

FactoryTalk Optix First steps

Contents

1.	Using web Help	3
2.	Using integrated help.....	3
3.	Configuring Tags from PLC access.....	3
4.	Displaying a PLC Tag.....	9
5.	Configuring an application as an MQTT client.....	14
6.	PLC data and MQTT.....	25
7.	OPC UA Client.....	45
8.	Writing to an SQL database	52
9.	Creating a OPC UA to SQL converter	74
10.	Subscribing to an MQTT broker with user and password. TTN example.....	80
11.	Web access.....	85
12.	Navigation panels.....	86
13.	Creating an Http REST API client	92
13.1.	GET request	92
13.2.	Installing InfluxDB on windows	104
13.3.	Inject your first data with node-red	107
13.4.	Injecting to InfluxDB from command line	109
13.5.	POST request to local InfluxDB.....	112
13.6.	POST request to remote InfluxDB	115
14.	FT Optix communicating with Micro850.....	117
15.	Using Github to store and share your repositories on the cloud	122
16.	Pushing your local Optix projects to Github	133
16.1.	Using Optix	133
16.2.	Using Git	148
16.3.	Remote repository creation with FT Optix version 1.3	160
17.	Sending Telegram messages	162
18.	Modbus TCP to OPC UA converter.....	162
19.	Modbus TCP to EtherNet/IP Gateway	163
20.	Modbus to MQTT data aggregator	164
21.	Creating, importing C# .Net libraries to your environment.....	165

21.1.	Design time libraries	165
21.2.	Runtime libraries	172
22.	First steps with OptixPanel	201
23.	NetLogic and C# tutorial	207
24.	Understanding compilation of NetLogic code	207
25.	Understanding Classes (Objects), methods and data	211
26.	Creating a Method	212
27.	Linking events with Methods without input parameters	216
28.	Linking events with Methods with input parameters.....	219
29.	Asynchronous tasks, creating callback functions.....	222
30.	Log info to emulator output	227
31.	Accessing variables	227
32.	Accessing objects	234
33.	Using Owner.....	240
33.1.	Accessing object using owner.....	240
33.2.	Detecting change on object using Owner	242
34.	Object events	244
35.	Open a cmd terminal to run a command or program	250
36.	Socket TCP using System.Net.Sockets without external libraries.....	256
37.	How to define NetLogic Parameters.....	292
38.	Using third party dll libraries in FactoryTalk Optix	293
38.1.	Using third party libraries in Visual Studio 2022 windows Forms Framework	293
38.2.	Thirt party libraries (dll) Socket Server application with Optix	311
38.3.	Thirt party libraries (dll) Socket Client application with Optix	322
38.3.1.	Installing external references (dll libraries) into the project Visual Studio 2022	327
38.3.2.	Installing external references (dll libraries) into the project with Visual Studio code ...	335
39.	Using NuGet libraries	337
40.	Modbus server with FTOptix.....	349

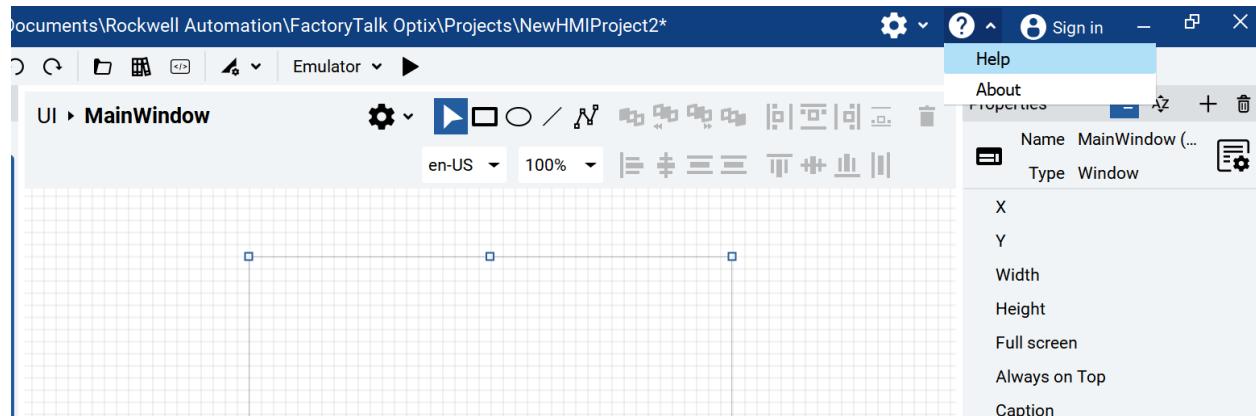
First steps with FT Optix

1. Using web Help

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

The screenshot shows a web browser window with the URL <https://www.rockwellautomation.com/docs/en/factorytalk-optix/1-00/contents-ditamap.html>. The page has a header with the Rockwell Automation logo and a search bar. Below the header is a section titled "Using the software" containing a 4x3 grid of buttons. The buttons represent various software components: Projects, Version control, Communication driver, OPC UA; Graphic and layout objects, Libraries, Manage project content, Trends; Dynamic links, Converters, Events, Aliases; Alarms, Recipes, Logger, DataStore (database); Reports, Users and groups, Translations, Audit signing. A vertical blue sidebar on the right is labeled "Help & Feedback".

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

← → ⌂ Archivo | C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/getting-started/quick-start/import-tag-variables.html ⌂ ⌂ ⌂ ⌂

Rockwell Node-RED : node-r... TTN Login TTN

- > Introduction
- > Basic concepts
- ✓ Quick start: develop a sample project
 - Create a project
 - > Configure and brand the main window
 - > Configure panels
 - > Configure dynamic graphic objects
 - > Configure variables
 - Create variables
 - Import controller variables
 - Configure temperature controls
 - > Configure alarms
 - > Configure recipes
- Save and commit changes

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

1. From the FactoryTalk Optix Studio toolbar, select [Open dashboard page](#).
2. In the central pane, select [I want to configure connected devices](#).
3. Select [New stations](#).
4. Select [RA EtherNet/IP Station](#) and select [Next](#).
5. (optional) To import the controller tags in the online mode, in [Route](#), enter [IP_Address\Backplane\Chassis_Slot_Number](#) and select [Next](#).
6. Select [Next](#).
7. Fetch the controller tags:
 - o To fetch the controller tags in the offline mode, select [Browse](#) and select the downloaded [LogixTags.ACD](#) file.
 - o To fetch the controller tags in the online mode, select the [Offline/Online](#) toggle to change it to the [Online](#) position.
8. 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

FT Optix | C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\PLC2MQTTClient

(F)

<

Home

+

New

Open

Close Project

Create a new project

Select a project configuration type to begin.



Default

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

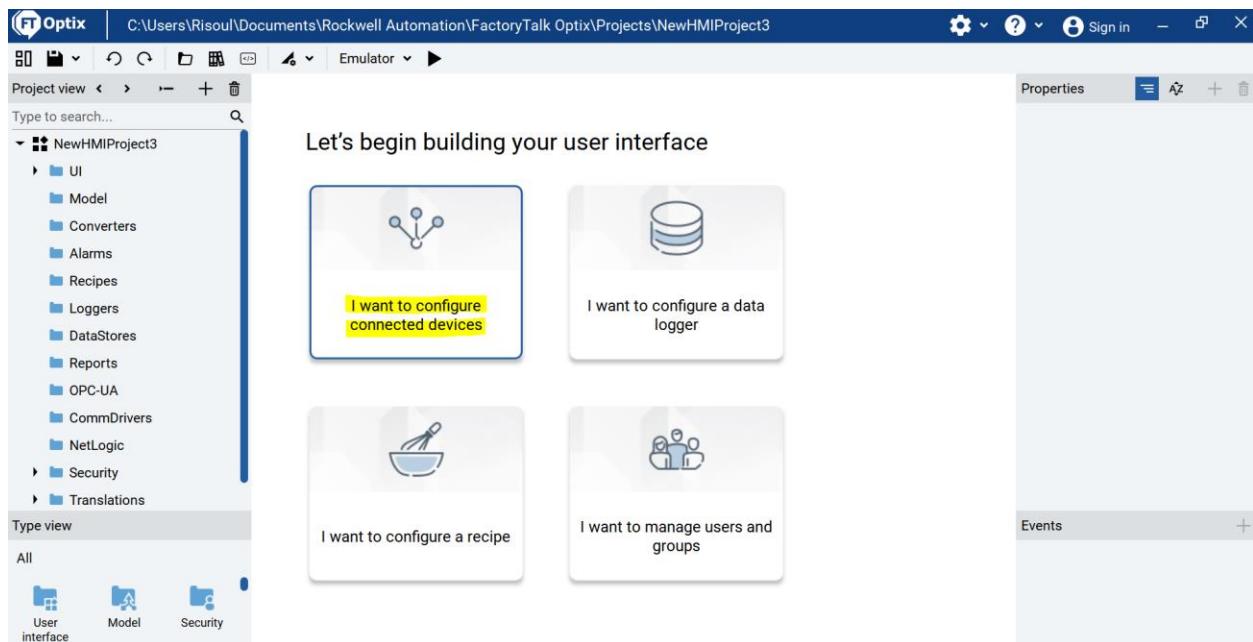


Blank

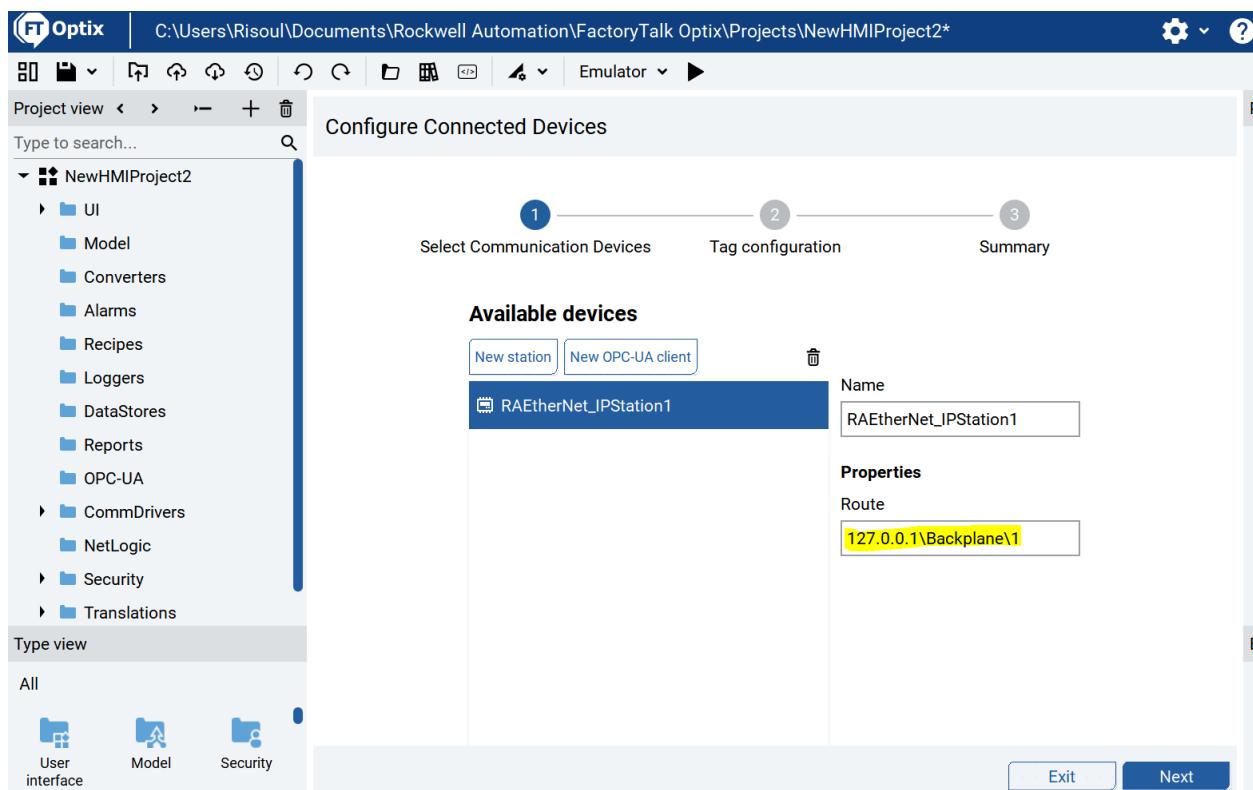
Creates an empty project with no predefined content.

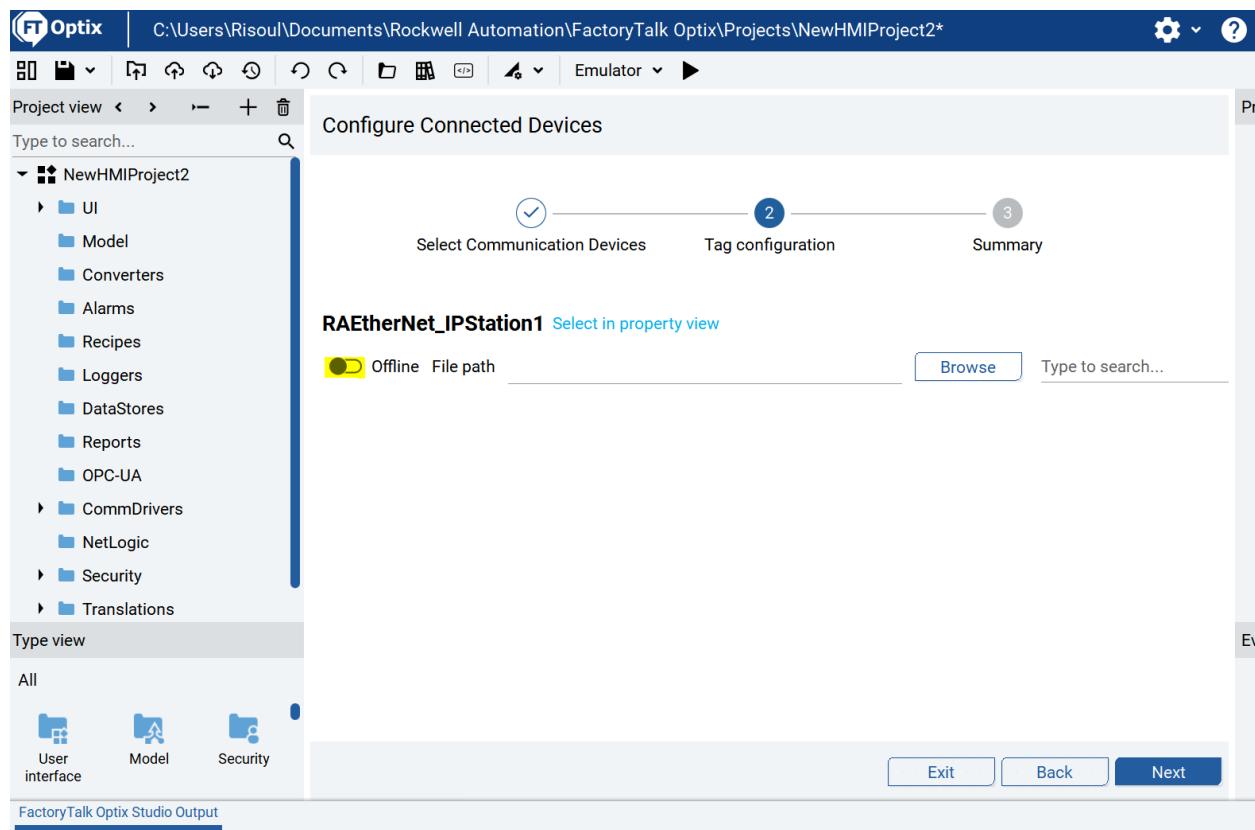
Recent projects

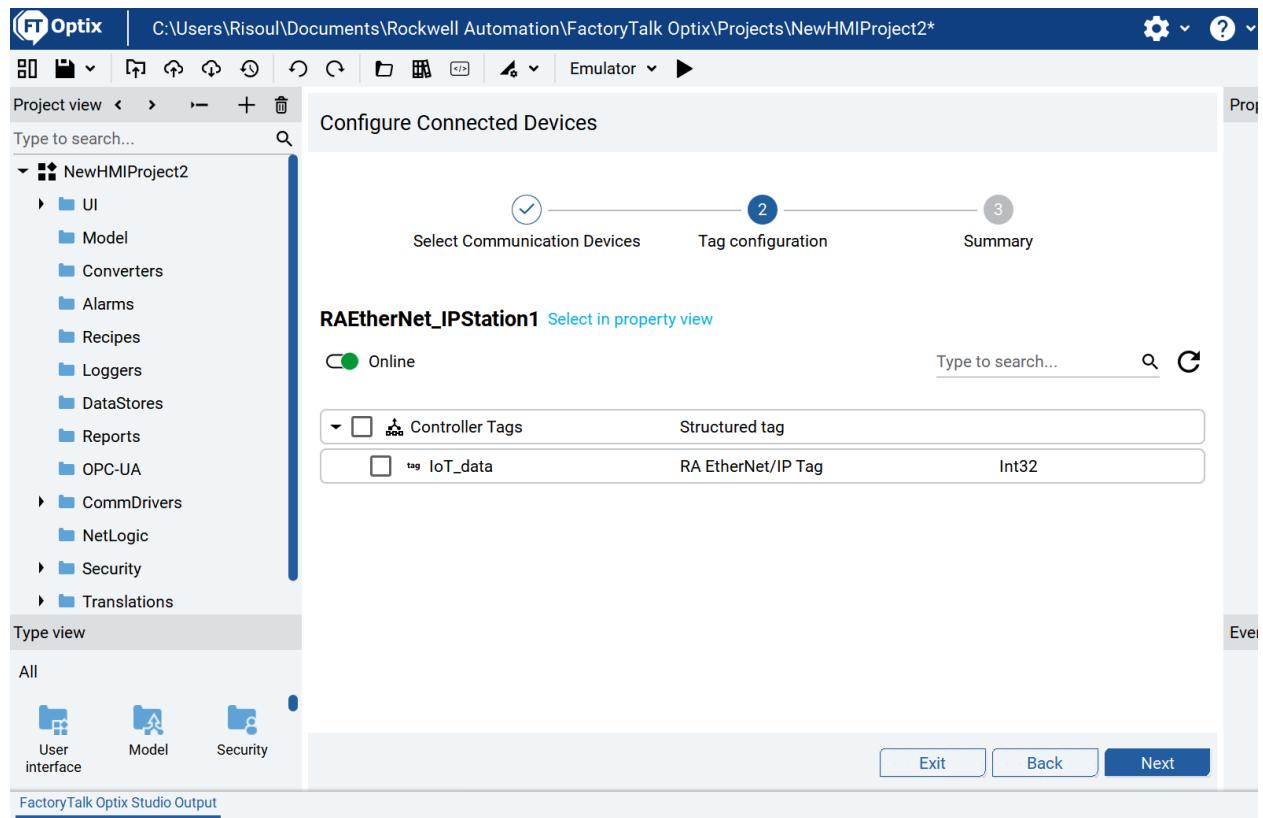
Type to search... Q



On both cases you will arrive to this point







Configure Connected Devices



RAEtherNet_IPStation1 [Select in property view](#)

Online

Type to search...

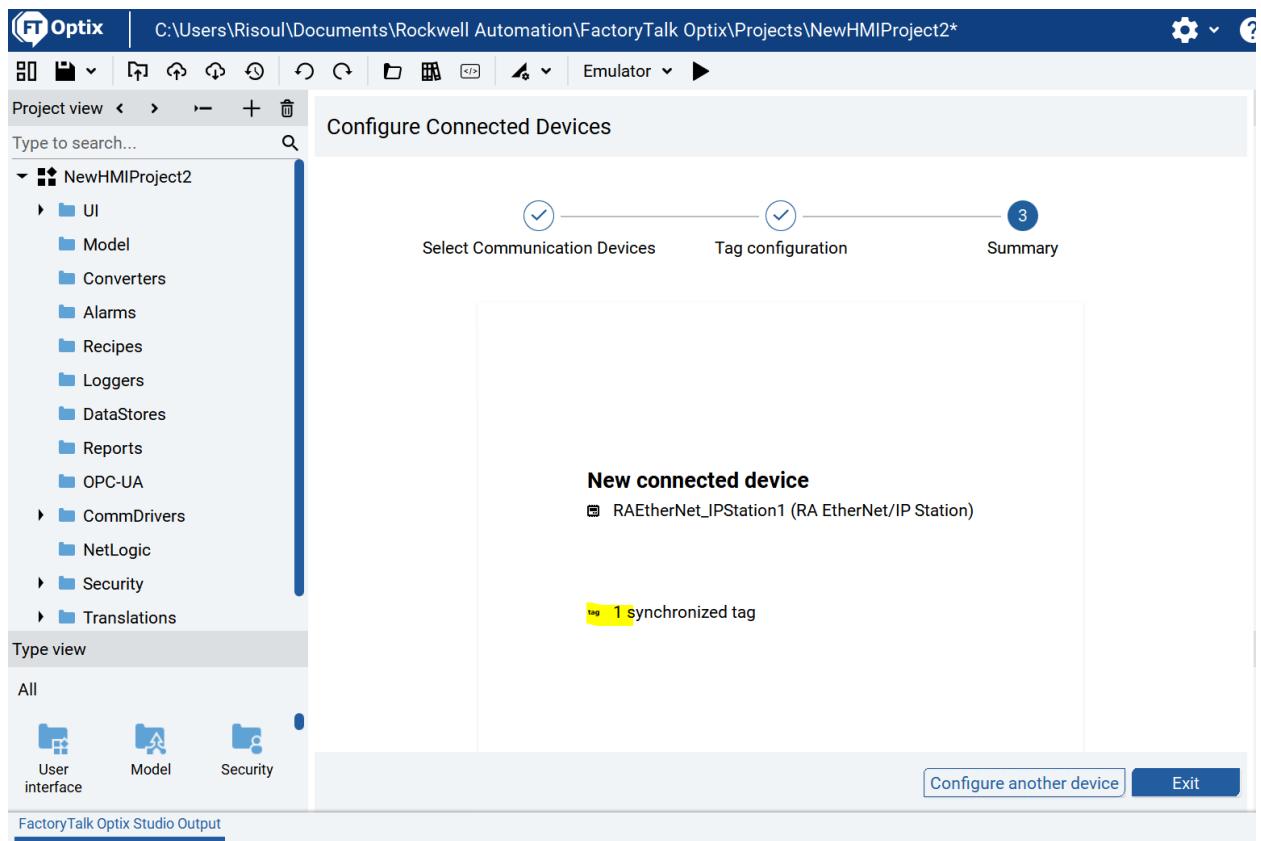


<input checked="" type="checkbox"/> Controller Tags	Structured tag
<input checked="" type="checkbox"/> IoT_data	RA EtherNet/IP Tag

[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>

← → C Archivo | C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/getting-started/quick-start/Configure-the-main-wind... 🔍 ☆ ⚙

Rockwell Node-RED : node-r... TTN Login TTN

Rockwell Automation

FactoryTalk Optix Studio Help Center

- Getting started
 - Introduction
 - Basic concepts
 - Quick start: develop a sample project
 - Create a project
 - Configure and brand the main window
 - Configure the main window

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

FT Optix | C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\NewHMIProject2

Project view Type to search... Window Emulator

NewHMIProject2

- UI
 - DefaultStyleSheet
 - NativePresentationEn...
- MainWindow (type)
 - Label1

Edit with UI editor

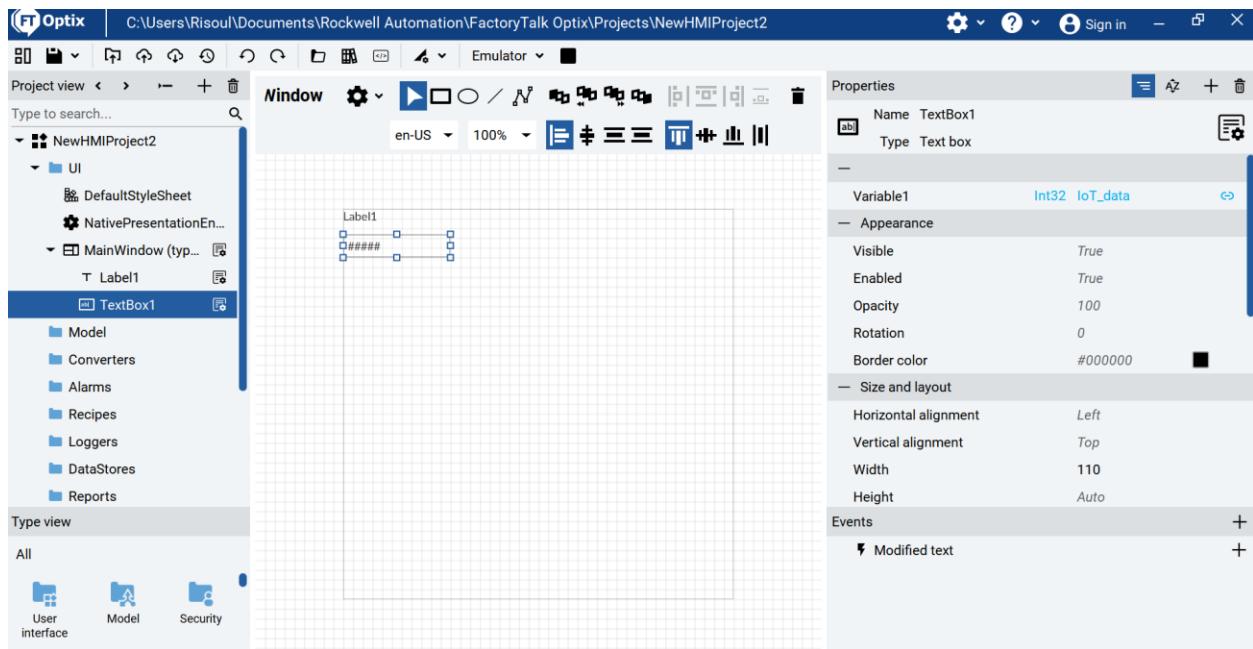
New Base controls T Label

- Copy Containers ▶ Text box
- Paste Contents ▶ Spin box
- Delete Data controls ▶ Editable label
- Rename Drawings ▶ LED
- Collapse all Native presentation engine ▶ Button
- Show References Web presentation engine ▶ Switch
- Show Instances Style sheet ▶ Check box
- Add custom behavior UI Session ▶ Option button
- Model Linear gauge
- ## Runtime NetLogic Circular gauge
- ## Design-time NetLogic Date and time
- Method invocation Duration

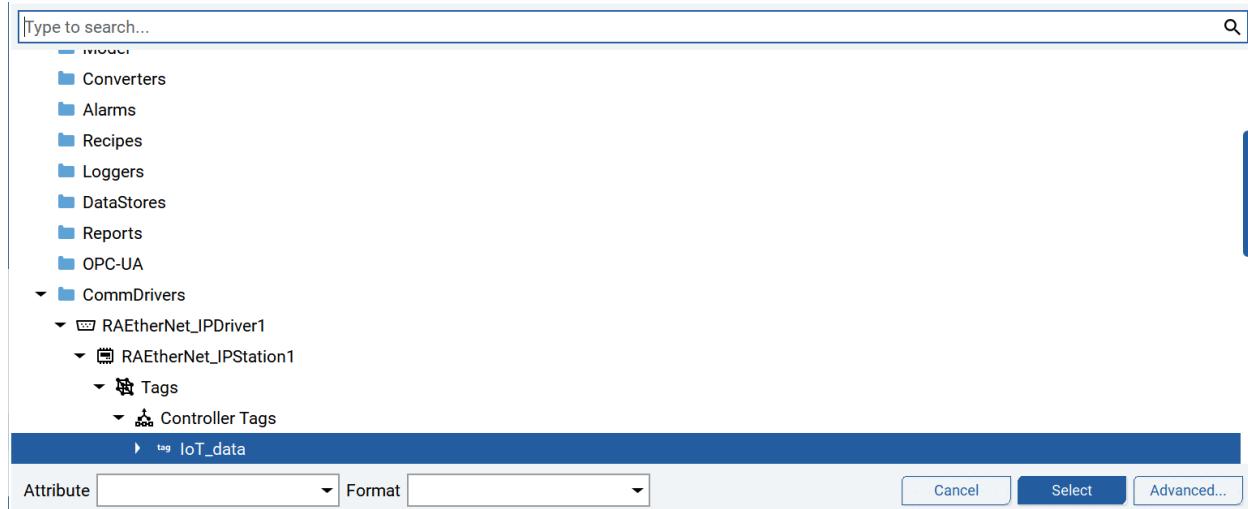
Type view All

User Model Security

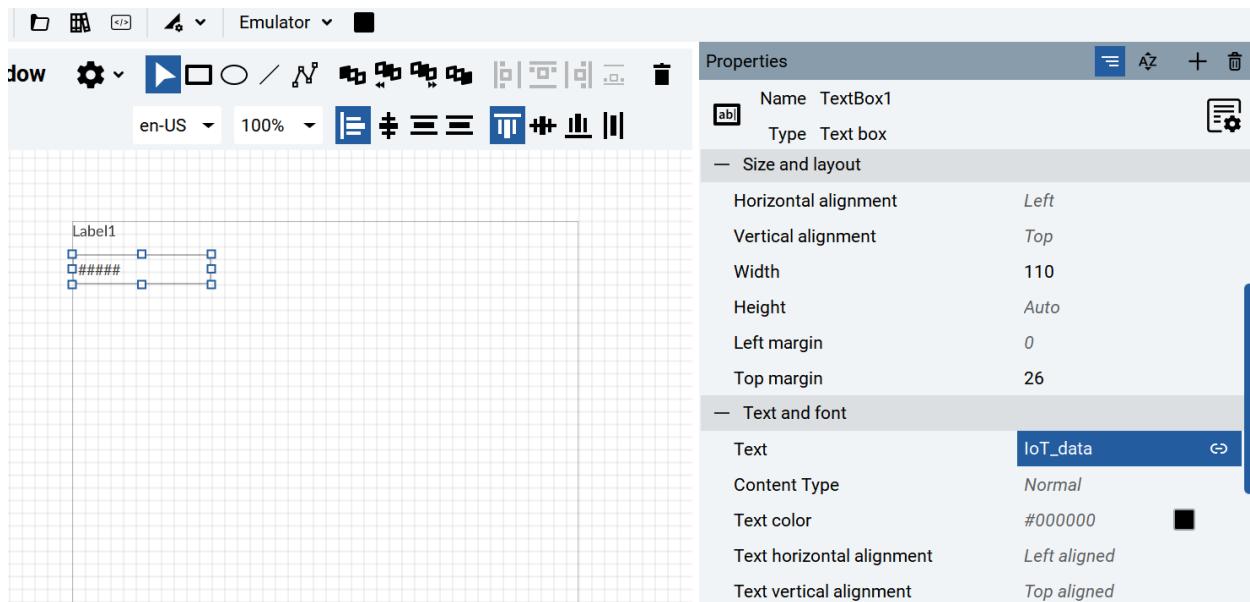
Double click on the object on the page



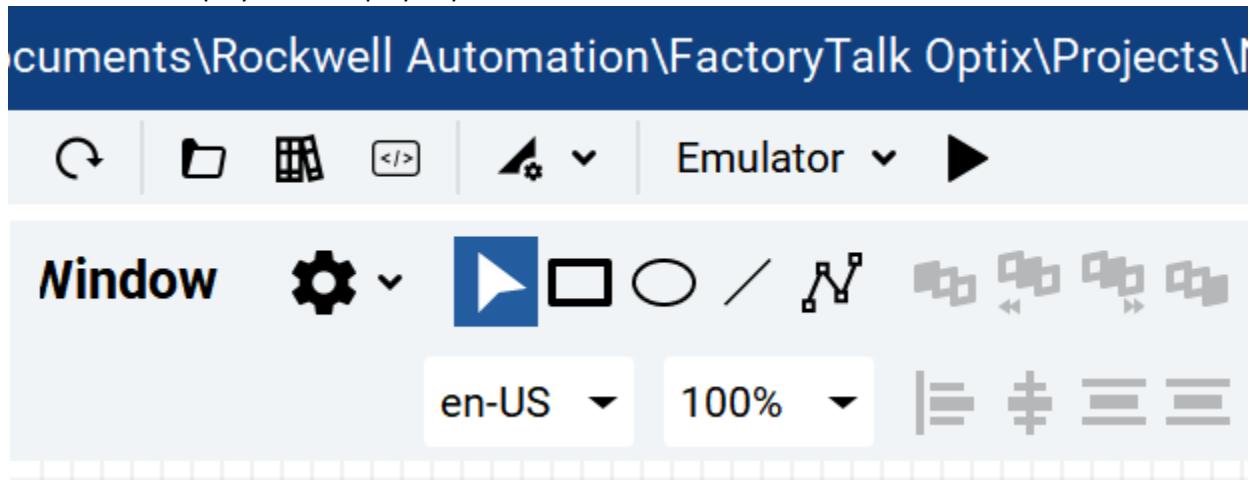
Go down in the properties until Text, and search for the Tag



Select, you will see this



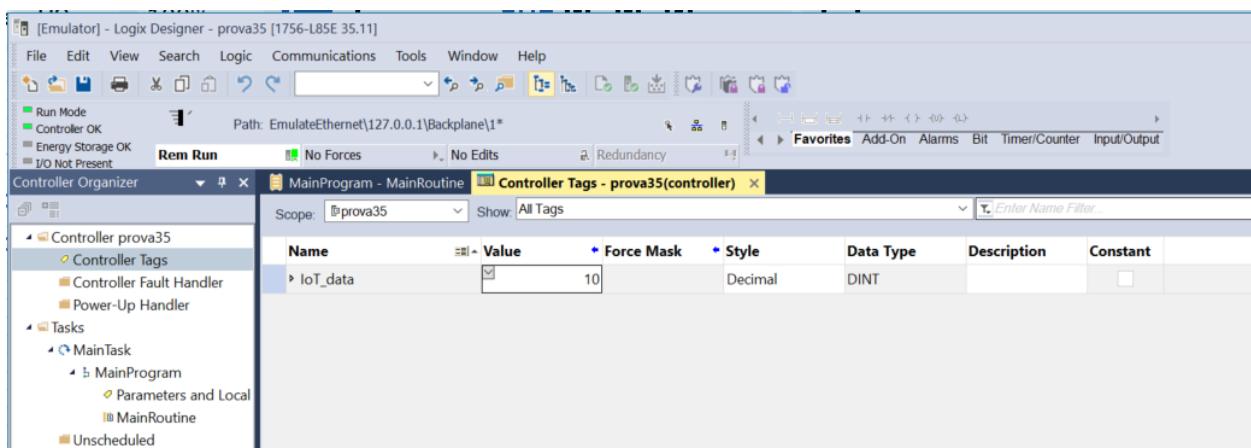
Let's test the display with the play key:



You will see this



Let's change the Tag value on Studio5000



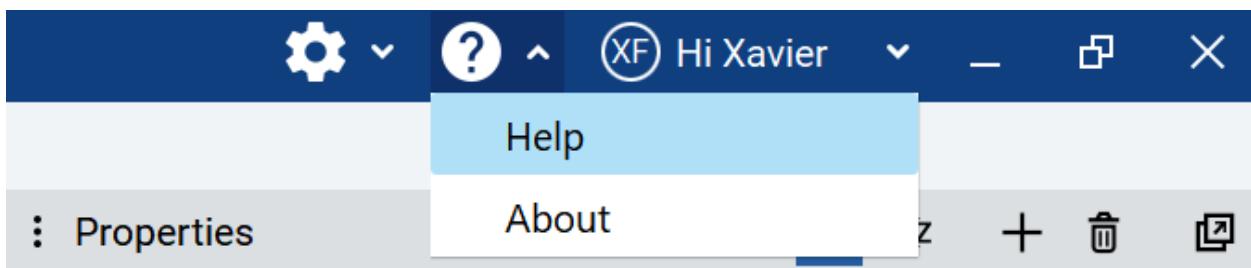


5. Configuring an application as an MQTT client

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 the Rockwell Automation FactoryTalk Optix Studio Help page. The main content area is divided into five sections:

- Getting started**: Become familiar with the terminology, design concepts, and workflows present in the FactoryTalk Optix™ environment and create your first project.
- Creating projects**: Create a project and your application logic.
- Extending projects**: Use C# to implement custom functionalities into your application logic.
- Deploying projects**: Compile projects to FactoryTalk Optix Applications and then deploy the applications to client devices.
- FactoryTalk Optix Release Notes**: View system features, system requirements, and application notes for visualization projects.

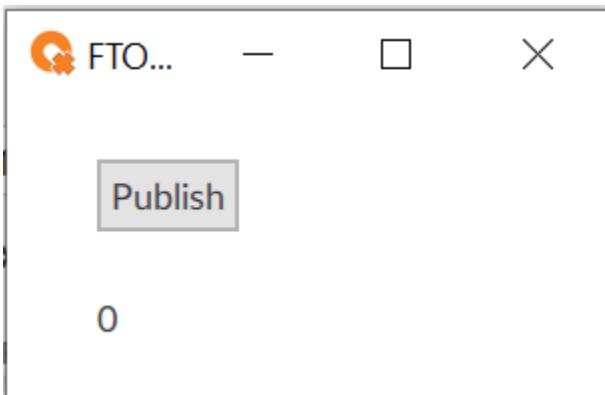
The screenshot shows the Rockwell Automation FactoryTalk Optix Studio Help page, specifically the "Creating projects" section. On the left, there is a navigation sidebar with the following structure:

- > Translations
- > Audit signing
- > Retentivity storage
- > FTP
- < IoT
 - > Push Agent
 - > Brokers
 - < IoT tutorial
 - < Configure an application as an MQTT client
 - Configure the message broker IP
 - Develop the publisher NetLogic and interface
 - Develop the subscriber NetLogic and interface

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

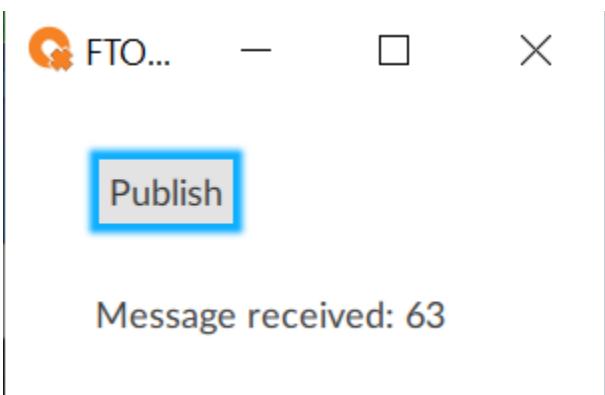
Tip: You can download a sample project from: [MQTTClient.zip](#)

Let's try opening the example



Click on "Publish"

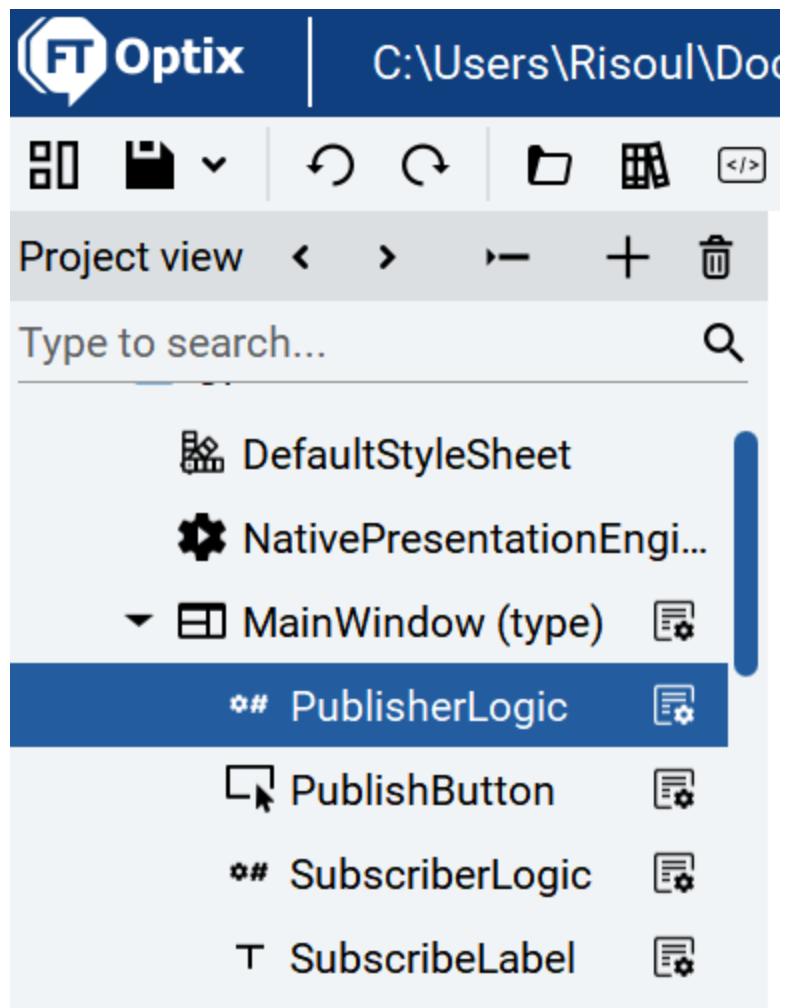
It works



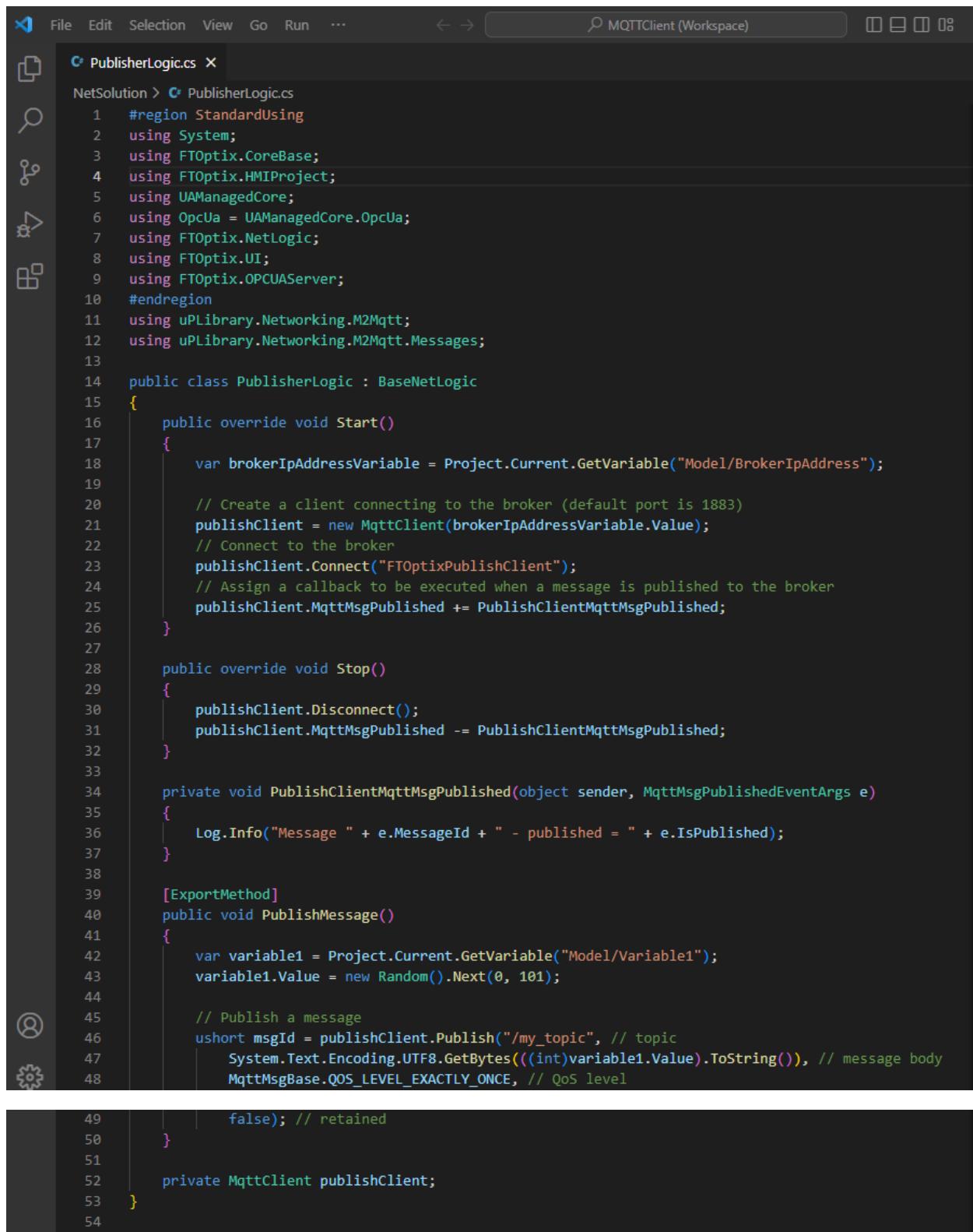
But let's look at the details

Like PublisherLogic

Doubleclick on PublisherLogic



Visual Studio code opens

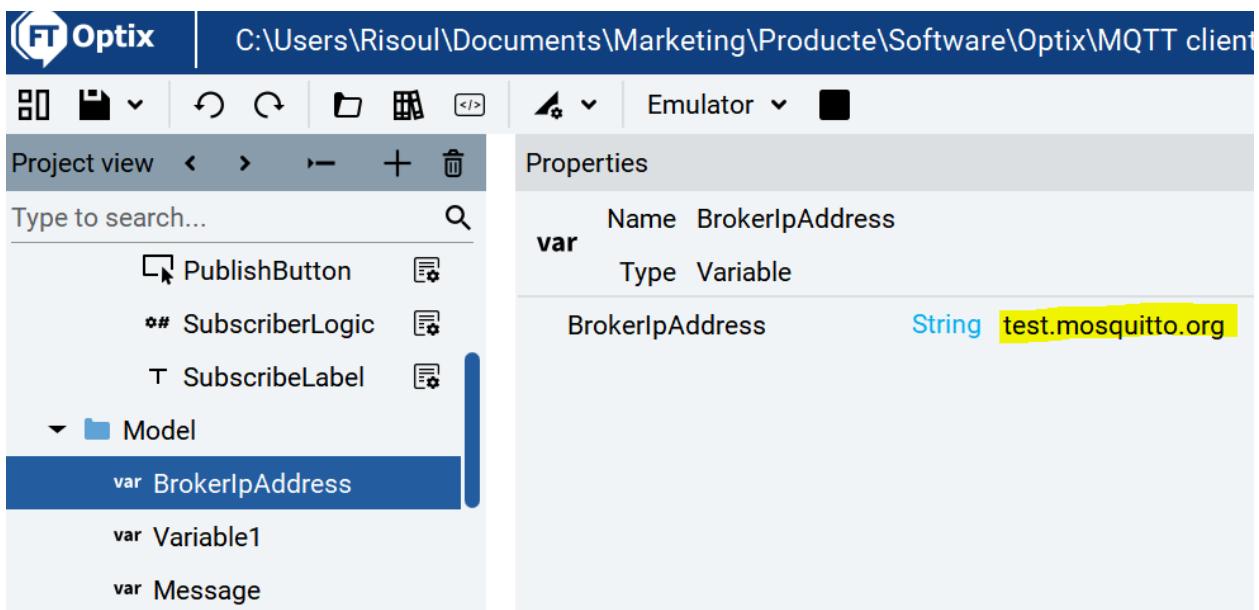


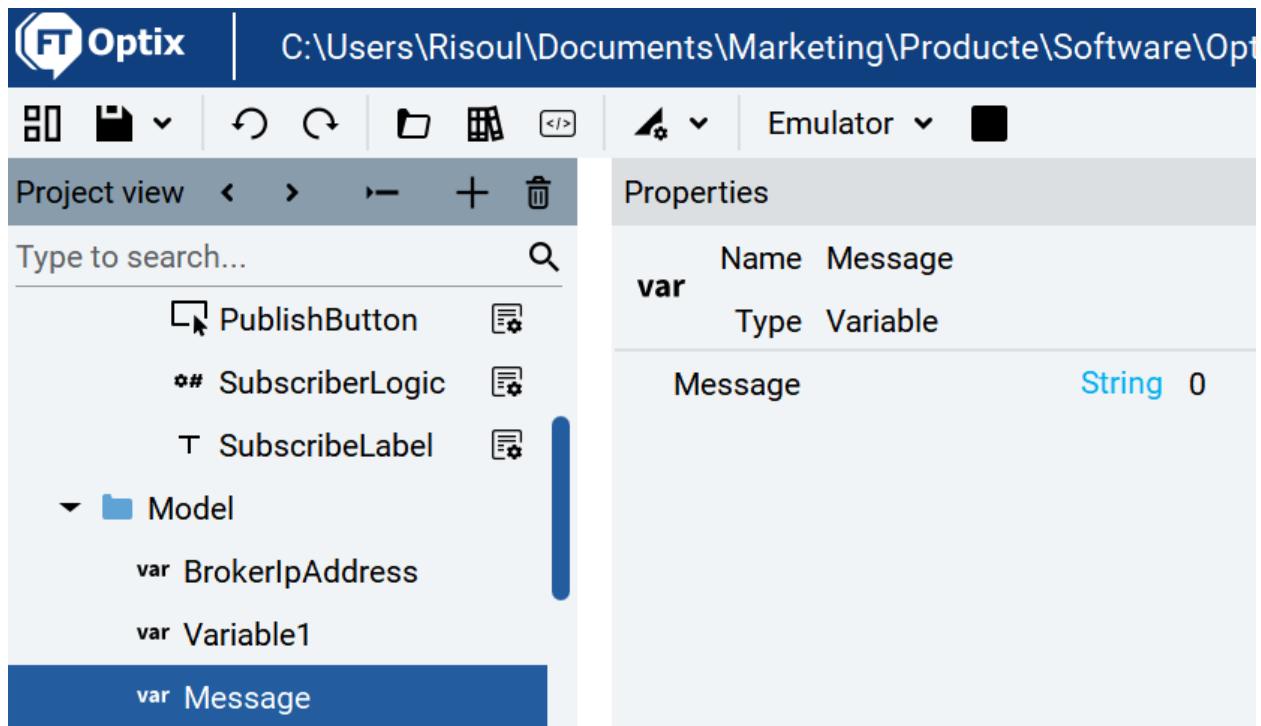
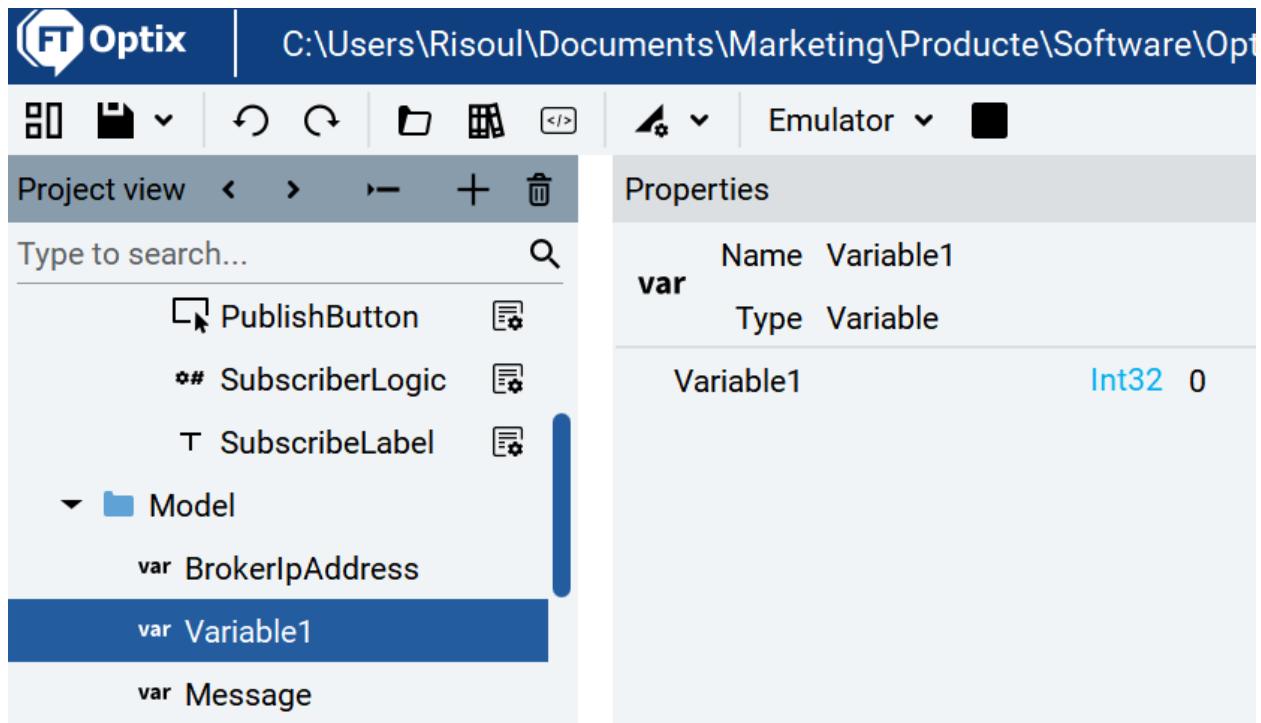
```
NetSolution > C# PublisherLogic.cs
1  #region StandardUsing
2  using System;
3  using FTOptix.CoreBase;
4  using FTOptix.HMIPProject;
5  using UAManagerCore;
6  using OpcUa = UAManagerCore.OpcUa;
7  using FTOptix.NetLogic;
8  using FTOptix.UI;
9  using FTOptix.OPCUAServer;
10 #endregion
11 using 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

```
▼ └ Model
    var BrokerIpAddress
    var Variable1
    var Message
└ Converters
```





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

```

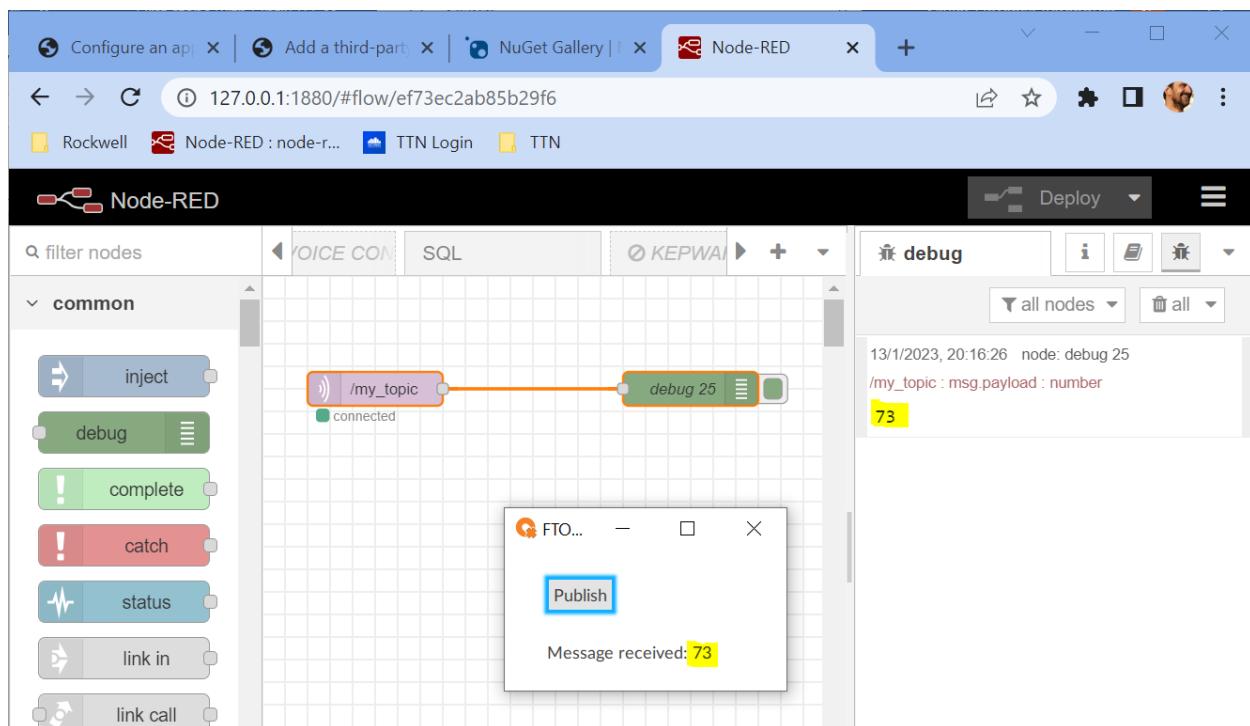
[ExportMethod]
public void PublishMessage()
{
    var variable1 = Project.Current.GetVariable("Model/Variable1");
    variable1.Value = new Random().Next(0, 101);

    // 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

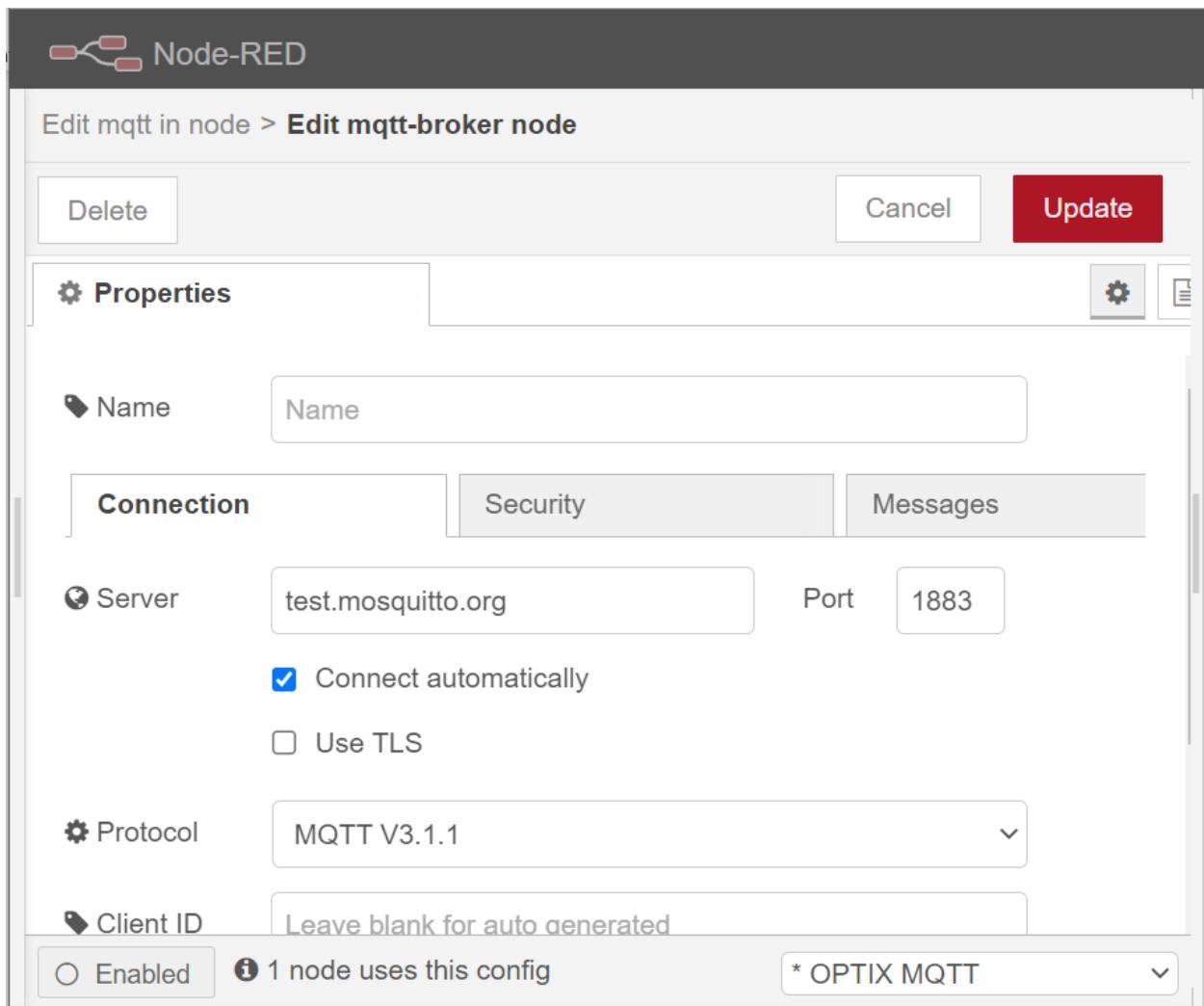


Output



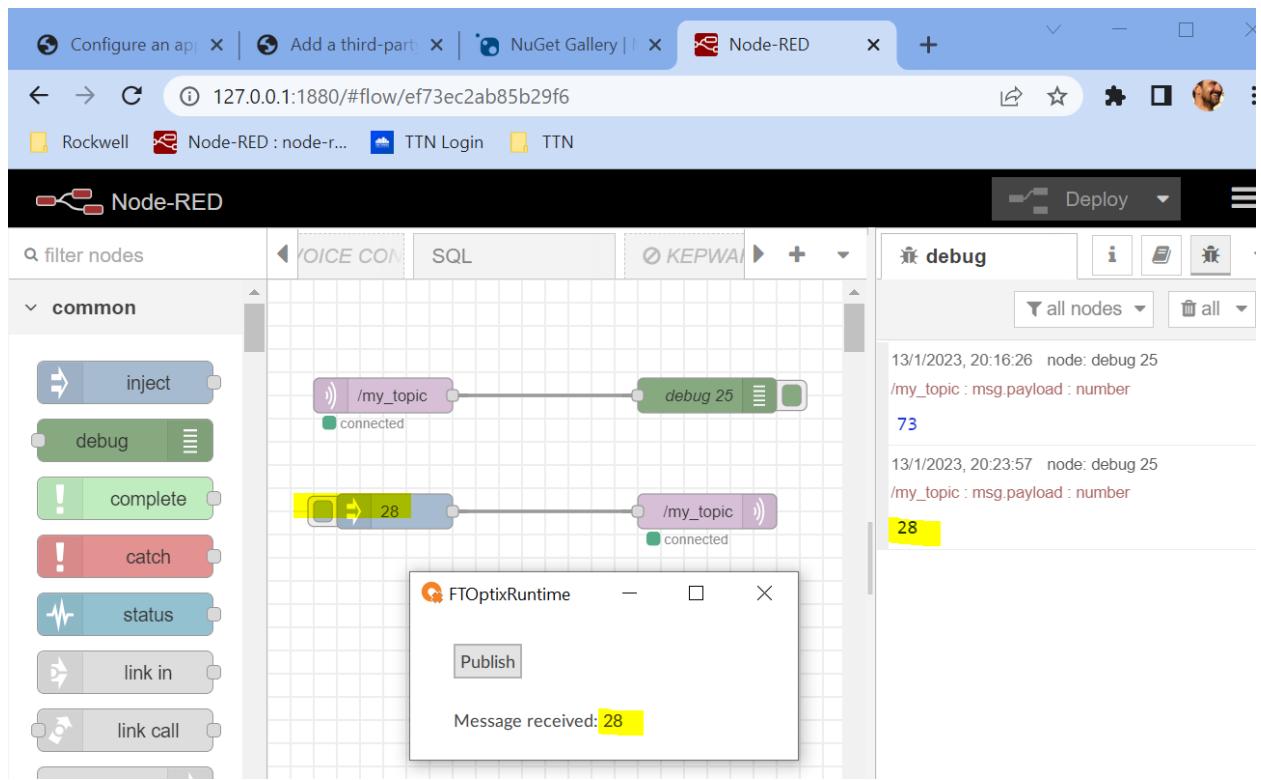
Name

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

Topic: /my_topic

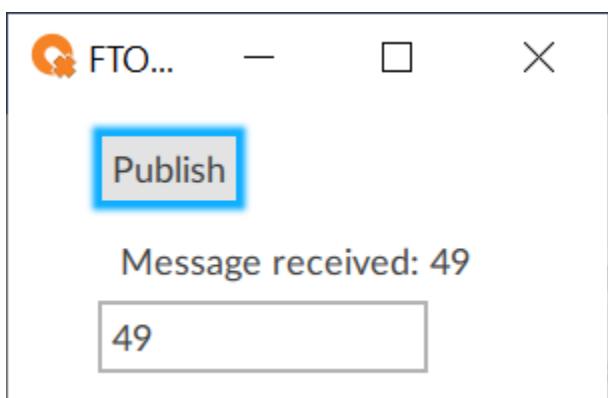
QoS: Retain:

Name: Name

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

Enabled

The dialog box is titled "Edit mqtt out node". It contains a "Properties" section with fields for "Server" (set to "test.mosquitto.org:1883"), "Topic" ("/my_topic"), "QoS" (a dropdown menu), "Retain" (a dropdown menu), and "Name" (a text input field). Below the properties is a yellow tip box containing the text: "Tip: Leave topic, qos or retain blank if you want to set them via msg properties.". At the bottom is a radio button group for "Enabled".



Next step is to use PLC data

6. PLC data and MQTT

This is what we will do on this example



You can find the code here

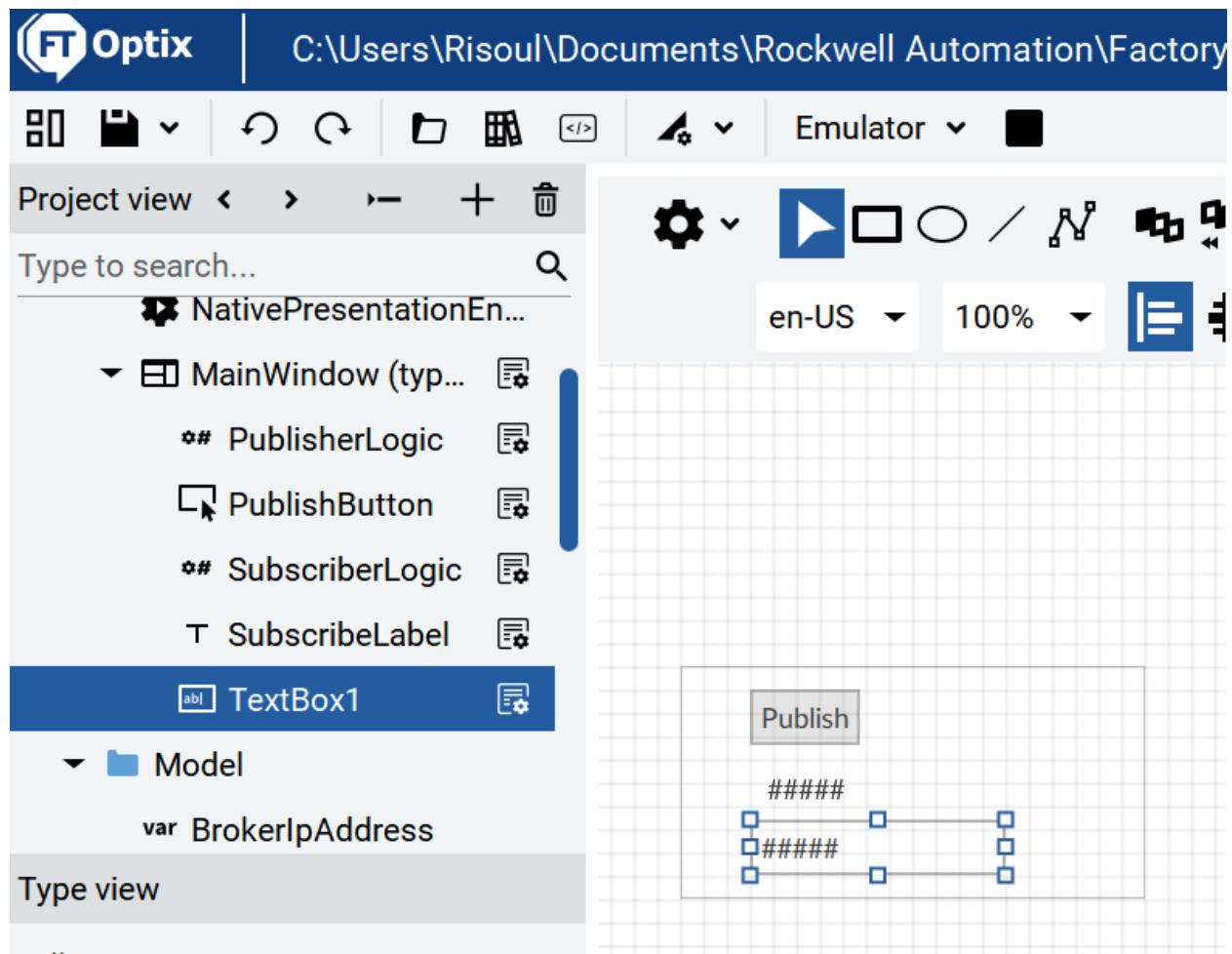
https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git

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

A screenshot of the "Type view" properties dialog. The search bar at the top contains the text "var". The list of properties includes "Aliases", "Types", "Commands", "Retained alarms", "CommController", "OPCUAClientController", "PLC2MQTTClient", "UI", "Model" (selected), "BrokerIpAddress", "Variable1" (selected), "Message", "Converters", and "Attribute". The "Variable1" property is highlighted with a blue selection bar. At the bottom of the dialog are buttons for "Cancel", "Select", and "Advanced...".

Like this

The screenshot shows the 'Properties' panel of the FactoryTalk Optix Studio interface. At the top, there are navigation icons: gear, question mark, sign in, minimize, maximize, and close. Below the title 'Properties' are buttons for sorting (list, A-Z, +, trash).

Component Properties:

- Name: TextBox1
- Type: Text box

Dimensions:

- Height: Auto
- Left margin: 30
- Top margin: 66

Text and font settings:

- Text: Variable1 (highlighted)
- Content Type: Normal
- Text color: #000000 (black square icon)
- Text horizontal alignment: Left aligned

Events:

- Modified text (with a lightning bolt icon)

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 (which is expanded), Configure and brand the main window, Configure panels, Configure dynamic graphic objects, Configure variables (expanded), Create variables, Import controller variables, 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

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

Project view < > - + Type to search... Search icon

Emulator ▾ ▶

UI ▶ MainWindow

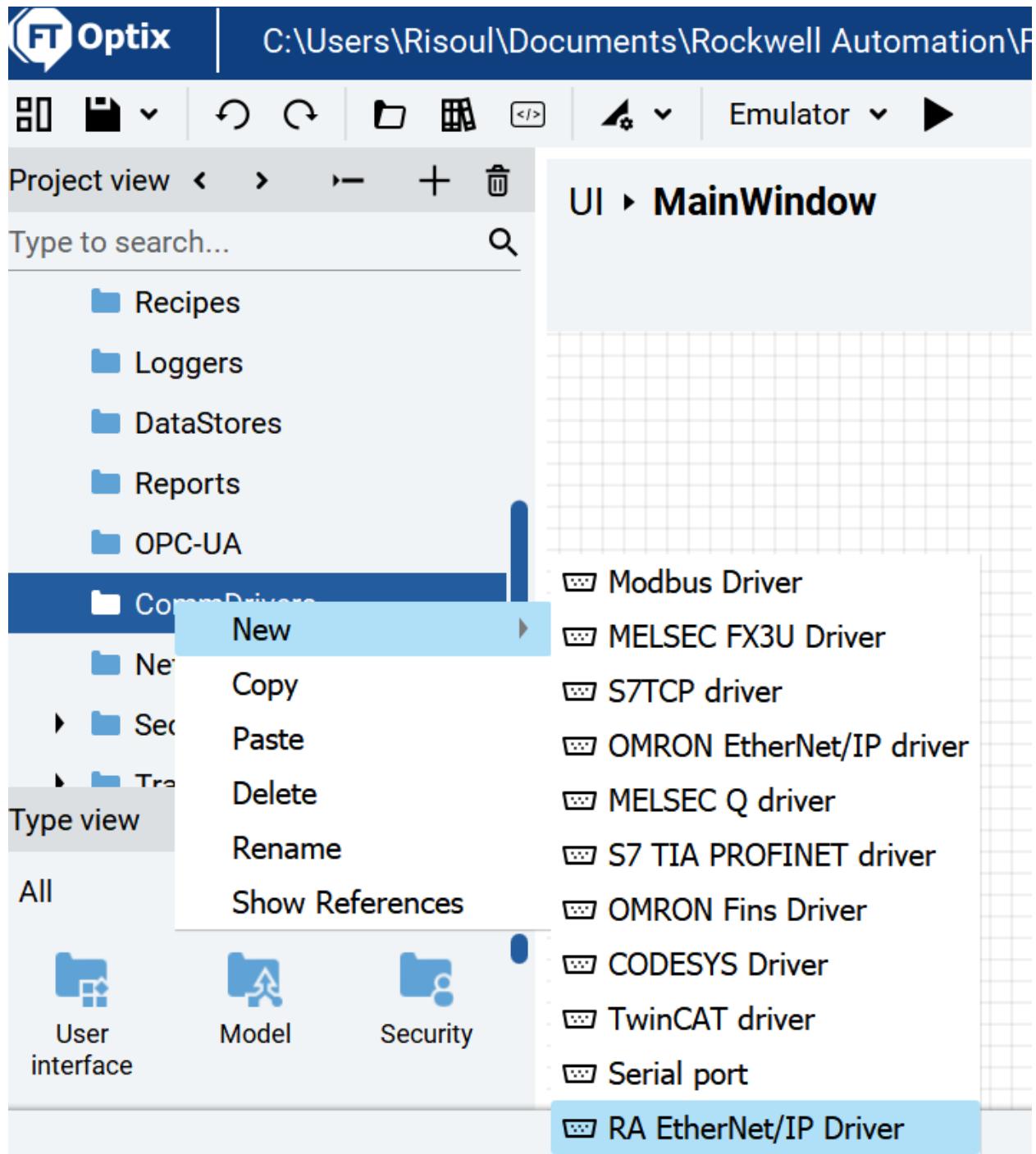
Recipes
Loggers
DataStores
Reports
OPC-UA
CommonDrivers ▾ New ▾ Copy Paste Delete Rename Show References

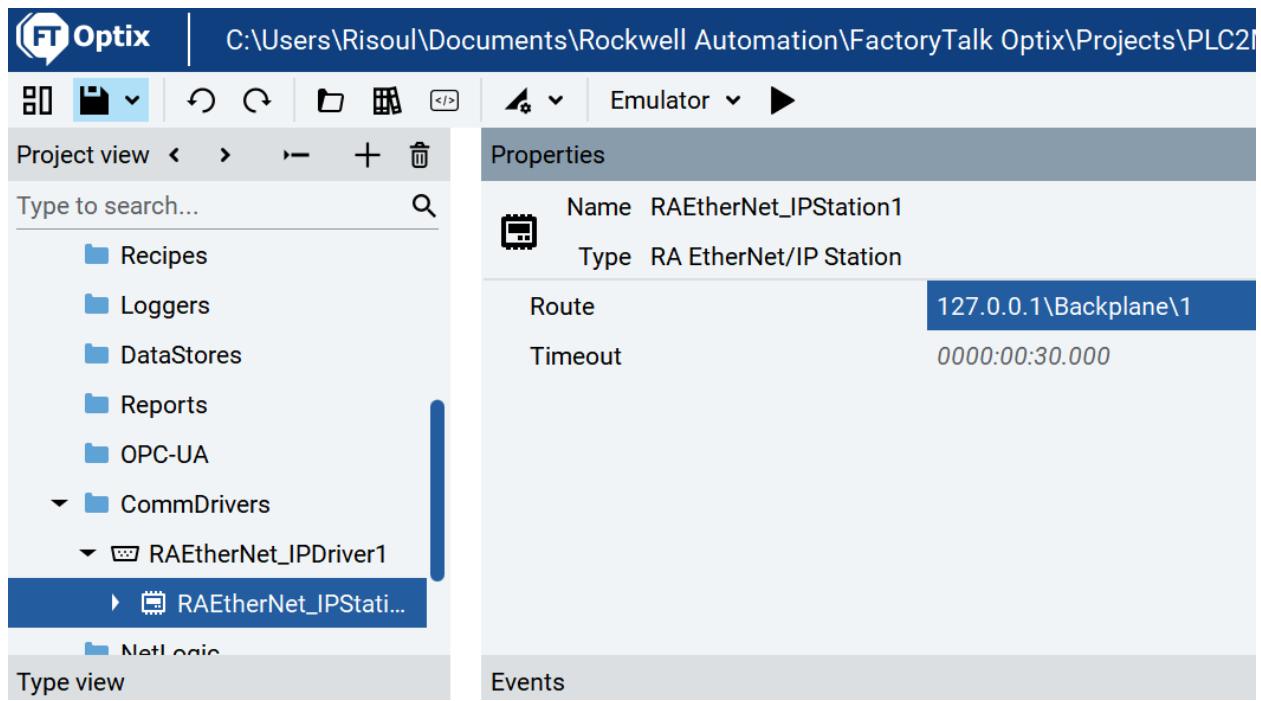
Type view
All

User interface Model Security

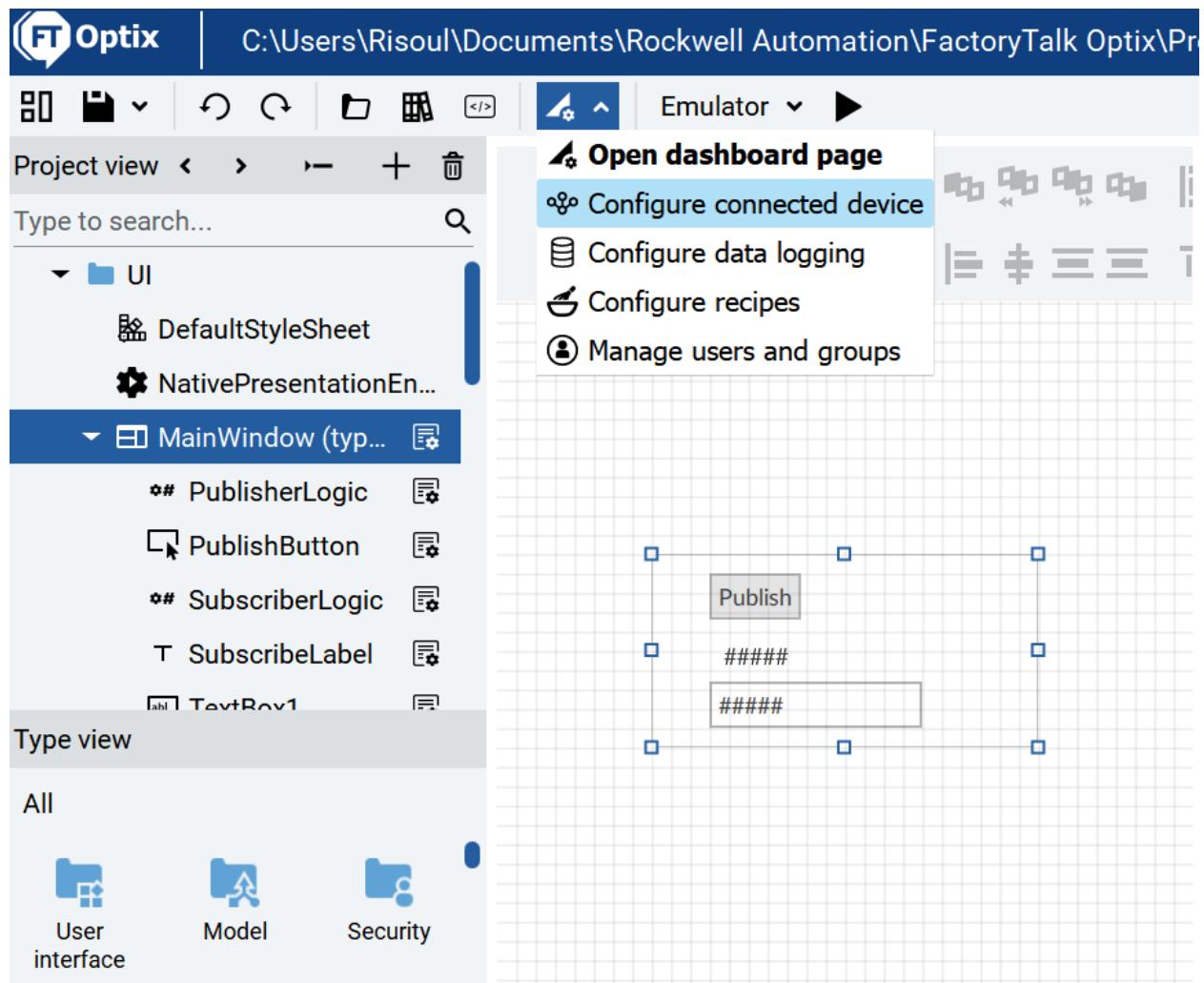
Modbus Driver
MELSEC FX3U Driver
S7TCP driver
OMRON EtherNet/IP driver
MELSEC Q driver
S7 TIA PROFINET driver
OMRON Fins Driver
CODESYS Driver
TwinCAT driver
Serial port
RA EtherNet/IP Driver

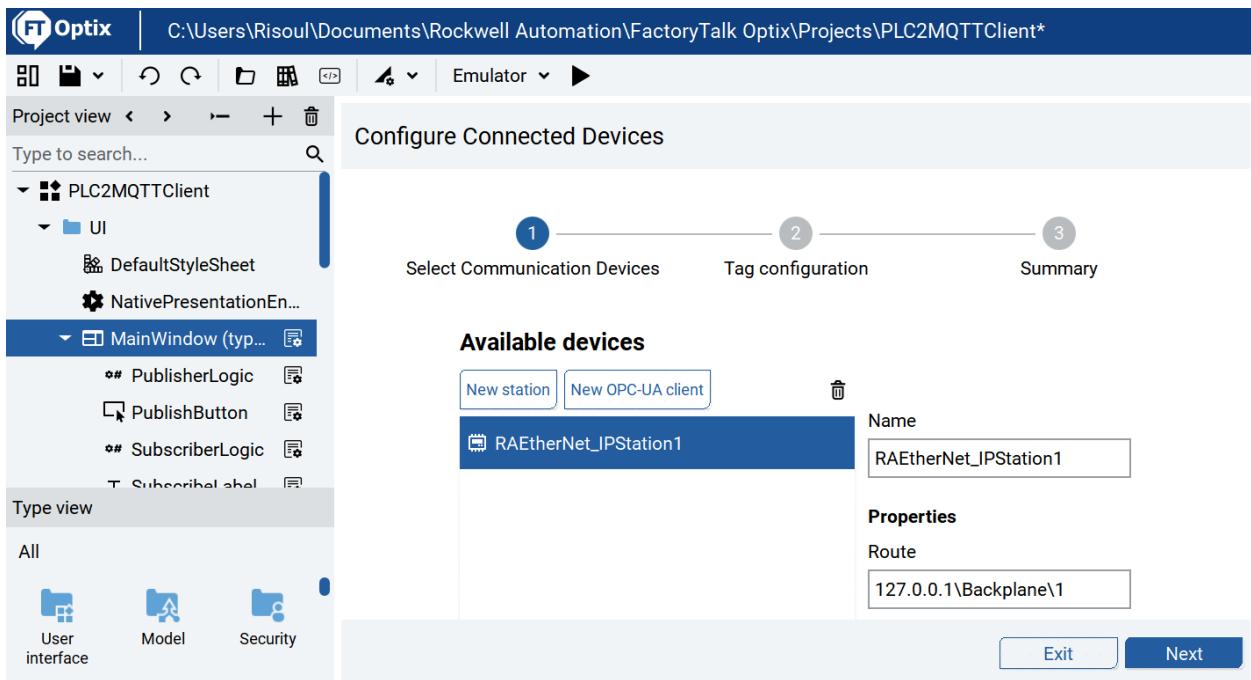
Add a new station



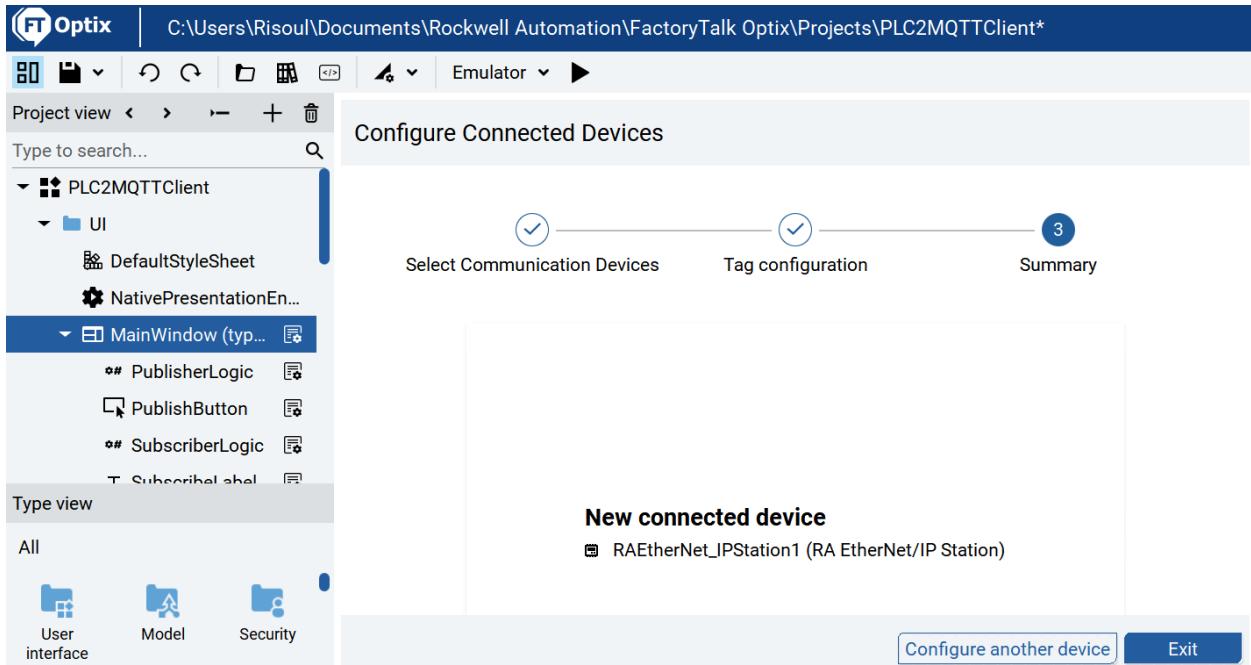


Go to configure connected device

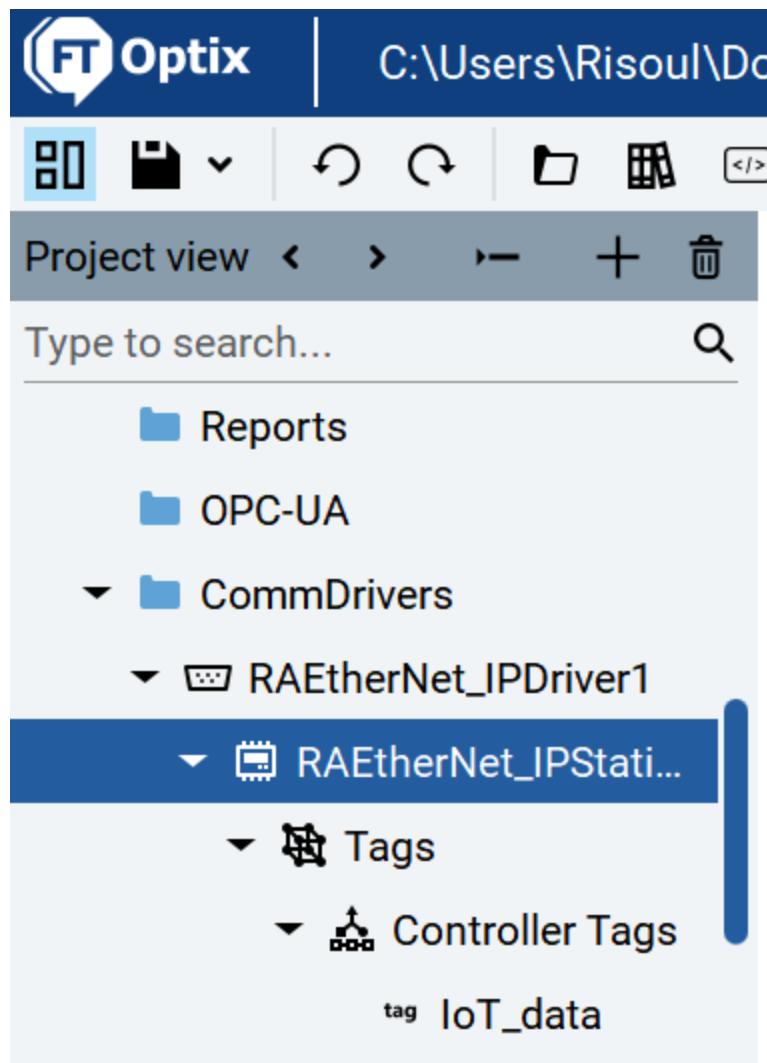




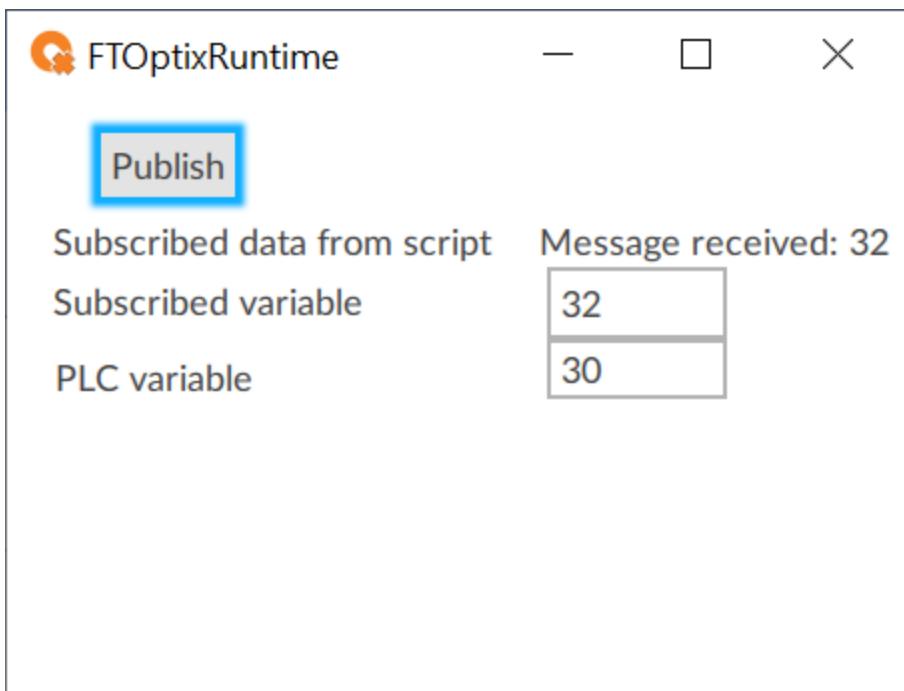
Continue as we did on chapter 2 until you reach this point



Verify that the Tag is there



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



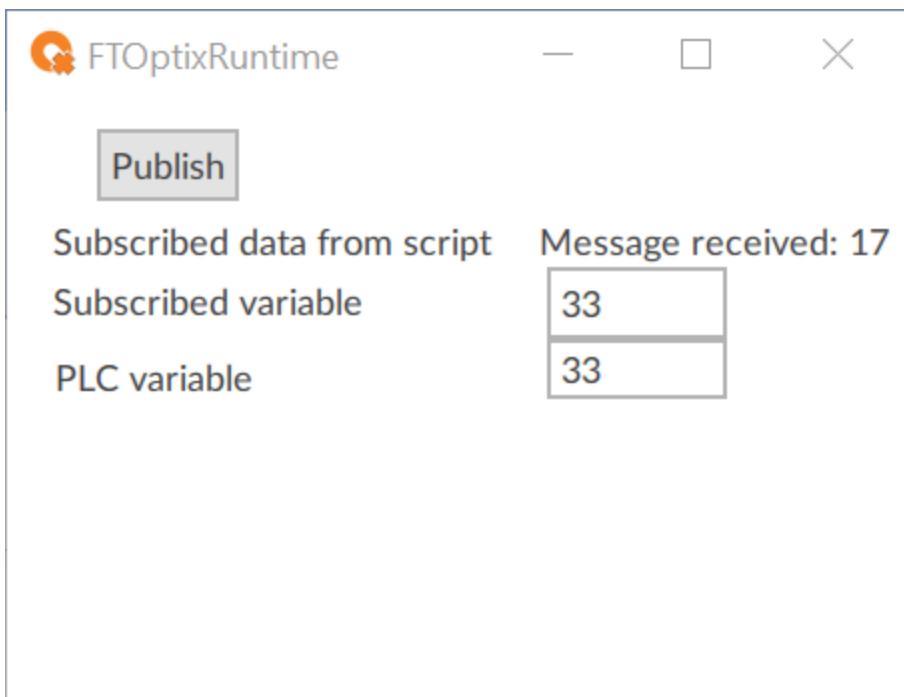
Now let's try to publish the data from PLC variable

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

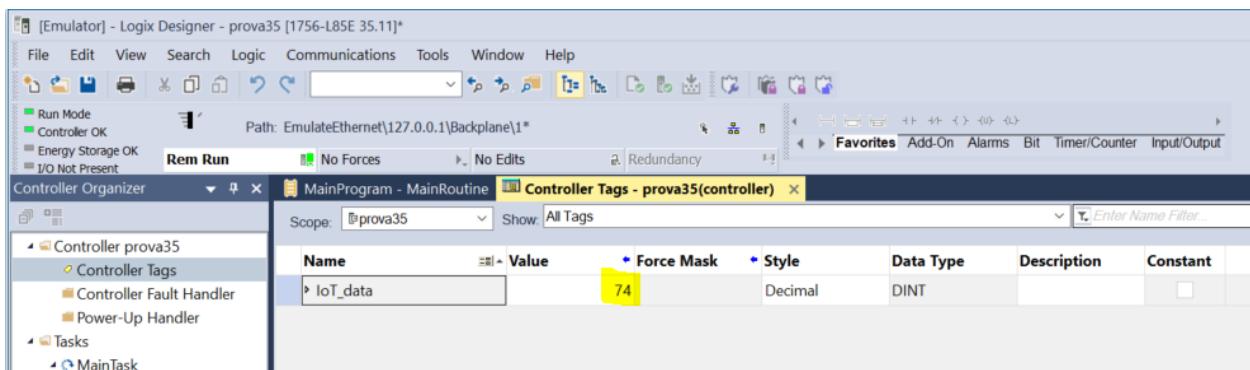
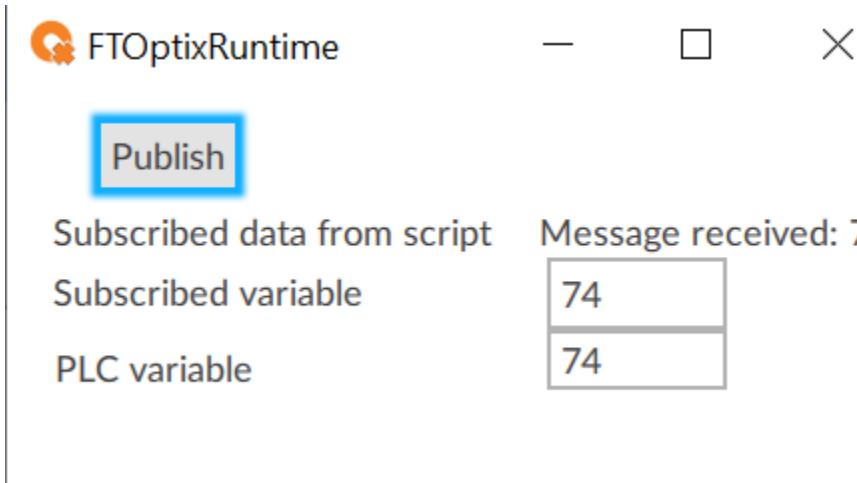
Modify the value on the PLC

Name	Value	Force Mask	Style	Data Type
IoT_data	33	33	Decimal	DINT

Then look at the Optix application

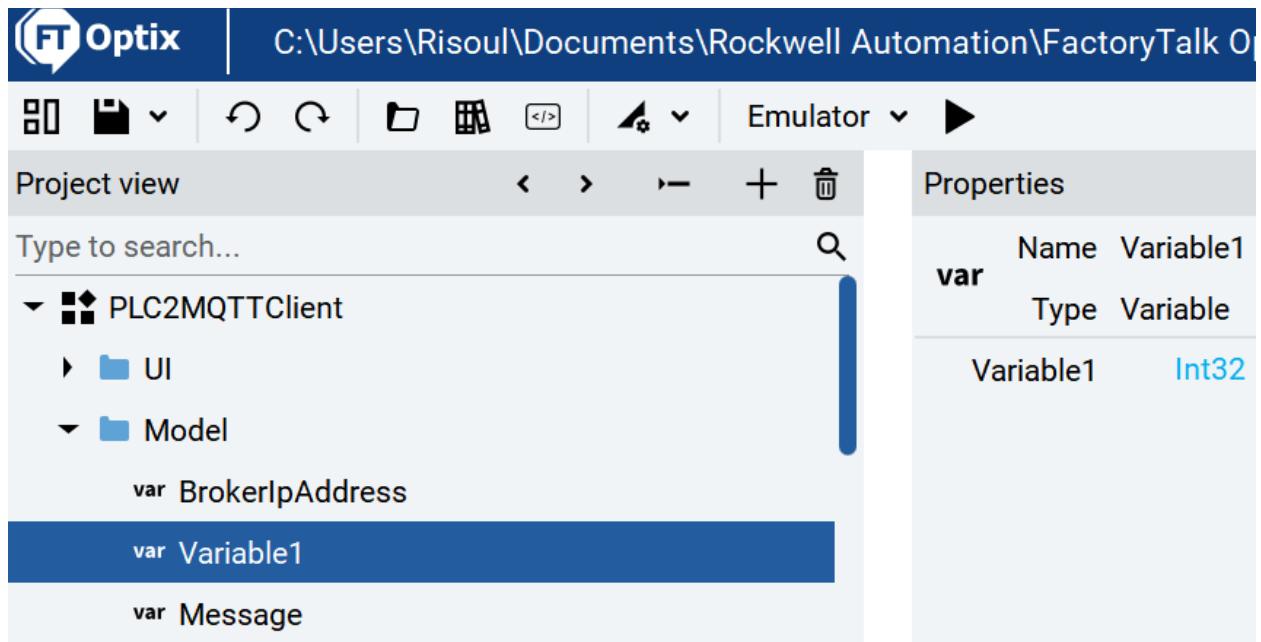


We wanted just to write on the PLC, but on the other hand if we click the button publish, then we are writing a random value on the PLC!!!



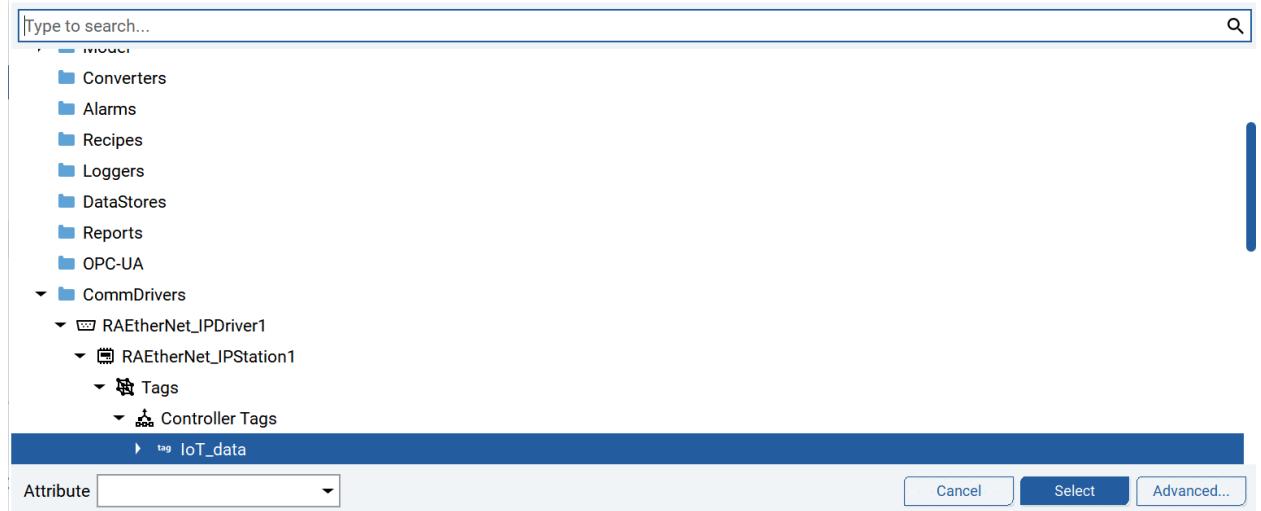
This is how to make this link

Just go to the Variable1 properties under model

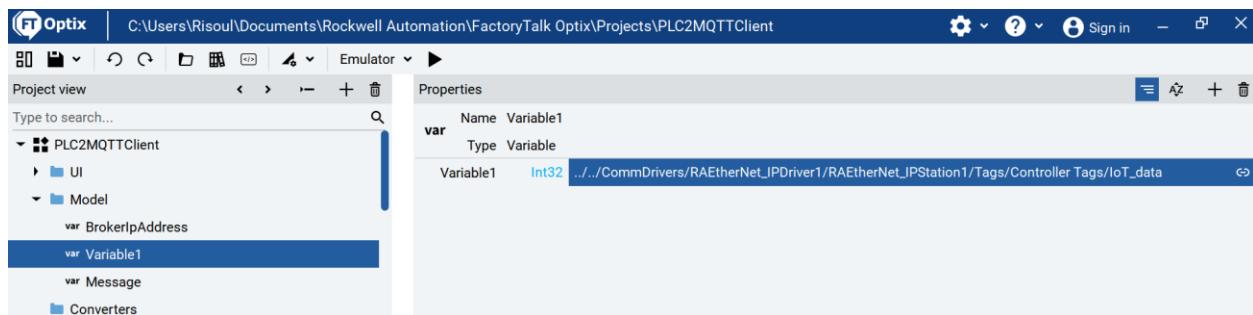


Then edit the link

Like this



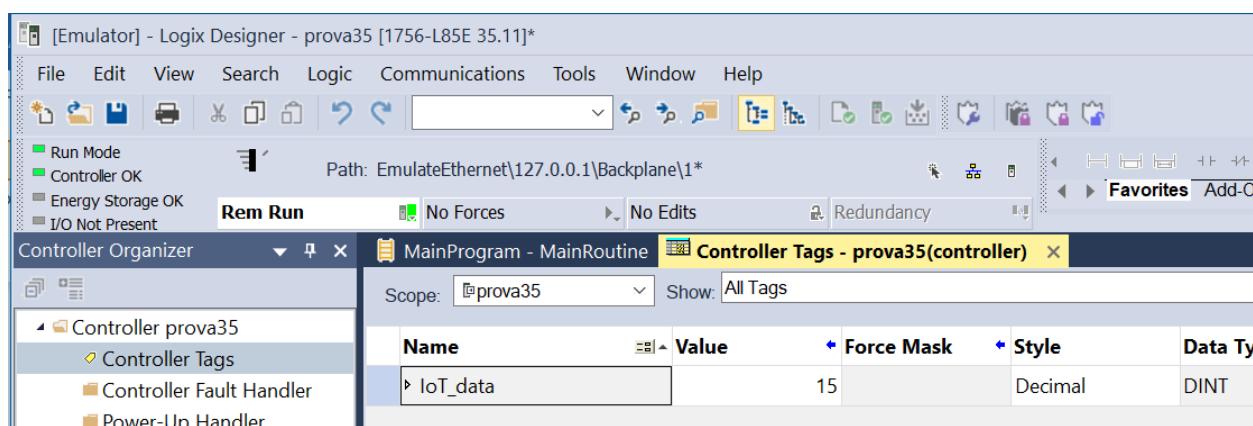
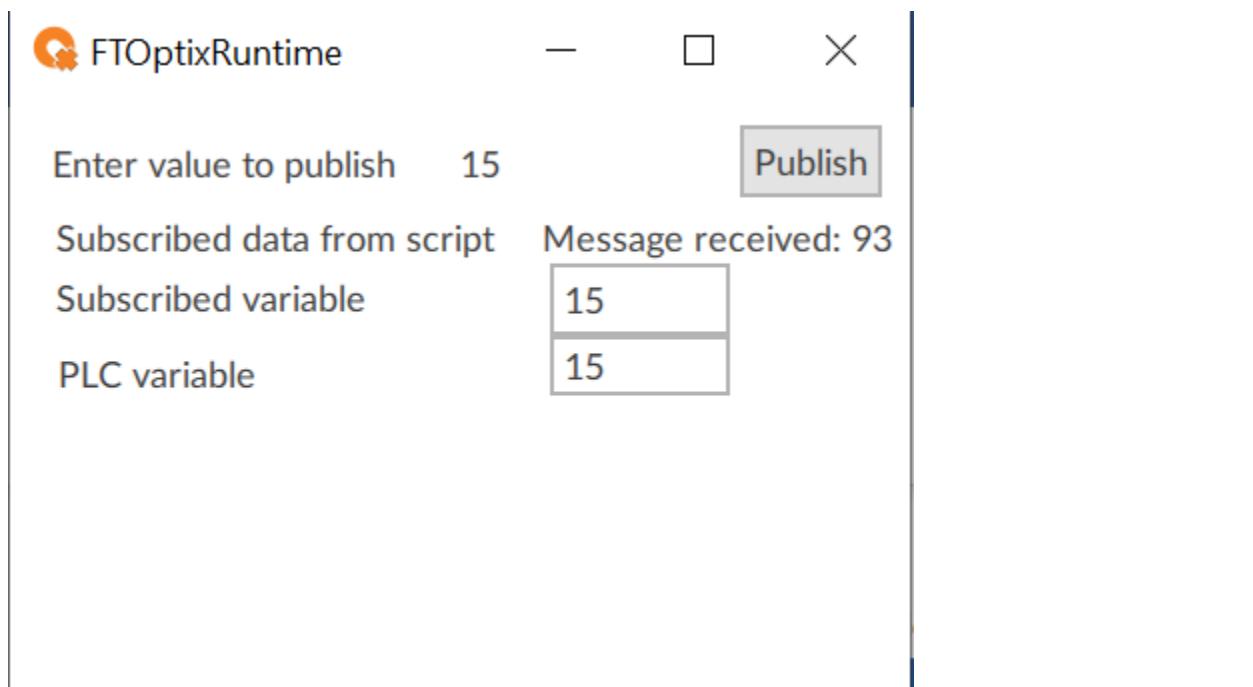
You will get this



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

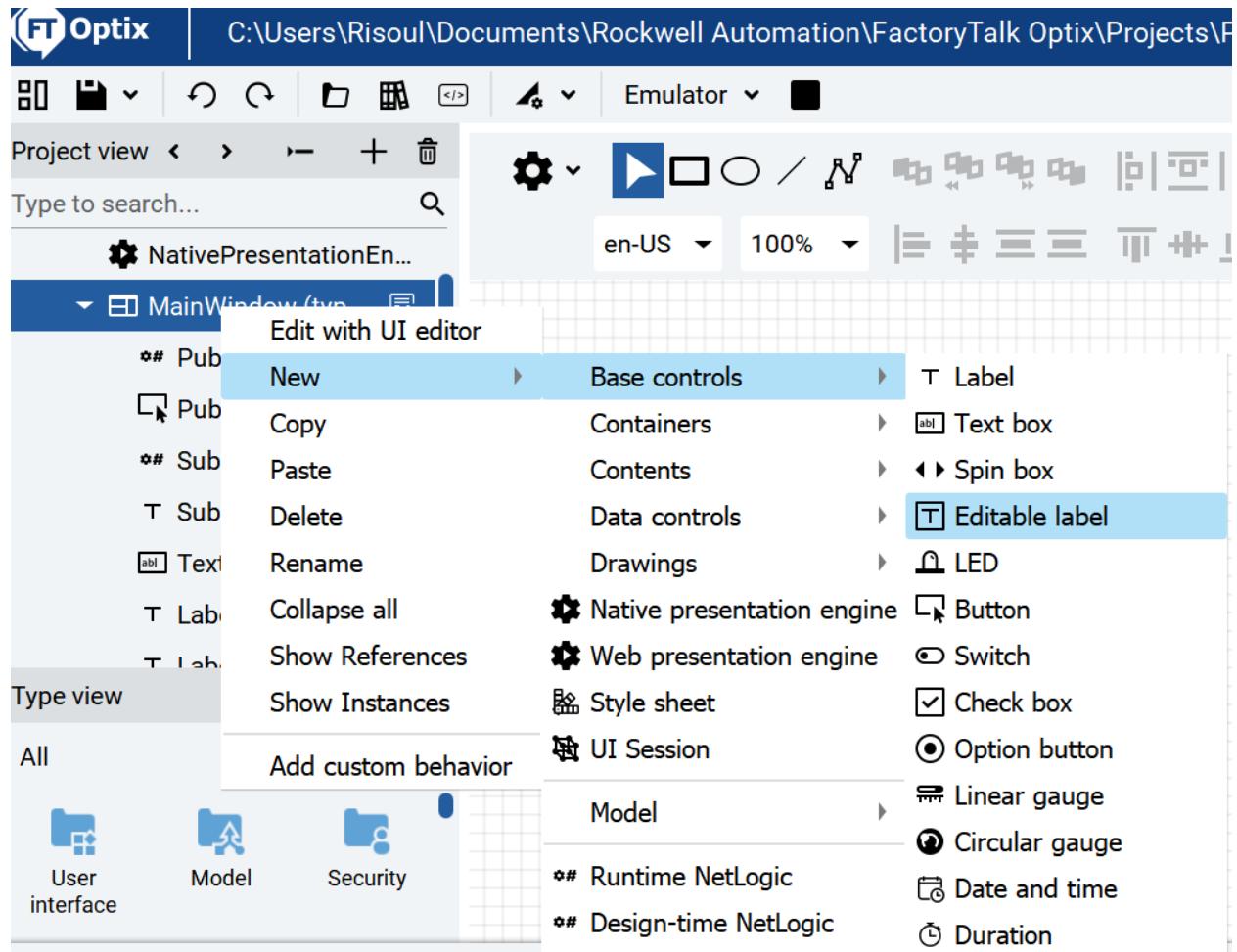
But the problem when writing to the PLC is that the variable is random. Let's try to be our desired value entered by the operator on Optix.

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

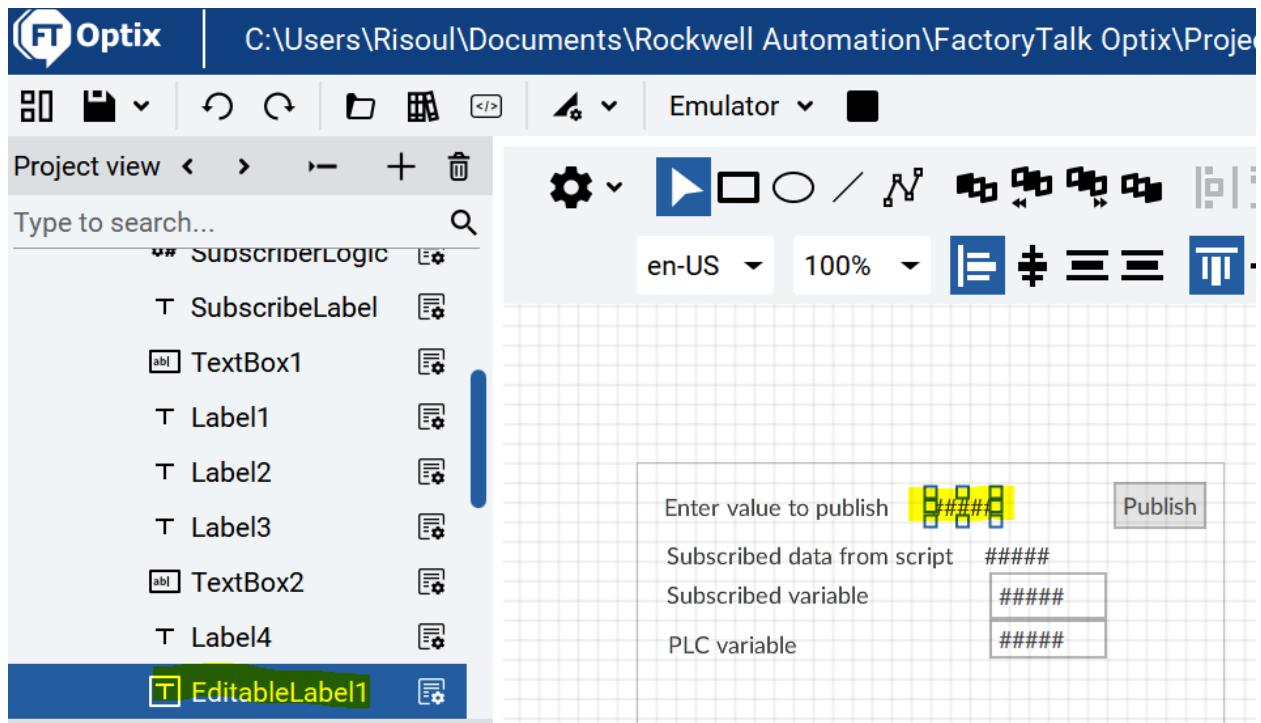


Let's see how to do it

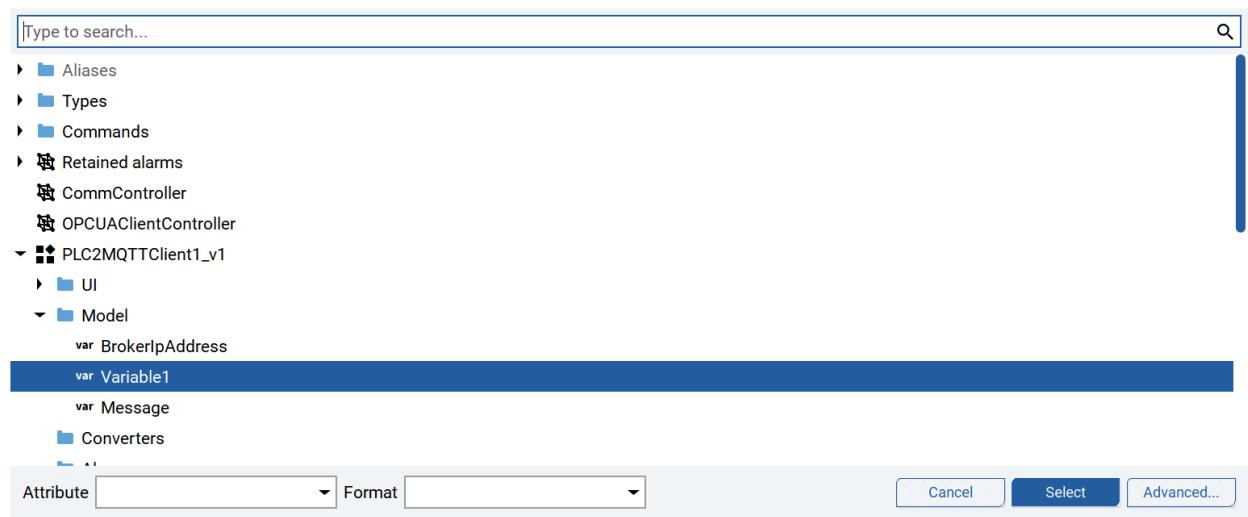
Let's create an editable Label



Like this



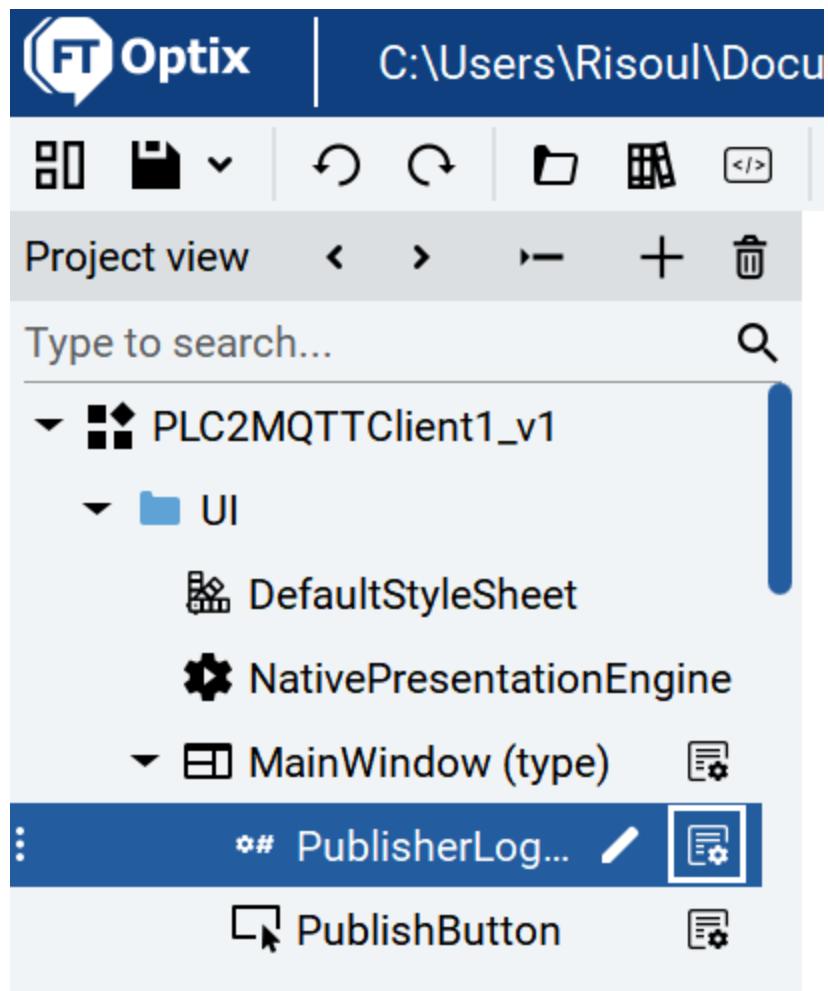
And on the text property, link to the variable



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
46         //variable1.Value = new Random().Next(0, 101);
47
48         // Publish a message
49         ushort msgId = publishClient.Publish("/my_topic", // topic
50             System.Text.Encoding.UTF8.GetBytes(((int)variable1.Value).ToString()), // message body
51             MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, // QoS level
52             false); // retained
53     }
54
55     private MqttClient publishClient;
56 }
57

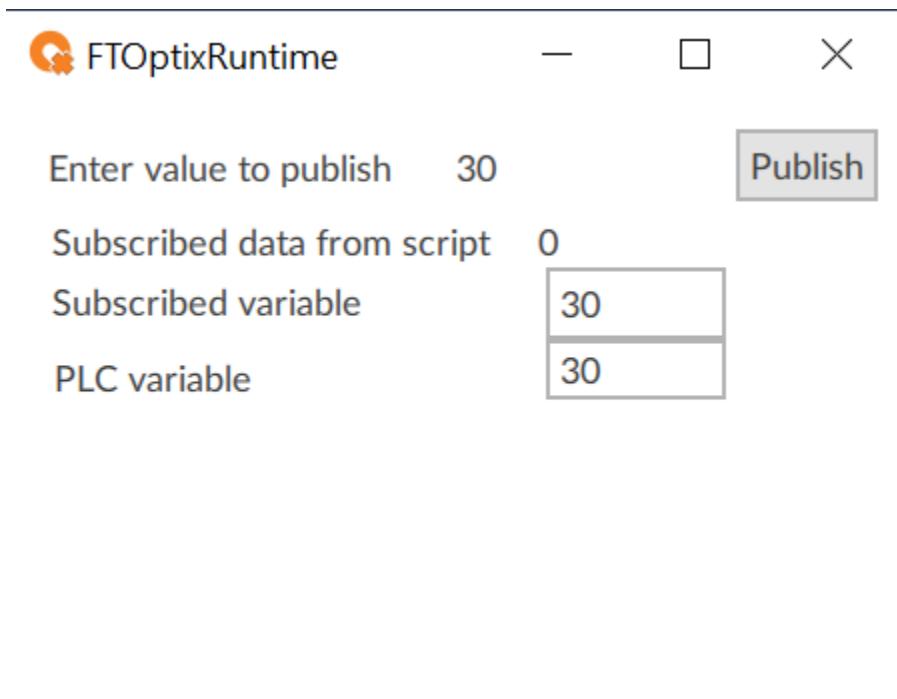
```

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

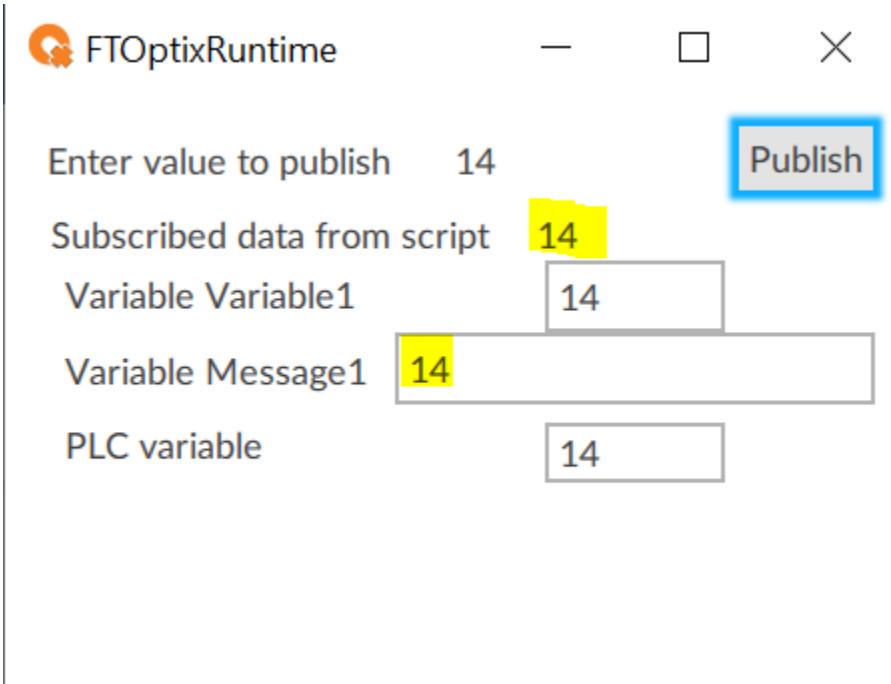
```
43 |     messageVariable.Value = "Message received: " + System.Text.Encoding.UTF8.GetString(e.Message);
```

And leaving like this

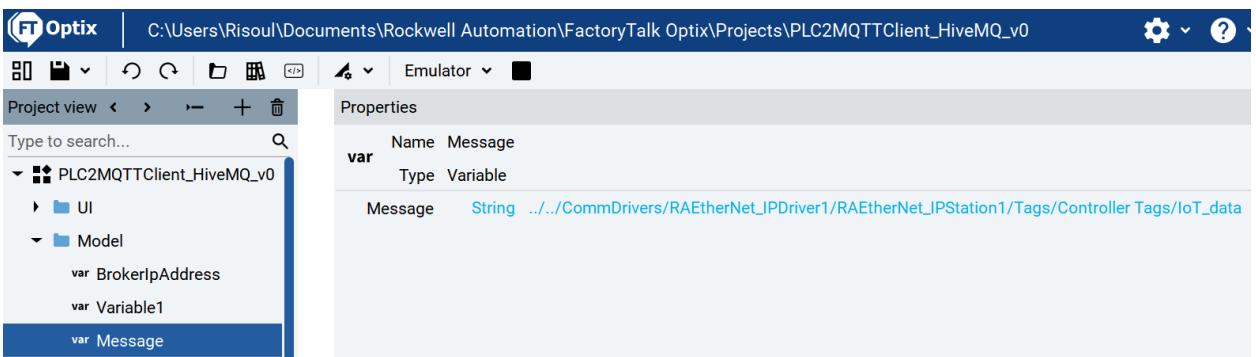
```
44 |     messageVariable.Value = System.Text.Encoding.UTF8.GetString(e.Message);
```

Then save on Visual Studio Code

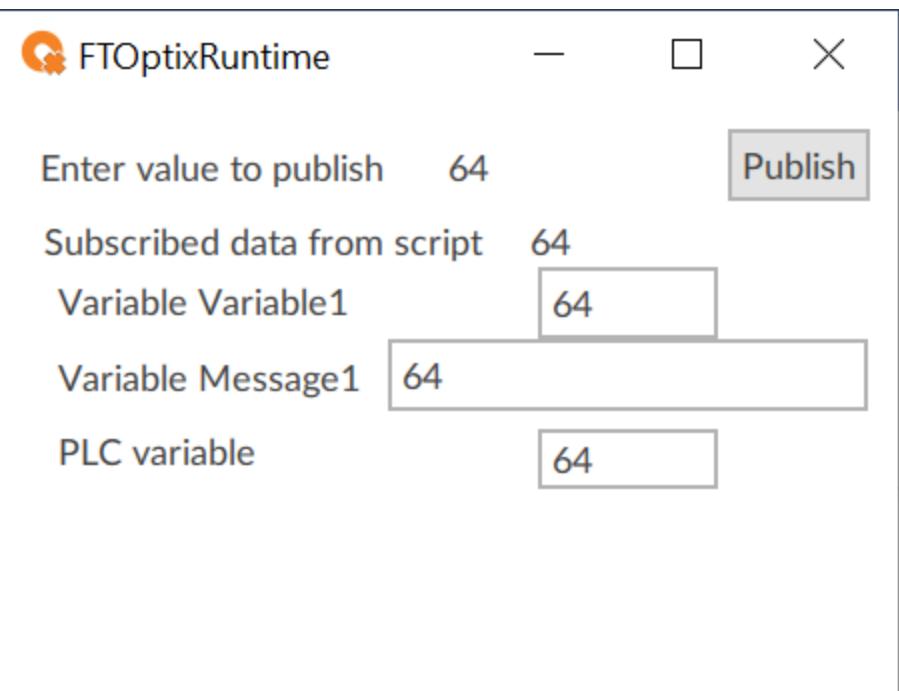
Now we do not have the string like before



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 Rem Run No Forces No Edits Redundancy

Path: EmulateEthernet\127.0.0.1\Backplane\1*

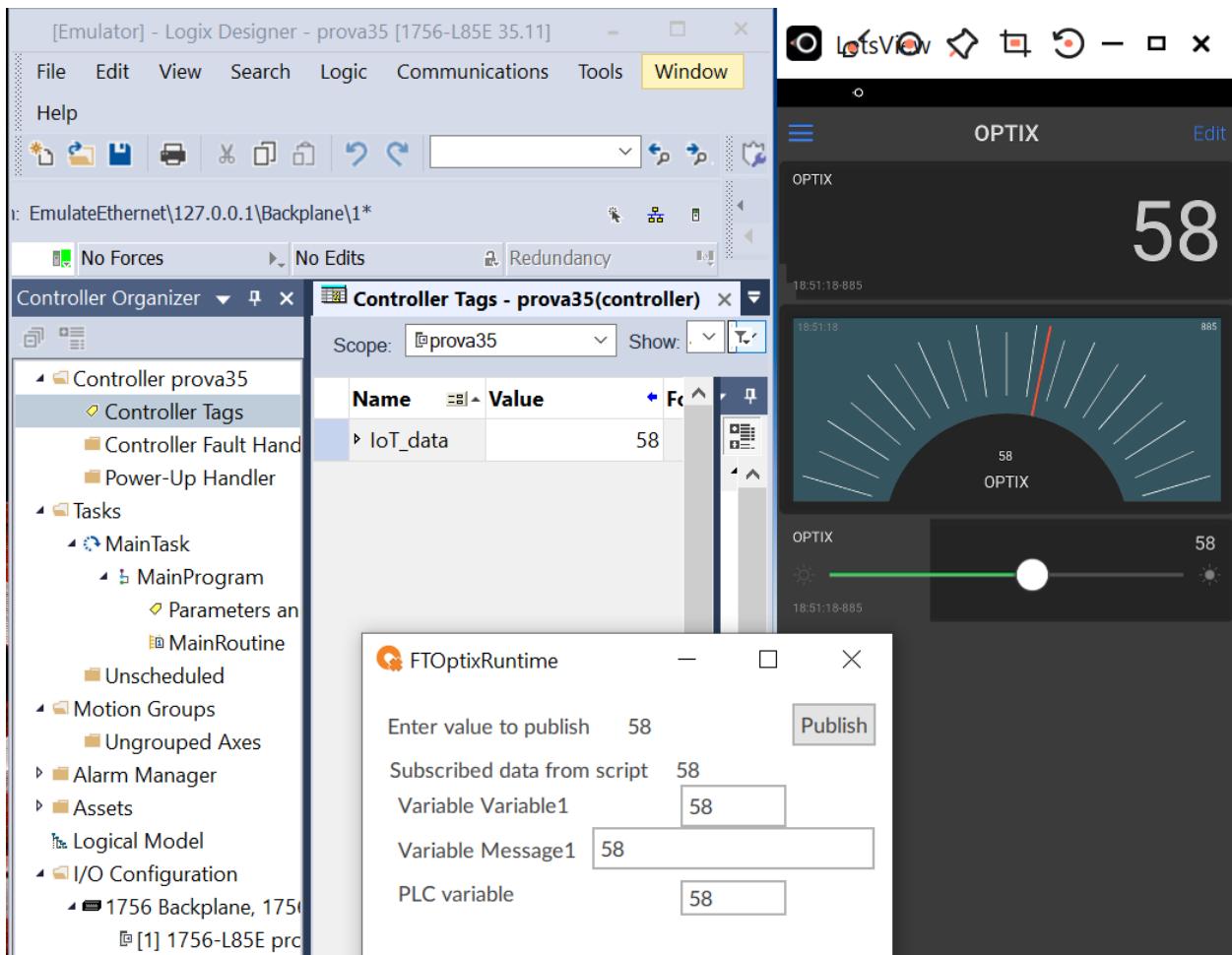
Controller Organizer

Controller Tags - prova35(controller)

MainProgram - Mai

Scope: prova35 Show: All Tags

Name	Value
IoT_data	64



As you can see on this video

<https://youtu.be/u6EEDMmJJBU>

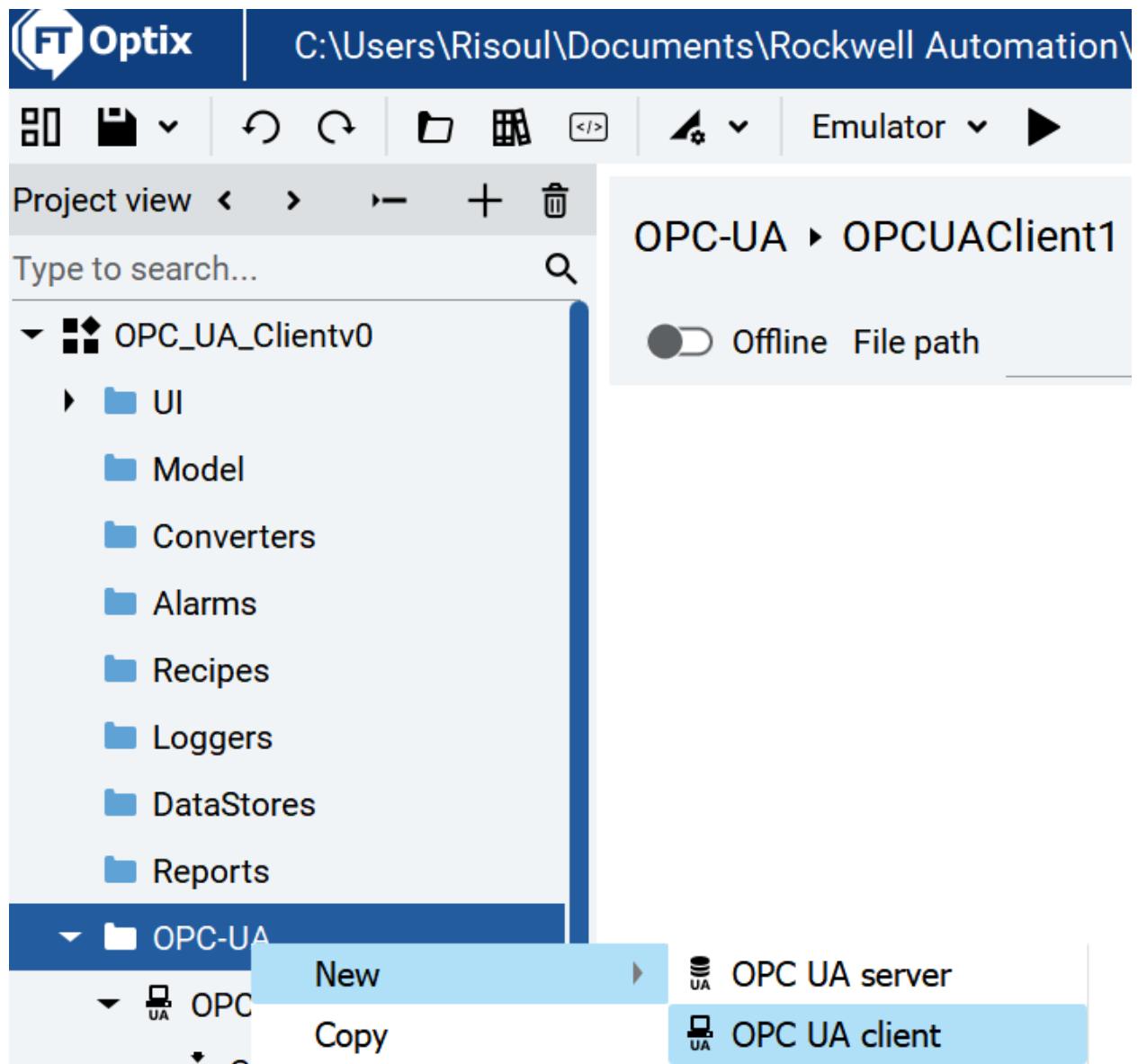
You can find the code here

https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git

7. 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>



Change the address and port of the OPC UA server

The screenshot shows the Rockwell FactoryTalk Optix interface. The left pane displays a project tree under 'OPC_UA_Clientv0' with categories like UI, Model, Converters, Alarms, Recipes, Loggers, DataStores, Reports, and OPC-UA. Under OPC-UA, 'OPCUAClient1' is selected, showing its properties. The properties panel on the right includes:

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

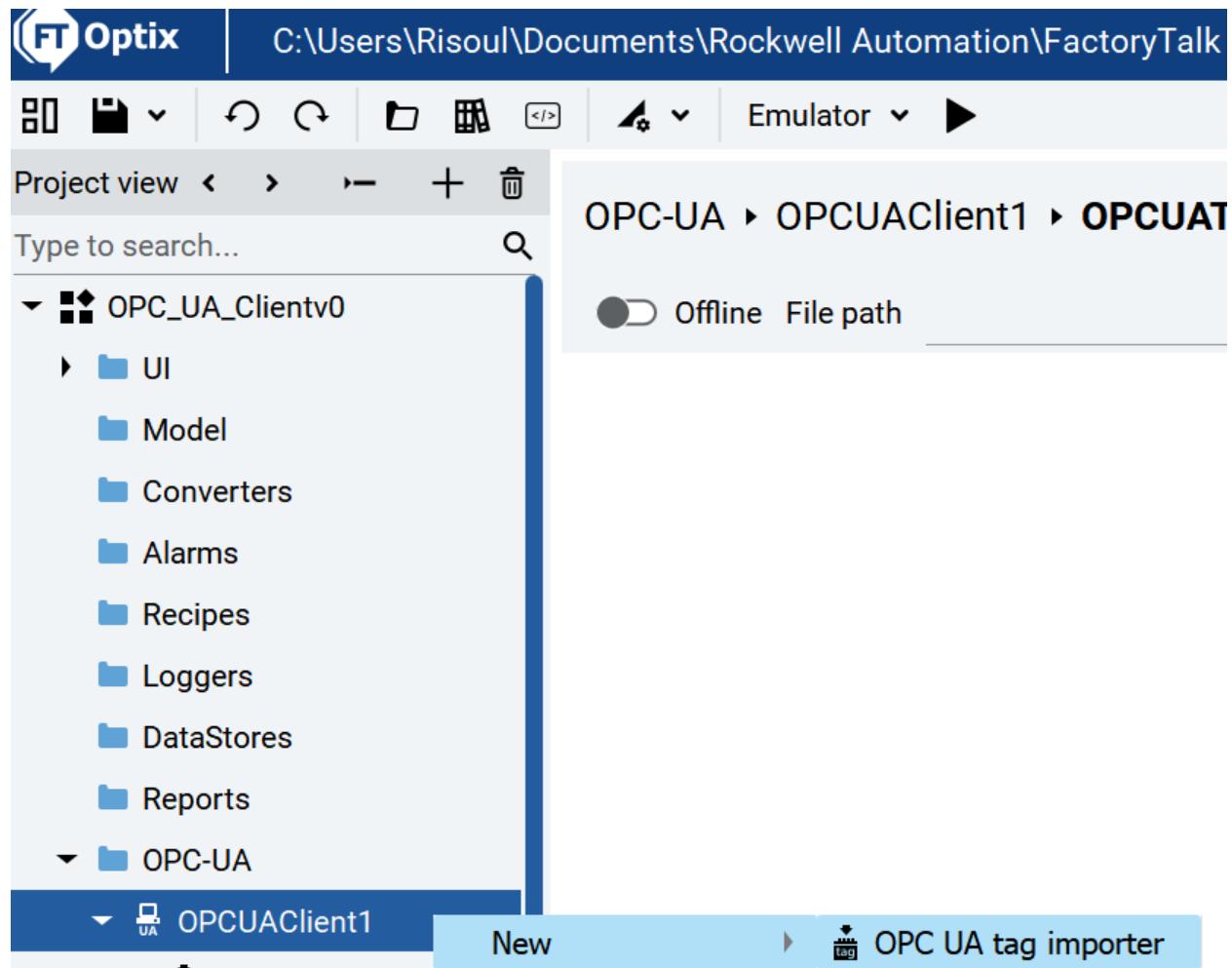
For instance with FT Kepserver Enterprise, port number is 49370

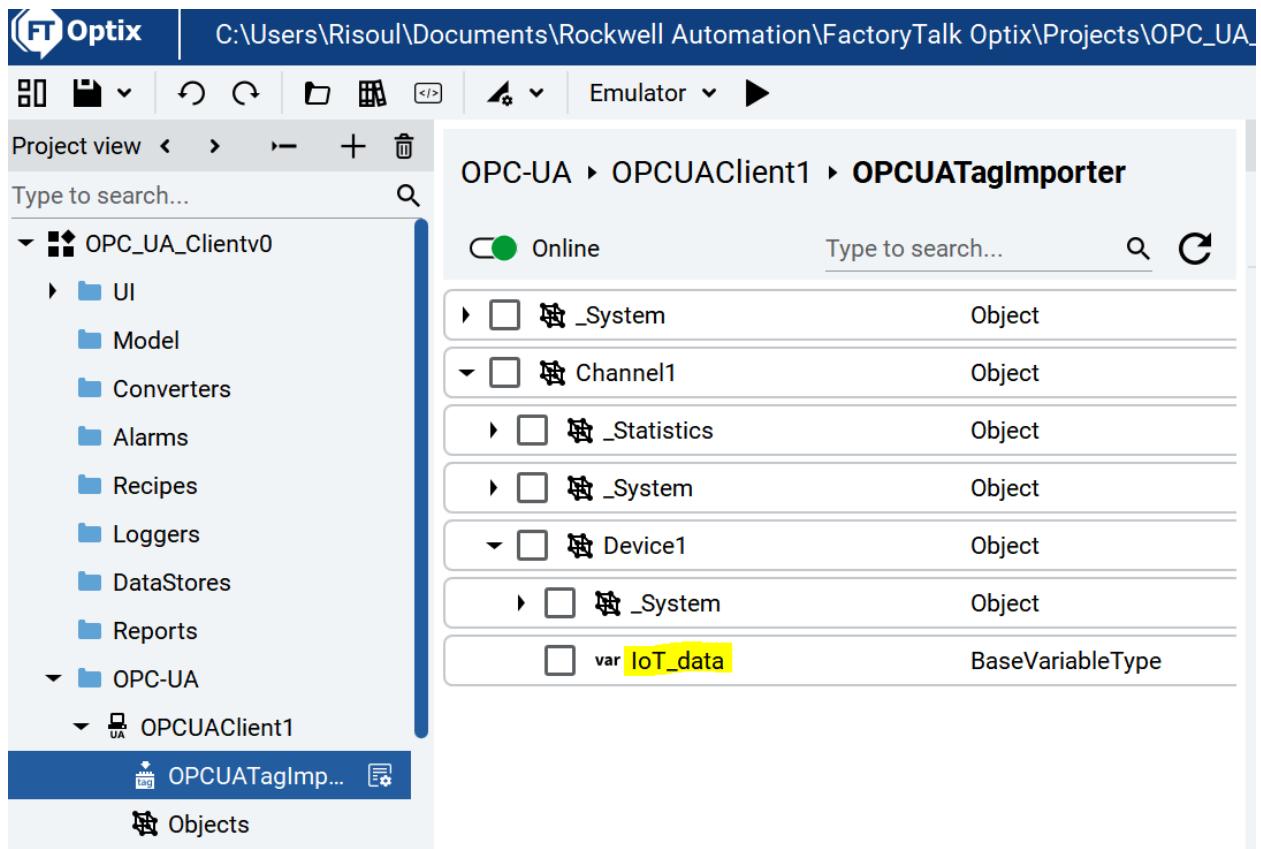
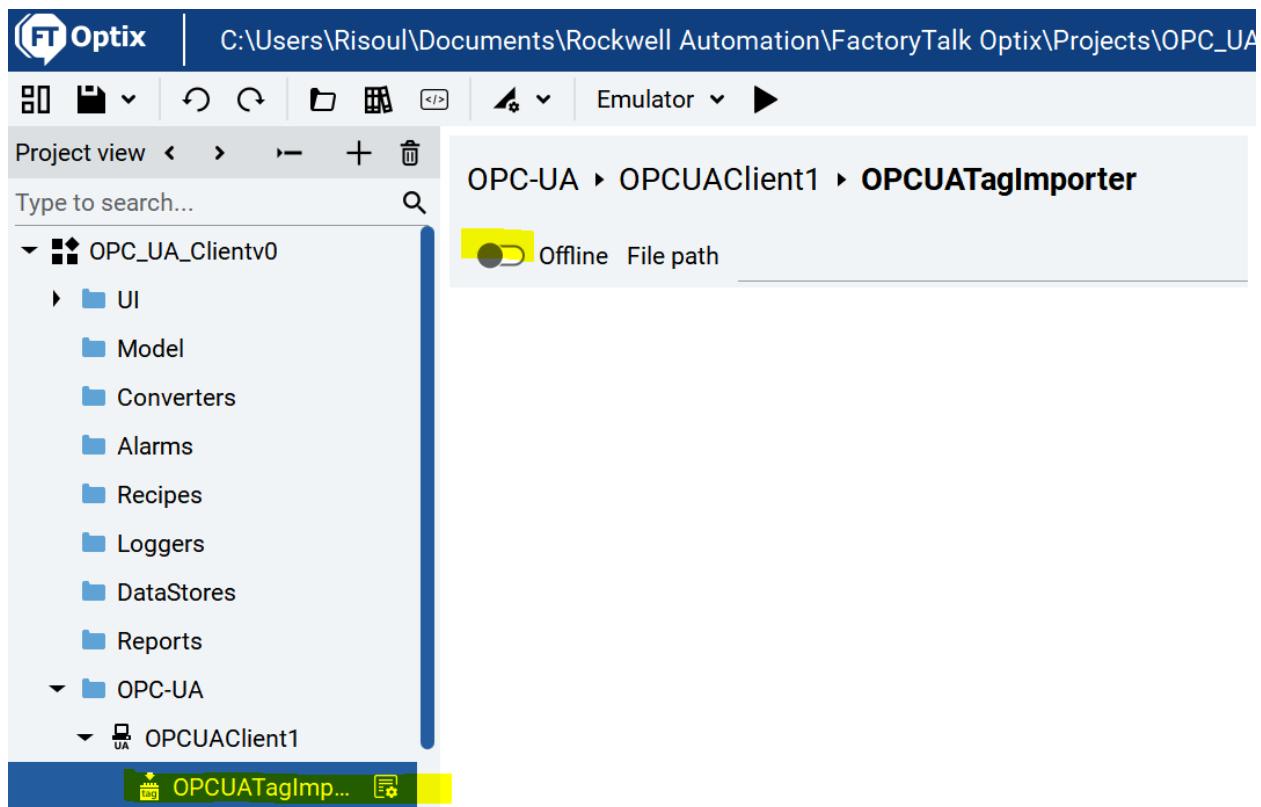
The screenshot shows the OPC UA Configuration Manager. The top navigation bar has tabs for Server Endpoints, Trusted Clients, Discovery Servers, and Instance Certificates. The 'Server Endpoints' tab is active, showing two entries in a table:

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)

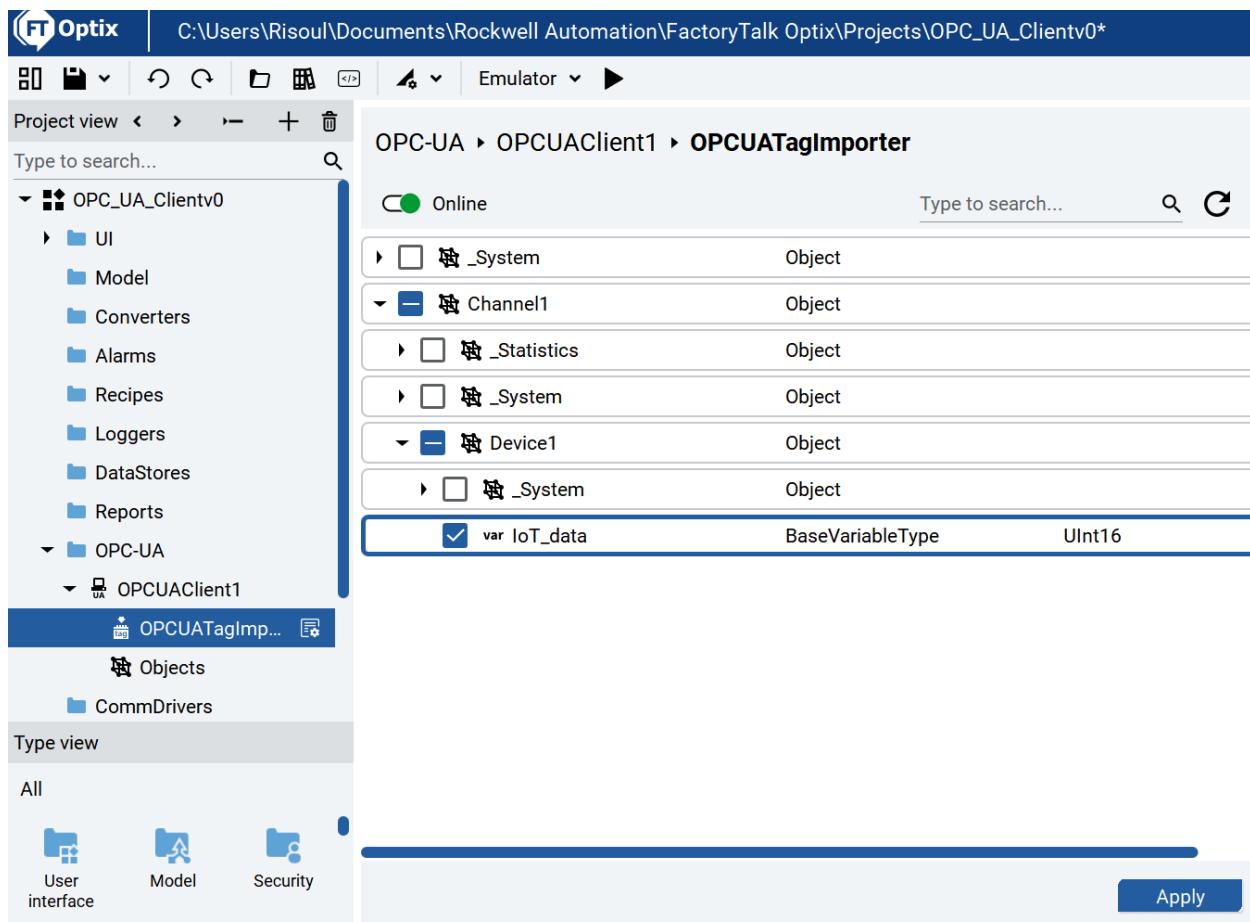
Right click to add a new OPCUA Tag Importer

Then double click on it

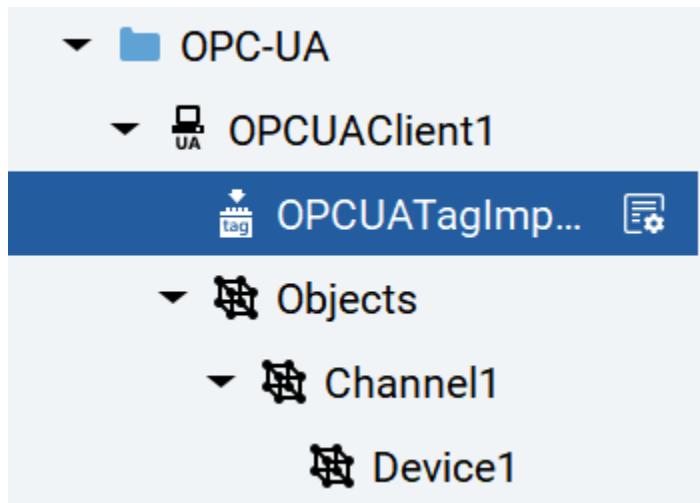




Click on apply



On Objects you will see this



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

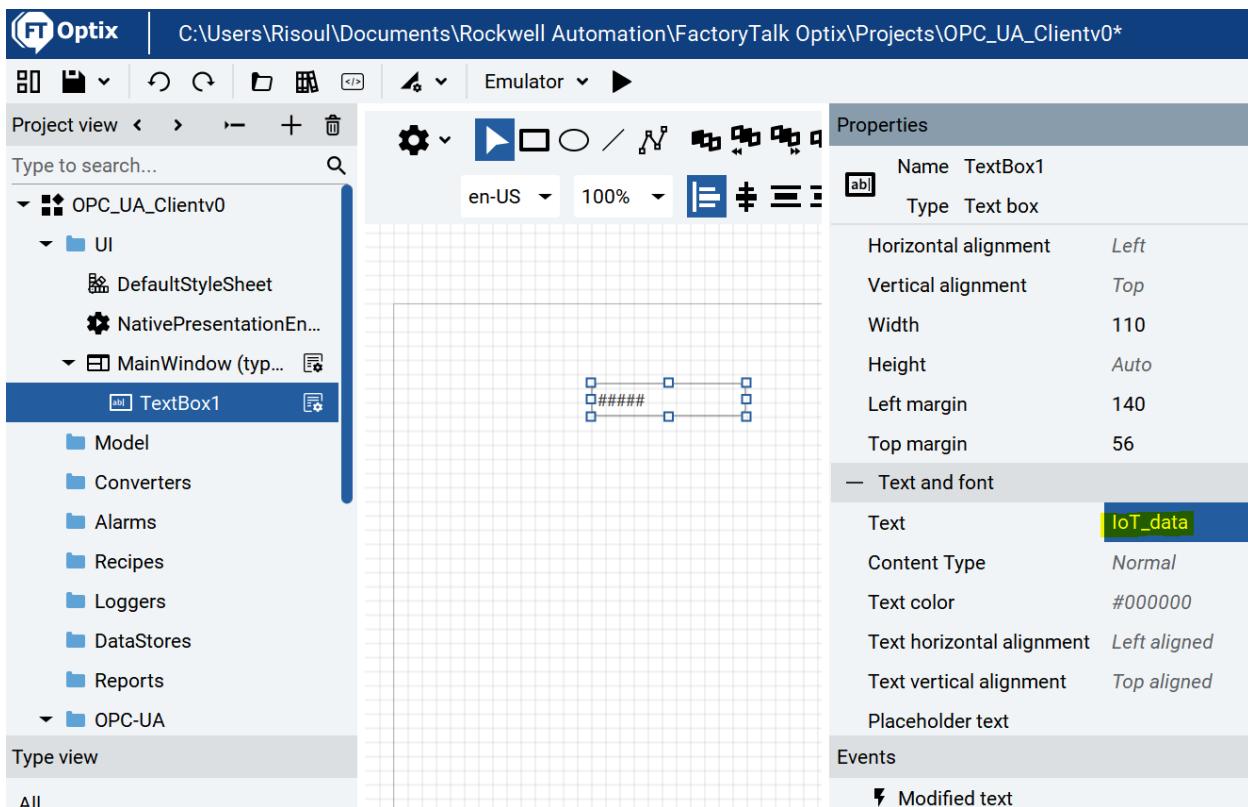
Type to search... 🔍

```

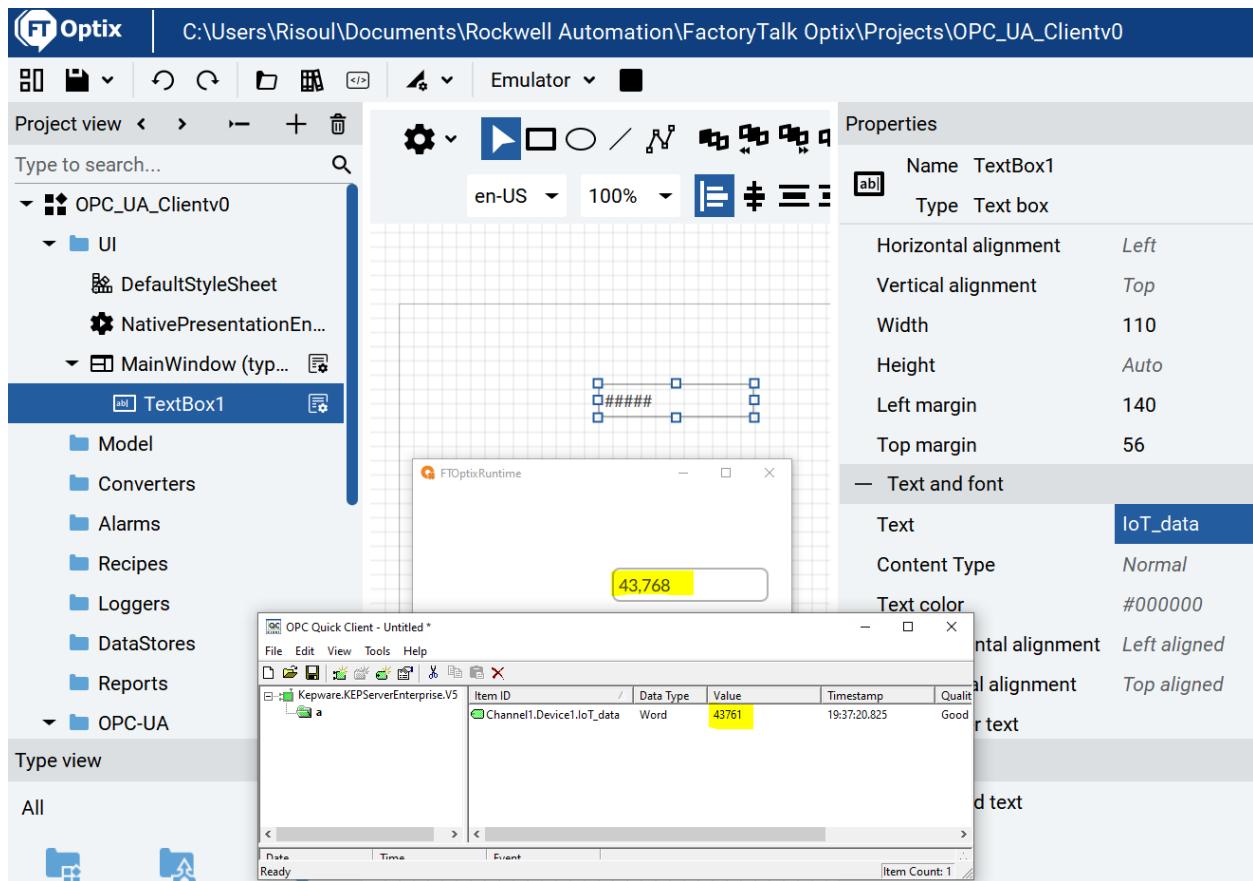
OPCUAClient1
  Objects
    Channel1
      Device1
        IoT_data
          BrowseName: IoT_data
          DataType: UInt16
OPCUATagImporter
  Connection status
  Server URI
  Enabled
  Events
  Synchronize the Node IDs on start
  Server endpoint URL
  Verify server identity

```

Attribute Cancel Select Advanced...



That's all



Let's add a label



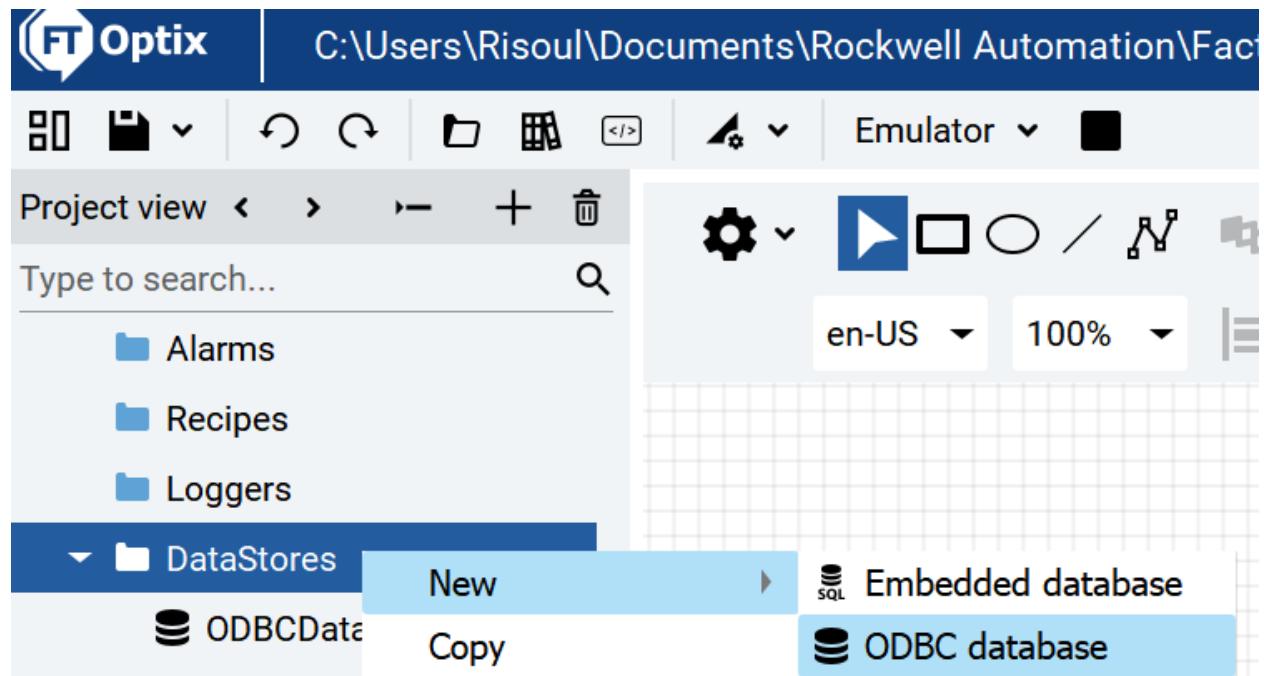
8. Writing to an SQL database

Let's write to an 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 navigating, saving, and executing queries. The Object Explorer on the left shows the database structure for 'LAPTOP-RJD8T884\SQLEXPRESS (SQL)'. It includes nodes for Databases, System Databases, Database Snapshots, the 'xavier' database, Database Diagrams, Tables, System Tables, FileTables, External Tables, Graph Tables, and the specific table 'dbo.Table1'. Under 'dbo.Table1', the 'Columns' node is expanded, showing a single column named 'PLC_data' with a data type of 'real, null'. The Results pane on the right displays the output of the query 'select * from Table1;', which shows four rows, each containing the value '25' in the 'PLC_data' column.

	PLC_data
1	25
2	25
3	25
4	25

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

Properties

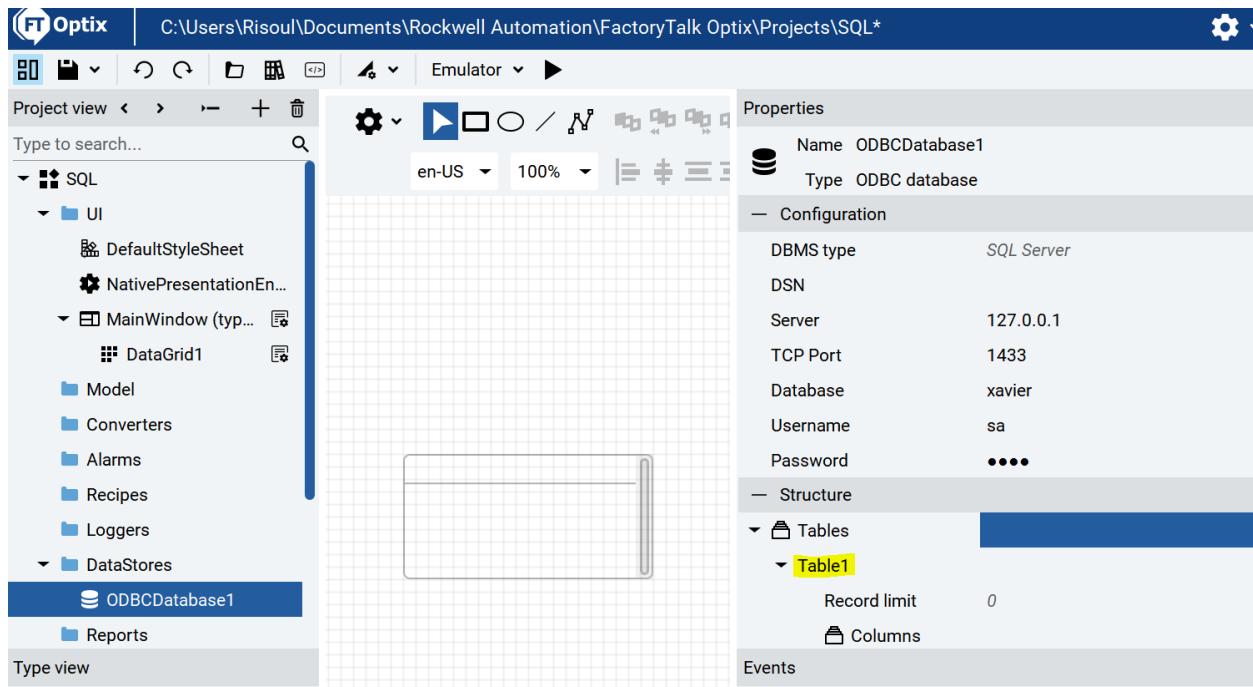
Name	ODBCDatabase1
Type	ODBC database
— Configuration	
DBMS type	SQL Server
DSN	
Server	127.0.0.1
TCP Port	1433
Database	xavier
Username	sa
Password	*****
Password	*****
— Structure	
▼ Tables	+ -
▼ Table1	
Record limit	0
▼ Columns	+ -
PLC_data	Decimal 0

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

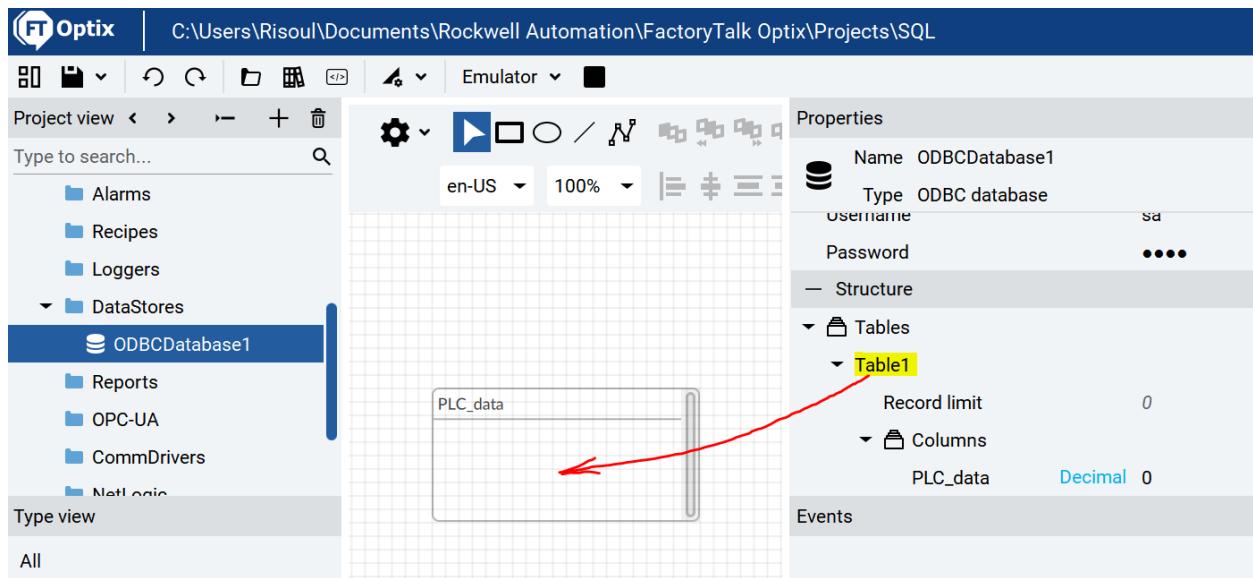
<file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/using-the-software/datastore/Display-database-table-data.html>

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

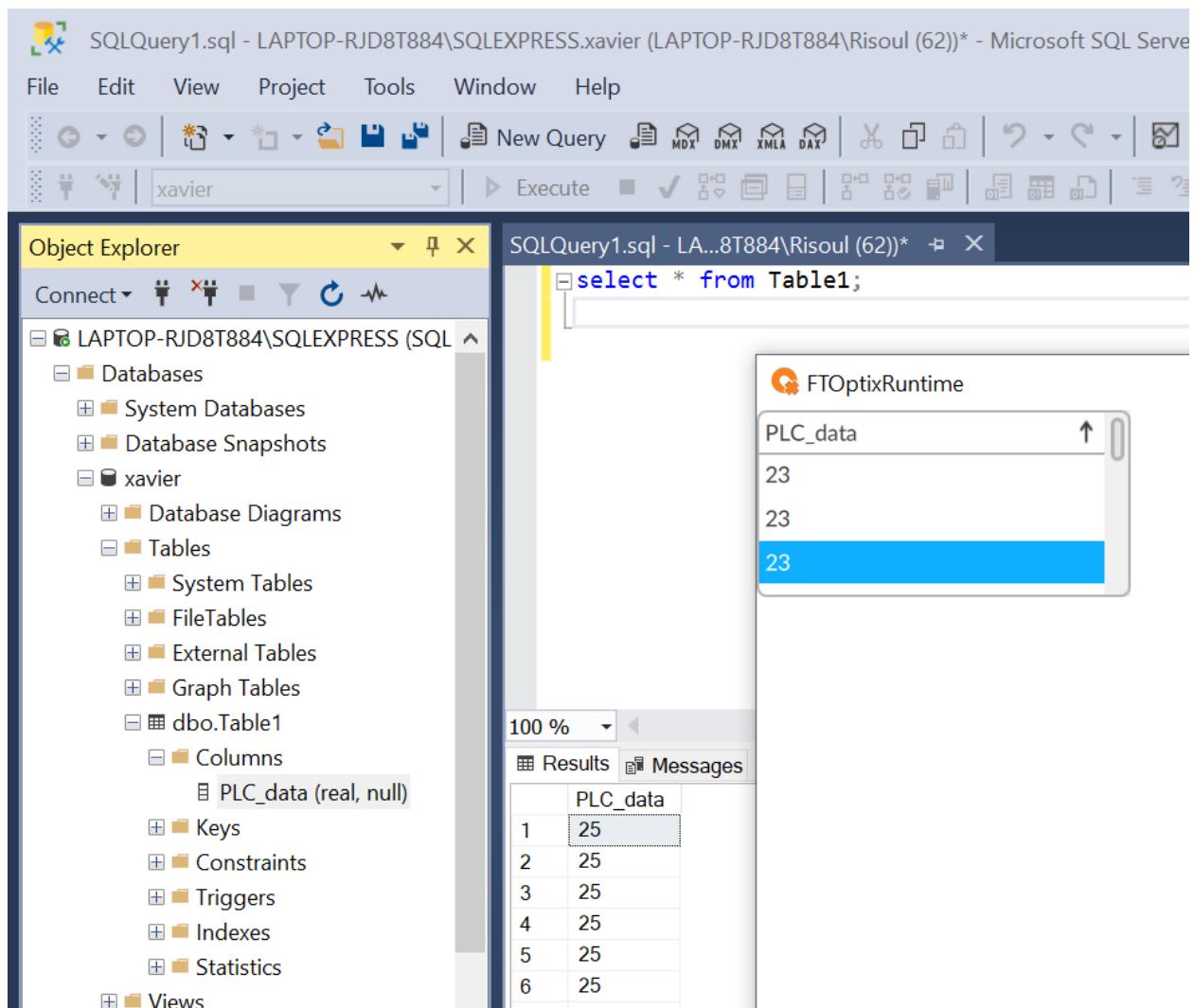


Do not forget to add a column minimum

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>

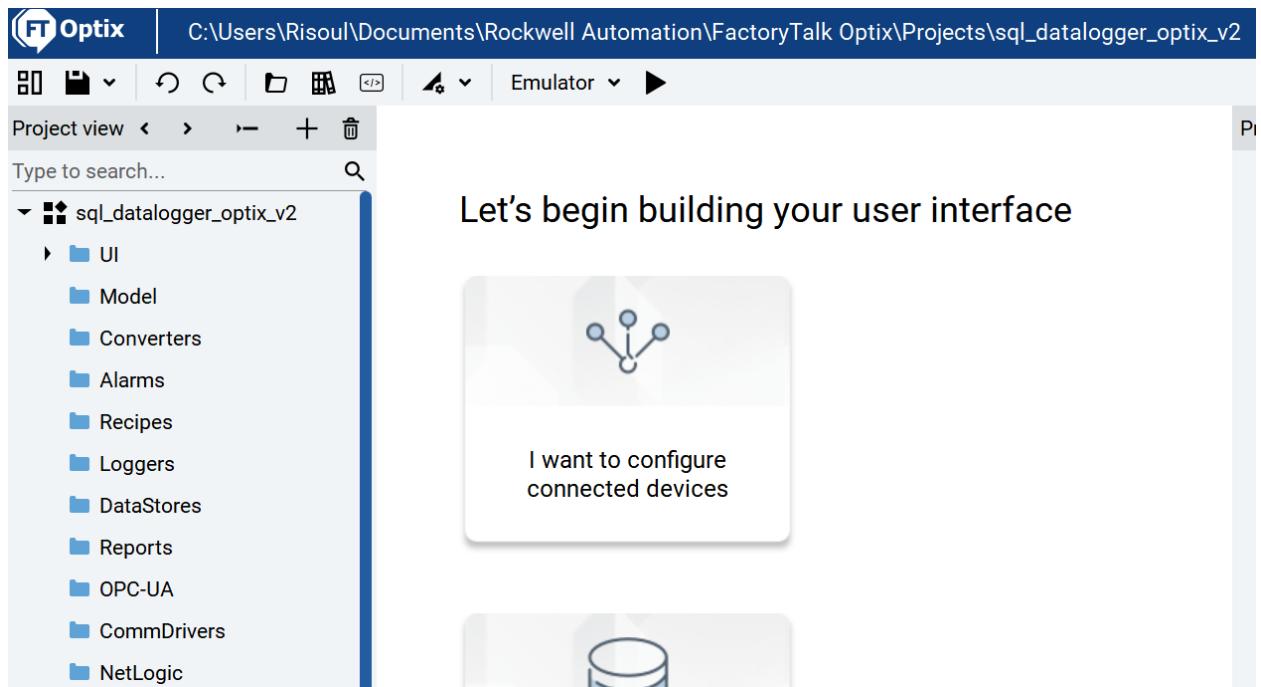
Download a sample project: [DataLoggerODBC.zip](#)

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

To see how it is made.

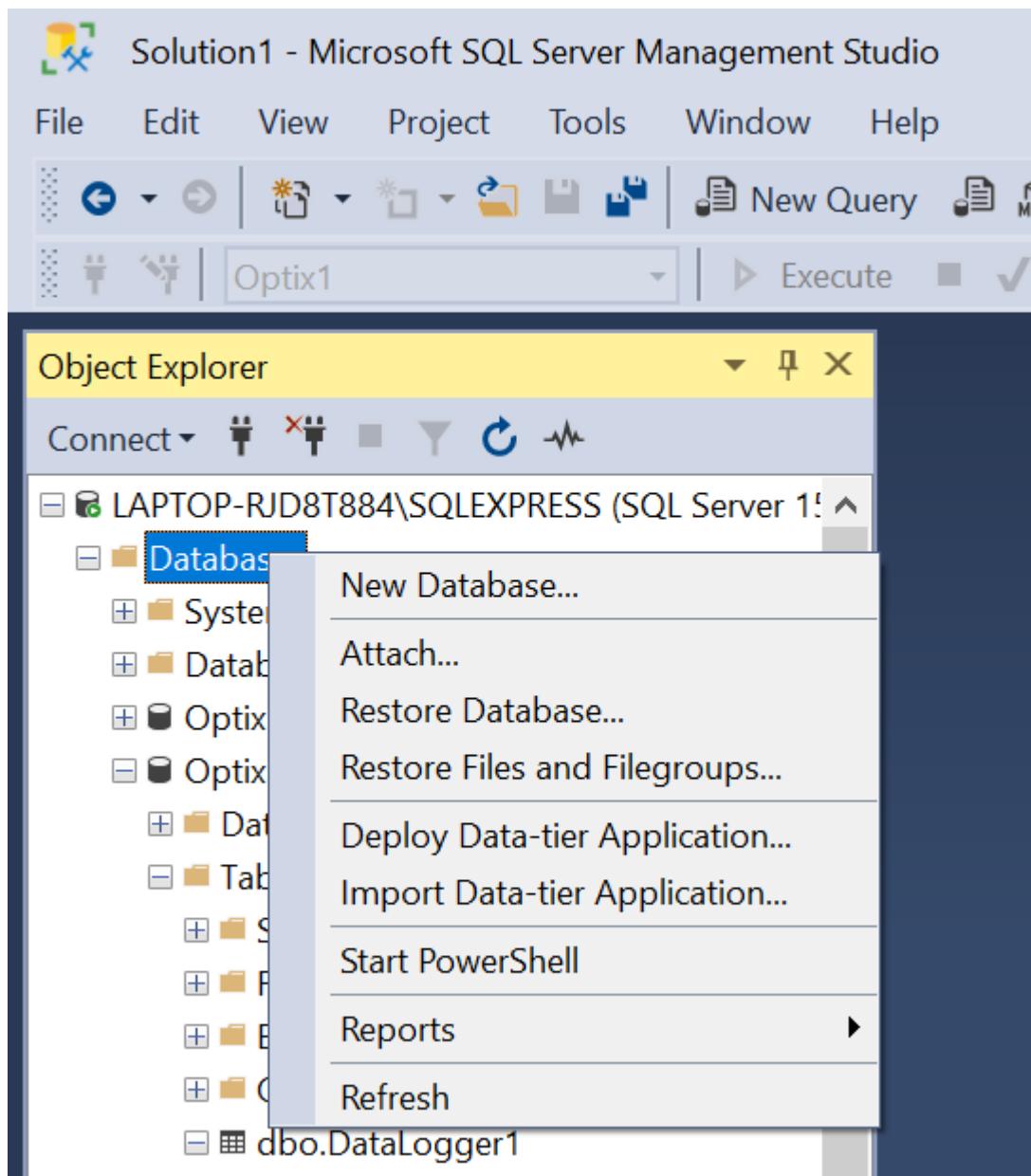
But let's start creating a complete new project

For instance sql_datalogger_optix_v2

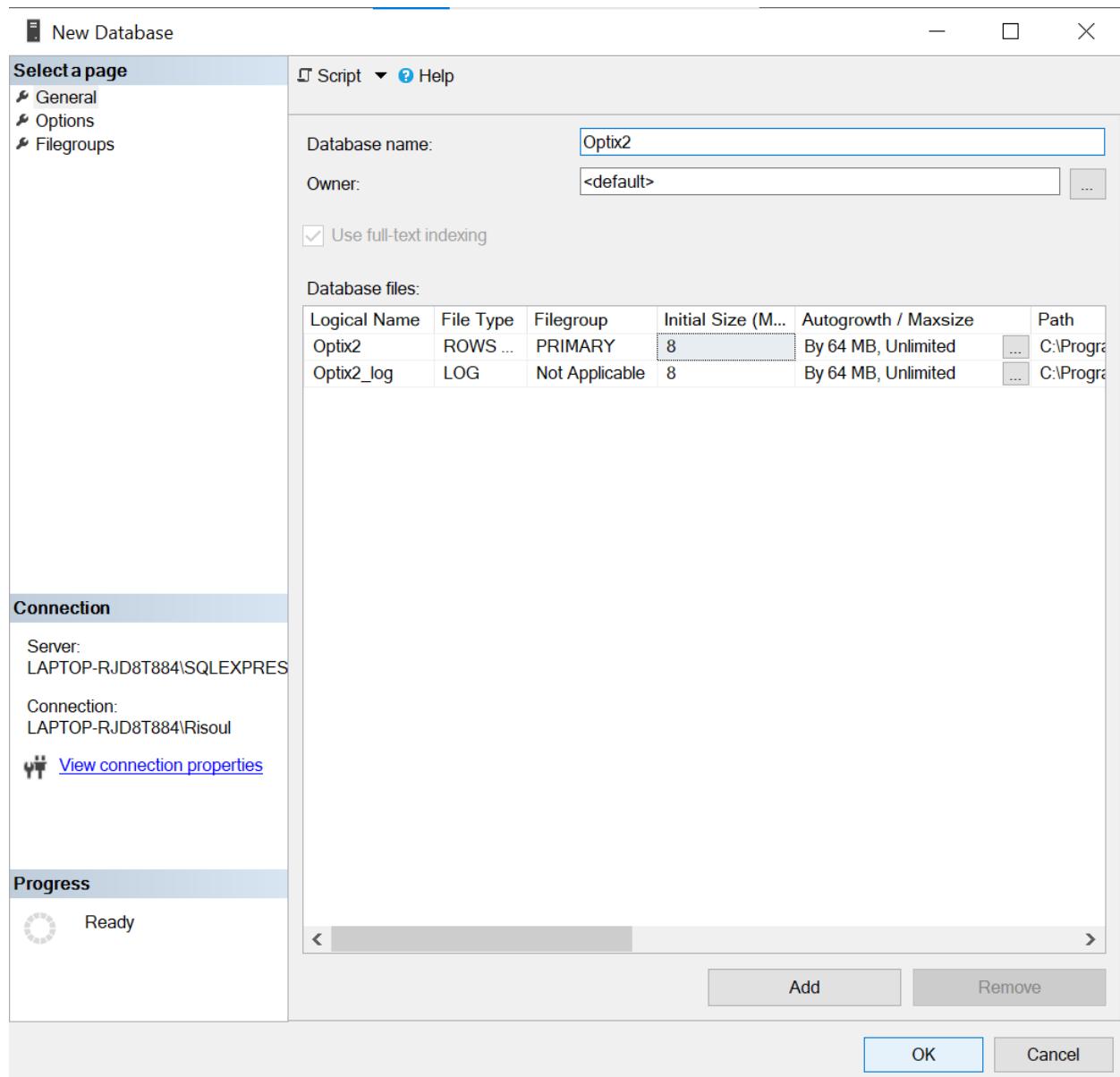


But first let's create a database without a Table (Optix and The Datalogger function will create the Table)

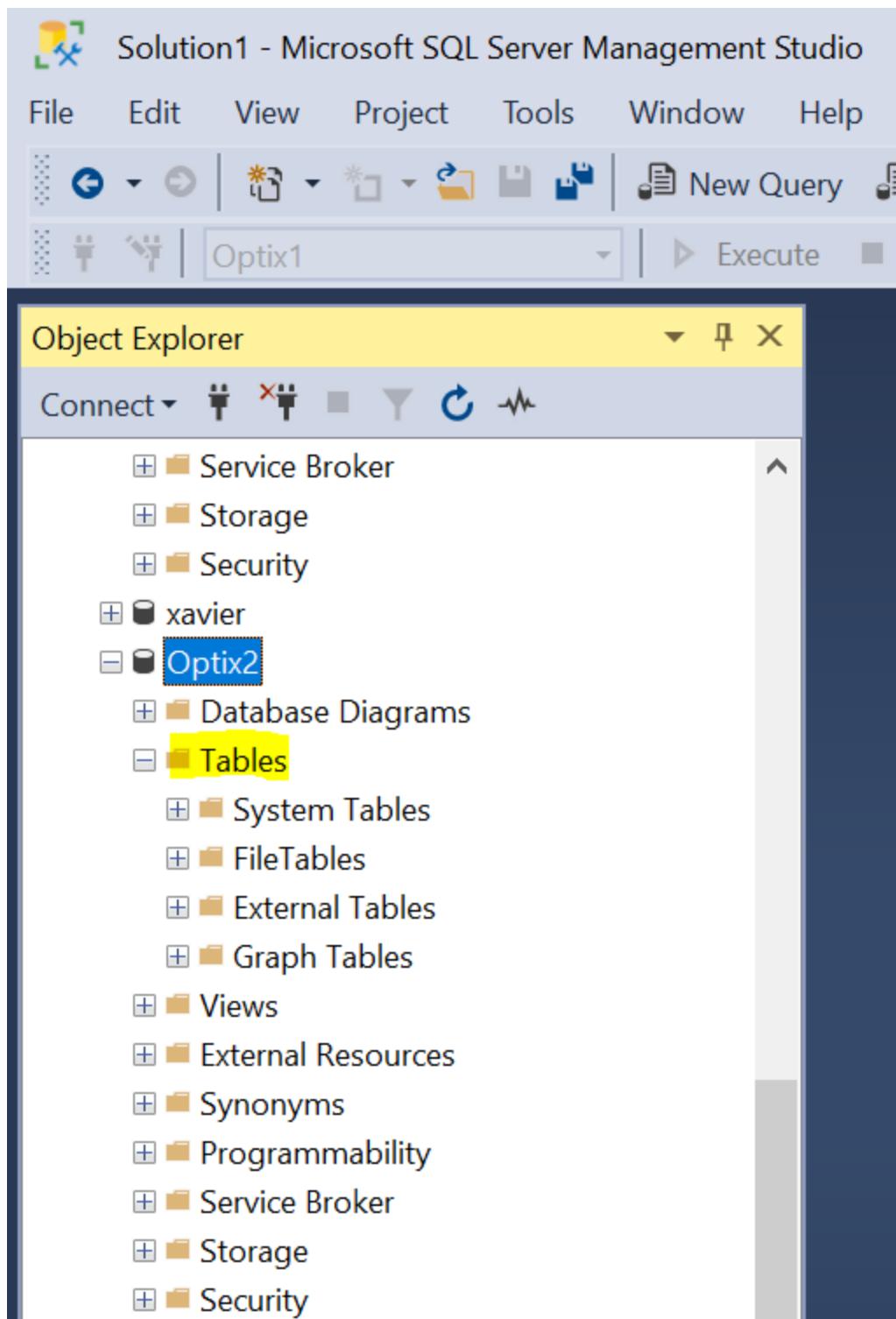
For instance let's create a new database Optix2



Just give a name, that's all

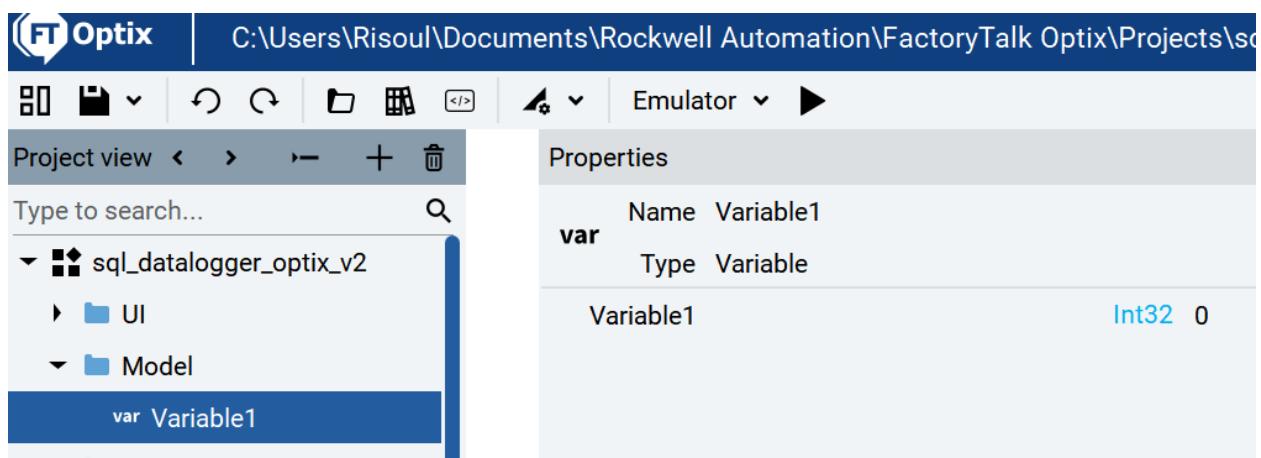
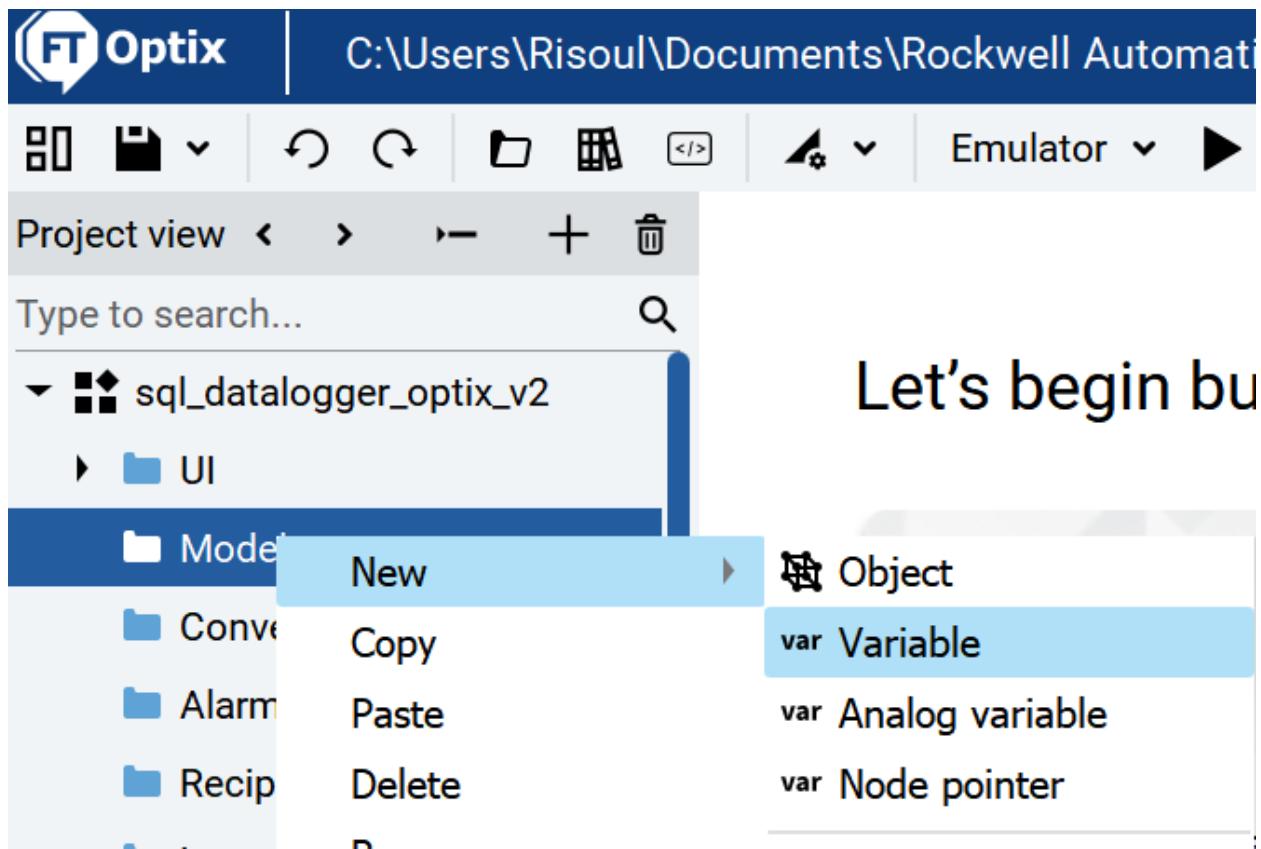


There are no user tables

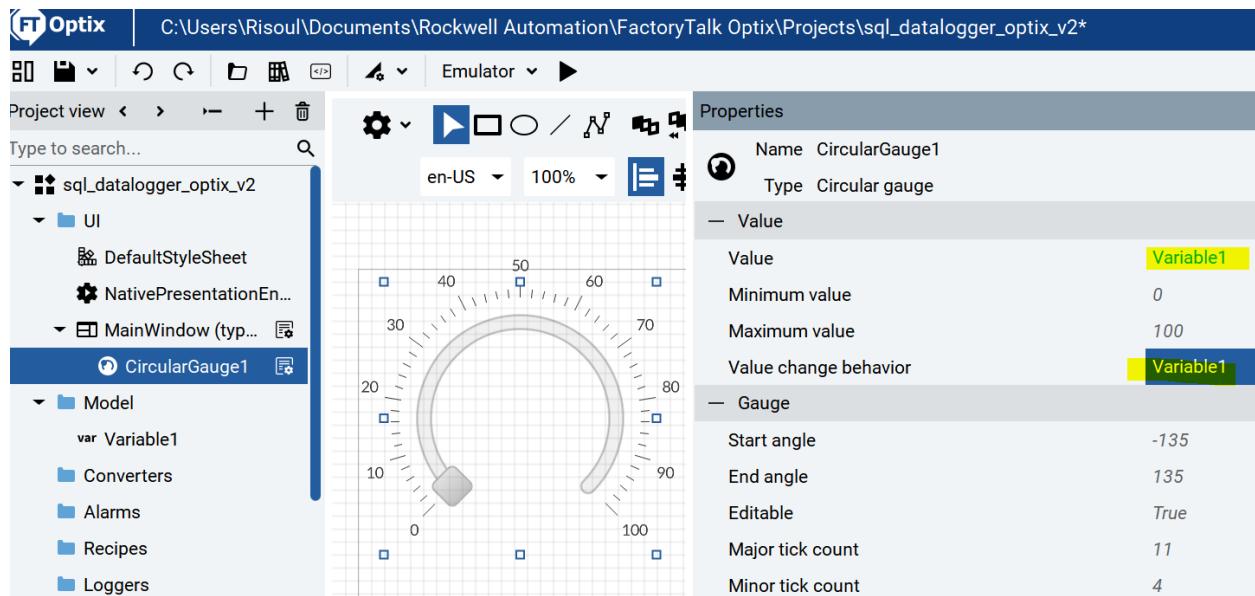


Now Let's go to Optix

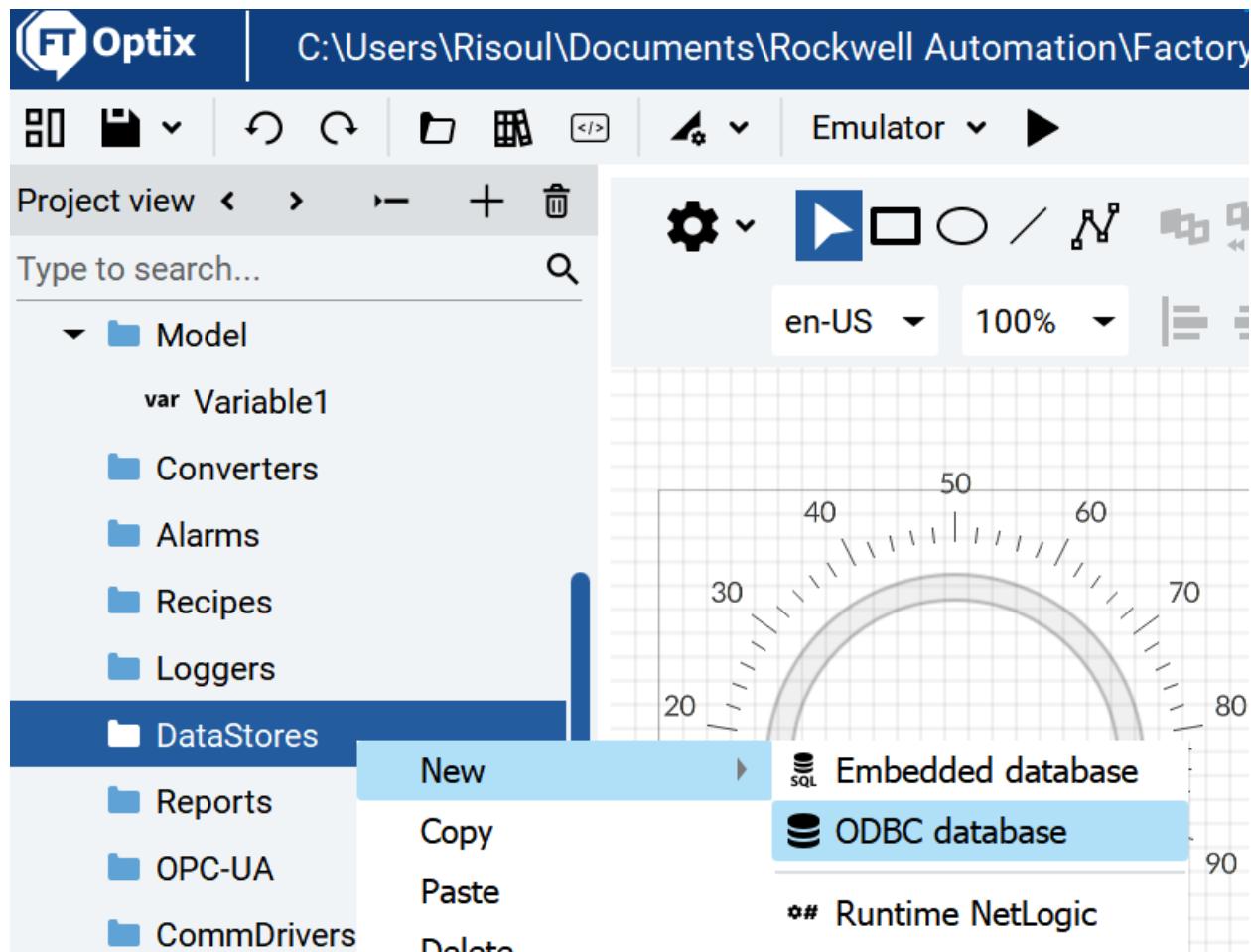
Add a variable to work with



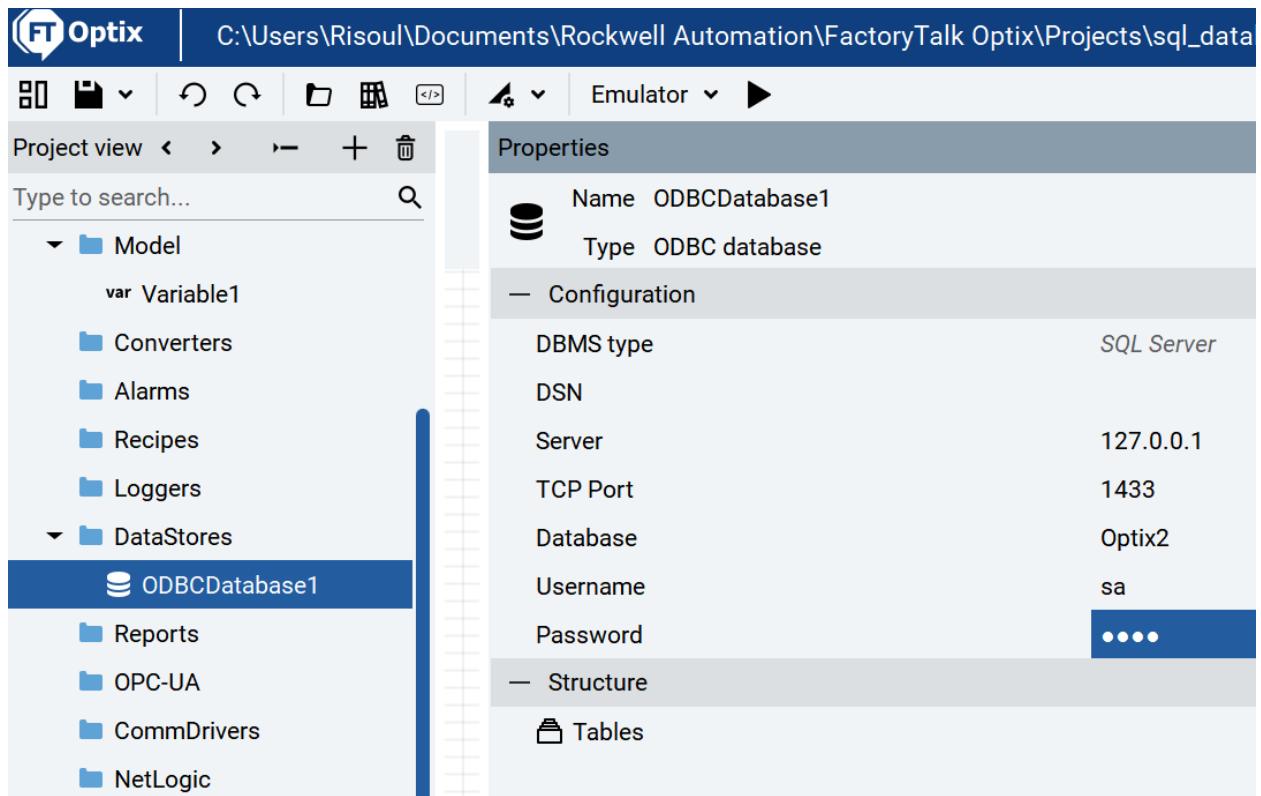
Add a Slider to enter values to such variable



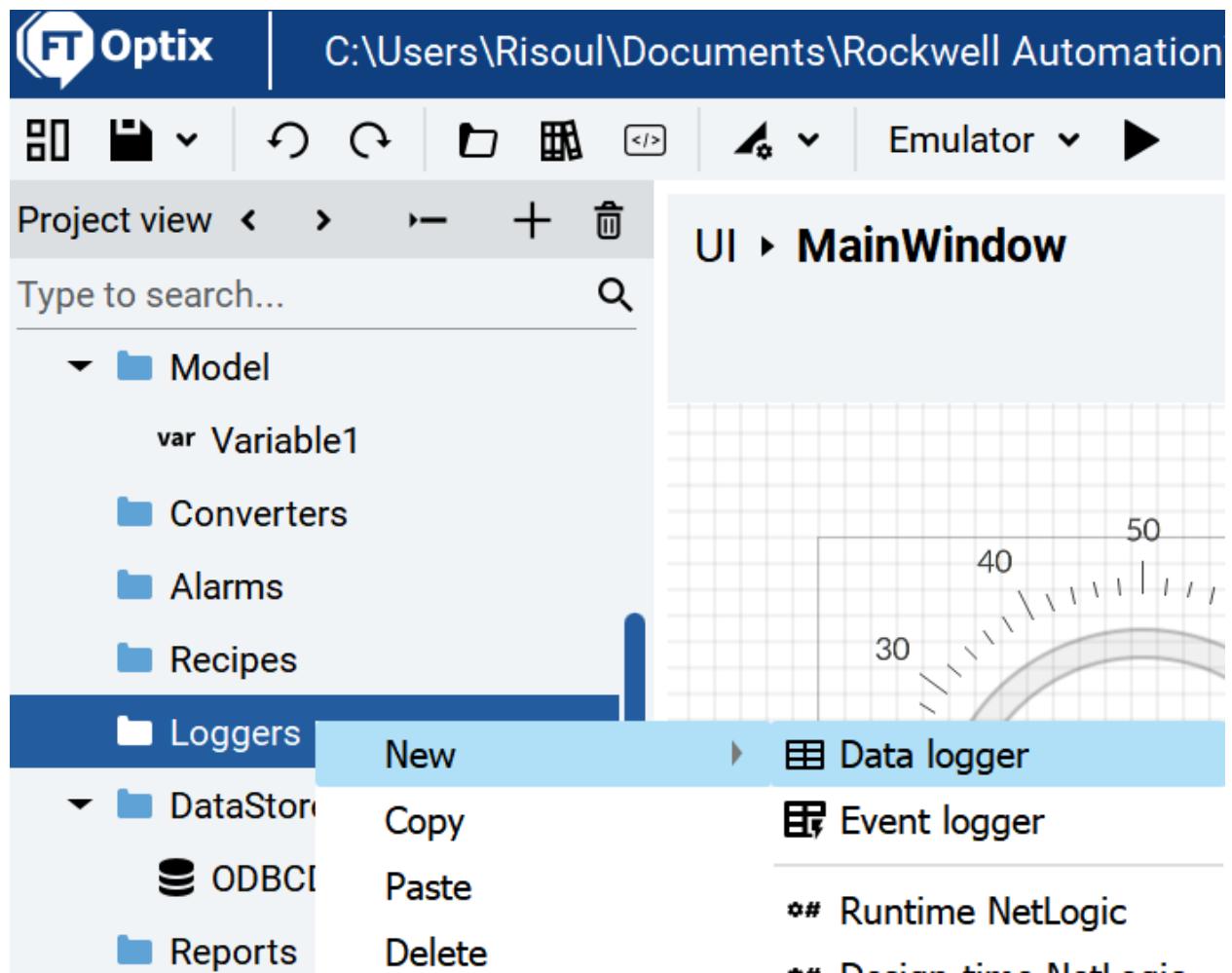
Next create the Database instance on Optix



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



Then create a new DataLogger



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

The screenshot shows the Rockwell Automation FactoryTalk Optix interface. The left pane displays a project tree with categories like Model, Loggers, and DataStores. A DataLogger1 object is selected in the tree. The right pane shows its properties:

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

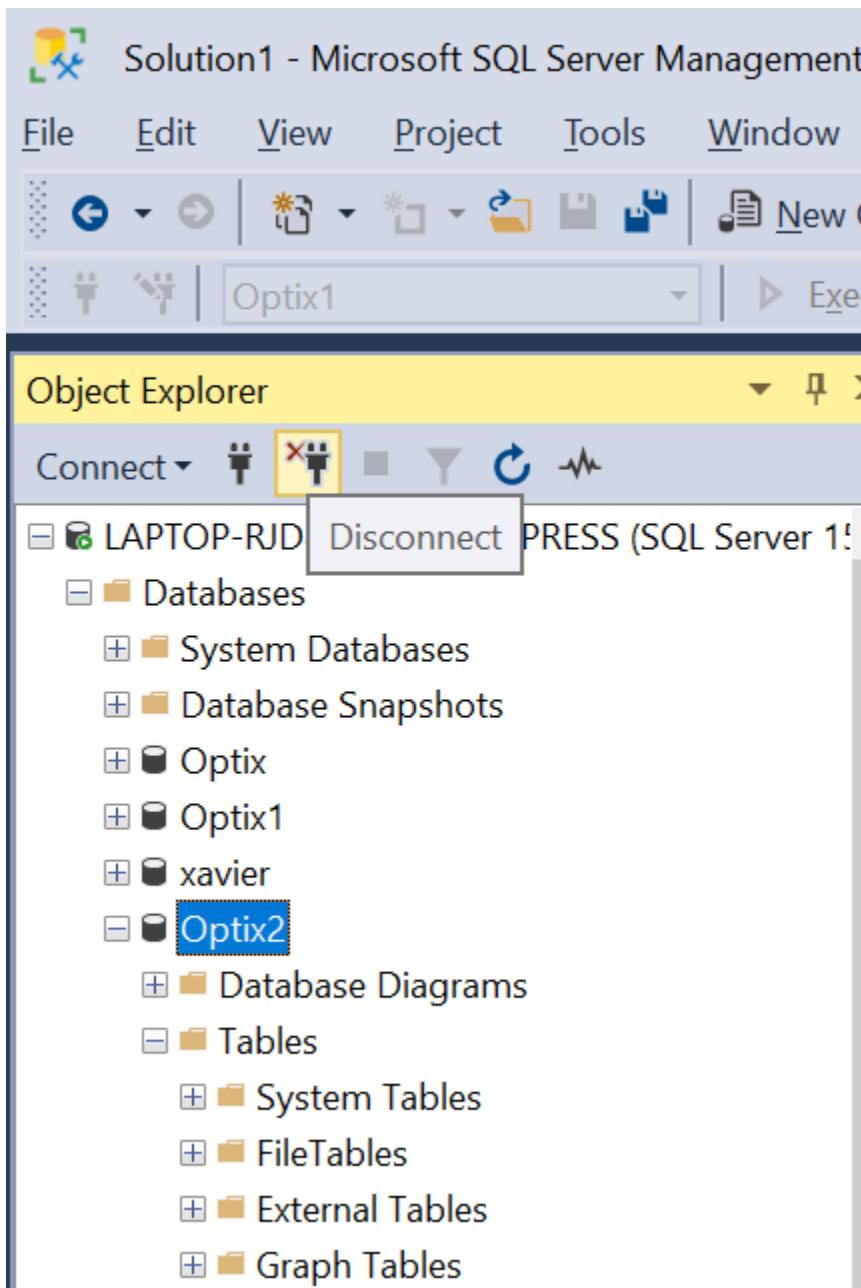
If we now open the ODBC object, surprise, a new Table an columns have been created for us

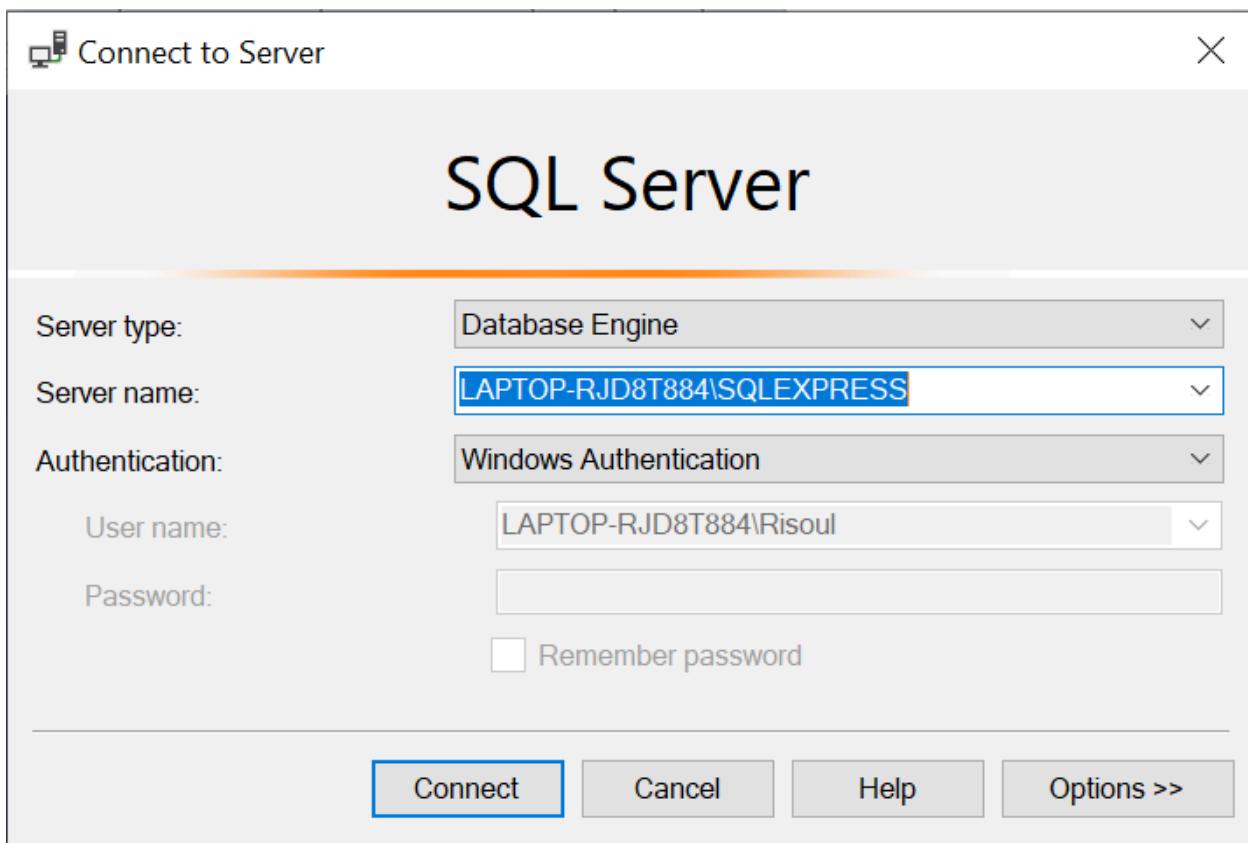
The screenshot shows the Rockwell Automation FactoryTalk Optix interface. The left pane displays a project tree with categories like Model, Loggers, and DataStores. An ODBCDatabase1 object is selected in the tree. The right pane shows its properties:

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

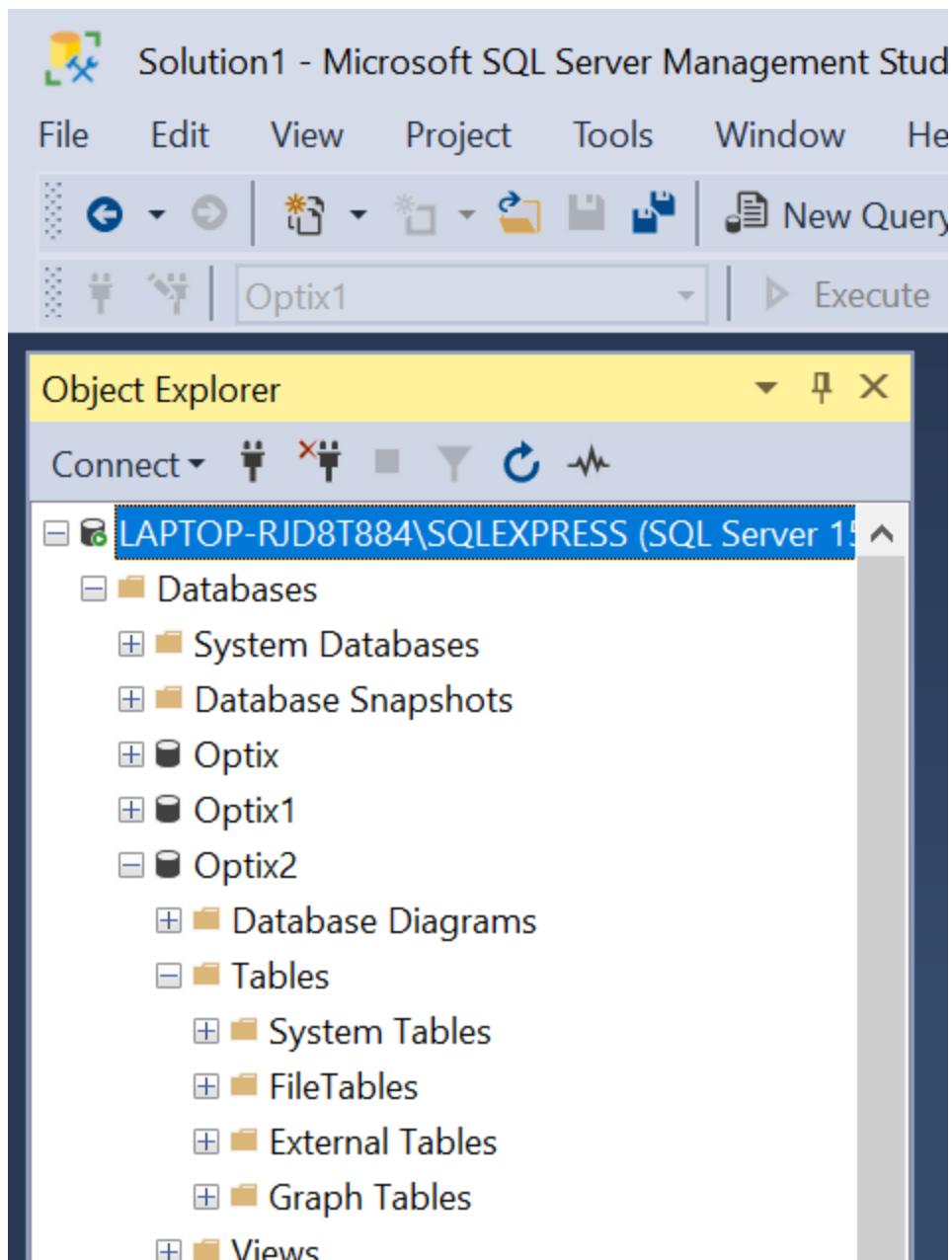
If we take a look at the database:

You have to disconnect and connect again to see the changes



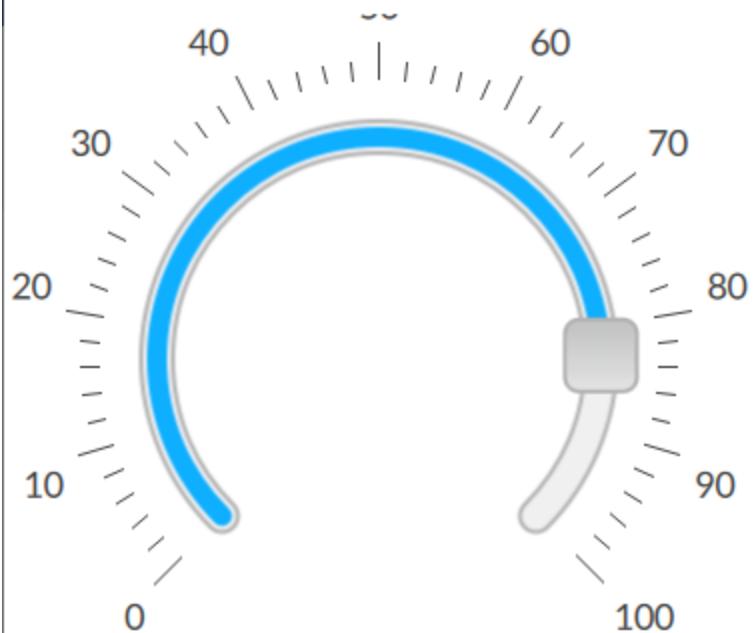


Still nothing



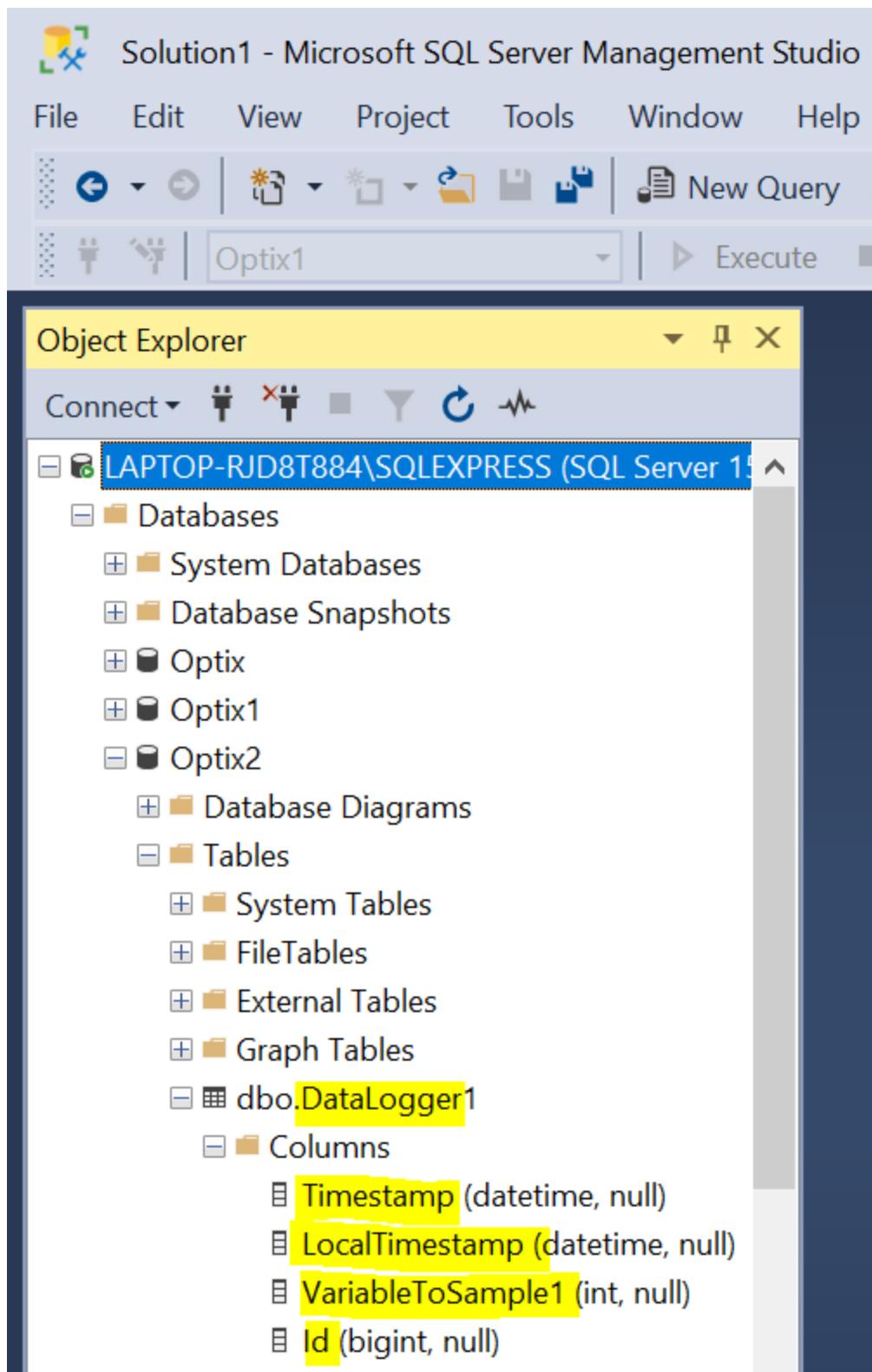


FTOptixRuntime



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

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer on the left, under the database 'Optix2', there is a table named 'dbo.DataLogger1'. The 'Tables' node is expanded, showing columns: Timestamp, LocalTimestamp, VariableToSample1, and Id. A query window titled 'SQLQuery5.sql' is open, displaying the following T-SQL code:

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

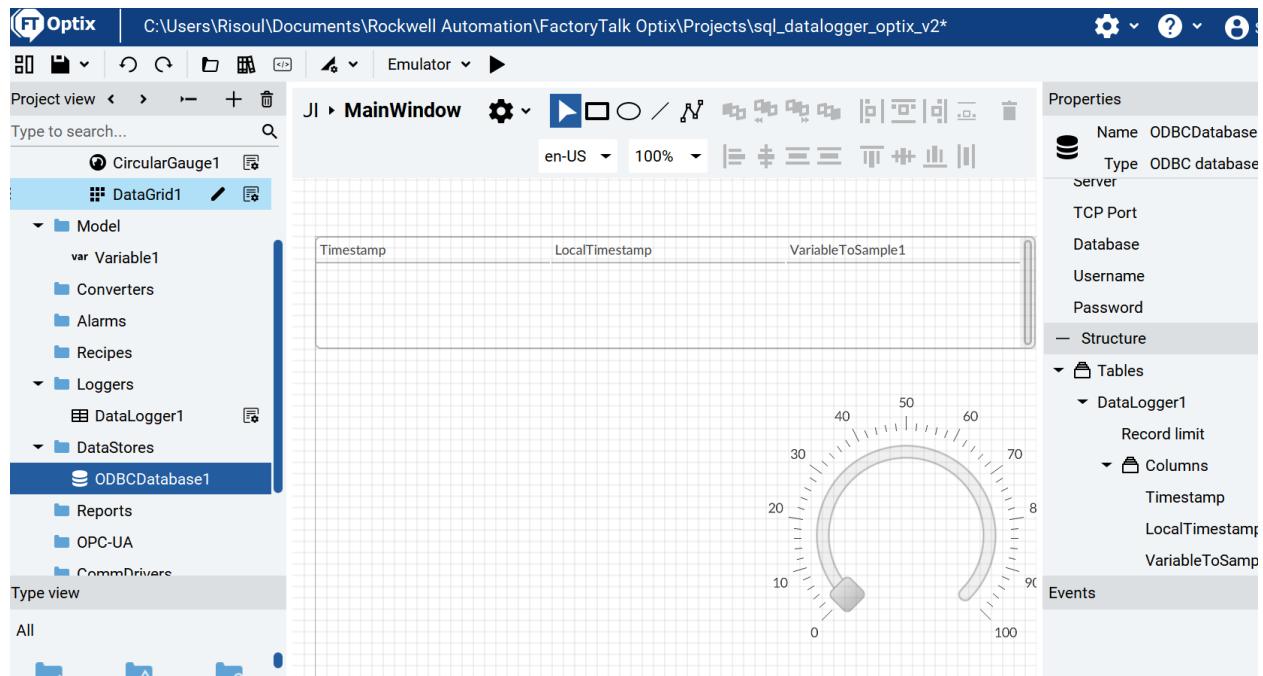
The results grid shows the following data:

	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

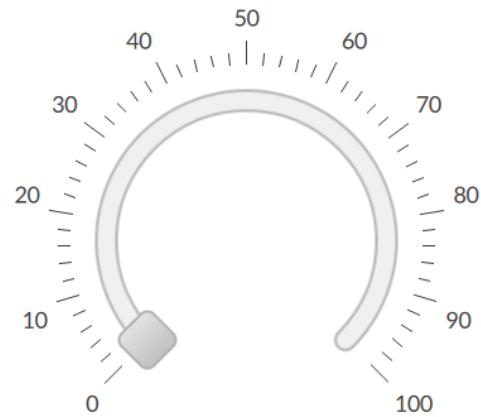
Drag and drop the table to the Datagrid

The screenshot shows the Rockwell Automation FactoryTalk Optix software interface. On the left, the 'Project view' pane shows a tree structure with nodes like CircularGauge1, DataGrid1, Model, Loggers, and DataStores. Under DataStores, 'ODBCDatabase1' is selected. In the center workspace, there is a circular gauge control. On the right, the 'Properties' panel is open for 'ODBCDatabase1', showing the 'Tables' section with 'DataLogger1' selected. The 'Columns' section lists 'Timestamp', 'LocalTimestamp', 'VariableToSample1', and 'Id'. The 'Events' section is also visible.



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

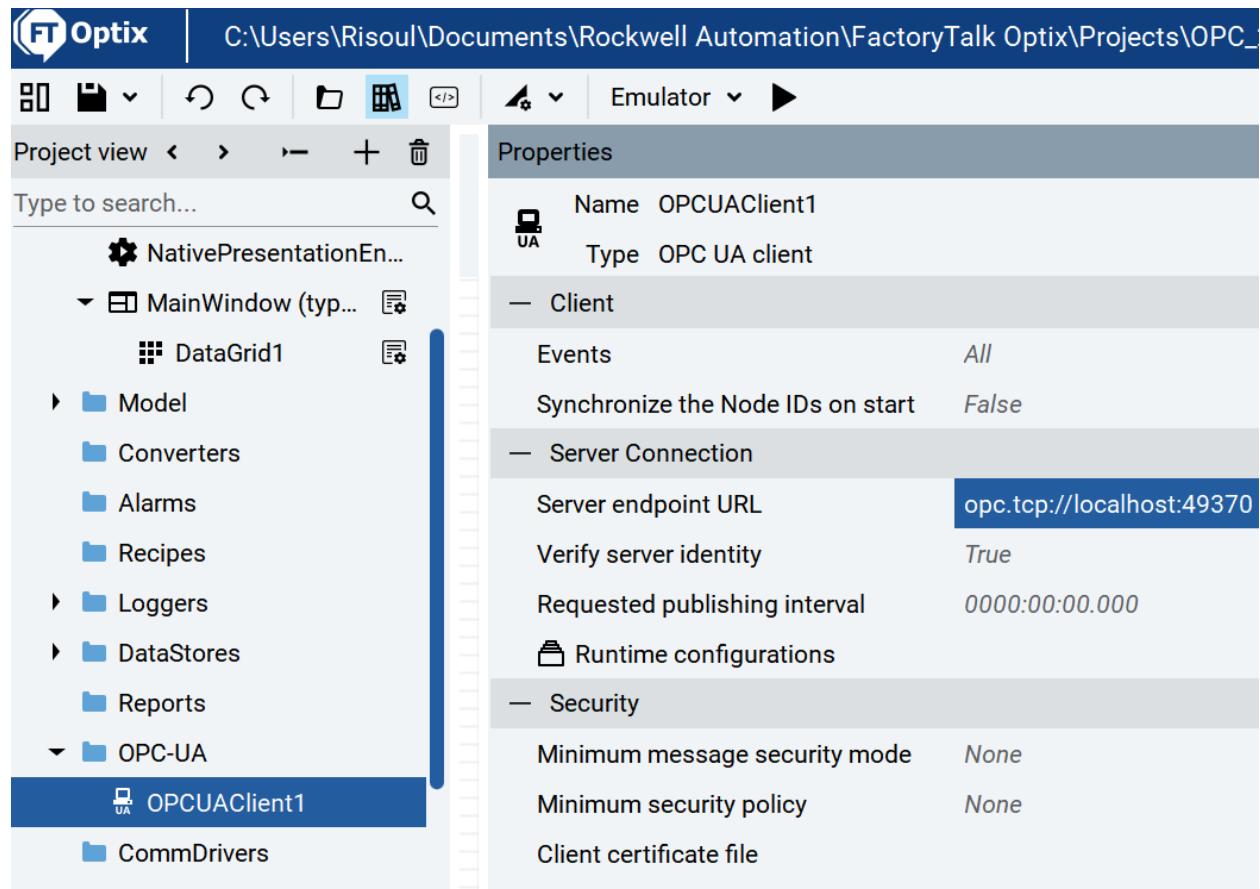
FTOptixRuntime		
Timestamp	LocalTimestamp	VariableToSample1
Jan 21, 2023, 12:19:09 PM	Jan 21, 2023, 1:19:09 PM	54
Jan 21, 2023, 12:19:10 PM	Jan 21, 2023, 1:19:10 PM	100
Jan 21, 2023, 12:19:11 PM	Jan 21, 2023, 1:19:11 PM	100



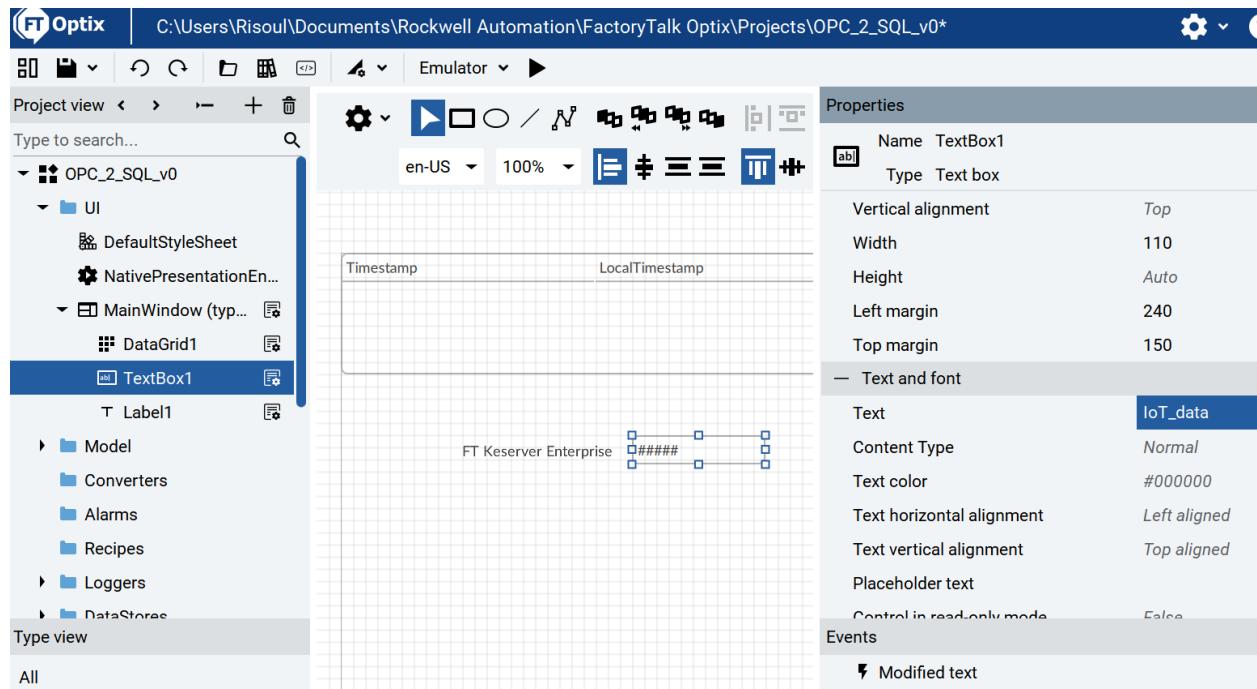
9. Creating a OPC UA to SQL converter

Take last SQL project and delete the gauge.

Then add the OPC UA client like in section 6.



Now we have access to IoT_data Tag



First of all test that we are reading OPC UA data

Yes

The screenshot shows the 'FTOptixRuntime' application window. It contains a DataGrid with three columns: 'Timestamp', 'LocalTimestamp', and 'VariableToSample1'. The data is as follows:

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

At the bottom of the window, there is a status bar with the text 'FT Keserver Enterprise' and a yellow-bordered text input field containing the value '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 title bar reads "SQLQuery1.sql - LAPTOP-RJD8T884\SQLEXPRESS.Optix2 (LAPTOP-RJD8T884\Risoul (55))* - Microsoft SQL Server". The menu bar includes File, Edit, View, Query, Project, Tools, Window, and Help. The toolbar has various icons for database management. The Object Explorer on the left shows the database structure: "LAPTOP-RJD8T884\SQLEXPRESS (SQL Server 1)" contains "Databases" (System Databases, Database Snapshots), "Optix" (Optix, Optix1, Optix2), and "Tables" (System Tables, FileTables, External Tables, Graph Tables). Under "Optix2", there is a "dbo.DataLogger1" table with columns: Timestamp (datetime, null), LocalTimestamp (datetime, null), VariableToSample1 (int, null), and Id (bigint, null). The main window displays a query editor with the following T-SQL code:

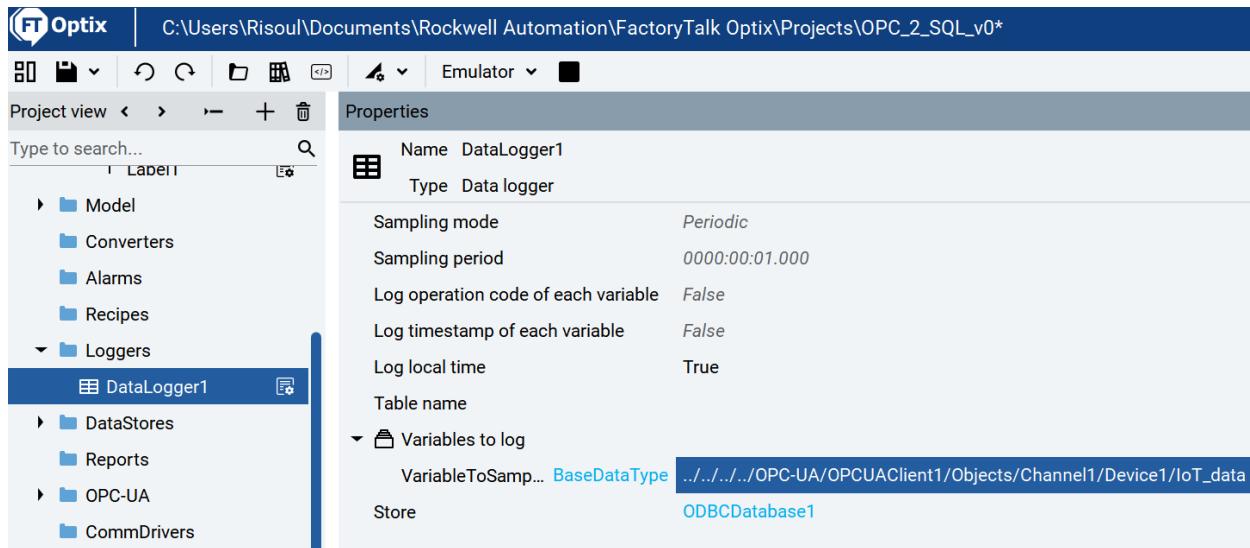
```
use Optix2;
delete from DataLogger1;
select * from DataLogger1;
```

Verify that the data is erased from Optix

The screenshot shows the FTOptixRuntime application window. The title bar says "FTOptixRuntime". The main area is a table with three columns: "Timestamp", "LocalTimestamp", and "VariableToSample1". Below the table, there is a status bar with the text "FT Kserver Enterprise" and a small box containing the number "45,137".

Timestamp	LocalTimestamp	VariableToSample1

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.

The screenshot shows Microsoft SQL Server Management Studio (SSMS) with a query window titled 'SQLQuery1.sql - LAPTOP-RJD8T884\SQLEXPRESS.Optix2 (LAPTOP-RJD8T884\Risoul (55))*'. The query is:

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

The results grid shows the following data:

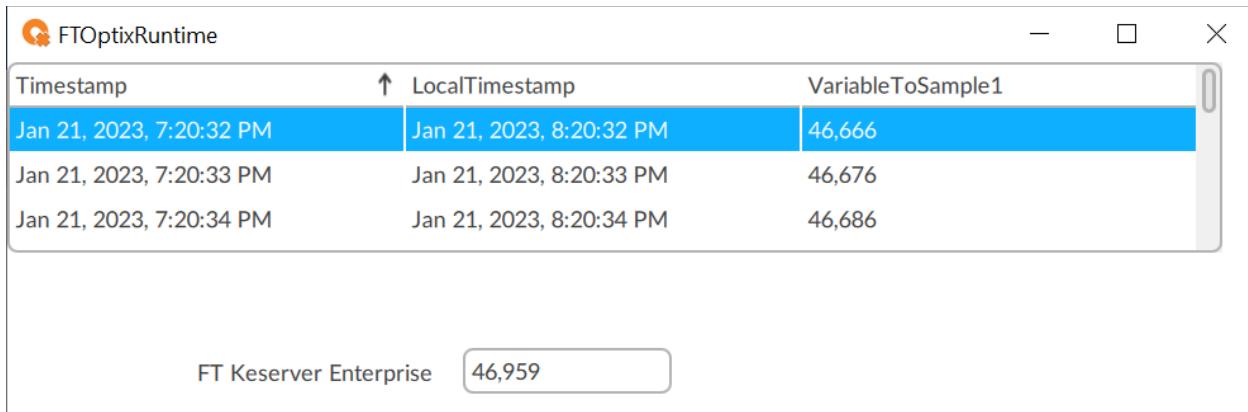
Timestamp	LocalTimestamp	VariableToSample1	Id	
147	2023-01-21 19:15:31.540	2023-01-21 20:15:31.540	0	1...
148	2023-01-21 19:15:32.540	2023-01-21 20:15:32.540	0	1...
149	2023-01-21 19:15:33.540	2023-01-21 20:15:33.540	0	1...
150	2023-01-21 19:15:50.237	2023-01-21 20:15:50.237	46471	1...
151	2023-01-21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

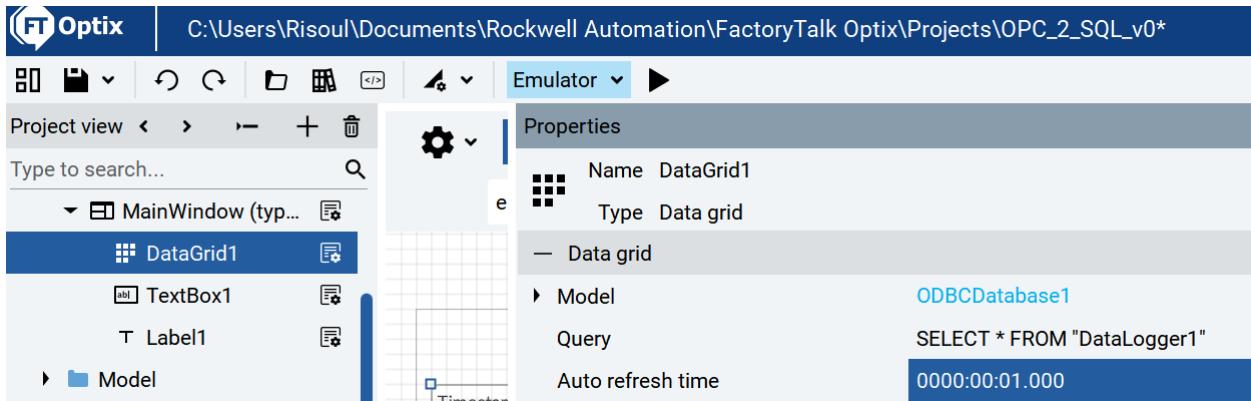
First time we see nothing on the Optix runtime datagrid window

Let's close the runtime and open it again

Now it works



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



FTOptixRuntime

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://www.youtube.com/watch?v=irL_CHUZaHY

The screenshot shows the FactoryTalk Optix Studio environment. On the left, there's a Project view pane with a search bar and a tree view of project components. In the center, a main window displays a data grid with columns: Item ID, Data Type, Value, Timestamp, and Quality. A text input field labeled "FT Keserver Enterprise OPC UA value" is also present. On the right, a separate window titled "FTOptixRuntime" shows a table with timestamped data for "VariableToSample1". Below these windows, a SQL Server Management Studio (SSMS) window is open, showing two queries in Object Explorer and Results panes. The first query is a simple SELECT statement from a table named "DataLogger1". The results pane shows a table with columns: Timestamp, LocalTimestamp, VariableToSample1, and Id, containing four rows of data.

Timestamp	LocalTimestamp	VariableToSample1	Id
2023-01-21 19:46:28.237	2023-01-21 20:46:28.237	55590	661
2023-01-21 19:46:27.237	2023-01-21 20:46:27.237	55579	660
2023-01-21 19:46:26.237	2023-01-21 20:46:26.237	55568	659
2023-01-21 19:46:25.237	2023-01-21 20:46:25.237	55557	658

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

Just use the example from chapter 12

But change some lines on subscribelogic

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...

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 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



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.HMIPrjct;
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 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()
    {
```

```

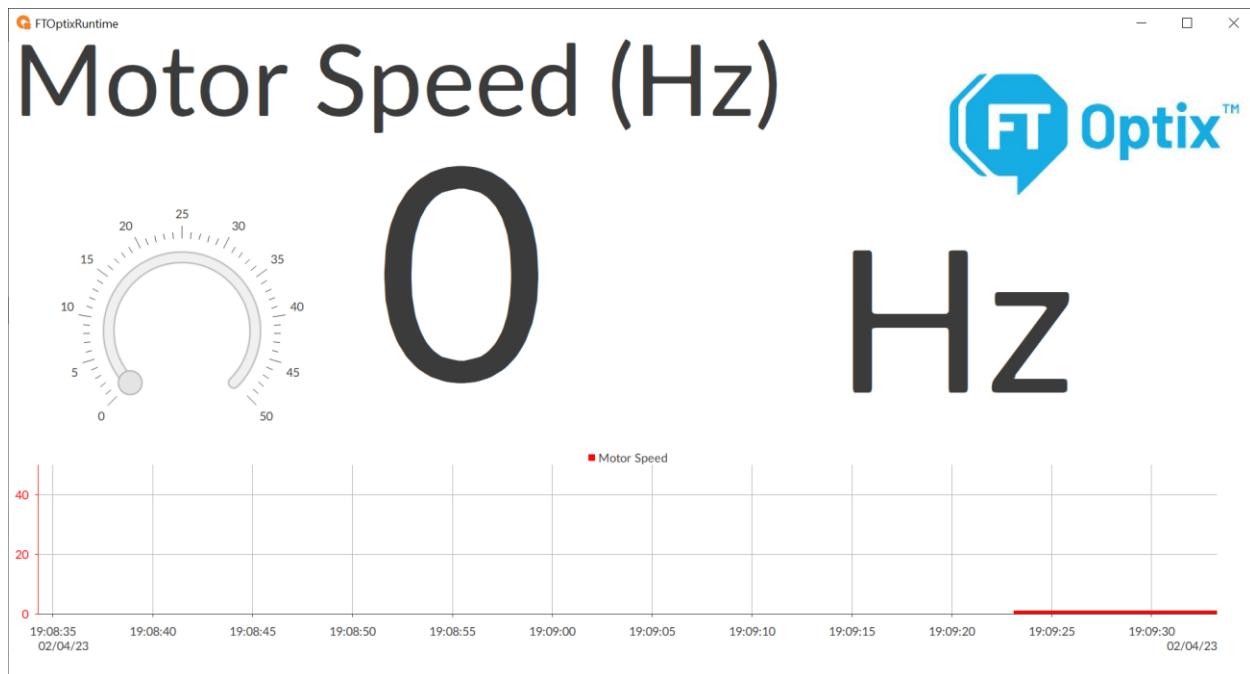
        subscribeClient.Unsubscribe(new string[] { "v3/smartbridge-sinci-
demo@ttn/devices/smartbridge-modbus-demo/up" });
        subscribeClient.Disconnect();
    }

    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



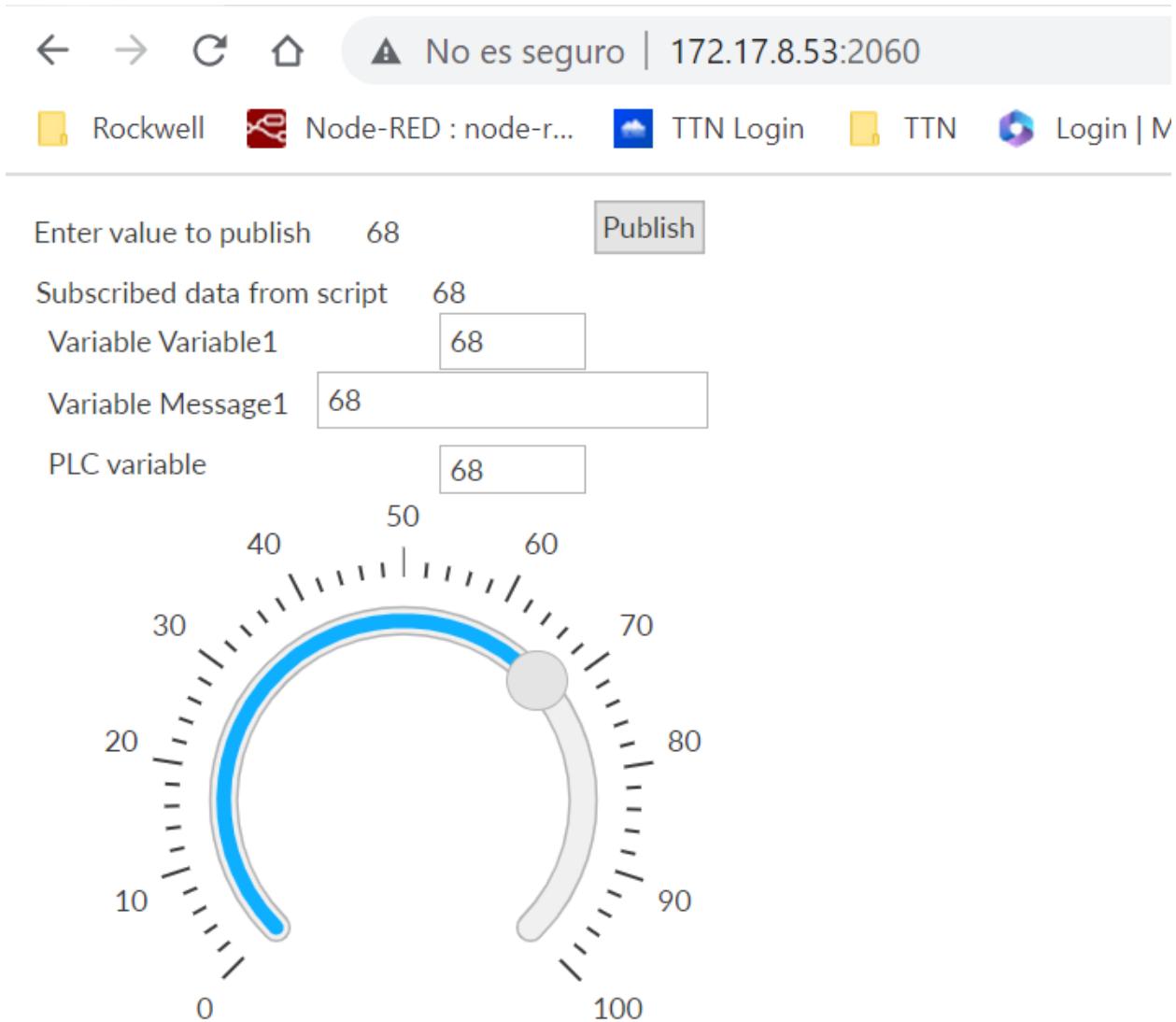
11. Web access

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.

Property	Value
Name	WebPresentationEngine1
Type	Web presentation engine
Dynamic variables polling time	0000:00:01.000
Protocol	http
IP address	172.17.8.53
Port	2060
Hostname	CATPROLP1003-23
Maximum number of connections	3
LoginWindow	

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

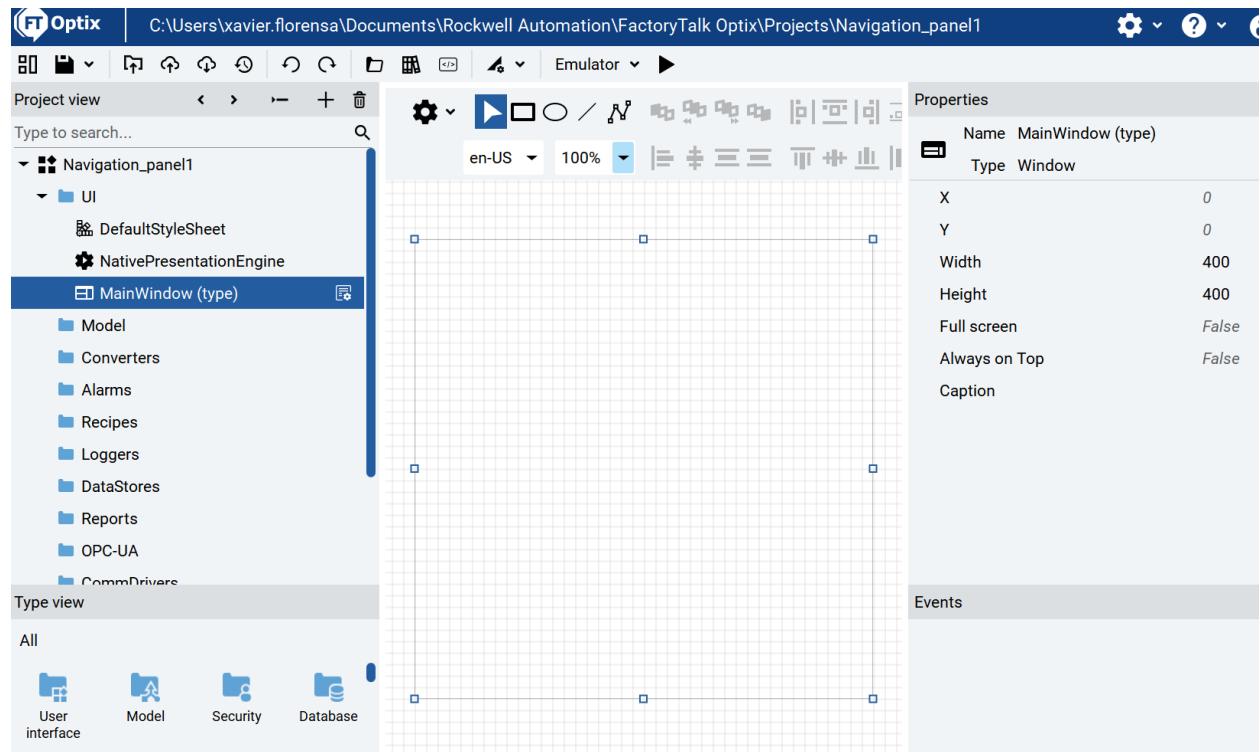
https://www.youtube.com/watch?v=78_cwp0iSJA

12. Navigation panels

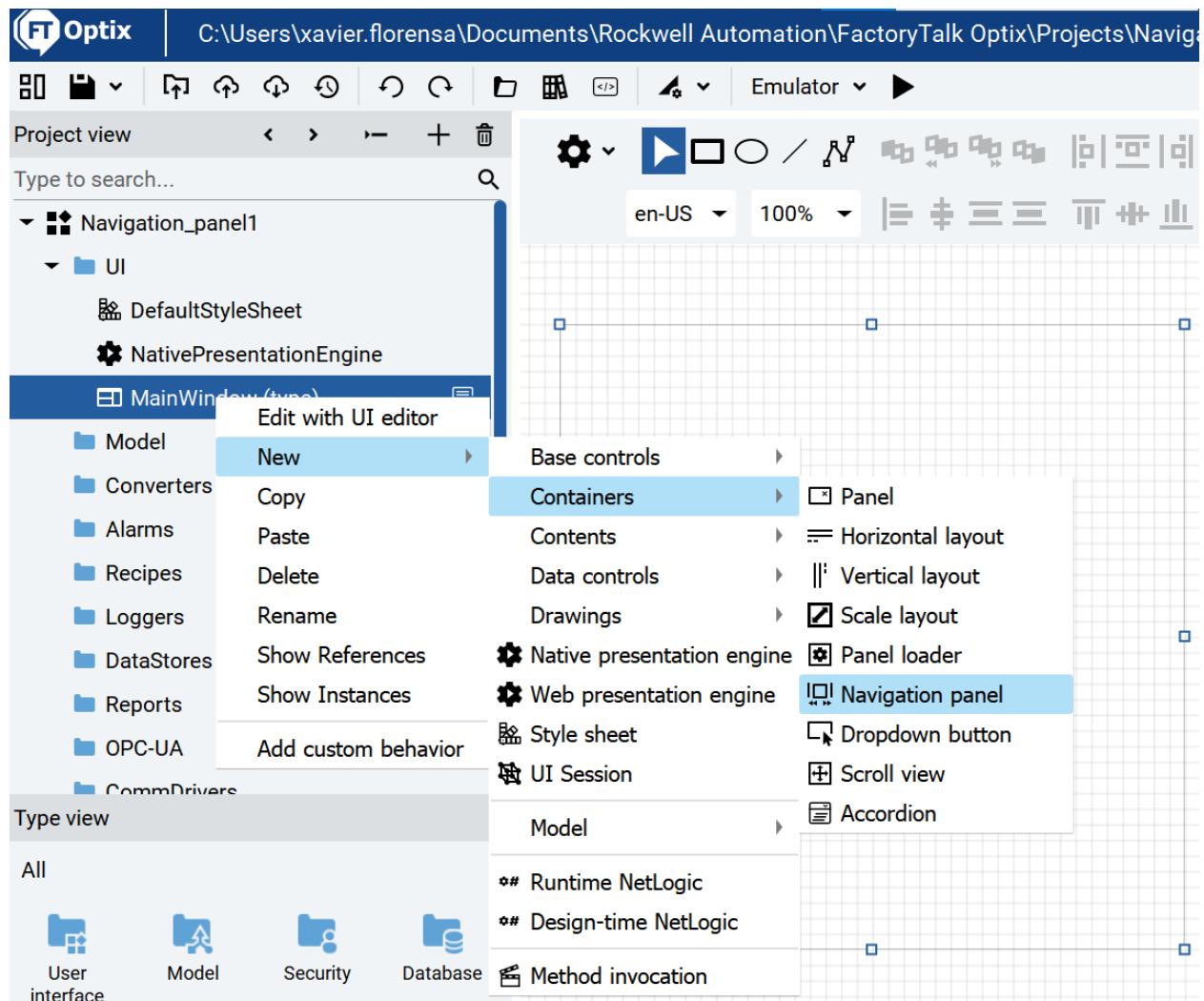
Start with this if you want to have different windows or panels visible on your project.

Create a blank new project

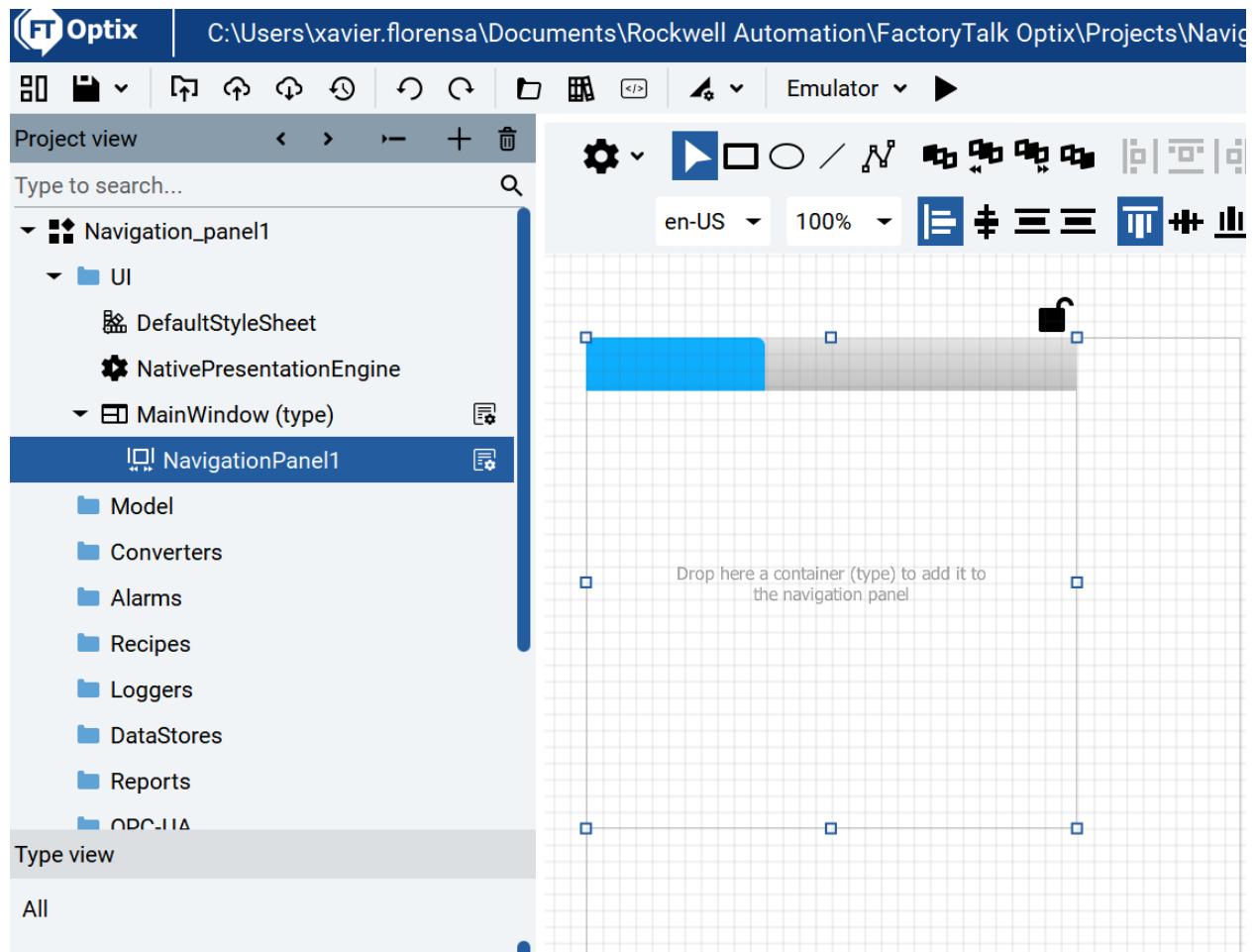
Go to Main window



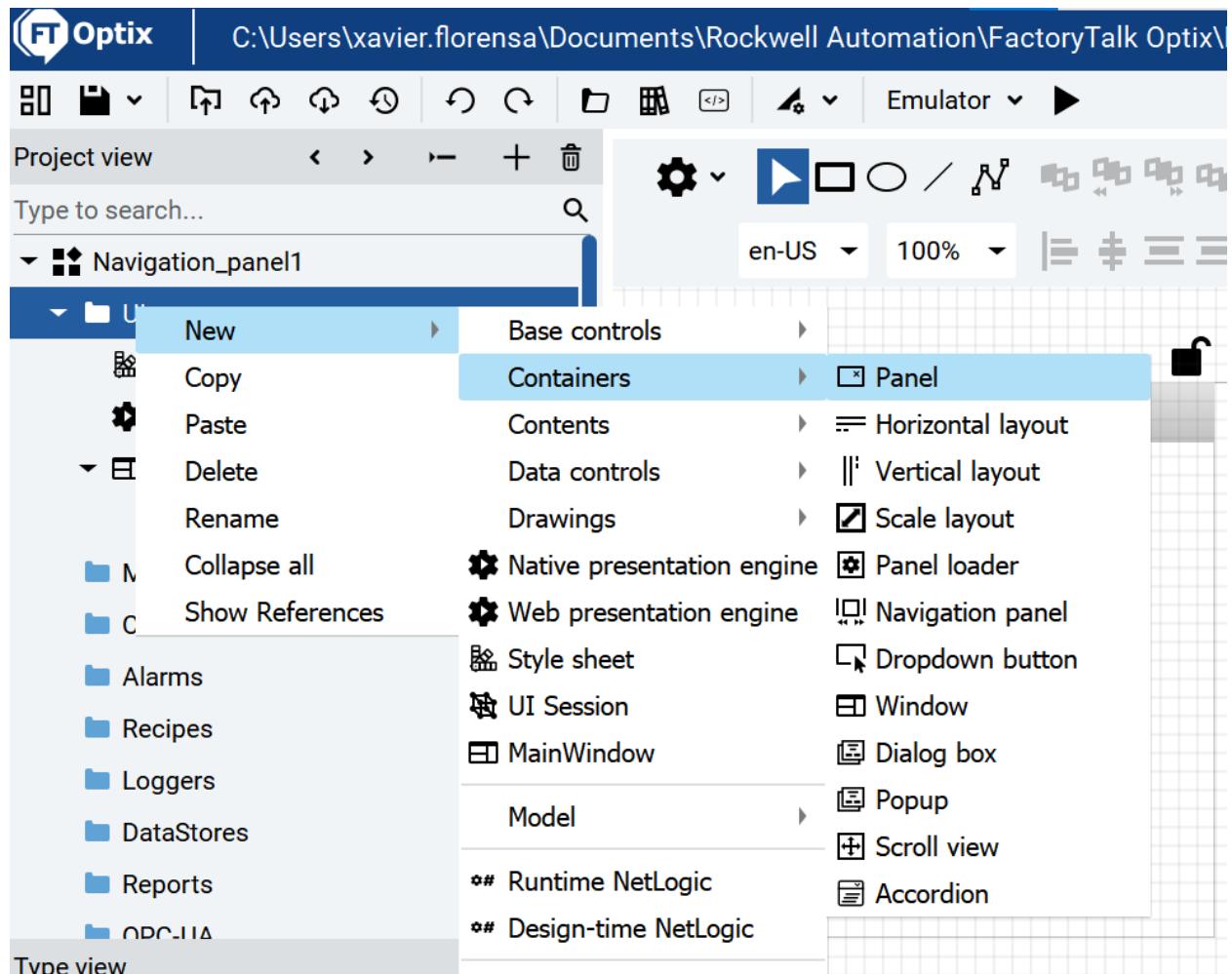
Create a Navigation Pannel with right button click



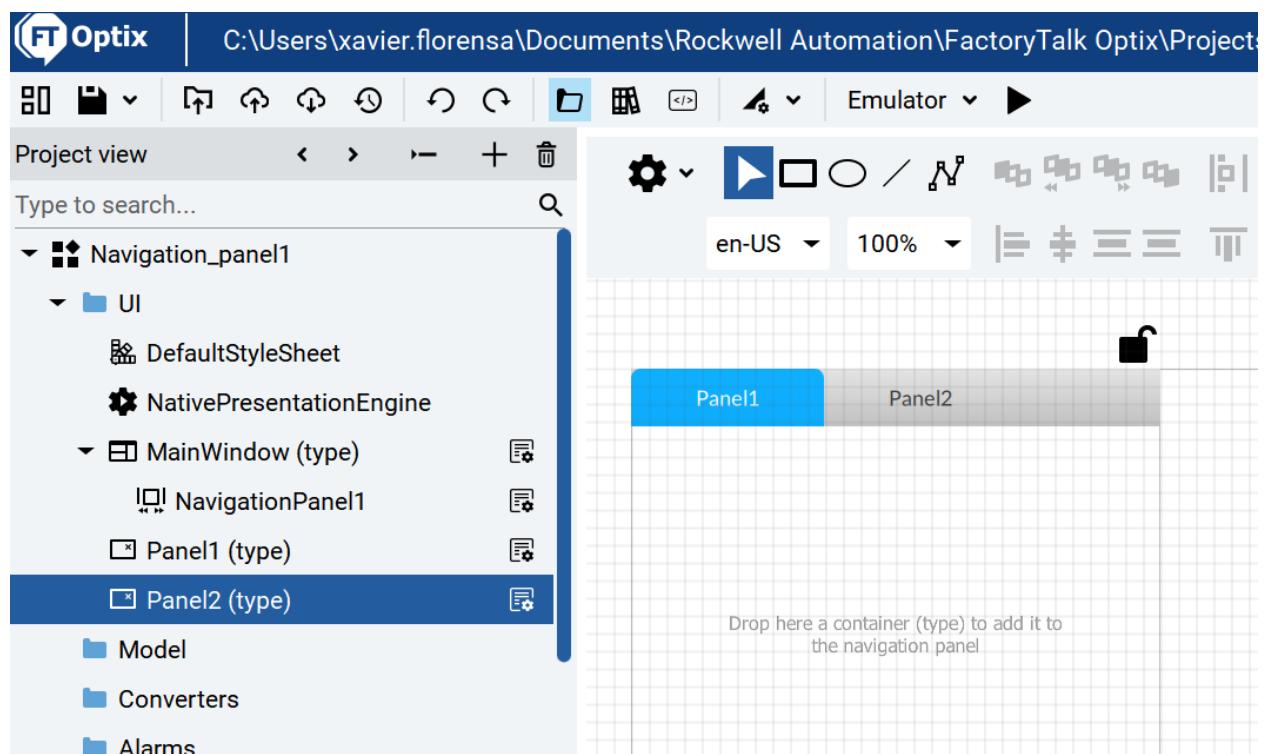
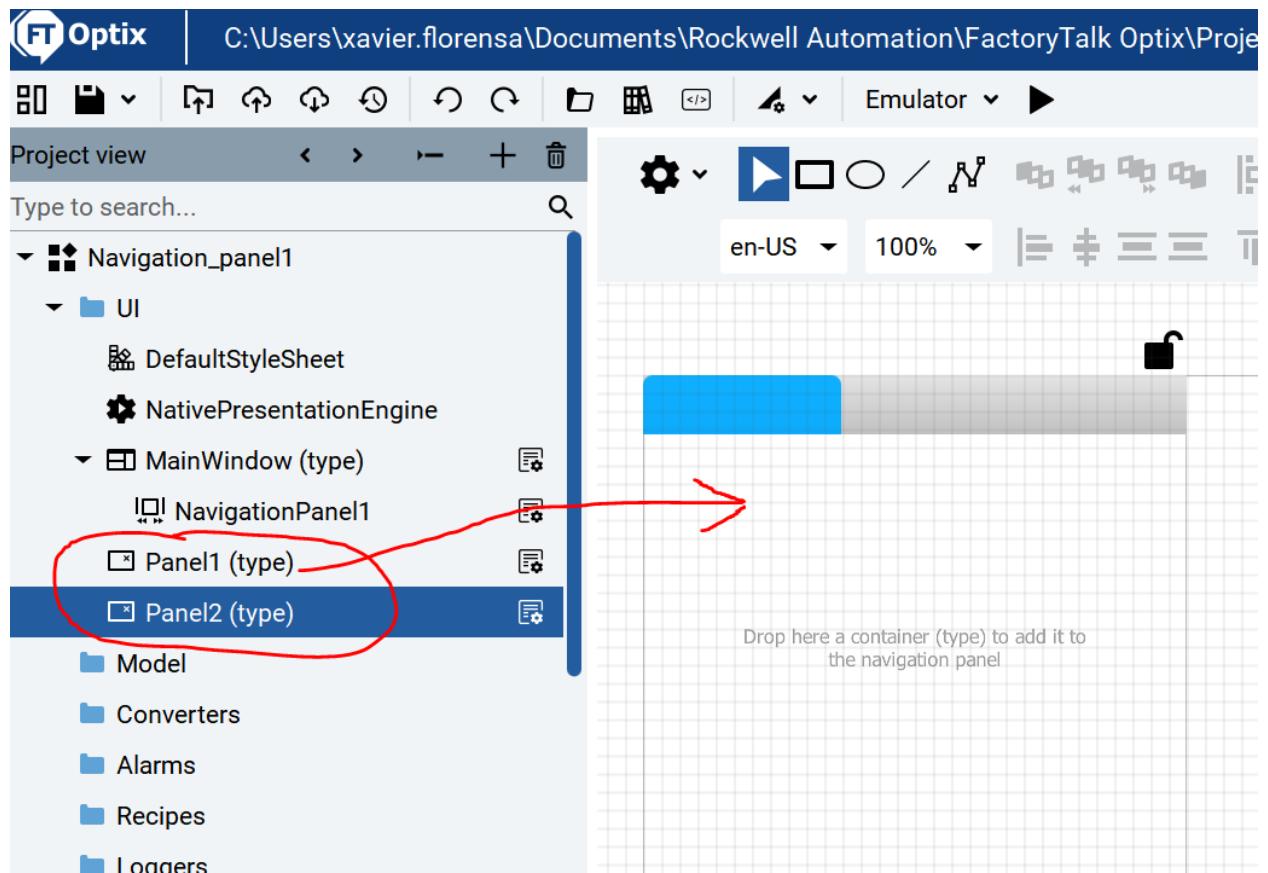
You will see this



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



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.

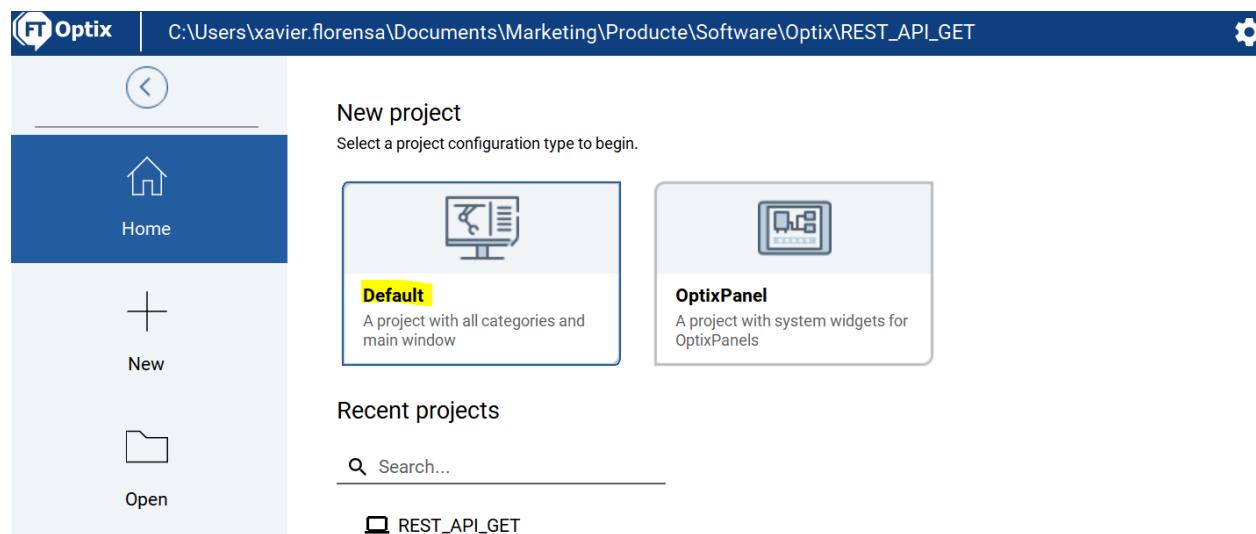
You can see a demo on Navigation panels and web access on this video

https://www.youtube.com/watch?v=78_cwp0iSJA

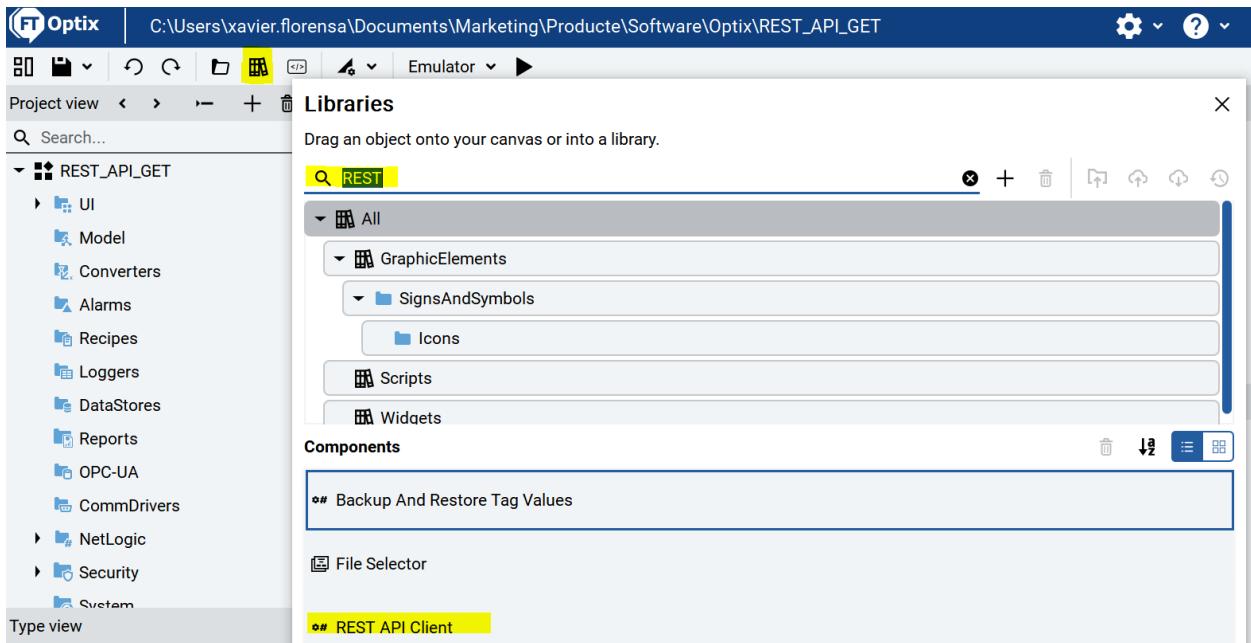
13. Creating an Http REST API client

13.1. GET request

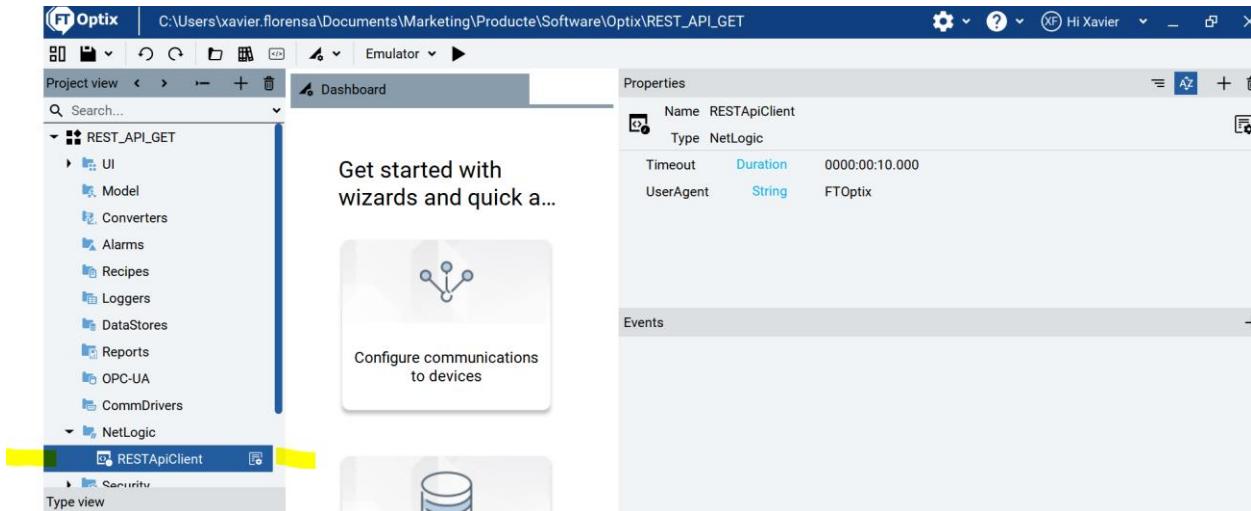
Create a new Optix project



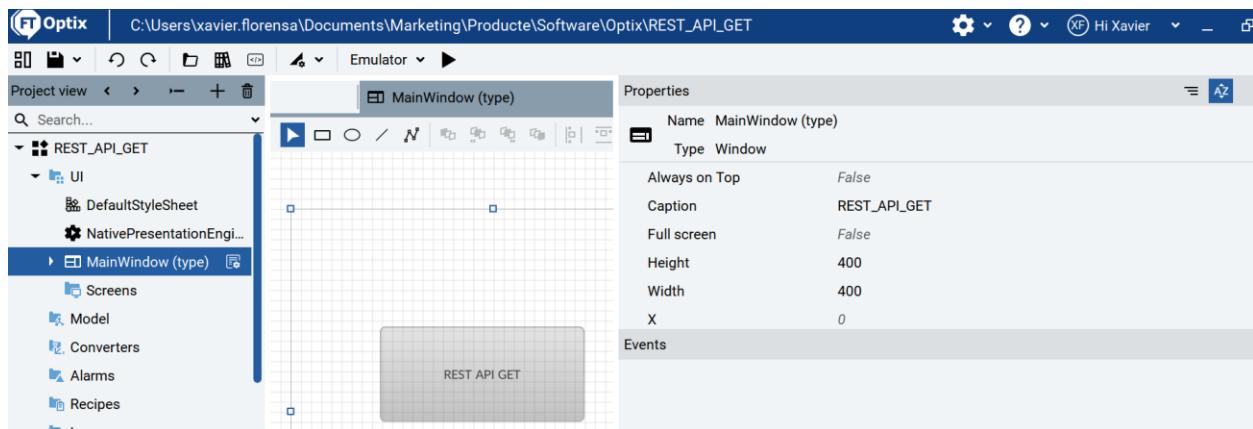
Go to libraries



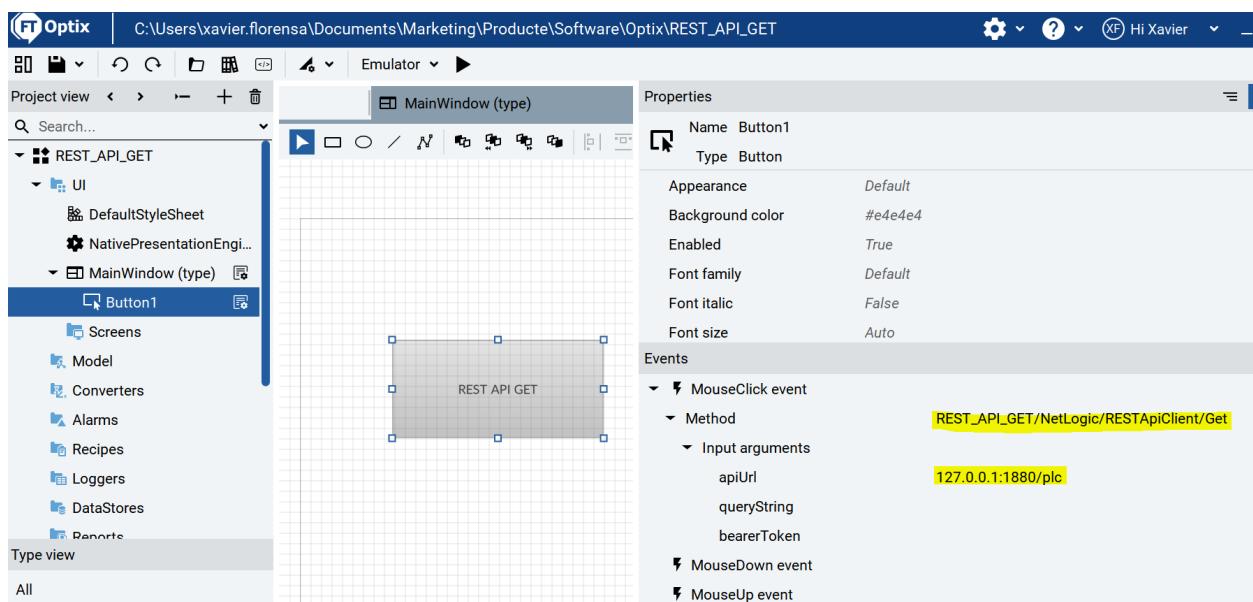
Drag and Drop REST API Client to NetLogic



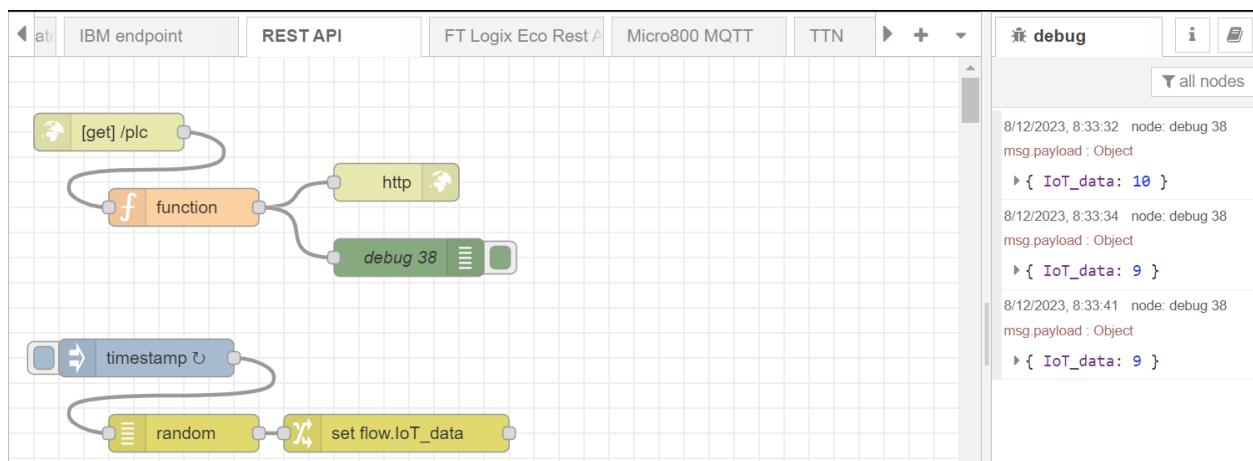
Now let's go to UI Main window and add a button

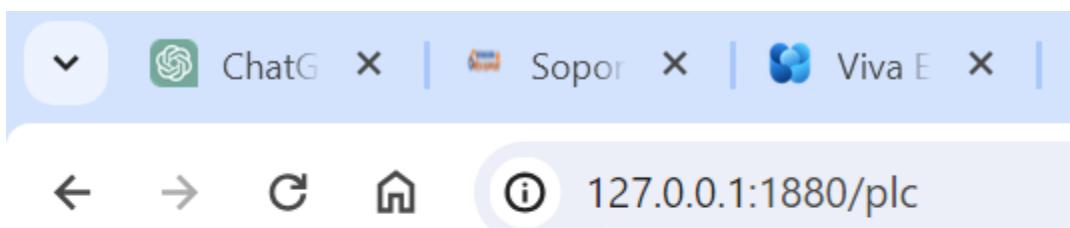
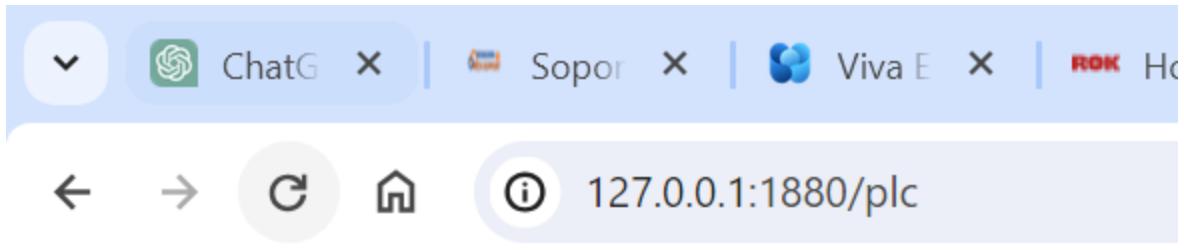


Click on the button to see the properties and add a Mouse Click event with the URL of the API



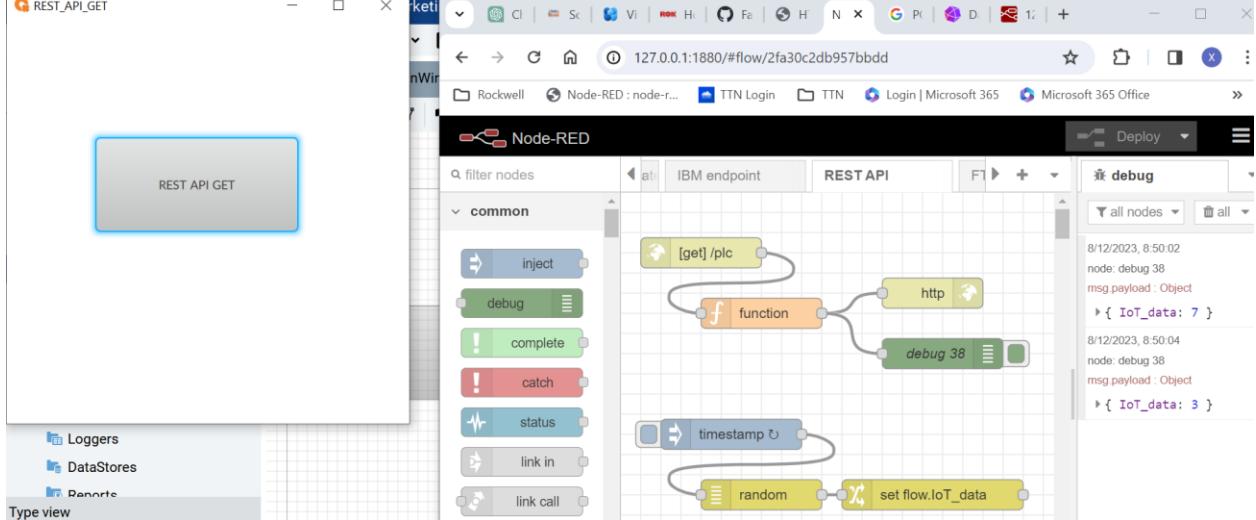
First of all we have to test these API URL, we are using node red here, to create an API that returns a random number between 1 and 10





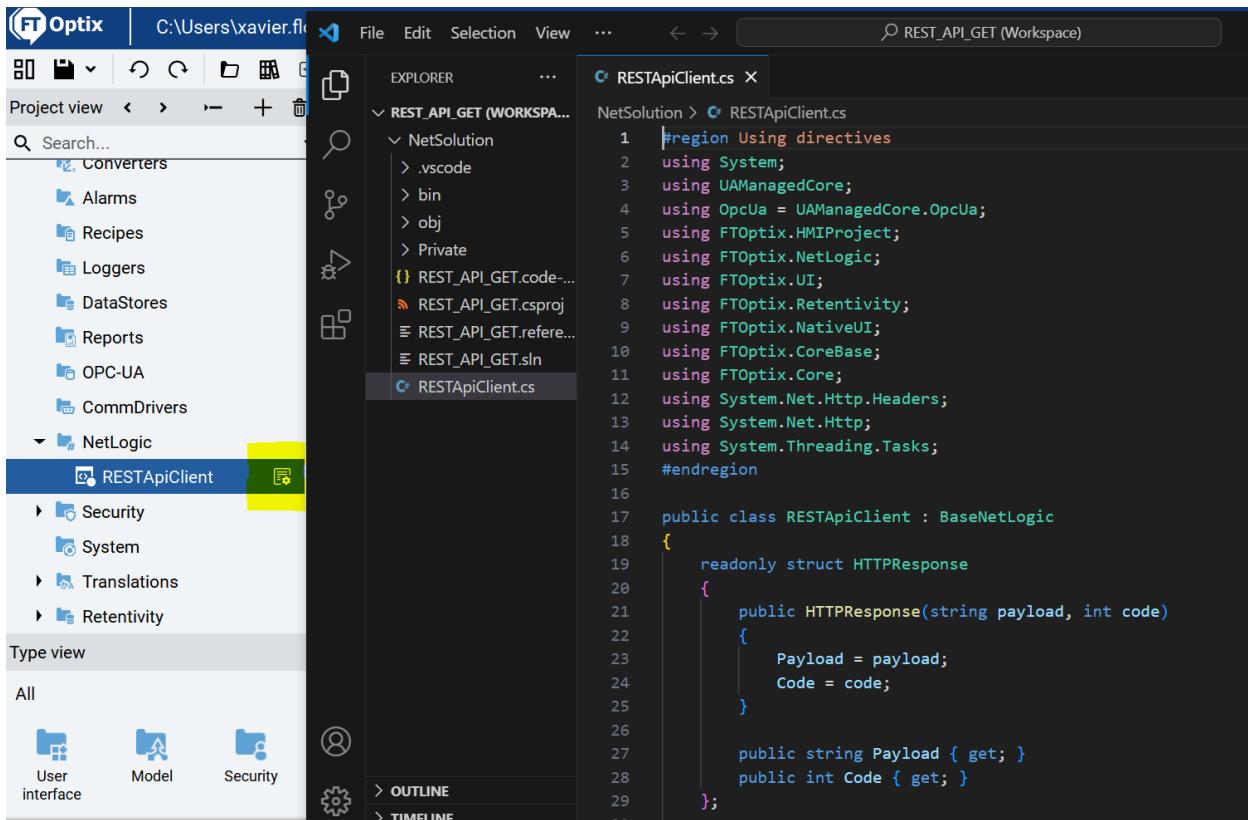
Now let's test the Optix runtime

This is working fine



But we need to get the data on Optix

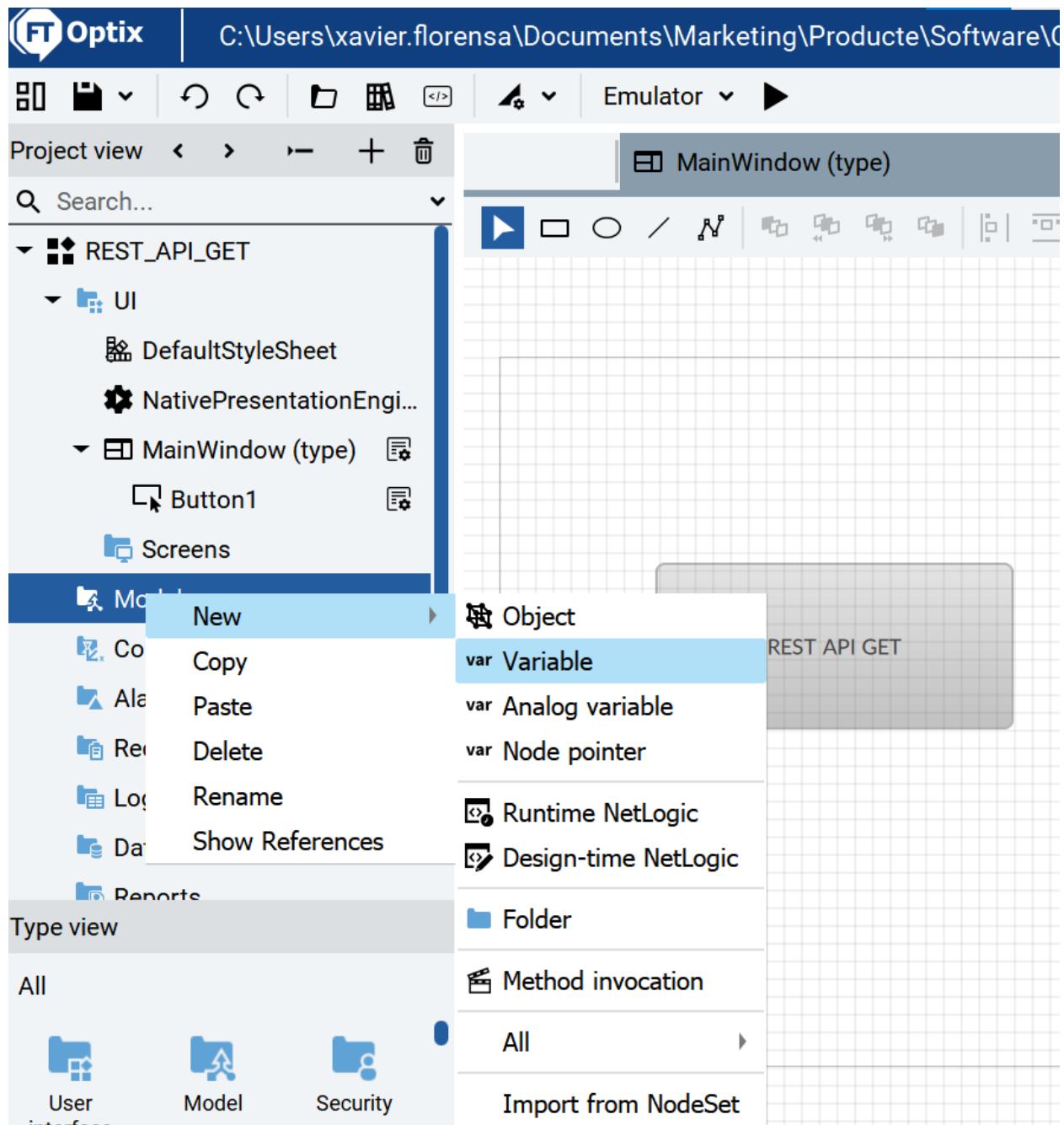
Let's open the C# script



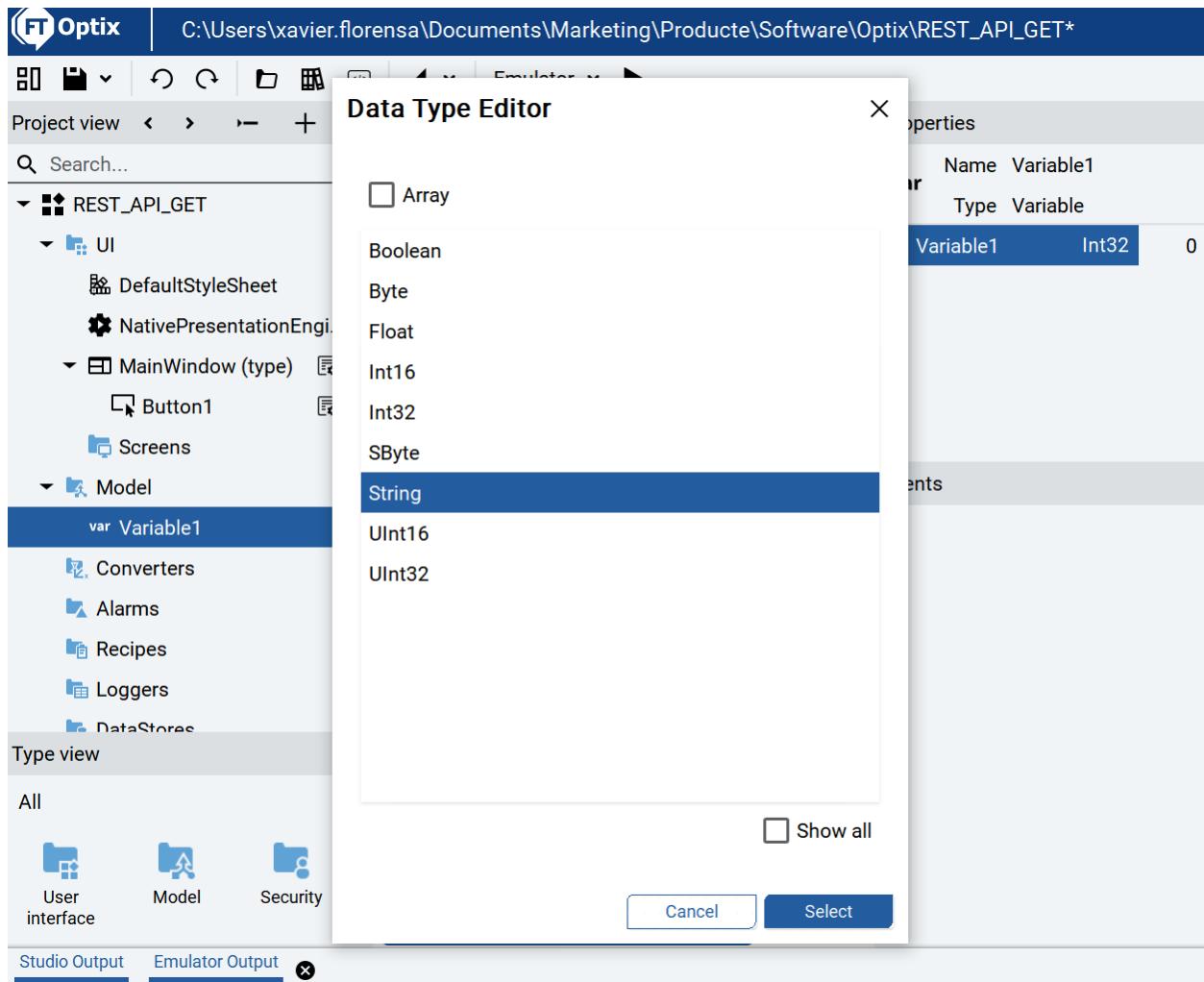
The screenshot shows the FTOptix IDE interface. The title bar says "FTOptix" and "C:\Users\xavier.fl...". The menu bar includes File, Edit, Selection, View, and a Help icon. The toolbar has icons for Open, Save, Undo, Redo, and others. The Explorer view on the left shows a project structure under "REST_API_GET (WORKSPACE)". The "NetSolution" folder contains ".vscode", "bin", "obj", "Private", "REST_API_GET.code...", "REST_API_GET.csproj", "REST_API_GET.referenc..., and "REST_API_GET.sln". A file named "RESTApiClient.cs" is selected and highlighted with a yellow background. The code editor on the right shows the following C# code:

```
1  #region Using directives
2  using System;
3  using UAManagerCore;
4  using OpcUa = UAManagerCore.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

```

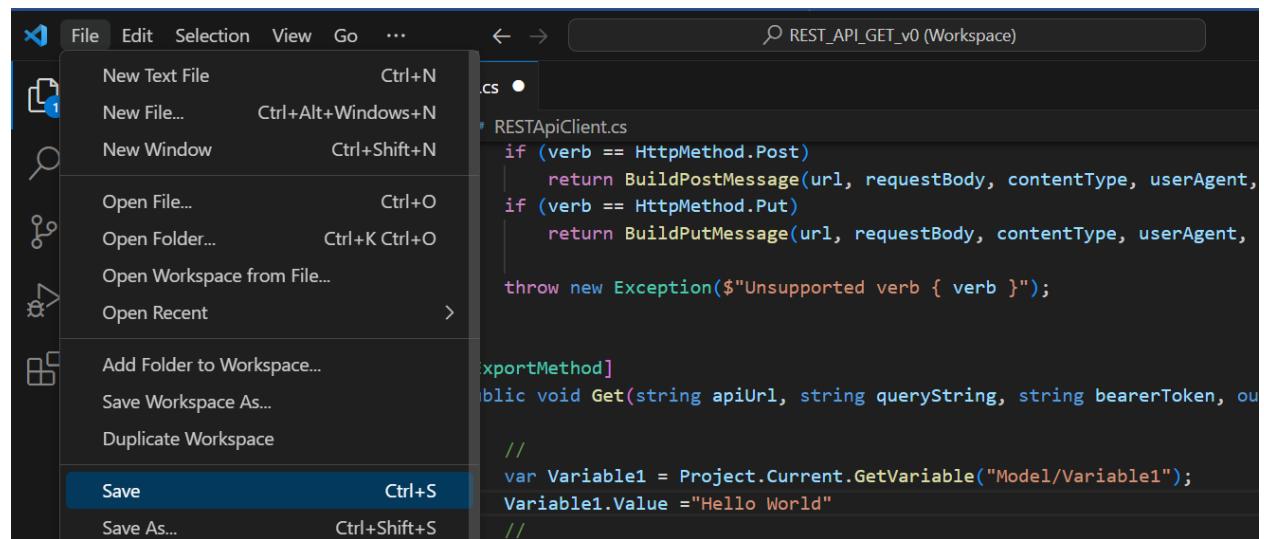
[ExportMethod]
public void Get(string apiUrl, string queryString, string bearerToken, out string response, out int
{
    //
    var Variable1 = Project.Current.GetVariable("Model/Variable1");
    Variable1.Value = "Hello World"
    //
    TimeSpan timeout = TimeSpan.FromMilliseconds(GetTimeout());
 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



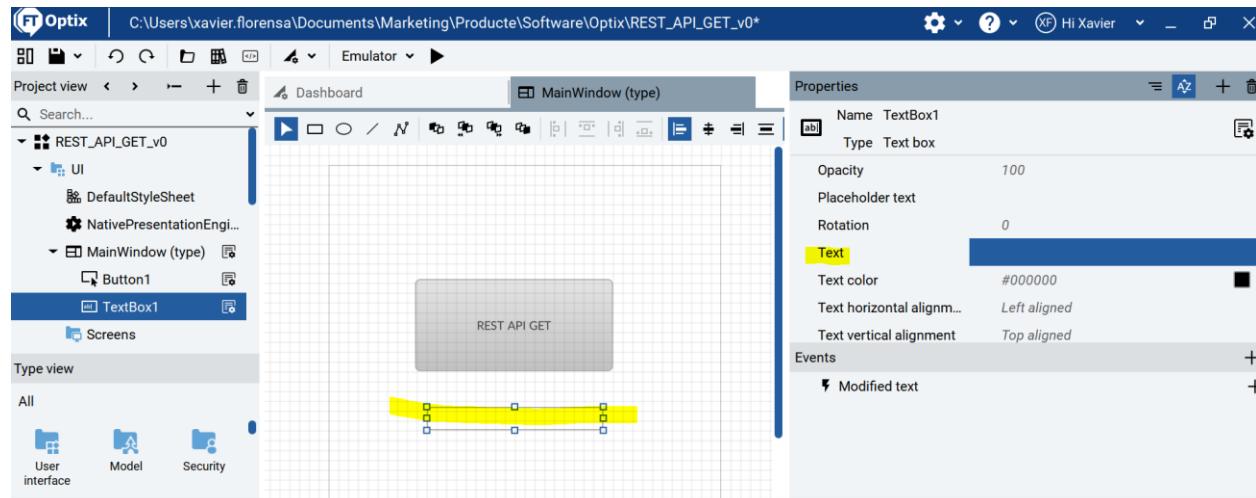
2023-12-08 09:17:07.807;NetHelper;222;Warning;16;User .NET solution failed to build:

C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\REST_API_GET_v0\ProjectFiles\NetSolution\RESTApiClient.cs(152,39): error CS1002: ; expected

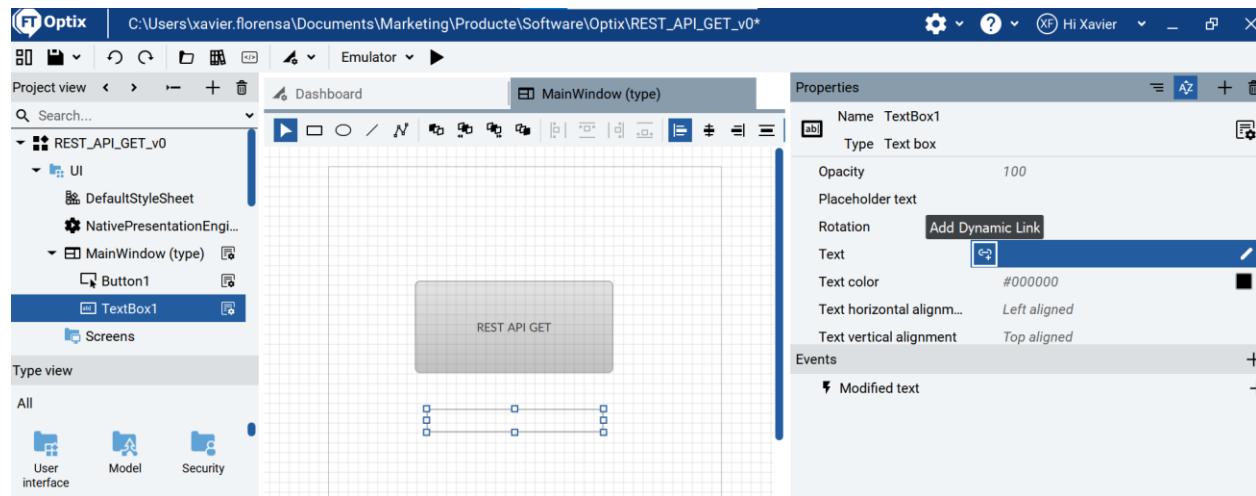
[C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\REST_API_GET_v0\ProjectFiles\NetSolution\REST_API_GET_v0.csproj]

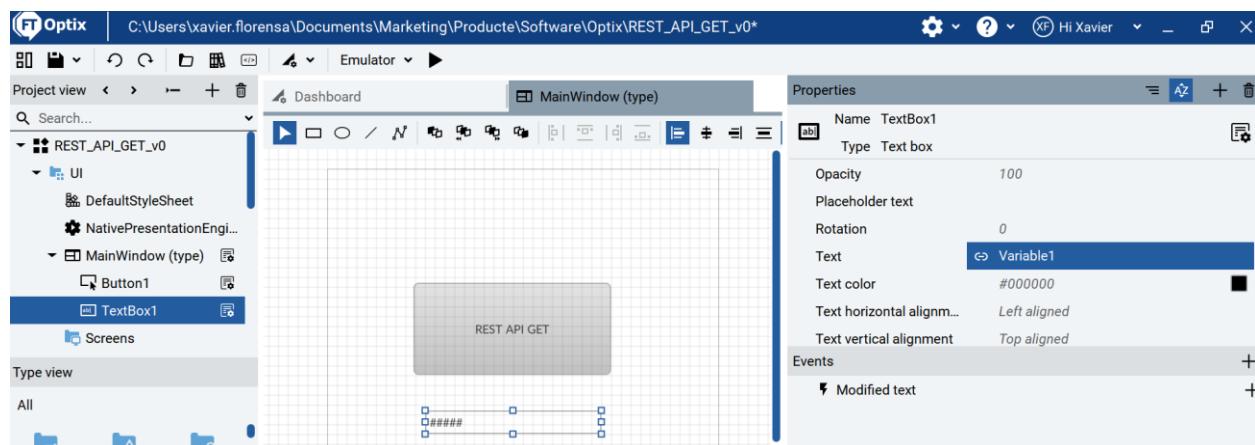
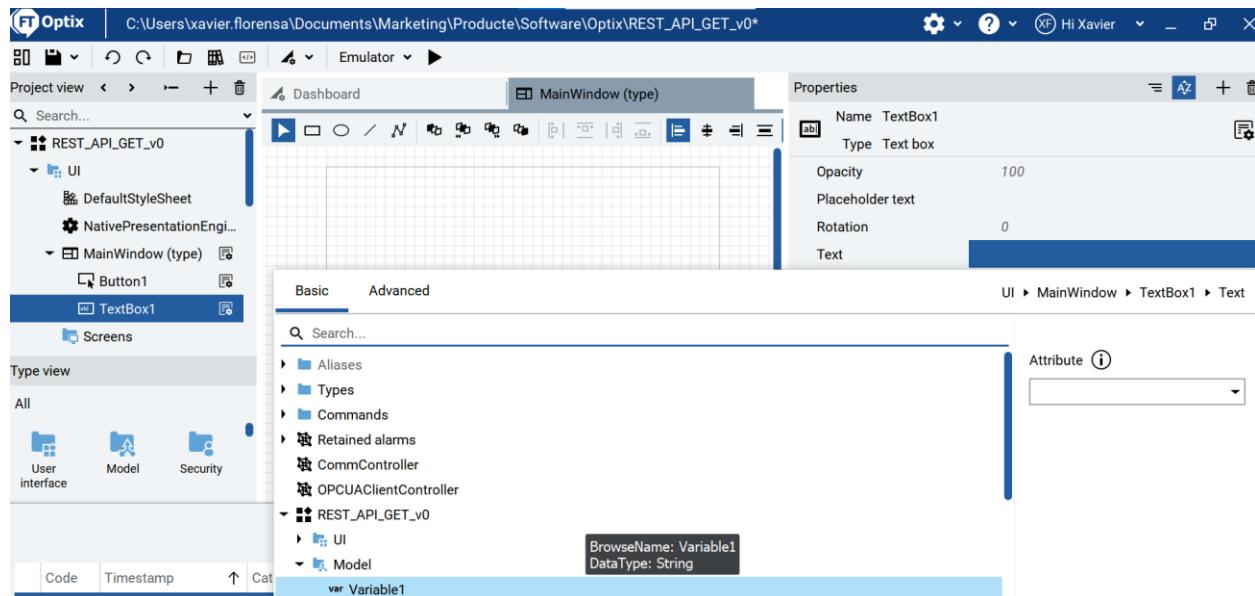
Let's correct this

Now introduce a text box

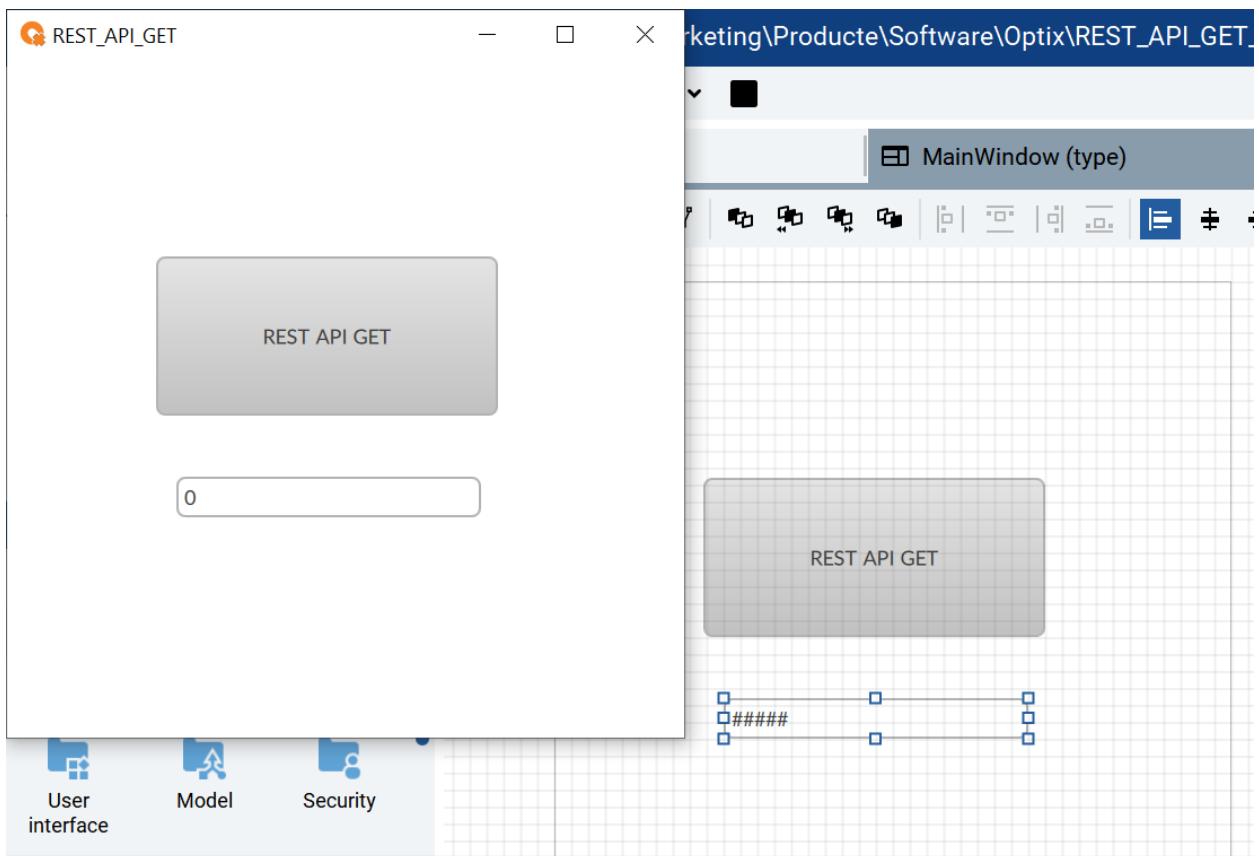


And link to Variable1

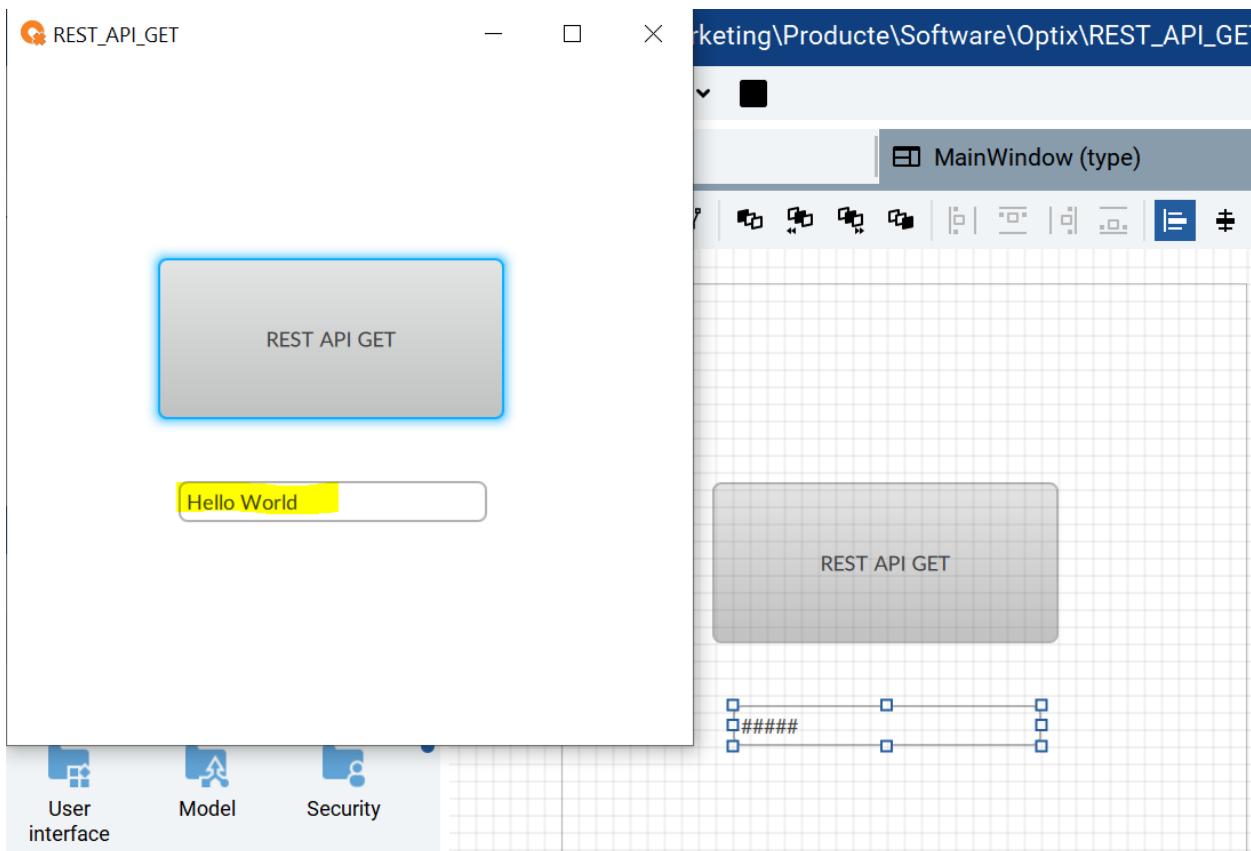




Let's test the application



Voilà



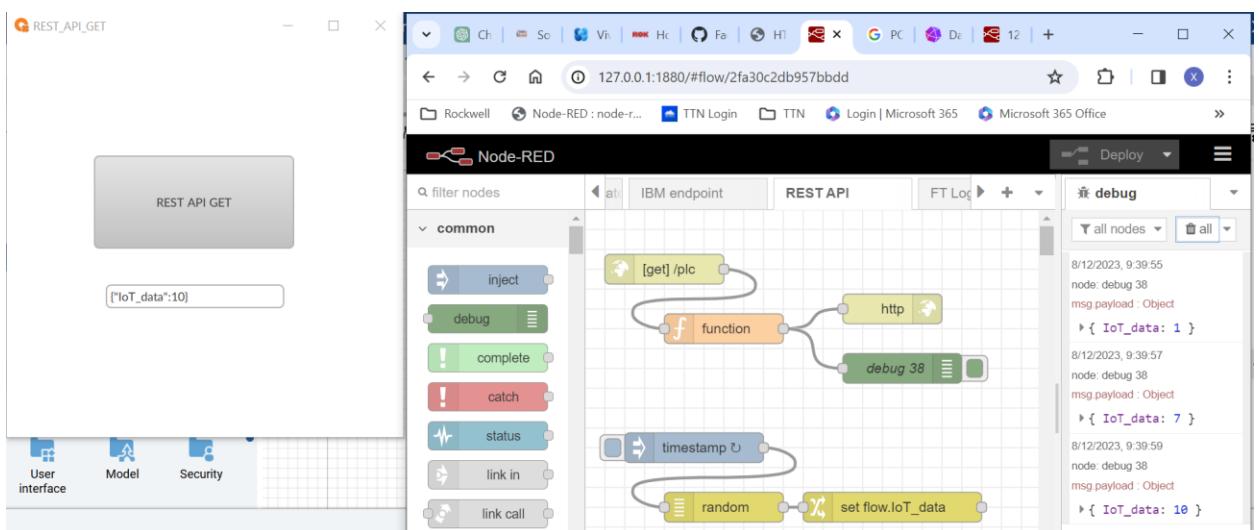
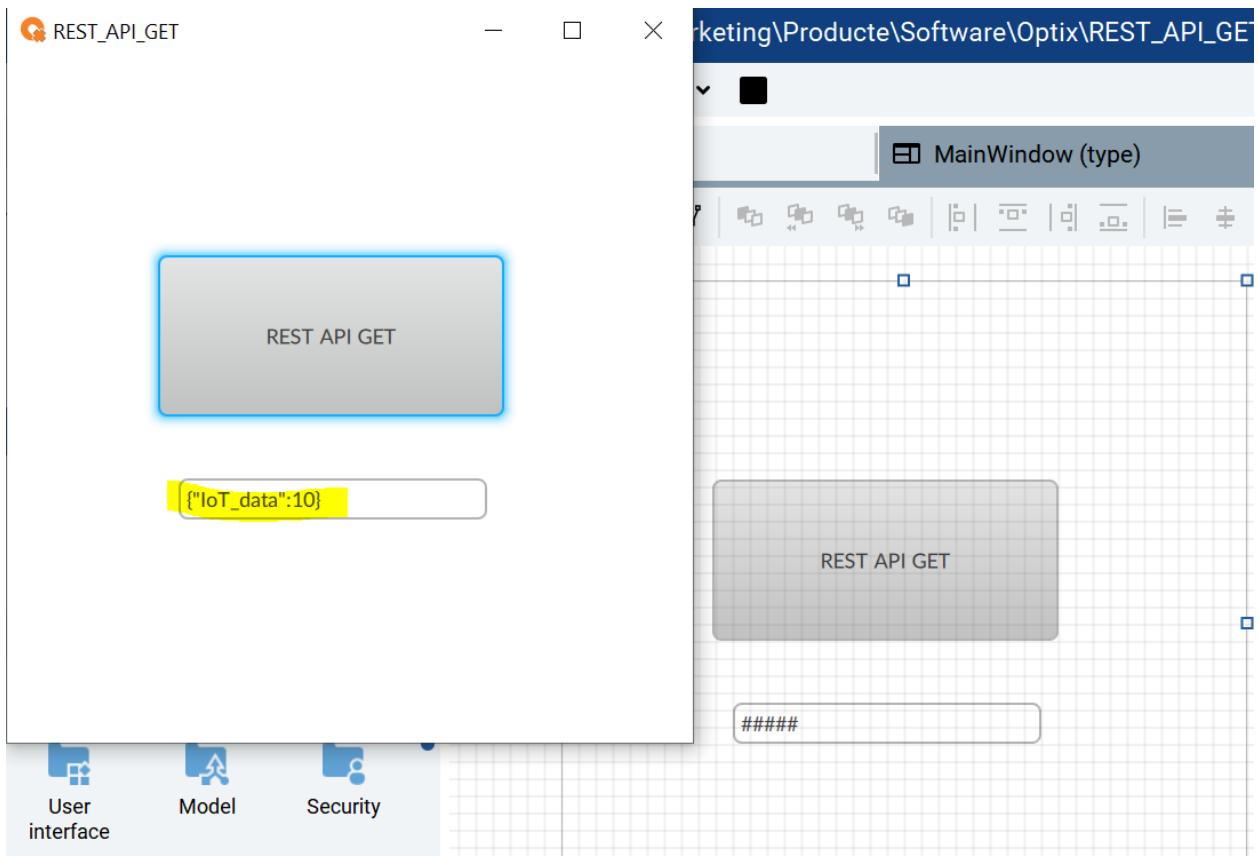
Not let's try to get the http response on that Textbox

```

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

13.2. 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'`

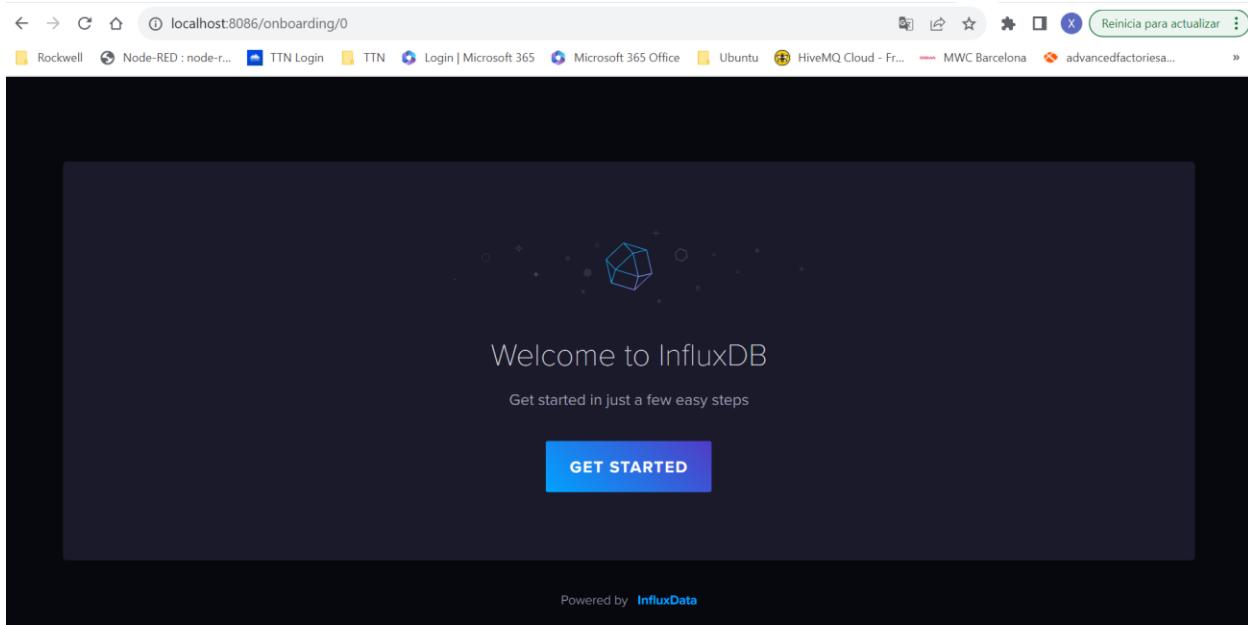
`./influxd`

Open an explorer with localhost and port 8086

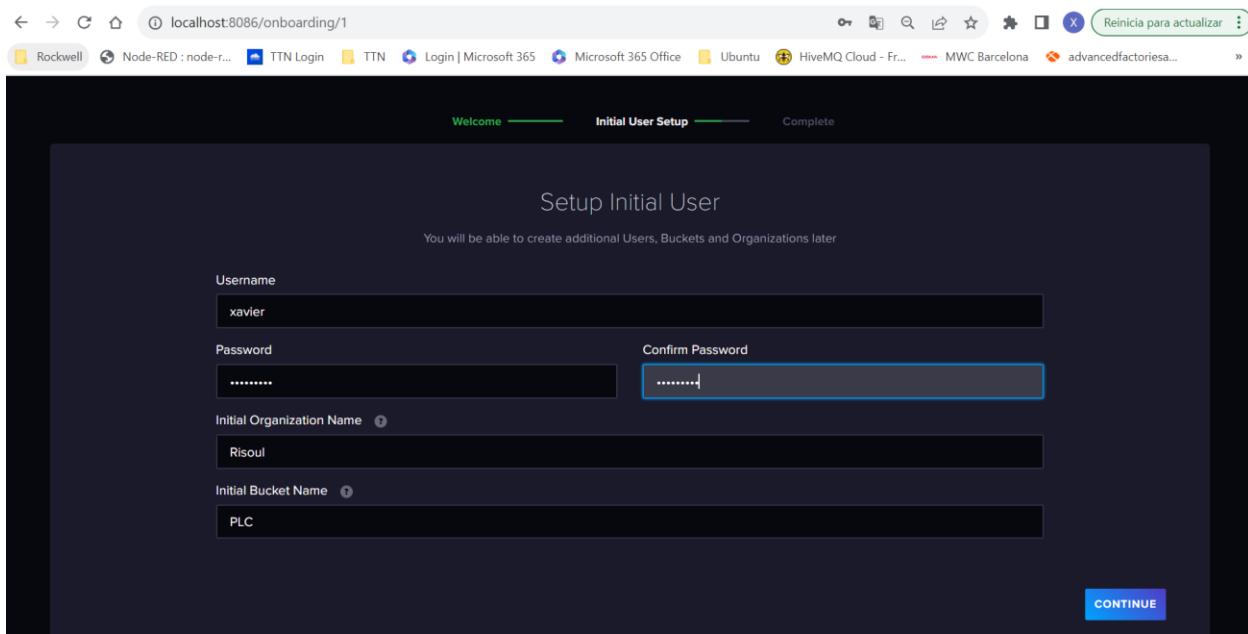
Username

xavier

Password



Start building your influxdb use cases



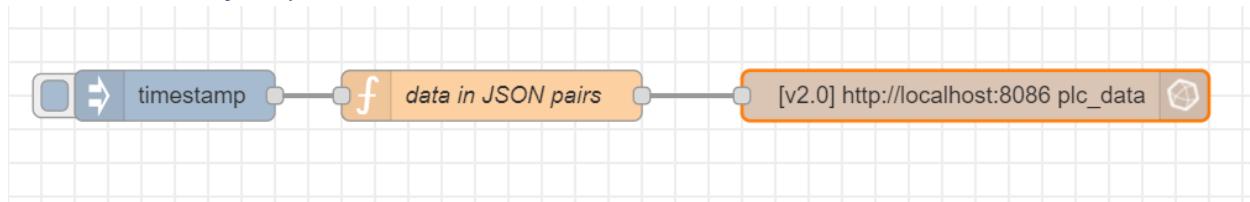
API token

MwQvFkNF8uI_Yx8327ohwgDG2qHBhO9ZbAqbpFPcFRX6amE9SooSyxAiA9zofuxj8c_C26cf-zGmMeLyGYKgHA==

Select Quickstart

The screenshot shows the InfluxDB 'Get Started' page. On the left, there's a sidebar with various icons and a search bar at the top. The main content area features four cards for different programming languages: Python (with a Python logo), Node.js (with a Node.js logo), Go (with a Go logo), and Arduino (with an Arduino logo). Below these cards is a section for the 'InfluxDB CLI'. A search bar and a 'Search Documentation' button are located in the top right corner. To the right of the search bar is a link to 'Reinicia para actualizar'. A 'USEFUL LINKS' sidebar on the right includes links to 'InfluxDB University', 'Get Started with Flux', 'Explore Metrics', 'Build a Dashboard', 'Write a Task', 'Report a bug', 'Community Forum', and 'Feature Requests'.

13.3. Inject your first data with node-red



The screenshot shows the 'Edit function node' dialog for the 'data in JSON pairs' function. At the top, there are 'Delete', 'Cancel', and 'Done' buttons. Below that is a 'Properties' tab with a gear icon. The 'Name' field is set to 'data in JSON pairs'. There are tabs for 'Setup', 'On Start', 'On Message', and 'On Stop', with 'On Message' being the active tab. The code editor contains the following JavaScript:

```

1 return {payload:{IoT_data:1024}}
2

```

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

Edit influxdb out node

[Delete](#)[Cancel](#)[Done](#)

Properties



🏷 Name

🌐 Servidor



Organización

⌚ Bucket

RSS Medición

⌚ Precisión



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-r... 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 Search buckets PLC _monitoring _tasks + Create Bucket

Filter _measurement plc

Filter _field Search _field tag values plc_data IoT_data

View Raw Data CSV Past 1h SCRIPT EDITOR SUBMIT

WINDOW PERIOD CUSTOM AUTO

auto (10s)

Fill missing values

AGGREGATE FUNCTION CUSTOM AUTO

13.4. 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_C26cf-
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.

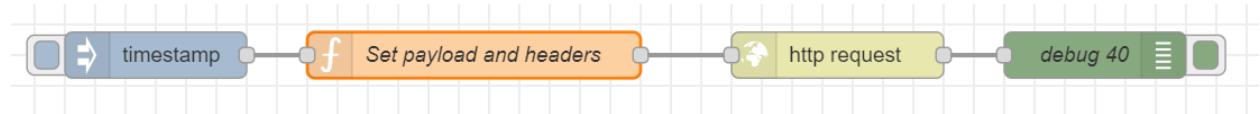
C:\Users\xavier.florensa>curl -POST "http://127.0.0.1:8086/api/v2/write?org=Risoul&bucket=PLC&precision=s" --header "Authorization: Token MwQvFkNF8uI_Yx8327ohwgDG2qHBhO9ZbAqbpFPcFRX6amE9SooSyxAiA9zofuxj8c_C26cf-zGmMeLyGYKgHA==" --data-raw "plc_data,host=host1 IoT_data=2324"

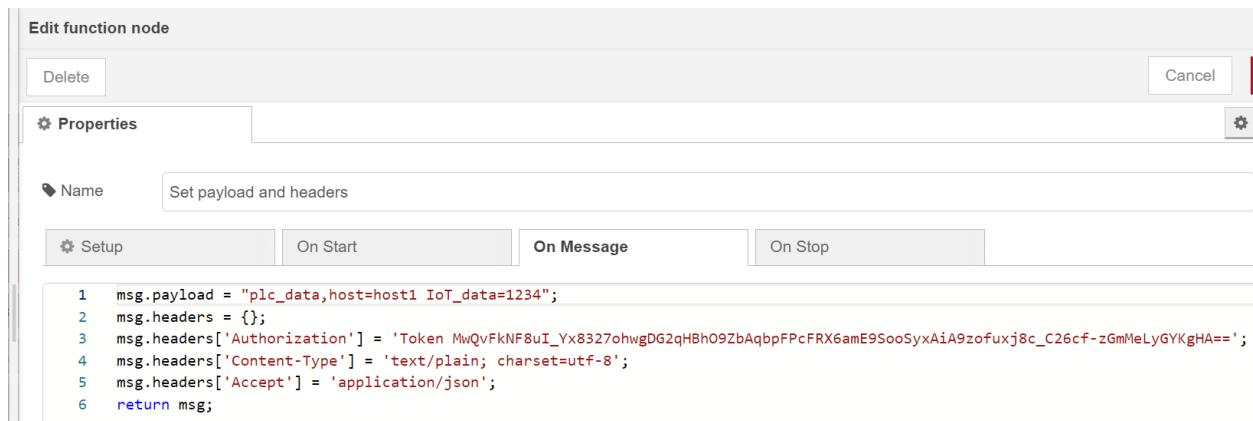
C:\Users\xavier.florensa>
```

The screenshot shows the Grafana Data Explorer interface. The main display area shows the value "2324,00". Below it, the query configuration is visible:

- Query 1 (0.01s)**
- FROM**: Search buckets, selected bucket: PLC
- Filter**: _measurement: plc, _field: IoT_data
- SCRIPT EDITOR** (button highlighted in blue)
- WINDOW PERIOD**: AUTO (selected)
- AGGREGATE FUNCTION**: AUTO (selected)

Let's do this from Node-RED

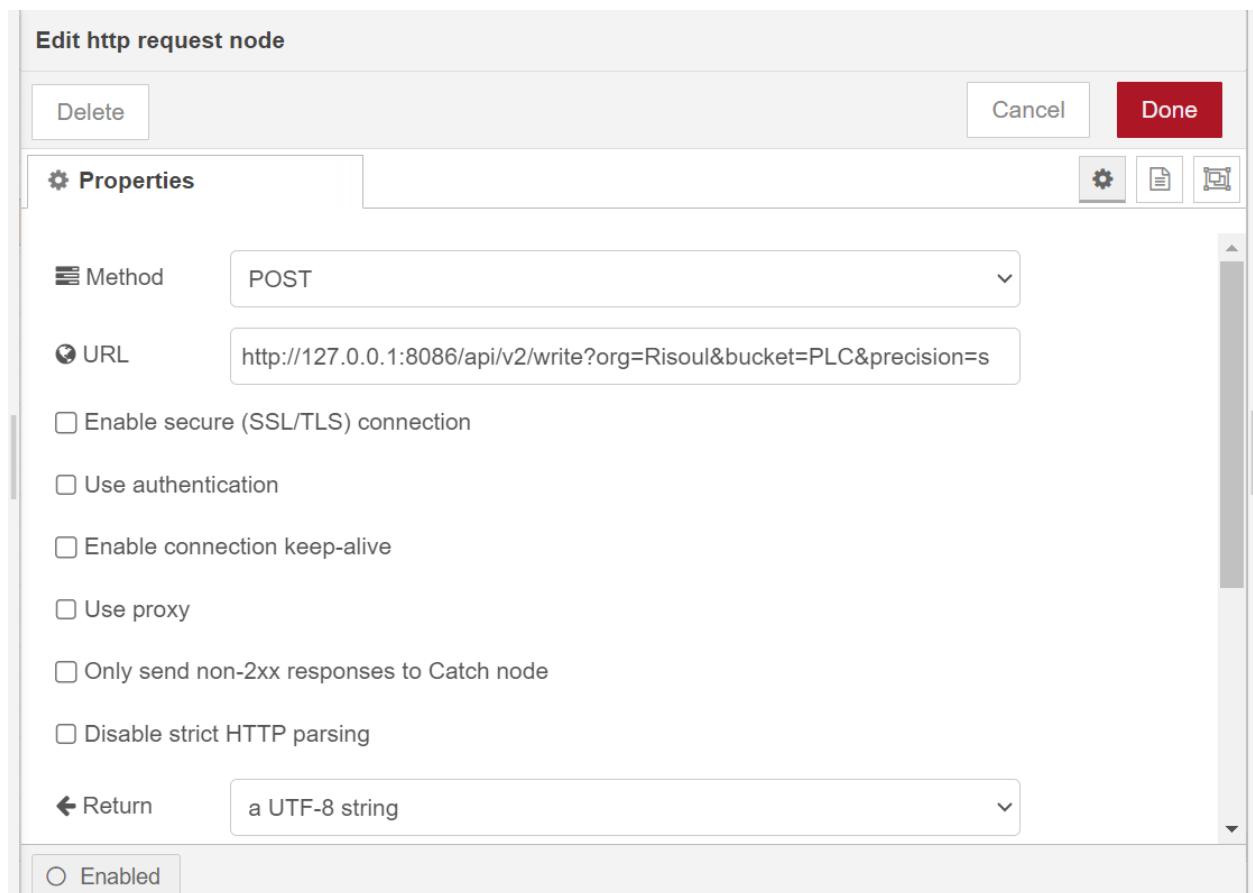




```

msg.payload = "plc_data,host=host1 IoT_data=1234";
msg.headers = {};
msg.headers['Authorization'] = 'Token
MwQvFkNF8uI_Yx8327ohwgDG2qHBh09ZbAqbpFPcFRX6amE9SooSyxAiA9zofuxj8c_C26cf-
zGmMeLyGYKgHA==';
msg.headers['Content-Type'] = 'text/plain; charset=utf-8';
msg.headers['Accept'] = 'application/json';
return msg;

```



Success

The screenshot displays two main interfaces: Node-RED and Data Explorer.

Node-RED: A flow editor window titled "Node-RED". It shows a sequence of nodes: "inject" → "timestamp" → "Set payload and headers" (with a configuration object) → "http request" → "debug 40". The "Set payload and headers" node has a "msg" variable with a value of "IoT_data:333". The "http request" node is configured to send a POST request to "http://127.0.0.1:8086/api/v2/write". The "debug 40" node shows the response: "msg.payload : string[0]" with value "OK".

Data Explorer: A visualization window titled "Data Explorer". It displays a large blue number "1234,00" representing the total value from the InfluxDB query. Below it, a query editor shows the following parameters:

- Query 1 (0.01s)
- FROM: PLC
- Filter: _measurement: IoT_data, _field: host, host: host1
- SCRIPT EDITOR: (empty)
- SUBMIT

The results pane indicates "No tag keys found in the current time range".

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

13.5. POST request to local InfluxDB

First let's try the right http request

API token example

```
c4K4HCsdkgH_9Vv1zBDLJ2ay8QR9ORKYTIPTclHo7PI8--  
BPCuhEBQkb0sI5QCbAEVozZgzuA9vWk2iHnYxijg==
```

```
C:\Users\xavier.florensa>curl -POST "http://127.0.0.1:8086/api/v2/write?org=Risoul&bucket=SEMINARI&precision=s" --header  
"Authorization: Token 3flg3FPeLRzm3nALax4VZdBLW4wGqUUbFhrHsSzBG9wSiiDyMVuuwj_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  
3flg3FPeLRzm3nALax4VZdBLW4wGqUUbFhrHsSzBG9wSiiDyMVuuwj_9hKwb0v2xEi_r7K_VhndhDEdwQTi  
R9g==" --data-raw "optix,host=host1 IoT_data=333"
```

With this result

The screenshot shows the Data Explorer interface from a web browser. At the top, there are several tabs: Node-RED, 127.0.0.1:1880/plc, Data Explorer | Risoul | InfluxDB, API keys - OpenAI API, and IUAVariable.VariableChange. Below the tabs, the URL is 127.0.0.1:8086/orgs/45bd74e06c16b567/data-explorer?fluxScriptEditor. The main area is titled "Data Explorer" with a "Single Stat" button and a "CUSTOMIZE" button. A large blue number "333,00" is displayed prominently. Below it, there's a "Query 1 (0.03s)" section with four filter panels: "FROM" (Search buckets, PLC, SEMINARI selected), "Filter _measurement" (optix selected), "Filter _field" (IoT_data selected), and "Filter host" (host1 selected). To the right, there are buttons for "View Raw Data", "CSV", "Past fm", "SCRIPT EDITOR" (selected), and "SUBMIT". On the far right, there are options for "WINDOW PERIOD" (CUSTOM, AUTO selected), "auto (1s)", "Fill missing values", and "AGGREGATE FUNCTION" (CUSTOM, AUTO selected).

Build same project as before but with POST method

The screenshot shows the Optix software interface. The title bar says "Optix" and "C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\Optix_2_InfluxDB_v5". The main area shows a "Dashboard" with a "MainWindow" component. The "Properties" panel on the right shows the following configuration for "Button1": Name: Button1, Type: Button, Appearance: Default, Background color: #e4e4e4, Enabled: True, Font family: Default, Font italic: False, Font size: 20. Under "Events", there is a "MouseClick event" with a "Method" set to "Optix_2_InfluxDB_v5/NetLogic/RESTApiClient/Post". The "Input arguments" for this method are: apiUrl: http://127.0.0.1:8086/api/v2/write?org=Risoul&bucket=SEMINARI&precision=s, requestBody: optix,host=host1 IoT_data=444...BLW4wGqUUbFhrHsSzBG9wSiiDyMVuuwj_9hKwb0v2xEi_r7K_VhndhDEdwQTiRg==, bearerToken: ..., contentType: text/plain.

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

3flg3FPeLRzm3nALax4VZdBLW4wGqUUbFhrHsSzBG9wSiiDyMVuuwj_9hKwb0v2xEi_r7K_VhndhDEdwQTiR9g==

Content type

text/plain



Optix_2_InfluxDB

InfluxDB injection

httpResponse.Payload**httpResponse.code**

The screenshot shows the Data Explorer interface from a web browser. The main display area features a large, prominent value of "444,00" in blue. Above this, there are two buttons: "Single Stat" and "CUSTOMIZE". To the right, there are buttons for "Local" and "SAVE AS".

Below the main display, the interface is divided into several sections:

- Query 1 (0.02s)**: Shows the current query duration.
- FROM**: A dropdown menu showing "Search buckets". Below it, under the "PLC" section, "SENTINARI" is selected, with other options like "_monitoring", "_tasks", and "+ Create Bucket".
- Filter**: Three filter panels. The first panel has dropdowns for "_measurement" (set to "optix") and "_field" (set to "IoT_data"). The second panel has dropdowns for "_measurement" (set to "host") and "_field" (set to "host1"). Both panels have search bars below them. The third panel is partially visible.
- View Raw Data**: A button to view raw data.
- CSV**: A button to export data as CSV.
- Past 1m**: A button to view data from the last minute.
- SCRIPT EDITOR**: A tab for writing and executing scripts.
- SUBMIT**: A button to submit the current query.

On the right side, there are additional settings:

- WINDOW PERIOD**: Set to "AUTO".
- CUSTOM** and **AUTO** buttons for window period selection.
- auto (1s)**: A setting for auto-resolution.
- Fill missing values**: A checkbox for handling missing data.
- AGGREGATE FUNCTION**: Set to "CUSTOM".
- CUSTOM** and **AUTO** buttons for aggregate function selection.

In the bottom right corner, a message states: "No tag keys found in the current time range".

As you can see here

<https://youtu.be/nqSctOoieOk>

13.6. 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);
```

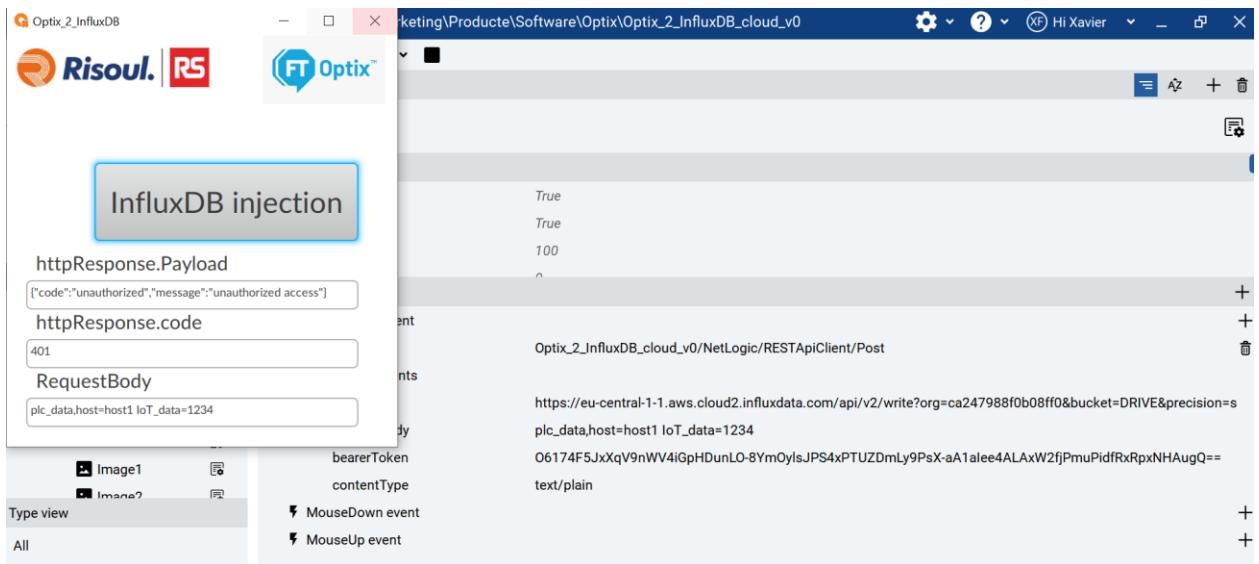
So if you do so, you will have success

The image shows two screenshots demonstrating the integration of Risoul Optix 2 with InfluxDB.

Top Screenshot: This is a screenshot of the Risoul Optix 2 software interface. It displays an "InfluxDB injection" configuration window. The "httpResponse.Payload" field contains the JSON payload: {"plc_data": "host=host1 IoT_data=1234"}. The "httpResponse.code" field is set to 204. The "RequestBody" field also contains the same JSON payload. Below the configuration, a table lists various event types and their corresponding values. The table includes rows for "bearerToken" (True), "contentType" (text/plain), and "body" (Optix_2_InfluxDB_cloud_v1/NetLogic/RESTApiClient/Post). The URL in the browser bar is eu-central-1-1.aws.cloud2.influxdata.com/orgs/ca247988f0b08ff0/data-explorer.

Bottom Screenshot: This is a screenshot of the InfluxDB Data Explorer. It shows a single stat query result with the value "1234,00". The query parameters are: FROM DRIVE, MEASUREMENT mem, Filter _field IoT_data, Filter host host1. The results table shows a single row with host: host1 and IoT_data: 1234. The URL in the browser bar is eu-central-1-1.aws.cloud2.influxdata.com/orgs/ca247988f0b08ff0/data-explorer.

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



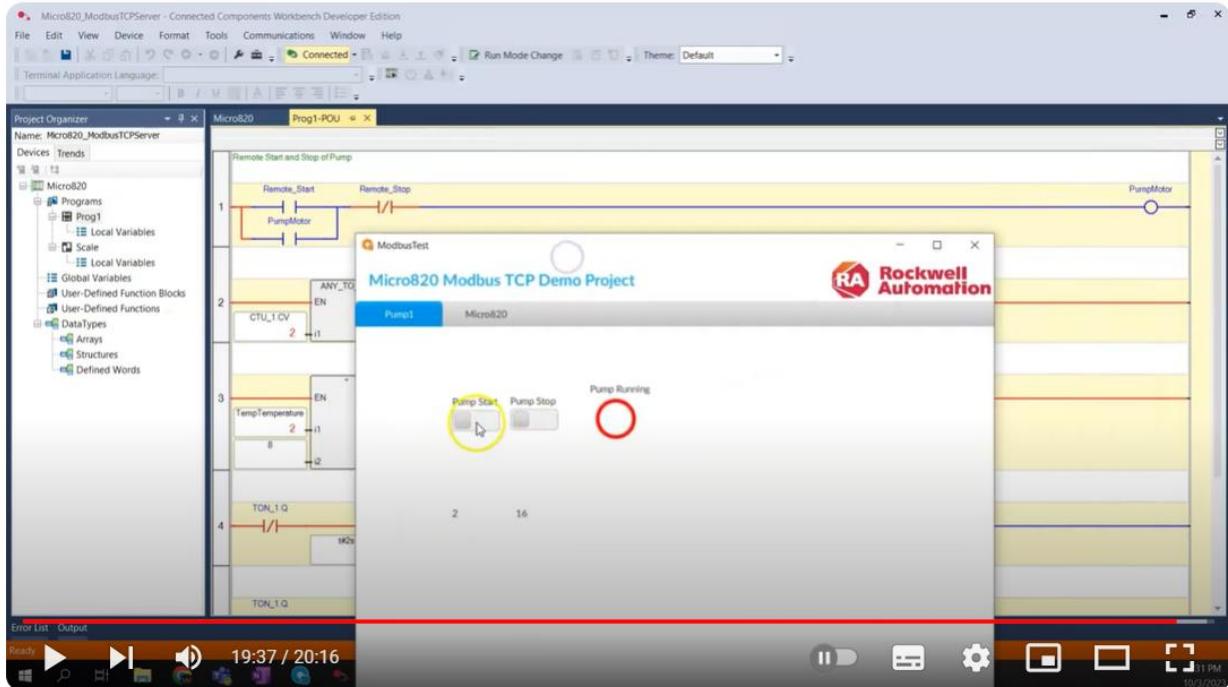
14. FT Optix communicating with Micro850

You can follow this video from Wayne Welk

<https://www.youtube.com/watch?v=k4Q0C0uCeG0&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

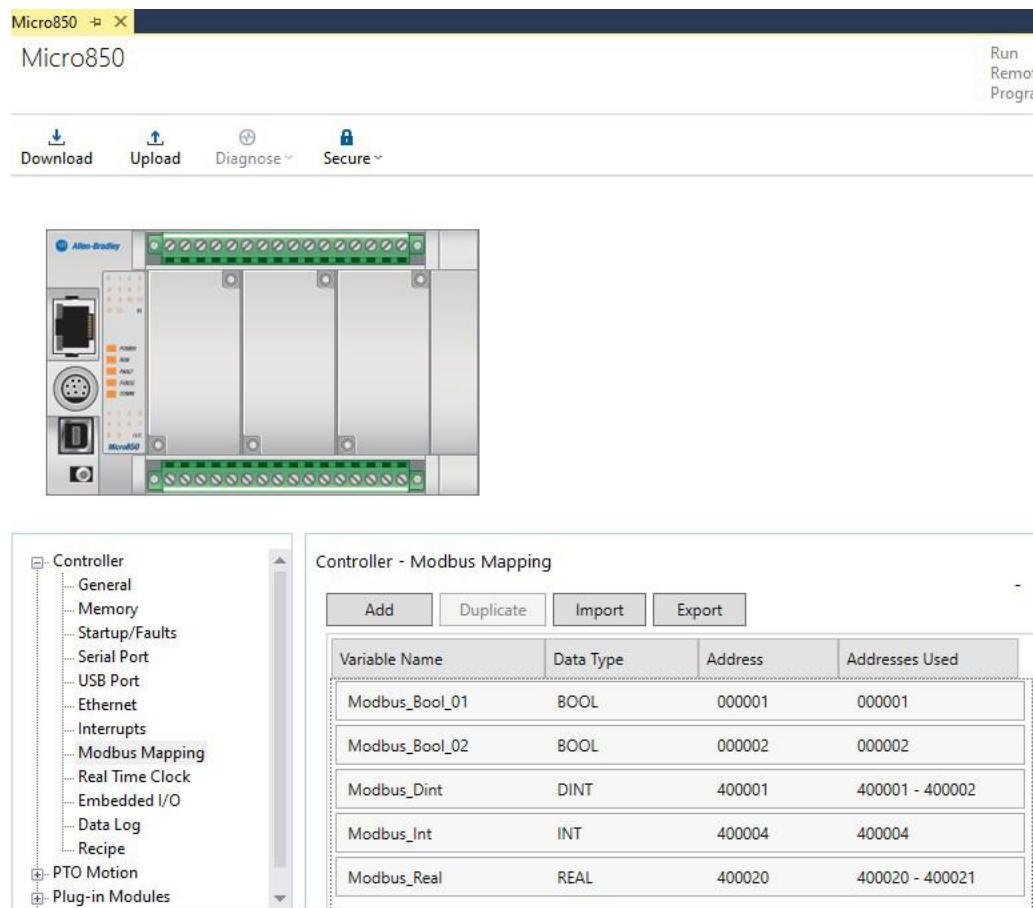
FactoryTalk Optix: Modbus TCP communication with Micro850

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)

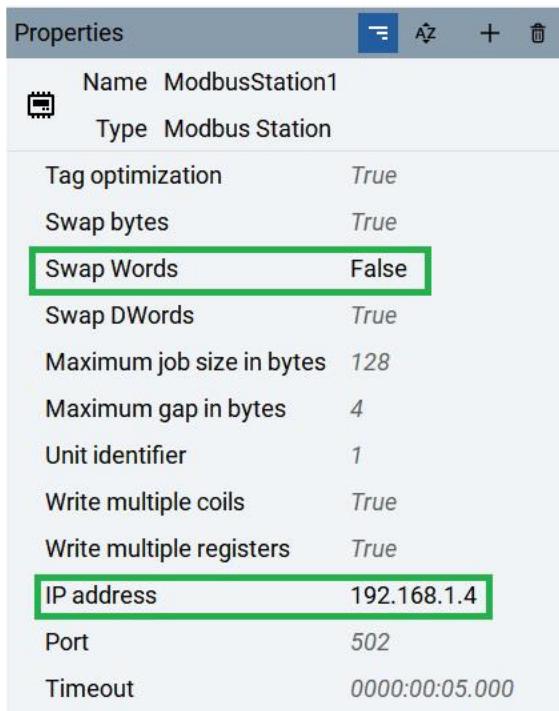
4xxxxx (holding registers)

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 controllers 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

Properties	
tag	Name ModbusTag1
	Type Modbus tag
ModbusTag1	Boolean False
Memory area	Coil
Maximum length	80
Coil number	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

Bool_Coil_0	1
Bool_Coil_1	1
Dint_Register_0	2
Int_Register_3	3
Real_Register_19	4.4
Dint_Register_3	0

15. 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

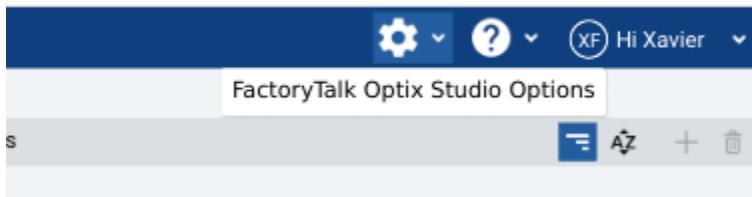
 Search Knowledgebase...

Document ID QA65456

Published Date 01/27/2023

https://rockwellautomation.custhelp.com/app/answers/answer_view/a_id/1138150/loc/en_US#_highlight

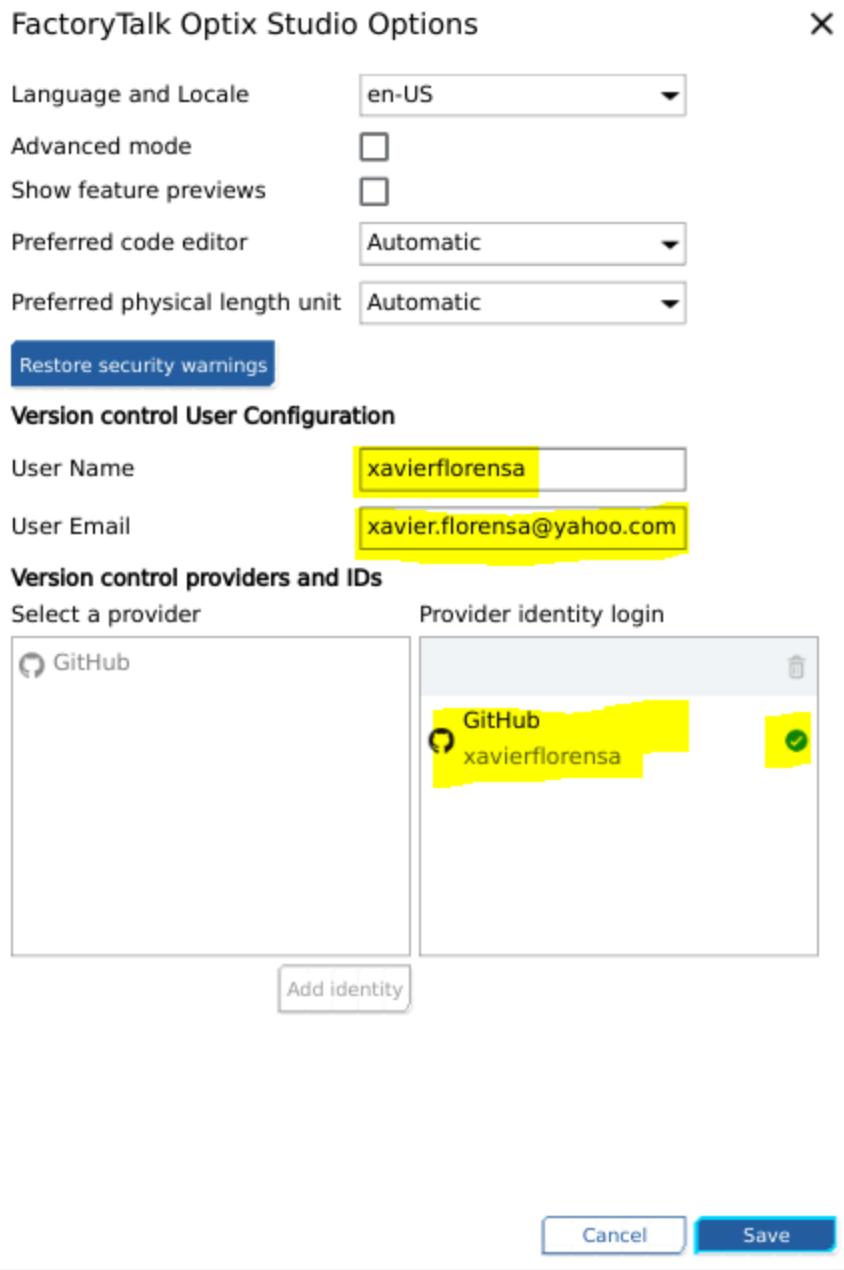
First of all modify your Optix Pro configuration on the toothwheel



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.

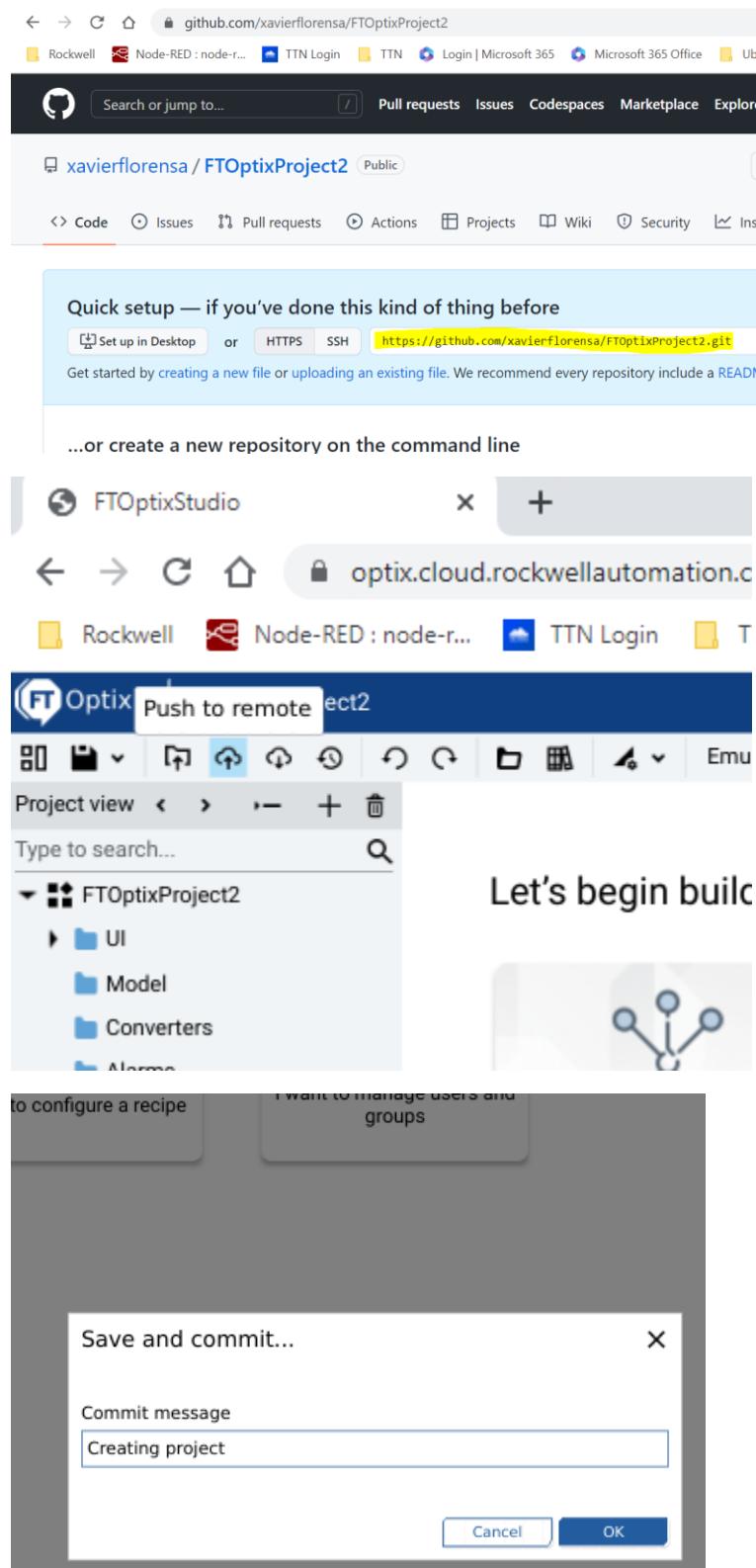
Go to Github

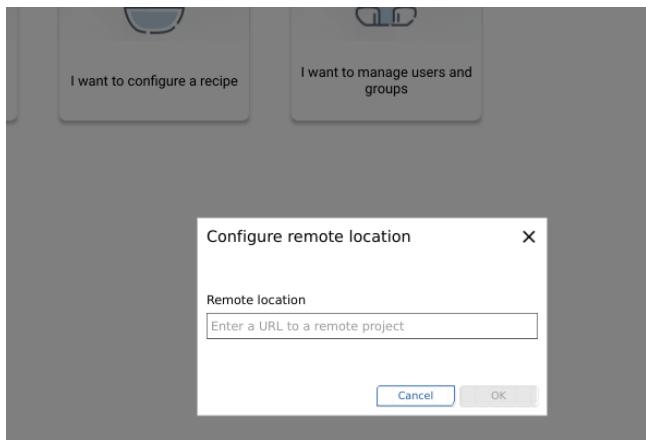
Create a new project on Github with the same name you use on Optix on Github

That's all.

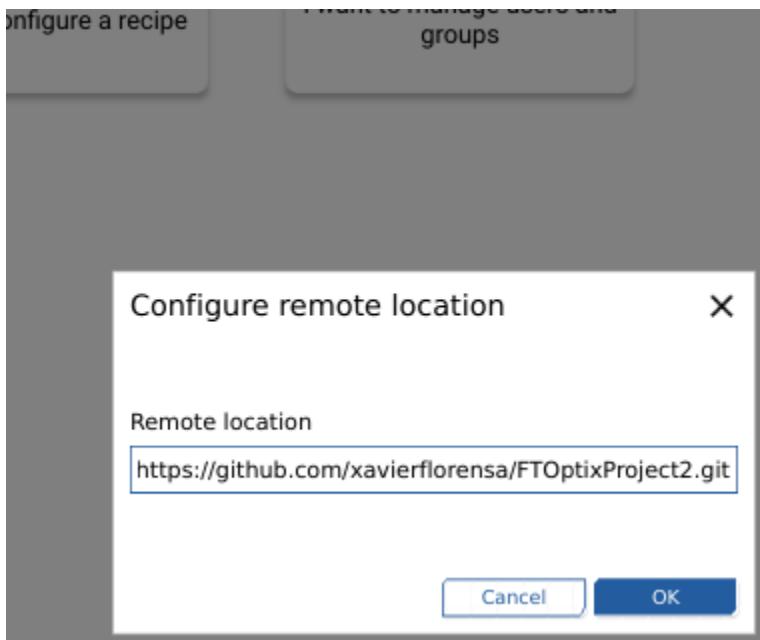
When you save and commit you will be prompted to give a comment and a destination url.

The destination url is this one

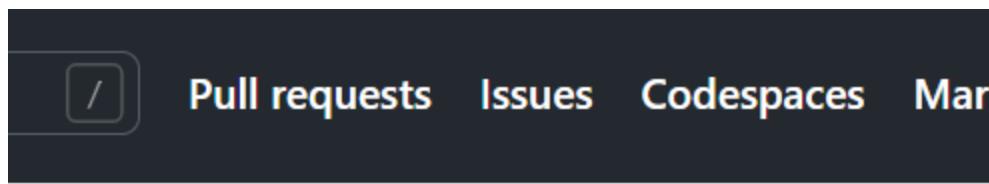




<https://github.com/xavierflorensa/FTOptixProject2.git>



Let's go to github



Find a repository...

FTOptixProject2

Updated now

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

xavierflorensa / FTOptixProject2 Public

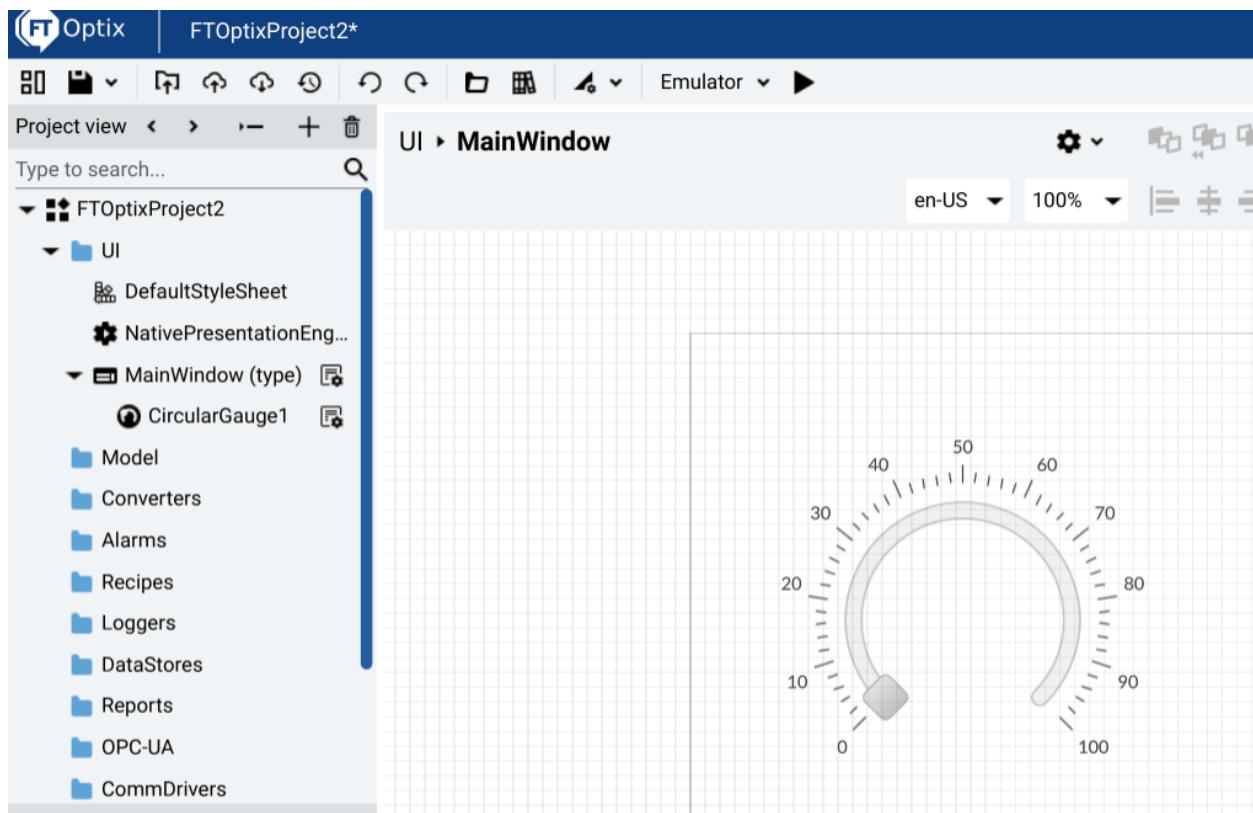
Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

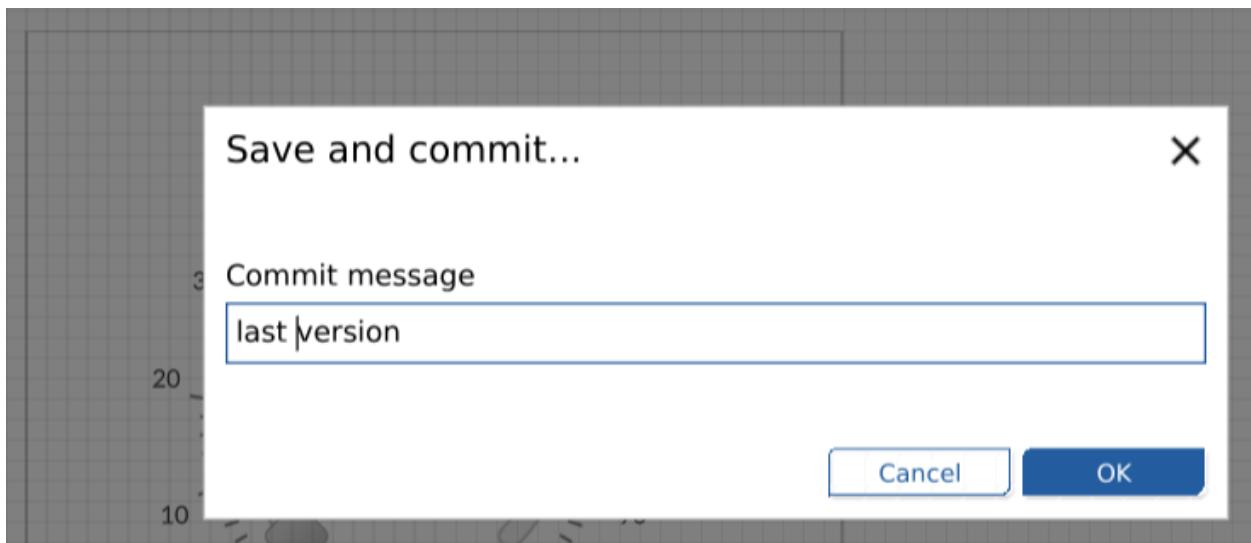
xavierflorensa	Creating project	655fb22 2 minutes ago	1 commit
Nodes	Creating project	2 minutes ago	
.gitattributes	Creating project	2 minutes ago	
.gitignore	Creating project	2 minutes ago	
FTOptixProject2.optix	Creating project	2 minutes ago	
FTOptixProject2.optix.design	Creating project	2 minutes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

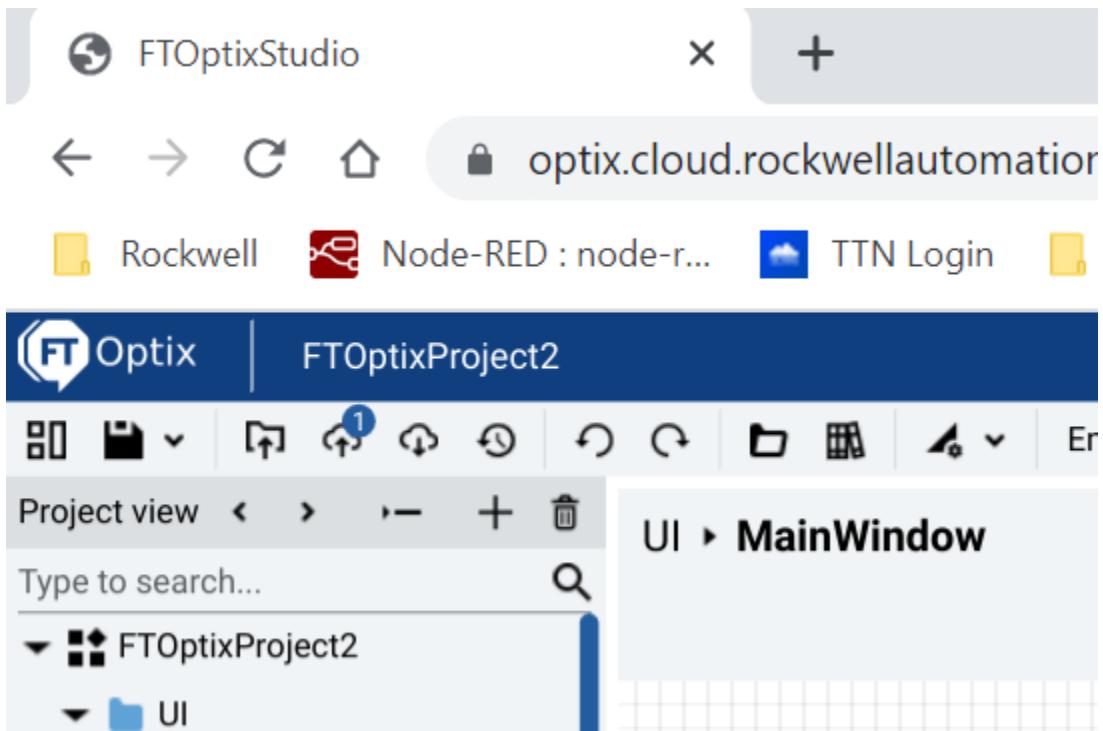
Now let's go to Optix and make some changes



Now save and commit



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



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 Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The Code tab is selected. Below the tabs, it shows `main`, 1 branch, 0 tags, Go to file, Add file, and a dropdown for Code. A list of recent commits by `xavierflorensa` is displayed:

File	Description	Time
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 Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and Insights. The Code tab is selected. Below the tabs, it shows `main`. A section titled "Commits on Mar 25, 2023" lists two commits:

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

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

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 'Code' tab is selected. A dropdown menu 'main' is open, showing two commits:

- last version** by xavierflorensa committed 7 minutes ago (commit `ff563b7`)
- Creating project** by xavierflorensa committed 14 minutes ago (commit `655fb22`)

Both commits have a 'View commit details' button. Navigation buttons 'Newer' and 'Older' are at the bottom.

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 by `xavierflorensa` 8 minutes ago. The commit message is empty.

Showing 1 changed file with 37 additions and 0 deletions.

`diff --git a/Nodes/UI/UI.yaml b/Nodes/UI/UI.yaml`

Line	Text
82	E2 E2 <ins>## -82,3 +82,48 ## Children:</ins>
83	E3 E3 <ins> - Name: Size</ins>
84	E4 E4 <ins> - ModellingRule: Optional</ins>
85	<ins> + - Name: CircularGauge</ins>
86	<ins> + - Type: CircularGauge</ins>
87	<ins> + - Children:</ins>
88	<ins> + - - Name: Value</ins>
89	<ins> + - - Type: BaseDataVariableType</ins>
90	<ins> + - - DataType: Float</ins>
91	<ins> + - - Value: 0.0</ins>
92	<ins> + - - - Name: MinValue</ins>
93	<ins> + - - - Type: BaseDataVariableType</ins>
94	<ins> + - - - DataType: Float</ins>
95	<ins> + - - - Value: 0.0</ins>
96	<ins> + - - - - Name: MaxValue</ins>
97	<ins> + - - - - Type: BaseDataVariableType</ins>
98	<ins> + - - - - DataType: Float</ins>
99	<ins> + - - - - Value: 100.0</ins>
100	<ins> + - - - - - Name: WarningZones</ins>
101	<ins> + - - - - - Type: BaseObjectType</ins>
102	<ins> + - - - - - - Name: Width</ins>
103	<ins> + - - - - - - Type: BaseVariableType</ins>
104	<ins> + - - - - - - DataType: Size</ins>
105	<ins> + - - - - - - ModellingRule: Optional</ins>
106	<ins> + - - - - - - Value: 210.0</ins>
107	<ins> + - - - - - - - Name: Height</ins>
108	<ins> + - - - - - - - Type: BaseVariableType</ins>
109	<ins> + - - - - - - - DataType: Size</ins>
110	<ins> + - - - - - - - ModellingRule: Optional</ins>
111	<ins> + - - - - - - - Value: 210.0</ins>
112	<ins> + - - - - - - - - Name: TopMargin</ins>
113	<ins> + - - - - - - - - Type: BaseVariableType</ins>
114	<ins> + - - - - - - - - DataType: Size</ins>
115	<ins> + - - - - - - - - ModellingRule: Optional</ins>
116	<ins> + - - - - - - - - Value: 90.0</ins>
117	<ins> + - - - - - - - - - Name: LeftMargin</ins>
118	<ins> + - - - - - - - - - Type: BaseVariableType</ins>
119	<ins> + - - - - - - - - - DataType: Size</ins>
120	<ins> + - - - - - - - - - ModellingRule: Optional</ins>
121	<ins> + - - - - - - - - - Value: 90.0</ins>

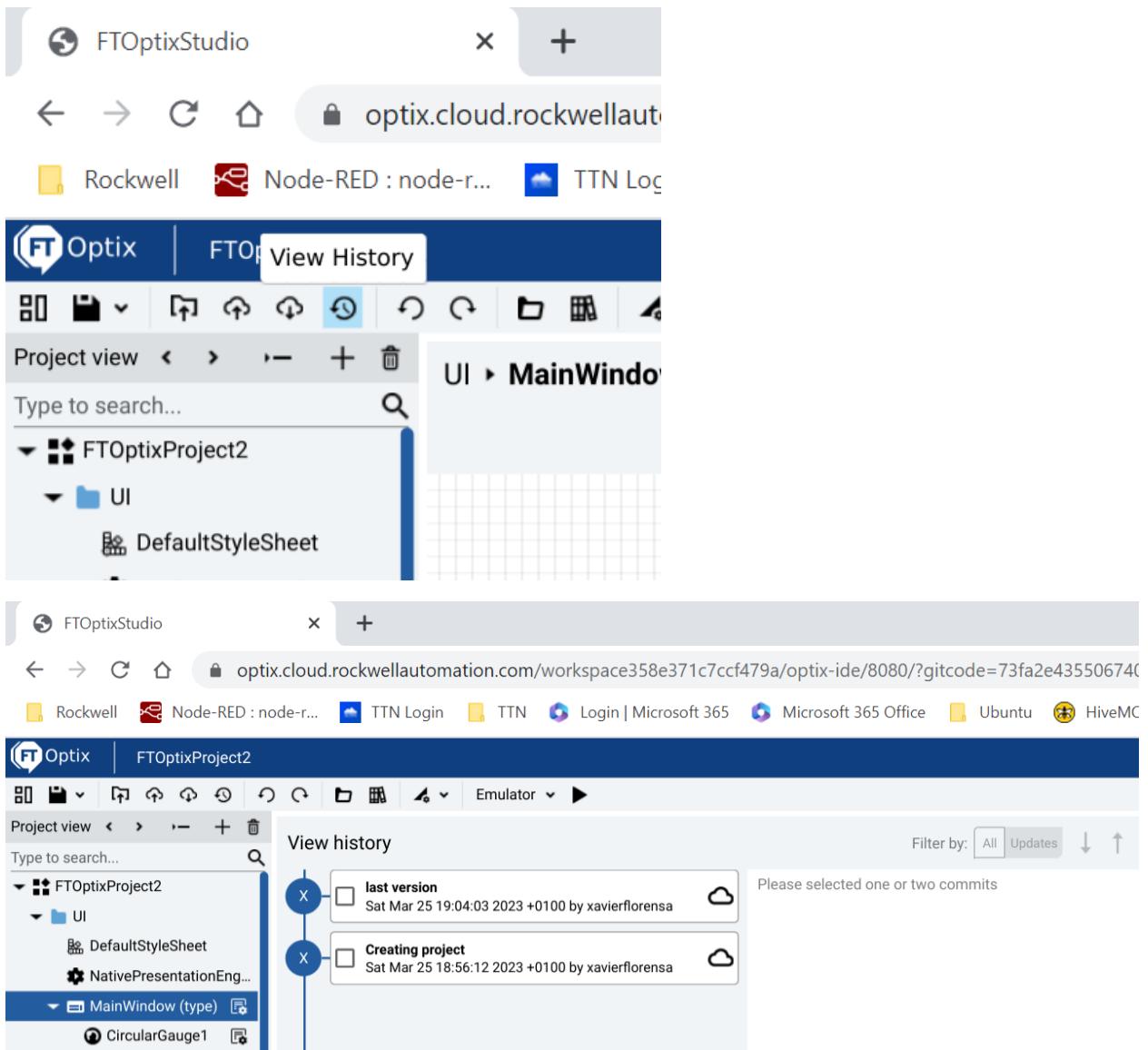
0 comments on commit `ff563b7`

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

You can also view history on Optix



16. Pushing your local Optix projects to Github

16.1. Using Optix

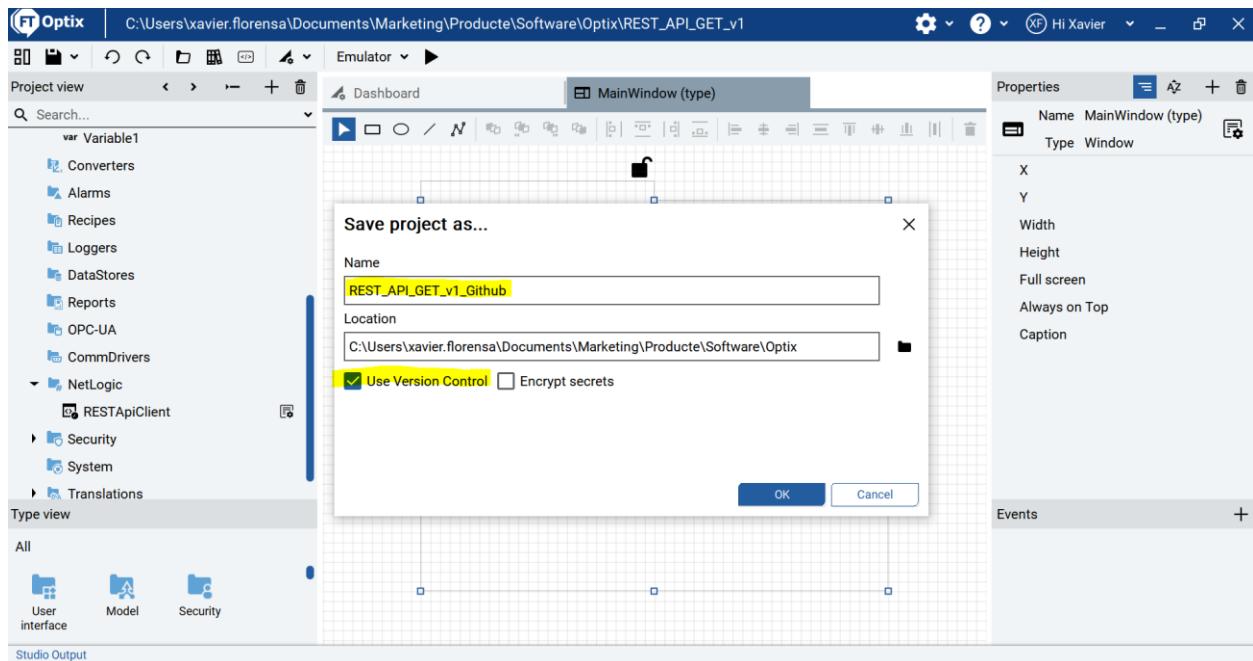
Probably you have started with Optix locally and created some projects that you want to store later on on Github without having to create them again.

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

For instance

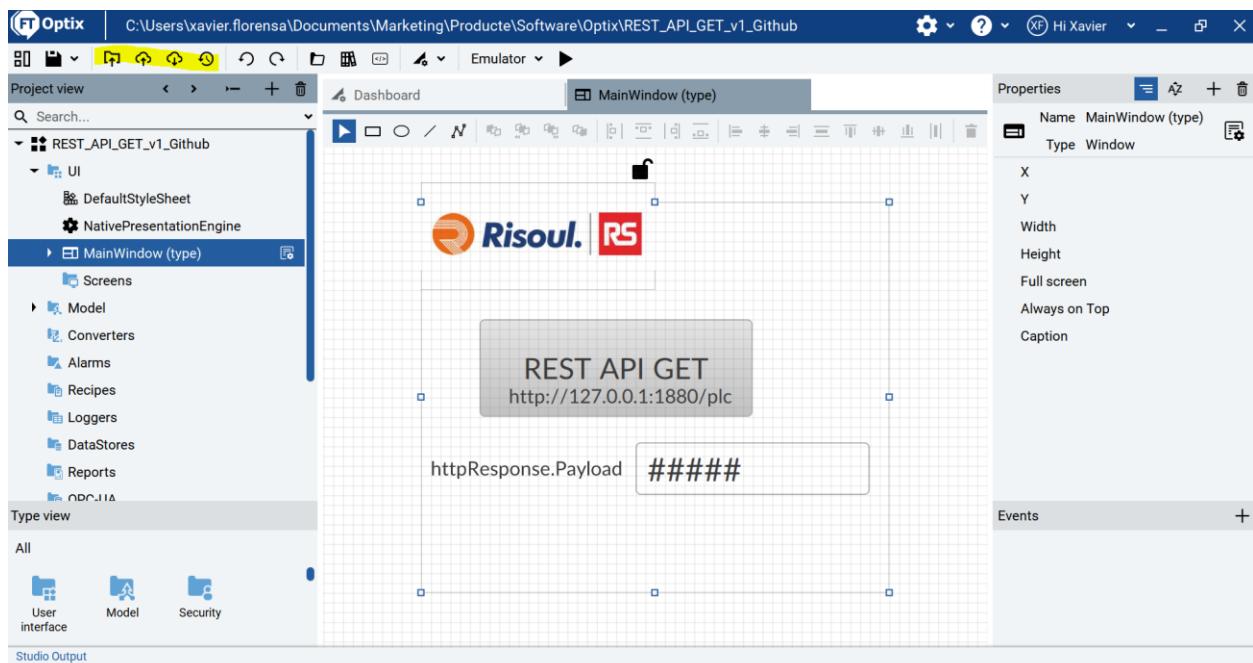
REST_API_GET_v1_Github

Mark "Use version control"



Click OK

Be aware that now your Optix local instance has changed



Also the files and directory has changed

Before

REST_API_GET_v1

File	Home	Share	View
This PC > Documents > Marketing > Producte > Software > Optix > REST_API_GET_v1			
Quick access	Name	Date modified	Type
Desktop	Nodes	12/9/2023 6:32 PM	File folder
Downloads	ProjectFiles	12/9/2023 2:55 PM	File folder
Documents	IDEVersion	12/4/2023 7:55 PM	Text Document
Enrafadora2-Ens	Logo	11/16/2023 8:55 PM	JPG File
Pictures	REST_API_GET_v1	12/9/2023 6:32 PM	FactoryTalk Optix Stu...
	REST_API_GET_v1.optix.design	12/9/2023 6:32 PM	DESIGN File

After

REST_API_GET_v1_Github

File	Home	Share	View
This PC > Documents > Marketing > Producte > Software > Optix > REST_API_GET_v1_Github			
Quick access	Name	Date modified	Type
Desktop	Nodes	12/17/2023 6:43 PM	File folder
Downloads	ProjectFiles	12/17/2023 6:43 PM	File folder
Documents	.gitattributes	7/28/2023 3:39 PM	Text Document
Enrafadora2-Ens	.gitignore	7/28/2023 3:39 PM	Text Document
Pictures	IDEVersion	12/4/2023 7:55 PM	Text Document
Guio	REST_API_GET_v1_Github	12/17/2023 6:43 PM	FactoryTalk Optix Stu...
	REST_API_GET_v1_Github.optix.design	12/17/2023 6:43 PM	DESIGN File

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 *

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)

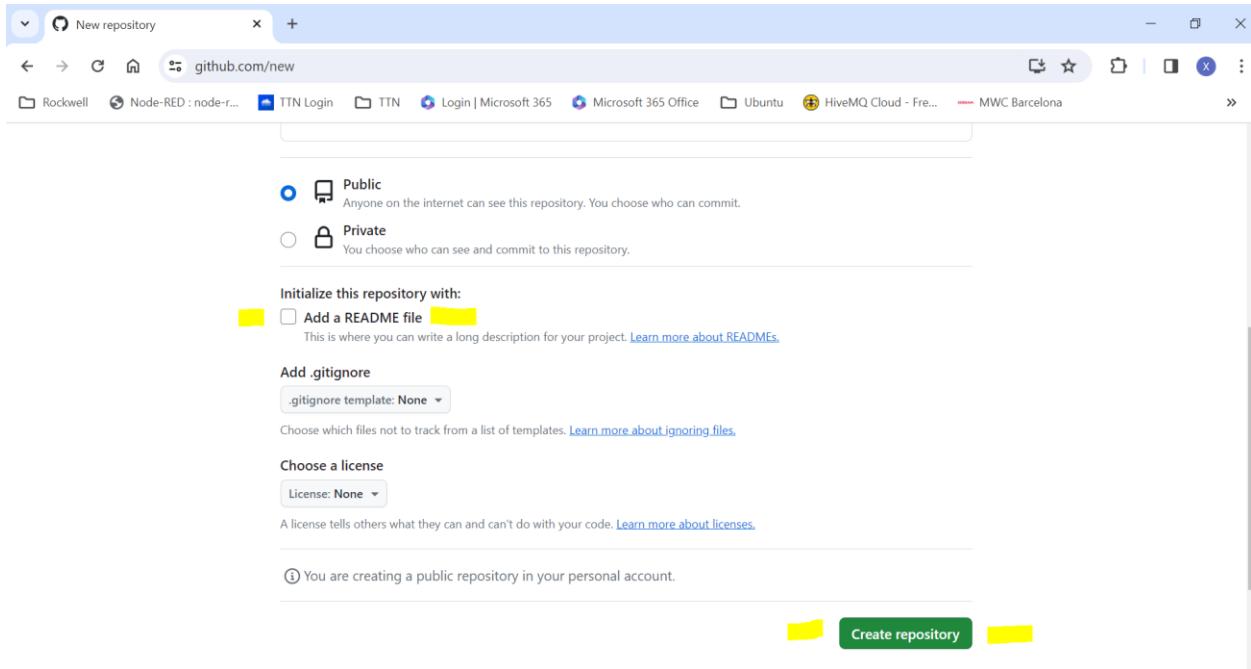
Public

Anyone on the internet can see this repository. You choose who can commit.

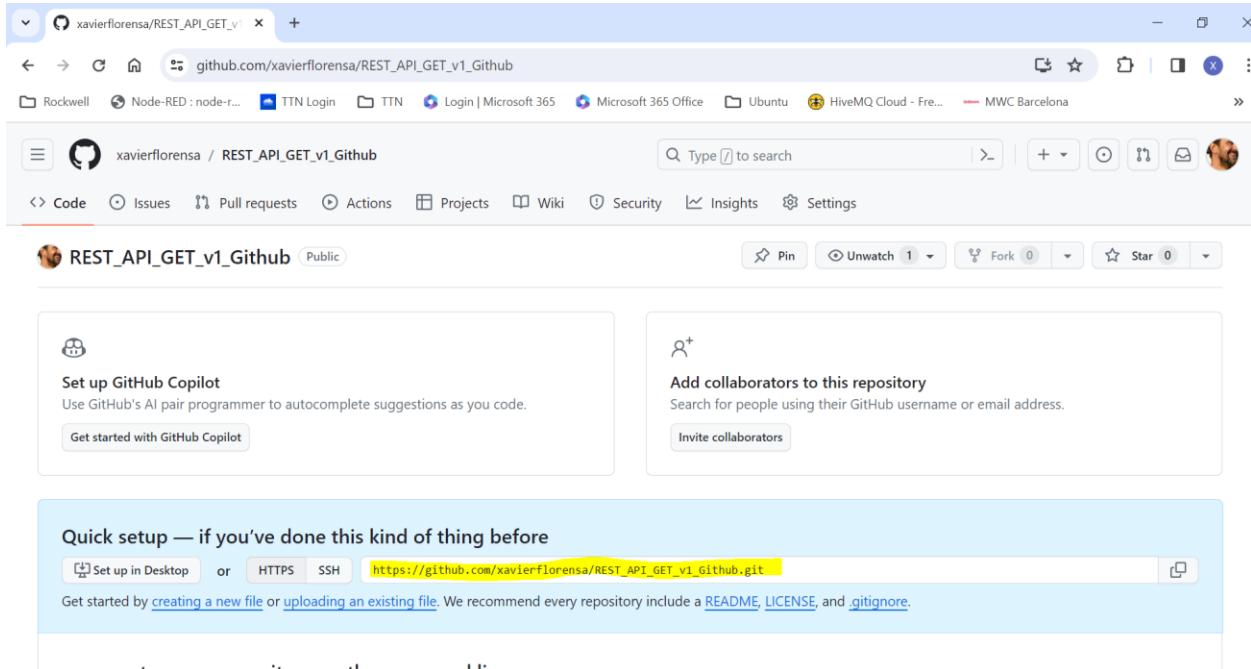
Private

You choose who can see and commit to this repository.

Do not check create a readme file



Click on create



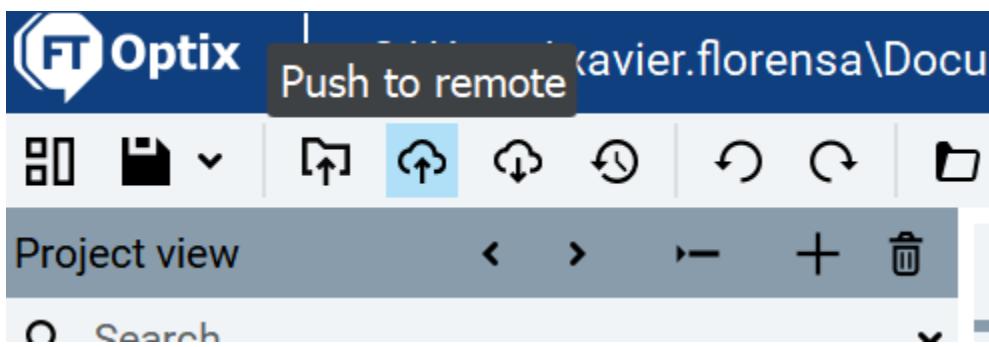
Copy this address

https://github.com/xavierflorensa/REST_API_GET_v1_Github.git

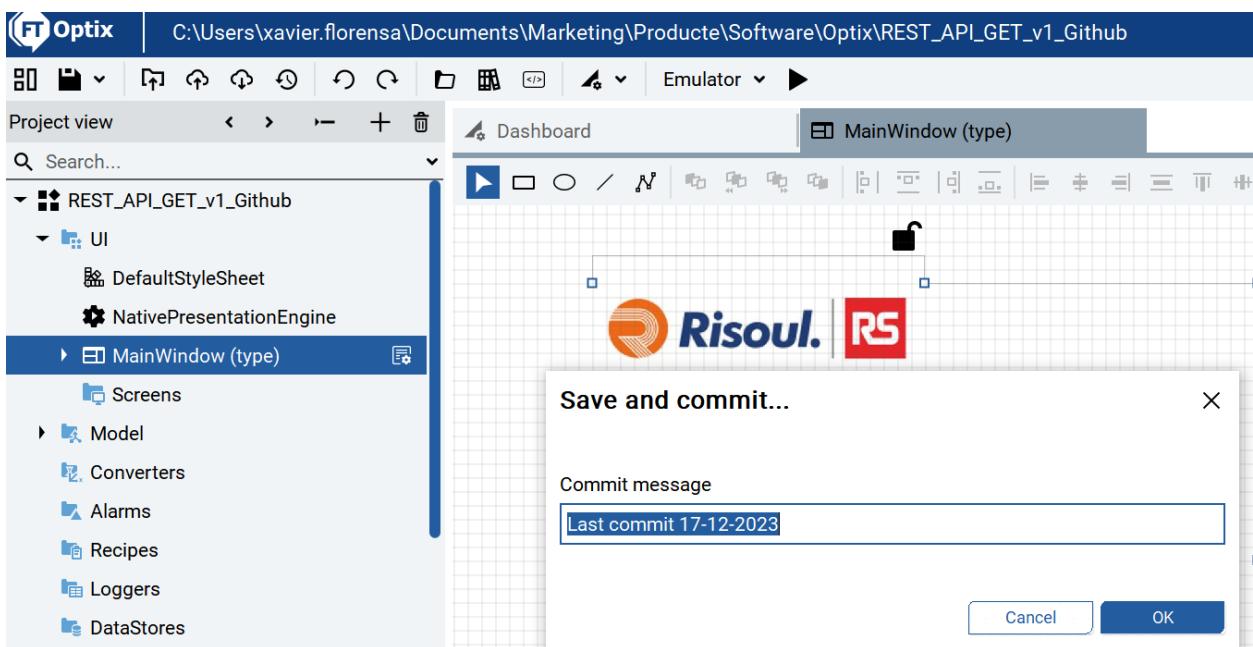
Your new repository is empty.

Now go to Optix Studio local

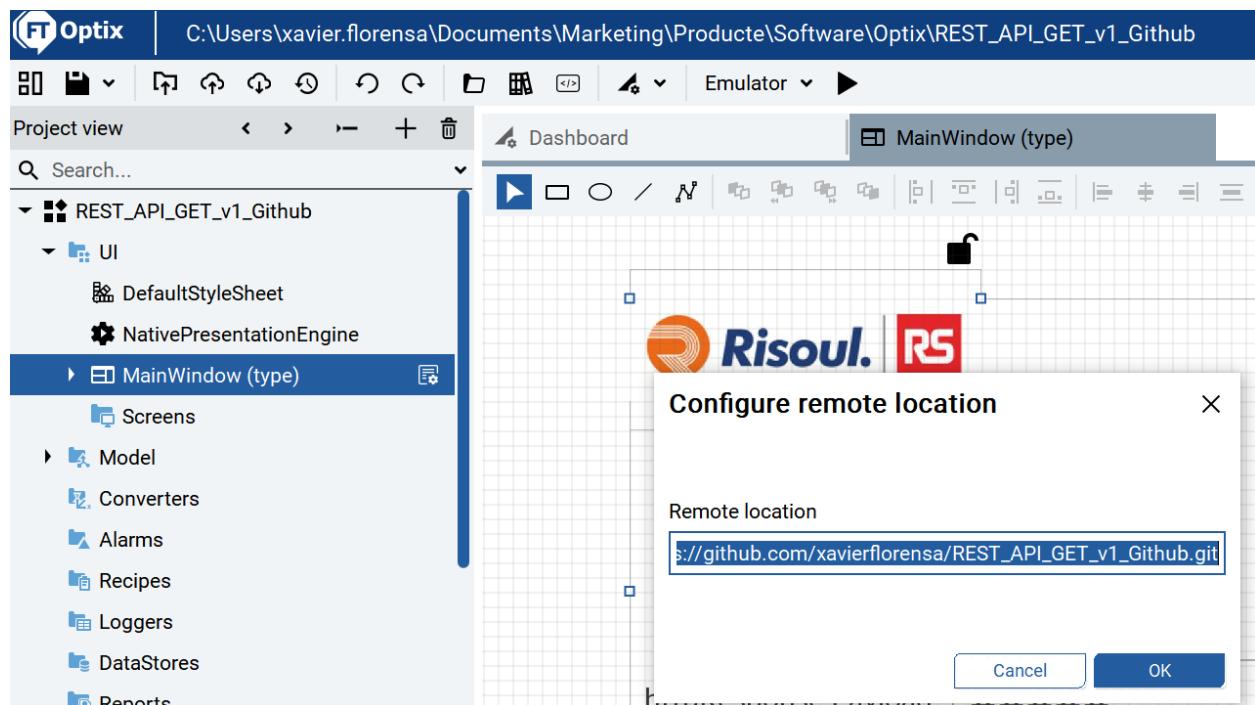
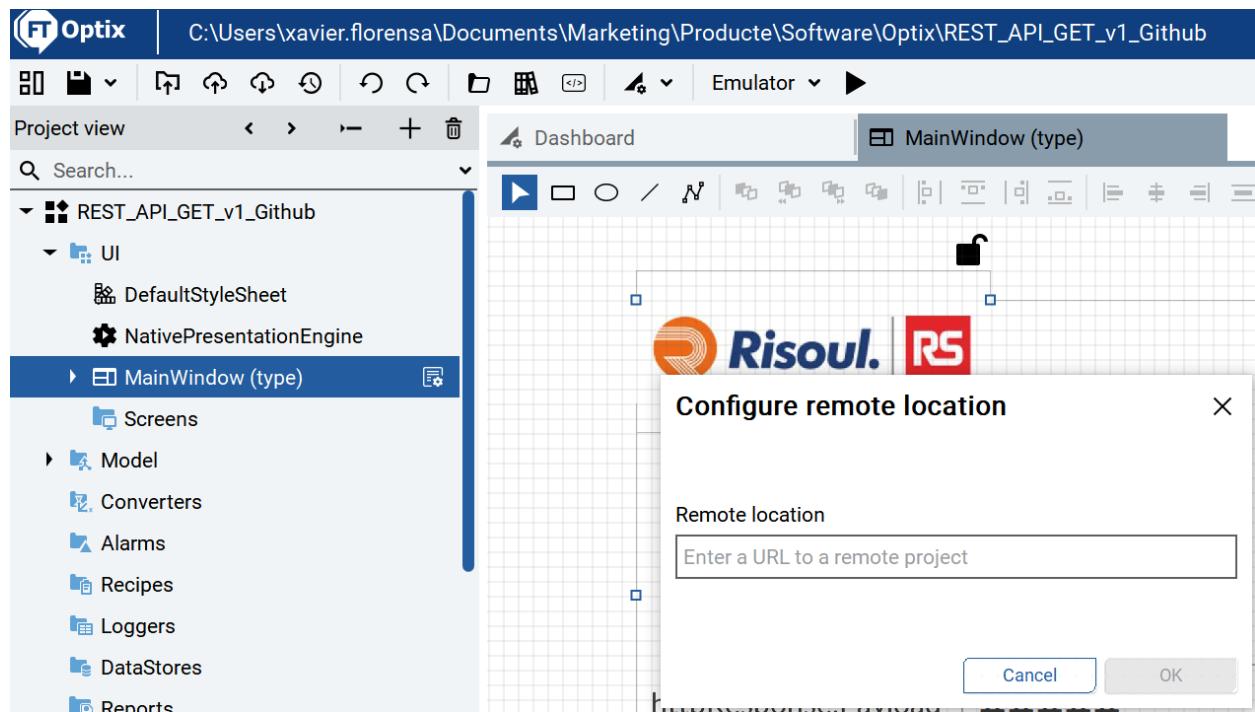
And hit on "Push to remote"



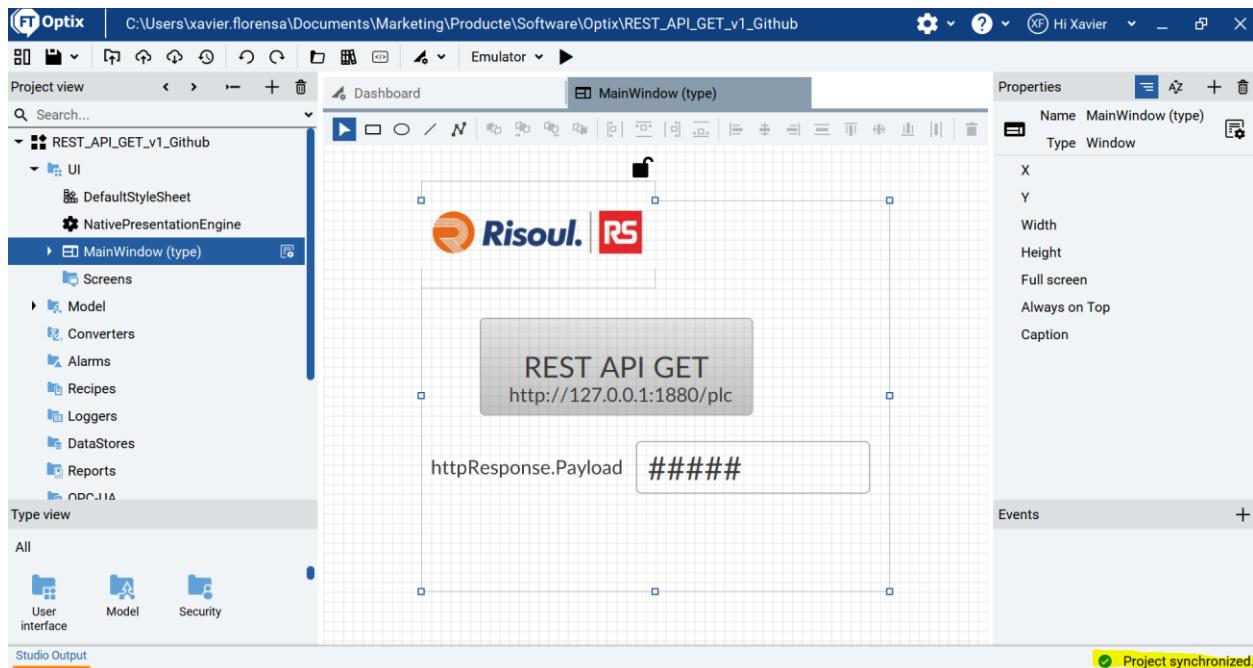
Give a comment if needed



Give here the copied url address



You get no errors and project is synchronized



Now look at your project in Github

xavierflorensa / REST_API_GET_v1_Github

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

REST_API_GET_v1_Github Public

main 1 Branch 0 Tags

Go to file Add file Code

About

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

github.com/xavierflorensa/REST_API_GET_v1_Github

Rockwell Node-RED : node-r... TTN Login TTN Login | Microsoft 365 Microsoft 365 Office Ubuntu HiveMQ Cloud

xavierflorensa / REST_API_GET_v1_Github Type to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

REST_API_GET_v1_Github Public Pin Unwatch 1

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

xavierflorensa Last commit 17-12-2023 58d1e73

+ Create new file
Upload files

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

https://github.com/xavierflorensa/REST_API_GET_v1_Github/upload/main

github.com/xavierflorensa/REST_API_GET_v1_Github/upload/main

Rockwell Node-RED : node-r... TTN Login TTN Login | Microsoft 365 Microsoft 365 Office Ubuntu HiveMQ Cloud - Fre... MWC Barcelona

xavierflorensa / REST_API_GET_v1_Github Type to search

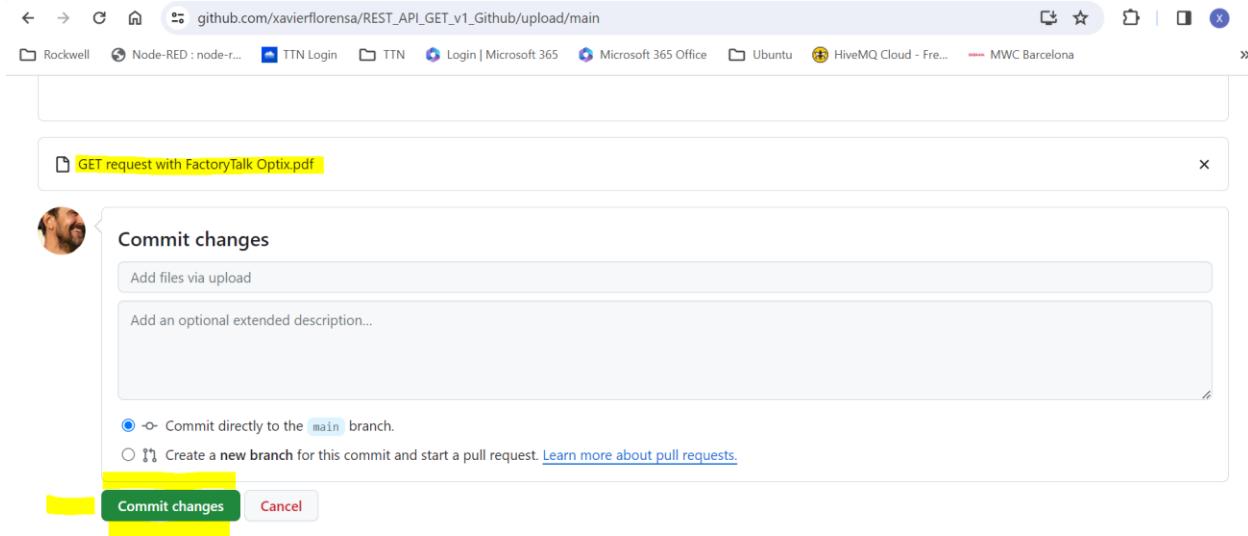
Code Issues Pull requests Actions Projects Wiki Security Insights Settings

REST_API_GET_v1_Github /

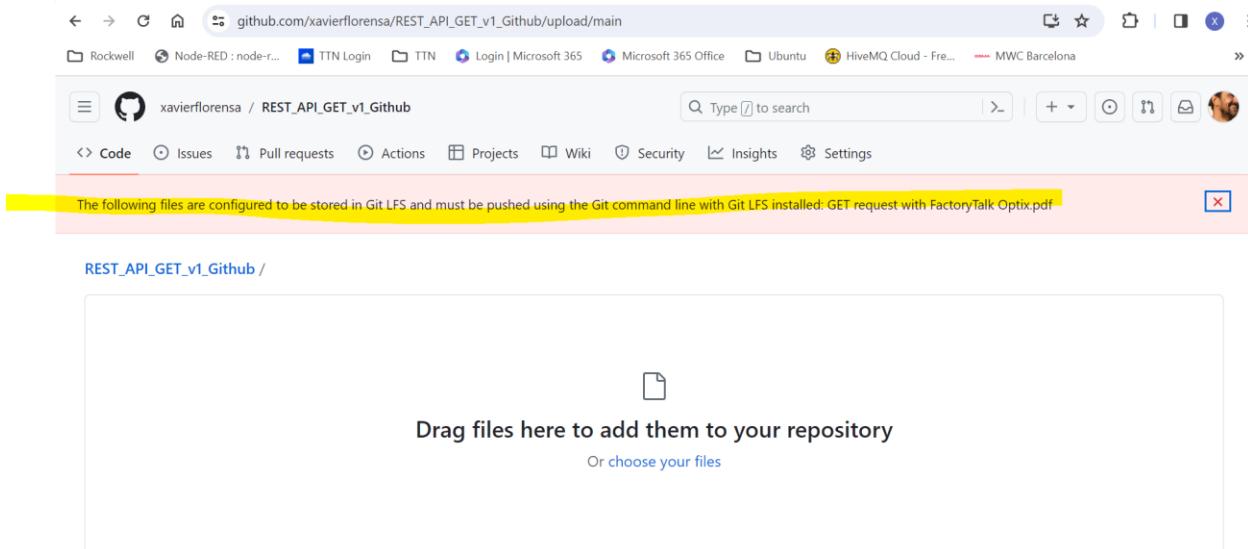
Drag files here to add them to your repository
Or choose your files

Commit changes

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.

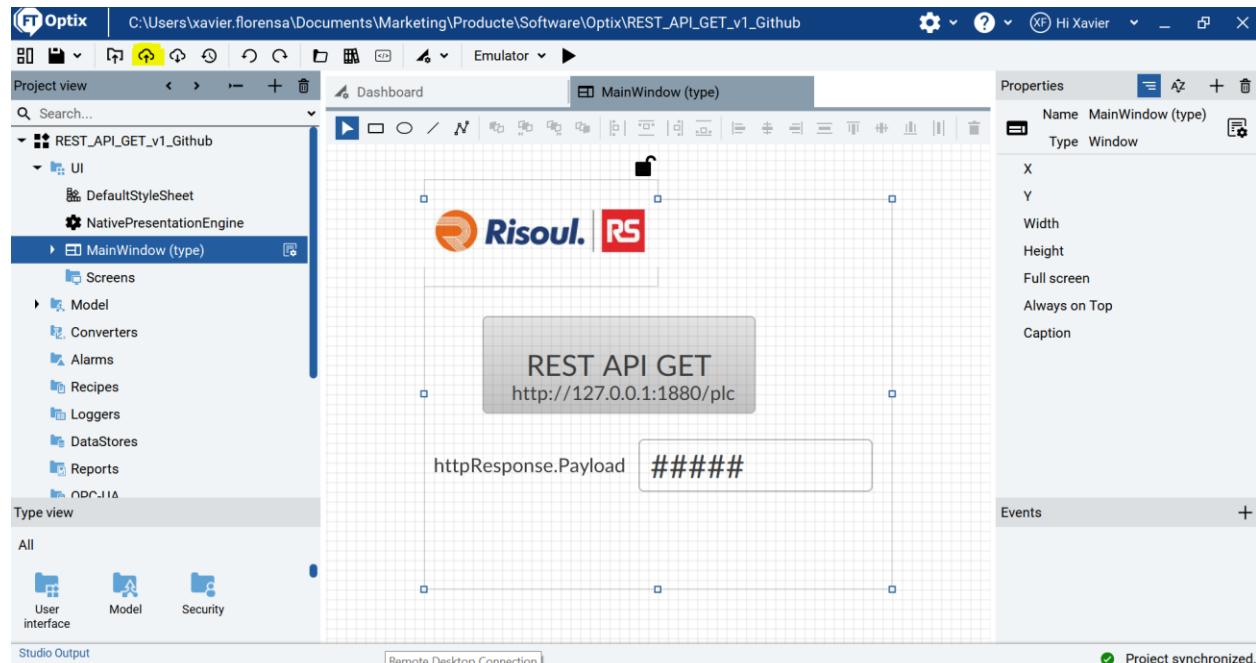
Like this

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
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

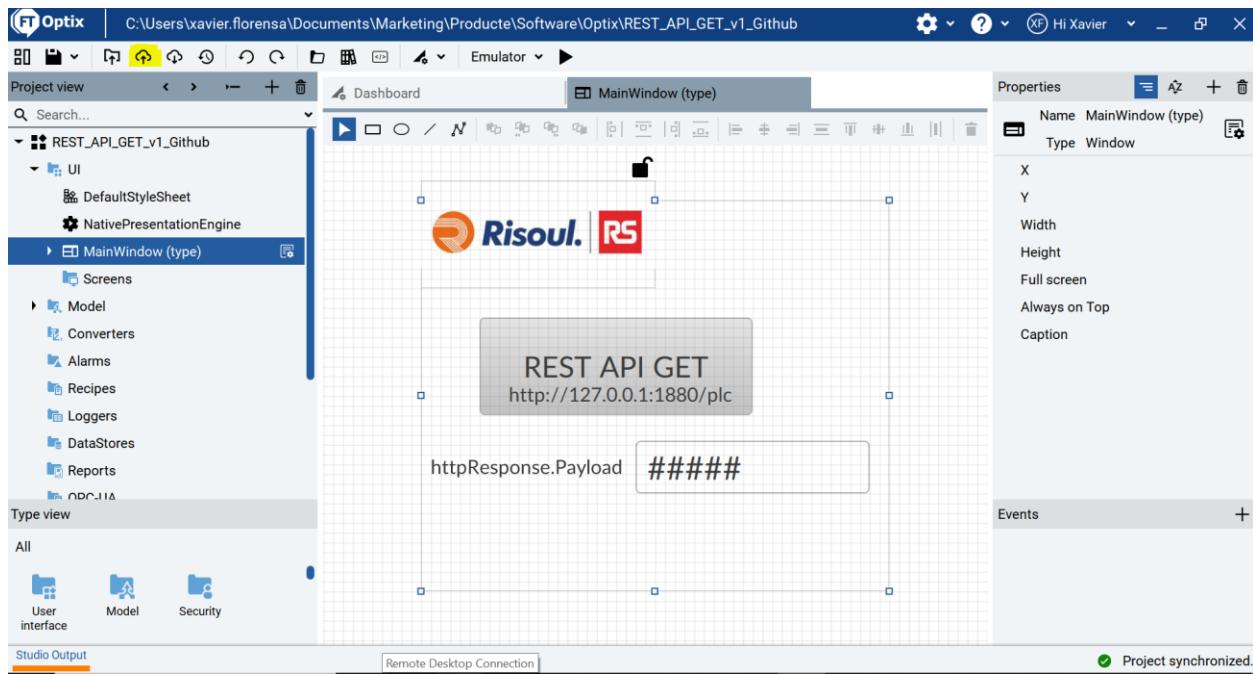
I add the documentation file I have forgot

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
GET request with FactoryTalk Optix	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

REST_API_GET_v1_Github (Public)

Last commit 17-12-2023

- Nodes
- ProjectFiles
- .gitattributes
- .gitignore
- REST_API_GET_v1_Github.optix
- REST_API_GET_v1_Github.optix.design

README

About

No description, website, or topics provided.

Activity

1 commits

0 stars

1 watching

0 forks

Releases

No releases published

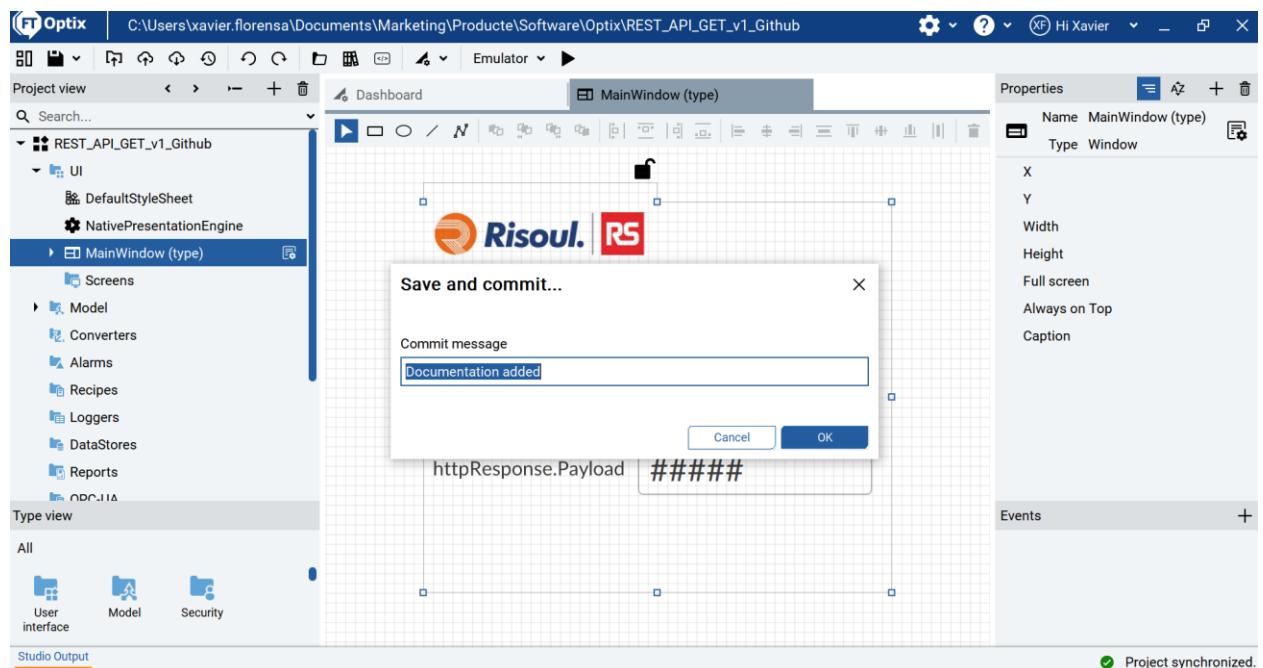
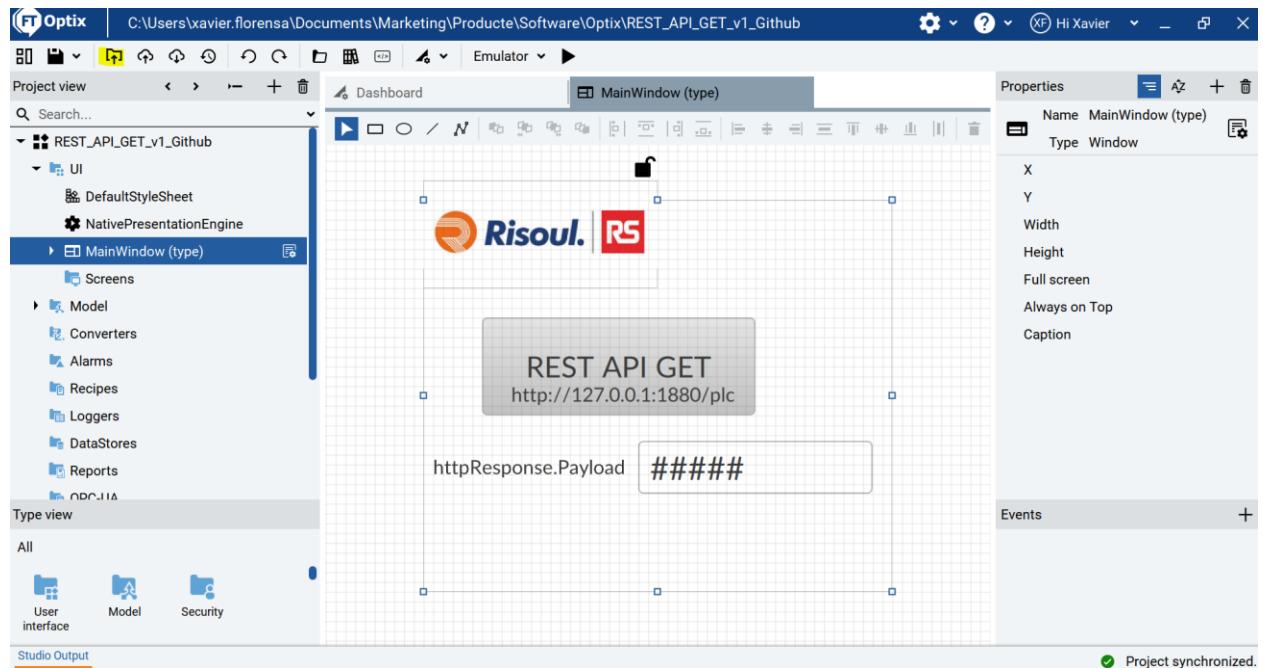
[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Let's try with save and commit



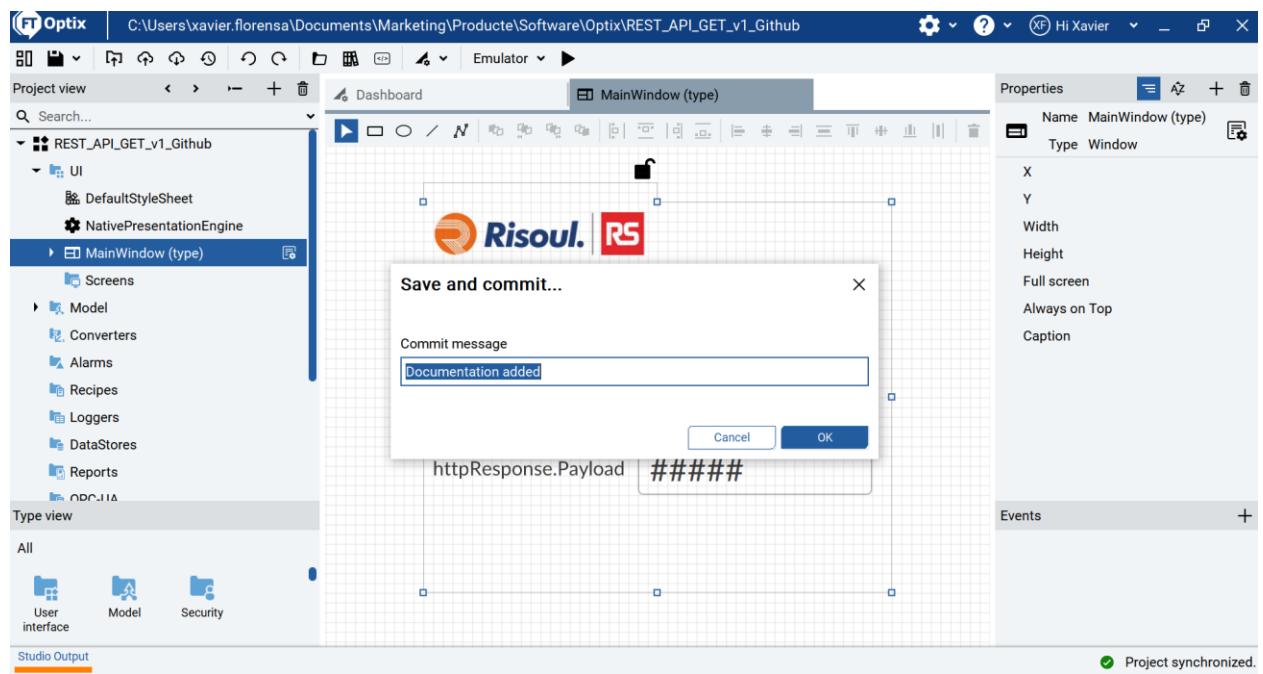
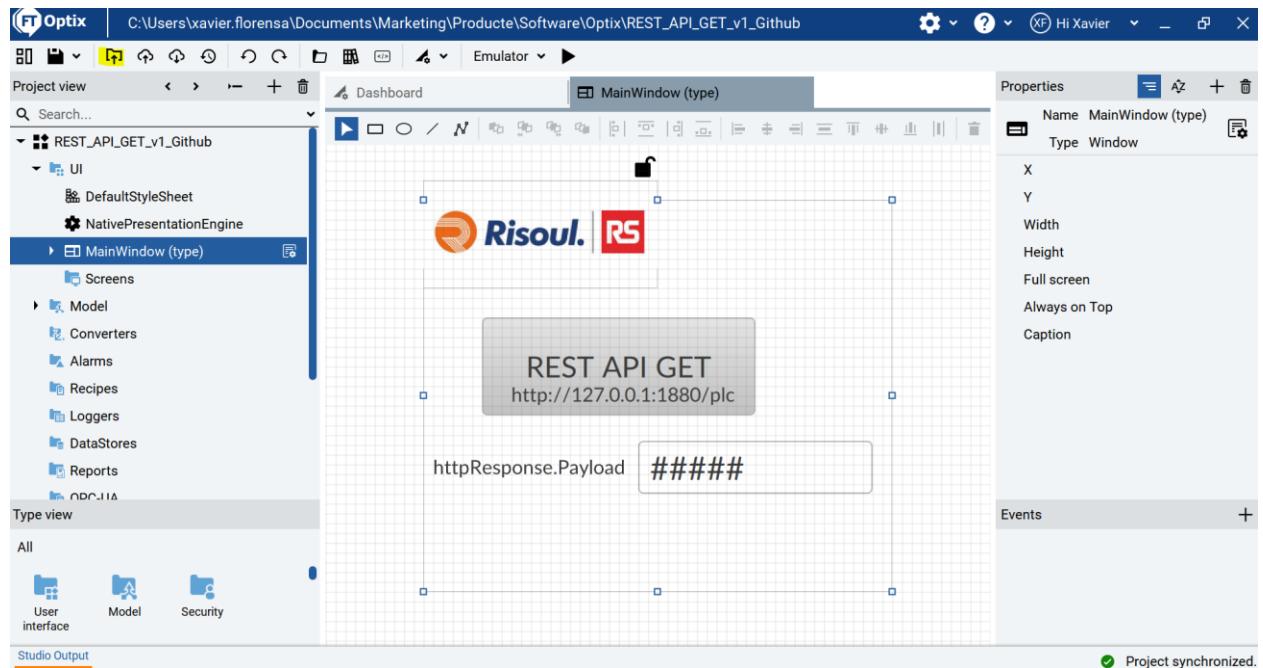
Nothing changes

Let's try to copy the file to Project files directory on our local directory

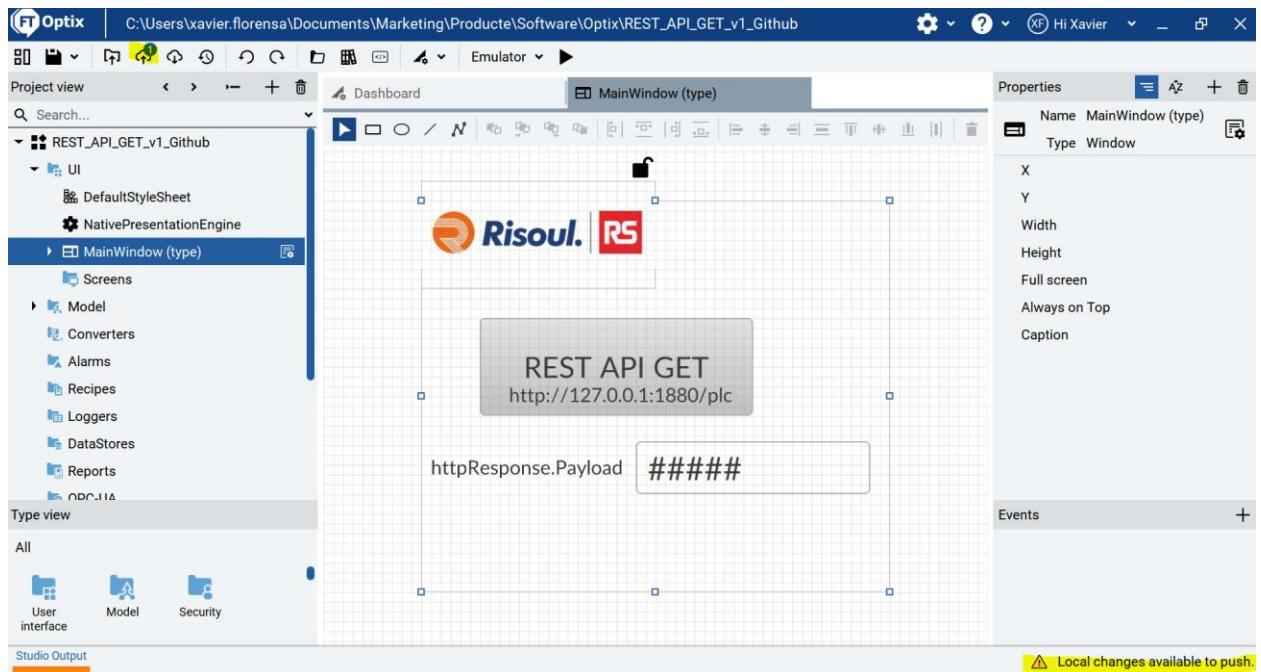
Close and open Optix project

Now let's repeat

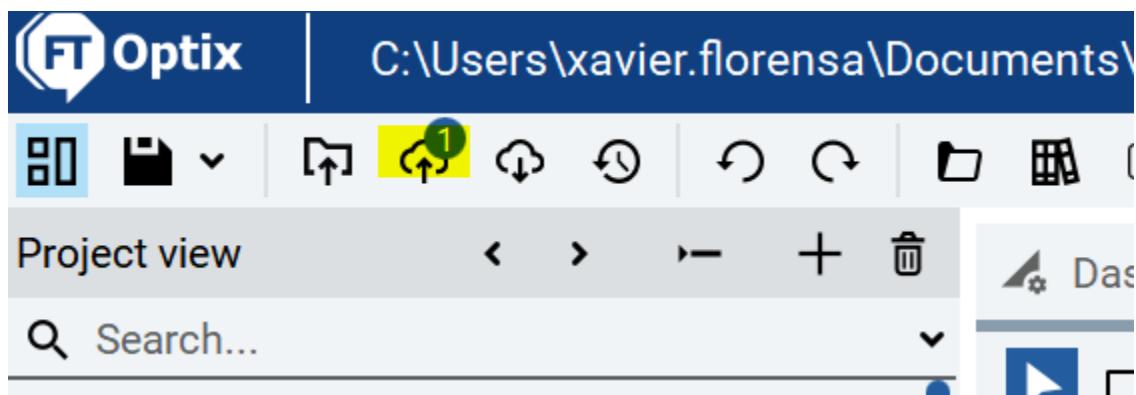
Save and commit



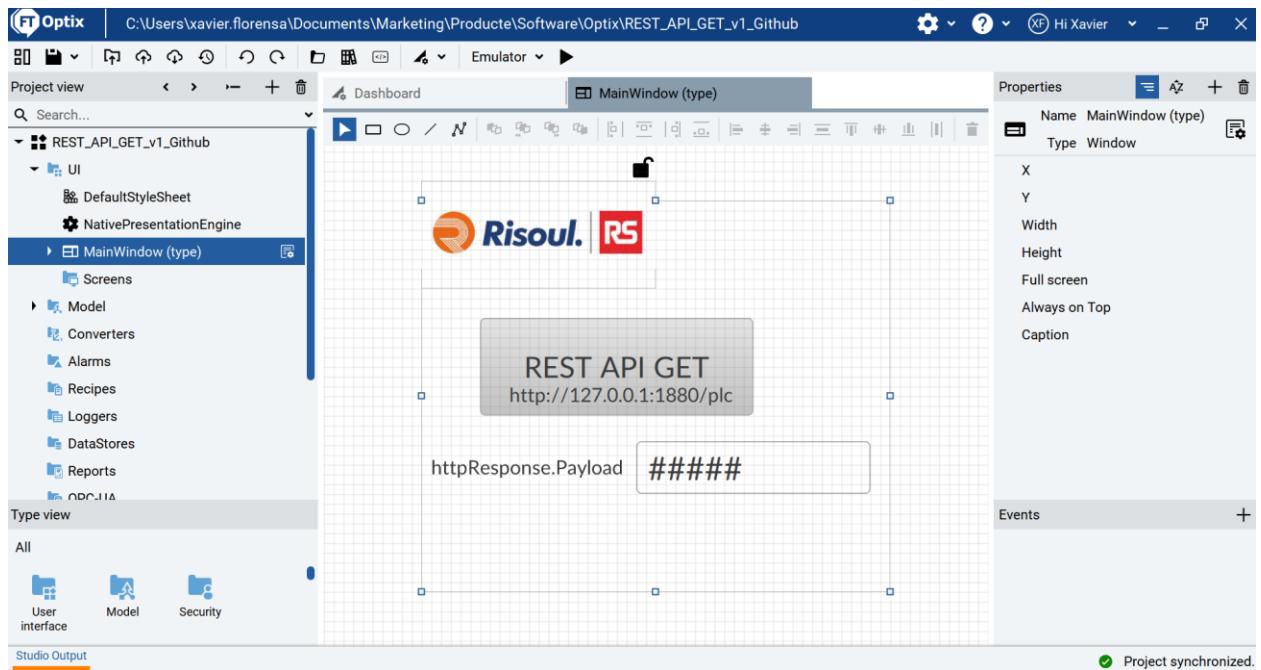
Something has changed now



If we click on Push



Now project is synchronized



And if we open our Github project

Voilà, here it is

The screenshot shows a GitHub repository page for 'xavierflorensa / REST_API_GET_v1_Github'. The left sidebar shows a tree view with 'main' selected, containing 'Nodes', 'ProjectFiles' (selected), 'NetSolution' (with files 'GET request with FactoryTalk O...', 'Logo.JPG', '.gitattributes', '.gitignore'), and other files like '.gitattributes' and '.gitignore'. The main area shows a commit history:

- xavierflorensa Documentation added 270fef5 - 6 minutes ago

Name	Last commit message	Last commit date
...		
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

16.2. Using 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

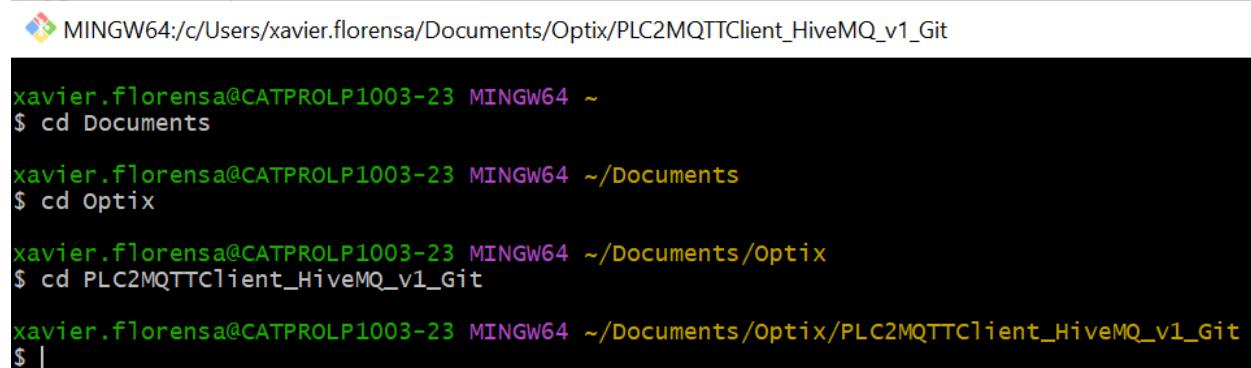
C:\Users\xavier.florensa\Documents\Optix

Open Git Bash



```
xavier.florensa@CATPROLP1003-23 MINGW64 ~
```

Navigate to the root directory of your project



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

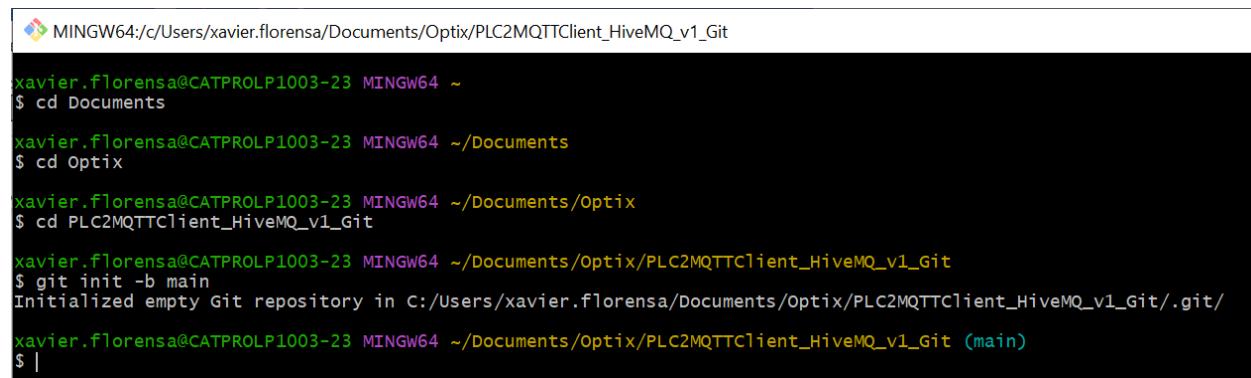
xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents
$ cd Optix

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

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

Iniciale a Git repository locally

git init -b main



```
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.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/reft/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 provav35v0.ACD
create mode 100644 provav35v0.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@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 (main)
$ git log
commit 8ca9b565b9b143c310ddbf6b92ba5e4d161d30f2 (HEAD -> main, origin/main)
Author: Xavier Florensa Berenguer <xavier.florensa@soporte.com>
Date:   Sun Dec 17 19:42:22 2023 +0100

    First commit

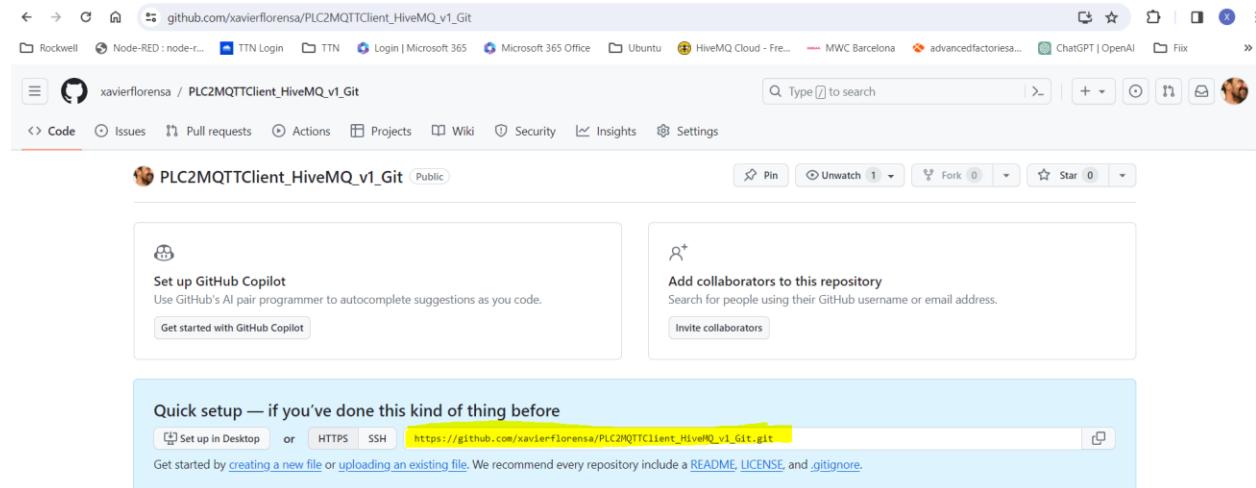
xavier.florensa@CATPROLP1003-23 MINGW64 ~/documents/optix/plc2mqttclient_hivemq_v1_git (main)
$ |

```

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

Copy this url

https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git



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

```
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

```
MINGW64:/c/Users/xavier.florensa/documents/optix/plc2mqttclient_hivemq_v1_git
xavier.florensa@CATPROLP1003-23 MINGW64 ~/documents/optix/plc2mqttclient_hivemq_v1_git (main)
$ git remote -v
origin  https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git (fetch)
origin  https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git (push)

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

```
git push -u origin main
You need to authenticate
```


Connect to GitHub

X

GitHub

Sign in

Browser/Device

Token

[Sign in with your browser](#)

[Sign in with a code](#)

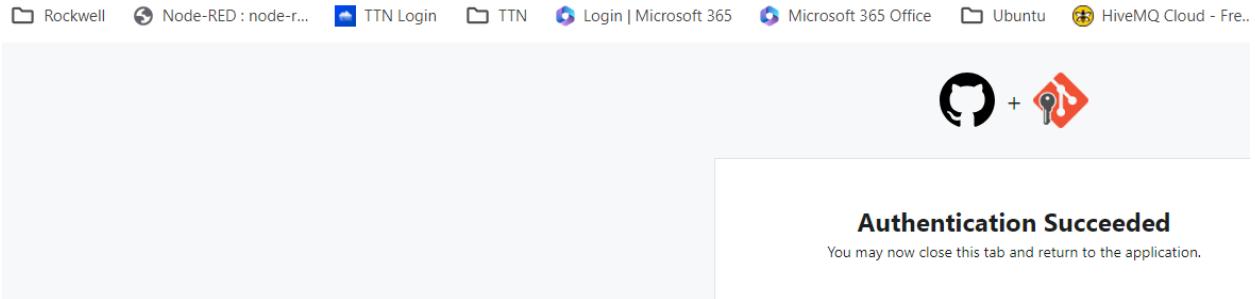
Don't have an account? [Sign up](#)

The screenshot shows the GitHub OAuth authorization interface. At the top, there's a header bar with navigation icons and the URL `github.com/login/oauth/authorize?response_type=code&client_id=0120e057bd645470c1ed&state=96a1d86d17794fb98cc3331f73a`. Below the header, a banner says "Authorize Git Credential Manager". It features a circular icon with a red key and a green checkmark next to a GitHub logo. The main content area displays the following information:

- Git Credential Manager by Git Ecosystem** wants to access your `xavierflorensa` account.
- Gists**: Read and write access
- Repositories**: Public and private
- Workflow**: Update GitHub Action Workflow files.
- Organization access**: PacktPublishing ✓

At the bottom are two buttons: "Cancel" and "Authorize git-ecosystem".

The screenshot shows the GitHub password confirmation interface. At the top, there's a header bar with navigation icons and the URL `github.com/login/oauth/authorize`. Below the header, a GitHub logo is displayed above the text "Confirm access". A message box shows "Signed in as @xavierflorensa". A form box contains a "Password" field with a blue border and a "Forgot password?" link. A large green "Confirm" button is at the bottom of the form. A tip at the bottom right reads: "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)

```
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 commit history lists several files added via upload, including 'DesignTimeNodes/CommDrivers/RAEtherNet...', 'Nodes', 'ProjectFiles', 'FactoryTalk Optix and MQTT.pdf', 'IDEVersion.txt', and 'PLC2MQTTClient_HiveMQ_v1_Git.optix'. A tooltip over the 'Add file' button indicates the option to 'Upload files'.

https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git/upload/main

And you have it there

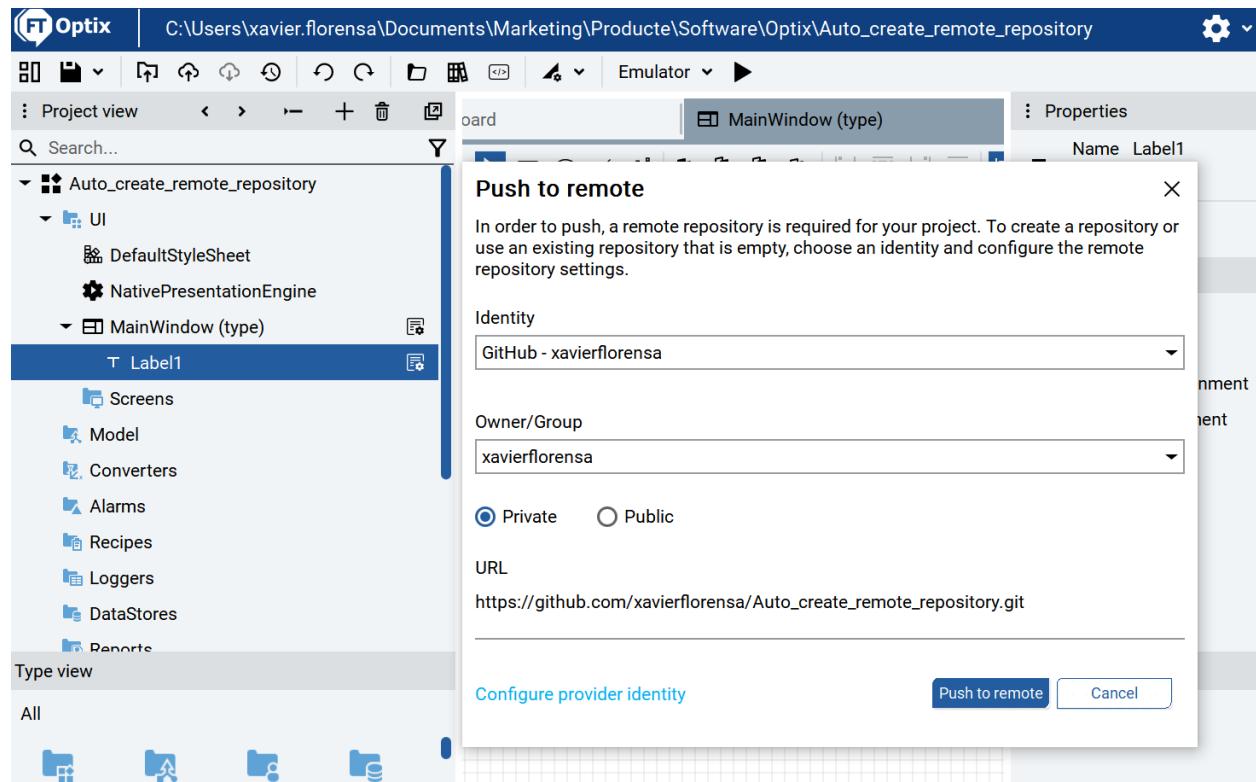
The screenshot shows the same GitHub repository page after the file has been uploaded. The commit history now includes the newly uploaded 'FactoryTalk Optix and MQTT.pdf' file. The file is highlighted in yellow in the commit list.

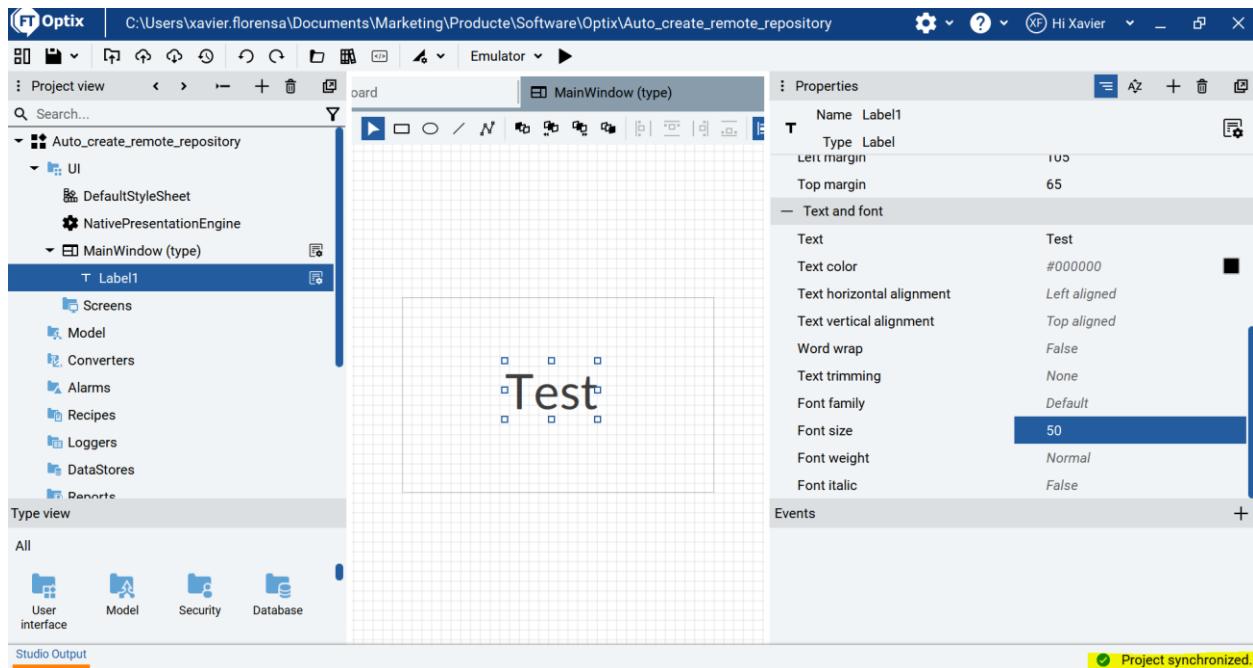
This is not possible if you have used Optix to push the repository

16.3. 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

Find a repository... Type Language Sort New

[Auto_create_remote_repository](#) Private Updated now

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

[NetlogicTutorialSocketServer5](#) Public Updated now

A new repository appeared

Auto_create_remote_repository (Private)

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

README

About: No description, website, or topics provided.

Activity: 0 stars, 1 watching, 0 forks.

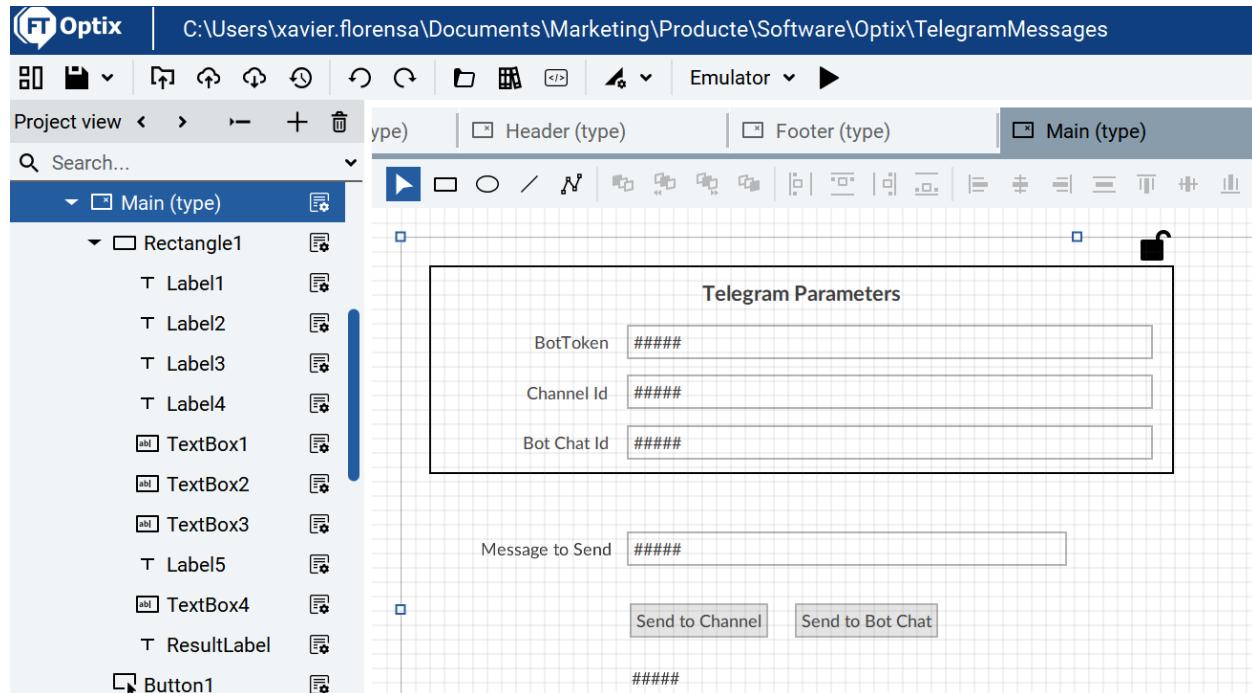
Releases: No releases published. Create a new release.

Packages: No packages published. Publish your first package.

17. Sending Telegram messages

You can use the example from this repository

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



18. 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

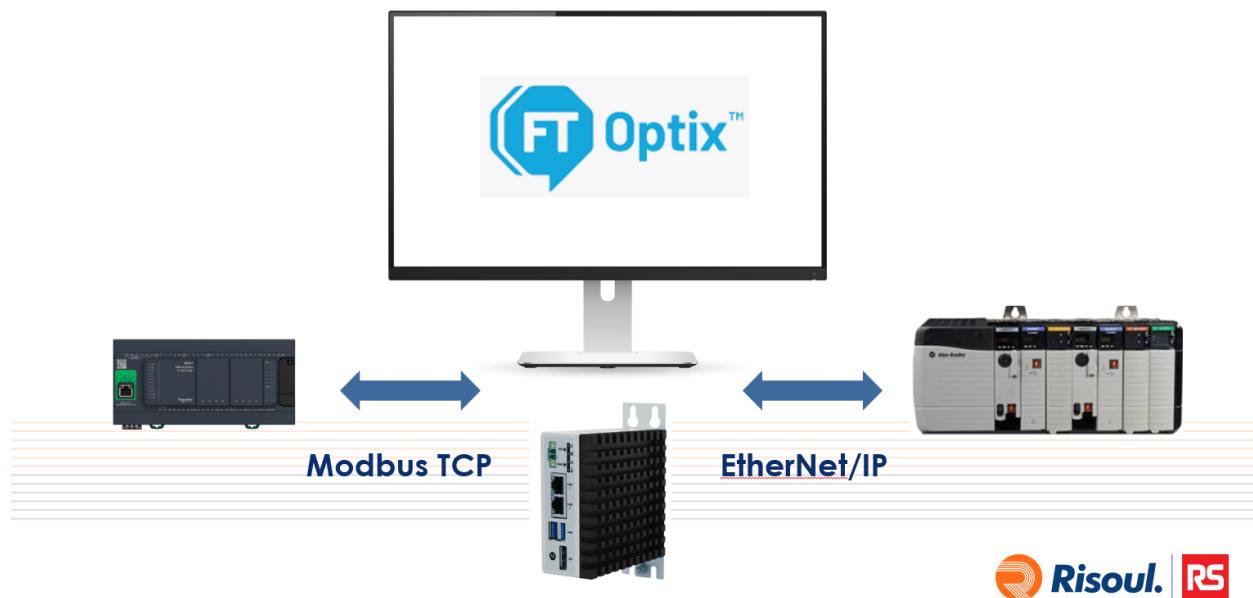
You can see the steps on this video

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

To be able to use the Tag importer take a look at chapter [Design time libraries](#)

19. 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%

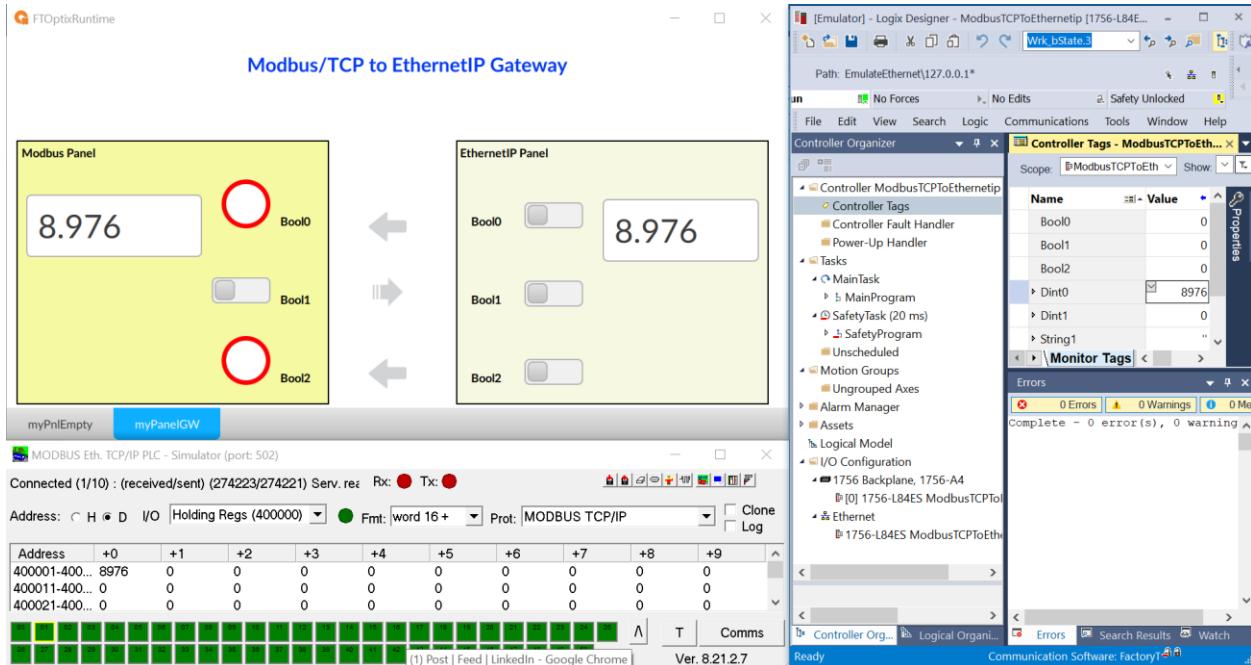
Gateway from Modbus to Ethernet IP with Optix

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

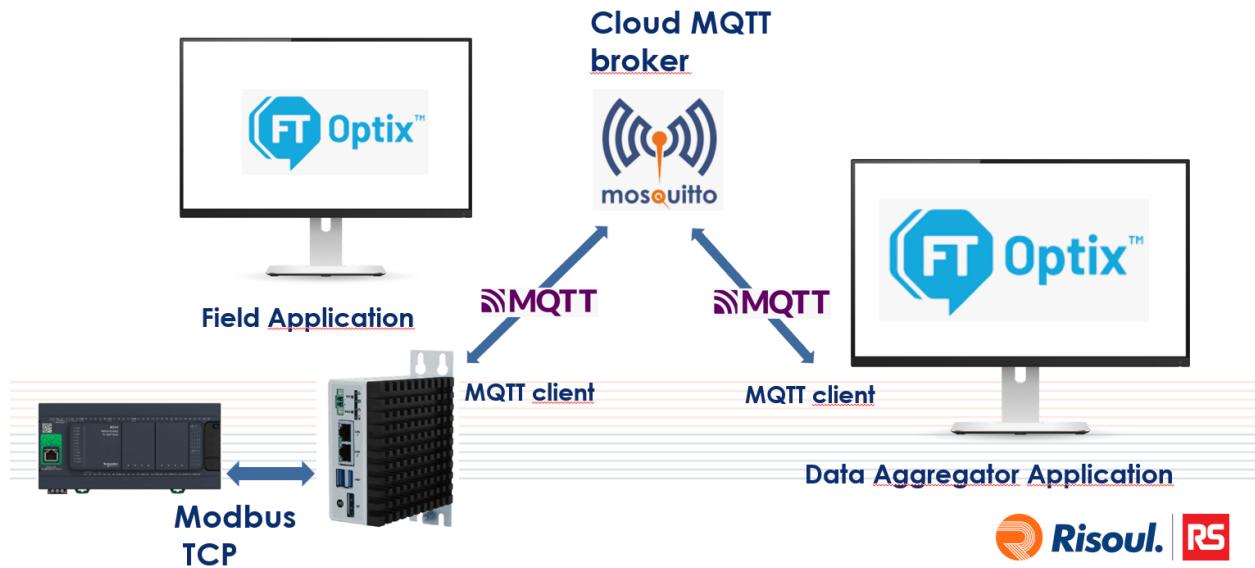


As you can see on this video

<https://youtu.be/UMII0TQ1uGM>

20. Modbus to MQTT data aggregator

Modbus TCP to MQTT data aggregator



You can see the process here

<https://youtu.be/PRgpzygpefA>

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

21.1. Design time libraries

Let's import one library to our environment.

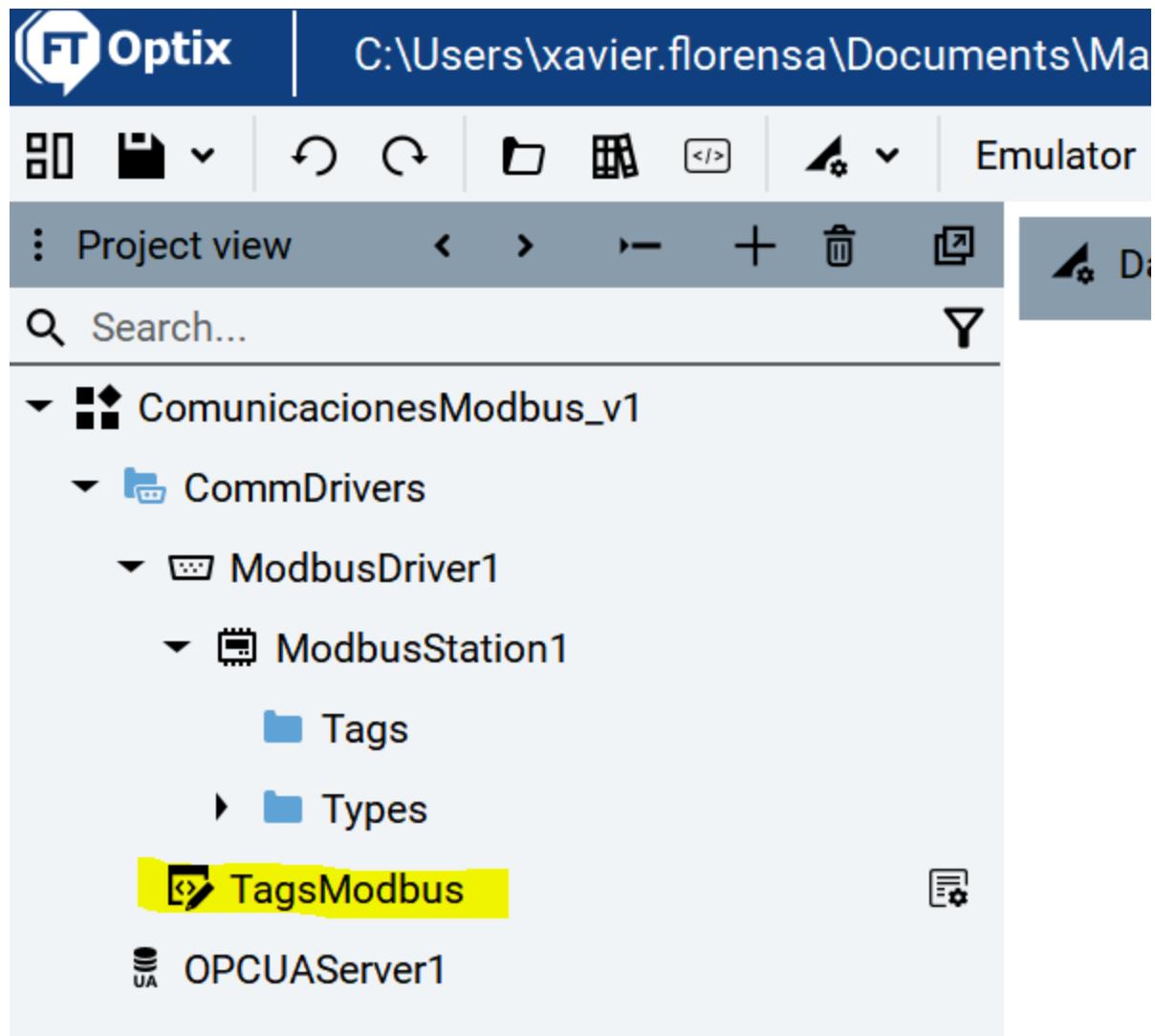
For instance this useful Modbus variable generator described on this video

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

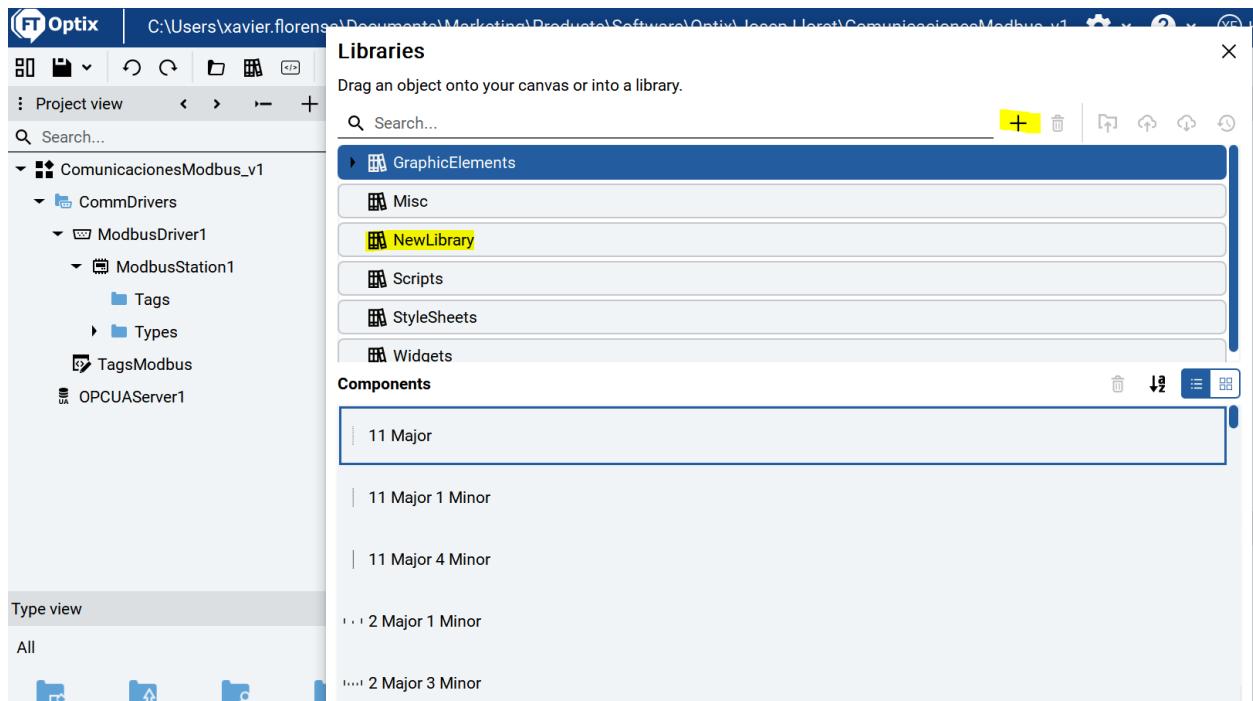
From an example code I can take the .net code and store in my libraries collection to be used later

Let's open the example program

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

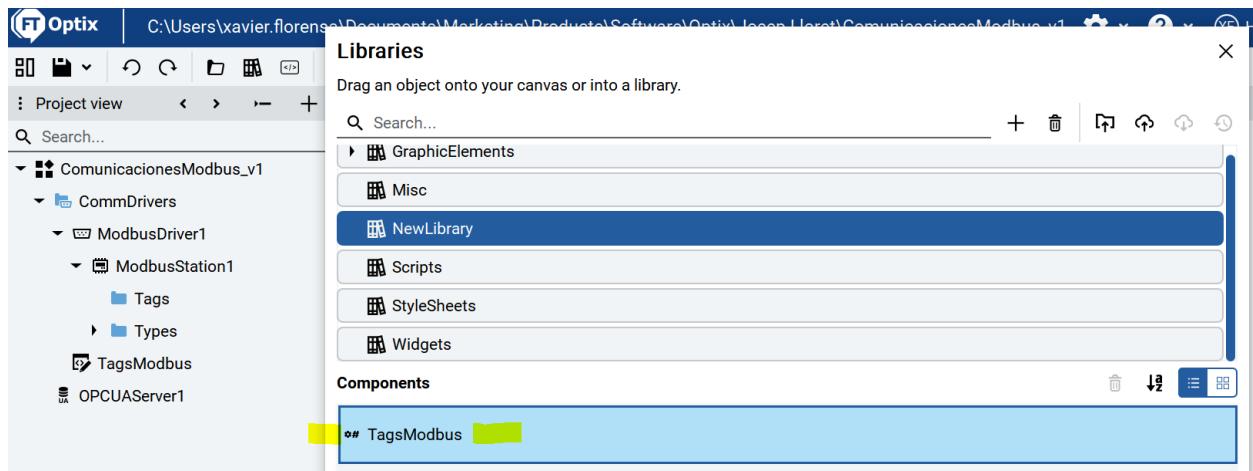


Let's open libraries and create a new library



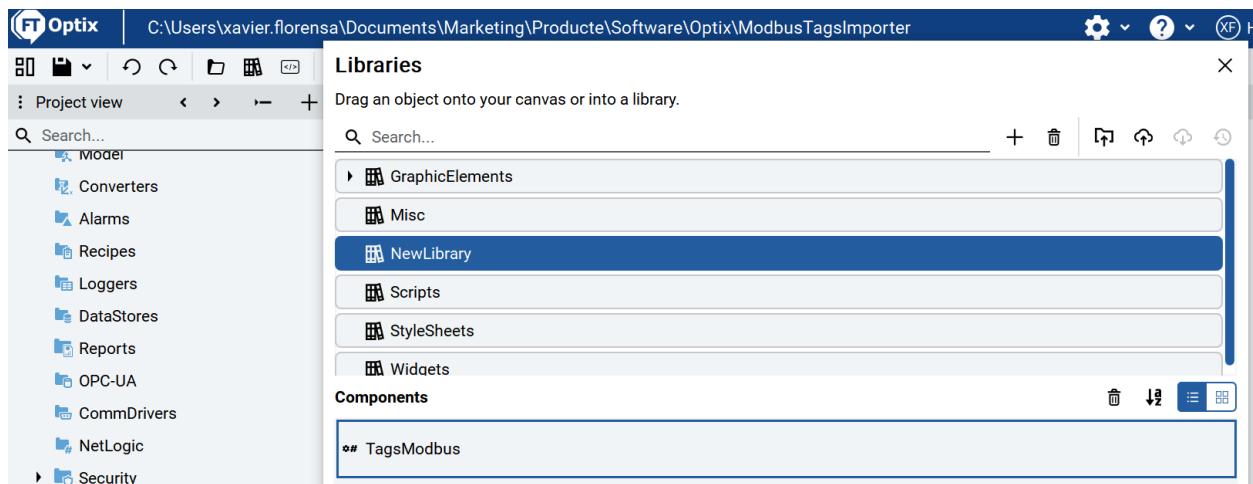
Then drag and drop the code object to the library

We have it

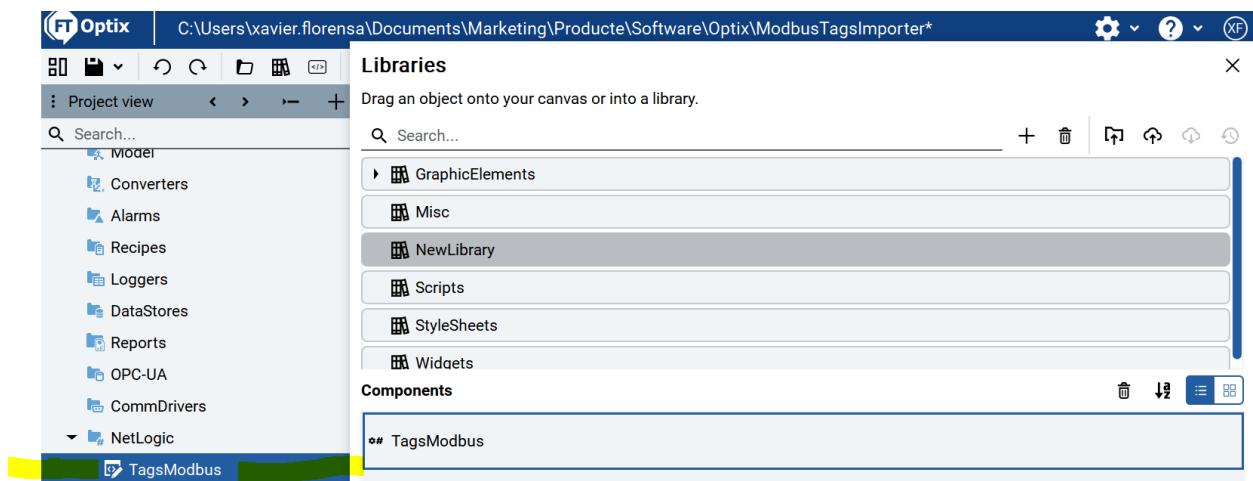


Now let's try to use on a new project

Create a new project



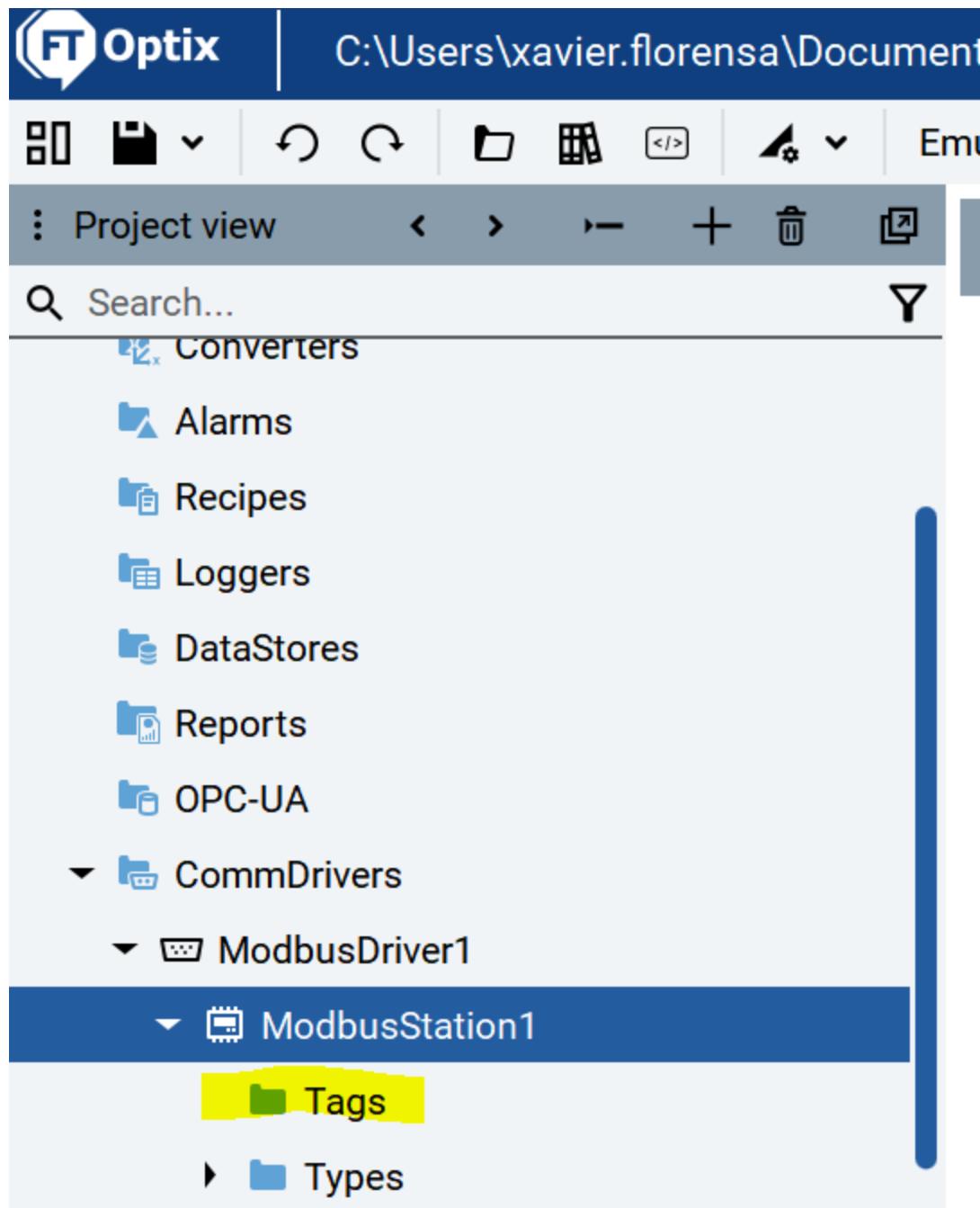
Drag and drop TagsModbus code object to Netlogic on the left



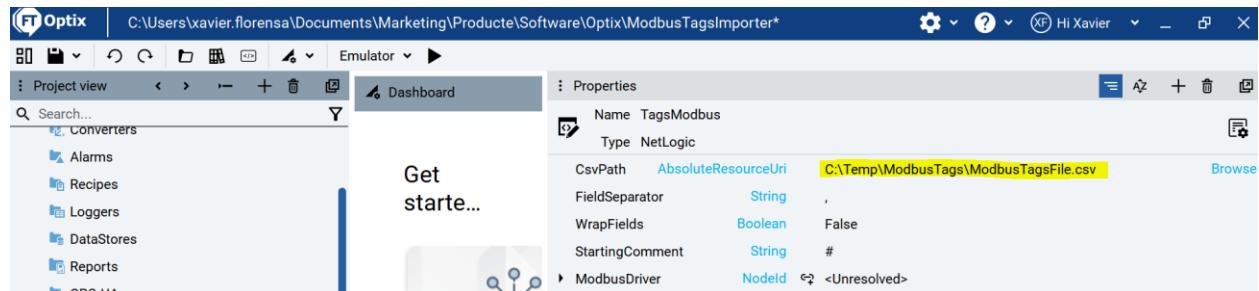
Now let's test it

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



AutoSave Off Search Xavier Florensa Berenguer XF

File Home Insert Page Layout Formulas Data Review View Automate Help PI Builder Comm

Clipboard Font Alignment Number Styles Cells Editing Sensitivity Ad

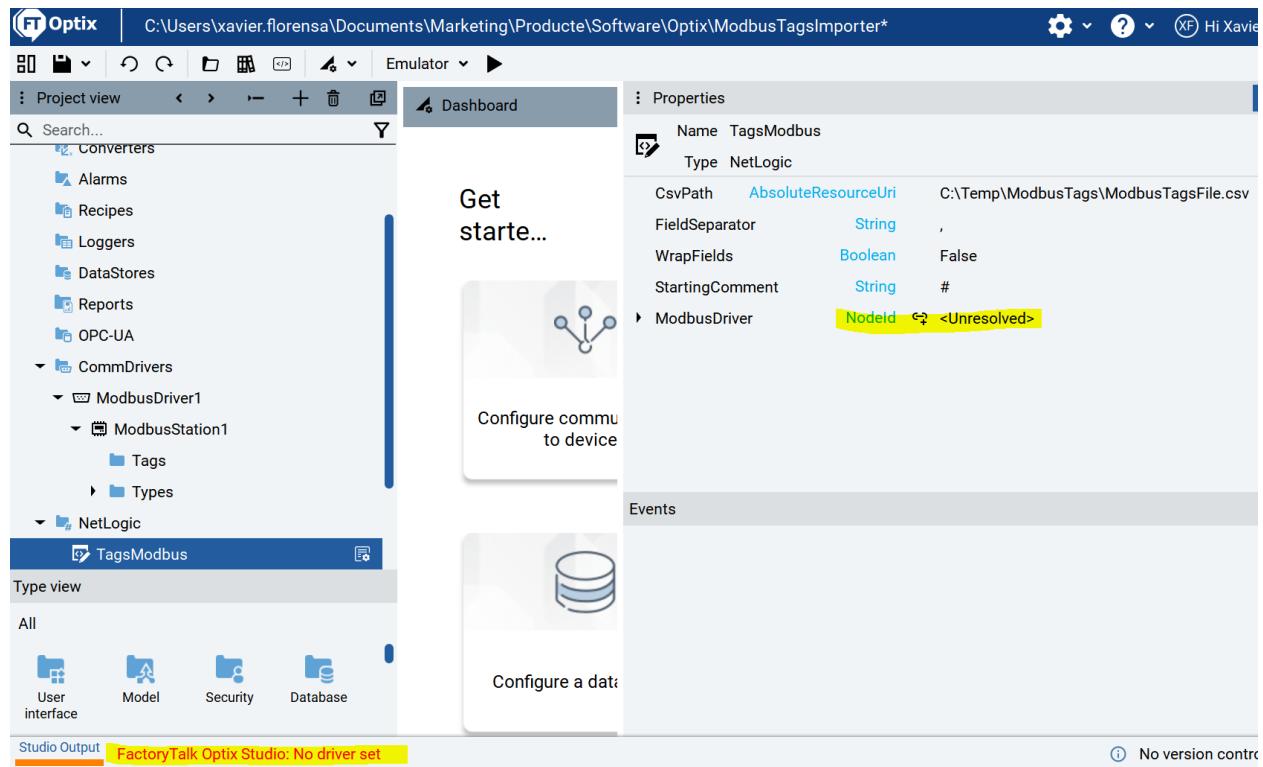
A1 Variable Name IsStructurType Array ElenArray Upd Maximum Memory A Register N BitOffset NumCoil NumDiscreteInput

Variable Name	Type	IsStructurType	Array	ElenArray	Upd	Maximum	Memory	A Register	N	BitOffset	NumCoil	NumDiscreteInput
Temperatu	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	0	0	0	0	0
LTB10610	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	10	0	0	0	0
LTB10620	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	20	0	0	0	0
LTB10630	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	30	0	0	0	0
LTB10640	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	40	0	0	0	0
LTB10650	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	50	0	0	0	0
LTB10660	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	60	0	0	0	0
LTB10670	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	70	0	0	0	0

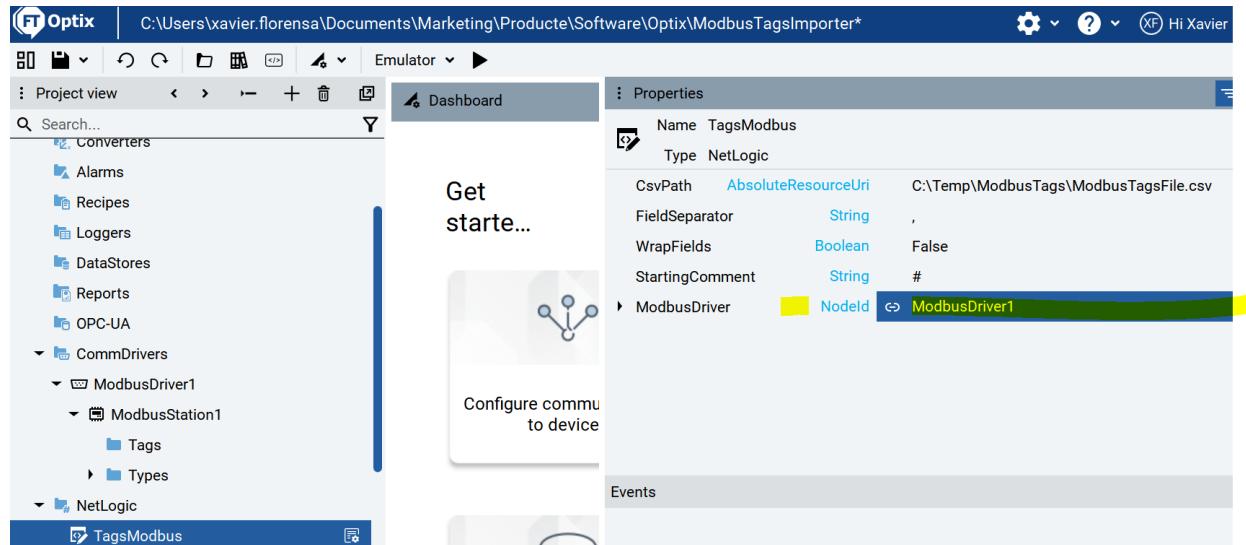
So let's execute the code on design time

The screenshot shows the Optix software interface. The left sidebar displays the Project view with a tree structure of project components: Converters, Alarms, Recipes, Loggers, DataStores, Reports, OPC-UA, CommDrivers (with ModbusDriver1 and ModbusStation1), and NetLogic. The main area features a dashboard with a "Get started..." message, a network icon, and several buttons: "Configure communication", "Execute ExportTags", "Execute ImportTags", "Edit with .NET code editor (external)", "New", and "Copy".

We get an error message



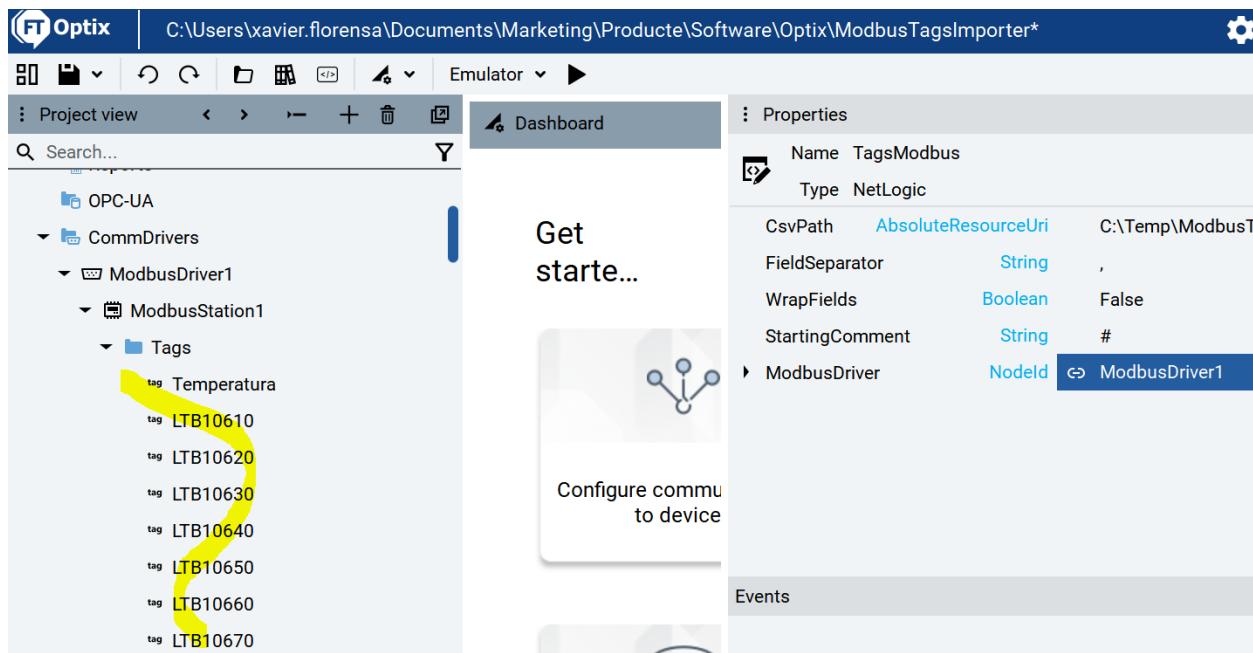
Be aware that the ModbusDriver is Unresolved, we have to solve this problem



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

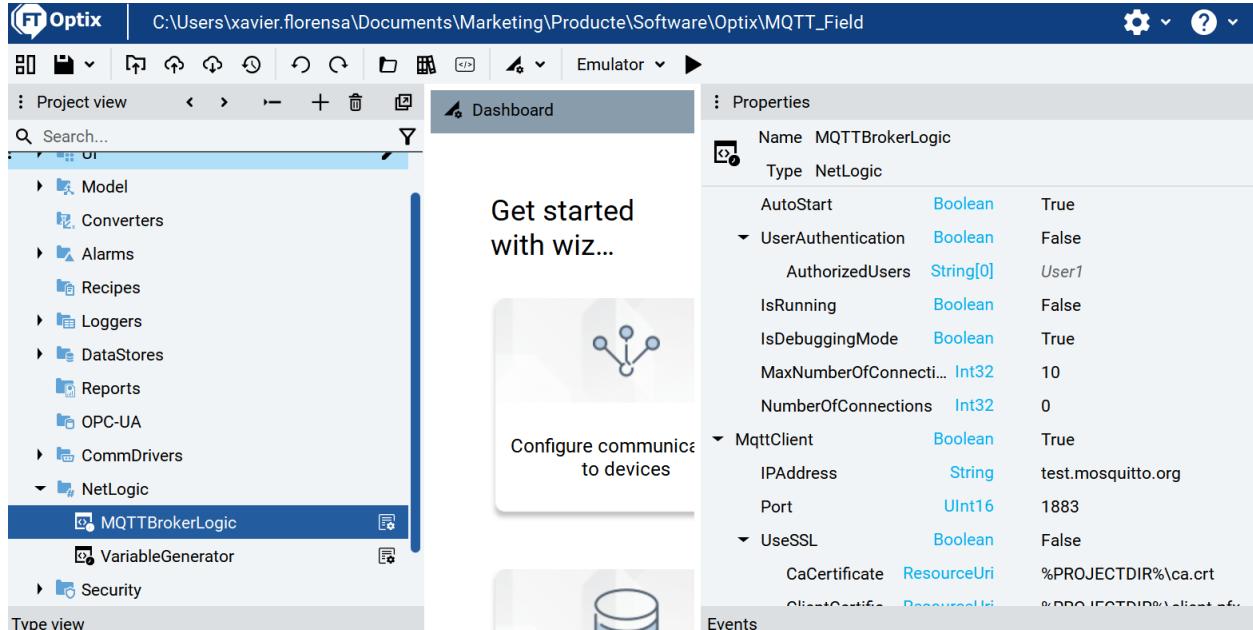


You can even drag and drop the code into Modbus driver instead of into Netlogic

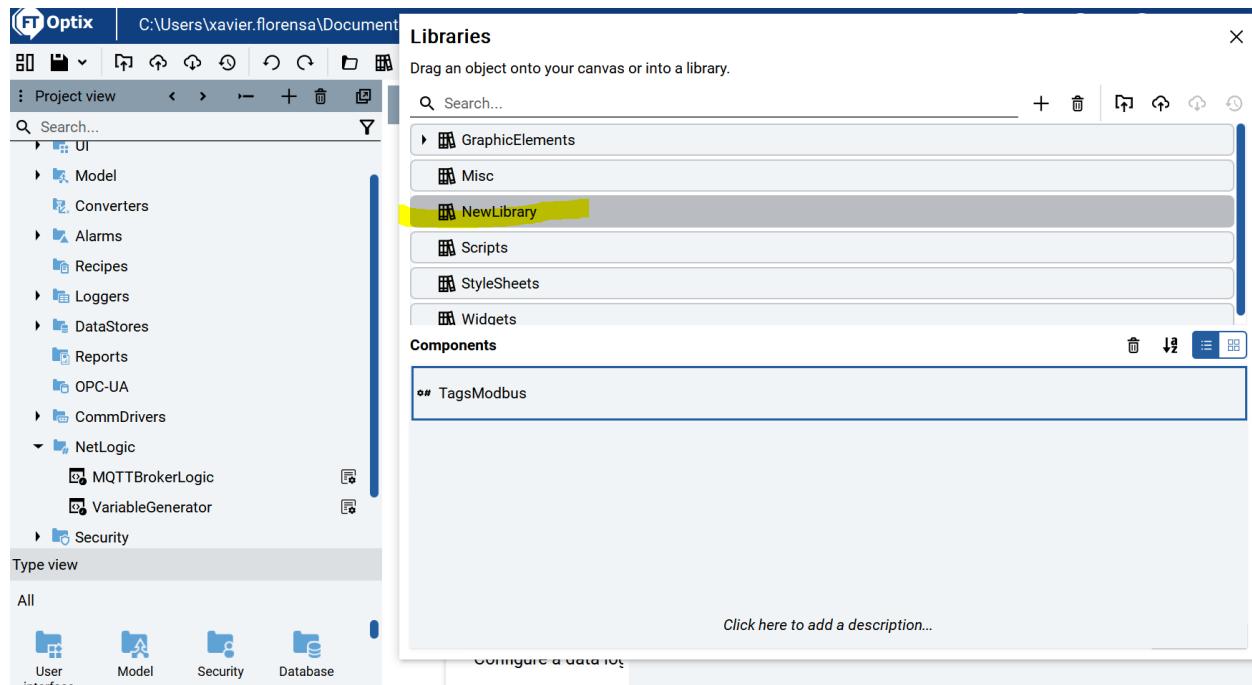
21.2. Runtime libraries

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

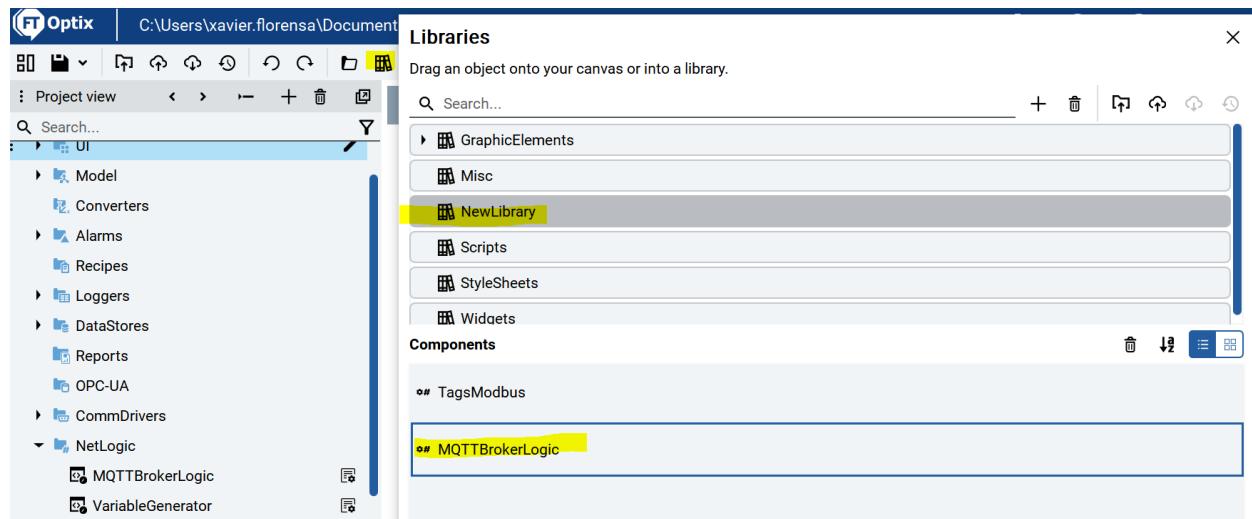
Let's store in our library to be used later on.



Let's create a new library to store our libraries

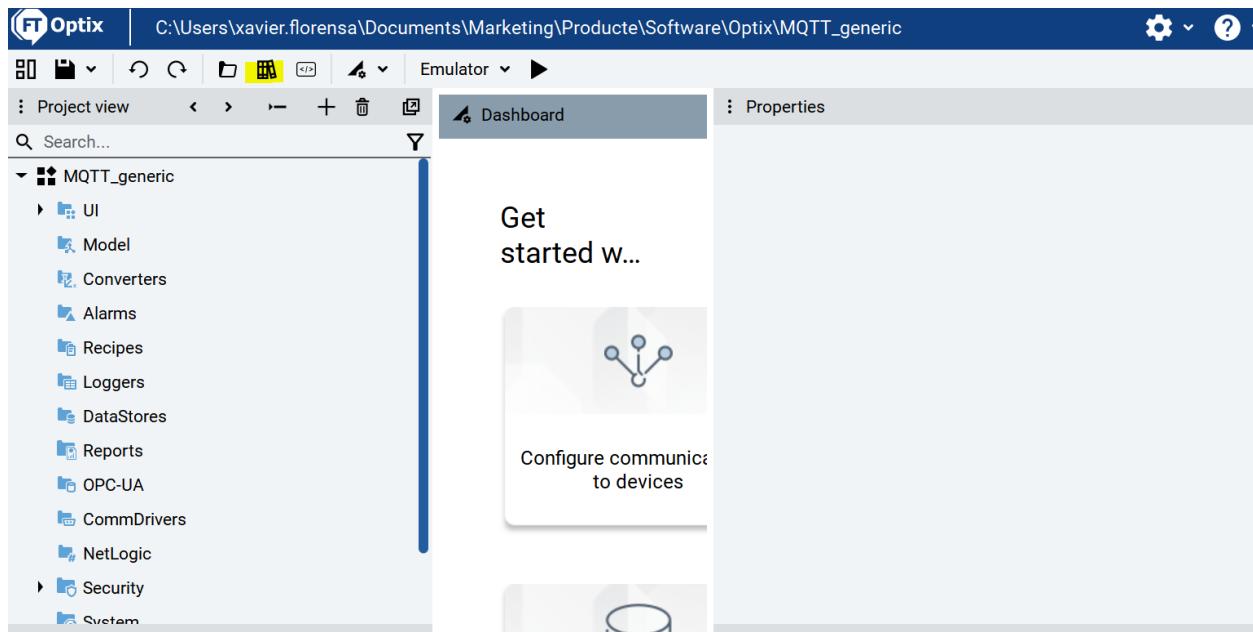


Drag and drop the library

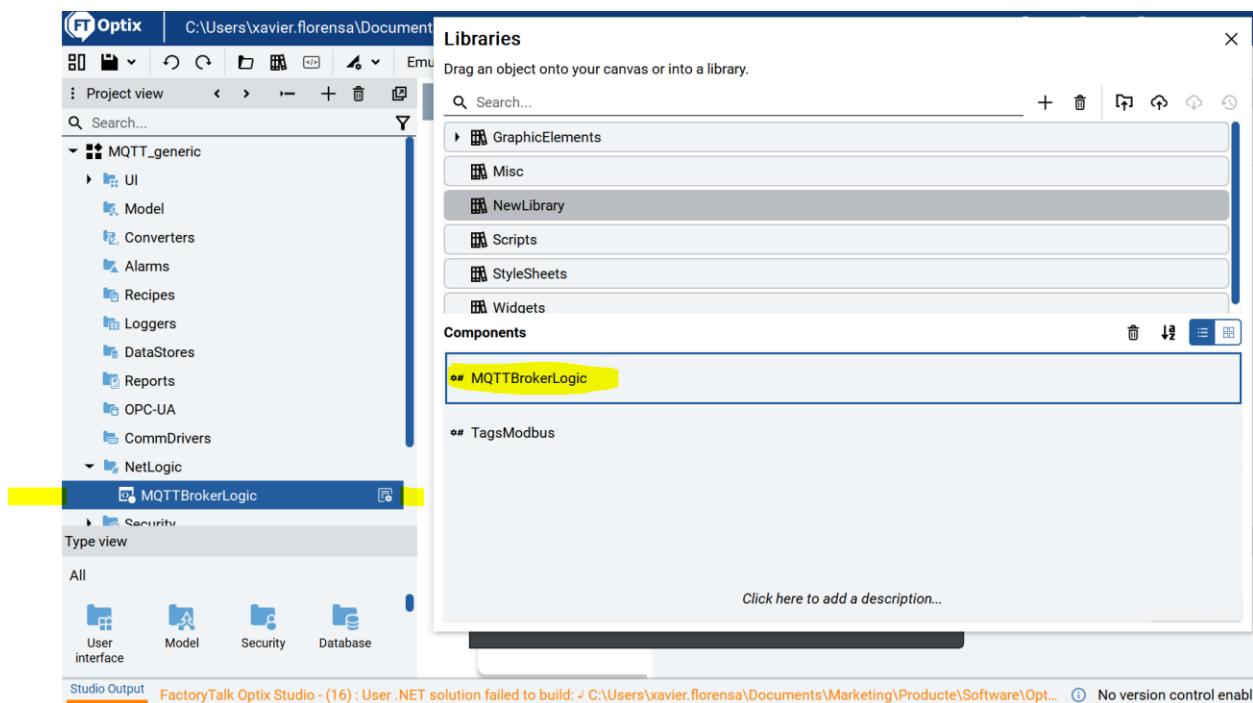


You can close the project and open a new one

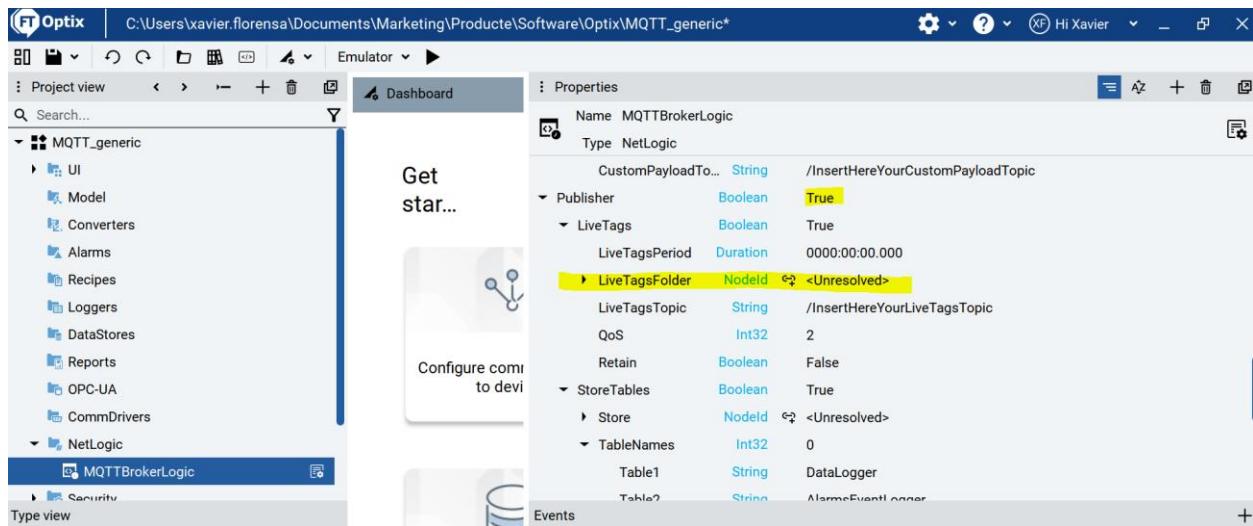
Go to libraries.



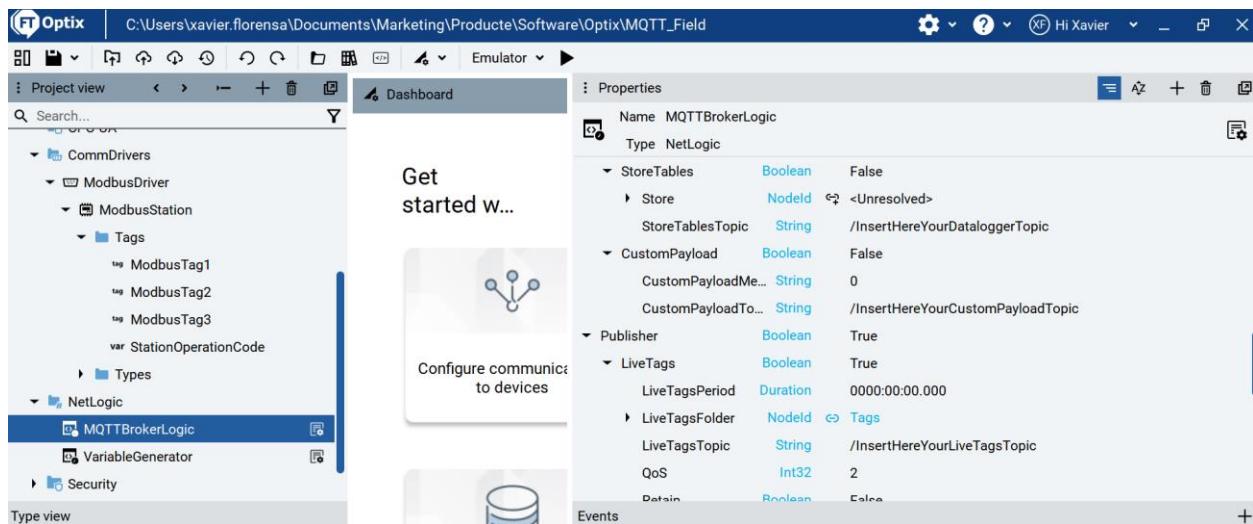
Drag and drop the library to the left over NetLogic



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

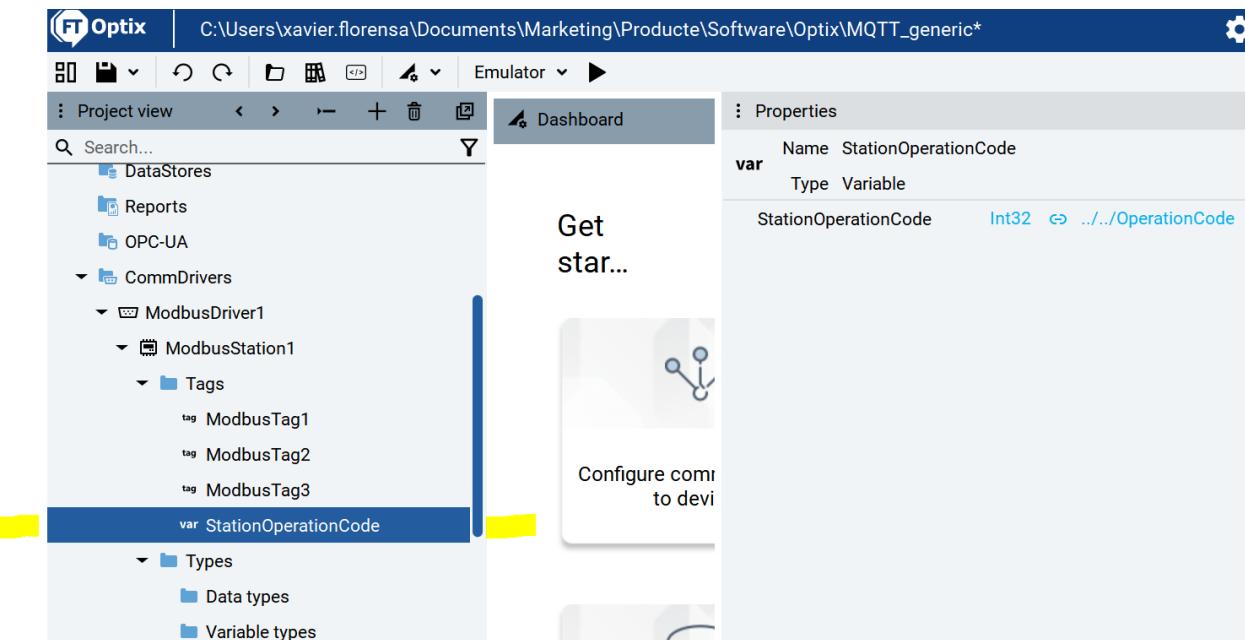


You have to create something like this

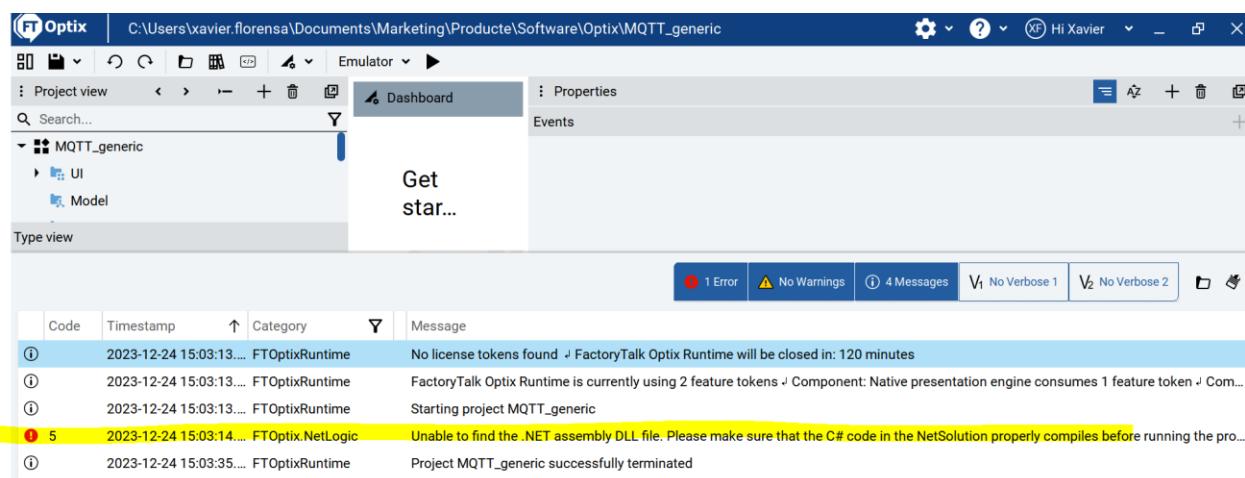


So let's adapt our application to the code accordingly

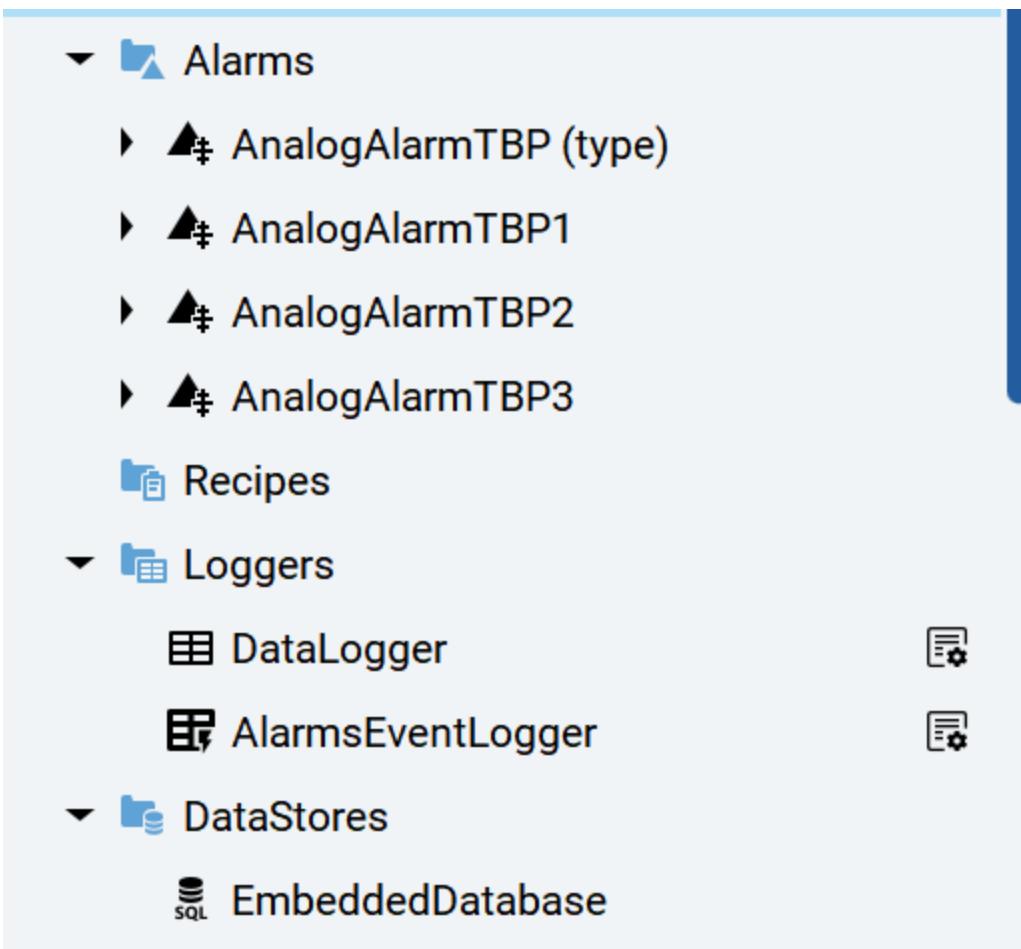
And copy paste the last variable

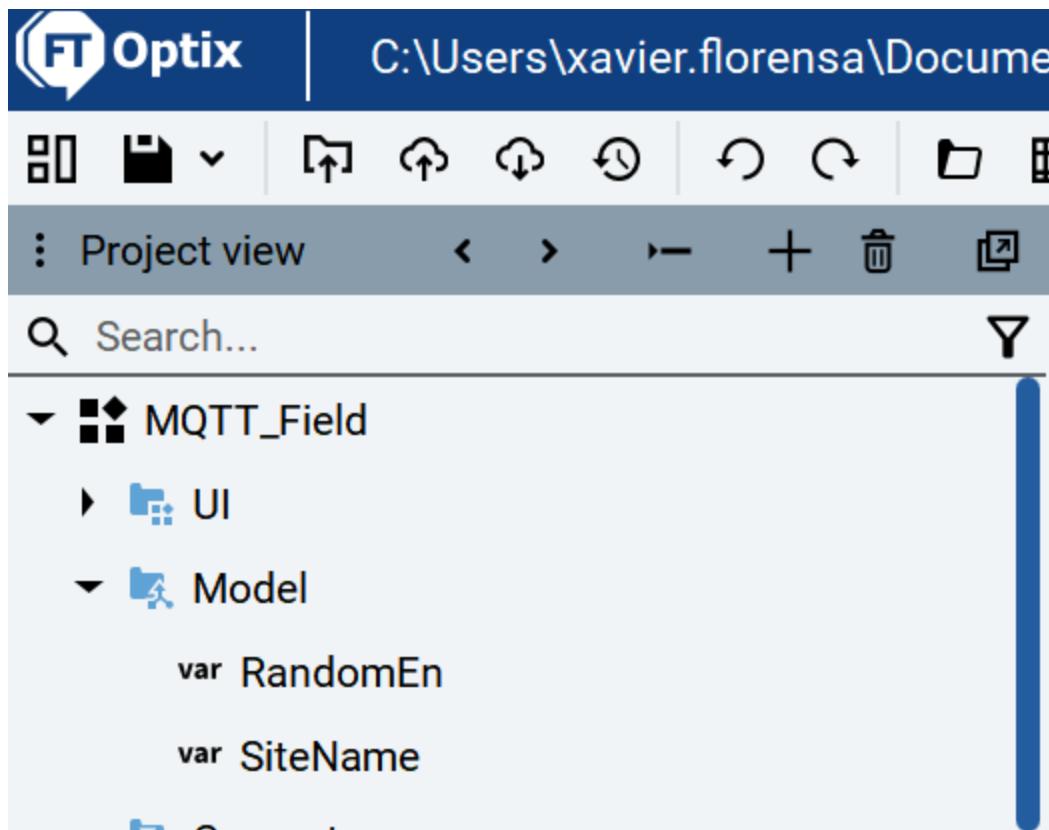


Let's run the code with a Modbus variable simulator



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





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

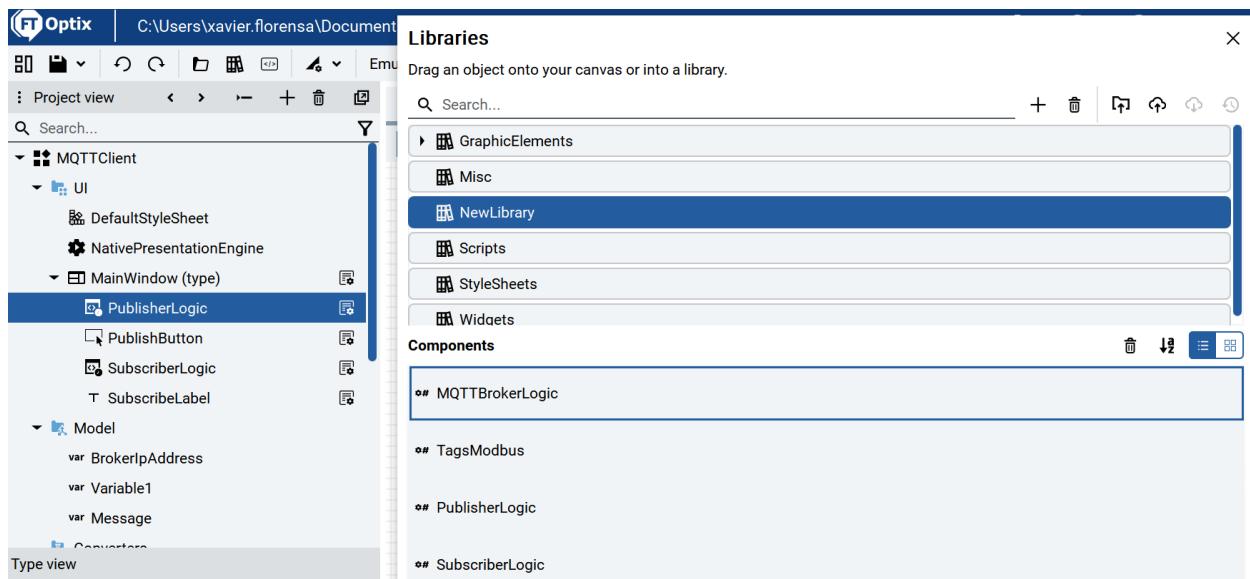
MQTT client you can download from here as we saw on chapter [Configuring an application as an MQTT client](#)

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

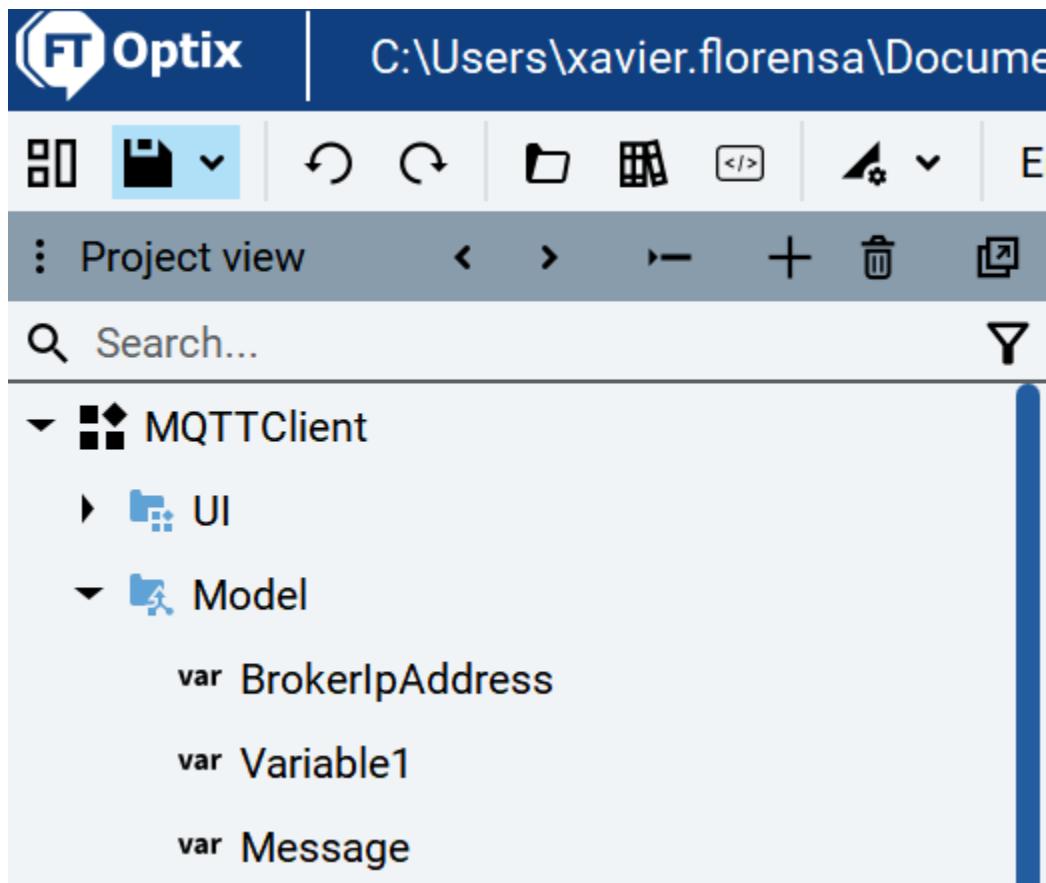


Tip: You can download a sample project from: [MQTTClient.zip](#)

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



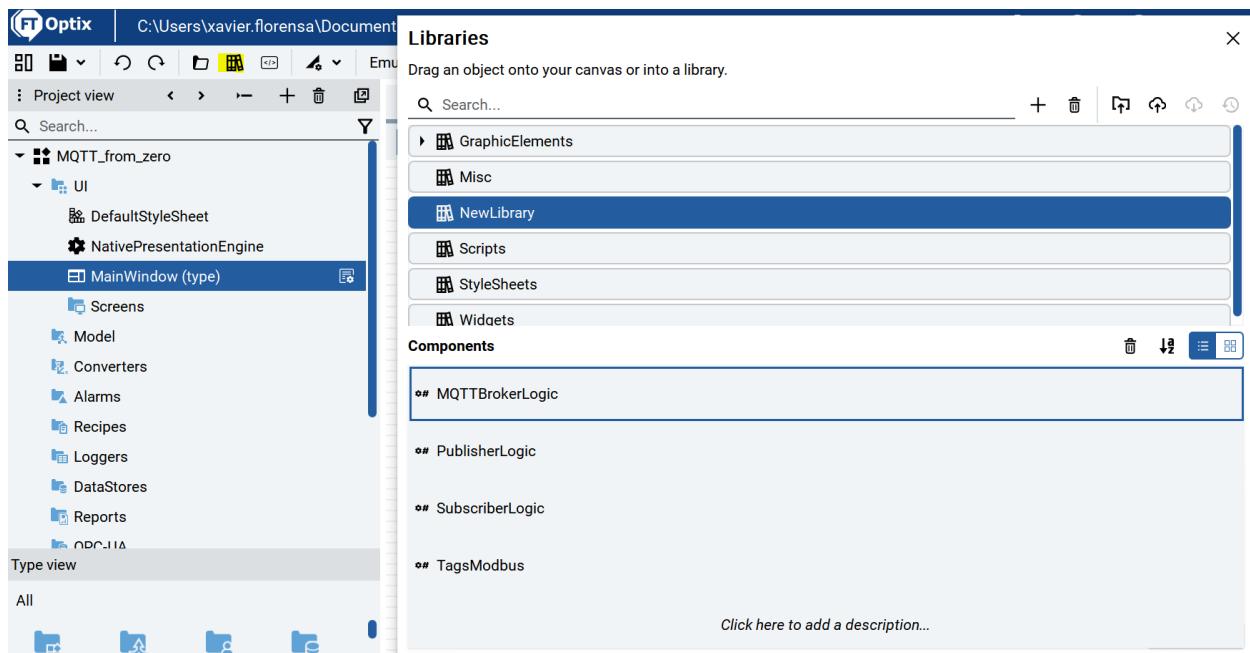
Be aware that this code works with these variables



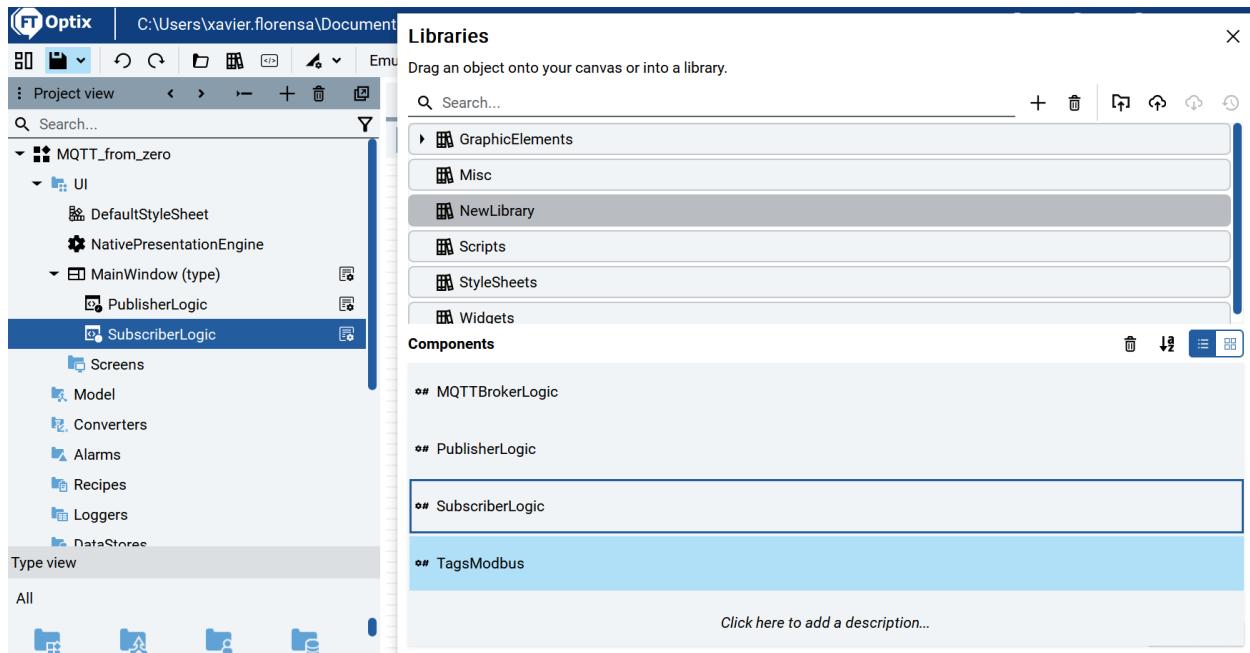
Close the project

Now create a new project and define the variables

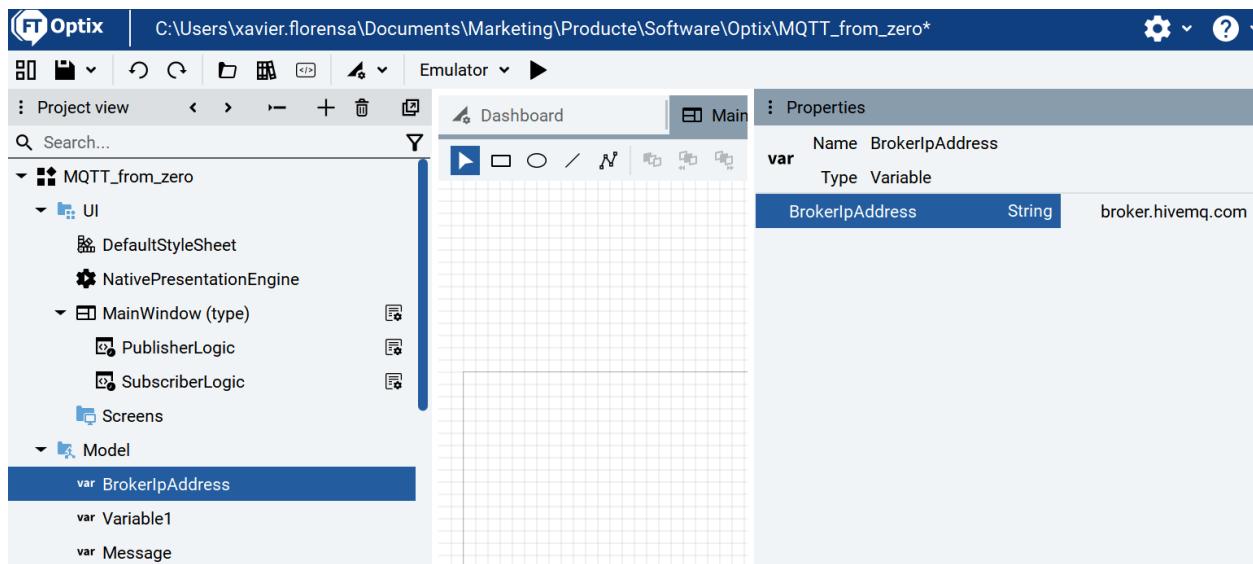
Open libraries



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

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 browser window with the URL <C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/creating-projects/iot/mqtt-client/Configure>. The page title is "Rockwell Automation". On the left, there's a sidebar with a navigation tree containing items like "Project nodes", "Dynamic links", "Converters", etc. The main content area has a tip about downloading a sample project and lists prerequisites (Create a project, Add the NuGet Gallery | MqttNet third-party library, See Third-party .NET libraries...) and steps to develop the project (Configure message broker IP, Develop publisher NetLogic and interface, Develop subscriber NetLogic and interface, Save the project). A note at the bottom says "Select Run to run the project using the client emulator to test your project. To run the project on a remote client, see Add a client device."

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

The screenshot shows the NuGet.org package page for "MQTTnet" version 4.3.3.952. The top navigation bar includes "nuget", "Packages", "Upload", "Statistics", "Documentation", "Downloads", and "Blog". A search bar is present. The package details section shows the logo, version, and download statistics (Total 9.9M, Current version 14.8K, Per day average 4.0K). It also includes sections for "About" (Last updated 16 days ago, Project website, Source repository, MIT license), "Dependencies", "Used By", "Versions", and "Release Notes". A command-line interface (CLI) example is shown: `dotnet add package MQTTnet --version 4.3.3.952`.

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

MQTTnet 4.3.3.952

.NET 5.0 .NET Core 3.1 .NET Standard 1.3 .NET Framework 4.5.2

.NET CLI Package Manager PackageReference Paket CLI Script & Interactive Cake

> dotnet add package MQTTnet --version 4.3.3.952

Downloads Full stats →

Total **9.9M**
Current version **14.8K**
Per day average **4.0K**

About

Last updated 16 days ago
Project website
Source repository
MIT license

[Download package](#) (1.46 MB)
[Download symbols](#) (654.39 KB)
[Open in NuGet Package Explorer](#)
[Open in FuGet Package Explorer](#)
[Report package](#)

[File:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/netlogic/Manage-third-party-dot-net-libraries.html](file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/netlogic/Manage-third-party-dot-net-libraries.html)

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

Third-party .NET libraries

Extend the C# programming language capabilities using third-party .NET libraries.

Important: FactoryTalk Optix Studio supports only .NET libraries with the .NET Standard 6.0 destination framework.

Add a third-party library in the editor of your choice. See [Add a third-party library to the Visual Studio project](#) and [Add a third-party library to the Visual Studio Code project](#).

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

Copy the downloaded file to the directory

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.cs	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.cs	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.cs	12/24/2023 6:04 PM	C# Source File	2 KB
SubscriberLogic.cs	12/24/2023 6:16 PM	C# Source File	2 KB

Installing Nuget packages in Visual Studio Code

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)

MQTT_from_zero (Workspace)

EXPLORER

NetSolution

MQTT_from_zero.csproj

MQTT_from_zero.references

MQTT_from_zero.sln

mqttnet4.3.3.952.nupkg

PublisherLogic.cs

SubscriberLogic.cs

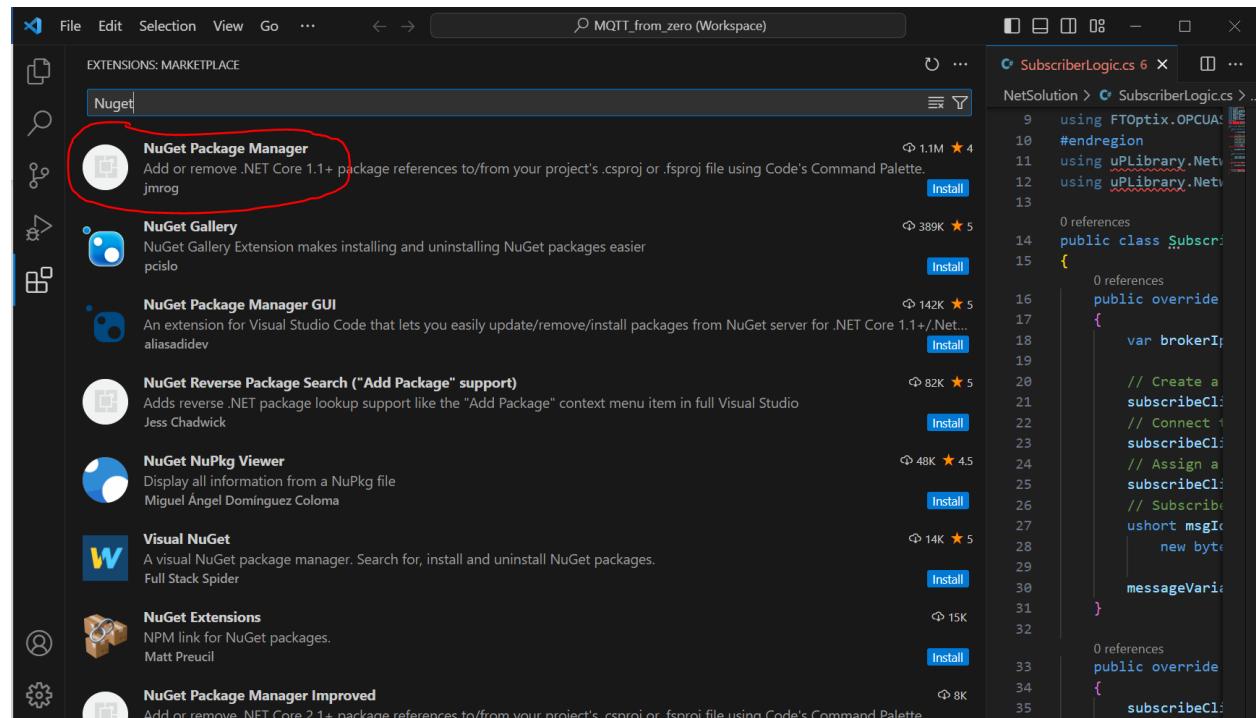
SubscriberLogic.cs

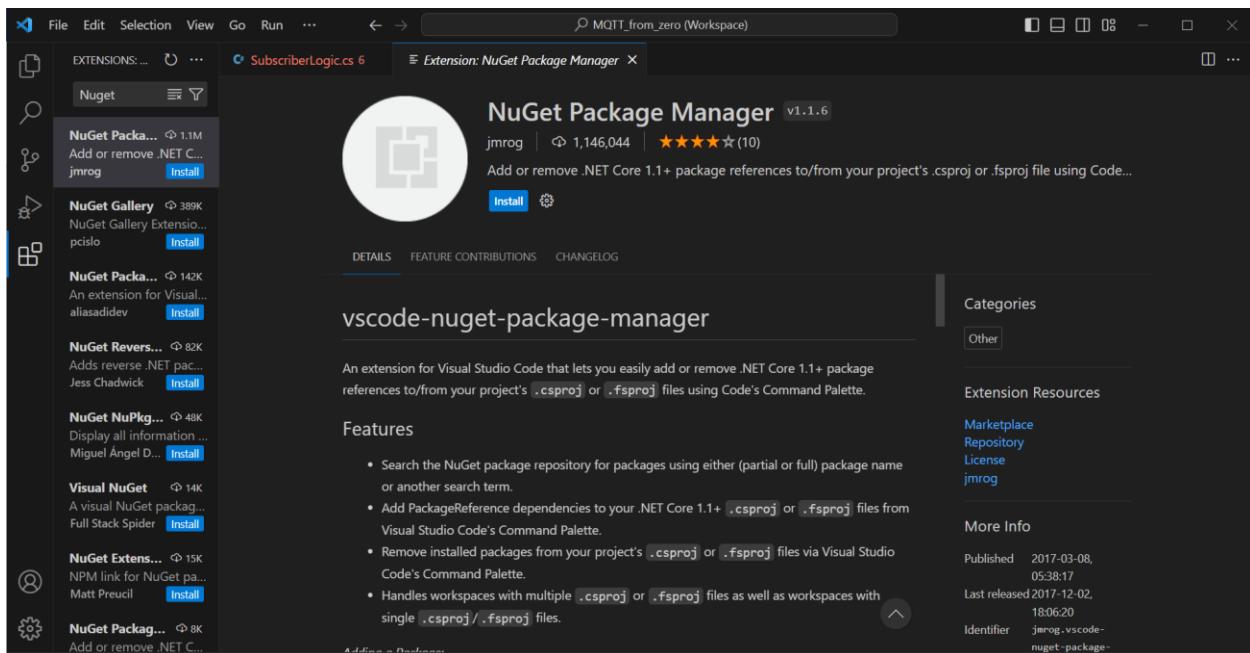
```
using FTOptix.OPCUAServer;
#endregion
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;

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;
        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" });
    }
}
```

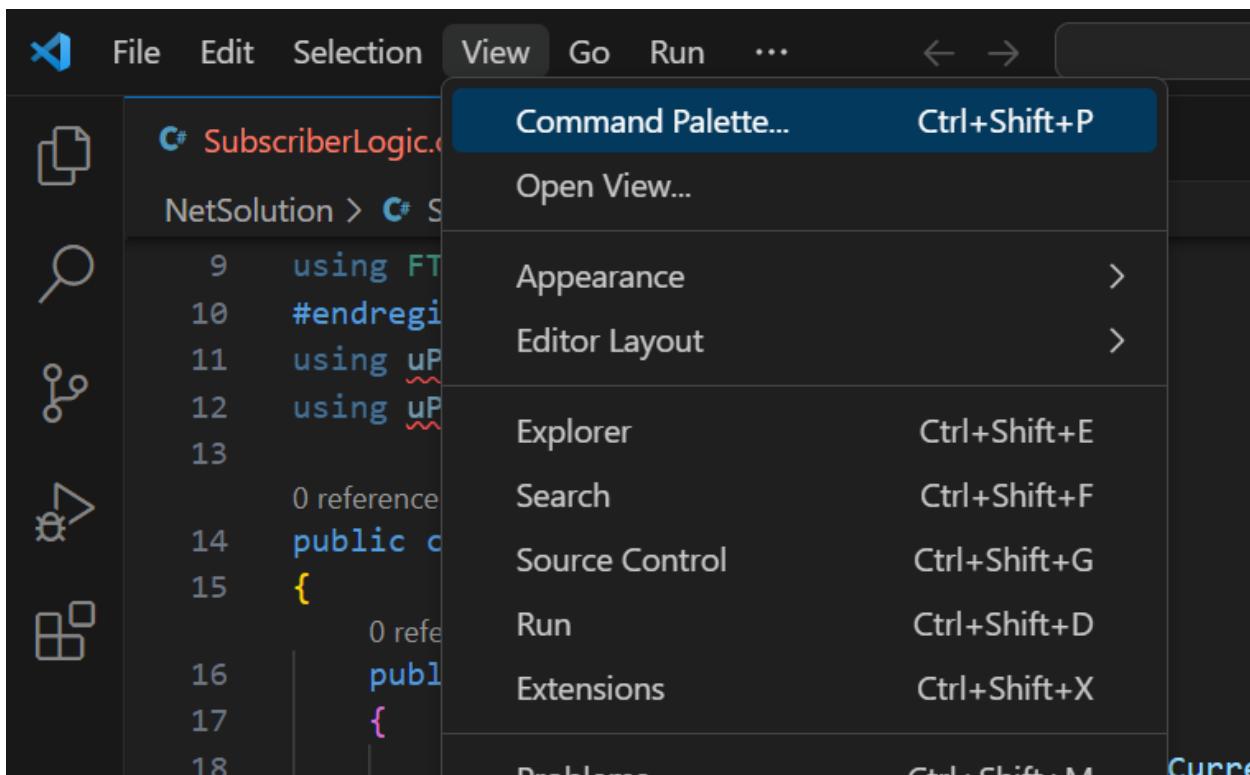
And search for NuGet Package Manager



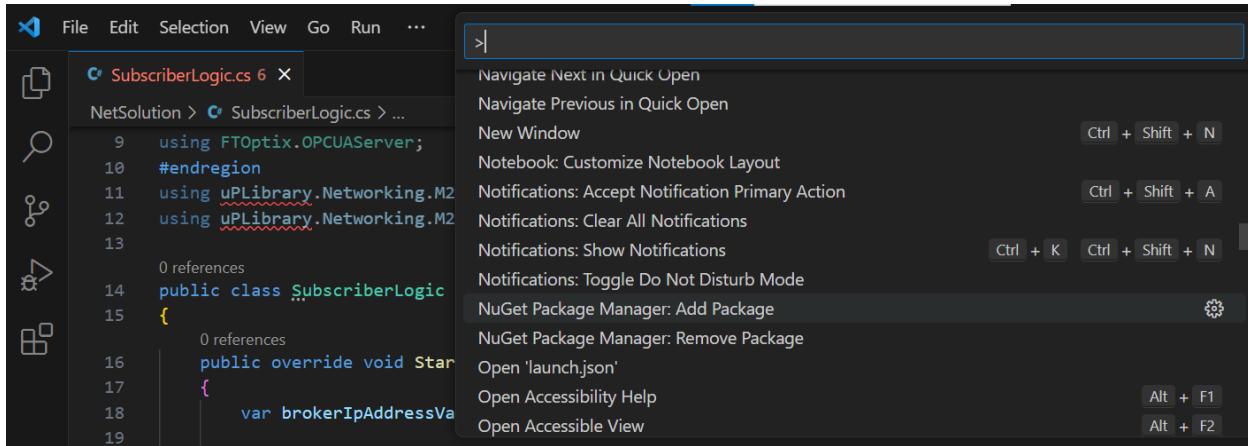


Now close clicking again on the Extensions lowest left icon,

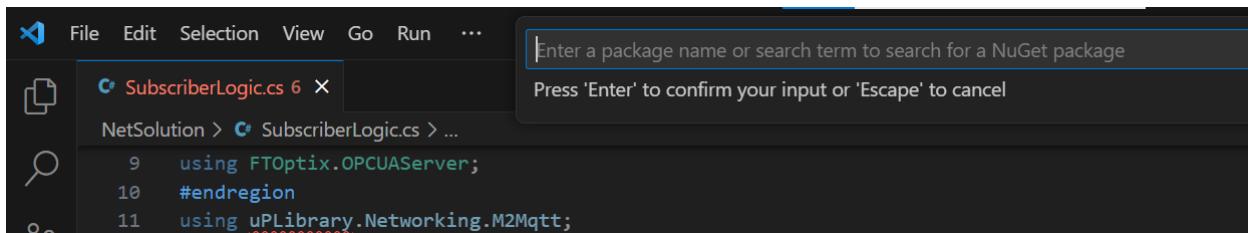
And open command pallete



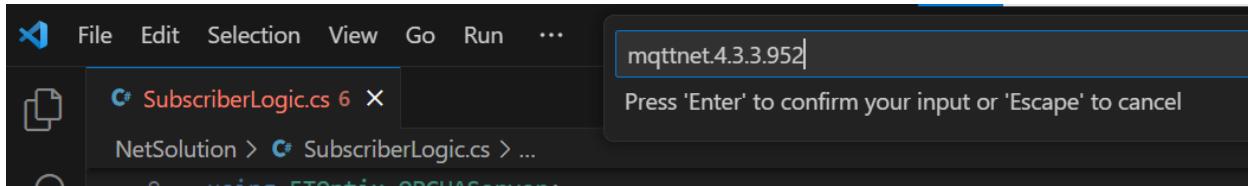
And select the Option Nuget Package manager. Add package



Enter a package name

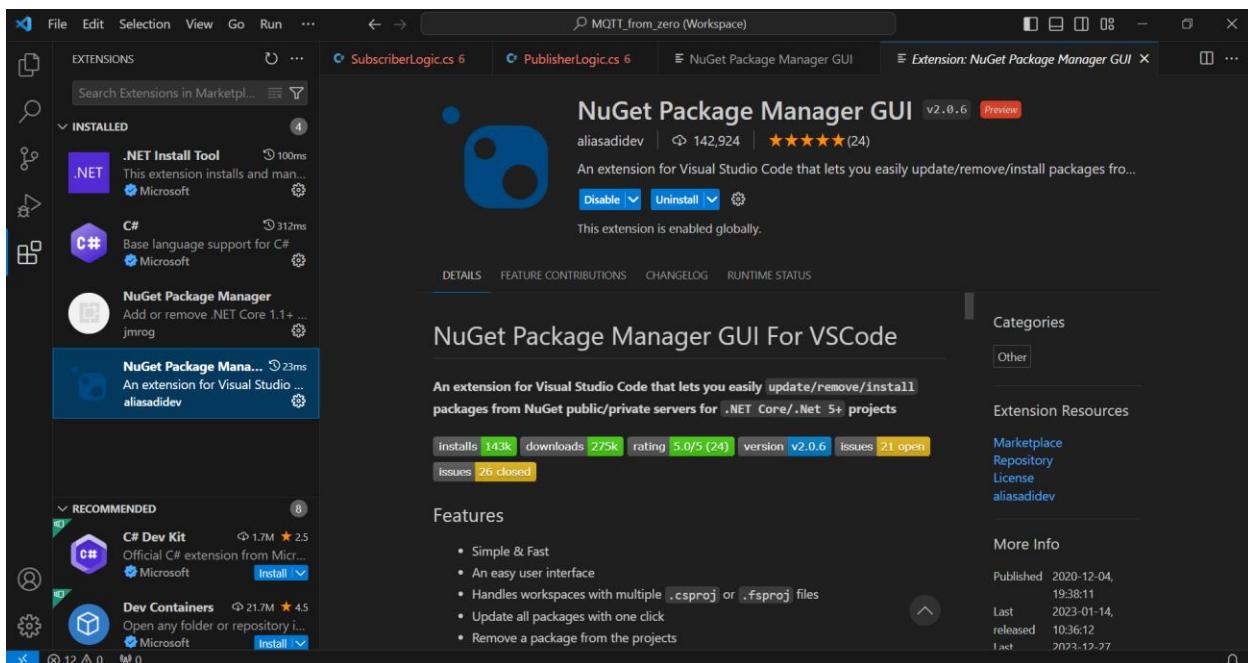


Type mqttnet.4.3.3.952 without the extension



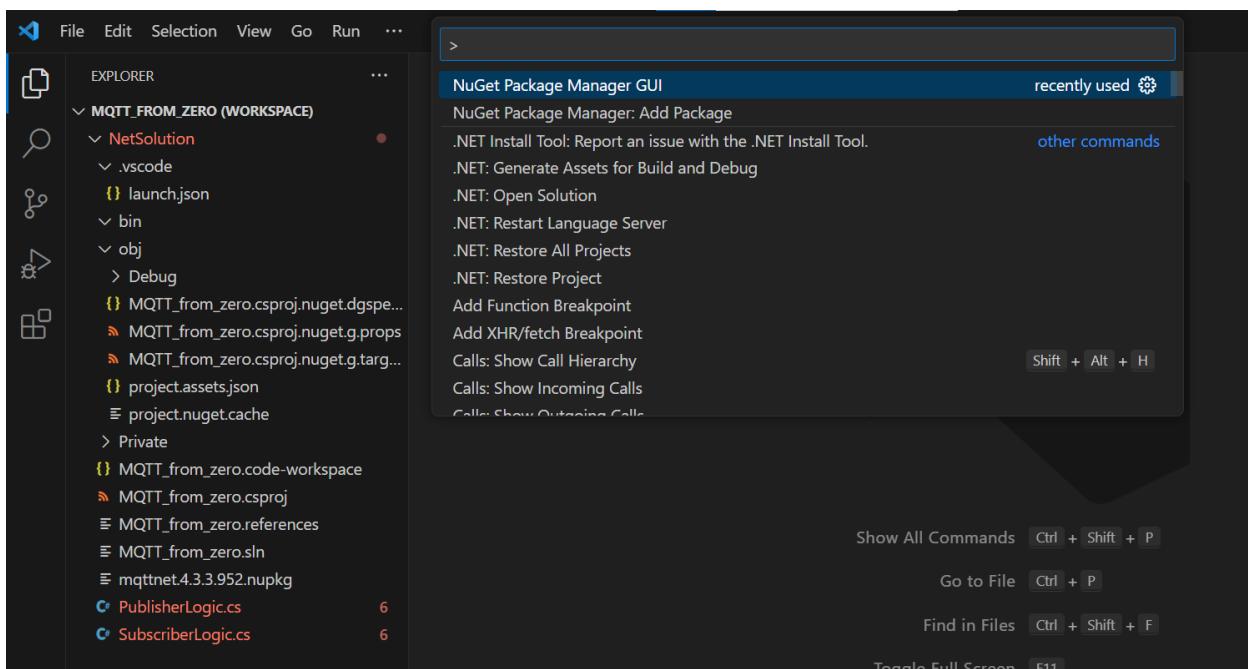
This is not working, the system can not find the file.

Let's try to use this Nuget Manager GUI

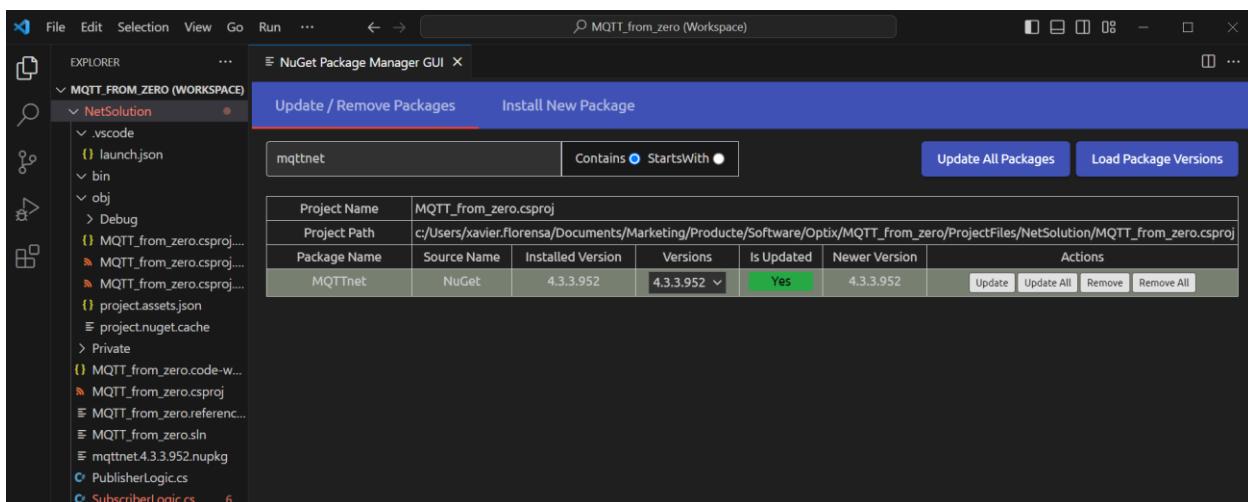
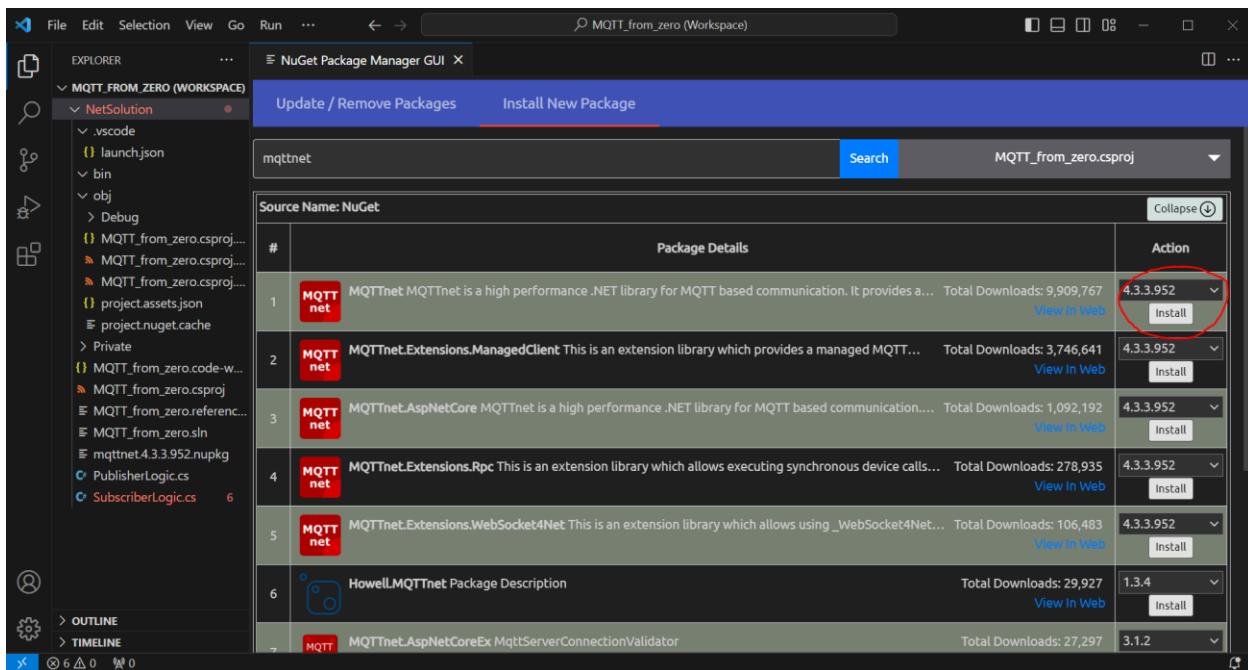


Install such package manager

Go to View / Command Pallete



Search for the library on the server and click on install



If we take a look at *.csproj

```

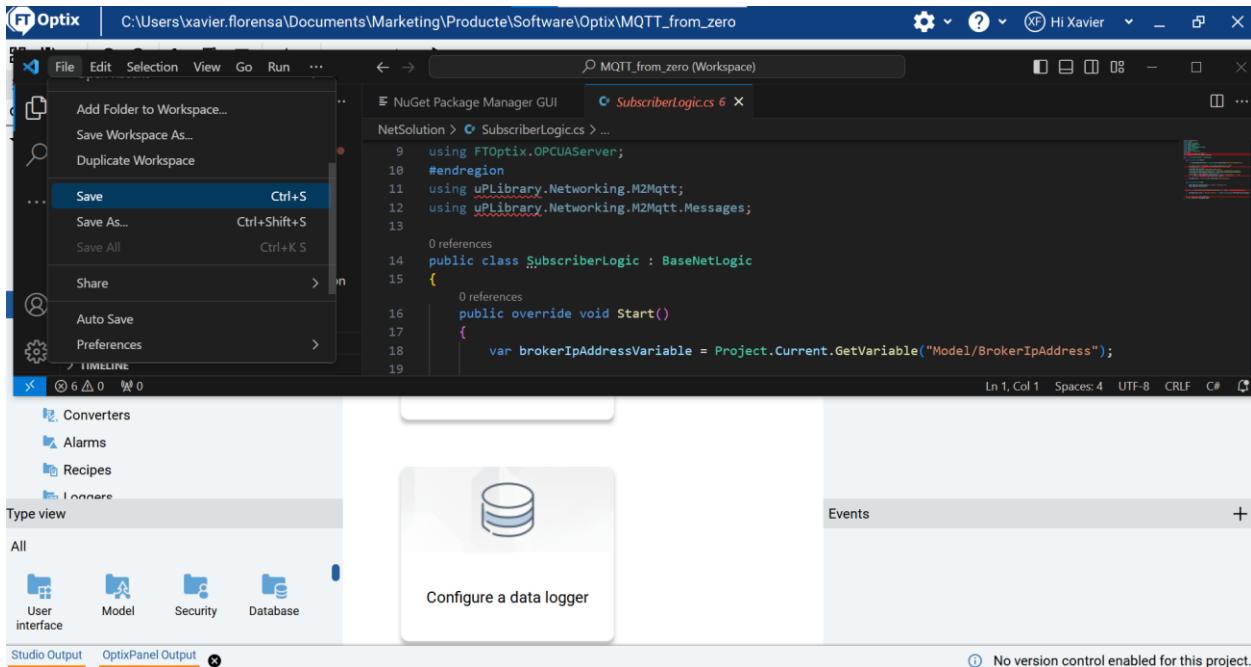
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <AppendTargetFrameworkToOutputPath>false</AppendTargetFrameworkToOutputPath>
    <CopyLocalLockFileAssemblies>true</CopyLocalLockFileAssemblies>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)'=='Debug|AnyCPU'">
    <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>

```

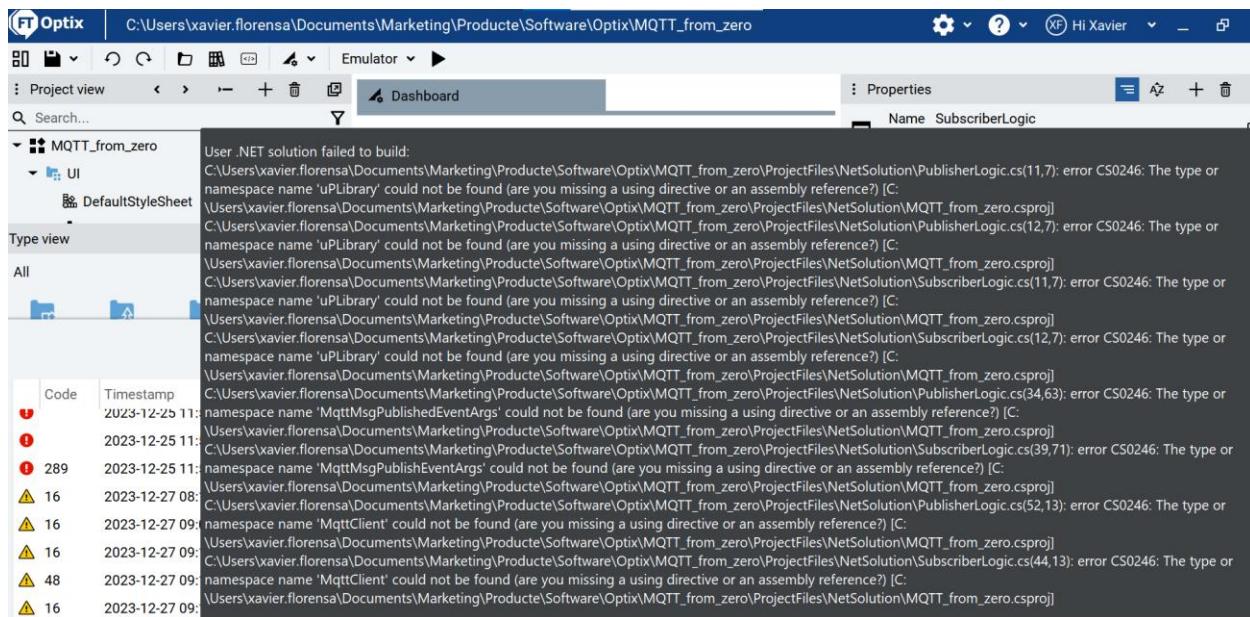
We have it installed as it was recommending on the web help

<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



2023-12-27 11:00:43.398;NetHelper;222;Warning;16;User .NET solution failed to build:

C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\PublisherLogic.cs(11,7): error CS0246: The type or namespace name 'uPLibrary' could not be found (are you missing a using directive or an assembly reference?)

[C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\MQTT_from_zero.csproj]

C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\PublisherLogic.cs(12,7): error CS0246: The type or namespace name 'uPLibrary' could not be found (are you missing a using directive or an assembly reference?)

[C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\MQTT_from_zero.csproj]

C:\Users\xavier.florensa\Documents\Marketing\Producte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\SubscriberLogic.cs(11,7): error CS0246: The type or namespace name 'uPLibrary' could not be found (are you missing a using directive or an assembly reference?)

[C:\Users\xavier.florensa\Documents\Marketing\Producet\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\MQTT_from_zero.csproj]

C:\Users\xavier.florensa\Documents\Marketing\Producte\Software\Optix\MQTT_from_zero\ProjectFile
s\NetSolution\SubscriberLogic.cs(12,7): error CS0246: The type or namespace name 'uPLibrary' could
not be found (are you missing a using directive or an assembly reference?)

[C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\MQTT_from_zero.csproj]

C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\PublisherLogic.cs(34,63): error CS0246: The type or namespace name

'MqttMsgPublishedEventArgs' could not be found (are you missing a using directive or an assembly reference?)

[C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\MQTT_from_zero.csproj]

C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\SubscriberLogic.cs(39,71): error CS0246: The type or namespace name 'MqttMsgPublishEventArgs' could not be found (are you missing a using directive or an assembly reference?)

[C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\MQTT_from_zero.csproj]

C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\PublisherLogic.cs(52,13): error CS0246: The type or namespace name 'MqttClient' could not be found (are you missing a using directive or an assembly reference?)

[C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\MQTT_from_zero.csproj]

C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\SubscriberLogic.cs(44,13): error CS0246: The type or namespace name 'MqttClient' could not be found (are you missing a using directive or an assembly reference?)

[C:\Users\xavier.florensa\Documents\Marketing\Produkte\Software\Optix\MQTT_from_zero\ProjectFiles\NetSolution\MQTT_from_zero.csproj]

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

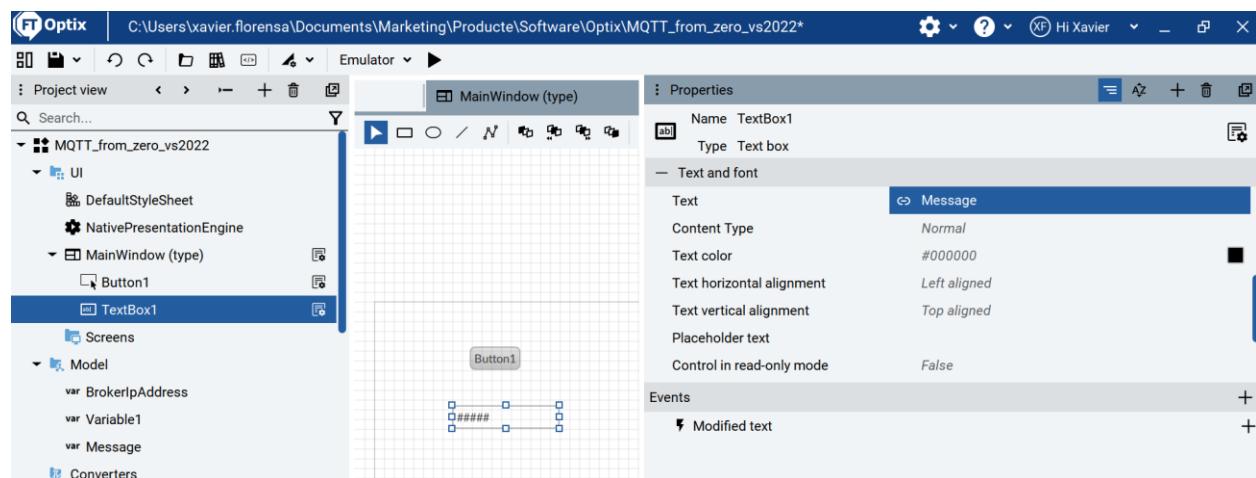
Then you will have no errors

But this is still not working.

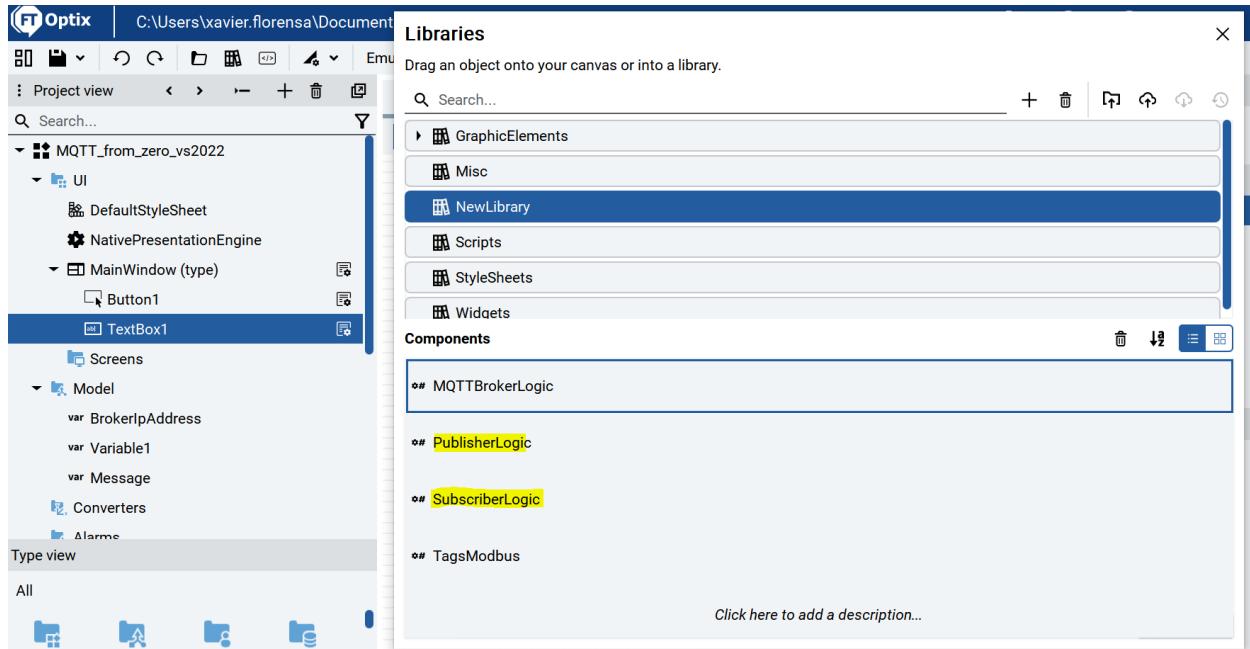
Let's try to install the two Nuget packages with Visual Studio 2022, and then once installed, let's move to Visual Studio Code.

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

Create the dynamic link between textbox and Message

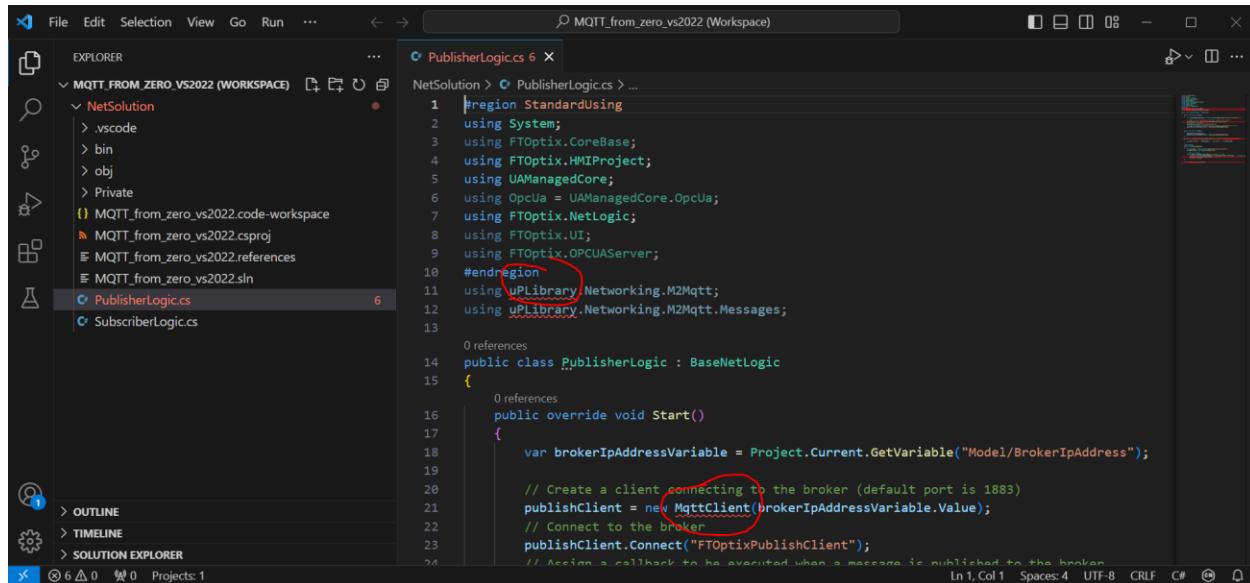


Drag and drop to MainWindow the two libraries that you stored in previous chapters. PublisherLogic and SubscriberLogic



Open PublisherLogic

Note that the compiler complains



Close VS Code

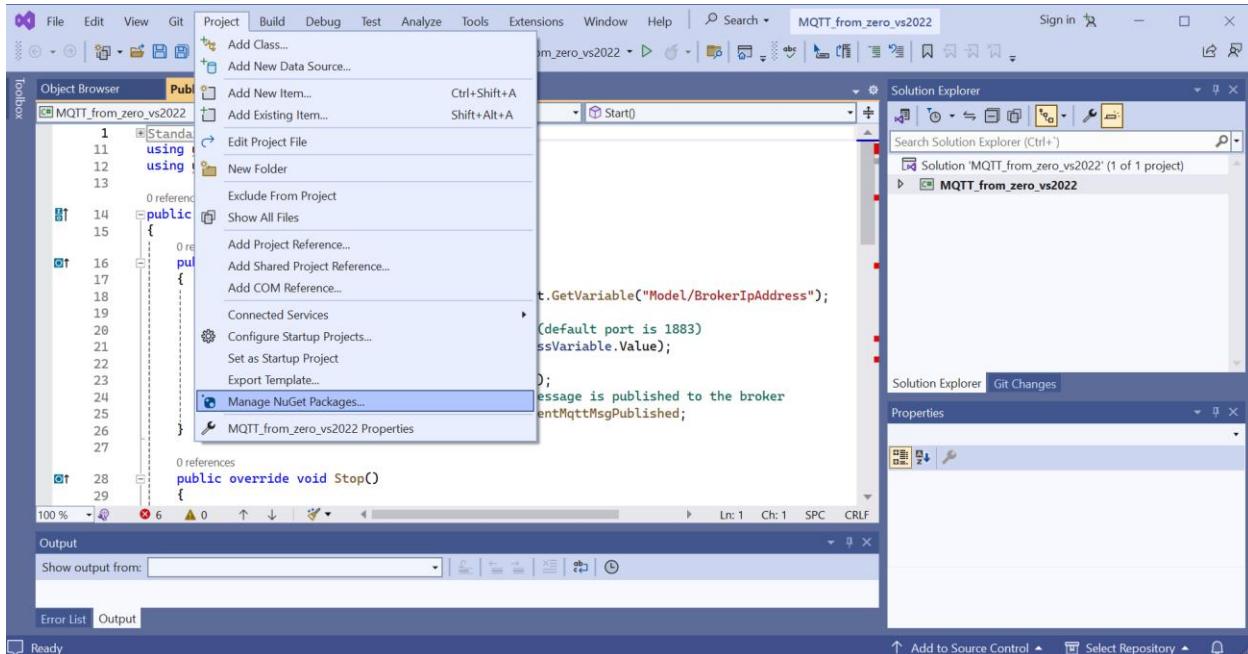
Now change the editor to Visual Studio 2022, save close and open FTOptix again to admit changes.

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

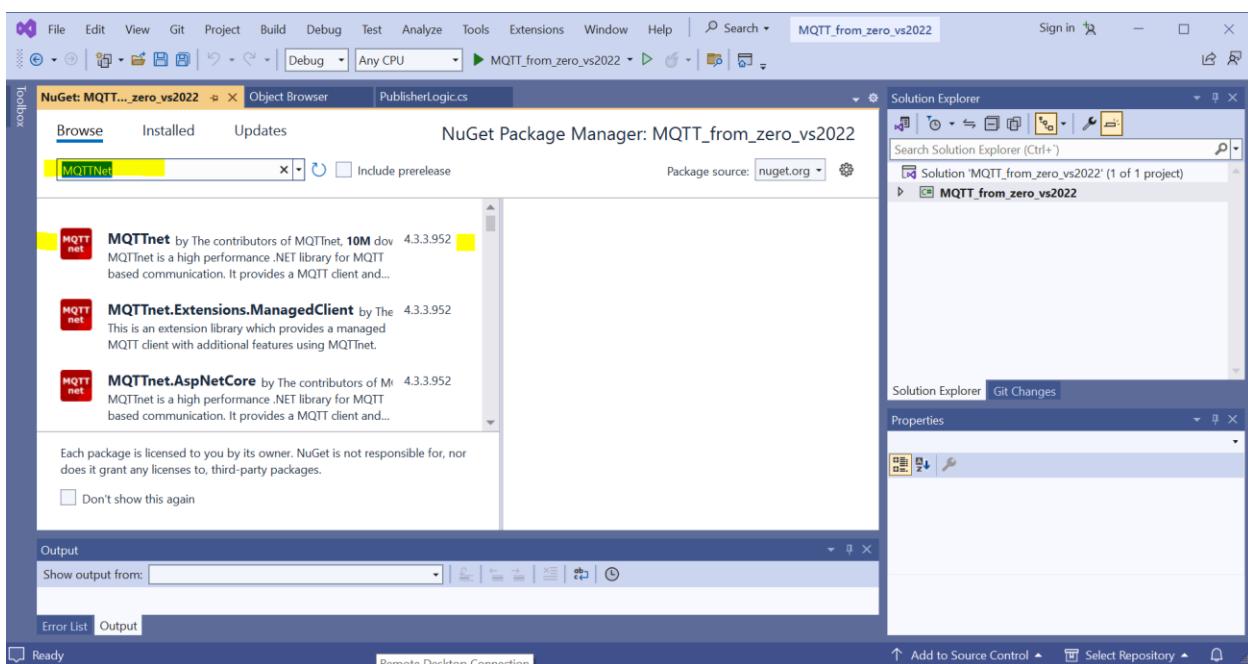
> This PC > Documents > Marketing > Product > Software > Optix > MQTT_from_zero_vs2022 > ProjectFiles > NetSolution > bin

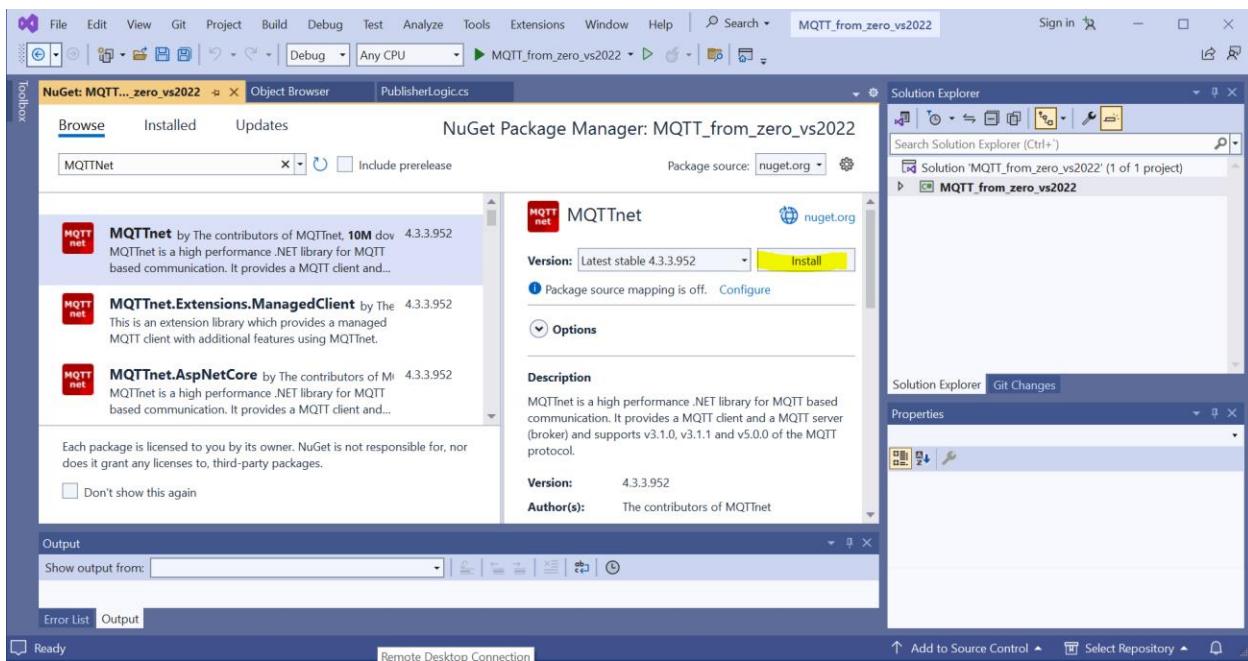
Name	Date modified	Type	Size
This folder is empty.			

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

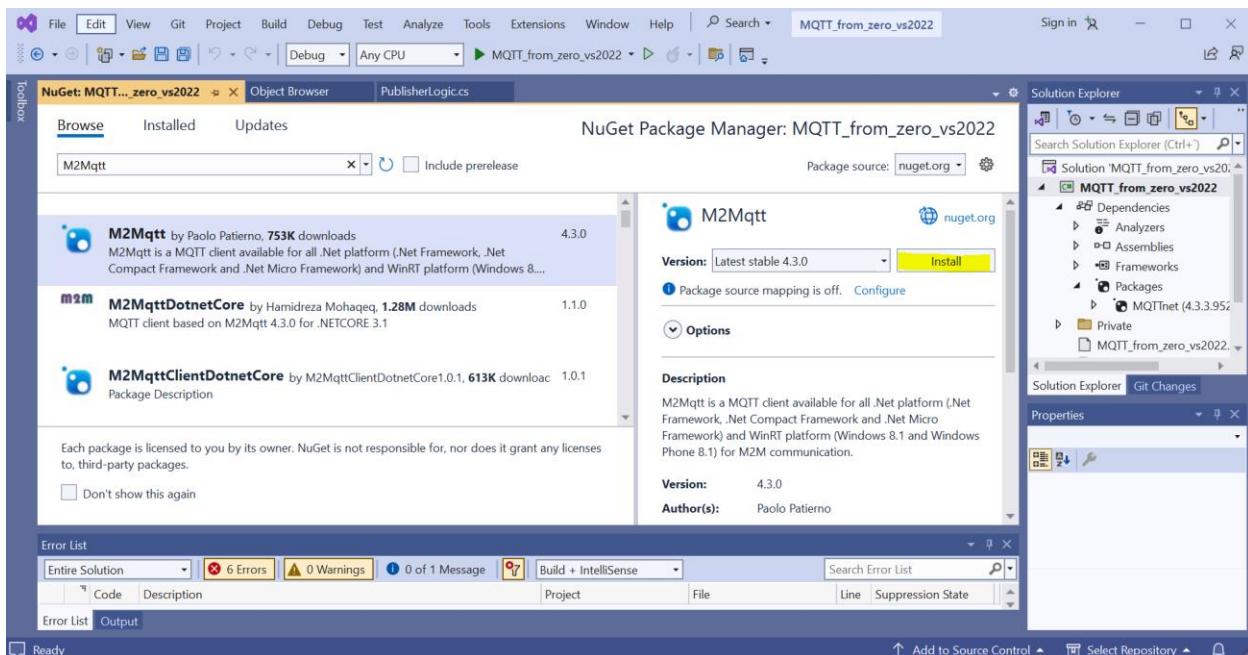


Select and install this library





Install also this library M2Mqtt



Now we have the two packages

The screenshot shows the Microsoft Visual Studio 2022 interface. The code editor displays `PublisherLogic.cs` with the following code:

```

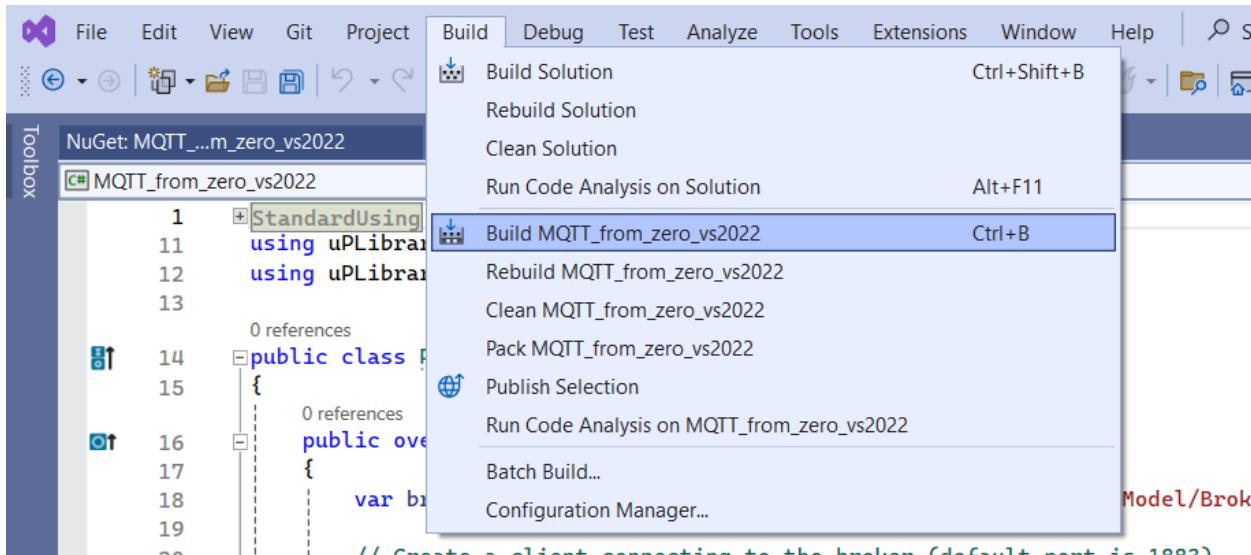
1  StandardUsing;
11  using uPLibrary.Networking.M2Mqtt;
12  using uPLibrary.Networking.M2Mqtt.Messages;
13
14  public class PublisherLogic : BaseNetLogic
15  {
16      public override void Start()
17      {
18          var brokerIpAddressVariable = Project.Current.GetVariable("Model/BrokerIpAddress");
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

```

The Solution Explorer pane shows the project `MQTT_from_zero_vs2022` with its dependencies, including `M2Mqtt (4.3.0)` and `MQTTnet (4.3.952)`. The Error List pane shows 0 errors and 1 warning.

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

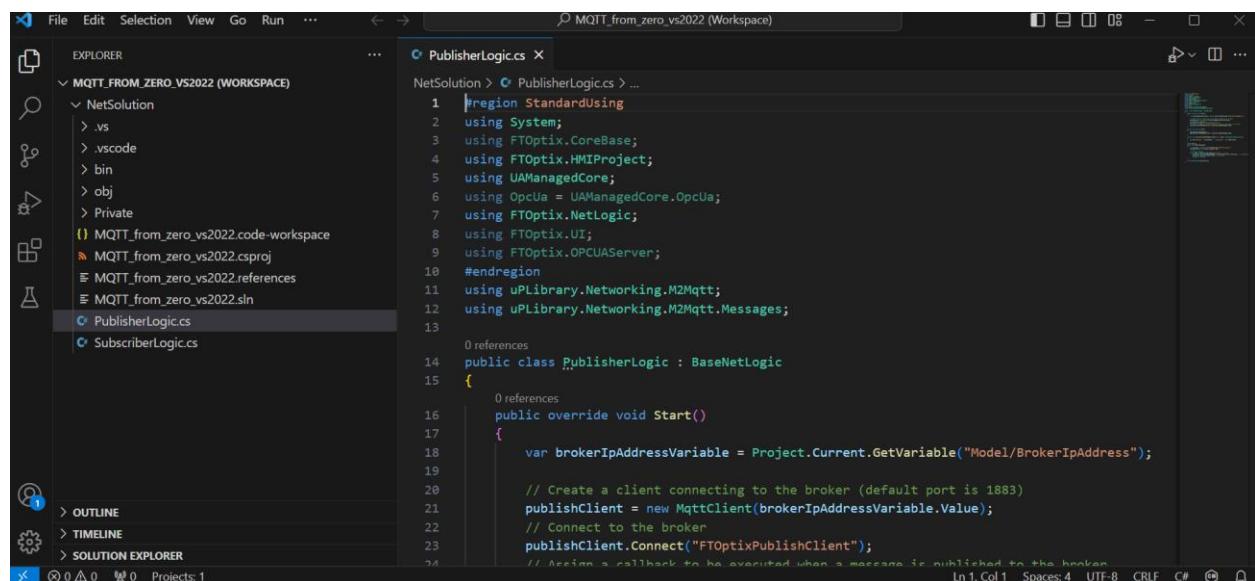
> This PC > Documents > Marketing > Product > Software > Optix > MQTT_from_zero_vs2022 > ProjectFiles > NetSolution > bin

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

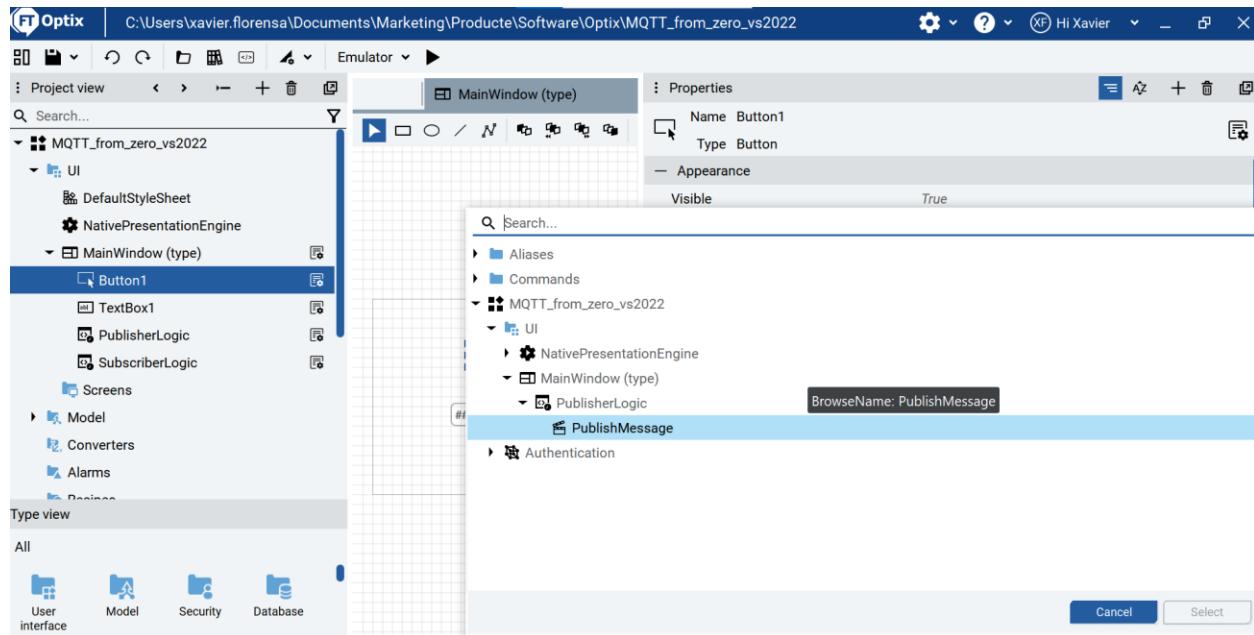
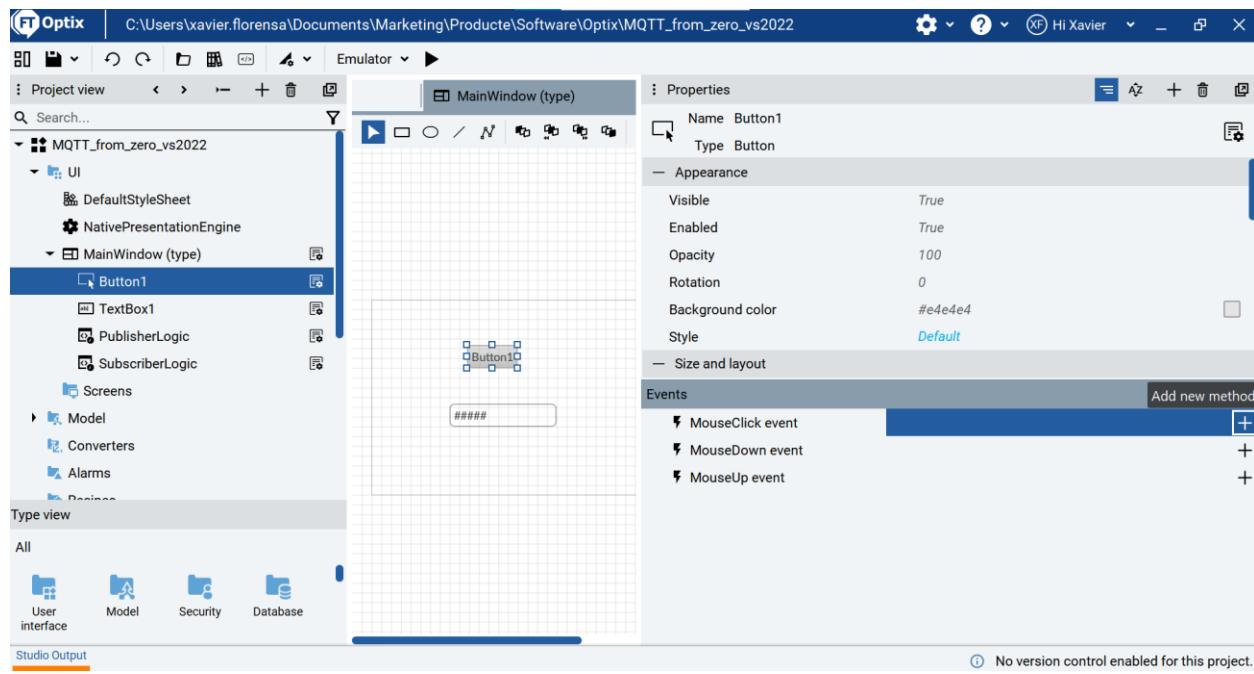
Open Publisherlogic

We have no errors now

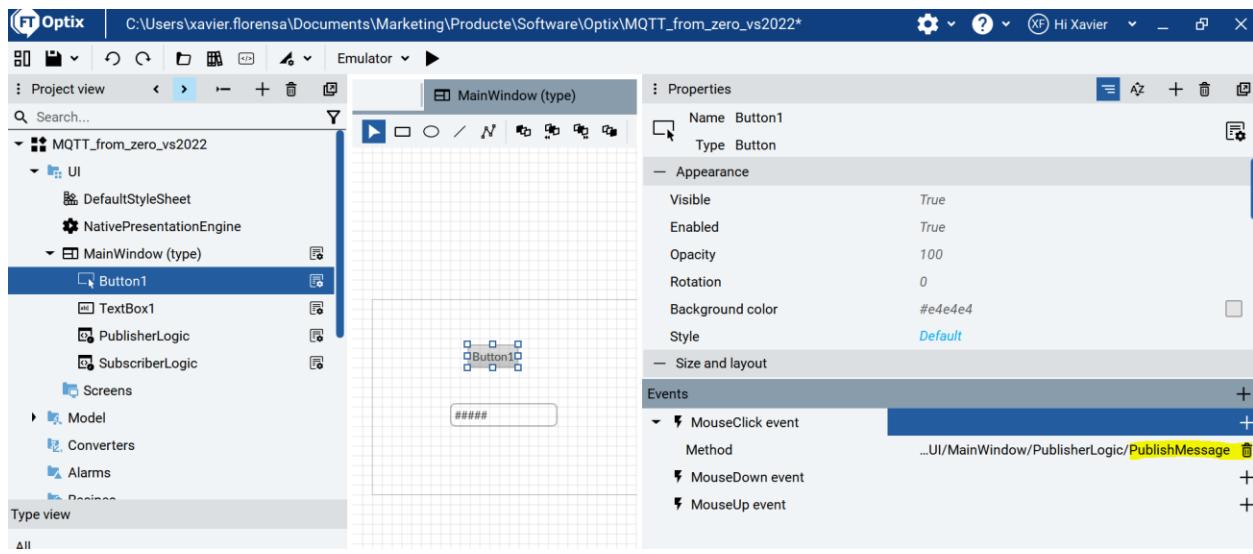


```
1 //region StandardUsing
2 using System;
3 using FTOptix.CoreBase;
4 using FTOptix.HMIProject;
5 using UAManagedCore;
6 using OpcUa = UAManagedCore.OpcUa;
7 using FTOptix.NetLogic;
8 using FTOptix.UI;
9 using FTOptix.OPCUAServer;
10 #endregion
11 using uPLibrary.Networking.M2Mqtt;
12 using uPLibrary.Networking.M2Mqtt.Messages;
13
14 public class PublisherLogic : BaseNetLogic
15 {
16     0 references
17     public override void Start()
18     {
19         var brokerIpAddressVariable = Project.Current.GetVariable("Model/BrokerIpAddress");
20
21         // Create a client connecting to the broker (default port is 1883)
22         publishClient = new MqttClient(brokerIpAddressVariable.Value);
23         // Connect to the broker
24         publishClient.Connect("FTOptixPublishClient");
25         // Assign a callback to be executed when a message is published to the broker
26     }
27 }
```

Let's build the mouseclick action



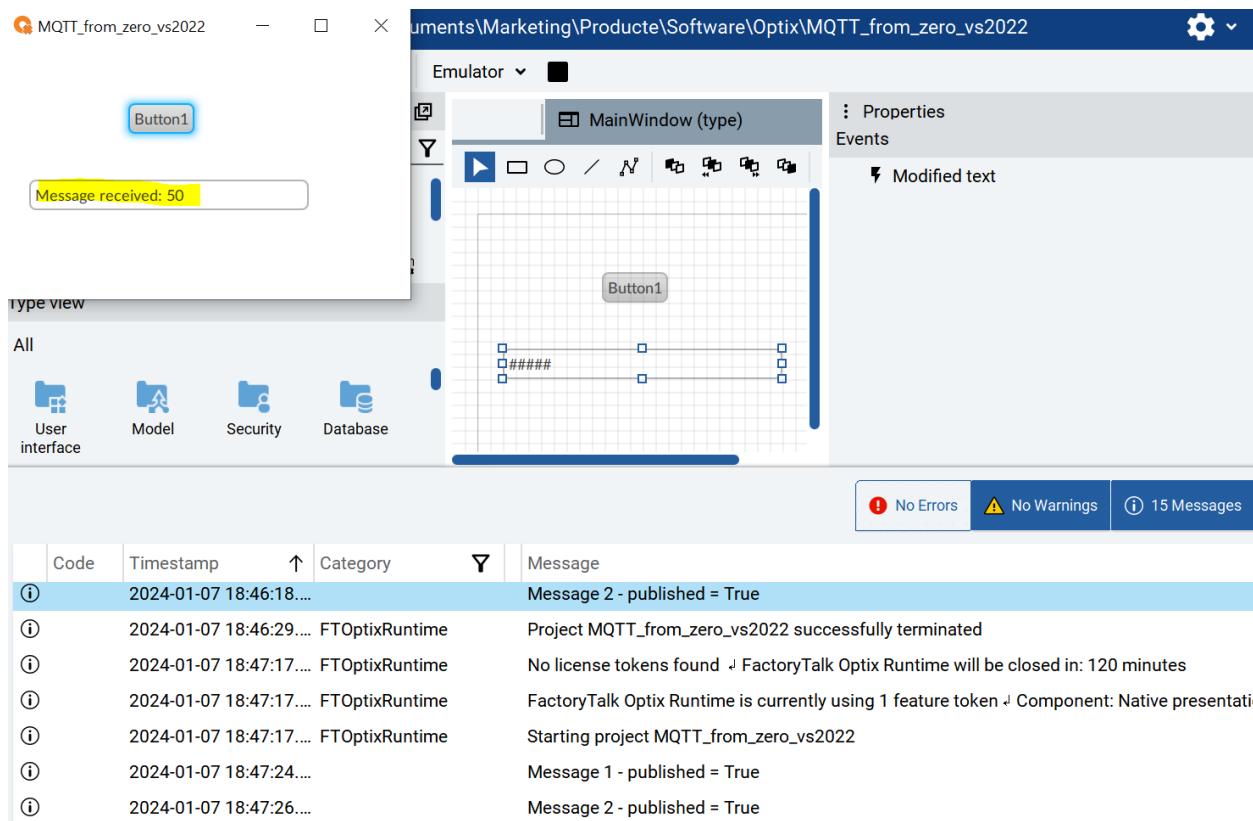
We see the only method declared



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 subscriberlogic 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);  
}
```

22. First steps with OptixPanel

Standard 12.1 inches

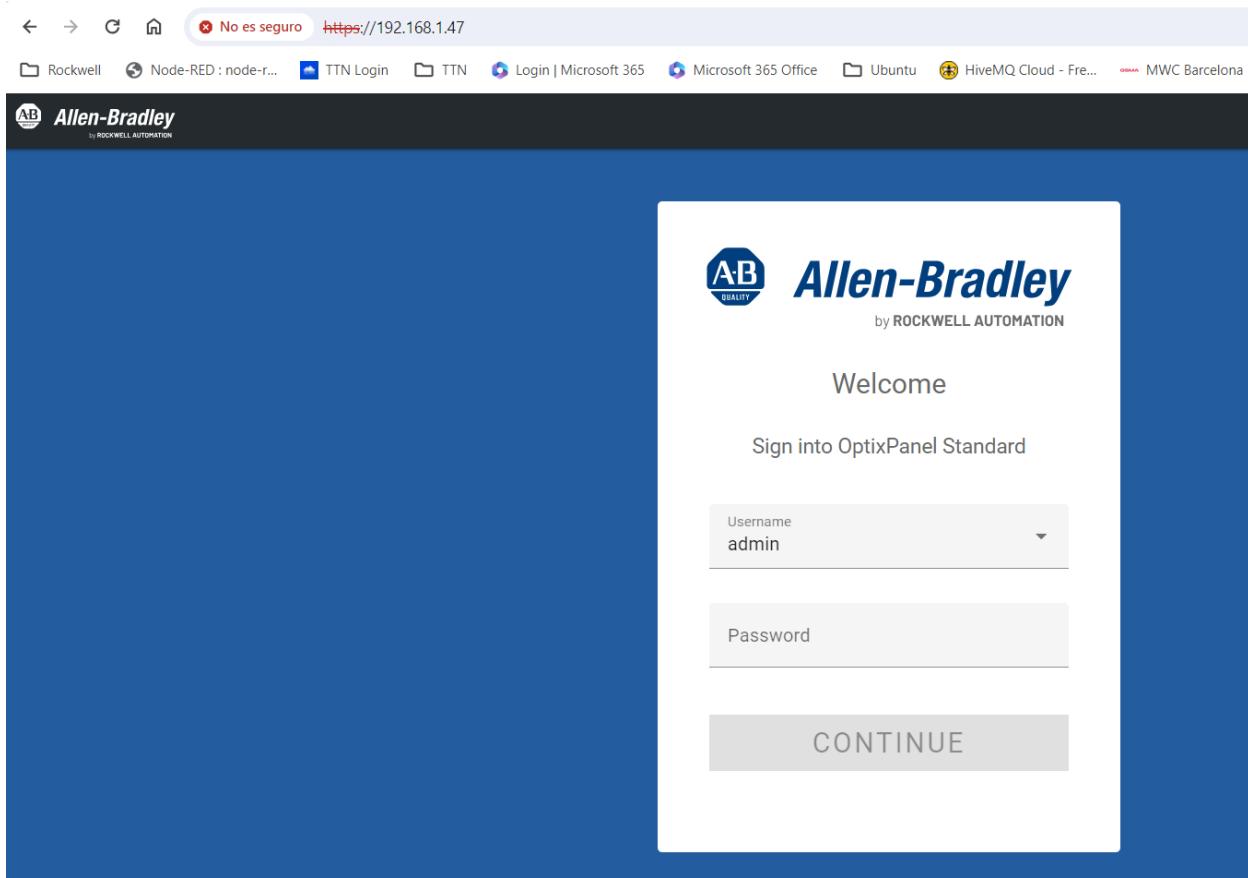
2800S-121FM-N1S

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

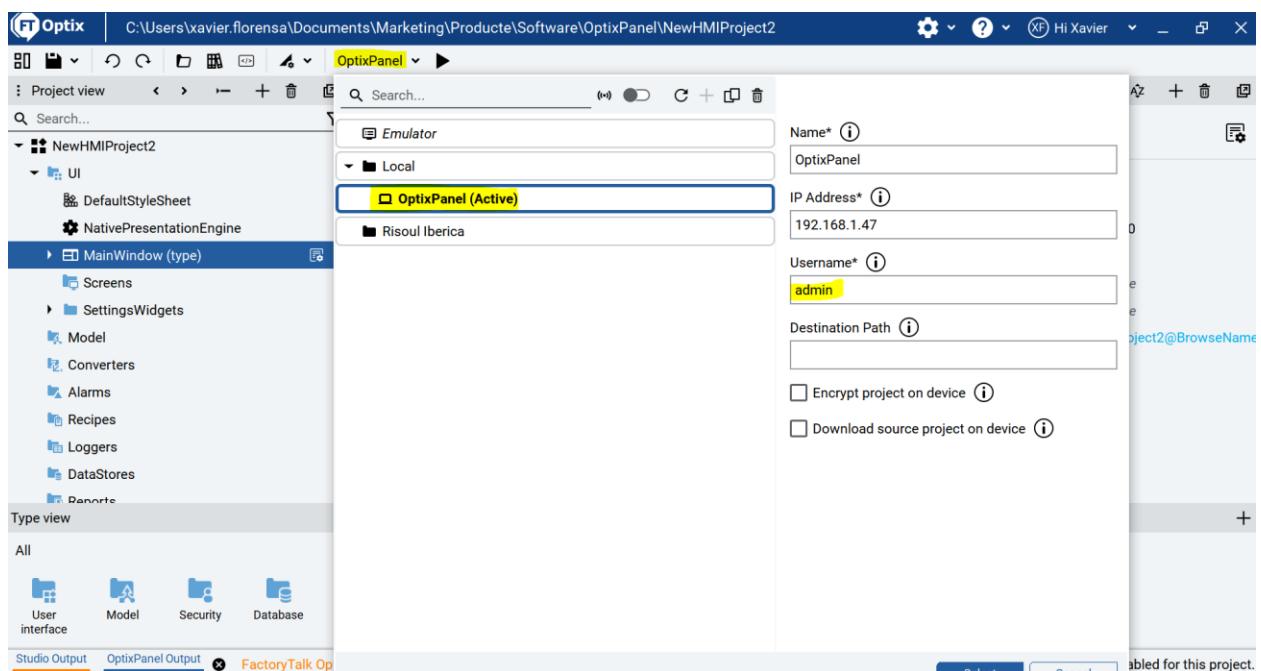
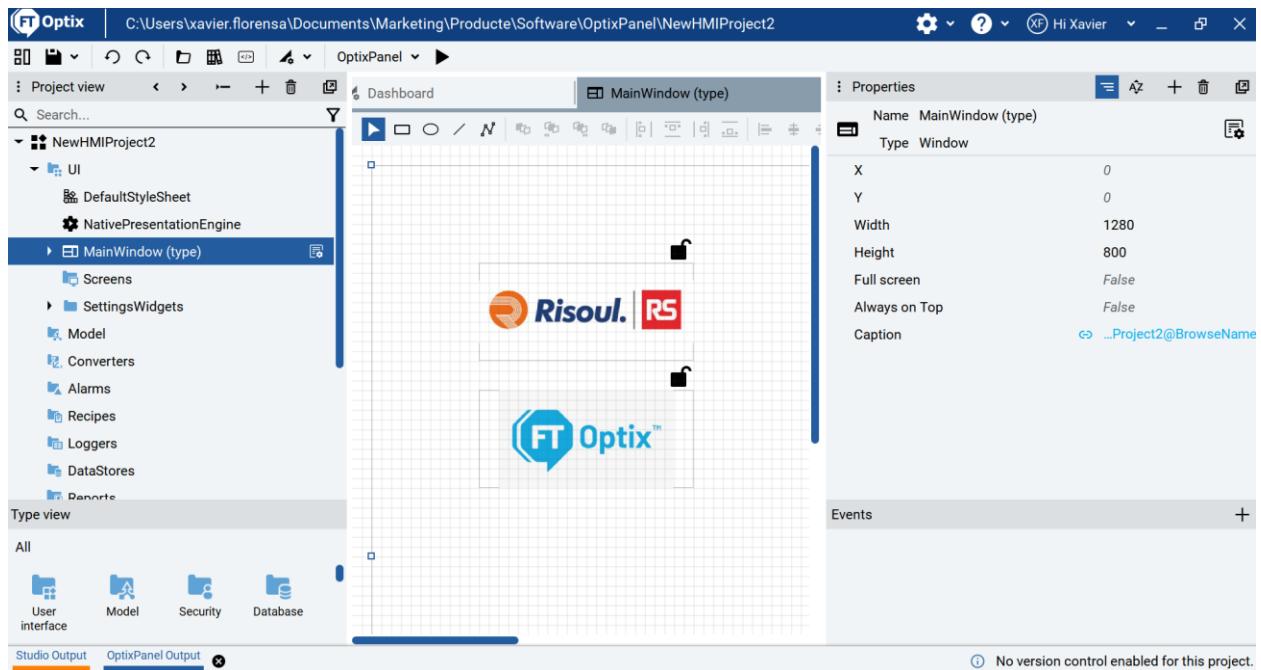
Password:admin

Let's change to Risoul10

A message with:

"No FactoryTalk Optix application found, please upload one using FactoryTalk Optix Studio"

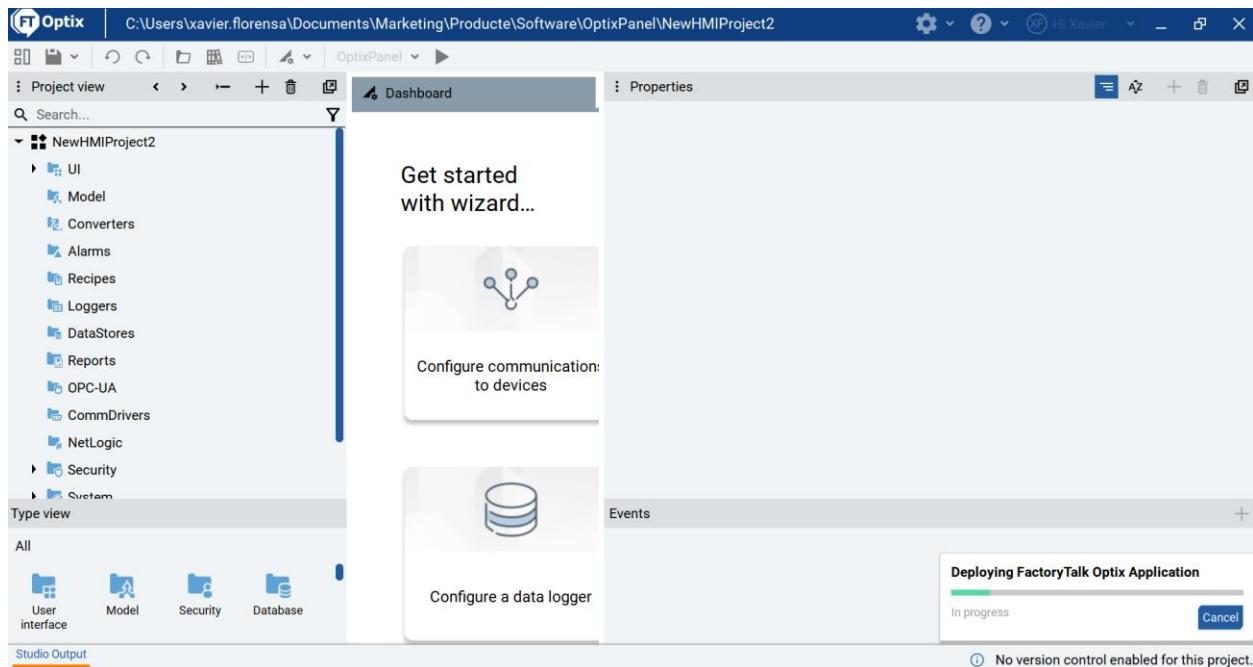
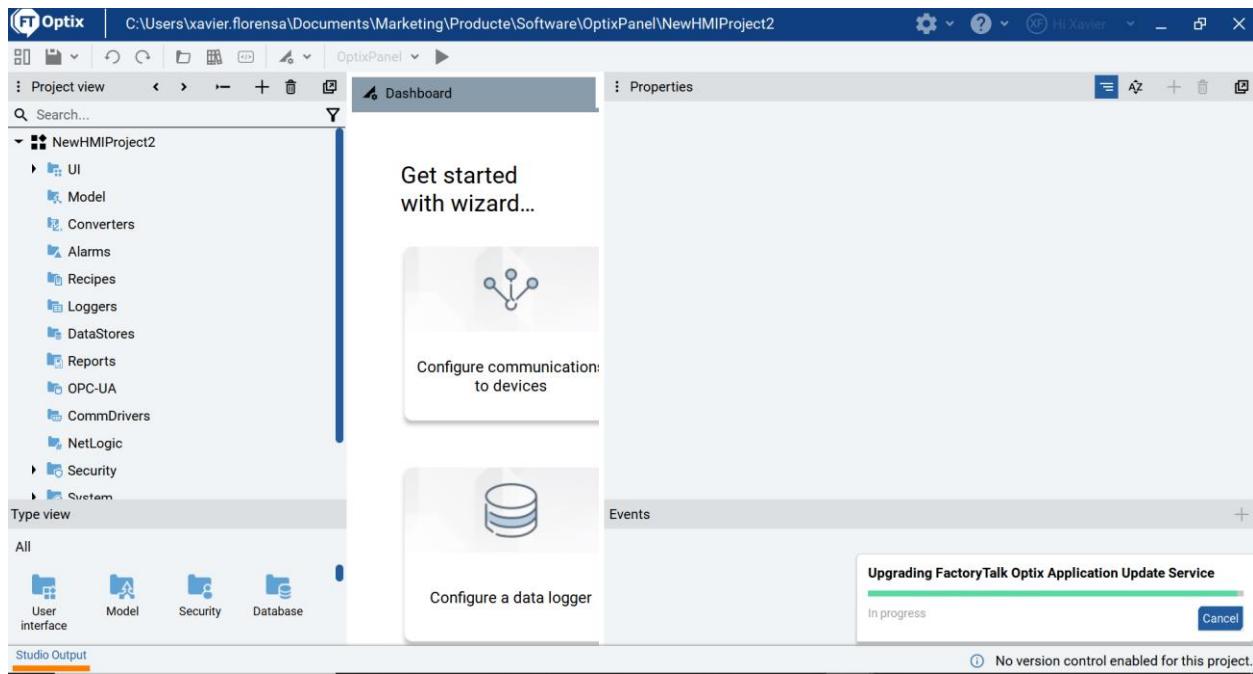
Let's create a new application and select the destination different from Emulator but OptixPanel



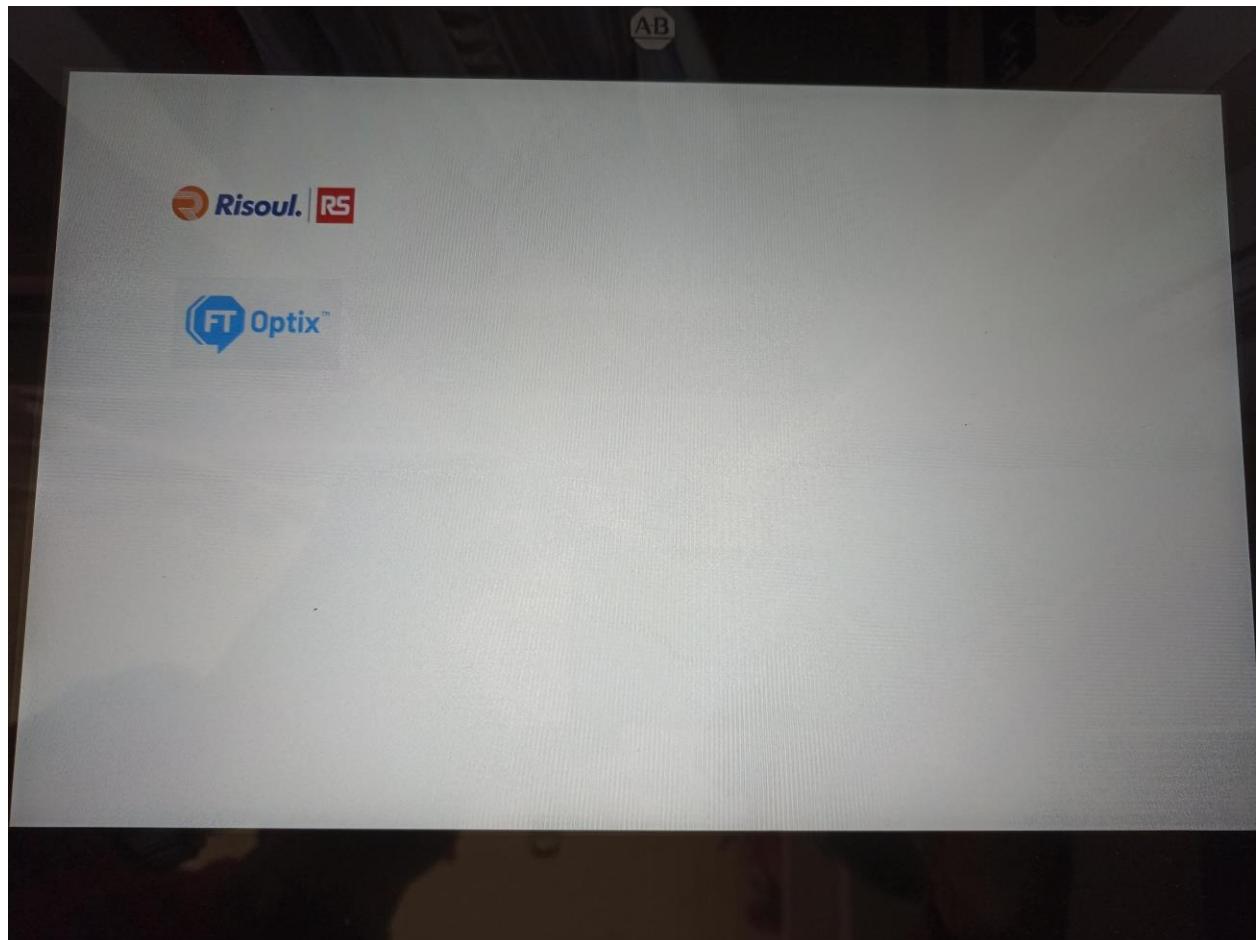
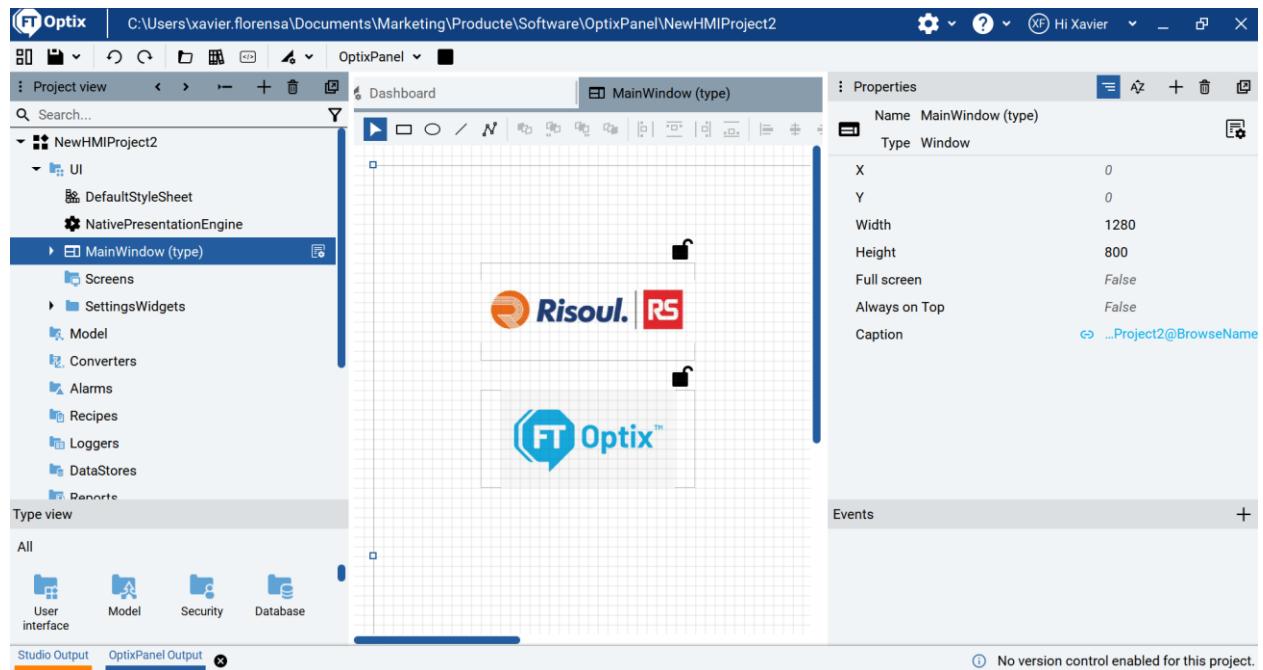
And test with the play button on the Panel

Admin

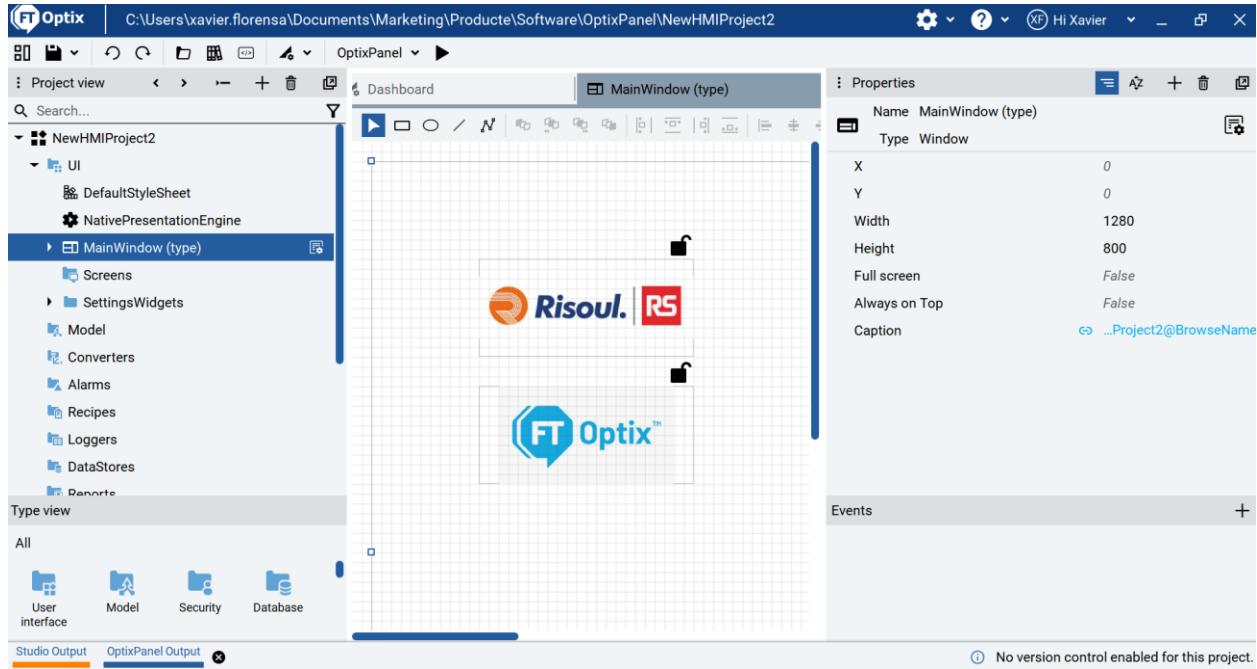
Password: Risoul10



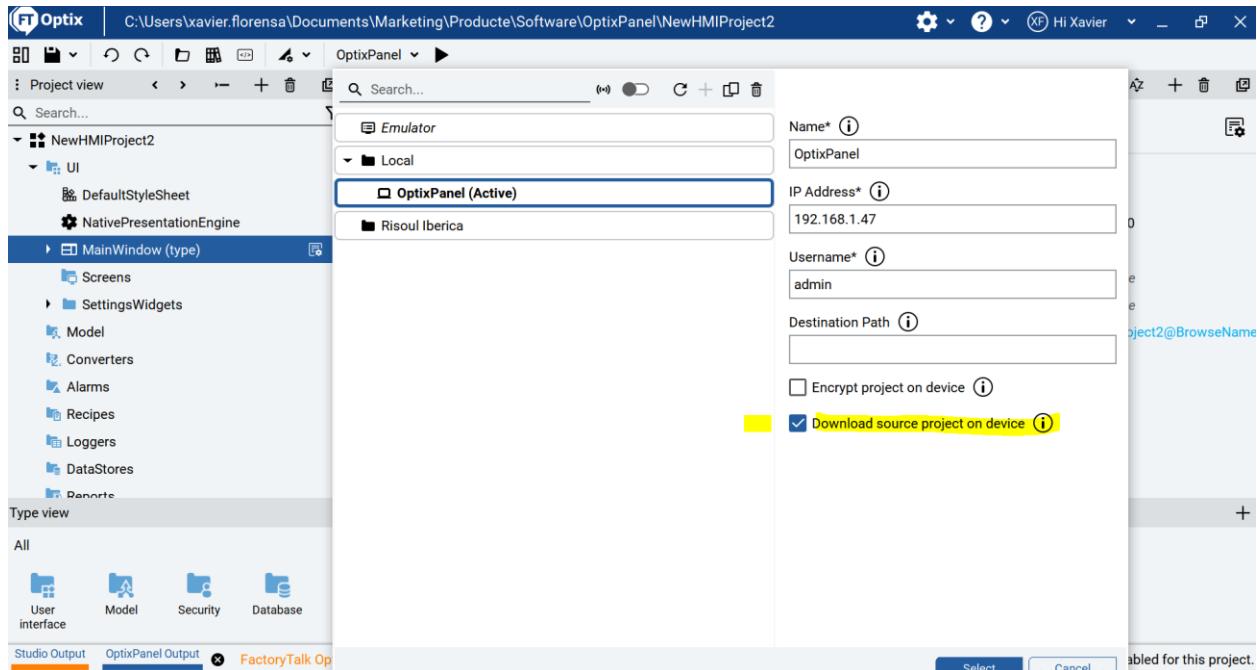
Success



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.

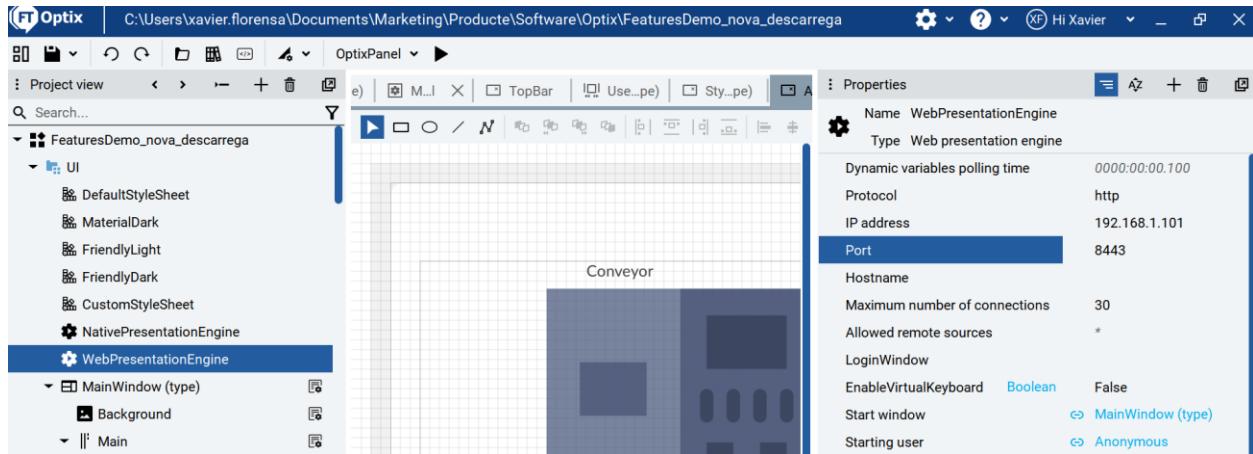
But if you power OptixPanel again it will open with the new application. After 10 seconds.

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



23. NetLogic and C# tutorial

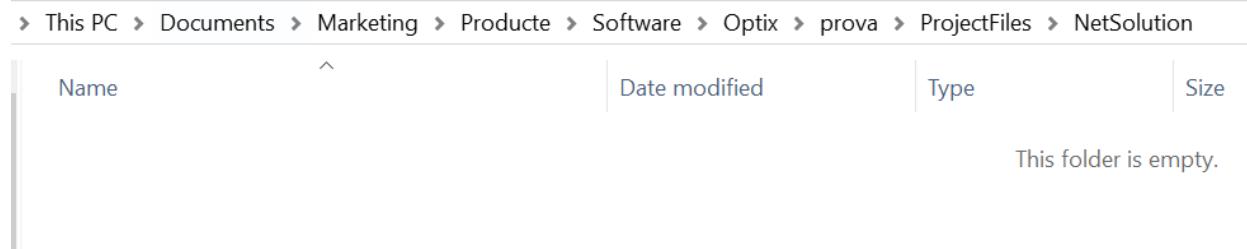
https://github.com/FactoryTalk-Optix/NetLogic_CheatSheet

24. Understanding compilation of NetLogic code

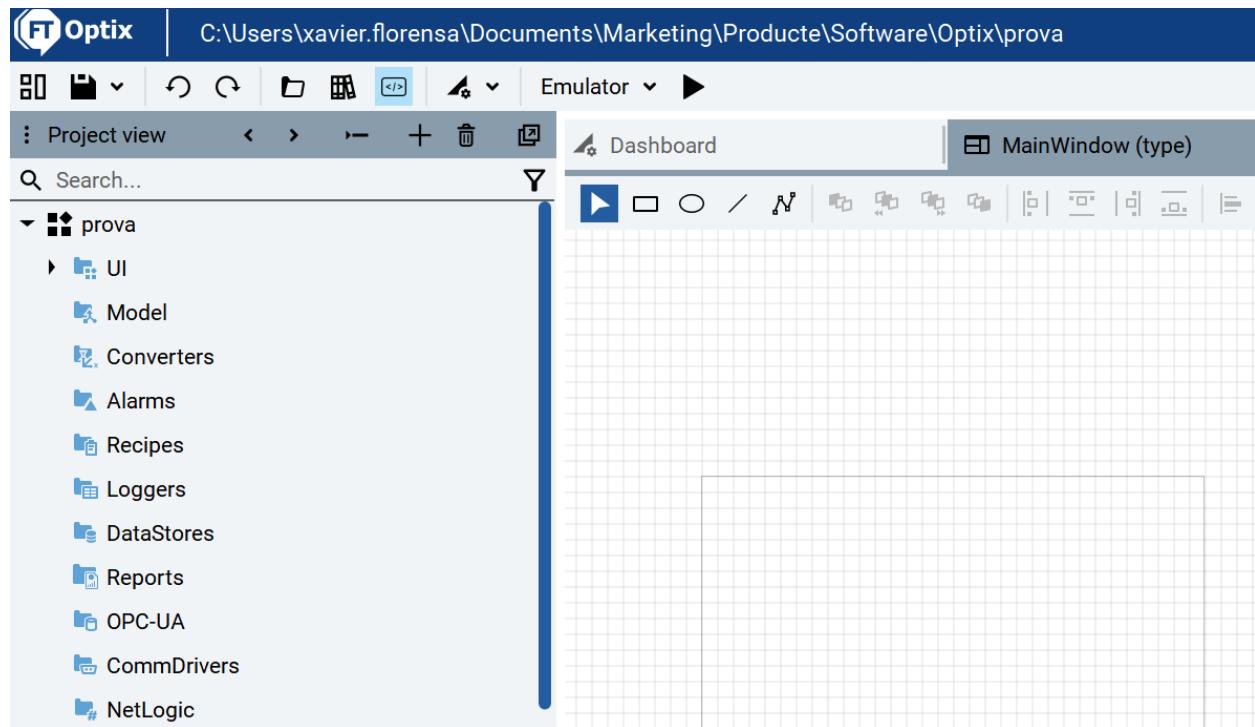
After you create a new standard project, before creating a Netlogic script and even compiling, you will have nothing on folder Projects/Netlogic/bin

Let's create a project in FT Optix Studio called prova

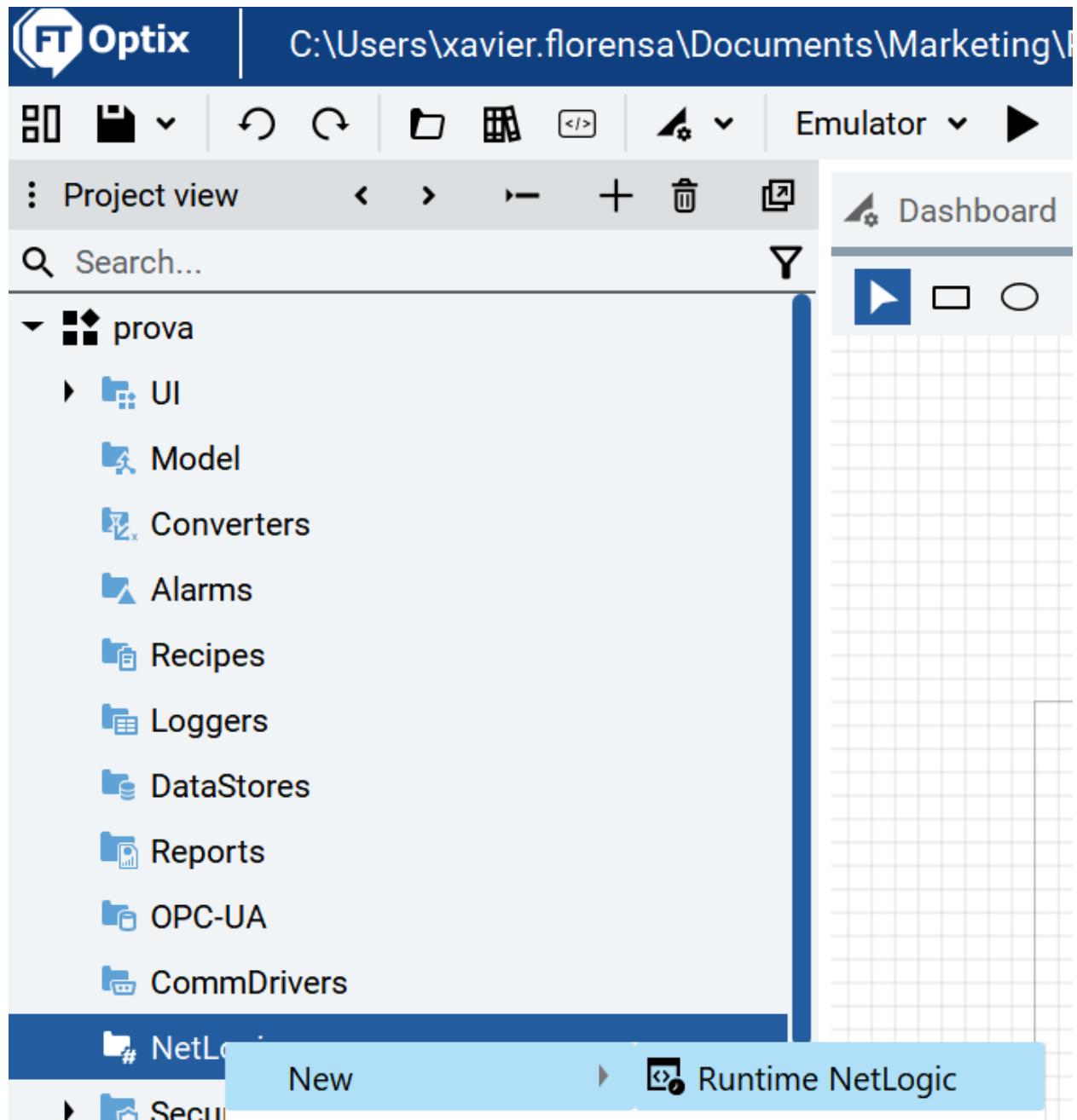
There is nothing on the NetSolution folder



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.

```

File Edit Selection View Go Run ... ← → prova (Workspace)
EXPLORER PROVA (WORKSPACE) ... RuntimeNetLogic1.cs
NetSolution > RuntimeNetLogic1.cs ...
1 #region Using directives
2 using System;
3 using UAManagedCore;
4 using OpcUa = UAManagedCore.OpcUa;
5 using FTOptix.HMIProject;
6 using FTOptix.Retentivity;
7 using FTOptix.UI;
8 using FTOptix.NativeUI;
9 using FTOptix.CoreBase;
10 using FTOptix.Core;
11 using FTOptix.NetLogic;
12 #endregion
13
0 references
14 public class RuntimeNetLogic1 : BaseNetLogic
15 {
16     0 references
17     public override void Start()
18     {
19         // Insert code to be executed when the user-defined logic is started
20     }
21
22     0 references
23     public override void Stop()
24     {
25         // Insert code to be executed when the user-defined logic is stopped
26     }
}

```

The screenshot shows the Visual Studio Code interface with the 'RuntimeNetLogic1.cs' file open in the center editor pane. The code itself is as follows:

```

#nullable enable
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;

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

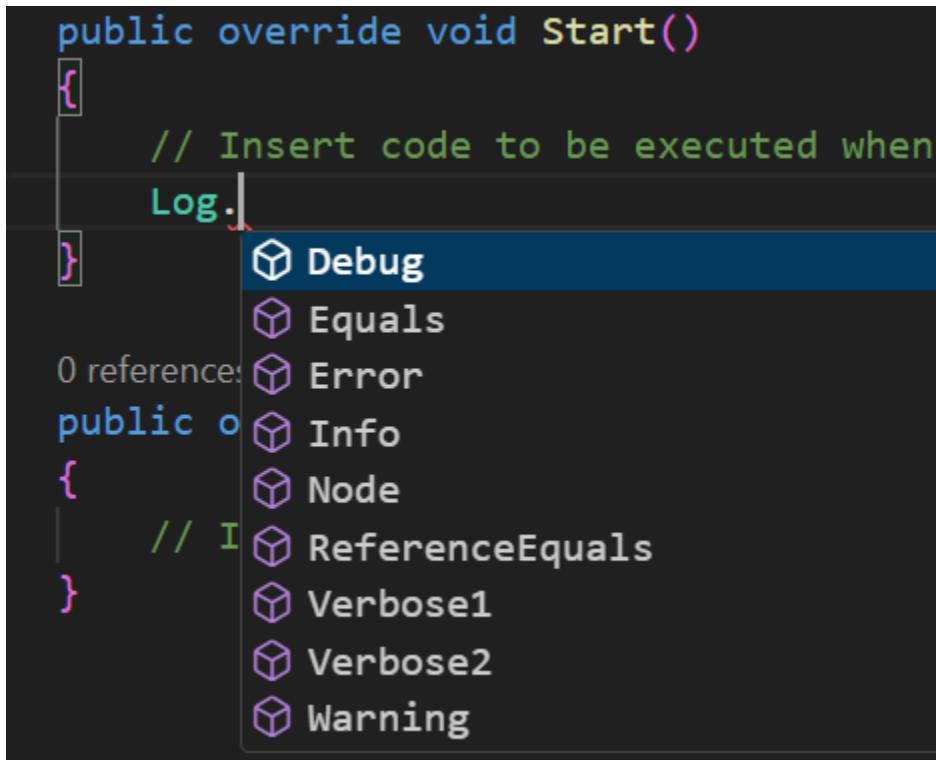
    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}

```

25. Understanding Classes (Objects), methods and data

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

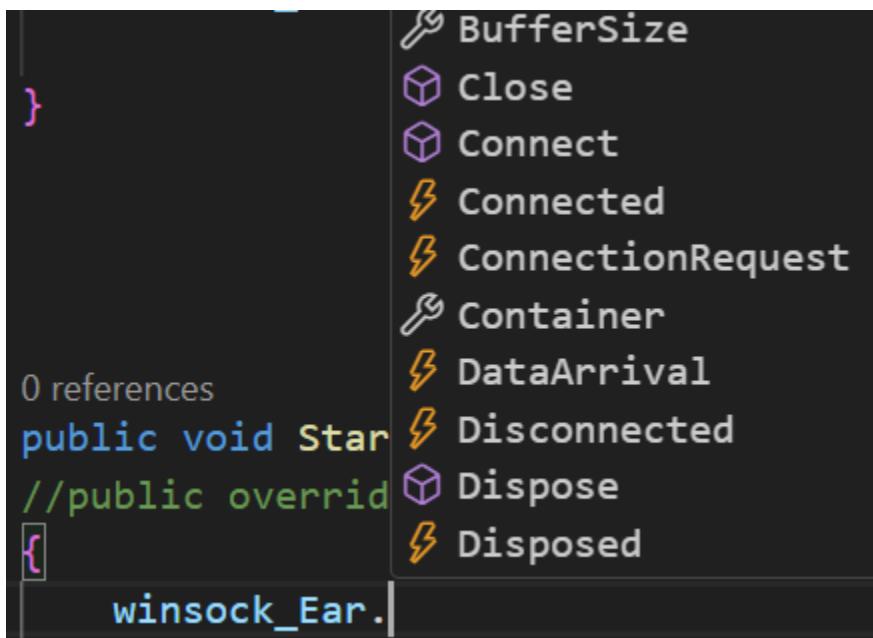
For example



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

The screenshot shows a code editor with a dropdown menu open at the cursor position. The menu contains several items: Debug, Equals, Error, Info, Node, ReferenceEquals, Verbose1, Verbose2, and Warning. The 'Debug' item is highlighted with a blue background, indicating it is the selected option.

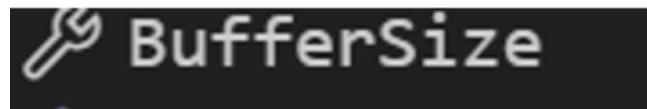
Or for example (an instance of a class is marked in sky blue)



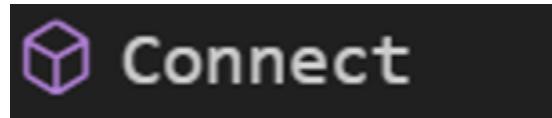
```
}
```

The screenshot shows a code editor with a dropdown menu open at the cursor position. The menu contains several items: BufferSize, Close, Connect, Connected, ConnectionRequest, Container, DataArrival, Disconnected, Dispose, and Disposed. The 'Connected' item is highlighted with a yellow background, indicating it is the selected option.

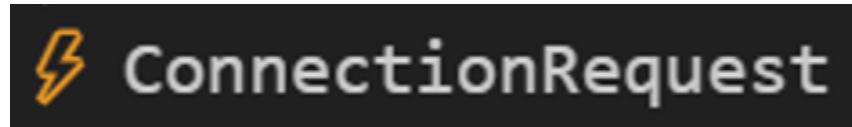
then you will write . and you will see (thank you to IntelliSense) a collection of



Tool icon: read write access data



Cube icon: method (action to perform a task)



Blizard: event

26. 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 UAManagedCore;
using OpcUa = UAManagedCore.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 RuntimeNetLogic1 : BaseNetLogic
{
    public int metodo_suma(int primero, int segundo)
    {
        return primero + segundo;
    }
    public override void Start()
```

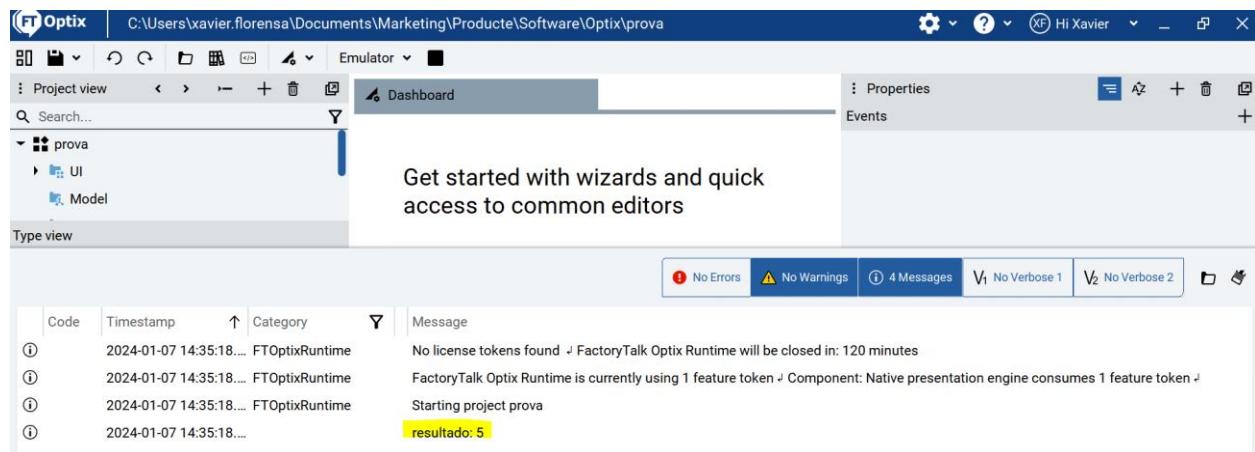
```

{
    // 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



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 UAManagedCore;

```

```

using OpcUa = UAManagedCore.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 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);
    }

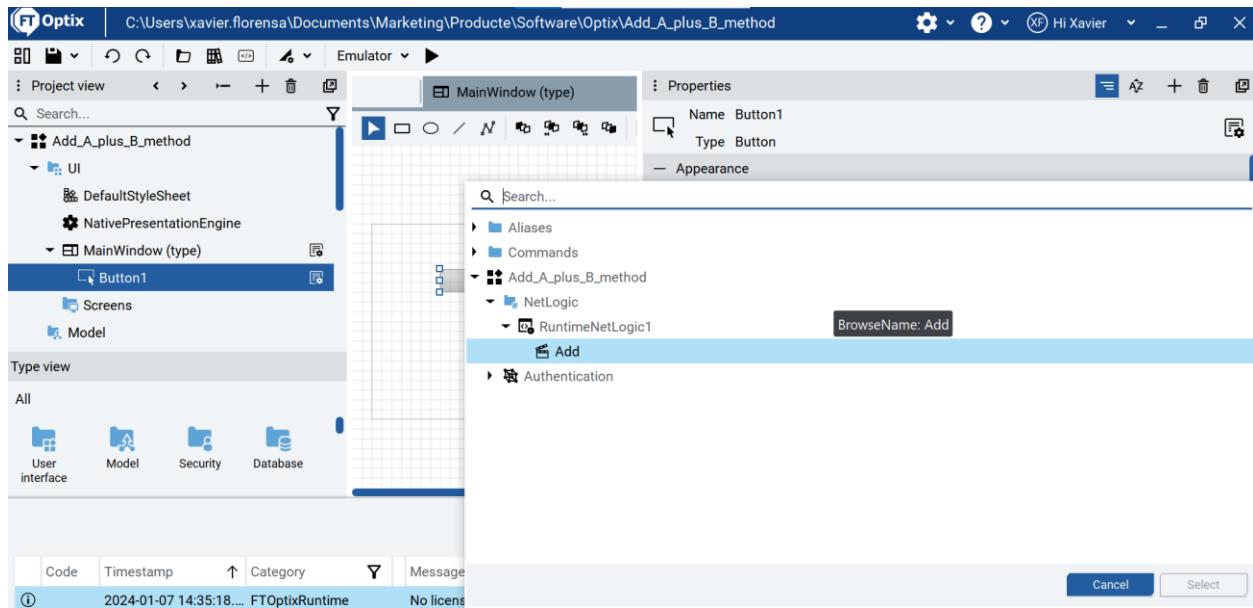
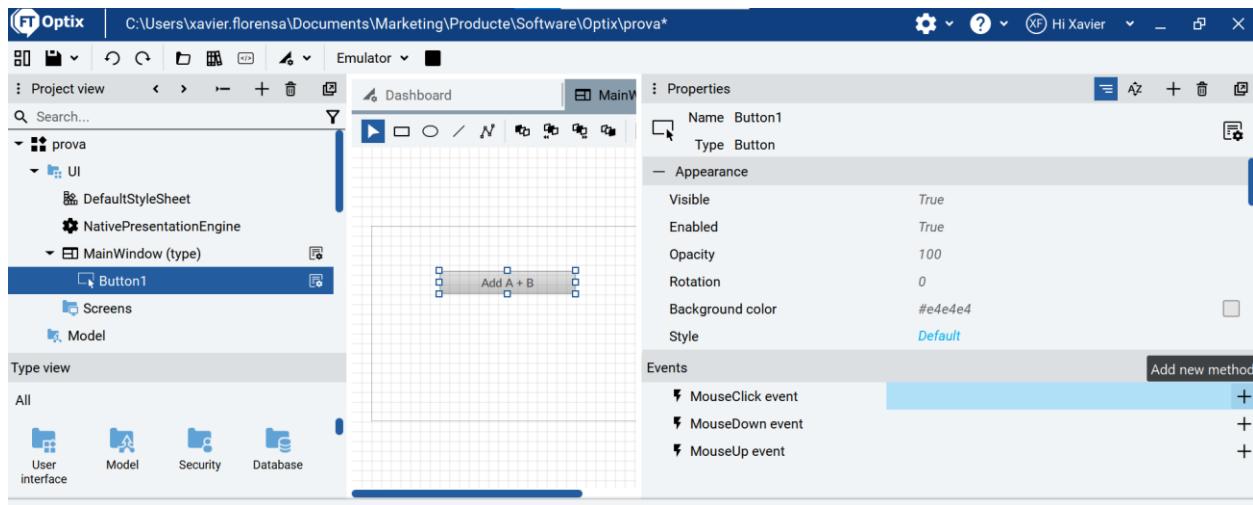
    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}

```

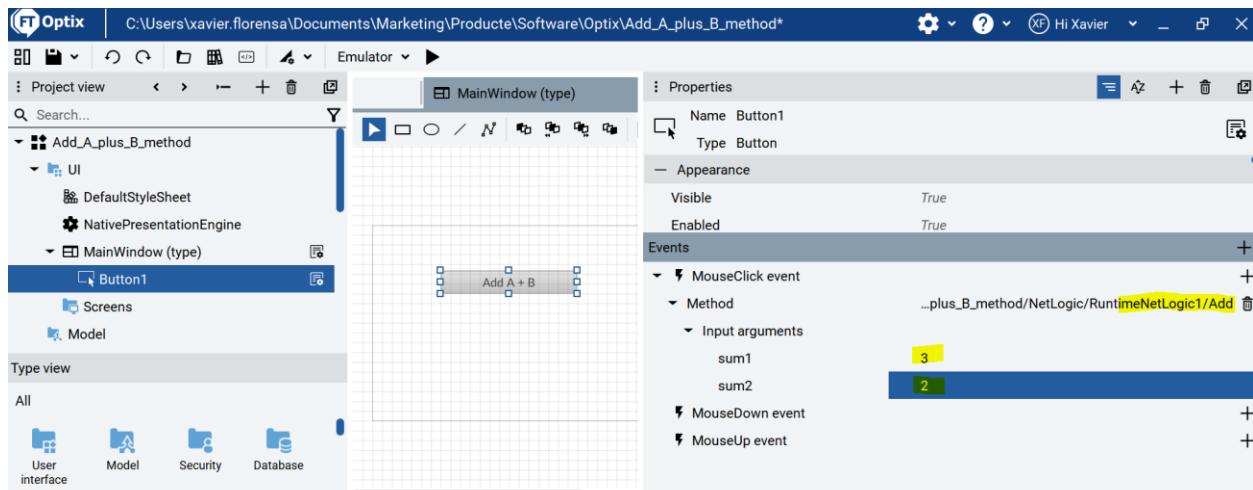
Save on VS Code

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

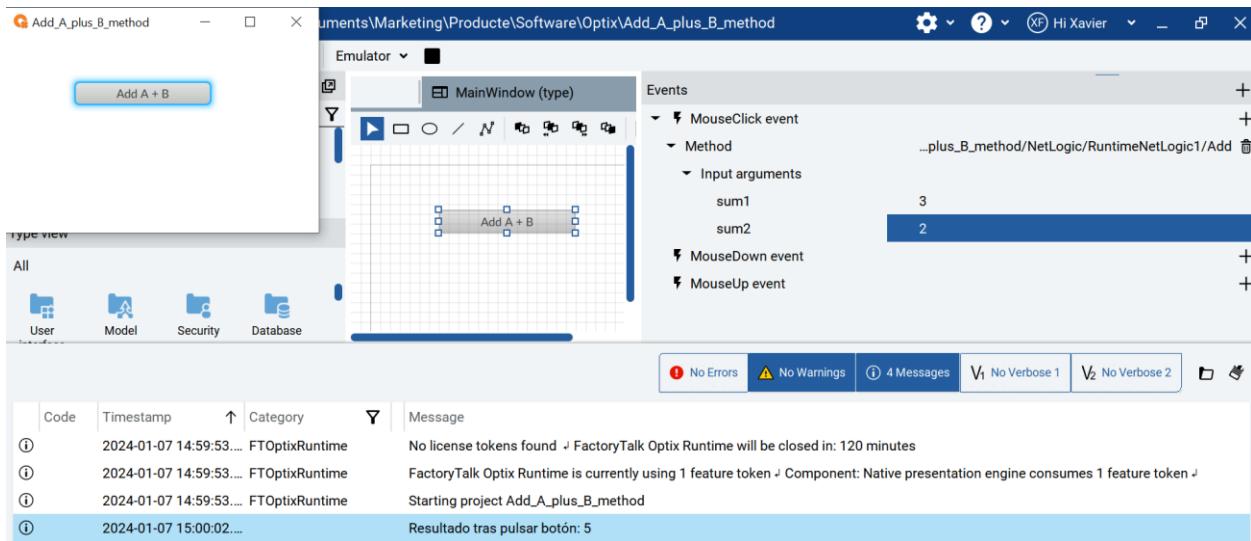
Create a button and configure Mouseclick event



Complete the input arguments



Save on Ft Optix and Test the application



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

public class RuntimeNetLogic1 : BaseNetLogic
{
```

```
[ExportMethod]
```

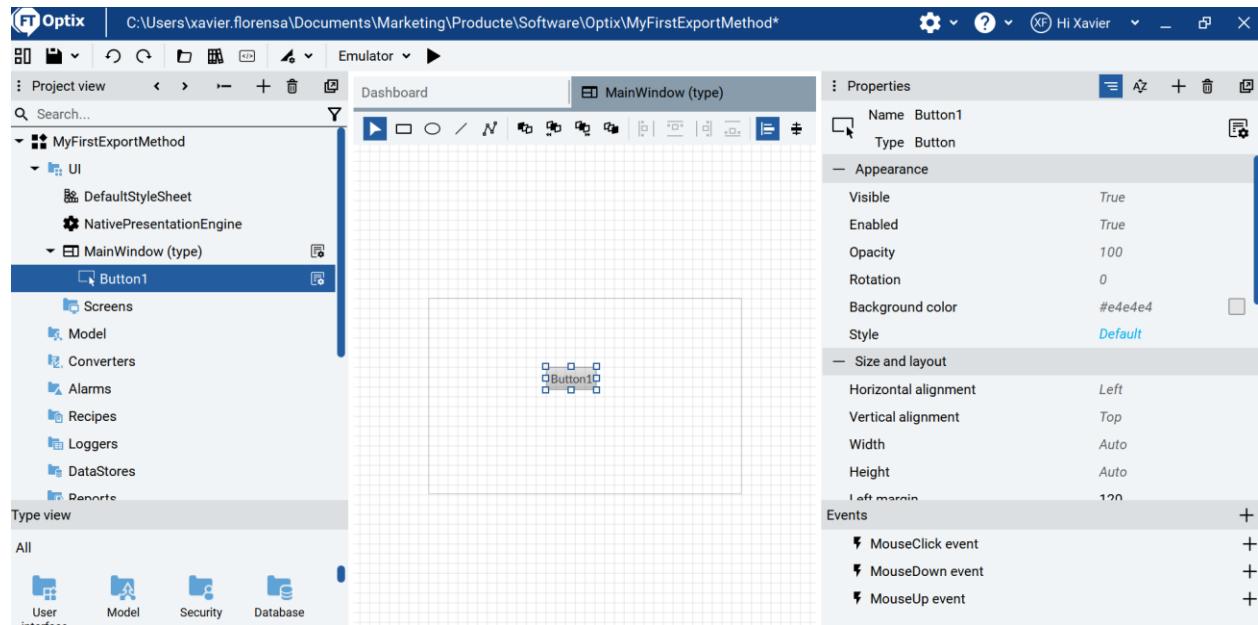
```
public void MyNewMethod()
{
    Log.Info("Hello World, a button has been pressed");
}

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

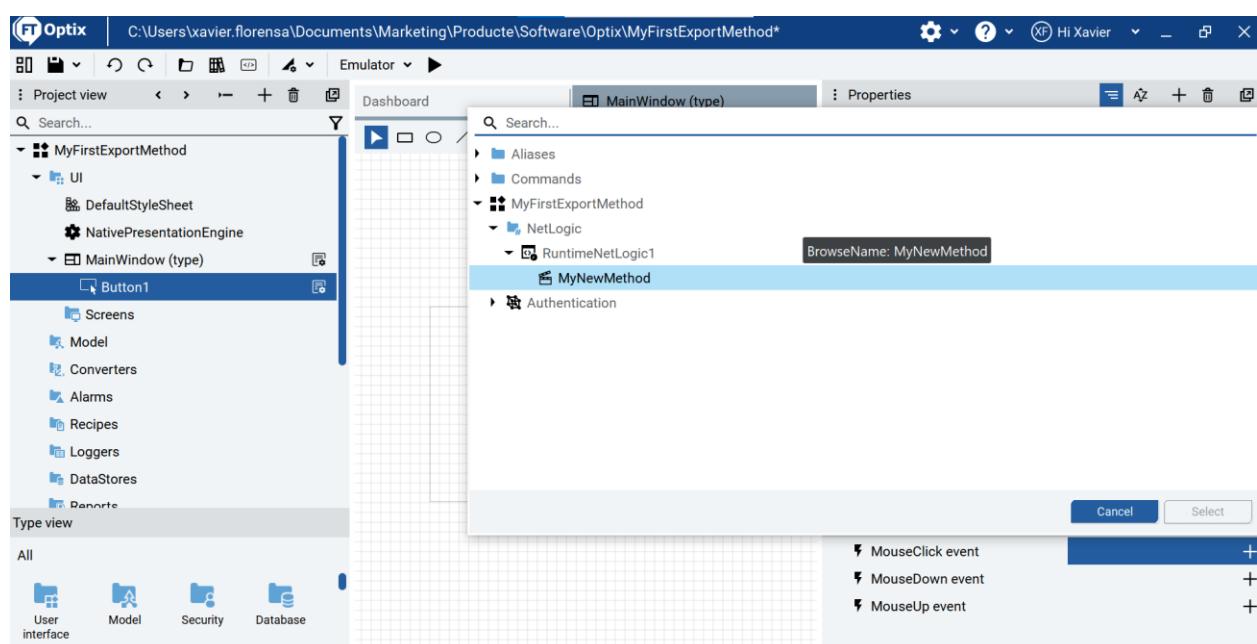
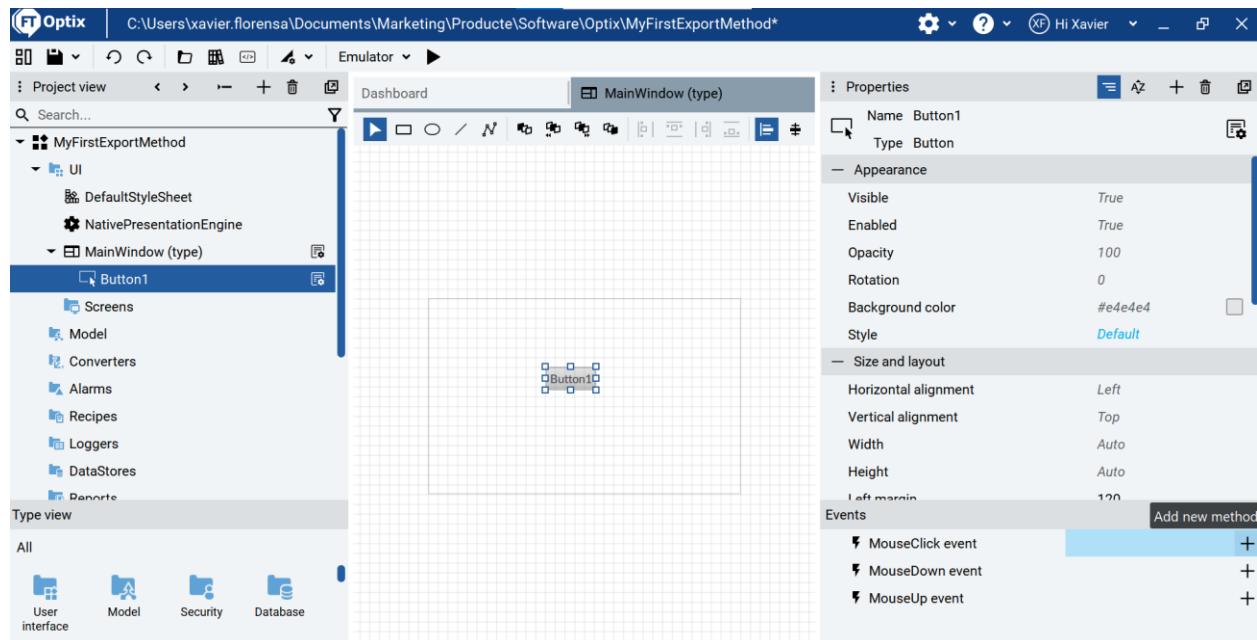
public override void Stop()
{
    // Insert code to be executed when the user-defined logic is stopped
}
}
```

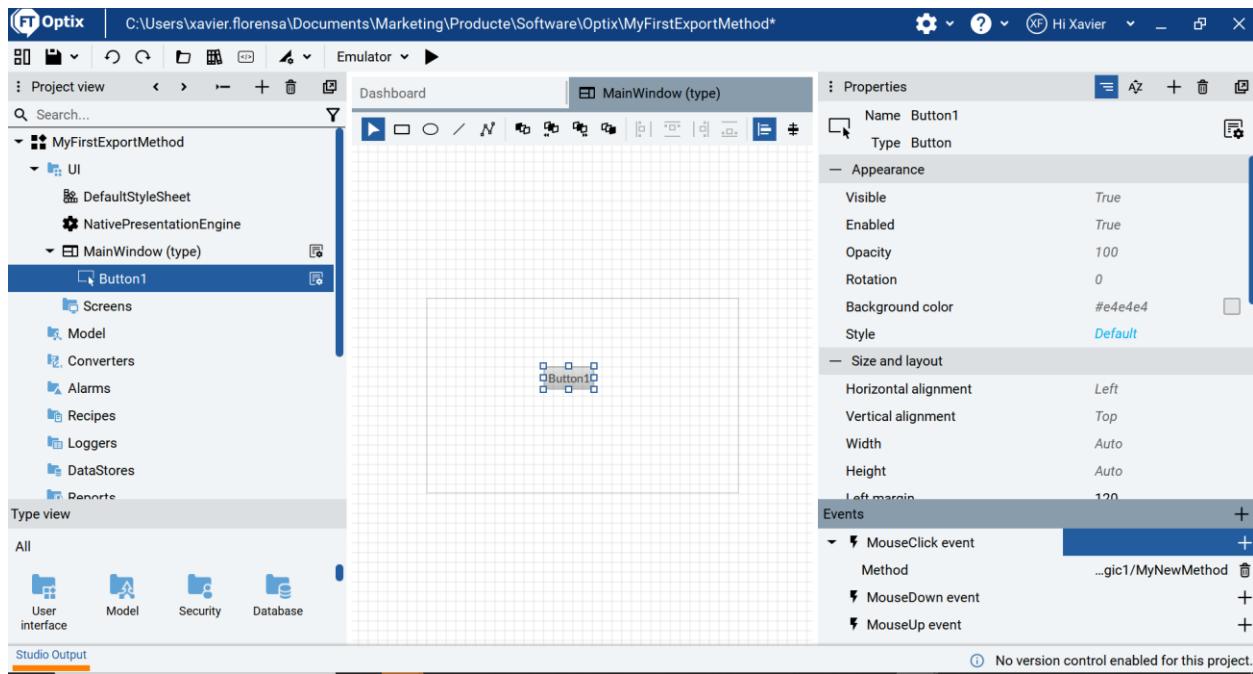
Next, save the code (in order for the compiler to build the new Method)

Add a new button

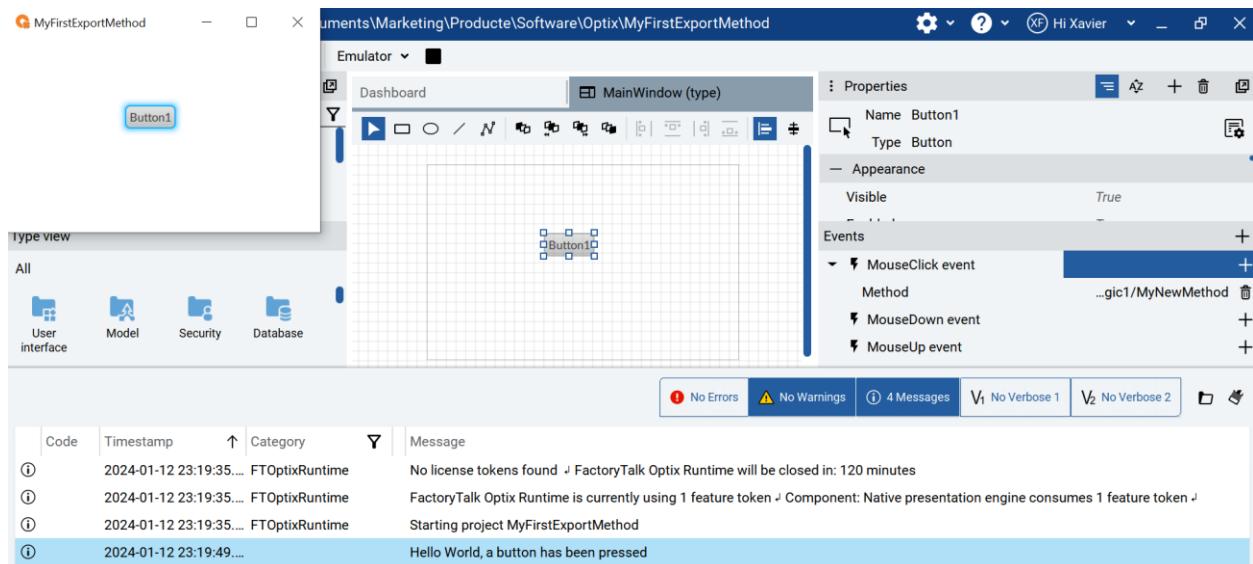


Add a new MouseClick event





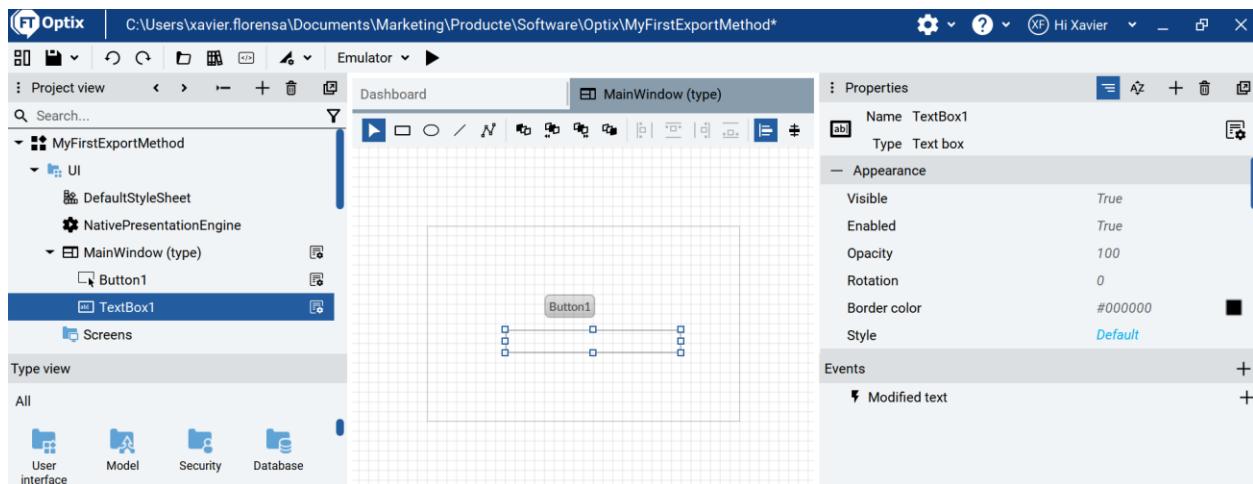
Now execute the project



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

28. Linking events with Methods with input parameters

Create a textbox continuing with last project



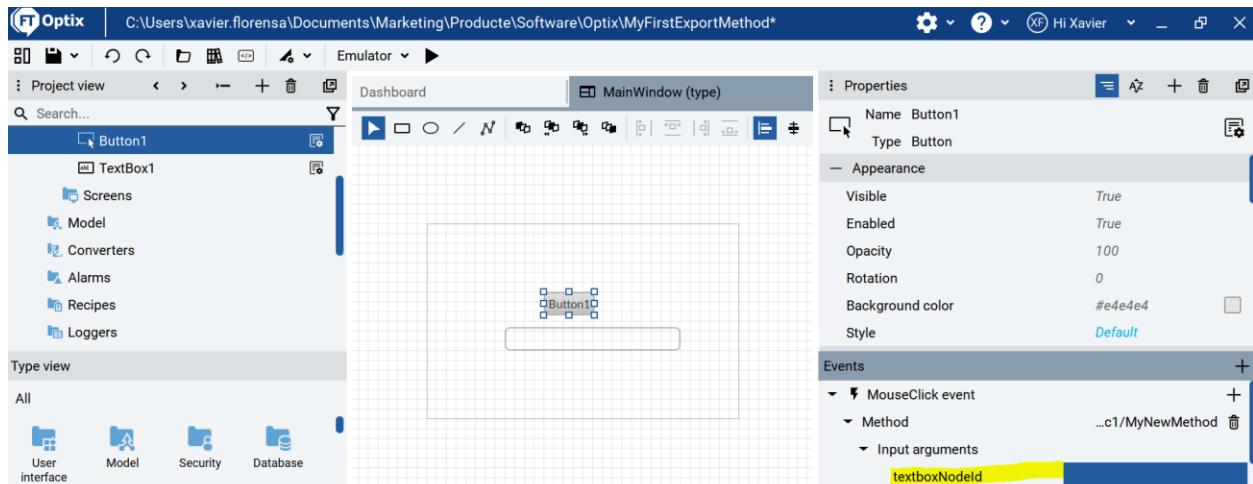
Add this code to the Netlogic code

```
[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



Add a dynamic link between input argument and TextBox1 object

Style

Default

Events

- MouseClick event
- Method
- Input arguments

...c1/MyNewMethod Delete

Select node

textboxNodeld

Basic Advanced nWindow > Button1 > MouseClickEventHandler1 > MethodsToCall > MethodContainer1 > InputArguments > textboxNodeld

Search...

Aliases
Types
Commands
Retained alarms
CommController
OPCUAClientController
MyFirstExportMethod
UI
DefaultStyleSheet
NativePresentationEngine
MainWindow (type)
Button1
TextBox1

Attribute (i) Node id

Cancel Select Remove link

FT Optix C:\Users\xavier.florensa\Documents\Marketing\Products\Software\Optix\MyFirstExportMethod*

Project view Dashboard MainWindow (type)

Search...

Button1 TextBox1 Screens Model Converters Alarms Recipes Loggers

Type view All User interface Model Security Database

Name: Button1 Type: Button

Appearance

- Visible: True
- Enabled: True
- Opacity: 100
- Rotation: 0
- Background color: #e4e4e4
- Style: Default

Events

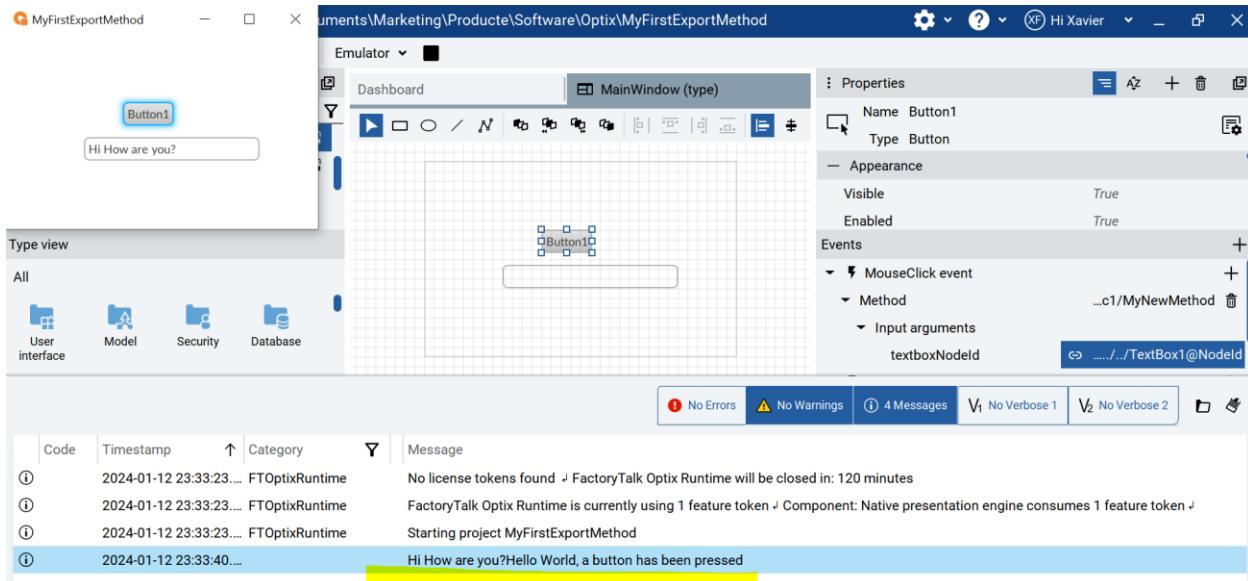
- MouseClick event
- Method
- Input arguments

...c1/MyNewMethod Delete

.../TextBox1@Nodeid

Execute the code

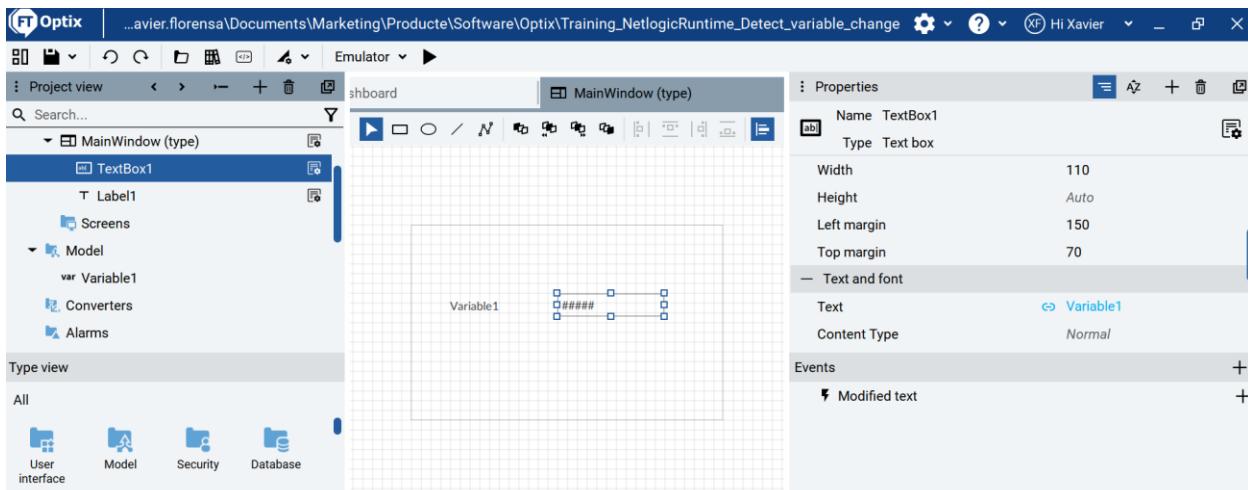
Write something on the box, hit enter and press the button



29. 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.HMIPrjject;
using FTOptix.Retentivity;
```

```

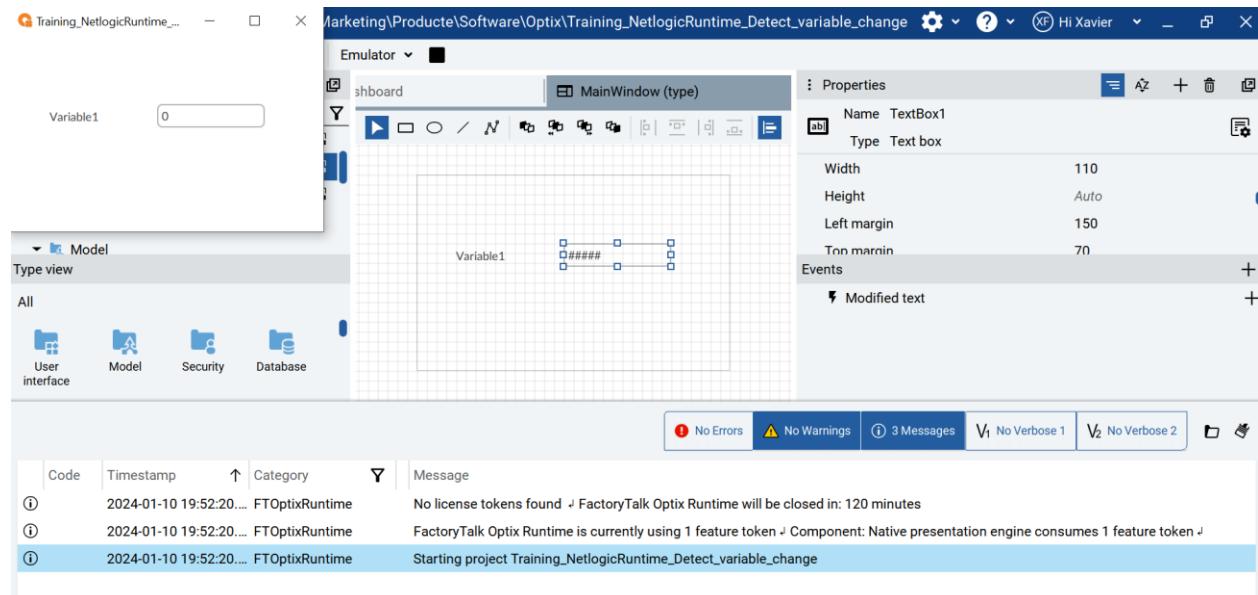
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

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;
    }

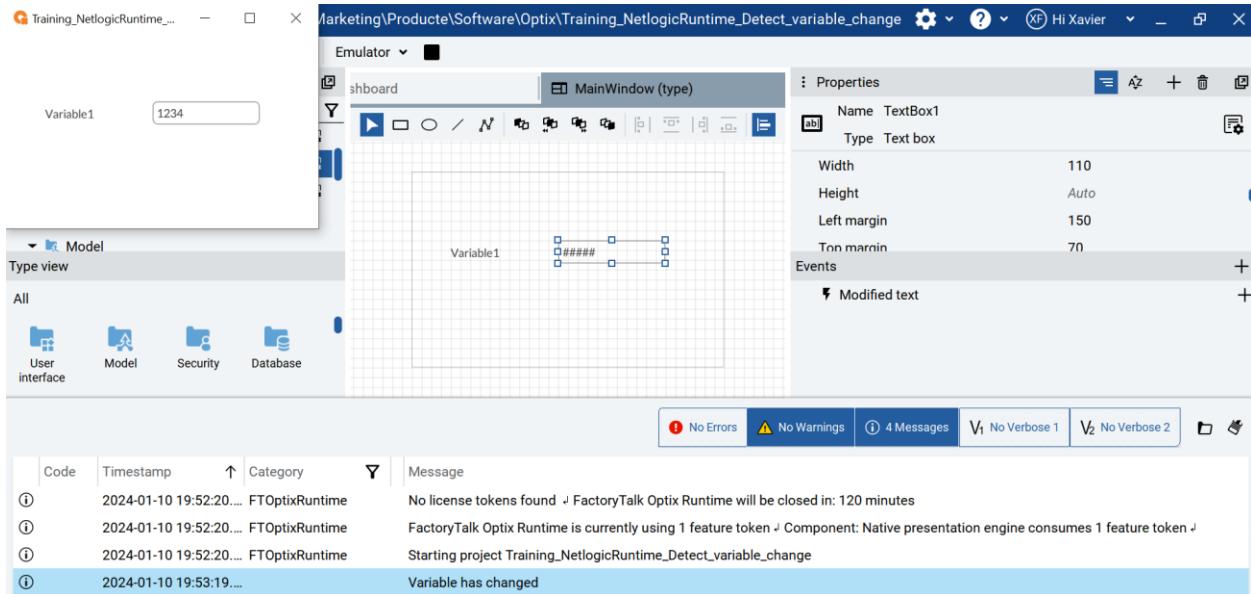
    public override void Stop()
    {
    }
    private void variable1_VariableChange(object sender, VariableChangedEventArgs e)
    {
        Log.Info("Variable has changed");
    }
}

```

Let's execute the code

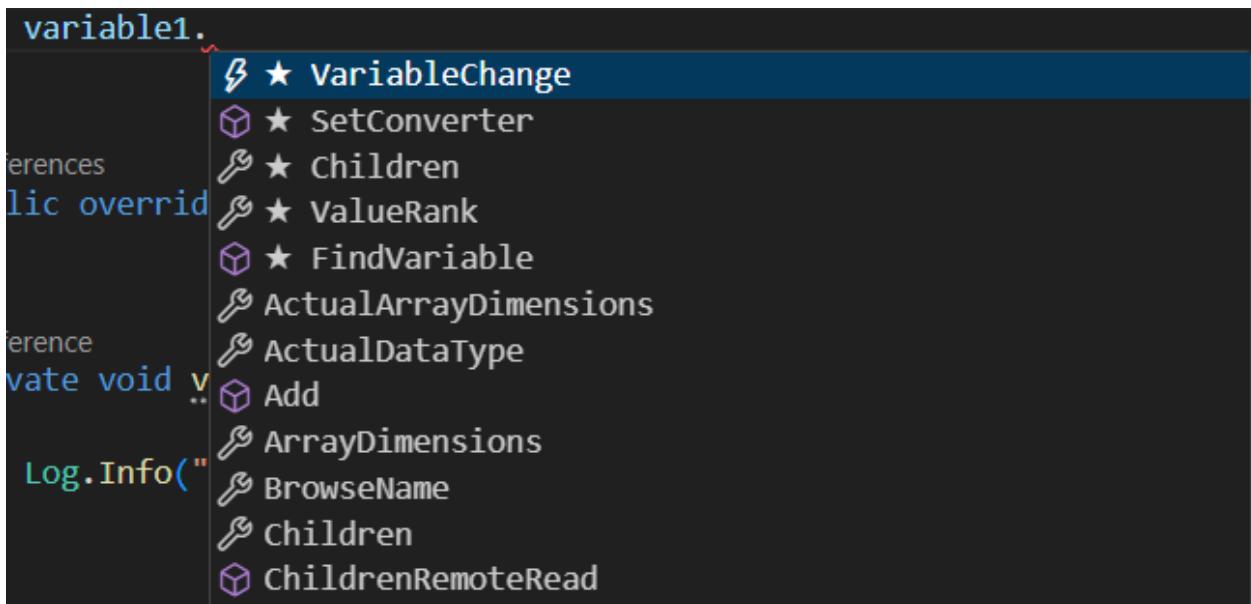


Now change the variable writing a number in the textbox



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.  
    ↴  
    ⚡ ChildrenRemoteWrite  
    ⚡ DataType  
    ⚡ DataValue  
    ↴ Delete  
    ⚡ Description  
    ⚡ DisplayName  
    ↴ Equals  
    ↴ Find  
    ↴ Find<>  
    ↴ FindByType<>  
    ↴ FindNodesByType<>  
    ↴ FindObject
```

```
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
```

```
variable1.  
    ↴  
    ↴ Remove  
    ↴ ResetDynamicLink  
    ↴ ServerTimestamp  
    ↴ SetAlias  
    ↴ SetConverter  
    ↴ SetDynamicLink  
    ↴ SourceTimestamp  
    ↴ Start  
    ↴ StatusCode  
    ↴ Stop  
    ↴ ToString  
    ↴ Value
```

```
    ↴ ValueRank  
    ⚡ VariableChange  
    ↴ VariableType
```

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.

30. Log info to emulator output

Just insert this sentence on your code

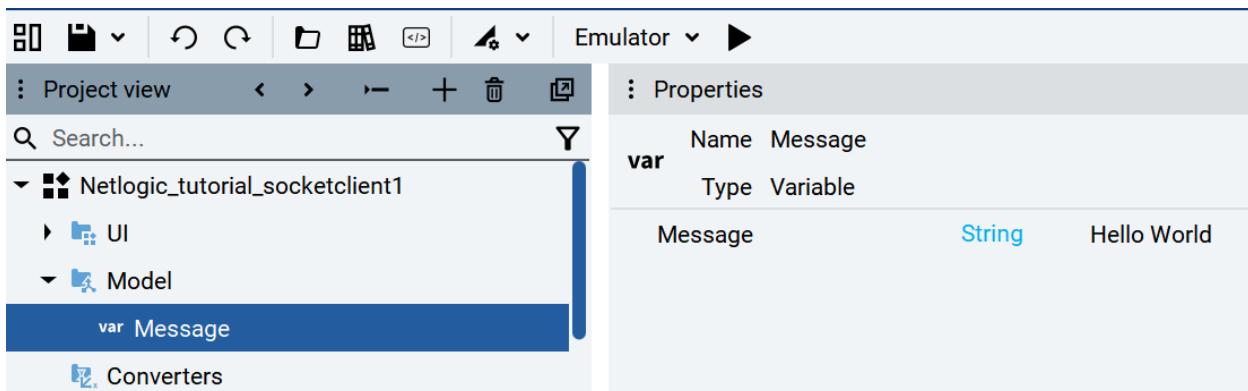
```
Log.Info("A button has been pressed");
```

Code	Timestamp	Category	Message
①	2024-01-02 09:09:37....	FTOptixRuntime	FactoryTalk Optix Runtime is currently using 1 feature token ↗ Component: Native presentation engine consumes 1 feature token ↗
①	2024-01-02 09:09:37....	FTOptixRuntime	Starting project Netlogic_tutorial_socketserver1
①	2024-01-02 09:09:47....		A button has been pressed
①	2024-01-02 09:10:06....		A button has been pressed
①	2024-01-02 09:12:13....	FTOptixRuntime	Project Netlogic_tutorial_socketserver1 successfully terminated

https://github.com/FactoryTalk-Optix/NetLogic_CheatSheet/blob/main/pages/log-output.md

31. 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
```

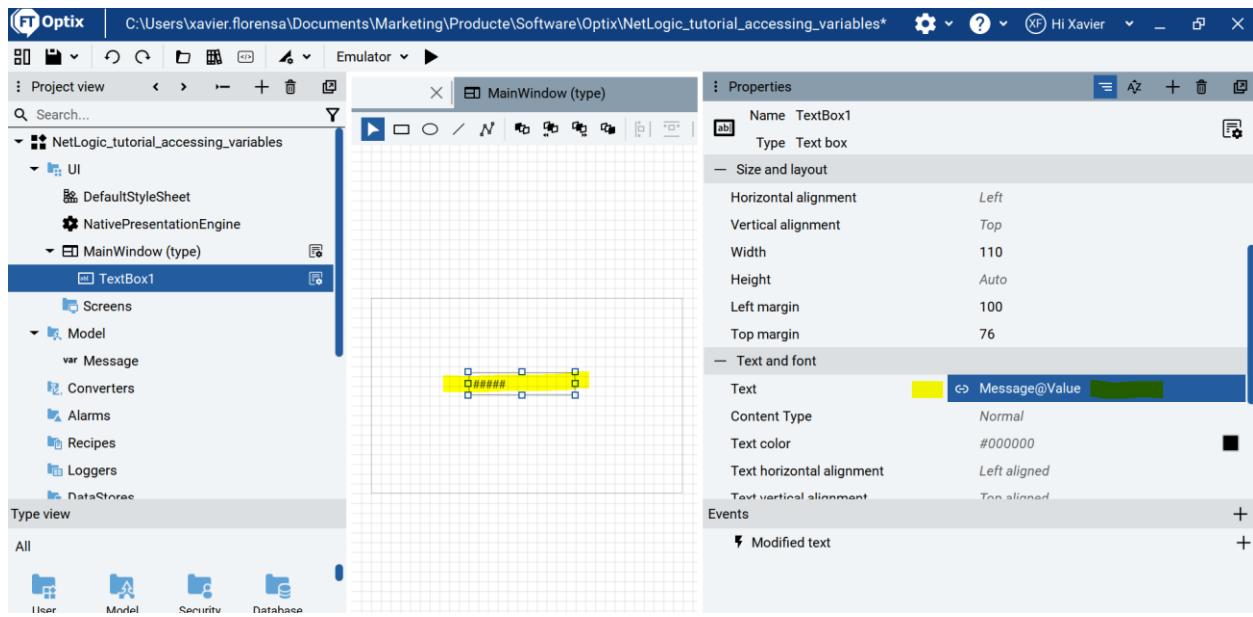
Or if you want to assign a new value in the code, you do this way

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

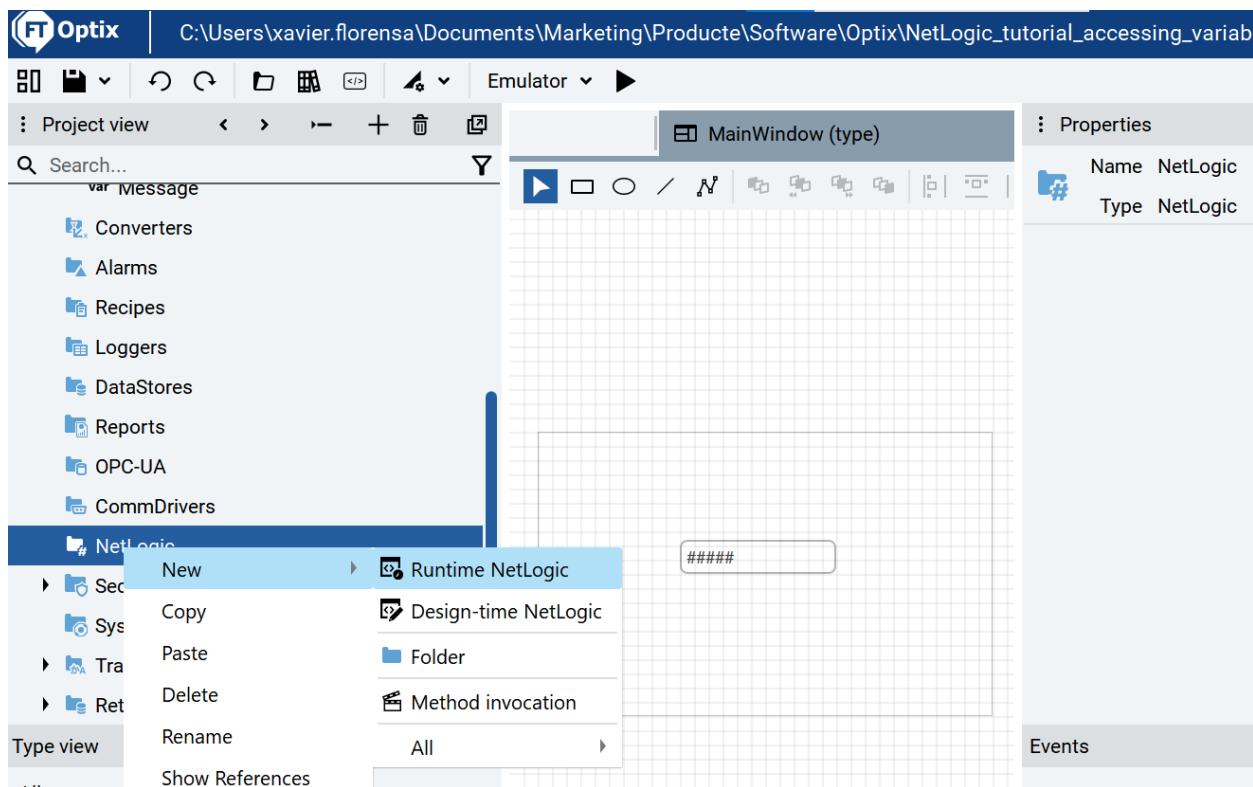
You can output the new code value to a User Interface Textbox with a dynamic link to the variable “Message”

Let's try it with this project

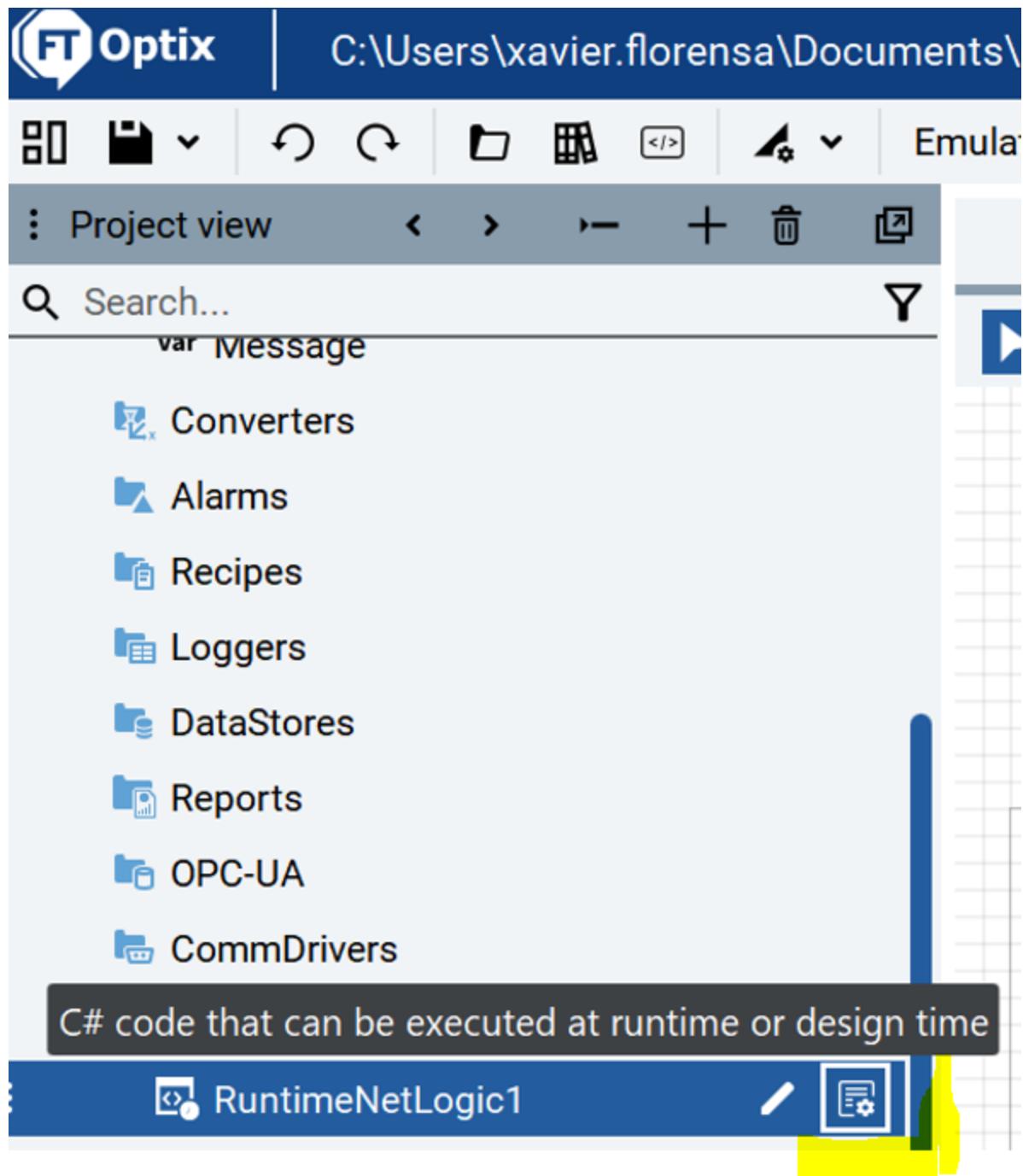
Let's create the dynamic link between a TextBox and a variable value



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 UAManagerCore;
using OpcUa = UAManagerCore.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

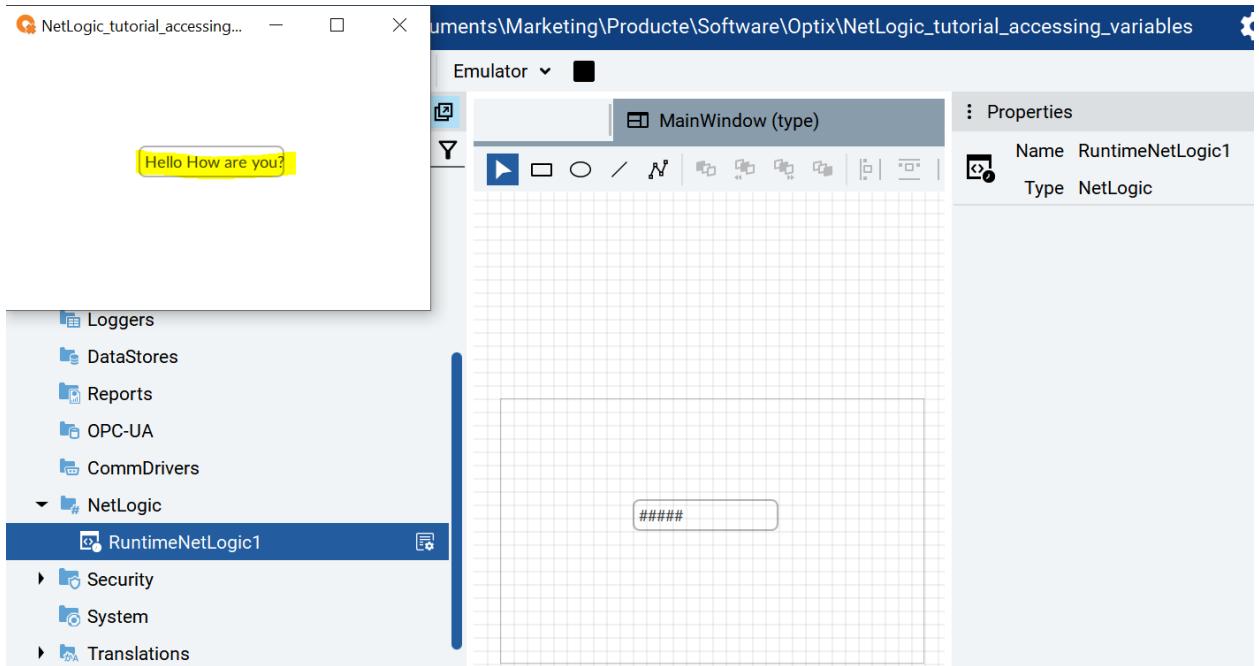
```

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
    }
}

```

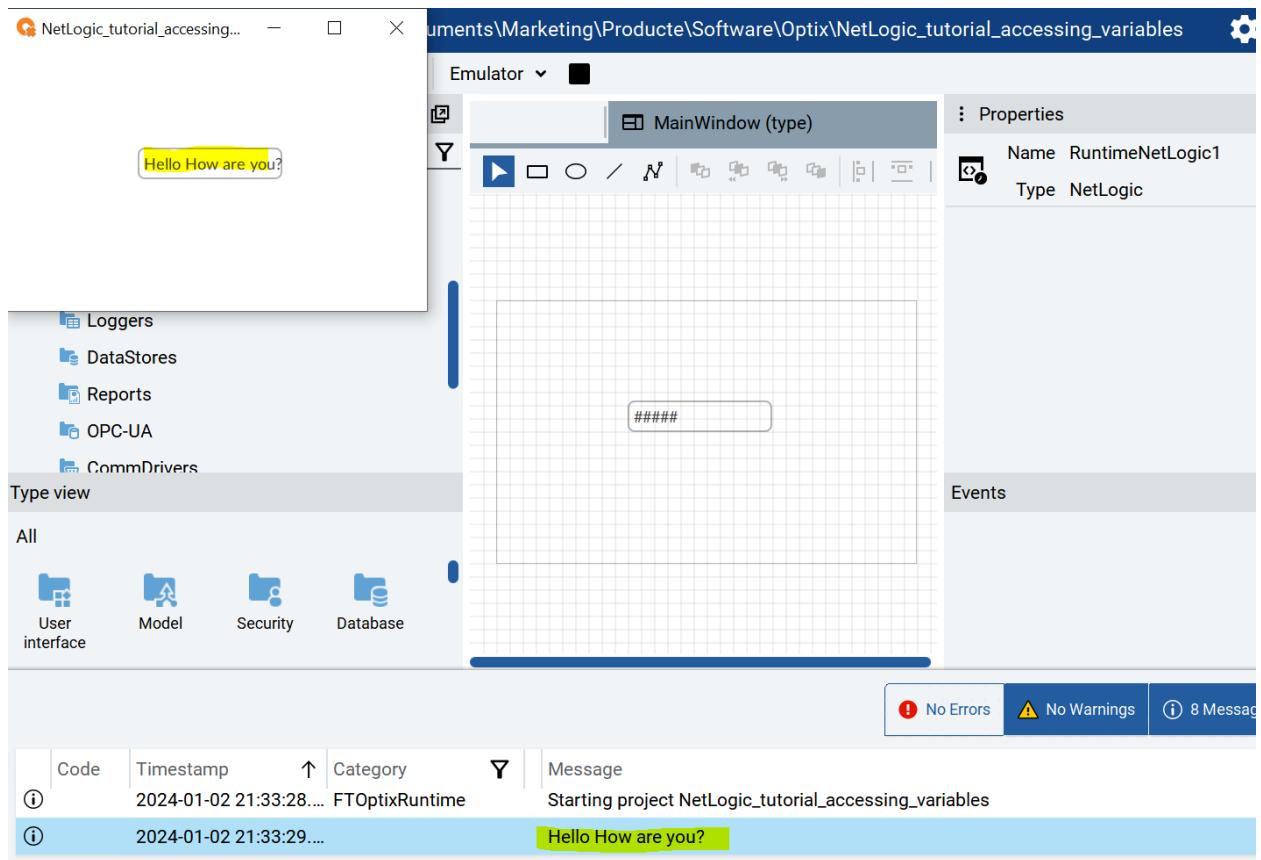
Save the code and test the emulator



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);
}
```

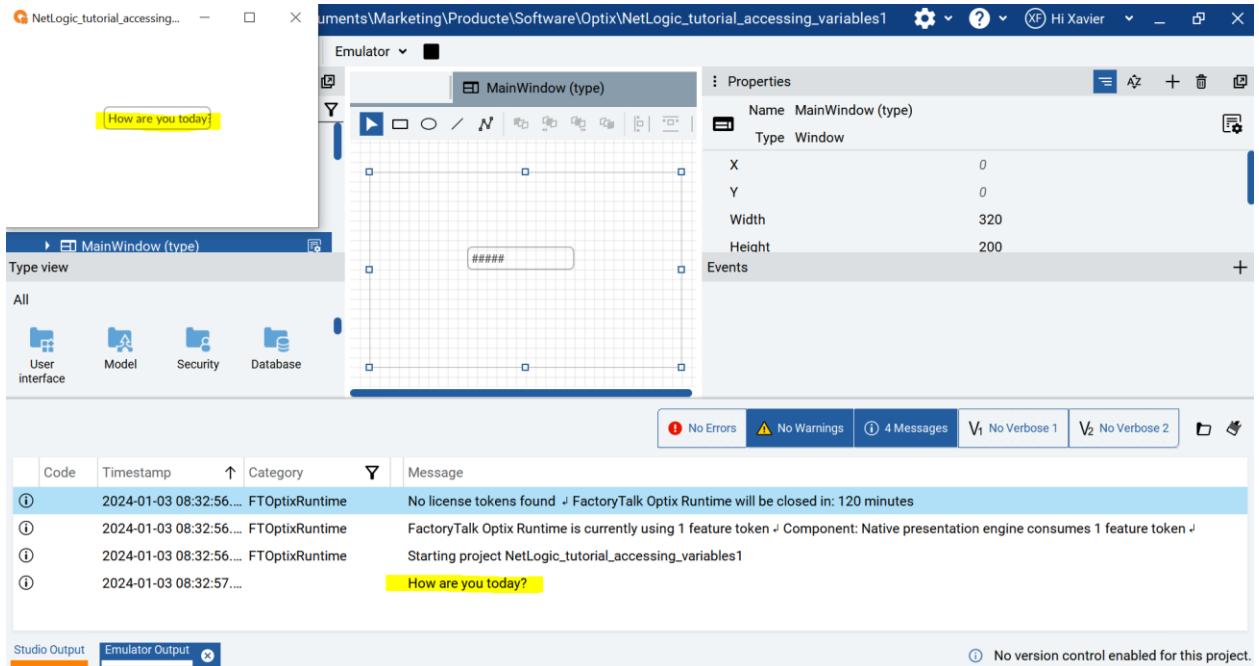


You can even access the UI variable without using another variable in the code, this way

```
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);
}
```



32. 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 Nod eid 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>

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 shows the FTOptix software interface. At the top, there is a toolbar with various icons. Below the toolbar is a navigation bar with tabs like 'Project view', 'Dashboard', and 'MainWindow (type)'. The main workspace is divided into two sections: a left pane for the UI structure and a right pane for the properties and code.

UI Structure (Left Pane):

- Training_NetlogicRuntime_Access_object
- UI
 - DefaultStyleSheet
 - NativePresentationEngine
 - MainWindow (type)
 - TextBox2
 - Label2
 - Screens
 - Model
 - Converters
 - Alarms
 - Recipes
 - Loggers

Properties (Right Pane):

Name	Type	Value
TextBox2	Text box	#000000
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		hello how are you?
Content Type		Normal

Code Editor (Bottom):

```
#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
{
    public override void Start()
```

```

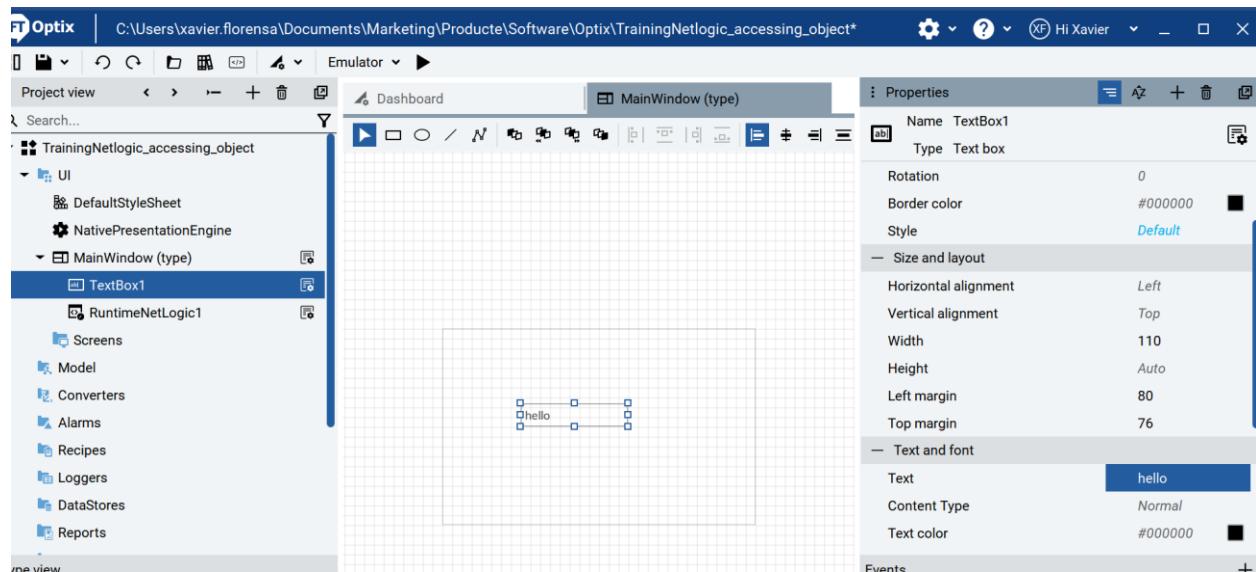
{
    var myTextbox = Project.Current.Get<TextBox>("UI/MainWindow/TextBox2");
    myTextbox.Text="hello how are you?";
}

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

```

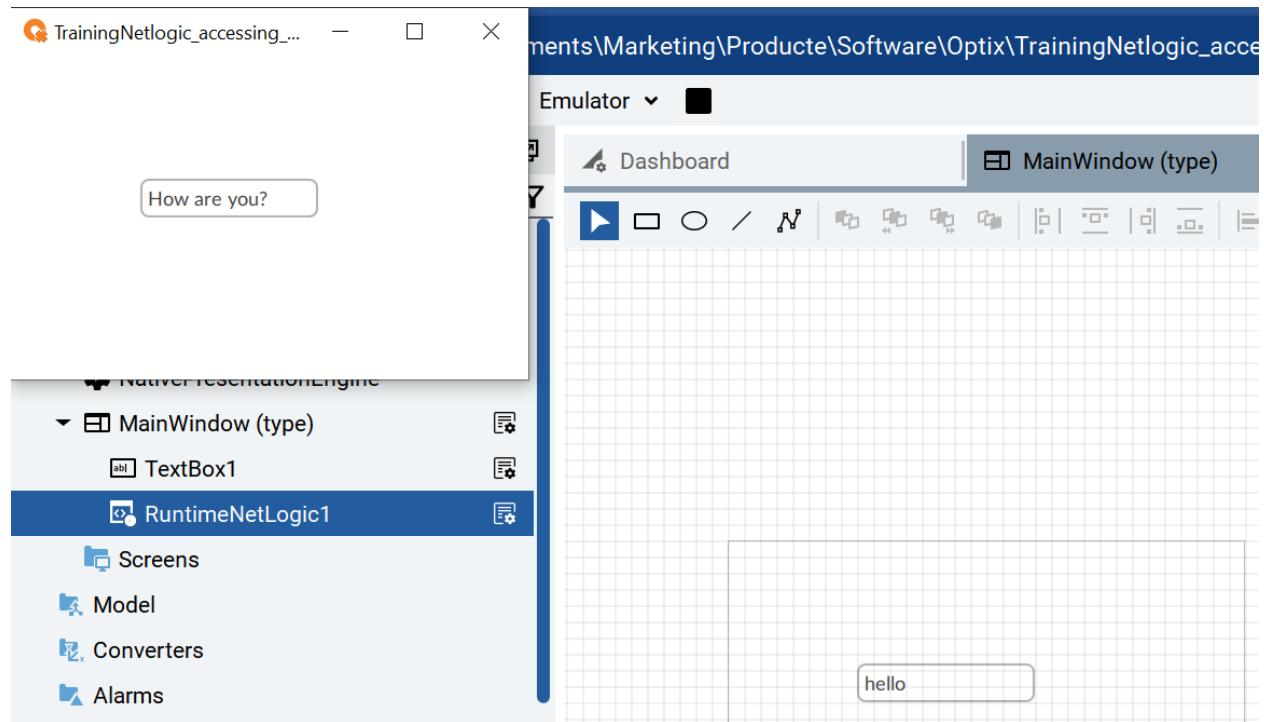
```

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.UAEvent

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

```
event EventHandler<UAEventArgs> UAEvent;
```

Event handler

```
public delegate void UAEvent(object sender, UAEventArgs e);
```

Event handler arguments

sender (object)

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

e (UAEventArgs)

A C# object that contains the following properties:

EventType (IUAObjectType)

The node of the type of event generated.

Arguments (UAEventArgumentList)

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

Example

```
public override void Start()
{
    var button1 = Owner.Get<Button>("Button1");
    button1.UAEvent += Button1_UAEvent;
}
```

```
private void Button1_UAEvent(object sender, UAEventArgs e)
{
    var label1 = Owner.Get<Label>("Label1");
    var button1 = (Button)sender;
    label1.Text = "Event on " + button1.BrowseName + " of type " + e.EventType.BrowseName + ", ";
}
```

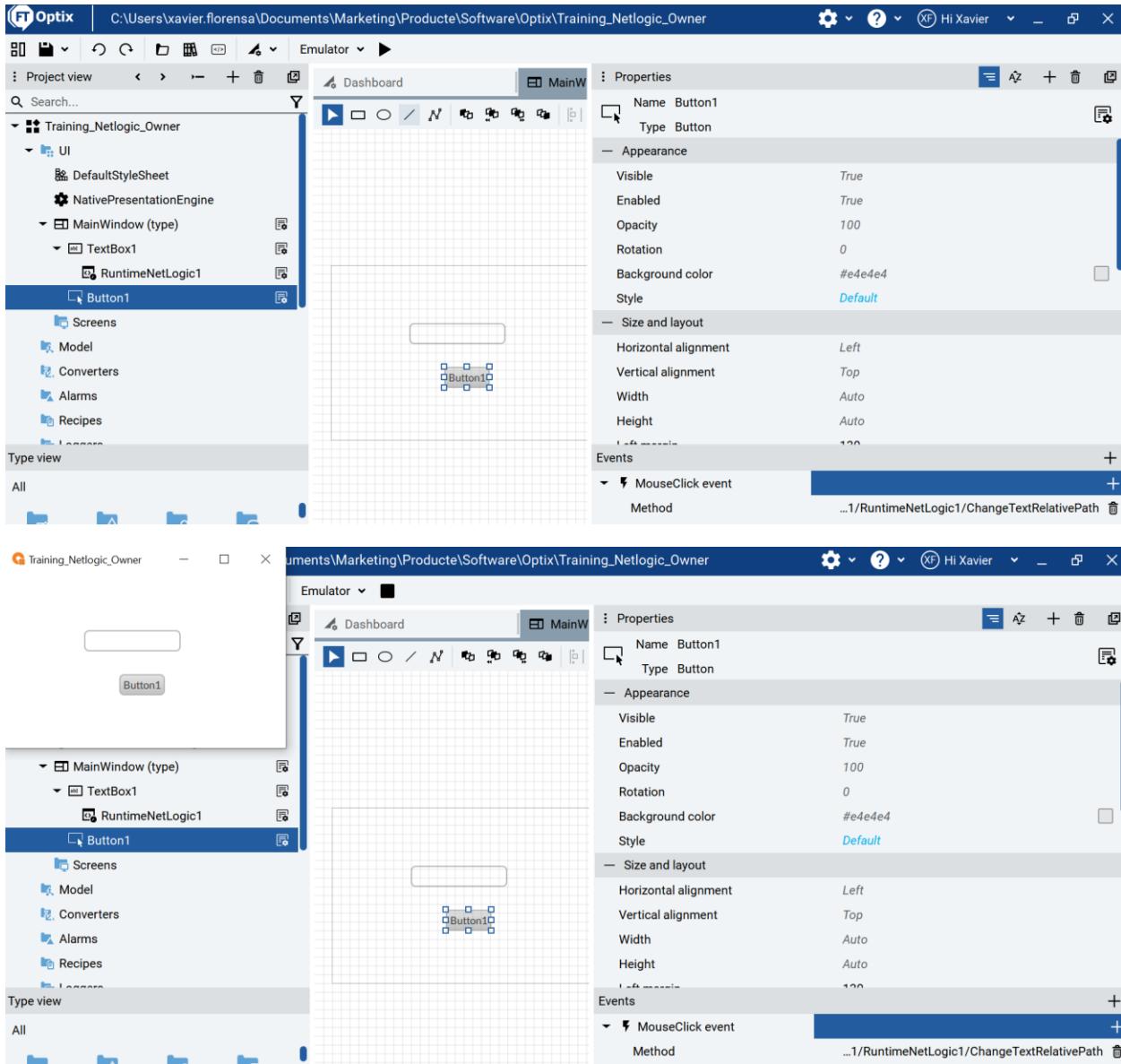
Create a project with a textbox

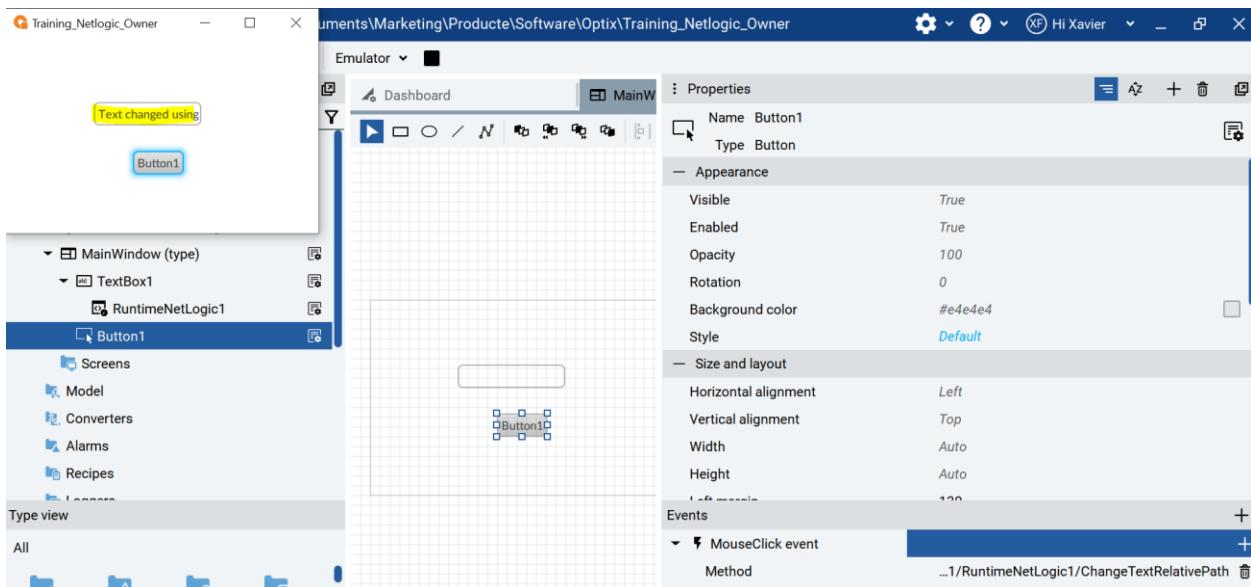
33. Using Owner

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

33.1. 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.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
    }

    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.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
        Owner.GetVariable("Text").Value = "Text changed using Owner";
    }

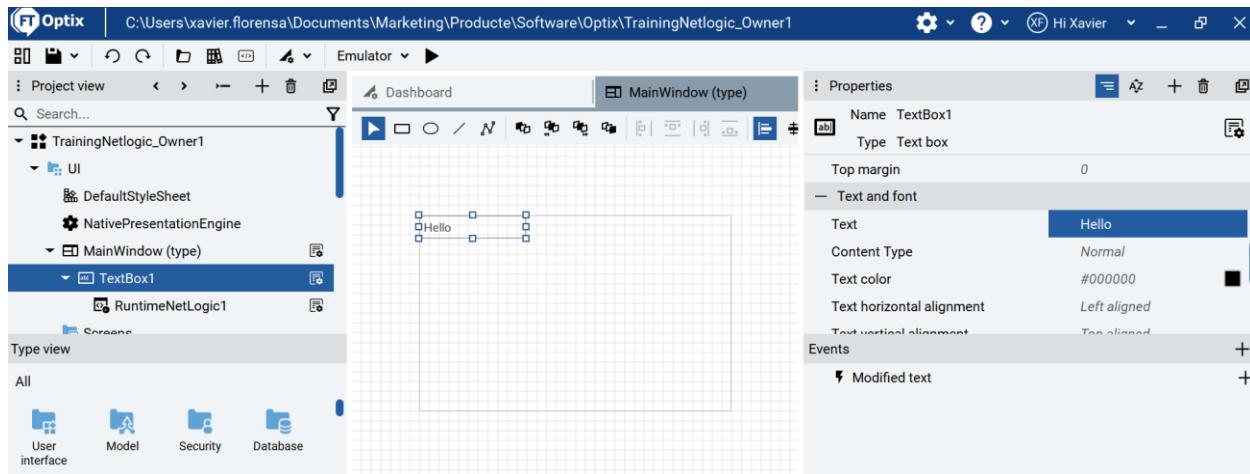
    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }

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

```

33.2. Detecting change on object using Owner

Create a Textbutton and insert a Runtime Netlogic on it



```
#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 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
    }
}
```

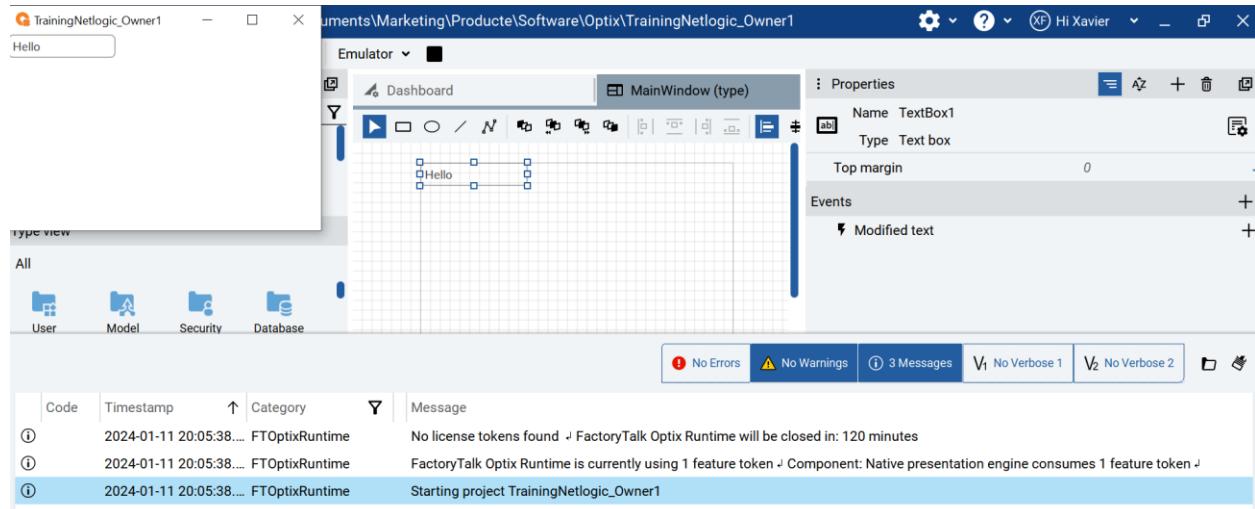
```

        Log.Info("Old value: " + e.OldValue.ToString());
        Log.Info("New value: " + e.NewValue.ToString());
    }

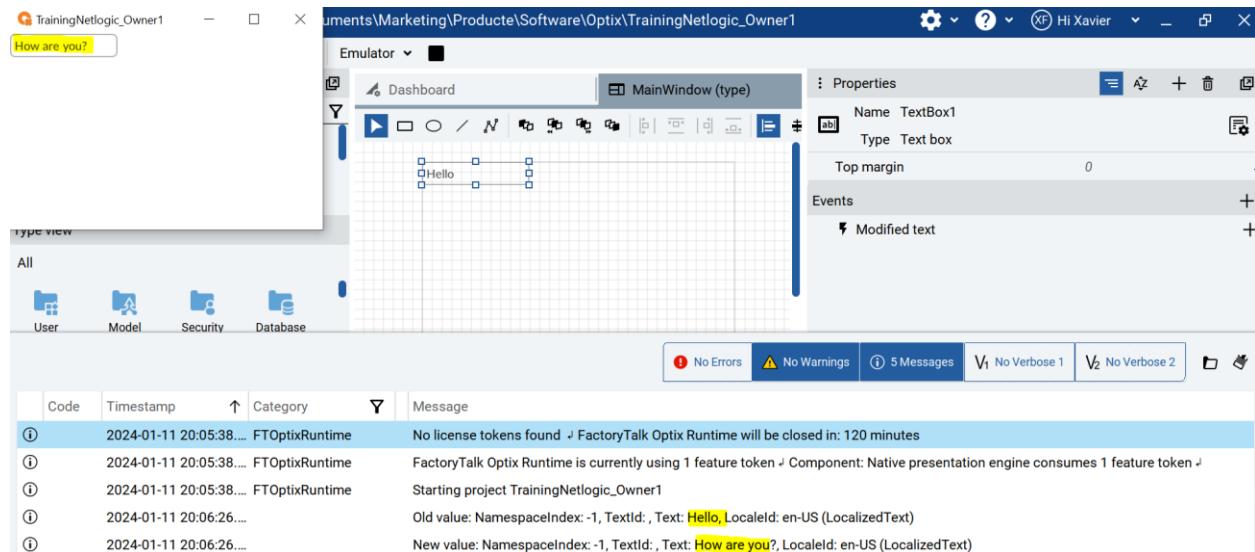
}

```

Execute the application



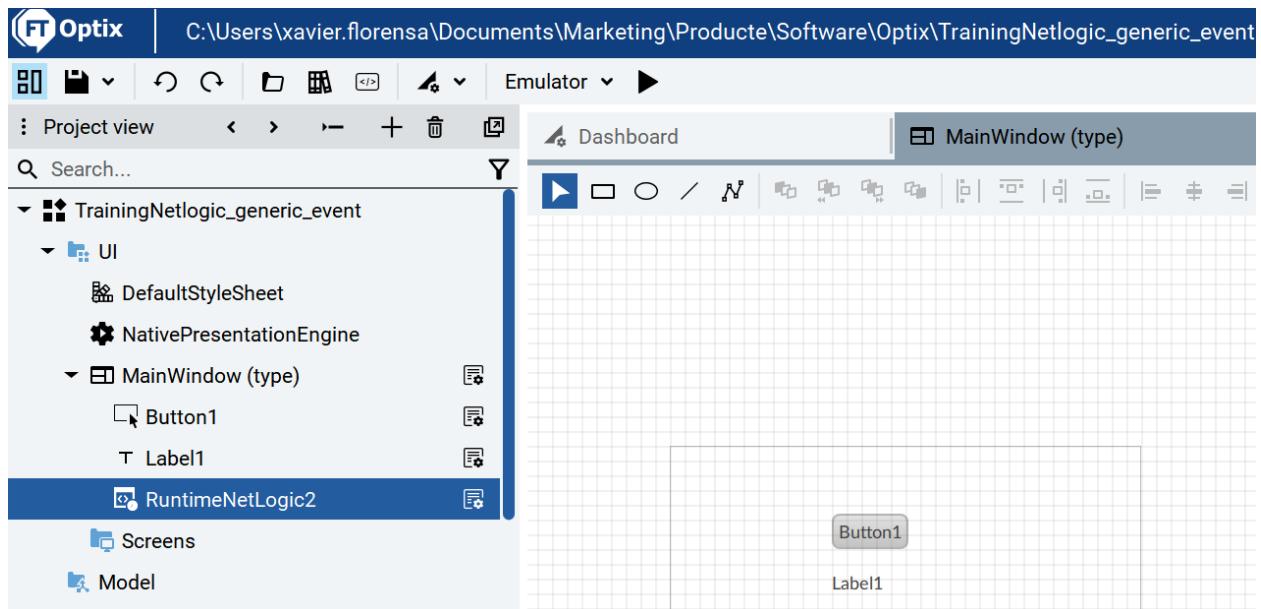
Change the text and hit enter



34. 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.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.UAEvent += 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)
```

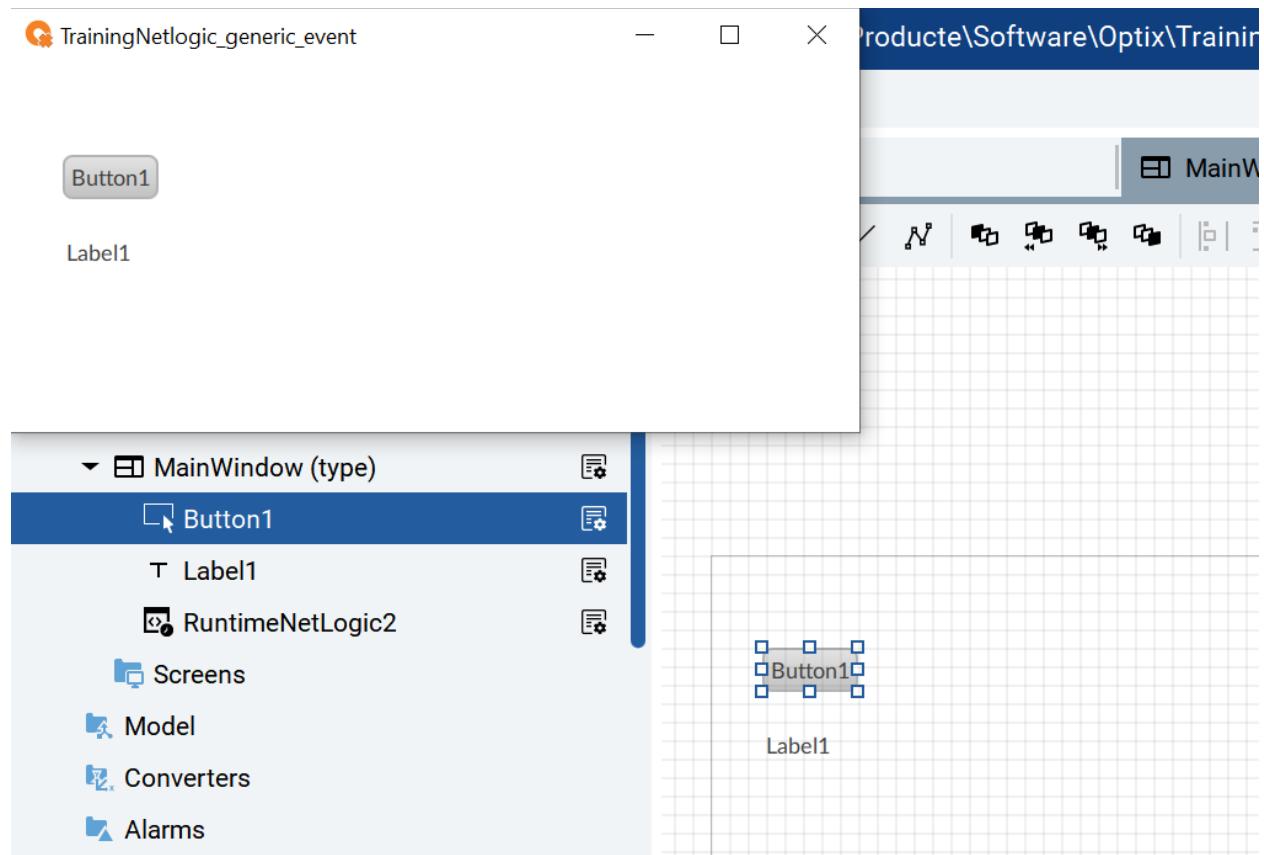
```

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

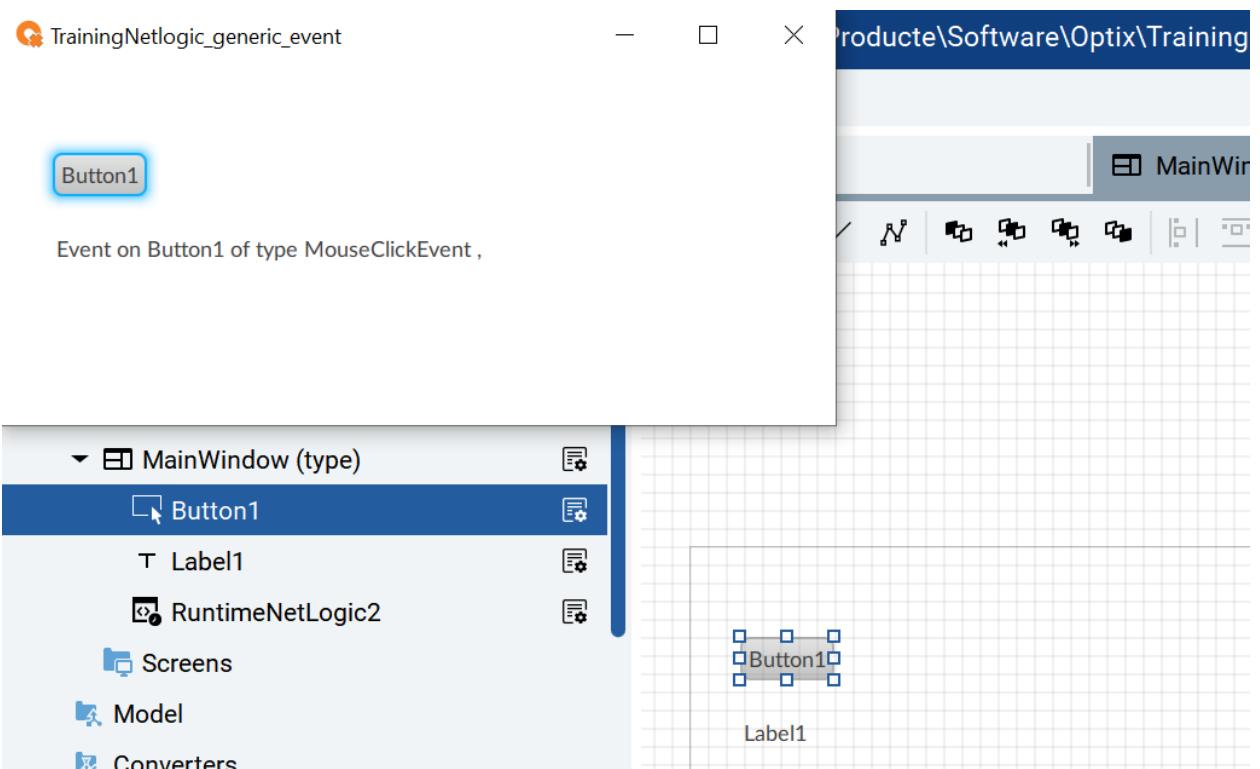
```

Execute the application

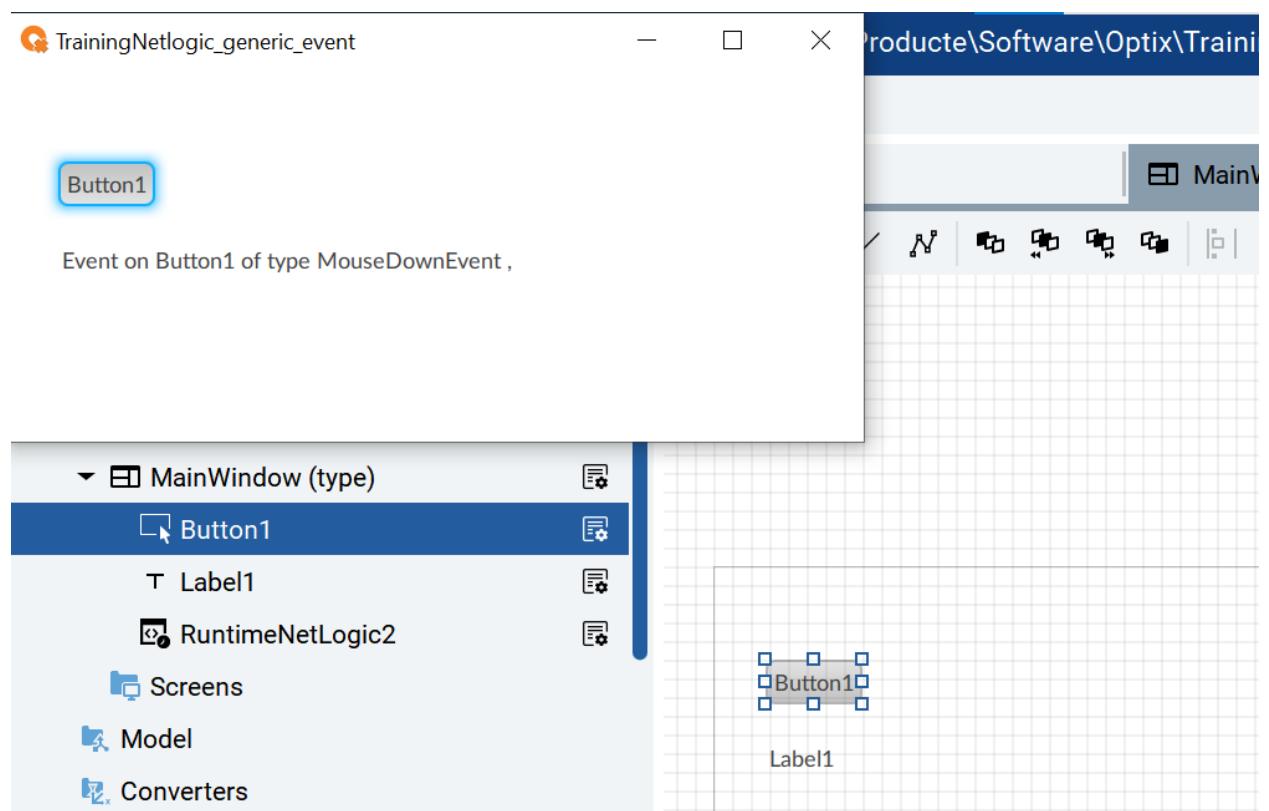
Click the button



Click on the button

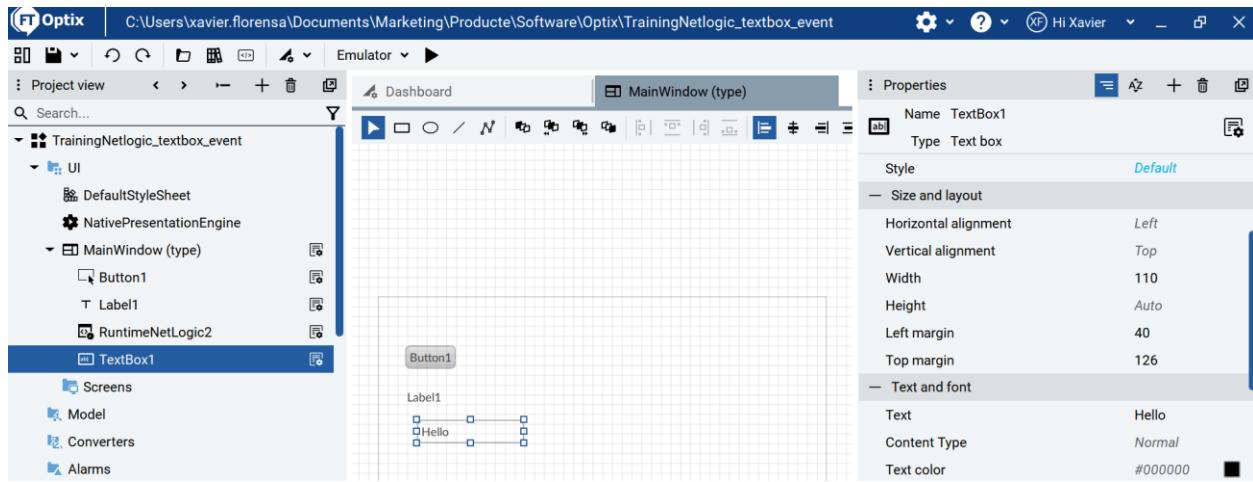


Hold pressing the button



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.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;
        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
    }
}
```

```

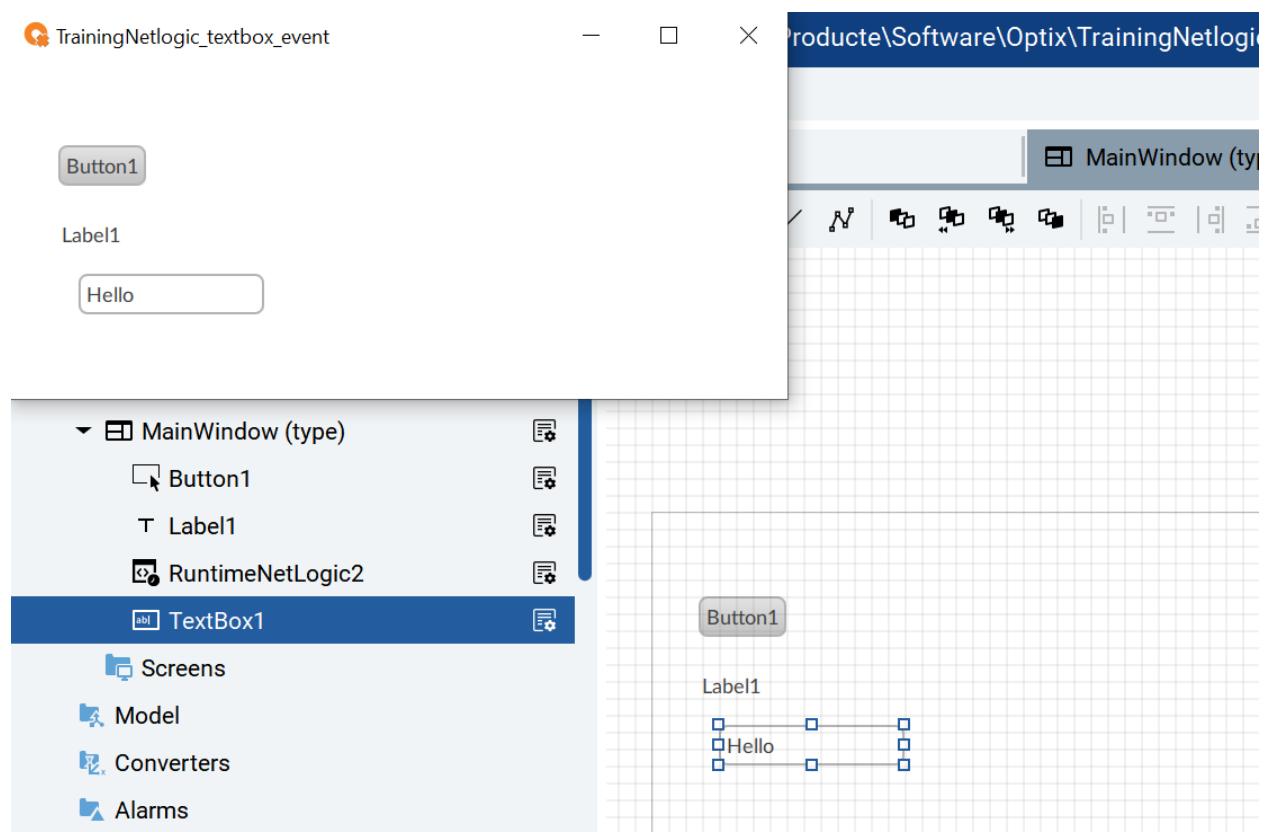
private void Button1_UAEEvent(object sender, UAEEventArgs e)
{
    var label1 = Owner.Get<Label>("Label1");
    var button1 = (Button)sender;
    label1.Text = "Event on " + button1.BrowseName + " of type " +
e.EventType.BrowseName + " , ";
}

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 + " , ";
}

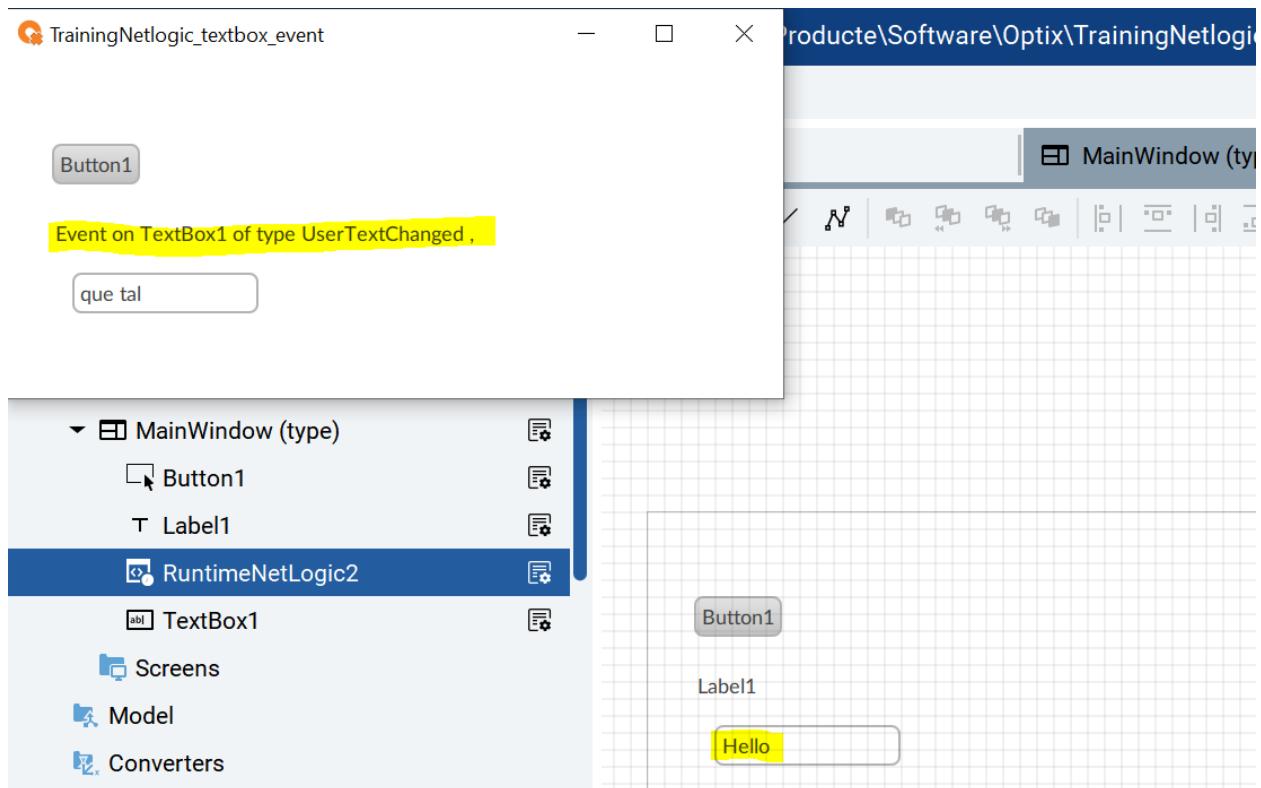
}

```

Run the project

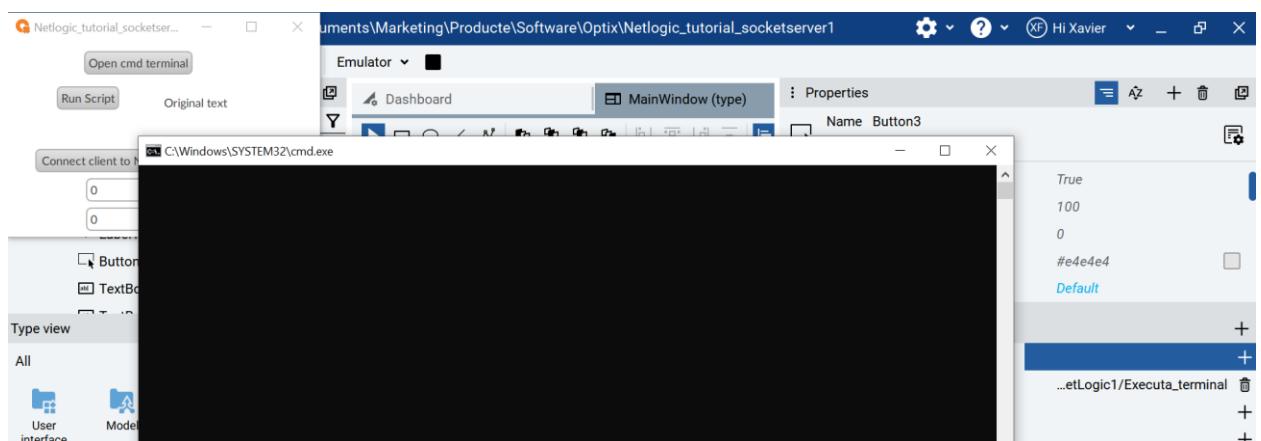


Now change the text and hit enter



35. 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 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.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;
    }
}

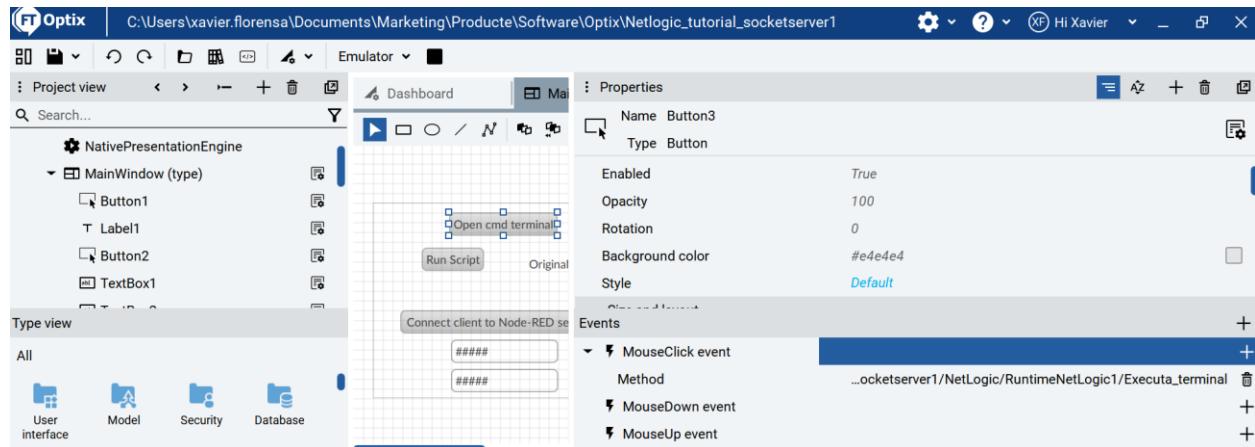
```

```

        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



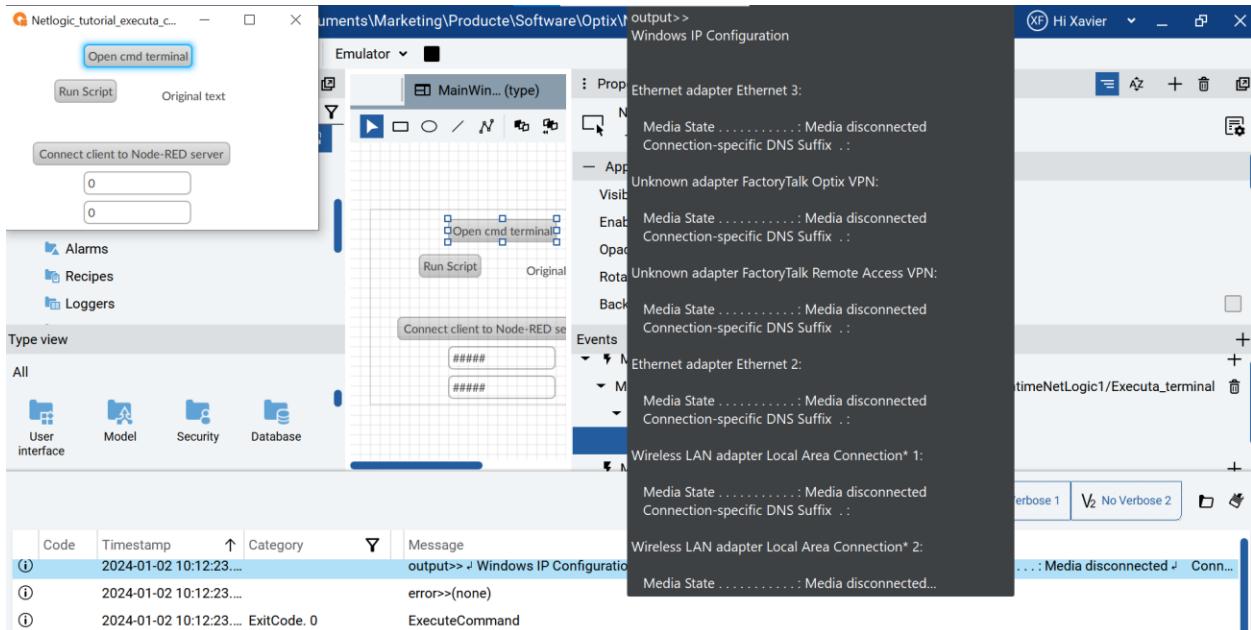
That's all.

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;

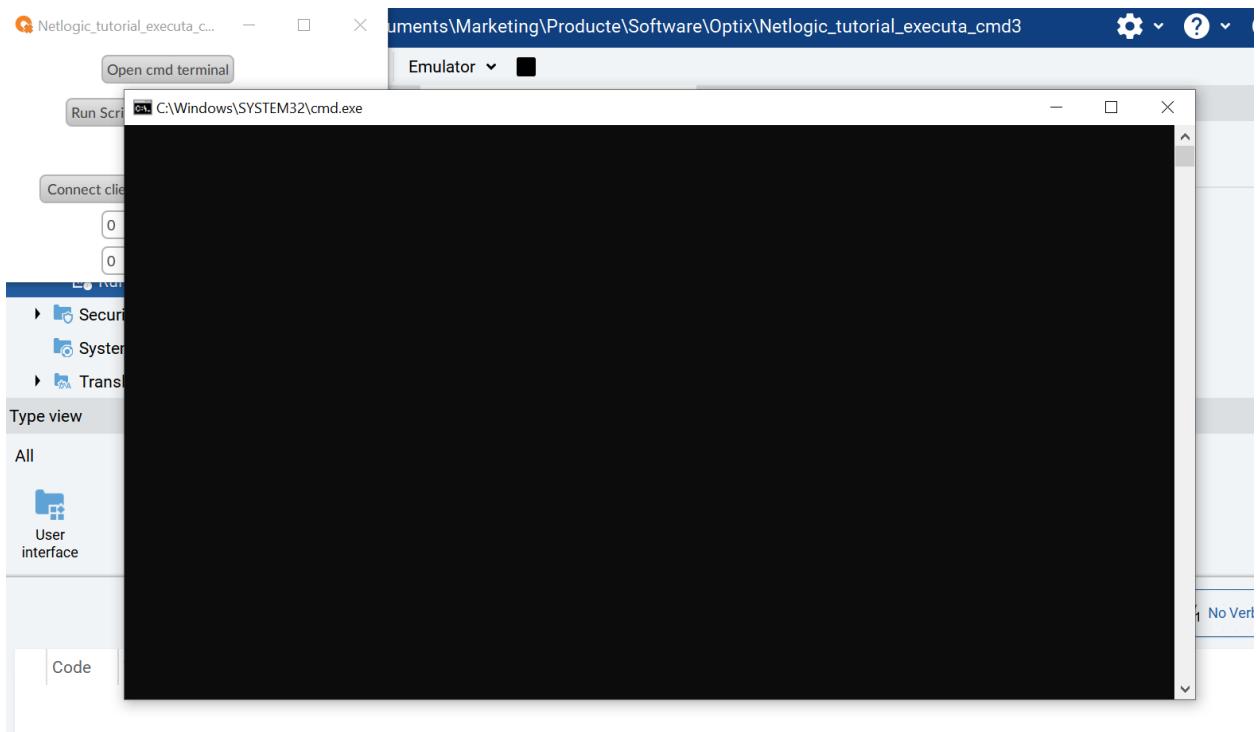
    processInfo = new ProcessStartInfo("cmd.exe", "/c " + command);
```

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

The screenshot shows a web browser window with the URL stackoverflow.com/questions/515309/what-does-cmd-c-mean. The page content is a question titled "What does cmd c mean?" with 335 answers. The left sidebar shows navigation links like Home, Questions (which is selected), Tags, Users, Companies, COLLECTIVES, LABS, and TEAMS. The main content area displays the command line syntax for 'cmd /?' and a detailed explanation of each parameter:

```
C:\>cmd /?
Starts a new instance of the Windows XP command interpreter

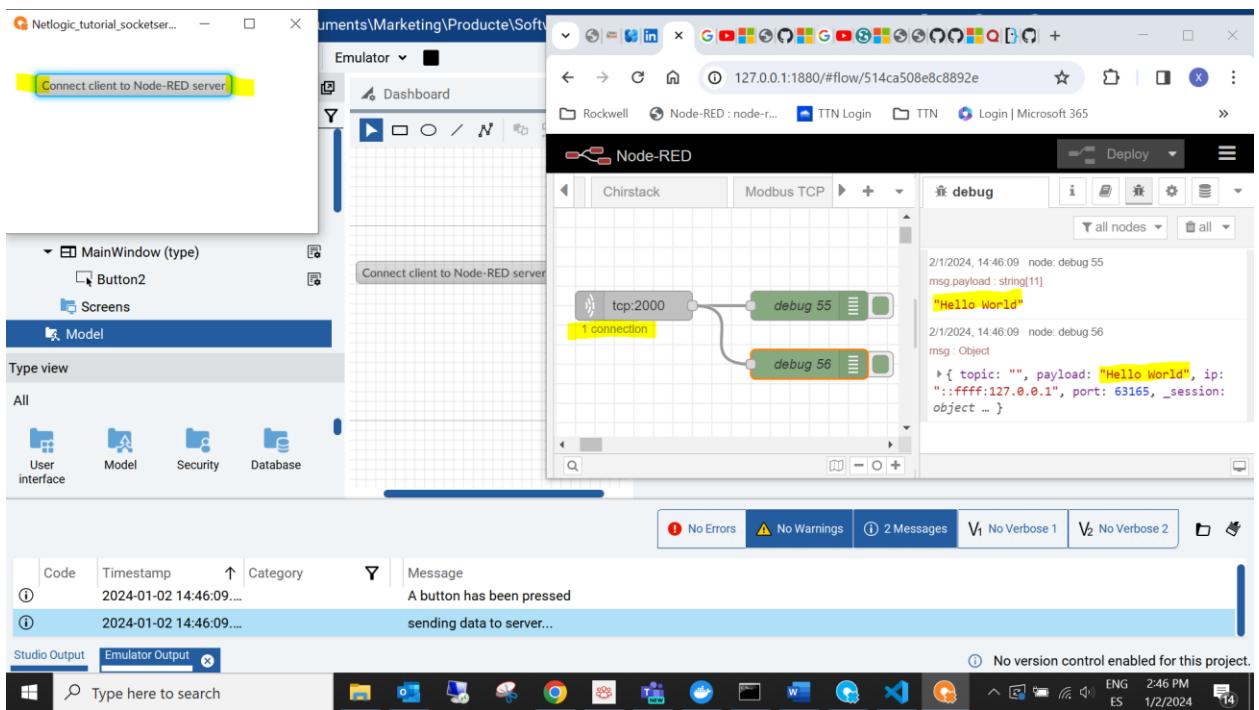
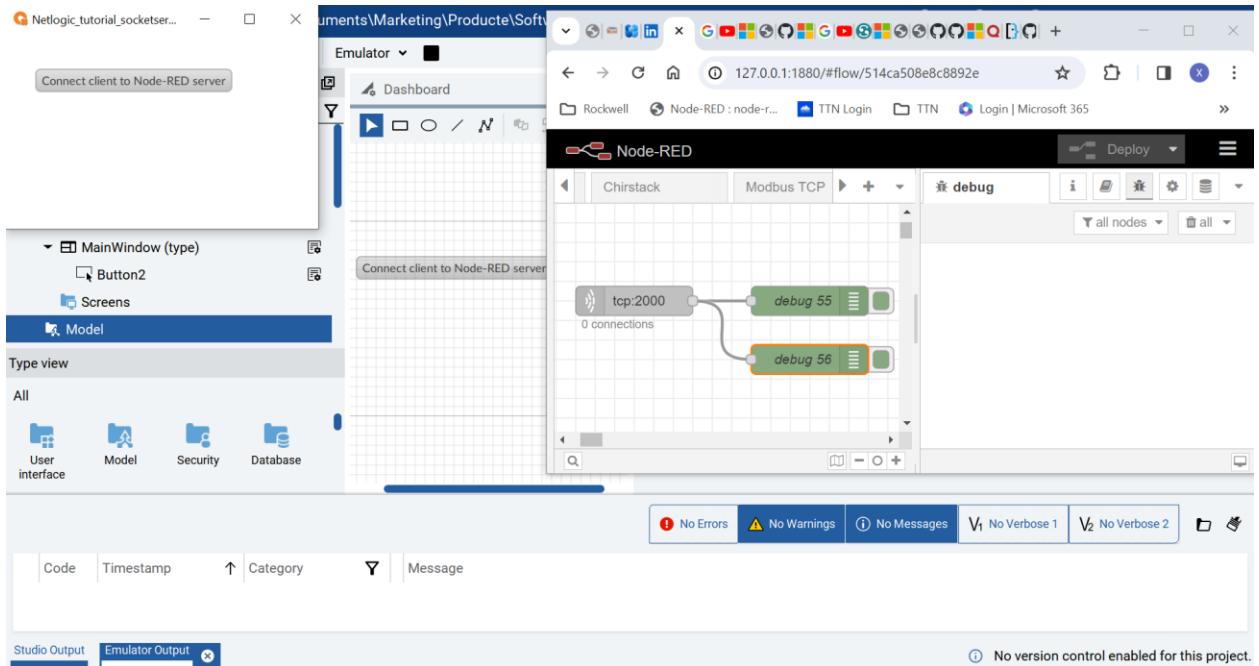
CMD [/A | /U] [/Q] [/D] [/E:ON | /E:OFF] [/F:ON | /F:OFF] [/V:ON | /V:OFF]
[[/S] [/C | /K] string]

/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
```

36. Socket TCP using System.Net.Sockets without external libraries

We have a socket server in Node-RED listening to port 2000.

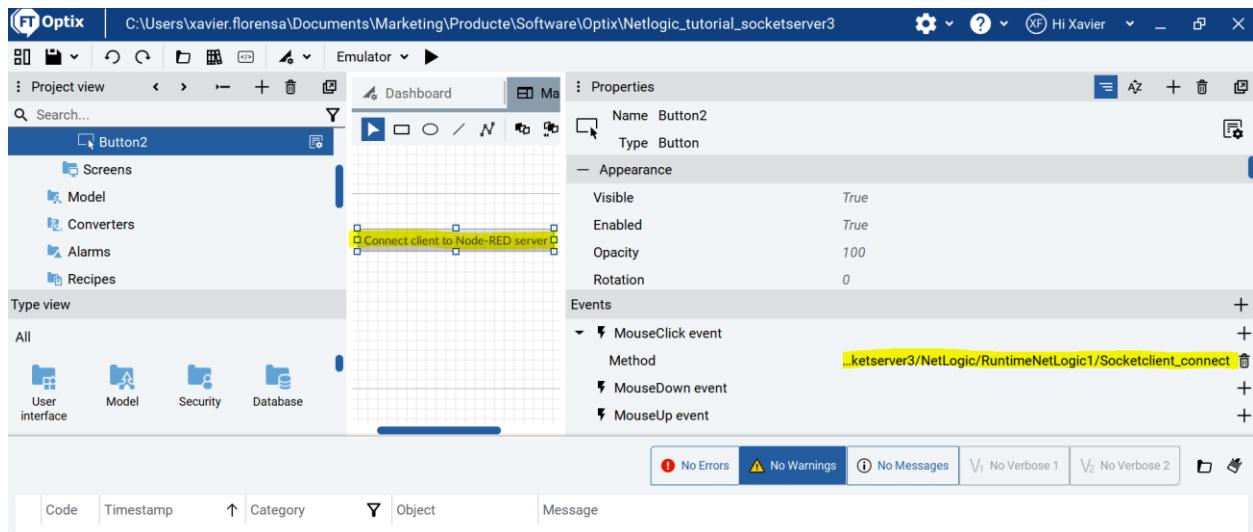
Let's develop an application in FTOptix to connect to the server and to send a message "Hello World".



Using code pasted from this example

<https://www.youtube.com/watch?v=g5yEWLJxNml&t=11s>

In this way inside Netlogic, using a Method



```
#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 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...");
    }
}
```

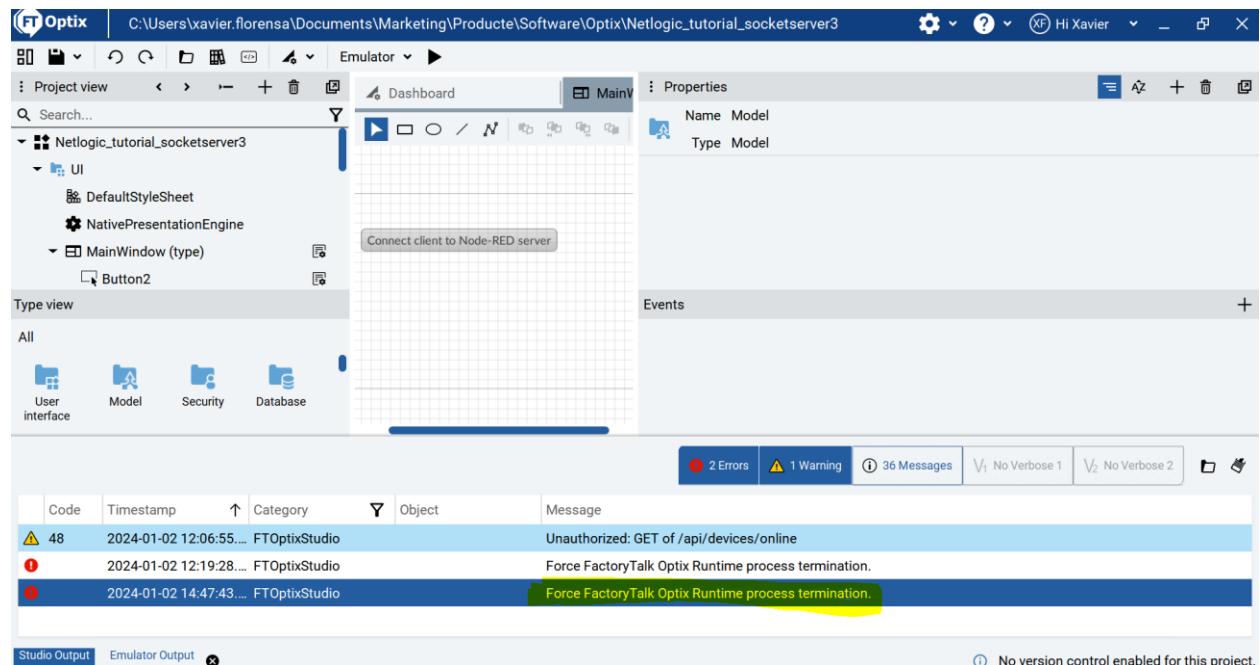
```

        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.

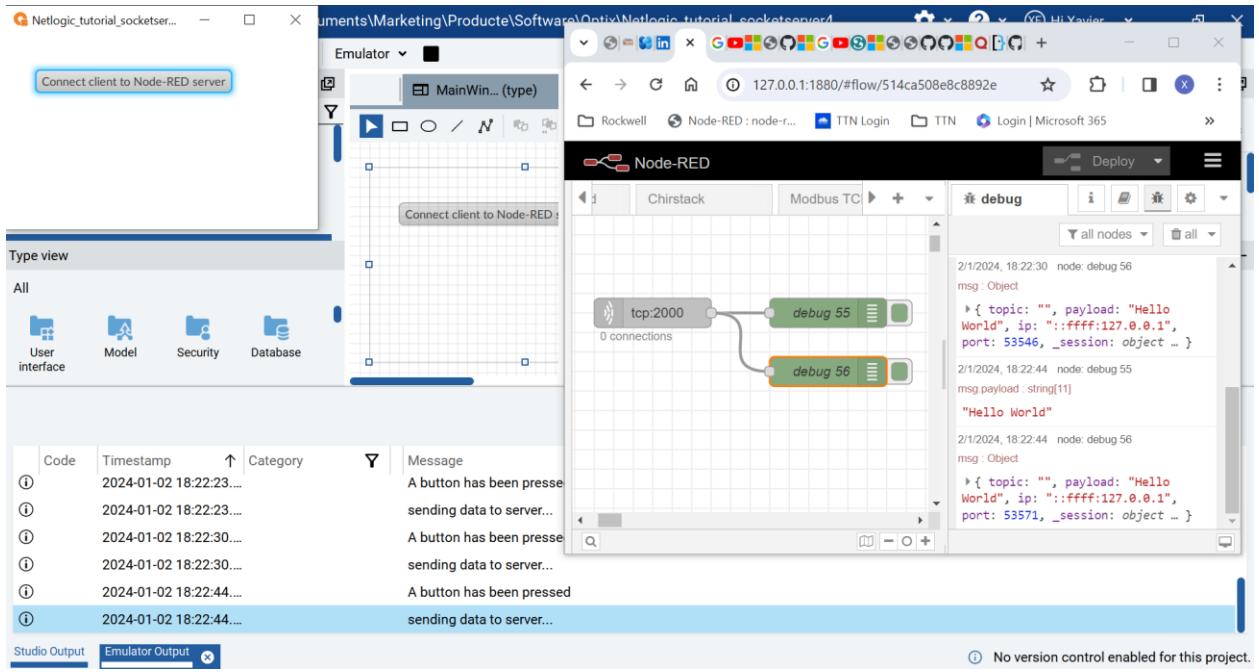


We can have better results if we comment the lines related to receiving data

```
0 references
18 public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
19 {
20     [ExportMethod]
21     0 references
22     public void ...Socketclient_connect()
23     {
24         Log.Info("A button has been pressed");
25         TcpClient client = new TcpClient("127.0.0.1", 2000);
26         string messageToSend = "Hello World";
27         int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
28         byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
29         NetworkStream stream = client.GetStream();
30         stream.Write(sendData, 0, sendData.Length);
31         //Console.WriteLine("sending data to server...");
32         Log.Info("sending data to server...");
33         //StreamReader sr = new StreamReader(stream);
34         //string response = sr.ReadLine();
35         //Console.WriteLine(response);
36         //Log.Info(response);
37         stream.Close();
38         client.Close();
39         //Console.ReadKey();
40     }
}
```

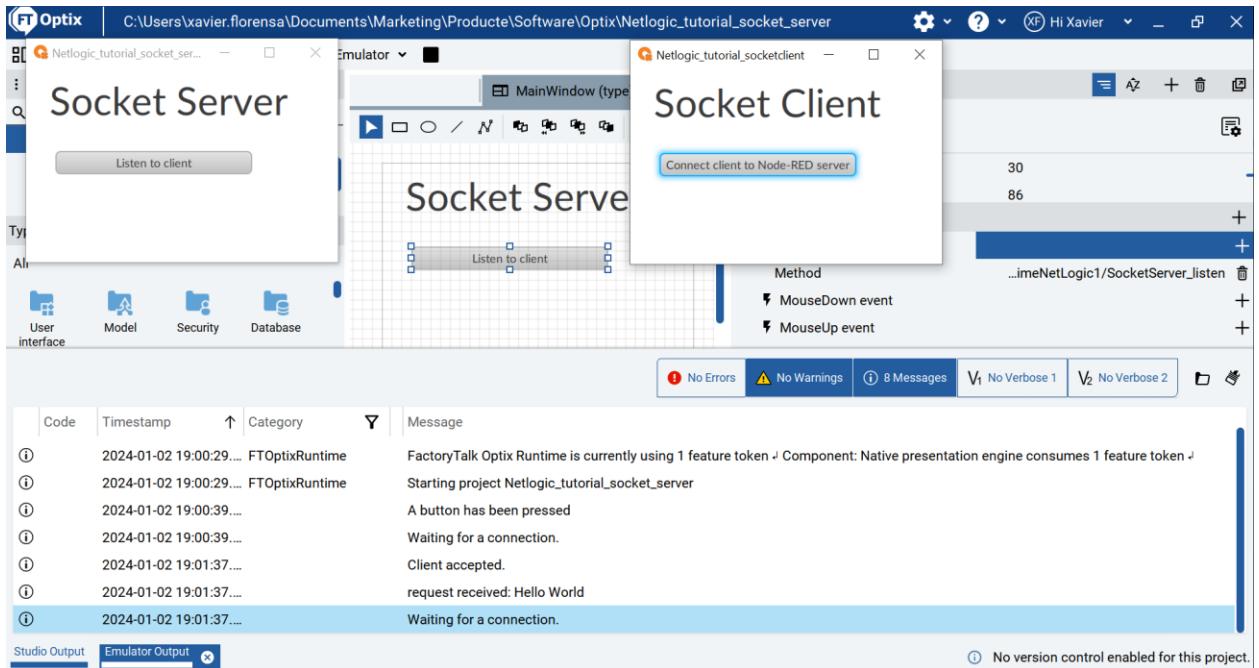
Now you can stop the program with the stop key

And you can execute several times the send button. But you will not see any connected client number on server side.



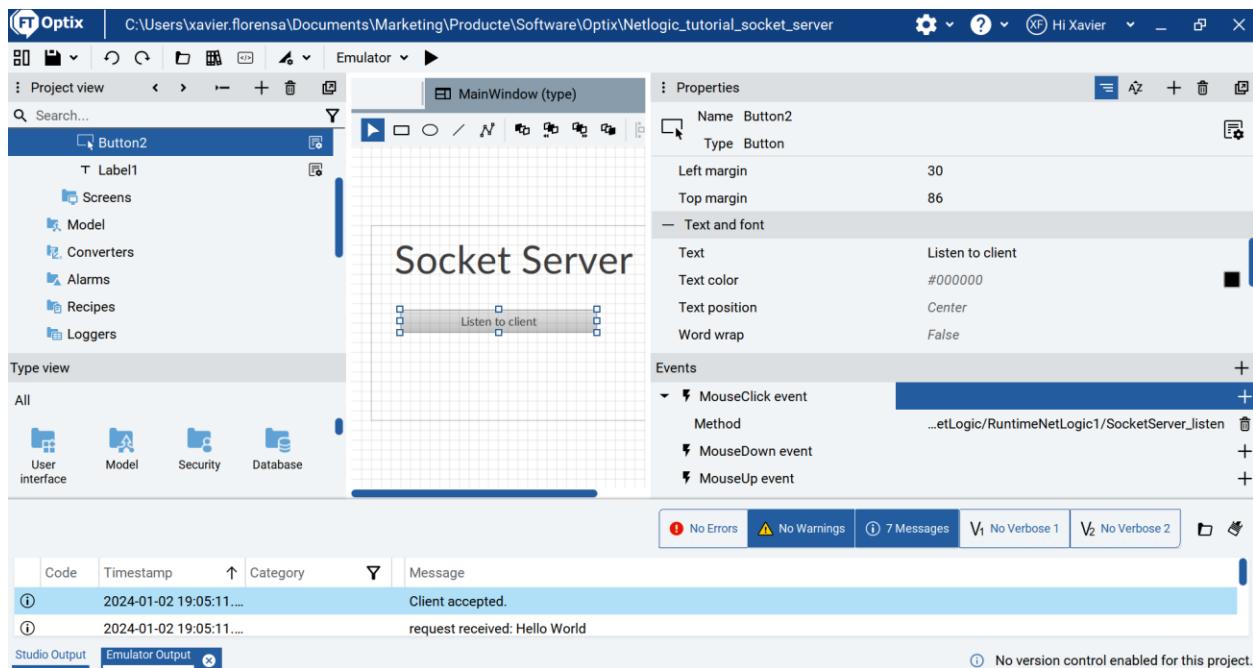
Now let's try to write an Optix application as a Socket server.

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 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()
    {
        Log.Info("A button has been pressed");
        TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
        listener.Start();
    }
}
```

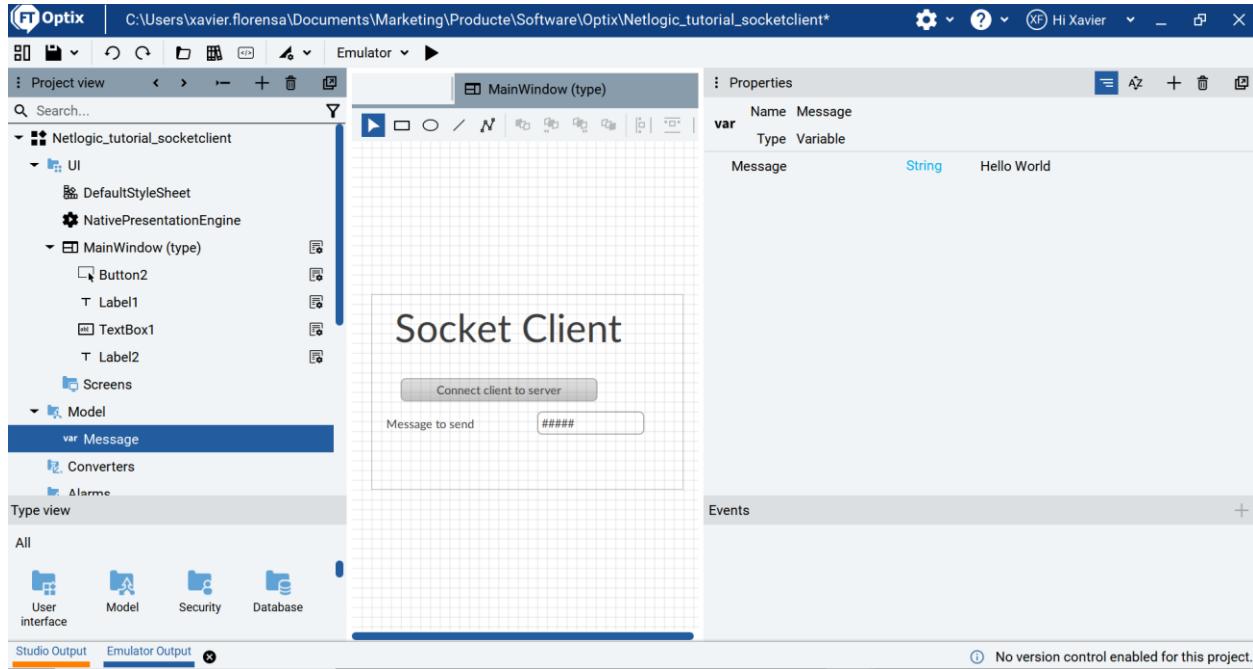
```
while (true)
{
    //Console.WriteLine("Waiting for a connection.");
    Log.Info("Waiting for a connection.");
    TcpClient client = listener.AcceptTcpClient();
    //Console.WriteLine("Client accepted.");
    Log.Info("Client accepted.");
    NetworkStream stream = client.GetStream();
    StreamReader 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

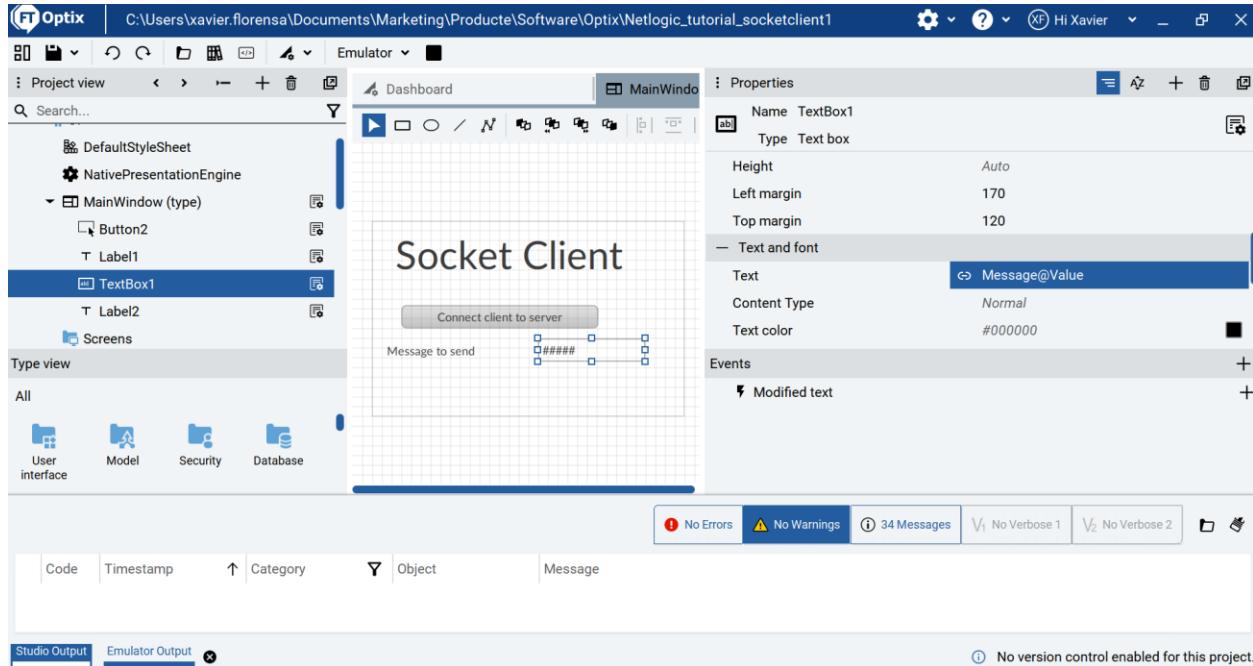
First of all let's introduce the text message to send manually from a Text Box

Let's go to client and add a Message string variable.

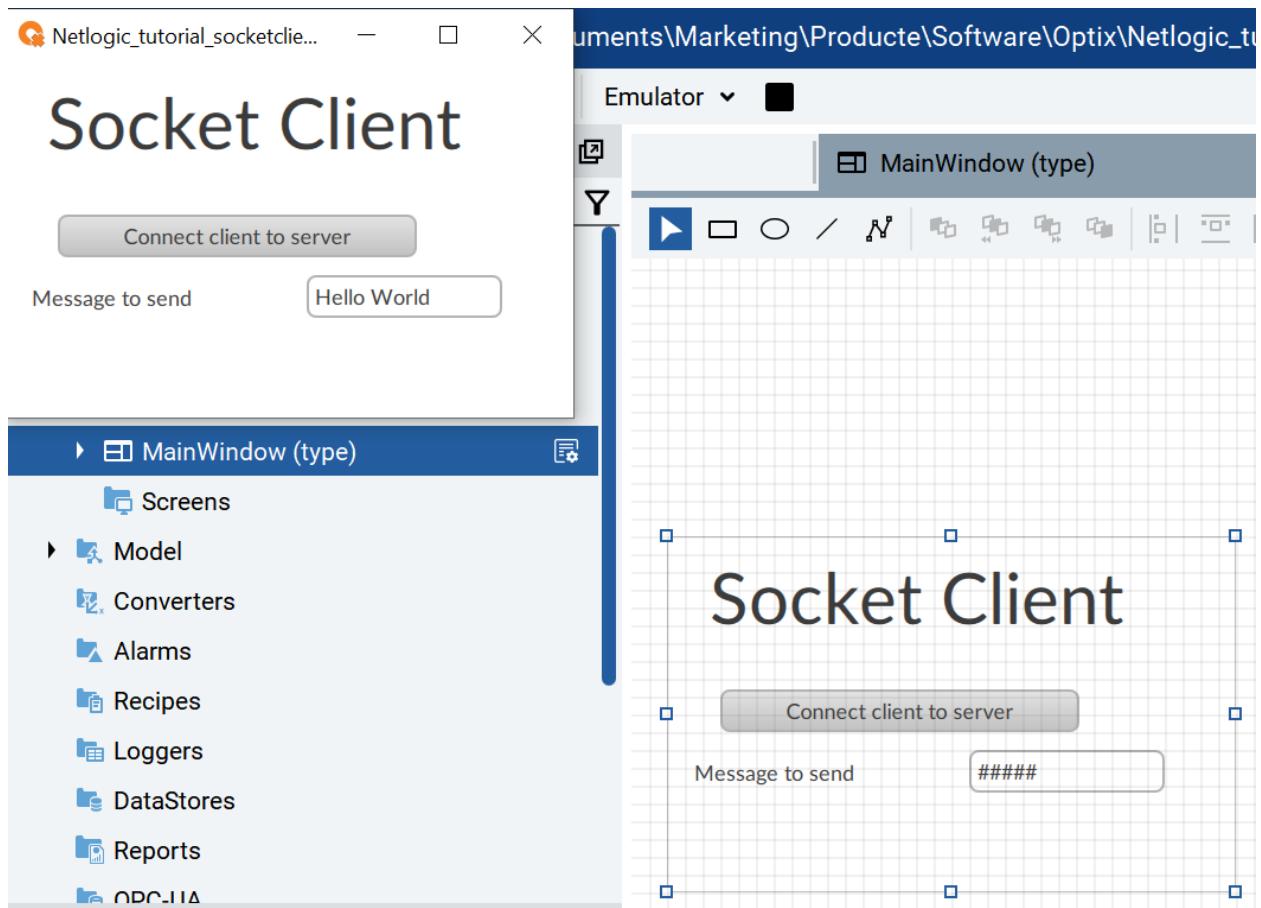
We use a default variable value “Hello World”



And a Textbox with a dynamic link to variable Message



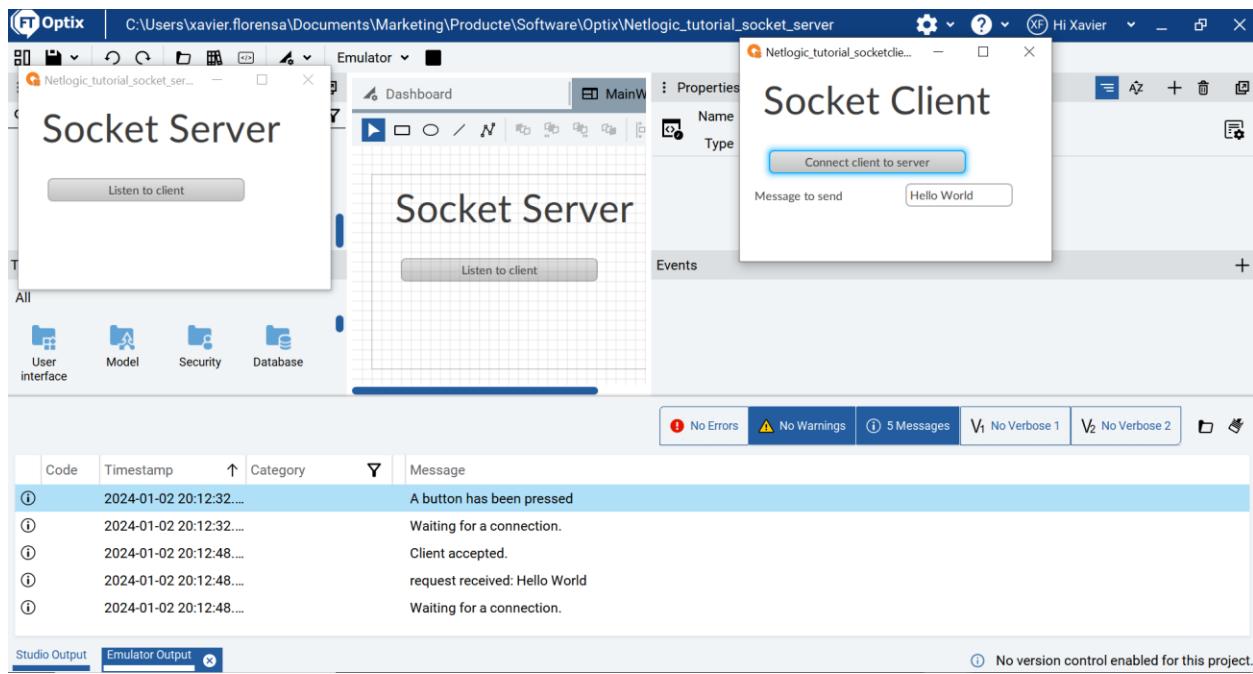
If we execute the application we will see the default value



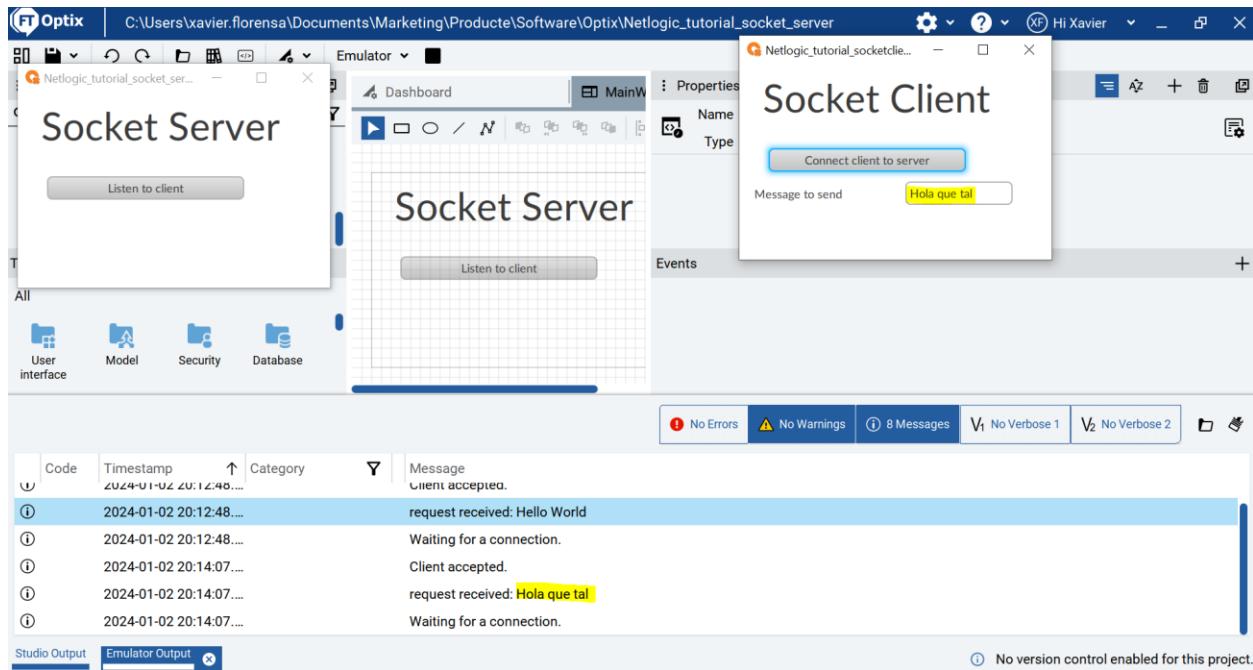
But we have to pass that string value to our code, by means of the UI variable Message

```
[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

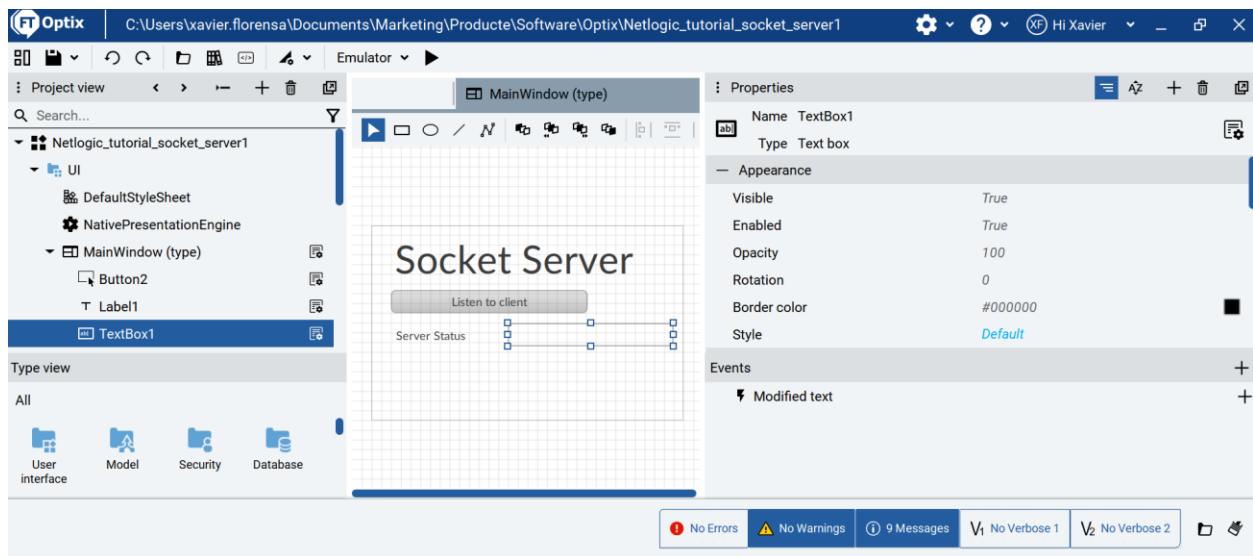


And we can send different messages



Let's improve the application with a status window on the server

Let's create a new Textbox



We will use the Nodeld of the Textbox to be able to display text on it.

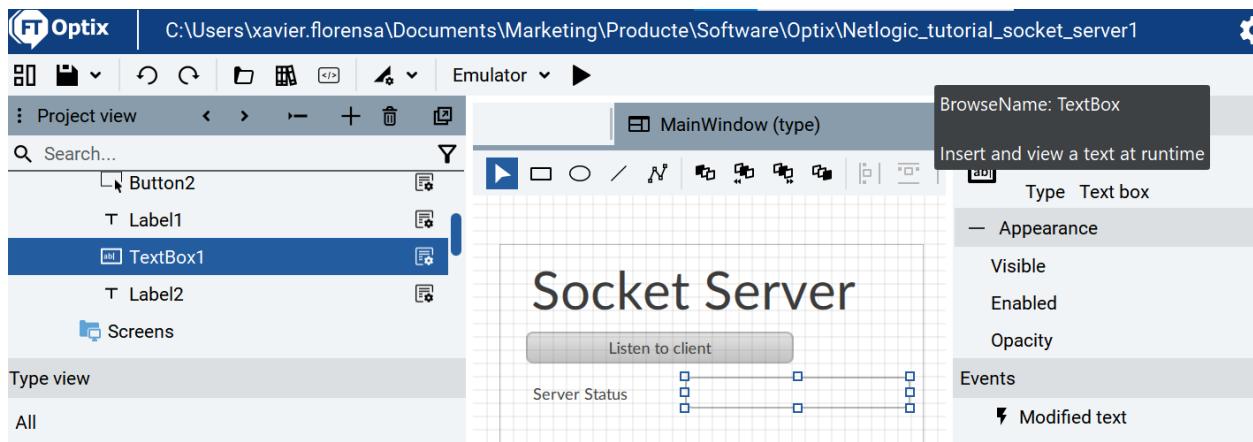
It's easy to use the Method we have already created to pass an argument as the Textbox Nodeld

First of all we have to change the Method to require an input parameter, the Nodeld.

This way

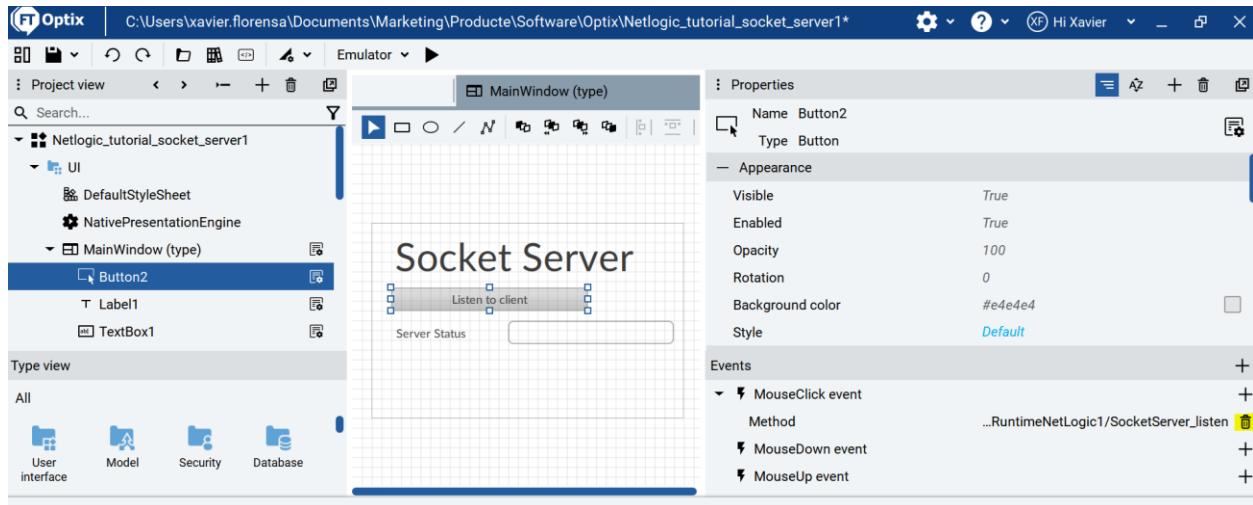
```
[ExportMethod]
0 references
public void SocketServer_listen(NodeId textBoxNodeId)
{
    var textBox = InformationModel.Get<TextBox>(textBoxNodeId);
    textBox.Text = "A button has been pressed";
    Log.Info("A button has been pressed");
    TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 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

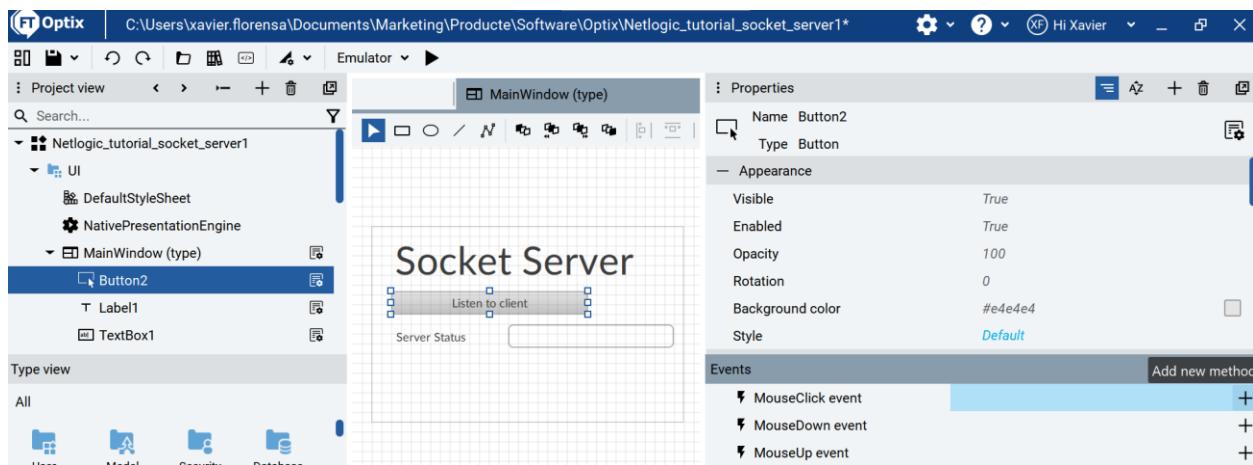


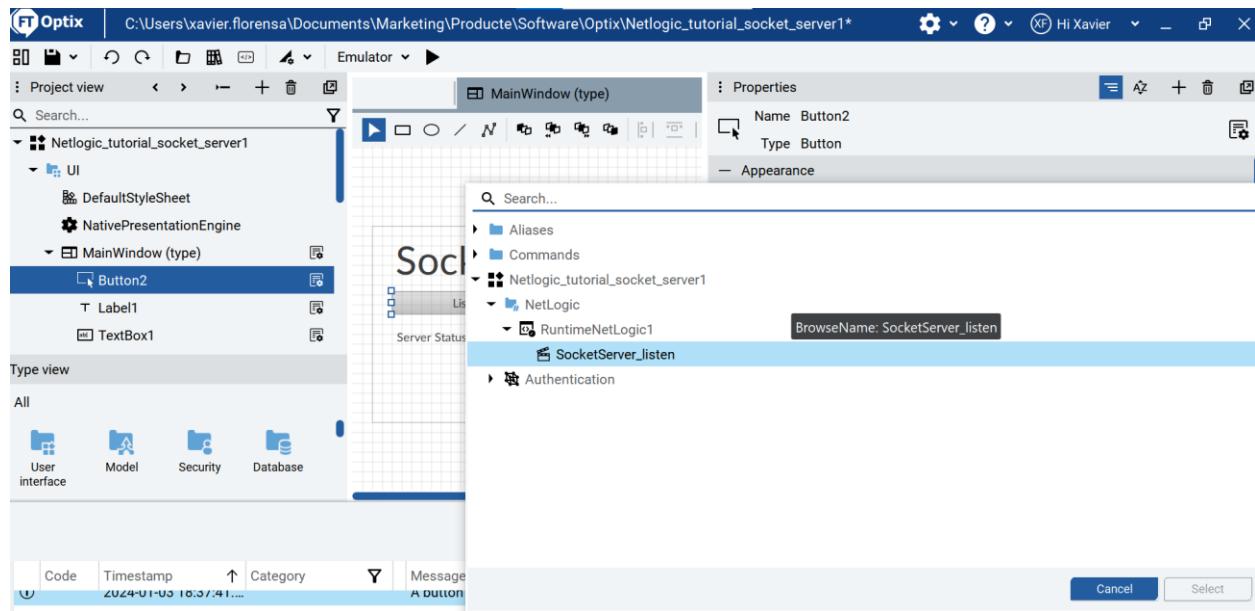
Now let's save the code and let's assign a dynamic link between the Method input parameter and TextBox1

We have to delete the method call after a button click

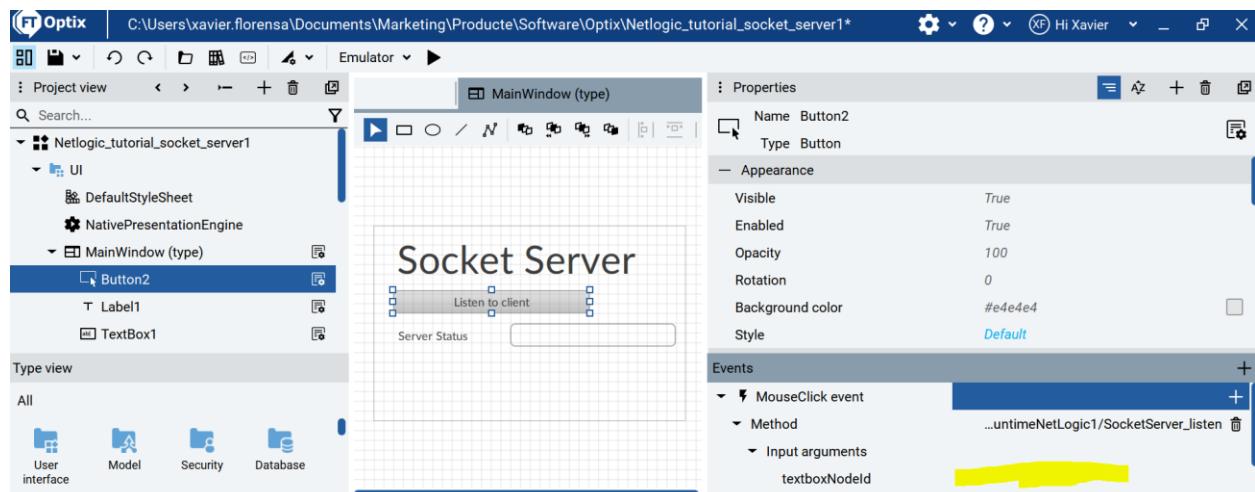


And create a new action on Button click

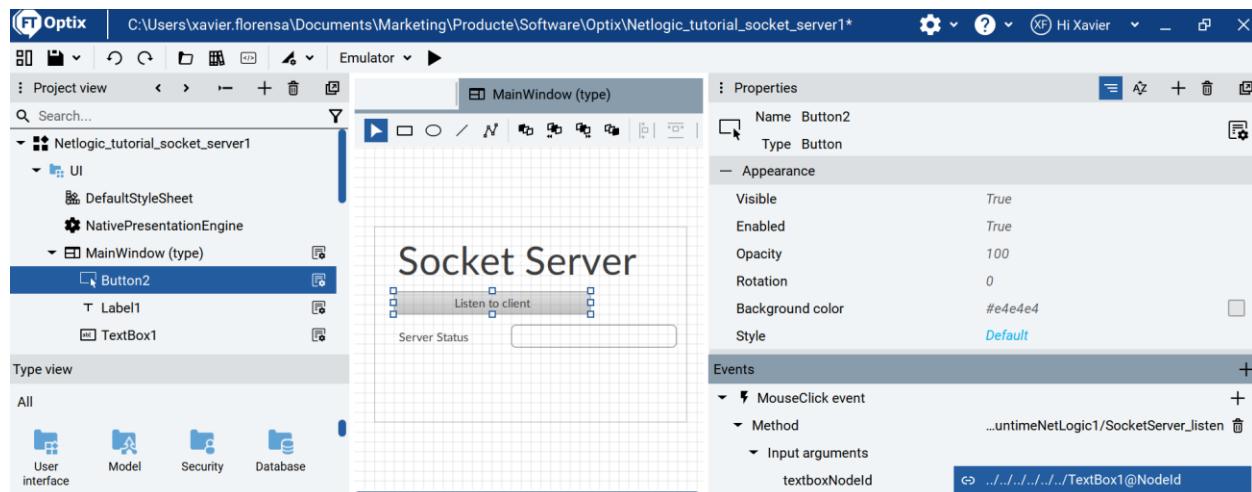
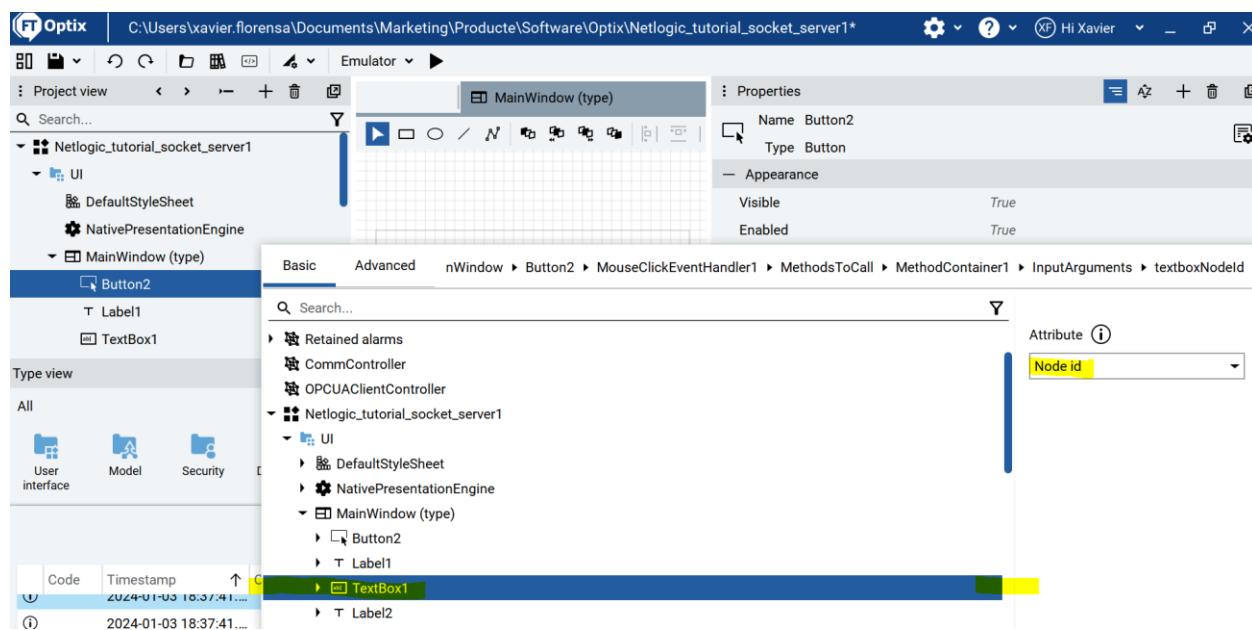
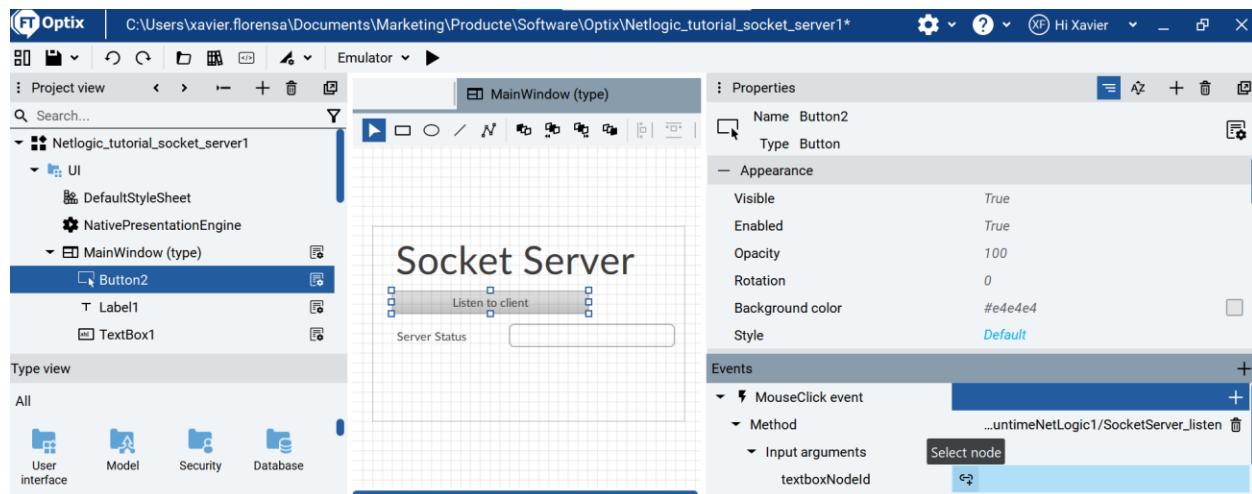




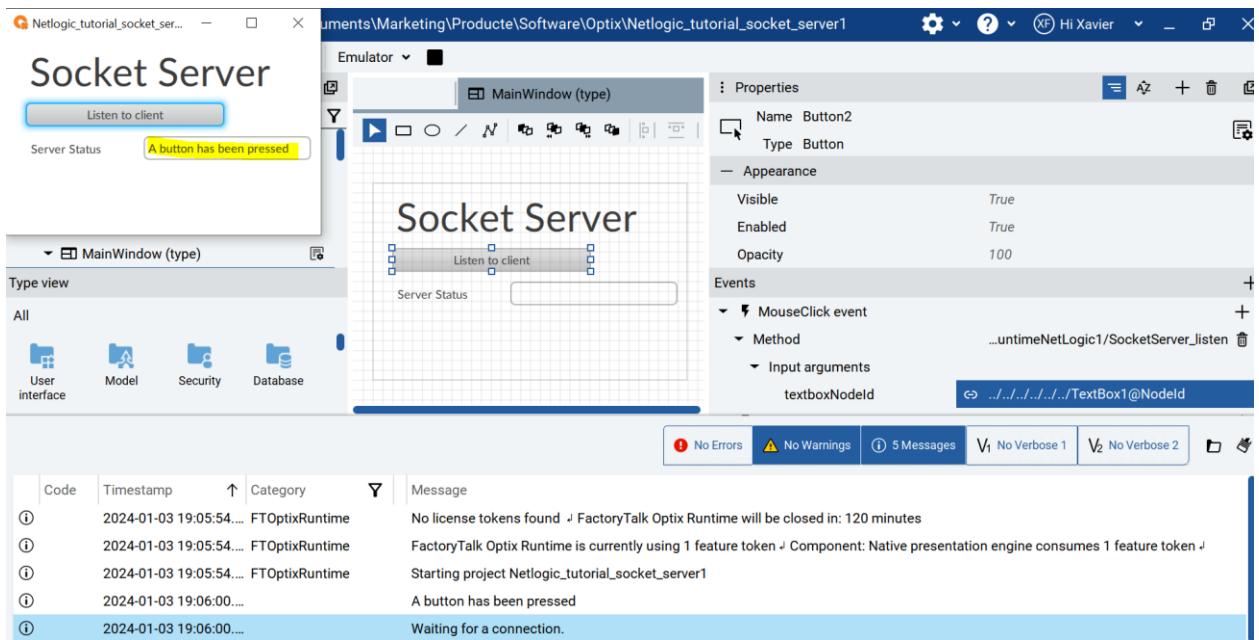
Now on Input arguments, let's create de dynamic link with the TextBox1



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

```
}
```

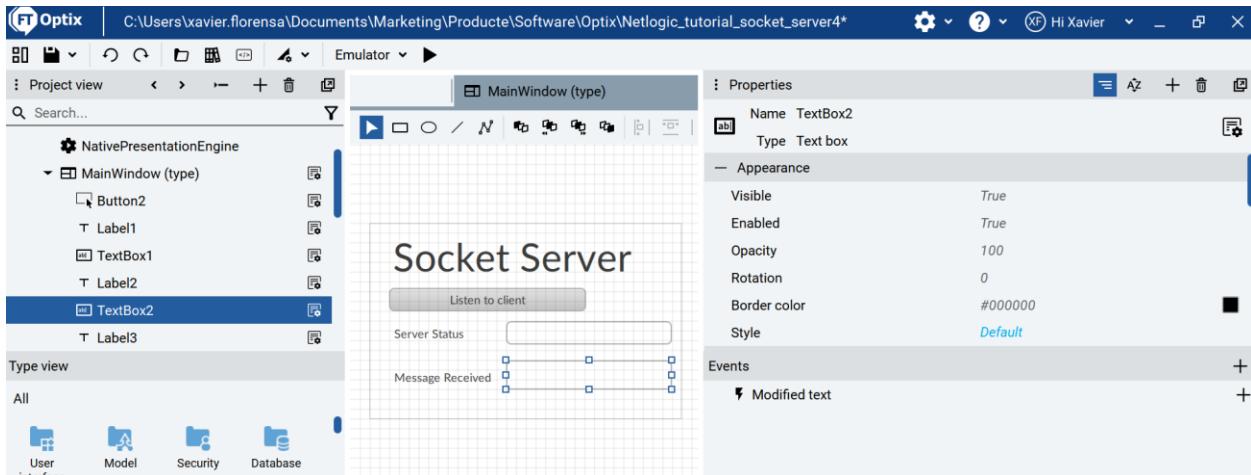
System.Threading.Thread.Sleep(1000);

```
}
```

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.

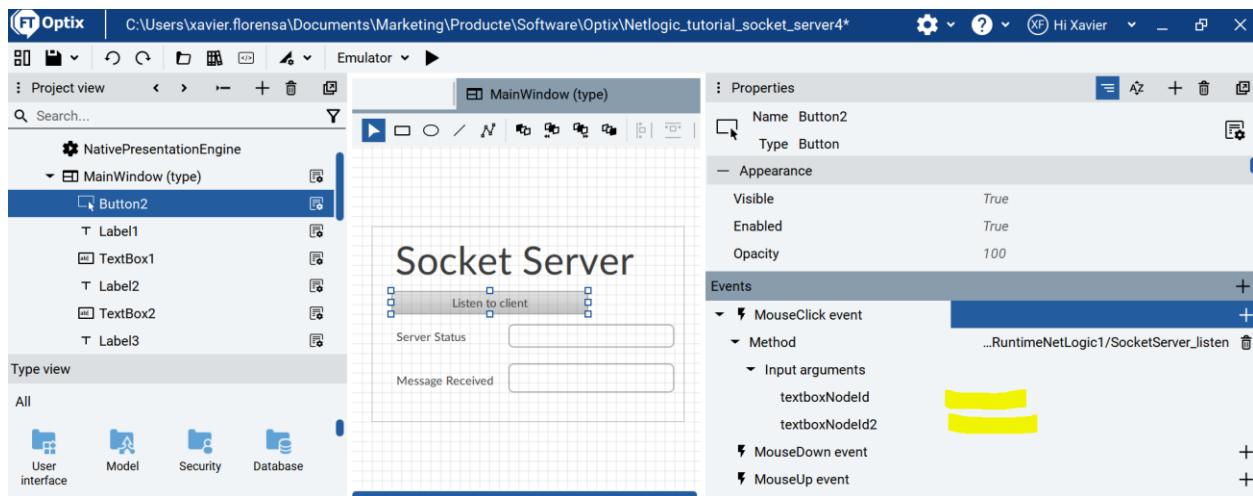
```
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen(NodeId textboxNodeId, NodeId textboxNodeId2)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
        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)
```

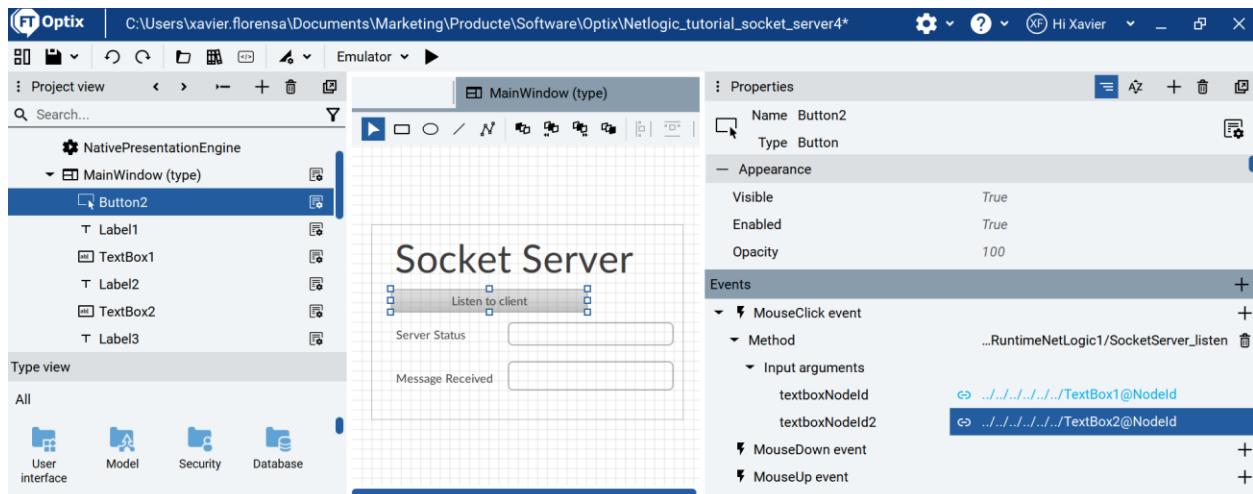
And this is the point where we will show the received message

```
string request = Encoding.UTF8.GetString(buffer, 0, recv);
//Console.WriteLine("request received: "+ request);
textBox2.Text = request;
Log.Info("request received: "+ request);
```

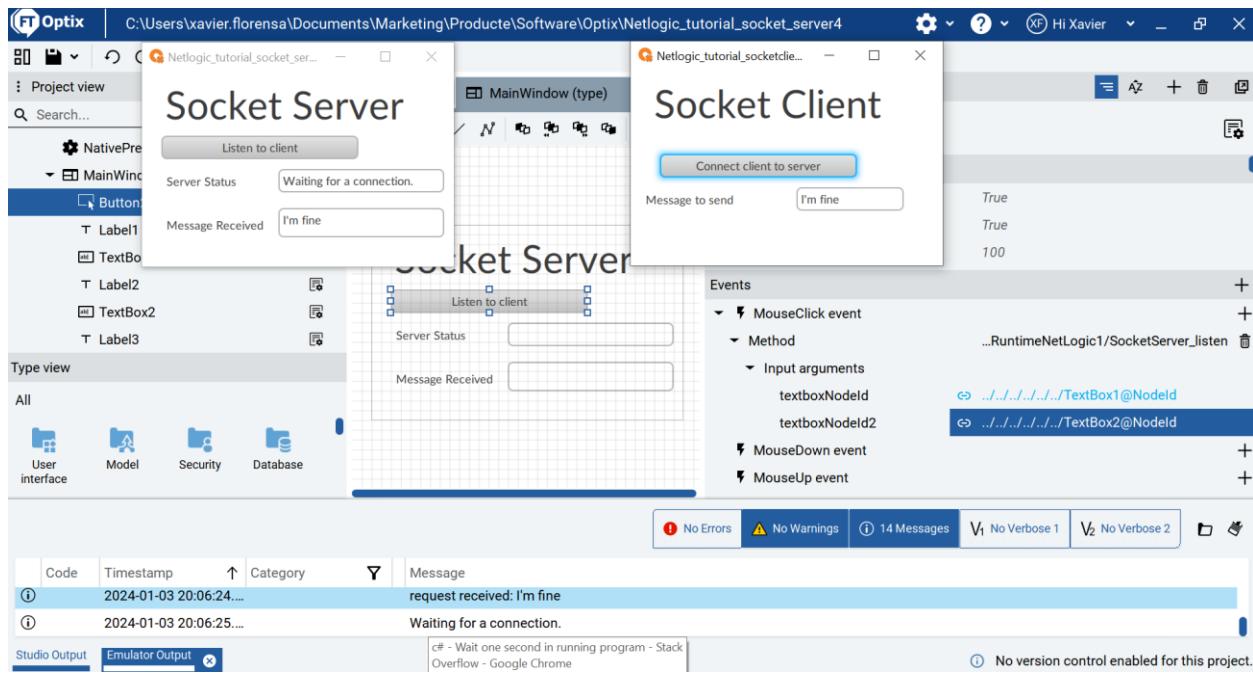
Let's save the code, erase the Method call after a button click and generate again the button click call with the two dynamic links to the corresponding textboxes



Like this



Let's run the application



It works!

This is the code for the server

```
#region Using directives
using System;
using 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(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();
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);
}

```

```
        }
    }
```

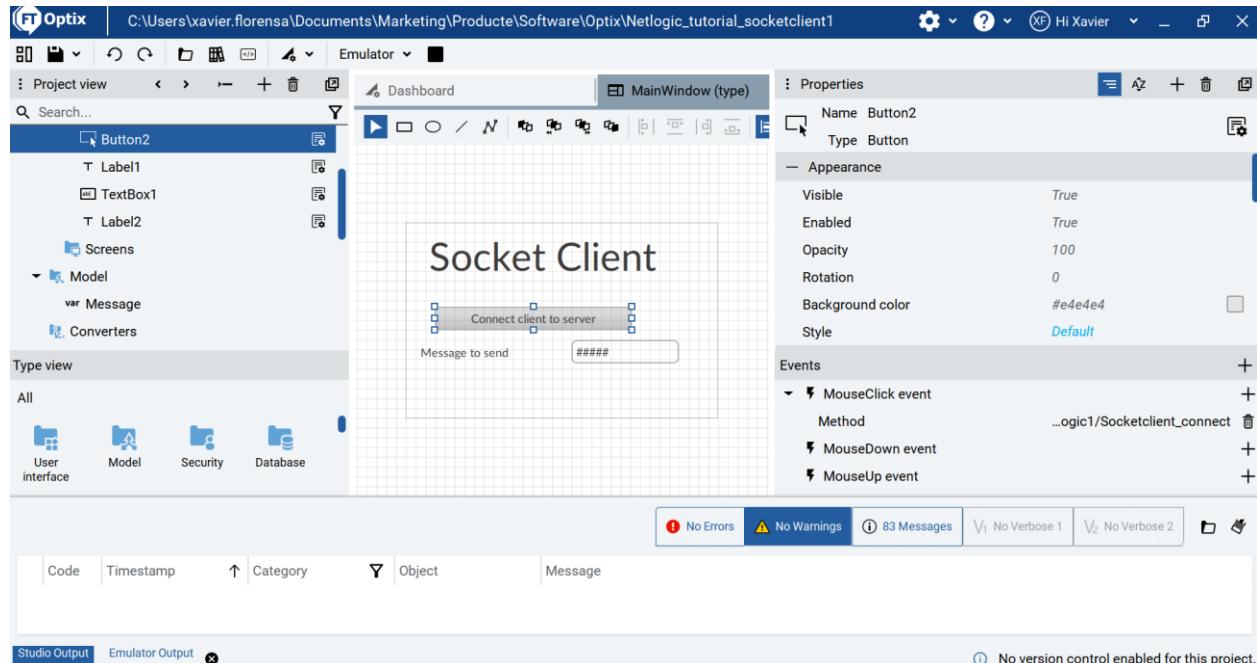
And this is the code for the client

```
#region Using directives
using System;
using 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.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/NetlogicTutorial_SocketServer5.git

https://github.com/xavierflorensa/NetlogicTutorial_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();
    }
}

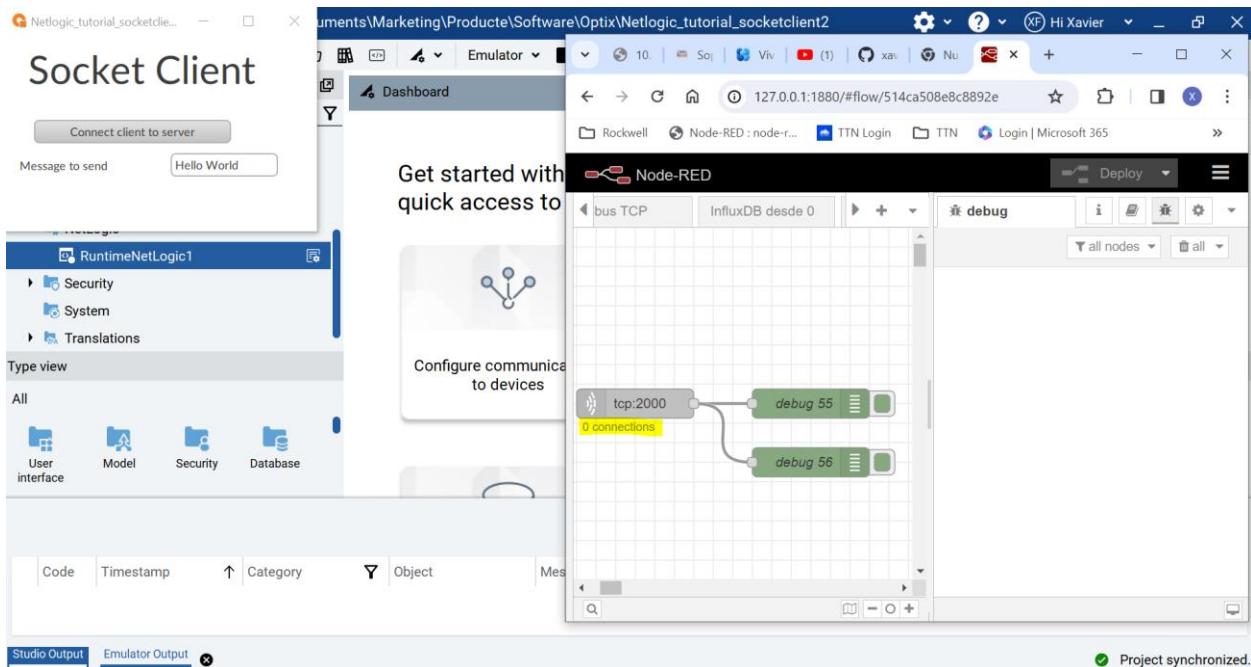
```

```
}
```

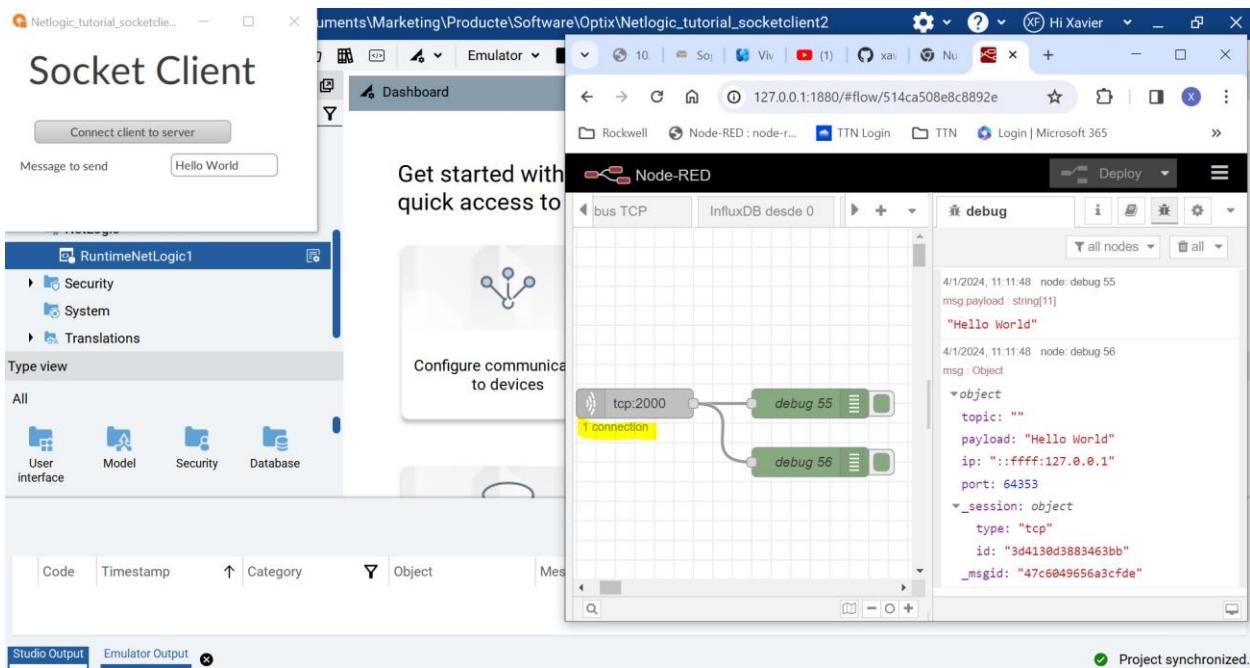
As we can monitor with Node-RED

Before clicking button

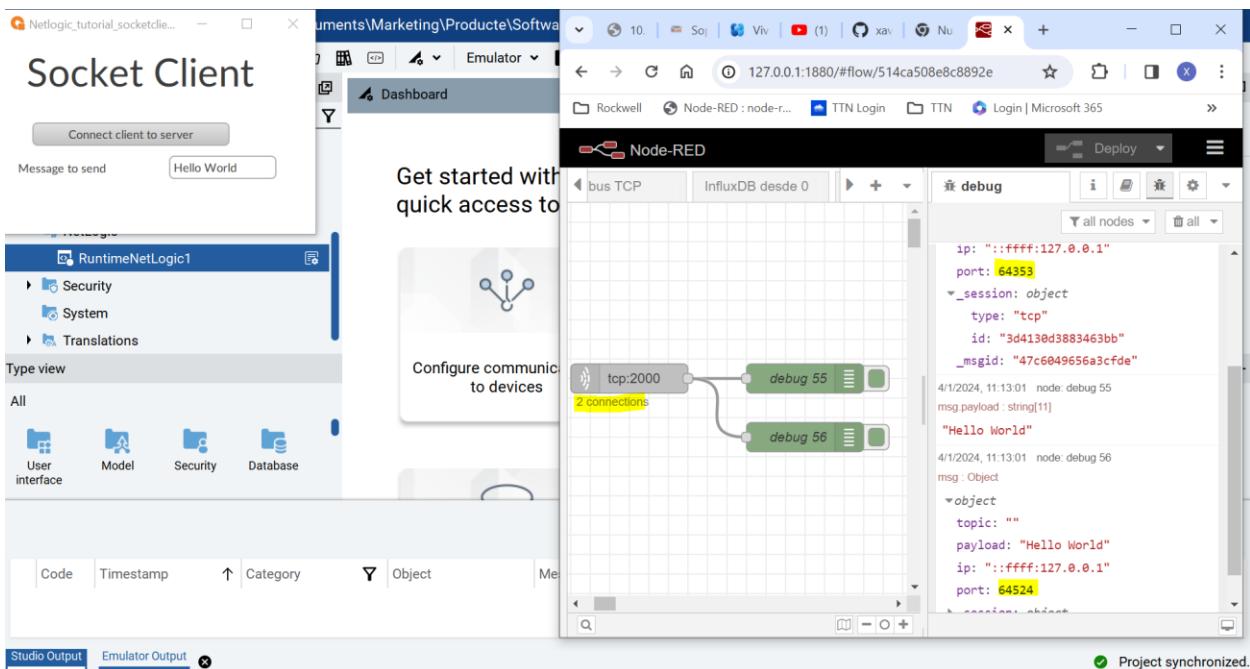
We have 0 connections



First button pressed 1 connection



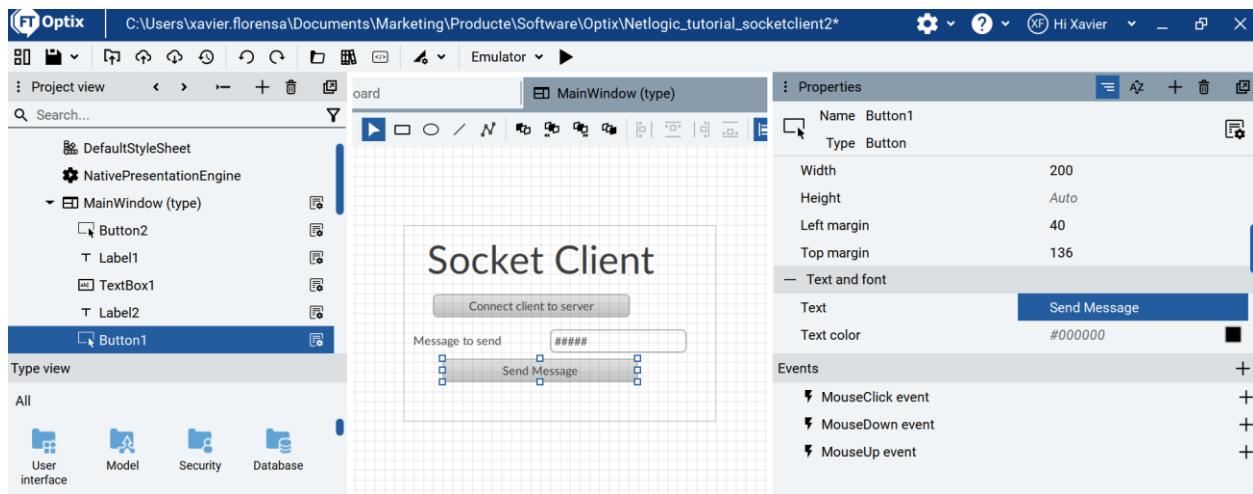
Second button pressed, 2 connections and two different ports



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);

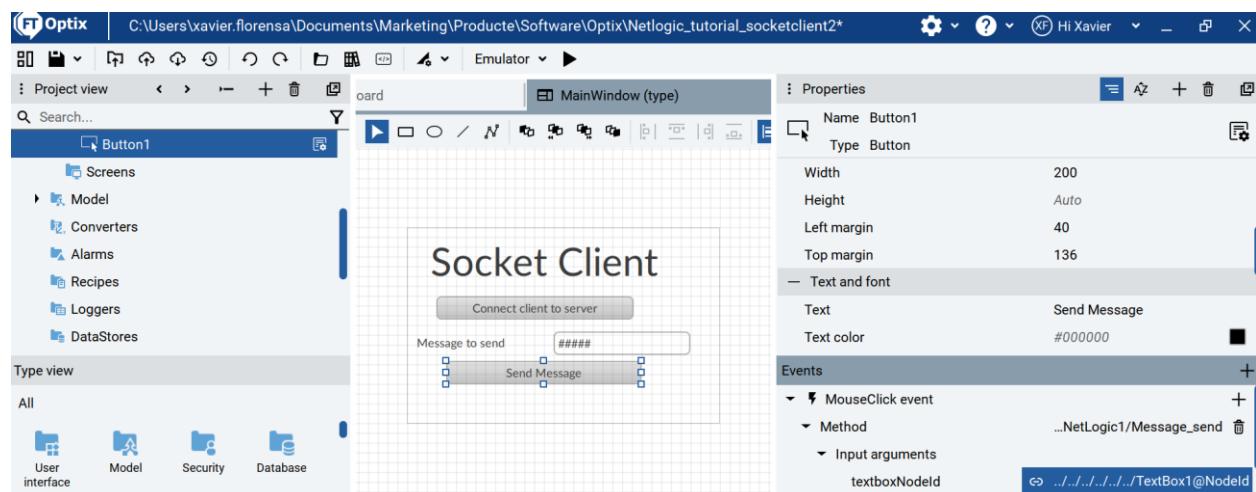
    [ExportMethod]
    public void Socketclient_connect()
    {
        Log.Info("A button has been pressed");
        //TcpClient client = new TcpClient("127.0.0.1", 2000);
        //var missatge = Project.Current.GetVariable("Model/Message");
        //string messageToSend = "Hello World";
        //string 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...");
    }
}

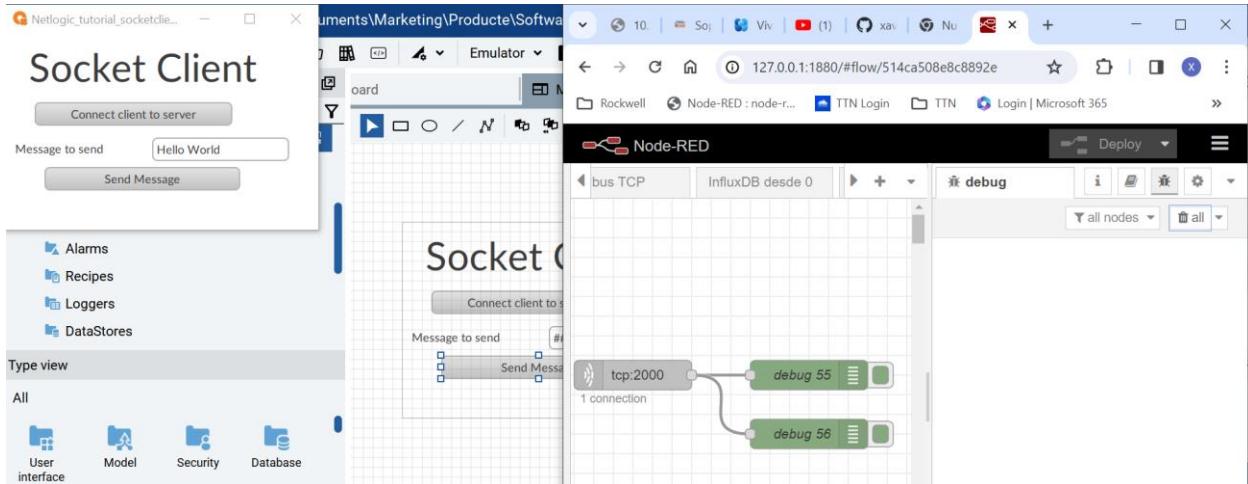
```

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.

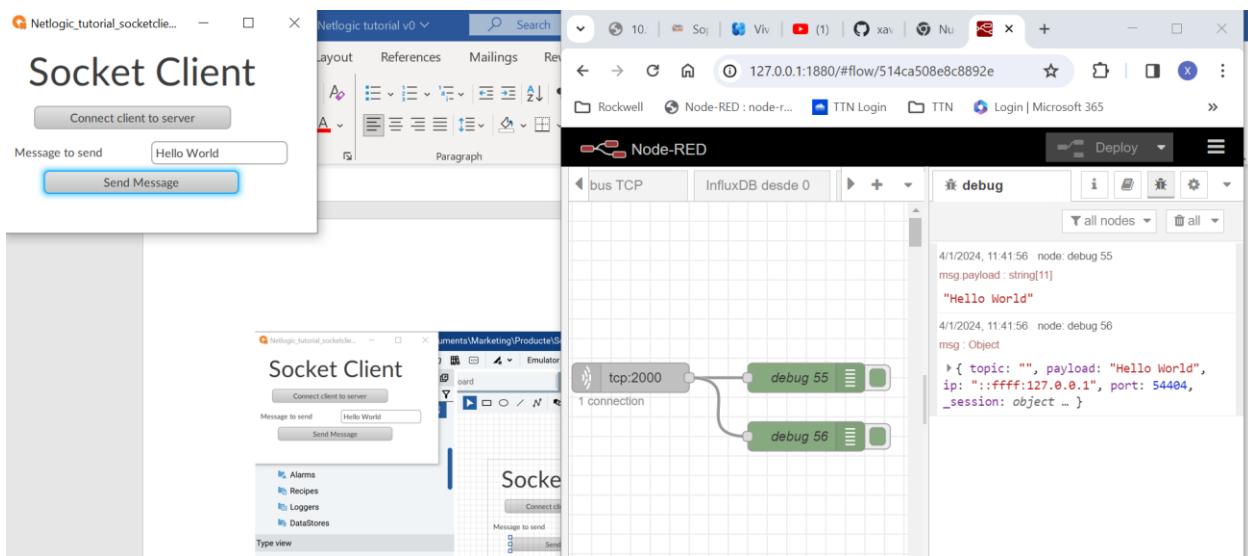


Test the application

Just starting the application, one client is connected



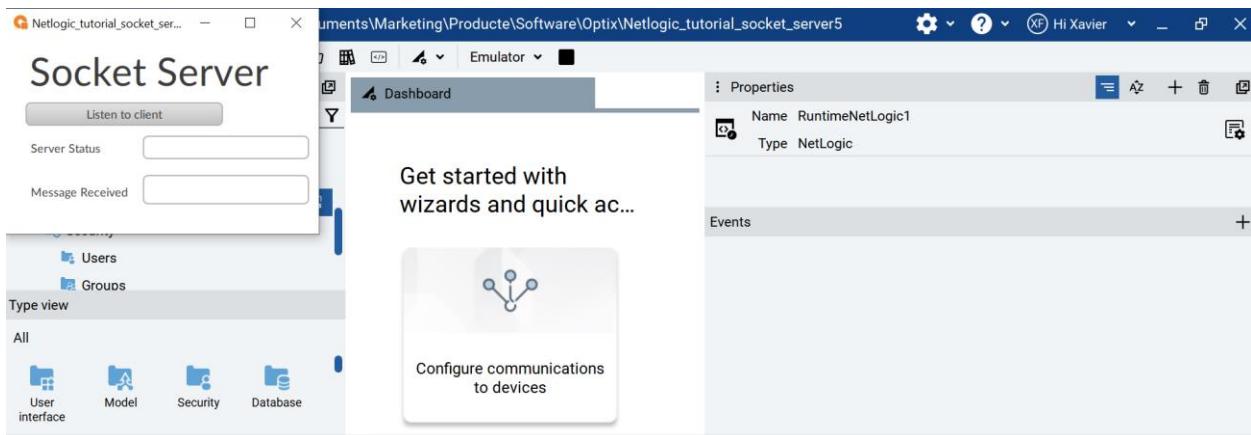
Send



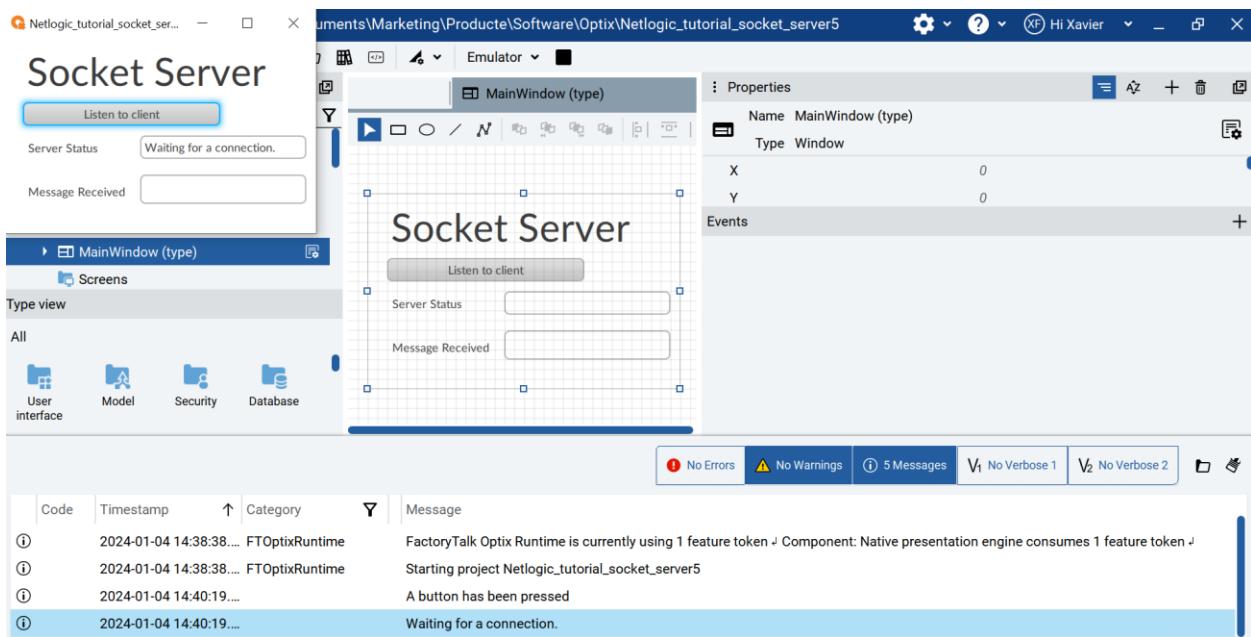
Second send, only one client is connected

Now let's try with our FactoryTalk Optix server

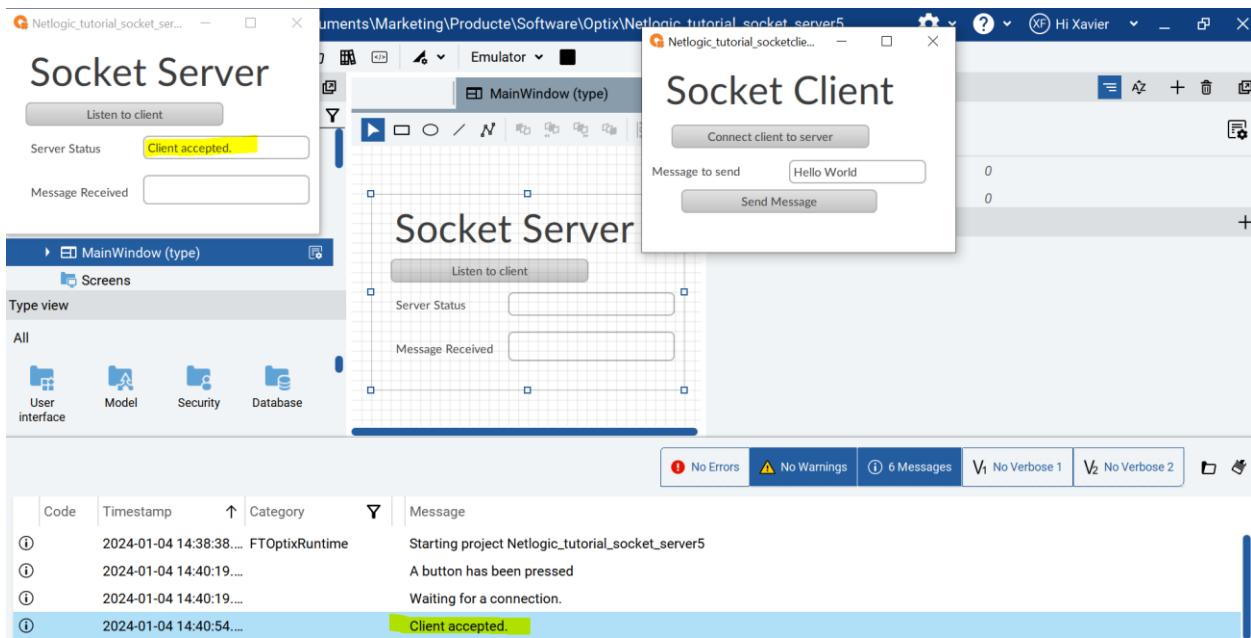
First we start only the server (since client makes connection just after starting, and if no server is listening then this will create an error exception)



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 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.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);
    }
}

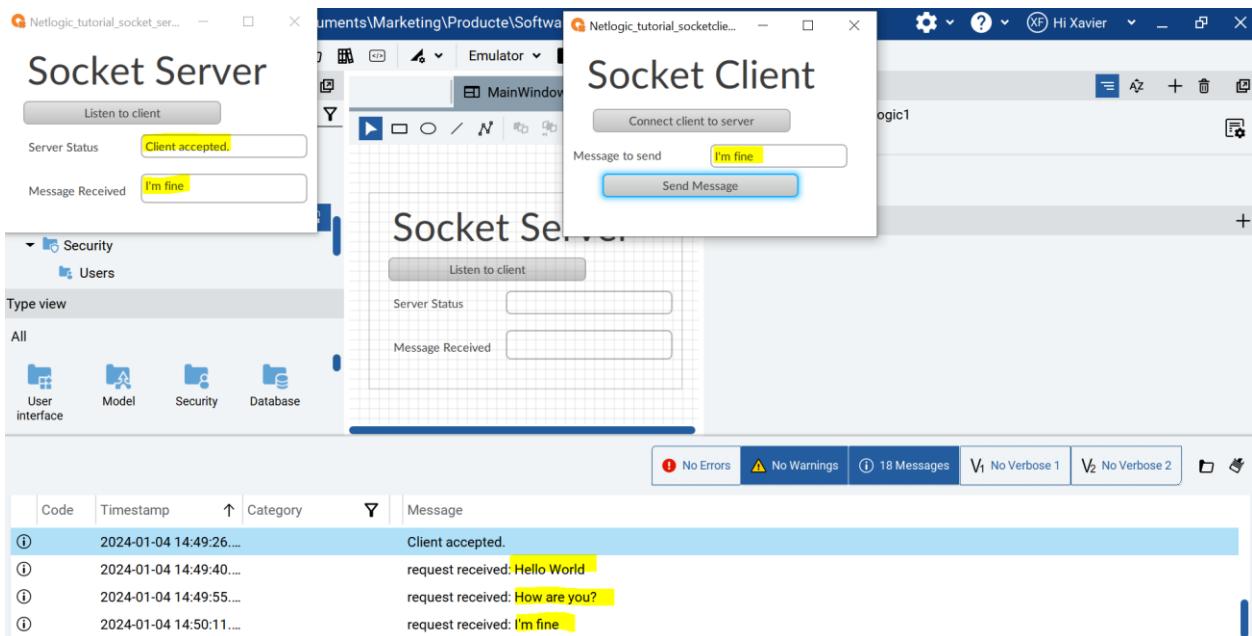
```

```

        }
    }
}

```

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.IPAddress.Any, 2000);
    listener.Start();
    textbox.Text = "Waiting for a connection.";
    Log.Info("Waiting for a connection.");
    TcpClient client = listener.AcceptTcpClient();
}

```

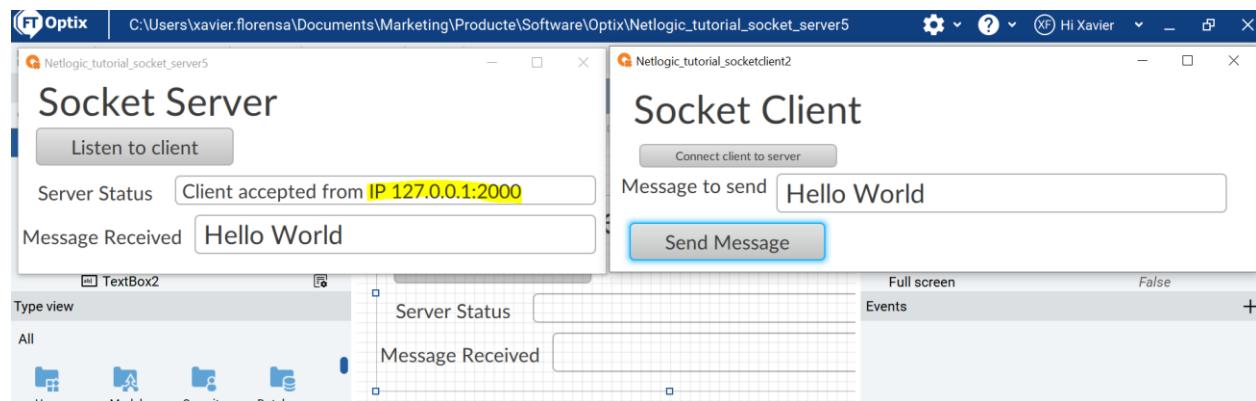
```

//-----
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

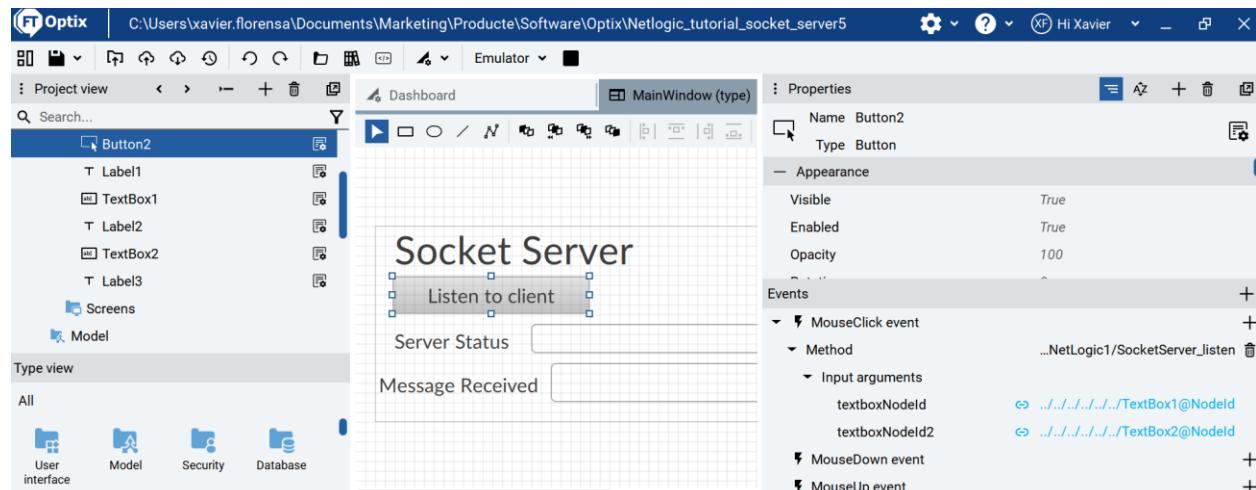


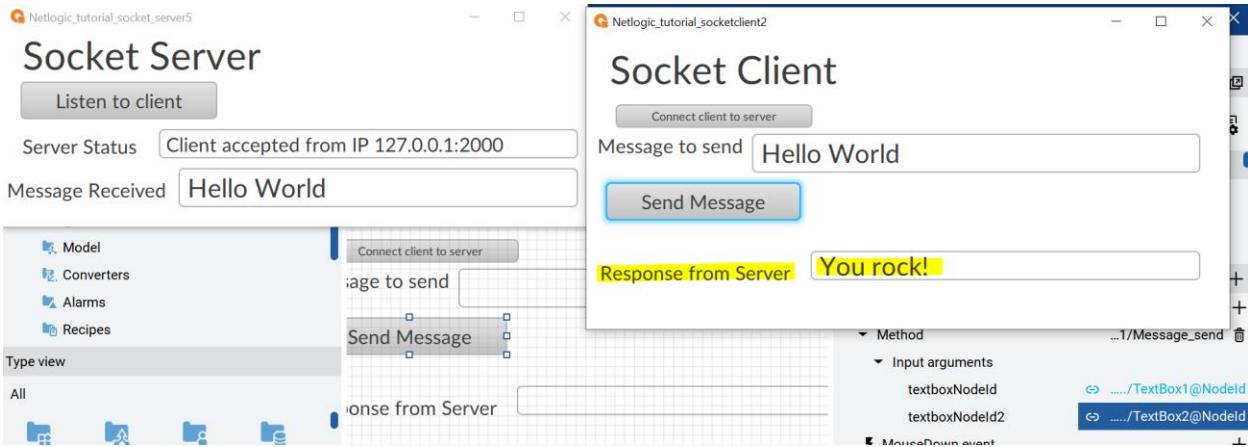
Now seems that if you close the client first and then you close the server, it stops immediately. Even with an error. But this is OK since the socket is broken after you stop the client application.

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 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
{
    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;
```

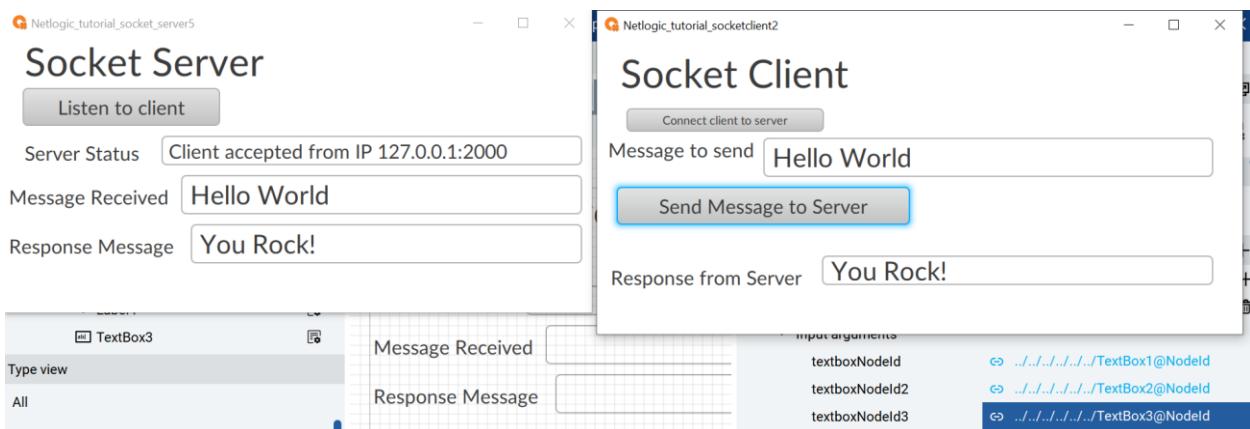
```

        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.HMIProject;
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.

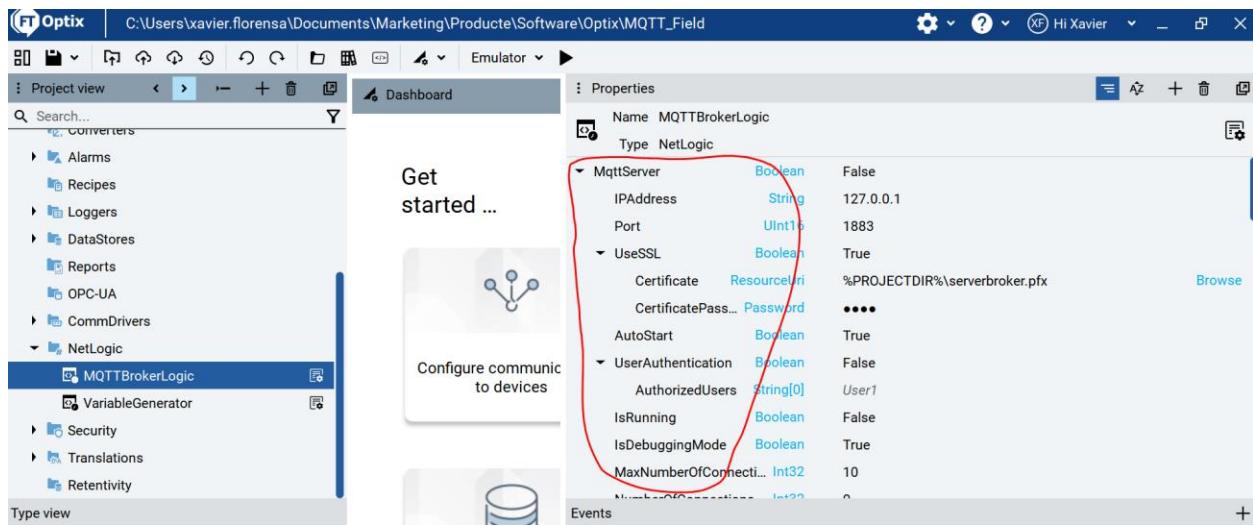
Next Step will be selecting the target IP addresses of the client.

37. How to define NetLogic Parameters

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 find a simple example

38. 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#

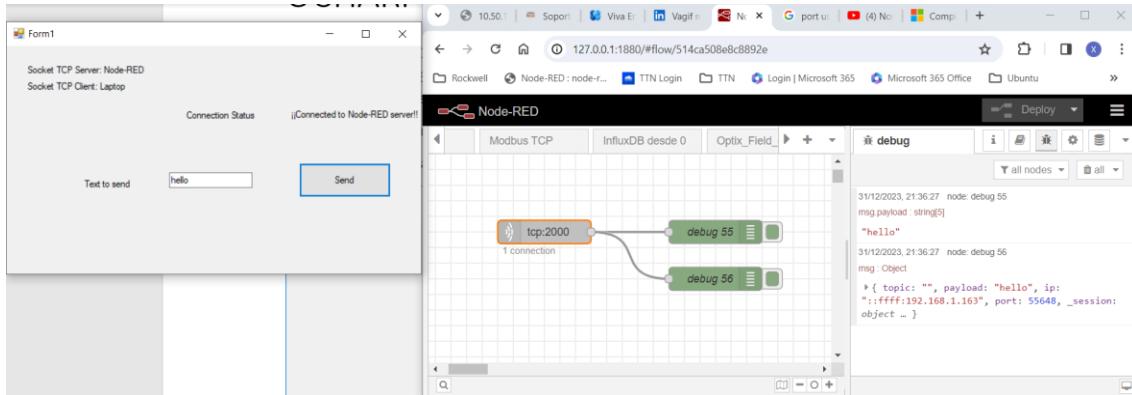
<https://www.codeproject.com/Articles/21443/Winsock-Revamped>

This DLL is tested in Visual Studio 2022. With this code in Visual Studio 2022

38.1. 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

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.WinsocConnectedEventArgs 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);
        }
    }
}

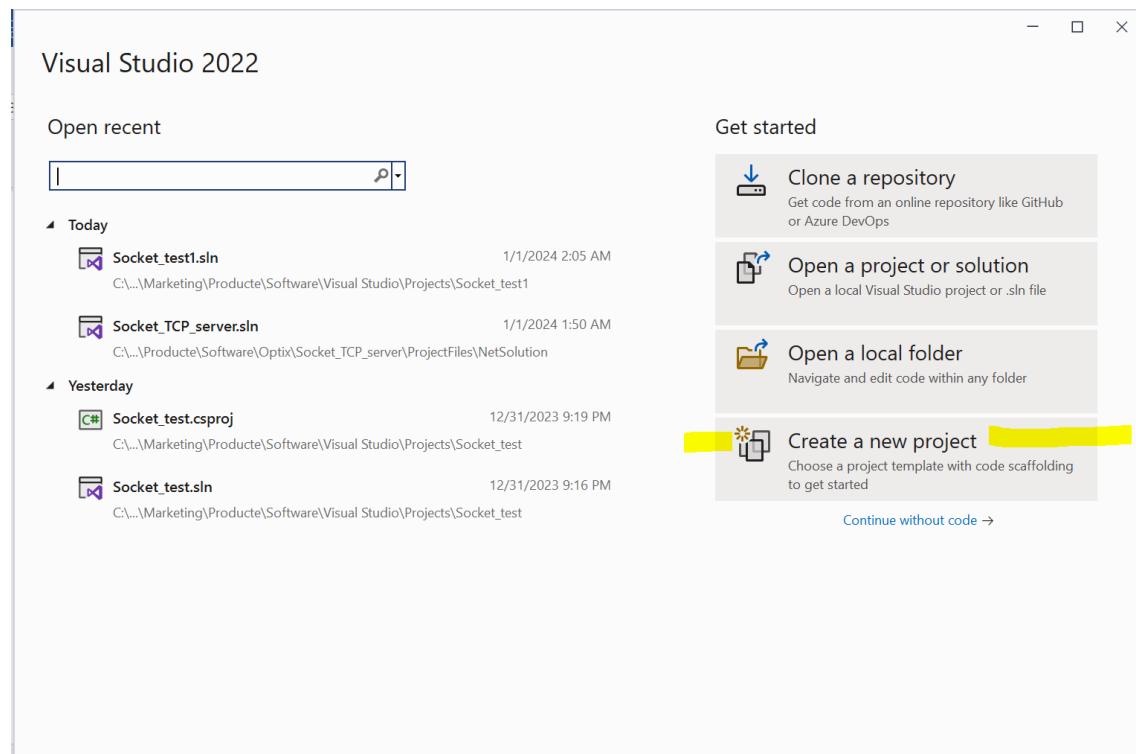
```

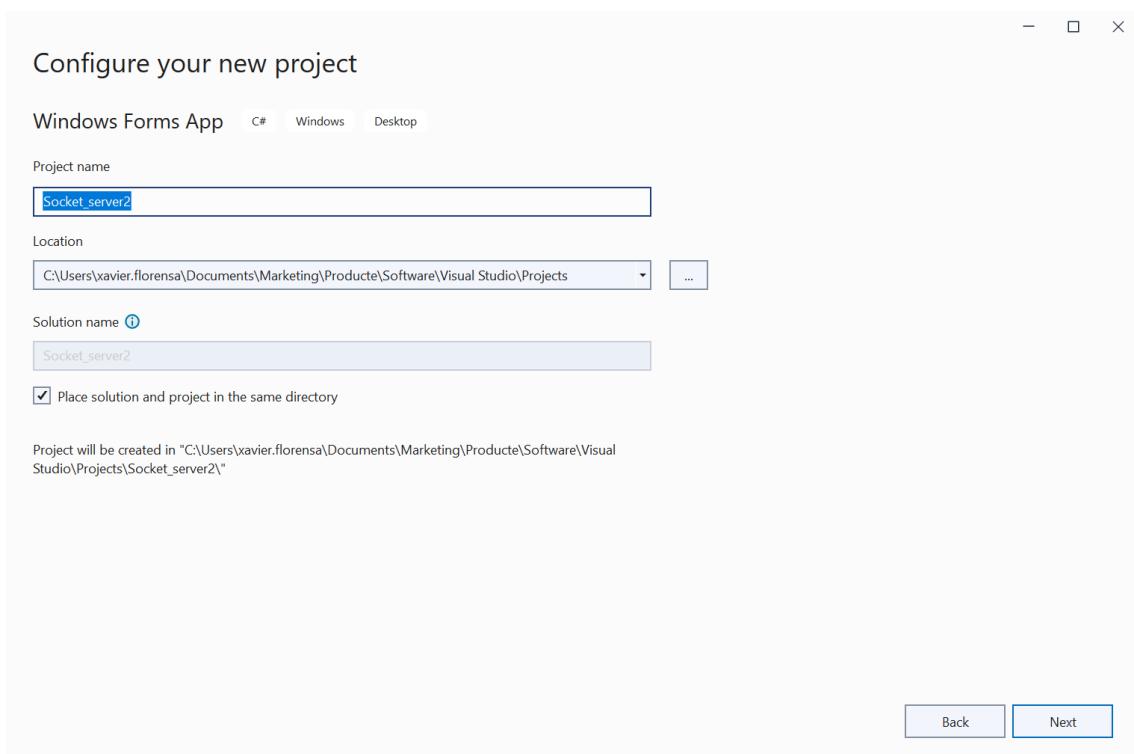
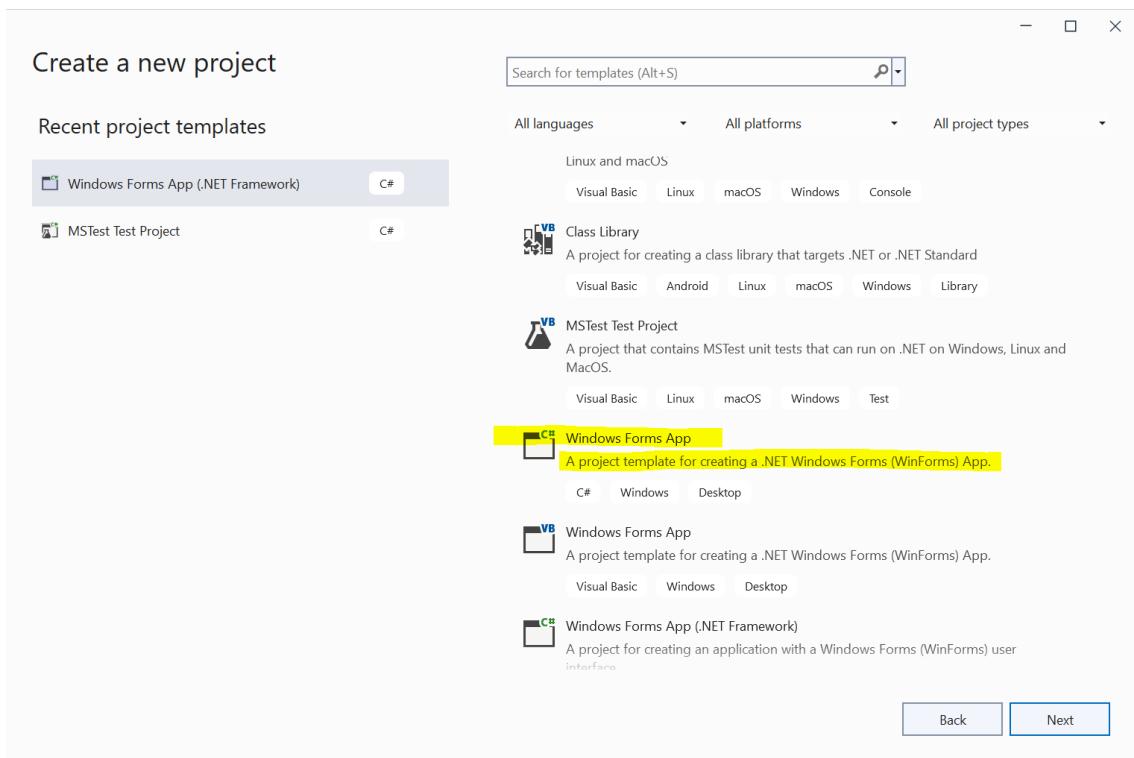
How to proceed if you want to use this code for your own applications:

Copy the Winsock Orcas dll on a known location on your hard disk

Open Visual studio

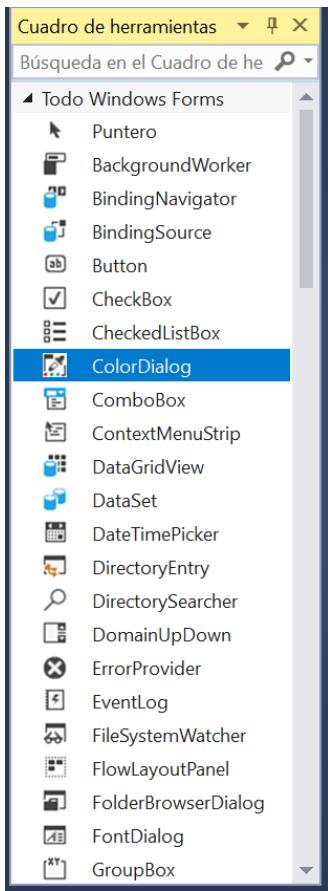
Create new project in visual studio (this is Visual studio 2022 community edition)



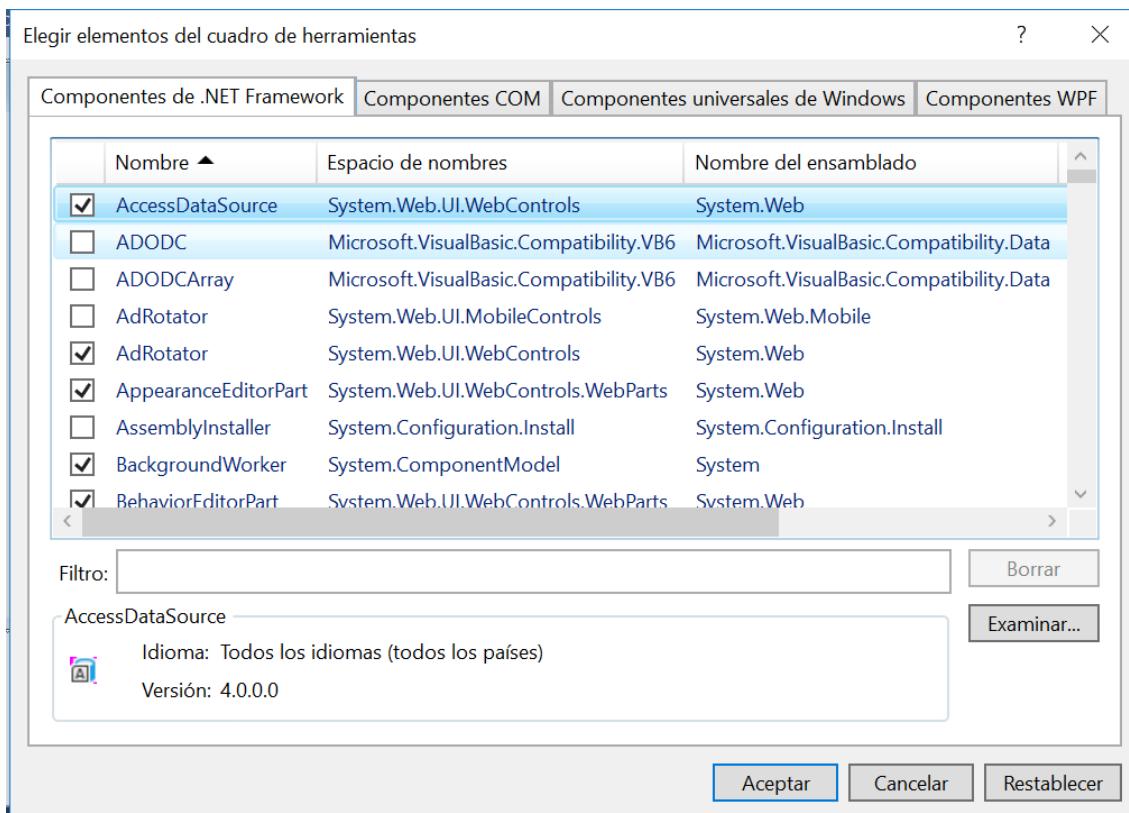


Once created,

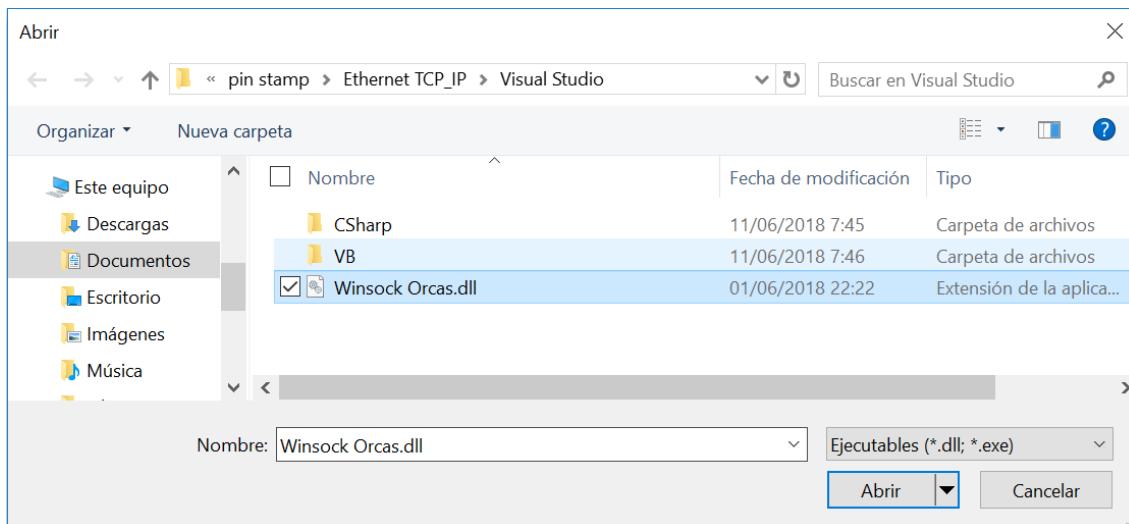
then right click the mouse anywhere on your toolbox list to add the Winsock Orcas control for easy programming



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)

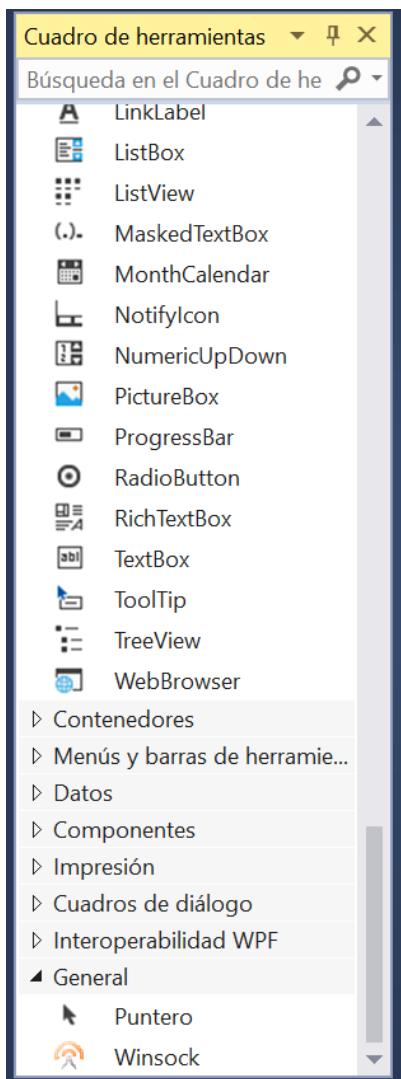
Versión: 4.0.0.0

Aceptar

Cancelar

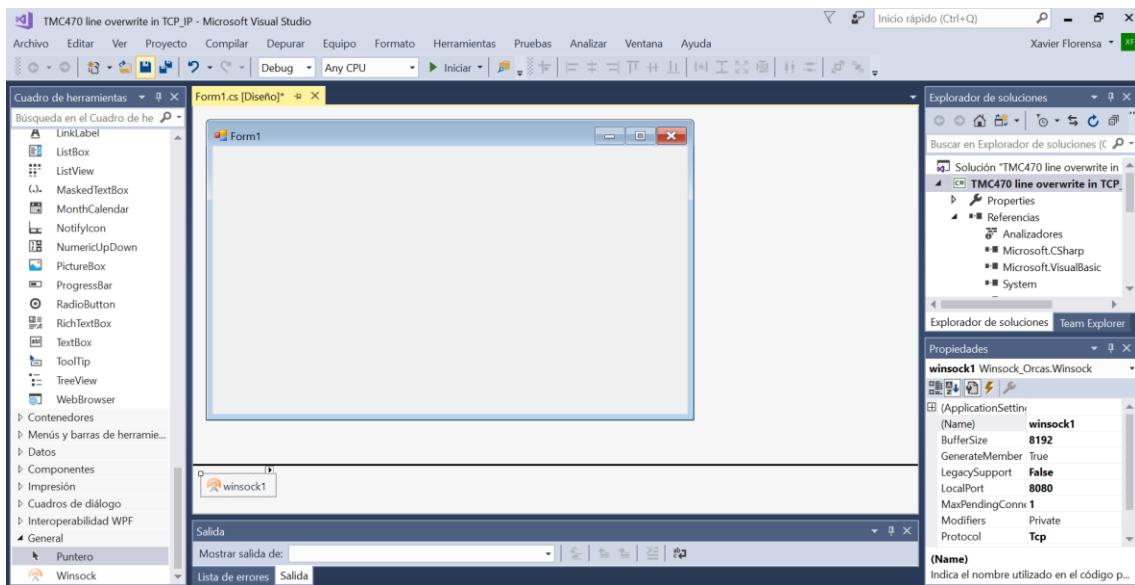
Restablecer

You will find it now at the end of your toolbox list



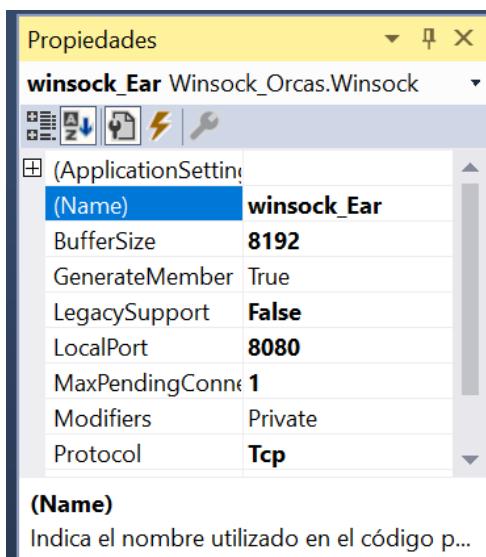
Click on the Winsock control icon and drag until your form

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 Legacy (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

```

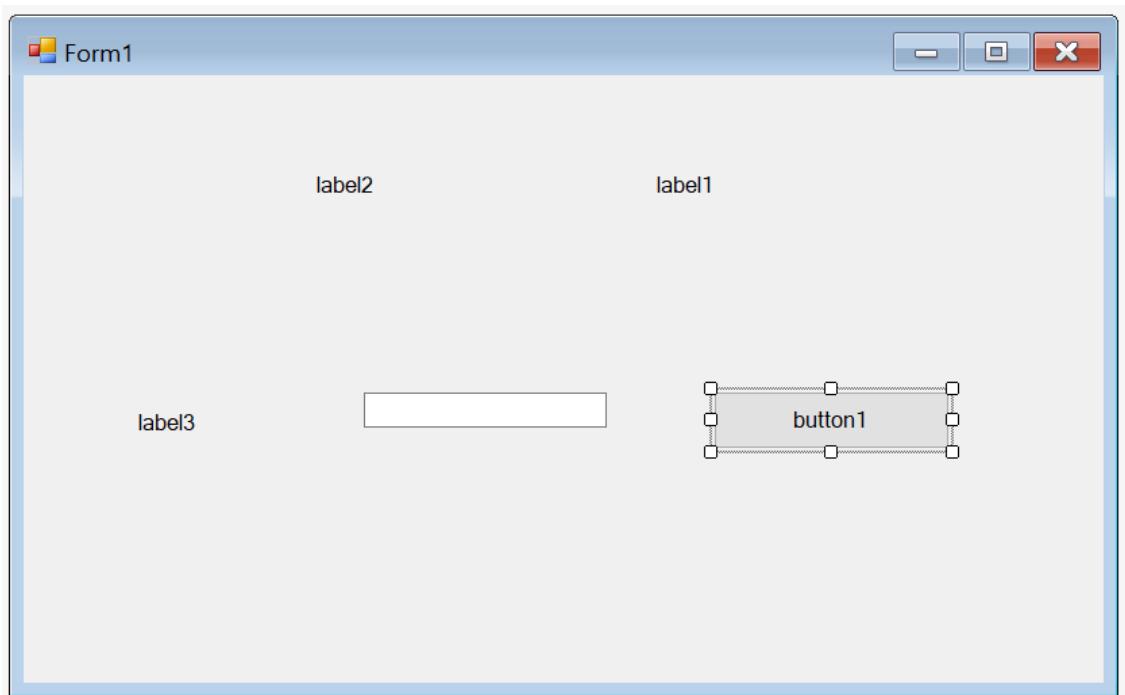
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10
11 namespace TMC470_line_overwrite_in_TCP_IP
12 {
13     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

```

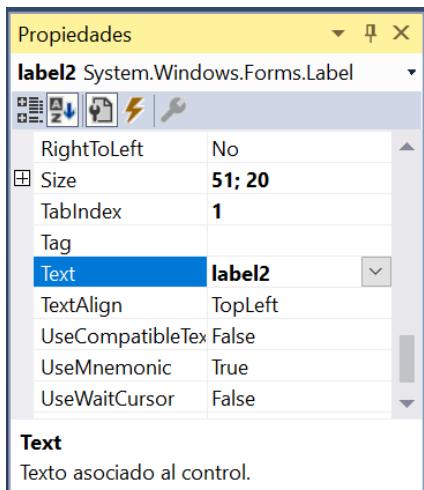
We move now to the form view

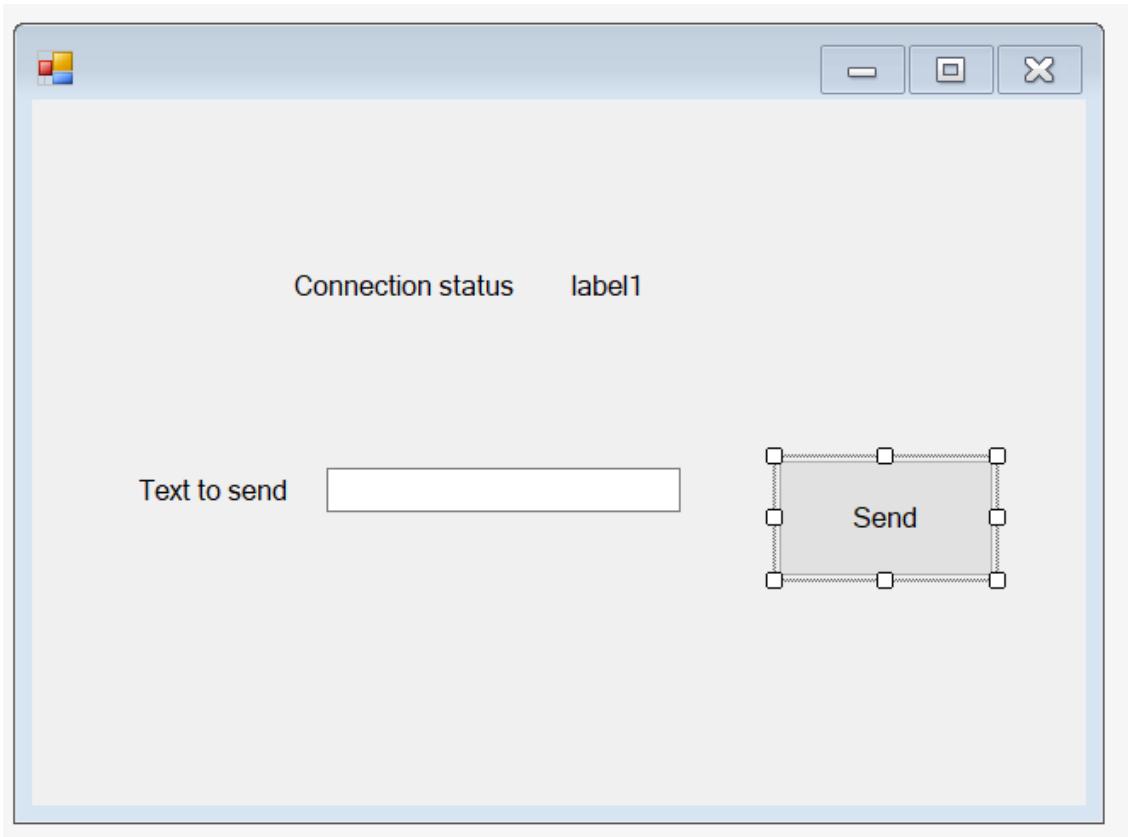
Then we start populating the form with the desired controls (click on the controls list and drag on the form)

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

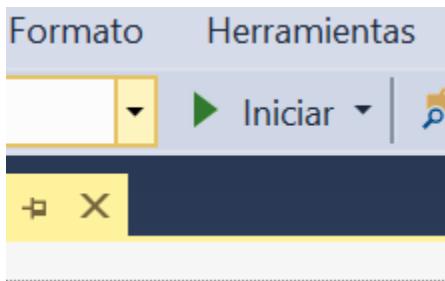
```

private void winsock_Ear_ConnectionRequest(object sender,
WinsockConnectionEventArgs e)
{
    winsock_Ear.Close();
    winsock_Ear.Accept(e.Client);
}

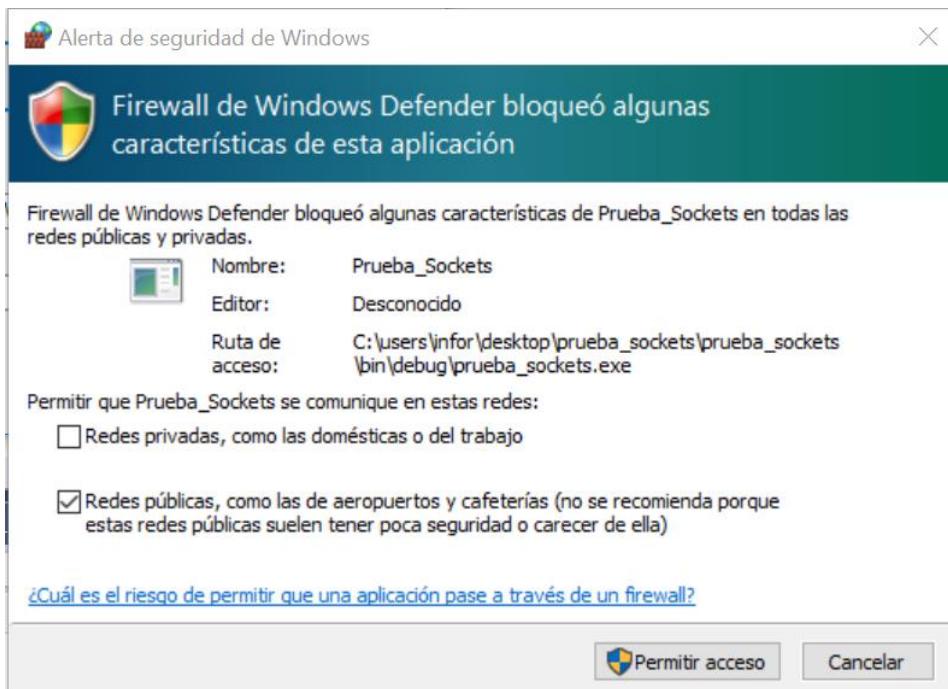
```

Now you can test the application to see if you get connection

Just click on green “play” arrow

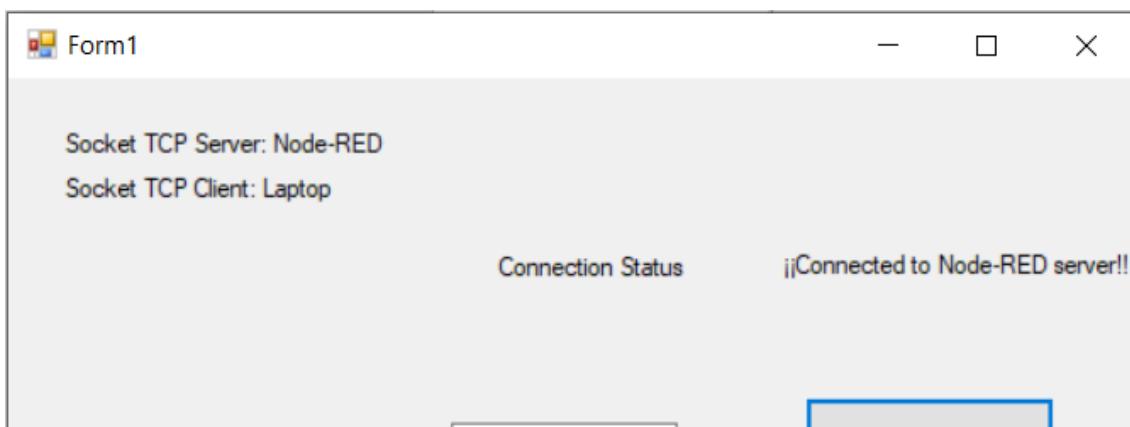


First time you will see this message:



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

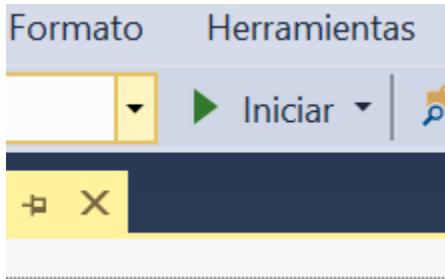
This action will introduce this code without typing anything

```
private void button1_Click(object sender, EventArgs e)
{
}
```

Then type (or copy / paste) this content

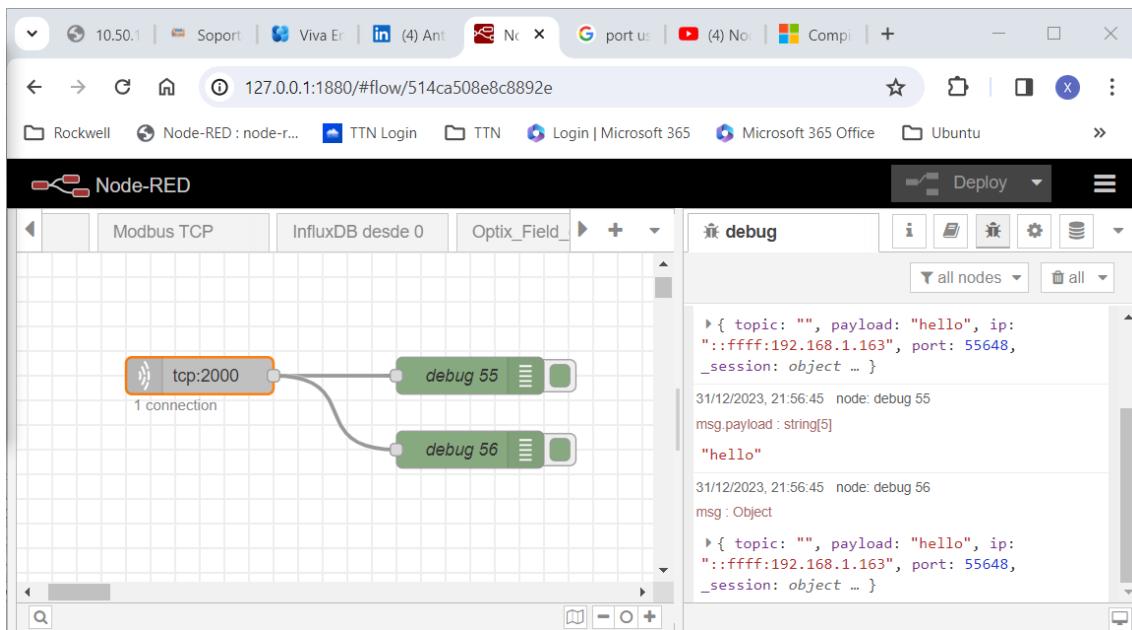
```
private void button1_Click(object sender, EventArgs e)
{
    //we get the text from the Textbox entered from the keyboard
    string text_to_send = this.textBox1.Text;
    winsock_Ear.Send(text_to_send);
}
```

You can finally test the application



And that's all!

This is the Node-RED application



Edit tcp in node

[Delete](#) [Cancel](#) [Done](#)

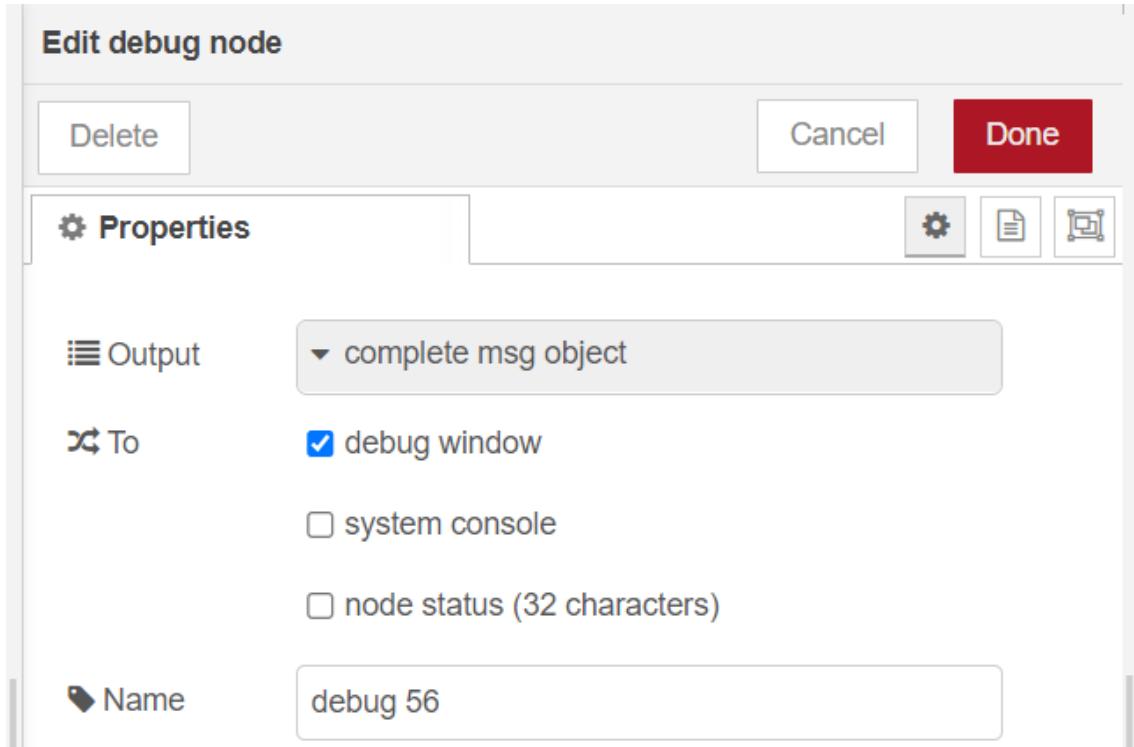
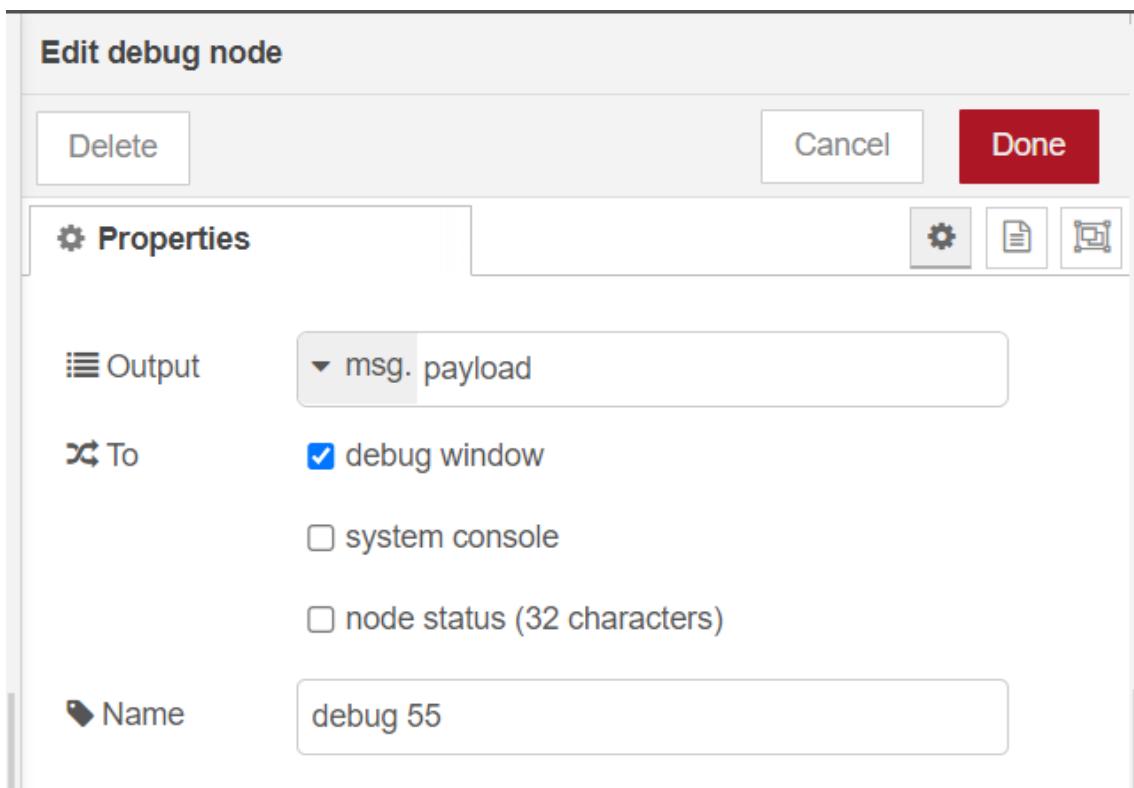
Properties

Type Listen on port
 Enable secure (SSL/TLS) connection

Output stream of String payload(s)
delimited by
 re-attach delimiter

Topic

Name



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.

38.2. Third party libraries (dll) Socket Server application with Optix

You can see the result here

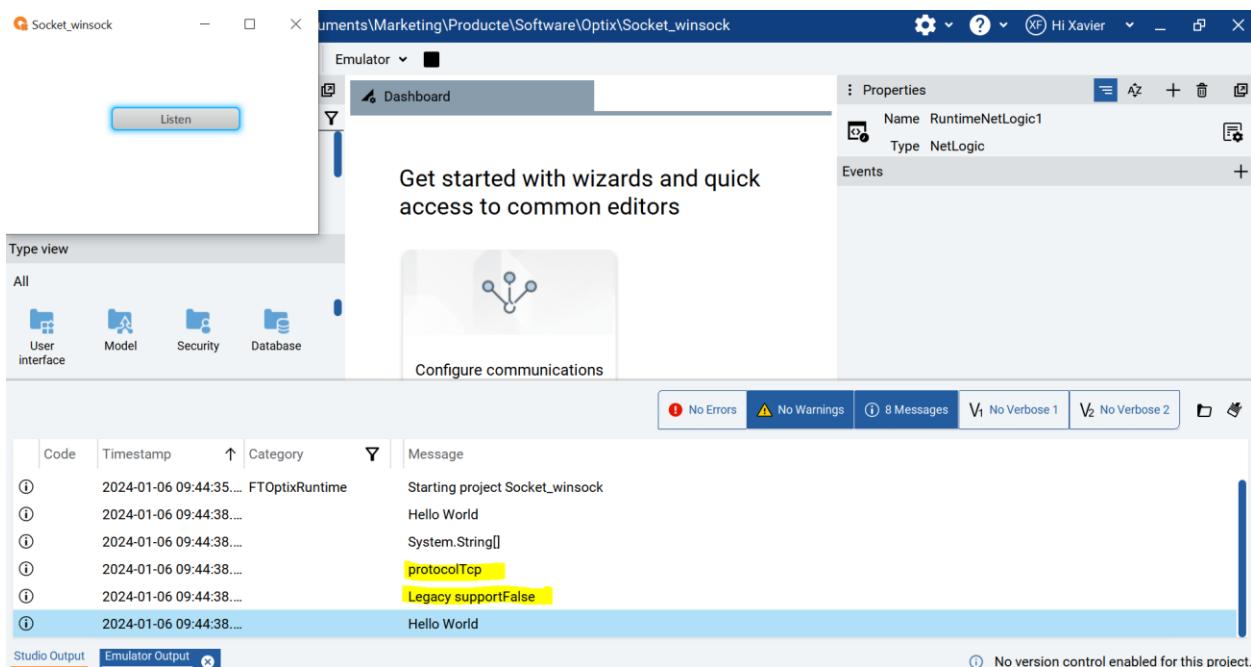
<https://youtu.be/6E4xt02Ohzw>

You can download the code from here

https://github.com/xavierflorensa/Optix_Winsock_Orcas_Server.git

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_winsock

[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 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 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,
WinsockConnectionEventArgs e)
{
    winsock_Ear.Close();
    winsock_Ear.Accept(e.Client);
    Log.Info("Connected to slave!!");

}

//private Winsock_Orcas.Winsokc winsock_Ear;
//public Winsock_Orcas.Winsokc winsock_Ear;
}

```

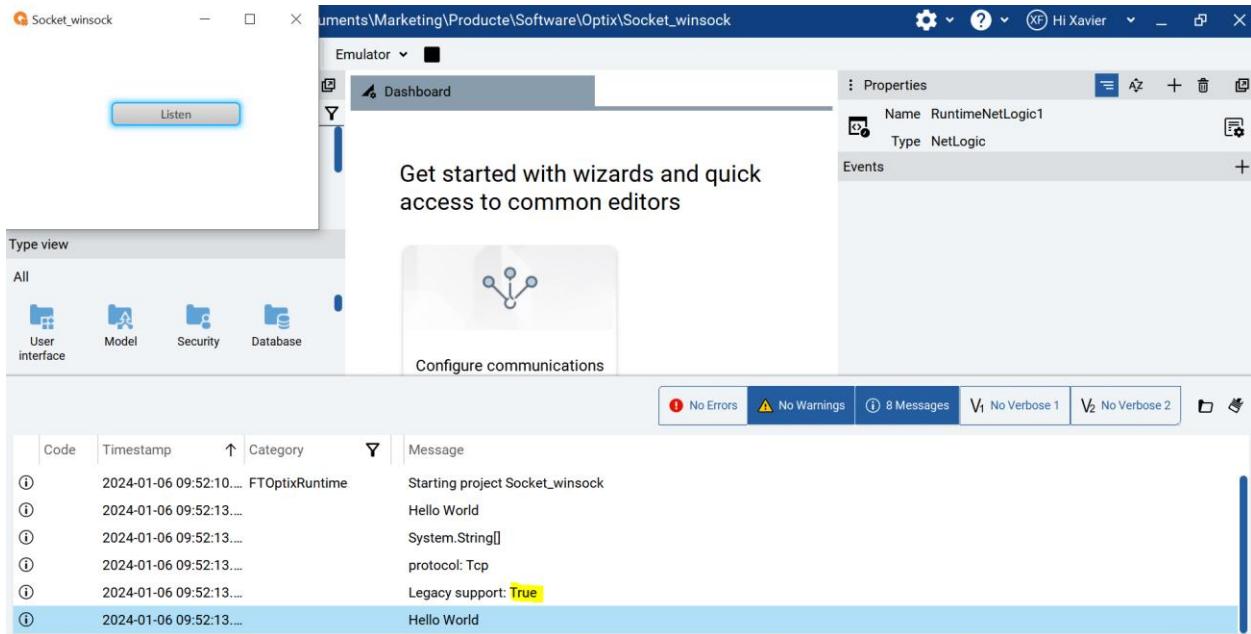
If we change status of legacy support we see

```

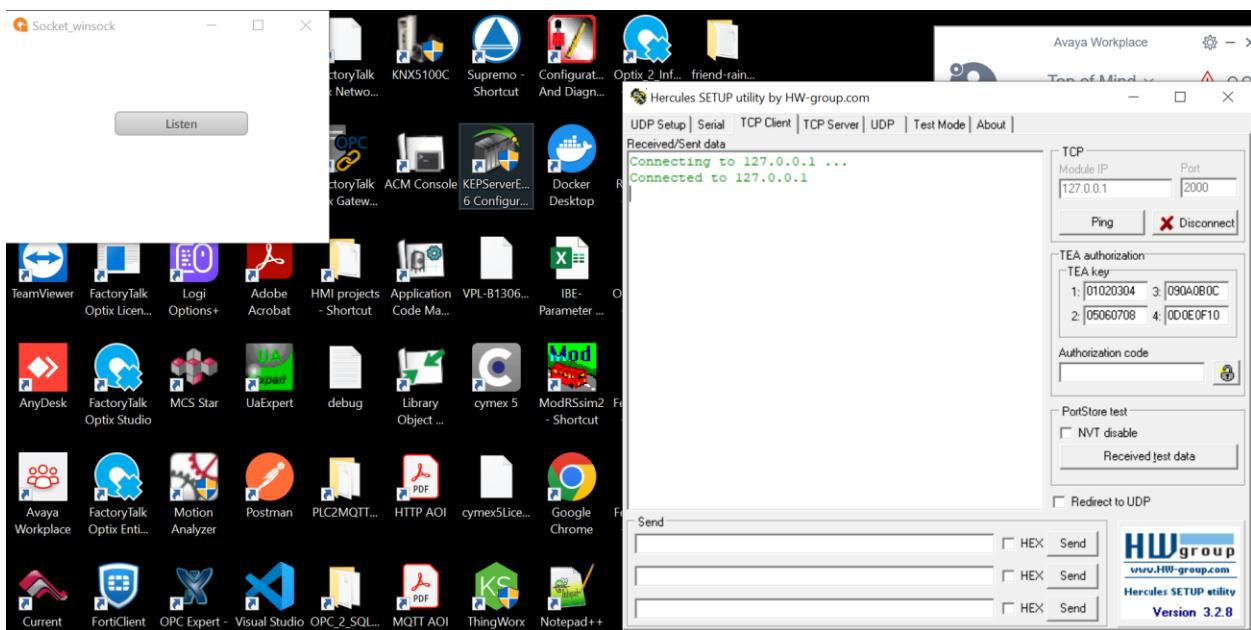
public void Send()
{
    Log.Info("Hello World");
    winsock_Ear.Listen(2000); //This is to make the PC act as host
                           //winsock_Ear.Connect("10.2.10.201", 2000);

    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

Socket_winsoc

Emulator

Dashboard

Get started with wizards and quick access to common editors

Type view

All

No Errors

	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;
}

```

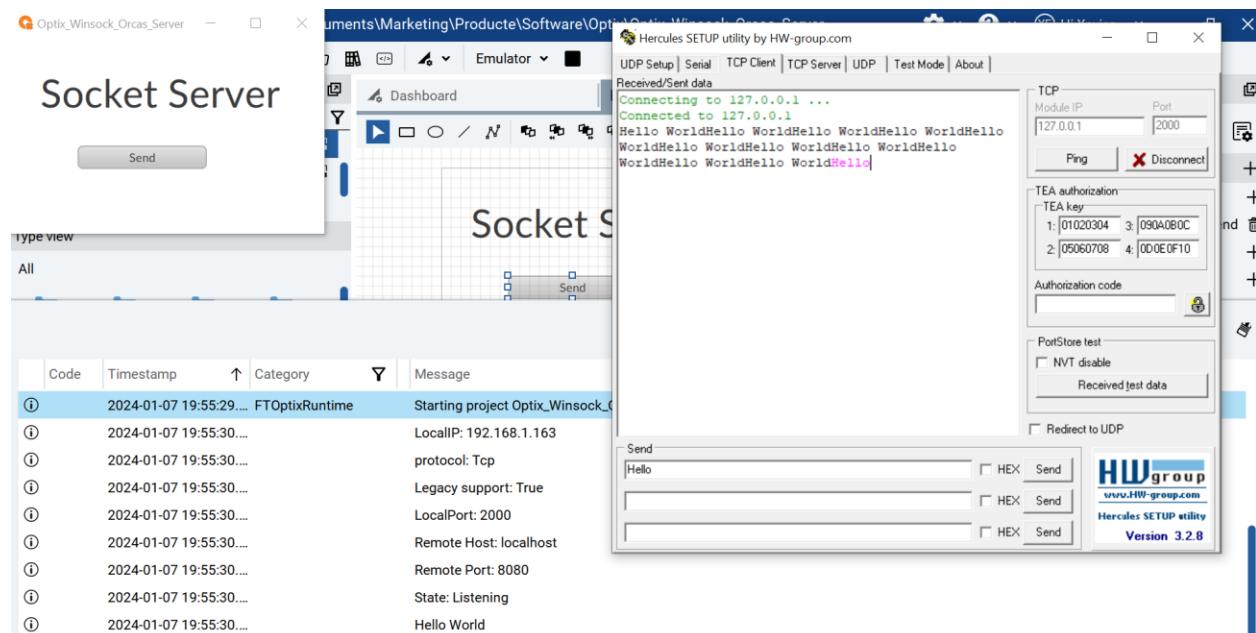
EventArgs e is a parameter called e that contains the event data, see the EventArgs MSDN page for more information.

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/xavierflorensa/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.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("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

38.3. Thirt party libraries (dll) Socket Client application with Optix

You can see the result here

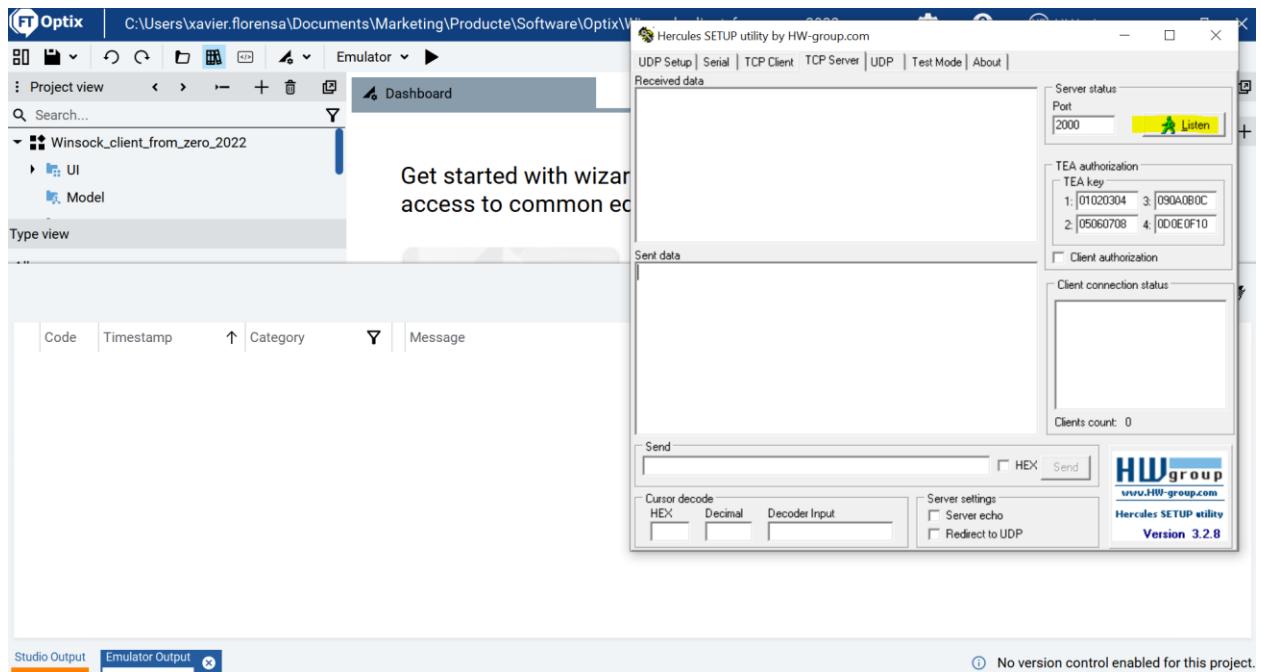
<https://youtu.be/6E4xt02Ohzw>

You can download the project from here

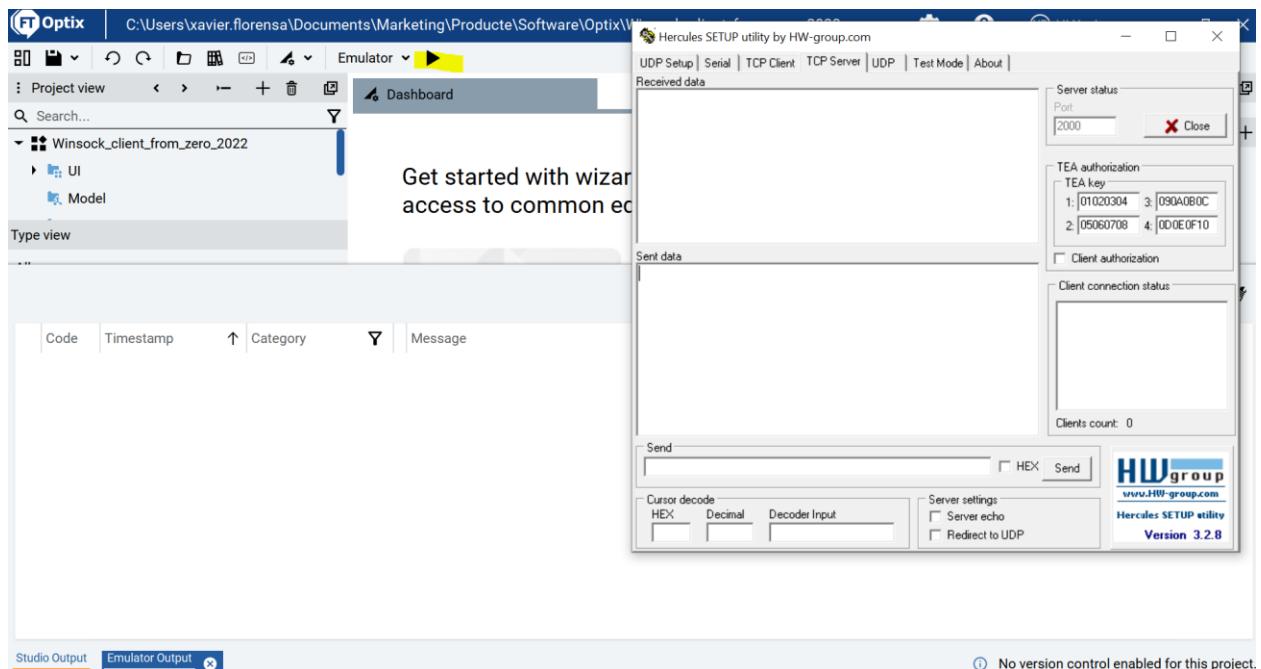
https://github.com/xavierflorensa/Optix_Winsock_Orcas_Client.git

Test the program with a TCP server

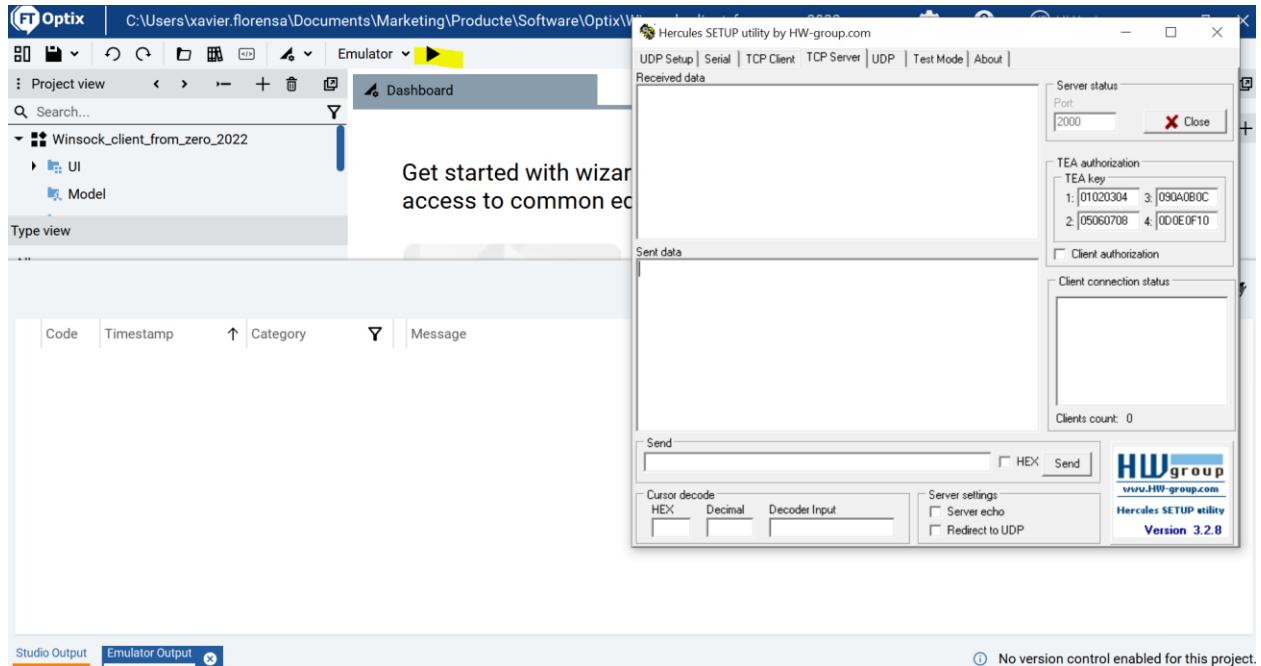
First open the server and put it to listen on port 2000



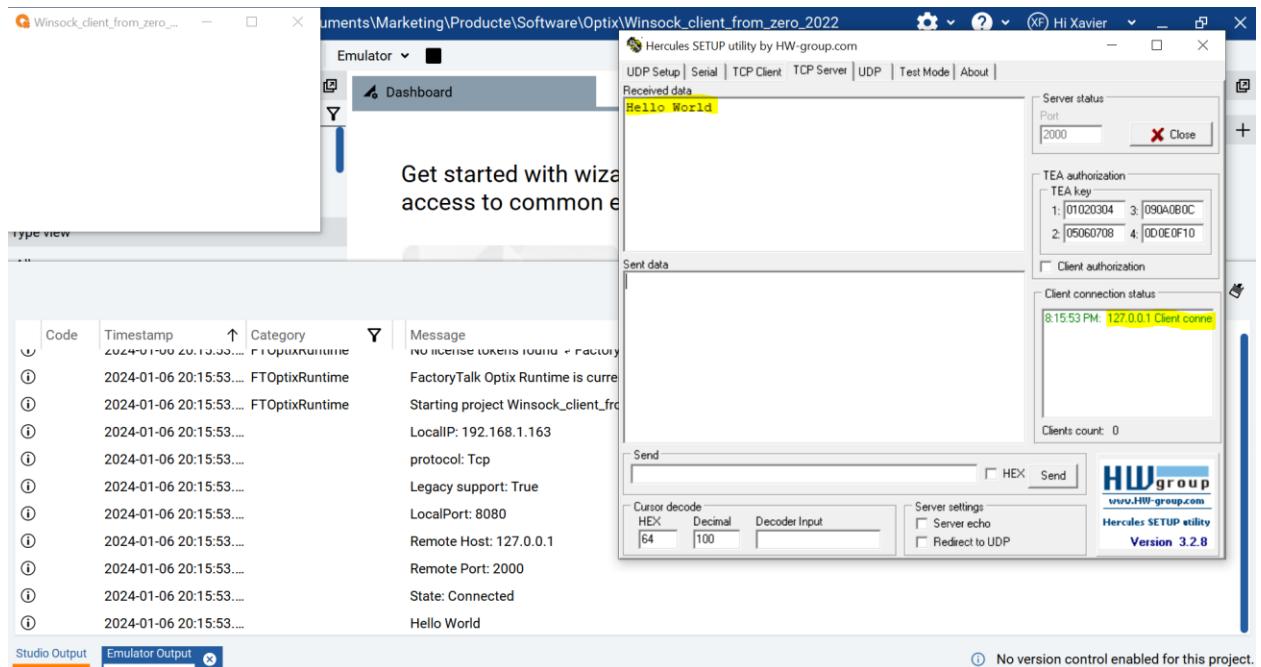
Now the server is listening



Next start the Optix project

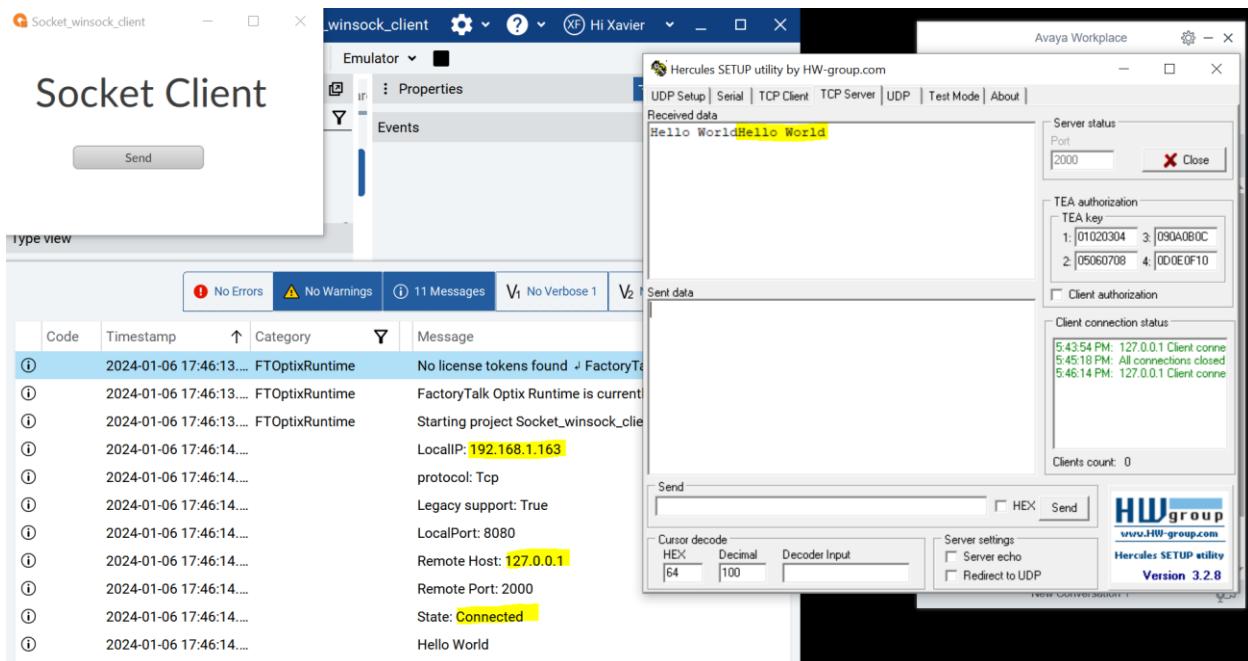


This is working.

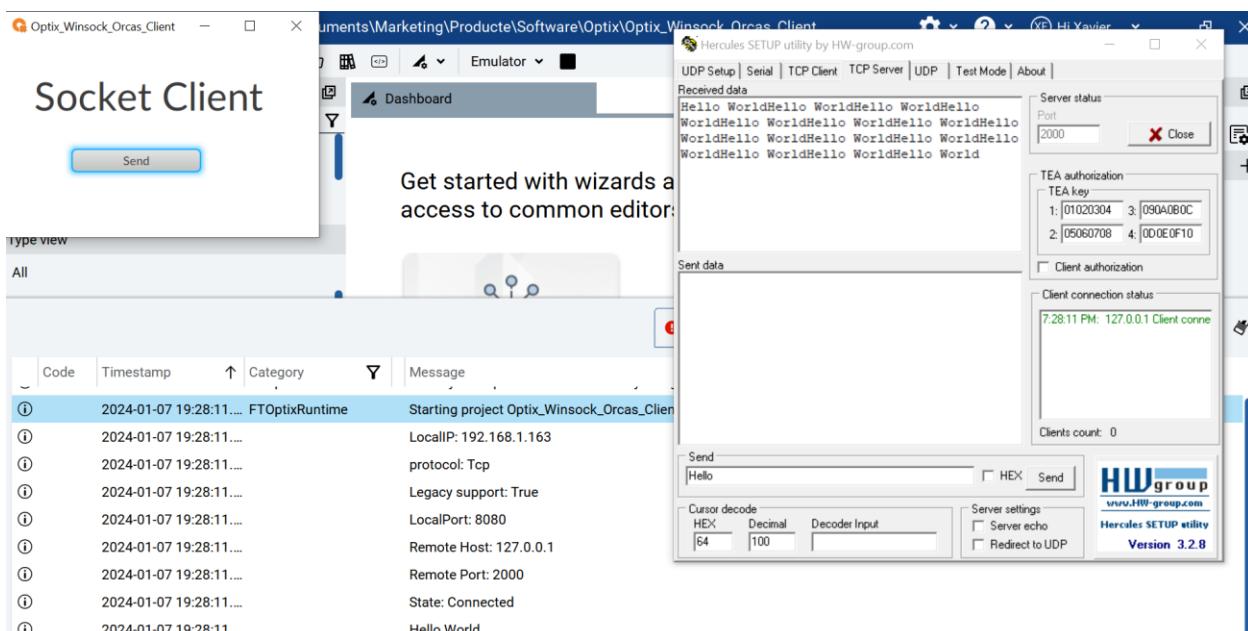


But the connection is made and sending one message is done in one step, and then no chances to send more messages.

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.WinsockConnectionEventArgs 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

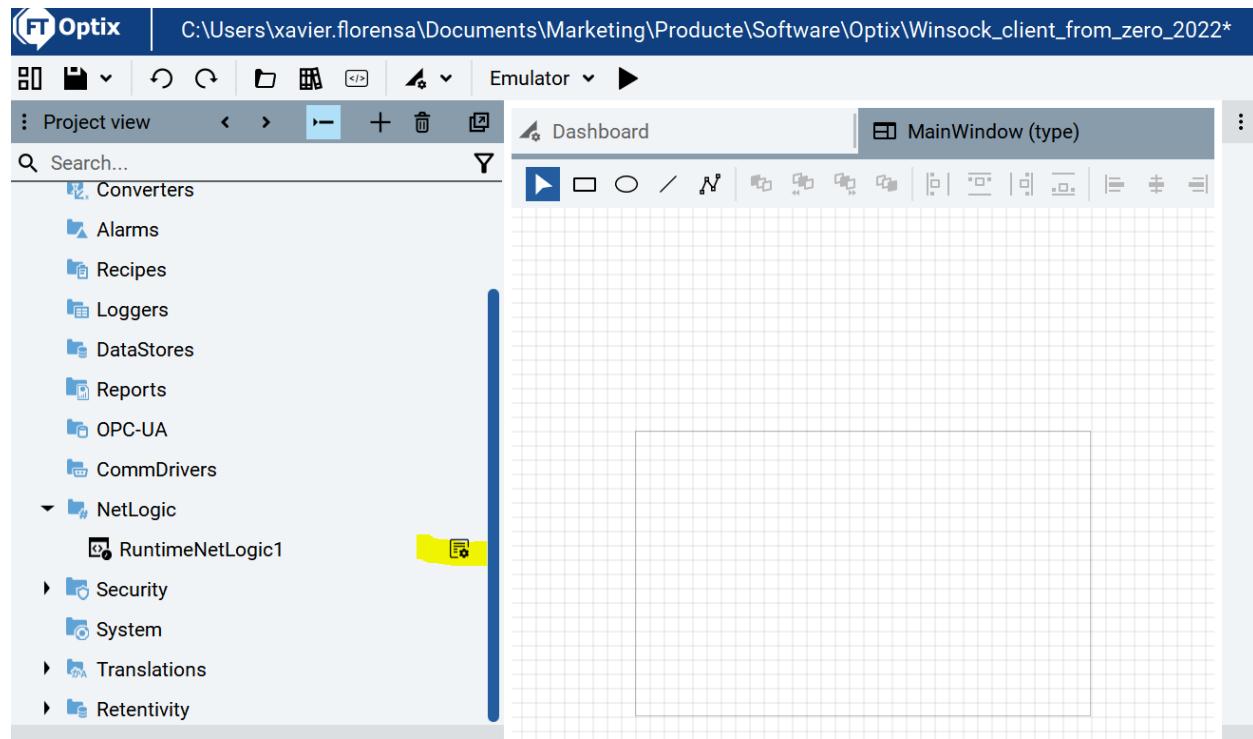
38.3.1. Installing external references (dll libraries) into the project Visual Studio 2022

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

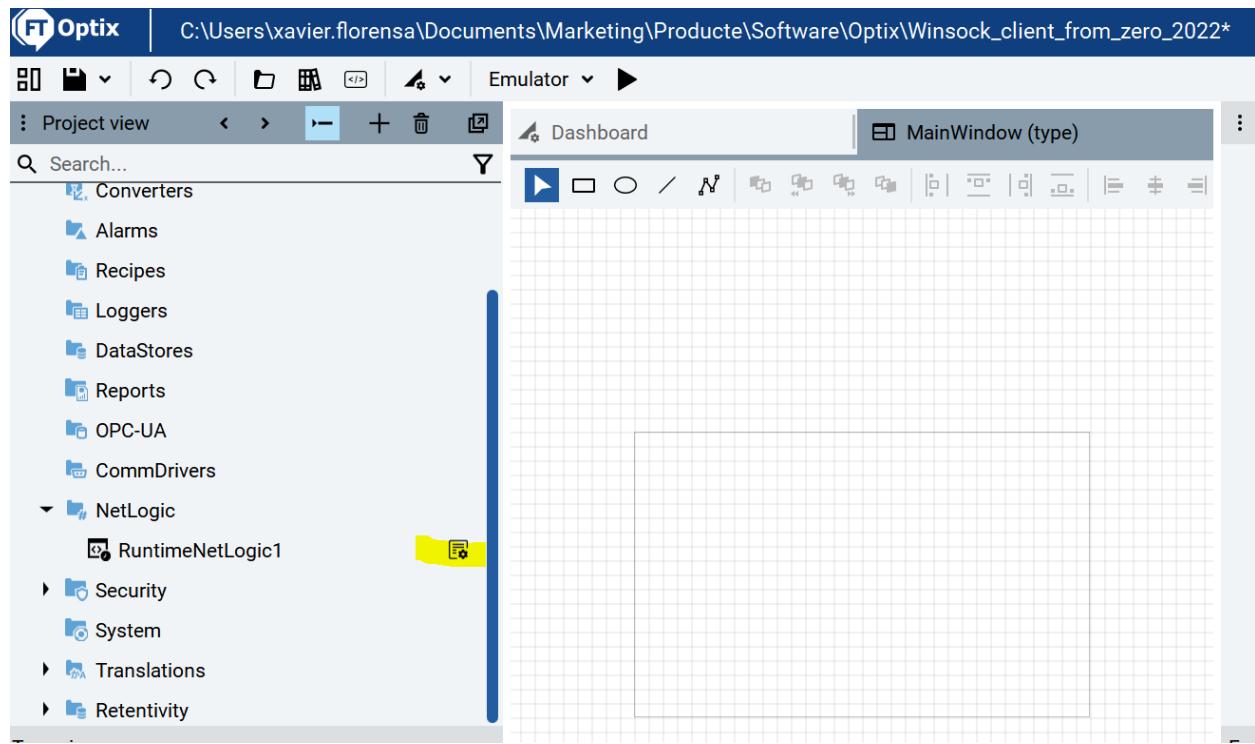
This PC > Documents > Marketing > Producte > Software > Optix > Winsock_client_from_zero_2022

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

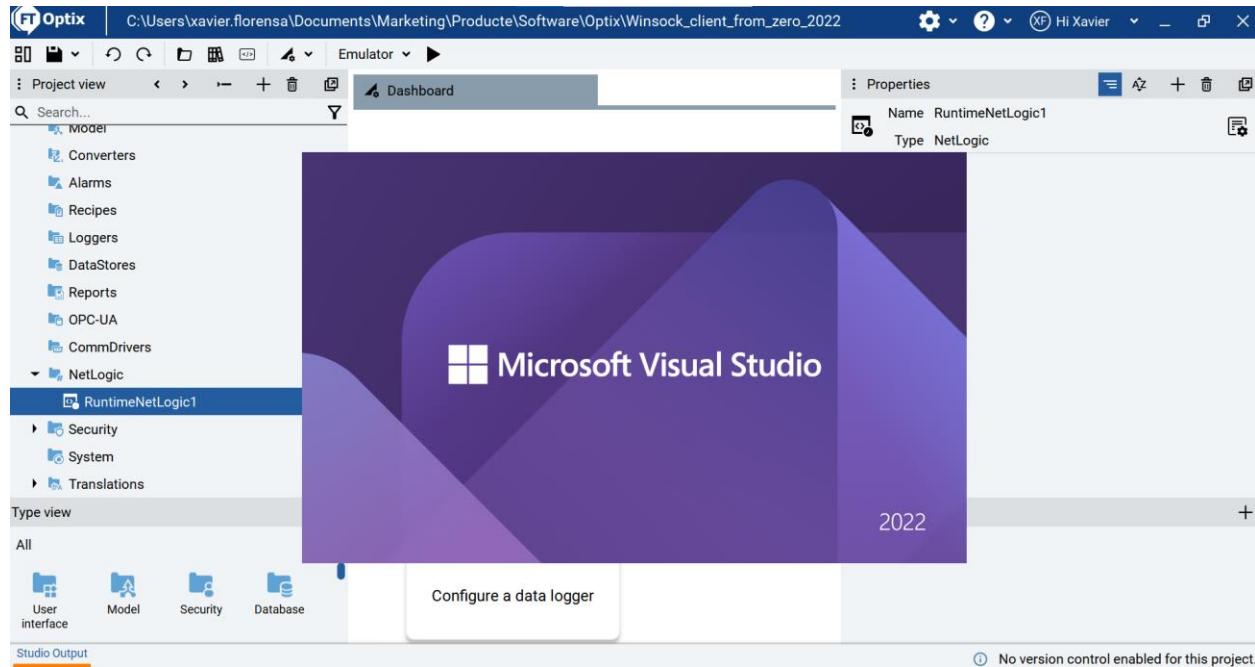
Select editor to Visual Studio 2022

Close and open FT Optix to have this change updated.

Click on the code



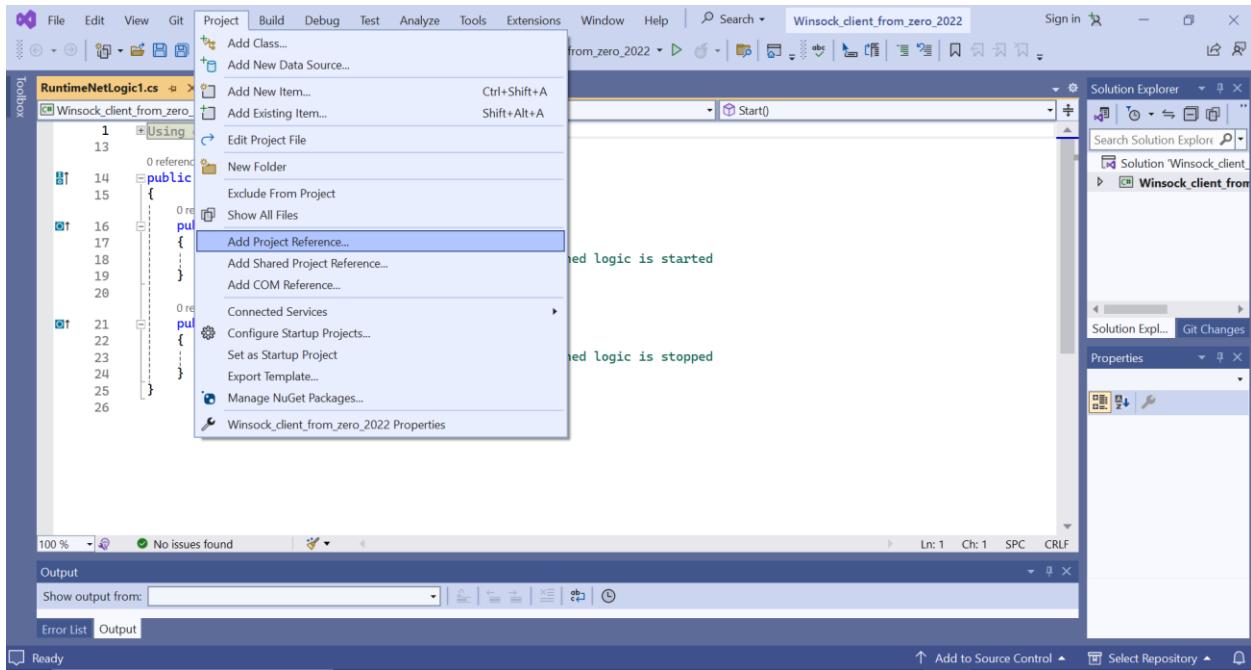
Visual Studio 2022 will open



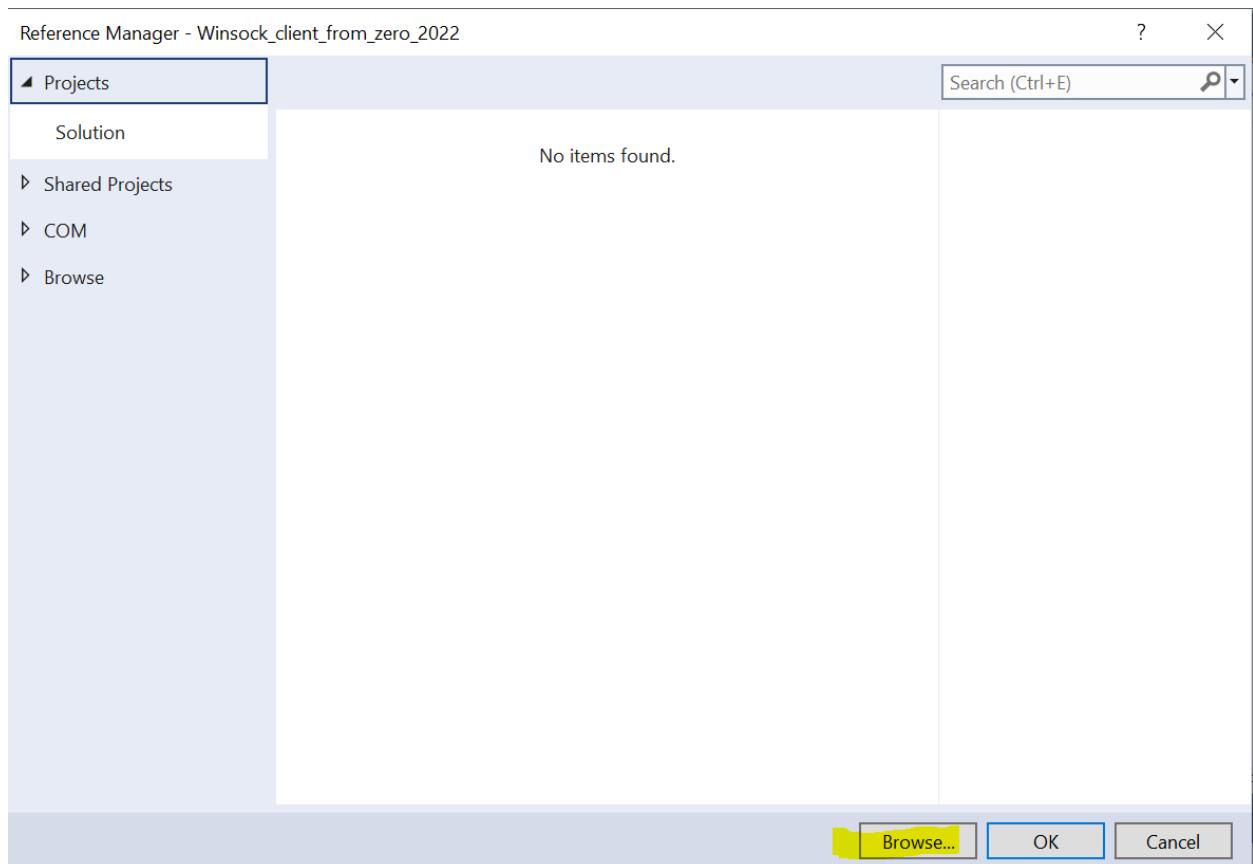
The screenshot shows the Visual Studio IDE interface. The main window displays the code for `RuntimeNetLogic1.cs`. The code defines a class `RuntimeNetLogic1` that inherits from `BaseNetLogic`. It contains two overridden methods: `Start()` and `Stop()`, each with a comment indicating its purpose. The Solution Explorer on the right shows a solution named "Winsock_client_from_zero_2022" containing a project named "Winsock_client_from_zero_2022". The Properties and Tools windows are also visible.

```
1  Using directives
13
14  0 references
15  public class RuntimeNetLogic1 : BaseNetLogic
16  {
17      0 references
18          public override void Start()
19      {
20          // Insert code to be executed when the user-defined logic is started
21      }
22
23      0 references
24          public override void Stop()
25      {
26          // Insert code to be executed when the user-defined logic is stopped
27      }
28}
```

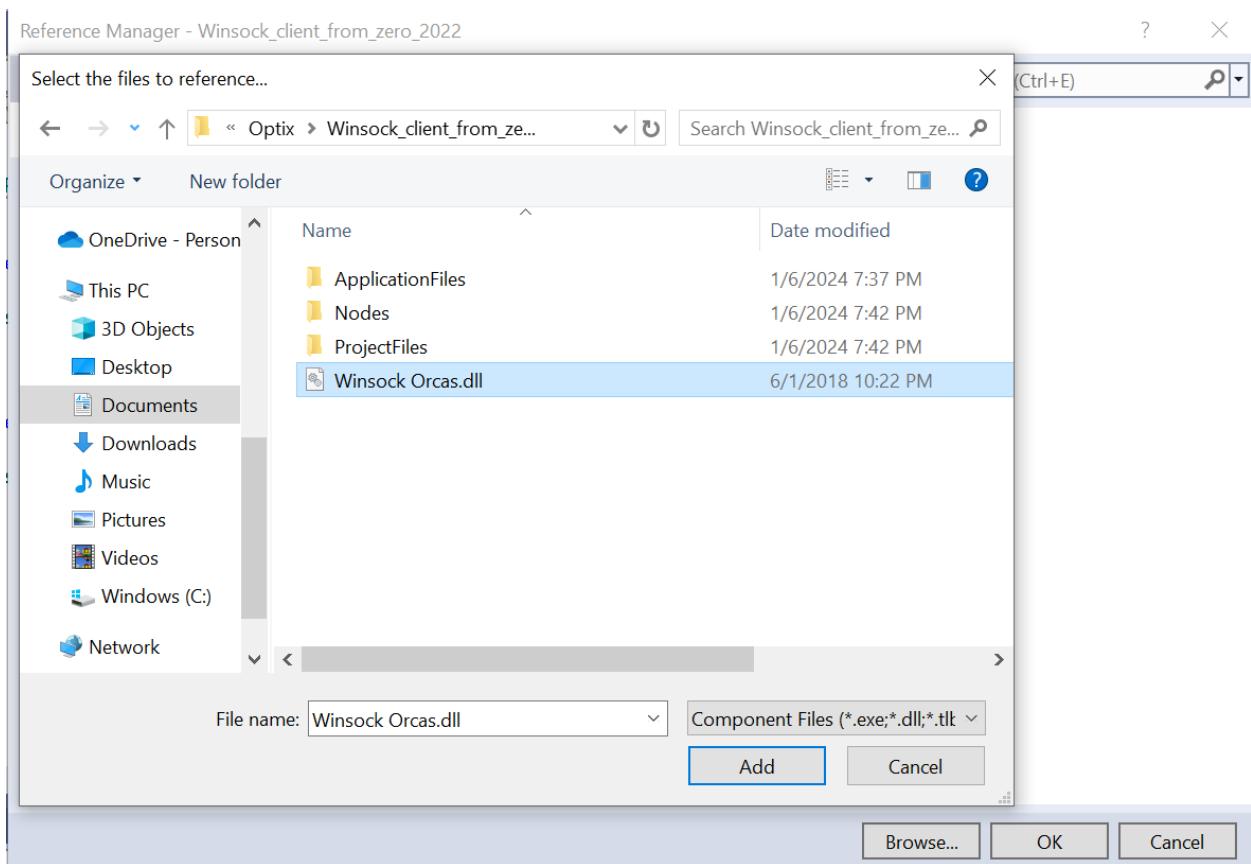
Go to Project / Add Project Reference



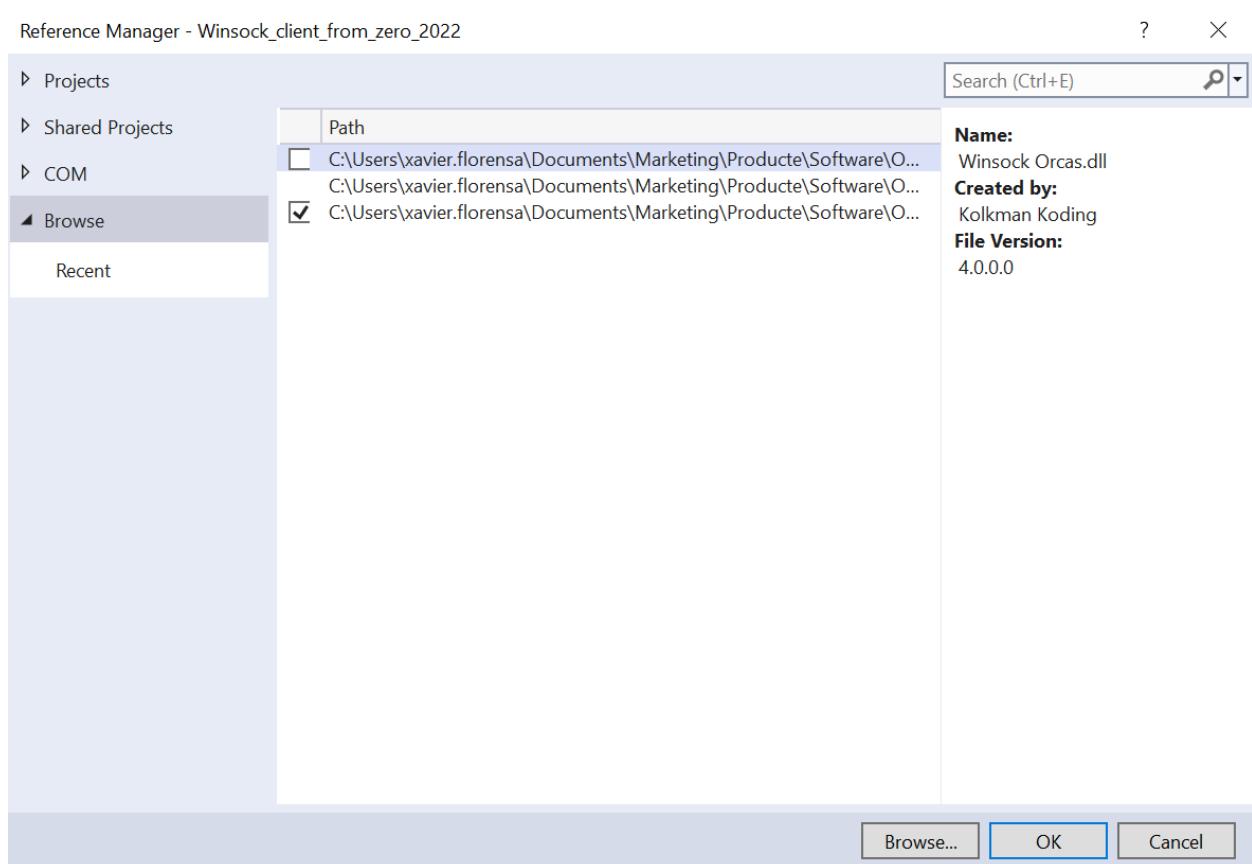
Click on Browse



Select the dll

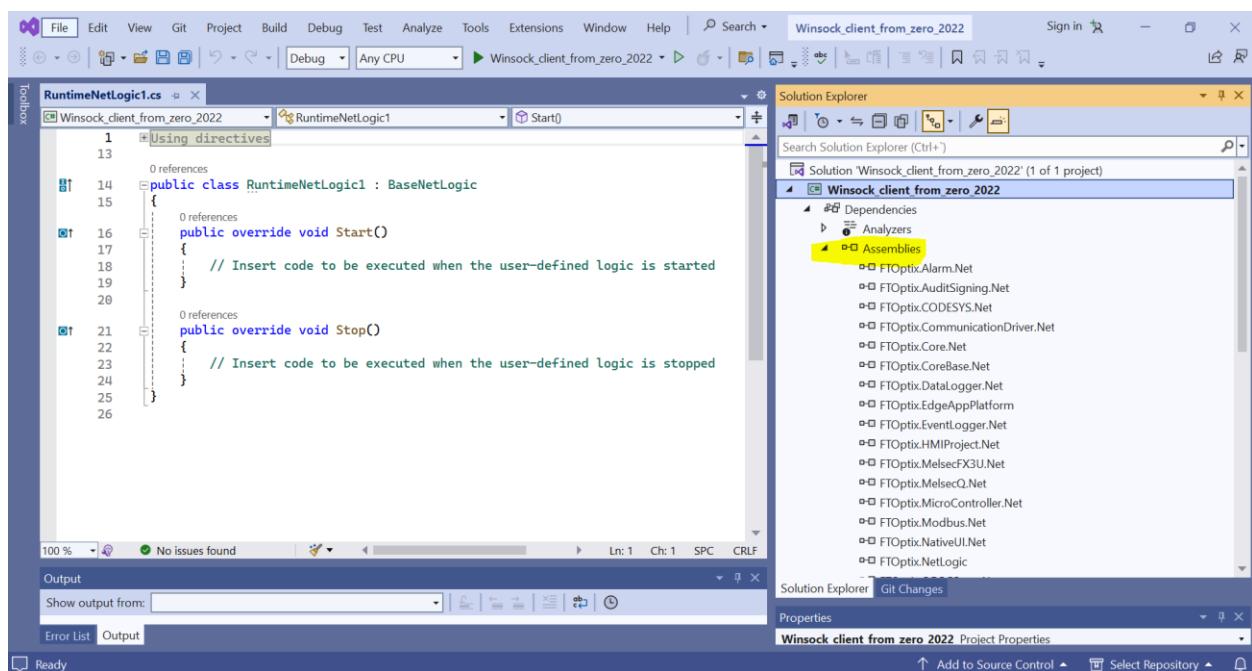


Verify that this is the right path

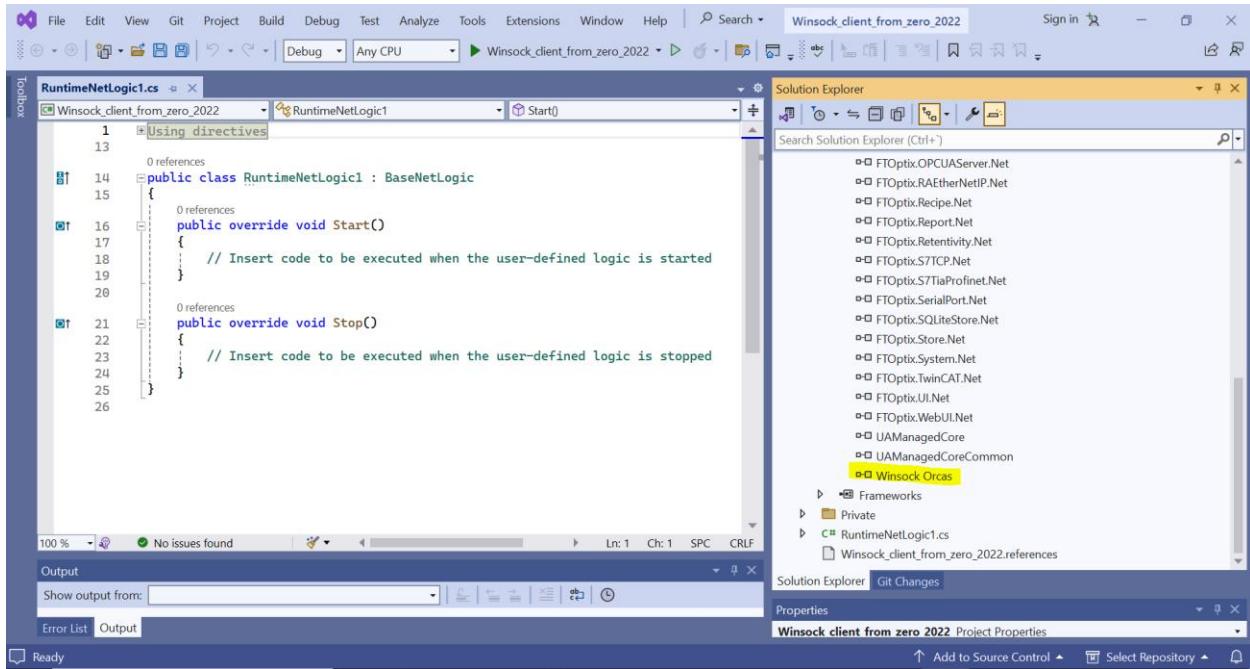


Click Ok

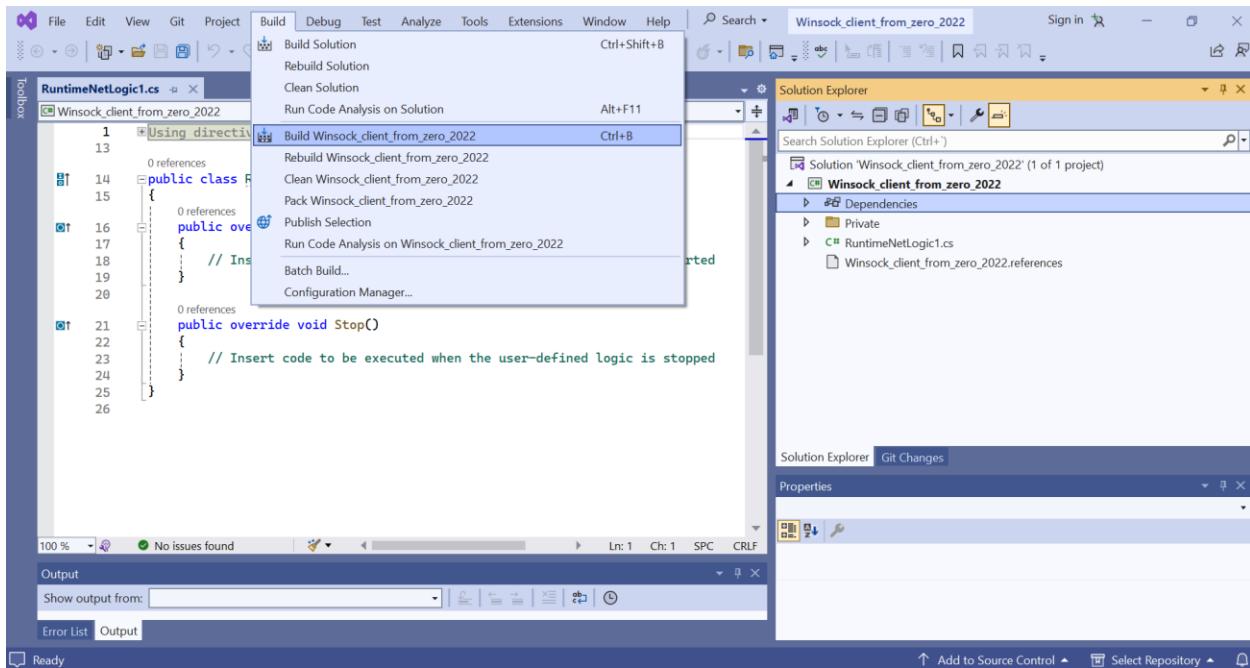
Unfold assemblies



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. The code editor displays the file `RuntimeNetLogic1.cs` which contains C# code for a logic class. The Solution Explorer on the left shows the project structure, including the `Winsock_client_from_zero_2022.csproj` file. The status bar at the bottom indicates "Projects: 1".

```

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 }

```

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 `Winsock_client_from_zero_2022.csproj` file open in the code editor. A red circle highlights the `<Reference Include="Winsock Orcas">` entry in the XML code, indicating where the external DLL was added.

```

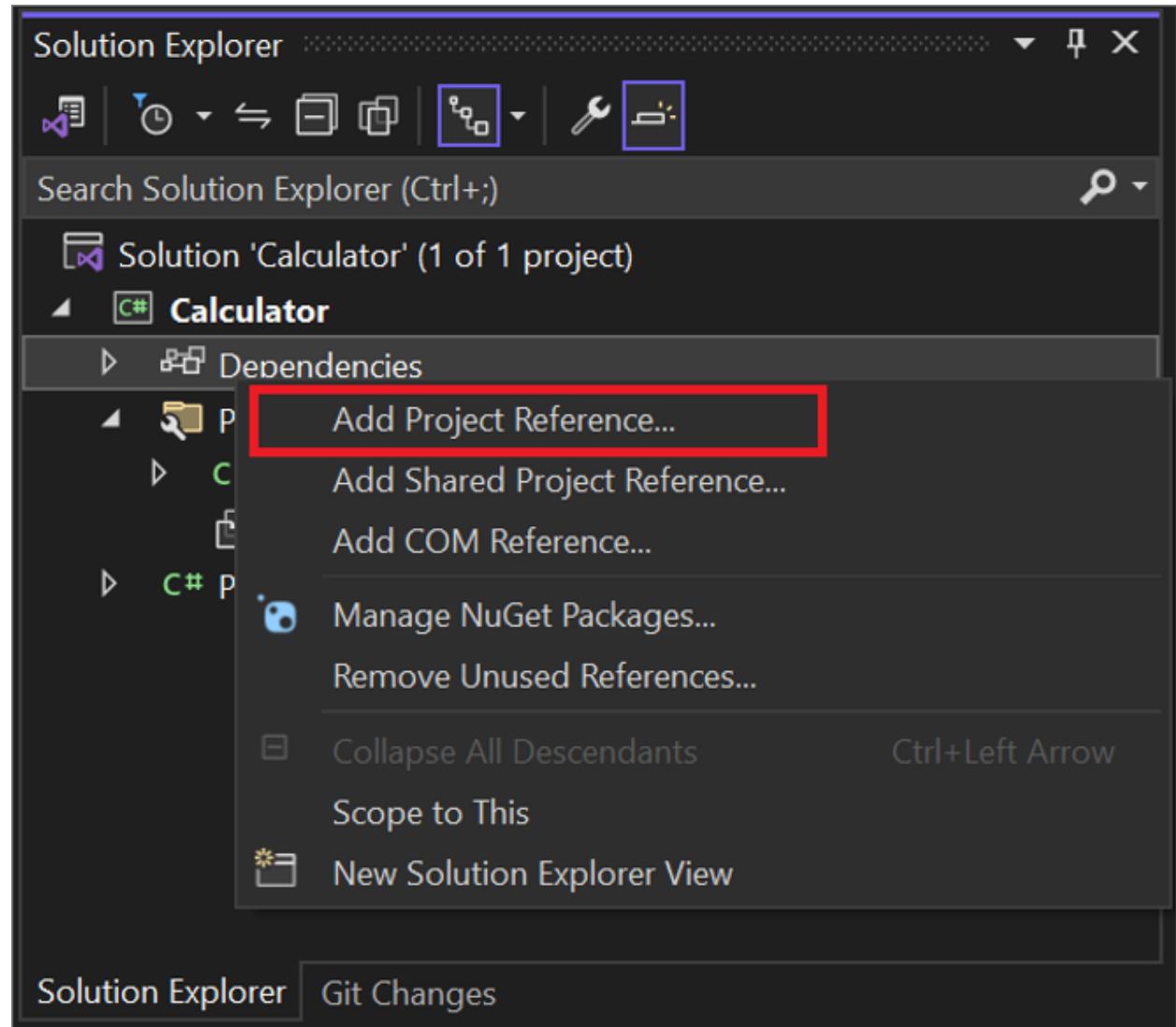
1 <Project Sdk="Microsoft.NET.Sdk">
2   <PropertyGroup>
3     <TargetFramework>net6.0</TargetFramework>
4     <AppendTargetFrameworkToOutputPath>false</AppendTargetFrameworkToOutputPath>
5     <CopyLocalLockFileAssemblies>true</CopyLocalLockFileAssemblies>
6   </PropertyGroup>
7   <PropertyGroup Condition="$(Configuration)|$(Platform) == '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>

```

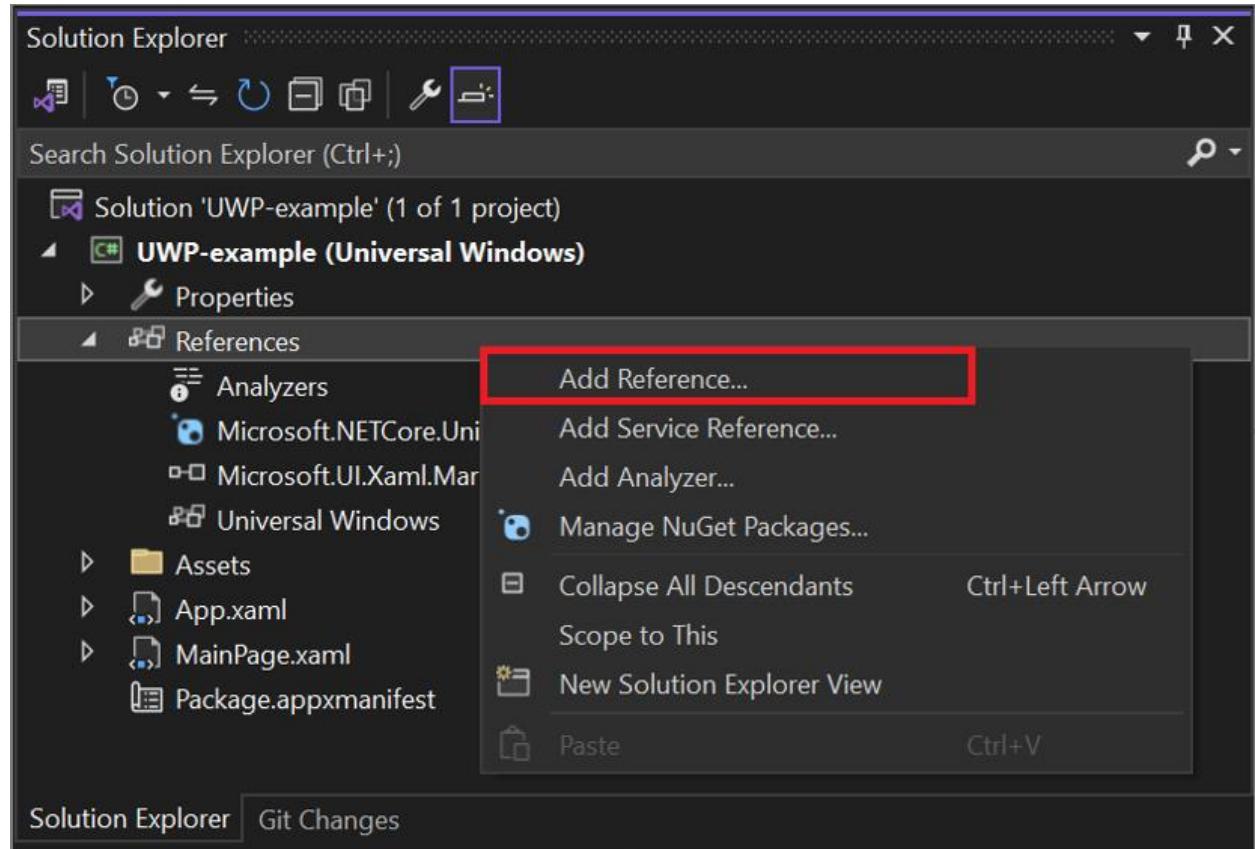
38.3.2. Installing external references (dll libraries) into the project with Visual Studio code
Pending to test this way

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](#).

39. Using NuGet libraries

Let's try to use this fancy library,

<https://github.com/hivemq/hivemq-mqtt-client-dotnet>

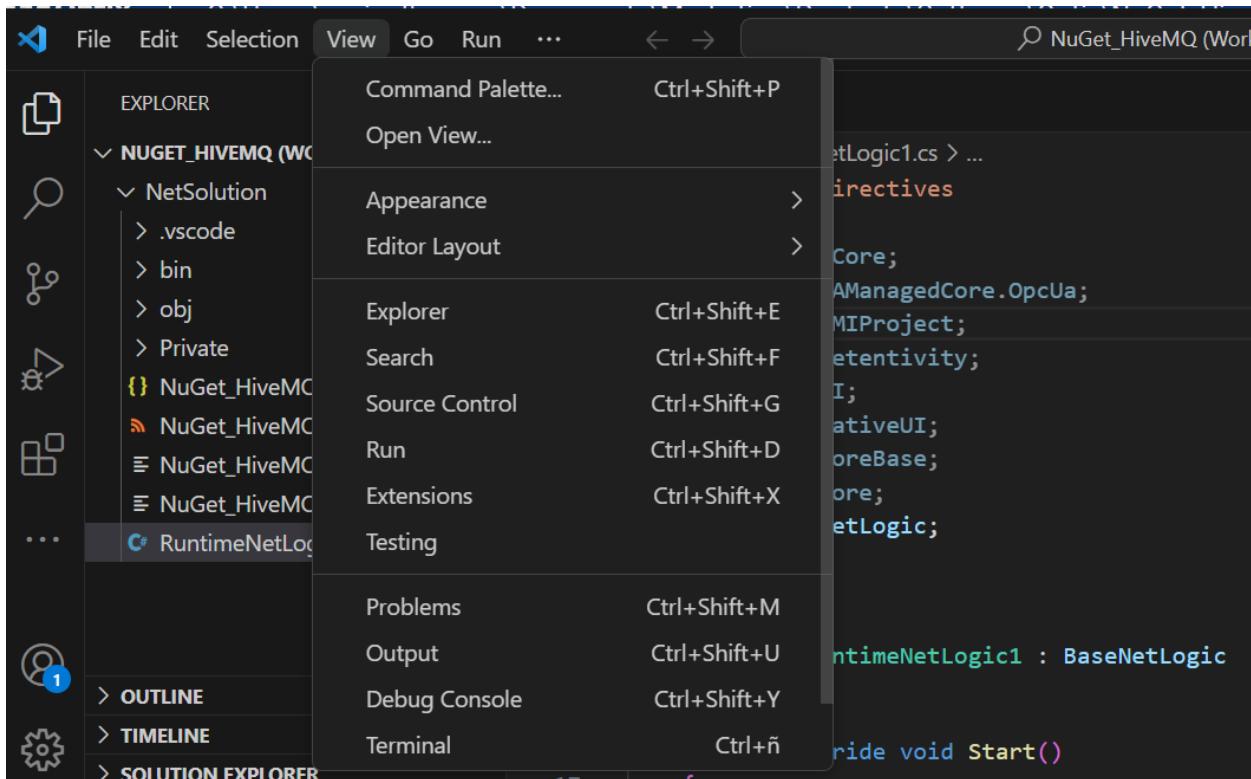
You can see the result on this video

<https://youtu.be/3g81mO2wqXQ>

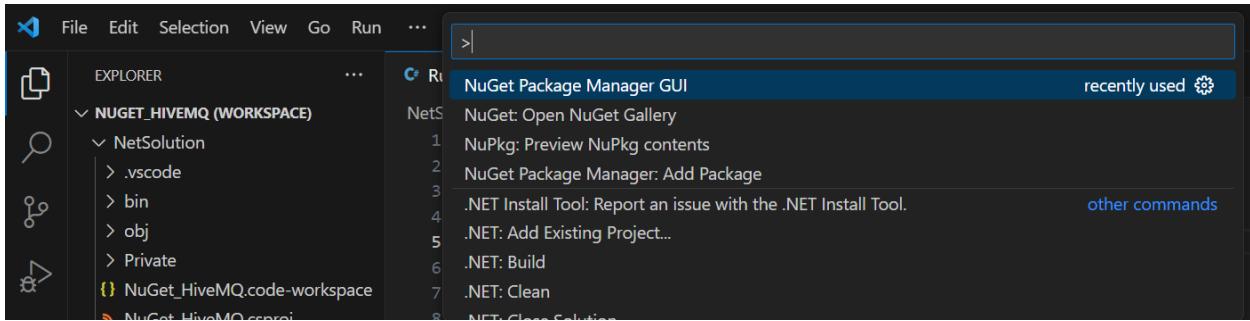
So create a new project and add a new Netlogic Runtime code

Open the code

Click on View/Command Pallete

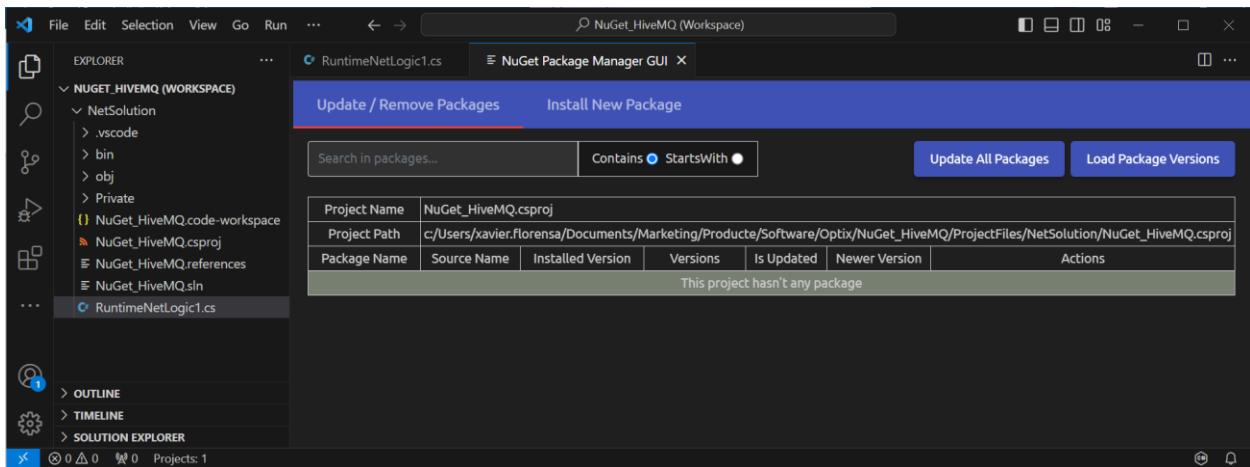


You will see this depending on what VScode extensions you have installed

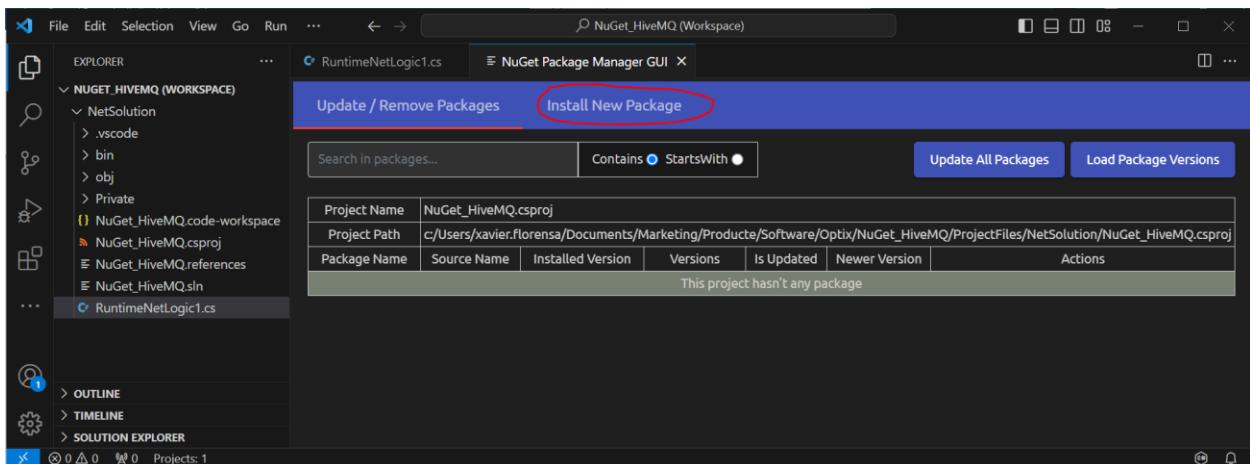


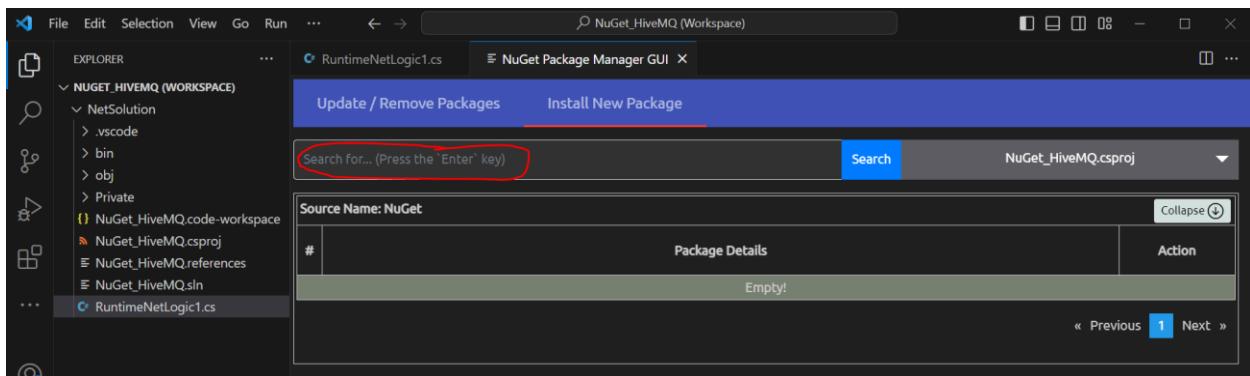
Select NuGet Package Manager GUI

You will see this

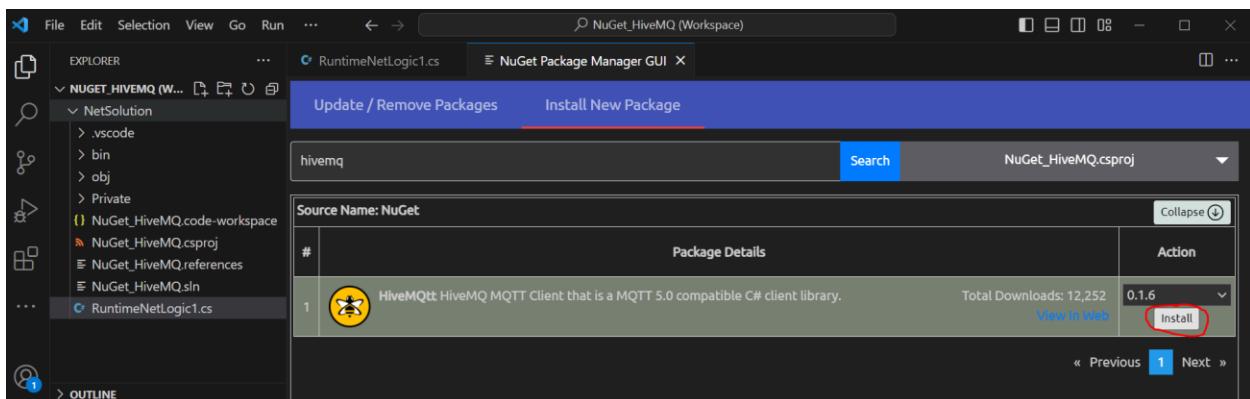


Click on Install new package

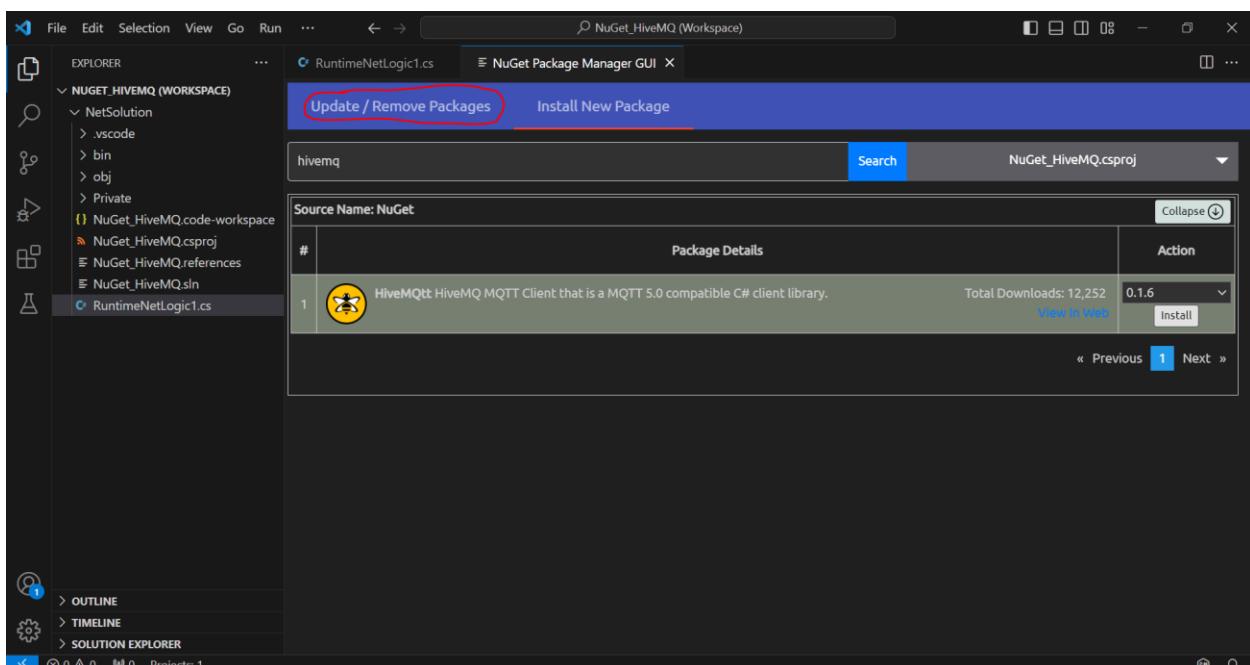




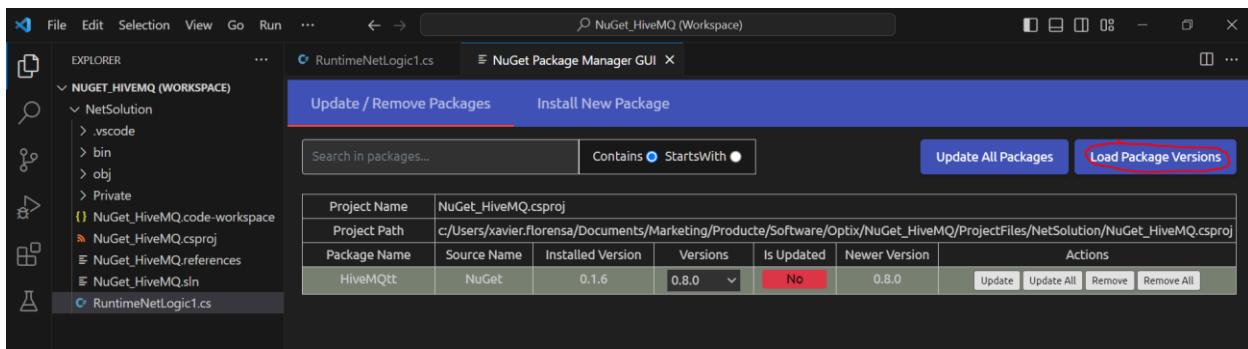
Search for hivemq, and click on install



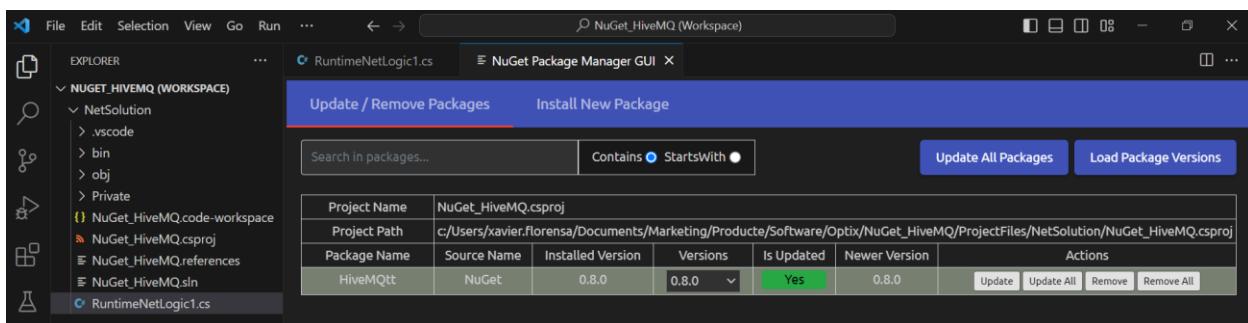
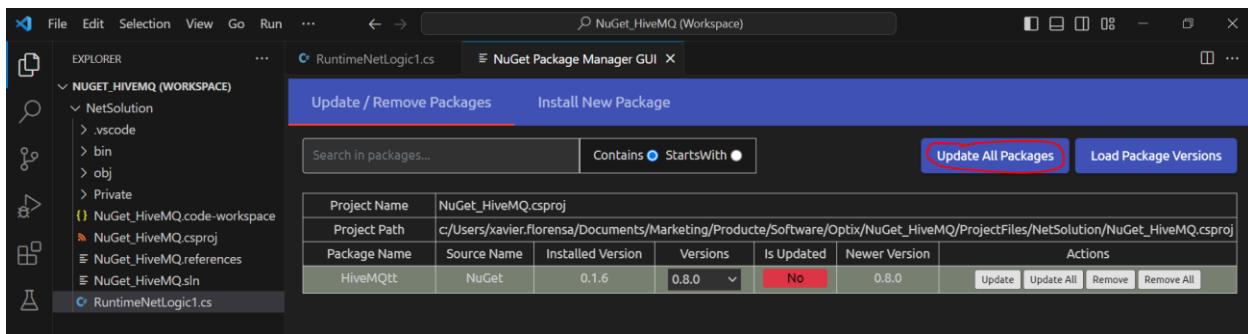
This takes just a second, now go to Update



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.HMIProject;
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").WithPort(1883).WithUser("user").WithPassword("password");
        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").WithPort(1883).WithUser("user").WithPassword("password");
    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").WithClientName("Node-RED");
    var client = new HiveMQClient(options);
    // Connect to the MQTT broker
    //var connectResult = await client.ConnectAsync();
    var connectResult = await client.ConnectAsync();
}

```

. 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").WithClientName("Node-RED");
    var client = new HiveMQClient(options);
    // Connect to the MQTT broker
    //var connectResult = await client.ConnectAsync();
    var connectResult = await client.ConnectAsync();
}

```

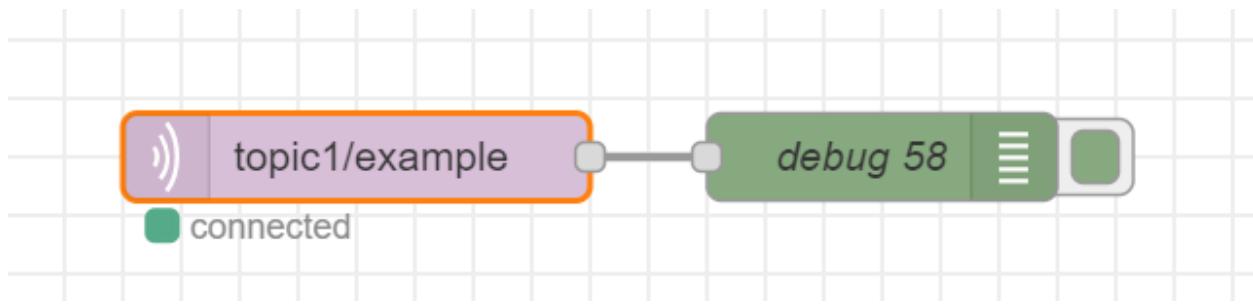
and so on until you get no errors, be aware of the change the compiler has made for you

```

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").WithClientName("Node-RED");
        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");
    }
}

```

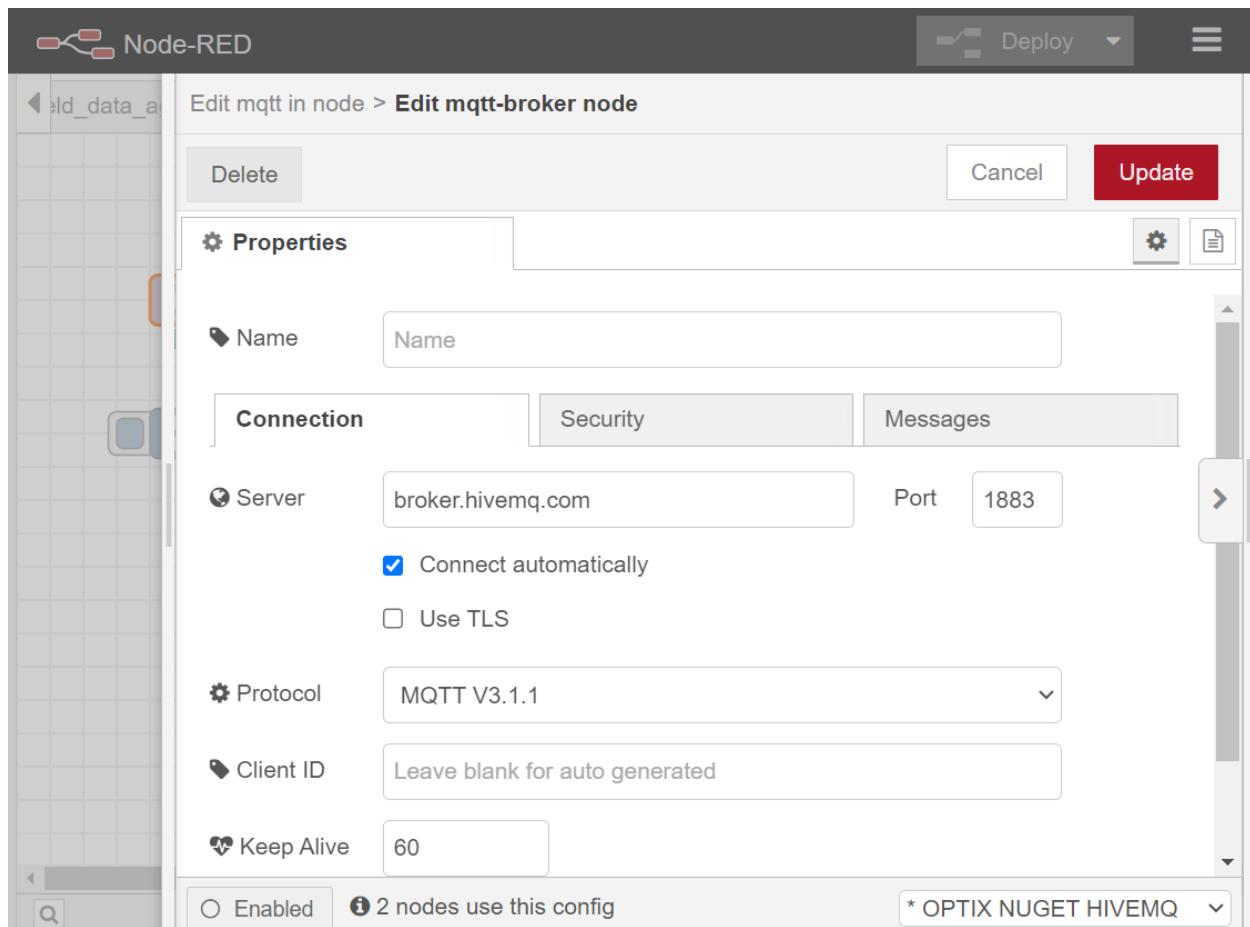
Then save and before executing open a node-red instance to verify the Optix project is working fine



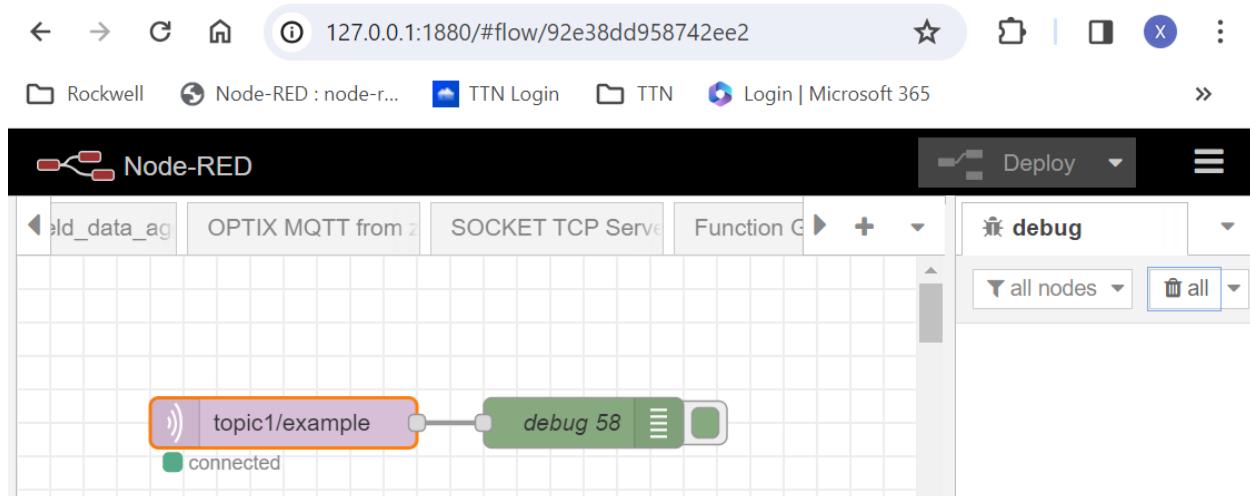
A screenshot of the "Edit mqtt in node" configuration dialog in Node-RED. The dialog has a header with "Delete", "Cancel", and "Done" buttons, and a "Properties" tab selected. The properties are listed as follows:

- Server**: broker.hivemq.com:1883
- Action**: Subscribe to single topic
- Topic**: topic1/example
- QoS**: 2
- Output**: auto-detect (parsed JSON object, string or t)
- Name**: Name

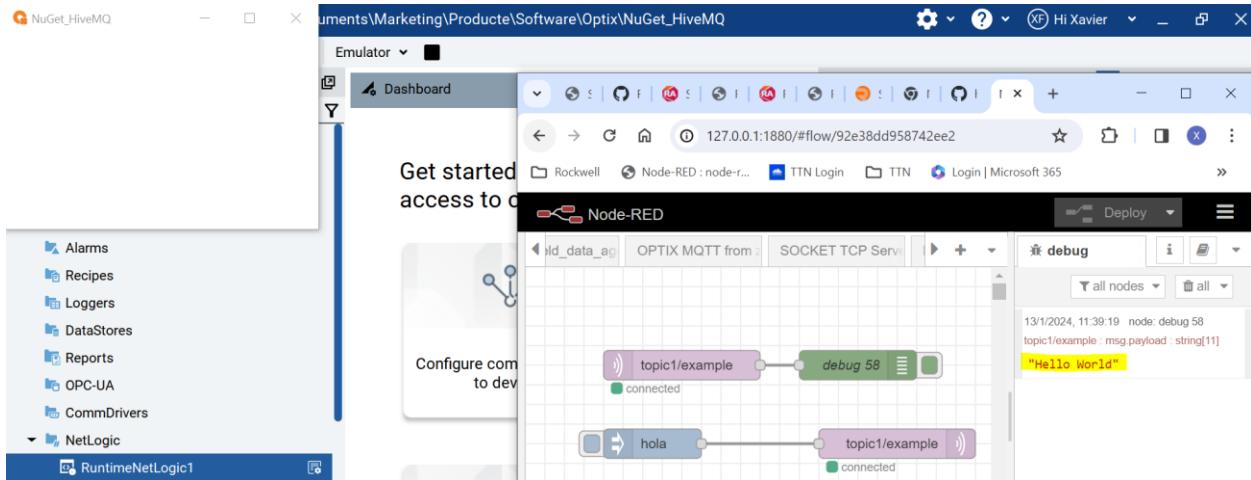
At the bottom left is a checkbox for "Enabled". On the right side of the dialog, there are three icons: a gear, a document, and a square.



Prepare the node-red instance



Then execute the Optix application



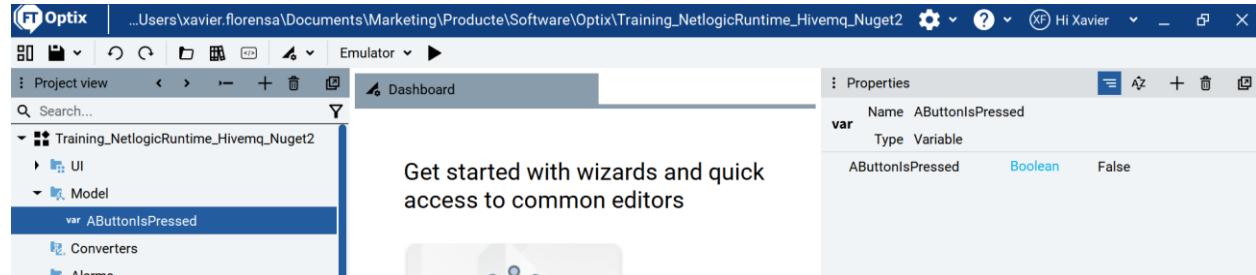
That's all.

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.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using HiveMQ.Client;
using HiveMQ.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()

```

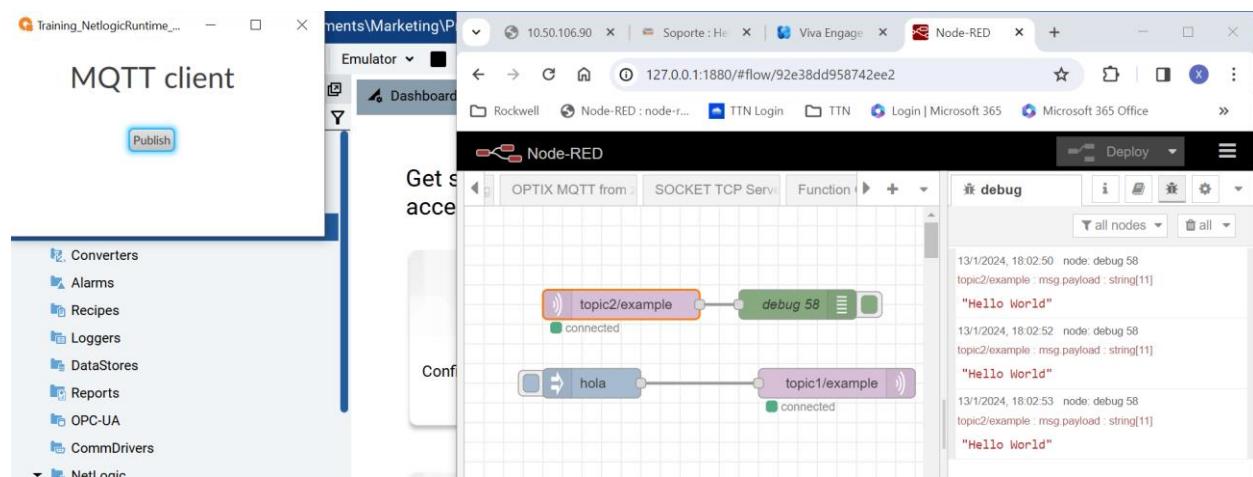
```

    {
        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>

40. Modbus server with FTOptix

Here the dll library EasyModbusTCP is used

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

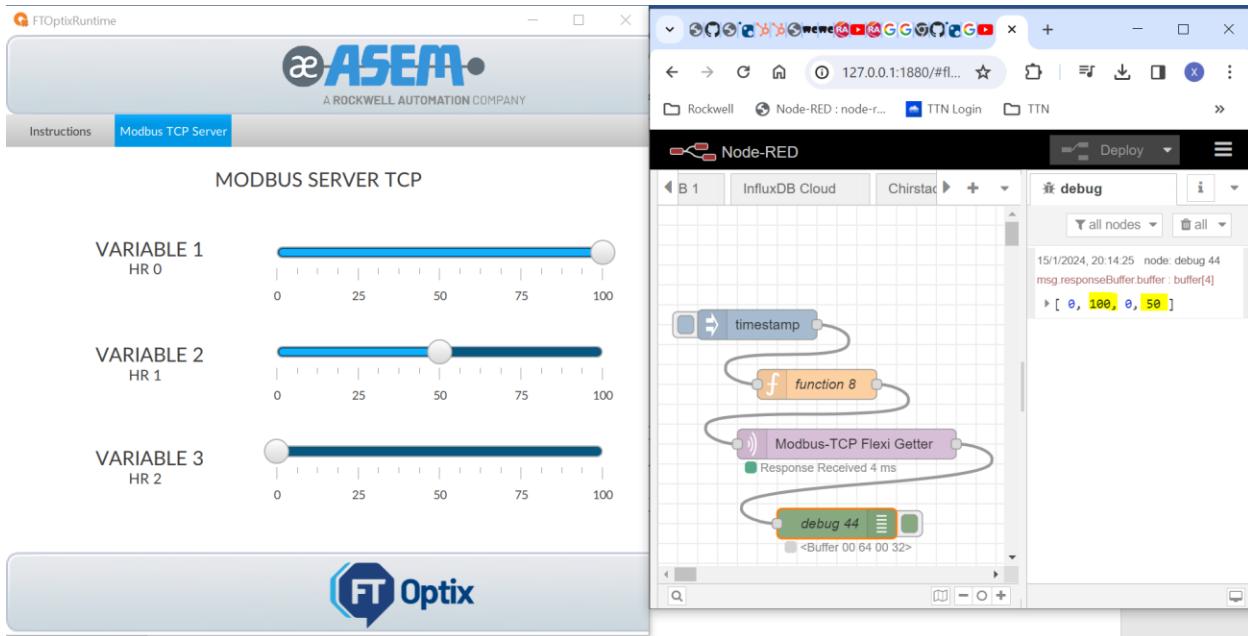
You can see how to use this library (with a client)

<https://www.youtube.com/watch?v=qcyJoDu7cok&t=726s>

You can download the Server FactoryTalk Optix code from here

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

Let's run the Optix application, and open a Modbus client



Let's try to build a Modbus client application with FT Optix and the EasyModbus library

The screenshot shows the 'Edit function node' dialog for 'function 8'. It includes tabs for 'Properties', '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

Let's try to build a Modbus client application with FT Optix and the EasyModbus library

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 and 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.

Change the editor to VS code

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

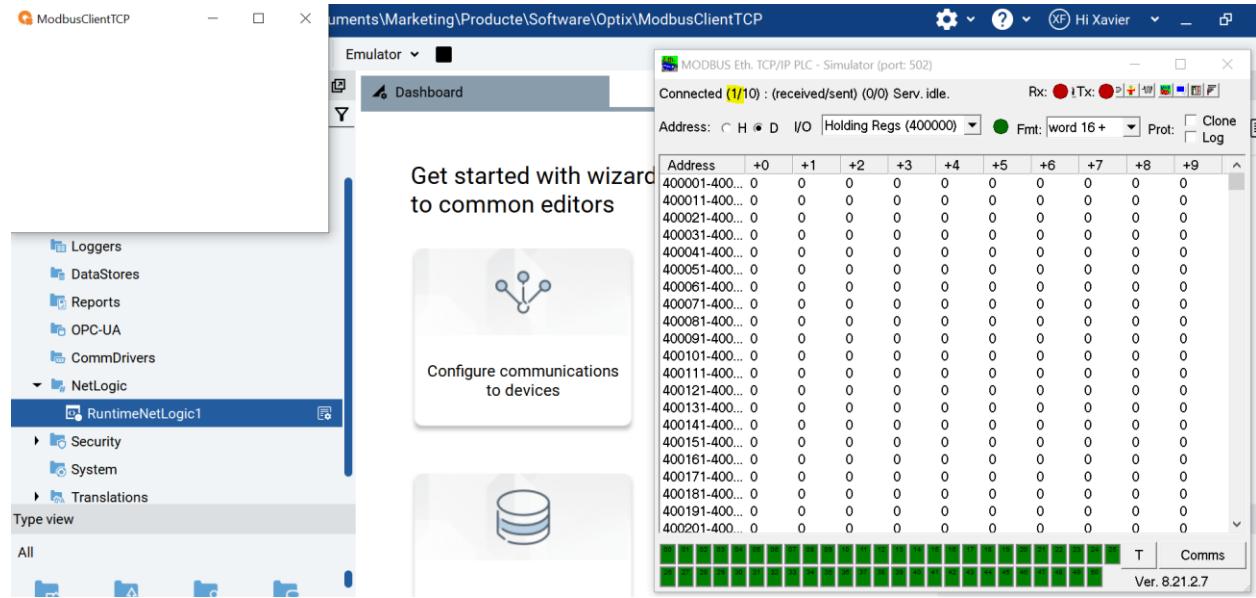
```
#region Using directives
using System;
using UAManagerCore;
using OpcUa = UAManagerCore.OpcUa;
using FTOptix.HMIPrject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using EasyModbus;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        ModbusClient myModbusClient = new ModbusClient("127.0.0.1", 502);
        myModbusClient.Connect();
        // 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

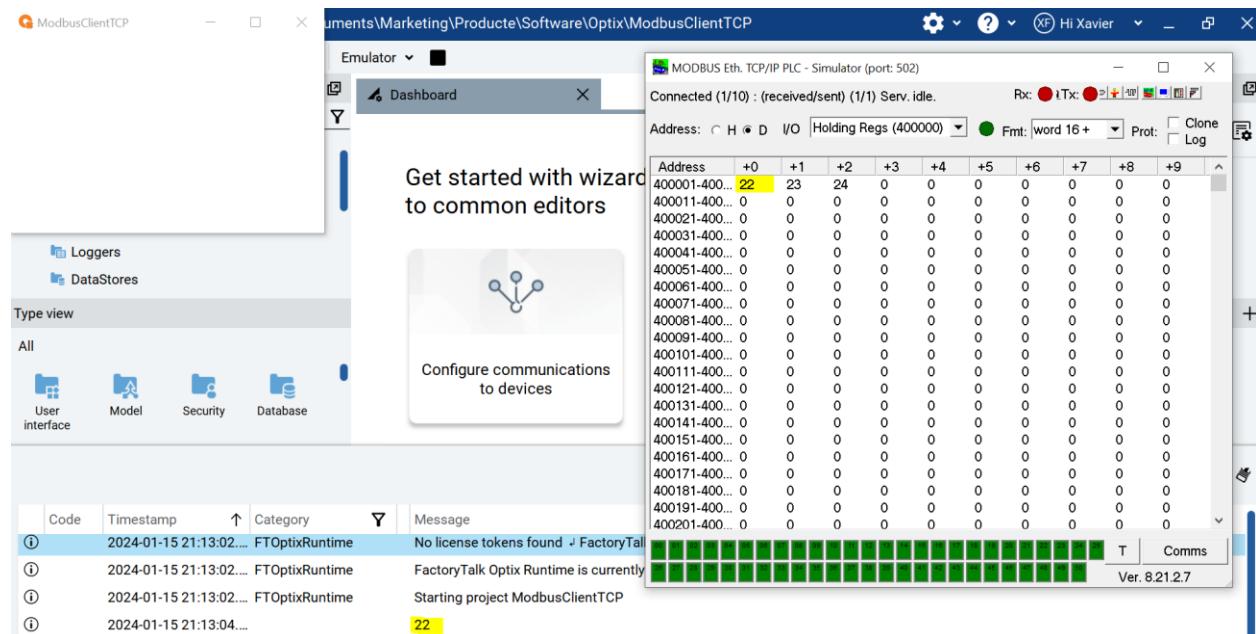
Opening a Server, we are connected to the server



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 UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIPrjject;
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;
```

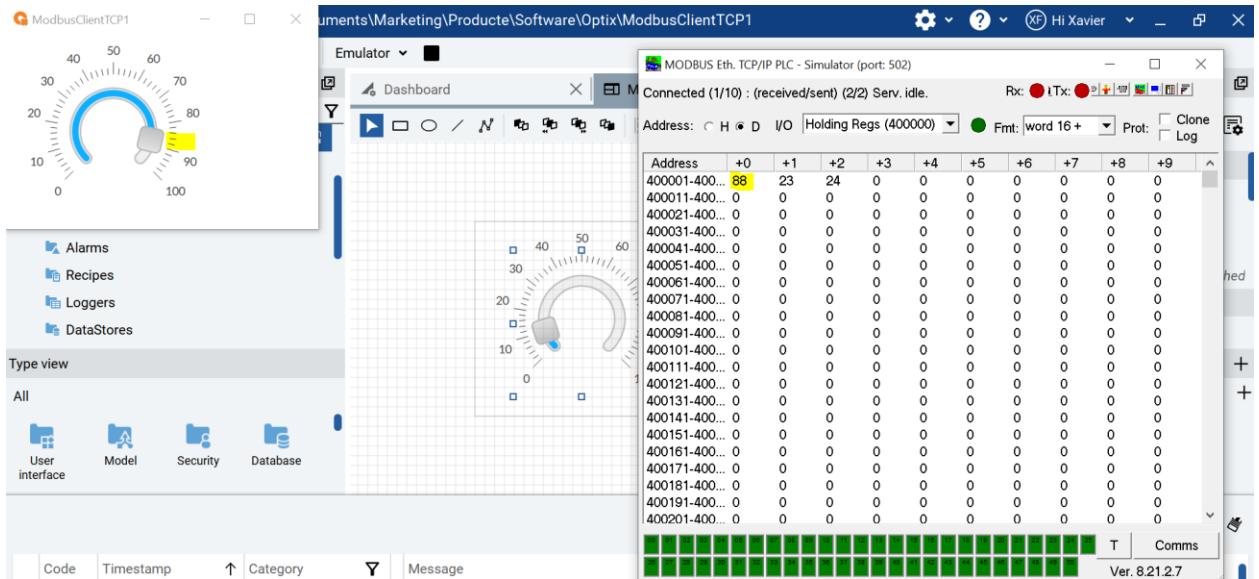
```

using 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();
        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