2019 19th International Symposium on Communications and Information Technologies (ISCIT)

# Development An Open-Source Industrial IoT Gateway

Phuc Nguyen-Hoang
*School of Electrical Engineering*
*International University - VNU HCMC*
Hochiminh city, Vietnam
jphuc96@gmail.com

Phuoc Vo-Tan
*School of Electrical Engineering*
*International University - VNU HCMC*
Hochiminh city, Vietnam
vtphuoc@hcmiu.edu.vn

***Abstract*—Bringing field devices and sensors to the cloud allows production plant to be connected, remotely monitored and controlled is a recently trending in industrial automation. The combination between SCADA and addition gateway, router with IoT protocols such as MQTT, CoAP... is a traditional solution in practice. This paper provides an alternative approach called Industrial IoT gateway and based on Linux tool-chain to build ”unique” software and hardware solution for cloud connectivity. The set of Linux-based software including Docker and Node-RED allows generating, packaging and quickly deploying application that facilitates the data exchange from different industrial fieldbuses (Siemens S7, Modbus TCP and RTU) and IoT protocols (MQTT, HTTP). An industrial plant prototype is built and communication latency is also taken into consideration for evaluating the proof of concept.**

***Index Terms*—Industrial IoT Gateway, Docker, Node-Red, S7 protocol, Modbus TCP/RTU, MQTT, REST API**

## I. Introduction

The convergence of operations technology (OT) and information technology (IT) within a manufacturing process is one of the key factors in the fourth industrial revolution or Industrial Internet of Things (IIoT for short). Production data from manufacturing sites and business information are centrally collected into Cloud and then are analyzed to make business decision and production planing. The manufacturing companies clearly get a lot of benefit from fusing these technologies such as gaining competitive advantage, increasing their operational efficiency. The role of IIoT today focuses on monitoring/supervision activities rather than substituting the existing automation devices [1].

The communication in IIoT is machine oriented with different wired or wireless protocols. The Open Process Communication Unified Architecture (OPC UA) released in 2008 is a tradition solution for ensuring the interoperability between machines in manufacturing sites. According to [2], the Unified Architecture provides service oriented architecture (SOA) for integrating old OPCs classic within a concrete information modeling framework, platform independent, security, extensible. However, we believe that OPC UA need additional gateway and router with IoT protocols such as MQTT, CoAP to exchange data with the Cloud Server.

The OT/IT gap and interoperability issue can be solved using a concept called heterogeneous IoT gateway as in [10]. They employed similar approach of OPC UA framework that defined a interchange data structure including field-buses (CAN,Modbus RTU/TCP) and IoT protocol (CoAP and MQTT). In [11], the authors proposed an IoT gateway to exchange Simatic S7-1200 PLC data with IBM cloud service using MQTT protocol and NodeRED [9]. The advantage of this approach is clearly shown by the quick and simplicity of NodeRED programming. In addition, there is only operational data sharing to the Cloud Platform without any VPN or remote connection to the internal network. In [12], the authors proposed distributed industrial IoT gateway concept which exchange data from S7 and Modbus TCP to Cloud server. The gateway is hosted on Raspberry Pi 2 and is performed stress-test with thousand packets sending by multiple Intel Xeon IoT nodes. The result is very promising with latency less than one second. However, their gateway is based partially on the proprietary software development kit.

Within this context, the main contribution of this paper is to exploit the micro-service of Docker [3] for developing an open-source industrial IoT gateway. Different industrial protocols such as Siemens S7, Modbus RTU/TCP and IoT protocol MQTT (for IBM Watson) and REST API (for AWS EC2) are implemented by NodeRED. They are then be put in containers using containerize service of Docker. All containers considered as micro-services work independently and harmonically with Docker engine under Raspberry Pi 3 - gateway host. The second contribution is based on the fact that this paper proposed a structure for bridging the IT/OT gap. The gateway can be implemented with an existing OPC UA framework. Within Docker environment, micro-services (protocols) can be flexibility added or removed from gateway easily. In case the number of micro-services increases, we can think about adding more gateways or increasing the network infrastructure with clustering like the success story of Docker [3], Google - Kubernetes [4]

The paper is organized as follows. Section II introduces the industrial plant prototype used in this paper to demonstrate the feasibility of our proposed solution. All related protocols discussed on the above paragraph are also briefly recalled. Section III describes in details the design of IIoT gateway. Section IV provides the overall result and the communication latency.

## II. Industrial Plant Prototype and Related Fieldbuses and IoT Protocols

Consider a air flow process given in Fig. 1. This industrial plant prototype is governed by a variable frequency drive (VFD for short) INVT GD-20 (A) that controls the rotational speed of the three-phases motor or the small turbine (B) to generate various air-flow. A Simatic S7-1200 programmable logic controller (C) is designed for ensuring the safety of industrial plant. It can monitor important operation data of three-phase motor such as speed, power and torque. In addition, it can provide some safety functions such as turning VFD on/off, changing direction of rotation and speed.

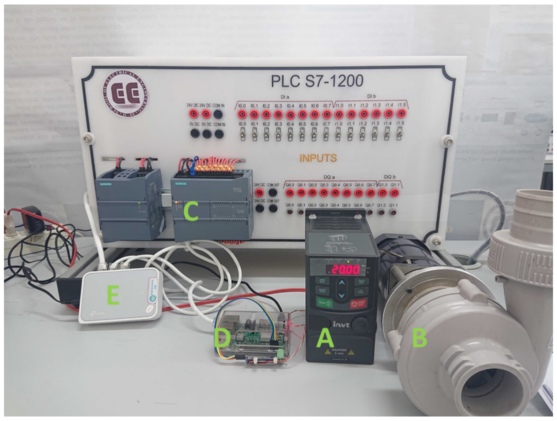

Fig. 1. Prototype used for testing our IIoT Gateway.

VFD uses Modbus RTU protocol and communicates directly with our proposed gateway Raspberry Pi (D) through RS-485/UART level converter. An IEEE 802.3 Ethernet hub (E) covers other communications of the industrial plant prototype, communication between gateway and Cloud server. All protocols discussed in this paper is depicted in Fig. 2 and will be briefly introduced in the sequel.

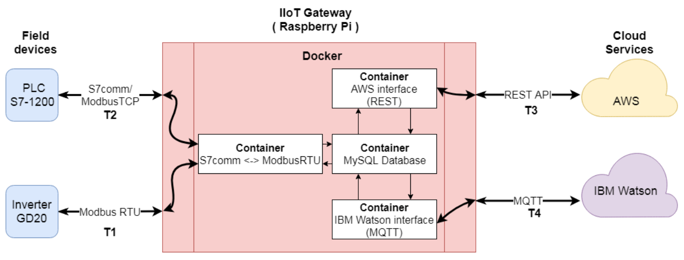

Fig. 2. Communication protocols discussed in this paper.

### A. Gooddrive GD-20 Inverter and Modbus RTU

The air flow process under consideration is simply an open-loop system where the rotation speed of 3 phases motor is controlled by writing the different value of register 0x2000H according to the manual of INVT GD-20 [5]. In addition, important operational data of VFD (output voltage, current) and motor (speed, power and torque) can be monitored by reading value of register ranging from 0x3000H to 0x3007H. The communication between VFD and Raspberry Pi gateway uses Modbus RTU protocol through physical line UART/ RS-485 level converter.

### B. Simatic S7-1200 PLC, Siemens S7 and Modbus TCP protocol

Simatic S7-1200 PLC (PLC for short) are responsible for local monitoring and safety protecting of air flow process. Operational data from the VFD are first encapsulated/de-encapsulated by gateway either in Siemens standard proprietary communication S7 or in Modbus TCP protocol. They are then exchanged with PLC via the Ethernet hub. On the PLC side, data is mapped directly into data block memories if the proprietary S7 protocol is used. In case of using Modbus TCP, data should be passed to the Modbus TCP block before mapping into data block memories. According to Subsection II-A, there are totally eight operational data to be monitored in PLC and two more data used for safety controlling. They are mapped directly into data block memories in PLC and takes 4 bytes (2 DB words). Both S7 and Modbus TCP share the same three lower layer in TCP/IP stack. A comparison TCP frame structure of S7 and Modbus TCP protocol is shown on Fig. 3. They are both relied on TCP/IP model where Modbus TCP is in the application layer and S7 use ISO TCP on the top of TCP/IP model.

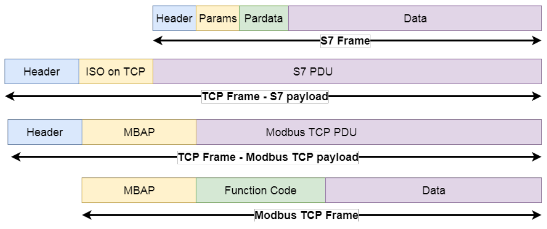

Fig. 3. TCP frame structure of S7 and Modbus TCP protocol.

### C. IoT Protocols : MQTT and REST API

In order to demonstrate the cloud connectivity, both MQTT and REST API are used since they are popular IoT protocols and are supported by Cloud services such as IBM Watson and AWS. These protocols are both in application layer of TCP/IP stack and use TCP protocol in their transport layer, client-server style. However, there are some important differences that make the advantages of one compared to another. The Representational State Transfer (REST) introduced by Feilding in [7] is a software architecture for building Web service. Data addressed by Uniform Resource Identifiers (URIs) is exchanged between Client and Server using HTTP methods. IoT services could be deployed quickly using REST API since it inherits the existing web infrastructure. The main drawback of using HTTP-based protocol is synchronous request-response pattern that leads to the delay time for processing a request or waiting a response. This delay time issue could get worse due to the network scale with diverse sizes of constrained devices.

On the other hand, MQTT is characterized by lightweight machine-to-machine communication protocol since data is message-oriented with compact header size. Data is also filtered as a topic and sent from one publisher to one/many subscriber(s) of the topic through a third-party called broker. Publishers, subscribers on the network are considered as clients and broker as server. MQTT is a effective communication method for resource-constrained devices. Several mechanisms described in [8] are proposed to ensure the effectiveness of exchanging data in such circumstance. The most important feature of this protocol is that clients communicate asynchronously. In case of absence publisher or subscriber, data is still queued and sent in the corresponding topic in broker. Thus, a sudden connection lost does not cause catastrophic failure of entire system.

## III. Design Industrial IoT Gateway

The design requirements of our proposed IIoT gateway are to firstly fulfill traditional SCADA applications that allow data exchange between local nodes. In this case, PLC can monitor and control locally the air flow process, eg. VFD device. Secondly, the gateway can send data into Cloud services without using addition router, proxy as in existing commercial SCADA products. Keeping these requirements in mind, the hardware and software solution for our IIoT gateway will be introduced in the following paragraph.

The Raspberry Pi 3 model B+ is chosen as our hardware solution since its low-cost, popular and available of UART and IEEE 802.3 PHY interfaces used in our industrial plant prototype. Besides that, the Raspbian Jessie is also an advantage since the our software solutions are mainly Linux-based, open source and official supported by this distro. Notice that all communication protocols described in Section II are mostly used TCP in their transport layer. The Modbus RTU of the VFD is an exception case since it exchanges data directly with our gateway via UART port. However, these data will be wrapped or unwrapped to TCP stream or serial data respectively by the gateway. Consequently, the choice of Raspberry Pi is adequate and only one serial UART/RS485 level converter is required.

The software solution is aimed to build up the peer-to-peer communication that encapsulate/de-encapsulate the incoming serial frames/TCP packages into different structures relied on TCP such as Siemens S7, Modbus TCP/RTU, MQTT and REST API as shown in Fig. 2. The peer-to-peer communication used Node-RED, an open source flow-based programming language and initiated by IBM [9]. Node-Red is one of our best references in this situation which helps us fulfilling the most software tasks without writing everything from scratch. The SCADA functions taken placed between PLC and VFD can be considered a first Node-RED process running in the gateway. It also copied SCADA operational data into local database and MySQL engine can be considered as the second process. The third and fourth are used to send operational data from local database to Cloud servers. Once all process are deployed successfully, they are packaged individually and then activated by Docker engine running in Raspbian Jessie. All process will be described in the sequel.

The first task is to monitor and control the air flow process locally. It was done by creating a flow program described in Fig. 4 to facilitate the two ways data exchange from PLC (S7 protocol) to VFD (Modbus RTU) using Modbus read/write and S7 in/out nodes. Different addresses corresponded to VFD's registers discussed in Subsection II-A are used to perform the reading/writing the monitoring and safety control information using Modbus read/write nodes. Data is sent to PLC by feeding directly to the corresponding S7-out nodes. Consequently, PLC will save them in their data block memories. The same fashion will be applied to perform safety control from PLC to VFD using S7 in and Modbus write nodes. In addtion, operation data are also save in gateway's SQL database using SQL node.

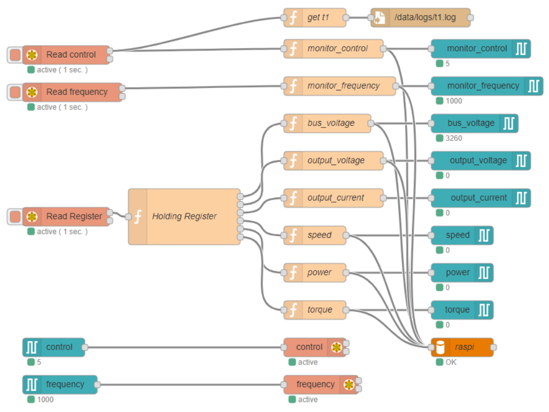

Fig. 4. Node-RED program for exchanging data between PLC and VFD

A Node-RED flow is deployed to push data on SQL database to the corresponding end-points of Amazon Elastic Compute Cloud (Amazon EC2) using HTTP methods. The program is given in Fig. 5. At the Amazon EC2 side, a Linux-based instance deployed RESTFUL server is created. It is simply a node.js application and uses Node-RED flow program handling all requests with their specific end-points coming from gateway and return corresponding responses.

Another Node-RED flow given in Fig. 6 is also built to send the data in SQL database to the Cloud server using MQTT. In our case, IBM Watson platform, which is a MQTT broker, was used.

## IV. Results and Discussion

After integrating all Node-Red functions with Docker, all requirements of our proposed gateway, mentioned in Section III,

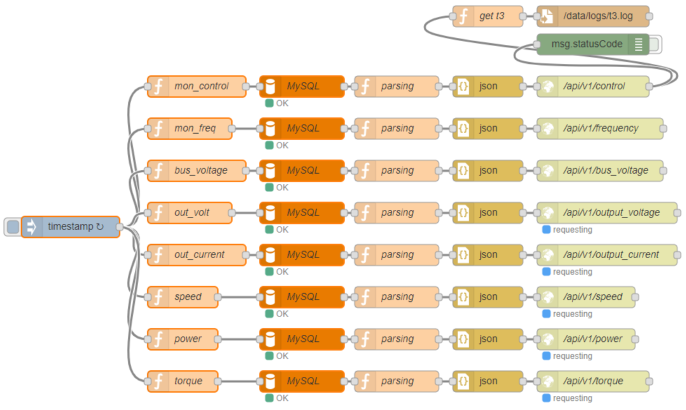

Fig. 5. Node-RED flow for sending data to RESTFUL server on AWS

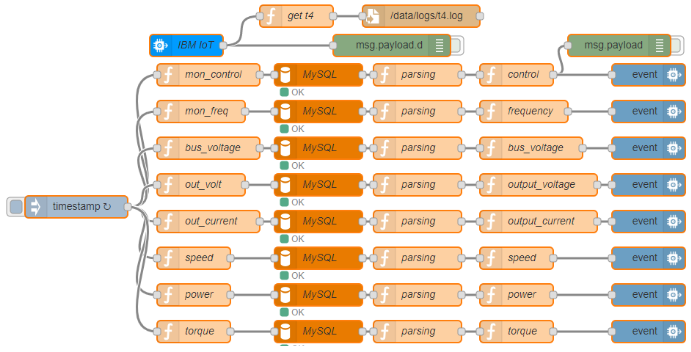

Fig. 6. Node-RED flow for sending data to IBM Watson platform

are verified. The local control and monitoring are performed with TIA portal v.13, two dashboards in AWS EC2 and IMB Watson with the corresponding charts are created and available at http://bit.ly/2vh8poM. Besides that, the time from T1 to T4 are also measured in order to test the communication latency. Note that all communications are taken place in the gateway. Thus, communication time was simply taken by measuring the local timestamp of Raspberry Pi. The measurement process is then repeated, taken minimum, maximum and average. The communication latency is given in the Table I. T1 and T2, appeared in both Fig. 2 and Fig. 4, are the interval between two consecutive reading/writing PLC, VFD respectively. T3 and T4, described consecutively in Fig. 2, Fig. 5 and Fig. 6, are the duration to send data the Cloud server. These intervals were taken with upload/download speed around 3 Mbps. The measuring of T3 and T4 may not make sense since they depend strongly on the internet speed and the server's loading. However, these data provide at least a critical feedback on the feasibility of proposed IIoT gateway with our industrial plant prototype.

The sum of T1 and T2 is the time for exchanging data locally, eg. between PLC and VFD. The total of T2 and T3 is considered as the duration for sending data from VFD to RESTFUL server. In case of pushing data to MQTT server, it takes the total of T2 and T4. According to Table I and Fig. 4-6, a payload of thirty six bytes has been exchanged through gateway within 100ms latency which is acceptable for our industrial plant prototype.

TABLE I
COMMUNICATION LATENCY IN MILLISECONDS

| | *Min.* | *Max.* | *Average* |
|---|---|---|---|
| **T1** | 11 | 36 | 13.44 |
| **T2** | 5 | 203 | 54.4 |
| **T3** | 1 | 61 | 21.38 |
| **T4** | 2 | 51 | 11.95 |

## CONCLUSION

This paper proposed an alternative SCADA solution without using OPC UA framework and its additional gateways or proxy. An industrial plant prototype has been built and evaluated the feasibility of our proposed IIoT gateway which is based on Raspbian, container technology Docker and flow based programming Node-RED. The main contribution relies on the micro-service of Docker which is considered similar to the concept service oriented architecture of OPC UA framework. The advantages have been clearly demonstrated where the fieldbuses and IoT protocols are deployed in a unique device called IIoT gateway and they can be enable/disable or even added more services by Docker. This work can be extended for other industrial process. It leads to the future investigation on the security aspect or the combination of Docker and OPC UA for a complex network structure.

## REFERENCES

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag and M. Gidlund, "Industrial Internet of Things: Challenges, Opportunities, and Directions," in IEEE Transactions on Industrial Informatics, vol. 14, no. 11, pp. 4724-4734, Nov. 2018.
[2] "Open Process Control", Accessed on: Mar. 16, 2019. [Online]. Available: https://opcfoundation.org
[3] "Docker", Accessed on: Mar. 16, 2019. [Online]. Available: https://www.docker.com
[4] "Google Kubernetes", Accessed on: Mar. 16, 2019. [Online]. Available: https://kubernetes.io
[5] "GD-20 User Manual", Accessed on: Mar. 16, 2019. [Online]. Available: https://www.invt.com/Ajax/Download.aspx?pid=2251
[6] "Modbus Protocol Specification", Accessed on: Mar. 16, 2019. [Online]. Available: http://www.modbus.org/specs.php
[7] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D dissertation, University of California, Irvine, 2000. Accessed on: March 16, 2019. [Online]. Available: https://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm
[8] "MQTT Protocol Specification", Accessed on: Mar. 16, 2019. [Online]. Available: https://www.oasis-open.org/committees/mqtt/
[9] "NodeRED", Accessed on: Mar. 16, 2019. [Online]. Available: https://nodered.org/about/
[10] V. Kulik and R. Kirichek, "The Heterogeneous Gateways in the Industrial Internet of Things," 2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Moscow, Russia, 2018, pp. 1-5.
[11] A. Gavlas, J. Zwierzyna, J. Koziorek, Possibilities of transfer process data from PLC to Cloud platforms based on IoT, IFAC-PapersOnLine, Vol. 51, Issue 6, 2018, Pages 156-161.
[12] M. Hemmatpour, M. Ghazivakili, B. Montrucchio and M. Rebaudengo, "DIIG: A Distributed Industrial IoT Gateway," 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, 2017, pp. 755-759.