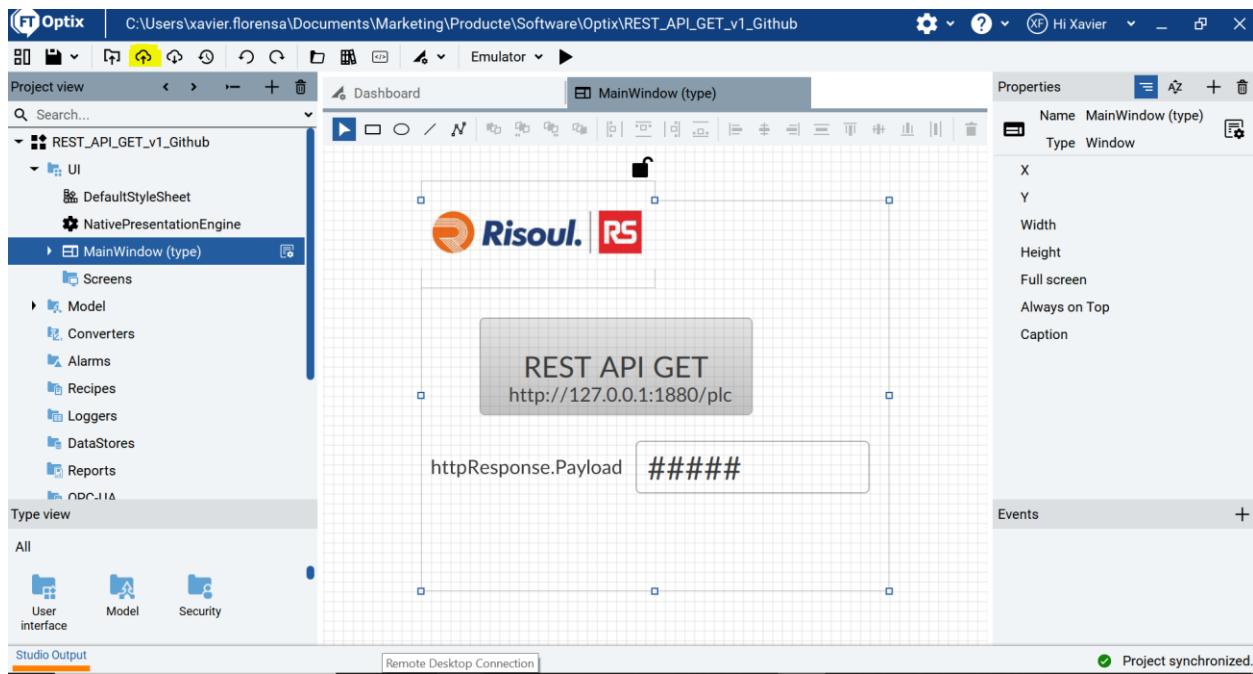
File Explorer View:

Name	Date modified	Type	Size
Nodes	12/17/2023 7:16 PM	File folder	
ProjectFiles	12/17/2023 6:43 PM	File folder	
.gitattributes	7/28/2023 3:39 PM	Text Document	13 KB
.gitignore	7/28/2023 3:39 PM	Text Document	1 KB
<b>GET request with FactoryTalk Optix</b>	12/18/2023 3:25 PM	Chrome HTML Docu...	1,957 KB
IDEVersion	12/4/2023 7:55 PM	Text Document	1 KB
REST_API_GET_v1_Github	12/17/2023 7:16 PM	FactoryTalk Optix Stu...	2 KB
REST_API_GET_v1_Github.optix.design	12/17/2023 7:16 PM	DESIGN File	1 KB

Now let's open the Optix project



Click on push



Nothing changes

Code

REST\_API\_GET\_v1\_Github Public

main 1 Branch 0 Tags

xavierflorensa Last commit 17-12-2023 58d1e73 - 20 hours ago 1 Commits

- Nodes Last commit 17-12-2023 20 hours ago
- ProjectFiles Last commit 17-12-2023 20 hours ago
- .gitattributes Last commit 17-12-2023 20 hours ago
- .gitignore Last commit 17-12-2023 20 hours ago
- REST\_API\_GET\_v1\_Github.optix Last commit 17-12-2023 20 hours ago
- REST\_API\_GET\_v1\_Github.optix.design Last commit 17-12-2023 20 hours ago

README

About

No description, website, or topics provided.

Activity

0 stars

1 watching

0 forks

Releases

No releases published

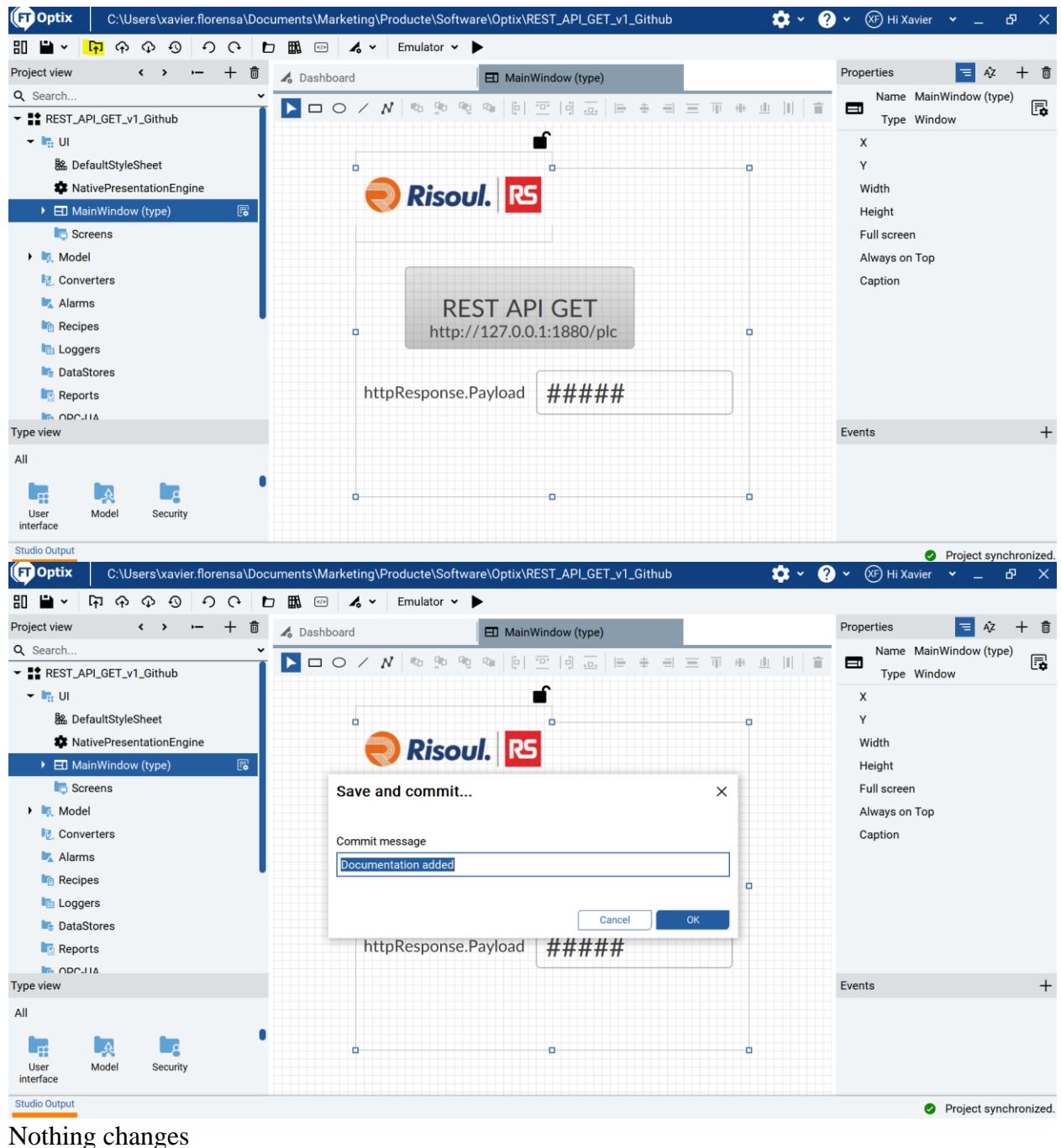
Create a new release

Packages

No packages published

Publish your first package

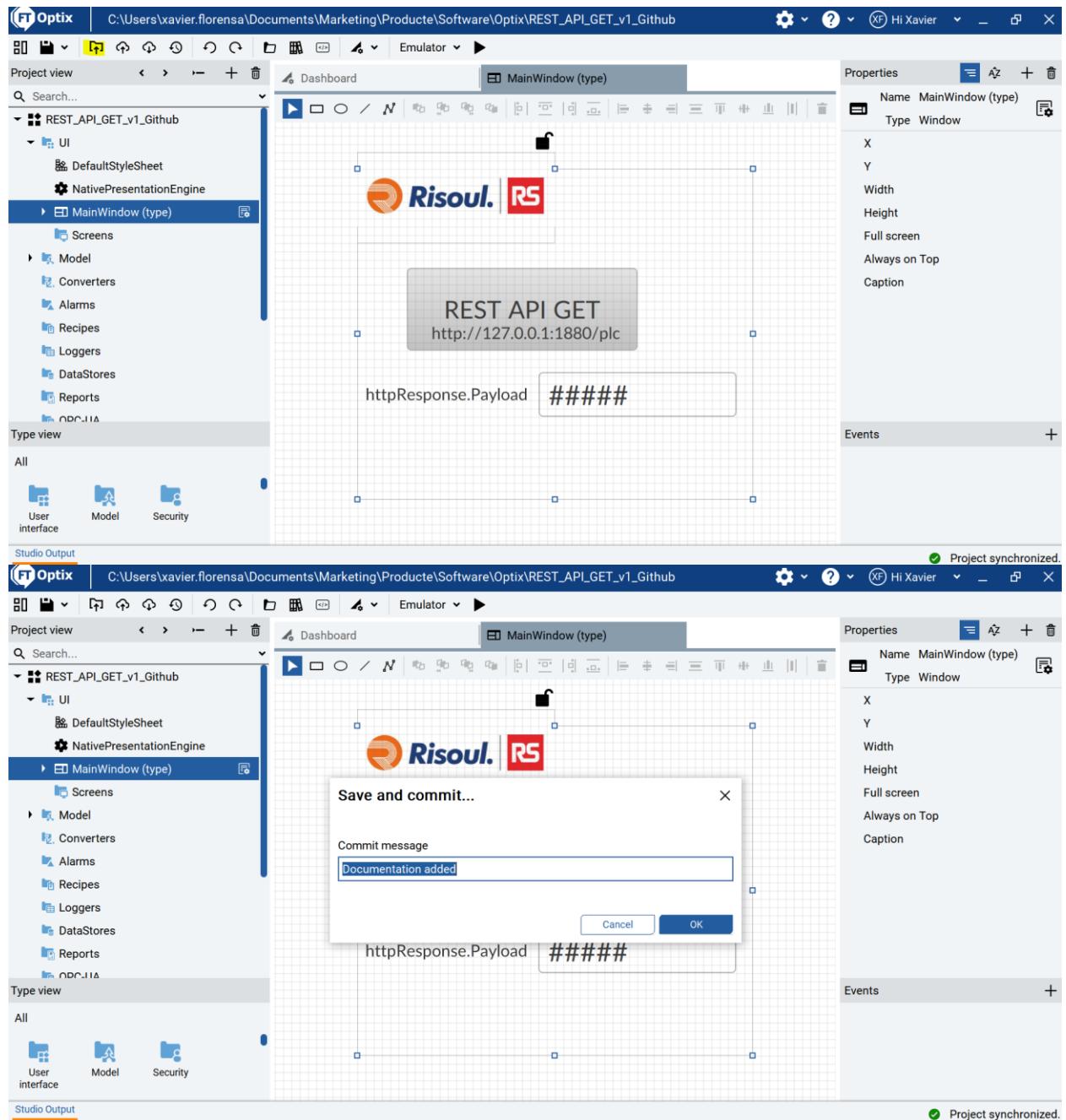
Let's try with save and commit



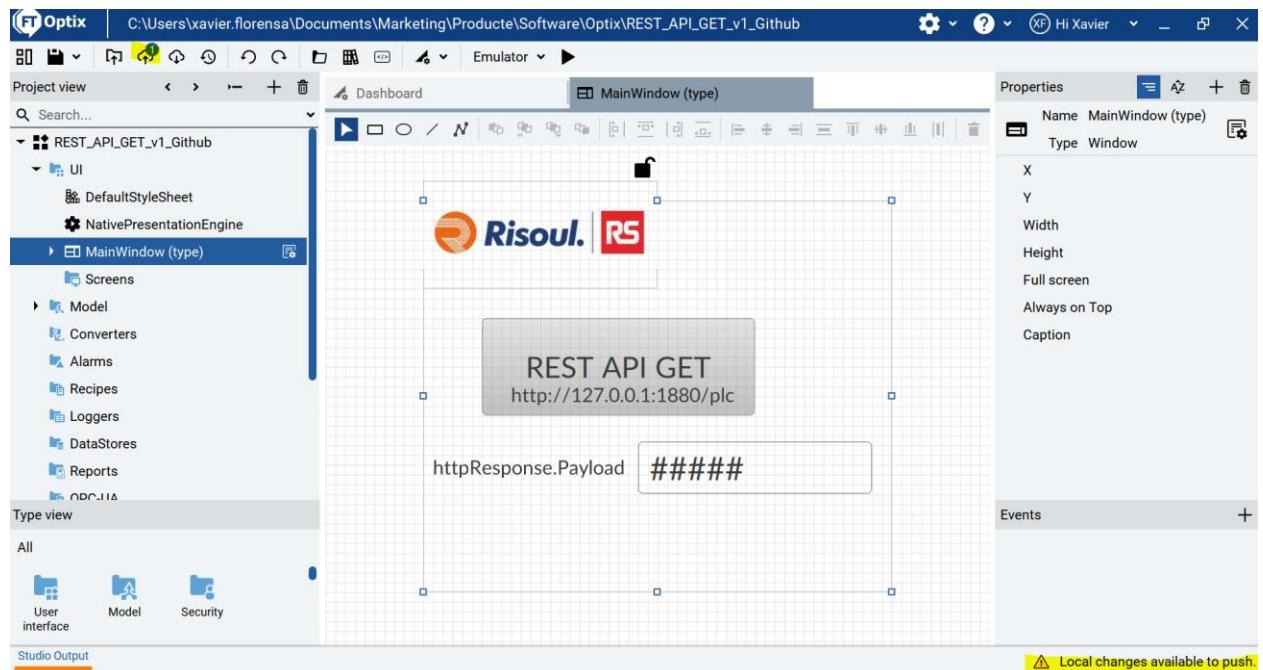
Nothing changes

Let's try to copy the file to Project files directory on our local directory  
Close and open Optix project

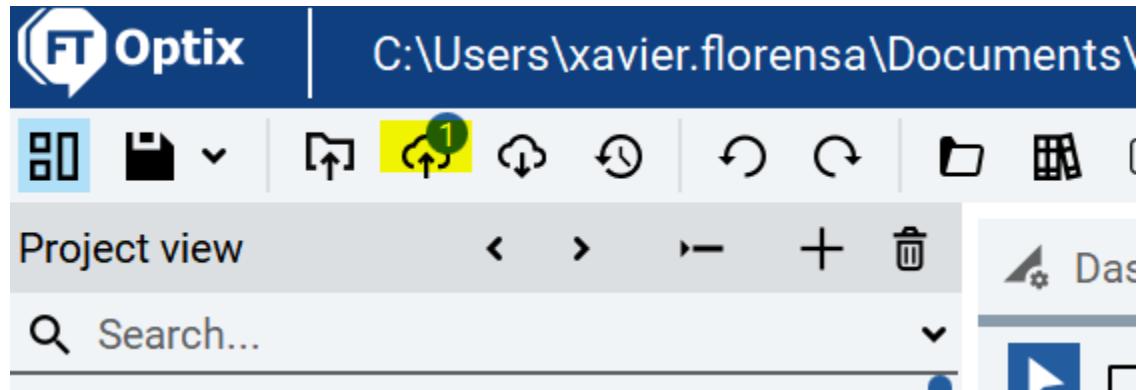
Now let's repeat  
Save and commit



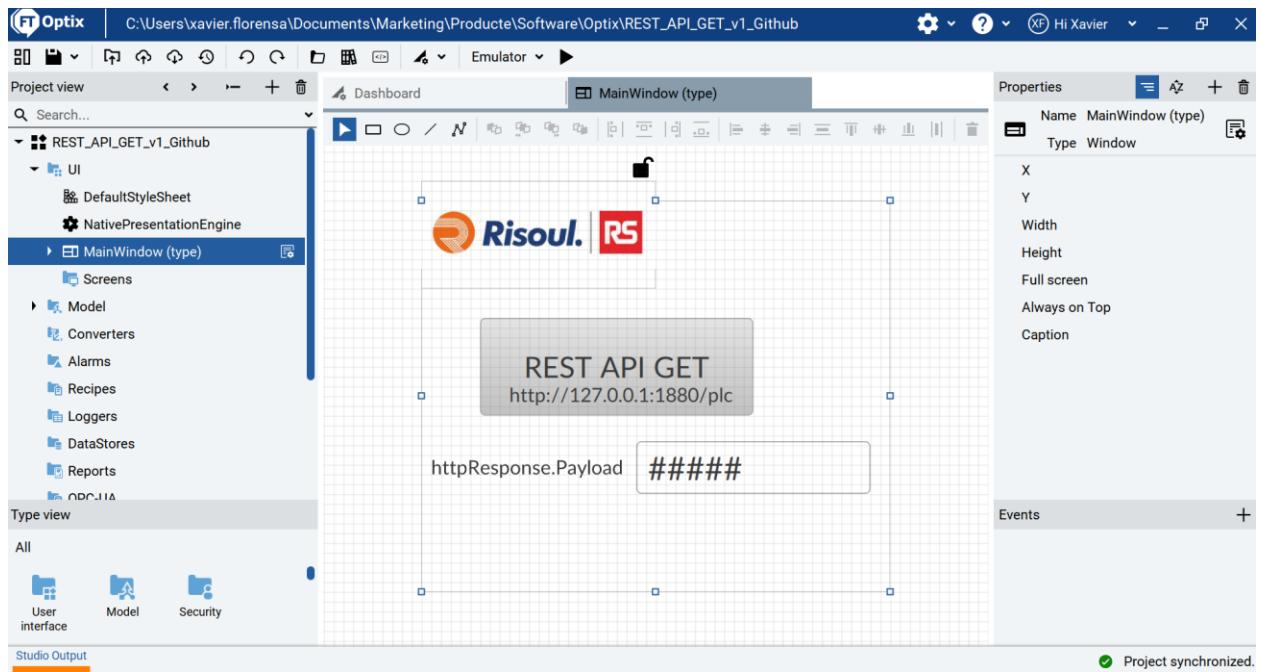
Something has changed now



If we click on Push



Now project is synchronized



And if we open our Github project  
Voilà, here it is

Name	Last commit message	Last commit date
...		yesterday
NetSolution	Last commit 17-12-2023	yesterday
GET request with FactoryTalk Optix.pdf	Documentation added	6 minutes ago
Logo.JPG	Last commit 17-12-2023	yesterday

You can add a Readme.md file after the project is created.  
Upload files

**Optix\_EasyModbusTCP\_client** Public

main 1 Branch 0 Tags Go to file Add file <> Code

xavierflorensa Add files via upload ce18621

Nodes Added Risoul Looa 2 days ago

You have it

Optix\_EasyModbusTCP\_client.optix.design Added Risoul Logo 2 days ago

Readme.md Add files via upload 4 minutes ago

README

**Modbus server with FTOptix**

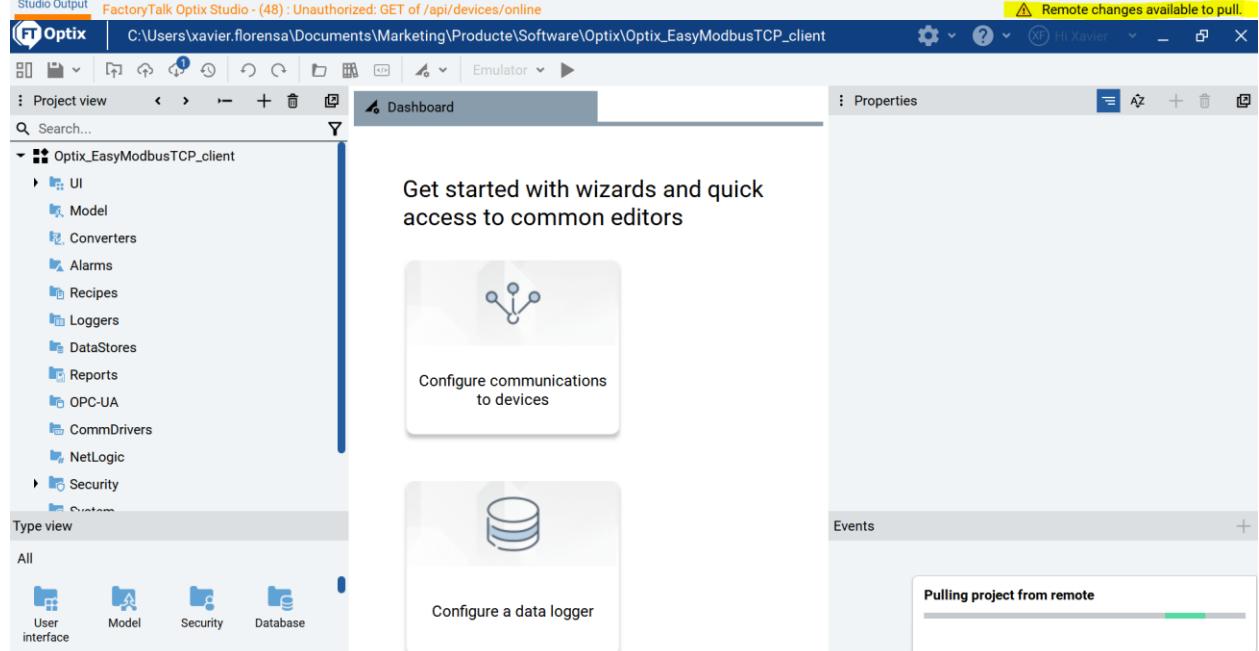
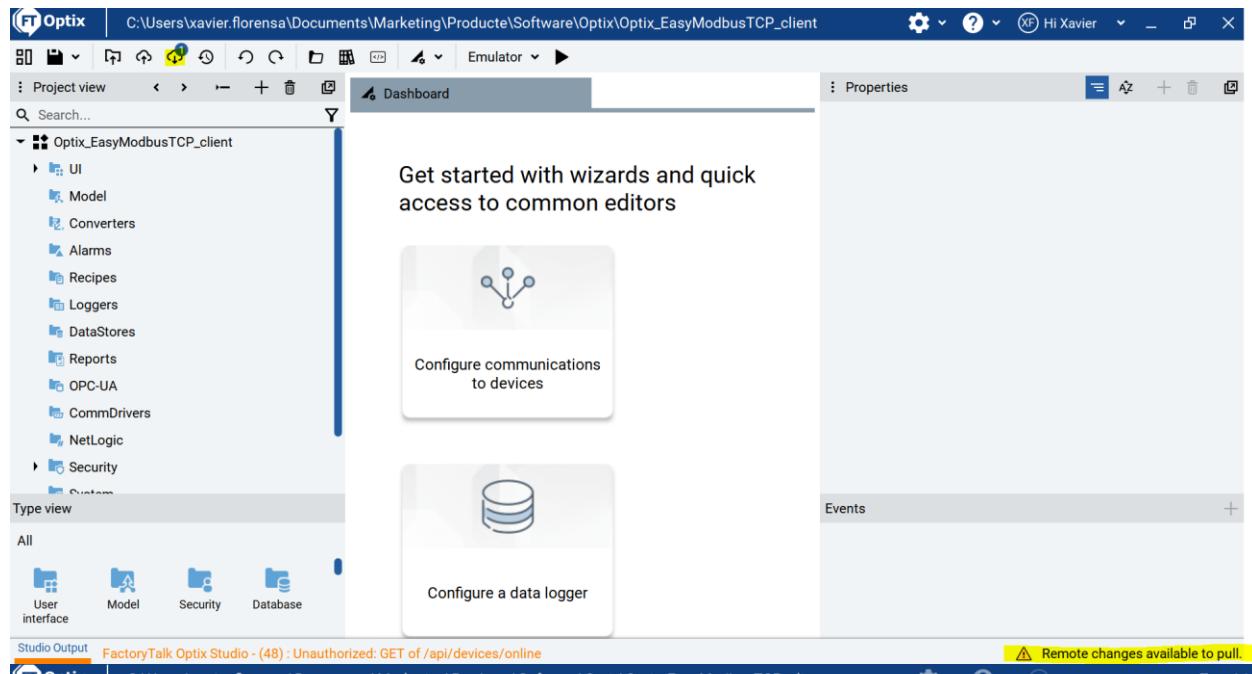
You can try this example on Modbus TCP Server from Rockwell Automation repository

You can download the Server FactoryTalk Optix code from here  
<https://github.com/FactoryTalk-Optix/ModbusServerTCP>

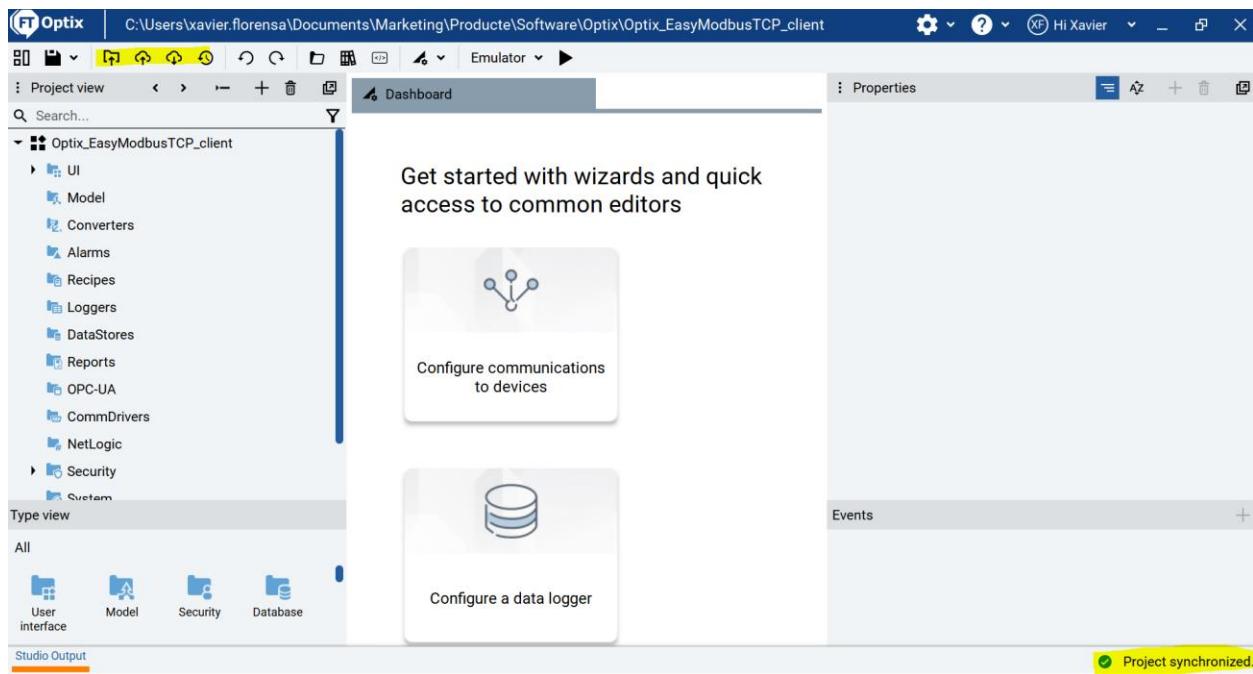
You can see this working in this video  
<https://youtu.be/UpRB72bkKkA>

.NET .NET Desktop

But your local project will not be synchronized  
Then when you open the project locally on your computer  
You will see this



Now your project is synchronized



## 18.2. initUsing Git

First install Git on your machine

This is explained here

<https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github>

Locate the directory with your Optix project

Take a look at the path, you can not have spaces in between.

C:\Users\xavier.florensa\Documents\Rockwell Automation\FactoryTalk Optix\Projects

If so copy the directory on a shorter path like this

C:\Users\xavier.florensa\Documents\Optix

Open Git Bash

```
MINGW64:/c/Users/xavier.florensa
xavier.florensa@CATPROLP1003-23 MINGW64 ~
$ |
```

Navigate to the root directory of your project

---

```
MINGW64:/c/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git

xavier.florensa@CATPROLP1003-23 MINGW64 ~
$ cd Documents

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents
$ cd Optix

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix
$ cd PLC2MQTTClient_HiveMQ_v1_Git

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git
$ |
```

Inicialize a Git repository locally

```
git init -b main
```

---

```
MINGW64:/c/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git

xavier.florensa@CATPROLP1003-23 MINGW64 ~
$ cd Documents

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents
$ cd Optix

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix
$ cd PLC2MQTTClient_HiveMQ_v1_Git

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git
$ git init -b main
Initialized empty Git repository in C:/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git/.git/
xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ |
```

Add the files to your local new repository

```
git add .
```

be aware that (main) in blue color is the branch you are working now

---

```
MINGW64:/c/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git

xavier.florensa@CATPROLP1003-23 MINGW64 ~
$ cd Documents

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents
$ cd Optix

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix
$ cd PLC2MQTTClient_HiveMQ_v1_Git

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git
$ git init -b main
Initialized empty Git repository in C:/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git/.git/
xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git add .

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ |
```

Commit the files you have staged in your local repository

```
git commit -m "First commit"
```

```
MINGW64:/c/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git commit -m "First commit"
[main (root-commit) 8ca9b56] First commit
Committer: Xavier Florensa Berenguer <xavier.florensa@soporte.com>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"
git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

65 files changed, 24341 insertions(+)
create mode 100644 DesignTimeNodes/CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/TagImporter.yaml
create mode 100644 IDEVersion.txt
create mode 100644 Nodes/Alarms/Alarms.yaml
create mode 100644 Nodes/CommDrivers/CommDrivers.yaml
create mode 100644 Nodes/CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPDriver1.yaml
create mode 100644 Nodes/CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/DataTypes/DataTypes.yaml
create mode 100644 Nodes/CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/RAEtherNet_IPStation1.yaml
create mode 100644 Nodes/CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/Tags/Tags.yaml
create mode 100644 Nodes/CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/VariableTypes/VariableTypes.yaml
create mode 100644 Nodes/Converters/Converters.yaml
create mode 100644 Nodes/DataStores/DataStores.yaml
create mode 100644 Nodes/Loggers/Loggers.yaml
create mode 100644 Nodes/Model/Model.yaml
create mode 100644 Nodes/NetLogic/NetLogic.yaml
create mode 100644 Nodes/OPC-UA/OPC-UA.yaml
create mode 100644 Nodes/PLC2MQTTClient_HiveMQ_v1_Git.yaml
create mode 100644 Nodes/Recipes/Recipes.yaml
create mode 100644 Nodes/Reports/Reports.yaml
create mode 100644 Nodes/Retentivity/Retentivity.yaml
create mode 100644 Nodes/Security/Groups/Groups.yaml
create mode 100644 Nodes/Security/Security.yaml
create mode 100644 Nodes/Security/Users/Users.yaml
create mode 100644 Nodes/Translations/Translations.yaml
create mode 100644 Nodes/UI/UI.yaml
create mode 100644 PLC2MQTTClient_HiveMQ_v1_Git.optix
create mode 100644 PLC2MQTTClient_HiveMQ_v1_Git.optix.design
create mode 100644 ProjectFiles/NetSolution/.vscode/launch.json
create mode 100644 ProjectFiles/NetSolution/MQTTClient.code-workspace
create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient.code-workspace
create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_v1.code-workspace
```

```

MINGW64:/c/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git
create mode 100644 Nodes/Security/Users/Users.yaml
create mode 100644 Nodes/Translations/Translations.yaml
create mode 100644 Nodes/UI/UI.yaml
create mode 100644 PLC2MQTTClient_HiveMQ_v1_Git.optix
create mode 100644 PLC2MQTTClient_HiveMQ_v1_Git.optix.design
create mode 100644 ProjectFiles/NetSolution/.vscode/launch.json
create mode 100644 ProjectFiles/NetSolution/MQTTClient.code-workspace
create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient.code-workspace
create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_v1.code-workspace
create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ.code-workspace
create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v0.code-workspace
create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v1.code-workspace
create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v1_Git.csproj
create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v1_Git.references
create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v1_Git.sln
create mode 100644 ProjectFiles/NetSolution/Private/TypeConstants.cs
create mode 100644 ProjectFiles/NetSolution/Private/UITypeDefinitions.cs
create mode 100644 ProjectFiles/NetSolution/PublisherLogic.cs
create mode 100644 ProjectFiles/NetSolution/SubscriberLogic.cs
create mode 100644 ProjectFiles/NetSolution/bin/M2mqtt.Net.dll
create mode 100644 ProjectFiles/NetSolution/bin/PLC2MQTTClient_HiveMQ_v1_Git.deps.json
create mode 100644 ProjectFiles/NetSolution/bin/PLC2MQTTClient_HiveMQ_v1_Git.dll
create mode 100644 ProjectFiles/NetSolution/bin/PLC2MQTTClient_HiveMQ_v1_Git.pdb
create mode 100644 ProjectFiles/NetSolution/obj/Debug/.NETCoreApp,Version=v6.0.AssemblyAttributes.cs
create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.AssemblyInfo.cs
create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.AssemblyInfoInputs.cache
create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.GeneratedMSBuildEditorConfig.editorconfig
create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.assets.cache
create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.csproj.AssemblyReference.cache
create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.csproj.CopyComplete
create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.csproj.CoreCompileInputs.cache
create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.csproj.FileListAbsolute.txt
create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.dll
create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.pdb
create mode 100644 ProjectFiles/NetSolution/obj/Debug/ref/PLC2MQTTClient_HiveMQ_v1_Git.dll
create mode 100644 ProjectFiles/NetSolution/obj/Debug/reftest/PLC2MQTTClient_HiveMQ_v1_Git.dll
create mode 100644 ProjectFiles/NetSolution/obj/PLC2MQTTClient_HiveMQ_v1_Git.csproj.nuget.dgspec.json
create mode 100644 ProjectFiles/NetSolution/obj/PLC2MQTTClient_HiveMQ_v1_Git.csproj.nuget.g.props
create mode 100644 ProjectFiles/NetSolution/obj/PLC2MQTTClient_HiveMQ_v1_Git.csproj.nuget.g.targets
create mode 100644 ProjectFiles/NetSolution/obj/project.assets.json
create mode 100644 ProjectFiles/NetSolution/obj/project.nuget.cache
create mode 100644 ProjectFiles/UserDefinedModule.xml
create mode 100644 ProjectFiles/prova35v0.ACD
create mode 100644 prova35v0.ACD

```

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient\_HiveMQ\_v1\_Git (main)  
\$ |

If you want, you can make a log

If you look at HEAD, it points to the local branch (main) where we are now, and you have also the remote branch (origin/main). Origin is the short name of the remote repository

```

MINGW64:/c/Users/xavier.florensa/documents/optix/plc2mqttclient_hivemq_v1_git

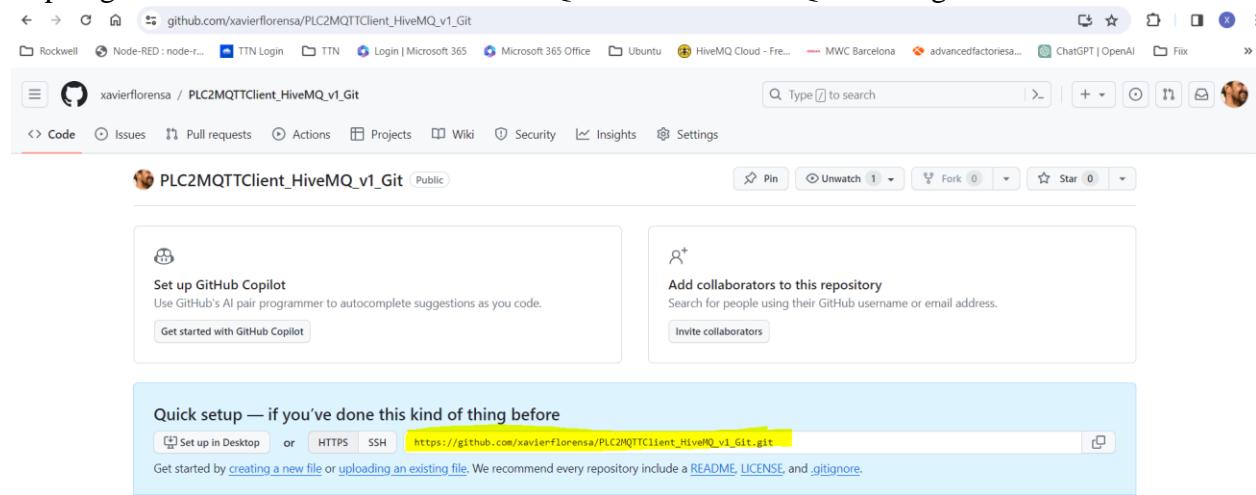
```

Xavier Florensa

Create a new repository with the same name (or similar) on Github

Copy this url

[https://github.com/xavierflorensa/PLC2MQTTClient\\_HiveMQ\\_v1\\_Git.git](https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git)



Quick setup — if you've done this kind of thing before  
Set up in Desktop or HTTPS SSH [https://github.com/xavierflorensa/PLC2MQTTClient\\_HiveMQ\\_v1\\_Git.git](https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git)  
Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

Type this on Git bash while on the project repository (origin is the shortname for the repository and may be whatever you want)

git remote add origin REMOTE-URL

so in this case

git remote add origin

[https://github.com/xavierflorensa/PLC2MQTTClient\\_HiveMQ\\_v1\\_Git.git](https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git)

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git remote add origin https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git
```

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ |
```

You can skip this step, the branch on the remote repository will be created anyway with the git push, you will do later.

git branch -M main

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git branch -M main
```

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ |
```

You have two branches now, the local and the remote

```
MINGW64:/c/Users/xavier.florensa/documents/optix/plc2mqttclient_hivemq_v1_git
xavier.florensa@CATPROLP1003-23 MINGW64 ~/documents/optix/plc2mqttclient_hivemq_v1_git (main)
$ git branch --all
* main
  remotes/origin/main
```

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~/documents/optix/plc2mqttclient_hivemq_v1_git (main)
$ |
```

You can list the remote repository connected to the local repository



Connect to GitHub

X

# GitHub

## Sign in

Browser/Device

Token

[Sign in with your browser](#)

[Sign in with a code](#)

Don't have an account? [Sign up](#)

The screenshot shows the GitHub OAuth authorization interface. At the top, there's a header bar with browser navigation icons and the URL [github.com/login/oauth/authorize?response\\_type=code&client\\_id=0120e057bd645470c1ed&state=96a1d86d17794fb98cc3331f73a](https://github.com/login/oauth/authorize?response_type=code&client_id=0120e057bd645470c1ed&state=96a1d86d17794fb98cc3331f73a). Below the header is a toolbar with various links: Rockwell, Node-RED : node-r..., TTN Login, TTN, Login | Microsoft 365, Microsoft 365 Office, Ubuntu, HiveMQ Cloud - Fre..., and MV.

The main content area features a circular icon containing a red key and a green checkmark, followed by the GitHub logo. The title "Authorize Git Credential Manager" is centered below these icons.

Below the title, a message from "Git Credential Manager by Git Ecosystem" states: "wants to access your xavierflorensa account".

The interface lists three scopes with checkboxes:

- Gists** Read and write access
- Repositories** Public and private
- Workflow** Update GitHub Action Workflow files.

Under "Organization access", there is one entry:  PacktPublishing ✓.

At the bottom are two buttons: "Cancel" and a prominent green "Authorize git-ecosystem" button.

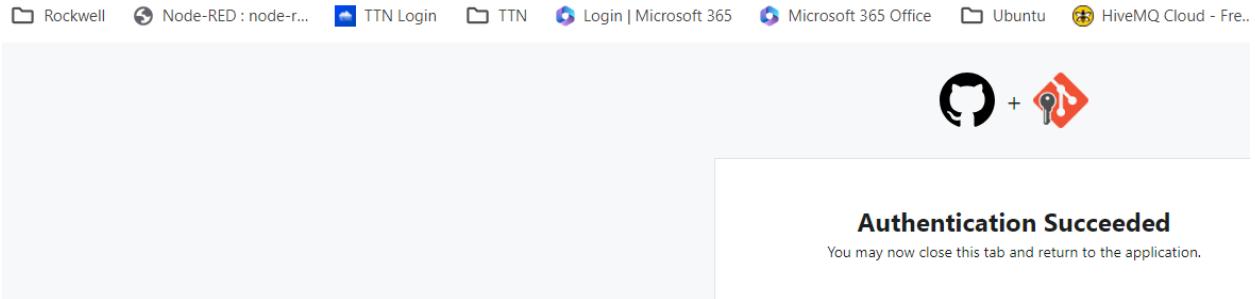
The screenshot shows the GitHub password confirmation interface. At the top, there's a header bar with browser navigation icons and the URL [github.com/login/oauth/authorize](https://github.com/login/oauth/authorize). Below the header is a toolbar with various links: Rockwell, Node-RED : node-r..., TTN Login, TTN, Login | Microsoft 365, Microsoft 365 Office, Ubuntu, HiveMQ Cloud - Fre..., and MV.

The main content area features the GitHub logo at the top. The title "Confirm access" is centered below it.

A message indicates the user is signed in as Signed in as @xavierflorensa.

A password input field is present with a placeholder "Password" and a "Forgot password?" link. A large green "Confirm" button is at the bottom of the form.

A tip at the bottom right states: "Tip: You are entering [sudo mode](#). After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity."



Now take a look at the Github repository  
Voilà (imagine the time you save doing with Optix)

The GitHub repository page for 'PLC2MQTTClient\_HiveMQ\_v1\_Git' shows the following commit history:

- DesignTimeNodes/CommDrivers/RAEtherNet\_L... First commit 22 minutes ago
- Nodes First commit 22 minutes ago
- ProjectFiles First commit 22 minutes ago
- IDEVersion.txt First commit 22 minutes ago
- PLC2MQTTClient\_HiveMQ\_v1\_Git.optix First commit 22 minutes ago
- PLC2MQTTClient\_HiveMQ\_v1\_Git.optix.design First commit 22 minutes ago
- prova35v0.ACD First commit 22 minutes ago

The terminal window shows the following git session:

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git branch -M main

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 93, done.
Counting objects: 100% (93/93), done.
Delta compression using up to 12 threads
Compressing objects: 100% (56/56), done.
Writing objects: 100% (93/93), 3.97 MiB | 1.84 MiB/s, done.
Total 93 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ |
```

Any file you have on the repository will be uploaded to Github, for instance a user manual, etc.  
What if you have forgotten to add a file, no problem you can add from Github directly.  
Just Add file / Upload file

The screenshot shows a GitHub repository page for 'PLC2MQTTClient\_HiveMQ\_v1\_Git'. The 'Code' tab is selected. A modal window titled 'Add file' is open, showing options to 'Create new file' or 'Upload files'. The main repository area lists several commits from 'xavierflorensa' made 19 hours ago, including folder uploads for 'DesignTimeNodes/CommDrivers/RAEtherNet...', 'Nodes', and 'ProjectFiles', and file uploads for 'FactoryTalk Optix and MQTT.pdf', 'IDEVersion.txt', and 'PLC2MQTTClient\_HiveMQ\_v1\_Git.optix'.

https://github.com/xavierflorensa/PLC2MQTTClient\_HiveMQ\_v1\_Git/upload/main

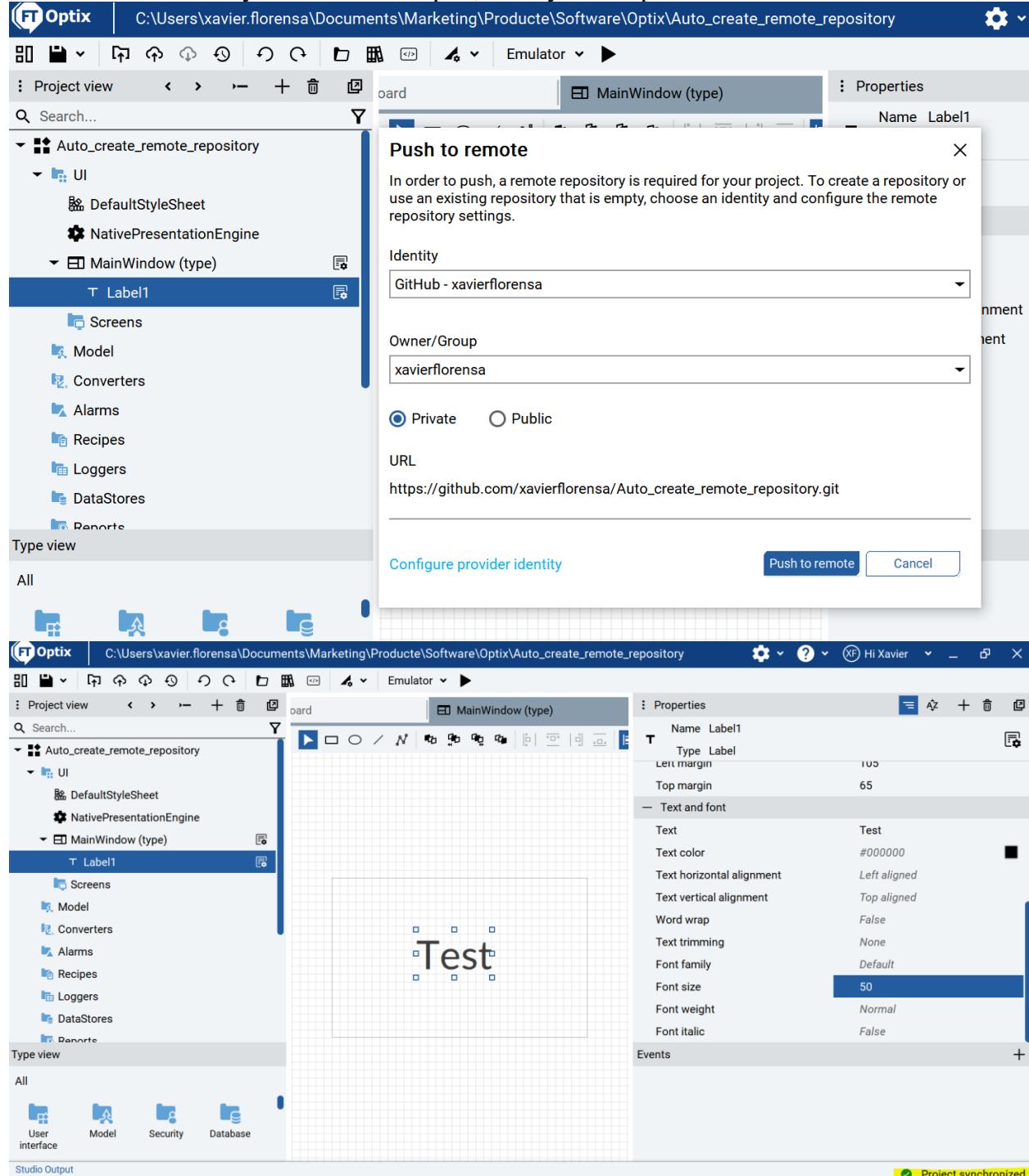
And you have it there

The screenshot shows the same GitHub repository page after pushing via Optix. The 'Code' tab is selected. The commit history now includes a yellow-highlighted entry for 'FactoryTalk Optix and MQTT.pdf', indicating it was added via upload. The rest of the commit history remains the same as in the previous screenshot.

This is not possible if you have used Optix to push the repository

### 18.3. Remote repository creation with FT Optix version 1.3

Now you do not need to create the remote repository before performing a Remote Push. It is done automatically from local FT Optix, in only one step.



You can open your Github account

[github.com/xavierflorensa?tab=repositories](https://github.com/xavierflorensa?tab=repositories)

Xavier Florensa (xavierflorensa) has 79 repositories.

**Auto\_create\_remote\_repository** [Private] Updated now

**NetlogicTutorialSocketClient2** [Public] Works with NetLogicTutorialSocketServer5. Pending to remove Socket close on each message sent.

**NetlogicTutorialSocketServer** [Public]

A new repository appeared

[github.com/xavierflorensa/Auto\\_create\\_remote\\_repository](https://github.com/xavierflorensa/Auto_create_remote_repository)

**Code** Issues Pull requests Actions Projects Security Insights Settings

**Auto\_create\_remote\_repository** [Private]

main Branch Tags Go to file Add file Code

**xavierflorensa** Project creation 01b9f87 · 5 minutes ago 1 Commit

- Nodes Project creation 5 minutes ago
- .gitattributes Project creation 5 minutes ago
- .gitignore Project creation 5 minutes ago
- Auto\_create\_remote\_repository.optix Project creation 5 minutes ago
- Auto\_create\_remote\_repository.optix.design Project creation 5 minutes ago

**About**  
No description, website, or topics provided.

**Activity**  
0 stars 1 watching 0 forks

**Releases**  
No releases published [Create a new release](#)

**Packages**  
No packages published [Publish your first package](#)

README

You can add a Readme.md file after the project is created.  
Upload files

**Optix\_EasyModbusTCP\_client** Public

main 1 Branch 0 Tags

xavierflorensa Add files via upload ce18621

Nodes Added Risoul Looa 2 days ago

**You have it**

github.com/xavierflorensa/Optix\_EasyModbusTCP\_client

Optix\_EasyModbusTCP\_client.optix.design Added Risoul Logo 2 days ago

Readme.md Add files via upload 4 minutes ago

**README**

**Modbus server with FTOptix**

You can try this example on Modbus TCP Server from Rockwell Automation repository

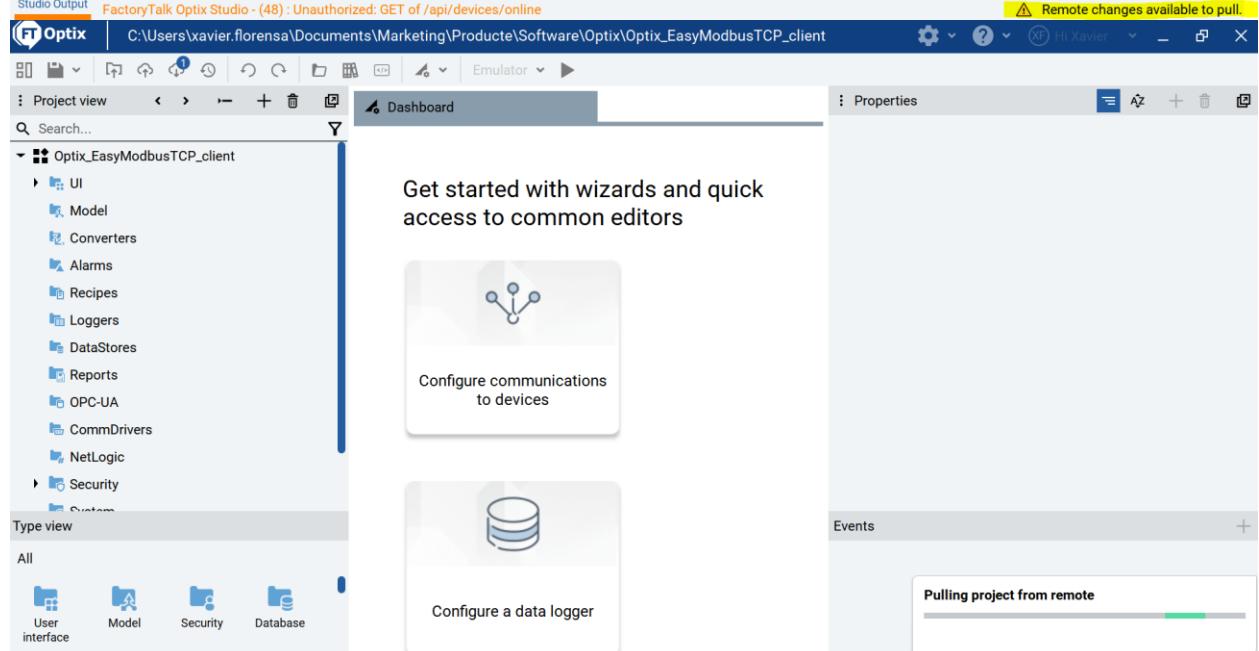
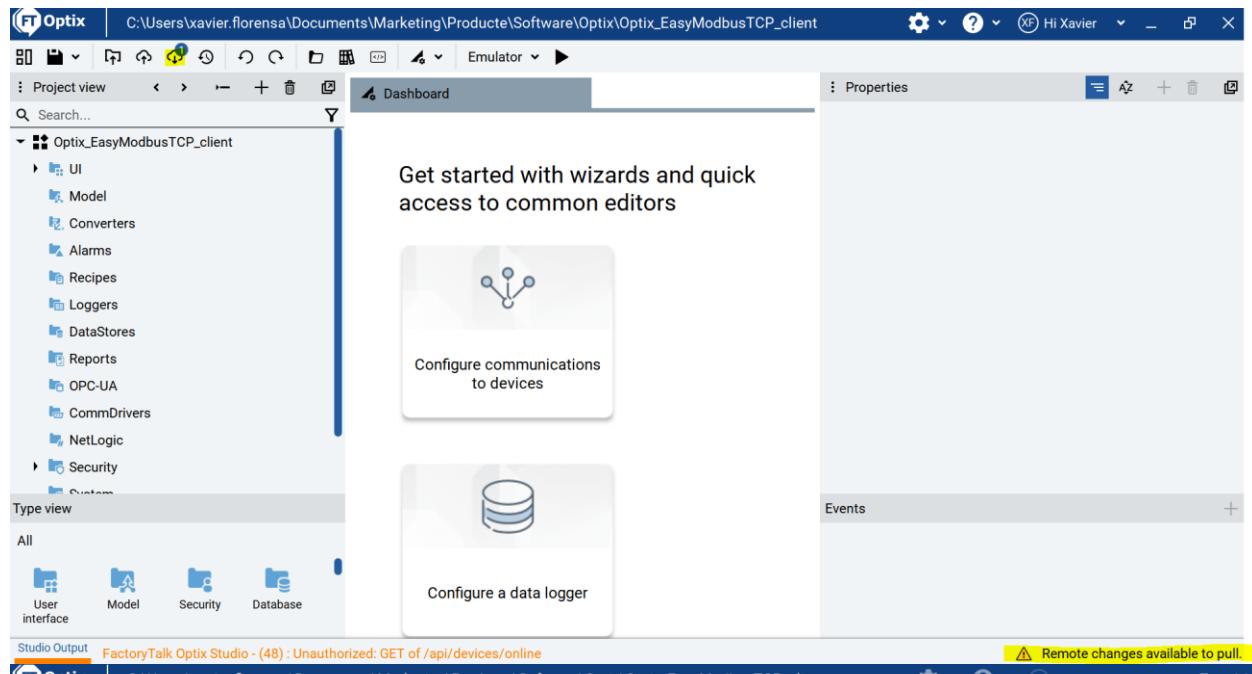
You can download the Server FactoryTalk Optix code from here  
<https://github.com/FactoryTalk-Optix/ModbusServerTCP>

You can see this working in this video  
<https://youtu.be/UpRB72bkKkA>

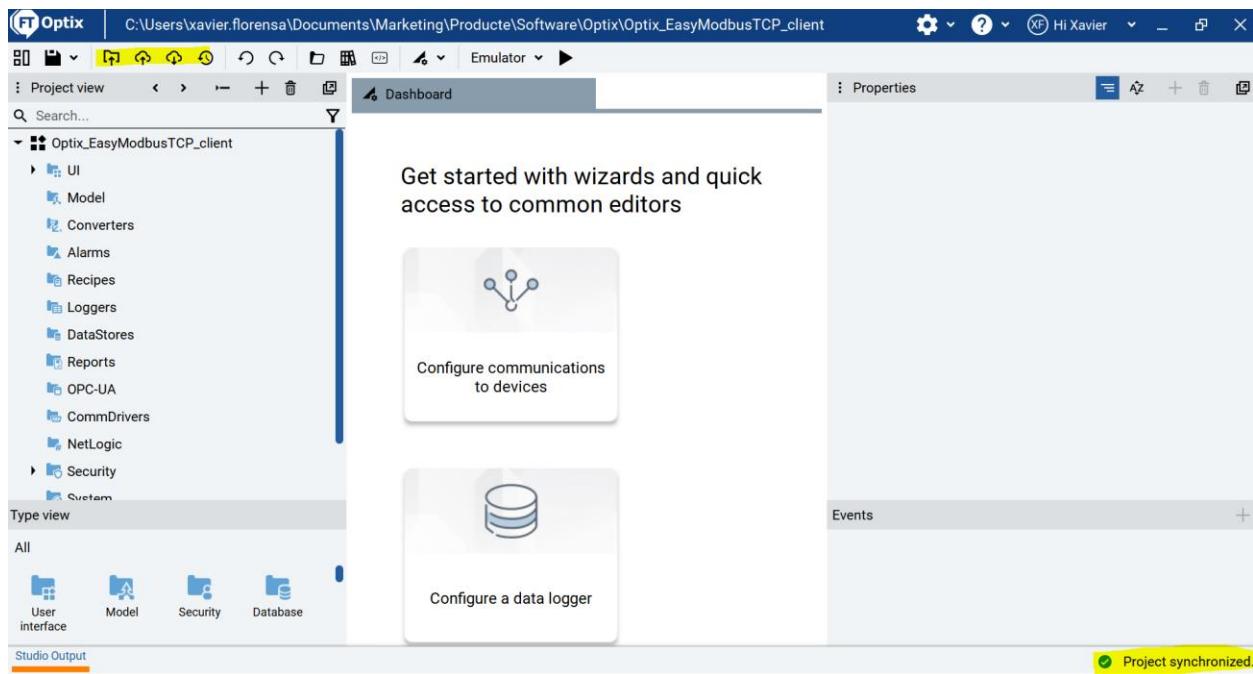
**.NET** .NET Configure Build and test a .NET or ASP.NET Core project.

**.NET Desktop** .NET Desktop Configure Build, test, sign and publish a desktop application built on .NET.

But your local project will not be synchronized  
Then when you open the project locally on your computer  
You will see this



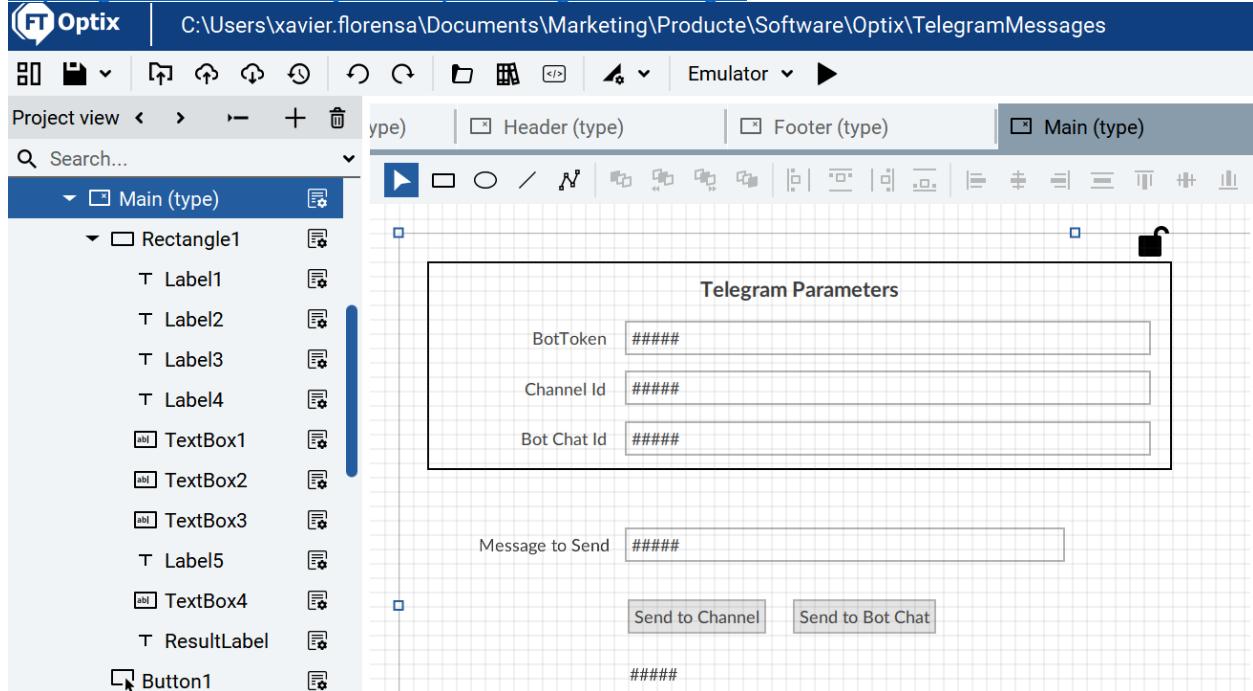
Now your project is synchronized



## 19. Sending Telegram messages

You can use the example from this repository

<https://github.com/FactoryTalk-Optix/TelegramMessages>



## 20. Modbus TCP to OPC UA converter

You can use Optix instead of Kepserver to do so. And in this case Optix is cheaper!

This can be used to read a Modbus PLC from FactoryTalk View

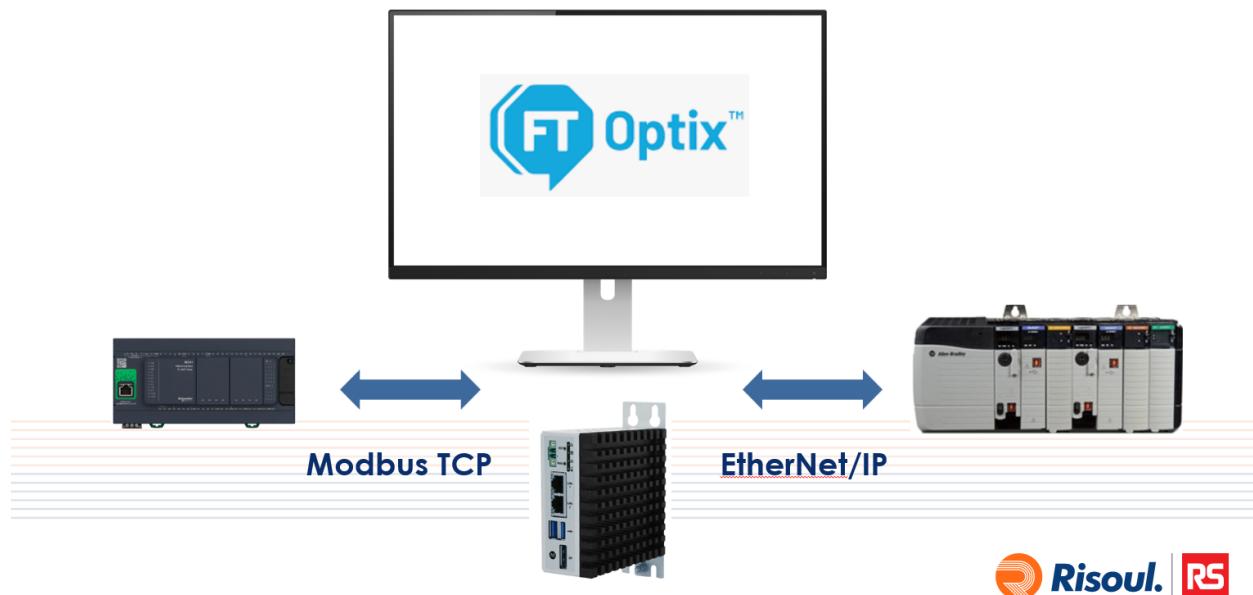
You can see the steps on this video

<https://www.youtube.com/watch?v=WLTmArGiwG8>

To be able to use the Tag importer take a look at chapter [Design time libraries. Importing Modbus Tags from CSV file](#)

## 21. Modbus TCP to EtherNet/IP Gateway

### Modbus TCP to EtherNet/IP Gateway



You can use the following example from Github

<https://github.com/FactoryTalk-Optix/ModbusTCPToRAEthernetIp>

FactoryTalk-Optix/ModbusTCP

<https://github.com/FactoryTalk-Optix/ModbusTCPtoRAEthernetip>

README.md Update README.md last month

Report repository

Releases No releases published

Packages No packages published

Languages C# 100.0%

## Gateway from Modbus to Ethernet IP with Optix

This project shows how to convert tags coming from a Modbus driver to an RA Ethernet IP. Tags are synchronized even if not used in UI.

### Instructions

- Open project
- Define input tags in the Modbus station
- Create Dynamic Links from Modbus tags to Ethernet IP tags (and not viceversa)
- Browse to NetLogic/VariableSynchronizer and populate the TagsToSync variable to the container of the tags to be synchronized
- Launch the application

### This is the result

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
400001-400...	8976	0	0	0	0	0	0	0	0	0
400011-400...	0	0	0	0	0	0	0	0	0	0
400021-400...	0	0	0	0	0	0	0	0	0	0

Connected (1/10) : (received/sent) (274223/274221) Serv. res: Rx: Tx: 0

Address: H D I/O Holding Regs (400000) Fmt: word 16+ Prot: MODBUS TCP/IP

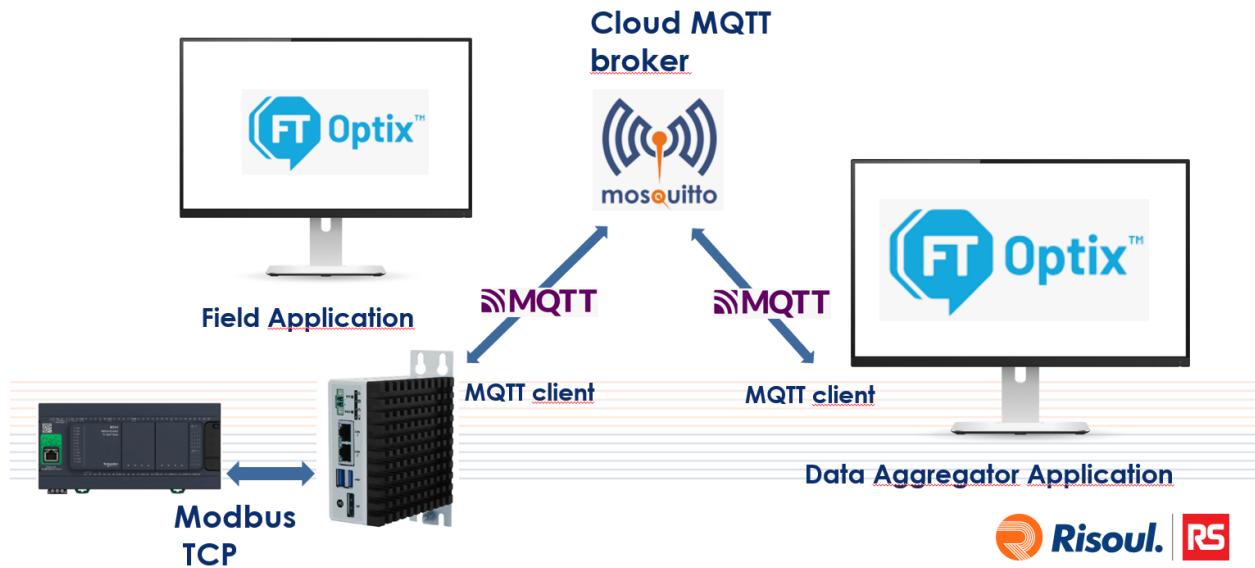
Communication Software: FactoryTalk

As you can see on this video

<https://youtu.be/UMII0TQ1uGM>

## 22. Modbus to MQTT data aggregator

## Modbus TCP to MQTT data aggregator



You can see the process here  
<https://youtu.be/PRgpzygpefA>

## 23. Creating, importing C# .Net libraries to your environment

### 23.1. Design time libraries. Importing Modbus Tags from a CSV File.

Let's import one library to our environment.

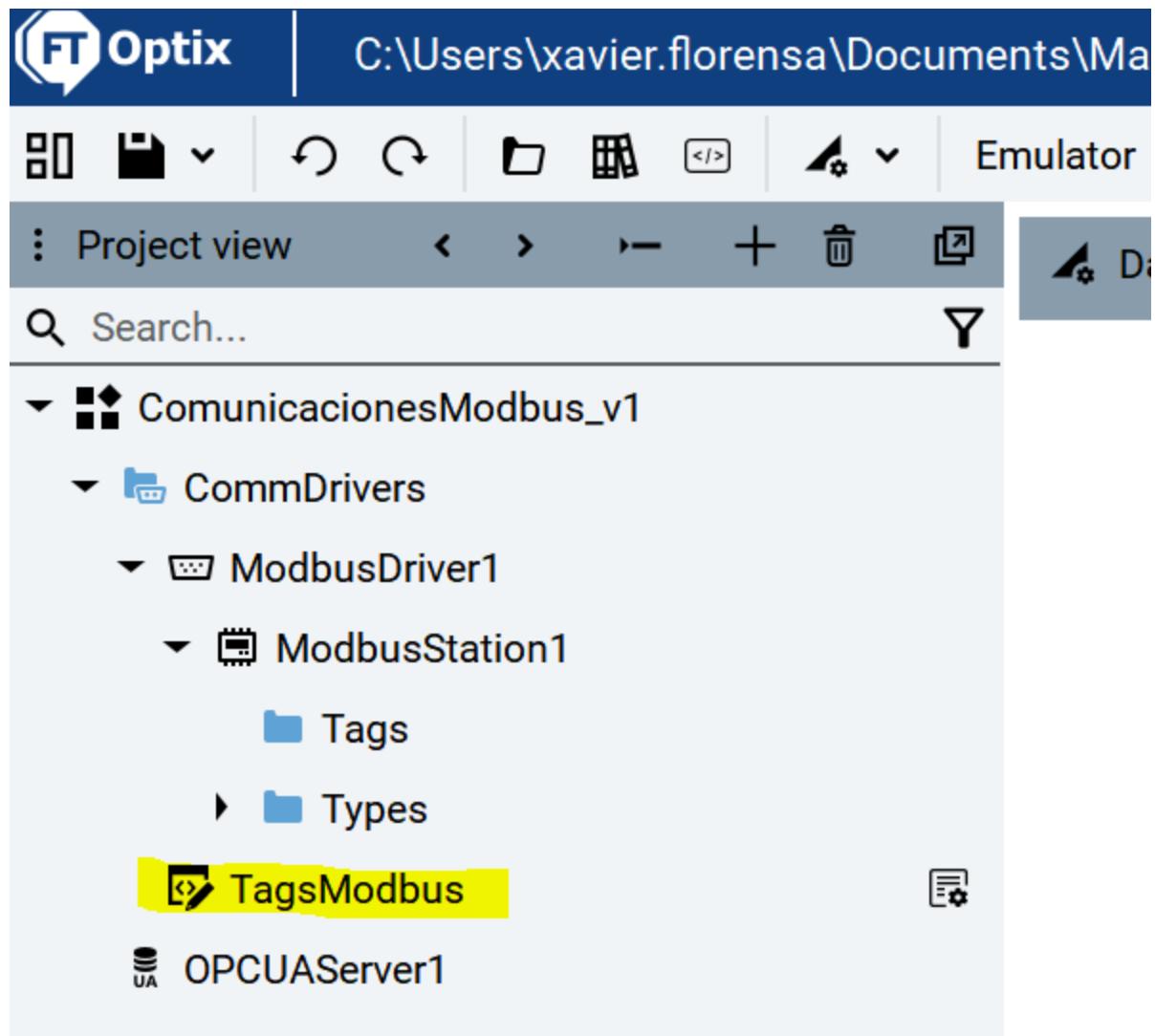
For instance this useful Modbus variable generator described on this video

<https://www.youtube.com/watch?v=WLTmArGiwG8>

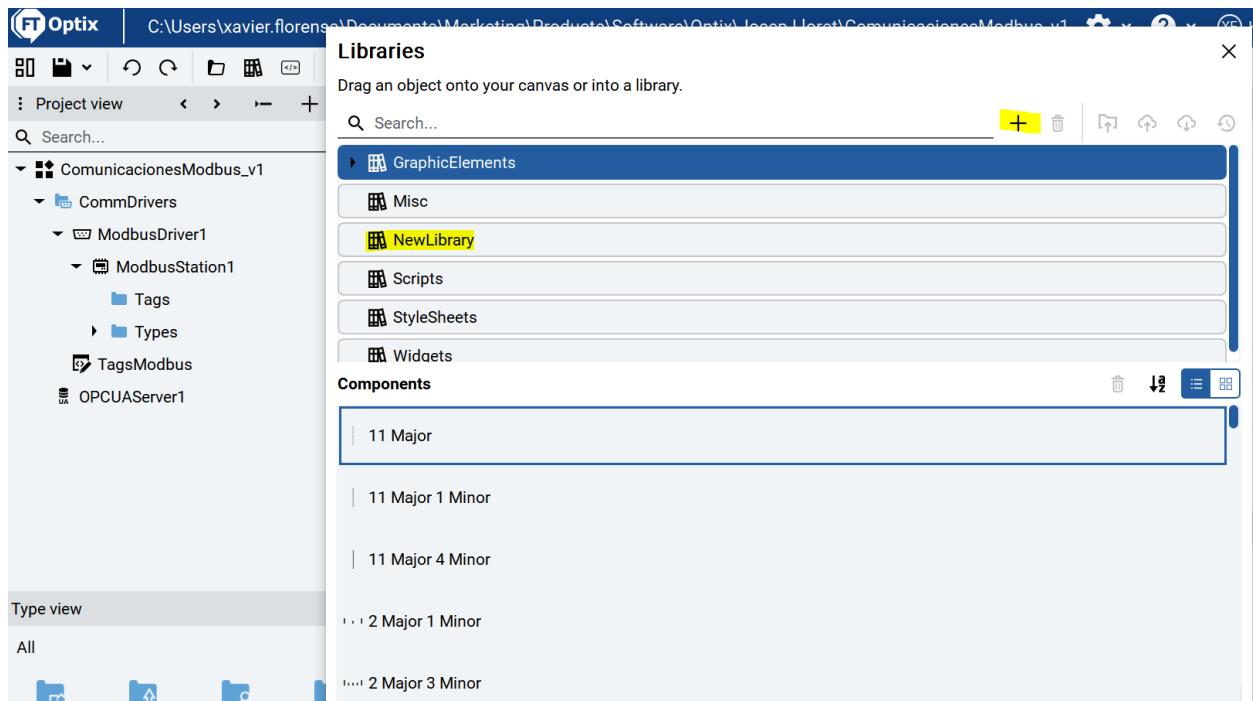
From an example code I can take the .net code and store in my libraries collection to be used later

Let's open the example program

And this is the code I want to store in my libraries

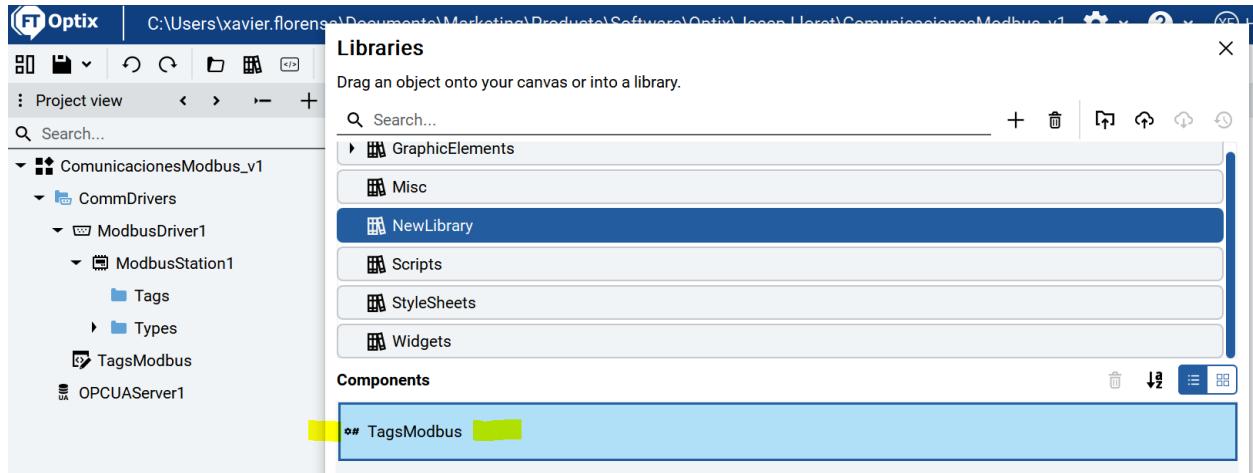


Let's open libraries and create a new library



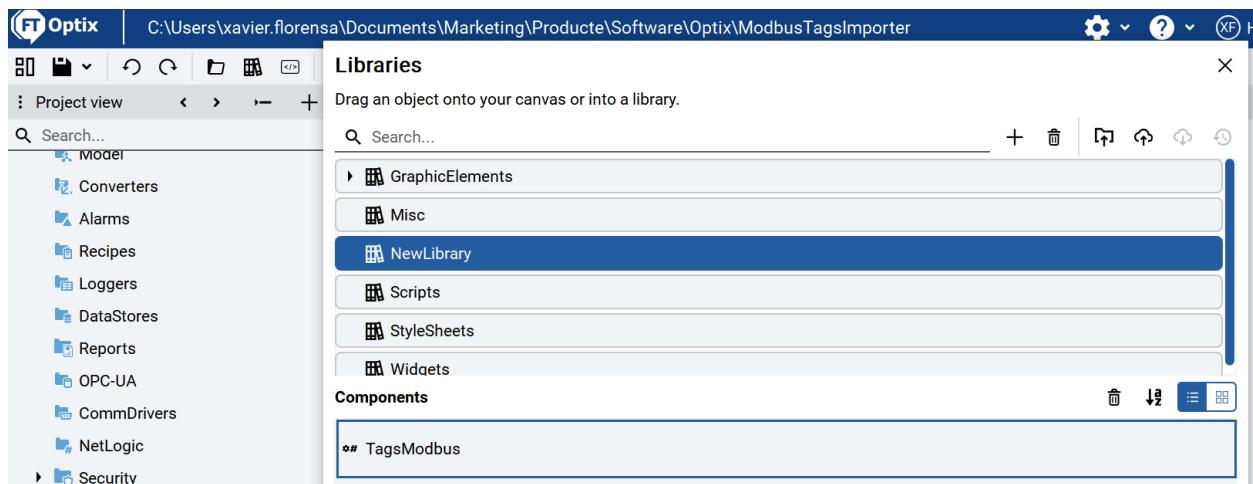
Then drag and drop the code object to the library

We have it

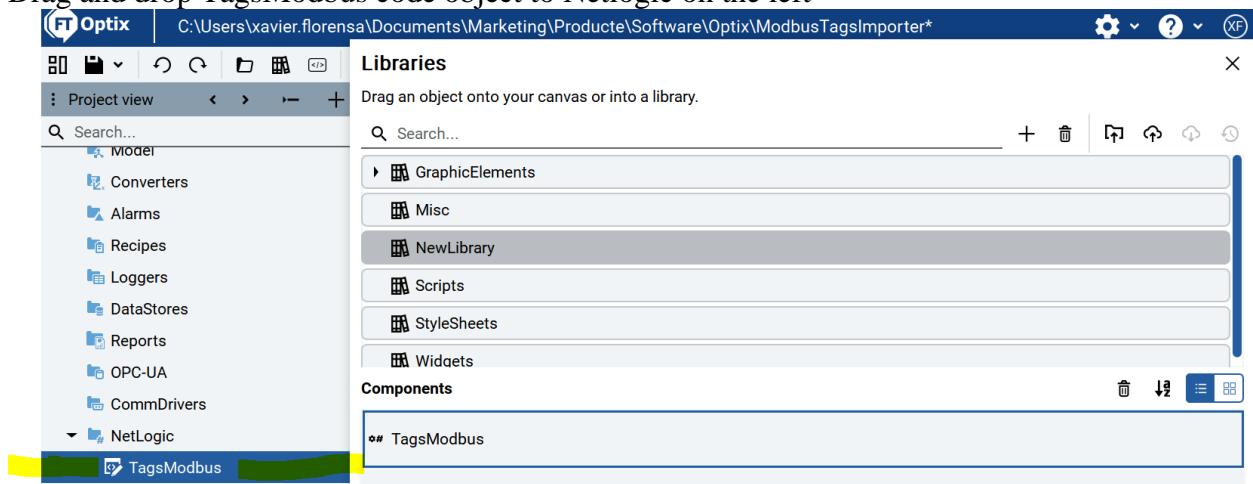


Now let's try to use on a new project

Create a new project



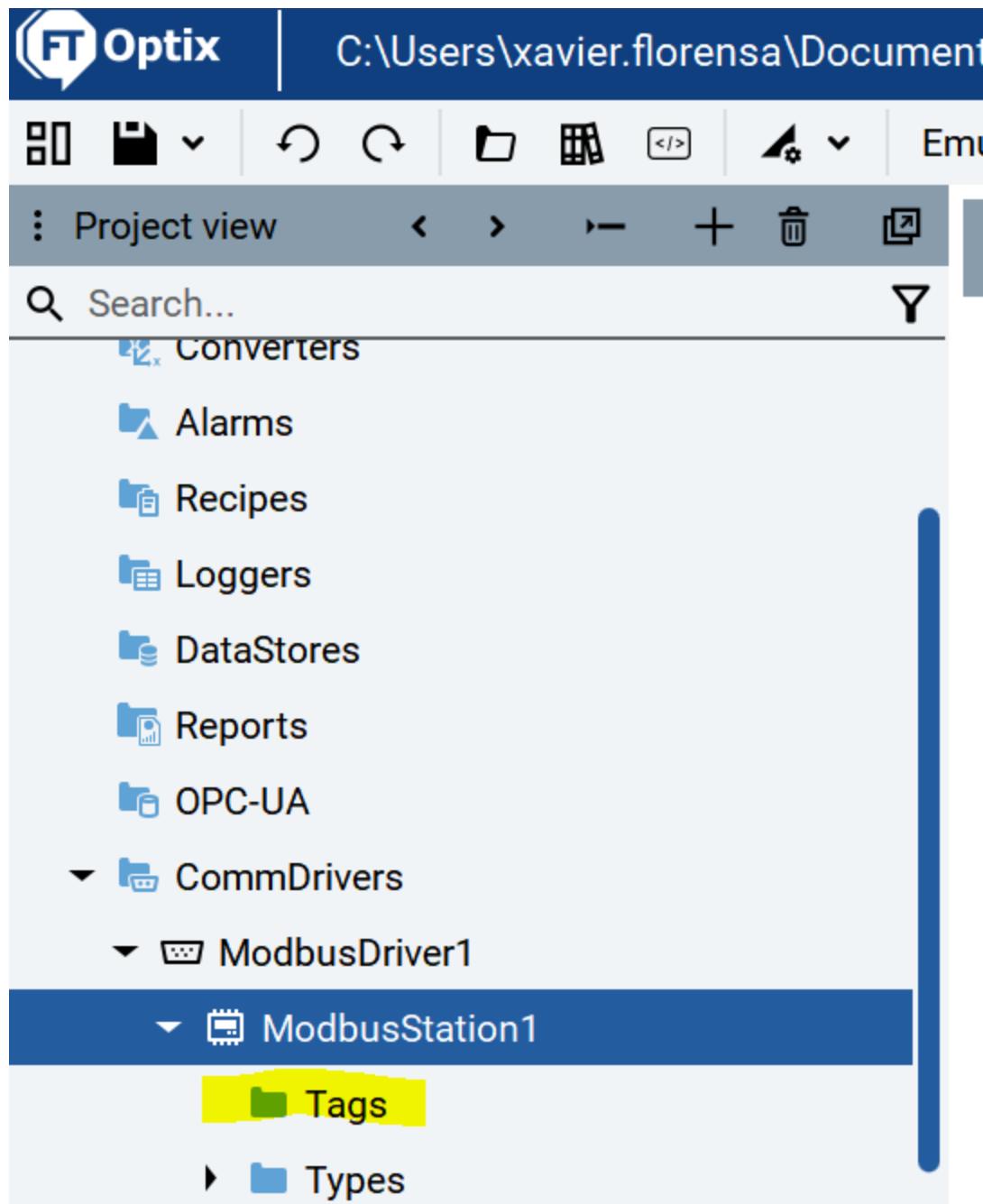
Drag and drop TagsModbus code object to Netlogic on the left



Now let's test it

First of all let's create a Modbus driver with default names

The Tags folder is empty



Now let's grab our Tag list in Excel. It must be on this location and must have this structure

Name	Type	Value
CsvPath	AbsoluteResourceUri	C:\Temp\ModbusTags\ModbusTagsFile.csv
FieldSeparator	String	,
WrapFields	Boolean	False
StartingComment	String	#
ModbusDriver	Nodeld	<Unresolved>

A	B	C	D	E	F	G	H	I	J	K	L	M
1	Variable Name		IsStruct	Type	Array	Elen	Array Upd	Maximum Memory	Register	BitOffset	NumCoil	NumDiscreteInput
2	Temperatu	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	0	0	0	0
3	LTB10610	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	10	0	0	0
4	LTB10620	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	20	0	0	0
5	LTB10630	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	30	0	0	0
6	LTB10640	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	40	0	0	0
7	LTB10650	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	50	0	0	0
8	LTB10660	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	60	0	0	0
9	LTB10670	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	70	0	0	0

So let's execute the code on design time

Project view

- Converters
- Alarms
- Recipes
- Loggers
- DataStores
- Reports
- OPC-UA
- CommDrivers
  - ModbusDriver1
    - ModbusStation1
      - Tags
      - Types
- NetLogic

Dashboard

Get started...

Execute ExportTags

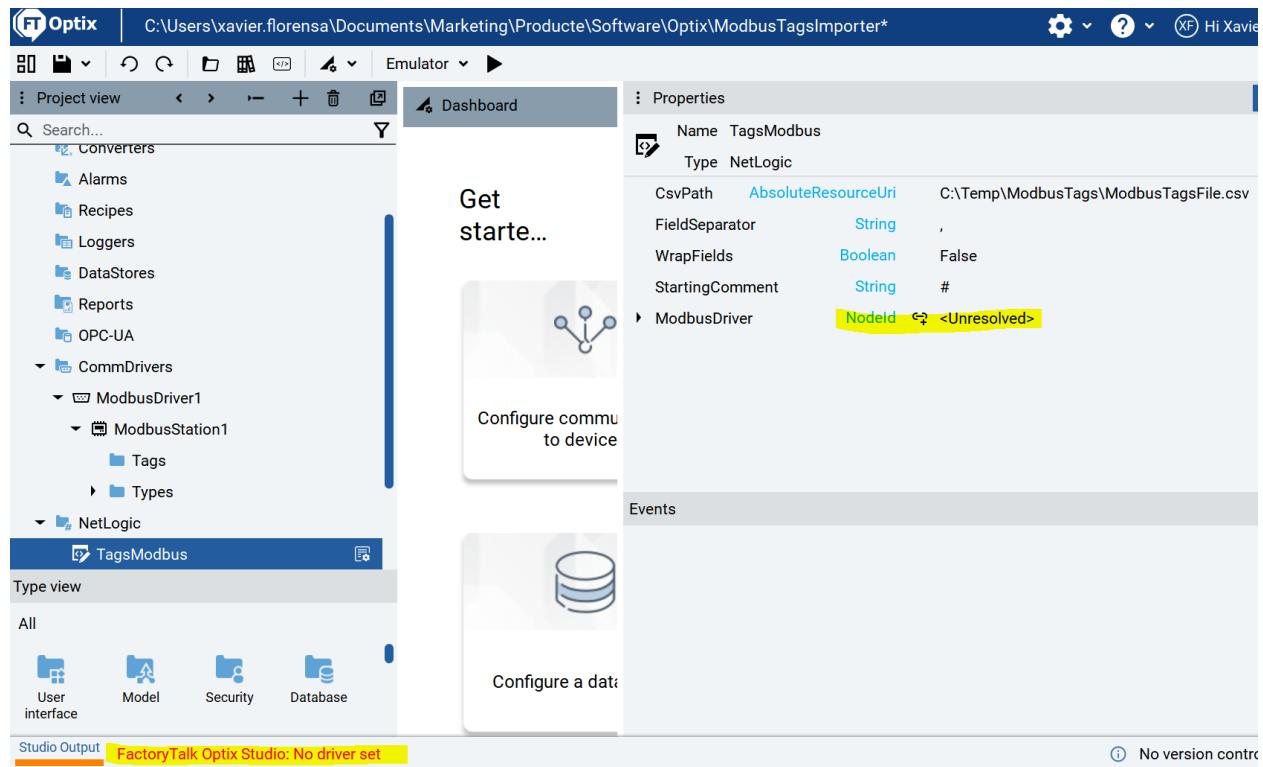
Execute ImportTags

Edit with .NET code editor (external)

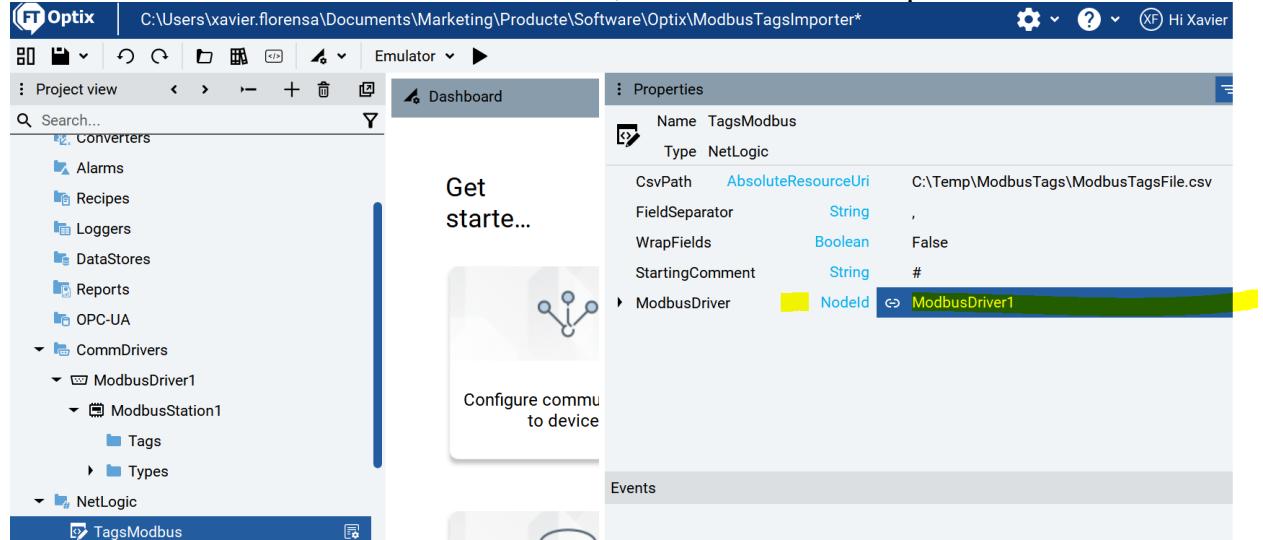
New

Copy

We get an error message



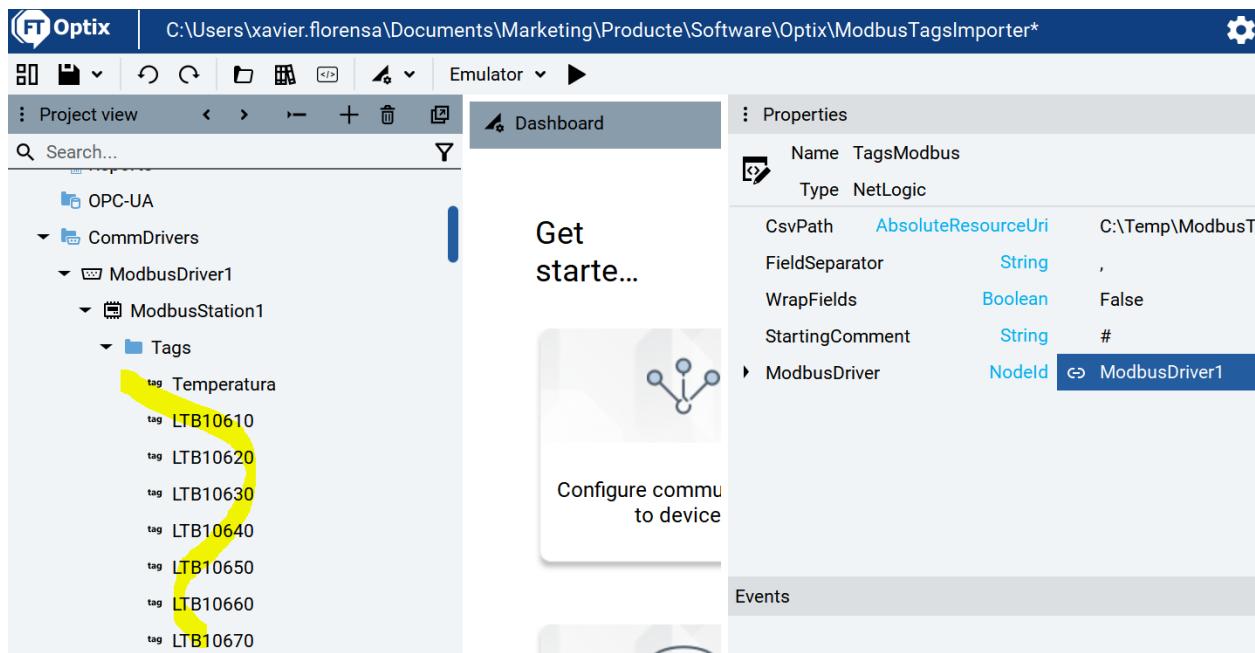
Be aware that the ModbusDriver is Unresolved, we have to solve this problem



Now let's try to execute again

Be sure to close the excel file. If you have the file open, Optix will complain and will not do the task

Voilà

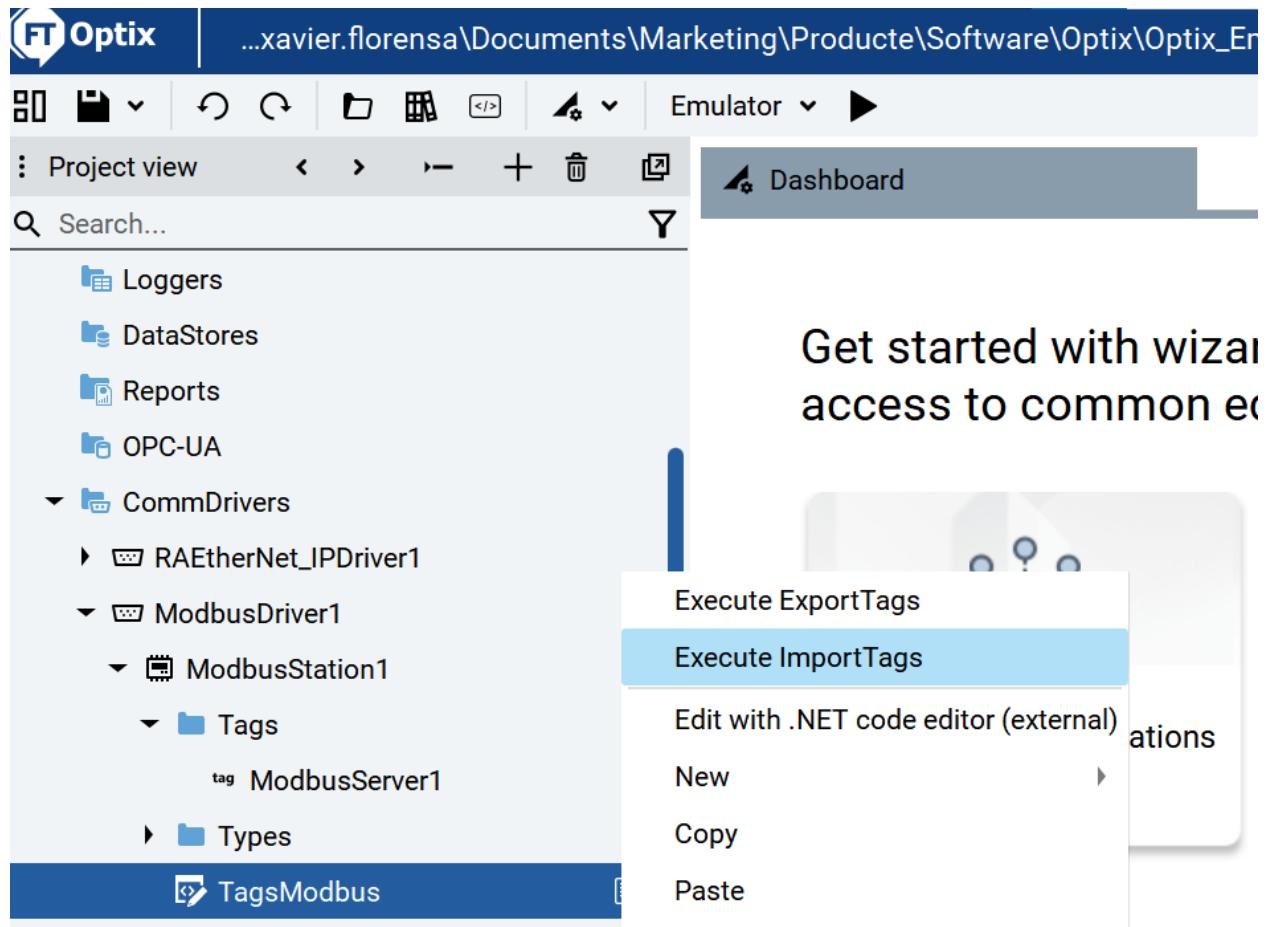


You can even drag and drop the code into Modbus driver instead of into Netlogic  
 Imagine you want to import an array of Tags

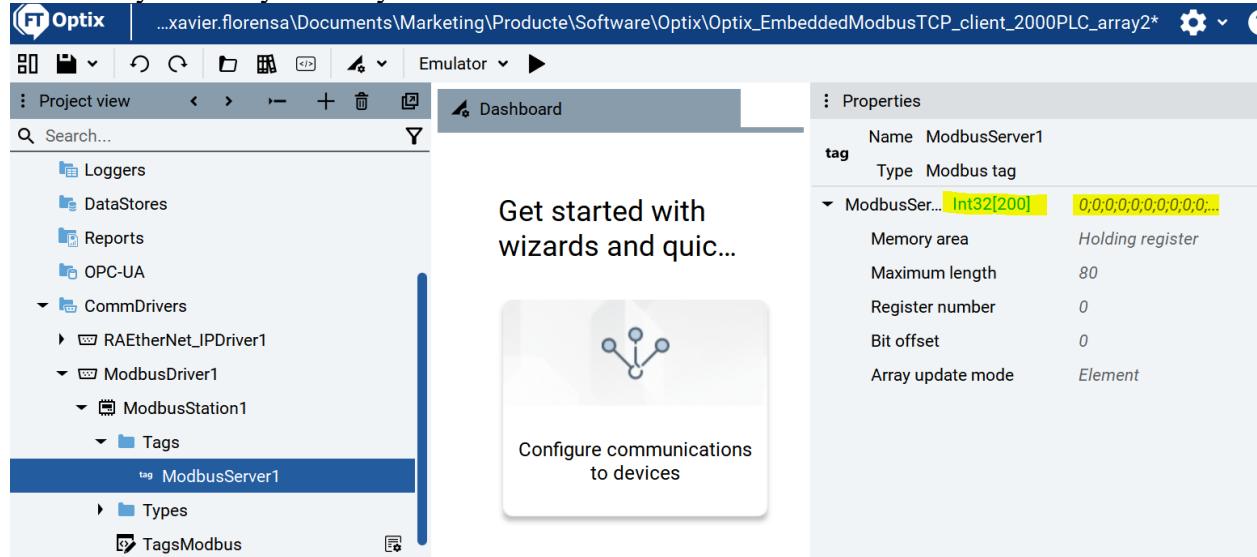
Then you have to set up your CSV file this way:

A	B	C	D	E	F	G	H	I	J	K	L	M
Variable Name	Path	IsStructure	Type	Array Elements	Update Mode	Maximum	Memory Area	Register N	BitOffset	NumCoil	NumDiscreteInput	
1	ModbusServer1 CommDrivers/ModbusDriver1/ModbusStation1/Tags		0 Int32	200 Element		80	HoldingRegister	0	0	0	0	

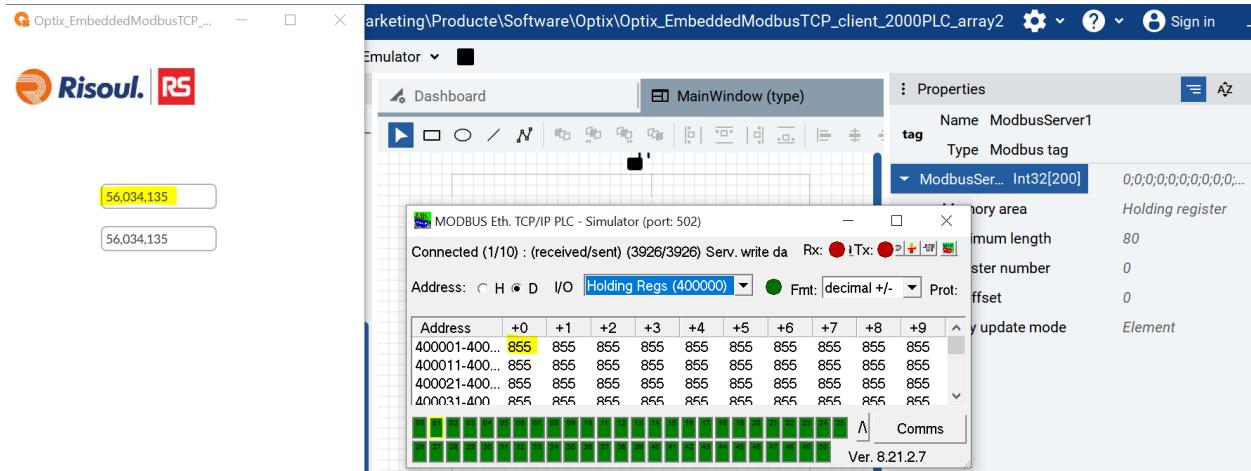
Then execute the ImportTags



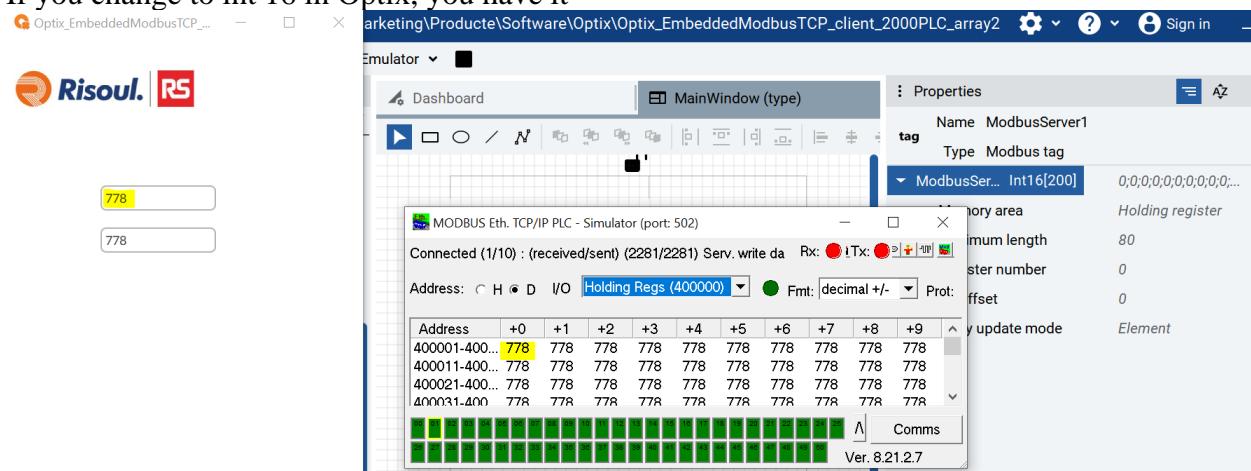
And here you have your array



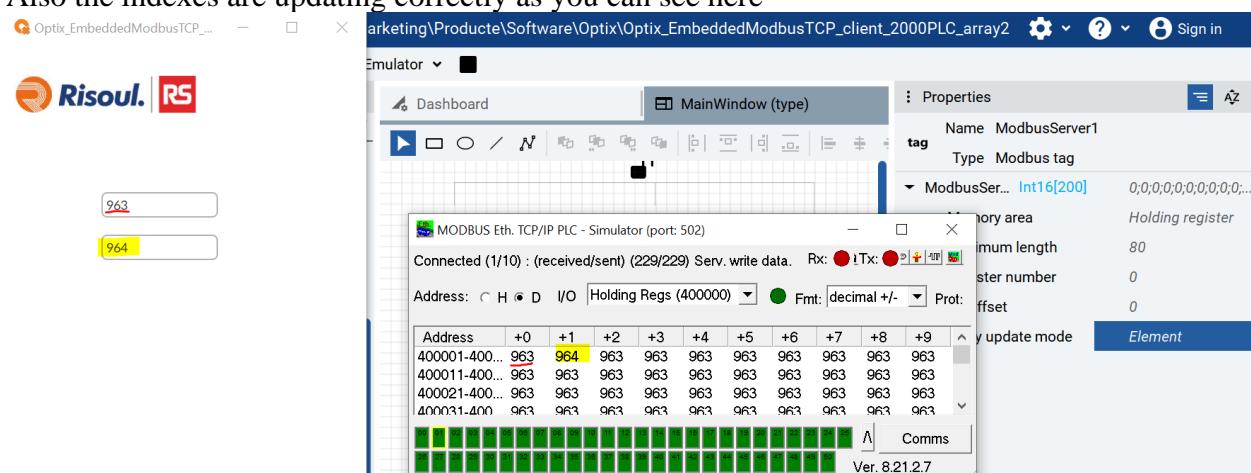
If you try to execute the project you will not see the same number on the Modbus Device and in Optix, since the Modbus device is Int16 and in Optix we have selected Int32.



If you change to int 16 in Optix, you have it



Also the indexes are updating correctly as you can see here



## 23.2. Runtime libraries

I realized that on the MQTT examples on the Rockwell automation repository they use the same code

Let's store in our library to be used later on.

The screenshot shows the Optix software interface with the following details:

- Project view:** Shows a tree structure with categories like Model, Alarms, Recipes, Loggers, DataStores, Reports, OPC-UA, CommDrivers, NetLogic, MQTTBrokerLogic (selected), VariableGenerator, and Security.
- Dashboard:** Displays a "Get started with wiz..." section featuring a network icon and a "Configure communication to devices" section with a cylinder icon.
- Properties:** A table showing the configuration for the MQTTBrokerLogic component.

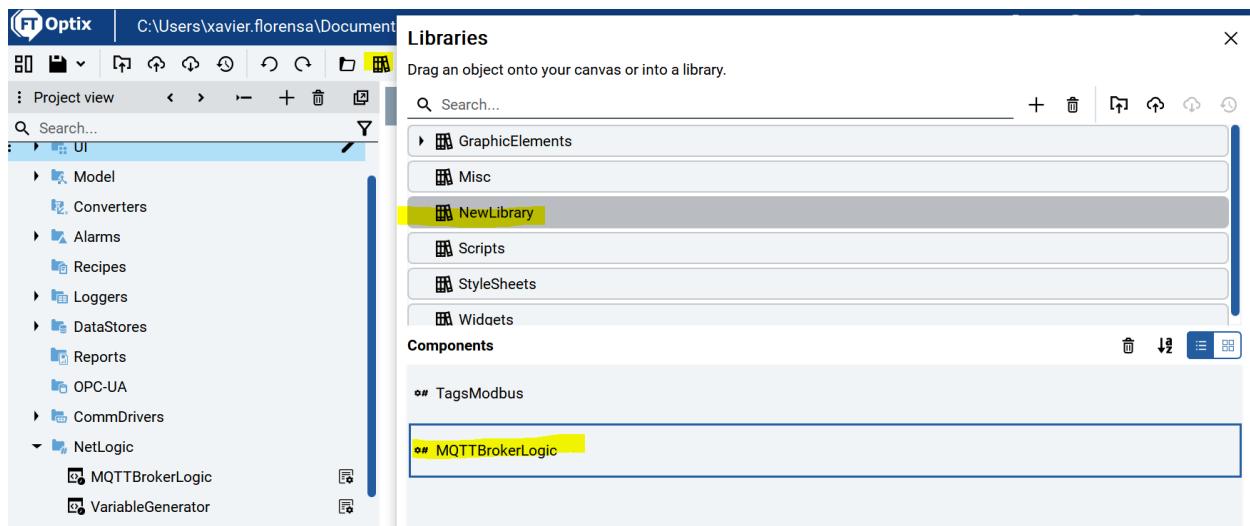
Name	Type	Value
Name	MQTTBrokerLogic	
Type	NetLogic	
AutoStart	Boolean	True
UserAuthentication	Boolean	False
AuthorizedUsers	String[0]	User1
IsRunning	Boolean	False
IsDebuggingMode	Boolean	True
MaxNumberOfConnecti...	Int32	10
NumberOfConnections	Int32	0
MqttClient	Boolean	True
IPAddress	String	test.mosquitto.org
Port	UInt16	1883
UseSSL	Boolean	False
CaCertificate	ResourceUri	%PROJECTDIR%\ca.crt
ClientCertificate	ResourceUri	%PROJECTDIR%\client.crt

Let's create a new library to store our libraries

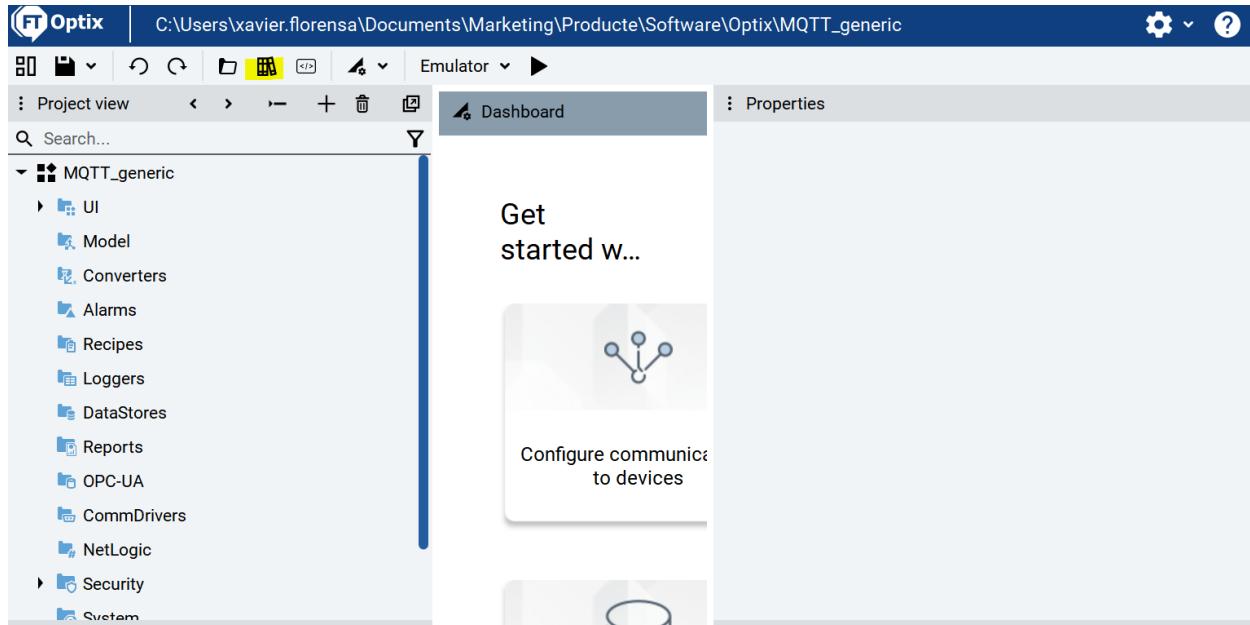
The screenshot shows the Optix software interface with the following details:

- Project view:** Shows a tree structure with categories like Model, Alarms, Recipes, Loggers, DataStores, Reports, OPC-UA, CommDrivers, NetLogic, MQTTBrokerLogic (selected), VariableGenerator, and Security.
- Libraries:** A modal window titled "Libraries" with the instruction "Drag an object onto your canvas or into a library." It contains a search bar and a list of items:
  - GraphicElements
  - Misc
  - NewLibrary (highlighted)
  - Scripts
  - StyleSheets
  - Widgets
- Components:** A list containing "TagsModbus".

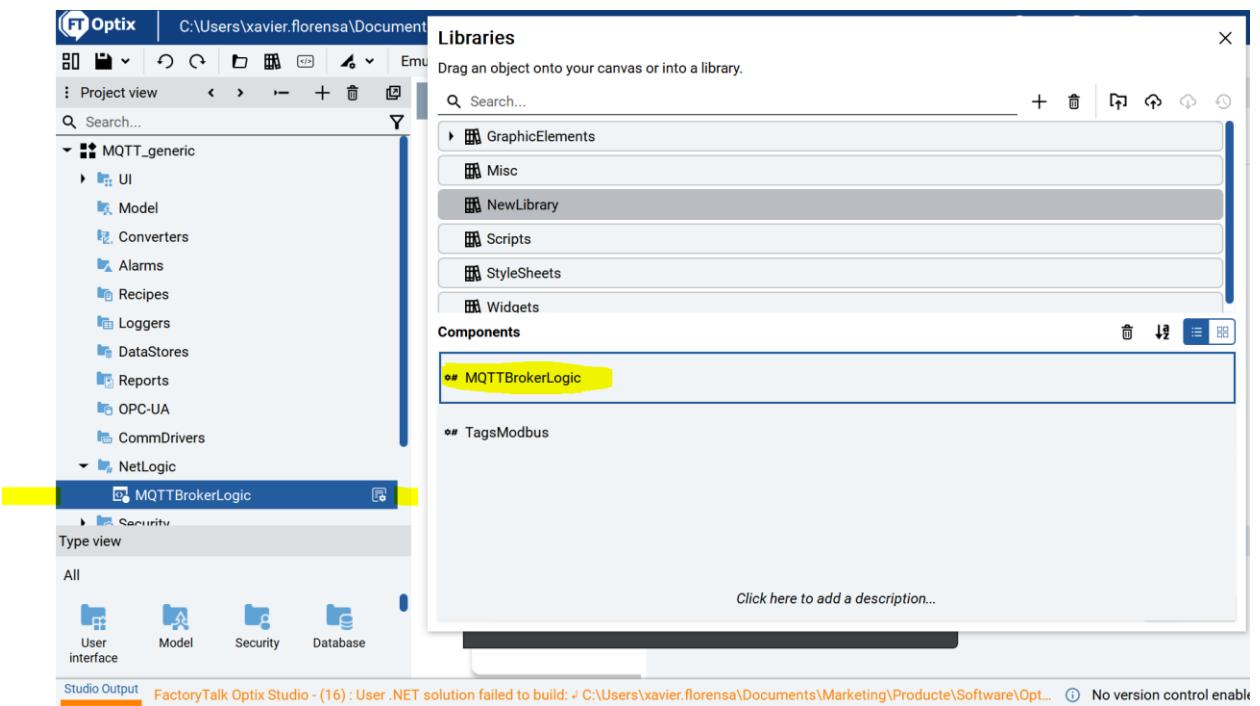
Drag and drop the library



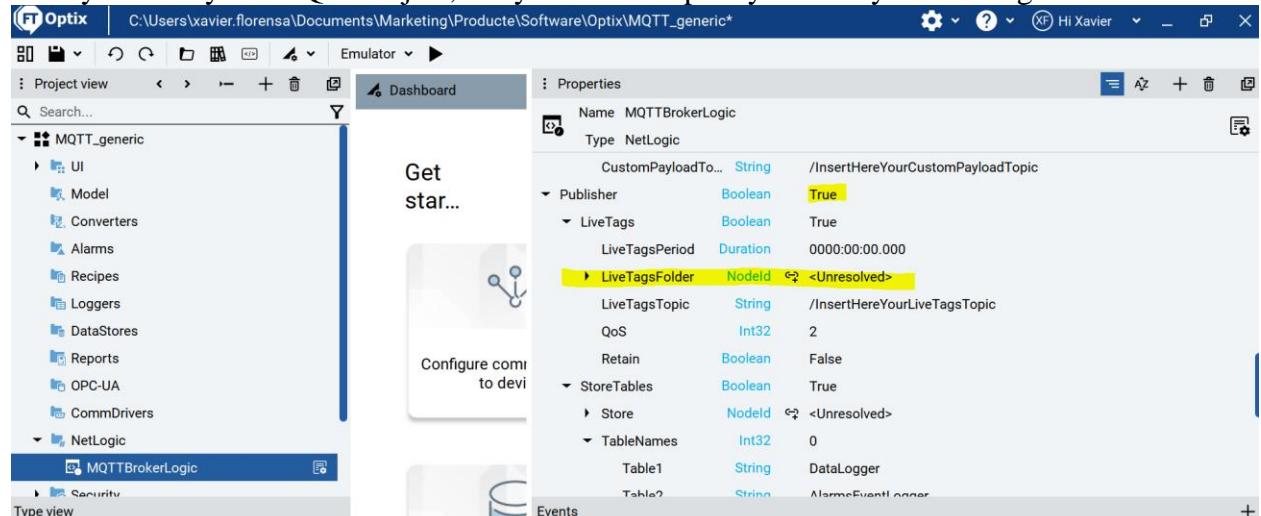
You can close the project and open a new one  
Go to libraries.



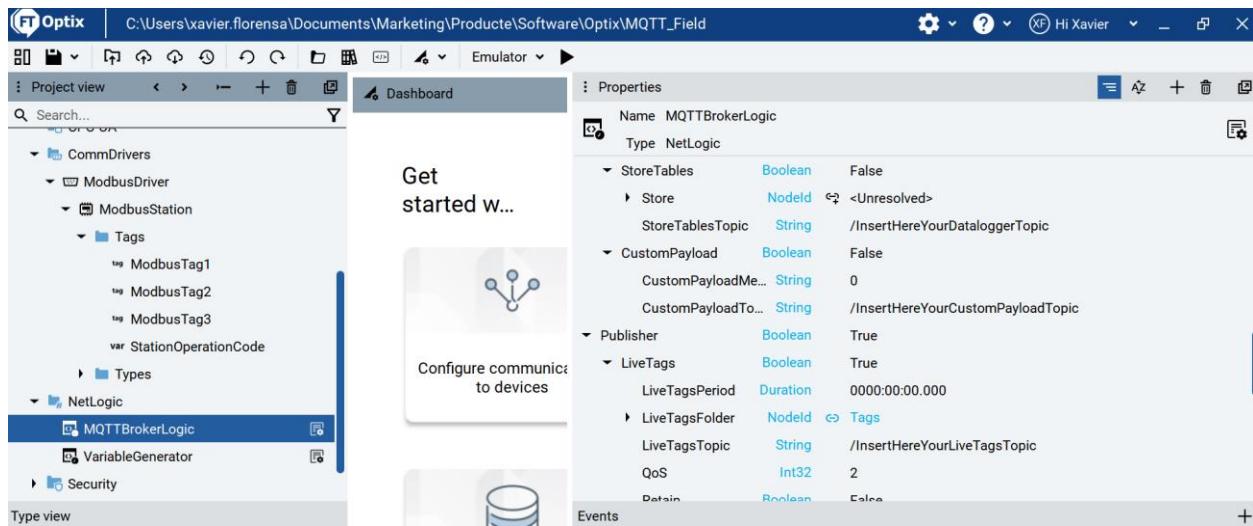
Drag and drop the library to the left over NetLogic



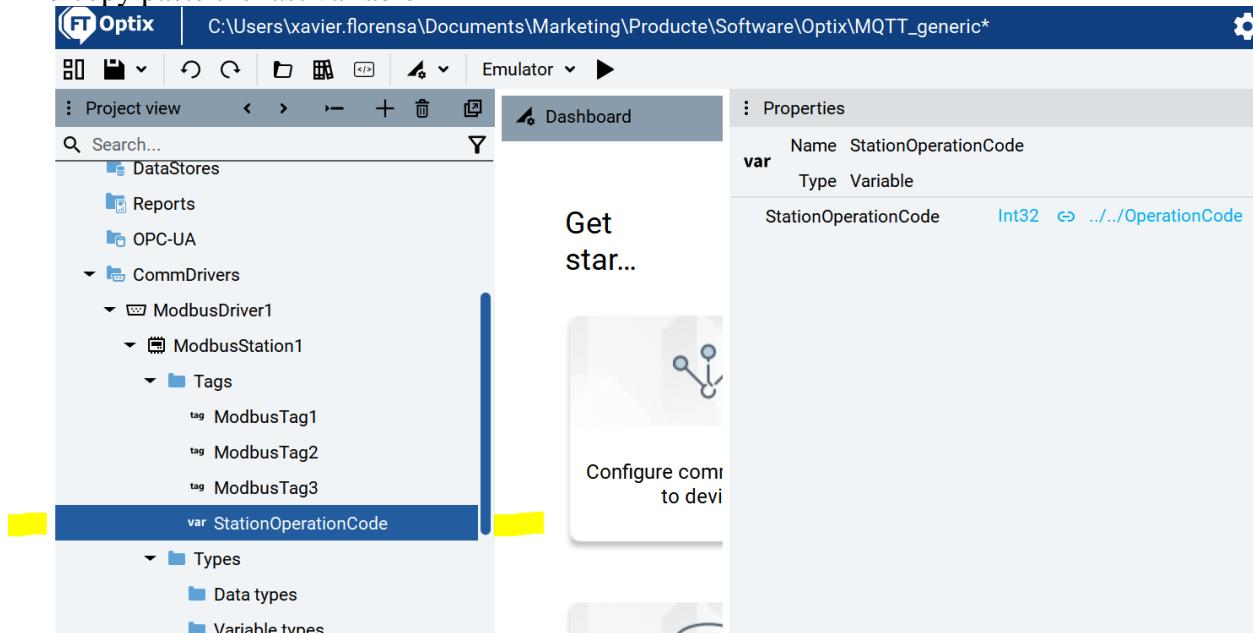
Now you have your MQTT object, but you have to specify where is your live tags folder



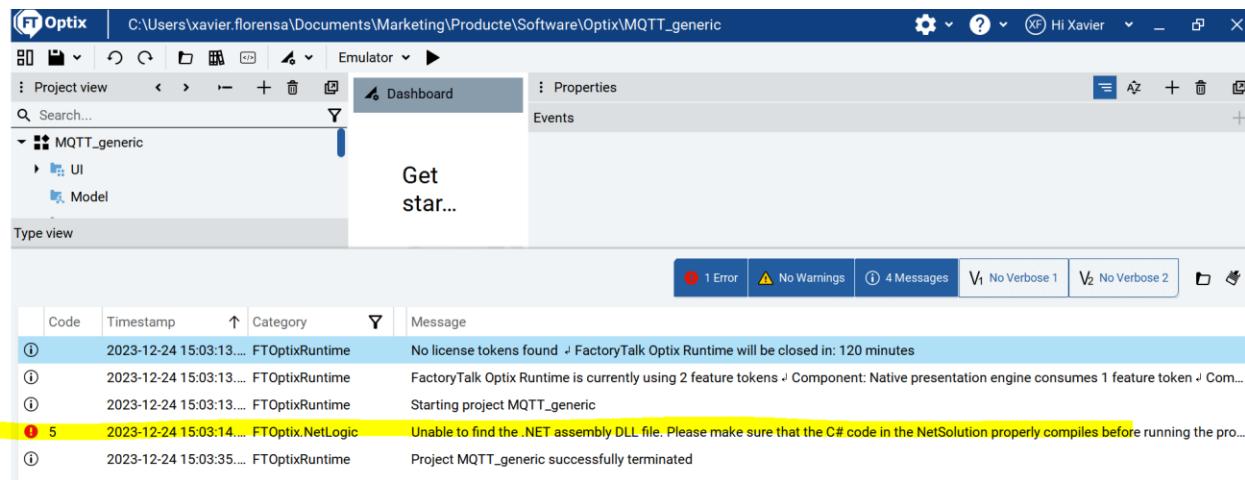
You have to create something like this



So let's adapt our application to the code accordingly  
And copy paste the last variable

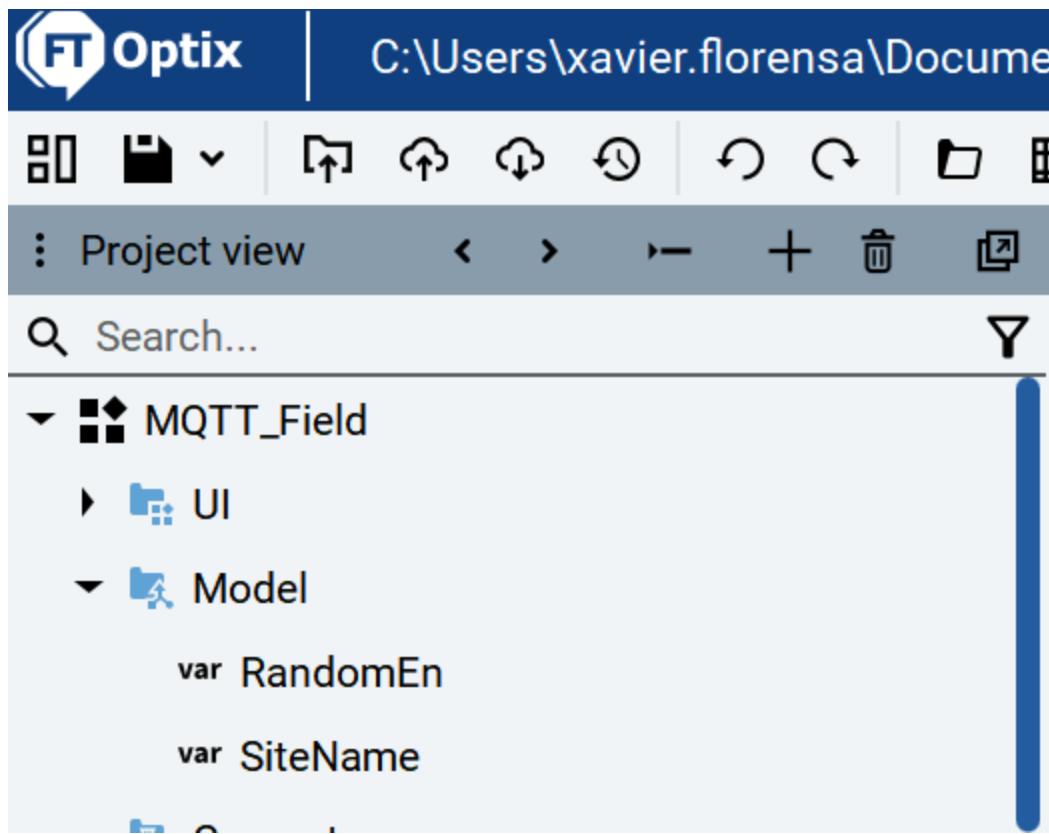


Let's run the code with a Modbus variable simulator



The project is using these elements that we do not have in our project

- ▼ Alarms
  - ▶ AnalogAlarmTBP (type)
  - ▶ AnalogAlarmTBP1
  - ▶ AnalogAlarmTBP2
  - ▶ AnalogAlarmTBP3
- Recipes
- ▼ Loggers
  - DataLogger
  - AlarmsEventLogger
- ▼ DataStores
  - EmbeddedDatabase



So sometimes it's not easy to reuse some code, especially if there is no simple application to extract the code from.

In this case the code has 2800 lines.

Let's use a simpler code with 50 lines

### 23.3. How to use .NET third party libraries in our projects

Let's use a simpler code with 50 lines

MQTT client you can download from here as we saw on chapter [Configuring an application as an MQTT client](#)

<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/creating-projects/iot/mqtt-client/Configure-an-application-as-an-mqtt-client.html>

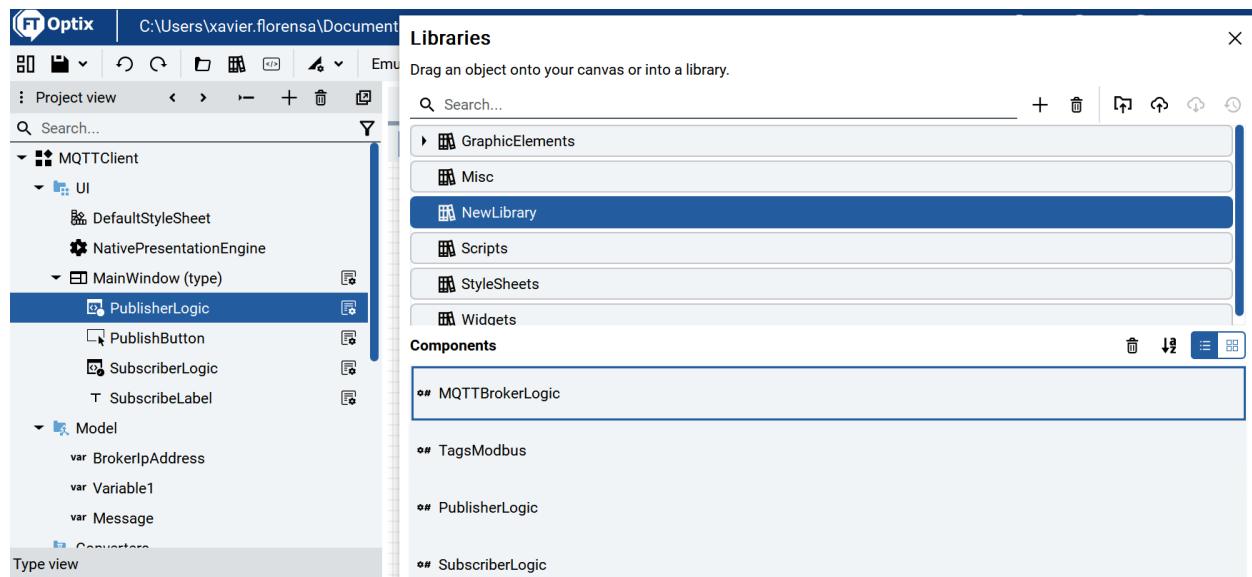


**Tip:** You can download a sample project from: [MQTTClient.zip](#)

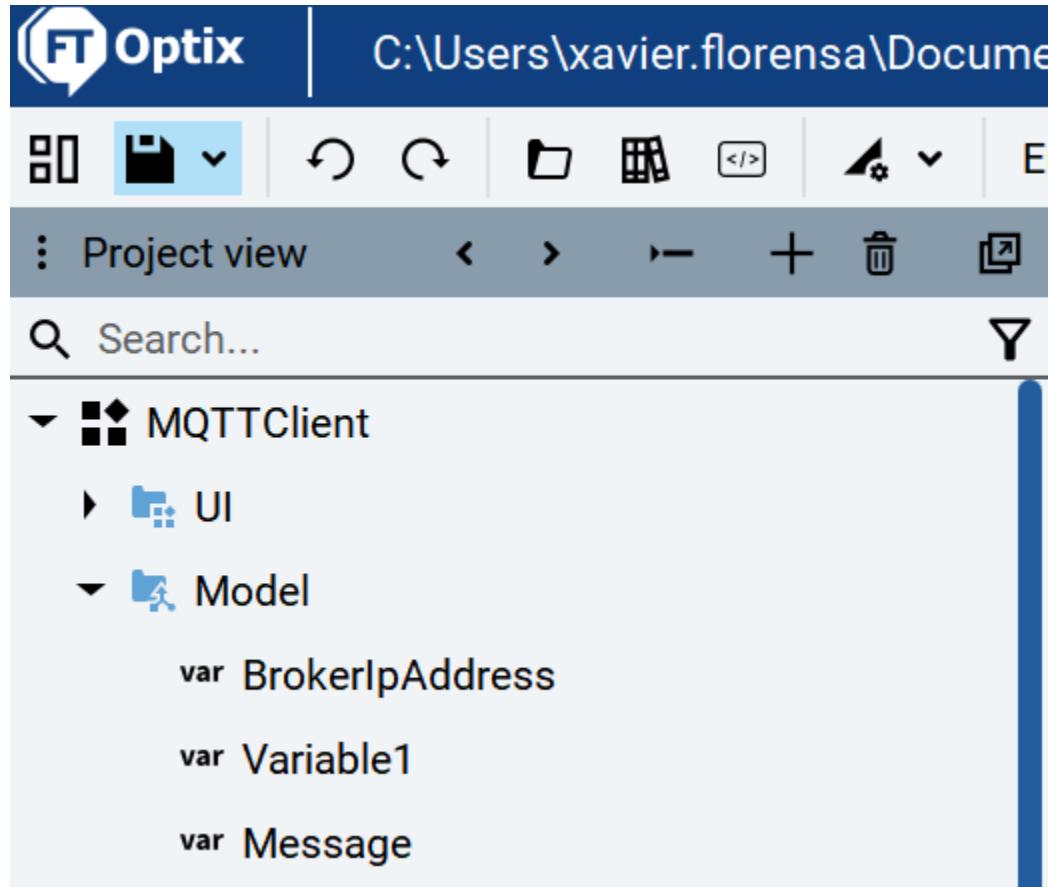
You can download the code from here

[https://github.com/xavierflorensa/MQTTClient\\_FTOptix\\_help](https://github.com/xavierflorensa/MQTTClient_FTOptix_help)

First of all, store the code objects in my libraries with drag and drop.

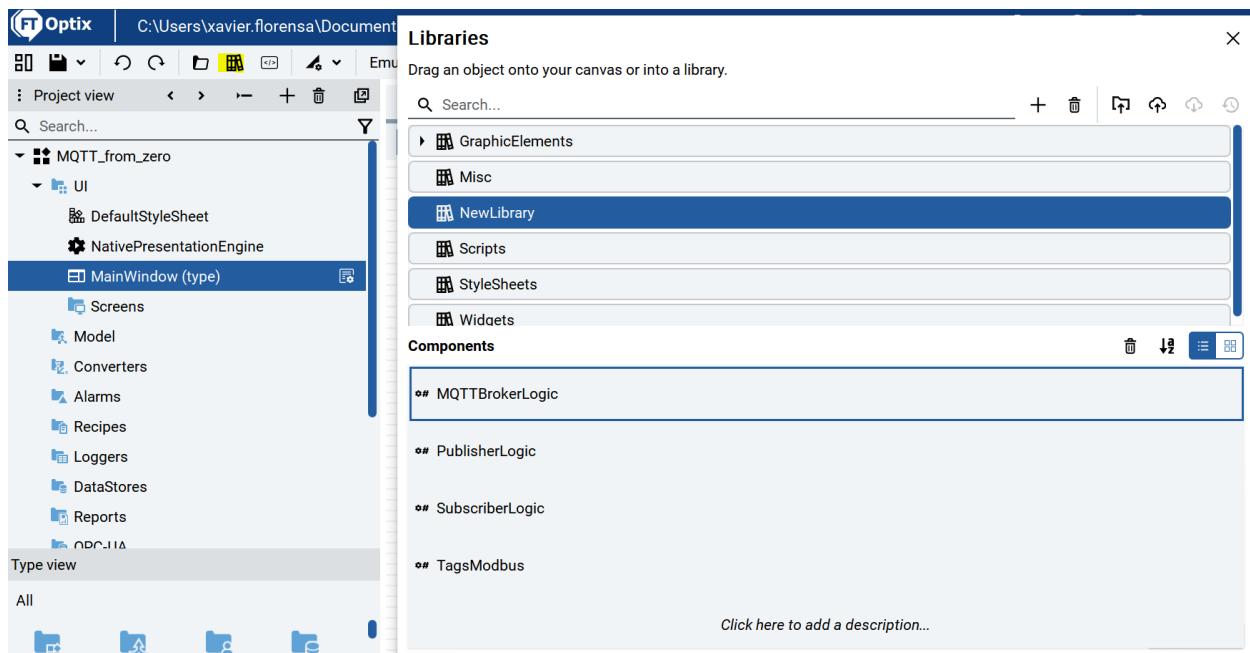


Be aware that this code works with these variables

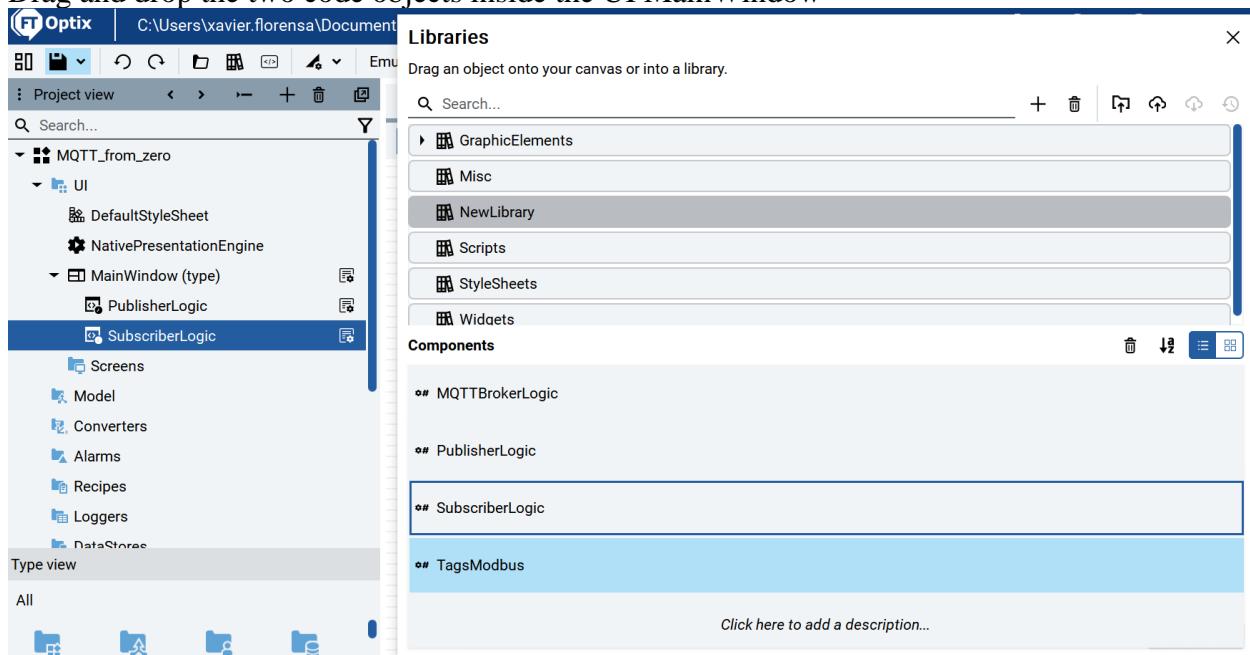


Close the project

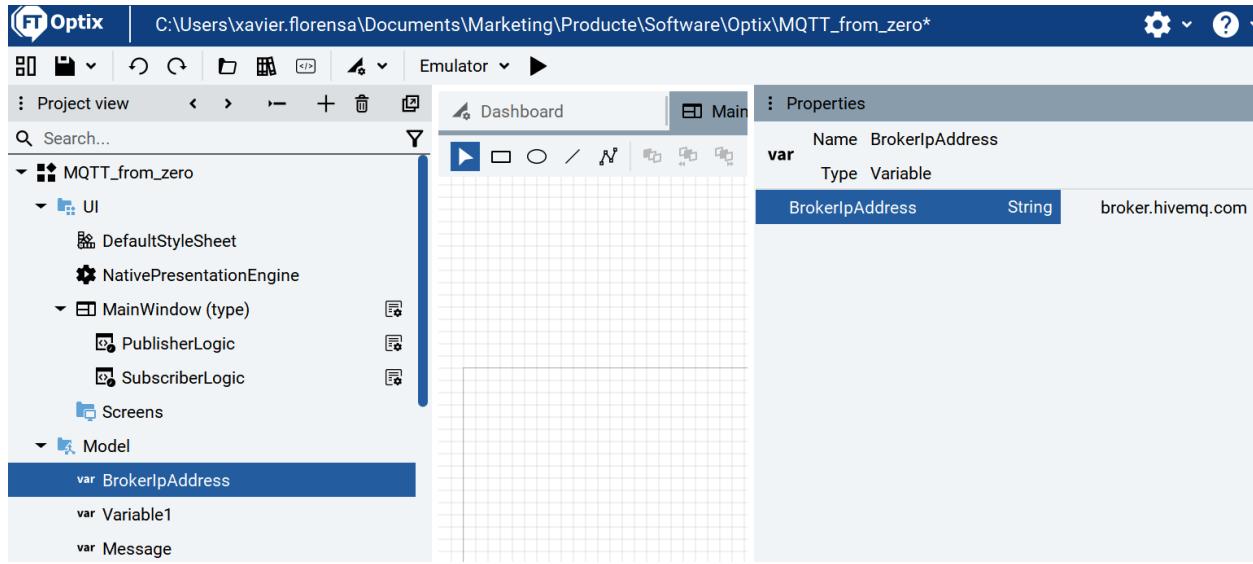
Now create a new project and define the variables  
Open libraries



Drag and drop the two code objects inside the UI MainWindow



Create the variables and initialize the IP address



Now let's put a box to show the subscribed message

We see that the topic is

/my\_topic

As we can see here

```
ushort msgId = subscribeClient.Subscribe(new string[] { "/my_topic" }, // topic
```

But this is not working since we did not installed the right libraries.

Following the guide we have to install a library

<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/creating-projects/iot/mqtt-client/Configure-an-application-as-an-mqtt-client.html>

The screenshot shows a web browser window with the Rockwell Automation logo at the top. The URL in the address bar is <file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/creating-projects/iot/mqtt-client/Configure-an-application-as-an-mqtt-client.html>. The page content includes a sidebar with links like 'Project nodes', 'Dynamic links', etc., and a main area with instructions for developing an MQTT client project, mentioning the 'MqttNet' library and a 'MQTTCient.zip' sample project.

<https://www.nuget.org/packages/MqttNet/>

The screenshot shows the NuGet.org website with the URL <https://www.nuget.org/packages/MqttNet/>. The page title is "MQTTnet 4.3.3.952". It displays the package details, including its version (4.3.3.952), supported frameworks (.NET 5.0, .NET Core 3.1, .NET Standard 1.3, .NET Framework 4.5.2), and download statistics (Total 9.9M, Current version 14.8K, Per day average 4.0K). The "Downloads" section has a "Full stats" link. Below the package details, there's an "About" section with links to Last updated (16 days ago), Project website, Source repository, and MIT license. A "Readme" tab is selected.

This screenshot is identical to the one above, showing the NuGet.org website for the MQTTnet package. It displays the same package details, download statistics, and "About" section with the "Readme" tab selected.

<File:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/netlogic/Manage-third-party-dot-net-libraries.html>

The screenshot shows a web browser window with the address bar set to 'C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/netlogic/Manage-third-parties.html'. The page content is titled 'Third-party .NET libraries' and contains instructions on how to extend C# programming language capabilities using third-party .NET libraries. It includes sections on adding a library to a Visual Studio project or Visual Studio Code project, and a note about supporting only .NET Standard 6.0 destination framework.

<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/netlogic/Add-third-party-library-the-vs-code-project.html>

The screenshot shows a web browser window with the address bar set to 'C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/netlogic/Add-third-party-library-the-vs-code-project.html'. The page content is titled 'Add a third-party library to the Visual Studio Code project' and provides step-by-step instructions for doing so. It includes a tip about using Microsoft Visual Studio 2022 or Microsoft Visual Studio Code as the default code editor.

Copy the downloaded file to the directory

NetSolution			
Share View			
<< Documents >> Marketing > Product > Software > Optix > MQTT_from_zero > ProjectFiles > NetSolution			
Name	Date modified	Type	Size
.vscode	12/27/2023 8:18 AM	File folder	
bin	12/27/2023 8:18 AM	File folder	
obj	12/27/2023 8:18 AM	File folder	
Private	12/27/2023 8:18 AM	File folder	
MQTT_from_zero	12/24/2023 6:11 PM	Code Workspace Sou...	1 KB
MQTT_from_zero	12/27/2023 8:18 AM	C# Project Source File	1 KB
MQTT_from_zero.references	12/27/2023 8:18 AM	REFERENCES File	10 KB
MQTT_from_zero	12/24/2023 6:04 PM	Microsoft Visual Stud...	2 KB
mqttnet.4.3.3.952.nupkg	12/24/2023 6:18 PM	NUPKG File	1,494 KB
PublisherLogic	12/24/2023 6:04 PM	C# Source File	2 KB
SubscriberLogic	12/24/2023 6:16 PM	C# Source File	2 KB

### 23.3.1. Installing Nuget packages in Visual Studio Code

It is much easier to install libraries in Visual Studio Code

[https://www.youtube.com/watch?v=Qy-vwB\\_TEW4](https://www.youtube.com/watch?v=Qy-vwB_TEW4)

Open for instance, subscriberlogic.cs on VS code  
 Be aware that VScode complaints with MqttClient  
 So let's go to Extensions (the lowest icon on the left)

```

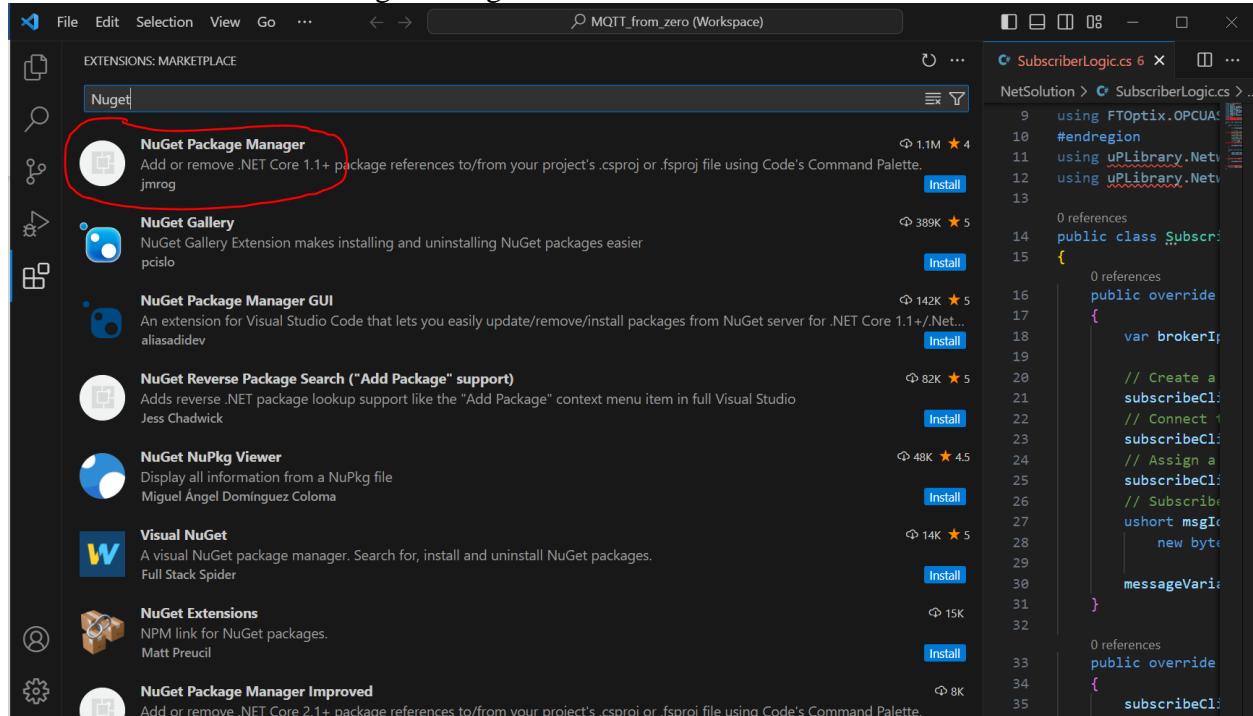
using FTOptix.OPCUA;
#endregion
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;

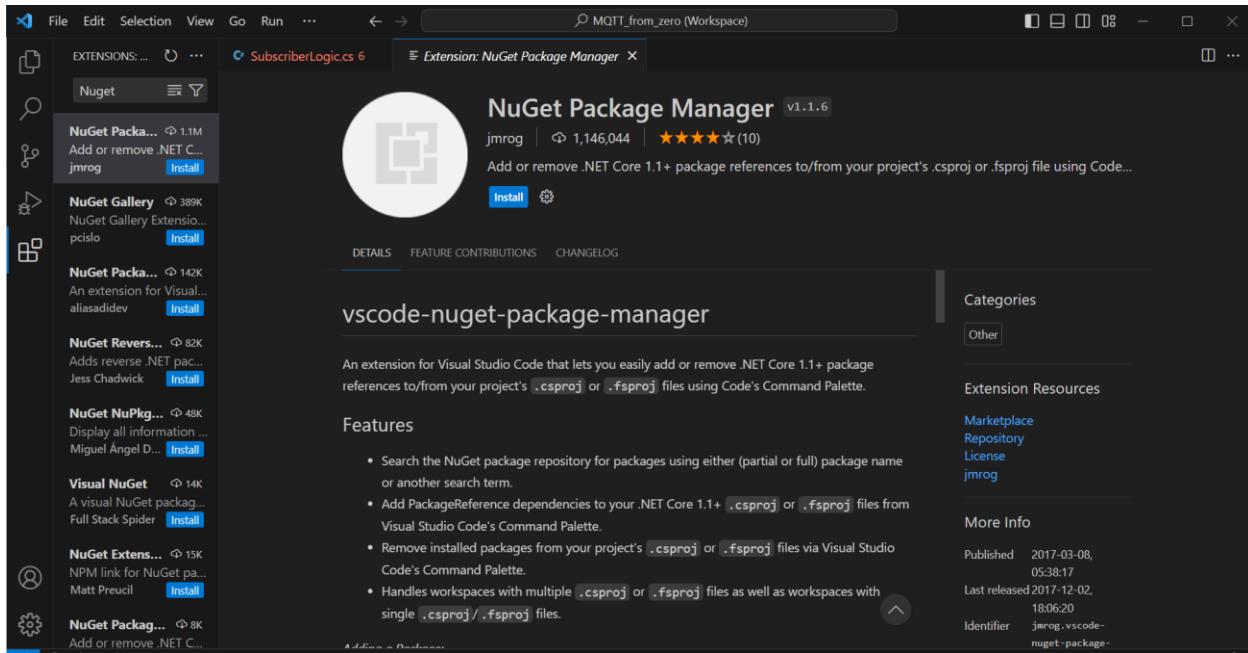
public class SubscriberLogic : BaseNetlogic
{
    public override void Start()
    {
        var brokerIpAddressVariable = Project.Current.GetVariable("Model/BrokerAddress");
        subscribeClient = new MqttClient(brokerIpAddressVariable.Value);
        subscribeClient.Connect("FTOptixSubscribeClient");
        subscribeClient.MqttMsgPublishReceived += SubscribeClientMqttMsgPub;
        subscribeClient.Subscribe(new string[] { "/my_topic" });
        ushort msgId = subscribeClient.Subscribe(new string[] { "/my_topic" },
            new byte[] { MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE });
        messageVariable = Project.Current.GetVariable("Model/Message");
    }
}

public override void Stop()
{
    subscribeClient.Unsubscribe(new string[] { "/my_topic" });
}

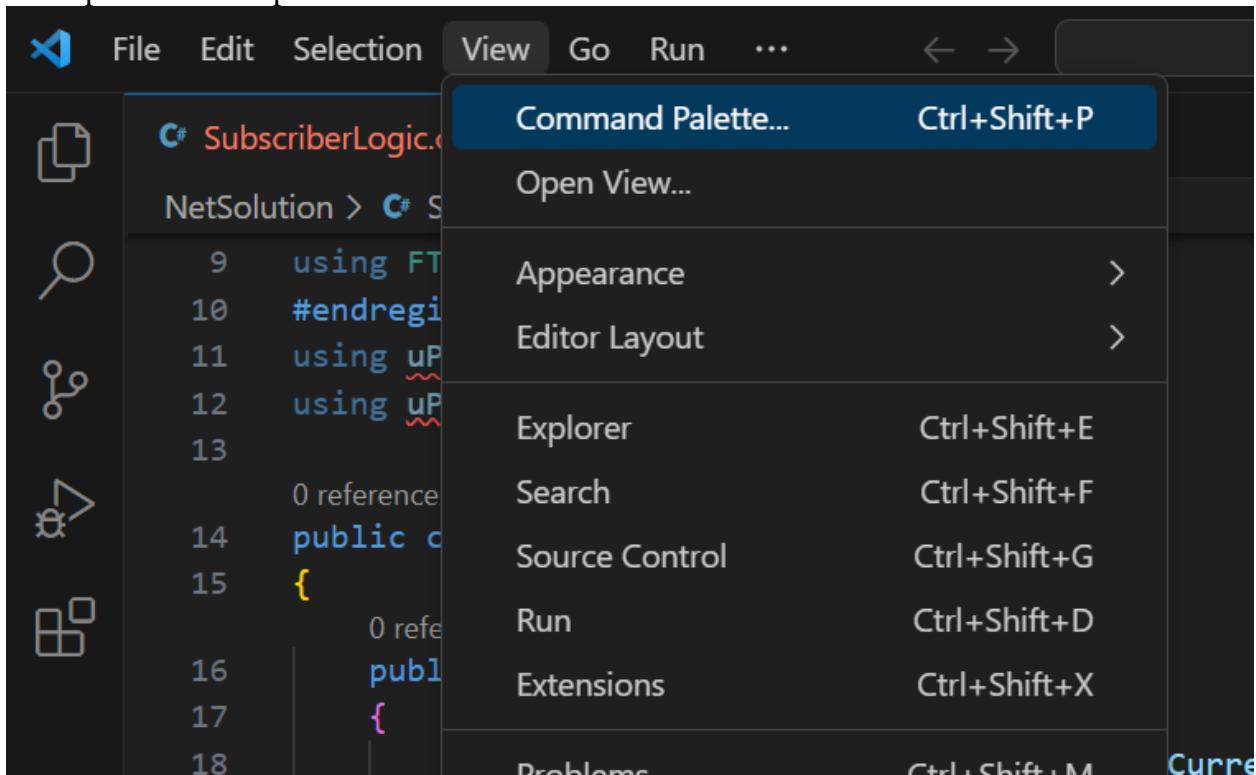
```

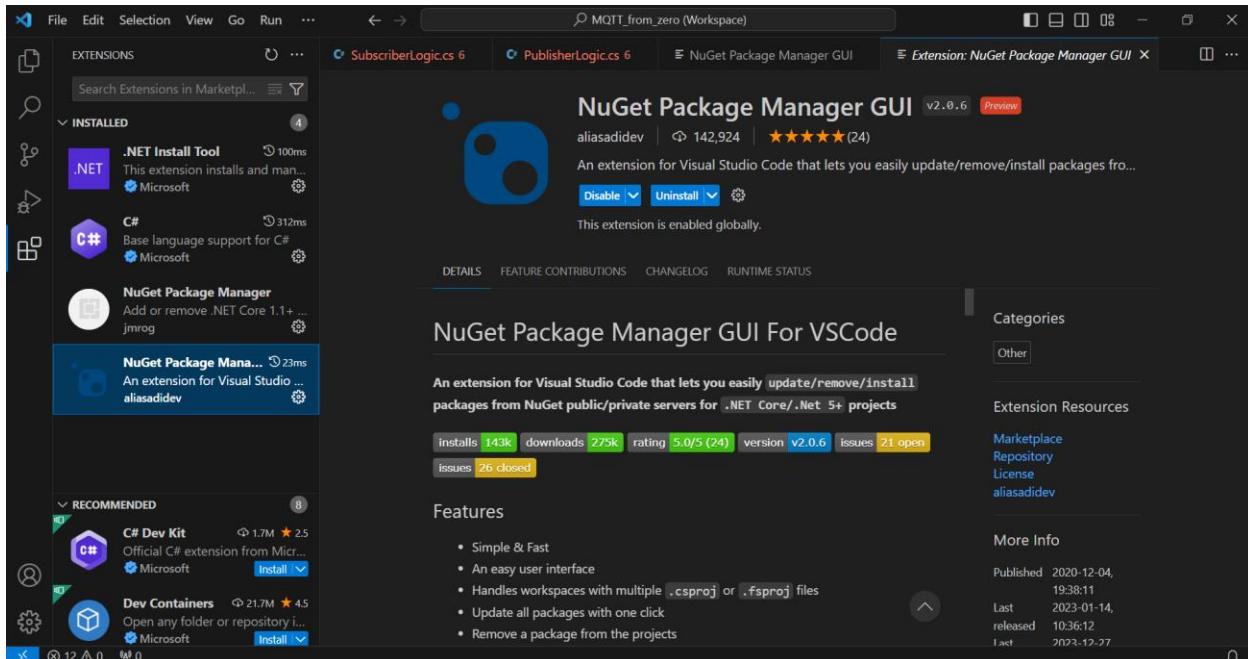
And search for NuGet Package Manager



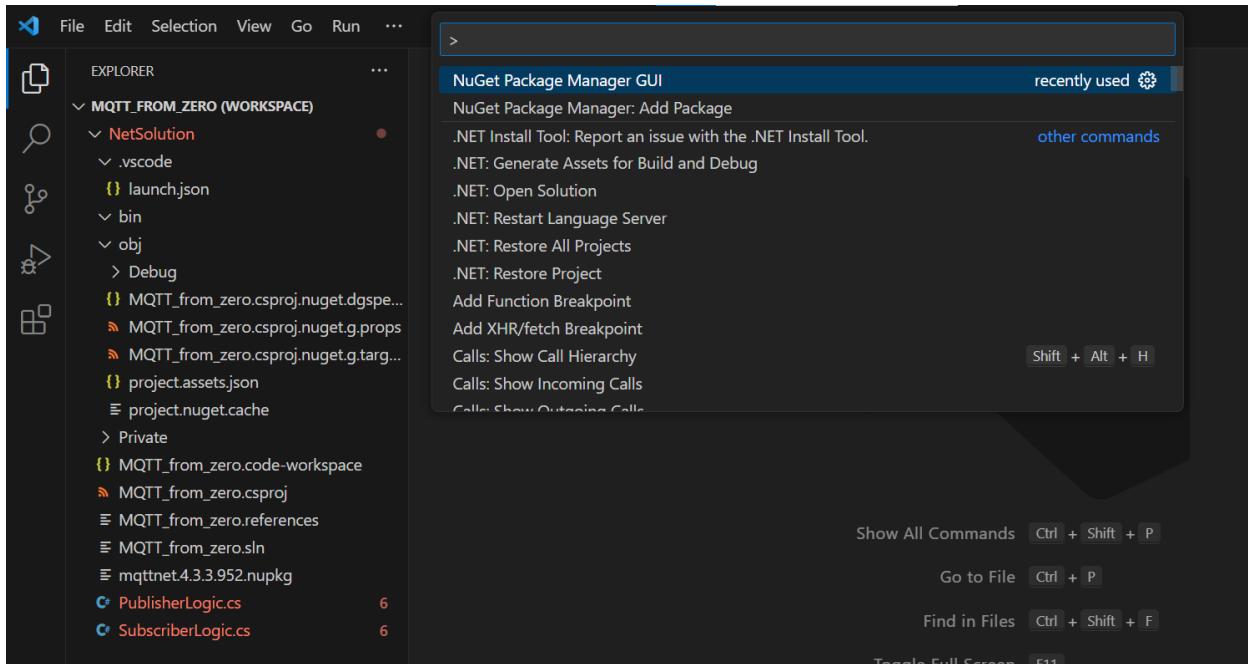


Now close clicking again on the Extensions lowest left icon,  
And open command palette

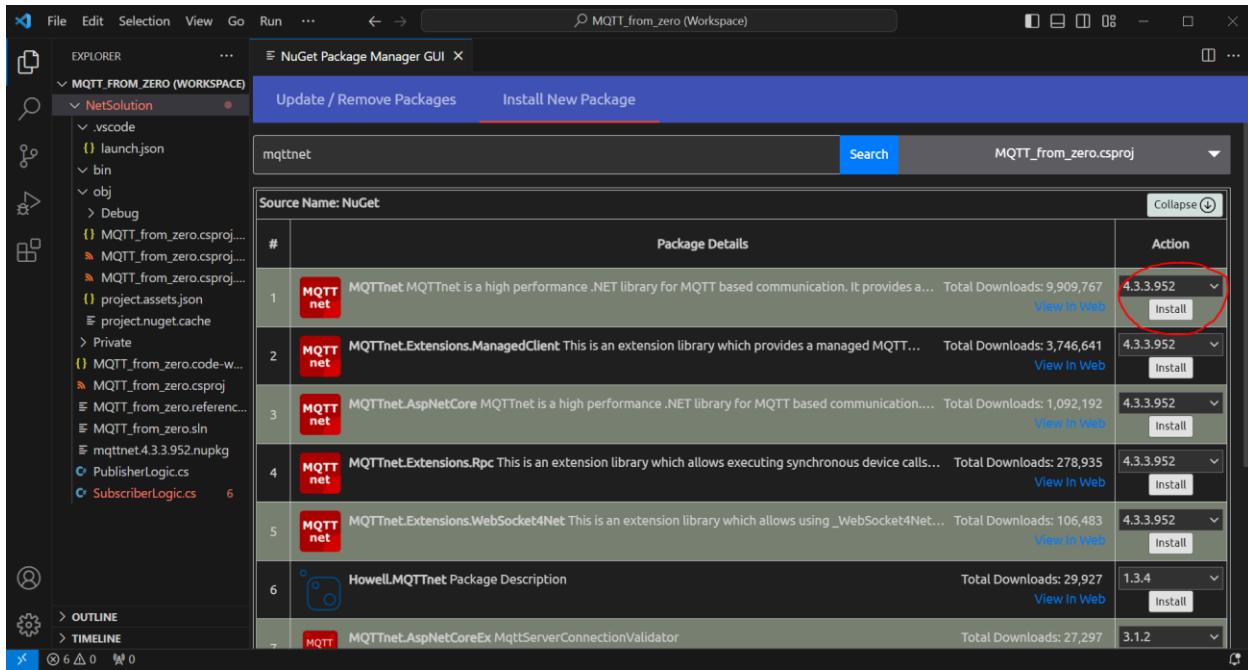




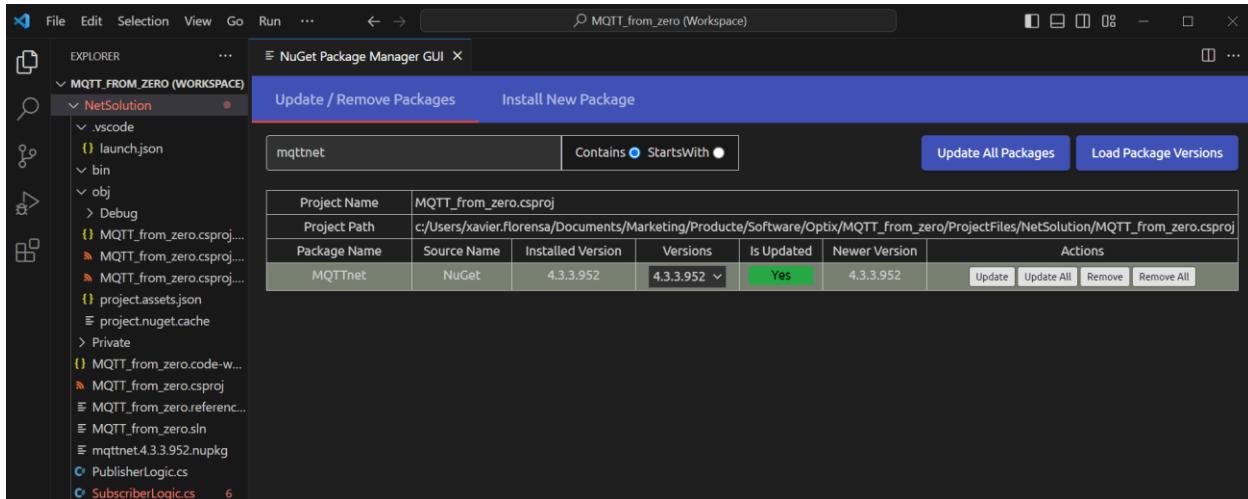
Install such package manager if you did do this step in the past  
Go to View / Command Pallette



Search for the library on the server and click on install



Then go to Upload/Remove Packages  
Proceed with Upload and Update



If we take a look at \*.csproj

```

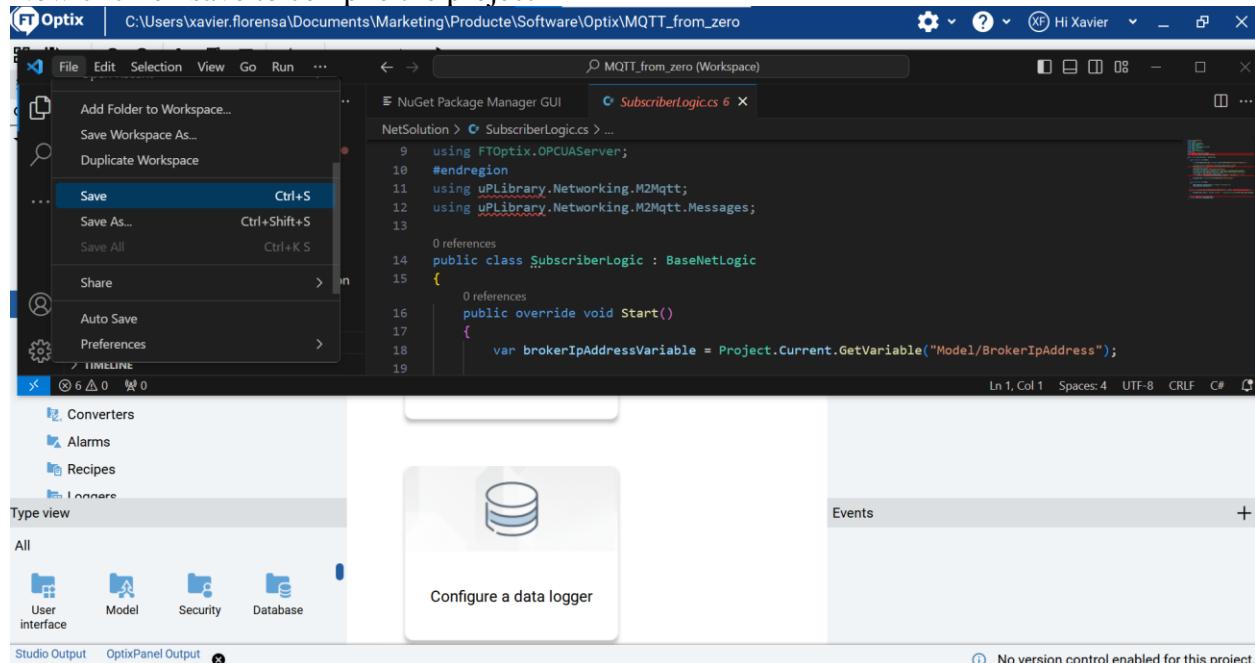
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <AppendTargetFrameworkToOutputPath>false</AppendTargetFrameworkToOutputPath>
    <CopyLocalLockFileAssemblies>true</CopyLocalLockFileAssemblies>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)'=='Debug|AnyCPU'">
    <OutputPath>bin\</OutputPath>
    <NoWarn>1701;1702</NoWarn>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)'=='Release|AnyCPU'">
    <OutputPath>bin\</OutputPath>
    <NoWarn>1701;1702</NoWarn>
  </PropertyGroup>
  <PropertyGroup>
    <ResolveAssemblyWarnOrErrorOnTargetArchitectureMismatch>None</ResolveAssemblyWarnOrErrorOnTargetArchitectureMismatch>
  </PropertyGroup>
  <Import Project="MQTT_from_zero.references" />
  <ItemGroup>
    <PackageReference Include="MQTTnet" Version="4.3.3.952" />
  </ItemGroup>
</Project>

```

We have it installed as it was recommending on the web help

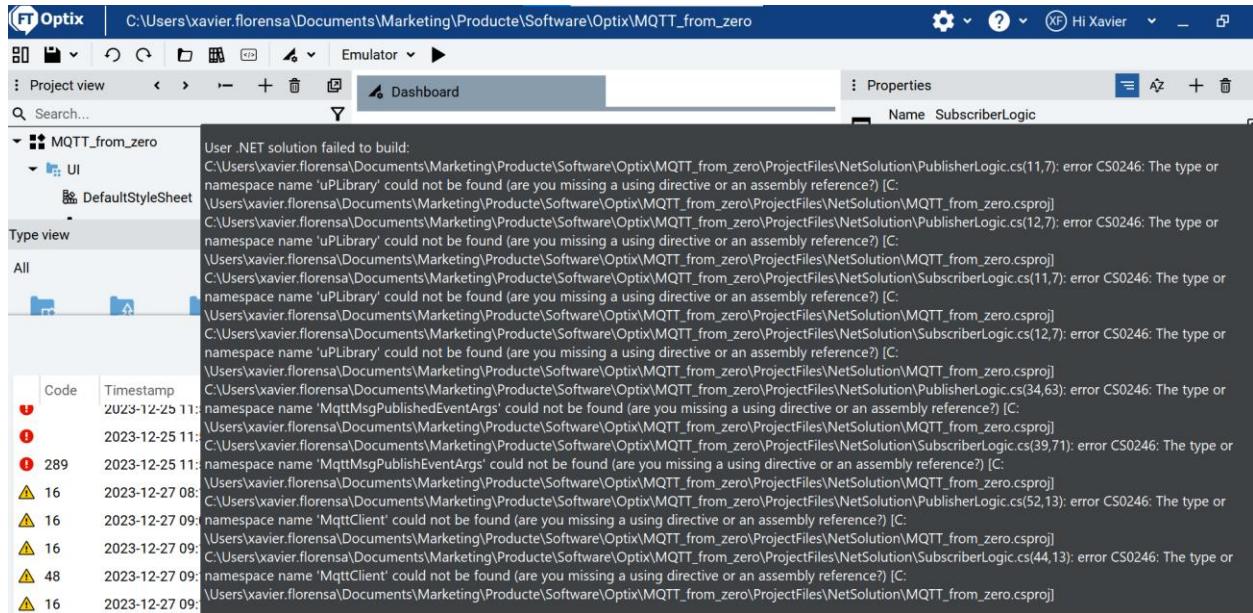
C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/netlogic/Add-third-party-library-the-vs-code-project.html

Now click on save to compile the project



We still have errors,

This is due to the fact that we still need the library M2MQtt.dll



Let's take a look at the example you can download from Help pages

Figure: Application example



C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/creating-projects/iot/mqtt-client/Configure-an-application-as-an-mqtt-client.html

We see that on that example there are two dll files.

To advance in this issue just copy these two DLLs on bin directory (as we see on the examples that you have these dlls that we are missing

ProjectFiles > NetSolution > bin



M2Mqtt.Net.dll



MQTTnet.dll

You also have to install the M2Mqtt NuGet package

Proceed as before when installing the library MQTTnet, install, load and update

Then you will have no errors

Not it works

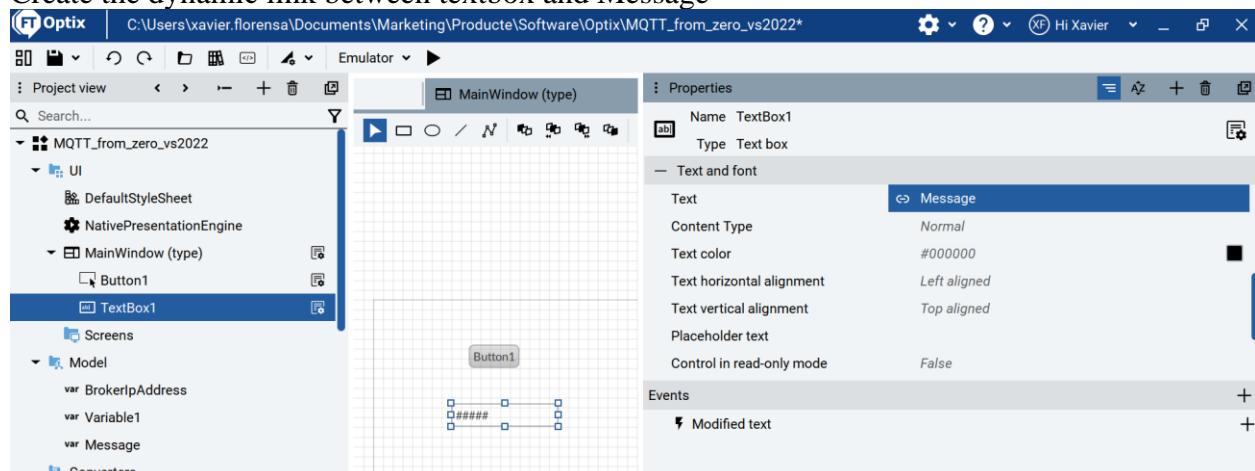
You can also use Visual Studio 2022 to install the .net packages

### 23.3.2. With Visual Studio 2022 community edition

Let's try to install the two Nuget packages with Visual Studio 2022, and then once installed, let's move back to Visual Studio Code.

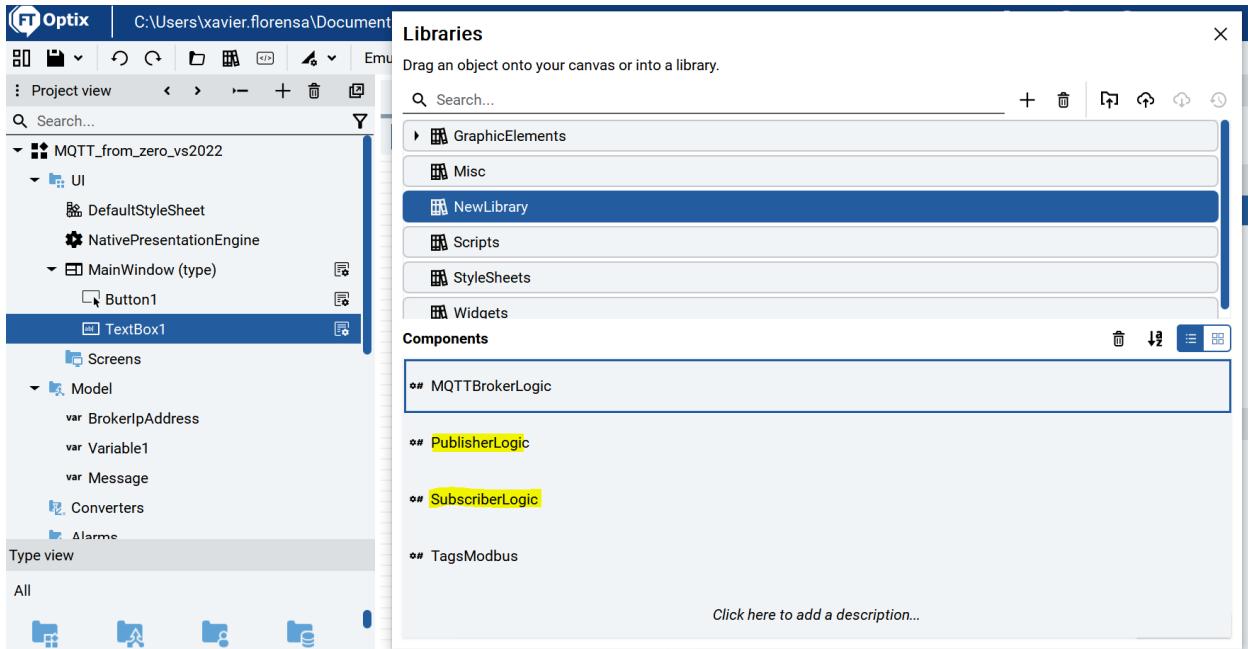
Create an application with FT Optix, with the three variables and the button and label.

Create the dynamic link between textbox and Message



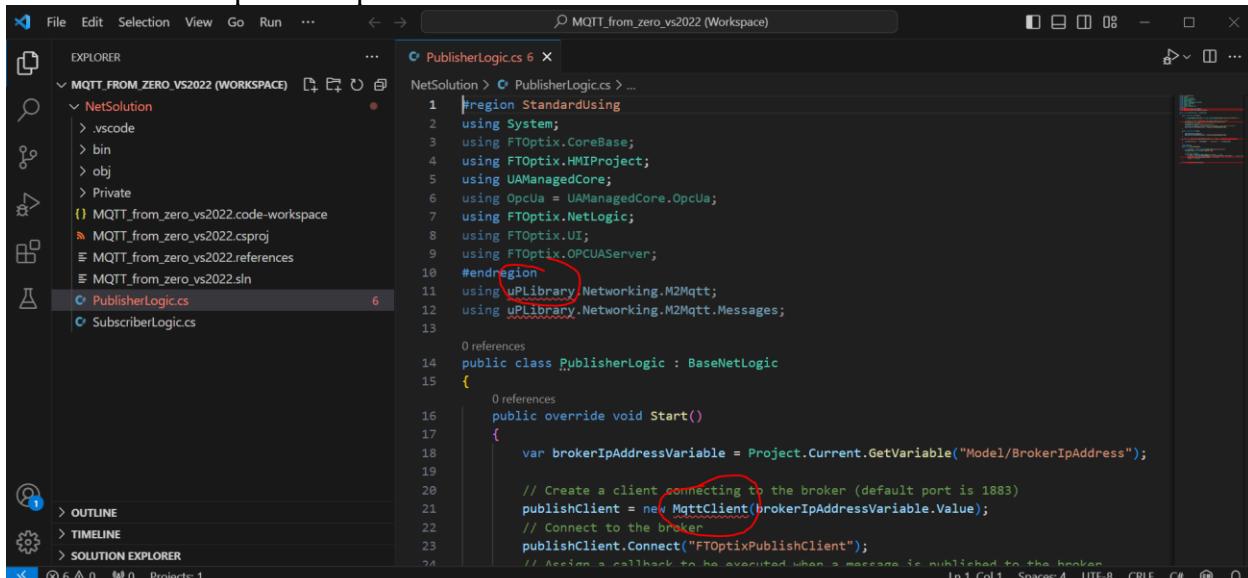
Drag and drop to MainWindow the two libraries that you stored in previous chapters.

PublisherLogic and SubscriberLogic



## Open PublisherLogic

Note that the compiler complains

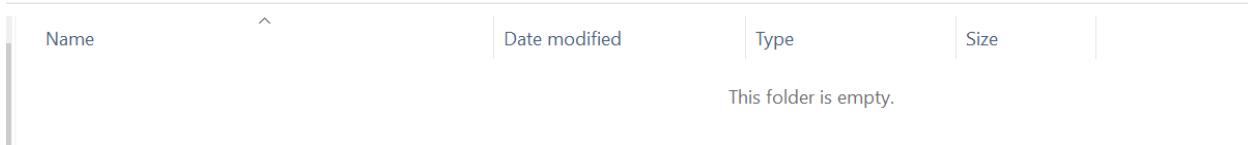


Close VS Code

Now change the editor to Visual Studio 2022, save close and open FTOptix again to admit changes.

Look at this directory, there is nothing since compiler has not succeed building the application.

> This PC > Documents > Marketing > Products > Software > Optix > MQTT\_from\_zero\_vs2022 > ProjectFiles > NetSolution > bin



Click on Publisherlogic. Go to Project/Manage NuGet Packages.

```

1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
10 10
11 11
12 12
13 13
14 14
15 15
16 16
17 17
18 18
19 19
20 20
21 21
22 22
23 23
24 24
25 25
26 26
27 27
28 28
29 29

```

Code snippet from MQTT\_from\_zero\_vs2022.cs:

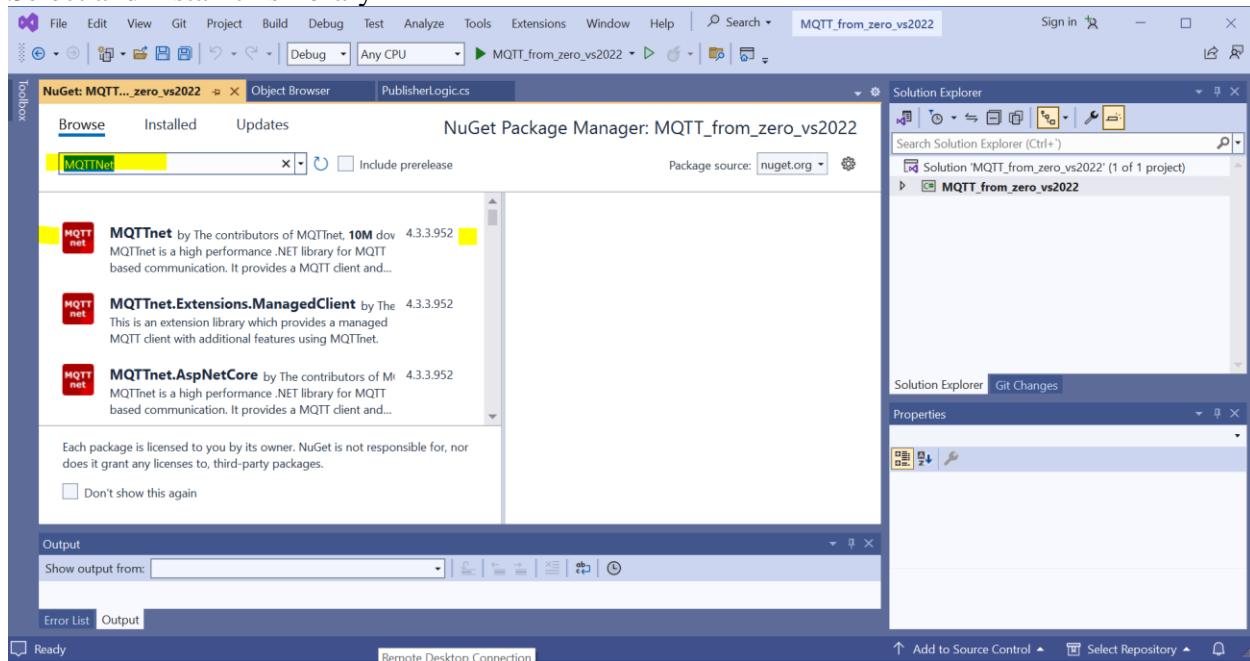
```

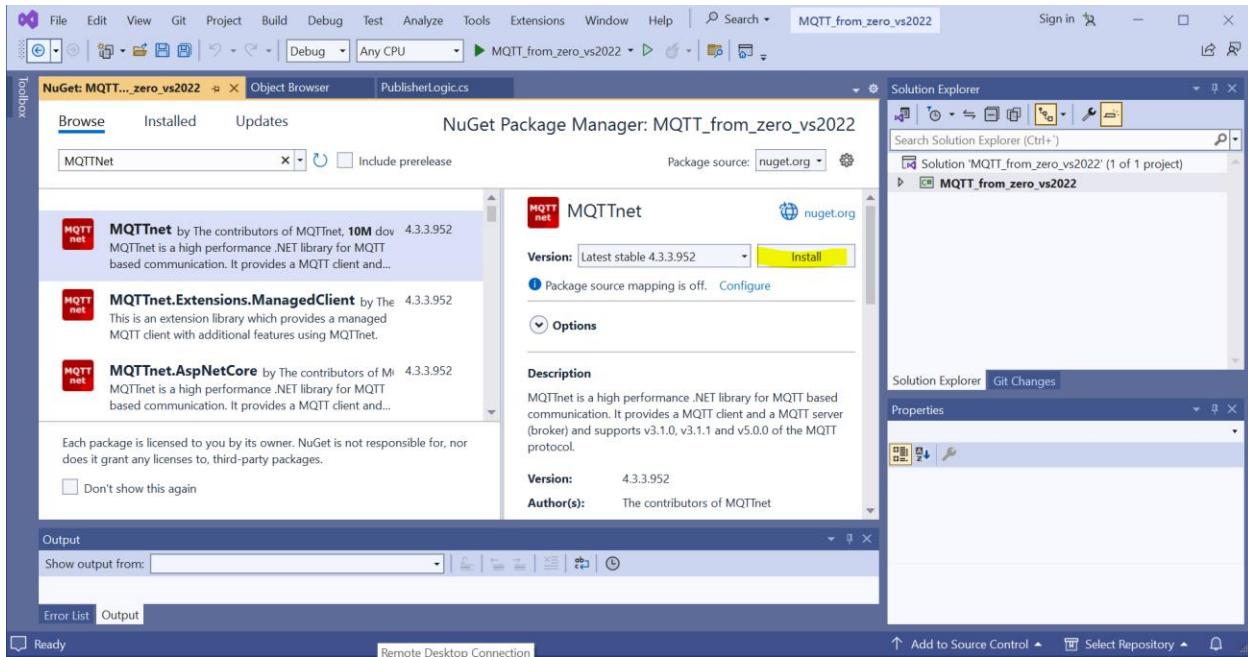
    t.GetVariable("Model/BrokerIpAddress");
    (default port is 1883)
    ssVariable.Value);
};

Message is published to the broker
eventMqttMsgPublished;
}

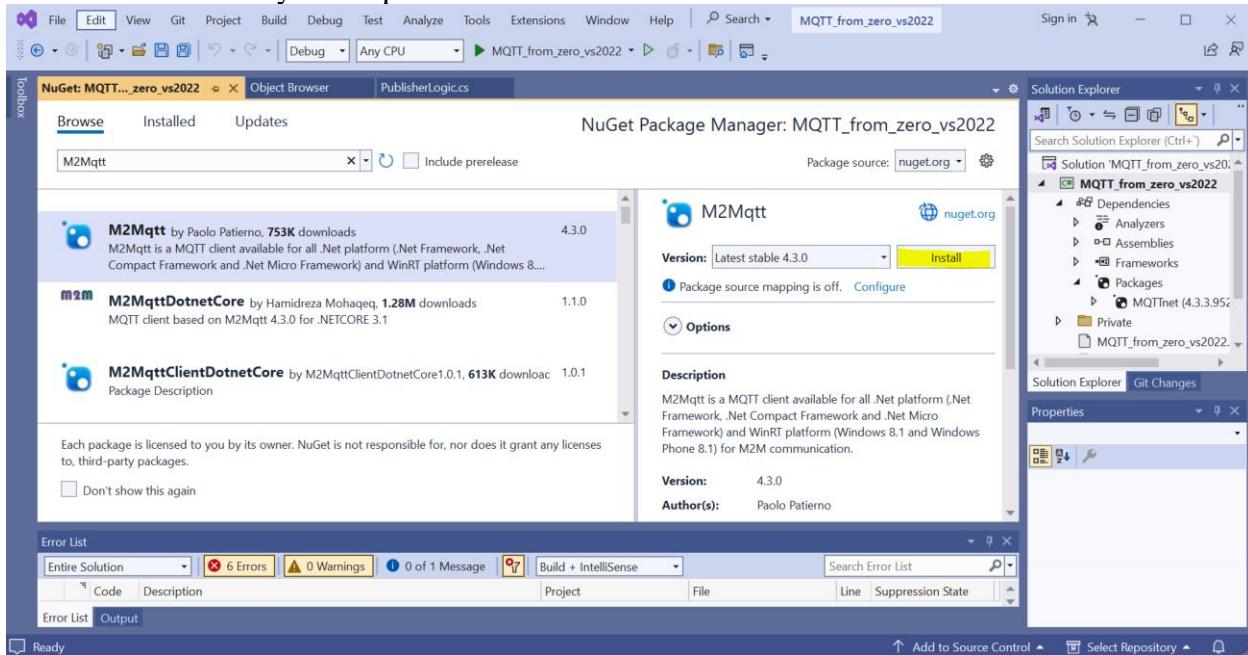
```

### Select and install this library





## Install also this library M2Mqtt



Now we have the two packages

NuGet: MQTT\_from\_zero\_vs2022 Object Browser PublisherLogic.cs Git Changes

```

1  StandardUsing;
11  using uPLibrary.Networking.M2Mqtt;
12  using uPLibrary.Networking.M2Mqtt.Messages;
13
14  public class PublisherLogic : BaseNetLogic
15  {
16      public override void Start()
17      {
18          var brokerIpAddressVariable = Project.Current.GetVariable("Model/BrokerIpAddress");
19
20          // Create a client connecting to the broker (default port is 1883)
21          publishClient = new MqttClient(brokerIpAddressVariable.Value);
22          // Connect to the broker
23          publishClient.Connect("FTOptixPublishClient");
24          // Assign a callback to be executed when a message is published to the broker
25          publishClient.MqttMsgPublished += PublishClientMqttMsgPublished;
26      }
27
28      public override void Stop()
29      {
30          publishClient.Disconnect();
31          publishClient.MqttMsgPublished -= PublishClientMqttMsgPublished;
32      }
33
34  private void PublishClientMqttMsgPublished(object sender, MqttMsgPublishedEventArgs e)
35  {
36      Log.Info("Message " + e.MessageId + " - published = " + e.IsPublished);
37  }
38

```

No issues found

Error List

Entire Solution 0 Errors 1 Warning 0 of 2 Messages Build + IntelliSense

Code Description

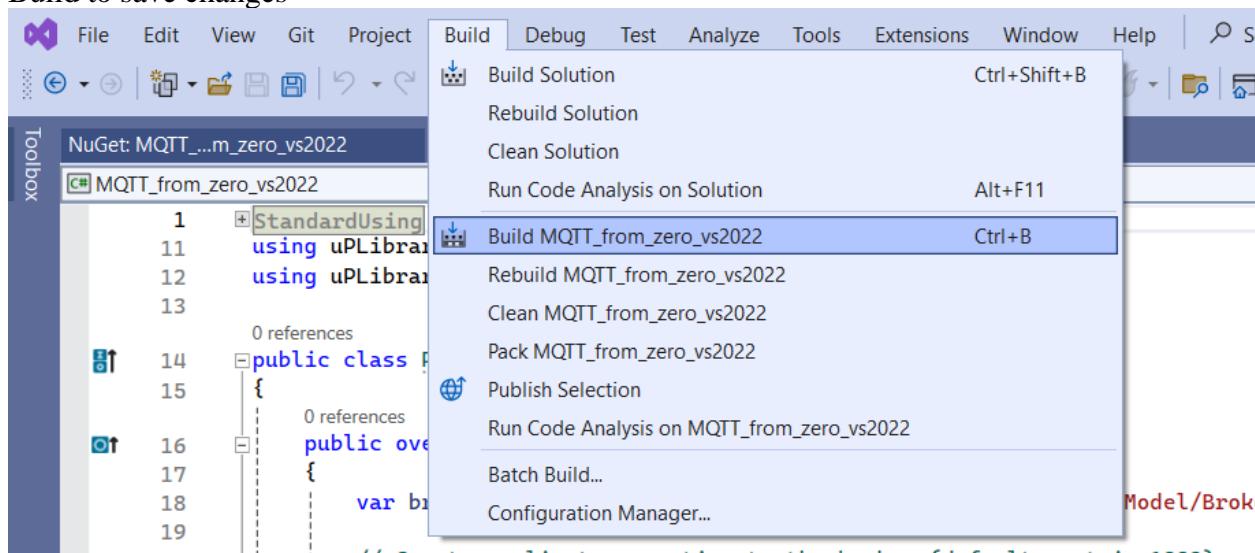
Project File Line Suppression State

Error List Output

This item does not support previewing

Be aware that there are no errors on the code like before.

Build to save changes



Close VS 2022

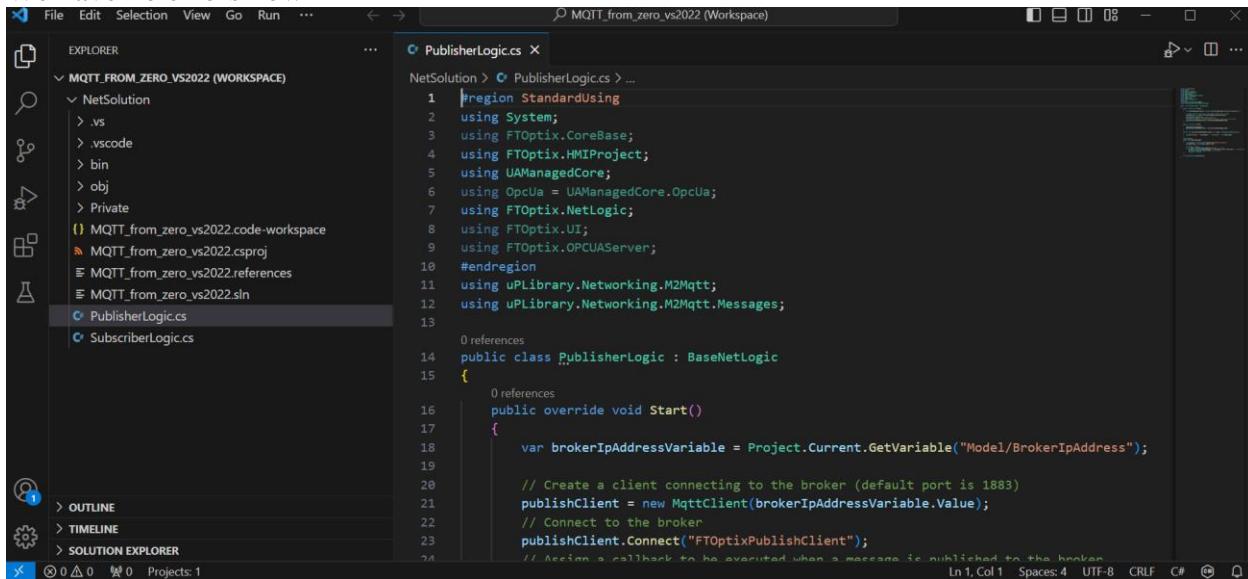
Change the editor on FT Optix to VS Code

Take a look at this directory

> This PC > Documents > Marketing > Product > Software > Optix > MQTT\_from\_zero\_vs2022 > ProjectFiles > NetSolution > bin

Name	Date modified	Type	Size
M2Mqtt.Net.dll	12/6/2015 4:10 PM	Application extension	51 KB
MQTT_from_zero_vs2022.deps	1/7/2024 6:25 PM	JSON Source File	2 KB
MQTT_from_zero_vs2022.dll	1/7/2024 6:25 PM	Application extension	7 KB
MQTT_from_zero_vs2022	1/7/2024 6:25 PM	PDB File	13 KB
MQTTnet.dll	12/8/2023 6:10 PM	Application extension	325 KB

Open FT Optix  
Open Publisherlogic  
We have no errors now



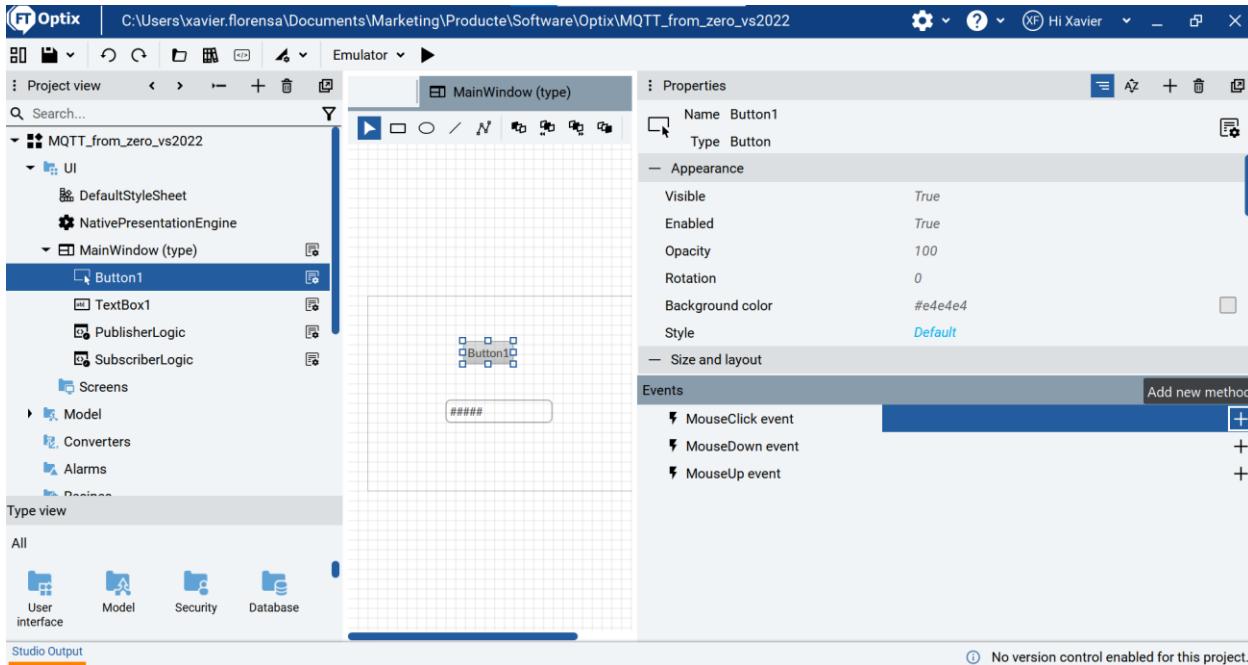
The screenshot shows the Microsoft Visual Studio interface with the title bar "MQTT\_from\_zero\_vs2022 (Workspace)". The left sidebar displays the "EXPLORER" view with a workspace named "MQTT\_FROM\_ZERO\_VS2022 (WORKSPACE)" containing files like ".vs", ".vscode", "bin", "obj", "Private", "MQTT\_from\_zero\_vs2022.code-workspace", "MQTT\_from\_zero\_vs2022.csproj", "MQTT\_from\_zero\_vs2022.references", "MQTT\_from\_zero\_vs2022.sln", "PublisherLogic.cs", and "SubscriberLogic.cs". The main area shows the "PublisherLogic.cs" file with the following code:

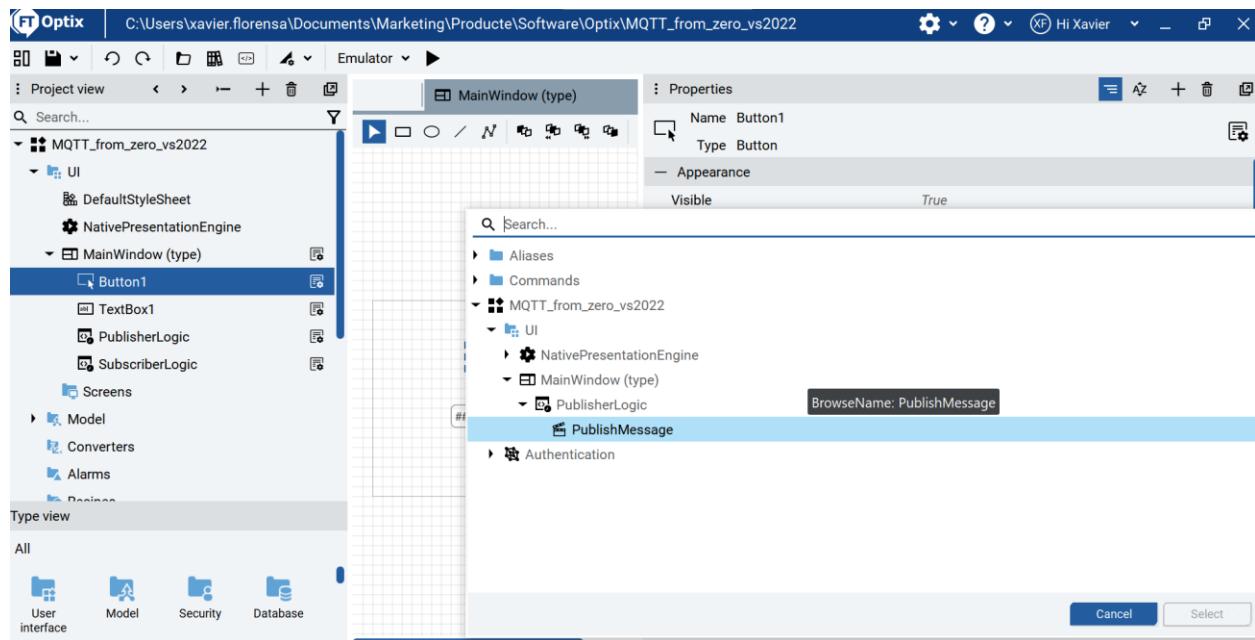
```
1  #region StandardUsing
2  using System;
3  using FTOptix.CoreBase;
4  using FTOptix.HMIProject;
5  using UAManagedCore;
6  using OpcUa = UAManagedCore.OpcUa;
7  using FTOptix.NetLogic;
8  using FTOptix.UI;
9  using FTOptix.OPCUAServer;
10 #endregion
11 using uPLibrary.Networking.M2Mqtt;
12 using uPLibrary.Networking.M2Mqtt.Messages;
13
14 public class PublisherLogic : BaseNetLogic
15 {
16     public override void Start()
17     {
18         var brokerIpAddressVariable = Project.Current.GetVariable("Model/BrokerIpAddress");
19
20         // Create a client connecting to the broker (default port is 1883)
21         publishClient = new MqttClient(brokerIpAddressVariable.Value);
22         // Connect to the broker
23         publishClient.Connect("FTOptixPublishClient");
24         // Action is a callback to be executed when a message is published to the broker
25     }
26 }
```

At the bottom right of the code editor, status information includes "Ln1, Col1", "Spaces:4", "UTF-8", "CRLF", "C#", and a bell icon.

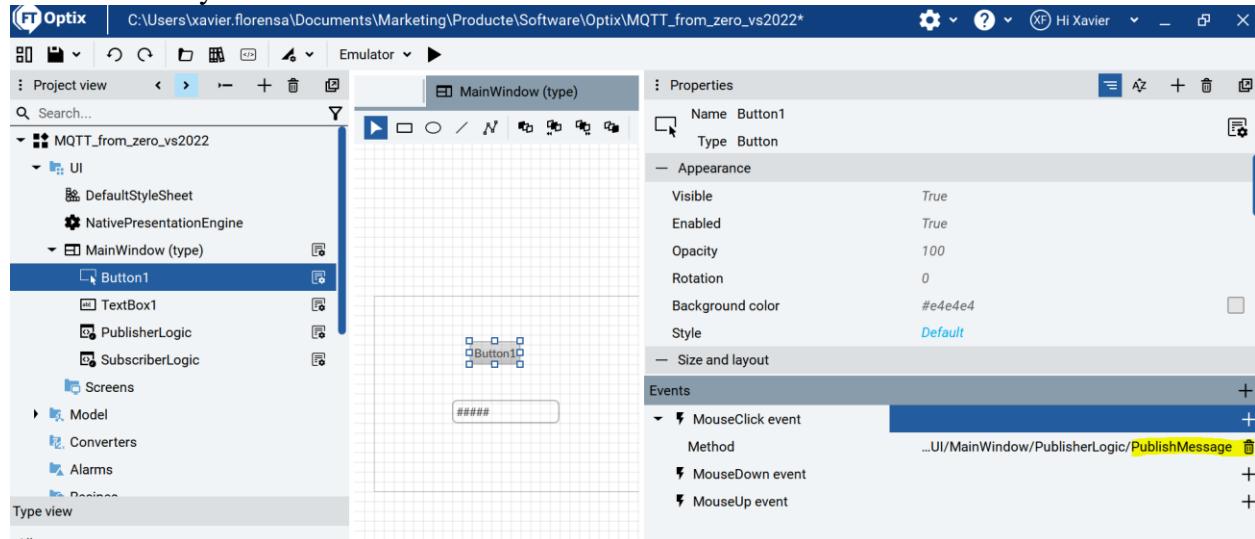
### 23.3.3. Complete your MQTTnet application

Let's build the mouseclick action





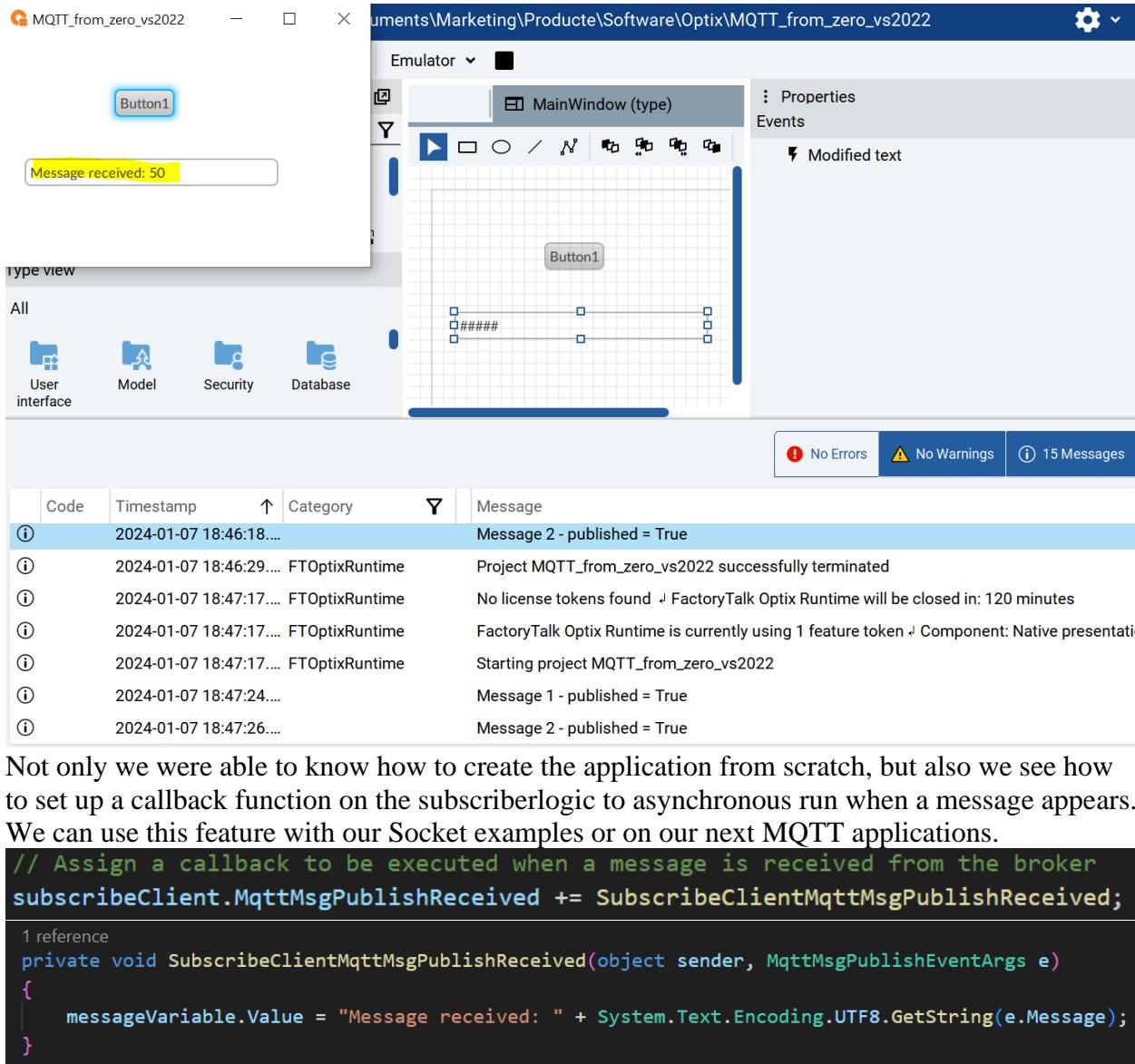
We see the only method declared



We can test the program

It works properly

Each time we click on the button, a random number is published and automatically received on the textbox thanks to the subscribe logic.



Not only we were able to know how to create the application from scratch, but also we see how to set up a callback function on the subscriber logic to asynchronous run when a message appears. We can use this feature with our Socket examples or on our next MQTT applications.

```
// Assign a callback to be executed when a message is received from the broker
subscribeClient.MqttMsgPublishReceived += SubscribeClientMqttMsgPublishReceived;

1 reference
private void SubscribeClientMqttMsgPublishReceived(object sender, MqttMsgPublishEventArgs e)
{
    messageVariable.Value = "Message received: " + System.Text.Encoding.UTF8.GetString(e.Message);
}
```

## 24. First steps with OptixPanel

Standard 12.1 inches

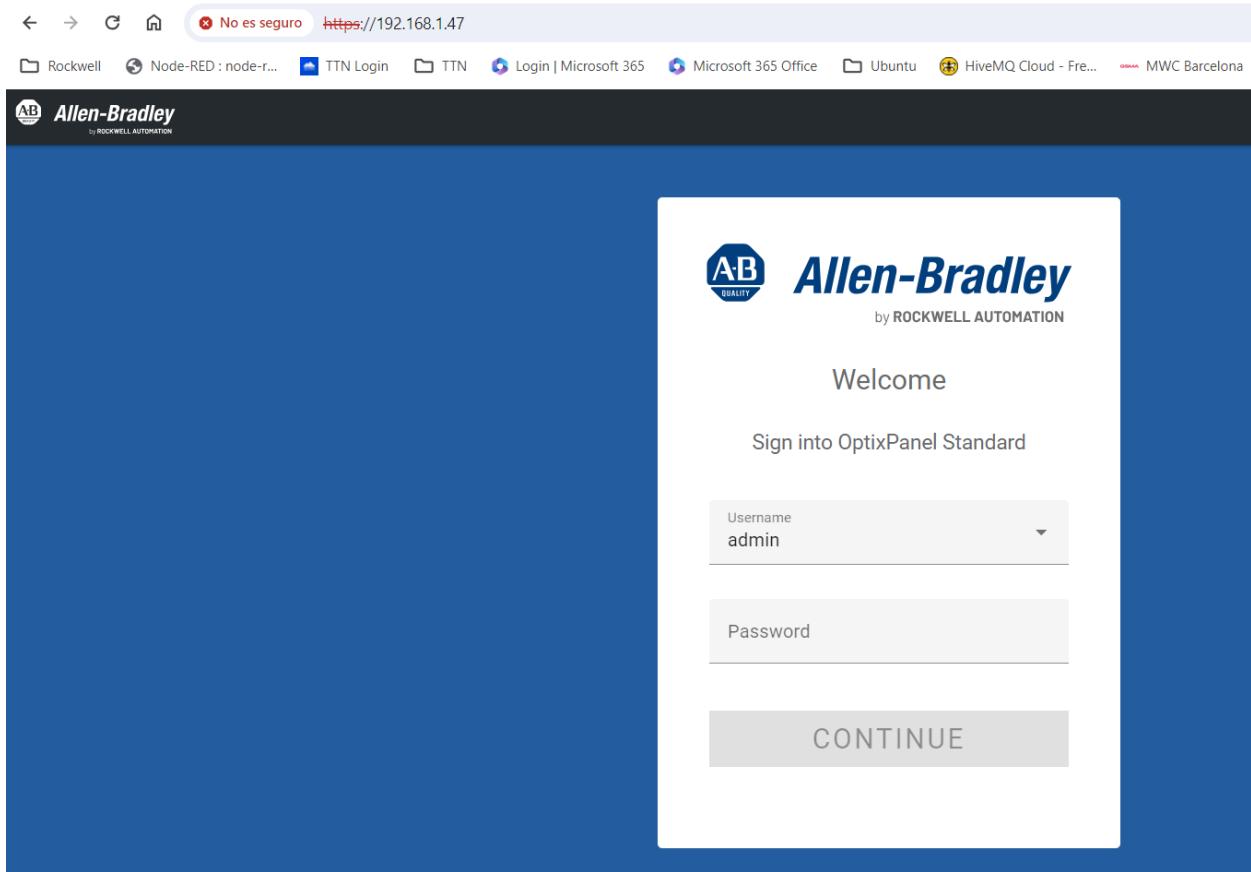
2800S-121FM-N1S

Ethernet Ports

WAN Eth 2 MAC 5C:88:16:FC:32:71 DHCP

LAN Eth1 MAC 5C:88:16:FC:32:70 Fixed IP 192.168.1.101/255.255.255.0

Accessing configuration



You have to introduce a password

First time is

User:admin

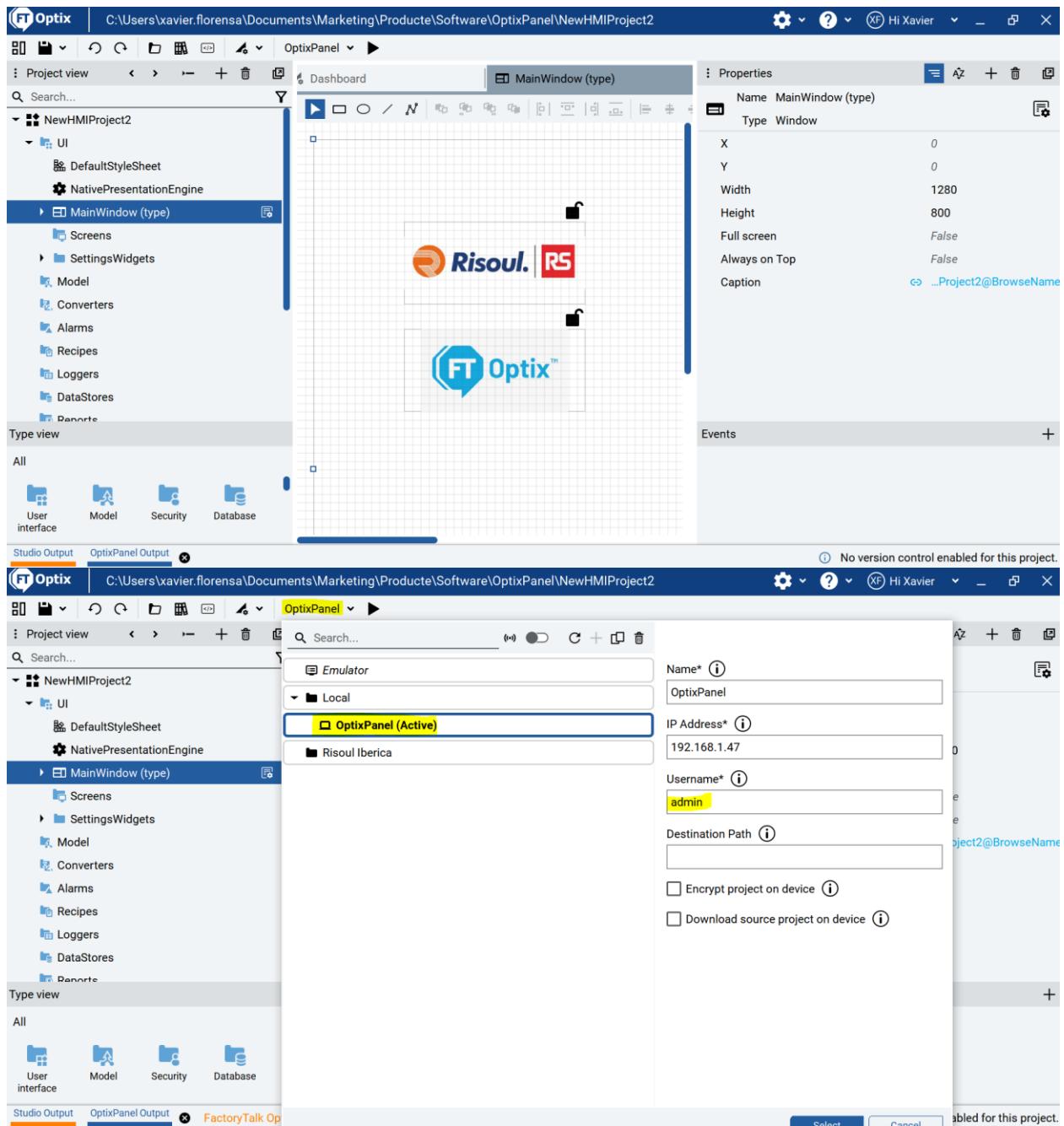
Password:admin

Let's change to Risoul10

A message with:

“No FactoryTalk Optix application found, please upload one using FactoryTalk Optix Studio”

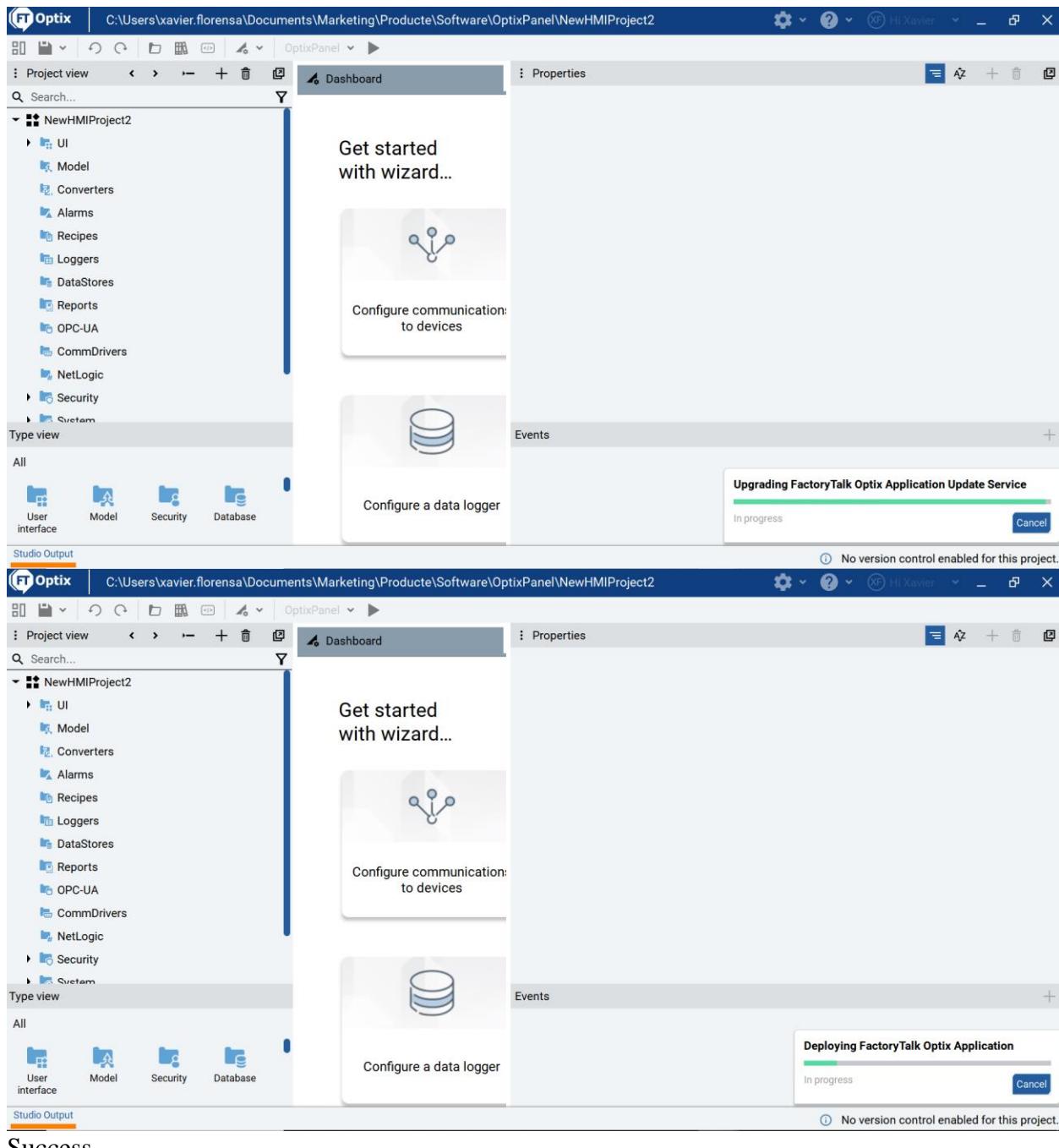
Let's create a new application and select the destination different from Emulator but OptixPanel

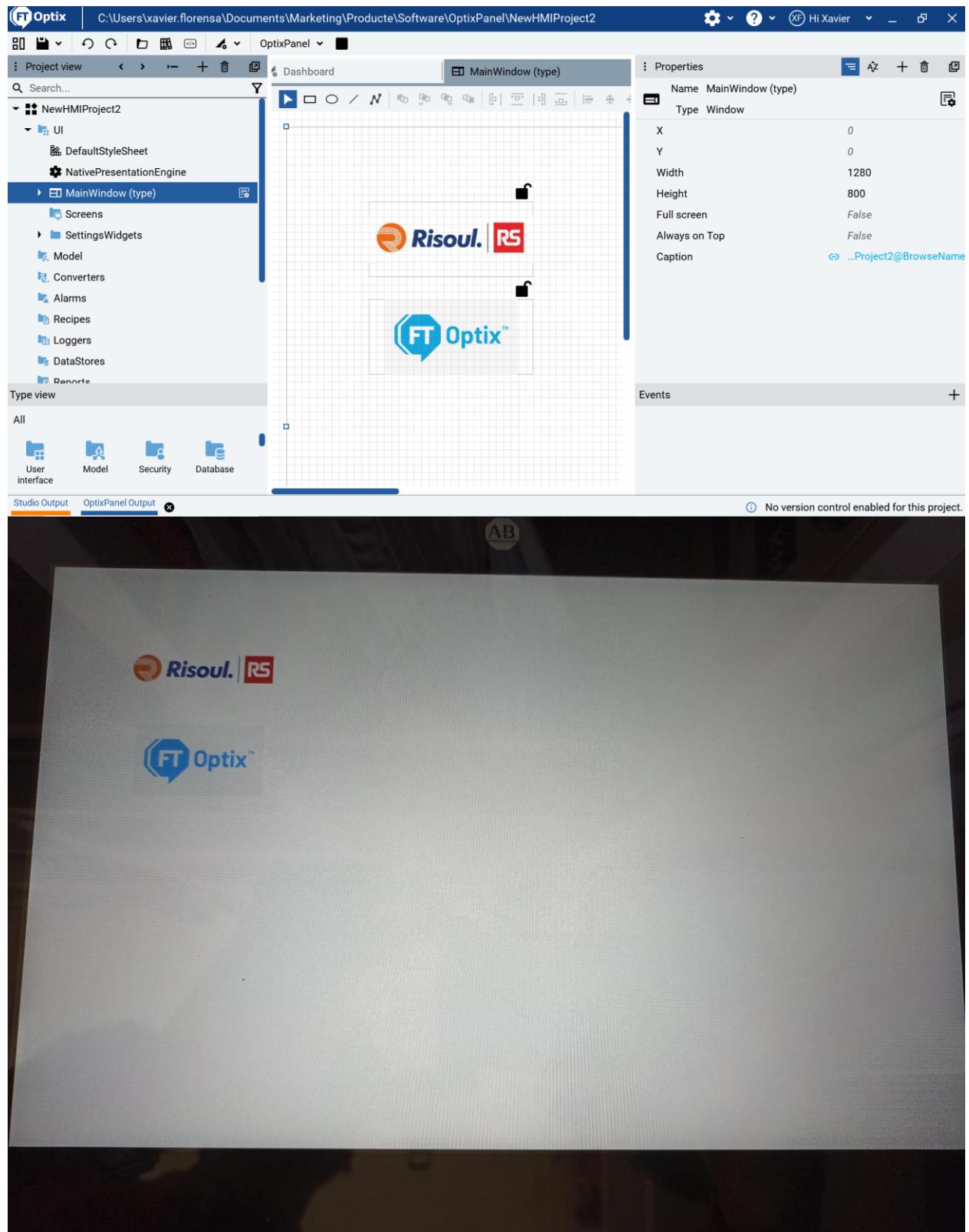


And test with the play button on the Panel

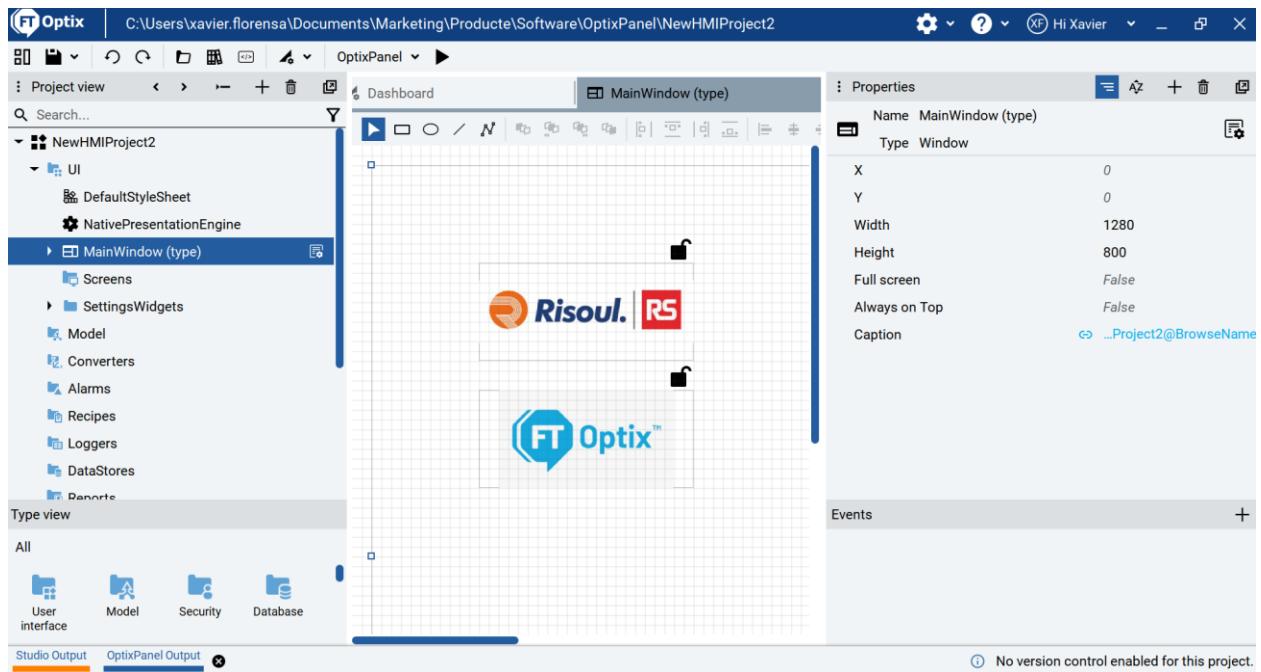
Admin

Password: Risoul10

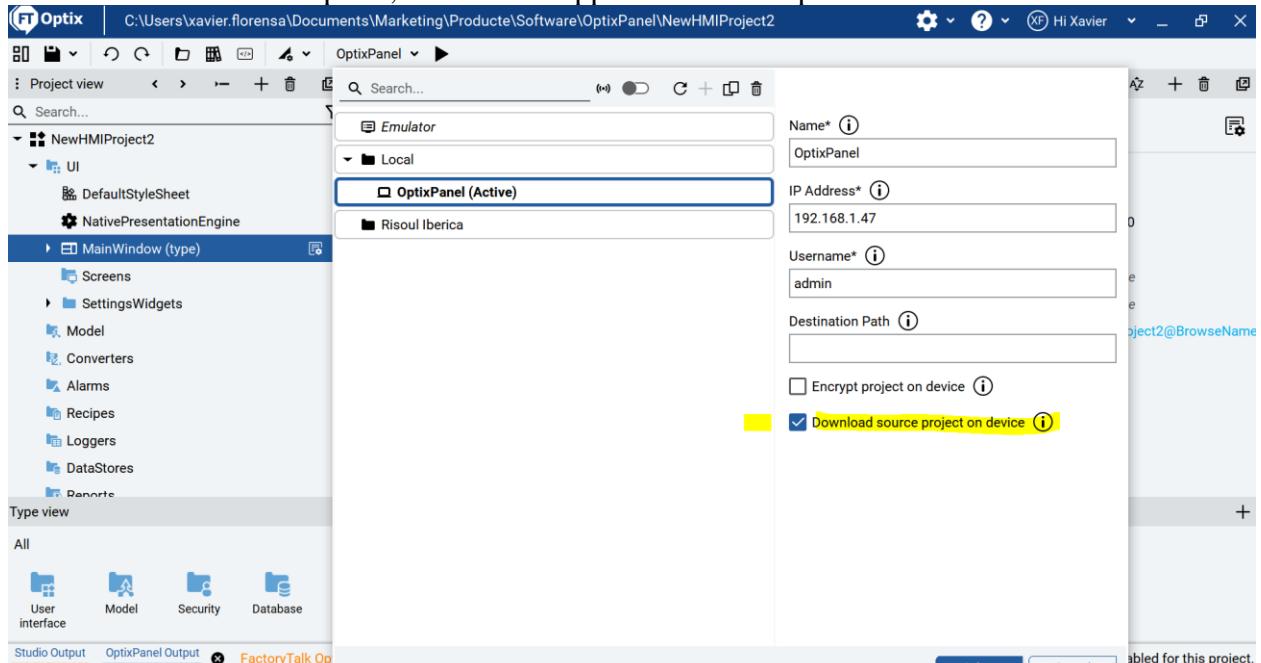




But when you hit the stop button, then the screen becomes full white.



You have to select this option, to leave the application on the panel.



But still when clicking stop the screen becomes full white.

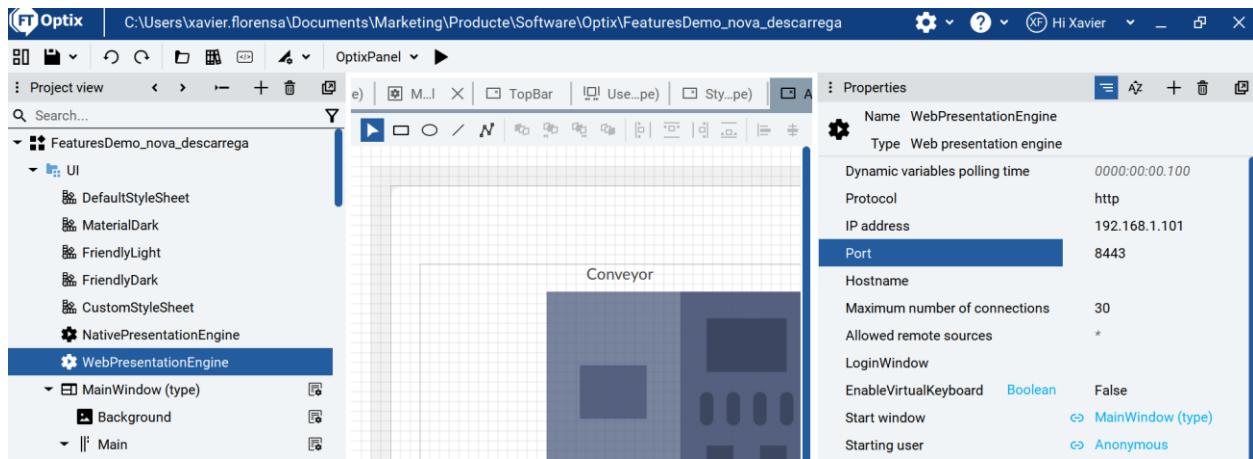
But if you power OptixPanel again it will open with the new application. After 10 seconds.

That's all.

You can use web access from your Laptop

Just plug your Laptop with a patchcord to Eth1 (LAN)

Set your IP laptop to 192.168.1.10 for instance



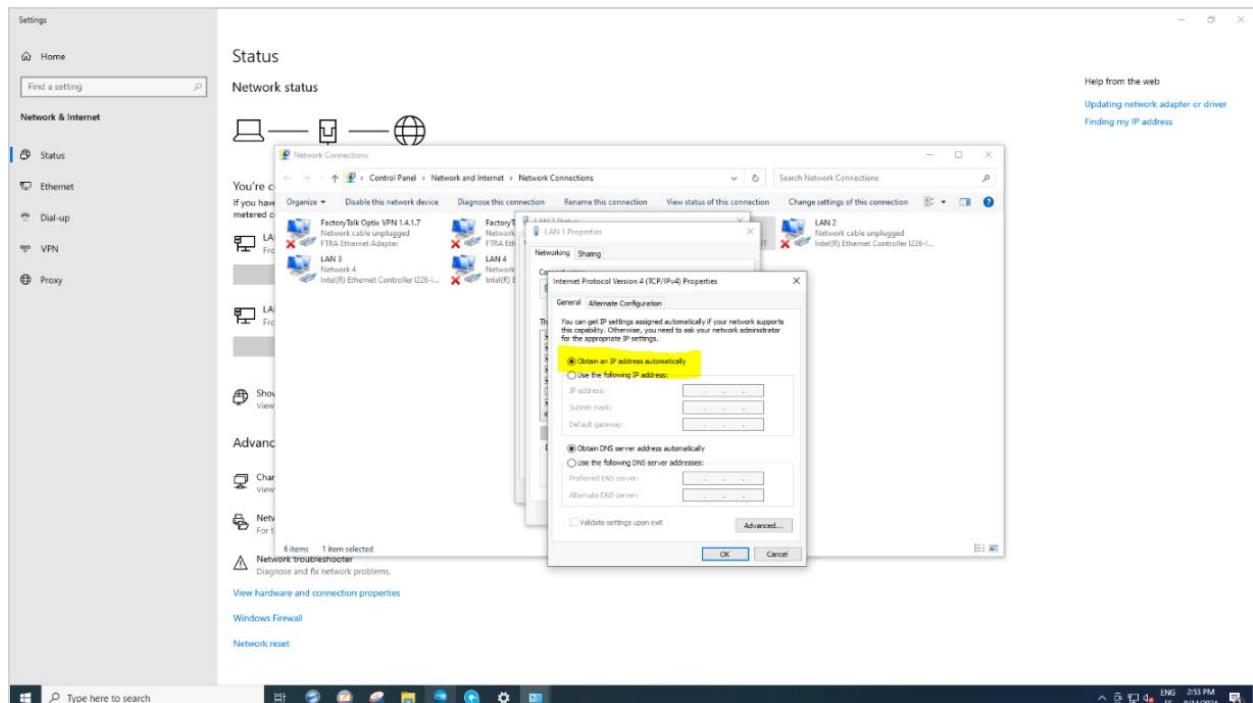
## 25. First steps with ASEM6300B PC and runtime

Install FactoryTalk Remote Access Tools on your ASEM6300B computer.  
Give a fixed known IP address to your ASEM6300B computer.

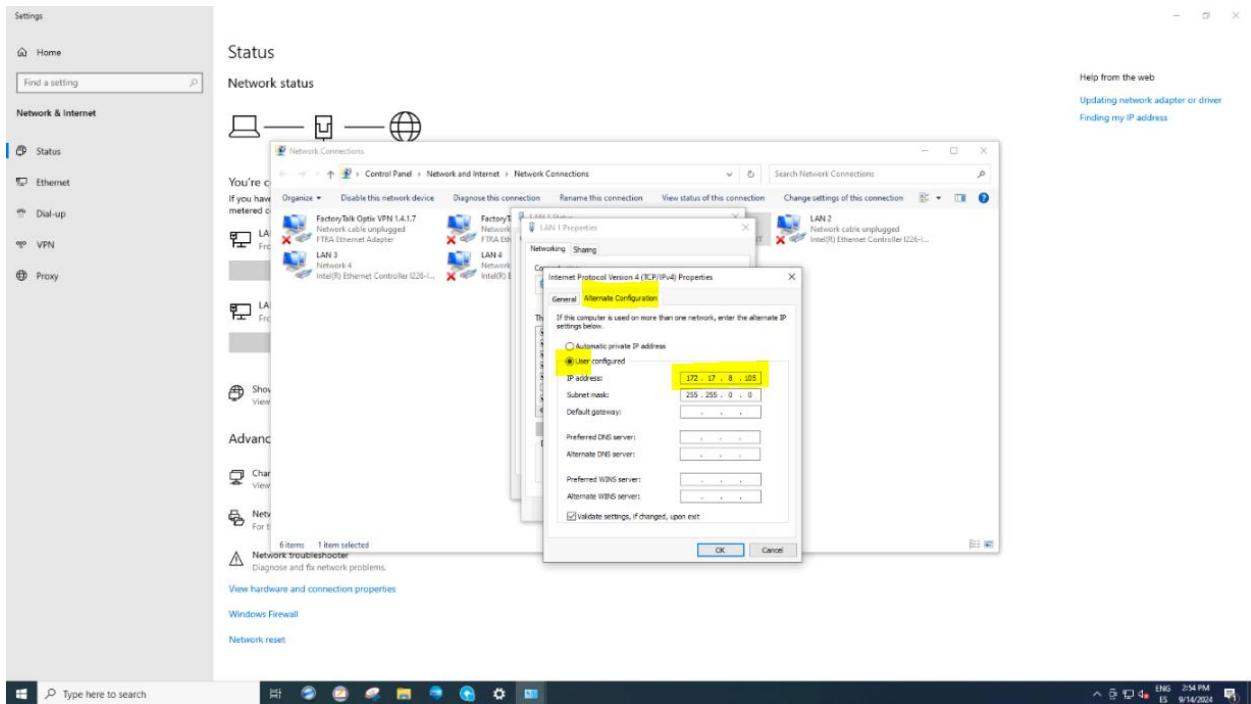
It is recommended to have a DHCP IP address (to be able to connect to the internet to any router everywhere) together with a fixed IP (to connect directly with your Laptop in range for instance 172.17.8.103)

So Laptop fixed IP address 172.168.18.3

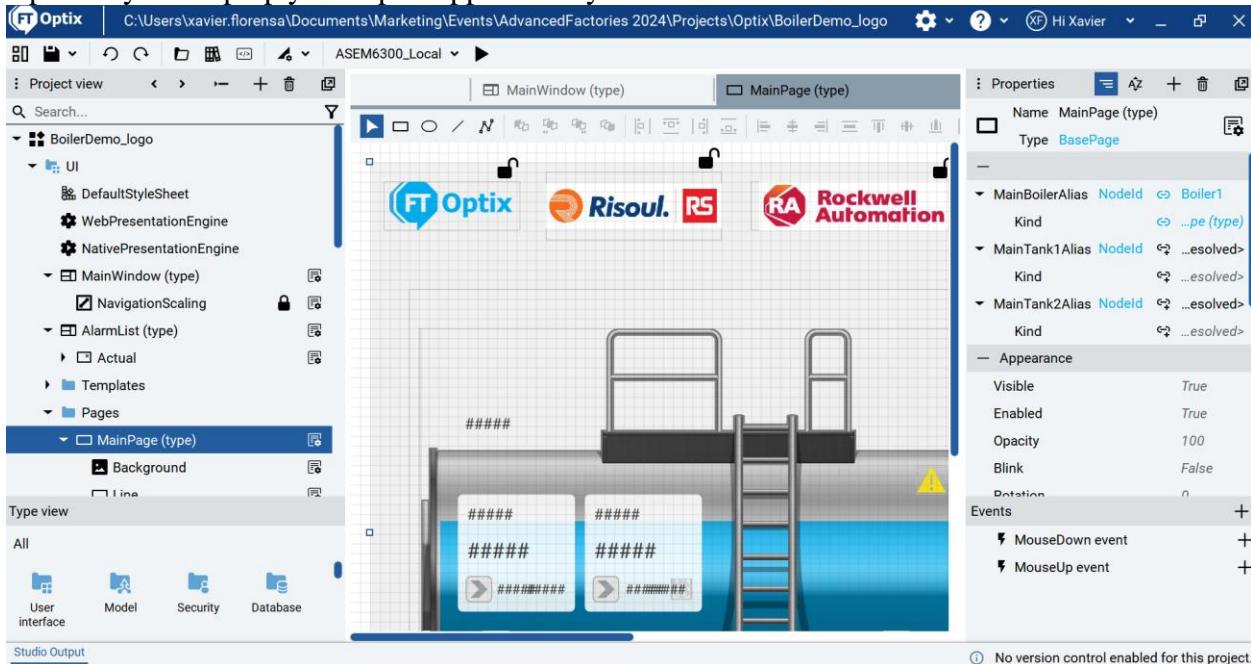
Set ASEM network adapter settings this way



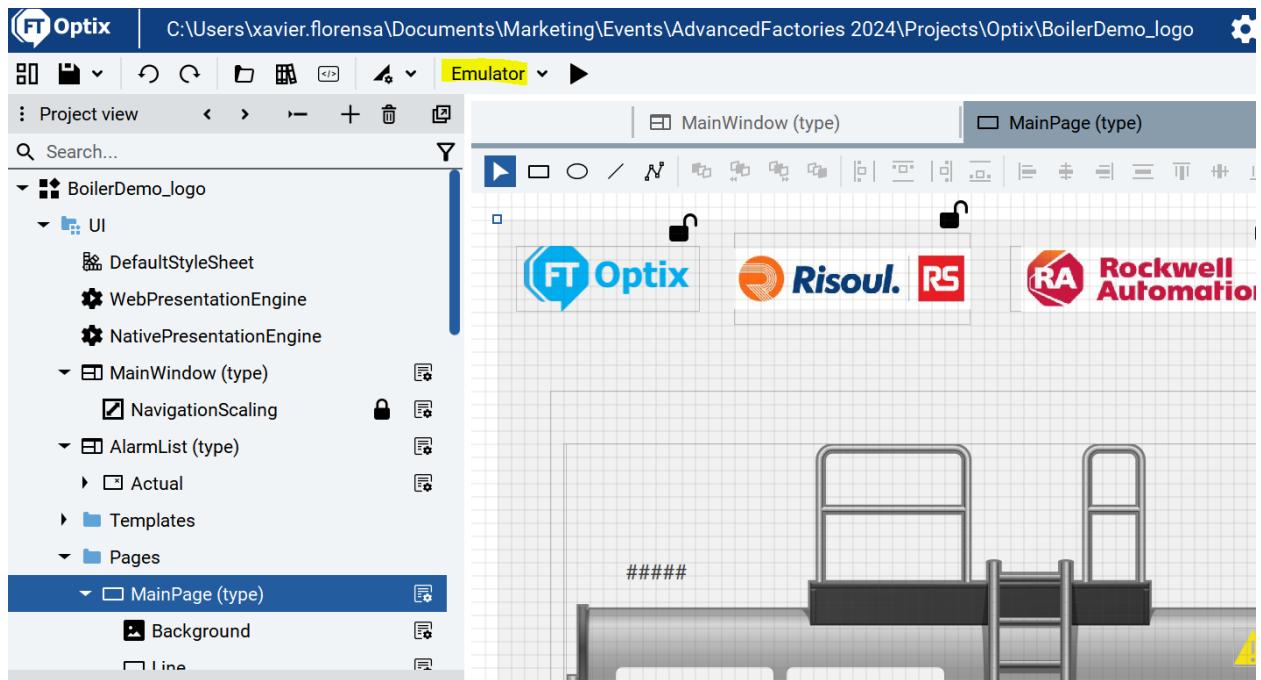
Then add a fixed alternate IP on ASEM 172.17.8.105



Open on your Laptop your Optix application you want to install on the ASEM



Click on the emulator (destination) pane



Add a new device this way

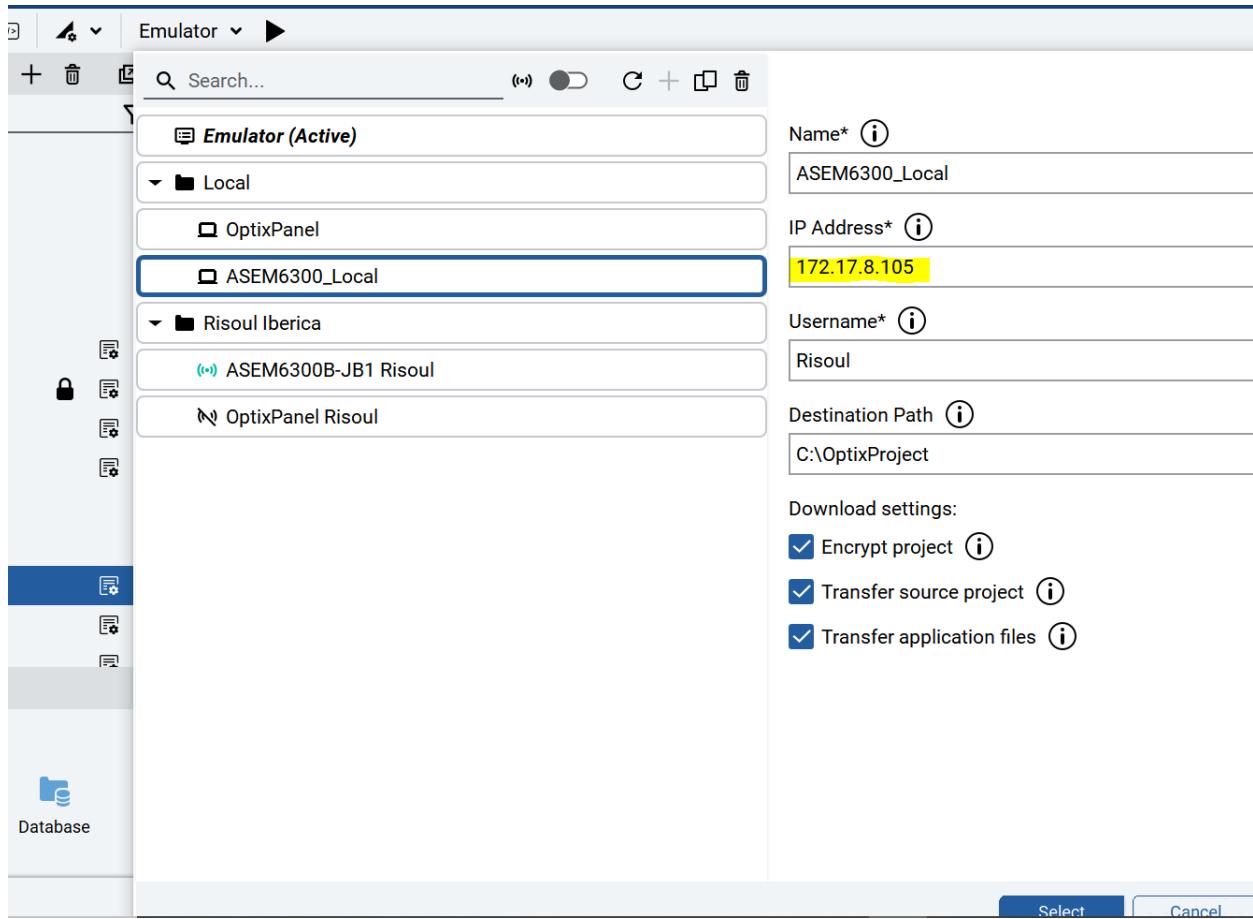
Giving the ASEM6300B fix IP address

Give the username

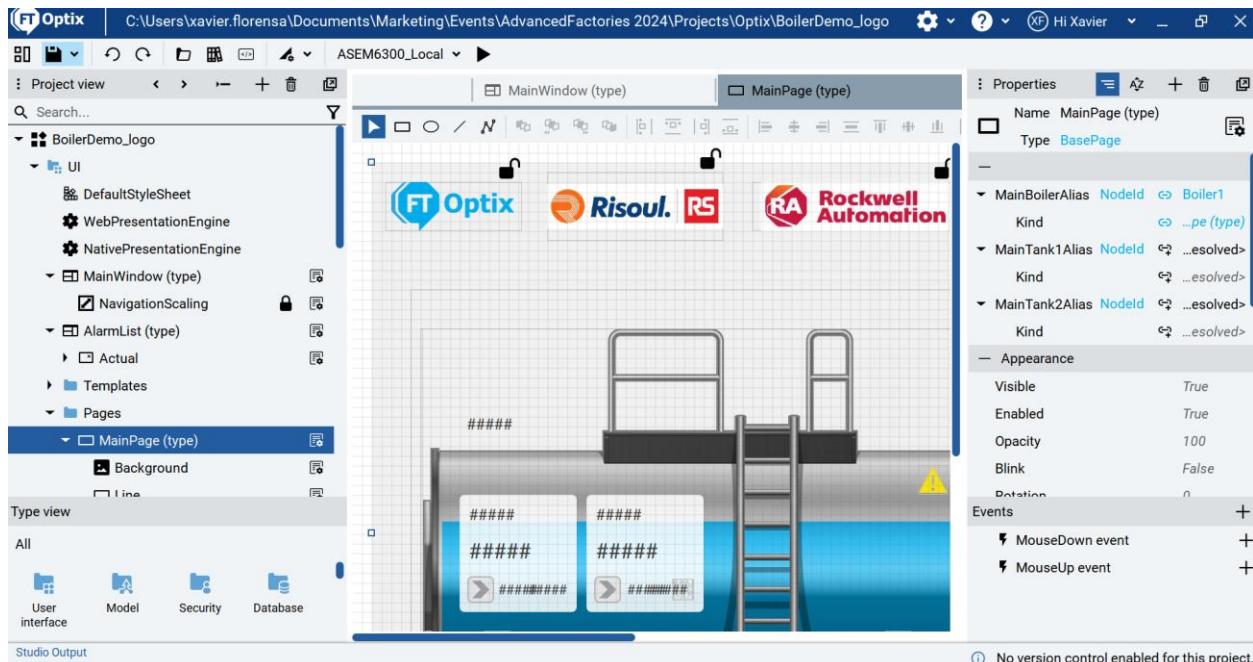
And a destination Directory (you have created previously on your ASEM) like for instance

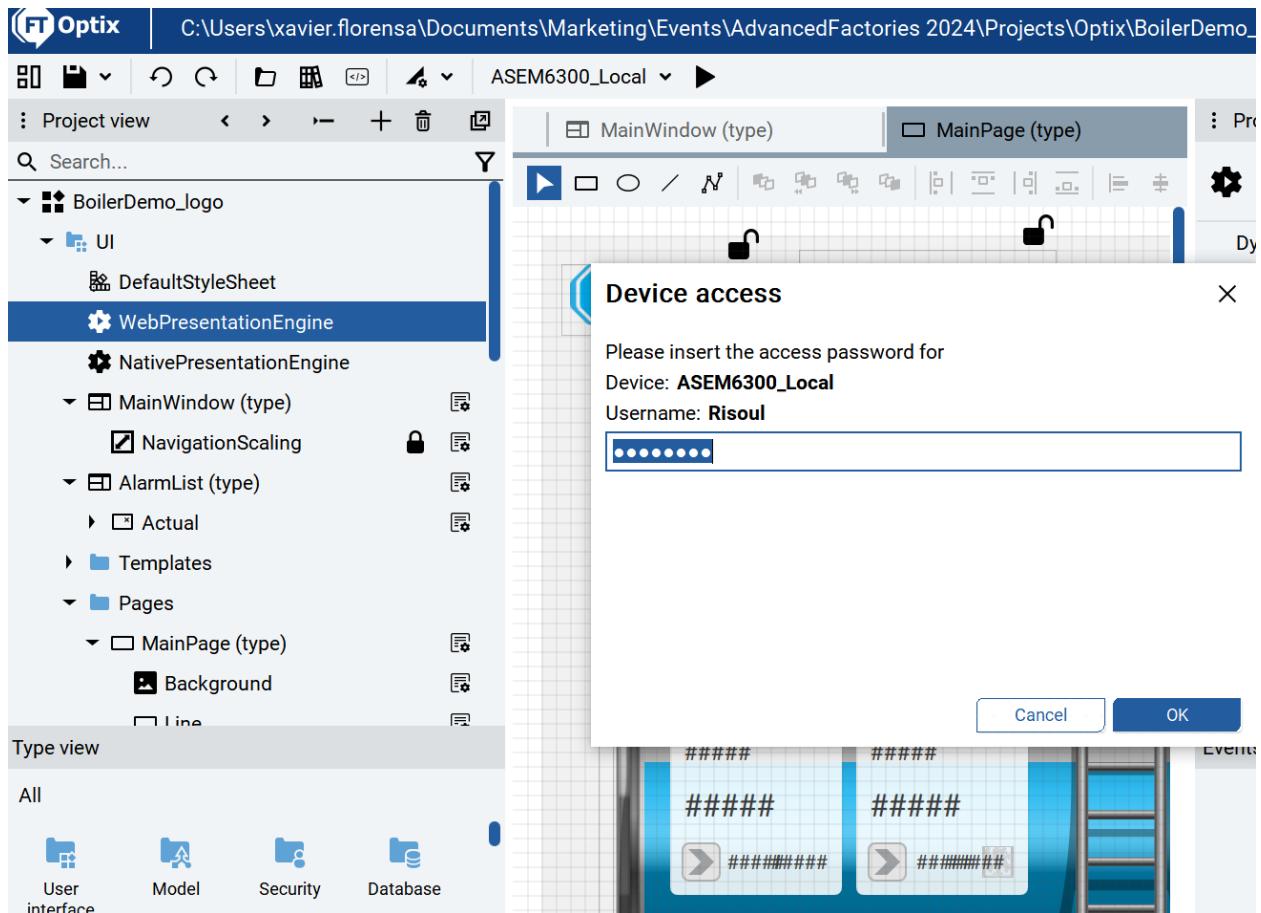
C:\OptixProject

Click on select



Then plug a patchcord between the ASEM6300 and the Laptop (both in same subnet)  
Click on play

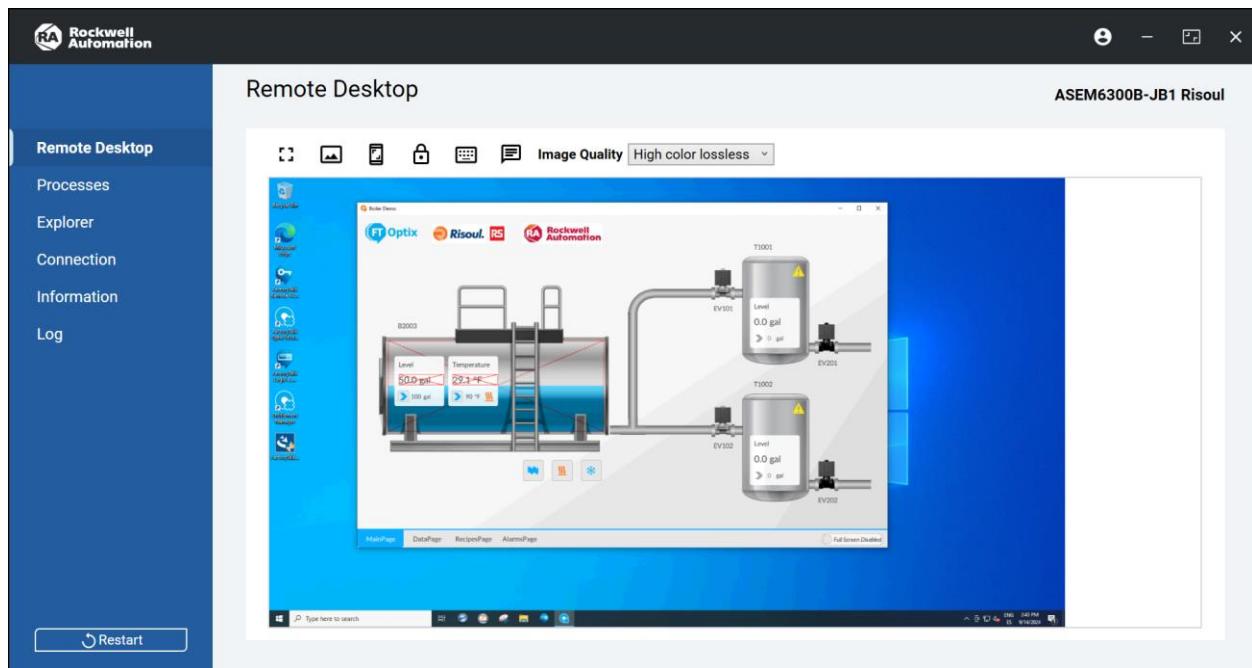




You will see a loading prompt

and the play symbol again.  
But your application is already running on the ASEM

Voilà



If there is a web access on the Optix application, you can open it from a web explorer provided you know the port

## 26. NetLogic and C# tutorial

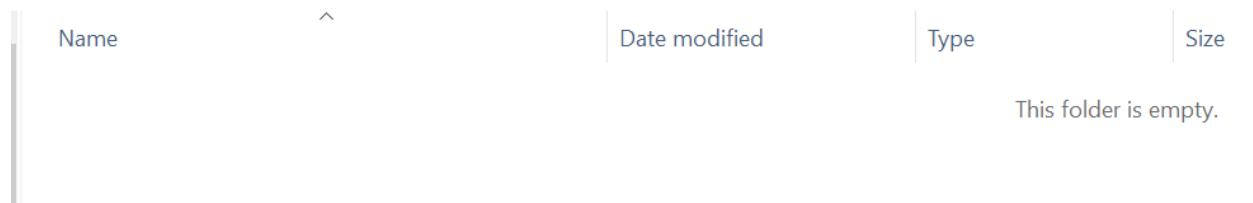
[https://github.com/FactoryTalk-Optix/NetLogic\\_CheatSheet](https://github.com/FactoryTalk-Optix/NetLogic_CheatSheet)  
here you will find these concepts explained

- 
- [General FT Optix good practices](#)
  - [NetLogic overview](#)
  - [Log output](#)
  - [Accessing project nodes](#)
  - [Managing aliases](#)
  - [Objects, types and instances](#)
  - [Asynchronous tasks](#)
  - [Database interaction](#)
  - [Random generation](#)
  - [Resource URI](#)
  - [Variables formatting](#)
  - [RegEx](#)
  - [DialogBox](#)
  - [Variables Interaction](#)
  - [Execute commands](#)
  - [Sessions](#)
  - [Translations](#)
  - [Users and groups](#)
  - [Mouse handling](#)
  - [Dynamic Links](#)
  - [Colors](#)

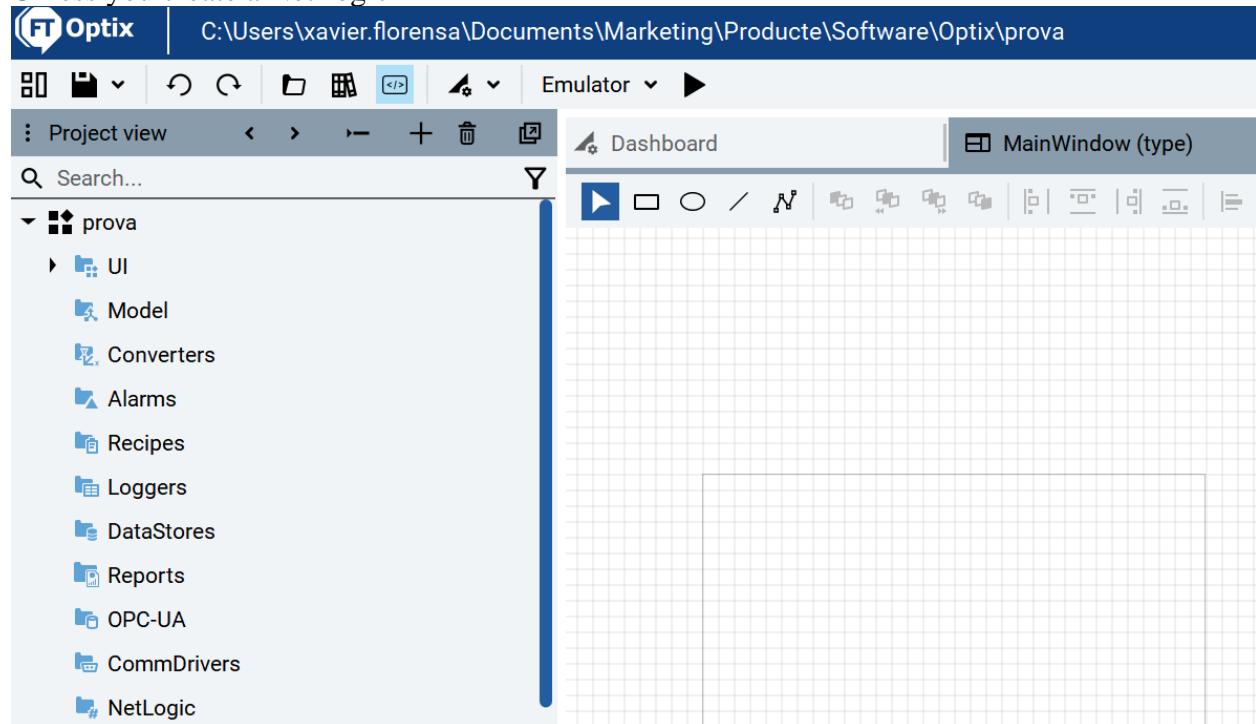
## 27. Understanding compilation of NetLogic code

After you create a new standard project, before creating a Netlogic script and even compiling, you will have nothing on folder Projects/Netlogic/bin  
Let's create a project in FT Optix Studio called prova  
There is nothing on the NetSolution folder

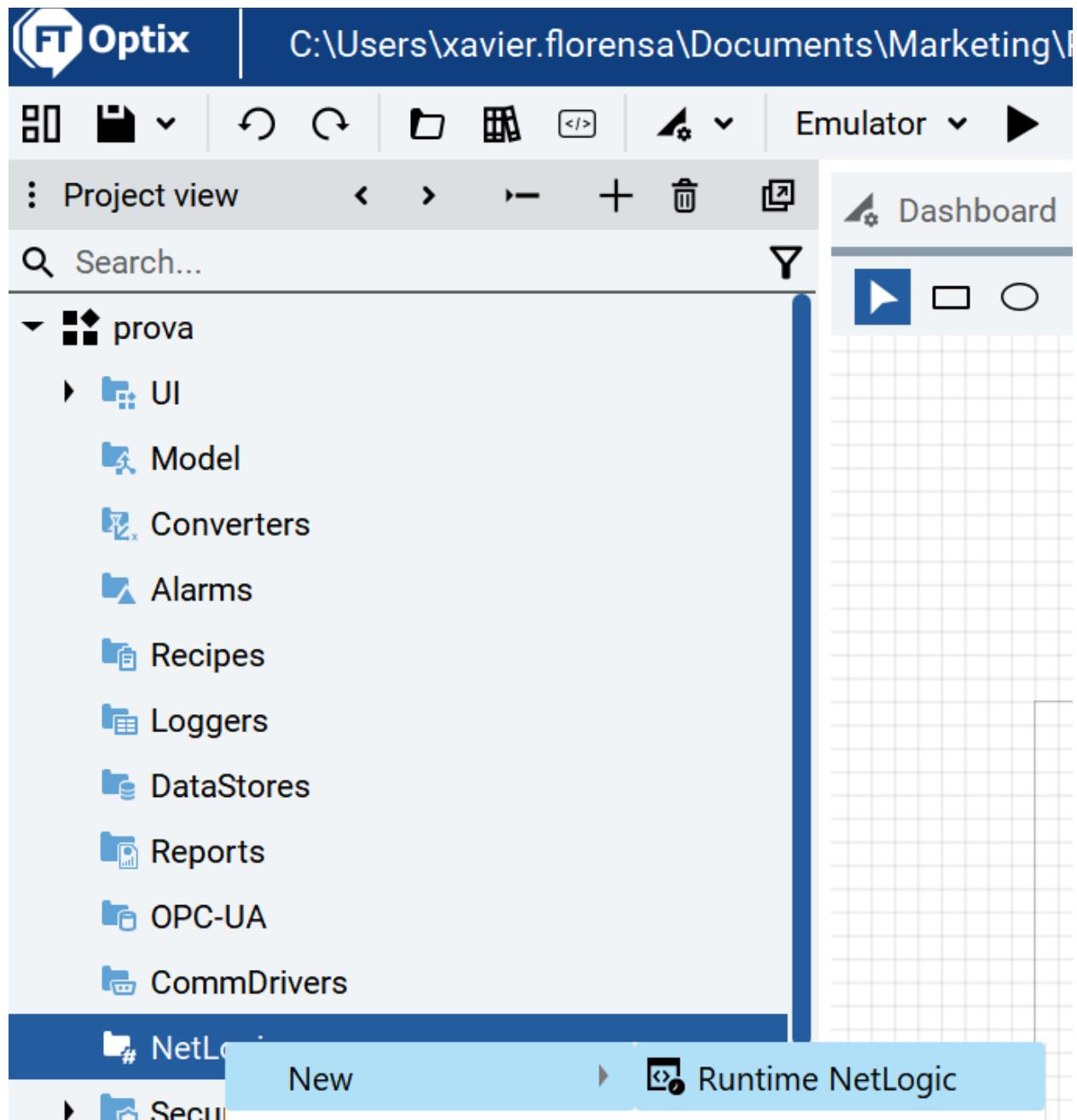
> This PC > Documents > Marketing > Producte > Software > Optix > prova > ProjectFiles > NetSolution



Unless you create a NetLogic



Now let's create a new Netlogic script



Look how is this populated

You will see the file RuntimeNetLogic1 where you have the script.

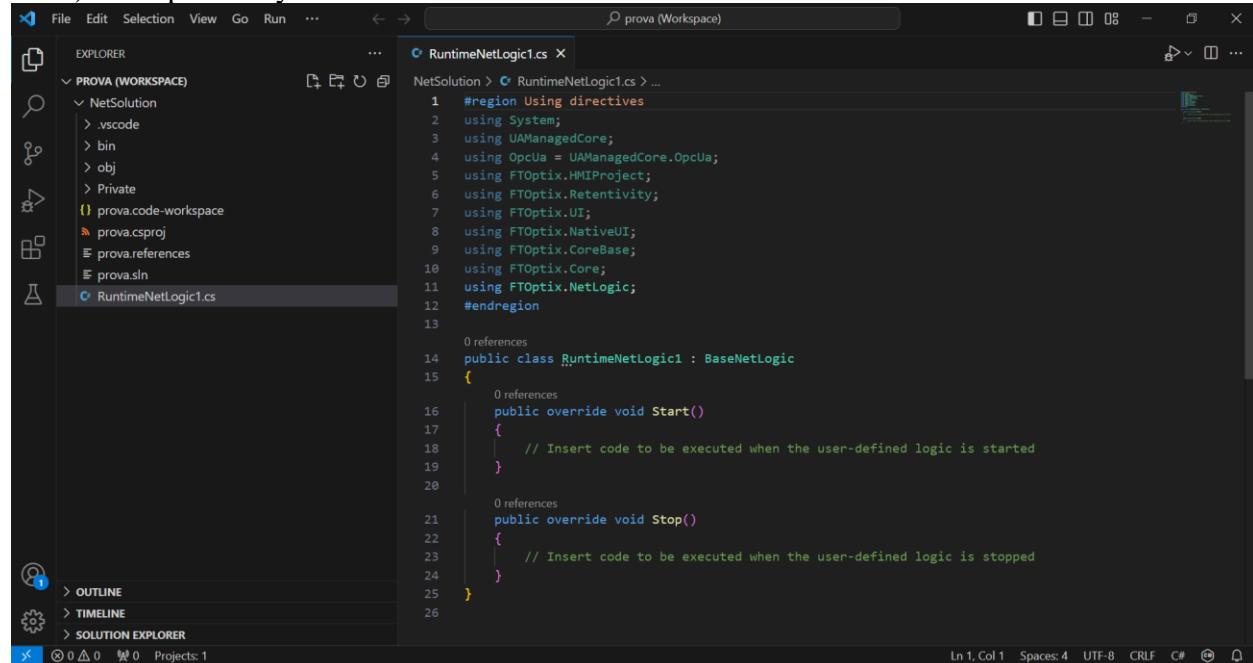
This PC > Documents > Marketing > Producte > Software > Optix > prova > ProjectFiles > NetSolution				
Name	Date modified	Type	Size	
bin	1/6/2024 7:54 AM	File folder		
obj	1/6/2024 7:54 AM	File folder		
Private	1/6/2024 7:54 AM	File folder		
prova.csproj	1/6/2024 7:54 AM	C# Project File	1 KB	
prova	1/6/2024 7:54 AM	REFERENCES File	10 KB	
prova.sln	1/6/2024 7:54 AM	Visual Studio Solution	2 KB	
RuntimeNetLogic1	1/6/2024 7:54 AM	C# Source File	1 KB	

If you look on bin, it is also populated, since creating a new Netlogic also means compiling it. These are the compiled files in binary.

This PC > Documents > Marketing > Producte > Software > Optix > prova > ProjectFiles > NetSolution > bin				
Name	Date modified	Type	Size	
prova.deps	1/6/2024 7:54 AM	JSON Source File	1 KB	
prova.dll	1/6/2024 7:54 AM	Application extension	5 KB	
prova	1/6/2024 7:54 AM	PDB File	13 KB	

When you make changes on the code, you need to compile it, before executing, this is done with save on Visual Studio Code, or with Build in Visual Studio 2022.

If you click on FT Optix Studio on RunTimeNetLogic1 the Visual Studio code (or Visual Studio 2022) will open and you will see this.



```

1 #region Using directives
2 using System;
3 using UAManagedCore;
4 using OpcUa = UAManagedCore.OpcUa;
5 using FTOptix.HMIPProject;
6 using FTOptix.Retentrivity;
7 using FTOptix.UI;
8 using FTOptix.NativeUI;
9 using FTOptix.CoreBase;
10 using FTOptix.Core;
11 using FTOptix.NetLogic;
12 #endregion
13
14 public class RuntimeNetLogic1 : BaseNetLogic
15 {
16     public override void Start()
17     {
18         // Insert code to be executed when the user-defined logic is started
19     }
20
21     public override void Stop()
22     {
23         // Insert code to be executed when the user-defined logic is stopped
24     }
25 }

```

## 28. Understanding Classes (Objects), methods and data

When you write an object or a class, (classes are marked in green color)

For example

```
public override void Start()
{
    // Insert code to be executed when
    Log.]
}
0 references
public o
{
    // I
}
}
0 references
```

The screenshot shows an IntelliSense dropdown menu for the 'Log' method. The menu items are: Debug, Equals, Error, Info, Node, ReferenceEquals, Verbose1, Verbose2, and Warning. The 'Debug' item is highlighted.

Or for example (an instance of a class is marked in sky blue)

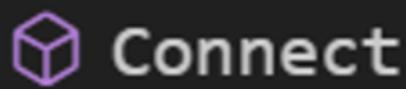
```
}
}
0 references
public void Star
//public overrid
{
    winsock_Ear.]
}
0 references
```

The screenshot shows an IntelliSense dropdown menu for a class instance. The menu items are: BufferSize, Close, Connect, Connected, ConnectionRequest, Container, DataArrival, Disconnected, Dispose, and Disposed. The 'BufferSize' item is highlighted.

then you will write . and you will see (thank you to IntelliSense) a collection of

The tool icon for 'BufferSize' is a wrench symbol.

Tool icon: read write access data



## Connect

Cube icon: method (action to perform a task)



## ConnectionRequest

Blizard: event

## 29. Creating a Method

Imagine we want to create a new Method to add  $2+3=5$

Create a new Optix project, and add a new Runtime Netlogic

You have to use the return statement

Copy and paste this code (or complete with what you have on the code)

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrjject;
using FTOptix.Rentativity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public int metodo_suma(int primero, int segundo)
    {
        return primero + segundo;
    }
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        int resultado = metodo_suma (2,3);
        Log.Info("resultado: "+resultado);
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```



## Save and run emulator

The screenshot shows the FT Optix software interface. The title bar says "C:\Users\xavier.florensa\Documents\Marketing\Products\Software\Optix\prova". The left sidebar shows a project view with a single project named "prova" containing "UI" and "Model" subfolders. The main area is titled "Dashboard" with the sub-section "Events". A central message log window displays the following log entries:

Code	Timestamp	Category	Message
①	2024-01-07 14:35:18...	FTOptixRuntime	No license tokens found ↴ FactoryTalk Optix Runtime will be closed in: 120 minutes
①	2024-01-07 14:35:18...	FTOptixRuntime	FactoryTalk Optix Runtime is currently using 1 feature token ↴ Component: Native presentation engine consumes 1 feature token ↴
①	2024-01-07 14:35:18...	FTOptixRuntime	Starting project prova
①	2024-01-07 14:35:18...		resultado: 5

We could use this method after executing the method of pushing a button.

Let's do this

Add this sentence in the same space

```
[ExportMethod]

    public void Add(int sum1, int sum2)
    {
        Log.Info("Resultado tras pulsar botón: "+metodo_suma(sum1,sum2));
    }
```

So the complete code will be like this

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Rettentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public int metodo_suma(int primero, int segundo)
    {
        return primero + segundo;
    }
}
```

```

}

[ExportMethod]

public void Add(int sum1, int sum2)
{
    Log.Info("Resultado tras pulsar botón: "+metodo_suma(sum1,sum2));
}

public override void Start()
{
    // Insert code to be executed when the user-defined logic is started
    //int resultado = metodo_suma (2,3);
    //Log.Info("resultado: "+resultado);
}

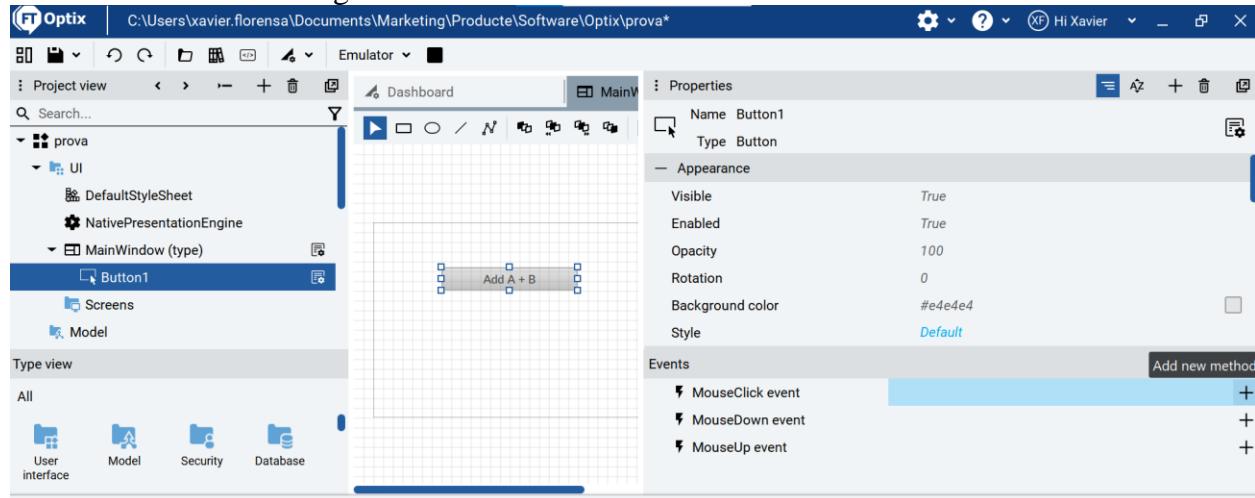
public override void Stop()
{
    // Insert code to be executed when the user-defined logic is stopped
}
}

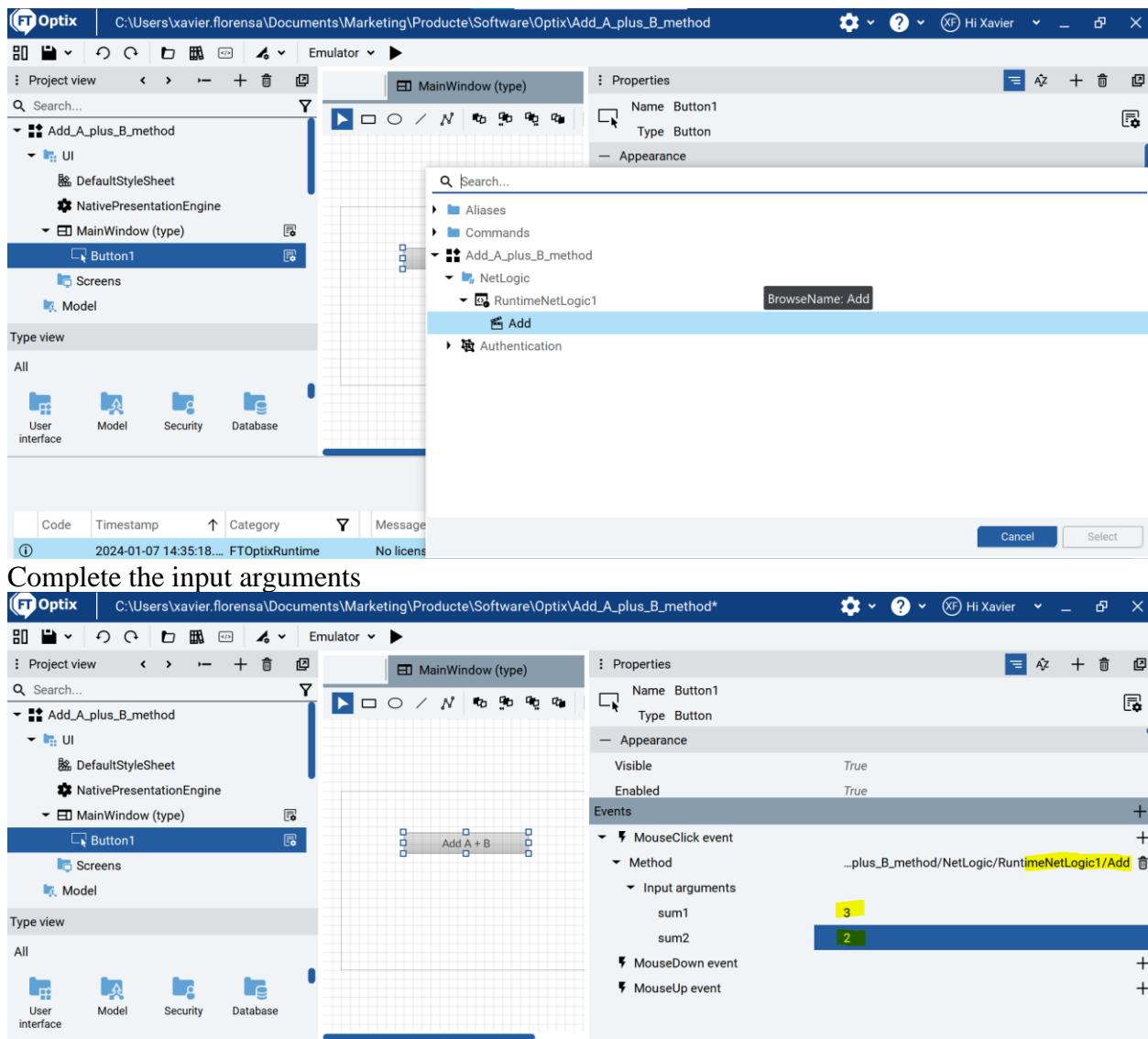
```

### Save on VS Code

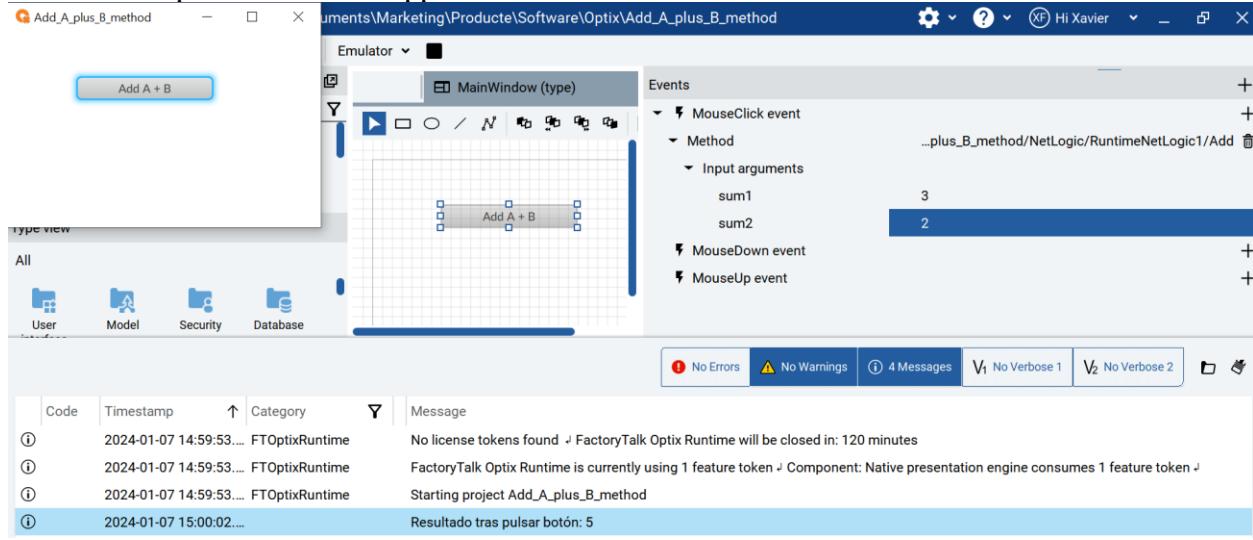
You have to configure this button click action on your Optix environment

Create a button and configure Mouseclick event





### Complete the input arguments



## 30. Linking events with Methods without input parameters

Create a new project and on Runtime Netlogic copy and paste this

```
[ExportMethod]
```

```
public void MyNewMethod()
{
}
```

Populate with the task you want to execute

For instance

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    [ExportMethod]

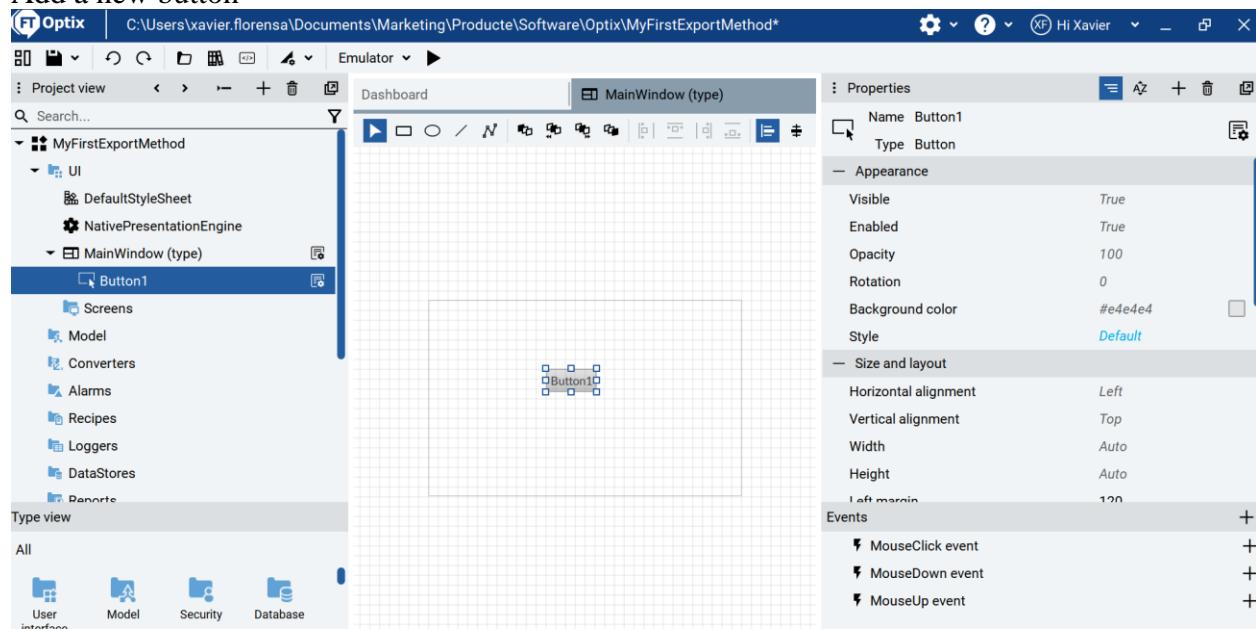
    public void MyNewMethod()
    {
        Log.Info("Hello World, a button has been pressed");
    }

    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
    }

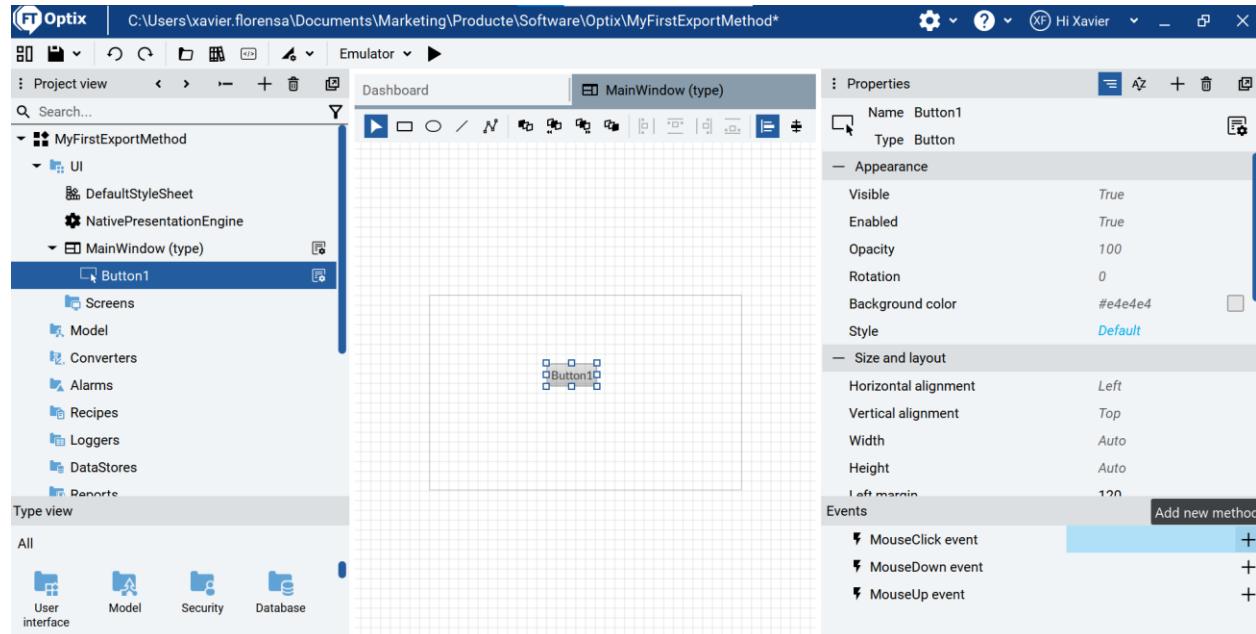
    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```

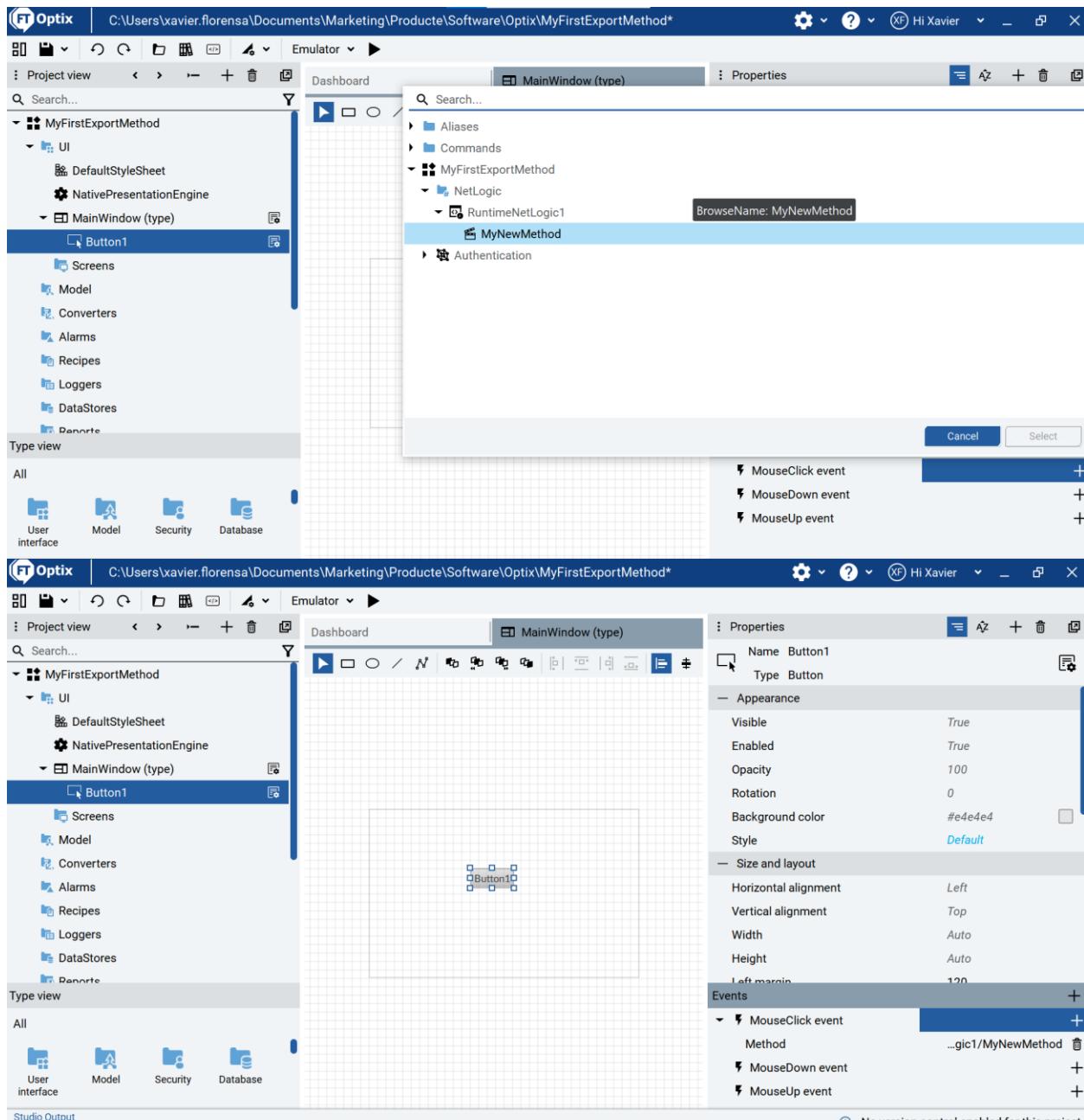
Next, save the code (in order for the compiler to build the new Method)

## Add a new button

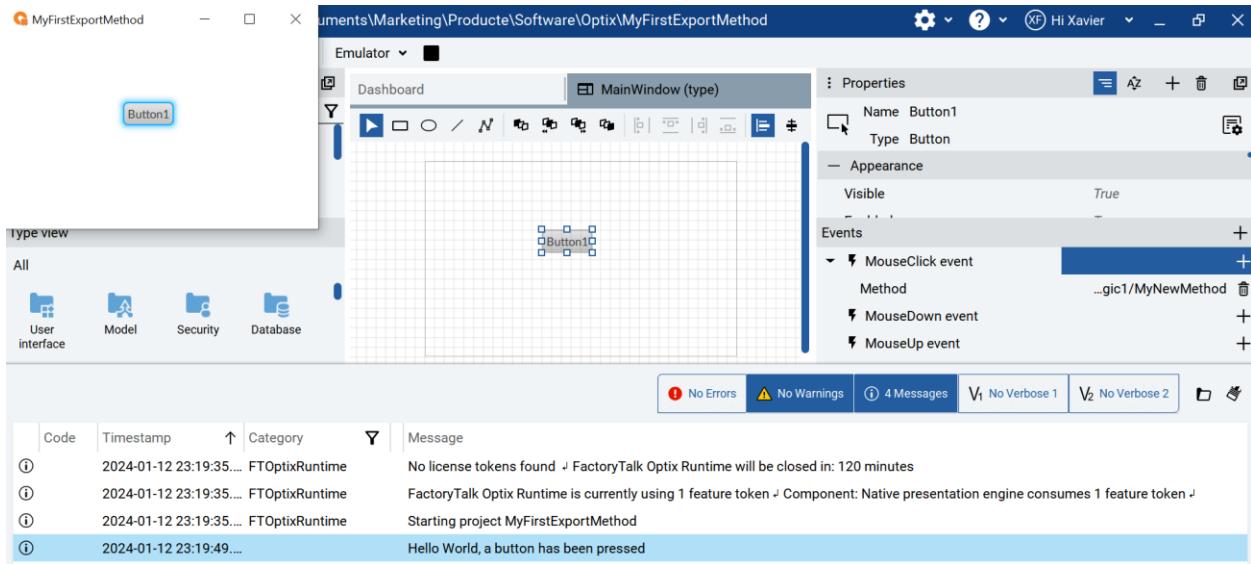


## Add a new MouseClick event





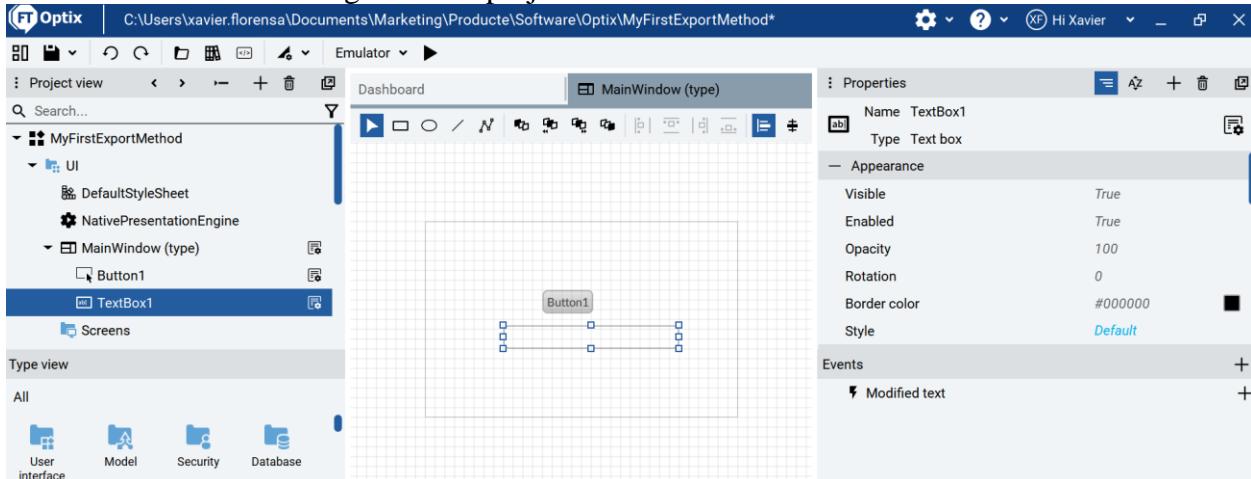
Now execute the project



Next you would like to write the text that you will output

### 31. Linking events with Methods with input parameters

Create a textbox continuing with last project



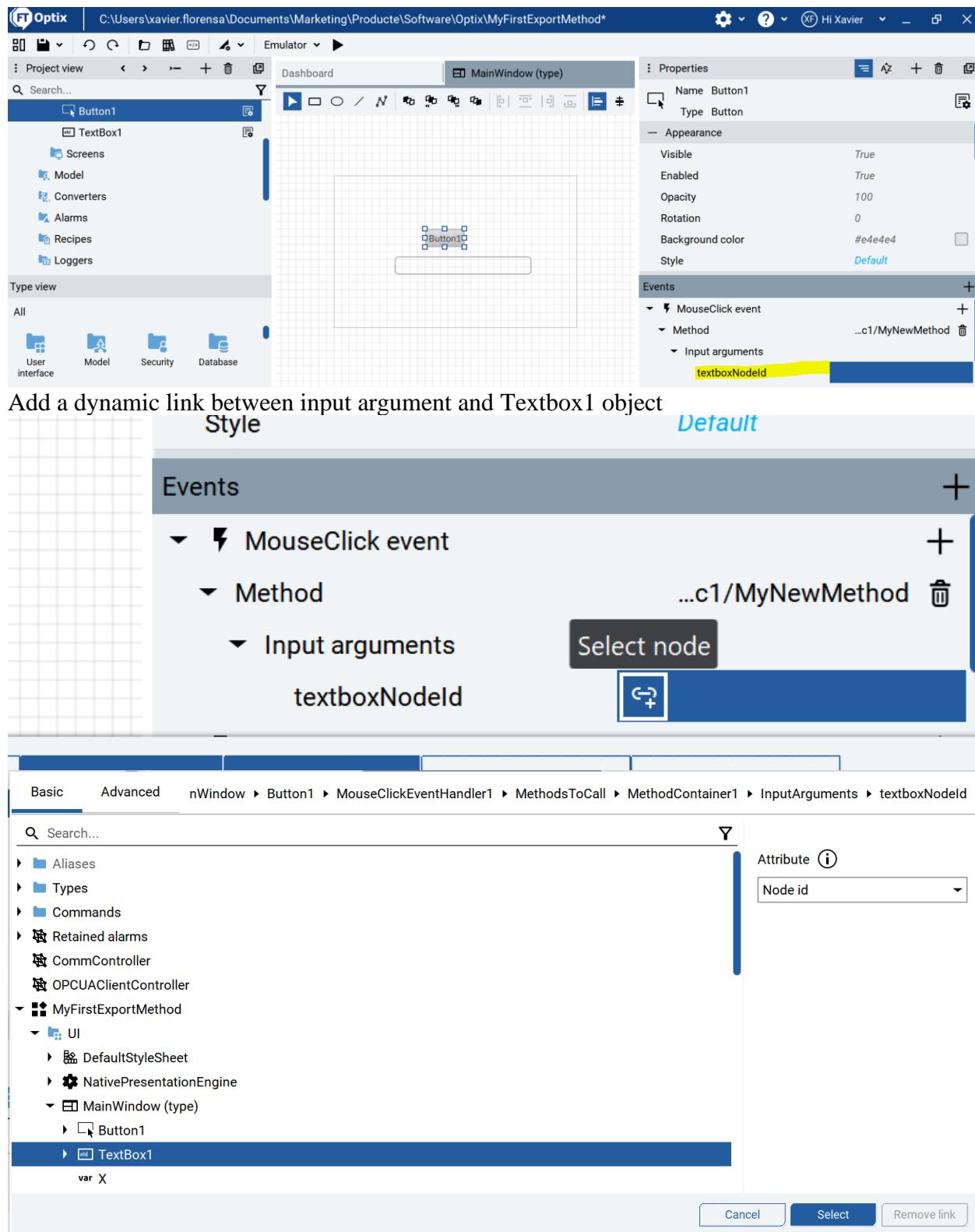
Add this code to the Netlogic code

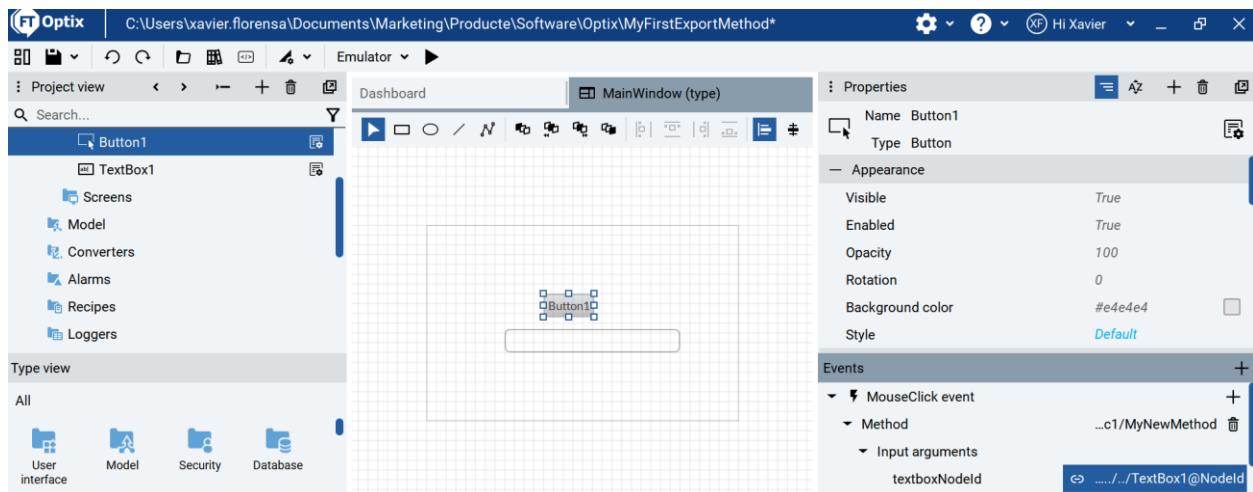
```
[ExportMethod]

    public void MyNewMethod(NodeId textboxNodeId)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        string messagetosend = textbox.Text;
        Log.Info(messagetosend+"Hello World, a button has been pressed");
    }
}
```

Save the code

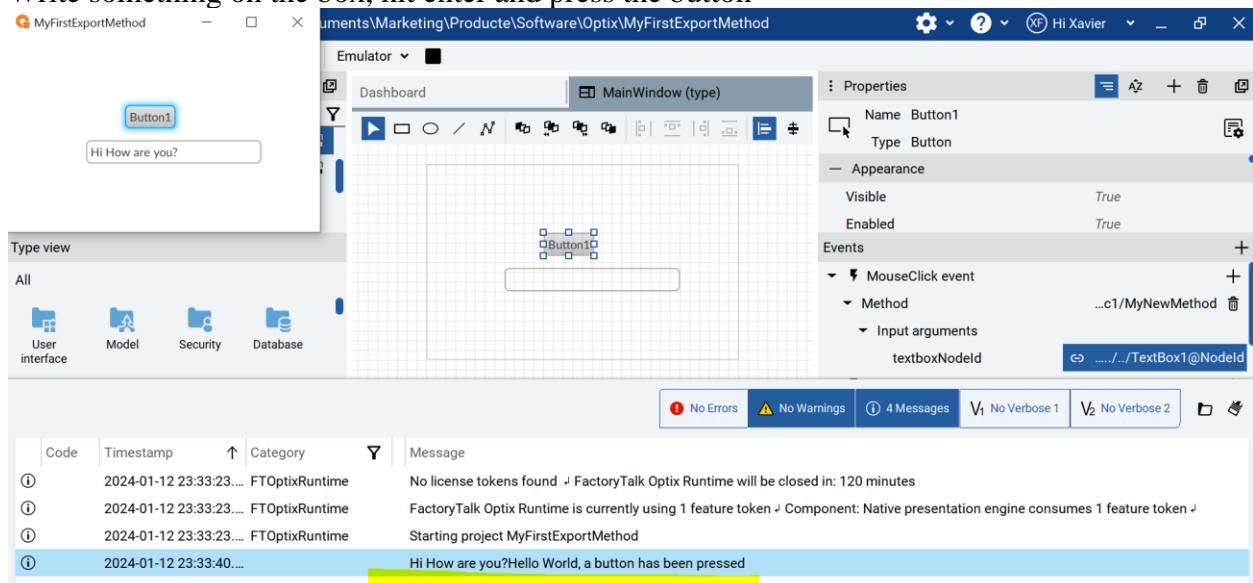
Now when you add a mouseclickbutton you will find an inputparameter





Execute the code

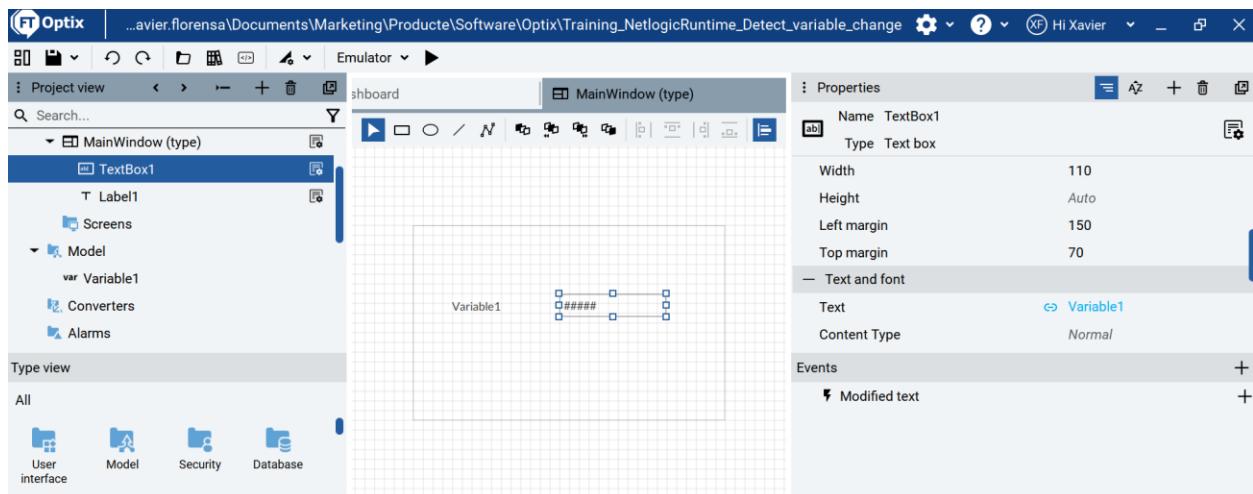
Write something on the box, hit enter and press the button



## 32. Asynchronous tasks, creating callback functions

Let's create a function to detect a variable change, without polling. This is called a callback function.

Create a new project with a textbox and a variable



### Create this Netlogic

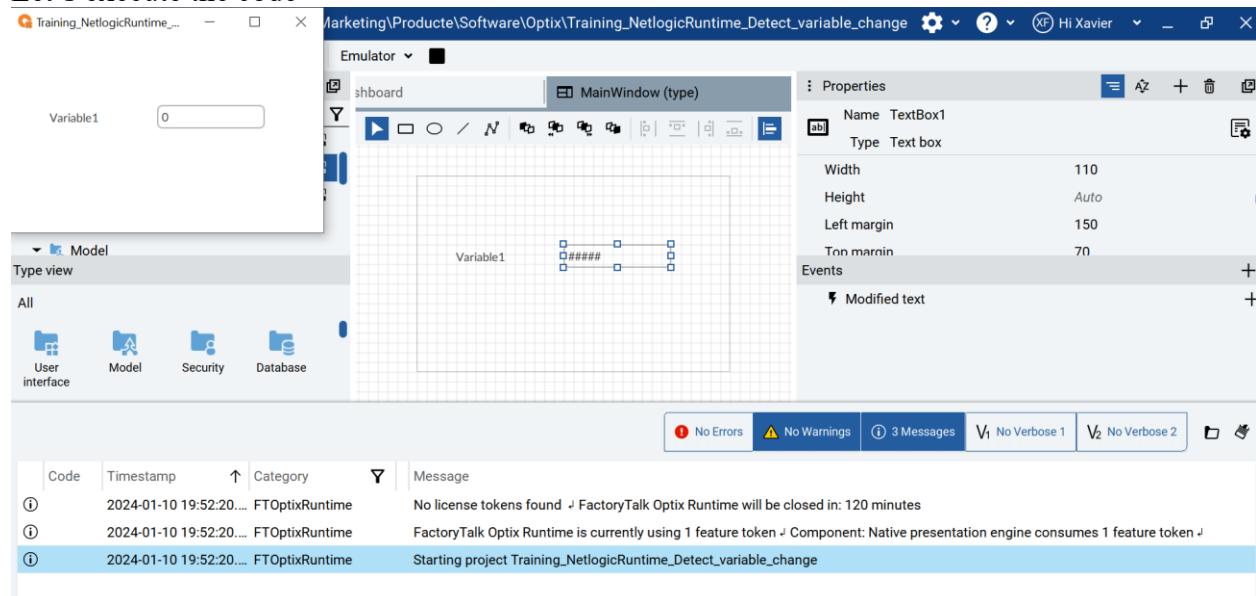
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    private IUAVariable variable1;
    public override void Start()
    {
        variable1 = Project.Current.GetVariable("Model/Variable1");
        //assing a callback function to be executed when the variable changes
        variable1.VariableChange += variable1_VariableChange;
    }

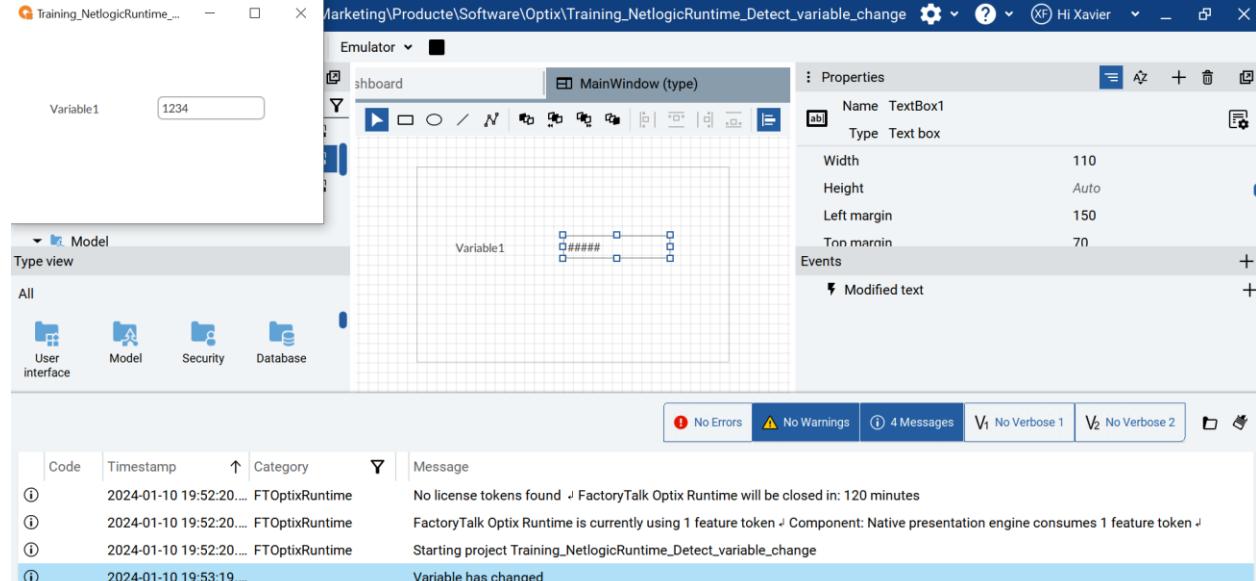
    public override void Stop()
    {
    }
    private void variable1_VariableChange(object sender, VariableChangeEventArgs e)
    {
        Log.Info("Variable has changed");
    }
}
```

}

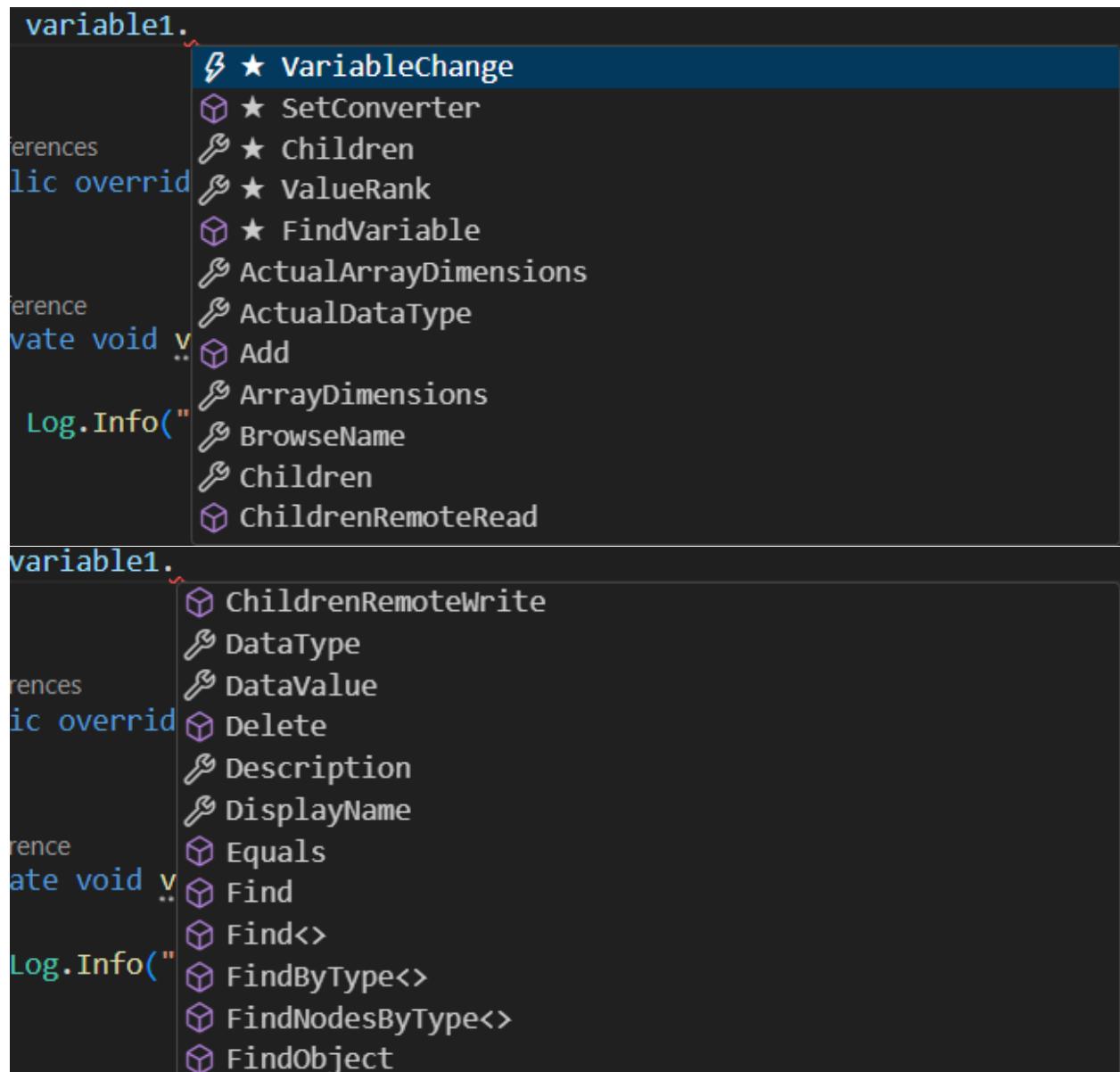
Let's execute the code



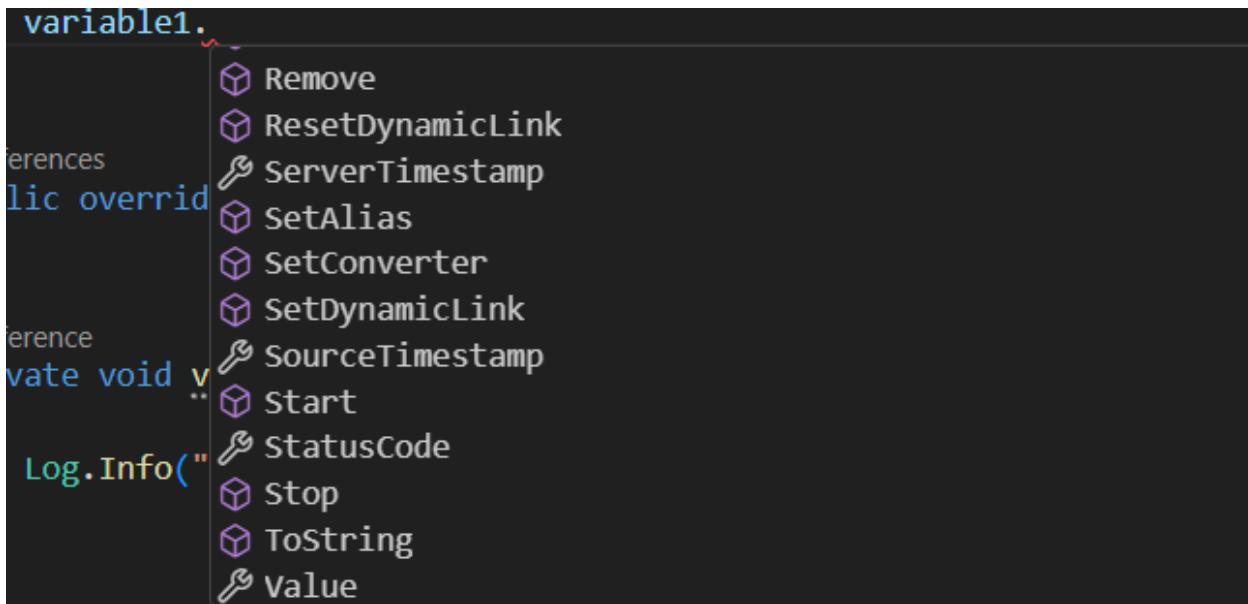
Now change the variable writing a number in the textbox



You can access other methods, events and data for the variable,  
Take a look, if you write the instance, followed by . you will see all:



```
variable1.  
    ↴ FindVariable  
    ↴ Get  
    ↴ Get<>  
    ↴ GetAlias  
    ↴ GetByType<>  
    ↴ GetHashCode  
    ↴ GetNodesByType<>  
    ↴ GetObject  
    ↴ GetPanelLoader  
    ↴ GetType  
    ↴ GetVariable  
    ↴ IsInstanceOf  
  
variable1.  
    ↴ IsInstanceOf  
    ↪ IsValid  
    ↴ MoveDown  
    ↴ MoveUp  
    ↪ NodeClass  
    ↪ NodeId  
    ↪ Owner  
    ↪ Prototype  
    ↪ QualifiedBrowseName  
    ↪ Quality  
    ↴ RemoteRead  
    ↴ RemoteWrite
```



So which input argument must be written on the callback function to make it work properly?  
For instance what to write here?

```
private void variable1_VariableChange(object sender, "What to write here?" e)
```

Just hover your **mouse over** the  
`variable1.VariableChange` event and you will have the solution:

```
object..  
//assing a EventHandler<VariableChangeEventArgs> IUAVariable.VariableChange  
variable1.VariableChange += variable1_VariableChange;
```

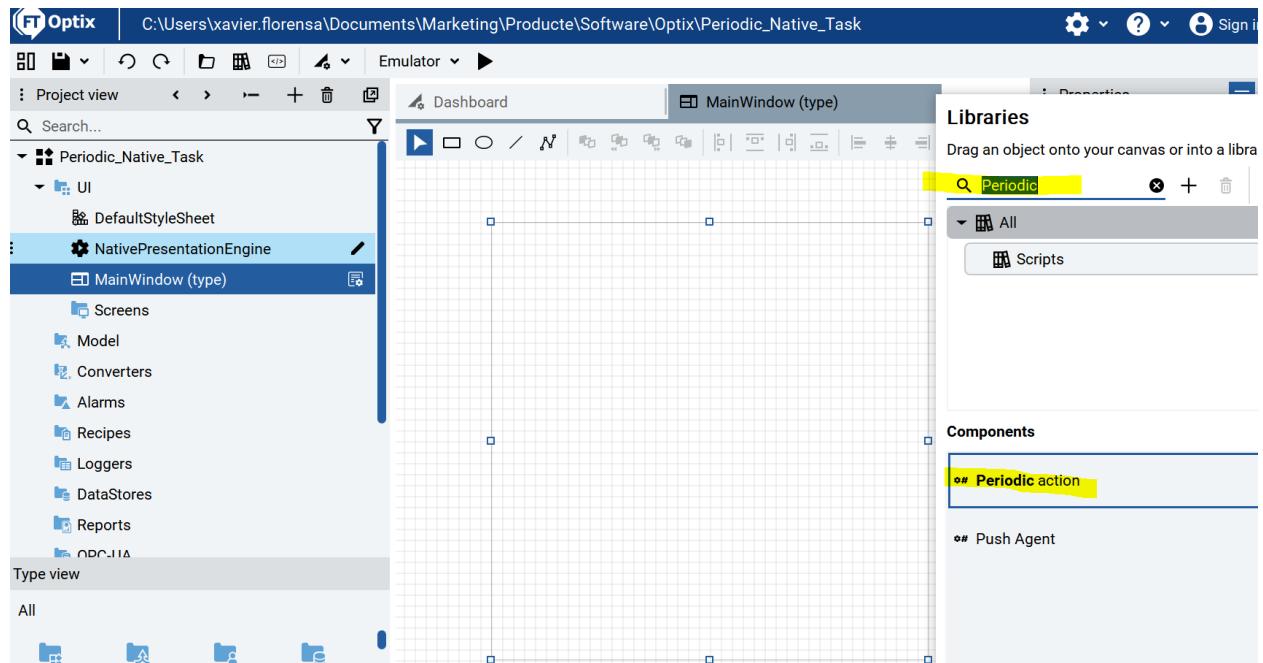


That's all.

## 33. Periodic Tasks

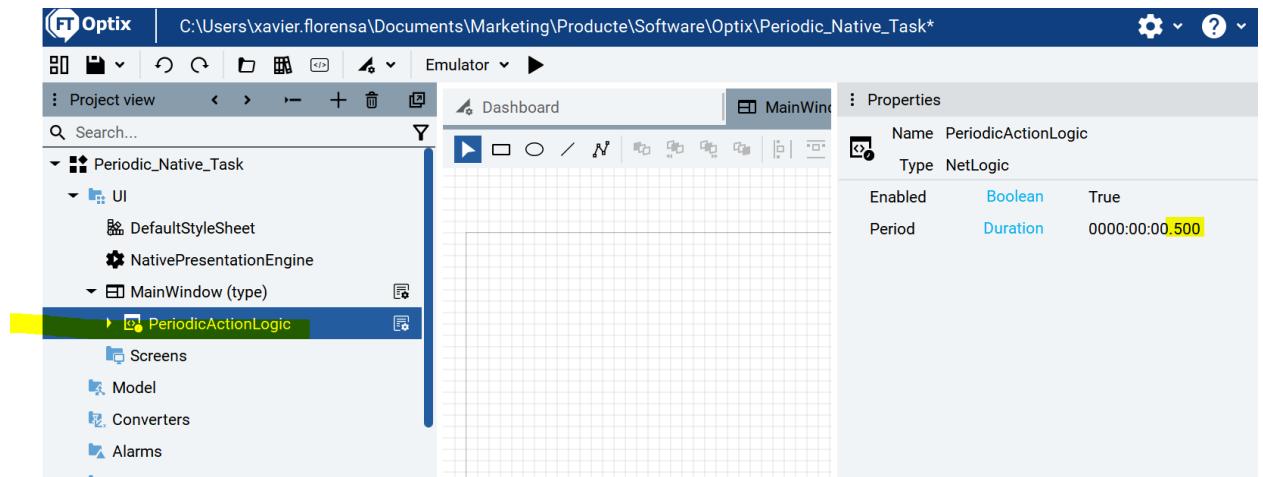
### 33.1. Native Periodic Tasks

You have a library to do so  
Let's create a blank project  
Open library tool and search for Periodic

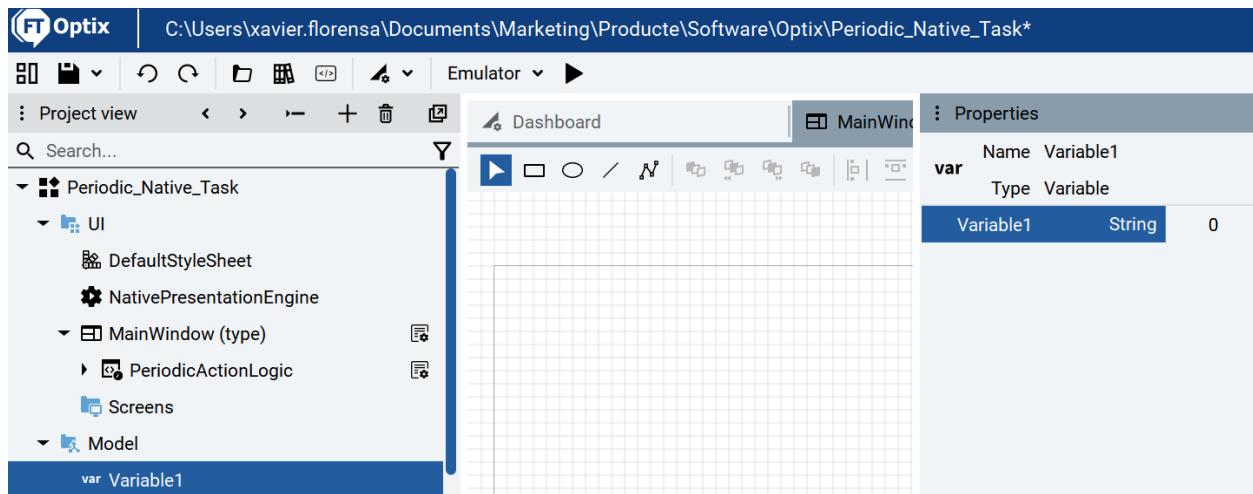


Drag and drop the “Periodic action” component to the desired scope, for instance, Main Window  
A new PeriodicActionLogic script will appear under Mainwindow (but you do not need to go to C#)

You have the period, for instance each 500 mseconds



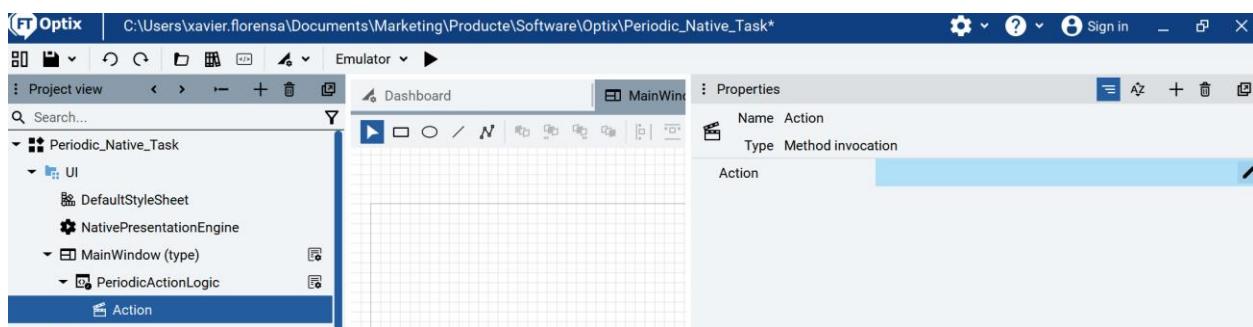
Then you can add an action, for instance modify a variable value  
Add a variable to Model



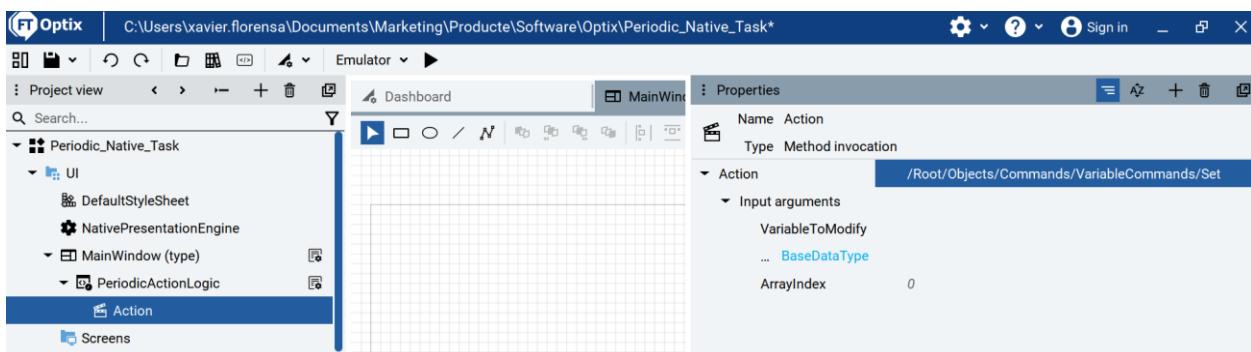
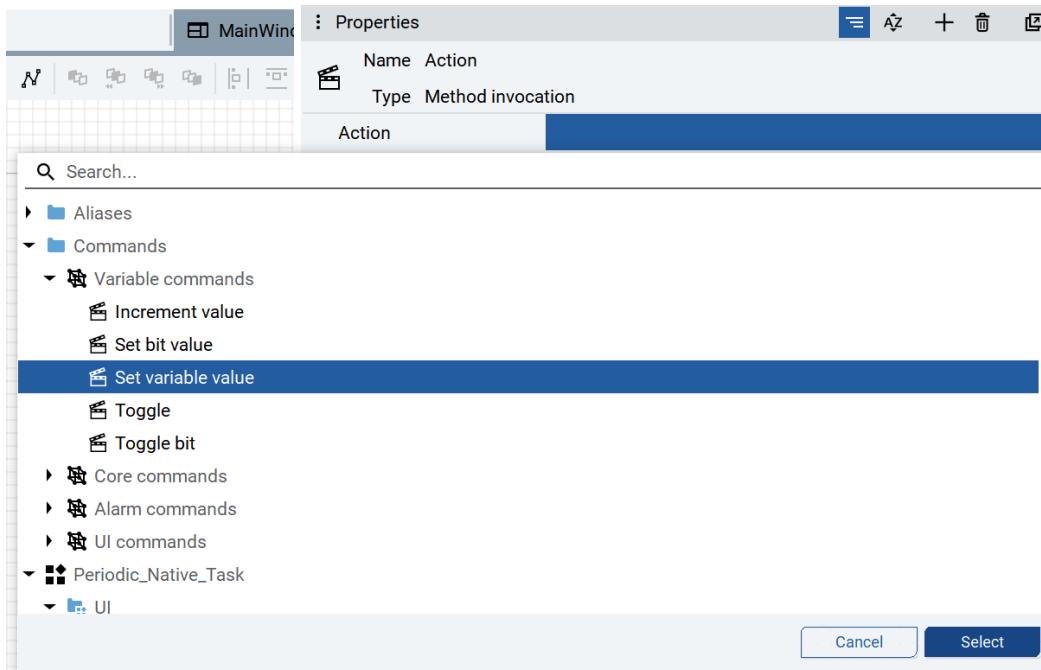
Then add an action to the Periodic

Click on Action

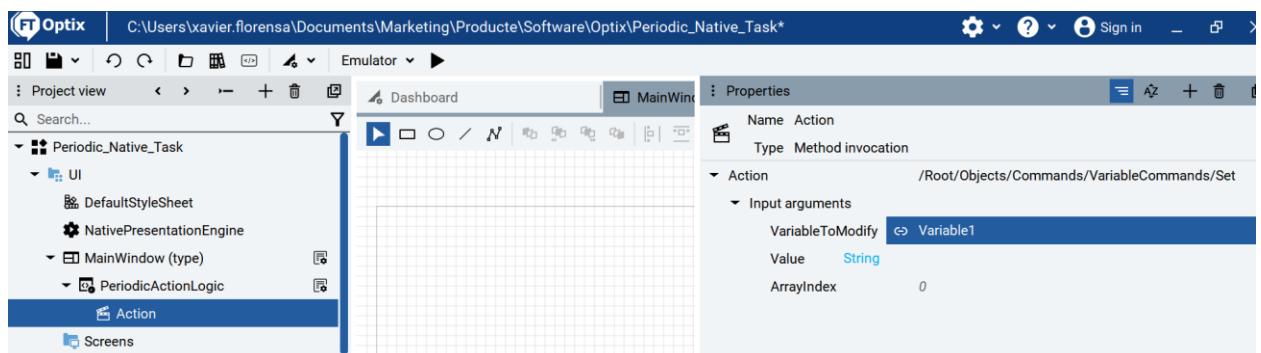
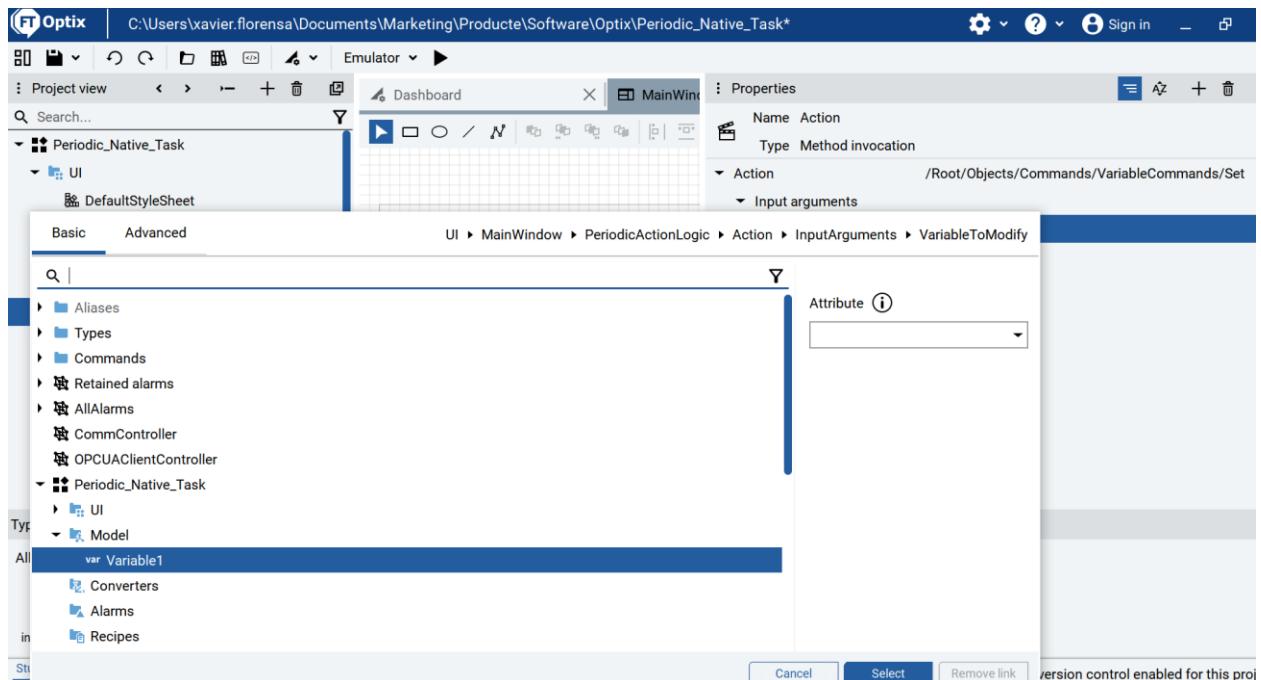
Click on the pen



Add a command



Set VariableToModify



Select the value, for instance, logged user (you have this on users section) or whatever you want.

### 33.2. Netlogic Periodic Tasks

Let's assume we want a function called "my\_Function" task to be executed each 1000 mseconds

We have to use these directives on our NetLogic

```
public class RuntimeNetLogic1 : BaseNetLogic
{
    PeriodicTask myTask1;

    public override void Start()
    {
        myTask1 = new PeriodicTask(my_Function, 1000, LogicObject);
        myTask1.Start();
    }
}
```

```

public override void Stop()
{
    // Insert code to be executed when the user-defined logic is stopped
    myTask1.Dispose();
}

```

And write your code inside this function after the Stop() function

```

private void my_Function()
{
}

```

You can even use input and output parameters

### 34. Log info to emulator output

Just insert this sentence on your code

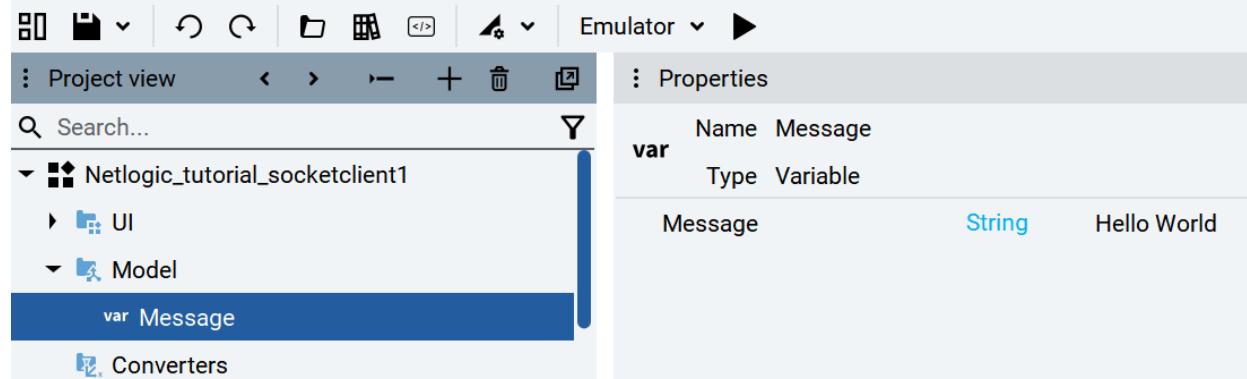
```
Log.Info("A button has been pressed");
```



[https://github.com/FactoryTalk-Optix/NetLogic\\_CheatSheet/blob/main/pages/log-output.md](https://github.com/FactoryTalk-Optix/NetLogic_CheatSheet/blob/main/pages/log-output.md)

### 35. Accessing variables

Let's define a string variable with an initial value.



The simplest way to access variables in our program is once declared the variable "Message", with this sentence

```
var missatge = Project.Current.GetVariable("Model/Message");
```

So we are coping the whole object "Message" variable with all attributes to the Netlogic variable "missatge"

But be careful, and aware that missatge is not a string type, it is an object.

If you want to refer to the value, you have to do this way

`missatge.Value;`

So if you want to copy the value to a string variable you do this way

`String myNewMessage = missatge.Value`

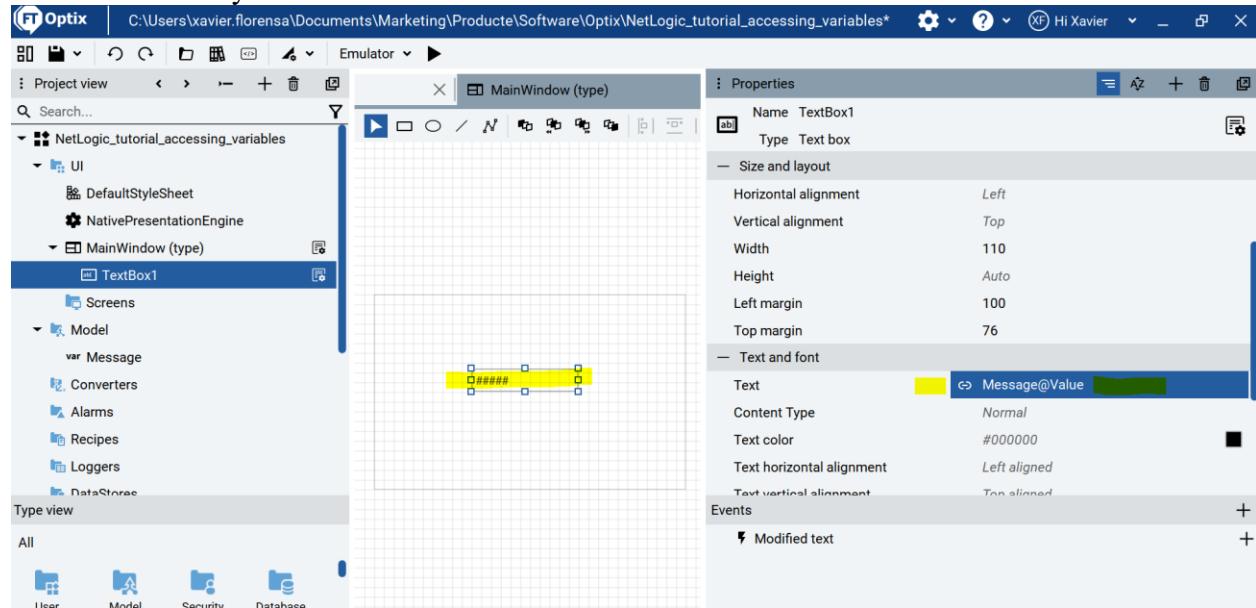
Or if you want to assign a new value in the code, you do this way

`missatge.Value = "Hello, how are you?"`

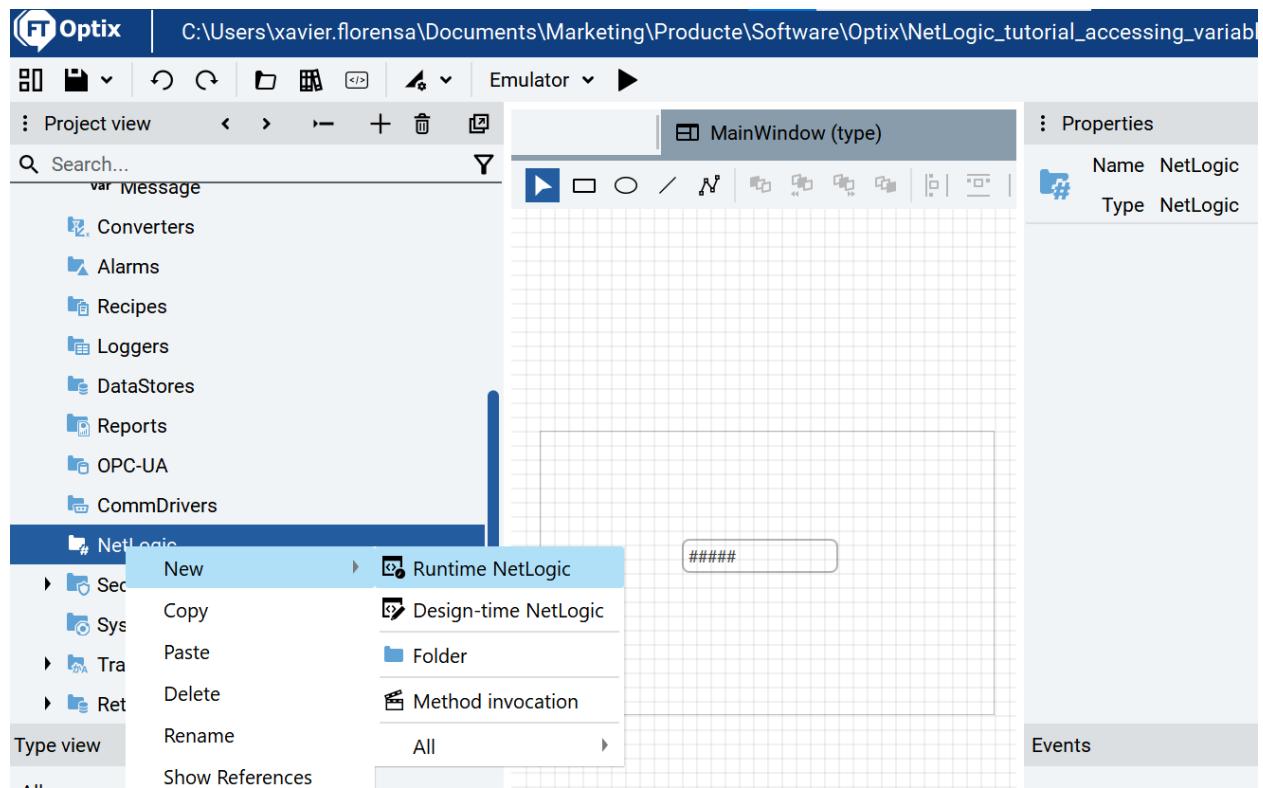
You can output the new code value to a User Interface Textbox with a dynamic link to the variable "Message"

Let's try it with this project

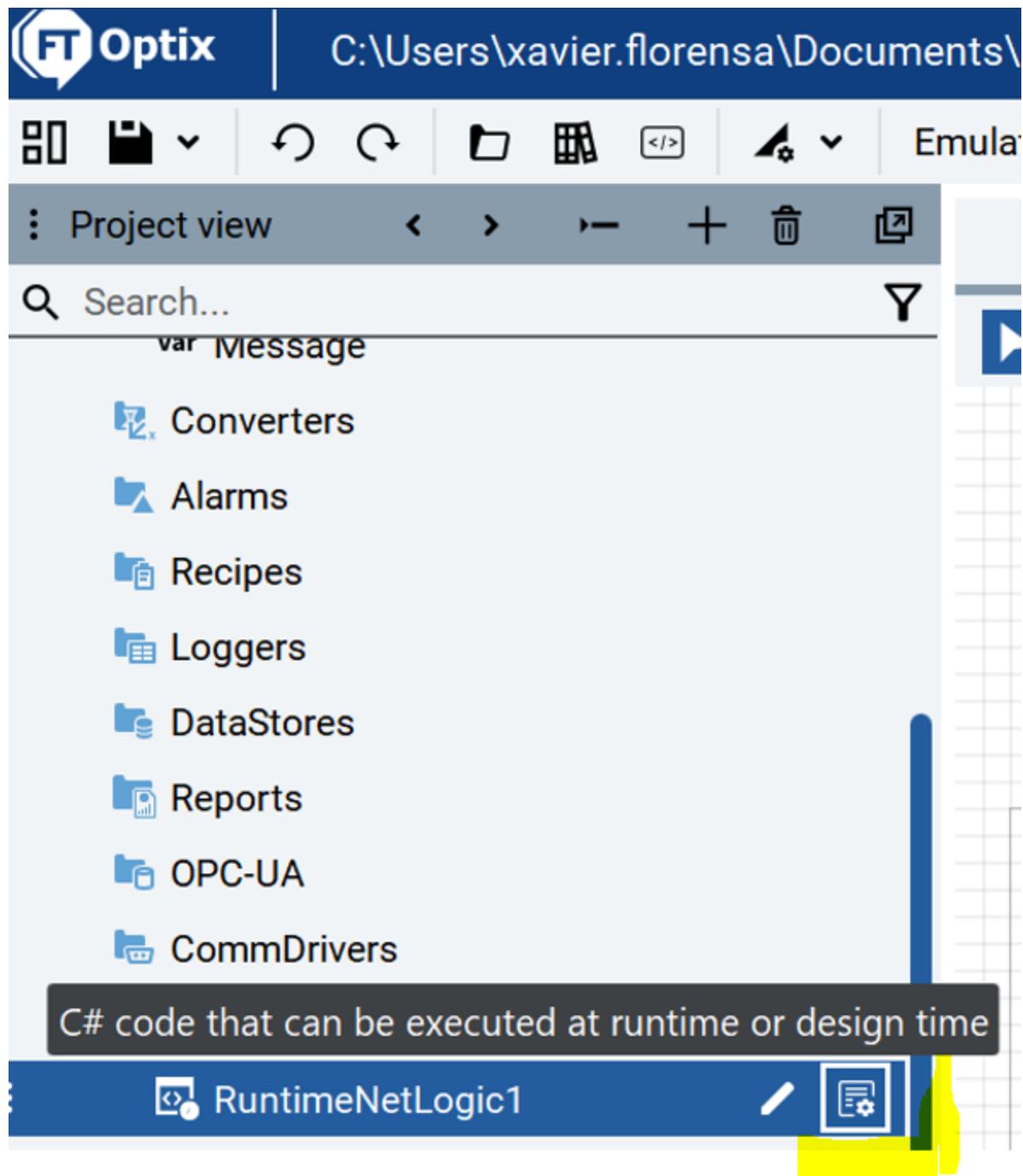
Let's create the dynamic link between a TextBox and a variable value



Now let's create some Netlogic



Click on the icon with textlines and gearbox



The NetLogic code will open

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
```

```

using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}

```

Introduce such code on the Start() routine

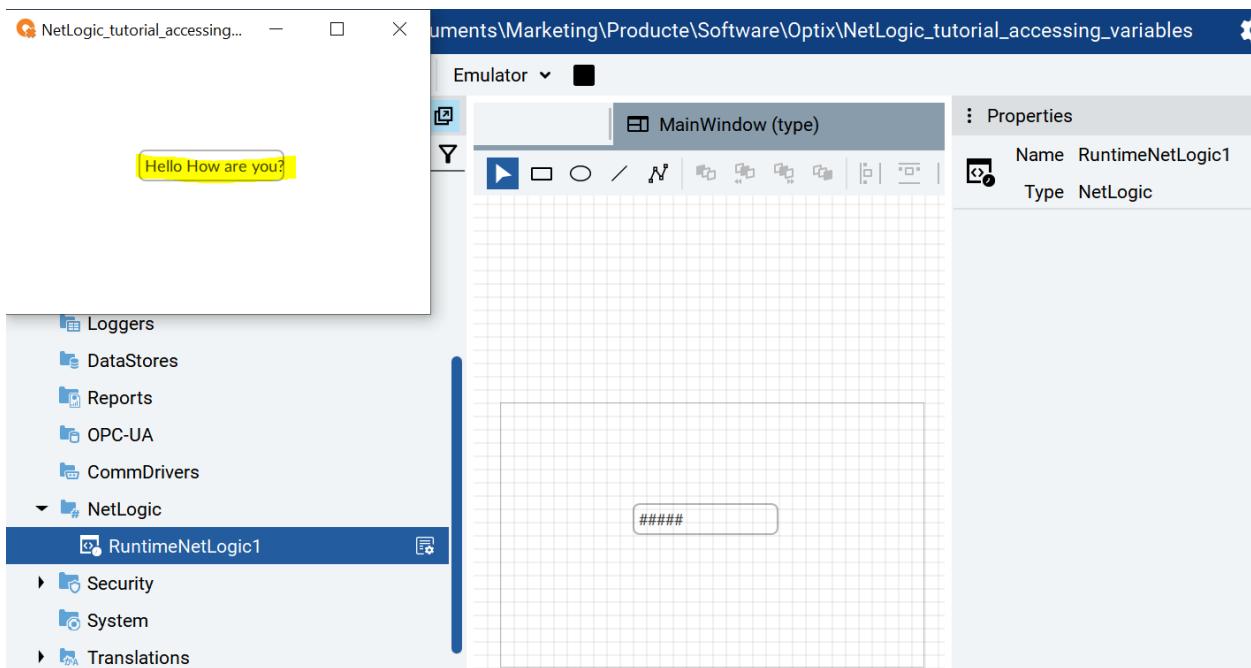
```

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        var missatge = Project.Current.GetVariable("Model/Message");
        missatge.Value="Hello How are you?";
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}

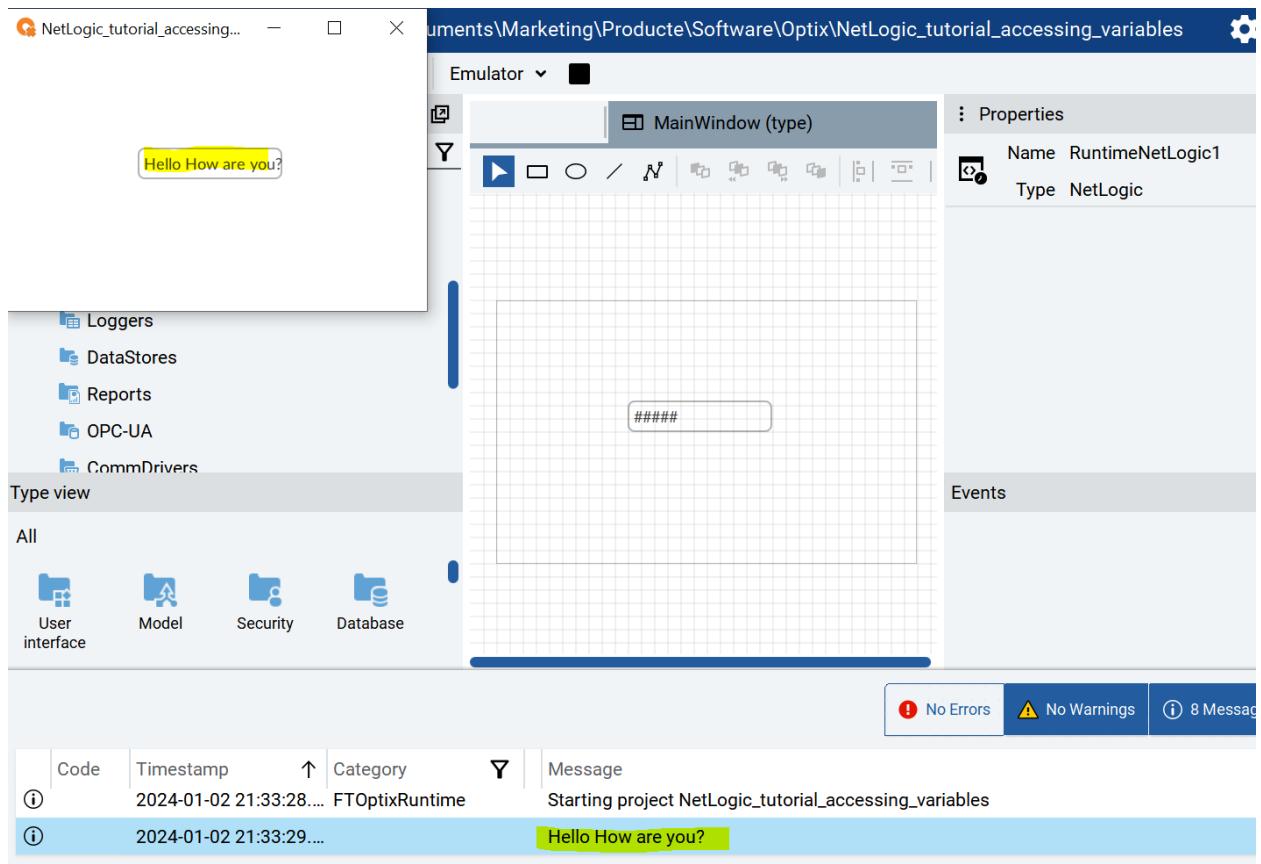
```

Save the code and test the emulator



Now imagine you want to display the value on the Emulator Output. As seen on previous chapter Introduce this line on NetLogic

```
public override void Start()
{
    // Insert code to be executed when the user-defined logic is started
    var missatge = Project.Current.GetVariable("Model/Message");
    missatge.Value="Hello How are you?";
    Log.Info(missatge.Value);
}
```

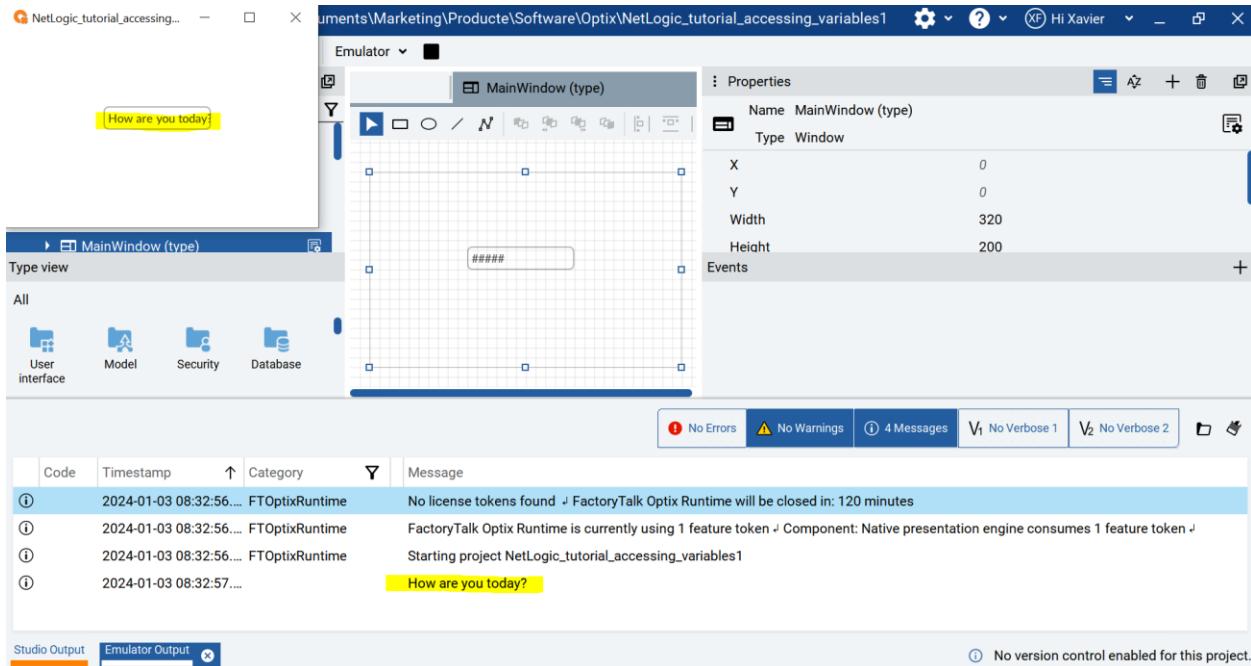


You can even access the UI variable without using another variable in the code, this way

```
public override void Start()
{
    // Insert code to be executed when the user-defined logic is started

    Project.Current.GetVariable("Model/Message").Value = "How are you
today?";

    Log.Info(Project.Current.GetVariable("Model/Message").Value);
}
```



## 36. Accessing objects

You can have access to project objects like labels, texboxes, etc from your code.

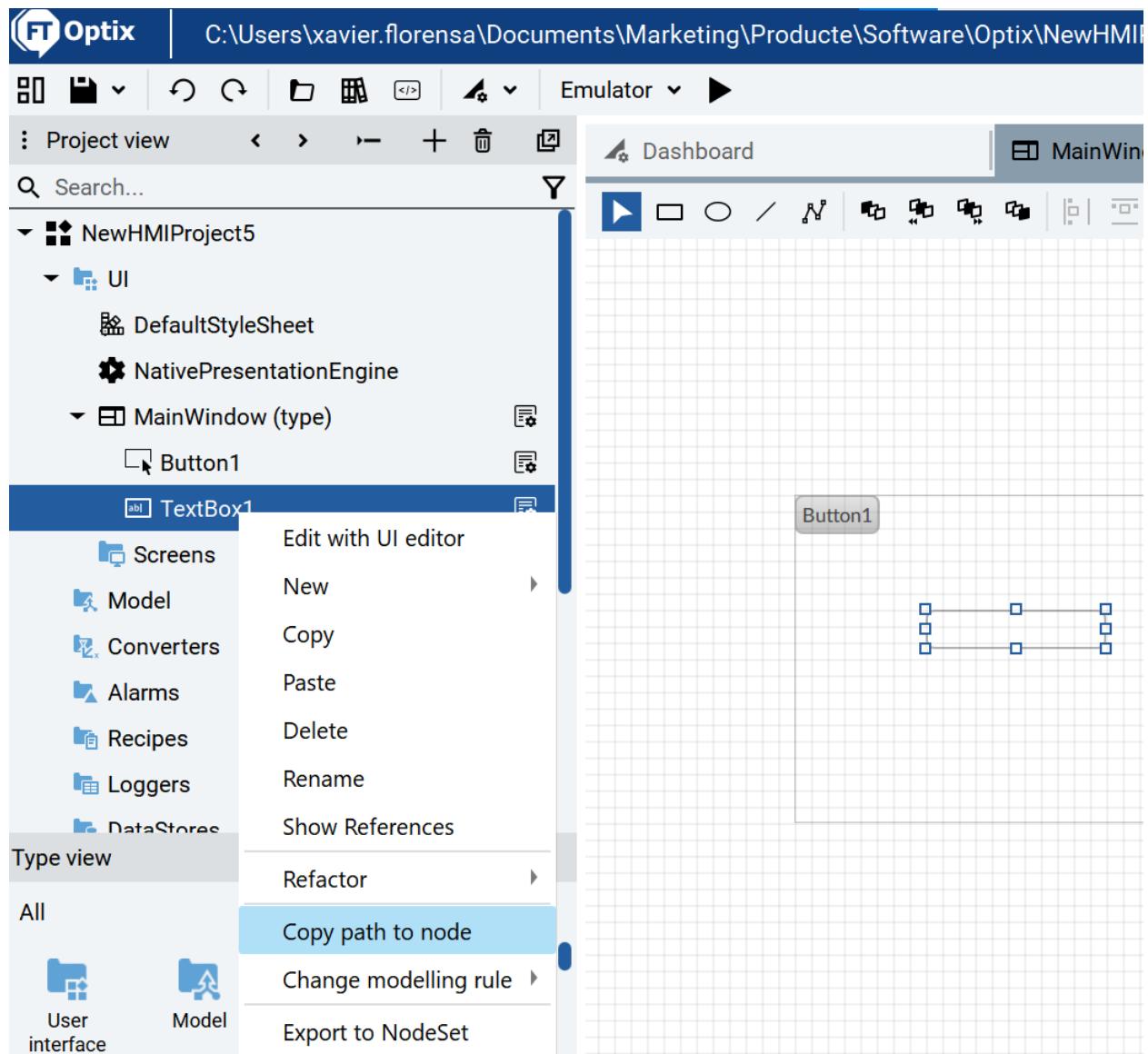
As explained here

<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/netlogic/label-text/Develop-a-button-that-changes-label-text.html>

The path to the NodeId of any object is

<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/api/log/Log-Node.html?hl=path%2Cnodeid>

To see the path to node you have to right click over the object and click on Copy Path to node



NewHMIProject5/UI/MainWindow/TextBox1

But you will not see this unless you select this checkbox on Optix configuration

Language and Locale

English (United States) ▾

New object naming language

English (United States) ▾

Advanced mode

Show feature previews



Preferred code editor

Visual Studio Code ▾

Preferred physical length unit

Automatic ▾

FTOptix Studio log level

Info ▾

Font style

Default ▾

Security warnings

Restore all

Save

Cancel

## Accessing project nodes

In order to manipulate a project element, you first need to access it, this can be either done by getting its `NodeId` or its node object

- Examples:

- `var myNodeId = Project.Current.Get( [path/to/node] ).NodeId;`
  - `var myObject = Project.Current.Get( [path/to/node] );`
  - Where `[path/to/node]` can be obtained by right clicking any element of the IDE and then clicking `Copy path to node`, here you need to remember to remove the project name from the pasted element
- Example:
    - `NewHMIProject25/Model/MyCustomMotor --> Model/MyCustomMotor`

The `Get` method also accepts an input type that will be passed to the variable type

- Example:
  - `var myButton = ... .Get<Button>("path/to/button");` -> Will create a `myButton` variable of Type `FToptix.UI.Button` pointing to the desired element
- After the object has been casted to the proper path, you can access its properties using IntelliSense
  - Example:
    - `myButton.Text = Hello;`

## Log.Node(node, verbose)

Returns a string that contains the `path` of the node passed in the argument. The second argument is optional and inserts the `NodeId` and object type in the returned string.

```
static string Node(IUANode node, bool verbose);
```

### Arguments

#### node (IUANode)

The node for which the `path` is to be generated in a string format.

#### verbose (bool)

`false` (default)

Does not insert additional information in the string returned.

`true`

Inserts the `NodeId` and object type in the string returned.

### Returns ↗

#### string

The `path` of the node passed as the argument. Based on the `verbose` argument value, it can also contain the `NodeId` and the object type.

### Examples ↗

The following example shows an API that generates a message consisting of the "Error on node " string and the `NetLogic` `path`, returned as a string by the `Log.Node` API:

```
Log.Info("Error on node " + Log.Node(LogicObject));
```

Let's define a textbox on UI

The screenshot displays the FT Optix software environment. On the left, the Project view shows a project named "Training\_NetlogicRuntime\_Access\_object" with a "UI" folder containing "DefaultStyleSheet", "NativePresentationEngine", and "MainWindow (type)". The "MainWindow (type)" node is selected, revealing a "TextBox2" component. The main workspace shows a "Dashboard" window with "MainWindow (type)" open, displaying a simple UI with a text box containing the text "hello how are you?". The Properties panel on the right shows the properties for "TextBox2", including:

- Name: TextBox2
- Type: Text box
- Border color: #000000
- Style: Default
- Size and layout:
  - Horizontal alignment: Left
  - Vertical alignment: Top
  - Width: 110
  - Height: Auto
  - Left margin: 150
  - Top margin: 110
- Text and font:
  - Text: (highlighted in yellow)
  - Content Type: Normal

Below the UI preview, the code editor shows the C# code for the "RuntimeNetLogic1" class:

```

#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        var myTextbox = Project.Current.Get<TextBox>("UI/MainWindow/TextBox2");
        myTextbox.Text="hello how are you?";
    }
}

```

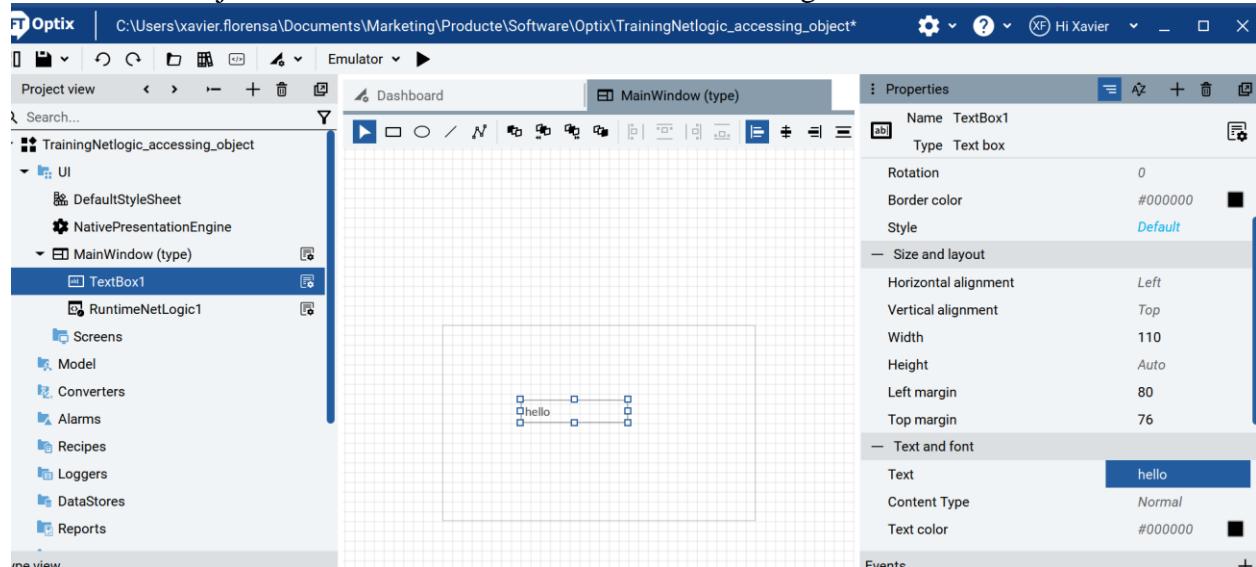
```

    public override void Stop()
    {
    }
}

```

You can also access an object this way

Build a new Project with a Textbox and a new Runtime Netlogic inside the MainWindow



Open code and insert this code

```

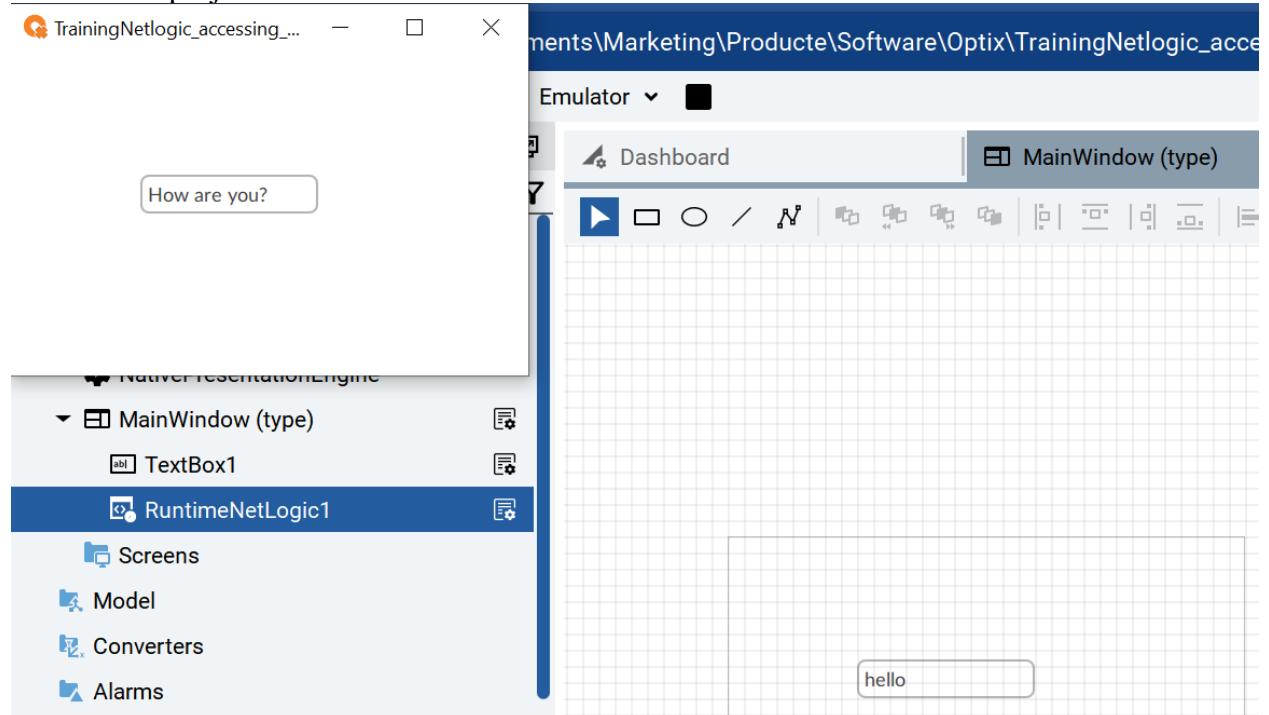
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrjject;
using FTOptix.Rettentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        var textbox1 = Owner.Get<TextBox>("TextBox1");
        textbox1.Text="How are you?";
    }
}

```

```
public override void Stop()
{
    // Insert code to be executed when the user-defined logic is stopped
}
}
```

### Execute the project



### Detecting a change on an object

# IUAObject.UAEEvent

This event occurs when the project object to which the `IUAObject` C# object refers generates any OPC UA event.

```
event EventHandler<UAEEventArgs> UAEEvent;
```

✓ Event handler 

```
public delegate void UAEEvent(object sender, UAEEventArgs e);
```

✓ Event handler arguments 

`sender` (object)

A C# object that corresponds to the object of the project origin of the event.

`e` (UAEEventArgs)

A C# object that contains the following properties:

`EventType` (IUAObjectType)

The node of the type of event generated.

`Arguments` (UAEEventArgsList)

A C# object that contains the arguments of the generated event.

## Example

```
public override void Start()
{
    var button1 = Owner.Get<Button>("Button1");
    button1.UAEEvent += Button1_UAEEvent;
}

private void Button1_UAEEvent(object sender, UAEEventArgs e)
{
    var label1 = Owner.Get<Label>("Label1");
    var button1 = (Button)sender;
    label1.Text = "Event on " + button1.BrowseName + " of type " + e.EventType.BrowseName +
    ", ";
}
```

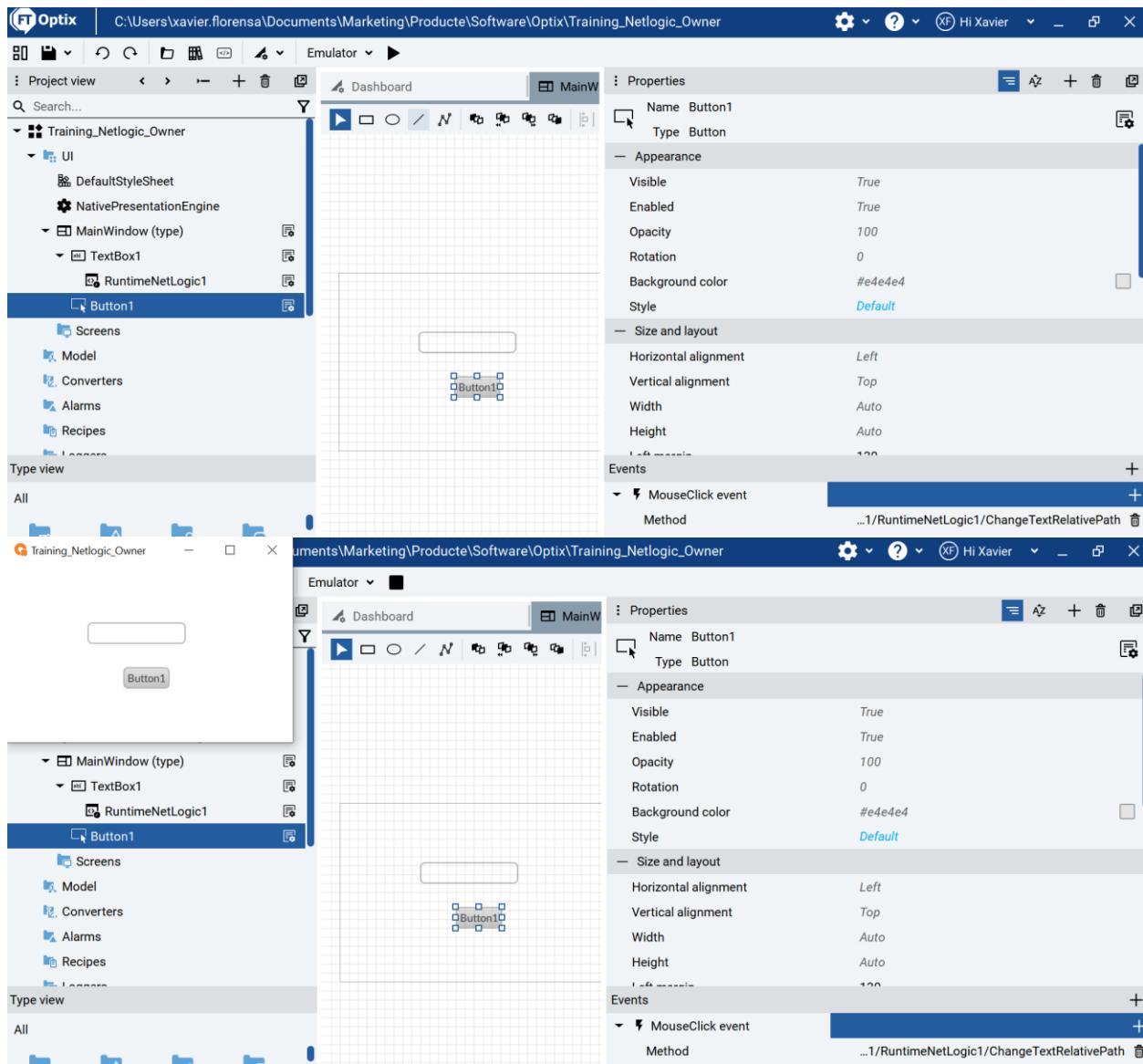
Create a project with a textbox

## 37. Using Owner

You can access object properties if you insert RuntimeNetlogic just inside an object

### 37.1. Accessing object using owner

You have to build the solution in order to find the routine



```

#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Retentivity;
using FTOptix.NetLogic;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }

    [ExportMethod]
    public void ChangeTextRelativePath()
    {
        // WORKING AT RUNTIME!
        Owner.GetVariable("Text").Value = "Text changed using Owner";
    }
}

```

```
    }  
}
```

You can also write on the Text field just after starting the program (without clicking on the button)

```
#region Using directives  
using System;  
using UAManagedCore;  
using OpcUa = UAManagedCore.OpcUa;  
using FTOptix.HMIPrject;  
using FTOptix.Rentativity;  
using FTOptix.NetLogic;  
using FTOptix.UI;  
using FTOptix.NativeUI;  
using FTOptix.CoreBase;  
using FTOptix.Core;  
#endregion  
  
public class RuntimeNetLogic1 : BaseNetLogic  
{  
    public override void Start()  
    {  
        // Insert code to be executed when the user-defined logic is started  
        Owner.GetVariable("Text").Value = "Text changed using Owner";  
    }  
  
    public override void Stop()  
    {  
        // Insert code to be executed when the user-defined logic is stopped  
    }  
  
    [ExportMethod]  
    public void ChangeTextRelativePath()  
    {  
        // WORKING AT RUNTIME!  
        //Owner.GetVariable("Text").Value = "Text changed using Owner";  
    }  
}
```

### 37.2. Detecting change on object using Owner

Create a Textbutton and insert a Runtime Netlogic on it

The screenshot shows the FT Optix software interface. On the left, the Project view displays a project named "TrainingNetlogic\_Owner1" with a "UI" folder containing "DefaultStyleSheet", "NativePresentationEngine", and "MainWindow (type)". The Type view shows "Screens" and categories like "User interface", "Model", "Security", and "Database". In the center, the Dashboard shows a "MainWindow (type)" window with a "Hello" text box containing the text "Hello". The Properties panel on the right shows the properties for "TextBox1": Name (TextBox1), Type (Text box), Top margin (0), Text (Hello), Content Type (Normal), Text color (#000000), Text horizontal alignment (Left aligned), Text vertical alignment (Top aligned), and Events (Modified text). Below the dashboard, a code editor window displays the following C# code:

```

#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

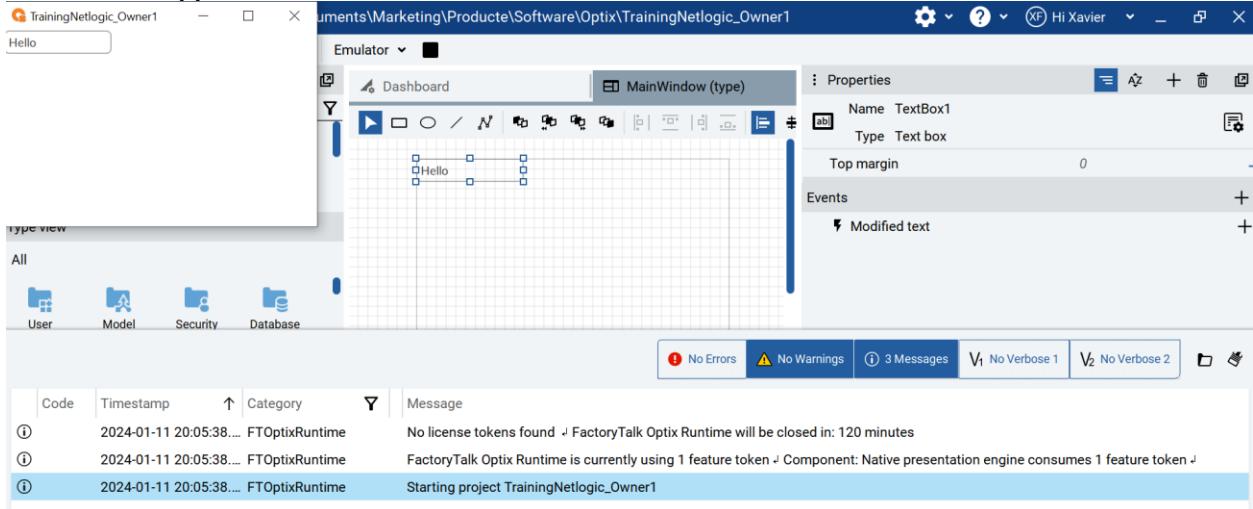
public class RuntimeNetLogic1 : BaseNetLogic
{
    private IUAVariable myVar;
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        myVar = ((TextBox)Owner).TextVariable;
        myVar.VariableChange += MyVar_VariableChange;
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
    private void MyVar_VariableChange(object sender, VariableChangeEventArgs e)
    {
        // Log some information
        Log.Info("Old value: " + e.OldValue.ToString());
        Log.Info("New value: " + e.NewValue.ToString());
    }
}

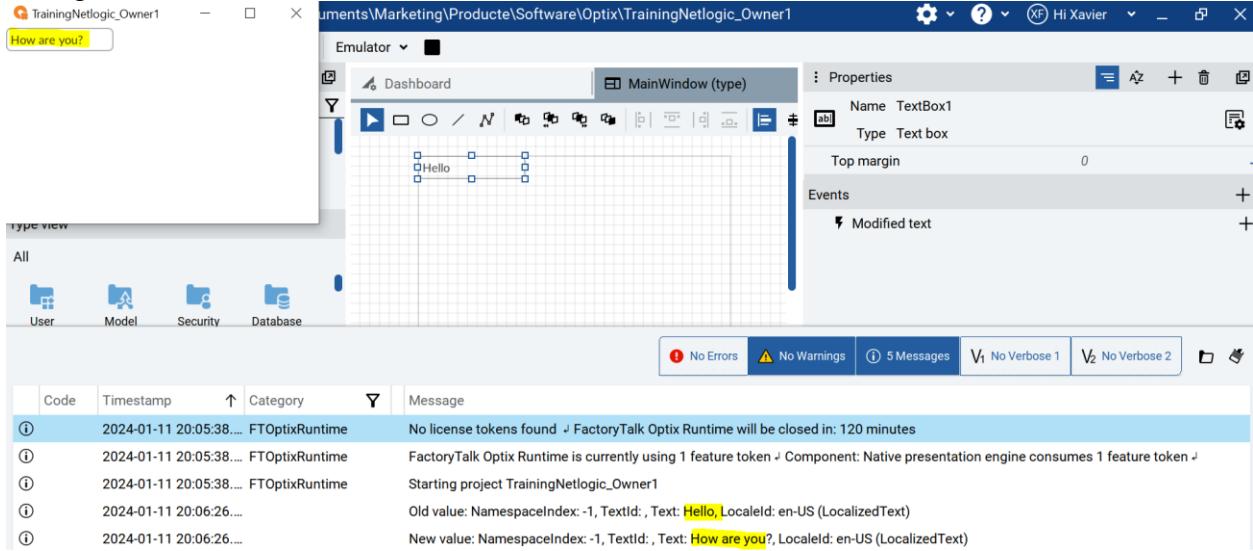
```



## Execute the application

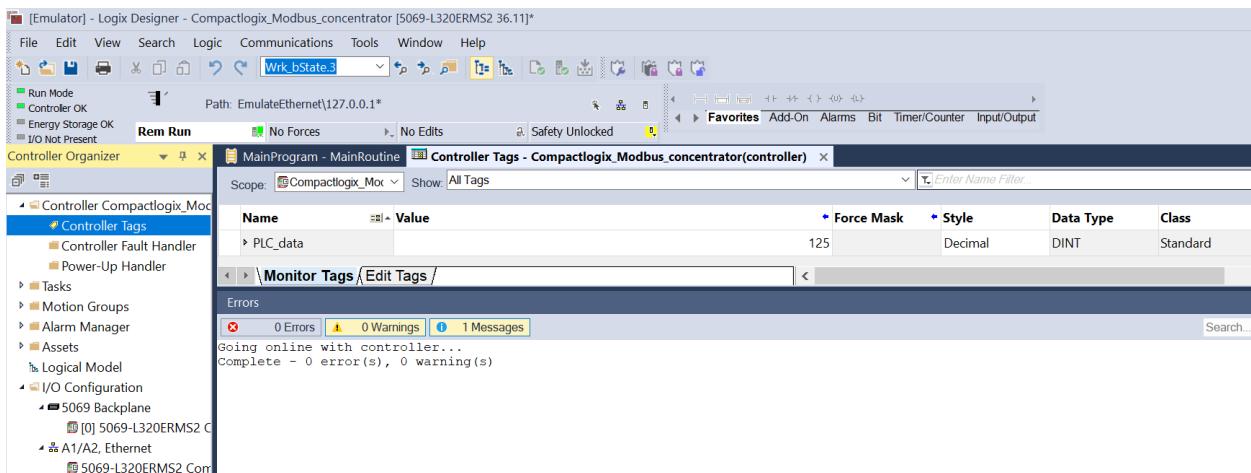


## Change the text and hit enter

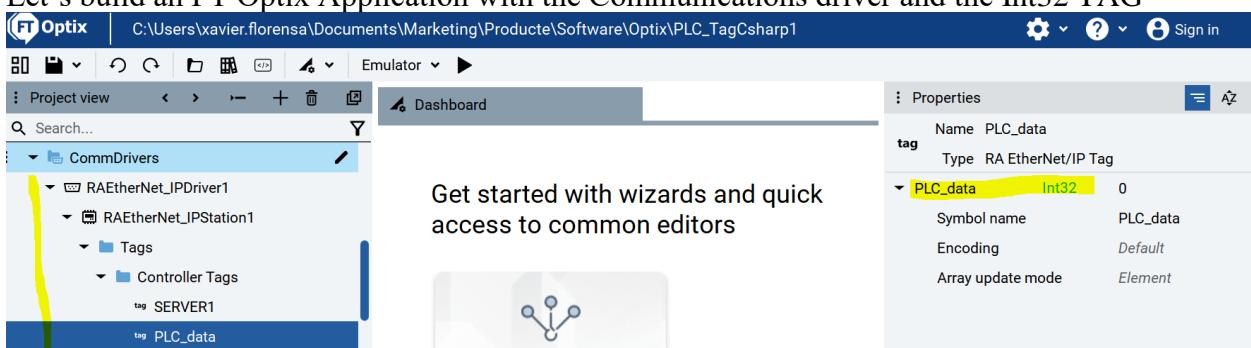


## 38. Accessing PLC Tags

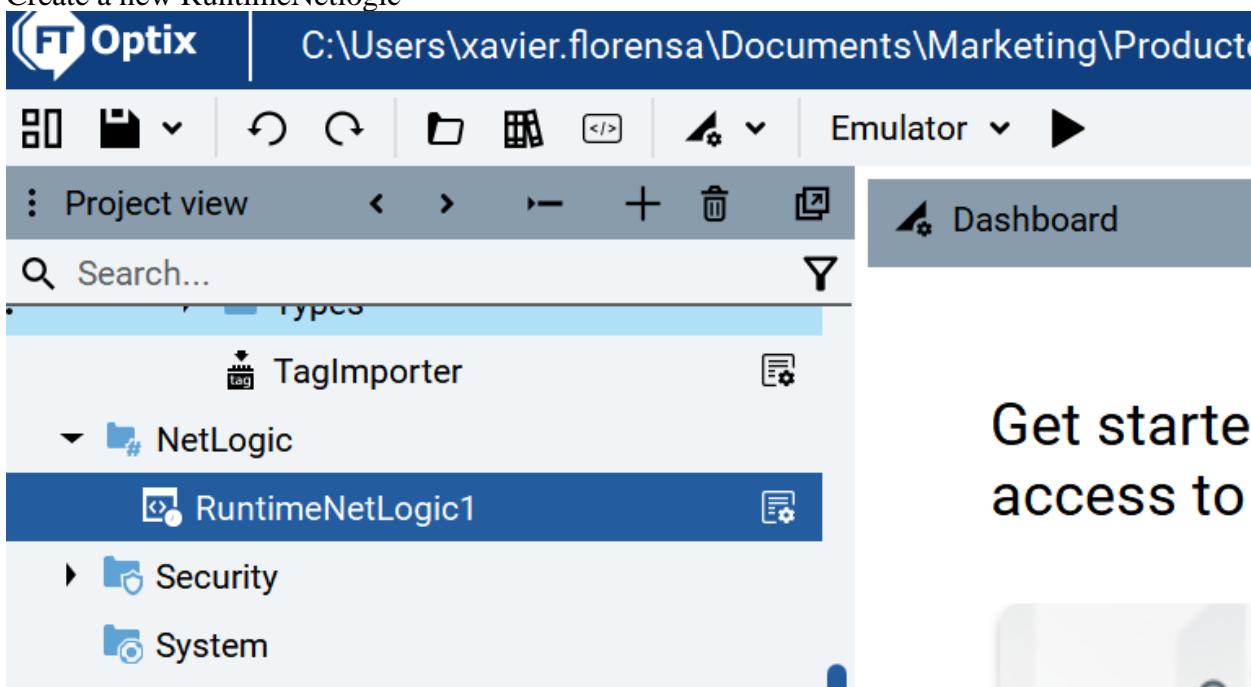
Let's build a PLC program with a DINT Tag, download and go online with our FactoryTalk Logix Echo emulator



Let's build an FT Optix Application with the Communications driver and the Int32 TAG



Create a new RuntimeNetlogic



### 38.1. Method0

This way

<https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/developing-solutions/developing-projects-with-csharp/csharp-apis-reference/read-or-write-field-variables/iuavariable-remotewrite.html>



FactoryTalk Optix Studio Help Center

- > Getting started
- > Using the software
- ▽ Developing solutions
  - ▽ Developing projects with C#
    - > NetLogic
    - > C# project nodes
    - > Methods and events in C#
    - > Sessions and users in C#
    - > C# APIs reference
      - > Return known nodes
      - > Return nodes by searching
      - > Add or remove nodes
      - > Create objects
      - > Create variables
    - > Read or write field variables

IUAVariable.RemoteWrite(value, timeoutMilliseconds)

Writes the value passed in the first argument in the variable on which it invokes. The second optional argument sets the timeout period.

```
void RemoteWrite(UAValue value, double timeoutMilliseconds);
```

**Arguments**

**value (UAValue)**  
The value to write.

**timeoutMilliseconds (double)**  
The timeout period, expressed in milliseconds, after which the API throws an exception.

**TIP:** If not specified, the default value of the argument is 30000 (30 seconds).

**Example**

In the following example, the 42 value is written in the Speed variable.

Use this code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using FTOptix.RAEtherNetIP;
using FTOptix.CommunicationDriver;
using System.Collections.Generic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        var currentSpeed =
Project.Current.GetVariable("CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/Tags/Controller Tags/PLC_data");
        currentSpeed.RemoteWrite(42);
```

```

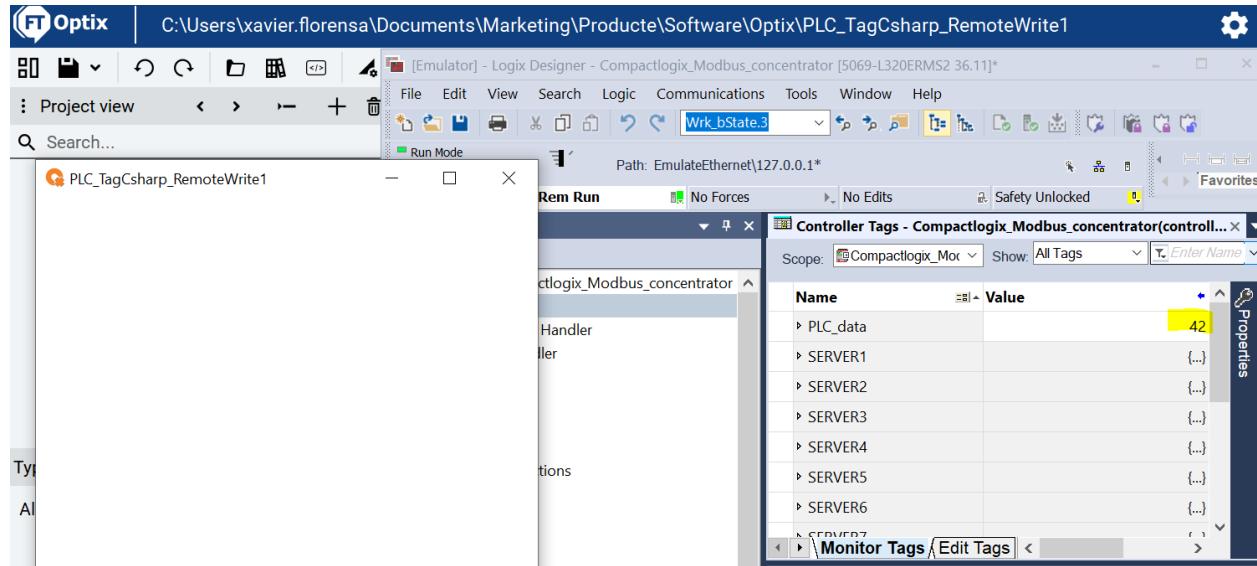
        }

        public override void Stop()
        {
            // Insert code to be executed when the user-defined logic is stopped
        }
    }
}

```

Execute the application

Voilà



### 38.2. Method1

This way

<https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/developing-solutions/developing-projects-with-csharp/csharp-apis-reference/read-or-write-field-variables/informationmodel-remotewrite.html>

**FactoryTalk Optix Studio Help Center**

- > Getting started
- > Using the software
- ▽ Developing solutions
  - ▽ Developing projects with C#
    - > NetLogic
    - > C# project nodes
    - > Methods and events in C#
    - > Sessions and users in C#
  - ▽ C# APIs reference
    - > Return known nodes
    - > Return nodes by searching
    - > Add or remove nodes
    - > Create objects
    - > Create variables

## InformationModel.RemoteWrite(variableValues, timeoutMilliseconds)

Writes the values in the variables on interest. The second optional argument sets the timeout period.

```
static void RemoteWrite(IEnumerable<RemoteVariableValue> variableValues, double timeoutMilliseconds);
```

### Arguments

**variableValues (IEnumerable<RemoteVariable>)**

The list of the variable values that you want to write, expressed as a pair of the `RemoteVariableValue` class following properties:  
**Variable(IUAVariable)**  
 The variable.

**Value(IUAValue)**

The value of the variable.

**timeoutMilliseconds (double)**


Use this code on the NetLogic

Do not forget to use the following directive

```
using System.Collections.Generic;
```

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using FTOptix.RAEtherNetIP;
using FTOptix.CommunicationDriver;
using System.Collections.Generic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started

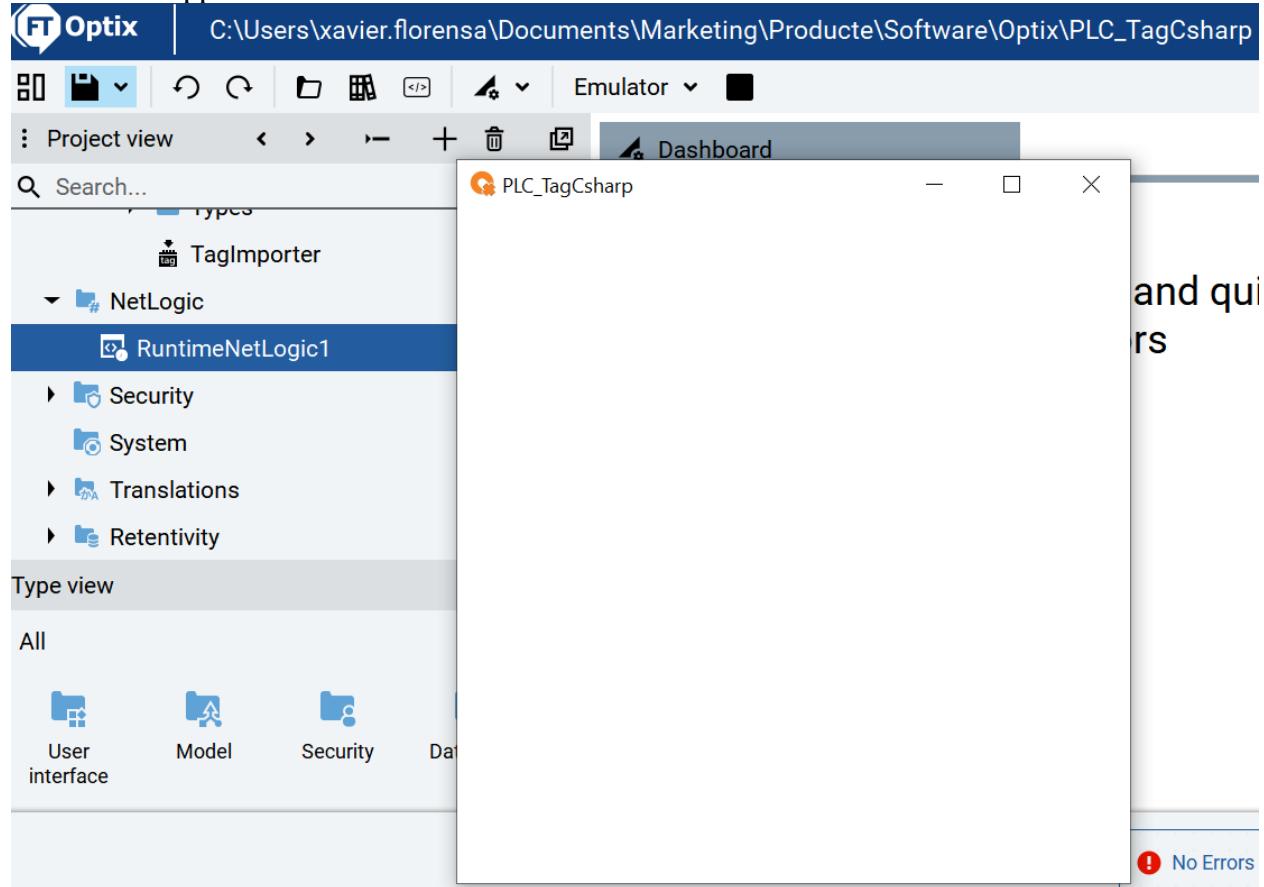
        var tag1 =
Project.Current.Get<FTOptix.RAEtherNetIP.Tag>("CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/Tags/Controller Tags/PLC_data");
        var remoteVariableValues = new List<RemoteVariableValue>()
    }
}
```

```
        new RemoteVariableValue(tag1, 125)
    };
    InformationModel.RemoteWrite(remoteVariableValues);
}

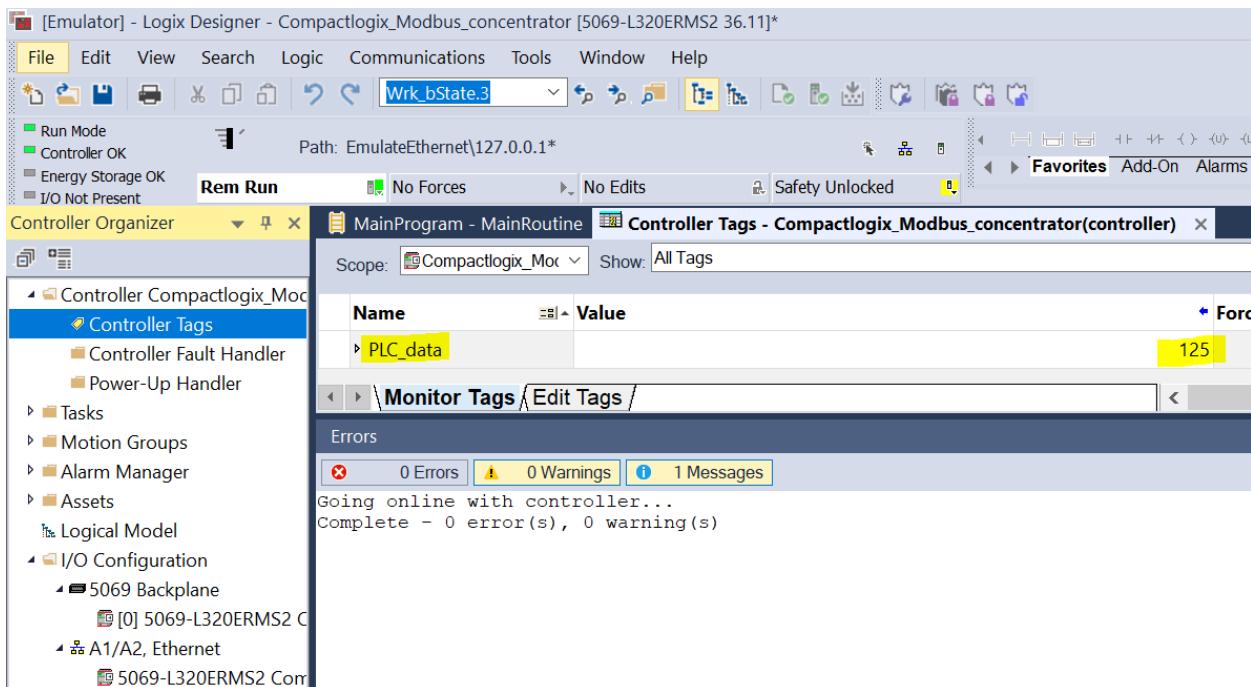
public override void Stop()
{
    // Insert code to be executed when the user-defined logic is stopped
}

}
```

Execute the application



And voilà



### 38.3. Method 2

This way

<https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/developing-solutions/developing-projects-with-csharp/csharp-apis-reference/read-or-write-field-variables/iuanode-childrenremotewrite.html>



FactoryTalk Optix Studio Help Center

- > Getting started
- > Using the software
- ▽ Developing solutions
  - ▽ Developing projects with C#
  - NetLogic
  - C# project nodes
  - Methods and events in C#
  - Sessions and users in C#
- ▽ C# APIs reference
  - Return known nodes
  - Return nodes by searching
  - Add or remove nodes
  - Create objects
  - Create variables

## IUANode.ChildrenRemoteWrite(childVariableValues, timeoutMilliseconds)

Writes the values of the variables of interest children of the node on which it invokes. The optional argument sets a timeout period.

```
void ChildrenRemoteWrite(IEnumerable<RemoteChildVariableValue> childVariableValues, double timeoutMilliseconds);
```

### Arguments

**childVariableValues (IEnumerable<RemoteChildVariableValue>)**  
The list of the variables with the value you want to write, expressed as a pair of the `RemoteVariableValue` class following properties:  
**Variable (IUAVariable)**  
The variable.  
**Value (UAValue)**  
The value of the variable.  
**timeoutMilliseconds (double)**

Use this code on the NetLogic

Do not forget to use the following directive

```
using System.Collections.Generic;
```

This is the code

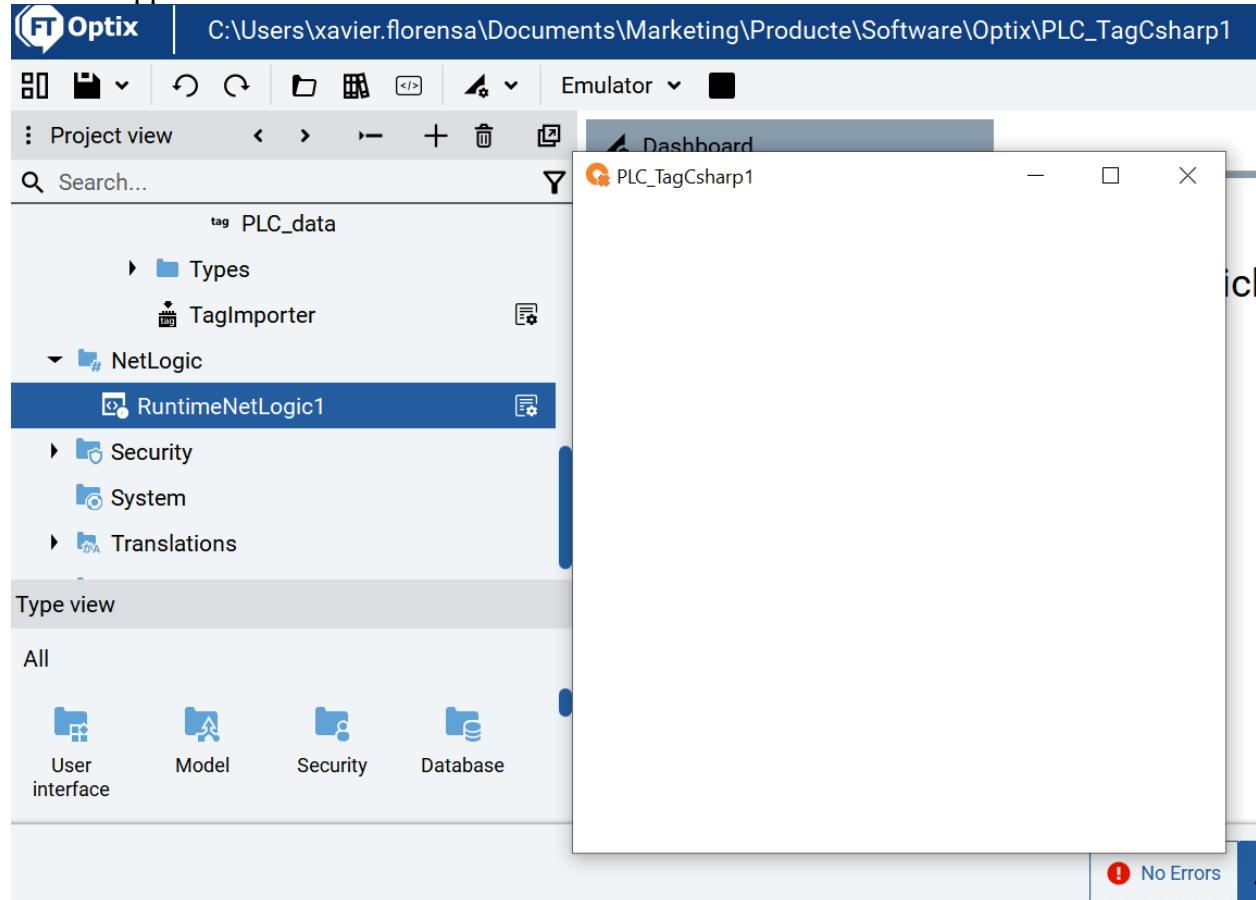
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using FTOptix.RAEtherNetIP;
using FTOptix.CommunicationDriver;
using System.Collections.Generic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        var valuesToWrite = new List<RemoteChildVariableValue>()
        {
            new RemoteChildVariableValue("PLC_data", 4),
            ...
        };
        var myNode =
Project.Current.Get("CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/Tags/
Controller Tags");
        try
        {
            myNode.ChildrenRemoteWrite(valuesToWrite);
        }
        catch (Exception ex)
        {
            Log.Error("ChildrenRemoteWrite failed: " + ex.ToString());
        }
    }

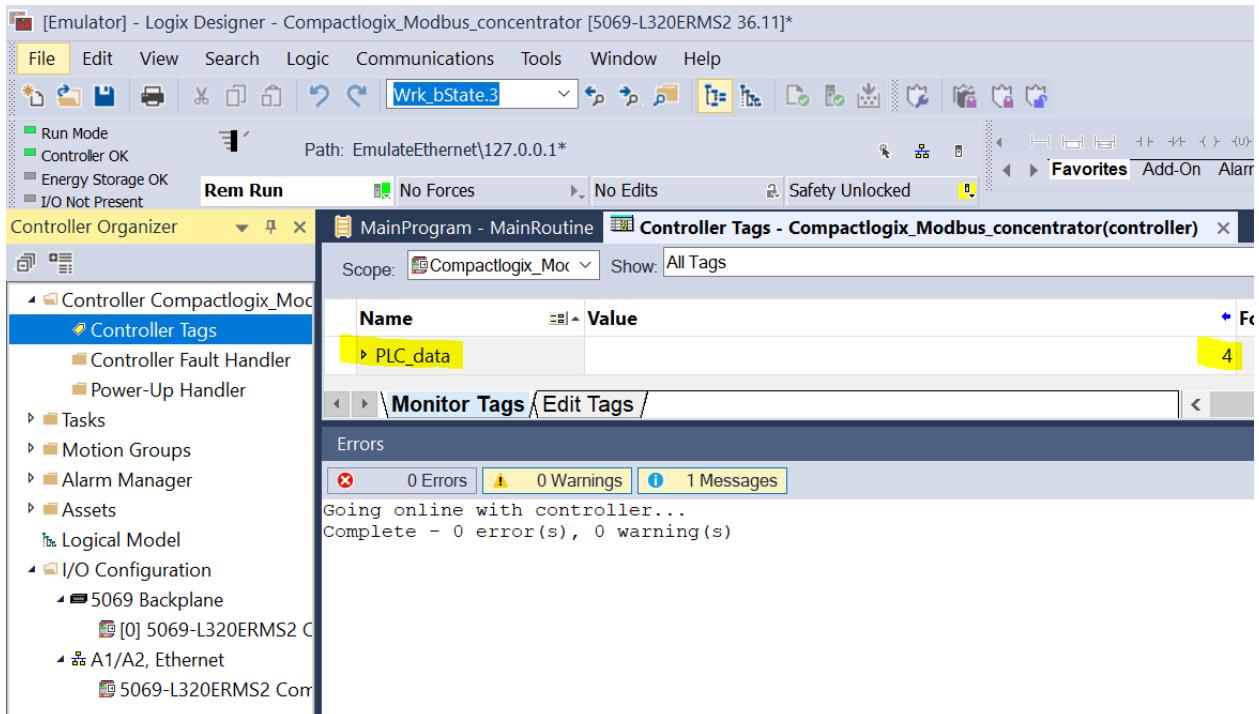
    public override void Stop()
```

```
{  
    // Insert code to be executed when the user-defined logic is stopped  
}  
}
```

Run the application



And voilà



### 38.4. Writing to an array of PLC values

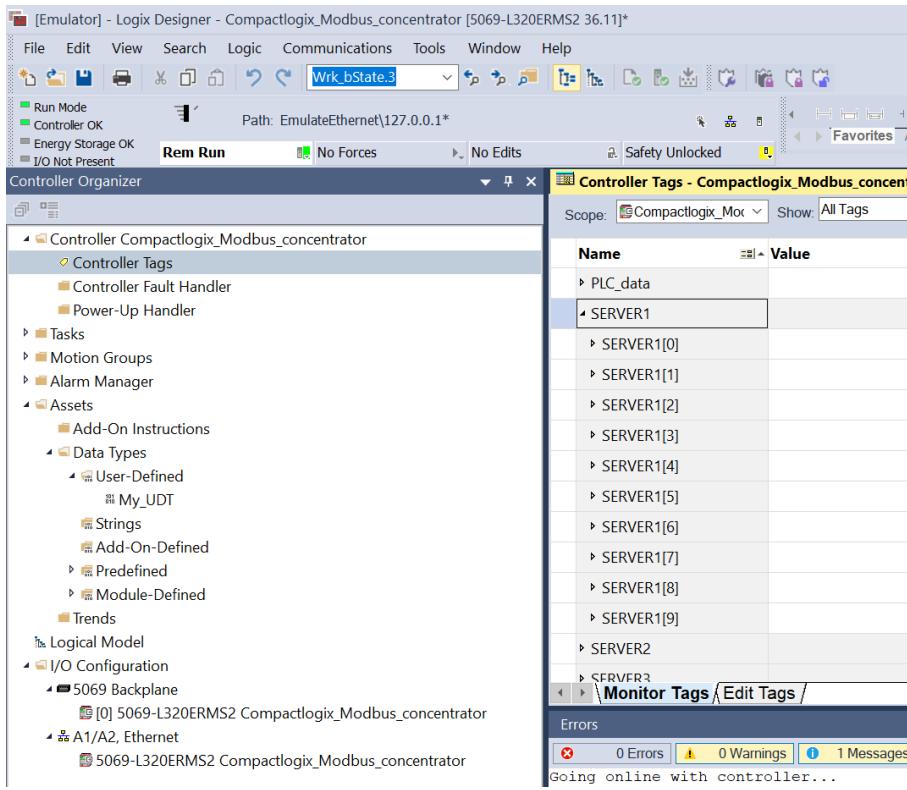
We will use this example, adapted to our needs.

[https://github.com/FactoryTalk-Optix/NetLogic\\_CheatSheet/blob/main/pages/variables-interaction.md](https://github.com/FactoryTalk-Optix/NetLogic_CheatSheet/blob/main/pages/variables-interaction.md)

#### Access mono-dimensional arrays

```
// Get the model variable
var MyVar = Project.Current.GetVariable("Model/Variable");
// Create a C# variable to manipulate values
var tempVar = (float[])MyVar.Value.Value;
// Set the elements of the C# variable
tempVar[0] = (float)Project.Current.GetVariable("CommDrivers/EthernetIPDriver1/EthernetIPStation1/TagName").Value;
// Write the new value(s) to the model variable
MyVar.SetValue(tempVar);
```

Let's Build a PLC program with an array of integers



Let's create a new EtherNet/IP driver in FTOptix to access PLC variables

Properties

- Name: SERVER1
- Type: RA EtherNet/IP Tag
- Symbol name: SERVER1
- Encoding: Default
- Array update mode: Element

Now create a Runtime NetLogic

And let's use this code

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
```

```

using FTOptix.RAEtherNetIP;
using FTOptix.CommunicationDriver;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        Log.Info("Starting application Xavier");
        var currentSpeed =
Project.Current.GetVariable("CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/Tags/Controller Tags/PLC_data");
        var SERVER1 =
Project.Current.GetVariable("CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/Tags/Controller Tags/SERVER1");

        var arraySERVER1 = (Int32[])SERVER1.Value.Value;
        arraySERVER1[0]=25;
        arraySERVER1[1]=26;
        arraySERVER1[2]=27;
        arraySERVER1[3]=28;
        arraySERVER1[4]=29;
        arraySERVER1[5]=30;
        arraySERVER1[6]=31;
        arraySERVER1[7]=32;
        arraySERVER1[8]=33;
        arraySERVER1[9]=34;
        SERVER1.SetValue(arraySERVER1);

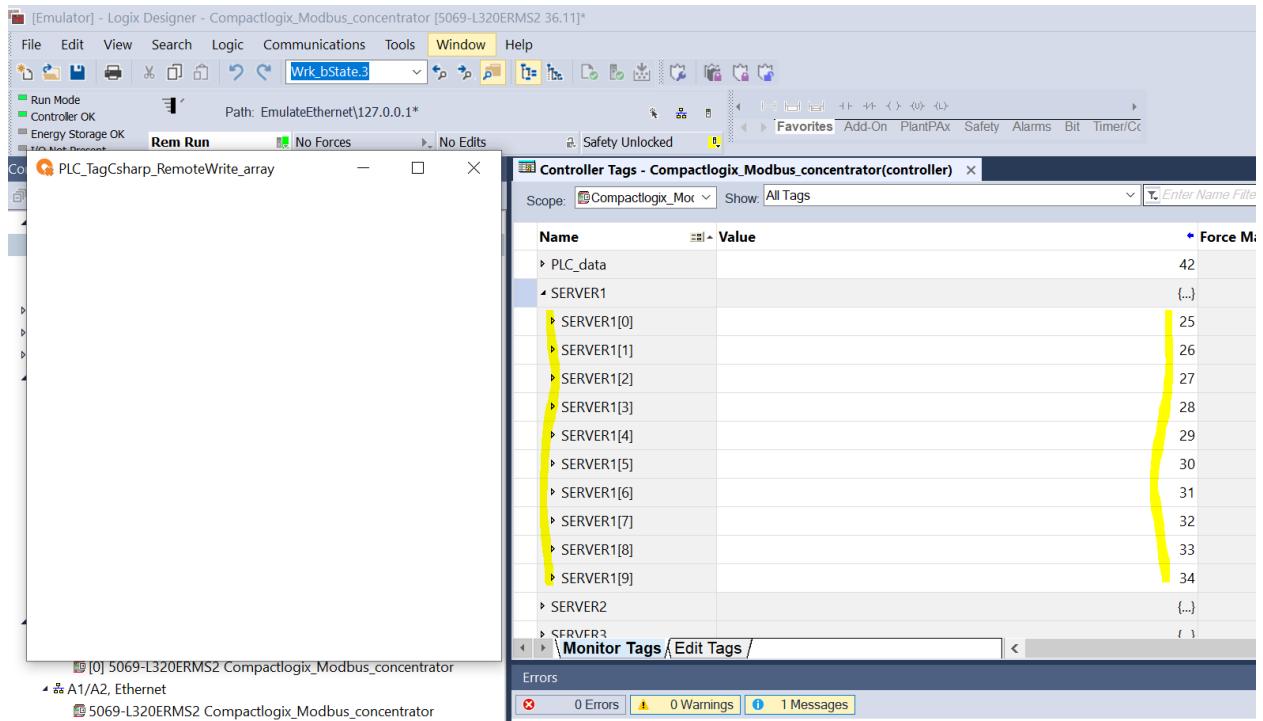
        currentSpeed.RemoteWrite(42);

        Log.Info("SERVER1 INDEX 0 VALUE" + arraySERVER1[0]);
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}

```

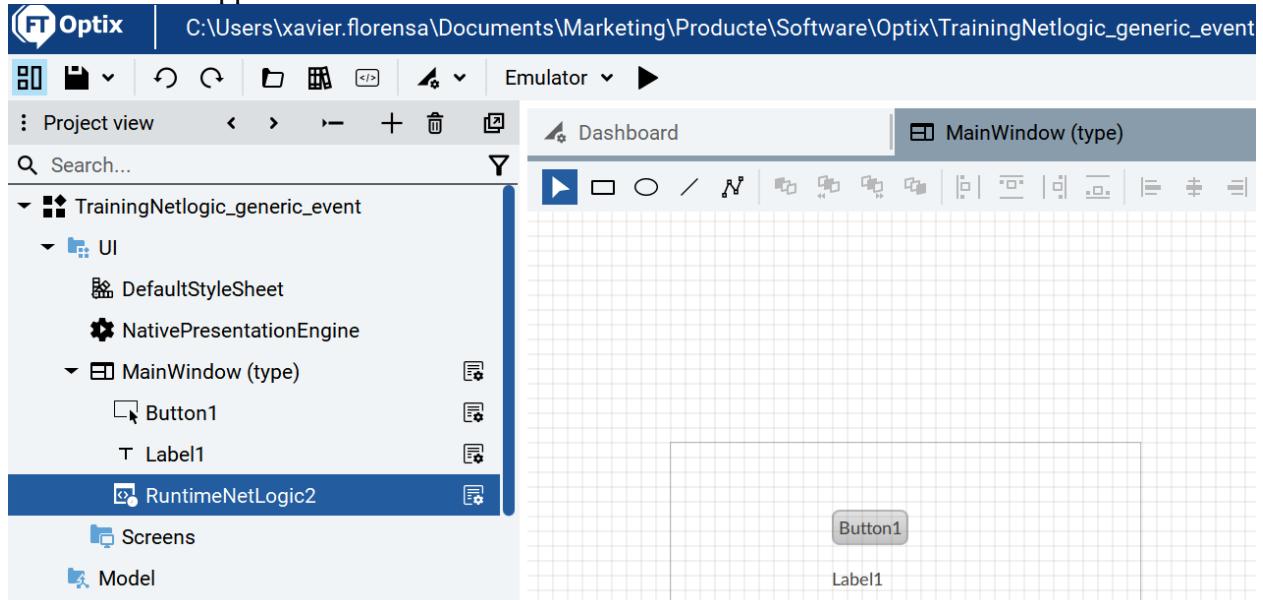
Run the application and ...  
voilà



## 39. Object events

You can also detect a change this way

Let's create an application with a Button and a label



Create a new RuntimeNetLogic under MainWindow

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
```

```

using FTOptix.HMIPrj;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

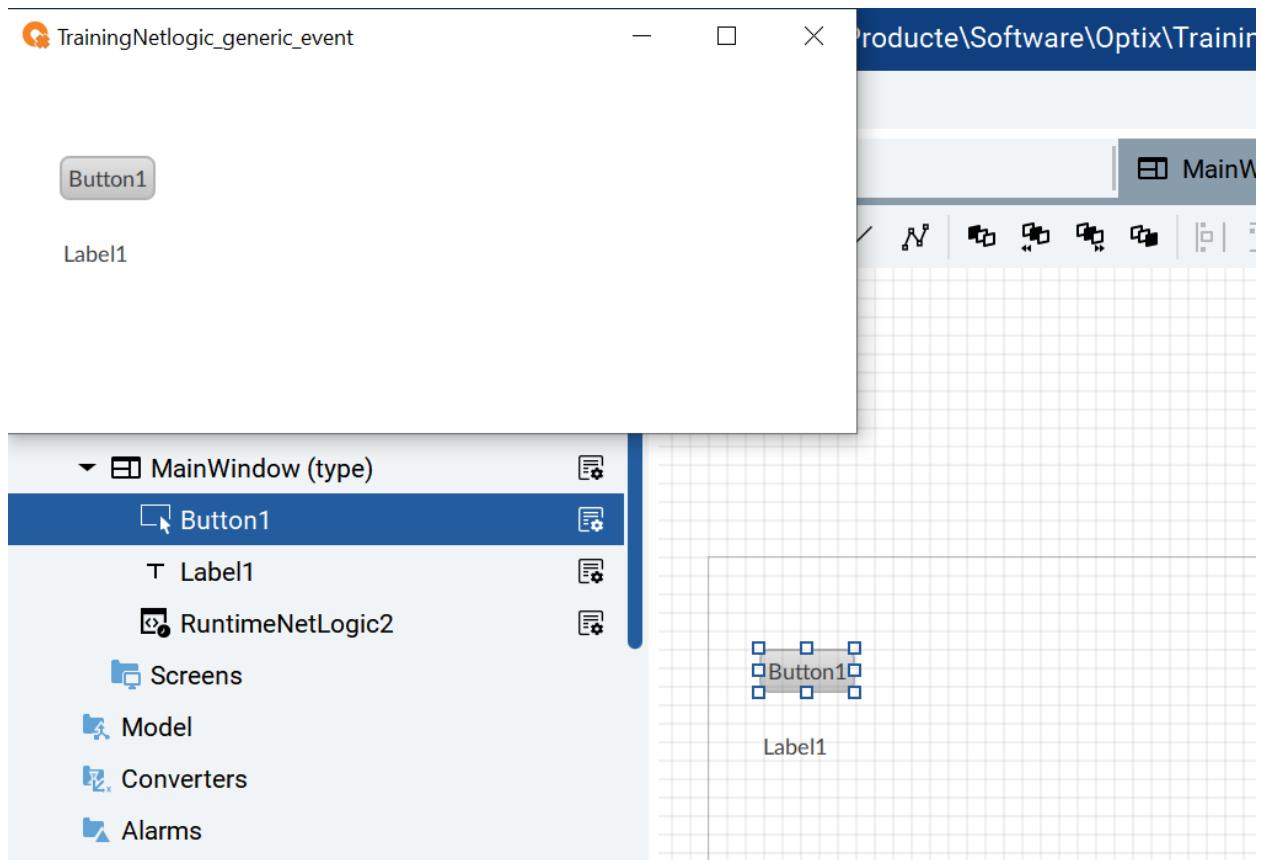
public class RuntimeNetLogic2 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        var button1 = Owner.Get<Button>("Button1");
        button1.UAEEvent += Button1_UAEEvent;
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
    private void Button1_UAEEvent(object sender, UAEEventArgs e)
    {
        var label1 = Owner.Get<Label>("Label1");
        var button1 = (Button)sender;
        label1.Text = "Event on " + button1.BrowseName + " of type " +
e.EventType.BrowseName + " , ";
    }
}

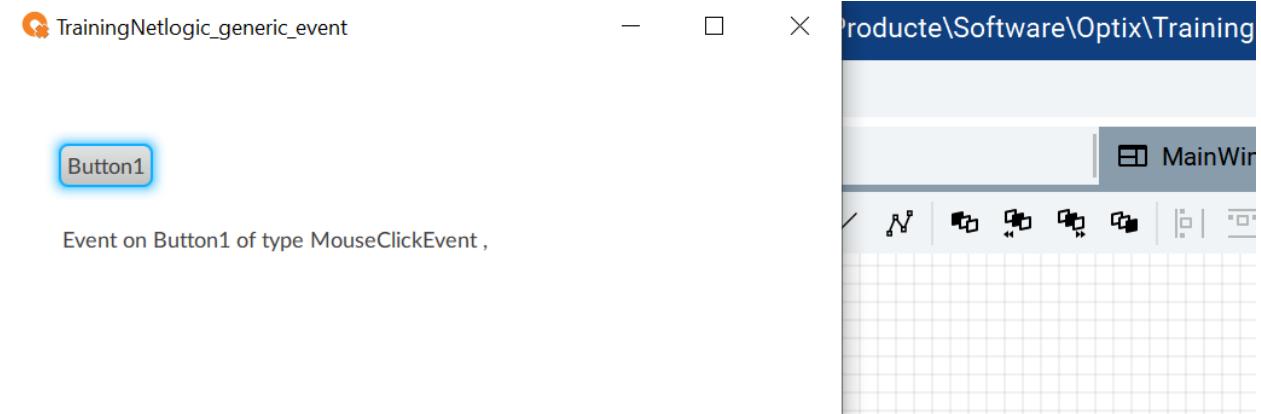
```

Execute the application

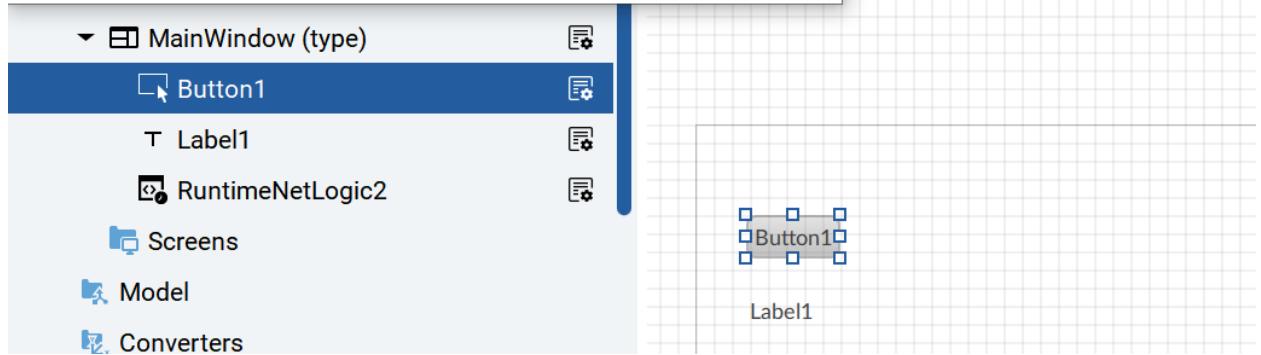
Click the button



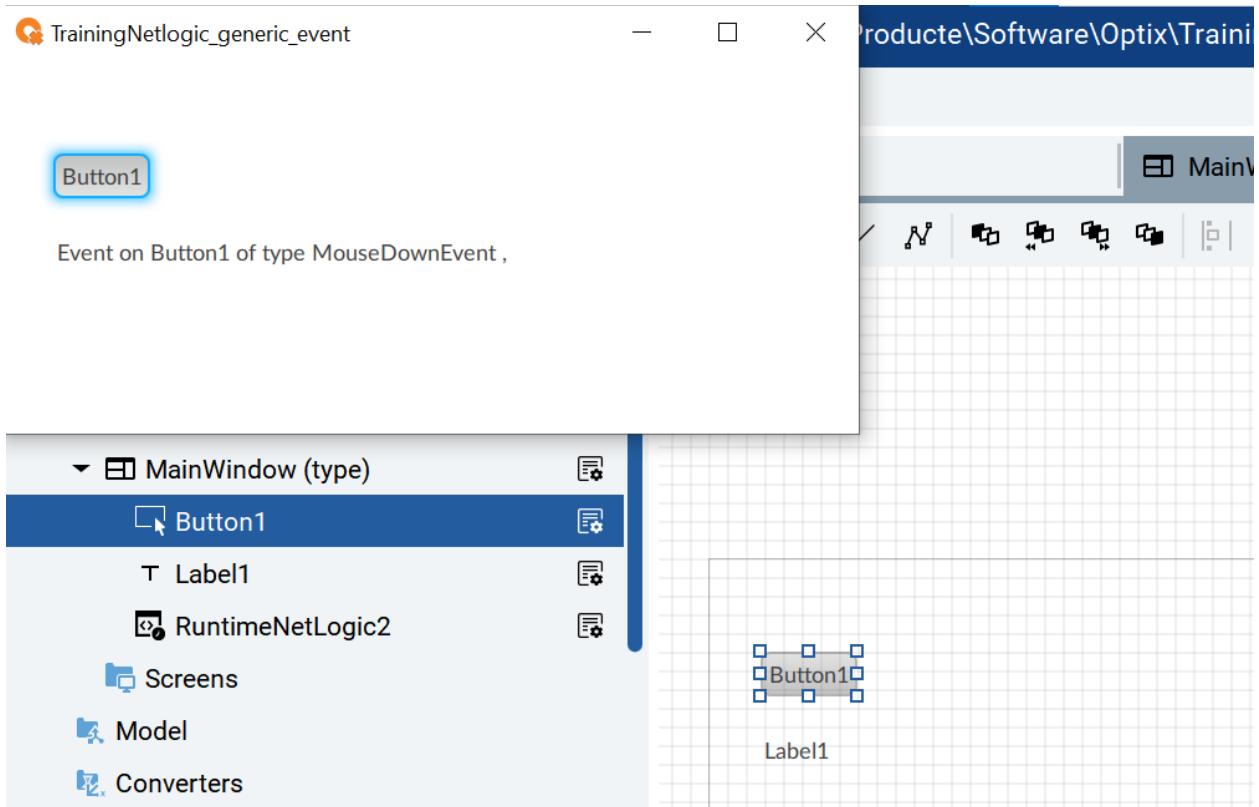
Click on the button



Event on Button1 of type MouseEvent ,

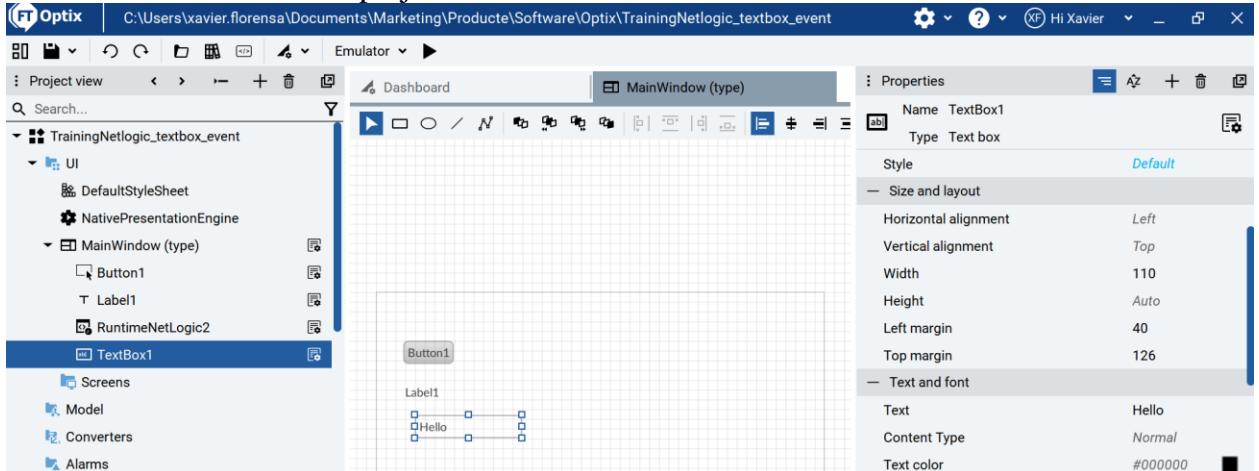


Hold pressing the button



Now let's try to detect a change in a Textbox

Add a Textbox to the same project



And use this code

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Rentativity;
using FTOptix.UI;
using FTOptix.NativeUI;
```

```

using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic2 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        var button1 = Owner.Get<Button>("Button1");
        button1.UAEEvent += Button1_UAEEvent;
        var textbox1 = Owner.Get<TextBox>("TextBox1");
        textbox1.UAEEvent += TextBox1_UAEEvent;

    }

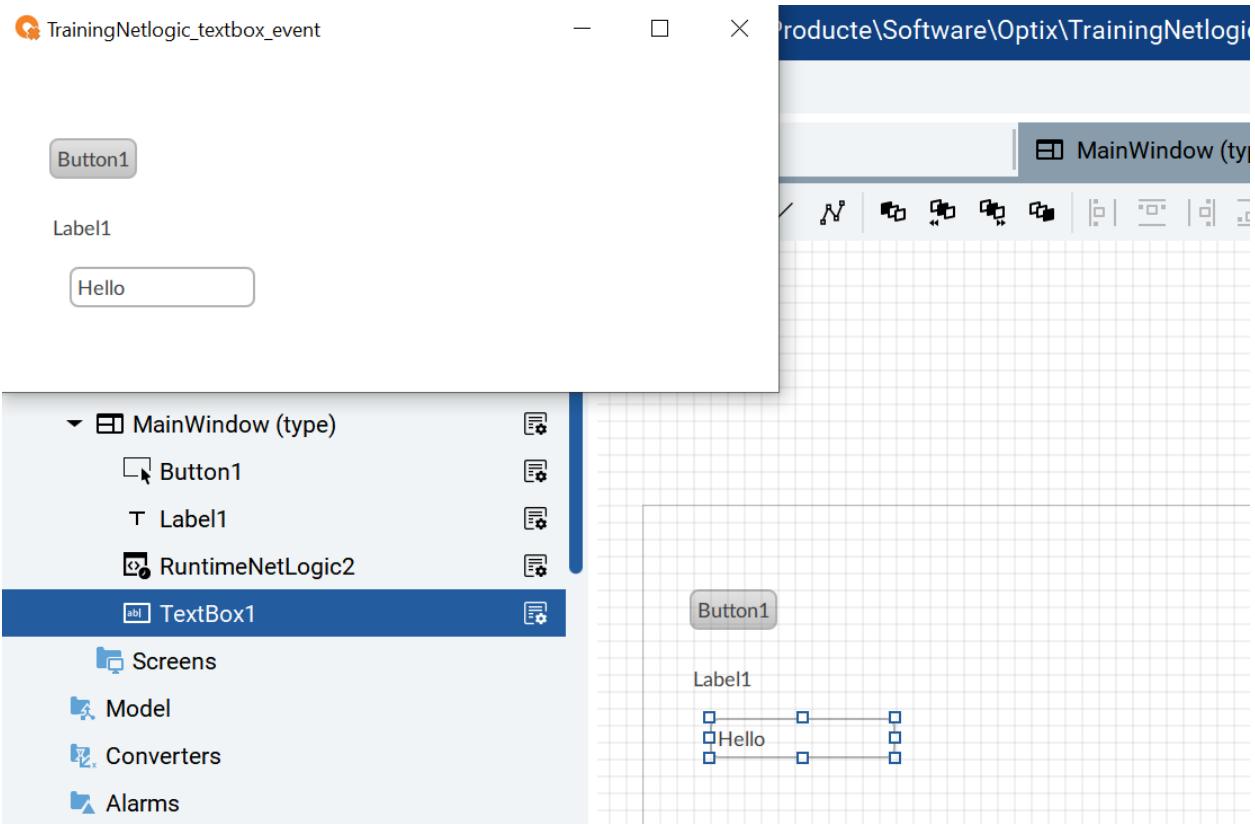
    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }

    private void Button1_UAEEvent(object sender, UAEEventArgs e)
    {
        var label1 = Owner.Get<Label>("Label1");
        var button1 = (Button)sender;
        label1.Text = "Event on " + button1.BrowseName + " of type " +
e.EventType.BrowseName + " , ";
    }

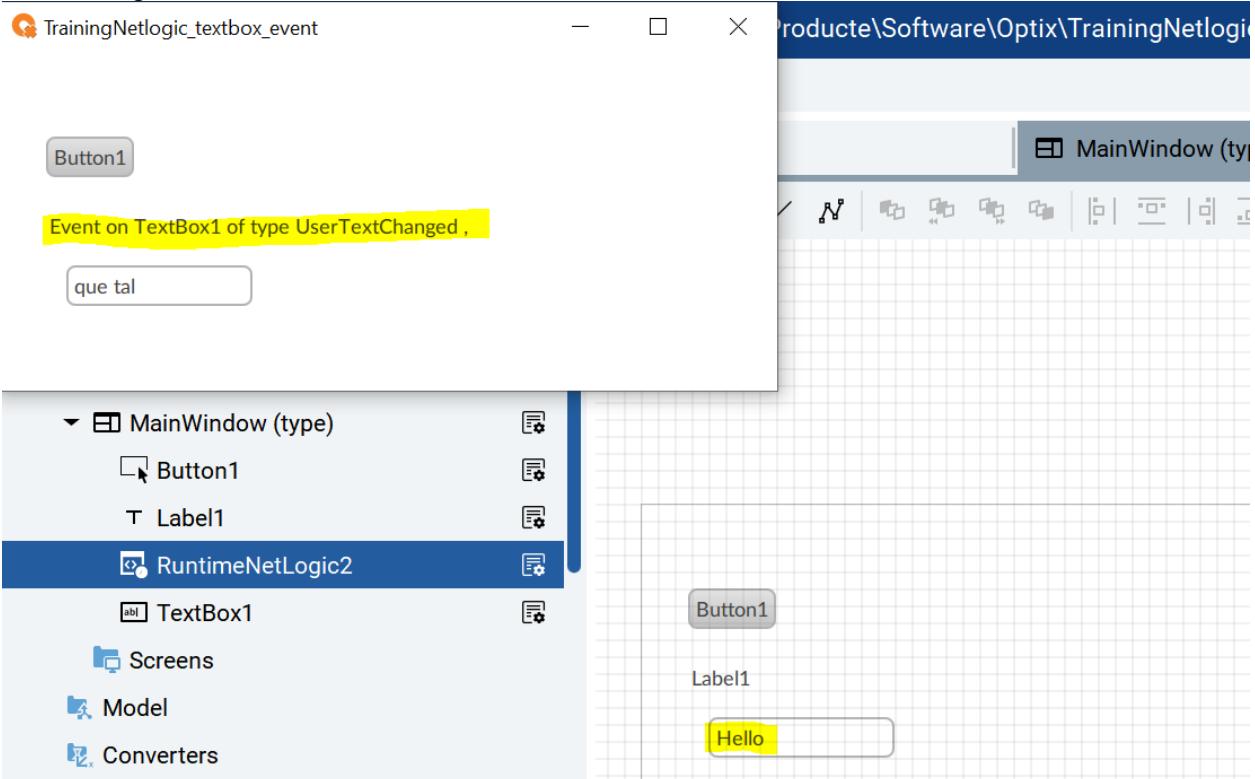
    private void TextBox1_UAEEvent(object sender, UAEEventArgs e)
    {
        var label1 = Owner.Get<Label>("Label1");
        var textbox1 = (TextBox)sender;
        label1.Text = "Event on " + textbox1.BrowseName + " of type " +
e.EventType.BrowseName + " , ";
    }
}

```

Run the project

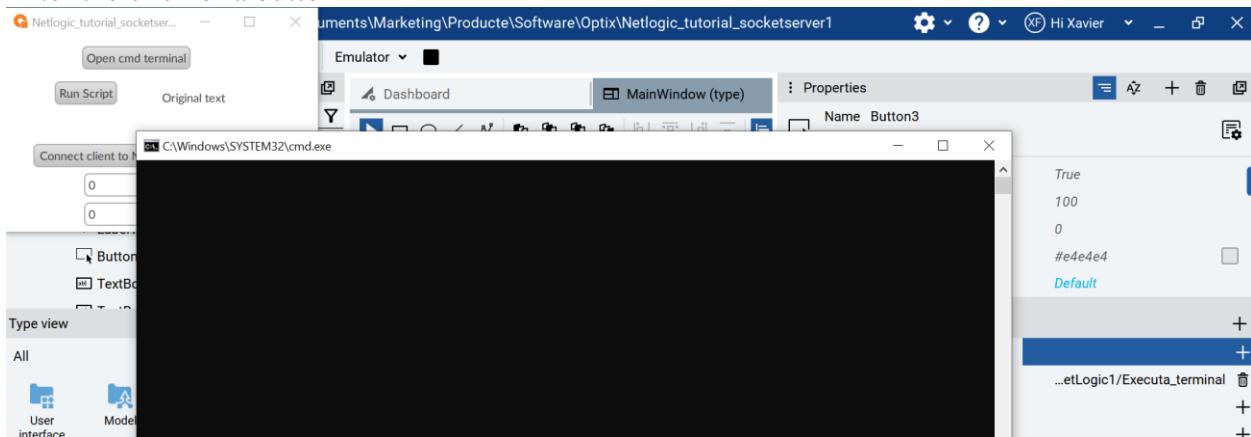


Now change the text and hit enter



## 40. Open a cmd terminal to run a command or program

After the click of a button



In Netlogic create a new Method

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrject;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.Text;
using System.Diagnostics;

#endregion

public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    //static void Executa_terminal(string command)
    public void Executa_terminal()
    {
        int exitCode;
        ProcessStartInfo processInfo;
        Process process;
```

```

//processInfo = new ProcessStartInfo("cmd.exe", "/c " + command);
processInfo = new ProcessStartInfo("cmd.exe");
processInfo.CreateNoWindow = false;
processInfo.UseShellExecute = false;
// *** Redirect the output ***
processInfo.RedirectStandardError = true;
processInfo.RedirectStandardOutput = true;

process = Process.Start(processInfo);
process.WaitForExit();

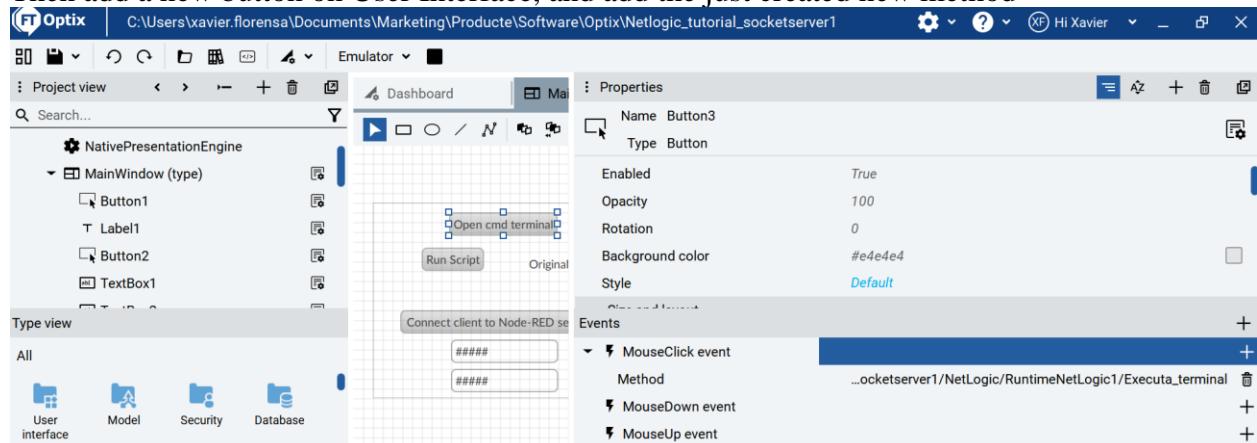
// *** Read the streams ***
// Warning: This approach can lead to deadlocks, see Edit #2
string output = process.StandardOutput.ReadToEnd();
string error = process.StandardError.ReadToEnd();

exitCode = process.ExitCode;

Log.Info("output>>" + (String.IsNullOrEmpty(output) ? "(none)" : output));
Log.Info("error>>" + (String.IsNullOrEmpty(error) ? "(none)" : error));
Log.Info("ExitCode: " + exitCode.ToString(), "ExecuteCommand");
process.Close();
}

```

Then add a new button on User Interface, and add the just created new method



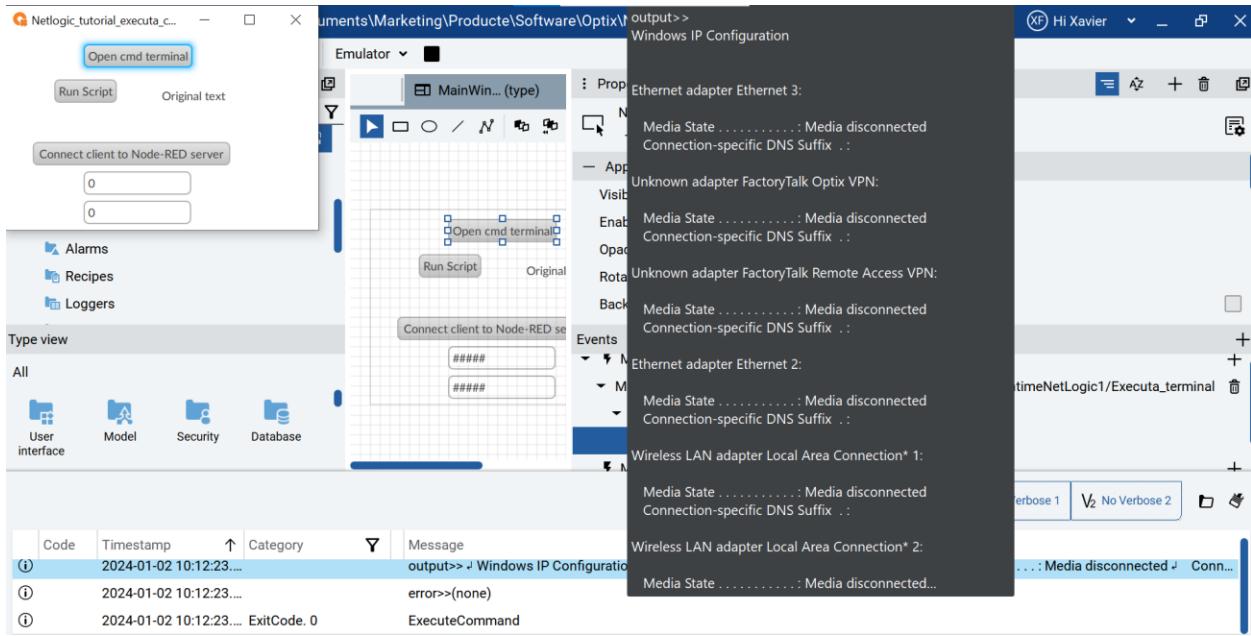
That's all.

Now we want to execute a command

Let's use the original code

But no way to keep the cmd window open,

Instead you get the result on the emulator output



Let's try to open a new window,  
changing

```
processInfo.CreateNoWindow = true;
```

To  
false

But the window is closed immediately  
Let's try with cmd arguments  
Cmd /k ipconfig  
Now the window remains open

```

C:\Windows\System32\cmd.exe
Ethernet adapter VMware Network Adapter VMnet8:

Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::4636:4deb:df47:49c3%5
IPv4 Address. . . . . : 192.168.175.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :
IPv4 Address. . . . . : 192.168.1.163
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1

Ethernet adapter Bluetooth Network Connection:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Ethernet adapter vEthernet (WSL):

Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::817a:e205:4dba:fc20%85
IPv4 Address. . . . . : 172.20.160.1
Subnet Mask . . . . . : 255.255.240.0
Default Gateway . . . . . :

C:\Windows\system32>

```

Let's try with this in Netlogic

Changing this

```

[ExportMethod]
public void Executa_terminal(string command)
{
    int exitCode;
    ProcessStartInfo processInfo;
    Process process;

    processInfo = new ProcessStartInfo("cmd.exe", "/c " + command);

```

To this

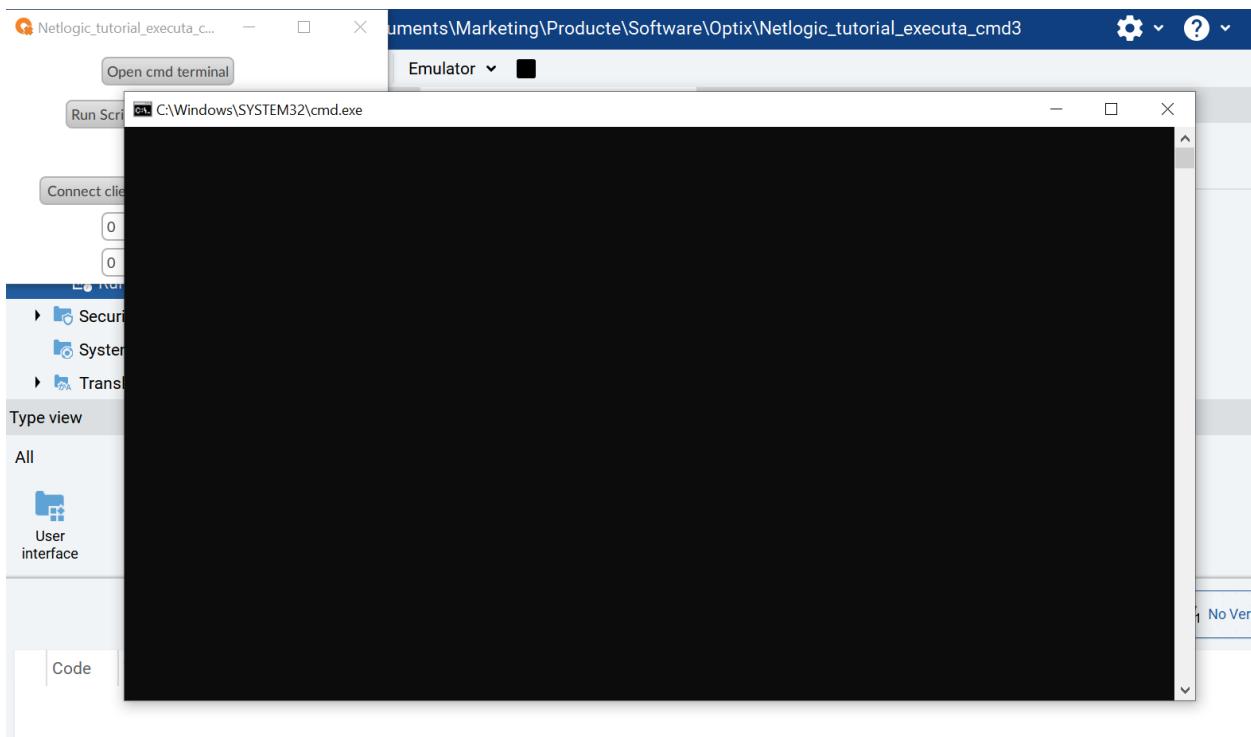
```

[ExportMethod]
0 references
public void Executa_terminal(string command)
{
    int exitCode;
    ProcessStartInfo processInfo;
    Process process;

    processInfo = new ProcessStartInfo("cmd.exe", "/k " + command);

```

But this time the terminal remains open with no output on it



But let's examine the meaning of cmd /C command

Microsoft Windows XP [Version 5.1.2600]  
(C) Copyright 1985-2001 Microsoft Corp.

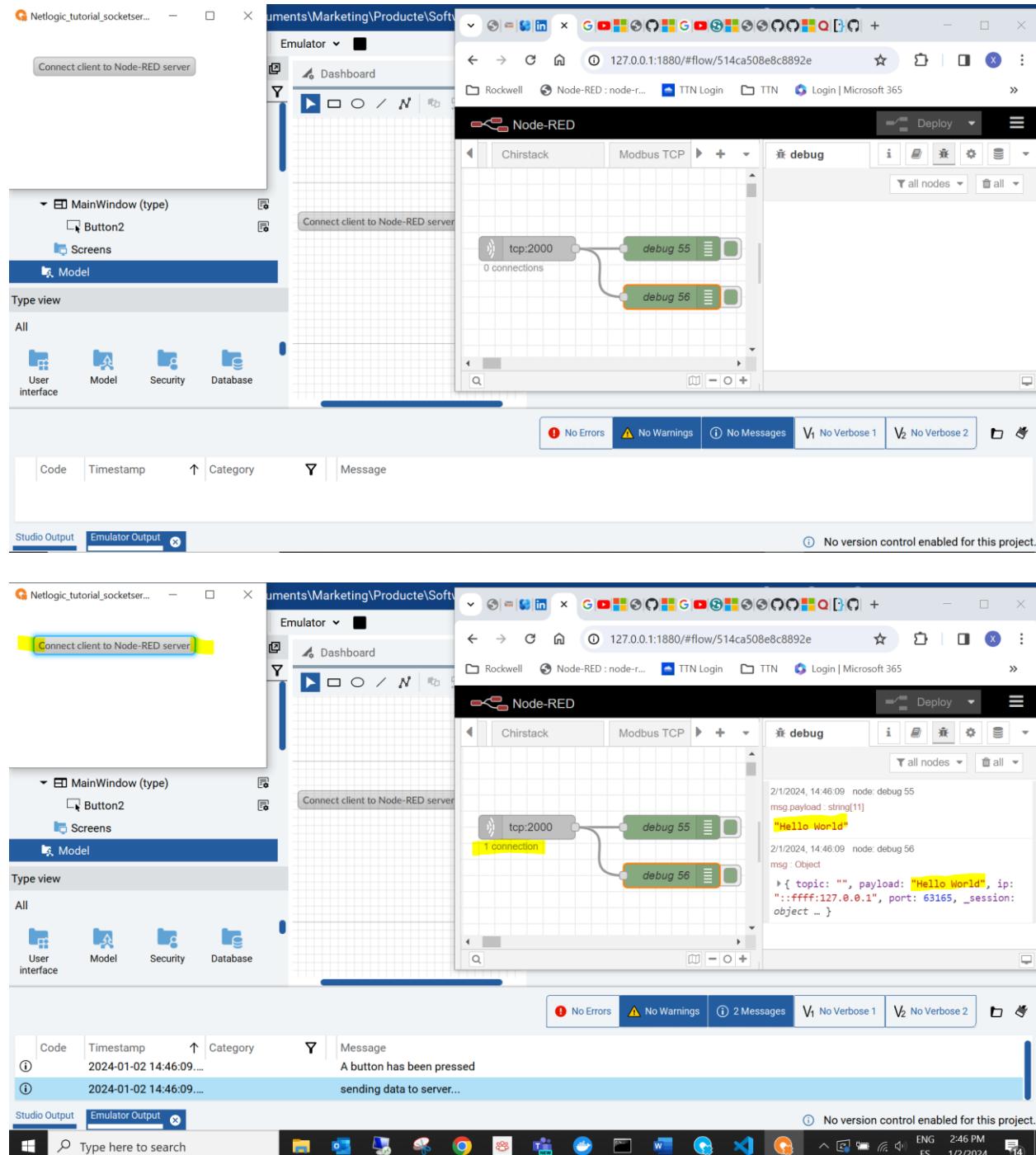
C:\>cmd /?  
Starts a new instance of the Windows XP command interpreter

CMD [/A | /U] [/Q] [/D] [/E:ON | /E:OFF] [/F:ON | /F:OFF] [/V:ON | /V:OFF]  
[[/S] [/C | /K] string]

/C      Carries out the command specified by string and then terminates  
/K      Carries out the command specified by string but remains  
/S      Modifies the treatment of string after /C or /K (see below)  
/Q      Turns echo off  
/D      Disable execution of AutoRun commands from registry (see below)  
/A      Causes the output of internal commands to a pipe or file to be ANSI  
/U      Causes the output of internal commands to a pipe or file to be Unicode  
/T:fg    Sets the foreground/background colors (see COLOR /? for more info)  
/E:ON    Enable command extensions (see below)  
/E:OFF   Disable command extensions (see below)  
/F:ON    Enable file and directory name completion characters (see below)  
/F:OFF   Disable file and directory name completion characters (see below)  
/V:ON    Enable delayed environment variable expansion using ! as the delimiter. For example, /V:ON would allow !var! to expand the

## 41. Socket TCP using System.Net.Sockets without external libraries

We have a socket server in Node-RED listening to port 2000.  
Let's develop an application in FTOptix to connect to the server and to send a message "Hello World".



Using code pasted from this example

<https://www.youtube.com/watch?v=g5yEWLJxNmI&t=11s>

In this way inside Netlogic, using a Method

The screenshot shows the FTOptix software interface. The top navigation bar displays the title 'FT Optix' and the path 'C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\NetlogicTutorial\_SocketServer3'. The top right corner shows a greeting 'Hi Xavier' and a user icon.

The left sidebar contains a 'Project view' tree with nodes like 'Button2', 'Screens', 'Model', 'Converters', 'Alarms', and 'Recipes'. Below it is a 'Type view' section with categories: User interface, Model, Security, and Database.

The main workspace is divided into several panels:

- Dashboard:** Shows a button labeled 'Connect client to Node-RED server'.
- Properties:** A panel on the right showing the properties of a selected 'Button2' component. It includes sections for 'Name' (set to 'Button2'), 'Type' (set to 'Button'), 'Appearance' (Visible: True, Enabled: True, Opacity: 100, Rotation: 0), and 'Events' (MouseClick event, MouseDown event, MouseUp event).
- Code Editor:** The bottom half of the screen displays C# code for a class named 'RuntimeNetLogic1'.

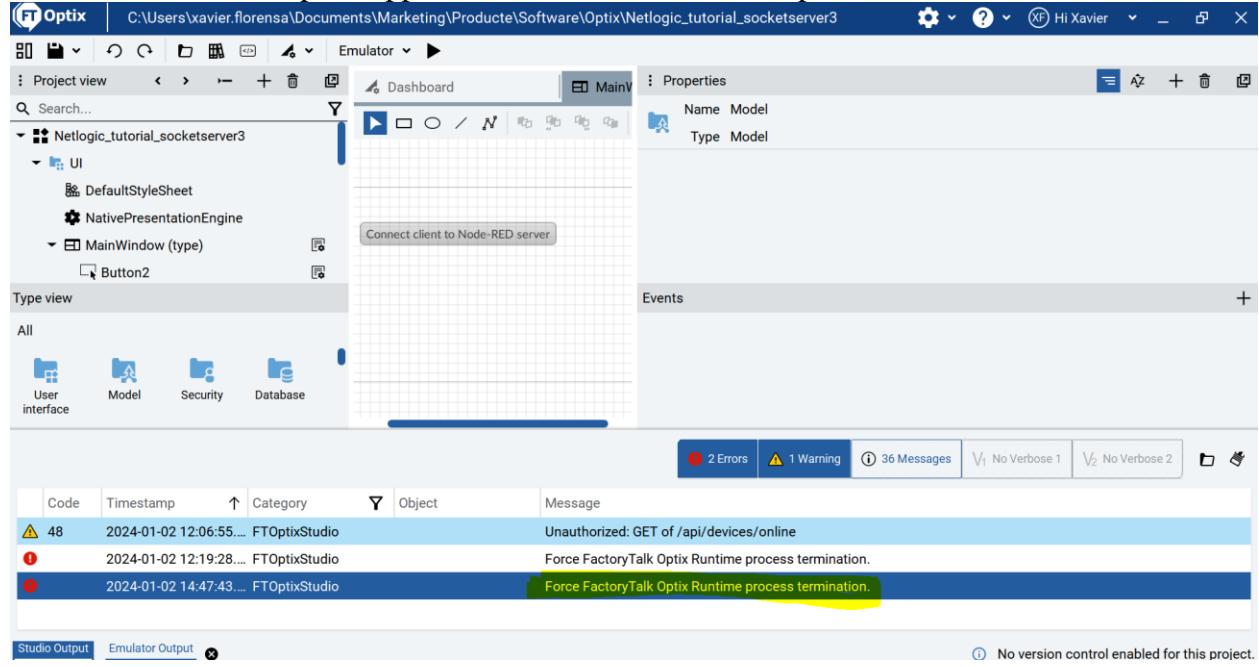
```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void Socketclient_connect()
    {
        Log.Info("A button has been pressed");
        TcpClient client = new TcpClient("127.0.0.1", 2000);
        string messageToSend = "Hello World";
        int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
        byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
        NetworkStream stream = client.GetStream();
        stream.Write(sendData, 0, sendData.Length);
        //Console.WriteLine("sending data to server...");
        Log.Info("sending data to server...");
    }
}
```

```

        StreamReader sr = new StreamReader(stream);
        string response = sr.ReadLine();
        //Console.WriteLine(response);
        Log.Info(response);
        stream.Close();
        client.Close();
        //Console.ReadKey();
    }
}

```

But there is a problem stopping this application  
 You are not able to stop the application with the Emulator stop button.



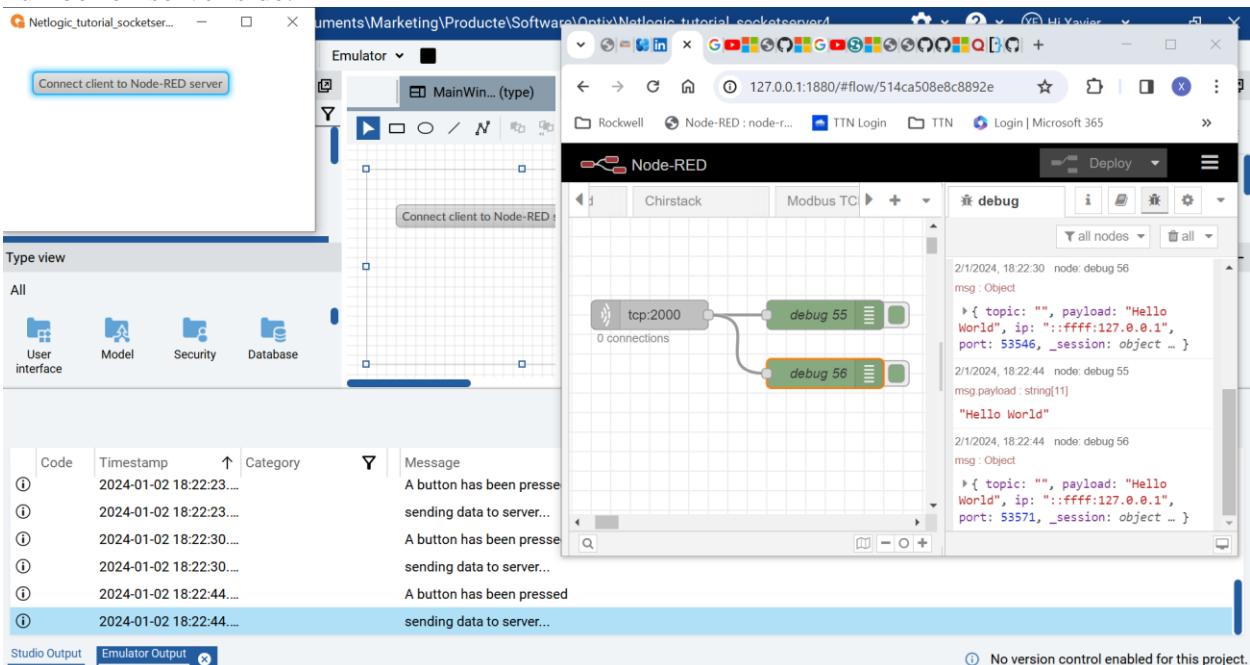
We can have better results if we comment the lines related to receiving data

```

0 references
18 public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
19 {
20     [ExportMethod]
0 references
21     public void ...Socketclient_connect()
22     {
23         Log.Info("A button has been pressed");
24         TcpClient client = new TcpClient("127.0.0.1", 2000);
25         string messageToSend = "Hello World";
26         int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
27         byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
28         NetworkStream stream = client.GetStream();
29         stream.Write(sendData, 0, sendData.Length);
//Console.WriteLine("sending data to server...");
30         Log.Info("sending data to server...");
31         //StreamReader sr = new StreamReader(stream);
32         //string response = sr.ReadLine();
33         //Console.WriteLine(response);
34         //Log.Info(response);
35         stream.Close();
36         client.Close();
37         //Console.ReadKey();
38     }
39 }
40 }

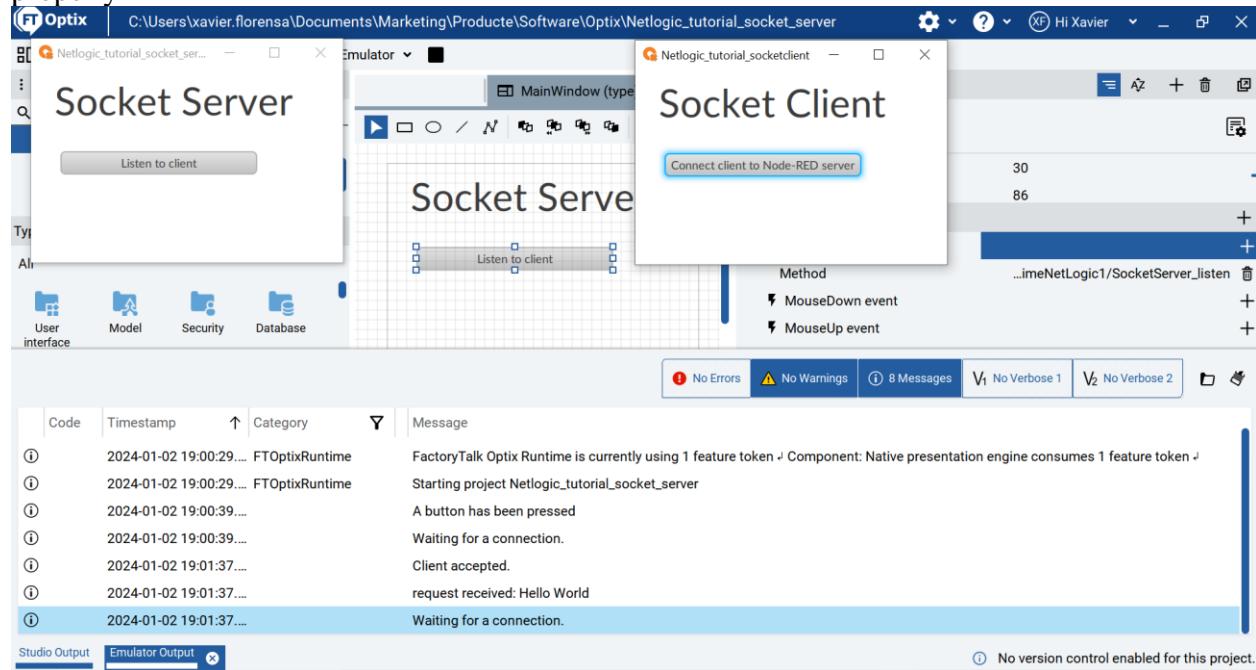
```

Now you can stop the program with the stop key  
 And you can execute several times the send button. But you will not see any connected client number on server side.



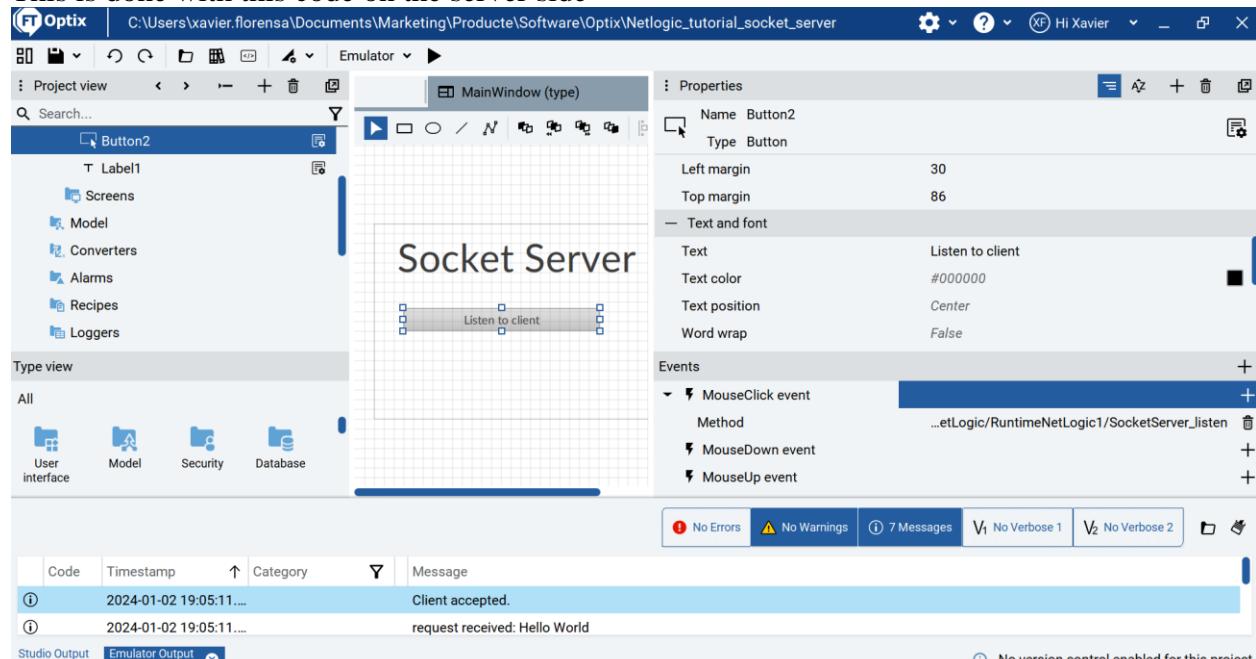
Now let's try to write an Optix application as a Socket server.

We managed to make a transaction of Hello World, but the server application does not stop properly



You can connect and send messages several times.

This is done with this code on the server side



And this code on the server side Netlogic

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
```

```

using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen()
    {
        Log.Info("A button has been pressed");
        TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
        listener.Start();
        while (true)
        {
            //Console.WriteLine("Waiting for a connection.");
            Log.Info("Waiting for a connection.");
            TcpClient client = listener.AcceptTcpClient();
            //Console.WriteLine("Client accepted.");
            Log.Info("Client accepted.");
            NetworkStream stream = client.GetStream();
            StreamReader sr = new StreamReader(client.GetStream());
            StreamWriter sw = new StreamWriter(client.GetStream());
            try
            {
                byte[] buffer = new byte[1024];
                stream.Read(buffer, 0, buffer.Length);
                int recv = 0;
                foreach (byte b in buffer)
                {
                    if (b!=0)
                    {
                        recv++;
                    }
                }
                string request = Encoding.UTF8.GetString(buffer, 0, recv);
                //Console.WriteLine("request received: " + request);
                Log.Info("request received: " + request);
            }
        }
    }
}

```

```

        sw.WriteLine("You rock!");
        sw.Flush();
    }
    catch(Exception e)
    {
        //Console.WriteLine("Something went wrong.");
        Log.Info("Something went wrong.");
        sw.WriteLine(e.ToString());
    }
}
}

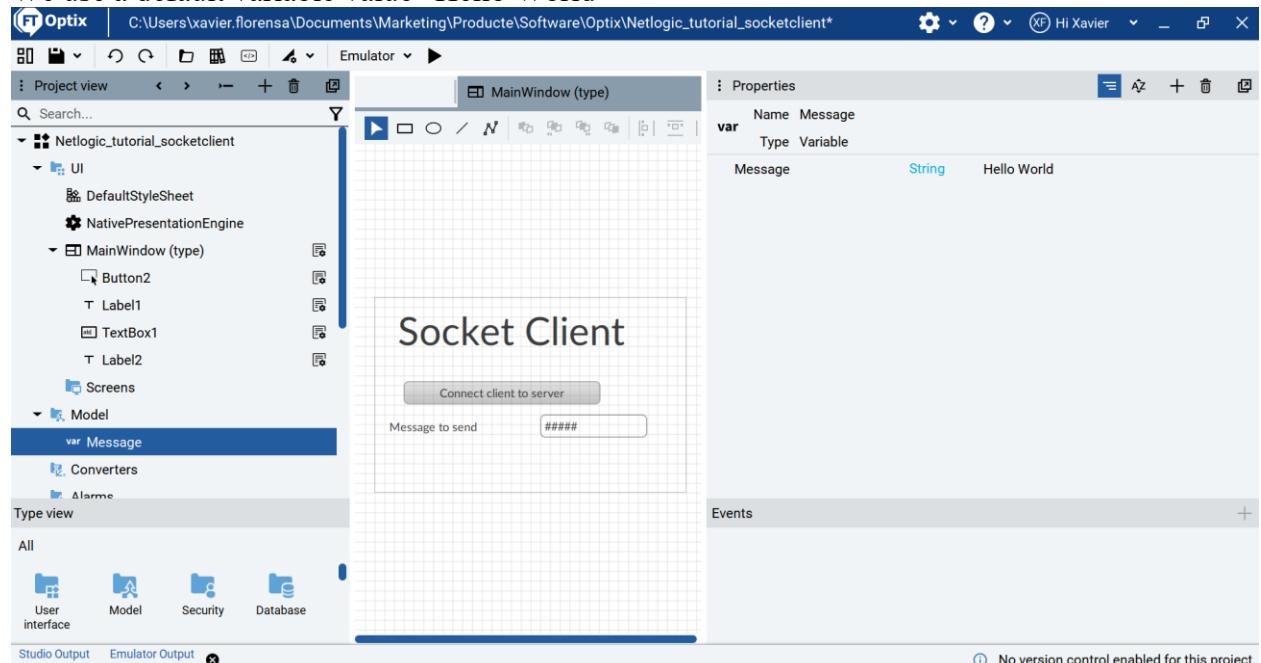
```

Now let's try to improve the code with some text boxes

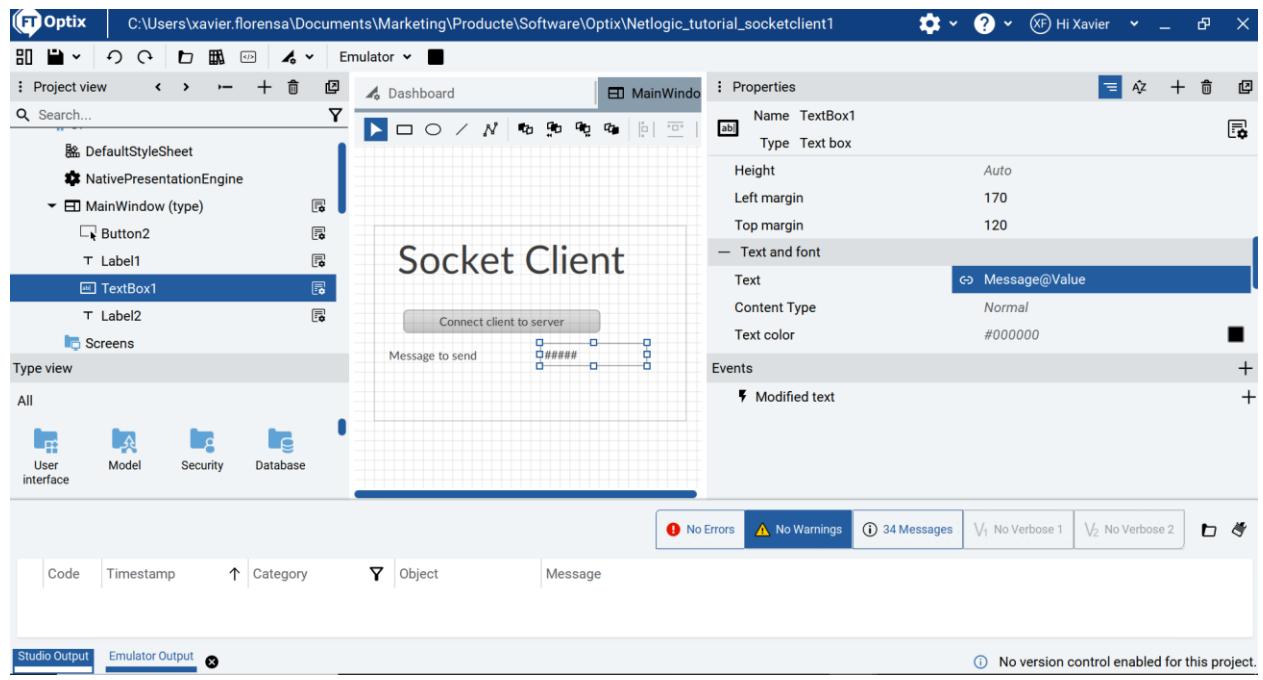
First of all let's introduce the text message to send manually from a Text Box

Let's go to client and add a Message string variable.

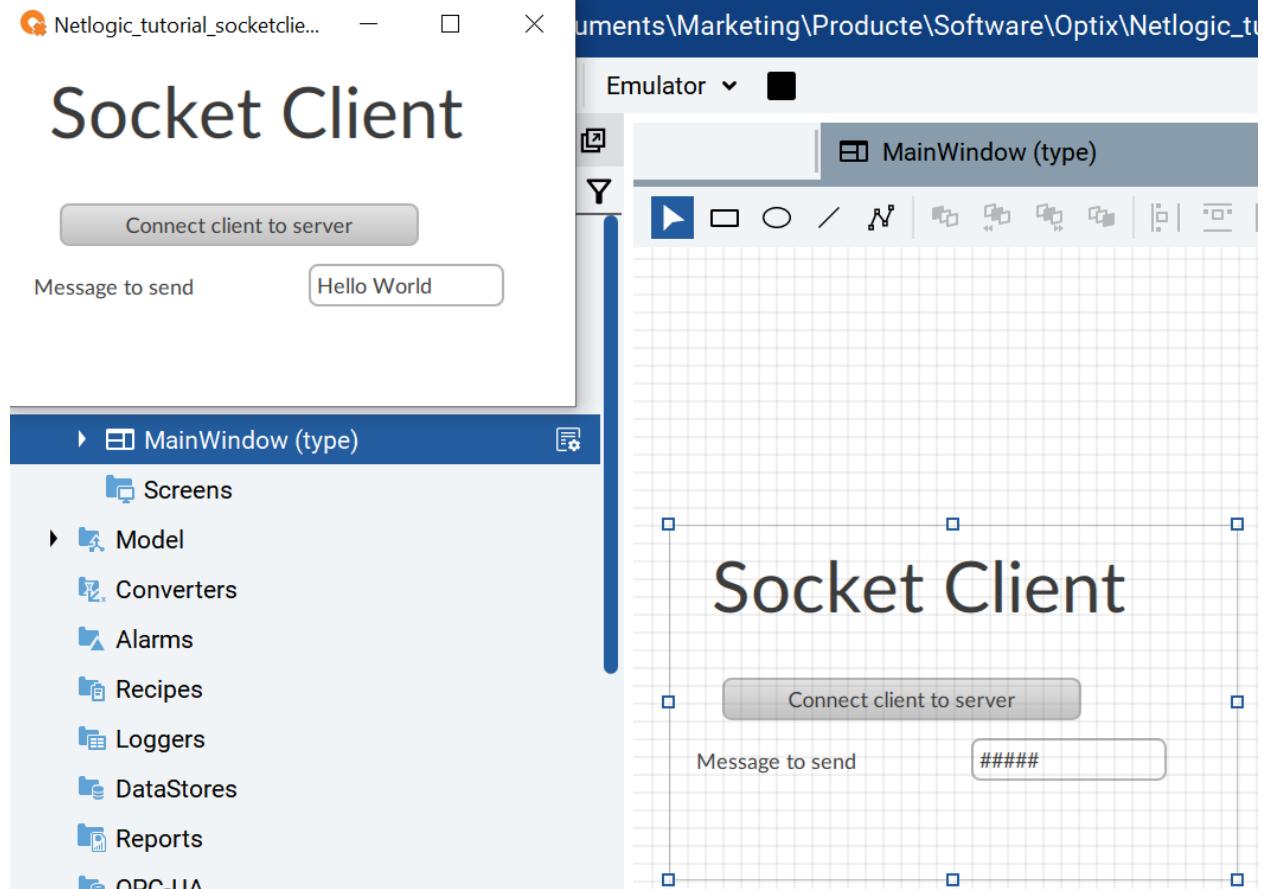
We use a default variable value "Hello World"



And a Textbox with a dynamic link to variable Message



If we execute the application we will see the default value



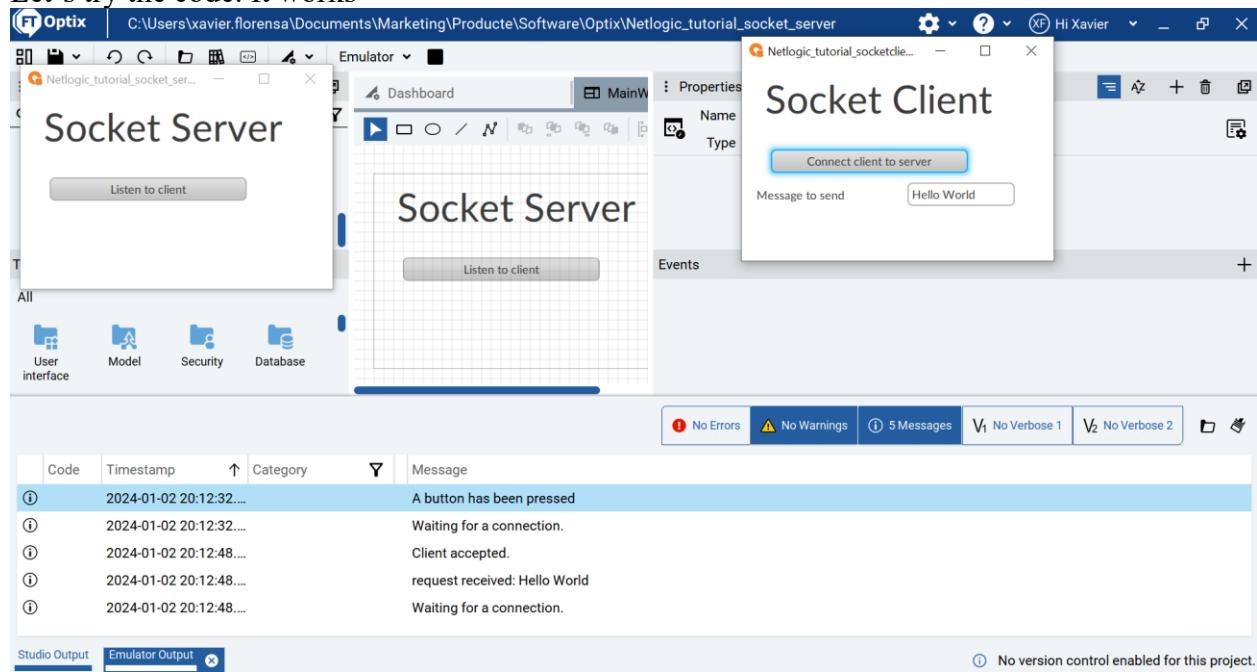
But we have to pass that string value to our code, by means of the UI variable Message

```
[ExportMethod]
```

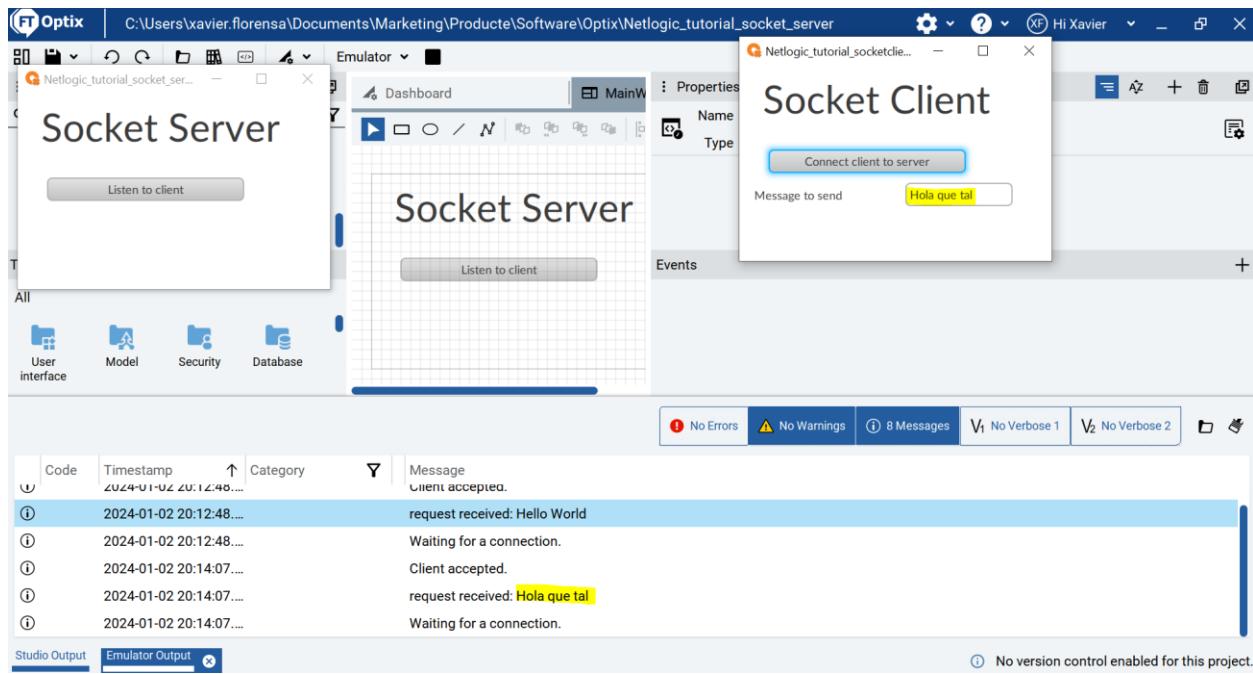
```
0 references
```

```
public void Socketclient_connect()
{
    Log.Info("A button has been pressed");
    TcpClient client = new TcpClient("127.0.0.1", 2000);
    var missatge = Project.Current.GetVariable("Model/Message");
    //string messageToSend = "Hello World";
    string messageToSend = missatge.Value;
}
```

Let's try the code. It works

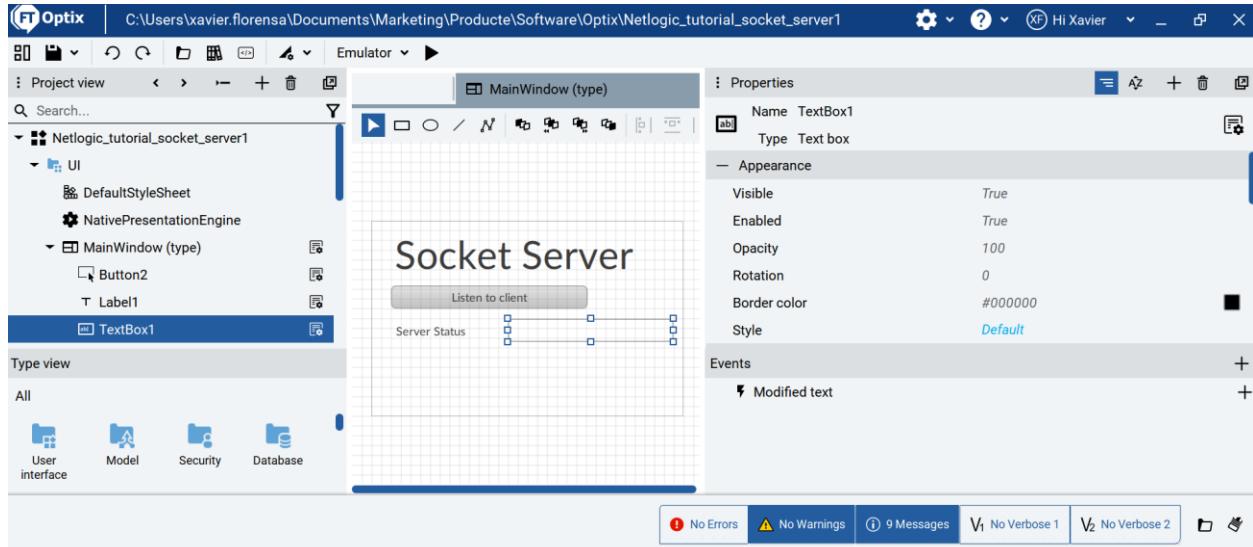


And we can send different messages



Let's improve the application with a status window on the server

Let's create a new Textbox



We will use the NodeId of the Textbox to be able to display text on it.

It's easy to use the Method we have already created to pass an argument as the Textbox NodeId

First of all we have to change the Method to require an input parameter, the NodeId.

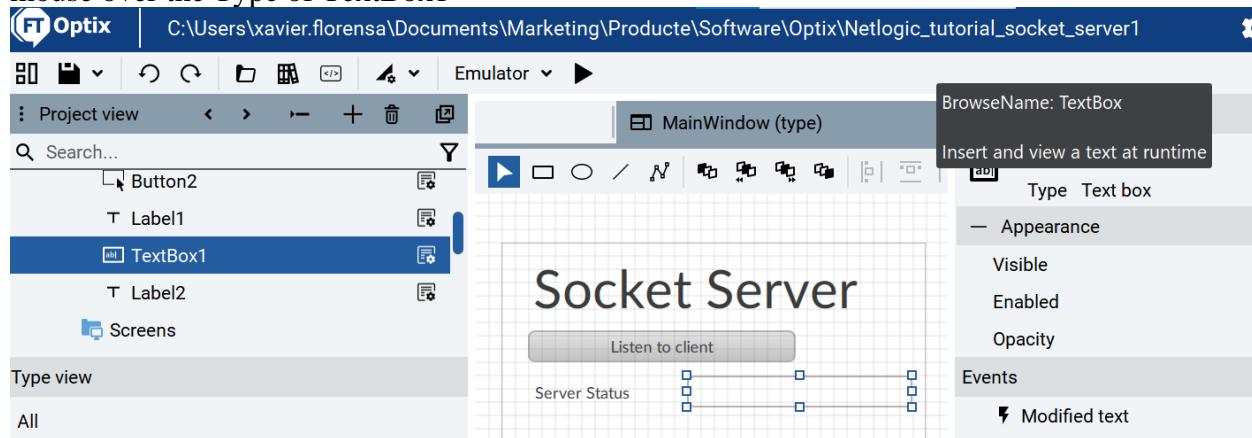
This way

```

[ExportMethod]
0 references
public void ...SocketServer_listen(NodeId textBoxNodeId)
{
    var textBox = InformationModel.Get<TextBox>(textBoxNodeId);
    textBox.Text = "A button has been pressed";
    Log.Info("A button has been pressed");
    TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
    listener.Start();
    while (true)
    {
}

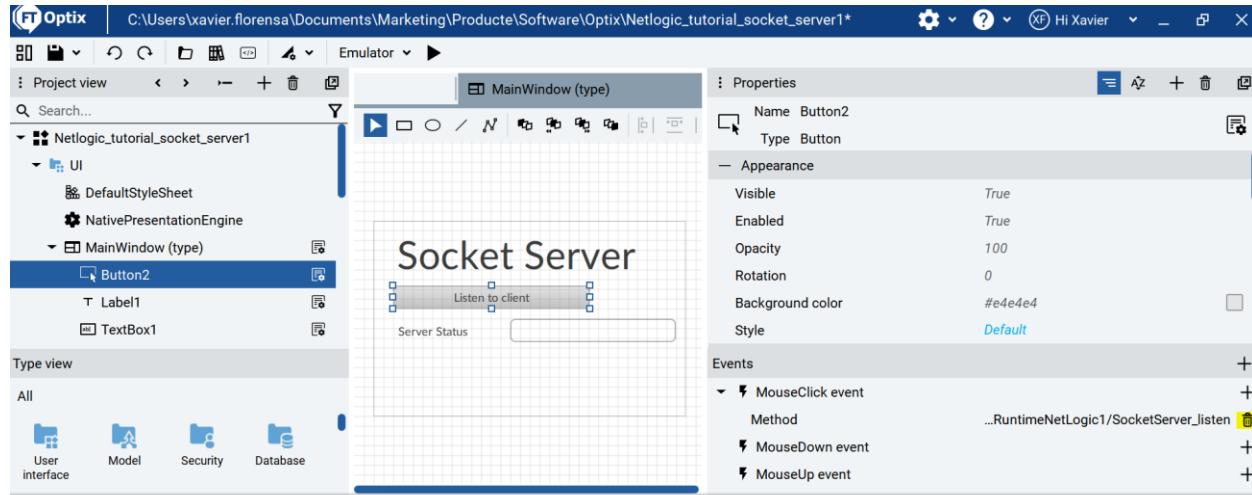
```

Use this name TextBox in brackets after Get, as is the name displayed here when you move the mouse over the Type of TextBox1

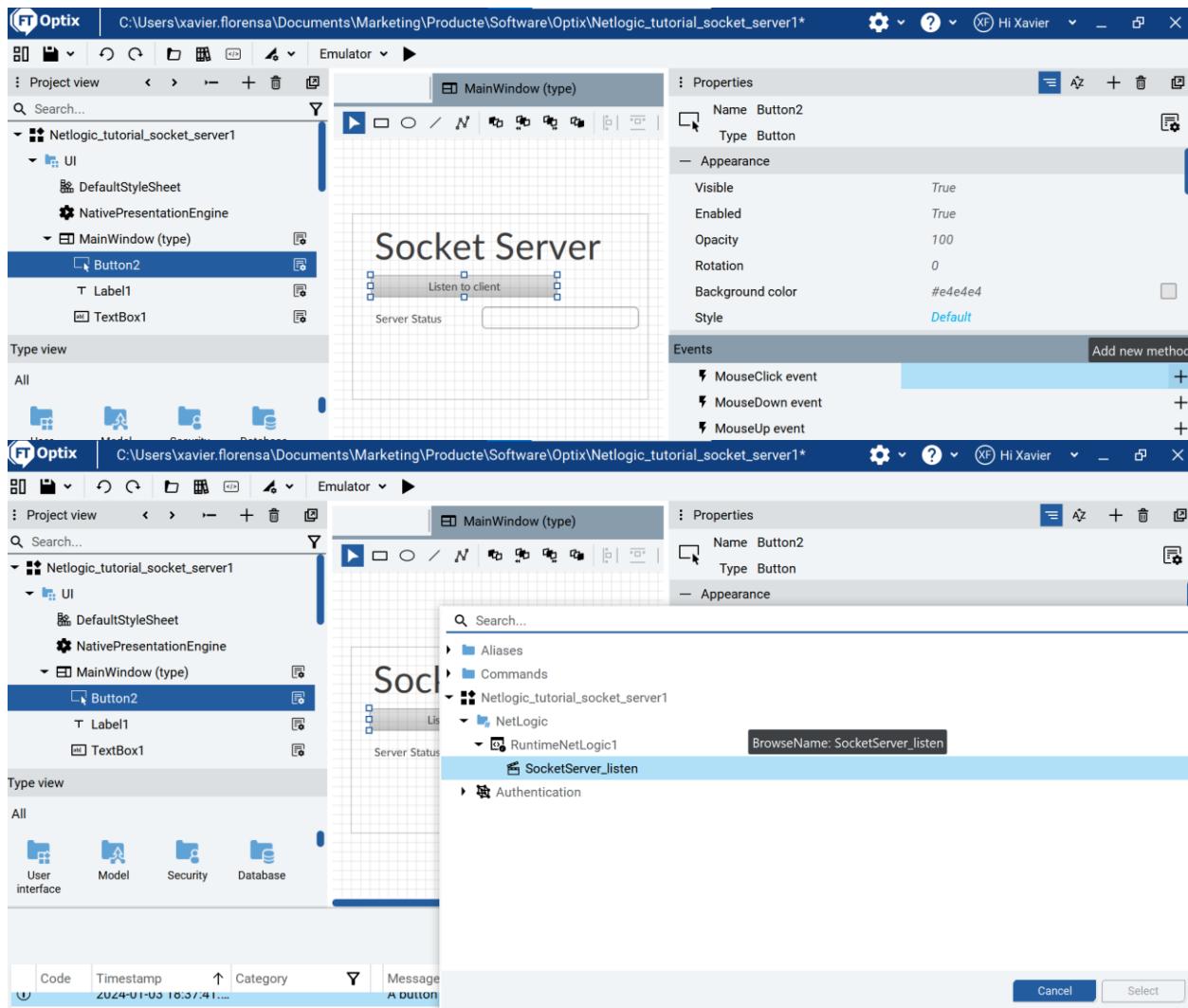


Now let's save the code and let's assign a dynamic link between the Method input parameter and TextBox1

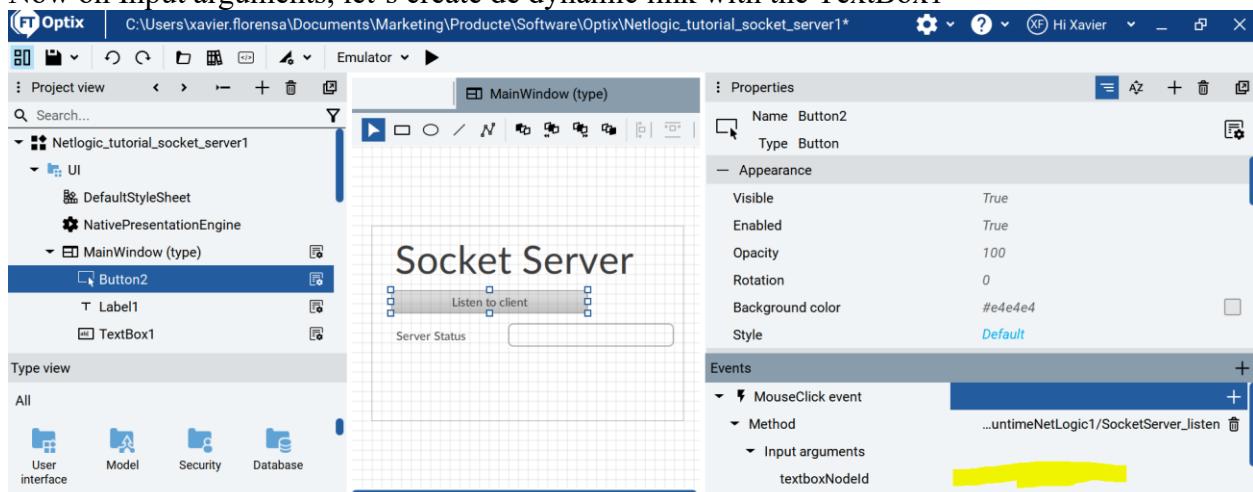
We have to delete the method call after a button click



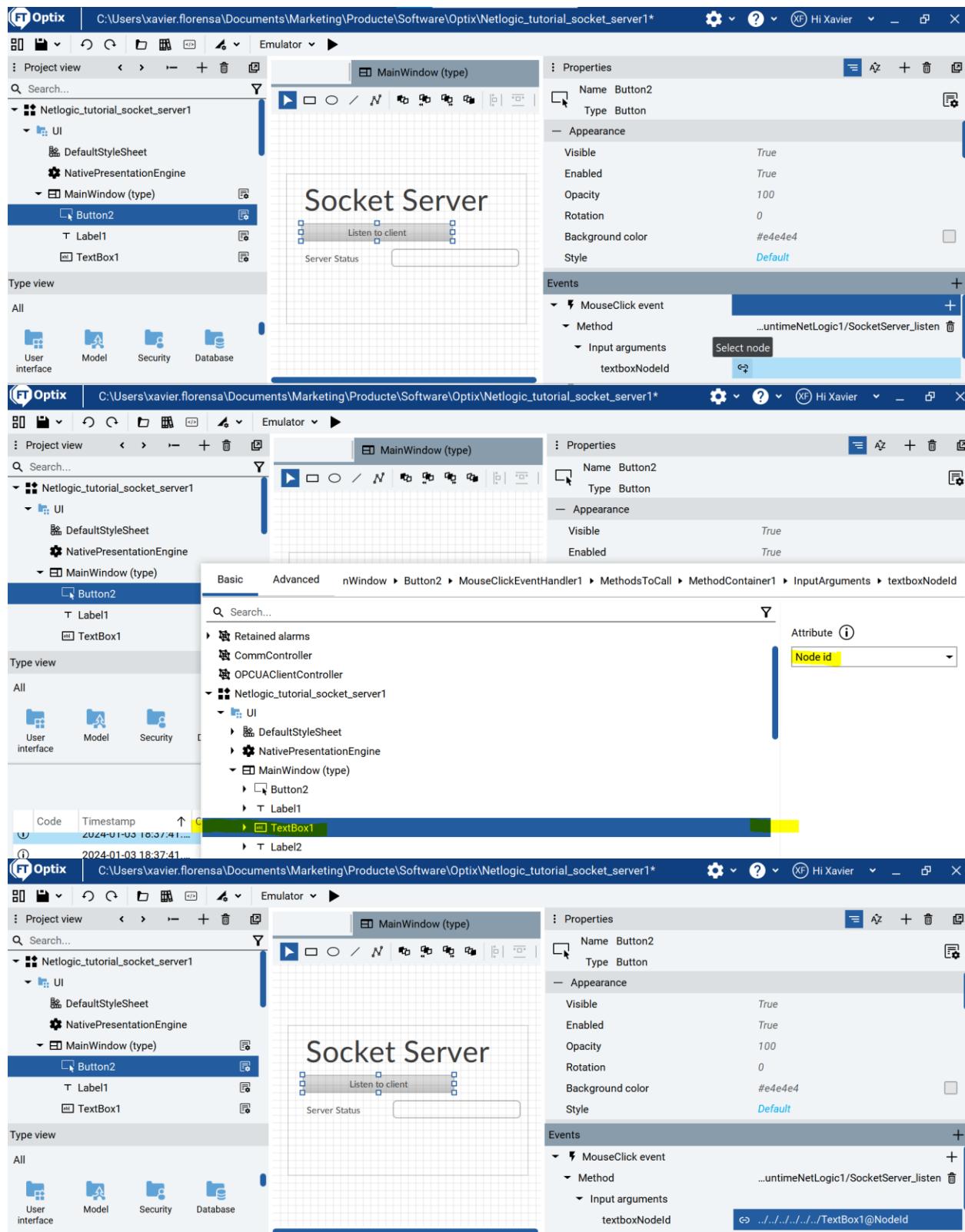
And create a new action on Button click



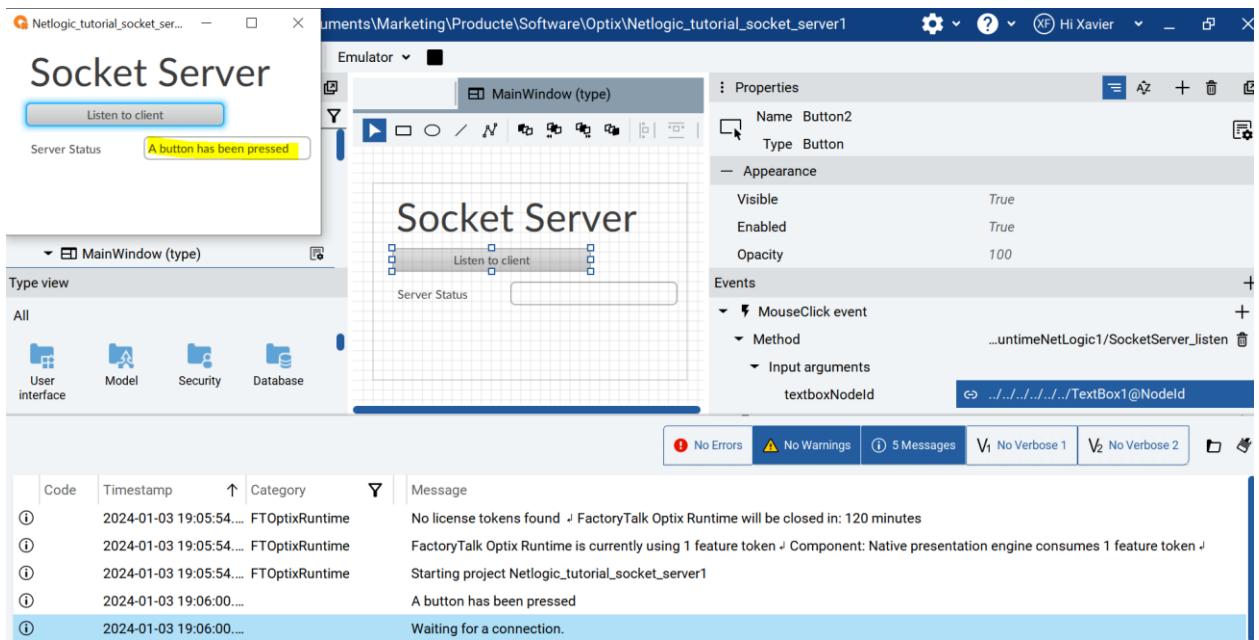
Now on Input arguments, let's create de dynamic link with the TextBox1



So select node



Let's try it to see that "a button has been pressed" on the Textox on Emulator runtime window



This is working,

Now let's display further messages by modifying Netlogic this way, before each Log.Info sentence, let's use our TextBox.

```
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen(NodeId textboxNodeId)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        textbox.Text = "A button has been pressed";
        Log.Info("A button has been pressed");
        TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
        listener.Start();
        while (true)
        {
            //Console.WriteLine("Waiting for a connection.");
            textbox.Text = "Waiting for a connection.";
            Log.Info("Waiting for a connection.");
            TcpClient client = listener.AcceptTcpClient();
            //Console.WriteLine("Client accepted.");
            textbox.Text = "Client accepted.";
            Log.Info("Client accepted.");
            NetworkStream stream = client.GetStream();
            StreamReader sr = new StreamReader(client.GetStream());
            StreamWriter sw = new StreamWriter(client.GetStream());
            try
            {
```

Save the code and run the Emulator

This is working but you do not see the message Client accepted since it changes too fast to another waiting for connection.

So let's add a little pause for 1 second at the end of the While (true) loop

```
        }

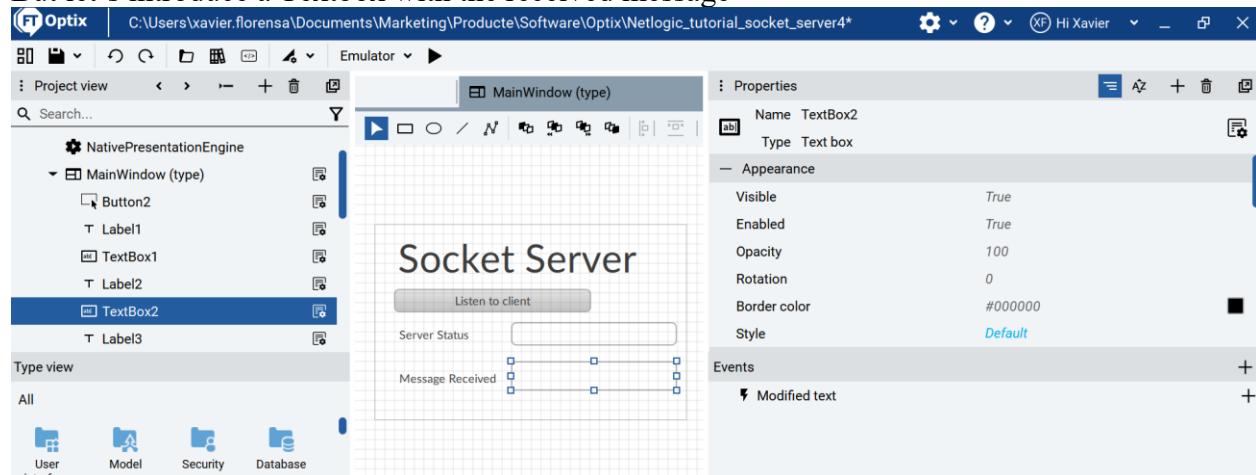
        System.Threading.Thread.Sleep(1000);

    }
```

Since this code closes the connection after having received the message.

(this is another improvement to make to the code later on, to keep the socket connection open)

But let's introduce a Textbox with the received message



And modifying Netlogic

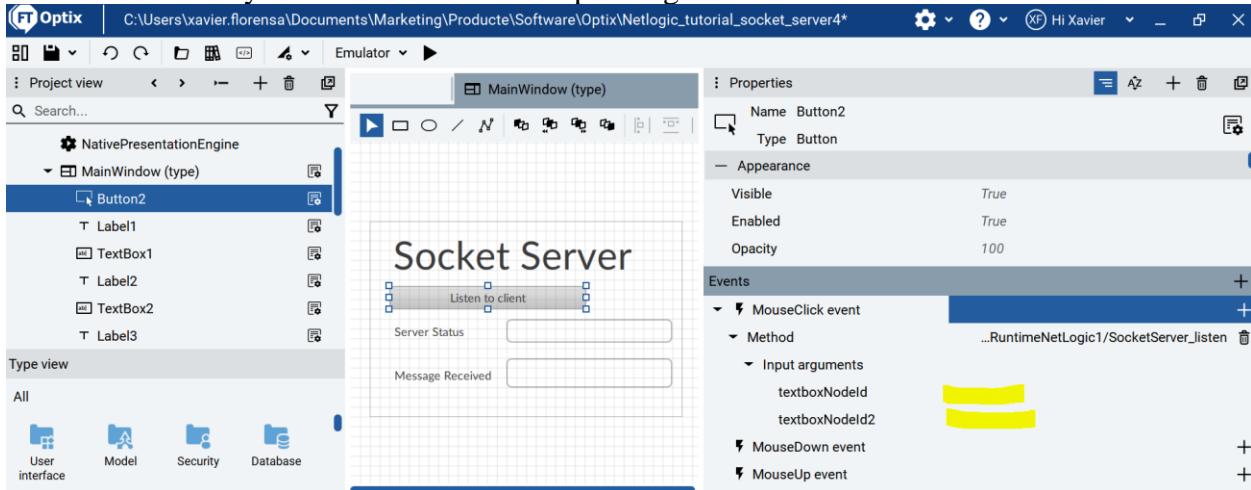
so let's change the Methods again with a second input parameter like this.

```
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen(NodeId textboxNodeId, NodeId textboxNodeId2)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
        textbox.Text = "A button has been pressed";
        Log.Info("A button has been pressed");
        TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
        listener.Start();
        while (true)
```

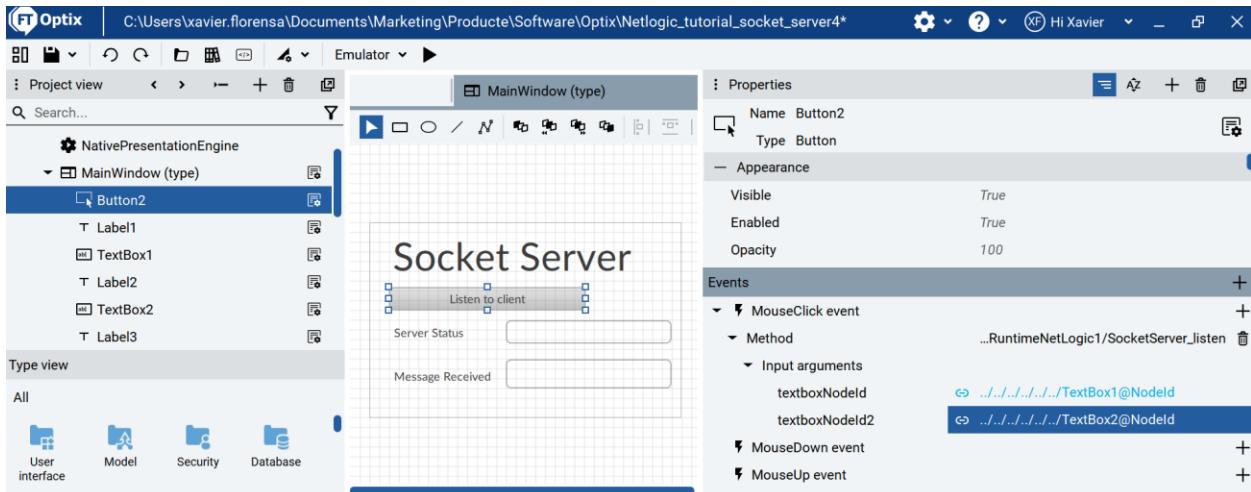
And this is the point where we will show the received message

```
    string request = Encoding.UTF8.GetString(buffer, 0, recv);
    //Console.WriteLine("request received: " + request);
    textbox2.Text = request;
    Log.Info("request received: " + request);
```

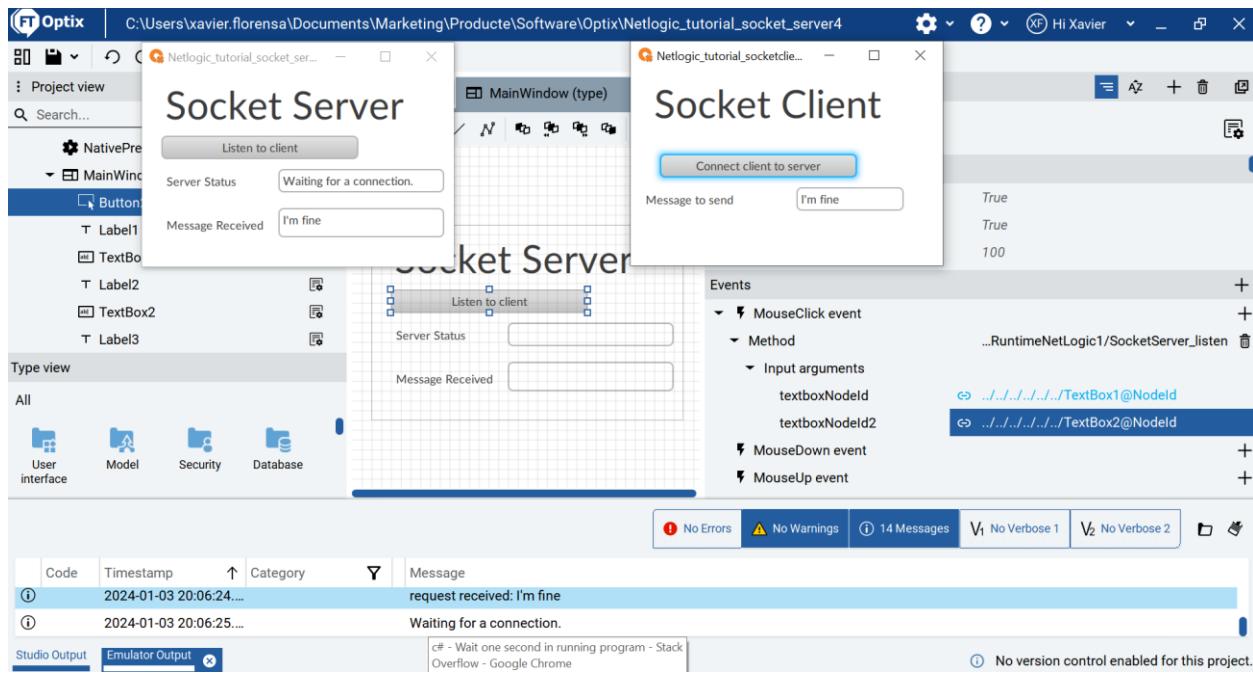
Let's save the code, erase the Method call after a button click and generate again the button click call with the two dynamic links to the corresponding textboxes



Like this



Let's run the application



It works!

This is the code for the server

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen(NodeId textboxNodeId, NodeId textboxNodeId2)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
        textbox.Text = "A button has been pressed";
        Log.Info("A button has been pressed");
    }
}
```

```

TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
listener.Start();
while (true)
{
    //Console.WriteLine("Waiting for a connection.");
    textbox.Text = "Waiting for a connection.";
    Log.Info("Waiting for a connection.");
    TcpClient client = listener.AcceptTcpClient();
    //Console.WriteLine("Client accepted.");
    textbox.Text = "Client accepted.";
    Log.Info("Client accepted.");

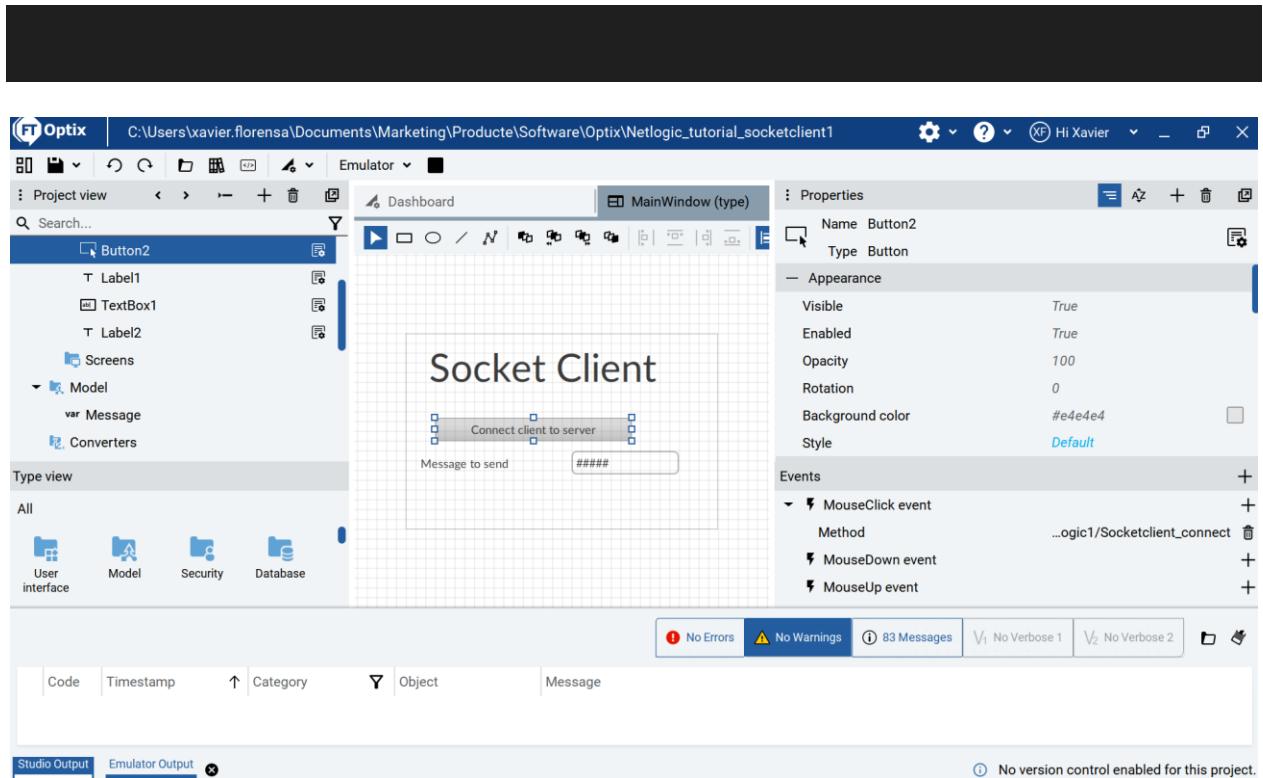
    NetworkStream stream = client.GetStream();
    StreamReader sr = new StreamReader(client.GetStream());
    StreamWriter sw = new StreamWriter(client.GetStream());

    try
    {
        byte[] buffer = new byte[1024];
        stream.Read(buffer, 0, buffer.Length);
        int recv = 0;
        foreach (byte b in buffer)
        {
            if (b!=0)
            {
                recv++;
            }
        }
        string request = Encoding.UTF8.GetString(buffer, 0, recv);
        //Console.WriteLine("request received: " + request);
        textbox2.Text = request;
        Log.Info("request received: " + request);
        sw.WriteLine("You rock!");
        sw.Flush();
    }
    catch(Exception e)
    {
        //Console.WriteLine("Something went wrong.");
        Log.Info("Something went wrong.");
        sw.WriteLine(e.ToString());
    }
    System.Threading.Thread.Sleep(1000);
}
}

```

And this is the code for the client

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void Socketclient_connect()
    {
        Log.Info("A button has been pressed");
        TcpClient client = new TcpClient("127.0.0.1", 2000);
        var missatge = Project.Current.GetVariable("Model/Message");
        //string messageToSend = "Hello World";
        string messageToSend = missatge.Value;
        int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
        byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
        NetworkStream stream = client.GetStream();
        stream.Write(sendData, 0, sendData.Length);
        //Console.WriteLine("sending data to server...");
        Log.Info("sending data to server...");
        //StreamReader sr = new StreamReader(stream);
        //string response = sr.ReadLine();
        //Console.WriteLine(response);
        //Log.Info(response);
        stream.Close();
        client.Close();
        //Console.ReadKey();
    }
}
```



As you can see on this video

<https://youtu.be/6al-MB8CBQg>

You can get the code here

<https://github.com/xavierflorensa/Netlogic tutorial socket server5.git>

<https://github.com/xavierflorensa/Netlogic tutorial socketclient2.git>

Now let's improve the code

We do not want the socket connection to be closed after sending the message.

Let's comment the stream close and client close sentences at the end of the code

```
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void Socketclient_connect()
    {
        Log.Info("A button has been pressed");
        TcpClient client = new TcpClient("127.0.0.1", 2000);
        var missatge = Project.Current.GetVariable("Model/Message");
        //string messageToSend = "Hello World";
        string messageToSend = missatge.Value;
        int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
        byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
        NetworkStream stream = client.GetStream();
        stream.Write(sendData, 0, sendData.Length);
        //Console.WriteLine("sending data to server...");
        Log.Info("sending data to server...");
        //StreamReader sr = new StreamReader(stream);
```

```

        //string response = sr.ReadLine();
        //Console.WriteLine(response);
        //Log.Info(response);
        //stream.Close();
        //client.Close();
        //Console.ReadKey();
    }
}

```

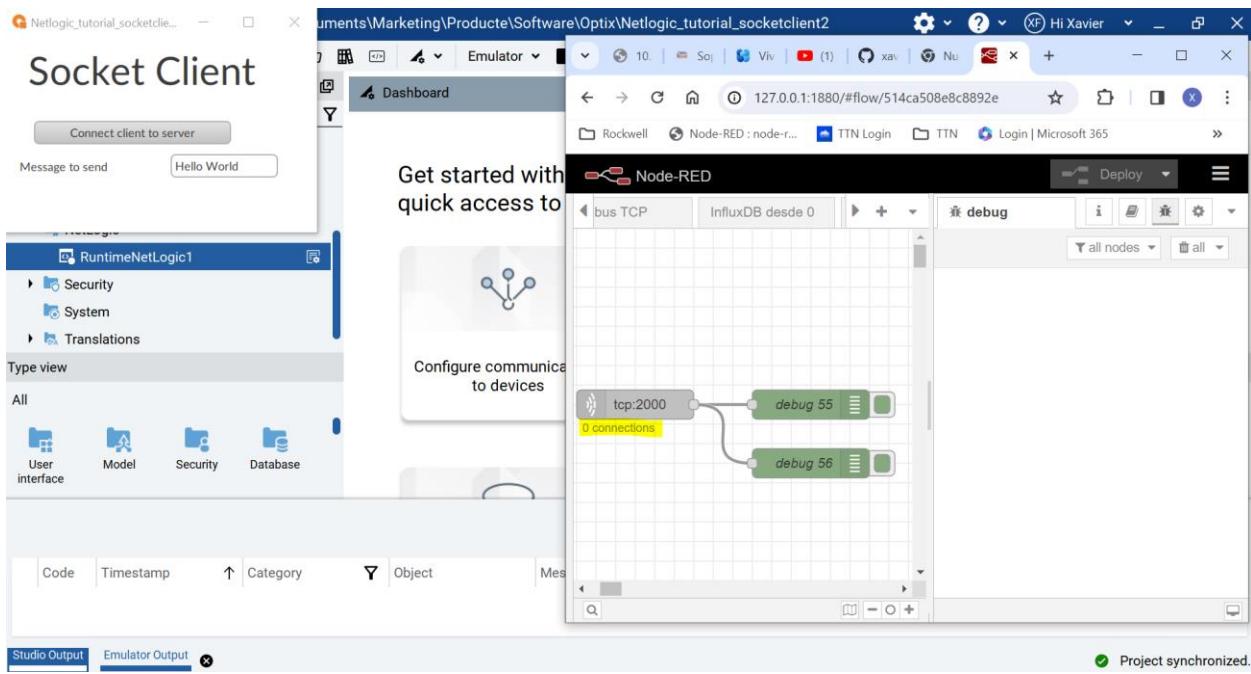
Now the connection is not closed. But each time we hit on the button a new socket is opened, and the message arrives on the new socket.

```

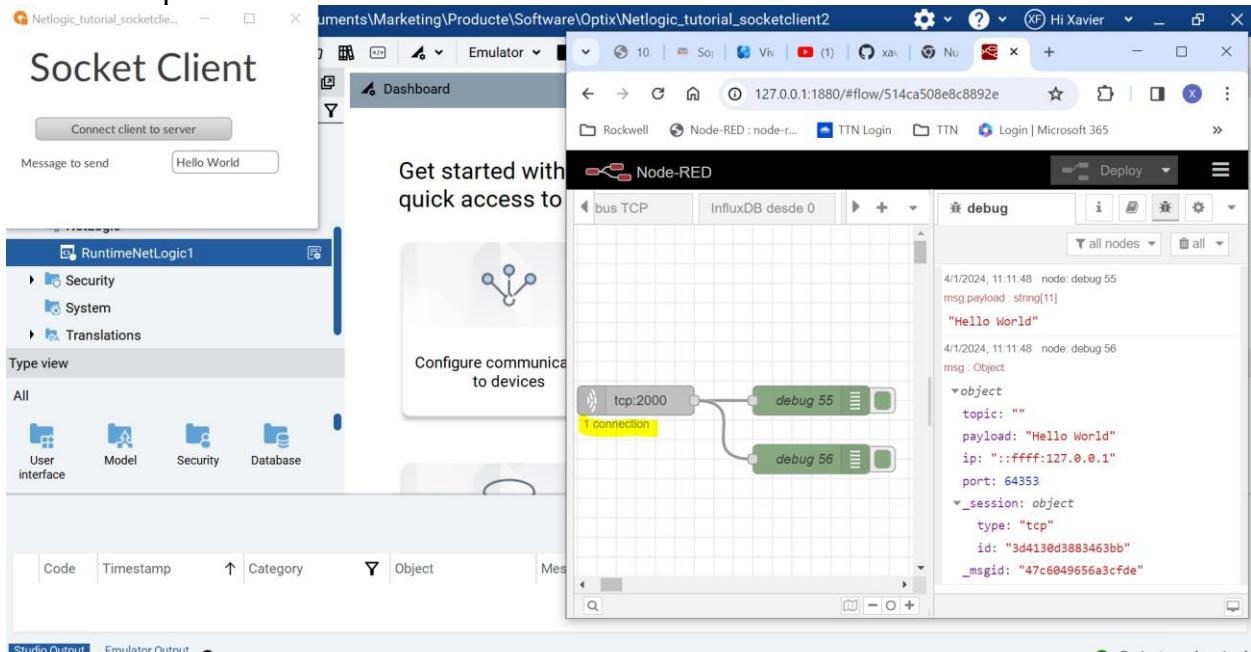
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void Socketclient_connect()
    {
        Log.Info("A button has been pressed");
        TcpClient client = new TcpClient("127.0.0.1", 2000);
        var missatge = Project.Current.GetVariable("Model/Message");
        //string messageToSend = "Hello World";
        string messageToSend = missatge.Value;
        int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
        byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
        NetworkStream stream = client.GetStream();
        stream.Write(sendData, 0, sendData.Length);
        //Console.WriteLine("sending data to server...");
        Log.Info("sending data to server...");
        //StreamReader sr = new StreamReader(stream);
        //string response = sr.ReadLine();
        //Console.WriteLine(response);
        //Log.Info(response);
        //stream.Close();
        //client.Close();
        //Console.ReadKey();
    }
}

```

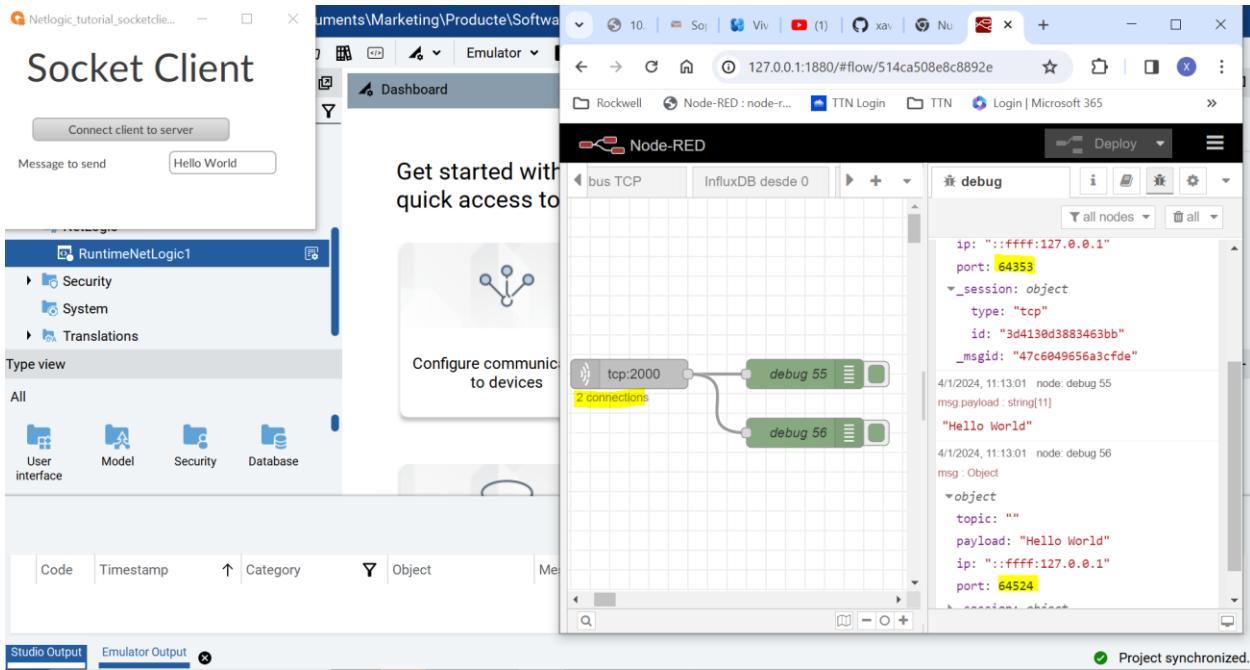
As we can monitor with Node-RED  
Before clicking button  
We have 0 connections



First button pressed 1 connection



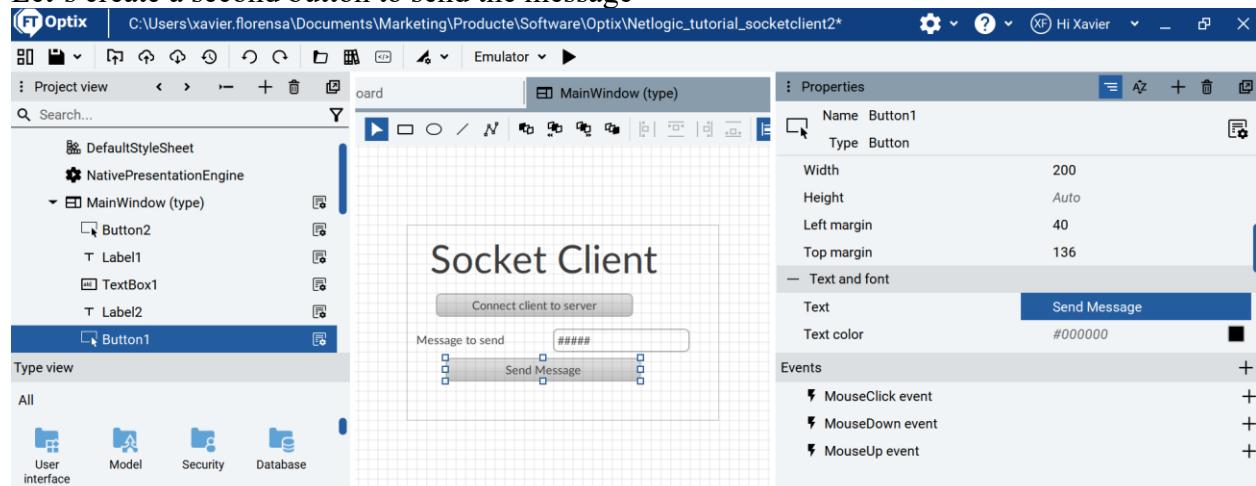
Second button pressed, 2 connections and two different ports



We do not want to have 2 connections.

Let's try to separate the connection and the message sending, with a second button to send the message

Let's create a second button to send the message



Let's create a new method with an input argument as we did on the server side.

And put the new client creation sentence outside of the first button, so the new client is created just when executing the application and the connect button is useless (we will improve this later on)

```
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    TcpClient client = new TcpClient("127.0.0.1", 2000);

    [ExportMethod]
    public void Socketclient_connect()
    {
```

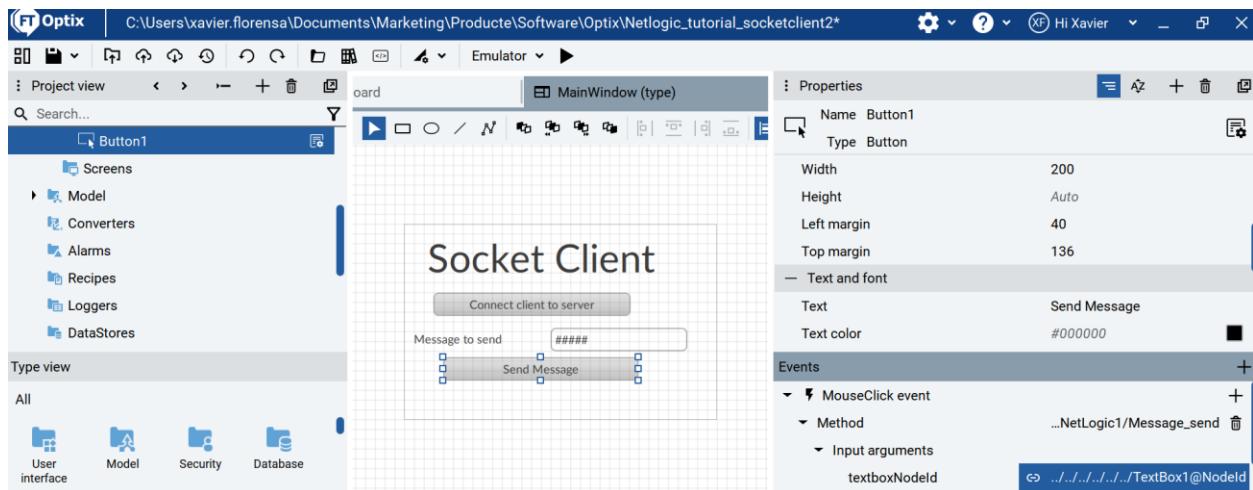
```

Log.Info("A button has been pressed");
//TcpClient client = new TcpClient("127.0.0.1", 2000);
//var missatge = Project.Current.GetVariable("Model/Message");
//string messageToSend = "Hello World";
//string messageToSend = missatge.Value;
//int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
//byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
//NetworkStream stream = client.GetStream();
//stream.Write(sendData, 0, sendData.Length);
//Console.WriteLine("sending data to server...");
//Log.Info("sending data to server...");
//StreamReader sr = new StreamReader(stream);
//string response = sr.ReadLine();
//Console.WriteLine(response);
//Log.Info(response);
//stream.Close();
//client.Close();
//Console.ReadKey();
}

public void Message_send(NodeId textboxNodeId)
{
    var textbox = InformationModel.Get<TextBox>(textboxNodeId);
    Log.Info("A button has been pressed");
    var missatge = textbox.Text;
    string messageToSend = missatge;
    int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
    byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
    NetworkStream stream = client.GetStream();
    stream.Write(sendData, 0, sendData.Length);
    //Console.WriteLine("sending data to server...");
    Log.Info("sending data to server...");
}
}

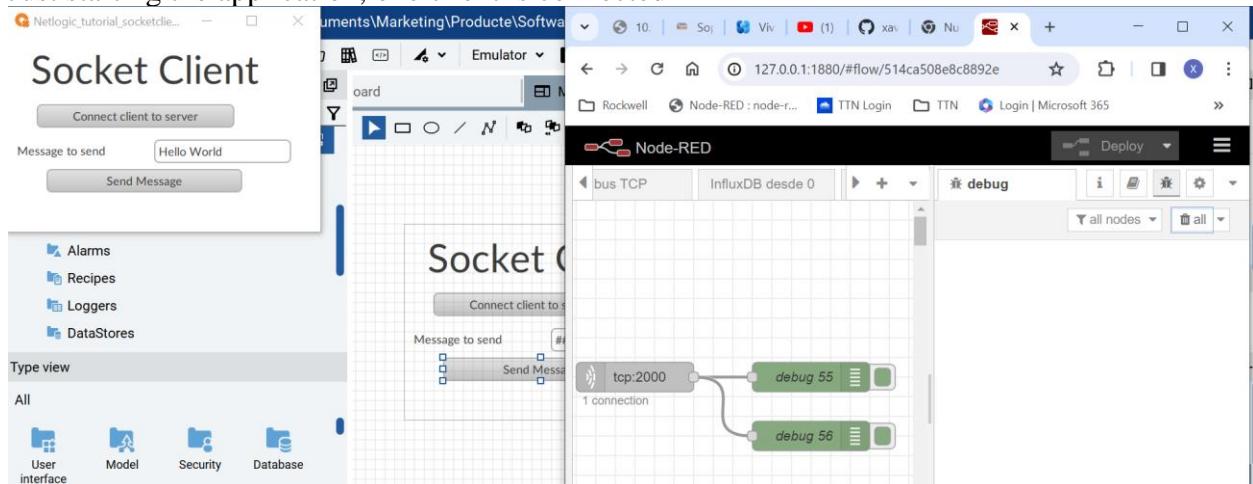
```

Save the code, erase the method action after pressing button 1 and create Method action after pushing button 2. Then create dynamic link between input parameter and textbox.

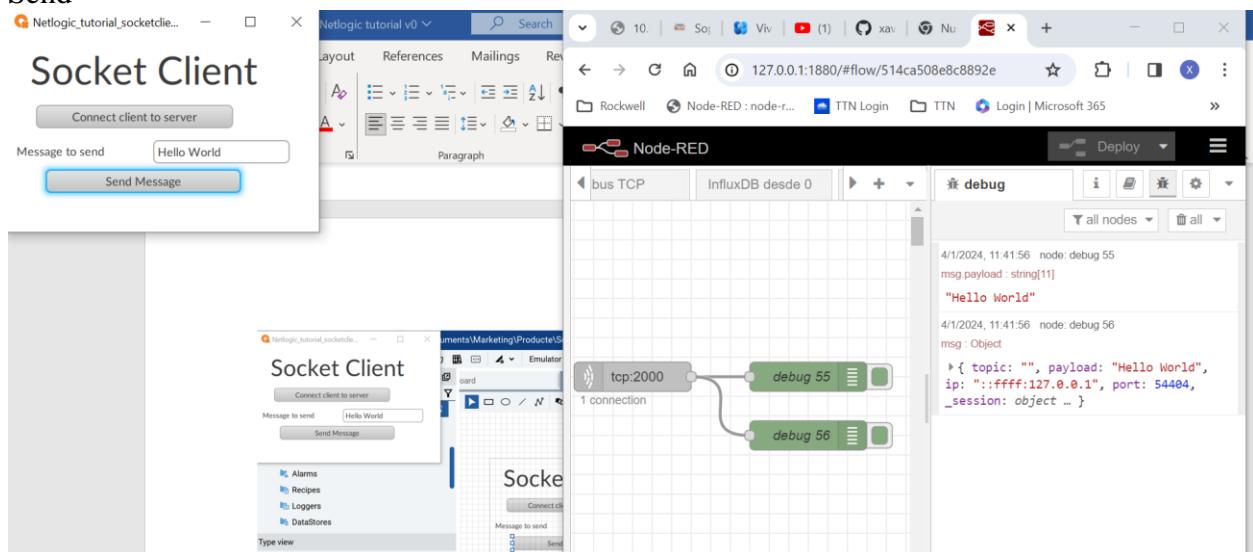


Test the application

Just starting the application, one client is connected



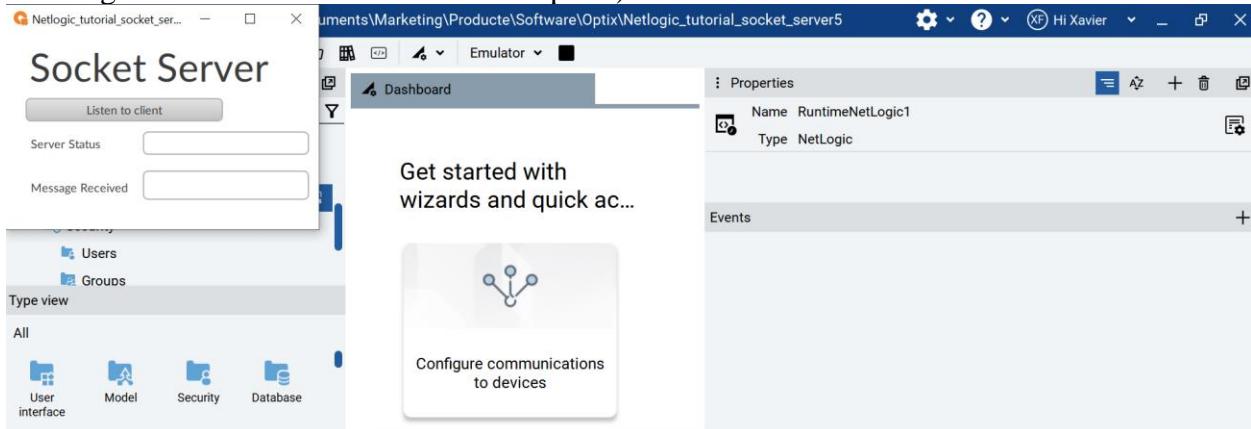
Send



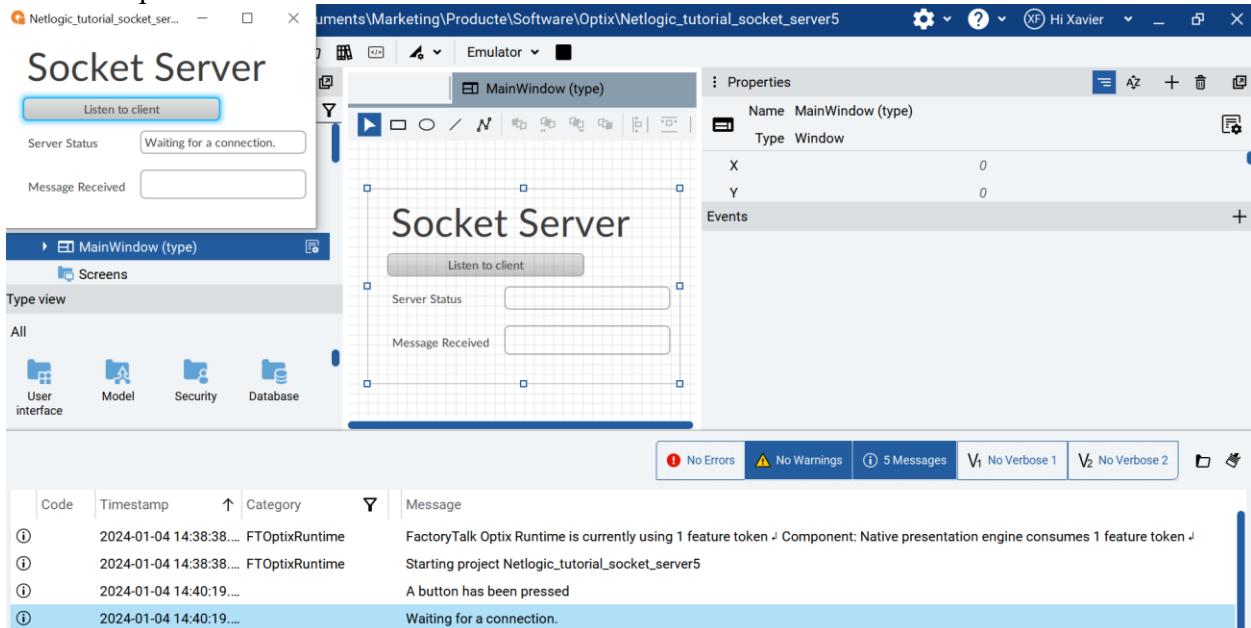
Second send, only one client is connected

Now let's try with our FactoryTalk Optix server

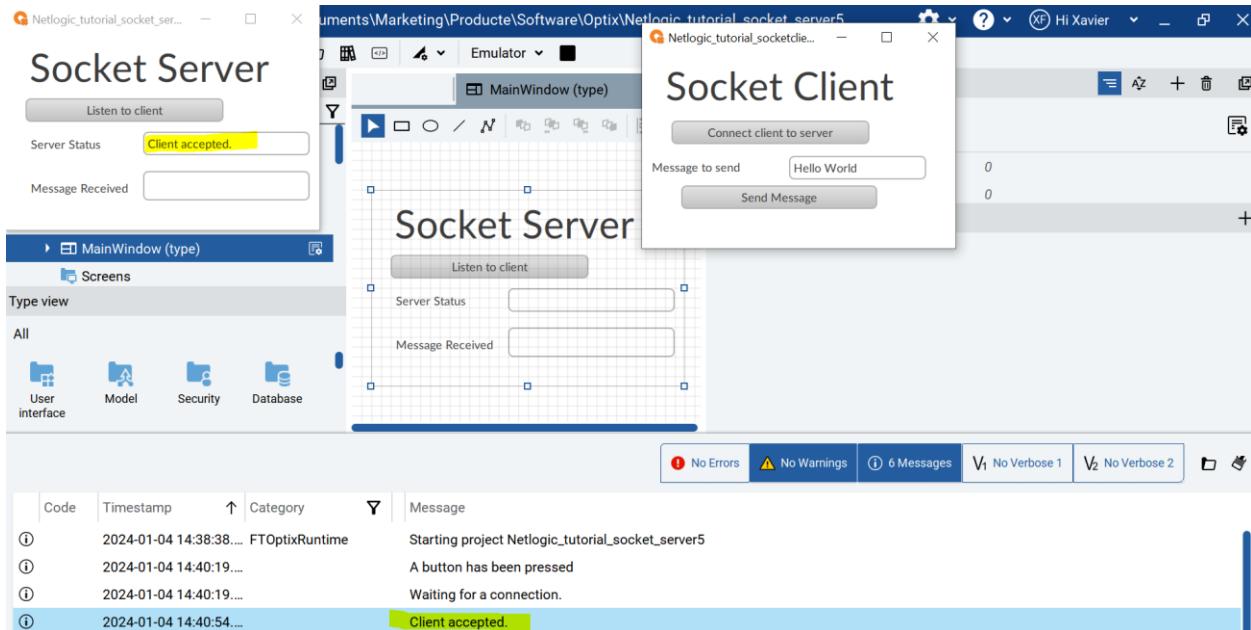
First we start only the server (since client makes connection just after starting, and if no server is listening then this will create an error exception)



Now let's put the server to listen



Now let's start the client



Now let's send the message

The socket is closed and there is no way to send more messages unless I stop and start the client. We should modify the server program in order not to close socket and wait for more connections. We modify the server code this way

```
#region Using directives
using System;
using UAMangedCore;
using OpcUa = UAMangedCore.OpcUa;
using FTOptix.HMIPrject;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen(NodeId textboxNodeId, NodeId textboxNodeId2)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
        textbox.Text = "A button has been pressed";
    }
}
```

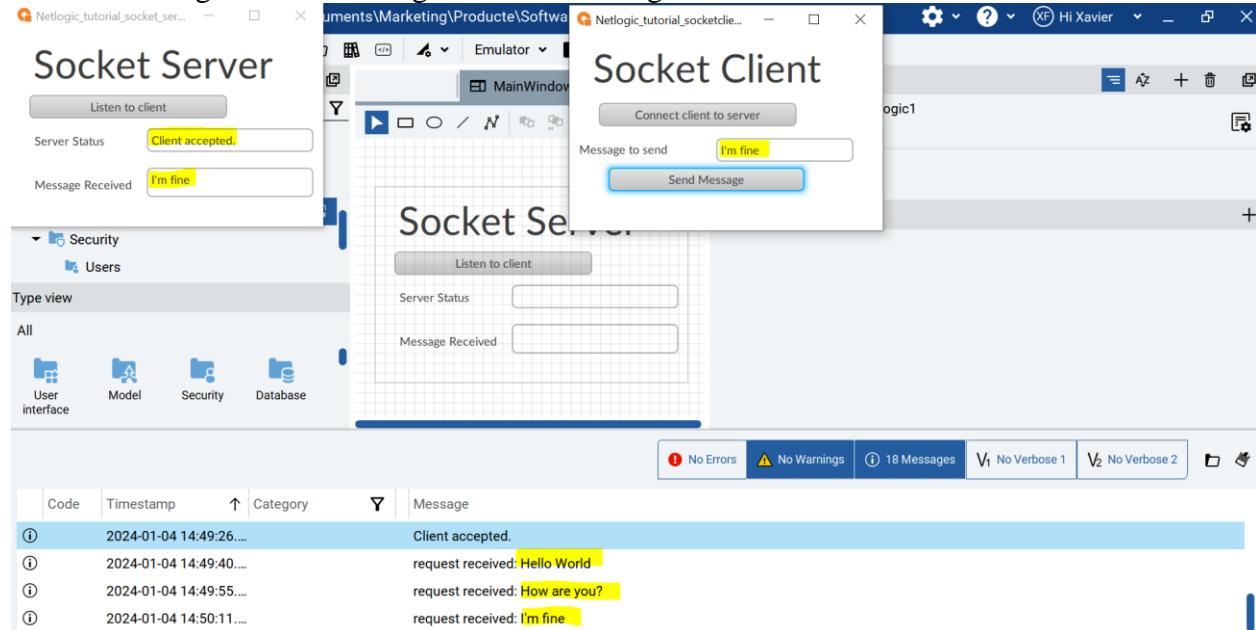
```

Log.Info("A button has been pressed");
TcpListener listener = new TcpListener(System.Net.IPEndPoint.Any, 2000);
listener.Start();
textbox.Text = "Waiting for a connection.";
Log.Info("Waiting for a connection.");
TcpClient client = listener.AcceptTcpClient();
//Console.WriteLine("Client accepted.");
textbox.Text = "Client accepted.";
Log.Info("Client accepted.");

while (true)
{
    NetworkStream stream = client.GetStream();
    StreamReader sr = new StreamReader(client.GetStream());
    StreamWriter sw = new StreamWriter(client.GetStream());
    try
    {
        byte[] buffer = new byte[1024];
        stream.Read(buffer, 0, buffer.Length);
        int recv = 0;
        foreach (byte b in buffer)
        {
            if (b!=0)
            {
                recv++;
            }
        }
        string request = Encoding.UTF8.GetString(buffer, 0, recv);
        //Console.WriteLine("request received: " + request);
        textbox2.Text = request;
        Log.Info("request received: " + request);
        sw.WriteLine("You rock!");
        sw.Flush();
    }
    catch(Exception e)
    {
        //Console.WriteLine("Something went wrong.");
        Log.Info("Something went wrong.");
        sw.WriteLine(e.ToString());
    }
    System.Threading.Thread.Sleep(1000);
}
}

```

This is working! Several messages without loosing the socket connection



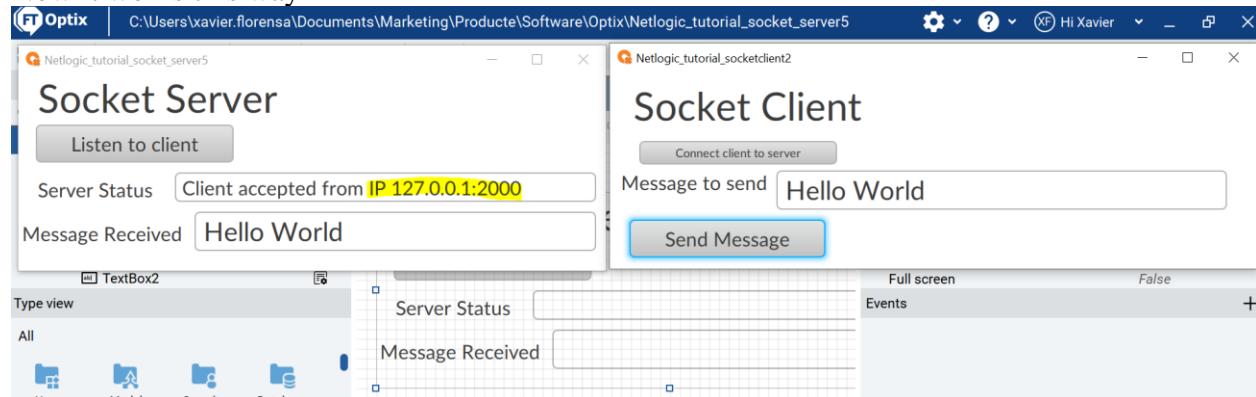
Still is the server difficult to stop (it hangs) so we still have to improve this.

Let's show the IP address and port from the connected client with this modification on the server address

```
[ExportMethod]
public void SocketServer_listen(NodeId textboxNodeId, NodeId textboxNodeId2)
{
    var textbox = InformationModel.Get<TextBox>(textboxNodeId);
    var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
    textbox.Text = "A button has been pressed";
    Log.Info("A button has been pressed");
    TcpListener listener = new TcpListener(System.Net.IPEndPoint.Any, 2000);
    listener.Start();
    textbox.Text = "Waiting for a connection.";
    Log.Info("Waiting for a connection.");
    TcpClient client = listener.AcceptTcpClient();
    //-----
    var clientIpLAN = client.Client.LocalEndPoint;
    Log.Info(clientIpLAN.ToString());
    //Console.WriteLine("Client accepted ");
    textbox.Text = "Client accepted from IP " + clientIpLAN.ToString() ;
    //-----
    Log.Info("Client accepted.");

    while (true)
```

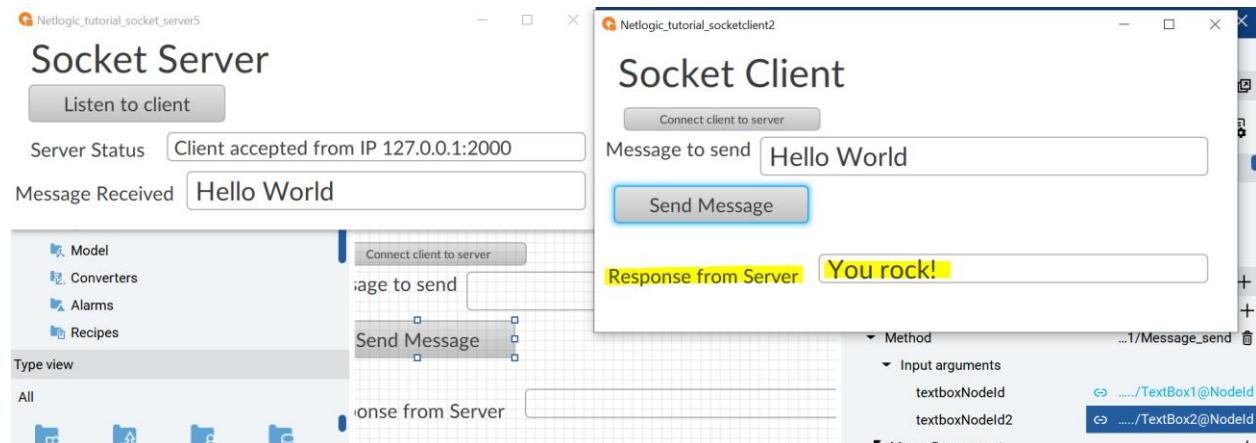
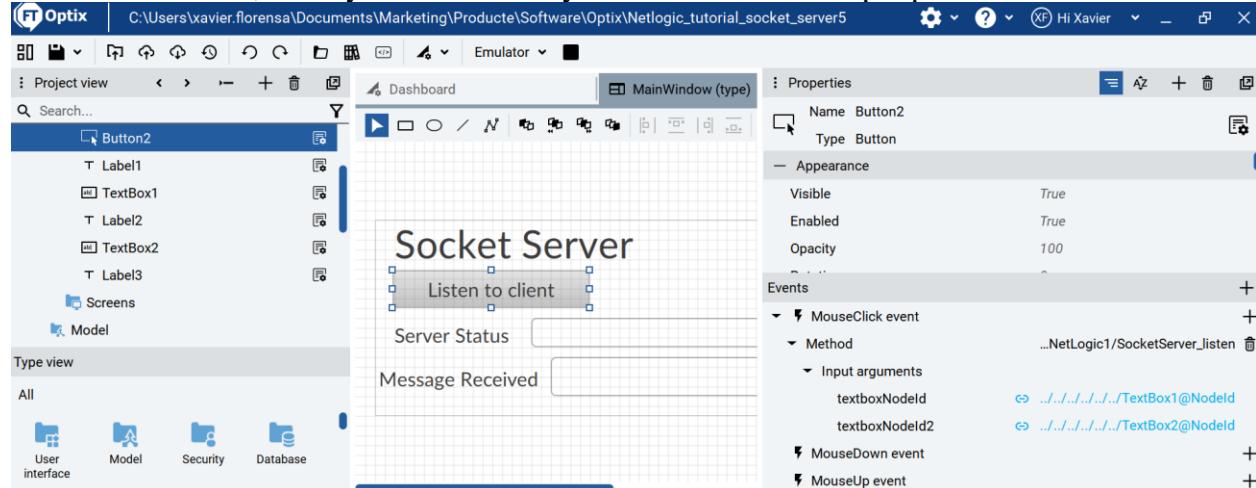
Now it works this way



Now seems that if you close the client first and then you close the server, it stops immediately. Even with an error. But this is OK since the socket is broken after you stop the client application. Next step will be to give a message on the client side that it is connected to..., and to make the connection after you hit a button.

To view responses from server on client side we have to modify the client code this way, and keep server as it was (server was sending message: "You rock!")

Add a new textbox, modify code and add dynamic link between input parameter2 and textbox2



### Client code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrjject;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    TcpClient client = new TcpClient("127.0.0.1", 2000);

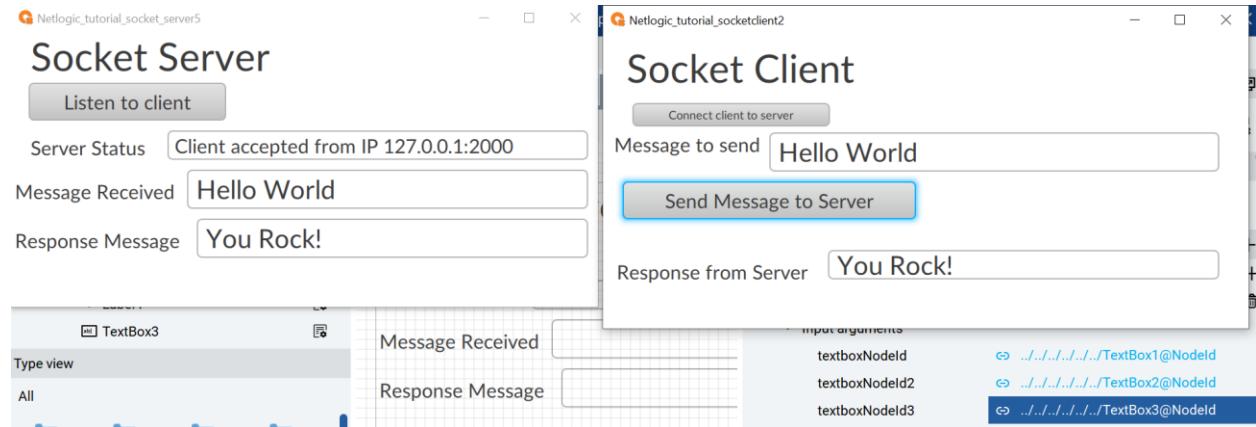
    [ExportMethod]

    public void Message_send(NodeId textboxNodeId,NodeId textboxNodeId2 )
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
        Log.Info("A button has been pressed");
        textbox.FontSize = 40;
        var missatge = textbox.Text;
        string messageToSend = missatge;
        int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
        byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
        NetworkStream stream = client.GetStream();
        stream.Write(sendData, 0, sendData.Length);
        //Console.WriteLine("sending data to server...");
        Log.Info("sending data to server...");
        StreamReader sr = new StreamReader(stream);
        string response = sr.ReadLine();
        //Console.WriteLine(response);
        Log.Info(response);
        textbox2.FontSize = 40;
        textbox2.Text = response;
    }
}
```

```
}
```

Now we will add a textbox to send a response message to the client

On the server



Modify the server code this way

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Net;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen(NodeId textboxNodeId, NodeId
    textboxNodeId2, NodeId textboxNodeId3)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
        var textbox3 = InformationModel.Get<TextBox>(textboxNodeId3);
        textbox3.FontSize = 40;
```

```

textbox.Text = "A button has been pressed";
Log.Info("A button has been pressed");
TcpListener listener = new TcpListener(System.Net.IPEndPoint.Any, 2000);
listener.Start();
textbox.Text = "Waiting for a connection.";
Log.Info("Waiting for a connection.");
TcpClient client = listener.AcceptTcpClient();
//-----
var clientIpLAN = client.Client.LocalEndPoint;
Log.Info(clientIpLAN.ToString());
//Console.WriteLine("Client accepted ");
textbox.Text = "Client accepted from IP " + clientIpLAN.ToString() ;
//-----
Log.Info("Client accepted.");

while (true)
{
    NetworkStream stream = client.GetStream();
    StreamReader sr = new StreamReader(client.GetStream());
    StreamWriter sw = new StreamWriter(client.GetStream());
    try
    {
        byte[] buffer = new byte[1024];
        stream.Read(buffer, 0, buffer.Length);
        int recv = 0;
        foreach (byte b in buffer)
        {
            if (b!=0)
            {
                recv++;
            }
        }
        string request = Encoding.UTF8.GetString(buffer, 0, recv);
        //Console.WriteLine("request received: " + request);
        textbox2.FontSize =40;
        textbox2.Text = request;
        Log.Info("request received: " + request);
        //sw.WriteLine("You rock!");
        sw.WriteLine(textbox3.Text);
        sw.Flush();
    }
    catch(Exception e)
    {
        //Console.WriteLine("Something went wrong.");
        Log.Info("Something went wrong.");
    }
}

```

```
        sw.WriteLine(e.ToString());
    }
    System.Threading.Thread.Sleep(1000);
}
}
```

Next step is to show a message on Client to tell that it is connected.  
This can be made locally on the client, or as a response from the Server.  
Let's do it as a local connection status.

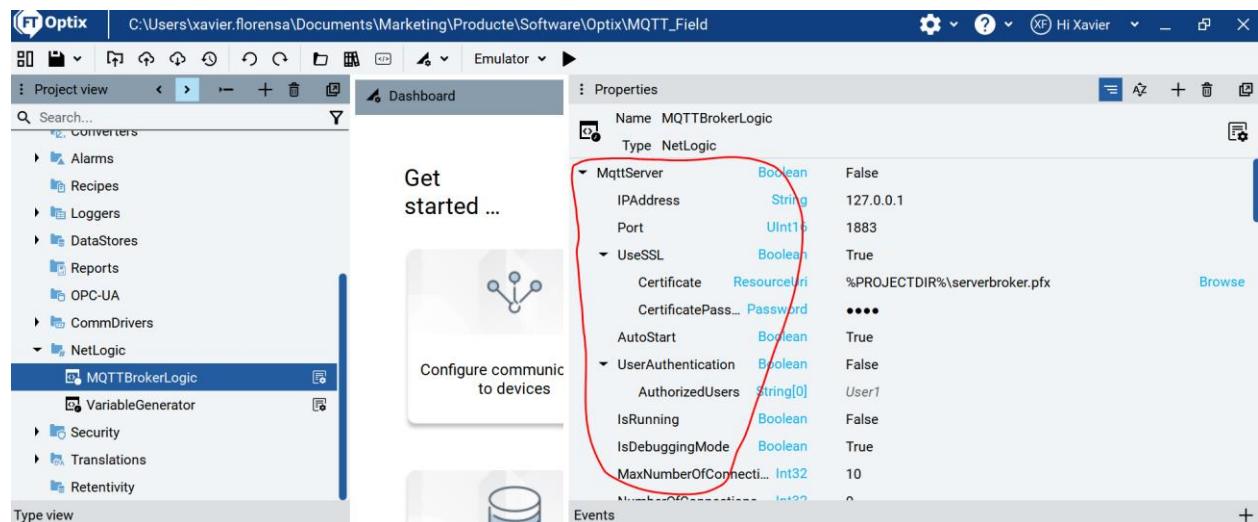
Next Step will be selecting the target IP addresses of the client.

## 42. How to define NetLogic Parameters

This is useful when we have a NetLogic script that needs some user customization and we do not want the user to enter in C# editor.

Take a look at some examples from Rockwell repository.

In some of them, when you click on Netlogic, you will see some parameters  
For instance



We see that it corresponds to a structure

```

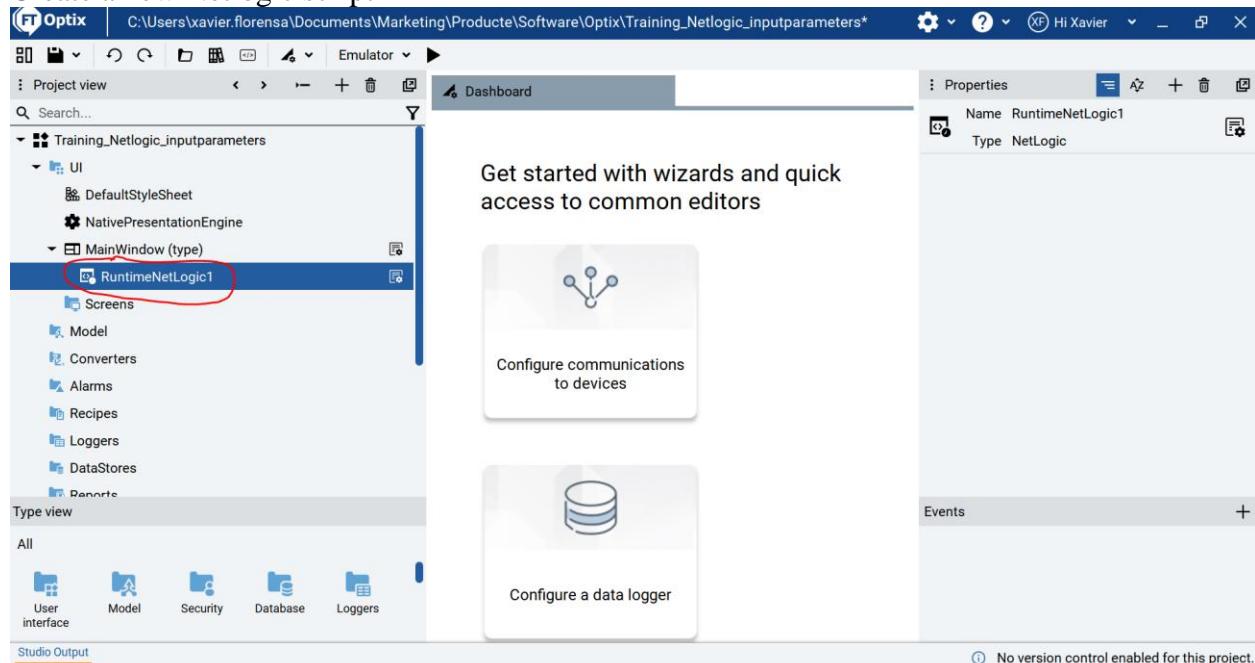
2416     private struct MQTTServerParameters
2417     {
2418         public IPAddress ipAddress;
2419         public ushort port;

```

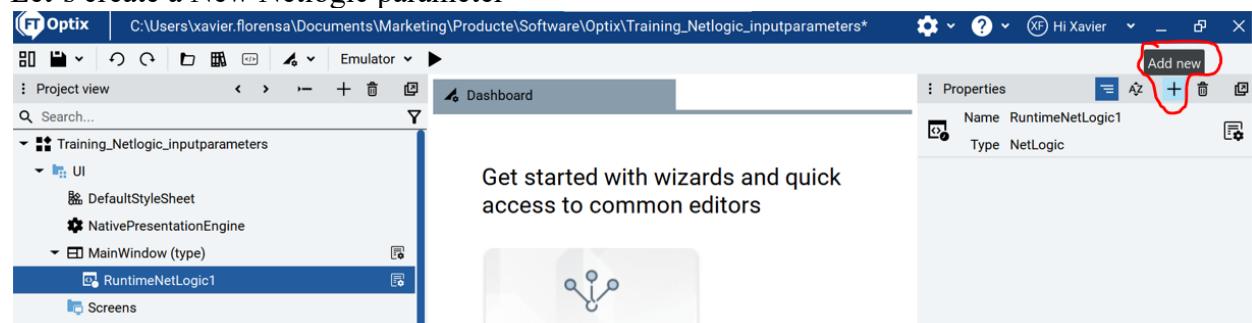
But let's see this with a simple example

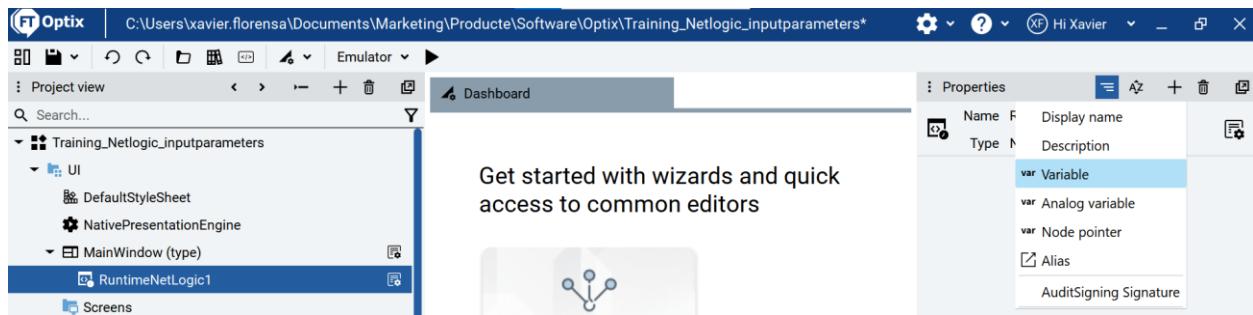
Let's create a new project

Create a new Netlogic script

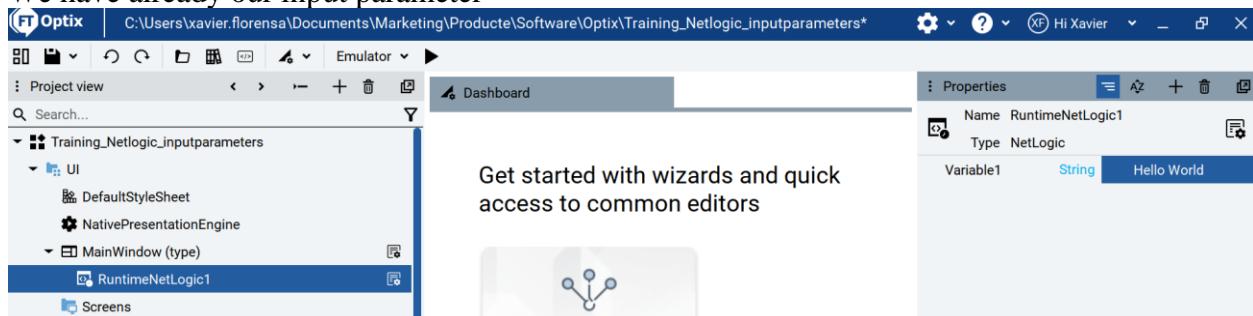


Let's create a New Netlogic parameter



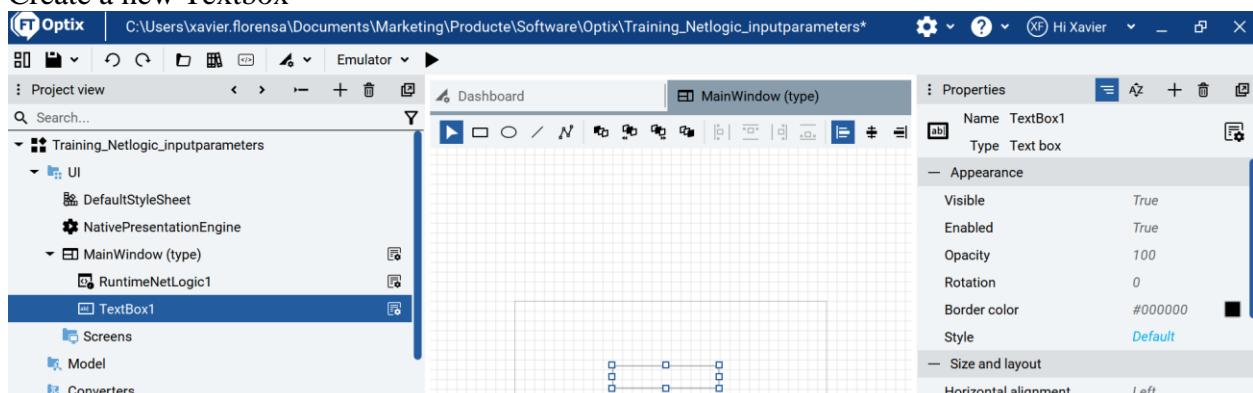


We have already our input parameter



Now let's go to access this value from C# and display display the value on a Textbox

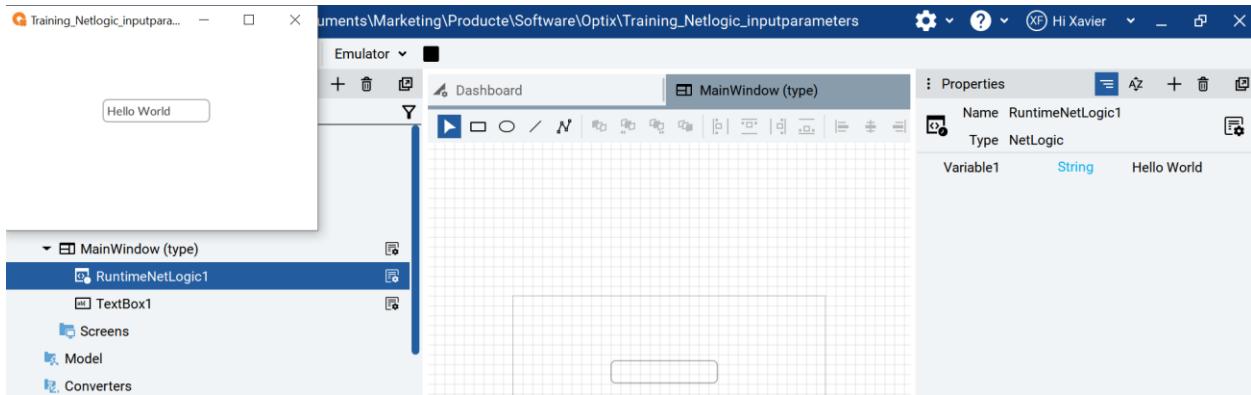
Create a new Textbox



Copy and paste this code on the Netlogic script

```
public override void Start()
{
    // Insert code to be executed when the user-defined logic is started
    var textBox1 = Owner.Get<TextBox>("TextBox1");
    var parameter = LogicObject.GetVariable("Variable1");
    textBox1.Text=parameter.Value;
}
```

Test the application



## 43. Using third party dll libraries in FactoryTalk Optix

You can see the result here

<https://youtu.be/6E4xt02Ohzw>

You can use third party libraries.

For instance Winsock\_Orcas.dll

Even though this library was created for Visual Basic, since it is compiled, it can be also used with C#

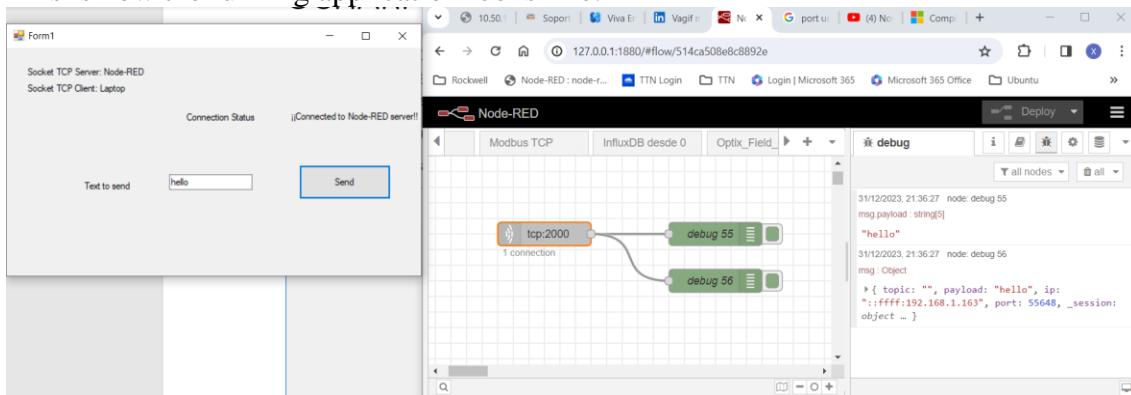
<https://www.codeproject.com/Articles/21443/Winsock-Revamped>

This DLL is tested in Visual Studio 2022. With this code in Visual Studio 2022

### 43.1. Using third party libraries in Visual Studio 2022 windows Forms Framework

TCP/IP PC to Node-RED communications using this routine written in C# (C sharp) and the socket dll library called Winsock orcas

This is how the running application looks like:



About C# programming language

[https://es.wikipedia.org/wiki/C\\_Sharp](https://es.wikipedia.org/wiki/C_Sharp)

About winsock

<https://en.wikipedia.org/wiki/Winsock>

About Winsock orcas dll library by Chris Kolkman

"Winsock Orcas is the result of refining a project that started when I graduated from VB6 to VB.NET 2003."

.NET no longer supported the old Winsock control that had been so easy to use in VB6. Instead they gave us something with much more power, but also much more complexity: the Socket.

It took me a bit of time to figure the socket out, but when I did I decided to create a wrapper that worked just like the old control I was familiar with - making sockets much easier.

The first version was wrought with bugs and wasn't thread-safe. When VS 2005 came out, and revealed even more functions with regards to the socket - I resolved to make a new version.

That was Winsock 2005. It was thread-safe (to a point), and fixed the major bugs of the previous version. It even had UDP support.

In April of 2007 I started work on Winsock 2007. Due to a project I was working on at the time, I was looking into Remoting to synchronize an object between server/client. I decided Remoting wasn't for my project (couldn't implement blacklist), thus a new version of Winsock was born.

Winsock 2007 enabled synchronizing of objects (via BinaryFormatter), making the Send/Get routines simpler. The thread-safe events have been improved, as has UDP support. IPv6 support was added making it much easier to use with Vista.

Winsock Orcas (version 4.0.0) was made just to keep this going. It had come to my attention that VS2008 had problems compiling the code for previous version, so I made this version. This version streamlines the code, making it simpler to read (mainly by removing the WinsockMonitor class), and also adds in some Generics support on the Get/Peek methods to do automatic conversion to the type you want (watch out, you could cause exceptions for casting to the wrong type).

## Why to use this dll?:

to create your application easily thus forgetting about communications layer, letting the Visual Studio environment write the code for you.

Customer normally uses Microsoft Visual Studio environment to edit, compile and run the application.

<https://docs.microsoft.com/en-us/visualstudio/#pivot=features>

This is the code:

But do not copy and paste, you have to put the controls on the form and then edit little changes.  
See below.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Socket_test1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //winsock_Ear.Listen(2000); //This is to make the PC act as host
            winsock_Ear.Connect("192.168.1.163", 2000); //This is to make the PC act
as client
        }

        private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsockConnectedEventArgs e)
        {
            label1.Text = ";;Connected to Node-RED server!!";
        }
    }
}
```

```

        }

    private void winsock_Ear_ConnectionRequest(object sender,
Winsock_Orcas.WinsocConnectionRequestEventArgs e)
{
    winsock_Ear.Close();
    winsock_Ear.Accept(e.Client);

}

private void button1_Click(object sender, EventArgs e)
{
    //we get the text from the Textbox entered from the keyboard
    string text_to_send = this.textBox1.Text;
    winsock_Ear.Send(text_to_send);
}
}

}

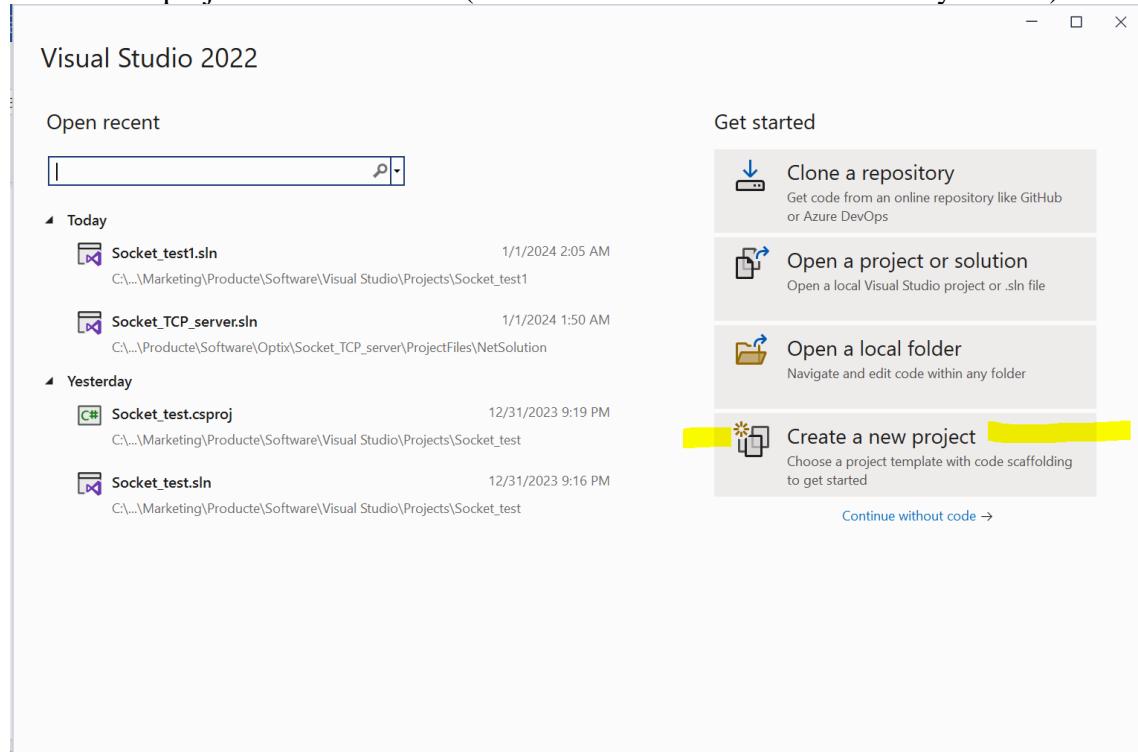
```

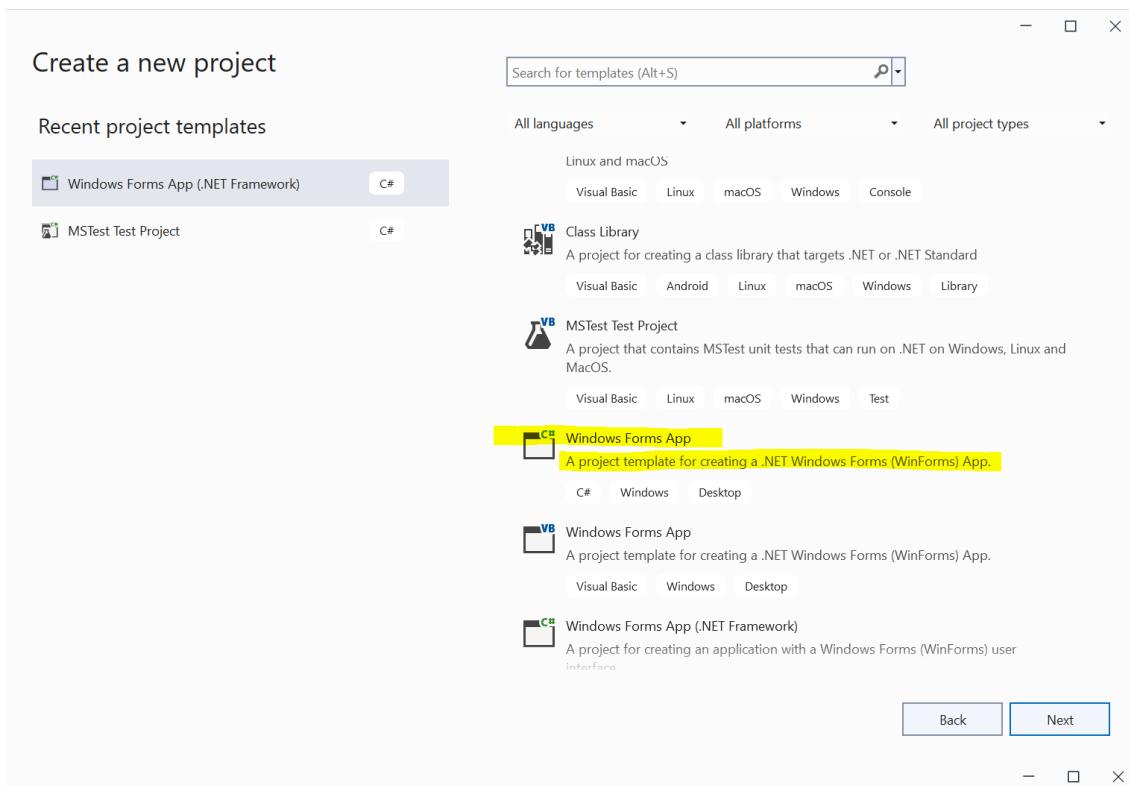
How to proceed if you want to use this code for your own applications:

Copy the Winsock Orcas dll on a known location on your hard disk

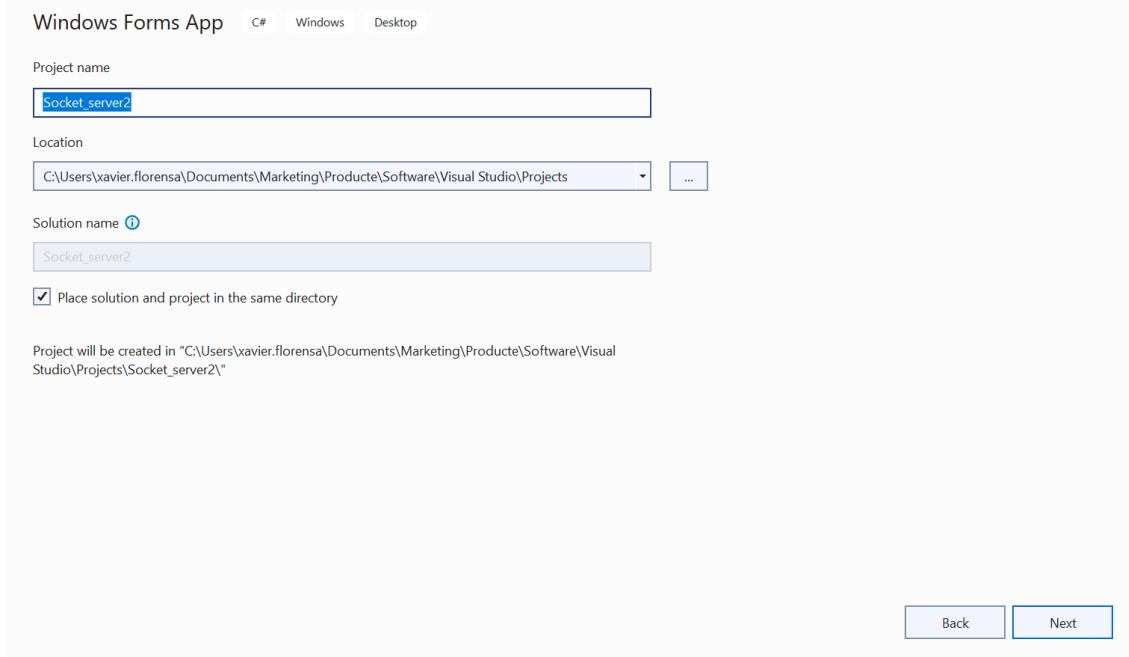
Open Visual studio

Create new project in visual studio (this is Visual studio 2022 community edition)

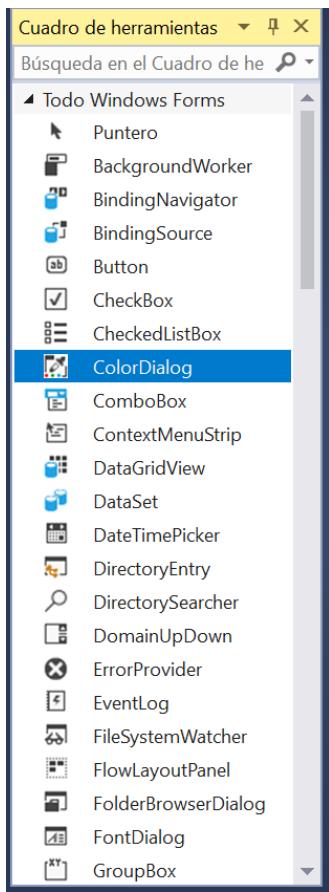




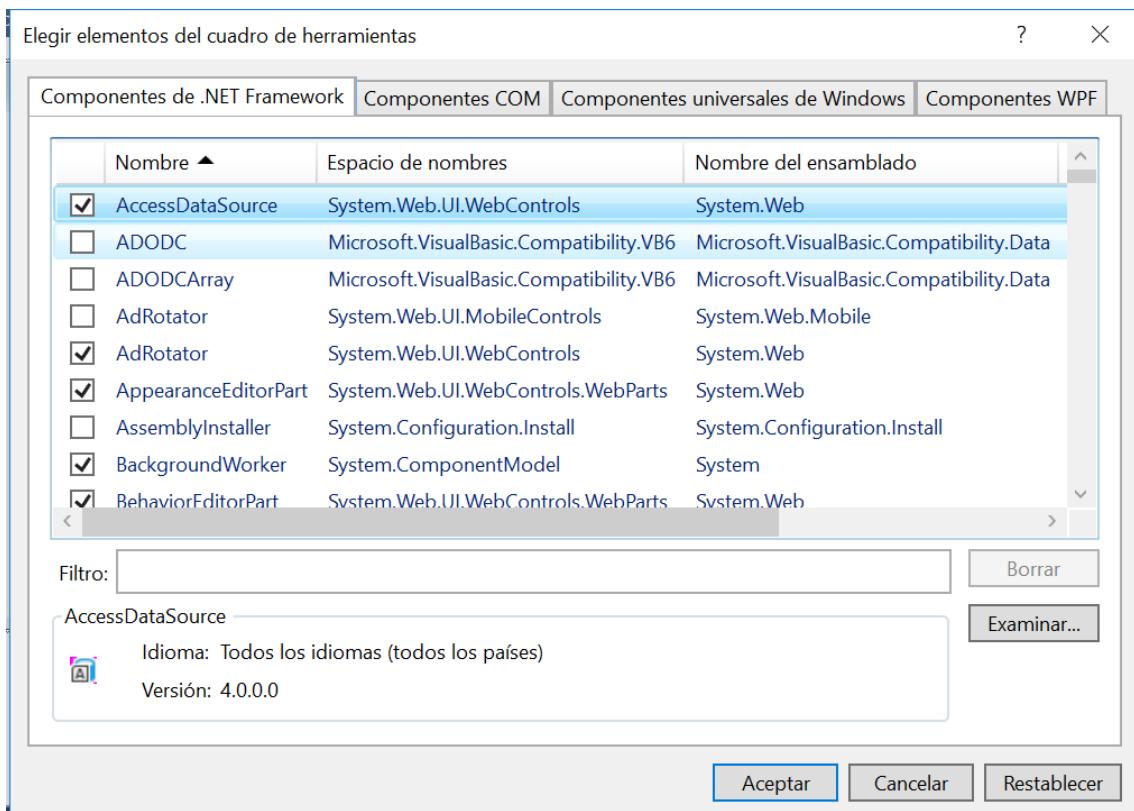
## Configure your new project



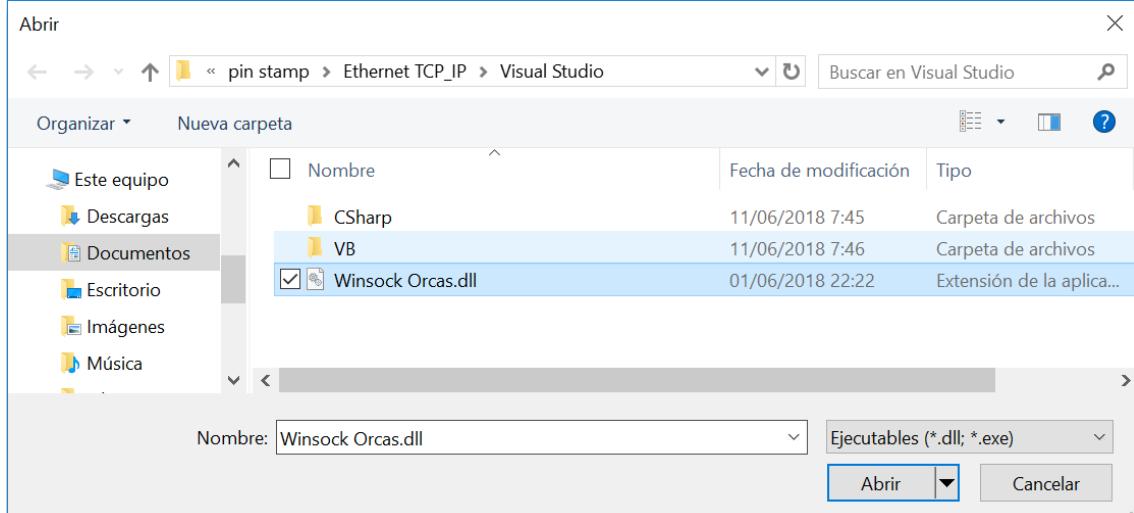
Once created,  
then right click the mouse anywhere on your toolbox list to add the Winsock Orcas control for  
easy programming



Then click on “Choose elements”



Click on “Examinar” to locate the path to the dll on your computer



Open it and accept

Elegir elementos del cuadro de herramientas

?

X

Componentes de .NET Framework Componentes COM Componentes universales de Windows Componentes WPF

	Nombre ▲	Espacio de nombres	Nombre del ensamblado
<input type="checkbox"/>	VScrollBarArray	Microsoft.VisualBasic.Compatibility.VB6	Microsoft.VisualBasic.Compatibility
<input checked="" type="checkbox"/>	WebBrowser	System.Windows.Forms	System.Windows.Forms
<input type="checkbox"/>	WebBrowserArray	Microsoft.VisualBasic.Compatibility.VB6	Microsoft.VisualBasic.Compatibility
<input type="checkbox"/>	WebClient	System.Net	System
<input checked="" type="checkbox"/>	WebPartManager	System.Web.UI.WebControls.WebParts	System.Web
<input checked="" type="checkbox"/>	WebPartZone	System.Web.UI.WebControls.WebParts	System.Web
<input type="checkbox"/>	WebService	System.Web.Services	System.Web.Services
<input checked="" type="checkbox"/>	Winsock	Winsock_Orcas	Winsock Orcas
<input checked="" type="checkbox"/>	Wizard	System.Web.UI.WebControls	System.Web

Filtro:

Borrar

Winsock

Examinar...



Idioma: Todos los idiomas (todos los países)

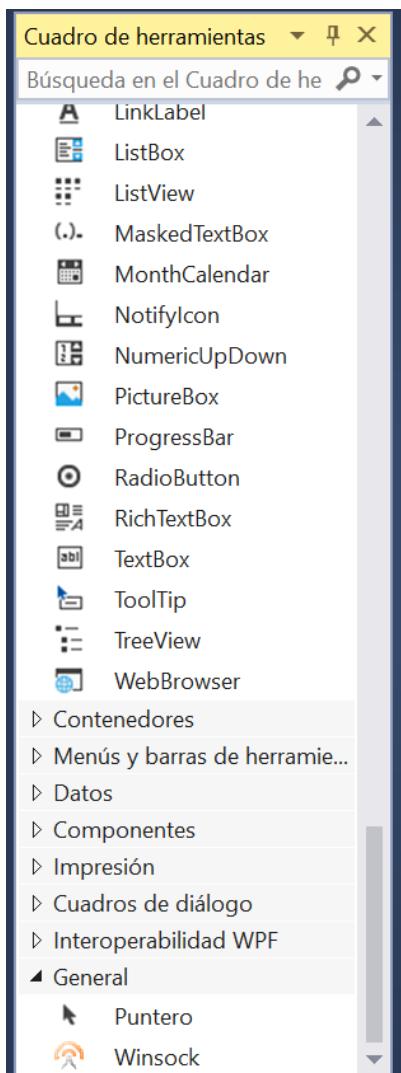
Versión: 4.0.0.0

Aceptar

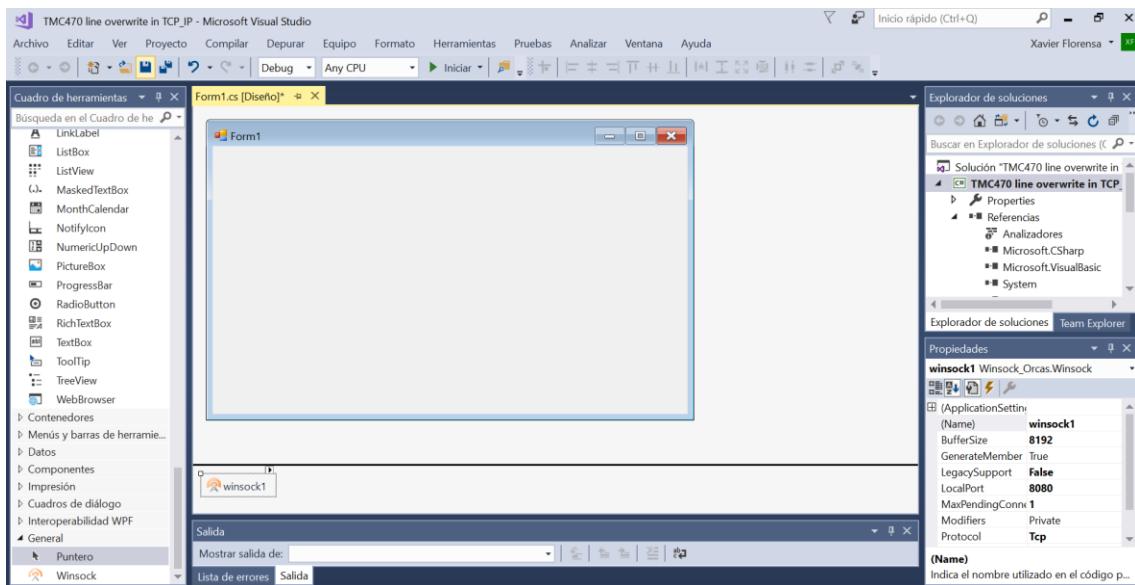
Cancelar

Restablecer

You will find it now at the end of your toolbox list

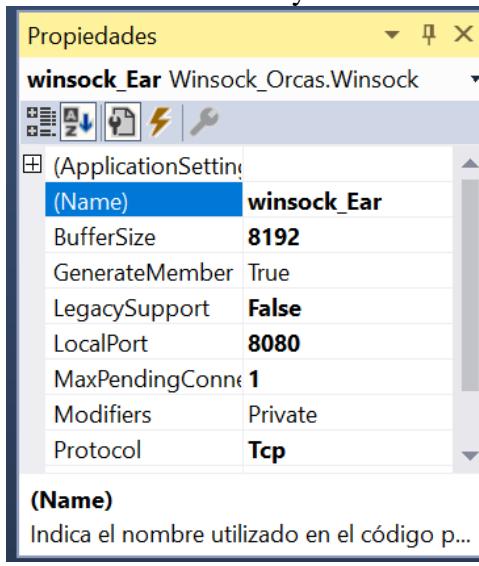


Click on the Winsock control icon and drag until your form  
Then you will see it inserted on your form (just below)



**Click on the arrow over the Winsock Orcas control on the form. Be sure to select the option enable Legagy (If not it will not be able to write data)**

Rename the control if you want on the Properties (having selected the control)



Double click on your form to see the generated code

```

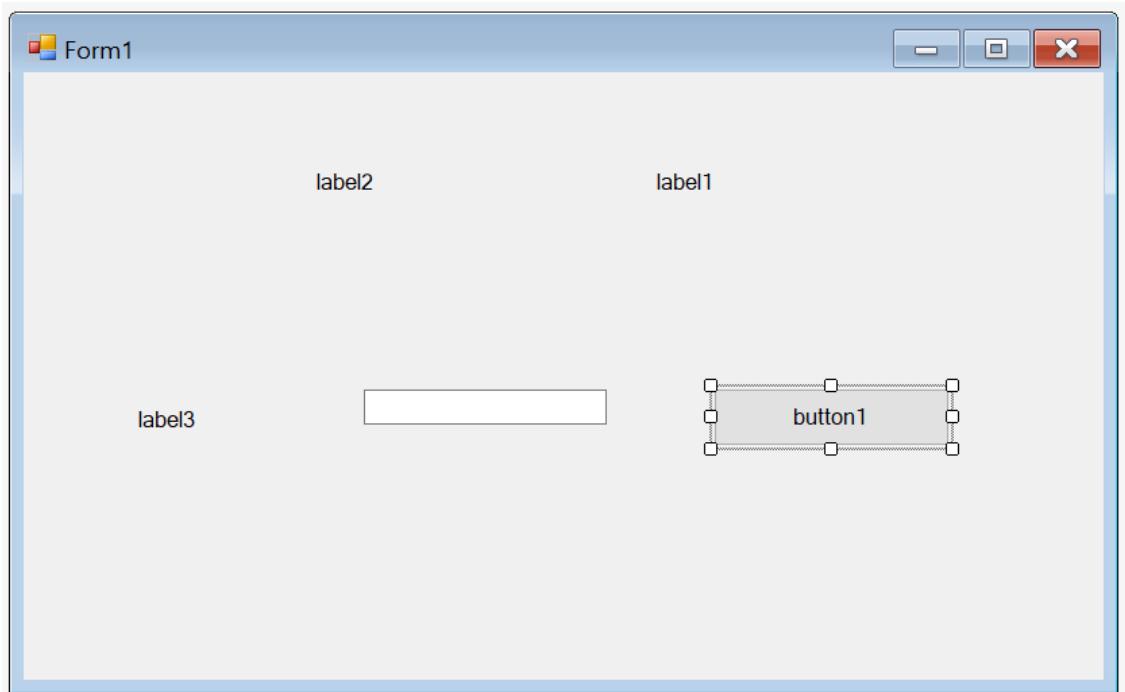
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10
11 namespace TMC470_line_overwrite_in_TCP_IP
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19
20         private void Form1_Load(object sender, EventArgs e)
21         {
22         }
23     }
24 }
25
26

```

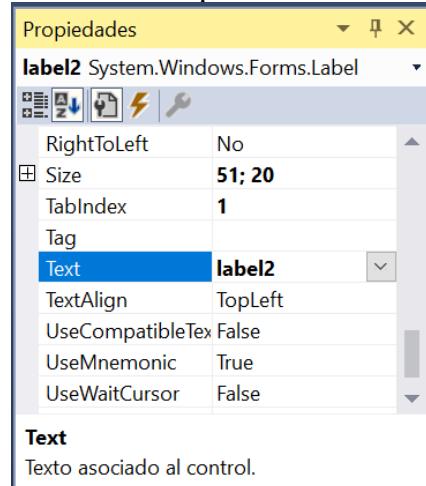
We move now to the form view

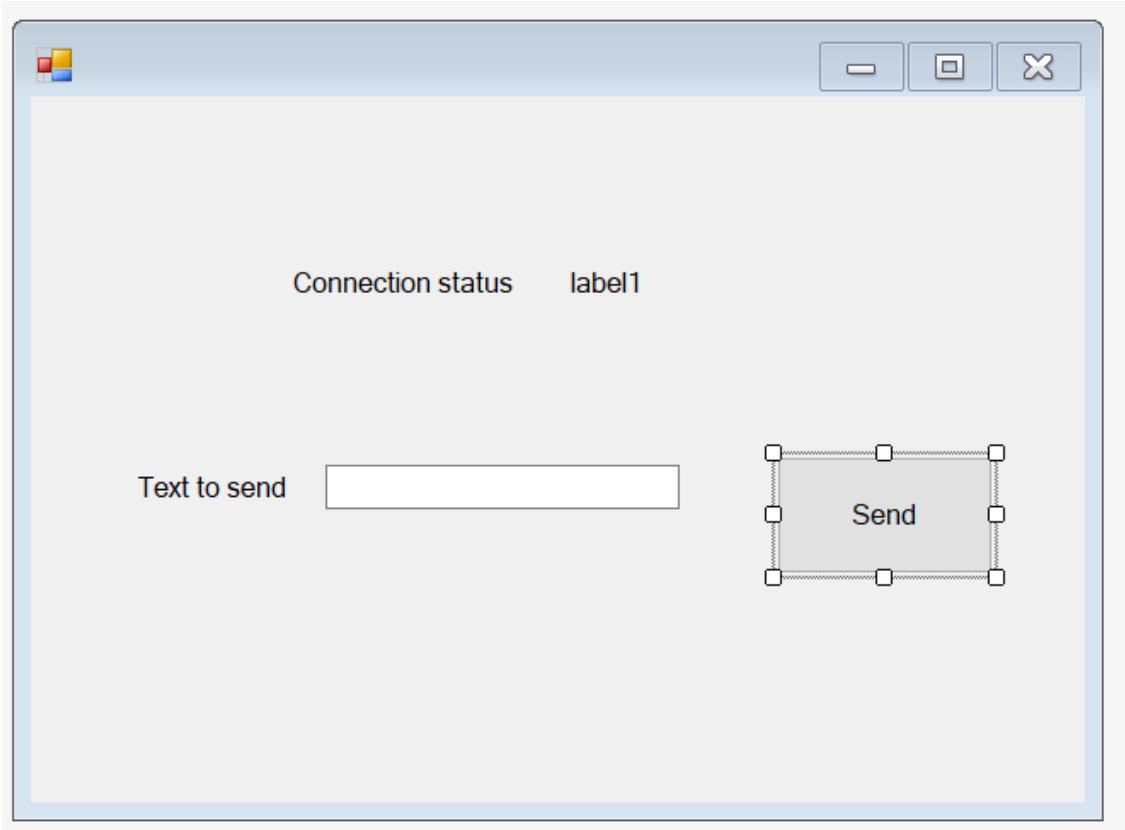
Then we start populating the form with the desired controls (click on the controls list and drag on the form)

In this case three labels, one text box and one button



Rename the captions for labels 2 and 3 and button as you want.





Now we will introduce three subroutines to get the connection

Click on the winsok\_Ear control on your form and select “Connected” from the methods list

This will add this code for you (without typing anything)

```
private void winsock_Ear_Connected(object sender, Winsock_Orcas.WinsockConnectedEventArgs e)
{
}
```

Then put the message that will appear when the TMC470 is connected, like this

```
private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsockConnectedEventArgs e)
{
    label1.Text = ";;Connected to Node-RED server!!";
}
```

Complete the default subroutine

```
private void Form1_Load(object sender, EventArgs e)
{
}
```

In this way, by typing (or copy / paste) this content

Be aware that the Laptop has this IP address per DHCP: **192.168.1.163**

```
private void Form1_Load(object sender, EventArgs e)
{
```

```

    //winsock_Ear.Listen(2000); //This is to make the PC act as host
    winsock_Ear.Connect("192.168.1.163", 2000); //This is to make the PC act as
client connecting to port 2000 of server
}

```

Then click on the Winsoc\_Ear control on your form and select “ConnectionRequest”  
 This will add this code to your program without typing anything:

```

private void winsock_Ear_ConnectionRequest(object sender,
WinsockConnectionEventArgs e)
{
}

```

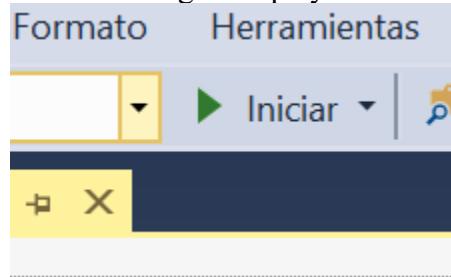
Then complete by typing this content

```

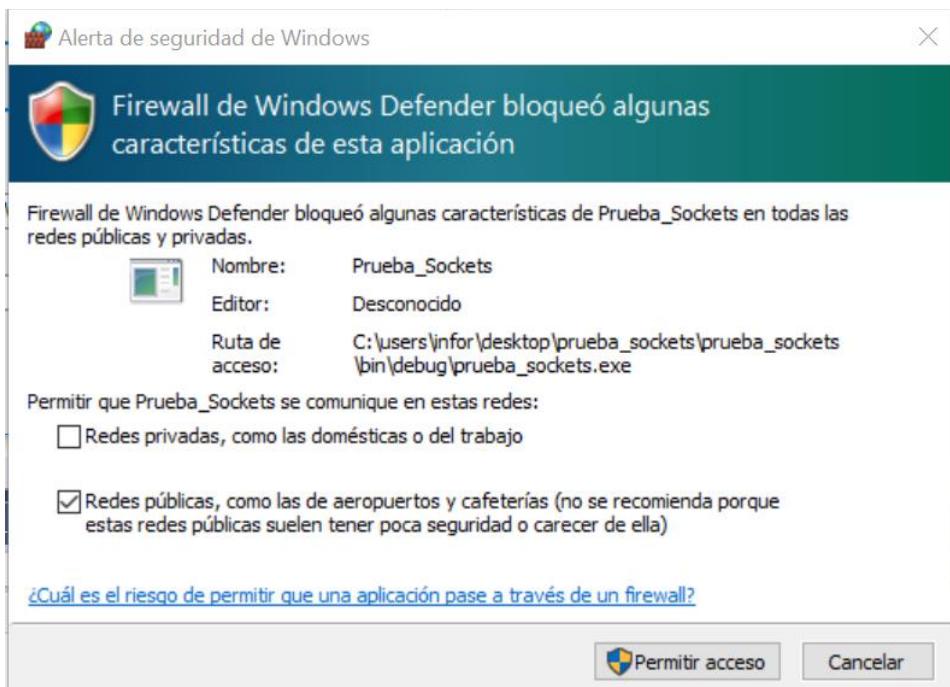
private void winsock_Ear_ConnectionRequest(object sender,
WinsockConnectionEventArgs e)
{
    winsock_Ear.Close();
    winsock_Ear.Accept(e.Client);
}

```

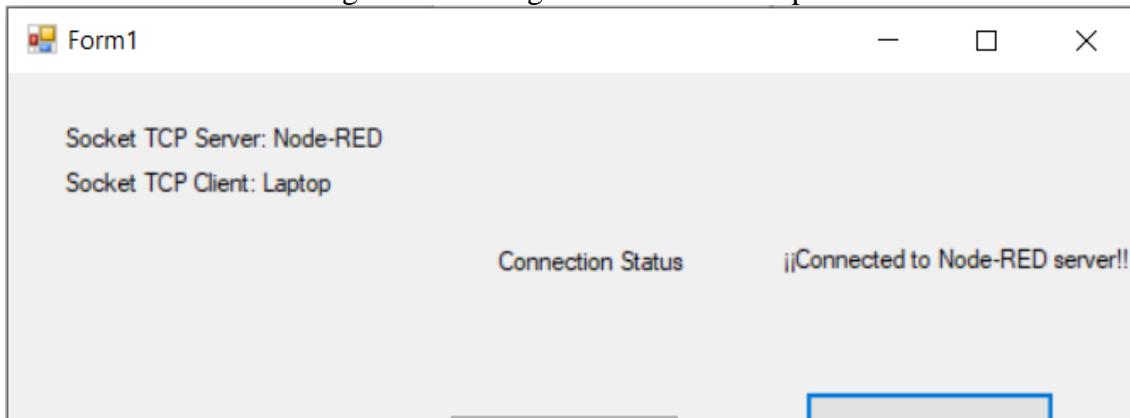
Now you can test the application to see if you get connection  
 Just click on green “play” arrow



First time you will see this message:



It's your firewall. Click on accept, otherwise you will not be able to test the application. You should see this message after waiting between few and up to 30 seconds



Let's program the rest of the application in order to send data  
Double click on the Button control on your form

This action will introduce this code without typing anything

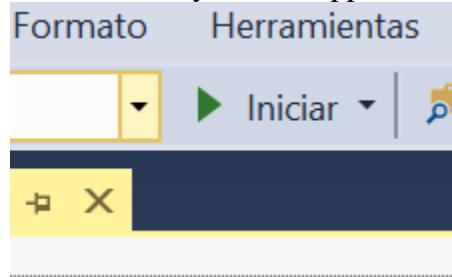
```
private void button1_Click(object sender, EventArgs e)
{
}
```

Then type (or copy / paste) this content

```
private void button1_Click(object sender, EventArgs e)
{
    //we get the text from the Textbox entered from the keyboard
    string text_to_send = this.textBox1.Text;
    winsock_Ear.Send(text_to_send);
```

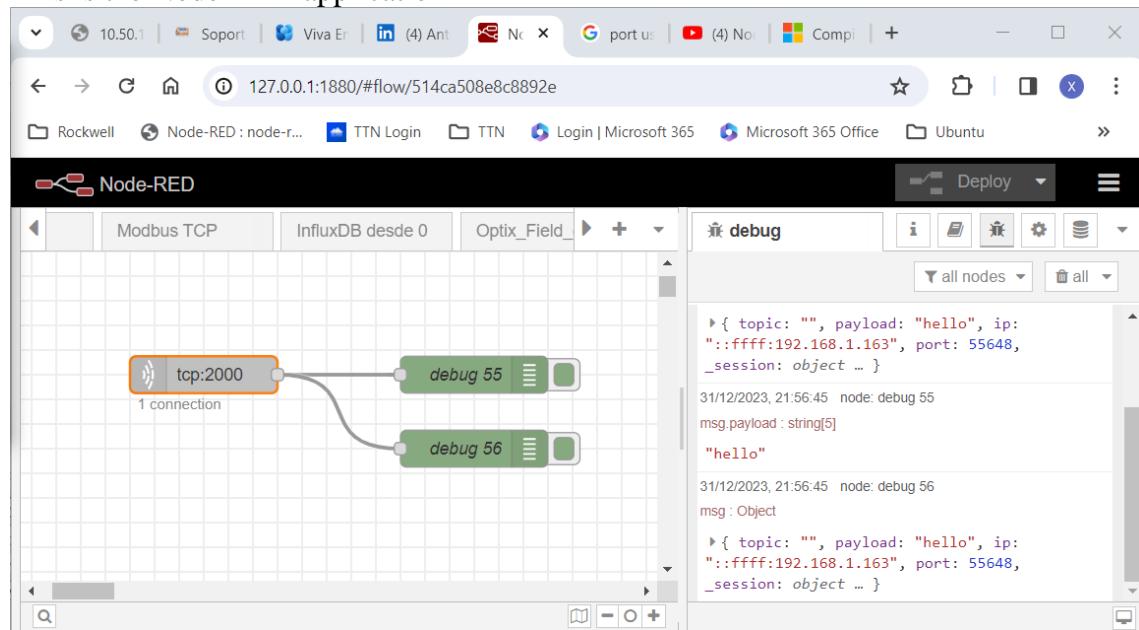
}

You can finally test the application



And that's all!

This is the Node-RED application



### Edit tcp in node

[Delete](#) [Cancel](#) [Done](#)

**Properties**

Type Listen on port 2000  
 Enable secure (SSL/TLS) connection

Output stream of String payload(s)  
delimited by (optional)  
 re-attach delimiter

Topic Topic

Name Name

### Edit debug node

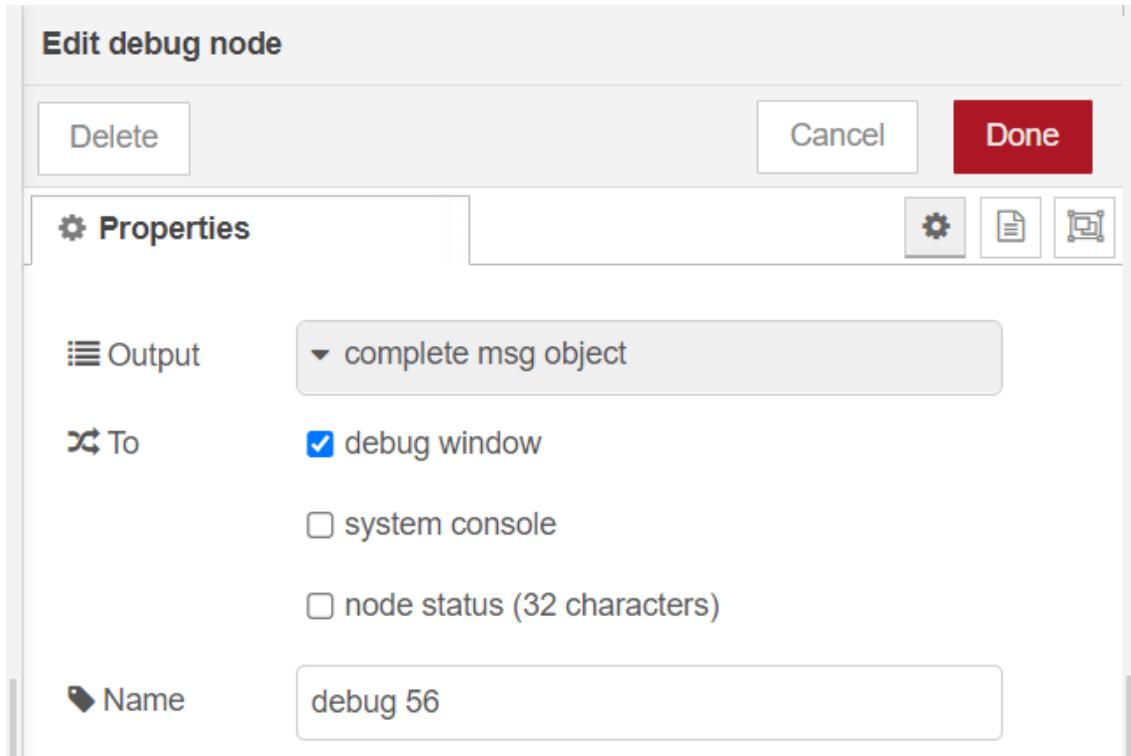
[Delete](#) [Cancel](#) [Done](#)

**Properties**

Output msg. payload

To  debug window  
 system console  
 node status (32 characters)

Name debug 55



We have to learn how to use this library without Forms framework.

At least we know how to use in Forms frameworks, but FT Optix does not use Forms framework.

#### 43.2. Thirt party libraries (dll) Socket Server application with Optix

You can see the result here

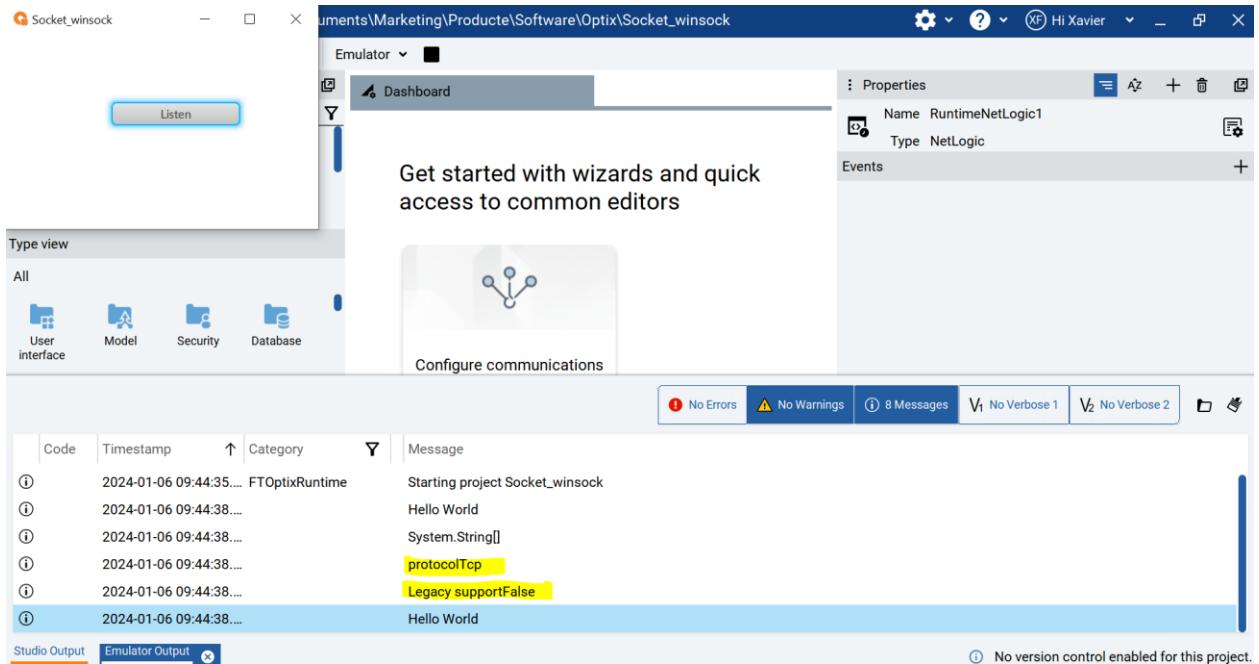
<https://youtu.be/6E4xt02Ohzw>

You can download the code from here

[https://github.com/xavierflorensa/Optix\\_Winsock\\_Orcas\\_Server.git](https://github.com/xavierflorensa/Optix_Winsock_Orcas_Server.git)

We can see some data from class Winsock\_Orcas

As you can see here, we have been able to see which one is the the Protocol used (TCP or UDP)



This is done with this code

At least we can have a dialog with the class to read, write data, when we hit the button.

Socket\_winsoc

```
[ExportMethod]

public void Send()
{
    Log.Info("Hello World");
    winsock_Ear.Listen(2000); //This is to make the PC act as host
                           //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client

    Log.Info(winsock_Ear.LocalIP.ToString());
    Log.Info("protocol"+winsock_Ear.Protocol.ToString());
    Log.Info("Legacy support"+winsock_Ear.LegacySupport.ToString());
    Log.Info("Hello World");
    string text_to_send = "Hello World";
    winsock_Ear.Send(text_to_send);
}
```

This is the complete code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
```

```

using FTOptix.HMIPrj;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using Winsock_Orcas;
using System.Reflection.Emit;
using System.Net.Sockets;
using System.Net.Security;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    Winsock winsock_Ear = new Winsock();

    private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsockConnectedEventArgs e)
    {
        Log.Info("Connected to slave!!");

    }

    [ExportMethod]

    public void Send()
    {
        Log.Info("Hello World");
        winsock_Ear.Listen(2000); //This is to make the PC act as host
                                //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client

        Log.Info(winsock_Ear.LocalIP.ToString());
        Log.Info("protocol"+winsock_Ear.Protocol.ToString());
        Log.Info("Legacy support"+winsock_Ear.LegacySupport.ToString());
        Log.Info("Hello World");
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);

    }
}

```

```

public void Start(object sender, EventArgs e)
//public override void Start(object sender, EventArgs e)
{
    winsock_Ear.Listen(2000); //This is to make the PC act as host
                           //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client

    // Insert code to be executed when the user-defined logic is started
}

public override void Stop()
//public override void Stop()
{
    Log.Info("Stopping");
    // Insert code to be executed when the user-defined logic is stopped
}
private void winsock_Ear_ConnectionRequest(object sender,
WinsockConnectionRequestEventArgs e)
{
    winsock_Ear.Close();
    winsock_Ear.Accept(e.Client);
    Log.Info("Connected to slave!!");

}

//private Winsock_Orcas.Winsoc winsock_Ear;
//public Winsock_Orcas.Winsoc winsock_Ear;
}

```

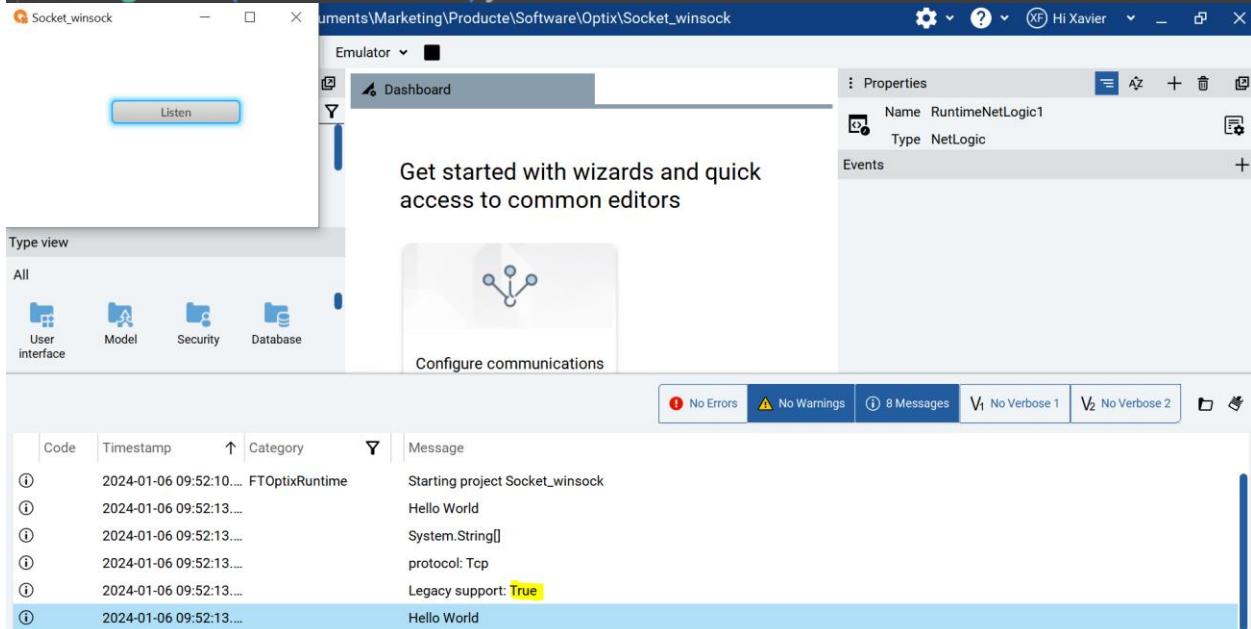
If we change status of legacy support we see

```

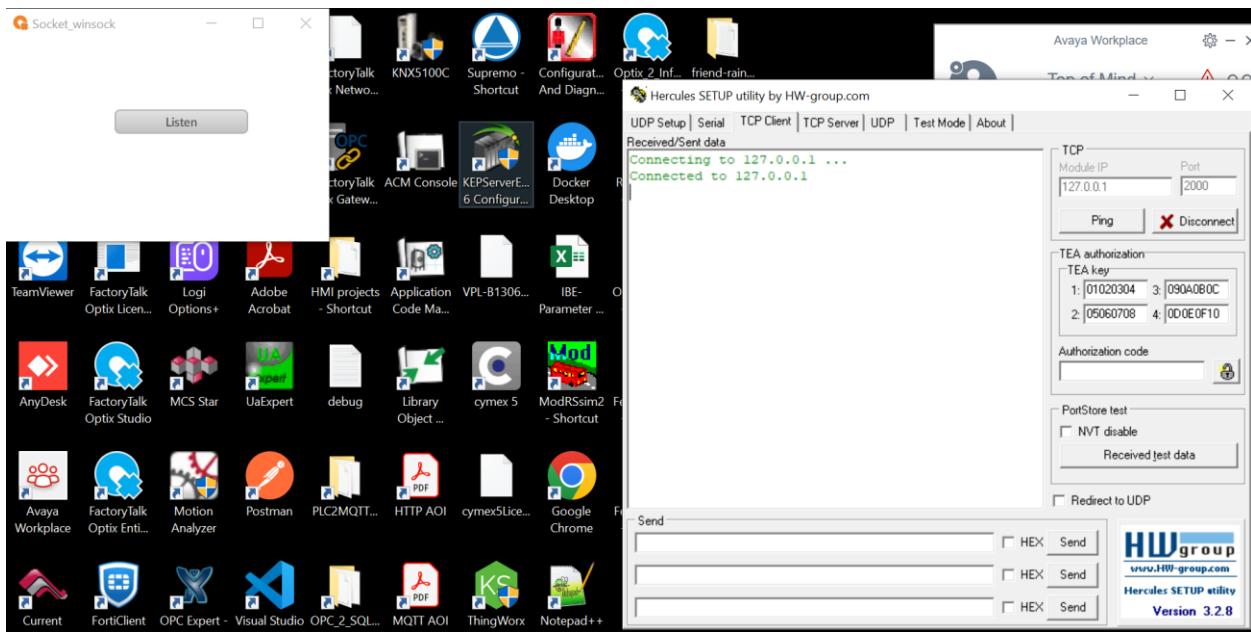
public void Send()
{
    Log.Info("Hello World");
    winsock_Ear.Listen(2000); //This is to make the PC act as host
                           //winsock_Ear.Connect("10.2.10.201", 2000);

    winsock_Ear.LegacySupport=true;
    Log.Info(winsock_Ear.LocalIP.ToString());
    Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
    Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
    Log.Info("Hello World");

```



We must check if the server is listening on port 2000.



We can even retrieve the IP from server host and from remote client

Type	Code	Timestamp	Category	Message
①		2024-01-06 13:18:31....		Hello World
①		2024-01-06 13:18:31....		LocalIP: 192.168.1.163
①		2024-01-06 13:18:31....		protocol: Tcp
①		2024-01-06 13:18:31....		Legacy support: True
①		2024-01-06 13:18:31....		LocalPort: 2000
①		2024-01-06 13:18:31....		Remote Host: localhost
①		2024-01-06 13:18:31....		Remote Port: 8080
①		2024-01-06 13:18:31....		State: Listening
①		2024-01-06 13:18:31....		Hello World

```

[ExportMethod]

public void Send()
{
    Log.Info("Hello World");
    winsock_Ear.Listen(2000); //This is to make the PC act as server host
                            //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client

    winsock_Ear.LegacySupport=true;
    Log.Info("LocalIP: "+winsock_Ear.LocalIP[0]);
    Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
    Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
    Log.Info("LocalPort: "+winsock_Ear.LocalPort.ToString());
    Log.Info("Remote Host: "+winsock_Ear.RemoteHost.ToString());
    Log.Info("Remote Port: "+winsock_Ear.RemotePort.ToString());
    Log.Info("State: "+winsock_Ear.State.ToString());
    Log.Info("Hello World");
    string text_to_send = "Hello World";
    winsock_Ear.Send(text_to_send);

}

```

Let's improve the code

Let's try to put the code on the start program

This works accepting connections but not more else

```

public class RuntimeNetLogic1 : BaseNetLogic
{

    //Winsock winsock_Ear = new Winsock();

    private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsockConnectedEventArgs e)
    {
        Log.Info("Connected to slave!!");

    }

    [ExportMethod]

    public void Send()
    {
        /*

```

```

        Log.Info("Hello World");
        winsock_Ear.Listen(2000);//This is to make the PC act as server host
                                //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client

        winsock_Ear.LegacySupport=true;
        Log.Info("LocalIP: "+winsock_Ear.LocalIP[0]);
        Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
        Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
        Log.Info("LocalPort: "+winsock_Ear.LocalPort.ToString());
        Log.Info("Remote Host: "+winsock_Ear.RemoteHost.ToString());
        Log.Info("Remote Port: "+winsock_Ear.RemotePort.ToString());
        Log.Info("State: "+winsock_Ear.State.ToString());
        Log.Info("Hello World");
        while (true)
        {
            string text_to_send = "Hello World";
            winsock_Ear.Send(text_to_send);
            System.Threading.Thread.Sleep(1000);
        }
        /*
    }
}

```

```

public override void Start()
//public override void Start(object sender, EventArgs e)
{
    Winsock winsock_Ear = new Winsock();
    winsock_Ear.LegacySupport=true;

    winsock_Ear.Listen(2000);//This is to make the PC act as host
                            //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client

    Log.Info("LocalIP: "+winsock_Ear.LocalIP[0]);
    Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
    Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
    Log.Info("LocalPort: "+winsock_Ear.LocalPort.ToString());
    Log.Info("Remote Host: "+winsock_Ear.RemoteHost.ToString());
    Log.Info("Remote Port: "+winsock_Ear.RemotePort.ToString());
    Log.Info("State: "+winsock_Ear.State.ToString());
    Log.Info("Hello World");
    /*
}

```

```

        while (true)
        {
            string text_to_send = "Hello World";
            winsock_Ear.Send(text_to_send);
            System.Threading.Thread.Sleep(1000);
        }
        */
    }

    public override void Stop()
    //public override void Stop()
    {
        Log.Info("Stopping");
        // Insert code to be executed when the user-defined logic is stopped
    }
    private void winsock_Ear_ConnectionRequest(object sender,
WinsockConnectionEventArgs e)
    {
        //winsock_Ear.Close();
        //winsock_Ear.Accept(e.Client);
        Log.Info("Connected to slave!!");

    }

    //private Winsock_Orcas.Winsoc winsock_Ear;
    //public Winsock_Orcas.Winsoc winsock_Ear;
}

```

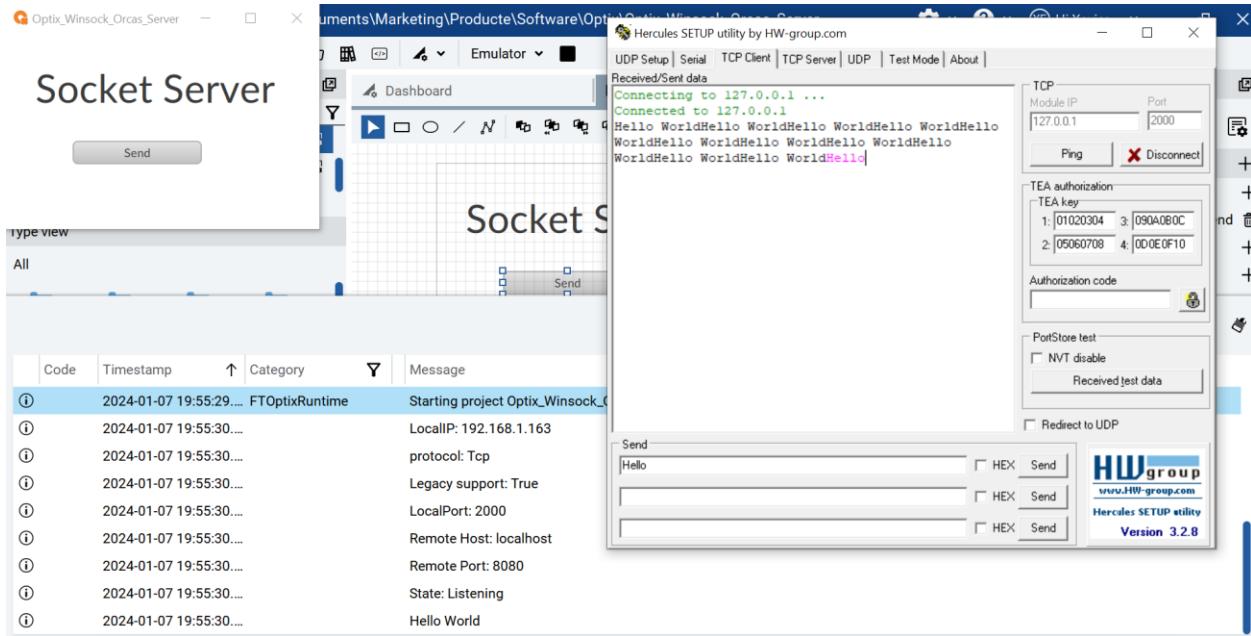
EventArgs e is a parameter called e that contains the event data, see the EventArgs MSDN page for more information.

Object Sender is a parameter called Sender that contains a reference to the control/object that raised the event.

You can download the code from here

[https://github.com/xavierflorenc/Optix\\_Winsock\\_Orcas\\_Server.git](https://github.com/xavierflorenc/Optix_Winsock_Orcas_Server.git)

Let's deal with events,



Now we can send whatever messages we want from server  
 You have to change the application and add this sentence  
 To assign a callback that be executed after first connectionrequest

```
public override void Start()
{
    winsock_Ear.LegacySupport=true;

    winsock_Ear.Listen(2000); //This is to make the PC act as host
                           //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client
    // Assign a callback to be excuted when the client is connected
    //Winsock_Ear.Connected += winsock_Ear_Connected;
    // Assign a callback to be executed when a message is received from the
server
    winsock_Ear.ConnectionRequest += winsock_Ear_ConnectionRequest;
```

This is the complete code

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
```

```

using Winsock_Orcas;
using System.Reflection.Emit;
using System.Net.Sockets;
using System.Net.Security;
using System.Net.Http;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    Winsock winsock_Ear = new Winsock(); //opens a new socket

    [ExportMethod]

    public void Send()
    {
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);
    }

    public override void Start()
    {
        winsock_Ear.LegacySupport=true;

        winsock_Ear.Listen(2000); //This is to make the PC act as host
                               //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client
        // Assign a callback to be excuted when the client is connected
        //Winsock_Ear.Connected += winsock_Ear_Connected;
        // Assign a callback to be executed when a message is received from the
server
        winsock_Ear.ConnectionRequest += winsock_Ear_ConnectionRequest;
        Log.Info("LocalIP: "+winsock_Ear.LocalIP[0]);
        Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
        Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
        Log.Info("LocalPort: "+winsock_Ear.LocalPort.ToString());
        Log.Info("Remote Host: "+winsock_Ear.RemoteHost.ToString());
        Log.Info("Remote Port: "+winsock_Ear.RemotePort.ToString());
        Log.Info("State: "+winsock_Ear.State.ToString());
        Log.Info("Hello World");
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);
    }

    public override void Stop()
    {

```

```

        Log.Info("Stopping");

    }

    private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsocConnectedEventArgs e)
{
    Log.Info("Connected to slave!!");

}

private void winsock_Ear_ConnectionRequest(object sender,
Winsock_Orcas.WinsocConnectionRequestEventArgs e)
{
    winsock_Ear.Close();
    winsock_Ear.Accept(e.Client);
}

private void winsock_Ear_DataArrival(object sender,
Winsock_Orcas.WinsocDataArrivalEventArgs e)
{
    string abRecibidos = winsock_Ear.Get<string>();
    Log.Info(abRecibidos);

}

private Winsock Winsock_Ear;
//private Winsock_Orcas.Winsoc winsock_Ear;
//public Winsock_Orcas.Winsoc winsock_Ear;
}

```

Let's try the client application with those callbacks

### 43.3. Thirt party libraries (dll) Socket Client application with Optix

You can see the result here

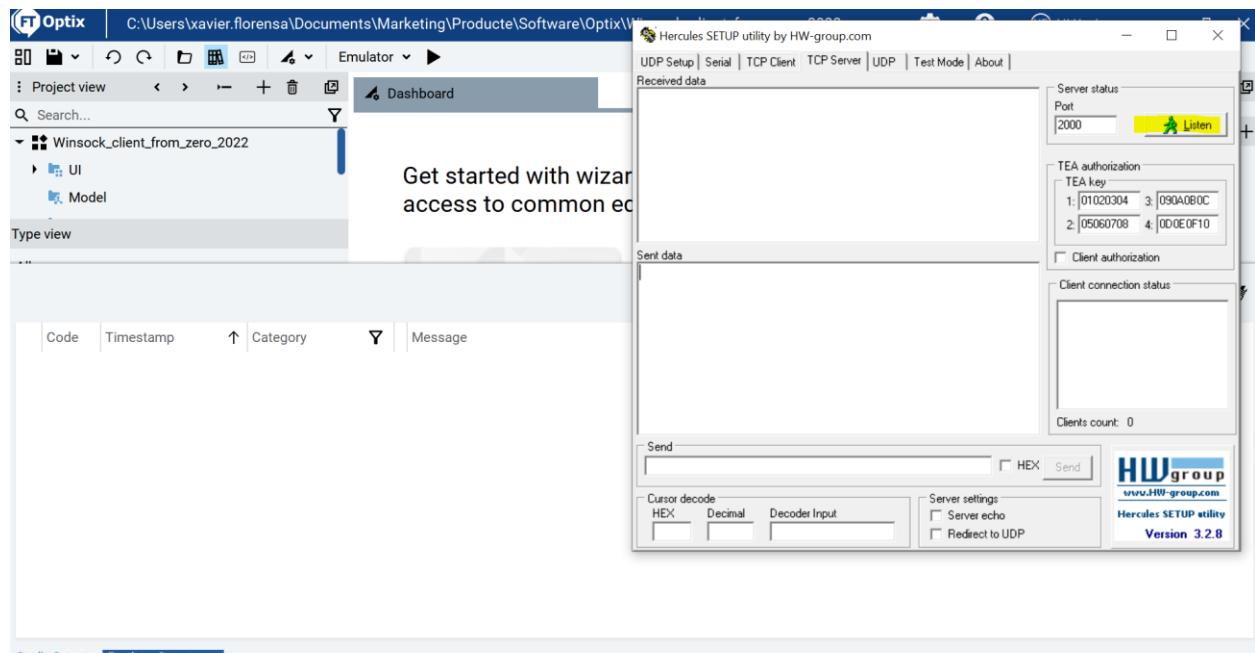
<https://youtu.be/6E4xt02Ohzw>

You can download the project from here

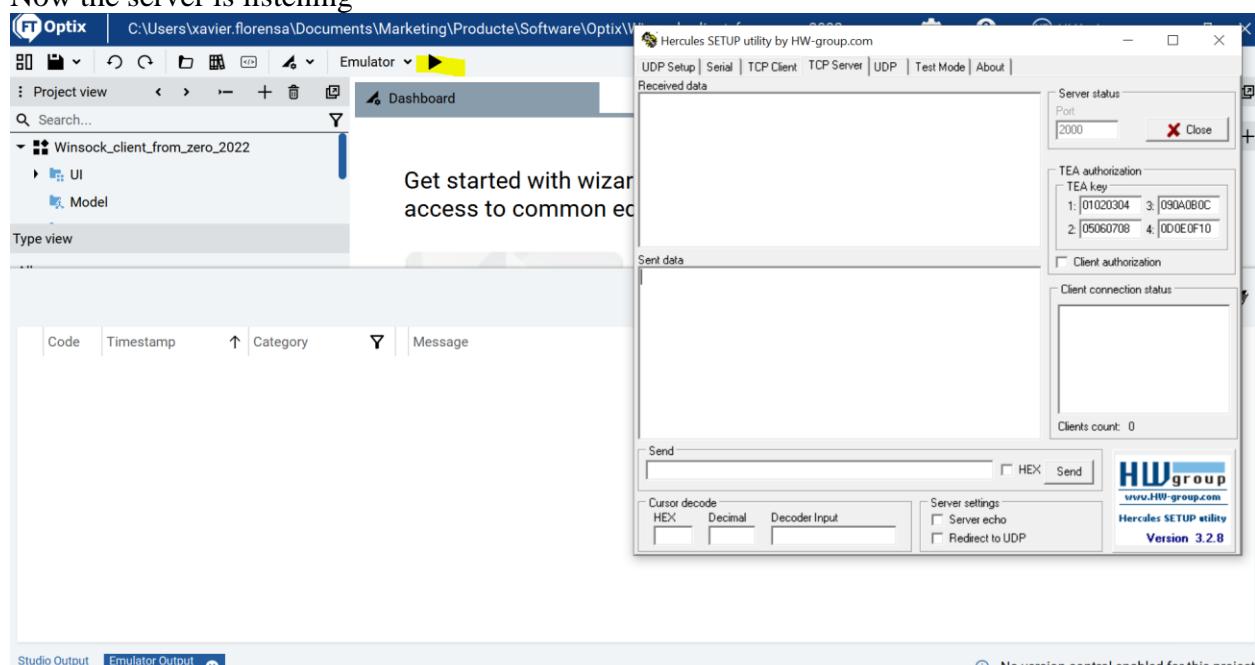
[https://github.com/xavierflorena/Optix\\_Winsock\\_Orcas\\_Client.git](https://github.com/xavierflorena/Optix_Winsock_Orcas_Client.git)

Test the program with a TCP server

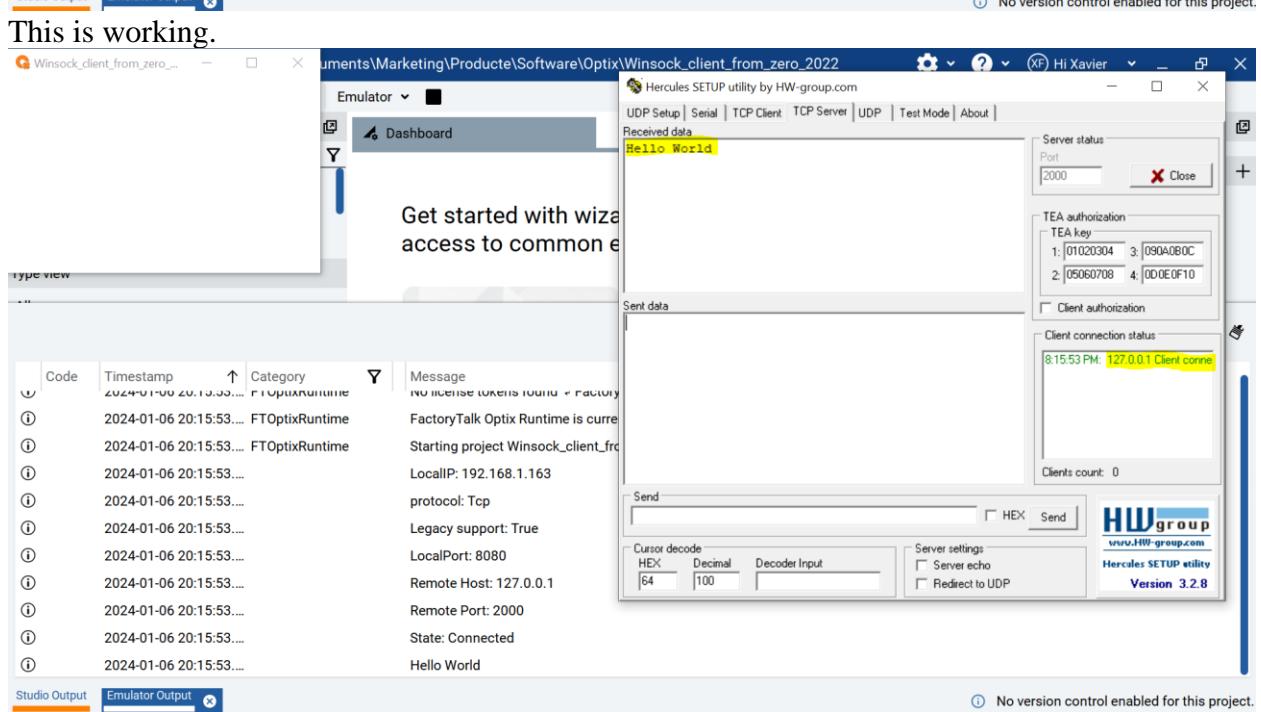
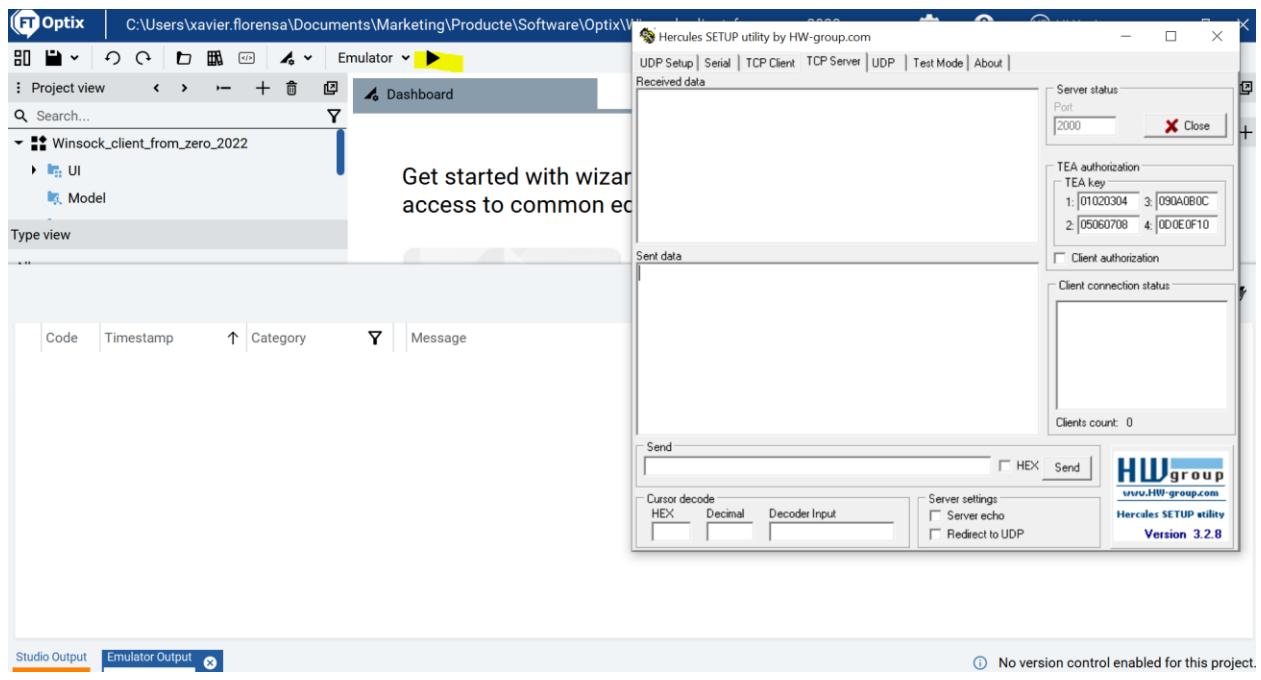
First open the server and put it to listen on port 2000



Now the server is listening

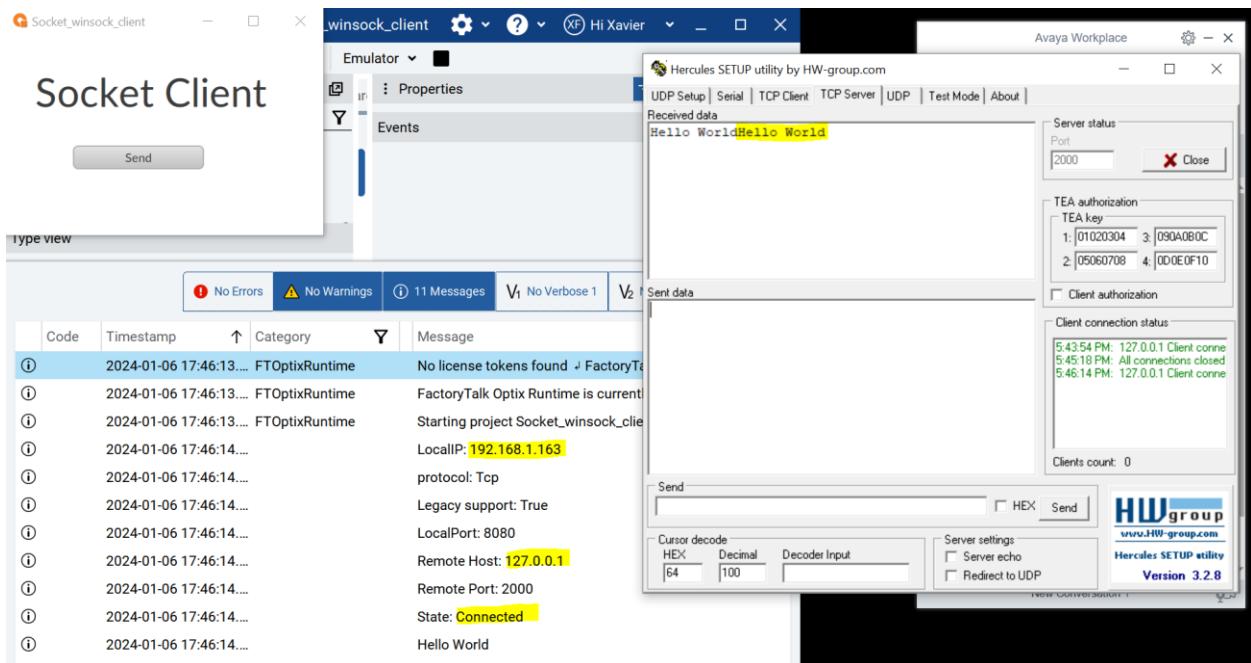


Next start the Optix project

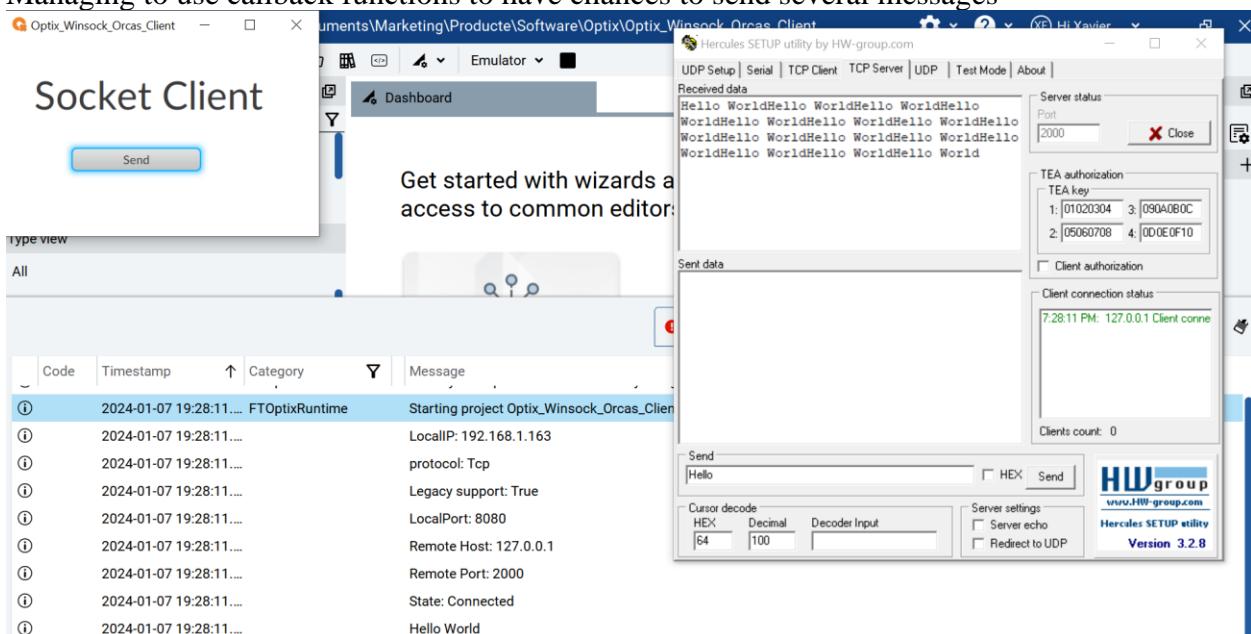


But the connection is made and sending one message is done in one step, and then no chances to send more messages.

Pending to improve this.



Managing to use callback functions to have chances to send several messages



Using this code with the callback function

```
//this is the callback function that will execute after the first connection
request
    private void winsock_Ear_ConnectionRequest(object sender,
Winsock_Orcas.WinsocConnectionRequestEventArgs e)
    {
        winsock_Ear.Close();
        winsock_Ear.Accept(e.Client);
    }
```

This is the complete code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using Winsock_Orcas;
using System.Reflection.Emit;
using System.Net.Sockets;
using System.Net.Security;
using System.Net.Http;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    Winsock winsock_Ear = new Winsock(); //opens a new socket

    [ExportMethod]

    public void Send()
    {
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);
    }

    public override void Start()
    {
        winsock_Ear.LegacySupport=true;
        //winsock_Ear.Listen(2000); //This is to make the PC act as host
        winsock_Ear.Connect("127.0.0.1", 2000); //This is to make the PC act as
client
        // Assign a callback to be excuted when the client is connected
        //Winsock_Ear.Connected += winsock_Ear_Connected;
        // Assign a callback to be executed when a message is received from the
server
        winsock_Ear.ConnectionRequest += winsock_Ear_ConnectionRequest;
        Log.Info("LocalIP: "+winsock_Ear.LocalIP[0]);
        Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
        Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
    }
}
```

```

        Log.Info("LocalPort: "+winsock_Ear.LocalPort.ToString());
        Log.Info("Remote Host: "+winsock_Ear.RemoteHost.ToString());
        Log.Info("Remote Port: "+winsock_Ear.RemotePort.ToString());
        Log.Info("State: "+winsock_Ear.State.ToString());
        Log.Info("Hello World");
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);
    }

    public override void Stop()
    //public override void Stop()
    {
        Log.Info("Stopping");
    }
    //this is the callback function that will execute when a client is connected
    private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsocConnectedEventArgs e)
    {
        Log.Info("||Conectado!!");
    }
    //this is the callback function that will execute after the first connection
request
    private void winsock_Ear_ConnectionRequest(object sender,
Winsock_Orcas.WinsocConnectionRequestEventArgs e)
    {
        winsock_Ear.Close();
        winsock_Ear.Accept(e.Client);
    }
    private Winsock Winsock_Ear;
}

```

[https://github.com/xavierflorenci/Optix\\_Winsock\\_Orcas\\_Client.git](https://github.com/xavierflorenci/Optix_Winsock_Orcas_Client.git)

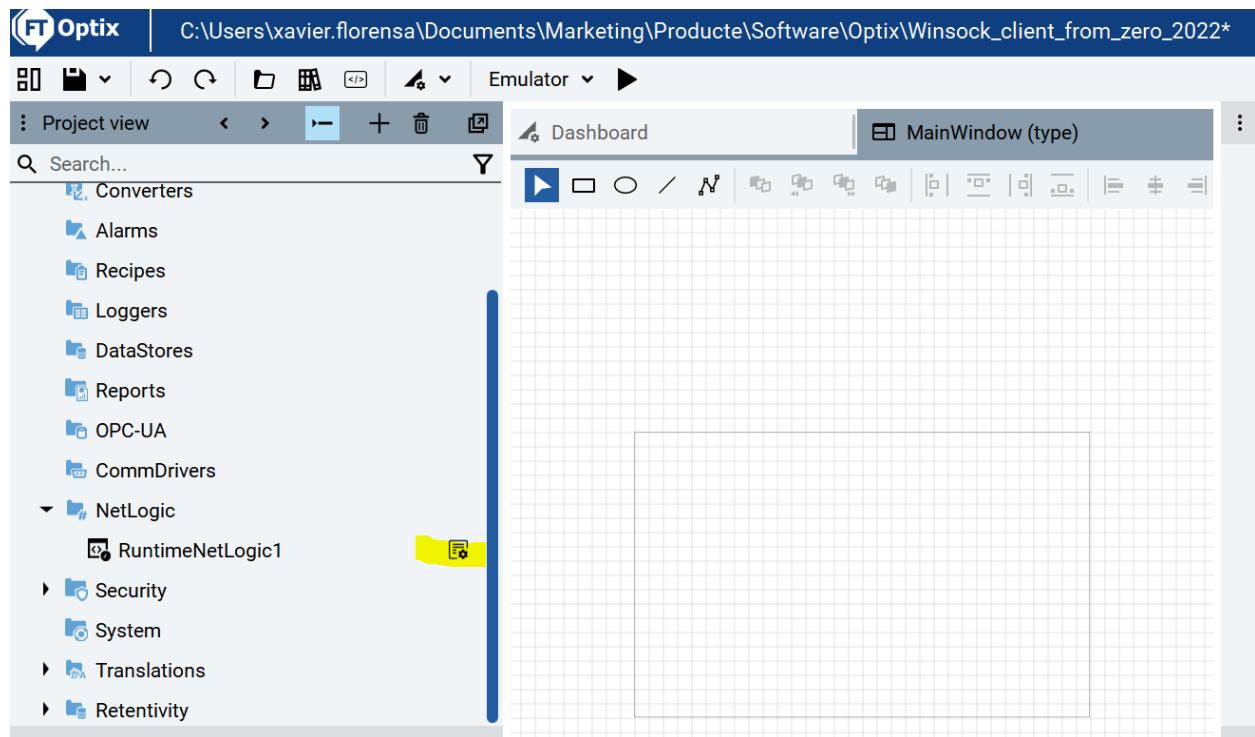
#### 43.3.1. Installing external references (dll libraries) into the project Visual Studio 2022

Let's start this application from zero

The easiest way to install external references to a project is using Visual Studio 2002, since it will do all the work for you. Then if you prefer you can continue with Visual Studio Code since it is much faster to open and close.

Start a new project in FT Optix Studio

Add a new NetLogic code



Save the project

Insert this dll on the root directory of the project

Winsock\_Orcas.dll

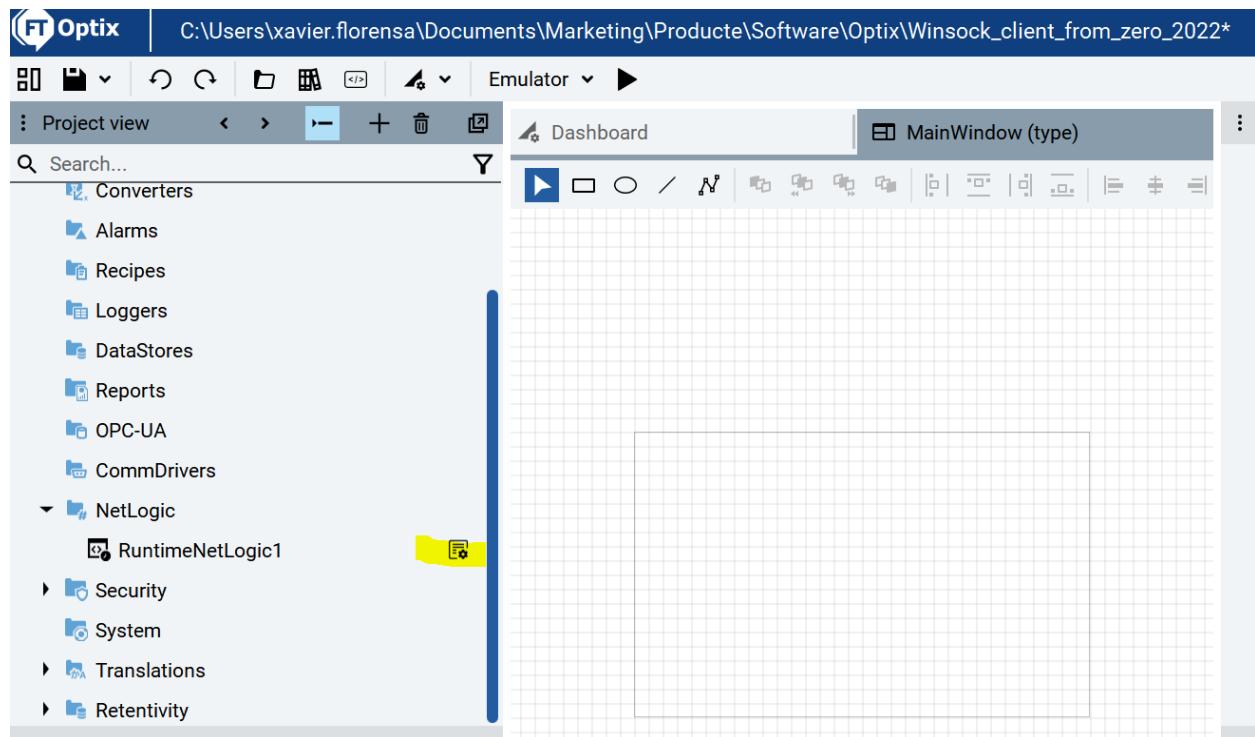
This PC > Documents > Marketing > Produkte > Software > Optix > Winsock\_client\_from\_zero\_2022

Name	Date modified	Type	Size
ApplicationFiles	1/6/2024 7:37 PM	File folder	
Nodes	1/6/2024 7:42 PM	File folder	
ProjectFiles	1/6/2024 7:42 PM	File folder	
IDEVersion	12/22/2023 10:10 AM	Text Document	1 KB
Winsock Orcas.dll	6/1/2018 10:22 PM	Application extension	110 KB
Winsock_client_from_zero_2022	1/6/2024 7:42 PM	FactoryTalk Optix Stu...	2 KB
Winsock_client_from_zero_2022.optix.design	1/6/2024 7:42 PM	DESIGN File	1 KB

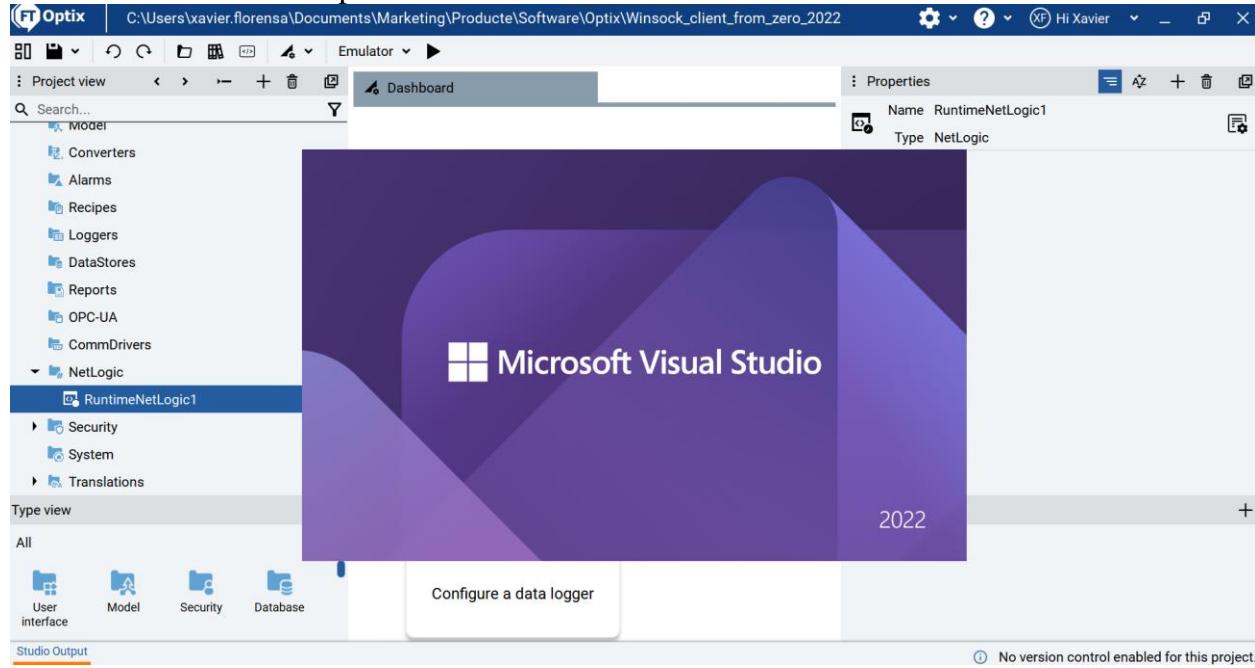
Select editor to Visual Studio 2022

Close and open FT Optix to have this change updated.

Click on the code

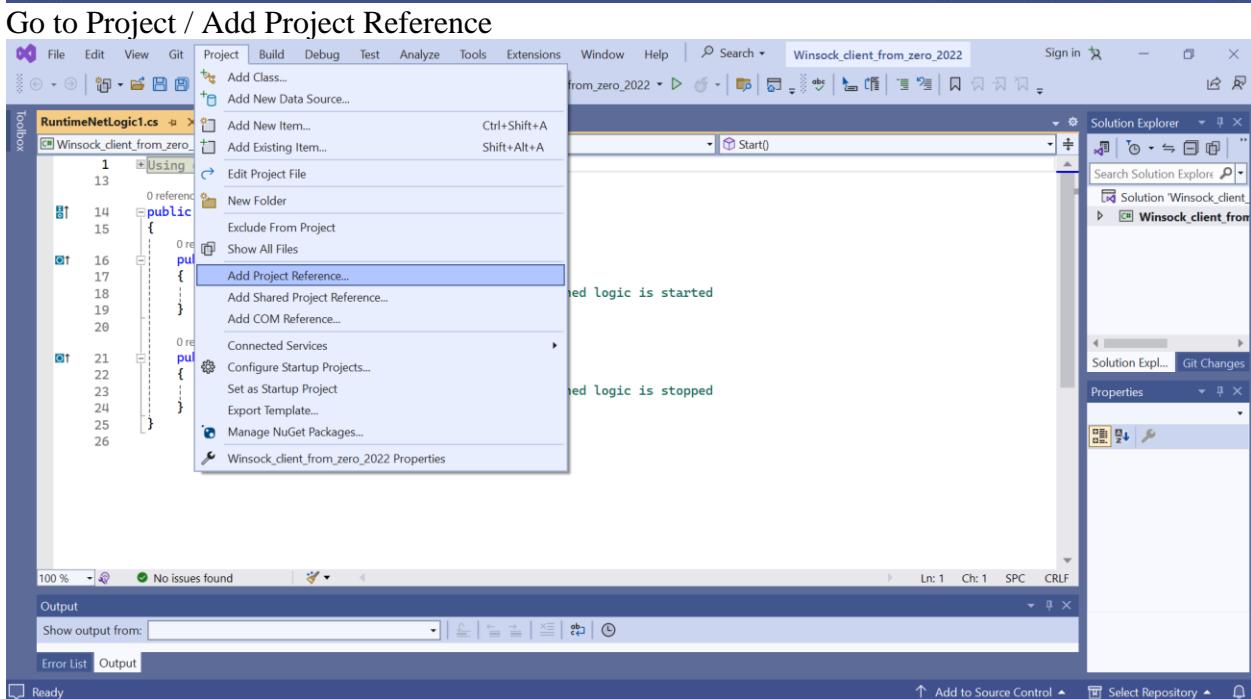


Visual Studio 2022 will open

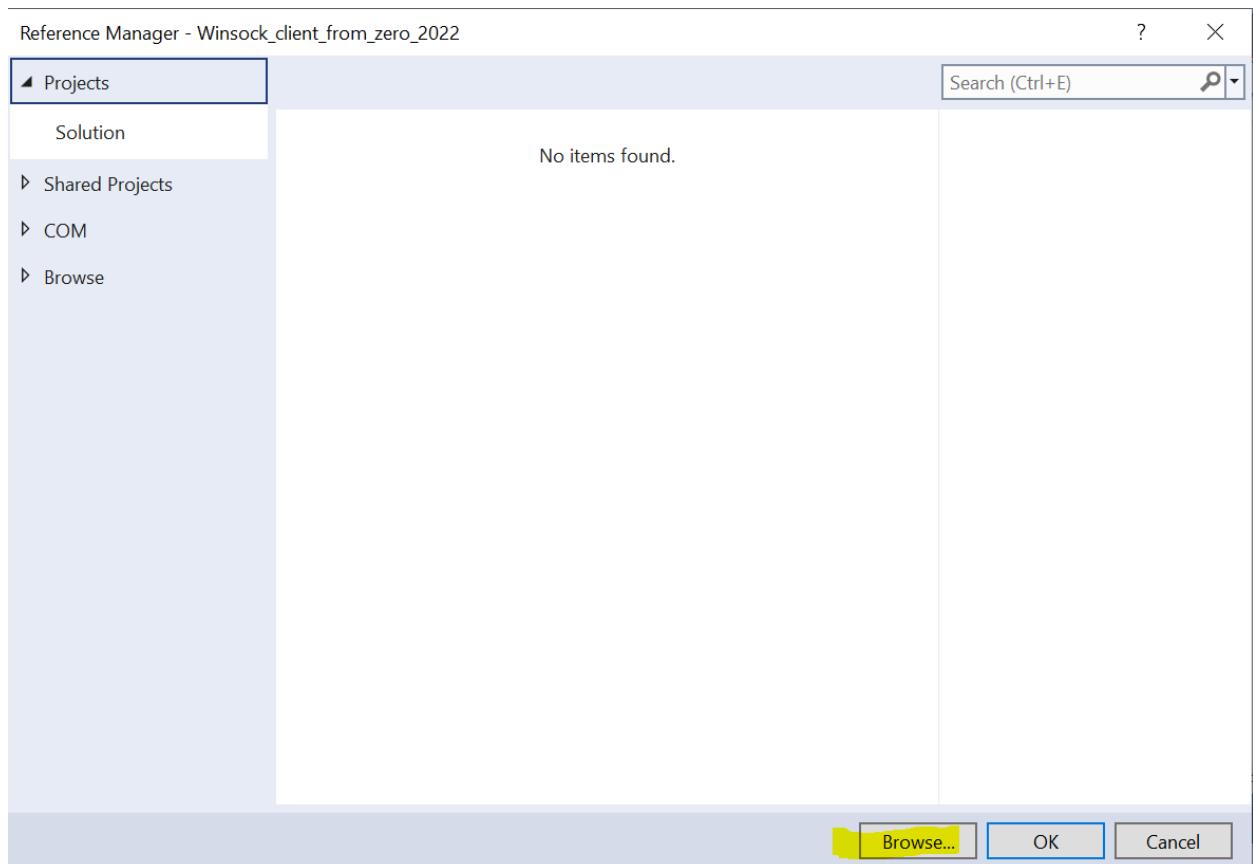


The screenshot shows the Visual Studio IDE interface. The main window displays the code for `RuntimeNetLogic1.cs`. The code defines a class `RuntimeNetLogic1` that inherits from `BaseNetLogic`. It contains two overridden methods: `Start()` and `Stop()`, each with a comment indicating its purpose. The Solution Explorer on the right shows a solution named "Winsock\_client\_from\_zero\_2022" containing a project named "Winsock\_client\_from". The Properties and Tools windows are also visible.

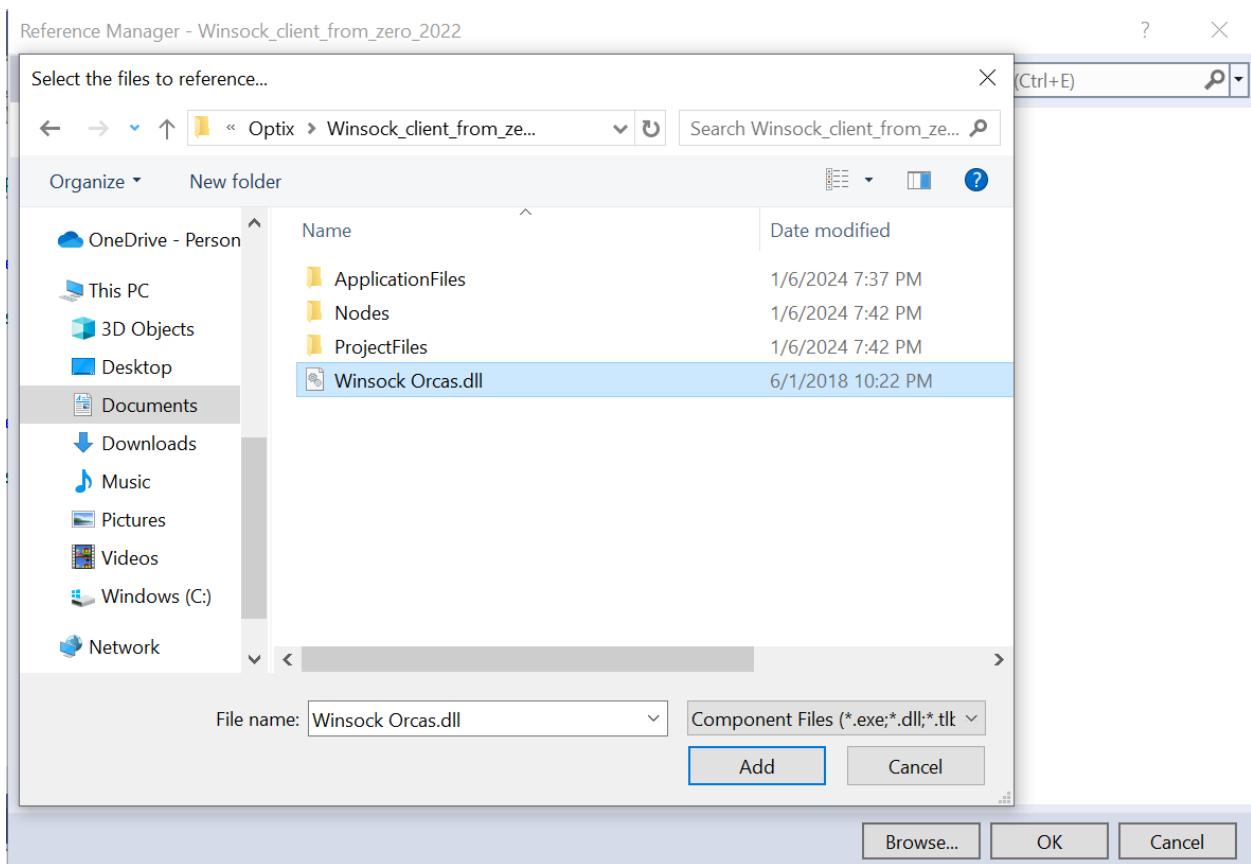
```
1  Using directives
13
14  public class RuntimeNetLogic1 : BaseNetLogic
15  {
16      public override void Start()
17      {
18          // Insert code to be executed when the user-defined logic is started
19      }
20
21      public override void Stop()
22      {
23          // Insert code to be executed when the user-defined logic is stopped
24      }
25  }
```



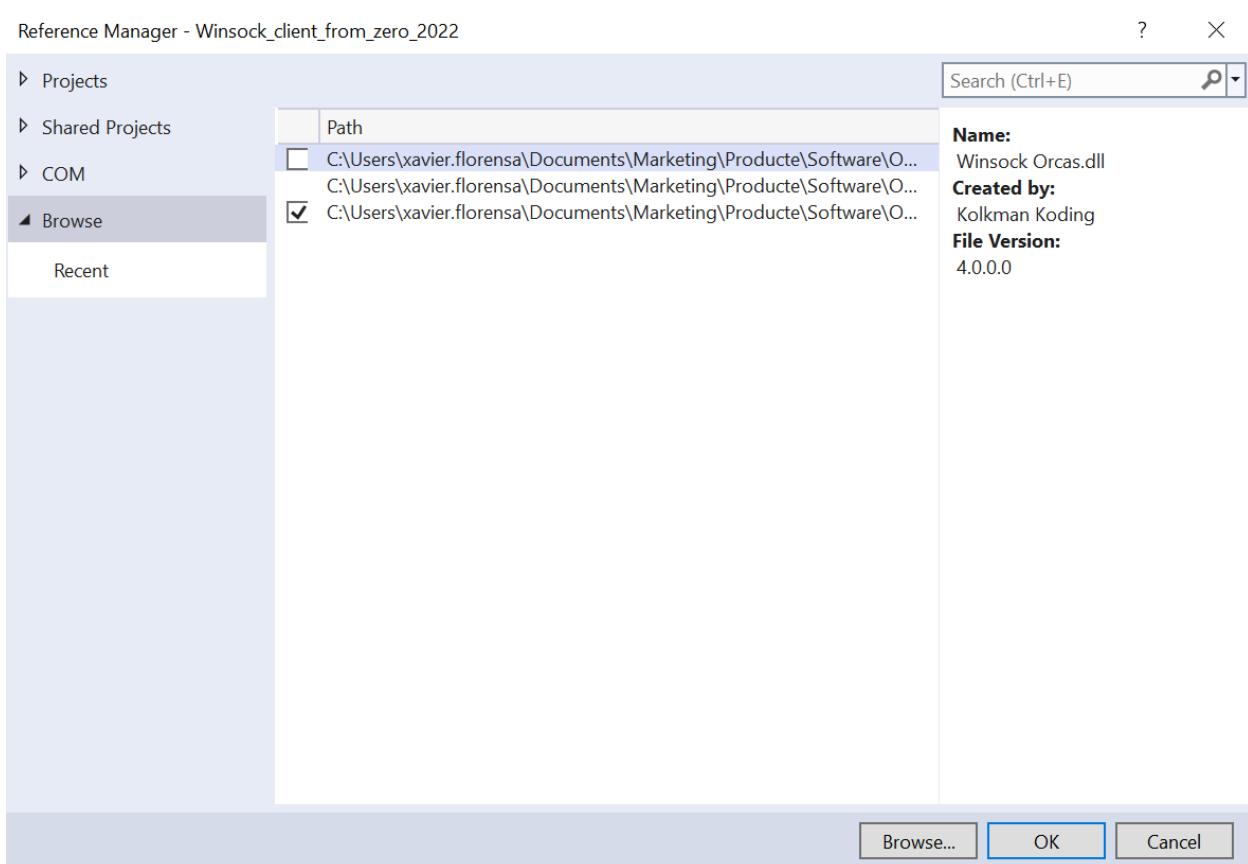
Click on Browse



Select the dll

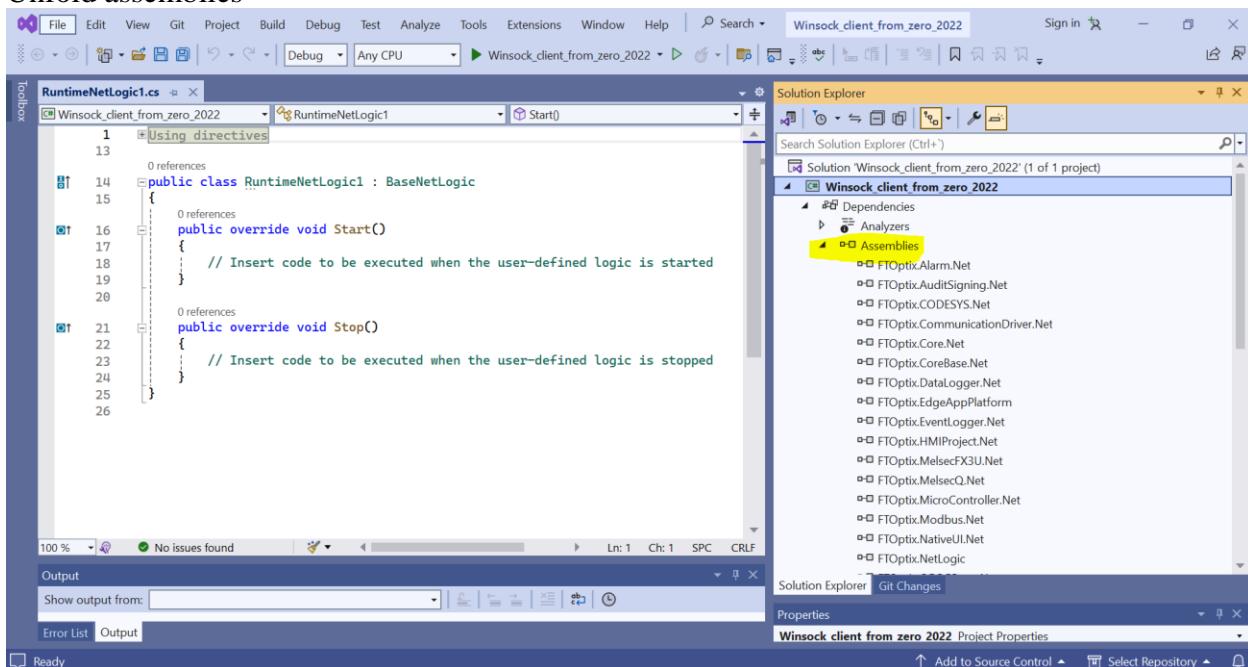


Verify that this is the right path

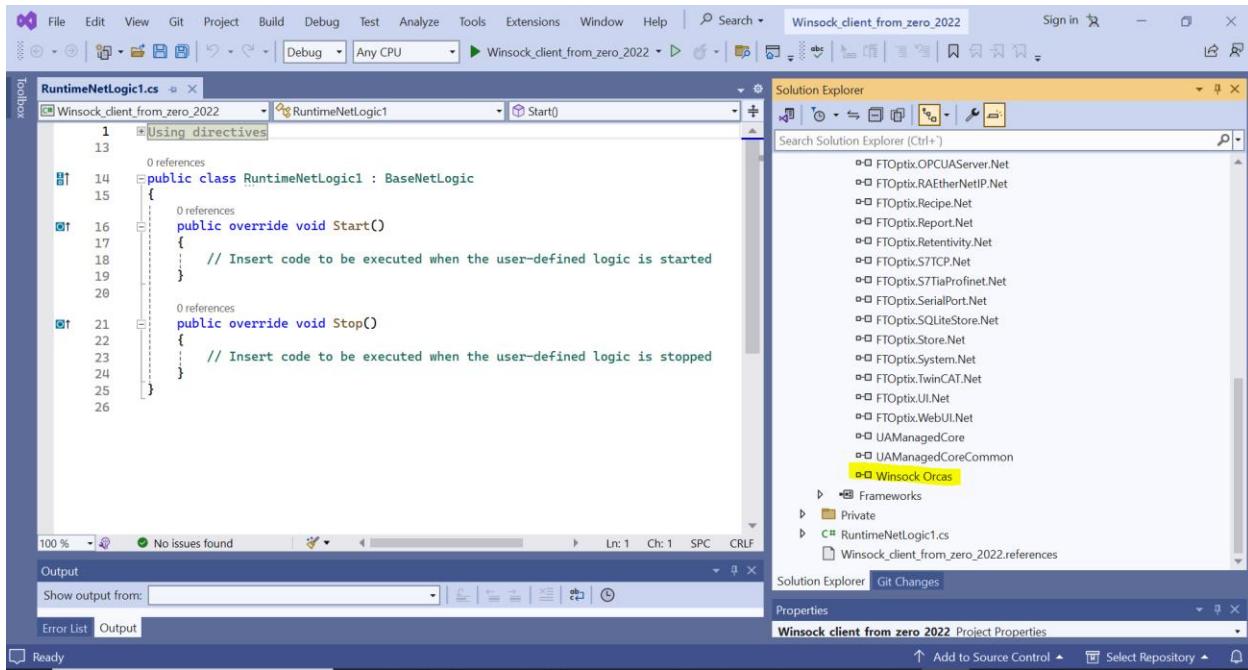


Click Ok

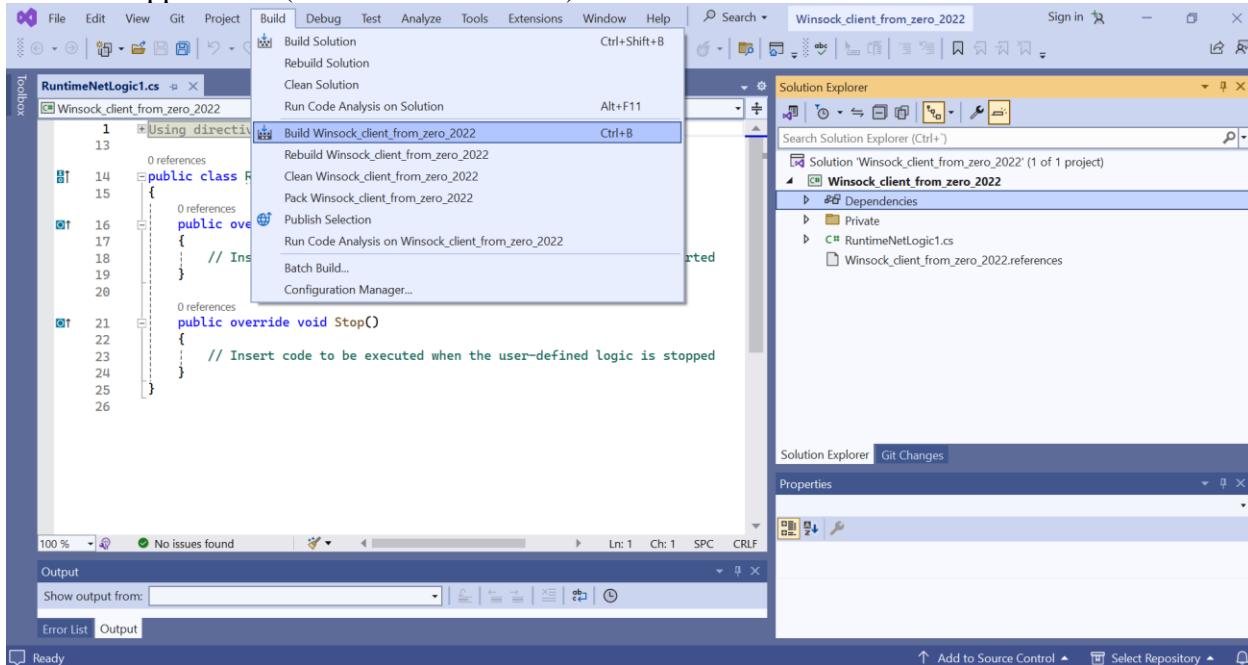
Unfold assemblies



You have it installed



## Build the application (Do not build solution)



You can close Visual Studio 2022

Change the code editor to Visual Studio Code in FT Optix Studio

Save close and open it again

Open Netlogic

The screenshot shows the Visual Studio 2022 interface. In the Solution Explorer, the project 'WINSOCK\_CLIENT\_FROM\_ZERO\_2022' is selected, showing files like .vs, .vscode, bin, obj, Private, and RuntimeNetLogic1.cs. The code editor displays the content of RuntimeNetLogic1.cs:

```

1  #region Using directives
2  using System;
3  using UAManagedCore;
4  using OpcUa = UAManagedCore.OpcUa;
5  using FTOptix.HMIProject;
6  using FTOptix.Rentativity;
7  using FTOptix.UI;
8  using FTOptix.NativeUI;
9  using FTOptix.CoreBase;
10 using FTOptix.Core;
11 using FTOptix.NetLogic;
12 #endregion
13
14 public class RuntimeNetLogic1 : BaseNetLogic
15 {
16     public override void Start()
17     {
18         // Insert code to be executed when the user-defined logic is started
19     }
20
21     public override void Stop()
22     {
23         // Insert code to be executed when the user-defined logic is stopped
24     }
25 }

```

At the bottom, it says 'Projects: 1'.

Open references file \*.csproj

VS 2022 has done the dll installation for you without editing this \*.csproj file

The screenshot shows the Visual Studio 2022 interface with the csproj file open. The code editor displays the XML content of Winsock\_client\_from\_zero\_2022.csproj:

```

1 <Project Sdk="Microsoft.NET.Sdk">
2   <PropertyGroup>
3     <TargetFramework>net6.0</TargetFramework>
4     <AppendTargetFrameworkToOutputPath>false</AppendTargetFrameworkToOutputPath>
5     <CopyLocalLockFileAssemblies>true</CopyLocalLockFileAssemblies>
6   </PropertyGroup>
7   <PropertyGroup Condition="$(Configuration)|$(Platform) == 'Debug|AnyCPU'">
8     <OutputPath>bin\</OutputPath>
9     <NoWarn>1701;1702</NoWarn>
10  </PropertyGroup>
11  <PropertyGroup Condition="$(Configuration)|$(Platform) == 'Release|AnyCPU'">
12    <OutputPath>bin\</OutputPath>
13    <NoWarn>1701;1702</NoWarn>
14  </PropertyGroup>
15  <PropertyGroup>
16    <ResolveAssemblyWarnOrErrorOnTargetArchitectureMismatch>None</ResolveAssemblyWarnOrErrorOnTargetArchitectureMismatch>
17  </PropertyGroup>
18  <ItemGroup>
19    <Reference Include="Winsock Orcas">
20      <HintPath>..\..\Winsock Orcas.dll</HintPath>
21    </Reference>
22  </ItemGroup>
23  <Import Project="Winsock_client_from_zero_2022.references"/>
24</Project>

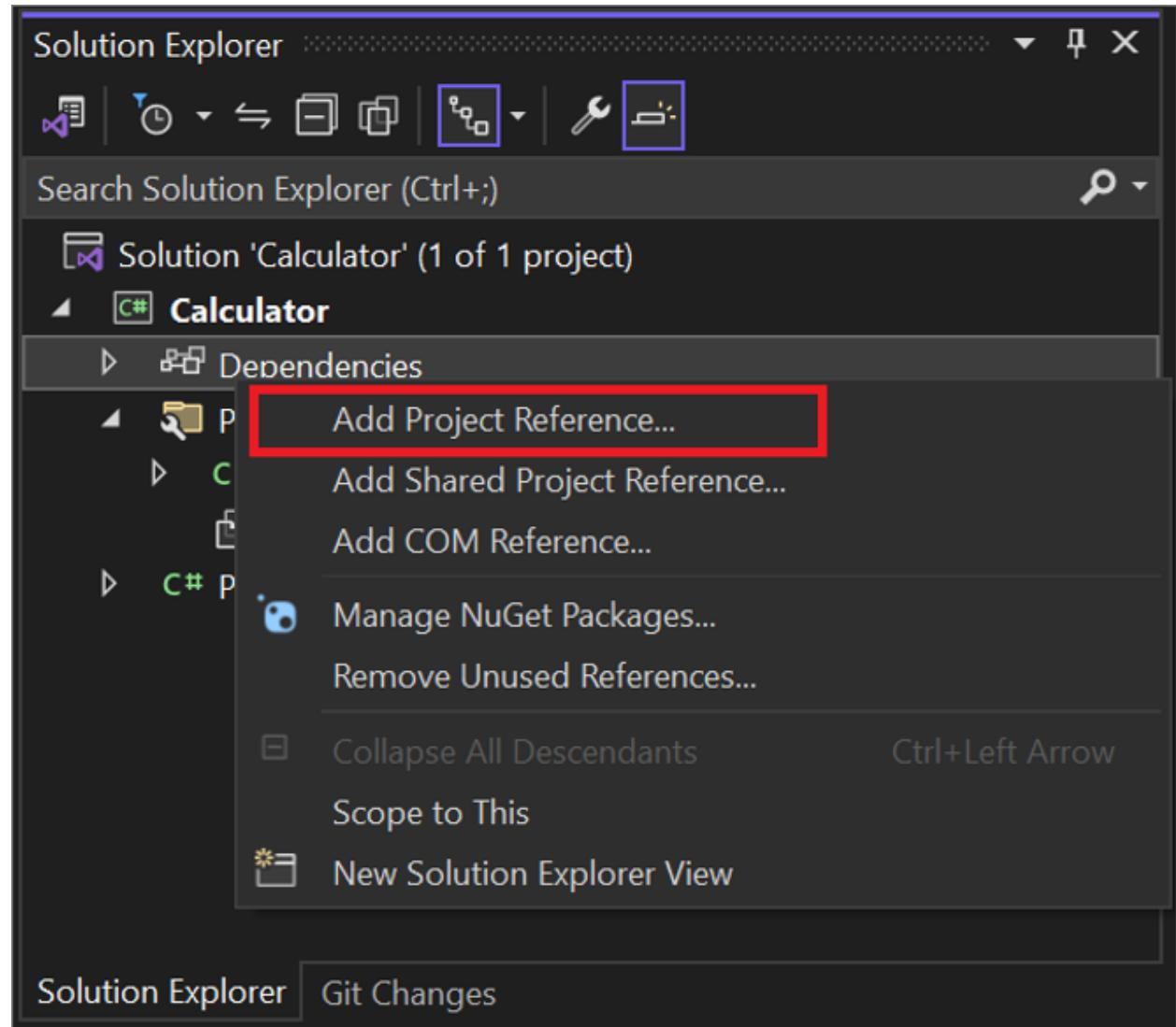
```

A red oval highlights the reference section from line 19 to 22. At the bottom, it says 'Projects: 1'.

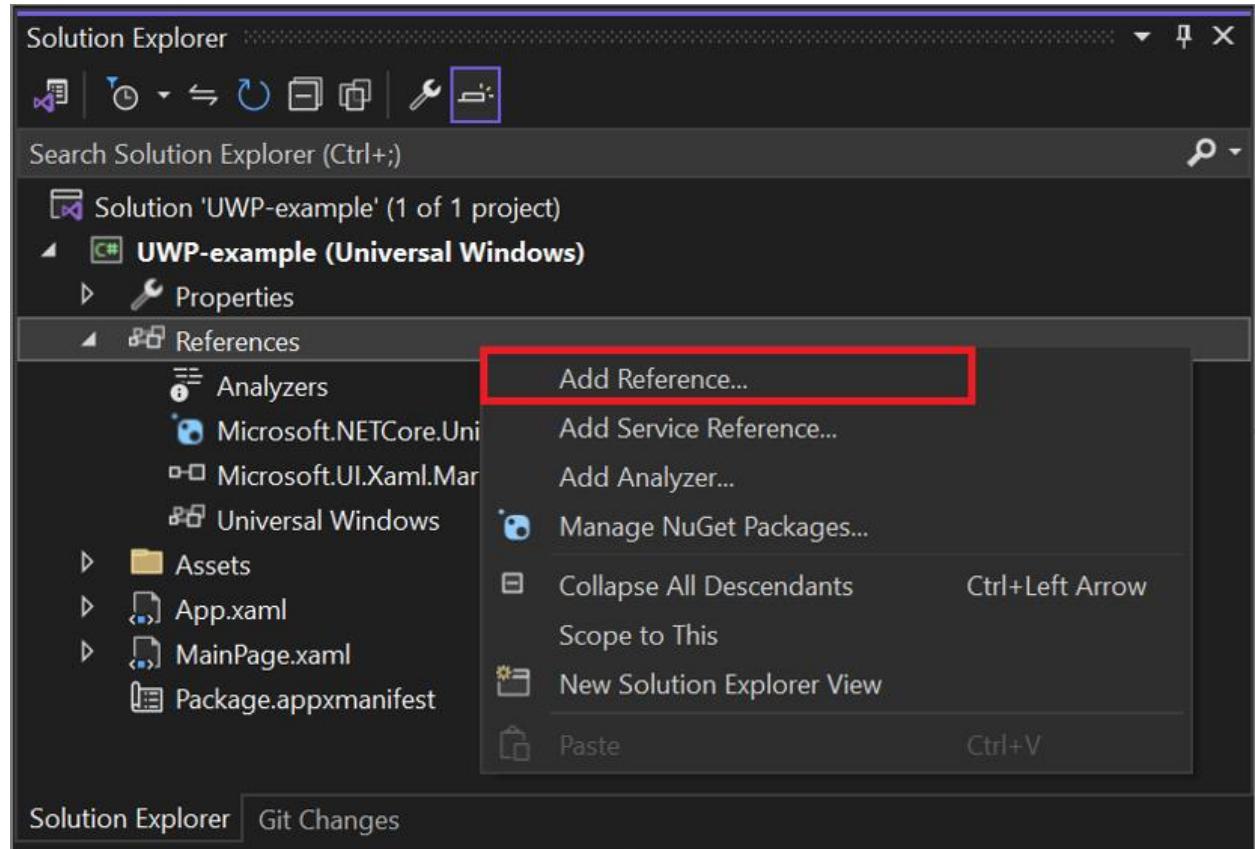
43.3.2. Installing external references (dll libraries) into the project with Visual Studio code  
Pending to test this way

How you add a reference depends on the project type for the code you're working on:

- If you see a **Dependencies** node in [Solution Explorer](#), you can use the right-click context menu to select **Add Project Reference**.



- If you see a **References** node in [Solution Explorer](#), you can use the right-click context menu to select **Add Reference**.



For more information, see [How to: Add or remove references](#).

#### 44. Using NuGet libraries (HiveMQ .Net example)

Let's try to use this fancy library,

<https://github.com/hivemq/hivemq-mqtt-client-dotnet>

You can see the result on this video

<https://youtu.be/3g81mO2wqXQ>

← → ⌂ ⌂ [github.com/hivemq/hivemq-mqtt-client-dotnet](https://github.com/hivemq/hivemq-mqtt-client-dotnet)

Rockwell Node-RED : node-red TTN Login TTN Login | Microsoft 365 Microsoft 365 Office Ubuntu HiveMQ Cloud - Fre... MWC Barcelona

[README](#) [Code of conduct](#) [Apache-2.0 license](#) [Security](#)

 **HIVEMQ**   
MQTT CLIENT

The Spectacular (BETA) C# MQTT Client for .NET

RELEASE **V0.8.0** BUILD **PASSING** DOWNLOADS **12K** LICENSE APACHE 2.0

This .NET MQTT client was put together with love from the HiveMQ team but is still in DEVELOPMENT. As such some things may not work completely until it matures and although unlikely, APIs may change slightly before version 1.0.

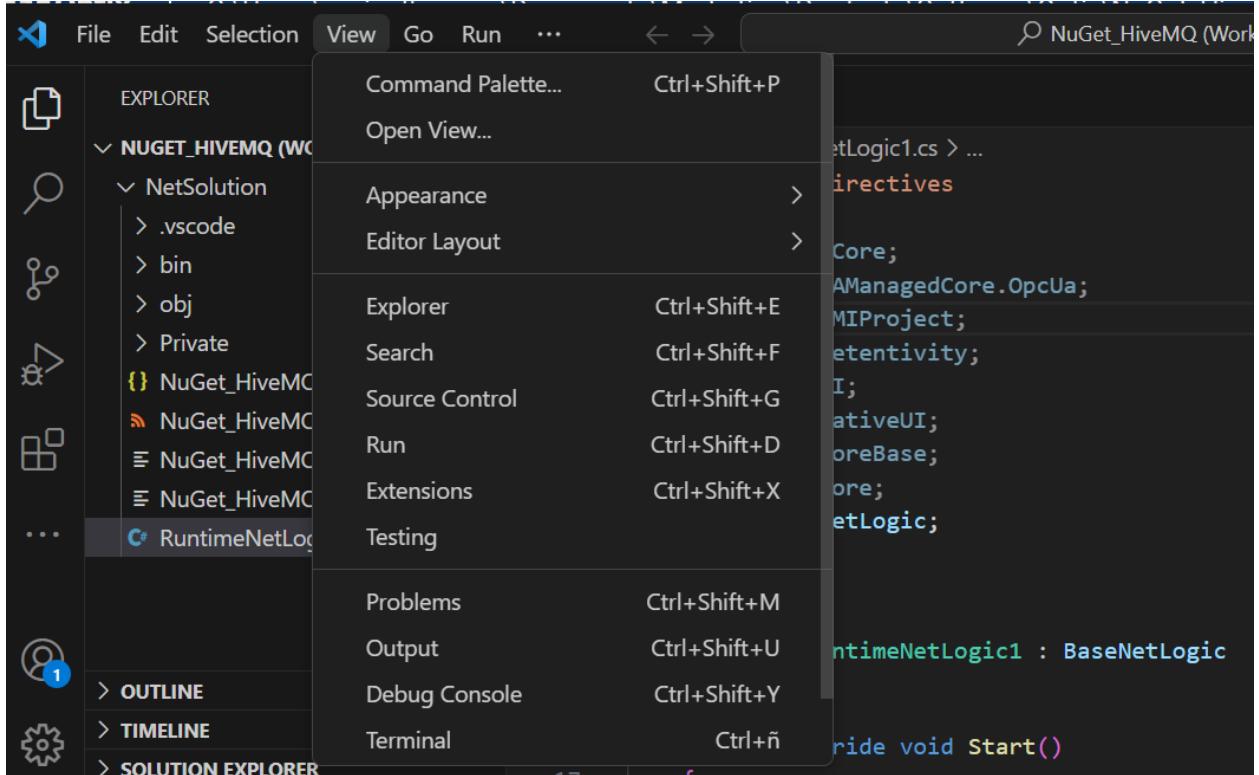
We'd appreciate any feedback you have. Happy MQTT adventures!

- Easy-to-Install: Available as a NuGet package.
- Opensource: No blackbox code. Only trusted, tested and reviewed opensource code.
- Works with Any MQTT 5.0 Broker: Built by HiveMQ but not only for HiveMQ.
- Easy to Use: Smart defaults, excellent interfaces and intelligent automation makes implementing a breeze.

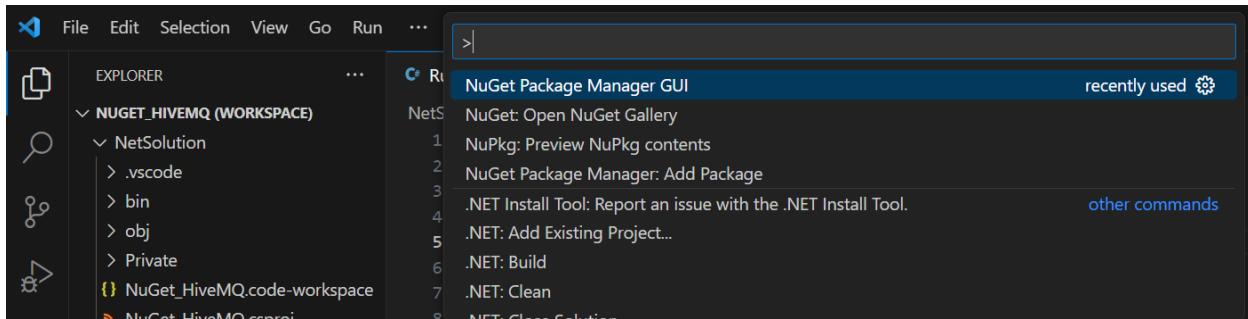
So create a new project and add a new Netlogic Runtime code

Open the code

Click on View/Command Pallete

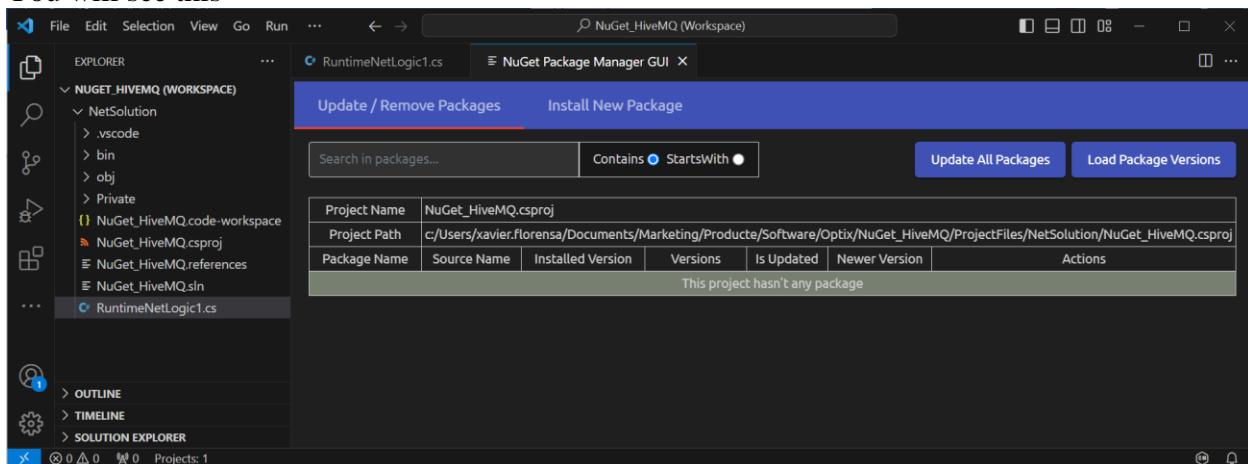


You will see this depending on what VScode extensions you have installed

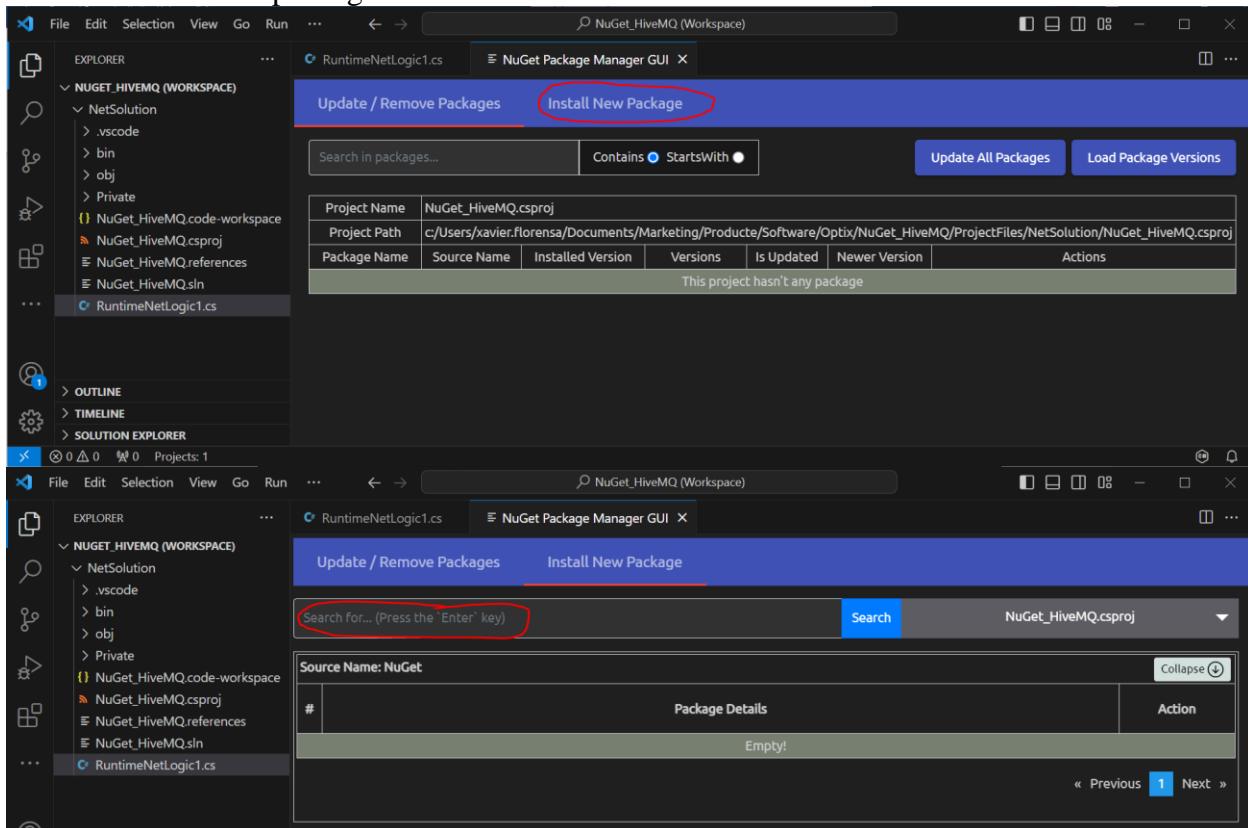


## Select NuGet Package Manager GUI

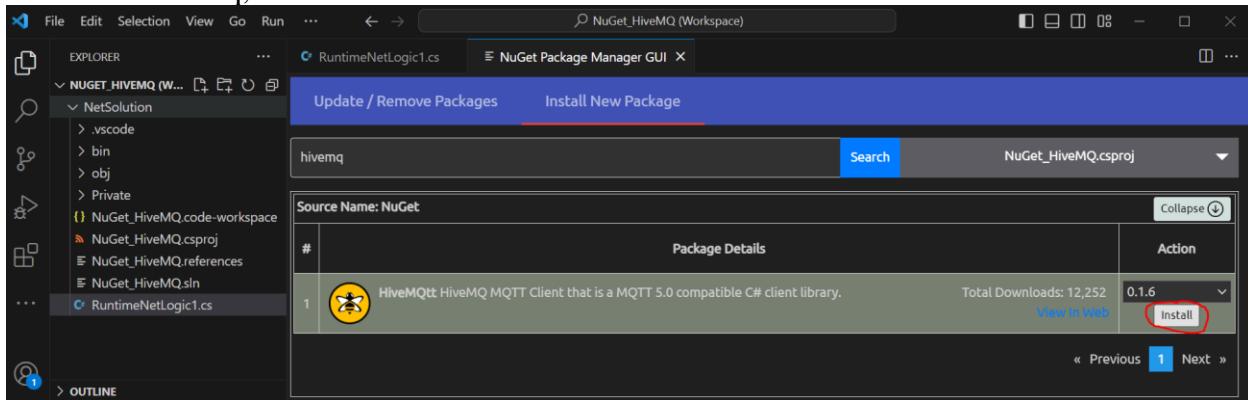
You will see this



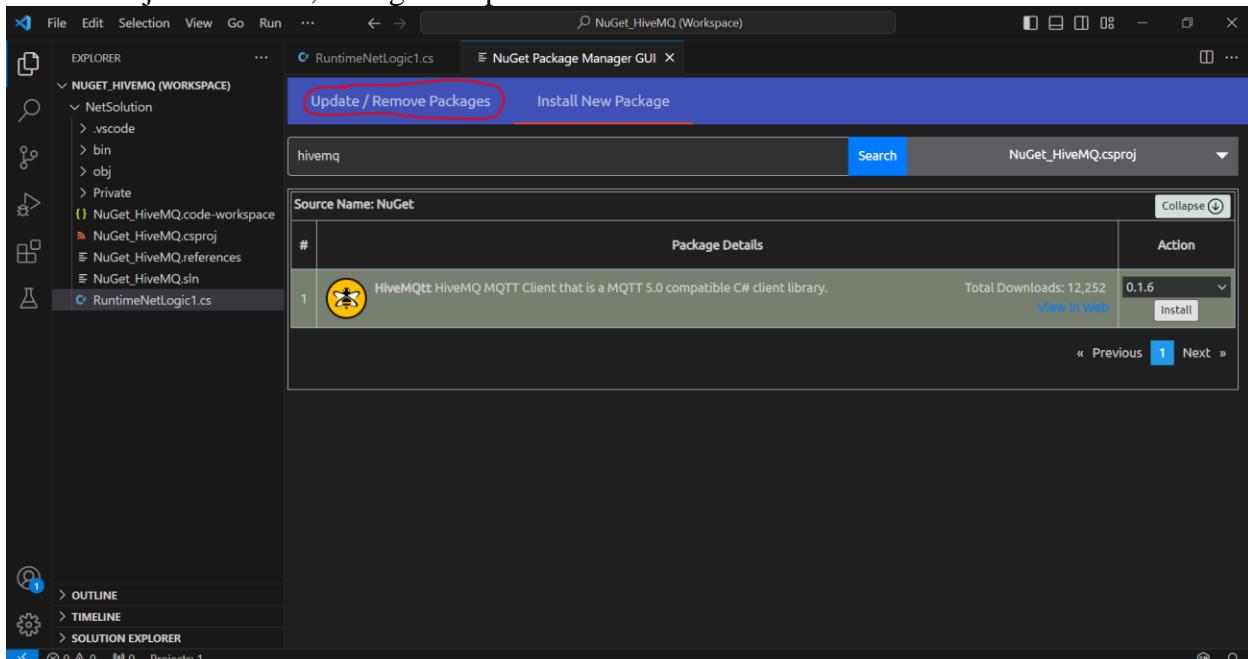
Click on Install new package



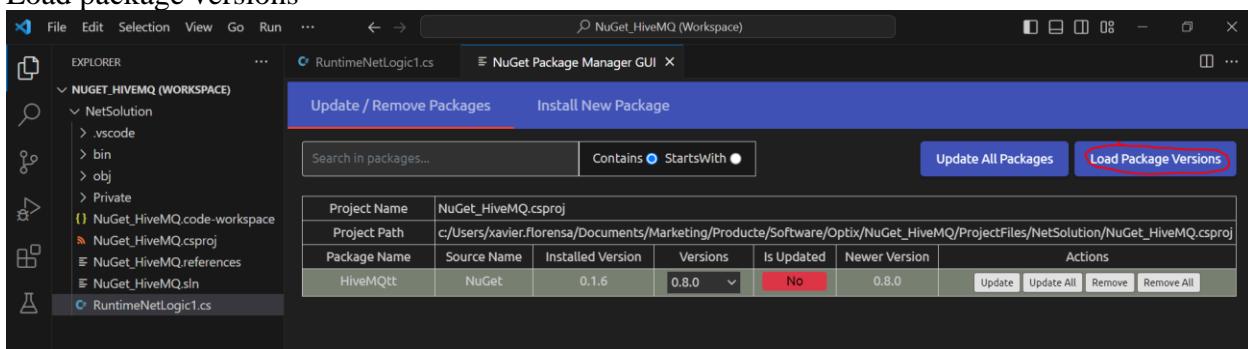
Search for hivemq, and click on install



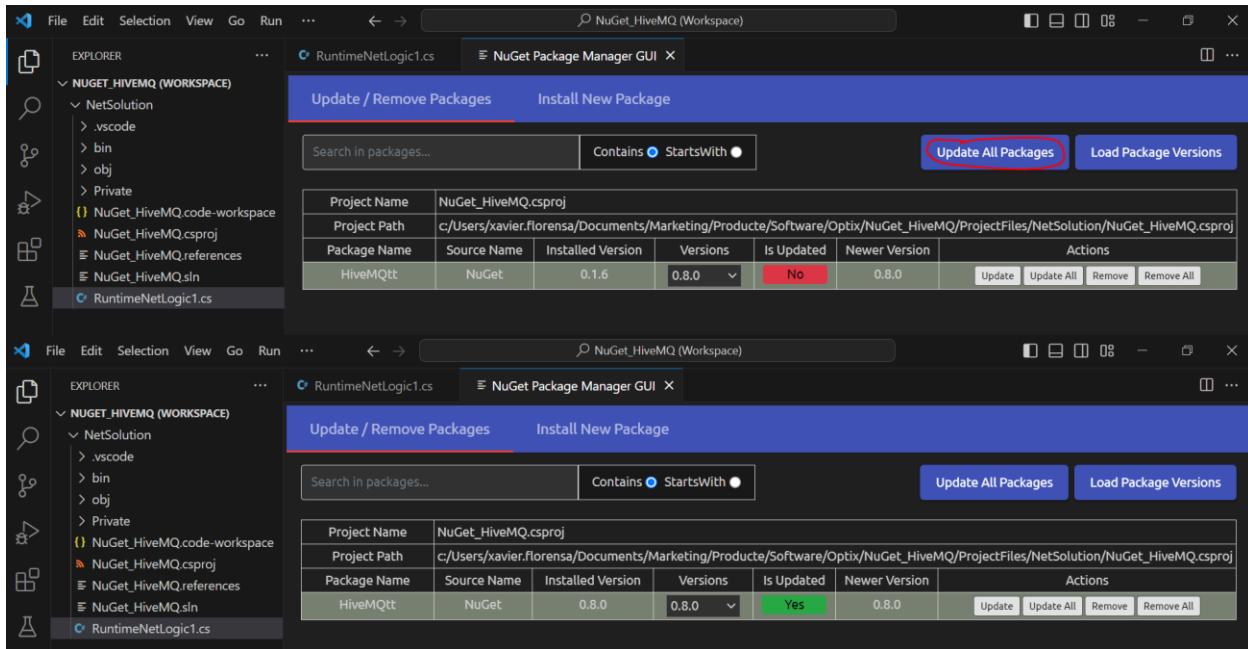
This takes just a second, now go to Update



Load package versions



Update all packages



You are done, take a look at \*.csproj file

```
</PropertyGroup>
<Import Project="NuGet_HiveMQ.references" />
<ItemGroup>
    <PackageReference Include="HiveMqtt" Version="0.8.0" />
</ItemGroup>
</Project>
```

You are ready to use this library

Copy these two directives to your code

```
using HiveMqtt.Client;
using HiveMqtt.MQTT5.Types;

#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using HiveMqtt.Client;
using HiveMqtt.MQTT5.Types;
#endregion
```

```

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
    }
}

```

Now insert this code on Start()

Be aware that the procedure is public override Start()

```

// Setup Client options and instantiate
    var options = new
HiveMQClientOptionsBuilder().WithBroker("broker.hivemq.com").WithPort(1883).WithU
seTls(false).Build();
    var client = new HiveMQClient(options);
    // Connect to the MQTT broker
    var connectResult = await client.ConnectAsync().ConfigureAwait(false);
    // Publish a message
    var publishResult = await client.PublishAsync("topic1/example", "Hello
World");

```

See these errors

```

public class RuntimeNetLogic1 : BaseNetLogic
{
    0 references
    public override void Start()
    [
        // Insert code to be executed when the user-defined logic is started
        // Setup Client options and instantiate
        var options = new HiveMQClientOptionsBuilder().WithBroker("broker.hivemq.com").WithUs
        var client = new HiveMQClient(options);
        // Connect to the MQTT broker
        var connectResult = await client.ConnectAsync().ConfigureAwait(false);
        // Publish a message
        var publishResult = await client.PublishAsync("topic1/example", "Hello World");
    ]
}

```

You can correct by typing the sentences again like this

```
var connectResult =
```

You can copy and comment the line and start writing

```
// Connect to the MQTT broker
//var connectResult = await client.ConnectAsync().ConfigureAwait(false);
var connectResult = await
```

Write client and click on the box client

```

public override void Start()
{
    // Insert code to be executed when the user-defined logic is started
    // Setup Client options and instantiate
    var options = new HiveMQClientOptionsBuilder().WithBroker("broker.hivemq.com");
    var client = new HiveMQClient(options);
    // Connect to the MQTT broker
    //var connectResult = await client.ConnectAsync().ConfigureAwait(false);
    var connectResult = await client.
        // Publish a message

```

Then . and select ConnectAsync

```

0 references
public override void Start()
{
    // Insert code to be executed when the user-defined logic is started
    // Setup Client options and instantiate
    var options = new HiveMQClientOptionsBuilder().WithBroker("broker.hivemq.com");
    var client = new HiveMQClient(options);
    // Connect to the MQTT broker
    //var connectResult = await client.ConnectAsync().ConfigureAwait(false);
    var connectResult = await client.

```

The screenshot shows an IDE's Intellisense feature. A dropdown menu is open to the right of the code editor, listing several methods starting with 'client.'. The methods listed are: AfterUnsubscribe, BeforeConnect, BeforeDisconnect, BeforeSubscribe, BeforeUnsubscribe, ConnectAsync, DisconnectAsync, Dispose, and Equals. The 'ConnectAsync' method is highlighted in the list.

. and select configureawait

```

0 references
public override void Start()
{
    // Insert code to be executed when the user-defined logic is started
    // Setup Client options and instantiate
    var options = new HiveMQClientOptionsBuilder().WithBroker("broker.hivemq.com");
    var client = new HiveMQClient(options);
    // Connect to the MQTT broker
    //var connectResult = await client.ConnectAsync().ConfigureAwait(false);
    var connectResult = await client.ConnectAsync().

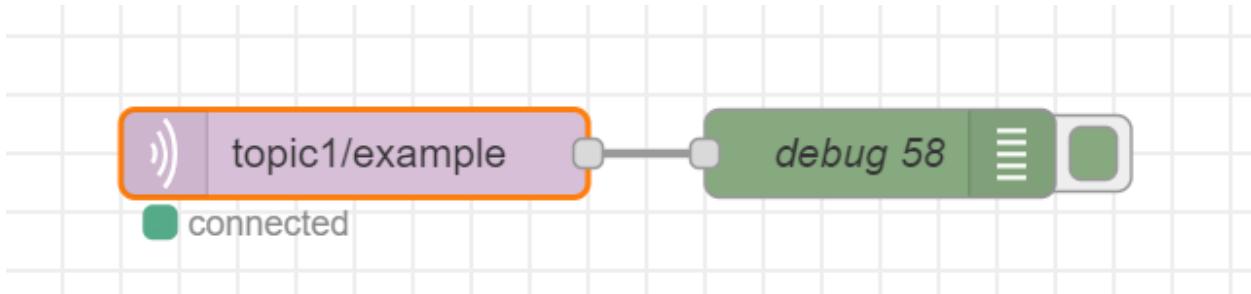
```

The screenshot shows an IDE's Intellisense feature. A dropdown menu is open to the right of the code editor, listing several methods starting with 'client.ConnectAsync()'. The methods listed are: AsyncState, ConfigureAwait, ContinueWith, ContinueWith<>, CreationOptions, Dispose, Equals, Exception, and GetAwaiter. The 'ConfigureAwait(false)' method is highlighted in the list.

and so on until you get no errors, be aware of the change the compiler has made for you

```
public class RuntimeNetLogic1 : BaseNetLogic
{
    0 references
    public override async void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        // Setup Client options and instantiate
        var options = new HiveMQClientOptionsBuilder().WithBroker("broker.hivemq.com").With
        var client = new HiveMQClient(options);
        // Connect to the MQTT broker
        var connectResult = await client.ConnectAsync().ConfigureAwait(false);
        // Publish a message
        var publishResult = await client.PublishAsync("topic1/example", "Hello World");
    }
}
```

Then save and before executing open a node-red instance to verify the Optix project is working fine



Node-RED

### Edit mqtt in node

Delete      Cancel      Done

**Properties**

Server: broker.hivemq.com:1883

Action: Subscribe to single topic

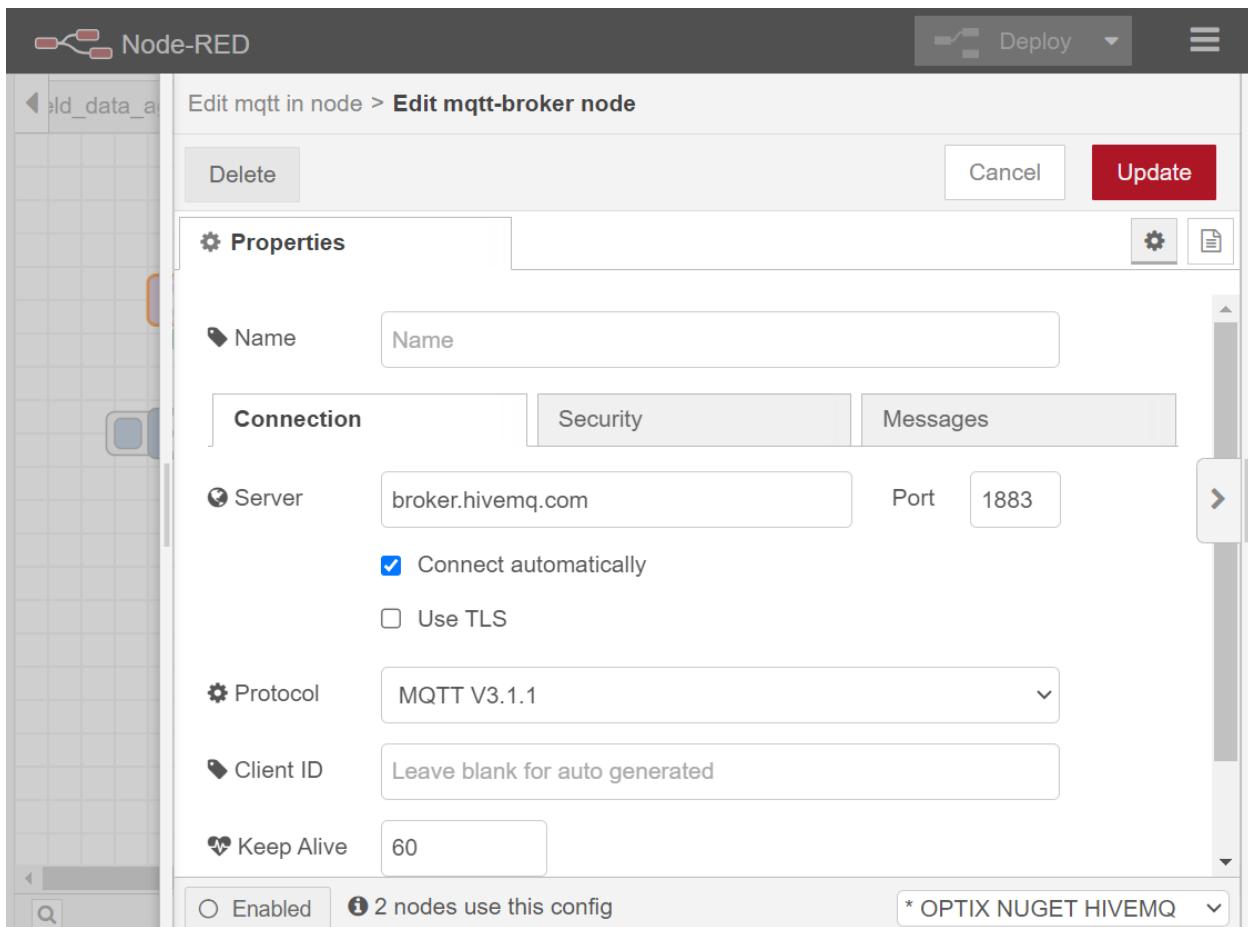
Topic: topic1/example

QoS: 2

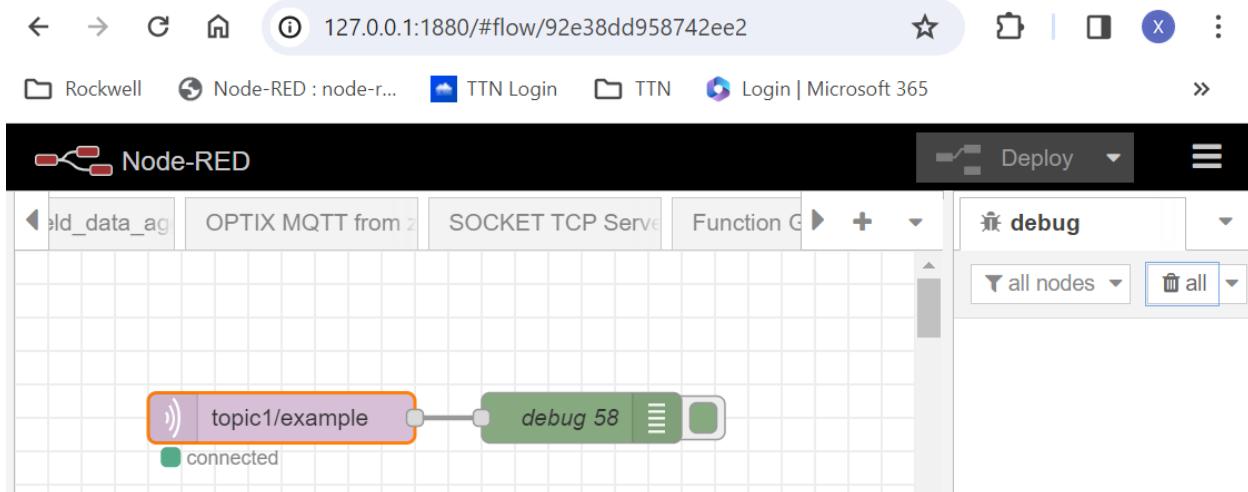
Output: auto-detect (parsed JSON object, string or list)

Name: Name

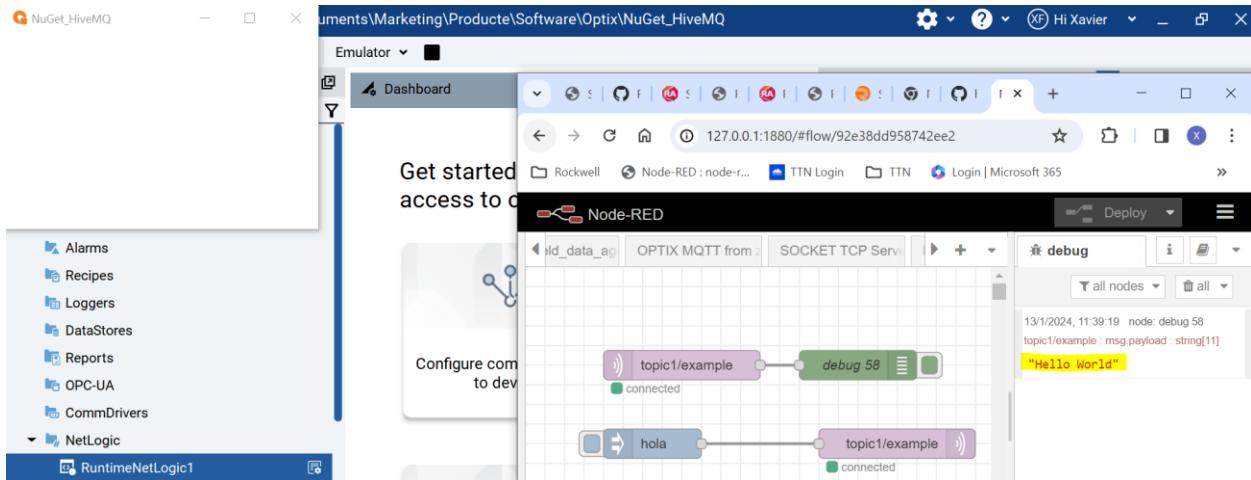
Enabled



Prepare the node-red instance



Then execute the Optix application



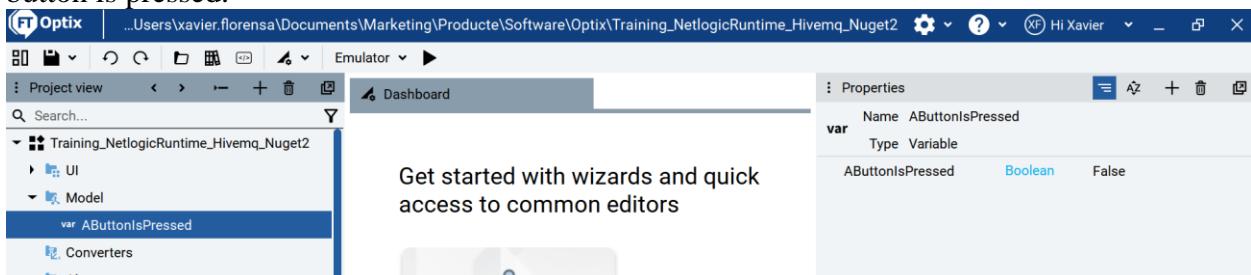
That's all.

Here you have connection and publishing message in one shot.

The ideal would be, first connecting, and then asynchronously publish.

You can try with this code, but be careful with infinite loops that may cause uncontrolled program behaviour.

Let's create a variable to store a boolean value that will be set to true and then false when a button is pressed.



Then use this code

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Rettentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using HiveMQtt.Client;
using HiveMQtt.MQTT5.Types;
using System.Net.Http;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
```

```

public override async void Start()
{
    bool abuttonispressed=false;
    var Abuttonispressed =
Project.Current.GetVariable("Model/AButtonIsPressed");
    // Insert code to be executed when the user-defined logic is started
    // Setup Client options and instantiate
    var options = new
HiveMQClientOptionsBuilder().WithBroker("broker.hivemq.com")
    .WithPort(1883)
    .WithUseTls(false)
    .Build();

    var client = new HiveMQClient(options);
    // Connect to the MQTT broker
    var connectResult = await client.ConnectAsync().ConfigureAwait(false);
    //hold until a button is pressed
    while (true)
    {
        while (!abuttonispressed)
        {
            //holds until a button is pressed

            abuttonispressed = Abuttonispressed.Value;
        }
        // Publish a message
        var publishResult = await client.PublishAsync("topic2/example", "Hello
World");
        abuttonispressed=false;
        Abuttonispressed.Value=false;
    }
}

public override void Stop()
{
    // Insert code to be executed when the user-defined logic is stopped
}
[ExportMethod]

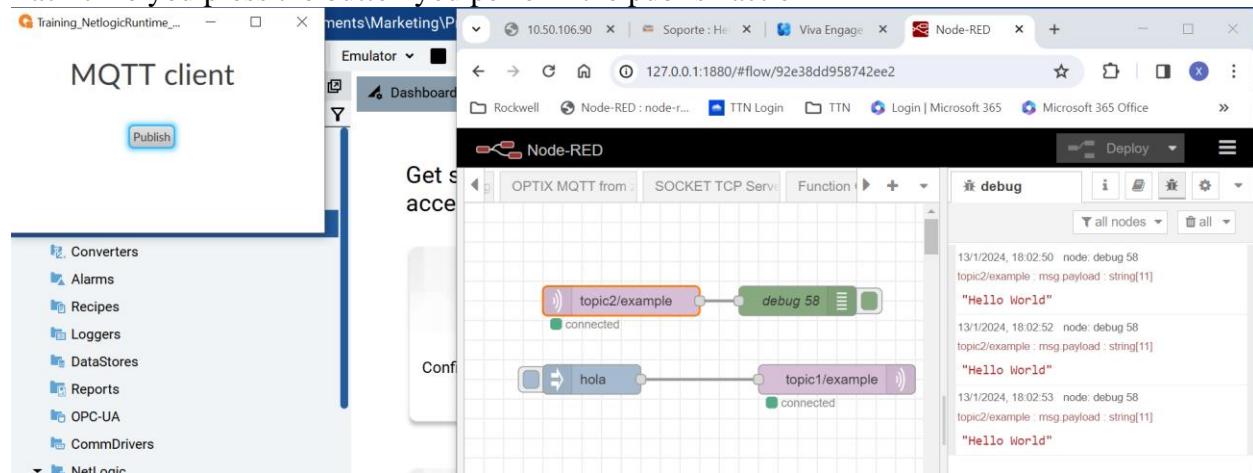
public void Publish()
{
    var Abuttonispressed =
Project.Current.GetVariable("Model/AButtonIsPressed");
    Abuttonispressed.Value=true;
}

```

}

Now it works

Each time you press the button you perform the publish action



As you can see on this video

<https://youtu.be/3g81mO2wqXQ>

## 45. Modbus server with FTOptix

You can try this example on Modbus TCP Server from Rockwell Automation repository

You can download the Server FactoryTalk Optix code from here

<https://github.com/FactoryTalk-Optix/ModbusServerTCP>

You can see this working in this video

<https://youtu.be/UpRB72bkKkA>



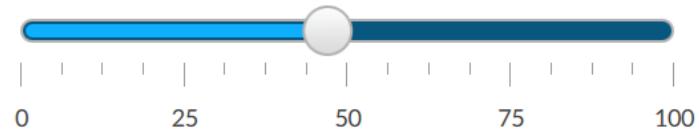
A ROCKWELL AUTOMATION COMPANY

Instructions

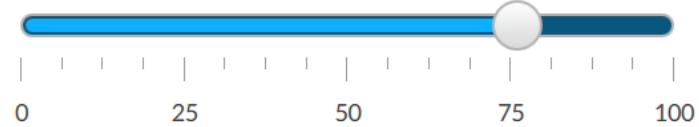
Modbus TCP Server

## MODBUS SERVER TCP

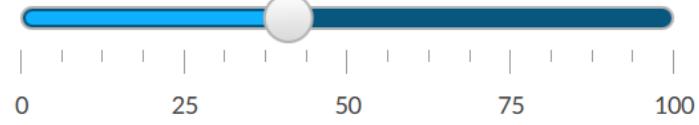
VARIABLE 1  
HR 0



VARIABLE 2  
HR 1



VARIABLE 3  
HR 2



Let's run the Optix application, and open a Modbus client

The screenshot shows a dual-pane interface. On the left, the FTOptixRuntime application displays three analog input variables: VARIABLE 1 (HR 0), VARIABLE 2 (HR 1), and VARIABLE 3 (HR 2). On the right, the Node-RED interface shows a flow for reading Modbus-TCP data. The flow consists of a timestamp node, a function node labeled 'function 8', a Modbus-TCP Flexi Getter node, and a debug node. The debug node output shows a message: '15/1/2024, 20:14:25 node: debug 44 msg.responseBuffer.buffer: buffer[4] > [ 0, 100, 0, 50 ]'. Below the Node-RED interface is an 'Edit function node' dialog for 'function 8'. It has tabs for Setup, On Start, On Message (selected), and On Stop. The On Message tab contains the following JavaScript code:

```

1 msg.payload = { "unitid": 1, "functioncode": 3, "address": 0, "quantity": 2 }
2 return msg;

```

### Edit modbus-tcp-ip node

[Delete](#) [Cancel](#) [Done](#)

**Properties**

Name	Modbus-TCP Flexi Getter
IP	127.0.0.1
Port	502
Log Errors	<input type="checkbox"/>

### Edit debug node

[Delete](#) [Cancel](#) [Done](#)

**Properties**

Output	<input type="button" value="▼ msg. responseBuffer.buffer"/>
To	<input checked="" type="checkbox"/> debug window
	<input type="checkbox"/> system console
	<input checked="" type="checkbox"/> node status (32 characters)
	<input type="button" value="▼ same as debug output"/>
Name	debug 44

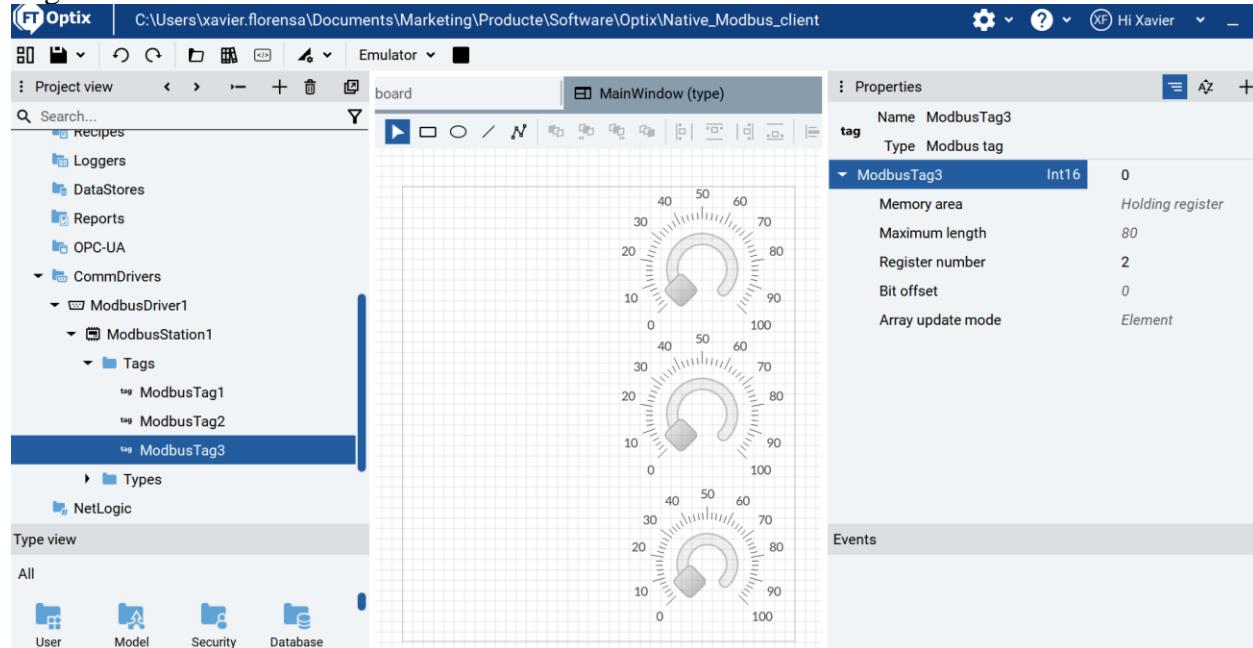
## 46. Native Modbus TCP client with FactoryTalk Optix

You can see it on this video

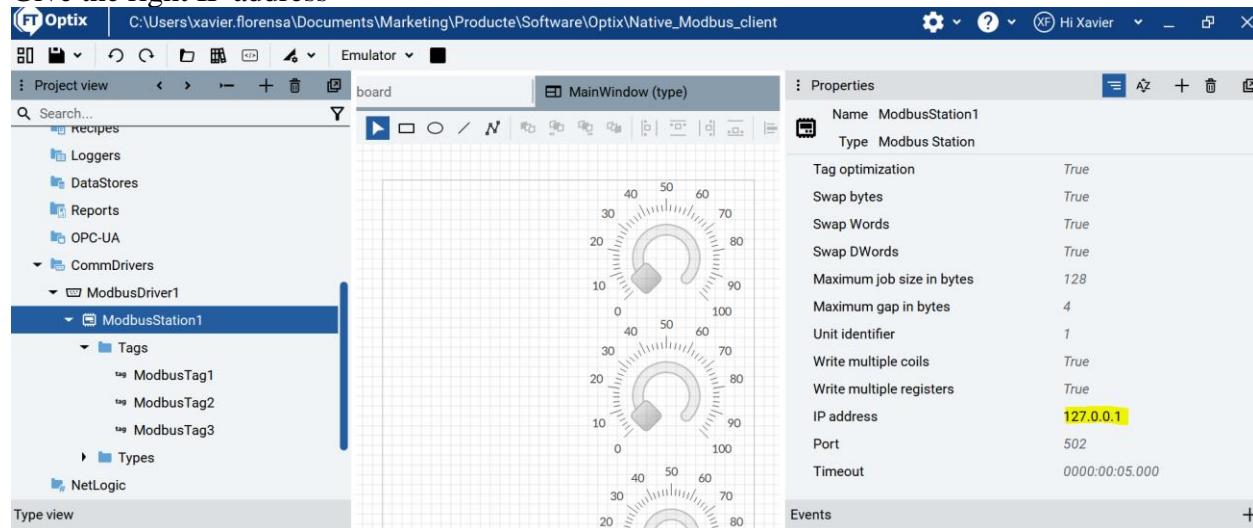
<https://youtu.be/UpRB72bkKkA>

Since FT Optix has drivers for Modbus, let's use it to test the above server

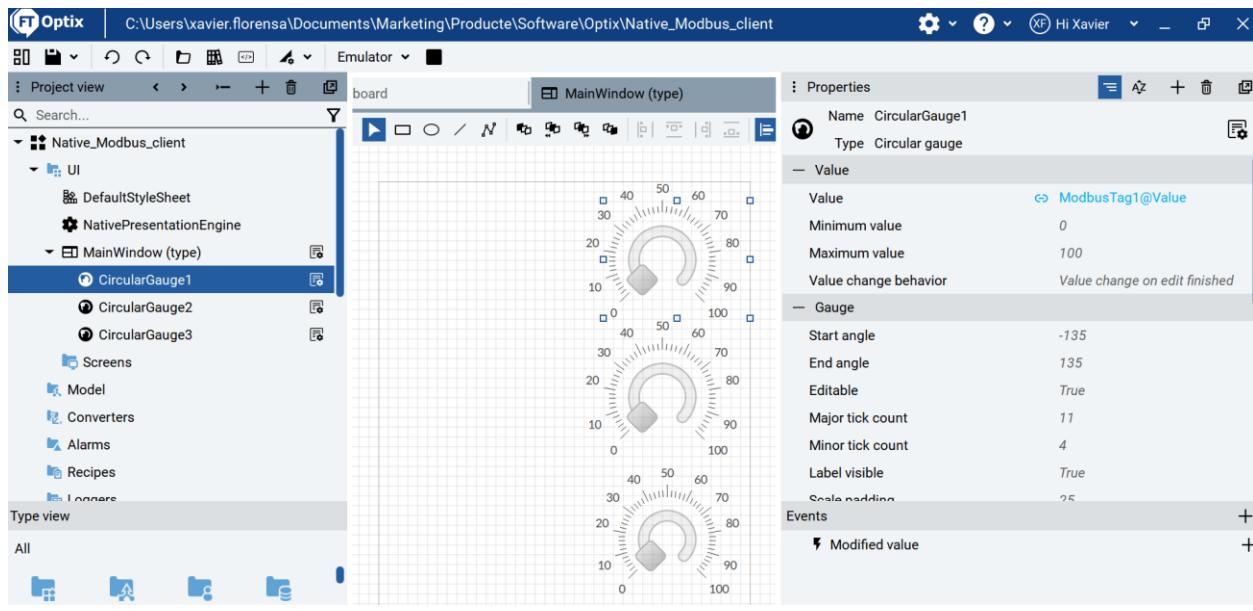
Build and application and on Communication drivers add a Modbus driver, and declare three Tags



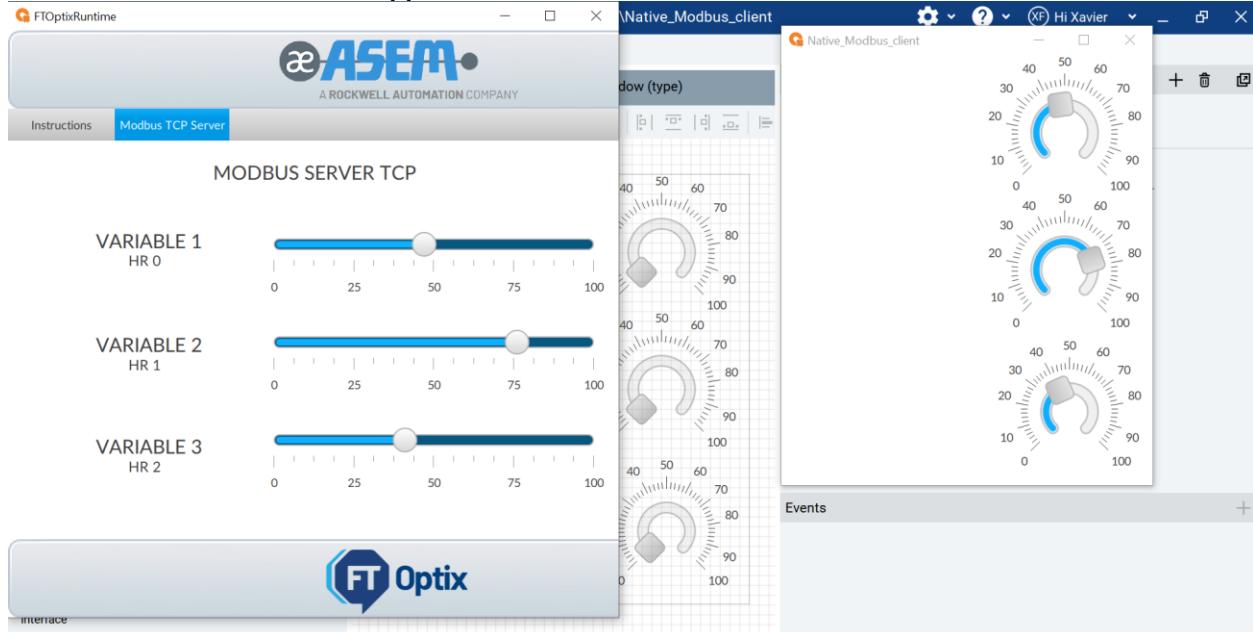
Give the right IP address



Make the corresponding dynamic links between Modbus variables and Values on circular gauges.



Run the server, and start the application to test our client



Here you can find the code

[https://github.com/xavierflorensa/Optix\\_Native\\_Modbus\\_client](https://github.com/xavierflorensa/Optix_Native_Modbus_client)

## 47. DotNet ModbusTCP client with C# and FactoryTalk Optix

Let's try to build a Modbus client application with FT Optix and the EasyModbus library  
Here the dll library EasyModbusTCP is used

<https://www.nuget.org/packages/EasyModbusTCP>

You can see how to use this library (with a client)

<https://www.youtube.com/watch?v=qcyJoDu7cok&t=726s>

You can see the final result on this video

<https://youtu.be/UpRB72bkKkA>

Copy the two dll from the FTOptix example

This PC > Documents > Marketing > Producte > Software > Optix > ModbusServerTCP > ProjectFiles > NetSolution > bi

Name	Date modified	Type	Size
runtimes	1/15/2024 7:36 PM	File folder	
EasyModbus.dll	12/31/2020 2:44 PM	Application extension	71 KB
ModbusServerTCP.deps	1/15/2024 8:33 PM	JSON Source File	7 KB
ModbusServerTCP.dll	1/15/2024 8:33 PM	Application extension	7 KB
ModbusServerTCP	1/15/2024 8:33 PM	PDB File	15 KB
System.IO.Ports.dll	10/18/2022 6:29 PM	Application extension	37 KB

And locate anywhere in your new client project

This PC > Documents > Marketing > Producte > Software > Optix > ModbusClientTCP > ProjectFiles

Name	Date modified	Type	Size
NetSolution	1/15/2024 8:43 PM	File folder	
PKI	1/15/2024 8:31 PM	File folder	
EasyModbus.dll	12/31/2020 2:44 PM	Application extension	71 KB
System.IO.Ports.dll	10/18/2022 6:29 PM	Application extension	37 KB
UserDefinedModule	1/15/2024 8:42 PM	Microsoft Edge HTM...	1 KB

Create a new Runtime Netlogic

Installing libraries with Visual Studio 2022

change editor to Visual Studio 2022

Open code and go to

Project / Add references

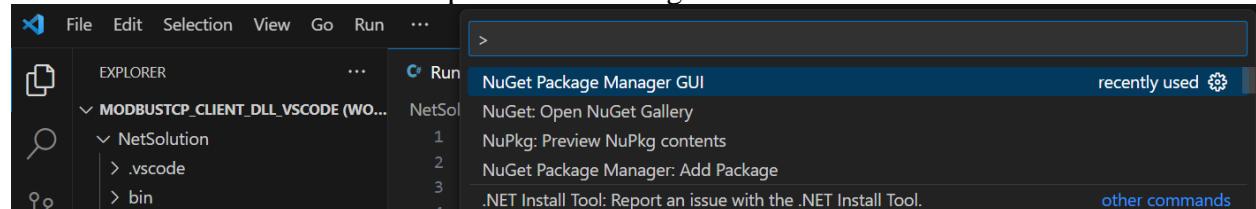
Browse and install the two dll one after the other.

Close Visual Studio.

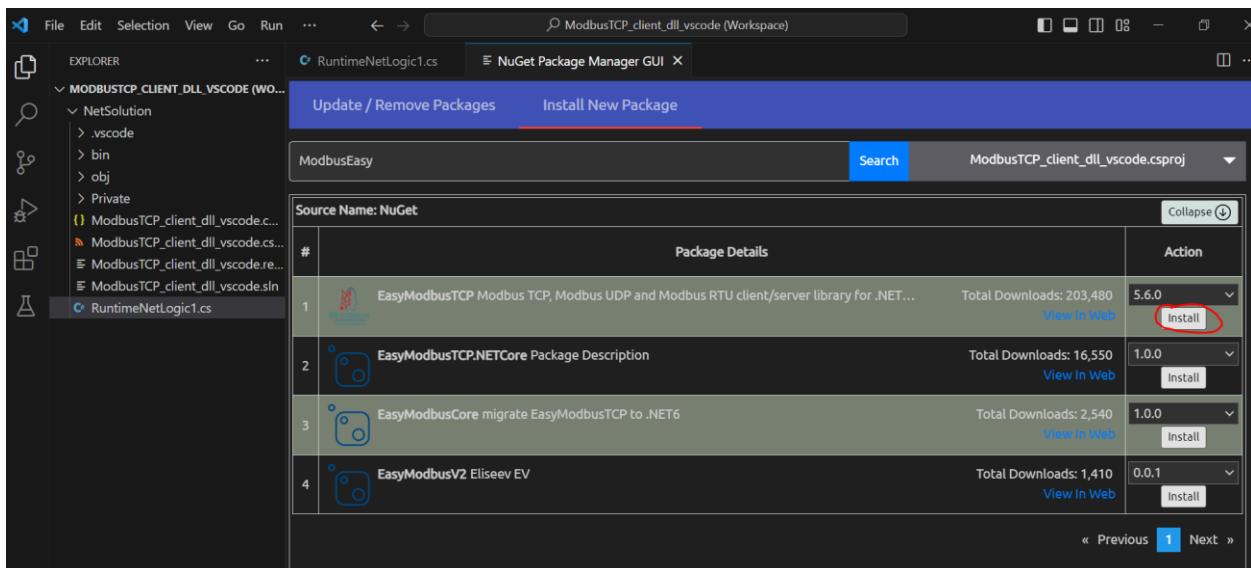
Installing libraries with Visual Studio Code

You can also do this on VScode, this way

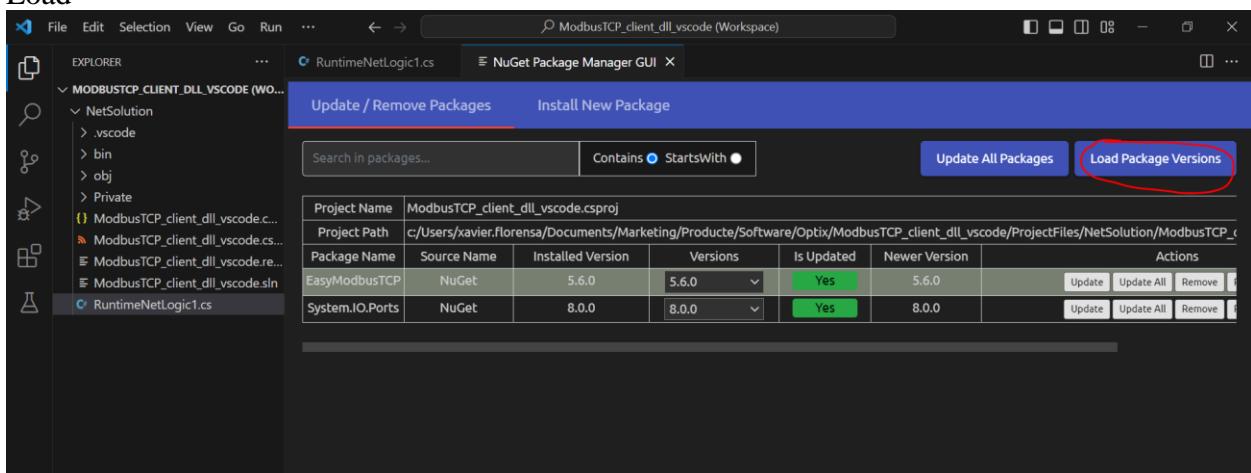
Got to View/Command Pallette/Open NUGetManager UI



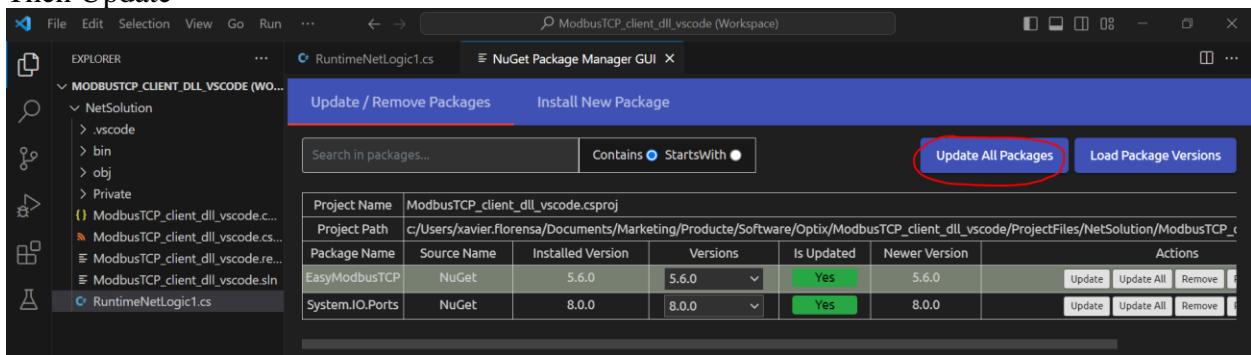
Search for ModbusEasy. Install



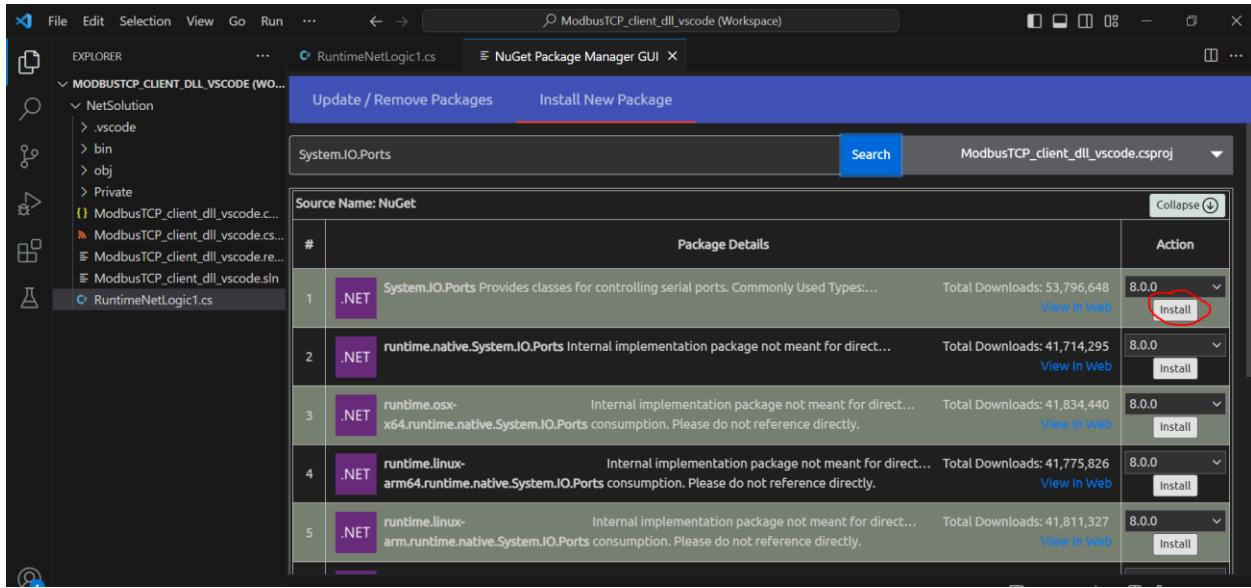
Then go to Update  
Load



Then Update



Install also System.IO.Ports



Then Load and update

Open the code,

Look at the \*.csproj file

```
<Reference Include="EasyModbus">
    <HintPath>..\EasyModbus.dll</HintPath>
</Reference>
<Reference Include="System.IO.Ports">
    <HintPath>..\System.IO.Ports.dll</HintPath>
</Reference>
```

Now add the using directive

Using EasyModbus;

And this code inside the start procedure

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Rentativity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using EasyModbus;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
```

```

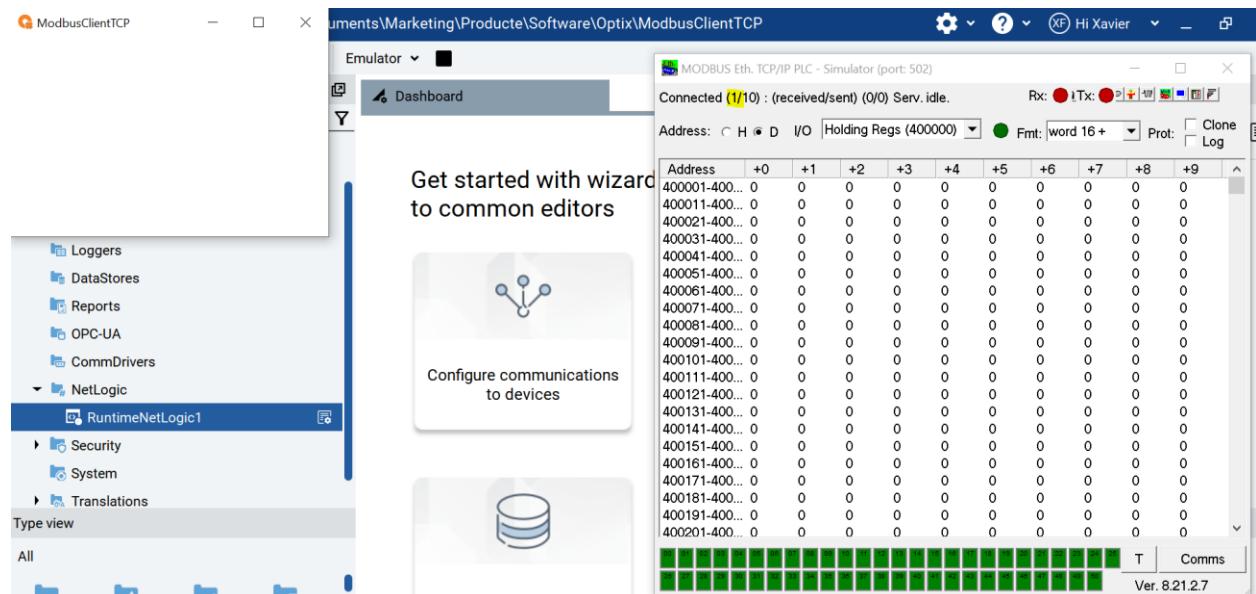
{
    public override void Start()
    {
        ModbusClient myModbusClient = new ModbusClient("127.0.0.1", 502);
        myModbusClient.Connect();
        // Insert code to be executed when the user-defined logic is started
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}

```

Execute the project

Opening a Server, we are connected to the server



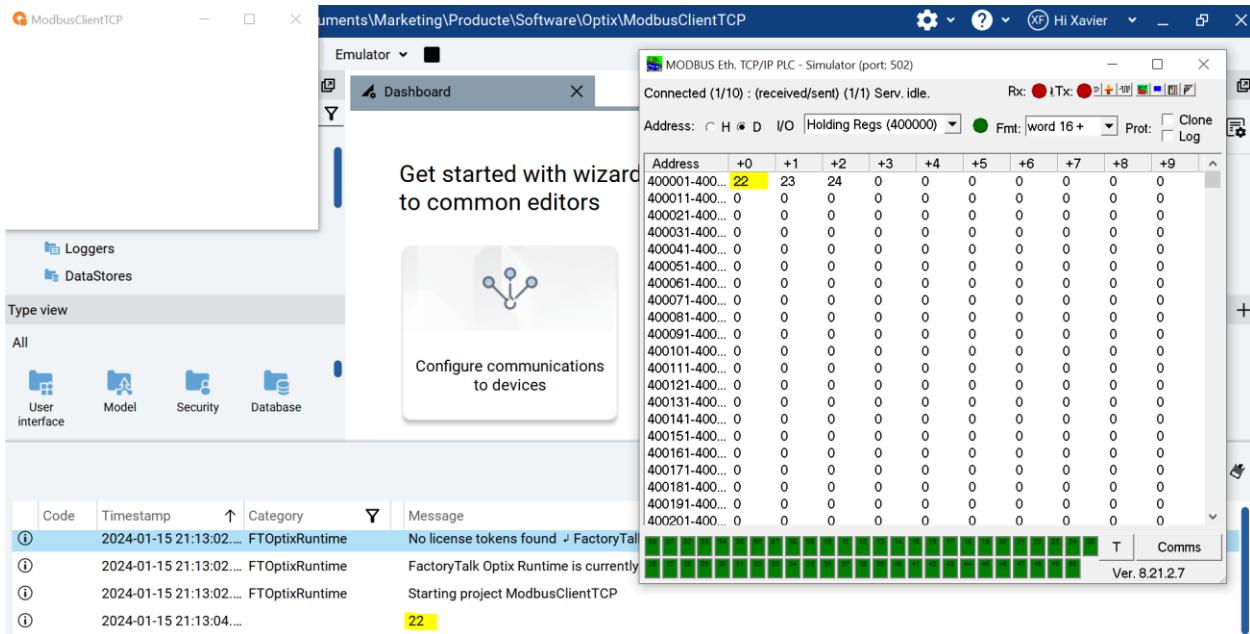
Now let's read a holding register

With this code

```

int[] readHoldingRegisters = myModbusClient.ReadHoldingRegisters(0, 2);
Log.Info(readHoldingRegisters[0].ToString());

```



This is the code

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using EasyModbus;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        ModbusClient myModbusClient = new ModbusClient("127.0.0.1", 502);
        myModbusClient.Connect();
        int[] readHoldingRegisters = myModbusClient.ReadHoldingRegisters(0, 2);
        Log.Info(readHoldingRegisters[0].ToString());
        // Insert code to be executed when the user-defined logic is started
    }

    public override void Stop()
    {
```

```

        // Insert code to be executed when the user-defined logic is stopped
    }
}

```

Let's use a dashboard,  
Add a new Gauge to your project and use this code

```

var myGauge =
Project.Current.Get<CircularGauge>("UI/MainWindow/CircularGauge1");
myGauge.Value=readHoldingRegisters[0];

```

This is the whole code

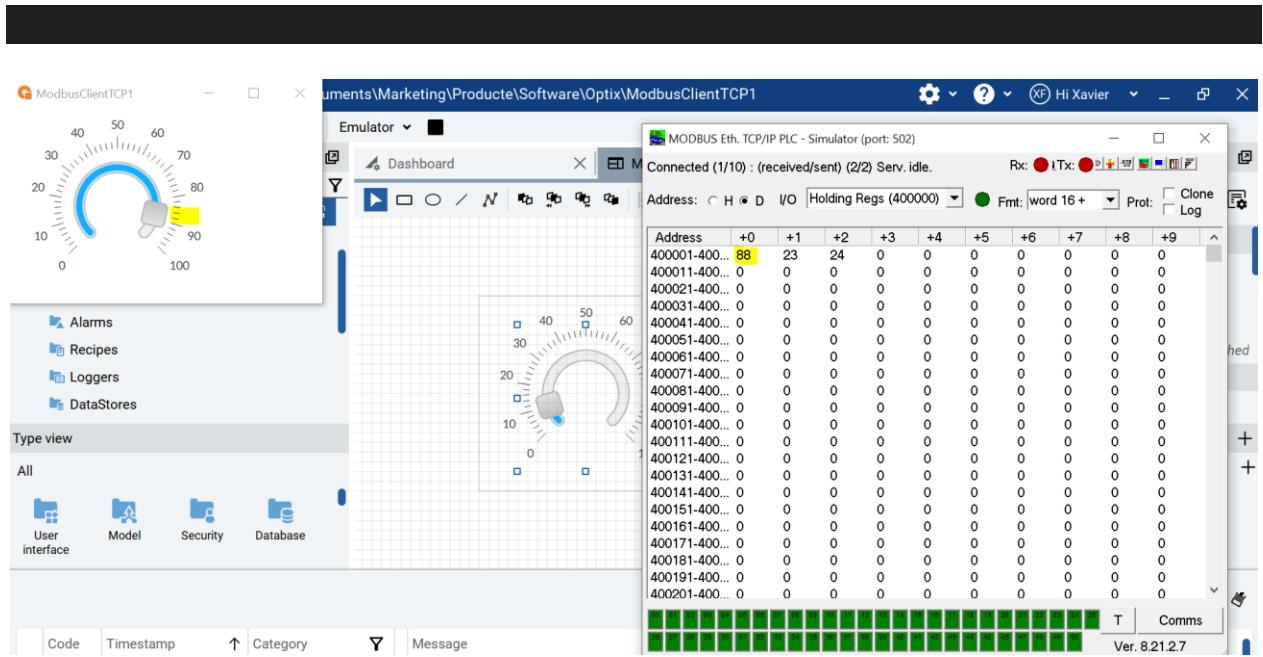
```

#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using EasyModbus;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        ModbusClient myModbusClient = new ModbusClient("127.0.0.1",502);
        myModbusClient.Connect();
        int[] readHoldingRegisters = myModbusClient.ReadHoldingRegisters(0,2);
        Log.Info(readHoldingRegisters[0].ToString());
        // Insert code to be executed when the user-defined logic is started
        var myGauge =
Project.Current.Get<CircularGauge>("UI/MainWindow/CircularGauge1");
        myGauge.Value=readHoldingRegisters[0];
    }

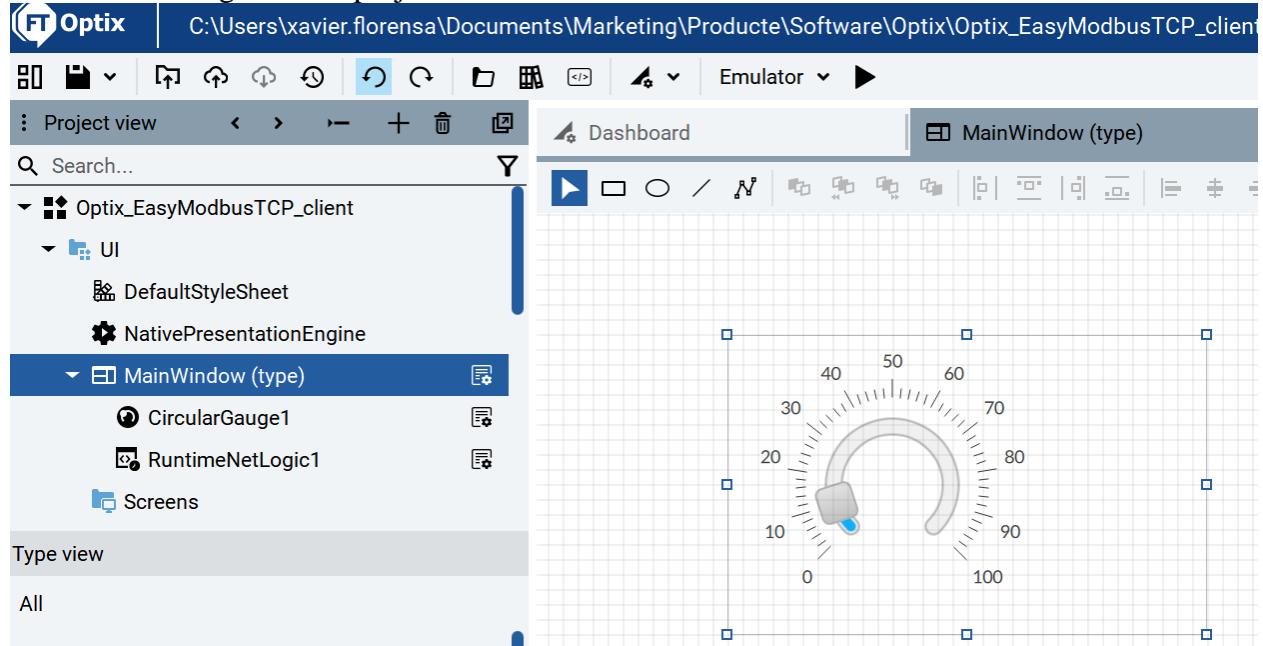
    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}

```



But we need a periodic task to update the value

First of all, delete the former Netlogic and create a Netlogic under the Main Window (it will not work with Netlogic on the project level)



Use this logic and test the project

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Retentivity;
using FTOptix.NetLogic;
```

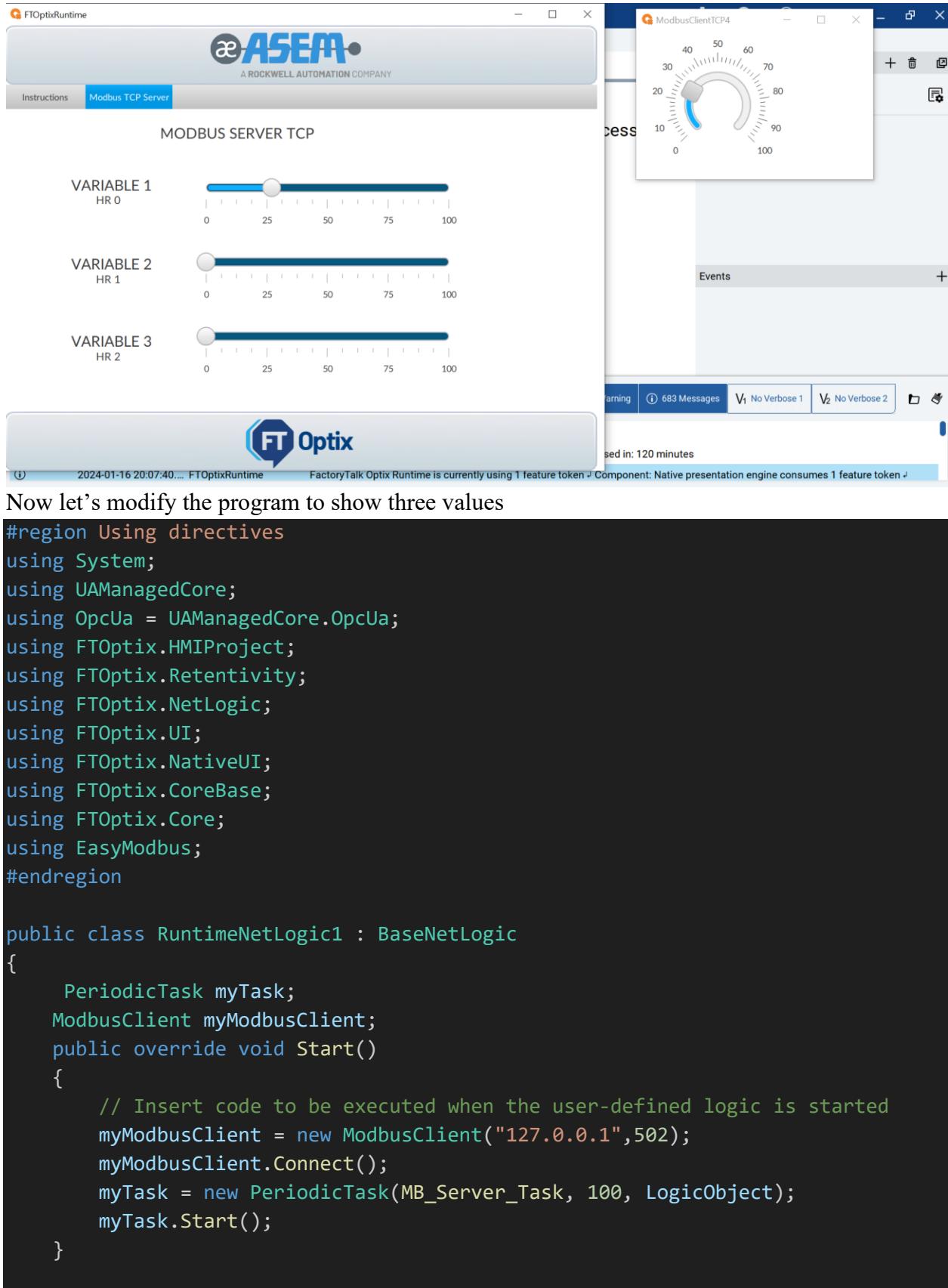
```

using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using EasyModbus;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    PeriodicTask myTask;
    ModbusClient myModbusClient;
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        myModbusClient = new ModbusClient("127.0.0.1",502);
        myModbusClient.Connect();
        int[] readHoldingRegisters = myModbusClient.ReadHoldingRegisters(0,2);
        Log.Info(readHoldingRegisters[0].ToString());
        myTask = new PeriodicTask(MB_Server_Task, 100, LogicObject);
        myTask.Start();
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
        myTask.Dispose();
    }
    private void MB_Server_Task()
    {
        int[] readHoldingRegisters = myModbusClient.ReadHoldingRegisters(0,2);
        Log.Info(readHoldingRegisters[0].ToString());
        Owner.Get<CircularGauge>("CircularGauge1").Value=readHoldingRegisters[0];
    }
}

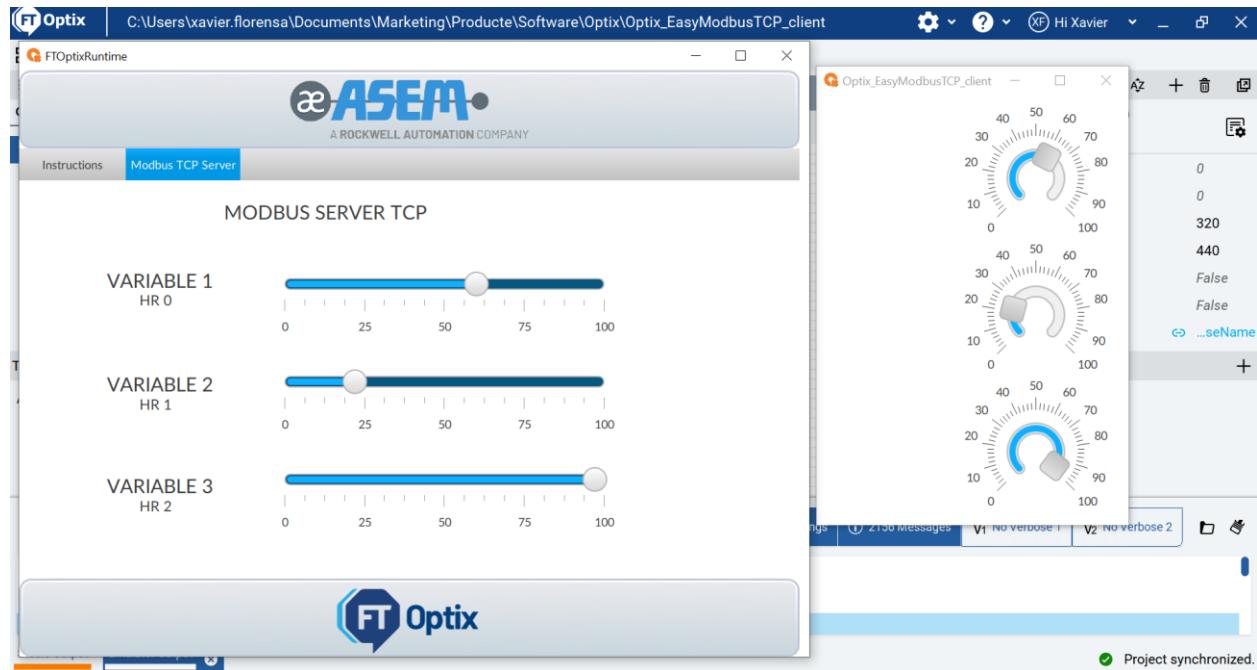
```



```

public override void Stop()
{
    // Insert code to be executed when the user-defined logic is stopped
    myTask.Dispose();
}
private void MB_Server_Task()
{
    int[] readHoldingRegisters = myModbusClient.ReadHoldingRegisters(0,3);
    Owner.Get<CircularGauge>("CircularGauge1").Value=readHoldingRegisters[0];
    Owner.Get<CircularGauge>("CircularGauge2").Value=readHoldingRegisters[1];
    Owner.Get<CircularGauge>("CircularGauge3").Value=readHoldingRegisters[2];
}
}

```



You have the code here

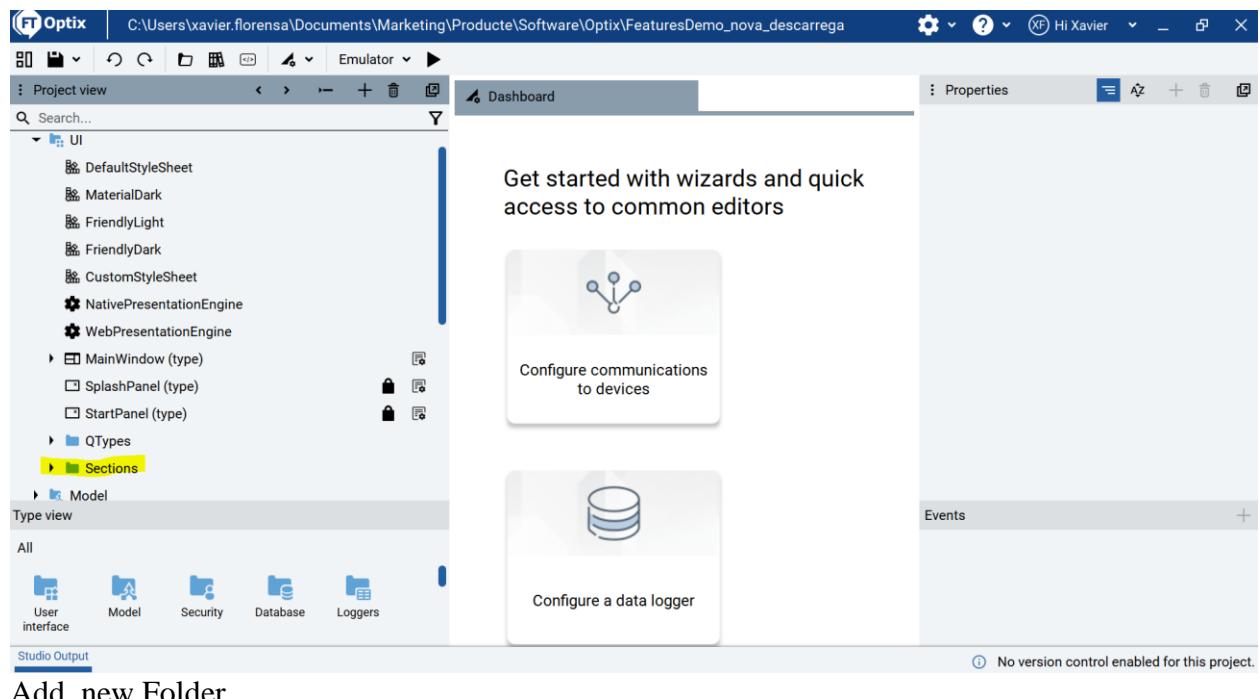
[https://github.com/xavierflorensa/Optix\\_EasyModbusTCP\\_client](https://github.com/xavierflorensa/Optix_EasyModbusTCP_client)

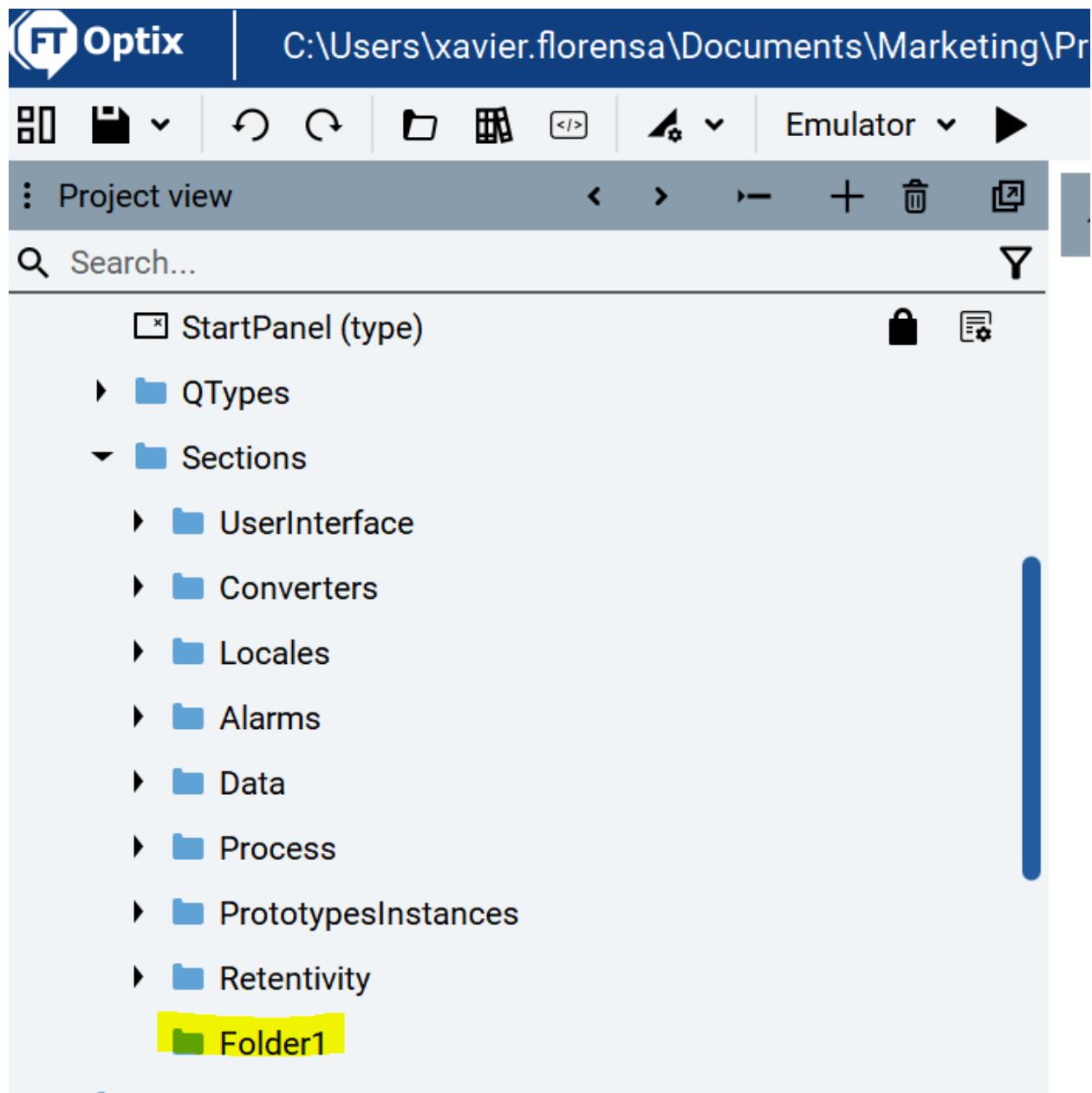
## 48. Modifying Existing projects for page navigation

Let's add some custom examples to the Features Demo on additional buttons and pages

Copy the Features demo to a new project

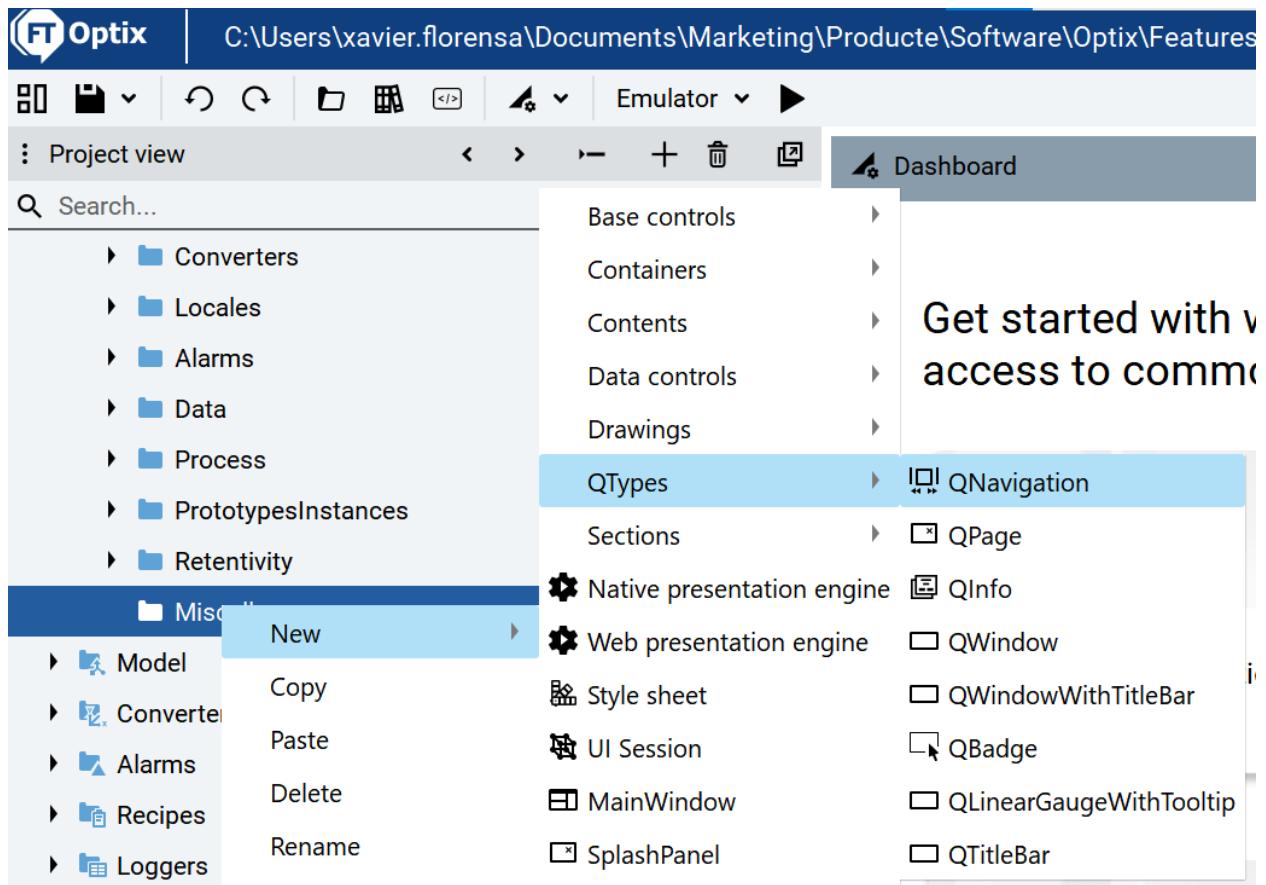
Go to Sections Folder



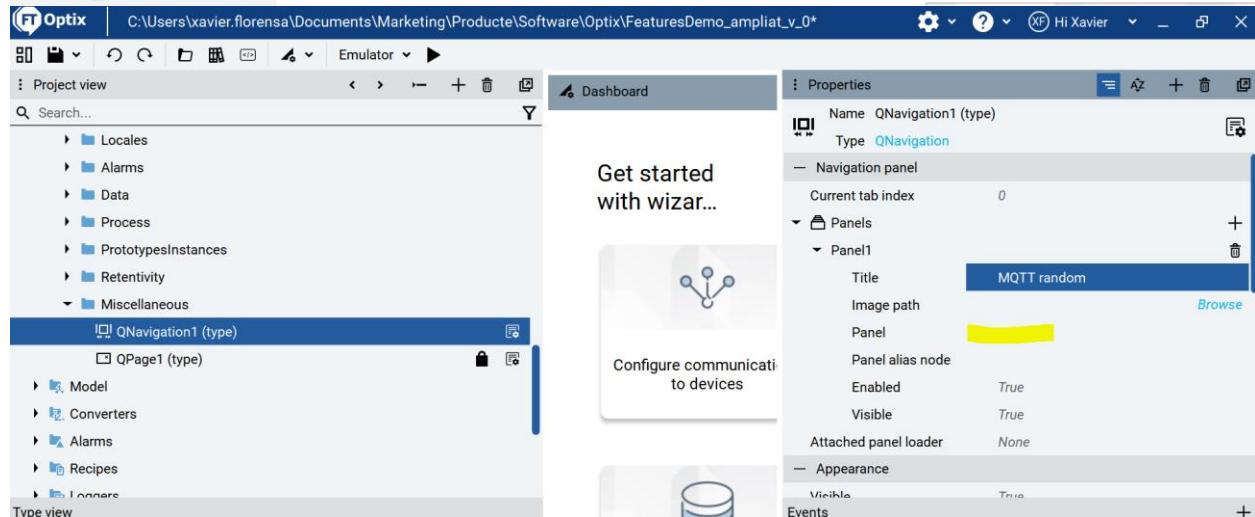
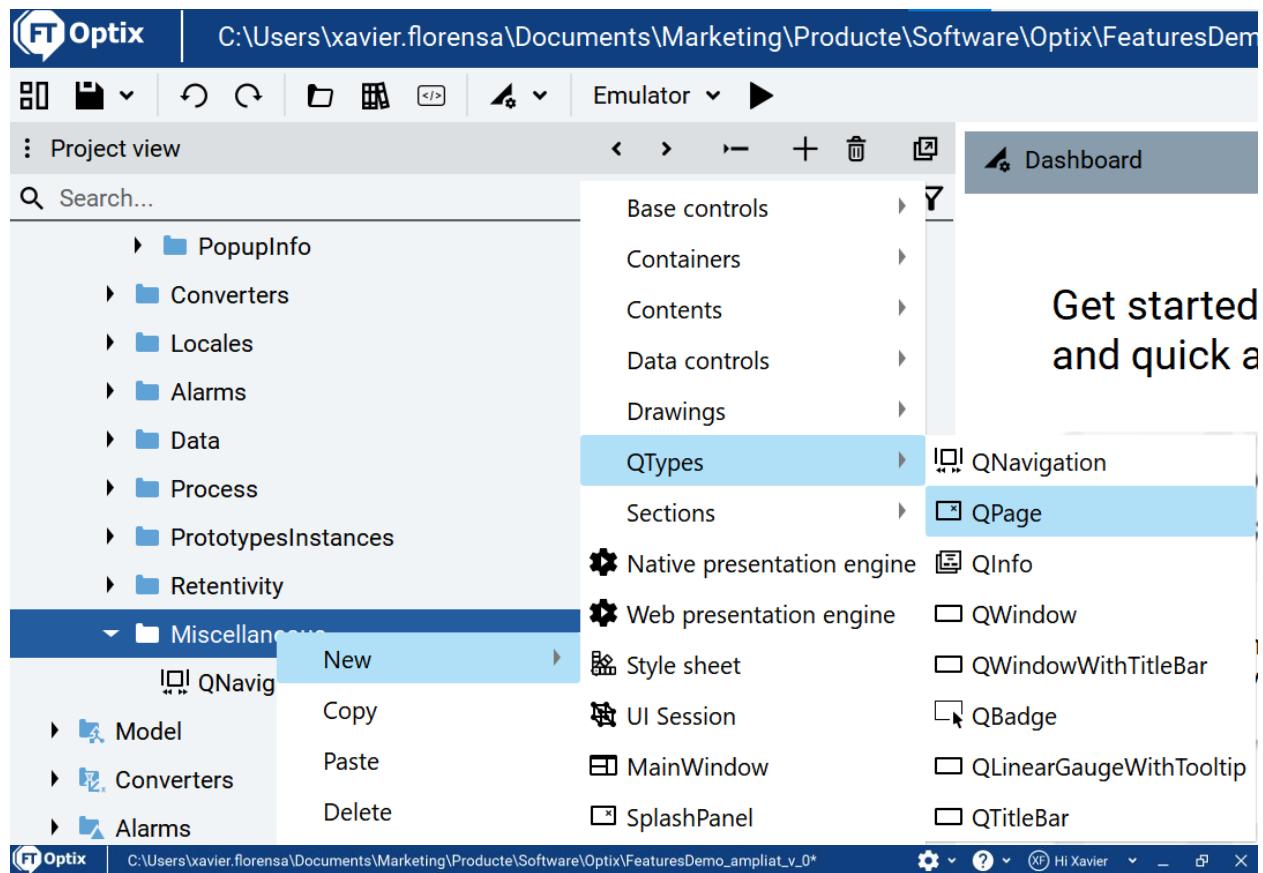


Rename it

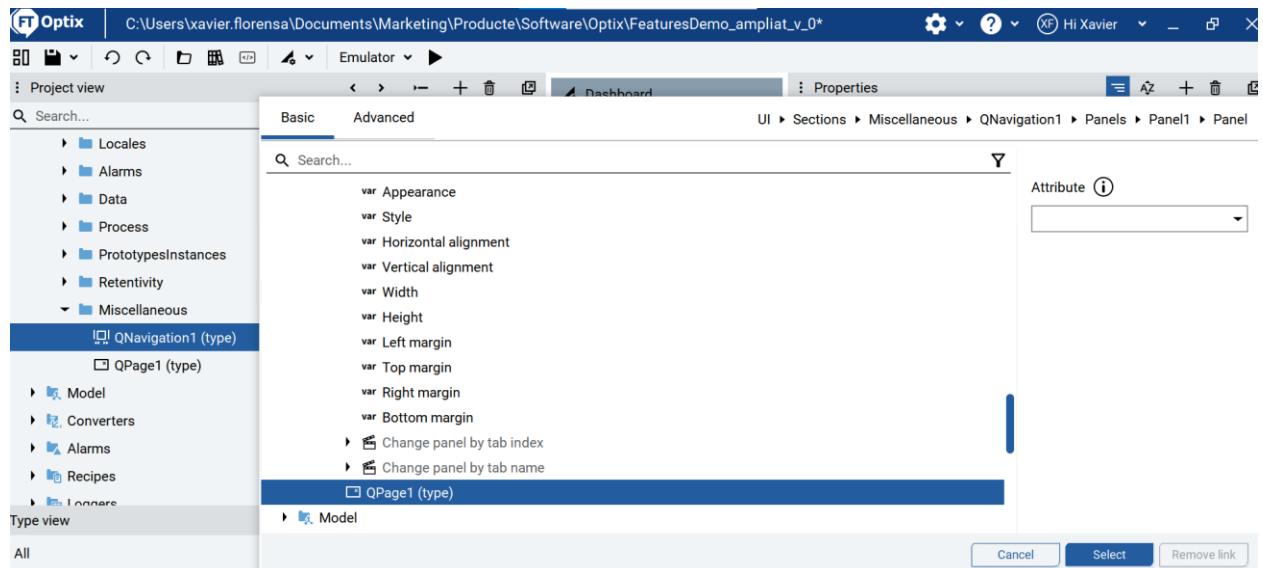
Add a new Qnavigation



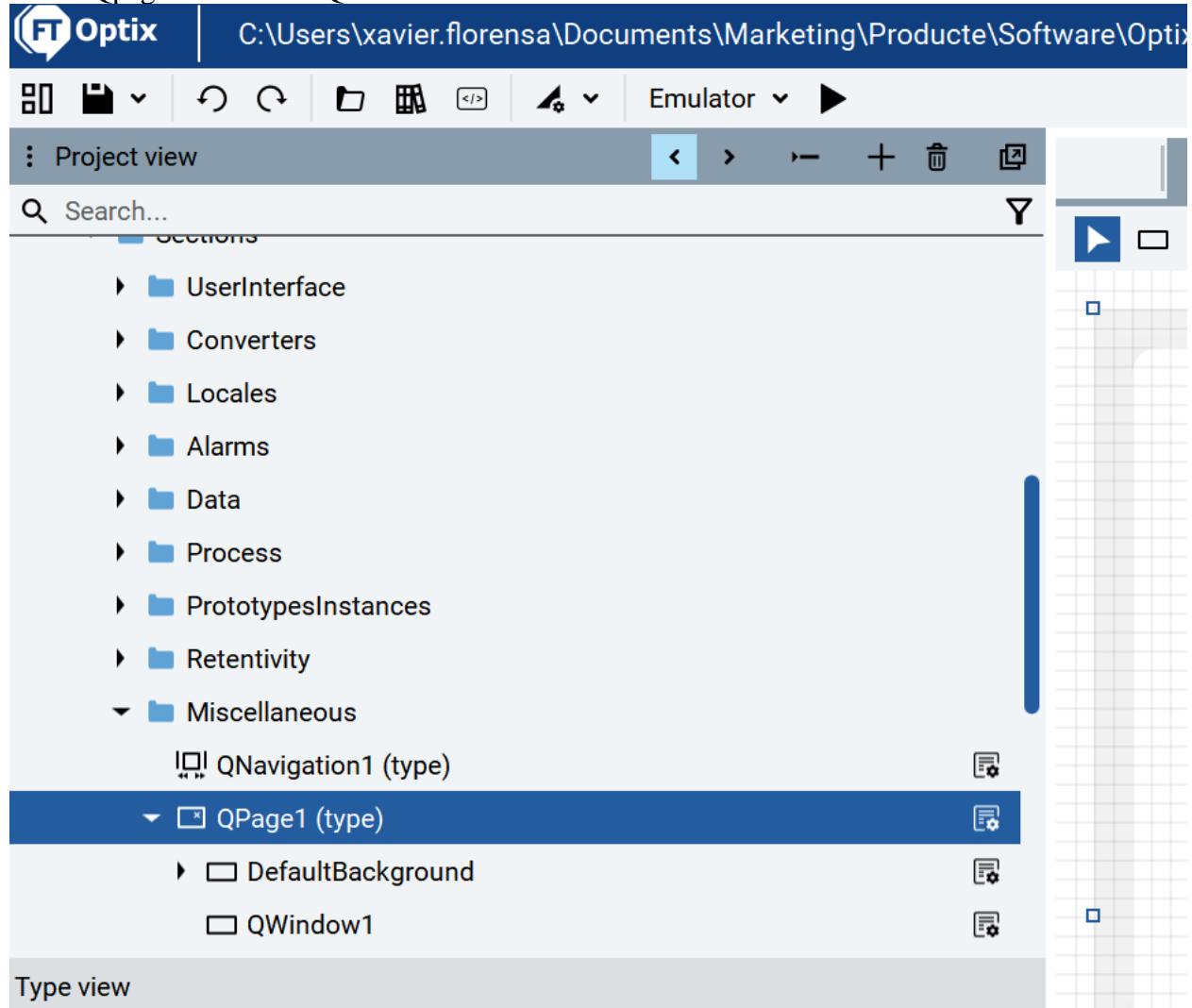
Add a new QPage



Add the dynamic link to QPage1

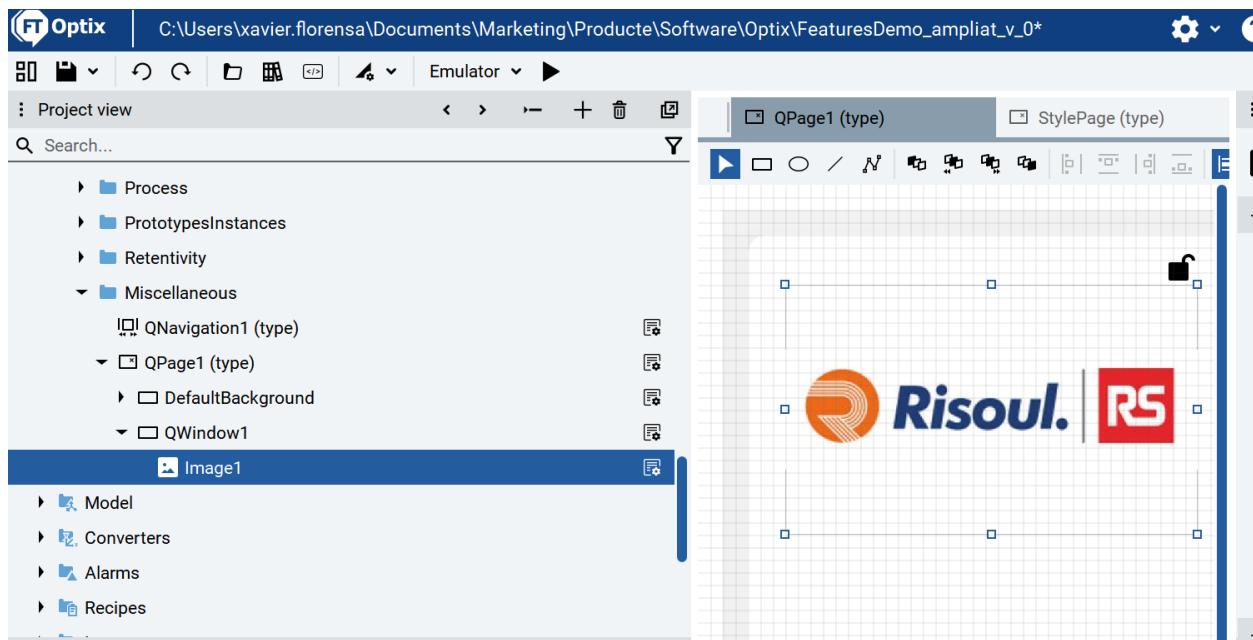


Inside Qpage1 add a new Qwindow

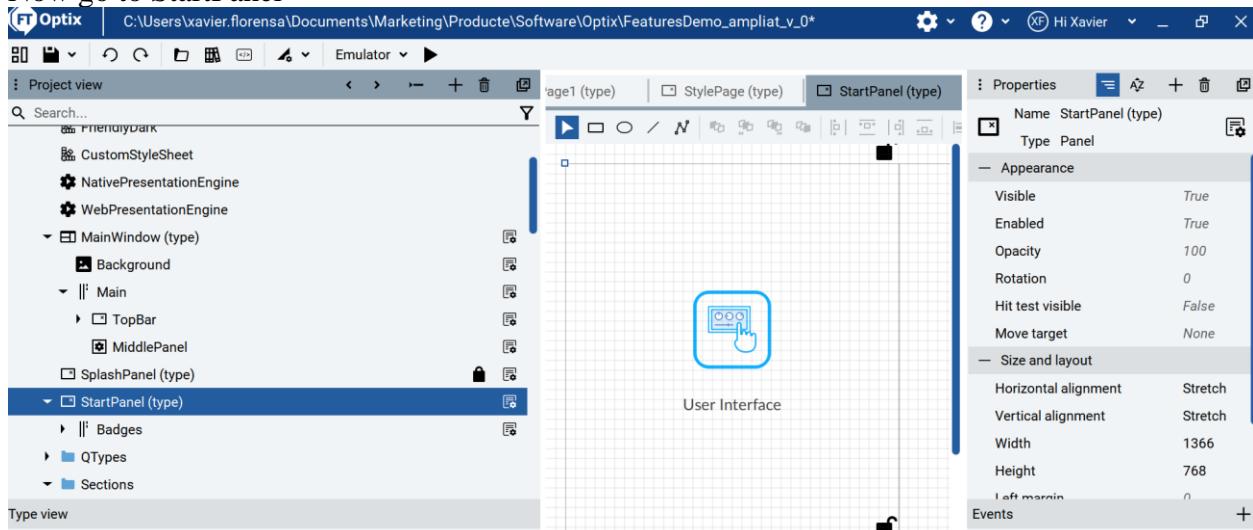


Type view

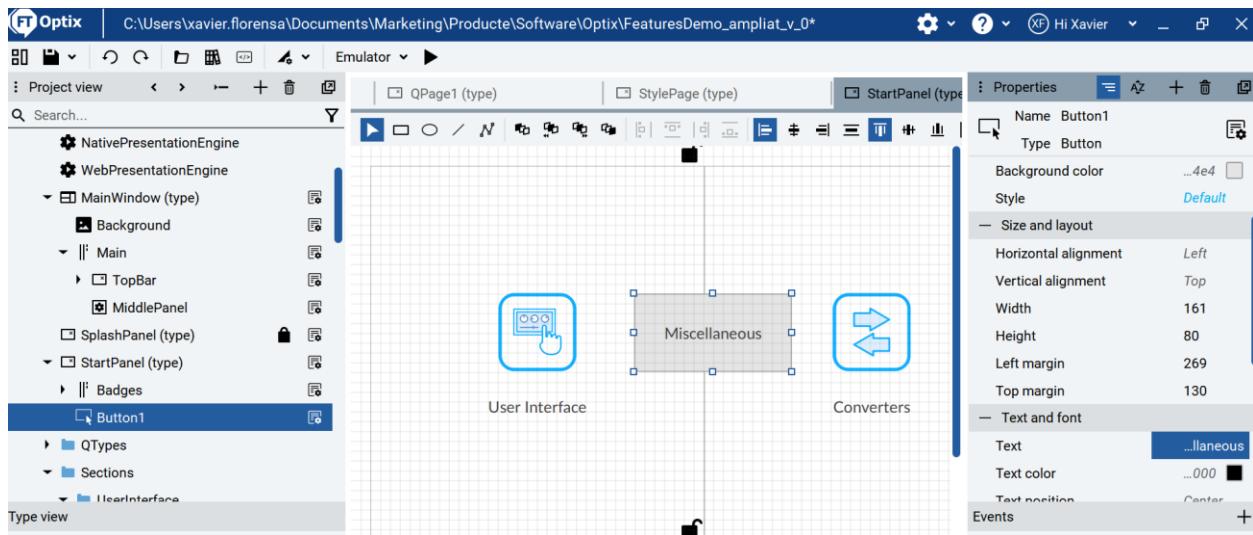
Add a new image so we can recognize our new window on the runtime



Now go to StartPanel

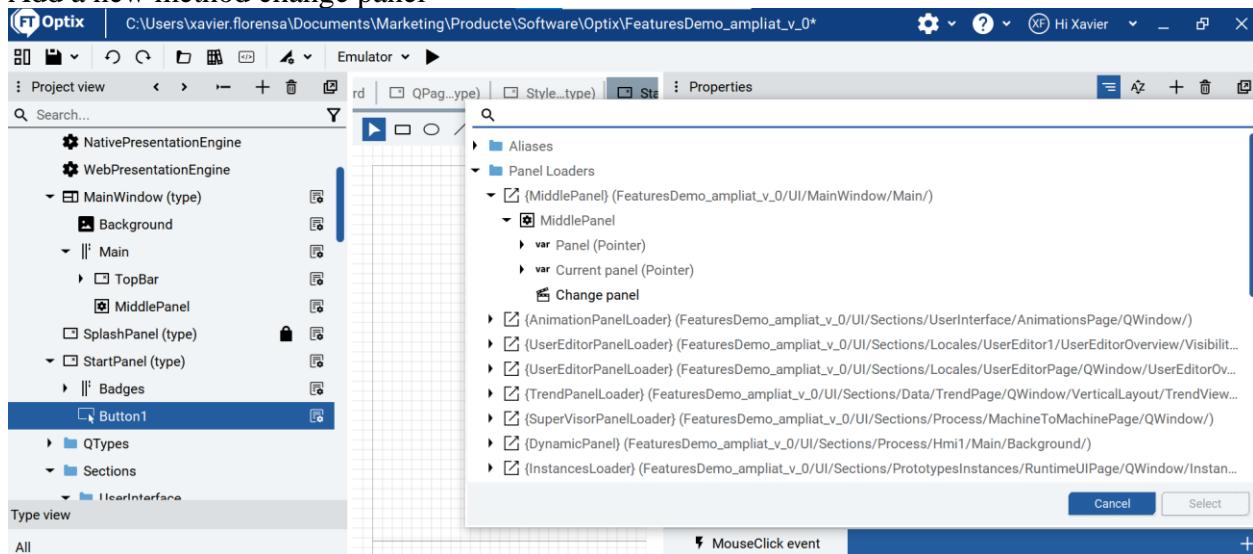


Add a button and change its text property

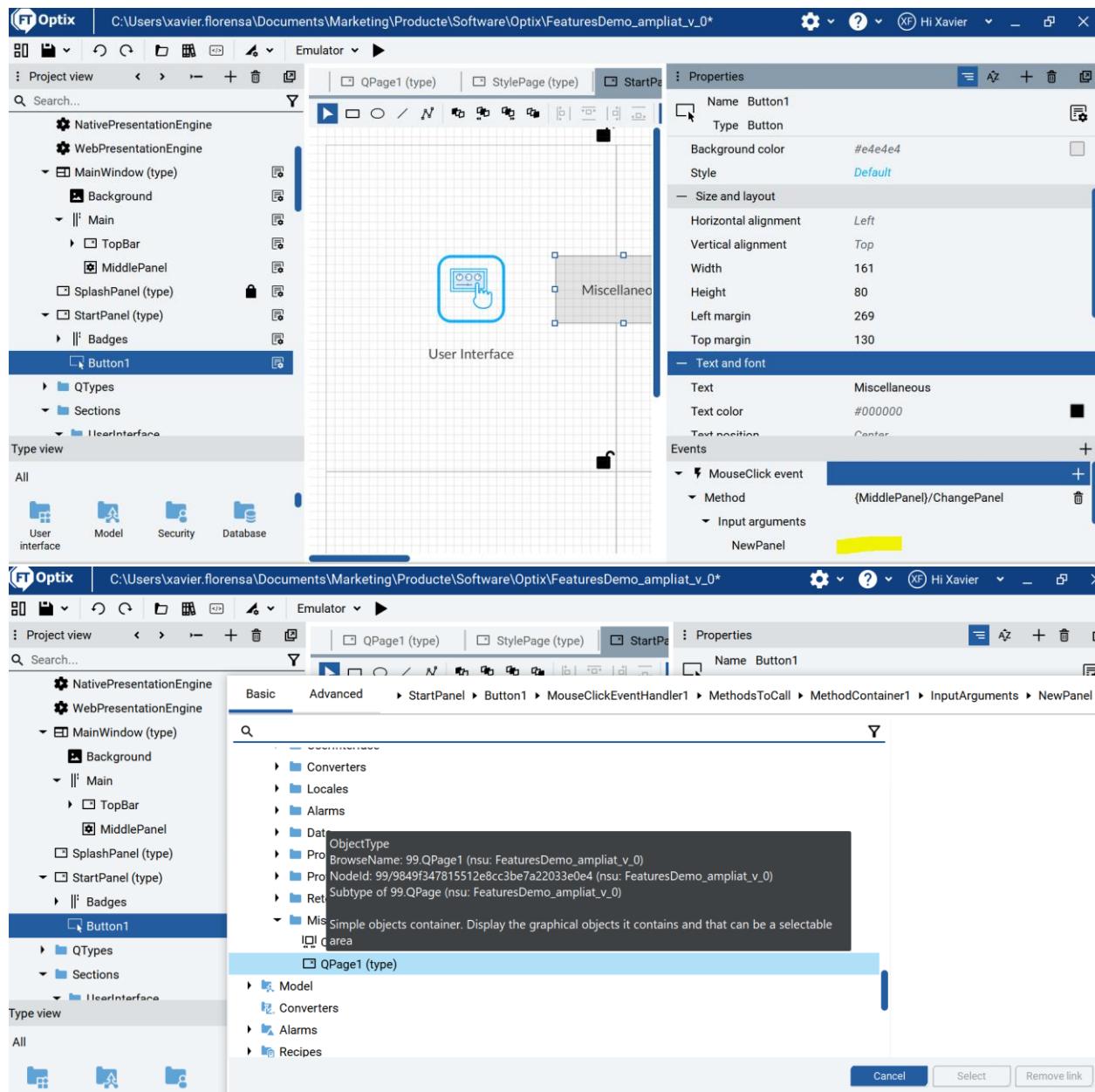


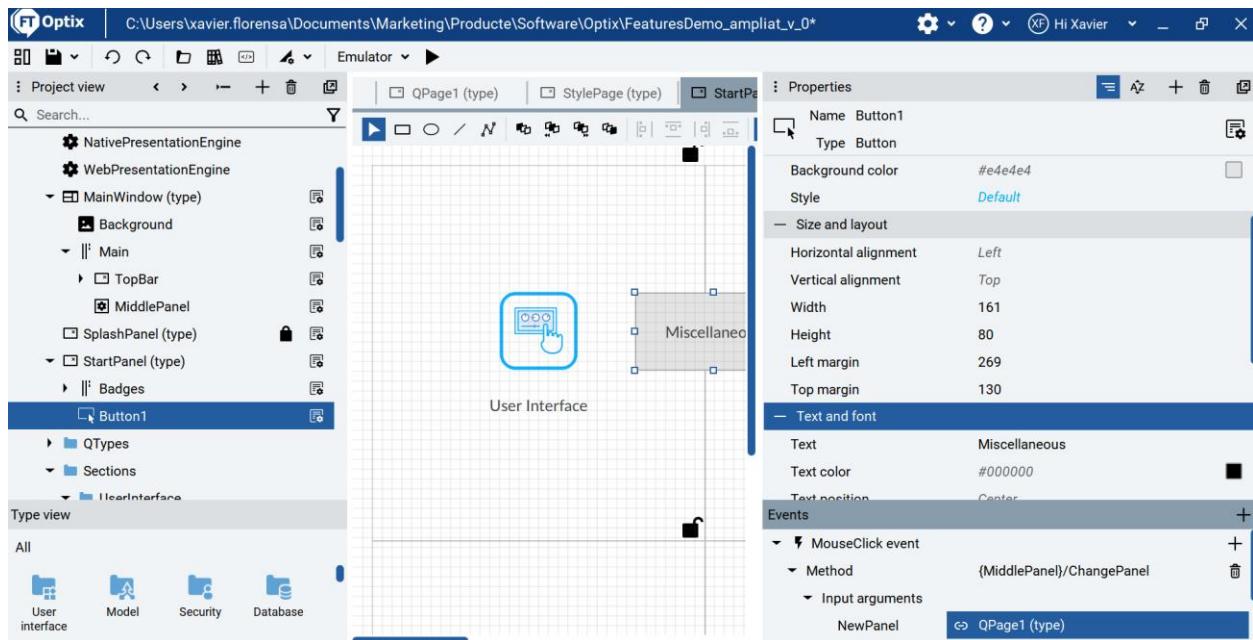
Click on the button and add a new click event

Add a new method change panel

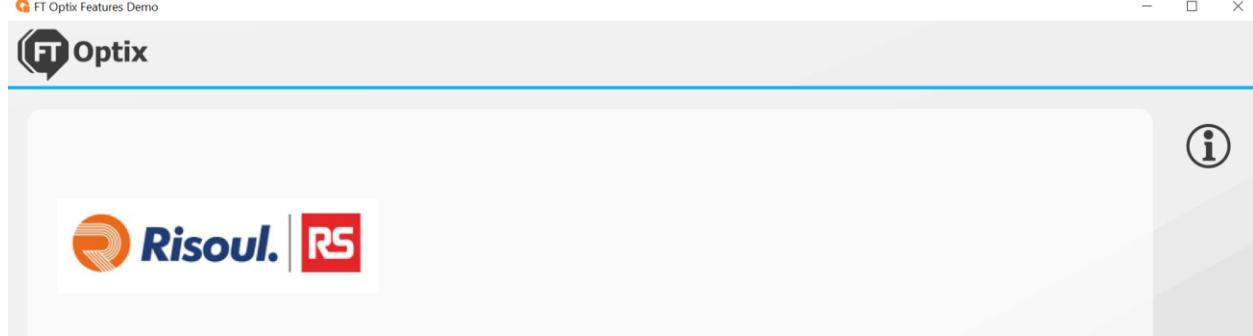
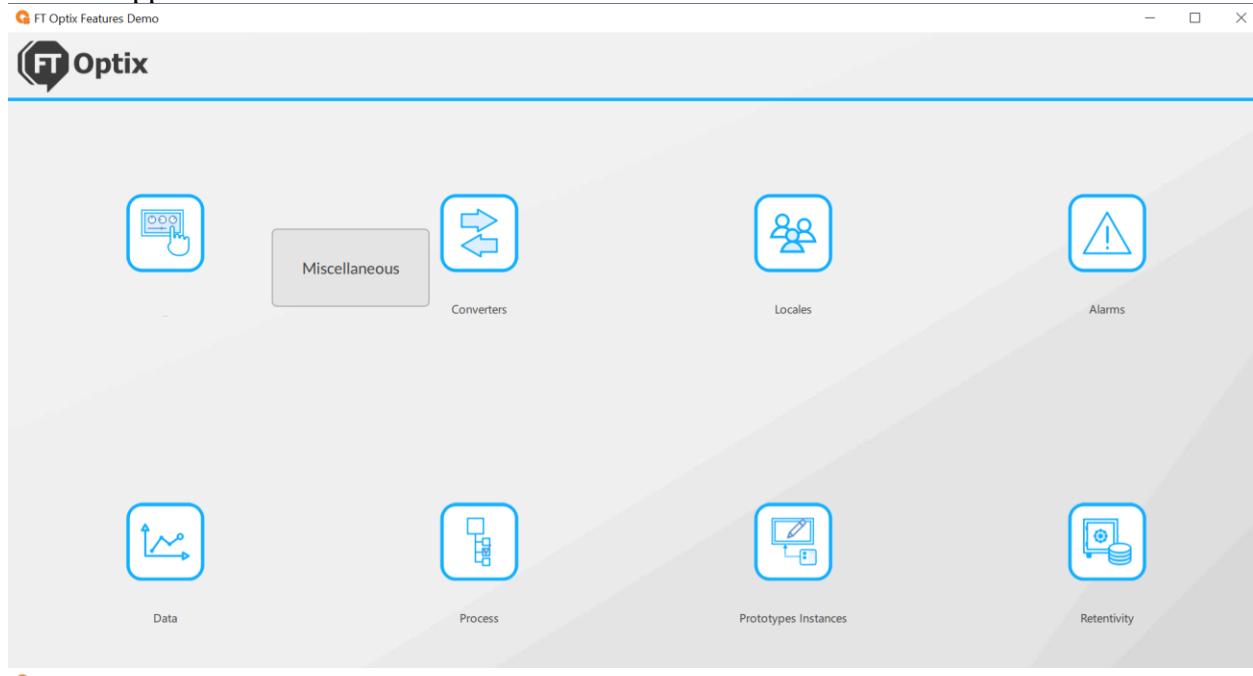


Add the new panel input argument





## Test the application



This is working,

Xavier Florensa

Automation Specialist

Risoul Ibérica SL

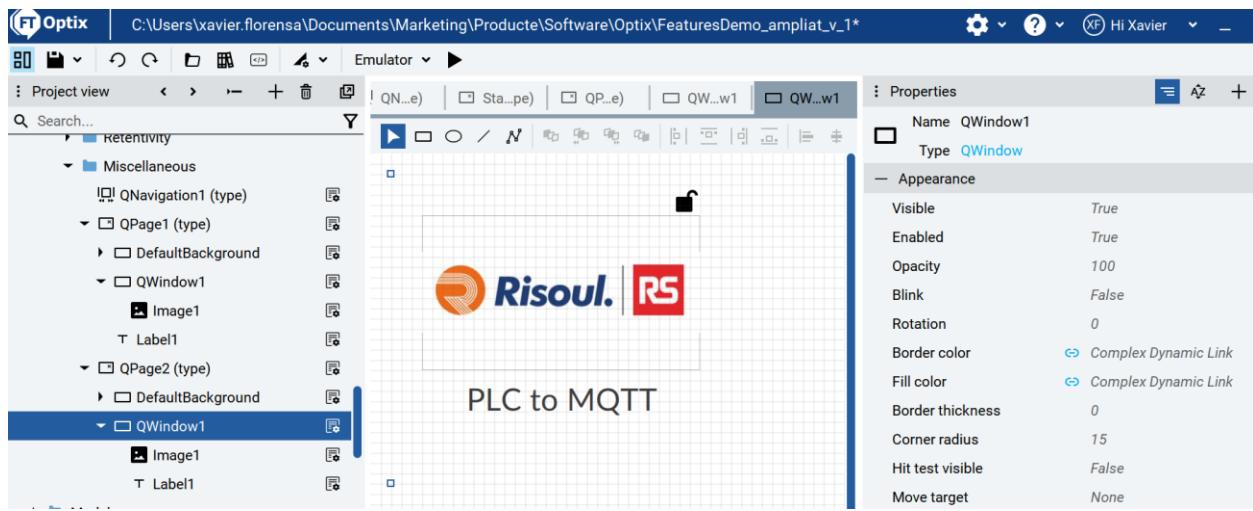
But we would prefer to have a panel loader

The screenshot shows the FT Optix software interface. At the top, there is a navigation bar with tabs: Style, Relative Position, Scroll View, Accordion, Animations, Advanced SVG, Video, PDF, and Web Browser. Below the navigation bar, a message says "We did a mistake with the input parameter of the button pressed". The main workspace displays a user interface diagram with a central panel labeled "Miscellaneous". To the left is a project tree showing a hierarchy of components like NativePresentationEngine, WebPresentationEngine, MainWindow, StartPanel, and various panels (Panel1, Panel2, Panel3, Panel4). On the right is a properties panel for a selected "Button1" component. The properties panel includes sections for Appearance, Size and layout, and Events. In the Events section, under the MouseClick event, the Method is set to "{MiddlePanel}/ChangePanel" and the Input arguments are set to "NewPanel". A red oval highlights the "NewPanel" argument. The bottom of the properties panel shows a preview of the UI with a button icon.

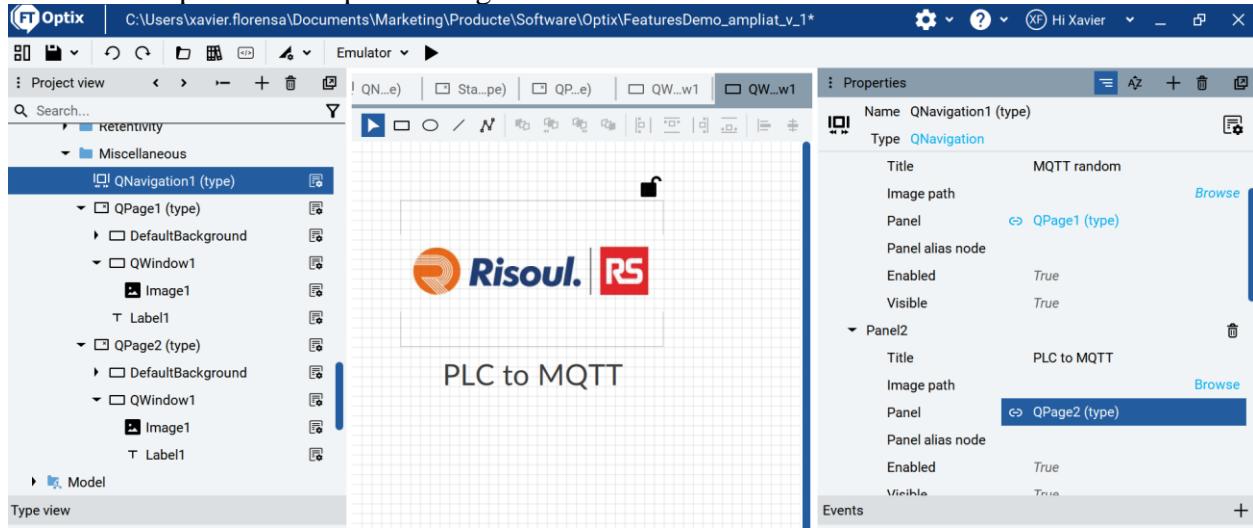
Now we have it

The screenshot shows a running application window. At the top, there is a header with the "FT Optix" logo and a "MQTT random" button. The main content area displays the "Risoul." logo, which consists of an orange stylized 'R' icon followed by the word "Risoul." in blue and a red 'RS' logo.

Let's add more Pannels  
Add a QPannel and a QWindow  
Populate the Qwindow



Add the new panel on the panel navigator



Test the application  
Now we have it

FT Optix Features Demo



MQTT random PLC to MQTT



## MQTT Random

FT Optix Features Demo



MQTT random PLC to MQTT



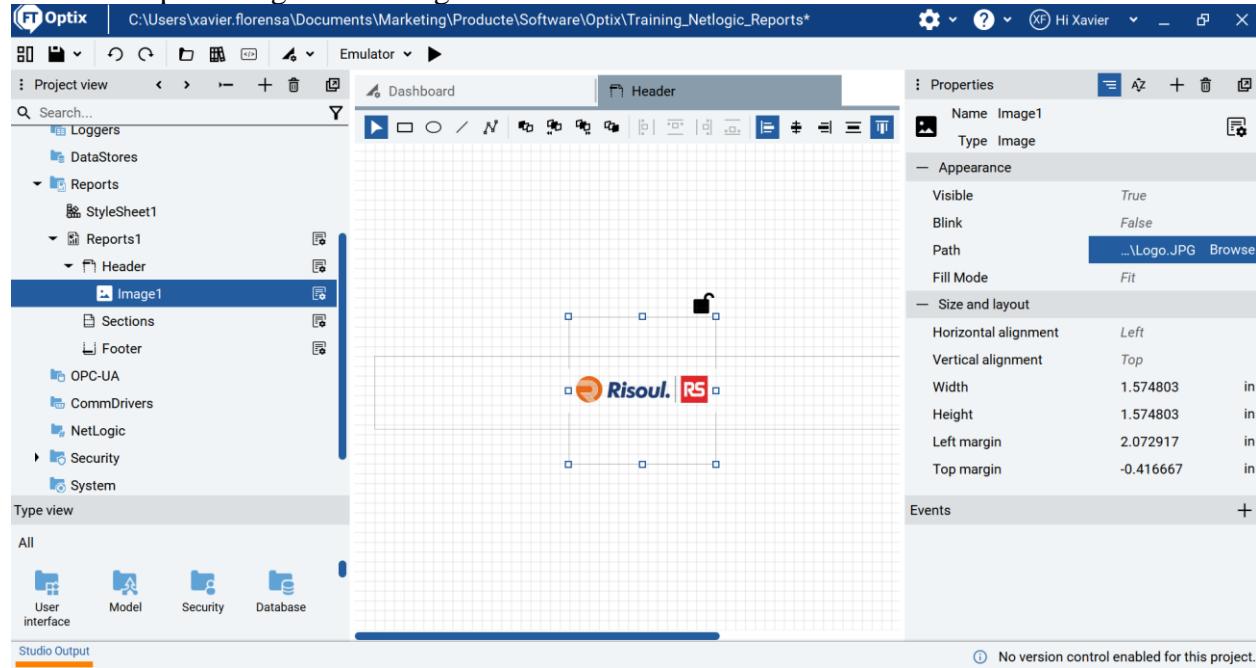
## PLC to MQTT

Let's add more pannels

We were not able to run MQTT client demo on this project.

## 49. Generating a pdf file from C#

### Create a Report design with a Logo



### Add a Netlogic Script

Populate with this code where you will insert the name of the file to be generated

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Rentativity;
using FTOptix.Report;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{

    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        String[] Parametros = { "%APPLICATIONDIR%/EtiquetaGranulacion.pdf", "" };
        Project.Current.GetObject("Reports/Reports1").ExecuteMethod("GeneratePdf"
, Parametros);
```

```

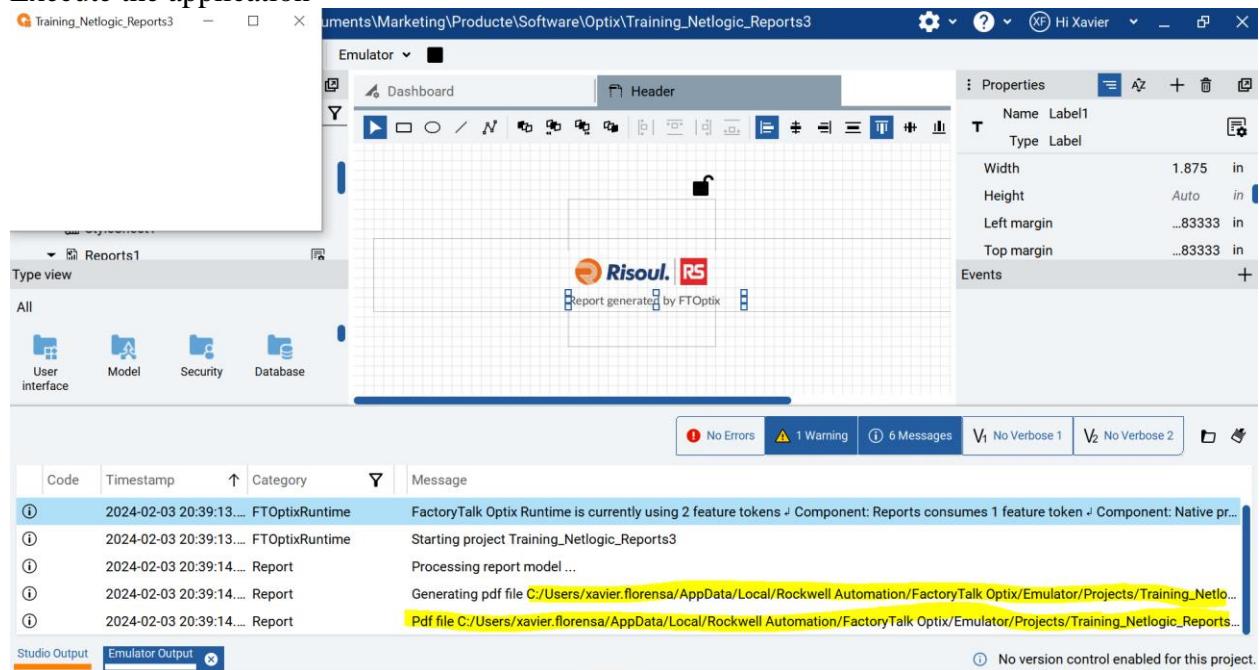
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }

}

```

## Execute the application



Look on the default application directory,

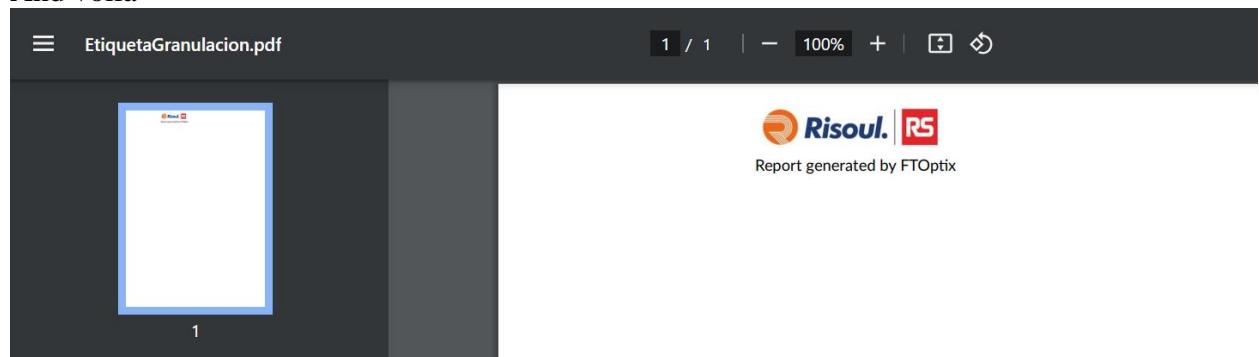
Be aware that this is not the directory where you store the application.

You have to look here with the right application name on the yellow market area

C:\Users\xavier.florensa\AppData\Local\Rockwell Automation\FactoryTalk

Optix\Emulator\Projects\Training\_Netlogic\_Reports3\ApplicationFiles

And voilà



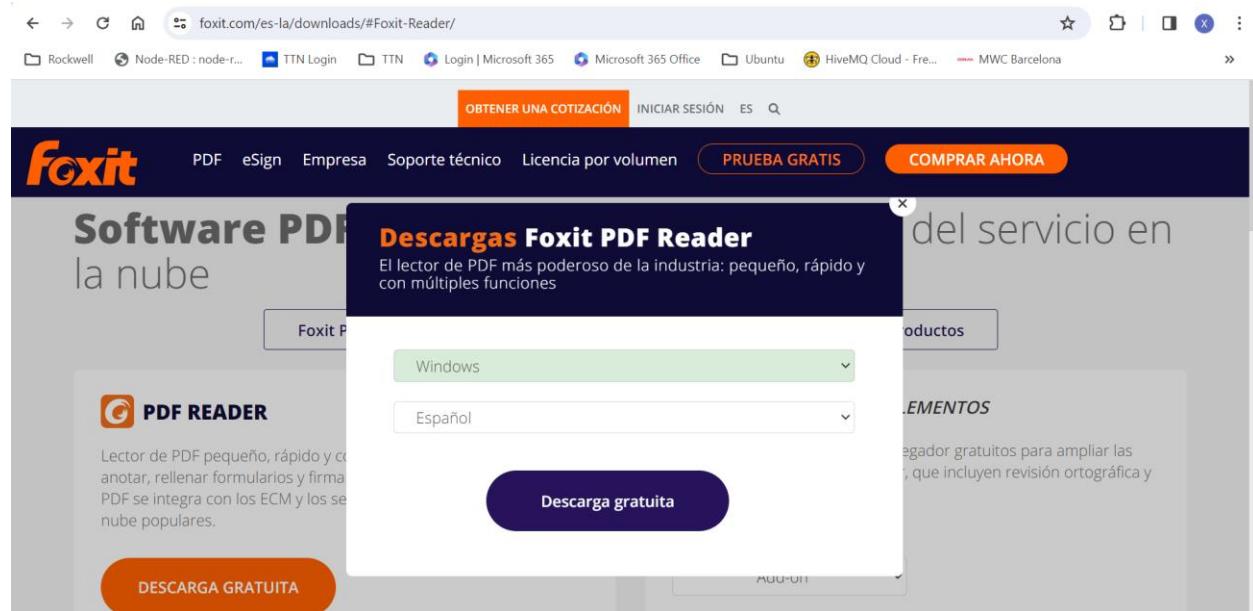
#### 49.1. Using Foxit Pdf

Which advantages do we have with Foxit Pdf,  
Pending to be cleared

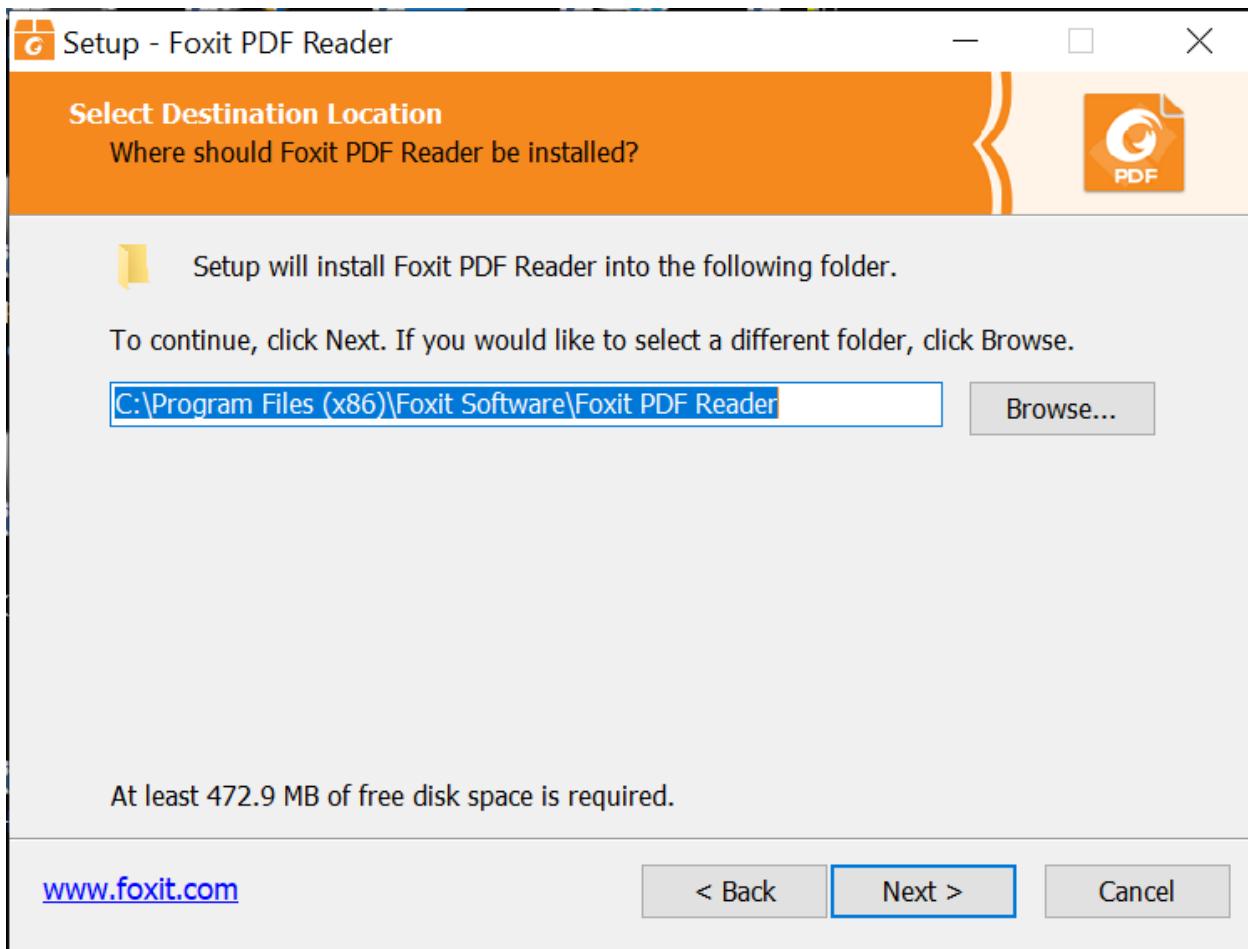
Let's say that you will have a File explorer to place your pdf file, but I was not able to generate any populated pdf on that location.

Download this pdf editor

<https://www.foxit.com/es-la/downloads/#Foxit-Reader/>

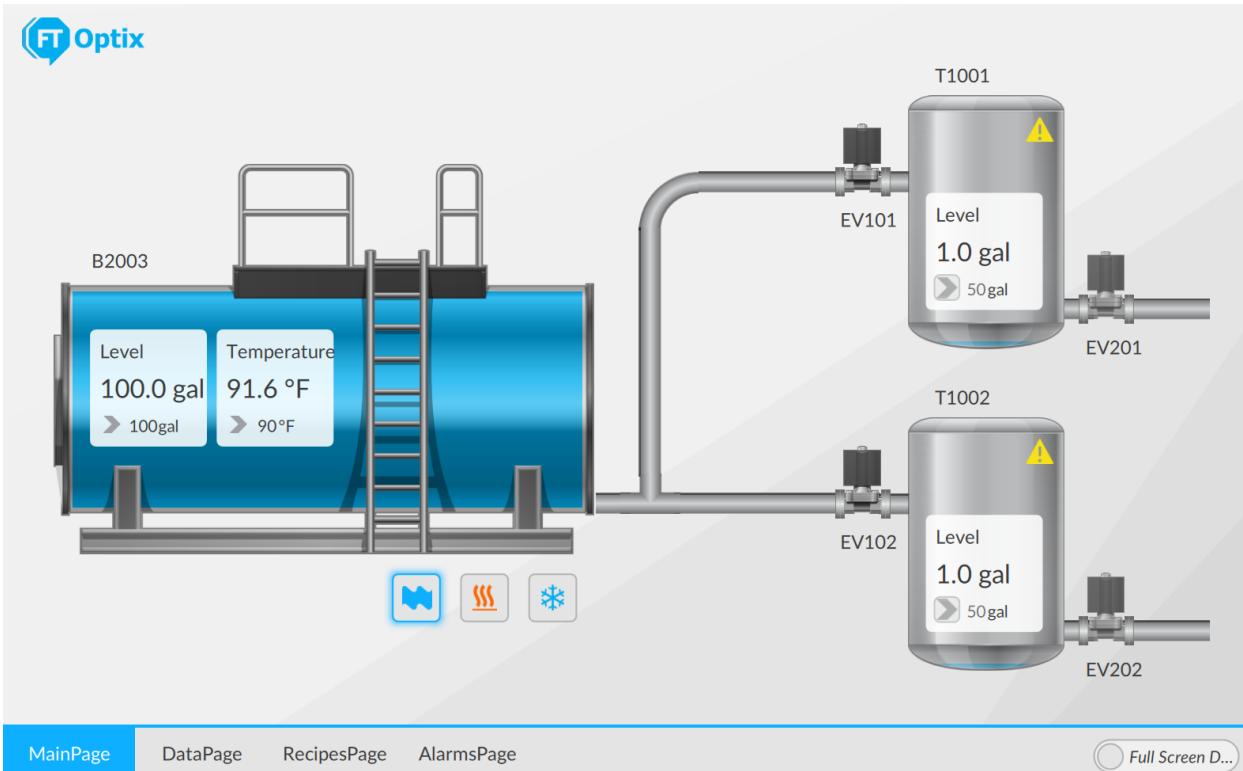


Use this default location



## 50. Using BoilerDemo

Probably you want to show this demo to your customers



But apart from the Optix project you will need the following:

- A Studio5000 \*.ACD file in order to execute the Tank1 and Tank2 on the PLC
- A OPC UA server example with predefined working variables on Boiler side.

Install your open source OPC UA server

<https://www.unified-automation.com/downloads/opc-ua-servers/opc-ua-ansi-c-demo-server.html>



Let's start our server with this icon



## UaAnsICServer

Ua AnsIC Server

```

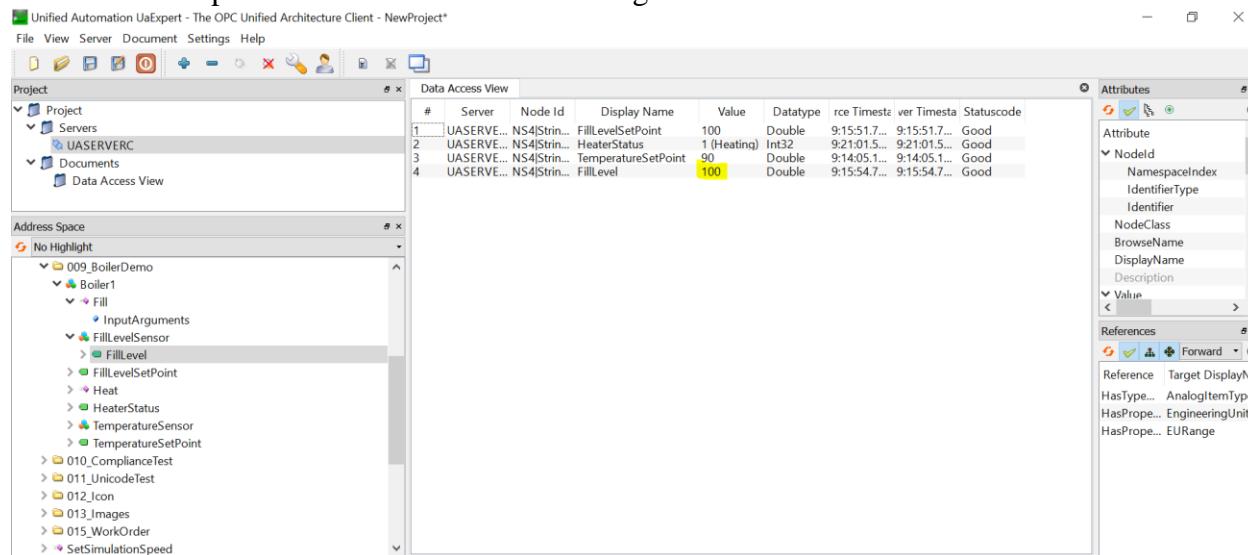
UA Server: Initializing Stack...
20:27:20.414Z|E|45B0* UA Server: Building Provider List...
20:27:20.414Z|E|45B0* UA Server: Loading Provider Modules...
20:27:20.415Z|W|45B0* Initialize Server Provider ...
20:27:20.420Z|W|45B0* Server Provider initialized!
20:27:20.420Z|W|45B0*    2284 Nodes created
20:27:20.421Z|W|45B0*    NS1:
20:27:20.421Z|W|45B0*    31 static nodes created
20:27:20.421Z|W|45B0*    66 static references created
20:27:20.421Z|W|45B0*    3 static methods created
20:27:20.426Z|W|45B0* Initialize Demo Provider ...
20:27:20.433Z|W|45B0* Demo Provider initialized!
20:27:20.433Z|W|45B0*    2566 Nodes created
20:27:20.433Z|W|45B0*    NS4:
20:27:20.433Z|W|45B0*    569 static nodes created
20:27:20.433Z|W|45B0*    1199 static references created
20:27:20.434Z|W|45B0*    23 static methods created
20:27:20.434Z|W|45B0* Configuration warning: SecurityPolicy 'http://opcfoundation.org/UA/SecurityPolicy#None' is enabled
, this allows clients to connect without security and certificate validation
20:27:20.441Z|E|45B0*
20:27:20.444Z|E|45B0* #####
20:27:20.444Z|E|45B0* # Server started! Press x to stop; r to restart the server!
20:27:20.445Z|E|45B0* #####
20:27:20.445Z|E|45B0* Endpoint URL 0: opc.tcp://CATPROLP1003-23.soporte.com:48020
20:27:20.445Z|E|45B0* Server started at 2024-02-08T20:27:20.445Z

```

Let's take a look at the OPC UA server used.

Let's take a look from a OPC UA client view

Fill level corresponds to the real time Boiler widget will value



Unified Automation UaExpert - The OPC Unified Architecture Client - NewProject\*

File View Server Document Settings Help

**Data Access View**

#	Server	Node Id	Display Name	Value	Datatype	rce Timest...	ver Timesta...	Statuscode
1	UASERVER...	NS4 String	FillLevelSetPoint	100	Double	9:15:51.7...	9:15:51.7...	Good
2	UASERVER...	NS4 String	HeaterStatus	1 (Heating)	Int32	9:21:01.5...	9:21:01.5...	Good
3	UASERVER...	NS4 String	TemperatureSetPoint	90	Double	9:14:05.1...	9:14:05.1...	Good
4	UASERVER...	NS4 String	FillLevel	100	Double	9:15:54.7...	9:15:54.7...	Good

**Address Space**

- No Highlight
  - 009\_BoilerDemo
    - Boiler1
      - Fill
        - InputArguments
        - Fill.levelSensor
          - Fill.level
        - FillLevelSetPoint
        - Heat
        - HeaterStatus
        - TemperatureSensor
        - TemperatureSetPoint

**Attributes**

Attribute

Nodeld
 

- NamespaceIndex
- IdentifierType
- Identifier
- NodeClass
- BrowseName
- DisplayName
- Description

Value

References

Forward

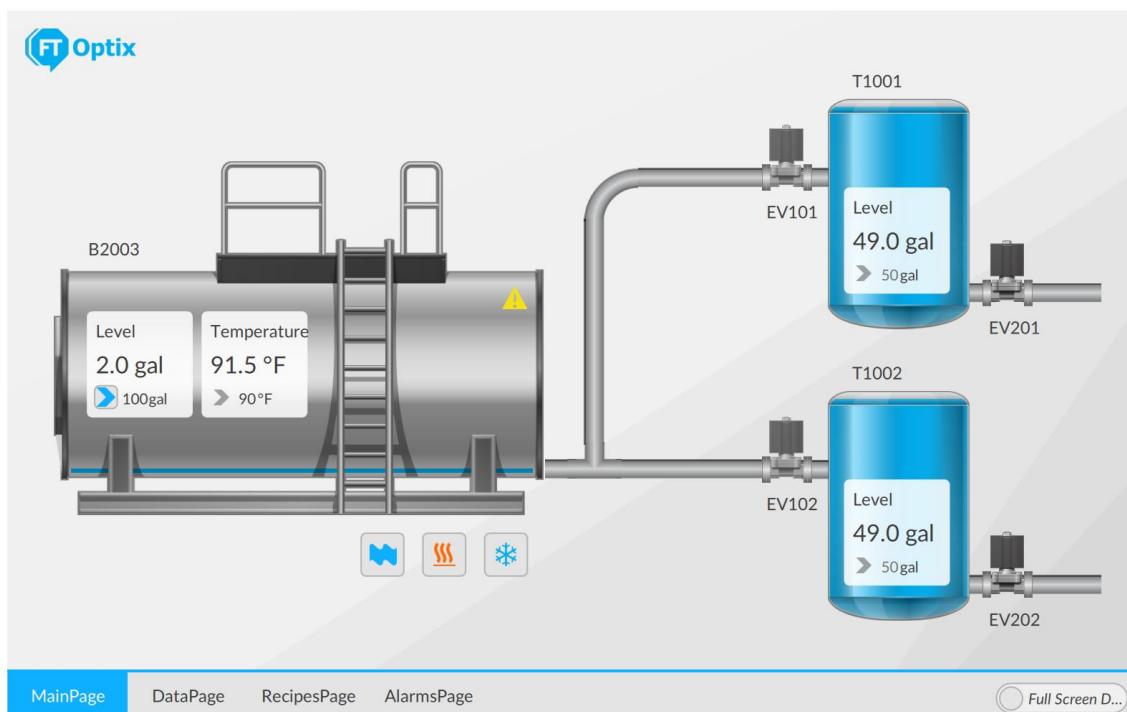
Reference

Target DisplayName

HasType... AnalogItemType

HasPrope... EngineeringUnits

HasPrope... EURange



Unified Automation UaExpert - The OPC Unified Architecture Client - NewProject\*

File View Server Document Settings Help

**Data Access View**

#	Server	Node Id	Display Name	Value	Datatype	rcr Timesta...	ver Timesta...	Statuscode
1	UASERVER...	NS4[String]	FillLevelSetPoint	2	Double	9:22:21.5...	9:22:21.5...	Good
2	UASERVER...	NS4[Int32]	HeaterStatus	1 (Heating)	Int32	9:23:01.9...	9:23:01.9...	Good
3	UASERVER...	NS4[Double]	TemperatureSetPoint	90	Double	9:14:05.1...	9:14:05.1...	Good
4	UASERVER...	NS4[Double]	FillLevel	2	Double	9:22:24.7...	9:22:24.7...	Good

**Address Space**

- No Highlight
- 009\_BoilerDemo
  - Boiler1
    - Fill
      - InputArguments
      - Fill.levelSensor
        - Fill.level
      - Fill.levelSetPoint
      - Heat
      - HeaterStatus
      - TemperatureSensor
      - TemperatureSetPoint
- 010\_ComplianceTest
- 011\_UnicodeTest
- 012\_Icon
- 013\_Images
- 015\_WorkOrder
- setSimulationSpeed

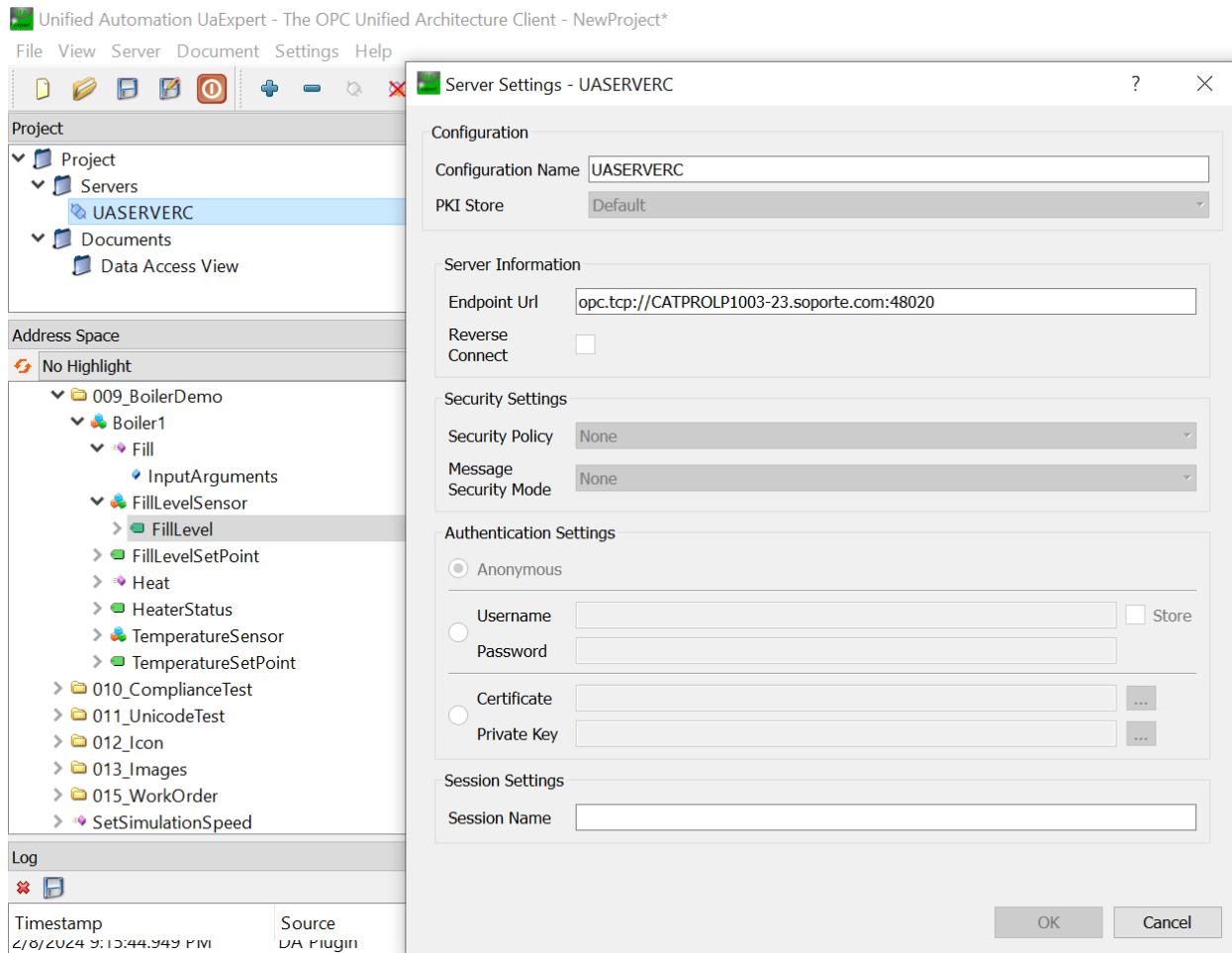
**Attributes**

**References**

**Log**

Timestamp	Source	Server	Message
2/8/2024 9:15:44.949 PM	DA Plugin	UASERVE...	Item [NS4[String]Demo.BoilerDemo.Boiler1.FillLevelSensor.FillLevel] : SamplingInterval=250, QueueSize=1, DiscardOldest=1, ClientHandle=>
2/8/2024 9:15:44.949 PM	DA Plugin	UASERVE...	CreateMonitoredItems succeeded [ret = Good]
2/8/2024 9:15:44.949 PM	DA Plugin	UASERVE...	Item [NS4[String]Demo.BoilerDemo.Boiler1.FillLevelSensor.FillLevel] succeeded : RevisedSamplingInterval=250, RevisedQueueSize=1, MonitoredItemId=4 [ret = Good]

Let's take a look at the Server configuration



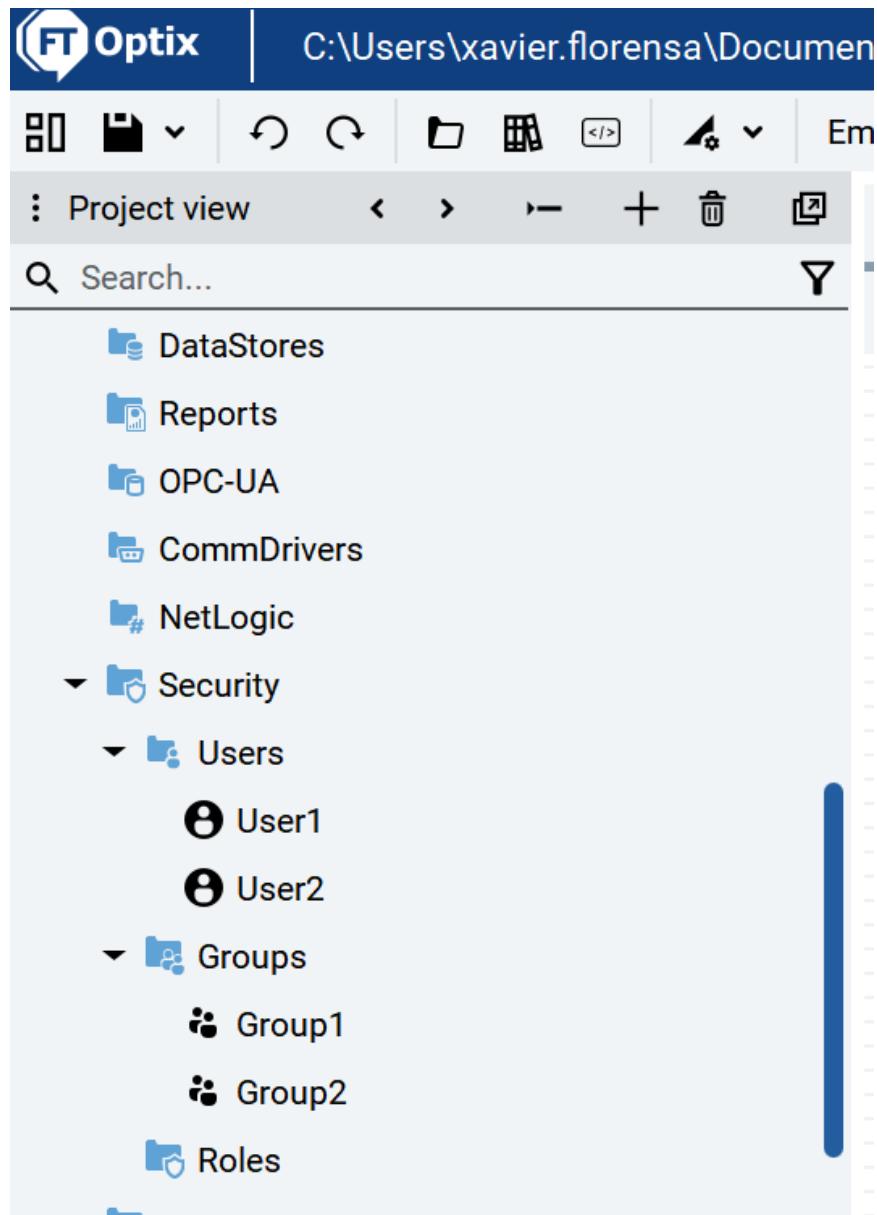
That's all

## 51. Users in Optix

### 51.1. Creating users and Groups manually

Create users on Security Folder

Create Groups on Security Folder



## 51.2. Create a Login form

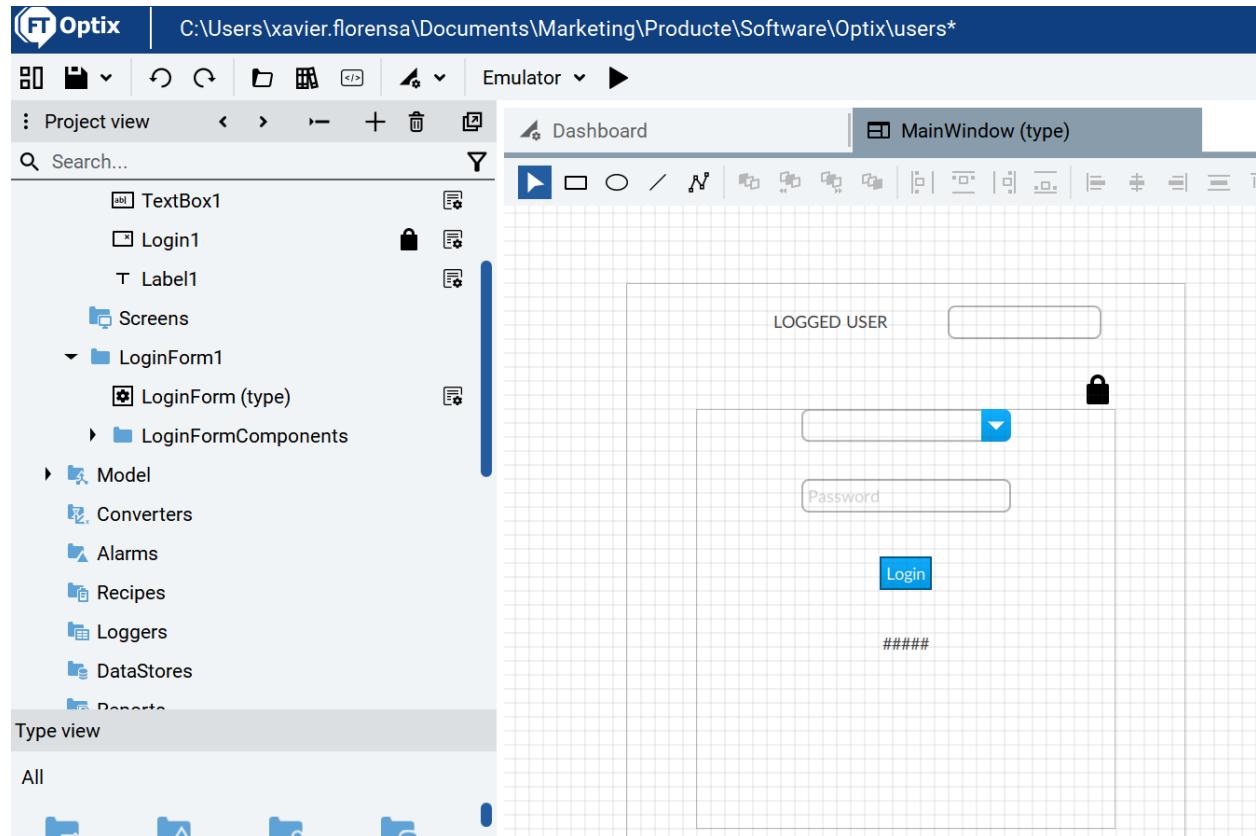
# Create a login form

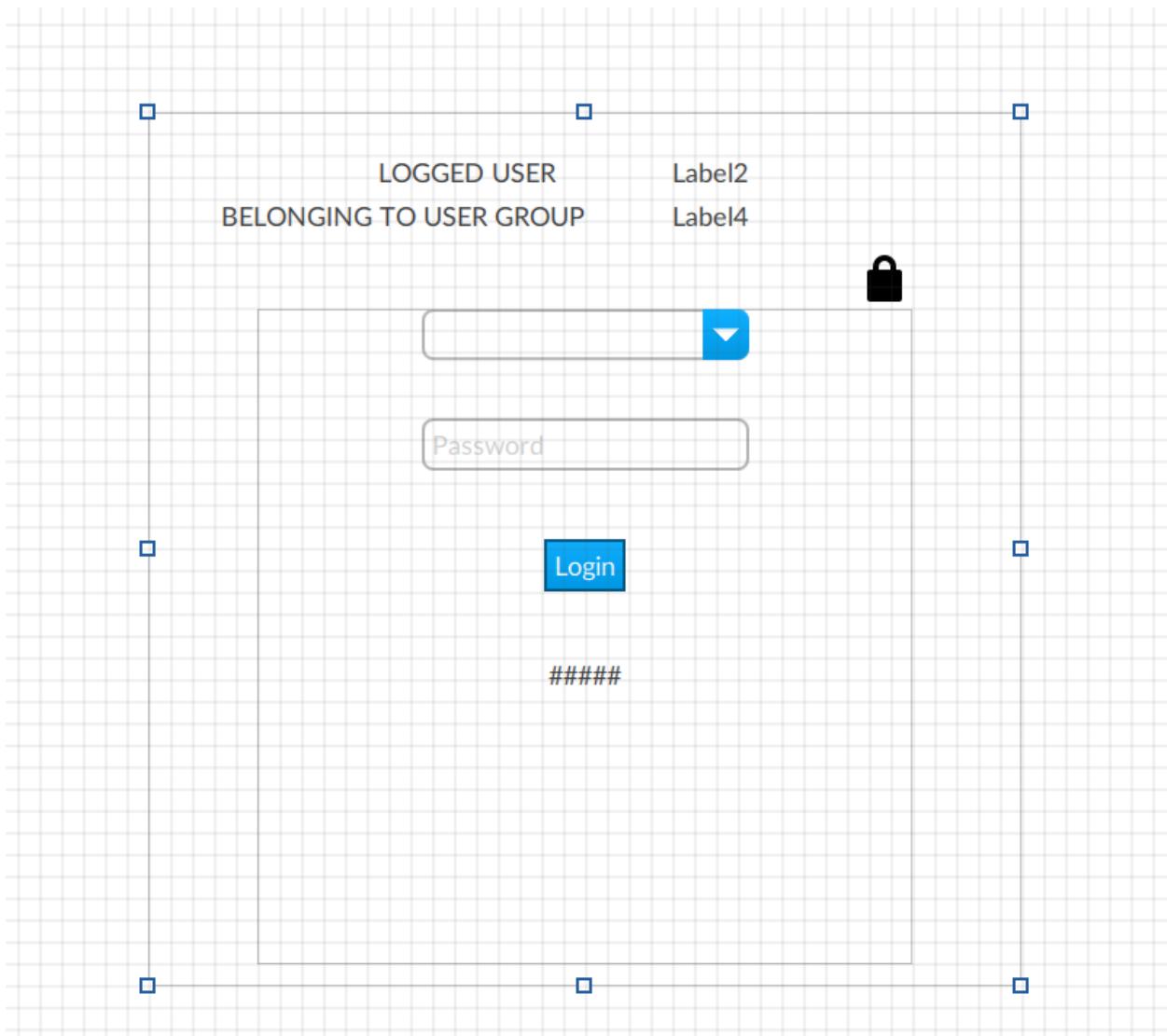
Create an interface that allows for logging in.

1. Add a preconfigured widget (type):
  - a. In the toolbar, select **Template Libraries**.
  - b. In **Libraries**, search for **Login form**.
  - c. Under **Components**, drag **Login form** onto UI in **Project view**.
2. In **Project view**, right-click **MainWindow (type)** and select **New > LoginForm1 > LoginFormComponents > Login**.
3. In **Properties**, create a dynamic link between **Users** and **ProjectName > Security > Users**.

For more information about dynamic links, see [Create dynamic links](#).

<https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/developing-solutions/application-examples/translations-tutorial/develop-a-multilingual-login-form/create-a-login-form.html>





### 51.3. Writing the logged user to the PLC

---

The screenshot shows a software window titled "users". At the top right are minimize, maximize, and close buttons. Below the title is a section labeled "LOGGED USER BELONGING TO USER GROUP" with a dropdown menu showing "User1". To the right of this section are two labels: "Anonymous" and "Label4". A large "UPDATE" button is positioned to the right of "Label4". Below the dropdown is a "Password" input field. At the bottom center is a blue "Login" button.

Login User1



LOGGED USER  
BELONGING TO USER GROUP

User1  
Group1

UPDATE

User1

Password

Login

Login user 2



LOGGED USER  
BELONGING TO USER GROUP

User2  
Group2

UPDATE

User2

Password

Login

It works!

This is the code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.WebUI;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using FTOptix.UI;
using FTOptix.Core;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Mostrar el usuario logado
        var usuario = Session.User;
```

```

var user = Session.User.BrowseName;
var mylabel = Project.Current.Get<Label>("UI/MainWindow/Label2");
mylabel.Text = user.ToString();
//Mostra el grup del usuari
var userGroups =
usuario.Refs.GetObjects(FTOptix.Core.ReferenceTypes.HasGroup, false);
foreach (var group in userGroups)
{
    Log.Info(group.BrowseName);
    Project.Current.GetVariable("Model/Locales/UserGroups").Value =
group.BrowseName;
}

public override void Stop()
{
    // Insert code to be executed when the user-defined logic is stopped
}
[ExportMethod]

public void Update()
{
    Actualitza();
    Log.Info("Updating: ");
}

private void Actualitza()
{
    // Mostrar el usuari logat
    var usuario = Session.User;
    var user = Session.User.BrowseName;
    var mylabel = Project.Current.Get<Label>("UI/MainWindow/Label2");
    mylabel.Text = user.ToString();
    //Mostra el grup del usuari
    var userGroups =
usuario.Refs.GetObjects(FTOptix.Core.ReferenceTypes.HasGroup, false);
    foreach (var group in userGroups)
    {
        Log.Info(group.BrowseName);
        var mylabel_group =
Project.Current.Get<Label>("UI/MainWindow/Label4");
        mylabel_group.Text = group.BrowseName.ToString();
        Project.Current.GetVariable("Model/Locales/UserGroups").Value =
group.BrowseName;
    }
}

```

```
}
```

```
}
```

As you can see on this video

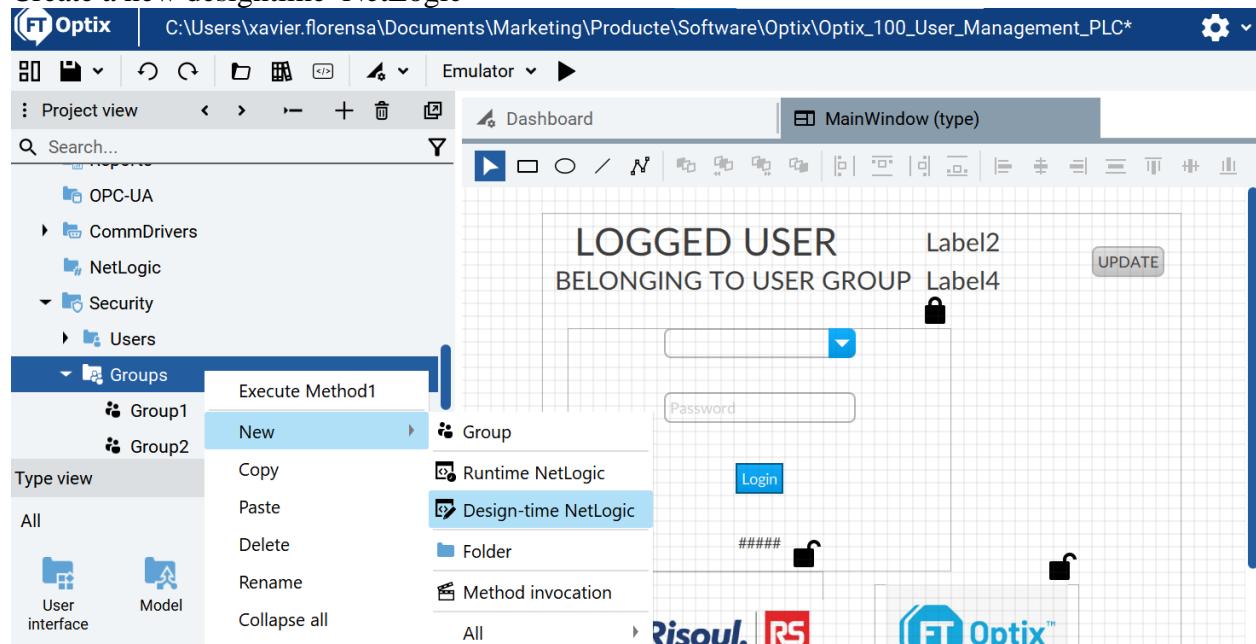
<https://www.youtube.com/watch?v=3D85iqPMuCo&t=9s>

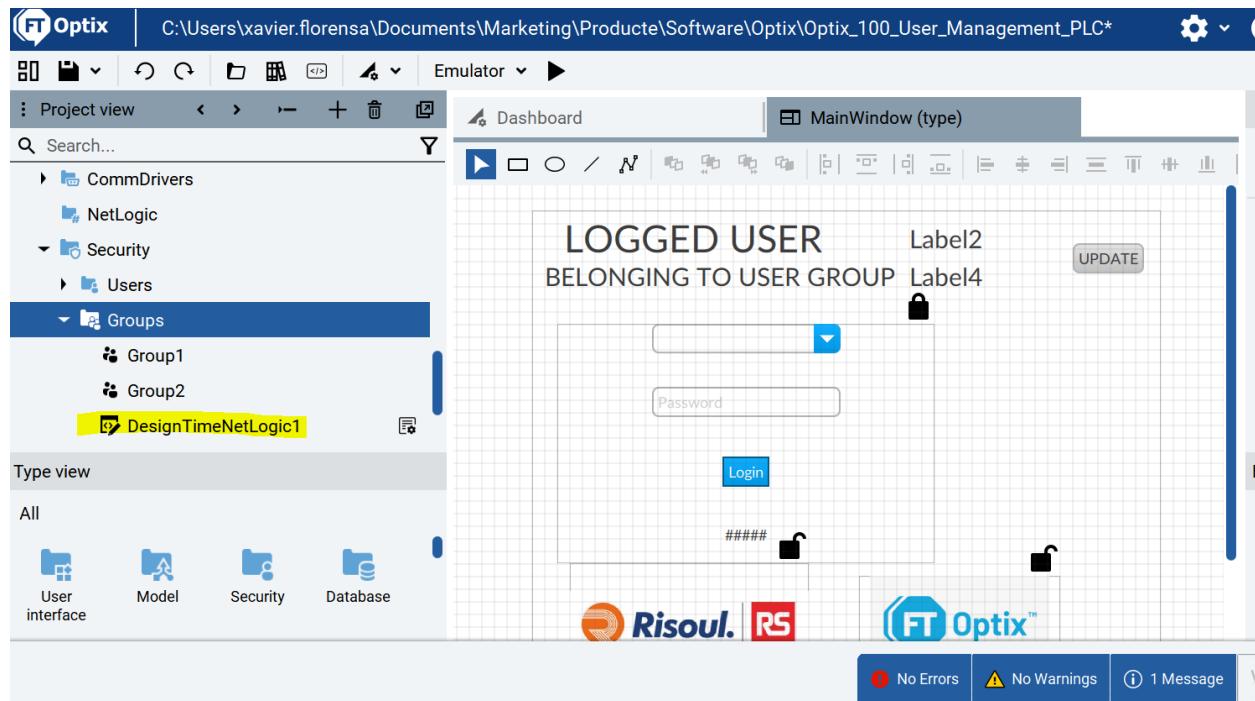
You can find the code here

[https://github.com/xavierflorensa/Optix\\_User\\_Management\\_PLC](https://github.com/xavierflorensa/Optix_User_Management_PLC)

#### 51.4. Creating Groups in design time

Create a new designtime NetLogic





Open the code

```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.UI;
using FTOptix.RAEtherNetIP;
using FTOptix.NativeUI;
using FTOptix.NetLogic;
using FTOptix.CoreBase;
using FTOptix.Retentivity;
using FTOptix.CommunicationDriver;
using FTOptix.Core;
#endregion

public class DesignTimeNetLogic1 : BaseNetLogic
{
    [ExportMethod]
    public void Method1()
    {
        // Insert code to be executed by the method
    }
}
```

Copy this code, as a simple way to create a new Group with gibe name: “Nou\_Grup”

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrject;
using FTOptix.UI;
using FTOptix.RAEtherNetIP;
using FTOptix.NativeUI;
using FTOptix.NetLogic;
using FTOptix.CoreBase;
using FTOptix.Retentivity;
using FTOptix.CommunicationDriver;
using FTOptix.Core;
#endregion

public class DesignTimeNetLogic1 : BaseNetLogic
{
    [ExportMethod]
    public void Method1()
    {
        // Insert code to be executed by the method
        Folder userFolder = Project.Current.Get<Folder>("Security/Groups");
        if (userFolder == null) {
            Log.Error("CreateNewGroup", "Cannot find Groups folder");
            return;
        }
        /*else if (string.IsNullOrEmpty(groupName)) {
            Log.Error("CreateNewGroup", "Cannot create group with empty name");
            return;
        } else if (userFolder.Get<Group>(groupName) != null) {
            Log.Error("CreateNewGroup", "Group already exists!");
            return;
        }*/
        // Create a new group
        var newGroup = InformationModel.Make<Group>("Nou_Grup");
        // Add the group to the folder
        userFolder.Add(newGroup);
    }
}
```

This is done using following RA recommendations:

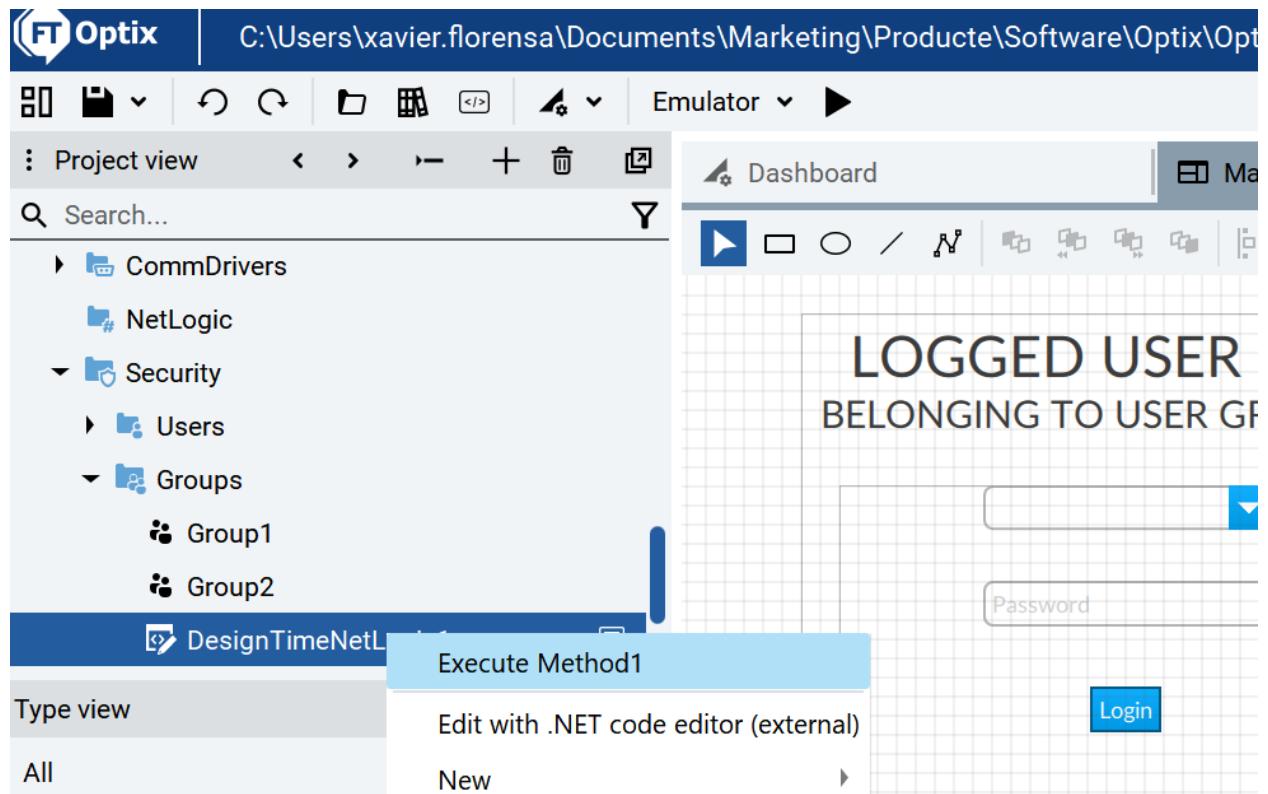
[https://github.com/FactoryTalk-Optix/NetLogic\\_CheatSheet/blob/main/pages/users-groups.md](https://github.com/FactoryTalk-Optix/NetLogic_CheatSheet/blob/main/pages/users-groups.md)

We are using only the inside part of the method

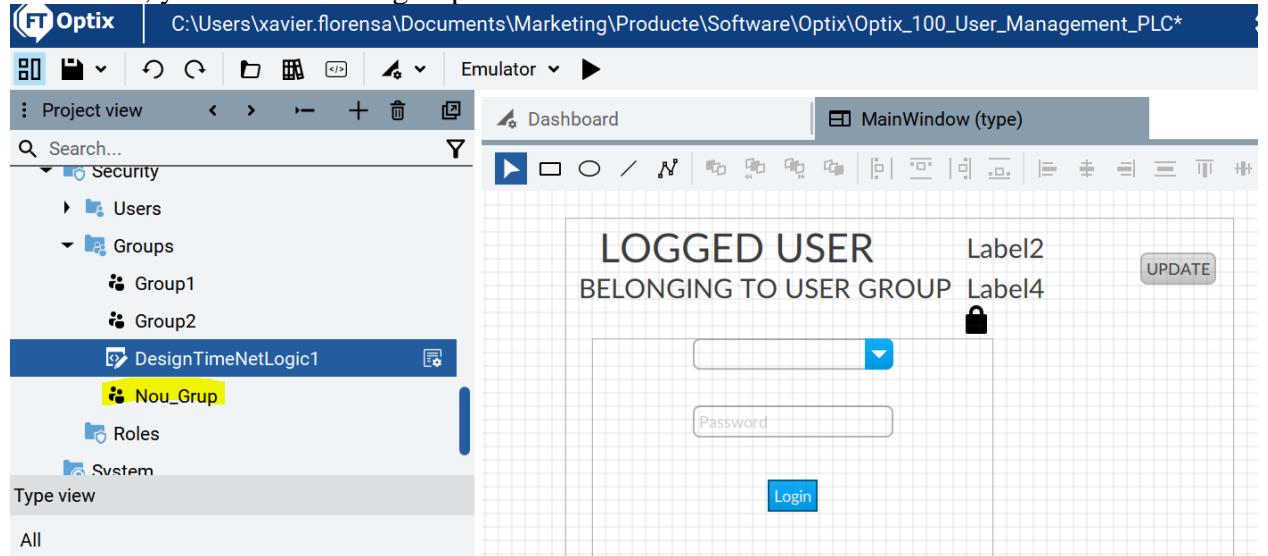
The screenshot shows a GitHub file viewer for the file `users-groups.md`. The left sidebar has a "Files" section with a dropdown set to "main" and a search bar. Below it is a tree view of files under "pages". The right panel has tabs for "Preview", "Code", and "Blame", with "Preview" selected. It shows the following content:

```
[ExportMethod]
public void CreateNewGroup(string groupName) {
    // Get to the base folder to create groups
    Folder userFolder = Project.Current.Get<Folder>("Security/Groups");
    if (userFolder == null) {
        Log.Error("CreateNewGroup", "Cannot find Groups folder");
        return;
    } else if (string.IsNullOrEmpty(groupName)) {
        Log.Error("CreateNewGroup", "Cannot create group with empty name");
        return;
    } else if (userFolder.Get<Group>(groupName) != null) {
        Log.Error("CreateNewGroup", "Group already exists!");
        return;
    }
    // Create a new group
    var newGroup = InformationModel.Make<Group>(groupName);
    // Add the group to the folder
    userFolder.Add(newGroup);
}
```

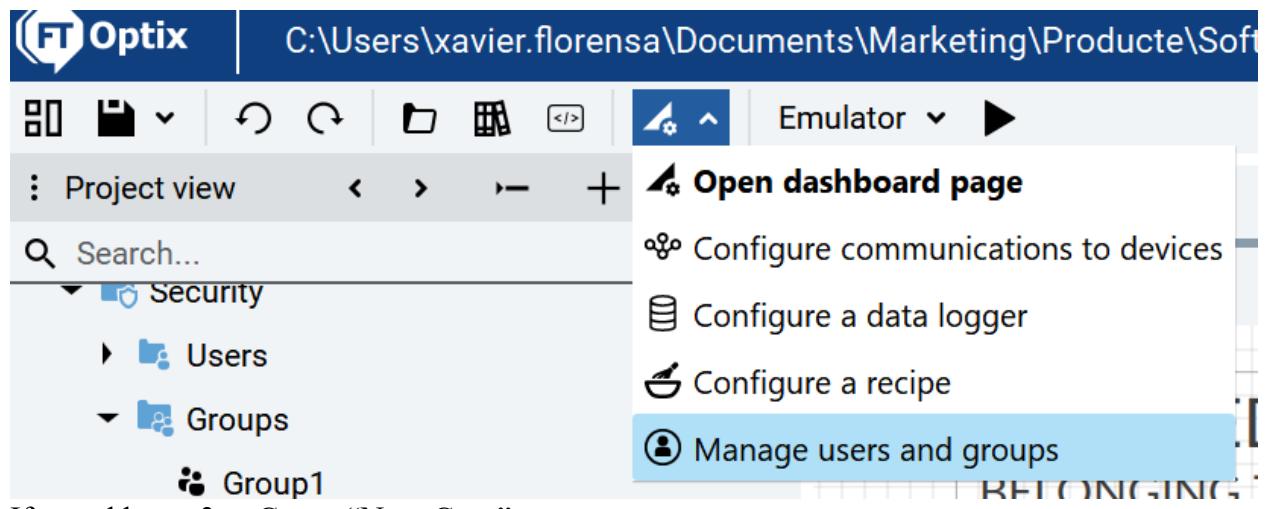
Now you are ready to execute the method this way in design time:



And Voilà, you have the new group created

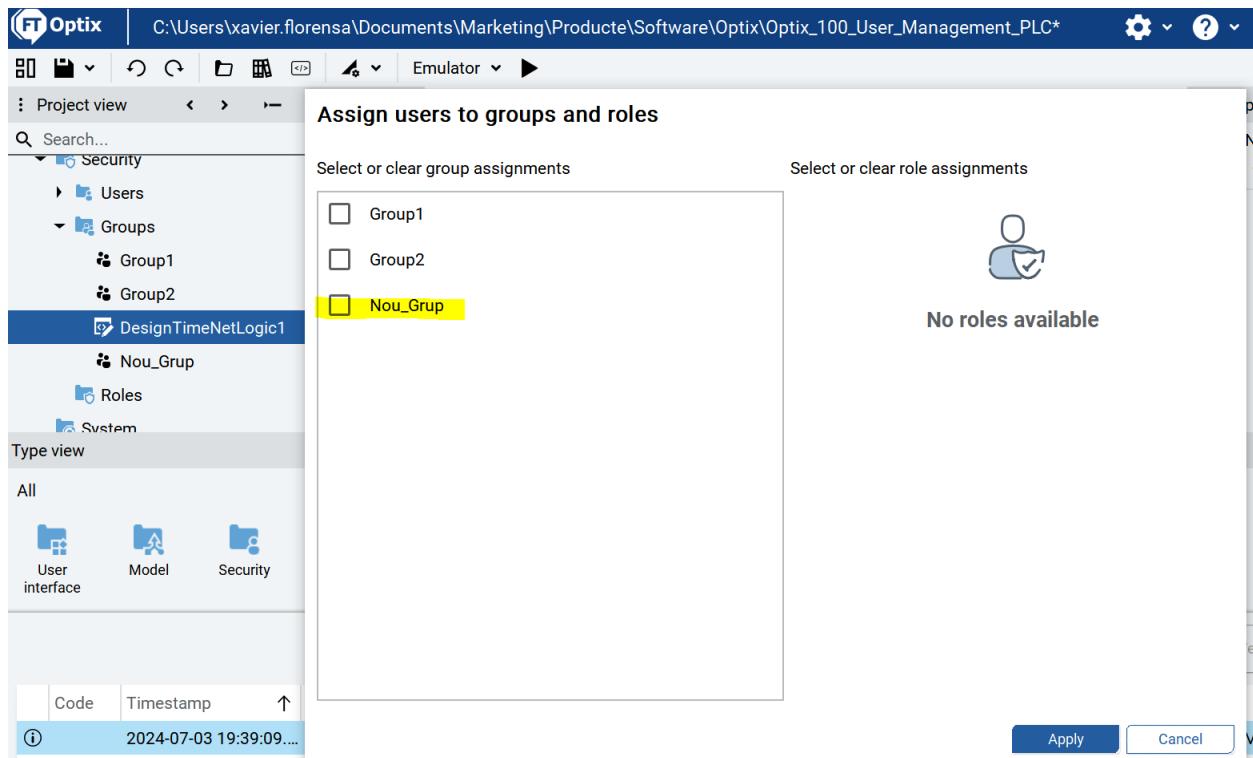


And you have this new group available to add users



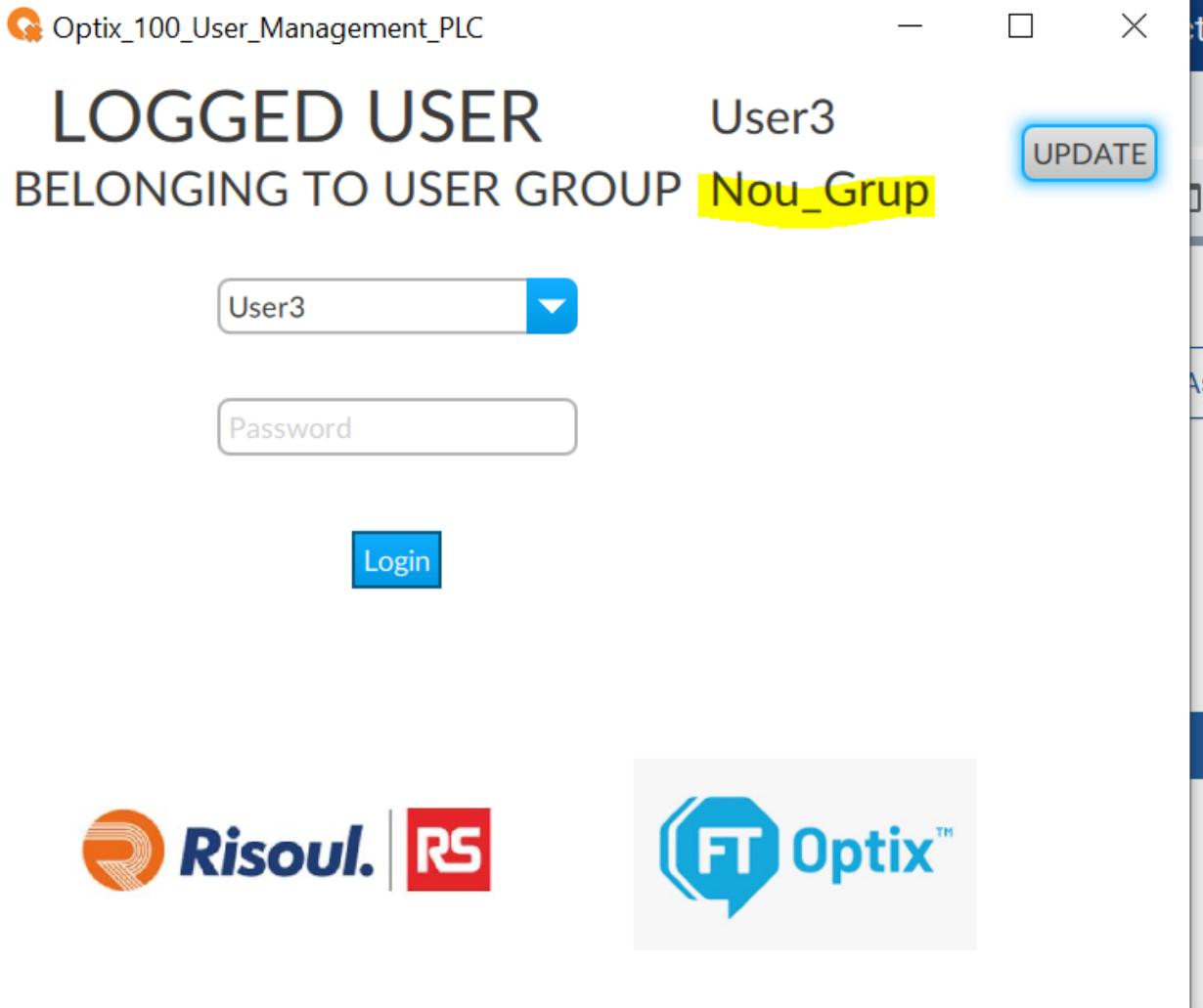
If we add user 3 to Group "Nou\_Grup"

The screenshot shows the Optix software interface with the title bar "C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\Optix\_10". The main window has a toolbar with icons for file operations and navigation. Below the toolbar is a "Project view" pane with a search bar. The left sidebar is titled "Security" and contains "Users", "Groups", "DesignTimeNetLogic1", "Nou\_Grup", "Roles", and "System" sections. Under "Groups", there are "Group1" and "Group2". A context menu is open on "User3", listing options: "Assign to groups and roles" (which is highlighted), "Rename", and "Delete". The background shows some blurred text.



If we execute the project, we will be able to log as user 3

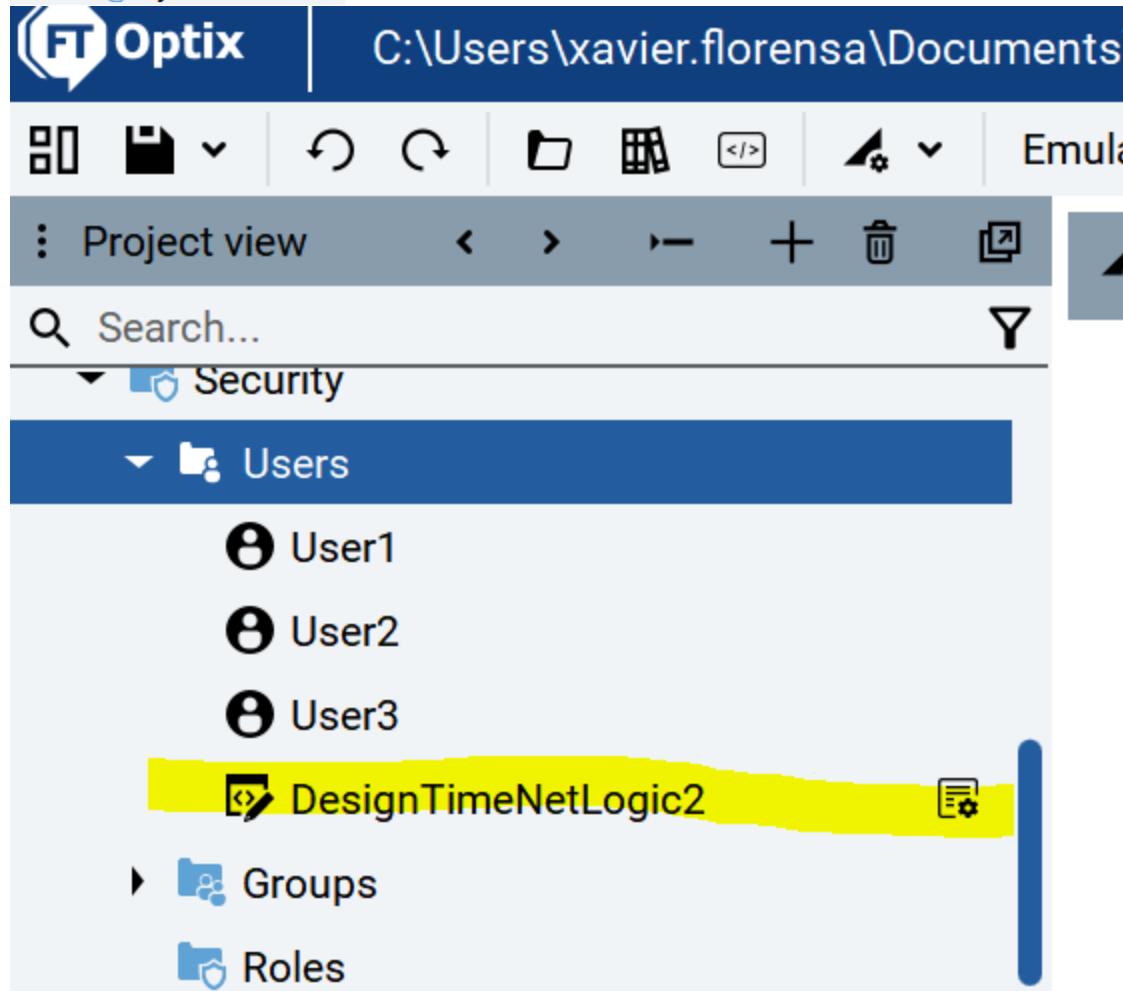
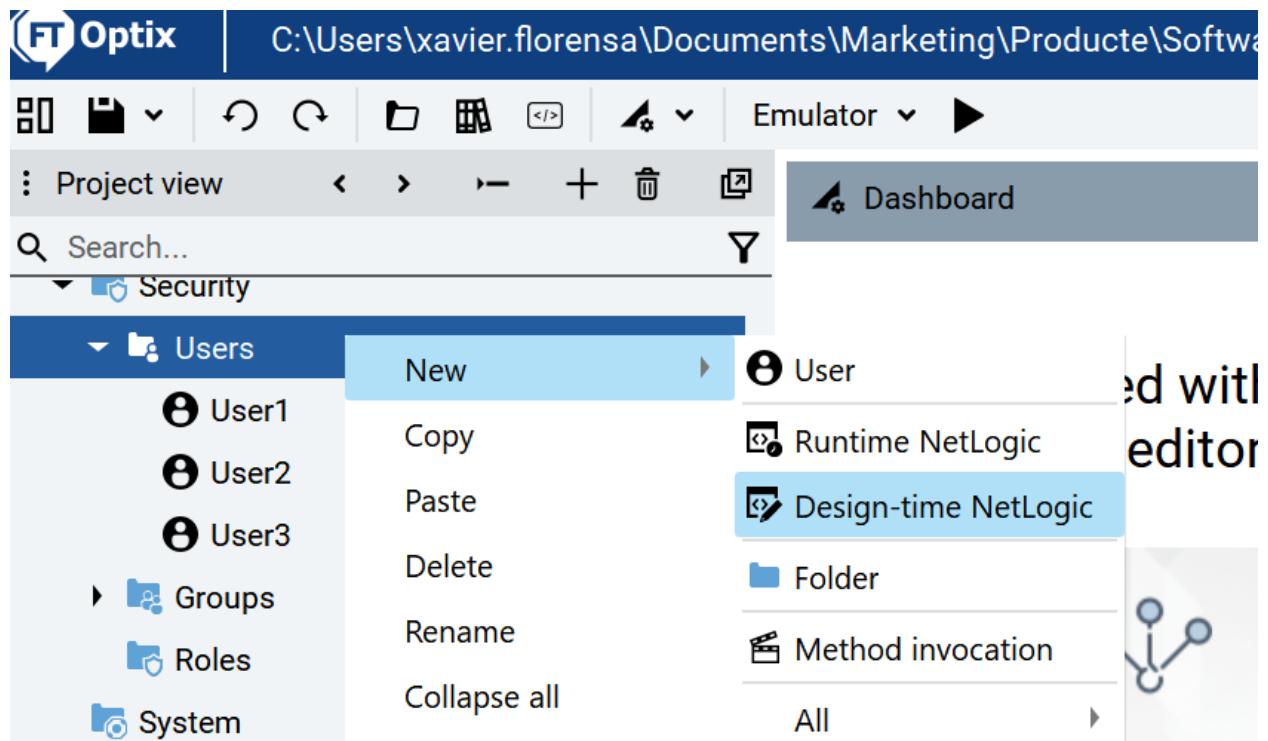
First log in and then click on UPDATE



### 51.5. Creating Users in Design time

Let's try with this code

Let's create a new Design Time Netlogic



Open the code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrject;
using FTOptix.UI;
using FTOptix.RAEtherNetIP;
using FTOptix.NativeUI;
using FTOptix.NetLogic;
using FTOptix.CoreBase;
using FTOptix.Retentivity;
using FTOptix.CommunicationDriver;
using FTOptix.Core;
#endregion

public class DesignTimeNetLogic2 : BaseNetLogic
{
    [ExportMethod]
    public void Method1()
    {
        // Insert code to be executed by the method
    }
}
```

Try with this code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FT Optix.HMIPrject;
using FTOptix.UI;
using FTOptix.RAEtherNetIP;
using FTOptix.NativeUI;
using FTOptix.NetLogic;
using FTOptix.CoreBase;
using FTOptix.Retentivity;
using FTOptix.CommunicationDriver;
using FTOptix.Core;
#endregion

public class DesignTimeNetLogic2 : BaseNetLogic
{
    [ExportMethod]
    public void Method1()
```

```

{
    // Insert code to be executed by the method
        // Insert code to be executed by the method
    Folder userFolder = Project.Current.Get<Folder>("Security/Users");
    if (userFolder == null) {
        Log.Error("CreateNewUser", "Cannot find Users folder");
        return;
    }
    /*else if (string.IsNullOrEmpty(groupName)) {
        Log.Error("CreateNewGroup", "Cannot create group with empty name");
        return;
    } else if (userFolder.Get<Group>(groupName) != null) {
        Log.Error("CreateNewGroup", "Group already exists!");
        return;
    }*/
    // Create a new group
    //var newGroup = InformationModel.Make<Group>("Nou_Grup");
    //var newUser = InformationModel.Create<User>("Xavier");
    var newUser = InformationModel.Make<User>("Xavier");
    // Add the group to the folder
    //userFolder.Add(newGroup);
    userFolder.Add(newUser);

}
}

```

Or even more clean with unnecessary lines

```

#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrj;
using FTOptix.UI;
using FTOptix.RAEtherNetIP;
using FTOptix.NativeUI;
using FTOptix.NetLogic;
using FTOptix.CoreBase;
using FTOptix.Retentivity;
using FTOptix.CommunicationDriver;
using FTOptix.Core;
#endregion

public class DesignTimeNetLogic2 : BaseNetLogic

```

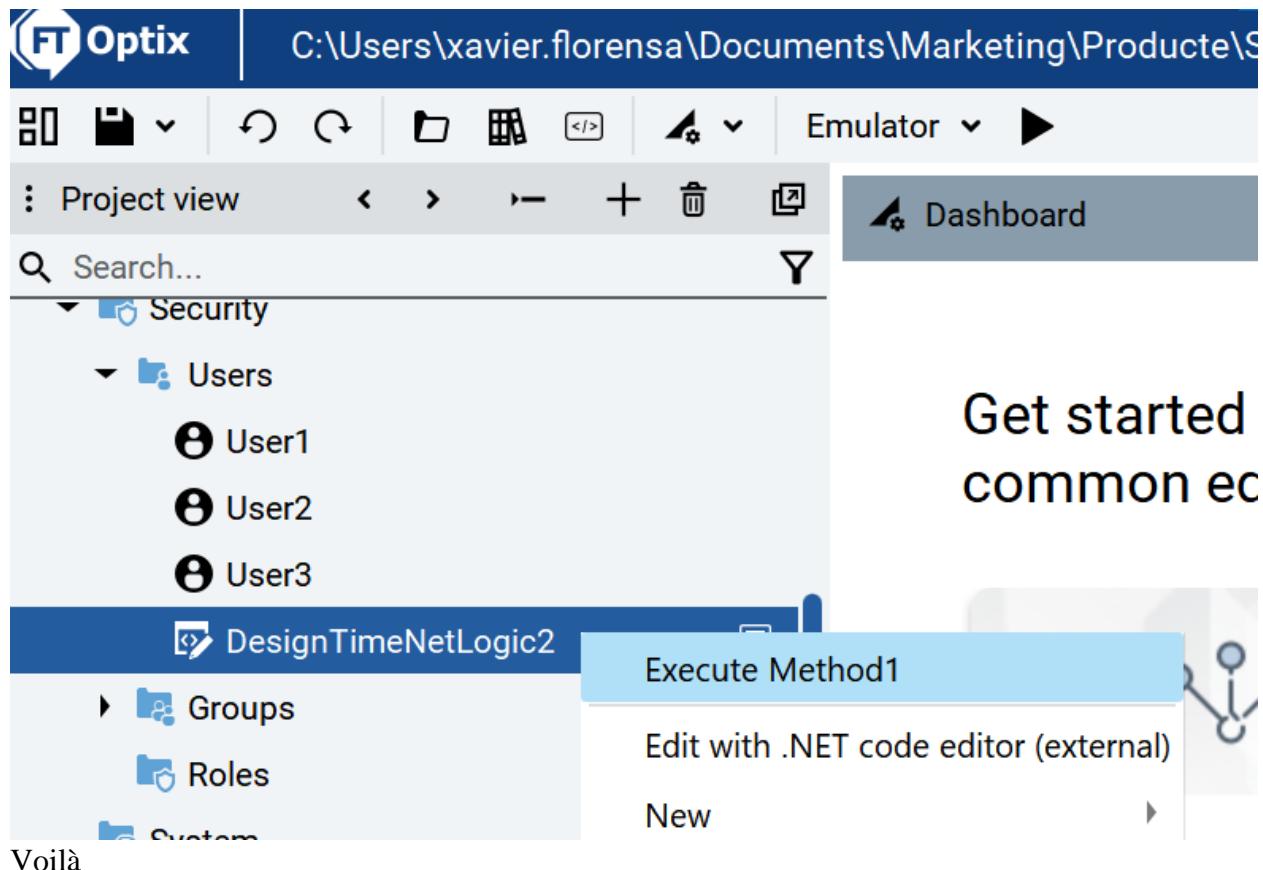
```

{
    [ExportMethod]
    public void Method1()
    {
        // Insert code to be executed by the method
        // Insert code to be executed by the method
        Folder userFolder = Project.Current.Get<Folder>("Security/Users");
        if (userFolder == null) {
            Log.Error("CreateNewUser", "Cannot find Users folder");
            return;
        }
        /*else if (string.IsNullOrEmpty(groupName)) {
            Log.Error("CreateNewGroup", "Cannot create group with empty name");
            return;
        } else if (userFolder.Get<Group>(groupName) != null) {
            Log.Error("CreateNewGroup", "Group already exists!");
            return;
        }*/
        // Create a new user
        var newUser = InformationModel.Make<User>("Xavier");
        // Add the user to the folder
        userFolder.Add(newUser);

    }
}

```

Now execute the design time script



Voilà

**Optix** | C:\Users\xavier.florensa\Documents\Marketing\Products\

Project view Emulator ▾ Dashboard

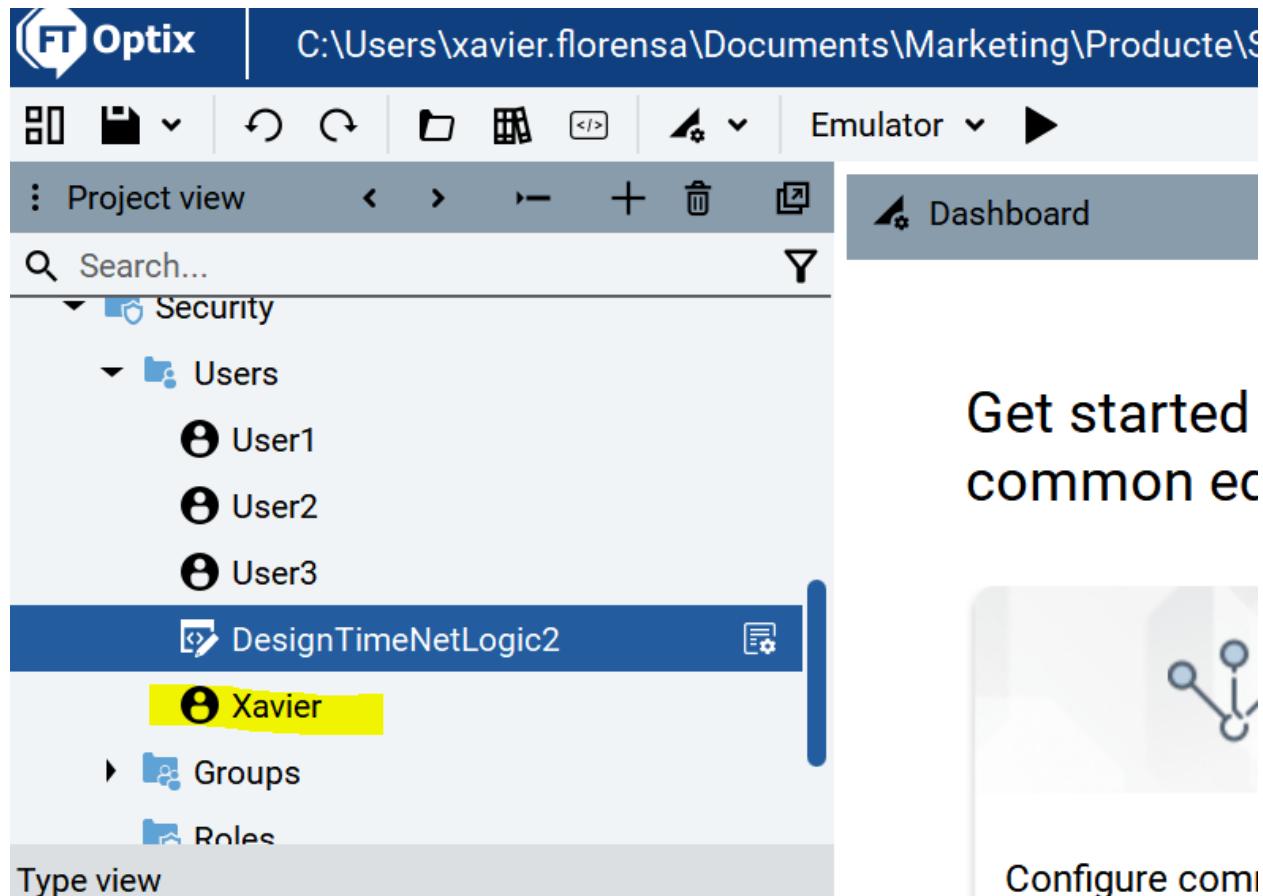
Search... ▾

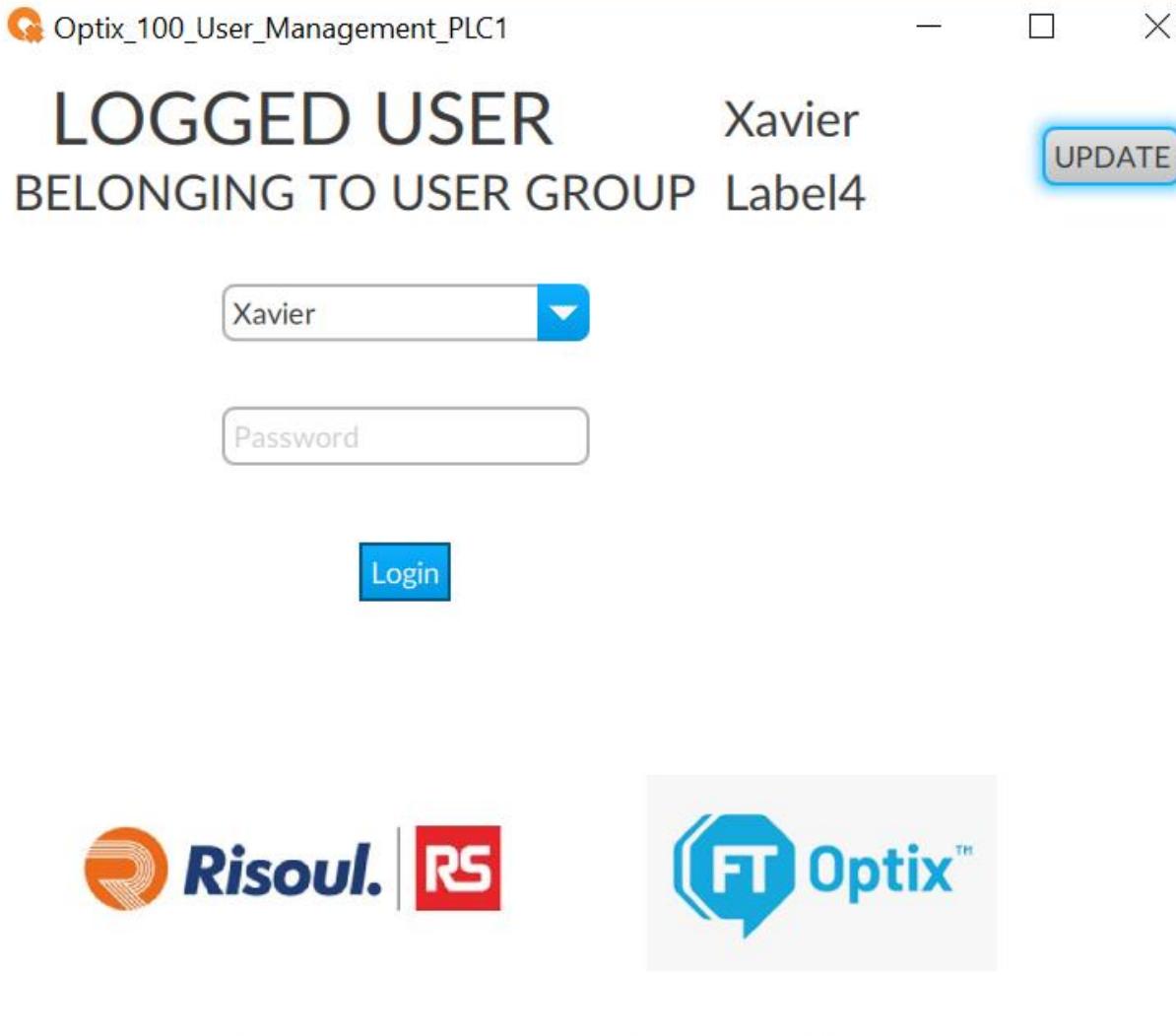
Security

- Users
  - User1
  - User2
  - User3
- Xavier
- Groups
- Roles

Type view

Get started common ec





#### 51.6. User batch creation in design time (100 users)

Now we can create 100 users as a batch task

 C:\Users\xavier.florensa\Documents\Marketing\Products\Software\Optix\Optix\_100\_User\_Management\_PlC2

As you can see in this video

<https://youtu.be/wpdAVwmzsEs>

The code is located here

[https://github.com/xavierflorensa/FTOptix\\_100\\_User\\_Management\\_PlC](https://github.com/xavierflorensa/FTOptix_100_User_Management_PlC)

#### 51.7. Manage user rights to Panels in a Native way

Create your Users, Groups and Roles.



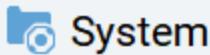
C:\Users\xavier.florensa\Docume



Project view



Search...



Configure different roles to different users

The screenshot shows a software application window titled "Emulator". The top navigation bar includes icons for file operations and tabs labeled "Panel...type), Panel...type), Panel...type), Panel...type)" and "Manage...roups".

**Users** panel:

- Add button
- Assign to groups or roles button
- Victor:
  - Admin
  - Administrator
- Xavier:
  - Tech
  - Technician

**Groups** panel:

- Add button
- Assign to users or r button
- Admin:
  - Victor
- Tech:
  - Xavier

**Roles** panel:

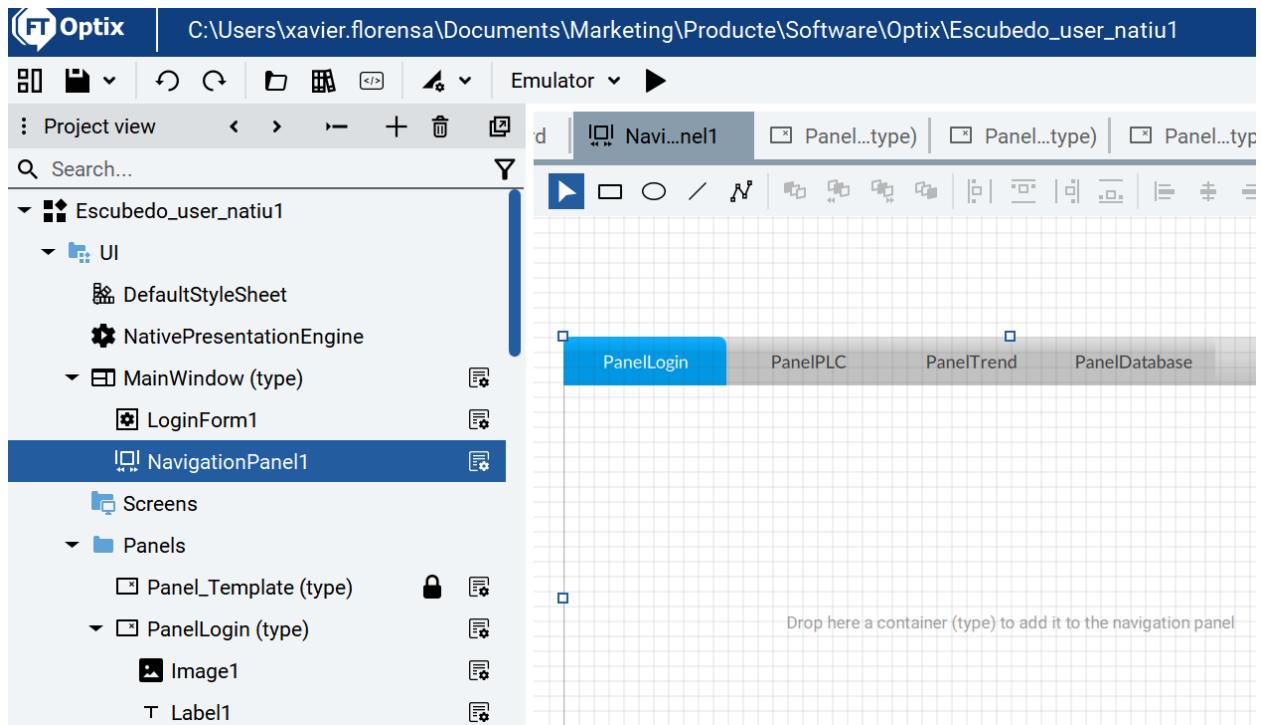
- Add button
- Assign to users or r button
- Administrator:
  - Victor
- Technician:
  - Xavier

**Buttons at the bottom:**

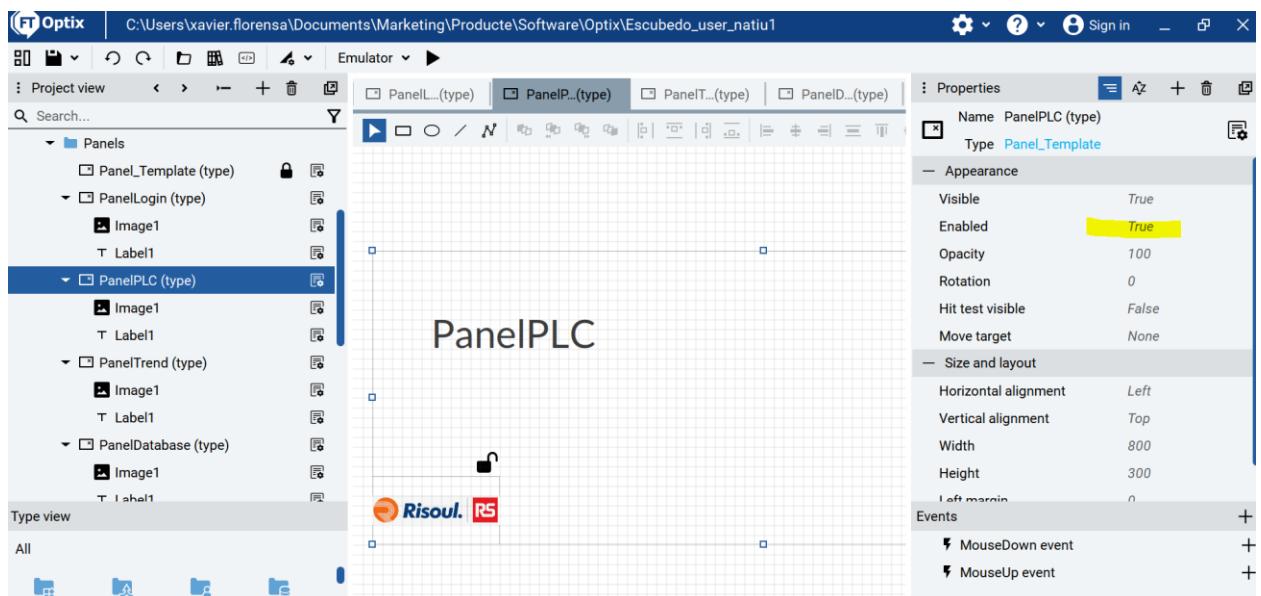
- Database icon
- Exit button

Insert your Login widget as explained early on this chapter

Create your Navigation Panel and Panels



Select your Panel (Do this for all Panels) and setup the dynamic link on Enabled



Add a dynamic link to Enabled

(type)

Properties

Name PanelPLC (type)

Type Panel\_Template

Visible Add Dynamic Link

Enabled True

Opacity 100

Rotation 0

Visible

Select the desired Role under Aliases / Session / Session / Roles

Basic Advanced

UI > Panels > PanelPLC > Enabled

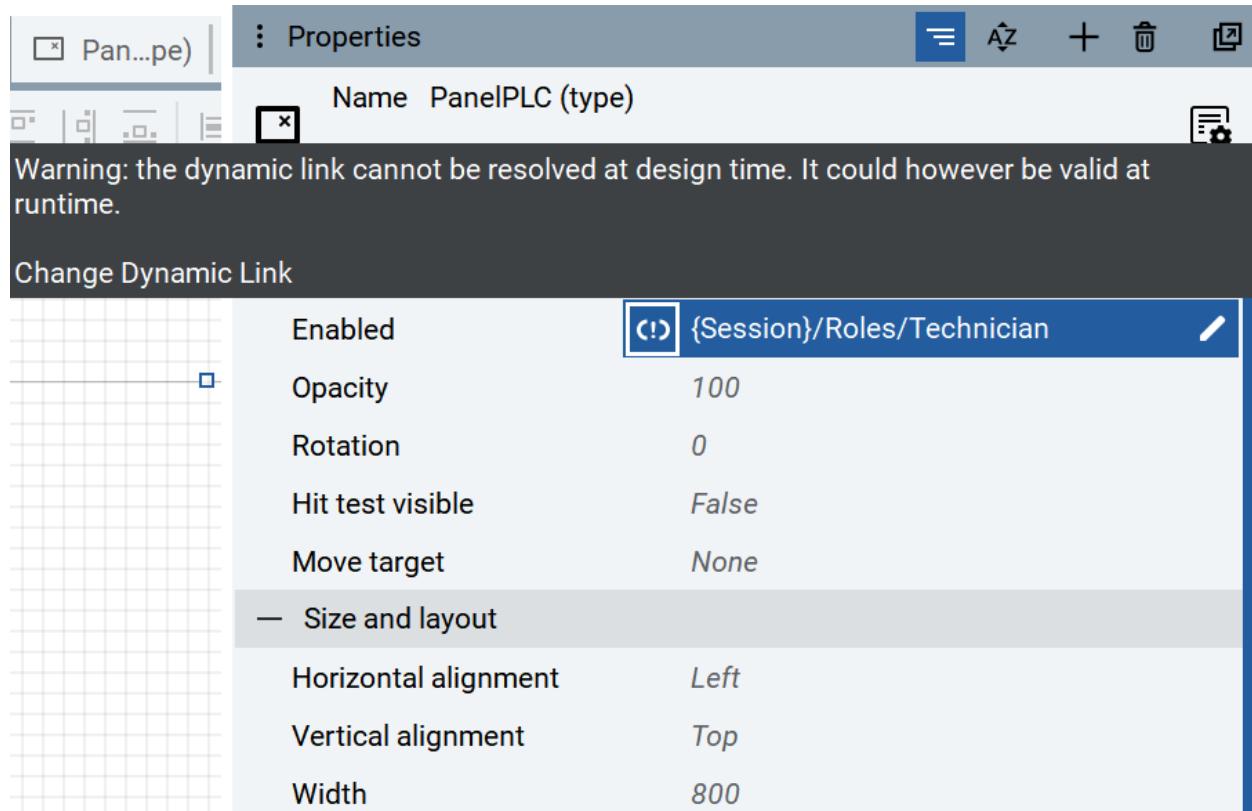
Search...

Aliases

- {Session}
- Session
  - User (Pointer)
  - Locale
  - Language
  - Measurement system
  - Actual locale
  - Actual language
  - Actual measurement system
  - Interactive session
  - Time zone offset
- Groups
- Roles
  - Administrator
  - Technician
  - Loggedin
  - ChannePassword

Attribute ⓘ

Cancel Select Remove link



That's all

When you execute and log in you will be the Panel disabled, as you can see on the picture with difuminated intensity on the contents

 Escubedo\_user\_natiu1



Current User:

Xavier

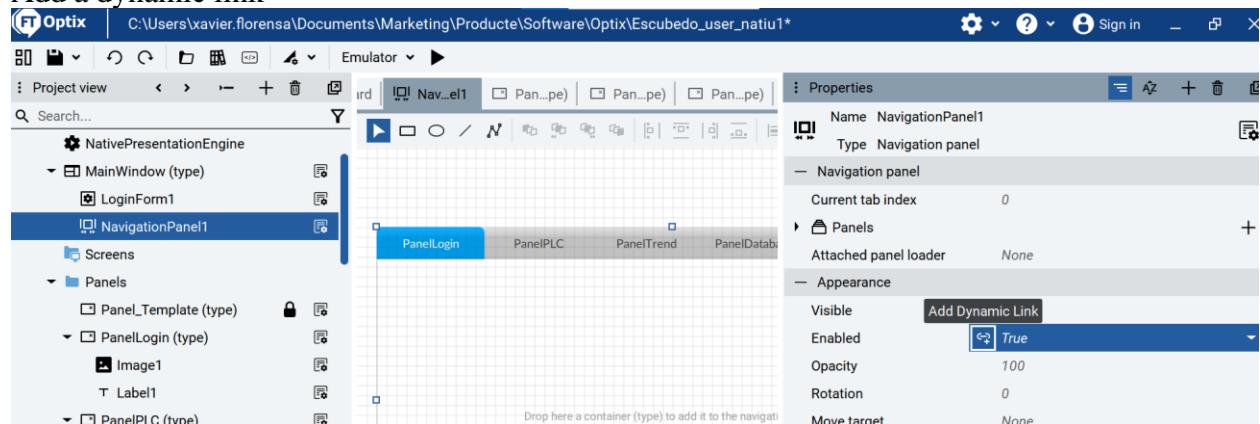
# Trends

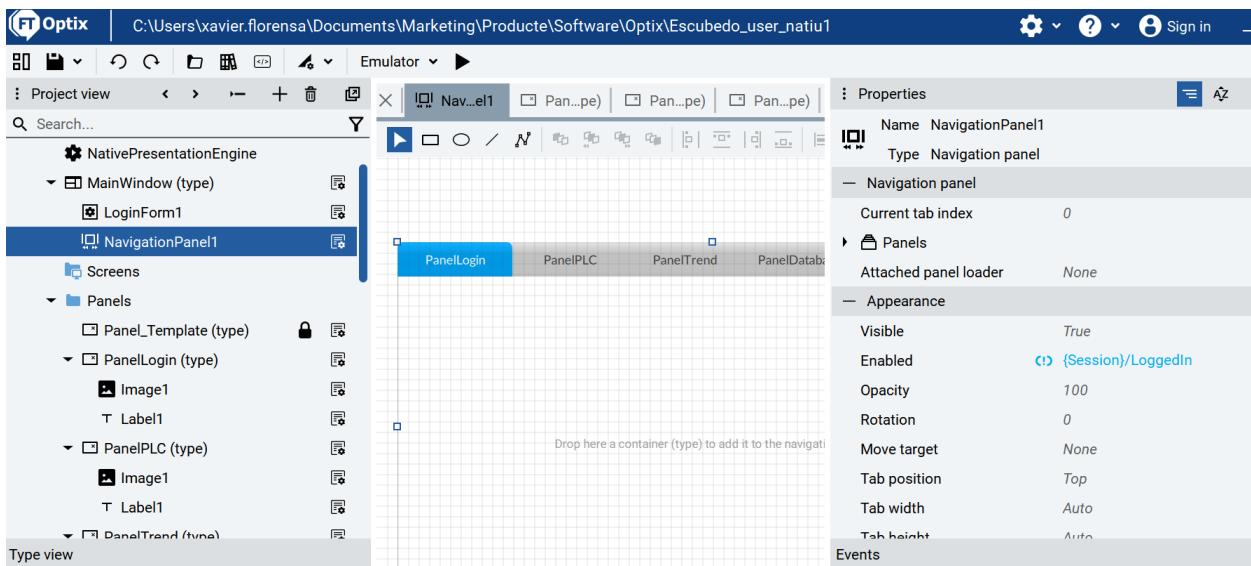
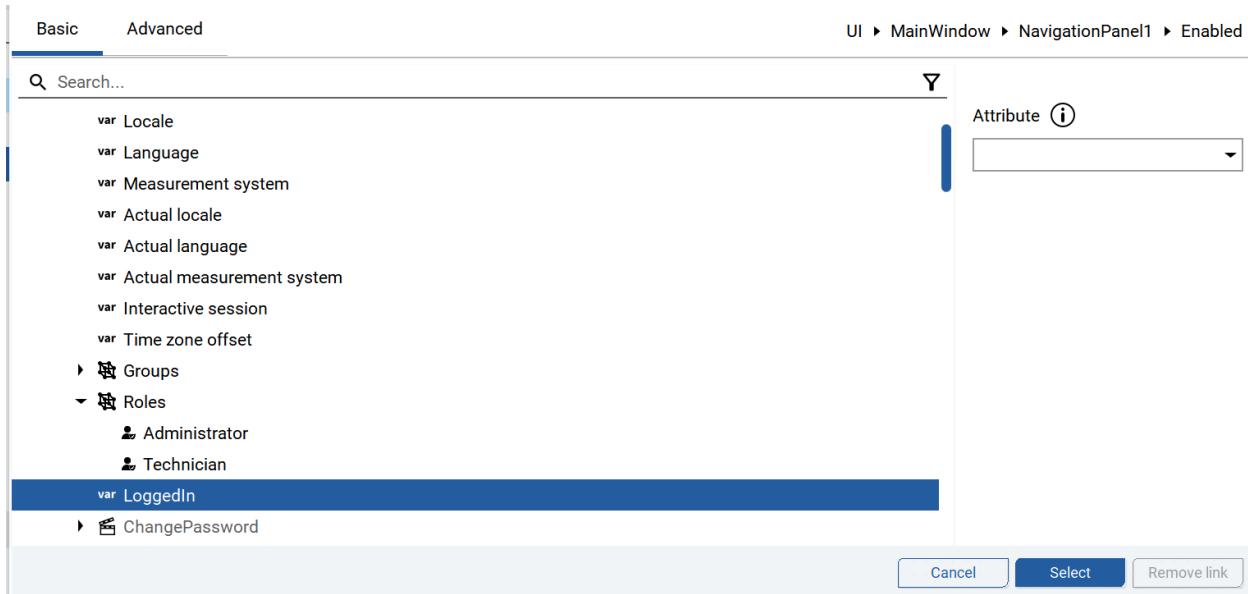
[Logout](#)



You can make the panels not enabled before any user is logged this way

Add a dynamic link



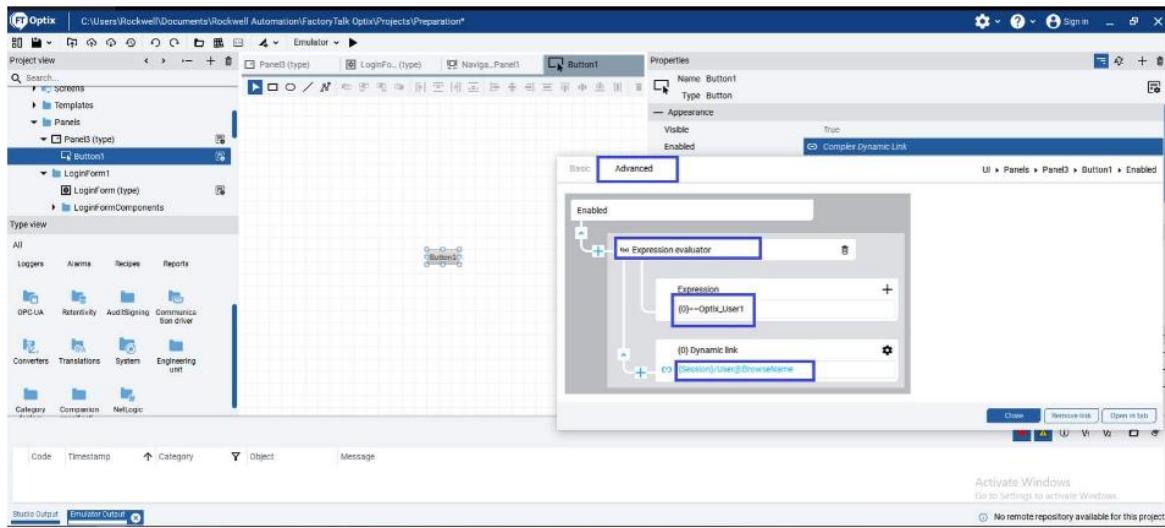


That's all

You can find the code here

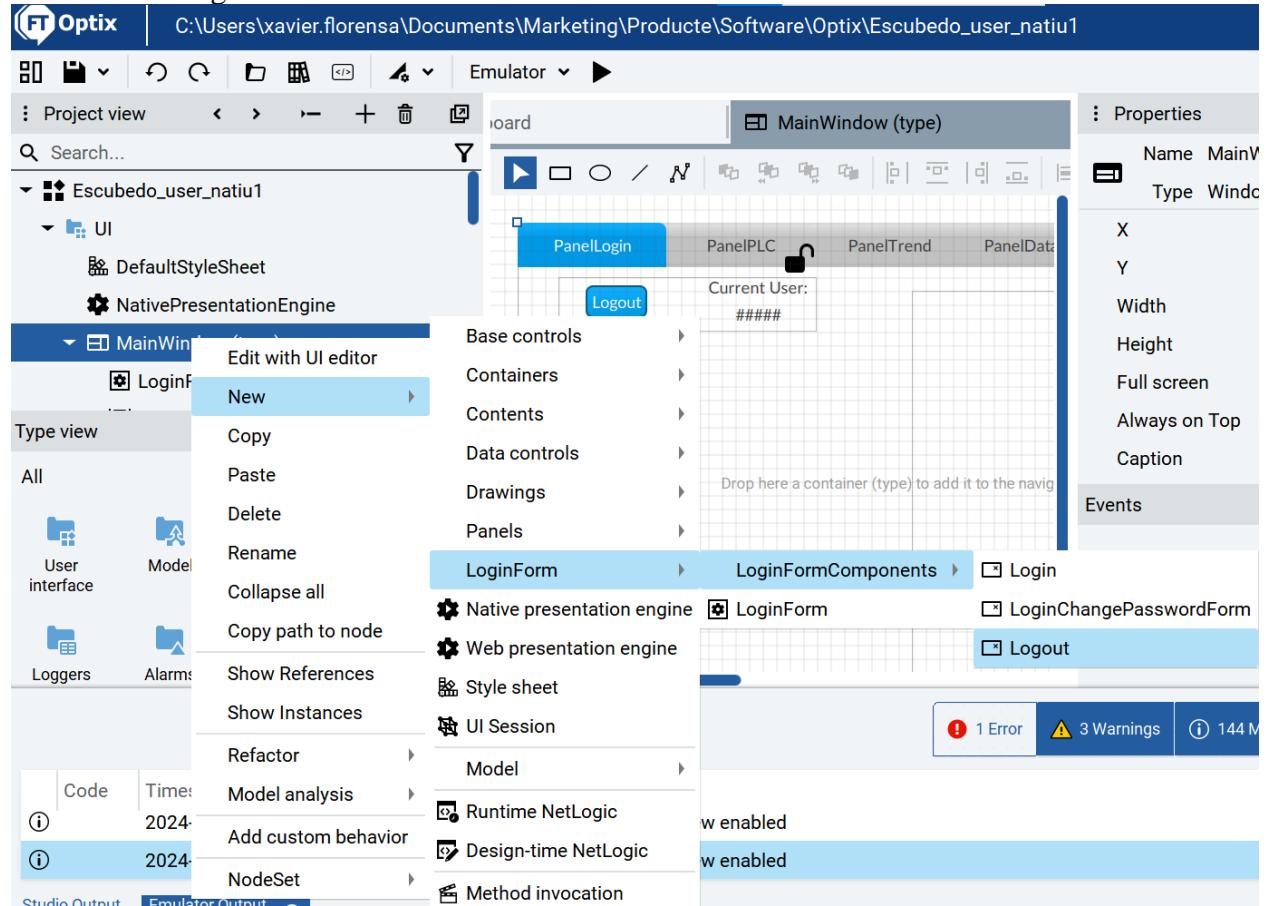
[https://github.com/xavierflorensa/FTOptix\\_user\\_management\\_panels\\_native](https://github.com/xavierflorensa/FTOptix_user_management_panels_native)

You can also use advanced dynamic link



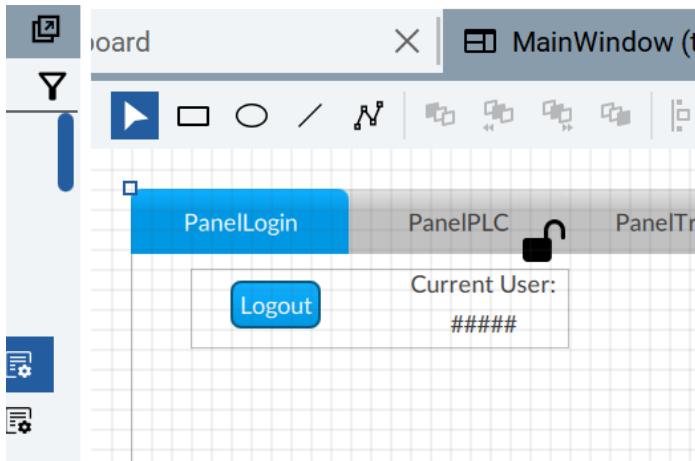
## 51.8. Making Login Form invisible when a user is Logged in

First of all you have to grant you have a logout button to perform a Logout  
Insert a new Logout



Make it little

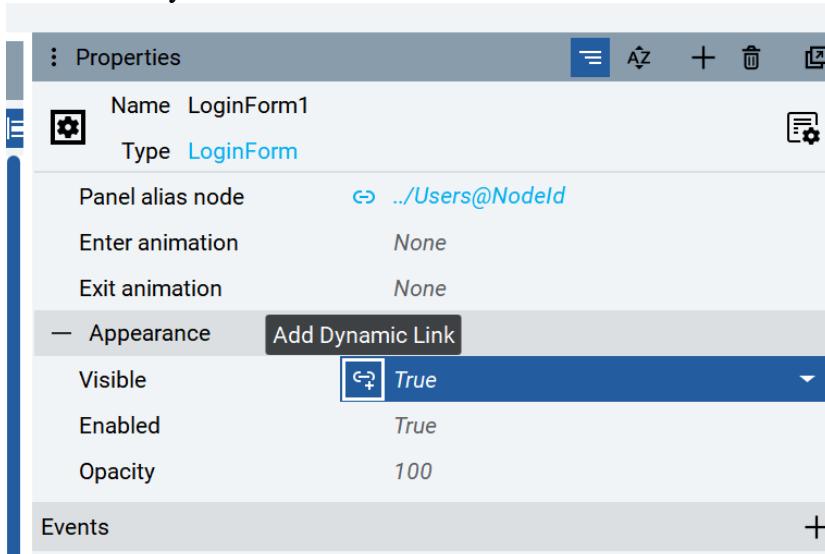
Clicking on the padlock to be able to modify it



Then apply to the Login visible property an advanced condition like

The equivalent to Visible = NOT (LoggedIn)

Add a new dynamic link



Place the cursor over LoggeId and select

Basic Advanced

UI ▶ MainWindow ▶ LoginForm1 ▶ Visible

Search... Y

Attribute ⓘ ▼

Attribute ⓘ ▼

Session

- var User (Pointer)
- var Locale
- var Language
- var Measurement system
- var Actual locale
- var Actual language
- var Actual measurement system
- var Interactive session
- var Time zone offset

Groups

Roles

**var LoggedIn**

Cancel Select Remove link

Enter again into the dynamic link to edit

Properties

Name **LoginForm1**

Type **LoginForm**

Panel alias node **./Users@NodeId**

Enter animation **None**

Exit animation **None**

Appearance

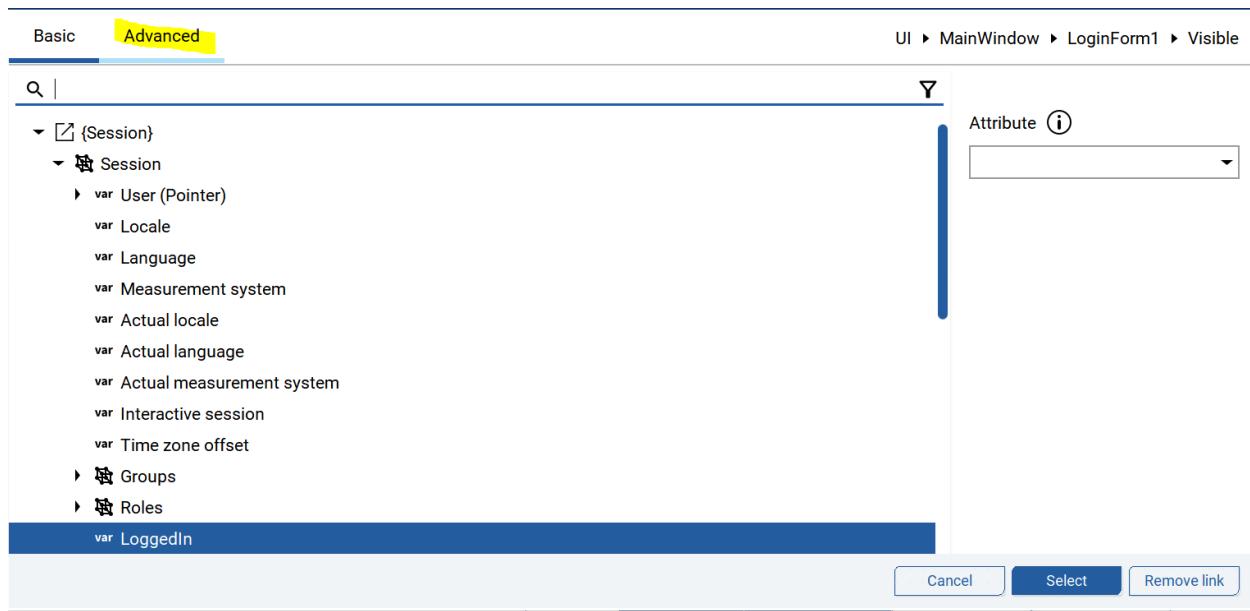
Visible **{Session}/LoggedIn**

Enabled **True**

Opacity **100**

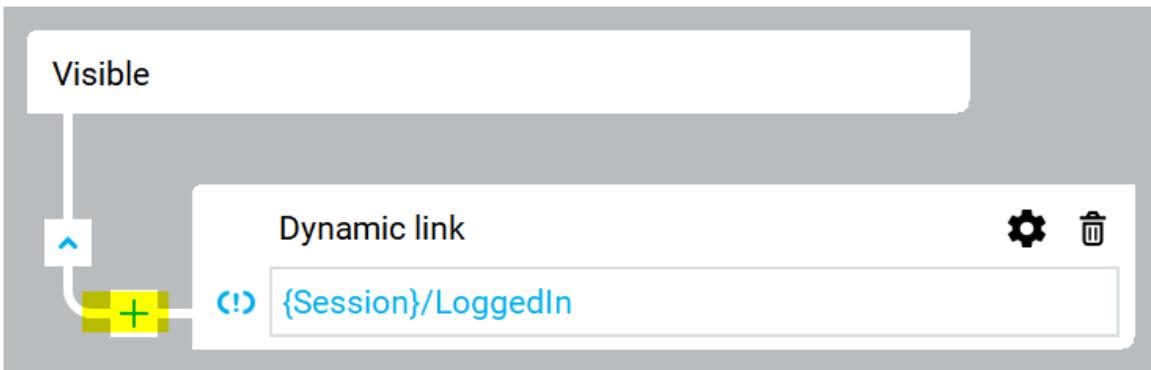
Events +

Go to Advanced



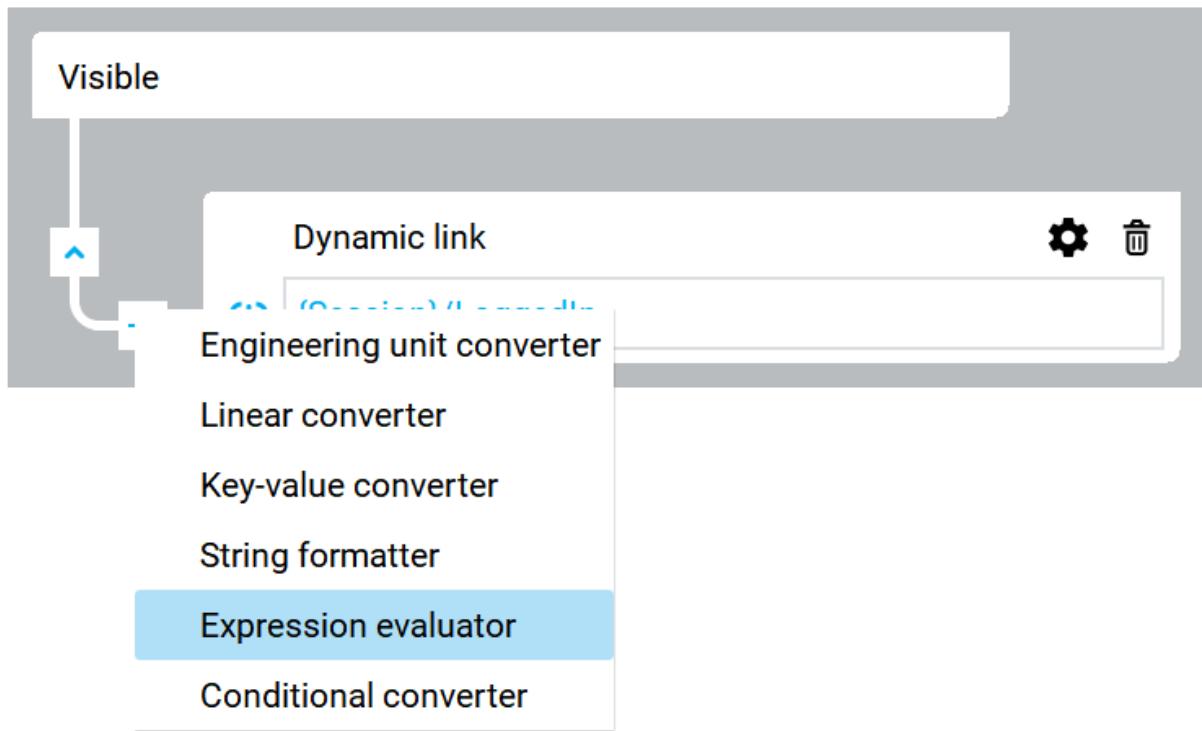
Add a Expression evaluator

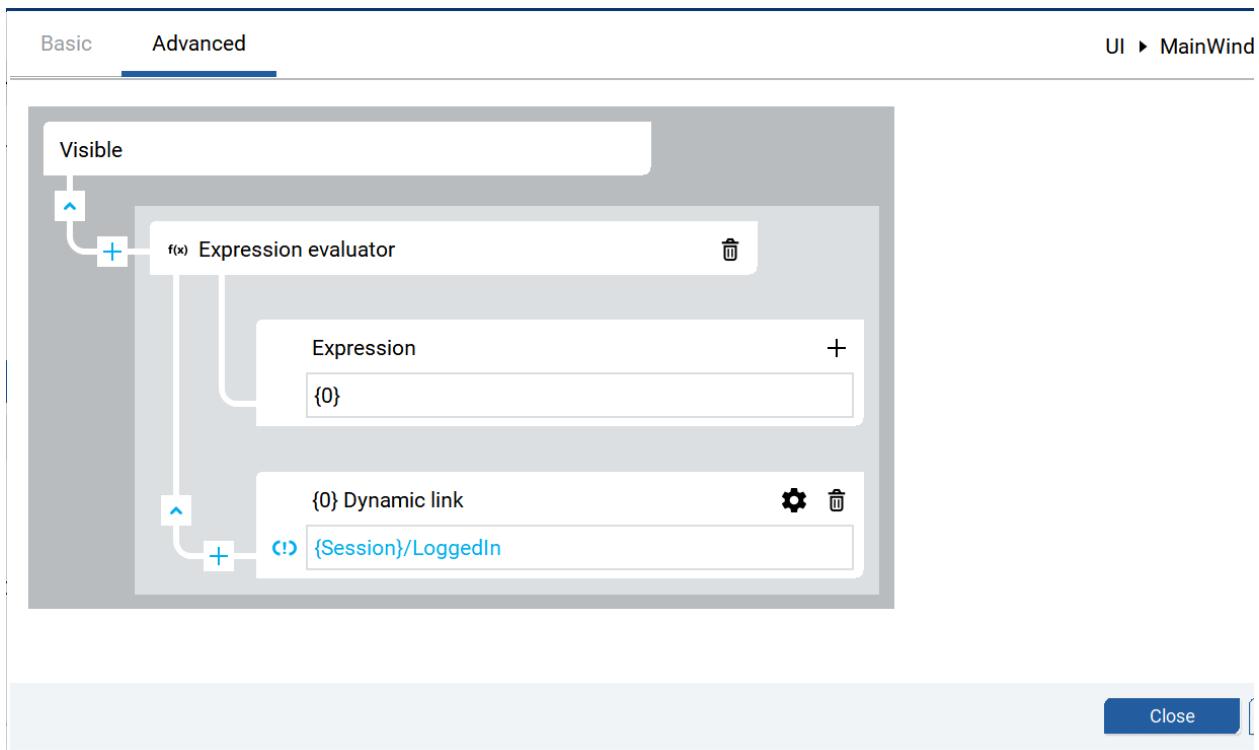
Basic Advanced



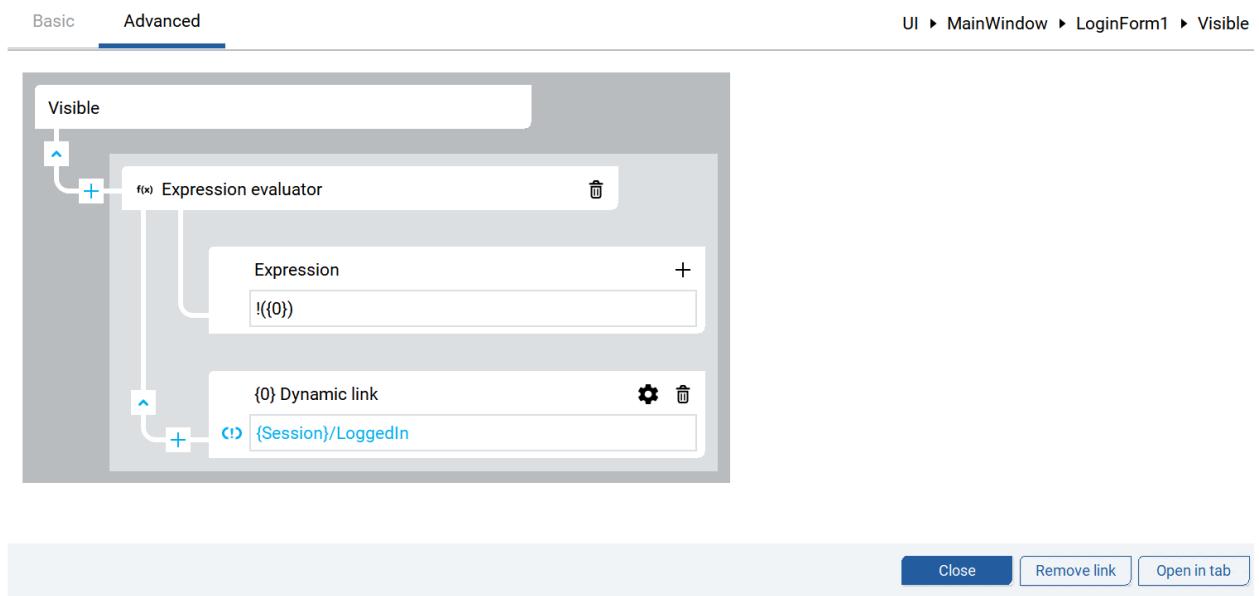
Basic      Advanced

---





Edit the expression this way `!({0})`



That's all

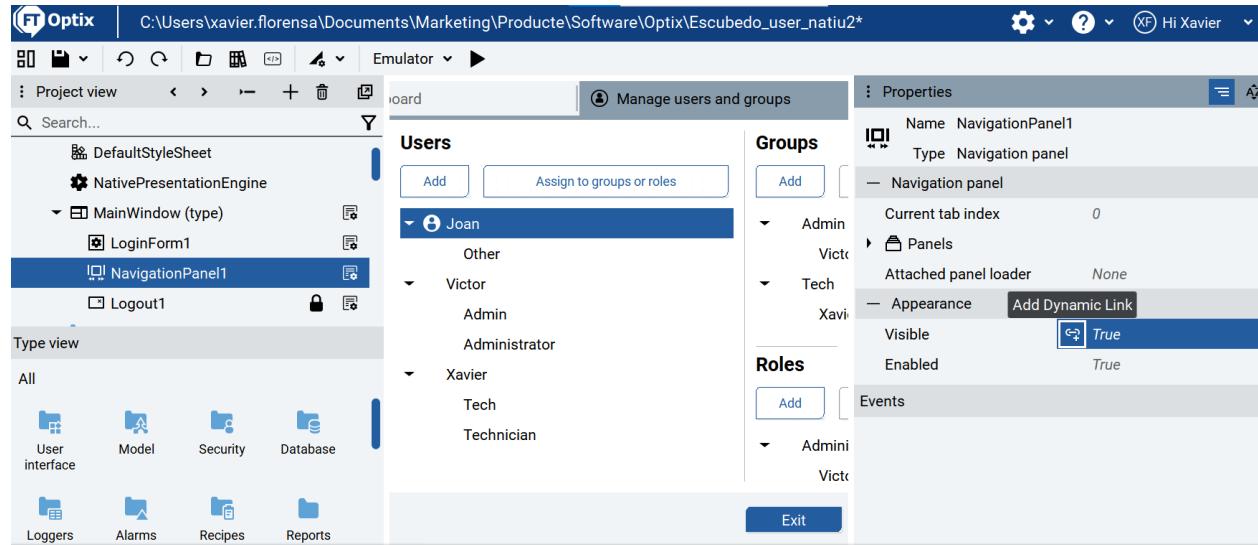
### 51.9. Make Panels visible if user belongs to one of a set of Roles

Imagine you want to exclude a user if he does not belong to any of a set of roles.

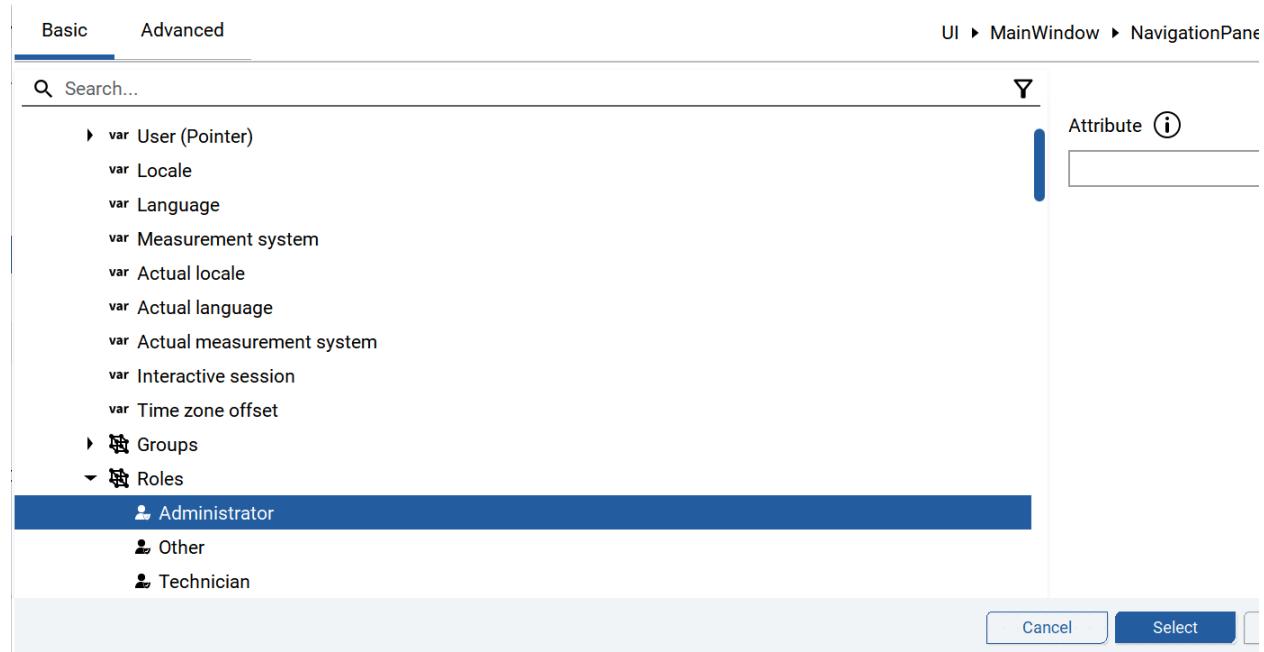
User Joan has Role Other, which is not Administrator or Technician Role.

Then you can do this way

Add a new dynamic link to Navigation Panel property

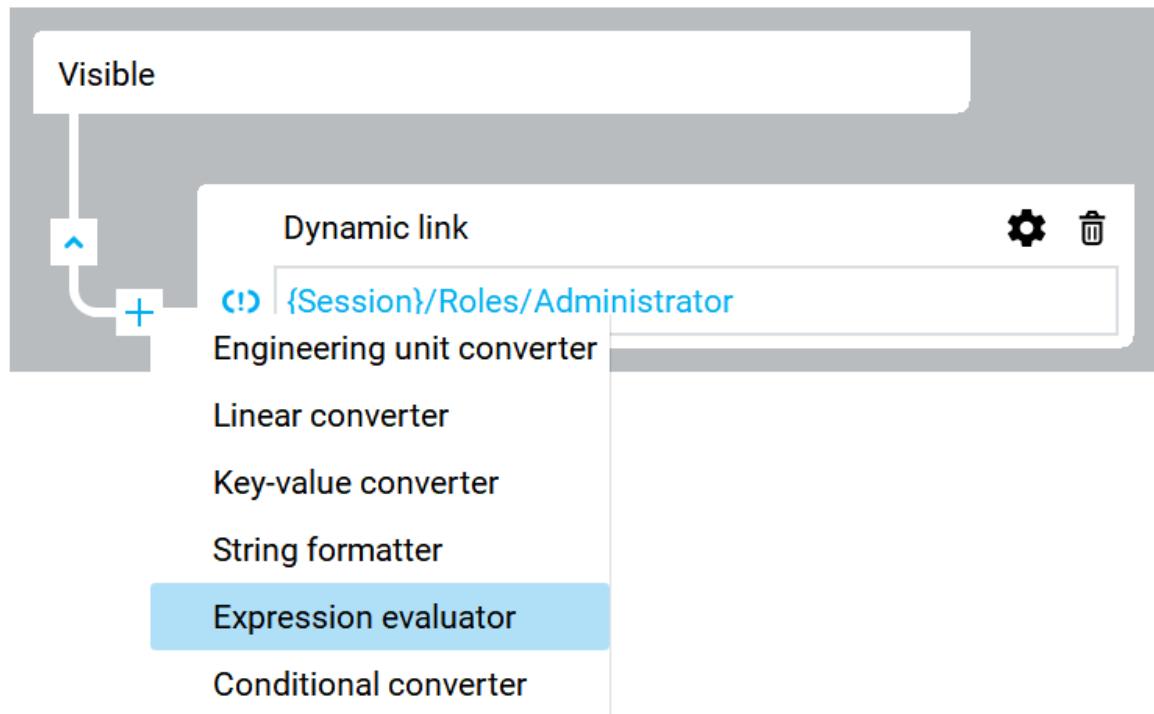


Highlight Administrator and click on Select



Open the dynamic link again to edit it on Advanced  
Add an expression evaluator

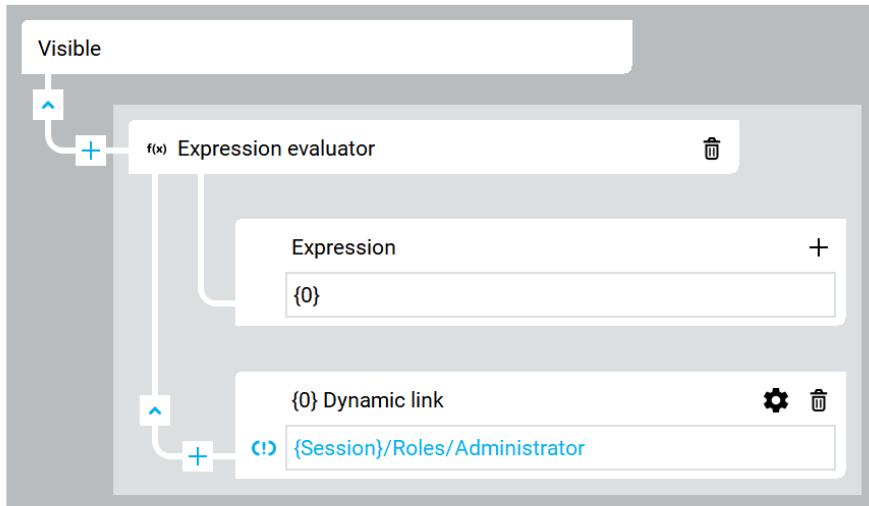
Basic Advanced



Basic

Advanced

UI ▶ MainWindow



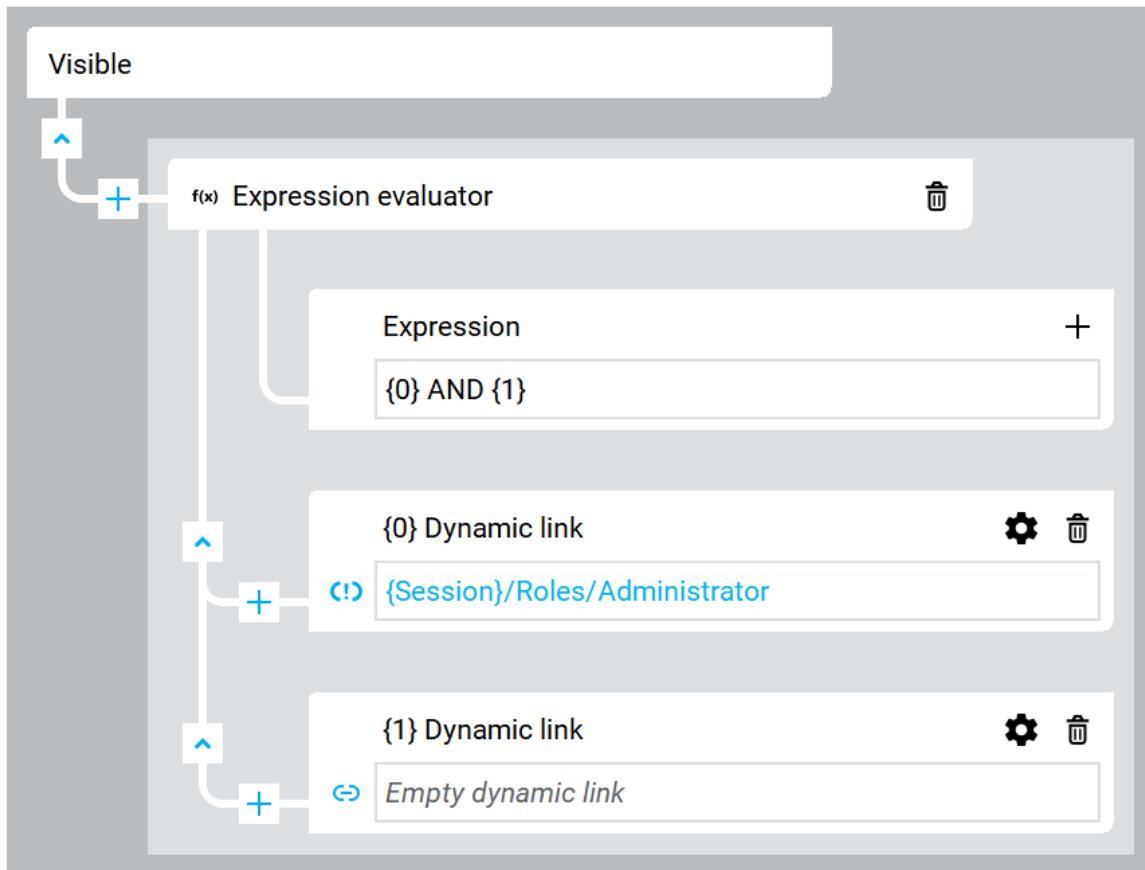
Close

Write {0} AND {1} on Expression

A new link will appear

Basic

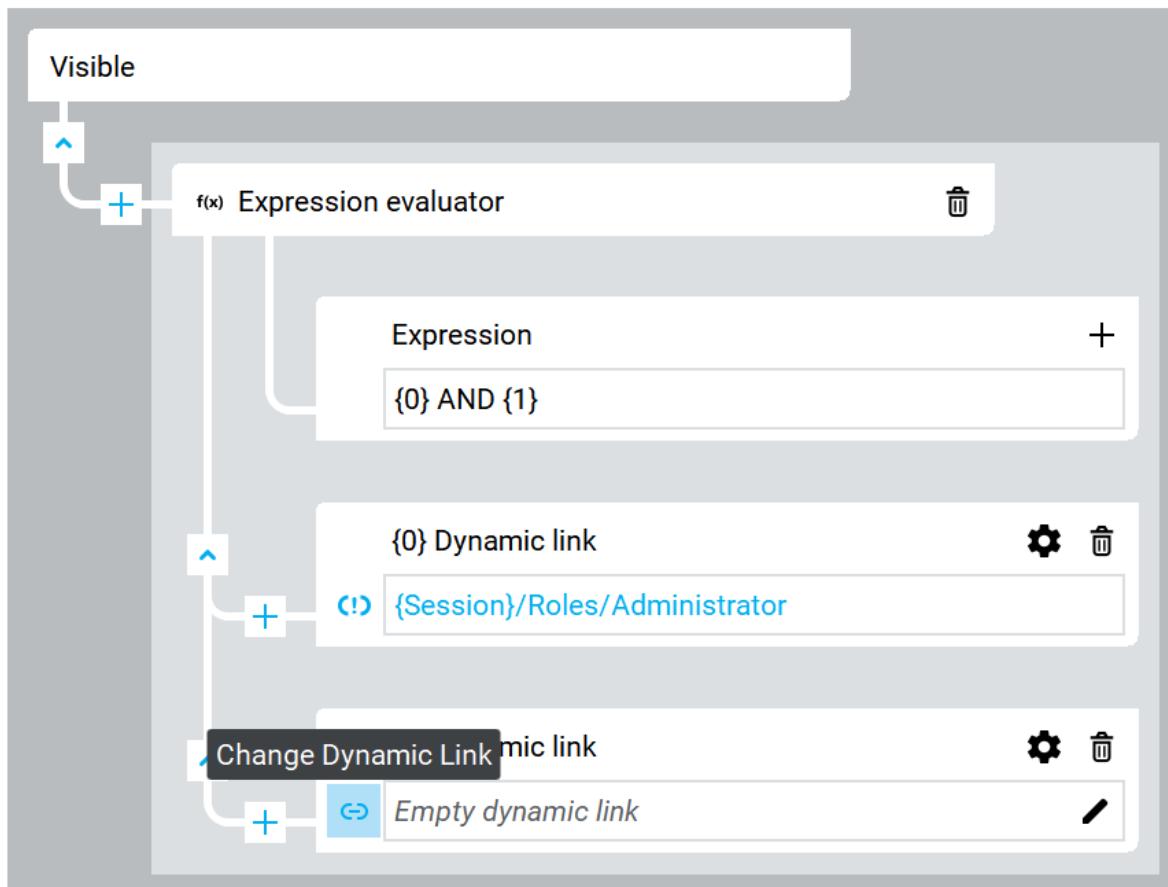
Advanced



Select the “Technician” role clicking on the dynamic link

Basic

Advanced



Point to technician role, and select

UI ▶ MainWindow ▶ NavigationPanel1 ▶ Visible ▶ ExpressionEvaluator1 ▶ Source1

Search... Y

Attribute   ▼

var Locale  
var Language  
var Measurement system  
var Actual locale  
var Actual language  
var Actual measurement system  
var Interactive session  
var Time zone offset

▶ Groups  
- Roles  
  Administrator  
  Other  
  **Technician**

var LoggedIn

Cancel Select



Basic Advanced UI ▶ MainWindow ▶

Visible

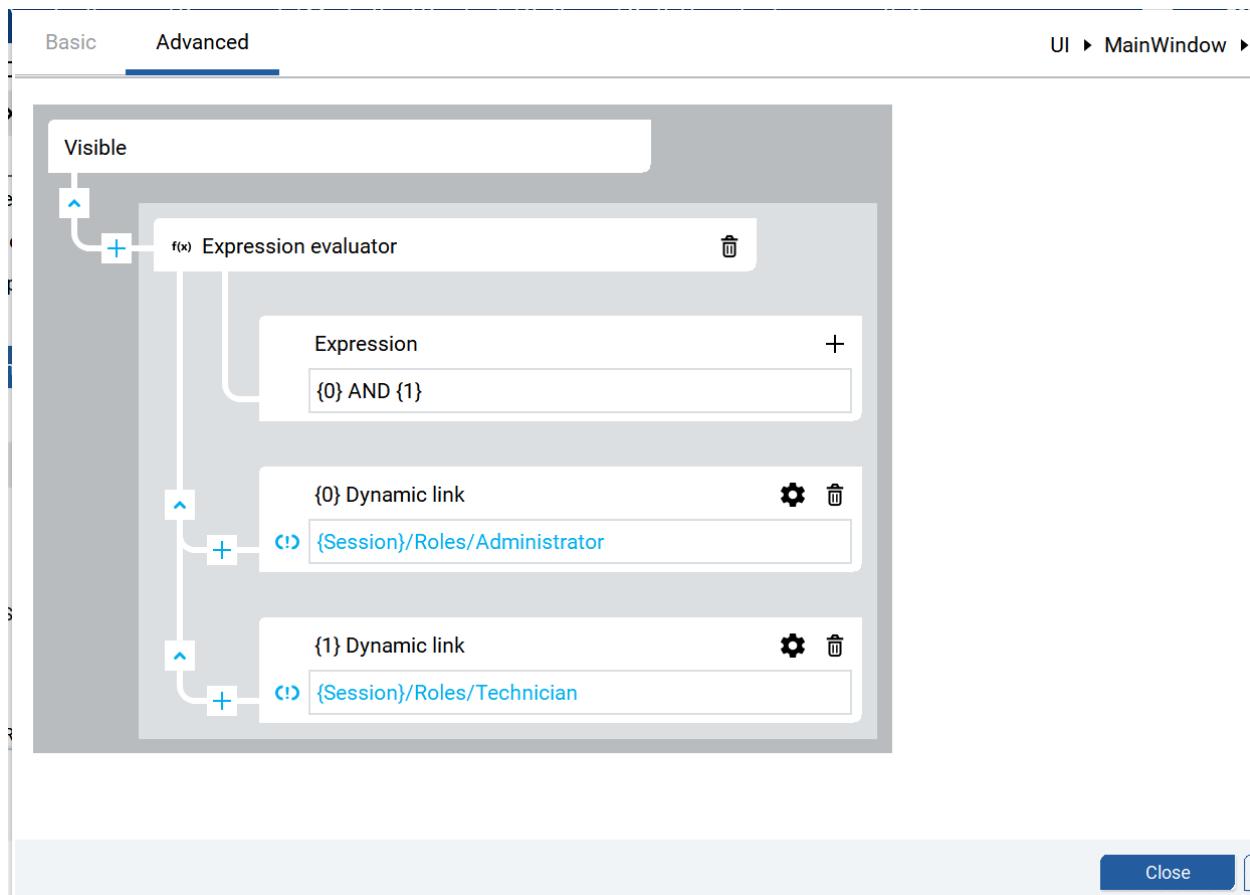
f(x) Expression evaluator Delete

Expression +  
{0} AND {1}

{0} Dynamic link ⚙️ Delete  
!{Session}/Roles/Administrator

{1} Dynamic link ⚙️ Delete  
!{Session}/Roles/Technician

Close



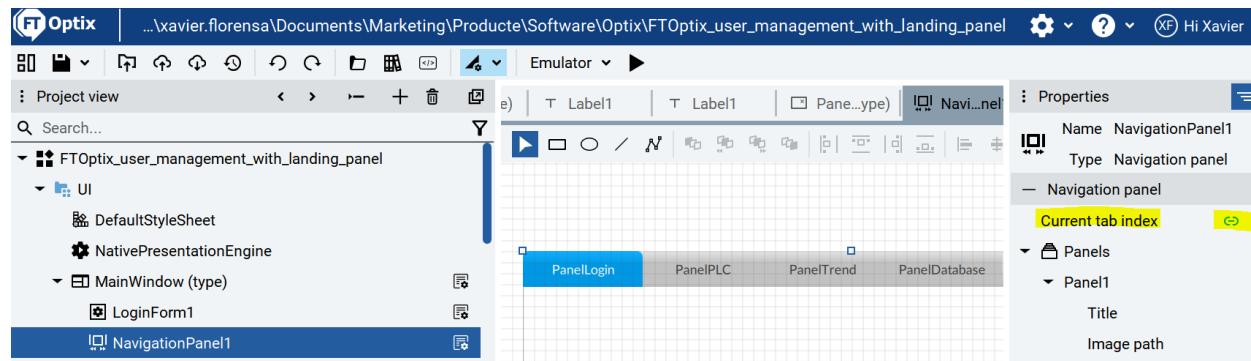
That's all

## 51.10. User management with landing panel

We want that after logging in, depending on the group, the first panel to open is different.

You have an index (0,1,2..) for that purpose on the Navigation Panel properties.

By default is 0, corresponding to Panel1



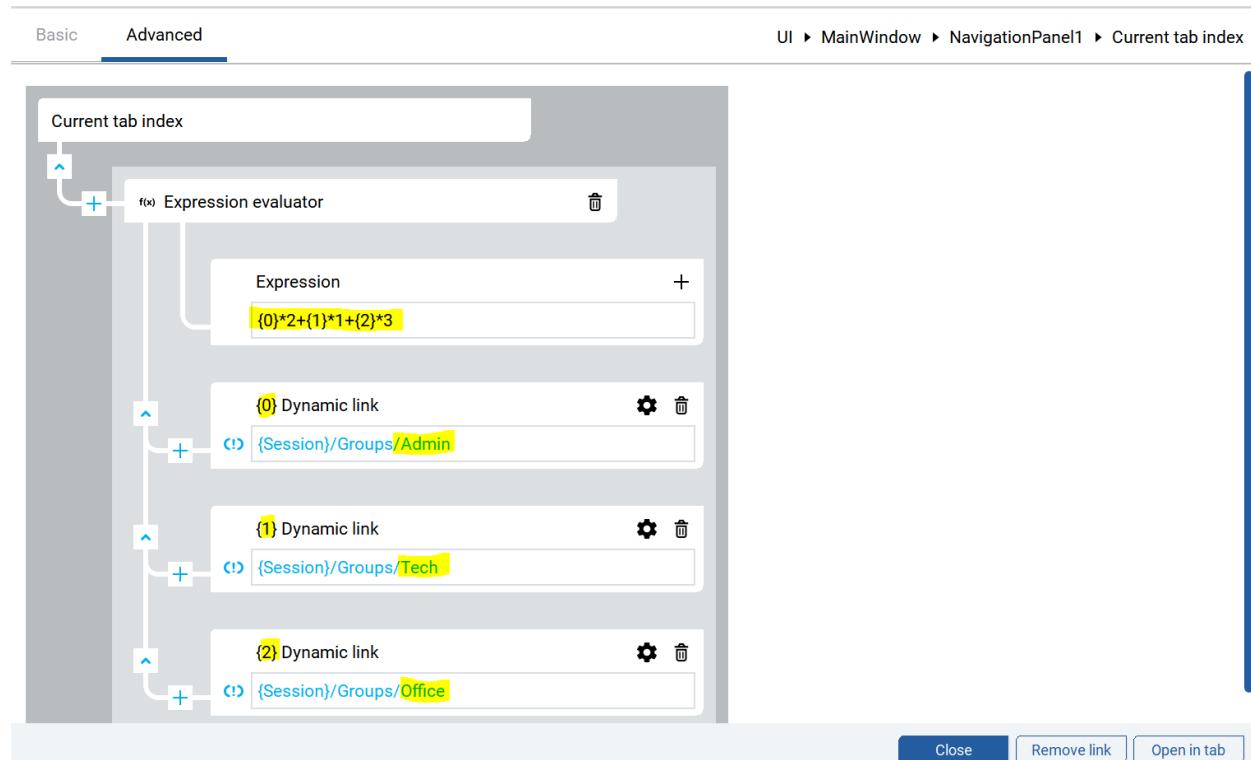
Let's introduce a complex dynamic link there with expression evaluator

With an arithmetic addition which result will be a number (1, 2 or 3).

So Admin group will open Panel number 2

Tech group will open Panel Number 1

And finally Office group will open Panel number 3



You can see the final result here

<https://youtu.be/uUqUdoh2VVc>

You can find the code here

[https://github.com/xavierflorensa/FTOptix\\_user\\_management\\_with\\_landing\\_panel.git](https://github.com/xavierflorensa/FTOptix_user_management_with_landing_panel.git)

### 51.11. Manage user rights to Panels on a NavigationPanel in C#

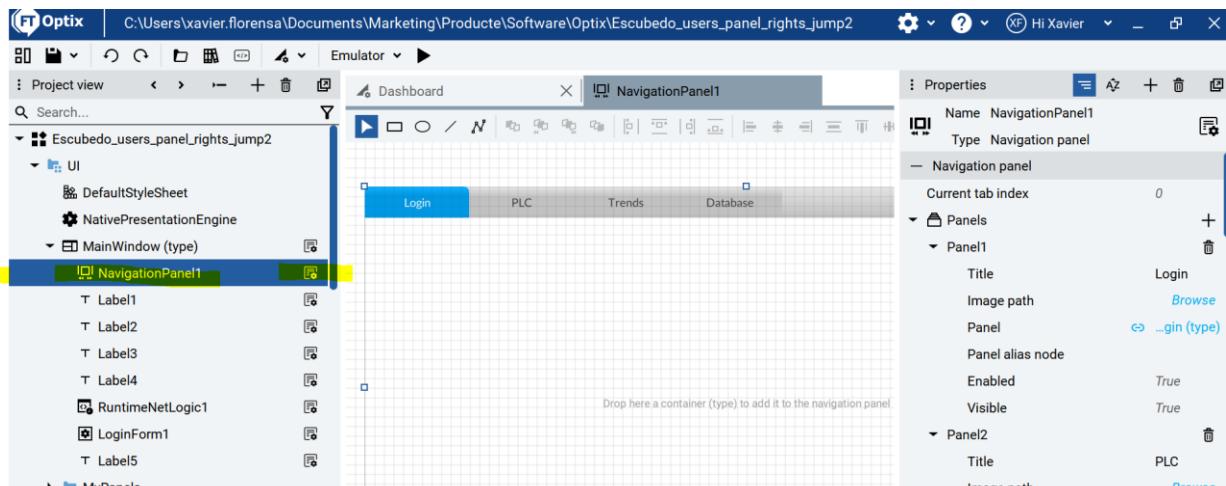
You can see the results on this video

<https://youtu.be/bp9QS3-gpKI>

The key to do this is to make each Panel Enabled or disabled, depending on the user group that is Logged.

This is easy since we already know how to be aware of which user group belongs to the logged user from previous chapters.

First of all you have to have access to the Navigation Panel in C# like this



```
var navigationPanel = Project.Current.Get<NavigationPanel>("UI/MainWindow/NavigationView1");
```

Then you can select which Panel is enabled and which is disabled, for instance like this

```
var panelPC = navigationPanel.Panels[1];
var panelTrends = navigationPanel.Panels[2];
var panelDatabase = navigationPanel.Panels[3];
if (group.BrowseName.ToString() == "Technician")
{
    navigationPanel.Enabled = true;
    panelPC.Enabled = false;
    panelTrends.Enabled = true;
```

```

        panelDatabase.Enabled = true;

    }
    else
    {
        if (group.BrowseName.ToString() == "Administrator")
        {
            navigationPanel.Enabled = true;
            panelPC.Enabled = true;
            panelTrends.Enabled = false;
            panelDatabase.Enabled = true;
        }
    }

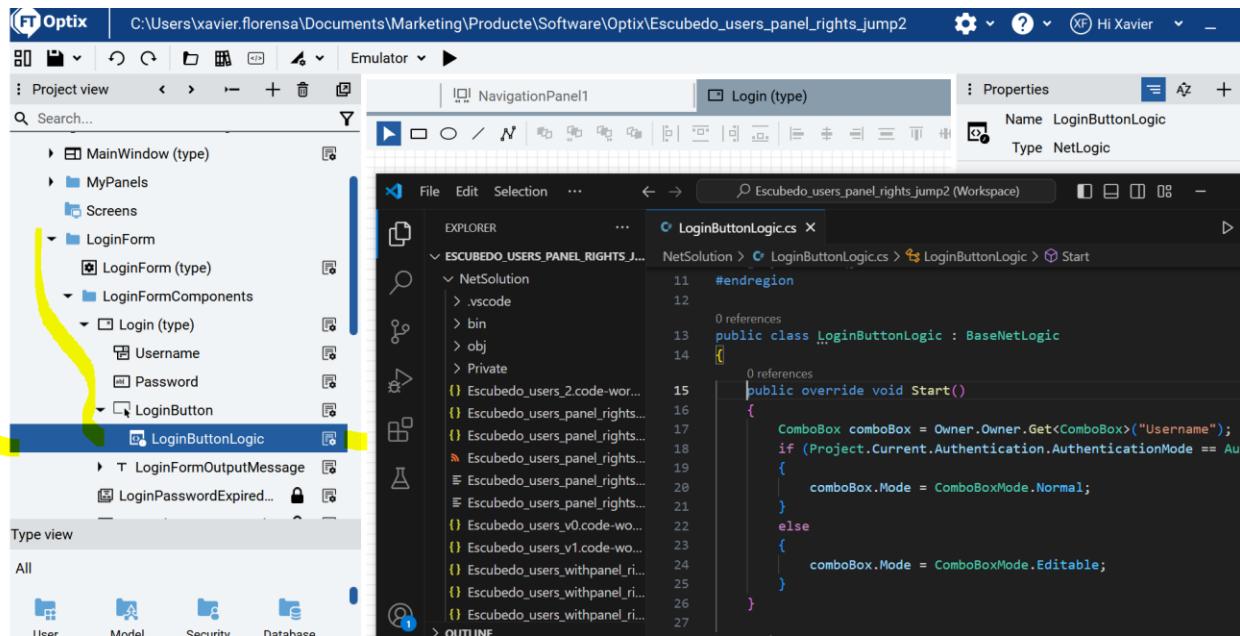
```

Be aware that we also block the enabling of the whole NavigationPanel before any user is logged, and, once logout, when the user corresponds to Anonymous, to force a login as a previous step to see and use the panels on runtime.

You can do this this way

```
navigationPanel.Enabled = true;
```

Another request from customers is that just after clicking the Login button, all these changes take effect. So let's introduce our code on the Login Form button script, located here:



So we can introduce this code after the logic on the click button export method

```

[ExportMethod]
0 references
public void PerformLogin(string username, string password)
{
}

```

Like this

```

//Start Custom
behaviour*****
    // Mostrar el usuari logat
    var usuari = Session.User;
    var usuari = Session.User.BrowseName;
    var mylabel = Project.Current.Get<Label>("UI/MainWindow/Label2");
    mylabel.Text = usuari.ToString();

    var navigationPanel =
Project.Current.Get<NavigationView>("UI/MainWindow/NavigationView1");
    //Mostra el grup del usuari
    var userGroups =
usuario.Refs.GetObjects(FTOptix.Core.ReferenceTypes.HasGroup, false);
    if (usuari== "Anonymous")
    {
        navigationPanel.Enabled = false;
    }
    foreach (var group in userGroups)
    {
        Log.Info("User Group: ",group.BrowseName);
        var mylabel_group =
Project.Current.Get<Label>("UI/MainWindow/Label4");
        mylabel_group.Text = group.BrowseName.ToString();
        //Make Navigation Panel visible if right usergroup is logged
        var panelPC = navigationPanel.Panels[1];
        var panelTrends = navigationPanel.Panels[2];
        var panelDatabase = navigationPanel.Panels[3];
        if (group.BrowseName.ToString()=="Technician")
        {
            navigationPanel.Enabled = true;
            panelPC.Enabled = false;
            panelTrends.Enabled = true;
            panelDatabase.Enabled = true;

        }
        else
    }
}

```

```

    {
        if (group.BrowseName.ToString() == "Administrator")
        {
            navigationPanel.Enabled = true;
            panelPC.Enabled = true;
            panelTrends.Enabled = false;
            panelDatabase.Enabled = true;
        }
        else
        {
            navigationPanel.Enabled = false;
        }
    }

}

//End custom behaviour*****

```

You can find the code here

[https://github.com/xavierflorensa/FTOptix\\_user\\_management\\_panels](https://github.com/xavierflorensa/FTOptix_user_management_panels)

And you can see the results on this video

<https://youtu.be/bp9QS3-gpKI>

## 51.12. Registering logged user on a database

First you have to store the logged user on a Variable and then use this variable for the datalogger, the drag and drop the Datalogger to adatagrid

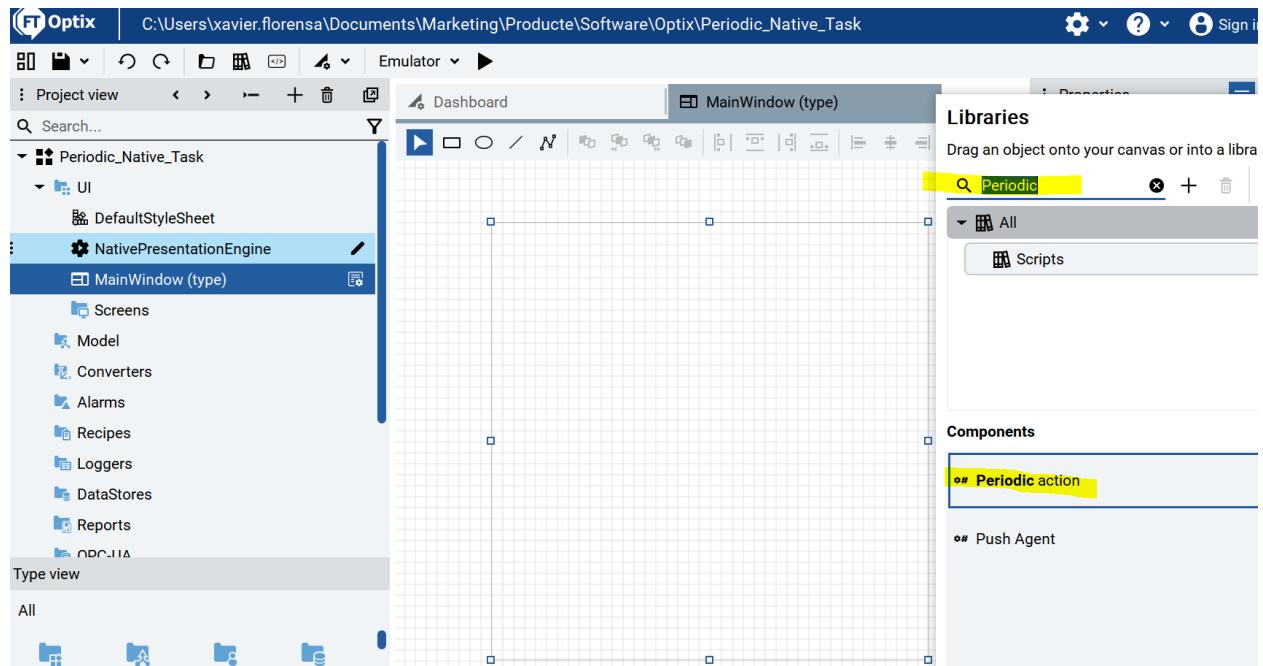
### 51.12.1. Adding the Logged user to a variable

as explained in the Periodic tasks section:

You have a library to do so

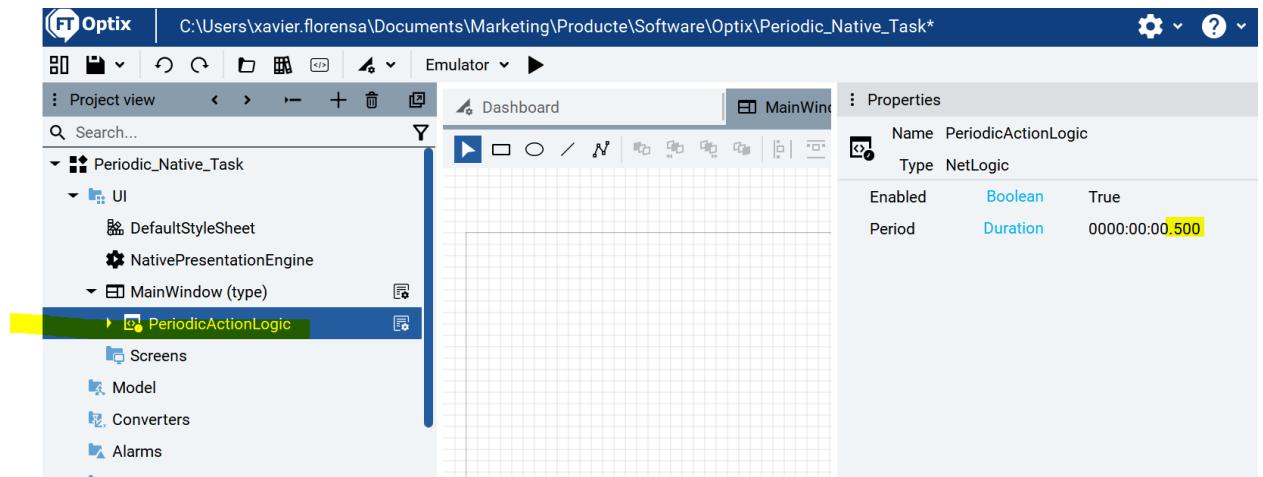
Let's create a blank project

Open library tool and search for Periodic

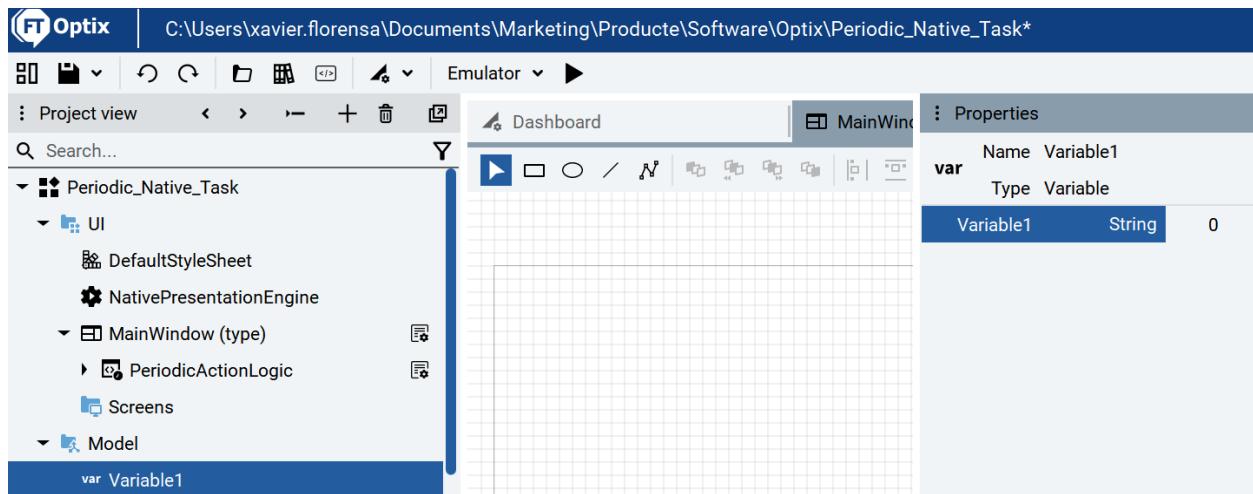


Drag and drop the “Periodic action” component to the desired scope, for instance, Main Window  
A new PeriodicActionLogic script will appear under Mainwindow (but you do not need to go to C#)

You have the period, for instance each 500 mseconds



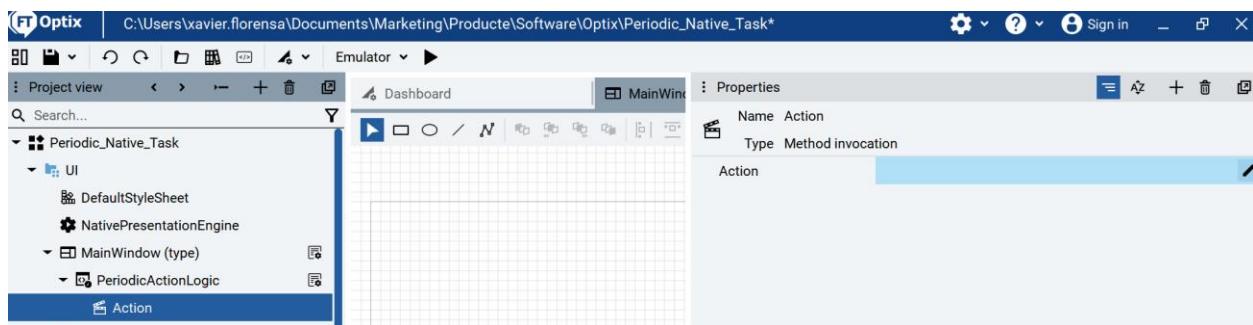
Then you can add an action, for instance modify a variable value  
Add a variable to Model



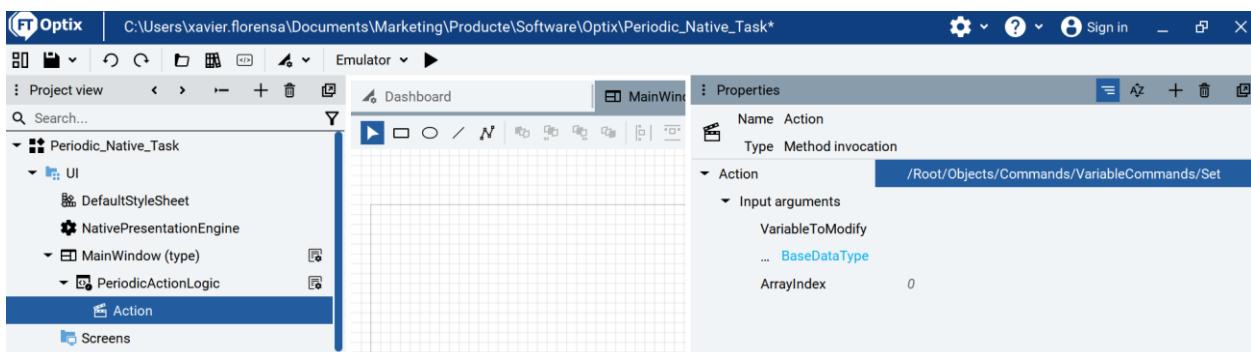
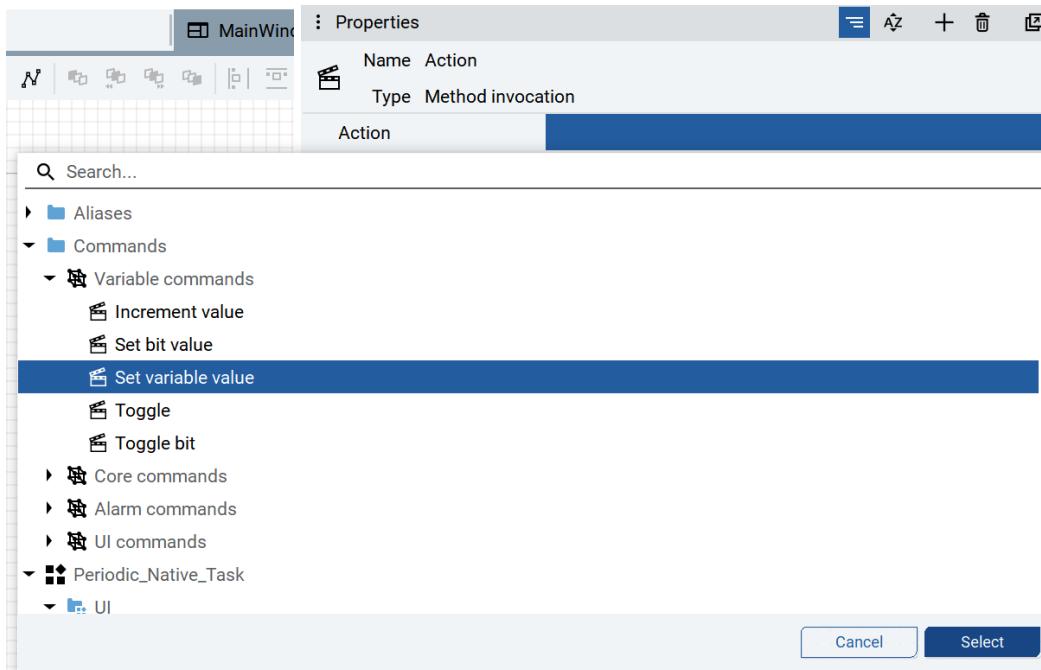
Then add an action to the Periodic

Click on Action

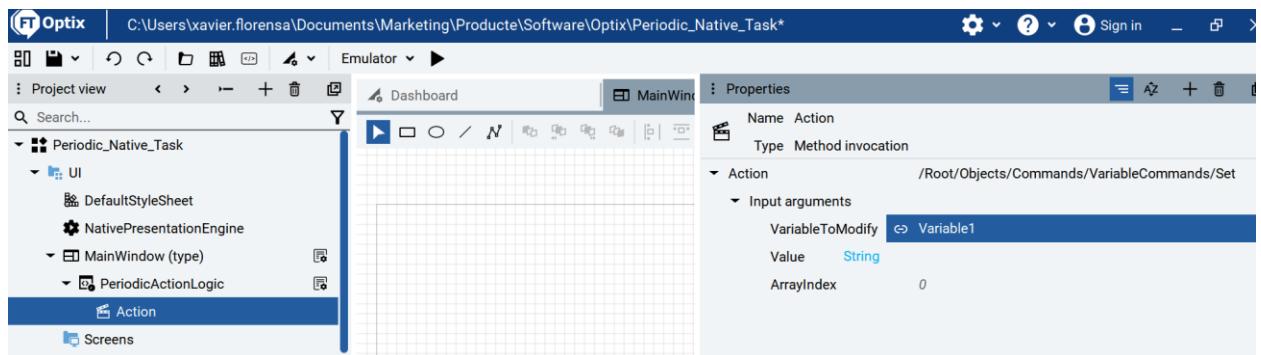
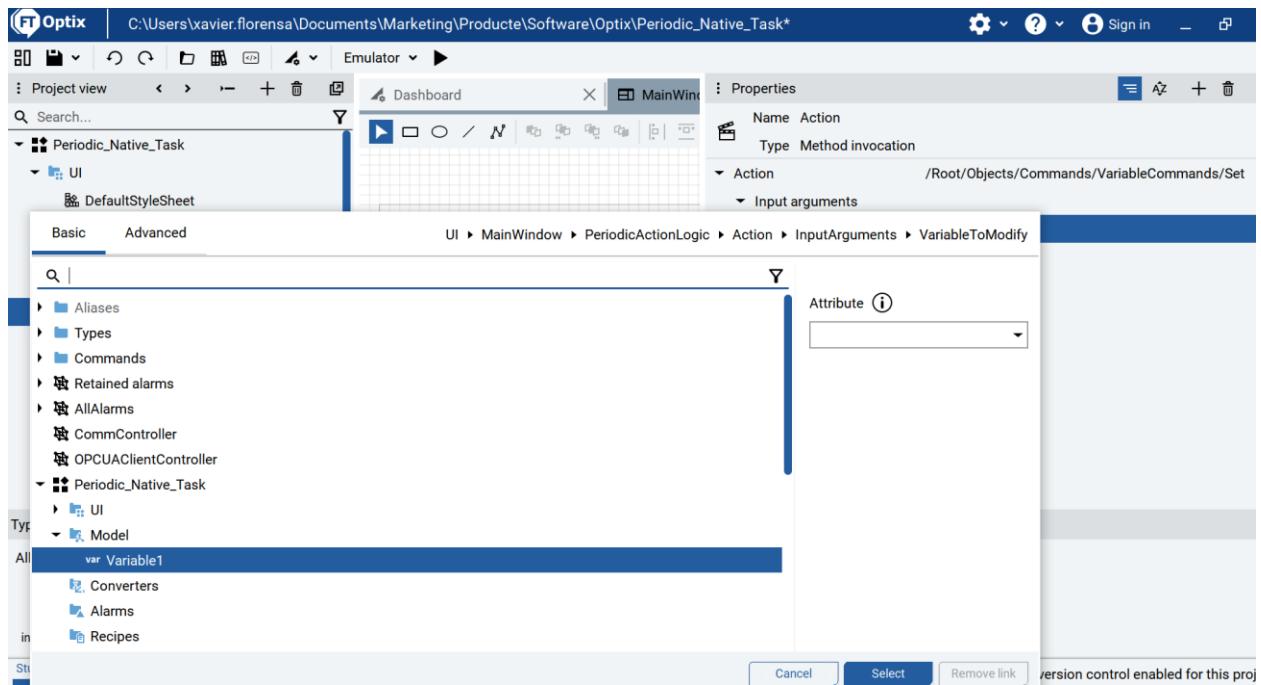
Click on the pen



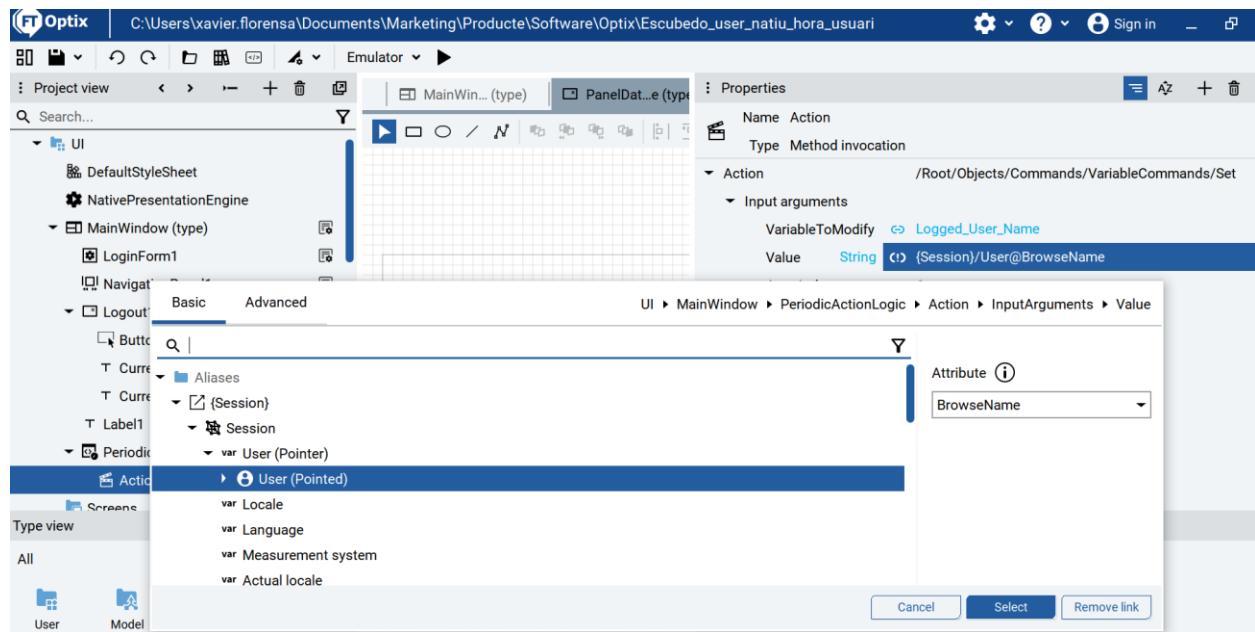
Add a command



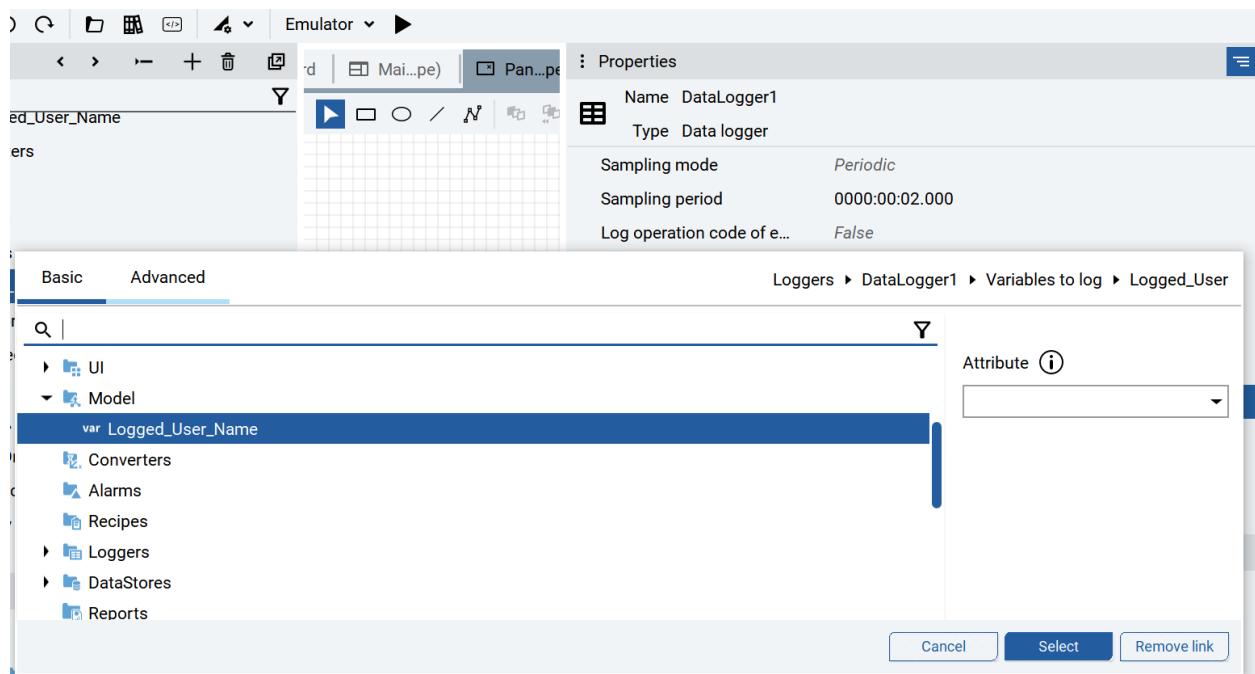
Set VariableToModify

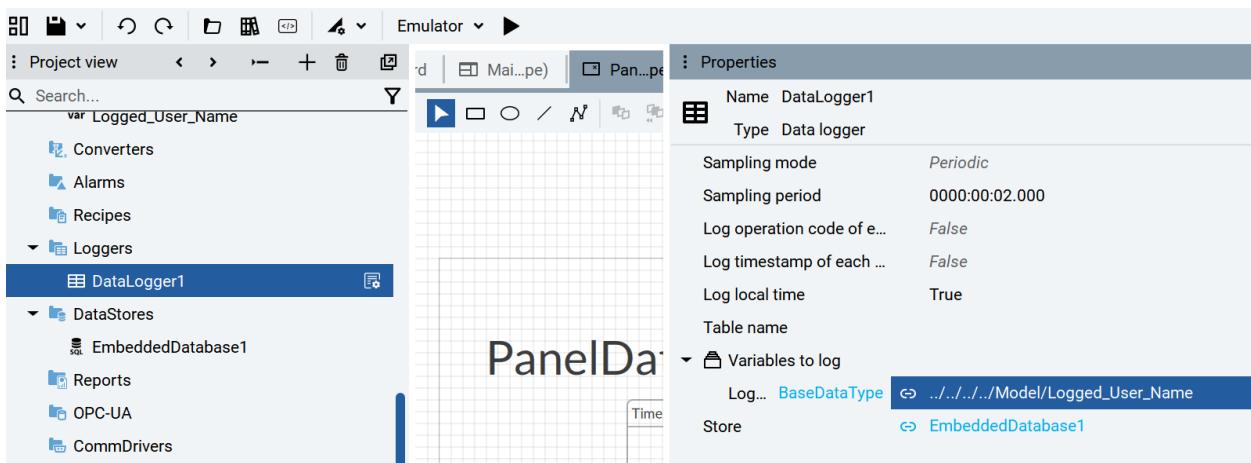


Select the value, for instance, logged user  
Click on the dynamic link on Value



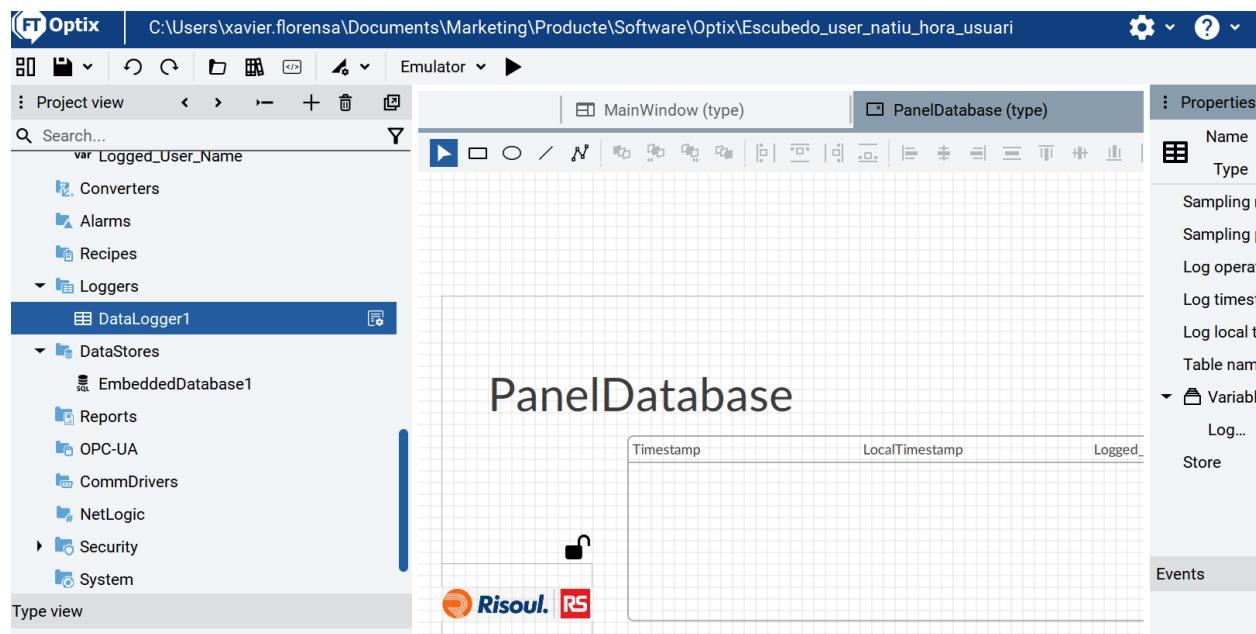
Then adjust the Variable to log on the Datalogger





This will record the user each 2 seconds but you can do on each variable change, on Sampling mode.

Finally, drag and drop the Dataloger to the newdatagrid on a desired panel.



That's all

[Logout](#)

Current User:  
Xavier

# PanelDatabase



Timestamp	LocalTimestamp	Logged
Sep 29, 2024, 7:04:56 AM	Sep 29, 2024, 9:04:56 AM	Xavier
Sep 29, 2024, 7:04:54 AM	Sep 29, 2024, 9:04:54 AM	Anonyr
Sep 29, 2024, 7:04:52 AM	Sep 29, 2024, 9:04:52 AM	Anonyr
Sep 29, 2024, 6:31:58 AM	Sep 29, 2024, 8:31:58 AM	Victor
Sep 29, 2024, 6:31:56 AM	Sep 29, 2024, 8:31:56 AM	Victor
Sep 29, 2024, 6:31:54 AM	Sep 29, 2024, 8:31:54 AM	Victor

You can find the code here

[https://github.com/xavierflorensa/FTOptix\\_logged\\_user\\_record.git](https://github.com/xavierflorensa/FTOptix_logged_user_record.git)

This is also explained in this video from RA Engage forum

<https://engage.rockwellautomation.com/HigherLogic/System/DownloadDocumentFile.ashx?DocumentFileKey=2d24f65e-70e0-437b-8e77-0190c6e4880c>

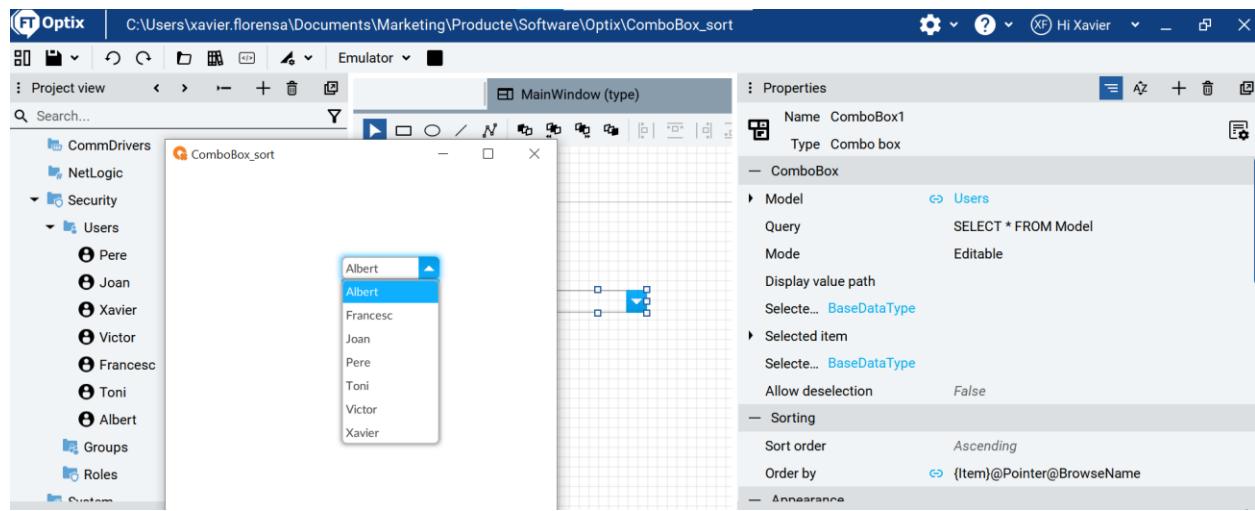
### 51.13. Alphabetical order on the Users ComboBox

Normally the users will appear on the User folder and on the ComboBox in the order they were created.

If you want to have them in Alphabetical order, you can use the Alias called {item} and point to {item}@BrowseName

<https://engage.rockwellautomation.com/discussion/username-in-alphabetical-order-on-combobox-when-login?ReturnUrl=%2fcommunities%2fcommunity-home%2fdigestviewer%3fListKey%3d13f227ed-c9e5-44bf-8301-493f3e399158#bm28484fce-d8c9-4e18-8b19-3a1529568deb>

This way



You can find the code here

[https://github.com/xavierflorencsa/FactoryTalk\\_Optix\\_User\\_ComboBox\\_Alphabetical\\_Sort.git](https://github.com/xavierflorencsa/FactoryTalk_Optix_User_ComboBox_Alphabetical_Sort.git)