Journal of Multimedia Information System VOL. 9, NO. 3, September 2022 (pp. 209-218): ISSN 2383-7632 (Online)
https://doi.org/10.33851/JMIS.2022.9.3.209

Smart Sensor Management System Supporting Service Plug-In in MQTT-Based IIoT Applications

Young-Ran Lee[1], Sung-Ki Kim[1*]

Abstract

Industrial IoT applications, including smart factories, require two problem-solving to build data monitoring systems required by services from distributed IoT sensors (smart sensors). One is to overcome proprietary protocols, data formats, and hardware differences and to uniquely identify and connect IoT sensors, and the other is to overcome the problem of changing the server-side data storage structure and sensor data transmission format according to the addition or change of service or IoT sensors. The IEEE 1451.4 standard-based or IPMI specification-based smart sensor technology supports the development of plug-and-play sensors that solve the first problem. However, there is a lack of research that requires a second problem-solving, which requires support for the plug-in of IoT sensors into remote services. To propose a solution for the integration of these two problem-solving, we present a IoT sensor platform, a service system architecture, and a service plug-in protocol for the MQTT-based IIoT application environment.

Key Words: Service Plug-In, Smart Sensor Management, IIoT Application, Smart Factory, Plug-and Play Sensor.

I. INTRODUCTION

The system architecture of industrial IoT applications, including smart factory and process plant, is developing based on a service-oriented architecture that supports customized production management based on customer order [1]. This is because it is necessary to meet the demand for timely release of new products and to support elastic changes in the manufacturing process according to market demand patterns for products and price change of product raw materials. Therefore, smart factory systems are required to be designed with the flexibility of adding, removing, and changing system components while maintaining the quality of services of the system. This needs a low-cost solution to manage changes in the attributes of management resources that make up systems such as sensors, actuators, and application services, including routine maintenance problems. To meet these needs, the architectural features of the system for the industrial IoT application shows the following three changes.

- Sensor Layer: widely use of smart sensors to support implementation of plug-and-play concept when connecting sensors to network edge devices.
- Middleware Layer: Enhancement of middleware capabilities that enable transparent connectivity of application services, sensors, and subsystems. regardless of the underlying protocol implementations.
- Application Layer: Enrichment of software development framework to support data format-independent application service implementations.

Open source hardware-based IoT sensor terminals that integrate embedded systems, wireless communication, and sensor technologies into one device are becoming widespread. The popularity of IoT device development using Arduino, Raspberry Pi, and Intel Edison products is driving market changes to replace existing industrial control system technologies based on PLC and PC with IoT technologies. These market changes are making various attempts to solve the problem of space constraints and reduction in implementation costs that existing industrial control systems do not solve to build the smart factory applications through IoT technology applications.

In smart factory applications, the management of sensor data requires two challenges to be addressed. The first challenge is to design a manageable sensor data structure in consideration of IoT sensor devices in heterogeneous distributed environments. According to the addition, removal, and change of the sensor, dynamic change of device management information is accompanied. Therefore, it is necessary to design a structure of storage data to accumulate sensor

Manuscript received August 29, 2022; Revised August 31, 2022; Accepted September 6, 2022. (ID No. JMIS-22M-08-028)
Corresponding Author (*): Sung-Ki Kim, +82-41-530-2466, skkim@sunmoon.ac.kr
[1]Dept. of IT Education, Sun Moon University, Asan, Korea, yrlee@sunmoon.ac.kr, skkim@sunmoon.ac.kr

data by reflecting these characteristics.

The second challenge is to design a system architecture that supports solving the first challenge. Whenever IoT sensor devices are installed, changed, and removed every time, sensor management information suitable for sensor data storage structure cannot be obtained manually through physical access. Therefore, it is necessary to design a system architecture and protocol between IoT sensor devices and application services in which IoT sensor devices and servers interact so that sensor data with sensor data structures that meet service needs can be collected.

These two challenges require the implementation of the concept of two consecutive plug-and-play as shown in Fig. 1. In Fig. 1., the implementation of the "1st plug-and-play" means plug-in the sensors into the IoT device, and the "2nd plug-and-play" means plug-in the IoT devices into services. Plug-in the sensors into the IoT device requires a user (or field operator) to do very time-consuming works such as the following: After physically connecting sensors and IoT devices, the user must run the software of the DAQ (Data Acquisition) system, set each sensor individually, select the correct gain, enter the engineering unit, apply appropriate scaling and offset factors, and name each channel. This work will run several hours of inquiry between the datasheet and the DAQ system software for each sensor, sometimes tracking the signal cable from the sensor to the DAQ system to ensure that it has not been misconnected. Moreover, in this work, it is too difficult to consider the concept of enterprise-level management in relation to sensors.

An open industry standard to support the development of plug-and-play smart sensors is the IEEE 1451.4 smart transducer interface standard [2]. Sensor descriptor data, called TEDS (transducer electronic data sheet), is stored in memory (PROM, EEPROM) within the sensor. TEDS data specifies the type of sensor present, describes its interface, and provides technical information such as manufacturer, type number, serial number, sensitivity, calibration date, reference conditions [3].

Using TEDS data, it is possible to develop various sensor management and monitoring applications, including automatic discovery and configuration of sensors. Sensor devices developed based on the IEEE 1451.4 standard are also called the plug-and-play sensor or the TEDS sensor [4]. TEDS sensor solves the difficulties users face when plugging in sensors to IoT devices through TEDS data provided by sensor device manufacturers and TEDS application software that deals with them. The TEDS application software connects computers and sensor devices with standard interfaces such as USB and RS-232C, and supports reading, editing, converting, and writing of TEDS data in the sensor's internal ROM memory.

To create TEDS sensors using a legacy sensor that does not support TEDS, there is a way to use ROM memory such as Maxim's DS24B33 EEPROM, which supports the Master/Slave 1-wire protocol. After interfacing this EEPROM between IoT devices and legacy sensors, TEDS data is written to the EEPROM via software programming. Through the connection between PC and IoT devices, TEDS data can be written in EEPROM using open source OneWire library [5]-based programming [6]. Another way to create TEDS sensors using legacy sensors is to use software that supports DAQ and TEDS data editing, such as LABVIEW of National Instrument. These software supports the creation of virtual TEDS data files for legacy sensors at the software level. Sensor device developers can edit the desired sensor management information into a virtual TEDS file and then use the software to write the virtual TEDS file in EEPROM in binary format [7]. This is similar to writing the firmware software in ROM via Rom Writer software after program source coding is completed.

In addition, there is a standard called the intelligent platform management interface (IPMI) [8], which provides the hardware management interface required to monitor the performance and status of the hardware on most PCs and server computers. Windows and Linux OS provide the implementation of this IPMI standard as a set of system commands

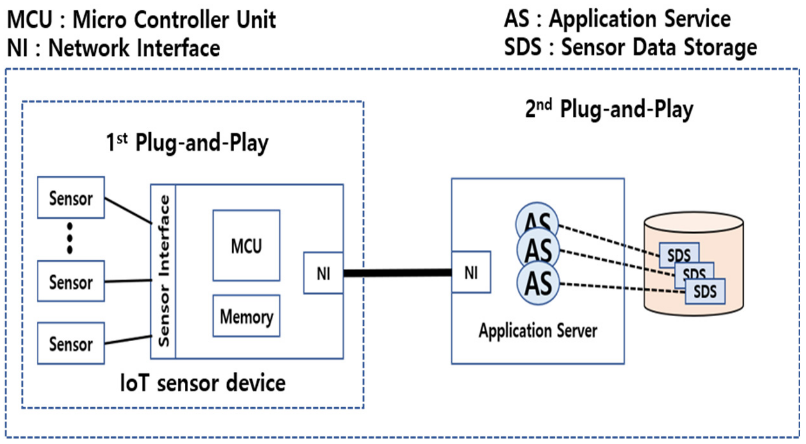


Fig. 1. The challenge of our research work: How to integrate the concept of two consecutive plug-and-play for smart sensor management.

and programming interfaces [9]. In industrial IoT applications, development of plug-and-play smart sensor applications based on IPMI interfaces using small PCs is active. There is a method[x] of storing and applying sensor management information in SDR (Sensor Data Repository) [10] in a form similar to TEDS data structure by connecting sensors and small PCs with USB, Bluetooth, and WiFi.

Even if IoT sensor devices support IEEE 1451.4 standards, it is difficult to automatically plug devices into smart factory or process plant control systems. For this to be possible, the system must support three problem-solving. Distributed IoT sensor devices should be connected to appropriate services, data collection protocols that conform to service data storage structures should be supported, and management of IoT sensor devices should be supported at the enterprise level. if IoT sensor devices support IEEE 1451.4 standards, it is difficult to automatically plug devices into smart factory or process plant control systems. For this to be possible, the system must support three problem-solving. Distributed IoT sensor devices should be properly connected to services, data collection protocols that conform to the service data storage structure should be supported, and management of IoT sensor devices should be supported at the enterprise level.

When management events such as physical addition, removal, and modification of IoT sensor devices are issued as maintenance activities, the management domain, production line, control group, device identifier and service connection network of IoT sensor devices are not automatically maintained.

Therefore, the system should be able to maintain the services regardless of the occurrence of the management events.

When various sensors are connected to a network edge node, the research results to support the concept of a plug-and-play sensor are as follows.

[11] presents a virtual TEDS-based framework that helps the development of plug-and-play IoT sensor devices (smart sensor devices) for web of things implementation without TEDS data structures and firmware programming knowledge. [12] presents a smart sensor implementation that performs sensor management in remote applications by embedding TEDS data in an embedded sensor device. [13] presents the development result of managing the RFID sensor system through a virtual TEDS data editing tool that supports RFID sensor data management. [6] introduces an approach that turns non-TEDS sensor devices into plug-and-play smart sensors. This work also introduces the development method of a writer tool that stores the virtual TEDS created using a commercial virtual TEDS editing tool as binary TEDS data in the EEPROM of the sensor device, and a mobile parser that converts the binary TEDS data into programmable data.

The research works that provide flexibility in system resource management by connecting distributed IIoT devices on a service-oriented basis are as follow.

[14] presents a framework for integrating and managing industrial IoT devices using OpenPnP reference structures. [15] presents an approach to the service-oriented system resource management by designing the OPC UA system architecture, which is a de facto standard in the traditional control system industry, based on the RESP protocol.

[16] presents a method to analyze the requirements for managing IoT devices based on open source platforms and evaluate solutions that correctly connect to application needs. [17] presents a service management method that supports the self-awareness of low-power IoT edge devices. [18] presents a virtual service device management technique for building a Universal PnP-based IoT network. [19] presents challenges and solutions in edge computing for building industrial IoT systems.

As the interface technology of the sensor device that measures a physical quantity continues to develop, it has come to support wired and wireless Internet connection. Therefore, the IEEE1451 standard is also upgrading the version. Existing research works have proposed methods to describe the specification of a sensor device based on the IEEE 1451.4 TEDS template and to embed it in the sensor internal memory as binary data. However, when a sensor management event occurs, it is difficult to find works in which a sensor device prepares a data protocol suitable for a service data storage structure. Therefore, a research work that implements the concept presented in Fig. 1 is necessary. Moreover, existing research on sensor management does not focus on management events such as process management and maintenance problem solving in industrial IIoT environments. Therefore, it is difficult to find a way to support maintenance of the management context in response to changes in sensor device and service implementation because management issues such as replacement, addition, and removal of sensors are not addressed. In most sensor management works, the focus of management is on the identification and control of sensors for the implementation of a single service. Our research work began with recognizing this problem.

By extending and integrating these results of research works, we propose a system architecture and protocol in which IoT sensor devices are plugged into services when management events such as addition, removal, and change of IoT sensor devices occur. Chapter 2 presents the IoT sensor platform architecture that configures sensor management information based on the TEDS template standard when a sensor is connected to an IoT device that is a network edge device. Chapter 3 presents an MQTT-based service structure in which IoT sensor devices and services automate discovery and service connection based on MQTT publishing and subscription protocols, and complete a manageable service-oriented data format. Chapter 4 discusses the results and chapter 5 concludes the paper.

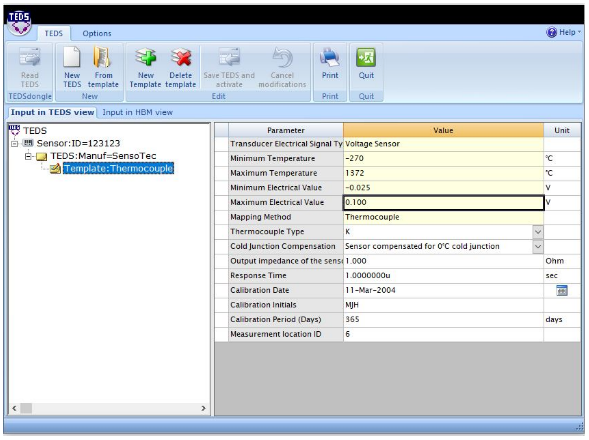
II. DESIGNING OF AN IoT SENSOR PLATFORM

2.1. Sensor Information Manager

Fig. 2 shows the IEEE 1451.4 TEDS template structure. It can be divided into four main sections. The first basic TEDS is a section that stores identifier information essential for sensor management. The second standard template TEDS section shows the sensor specifications according to the sensor type ID. The third calibration TEDS Template section shows the calibration information for the sensor selected in the second section. When you select a sensor type ID in the second Standard Template TEDS section, the Basic TEDS section and the calibration section change to information about that sensor. The basic TEDS section is important management information that identifies the sensor. We call the five pieces of information in the basic TEDS section "physical sensor information". The method of storing this information together with logical sensor information as SMI (sensor management information) in the flash memory of Fig. 5 was reflected in the design. Logical management information includes the management domain that distinguishes between “Factory1” and “Factory2”, the process name meaning “painting line” of “Factory1”, and the device group name included in the spray pressure control in the corresponding process.

TEDS data embedded in commercial IEEE 1451.4 compliant smart sensor is data in bit stream format. It can be accessed through the physical interface (TEDS dongle, USB etc.) provided by the manufacturer, the TEDS data editing tool (DAQ software), or the SDK. Therefore, in order to extract the management information about the sensor device, programmable text-based data such as XML and JSON must be obtained using the TEDS data editing tool or SDK. Fig. 3 shows the virtual TEDS template of the thermocouple sensor opened with HBM's TEDS Editor. Fig. 4 shows the result of converting this template data to XML data. Fig. 5 shows our proposed IoT sensor platform for sensor management.

As a network edge node, the IoT sensor platform includes a TEDS driver that connects commercial IEEE 1451.4 compliant smart sensors. Most TEDS drivers can read TEDS data from IEEE 1451.4 compatible sensors (TDES sensors). It is possible to convert binary TEDS data into XML or JSON format using the API provided by the SDK. In Fig. 5, the sensor information manager performs the function of extracting management information of TEDS sensors connected to the IoT sensor platform. Non-TEDS sensors, such as legacy sensors or DIY-manufactured sensors, need to be upgraded to TEDS sensors in the steps of [5] and [6] before installation. However, the sensor information manager manages the sensor management information in JSON or XML format before solving the driver



TEDS
Sensor:ID=123123
TEDS:Manuf=SensoTec
Template:Thermocouple

Parameter	Value	Unit
Transducer Electrical Signal Ty	Voltage Sensor	
Minimum Temperature	-270	°C
Maximum Temperature	1372	°C
Minimum Electrical Value	-0.025	V
Maximum Electrical Value	0.100	V
Mapping Method	Thermocouple	
Thermocouple Type	K	
Cold Junction Compensation	Sensor compensated for 0°C cold junction	
Output impedance of the sens	1.000	Ohm
Response Time	1.0000000u	sec
Calibration Date	11-Mar-2004	
Calibration Initials	MJH	
Calibration Period (Days)	365	days
Measurement location ID	6	

Fig. 3. Thermocouple template in commercial TEDS editor [21].

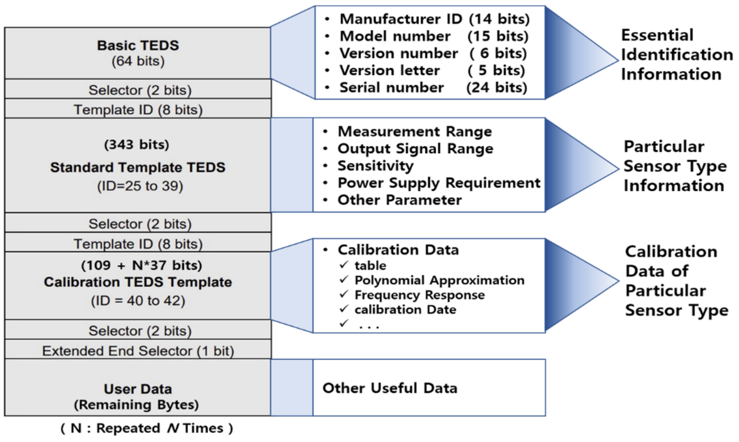


Fig. 2. IEEE 1451.4 TEDS template standard [19-20].

```xml
<?xml version="1.0" encoding="utf-8"?>
<TEDS ID="123123">
    <BasicTEDS>
        <Manufacturer>SensoTec</Manufacturer>
        <Model>0</Model>
        <VersionLetter>A</VersionLetter>
        <VersionNumber>1</VersionNumber>
        <SerialNumber>586124</SerialNumber>
    </BasicTEDS>
    <TEDS-Code length="10">
      2800002004
    </TEDS-Code>
    <USER-Section>
    <Commentary_Serialnumber>
        586124
    </Commentary_Serialnumber>
    <Commentary_Modeltype> 0 </Commentary_Modeltype>
    <Commentary_Calibrationtype> Test protocol
    </Commentary_Calibrationtype>
    <Commentary_Date>2022-08-28</Commentary_Date>
    </USER-Section>
  </TEDS>
```

Fig. 4. Thermocouple TEDS data expressed in XML.

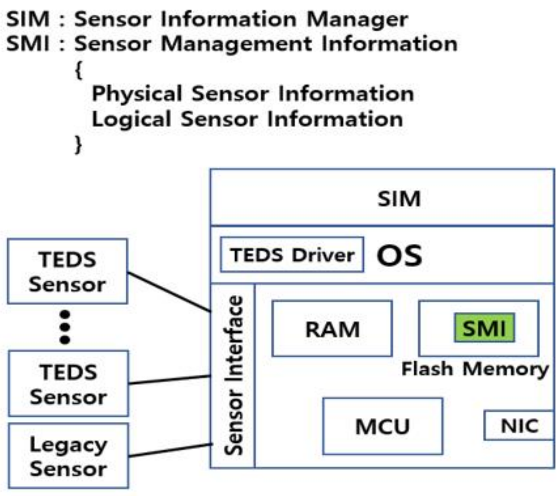

SIM : Sensor Information Manager
SMI : Sensor Management Information
{
Physical Sensor Information
Logical Sensor Information
}
SIM
TEDS Driver
OS
TEDS Sensor
TEDS Sensor
Legacy Sensor
Sensor Interface
RAM
SMI
Flash Memory
MCU
NIC

Fig. 5. Proposed IoT sensor platform.

compatibility and connection trust issues. The sensor management information managed by the sensor information manager is used for configuration for MQTT publish.

2.2. Sensor Management Information

SMI in Fig. 5 is a sensor management information file. This file contains information extracted from the Basic TEDS section of the sensor TEDS data and logical management information. The following Fig. 6 is the JSON code that expresses this.

```
{ "SMI" : {
      "PSI" : {
        "TEDS" : {
            "ID" : "123123",
            "BasicTEDS" : {
                "Manufacturer" : "SensoTec",
                "Model" : 0 ,
                "VersionLetter" : "A" ,
                "VersionNumber" : 1 ,
                "SerialNumber" : "586124" ,
            },
  "TEDS-Code" : {
  "length" : 10,
  "2800002004",
  },
  "USER-Section" : {
   "Commentary_Serialnumber" : "586124" ,
   "Commentary_Modeltype" : 0 ,
   "Commentary_Calibrationtype": "Test protocol",
   "Commentary_Date" : "2020-08-28",
},
  "LSI" :,
     "factory1/process/device_group"
  }
}
```

Fig. 6. JSON data for sensor management information.

III. PROPOSED SYSTEM AND ALGORITHM

3.1. Proposed System Architecture

The system architecture to support the service plug-in of the smart sensor device is shown in Fig. 7. The IoT sensor device completes MQTT discovery initialization by including the configuration in which its SMI is described in the service topic in order to plug in the service suitable for the sensor type.

In Fig. 7, the solid line means MQTT publish connection and the dotted line means MQTT subscribe connection. A two-way solid line indicates a direct connection that does not go through an MQTT broker. Supervisors access sensor data monitoring and analysis services through the plant process management application and supervise control status. In order to store the sensor data transmitted by the IoT sensor device to fit the service data storage structure through the sensor monitoring service, the algorithm introduced in

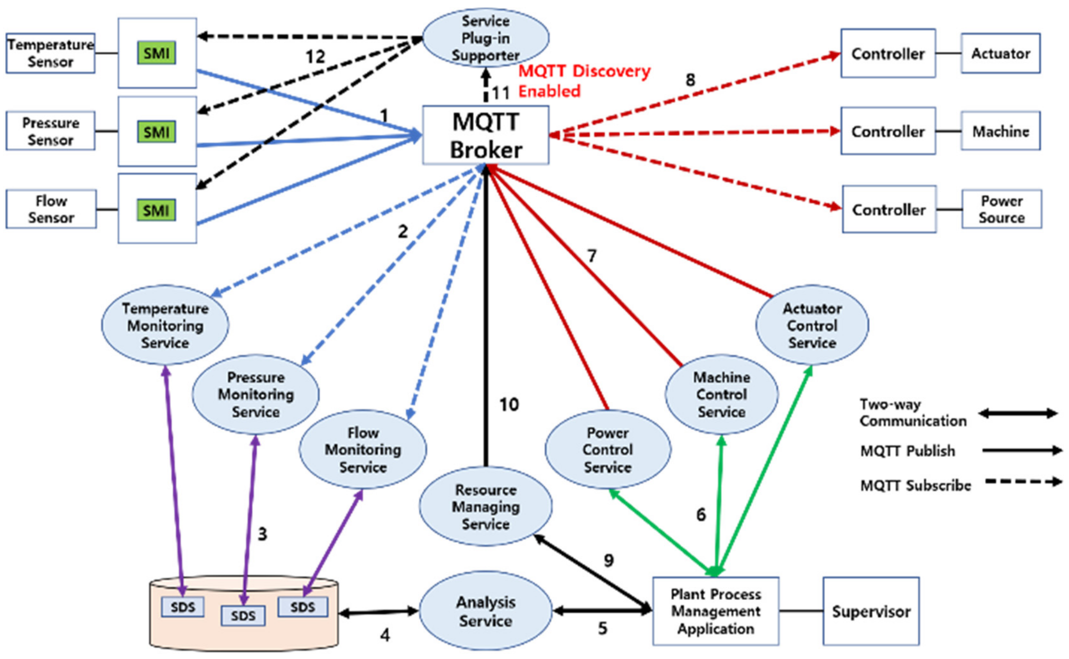


Fig. 7. Proposed system architecture for service plug-in support.

the next section is performed between the Service Plug-in Supporter and the IoT sensor device. Resource management service provides the schema of service data storage structure to Service Plug-in Supplier.

3.2. Proposed Algorithm

Sensors are the most important components for process control in a smart factory or process plant. This is because the accuracy and integrity of the sensor output data affects the overall control. In other words, the sensor data can have a big impact on the service implementation algorithm. In general, the DC current output in the 4–20 mA range of the industrial sensor is calculated as 0–100 % of the measurement range. Most industrial sensors such as temperature, pressure and flow sensors are identical. If the temperature monitoring service detects that the temperature sensor output is gradually lowering to 4 mA, the control service performs control to increase the temperature. At this time, if the input temperature sensor value was the actual pressure sensor value due to sensor management failure, the pressure rise and the temperature rise simultaneously. It can lead to accidents. Therefore, it is more important to identify the sensor device that is the sender of the sensor data than the value measured by the sensor.

The key contribution of our research work is to eliminate the risk of management failure in the service algorithm even if there is a management failure in the process of exchanging, adding, or removing sensor devices in the field.

The algorithm we propose includes the SMI code (see Fig. 6) containing the sensor's logical management information in the sensor's publish configuration file (configuration .json or configuration.yaml) in the process of performing the MQTT Discovery protocol. The algorithm of service implementation determines whether logical management information in the SMI of the existing sensor device matches the logical management information in the newly received SMI. If they match, it is determined whether the sensor type is the same as the existing sensor type in the physical device information of SMI again.

If they match, the sensor data transmission format matching the service data storage structure is completed through the service plug-in supporter in the system architecture of Fig. 7.

The following Fig. 8 is the proposed algorithm that completes the sensor data storage structure in order for the IoT sensor device to support the service plug-in.

IV. EXPERIMENT AND DISCUSSION

4.1. Experimental Conditions

The MAX6675 thermocouple module and K-type temperature sensor are configured as IoT sensor devices in Raspberry Pi 3B$^+$ device. And MQTT Broker and service were implemented in Laptop PC with Windows 10 OS installed. Laptop PC has Intel Pentium i7 CPU and 16GM RAM hardware. With reference to the open source of [21-23], the connection of IoT sensor devices, configuration and service of MQTT discovery environment were implemented in JSON and Python languages.

An XML sensor template was obtained with the TEDS editor of HBM [21], and a sensor PSI based on the Basic TEDS template was prepared. PSI of Sensor was changed

```
// Algorithm in Service Implementation
Input : Newly Received SMI of Sensor(i)
Output : Sensor Data Structure

// SMI : Sensor Management Information
// LMI : Logical Management Information
// PSI : Physical Sensor Information

if (
   Existing_Sensor(i).SMI.LMI ==
   Newly_Received_Sensor(i).SMI.LMI ) {

   if (
         Existing_Sensor(i).SMI.PSI ==
         Newly_Received_Sensor(i).SMI.PSI ) {

         existing_sensor_type =
            get_sensor_type(Existing_Sensor(i).SMI.PSI);

        newly_received_sensor_type =
         get_sensor_type(Newly_Received_Sensor(i).SMI.PSI);
         if ( existing_sensor_type ==
               newly_received_sensor_type) {
                     complete Sensor Data Structure;
        }
   }
}
```

Fig. 8. Proposed algorithm for service plug-in support.

to json format, and LMI of sensor in json format was written by hand. The PSI of sensor file and LMI of sensor file are combined into an SMI file. An MQTT application was prepared that stores SMI in the Raspberry Pi 3B+ device memory and connects the temperature sensor monitoring service in Fig. 8.

Algorithmic processing for SMI in a single sensor configuration resulted in an average delay of 120 ms. At this time, LMI and PSI expressing the management information of the sensor device are the same as the sensor device management information maintained by the service. That is, it is an experiment in a situation where there is no change in the physical sensor device. This delay is the time it takes the sensor device to save the sensor values to the sensor data store (a file on the laptop).

4.2. Cost for Performing the Algorithm

A case in which a raspberry Pi device has multiple same SMI configuration was tested. This configuration means that multiple sensors are connected to one IoT device interface. In this experimental condition, the size of each SMI data is 130 KBytes. An experiment is the same as transmitting an SMI data array. The algorithm processing delay in this experiment is shown in the Fig. 9 below.

4.3. Comparison of Experiments

In the second experiment, due to the replacement event of the sensor device, the LMI and PSI data of the sensor newly accessing the service do not match the data of the existing sensor device. The preceding experimental conditions in which there was no replacement event of the physical sensor were also tested. The following shows the difference between the two results. One is the same as the experimental conditions without the device replacement event, and the other is that the SMI data is input differently each time. This assumes that the different physical devices change every time.

In the Fig. 10, the gap between the delay in the two experiments is analyzed to reflect the delay that the sensor processes the transmission data formatting to suit the data storage schema of the service.

The results in Fig. 10 show that changing or adding a new physical sensor device is approximately three times higher latency than replacing it with the same physical sensor device.

V. CONCLUSION

In the application layer, services are designed to be flexible to meet market demands. So, there has been a lot of research to support the implementation of services that are independent of data representation. However, human effort is essential to organically managing connections between

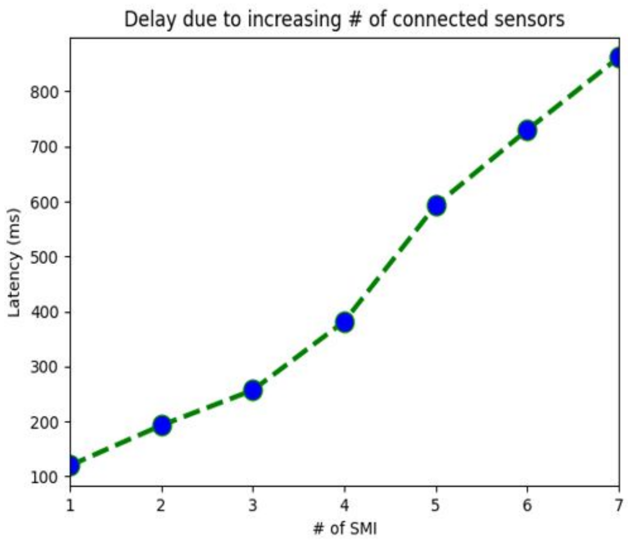


Fig. 9. Delay at same SMI input.

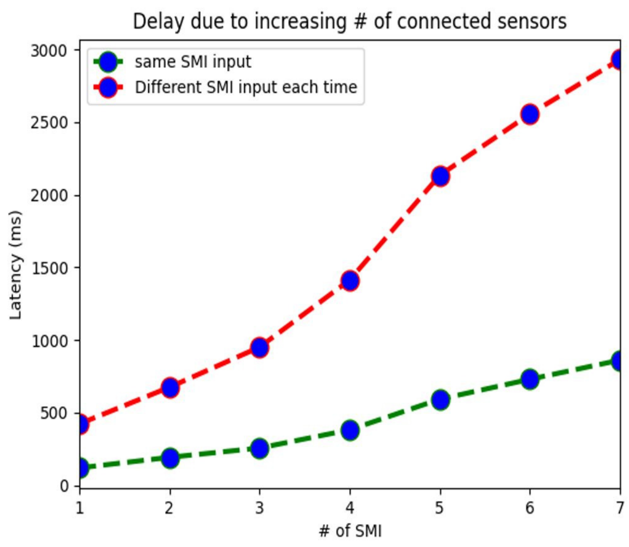


Fig. 10. Comparison of 1st and 2nd experiment results.

services, data, and devices. Many standards are available on the market to solve this problem. Nevertheless, management events of physical devices such as sensors require device management information processing that does not match the data storage structure of the service.

In this paper, we presented a method and system architecture that can support service transparency despite the occurrence of a management event of a physical sensor device in an MQTT-based IoT application environment.

The contribution of our work is to suggest a method for integrating logical management information and physical device information on sensor devices, and to suggest a system structure and algorithm for applying the MQTT discovery protocol to the integrated sensor device management information. Depending on the application, detailed management information may be requested for the sensor device. In future research, it is necessary to analyze the performance of the management requirement level of the device for practical service application.

REFERENCES

[1] IBM, SOA (Service-Oriented Architecture), Aug. 2022. https://www.ibm.com/cloud/learn/soa.

[2] IEEE Standard Association, IEEE 1451.4 Standard for Smart Transducers, https://standards.ieee.org/wpcontent/uploads/import/documents/tutorials/1451d4.pdf.

[3] IEEE Standard Association, An Overview of IEEE 1451.4 Transducer Electronic Data Sheets (TEDS), https://standards.ieee.org/wp-content/uploads/import/documents/tutorials/teds.pdf.

[4] NI, Using TEDS Sensors in Your Channel Specification, 2022. https://www.ni.com/docs/ko-KR/bundle/flexlogger/page/using-teds-sensors-in-chan nel-spec.html.

[5] mikroC, OneWire Libray, Aug. 2022. https://download.mikroe.com/documents/compilers/mikroc/pic/help/onewire_library.htm.

[6] V. Viegas, O. Postolache, and J. M. Dias Pereira, "Transducer electronic data sheets: Anywhere, anytime, anyway," *Electronics*, vol. 8, no. 11 p. 1345, 2009.

[7] NI, Using TEDS with NI-DAQmx in LabVIEW, Aug. 2022. https://knowledge.ni.com/Knowledge ArticleDetails?id=kA03q000000YIV0CAO&l=en-US.

[8] Oracle, Using IPMItool to View System Information, Jul. 2022. https://docs.oracle.com/cd/E19464-01/820-6850-11/IPMItool.html.

[9] github.com, ipmitool, Jun. 2022. https://github.com/ipmitool/ipmitool.

[10] Oracle, Sensor Data Repository, Jun. 2022. https://github.com/ipmitool/ipmitool.

[11] D. L. Hernández-Rojas, T. M. Fernández-Caramés, P. Fraga-Lamas, and C. J. Escudero, "A plug-and-play human-centered virtual TEDS architecture for the web of things," *Sensors,* vol. 18, no. 7, p. 2052, 2018.

[12] T. A. dos Santos Filho, A. C. Jr, Oliveira, D. M. Tavares, M. A. Batista, A. C. da Silva, and M. N. Rabelo, "Transducer module recognition in a network environment based on IEEE 1451," in *Proceedings of the International Conference on Wireless Networks (ICWN)*, 2015, p. 102.

[13] W. Luo, M. Bolic, J. Wang, and X. Qian, "Management of sensor-related data based on virtual TEDS in sensing RFID system," *International Journal of Distributed Sensor Networks*, 2015.

[14] H. Koziolek, A. Burger, M. Platenius-Mohr, J. Rückert, F. Mendoza, and R. Braun, "Automated industrial IoT-device integration using the OpenPnP reference architecture," *Software: Practice and Experience*, vol. 50, no. 3, pp. 246-274, 2020.

[15] W. Luo, M. Bolic, J. Wang, and X. Qian, "Management of sensor-related data based on virtual TEDS in sensing RFID system," *International Journal of Distributed Sensor Networks,* 2015.

[16] D. Gustin and J. Jasperneite, "IoT device management based on open source platforms - Requirements analysis and evaluation," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1-4.

[17] S. Attarha and A. Förster, "Service management for enabling self-awareness in low-power IoT edge devices," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2022, pp. 146-147.

[18] G. Kayas, M. Hossain, J. Payton, and S. R. Islam, "VSDM: A virtual service device management scheme

for UPnP-based IoT networks," in *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2020, pp. 426-433.

[19] IEEE Standard Association, IEEE 1451.4 Standard for Smart Transducers, https://stand ards.ieee.org/wp-content/uploads/import/documents/tu torials/teds.pdf.

[20] N. Jevtic and V. Drndarevic, "Plug and play geiger-muller detector for environmental monitoring," *Instrumentation Science & Technology*, vol. 43, no. 2, pp. 222-243, 2015.

[21] HBM,TEDS Editor, Jul. 2022. http:// www.hbm.com.

[22] Open Source: MQTT based Smart Sensor & Client | ABB, https://github.com/abb-motors-and-generators/smart-sensor-api-client.

[23] Steve, MQTT Application, http://www.steves-internet-guide.com.

[24] Roelof Jan, MQTT Discovery with a NodeMCU and Home Assistant, Jul. 2021. https://roelofjanelsinga.com/articles/mqtt-discovery-with-an-arduino/.

AUTHORS

Young-Ran Lee received the MS degree from the Busan University of Foreign Studies, South Korea, in 2002, and the Ph.D. degree from Department of Computer Information, Sun Moon University, in 2006. In March 2008. she joined the Division of IT education in Sun Moon University, South Korea, where she is currently an associate professor. She's research interests are Embedded System and Enterprise Blockchain IoT application.

Sung-Ki Kim received the BS, MS and Ph.D. degree in the Department of Computer Science and Engineering from Incheon National University (INU), South Korea, in 1996, 1998, and 2006, respectively. In March 2009, he joined the Division of IT education in Sun Moon University, South Korea, where he is currently an associate professor. His research interests are Distributed System Security and Enterprise Blockchain IoT application. From November 2010 to 2015, he served as an editorial board member of the Journal of Information Processing Systems. Since January 2018, he is also serving as an editorial board member of the Journal of Multimedia Information System.