An OPC UA PubSub MQTT Configuration Tool

Zepeng Liu

LTCI, Télécom Paris
Institut Polytechnique de Paris, France
Email: zepeng.liu@telecom-paris.fr

Patrick Bellot

LTCI, Télécom Paris
Institut Polytechnique de Paris, France
Email: patrick.bellot@imt.fr

***Abstract*—With the ongoing progression of incorporating massive emerging technologies into industrial systems, Information and Communication Technology (ICT) based systems face increasing difficulties of vertical and horizontal integration. One main objective of industrial system integration is to use flexible and scalable communication technology to fulfill requirements on the different levels of an automation pyramid. With the Client/Server model being the de facto standard in industrial automation at upper levels, Open Platform Communication Unified Architecture (OPC UA) is extended by adding a new communication architecture, called Publish Subscribe, to provide asynchronous information exchange capabilities. An OPC UA PubSub and Message Queuing Telemetry Transport (MQTT) based configuration tool is introduced in this paper. It can be easily added to any C/C++ OPC UA Server/Client implementation.**

Keywords–IoT; OPC UA; MQTT.

I. Introduction

Major advances in ICT have contributed to significant changes in the manufacturing domain, from the office to the field level. The massive ICT integrated industrial systems were deployed not only following the success of the Internet, which has broadly spread the idea of ubiquitous connectivity, but also because of increasing global competition [1].

In retrospective, the progress of the network of "things" in the manufacturing field has encountered different stages. From the beginning of the 1970s, Computer Integrated Manufacturing (CIM) was first introduced into manufacturing systems and designed for field-level engineering process management. This was followed by Product Data Management (PDM), networks within engineering departments that were responsible for product data and connection. Since the 1990s, the emerged Product Lifecycle Management (PLM) concept adopted services and interfaces provided by PDM to manage all product-related information, from product design, production, maintenance to retirement, essentially, the entire product life cycle. The Digital Factory concept was proposed at the turn of the century, and aims to simulate a real factory by integrating data, models, and the engineering process [2]. Enterprise Resource Planning (ERP) was derived from its Manufacturing Resource Planning (MRP) predecessor rather than tailored to particular market sectors. ERP aims at being a framework for integration of all business-related information including material and production management, human resource management, financial accounting, etc. Below the ERP, the Manufacturing Execution System (MES) accomplishes the short-term planning task and acts the role of bridge between strategic level and control level [3].

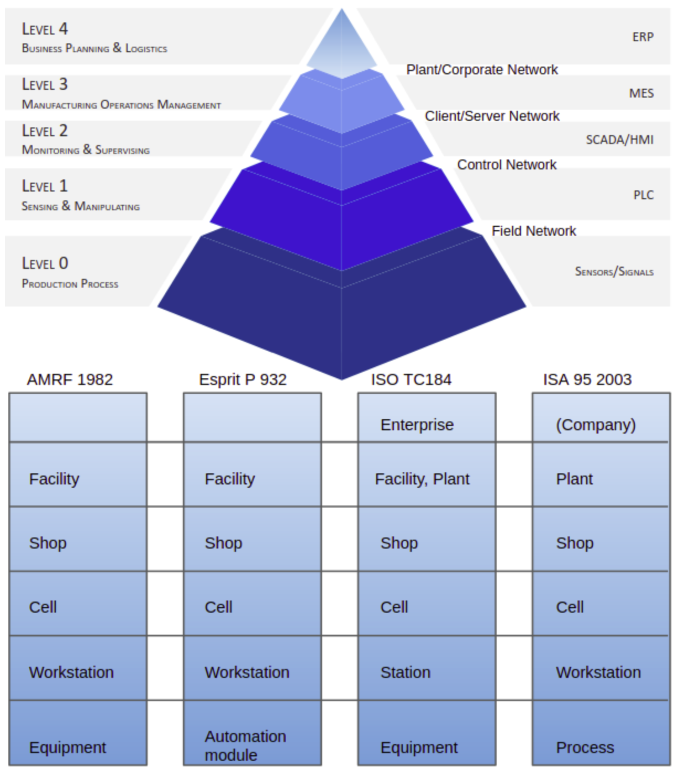


Figure 1. ISA 95 Automation Pyramid and development

Though it was difficult to convey the idea of ubiquitous connectivity during the fledgling IT and communication technologies of the 1980s, even data exchange on a broad scale within the factory was almost impossible to implement; CIM, PDM, and PLM were consequently unheard of at that time. Fortunately, the automation pyramid was established and is still evolving up to present day, as depicted in Figure 1. Eventually, the idea of product data as a basis for meaningful communication became a prerequisite for Industrial Internet of Things (IIoT) [4].

While PLM and the Digital Factory established a data backbone of IIoT, hardware for manufacturing, the evolution of mechatronics, and Cyber-Physical System (CPS) consider communication between subsystems and networked nodes as a necessary part of future mass customization manufacturing [5]. The systems described by multi-disciplined theories need to be integrated for manufacturing, which means various pieces of the system need to be implemented: networks, interfaces, databases, and various specific functions for manufacturing [4].

Overall, the trend of industrial automation is continuously evolving from local functionality to complex and systematic architecture. However, at least one area of industrial automation that deserves one's attention is how to use appropriate technologies to exchange and share information among various industrial components. For example, the Germany Industrie 4.0 [6] focuses on improving the vertical and horizontal integration of manufacturing system components. Solutions should be characterized by necessary technical, syntactic and semantic interoperability, and provide interaction-based information exchange and data communication capabilities. Among the solutions, OPC UA is one of the most up-to-date standards satisfying these requirements.

The main contribution of this paper is to present: 1) our implementation of the new extension of OPC UA, the OPC UA specification part 14 [7], and 2) an OPC UA PubSub MQTT configuration tool. The implementation includes necessary PubSub related entities, communication parameters and configuration steps. In order to facilitate the configuration of OPC UA Publisher and Subscriber, and reduce local configuration work, the OPC UA PubSub MQTT configuration tool is able to provide flexible configurations to OPC UA Publisher and Subscriber. Both the OPC UA PubSub component and the configuration tool are programmed in C and C++, so that the implementation could be easily integrated into other C/C++ projects.

This paper is organized as follows. Section II provides an overview of OPC UA and its latest specifications; part 14 OPC UA PubSub and MQTT will be introduced. Section III describes a configuration tool for parameters of MQTT and OPC UA PubSub. Finally, in Section IV, we conclude the paper and discuss future studies.

II. OPC UA AND MQTT

A. *OPC UA and OPC UA PubSub*

Classic Object Linking & Embedding for process control (OPC) is a series of specifications developed by an industrial automation task force in 1996 for providing real-time communication capabilities to the Component Object Model(COM) or the Distributed Component Object Model (DCOM) based industrial equipment. Its successor, OPC UA, was released in 2006 and integrated classic standards (e.g., OPC Data Access, OPC Historical Data Access, etc.) into a single standard. Nowadays, OPC UA is a widely recognized standard in industrial automation and is known for its interoperability, platform independence, securing communication, and scalability [8].

The latest specification of OPC UA Part 14 PubSub introduces PubSub logical components for Publisher and Subscriber, defines necessary communication parameters, and provides configuration procedures. In this section, we will present the OPC UA's interoperability, its communication capabilities, and the new added specification, PubSub.

1) Interoperability: Manufacturing companies face increasing global competition and market pressures which inversely drive them to adopt emerging and competitive technologies. However, manufacturing plants are often forced to invest a great deal of time on the installation, setup, and maintenance of such systems along with the difficulties of transferring existing automation systems to new ones and the adaptability to different suppliers' products. Therefore, more flexibility and

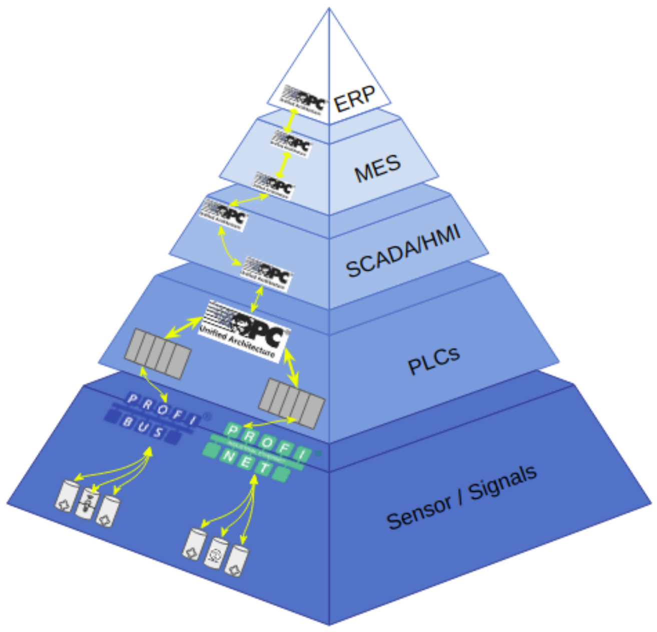


Figure 2. Automation Pyramid with OPC UA vertical and horizontal integration

adaptability within the automation system are required for future industrial automation.

Industrial automation systems based on Service-Oriented Architecture (SOA) emerged as a promising solution thanks to its advantages, such as integration capabilities and flexibility. Services provided by such components, machines, and systems are flexible and not tied to production tasks, but rather are a part of the service set, and invokable for other equipment [9].

Numerous technologies and specifications are created by following the SOA principle. For example, OPC UA, Devices Profile for Web Services (DPWS), EXI, CoAP, RESTful, Distributed Service Bus, Complex Event Processing (CEP), etc. European projects Service Infrastructure for Real-time Embedded Networked Applications (SIRENA) [10] and Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices (SOCRADES) [11] have implemented SOA-based approaches by using DPWS as a framework and base technology. Moreover, the Industrie 4.0 initiative regards interoperability as an essential part of future industrial automation, which implies industrial equipment within a plant should be able to understand, share, and process information from the equipment of different manufacturers. The Industrie 4.0 initiative has also resulted in the Reference Architecture Model Industrie 4.0 (RAMI4.0), which is a three-dimensional architecture showing structurally Industrie 4.0 related approaches and participants. The RAMI4.0 is based on SOA and ensures the vertical and horizontal integration of industrial system components [12].

The OPC Foundation's vision is to provide a new standard to bridge the differences among industrial ICT. OPC UA fulfills SOA requirements of industrial automation and has already been successfully used in several projects, automation and control systems, such as Beck-hoff TwinCAT [13], Siemens SIMATIC [14], B&R Automation Studio [15], etc.

Communication interfaces provided by OPC UA make Industrie 4.0 compliant communications between the control

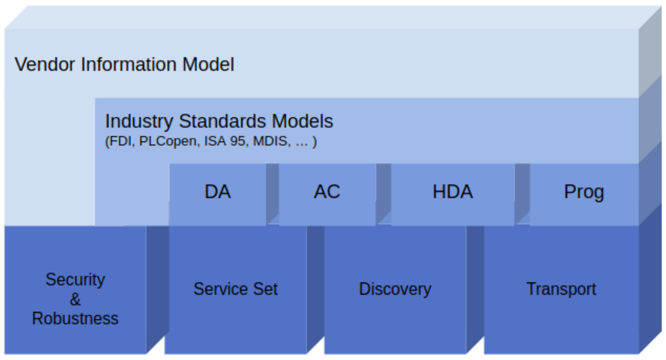


Figure 3. OPC Unified Architecture

level and MES, as well as MES and ERP possible [16] [17], as seen in Figure2. The OPC Foundation specified an object-oriented information model which can be used as a baseline to match the application's needs. The base information model is specified in OPC UA specification Part 5 (the Information Model) [18] by using instances and types defined in OPC UA specification Part 3 (the Address Space Model) [19]. OPC UA meta model uses instances and types provided by the base information model to describe other information models (e.g., Data Access, Historical Data Access, Alert and Conditions, specific industrial domain information models, etc.) and to build the OPC UA server address space. Together with OPC UA infrastructures, Service Sets, Transport, Discovery, Security and Robustness, OPC UA enables interoperability at the semantic level. The multi-layered architecture is shown in Figure 3.

Along with information model related specifications, OPC Foundation works with other organizations for providing the dedicated industrial-domain specifications, such as the OPC Unified Architecture for AutoID, the OPC Unified Architecture for MTConnect , the OPC Unified Architecture / PLCopen Information Model, etc.

For legacy COM/DCOM based OPC systems, OPC UA specification protects legacy OPC standards by using an OPC UA Wrapper software or OPC UA Proxy to enable the communication between COM/DCOM based OPC Client/Server and OPC UA based server/client [20].

2) Communication capability: Following the initial transition of industrial automation communication from dedicated automation networks (fieldbus systems) to Ethernet-based networks [21], and following industrial communication technologies with requirements, such as real-time, low-latency, etc., numerous standards and solutions have been proposed by researchers in recent decades. Industrie 4.0 poses new requirements for future manufacturing; smart factories use communication technologies to improve connectivity quality and efficiency at lower costs. These improvements can be guaranteed by two Industrie 4.0 objectives: cross-layer structural connectivity and semantic interoperability of components and systems along with open vendor-independent architecture [22]. As a result, communication takes places at every layer of the automation pyramid. It can be expected that the Internet of Things (IoT), Cyper-Physical System (CPS), Time Sensitive Network (TSN), etc., together with ICT technologies will create new opportunities for making information exchanges more efficient and comprehensive [23].

There are various industrial systems (e.g., ERP, MES, Supervisory Control And Data Acquisition [SCADA], Distributed Control System, PLCs, etc., as seen in Figure 2) that exist in industrial automation; the communication complexity among them is explicit [24]. The research in [25] presents the requirements of the Industrie 4.0 vertical and horizontal communication aspects and the author indicates communications from factory level to field level could be categorized in the following way:

- *Enterprise level and Control level.* Between the enterprise level and control level, cyclic communication with various latencies would be employed to meet the requirements of enterprise applications. The enterprise applications (e.g. the software used for tracking from materials to the final products) are usually running on PCs and providing the best-effort traffic with reliable transmission.
- *Human Machine Interface (HMI) and Control level.* HMI applications running on a PC or mobile device communicate with a control level device within latency 100ms, and provide the best effort traffic with a reliable transmission.
- *Control-to-Control (C2C).* C2C communication between high level embedded devices provides latency less than 1ms and prioritized traffic with an unreliable transmission.
- *Control and Field level.* The latency of communication between control level and field level devices is smaller than 1μs. The communication provides scheduled traffic with unreliable transmission.

According to different requirements and resources, industrial solution providers choose the Client/Server mode or the PubSub mode to implement communication between industrial devices or systems.

1) Client/Server mode
Physical industrial devices could be an OPC UA Client or an OPC UA Server or both, depending on the requirements. For example, if a field level device is used as an OPC UA Client, timely updates could be "pushed" to the OPC UA server. If a sensor is used as an OPC UA Server, physical world values can be accessed by the OPC UA Client through the server's address space. If the sensor is used as both client and server, both mechanisms mentioned above are available. However, with the connection-oriented communication, the Client/Server mode requires a great deal of resource allocation when devices are used as an OPC UA Server. This could potentially cause problems, especially for embedded devices.
2) PubSub mode
Different from the Client/Server modes connection-oriented communication and a one-to-many scenario, PubSub enables a many-to-many communication paradigm to address new, flexible, and scalable industrial application scenarios, especially for resource-constrained embedded devices.
The PubSub specification describes broker-based and brokerless-based communication between the sender (Publisher) and the receiver (Subscriber).

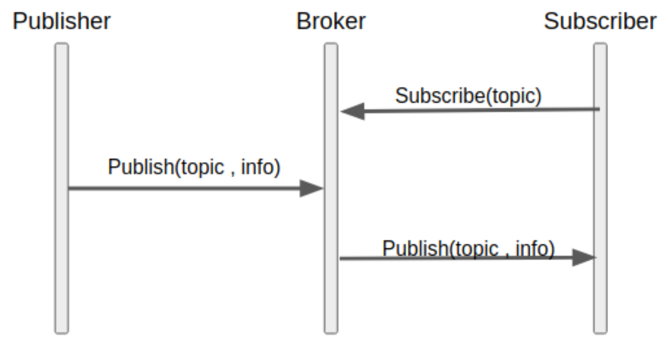


Figure 4. MQTT Publish/subscribe process

In broker-based mode, the Publisher sends a NetworkMessage to the broker. Then, the broker dispatches a NetworkMessage to subscribers according to the topic subscribed. NetworkMessages are created by using a PublishedDataSet defined DataSetMessages and topics. The broker-based mode decouples the communication between the entities in space, time, and synchronization.

In brokerless mode, the Publisher and Subscriber communication depends on the infrastructure provided by the network with less latency. Transport protocols, such as Advanced Message Queuing Protocol (AMQP), Message Queue Telemetry Transport (MQTT), OPC UA UDP, and OPC UA Ethernet can be used for message transmission.

B. *MQTT (Message Queue Telemetry Transport)*

MQTT is a topic-based Publish/Subscribe protocol. It is characterized as a simple, lightweight messaging protocol and aims at connecting resource constrained devices and low-bandwidth, high-latency or unreliable networks by using Message Oriented Middleware [26]. In Figure 4, the MQTT Subscriber sends a subscribe(topic) to inform the broker (Message Oriented Middleware) the topic which interests it, the MQTT Publisher sends a Pub(topic,data) message to the broker, if there is a match between the MQTT Publishers topic and the MQTT Subscribers topic, the broker then transfers the Pub(topic, data) message to the subscriber. The MQTT Subscriptions and publications can only be made on a specified set of topics. That means, the MQTT Publisher/Subscriber can only use the topics which are fixed during the design phase of an application.

III. OPC UA PUBSUB MQTT CONFIGURATION TOOL

In order to facilitate the distribution of data and control commands among various industrial components, many OPC UA based projects have been taking place during recent years. The research presented in [27] uses OPC UA and MQTT for SCARA robot's control command transmissions and monitoring data collections. The control commands are sent by an OPC UA Server running on computer A to an OPC UA Client integrated gateway, then be relayed to a SCARA robot. The monitoring data collected from the SCARA robot will be sent by the gateway's MQTT Publisher to a broker running on computer B for further distribution. The MQTT messages' topic is fixed in the implementation. Another research [28] presents a solution for low-level integration of

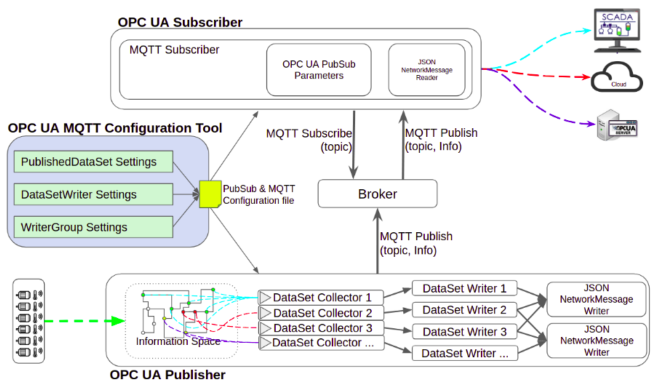


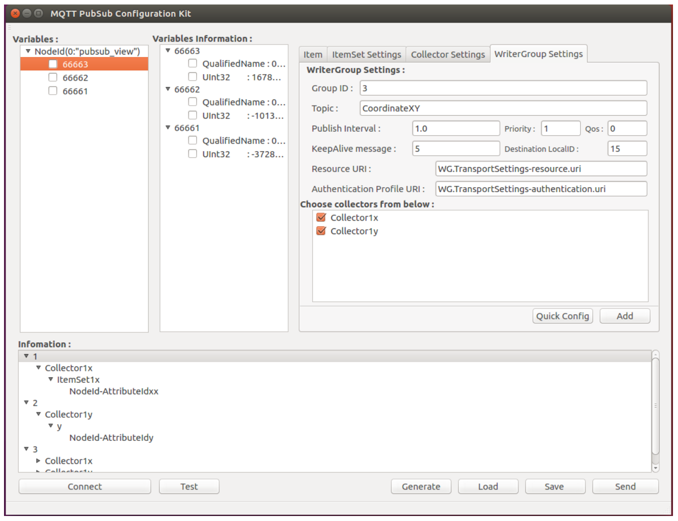
Figure 5. OPC UA Publisher&Subscriber and the configuration tool

CPSs. The proposal is described as "a Plug&Play solution that allows plugging a self-describing device producing self-described data to a larger system". The solution utilizes OPC UA Server/Client mode and Semantic Web technologies to achieve integration at various levels. The devices described in the research are OPC UA Server based sensors, and the related configuration is fixed during the development phase. Though the sensor configuration is repeatable during the design and development phase, the introduction of our OPC UA PubSub and configuration tool would provide more flexibility to device manufacturers and users to set up the OPC UA PubSub enabled devices and systems according to the specific application scenario.

A. *OPC UA PubSub Configuration Procedure*

In our implementation, the OPC UA Server is extended with PubSub related parameters and configuration procedures, thus the OPC UA Server is used as the Publisher. We use the subscriber of a MQTT implementation, Mosquitto [29], as our Subscriber. Correspondingly, the Subscriber is extended with PubSub related parameters and configuration procedures, too. The Publisher, the Subscriber, and the configuration tool are shown in Figure 5. The configuration tool is based on the OPC UA Client to which a pair of sockets is added for configuration transmission. Before the configuration tool launching, the OPC UA Publisher, the OPC UA Subscriber, and the Mosquitto broker are launched. After the connections between the configuration tool and OPC UA Publisher and Subscriber are successfully established, the connections between the Mosquitto broker and OPC UA Publisher and Subscriber would be established. Then, the operator launches the configuration tool, edits the configuration in the tool's GUI (see Figure 6) and sends it to the OPC UA Publisher and OPC UA Subscriber. The OPC UA Subscriber will subscribe with a specified topic and the OPC UA Publisher will start collecting data from the nodes, which are specified by the configuration, and creates DataSetMessages. DataSetMessages are packed into NetworkMessage for publishing.

After the security channel and session are established between the configuration tool and the OPC UA Publisher, the operator can browse nodes of a View by using a View service and Query Service provided by the OPC UA. The View is a subset of the AddressSpace, which includes the data source nodes. These data source nodes will be specified by the user in the configuration file for "pushing" data to other local or remote components of the industrial system. Available data



MQTT PubSub Configuration Kit

Variables :
- NodeId(0:"pubsub_view")
  - 66663
  - 66662
  - 66661

Variables Information :
- 66663
  - QualifiedName : 0...
  - UInt32 : 1678...
- 66662
  - QualifiedName : 0...
  - UInt32 : -1013...
- 66661
  - QualifiedName : 0...
  - UInt32 : -3728...

Item | ItemSet Settings | Collector Settings | WriterGroup Settings

WriterGroup Settings :
- Group ID : 3
- Topic : CoordinateXY
- Publish Interval : 1.0 Priority : 1 Qos : 0
- KeepAlive message : 5 Destination LocalID : 15
- Resource URI : WG.TransportSettings-resource.uri
- Authentication Profile URI : WG.TransportSettings-authentication.uri

Choose collectors from below :
- ☑ Collector1x
- ☑ Collector1y

Infomation :
- 1
  - Collector1x
    - ItemSet1x
      - NodeId-AttributeIdxx
- 2
  - Collector1y
    - y
      - NodeId-AttributeIdy
- 3
  - Collector1x

Figure 6. OPC UA PubSub MQTT Configuration Tool Graphic User Interface

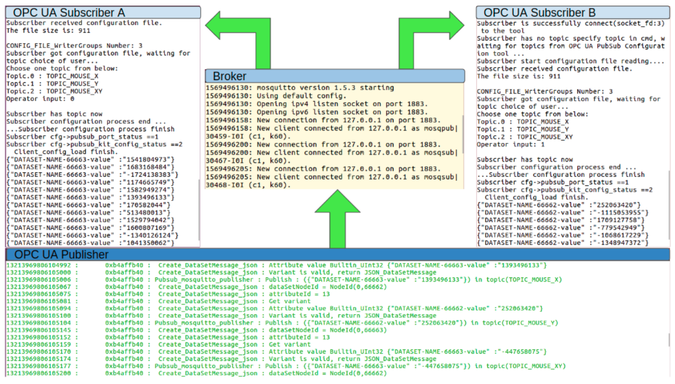
sources (nodes listed in the View) are displayed in the widget "Variables" of the GUI. The configuration tool can retrieve related nodes information (e.g., NodeID, Attributes, etc.) and display them in the widget named "Variables Information" of the GUI. Configuration parameters configured by the user can be saved locally in prefixed text format, or previewed in the "Information" widget of the GUI before being sent to the Publisher and Subscriber.

The work thread pubsub_server_config of the OPC UA Publisher uses the acquired parameters to initiate and create entities of the OPC UA Publisher, such as DataSet Collectors, DataSet Writers, and WriterGroups. Another work thread pubsub_mosquitto_publisher of the OPC UA Publisher is responsible for the output collection of field devices, creation of DataSetMessages and NetworkMessages, and publishing of MQTT. Correspondingly, the Subscriber is able to subscribe topics configured by the tool, receiving a NetworkMessage and decoding a DataSet received for further usage.

B. OPC UA PubSub Components and Parameters

The OPC UA Server and Client can be extended either as an OPC UA Publisher or Subscriber depending on the scenario. Existing PubSub entities in our implementation that require configuration include: the DataSet Collector, the DataSet Writer and Writer Group at the Publisher's end, and message topic at the OPC UA Subscriber's end. Transport protocol mappings and message mappings are grouped into different data types and/or classes. Parameter values are serialized and sent in prefixed text format to the Publisher and the Subscriber. Configuration parameter data types, such as DataSetMetaDataType, PublishedDataSetDataType are used for the OPC UA Publisher and the OPC UA Subscriber. DataSetWriterDataType and WriterGroupDataType are used in the OPC UA Publisher side. DataSetReaderDataType and ReaderGroupDataType are used in the OPC UA Subscriber. The descriptions of the functionalities of each parameter are given below:

- *DataSetMetaDataType.* It describes the content and semantics of a DataSet. The data structure DataSetMetaDataType defines the field data type and data structure of a DataSet (e.g., field name, data type, unit, etc.). The parameter is edited in the widget "Item" tab of the configuration tool's GUI. The parameter is received by both the Publisher and the Subscriber who are interested in a certain DataSet. The Publisher uses it to orderly fill collected filed values in the DataSetMessage. The Subscriber uses DataSet field types and data structure descriptions for decoding the received DataSet.
- *PublishedDataSetDataType.* It is used to configure a DataSet collector and filter of the Publisher. The parameter is configured in widget "Collector Settings" tab of the configuration tool GUI. The DataSet collection uses NodeId and AttributeId of a Node in AddressSpace to continually collect values produced by the Node, and fills the individual field of the DataSet.
- *DataSetWriterDataType.* It provides necessary information to DataSetWriter of the Publisher for the creation of the DataSetMessage from the DataSet. The parameter is configured in widget "Collector Settings" tab of the configuration tool GUI. A specified number of DataSetWriters would be created by using this parameter. A DataSetMessage is created from the DataSet specified by the PublishedDataSet. The DataSet field of a DataSetMessage may exist in different forms (Variant, DataValue, or RawData) according to the DataSetMessageContentMask of the DataSetWriter.
- *WriterGroupDataType.* This parameter configures the NetworkMessage creation process. The NetworkMessage is the container of DataSetMessages. It is configured in widget "WriterGroup Settings" tab of the configuration tool GUI. A specified number of WriterGroups would be created by using this parameter. The parameter indicates also the MQTT related parameters, such as the NetworkMessage topic of a certain WriterGroup and the Quality of Service (QoS) of the connected broker.



OPC UA Subscriber A

```
Subscriber received configuration file.
The file size is: 911

CONFIG_FILE_WriterGroups Number: 3
Subscriber got configuration file, waiting for
topic choice of user...
Choose one topic from below:
Topic.0 : TOPIC_MOUSE_X
Topic.1 : TOPIC_MOUSE_Y
Topic.2 : TOPIC_MOUSE_XY
Operator input: 0

Subscriber has topic now
Subscriber configuration process end ...
...Subscriber configuration process finish
Subscriber cfg->pubsub_port_status ==1
Subscriber cfg->pubsub_kit_config_status ==2
  Client_config_load finish.
{"DATASET-NAME-66663-value" :"1541804973"}
{"DATASET-NAME-66663-value" :"1683168484"}
{"DATASET-NAME-66663-value" :"-1724138383"}
{"DATASET-NAME-66663-value" :"1174665749"}
{"DATASET-NAME-66663-value" :"1582949274"}
{"DATASET-NAME-66663-value" :"1393496133"}
{"DATASET-NAME-66663-value" :"170582044"}
{"DATASET-NAME-66663-value" :"513480013"}
{"DATASET-NAME-66663-value" :"1529794042"}
{"DATASET-NAME-66663-value" :"1600807169"}
{"DATASET-NAME-66663-value" :"-1340126124"}
{"DATASET-NAME-66663-value" :"1041350062"}
```

Broker

```
1569496130: mosquitto version 1.5.3 starting
1569496130: Using default config.
1569496130: Opening ipv4 listen socket on port 1883.
1569496130: Opening ipv6 listen socket on port 1883.
1569496158: New connection from 127.0.0.1 on port 1883.
1569496158: New client connected from 127.0.0.1 as mosqpub|
30459-I0I (c1, k60).
1569496200: New connection from 127.0.0.1 on port 1883.
1569496200: New client connected from 127.0.0.1 as mosqsub|
30467-I0I (c1, k60).
1569496205: New connection from 127.0.0.1 on port 1883.
1569496205: New client connected from 127.0.0.1 as mosqsub|
30468-I0I (c1, k60).
```

OPC UA Subscriber B

```
Subscriber is successfully connect(socket_fd:3)
 to the tool
Subscriber has no topic specify topic in cmd, w
aiting for topics from OPC UA PubSub Configurat
ion tool ...
Subscriber start configuration file reading....
Subscriber received configuration file.
The file size is: 911

CONFIG_FILE_WriterGroups Number: 3
Subscriber got configuration file, waiting for
topic choice of user...
Choose one topic from below:
Topic.0 : TOPIC_MOUSE_X
Topic.1 : TOPIC_MOUSE_Y
Topic.2 : TOPIC_MOUSE_XY
Operator input: 1

Subscriber has topic now
Subscriber configuration process end ...
...Subscriber configuration process finish
Subscriber cfg->pubsub_port_status ==1
Subscriber cfg->pubsub_kit_config_status ==2
  Client_config_load finish.
{"DATASET-NAME-66662-value" :"252063420"}
{"DATASET-NAME-66662-value" :"-1115053955"}
{"DATASET-NAME-66662-value" :"1709127758"}
{"DATASET-NAME-66662-value" :"-779542949"}
{"DATASET-NAME-66662-value" :"-1068617229"}
{"DATASET-NAME-66662-value" :"-1348947372"}
```

OPC UA Publisher

```
13213969806104992 :       0xb4affb40 :  Create_DataSetMessage_json : Attribute value Builtin_UInt32 {"DATASET-NAME-66663-value" :"1393496133"}
13213969806105000 :       0xb4affb40 :  Create_DataSetMessage_json : Variant is valid, return JSON_DataSetMessage
13213969806105006 :       0xb4affb40 : Pubsub_mosquitto_publisher : Publish : ({"DATASET-NAME-66663-value" :"1393496133"}) in topic(TOPIC_MOUSE_X)
13213969806105067 :       0xb4affb40 :  Create_DataSetMessage_json : dataSetNodeId = NodeId(0,66662)
13213969806105075 :       0xb4affb40 :  Create_DataSetMessage_json : attributeId = 13
13213969806105081 :       0xb4affb40 :  Create_DataSetMessage_json : Get variant
13213969806105094 :       0xb4affb40 :  Create_DataSetMessage_json : Attribute value Builtin_UInt32 {"DATASET-NAME-66662-value" :"252063420"}
13213969806105100 :       0xb4affb40 :  Create_DataSetMessage_json : Variant is valid, return JSON_DataSetMessage
13213969806105104 :       0xb4affb40 : Pubsub_mosquitto_publisher : Publish : ({"DATASET-NAME-66662-value" :"252063420"}) in topic(TOPIC_MOUSE_Y)
13213969806105145 :       0xb4affb40 :  Create_DataSetMessage_json : dataSetNodeId = NodeId(0,66663)
13213969806105152 :       0xb4affb40 :  Create_DataSetMessage_json : attributeId = 13
13213969806105159 :       0xb4affb40 :  Create_DataSetMessage_json : Get variant
13213969806105170 :       0xb4affb40 :  Create_DataSetMessage_json : Attribute value Builtin_UInt32 {"DATASET-NAME-66663-value" :"-447658075"}
13213969806105174 :       0xb4affb40 :  Create_DataSetMessage_json : Variant is valid, return JSON_DataSetMessage
13213969806105177 :       0xb4affb40 : Pubsub_mosquitto_publisher : Publish : ({"DATASET-NAME-66663-value" :"-447658075"}) in topic(TOPIC_MOUSE_XY)
13213969806105200 :       0xb4affb40 :  Create_DataSetMessage_json : dataSetNodeId = NodeId(0,66662)
```

Figure 7. Configured OPC UA Publisher and OPC UA Subscriber

IV. OPC UA PubSub communication experiment

Figure 7 shows a validation experiment of data distribution between our configured OPC UA Publisher and OPC UA Subscribers. The OPC UA Publisher is configured by the configuration tool for publishing real-time mouse coordinates. The coordinates data are produced by dynamic libraries loaded after the OPC UA Publisher is launched. Coordinates x and y are described in an XML file and mapped into the address space as Variables. Their NodeIds are 66663 and 66662, respectively. The coordinates are published in three topics: “Topic MOUSE X, “Topic MOUSE Y, and “Topic MOUSE XY. The OPC UA Subscribers A and B show that they received topics from the configuration tool and are waiting for the operators topic choice for each OPC UA Subscriber. In our experiment, we let the OPC UA Subscriber A subscribe with topic MOUSE X and OPC UA Subscriber B subscribe with Topic MOUSE Y. Then, the OPC UA Subscribers send topic-subscriptions to the broker and start receiving MQTT messages that interest them.

V. Conclusion and future work

Firstly, this paper provided a brief history of ICT applied in the manufacturing field, from CIM to present-day Industrie 4.0. It also introduced OPC UA and related application scenarios. Then, we introduced its new specifications, the OPC UA Part 14 PubSub, and highlighted two OPC UA’s specific features, namely, the interoperability and communication capabilities. This was followed by a description of a configuration tool that we propose and its implementation is undergoing. The configuration tool can be easily added to any C/C++ OPC UA implementation. We tested it with our own implementation of an OPC UA Publisher and Subscriber. But our work is not integrated into other OPC UA implementations due to access limit to their nodes and attributes. We presented some necessary configuration parameters for applying OPC UA PubSub communication following Part 14 of the specifications. Future research and implementation are planned on the performance test of OPC UA PubSub, supporting of asynchronous message queues, and cybersecurity through Network Message encryption and decryption. When compared with manually editing the configuration for each OPC UA Server/Client and loading it locally, the configuration tool can provide flexible and modifiable configurations to an OPC UA Publisher/Subscriber.

Acknowledgment

This work is part of the project WIZARD, a FEDER-FSE, Operational Program of CORSE project.

References

[1] J.-Q. Li et al., “Industrial internet: A survey on the enabling technologies, applications, and challenges,” IEEE Communications Surveys & Tutorials, vol. 19, no. 3, 2017, pp. 1504–1526.

[2] S. Terzi et al., “Product lifecycle management-from its history to its new role,” International Journal of Product Lifecycle Management, vol. 4, no. 4, 2010, pp. 360–389.

[3] T. Sauter, S. Soucek, W. Kastner, and D. Dietrich, “The evolution of factory and building automation,” IEEE Industrial Electronics Magazine, vol. 5, no. 3, 2011, pp. 35–48.

[4] S. Jeschke, C. Brecher, T. Meisen, D. Özdemir, and T. Eschert, “Industrial internet of things and cyber manufacturing systems,” in Industrial Internet of Things. Springer, 2017, pp. 3–19.

[5] H. Xu, W. Yu, D. Griffith, and N. Golmie, “A survey on industrial internet of things: A cyber-physical systems perspective,” IEEE Access, vol. 6, 2018, pp. 78 238–78 259.

[6] “Industrie 4.0,” 2019, URL: https://www.plattform-i40.de/PI40/Navigation/EN/Industrie40/WhatIsIndustrie40/what-is-industrie40.html [retrieved: Sep. 2019].

[7] “Opc 10000-14 - part 14: Pubsub,” 2018, URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-14-pubsub/ [retrieved: Sep. 2019].

[8] “Opc ua technology,” 2019, URL: https://opcfoundation.org/about/opc-technologies/opc-ua/ [retrieved: Sep. 2019].

[9] F. Jammes and H. Smit, “Service-oriented paradigms in industrial automation,” IEEE Transactions on industrial informatics, vol. 1, no. 1, 2005, pp. 62–70.

[10] H. Bohn, A. Bobek, and F. Golatowski, “Sirena-service infrastructure for real-time embedded networked devices: A service oriented framework for different domains.” in ICN/ICONS/MCL, 2006, p. 43.

[11] D. S. et al., “Socrades: A web service based shop floor integration infrastructure,” in The internet of things. Springer, 2008, pp. 50–67.

[12] “Reference architectural model industrie 4.0 (rami4.0),” 2019, URL: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.html [retrieved: Sep. 2019].

[13] “Beck-hoff twincat,” 2019, URL: https://www.beckhoff.com/twincat/ [retrieved: Sep. 2019].

[14] “Siemens simatic,” 2019, URL: https://w3.siemens.com/mcms/topics/en/simatic/pages/default.aspx [retrieved: Sep. 2019].

[15] “Br automation,” 2019, URL: https://www.br-automation.com/en/technologies/opc-ua/opc-ua-br/ [retrieved: Sep. 2019].

[16] M. Müller, E. Wings, and L. Bergmann, “Developing open source cyber-physical systems for service-oriented architectures using opc ua,” in 2017 IEEE 15th International Conference on Industrial Informatics (INDIN). IEEE, 2017, pp. 83–88.

[17] M. Melik-Merkumians et al., “Towards opc ua as portable soa middleware between control software and external added value applications,” in Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012). IEEE, 2012, pp. 1–8.

[18] “Opc 10000-5 - part 5: Information model,” 2019, URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-5-information-model/ [retrieved: Sep. 2019].

[19] “Opc 10000-3 - part 3: Address space model,” 2019, URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-3-address-space-model/ [retrieved: Sep. 2019].

[20] “Upgrade your legacy opc systems to opc ua,” 2019, URL: https://opcconnect.opcfoundation.org/2017/12/upgrade-your-legacy-opc-systems-to-opc-ua/ [retrieved: Sep. 2019].

[21] J.-D. Decotignie, “Ethernet-based real-time and industrial communications,” Proceedings of the IEEE, vol. 93, no. 6, 2005, pp. 1102–1117.

[22] O. Givehchi, K. Landsdorf, P. Simoens, and A. W. Colombo, “Interoperability for industrial cyber-physical systems: An approach for legacy systems,” IEEE Transactions on Industrial Informatics, vol. 13, no. 6, 2017, pp. 3370–3378.

[23] M. Wollschlaeger, T. Sauter, and J. Jasperneite, “The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0,” IEEE Industrial Electronics Magazine, vol. 11, no. 1, 2017, pp. 17–27.

[24] A. Balador, N. Ericsson, and Z. Bakhshi, “Communication middleware technologies for industrial distributed control systems: A literature review,” in 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE, 2017, pp. 1–6.

[25] A. Eckhardt, S. Müller, and L. Leurs, “An evaluation of the applicability of opc ua publish subscribe on factory automation use cases,” in 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), vol. 1. IEEE, 2018, pp. 1071–1074.

[26] “Mqtt,” 2019, URL: http://mqtt.org/faq [retrieved: Sep. 2019].

[27] T. Mizuya, M. Okuda, and T. Nagao, “A case study of data acquisition from field devices using opc ua and mqtt,” in 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE). IEEE, 2017, pp. 611–614.

[28] V. Jirkovský, M. Obitko, P. Kadera, and V. Mařík, “Toward plug&play cyber-physical system components,” IEEE Transactions on Industrial Informatics, vol. 14, no. 6, 2018, pp. 2803–2811.

[29] “Mosquitto,” 2019, URL: https://mosquitto.org/ [retrieved: Sep. 2019].