MQTT Service Broker for Enabling the Interoperability of Smart City Systems

Anatolijs Zabasta
Institute of Industrial Electronics and Electrical Engineering
Riga Technical University
Riga, Latvia
Anatolijs.Zabasta@rtu.lv

Nadezda Kunicina
Institute of Industrial Electronics and Electrical Engineering
Riga Technical University
Riga, Latvia
Nadezda.Kunicina@rtu.lv

Kaspars Kondratjevs
Institute of Industrial Electronics and Electrical Engineering
Riga Technical University
Riga, Latvia
Kaspars.Kondratjevs@rtu.lv

Antons Patlins
Industrial Electronics and Electrical Engineering
Riga Technical University
Riga, Latvia
Antons.Patlins@rtu.lv

Leonids Ribickis
Institute of Industrial Electronics and Electrical Engineering
Riga Technical University
Riga, Latvia
Leonids.Ribickis@rtu.lv

Jerker Delsing
Dept. of Computer Science, Space and Electrical Engineering
Lulea University of Technology
Lulea, Sweden
Jerker.Delsing@ltu.se

Abstract— Smart city offers a concept of interconnection of modern digital technologies in the context of a city that provides a solution to enhance the quality and performance of urban services. However, interconnection of the Smart city systems still is a challenging process due to incompatibility of systems that apply a plenty of appropriate technical solutions and protocols that cause collaboration between automation devices such as sensors, actuators, controllers. Introduction of Internet-of-Things (IoT) provides promising opportunities to develop new services and integrate different application domains. The Arrowhead Framework aims to apply Service Oriented Architecture to the embedded systems' world. This research is focused on the one of the Arrowhead Framework core systems - Event Handler. This system supports the handling of events, and enriches service-oriented applications with the capabilities of interacting via the publish/subscribe paradigm. We implemented the Event Handler system as a MQTT enabled service broker, and deployed data flow programming tool Node-RED for wiring together divergent hardware devices and nodes, and APIs for online services. A case study of the service broker implemented for control of utilities systems in urban environment is presented and discussed.

Keywords— Smart city systems, utilities, automation, SOA, MQTT broker, Arrowhead Framework core systems.

I. INTRODUCTION

At present a lot of IT systems are deployed in the urban environment, which collaborate to make cities smarter, e.g. water distribution, waste management, traffic management, smart grid, building management systems, public safety, routing information, etc. However, many of those systems originate from different providers, and they are maintained by distinct public and private agents. Very often the systems use own computational infrastructure, and work in isolation [1]. Additionally, utilities systems (power supply, heat, water suppliers, building services providers, etc.) apply different automated meter reading tools, and methods for collecting metering data from the sensors, controlling the actuators and maintenance of the industrial processes. Usually, utilities systems automation is presented mostly by nodes, where SCADA controls water pumps stations or sewage water treatment equipment, which operate as automation island at the distribution networks.

One of the main problems with utilities systems automation is that once it's outdated it is very hard to evolve these systems when new requirements and newer technology becomes available. Maintenance cost, lack of common standards, time consuming installation and configuration of the networks is the other challenge. Thus, an approach that pursues migration from legacy technology and contributes for system upgrades to meet new requirements and easy migration to newer technologies is needed.

In a broader sense, we can consider utility's systems automation in the context of the Smart City systems interoperability. In this context the IoT has been recognized as a promising solution for many Smart City systems issues such as public utility systems automation [2], efficient transportation systems [3], centralized healthcare [4], efficient waste management, smart grids, digital tourism, etc. The business opportunity is to offer an advanced architecture of the Smart City systems based on service oriented architecture (SOA) [1], [5], [12], IoT approach and advanced industrial automation standards that provide highly compatibility with existing and emerging solutions.

In order to address complexity and scale of Smart City systems a Cloud computing as a solution is offered by [6], [13]. Different cloud service platforms for IoT have been offered. For example, the researchers are focused on how to build and scale a cloud-based IoT middleware that can be used across a broad range of Smart City research and in particular, the use of Smart City data hubs as a central approach to building urban-scale IoT systems [6]. Derhamy et al. [36] categorized a number of cloud service platforms as global cloud, peer-to-peer and local cloud. Furthermore, often "fog" or hybrids of global and internal clouds are positioned in between the global and local cloud paradigms.

The local cloud concept developed by the Arrowhead Framework [19] addresses many challenges related to IoT-based automation, and is unique in its support for integration of applications between secure localized clouds. The approach is that IoT devices are abstracted as services in order to enable interoperability between IoT devices [15].

Various researchers exploit traditional distributed systems approaches such as general message passing or data sharing approaches [7]. A macro-programming model that is able to oversee the network of distributed sensors as a whole rather than program individual pieces is suggested by [8]. Giang [9] and Blackstok at al. [10] offered an advanced version of dataflow model to express application logic in a Smart City setting that refers as "Adaptive Distributed Dataflow" as more suitable for large scale IoT.

In this paper, we discuss an approach for utility's systems interconnection in Smart cities, based on SOA when sensors data exchange is made through the services. The proposed technology is based on the Arrowhead Framework [19]. The proposed solution is based on the services provided by the Event Handler system.

The proposed developed smart services broker applies Visual data flow programming approach [10]. Node-RED [11] was selected among other tools to prove viability and advantages of offered architecture implemented as a local automation cloud of Smart city systems. The services broker resides on a gateway, which provides adapter and protocols translation functions, and on a tool for wiring together hardware devices, APIs for online services.

The paper is structured as follows. An overview of related works is done in Chapter 2. Authors give a brief description of the Arrowhead core systems in Section 3. Implementing Arrowhead Event Handler core system and description of the service broker is discussed in Section 4. The main contribution of the research and possible directions for future works are discussed in the Chapter 5.

II. AN OVERVIEW ON RELATED WORKS

The novel concept Industry 4.0 is based on the logic of Cyber Physical Production Systems (CPPS) that enables distributed intelligence for independent processes and production networks, which self-optimize, communicate with each other and optimize the production as a whole [16], [17]. In the work [18] the Industrial Internet Reference Architecture (IIRA) is presented as a standards-based open architecture under the Industrial Internet Consortium for designing Industrial Internet Systems. Based on ISO/IEC 42010 standard specifications for complex systems with multiple components and multiple interconnected systems IIRA defines the most important industrial internet architecture components. The IoT Architectural Reference Model [34] is defined by the European IoT-A project, which aims to handle all IoT-related issues. IoT-A provides high level models in order to specify a reference architecture.

The Arrowhead architecture approach in some way utilizes the COBRA model proposed by Gross [35]. The design, development and verification methodology for each service, system and system of system is described within the Arrowhead Framework. The papers [19] and [22] presents an overview of the framework together with its core elements - and provides guidelines for the design and deployment of interoperable, Arrowhead-compliant cooperative systems. Albano [37] described one of the Arrowhead core systems, i.e. the Event Handler (EH) system, as a part of the Arrowhead Framework, which aims to apply SOA to the embedded systems' world. The EH supports the handling of events, and in that sense it enriches SOA with the capabilities of interacting via the publish/subscribe paradigm.

For practical implementation of Event Handler system we considered several Data Flow Systems such as Blockly [41], [42], a client-side JavaScript library for creating visual block programming languages and editors. It is a project of Google and is open-source under the Apache 2.0 License. The other system is a Domoticz [43], an open-source Home Automation System, which lets to monitor and configure various devices such as lights, switches, temperature, rain, wind, UV and meters (electric, gas, water).

III. ARROWHEAD CORE SYSTEMS OVERVIEW

A. Arrowhead Mandatory Core Systems

The enabling of local cloud automation functionality requires that Arrowhead-compatible systems must use the mandatory Core Systems and their Core Services provided by the Arrowhead Framework (AF) to realize their operational targets to communicate with other systems using dedicated application protocols. There are two main groups of the Core Systems: the mandatory ones that need to be presented in each Arrowhead Local Cloud, and the automation supporting ones that further enhance the core capabilities of a Local Cloud [20], [22].

Three mandatory Core Systems were defined by AF such as Service Discovery system, Authorization and Authentication system and Orchestration system. The Service Discovery system stores all the Systems that are currently available in the network and their service offerings. Systems have to announce their presence, and the services they can offer. The registry takes note of this information, when systems come on-line, and might have to revoke them when they go off-line. The Authorization and Authentication system – as its name suggests – manages authentication and authorization (AA) tasks, however, it covers some other security-related issues as well (e.g. certificate handling). The Orchestration system is responsible for instrumenting each system in the cloud: where to connect and what to consume. It instructs systems by pointing towards specific Service Providers to consume specific Service(s) from. This has to be done by a simple request-response sequence, which ends with the requester System receiving a Service endpoint. After these, the System is obligated to consume from that Service instance [19].

There are further automation supporting Core Systems available according to AF. These either were defined during the 2nd generation or currently being integrated in the framework as 3rd generation systems. Based on additional

requirements collected from five industrial domains, additional local cloud feature enabling system aspects important for implementation of robust automation systems have been defined [22]:

- *Quality of Service system* - e.g. latency measurement and control.
- *Event handling system* - e.g. event filtering, alarm handling.
- *Historian system* - e.g. data logging and auditing.
- *Deployment system* - e.g. system and service deployment.
- *Configuration system* - e.g. system and service configuration.
- *Meta data system* - e.g. non-functional information about a system or service.
- *Factory design system* - e.g. integration of physical layout and functional automation model.

Arrowhead Local Clouds are generally autonomous and independent from each other, since they are deployed separately for various reasons. However, there are specific cases, where Systems from different clouds need to consume Services from one another. To this end, an inter-Cloud servicing architecture was introduced in [23] based on two, new automation supporting Core Systems: *Gatekeeper and the Network Manager*.

B. The Role of Arrowhead Event handler system

The Event Handler System [36], [37] provides functionality for the handling of events that occur in an Arrowhead network, as is depicted in Fig. 1. The Event Handler receives events from Event Producers and dispatches them to registered Event Consumers. The Event Handler logs events to persistent storage, registers producers and consumers of events and applies filtering rules (configured by Event Consumers) to events distribution. An Event Consumer is a component that consumes events.

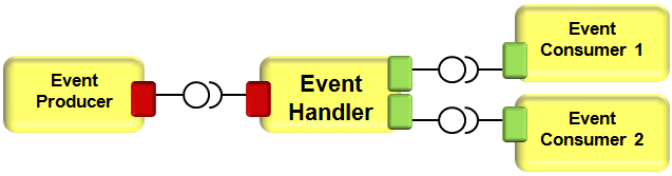

Fig. 1. Connection between Event Handler, Event Consumer and Producer (Adopted from [36])

The filtering rules can be based on message content, on applied constrains or based on the severity level of the message, which varies from Debugging to Critical. The Event Handler can also be connected to an internal or external database system which is responsible for the permanent storage of all events [37].

The Event Handler system can handle a variety of communication protocols applied by IoT embedded devices, such as MQTT (Message Queuing Telemetry Transport) [38], [40], CoAP [39] (Constrained Application Protocol) and REST (Representational state transfer), and can decode commonly used semantics, e.g. SenML (Sensor Markup Language), XML, JSON (JavaScript Object Notation), and plain text. EH system should be able to convert between protocols in a message exchange between different users. For example, a device can push data to the Event Handler CoAP or MQTT, while clients can either use MQTT, or poll data using HTTP.

IV. Implementing Arrowhead Event Handler Core Systems

In our research we developed Event Handler System as a service broker, which enables SOA based services and data flow between divergent type of embedded devices and nodes, for example: sound, strain, water flow, water pressure, outdoors/ indoor temperature sensors, etc., as well as services for control of ventilation and humidity actuators. To a certain extent, such types of sensors and actuators are typical for monitoring and control of Smart City systems and utility networks (Water supply, District heating, Building maintenance systems). Additionally, the service broker demonstrates how Arrowhead Event Handler system can be implemented although in restricted, however a real urban environment.

The Event Handler system' working prototype is located and maintained at the "Ventspils Digital Center" (VDC) servers cloud. VDC belongs to Ventspils City Council as an institution responsible for development and maintenance of ICT infrastructure in the Ventspils city, Latvia. SME "Smart Meter" – the partner of the Riga Technical University in the Arrowhead project – is responsible for sensors installation and maintenance at client' sites and premises.

We implemented Event Handler core system as a MQTT service broker that applies a Software integration platform Node-RED to interconnect heterogeneous systems in IoT way. Node-RED is a Web-based visual interface for connecting and building compositions of hardware devices, APIs, and online technical and social services [25].

The service broker applies Message Queuing Telemetry Transport (MQTT) protocol, which is a Client Server subscribe messaging transport protocol. It is lightweight, open, simple, and designed so as to be easy to implement. These characteristics make it suitable for constrained environments such as for communication in Machine-to-Machine (M2M) and Internet of Things (IoT) contexts [24]. Sensor data are sent using MQTT protocol to the Node-RED in the visually structured IoT user-friendly format. It allows to configure the connection between the devices and incoming MQTT messages, and also send those data to databases. IoT-generated data, such as sensor data from multiple devices, are stored in MongoDB [31] and MySQL databases.

The advantage of the use of a document-oriented database is due to offering a dynamic schema [26], because new data with different structures can be accommodated within the database, along with earlier data, without the need for large scale data manipulation on the whole database [27]. MongoDB is also a good choice for storing JSON encoded

data. It internally stores data in an efficient binary JSON format (BSON) [28]. It also exhibits a better runtime performance of simple operations of small-scale [29].

Emoncms, an open-source web-app for processing, logging and visualizing energy, temperature and other environmental data [30] is used for further data processing. Sensor data like payloads to be processed to MongoDB and MySQL databases within Emoncms solution [21]. Fig. 2 shows sensor data (payloads) flow using sequence diagram, in which sensor data (payloads) are processed to MongoDB database.

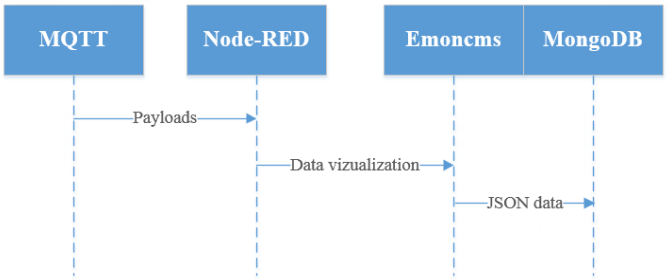

Fig. 2. Sequence diagram for sensor data (payloads) flow

Authentication of REST API connection to MongoDB is provided through SSL securing REST API. Database is configured to accept only SSL authentication. It provides nowadays security standards for sensitive data transport through internet [32]. However, APIs calls do not use authorization for getting sensor temperature' measurement data. Any user knowing APIs URL could get those measurement data. In the future, the levels of security could be based on the same SSL security authorization using two levels authorization: firstly, client user name or e-mail and password, secondly API key, which is signed with client side API or in other words API server IP. Fig. 3 shows client API call for data request using a sequence diagram, in which client using API call (POST) via REST API gets measurement data.

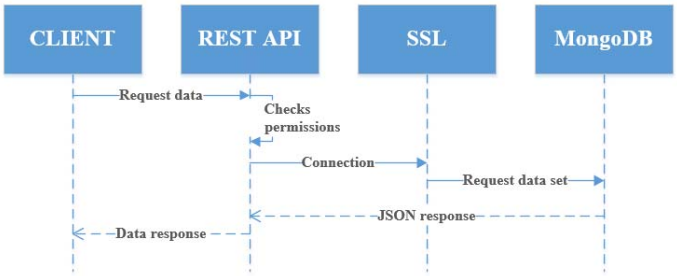

Fig. 3. Data request of a client API call

Before saving information for analysis and representation, pre-calculation like impulse value adjustments for water and electricity sensors and offset values are done to calculate the actual sensor readout value (see Fig.4).

In MongoDB, it is possible to select only necessary data rather than selecting all data of a document, e.g. if a document has five fields, but only 3 of them are necessary, one can select only 3 fields.

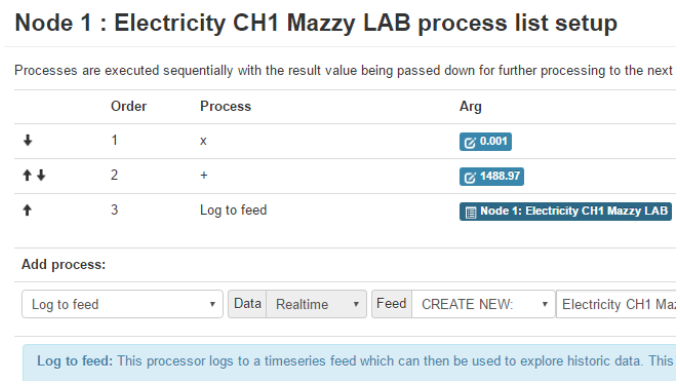

Fig. 4. Pre-calculation algorith imlementation in Emoncms

In order to enable meter data recording in MongoDB, a block "Query Emocms API" is created, in which a HTTP response returns as a parsed JSON object, otherwise msg.payload is not developed and a response is a null (see Fig.5 and Fig.6).

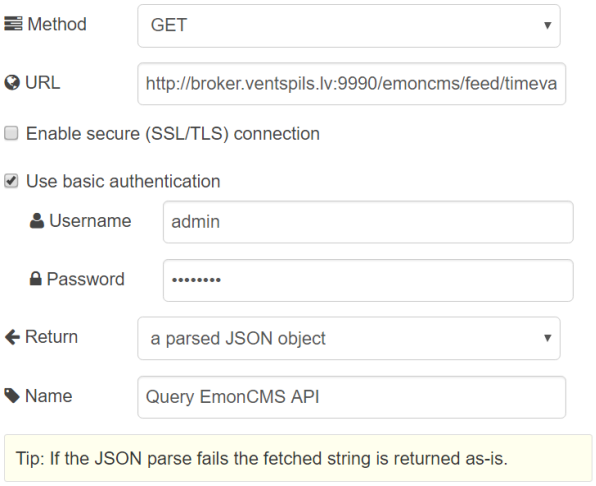

Fig. 5. An algorithm of the block for Query Emoncms API

MongoDB' "*find*" method accepts an optional parameter, which is a list of fields that one wants to retrieve. To limit the number of retrieved fields, it is necessary to set a list of fields with value 1 or 0. "One" is used to show the filed while "zero" is used to hide the field.

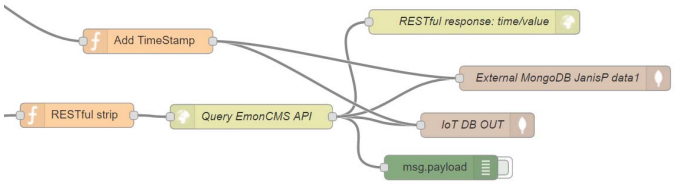

Fig. 6. The block for Query Emoncms implementation in NodeRED

Further, we consider an example of the Service broker connect a third party control application, thus demonstrating IoT system by wiring other independent service providers.

For the ventilation and humidity control two applied services are used: CO^2 and Humidity at selected sensors

locations in several cottages in Ventspils city. A power control hardware and software of ITEAD company was used, which is a cloud-based system [33]. We applied a power relay Sonoff capable to handle loads up to 10A with a maximum rating of 2200Watts (see Fig. 7).

Fig. 7. Sonoff relay connection to ventilation and air conditioner

This Wi-Fi enabled smart relay is able to communicate using MQTT or CoAP protocols. It interacts in a bidirectional way and performs smart monitoring and control functions. Two systems are controlled – a TwinFresh Comfo RA1-50, ventilator with regeneration system, and a dehumidifier Ballu BDH-35L. Emoncms is used for processing incoming data feeds, triggering response events or response streams to control external applications, provide new status data and combine measurement results.

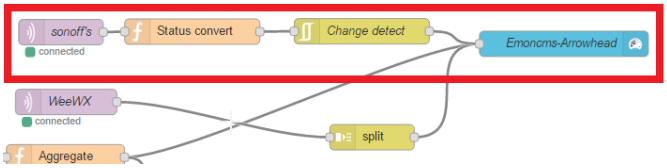

Fig. 8. Control of the status of Sonoff relay and controlled equipment

Node-RED posts data to Emoncms. The msg.payload can contain either a comma separated list of name value pairs e.g. name:value, or a comma separated list of values e.g. 1,2,... or a simple java script object e.g. msg.payload = {temp:12}. Insertion time can be manipulated by setting msg.time. This must be in epoch format - seconds since 1970.

Sonoff relays publish a status topic and send messages if their state is changed. The service broker feeds status data into Emoncms for bidirectional monitoring in case of uncontrolled state changes (e.g., a user turns off the device, power outages etc.). The status of the relays and the controlled equipment is visualized in Emoncms dashboard.

Sonoff relay handles the status of a published topic of controlled devices. Each device unique identifier is the subtopic, which corresponds to the device MAC address (all Sonoffs are equipped with a Wi-Fi interface). A "Status convert" node extracts the MAC address and builds a new MQTT message with a corrected structure by stripping colon symbols that might interfere with processing systems (see Fig. 8). A "Change detect" node – blocks repeated MQTT messages and enables the flow only if the last message topic and payload differs from the previous, thus throttles the processing load if the active elements (in this case Sonoffs) are also handled by other systems.

After triggering the Sonoff relay, a new status is published to the MQTT broker. The topics and values can differ as the manufacturer of the third party system can implement his own topic structure. For example: to turn on the ventilation the MQTT topic/payload used is: upes20/ventilation/1; the received status information is in form: stat/18:FE:34:CB:F6:FB/POWER 1.

V. Discussions, Conclusions and Future Work

Maintenon, nine clients, who represent small real estate service companies and private housekeepers, use the control system based at the service broker. Approximately three hundred divergent smart meters, gateways and actuators have been installed in several Latvian cities. In a certain approach, the service broker demonstrates the ways how to interact and communicate with the devices across multiple layers and hierarchies of networks, in a cloud of utilities services through internet, GSM or other networks directly to the device, using SCADA or other automation systems, and existing technologies.

There is no doubt that deployment of Node-RED as a distributed data flow tool to wire sensors, nodes and gateways enables time and cost saving in comparison with traditional programming methods. However, we need to create reliable indicators aiming to evaluate performance of network connectivity and cost of setup and maintenance of such monitoring and control systems.

However, some issues are needed to be discussed further. We implemented the Event Handler as MQTT based services within Arrowhead framework, however, the semantics of the Event Handler should be investigated more carefully.

Additionally, further research efforts should be devoted to security issues for authentication and authorization of different systems in the Smart City systems local cloud, as this issues have not been considered in details in this research.

References

[1] F. Lopes, S. Loss, A. Mendes, T. Batista, R. Lea, SoS-centric Middleware Services for Interoperability in Smart Cities Systems, Proceedings of the 2nd International Workshop on Smart, Article No. 4, Trento, Italy — December 12 - 16, 2016. ACM New York, NY, USA, 2016. P.1-6.

[2] M. Kunickis, A. Dandens, U. Bariss, Justification of the Utility of Introducing Smart Meters in Latvia. Latvian Journal of Physics and Technical Sciences. Volume 52, Issue 6, 1 December 2015, Pages 13-21.

[3] I. Alps, M. Gorobecs, A. Beinarovica, A. Ļevcenkovs. Immune Algorithm and Intelligent Devices for Schedule Overlap Prevention in Electric Transport. No: 2016 57th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Latvia, Riga, 13.-14. October, 2016. Riga: IEEE, 2016, 1.-7.lpp.

[4] E. Sultanovs, A. Skorobogatjko,A. Romānovs, Centralized Healthcare Cyber-Physical System's Architecture Development. In: Proceedings of

the 2016 57th International Scientific Conference on Power and Electrical Engineering of Riga Technical University, Latvia, Riga, 13-14 October, 2016. Riga: RTU Press, 2016, pp.153-158.

[5] D. Alessandrelli, M. Petraccay, and P. Pagano. T-Res: Enabling reconfigurable in-network processing in IoT-based WSNs. In Proceedings - IEEE International Conference on Distributed Computing in Sensor Systems, DCoSS 2013, pages 337-344, 2013.

[6] R. Lea and M. Blackstock. City Hub: A Cloud-Based IoT Platform for Smart Cities. In 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, pages 799-804, 2014.

[7] A. Krylovskiy, M. Jahn, and E. Patti. Designing a Smart City Internet of Things Platform with Microservice Architecture. 2015 3rd International Conference on Future Internet of Things and Cloud, pages 25-30, 2015.

[8] R. Newton, Arvind, and M. Welsh. Building up to macroprogramming: An intermediate language for sensor networks. In 2005 4th International Symposium on Information Processing in Sensor Networks, IPSN 2005, volume 2005, pages 37-44, 2005.

[9] N.Giang, R. Lea, M. Blackstock, V. C.M. Leung, On Building Smart City IoT Applications: a Coordination-based Perspective, in Proceeding SmartCities '16, December 12-16, 2016, Trento, Italy pages 1-6.

[10] Blackstock, M. and Lea, R. 2014. Toward a Distributed Data Flow Platform for the Web of Things (Distributed Node-RED). Proceedings of the 5th International Workshop on Web of Things (New York, NY, USA, 2014), 34–39.

[11] Node RED Programming Guide. Programming the IoT. http://noderedguide.com. [Accessed in March 2018].

[12] S. Karnouskos, A. W. Colombo, F. Jammes, J. Delsing, T. Bangemann, Towards an architecture for service-oriented process monitoring and control, in: 36th Annual Conference of the IEEE Industrial Electronics Society (IECON-2010), Phoenix, AZ., 2010.

[13] A. W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, E. Jose L. Martinez Lastra, Industrial Cloud-Based Cyber-Physical Systems - The IMC-AESOP Approach, Springer, 2013.

[14] "The next generation of information and communication technologies (ICT) research state program", http://www.lumii.lv/resource/show/761, [accessed in December 2016]

[15] F. Blomstedt, L. Ferreira, M. Klisics, Chrysoulas, C., de Soria, I., Zabašta, A., Moris, B., Eliasson, E., 2014. "The Arrowhead Approach for SOA Application Development and Documentation". In: Proceeding 40th Annual Conference of the IEEE Industrial Electronics Society (IECON 2014), United States of America, Dallas, 29 Oct-1 Nov., 2014. Dallas: The Institute of Electrical and Electronics Engineers (IEEE), pp.2637-2637.

[16] L. Monostori, "Cyber-physical production systems: roots, expectations and r&d challenges," Procedia CIRP, vol. 17, pp. 9–13, 2014.

[17] R. Drath and A. Horch, "Industrie 4.0: Hit or hype?" IEEE industrial electronics magazine, 2014.

[18] I. I. Consortium et al., "Industrial internet reference architecture," Industrial Internet Consortium, Tech. Rep., June, 2015.

[19] J. Delsing, ed., IoT based Automation - made possible by Arrowhead Framework. CRC Press, Taylor & Francis Group, 2016.

[20] C.Hegedus, D.Kozma G. Soos and P.Varga, Enhancements of the Arrowhead Framework to refine inter-cloud service interactions. Industrial Electronics Society, IECON 2016 - 42nd Annual Conference of the IEEE. 5

[21] Zabašta, A., Kondratjevs, K., Kuņicina, N., Pekša, J., Ribickis, L., Čaiko, J. Smart Municipal Systems and Services Platform Development. No: Proceedings of the 2016 17th International Conference on Mechatronics – Mechatronika (ME) 2016, Czech, Prague, 7-9th December, 2016, 323.-329.p

[22] P. Varga, F. Blomstedt, L. Ferreira, J. Eliasson, M. Johansson, and J. Delsing, "Making system of systems interoperable - the core components of the Arrowhead Technology Framework," IEEE Internet of Things Journal, vol. accepted for publication, August 2015. P.1-24

[23] P. Varga and C. Hegedus, "Service interaction through gateways for inter-cloud collaboration within the arrowhead framework," in 5th International Conference Wireless VITAE, 2015, 1-7.p.

[24] ISO/IEC 20922:2016. [Online]. Available: http://www.iso.org/iso/catalogue [accessed December 2016].

[25] N. K. Giang, M. Blackstock, R. Lea, V. C.M. Leung, "Developing IoT Applications in the Fog: a Distributed Dataflow Approach," in 5th International Conference on the Internet of Things (IoT), University of British Columbia, 2015.

[26] Internet of Things : MongoDB. [Online]. Available: https://www.mongodb.com/use-cases/internet-of-things69466 [accessed December 2016].

[27] T. M. Connolly, C. E. Begg, Database Systems: A Practical Approach to Design, Implementation, and Management, SIXth edition. Paisley, Scotland: University of the west of Scotland, 2015.

[28] Z. Parker, S. Poe, and S. V. Vrbsky, Comparing NoSQL MongoDB to an SQL DB. No: Proceedings of the 2013 51st ACM Southeast Conference on ACMSE '13 2013. New York, NY, USA, 4-6th April, 2013, 1.-6.p.

[29] Emoncms. [Online]. Available: https://emoncms.org/ [accessed December 2016].

[30] MongoDB. [Online]. Available: https://docs.mongodb.com/manual/ [accessed December 2016].

[31] J. Han, H. E, G. Le, Survey on NoSQL Database. No: 6th International Conference on Pervasive Computing and Applications (ICPCA) 2011. Port Elizabeth, South Africa, 26-28th October, 2011, 363.-366.p.

[32] Z. H. Zhang, J. Xue-Feng, J. J. Li, W. Jiang, An Identity-Based Authentication Scheme in Cloud Computing. No: 2012 International Conference on Industrial Control and Electronics Engineering (ICICEE) 2012. Xi'an, China, 23-25th August, 2012, 984.-986.

[33] Smart Home, ITEADE, https://www.itead.cc/smart-home.html. [Accessed in September 2017].

[34] Bassi, A., Bauer, M., Fiedler, M., Kramp,T., van Kranenburg,R., Lange, S., Meissner, S., 2013. Enabling Things to Talk – Designing IoT solutions with the IoT Architectural Reference Model. Springer.

[35] H-G. Gross, "Component-Based Software Testing with UML", Springer-Verlag Berlin Heidelberg, ISBN 3-540-26733-6, 2005.

[36] Derhamy, H., Eliasson, J., Delsing, J., Priller, P., 2015. A survey of commercial frameworks for the internet of things. In: IEEE the 20th International Conference on Emerging Technologies and Factory Automation (EFTA).

[37] M. Albano, Luis Lino Ferreira, José Sousa, Extending publish/subscribe mechanisms to SOA applications. 2016 IEEE World Conference on Factory Communication Systems (WFCS), P. 1-4.

[38] MQTT, [Online]. Available: https://home-assistant.io/components/mqtt/ [accessed: August 2017]

[39] Z. Shelby, K. Hartke, C. Bormann, The constrained application protocol (coap) (2014).

[40] Z. Shelby, K. Hartke, C. Bormann, Message queuing telemetry transport (mqtt) version 3.1.1 (2014).

[41] Google Blockly Homepage : https://developers.google.com/blockly/

[42] Architecture for Multi-Criticality Agile Dependable Evolutionary Open System-of-Systems. Deliverable D2.3 AMADEOS conceptual model – revised. http://amadeos-project.eu/. [Accessed in July 2017].

[43] MySensors is an open source hardware and software community. https://www.mysensors.org/controller/domoticz [Accessed in July 2017].

This is a post-print of a paper published in Proceedings of the 2018 International Conference in Energy and Sustainability for Small Developing Economies (ES2DE 2018), Funchal, Portugal, 9-12 July 2018 and is subject to IEEE copyright.
https://doi.org/10.1109/ES2DE.2018.8494341
ISBN 978-1-5386-8092-6 ; e-ISBN 978-1-5386-8091-9.