FactoryTalk Optix and MQTT

Contents

1. Configuring an application as an MQTT client

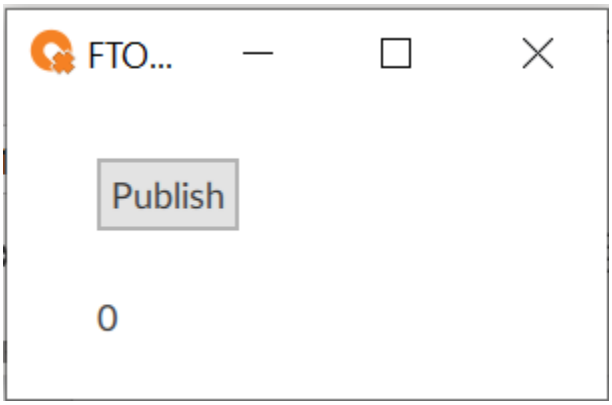
This example is using a cloud public broker like test.mosquitto.org

The Optix application is both subscribed and publishing to the same topic

file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/developing-solutions/app-ex/netlogic/mqtt-client/Configure-an-application-as-an-mqtt-client.html

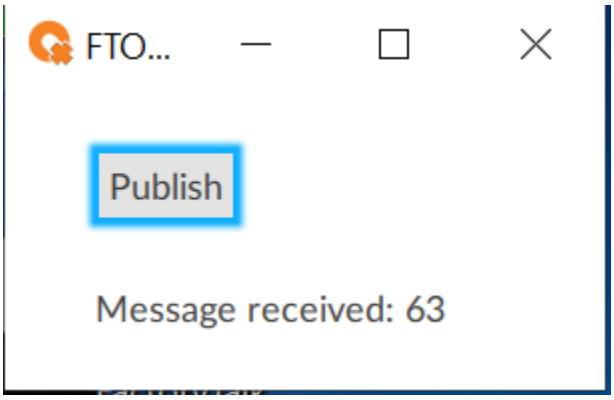
Tip: You can download a sample project from: MQTTClient.zip

Let's try opening the example



Click on "Publish"

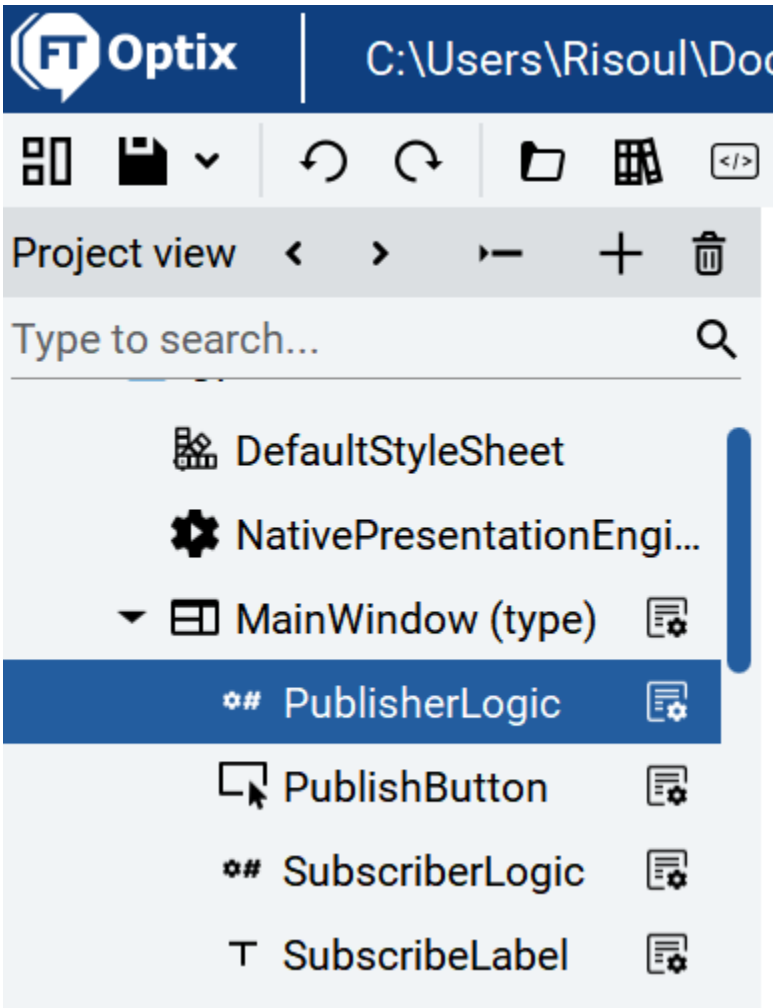
It works



But let’s look at the details

Like PublisherLogic

Doubleclick on PublisherLogic



Visual Studio code opens

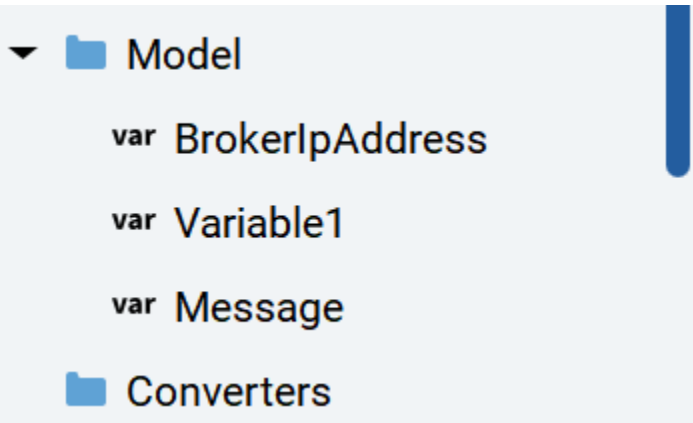
File Edit Selection View Go Run ··· MQTTClient (Workspace)

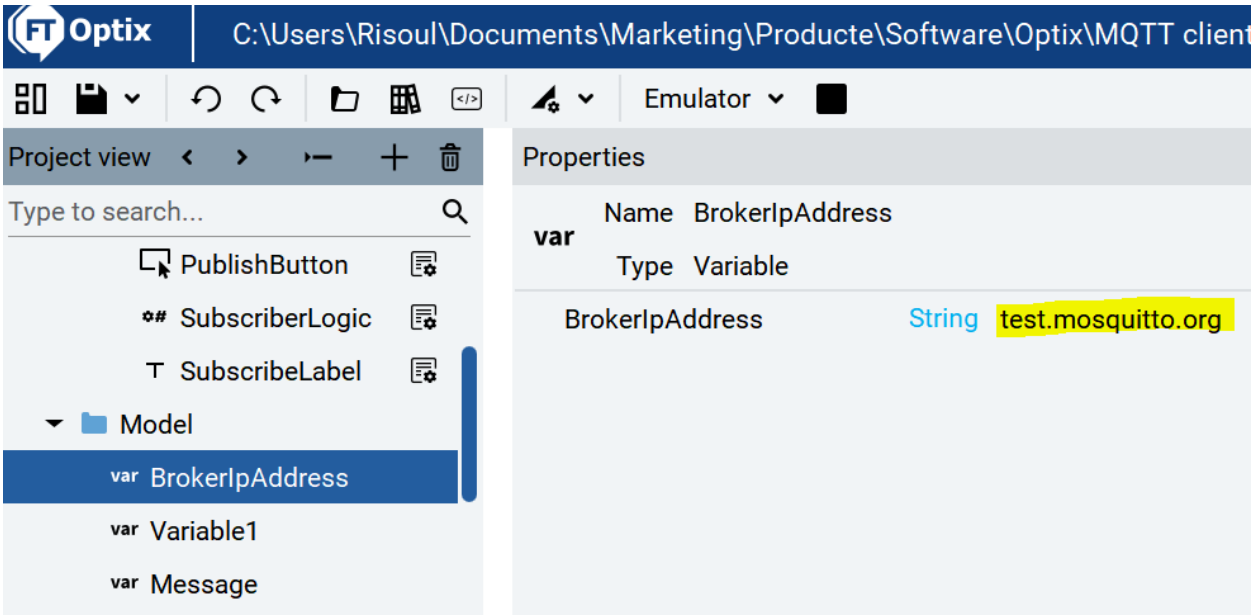
PublisherLogic.cs

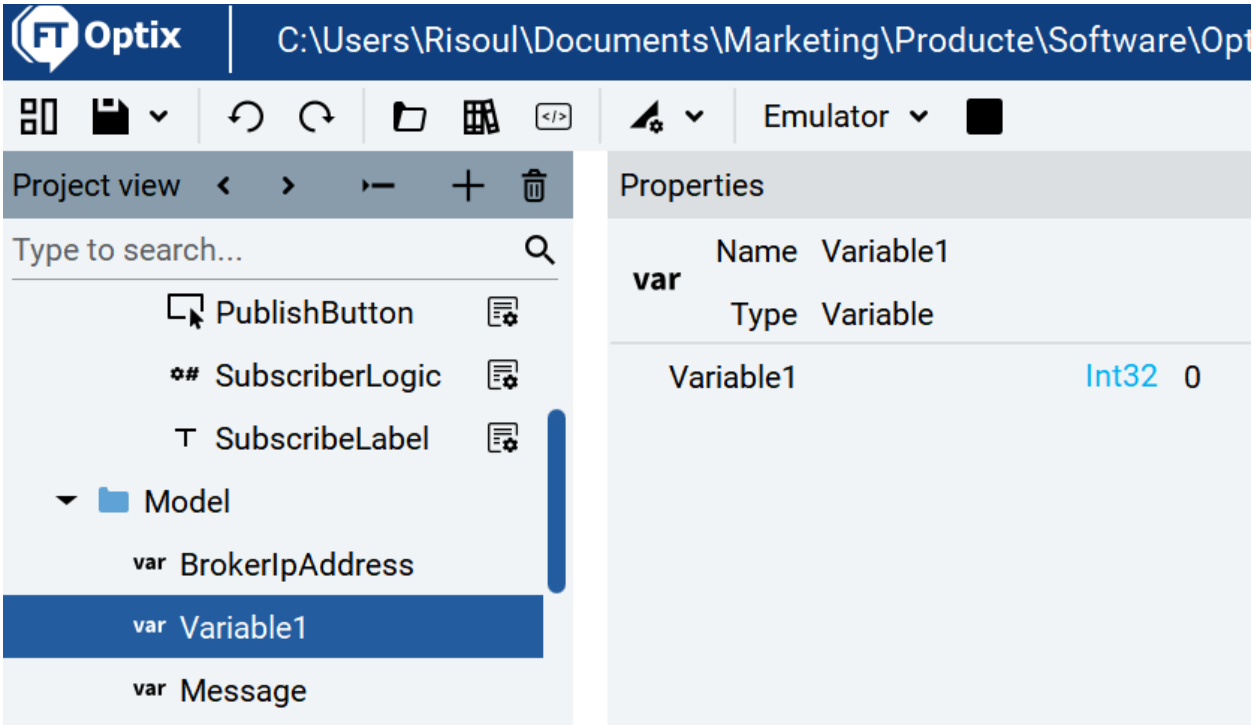
NetSolution > PublisherLogic.cs

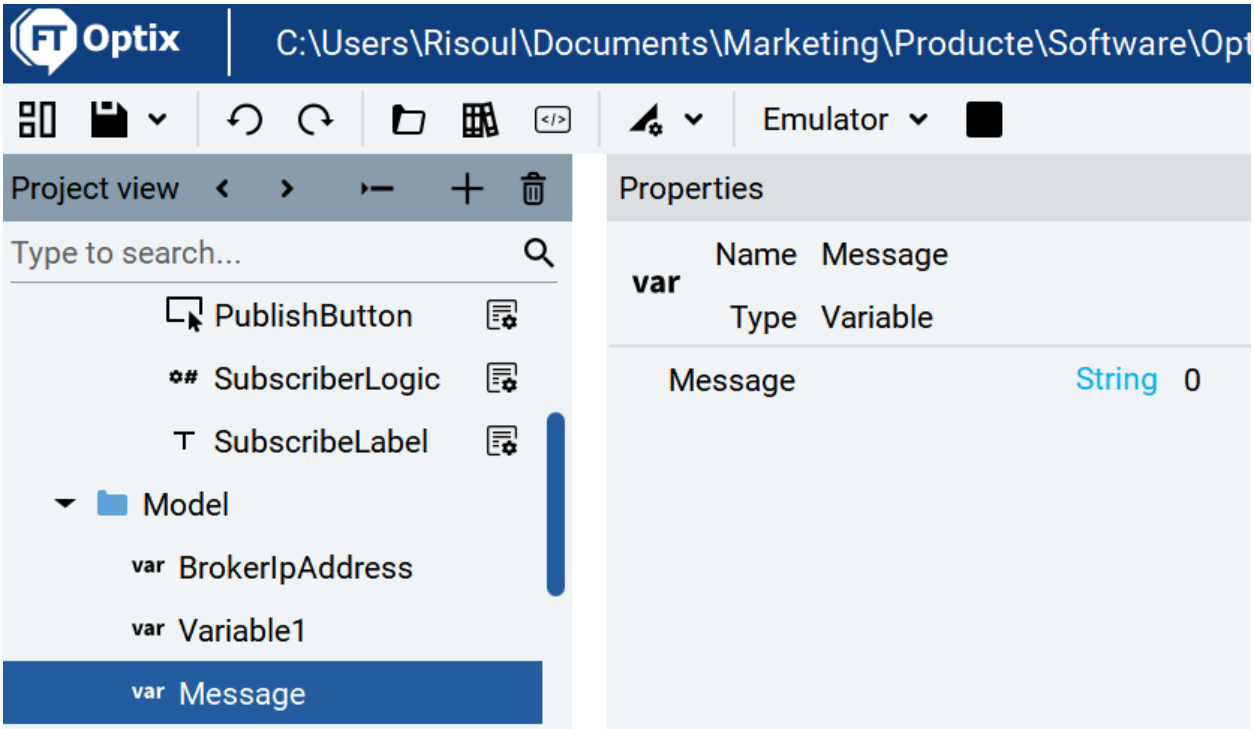
```
#region StandardUsing
using System;
using FTOptix.CoreBase;
using FTOptix.HMIProject;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.NetLogic;
using FTOptix.UI;
using FTOptix.OPCUAServer;
#endregion
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;

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

        // Create a client connecting to the broker (default port is 1883)
        publishClient = new MqttClient(brokerIpAddressVariable.Value);
        // Connect to the broker
        publishClient.Connect("FTOptixPublishClient");
        // Assign a callback to be executed when a message is published to the broker
        publishClient.MqttMsgPublished += PublishClientMqttMsgPublished;
    }

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

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

    [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
    }

    private MqttClient publishClient;
}
```

Look at BrokerIpAddress under UI


Model
var BrokerIpAddress
var Variable1
var Message
Converters


Optix
C:\Users\Risoul\Documents\Marketing\Producte\Software\Optix\MQTT client
Emulator
Project view
Type to search...
PublishButton
SubscriberLogic
SubscribeLabel
Model
var BrokerIpAddress
var Variable1
var Message
Properties
var
Name BrokerIpAddress
Type Variable
BrokerIpAddress String test.mosquitto.org



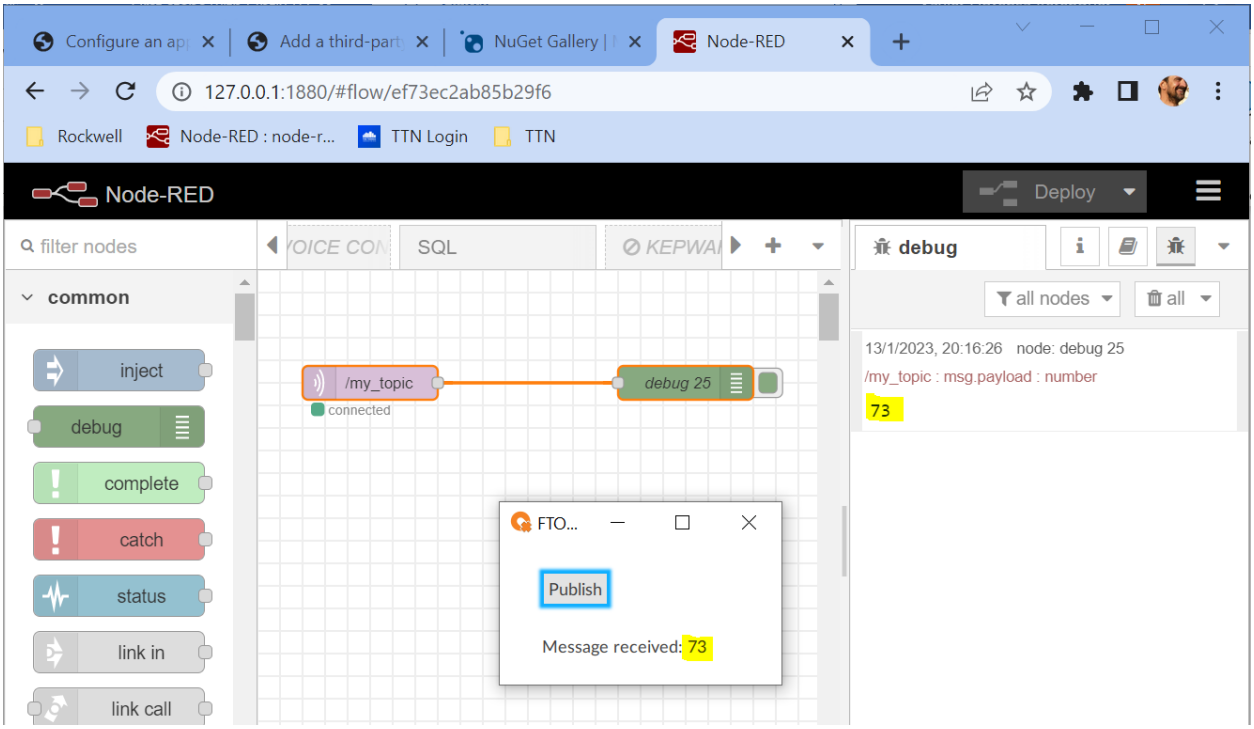


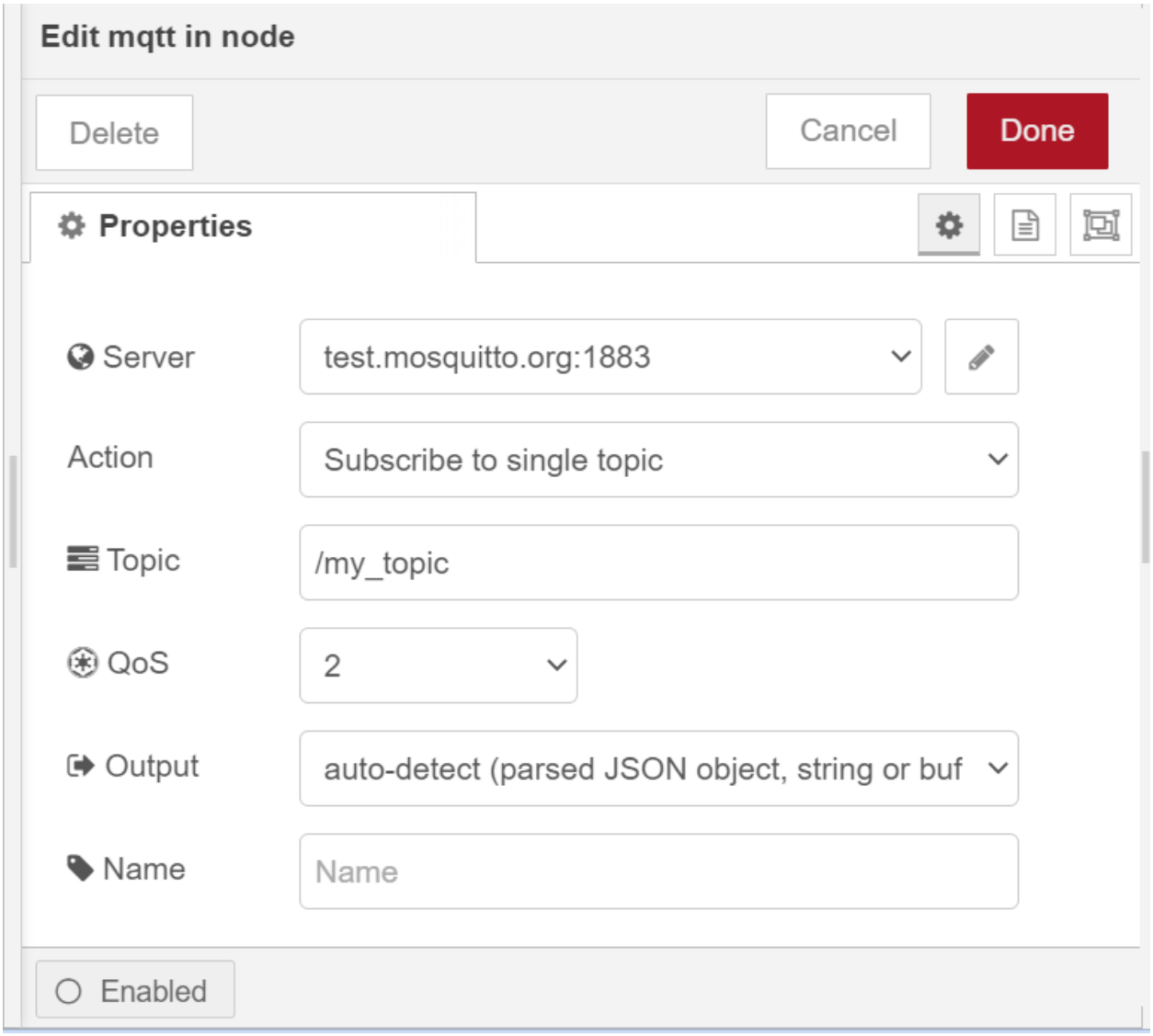
So we are publishing a random value to mosquito on the cloud

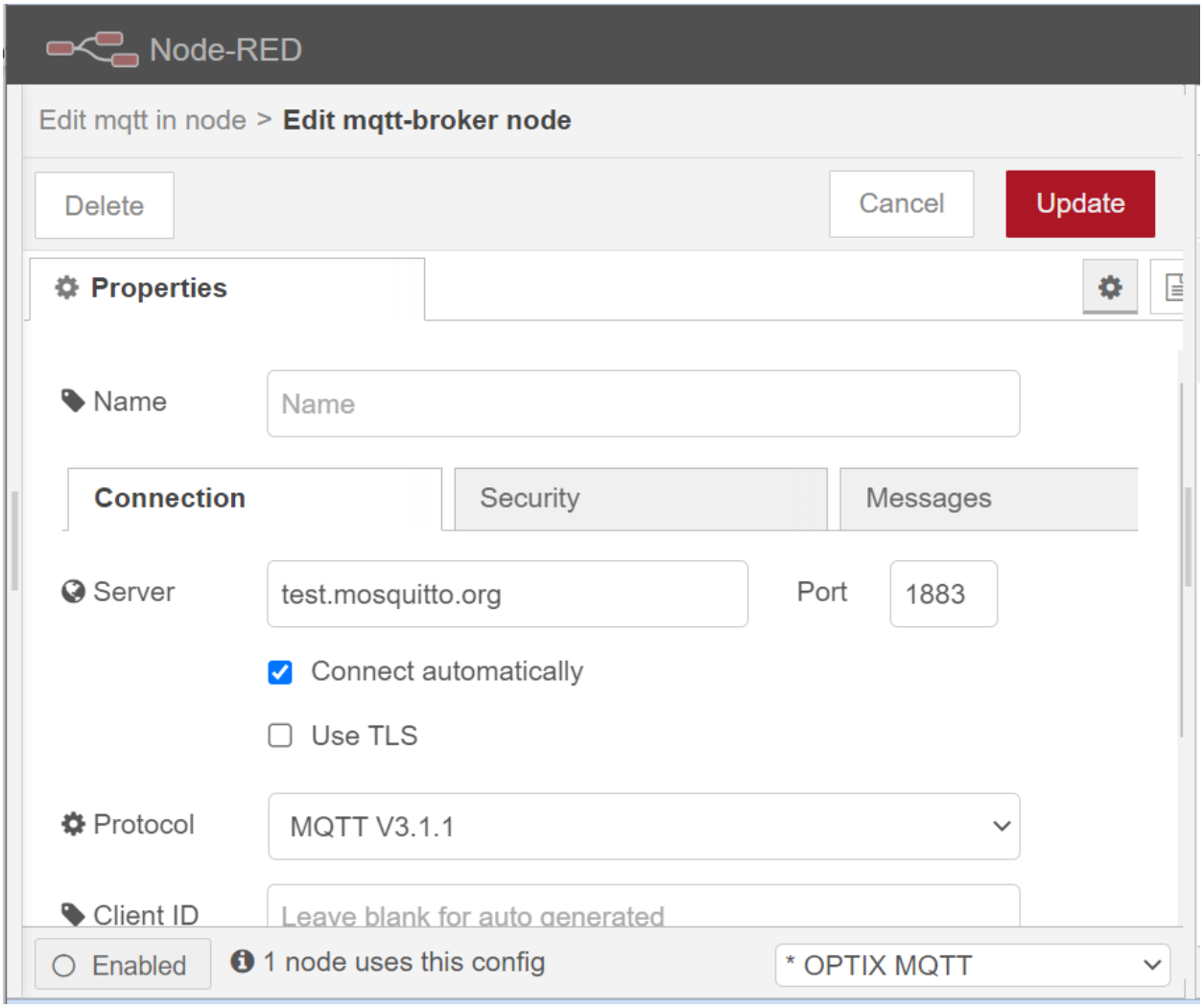
```
[ExportMethod]
public void PublishMessage()
{
    var variable1 = Project.Current.GetVariable("Model/Variable1");
    variable1.Value = new Random().Next(0, 101);

    // 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

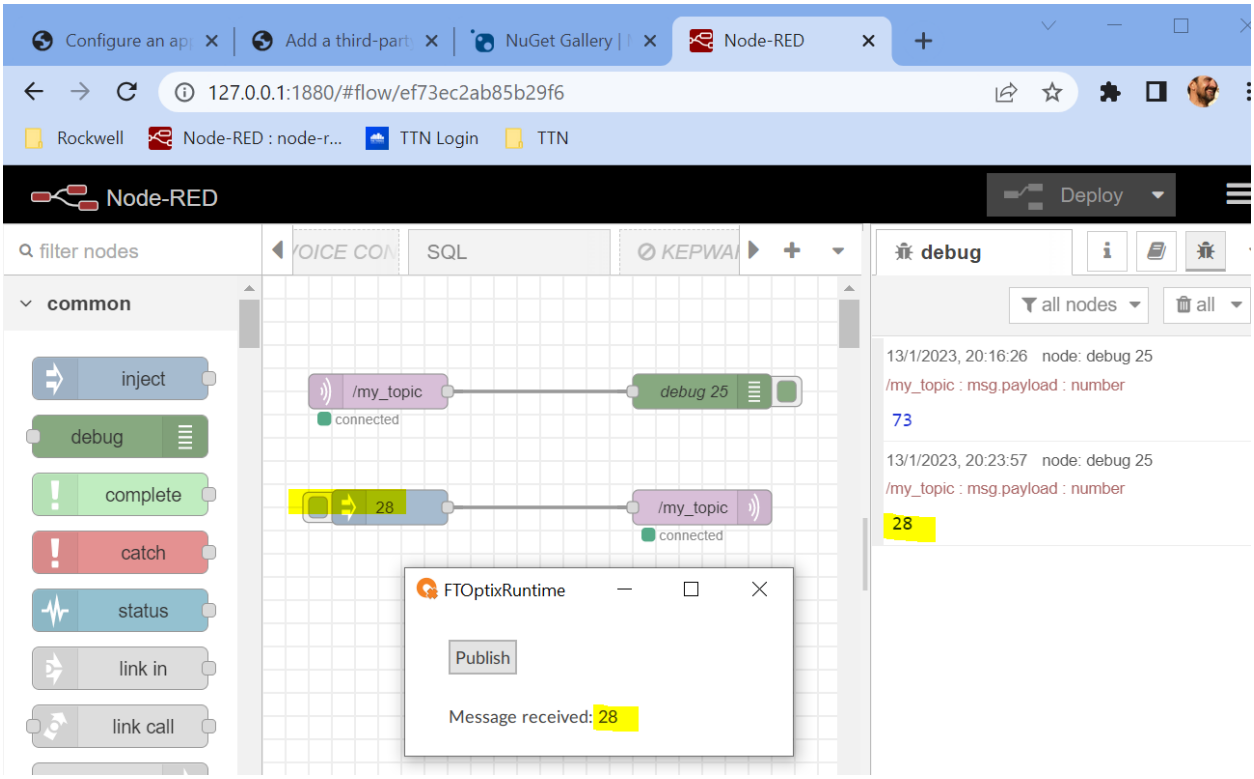


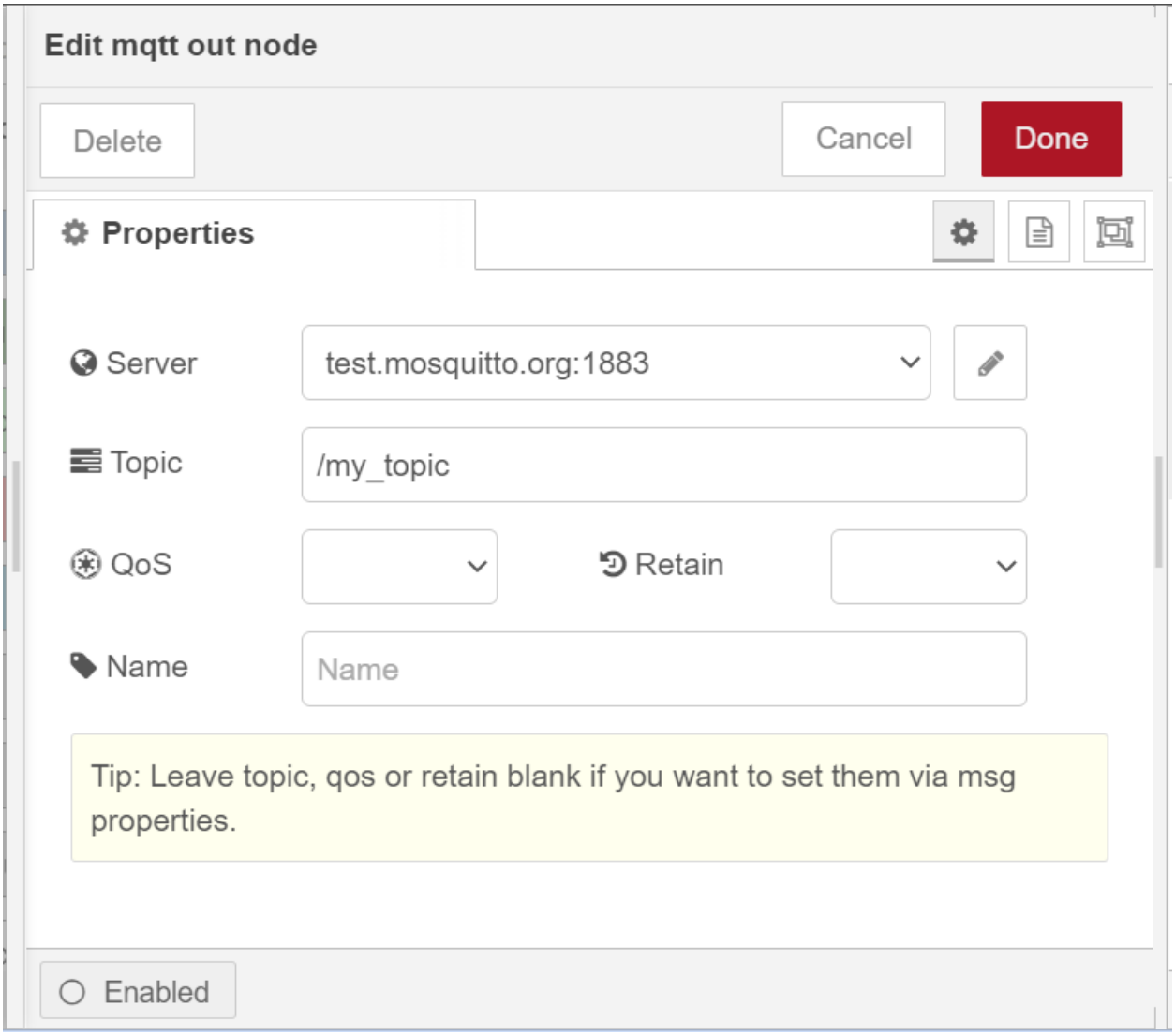

Edit mqtt in node
Delete
Cancel
Done
Properties
Server
test.mosquitto.org:1883
Action
Subscribe to single topic
Topic
/my_topic
QoS
2
Output
auto-detect (parsed JSON object, string or buf
Name
Name
Enabled

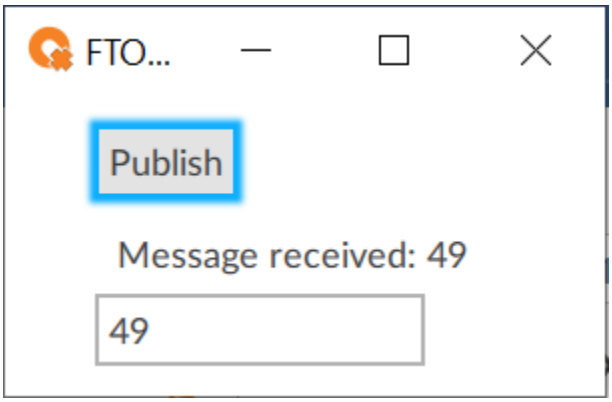


Let's test the subscription from Optix application

Yes, if we inject on Node-RED, the Optix Application gets the value


Node-RED
127.0.0.1:1880/#flow/ef73ec2ab85b29f6
Deploy
filter nodes
SQL
common
inject
debug
complete
catch
status
link in
link call
/my_topic
connected
debug 25
28
FTOptixRuntime
Publish
Message received: 28
debug
all nodes
all
13/1/2023, 20:16:26 node: debug 25
/my_topic : msg.payload : number
73
13/1/2023, 20:23:57 node: debug 25
/my_topic : msg.payload : number
28





Next step is to use PLC data

2. PLC data and MQTT

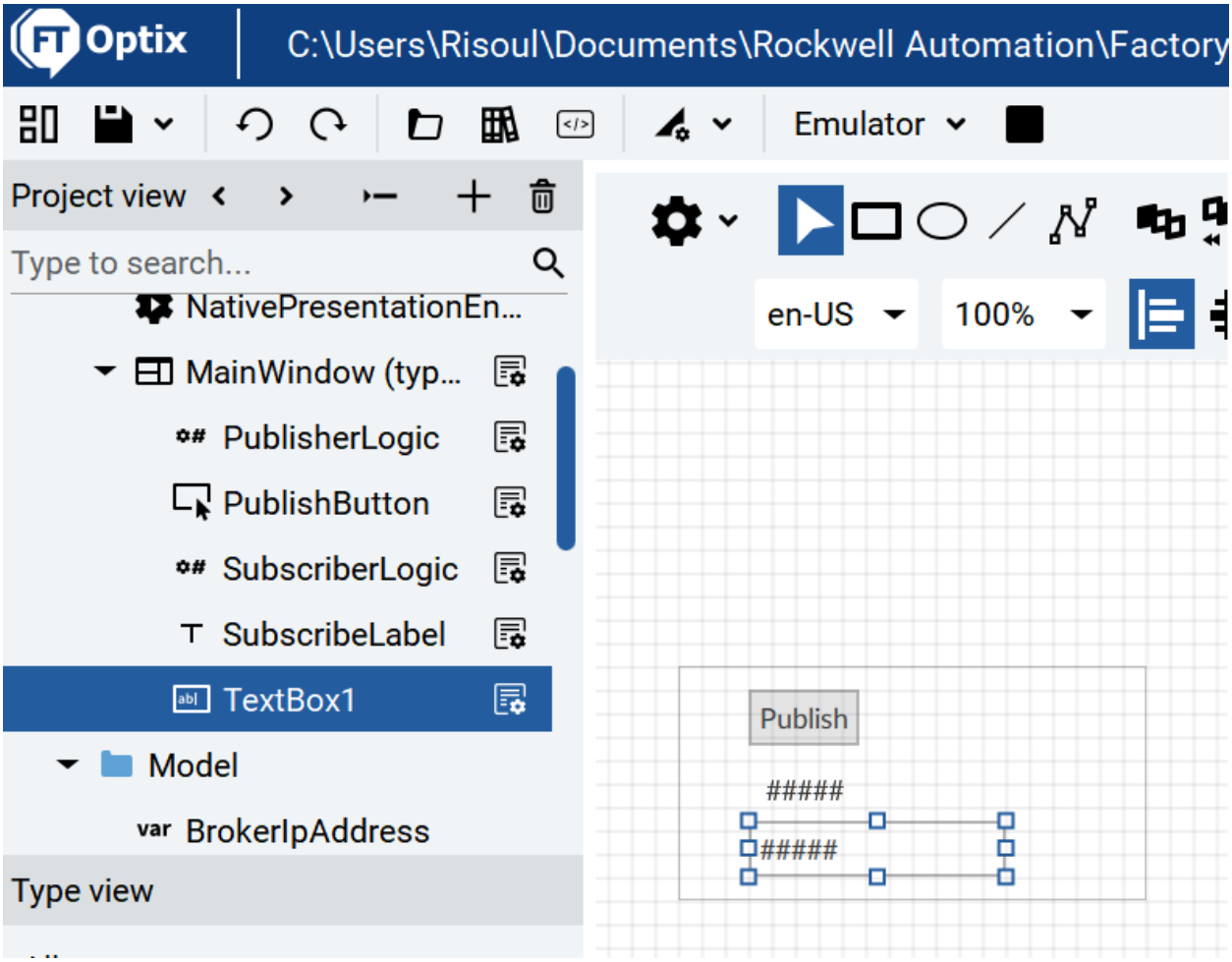
This is what we will do on this example

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

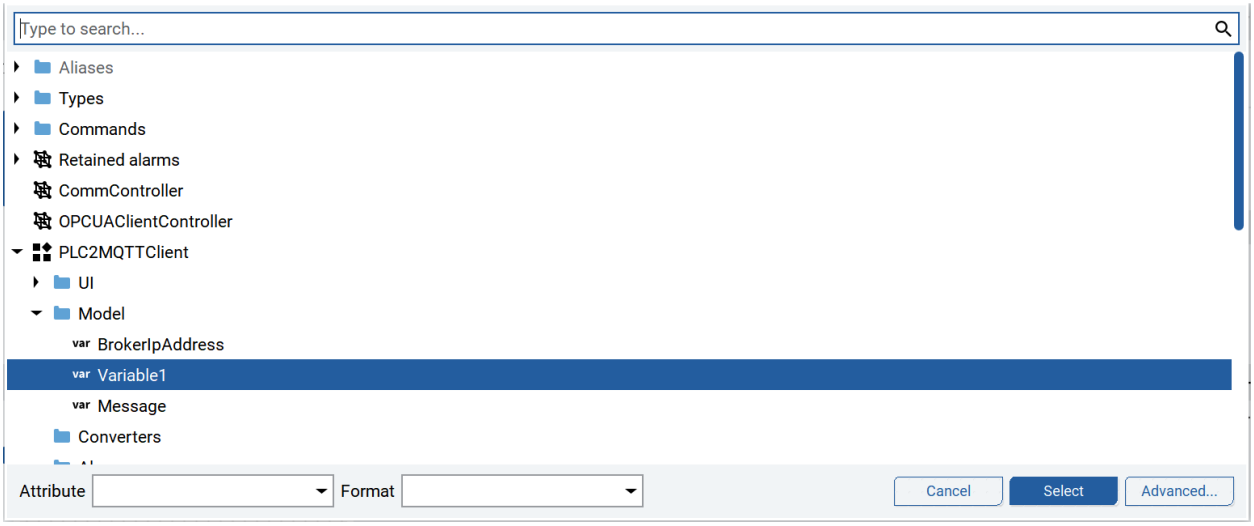
Let's take a look on how to work in our project with the variables provided by MQTT

Reading the subscribed variable

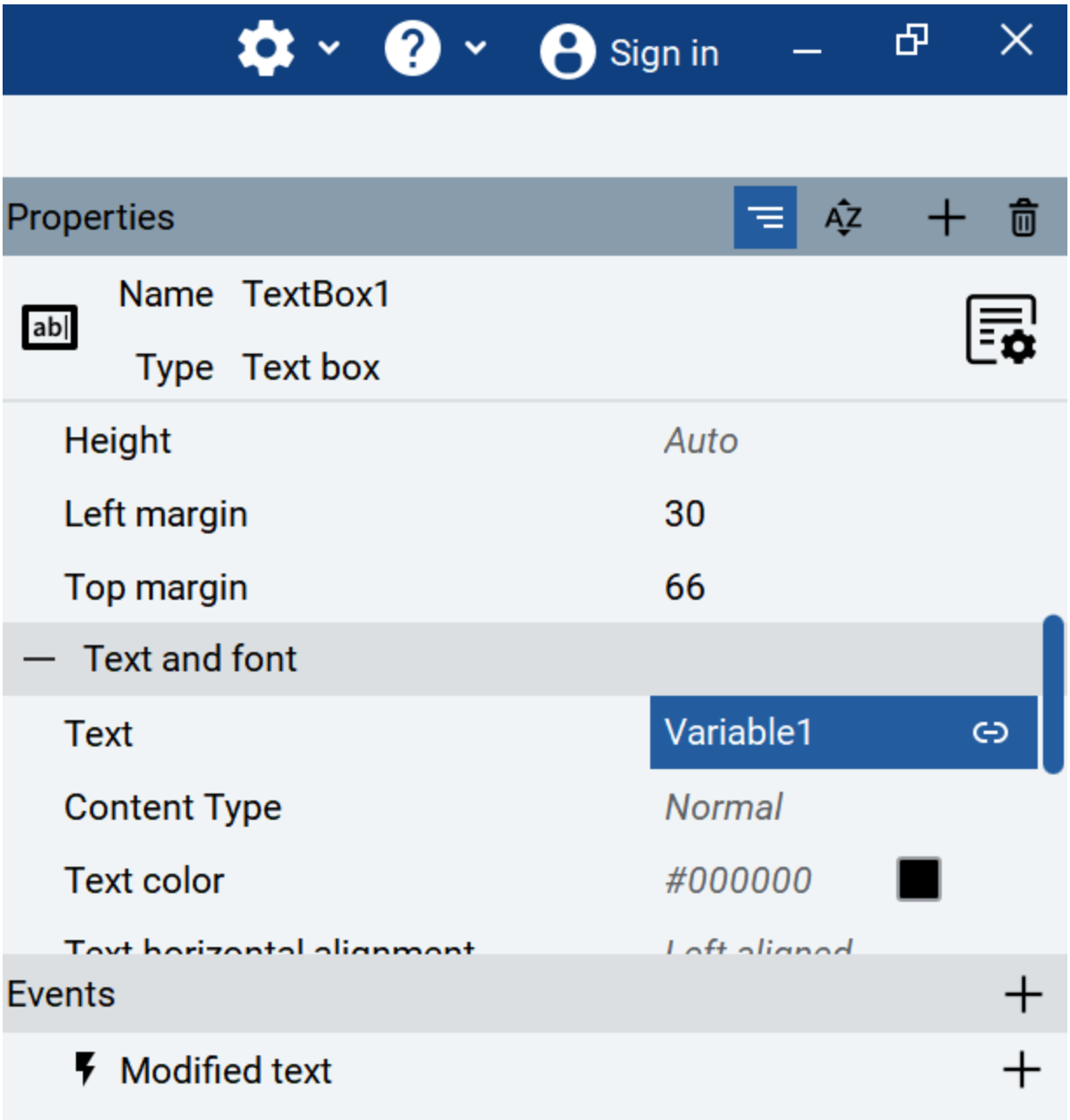
Create a new Texbox



On properties point to Model.variable1



Like this



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

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

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

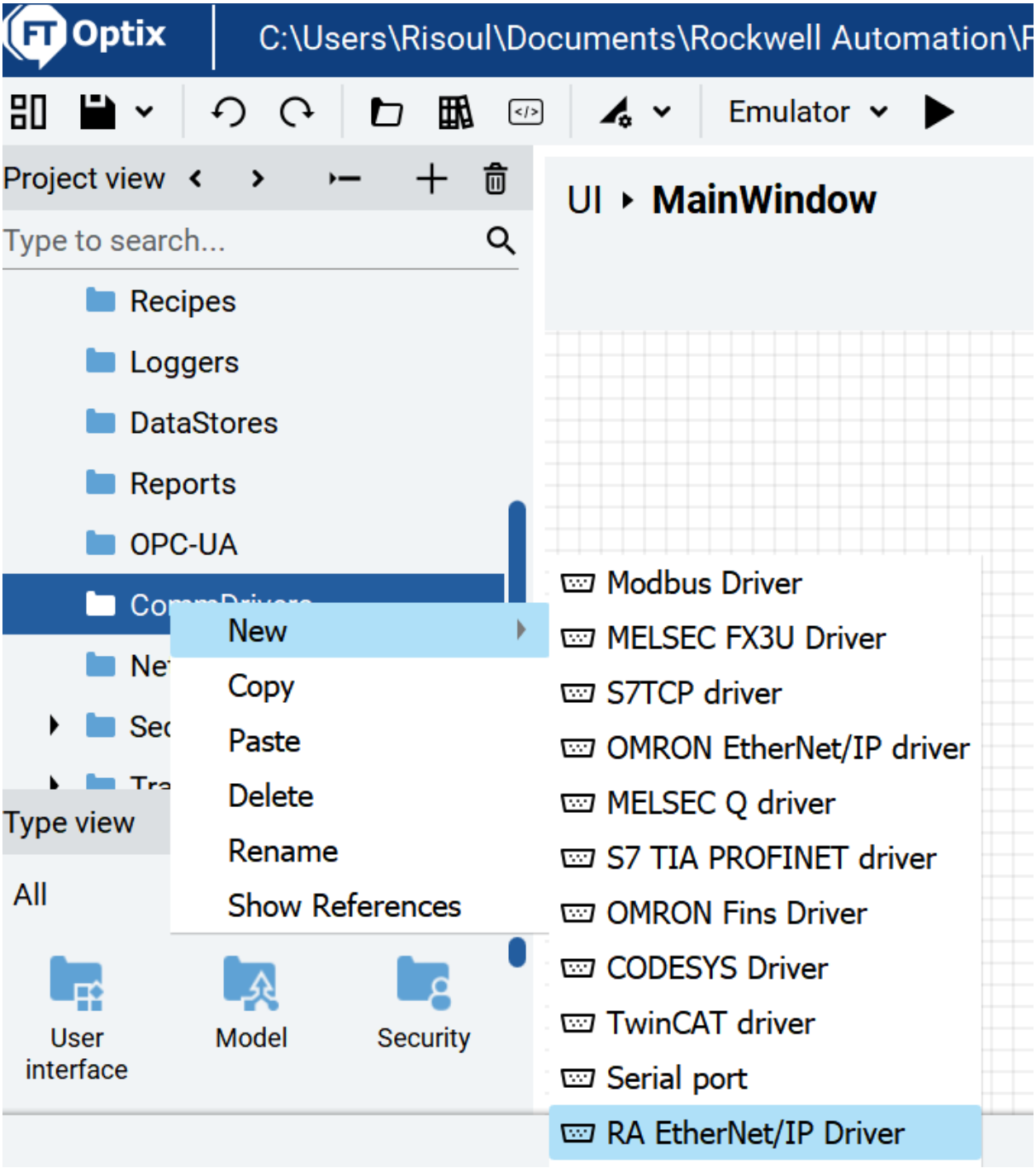
In Logix Designer, download the LogixTags.ACD project to a physical Logix controller or an emulated FactoryTalk® Logix Echo™ controller. Set the controller in the run mode.
For more information, see the Logix Designer online help.

To import controller variables

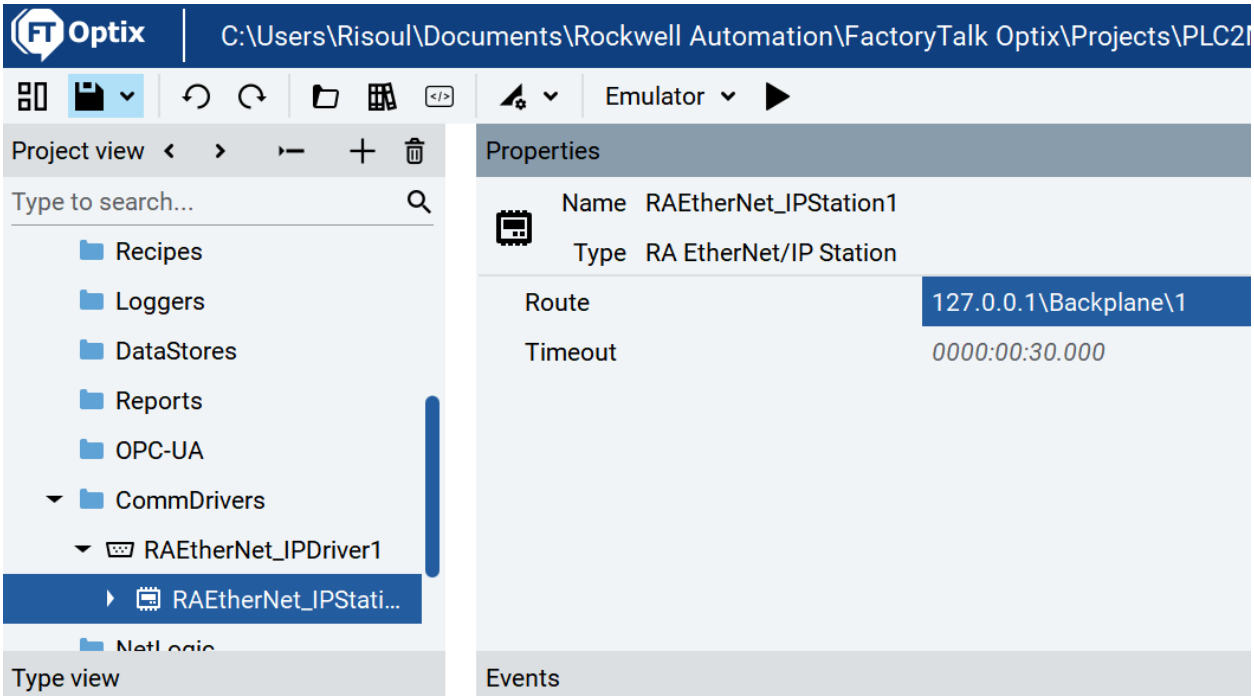
1. From the FactoryTalk Optix Studio toolbar, select **Open dashboard page**.
2. In the central pane, select **I want to configure connected devices**.
3. Select **New stations**.
4. Select **RA EtherNet/IP Station** and select **Next**.
5. (optional) To import the controller tags in the online mode, in **Route**, enter `IP_Address\Backplane\Chassis_Slot_Number` and select **Next**.
6. Select **Next**.
7. Fetch the controller tags:
   - 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.
8. Select all controller tags and select **Next**.
   Figure: Selected tags in the online mode

Configure communication driver

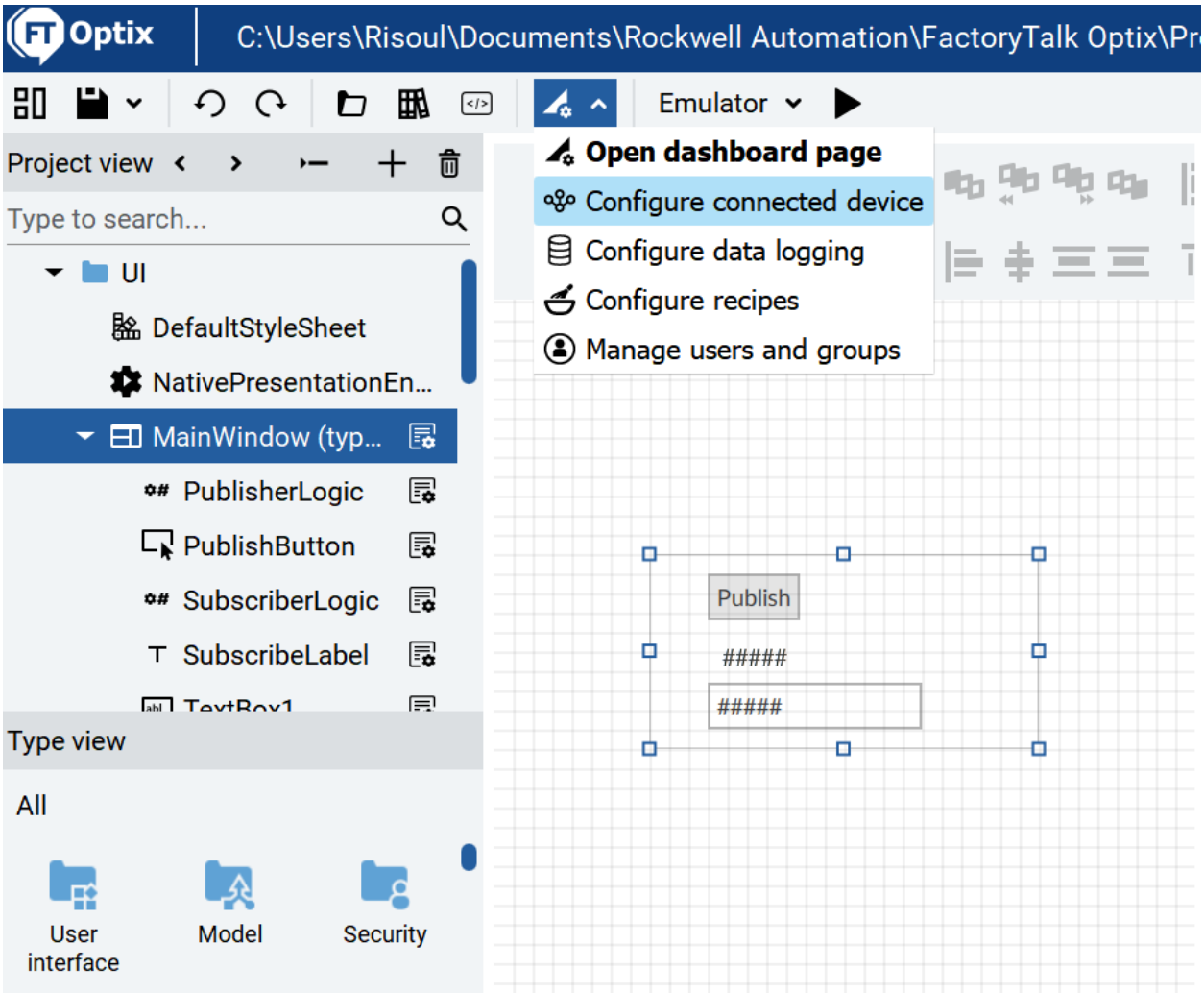
First of all create a communications driver

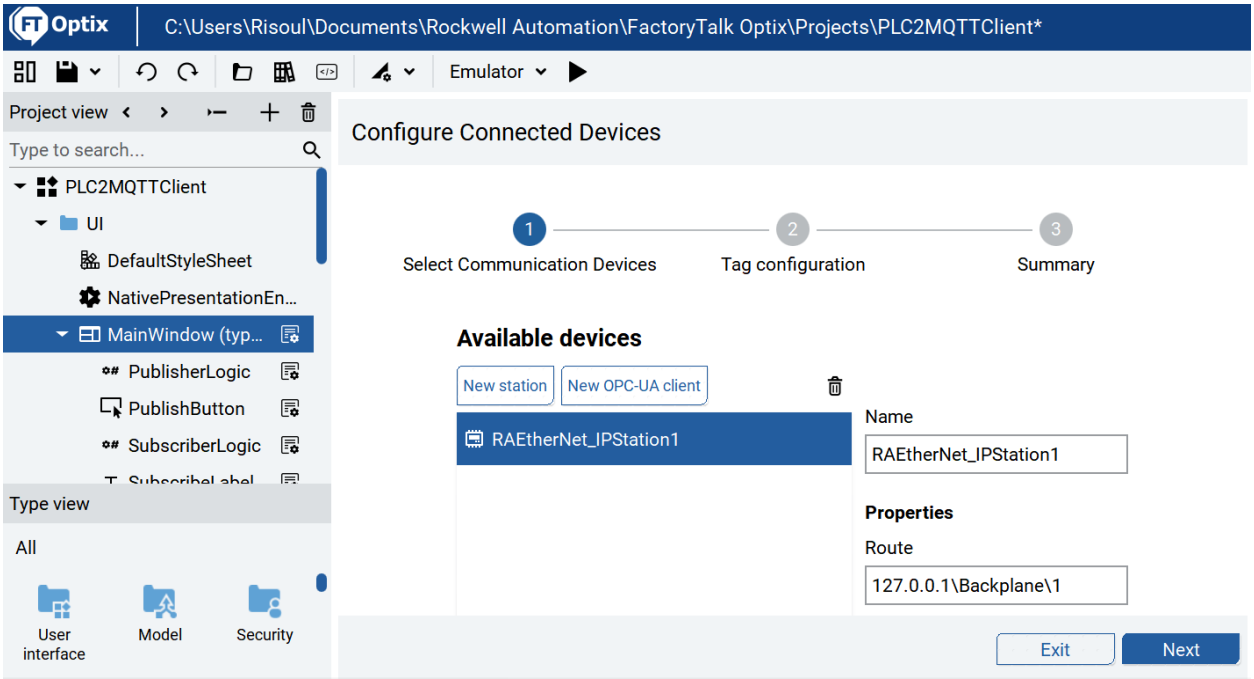


Add a new station

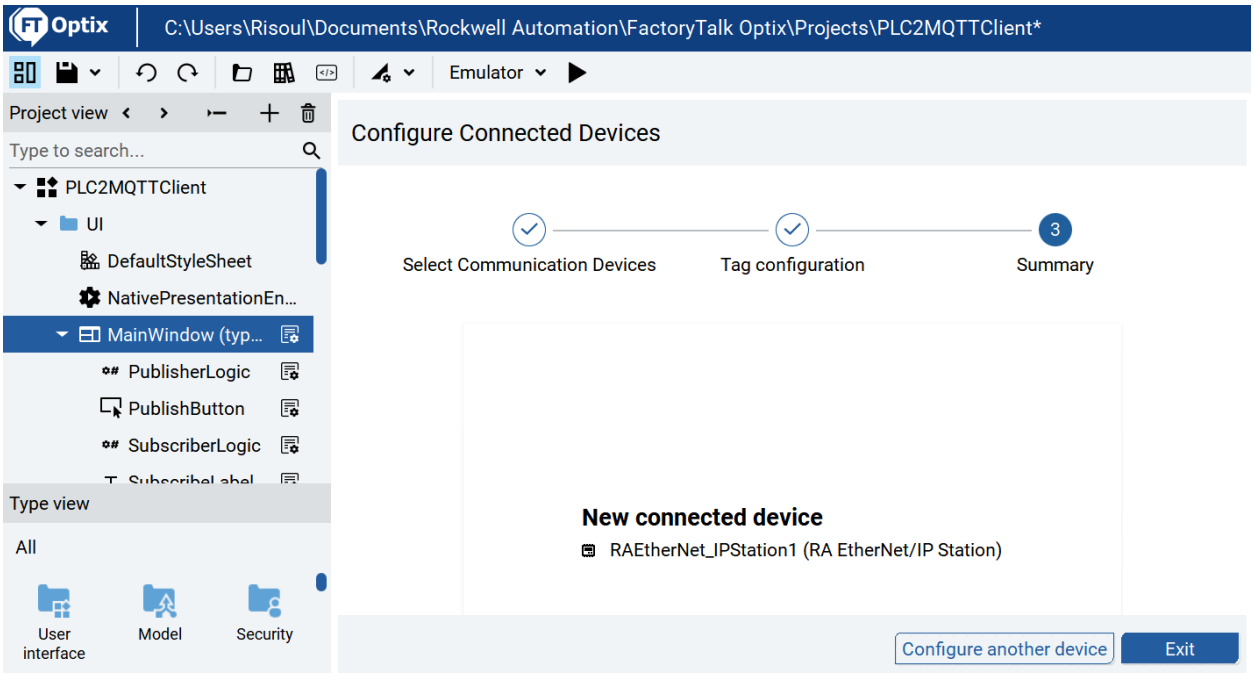


Go to configure connected device

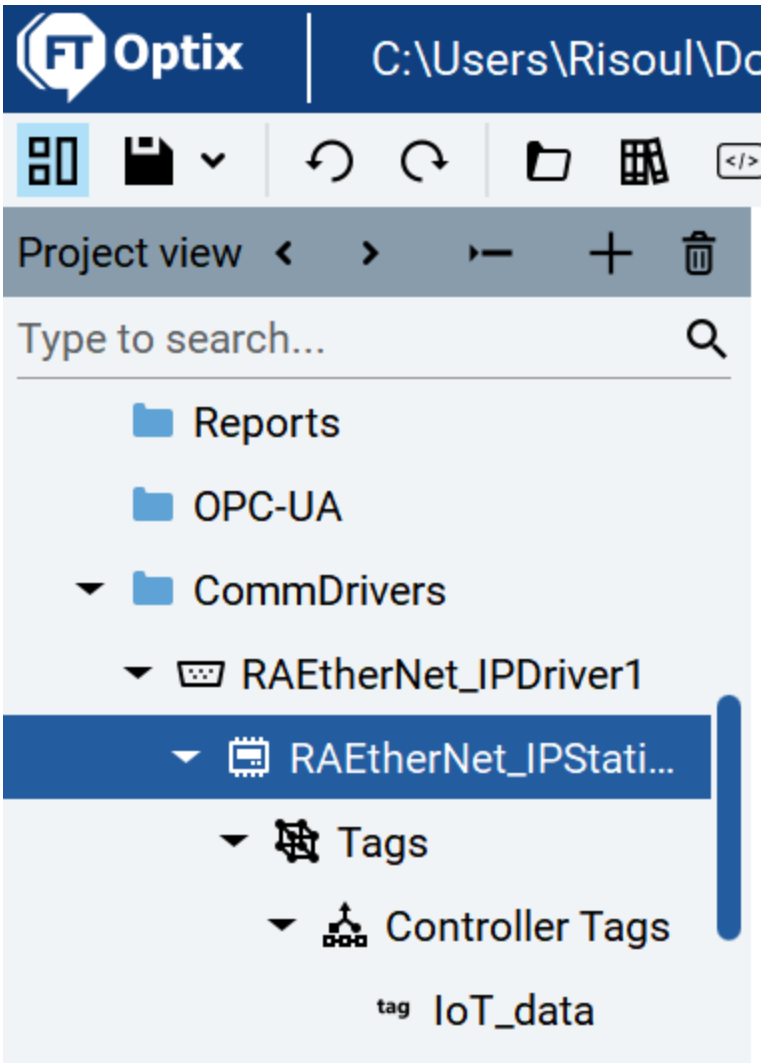

Optix
C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Pr
Emulator
Open dashboard page
Configure connected device
Configure data logging
Configure recipes
Manage users and groups
Project view
Type to search...
UI
DefaultStyleSheet
NativePresentationEn...
MainWindow (typ...
PublisherLogic
PublishButton
SubscriberLogic
SubscribeLabel
Type view
All
User interface
Model
Security
Publish
#####
#####



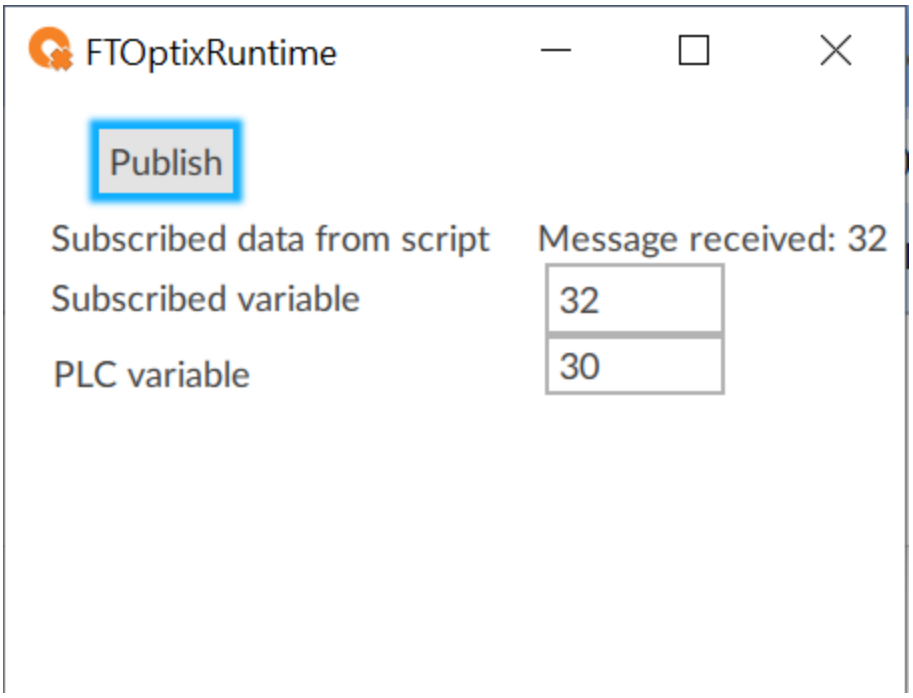
Continue as we did on chapter 2 until you reach this point



Verify that the Tag is there



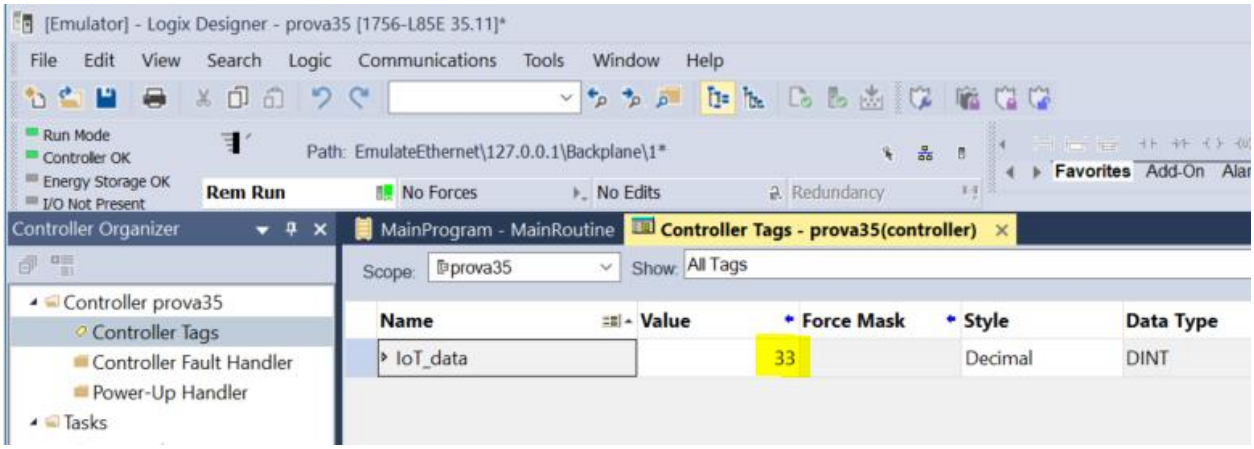
Now let’s display the PLC data on a new text label



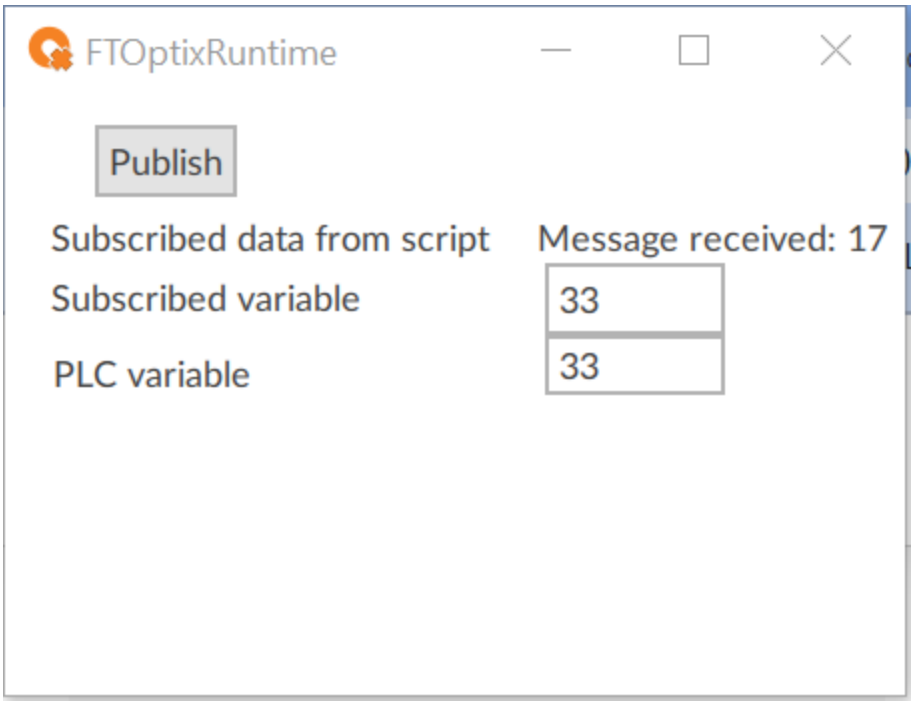
Now let's try to publish the data from PLC variable

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

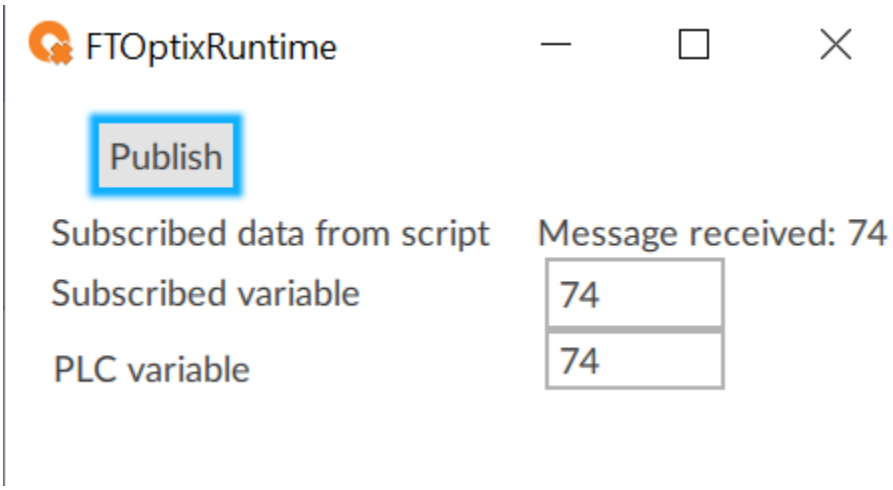
Modify the value on the PLC

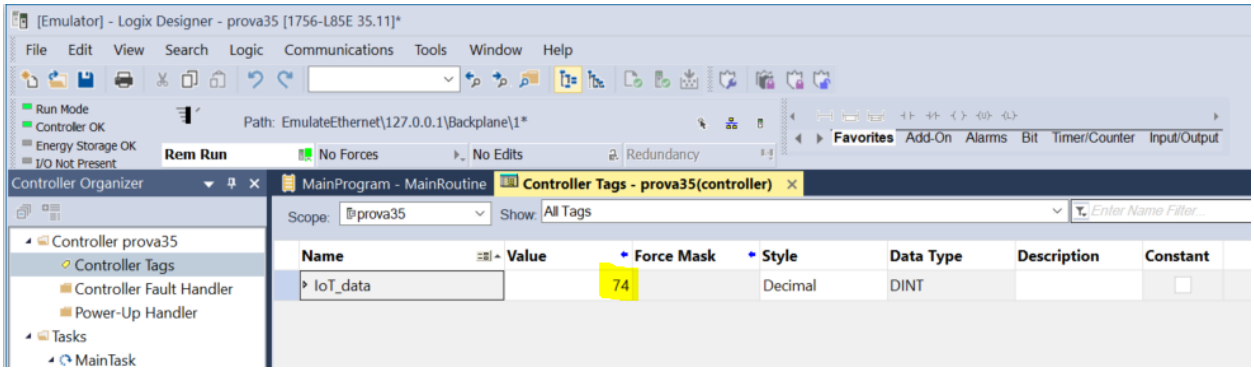


Then look at the Optix application



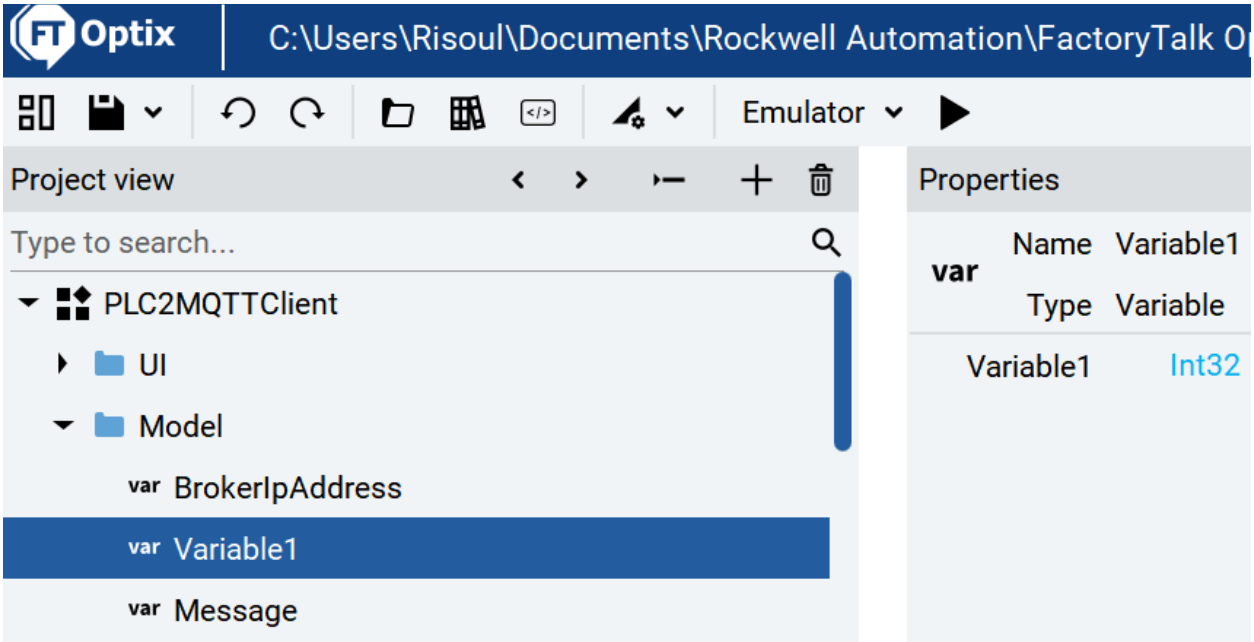
We wanted just to write on the PLC, but on the other hand if we click the button publish, then we are writing a random value on the PLC!!!





This is how to make this link

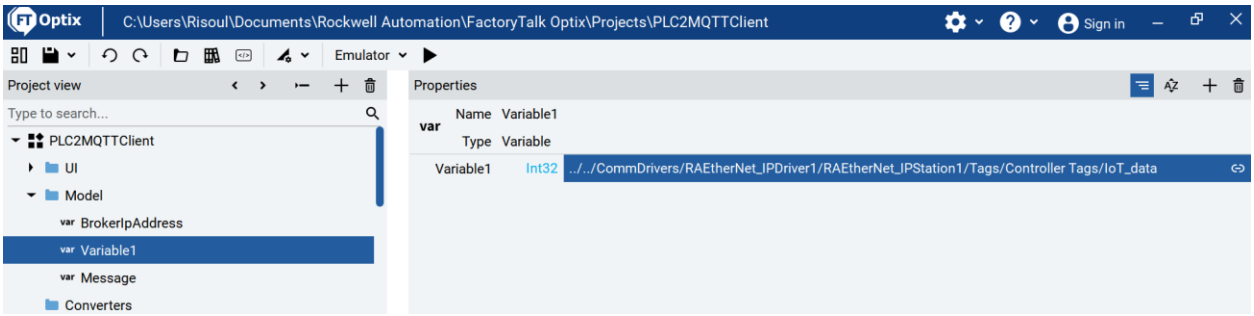
Just go to the Variable1 properties under model



Then edit the link

Like this

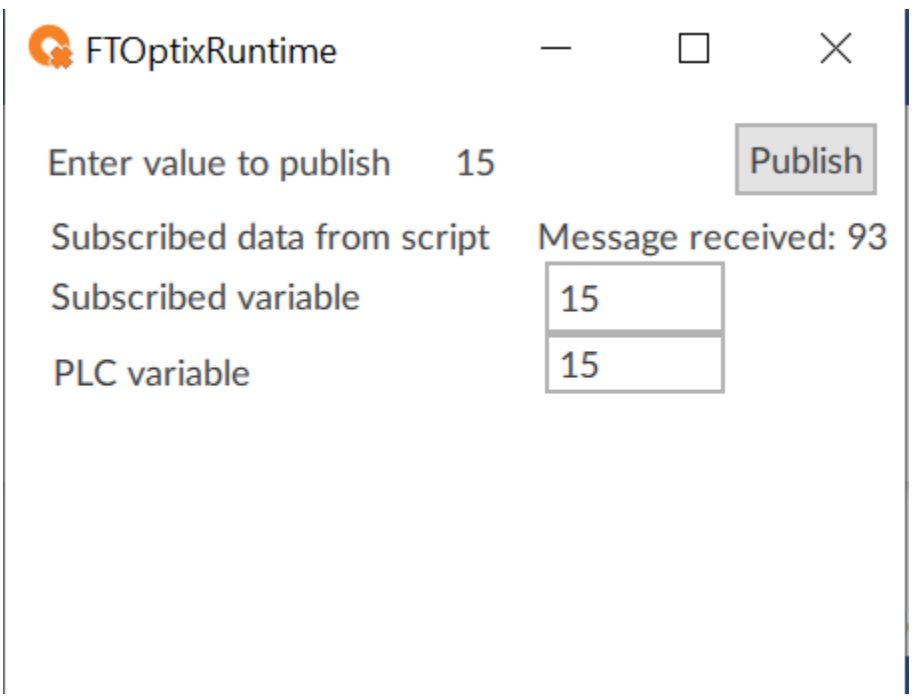
You will get this

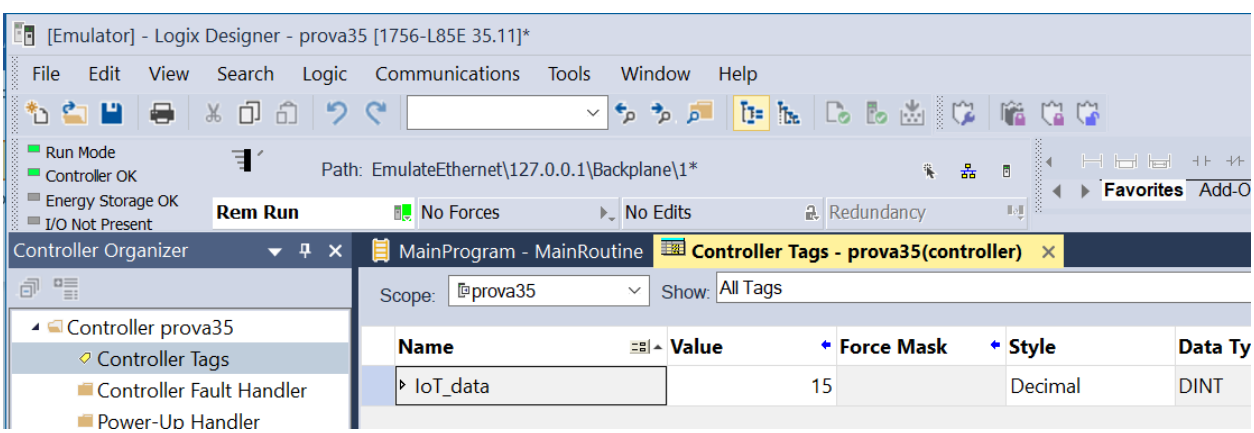


And that's all, you have the variables linked in both directions

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.

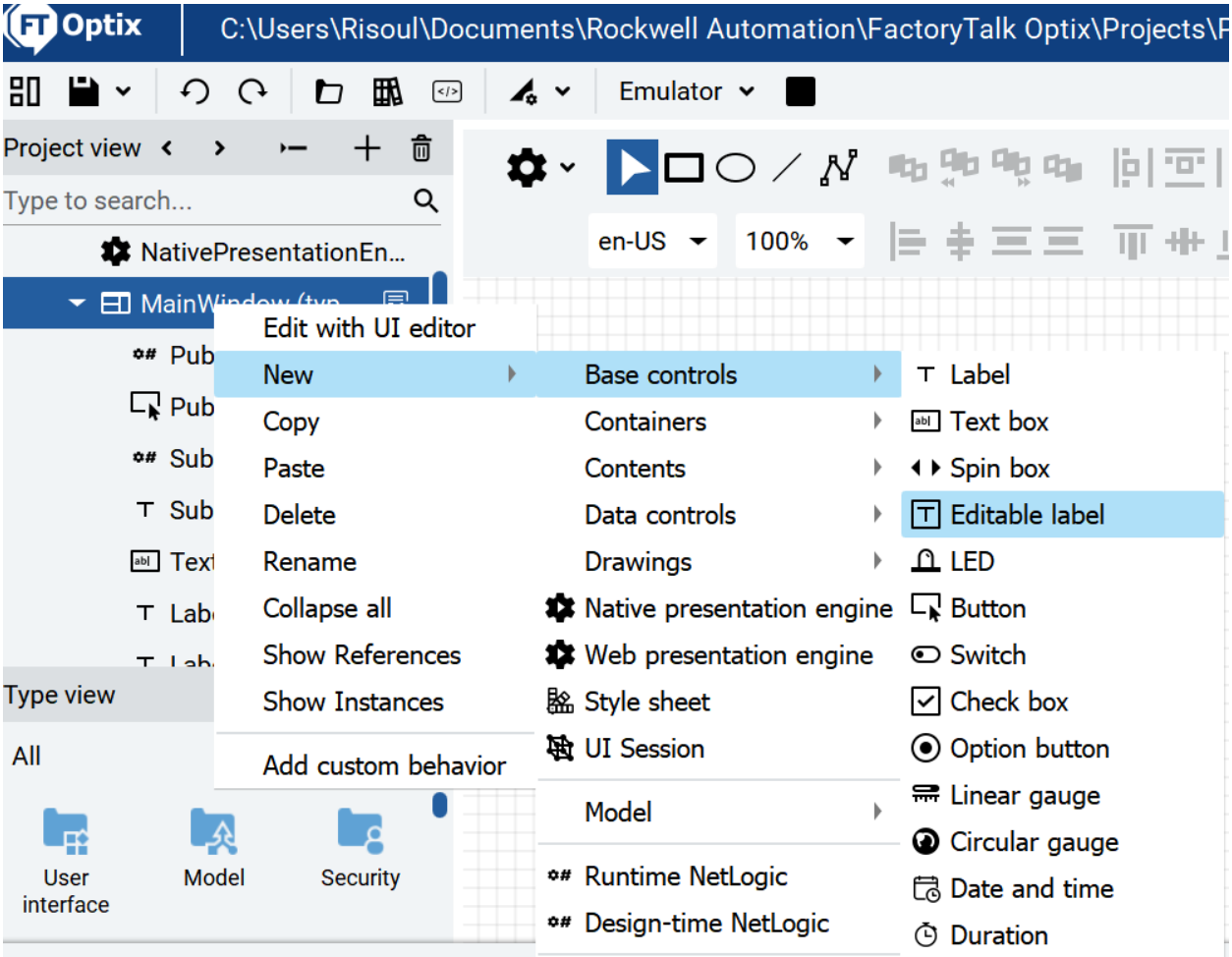
Now we are able to write the desired value on the PLC as soon as we hit enter



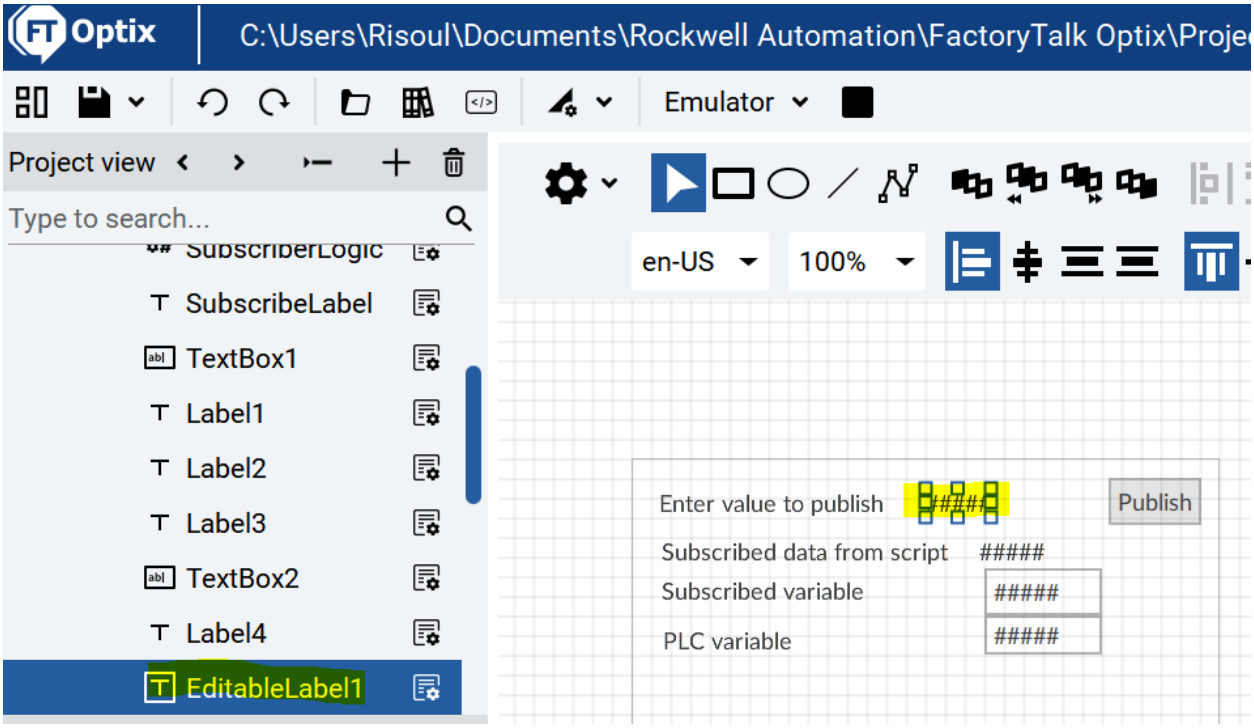


Let’s see how to do it

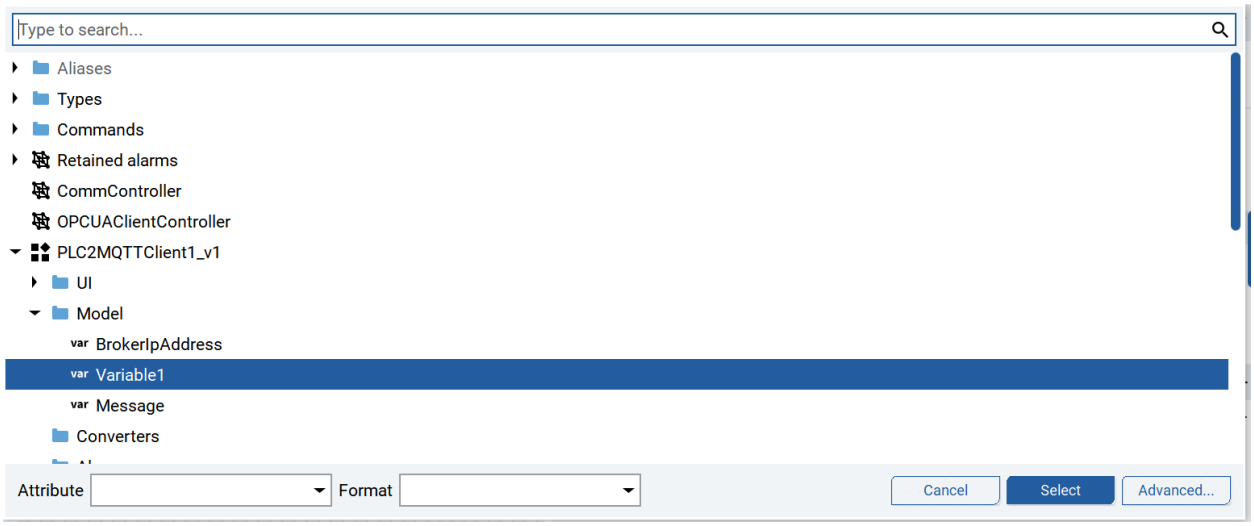
Let’s create an editable Label



Like this



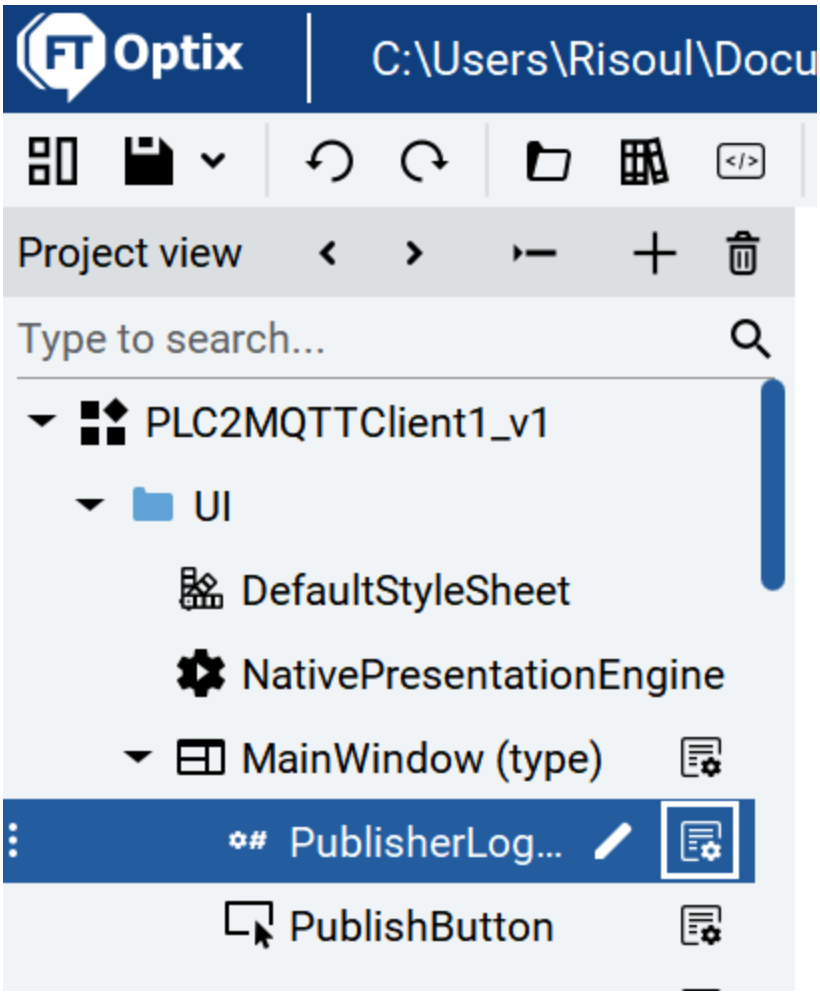
And on the text property, link to the variable



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

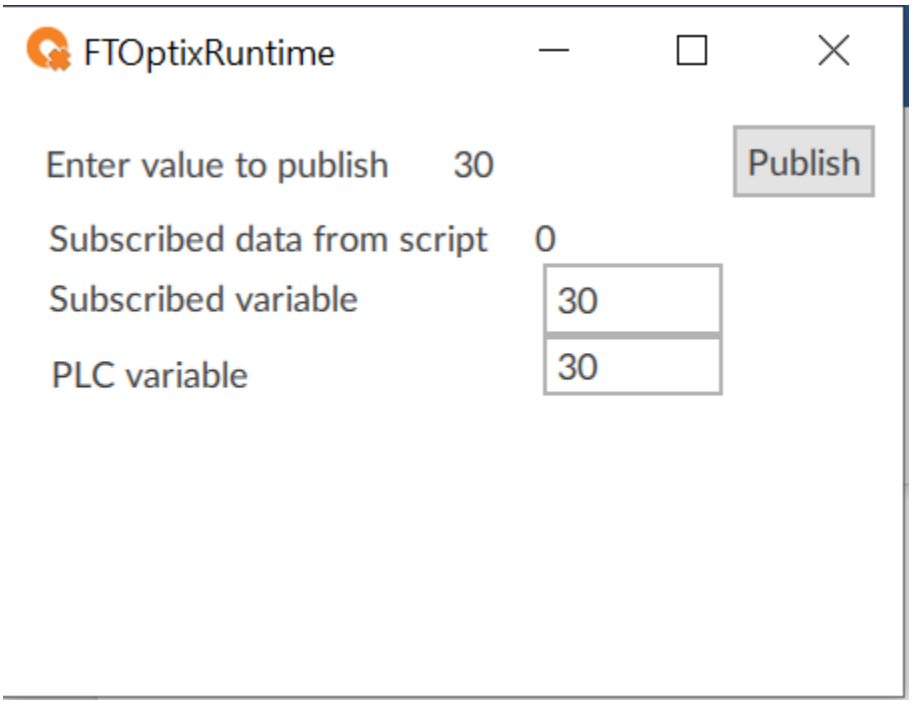
```
36      private void PublishClientMqttMsgPublished(object sender, MqttMsgPublishedEventArgs e)
37      {
38          Log.Info("Message " + e.MessageId + " - published = " + e.IsPublished);
39      }
40
41      [ExportMethod]
42      public void PublishMessage()
43      {
44          var variable1 = 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

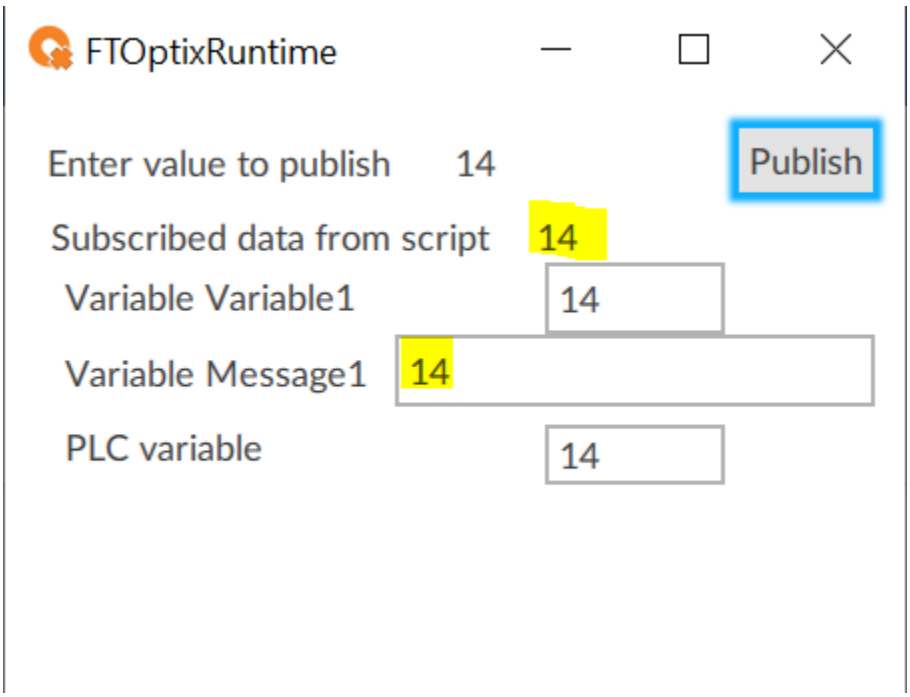
```
43        messageVariable.Value = "Message received: " + System.Text.Encoding.UTF8.GetString(e.Message);
```

And leaving like this

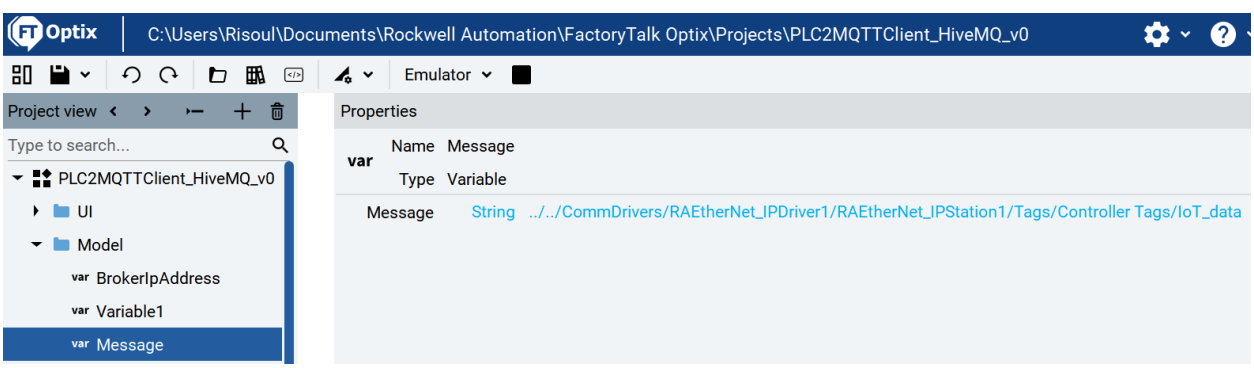
```
44        messageVariable.Value = System.Text.Encoding.UTF8.GetString(e.Message);
```

Then save on Visual Studio Code

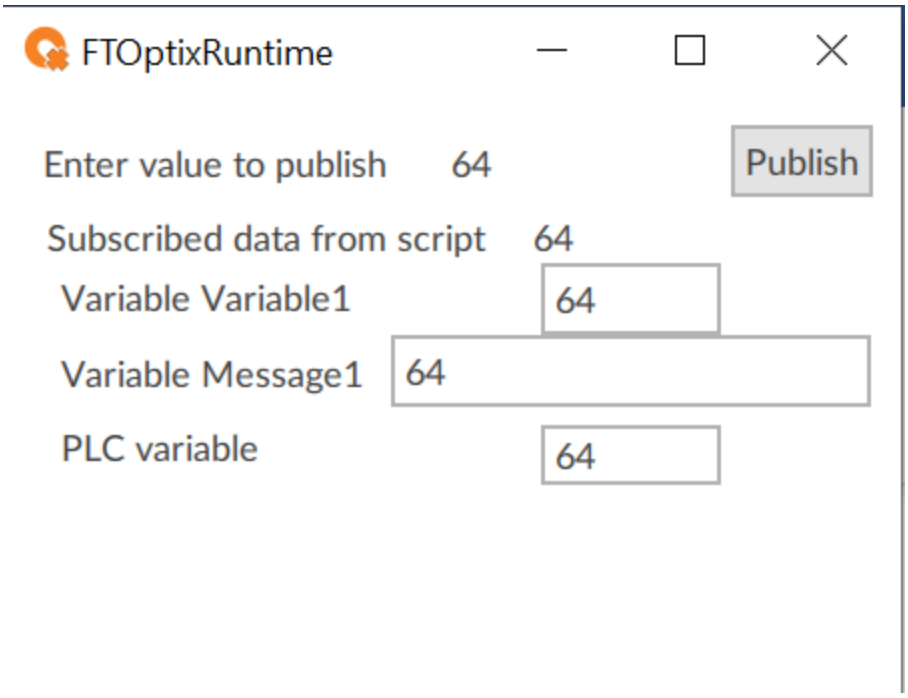
Now we do not have the string like before

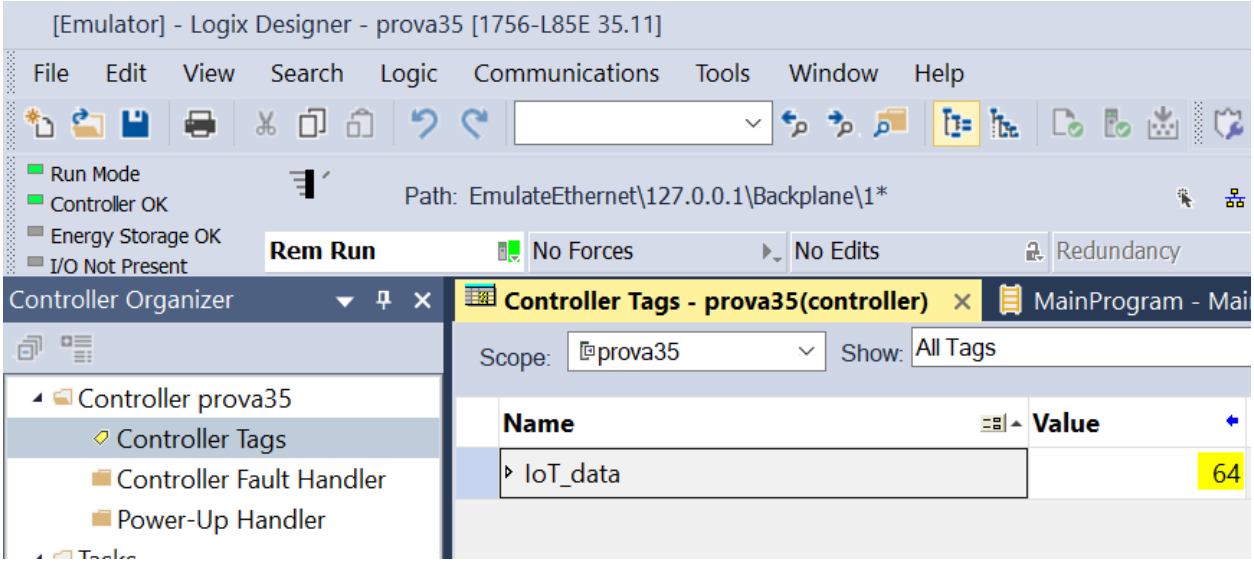


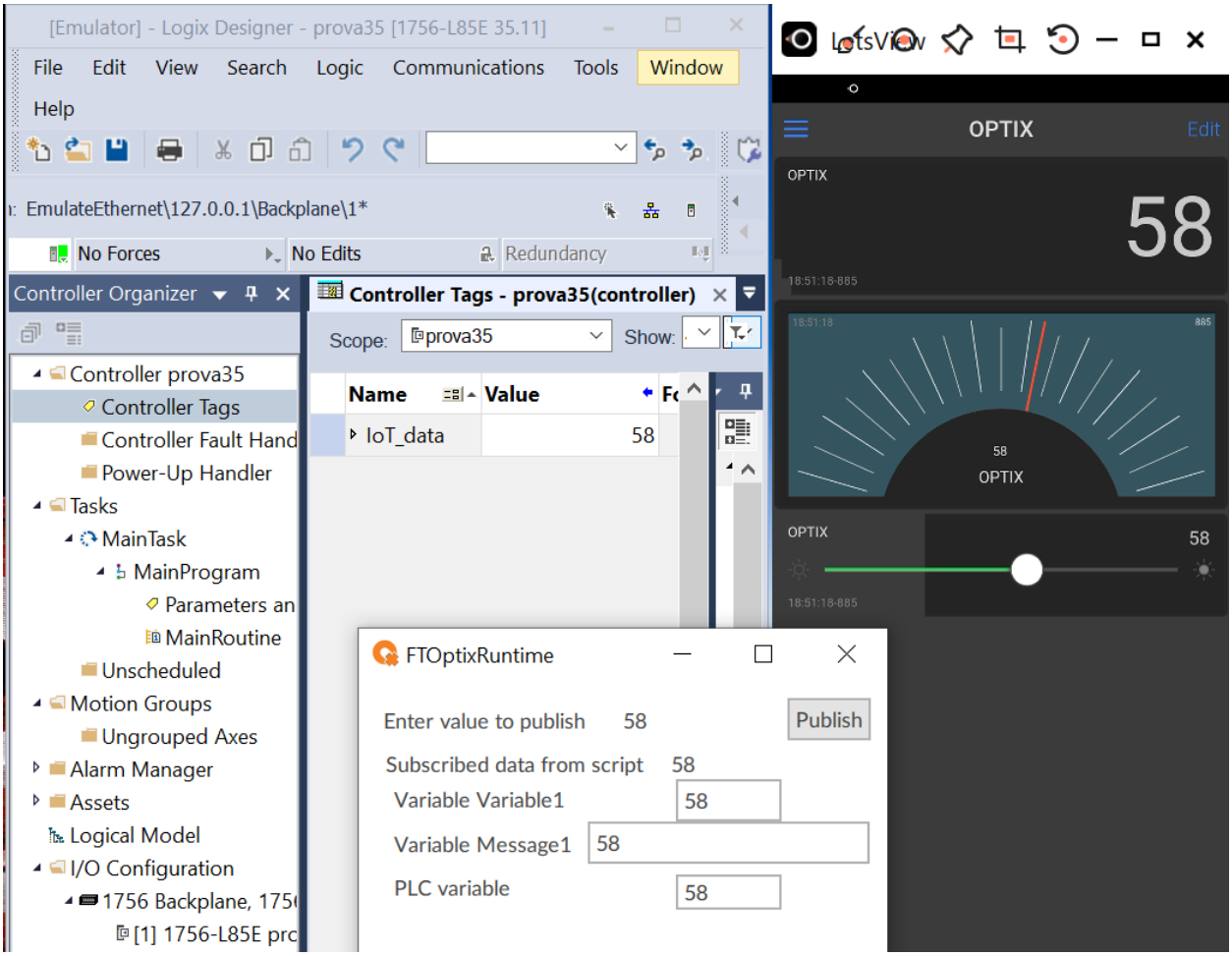
On the other hand, we have to link the variable Message to the PLC Tag like this



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


FTOptixRuntime
Enter value to publish 64
Publish
Subscribed data from script 64
Variable Variable1
64
Variable Message1
64
PLC variable
64


[Emulator] - Logix Designer - prova35 [1756-L85E 35.11]
File Edit View Search Logic Communications Tools Window Help
Run Mode
Controller OK
Energy Storage OK
I/O Not Present
Path: EmulateEthernet\127.0.0.1\Backplane\1*
Rem Run
No Forces
No Edits
Redundancy
Controller Organizer
Controller Tags - prova35(controller)
MainProgram - Mai
Scope: prova35
Show: All Tags
Controller prova35
Controller Tags
Controller Fault Handler
Power-Up Handler
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