Journal of KIIT. Vol. 23, No. 2, pp. 209-220, Feb. 28, 2025. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2025.23.2.209

디바이스 레지스트리 기반 아두이노 코드 자동 생성 시스템

김재엽*, 이석훈**

Automated Code Generation System for Arduino based on Device Registry

Jaeyeob Kim*, Sukhoon Lee**

요 약

사물인터넷 영역에서도 자동화 시스템이 발전하고 있으며, 코드 자동 생성 기술은 이를 가속하고 있다. 하지만 IoT 디바이스의 액추에이터 제어를 위해서는 코드 자동 생성 시 디바이스의 정확한 정보의 입력이 필수적이다. 따라서, 본 논문에서는 디바이스 레지스트리 기반의 아두이노 코드 자동 생성 시스템을 제안한다. 제안 시스템은 사용자가 디바이스 레지스트리를 활용하여 센서 및 액추에이터와 관련된 메타데이터를 정의하고 등록한다. 또한, GPT-3.5 API를 활용하여 등록된 메타데이터 기반의 아두이노 코드를 생성할 수 있도록 프롬프트를 정의한다. 마지막으로 구현 및 시나리오를 통해 제안 시스템이 잘 동작하는지 검증한다. 이를 통해 제안 시스템은 효율적인 IoT 디바이스 개발 방식을 제공하며, IoT 기술의 접근성과 활용성을 크게 향상할 수 있을 것으로 기대된다.

Abstract

Automation systems are advancing in the Internet of Things(IoT) domain, with code generation technology accelerating this process. However, to effectively control actuators in IoT devices, accurate device information input is essential during code generation. Therefore, this paper proposes an Arduino code generation system based on a device registry. The proposed system allows users to define and register metadata related to sensors and actuators by utilizing a device registry. Additionally, prompts are defined using the GPT-3.5 API to generate Arduino code based on the registered metadata. Finally, implementation and scenario testing verify that the proposed system functions as expected. This system offers an efficient approach to IoT device development and is expected to significantly improve the accessibility and usability of IoT technology.

Keywords

code generation system, automated code generation, chatgpt, device registry, metadata

* 국립군산대학교 소프트웨어학과 학부과정
- ORCID: https://orcid.org/0009-0000-5988-3416
** 국립군산대학교 소프트웨어학과 교수(교신저자)
- ORCID: https://orcid.org/0000-0002-3390-5602

· Received: Nov. 14, 2024, Revised: Dec. 31, 2024, Accepted: Jan. 03, 2025
· Corresponding Author: Sukhoon Lee
Dept. of Software Science & Engineering, Kunsan National University, Korea
Tel.: +82-63-469-8914, Email: leha82@kunsan.ac.kr

Ⅰ. 서 론

최근 사물인터넷(IoT, Internet of Things)와 자동화 시스템의 확산으로 인해 다양한 센서와 액추에이터를 제어하는 시스템에 대한 수요가 급격히 증가하고 있다[1]-[3]. 이러한 시스템은 환경 감지, 스마트 홈 제어, 산업 자동화 등 여러 분야에서 필수적인 기술로 자리 잡고 있다.

특히, IoT 플랫폼의 경우 일반적으로 서버와 디바이스 간의 통신 구조, 방식, 프로토콜 및 데이터 포맷 등의 문제로 인해 IoT 디바이스들은 플랫폼에 종속된다[4]. 즉, IoT 디바이스를 개발하려면 해당 플랫폼의 API, 동작 방식 등에 대한 정확한 이해를 필요로 하므로, 다양한 플랫폼에서 적용시키기 어렵다는 문제를 지닌다.

최근에는 생성형 AI를 활용한 자동화 기술이 빠르게 발전하고 있으며, 특히 코드 자동 생성과 관련된 연구가 활발히 진행되고 있다[5]-[7]. 이러한 기술 발전은 비전문가가 쉽게 사용할 수 있는 IoT 개발 환경을 제공하여 프로그래밍에 대한 지식 없이도 IoT 장치를 구성하고 제어할 수 있게 도와주는데, ChatGPT와 같은 LLM(Large Language Model)을 활용하는 방식이 주목받고 있다[8]. 그러나 LLM을 활용하더라도 사용자가 제공하는 정보가 모호하거나 불완전할 경우에는 원하는 코드를 정확히 생성하기 어렵고, 특히 액추에이터 제어와 같은 동작에서는 예상과 다른 비효율적인 결과가 발생할 수 있다[9]. 이러한 LLM을 효과적으로 활용하기 위해서는 명확한 요구사항과 필요한 디바이스 정보가 정확히 전달될 수 있는 구조가 필요하다.

이러한 구조는 메타데이터를 정의함으로써 해결할 수 있으며 메타데이터 기반으로 액추에이터 제어를 하는 연구가 진행되었다[10]. 이 연구에서는 디바이스 레지스트리를 사용하여 IoT 플랫폼 내에서 디바이스의 정보를 등록하고 관리하며 키-값의 쌍 형태로 유연한 정보 관리를 하였다[11]. 디바이스 레지스트리에 등록된 IoT 디바이스의 제어를 진행하였지만, 코드 작성은 사용자가 직접 해야 한다는 한계가 존재한다.

따라서, 본 논문은 디바이스 레지스트리 기반의 아두이노 코드 자동 생성 시스템을 제안한다. 먼저, 디바이스 레지스트리를 활용하여 개발하고자 하는 IoT 디바이스의 메타데이터를 정의하고, 아두이노 코드 생성 프롬프트 정의 시 메타데이터를 활용한다. 이후 서버와 아두이노의 통신을 위해 MQTT(Message Queuing Telemetry Transport) 토픽을 정의한다.

제안 시스템의 구현을 위하여 IoT 디바이스를 개발하고 센서값 및 액추에이터를 제어할 수 있는 모니터링 시스템을 개발한다. 또한, 시나리오를 통하여 제안 시스템의 기능을 검증한다.

본 논문의 구성은 다음과 같다. 제2장은 관련 연구를 기술하며, 제3장에서는 제안 시스템의 구조와 메타데이터 기반 코드 생성 기법에 관해 기술한다. 제4장에서는 구현 결과를 기술하며, 제5장에서는 시나리오를 자세하게 기술한다. 마지막으로, 제6장은 결론 및 향후 연구에 대하여 기술한다.

Ⅱ. 관련 연구

이 장에서는 코드 생성 시스템에 관한 연구와 아두이노 액추에이터 제어를 위한 기존 연구들을 기술한다.

2.1 코드 생성 시스템 관련 연구

최근 LLM 모델을 이용한 다양한 코드 생성 연구가 진행되었다[12]-[14]. LLM은 방대한 양의 텍스트 데이터를 학습하여 자연어 처리 및 생성, 텍스트 이해 능력을 갖춘 인공지능 모델로, 최근에는 코드 생성에도 활용된다.

S. Yeo et al.[12]의 연구는 소스 코드 생성을 위한 LLM의 성능과 가능성을 종합적으로 조사한 연구로, 코드 생성 시 LLM의 역할을 체계적으로 정리하고 평가하였다. 이 연구는 32개의 LLM 모델을 파라미터 크기에 따라 분류하고, HumanEval, MBPP 벤치마크 데이터셋을 활용하여 LLM 모델들의 종합적인 능력을 비교 분석하였다. 그 결과 GPT-3.5, GPT-4, Gemini Ultra, Gemini Pro, Magicoder, Deepseek- Coder-Instruct가 코드 생성 작업에서 우수한 성능을 보이는 것으로 측정되었다.

S. Kim[13]의 연구는 생성형 AI의 텍스트 생성 기능을 활용하여 프로그래밍 교육에 적합한 코드와 학습 자료를 생성하는 프롬프트 개발을 제안하였다. 학습자의 수준에 맞춘 코드와 연습 문제를 제공하기 위해 프롬프트 엔지니어링을 적용하였으며, 맞춤형 코드 생성에 중점을 두었다. 평가 결과, 개발된 프롬프트로 생성된 코드가 일반적인 프롬프트보다 더 일관성 있고 높은 유사도의 코드를 생성하는 데 효과적임을 확인하였다.

J. Shin et al.[14]의 연구는 자연어를 이용한 자동 코드 생성의 다양한 접근 방식을 조사하고 이를 카테고리별로 분석하였다. 연구에서는 자연어 설명을 기반으로 소스 코드를 생성하는 다양한 접근 방식을 입력 및 출력 형태에 따라 분류하였다. 분석 결과, 제한된 도메인에서는 구체적이고 완전한 코드 생성이 가능했으나, 일반화된 프로그램에서는 추상적인 설명 처리가 어려움을 보였다.

2.2 플랫폼에서의 액추에이터 제어 연구

IoT 디바이스를 제어하기 위한 플랫폼이 지속적으로 개발되고 있다[15]-[17]. 이러한 플랫폼은 대부분 시뮬레이션 환경에서 IoT 디바이스를 실험하기 위해 개발하였다.

I. Hwang et al.[15]의 연구는 초보자를 위해 GUI 기반으로 아두이노 제어를 쉽게 학습할 수 있는 시스템을 제안하였으며, Drag & Drop 방식의 GUI를 통해 C 언어 지식이 없어도 아두이노를 제어할 수 있도록 설계되었다. GUI에서 입력한 내용을 바탕으로 자동 생성된 아두이노 제어 코드를 C 언어로 변환하여 학습자가 확인할 수 있게 함으로써 전통적인 IDE에 비해 시간 절약 효과가 있음을 확인하였다. 그러나 이 시스템은 사용자가 직접 코드 구성에 관여해야 하므로, 입력 오류가 발생할 가능성이 있다는 한계가 있다.

A. C. F. D. Silva et al.[16]의 연구는 IoT 환경의 관리와 데이터 처리를 지원하는 MBP(Multi-purpose Binding and Provisioning Platform)를 제안하여, 비전문가도 쉽게 사용할 수 있도록 설계하였다. MBP는 CEP(Complex Event Processing)를 기반으로 IoT 환경을 모니터링하고, 이벤트-조건-액션 규칙으로 자동화된 IoT 애플리케이션을 지원한다. 그러나 규칙을 미리 설정하기 때문에 세부적인 설정에는 한정적인 한계가 있다.

S. Yang et al.[17]의 연구는 전기화재 예측과 예방을 위한 IoT 플랫폼 시스템을 제안하여, 과부하, 유독 가스, 과열 센서로 구성된 복합 센서를 통해 화재 위험을 감지한다. 이 시스템은 재난 예방에 초점을 맞추어 설계되었지만, 한정된 센서 구성으로 인해 범용적인 IoT 제어에는 한계가 있다.

기존 연구를 통해 코드 생성에 필요한 프롬프트 방식을 확인하고, 기존 액추에이터 제어 플랫폼에서의 한계점을 확인한다. 한계점으로는 사용자가 직접 코드 구성을 하거나 디바이스의 세부적인 설정, 범용적인 IoT 제어가 있었고, 이를 해결하기 위해 메타데이터 기반의 아두이노 코드 자동 생성 시스템을 제안한다.

III. 제안 시스템

이 장에서는 제안 시스템에 대한 구조를 설명한다. 먼저, 제안 시스템의 모듈 중 디바이스 레지스트리 데이터 구조를 기술하고 코드 생성, 센싱, 액추에이팅을 위한 메타데이터를 정의한다. 정의한 메타데이터를 통해 코드 생성 프로세스를 기술하고 마지막으로 MQTT 기반의 메시지 토픽 정의 방법을 기술한다.

3.1 제안 시스템 구조

그림 1은 제안 시스템의 구조이다. 구조는 크게 사물인터넷(IoT) 영역과 서버(Server) 영역으로 구성되어 있으며, IoT의 구성요소는 Device와 이를 관리하는 Device Manager로 구성되어 있다. Device는 실제로 센싱과 액추에이팅 동작하는 하드웨어에 해당한다. 이때 하드웨어는 아두이노에 해당하고 아두이노에 부착된 센서와 액추에이터의 상태는 시리얼 통신을 통해 Device Manager인 라즈베리파이로 전송하게 된다. Device Manager는 Device로부터 받은 센서 데이터를 MQTT 토픽 메시지를 통해 서버로 전송하거나 서버로부터 받은 액추에이팅 요청이나 코드 업로드 요청을 Device에 전달한다.

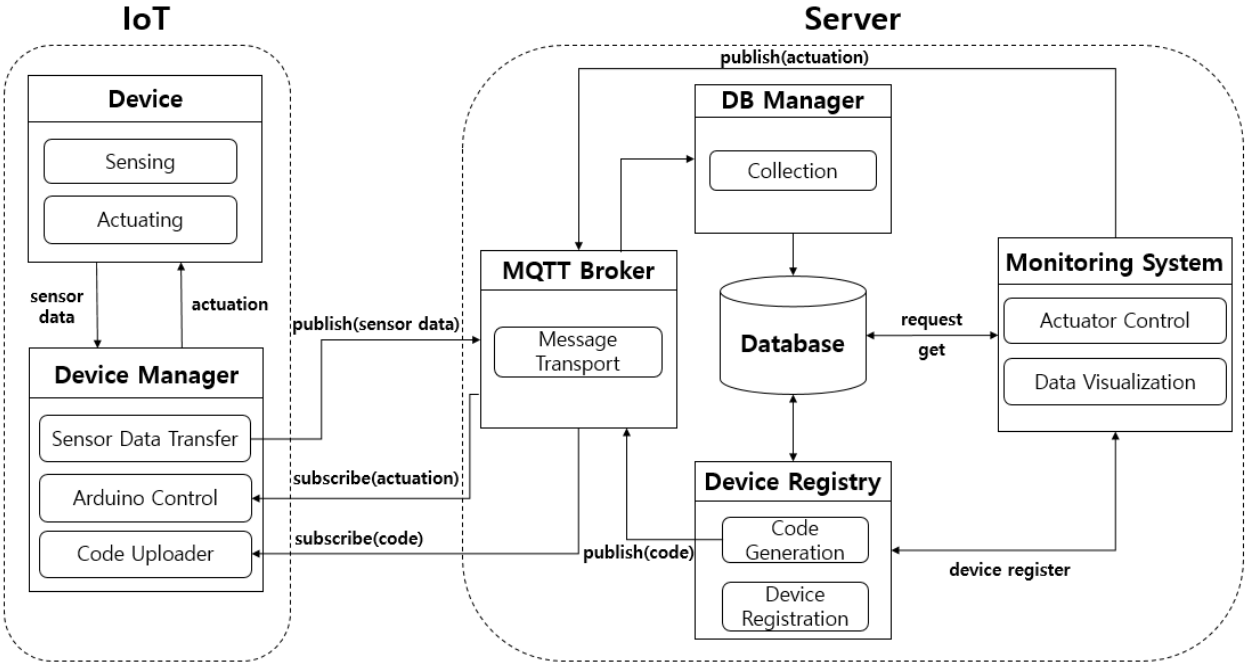

그림 1. 제안 시스템의 구조
Fig. 1. Architecture of proposed system

서버로부터 생성된 코드는 Arduino CLI를 이용하여 직접 아두이노에 코딩하지 않아도 업로드를 진행할 수 있도록 하였다. 서버는 MQTT 브로커(MQTT Broker), DB Manager, 데이터베이스, 모니터링 시스템, 디바이스 레지스트리로 구성된다.

MQTT 브로커는 IoT 디바이스와 웹 서버로부터 받는 메시지를 수집하고 해당 메시지를 subscribe하고 있는 장치에 전달해 주는 중계 역할을 한다[18]. 제안 시스템은 IoT 디바이스로부터 수집된 데이터를 데이터베이스에 저장하며 사용자로부터의 제어 요청이나 코드 생성 요청을 Device Manager로 전달한다.

DB Manager는 MQTT 브로커로부터 센서 데이터를 받게 되면 이를 데이터베이스로 저장하는 역할을 한다. 즉, 각 디바이스들의 토픽을 subscribe하고 있다가 센싱된 데이터를 받게 되면 디바이스를 구분하여 데이터베이스에 올바르게 저장한다.

모니터링 시스템은 IoT 디바이스의 센서 데이터를 보여주고 액추에이터 제어를 요청하는 웹을 의미한다. 사용자는 자신이 구현한 IoT 디바이스를 디바이스 레지스트리에 등록한 뒤, 코드 생성을 진행할 수 있고, 코드 생성을 마치고 나면 모니터링 시스템을 통해 직접 액추에이터를 제어하거나 센싱되는 모습을 확인할 수 있다.

Device Registry는 디바이스의 정보를 유연하게 관리하기 위해 키-값 형태의 메타데이터를 정의하는 플랫폼이다. 제안 시스템의 구현을 위해 기존 Device Registry의 메타데이터가 아닌 액추에이터 제어 및 코드 생성에 초점을 맞추어 메타데이터를 추가로 정의하였다. Device Registry에 디바이스 정보를 입력하면 메타데이터 기반의 코드 생성을 진행할 수 있다.

3.2 디바이스 레지스트리 데이터 구조

이 절에서는 다양한 종류의 IoT 디바이스의 정보를 등록 및 관리하기 위해 선행 연구인 디바이스 레지스트리를 활용한다. 디바이스 레지스트리는 다양한 특정 정보의 IoT 디바이스의 메타데이터를 유연하게 등록 및 관리할 수 있는 플랫폼이다. 본 논문에서는 해당 플랫폼을 통하여 코드 생성, 액추에이터 제어를 위한 메타데이터를 키-값 형태로 정의한다.

표 1은 디바이스 레지스트리의 특정 정보의 데이터 구조를 보인다. 하나의 IoT 디바이스에 대하여 group, key, value로 구성되어 보다 더 유연하게 정보를 관리할 수 있다는 장점이 있다.

표 1. 디바이스 레지스트리 특정 정보 구조
Table 1. Data structure for specific information

specific information	data type	description
group	integer	number that binds the same type of key and value
key	string	classification name of various specifications of the device
value	string	value corresponding to key

표 2. 특정 정보에서 정의된 IoT 디바이스의 예
Table 2. Example of IoT device defined in specific information

group	key	value
1	sensor	MQ7
1	min_value	5
1	max_value	70
1	unit_of_measure	ppm
1	data_type	integer
2	actuator	RGBLED
2	min_value	0
2	max_value	1
2	data_type	integer

표 2는 디바이스 레지스트리 메타데이터의 예시이다. 가스 센서(MQ-7)와 RGBLED가 부착된 디바이스인 것을 확인할 수 있으며 측정 단위, 측정값의 최소 및 최댓값을 확인할 수 있다.

3.3 코드 생성을 위한 메타데이터 정의

본 논문에서는 아두이노 코드 생성을 위한 메타데이터를 정의한다. 아두이노 코드 생성을 위한 메타데이터 정의는 아두이노 코드 생성을 자동화하고 기기의 다양한 설정을 유연하게 처리하는 역할이다.

따라서 본 논문에서는 선행 연구에서 정의한 디바이스 메타데이터 스펙 정보의 형태인 키-값 쌍의 형태로 코드 생성을 위한 메타데이터를 정의한다. 표 3은 본 논문에서 진행한 IoT 디바이스들의 코드 생성을 위한 메타데이터 정의 및 예시이다. 각 기기의 핀 번호와 명령어, 초기 설정값, 지연 시간 등을 포함한 메타데이터를 통해 해당 기기의 코드가 자동으로 생성될 수 있도록 설계되었다. 이러한 메타데이터는 키-값 쌍의 형식으로 구성되어 기기별로 고유한 설정 정보를 제공한다. 이를 통해 코드 생성 시 각 기기의 특성에 맞는 적절한 핀 설정, 명령 처리, 초기화 및 상태 전송 등이 가능하다.

3.4 아두이노 코드 생성 프롬프트 정의

본 논문에서는 아두이노 액추에이터 제어를 위한 코드 생성을 위해 OpenAI의 GPT-3.5 모델을 사용한다. GPT-3.5는 자연어 처리 기반의 언어 모델로 다양한 형태의 명령어를 이해하여 코드를 생성하는 특징을 지니며, 타 언어모델과 비교하였을 때 비교적 높은 pass@k 성능을 보인다[12]. pass@k는 모델이 여러 개의 코드 샘플을 생성했을 때 그중 하나라도 문제를 해결하는지를 기준으로 측정된다.

본 논문에서는 코드 생성을 위해 적절한 프롬프트 엔지니어링을 수행하여 메타데이터를 기반의 프롬프트를 활용한 아두이노 코드를 생성기법을 활용한다[19]. 메타데이터 기반 프롬프트는 기존의 프롬프트 기반 코드 생성 기법보다 코드 동작률 및 생성 시간에서 더 높은 성능을 보인다. 따라서 제안 시스템은 표 3에서 정의한 메타데이터를 활용하여 프롬프트를 정의한다.

표 3. 코드 생성을 위한 메타데이터 정의 및 예시
Table 3. Definition and examples of metadata for code generation

key	data type	description	example
actuator_pin	integer	describe the actuator pin number	9,10,11
sensor_pin	string	describe the pin number of the sensor	A0
commands	string	describe the actuator control instructions	ON, OFF, SET_COLOR
initial_value	string	describes the initial setting value of the actuator	RED
delay	integer	describes the delay time after the command is executed	5000(ms)
purpose	string	describes the purpose of the device	color control
library	string	describes the library required for arduino code	LiquidCrystal

표 4는 메타데이터에 기반한 프롬프트의 주요 단계들을 정의한 것이다. 각 단계는 명확한 요구사항 전달, 코드 제한 사항, 구체적인 변수 선언, 코드 구조 명시, 그리고 상태 전송 로직 명시로 구성된다. 이러한 단계들은 각 디바이스에 맞는 코드를 정확하게 생성할 수 있도록 설계되었으며, 불필요한 코드를 생성하지 않도록 하였고, 이를 통해 메타데이터 기반의 코드 생성을 구현한다.

3.5 MQTT 메시지 토픽 정의

본 논문에서는 아두이노, 웹, 라즈베리파이 간의 통신을 효과적으로 관리하기 위해 MQTT 메시지 프로토콜을 사용한다. 이를 위해 아두이노 디바이스의 센서 데이터 및 액추에이터 상태를 실시간으로 전송하고, 웹 또는 라즈베리파이로부터 제어 명령을 수신하기 위한 MQTT 메시지 토픽을 정의하였다.

각 디바이스의 고유한 device_id를 기반으로 한 메시지 토픽을 사용하여 통신 경로를 구분하며, 주로 센서 데이터 전송, 액추에이터 제어 명령 수신, 그리고 상태 응답 메시지로 구분된다. .

이러한 MQTT 메시지 토픽 구조는 각 디바이스에 맞는 데이터 전송과 제어가 가능하도록 설계되었으며, 제안 시스템에서는 ‘actuator/(device_name)’, ‘sensor/(device_name)’,‘compile/(device_name),‘upload/(device_name)’형식으로 토픽을 정의한다. 이때 device_name은 각 디바이스를 고유하게 구분할 수 있는 형태로 되어 있으며, actuator와 sensor를 나누어 해당 토픽이 센서에 관한 것인지, 제어 요청에 관한 것인지에 대해 구분한다. compile은 해당 코드의 컴파일을 라즈베리파이에게 요청하는 것이고, upload는 메타데이터 기반 자동으로 코드가 생성되면 해당 형식을 통해 코드를 전송하게 된다.

IV. 구현 결과

이 장에서는 제안 시스템의 구현 결과를 기술한다. 이를 위하여 IoT 디바이스를 실제로 개발하고 개발한 디바이스를 통해 메타데이터 정의가 유효한지, 생성된 코드가 올바르게 동작하는지를 보인다.

표 5는 제안 시스템의 서버 구현 환경을 보인다. GPT-3.5 API를 이용하여 메타데이터 기반 코드 생성 기능을 구현하였고, Mosquitto MQTT 브로커를 통해 아두이노와 라즈베리파이의 통신을 처리하였다.

평가에 사용될 IoT 디바이스는 RGBLED와 MQ-7으로 이루어진 아두이노 디바이스이다. 표 6은 구현된 IoT 디바이스들의 스펙이고 그림 2는 구현된 IoT 디바이스를 보인다.

표 4. 아두이노 코드 생성을 위한 프롬프트 정의 및 예시
Table 4. Definition and example prompts for generating arduino code

sequence	description	prompt examples
1	delivering clear requirements	check the features required for the device and write the code accordingly
		Generate Arduino code for the following device data: {"device_id":${deviceData.device_id}",device_name": "${deviceData.device_name}", ${Object.keys(metadata).map(key => `"${key}": "${metadata[key]}"`).join(',\n')}}
2	code restrictions	It does not include code block markers or additional descriptions (ex:```)
		Do not include code block markers like \`\`\` or comments
3	declaration of specific variables	Declare the variables you need and set the initial value to the metadata you received.
		Declare any other necessary variables based on the provided metadata.
4	code structure specification	Defines how each device operates within setup() and loop() functions.
		Initialize the pins and set up the serial communication in the \`setup()\` function.
5	state transfer logic specified	We implement a logic that periodically transmits the current state to Raspberry Pi.
		if the device has a metadata.sensor_pin, send the sensor data as well in the same format:-"{\\"device_id\\":\\"<device_id>\\",\\"actuator\\": \\"<currentActuatorState>\\", \\"sensor\\": \\"<currentSensorData>\\" }"

표 5. 서버 구현 환경
Table 5. Server implementation environment

features	specification
OS	Window 10(x64)
Device Registry	Node.js, Javascript
database	MySQL
DB Manager	Python
Device Manager	Python
MQTT Broker	mosquitto
monitoring system	Node.js, Javascript
code generate	GPT-3.5 API

표 6. 구현된 IoT 디바이스 스펙
Table 6. Implemented IoT device specifications

features	specification
platform	Arduino Uno, Raspberry Pi 4
sensor	MQ-7
actuator	RGBLED
communication	serial port

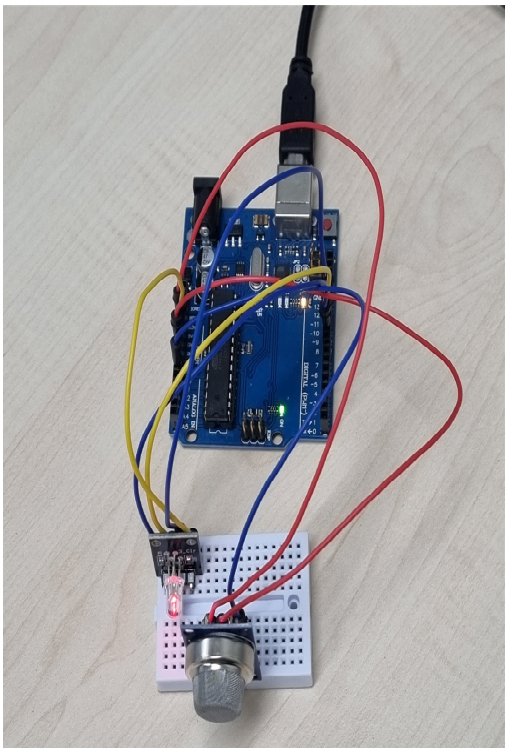

그림 2. IoT 디바이스의 구현
Fig. 2. Implementation of IoT device

구현한 디바이스의 정보 등록을 마치면 코드 생성을 진행할 수 있다. 코드 생성을 위해 메타데이터를 정의해야 하는데, 표 7은 본 논문에서 제시하는 코드 생성을 위한 메타데이터를 등록하는 과정이다.

표 7. 코드 생성을 위한 메타데이터
Table 7. Metadata for code generation

metadata(key)	value
Actuator pin	9,10,11
Sensor pin	A0
Purpose	Arduino detects CO levels with MQ7 and signals to RGBLED.
Commands	ON,SET_COLOR_RED,SET_COLOR_BLUE,SET_COLOR_GREEN,OFF
Initial value	red
Delay	5000

그림 3은 메타데이터 기반으로 생성된 코드를 보인다. 사용자는 생성된 코드를 컴파일 버튼을 통해 코드가 정상 작동하는지 확인 후 업로드 버튼을 통해 아두이노에 업로드 시킬 수 있다. 컴파일에 이상이 없다면, 업로드 버튼이 활성화되고 해당 버튼을 통해 아두이노에 생성된 코드를 업로드 시킬 수 있다. 최종적으로 업로드가 되면 그림 4와 같이 액추에이터 제어를 위해 개발한 모니터링 페이지에서 제어를 진행할 수 있다.

액추에이터 제어는 사용자가 버튼을 클릭하는 형식으로 버튼을 클릭하게 되면 MQTT 브로커에게 액추에이터 제어 요청 메시지를 발행하게 되고 Device Manager에게 전달되어 액추에이터의 상태가 변하게 된다. 그림 5는 메타데이터 기반의 생성된 코드를 업로드 시킨 디바이스의 시리얼 모니터이다.

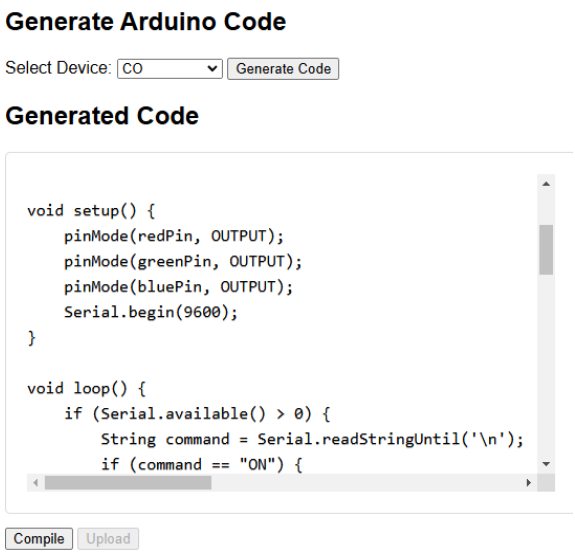

그림 3. 메타데이터 기반 생성된 코드 스크린샷
Fig. 3. Metadata-based generated code screenshots

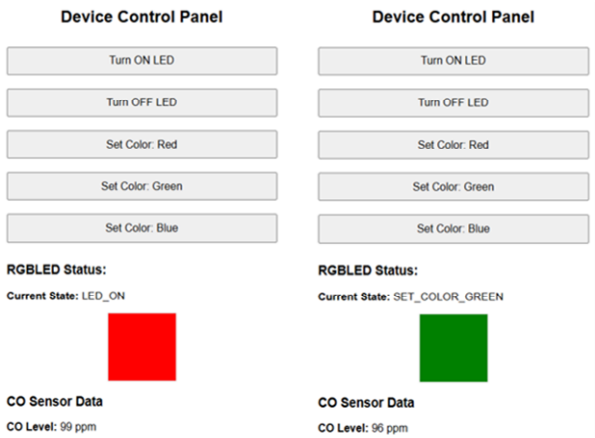

그림 4. RGBLED의 모니터링 및 제어 페이지
Fig. 4. RGBLED Monitoring and control page

```
16:22:16.177 -> { "device_id": "1", "actuator": "red", "sensor": "88" }
16:22:21.164 -> { "device_id": "1", "actuator": "red", "sensor": "87" }
16:22:26.174 -> { "device_id": "1", "actuator": "red", "sensor": "87" }
16:22:41.210 -> { "device_id": "1", "actuator": "red", "sensor": "87" }
16:22:46.222 -> { "device_id": "1", "actuator": "red", "sensor": "87" }
16:22:51.235 -> { "device_id": "1", "actuator": "red", "sensor": "87" }
16:22:56.262 -> { "device_id": "1", "actuator": "red", "sensor": "88" }
16:23:01.270 -> { "device_id": "1", "actuator": "red", "sensor": "88" }
16:23:06.296 -> { "device_id": "1", "actuator": "red", "sensor": "88" }
```

그림 5. RGBLED의 시리얼 모니터
Fig. 5. RGBLED serial monitor

이를 통해 본 논문에서 제안한 메타데이터 기반 아두이노 코드 생성을 통해 액추에이터를 제어할 수 있음을 확인한다. 표 8은 제안된 시스템으로 생성된 코드의 동작 여부 확인을 위하여 실제 구성한 센서 및 액추에이터에 대한 설명과 동작 여부를 보인다. 이를 통하여, 제안된 시스템이 다양한 센서 및 액추에이터에도 정상적으로 코드 생성 및 동작이 가능한 것을 확인할 수 있다.

Ⅴ. 시나리오

이 장은 제안 시스템이 여러 조합의 센서 및 액추에이터를 포함한 시나리오에서 코드가 예상대로 동작하는지 검증하고 이를 통해 시스템의 기능성과 다양한 IoT 환경에서 실용성을 입증한다.

본 논문의 시스템은 IoT 환경을 원하는 사용자들이 별도의 코딩 작업 없이 손쉽게 구축할 수 있도록 설계되었다. 예를 들어, 연구실에서 미세먼지 농도가 높아질 때 자동으로 환기팬을 작동시키는 작업을 사용자는 복잡한 프로그래밍 없이 단순히 메타데이터를 입력하여 IoT 환경을 구축할 수 있다.

IoT 환경은 다음과 같이 가정한다. 실내에서 미세먼지 농도를 지속적으로 모니터링하며, 미세먼지 수치가 일정 기준을 초과하면 공기 질 개선을 위한 자동화 조치를 할 수 있도록 설정한다. 이를 위해 미세먼지키트를 개발하고, 자동화 조치는 환기팬을 이용한다.

본 논문에서 시나리오는 그림 6과 같이 이루어진다. 사용자는 IoT 디바이스를 구축하기 위해 디바이스 레지스트리에 구축하고자 하는 디바이스의 정보를 등록하고 해당 디바이스의 코드를 생성할 수 있다. 생성된 코드는 MQTT를 통해 디바이스에 코드를 업로드 시키고, 업로드된 기기는 모니터링 시스템을 통해 제어할 수 있다. 먼저, 사용자는 사무실 내의 미세먼지키트를 설치하여 미세먼지 농도를 측정하고자 한다. 이때, 미세먼지키트는 라즈베리파이를 통해 연결되어 있으며, 시리얼 통신을 통해 원격 제어를 할 수 있도록 개발한다. 메타데이터를 관리하기 위해 개발한 디바이스 레지스트리에 사용자의 디바이스 정보 등록을 진행한다.

표 8. 실제 구성한 센서 및 액추에이터에 대한 설명과 동작 여부
Table 8. Description and operational status of the implemented sensors and actuators

num.	sensor	actuator	description	operation
1	-	Servo motor	- Rotate to the desired angle between 0 and 180 degrees.	V
2	-	LCD Display 1602	- When a string of 32 alphabetic characters is entered, the string is displayed.	V
3	MQ7	RGBLED	- The measurements from the MQ7 sensor are periodically stored in the database. - A string code for a color is entered, and the RGB LED displays the corresponding color (e.g., RED, GREEN, BLUE). - Three values between 0 and 255 for RGB color are entered, and the RGB LED displays the corresponding color.	V

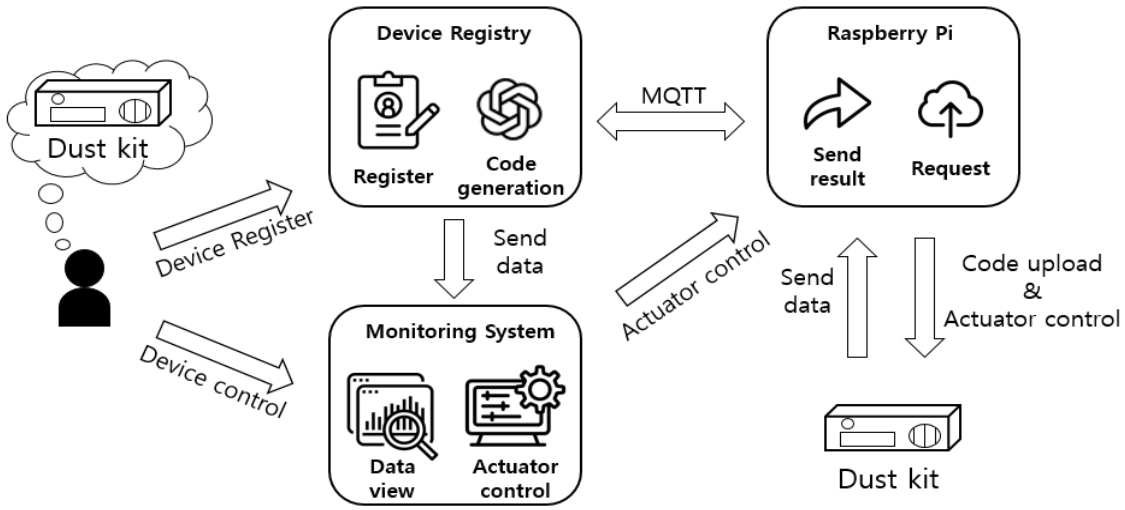

그림 6. 제안 시스템을 이용한 미세먼지키트 구축 방법
Fig. 6. How to build a dust kit using the proposed system

그림 7은 디바이스 레지스트리에 등록된 미세먼지키트와 환기팬에 대한 정보를 보여주는 스크린샷이다. 그림과 같이 등록이 완료되면 해당 디바이스의 DB 테이블이 자동으로 생성되고, 아두이노 코드 생성 리스트에 추가가 된다. 정보 등록을 완료한 뒤, 코드 생성 페이지에서 메타데이터를 기입하여 디바이스에 맞는 코드를 생성하고 즉시 업로드를 진행할 수 있다.

Item Detail

[5] Dust kit

modify back

Common Information

Attribute	Value
Item id	5
Registration time	2024-11-04 10:16:03
Model name	Dust kit
Device type	Dust kit
Manufacturer	DSEM Lab.
Category	actuator & sensor

Specific Information

Group	Key	Value
1	sensor	temperature
1	unit_of_measure	C
2	sensor	humidity
2	unit_of_measure	%
3	sensor	dust
3	unit_of_measure	ug/m3
4	actuator	rgb_led
5	actuator	fan

그림 7. 디바이스 레지스트리 정보 등록
Fig. 7. Register device registry information

그림 8은 코드 생성에 필요한 메타데이터를 입력받는 것을 보인다. 정의한 메타데이터를 기반으로 코드 생성을 진행하게 되고, 생성된 코드는 사용자의 컴파일 요청과 업로드 요청을 통해 라즈베리파이로 전송하게 된다.

Code Generation Metadata

Actuatorpin rgb_led:
9,10
Actuatorpin fan:
8
Actuatorpin lcdDisplay:
13,12,5,4,3,2
Sensorpin dht11:
7
Sensorpin dust:
A0, 11
Commands:
Fan_On, Fan_Off
Fan Initial Value:
OFF
Delay
5000
Purpose
Collect and manage air quality data
Library
DHT11, LiquidCrstal
Threshold:
70

그림 8. 미세먼지키트 코드 생성을 위한 메타데이터 정의
Fig. 8. Metadata definition for dust kit code generation

컴파일 버튼을 누르게 되면 라즈베리파이로 전송하여 컴파일 여부를 확인하고 반환 값에 따라 업로드 버튼이 활성화된다. 업로드 버튼 클릭 후 모니터링 시스템을 통해 액추에이터를 제어할 수 있다. 모니터링 시스템은 사용자가 설정한 명령에 따라 제어를 할 수 있게 개발하였다.

현재 설정된 명령은 'Fan_On'과 'Fan_Off'로 모니터링 시스템에서 버튼으로 구현하였다. 해당 버튼을 누르게 되면 모니터링 시스템은 MQTT 브로커로 'Actuator/Fan_On', 'Actuator/Fan_Off'와 같은 토픽을 발행하고, 라즈베리파이는 해당 토픽을 구독하여 아두이노에 명령을 내려 제어할 수 있게 한다.

그림 9와 10은 미세먼지키트의 최종 결과화면을 보인다. 현재 수집되고 있는 데이터를 실시간으로 볼 수 있으며 명령을 보내 제어를 할 수 있으며 그림 11과 같이 미세먼지키트 데이터가 수집된다.

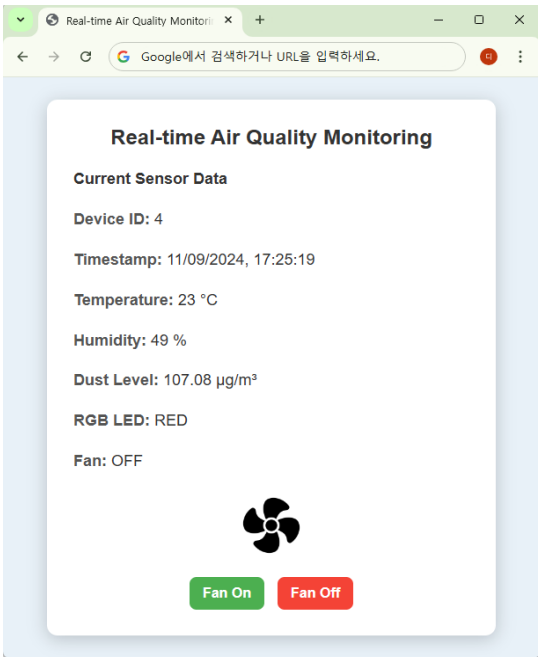

그림 9. 미세먼지키트의 모니터링 시스템 스크린샷
Fig. 9. Screenshot of dust kit monitoring system

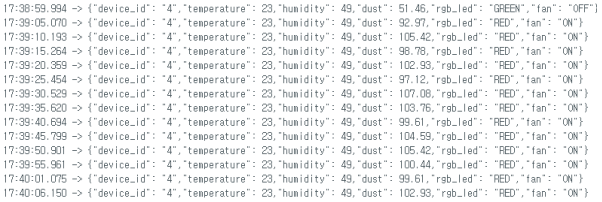

그림 10. 미세먼지키트의 시리얼 모니터
Fig. 10. Serial monitor of a dust kit

id	timestamp	temperature	humidity	dust	rgb_led	fan
1	2024-11-05 16:22:16	21	38	87.99	ON	ON
2	2024-11-05 16:22:21	21	38	98.78	ON	ON
3	2024-11-05 16:22:26	21	38	85.5	ON	ON
4	2024-11-05 16:22:31	21	38	94.63	ON	ON
5	2024-11-05 16:22:36	21	38	95.46	ON	ON
6	2024-11-05 16:22:42	21	38	86.33	ON	ON
7	2024-11-05 16:22:47	21	38	69.73	ON	ON

그림 11. 미세먼지키트의 데이터 수집 테이블
Fig. 11. Data collection table of a dust kit

이를 통해 본 논문에서 제안한 시스템이 실제 미세먼지키트를 통해 정상 동작함을 확인하며, 사용자는 별도의 프로그래밍 없이도 메타데이터 입력만으로 원하는 제어와 모니터링 기능을 구현할 수 있으며, 다양한 IoT 환경에서 유연하게 적용 가능하다는 것을 확인한다.

Ⅵ. 결 론

본 논문에서는 메타데이터 기반의 액추에이터 제어를 위한 아두이노 코드 생성 시스템을 제안하고, 이를 통해 IoT 환경에서 센서 및 액추에이터를 손쉽게 제어할 수 있는 시스템을 구현하였다. 제안 시스템은 메타데이터를 기반으로 필요한 코드를 자동으로 생성하여, 비전문가도 별도의 프로그래밍 지식 없이 IoT 장치들을 구성하고 제어할 수 있도록 설계하였다. 시나리오를 통해 제안 시스템이 생성한 코드가 사용자의 의도대로 생성이 되고 실제로 동작하는지에 대해 검증하였으며, 이를 통해 제안 시스템이 어떻게 신뢰성 있고 유연하게 동작하는지를 확인하였다.

향후 연구에서는 다양한 IoT 디바이스의 지원을 확대하고 실시간 오류 검출 및 자동 수정 기능을 추가하여 시스템의 안정성과 효율성을 높일 수 있으며, 메타데이터 기반의 코드 생성 신뢰성 향상을 위한 연구가 추가로 진행되어야 한다.

References

[1] C. Stolojescu-Crisan, C. Crisan, and B.-P. Butunoi, "An IoT-based smart home automation system", Sensors, Vol. 21, No. 11, pp. 3784, May 2021. https://doi.org/10.3390/s21113784.

[2] P. K. Tripathy, A. K. Tripathy, A. Agarwal, and

S. P. Mohanty, "MyGreen: An IoT-enabled smart greenhouse for sustainable agriculture", IEEE Consumer Electronics Magazine, Vol. 10, No. 4, pp. 57-62, Jul. 2021. https://doi.org/10.3390/app12073396.

[3] S.-O. Kim and E.-J. Kwon, "IoT-based taking medicine automation system", Journal of the Korea Society of Computer and Information, Vol. 26, No. 4, pp. 161-168, Apr. 2021. https://doi.org/10.9708/jksci.2021.26.04.161.

[4] L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "A survey on IoT platforms: Communication, security, and privacy perspectives", Computer Networks, Vol. 192, pp. 108040, Jun. 2021. https://doi.org/10.1016/j.comnet.2021.108040.

[5] F. F. Xu, B. Vasilescu, and G. Neubig, "In-IDE code generation from natural language: Promise and challenges", ACM Transactions on Software Engineering and Methodology, Vol. 31, No. 2, pp. 1-47, Mar. 2022. https://doi.org/10.1145/3487569.

[6] X. Jiang, Y. Dong, L. Wang, Z. Fang, Q. Shang, G. Li, Z. Jin, and W. Jiao, "Self-planning code generation with large language models", ACM Transactions on Software Engineering and Methodology, Vol. 33, No. 7, pp. 1-30, Sep. 2024. https://doi.org/10.1145/3672456.

[7] Y. Dong, X. Jiang, Z. Jin, and G. Li, "Self-collaboration code generation via ChatGPT", ACM Transactions on Software Engineering and Methodology, Vol. 33, No. 7, pp. 1-38, Sep. 2024. https://doi.org/10.1145/3672459.

[8] K. Lee and M. Kim, "Trends in generative AI technology and applications in the content industry", OSIA Standards & Technology Review Journal, Vol. 37, No. 2, pp. 24-29, 2024.

[9] C. Liu, X. Bao, H. Zhang, N. Zhang, H. Hu, X. Zhang, and M. Yan, "Improving ChatGPT prompt for code generation", arXiv preprint, arXiv:2305.08360, May 2023. https://doi.org/10.48550/arXiv.2305.08360.

[10] W. Jang, D. Jung, and S. Lee, "MQTT-based IoT object control technique for enhancing mobility of objects", Journal of Korean Institute of Information Technology, Vol. 20, No. 3, pp. 107-119, Mar. 2022. http://dx.doi.org/10.14801/jkiit.2022.20.3.107.

[11] S. Lee, D. Jung, Y. Seo, and S. Lee, "Development of a device registry for flexible device information management in IoT platforms", in Proceedings of the KIIT Conference, Cheongju, Korea, pp. 323-327, Oct. 2020.

[12] S. Yeo, Y. Ma, H. Jun, T. Kim, and S. C. Kim, "Survey on recent large language models for code generation", KIISE Transactions on Computing Practices, Vol. 30, No. 8, pp. 372-381, Aug. 2024. https://doi.org/10.5626/KTCP.2024.30.8.372.

[13] S. Kim, "Developing Code Generation Prompts for Programming Education with Generative AI", The Journal of Korean Association of Computer Education, Vol. 26, No. 5, pp. 107-117, 2023. https://doi.org/10.32431/kace.2023.26.5.009.

[14] J. Shin and J. Nam, "A Survey of Automatic Code Generation from Natural Language", Journal of Information Processing Systems, Vol. 17, No. 3, pp. 537-555, Jun. 2021. https://doi.org/10.3745/JIPS.04.0216.

[15] I. Hwang and B. Kim, "Implementation of a GUI-based Physical Computing System for Beginners Using Arduino", Journal of the Korean Society of Information Technology, Vol. 20, No. 8, pp. 29-39, Aug. 2022. https://doi.org/10.14801/jkiit.2022.20.8.29.

[16] A. C. F. D. Silva, et al., "MBP: Not Just an IoT Platform", 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 1-3, Mar. 2020. https://doi.org/10.1109/PerComWorkshops48775.2020.9156156.

[17] S. Yang, S. Lee, and H. Jeong, "IoT Platform

System for Electric Fire Prediction and Prevention", Journal of the Korea Institute of Information and Communication Engineering, Vol. 26, No. 2, pp. 223-229, Feb. 2022. http://doi.org/10.6109/jkiice.2022.26.2.223.

[18] O. Deschambault, A. Gherbi, and C. Légaré, "Efficient implementation of the MQTT protocol for embedded systems", Journal of Information Processing Systems, Vol. 13, No. 1, pp. 26-39, Feb. 2017. https://doi.org/10.3745/JIPS.04.0028.

[19] J. Kim, D. Kim, S. Lee, M. Lee, U. Lee, and S. Lee, "A metadata-based Arduino code generation technique", in Proceedings of the 2024 KIIT Autumn Conference and Undergraduate Paper Competition, Jeju, Korea, pp. 1123-1125, Nov. 2024.

저자소개

김 재 엽 (Jaeyeob Kim)

2019년 3월 ~ 현재 : 국립군산대학교 소프트웨어학과 학부과정

관심분야 : 사물인터넷, IoT, LLM, 데이터 분석

이 석 훈 (Sukhoon Lee)

2009년 2월 : 고려대학교 전자및정보공학부(학사)

2011년 2월 : 고려대학교 컴퓨터·전파통신공학과(공학석사)

2016년 2월 : 고려대학교 컴퓨터·전파통신공학과(공학박사)

2016년 3월 ~ 2017년 3월 : 아주대학교 의료정보학과 연구강사

2017년 4월 ~ 현재 : 국립군산대학교 소프트웨어학과 부교수

관심분야 : 사물인터넷, 메타데이터 레지스트리, 데이터 품질, 연합 학습