OPC-UA Integration Within Glass Industries

1st Arya Karimi Jafari
Hochschule Hamm-Lippstadt
Lippstadt, Germany
arya.karimi-jafari@stud.hshl.de

2nd Muhammaad Mustafa Qaiser
Hochschule Hamm-Lippstadt
Lippstadt, Germany
muhammad-mustafa.qaiser@stud.hshl.de

3rd Omar Abdellatif
Hochschule Hamm-Lippstadt
Lippstadt, Germany
omar.abdellatif@stud.hshl.de

***Abstract*—The automated glass industries are going through an accelerated transition toward data-driven manufacturing. Therefore, relying on different brand-specific networks that make data sharing slow and difficult. Open Platform Communications-Unified Architecture (OPC-UA) is a single, neutral standard that can connect to machines within the production line to the cloud and higher-level systems, while keeping the data secure. This paper dives into the details and characteristics of OPC-UA highlighted through the real-life example of implementing OPC-UA into an "Annealing Lehr". Using time-sensitive networking (TSN), we achieve 1ms data updates, fast enough for precise motion and control over the systems connected. Recent studies show that adopting a standardized OPC-UA server reduces integration effort, reduces engineering effort, and improves data quality. These findings confirm OPC-UA allows for strong security and timing demands of the glass industry.**

I. Introduction

Over the past two decades, a push towards implementing **IoT** protocols and **Industry 4.0** has highlighted a weakness in many production environments. **OPC-UA** addresses these issues by combining machine to machine communication services, cloud information storing model, and built in cyber-security in a single architectural standard. At the application level, this allows the customer to be exposed to structured address space of objects, variables, and methods, allowing the client to connect their **Programmable Logic Controllers** (PLCs) to cloud analytics, in order to browse and consume data, as well as make changes based on their desired outcome.

II. Application Areas

Although glass industries benefit from OPC-UA's real-time data exchange and security, many other industries and systems also leverage OPC-UA for interoperability, data consolidation, and digital transformation. Notable application areas include automotive and discrete manufacturing, building automation and smart infrastructure. Modern automotive assembly lines often combine PLCs, servo drives, and vision-based inspection systems. OPC-UA's structured address space allows robots to expose their joint positions, status flags, and diagnostics as standardized "objects" that higher-level MES (Manufacturing Execution Systems) or SCADA (Supervisory Control and Data Acquisition) solutions can consume. Due to OPC-UA's platform independence, and not strict hardware and software requirements, many different clients can implement OPC-UA into their business, whether it being a smart factory, or a startup [1], [2].

III. Key Features

OPC-UA stands on six technical pillars: **secure communication**, **rich information modelling**, **automatic discovery**, **platform independence**, and **extensibility**. Every server or session, regardless of it being client / server or Pub / Sub, is authenticated, signed and encrypted if required, removing any risk found in many field buses [3]. Furthermore, there are different forms of communicating with the OPC server, methods include TCP for local requirements, and HTTPS when crossing firewalls. Finally, OPC-UA accepts all forms of data types, allowing the user to define the data type of their choice, reading any form of sensor. In section VI, Figure 5 depicts temperature values being defined as "real", while Figure 10 highlights other forms of data types including string, and boolean data types.

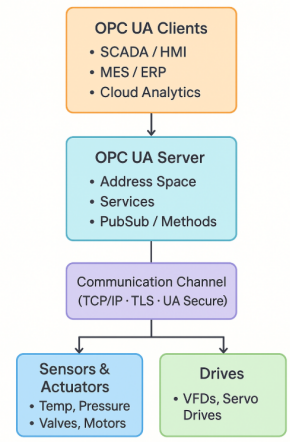

Fig. 1. OPC-UA Block Diagram

IV. Hardware and Software

In order to implement OPC-UA onto a device, a demanding industrial PC is not required. In contrast, field trials show that 32-bit microcontrollers are also enough to run OPC-UA servers [4]. The requirements in order to run an OPC-UA server rely on smart hardware, and solid software. Modern PLCs such as Siemens and Schneider Electric have enough memory and CPU power to host an embedded OPC-UA Server. An important detail to note is that each hardware with OPC needs to have a separate IP address, therefore at times relying on external components such as Ewon modules

which can be used as network bridges in order to assign a separate IP address to each hardware component [5]. As for the software requirements, it truly depends on the hardware component that is being used. Siemens has an integrated software system called *TIA Portal* and within this software that is used for programming the Siemens PLCs and HMIs, there is also integrated OPC-UA server configuration, which allows the user to easily configure and create a remote access server. However, OPC-UA is rich in openly available stacks that cover every major programming language. For C / C++, the *open62541* project delivers a fully featured client / server [6]. Java programmers work with *Eclipse Milo*, which offers high-performance client and server SDKs [7]. Overall, there are many software applications which allow the developer to program an OPC-UA server either from scratch, or with the help of a pre-existing configuration tab in applications like TIA Portal of Siemens.

V. Implementing OPC-UA

As previously mentioned, the glass industry has been developing and implementing IoT protocols to achieve Industry 4.0 standards. In this paper, a detailed explanation as to how the OPC-UA server was created, and implemented will be described. As well as revealing how the PLC, OPC-UA Server, and the Human Machine Interface (HMI) are linked in order to communicate with each other and perform the correct adjustments made on the server. The client required a method of being able to securely store data that is found on the PLC, as well as be able to make adjustments from the cloud in order to reduce effort and improve efficiency. In order to complete such a task, some key characteristics had to be considered. Security, Communication, and Parameter Limits have to be taken into consideration when programming and creating the OPC-UA Server.

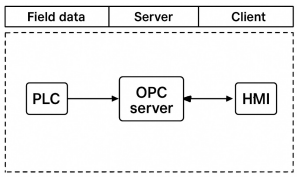

Fig. 2. Communication path between PLC, OPC Server, and HMI

VI. Data Collection

In order to program and implement the OPC-Server in a pre-existing Annealing Lehr program, many different parameters had to be understood and considered. For this project, a Schneider Electric LMC Pacdrive programmable logic controller was used to provide the field data to the OPC-UA Server. To program a Schneider Electric PLC, SoMachine was used to program and create the OPC-Server. In addition, all variables that were required to cause an impact onto the Schneider Electric Human Machine Interface were collected via the software Vijeo Designer, depicting crucial information such as the number of zones for the Annealing lehr, the number of different burners, location of flaps/exhaust fans, and the machine belt motor. Gathering this information as well as the name of each variable responsible for making adjustments to these parameters were noted for creating a connection between the PLC, OPC Server and the HMI. Figure 3 depicts the screen which shows all the required information regarding the variables from Vijeo Designer, by clicking on each item found within the attached screen, the variable which controls that specific field can be found and used within the code to form communication between the PLC, HMI, and OPC-Server. Upon collecting all necessary data, a separate folder to hold

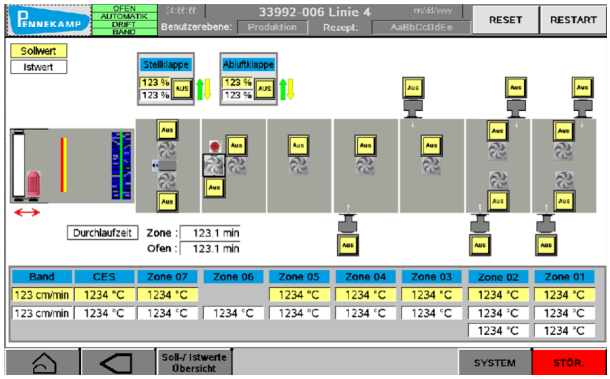

Fig. 3. Vijeo Designer Screen

all the information regarding the OPC-UA server was created. Within this folder, a subfolder containing structs was created to group multiple related variables, as visible in Figure 4.

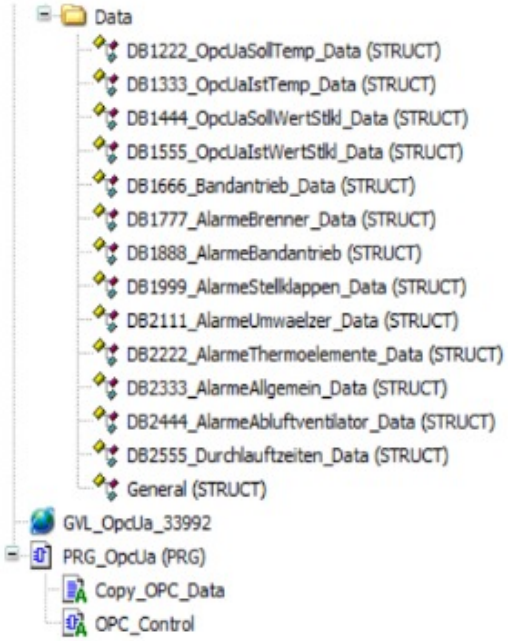

Fig. 4. Data Struct Blocks

Within the struct blocks, related variables were declared, once for the HMI, and once for the OPC Server. Figure 5 dives deeper into what is found within the struct blocks. Upon creating the structs, a Data Block (DB) was cre-

ated for each struct as a way to nest the structure, allowing more accurate calling functions when programming. In addition, the same Data Blocks were used as the Global Variables that were available in the OPC-UA Server. For example, Figure 4 displays the substructure "Data" that holds the **DB1222-OpcUaSollTemp-Data** information, allowing the **Temperaturen-Sollwert** Struct to hold a more complex data structure. This approach organizes data hierarchically and helps modular programming.

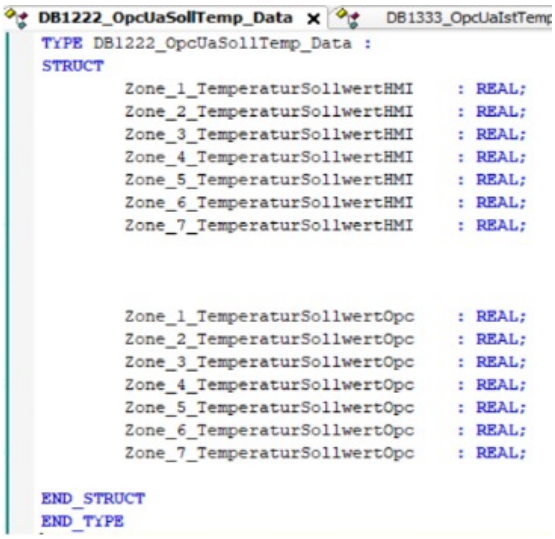

Fig. 5. Within the Data Blocks

VII. Security System

As OPC-UA has an integrated security system for data transfer from the field data, client and server. An external and additional security system was considered to control who is able to make changes to the PLC and HMI variables via the OPC-Server. A simple, yet effective security system that would allow the user to only make changes once the valid Username and Password were entered. Figure 6 presents the code that was used to create a password protected server for the customer.

```
IF iOpcUaUser = 'PKP' AND iOpcUaPassword = '1234' AND TrigLogin.Q THEN

xLogin:=TRUE;

UserlLoggedIn := TRUE;

 ELSE

UserlLoggedIn := FALSE;

END_IF
```

Fig. 6.Remote Access Server Security System

The variables presented are also the variables that were visible on the OPC Server as depicted in Figure 7. The operator would have to first enter the Username and Password on the remote server, and if the correct credentials were used, the "xLogin", and "UserLoggedIn" variables would turn true, allowing communication between the server and the HMI.

#	Server	Node Id	Display Name	Value
1	OpcServer@172...	NS2|String|...	st_OpcUaPassword	1234
2	OpcServer@172...	NS2|String|...	st_OpcUaUser	PKP
3	OpcServer@172...	NS2|String|...	xLogin	true

Fig. 7. Remote Access Server Security System

VIII. Variable Communication

The connection forming code that was written in order to ensure communication between the remote access server and the HMI consisted of IF loops that contained 4 parameters. This included checking whether the HMI value is different to the OPC value with the use of the Not Equal To operation, a boolean flag in the form of the variable "UserLoggedIn", an AND operation ensuring both condition are true before the block is executed, as well as a limit boundary set in order for the parameter to stay within a safe and regulated range. For example, a regular beverage bottle requires less temperature across the heating zones compared to a larger bottle for wine or champagne.

As previously mentioned, the OPC-UA Server can also be used in order to monitor any faults or alarms that are occurring in the production line. For that, a separate section of code was written in order to display alarm messages. Figure 8 shows the base programming section used for the different alarms, varying in text, and variables. It is also important to note that these segments of code had to be written for over 150 parameters, variables and alarms.

```
390  //############################  Alarme Thermoelemente
391  // Thermoelement 1.1
392  //Alarm 1
393  IF HMI.G2Diag_astZ01TE[1].Drahtbruch OR i_xTest THEN
394      Thermoelement1_1_Alarm1 := ('Drahtbruch');
395  ELSE
396      Thermoelement1_1_Alarm1 := ('');
397  END_IF
398  //Alarm 2
399  IF HMI.G2Diag_astZ01TE[1].GrenzeWertebereich OR i_xTest THEN
400      Thermoelement1_1_Alarm2 := ('Ungueltiger Wert');
401  ELSE
402      Thermoelement1_1_Alarm2 := ('');
403  END_IF
```

Fig. 8. Sample Alarm Segment of Code

Furthermore, in order to specify which variables are found in the OPC-UA Server, they had to be in the form of Global Variables. Figure 9 depicts the Global Variables of the written program, who consist of the data blocks discussed previously and visible in Figure 5.

Another purpose for the Global Variables is that they can be used in other sections of the program, deeming extremely useful when trying to form further connections outside of the OPC-UA folder. Certain connections in the Machine-Tasks sections of the program require continuity between shared variables, and with the use of Global Variables, an easier

connection can be made between the HMI variables and OPC variables outside of the OPC-UA function blocks.

GVL_OpcUa_33992

```
{attribute 'qualified_only'}
VAR_GLOBAL
    DB1222_OpcUaSollTemp          : Temperaturen_Sollwert;
    DB1333_OpcUaIstTemp           : Temperaturen_Istwert;
    DB1444_OpcUaSollWertStlkl     : Stellklappe_Sollwert;
    DB1555_OpcUaIstWertStlkl      : Stellklappe_Istwert;
    DB1666_Bandantrieb            : Bandantrieb_Para;
    DB1777_AlarmeBrenner          : Alarme_Brenner;
    DB1888_AlarmeBandantrieb      : Alarme_Bandantrieb;
    DB1999_AlarmeStellklappen     : Alarme_Stellklappen;
    DB2111_AlarmeUmwaelzer        : Alarme_Umwaelzer;
    DB2222_AlarmeThermoelemente   : Alarme_Thermoelemente;
    DB2333_AlarmeAllgemein        : Alarme_Allgemein;
    DB2444_AlarmeAbluftventilator : Alarme_Abluftventilator;
    DB2555_Durchlauftzeiten       : Durchlauftzeiten;
    General                       : General;

END_VAR
```

Fig. 9. OPC-UA Global Variables

IX. Completed OPC-UA Server

In order to visualize and test the OPC-UA Server, the software application **UA-Expert** was used. A powerful OPC client developed by Unified Automation, UA-Expert is widely used for testing, monitoring, and debugging OPC-UA servers.

By using the IP address of the PLC, as well as the OPC gateway code, one can add a server to their UA-Expert project, in which all the desired variables will be available as "Nodes". The customer is then able to add all their preferred Nodes onto the main screen, and edit them accordingly. For added security, Nodes resembling a Username and Password can be used to only allow parameterizing to certain levels of management. Furthermore, time at which a certain variable has been changed is also saved under the "Server Timestamp" column. Figure 10 depicts the OPC-UA server created and formed in the UA-Expert software, portraying the different security nodes, and also the data variables which are parametrized.

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp
1	OpcServer@172...	NS2|String|...	st_OpcUaPassword	1234	String	11:59:25.480 AM	11:59:25.480 AM
2	OpcServer@172...	NS2|String|...	st_OpcUaUser	PKP	String	11:59:22.730 AM	11:59:22.730 AM
3	OpcServer@172...	NS2|String|...	xLogin	true	Boolean	11:59:20.230 AM	11:59:20.230 AM
4	OpcServer@172...	NS2|String|...	xLogout	false	Boolean	11:59:17.230 AM	11:59:17.230 AM
5	OpcServer@172...	NS2|String|...	Zone_1_TemperaturSollwertHMI	200	Float	12:01:54.730 PM	12:01:54.730 PM
6	OpcServer@172...	NS2|String|...	Zone_1_TemperaturSollwertOpc	200	Float	12:01:54.730 PM	12:01:54.730 PM
7	OpcServer@172...	NS2|String|...	Zone_2_TemperaturSollwertHMI	650	Float	12:02:00.480 PM	12:02:00.480 PM
8	OpcServer@172...	NS2|String|...	Zone_2_TemperaturSollwertOpc	700	Float	12:06:22.981 PM	12:06:22.981 PM
9	OpcServer@172...	NS2|String|...	Zone_3_TemperaturSollwertHMI	0	Float	12:06:03.481 PM	12:06:03.481 PM
10	OpcServer@172...	NS2|String|...	Zone_3_TemperaturSollwertOpc	-100	Float	12:06:03.481 PM	12:06:03.481 PM
11	OpcServer@172...	NS2|String|...	Zone_4_TemperaturSollwertHMI	450	Float	12:06:08.981 PM	12:06:08.981 PM
12	OpcServer@172...	NS2|String|...	Zone_4_TemperaturSollwertOpc	450	Float	12:06:08.981 PM	12:06:08.981 PM
13	OpcServer@172...	NS2|String|...	Zone_5_TemperaturSollwertHMI	579	Float	12:06:16.231 PM	12:06:16.231 PM
14	OpcServer@172...	NS2|String|...	Zone_5_TemperaturSollwertOpc	579	Float	12:06:16.231 PM	12:06:16.231 PM
15	OpcServer@172...	NS2|String|...	Zone_7_TemperaturSollwertHMI	80	Float	12:06:37.981 PM	12:06:37.981 PM
16	OpcServer@172...	NS2|String|...	Zone_7_TemperaturSollwertOpc	80	Float	12:06:37.981 PM	12:06:37.981 PM
17	OpcServer@172...	NS2|String|...	Zone_6_StellklappeSollwertHMI	100	Float	12:06:52.231 PM	12:06:52.231 PM
18	OpcServer@172...	NS2|String|...	Zone_6_StellklappeSollwertOpc	150	Float	12:06:52.231 PM	12:06:52.231 PM
19	OpcServer@172...	NS2|String|...	Zone_7_StellklappeSollwertHMI	80	Float	12:06:57.231 PM	12:06:57.231 PM
20	OpcServer@172...	NS2|String|...	Zone_7_StellklappeSollwertOpc	80	Float	12:06:57.231 PM	12:06:57.231 PM

Fig. 10. UA-Expert Server View

As previously mentioned, to form a connection between an HMI variable and OPC variable, four different components were used within each line of code. Connecting the HMI Variable to the OPC Variable, Security, and finally putting limits in order to keep the range of the variables within a safe and productive range. The user should not be able to input temperature values above 650 degrees as it could not only deform the bottle, but also impact energy saving and cost efficiency. Similarly, the user should not be able to input values less than 0, as that could cause problem within the burner configuration and overall setup of the annealing Lehr. Figure 11 provides an insight of testing the limits induced by the program, and see whether the UA-Expert Server can oblige to the limits and restrict the user from inputting values outside of the given range.

5	OpcServer@172...	NS2|String|...	Zone_1_TemperaturSollwertHMI	200
6	OpcServer@172...	NS2|String|...	Zone_1_TemperaturSollwertOpc	200
7	OpcServer@172...	NS2|String|...	Zone_2_TemperaturSollwertHMI	650
8	OpcServer@172...	NS2|String|...	Zone_2_TemperaturSollwertOpc	700
9	OpcServer@172...	NS2|String|...	Zone_3_TemperaturSollwertHMI	0
10	OpcServer@172...	NS2|String|...	Zone_3_TemperaturSollwertOpc	-100
11	OpcServer@172...	NS2|String|...	Zone_4_TemperaturSollwertHMI	450
12	OpcServer@172...	NS2|String|...	Zone_4_TemperaturSollwertOpc	450

Fig. 11. Parameter Limitations

As shown in Figure 11, when a value greater than 650 is inputted as the OPC Value, the HMI value will only be able to process the greatest value found within the limitations, which in this case is 650. The same methodology is applied when a number below 0 is entered as the OPC Value, the HMI variable will only be able to go to the minimum value possible, which is 0.

X. Recap of Implemented Work

The shared real life example of implementing OPC-UA into a running Annealing Lehr program provides meaningful insight as to how a professional, and industrial OPC-Server can be created, highlighting its strength in terms of security, scalability, and data analysis. This example focuses solely on the glass industry, and more specifically Annealing Lehrs, but with the rapid growth of interest in cloud computing, and remote access servers, OPC-UA deems as a very powerful and helpful industrial communication standard, as it can be used for data monitoring, checking for alarms / errors, and also applying changes to different parameters only with authorized access.

XI. OPC-UA: Benefits and Challenges

OPC-UA has a lot of benefits which have also been mentioned throughout this paper. **Decentralization** is a key advantage of OPC-UA, as it means this industrial communication method is flexible in terms of data modeling structures in a mesh network. OPC-UA achieves this by defining consistent data structure that all components use. In addition, its **platform independence**, is also a major benefit, as it means

industrial systems can seamlessly integrate software from any vendor using any operating system. OPC-UA can be implemented on embedded systems and in the cloud. Furthermore, **scalability**, is also a strong asset of OPC-UA as it makes the communication protocol future proof. It enables organizations to develop scalable SCADA systems in order for the existing plant equipment can integrate without additional configuration. Finally, **interoperability** is another great feature of OPC-UA as it allows the end users to build custom industrial systems using devices and software from different vendors, and as mentioned in section IV, the hardware and software requirements for OPC-UA are not too complicated.

However, there are also some challenges that need to be considered when trying to implement OPC-UA. The high complexity and implementation is the biggest challenge of OPC-UA as reports have raised an issue when trying to mix multiple different vendors and varied data structures. Furthermore, other more general challenges such as security issues at times, performance related problems in terms of low latency, and infrastructure requirements, can all be challenges that need to be studied before wanting to implement an OPC-UA protocol. All information regarding the benefits and challenges of OPC-UA have come from [1].

XII. Why OPC-UA?

This section of the paper will compare OPC-UA to other industrial communication methods such as MQTT and MODBUS TCP. Although the protocol being used highly depends on the application of the client, OPC-UA has great benefits over these two communication methods. Since OPC-UA is known for its interoperability, it can support both client-server, and publisher-subscriber models. On the other hand, MQTT, is a lightweight messaging protocol mainly used for IoT. In contrast to OPC-UA, it lacks the advanced data structure and security algorithms [8].

Modbus TCP and OPC-UA serve similar purposes but differ in terms of how the objective is reached. Modbus TCP is a simple and open protocol, making it easier to implement than OPC-UA, but similar to MQTT, it lacks in the security features and is not designed for real time communication. OPC-UA's advanced protocol makes it more suitable for critical applications like smart factories and the automotive industry [9].

XIII. Conclusion

In this study, it was made clear how OPC-UA can be integrated into an existing annealing lehr program to create a secure, data rich and cloud ready environment. By mapping vital PLC, and HMI variables into a structured web server, the client was able to monitor and control all important parameters which can improve their production. In addition, a role-based access setup was created for added security through authorized logins. The final version of the OPC-UA server was tested via UA-Expert, where both the client and developer are able to check whether the security, and limits introduced were functioning appropriately. These results confirm that OPC-UA offers the glass industry, and other comparable manufacturing sectors a practical path to industry 4.0. Its native security, flexibility, and transport options reduce the integration efforts while safeguarding operations. OPC-UA's scalability in terms of hardware and software also allow enterprises to protect future investments. Implementing this industrial communication method therefore not only improves current plant connectivity, but also positions facilities to embrace advanced embedded analytics and remote services with confidence.

References

[1] Paessler, "What is OPC UA? Definition and Details," Web article, Paessler IT-Explained, 2025, accessed: Jun. 12, 2025. [Online]. Available: https://www.paessler.com/it-explained/opc-ua

[2] Vismec. (2023) Opc ua protocol: All about it and the benefits for your business. Accessed: 2025-06-04. [Online]. Available: https://www.vismec.com/en/news/opc-ua-protocol/

[3] O. Foundation. (2024) Opc 10000-1: Ua part 1 – overview and concepts, version 1.05. Accessed: 2025-06-11. [Online]. Available: https://reference.opcfoundation.org/Core/Part1/v105/docs/

[4] M. OPC. (2025) Matrikon flex opc ua embedded server software development kit (sdk). Accessed: 2025-06-11. [Online]. Available: https://www.matrikonopc.com/opc-ua/embedded/sdk.aspx

[5] H. Networks. (2025) Ewon cosy$^+$ ethernet – industrial vpn gateway (item ec71330-00ma). Accessed: 2025-06-11. [Online]. Available: https://www.hms-networks.com/p/ec71330-00ma-ewon-cosy-ethernet

[6] open62541 Project. (2025) open62541: Open source opc ua implementation. Accessed: 2025-06-11. [Online]. Available: https://www.open62541.org/

[7] E. Foundation. (2025) Eclipse milo – an open source implementation of opc ua (iec 62541). Accessed: 2025-06-11. [Online]. Available: https://github.com/eclipse-milo/milo

[8] TEEPTRAK. (2025) Opc ua: Optimized protocol for industrial communication. Accessed: 2025-06-04. [Online]. Available: https://teeptrak.com/en/2025/03/08/opc-ua-protocol-optimization-for-industrial-communication/

[9] Vessel Automation, "Communication protocol of a future: Opc-ua or modbus?" Web article, Vessel Automation, 2023, accessed: Jun. 12, 2025. [Online]. Available: https://vesselautomation.com/communication-protocol-of-a-future-opc-ua-or-modbus/