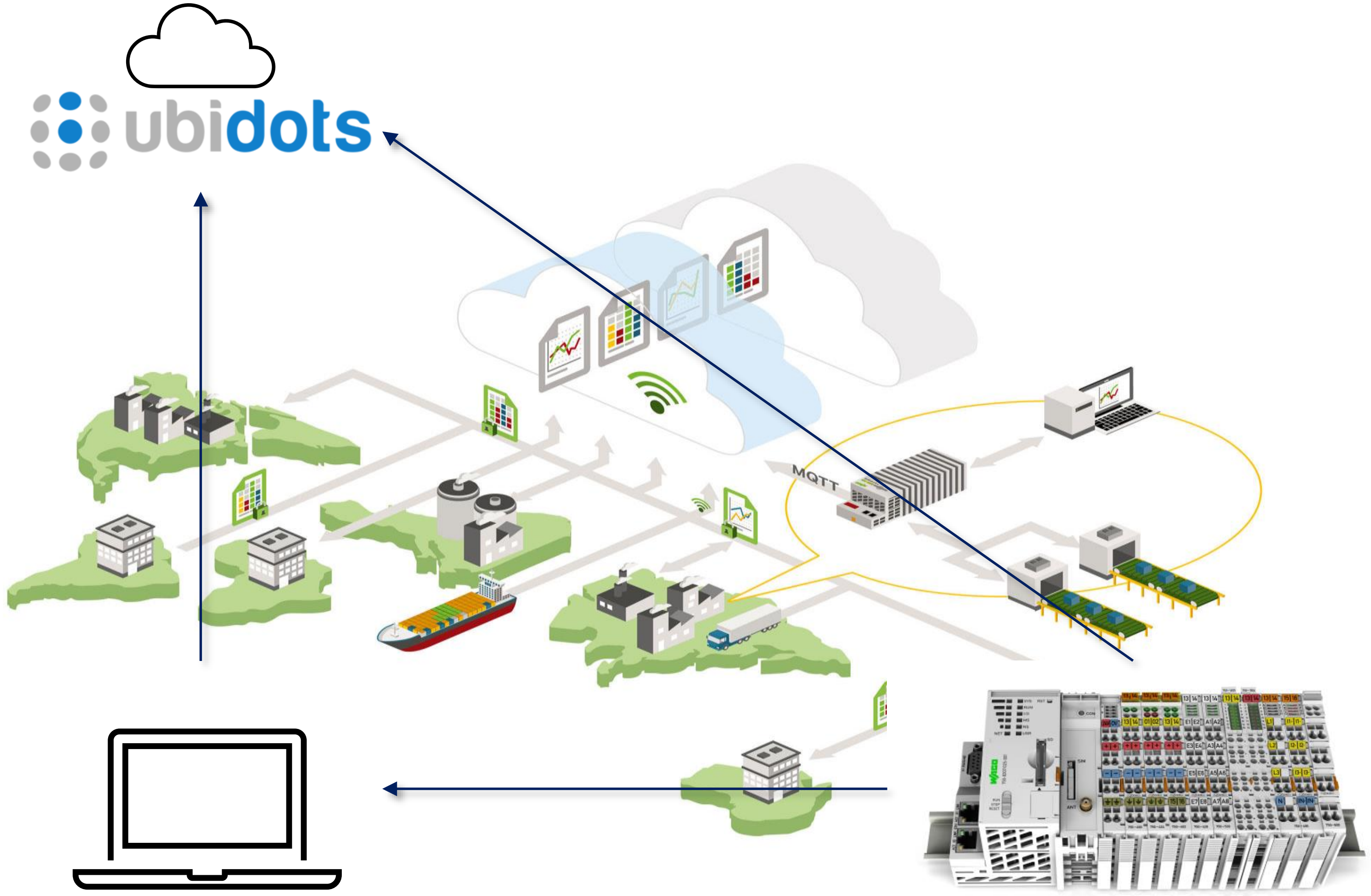
Connecting the Wago PFC200 to Ubidots Via Codesys

Table of Contents:



Connecting the Wago PFC200 to Ubidots Via Codesys

Ubidots Setup

1a) Create and Login to your Ubidots Account

(https://ubidots.com/community/)

1b)

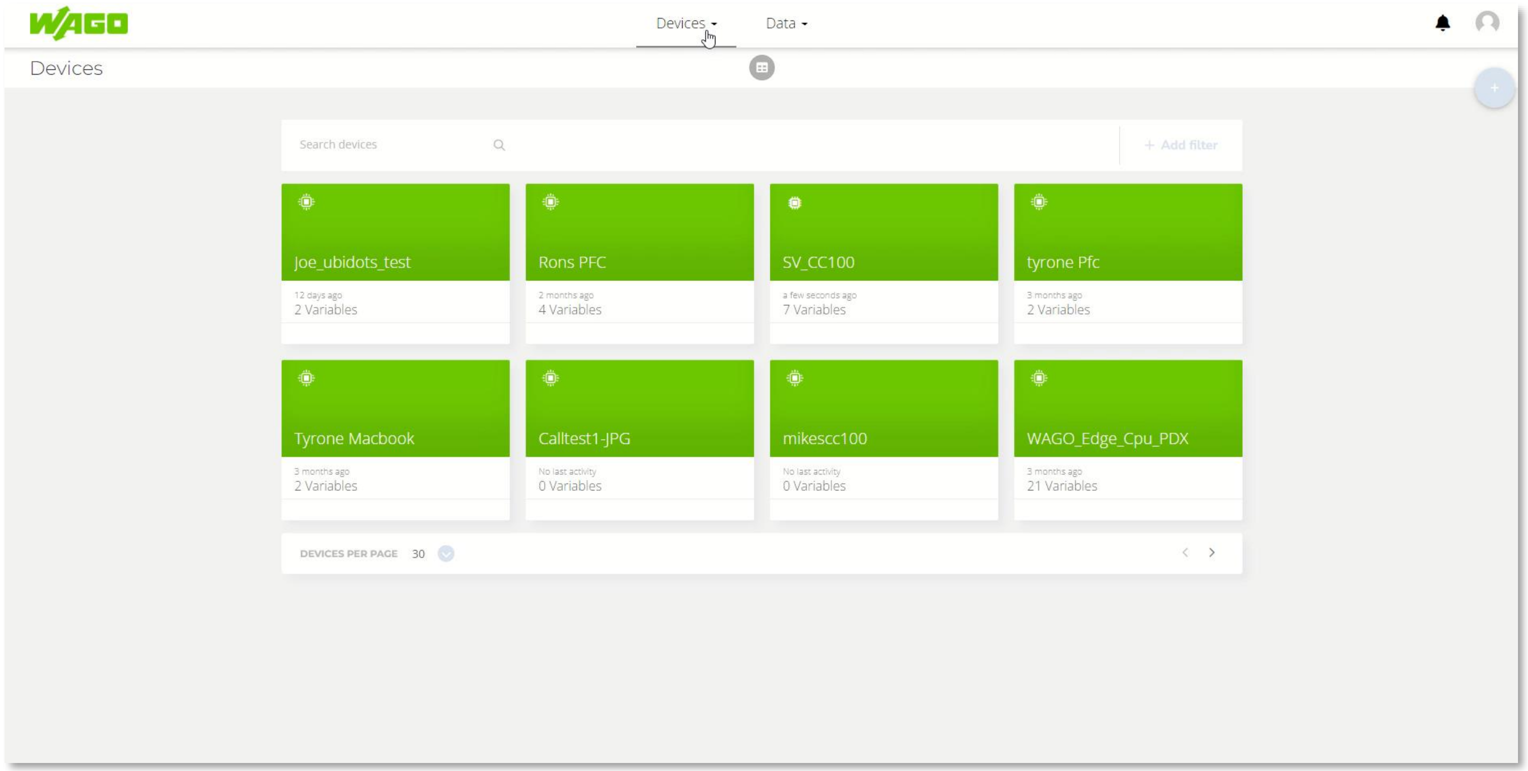
Go to the 'Devices' tab on the top of the page, then click 'devices' in the drop-down.

Click the '+' symbol on the top right corner of the screen.

Select 'Blank device'

Create a device name and then you should see the new device pop up.

Video: Add device in Ubidots



Device	Last activity	Variables
Joe_ubidots_test	12 days ago	2 Variables
Rons PFC	2 months ago	4 Variables
SV_CC100	a few seconds ago	7 Variables
tyrone Pfc	3 months ago	2 Variables
Tyrone Macbook	3 months ago	2 Variables
Calltest1-JPG	No last activity	0 Variables
mikescc100	No last activity	0 Variables
WAGO_Edge_Cpu_PDX	3 months ago	21 Variables

DEVICES PER PAGE 30

Connecting the Wago PFC200 to Ubidots Via Codesys

Ubidots Setup

1c)

Click on the newly created device and record the API Label, ID, and Token of the device.

You'll need it for later.

1d)

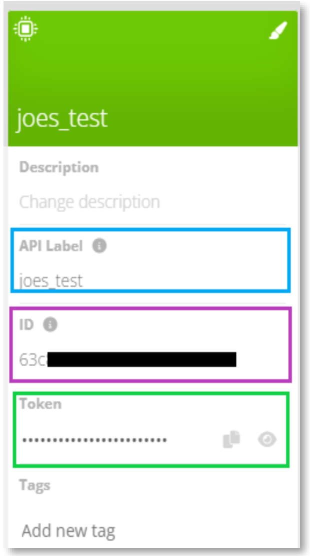
You'll need the Ubidots PEM certificate for TLS encryption to secure messages going to the cloud.

Found here:

https://docs.ubidots.com/v1.6/reference/broker-urls

Click on the link PEM cert download link then save the file as 'roots.crt'

Ubidots device screen



joes_test

Description

Change description

API Label

joes_test

ID

63c[redacted]

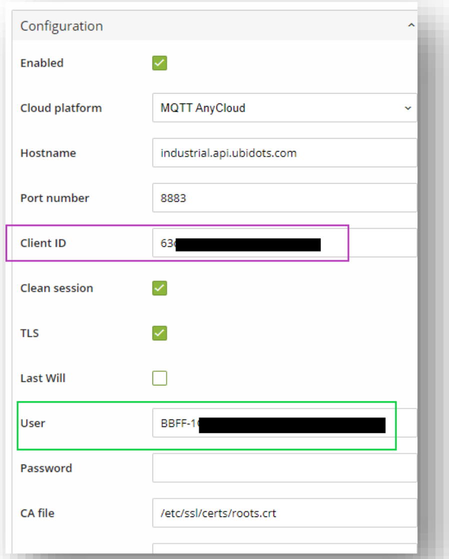
Token

••••••••••••••••••••••••

Tags

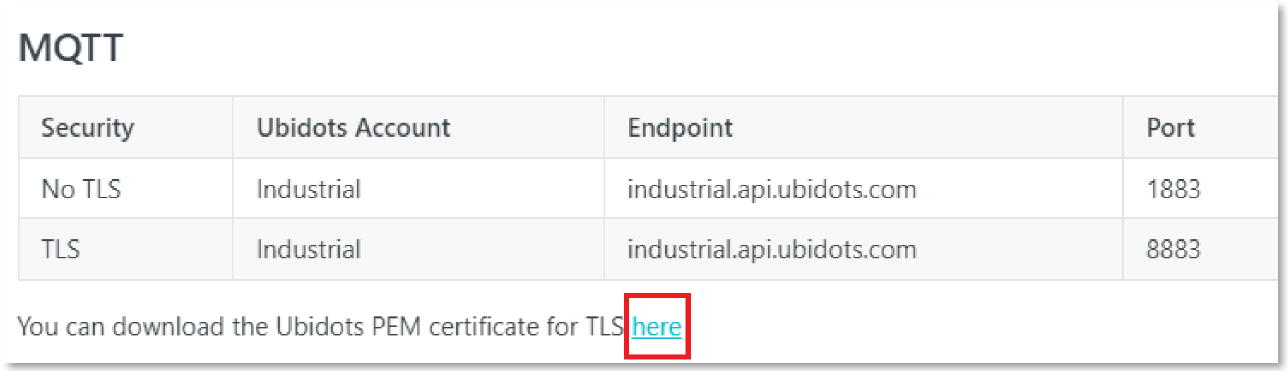
Add new tag

Wago WBM



Configuration

Enabled	✓
Cloud platform	MQTT AnyCloud
Hostname	industrial.api.ubidots.com
Port number	8883
Client ID	63[redacted]
Clean session	✓
TLS	✓
Last Will	☐
User	BBFF-1[redacted]
Password	
CA file	/etc/ssl/certs/roots.crt



MQTT

Security	Ubidots Account	Endpoint	Port
No TLS	Industrial	industrial.api.ubidots.com	1883
TLS	Industrial	industrial.api.ubidots.com	8883

You can download the Ubidots PEM certificate for TLS here

Connecting the Wago PFC200 to Ubidots Via Codesys

Video: PFC Config in WBM

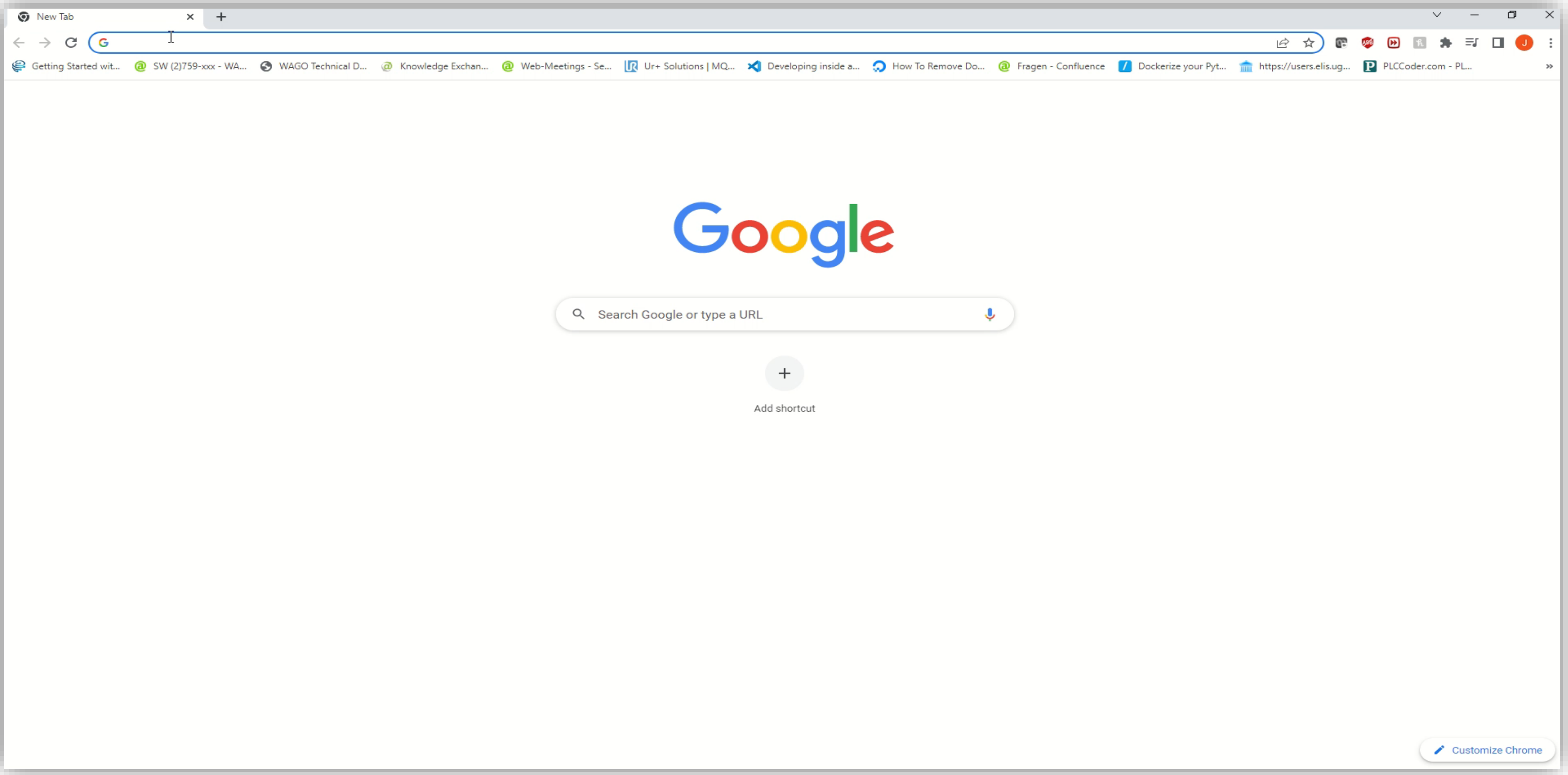
PFC200 Setup

2a)

i. Log onto the WBM of your PFC200 by using any web browser and type in the IP address of your device.

ii. Type in default username/password = admin/wago if prompted.

iii. Ensure the FW of the device is FW23 or above. Then check that the IP address, gateway, and subnet are all set for an outbound connection.



Note: My router is set to IP address 192.168.1.1, thus I've used it as the gateway address

Note: If you need to change the FW use the below link as a guide:

https://www.youtube.com/watch?v=6CxdrmHIIMo

https://github.com/WAGO/pfc-firmware

Connecting the Wago PFC200 to Ubidots Via Codesys

PFC200 Setup

2a)

iv. Check that your clock on the PFC200 is formatted to the correct time/timezone

v. Verify that you are connected online by using any ssh program to access the PFC terminal.

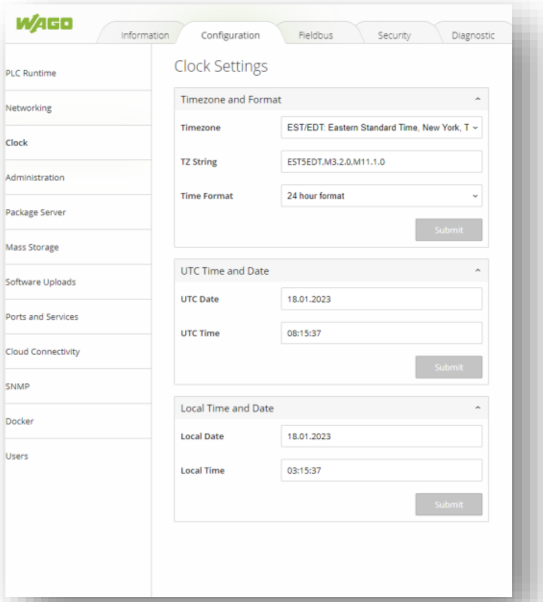
default credentials: root/wago

ping www.google.com

If you receive an immediate response with ping statistics, then you are connected.

2b)

i. Within the Wago web-based management go to the 'Cloud connectivity' tab under 'Configuration. Click on 'Connection 1'. Here is where we input the Ubidots credentials.



Clock Settings

Timezone and Format

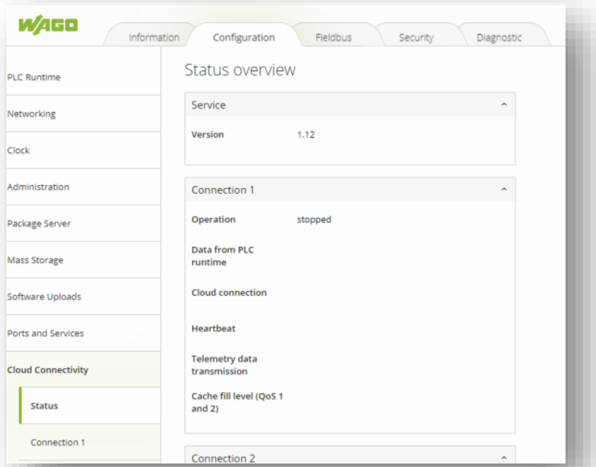
Timezone	EST/EDT: Eastern Standard Time, New York, T
TZ String	EST5EDT,M3.2.0,M11.1.0
Time Format	24 hour format

UTC Time and Date

UTC Date	18.01.2023
UTC Time	08:15:37

Local Time and Date

Local Date	18.01.2023
Local Time	03:15:37



Status overview

Service

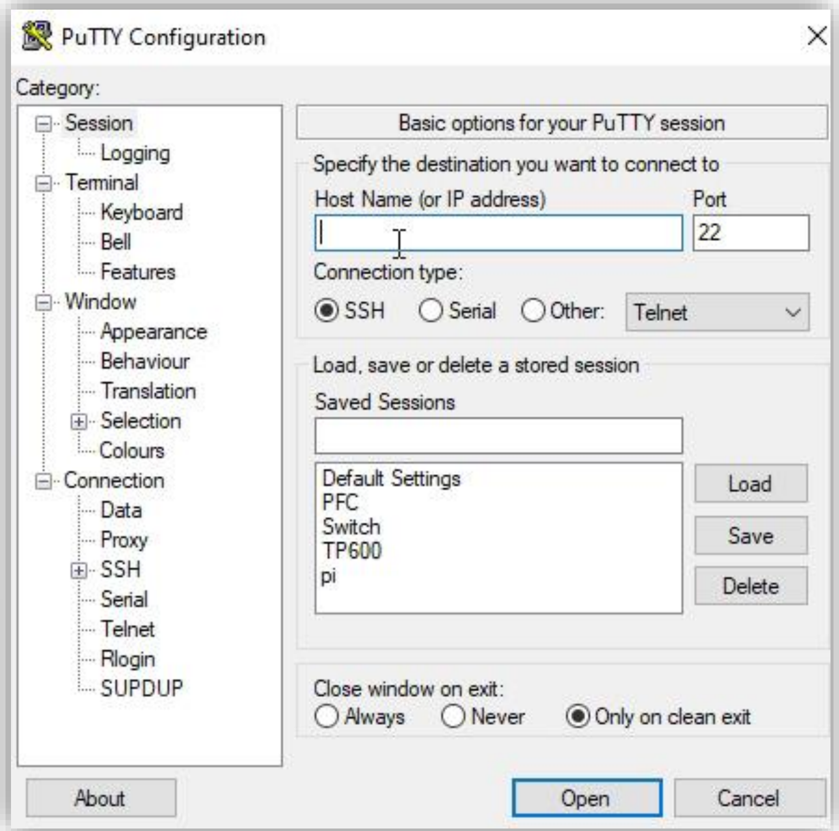
Version	1.12

Connection 1

Operation	stopped
Data from PLC runtime	
Cloud connection	
Heartbeat	
Telemetry data transmission	
Cache fill level (QoS 1 and 2)	

Connection 2

Video: Confirm online connection via SSH



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address)

Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

Load, save or delete a stored session

Saved Sessions

Default Settings
PFC
Switch
TP600
pi

Close window on exit: Always Never [✓ Only on clean exit]

Connecting the Wago PFC200 to Ubidots Via Codesys

PFC200 Setup

2b)

ii. Once on the 'Connection 1' page, select 'MQTT AnyCloud.

Input the Hostname 'industrial.api.ubidots.com'

iii. Click the 'TLS' box then use port number 8883.

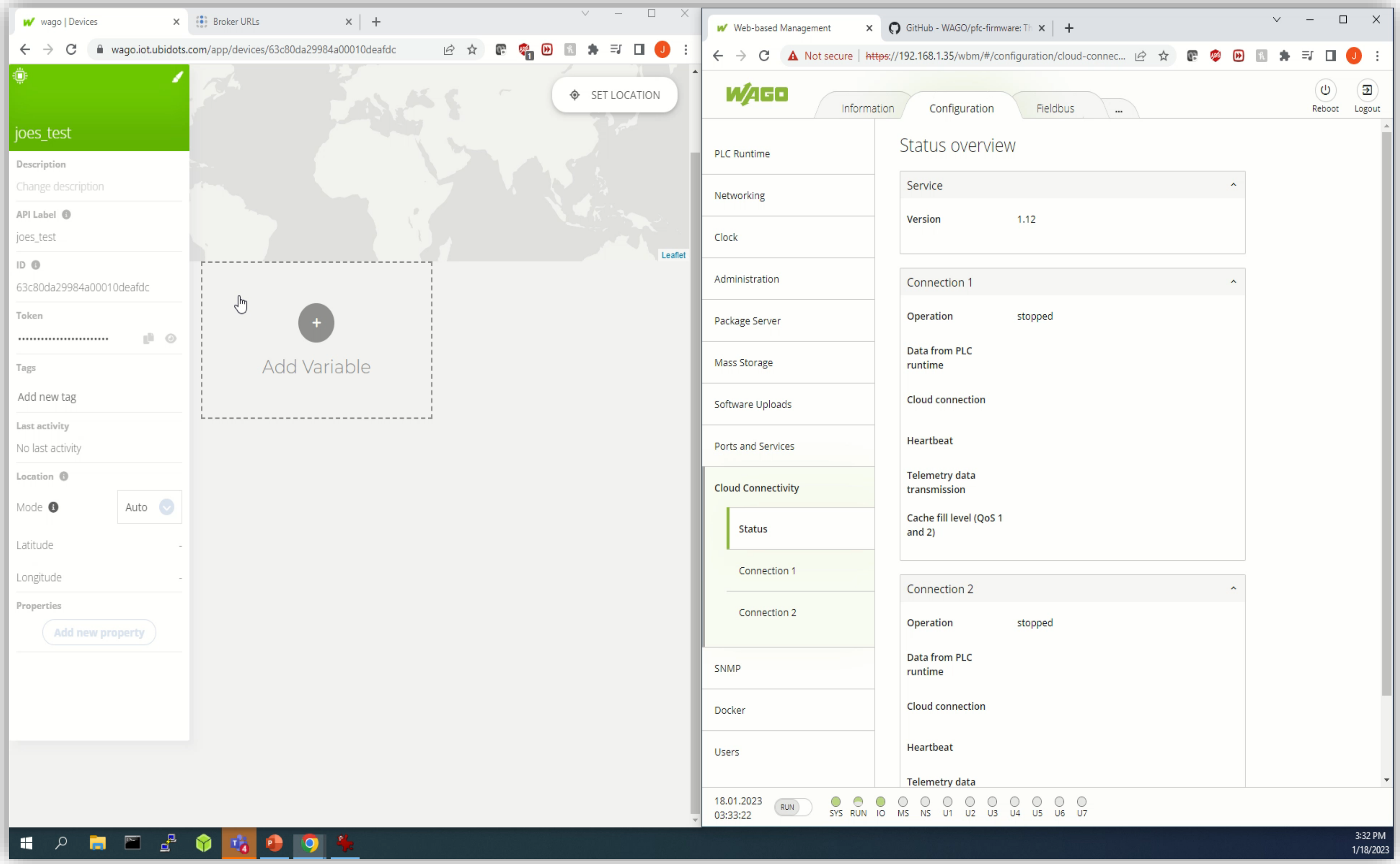
iv. Input the Ubidots ID into the Client ID field of the Wago WBM.

v. Input the Ubidots Token into the User field in the Wago WBM.

vi. Select 'Native MQTT' as the Data protocol in the WBM.

Note: Ensure there are no leading or trailing spaces within the input fields! This will cause the value to not be valid.

Video: Cloud setup in WBM



joes_test

Description: Change description

API Label: joes_test

ID: 63c80da29984a00010deafdc

Token: ••••••••••••••••••••••••

Tags: Add new tag

Last activity: No last activity

Location

Mode: Auto

Latitude: -

Longitude: -

Properties: Add new property

Status overview

Service

Version	1.12

Connection 1

Operation	stopped
Data from PLC runtime	
Cloud connection	
Heartbeat	
Telemetry data transmission	
Cache fill level (QoS 1 and 2)	

Connection 2

Operation	stopped
Data from PLC runtime	
Cloud connection	
Heartbeat	
Telemetry data	

18.01.2023 03:33:22 RUN SYS RUN IO MS NS U1 U2 U3 U4 U5 U6 U7

3:32 PM 1/18/2023

Connecting the Wago PFC200 to Ubidots Via Codesys

PFC200 Setup

2b)

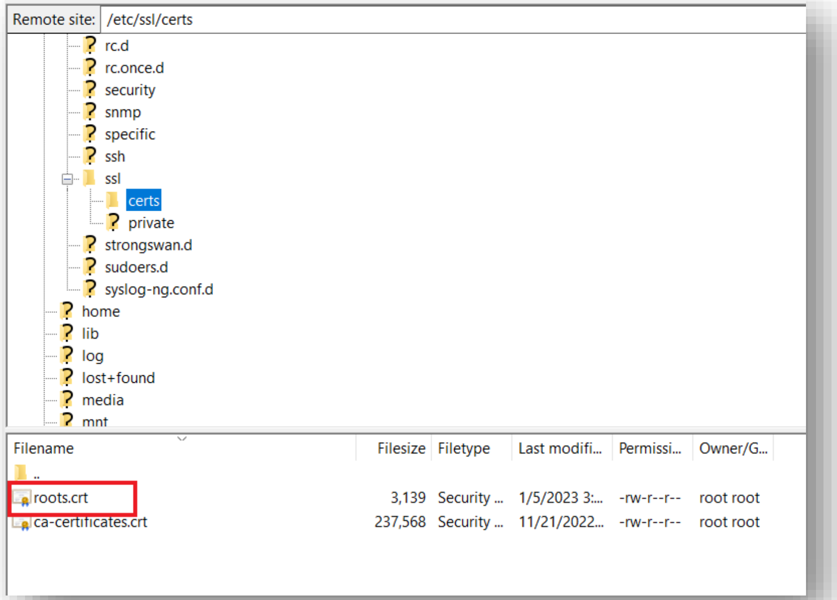
vii. Enable FTP via the ‘Ports and Services’ tab, then FTP into the Wago controller. Once this is done, insert the root.crt file into the CA file path ‘etc/ssl/certs/roots.crt’.

You can use any FTP utility for the file transfer, such as FileZilla

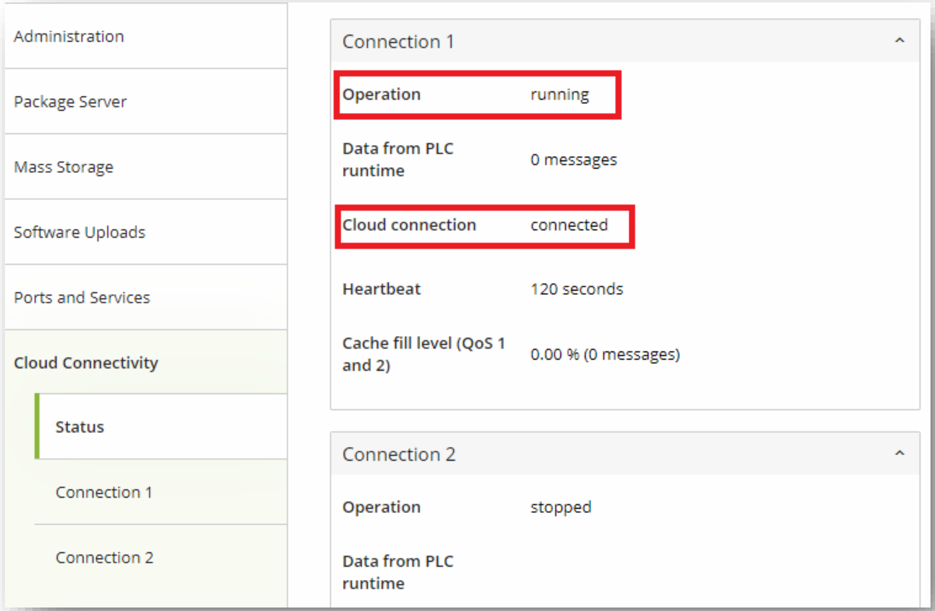
viii. Hit ‘Submit’, then reboot the controller via the ‘Reboot’ button the top right corner of the WBM.

ix. After waiting for the reboot, revisit the connection status on the WBM. You should now see a running connection.

We are now ready to create a Codesys project to publish data to Ubidots.


Remote site: /etc/ssl/certs

Filename	Filesize	Filetype	Last modifi...	Permissi...	Owner/G...
..					
roots.crt	3,139	Security ...	1/5/2023 3:...	-rw-r--r--	root root
ca-certificates.crt	237,568	Security ...	11/21/2022...	-rw-r--r--	root root

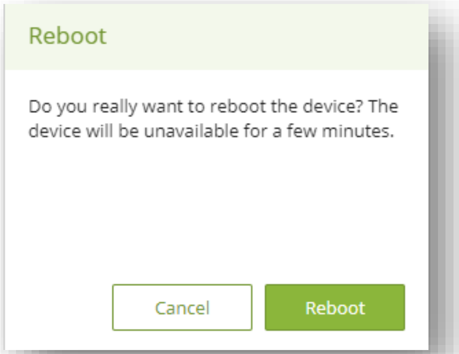


Connection 1

Operation	running
Data from PLC runtime	0 messages
Cloud connection	connected
Heartbeat	120 seconds
Cache fill level (QoS 1 and 2)	0.00 % (0 messages)

Connection 2

Operation	stopped
Data from PLC runtime	


Reboot

Do you really want to reboot the device? The device will be unavailable for a few minutes.

Connecting the Wago PFC200 to Ubidots Via Codesys

Codesys Setup

3a)

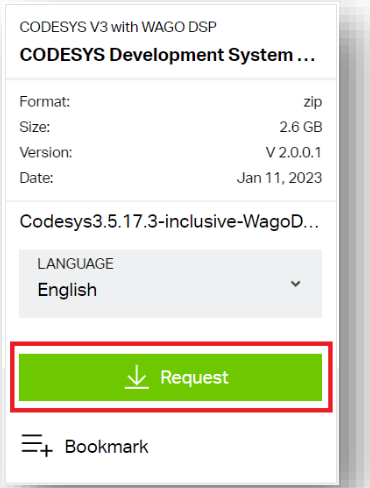
i. Go to Wago website to download and install the newest version of Codesys 3.5 w/ Device support package here:

https://www.wago.com/us/d/swreg_codesys_v3_dsp_c

ii. Create a 'New project', select empty project, and then name the project.

3b)

i. Add your Wago device. I'll be using the 750-8212 for this project (shown in video).


CODESYS V3 with WAGO DSP

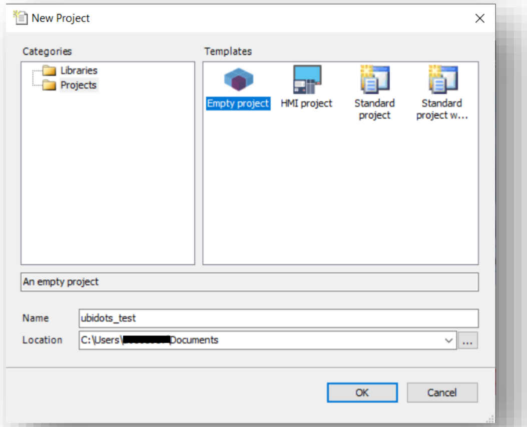
CODESYS Development System ...

Format:	zip
Size:	2.6 GB
Version:	V 2.0.0.1
Date:	Jan 11, 2023

Codesys3.5.17.3-inclusive-WagoD...

LANGUAGE
English

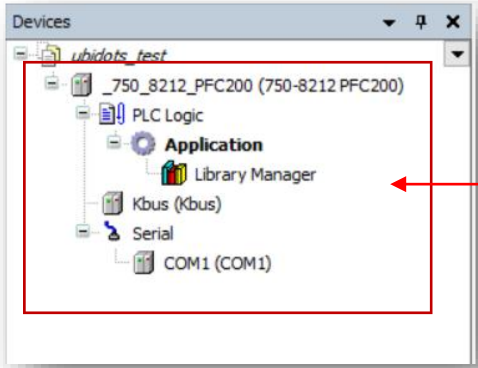
Note: At the time of this How-To the project is created with Codesys 3.5 SP17 Patch 3



An empty project

Name: ubidots_test

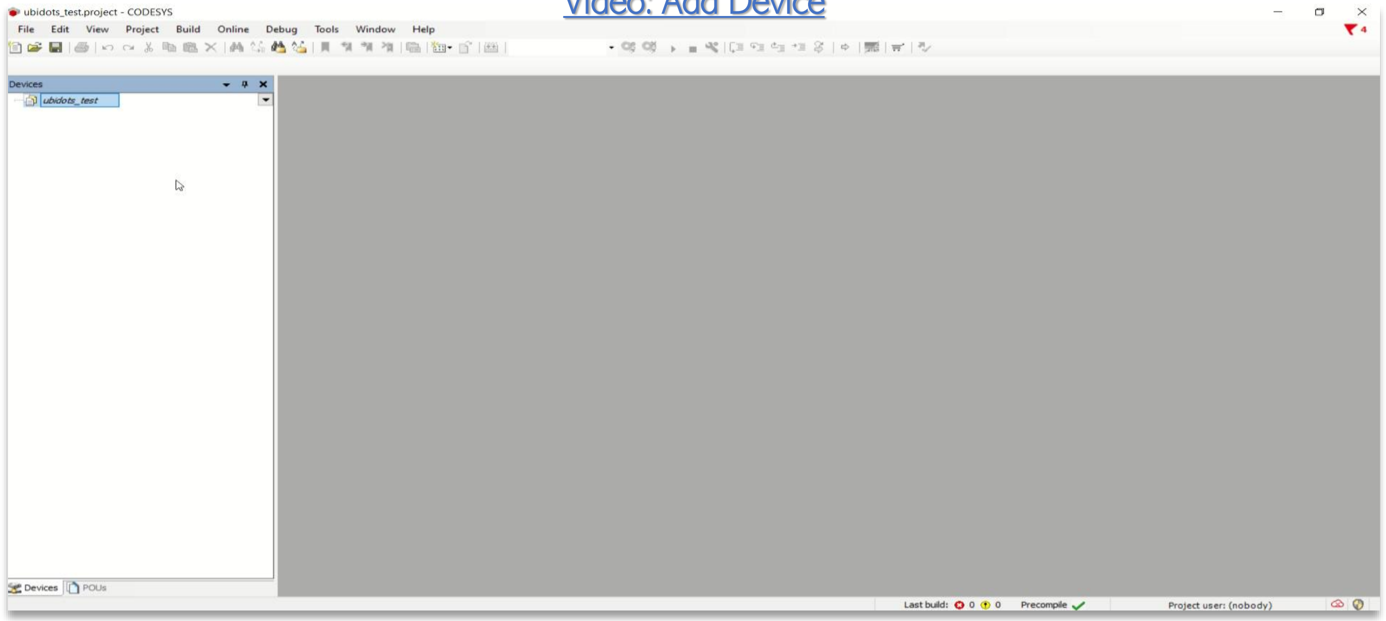
Location: C:\Users\[redacted]Documents



Note:

You'll see this added to the device screen once you've added your Wago Device

Video: Add Device



Connecting the Wago PFC200 to Ubidots Via Codesys

Codesys Setup

3c)

i. Create a Structured text POU called Main and Task configuration. Link that task to the POU via the ‘Add Call’ button.

ii. Double click on the device. Codesys will prompt you to create an active path

iii. Type in the IP address to the right field searching for active path.

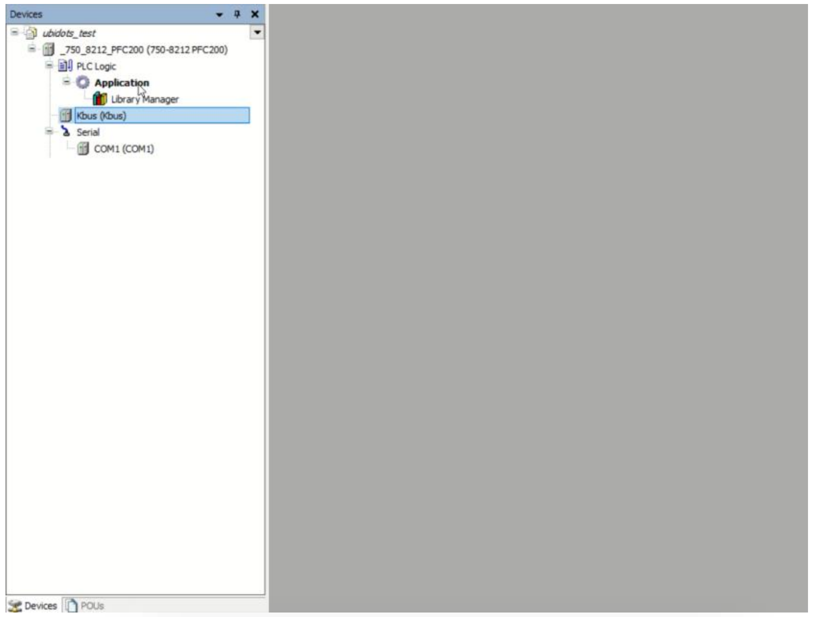
Default: admin/wago

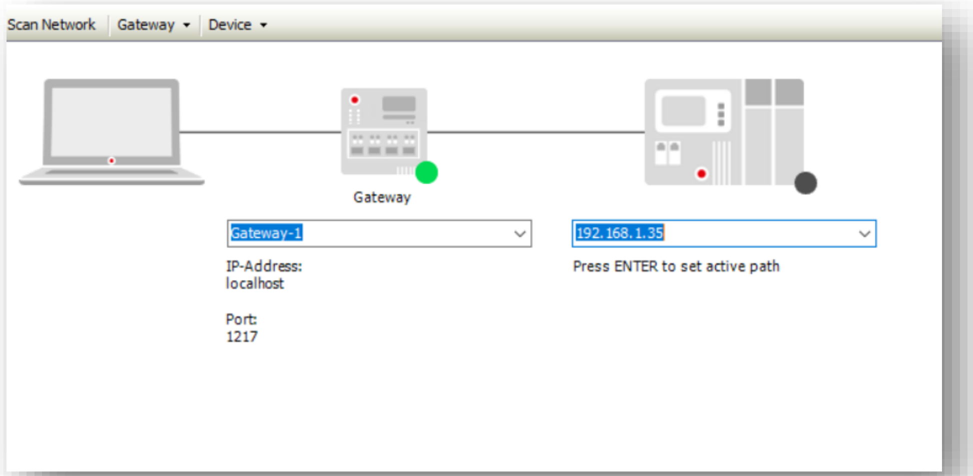
iv. Login and now you should be online!

v. Back to the device tab, right click on the kbus tab and click scan devices. This provides your IO count.

vi. Go back offline. Go to library manager, add library, and install WagoAppCloud & WagoAppJSON

Video: POU/Task config





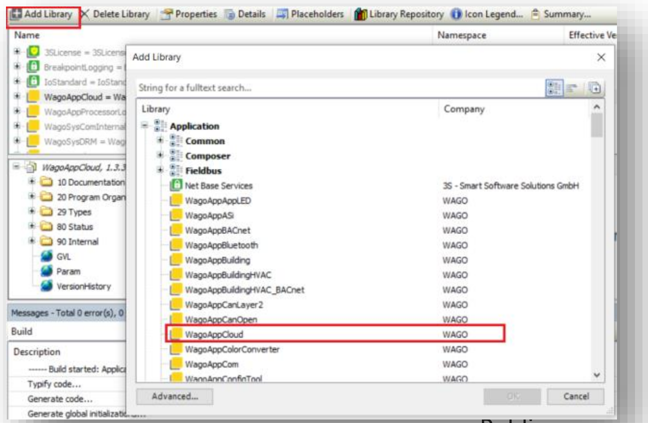
Gateway-1

IP-Address:
localhost

Port:
1217

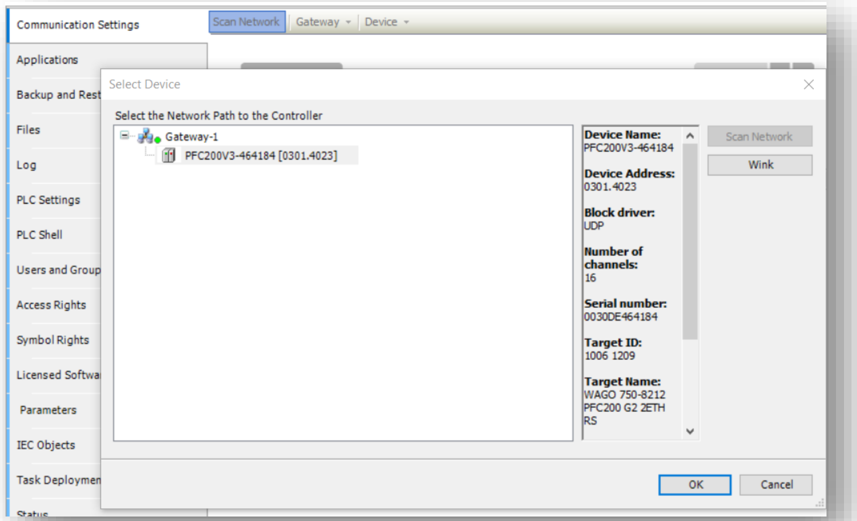
192.168.1.35

Press ENTER to set active path



```
------ Build started: Applica
Typify code...
Generate code...
Generate global initializatio...
```

Library	Company
Application	
Common	
Composer	
Fieldbus	
Net Base Services	3S - Smart Software Solutions GmbH
WagoAppAppLED	WAGO
WagoAppASi	WAGO
WagoAppBACnet	WAGO
WagoAppBluetooth	WAGO
WagoAppBuilding	WAGO
WagoAppBuildingHVAC	WAGO
WagoAppBuildingHVAC_BACnet	WAGO
WagoAppCanLayer2	WAGO
WagoAppCanOpen	WAGO
WagoAppCloud	WAGO
WagoAppColorConverter	WAGO
WagoAppCom	WAGO
WagoAppConfigTool	WAGO



Select the Network Path to the Controller

Gateway-1
PFC200V3-464184 [0301.4023]

Device Name:
PFC200V3-464184

Device Address:
0301.4023

Block driver:
UDP

Number of channels:
16

Serial number:
0030DE464184

Target ID:
1006 1209

Target Name:
WAGO 750-8212 PFC200 G2 2ETH RS

Connecting the Wago PFC200 to Ubidots Via Codesys

The Code (Publish): MQTT Library

For this application, we'll be using the FbPublishMQTT_2 function block.

sTopic will be a string representing the MQTT topic = '/v2.0/devices/{DEVICE_LABEL}'

In this example the topic is as follows: '/v2.0/devices/joes_test'

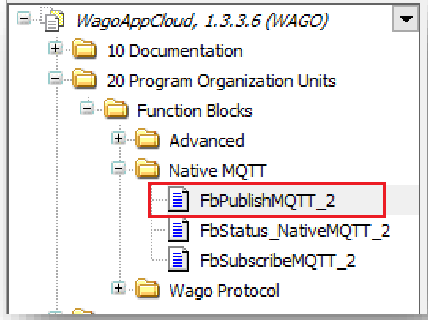
eQualityOfService = 1 which indicates that the MQTT publisher will send the message at least once to the MQTT Broker aka Ubidots.

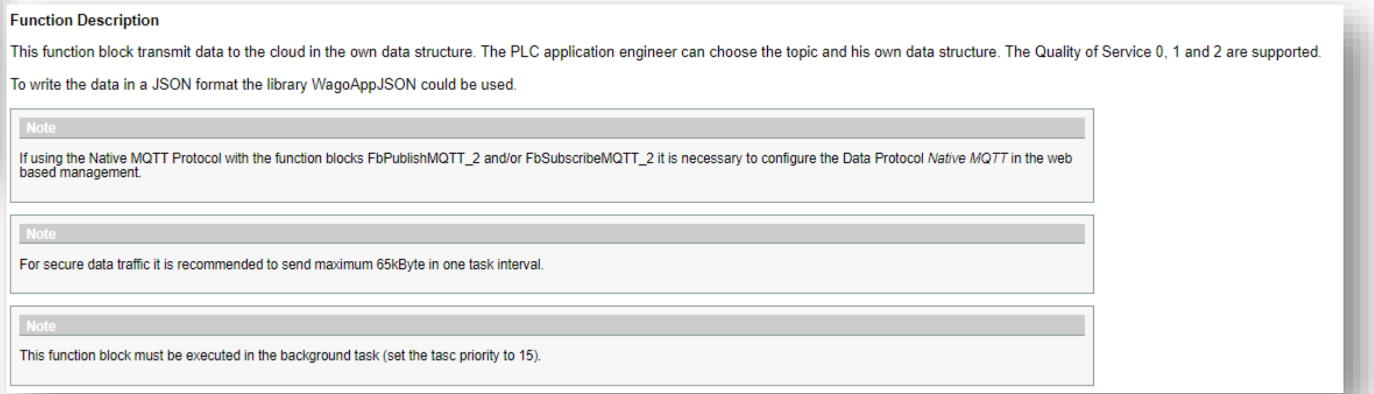
dwSize will be a dynamic variable that determines the length of the message string that will be sent to the broker, in bytes.

aData is the pointer of our JSON payload data to be published to the Broker. The data must be copied over from the JSON string to this aData array.

xTrigger will act as the trigger to begin publishing data and the outputs xBusy, xError, oStatus are troubleshooting indicators.

To copy the code on the next slides, use the following link: https://github.com/jabdelmalak/Ubidots-How-to





Function Description

This function block transmit data to the cloud in the own data structure. The PLC application engineer can choose the topic and his own data structure. The Quality of Service 0, 1 and 2 are supported.

To write the data in a JSON format the library WagoAppJSON could be used.

Note

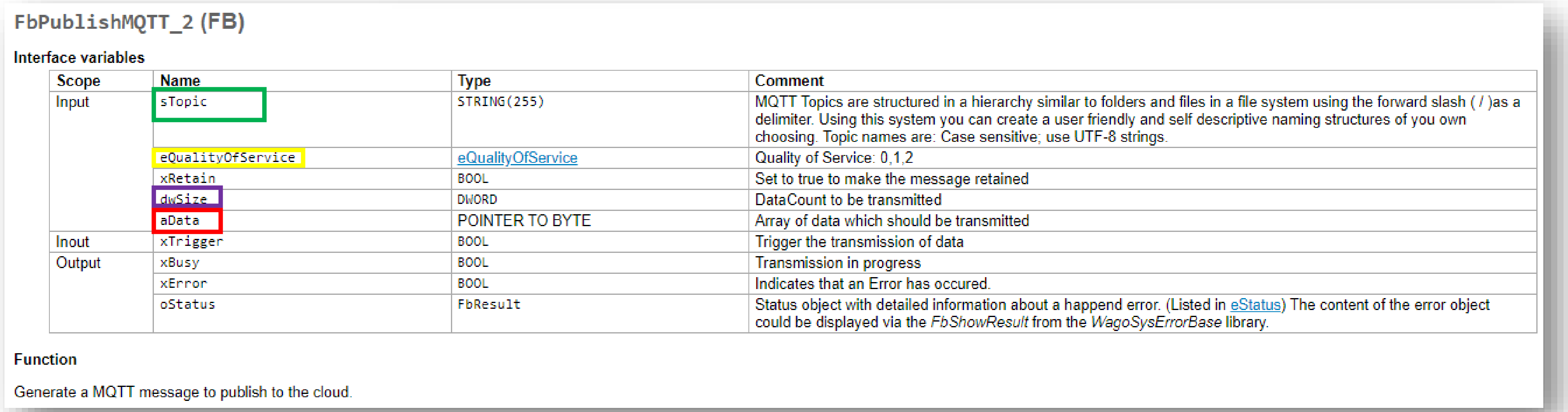
If using the Native MQTT Protocol with the function blocks FbPublishMQTT_2 and/or FbSubscribeMQTT_2 it is necessary to configure the Data Protocol *Native MQTT* in the web based management.

Note

For secure data traffic it is recommended to send maximum 65kByte in one task interval.

Note

This function block must be executed in the background task (set the tasc priority to 15).



FbPublishMQTT_2 (FB)

Interface variables

Scope	Name	Type	Comment
Input	sTopic	STRING(255)	MQTT Topics are structured in a hierarchy similar to folders and files in a file system using the forward slash (/)as a delimiter. Using this system you can create a user friendly and self descriptive naming structures of you own choosing. Topic names are: Case sensitive; use UTF-8 strings.
	eQualityOfService	eQualityOfService	Quality of Service: 0,1,2
	xRetain	BOOL	Set to true to make the message retained
	dwSize	DWORD	DataCount to be transmitted
	aData	POINTER TO BYTE	Array of data which should be transmitted
Inout	xTrigger	BOOL	Trigger the transmission of data
Output	xBusy	BOOL	Transmission in progress
	xError	BOOL	Indicates that an Error has occured.
	oStatus	FbResult	Status object with detailed information about a happend error. (Listed in eStatus) The content of the error object could be displayed via the *FbShowResult* from the *WagoSysErrorBase* library.

Function

Generate a MQTT message to publish to the cloud.

Connecting the Wago PFC200 to Ubidots Via Codesys

```
PROGRAM Main
VAR
   oFbPublishMQTT_2 : WagoAppCloud.FbPublishMQTT_2(eConnection := eConnectionId.Connection1);
   aBuffer           : ARRAY[0..1999] OF BYTE;
   dwBytesCount    : DWORD;
   sPayload       : STRING(1024);
   xTrigger       : BOOL;
   TimerOn: TON;
   dwBusyCounter: DWORD;
   dwErrorCounter: DWORD;
END_VAR
```

Example one: Simple Hard Code

Here we focus on initializing the MQTT publish function block as well as all of the variables the FB requires.

It's important to initialize the MQTT function block by specifying the specific cloud connection. Here we are using cloud connection 1.

Create a timer to act as the publishing interval, which in this case is every 1 second.

Create the JSON message payload. It's important to note that the type is a string with the format of quotes around the variable name and nothing around the value.

Example of one variable: '{"var_name": var_value}'
Example of two variables: '{"var_name_1": var_1_val, "var_name_2": var_2_val}'

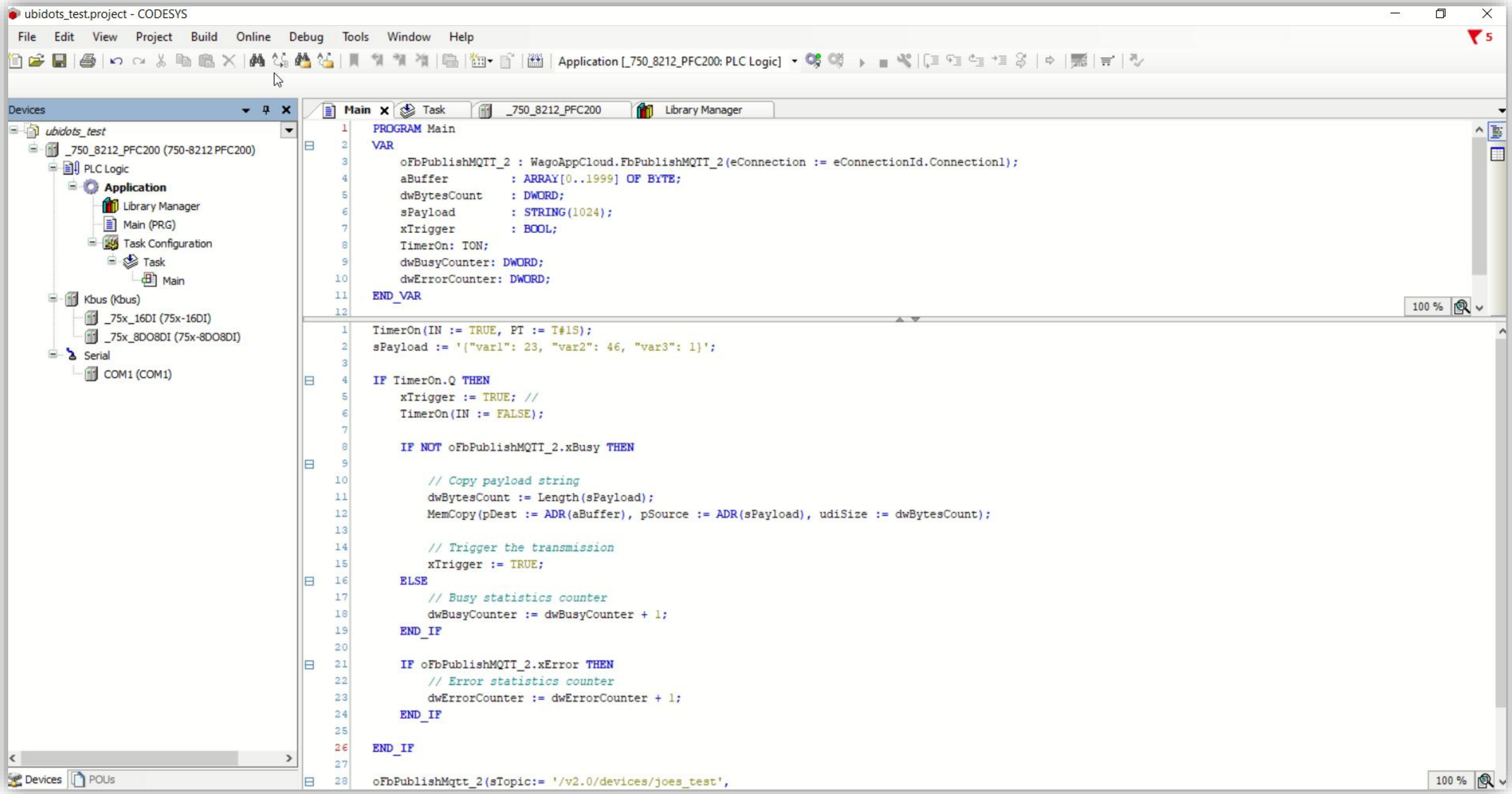
Once the timer has completed the elapsed time and the previous publishing function is complete, store length of the payload and copy the payload value to the publishing buffer array.

Actuate the publish function block with the data and length of data you'd like to transmit

Assuming the code has run with no issues, you should be able to reload the Ubidots page and find the new values auto populated in the device tab.

```
TimerOn(IN := TRUE, PT := T#1S);
sPayload := '{"var1": 23, "var2": 46, "var3": 1}';

IF TimerOn.Q THEN
   xTrigger := TRUE;
   TimerOn(IN := FALSE);

   IF NOT oFbPublishMQTT_2.xBusy THEN

      // Copy payload string
      dwBytesCount := Length(sPayload);
      MemCopy(pDest := ADR(aBuffer), pSource := ADR(sPayload), udiSize := dwBytesCount);

      // Trigger the transmission
      xTrigger := TRUE;
   ELSE
      // Busy statistics counter
      dwBusyCounter := dwBusyCounter + 1;
   END_IF

   IF oFbPublishMQTT_2.xError THEN
      // Error statistics counter
      dwErrorCounter := dwErrorCounter + 1;
   END_IF

END_IF

oFbPublishMqtt_2(sTopic:= '/v2.0/devices/joes_test',
   eQualityOfService:= 1,
   dwSize := dwBytesCount,
   aData := aBuffer,
   xTrigger := xTrigger);
```

Connecting the Wago PFC200 to Ubidots Via Codesys



```
PROGRAM Main
VAR
    oFbPublishMQTT_2 : WagoAppCloud.FbPublishMQTT_2(eConnection := eConnectionId.Connection1);
    aBuffer          : ARRAY[0..1999] OF BYTE;
    dwBytesCount     : DWORD;
    sPayload         : STRING(1024);
    xTrigger         : BOOL;
    TimerOn: TON;
    dwBusyCounter: DWORD;
    dwErrorCounter: DWORD;
END_VAR
```

```
TimerOn(IN := TRUE, PT := T#1S);
sPayload := '{"var1": 23, "var2": 46, "var3": 1}';

IF TimerOn.Q THEN
    xTrigger := TRUE; //
    TimerOn(IN := FALSE);

    IF NOT oFbPublishMQTT_2.xBusy THEN

        // Copy payload string
        dwBytesCount := Length(sPayload);
        MemCopy(pDest := ADR(aBuffer), pSource := ADR(sPayload), udiSize := dwBytesCount);

        // Trigger the transmission
        xTrigger := TRUE;
    ELSE
        // Busy statistics counter
        dwBusyCounter := dwBusyCounter + 1;
    END_IF

    IF oFbPublishMQTT_2.xError THEN
        // Error statistics counter
        dwErrorCounter := dwErrorCounter + 1;
    END_IF

END_IF

oFbPublishMqtt_2(sTopic:= '/v2.0/devices/joes_test',
```

Public

Connecting the Wago PFC200 to Ubidots Via Codesys

The Code (Publish): JSON Library

For this application, we'll be utilizing the Fb_JSON_Writer_02 function block in the WagoAppJSON library.

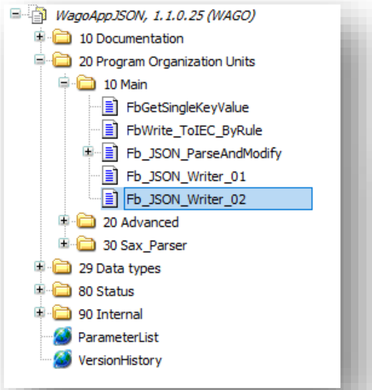
The functionality of this library is broken up into three parts: Creating the template string, the value string, and the execution of the function block

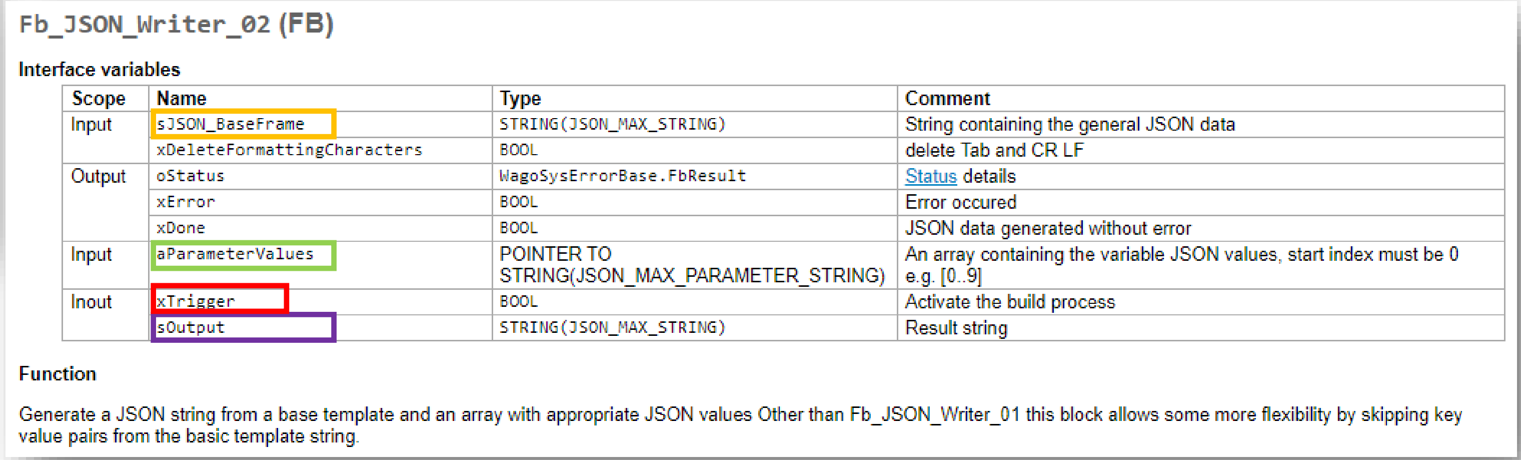
sJSON_BaseFrame will house the template of your JSON string. Any dynamic variables will have the placeholder '#Parameter' within the template string.

aParameterValues is an array of all variables that are represented in the template string as #Parameter. Since this array is a string, you'll need to convert your values into strings before putting them into the array.

xTrigger will cause the JSON function block to trigger its execution and create a JSON string represented by the string template and value array.

sOutput will be the final JSON string result.


```
WagoAppJSON, 1.1.0.25 (WAGO)
  10 Documentation
  20 Program Organization Units
    10 Main
      FbGetSingleKeyValue
      FbWrite_ToIEC_ByRule
      Fb_JSON_ParseAndModify
      Fb_JSON_Writer_01
      Fb_JSON_Writer_02
    20 Advanced
    30 Sax_Parser
  29 Data types
  80 Status
  90 Internal
  ParameterList
  VersionHistory
```

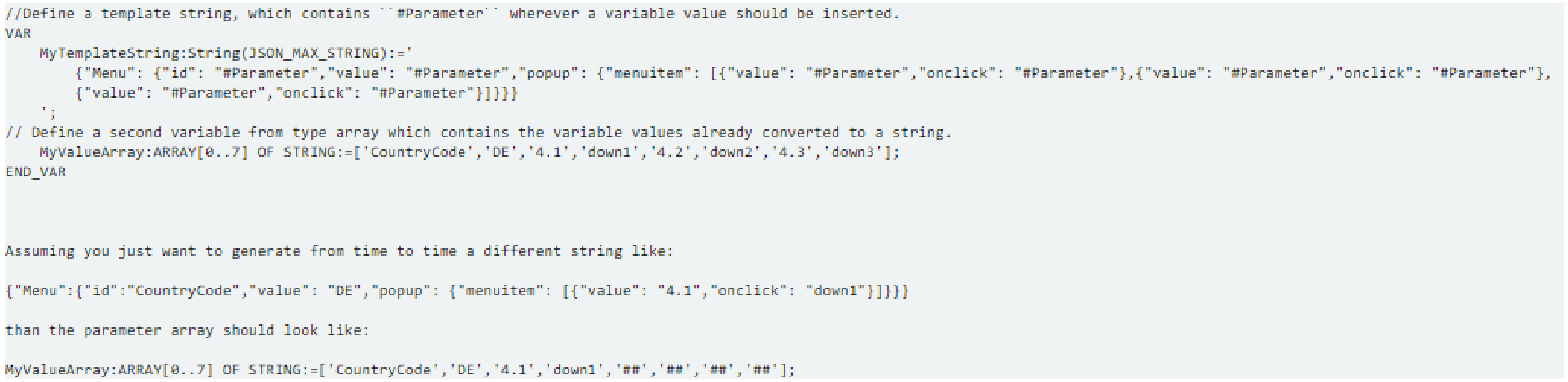

Fb_JSON_Writer_02 (FB)

Interface variables

Scope	Name	Type	Comment
Input	sJSON_BaseFrame	STRING(JSON_MAX_STRING)	String containing the general JSON data
	xDeleteFormattingCharacters	BOOL	delete Tab and CR LF
Output	oStatus	WagoSysErrorBase.FbResult	Status details
	xError	BOOL	Error occured
	xDone	BOOL	JSON data generated without error
Input	aParameterValues	POINTER TO STRING(JSON_MAX_PARAMETER_STRING)	An array containing the variable JSON values, start index must be 0 e.g. [0..9]
Inout	xTrigger	BOOL	Activate the build process
	sOutput	STRING(JSON_MAX_STRING)	Result string

Function

Generate a JSON string from a base template and an array with appropriate JSON values Other than Fb_JSON_Writer_01 this block allows some more flexibility by skipping key value pairs from the basic template string.


```
//Define a template string, which contains ``#Parameter`` wherever a variable value should be inserted.
VAR
    MyTemplateString:String(JSON_MAX_STRING):='
        {"Menu": {"id": "#Parameter","value": "#Parameter","popup": {"menuitem": [{"value": "#Parameter","onclick": "#Parameter"},{"value": "#Parameter","onclick": "#Parameter"},
        {"value": "#Parameter","onclick": "#Parameter"}]}}}
    ';
// Define a second variable from type array which contains the variable values already converted to a string.
    MyValueArray:ARRAY[0..7] OF STRING:=['CountryCode','DE','4.1','down1','4.2','down2','4.3','down3'];
END_VAR



Assuming you just want to generate from time to time a different string like:

{"Menu":{"id":"CountryCode","value": "DE","popup": {"menuitem": [{"value": "4.1","onclick": "down1"}]}}}

than the parameter array should look like:

MyValueArray:ARRAY[0..7] OF STRING:=['CountryCode','DE','4.1','down1','##','##','##','##'];
```

Connecting the Wago PFC200 to Ubidots Via Codesys

The Code (Publish): Example 2 Parameterized JSON

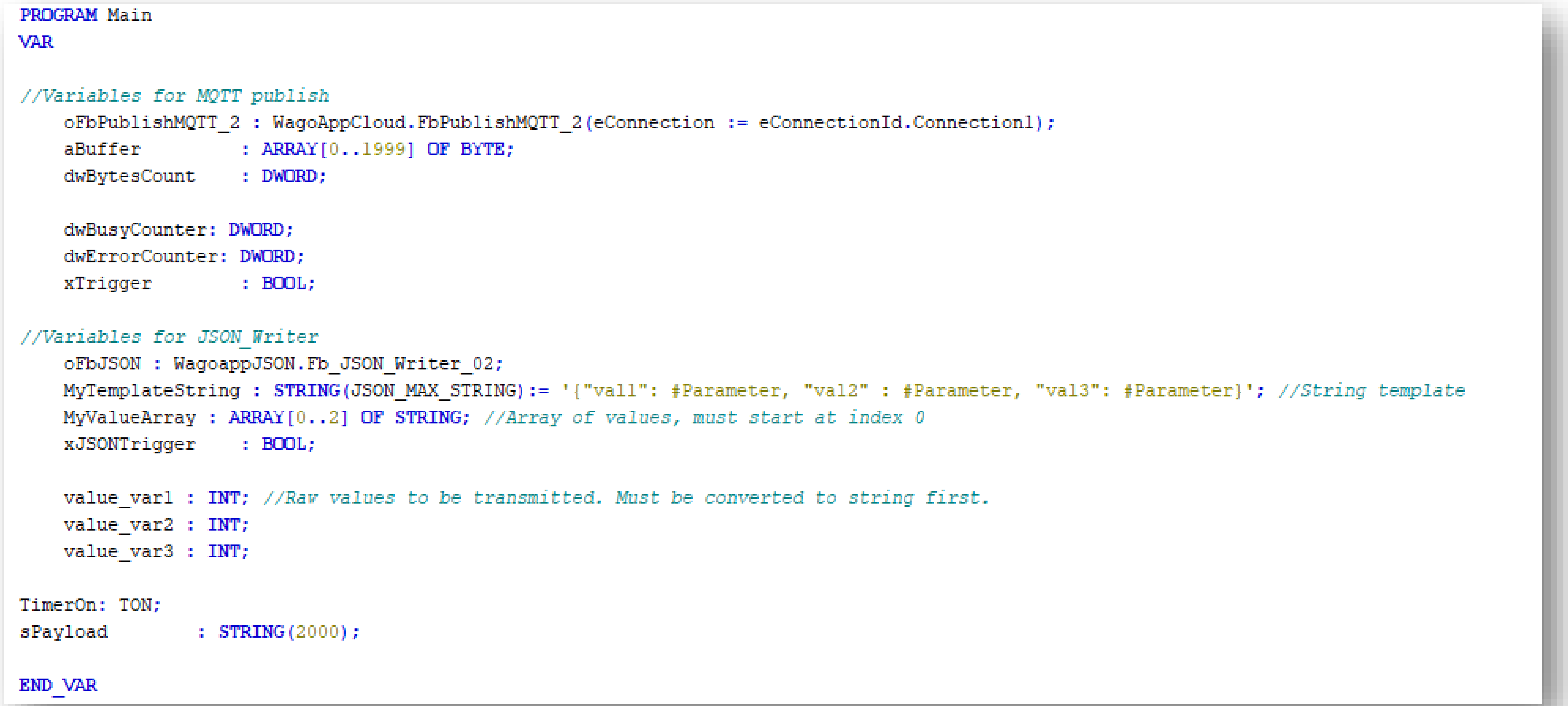
Initializing the Template as MyTemplateString

Creating the value array of size three because we are passing three variables

Initializing the three integer values and creating a new xJSON trigger variable

To copy the code, use the following link: https://github.com/jabdelmalak/Ubidots-How-to

Variable Declaration



```
PROGRAM Main
VAR

//Variables for MQTT publish
    oFbPublishMQTT_2 : WagoAppCloud.FbPublishMQTT_2(eConnection := eConnectionId.Connection1);
    aBuffer          : ARRAY[0..1999] OF BYTE;
    dwBytesCount     : DWORD;

    dwBusyCounter: DWORD;
    dwErrorCounter: DWORD;
    xTrigger         : BOOL;

//Variables for JSON_Writer
    oFbJSON : WagoappJSON.Fb_JSON_Writer_02;
    MyTemplateString : STRING(JSON_MAX_STRING):= '{"val1": #Parameter, "val2" : #Parameter, "val3": #Parameter}'; //String template
    MyValueArray : ARRAY[0..2] OF STRING; //Array of values, must start at index 0
    xJSONTrigger     : BOOL;

    value_var1 : INT; //Raw values to be transmitted. Must be converted to string first.
    value_var2 : INT;
    value_var3 : INT;

TimerOn: TON;
sPayload         : STRING(2000);

END_VAR
```

Connecting the Wago PFC200 to Ubidots Via Codesys

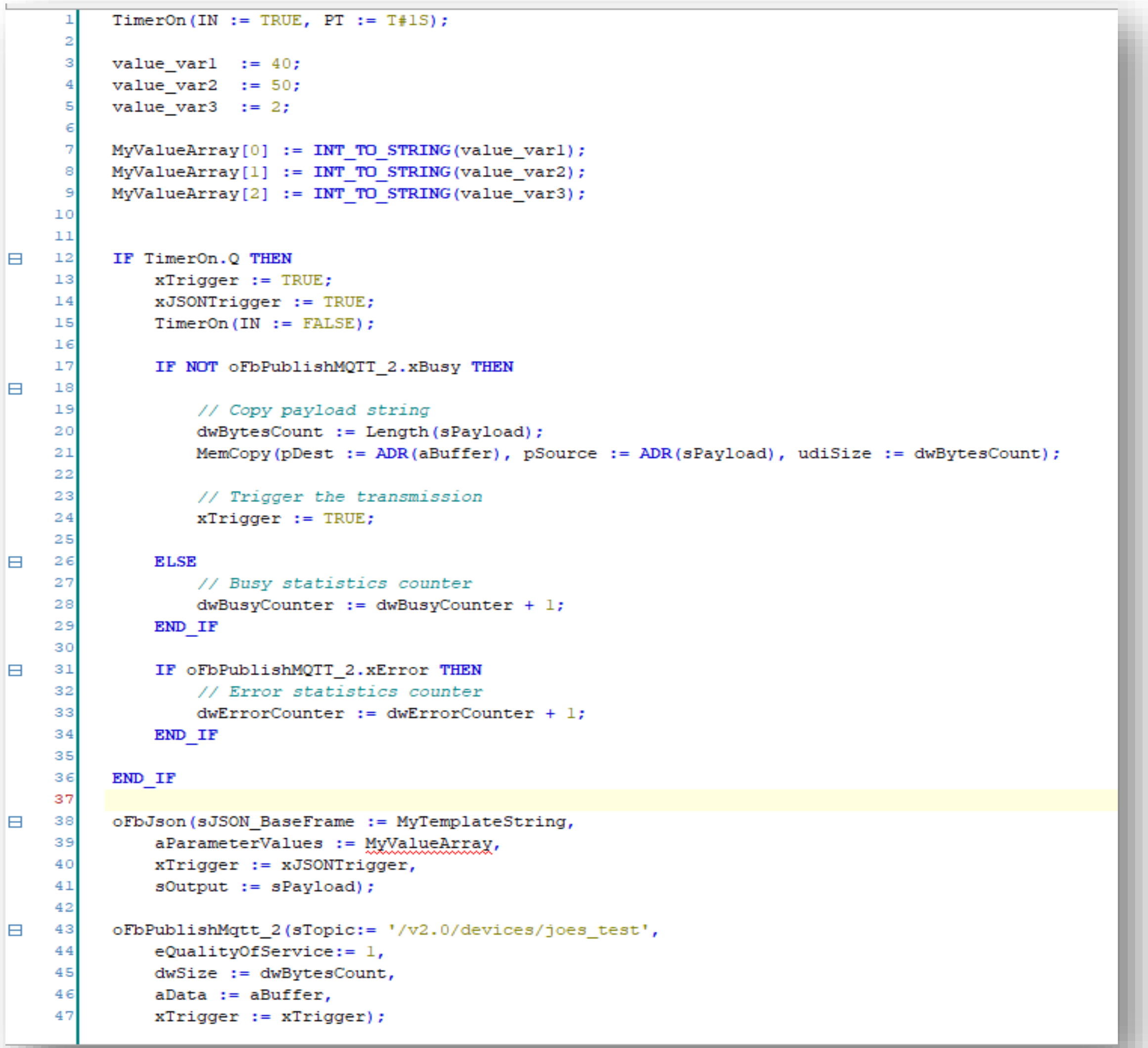
The Code (Publish): Example 2 Parameterized JSON

Individually setting values to the variables

We are then setting each value array element to the string form of the variable via Int_to_string conversion function

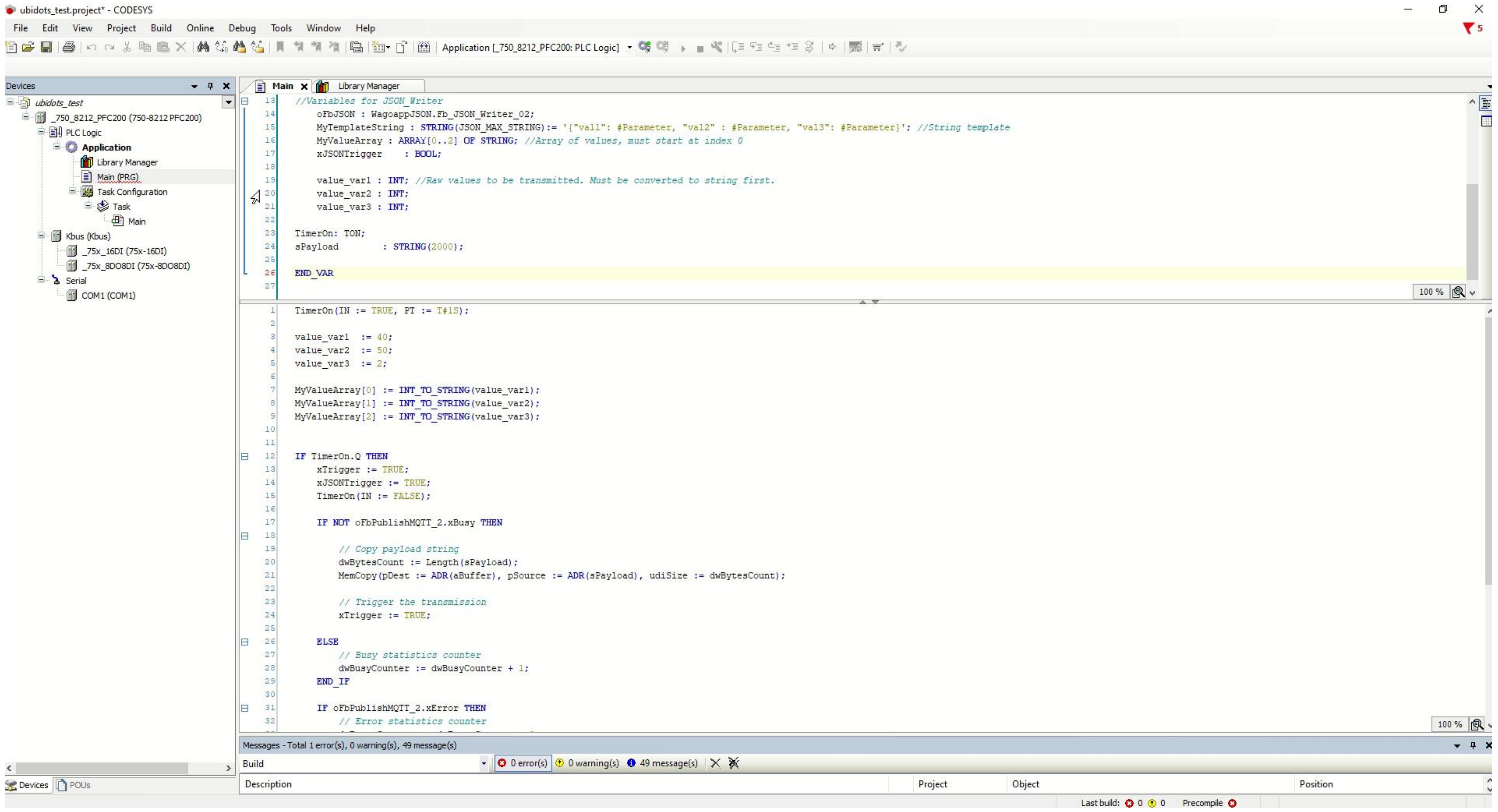
Add the JSON trigger to the timing if statement and then run the function block with sPayload as the output parameter.

Program Sequence



```
TimerOn(IN := TRUE, PT := T#1S);

value_var1  := 40;
value_var2  := 50;
value_var3  := 2;

MyValueArray[0] := INT_TO_STRING(value_var1);
MyValueArray[1] := INT_TO_STRING(value_var2);
MyValueArray[2] := INT_TO_STRING(value_var3);


IF TimerOn.Q THEN
    xTrigger := TRUE;
    xJSONTrigger := TRUE;
    TimerOn(IN := FALSE);

    IF NOT oFbPublishMQTT_2.xBusy THEN

        // Copy payload string
        dwBytesCount := Length(sPayload);
        MemCopy(pDest := ADR(aBuffer), pSource := ADR(sPayload), udiSize := dwBytesCount);

        // Trigger the transmission
        xTrigger := TRUE;

    ELSE
        // Busy statistics counter
        dwBusyCounter := dwBusyCounter + 1;
    END_IF

    IF oFbPublishMQTT_2.xError THEN
        // Error statistics counter
        dwErrorCounter := dwErrorCounter + 1;
    END_IF

END_IF

oFbJson(sJSON_BaseFrame := MyTemplateString,
    aParameterValues := MyValueArray,
    xTrigger := xJSONTrigger,
    sOutput := sPayload);

oFbPublishMqtt_2(sTopic:= '/v2.0/devices/joes_test',
    eQualityOfService:= 1,
    dwSize := dwBytesCount,
    aData := aBuffer,
    xTrigger := xTrigger);
```

Connecting the Wago PFC200 to Ubidots Via Codesys



```
//Variables for JSON_Writer
    oFbJSON : WagoappJSON.Fb_JSON_Writer_02;
    MyTemplateString : STRING(JSON_MAX_STRING):= '{"val1": #Parameter, "val2" : #Parameter, "val3": #Parameter}'; //String template
    MyValueArray : ARRAY[0..2] OF STRING; //Array of values, must start at index 0
    xJSONTrigger     : BOOL;

    value_var1 : INT; //Raw values to be transmitted. Must be converted to string first.
    value_var2 : INT;
    value_var3 : INT;

TimerOn: TON;
sPayload        : STRING(2000);

END_VAR

```

```
TimerOn(IN := TRUE, PT := T#1S);

value_var1  := 40;
value_var2  := 50;
value_var3  := 2;

MyValueArray[0] := INT_TO_STRING(value_var1);
MyValueArray[1] := INT_TO_STRING(value_var2);
MyValueArray[2] := INT_TO_STRING(value_var3);


IF TimerOn.Q THEN
    xTrigger := TRUE;
    xJSONTrigger := TRUE;
    TimerOn(IN := FALSE);

    IF NOT oFbPublishMQTT_2.xBusy THEN

        // Copy payload string
        dwBytesCount := Length(sPayload);
        MemCopy(pDest := ADR(aBuffer), pSource := ADR(sPayload), udiSize := dwBytesCount);

        // Trigger the transmission
        xTrigger := TRUE;

    ELSE
        // Busy statistics counter
        dwBusyCounter := dwBusyCounter + 1;
    END_IF

    IF oFbPublishMQTT_2.xError THEN
        // Error statistics counter
```

Messages - Total 1 error(s), 0 warning(s), 49 message(s)

Build | 0 error(s) | 0 warning(s) | 49 message(s)

Description	Project	Object	Position

Last build: 0 0 Precompile