Sensor Monitoring

PLC versus periSNOOP for Sensor Data Monitoring

Application Note

Perinet
Seamless IoT Connectivity

Abstract

A periSNOOP 4-20mA product easily integrates into 4-20mA sensor to the PLC loop, making sensor data reading digitilized and readily available using periCORE SPE communication module. Without the need to modify the existing installations and additional PLC software, the sensor data reading is made easily accessible.

Doc-No.: PRN.100.921 rev: 1 2025-10-20

Document Information

Title	Sensor Monitoring
Subtitle	PLC versus periSNOOP for Sensor Data Monitoring
Type	Application Note
Status	Release
Version	1
Date	2025-10-20
Disclosure Restriction	

Contents

1 Introduction

The key takeaway of this application note is the dramatic simplification periSNOOP brings to sensor data access compared to a PLC-only approach. With periSNOOP, all it takes is clipping the device onto the existing 4–20 mA loop, opening the built-in web UI, and instantly viewing live current values or direct engineering-unit conversions. No TwinCAT installation, task creation, library imports, or AMS Net-ID routing are required.

In contrast, the PLC route demands significant overhead: installing software, configuring tasks, importing libraries, and setting up communication routing—plus writing two custom programs to manage a 300,000-element array, cycle-time buffering, and file operations for storing measurement data.

By taking a non-invasive approach, periSNOOP makes sensor data immediately available in IT-ready formats, whether raw or transformed, and streams it seamlessly into dashboards or existing systems. This eliminates the need for OPC UA licenses, ADS scripting, or complex intermediate steps—streamlining workflows, reducing setup effort, and lowering costs.

This application note demonstrates how effortlessly sensor data can be digitized and evaluated using the periSNOOP smart component, offering a streamlined alternative to traditional PLC-based approaches. By integrating seamlessly into existing 4-20 mA sensor loops, periSNOOP enables quick, non-intrusive access to sensor information without requiring changes to installations or PLC software. Combined with the periCORE SPE communication module, it delivers sensor data in IT-ready formats, simplifying monitoring and analysis. To provide a solid basis for comparison, this document presents a detailed evaluation of data extraction and processing between a PLC with EtherCAT and a sensor, versus the periSNOOP 4-20 mA component with the same sensor.

2 periSNOOP-based Data Monitoring

As PLCs exist in numerous variants and covering them all is beyond the scope of this document, we will focus on one widely used example in the industry: the Beckhoff C6015 PLC with EK1100 EtherCAT bus coupler hosting an EL3152 analog input terminal. This PLC setup is connected to the periSNOOP 4–20 mA component, which, as its name implies, snoops the sensor data from the 4–20 mA ultrasonic distance sensor (model AU016) connected to the PLC. In summary, the setup contains the following components, with the interaction flow shown in Figure 1:

1. AU016 4-20mA ultrasonic sensor [1]
2. periLINE [2]
3. periSNOOP 4-20mA [4]
4. periSTART [5]
5. periSWITCH [6]
6. Beckhoff C6015 Industrial PC [7]
7. Beckhoff EK1100 EtherCAT Coupler [8]
8. Beckhoff EL3152 Analog Input Terminals [9]
9. Development computer running Beckhoff TwinCat XAE Software [10]

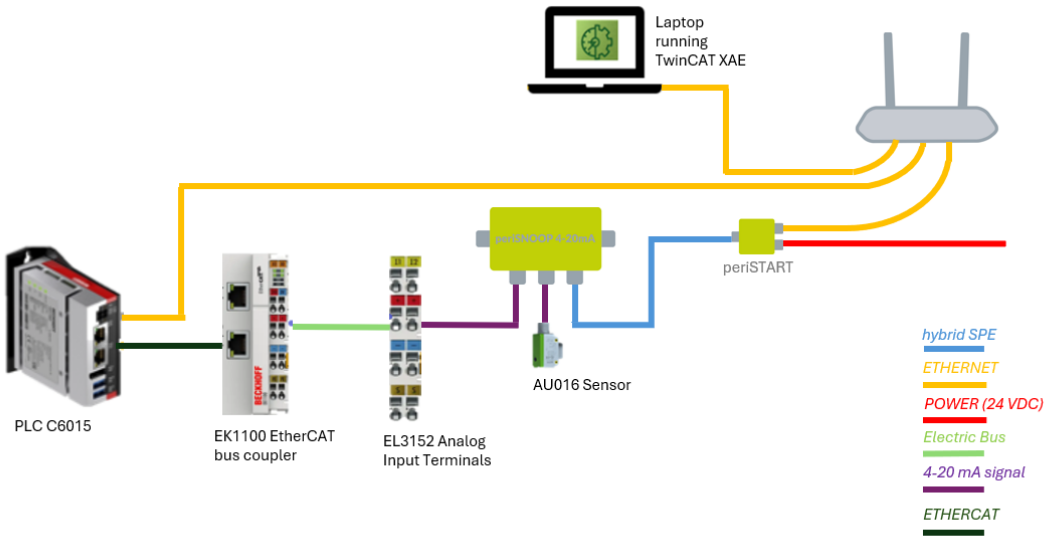

Figure 1: Block diagram of the PLC and periSNOOP setup

For the users who would like to get further details about periSNOOP 4-20mA product, we recommend them to refer to the detailed product summary [4] and its datasheet [3]. A simple physical connection without requiring any tools, using the cut and grip technique to avoid

the need for a conventional connector and socket combination is all that is necessary. The configuration of the periSNOOP 4-20mA is even easier from the web user interface. You can either track the sensor data in mA as seen in Figure 2, or alternatively enter your simple polynomial function for converting the 4-20mA analog sensor data to 40-300 mm distance data from the configuration section given in Figure 3.

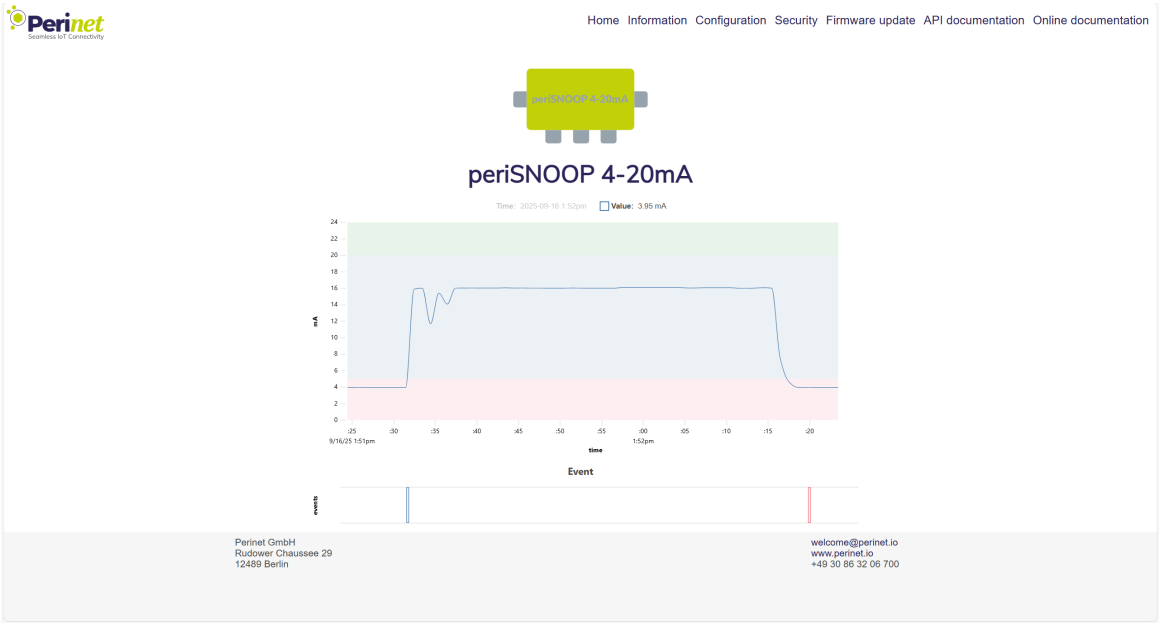

Figure 2: periSNOOP Home page

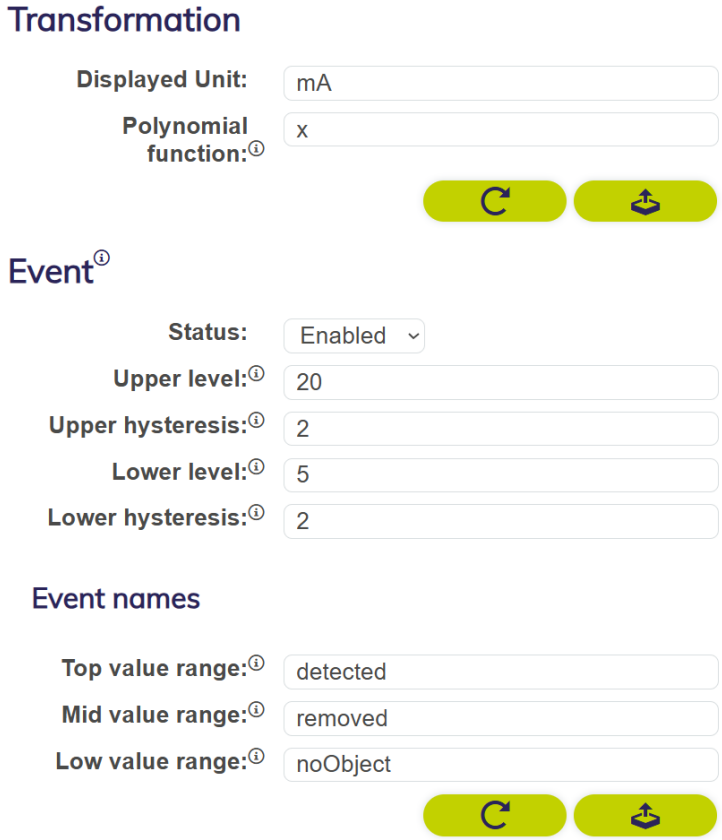

Figure 3: periSNOOP Transformation function

3 PLC-based Data Monitoring

After starting with the Perinet products based easy data monitoring and accessing, the next step would be to list down possible scenarios for sensor data monitoring and extraction using the existing PLC-based solutions. There are several ways to do it, depending on the engineering effort and the budget that can be invested. The common thing for all these options is that they are not as straightforward as using the periSNOOP for live monitoring and periSNOOP and periMICA together for both live and historical monitoring. A couple of options for the C6015 PLC can be listed down:

- Use ADS client API and run an additional script
    a) TwinCAT exposes every mapped variable through the ADS protocol. You need to add an ADS route (e.g. using the AMS Net-ID of the PLC, TwinCAT 3 runtime port, IP address of the C6015) from your PC to read/subscribe to get these variable values by using libraries such as pyads (python), TwinCAT.ADS (C#) or ads-client (Node.js). So, you can feed live values into Python / SQL / web dashboards. A data update rate that is close to the PLC-task cycle (e.g. in a few ms) duration would be possible to achieve.
- Stream your EL3152 variable every PLC cycle to any Scope View (e.g. Visual Studio Shell) for live only monitoring
    a) TwinCAT 3 Engineering (XAE) Beckhoff drops all its plug-ins – PLC editor, I/O configuration, Scope, HMI, etc. – straight into Microsoft Visual Studio. Under your Solution Explorer, you can right-click and add TwinCAT Scope YT Project create a dedicated measurement project for live-only scope. You can visualize in a rate equivalent to the PLC-task cycle (e.g. 1ms).
- A plant-wide standardised interface using OPC UA server or TwinCAT IoT communication (MQTT)
    a) By enabling a TF6100 server you can map symbols and create UA client subscription with licence fee. A data update rate close to the PLC-task cycle (e.g. in few ms) duration would be possible to achieve.
    b) The TwinCAT Iot Data Agent can provide the bi-directional communication with a licence fee.
- Store into a PLC-side CSV file and analyze
    a) This is the quickest to implement and licence-free option that requires additional offline analysis, which we present here in this paper for a limited value of 50 000 consecutive input values, i.e. 50 sec of measurement for 1 ms cycle time.

4 Setting up PLC data

From the options described in the previous section, in this application note we provide the steps to store the sensor data into a CSV file on the PLC-side and its analysis. TwinCAT XAE is the required development environment to be set up on a computer to establish a connection with the Beckhoff C6015 PLC. In this section, the steps for storing the PLC data into a CSV file and an offline analysis of the stored sensor data will be presented.

1. After the download of the program, the user should open the TwinCAT XAE shell program and start with creating a new solution from File -> New -> Project -> TcXaeShell Solution to create a blank solution as seen in Figure 4

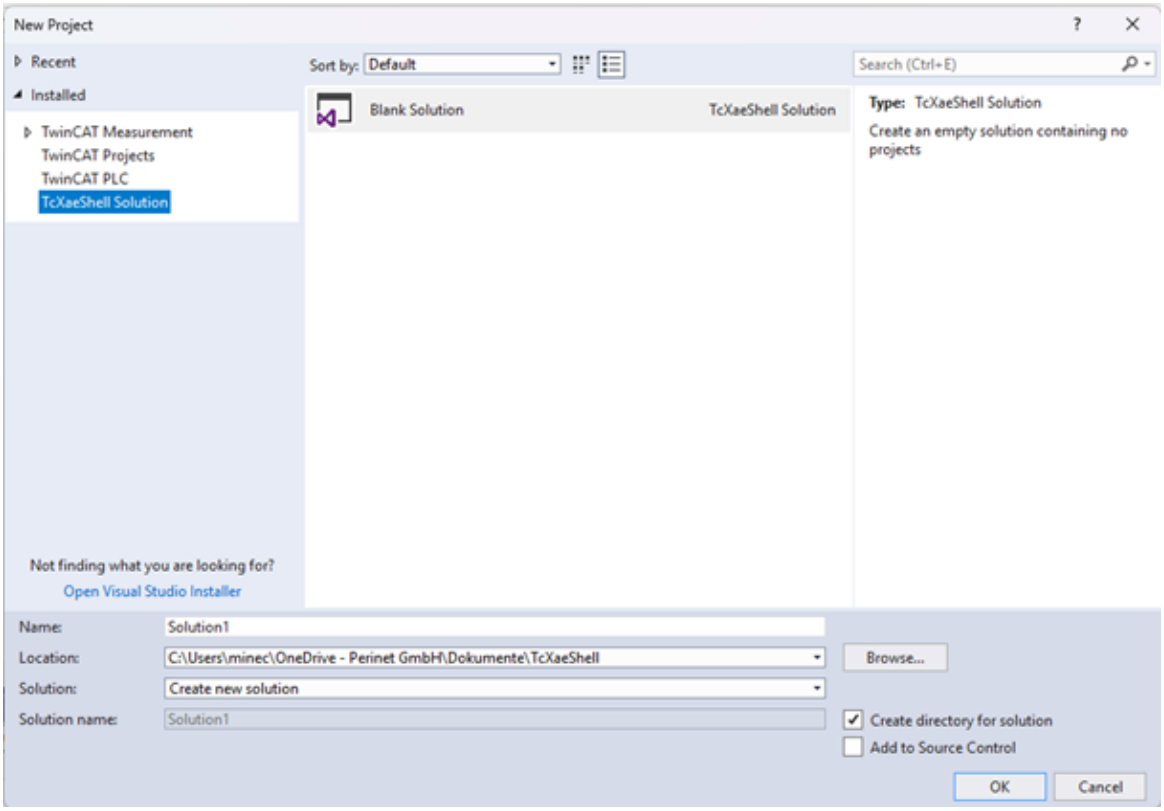

Figure 4: Create a blank solution

2. Then you need to add to this solution a new project from File -> Add -> New Project -> TwinCAT Projects to create a TWINCAT XAE Project as seen in Figure 5. On the solution explorer on the left side of the program, you can see the PLC and I/O dropdown menus that have already been created as seen in Figure 6

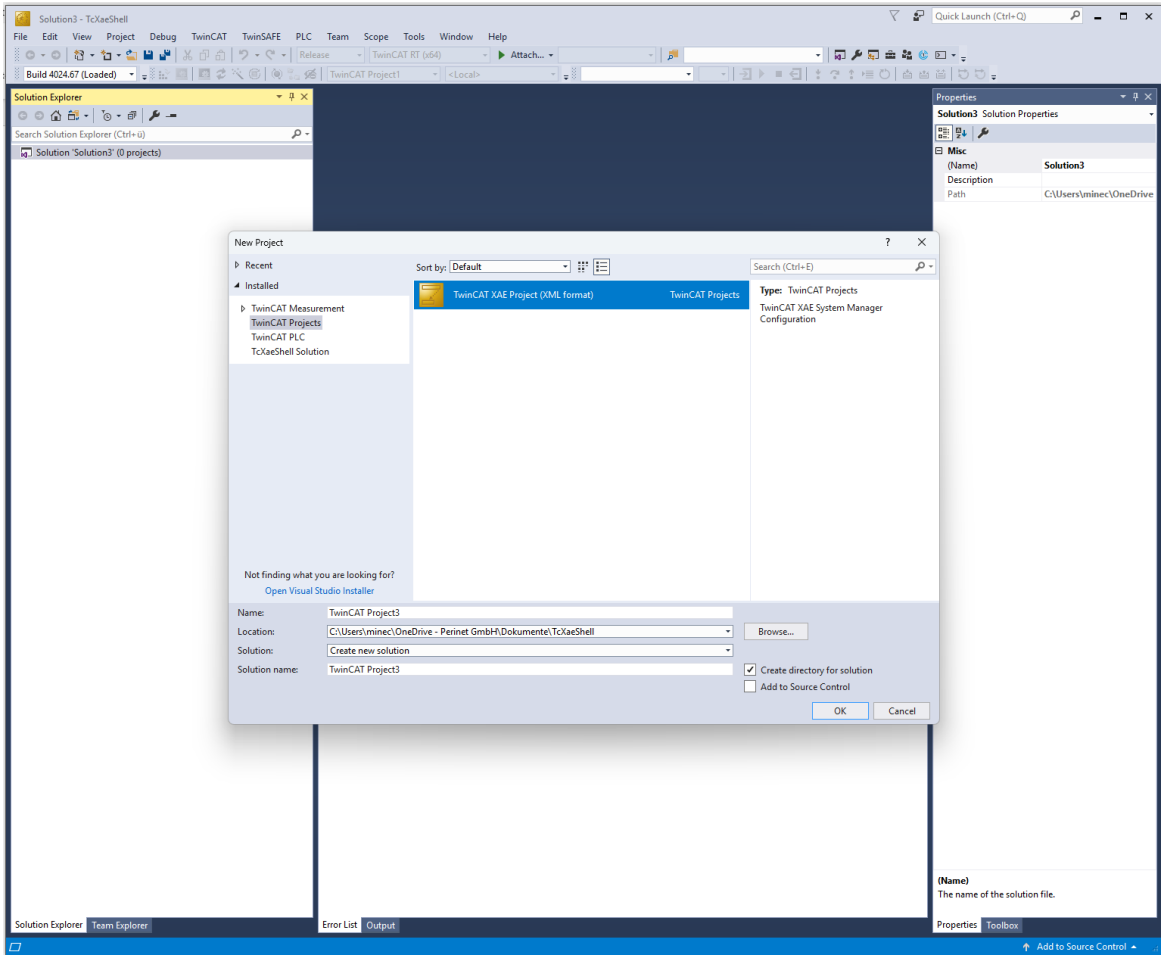

Figure 5: Create a TwinCAT project

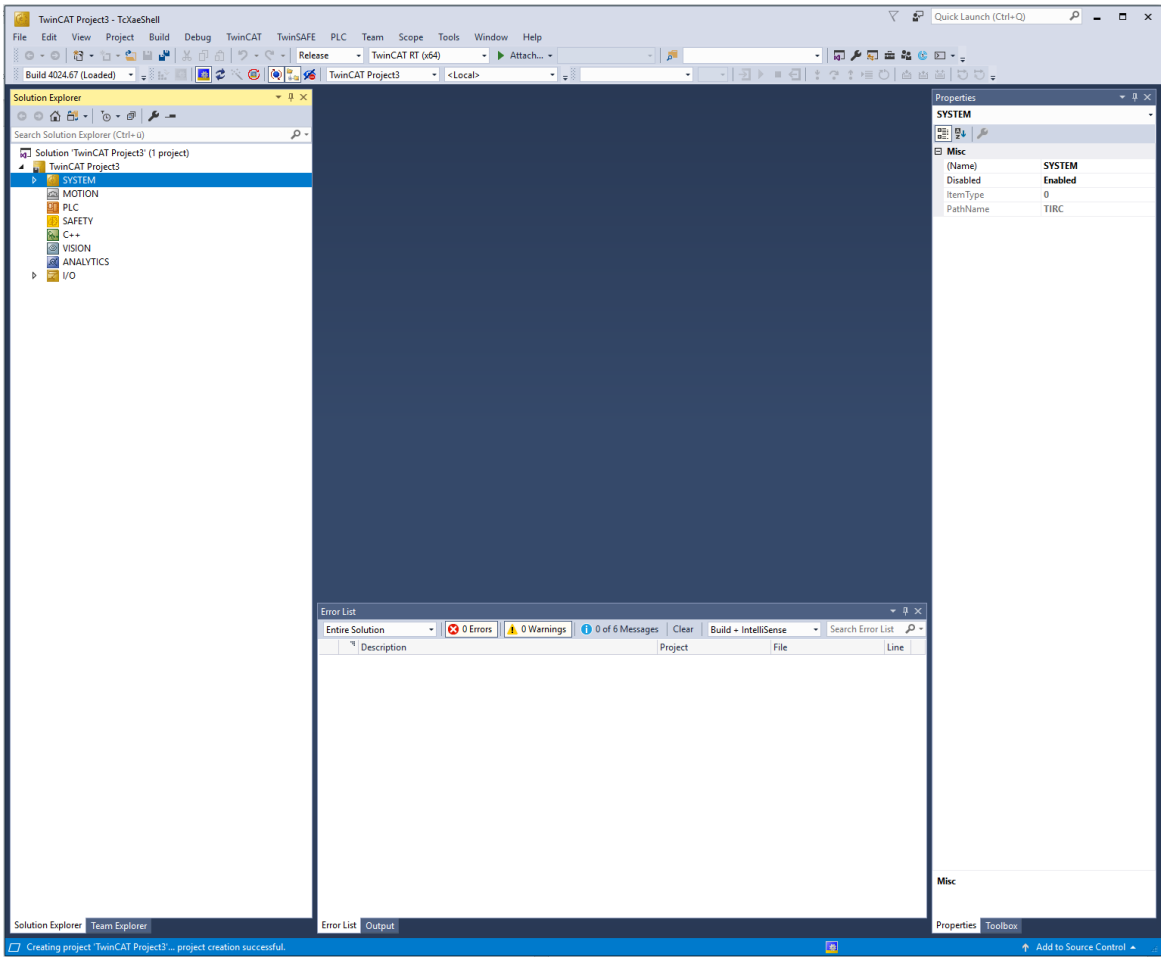

Figure 6: PLC and I/O sections on the Solution Explorer

3. Now, you need to right-click on the PLC on the Solution Explorer to add an “Empty PLC Project” template for library projects as seen in Figure 7

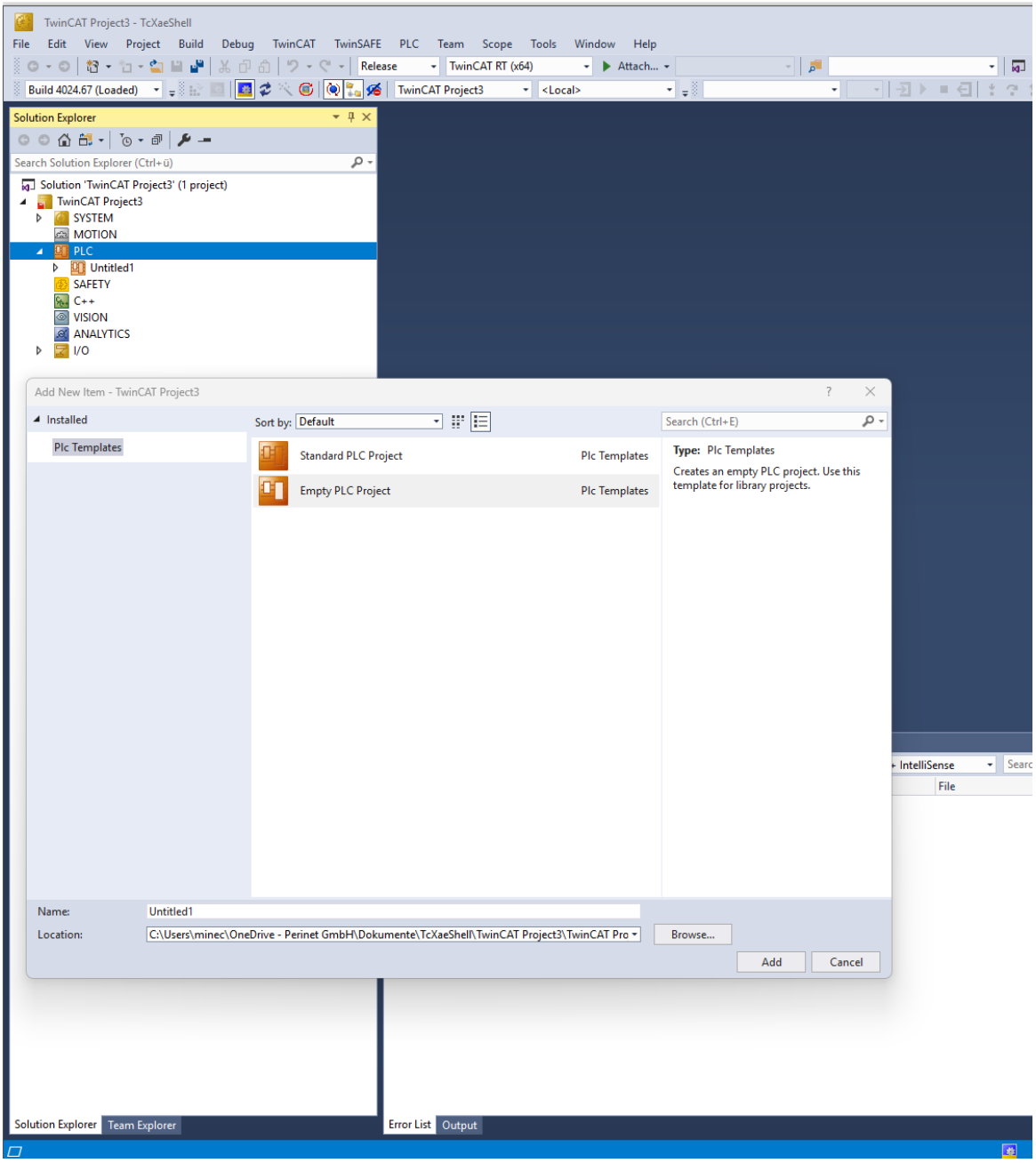

Figure 7: Create a PLC Project

4. Under this added PLC Project, you need to navigate to the References from the drop-down list and right-click and from “Add library” option, find the “System” submenu and add one by one the 3 libraries: Tc2-Standard, Tc2-System, Tc3-Module as seen in Figure 8

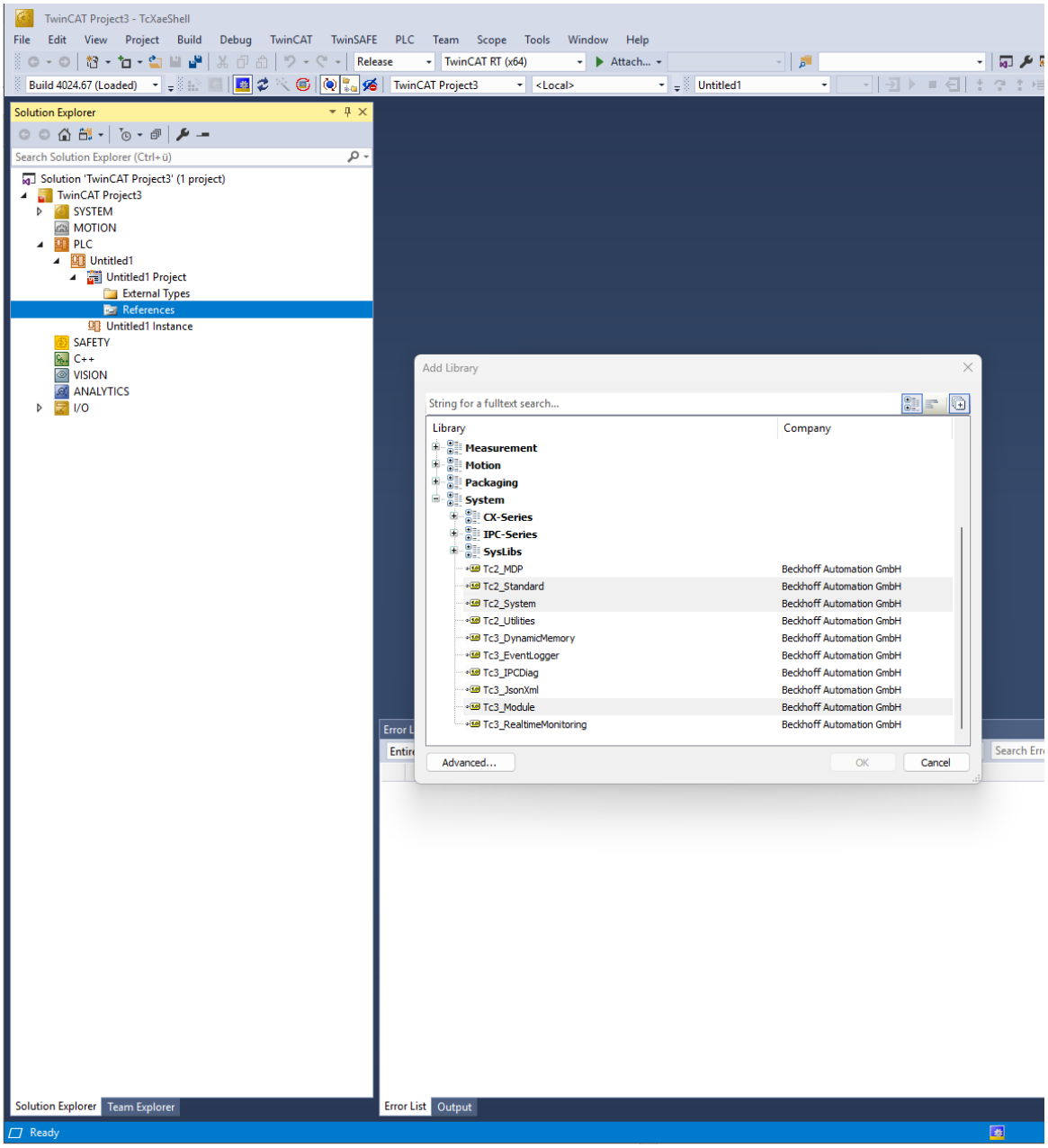

Figure 8: Add PLC libraries

5. In this step, you need to add 2 program definitions under the POUs folder to extract the PLC sensor data, namely MAIN and WriteData. First, right click on the Project under PLC and create a new folder named POUs. Then, right-click on the Project under PLC and create the MAIN and WriteData PRG files as seen in Figure 9

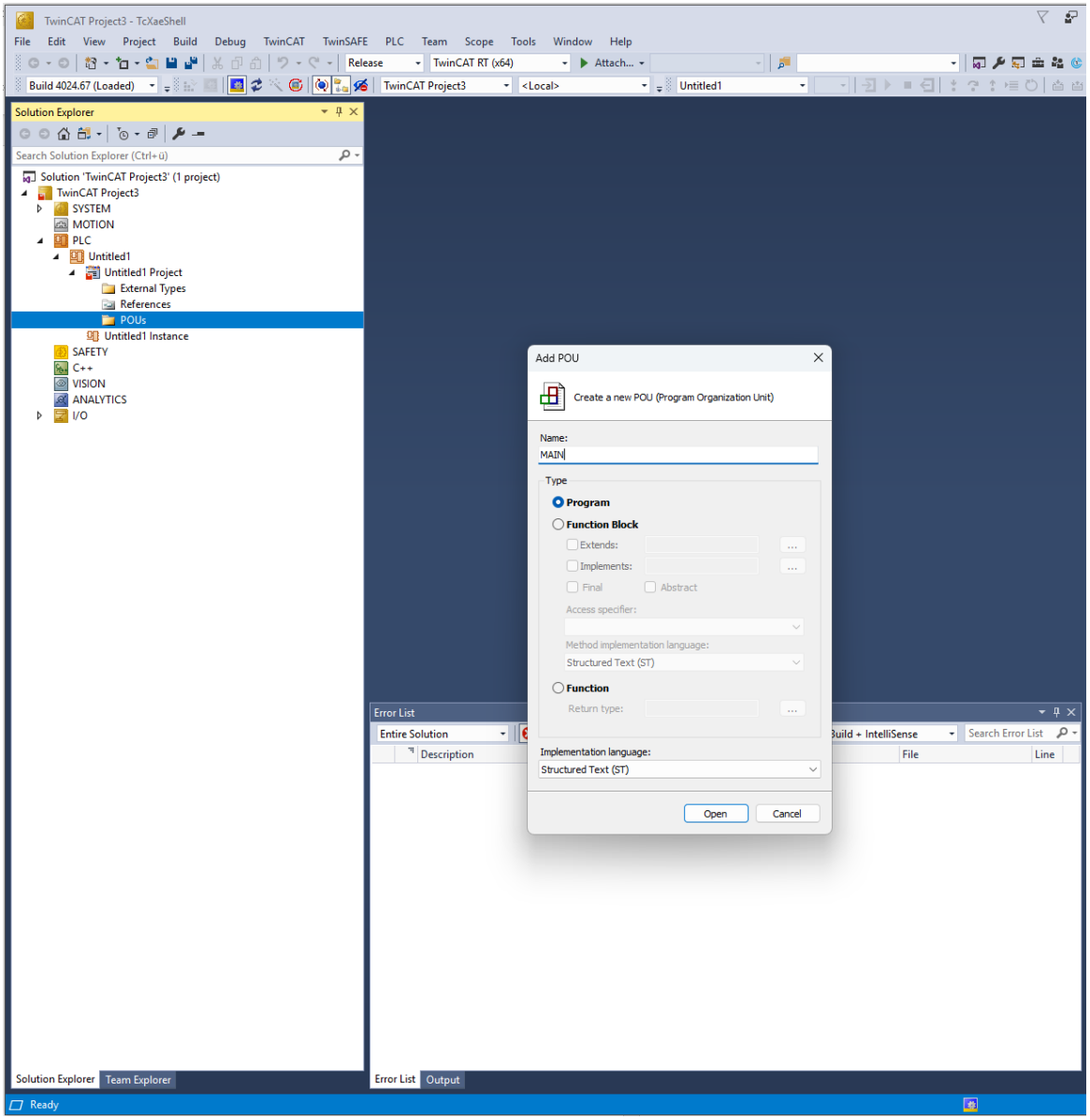

Figure 9: Add MAIN and WriteData PRG files

6. The MAIN.PRG file should be in the following format:

Listing 1: `MAIN.PRG`

```
PROGRAM MAIN
VAR
    Underrange AT%I*    : BOOL;
    Overrange AT%I*     : BOOL;
    Error AT%I*     : BOOL;
    Signal AT%I*     : INT;
END_VAR
VAR_INPUT
    ResetAndStart : BOOL;
    WriteToCsv : BOOL;
```

```
                END_VAR
                VAR
                    Minimum : INT;
                    Maximum : INT;
                    Delta : INT;
                    Data : ARRAY[0..300000] OF INT;
                    Cycle : ARRAY[0..300000] OF UDINT;
                    Tme : ARRAY[0..300000] OF LINT;
                    index : UDINT :=0;
                    UnderrangeDetected : BOOL;
                    OverrangeDetected : BOOL;
                    ErrorDetected : BOOL;
                END_VAR
                IF ResetAndStart THEN
                    MEMSET(destAddr := ADR(Data),0,SIZEOF(Data));
                    MEMSET(destAddr := ADR(Cycle),0,SIZEOF(Cycle));
                    MEMSET(destAddr := ADR(Tme),0,SIZEOF(Tme));
                    Minimum := Signal;
                    Maximum := Signal;
                    index := 0;
                    Delta := 0;
                    ResetAndStart := FALSE;
                    UnderrangeDetected := OverrangeDetected := ErrorDetected :=
FALSE;
                END_IF

                IF Minimum > Signal THEN
                    Minimum := Signal;
                    Delta := Maximum - Minimum;
                END_IF

                IF Maximum < Signal THEN
                    Maximum := Signal;
                    Delta := Maximum - Minimum;
                END_IF


                IF index < 50000 AND index >=0 THEN
                    Data[index] := Signal;
                    Cycle[index] := ._TaskInfo[1].CycleCount;
                    Tme[index] := ._TaskInfo[1].DcTaskTime;
                    index := index +1;
                END_IF

                WriteData();

                IF Underrange THEN UnderrangeDetected := TRUE; END_IF
                IF Overrange THEN OverrangeDetected := TRUE; END_IF
                IF Error THEN ErrorDetected := TRUE; END_IF
```

7. The WriteData.PRG file should be in the following format:

Listing 2: `WriteData.PRG`

```
VAR
    AmsNetIdRemoteHost : STRING := '192.168.188.222.1.1';
    iFB_FileOpen : FB_FileOpen;
    iFB_FilePuts : FB_FilePuts;
    iFB_FileClose : FB_FileClose;
    Step : UDINT;
    index : UDINT;
    Value : T_MAxString;
END_VAR
iFB_FileOpen(
    sNetId:= AmsNetIdRemoteHost,
    sPathName:= 'C:\Users\minec\plcdata.txt',
    nMode:= FOPEN_MODEWRITE OR FOPEN_MODETEXT,
    ePath:= ,
    bExecute:= ,
    tTimeout:= ,
    bBusy=> ,
    bError=> ,
    nErrId=> ,
    hFile=> );

iFB_FilePuts(
    sNetId:= AmsNetIdRemoteHost,
    hFile:= iFB_FileOpen.hFile,
    sLine:= ,
    bExecute:= ,
    tTimeout:= ,
    bBusy=> ,
    bError=> ,
    nErrId=> );

iFB_FileClose(
    sNetId:= AmsNetIdRemoteHost,
    hFile:= iFB_FileOpen.hFile,
    bExecute:= ,
    tTimeout:= ,
    bBusy=> ,
    bError=> ,
    nErrId=> );

CASE Step OF

0: IF Main.WriteToCsv THEN
    index := 0;
    iFB_FileOpen.bExecute := TRUE;
    IF iFB_FileOpen.bBusy THEN
        iFB_FileOpen.bExecute := FALSE;
        Step := 100;
    END_IF
END_IF

100:
    IF NOT iFB_FileOpen.bBusy THEN
        Step := 200;
    END_IF
```

```
200:
    IF Main.Cycle[index] = 0 THEN
        Main.WriteToCsv := FALSE;
        Step := 300;
    ELSE

    Value := UDINT_TO_STRING(Main.Cycle[index]);
    Value := CONCAT(Value,',');
    Value := CONCAT(Value,LINT_TO_STRING(Main.Tme[index]));
    Value := CONCAT(Value,',');
    Value := CONCAT(Value, INT_TO_STRING(Main.Data[index]));
    Value := CONCAT(Value, '$N');

    IF index+1 < 300000 AND Main.Cycle[index+1] <> 0 THEN
        Value := CONCAT(Value,UDINT_TO_STRING(Main.Cycle[index+1]));
        Value := CONCAT(Value,',');
        Value := CONCAT(Value,LINT_TO_STRING(Main.Tme[index+1]));
        Value := CONCAT(Value,',');
        Value := CONCAT(Value, INT_TO_STRING(Main.Data[index+1]));
        Value := CONCAT(Value, '$N');
    END_IF

    iFB_FilePuts.sLine := Value;
    iFB_FilePuts.bExecute := TRUE;
    IF iFB_FilePuts.bBusy THEN
        iFB_FilePuts.bExecute := FALSE;
        Step := 210;
    END_IF
    END_IF

210:
    IF NOT iFB_FilePuts.bBusy THEN
        IF index < 300000 THEN
            index := index +2;
            Step := 200;
        ELSE
            Step := 300;
            Main.WriteToCsv := FALSE;
        END_IF
    END_IF

300:
    iFB_FileClose.bExecute := TRUE;
    IF iFB_FileClose.bBusy THEN
        iFB_FileClose.bExecute := FALSE;
        Step := 0;
    END_IF


END_CASE
```

8. You need to assign these two files as PLC tasks. To add one as a PLC Task, right-click on

the project under the PLC again and select “PLC Task” from the “Add Referenced Task” window. Under this PLC Task, you will assign it to the already created MAIN POU. To do this, right-click on PLC Task - > Add existing Item -> Find your MAIN Program and click OK as seen in Figure 10 and repeat the same for WriteData task.

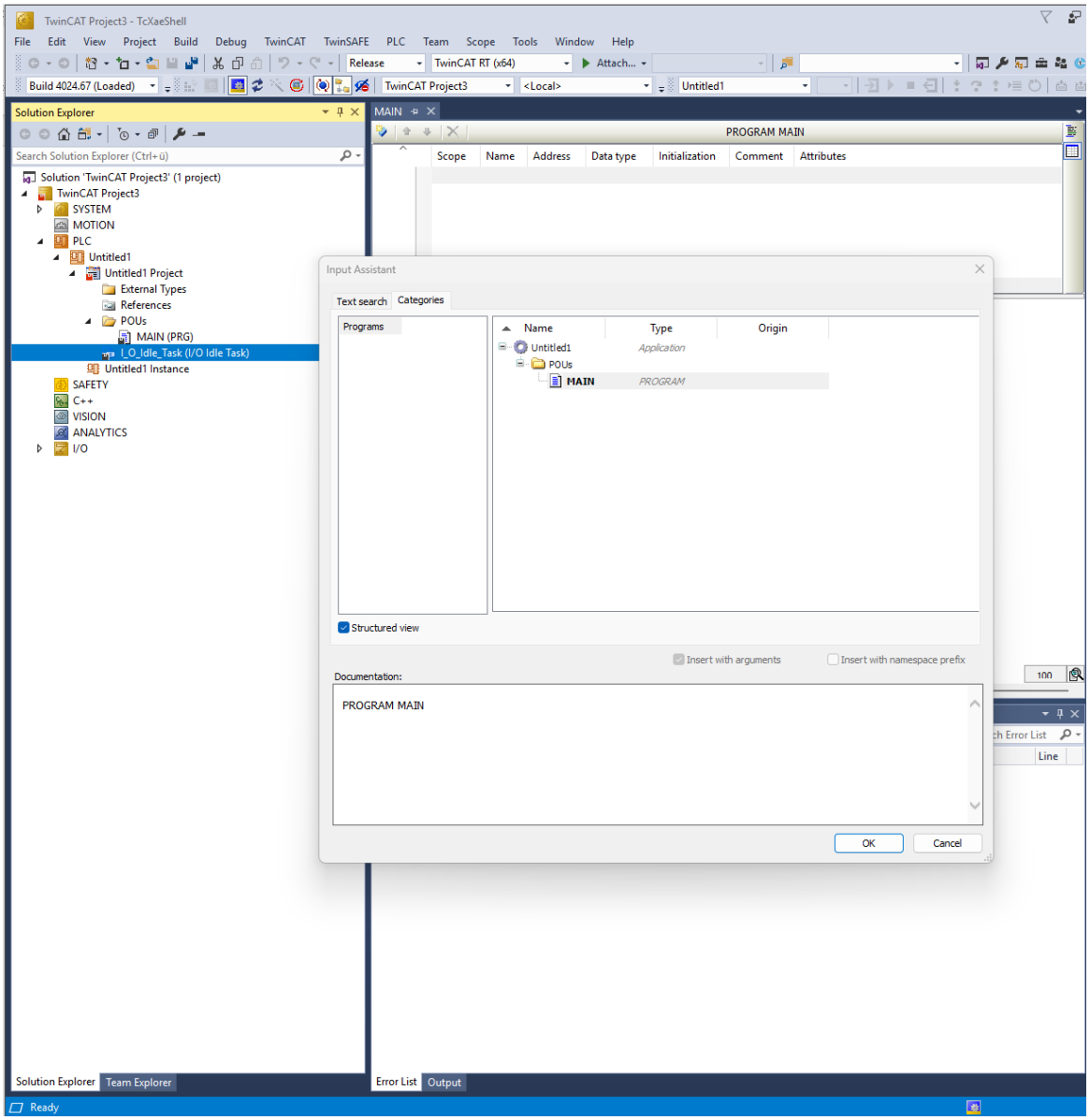

Figure 10: Add MAIN and WriteData as PLC Tasks

9. Now, the EtherCAT Master device should be added under I/O Devices section. Right-click on the Devices and select Insert Device and add the EtherCAT Master as seen in Figure 11

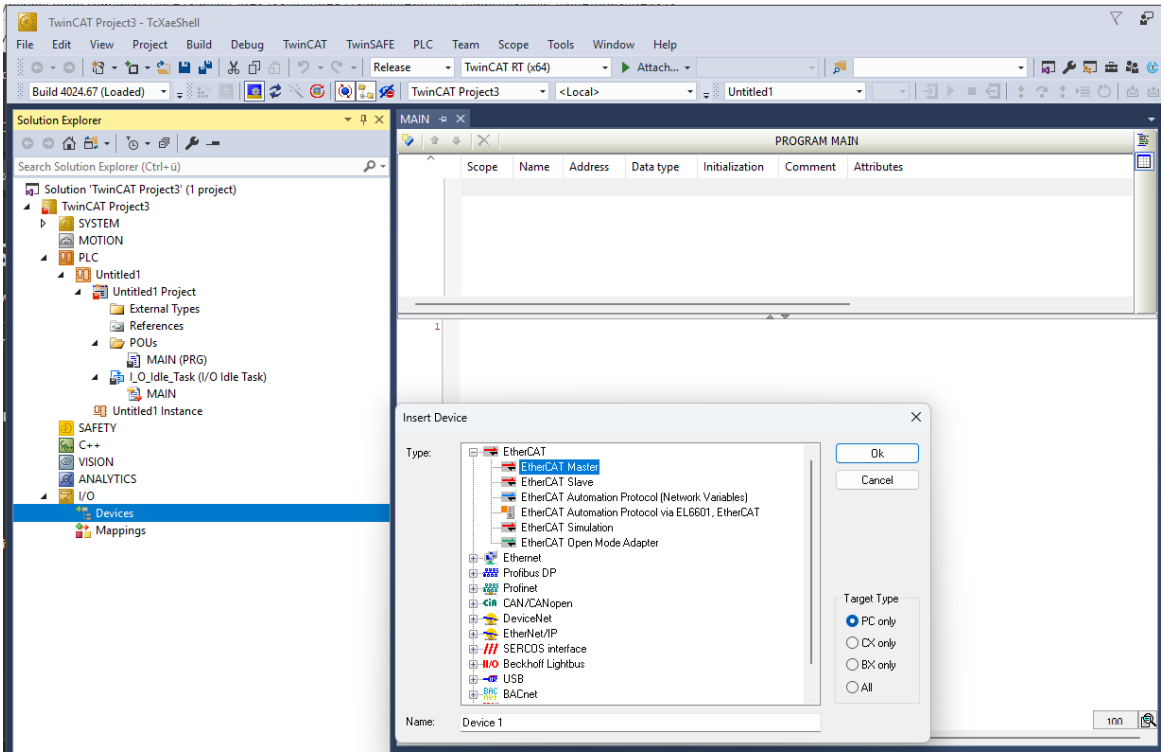

Figure 11: Insert Device as EtherCAT Master

10. After making the physical connections of the system, to establish a connection from the development environment to the controller, the AMS Net ID set on the controller is required. From the menu, where you see <Local>, click on it to “Choose Target System”. Select Search (Ethernet) if the device is not listed and click OK once the device is found as seen in Figure 12 Click on Yes if the Update of NetIds of I/O Devices or changing platform solution due to target platform difference is asked. You should be able to add the route successfully and then see the CP-454A88 reachable on the target system tab as highlighted yellow in Figure 13

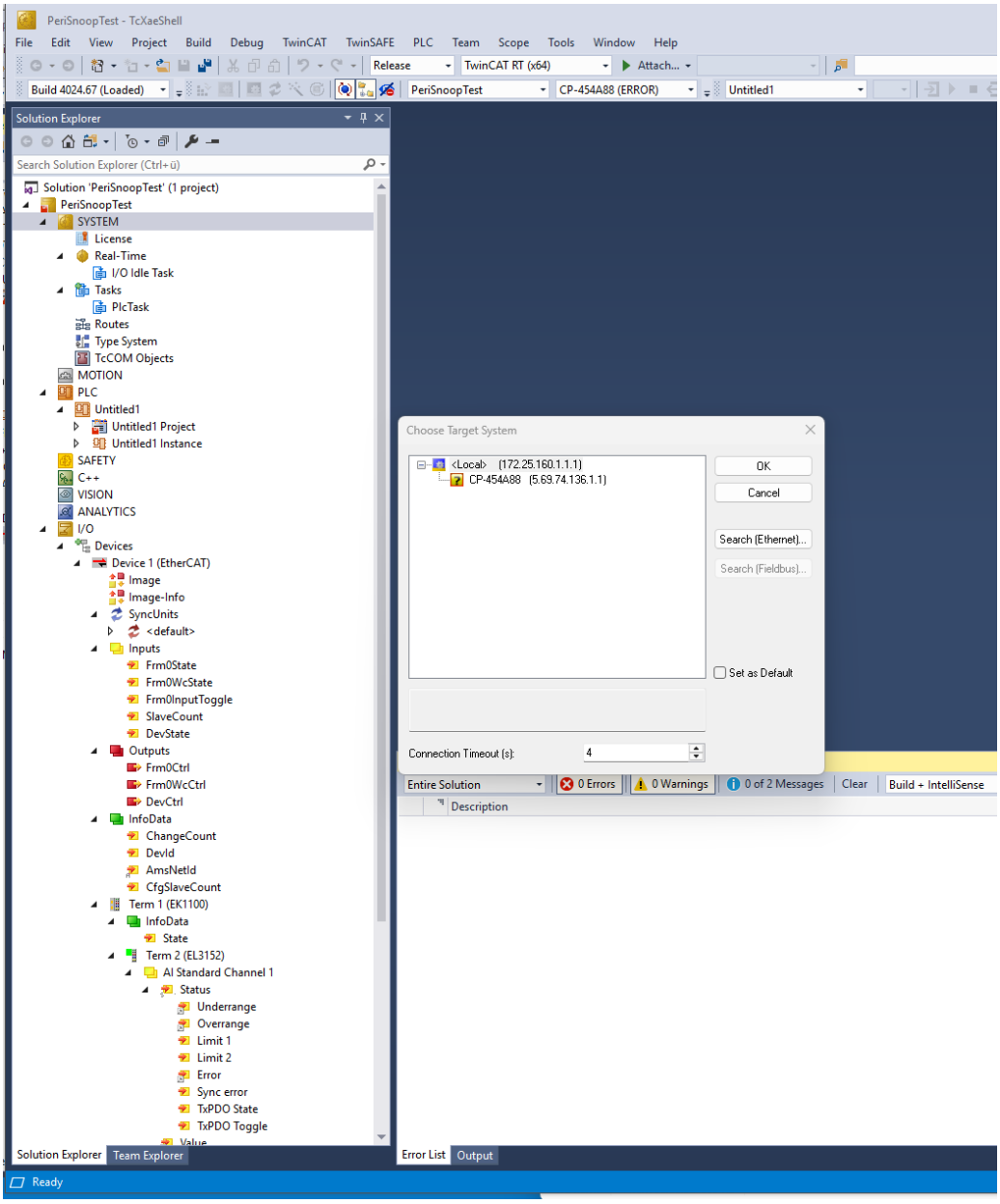

Figure 12: Find Target Controller

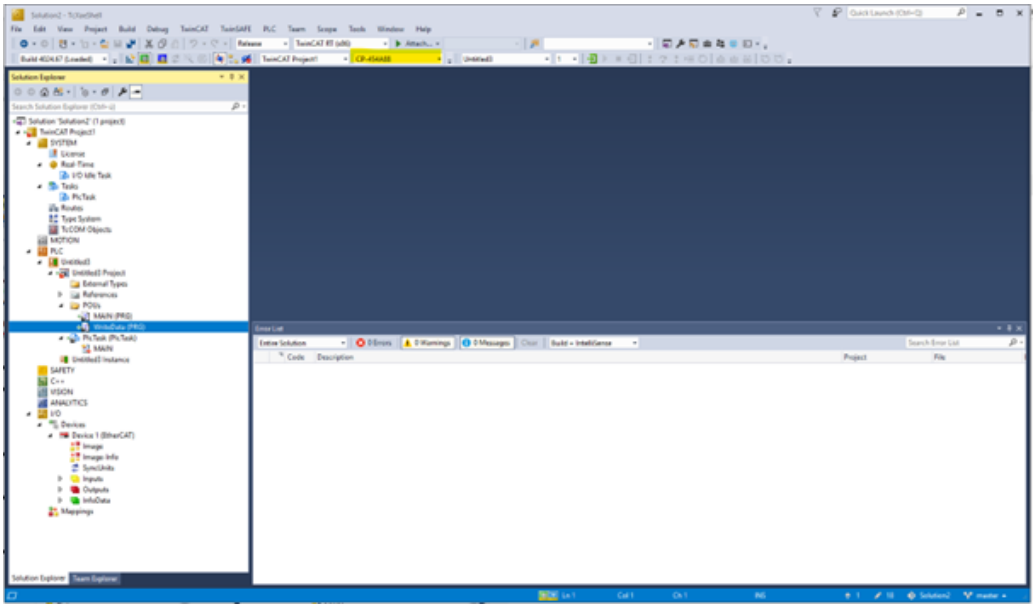

Figure 13: Select the CP-454A88 Target Controller

11. To load the project to the controller, click the “Activate Configuration” button as marked in Figure 14.

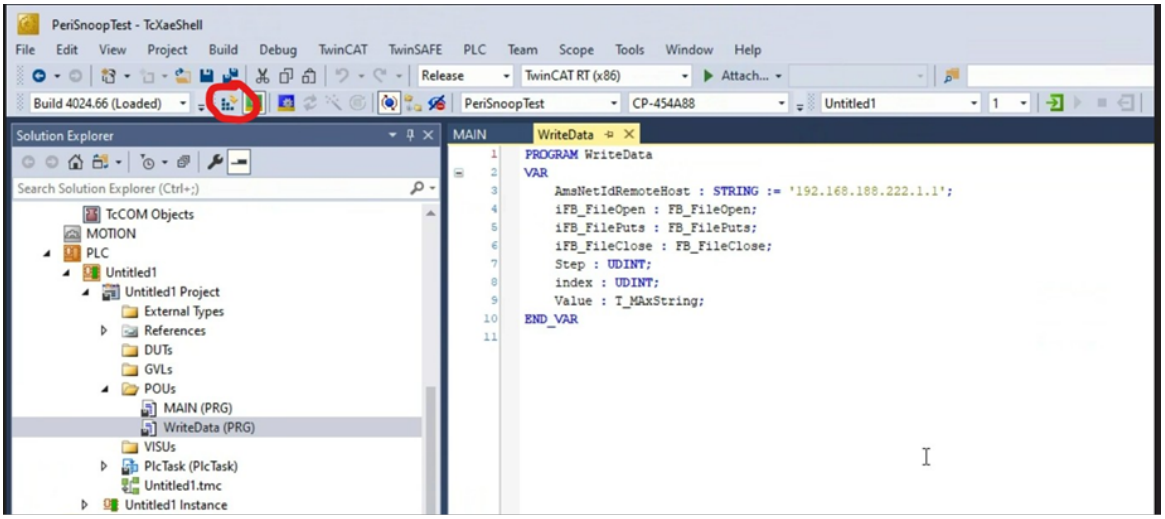

Figure 14: Load the Project to the Controller

12. Finally, switch to the running program view by clicking on the Login button as marked in Figure 15.

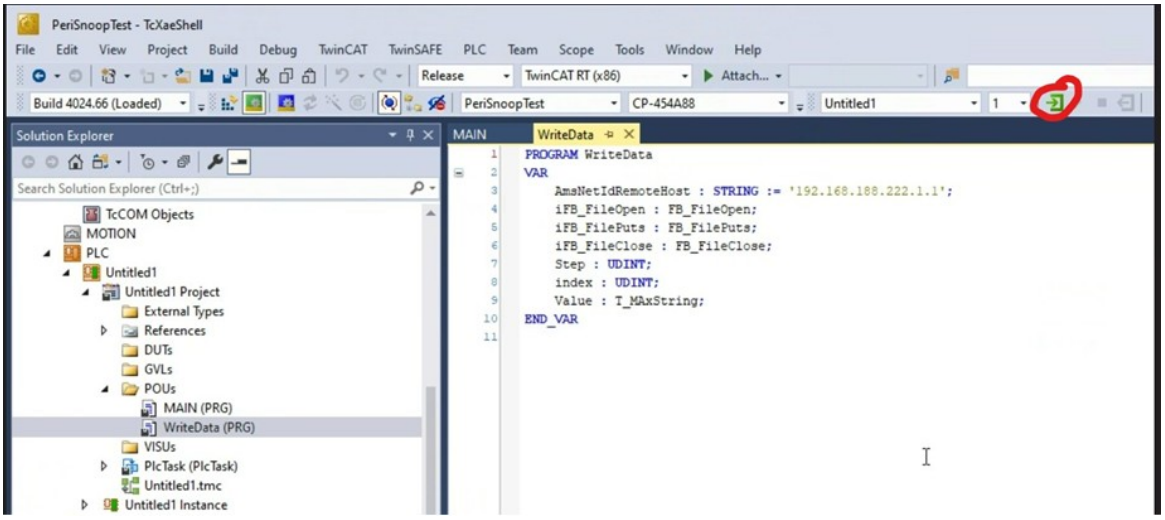

Figure 15: Load the Running View by Login

5 Recording, Storing and Interpreting a Measurement Series with the PLC

A measurement series consists of the cycle numbers, the timestamps and the measured values. For each cycle, the cycle number, its timestamp and the measured value are stored in an array as seen in Listing 3

Listing 3: `Measurement series with cycle numbers, timestamps and values measured`

```
3435174993,80218456041325000,16366
3435174994,80218456041625000,16407
3435174995,80218456041650000,16419
3435174996,80218456041675000,16344
3435174997,80218456041700000,16367
3435174998,80218456041725000,16337
3435174999,80218456041750000,16341
3435175000,80218456041775000,16381
3435175001,80218456041800000,16459
3435175002,80218456041825000,16366
3435175003,80218456041850000,16300
3435175004,80218456041875000,16301
3435175005,80218456041900000,16336
3435175006,80218456041925000,16379
3435175007,80218456041950000,16417
3435175008,80218456041975000,16375
3435175009,80218456042000000,16340
3435175010,80218456042025000,16386
3435175011,80218456042050000,16355
3435175012,80218456042075000,16354
3435175013,80218456042100000,16359
3435175014,80218456042125000,16379
...
```

The cycle number is a consecutive integer and represents the cycle and the distance between two cycles is always 1. The counter may overflow. A cycle's timestamp is a logical timestamp that is constant within a cycle and between two cycles matches the PLC's cycle time. The timestamp's unit is nanoseconds. The timestamp may overflow. In this setup, 50000 consecutive input values of the signal are stored in the memory.This limit can be adjusted and must lie below the array limits, which can also be increased. The actual duration of the recording depends on the cycle time, where in this case a cycle time of 1 ms, the measurement series would span 50 seconds. The values "UnderrangeDetected" and "OverrangeDetected" are linked to the input terminal and in the case of UnderrangeDetected, it indicates a drop below the lower limit (4 mA).

Saving a measurement series to the file system is decoupled from recording the series in RAM, because accessing the file system during recording would not be possible within the required cycle times.

1. To record a measurement series, set the Boolean variable "ResetAndStart" to TRUE by clicking on the "Prepared value" column and to store it to the PLC select the marked

button as seen in Figure 16.

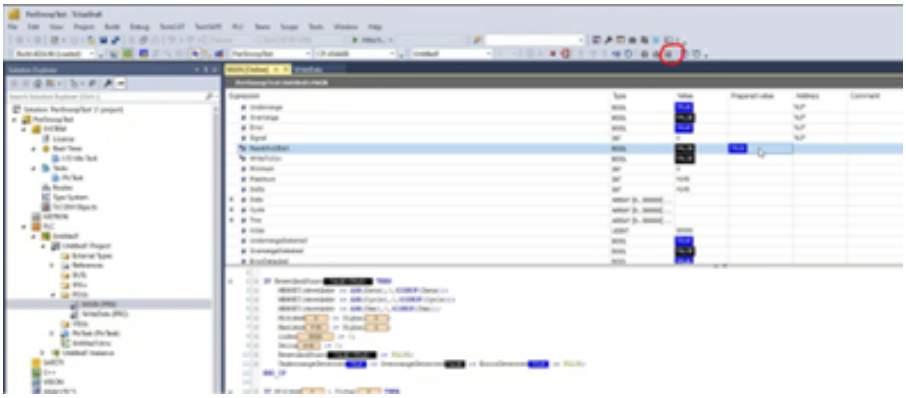

Figure 16: Record the measurement series

2. To save the measurement to a file, set the Boolean variable “WriteToCsv” to TRUE by clicking on the “Prepared value” column, which is one line below the ”ResetandStart variable as seen in Figure 16. The data is not saved to the controller, but instead to the development PC connected. Therefore, the AMS Net ID of the development PC must be stored in the controller’s program as seen in Figure 17 The path and file name of the destination on the development PC can be adjusted at the function block iFB_FileOpen as seen in Figure 18

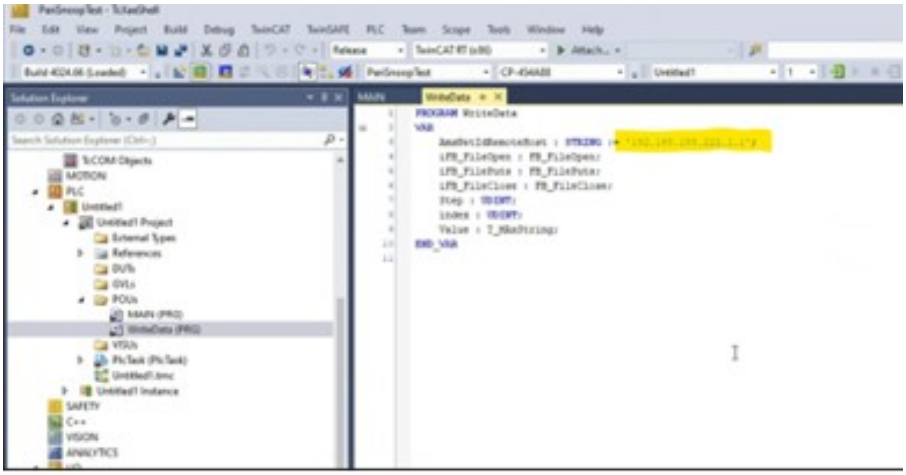

Figure 17: Saving the measurement series - AMS Net ID of the PC

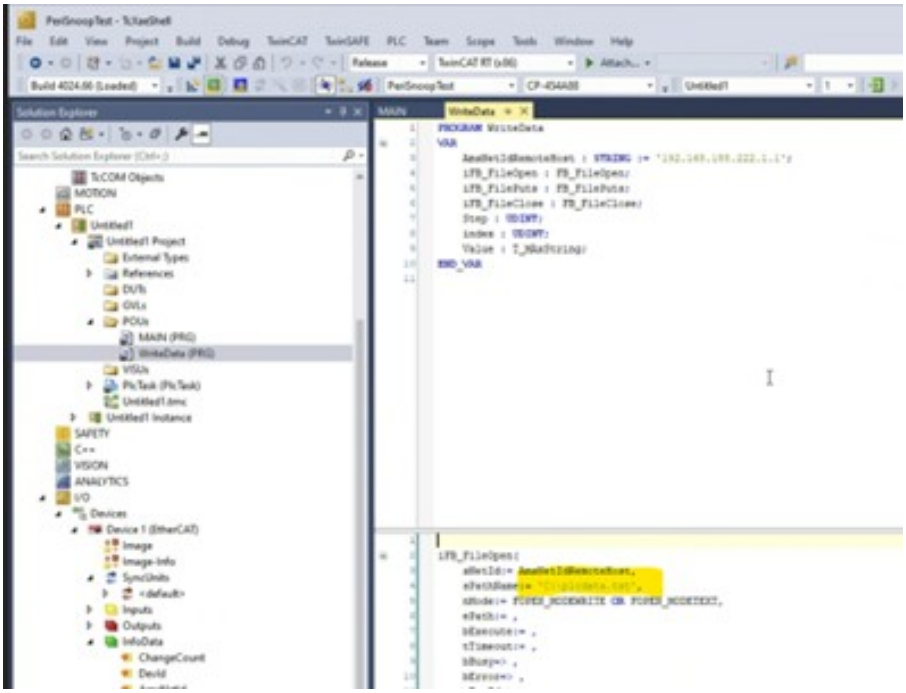

Figure 18: Saving the measurement series - File storage location on the PC

3. At the final step, displaying and interpreting the measurement series will take place. The user can use gnuplot to view the sample measurement series collected and see the plot similar to the one given in Figure 19

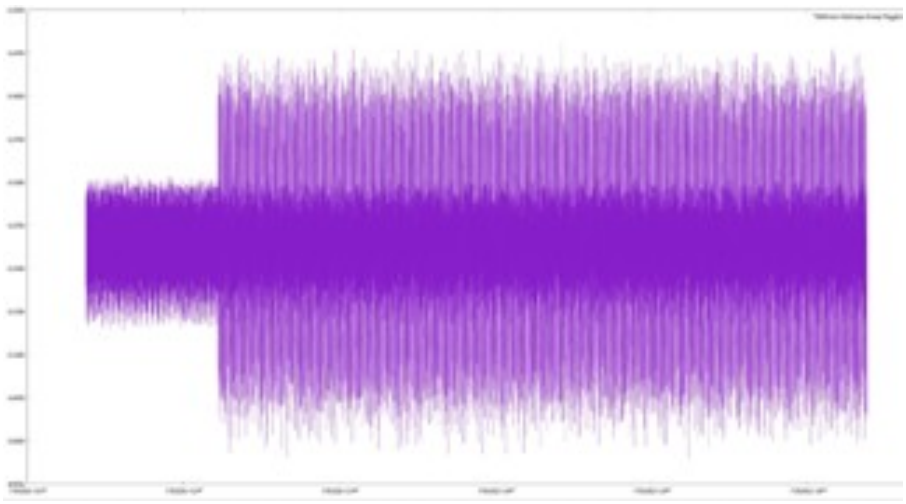

Figure 19: Displaying the measurement series - sample GNU plot

The analogue input signal is represented at the terminal as a 16-bit integer value. A measured value less than or equal to 4 mA corresponds to the integer representation 0 and a measured value that is greater than or equal to 20 mA corresponds to 32768. The current can be calculated with the following formula, where int denotes the integer value:

$$Current(int) = ((intx16)/32768) + 4mA \quad (1)$$

6 Ordering Information

Ordering Code	Product Name	Description
PRN.000.069	periSNOOP 4-20mA	Single Pair Ethernet 4-20mA based sensor monitor

Ordering Code	Product Name	Description
PRN.000.008	periSWITCH 3-port	3-port hybrid SPE switch
PRN.000.003	periSTART standard	Media converter from SPE to standard Ethernet
PRN.000.017	periLINE 0.2m	Hybrid SPE cable with M8 connectors
PRN.000.025	periMICA	Open modular edge computer
PRN.000.020	periCORE Development Kit	Full featured firmware development setup

7 Contact & Support

For customer support, please call us at **+49 30 863 206 701** or send an e-mail to *support@perinet.io*.

For complete contact information visit us at www.perinet.io

A List of Figures

B References

[1] autosen Automation and Sensors. autosen AU016 Ultrasonic diffuse reflective sensor. URL: https://media.autosen.com/Shopsystem/Assets/en/AU011-AU018_manual.pdf (visited on 06/25/2025).

[2] Perinet GmbH. periLINE basic hybrid SPE cable. URL: https://perinet.io/en/products/accessories/periline-basic (visited on 09/16/2025).

[3] Perinet GmbH. periSNOOP 4-20mA Datasheet. PRN.100.699. https://docs.perinet.io/PRN100699-periSNOOP4-20mADatasheet.pdf.

[4] Perinet GmbH. periSNOOP 4-20mA Standard Product Summary. PRN.100.695. https://docs.perinet.io/PRN100695-periSNOOP4-20mAProductSummary.pdf.

[5] Perinet GmbH. periSTART SPE Media Converter. URL: https://perinet.io/en/products/smart-components/periswitch (visited on 09/16/2025).

[6] Perinet GmbH. periSWITCH 3-Port SPE Network Switch. URL: https://perinet.io/en/products/smart-components/periswitch (visited on 09/16/2025).

[7] Beckhoff Automation GmbH & Co. KG. C6015 Industrial PC Manual. URL: https://download.beckhoff.com/download/Document/ipc/industrial-pc/C6015_EN.pdf (visited on 06/25/2025).

[8] Beckhoff Automation GmbH & Co. KG. EK110x-00xx, EK15xx EtherCAT Bus Coupler. URL: https://download.beckhoff.com/download/Document/io/ethercat-terminals/ek110x_ek15xx_en.pdf (visited on 06/25/2025).

[9] Beckhoff Automation GmbH & Co. KG. EL31xx Analog Input Terminals. URL: https://download.beckhoff.com/download/Document/io/ethercat-terminals/el31xx_en.pdf (visited on 06/25/2025).

[10] Beckhoff Automation GmbH & Co. KG. TwinCAT 3 eXtended Automation Engineering. URL: https://www.beckhoff.com/de-de/support/downloadfinder/suchergebnis/?download_group=97028248 (visited on 06/25/2025).

C Revision History

Revision	Date	Author(s)	Description
1	2025-10-20	mcet	Initial Release