webdyn

Flexitron group

Titan Router

V6 Firmware

Sending data to
ThingsBoard platform

Scenario Details

TITAN routers have all the typical functionalities of 4G/3G/2G routers, as well as a series of added features that make them one of the most feature-packed routers on the market.

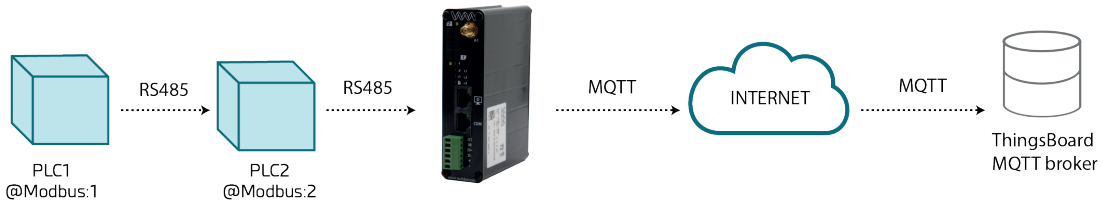
One of the added features is the datalogger, where the TITAN router can store a number of types of records in its non-volatile memory in JSON format. These records can come from MODBUS readings, SERIAL data captures via the RS232 / RS485 ports, or GPS positions, etc. These JSON-type records are stored in the TITAN router's internal non-volatile memory and can subsequently be sent to remote platforms via protocols such as HTTP, HTTPS, MQTT, MQTTS, FTP and FTPS.

As mentioned, the TITAN router stores the JSON registers in its internal memory in a proprietary format by default. This can sometimes be a problem when communicating with platforms that expect to receive information in a certain format (i.e. a format other than JSON, the one used by the TITAN router).

The "JSON Transformer Function Script" enables the TITAN router to format any JSON object before it is stored in the internal memory. This means JSON objects can be converted to the appropriate format for each platform. You don't really need to convert the data into a JSON format, since it can be converted into any String format.

In this application note, we will guide you through an entire example of how send to data to the well-known ThingsBoard platform (https://www.thingsboard.cloud/), which requires the sent JSONs to be in a special format.

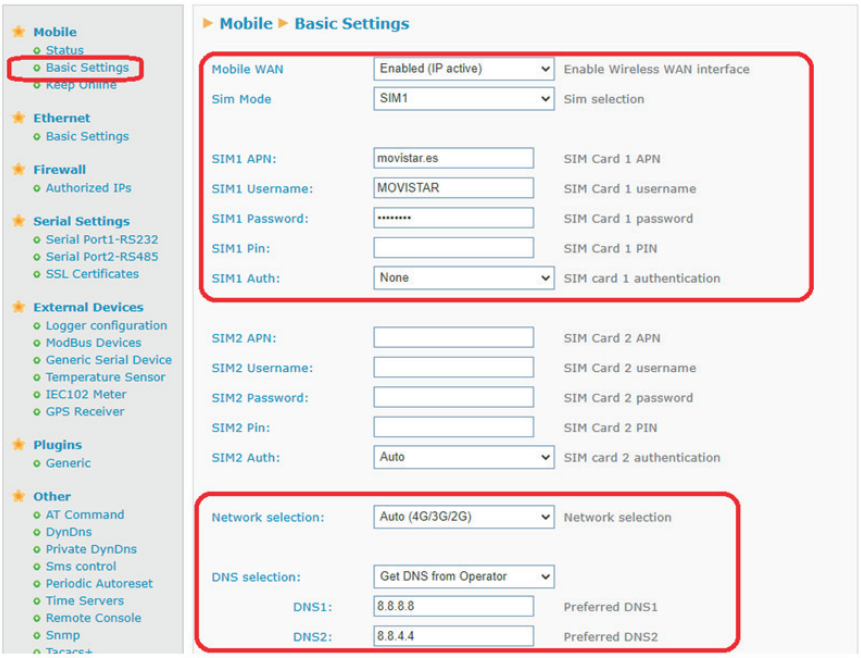
In this particular application note, we will assume that 2 registers are to be read from 2 PLCs with Modbus communications connected to a Webdyn - EasyTunnel via their RS485 port



The main aim is for the "Webdyn - EasyTunnel" device to read the Modbus registers with addresses 40000 and 40001 of PLC1, and registers 40000 and 40001 of PLC2 every minute. In both PLCs, register 40000 corresponds to the measured temperature, and register 40001 to the humidity level. The readings taken must be stored in the internal non-volatile memory of the "Webdyn - EasyTunnel" (in its datalogger), which must send the read data to the DEXMA platform whenever possible (coverage, IP connectivity, etc.). Communication with the PLCs is carried out via an RS485 bus with a 9600,8,N,1 configuration

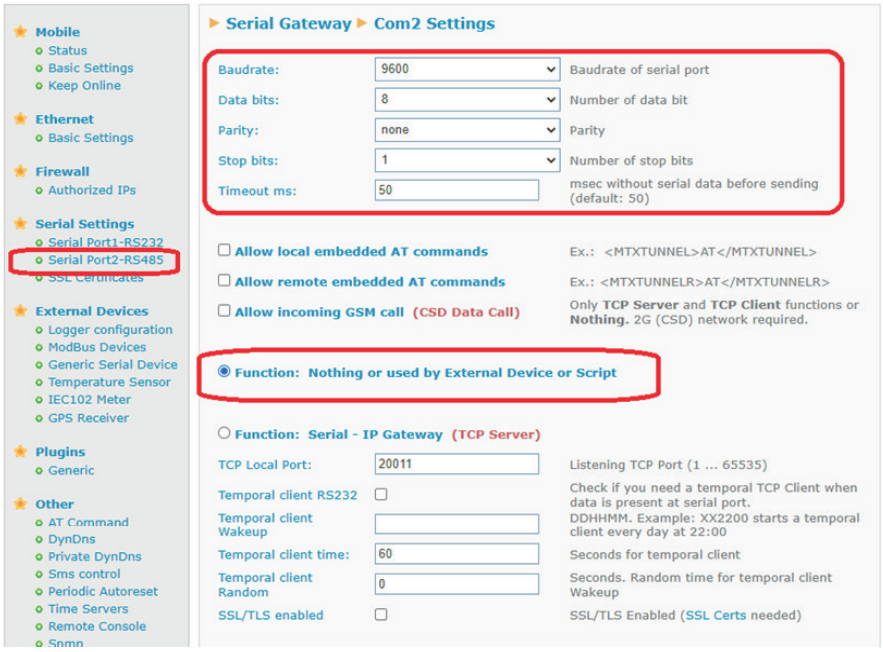
1. WAN mobile configuration.

The “Webdyn - EasyTunnel” must communicate with the ThingsBoard platform via 4G/3G/2G communications, so the "Mobile- Basic Settings" section must be configured correctly according to the SIM card used.



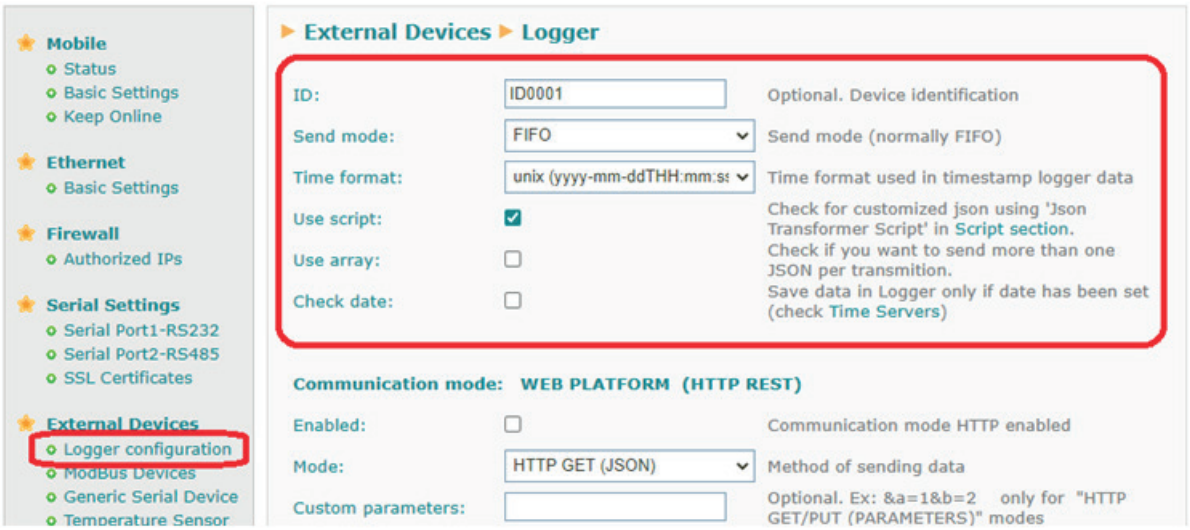
2. Configuring the RS485 serial port

The two PLCs will connect to 9600,8,N,1 via the RS485 serial port, so the "Serial Settings- Serial Port2-RS485" section must be configured by setting the parameters as shown below.

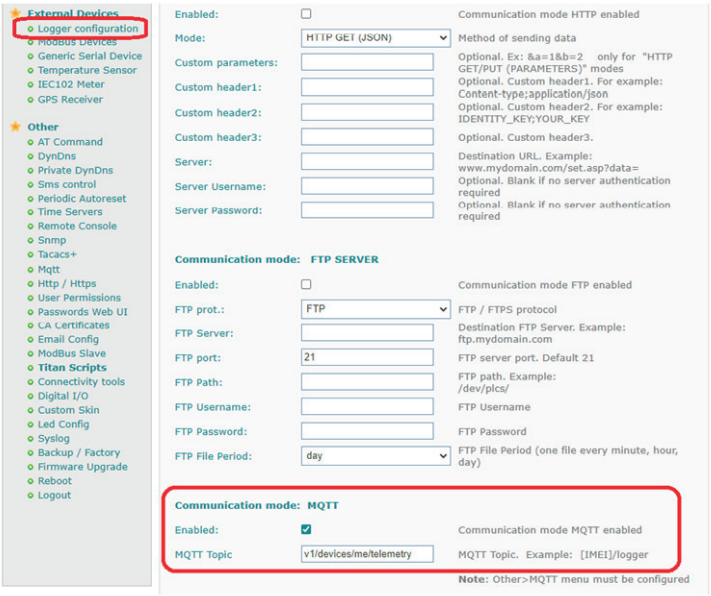


3. Logger configuration

The next step is to configure the internal datalogger of the device. Go to the “External Devices- Logger configuration” menu. The configuration should be similar to the one shown in the screenshot below:



You must select the MQTT communication mode at the bottom of the same display and enter v1/devices/me/telemetry in the telemetry delivery topic field.



The following parameters are particularly important:

• "**Time format"**: select the "unix" time format.

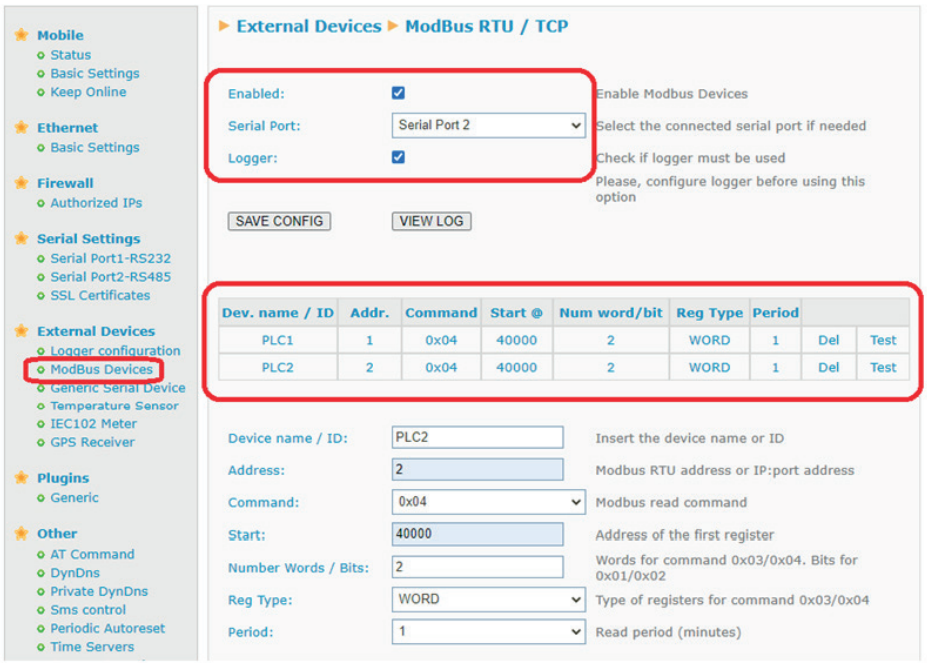
• "**Use script**": you must check this box, because activating it will run the script that you will integrate later. This script will also convert the standard JSON created by the device, with the data, into a JSON with the required format supported by the ThingsBoard platform.

• "**MQTT Enabled**": The logger must use the MQTT delivery method to send data to the ThingsBoard platform, so this box needs to be checked.

• **"MQTT Topic"**. Indicates the telemetry delivery topic – "v1/devices/me/telemetry" – the topic required by the ThingsBoard platform.

4. Configuring the Modbus section

In this configuration section ("External Devices- Modbus Devices"), you will configure the Modbus readings to be performed on the device on the 2 PLCs.



As can be seen in the previous screenshot, the Modbus service must be enabled by activating the "Enabled" checkbox. You must select the "Serial Port 2" serial port, since the reading will be made via the RS485 port. The "Logger" box must also be checked, since the Modbus registers read from the PLCs must be stored in the internal datalogger of the Titan-based device.

Two new devices must also be created, which we have called PLC1 and PLC2 in this example. You must indicate the Modbus address of each of the PLCs in the "address" field. PLC1 shall have Modbus address 1, and PLC2 shall have Modbus address 2. The Modbus command you will use in this example to read registers 40000 and 40001 will be 0x04, so you must select 0x04 in the "Command" field. The first register to be read in both PLCs is register 40000, so enter 40000 in the "Start" field. The aim is to read 2 Modbus registers from each PLC (40000 and 40001), so enter the value "2" in the "Number Words"

field. Select “WORD” for the register type (“Reg Type”) and 1 for the “Period”, because you will want to receive readings from the Modbus registers every minute.

5. Configuring the SCRIPT to convert the standard formatted JSON generated by the Titan-based device to the format we want to use to communicate with the ThingsBoard platform.

The standard JSON format generated by the Titan-based device, with the data read, and which, in its standard form (not converted), will be sent to a platform, will take the following format:

For PLC1:

{“IMEI”:”869101054286683”,”TYPE”:”MODB”,”TS”:”2022-11-11T12:17:00Z”,”ID”:”PLC1”,”A”:”1”,”ST”:”40000”,”N”:”2”,”V”:[225,62],”P”:”ID0001”}

For PLC2:

{“IMEI”:”869101054286683”,”TYPE”:”MODB”,”TS”:”2022-11-11T12:17:01Z”,”ID”:”PLC2”,”A”:”2”,”ST”:”40000”,”N”:”2”,”V”:[241,71],”P”:”ID0001”}

But to make things easier, you will ideally want to send the data to the ThingsBoard platform as follows:

For PLC1:

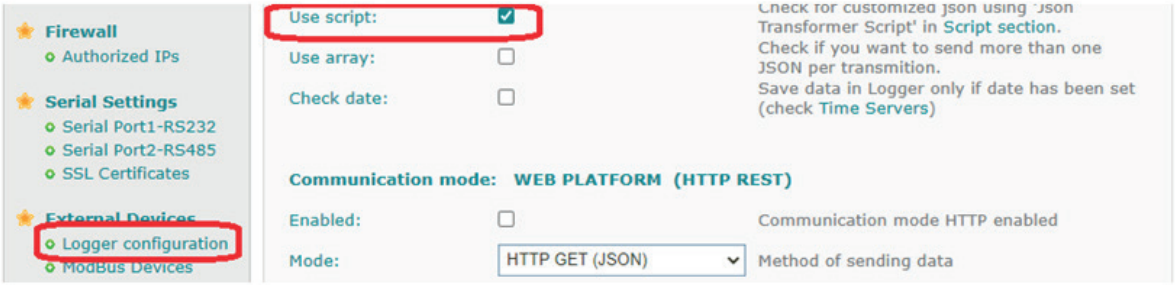
{“TS1”:”2022-11-11T12:17:00Z”, “TEMP1”:22.5,”HUM1”:62}

For PLC2:

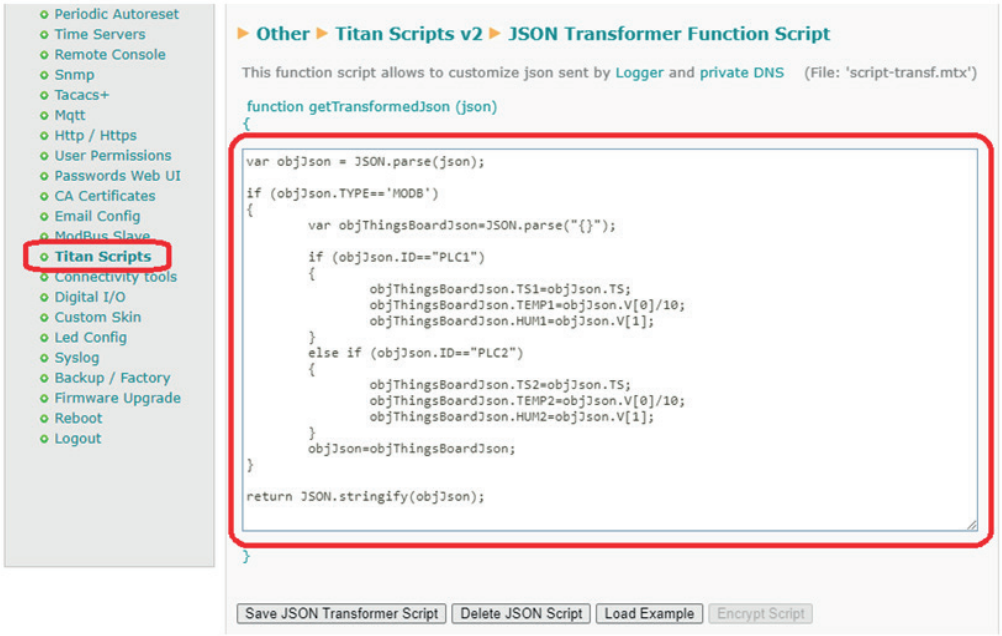
{“TS2”:”2022-11-11T12:17:00Z”, “TEMP2”:24.1,”HUM2”:71}

In other words, the aim is to assign the temperature of each PLC (further dividing each value by 10 to obtain the temperature in degrees) to the JSON fields "TEMP1" and "TEMP2". The fields "HUM1" and "HUM2" shall be assigned the humidity values obtained from the Modbus reading.

You must therefore create a short SCRIPT to convert the data from the standard format to that required for sending data to ThingsBoard. Remember that this conversion script will be run because you previously checked the “Use script” box in the “External Devices- Logger Configuration” section.



To enter the code for the data conversion, go to the “Other- Titan Scripts” menu. The conversion script must be entered in the “JSON Transformer Function Script” section.



The SCRIPT will have the following code.

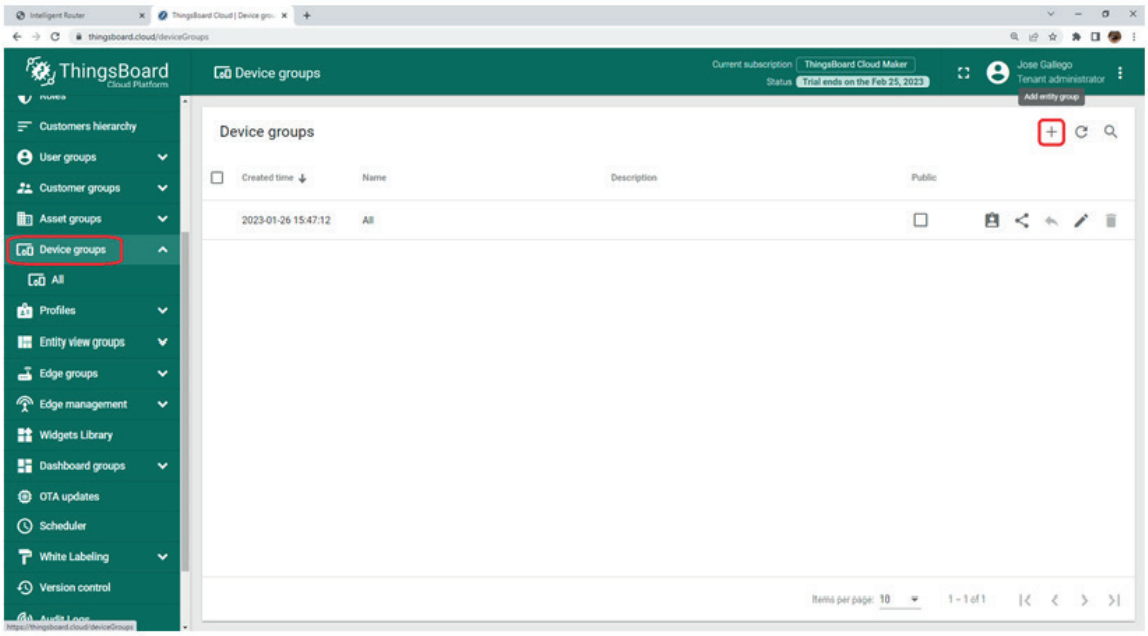
```
//Create a JSON-type object variable, because the "json" variable that receives the
//function as an argument is a "String"-type variable.
var objJson = JSON.parse(json);

//If it is indicated in the original JSON received that it is a Modbus data JSON ...
if (objJson.TYPE==’MODB’)
{
    //Create a new JSON object where you will store the converted data
    //transformados
```

```
	var objDexmaJson=JSON.parse("{}");

	//If the data corresponds to PLC1 readings ....
	if (objJson.ID=="PLC1")
	{
		//Add the date of the data.
		objThingsBoardJson.TS1=objJson.TS;

		// Add the temperature reading, divided by 10, in the TEMP1 field.
		objThingsBoardJson.TEMP1=objJson.V[0]/10;

		// Add the humidity reading in the HUM1 field.
		objThingsBoardJson.HUM1=objJson.V[1];
	}
	//If the data corresponds to PLC2 readings ....
	if (objJson.ID=="PLC2")
	{
		//Add the date of the data.
		objThingsBoardJson.TS2=objJson.TS;

		// Add the temperature reading, divided by 10, in the TEMP2 field.
		objThingsBoardJson.TEMP2=objJson.V[0]/10;

		// Add the humidity reading in the HUM2 field.
		objThingsBoardJson.HUM2=objJson.V[1];
	}

	//Then, copy the converted JSON to return it as a result
	// of the conversion function.
	objJson=objThingsBoardJson;
}
```

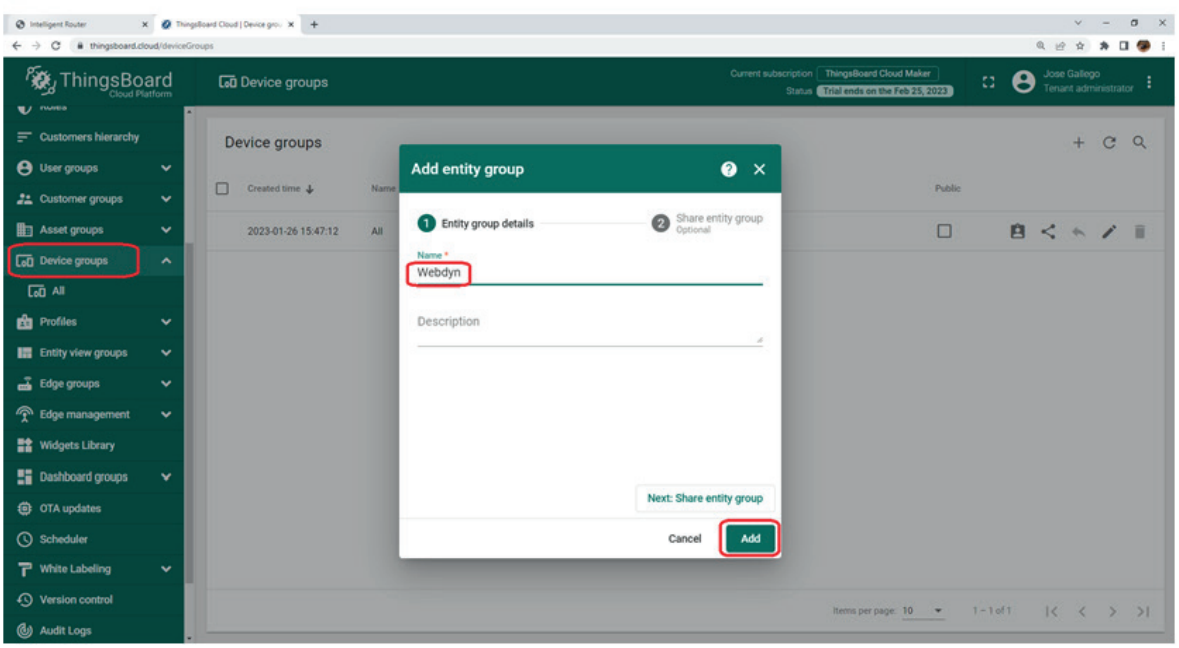
6. Configuring the MQTT section and ThingsBoard platform

You must also correctly configure the "Other - Mqtt" section of the Titan-based device so that it can connect to the ThingsBoard platform. But first, you must go to the web platform to add the “Webdyn - EasyTunnel” device and obtain the mqtt authentication attributes.

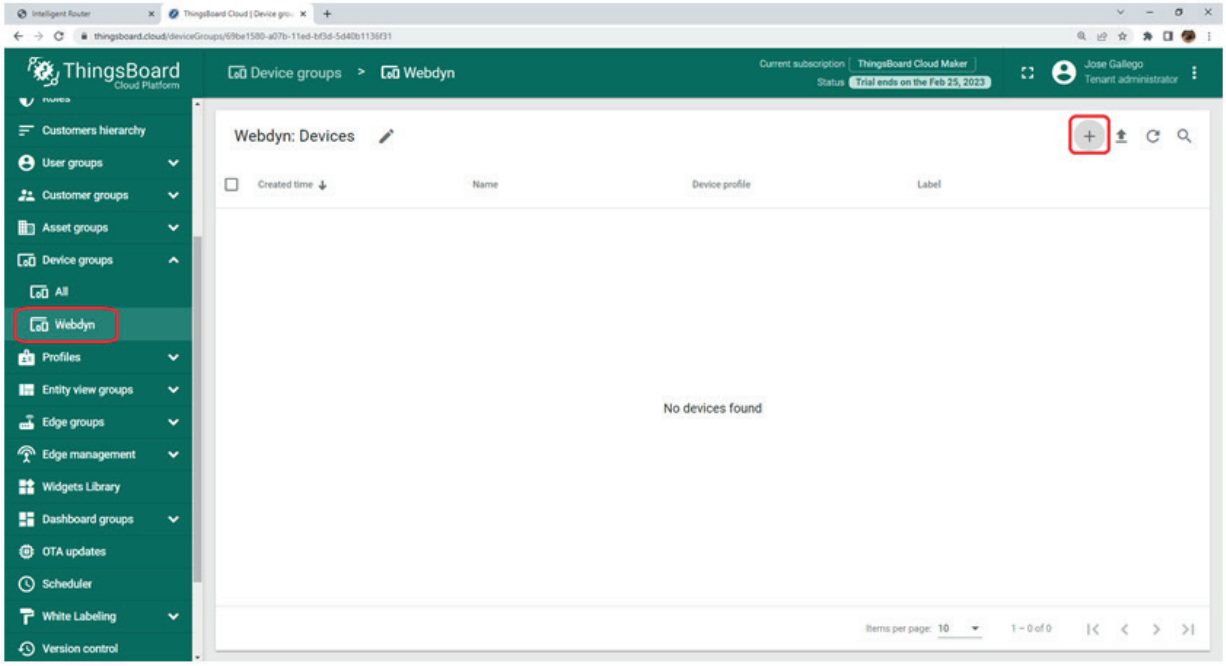
Once you have accessed your http://thingsboard.cloud account, go to the "Device groups" section. Click on the "+" icon to create a new group.



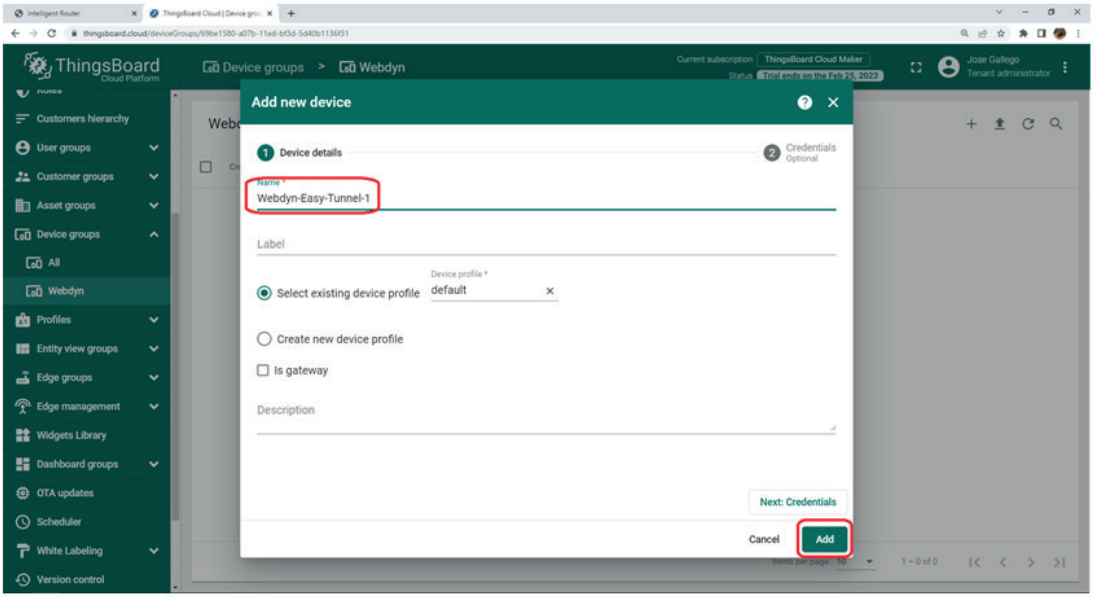
For example, you can enter the group name "Webdyn". Then, click on the “Add” button.



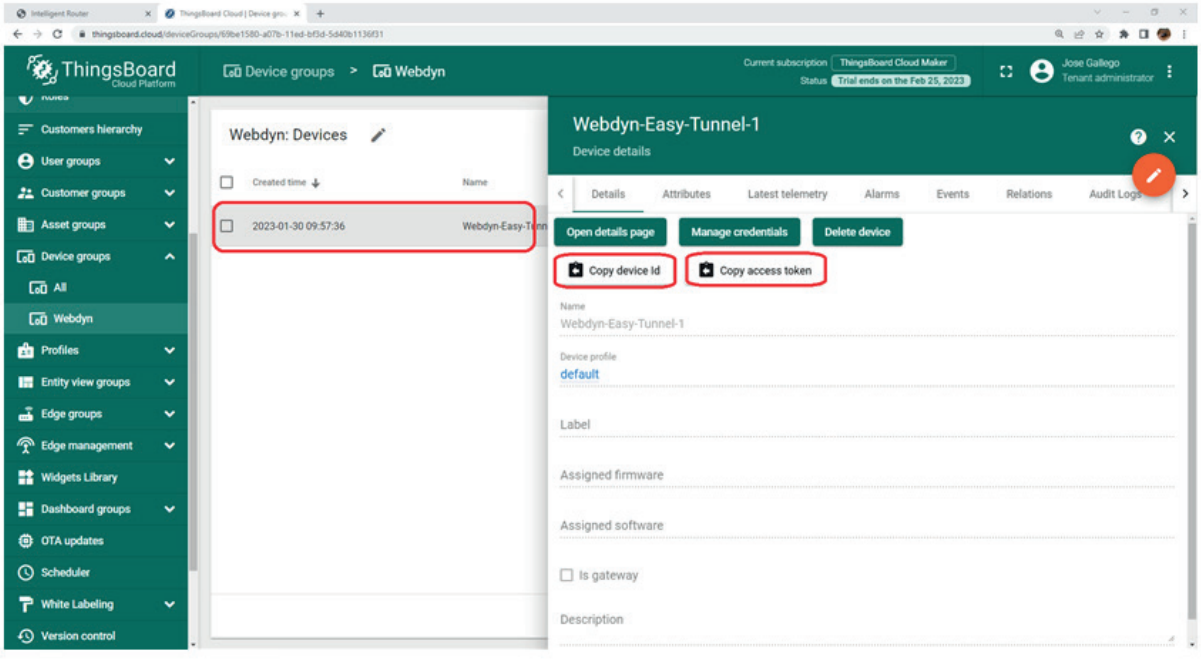
Once the "Webdyn" group has been created, enter the group created (by clicking on the line corresponding to that group) and then create a new device by clicking again on the "+" icon.



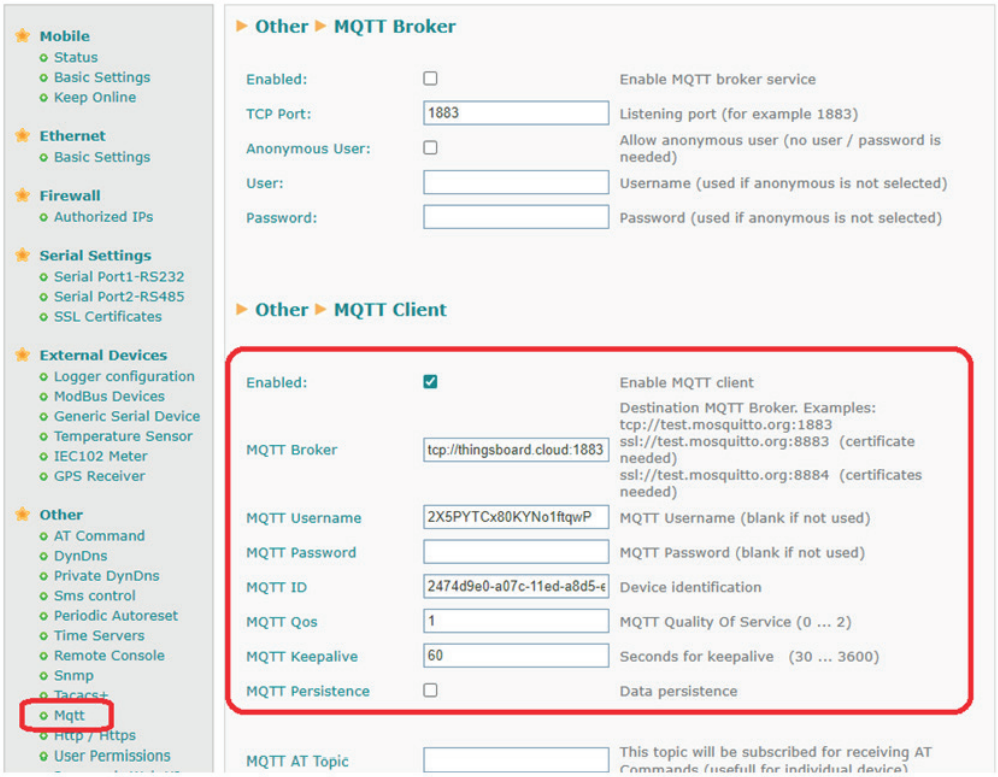
You can enter "Webdyn-Easy-Tunnel-1" as the device name. Then, you must click on the "Add" button.



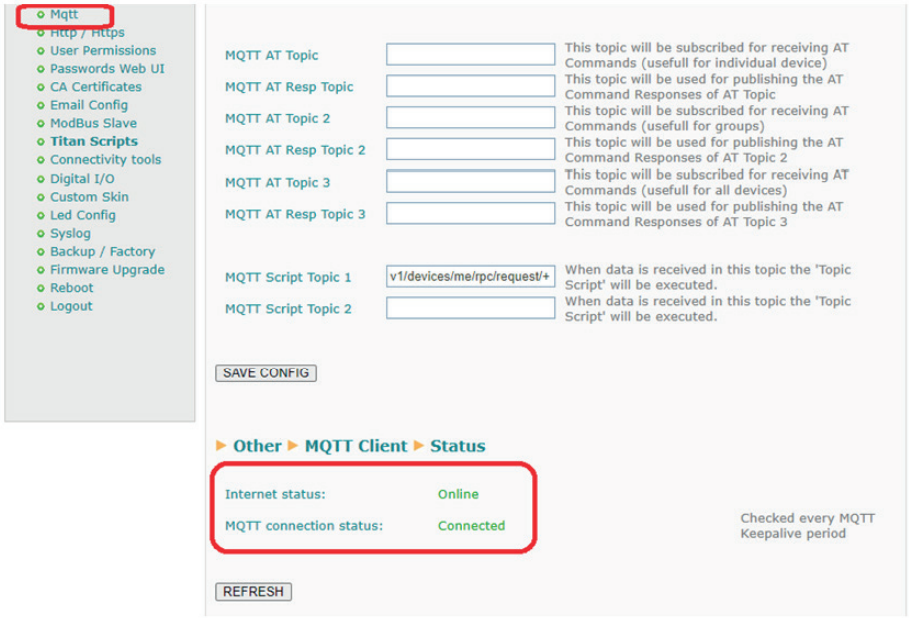
Next, click on the line corresponding to the newly created device and click on the "Copy device ID" and "Copy access token" buttons. You will need to note down these values, as they will need to be integrated into the "Other - MQTT" section of the Titan-based device.



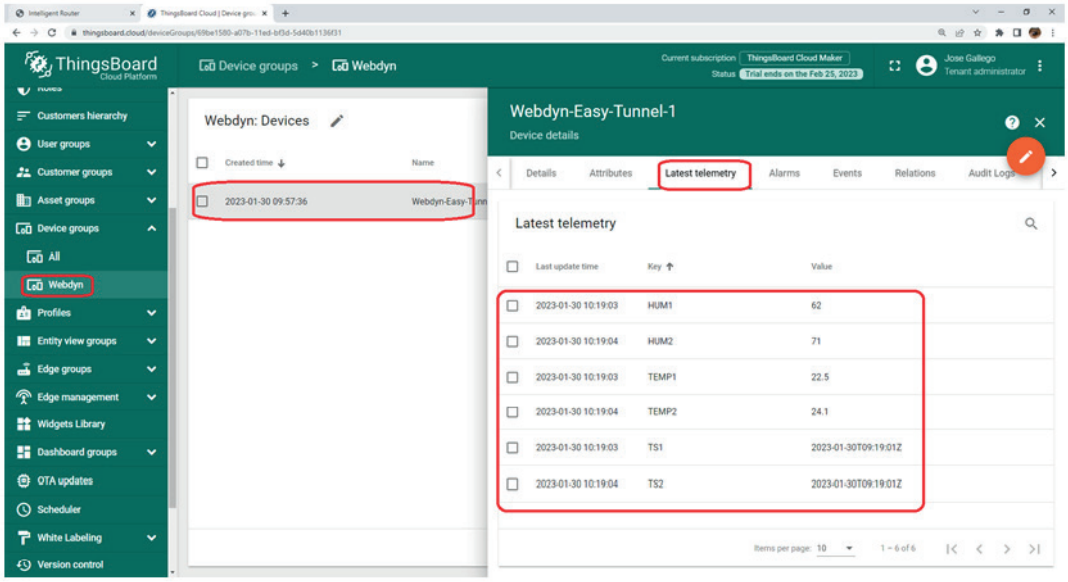
Now, in the "Other-Mqtt" section of the Titan-based device, you must enable the "MQTT Client" service so that it can connect to the ThingsBoard platform. Enter the value “tcp://thingsboard.cloud:1883” in the "MQTT Broker" field, the “access token” in the "Mqtt Username" field and the “device ID” in the "MQTT ID" field, with all these fields copied from the previous step. Then, press the "SAVE Config" button and restart the Titan-based device to apply the new configuration.



After restarting the Titan-based device and waiting a few seconds for the IP connection, as long as everything is in order, the "Other - MQTT" section will display the status of the mqtt connection as follows:

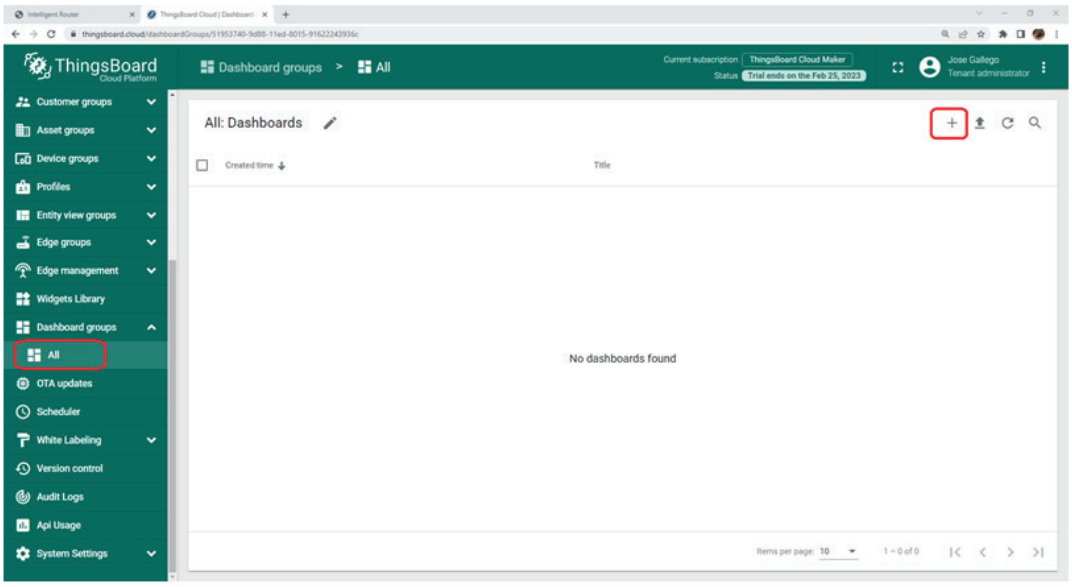


At this point, you cannot tell whether the ThingsBoard platform is receiving the data from the Modbus registers corresponding to the temperature and humidity readings from both PLCs via Modbus. To check this, click again on the line corresponding to the device, then click on the "Latest telemetry" section. In this particular section of the ThingsBoard platform, if the data sent by the Titan-based device is being received correctly, it should appear as shown in the screenshot below:

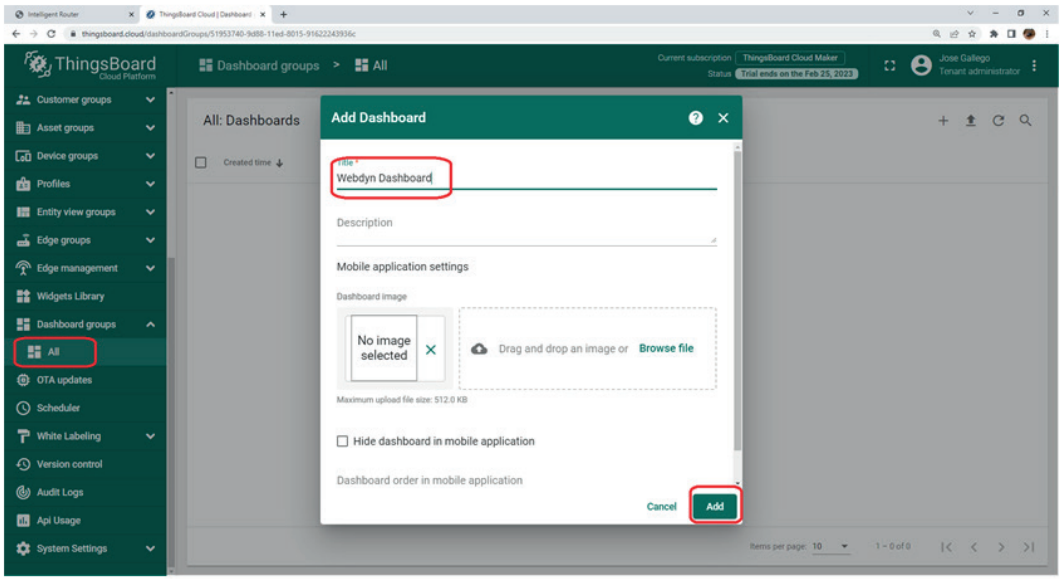


7. Building a Dashboard in ThingsBoard to visualise temperature and humidity data.

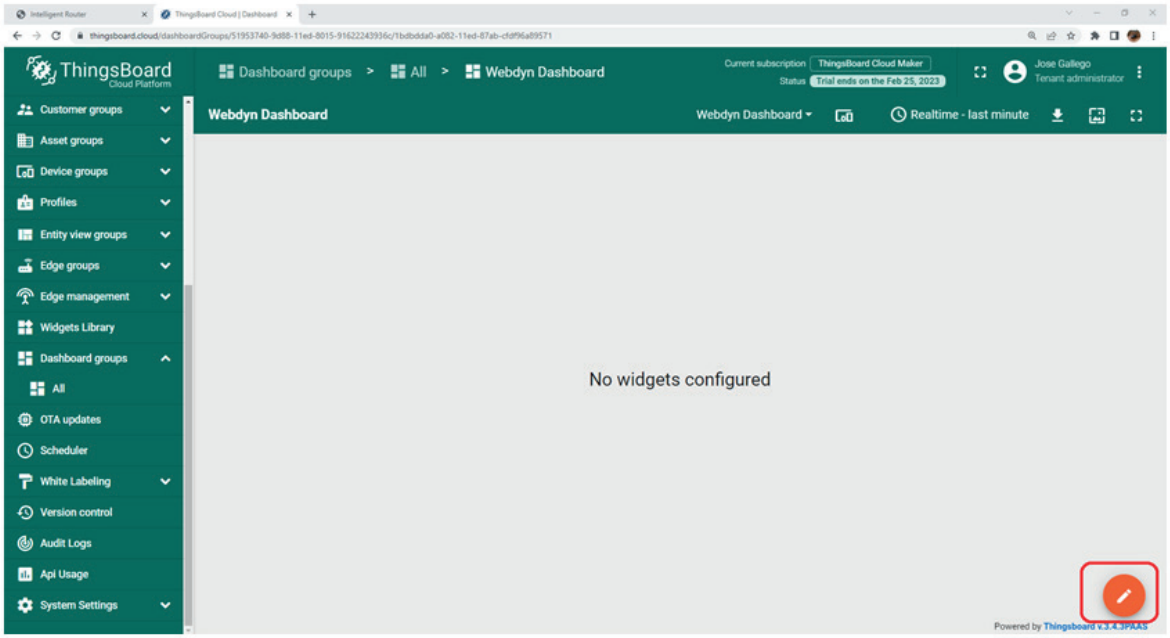
To create a dashboard to graphically represent the data, click on the "DashBoard groups - All" section, and then on the “+” icon, to create a new Dashboard.



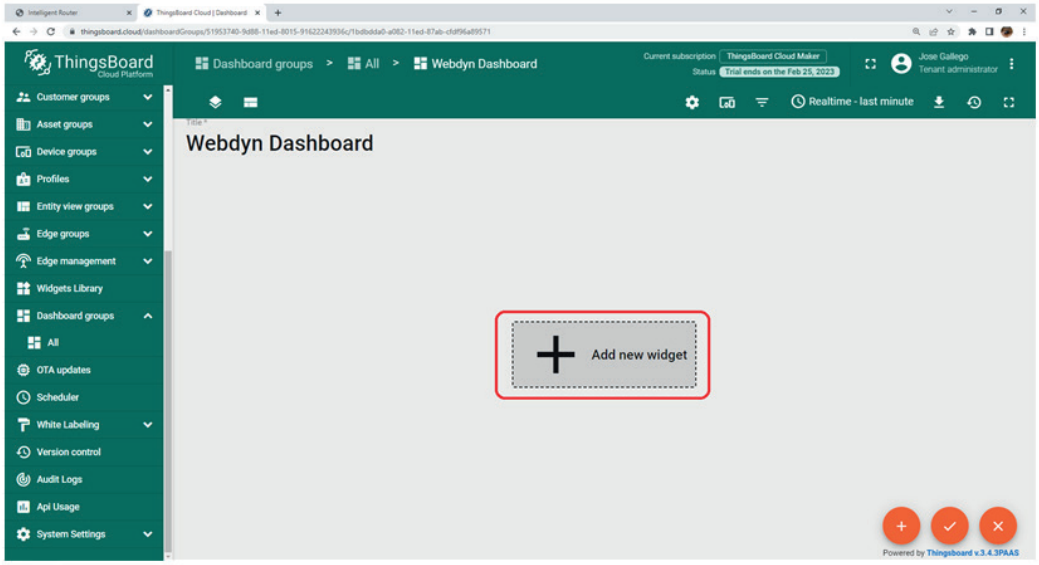
You can enter "Webdyn Dashboard" as the name of the dashboard and click on the "Add" button.



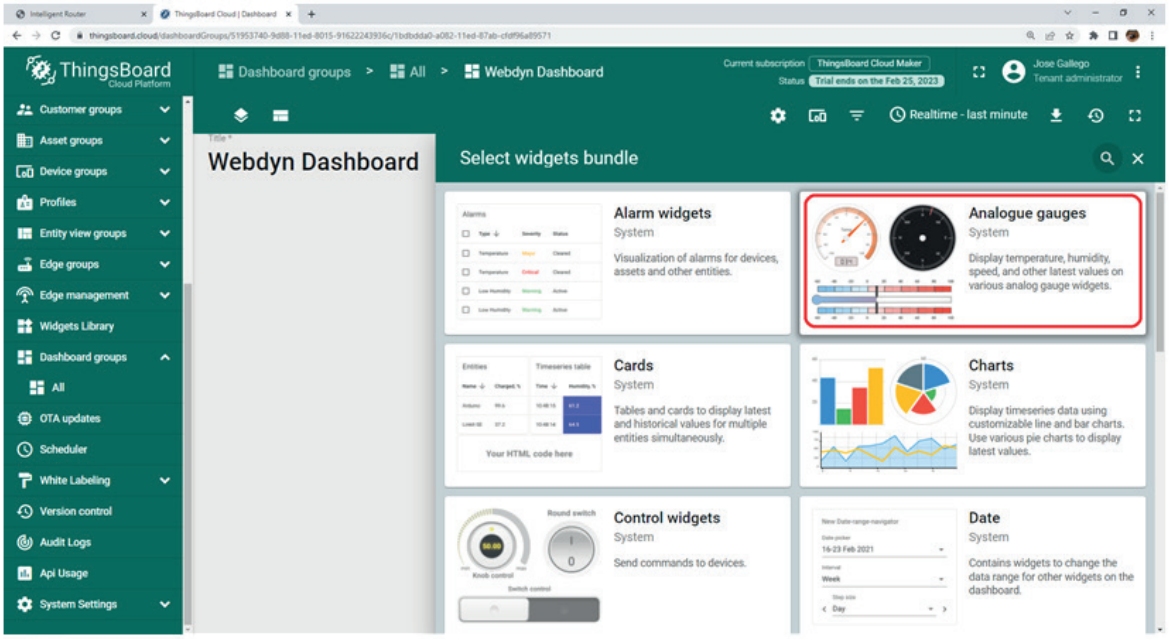
Next, click on the line corresponding to the newly created dashboard, then on the edit icon at the bottom right.



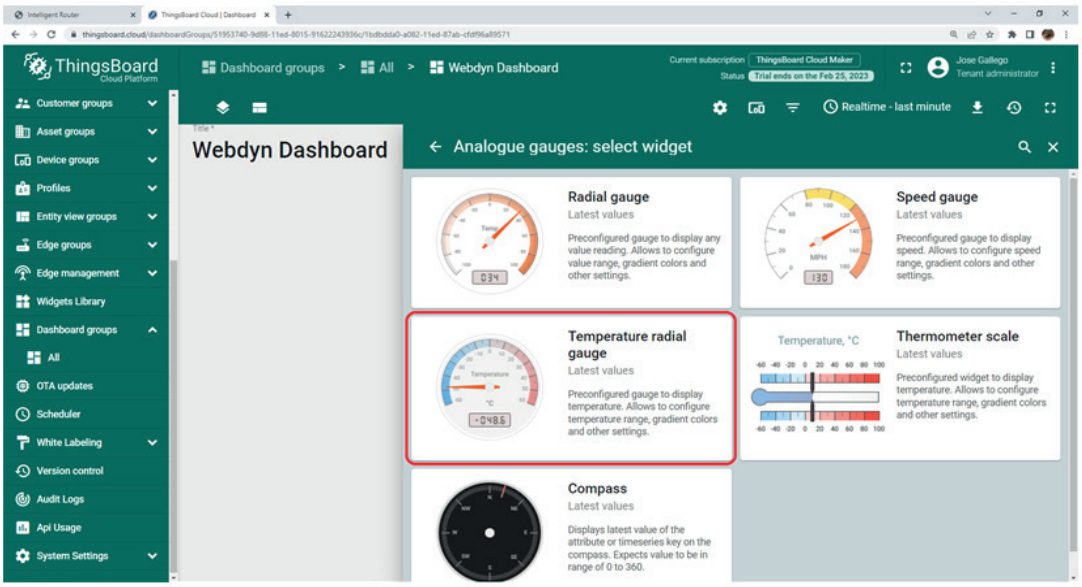
Click again on the "Add new widget" button to add a new widget to the dashboard.



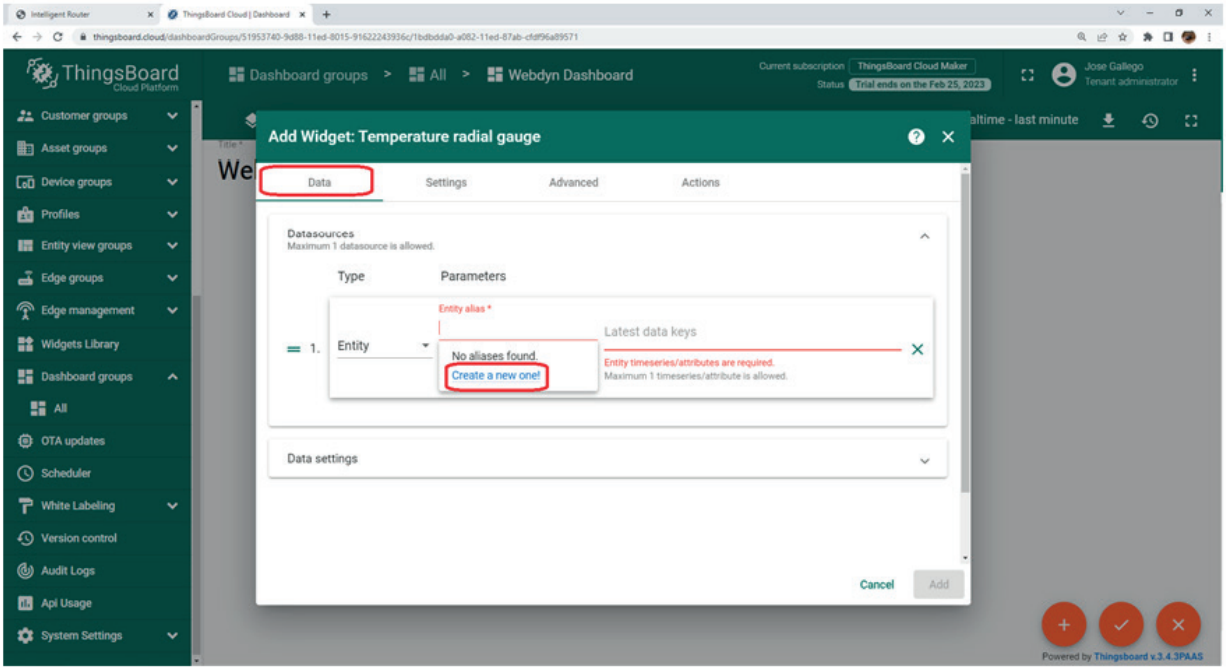
Select "Analogue gauges" as the gauge type.



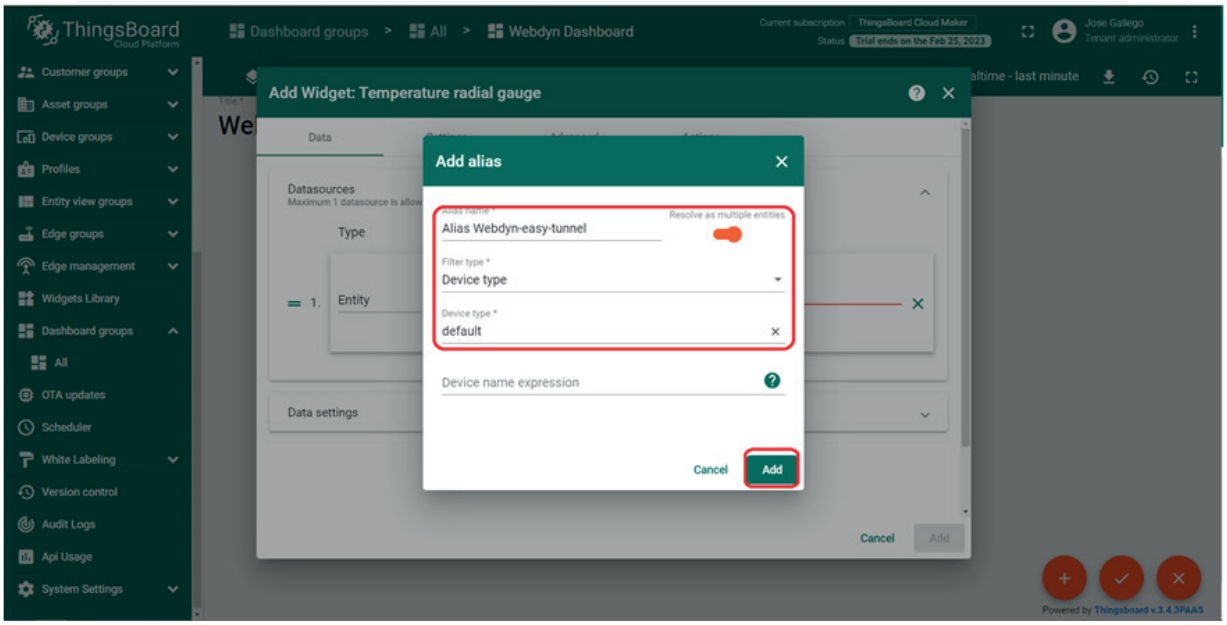
A "Temperature Radial gauge" will then be added.



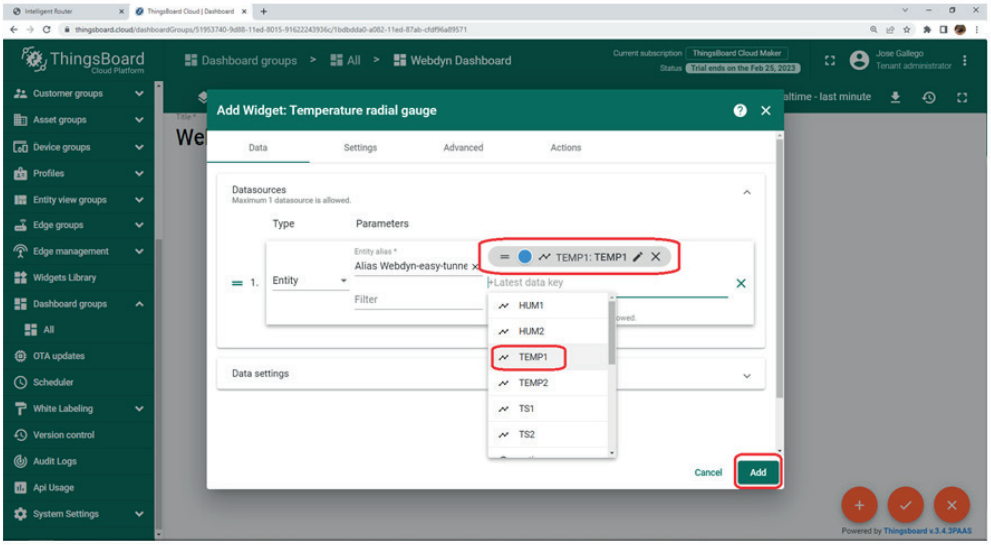
Once added, click on the "Create a new one" link in the "Temperature radial gauge" properties area of the "Data" section to create a new Alias referring to the Titan-based “Webdyn - EasyTunnel” device.



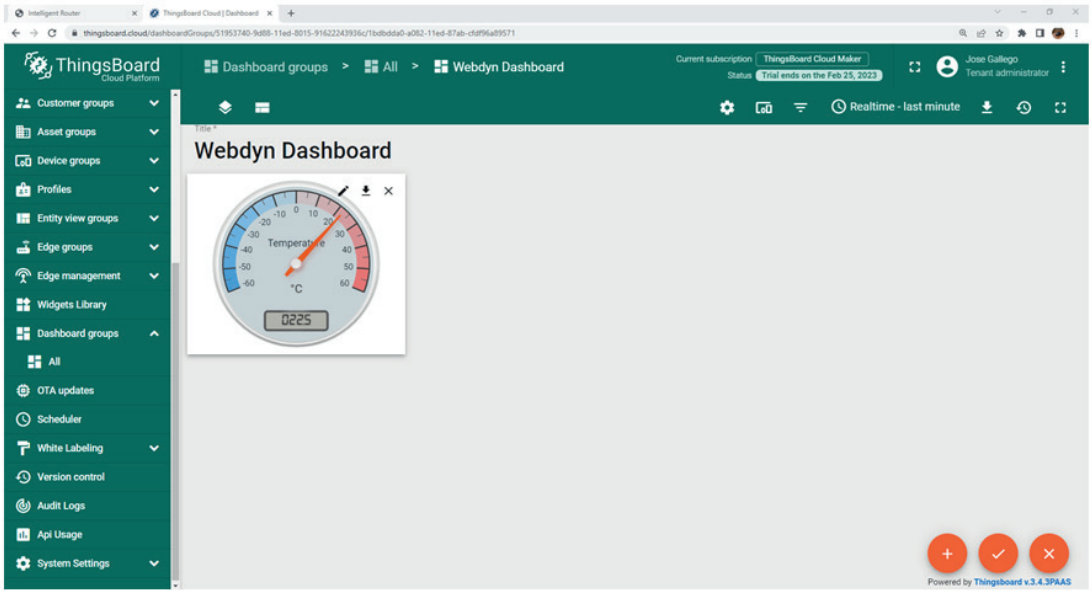
Create the Alias to be added to the device and click the "Add" button.



After creating the Alias, you must select the "TEMP1" field from the drop-down menu next to it. The drop-down fields will be displayed automatically if the connection to the Titan-based device is established and the modem has already sent data. You must then click on the "Add" button to finish.



At this point, the widget will be added to the dashboard and should display the current temperature value read by PLC1: 22.5°C in the example scenario.

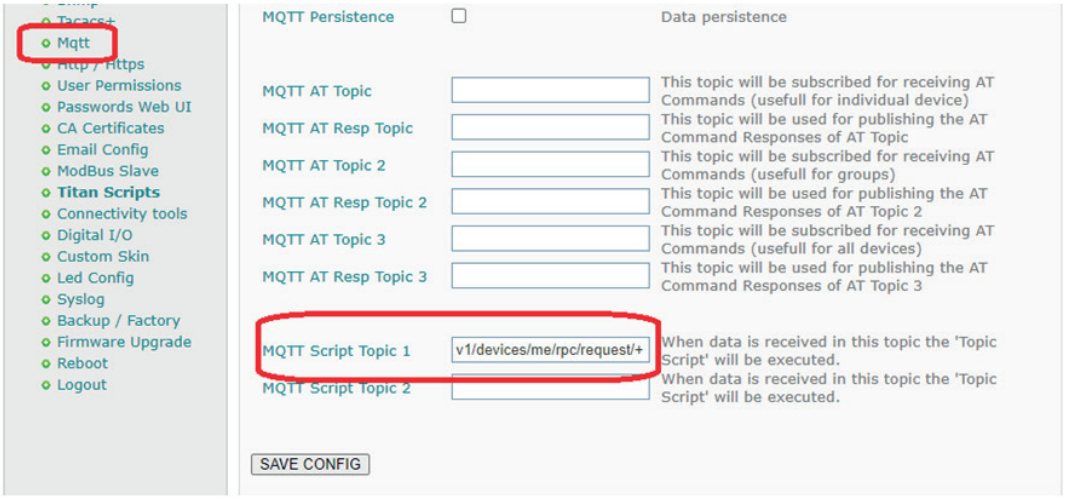


To add other widgets containing the temperature value of PLC2 and the humidity values of PLC1 and PLC2, simply repeat the latest steps, selecting the corresponding TEMP2, HUM1 and HUM2 fields for each.

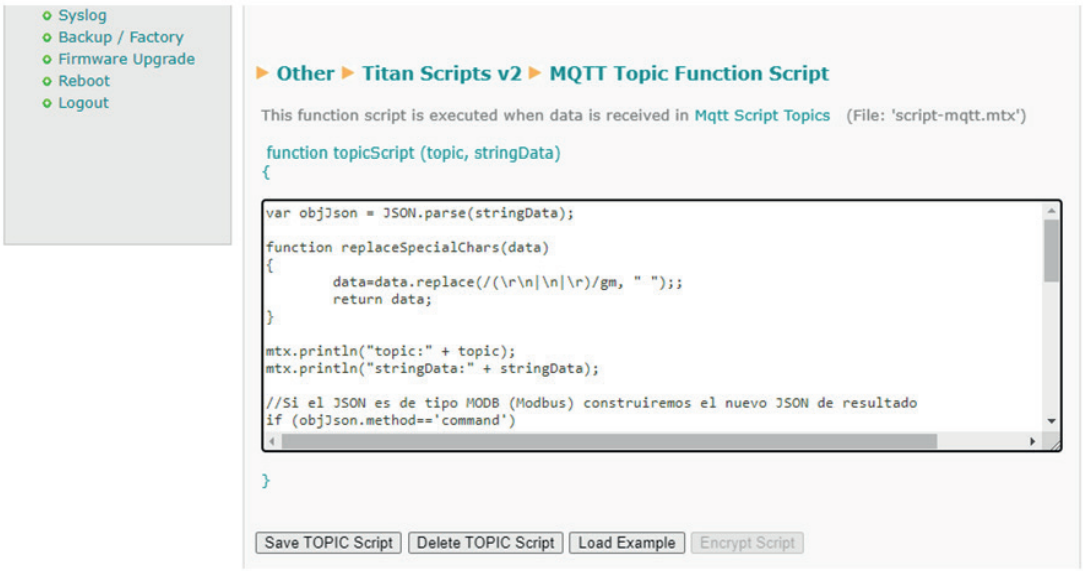
8. Direct communication between the ThingsBoard platform and the Titan-based "Webdyn-Easy-Tunnel" device for status reading and remote configuration changes.

To send commands from the ThingsBoard platform to the Titan-based device, you must first add the appropriate MQTT topics. To do so, add the "v1/devices/me/rpc/request/+" topic in the "MQTT Script Topic 1" field of the "Other ☐ Mqtt" section. This will internally forward any data received by the Titan-based device in the "v1/devices/me/rpc/request/xxxxx" topic to the "MQTT Topic Function Script". Please note that this functionality (the ability to add the "+" character in the topic to subscribe to any topic starting with v1/devices/me/rpc/request/xxxxx, where xxxxx can be anything) is supported in firmware version 6.17 and later versions. After the configuration change, you must restart the Titan-based device.

This is required, because the ThingsBoard platform will send commands intended for the Titan-based device to the "v1/devices/me/rpc/request/xxxxx" topic, where xxxxx will represent a number that will increase with each command sent. The Titan-based device shall send the response to the "v1/devices/me/rpc/response/xxxxx topic.



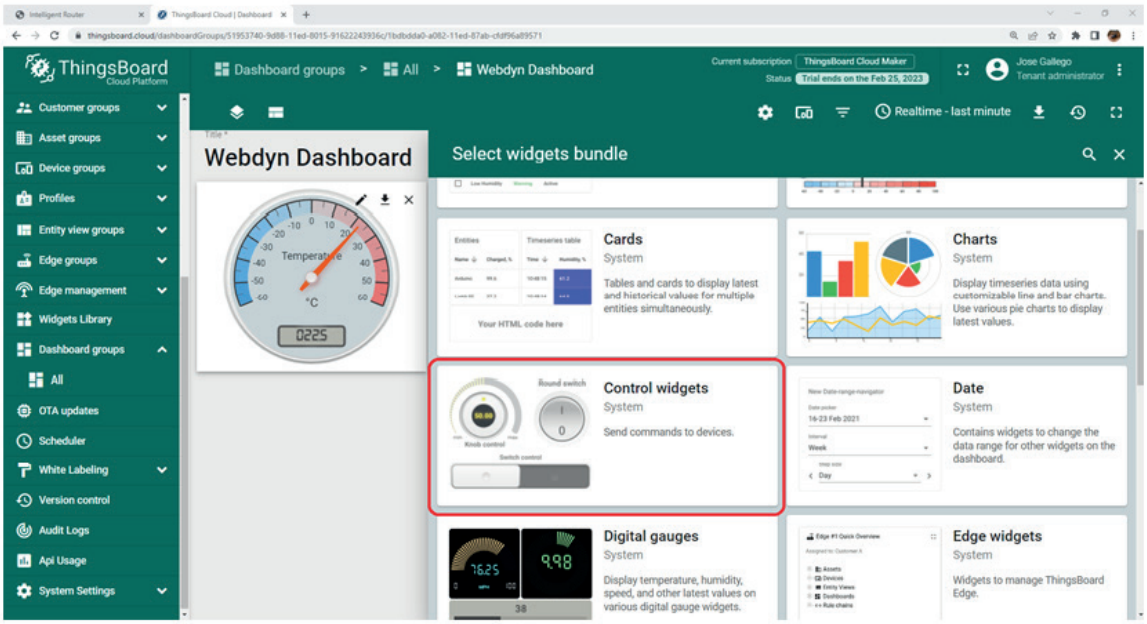
Now, add the following script in the "Other - MQTT" section:



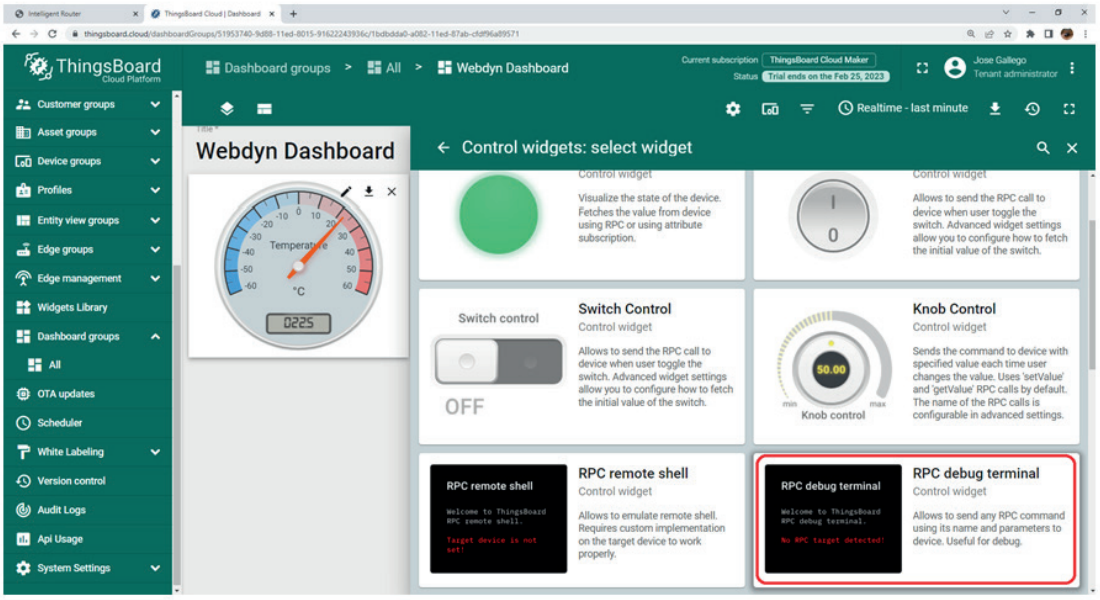
Here is the complete and fully explained script code.

```
//Create a JSON object with the data received from ThingsBoard
var objJson = JSON.parse(stringData);

        //Function to be used to replace special characters such as ENTER, etc. with blank spaces.
function replaceSpecialChars(data)
{
                data=data.replace(/(/(\r\n|\n|\r)/gm, " ");;
                return data;
}

        //The topic and data received by the platform are printed through the debug output
mtx.println("topic:" + topic);
mtx.println("stringData:" + stringData);

//If the ID "command" is present in the "method" field of the json received from the platform
if (objJson.method=='command')
{
                //If there is a command to execute in the "params" field of the received JSON
                if (objJson.params!=null)
                {
                    //Execute AT command
                    var res=mtx.atSend(objJson.params,2000);
                    //Create the response topic, replacing "request" (text) with "response"
                    var topicResponse=topic.replace("request", "response");
                    //Send a JSON with the response to the executed command
                    var answer="{"method":command",{"params":command"] + replaceSpecialChars(res)
                    var r=mtx.mqttSend(answer,topicResponse,1);
                }
}
```

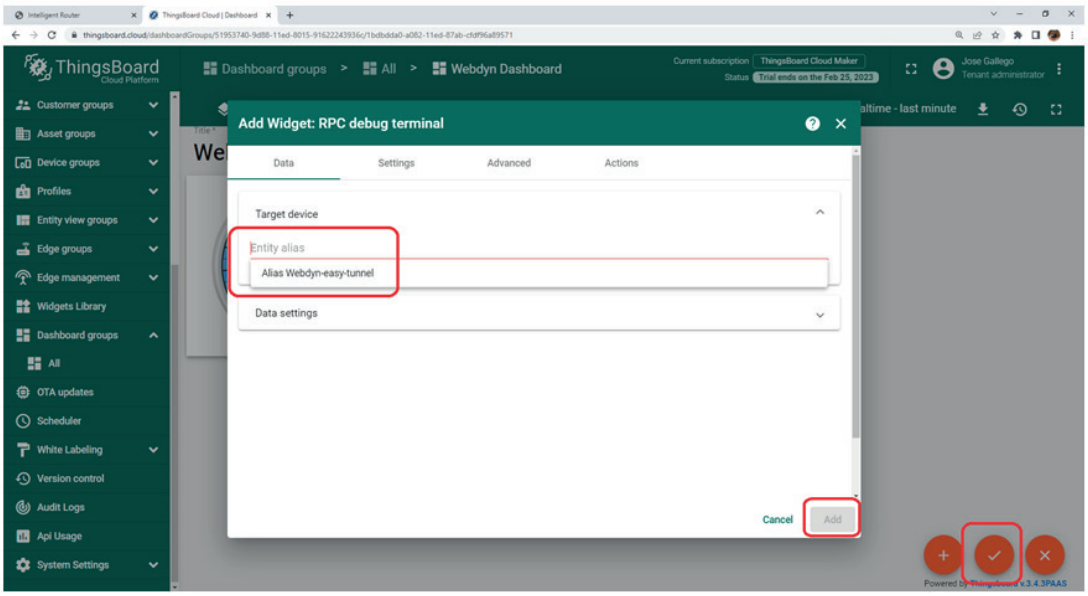
Another widget must be added to the ThingsBoard platform, in this case, by clicking on the "Control widgets" type.



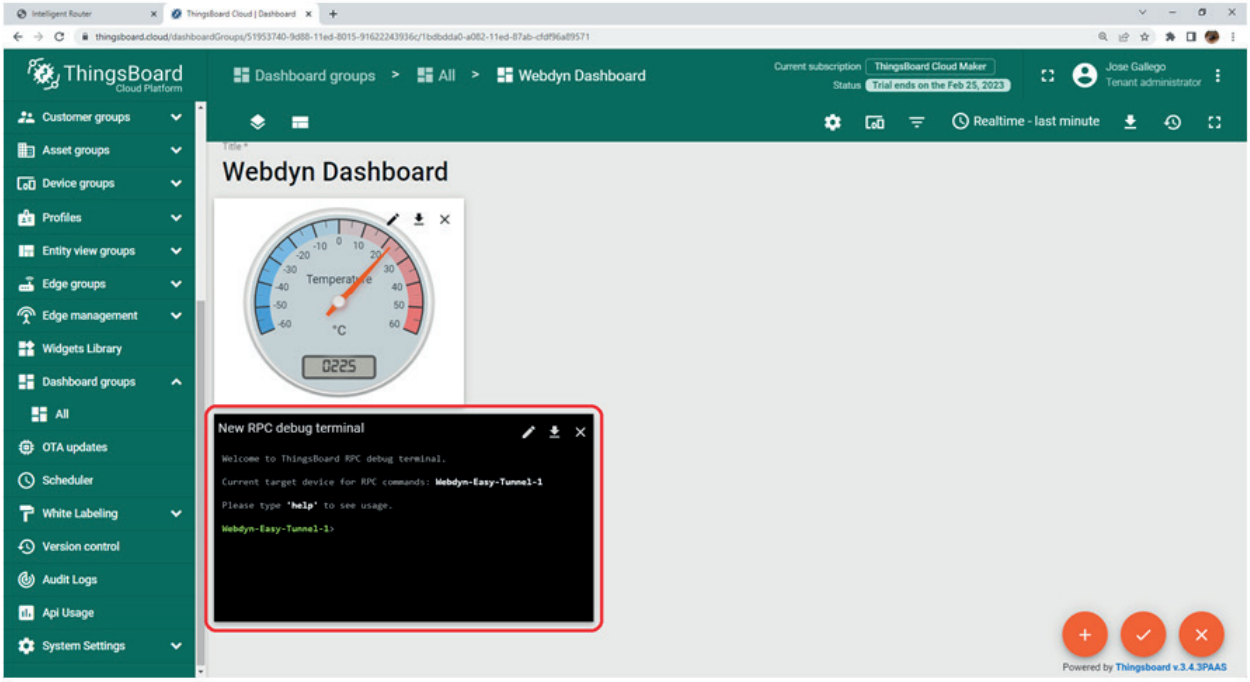
And, in the "Control widgets" section, you must select the "RPC debug Terminal" widgets type:



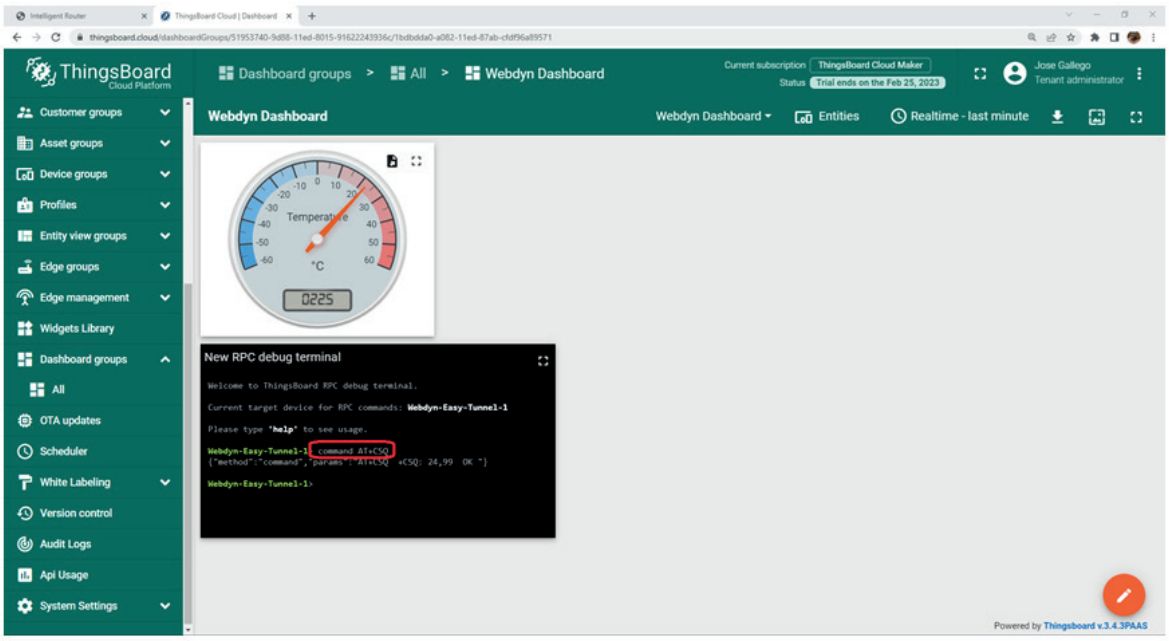
In the "Entity Alias" field, you must select the previously created alias, "Alias Webdyn-Easy-Tunnel", then click on the "Add" button > "V" icon.



At this point, the widget that allows you to send AT commands to the Titan-based device remotely will appear.



For example, to send any command, you simply need to type the text "command", followed by a blank space, and enter the command to be executed. For example, if you wanted to execute the AT+CSQ command to find out about coverage remotely, you would simply need to follow the steps in the following screenshot:



Any questions?

Please direct your enquiries to iotsupport@mtxm2m.com