IIoT Decentralized System Monitoring for Smart Industry Applications

Razvan-Adrian Luchian
Automation and Industrial Informatics
University "Politehnica" of Bucharest
Bucharest, Romania
razvan.luchian@upb.ro

Grigore Stamatescu
Automation and Industrial Informatics
University "Politehnica" of Bucharest
Bucharest, Romania
grigore.stamatescu@upb.ro

Iulia Stamatescu
Automation and Industrial Informatics
University "Politehnica" of Bucharest
Bucharest, Romania
iulia.stamatescu@upb.ro

Ioana Fagarasan
Automation and Industrial Informatics
University "Politehnica" of Bucharest
Bucharest, Romania
ioana.fagarasan@upb.ro

Dan Popescu
Automation and Industrial Informatics
University "Politehnica" of Bucharest
Bucharest, Romania
dan.popescu@upb.ro

***Abstract*—Convergence of operation technology (OT) and information technology (IT) in industrial automation is currently being adopted as an accelerating trend. The Industrial Internet of Things (IIoT) consists of heterogeneous sensing, computing and actuation nodes that are meshed through a layer of communication protocols, and represents a key enabler for this convergence. Experimental test beds are required to validate complex system designs in terms of scalability, latency, real-time operation and security. We use the open source Coaty – distributed industrial systems framework to present a smart industry application integrating field devices and controllers over the OPCUA and MQTT protocols. The experimental evaluation, using both proprietary automation components and open software modules, serves as a reference tool for building robust systems and provides practical insights for interoperability.**

***Index Terms*—industrial internet of things, operation technology, information technology, distributed systems, communication protocols**

I. Introduction

In modern automation applications, efficient monitoring of factory data is a fundamental requirement for increasing productivity and achieving economic and environmental efficiency. It is no longer enough to collect data using proprietary communication protocols and programmable logic controllers (PLCs) using closed-source software components. To this end, the Industry 4.0 concept brings as a novelty the convergence of operation technology (OT) with information technology (IT) specific design patterns, tools and libraries. It allows for open inter-connectivity among various system components as well as bridging previous gaps between the field, local control, supervisory control and optimization/management layers of the automation solution.

IT mainly deals with the retrieval, processing, security and transmission of electronic data using computers, storage equipment and network elements. Thus, any equipment that has these characteristics is part of IT. Information Technology is a subset of information and communication technology (ICT), which makes pervasive communications a key element of IT. OT is specific to the industrial environment and consists of the control and monitoring of the physical elements through the supervisory control and data acquisition (SCADA) system. Legacy equipment used such as sensors and actuators are usually hard-wired and not computerized, and those that are computerized transmit data using proprietary protocols. These proprietary protocols typically require a closed network [1]. Benefiting from recent standardization efforts in the field of Industry 4.0 networking, several options have become available to integrate both proprietary and open industrial system components through standards such as OPC UA [2].

The convergence of OT with IT is practically achieved through the Industrial Internet of Things (IIoT), as networks of sensing, control and communication devices which collaborate to achieve common goals while adhering to the constraints of the industrial environment. This concept consists of the integration of industrial equipment such as sensors and actuators with software elements to increase the reliability and efficiency of industrial processes. Today, IIoT is often used to gather, process and transmit data from the industrial environment. Traditionally, the flow of data from one piece of equipment to another is rigidly controlled and is based on a centralized architecture [3]. This potentially creates a single point of failure in the network which can be mitigated through a decentralized approach with multiple stakeholder agents.

We present our contribution with regard to an open, decentralized implementation of plant-level monitoring using new software tools where the main contributions are two-fold:

- An IIoT decentralized system architecture for system monitoring based on the Coaty [1] lightweight open-source framework for collaborative IoT;
- A practical evaluation of the architecture implementation in a dedicated case-study using modern commercial in-

[1] https://coaty.io

dustrial automation equipment and software-based agents.

The rest of the paper is structured as follows. Section II frames our contribution within the state-of-the-art with regard to IIoT-based decentralized system monitoring solutions. Section III presents the conceptualised system architecture for scalable automation using distributed agents and the enabling technologies. An experimental implementation using both commercial and open components is evaluated in Section IV. Section V concludes the paper with outlook on future work.

II. Related Work

In [4] the authors evaluate the performance of various IIoT protocols in a distributed control platform by focusing on transmission time and bounds for various parametrisation options. A standardized system configuration using embedded RaspberryPi development boards with open-source protocol implementations is used for performance measurements, including scalability performance with multiple data consumers in the automation network.

Leveraging the on-board computing resources of distributed embedded systems within the edge computing paradigm for industrial automation is argumented in [5]. A comparative analysis is presented for various communication protocols implemented on IIoT gateway platforms to integrate field-level production data with a remote cloud platform for data analysis, presentation and modelling.

In [6] an application for a low-cost historian module for SCADA systems in the water industry is described. The system is based on NodeRed technology and an OPC UA integration. The parametrisation of the OPC UA server in terms of security is highlighted as well as the functionality provided to the end-user through a dedicated graphical user interface (GUI).

A novel approach to dynamic reprogramming of IIoT application devices is introduced by [7] under the form of dispersed automation. This allows for allocation of on-channel computing tasks that are based on the specific context and role of each network node. A scalable and flexible network fabric is thus constructed which efficiently leverages all the resources available in the system.

Several applications of the Coaty solution for collaborative IIoT are presented in [8] and [9]. These showcase modules such as augumented reality (AR) and blockchain-based app marketplaces in an industrial context. Edge devices are classified according to their available resources and the applications and services that they can access. The blockchain operates as enabling technology for validation of application traceability.

In previous works we have argumented the integration on wireless sensor network with cloud systems as support backend infrastructure for complex monitoring [10]. In [11] the fog computing paradigm was employed to exploit on-node computing resources for efficient use of constrained radio communication channels. Consensus algorithms [12] can represent a feasible solution for distributed computing, in the absence of a centralised coordinator for industrial automation and control tasks. Localized models can be built based on the collected data for on-line inference in various process scenarios such as energy management [13].

III. Methodology

This paper proposes a system architecture for decentralized monitoring for smart industry applications. In order to leverage the recent developments in the field, this system is implemented using specific IIoT communication protocols and new programming instruments.

The main element of this architecture is the Coaty agent. An agent is a standalone application that is able to retrieve information from an industrial system and further communicate information to other peer agents in collaborative and ad-hoc manner. The connection between agents is loosely coupled and it does not need an entity to manage communication between the entities.

The basic architecture to enable field data collection from the sensors by the software agents is presented in Figure 1. The software agent includes both the OPC-UA server and client functionality and connects through MQTT publish-subscribe streams to the data source, in our case an industrial temperature transmitter, and to the user interface, implemented as a Node Red user front-end. Data is read from the software agent using OPC-UA methods. The parametrisation of the sensor can be realised directly from the user interfaced using a standardized JSON (JavaScript Object Notation) format file, easily accessible to compliant third parties. The JSON schema used includes the device type and identifier as well as the supported methods for requesting the data and adjusting device parameters such as data format, communication and additional services.

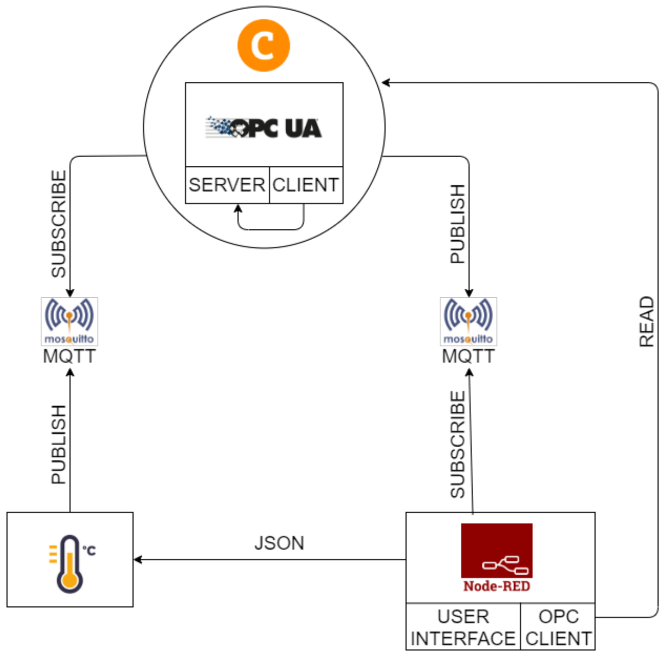


Fig. 1. IIoT System Architecture for Pervasive Monitoring and Control - Basic Architecture

The connectivity is handled through the Message Queue Telemetry Transport (MQTT) communication protocol [14], which is one of the main protocols used by the IIoT concept. MQTT is know as a many-to-many communication convention as it exchanges messages employing a central broker between numerous devices. MQTT devices associated to the broker with seemingly perpetual active TCP connection, this connection is initially overwhelming on obliged devices. MQTT does not back the message labeling with metadata or sorts to assist devices to get it. The protocol design contains three main components: a publisher, a broker and a subscriber. The devices that inquires about a particular point registers on it as a subscriber to be updated by the broker when the publishers distribute his topic. The publisher exchanges the data to the endorsers by means of the broker and operates as a generator of information, at that point, the authorization of the subscribers and the publishers are checked by the broker for security assurance. The reference structure of a MQTT data package is presented in Figure 2.

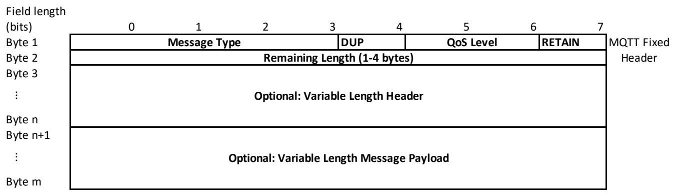


Fig. 2. MQTT Data Package

An important aspect in this architecture is the data integrity at the MQTT protocol level that can make the difference between a functional and a non-functional solution. Data security is ensured by Transport Layer Security (TLS). TLS is a protocol that runs on top of Transmission Control Protocol (TCP) and it is the successor to Secure Socket Layer (SSL). TLS enables safe communication between two machines from beginning to end. TLS employs cryptography to ensure data and end-point integrity, as well as data confidentiality. The handshake protocol helps the two parties to agree on how they will authenticate each other when creating a bond. During this step, the cipher spec protocol also allows the application data protocol to select the cryptographic primitives that will be used to ensure data authenticity and confidentiality during actual data transmission [15]. In our case, the temperature sensor is connected to an IO-Link master with IoT physical interface. Using this interface, the sensor is connected to the Mosquitto broker and publishes messages to a specific topic. The developed agent also connects to the same MQTT broker and subscribes to that topic.

Another communication protocol used in the IIoT concept is OPC Unified Architecture (OPC UA). OPC-UA has a multilateral use and it isn't only a client-server architecture. It is also developed for PLCs and control devices, not only for production management [16]. The introductory engineering of OPC UA was based on the client-server communication demonstrate where OPC UA clients can get to information from the OPC UA server address space through the request-response instrument. Be that as it may, concurring to the current OPC UA details (portion 14) discharged by the OPC Establishment, OPC UA moreover underpins the publisher-subscriber (pub-sub) mode of communication. Based on their part, OPC UA applications can be considered either publisher which sends information or subscriber which gets information. The pubsub component permits offbeat information communication and message-oriented middleware, which can be a program or a hardware-based framework, is utilized to decouple the OPC UA publishers and subscribers from each other. The OPC UA middleware bolsters both broker-based and broker-less messaging procedures [17]. The framework that we use for development includes an OPC UA connector. With its help, the agent connects to the OPC UA server and transmits the information received from the sensor. The architecture is based on the decentralized data transmission between several monitoring systems that have direct access only to certain sensors and can be scaled up as shown in Figure 3. In this situation, the architecture accommodates multiple distributed entities that can be sensors, controllers and actuation devices. Their variables and methods are discoverable within the network without a centralised repository. The user interface and physical device components are optional and we can deploy software agents that only process data and integrate with their peers without collecting it directly or displaying it to the user.

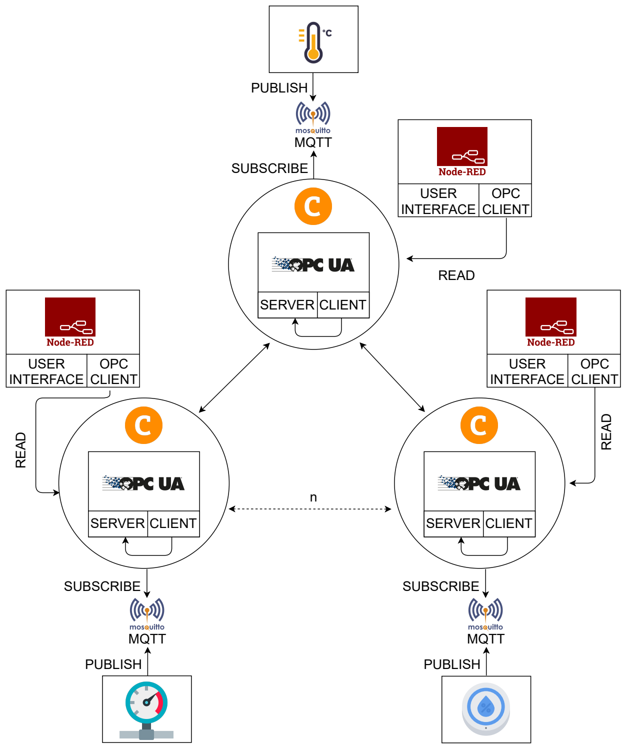


Fig. 3. IIoT System Architecture for Pervasive Monitoring and Control - Advanced Architecture with Scalability

These monitoring systems, depending on the needs, must also collect data from certain sensors to which they do not have direct access. The agent maps the values of the sensors stored in the OPC UA server to the Coaty Sensor Things API. Sensor Things objects are similar to OCG SensorThings API [18], but the entity schema is simplified. The API contains its own custom object types which describe sensor metadata, things where sensors are attached to, and observations, i.e. the concrete sensor data which can be published using either the Channel or the Advertise pattern. The developed agents multicast the Sensor Things objects created with sensors values from the OPC UA server to other agents from the same network. These objects are delivered through a channel with a specific channel identifier.

IV. Results

The implementation of the system is performed on two levels:

- node level
- network level

A. Node level

Each node consists of a Raspberry Pi model 3B+, an IoT communication module, IFM AL1302 IO-Link master with Profinet interface, and one or more sensors that use the IO-Link protocol, such as the IFM TD2501 temperature transmitter. The IO-Link master provides 16 digital inputs and 8 digital outputs alongside two Ethernet network interfaces for Modbus TCP and MQTT JSON communication and a dedicated IO-link bus. The temperature transmitter communicates using IO-link version 1.1 as COM2 with 38.4 kBaud or as standardized 4..20mA current loop. The experimental implementation is presented in Figure 4.

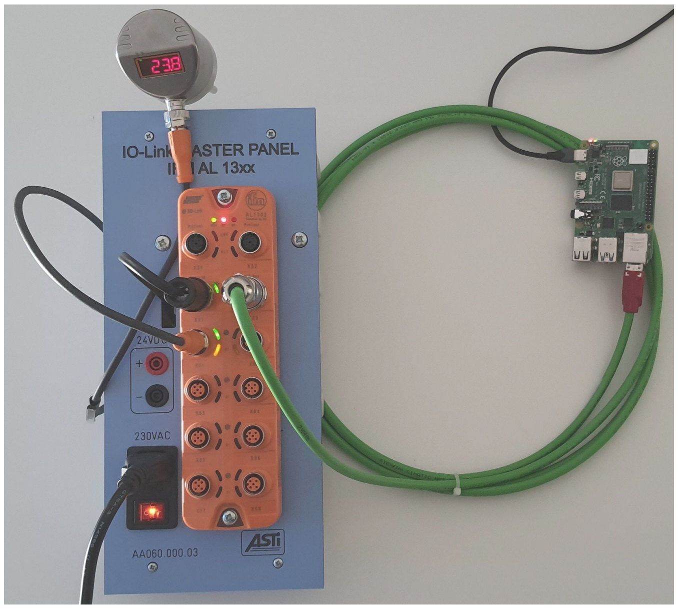


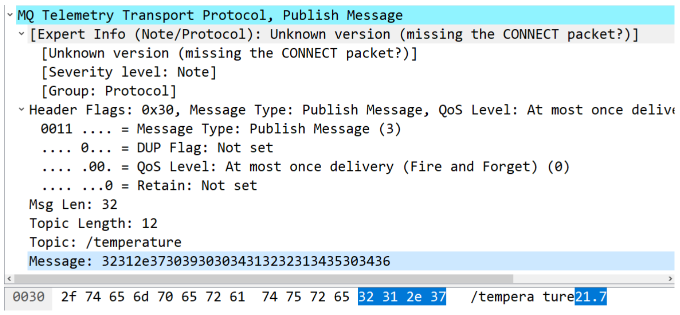
Fig. 4. Node configuration

For the interface with the sensors that use IO-Link, the communication module provides an IoT port with the specifications from Table 1.

TABLE I
IoT port specifications

Features	Description
Transmission standard	10Base-T; 100Base-TX
Transmission rate	10; 100
Protocol	MQTT JSON
Note on interfaces	Secure protocol; HTTPS

At the level of each node, the data from the assigned sensors is collected by MQTT protocol. The Coaty agent acts as a gateway to the communication module to which the sensor is connected, which communicates via the IoT port using the MQTT protocol. The MQTT message has a size of 32 bytes, of which the length of the topic is 12 bytes. Using the Wireshark network protocol analyzer [19] with the appropriate package parsing profile, we visualize this information, including the name of the topic, but also the value transmitted by the sensor. Figure 5 presents a screenshot from the Wireshark GUI as a snasphot of the implemented industrial communication.



```
˅ MQ Telemetry Transport Protocol, Publish Message
  ˅ [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet?)]
      [Unknown version (missing the CONNECT packet?)]
      [Severity level: Note]
      [Group: Protocol]
  ˅ Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delive
      0011 .... = Message Type: Publish Message (3)
      .... 0... = DUP Flag: Not set
      .... .00. = QoS Level: At most once delivery (Fire and Forget) (0)
      .... ...0 = Retain: Not set
    Msg Len: 32
    Topic Length: 12
    Topic: /temperature
    Message: 32312e373039303034313232313435303436

0030  2f 74 65 6d 70 65 72 61  74 75 72 65 32 31 2e 37   /tempera ture21.7
```

Fig. 5. MQTT message

In order for the software agent to receive the sensor information, a raw pattern event is used. The ”observeRaw” method, which receives the topic as a parameter, is used to observe received messages that match the specified topic. The result of this method is a message consisting of 2 elements, the publication topic and the payload. The payload is a *Uint8Array* object that is decoded to obtain the value of the sensor.

The agent also acts as a gateway to the NodeRed programming tool for wiring together hardware devices. Thus, it provides the data received from the sensors in two ways, through an MQTT broker and through an OPC UA server. The information is published using the ”publishRaw” method which receives as parameters the payload and the topic to which NodeRed will later subscribe.

On each device where the agent is located, an OPC UA server developed in Node.js is also open. Using a connector service for OPC UA, the Coaty agent has the OPC UA client functionality. It connects to the OPC UA server using its address as a parameter (EndPointUrl). The OPC UA client has a specified name, but also the lifetime of the connection to the server. The data sources are mentioned directly in the options of the OPC UA client. Table 2 shows the options of the OPC UA client. The sensor information collected by MQTT, using the created OPC UA client, is stored in the OPC UA server using the "writeDataValue" method which has as parameters the data source and the data value.

TABLE II
CONFIGURATION FOR OPC UA CLIENT

Options	Description
EndPointUrl	OPC UA server address
ConnectionOptions	Options such lifetime or client name
DataSources	These are defined the OPC UA data sources to be monitored

In our situation, the agent is a data monitoring point. Therefore, a graphical interface is needed to display the collected data. The solution chosen is to use the NodeRed tool [20] as in Figure 6, due to the ease of connection to the Coaty agent, the graphical benefits and also due to the fact that at the base of this tool and the Coaty framework is Node.js. Retrieving information from the Coaty agent is carried out in two ways: using MQTT or using OPC UA.

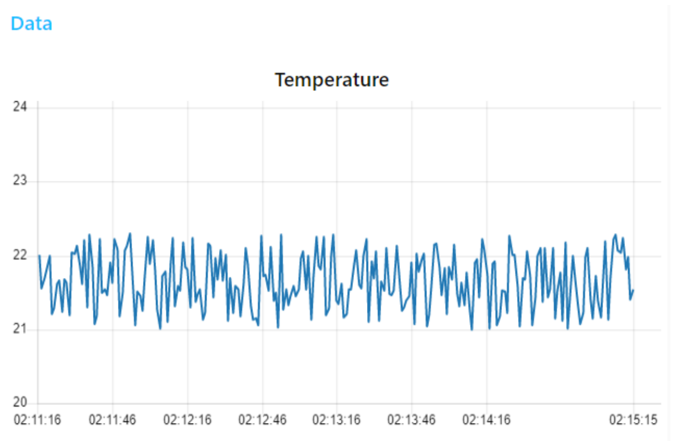


Fig. 6. GUI in NodeRed for Data Visualization

A sample of the monitored network traffic is provided in Figure 7. This illustrates both MQTT and OPC UA traffic, with a higher transmission rate (once every five seconds) given the temperature process variable.

In order for the information to be passed on to other Coaty agents, it is mapped in the form of Coaty objects, more precisely in the form of Sensor objects that observe and measure one property and have a predefined structure that can be configured:

```
{
    name: string,
    objectType: SensorThingsTypes.OBJECT_TYPE_SENSOR
        ,
    coreType: "CoatyObject",
    objectId: Uuid,
    parentObjectId: Uuid,
    description: string,
    unitOfMeasurement: UnitOfMeasurement,
    observationType: ObservationType,
    observedArea?: Polygon,
    phenomenonTime?: TimeInterval,
    resultTime?: TimeInterval,
    observedProperty: ObservedProperty,
    encodingType: SensorEncodingType,
    metadata: any
}
```

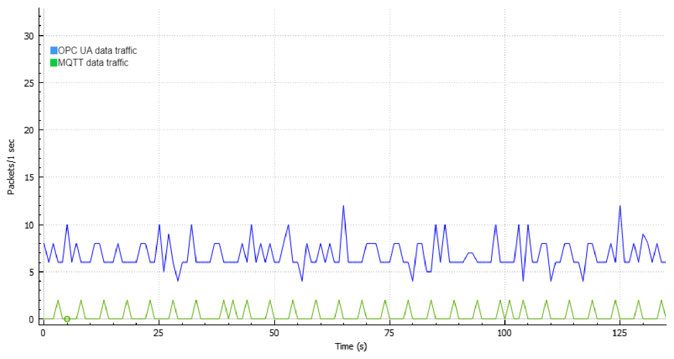


Fig. 7. Captured network traffic

These Sensor objects are registered to a "Thing" type object, which is actually a Raspberry Pi where the Coaty agent is located. The "Thing" object is also a Coaty object that can be parameterized according to a predefined structure:

```
{
    name: string,
    objectType: SensorThingsTypes.OBJECT_TYPE_THING,
    coreType: "CoatyObject",
    objectId: Uuid,
    description: string,
    properties?: { [key: string]: any; },
    locationId?: Uuid
}
```

B. Network level

In the network, sensor objects are transmitted using an event-based communication pattern. In the event-based paradigm, messages are generated aperiodically given changes in the process variables or specific consumer requests. The Channel pattern is used, which offers an efficient method of pushing Coaty objects to the observers in the network that use a specific channel identifier. The software entities use multicast transmissions to distribute the objects to interested parties. The associated channel event data structure uses a JSON format that includes both the single or multiple object references and the privateData field. As optional component, the privateData field can be used to distribute application-specific options through key-value pairs.

For this application, we considered the channel identifier to be the very MQTT topic used for data collection. Thus, each observer of each agent observes the objects that interests him specifying the channel identifier:

- Temperature Sensor: "/temperature"
- Pressure Sensor: "/pressure"
- Humidity Sensor: "/humidity"

Within the network, the software agent does not know about the existence of the other agents, it only provides the information in the form of objects and observes other messages from other agents depending on the channel identifiers used. Given this fact, the expansion of the network can be done easily, without intervening in its configuration, thereby achieving scalability and eliminating single points of failure for increased robustness. For larger networks a clustered implementation can be envisioned to assemble agents into functional groups that serve persistent or opportunistic application goals depeding on the design objectives.

V. Conclusion

We presented an application of smart industry decentralized monitoring integrating industrial internet of thing technologies. The main advantage of this approach is that it provides an uniform abstraction layers based on state-of-the-art open industrial technologies that can integrate control equipment, sensors and actuators across manufacturers. The system can be adapted and configured for each particular application domain or industry with competing priorities: latency, throughput, resilience and security. The main research scope has been to validate the interplay of various conceptual, open-source and standardized technological components into a unitary approach. The results are promising towards enabling easier Industry 4.0 application development using reusable plug-and-play function blocks in increasing levels of complexity e.g. from reading a single sensor values to distributed queries across networks of interconnected agents. Both wired and wireless network connections are supported for heterogeneous integration given the increasing penetration of wireless devices in industrial networks that significantly reduce installation and maintenance costs over large lifetime of automation systems and equipment.

The current goal is to build out a test-bed for industrial internet of things for extensive benchmarking and implementation in closed-loop control for process control (level, flow, pressure) and smart manufacturing (position, speed). By including the controller in the network e.g. as industrial PC or specialized PLC-type device we can thoroughly evaluate the end-to-end performance of the architecture and implement and adaptive parametrisation scheme that considers the network and field level status for achieving the control objectives. This can be helpful for validating theoretical bounds of control performance derived from the analytical design based on the system models for both continuous, hybrid and discrete processes. As previously closed industrial technologies open up, the need for reliable security mechanisms is also acknowledged and will be further considered in future work based on a self-evaluation of process task criticality.

Acknowledgment

The support of the company Asti Automation for providing the equipment used in the practical experiments from this paper is gratefully acknowledged. This work was partially supported by the Romanian Ministry of Education and Research under grant 78PCCDI/2018-CIDSACTEH.

References

[1] P. K. Garimella, "It-ot integration challenges in utilities," in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, 2018, pp. 199–204.

[2] P. Drahoš, E. Kučera, O. Haffner, and I. Klimo, "Trends in industrial communication and opc ua," in *2018 Cybernetics Informatics (K I)*, 2018, pp. 1–5.

[3] T. Lewandowski, D. Henze, M. Sauer, J. Nickles, and B. Bruegge, "A software architecture to enable self-organizing, collaborative iot ressource networks," in *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, 2020, pp. 70–77.

[4] D. L. Tran, T. Yu, and M. Riedl, "Integration of iiot communication protocols in distributed control applications," in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, 2020, pp. 2201–2206.

[5] S. Raileanu, T. Borangiu, O. Morariu, and I. Iacob, "Edge computing in industrial iot framework for cloud-based manufacturing control," in *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)*, 2018, pp. 261–266.

[6] A. Nicolae and A. Korodi, "Node-red and opc ua based lightweight and low-cost historian with application in the water industry," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, 2018, pp. 1012–1017.

[7] G. Quirós, D. Cao, and A. Canedo, "Dispersed automation for industrial internet of things," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1176–1181, 2020.

[8] A. Seitz, D. Henze, J. Nickles, M. Sauer, and B. Bruegge, "Augmenting the industrial internet of things with emojis," in *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, 2018, pp. 240–245.

[9] A. Seitz, D. Henze, D. Miehle, B. Bruegge, J. Nickles, and M. Sauer, "Fog computing as enabler for blockchain-based iiot app marketplaces - a case study," in *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, 2018, pp. 182–188.

[10] O. Chenaru, G. Stamatescu, I. Stamatescu, and D. Popescu, "Towards cloud integration for industrial wireless sensor network systems," in *2015 9th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, 2015, pp. 917–922.

[11] V. Mihai, C. Dragana, G. Stamatescu, D. Popescu, and L. Ichim, "Wireless sensor network architecture based on fog computing," in *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2018, pp. 743–747.

[12] G. Stamatescu, I. Stamatescu, and D. Popescu, "Consensus-based data aggregation for wireless sensor networks," *Journal of Control Engineering and Applied Informatics*, vol. 19, no. 2, pp. 43–50, 2017.

[13] G. Stamatescu, R. Entezari, K. Römer, and O. Saukh, "Deep and efficient impact models for edge characterization and control of energy events," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, 2019, pp. 639–646.

[14] M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh, and R. Al-Hatmi, "Internet of things: Survey and open issues of mqtt protocol," in *2017 International Conference on Engineering MIS (ICEMIS)*, 2017, pp. 1–6.

[15] C. Lesjak, D. Hein, M. Hofmann, M. Maritsch, A. Aldrian, P. Priller, T. Ebner, T. Ruprechter, and G. Pregartner, "Secure smart maintenance services: Hardware-security and tls over mqtt," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, 2015, pp. 1243–1250.

[16] M. H. Schwarz and M. Börcsök, "A survey on opc and opc-ua," in *2013 XXIV International Conference on Information, Communication and Automation Technologies (ICAT)*, 2013, pp. 1–6.

[17] S. K. Panda, M. Majumder, L. Wisniewski, and J. Jasperneite, "Real-time industrial communication by using opc ua field level communication," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020, pp. 1143–1146.

[18] A. Kotsev, K. Schleidt, S. Liang, H. Van der Schaaf, T. Khalafbeigi, S. Grellet, M. Lutz, S. Jirka, and M. Beaufils, "Extending inspire to the internet of things through sensorthings api," *Geosciences*, vol. 8, no. 6, p. 221, 2018.

[19] C. Sanders, *Practical packet analysis: Using Wireshark to solve real-world network problems*. No Starch Press, 2017.

[20] M. Lekić and G. Gardašević, "Iot sensor integration to node-red platform," in *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*. IEEE, 2018, pp. 1–5.