A Multi-Robot Platform based on OPC UA

To Viet Dung, Duong Nguyen Khang, Nguyen Hoang Giap
Dept of Control Engineering and Automation
Ho Chi Minh City University of Technology (HCMUT), VNU-HCM
Ho Chi Minh City, Vietnam
Email: {dung.tobk_2k1, khang.duong061101, nhgiap}@hcmut.edu.vn

***Abstract*—Robot interoperability is a critical challenge faced by users in the manufacturing industry despite the fact that robotics companies provide extensive product lines and services. This is primarily due to the dependence on third-party communication standards or the use of proprietary standards. In this paper, we present an OPC UA platform that addresses this issue by providing a unified architecture for connecting all devices in a robot system. The objective of our research is to build an optimized robot management system utilizing OPC UA that encompasses various components such as robots, controllers, sensors, server computers, client computers, and other related elements. Our project involves a laboratory robot system that employs the EtherCAT Master Robot Controller. The purpose of this system is to improve productivity and efficiency in factories by synchronizing and updating robot data, simulating robot behavior before implementation to ensure safety and efficiency, and allowing direct control of the robot by users through user-adjustable commands.**

***Index Terms*—IIoT, inteoperability, OPC UA, robot system**

I. Introduction

In recent years, the manufacturing industry has witnessed a paradigm shift towards Industry 4.0. One of the major challenges of this connective era applications in manufacturing is the lack of interoperability among different robots and devices from different manufacturers. For instance, Mitsubishi at the field-layer uses CC Link IE Field Protocol, Yaskawa using Profinet and Mechatrolink, while ABB and KUKA also have their own unique architectures. Realizing that issue, in 2011, ABB developed the OPC Server for the IRC5 Controller to provide an alternative way for users to access data from their robots and robot controllers. While ABB RobotStudio (the ABB's standard software for ABB robots) provides a comprehensive software solution, it may not be the ideal solution for all users from different software systems and hardware devices, or technologies employed [1]. In 2017, the CC-Link Partner Association (CLPA), in association with the OPC Foundation, has introduced an OPC UA companion specification for the CLPA's new "CSP+ for Machine" technology, which takes the technology one step further by essentially allowing whole machines to be treated in the same way [2]. Hsien-I Lin and Yu-Che Hwang also proposed a solution that enables smart robotic arms to be connected to the OPC UA network and belong to a smart factory, visualizing the entire production system in 2019 [3]. These attempts indicate the importance of resolving critical challenges of machine interoperability in Industry 4.0. However, there still exists a significant challenge in dealing with the many complex robot systems from various vendors and dependence on third-party solutions. These solutions are often not compatible with multiple platforms [4] and may lack user-friendliness for robotics systems.

In this article, we mainly focus on implementation of an architecture based on OPC UA that allows robot users to easily monitor and control robots in a laboratory. According to the reviews above, standardized communication between different devices and connecting all devices from different levels without confliction or blocking in communication standards are the two key elements of this platform. Due to the strongly suitable integration capabilities, OPC UA is used as a standard communication for software and hardware information integration. The proposed architecture uses Delta Robots from VAS Corporation and RoboStar Scara Robots as testbeds. All of these robots commonly use the same upper level (SCADA level) deployed based on OPC UA.

After addressing the interoperability among different robots and devices, we select EtherCAT [5] as a real-time communication protocol at the control level. Because EtherCAT supports a large number of devices on a single network, drivers which control robot's motor are connected through a distributed clock mechanism, so that all drivers on the network are synchronized with the same clock signal, thereby reducing the amount of data that needs to be transmitted and improving overall network performance.

The rest of the article is organized as follows. Section II discusses the theories, including the overall OPC UA in manufacture. Section III discusses the implementation of OPC UA Server and PC Clients. Section IV presents the proposed practical experimental results and analysis. Finally, the conclusion and future work are given.

II. OPC UA Overview

OPC UA is a set of standards designed as a universal communication protocol that is open and royalty-free. It offers a state-of-the-art security model, fault-tolerant communication, and an information modeling framework that allows developers to represent data in a way that makes sense to them [6]. This communication protocol has a wide range of applications that provide benefits of economies of scale for both application developers and robotics developers. As a result, there is increased availability of high-quality systems for monitoring

and controlling robots at a reasonable cost. Generally, OPC UA's security model encrypts data at its source, making it easier for devices to share requests or commands securely through the network. This makes it easier for other devices to share requests or commands since the data is already protected. Specifically, security measures are implemented on edge devices rather than relying solely on network devices. In contrast to common industrial communication protocols such as Modbus TCP and Profinet which can only provide real-time data and cannot guarantee the safety and reliability of historical data, OPC UA overcomes these limitations and revolutionizes industrial communication [3]. OPC UA is also designed to continue operating and providing reliable communication even in the presence of faults or errors. It is built with redundancy features that ensure that if one component or system fails, another one can take over and maintain communication without disruption. This makes OPC UA a highly reliable and robust protocol that can be used in critical industrial applications where communication failures can have serious consequences. Finally, OPC UA offers an information modeling framework that allows developers to easily model objects representing data, processes, or systems, making it easier for clients to access the data they need and obtain a comprehensive and valuable set of information for their own purposes. Table I shows the function comparison between OPC UA and two other same-segment communication standards in industrial scope: Modbus TCP/Profinet following the approach described in [3].

TABLE I
FUNCTION COMPARISON BETWEEN OPC UA AND MODBUS TCP/PROFINET

	OPC UA	***Modbus TCP***	***Profinet***
Variable TagName / Register definition	TagName	Register	TagName
Object-oriented data format	Yes	No	No
Realtime data acquisition	Yes	Yes	Yes
Alarm event log and history data log	Yes	No	Yes
Security	Yes	No	Less

Overall, OPC UA is a highly reliable and robust communication protocol that is well-suited to critical industrial applications of Industry 4.0 and IIoT. It enables both horizontal (machine-to-machine) and vertical (field devices to plant-level and enterprise-level systems) integration, as well as the communication and management of devices and data from every part of the network, from the field to the enterprise. Figure 1 shows the scope of OPC UA within an Enterprise in IIoT.

III. IMPLEMENTATION

A. Architecture

We are focusing on the two specific levels within the automation pyramid for this laboratory's robot system, which

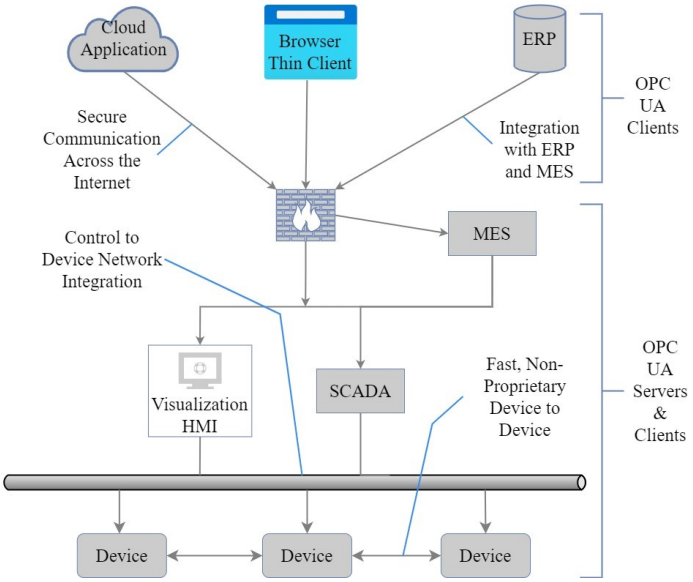

Fig. 1. The Scope of OPC UA within an Enterprise.

are the Supervisory level (SCADA) and Control level (Robot Controller).

Figure 2 presents a graphical representation of how OPC UA is implemented in all the constituent parts of this architecture, including the control-bus layer, field-bus layer, and upper application layer. The Introduction section of the article explains the rationale behind using OPC UA in these components, as it enhances interoperability and standardizes data exchange across the entire system. The platform is presented in the following description, starting from the bottom and progressing upwards. Our laboratory employs DELTA and SCARA robots that are presently governed by Robot Controllers which use EtherCat protocol, a highly advanced and fast protocol in the industry. To incorporate OPC UA into our system, we have devised a small bundle that comprises OPC UA communication and embedded it into the Robot Controller's source code. Therefore, the Robot Controllers function as OPC UA Clients, which enable them to exchange and accept data, execute and respond to commands, and carry out other relevant tasks. The purpose of building an OPC UA Server is to manage and regulate the communication between all OPC UA clients. It serves as a central repository for storing various types of information, including data, event subscriptions, method calls, and more. The server facilitates the exchange of data between clients, allowing them to access and manipulate the shared data in a coordinated and controlled manner. Apart from the two previously described components (the OPC UA Robot Controller Client and the OPC UA Server), the PC Client and Web Client also have significant roles in the platform. These clients also act as OPC UA Clients, which enable them to retrieve data, issue commands, and execute other operations.

B. OPC UA Server

The OPC UA Server is responsible for managing access to data and acts as a repository for information (in-

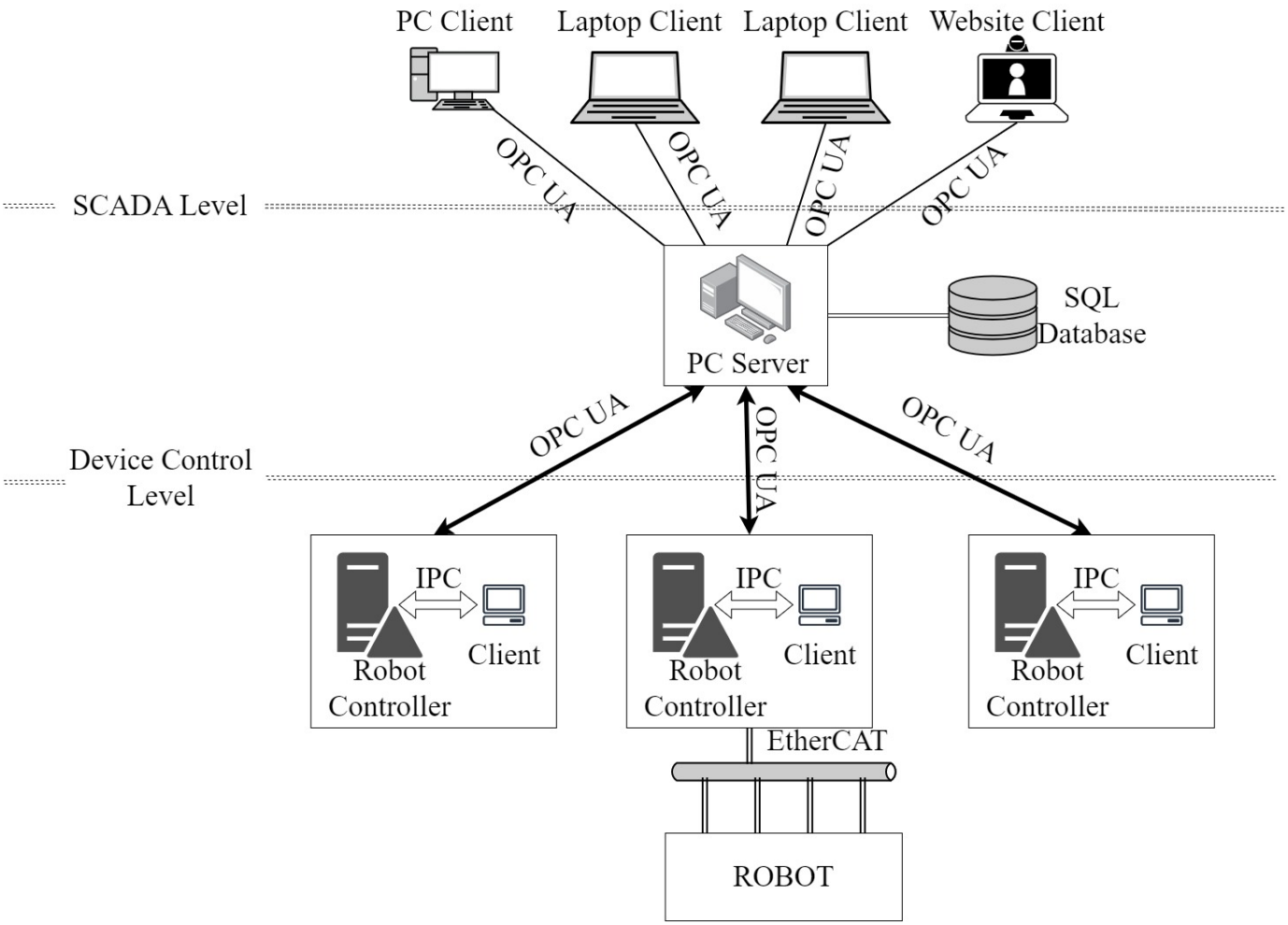

Fig. 2. OPC UA architecture in laboratory's robot system.

formation model) [7]. It is built with C# and uses the Opc.UaFx.Advanced NuGet Package for OPC UA SDK for .NET [8]. This package offers optimized algorithms and data structures for efficient handling of large amounts of data, advanced security features such as encryption and authentication, and compatibility with various OPC UA specifications. As depicted in Figure 2, an OPC UA Server contains a database for a group of laboratory robots. To provide a more detailed insight into the server's structure, a class called "Robot" is defined, which acts as an OpcNodeManager consisting of various robot types. Each robot is represented by an OpcFolderNode with a user-defined name, and each OpcFolderNode can contain a large quantity of OpcDataVariableNode nodes that represent different types of robot data such as joint values of each axis, the robot's position in world coordinate, command strings, and servo status. In addition, Alarm&Events and OpcNodeHistorian can be established to provide additional functions. The server can also include other categories such as "Sensor" or "Motor". To keep track of the robots' status, values, and other information, we use a SQL database connected to the OPC UA Server using the SqlConnection class and a connection string. After establishing the connection, an insert query is used to write the robot's status and information to the database. While the primary function of our database is storage, we aim to enhance its capabilities to analyze the robots' behavior and predict potential issues in the future. Figure 3 illustrates the node-tree structure of an OPC UA Server.

C. OPC UA Client

In brief, the OPC UA Clients discussed earlier were developed in C++ and utilized the open source open62541 library [9], which is one of the most commonly used open-source implementations for OPC UA. The other three implementations are nodeopcua, UA-.NETStandard, and python-opcua. After examining all these available open-source implementations for OPC UA, we found that open62541 is the most dependable and suitable choice. This is supported by the fact that open62541 has the highest number of stars (2070) on their GitHub pages, indicating strong community support. Additionally, the authors of [10] found no significant security vulnerabilities in open62541, further confirming its reliability.

- **PC Client**
  The PC Client is utilized to observe and manage the

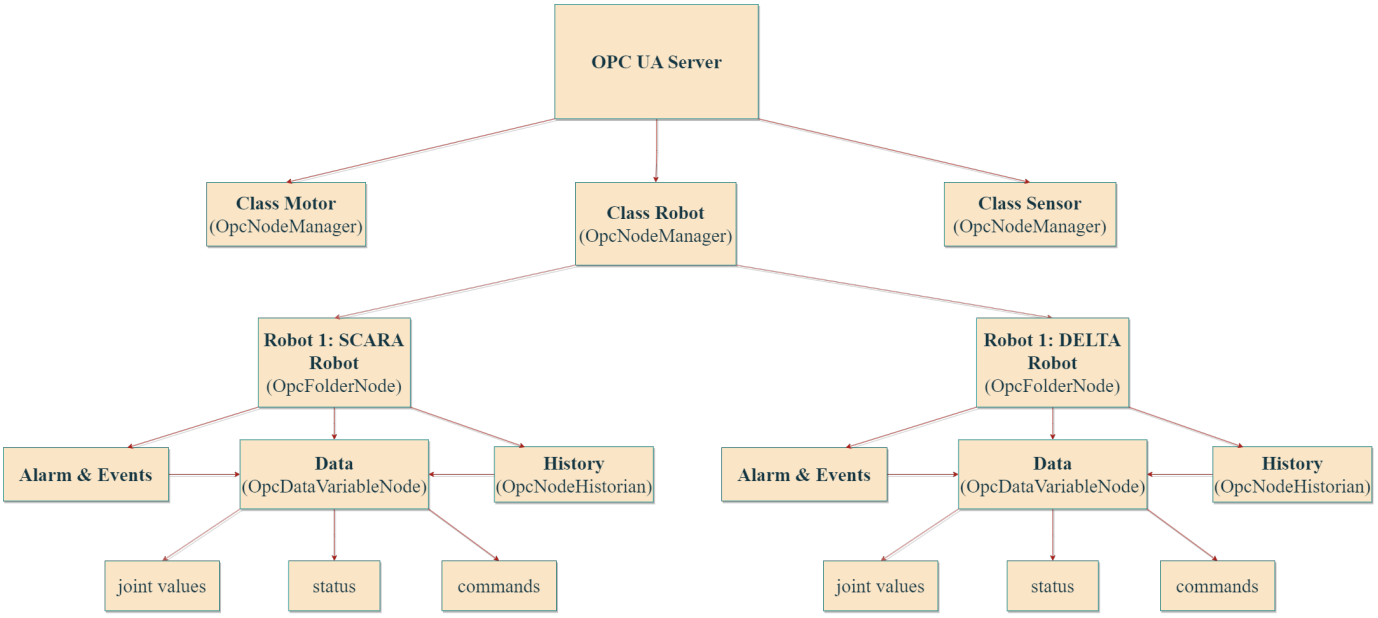

Fig. 3. Node-tree structure of an OPC UA server.

robot system. In regards to its functions, the PC Client possesses the majority of the necessary features found in a robot controller: observing the status parameters of the robot system including Tool Number, Jogging Speed, Operation Coordinate System, Servo Status, System Status, Mode, Security Level, Controller Access Permission; showing the joint value of each axis and the position of the robot in World Coordinate; indicating IO signals, alarm messages, and operation history; remote controlling robot available in two modes. In terms of manipulation, in TEACH mode, users are able to jog the manipulator and access programming, editing, and customization functions based on their level of security clearance. Engineers can further customize their control by adjusting the Jogging Speed, choosing the Operation Coordinate, changing the Active Tool, toggling DO signals, saving new teaching points, and using them in programming. In AUTOMATIC mode, the following tasks can be performed: executing a previously saved program and monitoring the status, I/O signals, and program information [11]. The interface of a PC Client is shown in Figure 4.

- **Website Client**
  Current literature offers solutions that allow OPC UA connectivity from web devices, but most of them require specialized knowledge of communication services and data modelling of the OPC UA standard. Our approach was motivated by a desire to design a web-based platform that does not necessitate such expertise [12]. To this end, we developed a Web Platform using NodeJS [13]. From the viewpoint of OPC UA Servers, this platform acts as an OPC UA Client. Users interact with the front-end website, which sends HTTP requests to a NodeJS Server. This Server executes operations to return results to the

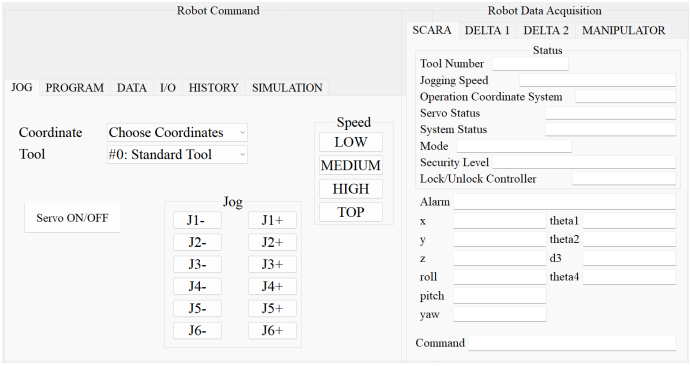

Fig. 4. Capture of OPC UA Client on PC.

web page and connect to a PC Server for monitoring and sending commands to a Robot Controller using the OPC UA protocol. Our platform enables supervisors to analyze the robots' performance, working hours, and other relevant metrics on a daily basis, as well as view all robots in the laboratory. Figure 5 illustrates the Website Client of this plant-level platform.

- **Robot simulation**
  Our proposed platform includes a 3D simulation of both SCARA and DELTA robots, enabling us to visualize a simulated version of the robot alongside the actual model. To achieve this, we employed forward and inverse kinematics of the robots. The kinematic equation of the SCARA robot was formed using the Denavit-Hartenberg convention method [14], while the kinematics analysis of the DELTA robot was performed using the geometric method [15]. Using the real robot's specifications, we determine the most appropriate way to generate its 3D

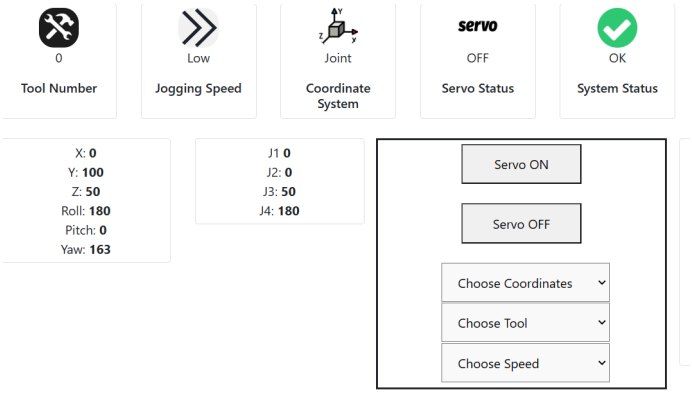

Fig. 5. Capture of OPC UA Client on Website.

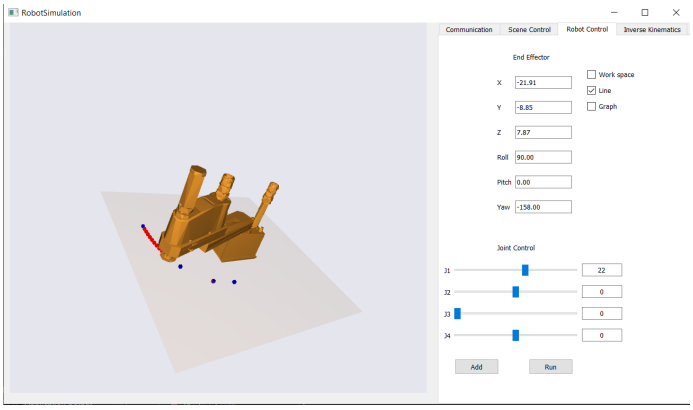

Fig. 6. Capture of SCARA Robot simulation.

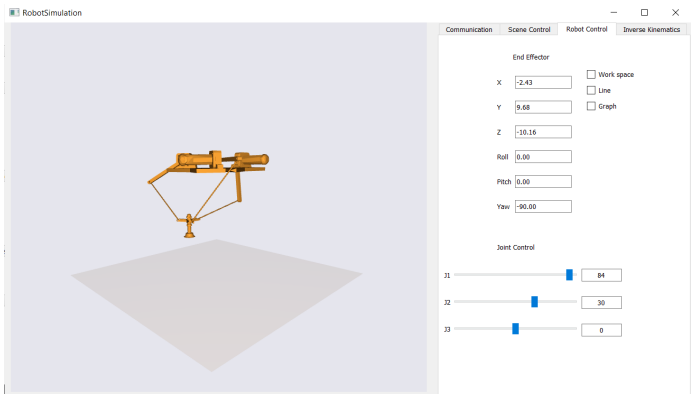

Fig. 7. Capture of DELTA Robot simulation.

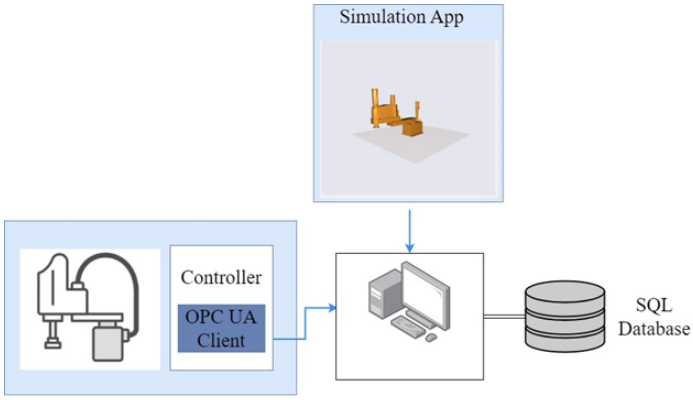

Fig. 8. Connection between simulation application and Robot Controller.

model. We used the Assimp library, which is an open-source library for importing various 3D model formats into an application and can be used in conjunction with OpenGL [16]. The library provides functions to read the data of materials, meshes, and nodes from 3D files and then we use these outputs to reassemble robot's parts and develop conventional simulation functions, which enables users to change perspective, create simulation programs, and simulate the robot before actually sending the program to the Robot controller in simulation mode. In companion mode, the simulation model tracks the real model's movements and displays the robot's behavior, allowing users to intuitively verify the operation of the actual model. Figure 6 and 7 illustrate the SCARA model and DELTA model, respectively. Figure 8 depicts the connection between simulation application and Robot Controller.

- **Integrating OPC UA to Robot Controller**
  In this section's Architecture, we briefly explained how we integrated OPC UA into the Robot Controller. However, we will provide a more detailed explanation to enhance the reader's understanding. Our integration method involves embedding the open62541 library into the Robot Controller's source code folder to enable OPC UA functionality. We also utilize the Inter-process communication (IPC) method, specifically the shared-memory approach, to facilitate the exchange of data between the Robot Controller and the OPC UA Client. This technique involves the creation of a shared memory segment that grants both the Robot Controller and the OPC UA Client access to a common block of memory, which creates a shared buffer for seamless communication between the two processes. By attaching to the shared memory segment, both processes are able to read from and write to the same block of memory. This shared memory segment will persist until the server is terminated, at which point the processes will detach from the segment and it will be automatically deleted, as it is no longer necessary.

D. Experimental Results and Analysis

In order to implement the platform in a real-robot system, it is necessary to assess its performance. To facilitate this, we employed Wireshark, an open-source network protocol analyzer [17] [18]. With Wireshark, we were able to capture and examine real-time network traffic, decode OPC UA packets, and present the findings in a simple-to-understand I/O graph format. The system we experimented on is illustrated in Figure 9.

The system's performance met our expectations, and our PC Client can now receive information and status updates from

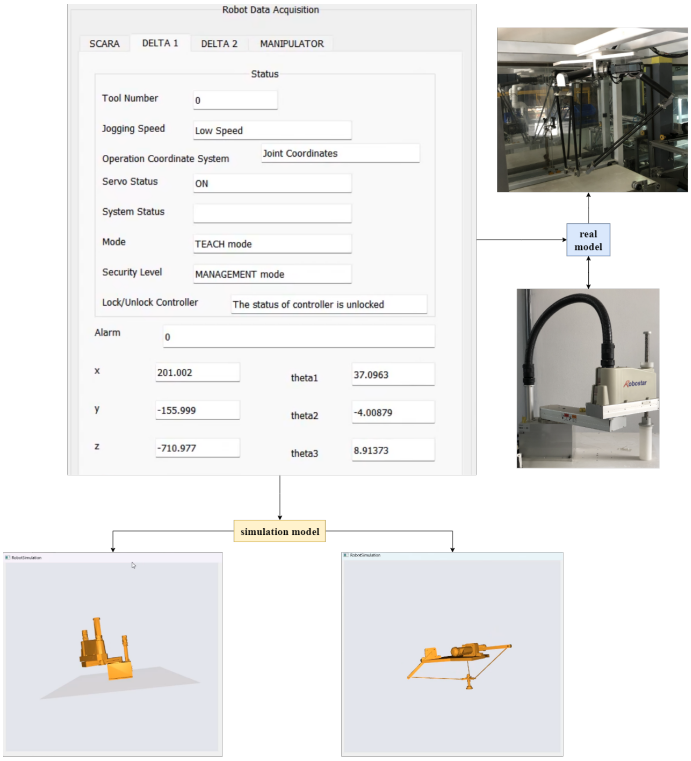

Fig. 9. Experimental robot system layout.

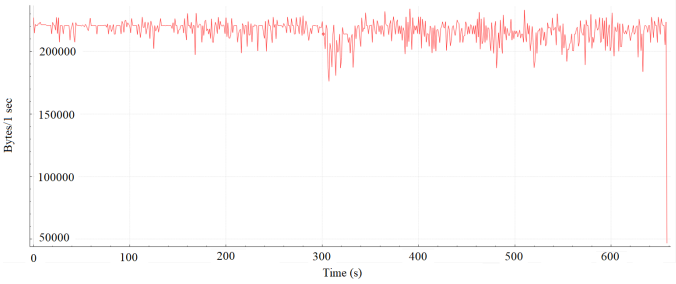

Fig. 10. Bandwidth capture in I/O graph.

the robots, which are then displayed on the application as images and numbers. Additionally, we can send commands to the robot controllers with the push of a button, and the real robot models respond concurrently. After measuring OPC UA network traffic, Figure 10, Table II and III depict the measured result.

TABLE II
INTERFACES

Time elapsed (s)	10:58
Interface	Ethernet
Dropped packets	0(0.0%)
Link type	Ethernet

We observed that no data packets were lost during experiment, and all packets were captured and exhibited accurately in Wireshark. The average bandwidth of the network was around 216,000 bytes per second, indicating efficient and effective communication. Additionally, we measured the average packet size, which was approximately 153 bytes.

TABLE III
STATISTICS

Measurement	*Captured*	*Displayed*
Packets	929729	929729 (100.0%)
Average packet size, B	153	153
Average byte/s	216k	216k

E. Conclusion and Future Work

As previously noted in the Introduction, the lack of standardization and interoperability in robot systems has greatly diminished their efficiency and effectiveness in factory operations. This paper presents the implementation of an OPC UA platform in a plant-level operating system. The platform includes an OPC UA architecture, which consists of a server with a SQL database, clients with robot simulation, and integration between OPC UA and Robot Controller. This architecture promotes interoperability among different components running distinct application software, enabling seamless standardized communication. In addition, we conducted an experiment in our laboratory on our robot system and comprehensively analyzed the results of the network traffic, including network bandwidth, dropped data packets, average packet size and more. However, the evaluation was limited to a small number of robots and clients for a short period, and therefore not sufficient to conclusively determine the performance and stability of the platform. Future research will involve applying this approach to a wider operating system that may include dozens of robots to obtain a more accurate evaluation.

ACKNOWLEDGMENT

"This research is funded by Ho Chi Minh City University of Technology – VNU - HCM under grant number SVKSTN-2022-DDT - 30". We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for this study.

REFERENCES

[1] ABB Robotics, Application manual IRC5 OPC Server help. https://library.e.abb.com/public/dea868184c0fe423c12578a8005bcb98/3HAC023113-001_en.pdf Access 26th March. 2023.
[2] OPC Foundation, OPC Technologies – Unified Architecture (UA), "CLPA announces OPC UA companion specification for new "CSP+ for Machine" technology". https://opcfoundation.org/news/press-releases/clpa-announces-opc-ua-companion-specification-new-csp-machine-technology/ Published 28th November 2017.
[3] H. -I. Lin and Y. -C. Hwang, "Integration of Robot and IIoT over the OPC Unified Architecture," 2019 International Automatic Control Conference (CACS), Keelung, Taiwan, 2019, pp. 1-6.
[4] .NET Based OPC UA Client & Server SDK (Bundle), https://www.unified-automation.com/products/server-sdk/ Access 26th March. 2023.
[5] EtherCAT Technology Group, Features of EtherCAT. https://www.ethercat.org/en/technology.html Access 26th March. 2023.

[6] OPC Foundation, OPC Technologies – Unified Architecture (UA). https://opcfoundation.org/about/opc-technologies/opc-ua / Access 26th March. 2023.
[7] Pauker, F., Ayatollahi, I., & Kittl, B. (2014). OPC UA for machine tending industrial robots. In Proc. of the 2nd Intl. Conf. on Advances In Robotics Engineering (pp. 79–83). http://hdl.handle.net/20.500.12708/67093.
[8] OPC UA SDK for .NET, Development Guides. https://docs.traeger.de/en/software/sdk/opc-ua/net/start#opc-ua-sdk-for-net Access 26th March. 2023.
[9] Open62541, open62541 release 1.3 Documentation. https://www.open62541.org/doc/open62541-1.3.pdf Access 26 March. 2023.
[10] N. Mühlbauer, E. Kirdan, M. -O. Pahl and G. Carle, "Open-Source OPC UA Security and Scalability," 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 2020, pp. 262-269.
[11] VAS Corporation, XMC-ROBOT EtherCAT Master Robot Controller User's Manual, Published 30th December. 2022.
[12] S. Cavalieri, D. Di Stefano, M. G. Salafia and M. S. Scroppo, "A web-based platform for OPC UA integration in IIoT environment," 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 2017, pp. 1-6.
[13] Sterfive SAS, node-opcua/node-opcua-htmlpanel. https://github.com/node-opcua/node-opcua-htmlpanel Access 26th March. 2023.
[14] A. N. Barakat, K. A. Gouda and K. A. Bozed, "Kinematics analysis and simulation of a robotic arm using MATLAB," 2016 4th International Conference on Control Engineering & Information Technology (CEIT), Hammamet, Tunisia, 2016, pp. 1-5.
[15] C. LIU, G. -H. CAO and Y. -Y. QU, "Motion simulation of Delta parallel robot based on Solidworks and Simulink," 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 2019, pp. 1683-1686.
[16] Assimp, Model Loading. https://learnopengl.com/Model-Loading/Assimp Access 26th March. 2023.
[17] Wireshark. https://www.wireshark.org/ Access 26th March. 2023.
[18] OPC Foundation, Analyzing OPC UA Communications with Wireshark. https://opcconnect.opcfoundation.org/2017/02/analyzing-opc-ua-communications-with-wireshark/ Access 26th March. 2023.