

Weintek iP series

In this tutorial, we are going to demonstrate how to connect the Weintek iP series HMI device with UniPi controller running Mervis.

The goal of this tutorial is to display temperature measured by the 1-Wire temperature sensor connected to UniPi on the HMI display and have a touch button on the display that will turn the relay output of the UniPi ON and OFF.

Necessary equipment and setup

UniPi Neuron controller with Mervis installed (see [this guide](#))

Weintek HMI device (<https://www.unipi.technology/hmi-devices-c29>).

24V power supply (<https://www.unipi.technology/power-supplies-c15>).

3 ethernet connections - one for UniPi controller, one for HMI and one for your PC

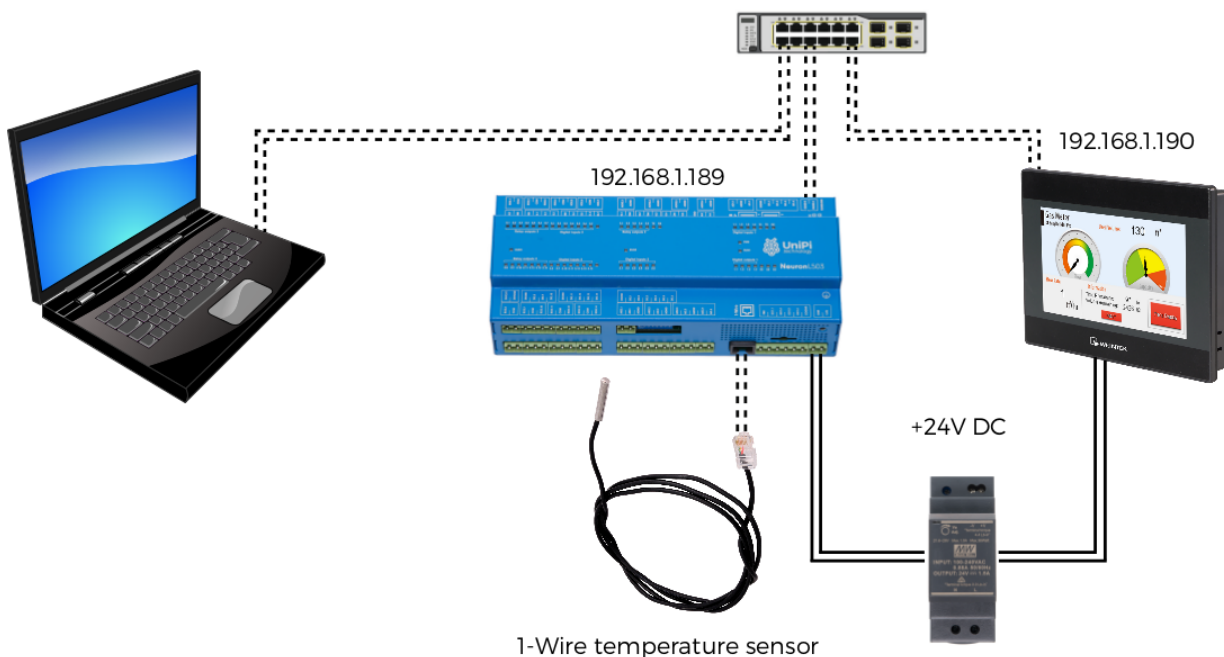
In this tutorial, we will use:

Unipi Neuron L503 (<https://www.unipi.technology/unipi-neuron-l503-p105>).

Weintek MT8051iP (<https://www.unipi.technology/weintek-mt8051ip-p198>).

1-Wire temperature sensor (<https://www.unipi.technology/1-wire-temperature-sensor-p63?categoryId=3>).

Wiring of the test setup



Installation of software

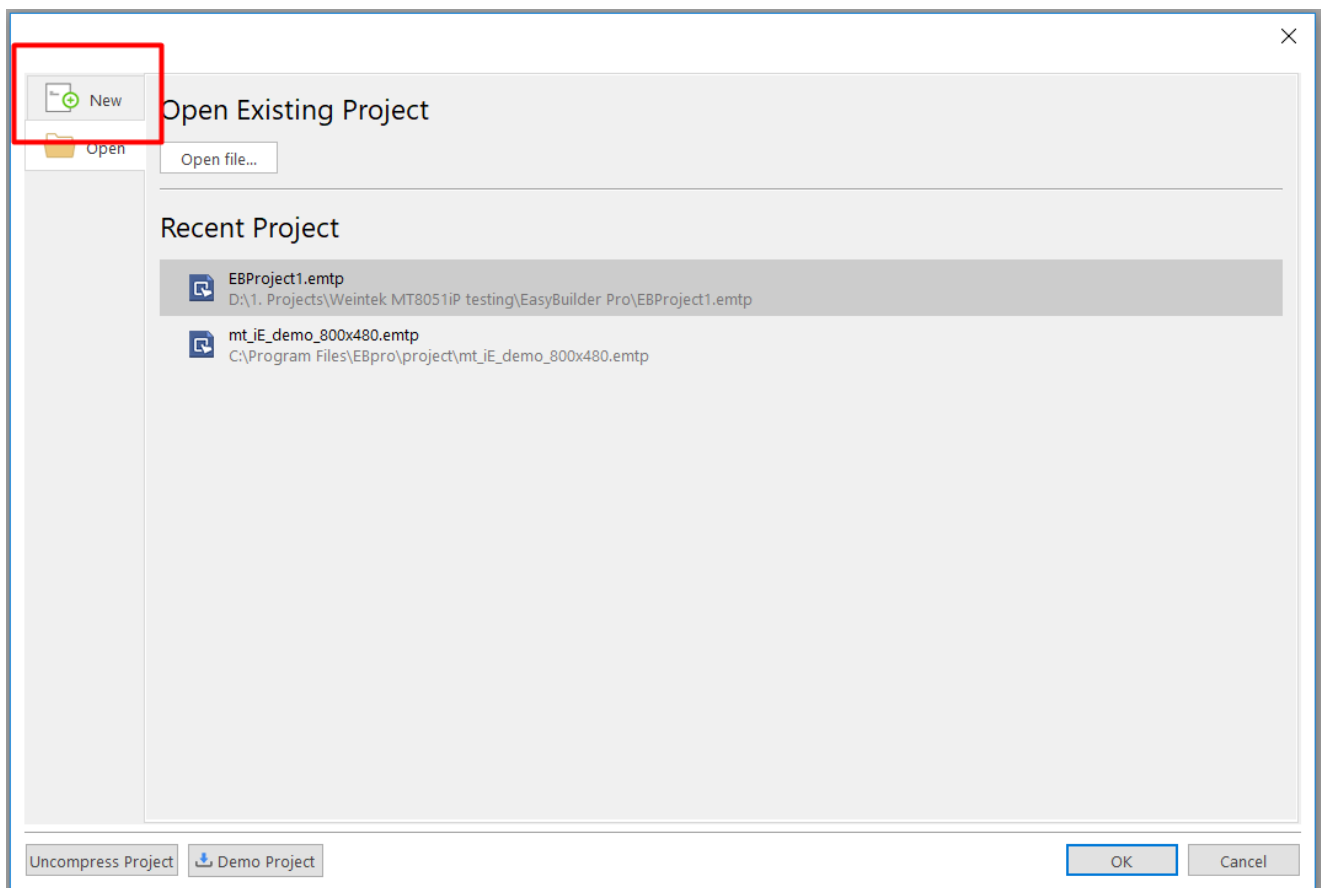
For basic configuration of the UniPi controller and installation of Mervis on your computer, please follow [this guide](#).

For configuration of the HMI we need to install the Weintek EasyBuilder Pro. You can download the version used in this tutorial [here](#), or you can download the latest version from the [manufacturer's website](http://www.weintek.com/) (<http://www.weintek.com/>).

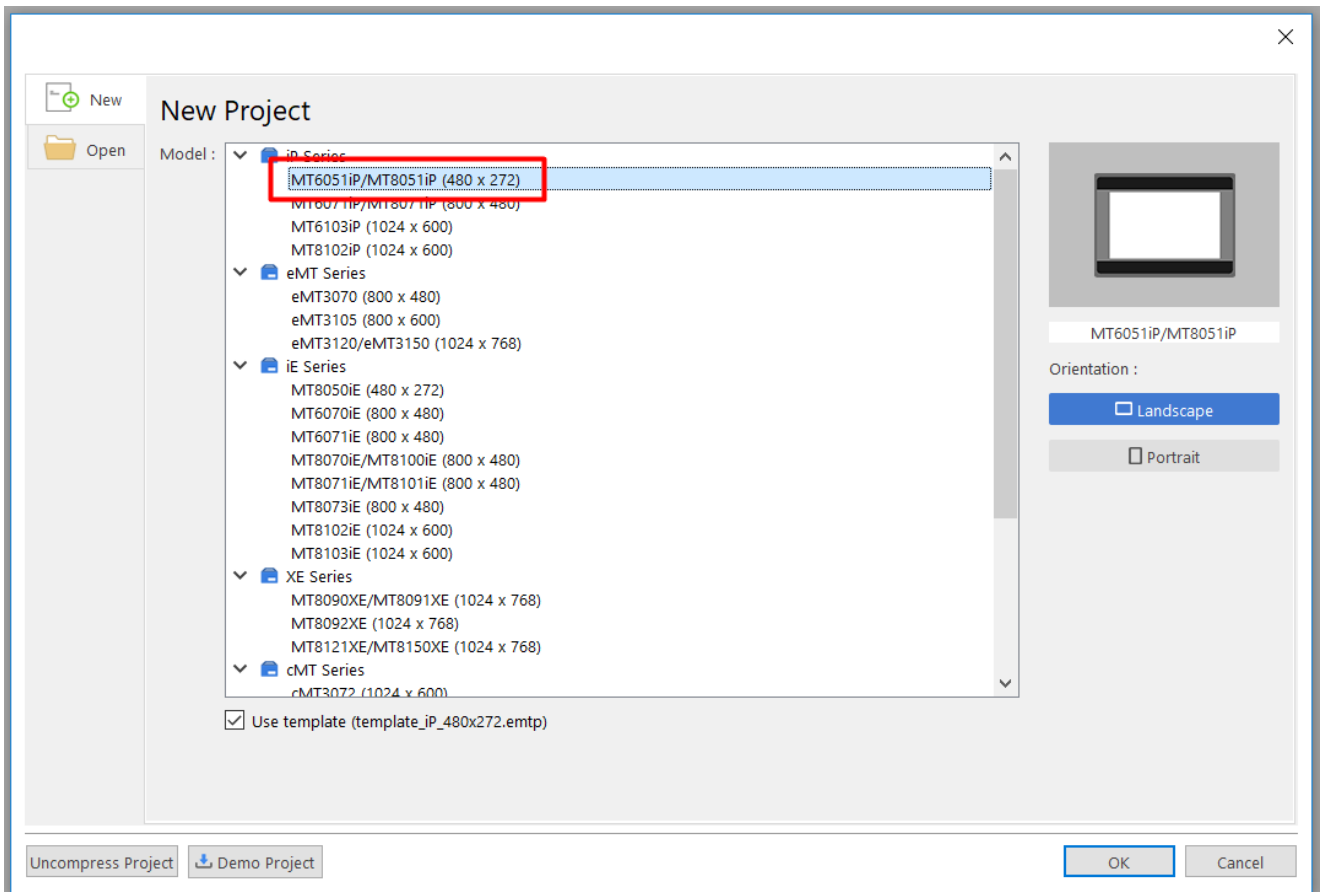
The installation file is not signed by verified publisher, so you will be asked for confirmation. The rest of the installation is easy and you can confirm all the options by clicking on **Next**.

Configuration of the HMI

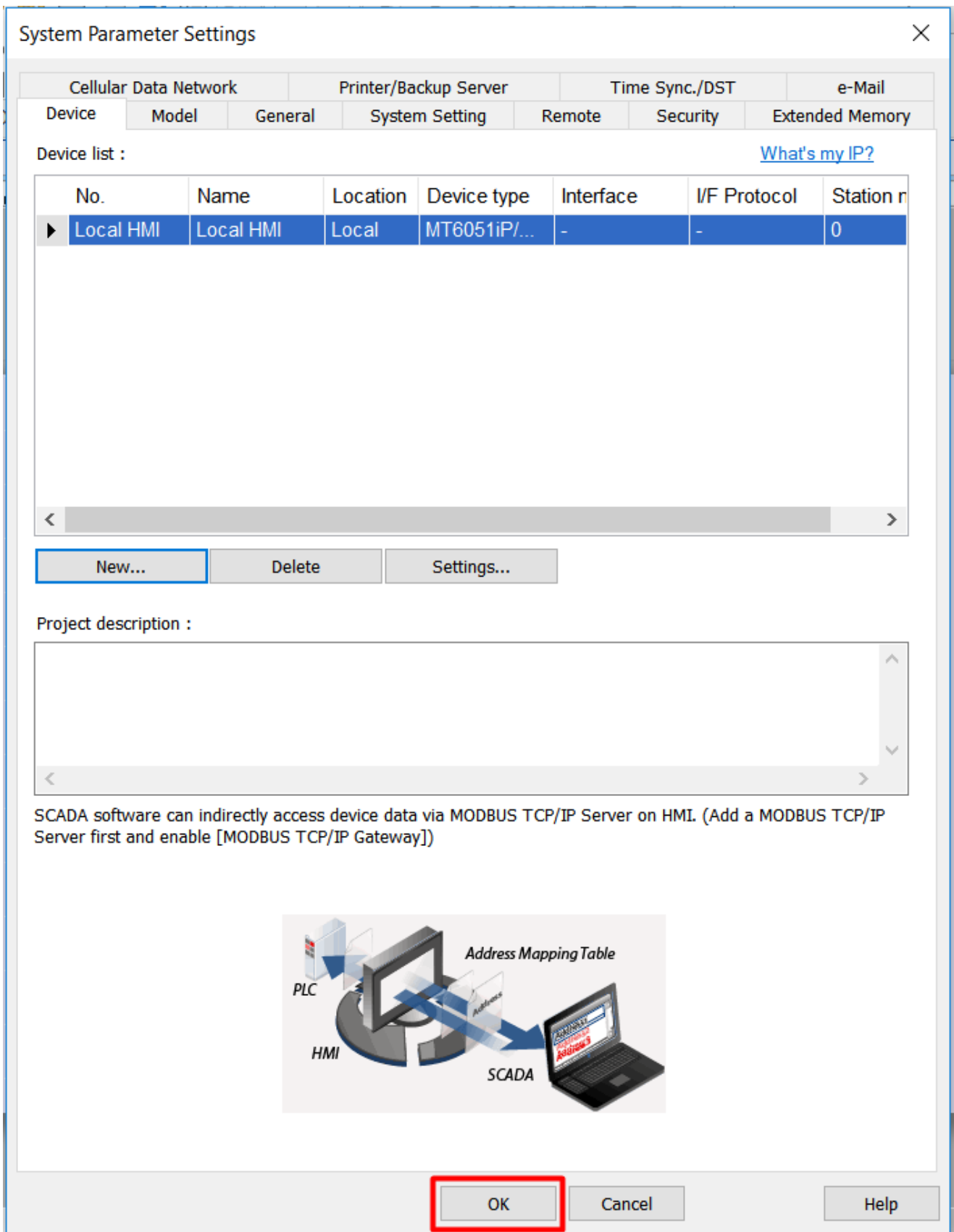
From Start menu on your PC, open the Easybuilder Pro. You will be greeted by the welcome screen, where you can open recent project, or create a new one. We will create a new project by clicking on the **New** button.



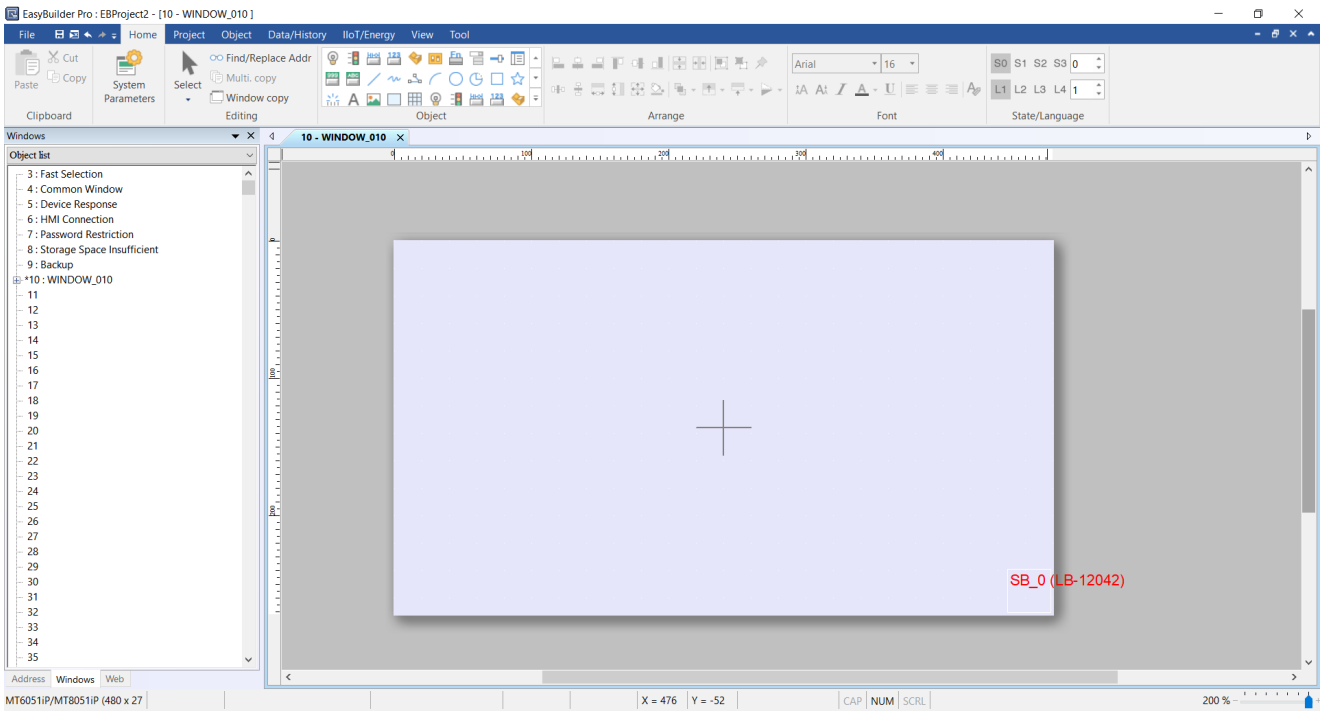
In the next dialog, you have to pick to correct model of the display you want to configure. We will pick the MT8051iP and then hit OK.



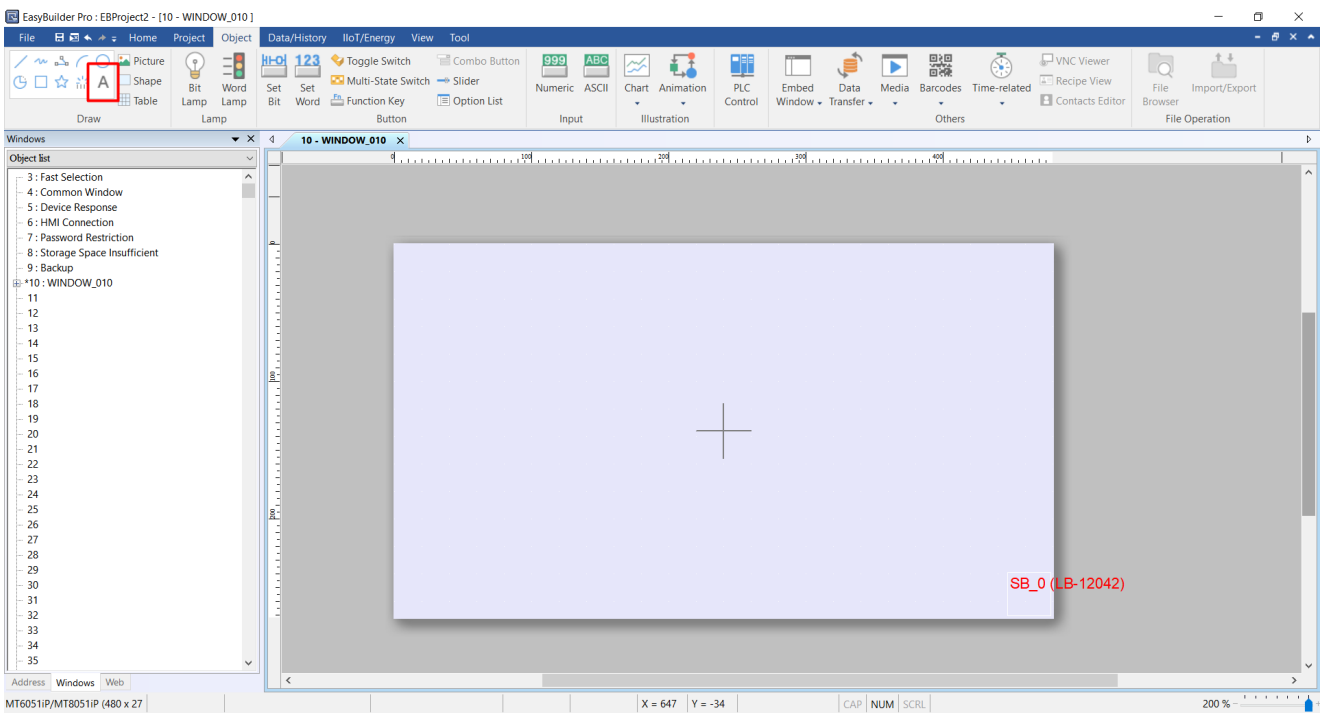
Next dialog is the **System properties**, where we will manage the connections to Modbus slaves or configuration of the Modbus master. We will get to this later, so for now just hit OK.



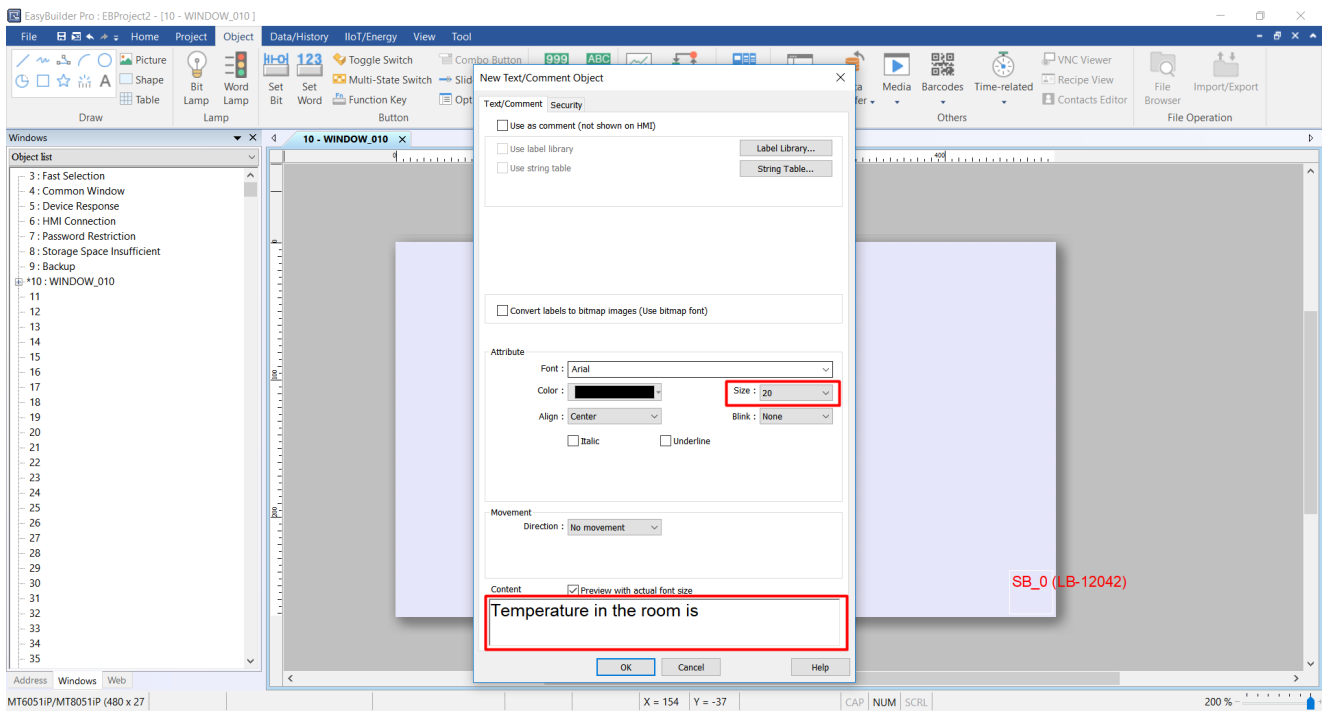
Now you see the main window. On the top of the window, you have a list of panels called **Home**, **Project**, **Object**, etc. We will call this list a **Ribbon**. On the left side, you can see a list of **Windows**. Each window is a graphical representation of what can appear on the HMI display. On the center of the main window, you have a blank window canvas. The window is named WINDOW_010, which is the default window displayed on the device after reboot.



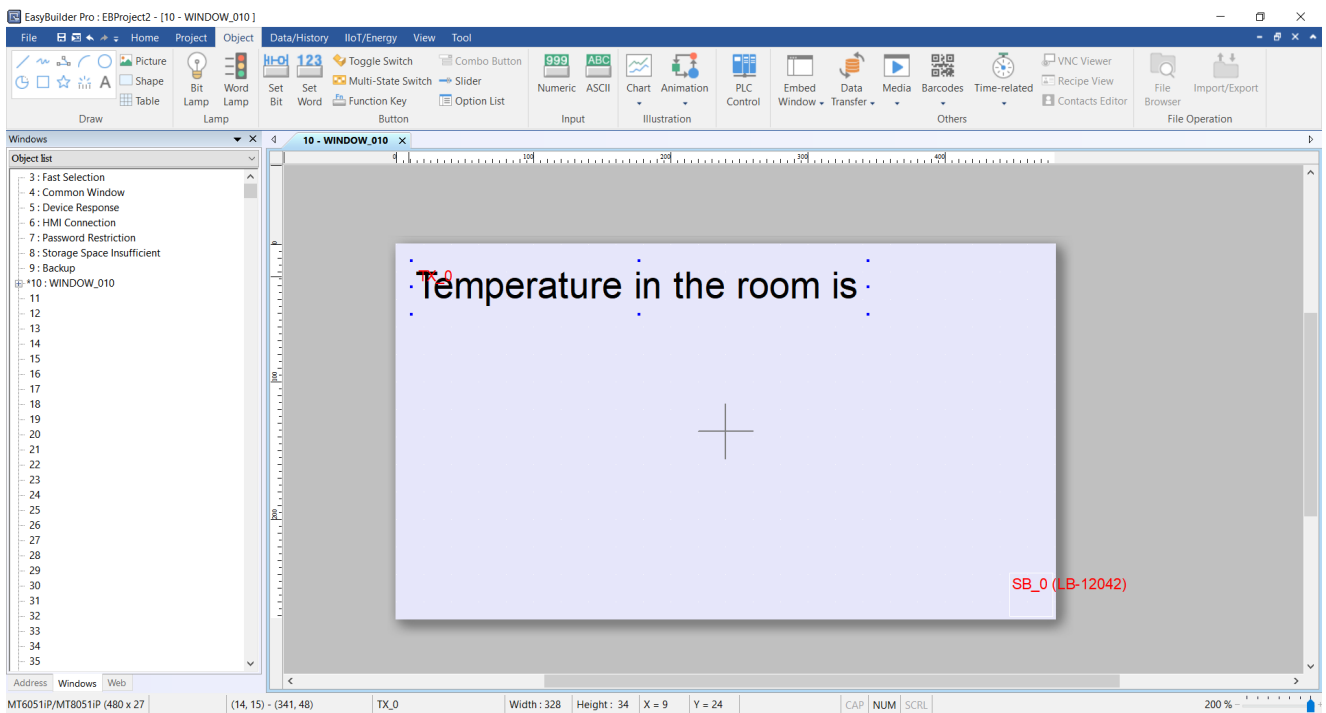
First, we will create some text object and place it onto the canvas. Click on the **Object** on the **Ribbon** and click on **A** in the section **Draw**.



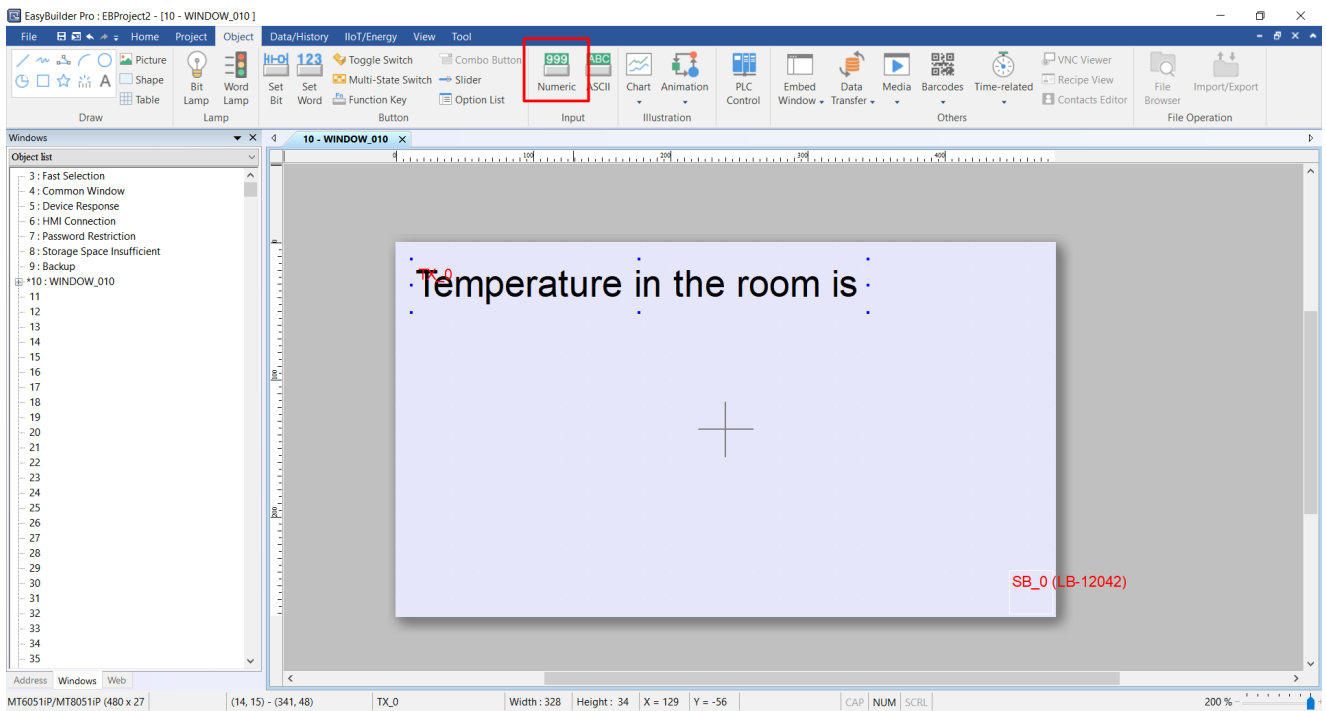
In the **New text** dialog, we can set all different kinds of parameters. We will fill the content and set the font size to 20 and hit **OK**.



After clicking **OK** the dialog will close and you can place the text on the window. The result will look like this:



The text object is static, which means the content cannot be dynamically changed e.g. from some variable or register. For displaying content of local HMI variable or Modbus register, we have to add numeric display box. Click on the **Object** panel on the **Ribbon** and then select **Numeric** in **Input** section.



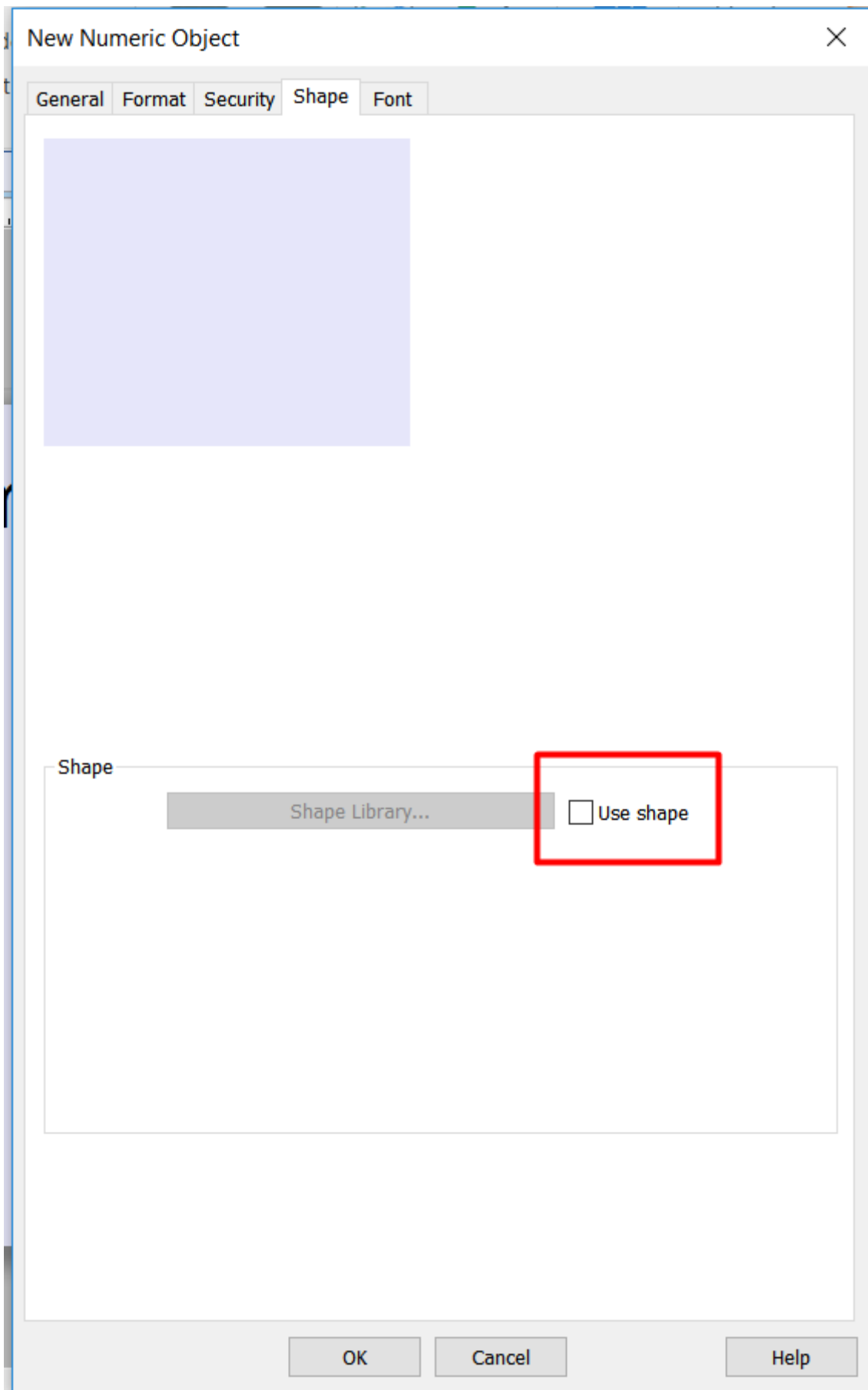
In the **New Numeric Object** dialog, we will set some description. The description acts as a comment - it will not be displayed anywhere on the HMI device. Since we want to display the temperature, we can uncheck the **Allow input**. This will disable the ability of setting the value from touch display. The **Read address** box is for mapping the **Numeric** object to some variable or register. Because we haven't set the communication with UniPi yet, we will leave the **Read address** box as is. Therefore the new numeric object will display the content of local word (LW) variable number 0. Don't bother with this, this is just for demonstration.

The image shows a 'New Numeric Object' dialog box with the following fields and settings:

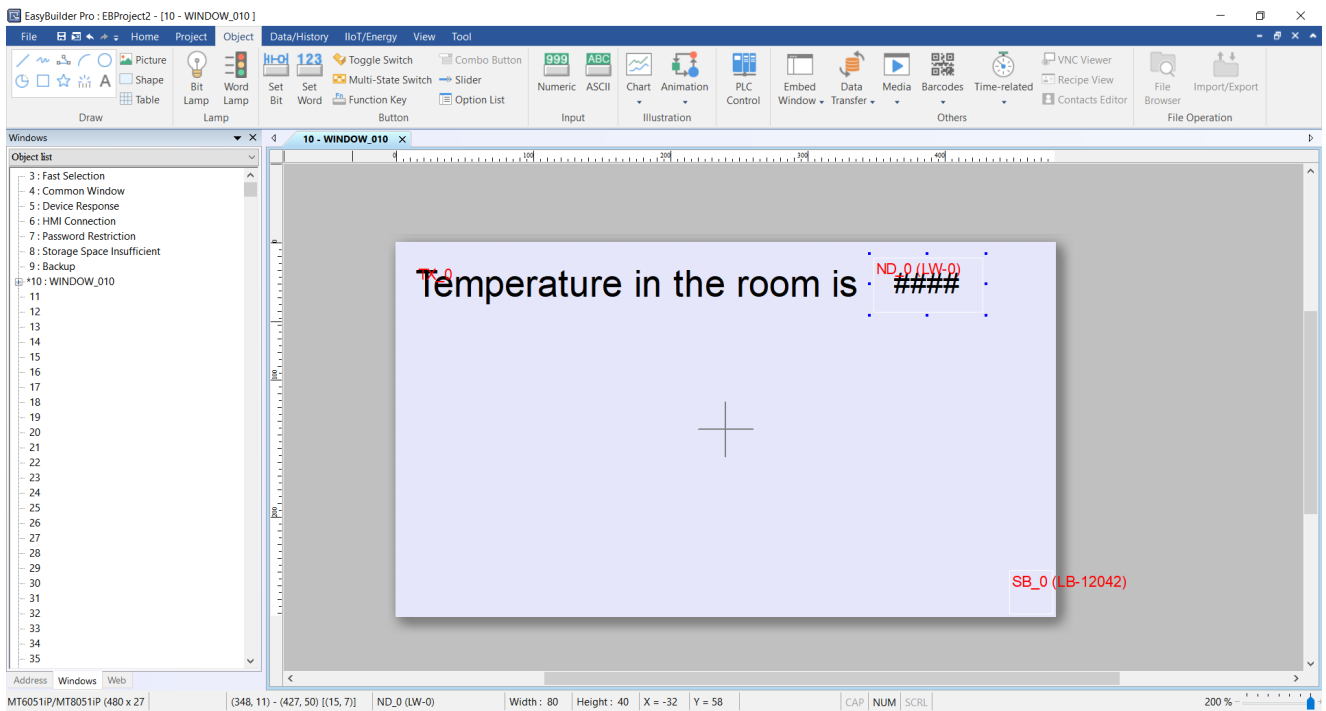
- General** (selected tab)
- Description:** officeTemperature
- Allow input:**
- Read address:**
 - Device:** Local HMI
 - Address:** LW
 - Value:** 0

Buttons: OK, Cancel, Help

The last thing what we will do, is to change the appearance. By default, the Easybuilder will display a nasty border around the object, which we don't want to. In the same dialog, click on the panel **Shape** and uncheck the value **Use shape** and then hit **OK**.

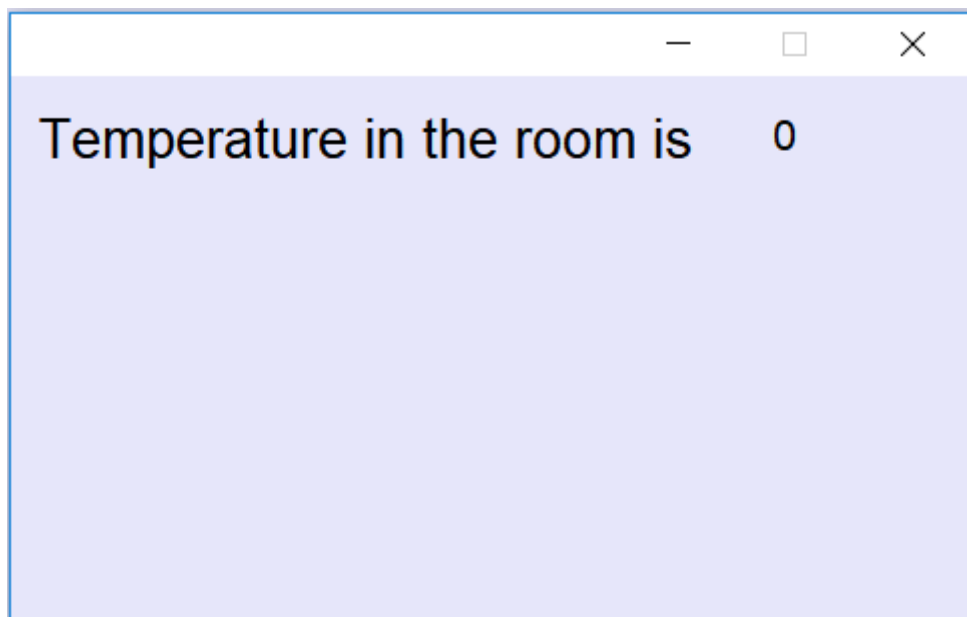


As with the text, you can place the new object on the window. The result can look like this.

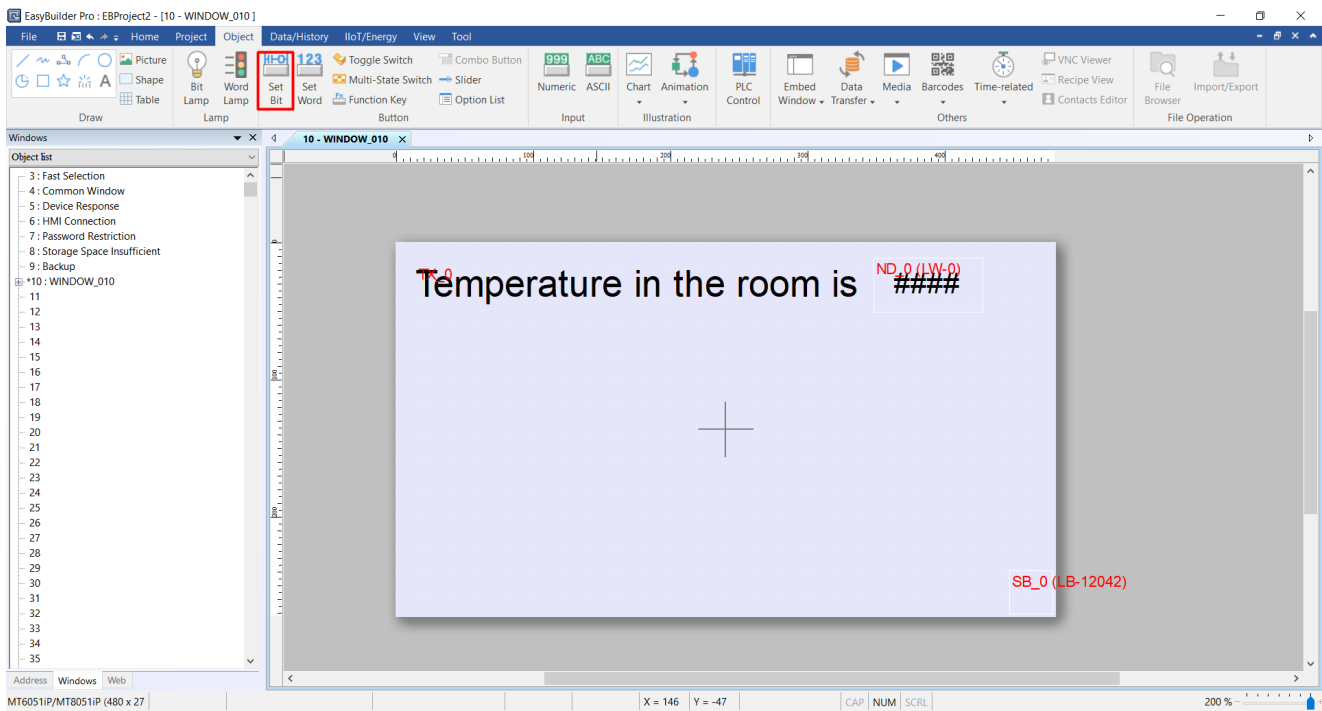


Now that we have some basic window settings, we can check, how it will look on the HMI device. The Easybuilder Pro makes it really easy, because you can simulate the project on your computer, without having the actual HMI device. Click on the **Project** on the **Ribbon** and you can see, that you have two buttons available: **Online simulation** and **Offline simulation**. Both of the buttons will compile the project and run it in local window, and you can interact with it as you would with the physical HMI device. The **Online simulation** will start the simulation and it will also start polling data from configured Modbus slaves, or start the Modbus server so the PLCs can fill the HMI with relevant data. The **Offline simulation** starts the simulation without this polling.

Since we don't have any connection configured, we can start the **Offline simulation**. After clicking, the compilation of the project will appear and after few seconds, you will see the actual simulated project window.



So far we have created simple window, which displays a numeric value of some register. To make this tutorial little bit more versatile, let's learn, how to create a button, which - in the end - will operate relay output on our UniPi. To do this, we have to place new object onto the window canvas, called **Set bit**. On **Object** panel, click on the **Set bit**.



In the **New Set Bit Object** dialog you can fill the comment. The comment will not be shown anywhere in the window, it is just for your better orientation in the project. The box **Write address** looks very similar to the **Read address** block we saw in the **New Numeric object**. But in this instance, we point the output value of the button (ON/OFF, TRUE/FALSE) to some **bit** variable. In numeric object, we pointed it to **word** variable. Since we still didn't configure connection to UniPi, we will leave it as is. What we will change is the **Attribute**. This attribute sets how the button will output the value. We will change it to **Momentary**, which will make the button behave as a momentary switch. When the button is pressed (and hold), it will set the **Write address** register to 1 and when it is released, it will set the register to 0.

New Set Bit Object ✕

General Security Shape Label

Comment : RO_2.01

Write address

Device : Local HMI Settings...

Address : LB 0

Attribute

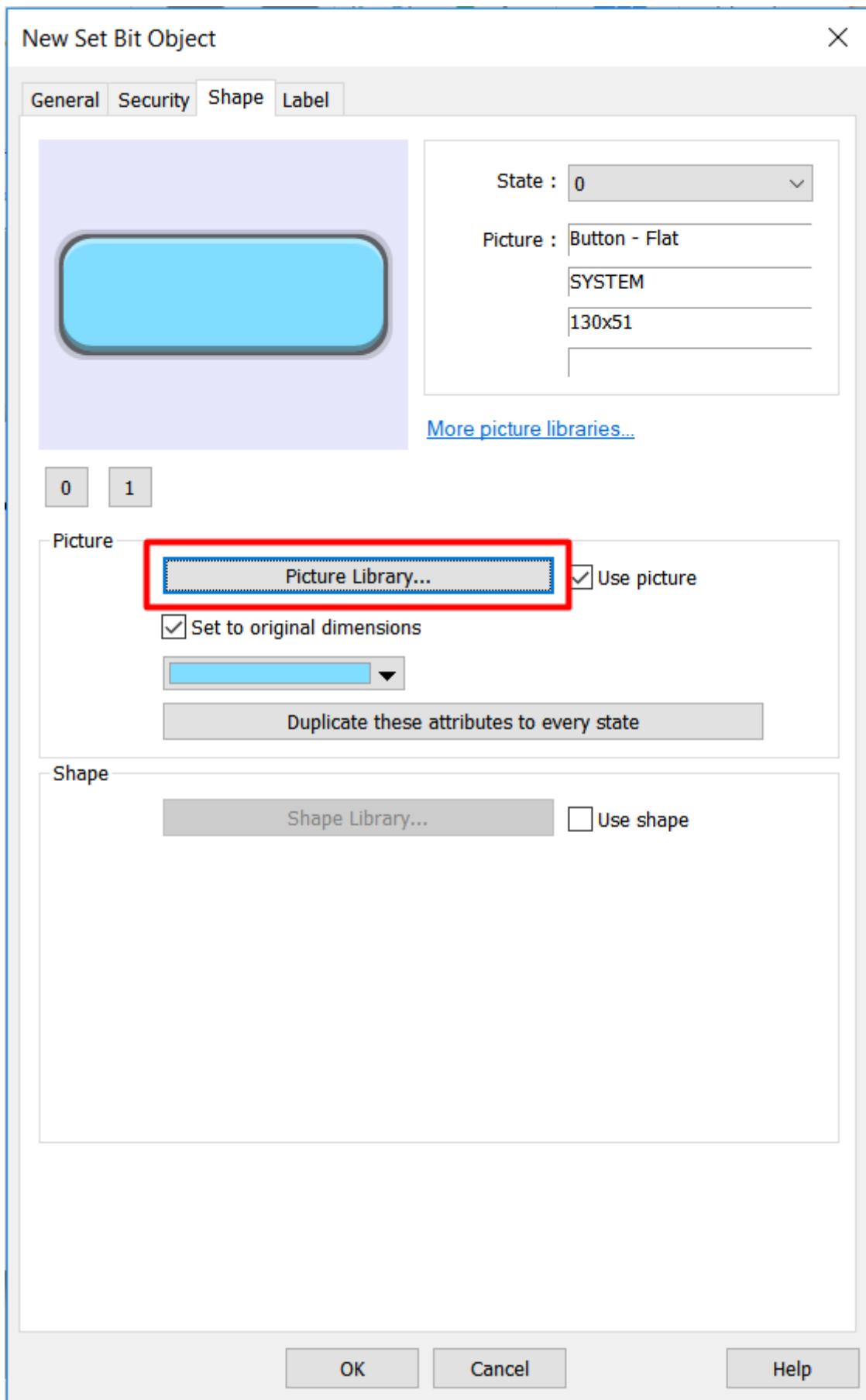
Set style : Momentary

Macro

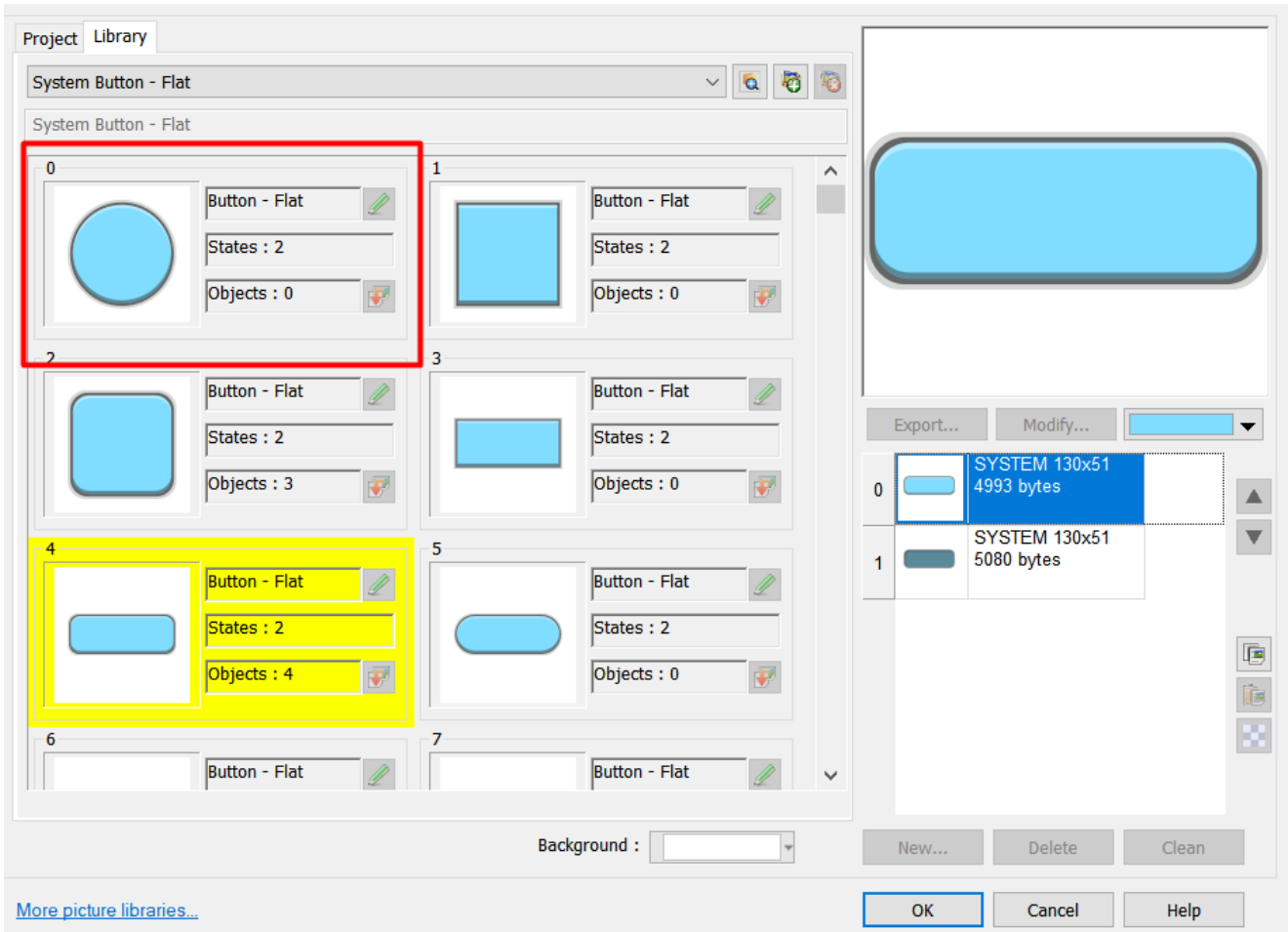
Execute macro

OK Cancel Help

The last thing we will change is the button looks, or in Weintek terminology, **Shape**. Click on the **Shape** panel and change the picture of the button by clicking on the **Picture library**.



In the **Picture library** dialog you can find all different kinds of button images and even have an option to upload your own. We will select the first, round one and then click **OK**.



We are back in the **Shape** dialog and as a last thing, we will change the color of the button to light green and then hit **OK**.

New Set Bit Object



General Security Shape Label

State : 0

Picture : Button - Flat
SYSTEM
74x74

[More picture libraries...](#)

0 1

Picture

Picture Library... Use picture

Set to original dimensions

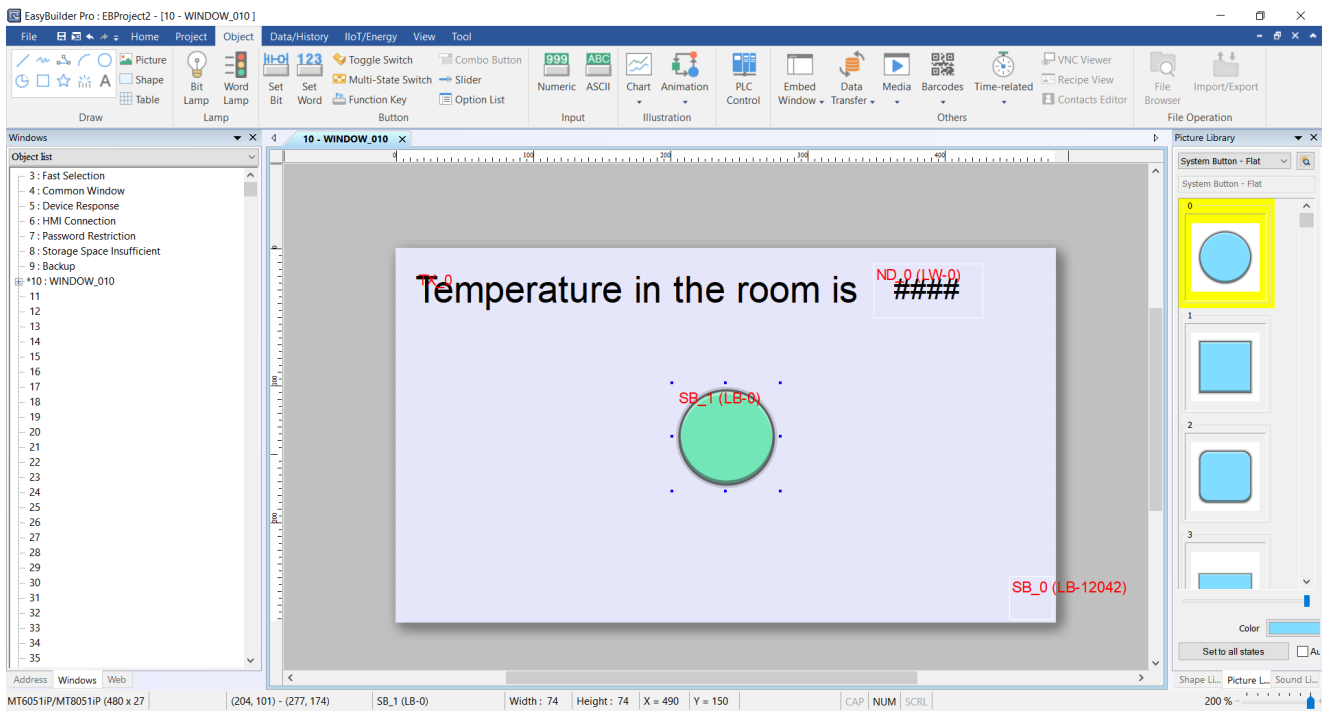
Shape

Contributes to every state

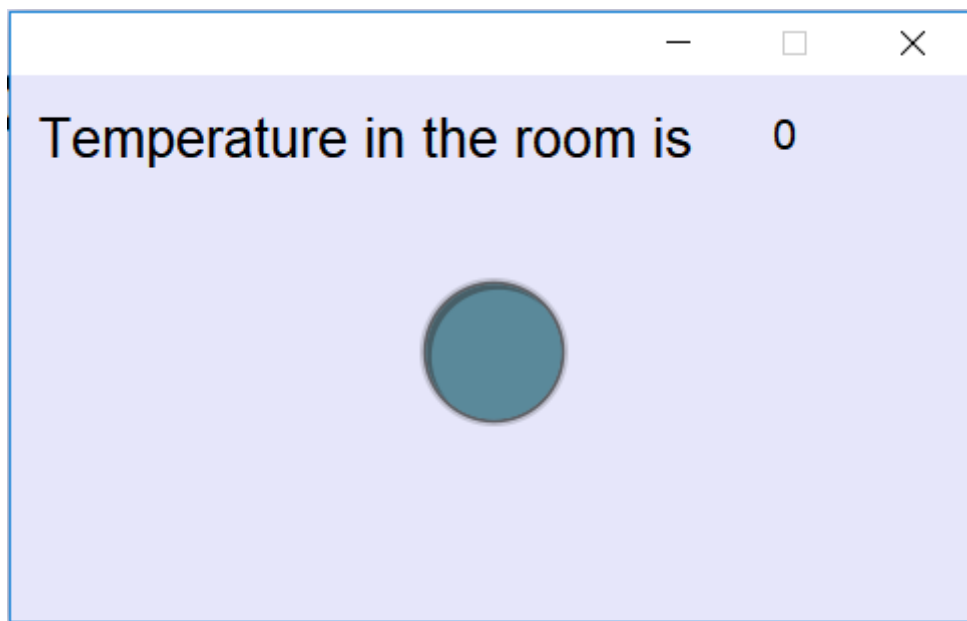
Use shape

OK Cancel Help

Now we can place the button onto the window canvas.

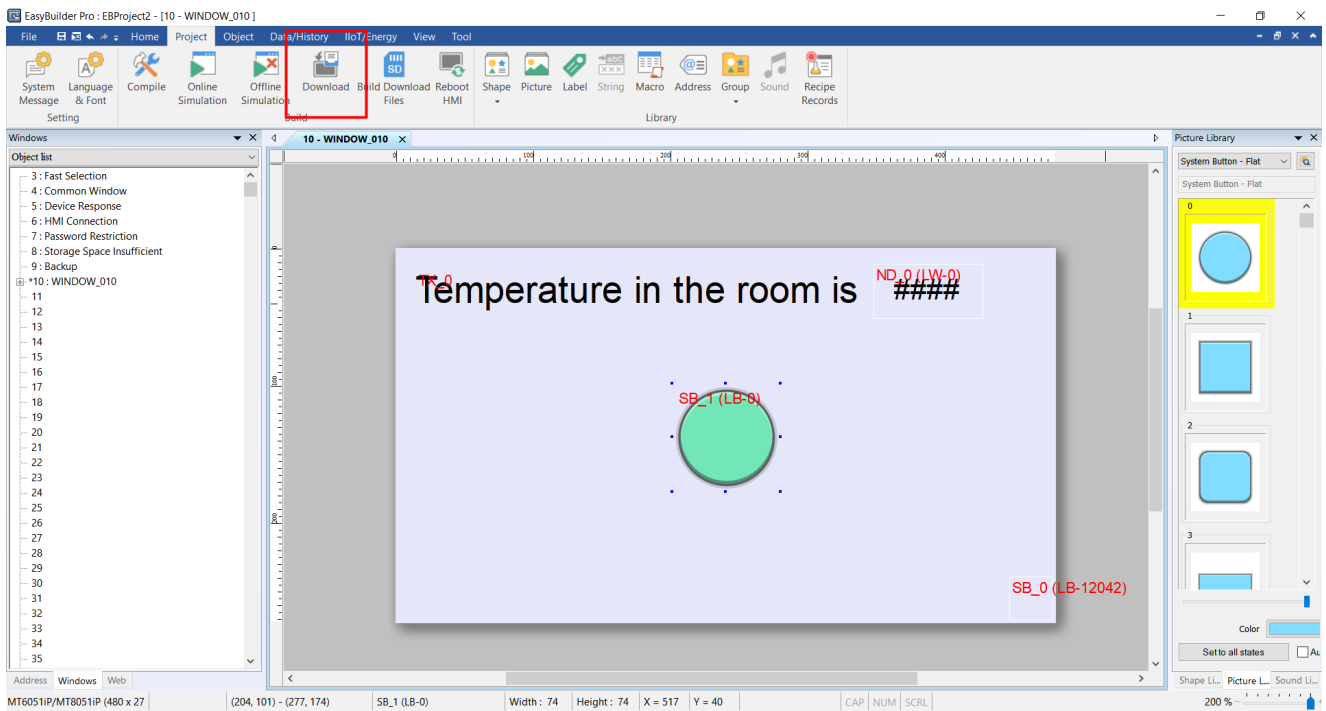


And we can simulate the project by clicking on the **Project** on the **Ribbon** and then on **Offline simulation**. And you can try pressing the button.



Let's download the project into HMI device. First, we need to get its IP address. The device in default settings gets IP address from DHCP server and you can find this information directly on the HMI device. On the touch screen, there is a left arrow icon at the bottom right corner. By clicking on it, a small menu will appear. Then click on the icon of page with (I) and there you can find the IP address.

After acquiring the IP address, we can proceed with the actual download. On the **Ribbon** click on the **Project** and then click on **Download**.



The only operation you have to do in the **Download** dialog is filling the IP address. Then hit **Download**. The HMI device will reboot and start your project. The result should look the same as in the simulation. The temperature will be 0 and pressing the button does nothing.

Communication between HMI and UniPi

So far we have fully functional HMI design and now we have to connect it to UniPi. But first, we need to discuss the data exchange options we have.

The HMI device can communicate with UniPi via two different protocols:

1. ModbusTCP over ethernet
2. ModbusRTU over two wire RS485

In Modbus protocol, the communication is always initialized by the master (client) device. The slave (server) device responds to the commands issued by the master. Master is the one who reads data from slave, or writes data to slave.

Since the UniPi and Weintek iP series HMI can act both as a master and slave, we have to decide which way to go. That depends on your application and overall logic of your system. In this tutorial, we will investigate both options and demonstrate the pros and cons.

UniPi as master, HMI as slave

This is the traditional data exchange method. PLC downloads data from Modbus slaves (UniPi extensions, energy meters,...), decides what to do and then uploads the data to the slaves.

Pros:

Traditional way how to exchange data which will be probably a better fit for your current infrastructure

Cons:

The HMI has to be configured in Mervis as a Modbus slave device with complicated parsing of data

The HMI doesn't have easy method to check, whether the connection to UniPi is running and the registers contain up-to-date data

HMI as master, UniPi as slave

In this setup, the data exchange between HMI and UniPi is initiated by the HMI device. The HMI reads the temperature from UniPi, which shares this information via its ModbusTCP Server register.

Pros:

HMI can detect broken connection and act accordingly

Easier setup on the Mervis side

Cons:

On RS485 connection only one device can act as a master. If the PLC is already a master, you have to use another RS485 connection to HMI

For the rest of the tutorial, we will use this method.

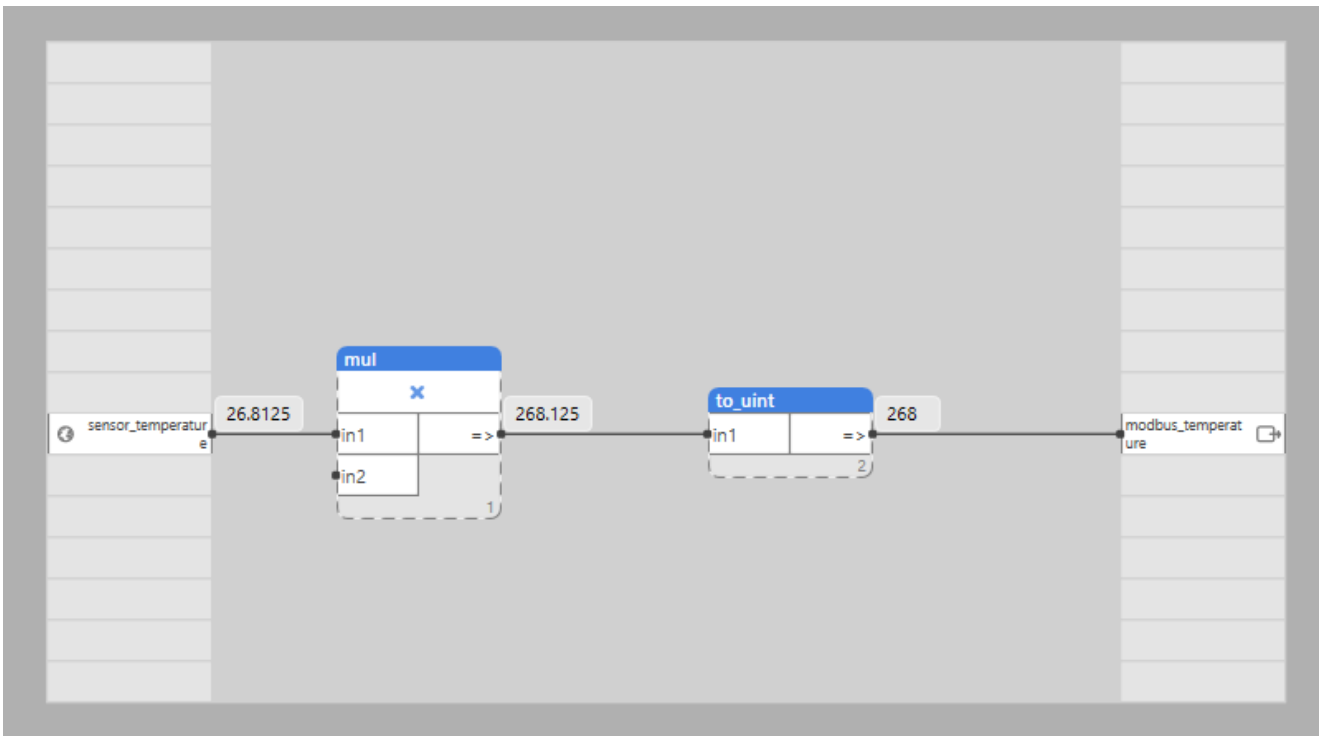
Setting up the UniPi for sharing temperature and relay output

For the purposes of our demonstration, we need UniPi controller running Mervis. You can find all the necessary information in our [Mervis section](#).

Next thing we need to configure is the temperature reading from 1-Wire temperature sensor. You can find it [here](#) how to do so.

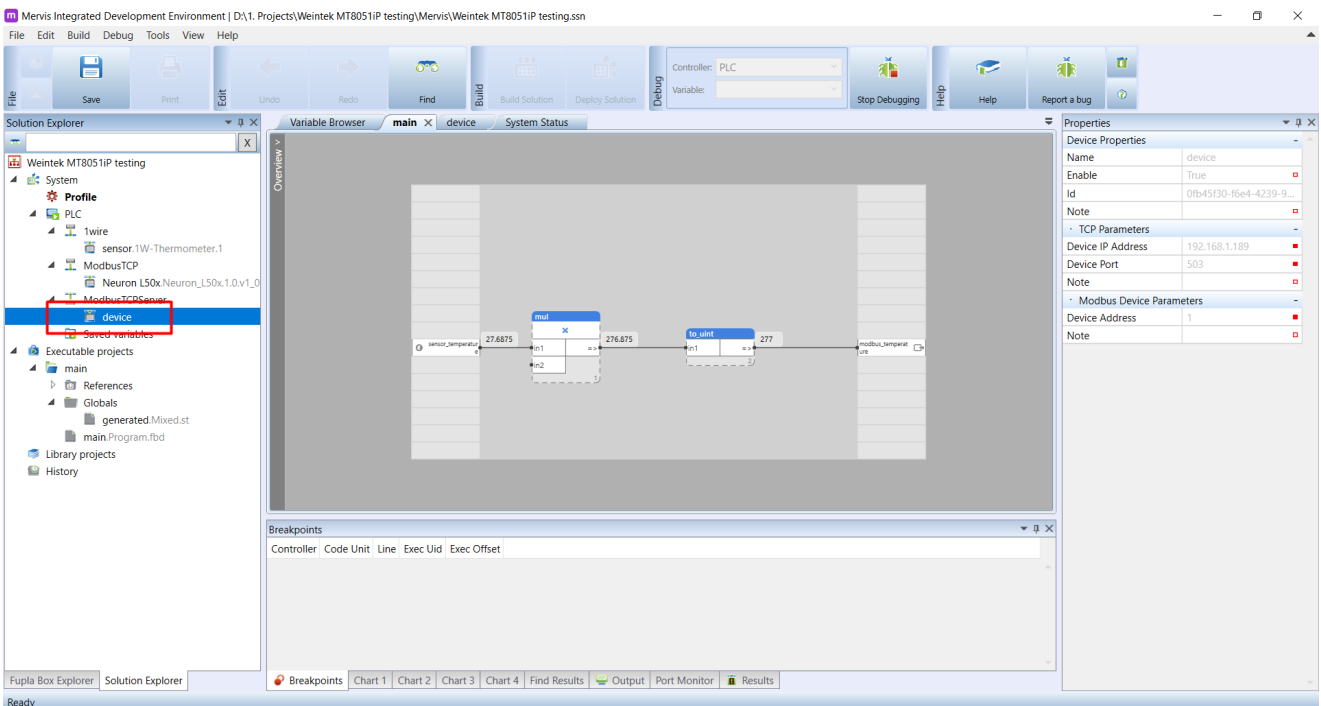
In order to share the temperature over ModbusTCP, you need to configure ModbusTCP server channel. We have a tutorial ready for this as well: [Setting a Modbus server](#).

Now we need to clarify, what type of data we need to set in the Mervis ModbusTCP server. Modbus protocol knows only two types of data: 16bit registers for numeric values and 1bit "coils" for state values. The temperature in the Mervis is represented as a real number - 32bit unsigned integer. To transfer this value via Modbus, we would need to split it into two 16bit registers and put them together on the HMI side, which is out of the scope of this tutorial. But we can convert the real temperature value to roughly fit the 16bit register. Let's assume we measure room temperature, where it can be in range of 0 - 40°C, and we want precision at least to 0.1°C. That means we need to cover values of 0 - 400. That can easily fit into the 16bit register we have. Let's take a look, how to achieve it in FB diagram:

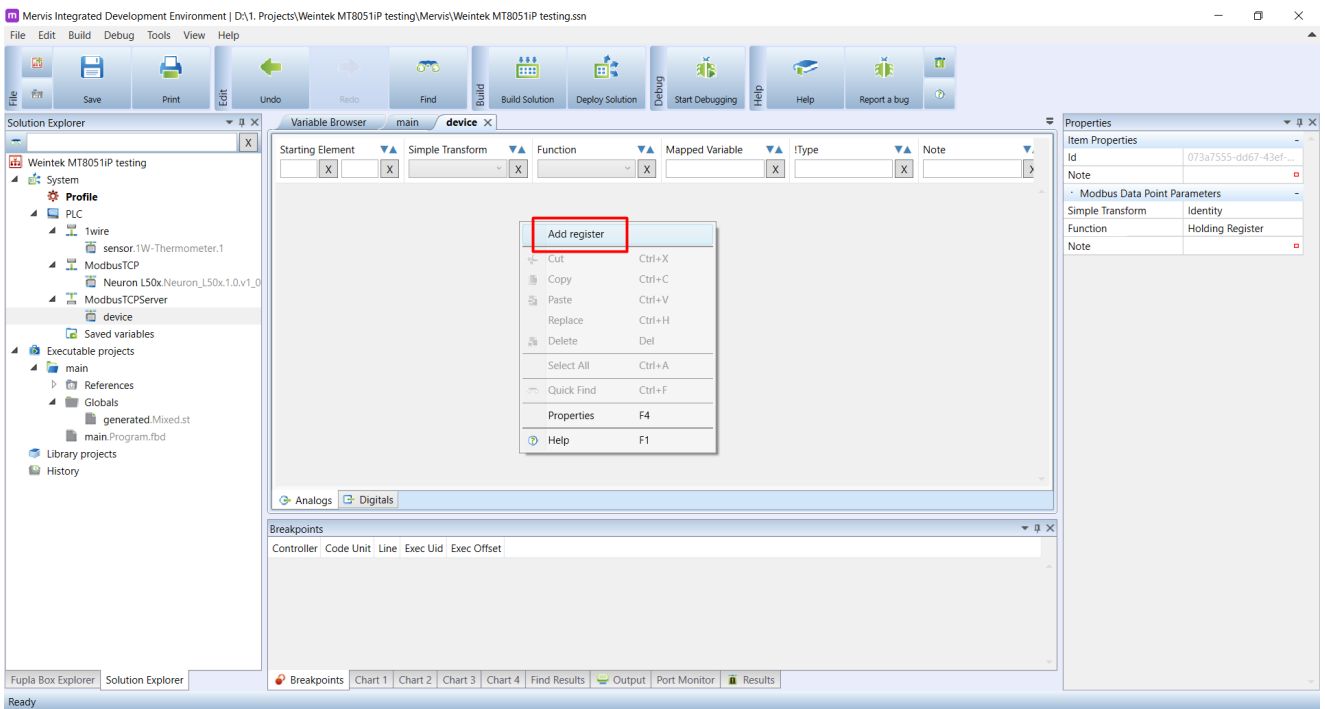


On the input side, we have variable `sensor_temperature` which is the value of the 1-Wire sensor. On the output side we have variable `modbus_temperature` - that's the value exported by ModbusTCP server as holding register number 1. The actual temperature is 26.8125°C. We multiply it by 10 in the `MUL` block to value 268.125. This value is type "real" and we need to convert it to integer to fit the holding register type. The block `TO_UINT` converts any value to integer, and if the input value is less than 0, it will output 0 - hence the name **to unsigned integer**.

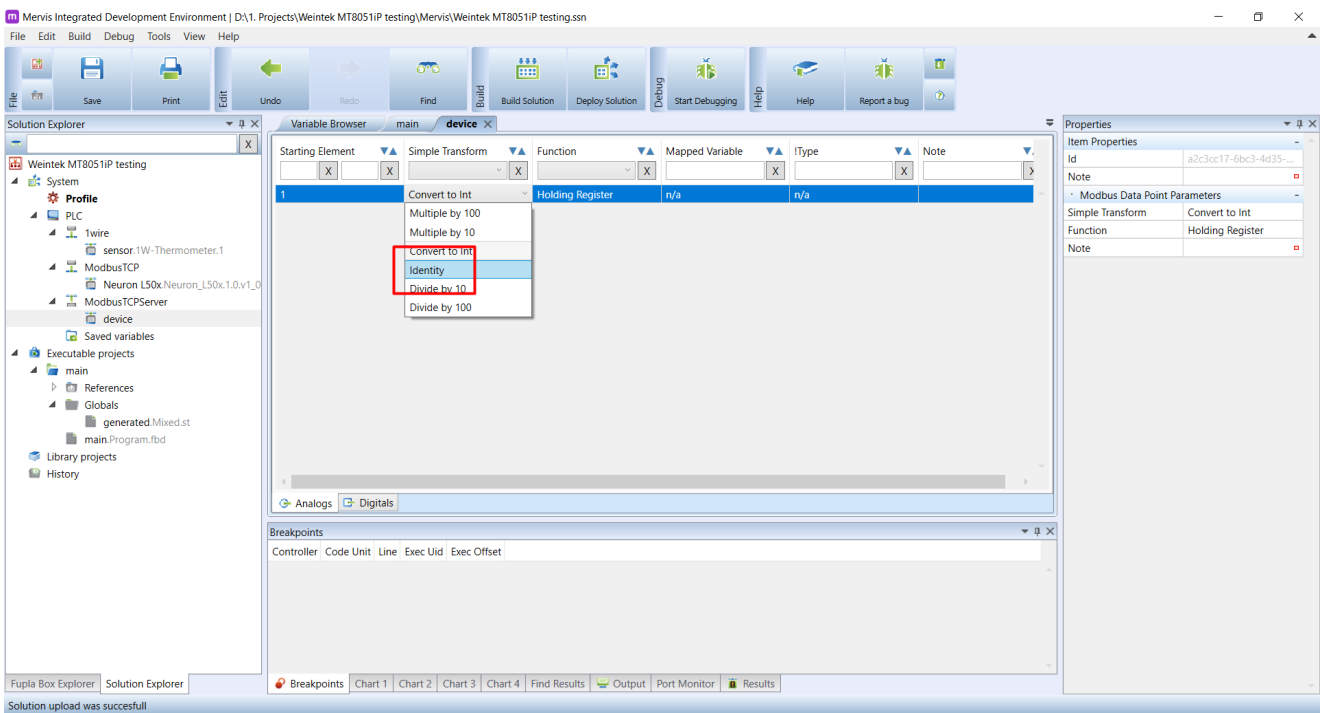
We cheated the measured values a bit, but for our purposes, it works OK. Now we can export this variable in ModbusTCP server. In Mervis, double click on the ModbusTCP server device.



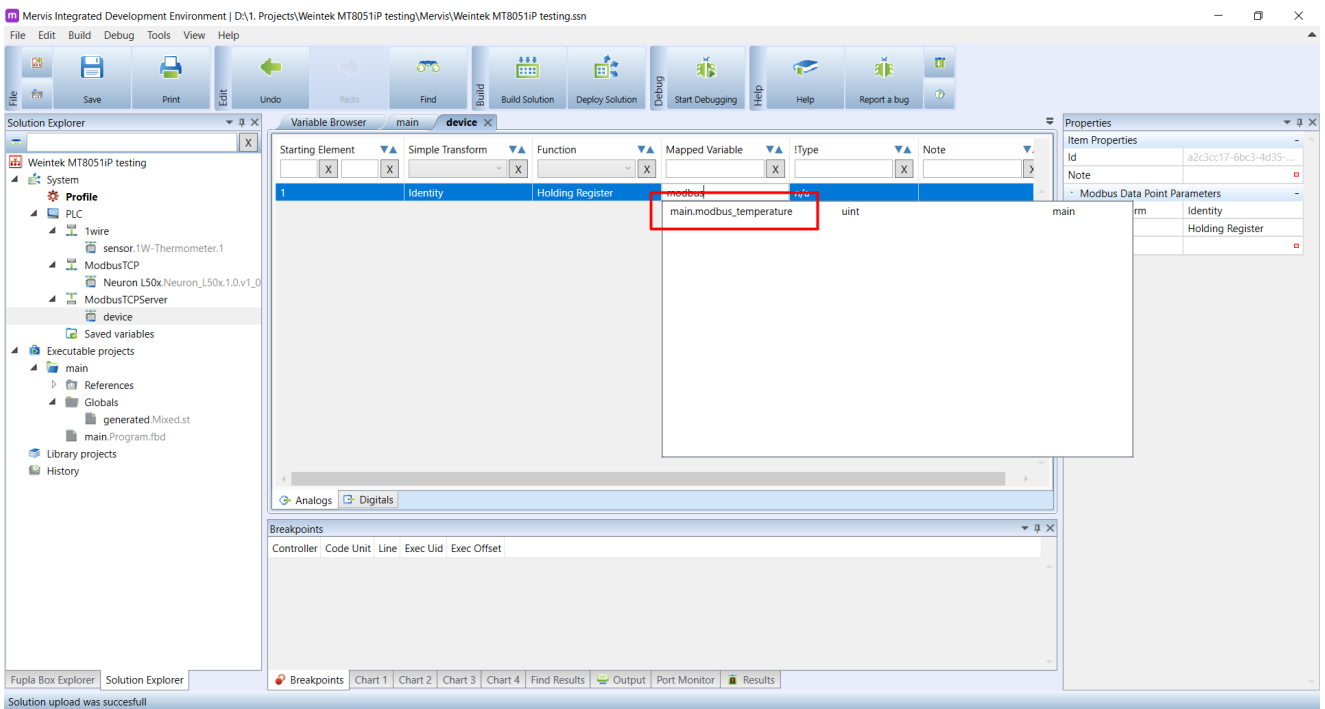
In the center window, the list of registers will appear. The list is empty and we need to add register. Right click in the empty list and select **Add register**.



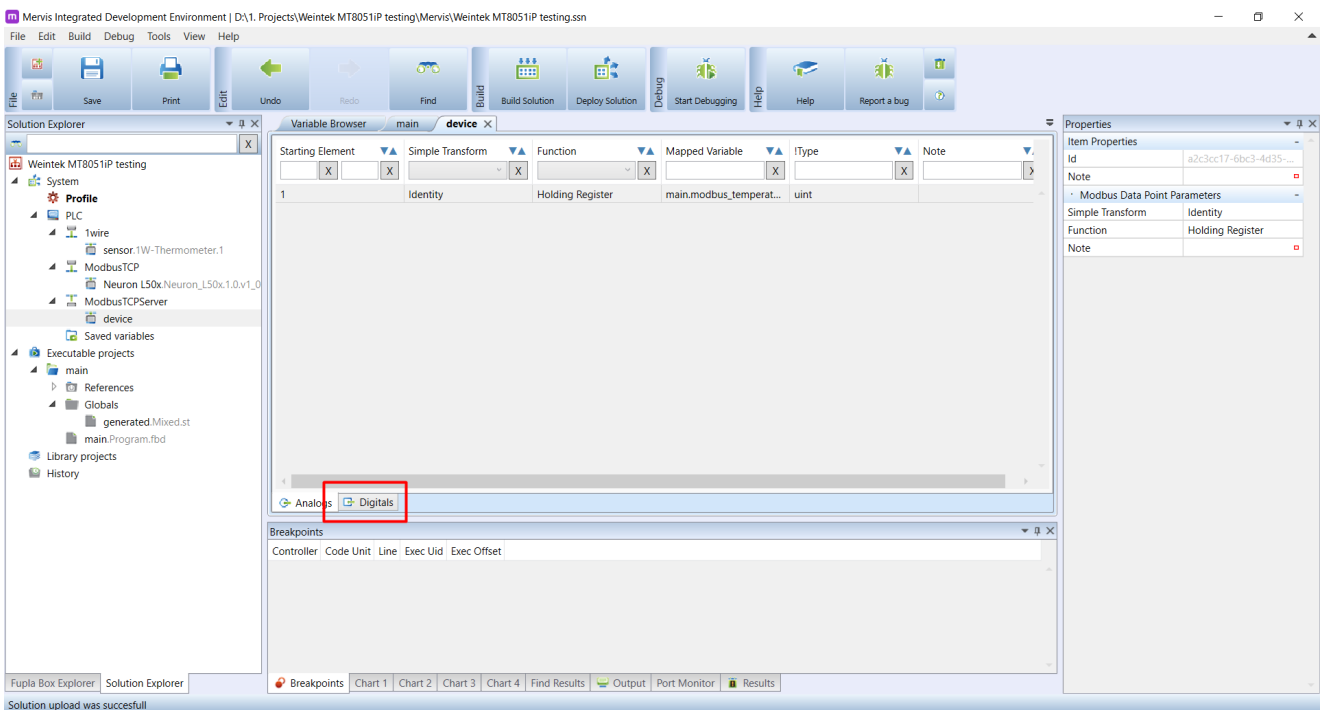
New register will appear. Change the **Simple Transform** value to **Identity**. We already have the modbus_temperature variable in format we want.



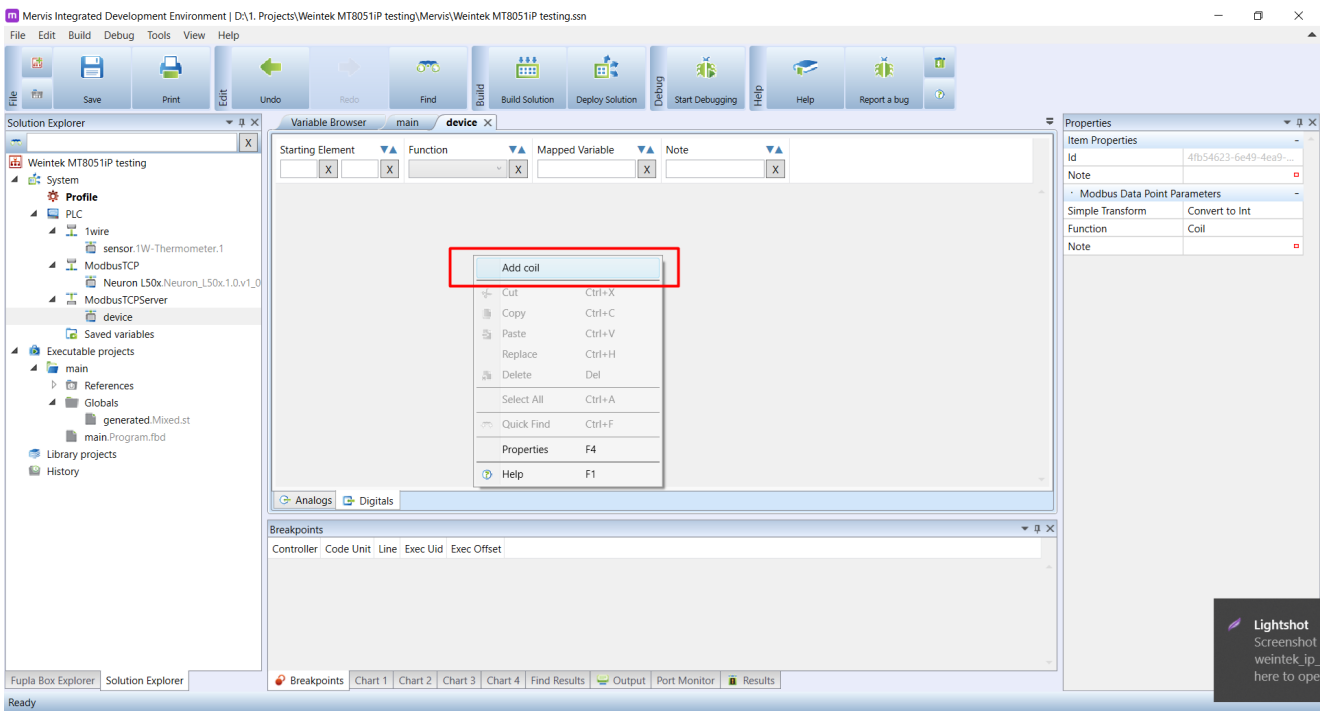
Leave the **Function** to **Holding Register**. Click on the **n/a** under the **Mapped Variable** and start typing the **modbus_temperature**. The variable should appear in the list and you can select it by click on its full name.



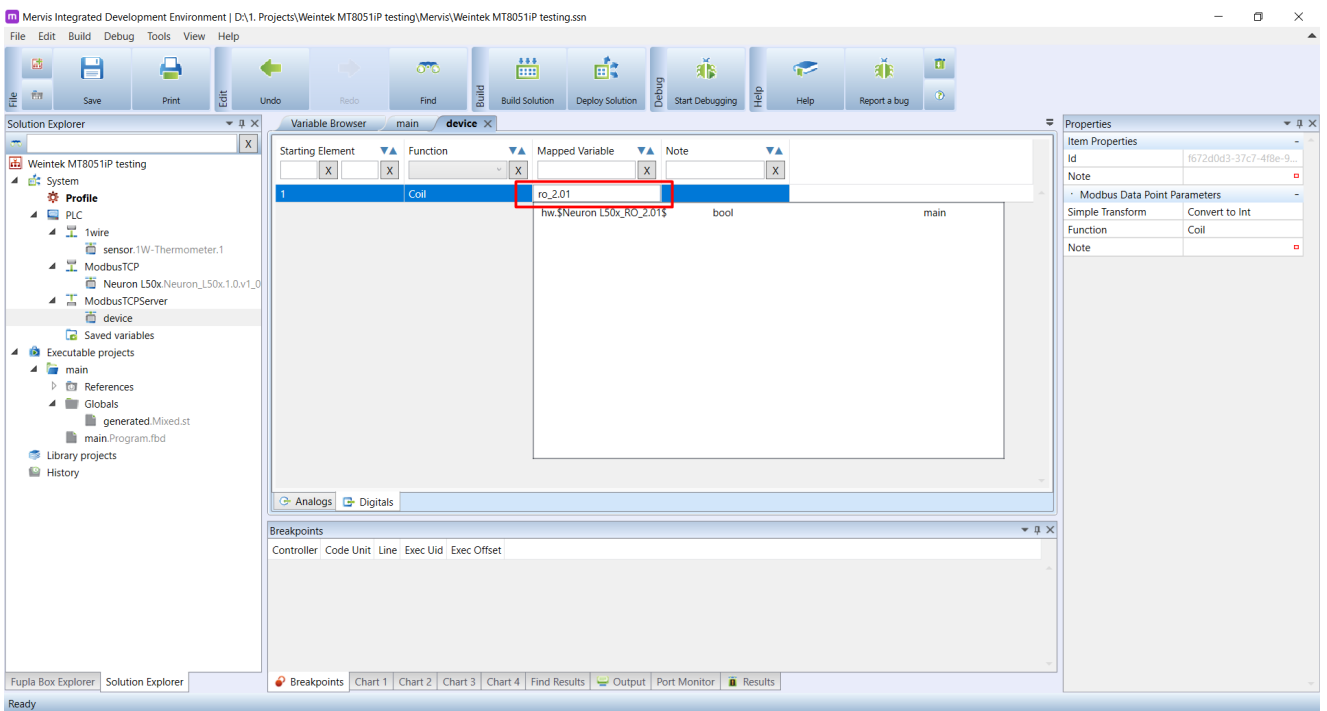
Now we need to add Modbus “coil” to operate relay output. At the bottom of the device panel, there is list panel named **Digitals**. Click on it.



The blank list of digital registers will appear. Right click on the free space and then click on **Add coil**.



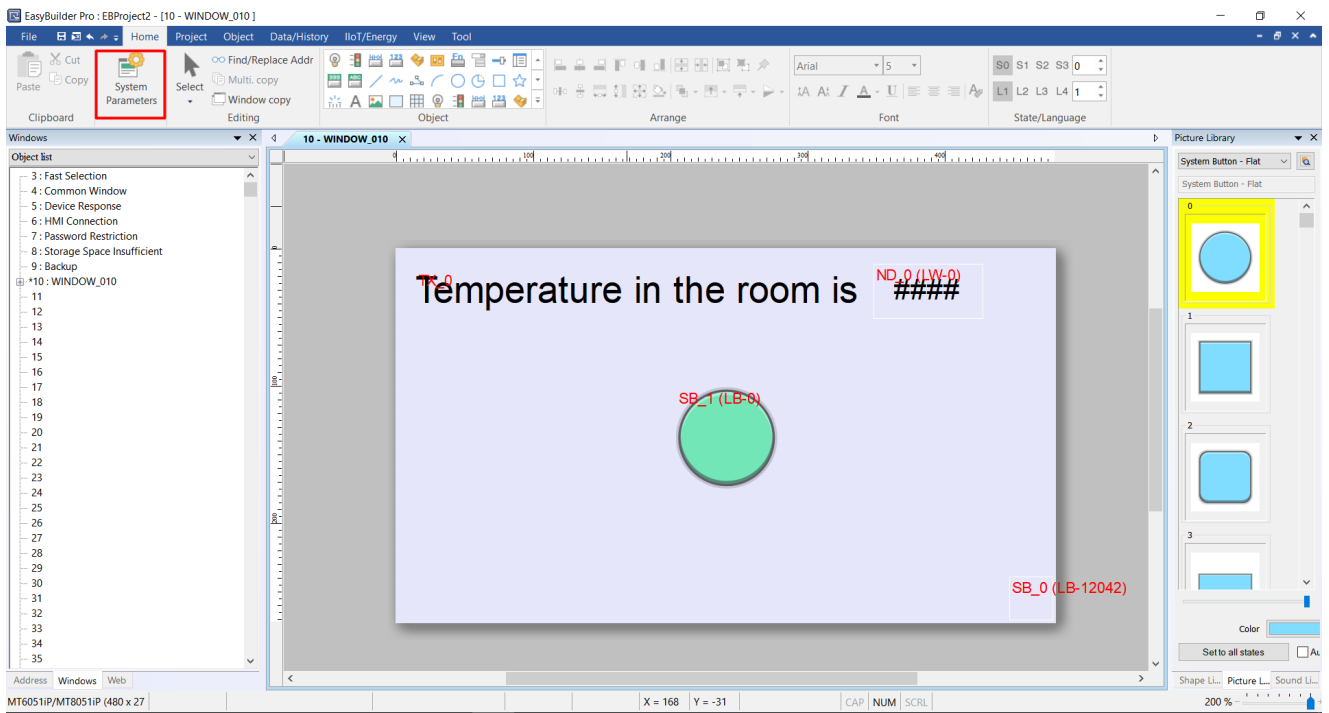
Double click on the **n/a** under the **Mapped Variable** and type **RO.2_01**. If you don't have relay output 2.1 on your unit, you can type DO.1_01. Select the found variable in the list.



Deploy the solution and start the UniPi controller in **Full mode**.

Setting the HMI to read data from Mervis

Now we have everything prepared on the Mervis side and we can configure the PLC in the HMI. On the **Ribbon** click on the **Home** and then on **System parameters**.



On **System parameters** dialog, click on the **New...**

System Parameter Settings

Cellular Data Network Printer/Backup Server Time Sync./DST e-Mail

Device Model General System Setting Remote Security Extended Memory

Device list : [What's my IP?](#)


No.	Name	Location	Device type	Interface	I/F Protocol	Station n
▶ Local HMI	Local HMI	Local	MT6051iP/...	-	-	0

< >

New... Delete Settings...

Project description :

SCADA software can indirectly access device data via MODBUS TCP/IP Server on HMI. (Add a MODBUS TCP/IP Server first and enable [MODBUS TCP/IP Gateway])



PLC HMI Address Mapping Table Address SCADA

OK Cancel Help

Change the **Name** into something descriptive and click on the **Device type** option.

Name : UniPi L503 - office

HMI Device

Location : Local [Settings...](#)

* Select Local for a device connected to this HMI, or Remote for a device connected through another HMI.

Device type : MODBUS RTU, RTU over TCP

PLC ID : 4, V.3.30, MODBUS_RTU.e30

I/F : RS-485 2W [Open Device Connection Guide...](#)

* Support off-line simulation on HMI (use LB-12358)

* Support communications between HMI and device in pass-through mode

* Set LW-9903 to 2 to enhance the speed of download/upload device program in pass-through mode

COM : COM1 (9600,E,8,1) [Settings...](#)

Device default station no. : 1

Default station no. use station no. variable

Use broadcast command

[How to designate the station no. in object's address?...](#)

Interval of block pack (words) : 5 [Address Range Limit...](#)

Max. read-command size (words) : 120 [Data Conversion...](#)

Max. write-command size (words) : 120

[OK](#) [Cancel](#)

From the list of devices select **Modbus IDA** and from the sublist select **MODBUS TCP/IP** and confirm **OK**

Device Properties

The screenshot shows the 'Device Properties' dialog box. At the top, there is a dropdown menu labeled 'MODBUS IDA' with a search button to its right. Below the dropdown is a list of MODBUS protocols. The 'MODBUS TCP/IP' option is highlighted in blue and has a red rectangular box drawn around it. Below the list is a table with columns: 'Address type', 'Bit/Word', 'Address format', 'Max. ad...', 'Min. ...', and 'Shared from'. The table contains 11 rows of data. At the bottom of the dialog, there is a link 'Open Device Connection Guide...', an 'OK' button, and a 'Cancel' button.

Address type	Bit/Word	Address format	Max. ad...	Min. ...	Shared from
4x_String_Central_...	Word	DDDDD	65535	1	4x
4x_string_Central_...	Word	DDDDD	65535	1	4x
6x	Word	DDDDD	65535	1	4x
5x	Word	DDDDD	65535	1	4x
4x_Double	Word	DDDDD	65535	1	4x
4x	Word	DDDDD	65535	1	
3x_Double	Word	DDDDD	65535	1	3x
3x	Word	DDDDD	65535	1	
0x	Bit	DDDDD	65535	1	
0x_single_Bit	Bit	DDDDD	65535	1	0x
0x_multi_coils	Bit	DDDDD	65535	1	0x

In the **IP** box, click on **Settings....**

Name : UniPi L503 - office

HMI Device

Location : Local

* Select Local for a device connected to this HMI, or Remote for a device connected through another HMI.

Device type : MODBUS TCP/IP

PLC ID : 58, V.2.30, MODBUS_TCPIP.e30

I/F : Ethernet

* Support off-line simulation on HMI (use LB-12358)

IP : 192.168.1.111, Port=502

Use UDP (User Datagram Protocol)

Device default station no. : 1

Default station no. use station no. variable

Use broadcast command

[How to designate the station no. in object's address?...](#)

Interval of block pack (words) : 32

Max. read-command size (words) : 120

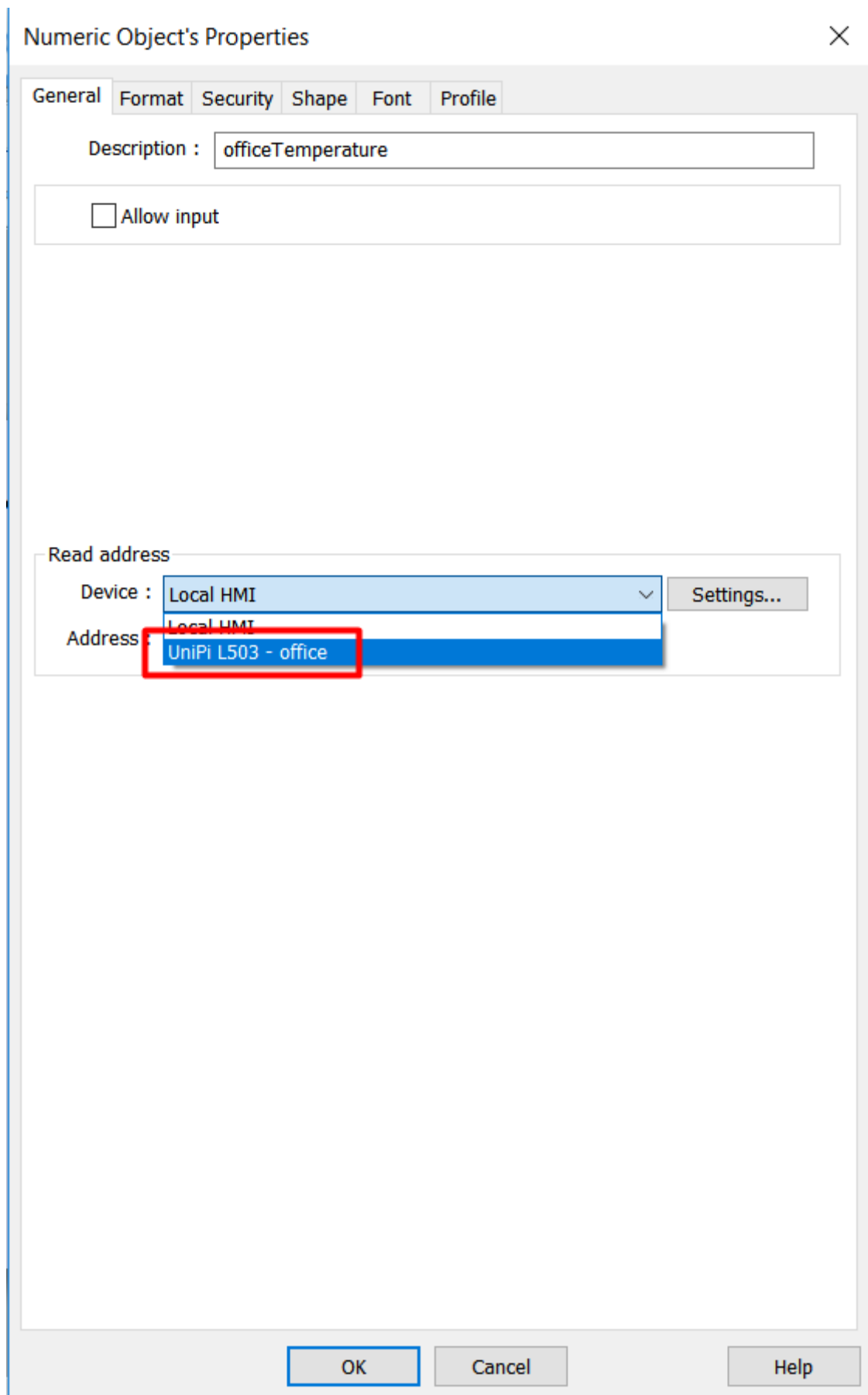
Max. write-command size (words) : 120

In the dialog **IP address settings** set the parameters that you have setted in the ModbusTCP server settings in Mervis. Confirm all the dialogs by clicking on **OK** until you will see the main window.

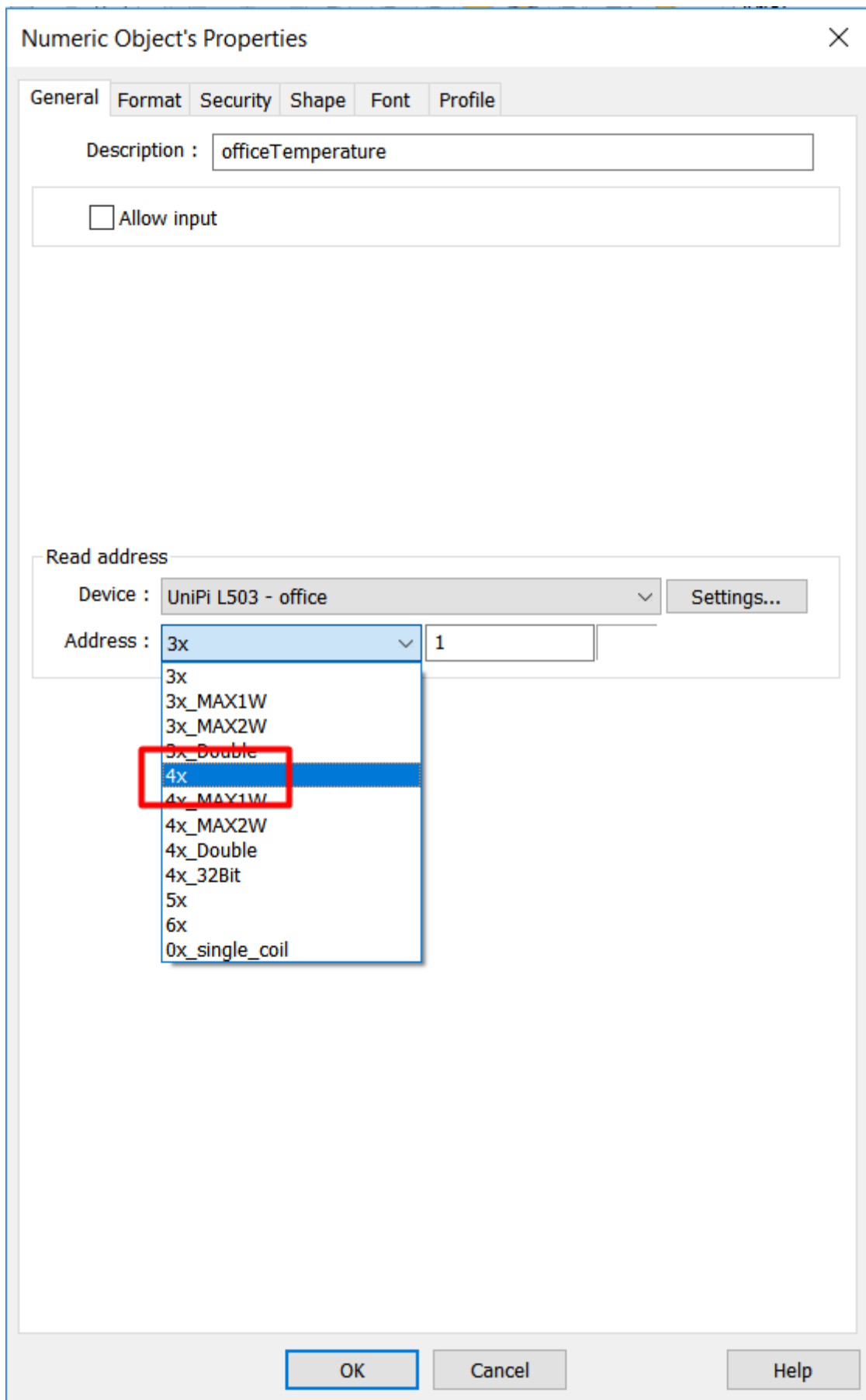
IP Address Settings

IP address :	<input type="text" value="192 . 168 . 1 . 189"/>		
Port no. :	<input type="text" value="503"/>		
Timeout (sec) :	<input type="text" value="1.0"/> ▾	Turn around delay (ms) :	<input type="text" value="0"/>
The number of resending commands :		<input type="text" value="0"/> ▾	
<input type="button" value="OK"/>		<input type="button" value="Cancel"/>	

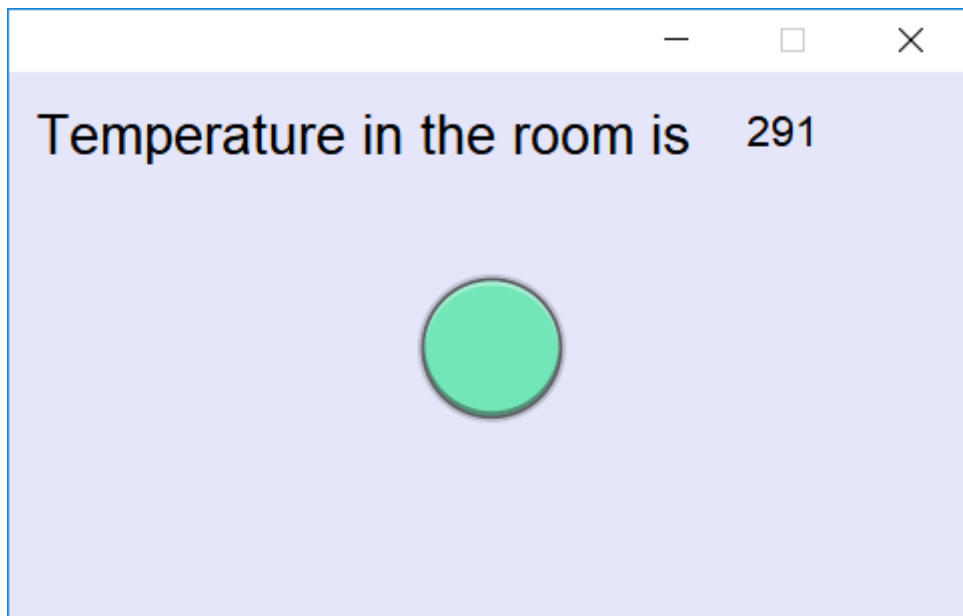
Now that we have the connection configured, we need to point the **Numeric** and **Set bit** objects to the correct devices and registers. Double click on the temperature **Numeric** object. In the **Numeric Object's Properties** dialog, select the UniPi controller in the **Device** drop down menu.



Select the **Address** to **4x** which is a designation of holding registers and leave the address number to 1 - this is the number of the register in Mervis.



Now we can run the **Online simulation** by clicking on the **Project** tab on the **Ribbon** and then clicking on the **Online simulation**. And if everything is OK, you should see similar window.



You can see the number 291, which is the converted number. But we want to display it in 29.1 format. To achieve this, we have to change the format of the numeric object. So exit from the simulation and again double click on the temperature's numeric object. In the **Numeric Object's Properties** select the **Format** panel. On the panel, change the **Left of decimal Pt.** to **2** and **Right of decimal Pt.** to **1** and confirm by clicking **OK**.

Numeric Object's Properties ✕

General **Format** Security Shape Font Profile

Display

Device data format : 16-bit Unsigned Mask

Number of digits

Left of decimal Pt. : 2 Right of decimal Pt. : 1

Display format

Enable

Scaling

Method : None

Limits

Direct Dynamic limits

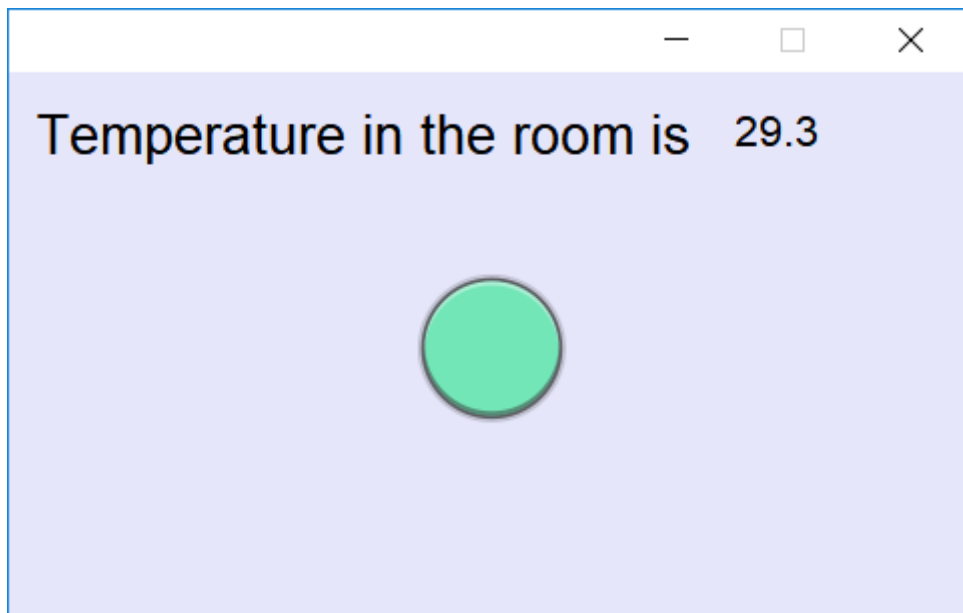
Device low : 0 Device high : 999

Input low : 0.0 Input high : 99.9

Use alarm color

OK Cancel Help

Run the **Online simulation** again and the temperature should look OK



The last missing step is the address on the button. Double click the button and on the **Set Bit Object's Properties** change the **Write address** device to the UniPi controller, **Address** to **0x** which is a designation of "coil" and the coil number leave at **1**.

Set Bit Object's Properties



General Security Shape Label Profile

Comment :

Write address

Device UniPi L503 - office

Address 0x

Attribute

Set style : Momentary

Macro

Execute macro

Run the **Online simulation** and try pressing the button. If you set it to relay output in Mervis, you should hear the clicking sound of the relay. If you set it to digital output, the LED on the controller should be turning on and off.

If everything seems to be OK, you can download the project into the HMI device.

Downloads

[EasyBuilder project files](#)

[Mervis project files](#)

[Weintek EasyBuilder Pro software](#)

[Weintek EasyBuilder Pro complete manual](#)

[Weintek MT8051iP datasheet \(https://www.unipi.technology/shop/product/download?fileId=404\)](https://www.unipi.technology/shop/product/download?fileId=404)

[\(Still can't find what you're looking for?\)](#)