https://doi.org/10.7840/kics.2024.49.3.453

Web of Things 표준 프로토콜 기반 고가용 복합 환경센서 장치 개발

권 동 우◆, 신 슬 비*, 김 재 현*, 지 영 민°

Development of Web-of-Things Standard Protocol-Based Complex Environmental Sensor Device with High Availability Support

Dongwoo Kwon◆, Seulbi Shin*, Jaehyeon Kim*, Youngmin Ji°

요 약

본 논문에서는 효과적이고 안정적인 실내 환경 관제 및 실시간 데이터 수집을 위해 Web of Things (WoT) 표준 프로토콜 기반 복합 환경센서 장치를 개발한다. 제안하는 환경센서 장치는 사람이 실내 공간에서 느끼는 열, 시각, 청각, 공기질 쾌적도를 산출하기 위해 온도, 습도, 조도, 소음, 이산화탄소, 미세먼지, 폼알데하이드 등의 복합센서를 탑재하고 있으며, 다양한 응용/분석 서비스와 연동하기 위한 WoT 표준 프로토콜을 적용하여 개발한다. 또한 장시간 데이터 수집의 안정성을 확보하기 위해서 CPU 외에 MCU를 탑재하여 하드웨어 및 네트워크 상태를 감시한다. 소프트웨어 측면에서는 마이크로서비스 아키텍처를 채용하여 부하분산 및 고가용성을 제공한다. 또한 클라우드 관리 플랫폼을 개발하여 다양한 장소에 설치된 다수 장치들의 관리 용이성을 향상시킨다.

Key Words : Internet of Things (IoT), Web of Things (WoT), environmental sensor, microservice

ABSTRACT

In this paper, we develop a complex environmental sensor device based on WoT (Web of Things) standard protocol for effective and stable indoor environment control and real-time data collection. The environmental sensor is equipped with complex sensors such as temperature, humidity, illumination, sound, carbon dioxide, particulate matter, and formaldehyde to calculate the thermal, visual, auditory, and air quality comfort that people feel in indoor spaces, and is developed by applying the WoT standard protocol for linking with various application/analysis services. Additionally, to ensure the stability of long-term data collection, an MCU is installed in addition to the CPU to monitor hardware and network status. On the software side, it adopts a microservice architecture to provide load balancing and high availability. Additionally, a cloud management platform have been developed to improve the manageability of multiple devices installed in various locations.

※ 본 연구는 산업통상자원부(MOTIE)와 한국에너지기술평가원(KETEP)의 지원을 받아 수행한 연구 과제입니다. (No. 20212020800120)
◆ First Author : Korea Electronics Technology Institute, dwkwon@keti.re.kr, 정회원
° Corresponding Author : Korea Electronics Technology Institute, ym.ji@keti.re.kr, 정회원
* Korea Electronics Technology Institute, seulbi0108@keti.re.kr, 정회원; jh.kim@keti.re.kr
논문번호 : 202311-129-C-RU, Received November 3, 2023; Revised November 29, 2023; Accepted December 1, 2023

Ⅰ. 서 론

최근 기후 변화 위기에 따른 국내외 탄소중립 문제와 미세먼지 등 건강한 생활 및 근무 환경에 관한 관심이 높아지면서 실내 공간을 점유하고 있는 재실자의 쾌적도와 에너지 절감 사이의 균형 있는 제어에 관한 연구개발[1]이 활발히 수행되고 있다. 실내 쾌적도를 유지하면서 동시에 에너지 절감이라는 두 가지 상충한 목표를 달성하기 위해서는 실내 환경에 대한 실시간 감시가 선행되어야 한다. 이러한 실시간 환경 감시를 바탕으로 현재 실내 상태에 대한 추정이 이루어지고 이 추론 결과에 따라 건물 내 다양한 설비에 대한 최적 제어를 수행함으로써 실내 쾌적도 유지와 에너지 절감이라는 두 가지 목표에 다가설 수 있다.

하지만 실내 공간에 대한 실시간 감시는 실 환경에서 다음과 같은 여러 가지 실질적인 어려움에 직면해 있다. 첫째는 사람이 느끼는 쾌적의 다양성이다. 특정 공간을 점유하고 활동하는 사람은 시각에 대한 가시 쾌적도, 체온에 영향을 주는 열 쾌적도, 소음에 대한 청각 쾌적도 등이 있으며, 여기에 이산화탄소와 미세먼지 농도에 따른 공기질 쾌적도 등 다양한 인자에 영향을 받는다[2]. 이것은 일반적인 생활환경에서뿐 아니라, 근무 환경에도 큰 영향을 미친다. 예를 들어, 공장 작업 환경에서는 근무자 건강을 위해 특정 조도 이상을 유지하고 특정 소음 이상에서는 안전 장비를 착용하도록 법제화되어 있다. 또한 생산성 측면에서도 실내 온도와 공기질 등은 공장에서 생산되는 제품 품질에 직접적인 영향을 준다. 따라서 다양한 측면에서 실내 환경에 대한 복합적인 감시가 요구된다[3].

둘째는 다양한 응용 서비스 및 분석 서비스와 연동하기 위한 상호운용성과 서비스 인터페이스 문제이다. 실내 환경에 대한 센싱 결과는 센서 장치에서 출력한 데이터로 표현된다. 이러한 데이터를 교환하는 프로토콜과 데이터 형식은 장치 제조사의 독자적인 형태가 대부분이다. 비단 장치에서 일반 건물과 공장에서 주로 사용되는 Modbus, OPC UA 등과 같은 프로토콜을 지원하더라도 메모리 맵 또는 데이터 맵을 통해 장치 및 데이터에 대한 사전 지식이 필요하며, 이러한 프로토콜들은 고수준의 응용 서비스와 연동하기 위해 만들어진 프로토콜이 아니다.

셋째는 장기간 실내 환경 측정에 대한 장치 안정성 문제이다. 최근 환경 센서 장치는 센싱 장치이자 일부 제한적인 지능화 부여 및 에지 컴퓨팅(edge computing) 단말로의 역할로 인해 점차 고성능화되어 가고 있다.[4] 또한 유증기, 진동 등 가혹 환경에 노출됨에 따라 기존의 간단한 회로 구조를 가진 센싱 장치 대비 상대적으로 장치 안정성 문제가 발생할 가능성이 커지고 있다. 장시간 데이터 수집에 대한 결측은 근래의 고도화된 기계학습 알고리즘으로도 정확한 보간이 어려울뿐더러, 실시간 설비 제어에 대한 기반 데이터로 활용되는 특성상 데이터 수집 중단은 실내 쾌적도 유지와 에너지 절감 목표 달성에 큰 차질을 줄 수 있다.

넷째는 다양한 장소에 설치된 많은 수의 장치를 어떻게 효율적으로 관리하는지에 대한 문제를 들 수 있다. 장치 관리 문제는 장치 설치 장소에 따라 각기 다른 설정 관리, 다수 장치에 대한 초기 세팅의 용이성, 센서 장치들의 실시간 관제 및 관리 문제가 있다.

본 논문에서는 이러한 문제점들을 해결하기 위해 W3C의 HTTP 기반 표준 프로토콜인 사물 웹(Web of Things, WoT)[5] 기반 고가용 복합 환경센서 장치를 제안한다. 제안하는 복합 환경센서 장치는 실내 환경의 다양한 쾌적도를 계측하기 위해서 온도, 습도, 이슬점, 조도, 소리, 이산화탄소(CO_2) 농도, 미세먼지 농도, 폼알데하이드(HCHO) 농도를 측정하며, 이는 예상 평균 온열감(Predicted Mean Vote, PMV)/예상 불만족도(Predicted Percentage of Dissatisfied, PPD)[6], 실내공기 환경지수 등의 쾌적도 산출에 사용된다.

장치 상호운용성 및 응용/분석 서비스 연계 문제를 해결하기 위해서는 WoT 프로토콜을 개발하여 적용한다. WoT 프로토콜은 사물 명세(thing description, TD)를 통해 장치에 대한 사전 지식 없이 센서 데이터를 교환한다. TD는 크게 사물의 상태를 나타내는 프로퍼티(property), 사물의 기능 수행을 담당하는 액션(action), 사물의 장치 이벤트(event)로 구성된다. 그리고 망 내 WoT 지원 장치 간 상호 발견 및 TD 획득을 위한 주소 공유를 위해서 mDNS(multicast domain name system) 프로토콜 기반 장치 발견(device discovery) 서비스를 제공한다. 네트워크 하드웨어 측면에서는 유선 랜, 무선 랜, 저전력 블루투스를 지원하며, 설비 자동화 시스템과 연동되지 않은 주변 소규모 설비 제어 지원을 위해 적외선 송/수신부(IrDA)를 내장한다.

장치 안정화 관점에서는 하드웨어 및 소프트웨어 측면을 나누어 설계 제안한다. 하드웨어 측면에서는 CPU가 관장하는 센서 데이터 수집, 후처리, 전송 부분과 하드웨어 및 네트워크 상태를 감시하고 복구하는 별도의 MCU를 두도록 구성한다. 이를 통해 하드웨어 및 네트워크 상태 관점에서의 안정성을 향상시킨다. 소프트웨어 측면에서는 내부 소프트웨어 구조에 마이크로서비스 아키텍처(microservice architecture, MSA)를

채용하여 서비스 구성요소를 분할 설계한다. 개발된 개별 마이크로서비스는 컨테이너화되어 마이크로서비스 간 독립적인 인터페이스를 연동하여 전체 센서 장치가 동작하도록 개발한다. 마이크로서비스는 크게 Sensor Data Collector, Sensor Data Renderer, Sensor Data Aggregator, Sensor Data Forwarder, Sensor WoT Discovery, Sensor WoT Service 등으로 구성된다. 이러한 마이크로서비스들은 건물 설비/제어에 활용되는 특성상 서비스 고가용성을 보장하고 자동 장애 복구 기능을 지원한다.[7]

센서 장치 관리 관점에서는 장치의 초기 세팅에서부터 설치 장소에 따라 달라지는 설정 관리, 센싱 데이터 수집 및 처리, 센서 장치의 실시간 관제까지 모두 클라우드 기반의 관리 플랫폼의 통제 속에 운용이 이루어지도록 개발한다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구에 관해 기술한다. 3장에서는 하드웨어 및 소프트웨어 개발 내용 및 결과에 관해 기술하며 WoT 프로토콜과 장치 적용 결과도 함께 제시한다. 4장에서는 클라우드 플랫폼 기반 센서 데이터 수집 및 장치 관리에 대해 기술한다. 마지막으로 5장에서는 이 논문의 결론을 논의한다.

Ⅱ. 관련 연구

최근 ICT 기술의 발달로 인해 저비용으로 많은 사물인터넷(Internet of Things, IoT) 장치들을 설치하여 다양한 분야에서 활용하고 있다. 특히 사람이 자주 머무는 사무실, 공장, 주택 등 다양한 장소에 IoT 기술을 결합한 장치들을 배치하여 생활환경의 질을 분석하고 개선해 가고 있다. 이처럼 실내 환경 및 생활 조건을 개선하기 위해서는 서비스를 제공하고자 하는 장소의 데이터 측정 및 수집이 중요하다.

남재현[8]은 실내 공기질 및 오염 물질 상태를 실시간 측정하여 데이터베이스에 저장한 후 환경 분석을 수행할 수 있는 시스템을 제안하였다. 제안하는 센서는 아두이노를 기반으로 제작되었으며 LED 표시등을 통해서 설치 장소에서 실내 환경 상태를 즉시 인지 가능하도록 개발했다. 개발한 시스템은 수집된 데이터를 바탕으로 주변 공기 관리 장치에서 활용할 수 있게 하였다.

오창세 외 4명[9]은 실시간 실내 공기질을 모니터링하기 위해서 온도, 습도, CO_2, CO, VOC를 측정할 수 있는 센서 모듈을 개발하였다. 개발된 센서 장치는 Zigbee 프로토콜로 통신하며 별도의 소형 서버를 통해 데이터를 수집하고 공기질을 분석하였다.

강정훈 외 4명[10]은 IoT 장치를 사용해서 산업 데이터를 수집하는 모델을 제안하였다. 데이터를 수집하여 전처리한 이후 데이터베이스에 저장하는 과정에서 일반적인 병렬 데이터 처리 방식인 멀티 프로세싱을 사용할 경우, 데이터 손실 문제가 발생할 수 있음을 지적했다. 이를 해결하기 위해 병렬로 동작하던 데이터 처리 과정을 선형적으로 바꿔 구성하였다. 탐색, 전처리, 저장/전송의 과정이 하나의 파이프라인에서 동작해야 하므로 해당 논문에서는 생산자-소비자 패턴을 적용하였다. 이를 구현하기 위해 Redis 큐를 사용하였으며, 컨테이너 기술을 사용해 각 프로세스를 모듈화하였다.

이근혁 외 4명[11]은 데이터 수집 프로토콜의 관점에서 문제를 해결하는 방법을 제안하였다. 기존 MQTT 프로토콜은 통신이 필요하지 않을 때도 연결을 유지해야 한다는 단점이 있다. 해당 논문에서는 위 단점에 대한 몇 가지 해결 방법을 제안한다. 그중 한 가지는 MQTT의 연결 유지로 인해 발생하는 불필요한 통신을 줄이기 위해 Polling을 사용하는 방법이다. MQTT는 초기 연결 비용이 많이 들어가기 때문에 연결을 유지하는 방법을 사용한다. 해당 논문에서는, 연결 유지에 발생하는 트래픽과 자원 소모를 줄이기 위해 장치가 수행해야 할 작업이 있을 때만 서버와 연결하도록 설계하여 불필요한 통신을 줄였다.

기존 연구는 주로 사람의 다양한 쾌적도를 측정하기보다 공기질과 같은 특정 환경 상태를 계측하고 분석하는 연구가 대부분이고, 군수 산업이나 항공 장비 등의 특수한 영역을 제외하고는 하드웨어/소프트웨어 관점에서의 장치 고가용성에 대한 고려가 크게 이루어지지 않고 있는 것으로 보인다. 또한 분석/응용 서비스 연동 관점에서 네트워크 프로토콜에 대한 문제 제기나 다양한 특성을 가진 공간에 다수의 장치가 배치되는 IoT 장치의 관리 측면에 관한 연구가 상대적으로 부족한 실정이다. 본 논문에서는 이와 같은 문제점들을 보완할 수 있는 장치 설계를 제안하고 실제 개발된 결과물에 대해 논의한다.

Ⅲ. IoT 복합 환경센서 장치 개발

3.1 하드웨어 구조

본 논문에서 제안하는 IoT 복합 환경센서 장치는 주변의 상태를 다양한 센서를 통해 복합적으로 측정하여 장치 내부와 클라우드 플랫폼에 저장하고, 실시간 실내 환경 감시 및 상황 학습/추론 등의 분석과 응용 서비스 연계를 위해 개발된다. 센서 요구사항에 따라 높은 동작 온도에서도 안정적으로 작동할 수 있도록 내구성을 높

임과 동시에 심미성을 위해 철판(냉간압연강판)을 이용하여 하우징을 설계하였고, 조도, 온도, 습도, 소음, 이산화탄소 농도, 미세먼지 농도(PM-1, PM-2.5, PM-10), 폼알데하이드 농도 검출 센서를 탑재하였다. 특히, 위치에 의해 측정값이 크게 변화하는 온도/습도/조도 센서와 마이크/스피커는 외장으로 배치할 수 있도록 설계하였다. 이를 통해 실내 공간의 가시 쾌적도, 열 쾌적도, 청각 쾌적도, 공기질 쾌적도를 산출한다. 그림 1은 하드웨어 개발에 사용된 CPU/마더보드 사양과 조감도를 나타낸다.

그림 2는 개발된 실제 하드웨어 결과물을 나타낸다. 그림 2를 살펴보면 전면 중앙에 대형 주 디스플레이와 하단 오른쪽에 소형 보조 디스플레이를 장착하고 있다. 대형 주 디스플레이는 인터렉티브한 3D 렌더링 화면을 출력할 수 있어, 센서 장치가 설치되는 장소에 적합한 상태 화면을 보여준다. 화면에 출력되는 내용은 통합 대시보드, 센서별 실시간 검출 값과 히스토리 그래프 등이다. 각 정보 화면 간 전환은 하단에 위치한 하드웨어 스위치를 통해 이루어진다.

소형 보조 디스플레이는 하드웨어 및 네트워크 상태를 감시하고 출력한다. 그림 3은 보조 디스플레이 화면을 확대한 것으로, 현재 할당된 IP 주소와 MAC 식별자,

Quad core (64bit SoC @1.5GHz) + GPU (Cortex-A72)
2.4/5.0GHz IEEE 802.11 b/g/n/ac
Bluetooth 5.0, BLE,
Gigabit Ethernet PHY
Dual code MCU (Cortex-M0+)
3 x USB2.0 ports
8 x UART ports
6 x SPI, 4 x I2C ports
18 x PWM, 4 x ADC
UAC2.0 compliant ADC, DAC,

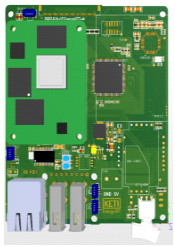

그림 1. 하드웨어 사양
Fig. 1. Hardware specification

그림 2. 복합 환경센서 장치
Fig. 2. Complex environmental sensor device

IP Address/ Network Mode
MAC ID/ Sensor Status
System Uptime

그림 3. 센서 장치의 보조 디스플레이
Fig. 3. Subdisplay of sensor device

장치 동작시간, 장치 모드, 그리고 알파벳 문자로 개별 센서의 연결 상태를 나타낸다.

그림 4는 본 논문에서 제안하는 복합 환경센서 장치의 하드웨어 구조를 나타낸다. 장치 하드웨어는 단일 프로세서 기반 하드웨어 안정성 문제를 해결하기 위해서 CPU와 MCU 두 개의 프로세서 유닛을 가지도록 설계된다.

CPU는 주로 센서들과 연결되어 데이터를 수집하고 처리하는 역할을 수행한다. 그리고 전면 주 디스플레이가 연결되어 3D 렌더링 데이터 시각화와 오디오 변환 처리를 담당한다. 여기서 3D 시각화는 후처리 된 센서 데이터를 GPU에 전달하여 그래픽 모형을 생성하고 주 디스플레이에 표출한다. 시각화 처리를 GPU에 위임하기 때문에 시각화에 필요한 CPU 자원을 대폭 절약할 수 있다. 데이터 시각화 화면은 복합센서 장치가 배치되는 공간의 특성에 적합한 쉐이더를 구성할 수 있도록 유연한 구조를 가지고 있다. 네트워크는 유선 랜, 무선 랜(IEEE 802.11 b/g/n/ac), 저전력 블루투스(BLE)를 지원한다. 또한 자동 제어 시스템과 연동되어 있지 않은 센서 장치 주변의 소규모 설비 제어를 위해서 적외선 송/수신부(IrDA)를 내장하였다.

MCU는 CPU, 센서, 네트워크, 나머지 하드웨어 전반 상태를 감시하고 관리하는 역할을 수행한다. 제안하는 복합센서 장치는 범용 고성능 CPU를 내장하고 리눅스 기반 운영체제에서 동작하기 때문에 단순 센서로의 역할 뿐 아니라, 에지 컴퓨팅 노드로의 기능도 함께 수행이 가능하다. 일례로 광각 카메라를 복합센서에 장착

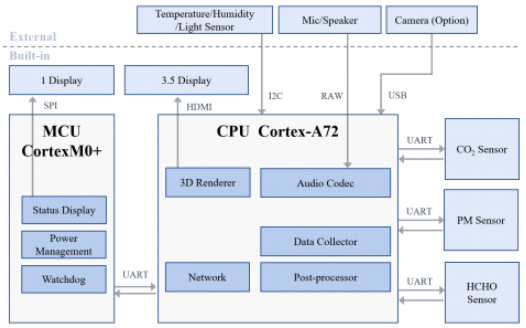

그림 4. 복합센서 하드웨어 구조[12]
Fig. 4. Hardware structure of complex sensor[12]

하여 경량의 딥러닝 모델을 탑재한 후 이미지 기반 재실자 인식 및 응용 서비스 연동이 가능하다. 그리고 실제 생산 공장에 배치될 경우에 수증기, 유증기, 진동 등과 같은 가혹 조건에 노출된 상태로 장시간 운용되어야 하므로 상대적으로 간단한 회로 구조를 가진 MCU 모듈을 별도로 두어 전체 하드웨어 및 네트워크 상태를 관제하여 시스템 안정성을 높인다.

MCU는 와치독 설정을 통해 이상 상태 및 대응 정책을 지정한다. 센서 연결 및 센싱 값 상태, 네트워크 연결 상태, 주 마더보드 상태에 따라 로그를 남기고 CPU 리셋 또는 MCU 리셋을 수행한다. MCU는 CPU의 전원 관리용 핀을 통해 직접 전원 상태를 제어할 수 있다. 절전 모드 전환, 정기적 리셋, 상태 기반 리셋 등을 제공하며, 그림 3과 같이 MCU에 연결된 보조 디스플레이에 현재 장치 상태 및 네트워크 정보를 나타낸다.

3.2 소프트웨어 구조

본 논문에서 제안하는 복합센서 장치는 Buildroot를 사용해 리눅스 기반의 경량 운영체제를 빌드하여 적용한다. 장치에 연결된 각종 센서로부터 데이터를 수집, 변환, 처리, 시각화하고, 데이터 연동 기능을 제공하는 소프트웨어들은 그림 5와 같이 모두 마이크로서비스 아키텍처 기반 서비스로 구현된다.

마이크로서비스는 기능을 분할하여 작은 독립된 단위의 서비스로서 상호 간 API를 사용하여 통신한다. MSA를 채용함으로써 각 마이크로서비스 기능을 구현하는 작업에 특화된 프로그래밍 언어를 사용할 수 있으며, 독립된 마이크로서비스 단위로 부하분산 및 장애복구 등의 고가용성 지원이 가능하다.

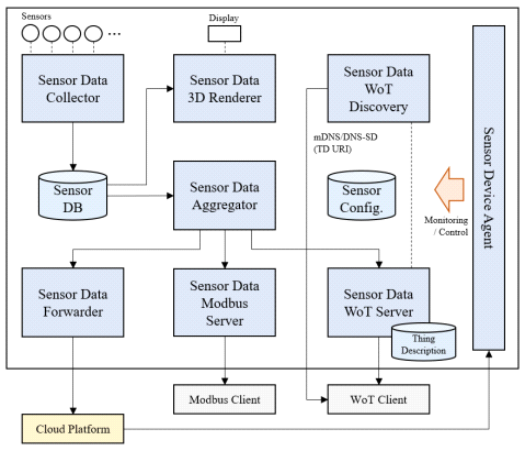

그림 5. 복합센서 마이크로서비스 구조[12]
Fig. 5. Microservice architecture of complex sensor[12]

그림 5에서 전체 시스템은 기능에 따라 서비스가 분리된 마이크로서비스 구조를 갖추고 있다. 복합센서 장치의 하드웨어/소프트웨어 관련 설정 및 전체 마이크로서비스 구성은 표 1과 같이 Sensor-Config 파일에 의해 관리된다. 이를 통해 특정 마이크로서비스 활성 유무, 센서 활성 유무, 서버 주소, 화면 설정 등 복합센서의 설정을 관리한다. 이를 통해 복합센서 장치 설치 요구조건에 맞춰 설정 및 마이크로서비스를 공간 및 응용 서비스 특성에 맞게 적용할 수 있다. 복합센서의 마이크로서비스는 호스트 운영체제에서 바로 실행되는 것이 아니라 컨테이너로 형태로 실행되어 동작한다. 이를 통해 복합센서 장치 개발 및 유지보수 관리 용이성을 높일 수 있다.

표 1은 Sensor-Config 설정의 기본 형식을 정리한 내용이다. Sensor Node는 센서마다 변동될 수 있는 요소를 담고 있다. 정확한 실내 상황 파악과 복합센서의 독립적인 관리 및 유지보수를 위해서 Metric과 Mac Address로 위치와 독립적인 구분 정보를 습득한다. 이는 클라우드 서버에서 설치 장소를 구분하여 데이터를 보여주기 위한 정보로도 사용된다. 그 외에도 설정을 크게 Device, Sensors, Data Source, Service로 나눠 관리한다. 표 1에 나타난 내용의 세부 항목들이 더 존재하며, 센서의 작동에 관여하고 있다.

복합센서 장치 시스템은 크게 데이터 수집, 가공, 전송, 조회 및 센서 관리로 분리할 수 있고, 각 기능들은 그림 5의 마이크로서비스로 구현되어 있다. Sensor Data Collector는 데이터 수집 기능을 제공한다. 독립된 센서에서 얻은 환경 정보를 전처리하여 Sensor Local Database에 적재한다.

그림 6은 센서 장치의 내부 데이터베이스 구조를 나타낸다. Sensor Database는 Id, Datetime, Value. Transmission Column 구조를 가진다. Id는 데이터마다 고유의 값을 가지며, Datetime은 수집 시각을 나타낸다. Value는 측정한 센서 데이터를 의미하고, Transmission은 원격 서버에 데이터가 정상적으로 저장되었는지 판단하는 값이다. Transmission은 HTTP, HTTPS, MQTT 등 프로토콜에 따라 개별적인 문자를 부여하고 저장하여 데이터의 중복 전송을 방지한다.

미세먼지 데이터를 저장하는 dust table은 대기 중 먼지 입자의 크기에 따라 값의 차이가 있다. 따라서 하나의 테이블에 여러 값을 저장하여 관리한다. pm1, pm25, pm10 컬럼은 각각 PM-1, PM-2.5, PM-10을 의미한다. PM은 Particulate Matter의 약자이며 보정 유무와 입자 크기 범위에 따라서 컬럼이 구분된다.

수집한 데이터는 Sensor Data Renderer를 통해 3D

표 1. Sensor-Config 설정 파일 구조
Table 1. Sensor-Config file format

Sensor Config			Explanation
Name			Configuration Name
Version			Configuration Version
Last Modified			Last Modified Date
Sensor Node	Service Id		Service Id
	Version	SW	Software version
		HW	Hardware version
	Metric		Environment Identifier (.env)
	Device Id		Device Id
	Default Tags		Tags
	Device	Mac Address	Change Mac Address
		Main Display	Main Display Configuration
		Sub Display	Sub Display Configuration
		Restart	Auto Restart Sensor
		Keep Alive	Keep a Sensor Alive
	Sensors : List [{'Name','enable'},···]		Collection Data List
	Data Source	Type	Database Type
		Path	Database Path
	Services	Sensor Data Aggregator	Enable, Version, Binding Address/Port
		Sensor Data Forwarder	Enable, Version, Forwarding Server Information
		Sensor Modbus Service	Enable, Version, Modbus Setting, Binding Address/Port
		Sensor WoT Service	Enable, Version, WoT Options, Binding Address/Port
		Sensor Discovery Service	Enable, Version, MDNS Setting WoT Options
		Sensor ThingWire Agent	Enable, Version, Binding Address/Port
		Sensor ThingWire Proxy	Enable, Version, ThingWire Server Information

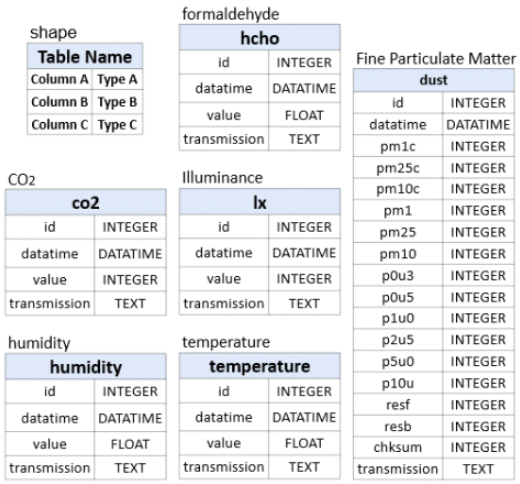

그림 6. 센서 내부 데이터베이스 구조
Fig. 6. Internal database schema for sensor data

그래픽 화면 또는 2D 차트 형식으로 시각화되어 디스플레이로 표출된다. 3D 그래픽은 OpenGL ES[13]를 사용하여 제공한다. OpenGL은 2차원 및 3차원 그래픽 처리를 위한 표준 API 규격이다. 2D 차트는 terminal-tables 라이브러리를 사용하여 터미널에서 표 형식의 데이터를 출력한다. Terminal-tables는 터미널에서 테이블 형식으로 데이터를 시각화할 수 있도록 도와주는 라이브러리이다. 따라서 3D 그래픽 표출 시보다 안정적이고, 자원 사용량 및 발열이 적다. 3D 그래픽은 심미성을 높이고 데이터 시각화를 통해 현재 환경 상태를 한눈에 파악할 수 있다. 센서 장치가 배치되는 장소의 실내 환경 분위기에 맞춰 시각화 방법을 변경하여 적용할 수 있다.

Sensor Data Renderer는 Sensor-Config의 Main Display와 Sub Display 설정을 참조한다. Main Display와 Sub Display 설정은 비슷한 구조를 가지지만, Sub Display는 Rotation과 Default Screen 설정이 존재하지 않는다. 디스플레이는 조도 값에 맞춘 화면 밝기 조절이 기능을 기본적으로 갖추고 있다. Sensor-Config에 정의된 내용은 해당 기술을 세부적으로 정의한다. 해당 기능을 통해 다양한 실내 환경을 고려한 설치가 가능하고, 복합센서가 설치된 공간에서 디스플레이 밝기로 인한 재실자의 불편함을 줄일 수 있다. 추가로 저녁 등 특정 시간대에서는 Night Mode의 밝기로 고정된다. 복합센서의 전력 소비를 줄이면서 야간 재실자를 배려할 수 있다.

Sensor Data Aggregator는 가공한 데이터를 조회할 수 있는 API를 제공한다. 가공된 데이터는 복합센서를

특정하기 위한 정보와 환경 정보를 포함한다. 이외에도 데이터 전송 여부 검사 등 Sensor Database에 접근해야 하는 작업을 처리한다. 해당 서비스는 다른 서비스의 요청을 처리하기 위해 Bjoern 모듈을 사용한다. Bjoern은 C 언어로 작성된 WSGI 서버이다. 트래픽이 집중되는 서비스이기에 비동기 WSGI 서버를 사용하여 안정성 및 성능을 개선하였다. Sensor Data Forwarder는 가공된 데이터를 서버로 전송한다. 지원하는 프로토콜은 HTTP/HTTPS, TCP/UDP, MQTT 등이 있다. Sensor Database에 저장된 과거 데이터를 서버로 전송하기 위해 매번 서버로 전송 확인이 안 된 50개의 가공된 데이터를 질의한다. 서버로 데이터 전송을 완료하면 Sensor Data Aggregator를 통해 데이터 전송 완료 여부를 transmission Column에 저장한다. 반복적으로 조회와 전송 확인을 수행하면서 중복 데이터 없이 보내는 서비스이다. 클라우드 플랫폼에 데이터를 전송하기 위해 해당 서비스를 사용하고 있다.

복합센서 장치 내부에 수집 저장된 데이터는 두 가지 프로토콜 서비스를 이용하여 질의가 가능하다. 첫째로 Sensor Modbus Server는 Modbus 프로토콜을 사용하여 데이터를 조회할 수 있다. Modbus는 OSI 모델의 레벨 7에 위치한 응용 계층 메시징 프로토콜이다. 환경센서 장치는 Holding Registers에 정보를 저장하고 TCP 또는 UDP로 데이터를 조회할 수 있도록 Modbus 서버를 제공한다.

둘째로 Sensor WoT Server는 WoT 프로토콜을 이용한 데이터 조회를 지원한다. WoT는 World Wide Web Consortium (W3C)에서 사물 인터넷의 상호 운용성과 활용성을 향상시키기 위해[5] 규정한 표준이다.

Sensor WoT Discovery는 mDNS 프로토콜을 사용하여 WoT 자원 명세 주소를 제공하며, mDNS를 통해 전달되는 장치 서비스 정보는 Sensor-Config에서 설정할 수 있다. 복합센서 장치에 WoT 프로토콜을 적용하여 개발한 부분은 다음 절에서 자세히 기술한다.

Sensor ThingWire Proxy와 Sensor ThingWire Agent는 복합센서 장치의 원격 관리를 위해 개발된 서비스이다. Sensor ThingWire Agent는 복합센서 내부에 직접 명령어를 보내고 결과를 가져오는 형식의 API를 제공한다. 이 API를 통해 장치 내부 상태를 질의하고 제어 명령을 수행하도록 할 수 있다. 빈번히 사용되지만 길고 복잡한 제어 명령은 미리 짧은 키워드로 정의하여 사용할 수 있다.

Sensor ThingWire Proxy는 외부의 클라우드 관리 플랫폼 서버와 통신하며 복합센서 장치의 내부 상태 정보 및 제어 명령을 중개하는 서비스이다. 대부분 환경에서 복합센서 장치는 방화벽 또는 무선 접속점과 같은 NAT(Network Address Translation) 장치 하부에 위치하고 있기 때문에 외부의 클라우드 관리 서버에서 복합센서 장치로 먼저 서비스를 요청하는 것이 불가능하다. 따라서 Sensor ThingWire Proxy가 관리 플랫폼 서버에게 수행해야 할 태스크 목록을 질의하고 해당 태스크의 수행 결과를 응답하는 능동적인 형태로 동작하도록 설계하였다. ThingWire 관리 플랫폼과 복합센서 장치와의 상호 작용은 제4절에서 자세히 기술한다.

3.3 WoT 표준 프로토콜 적용

WoT는 W3C에서 제정 중인 표준 기술로 HTTP 프로토콜을 기반으로 JSON-LD (JSON for Linked Data) 형식의 메시지를 교환하여 장치 간 상호운용성, 안정성, 신뢰성을 보장하는 프로토콜이다. WoT 프로토콜은 그림 7과 같이 모든 장치에 대해서 사물 명세(thing description, TD)라 불리는 JSON-LD 문서에 장치(사물)의 속성 및 제공하는 기능들을 명시한다.

사물 명세는 프로퍼티, 액션, 이벤트로 구성되어 있다. 프로퍼티는 사물의 목록, 이름, 데이터 유형, 설명 등과 같이 상태나 구성을 나타낸다. 읽기와 쓰기 권한을 제공할 수 있다. 복합센서는 데이터 조회를 위한 서비스이기에 모든 프로퍼티가 읽기 권한만 지원한다. 환경 데이터는 짧은 시간 내에 변화된다. 따라서 Property Handler를 정의하여 실시간 데이터 조회가 가능하도록 제공한다. 액션은 디바이스 또는 서비스를 제어하기 위한 명령을 나타낸다. 장치의 원격 제어와 상호 작용을 위해 기술된다. 복합센서 장치에서는 내부의 센서나 서비스 설정 제어도 가능하지만, 내장된 적외선 송신 기능을 활용하여 주변 장치로 제어 신호를 보내 해당 공간의 제어기로서 역할 수행이 가능하다. 액션은 URI Variables를 지정하여, 해당 액션에서 요구하는 여러 파라미터의 데이터 타입과 데이터 범위를 지정할 수 있다. 전달받은 파라미터 값을 서비스 내부의 Action Handler

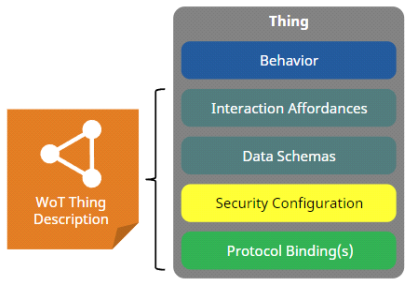

그림 7. WoT 사물 개념 구조 [5]
Fig. 7. Concept of WoT thing [5]

에 전달하여 액션 요청에 따른 동작을 실행한다. 이벤트는 특정 환경 조건을 트리거로 사용하거나 브로드캐스트로 알리는 등 상호 작용과 모니터링 기능을 제공한다.

WoT 구조에서 사물이 제공하는 TD를 읽어와 데이터 획득, 기능 실행, 이벤트 감시를 수행하는 다른 장치 또는 서비스를 소비자(consumer)라고 하며, TD를 통해 모든 서비스가 연동된다. 그림 8은 사물과 소비자의 관계를 도식화한 것이다.

복합센서 장치는 Sensor WoT Server와 Sensor WoT Discovery 마이크로서비스를 이용해서 WoT 표준 프로토콜을 지원하고 있다. WoT는 TD를 통해 데이터를 정의한다. TD는 사물에 대한 정보를 담고 있으며, JSON-LD 형식을 기반으로 작성된다. IoT 장치의 상호작용에 필요한 모든 정보를 정의하는 것에 목표를 두고 있다[5].

표 2는 본 논문에서 개발한 복합센서 장치의 TD를 축약하여 나타낸 것이다. 복합센서 TD에서는 열 쾌적, 가시 쾌적, 청각 쾌적, 공기질 쾌적을 계측할 수 있는 프로퍼티를 정의하고 있다. 프로퍼티에서는 개별 센서 데이터와 수집 시각, 외부 장치 수집 정보를 제공한다. 만약 데이터가 측정된 시각이 다르다면 가장 최근에 조회한 시각을 표시한다. 추가로 센서 장애로 과거의 데이터가 계속해서 노출되는 것을 방지하기 위해 현재 시각과 비교하여 30초 이상 차이가 발생하면 데이터는 Null 값을 응답한다. 표 3은 표 2에서 정의한 TD의 세부 항목을 기술하고 있다.

또한 TD에는 프로퍼티 외에도 IrDA를 통한 공간 내 냉난방 설비, MQTT 프로토콜을 통한 환기청정기 설비, 전용 API를 이용한 조명 제어를 위한 액션과 이벤트가 정의되어 있다. 표 4와 표 5는 각각 제안하는 복합센서 장치의 TD 액션과 이벤트를 나타낸다.

WoT 프로토콜의 TD가 해당 센서의 데이터, 기능, 이벤트들을 정의하고 있지만 WoT 소비자가 TD를 사용해서 데이터를 수집하거나 응용 서비스를 연동하기 위해서는 우선 TD가 위치하는 URL를 획득해야 한다. 이 TD의 위치를 가져오는 방법에는 TD 디렉토리 서비스, DNS-SD/mDNS, 분산 식별자(Decentralized Identifier, DID) 등이 있다. 제안하는 복합센서 장치에서는 mDNS 기반의 서비스 발견 기능을 제공한다. 이

그림 8. WoT 사물 장치과 소비자 [5]
Fig. 8. WoT thing and consumer [5]

표 2. WoT 사물 명세 JSON-LD
Table 2. Thing Description JSON-LD

```
{
  title: "KETI IoT Environmental Sensor",
  titles: {"en": "KETI IoT Environmental Sensor"},
  description: "KETI IoT as WoT",
  descriptions: {...},
  support: github URL,
  "@context": [...],
  properties: {
    timestamp: {...}, datetime: {...},
    temperature: {
      type: "float or null",
      description: "temperature",
      descriptions: {"en": "temp"},
      "iot:Custom": "example",
      observable: true,
      readOnly: true
    },humidity: {...}, dewpoint: {...}, lx: {...}, sound:
{...}, co2: {...}, hcho: {...}, pm1: {...}, pm10: {...},
pm25: {...}, aircontemp: {...}, airconoonoff: {...},
ventairvolume: {...}, ventonoff: {...}, lightonoff: {...}
  },
  actions: {
      ventairvolume: {
      description: "Ventilation Set Air Volume",
      descriptions: {...},
      uriVariables: {
        volume: {"type": "integer",
                 "minimum": 1,
                 "maximum": 6 }}},
    aircontemp: {...}, airconoonoff: {...}, ventonoff:
{...}, lightonoff: {...}
  },
  events: {
    aircontemp: {
      description: "aircon temp change event",
      descriptions: {"en": ""}
    },
    airconoonoff: {...}, ventairvolume: {...}, ventonoff:
{...}, lightonoff: {...}
  }
}
```

를 통해 지역망 내에서는 WoT 장치 간의 TD URL을 쉽게 발견하고 공유한다.

이러한 WoT 사물 발견 기능은 Sensor WoT Discovery 마이크로서비스에서 제공한다. mDNS는 DHCP와 같이 사전 정보 교환 없이 동일 네트워크에 속해있는 장치를 식별하는 멀티캐스트 프로토콜이다. 각 장치와 서비스를 식별하기 위해 Hostname과 Port, 사용 중인 TD 경로 조회가 가능하다. 그림 9와 같이 장치 또는 서버가 특정 서비스 식별 키워드(_wot._tcp)와 함께 mDNS 프로토콜을 사용하여 전송하면 요청받

표 3. WoT Properties 사물 명세표
Table 3. Thing Properties Description

TD		Description	Type
Title		KETI IoT Environmental Sensor	String
Description		KETI IoT as WoT Thing	String
Support		git://github.com/eclipse/thingweb.node-wot.git	Scripting API
Properties	Timestamp	timestamp	Int
	Datetime	datetime	String
	Temperature	temperature sensor	Float, Null
	Humidity	humidity sensor	Float, Null
	Dew Point	dewpoint	Float, Null
	Illuminance	light sensor	Int, Null
	Sound	sound sensor	Int, Null
	CO2	carbon dioxide	Int, Null
	HCHO	formaldehyde	Float, Null
	PM1	PM-1.0	Int, Null
	PM10	PM-10	Int, Null
	PM25	PM-2.5	Int, Null
	Aircon Temperature	air conditioner temperature	Int
	Aircon On Off	air conditioner on/off	Boolean
	VENT Air Volume	ventilation Air volume	Boolean
	VENT On Off	ventilation on/off	Boolean
	Light On Off	light on/off	Boolean

표 4. WoT Action 사물 명세표
Table4. Thing Action Description

TD		Description	URI Variables
Action	Aircon Temperature	air conditioner set temperature	Int (1 - 6)
	Aircon On Off	air conditioner on/off	Boolean
	VENT Air Volume	ventilation air volume	Boolean
	VENT On Off	ventilation on/off	Boolean
	Light On Off	light on/off	Boolean

표 5. WoT 이벤트 사물 명세표
Table 5. Thing Event Description

TD		Description
Events	Aircon Temperature	change event of air conditioner temperature
	Aircon On Off	change event of air conditioner power
	VENT Air Volume	change event of ventilation air volume
	VENT On Off	change event of ventilation power
	Light On Off	light change event

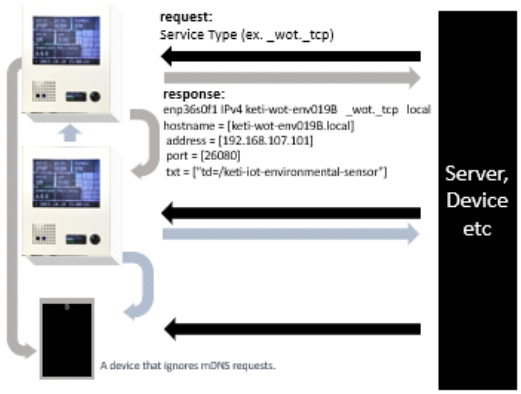

그림 9. mDNS 서비스 발견 흐름도
Fig. 9. mDNS service discovery flow

그림 10. 복합센서 mDNS TD 서비스 발견 질의 결과
Fig. 10. TD service discovery using mDNS

은 장치가 자신의 IP, Hostname 등 정보를 멀티캐스트로 다시 보내주는 방식으로 동작한다. 센서 내부에서는 mDNS 캐시를 사용해서 장치 식별 결과를 저장한다. 이렇게 캐시를 적용하면 네트워크 트래픽을 줄일 수 있고 서비스 응답 속도가 향상된다. 그림 10은 mDNS 기반 WoT 서비스 발견 기능을 적용한 복합센서 장치의 검색 결과를 나타낸다.

Ⅳ. IoT 복합 환경센서 데이터 수집 및 장치 관리

4.1 클라우드 서버 데이터 수집/처리

본 논문에서 제안하는 IoT 복합 환경센서 장치들은 내부 데이터베이스에 1차적으로 수집된 데이터를 저장한다. 외부 관리 플랫폼 연동 서버에 관한 정보가 Sensor-Config에 설정되어 있으면, 내부 데이터베이스에 저장되는 데이터를 클라우드 데이터 플랫폼으로도 전송하여 저장한다. WoT 표준 프로토콜과 Modbus 표준 프로토콜은 장치가 서버를 열고 있고 소비자가 데이터를 당겨(pull) 가는 방식이기 때문에 방화벽이나 NAT 뒷단에 존재하는 센서 특성상 외부에서 직접적으로 데이터 수집이 용이하지 않다. 따라서 클라우드 플랫폼으로 데이터 전송은 장치가 외부의 서버로 전송해 밀어 넣는(push) 방식으로 동작한다.

데이터는 HTTP REST API를 사용하며 JSON 형식으로 서버에 전송한다. 데이터 형식은 크게 Service, Auth 부분과 Data 부분으로 나눠진다. Service는 장치 식별과 해당 장치 유형의 데이터 처리 방법에 대한 고유 식별자를 담고 있으며, Auth는 인증 관련 정보를 가지고 있다. Data 부분은 해당 장치의 컴퓨팅 자원 및 네트워크 상태 정보와 연결된 센서들의 데이터를 포함한다.

그림 11은 클라우드 데이터 플랫폼의 복합 환경센서 데이터 처리 파이프라인을 나타낸다. 주요한 부분을 살펴보면 Data Preprocessor에서 데이터 형식 변환, 센서 값 보정, 센서 데이터 통계치 산출 등이 수행되며, Data Enricher에서 이산화탄소 기반 재실 추론, 인공신경망 등의 기계학습 기반 재실 추론, PMV/PPD 산출 등과 같은 분석 작업이 이루어진다. 그리고 복합센서 장치로부터 전송받은 센서 데이터들과 분석 데이터들은 모두 시계열 데이터베이스로 전송되어 저장된다. 또한 서비스 식별자(service ID)에 따라서 MQTT와 Redis 구독/발행 브로커를 통해 설비 제어 메시지나 이벤트 메시지를 클라우드 플랫폼 단에서 발행하거나 처리한다.

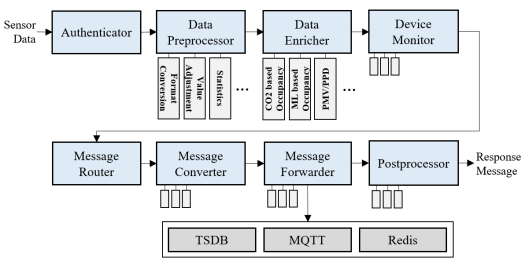

그림 11. 클라우드 데이터 플랫폼 데이터 수집/처리 [12]
Fig. 11. Data collection and processing of cloud data platform [12]

4.2 IoT 복합환경 센서 관리 체계 및 서비스

ThingWire는 복합 환경센서들을 포함하여 여러 장소에 설치되어 있는 다양한 장치들을 관리하기 위해 개발된 클라우드 플랫폼이다. 웹 관리 GUI를 통해 IoT 장치뿐만 아니라 네트워크상 다양한 장비들을 관리할 수 있다. 여러 유형의 장치들을 그룹, 조직 단위로 관리할 수 있으며, 조직마다 관리자의 역할에 기반한 자원 접근 제어를 수행한다.

그림 12는 ThingWire의 장치 기본 관리 체계를 나타낸 것이다. 여기서 장치는 복합센서를 포함한 여러 종류의 말단 기기를 의미한다. 이러한 장치들이 모인 논리적 집합이 그룹이며, 하나의 장치는 여러 개의 그룹에 중복되어 포함될 수 있다. 그룹이 모인 논리적 집합은 조직이라고 하며 하나의 그룹은 하나의 조직에만 포함된다. 하지만 장치는 여러 조직에 포함될 수 있다. 하나의 장치더라도, 장치를 관리하는 주체가 둘 이상일 수 있으므로 그림 12과 같이 기본 관리 체계를 설계한다.

ThingWire는 그림 13과 같이 서버 측과 클라이언트(복합 환경센서) 측에서 동작하는 여러 가지 서비스들을 통해 운영된다. 서버 측에는 크게 ThingWire WAS, Web SSH Manager, Device Manager 세 개의 백엔드 서비스가 동작한다. 클라이언트 측은 Sensor ThingWire Proxy, Sensor ThingWire Agent, Web SSH Manager 세 개의 서비스가 ThingWire에 연동되어 동작한다. 각 내부 서비스 통신 및 외부 서버 연동에는 HTTPS를 이용한 TLS 암호화 통신이 수행된다.

ThingWire WAS는 Express 프레임워크를 기반으로 구현된 웹 애플리케이션으로 ThingWire 플랫폼에 대한 클라이언트의 요청을 받아 처리를 수행하는 관문 역할을 한다. Device Manager는 장치와의 통신을 책임지는 웹 애플리케이션이다.

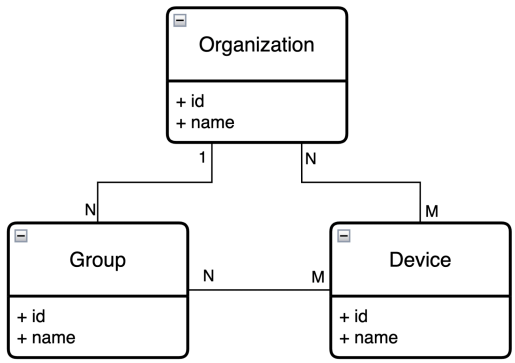

그림 12. 장치 기본 관리 체계
Fig. 12. Device management entity relation

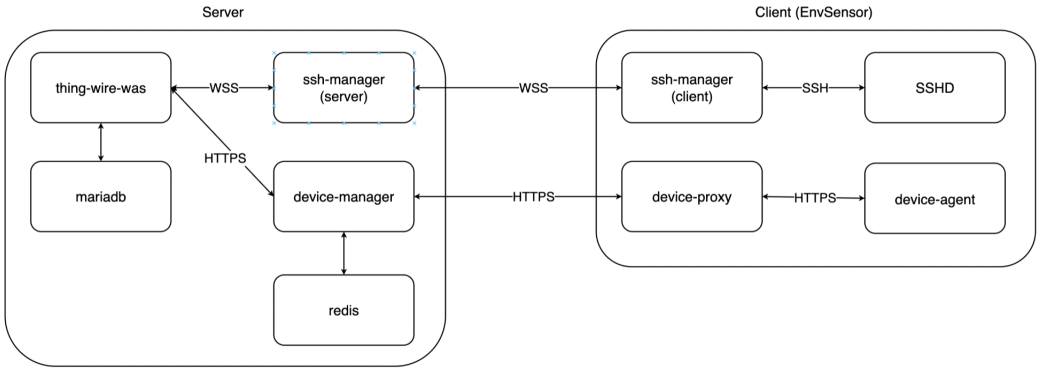

그림 13. ThingWire 서비스 다이어그램
Fig. 13. ThingWire service diagram

복합센서 장치들의 경우 무선 메시 네트워크를 구성하여 네트워크 게이트웨이가 동적으로 변하거나 방화벽 등 보안상의 이유로 외부에서 장치에 직접 접근하는데 제약사항이 많다. 이러한 이유로 ThingWire 관리 플랫폼에서 복합센서 장치에 요청해야 하는 질의가 있더라도 해당 센서의 엔드포인트 주소를 특정지을 수 없거나, 접근 불가능하여 직접 질의하는 방법은 사용할 수 없다. 대신 ThingWire 서버는 고정된 장소에서 운용되고, 도메인 이름을 통해 일관된 엔드포인트로 서비스를 제공할 수 있기 때문에 역으로 복합센서가 ThingWire 서버로 요청 메시지가 존재하는지 질의하는 방식으로 동작하도록 설계한다.

클라우드 관리 플랫폼과 복합센서 장치 사이의 동작 과정은 다음과 같다. 우선 클라이언트는 장치로 보낼 요청(제어 명령, Sensor-Config 설정, SSH 연결 요청 등)을 ThingWire WAS에게 전송한다. ThingWire WAS는 요청이 복합센서와 관련된 작업일 경우, Device Manager에게 해당 요청을 위임하고 결과에 대한 Redirect URL을 받아 결과를 기다린다. Device Manager는 메시지 브로커인 Redis에 해당 요청들을 캐싱하게 된다. 장치(복합 환경센서)는 주기적으로 ThingWire에서 전달할 요청 메시지가 있는지 Device Manager를 통해 확인한다. 만약 대기 중인 요청 메시지가 있다면 해당 메시지를 Device Manager로부터 가져와 요청을 수행한 후 Device Manager에게 결과를 응답한다. Device Manager는 장치의 명령 수행 결과를 받으면 Redirect URL에 결과를 게시하고 ThingWire WAS는 Redirect URL에 게시된 결과를 클라이언트에게 전송해서 요청에 대한 수행이 완료된다.

Web SSH Manager는 ThingWire 웹 관리 UI에서 복합센서 장치로 직접 SSH 서비스를 제공하기 위한 서비스이다. 복합센서 장치는 이전에 언급했듯이 네트워크 직접 접근이 힘든 특성 때문에 일반적인 SSH 연결 수립이 어렵다. 이를 극복하기 위해 Web SSH Manager가 서버와 클라이언트 측에서 동작하며 SSH 서비스를 지원한다. Web SSH Manager의 동작 과정은 다음 절에서 자세히 기술한다.

4.3 IoT 복합환경 센서 관리 API 개발

이 절에서는 클라우드 관리 플랫폼의 장치 등록, 제어 요청, 웹 SSH 동작 등 주요 API에 관해서 기술한다.

그림 14는 복합센서 장치 등록 시퀀스 다이어그램을 나타낸다. 그림에서 Sensor ThingWire Agent와 Sensor ThingWire Proxy는 환경센서 장치 내부에서 동작하는 마이크로서비스이다. Sensor ThingWire Agent는 관리 플랫폼의 요청에 따라 장치의 현재 자원 및 동작 상태를 제공하고 쉘 명령 등을 수행하는 HTTP REST API를 제공하는 서비스이다. Sensor ThingWire Proxy는 복합센서 장치가 클라우드 관리 플랫폼과 연동하기 위해 필요한 프록시 서비스이다. 새로운 장치는 장치 등록을 위해 ThingWire 서버에게 주기적으로 등록 요청 메시지를 보내고, ThingWire 내부에서는 해당 장치의 등록 절차를 진행한다. 그리고 조직 관리자가 관리 웹 UI를

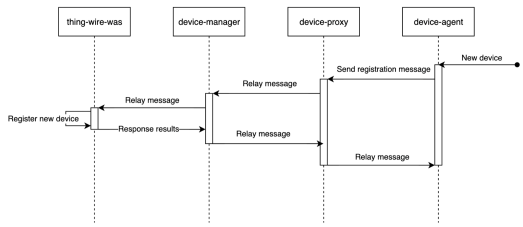

그림 14. 장치 등록 시퀀스 다이어그램
Fig. 14. Sequence diagram of device registration

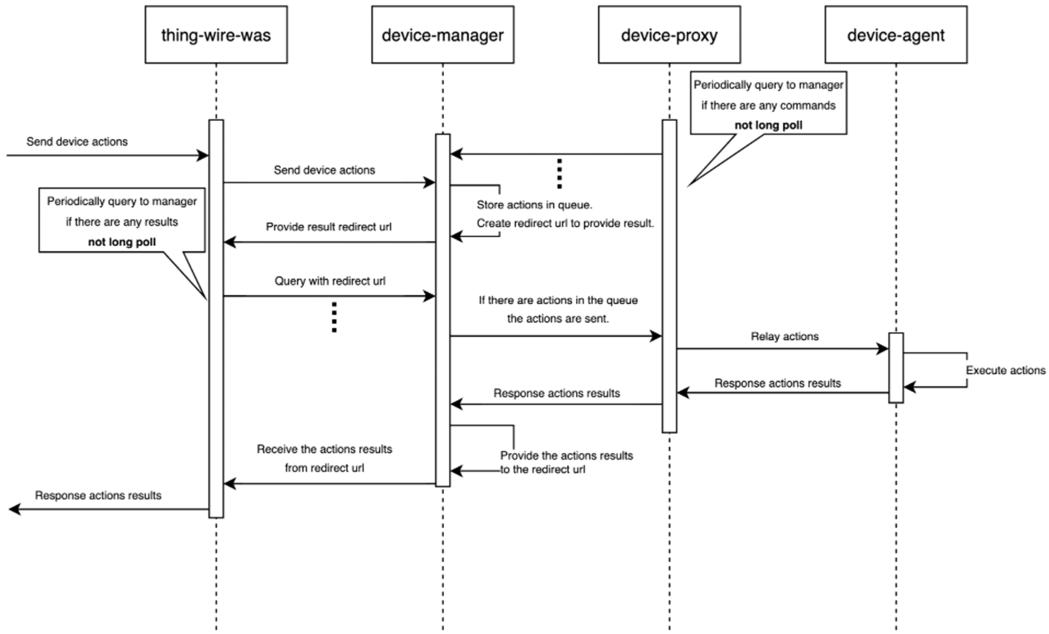

그림 15. 장치 제어 시퀀스 다이어그램
Fig. 15. Sequence diagram of device control

통해 그림 12의 관리 체계를 기반으로 복합센서 장치의 등록을 완료하게 된다.

그림 15는 장치 제어 시퀀스 다이어그램을 나타낸다. 장치 제어 요청은 상대적으로 많은 과정을 거치게 된다. 이전에 설명한 것처럼 복합센서 장치의 네트워크 엔드포인트로 직접 접근할 수 없는 경우가 대부분이기 때문에 장치에 송신해야 할 제어 메시지가 있으면 그 메시지를 장치가 스스로 질의해서 가져가는 동작 구조여야 한다. 이를 위해 Device Manager는 명령 큐를 사용한다. 장치로 보낼 제어 명령은 Device Manager 명령 큐에 저장하고, 장치는 자신의 식별자가 할당된 명령 큐에 수행해야 할 태스크가 있는지 주기적으로 질의한다. 이 주기는 선형적으로 증가하는 백오프 알고리즘을 사용하며, 응답이 없다면 최대 시간까지 백오프 시간까지 증가하고 응답이 있다면 다시 최소 백오프 시간까지 줄어드는 형태로 동작한다.

사용자가 장치로 명령을 보내게 되면, ThingWire WAS는 이 명령을 Device Manager로 보낸다. Device Manager는 해당 명령을 명령 큐에 저장하고 ThingWire WAS에게는 새로운 Redirect URL을 제공하게 된다. 이 Redirect URL은 복합센서 장치로부터 명령 처리가 완료되면, 해당 결과를 게시하는 임시 URL로써, ThingWire WAS는 이 Redirect URL을 주기적으로 확인한다. Device Manager의 명령 큐는 복합센서 장치의 명령 큐 요청이 오기 전까지 명령을 저장 및 대기하다가 요청이 들어오면 명령 큐에 저장된 모든 명령을 장치에게 제공한다. 이후 장치는 명령들을 처리하여 Device Manager로 전달하게 되고, Device Manager는 명령 결과를 Redirect URL에 게시한다. 주기적으로 Redirect URL를 확인하는 ThingWire WAS는 게시된 결과를 확인하고 사용자의 웹 화면에 전달한다.

장치와 서버 간 통신에 웹소켓과 같은 양방향 통신 프로토콜을 적용할 수도 있었지만, 무선으로 연결되는 복합센서 장치 특성상, 연결에 대한 신뢰성이 떨어져 위와 같은 방법을 사용하여 개발한다.

웹 SSH는 웹 UI를 통해 장치에 SSH 프로토콜로 접근할 수 있는 서비스다. 그림 16은 웹 SSH의 동작 과정을 나타낸다. 사용자가 SSH 연결을 요청하면 사용자와 서버 측 Web SSH Manager (server)의 WSS (WebSocket Secure) 연결이 수립된다. 이후 SSH 연결 요청이 Device Manager로 전달되고 Device Manager는 Web SSH Manager (server) 에게 Session ID를 제공한다. 제공된 Session ID는 Device Manager의 SSH 연결 요청 큐에도 저장된다. 장치가 연결 요청을 큐에서 가져가면 Web SSH Manager (client)는 localhost로 SSH 연결을 수립하고, 제공된 Session ID로 Web SSH Manager (server)와 WSS 연결을 수립한다. Web

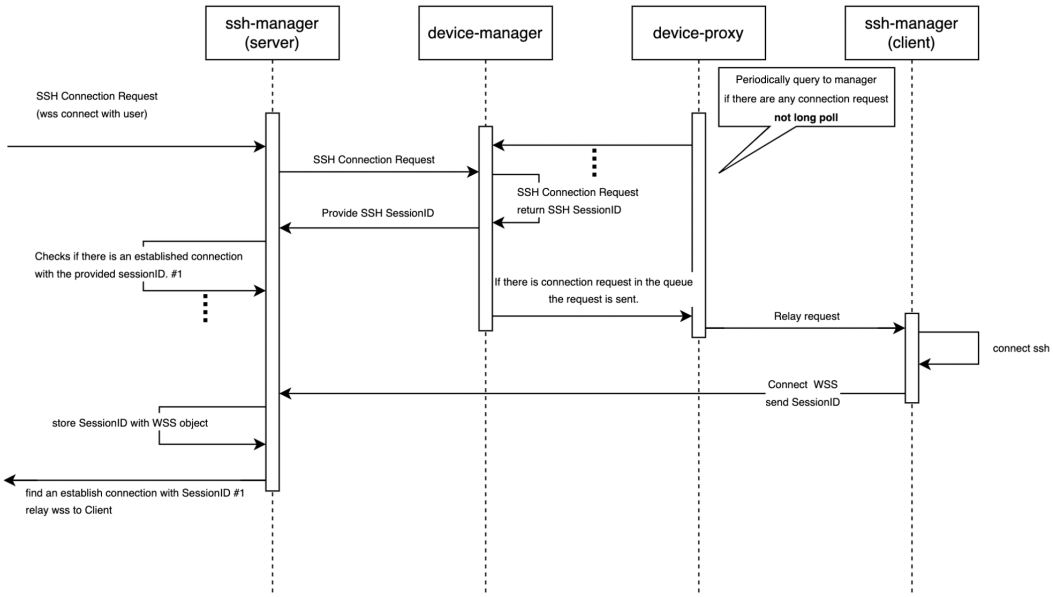

그림 16. 웹 SSH 서비스 시퀀스 다이어그램
Fig. 16. Sequence diagram of Web SSH service

SSHManager (server)는 제공된 Session ID로 수립된 연결이 있다면 해당 연결을 클라이언트 웹 UI로 중개해 줌으로서 SSH 서비스가 정상적으로 동작한다.

IoT 장치를 제어하는 ThingWire에서는 높은 수준의 보안이 요구된다. IoT 장치를 직접 제어할 수 있을 뿐 아니라, 웹 SSH 터미널을 통한 접근이 가능하기 때문에 더욱 높은 보안 요구사항이 적용되어야 한다. ThingWire는 2레벨로 보안을 제공한다. 첫째로 사용자 수준에서 보안이 제공된다. 사용자와 ThingWire 간 제공되는 보안으로서, 역할 기반 접근 제어 방식을 통해 자원에 대한 사용자의 접근을 제어한다. 권한은 각 조직의 관리자가 부여할 수 있으며 역할 단위로 부여가 된다. 역할은 여러 권한들의 집합으로서, 관리자는 여러 조직별의 권한들을 조합하여 하나의 역할을 만들고 해당 역할을 사용자에게 부여함으로써 사용자의 원하지 않는 접근을 제어할 수 있다. 둘째 보안은 시스템 수준에서의 보안으로 ThingWire IoT 장치 간 제공되는 보안이다. 이 보안은 악의적인 사용자 접근을 차단한다. 이 수준의 보안은 토큰을 통한 접근 제어로 제공된다. 정상적인 사용자는 토큰을 발급받아, 장치에 접근할 수 있다. 하지만 토큰이 없는 사용자나, 정상적이지 않은 토큰을 가지고 있는 사용자의 경우 인증받지 못해 서비스에 대한 접근이 거부된다. 기밀성 측면에서는 내부 서비스 간, 외부-내부 서비스 간의 모든 구간에서 TLS 통신이 적용된다.

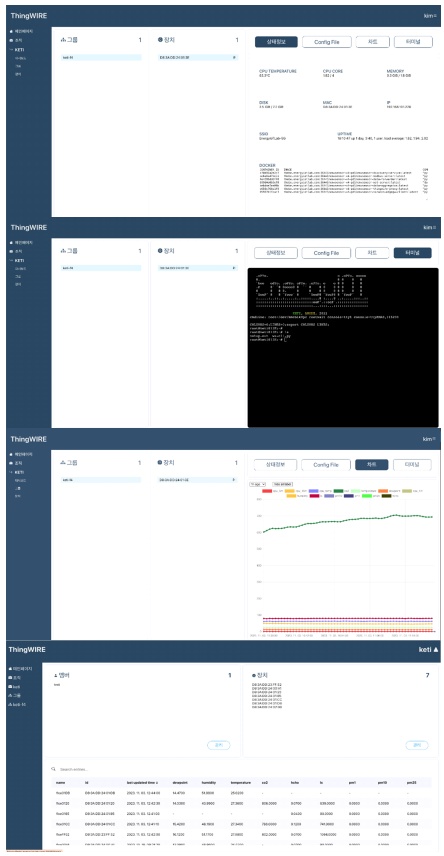

그림 17. 관리 웹 UI
Fig. 17. Management Web UI

관리 웹 UI는 뷰(view) 모듈이라고 불리는 독립된 화면 영역의 조합으로 구성되며, 개별 뷰는 별도의 독립된 스크립트로 개발된다. 뷰를 이용한 화면의 구성과 뷰 간의 연동은 브라우저의 화면 영역을 정의하는 설정 파일로 명세하기 때문에 뷰의 재활용성과 구성 유연성이 높다. 다음 그림 17은 뷰 조합을 이용한 웹 관리 UI 화면의 예시를 나타낸다.

개발된 복합센서 장치들은 현재 연구원, 사무시설(고층빌딩), 상업시설(대형마트) 등에 설치되어 그림 18과 같이 실내 환경 데이터를 수집하고 있으며 수집된 데이터는 실내 공조기 등의 제어 기반 데이터로 사용되고 있다. 그림 19은 복합센서 관제 웹 서비스의 일부 화면을 나타낸다. 그림 19의 상단 그림은 현재 실내 공간에 배치된 전체 복합센서 장치 상태와 센서값을 한눈에 관제할 수 있는 화면이다. 그림 19의 하단 그림은 도면을 3D 시각화 형태로 구성하여 실내 내부 공간을 직접 이동하며 실내 환경 상태와 재실자 수를 관제할 수 있는 메타버스 개념이 도입된 UI 화면을 나타낸다.

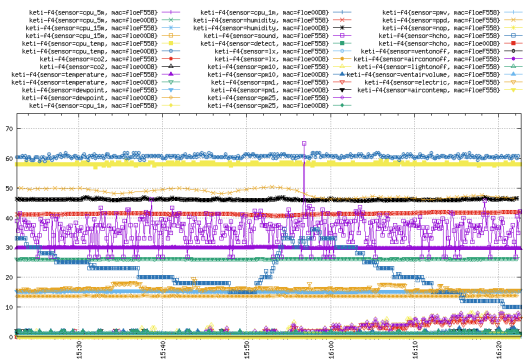

그림 18. 실내 환경 데이터 수집 처리
Fig. 18. Indoor environmental data collection

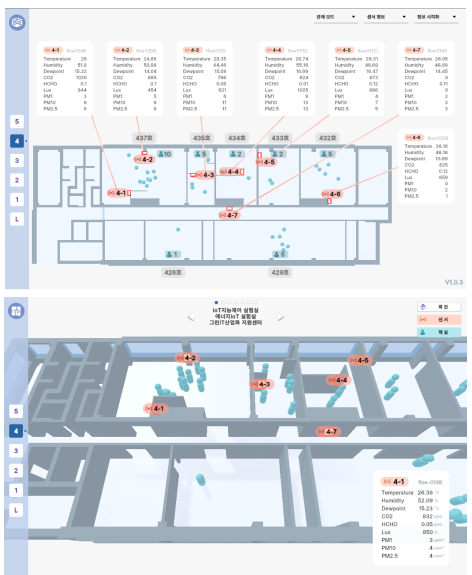

그림 19. 복합센서 장치 관제 시각화
Fig. 19. Sensor Device Monitoring

V. 결 론

본 논문에서는 효과적인 실내 환경 관제 및 안정적인 데이터 수집/처리를 위한 IoT 복합 환경센서 장치를 제안하였다. 제안하는 복합센서 장치는 온도, 습도, 조도, 소음, 이산화탄소, 미세먼지, 이산화탄소 등과 같은 실내 환경 상태를 측정할 수 있는 복합센서를 탑재하여 사람의 다양한 쾌적도를 산출한다. 또한 안정적인 데이터 수집을 위하여 CPU/MCU 기반 하드웨어 구조와 마이크로서비스 아키텍처 기반 데이터 수집 구조를 제안하였다. 또한 응용/분석 서비스와 유연한 연동을 지원하기 위해서 HTTP 기반의 WoT 프로토콜을 개발하여 적용하였고 그 결과 센서 장치에 대한 사전 정보 없이도 센서 데이터를 연동할 수 있는 방안을 제시하였다. 마지막으로 클라우드 데이터 플랫폼의 데이터 처리 파이프라인과 장치 관리 플랫폼인 ThingWire를 제안하였다.

이 개발 결과는 건물 에너지 절감과 건물 공간 사용자 쾌적도 사이의 균형 있는 설비 제어를 위한 응용 서비스에서 실시간 근거 데이터 수집 처리에 적극 활용 가능할 것으로 기대된다. 향후 연구는 열화상/광각 이미지 등을 활용하여 익명 재실자 수를 추론해내는 센서 장치와 소프트웨어 프로세서를 설계/구현하는 것이다.

References

[1] W. Zhang, Y. Wu, and J. K. Calautit, "A review on occupancy prediction through machine learning for enhancing energy efficiency, air quality and thermal comfort in the built environment," *Renewable and Sustainable Energy Rev.*, vol. 167, 2022. (https://doi.org/10.1016/j.rser.2022.112704).

[2] H. Zhu, X. Lian, Y. Liu, Y. Zhang, and Z. Li, "Consideration of occupant preferences and habits during the establishment of occupant-centric buildings: A critical review," *Energy and Buildings*, vol. 280, 2023. (https://doi.org/10.1016/j.enbuild.2022.112720).

[3] S. Khadka, M. Shrestha, and H. B. Rijal,

"Investigation of the thermal comfort and productivity in Japanese Mixed-Mode office buildings," *The J. Engineering Res.*, vol. 19, no. 1, pp. 63-72, 2022. (https://doi.org/10.53540/tjer.vol19iss1pp63-72)

[4] D. Sanders, "Environmental sensors and networks of sensors," *Sensor Rev.*, vol. 28, no. 4, 2008. (https://doi.org/10.1108/sr.2008.08728daa.002)

[5] *Web of Things Architecture*(2020), Retrieved Nov., 3, 2023, from https://www.w3.org/TR/2020/REC-wot-architecture-20200409/

[6] ISO 7730:2005, *Ergonomics of the thermal environment - Analytical determination and interpretation of thermal comfort using calculation of the PMV and PPD indices and local thermal comfort criteria*. 2005, https://www.iso.org/standard/39155.html

[7] B. Butzin, F. Golatowski, and D. Timmermann, "Microservices approach for the Internet of Things," in *2016 IEEE 21st Int. Conf. ETFA*, pp. 1-6, 2016. (https://doi.org/10.1109/etfa.2016.7733707)

[8] J.-H. Nam, "Implement of analysis system with indoor environment monitoring based on IoT," *J. KIICE*, vol. 23, no. 12, pp. 1687-1692, 2019. (https://doi.org/10.6109/jkiice.2019.23.12.1687)

[9] C.-S. Oh, M.-S. Seo, J.-H. Lee, S.-H. Kim, Y.-D. Kim, and H.-J. Park, "Indoor air quality monitoring systems in the IoT environment," *J. KICS*, vol. 40, no. 5, pp. 886-891, 2015. (https://doi.org/10.7840/kics.2015.40.5.886)

[10] J. H. Kang, C. S. Chae, H. W. Kang, C. W. Kim, and H. S. Choi, "Industrial IoT big data management system design," *Annual Conf. IEIE*, pp. 2133-2136, 2020.

[11] G. H. Lee, D. H. Kim, C. H. Jeon, H. S. Jeon, and H. J. Park, "A study for improving efficiency of IoT environment and solution based on MQTT Protocol," *J. KICS*, vol. 44, no. 7, pp. 1318-1326, 2019. (https://doi.org/10.7840/kics.2019.44.7.1318).

[12] D. Kwon, S. Shin, and Y. Ji, "A study on the microservice-based IoT sensor and cloud platform for indoor environment monitoring and occupancy inference," in *Proc. Symp. KICS*, pp. 233-234, 2023.

[13] khronos, *OpenGL ES Overview*, Retrieved Nov., 26, 2023, from https://www.khronos.org/opengles/

권 동 우 (Dongwoo Kwon)

2010년 : 계명대학교 공학사
2012년 : 계명대학교 공학석사
2017년 : 계명대학교 공학박사
2018년 : 계명대학교 산업기술연구소 연구원
2018년~현재 : 한국전자기술연구원 에너지IT융합연구센터 선임연구원
<관심분야> IoT, 데이터 플랫폼, 에너지데이터 분석

신 슬 비 (Seulbi Shin)

2020년~현재 : 성결대학교 컴퓨터공학과 학사
2022년~현재 : 한국전자기술연구원 에너지IT융합연구센터 연구원
<관심분야> IoT, 빅데이터, 인공지능

김 재 현 (Jaehyeon Kim)

2023년 : 경기대학교 공학사
2021년~현재 : 한국전자기술연구원 에너지IT융합연구센터 연구원
<관심분야> IoT, 빅데이터, 클라우드

지 영 민 (Youngmin Ji)

2005년 : 경희대학교 공학사
2007년 : 고려대학교 공학석사
2007년~2009년 : 모토로라 Things to Things Research Center 연구원
2010년~2012년 : 삼성SDS 정보통신기술연구소 선임 연구원
2012년~현재 : 한국전자기술연구원 에너지IT융합연구센터 책임연구원
<관심분야> IoT, 빅데이터, 인공지능 기반 상황 인지 분석