OPC UA Integration for Field Devices

Armin Veichtlbauer, Martin Ortmayer, Thomas Heistracher
Information Technology & Systems Management
Salzburg University of Applied Sciences, Puch/Salzburg, Austria
Email: {armin.veichtlbauer, mortmayer.itsb-m2014, thomas.heistracher}@fh-salzburg.ac.at

***Abstract*—In industrial automation systems, deploying the well-established automation pyramid model is best practice. However, the trend to massively distributed systems, which are foreseen to co-operate using standardized protocols and common semantics, shows the limits of these traditional approaches. In order to enable for Industry 4.0 compliant solutions, appropriate means for scalable internetworking have to be developed and utilized. New modelling technologies have been developed to represent such distributed automation systems, incorporating a multidimensional layered approach. At higher hierarchy levels of these automation models, standardization approaches are quite common. However, at the field layer there are still many different field busses, which in most cases do not allow common semantics, but come along with their own object models. In contrast, the use of OPC UA at the field level, with its standardized protocol stack and semantic annotations, would allow for enabling field devives to fully participate in large scaled systems as Industry 4.0 components. This paper evaluates the potentials and limits of integrating OPC UA into legacy field devices with limited communication and calculation resources and provides quantitative measurement results of selected test scenarios.**

I. Introduction

Automation systems, as described in [1], interface the real physical world by use of sensors and actuators. Automated control tasks are performed by controllers, while some backend devices are used for human supervisory. Whereas the communication between controllers and backend devices is usually based on state of the art protocol stacks (having a bigger footprint, but allowing for added value data processing), the connection to field devices is still commonly based on field bus systems (having a much lower footprint, but lacking added values as complex data structures and semantic annotations). This leads to the use of gateways between the different parts of the automation system, which most likely comes along with loss of information.

Through the recent development of "Internet of Things", i.e., bringing the IP stack into field devices (e.g., by using technologies as 6LoWPAN [2]), the chance to seamlessly integrate field devices with backend devices emerges. However, IP is just part of the solution; in order to exchange data from automation systems of different vendors, semantic annotation of exchanged data points and standards for describing complex data structures are necessary. A technology which gained much attention (e.g., [3]) in this context is "Open Platform Communications Unified Architecture" (OPC UA) [4]. The use of OPC UA in all devices of an automation system would allow for a seamless integration of all parts of the system without information loss.

However, the use of OPC UA at field level is still not widely common. Thus, the research work at hand tries to answer following questions: To what extent are resources used for the OPC UA information model and basic functionality to work as specified (including latency issues)? Is the OPC UA meta model able to represent dynamic field devices and to replace object models of traditional field buses? Can state of the art security profiles be provided?

In order to answer these research questions, a protyotype had to be realized, fulfilling following functional requirements:

- The prototype had to support the address space and node classes defined in [4]
- The services for reading and writing attributes had to be fully functional
- The services for managing subscriptions to "monitored items" had to be fully functional
- A service for calling added value methods including the exchange of appropriate input and output parameters had to be fully functional
- Objects from legacy field busses had to be represented as OPC UA nodes, and creation and deletion had to be possible at runtime

Additionally, some non-functional requirements had been defined, concerning resource usage, influence on other (control and non-control) traffic, security mechanisms, as well as hard- and software environments, as basis for the prototypical implementation.

The rest of this paper is organized as follows: Chapter II gives a short overview of related technologies and their utilization in research and industry. The prototypical architecture is shown in chapter III, followed by implementation and validation issues in chapter IV. The results of the conducted analysis are presented in chapter V. Finally, in chapter VI a short conclusion is given, and some potential further research activities are named.

II. Related Work

Distributed automation systems consist of a number of interlinked devices, i.e., sensors, actuators, controllers, and backend devices. These devices can be logically grouped along their degree of abstraction, reaching from concrete physical interaction with the surrounding world to abstract business oriented systems as enterprise ressource planning (ERP), which uses data from the underlying systems. This layered system is usually called "automation pyramid", as described in [5].

A more complex modelling has been introduced by the German Electrical and Electronic Manufacturers Association (ZVEI) named "Reference Architecture Model Industry 4.0" (RAMI 4.0) [6]. As shown in Fig. 1, it sets up a three-dimensional layered system consisting of the dimensions abstraction layers (from devices to business), hierarchy levels (from field devices and products to the interconnected world), as well as product lifecycle. It is designed for scalability, connectivity, and multi-vendor integration [7], thus overcoming limitations of the classical automation pyramid which is rather single-vendor.

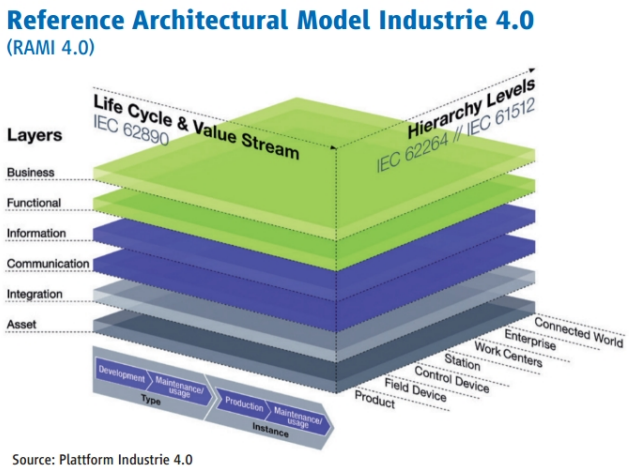

Fig. 1. RAMI 4.0 [8]

Multi-vendor integration, however, raises the need for interoperability of data and services within this model [9]. Components have to be able to interact in a well-defined manner, thus evolving to "Industry 4.0" components. This again requires a common semantic description for interconnected components, which is called "administration shell" [10] in the context of Industry 4.0.

For devices belonging to the upper hierarchy levels of the RAMI 4.0 model (i.e., the more abstract ones), the use of proper semantic descriptions of exchanged data is already quite common [5]. Devices in these layers are normally connected via TCP/IP and have enough calculatoric power to cope with technologies which have a certain footprint. Thus, the use of loosely coupled systems and service-oriented architectures (SOA) allows for the exchange of complex structured data [3], e.g., based on XML.

On the other side, field devices at the lower layers of the hierarchy often have low-power constraints, narrowband connectivity, and real-time bounds. The need to "optimize every bit" hampers the usage of complex and elaborated communication and data processing technologies. Information has to be concentrated on a few bits, thus leading to strongly coupled systems. In the field layer, this is normally done using field busses [5], which have a limited set of variables, representing data points in pre-defined "objects models", e.g., Modbus RTU / TCP [11].

In spite of these resource limitations, solutions derived from "classical" networking gained more interest in recent years. Many field devices got IP capable, often using IPv6 in conjunction with header compression technologies (e.g., 6LoWPAN [2]) to use sparse resources efficiently. Also, Ethernet based technologies can be used to exchange control frames in a real-time manner, e.g., via Ethernet Powerlink [12].

Thus, the next step is to bring semantics to the field level. As mentioned, the most promising candidate [13] for this purpose seems to be OPC UA [4]. OPC UA is a SOA that has been defined by the OPC Foundation (an industry consortium) as successor of the classical OPC technology, which is based on the "Distributed Component Object Model" (DCOM) and thus limited to Microsoft Windows ©. The idea of OPC was to provide a standardized "application programming interface" (API), e.g., for SCADA applications, abstracting concrete devices via appropriate device drivers.

As for the transport protocol, a binary variant utilizing TCP/TLS can be used as well as an XML variant utilizing HTTP. The main innovation for automation systems yet lies in the definition of an informational meta-model, which allows to structure a model-world representation of all real-world devices of an existing automation system. The OPC UA information model enables a semantic description also of the according data points, i.e., of the underlying object models of legacy field busses [1]. Thus, OPC UA is well-suited when it comes to cascading field busses.

All such individual mappings, e.g., between Ethernet Powerlink and CANopen, are obsolete, once a mapping to a highly complex information model as provided by OPC UA is done. The semantic annotation thus allows for usage in massively distributed multi-vendor RAMI 4.0 environments. Furthermore, it is considered as important technology in the smart grid domain, e.g., [14]. However, to be widely accepted in the automation industry, real-time capabilities are required. For that purpose, the OPC foundation is working on an extension of OPC UA to integrate real-time services in combination with "time-sensitive networking" (TSN), a set of IEEE 802.1 standards, e.g., [15].

Several publications have researched the potential of OPC UA to get integrated into field devices, e.g., [16] have shown that integration of an OPC UA server in field devices can be performed with very low footprint; however, this result has been achieved by relinquishing dynamic changes in the information model. In [17], performance issues especially of the OPC UA security model have been examined, whereas in [18] Devices Profile for Web Services (DPWS) have been compared with OPC UA. None of the mentioned publications, however, have explored the interplay of OPC UA traffic with real time traffic.

III. Prototype Architecture

In order to answer the aforementioned research questions, a prototypical solution had to be set up, which served as the basic platform for all test scenarios described in sections IV and V.

The prototype was based on an Altera Cyclone I FPGA [19] equipped with a proprietary 32 bit soft core CPU with a clock rate of 50 MHz, as well as with 4 MB flash and 2 MB static Random-Access Memory (RAM). The board was supposed to be representative for many field devices to justify that results obtained in the conducted validation scenarios can be taken as significant for the field level in general.

As operating system, FreeRTOS [20] was chosen as it provides real-time capabilities, has very low resource needs, and is open source. FreeRTOS basically consists of a CPU specific porting, a task scheduler which allows for pre-emption, and an API. However, in order to better support the envisaged test scenarios, some additional functionality had to be added, as shown in Fig. 2: First, a real-time clock had been implemented to provide time stamping with 100 ns accuracy; additionally, NTP synchronization was integrated. Second, a command line interface (CLI) had been realized to be able to control the test runs as needed. Finally, a dynamic memory allocator had been added for storing OPC UA nodes at run-time.

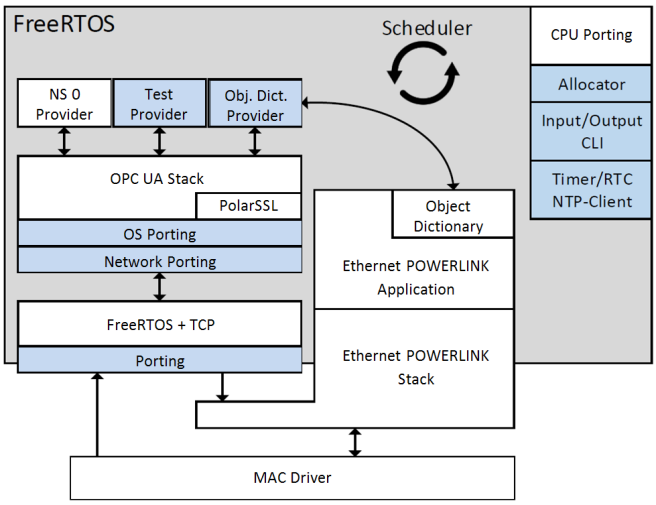

Fig. 2. Software Components on Altera Cyclone Platform

The prototype platform already came along with a MAC controller supporting Ethernet Powerlink [12]. Ethernet Powerlink enables a co-existance of real-time and none real-time traffic; as OPC UA currently lacks real-time capabilities, the prototype had to use the asynchronous parts of Ethernet Powerlink. Consequentely, the chosen setup was independent on all parallel real-time communication.

As the binary transport version of OPC UA requires a TCP/IP stack, the FreeRTOS+TCP [21] stack in version 160112 was added to the existing configuration. This stack supports Address Resolution Protocol (ARP) and Dynamic Host Control Protocol (DHCP). To bind the TCP/IP stack to the already existing MAC driver, a porting layer had to be implemented, which receives frames from the driver and proceeds them to the TCP/IP stack. For sending frames, they must be forwarded to the Ethernet Powerlink stack to ensure that they are sent in the appropriate time slot.

Furthermore, the OPC UA server itself, along with mbed TLS (formerly known as PolarSSL) [22] to allow for encrypted communication, had to be integrated in the prototypical setup. The OPC UA stack was provided by Unified Automation under the label "High Performance OPC UA SDK" [23]. It is written in C and is intended for usage in low-resource embedded systems. Again, a porting layer had to be provided to link the OPC UA stack to the prototype's operating system, which includes access to RTOS's thread synchronization and security features. Also, the basic networking functionalities are integrated via a porting layer. Thereby, the callback interface of FreeRTOS+TCP was linked to the asynchronous network API of the Unified Automation stack for allowing non-blocking communication.

As the core component of the test setup, the OPC UA stack had to be equipped with several "data providers", i.e., components which provide data from one or more defined "name spaces" to the OPC stack. The name spaces have to be used to separate different application's naming of unique data points from each other. The OPC UA stack comes along with an integrated data provider with a pre-defined name space, called the "Name Space 0 Provider", which is used pre-dominantely for accessing static elements of the OPC UA address space in the device's Read-Only Memory (ROM).

For using this setup as testbed of OPC UA integration in field devices, two other data providers had to be implemented. First, a test provider was used to add validation nodes to the OPC UA address space. Second, the "Object Dictionary" data provider is used to provide the data from Ethernet Powerlink's object dictionary to the OPC UA server. This data provider is crucial for the test system's efficiency; especially the storage utilization is dependent on the concrete implementation, as will be shown in the next section.

IV. Implementation of Object Dictionary Provider

The simplest way to implement a data provider for the Ethernet Powerlink object model would be to define one additional OPC UA node for each object by using already integrated standard libraries. However, the Ethernet Powerlink device allows for 253 additional hardware modules extending the system under test, each of which containing 45 or more objects. As one OPC UA node which is administered by standard libraries requires at least 92 Byte of RAM, the total RAM utilization would sum up to a minimum of $253 * 45 * 92 = 1.047.420$ Byte [1], which consumes about the half of the available RAM. Thus, for the implementation of the prototype, a more efficient solution had to be found.

For doing so, the necessary data had to be provided dynamically, e.g., when an OPC UA client browses the instance of the information model. As all Ethernet Powerlink objects can be identified by a two Byte "index" and a one Byte"subindex", a three Byte "Node-ID" can be calculated via the simple formula $Node - ID = index * 256 + subindex$. With this Node-ID, OPC UA can identify and address each node; all other data can be provided by the data provider just at the time of the query.

When these nodes are added into the OPC information model, they have to be placed at a suitable position of the existing object hierarchy. Figure 3 shows the part of the OPC UA information model, where they can be mounted. The specification extension [24] defines an object with the browse name "DeviceSet", which groups all devices the OPC server should contain, i.e., the Ethernet Powerlink device, which is called "EplDevice" in the information model depicted in Fig. 3. It contains a node "ObjectDictionary", where the original Ethernet Powerlink object dictionary is mapped, i.e., objects are grouped along their "index" and "subindex".

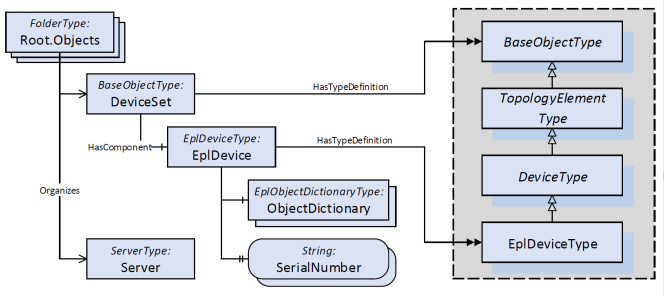

Fig. 3. Object Dictionary within OPC UA Information Model [24]

After having realized the OPC UA integration with the used board, a testbed as shown in Fig. 4 had been set up in order to validate that the access to the onboard OPC UA server was possible and fulfilled the basic requirements. A PLC was connected to the system under test ("Prototype") via an Ethernet Powerlink connection. A Powerlink analysing device ("X20ET8819") [25] was put in-between to monitor the Powerlink traffic. This device was also able to timestamp the frames with an accuracy of 20 ns. The results of the tests were sent to a PC, on which the analysis could take place. The test PC was also used to schedule and start the tests; the switch was used to get remote access to the PC.

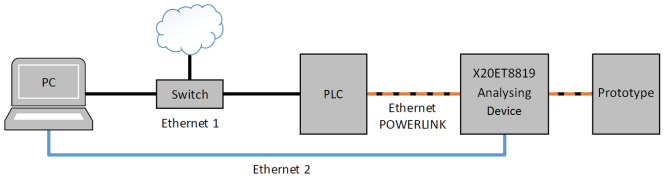

Fig. 4. Test Environment

Both the PLC and the PC were equipped with an OPC UA client for querying and browsing the OPC server instance of the prototype. Hereby, the free OPC UA client "UaExpert" from Unified Automation [26] had been used at the querying PC; the PLC came along with an integrated OPC UA client. For some of the tests (especially for the parallel queries), an additional PC was integrated in network segment Ethernet 1, while the analysing device was unused. For validation, the following tests had been conducted:

First, the browsing of the OPC information model was tested by accessing some already known nodes. Then, setting and getting values via OPC UA was tested via attributes of test nodes, where valid and unvalid values were used. Whereas the valid values could successfully be written with the OPC client, the unvalid values yielded errors. Also, subscriptions to OPC UA "monitored items" were tested with test nodes. These incremented a test variable every second and published its value to the client. Hereby, a parallel query from up to five client instances had been performed. Furthermore, dynamically adding and removing nodes in the OPC information model could successfully be realized. Finally, some tests were made to assess the usage of Microsoft's security guideline "Basic128RSA15" [27].

V. Performance and Efficiency Evaluation

The prototypical setup as described in the previous chapter had been used to conduct some measurements of performance and efficiency of the OPC UA integration, along with a comparison with values of similar tests without the use of OPC UA. Hereby, the tests covered three aspects of interest: Response times, memory utilization, and CPU utilization.

A. Response Times

Table I depicts the results of measuring the response times. The time measured was the difference between query and response.

TABLE I
Response Times in Milliseconds

	None			Basic128RSA15		
	Avg	Min	Max	Avg	Min	Max
Read 1	3.6	3.49	3.72	6.84	6.69	7.2
Read 10	5.22	5.19	5.49	11.21	11.09	11.32
Read 63	15.45	14.79	15.61	36.3	35.59	38.29
Write 1	3.6	3.59	3.72	6.6	6.39	6.92
Write 10	4.81	4.79	4.92	9.44	9.2	9.6
Write 63	12.91	12.39	13.09	26.37	25.99	26.69

Hereby, an object with the index 0x8000 and 200 subindexes had been defined; thus, 201 OPC UA nodes with NodeIDs ranging from 0x800000 to 0x8000C8 had to be instantiated. Reading and writing access was then conducted with a single object, 10 objects in parallel, and 63 objects in parallel; each of these six scenarios was conducted 100 times, calculating minimum, maximum, and average values for the response times. The measurements of the response times for reading and writing access to OPC UA nodes have been conducted using security guidelines "None" and "Basic128RSA15", as shown by the respective columns in Table I.

Additionally, similar queries have been made without the use of OPC UA, i.e., directly to the Ethernet Powerlink object dictionary using the Powerlink "Service Data Object" (SDO) [12]. However, these reference tests could only be performed with a single object, as conjoint access to more than one object

is not yet implemented with SDO. For a single query, the response times of OPC UA without security guideline are about twice as high than with SDO; by using "Basic128RSA15", the values again are about doubled. Although the values are clearly increasing with the number of objects, the increase is lower that linear, i.e., conjoint access is worth the effort.

B. Memory Utilization

Table II shows the memory utilization of the different software parts of the system under test, using four different test scenarios: SDO access without OPC UA was compared with the use of OPC UA; in both cases a static variant where memory is used by the code image was compared to a dynamic variant with additional memory needs.

TABLE II
MEMORY UTILIZATION IN BYTE

	Without OPC UA		With OPC UA	
	Stat.	Dyn.	Stat.	Dyn.
Basic Components	76 202	326 130	90 106	340 034
FreeRTOS	8 643	25 767	9 041	30 793
FreeRTOS + Extensions	23 069	23 649	23 277	23 925
FreeRTOS + TCP	65 645	73 493	66 429	74 773
Ethernet Powerlink	129 082	247 014	135 215	253 147
OPC UA Server	0	0	271 766	758 698
Data Provider Namespace 0	0	0	225 788	225 788
Data Provider Object Dir.	0	0	17 010	17 414
PolarSSL	0	0	141 215	141 803
Sum	302 641	696 053	979 847	1 866 375

The conducted test scenarios included up to 500 OPC UA monitored items with up to ten parallel subscriptions. Monitored items provide a publish/subscribe service, where changes of data point values exceeding a configurable interval are reported to the querying OPC UA client. The values of Table II refer to the maximum numbers. These parameters have influence on the allocated dynamic memory; however, the allocation takes place at the initialization of the OPC UA stack, i.e., in advance of the actual tests.

Thus, the actual number of used monitored items and subscriptions has no influence on the memory utilization; yet, the number of sessions linked to the server has impact. Especially the component FreeRTOS+TCP will require additional memory for each session with linked clients. The values given in Table II had been obtained without any clients linked to the server. The security features were activated, which resulted in a memory requirement of about 200 000 Byte; the main part of that memory utilization is due to the integration of the PolarSSL functionality.

C. CPU Utilization

Also for the assessment of the CPU utilization, OPC UA monitored items had been used. Figure 5 shows the CPU utilization dependent on the number of monitored items.

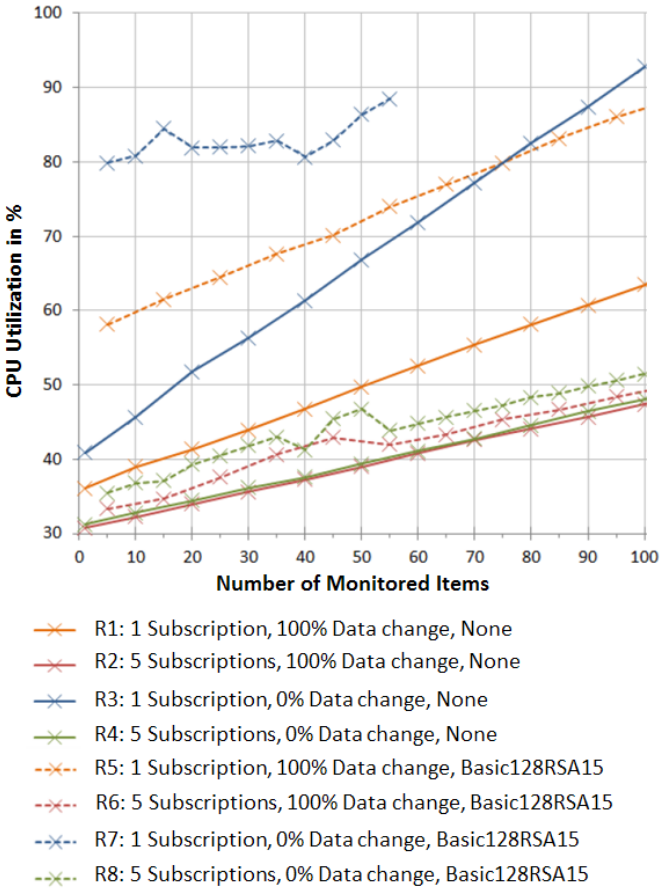

Fig. 5. CPU Utilization in Selected Scenarios

Besides the number of monitored items, further influencing parameters had been taken into account: the number of subscriptions, the sampling rate, the number of changed values of data points, and the used security guideline. For the results shown in Fig. 5, the sampling rate was configured to 50 ms, i.e., the server sent out the new values every 50 ms in the case of changes. For the cases without any changes in the monitored data, a keep alive counter of 10 was configured, i.e., every 500 ms the server sent the values in any case. All of the tested scenarios show a near linear dependency on the number of monitored items.

VI. CONCLUSION AND FURTHER WORK

With the research work at hand we have analyzed, under which conditions the use of OPC UA in field devices is a feasible undertaking. The advantages can be clearly seen in the seamless integration of field devices and backend devices, alongside with the potential to use OPC UA's semantic power to build up big scaled multi-vendor automation systems. It has been proved that this usage is possible, even when other

(real-time constrained) control tasks have to be performed on the same network resources. However, the rather big footprint of OPC UA has also its disadvantages for field devices, as resource utilization and communication latencies are concerned. This tradeoff could be expected in advance; however, the work at hand gives some quantitative results which can help engineers to decide whether the use of OPC UA is feasible or not in their respective environment.

As devices tend to become more powerful, also in embedded setups, the use of OPC UA will be more and more reasonable. One of the most important current drawbacks of OPC UA is the lack of real-time capabilities, as this is a pre-condition for most automation systems. Thus, the ongoing work on real-time capable OPC UA [15] is a very interesting recent development in this context, as this would address the timing issues and hence broaden the applicability of this technology. Once this is available, a similar evaluation would be very desirable, first for testing the fulfillment of real-time constraints, and second concerning the other conditions under which such approaches can be deployed in real world environments.

The semantic interoperability offered by OPC UA in industrial field devices is an important asset in context of IoT proliferation [28]. Multi-vendor setups are common in this field, which are in need of a conceptual basis for system integration and service orchestration. OPC UA can provide these central roles and can also support the distribution of "intelligence" [29] that is currently on its way in the light of distributed (micro) services. In order to manage semantics in massively distributed systems, the use of ontologies [30] will lead to interesting new research fields.

ACKNOWLEDGMENT

This work is funded by the Austrian Ministry for Transport, Innovation and Technology (bmvit) and the Austrian Research Promotion Agency (FFG) under the "ICT of the Future" program in the OpenNES project (FFG No. 845632).

REFERENCES

[1] M. Ortmayer, "OPC UA at the Field Layer," Master's thesis, University of Applied Sciences Salzburg, Salzburg, Austria, Jan. 2017, in German.

[2] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*. John Wiley & Sons, 2011, vol. 43.

[3] M. Melik-Merkumians, T. Baier, M. Steinegger, W. Lepuschitz, I. Hegny, and A. Zoitl, "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 and Factory Automation (ETFA 2012)*, Krakow, Sep. 2012.

[4] OPC Foundation. (2012) OPC – The Interoperability Standard for Industrial Automation & Other. [Online]. Available: http://www.opcfoundation.org

[5] M. Hollender, *Collaborative Process Automation Systems*. International Society of Automation (ISA), 2010.

[6] Plattform Industrie 4.0, "Status Report: Reference Architecture Model Industrie 4.0 (RAMI4.0)," ZVEI German Electrical and Electronic Manufacturers Association, Tech. Rep., Jul. 2015. [Online]. Available: http://www.vdi.de

[7] M. Weyrich and C. Ebert, "Reference Architectures for the Internet of Things," *IEEE Software*, vol. 33, no. 1, pp. 112–116, 2016.

[8] Plattform Industrie 4.0, "Umsetzungsstrategie Industrie 4.0," ZVEI German Electrical and Electronic Manufacturers Association, Tech. Rep., Apr. 2015, in German.

[9] J. Åkerberg, M. Gidlund, and M. Björkman, "Future research challenges in wireless sensor and actuator networks targeting industrial automation," in *Proceedings of the 9th IEEE International Conference on Industrial Informatics (INDIN 2011)*, Jul. 2011, pp. 410–415.

[10] Plattform Industrie 4.0, "Structure of the Administration Shell," ZVEI German Electrical and Electronic Manufacturers Association, Tech. Rep., Apr. 2016. [Online]. Available: https://www.plattform-i40.de

[11] X. Hong and W. Jianhua, "An extendable data engine based on OPC specification," *Computer Standards & Interfaces*, vol. 26, no. 6, pp. 515–525, 2004.

[12] *Ethernet POWERLINK Communication Profile Specification DS301*, Ethernet POWERLINK Standardisation Group Std., 2016. [Online]. Available: http://www.ethernet-powerlink.org/en/downloads/technical-documents/

[13] M. Stopper and B. Katalinic, "Service-oriented Architecture Design Aspects of OPC UA for Industrial Applications," in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol II (IMECS 2009)*, Hong Kong, Mar. 2009.

[14] S. Rohjans, "(S2)In – Semantic Service Integration for Smart Grids," Ph.D. dissertation, Carl von Ossietzky University, Oldenburg, Sep. 2012.

[15] *802.1Qbu-2016 - IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amendment 26: Frame Preemption*, Time-Sensitive Networking Task Group Std., 2016. [Online]. Available: https://standards.ieee.org/findstds/standard/802.1Qbu-2016.html

[16] J. Imtiaz and J. Jasperneite, "Scalability of OPC-UA down to the chip level enables "Internet of Things"," in *Proceedings of the 11th IEEE International Conference on Industrial Informatics (INDIN 2013)*, Jul. 2013, pp. 500–505.

[17] O. Post, J. Seppälä, and H. Koivisto, "The Performance of OPC-UA Security Model at Field Device Level," in *Proceedings of the 6th International Conference on Informatics in Control, Automation, and Robotics (ICINCO 2009)*, Milan, Jul. 2009.

[18] G. Candido, F. Jammes, J. B. de Oliveira, and A. W. Colombo, "SOA at device level in the industrial domain: Assessment of OPC UA and DPWS specifications," in *Proceedings of the 8th IEEE International Conference on Industrial Informatics (INDIN 2010)*, Jul. 2010.

[19] Intel Corporation. (2017) FPGA – Cyclone Series. [Online]. Available: https://www.altera.com/products/fpga/cyclone-series.html

[20] Real Time Engineers Ltd. (2014) FreeRTOS. [Online]. Available: http://www.freertos.org

[21] FreeRTOS Labs. (2016) FreeRTOS+TCP. [Online]. Available: http://www.freertos.org/FreeRTOS-Plus/FreeRTOS_Plus_TCP/

[22] ARM. (2016) ARM mbed. [Online]. Available: https://tls.mbed.org/

[23] Unified Automation GmbH. (2016) High Performance OPC UA Server SDK. [Online]. Available: https://www.unified-automation.com/products/server-sdk/highperf-ua-server-sdk.html

[24] OPC Foundation. (2017) OPC Unified Architecture for Devices (DI) - Companion Specification. [Online]. Available: https://opcfoundation.org/developer-tools/specifications-unified-architecture/opc-unified-architecture-for-devices-di

[25] Bernecker + Rainer. (2017) X20ET8819. [Online]. Available: http://www.br-automation.com/downloads_br_productcatalogue/BRP4440000000000000480708/X20ET8819-ENG_V1.05.pdf

[26] Unified Automation GmbH. (2017) UaExpert – A Full-Featured OPC UA Client. [Online]. Available: https://www.unified-automation.com/products/development-tools/uaexpert.html

[27] Microsoft. (2016) SecurityAlgorithmSuite.Basic128Rsa15 Property. [Online]. Available: https://msdn.microsoft.com/en-us/library/system.servicemodel.security.securityalgorithmsuite.basic128rsa15(v=vs.110). aspx

[28] S. Back, S. Kranzer, T. Heistracher, and T. Lampoltshammer, "Bridging SCADA Systems and GI Systems," in *Proceedings of the IEEE World Forum on Internet of Things (WF-IoT 2014)*, Seoul, Korea, Mar. 2014.

[29] T. Lampoltshammer, E. Pignaton de Freitas, T. Nowotny, S. Plank, J. da Costa, T. Larsson, and T. Heistracher, "Use of Local Intelligence to Reduce Energy Consumption of Wireless Sensor Nodes in Elderly Health Monitoring Systems," *Sensors*, vol. 14, no. 3, pp. 4932–4947, 2014.

[30] T. Lampoltshammer and T. Heistracher, "Ontology Evaluation with Protg using OWLET," *Infocommunications Journal*, vol. 6, no. 2, pp. 12–17, 2014.