

PAPER • OPEN ACCESS

Modular and flexible Automation Middleware based on LabVIEW and OPC UA

To cite this article: A Künzel *et al* 2021 *IOP Conf. Ser.: Mater. Sci. Eng.* **1193** 012109

View the [article online](#) for updates and enhancements.

You may also like

- [The application of the LabVIEW environment to evaluate the accuracy of alternating voltage measurements](#)
P Otomaski, Z Krawiecki and A Odon
- [Video streaming technologies using ActiveX and LabVIEW](#)
M Panoiu, C L Rat and C Panoiu
- [Smart House Control using LabVIEW](#)
S Angalaeswari, T Deepa, D Subbulekshmi et al.



*Benefit from connecting
with your community*

ECS Membership = Connection

ECS membership connects you to the electrochemical community:

- Facilitate your research and discovery through ECS meetings which convene scientists from around the world;
- Access professional support through your lifetime career;
- Open up mentorship opportunities across the stages of your career;
- Build relationships that nurture partnership, teamwork—and success!

Join ECS! **Visit electrochem.org/join**



Modular and flexible Automation Middleware based on LabVIEW and OPC UA

A Künzel¹, A Puchta^{1*}, P Gönheimer¹ and J Fleischer¹

¹ Karlsruhe Institute of Technology, wbk Institute of Production Science, Kaiserstraße 12, 76131 Karlsruhe, Germany

*Corresponding author: alexander.puchta@kit.edu

Abstract: The increasing automation level of processes in production systems leads to new technical challenges, especially in the implementation and maintenance of software architectures. New requirements arise regarding the interface between Programmable Logic Controllers (PLC), robots, Human Machine Interfaces (HMI) and superordinate information systems (e.g. ERP). Industry 4.0 demands, among other things, an increase in flexibility, adaptability and transparency to achieve vertical and horizontal interoperability and a continuous integration. The innovative automation middleware is capable of replacing the heterogeneous interface landscape, which currently exists in many companies and institutions. The basic idea is the implementation of a modular and standardized middleware. Due to relevant characteristics, such as dataflow orientation and graphical programming interface, using LabVIEW as a programming language turned out to be the most suitable solution. The system deploys OPC Unified Architecture (OPC UA) to connect all required components across multiple enterprise levels. Moreover, the software solution controls the workflow and collects process data for further analysis. In contrast to software products available on the market, which usually come along with manufacturer dependencies, the established middleware based on the combination LabVIEW and OPC UA is transparent, extensible and independent.

Keywords: Middleware, OPC UA, LabVIEW, Standardization, Modularization

1. Introduction

The increasing automation level of processes in production systems as a consequence of the fourth industrial revolution, leads to new technical and organizational challenges. The so-called Industry 4.0 is an initiative of the German industry and government to accelerate digitalization [1]. The concept is based on the trend, that the real world and the virtual world are converging and form an ‘Internet of Things’ (IoT). These new requirements have led to the dissolution of the conventional hierarchical structure of a company, also known as the automation pyramid. The modern approach of a cyber-physical system (CPS) can be considered as the technological foundation for modern IoT productions in the context of Industry 4.0. In order to meet the requirements, software systems are expected to provide a high level of flexibility, adaptability and transparency [2]. Additionally, horizontal and vertical interoperability and integration must be ensured.

Over the years, many companies and institutions have developed a heterogeneous system and process landscape with mainly manufacturer-specific interfaces [3]. In the field of production automation, both hardware and software components are affected by this. Generally, machines, controllers, sensors, actors or robots are developed and manufactured by a wide variety of manufacturers. This results in diverse interfaces, communication channels and data transmission methods. Especially, the problems arising



from the interfaces between Programmable Logic Controller (PLC), robots, Human Machine Interfaces (HMI) and superordinate information systems (e.g. ERP) occupy many manufacturing companies. In the long term, the heterogeneous system landscape needs to be replaced by a flexible system that enables horizontal and vertical communication across different levels.

There are several approaches to address the interface challenge in modern industrial companies. Made-to-measure approaches address the individual implementation and adaptation to specific requirements but lead to various problems. For instance, the lack of standardization results in inflexible systems that are difficult to expand. The obscurity of individual software solutions and insufficient error handling lead to a low level of robustness and reliability. Another approach is the use of external software packages. Although these solutions offer a standardized interface, they are usually not flexible enough to be applied to different use cases and environments. In addition, there is a continuing dependency on the developers and their knowledge base. In this paper, the scenario of the existence of a standardized and independent communication system with a transparent and accessible code is examined. Therefore, developers from different fields and levels of expertise are able to understand, adapt and further extend the software code.

This paper intends to lay a foundation for the design, architecture, and implementation of such software in a production environment. Based on elaborated requirements, the system architecture, the communication channel and the programming language are to be analyzed and selected. A suitable combination of existing standards and concepts forms the basis of the innovative solution, respectively middleware.

2. State of the Art

Systematic benchmarking and targeted technology research are used to examine the ready-made software products available on the market for suitability and possible analogies. There are many ready-made software solutions on the market. However, many of them do not allow a transparent insight into the source code and the overall structure. Useful approaches are the Totally Integrated Automation (TIA) Portal from Siemens AG and the SecurePLUGandWORK concept from Fraunhofer IOSB. These concepts lay a foundation for standardized and modular automation systems and focus on the principle of interoperability. A few aspects of interesting realization methods are presented below.

The implementation of a programming standard in the TIA Portal is based on the following five aspects: the definition of one or several compatible *programming languages*, the establishment of a *programming style guide*, the standardized *interface definition*, the realization of *modularity* and the provision of *libraries*. In this paper, the aspect of modularization in combination with standardized interfaces plays an especially decisive role for the target system. The basic idea is the division into three different and interrelated groups: Units, Equipment Modules and Control Modules. The architecture is hierarchically structured. Equipment modules can, for example, contain other equipment modules or control modules [4].

The Fraunhofer SecurePLUGandWORK concept is characterized by an innovative communication concept. The basic idea is the differentiation into two modules, named communication level and comprehension level. These two components combined can be interpreted as a kind of middleware. The communication level is realized by the modern and powerful industry standard OPC Unified Architecture (OPC UA). The comprehension level is formed by a neutral data format named Automation Markup Language (AutomationML). The format is based on Extensible Markup Language (XML) and defines the syntax. This allows different languages of the individual devices and plants to be transformed into one uniform language. Event-driven communication between servers and clients is realized with the help of the OPC UA communication standard. Integrated security mechanisms allow data to be sent securely and between different locations [3, 5].

3. Modular and flexible Automation Middleware

In order to select a suitable system basis, the requirements for the overall system and the middleware must be defined. Based on this, various communication channels of the target system are examined for

their compatibility with the production environment. Subsequently, the use of different programming languages is analysed. As a result, the basic concept of the target system is defined as a suitable combination.

3.1. Requirements

In the overall system, there are different stakeholders with different requirements. Relevant stakeholders of a production company include developers in the field of automation technology, technicians working on site, the management department and possible customers. The most relevant requirements for the overall system and middleware, based on VDI/VDE 2657 [6], in combination with stakeholder demands, are listed below.

- Uniform communication interface: e.g. Manufacturer- and technology-independence, Standardized data exchange, Defined information model
- Code, data flow and structure transparency
- Accessibility of the middleware and source code: e.g. Comprehensibility, Learnability, Usability
- Vertical and horizontal interoperability
- Flexibility: e.g. Extensibility, Scalability, Configurability, Plug & Play functionality
- Maintainability
- Reliability
- Robustness and error tolerance
- Performance: e.g. Real-time capability, Data processing features

3.2. Architecture

From these requirements, the basic structure for the overall system and the middleware are elaborated. Since the architecture plays a decisive role at various levels different architectural levels must be distinguished. These are: the basic structure, the architectural pattern of the software system and the program structure on the middleware level.

Considering the context, it is reasonable to choose a CPS as the basic system structure. CPSs can be considered a foundation for digitized companies in the context of Industry 4.0. By networking physical and virtual components, the intelligence and flexibility of the overall system increases. Through the composition of embedded computers and networks, CPSs can autonomously monitor, control and analyze processes. This increases the reliability and resilience of the overall system [7]. Figure 1 shows an example architecture for a typical CPS system.

When considering various architecture patterns, modular concepts like the client-server architecture, the service-oriented architecture (SOA) and the microservices architecture turned out to be particularly suitable. Based on research, SOA appears to be the best option. The total system is divided into individual components, and/or services, which can be embedded over a common interface into the total system. The individual subsystems are reusable, independently executable and operable. Clients and servers can work autonomously or interact if required. In addition, studies have shown that the use of SOA for a CPS brings decisive advantages. SOA is kind of a paradigm, which is used as a basis for the implementation of CPSs and supports the implementation of the requirements for the target system. This enables a dynamic combination of resources, including the fast and easy exchange of physical components - such as actuators and sensors. Additionally, SOA supports a CPS in its ability to package physical components and resources as services. This allows software and hardware components to be represented as interoperable services. In conclusion, the use of SOA as an architecture pattern improves sustainability and predictability in CPS and leads to real-time capability [7].

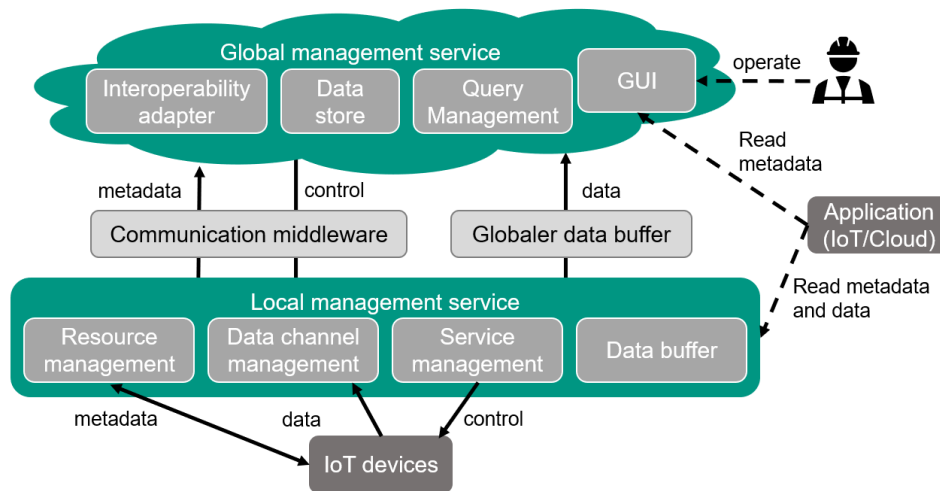


Figure 1. Prototype of a CPS architecture, own illustration based on [8]

On the level of the program structure, requirements like transparency, accessibility, flexibility and robustness are to be fulfilled. Therefore, it is reasonable to structure the program code modularly and make it accessible via standardized interfaces. Individual functions or definitions can be stored in libraries and thus easily be reused. This modular structure can be extended over several levels. For example, a module contains other sub-modules, which in turn can consist of even smaller sub-modules. Thus, the program code becomes transparent and the development is simplified by reusable components. An example of the structure of such a modular program code is shown in figure 2. If the production system and, accordingly, the program code is expanded, the attached library can be extended as well. The hierarchical structure by the main module, module and sub-module is not obligatory. Modules with superordinate or extensive functions can be used in sub-modules. An example for the nested component structure is Module 2 in figure 2, which contains other sub-modules (Sub-module 1) and is used simultaneously in other modules (Module 4) or even sub-modules.

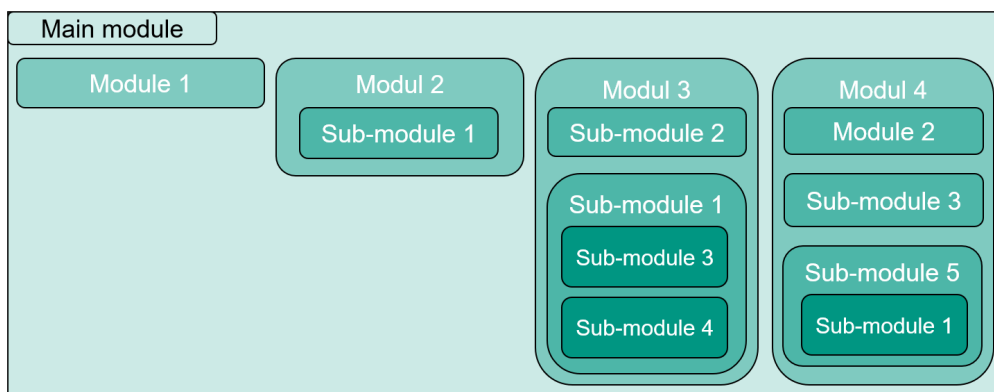


Figure 2. Exemplary structure of a modular program code with corresponding library.

The program code can be assembled, extended or adapted according to a kind of construction kit principle. Typical system restructurings contain the integration of new modules into existing systems, the removal of modules or the exchange of modules with a similar interface. Restructuring can change the system input and output, but the selected data format remains the same. No configuration or adaptation of the interface is necessary, since a suitable definition of the system boundaries is already made at the implementation of the modules.

3.3. *Communication*

The goal of the paper is to enable an innovative Industry 4.0 production in the environment of a flexible IoT infrastructure. For this, the communication channel must meet the requirements of a modern CPS. For example, vertical and horizontal integration and a standardized, reliable, scalable, secure and robust communication channel between all participants of the network. Industrial Ethernet systems like Process Field Network (PROFINET), Ethernet for Control Automation Technology (EtherCAT), Modbus Transmission Control Protocol (ModbusTCP) or EtherNet/IP cannot adequately meet these requirements [9]. Therefore, the selection in this paper is limited to the three appropriate possibilities Message Queue Telemetry Transport (MQTT), Representational State Transfer (REST) in combination with Hypertext Transfer Protocol (HTTP) and OPC UA.

Machine-to-Machine (M2M) communication in the IoT is often realized by a modern network protocol called MQTT. This IoT connectivity protocol is nowadays frequently used in the field of social networks but also in sensor networks, e.g. via satellite. The data communication is based on the modern Publish Subscribe (Pub/Sub) concept. The principle of operation involves servers that publish data and clients who subscribe to this data. Communication takes place based on TCP via a so-called broker, i.e. an MQTT server [10].

An alternative approach is to realize communication using web services. Such services can be based on various protocols and principles. Due to high performance and flexibility REST in combination with HTTP has proven to be particularly suitable [11, 12]. In contrast to the industrial standards and fieldbus systems presented, REST represents an abstracted concept that defines an architectural style for network applications. A client-server model forms the basis and the elements involved can be uniquely identified via a Uniform Resource Identifier (URI) [12].

OPC UA is a service-oriented M2M protocol and defined in IEC 62541 specification [10]. The communication protocol is based on a client-server architecture. The concept is considered one of the most suitable approaches for the realization of middleware for innovative CPS. However, OPC UA can be interpreted not only as a cross-platform communication middleware, but also as an information model. Therefore, the transferred data provide a high semantic content and intelligent client applications can be developed. The communication can be implemented via the modern Pub/Sub mechanism [13].

The following summary of the concept selection is based on various papers that have dealt intensively with the topic of communication in the context of Industry 4.0 and IoT. Considering the Round Trip Time (RTT) for different package sizes, OPC UA with Pub/Sub implementation is faster compared to MQTT. As a result, OPC UA has the best approach to modeling information, which leads to a high performance. MQTT, on the other hand, can be considered as a lightweight protocol, which comes along with a small overhead and a slower RTT of packages sent to the server [10]. The REST architecture is a very successful method and already established in nearly every domain. The main weakness is the data transfer without type data and meta-data. Therefore, the data values in REST architectures are changing their data types during the transfer. In contrast, OPC UA provides the capability of information modeling, a more robust transport system and mature security mechanisms. The transferred data values retain their type and accuracy [14]. In conclusion, REST is particularly suitable for the Internet with its constantly changing applications and requirements whereas OPC UA is particularly suitable for the more structured manufacturing environment. Furthermore, OPC UA communication is based on an SOA and thus fits into the selected architecture scheme [15]. For these reasons, OPC UA is chosen as the communication channel for this middleware. However, for some applications an entire system conversion is not practical, therefore REST and OPC UA can be combined if required [16].

3.4. *Programming language*

The requirements for the target system also affect the selection of a suitable programming language. Well-known object-oriented languages such as Java, C# or C++ can be used for this purpose. Modern languages like Python, which support multiple programming paradigms (e.g. object-oriented or functional), can also be used to implement the middleware. An alternative approach is the use of graphical programming languages. In this context, Simulink and LabVIEW are of particular importance.

The programming language should enable clear, modular and user-friendly program codes. Purely textual languages, however, require a lot of training and can quickly become complex in the case of complicated overall systems. Especially in the area of manufacturing, it is important that many employees with different levels of knowledge can gain insight into the program code. A graphical programming language allows the required transparent insight into the program code, execution and even data flows. Moreover, the modular structure can be easily realized graphically by using the drag and drop approach. Furthermore, the programming language should provide a standardized interface for common physical ports, such as RS232 and USB, as well as for devices, actors and sensors from different manufacturers. Therefore, manufacturers of programming languages that require payment and licenses, in this case Simulink and LabVIEW, provide ready-made and officially tested libraries and drivers. The programming language should also be suitable for the use in automation technology. Large amounts of data, real-time processing and parallel processes should not be a problem. LabVIEW is particularly suitable, since the system was designed for exactly this application. The manufacturer National Instruments (NI) also offers many toolboxes that include Industry 4.0 functions. Examples are Data Processing Toolboxes or modern interfaces, especially for the selected OPC UA communication protocol. Based on the facts mentioned above, it is reasonable to use LabVIEW as a programming system.

3.5. Basic concept

By identifying the requirements for the target system, important framework conditions could be specified. The overall system consists of a combination of the solution modules developed. The basic structure is a CPS that enables an innovative, flexible and comprehensive production environment in the context of Industry 4.0. In combination with SOA as a paradigm, a high-performance system is created. At the deeper level of the program code, transparency and reusability can be combined through a modular approach. With OPC UA as the communication channel, a good foundation for standardized communication based on a powerful information model is created. Based on the already defined solution modules, LabVIEW is suitable as a programming system. Due to the graphical user interface, especially the representation of data flows, a high level of modularity and transparency are given.

4. Implementation Middleware

Due to the complexity of the middleware, completion on a universal level is not achievable within the scope of the paper. Rather, the goal must be to lay the foundation for a long-term project that will build upon the results. Therefore, it is reasonable to first develop and validate the system on the basis of a concrete use case. Despite direct reference, the focus has to be on the general requirements of the target system. Consequently, the middleware needs to be built of individual modules with standardized interfaces, according to the modular principle. In this way, an extension or the application for a new system can be carried out smoothly in the later phase of the project. The focus lays on the presentation of the system architecture, the code structure and the communication method.

Figure 3 shows the concrete use case and its system architecture. An important component is the cloud-based management and planning layer, which contains different applications, respectively clients or servers. This layer is in permanent exchange with the middleware via OPC UA or REST. LabVIEW as the framework of the middleware can host both, an OPC UA client and an OPC UA server. A local database (DB), e.g. an SQL DB, as well as IoT devices or robots that have an OPC UA interface can be integrated easily. Additionally, devices for which NI offers drivers or standardized interfaces can be embedded without any problems and the direct connection to devices that are controlled via Dynamic Link Library (DLL) files is possible as well. Due to the division into front and back panel, LabVIEW can be used as a kind of HMI for configurations or parameter inputs. The OPC UA client is used to communicate with the control level, for example with a PLC. The PLC hosts an OPC UA server and is connected to the field devices, i.e. actuators, sensors, robots and other devices.

In this application, the middleware is mainly responsible for the following tasks:

- Control of the higher-level system workflow
- Data acquisition, processing and storage
- Interaction with higher level (e.g. cloud)
- Control of compatible devices, robots, sensors, actuators

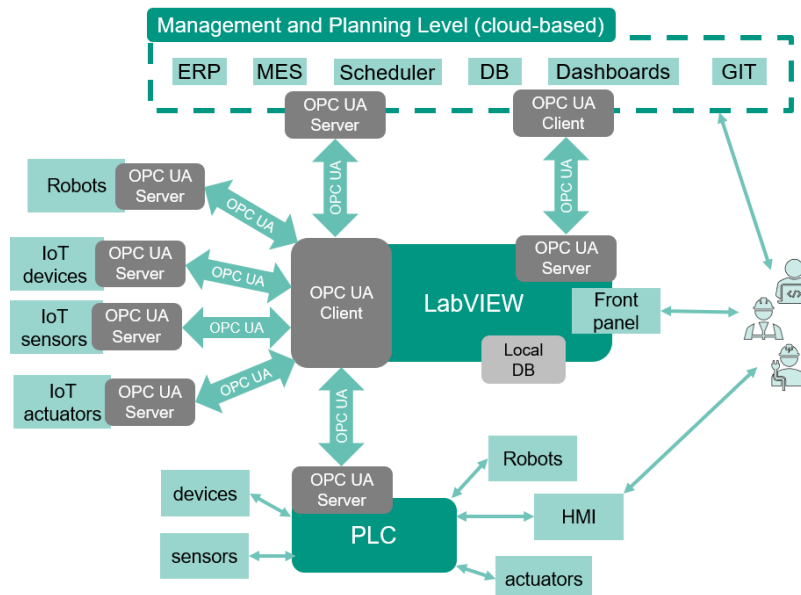


Figure 3. System architecture of the concrete Use Case

In contrast, the PLC operates and controls at a lower level and includes small work steps, concrete positions or the control of sensor values.

In LabVIEW, the code structure can be easily visualized by the graphical user interface. There is a large library that contains many important functions and function blocks. Examples are numerical methods, data format conversions, timers or loops. These subprograms are called Virtual Instruments (VI) or SubVIs, which can be interpreted as modules. Each VI has a defined input and output and is independently executable. Self-implemented functions, workflows, read/write requests or event structures can be stored in individual SubVIs, which extend the existing library and can be reused without difficulty. State machines can easily be implemented using predefined structures. This ensures a clear code as well as transparent states. Internal and external event cases can also be integrated into the state machine concept. Accordingly, the program code is composed of individual modules (VIs, SubVIs), state machines and events structures. The variables of the PLC source code are accessed via OPC UA. The generally known data format JSON defines the data structure. Since OPC UA offers the high-performance communication mechanism Pub/Sub, this method is implemented in LabVIEW. For this purpose, one or more items to be monitored can be selected by using IDs. As a result, it is possible to react to value changes using event structures.

5. Results and Discussion

Modern production facilities in Industry 4.0 require new flexible and modular approaches. For this purpose, the use of a middleware concept is suitable since it standardizes the communication, takes over functions of the higher-level system control and enables a seamless connection across all company divisions and levels. The developed solution is an innovative combination of modern and standardized approaches. The combination of OPC UA as communication channel, the Pub/Sub communication mechanism, JSON as data format, code modularity and LabVIEW as the programming system creates a powerful and flexible middleware. The implemented software solution can easily be embedded in a

SOA-based CPS and supports its benefits. The system has proven itself in various continuous operations and system restructurings. It serves as a basis for the further stages of the project. The elaborated code structures, functions, communication mechanisms and interface definitions can be reused and extended as required. Thus, more and more components of a CPS can be integrated easily. Further research will focus on the validation of system security, performance and real-time ability, including the handling of large datasets.

References

- [1] BMBF 2014 *The new High-Tech Strategy Innovations for Germany* (https://www.research-in-germany.org/dam/jcr:8d61ee98-26bd-4606-a21d-1b97b17ca611/HTS_Broschuere_eng) accessed 5 February 2021
- [2] Schleipen M, Gilani S, Bischoff T and Pfrommer J 2016 OPC UA & Industrie 4.0 - enabling technology with high diversity and variability *Procedia CIRP* **57** pp 315–320
BMBF *SecurePLUGandWORK* (<https://www.plattform-i40.de/PI40/Redaktion/DE/Anwendungsbeispiele/004-secure-plug-and-work-fraunhofer-iosb/beitrag-secure-plug-and-work-fraunhofer-iosb.html>) accessed 25 January 2021
- [3] Abbing J and Reinisch K 2019 *Standardisierung mit dem TIA Portal* (Siemens AG) (<https://assets.new.siemens.com/siemens/assets/api/uuid:b16d1307-f59d-4bf3-a363-a2ee102b106f/StandardisierungsWorkshopSuedKundenversion.pdf>) accessed 1 February 2021
- [4] Barton D, Gönzheimer P, Qu C and Fleischer J 2018 Self-describing connected components for live information access within production systems *Procedia Manufacturing* **24** pp 250–257
- [5] VDI e.V. 2013 *Middleware in der Automatisierungstechnik*, VDI/VDE 2657 Blatt 1 pp 2-11
- [6] Gruettner A, Richter J, Basten D 2017 Explaining The Role Of Service-Oriented Architecture For Cyber-Physical Systems By Establishing Logical Links *25th European Conference on Information Systems (ECIS) (Portugal: Guimarães)* pp 1853–1868
- [7] Mohalki S K, Narendra N C, Badrinath R, Le D 2017 Adaptive Service-Oriented Architecture for Cyber Physical Systems *IEEE Symposium on Service-Oriented System Engineering (SOSE) (San Francisco)* pp 57–62
- [8] Drahos P, Kucera E, Haffner O, Klimo I 2018 Trends in Industrial Communication and OPC UA 2018 *Cybernetics & Informatics K&I* (Slovakia: Lazy pod Makytou) pp 1–5
- [9] Profanter S, Tekat A, Dorofeev K, Ricker M, Knoll A 2019 OPC UA versus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols *IEEE International Conference on Industrial Technology (ICIT) (Australia: Melbourne)* pp 955–962
- [10] Mumbaikar S, Padiya P 2013 Web Services Based On SOAP and REST Principles *International Journal of Scientific and Research Publications* **3** (5)
- [11] Tilkov S, Eigenbrodt M, Schreie S 2015 *REST und HTTP: Entwicklung und Integration nach dem Architekturstil des Web* (Dpunkt.Verlag)
- [12] Graube M, Hensel S, Iatrou C, Urbas L 2017 Information Models in OPC UA and their Advantages and Disadvantages *22nd IEEE International Conference on Emerging Technologies and Factory Automation* (Cyprus: Limassol) (<http://ieeexplore.ieee.org/servlet/opac?punumber=8233358>) accessed 5 February 2021
- [13] Rinaldi J 2019 *Why use OPC UA instead of a RESTful interface?* Maintworld - magazine for maintenance & asset management professionals (<https://www.maintworld.com/Partner-Articles/Why-use-OPC-UA-instead-of-a-RESTful-interface>) accessed 6 February 2021
- [14] Eckhardt A, Mueller S, Leurs L 2018 An evaluation of the applicability of OPC UA Publish Subscribe on factory automation use cases *IEEE 23rd International Conference on Emerging Technologies and Factory Automation (Italy: Turin)* pp 1071–107
- [15] Cavalieri S, Di Stefano, Salafia M G, Scroppe MS 2017 Integration of OPC UA into a Web-based Platform to enhance interoperability *26th International Symposium on Industrial Electronics (ISIE) (United Kingdom: Edinburgh)* pp 1206–1211