Multi-Protocol Interoperability Between Distributed Cyber-Physical Systems Towards Industry 4.0 Collaborative Optimization

Md Sabbir Bin Azad
Dept. of Mathematics and Industrial Engineering
Polytechnique Montreal
Montreal, Canada
e-mail: sabbirbinazad@gmail.com

Dr. Christophe Danjou
Dept. of Mathematics and Industrial Engineering
Polytechnique Montreal
Montreal, Canada
e-mail: christophe.danjou@polymtl.ca

Abstract— **Industry 4.0 has emerged as a potential strategy to providing extensive connectivity in the production environment, which is rapidly evolving combined with rising commercial demand, mass personalized manufacturing. Mass customization and complicated products necessitate more data and more adaptable Machine-to-Machine (M2M) communication that facilitates smooth data interchange and interaction between industrial components in smart manufacturing. Integrating industrial Internet of Things (IoT) devices to benefit different industry sectors simultaneously requires extensive network connectivity, interoperable communication, and collaboration among the networked machines. While critical technical issues with network connectivity have been properly addressed, the technology is not ready for flexible and seamless communication between disparate machines. One of the challenges that arises because of this development is the growing need for interoperable standards and protocols at various levels of the manufacturing ecosystem. Considering the interoperable infrastructure required for Industry 4.0, the paper provides a secure and cost-effective interoperable solution for multi-protocol translators. The key contribution of the paper is a method for mapping IoT multi-protocols into a low-cost gateway, as well as providing effective full-duplex interoperable M2M communication and cloud integration for compatible platforms.**

Keywords- IioT; Gateway; Interoperability; Protocols; M2M; Raspberry Pi.

I. INTRODUCTION

Industry 4.0, fourth industrial revolution brought about by introduction of Internet of Things(IoT) and Cyber Physical Systems (CPSs) [1], has emerged as a promising approach to provide extensive connectivity in manufacturing environment [2]. The development of smart manufacturing technologies is fast changing, and when it is coupled with increasing commercial demand, additive manufacturing shows numerous advantages in providing customized and specifically designed products [3]. Mass customized and complex products leads to a greater needs of information and more flexible automation solutions [4]. This flexibility and more advanced information handling requires more intelligence in the system [5]. One of the resulting challenges of this flexible information management is the increased need for interoperability at different levels of the manufacturing ecosystem [6]. Successful system integration requires good strategies to manage middleware connectivity.

However, integrating new devices to benefit different industry sectors simultaneously requires significant challenges as part of what is being called the Industrial Internet of Things (IIoT) [7]. IIoT devices have unique features, such as low processing and memory, low bandwidth for data download and upload, and limited battery life [8]. Given the ubiquity of these devices and facing such limitations, it is necessary to develop new types of communication protocols designed to deal with these limitations.

Generally, the used protocols are based on communication through cloud and between machines [9]. Semantic interoperability should be achieved in an interworking solution in order to provide a common meaning of the data exchanged by heterogeneous devices, even if they belong to different domains [10]. Different communication protocols are employed in IoT, e.g., Hyper Text Transfer Protocol (HTTP), Constrained Application Protocol (CoAP), MQ Telemetry Transport (MQTT), Modbus Transmission Control Protocol and Internet Protocol (TCP/IP), Web Sockets, Advanced Message Queuing Protocol (AMQP). The main driving force for the design of such protocols is the hardware limitations of embedded devices, which impede the use of traditional network protocols. The integration of communication protocols would cause interoperability among several devices and services, and a possible solution is the conception of a multiprotocol strategy [11]. Though existing standards, e.g., MTConnect, OPC Unified Architecture (OPC-UA) and AutomationML, allow for specifications of industrial objects and information-rich machine-to-machine (M2M) communications, the information models generated from these standards are not semantically defined, making the semantic understanding and intelligent decision-making a challenge [12]. In this regard, an IoT system interchanging between access protocols may overcome the said challenges in interconnected heterogenous systems.

The primary contributions of this research can be summarized as follows:

- Design protocol selection framework among HTTP, MQTT, CoAP, WebSocket Modbus TCP protocol based on the capabilities and requirements of the device and sensor data transmission.

- Design and Implement model of interoperable interpreter for effective M2M information exchange and action plan at a faster pace towards sustainable Industry 4.0 manufacturing

The remainder of the paper is organized as follows: in Section II, related work and interoperability standards are discussed; the strategy proposed, and its characteristics are defined in Section III; Section IV focuses on development model and strategy of proposed architecture; validation of the proposed system through case study is demonstrated in Section V; finally, in Section VI, we discussed conclusions and future work.

II. Related Work

The popularity of the internet has led to the emergence of internet protocol-based communication standards, creating a unified infrastructure for the integration of disparate systems and devices. There are several solutions proposed or implemented with the aim of increasing IoT interoperability.

H. Derhamy et al. [13] developed a multi-protocol solution for IoT interoperability issue. The solution includes protocol implementation translators based on Service Oriented Architecture (SOA), intermediate format to reduce the number of translations necessary. The system also detects protocol incompatibilities and performs the translation.

Barros et al. [14] introduced Internet of Things Multiprotocol Message Broker (IoTM2B) strategy to integrate various communication protocols such as HTTP, MQTT and CoAP and their performance evaluation based on two scenarios, machine-to-machine (M2M) protocols Communication and cloud-based environment. This strategy extends IoT DSM to provide integration with embedded devices Via various protocols.

Derhamy et al. [15] proposed interoperability solution consists of a multi-protocol translator that is injected into the service exchange on demand. The main contribution of this research is to suggest ways to map OPC UAs to intermediate formats. Intermediate formats can be mapped to other standard IoT protocols such as CoAP, HTTP, and MQTT.

The main issue with current interoperable solutions is that there isn't a method that fits well with integration of IoT multi protocols in a gateway and effective full duplex interoperable communication to interconnect the sensors, IIoT devices, and machines and cloud integration for compatible platforms.

III. Proposed Solution

To address this integration and interoperability challenges, we proposed a platform to develop an interoperable system which can connect heterogenous devices with different protocol, process and exchange the data with different machines and cloud platform. Fig. 1 illustrates the architecture of the proposed research framework. We defined the architecture of the gateway with four modules which are the key functions of the embedded communication.

A. Protocol Selection Framework

To facilitate safe and high-speed data transfer among end IoT devices, users set protocols among MQTT, CoAP, HTTP, WebSocket and Modbus TCP for the nodes and/or sensors to send data to the gateway. Each of these protocols have distinct features and capabilities, which add complexity to the identification of a protocol suitable for specific use cases.

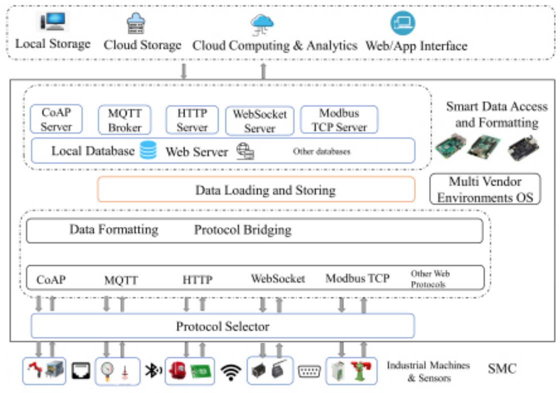


Figure 1. Architecture of the proposed research framework

For instance, MQTT is recommended in network scenarios where bandwidth consumption must be reduced and the devices involved in communication have low processing and memory capacity [16]. WebSocket protocol's standard connectivity helps simplify many of the complexities and difficulties involved in the operation of bi-direction communication.

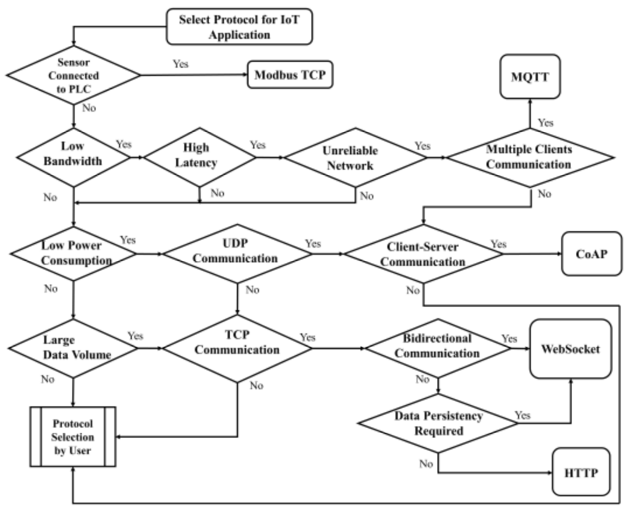


Figure 2. Protocol Selector for IoT Application

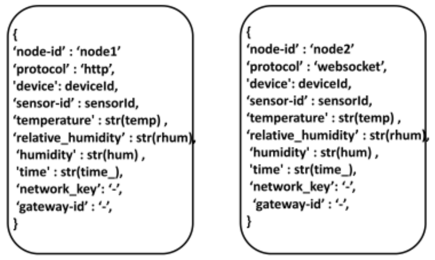
However, a difficulty is that the sensors in industries and shop floors are mainly connected to Programmable Logic Controllers (PLCs) to collect large amount of sensor data and to transfer it to a communication system [17]. To enable IoT connectivity for these sensors connected to PLCs, the controller needs to configure with Modbus TCP protocol. Fig. 2 demonstrates protocol selection framework based on the capabilities and requirements of the sensor data transmission to gateway. In addition to the protocol selection for the nodes, it is necessary for the processor unit to have the ability to run and execute a Web Server, on which services will be deployed.

B. *Connectivity with Heterogenous Nodes*

The First module of the IoT gateway system is to receive data from heterogenous nodes and/or sensor with multi-protocol communication. The gateway uses wireless communication protocol (e.g., Zigbee, Bluetooth, Wi-Fi) and Local Area Network (LAN) to acquire the packet from the heterogenous sensor nodes, and use the 3G/4G and, other network interfaces to read and parse the data and then send it to data format module for standardizing.

C. *Data Formatting*

Since heterogeneous nodes and sensors send data over different protocols, they send data with different data format. The gateway uses JavaScript Object Notation (JSON) format to systematize the representation of data coming from the nodes and sensors. An example of the representation of the data recorded by the temperature and humidity sensor, in JSON format, is shown in Fig. 3.



```
{
'node-id' : 'node1'
'protocol' : 'http',
'device': deviceId,
'sensor-id' : sensorId,
'temperature' : str(temp) ,
'relative_humidity' : str(rhum),
 'humidity' : str(hum) ,
 'time' : str(time_),
 'network_key': '-',
 'gateway-id' : '-',
}
```

```
{
'node-id' : 'node2'
'protocol' : 'websocket',
'device': deviceId,
'sensor-id' : sensorId,
'temperature' : str(temp) ,
'relative_humidity' : str(rhum),
 'humidity' : str(hum) ,
 'time' : str(time_),
 'network_key': '-',
  'gateway-id' : '-',
}
```

Figure 3. Example of systematized data format

This format has important advantages such as simplicity and low resource consumption [16].

D. *Protocol Bridging*

The proposed IoT gateway acts as a bridge between different protocols, mainly among HTTP, MQTT, CoAP, WebSocket and Modbus TCP. The gateway is continuously being ready for listening for these multiprotocol connection requests and message payload with the standardized format. An example is given in Fig. 4 to illustrate the interoperable protocol communication among the nodes and gateways in the proposed architecture.

The broker Mosquitto is a message broker that implements several versions of the MQTT protocol and it is relatively a lightweight software [18] and it has low power profile. The advantages of data transfer MQTT is good, reliable, easy to build and it uses less network resources even in conditions of unstable network [19]. In the gateway, WebSocket communication technology also is adopted in MQTT broker as WebSocket provides full-duplex communication channels over a single TCP/IP connection. Another most familiar protocol is HTTP which user can set for communication between nodes and gateway server. The data sent to the server with POST is stored in the request body of the HTTP request. Another high interoperability protocol for embedded devices with an increased level of security is CoAP.

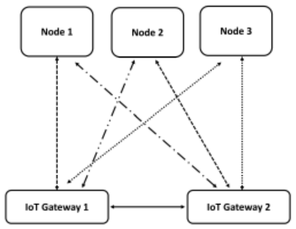


Figure 4. Interoperable communication among the nodes and gateways

The node which acts as CoAP Client could send data to the server on a particular port 5683 with the help of browser add-on Copper (Cu) CoAP user-agent. A relevant characteristic of Modbus TCP is that it is supported by both proprietary and open-source hardware/software, so different equipment can seamlessly share information enabled by this protocol [20].

IV. RESEARCH DEVELOPMENT

Key components and flows are described at a high level below.

A. *Proposed System Architecture*

The solution proposed in this research aims at enabling interoperable connectivity from heterogenous devices and data acquired from different communication protocols and extending these networks towards the IoT universe. To this purpose, we implemented 5 modules within the application: (i)Multi-protocol gateway development (ii)Multi-protocol server integration (iii)Node MCUs and Sensors Integration (iv)Nodes and gateway interoperable communication (v)Data storage to multiple databases. Fig. 5 illustrates the implementation phases of the architecture.

B. *Multi-Protocol Gateway Development*

For the development of the gateway, Raspberry Pi 4B was used which is illustrated in Fig. 5(f). At first, multiple single board computers were compared to find the cost efficient and complex task management compatible gateway. Table 1 is provided to show the comparison among the development boards.

TABLE I. COMPARISON OF SINGLE BOARDS

	Raspberry Pi 4	**Raspberry Pi 3**	**Beagle Bone**	**Intel Galileo**
Processor	Quad core 64-bit ARM-Cortex A72	Broadcom BCM2837 Quad Core	ARM Cortex-A8	Quark SoC X1000, 32-bit Intel
Frequency& RAM	1.5GHz, 4GB	1.2GHz, 1GB	1GHz, 512MB	400MH, 512 KB SRAM 256Mb DRAM
Operating System	Raspbian, Debian, Fedora, ARCH Linux ARM, Ubuntu etc.	Raspbian, Debian, Fedora, ARCH Linux, ARM etc.	Android, Debian, Angstrom, Yocto, Fedora, Ubuntu etc.	Arduino, Linux distribution for Galileo, Rocket etc.
Power	15.3W	10W	15W	15W
Cost	70 CAD	45 CAD	70 CAD	100 CAD

Raspberry Pi is proposed as gateway because of its lower cost, high processing capability, random-access memory (RAM), 40 general purpose input/output (GPIO) pins, RJ45 port and Wi-Fi connectivity. The Pi 4 is continuously being ready for listening for these multiprotocol connection requests and message payload with the standardized format through 802.11 b/g/n/ac Wireless LAN network.

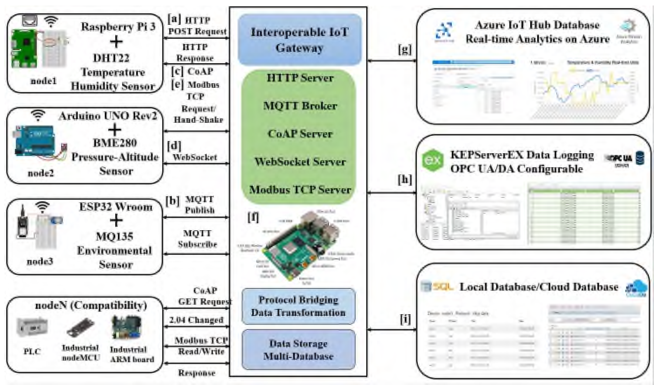
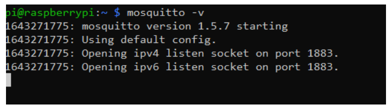


Figure 5. Proposed Architecture Implementation Framework, [a]HTTP, [b]MQTT, [c]CoAP, [d]WebSocket, [e]ModbusTCP, [f]Raspberry Pi 4 as Gateway, [g]Azure IoT Hub Database, [h]KepwareServer Data Logging, [i]Local/Cloud Database

C. *Multiple Server Configuration on the Gateway*

MQTT, HTTP, CoAP, Modbus TCP and WebSocket server are integrated in the gateway. Raspbian OS has been installed in the system to configure all the servers and install the required software and libraries.

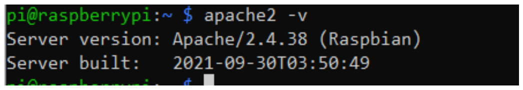
1) MQTT Broker Configuration: The software that is being used here is Mosquitto. After installation series of packages, the gateway broker was ready to publish or subscribe the topic. Fig. 6 illustrates the start-up Mosquitto MQTT broker on the terminal.



```
pi@raspberrypi:~ $ mosquitto -v
1643271775: mosquitto version 1.5.7 starting
1643271775: Using default config.
1643271775: Opening ipv4 listen socket on port 1883.
1643271775: Opening ipv6 listen socket on port 1883.
```

Figure 6. Mosquitto MQTT Broker Running on Terminal

2) HTTP Server: The Apache web server (HTTP server) is installed on the Raspberry Pi 4 IoT gateway. Apache can communicate between nodes and the server over the HTTP. The Apache profile opens the port 80. Fig. 7 demonstrates apache2 Raspbian version running on the gateway.



```
pi@raspberrypi:~ $ apache2 -v
Server version: Apache/2.4.38 (Raspbian)
Server built:   2021-09-30T03:50:49
```

Figure 7. Apache3 Raspbian Version Webserver Installed

3) CoAP Server Implementation: In this work, the aiocoap Python CoAP library was used to implement the CoAP protocol [21]. Two access methods are initiated here one is PUT which allows the node connected with the sensors transfer data to the gateway and, the other is GET which allows the nodes with actuators to register with the resource so that nodes can be notified when gateway initiates any feedback command and transfer data.

4) Modbus TCP Server: For Modbus TCP/IP communication, an open-source and full modbus protocol called "pyModbus" is used. It works as fully implemented modbus server and supports read/write on discrete and register. 30 registers are written initially for receiving read/write requests from the clients. More registers can be added for further development.

5) WebSocket Server Deployment: As WebSocket enables bidirectional communication in real time over the web, WebSocket server is deployed in the gateway for horizontal interoperable communication. After enabling Node.js html server, http does the handling requests and serving content and Uniform Resource Identifier (URL) helps to parse requested URLs.

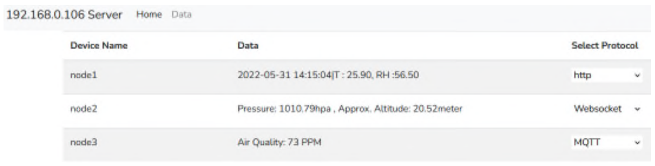
D. *Communication between Node MCUs and Sensors*

For the development of the proposed design, sensors are connected through the GPIO pins and input/output (I/O) interface of the node MCUs. Three sensors DHT22 sensor, BME280 sensor, MQ-135 are connected to three node MCUs Raspberry Pi 3, Arduino Uno Rev2 and ESP32. These sensors collect the data from surroundings like temperature, humidity, pressure, altitude, and air quality.

E. *Communication between Node MCUs and Gateway*

In this development, node2 Arduino Uno with bme280 pressure sensor send data over WebSocket Protocol. Node3 ESP32 send data over MQTT protocol. Node1 is configured to communicate with HTTP, CoAP and Modbus TCP protocol.

In Fig. 5(d) node2 creates client socket and tries to establish a communication link to the gateway server using its IP address and port number 80. When the communication established between node2 and gateway, it received pressure and altitude from node2 Arduino Uno Wi-Fi and showed the data to the interface. Here, the gateway is also devising as MQTT broker which facilitates the communication from node3 transferring messages from publisher to subscriber and subscriber to publisher in Fig. 5[b]. Node3 ESP32 with environmental sensor provides air quality data in ppm unit. In regards, reading data from sensor and sending data to the gateway broker, PubSubClient MQTT library is used on the node3 end. Node1 will use the Adafruit DHT library to retrieve the DHT22 sensor's current temperature and humidity readings. After the gateway is configured to receive data from node1 over CoAP protocol shown in Fig. 5(c), CoAPthon Python library the script is activated to create a CoAP endpoint. Fig. 8 shows sensor data receiving in the gateway from node1, node2 and node3.


192.168.0.106 Server Home Data

Device Name	Data	Select Protocol
node1	2022-05-31 14:15:04|T : 25.90, RH :56.50	http
node2	Pressure: 1010.79hpa , Approx. Altitude: 20.52meter	Websocket
node3	Air Quality: 73 PPM	MQTT

Figure 8. Node1, node2, node3 data collected from gateway

To demonstrate Modbus TCP communication between the client and gateway, in Fig. 5 (e) node1 was configured as Modbus client that transfers temperature and humidity data to gateway which works as Modbus TCP server. As the gateway had register map for all data types with desired size, these registers received read/write requests from the node. For the demonstration of http protocol communication, node1 was also configured as HTTP client.

F. *Data Collection and Storage*

Three databases are presented to visualize and process for server and device communications. We divided the databases into (i) local and cloud database (ii) KEPServerEX data logging and communication and (iii) Azure IoT hub databases.

After processing data received from the nodes, the gateway sends data to MariaDB which is shown in Fig. 5(i). In the experiment, the gateway receives data from the different nodes and edge devices and provides it to KEPServerEX OPC server MQTT client by Kepware. From the Fig. 5(h), we demonstrated Air Quality data with the corresponding timestamp that received from the MQTT client through the KepServerex and stored into Microsoft Access Databases. Azure IoT hub environment is deployed to connect the gateway with nodes to the cloud. After obtaining primary connection string, gateway establishes communication with the IoT hub and sends data over MQTT protocol to the hub. In Fig. 5(g) data ingestion is shown in Azure Data Explorer Database.

V. VALIDATION

This section contains a case study to analyze a workflow involving data collection from nodes with sensors, gateway integration to cloud with the goal of validating the interoperability concept of integrating any platforms with the developed multi-protocol gateway.

A. *Case Study: Implementation on ThingsBoard*

The case study refers to implementation communication between gateway and ThingsBoard IoT platform. ThingsBoard is an open-source IoT platform built on the Java 8 platform that functions as an IoT gateway between registered devices communicating via HTTP, CoAP, and MQTT protocols to collect, analyse, visualize, and manage data [22].

1) Configuration: ThingsBoard is configured to monitor and visualize data by creating IoT Dashboards and updating in real-time. The multi protocol gateway receives sensor data from three different nodes with different protocols Modbus TCP, WebSocket and MQTT and, process the data as JavaScript Object Notation (JSON) format before sending to ThingsBoard Platform. Fig. 9 illustrates the integration details between the gateway and the ThingsBoard platform.

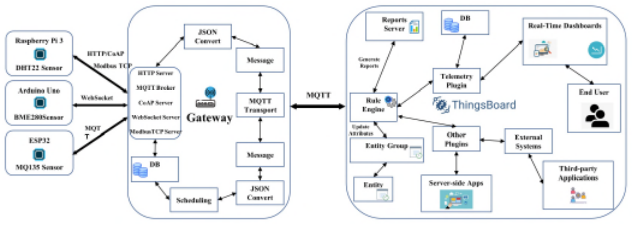


Figure 9: ThingsBoard platform integration with muti-protocol gateway

The gateway uses an access token to access the web interface of the ThingsBoard cloud server. Telemetry sent by the gateway for logging in the platform is inserted into the SQLite database table before being transferred to the ThingsBoard. To publish telemetry data to ThingsBoard, gateway publish message to the following topic: “v1/devices/me/telemetry”. Finally, the telemetry data get uploaded to the ThingsBoard using the MQTT publishing feature.

2) Real-time Visualization on ThingsBoard: ThingsBoard's integration APIs allow custom applications to be built, and they use their own data visualisation tools.

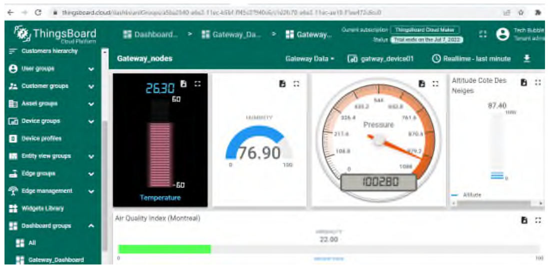


Figure 10: Real-time data visualization dashboard on ThingsBoard

Devices configured as Gateway_device01 in the ThingsBoard received telemetry data from the multi-protocol gateway. As seen in Fig. 10, gateway is sending all the sensor values received by ThingsBoard platform with timestamp.

B. Results

Successful integration of the multi protocol gateway to the open source IoT ThingsBoard platform validates interoperable data exchanges compatibility of the proposed gateway. The gateway receives data from three different embedded devices over three different protocols such as Modbus TCP, WebSocket and MQTT. The gateway receives the signal, processes, and transfers the sensors values real time into ThingsBoard IoT platform over MQTT protocol. To address the IoT Interoperability challenges, the gateway can provide the strategy of using different protocol integration, and data conversion, as well as integrating real-time data analysis with Kafka and Spark platforms on ThingsBoard platform for big data analytic applications. Thus, Integrating different protocol-enabled devices into each system enables the scalability, automation, and flexibility of Industry 4.0 manufacturing systems.

VI. Conclusion and Future Work

For interoperable M2M communication between heterogenous devices, we proposed and developed an interoperable middleware gateway by enabling multi protocol integration. In Section V, we evaluated the potential gain of deploying the middleware gateway in a case study to provide a real-time cloud-based integration and visualization that facilitates sensor-based data collection, gateway integration and interoperability management, and to validate the interoperability concept of integrating any platforms via any of the industry standard protocols MQTT, CoAP, HTTP, Modbus TCP and WebSockeT. Future development can be expanding the supported OPC UA service and AMQP, CAN, CC-Link protocols implementation and PLC and other controllers bridging in the gateway. Future experimental tests will evaluate the performance of data analytics systems in terms of responsiveness, flexibility, and scalability in large, real-world scenarios.

References

[1] H. Kagermann, J. Helbig, A. Hellinger, and W. Wahlster, Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry;report of Industrie 4.0 Working G. Forschungsunion, 2013.

[2] H. S. Li, L. F. Lai, and H. V. Poor, "Multicast Routing for Decentralized Control of Cyber Physical Systems with an Application in Smart Grid," IEEE J. Sel. Areas Commun., Article vol. 30, no. 6, pp. 1097-1107, 2012, doi: 10.1109/jsac.2012.120708.

[3] Y. Meng, Y. Yang, H. Chung, P.-H. Lee, and C. Shao, "Enhancing Sustainability and Energy Efficiency in Smart Factories: A Review," Sustainability, vol. 10, no. 12, 2018, doi: 10.3390/su10124779.

[4] H. A. ElMaraghy, "Flexible and reconfigurable manufacturing systems paradigms," Int. J. Flexible Manuf. Syst., Article; Proceedings Paper vol. 17, no. 4, pp. 261-276, 2005, doi: 10.1007/s10696-006-9028-7.

[5] D. Zuehlke, "SmartFactory – from Vision to Reality in Factory Technologies," IFAC Proceedings Volumes, vol. 41, no. 2, pp. 14101-14108,2008,doi:https://doi.org/10.3182/20080706-5-KR-1001.02391.

[6] A. Zeid, S. Sundaram, M. Moghaddam, S. Kamarthi, and T. Marion, "Interoperability in Smart Manufacturing: Research Challenges," Machines,Review vol. 7, no. 2, 2019, doi: 10.3390/machines7020021.

[7] D. Serpanos and M. Wolf, "Industrial Internet of Things," IoT Systems: Architectures, Algorithms, Methodologies. Cham: Springer International Publishing, 2018, pp. 37-54.

[8] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, Opportunities, and Directions,"IEEE Trans. Ind. Inform., Article vol. 14, no. 11, pp. 4724-4734, 2018, doi: 10.1109/tii.2018.2852491.

[9] N. Kshetri, "Can Blockchain Strengthen the Internet of Things?," IT Prof.,vol.19,no.4, pp. 68-72, 2017, doi: 10.1109/MITP.2017.3051335.

[10] S. Cavalieri, "Semantic Interoperability between IEC 61850 and oneM2M for IoT-Enabled Smart Grids," Sensors, Article vol. 21, no. 7, 2021, doi: 10.3390/s21072571.

[11] P. Desai, A. Sheth, and P. Anantharam, "Semantic Gateway as a Service architecture for IoT Interoperability," IEEE Int. Conf. on Mob. S., NewYork, 2015, pp. 313-319, doi: 10.1109/ms.2015.51.

[12] I. Grangel-González, Semantic Data Integration for Industry 4.0 Standards. 2017, pp. 230-237.

[13] H. Derhamy, J. Eliasson, and J. Delsing, "IoT Interoperability—On-Demand and Low Latency Transparent Multiprotocol Translator," IEEE IoT Journal, vol.4, no.5, pp.1754-1763, 2017, doi: 10.1109/JIOT.2017.2697718.

[14] V. Barros, S. Junior, S. Bruschi, F. Monaco, and J. Estrella, "An IoT Multi-Protocol Strategy for the Interoperability of Distinct Communication Protocols applied to Web of Things," Brazillian Symposium on Multimedia and the Web, 2019.

[15] H. Derhamy, J. Rönnholm, J. Delsing, J. Eliasson, and J. v. Deventer, "Protocol interoperability of OPC UA in service oriented architectures," IEEE Int. Conf. on Ind. Inf. (INDIN), 2017.

[16] C. Wilder, M. Jose, P. Harold, and J. D. Alvarado, "Internet of things: a multiprotocol gateway as solution of the interoperability problem," in Mechatronics, Electronics and Telecommunications Advances Towards Industry 4.0, Bonaventuriana Ed., 2021, ch. 5, p. 24.

[17] T. John and M. Vorbröcker, "Enabling IoT connectivity for ModbusTCP sensors," IEEE Int. Conf. on Emerging Tech. and FactoryAutomation(ETFA),2020.Available:https://ieeexplore.ieee.org /document/9211999/.

[18] R. Light, "Mosquitto: server and client implementation of the MQTT protocol," The Journal of Open Source Software, vol. 2, 2017, doi: https://doi.org/10.21105/joss.00265.

[19] N. Q. Uy and V. H. Nam, "A comparison of AMQP and MQTT protocols for Internet of Things," NAFOSTED Conf. on Info. and C.

Science(NICS),2019.Available:https://ieeexplore.ieee.org/document/9023812/.

[20] I. González, A. J. Calderón, and J. M. Portalo, "Innovative Multi-Layered Architecture for Heterogeneous Automation and Monitoring Systems: Application Case of a Photovoltaic Smart Microgrid," Sustainability, vol. 13, no. 4, p. 2234, 2021, doi: https://doi.org/10.3390/su13042234.

[21] Maciej Wasilak and C. Amsüss. "chrysn/aiocoap.", 2014, http://github.com/chrysn/aiocoap/

[22] L. T. D. Paolis, V. D. Luca, and R. Paiano, "Sensor data collection and analytics with thingsboard and spark streaming," IEEE W. on Environmental, Energy, and Structural Monitoring Systems (EESMS),2018, pp. 1-6, doi: 10.1109/EESMS.2018.8405822.