Journal of Korea Multimedia Society Vol. 22, No. 5, May 2019(pp. 601-612)
https://doi.org/10.9717/kmms.2019.22.5.601

스마트 팩토리에서 원격 실시간 모니터링을 위한 게이트웨이 인터페이스 연동 API 설계 및 구현

전동철[†], 이병문[††], 황희정[†††]

Design and Implement of Smart Gateway Interface API for Real-time Monitoring in Smart Factory

Dong-cheol Jeon[†], Byung Mun Lee[††], Heejoung Hwang[†††]

ABSTRACT

As the 4th industrial revolution is accelerating, IT convergence application technologies are attracting attention in various fields. In the manufacturing industry, Smart Factory technology, which is blended with IT technology, has been developed to solve the problem casued by the decrease of the labor force, and a monitoring server is required to remotely control the equipment or to inquire about the operation status of the factory. In this paper, we designed and implemented RESTful API for data sharing between factory equipment and monitoring server in Smart Factory. In order to verify the designed API, a testbed was operated for an actual plastics manufacturing plant. As a result, it was confirmed that the testbed can be operated normally in actual operating environment.

Key words: Smart Factory, REST, API, Design, Web

1. 서 론

IT 기술의 발전과 4차 산업혁명의 가속화로 다양한 분야에서 IT 융복합 응용 기술들이 주목을 받고 있다[1]. 특히 제조업에서는 노동력이 감소하면서 IT 기술과 제조업을 융합시킨 기술들이 주목을 받고 있으며 IoT 기술과의 결합을 통해 스마트 팩토리로 발전해 나가고 있다[2]. 스마트 팩토리는 제품의 기획 및 설계, 생산, 유통 및 판매 등 전 과정을 정보통신기술과의 융합을 통해 최소 비용과 시간으로 고객 맞춤형 제품을 생산하는 미래형 공장을 뜻하는데, 인터넷으로 연결된 장비들이 스스로 정보를 획득하고 이를 활용하여 효율적인 생산 공정을 하는 시스템을 말한다[3]. 이러한 스마트 팩토리는 인력 낭비를 줄이고 사람으로부터 발생할 수 있는 변수를 통제할 수 있으며 효율성과 생산성을 높일 수 있다는 장점이 있다[4]. 특히 소프트웨어 기술과의 결합을 통해 공장 운영 중 발생하는 각종 상황, 문제, 생산율 등의 전반적인 데이터를 조회하거나 장비를 효과적으로 원격 통제할 수 있는 기술들이 요구되고 있으며 이를 위해서는 장비 제어를 위한 통합 인터페이스와 데이터 중계 서버 및 모니터링 소프트웨어 기술들이 필요하다.

본 논문에서는 첨단 기술의 확보가 어려운 중소규모의 공장 환경 중에서 플라스틱 제조 공장을 대상

※ Corresponding Author : Heejoung Hwang, Address: (13120) Seongnamdaero 1342, Seongnam-si, Gyeongi-do, Korea, TEL : +82-31-750-4758, FAX : +82-, E-mail : hwanghj@gachon.ac.kr
Receipt date : Apr. 15, 2019, Approval date : Apr. 29, 2019
[†] Dept. of IT Convergence Engineering, Gachon University (E-mail : aseah@naver.com)
[††] Dept. of Computer Engineering, Gachon University (E-mail : bmlee@gachon.ac.kr)
[†††] Dept. of Computer Engineering, Gachon University
※ This work was supported by the Technology development Program(Grants No. S2598497) funded by the Ministry of SMEs and Startups(MSS, Korea).
※ This work was supported by the Gachon University research fund of 2017.(GCU-2017-0606)

으로 장비들에 대한 정보를 통합관리하기 위한 스마트 게이트웨이와 원격 모니터링 시스템과의 정보 공유를 위한 REST 기반 API를 설계하고 구현하였다. 공장 장비에서 발생한 데이터는 네트워크 인터페이스를 통해 스마트 게이트웨이로 전달되고 게이트웨이는 센서를 통해서 얻은 다수의 정보들을 실시간 모니터링 웹 서버로 전달을 해준다. 서버는 이에 대한 결과 응답을 보내주거나 또는 서버에서 전달하는 제어 및 설정 값을 게이트웨이로 보내고, 게이트웨이가 해당 제어를 하는 제습 건조기로 요청을 보내주게 된다. 본 논문에서는 이러한 과정에서 필요한 제습건조기, 스마트 게이트웨이, 그리고 서버 간의 데이터 공유에 필요한 요구사항을 분석하고 공장 운영환경에 따른 동기/비동기 처리 모델의 조합과 실시간 데이터를 통합 대시보드로 전달하기 위한 아키텍처를 설계하고 실제 공장 환경에서 실험을 통해 실제 운용 가능함을 보였다.

2. 관련 연구

2.1 스마트 팩토리

스마트 팩토리는 제품의 기획, 설계, 생산, 유통, 판매 등의 전 과정을 IT 기술로 융합해서, 최소 비용 및 시간으로 고객 맞춤형 제품을 생산하는 미래형 공장 시스템을 뜻한다. 제조 분야에서는 전 세계적으로 제조업의 위기로 인한 경기 불황과 고용한계 문제를 극복하고 생산성과 효율성을 극대화하기 위한 방법을 찾으면서 주목을 받기 시작했다. 스마트 팩토리를 통해 산업 기기와 생산 전 과정이 네트워크로 연결되며, 나아가 고객의 요구사항에 대해 유연히 대처할 수 있는 체계를 구현할 수 있게 됐으며 생산 공정·인터페이스·운영의 최적화 기술, 다품종 복합생산, 조달 및 물류 혁신, 장비와 인간의 협업을 가능케 하여 고객 만족을 높이고 기업은 경쟁력을 높일 수 있는 발판이 되었다.

현재 국내에서는 제조업 시장 경쟁력 강화를 위해 기존 제조기술과 센서, 클라우드, 빅데이터 등 다양한 ICT 기술과의 융합을 통해 스마트 팩토리에 대한 국가적 전략을 추진하고 있으며, 대기업을 중심으로 제조현장을 혁신시키고자 일부 진행하고 있으나, 아직 초기 단계로, 해외에서 가져온 솔루션을 도입하는 경우가 많다. 국내에서 스마트 팩토리 요소 기술에 대한 수준은 통신/네트워크 기술이 제일 높지만, 이를 제외한 모든 부분에서 낮게 나타나고 있어 실시간 연계 운영이 가능한 핵심 요소 기술 분야(애플리케이션, 플랫폼, 디바이스/네트워크) 별로 통합 연계를 할 수 있는 솔루션 분야의 개발이 시급한 상황이다 Table 1은 2017년 한국과학기술기획평가원의 이슈위클리에서 발간한 보고서 자료이다[5].

해외 사례를 들어보면, 독일에서는 스마트 팩토리의 실현을 위하여 Industrie 4.0이라 일컫는 거대 프로젝트를 정부 주도로 진행하고 있고, 미국에서도 첨단 제조 능력 강화를 통한 제조업의 국가경쟁력 부활을 목표로 국가 차원의 스마트 팩토리 프로젝트를 수행하고 있다. 아시아에서는 중국은 '제조강국2025', 일본은 '산업재흥플랜' 등 다양한 스마트 팩토리 실현 전략을 내세우고 이를 수행하고 있다.

2.2 REST

REST란 Representational State Transfer의 약자로 자원의 이름으로 구분하여 해당 자원의 상태, 정보를 주고받는 모든 것을 의미한다. 이는 네트워크상에서 클라이언트와 서버 사이의 통신 방식 중 하나로 자원(Resource)의 표현(Representation)에 의해서 상태를 전달하는 것을 말한다[6]. 여기서 자원은 소프트웨어가 관리하는 모든 것을 의미하는데, 예를 들어 문서, 그림, 데이터 등이 있다. 상태 및 정보 전달은 데이터가 요청되어지는 시점에서 자원의 상태를

Table 1. Korea's level of smart factory component technology

	App	IoT/ M2M	CPS	Big Data	3D Printing	Cloud	Robot	Network	Sensor	Security	Standard
High	20.2	18.3	11.0	11.9	11.0	16.5	19.3	58.7	30.3	10.1	7.3
Middle	36.7	40.4	37.6	33.0	39.4	37.6	45.0	33.0	41.3	34.9	40.4
Low	43.1	41.3	51.4	55.0	49.5	45.9	35.8	8.3	28.4	55.0	52.3
Avg(5)	2.71	2.71	2.44	2.42	2.54	2.65	2.83	3.64	2.96	2.42	2.44

전달하는 것을 의미하는데, 여기서 JSON 혹은 XML을 통해서 데이터를 주고받는 것이 일반적이다. REST는 기본적으로 웹의 기존 기술과 HTTP 프로토콜을 그대로 활용하기 때문에 웹의 장점을 최대한 활용할 수 있다. REST는 HTTP URI를 통해서 자원을 명시하고 HTTP 메소드를 통해서 해당 자원에 대한 CRUD를 적용할 수 있다[7]. 즉, 자원 기반의 구조 설계의 중심에 자원이 있고 HTTP 메소드를 통해 자원을 처리하도록 설계된 아키텍처를 의미한다. REST는 HTTP 표준 프로토콜에 따르는 모든 플랫폼에서 사용할 수 있고, Hypermedia API의 기본을 충실히 지키면서 범용성을 보장할 수 있다는 장점이 있다. 그리고 HTTP 프로토콜의 인프라를 그대로 사용하기 때문에 별도의 인프라를 구축할 필요가 없어서 API 구축이 복잡하지가 않다.

REST API는 REST를 기반으로 서비스 API를 구현한 것이다. REST API를 사용하면 사내 시스템들도 REST 기반으로 시스템을 분산해 확장성과 재사용성을 높여 유지보수 및 운용을 편리하게 할 수 있다. 그리고 REST가 HTTP 표준으로 구현하기 때문에, HTTP를 지원하는 프로그램 언어로 클라이언트와 서버를 구현할 수 있다. REST API의 설계 기본 규칙에는 두 가지가 있는데, 첫 번째는 URI는 정보의 자원을 표현해야 하며, 두 번째는 자원에 대한 행위는 HTTP 메소드로 표현해야 한다[8]. HTTP의 기본 CRUD 메소드는 Table 2과 같다.

RESTful이란 일반적으로 REST라는 아키텍처를 구현하는 웹 서비스를 나타내기 위해 사용되는 용어로 REST API를 제공하는 웹 서비스를 'RESTful'하다고 한다. 즉, REST 원리를 따르는 시스템을 RESTful이라는 용어로 지칭한다. RESTful 웹 서비스는 개발자들이 다양한 표현이 가능한 자원을 HTTP 기본 메소드만으로 제공받기 위하여 만들어졌다[10]. RESTful의 목적은 이해하는 것과 사용하는 것이 쉬운 REST API를 만드는 것이다. 그리고 성능보다는 일관적인 컨벤션을 통한 API의 이해도 및 호환성을 높이는 것이 주된 목적이다.

Table 2. HTTP's CRUD methods[9]

Method	CRUD	Description
GET	Read	Read resource's expression.
POST	Create	Create new resource.
PUT	Update	Create resource or update existing resource.
DELETE	Delete	Delete existing resource.

3. 실시간 모니터링 API 설계

기존 플라스틱 제조 공장에서 제습 및 건조 과정은 사람이 직접 기기를 가동시키면서 상태를 확인하고 장애가 발생하면 인터페이스 모듈을 통해 정지시킨 다음에 점검하고 재가동 시키는 과정이었다. 그리고 시리얼 통신으로 연결된 기기를 통해 가공하는 원료에 맞게 제습 및 건조 온도에 대한 설정 및 기타 환경 설정 값을 입력해야 했다. 모든 과정이 사람에 의해 관리되기 때문에 장애 발생에 대해서 즉각적으로 대응하는 것에 어려움이 있었고, 정확한 장애 탐지 및 처리가 어렵다는 단점이 있었다. 이를 극복하기 위해 스스로 가동하는 과정에서 실시간으로 제습건조기에 대한 정보를 파악하면서 장애 위험을 탐지하여 사전에 처리하거나 장애가 발생했을 때 즉각적으로 대응할 수 있는 기술이 필요하게 되었다.

3.1 API 스키마 설계

본 연구의 대상 환경인 플라스틱 제조공장의 경우 제습건조기가 주요 장비이며, 많으면 100대까지 한 공장에서 운용될 수 있는데, 동시에 다수의 제습건조기에서 발생하는 데이터를 공유할 경우 공장 운영 환경의 특징으로 인한 동기방식 요청을 효과적으로 처리하기 어렵고 데이터 병목 현상이 발생할 수 있기 때문에 여러 대의 게이트웨이를 두고 제습건조기의 정보를 모아서 전달하거나 서버에서 발생한 데이터를 제습건조기로 전달할 수 있어야 한다. 이러한 요구 사항을 처리하기 위해 Table 3와 같이 API를 설계했다. 설계한 API는 제습건조기에 있는 센서와 직접적인 통신을 하는 것이 아닌 게이트웨이를 통해서 통신을 하는 과정에서 사용되는 API의 설계이다.

요구사항에 대해 설계된 Request는 모두 4가지 유형이며 다음과 같은 역할을 수행한다.

가. Register : 게이트웨이와 제습건조기를 등록하는 과정에서 사용되는 API

나. Management : 장비를 운영하는 과정에서 상태, 제어 정보를 저장하는 API

다. Data : 제습건조기 운영 과정에서 발생되는 계

Table 3. List of RESTful API

Request	URI	Description
Register	/gwsvc/reg/gw	Register gateway
	/gwsvc/reg/dry	Register drying machine
	/gwsvc/reg/confirm	Confirm connection between gateway and drying machine
	/gwsvc/drysvc/error	Send gateway's IP if drying machine can't connect with gateway
Management	/gwsvc/save/metaData	Save meta-data
Data	/gwsvc/save/measureData	Save real-time measurement data
Log	/gwsvc/save/eventLog	Save event log

측 데이터를 저장하는 API

라. Log : 장비를 운영하면서 발생하는 장애 데이터를 저장하는 API

3.2 API 프로세스 설계

3.2.1 등록 프로세스

장비에서 발생한 데이터를 받기 위해서 먼저 서버에 장비 정보를 등록해야한다. 등록 요청을 보내기 전 서버에 해당 장비에 대한 고유ID가 데이터베이스에 저장이 되어 있어야 하며, 게이트웨이를 먼저 등록해야 제습건조기가 게이트웨이와 연결할 수 있기 때문에 게이트웨이를 먼저 등록해야 한다. 등록 요청을 통해 장비가 등록 되면, 게이트웨이와 제습건조기는 서로 연결을 확인하기 위해 서버에 해당 게이트웨이의 정보와 연결된 제습건조기의 정보를 보내게 되고 정상적으로 연결이 되면 서버는 성공적으로 연결이 되었다는 응답을 보낸다. 제습건조기와 게이트웨이 간의 연결이 제대로 되지 않았을 경우 에러 메시지를 응답 코드로 보내게 된다. Fig. 1은 게이트웨이와 제습건조기의 연결에 대한 시퀀스 다이어그램이다.

3.2.2 메타 데이터 전송 프로세스

장비가 서버에 등록 되고 제습건조기와 게이트웨

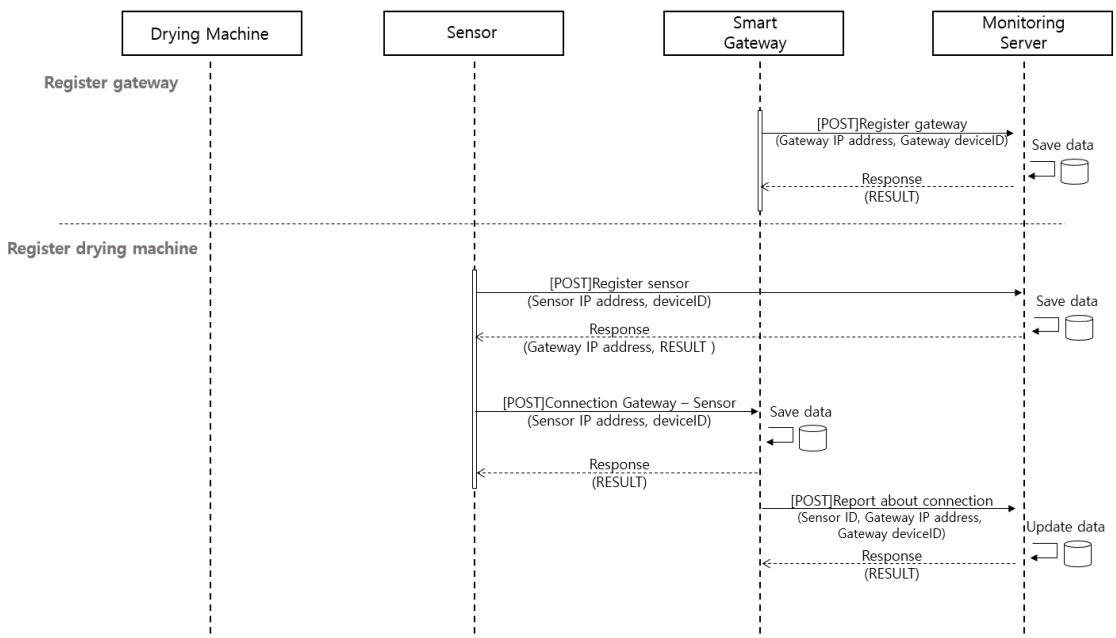

Fig. 1. Sequence diagram of registering gateway and drying machine.

이 간의 연결이 완료되면, 서버는 장비에 대한 상태 정보와 관리자에 의해 설정된 제습건조기 설정 정보를 메타 데이터로 전달 받는다. 장비에 대한 상태 정보는 게이트웨이와 제습건조기에 대한 운영 상태를 말하는데, 장비가 가동을 시작하거나 종료하면 메타 데이터로 서버에 전달하고 이를 받은 서버는 장비 상태 정보를 변경한다. 또한, 관리자가 장비를 직접 제어하면 서버에서도 제어 정보를 변경할 필요가 있는데, 메타 데이터 API를 통해 변경한 설정 값이 서버로 전달되고, 서버는 변경된 설정 값을 저장한다. Fig. 2는 메타 데이터 전송에 대한 시퀀스 다이어그램이다.

3.2.3 계측 데이터 전송 프로세스

연결이 완료 되면 제습건조기는 게이트웨이를 통해 가동 여부를 전송하고, 정상적으로 가동이 되었다는 응답 메시지를 받으면 게이트웨이는 제습건조기에서 측정된 계측데이터를 받아 2초 단위로 서버로 보내게 된다. 장비를 운영하면서 장애 상황이 발생하여 계측 데이터를 서버로 전송하지 못하게 되면 게이트웨이는 누적된 계측 데이터를 가지고 있다가 다시 정상적으로 작동이 되면 서버에 한꺼번에 전송하고 서버는 이를 한 번에 처리하게 된다. Fig. 3은 제습건조기에서 계측된 데이터를 전송하는 시퀀스 다이어그램이다.

3.2.4 장애 데이터 전송 프로세스

만약 제습건조기에서 설정 값의 범위에 벗어난 데이터가 측정 되거나 제습건조기 혹은 게이트웨이에 이상이 발생하면 Event Log라는 장애 데이터가 서버에 전달된다. 장애 데이터를 보내면 제습건조기는 자동으로 가동을 중지하고 장애 내용을 저장한다. 추후 장애 처리가 완료되면 제습건조기는 해당 장애에 대해 처리가 완료 되었다는 메시지를 보내게 된다. 장애 데이터 전송에 대한 시퀀스 다이어그램은 Fig. 4와 같다.

3.3 API 메시지 설계

3.1에서 정의한 요구사항을 바탕으로 API 메시지

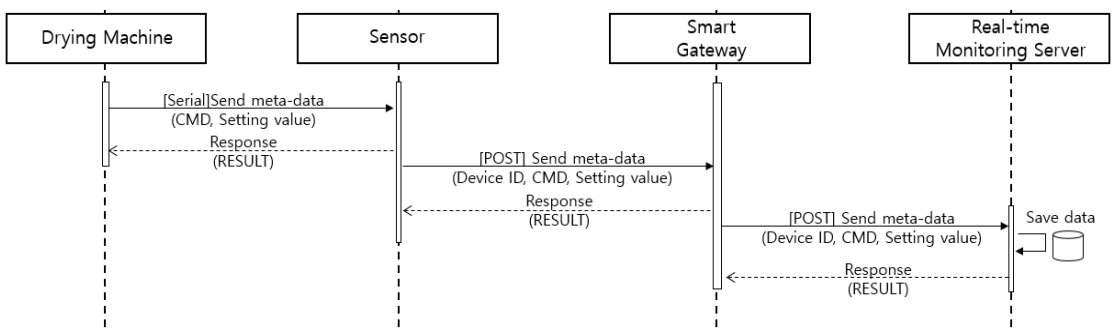

Fig. 2. Sequence diagram of sending meta-data.

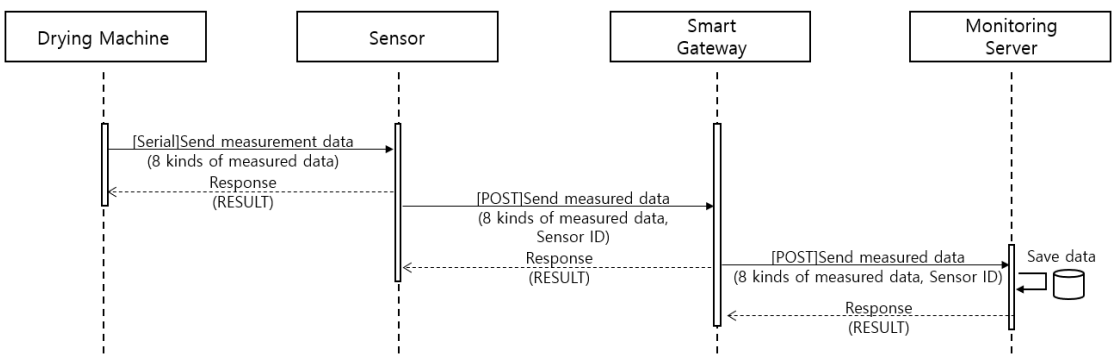

Fig. 3. Sequence diagram of sending measurement data and saving.

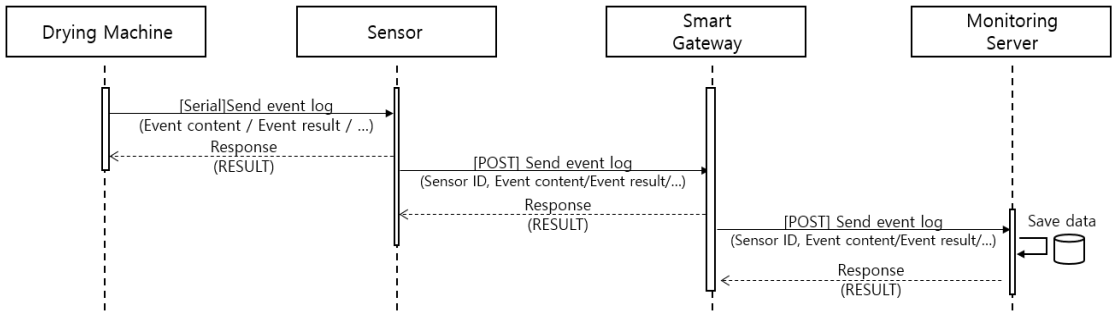

Fig. 4. Sequence diagram of sending event log and saving.

구조를 설계 했다. Fig. 5는 전체 API 메시지 설계에 대한 기본적인 구조이다.

설계한 API는 스마트 게이트웨이를 통해서 서버로 전달되며, IN은 요청할 때 보내는 파라미터이며, OUT은 받은 요청에 대한 처리 결과를 보낼 때 포함하는 파라미터이다. 파라미터에 대한 설명은 다음과 같다.

가. deviceID : 요청을 보내는 장비 ID를 담는 파라미터이다. 모든 API 메시지에는 장비 ID 파라미터가 반드시 포함되어 있어야한다.

나. data : API 유형에 따라 보내는 데이터이다. 데이터는 종류에 따라 파라미터가 나뉜다.

다. timestamp : API를 요청한 시간을 담는 파라미터이다.

라. status : 요청한 API에 대한 처리 결과 코드를 담는 파라미터이다. status가 200이면 정상적으로 처리되었음을 의미하고, 400~500대는 API를 처리하는데 문제가 발생했음을 뜻한다.

마. message : 요청에 대한 처리한 결과 메시지를 담는 파라미터이다.

바. path : 요청한 API URL을 담는 파라미터이다.

3.2.1 등록 메시지

장비를 등록하는 과정에서 사용되는 API는 '/gwsvc/reg/gw', '/gwsvc/reg/dry', /gwsvc/reg/confirm' 총 3개로 이루어져있다. 서버에 장비를 먼저 등록해야지 서버는 장비에서 발생한 데이터를 받을 수 있다. 등록 API는 해당 장비의 ID이외에 IP 주소를 파라미터에 담아 서버로 전달한다. Table 4는 각 등록 API에 대한 설명이다.

3.2.2 메타 데이터 전송 메시지

메타 데이터 전송 API URI는 '/gwsvc/save/metaData'이다. 메타 데이터는 장비의 상태 정보 전송 혹은 관리자가 직접 설정한 세팅 값에 대한 정보를 담는 API이다. 메타 데이터가 담는 data는 cmd와

POST /gwsvc/*

Smart Gateway → Send data → Monitoring Server

Smart Gateway ← Send result ← Monitoring Server

Parameter	Way		Type	Description
	IN	OUT		
deviceID	○		String	Request Device ID
data	○		String	Request data by API
timestamp		○	String	Server response date
status		○	String	Result code
message		○	String	Result message
path		○	String	Path

Fig. 5. Base structure of API message.

Table 4. Description of API of registering device

API URL	Parameter	Description
/gwsvc/reg/gw	gwIp	Register gateway to server
/gwsvc/reg/dry	dryIp	Register drying machine to server
/gwsvc/reg/confirm	gwId	Confirm connection between gateway and drying machine.
	gwIp	

code로 이루어져있는데, cmd는 메타 데이터 중에서 어떤 유형의 메타 데이터를 전송하는지를 담는 파라미터이며, code는 해당 유형에 대한 실제 값을 담는 파라미터이다. 메타 데이터의 유형은 Table 5와 같다.

3.2.3 계측 데이터 전송 메시지

계측 데이터 전송 URI는 '/gwsvc/save/measureData'이며 제습건조기에서 측정된 원료 가공 상태 정보를 전송하는 API이다. 제습건조기에서 연결된 게이트웨이로 2초 단위로 전송하며, 게이트웨이는 해당 계측 데이터 정보를 받아서 서버로 전달해준다. 계측 데이터 전송 API의 파라미터는 Table 6으로 구성이 되어있다.

3.2.4 장애 데이터 전송 메시지

장비를 운영하는 과정에서 제습건조기에서 측정된 원료 상태가 설정한 한계치를 벗어나거나 장비 자체에 문제가 발생하면 데이터 전송 메시지 API인 '/gwsvc/save/eventLog'를 통해 서버로 요청한다. 장애 데이터 전송 API의 파라미터는 구조상 3.2.2 메타 데이터 전송 API의 파라미터와 동일한 cmd와 code로 이루어져있다. cmd는 발생한 장애의 유형을 담는 파라미터이며, code는 장애 처리 여부를 담는 파라미터이다.

4. 구현 및 실험

본 논문은 중소기업벤처부의 2019 산업협력기술개발사업의 연구결과물로 주관기관인 ㈜아성프랜트의 플라스틱 사출공장에서 일반적인 규모의 공장 환경을 고려하여 1주일 동안 하루 6시간씩 50대의 제습건조기와 10대의 게이트웨이를 토대로 실제 환경에서 테스트베드를 구축하고 실험을 진행하였다. 전체 실험환경 구성은 Fig. 6과 같다.

Table 5. Type of meta data

CMD	CODE	Description
dryingmachine	01	Start running drying machine
	02	Stop running drying machine
dryingTemp	setting value	Setting limit of drying temperature
regenTemp	setting value	Setting limit of regeneration temperature
moistureRatio	setting value	Setting limit of moisture ratio
gateway	01	Start running gateway
	02	Stop running gateway

Table 6. Parameter of measured data API

API	Parameter	Description
/gwsvc/save/measureData	date	Generation date of measured data
	dryingTemp	Drying temperature of material
	regenTemp	Regeneration temperature of material
	moistureRatio	Moisture ratio of material
	dewPoint	Dew point of material

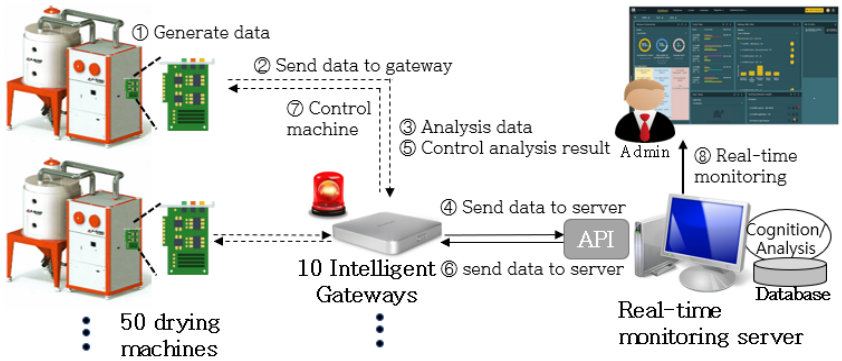

Fig 6. System configuration.

```
            ...
<dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <scope>provided</scope>
</dependency>
<dependency>
      <groupId>org.mariadb.jdbc</groupId>
      <artifactId>mariadb-java-client</artifactId>
</dependency>
<dependency>
      <groupId>commons-io</groupId>
      <artifactId>commons-io</artifactId>
      <version>2.5</version>
</dependency>
<dependency>
      <groupId>com.googlecode.json-simple</groupId>
      <artifactId>json-simple</artifactId>
      <version>1.1.1</version>
</dependency>
            ...
```

Fig. 7. Maven dependencies of API.

API의 구현은 Spring Boot 2.x와 JPA를 이용해 진행했고 데이터베이스는 MariaDB를 사용했다. Fig. 7은 API를 구현 프로젝트에 사용한 Maven build dependency설정의 일부이다.

4.1 API 구현

Fig. 8은 전체 API에 대한 클래스 다이어그램이다. 전체 API는 Controller와 Entity를 중심으로 공장에서 발생하는 메시지를 처리하기 위한 클래스들로 구성되어 있으며 본 논문의 대상은 아니지만 관리자 인터페이스 구현을 위한 MVC Web Controller를 포함하고 있다.

또한 API 보안을 위해 등록된 게이트웨이마다 OAuth2 인증을 통해 등록된 JWT(Jason Web Token)를 키로 사용하여 API 요청에 대한 인증을 처리해 발생할 수 있는 비인가 API요청을 차단할 수 있다. 이를 위해 Spring Security를 이용한 사용자 인증 및 권한관리, API 사용인증을 처리하도록 클래스를 설계 하였다. Fig. 9는 이에 대한 클래스 다이어그램이다.

데이터베이스 연동은 JPA(Java Persistance API)를 사용하였으며 Spring framework에서 기본적으로 사용하는 Hibernate를 이용해 구현 하였다. 일부

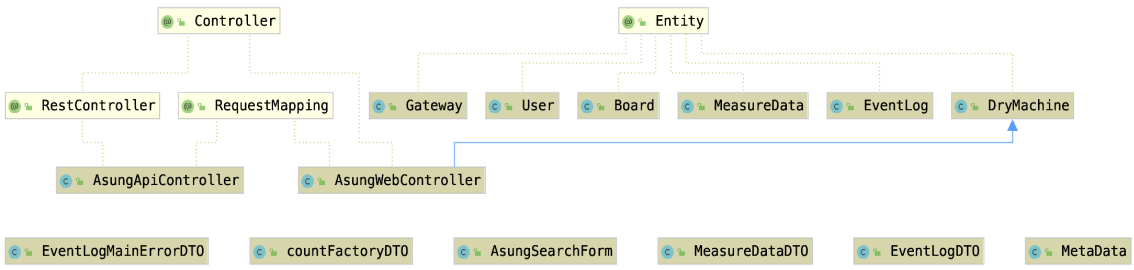

Fig. 8. Class diagram of API.

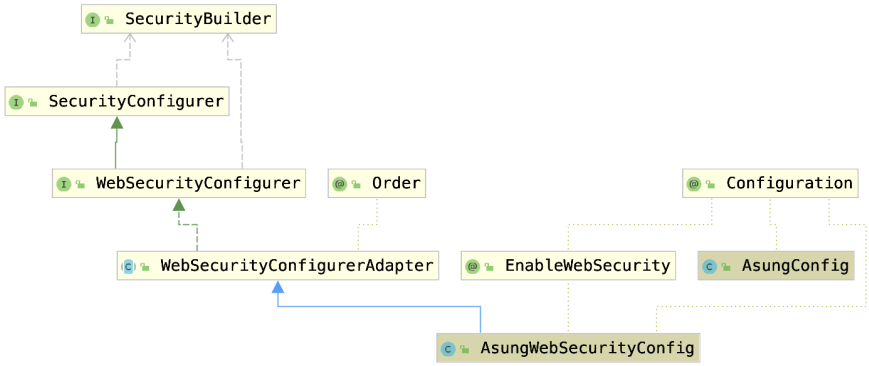

Fig. 9. Class diagram of API security.

요청은 표준 JPA 인터페이스로 구현되지 않는 데이터 요청이 있기 때문에 이 경우 별도의 DTO(Data Transfer Object) 클래스를 두어 JPQL을 직접 작성해 문제를 해결 하였다. Fig. 10은 데이터베이스 연동에 대한 클래스 다이어그램이다.

설계된 클래스 아키텍처에 따라 프로그램을 구현하였으며 Fig. 11은 구현 API 중 스마트 게이트웨이 등록에 관한 소스 코드 중 일부이다.

4.2 API 단위 테스트

요구사항에 따라 설계된 API 검증을 위해 실제 환경에서 정상적으로 동작하는지 확인하기 위해 API별 단위 테스트를 진행하였다. 테스트는 IntelliJ에서 제공되는 HTTP 테스트 도구를 사용해 진행하였으며, Fig. 12는 '/gwsvc/save/eventLog', '/gwsvc/save/measureData', '/gwsvc/save/metaData' API에 대한 단위 테스트 HTTP와 결과 코드이다. 단위 테스트를 한 결과 API가 정상적으로 동작이 되는 것을 확인하였다.

일반적인 규모의 공장 환경과 공장이 24시간 운영되는 것이 아닌 일정 시간 동안만 가동이 된다는 점을 고려하여 ㈜아성프랜트에서 1주일 동안 하루 6시간씩 테스트베드를 운영했다. 실험에 사용된 API는 7종류이며, 총 누적 API 요청 건수는 3,703,500 건이었다. 평균 정상 처리율은 약 99.61%였으며, 발생한 데이터의 유형은 Table 4와 같다.

7종의 API 중에서 '/gwsvc/save/measureData' API는 50대의 제습건조기에서 2초 단위로 한 번씩 계측 데이터 전송 요청이 왔기 때문에 제일 많았다. 제습건조기와 게이트웨이를 등록하는 과정에서 사용되는 '/gwsvc/reg/*' API는 장비 한 대당 하루 평균 한 번씩 등록 요청을 보냈다. 이는 장비의 IP 주소가 DHCP를 사용하여 할당되는데 장비를 리부팅하게 되면 등록 API를 통해 서버에 갱신된 IP주소를 보내서 리부팅한 장비를 재등록했기 때문이다. '/gwsvc/save/eventLog' API는 제습건조기에서 측정한 데이

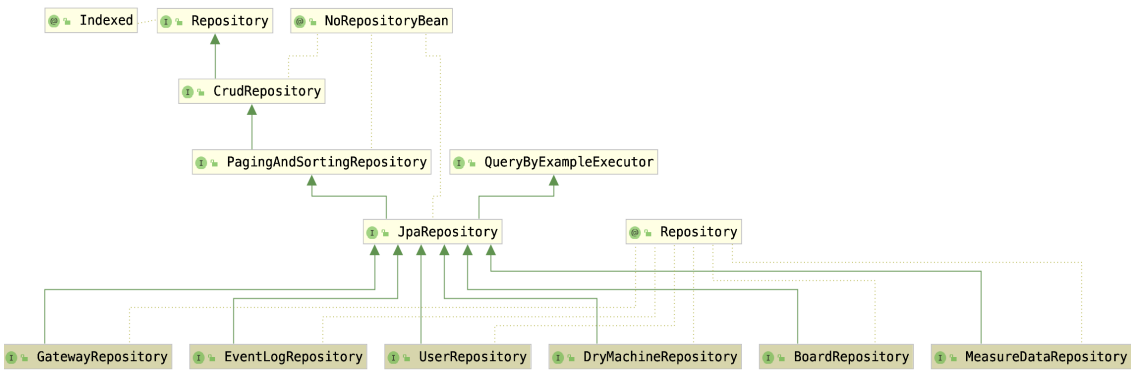

Fig. 10. Class diagram of database connection.

```
public ResponseEntity<ApiErrorMsg> registGateway(@RequestBody JSONObject body,
HttpServletRequest request) {
    Optional<Gateway> gwo = gwRepo.findById(body.get("gwId").toString());
    // Error11: There's no Gateway requested about gwId
    if(!gwo.isPresent()) {
      //If gateway that has gwId dosen't exist, save log to EventLog.
      ...
    }
    Gateway gw = gwo.orElseThrow(() ->
            new DataHandleException("There's no Gateway")
    );
    ...
    ApiErrorMsg msg;
    // Error2: Having problem during registering gateway IP
    if(ret == null) {
        throw new DataHandleException("Gateway Registration Error");
    }
    else {
        //To set the response message's body
        msg = new ApiErrorMsg.Builder()
                .message("Success to register gateway")
                .status(HttpStatus.OK.value())
                .path(request.getRequestURI()).build();
    }
    //Information about HTTP response header
    return new ResponseEntity(msg, HttpStatus.OK);
}
```

Fig. 11. Source code about registering gateway.

```
## Save event log
POST /gwsvc/save/eventLog HTTP/1.1
Host: 192.9.45.201:8080
Content-Type: application/json
{
 "deviceId":"DRY-001",
 "date":"2019-03-16T15:18:19.01",
 "cmd":"moistureRatio",
 "code":"02"
}

## Result of API test saving event log
{
 "timestamp": "2019-03-19T20:00:01.425",
 "status": 200,
 "message": "Success to save event log",
 "path": "/gwsvc/reg/dry"
}
```

```
## Save measured data
POST /gwsvc/save/measureData HTTP/1.1
Host: 192.9.45.201:8080
Content-Type: application/json
{
 "dryId":"DRY-001",
 "date":"2019-03-15T21:21:00.46",
 "dryingTemp":"200.0",
 "regenTemp":"190.0",
 "moistureRatio":"0.03",
 "dewPoint":"200.0"
}

## Result of API test saving measured data
{
 "timestamp": "2019-03-19T19:56:59.863",
 "status": 200,
 "message": "Success to save measure data",
 "path": "/gwsvc/drysvc/error"
}
```

```
## Save meta data
POST /gwsvc/save/metaData HTTP/1.1
Host: 192.9.45.201:8080
Content-Type: application/json
{
 "deviceId":"DRY-001",
 "cmd":"regenTemp",
 "data":"190.0"
}

## Result of API test saving meta data
{
 "timestamp": "2019-03-19T19:54:28.195",
 "status": 200,
 "message": "Success to save meta data",
 "path": "/gwsvc/save/metaData"
}
```

Fig. 12. API unit tests and results.

터가 세팅 값을 벗어나거나 장비에 장애가 생겼을 때 요청하는데, 398건 중 3건이 장비에 장애 상황이 발생하여 요청 되었으며, 나머지 395건은 세팅 값에서 벗어나면서 요청되었다. '/gwsvc/save/metaData'는 장비의 작동 정보 및 관리자가 직접 장비를 제어할 때 발생하는 API이며, 로그 데이터를 분석한 결과, 2,120건의 요청 중에서 1,128건이 장비의 작동 정보를 전송하는 요청이었으며, 나머지는 장비 제어에 대한 API 요청이었다.

실제 공장에서 테스트를 진행하면서 요청된 API

Table 7. Result of API test in factory

API URL	Total	Success	Error	Rate
/gwsvc/reg/gw	70	70	0	100%
/gwsvc/reg/dry	353	350	3	99.1%
/gwsvc/reg/confirm	353	350	3	99.1%
/gwsvc/drysvc/error	3	3	0	100%
/gwsvc/save/measureData	3,700,193	3,700,193	0	100%
/gwsvc/save/eventLog	400	398	2	99.5%
/gwsvc/save/metaData	2,129	2,120	9	99.6%

가 정상적으로 처리되지 못하는 상황이 발생되었다. 원인을 분석한 결과, 제습건조기를 교체를 하고 등록하는 과정에서 IP주소를 잘못된 값으로 전송해 연결이 정상적으로 이루어지지 않았다. 이로 인해 제습건조기에서 API를 통해 메타 데이터 및 이벤트 로그를 전송할 때 등록한 장비를 찾지 못해 API에 대한 요청이 처리되지 않았다. 이는 설계한 API에 문제가 있어서 발생한 장애 상황이 아닌 테스트하는 과정에서 외부적인 요인으로 인해 발생한 장애 상황이다. 그 외 다른 API 요청은 성공적으로 처리했으며, 구현한 API가 실제 환경에서 사용되는데 문제가 없음을 확인했다.

5. 결 론

본 논문에서는 본격적인 첨단 시설을 확보하기 어려운 중소규모의 플라스틱 제조 공장을 대상으로 최소한의 노력으로 스마트 팩토리를 구현하는데 필수적인 통합 모니터링과 원격제어를 위한 API시스템을 설계하고 구현 하였다. 본 연구는 중소기업벤처부 2019 산학협력기술개발사업의 성과물로 경기도 시화공단에 위치한 ㈜아성프랜트 본사 공장에서 실제 운영 장비들을 대상으로 시험환경을 구축해 테스트베드를 운영하였으며 실험결과 현업에서 요구되는 기능들이 모두 정상 구현되었음을 보였으며 실제 운영환경에서 발생하는 다양한 문제에도 대응할 수 있음을 확인할 수 있었다.

본 연구를 통해 사람이 직접 공장을 제어하고 관리하는 기존 시스템에서 기기가 자동으로 현재 상황을 모니터링하고 실시간 문제 대응 및 원격제어를 통해 효율적인 공장운영이 가능함을 확인할 수 있었다. 향후 연구 계획으로는 본 논문에서 설계된 API를 바탕으로 스마트 팩토리 관리 및 운영을 위한 통합 대시보드를 구현하는 것이며 이를 통해 설계된 시스템의 상용화가 가능할 것으로 기대 한다.

REFERENCE

[1] KIAT, *Research Trend of Smart Manufacturing in U.S.A*, KIAT Industrial Technology Policy Brief, 2014-21, 2014.

[2] KOSF, *Industrial Reference Models for Smart Factory Propagation*, KOSF Report, v.2.0, 2015.

[3] S. Shin, J. Woo, and W. Seo, "Developing a Big Data Analytics Platform Architecture for Smart Factory," *Journal of Korea Multimedia Society*, Vol. 19, No. 8, pp. 1516-1529, 2016.

[4] L. Li, W. Chou, W. Zhou, and M. Luo, "Design Patterns and Extensibility of REST API for Networking Applications," *IEEE Transactions on Network and Service Management*, Vol. 13, No. 1, pp. 154-167, 2016.

[5] KISTEP, *R&D Status and Implications of Smart Factory for the 4th Industrial Revolution*, KISTEP Issue Weekly Issue Paper 07, 2017.

[6] R. Fielding, *Architectural Styles and The Design of Network-based Software Architectures*, Doctoral Thesis, University of California, 2000.

[7] S. Fan, Y. Cha, and C. Kim, "Definition of RESTful Web Services for CCTV Sites, and Performance Anaylsis and Implementation of

Integrated Management System," *Journal of Korean Institute of Information Technology*, Vol. 13, No. 1, pp. 79-86, 2015.

[8] S. Vinoski, "RESTful Web Services Development Checklist," *IEEE Internet Computing*, Vol. 12, No. 6, pp. 96-95, 2008.

[9] Y. Lee, "Resource Matchmaking for RESTful Web Services," *Journal of Korean Institute of Information Technology*, Vol. 11, No. 8, pp. 135-143, 2013.

[10] Y. Park, A. Moon, H. Yoo, Y. Jung, and S. Kim, "SOAP-based Web Servies vs. RESTful Web Services," *Electronics and Telecommunications Trends*, Vol. 25, No. 2, pp. 112-120, 2010.

전 동 철

2019년 가천대학교 컴퓨터공학과 (공학사)
2019년~현재 가천대학교 일반대학원 IT융합공학과 컴퓨터공학 석사과정
관심분야 : Database, u-Health, Mobile, Web

이 병 문

1988년 동국대 전자계산학과 (공학사)
1990년 서강대 전자계산학과 (공학석사)
2007년 인천대 컴퓨터공학과 (공학박사)
1990년~1997년 LG전자(구 LG정보통신) 중앙 연구소 네트워크 연구실 선임연구원
1998년~현재 가천대학교 IT대학 컴퓨터공학과 교수
관심분야 : 사물인터넷, 유헬스, 센서네트워크, 운영체제, 헬스케어

황 희 정

2000년 인하대학교 컴퓨터공학과 (공학석사)
2008년 인천대학교 컴퓨터공학과 (공학박사)
2000년~현재 가천대학교 IT대학 컴퓨터공학과
관심분야 : Software Engineering, u-Health, Big Data, Medical Informatics, Ubiquitous Computing