Regular paper | Journal of the KIECS. pp. 803-808, vol. 18, no. 5, Oct. 31. 2023, t. 121, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2023.18.5.803

IoT 환경을 위한 Local WAS에서 디바이스 이질성을 줄이는 독립적인 Firmware 설계

이경호* · 문은아**

Independent Firmware Design to Reduce Device Heterogeneity in LAN WAS for IoT Environment

Kyung-Ho Lee* · Eun-Ah Moon**

요 약

IoT 산업은 매년 기록적인 성장률을 기록하며 성장하고 있으나 IoT 플랫폼을 개발하는데 앞서 개발자들은 보안, 데이터 저장, 디바이스간 이질성 등의 현실적인 문제들에 직면하게 된다. 특히 디바이스간 이질성은 네트워크 유형과 프로토콜로 발생하는데, 다비이스 Firmware를 변경하거나 경우에 따라서는 여러 개의 IoT 플랫폼을 사용해야 한다. 또한 무분별한 IoT 디바이스가 넘쳐나면서 중복된 센싱으로 데이터가 낭비되기도 한다. 본 논문에서는 Local WAS가 MQTT 프로토콜을 사용하는 IoT 플랫폼 환경에서 디바이스간 이질성 해결을 위한 디바이스 독립적인 Firmware 설계를 제안하고자 한다.

ABSTRACT

The IoT industry is growing at a record growth rate every year, but developers face practical problems such as security, data storage, and heterogeneity between devices before developing an IoT platform. In particular, heterogeneity between devices occurs due to network type and protocol, and device firmware must be changed or multiple IoT platforms must be used in some cases. In addition, data is wasted due to redundant sensing due to the overflow of indiscriminate IoT devices. In this paper, we propose a device-independent firmware design to solve the heterogeneity between devices in the IoT platform environment where Local WAS uses the MQTT protocol.

키워드

MQTT, Firmware, IoT, Local, WAS, Heterogeneity, Platform, Protocol
MQTT, 펌웨어, 사물인터넷, 로컬, WAS, 이질성, 플랫폼, 프로토콜

Ⅰ. 서 론

IoT는 4차 산업의 다양한 기술과 결합하여 가정, 교통, 의료 등 일상생활의 전반에 일부분으로 자리 잡았다. 한국 ICD에 의하면 IoT의 시장 규모는 꾸준히 상승세를 유지하고 있으며 2025년 국내 IoT 시장은 38조 1870억 원을 기록할 것으로 예상된다[1][2][3]. IoT 플랫폼은 다양한 디바이스들을 연결하여 데이터

* 에이치엠테크놀로지 연구소장(rfsoft@naver.com)
** 교신저자 : 조선이공대학교 전기과
· 접 수 일 : 2023. 08. 02
· 수정완료일 : 2023. 09. 06
· 게재확정일 : 2023. 10. 17
· Received : Aug. 02, 2023, Revised : Sep. 06, 2023, Accepted : Oct. 17, 2023
· Corresponding Author : Eun-Ah Moon
Dept. Department of Electricity, Chosun College of Science & Technology
Email : eamoon@cst.ac.kr

를 수집하고 관리할 뿐 아니라, 다비이스로부터 수집된 데이터를 실시간으로 모니터링하거나 제어하는 기술을 지원해야한다. 이때 두 가지 문제점을 가지는데 첫째는 기존의 IoT 플랫폼들은 데이터의 유형 및 프로토콜 등이 상이하여 플랫폼 간 디바이스를 모니터링 하거나 제어하지 못하는 문제가 있기 때문에 IoT를 지원하는 디바이스마다 스마트폰에 여러 IoT 앱을 사용해야 하는 번거로움이 있다[4][5]. 둘째는 디바이스마다 IoT 환경에서 동작하기 위해 모니터링과 제어를 자체적으로 동작 하다보니 중복되는 데이터가 발생하게 된다. 또한 정확한 모니터링과 제어를 위해서는 모니터링한 데이터의 센서 위치가 중요한데, 제어하는 곳과 센서의 위치가 다를 경우 정확한 모니터링이 불가능하다. 이를 해결하기 위해서는 디바이스간 이질성을 최소화하여 하나의 IoT 플랫폼에서 네트워크 유형과 프로토콜을 관리하고 규격화된 디바이스 제어가 필요하다. 본 논문에서는 디바이스의 이질성을 줄이기 위한 독립적인 Firmware 설계와 이를 바탕으로 디바이스들의 데이터를 통합 관리할 수 있는 Local WAS 설계를 제안하고자 한다.

II. 관련연구

2.1 IoT 플랫폼의 프로토콜

Amazon의 AWS IoT는 MQTT, HTTPS, MQTT 오버 WSS, LoRaWAN을 포함해 소비자가 선호하는 통신 프로토콜을 선택할 수 있고 플랫폼에 전달되는 데이터는 Amazon Cloud Watch, Amazon DynamoDB등 AWS에서 서비스하는 DB와 연계하여 활용할 수 있다[6][7]. Google의 Cloud IoT는 MQTT 및 HTTP 프로토콜을 지원하며, Google 지도를 활용한 위치기반의 IoT 솔루션을 제공하고, TensorFlow를 이용하여 플랫폼에 전달된 데이터를 가공할 수 있다. Microsoft의 Azure IoT는 end-to-end 보안 솔루션과 Azure Sphere를 적용하여 한층 더 강화된 MCU IoT를 구현할 수 있지만 2023년 8월부로 서비스 종료를 선언했다[6][8]. 네이버 클라우드 플랫폼은 MQTT 기반으로 인증서를 통해 증명된 IoT 디바이스만 연결할 수 있으며 데이터 전송 구간은 Trasport Layer Security방식으로 암호화되어 통신한다.

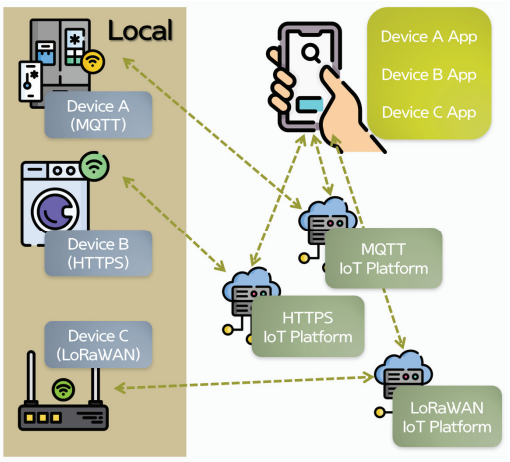

그림 1. IoT 플랫폼 네트워크 형태
Fig. 1 IoT platform network type

그림 1과 같이 IoT 플랫폼에 따라 지원해주는 프로토콜이 다르기 때문에 추가되는 다바이스가 IoT 플랫폼에서 지원하지 않는 프로토콜을 사용한다면 여러 IoT 플랫폼을 사용하던지 프로토콜 변환해주는 모듈이 필요하다. 전자의 경우가 현실적이나 디바이스의 모니터링이 분산되는 단점이 있어 그림 2에서와 같이 디바이스의 프로토콜이 서로 다르더라도 하나의 IoT 플랫폼에서 작동할 수 있는 프로토콜 변환 장치가 필요하다.

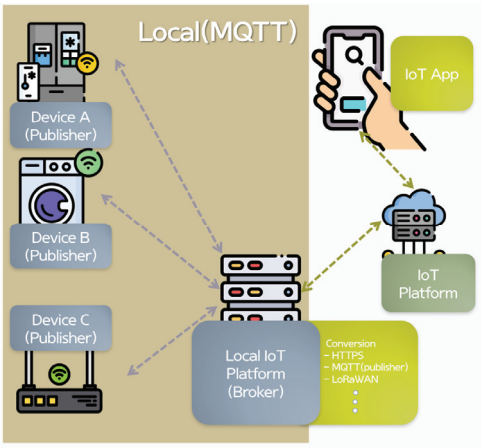

그림 2. Local WAS을 이용한 예시
Fig. 2 Example using Local IoT platform

2.2 디바이스간 이질성

대부분의 IoT 플랫폼은 유무선 통신이 가능한 디바이스가 필수 요소이다. 사용자는 엣지 디바이스를 탐색하고 디바이스가 제공하는 서비스를 선택하여 이용한다. 하지만 이러한 방식은 IoT 디바이스가 많아질 경우 사용자 경험을 해칠 수 있고, 수많은 디바이스 중에 사용자가 원하는 서비스가 있는지 일일이 찾아서 서비스를 이용하는 방식으로 시간이 많이 소요되고 사용자에게 최적화된 디바이스의 서비스를 찾는데 어려움이 있다[9]. 또한, IoT의 보안 정책 및 체계가 구축되지 않은 상태에서 IoT와 관련한 디바이스와 서비스가 지속적으로 출시되고 있다. 그에 따라, IoT 디바이스 및 서비스와 관련한 취약점이 계속적으로 발견되고, 늘어나고 있다. 취약점으로는 인증정보 보호, 운영체제 명령어 삽입 공격, 버퍼 오버플로우 공격, 불필요한 서비스 비활성화, 알려진 취약점 제거로 구분된다[10]. 이를 해결하기 위해 서비스 기반 IoT 플랫폼 중 SoPIoT 플랫폼이 연구되고 있다. SoPIoT 플랫폼은 디바이스를 관리해 통신 프로토콜 지원, 써드 파티 플랫폼 지원, 디바이스 군집화 기능을 제공한다[9].

IoT 디바이스는 모니터링하기 위한 센싱, 모니터링된 데이터 처리하고 제어하게 되는데 디바이스마다 센싱, 데이터 처리, 제어가 구현되어 지면서 디바이스들 간에 중복되는 센싱 정보와 데이터 처리가 발생한다. 특히 제어와 센싱의 위치가 항상 동일 할 수 없고 디바이스에 따라 IoT 플랫폼이 다를 경우 데이터가 서로 공유될 수 없다. 그래서 디바이스간 중복되는 데이터를 최소화하거나 데이터를 서로 공유하여 통합 관리할 필요가 있다.

디바이스 Firmware는 하드웨어가 설계되면 고유의 Firmware를 가지게 되어 디바이스간 이질성이 발생해 IoT 플랫폼을 구현하는 큰 장애물로 작용한다. 또한 변화하는 요구사항에 업데이트 매커니즘은 복잡하여 수동으로 접근하여 수정, 보완되기 때문에 오류가 발생하기 쉽다. 이를 해결하기 위해서 Firmware의 동작을 MCU의 레지스터별로 구분하여 동작하게 설계하고 동일한 Firmware로 동작하게 한다면 디바이스의 이질성을 줄여줄 수 있다.

III. 디바이스 독립적인 Firmware와 Local WAS 설계

3.1 디바이스 독립적인 Firmware

디바이스는 개발자의 목적이나 전자 부품에 따라 회로도가 다르고 센서 모듈의 경우 통신 규격과 통신 패킷이 다르기 때문에 디바이스마다 고유한 Firmware를 가질 수밖에 없다. 또, IoT를 지원하는 경우 네트워크 유형이나 프로토콜도 다양해 디바이스 간 이질성이 발생하며 디바이스 Firmware에 따라 여러 IoT 플랫폼을 사용해야 하는 문제점도 발생한다. 그러나 회로도가 다르거나 통신방식을 이용한 센서 모듈은 Fimware로 제어가 되어야 하기 때문에 마이크로프로세서의 I/O 포트에 연결되는 점을 활용한다면 종속적인 Fimware를 디바이스 독립적인 Firmware로 설계가 가능하고 제어를 Local WAS에서 한다면 다바이스간 이질성을 해결할 수 있다. 디바이스 독립적인 Firmware 설계를 가능하게 하는 세 가지 특징은 첫째, 마이크로프로세서는 제조사가 서로 다르더라도 c, c++언어로 제어가 가능하다. 둘째, I/O 제어, 인터럽트, 타이머/카운터, ADC, SPI, USART 등의 기능은 표준화가 되어 있다. 셋째, 마이크로프로세서에서 사용되는 레지스터들을 기능에 맞게 규격화가 가능하다. 그리고 Firmware는 폴링 방식에 기반을 두고 동작하므로 동작을 세분화한 단위 동작과 단위 동작을 연결하여 동작하는 구분 동작으로 나눌 수 있다.

3.1.1 구분 동작과 단위 동작

단순히 레지스터 제어를 하는 형태로 설계가 된다면 IoT 디바이스는 Local WAS에 의존적으로 되어 플랫폼이 없이 사용되기 힘들게 된다. 그렇다면 Local WAS에 의존적이지 않고 디바이스 독립적인 Firmware를 설계하기 위해서는 레지스터 제어 형태가 아닌 마이크로프로세서에서 제공되는 기능을 중심으로 레지스터를 묶어서 제어하는 것이 더 나은 방법이다. 다만 제조사나 마이크로프로세서 모델에 따라 레지스터 제어 형태가 조금씩 달라지기 때문에 이 부분을 개별적으로 규격화해 주는 작업이 필요하다. 제조사별로 마이크로프로세서 모델에 사용되는 레지스터들의 형태는 비슷한 구조로 되어 있기 때문에 c++을 지원하면 객체지향프로그래밍(OOP) 형태로 추상

화하여 제작이 가능하고 c언어로는 구조체로 설계가 가능하다. 그럼 독립적인 Firmware를 설계하기 위해 구분 동작과 단위 동작을 구분 짓는 방법으로는 플로우 방식을 이용하여 구분 동작을 구별하고 기능에 따라 여러 레지스터 비트 설정을 규격화하여 단위 동작을 구별해 주면 되는데, 구분 동작과 단위 동작은 표 1과 표 2로 분류될 수 있다.

표 1. 플로우를 기준으로 한 구분 동작
Table 1. Category action based on flow

Flow Method	Polling	device initialize
		main repeat section sub repeat section
	Interrupt	Interrupt service routines such as reset, external interrupt, timer/counter, serial communication, etc.

표 2. 기능을 기준으로 한 단위 동작
Table 2. Unit action based on function

I/O control	input
	output
Timer/Counter	common
	CTC
	PWM
serial communication	USART
	SPI
	TWI
ADC	
Direct register control	
delay	

이때, 구분 동작과 단위 동작은 몇 가지 규칙이 존재하는데 아래와 같다.

① 구분 동작을 기준으로 단위 동작을 연결하여야 한다.

② 구분 동작의 인터럽트는 단위동작의 타이머/카운터, 직렬 통신, ADC, 레지스터 직접제어, 지연의 사용을 지양하거나 회피해야 한다.

③ 구분 동작의 폴링은 보조 반복 구간에서는 반드시 탈출하여 주 반복 구간으로 돌아가야 한다.

④ 구분 동작에 따른 단위 동작을 RAM과 ROM에 저장하여 실행되기 때문에 마이크로프로세서의 RAM과 ROM의 크기에 동작이 구속받는다. 따라서 초기화 동작은 반드시 ROM에 저장한다.

디바이스 독립적인 Firmware를 사용할 경우 모니터링과 제어는 구분 동작과 단위 동작을 이용해서 충분히 제어가 가능하다. 그러나 데이터를 처리하는 부분이 복잡해질수록 디바이스 Firmware 내에서 처리가 힘들어진다. 구분 동작과 단위 동작을 상황에 따라 RAM이나 ROM에 저장하여 사용하기 때문에 디바이스 종속적인 Firmware보다는 리소스 관리에 신경을 써야하는 단점이 발생한다. 하드웨어 개발 단계에서 리소스 부분을 고려하여 마이크로프로세서 모델 선정해줘야 한다.

3.1.2 MQTT 프로토콜

IoT 디바이스들이 디바이스 간 이질성이 발생하는 이유 중 하나는 IoT 플랫폼마다 지원하는 프로토콜이 다르기 때문이다. Local WAS 내에서는 자유롭게 프로토콜을 구현해도 상관은 없으나 현재 서비스되고 있는 IoT 플랫폼에서 MQTT 프로토콜 방식을 다수 제공해주고 있어 디바이스 또한 MQTT 프로토콜을 제공하는 제품군이 증가할 것으로 예상할 수 있다. 따라서, Local에서는 MQTT 프로토콜을 사용해 IoT 디바이스는 푸블리셔(Publisher), Local WAS은 브로커(Broker)로 설계하는 것이 바람직하다.

3.2 Local WAS

디바이스 독립적인 Firmware는 플로우 방식에 따라 구분 동작, 기능에 따라 단위 동작으로 구분되기 때문에 복잡한 데이터 처리가 힘들다. 대부분의 센서는 간단한 데이터 처리 해결이 가능하나 미적분이 필요한 데이터 처리의 경우, Raw 데이터를 전달받아 Local WAS에서 Calibration 처리가 되어야 한다. Local WAS은 WAN 환경에도 연결이 되기 때문에 자체적인 업데이트가 가능하고 필요한 센서의 Raw 데이터를 처리하는 부분이나 제어 모듈에 따른 동작 알고리즘은 IoT 서비스에서 라이브러리 형태로 다운로드 받아 사용할 수 있게 설계해야 한다. 또한 디바이스 독립적인 Firmware의 동작이 복잡해져 디바이

스의 RAM, ROM에서 처리가 안될 경우 Local WAS에서 상황에 따라 구분 동작을 선택적으로 처리할 수 있게 설계해야 한다.

Local에서 통신은 MQTT 프로토콜로 통합하여 Local WAS에서 데이터를 관리하여 기존의 IoT 디바이스와 호환성을 확보해야한다. 이 때, Local WAS은 브로커로 작동하게 되고 푸블리셔인 디바이스들에게 전달받아 외부의 IoT 플랫폼에서 지원하는 프로토콜에 맞게 변환하여 전달할 수 있게 설계되어야 한다.

3.3 개발 환경 구축

설계된 독립적인 Firmware를 구현하기 위해서는 공유기, Local WAS용 임베디드 모듈, 독립적인 Fimware가 다운로드 된 디바이스가 필요하다.

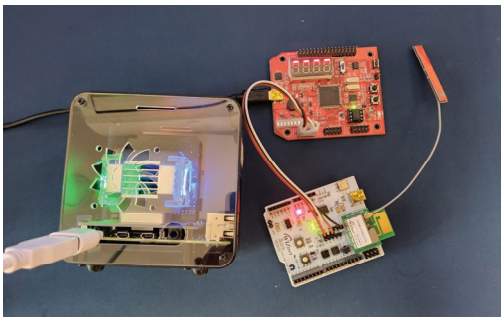

그림 3. 하드웨어 구성
Fig. 3 Hardware configuration

공유기는 일반 가정에서 제공되는 SK 공유기를 사용하고 그림 3과 같이, Local WAS용 임베디드 모듈은 RaspberryPi 4, 디바이스는 JKit-128-1에 WizFi250의 WiFi 모듈을 연결하였다. 그림 4는 Local WAS에 Spring boot를 이용해 MVC패턴을 사용하여 공유기에 연결된 PC에서 웹 브라이저로 접속하여 JKit-128-1(독립적인 Fimware)을 제어할 수 있게 구축한 화면이다.

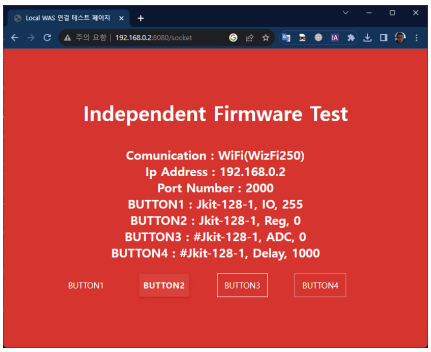

그림 4. PC에서 접속한 Local WAS
Fig. 4 Local WAS accessed from PC

Ⅳ. 결론 및 추후 과제

디바이스 독립적인 Firmware는 마이크로프로세서의 규격화된 동작을 하므로 IoT 디바이스가 서로 다르더라도 센싱과 제어를 간단히 구현할 수 있다는 장점을 가지고 있으나 복잡한 데이터를 처리하는 경우 Raw 데이터를 Local WAS에서 처리해야 하는 단점 또한 있다. 그러나 디바이스의 동작이 하나의 Firmware로 호환이 가능하고 Local망에 연결되어 MQTT 프로토콜을 사용함으로써 데이터를 공유하고 제어될 수 있어 디바이스간 이질성을 제거해준다. 그리고 한 디바이스에 센싱과 제어를 처리하면서 오는 자원의 낭비를 최소화 할 수 있으며, 센싱과 제어의 위치를 여러 디바이스로 분리하더라도 Local WAS에서 관리가 가능해 정확한 모니터링과 제어가 가능하다. 또한 서로 다른 프로토콜을 사용하는 IoT 플랫폼에서도 Local WAS에서 디바이스의 데이터를 관리하여 필요한 프로토콜에 맞게 변환해 주기 때문에 호환성을 높일 수 있고, WAN에 연결되어 있어 업데이트가 가능하기 때문에 디바이스에 연결된 센서 및 제어 모듈을 라이브러리 형태로 제작 확장할 수 있다. 이렇게 디바이스 독립적인 Firmware는 IoT 환경에서 디바이스간 이질성을 제거하는 한편 IoT 디바이스 개발을 단축시키고 중복되는 데이터를 줄일 수 있을 것이다. 추후 연구로는 디바이스 독립적인 Firmware의 구분 동작과 단위 동작을 구현하고 RAM과 ROM에 규격화된 데이터를 저장할 수 있는 코덱 설계가 필요할 것으로 보인다.

References

[1] IDC Korea, “Wordwide Semiannual Internet of Things Spending Guide,” *Report,* Nov. 2021.

[2] D. Kang and J. Lim, “Network Security Protocol Performance Analysis in IoT Evironment,” *Journal of The Korea Institute of Information Security & Cryptology*, vol. 32, no. 5, 2022, pp. 955-963.

[3] KIAT, “Domestic and international IoT industry trends,” *Report,* Apr. 2020.

[4] A. Deohate and D. Rojatkar, “Middleware Challenges and Platform for IoT-A Survey,” *2021 5th International Conference on Trends in Electronics and Information, Tirunelveli*, India, Jun. 2021, pp. 463-467.

[5] K. Lee, W. Jang, B. Yang, S. Lee, D. Jeong, and S. Lee, “Ontology-based IoT Platform for Improving Interoperability,” *Proceedings of the Korean Institute of Information and Commucation Sciences Conference*, Gunsan, South Korea, Oct. 2021, pp. 6-8.

[6] S. Lee, “A model-driven framework for IoT platform device application,” Master’s Thesis, *Hanyang University,* Feb. 2021.

[7] AWS, "AWS IoT Core Developer Guide,” *Report,* 2020.

[8] D. Kim, H. La, and S. Kim, "A Framework for Effectively Managing Heterogeneity of IoT Devices,” *Journal of Korean Institute of Information Scientists and Engineers, Software and Application*, vol. 41, no. 5, 2014, pp. 353-366.

[9] T. Son, “Hierarchical device management techniques in service-based IoT platforms,” Master’s Thesis, *Seoul National University,* Aug. 2022.

[10] Y. Cho, “A Study on System Development to Impreove IoT Firmware Vulnerability Analysis Efficiency,” *Korean Journal of Industrial Security*, vol. 12, no. 1, 2022, pp. 31-50.

저자 소개

이경호(Kyung-Ho Lee)

2003년 세한대학교 메카트로닉스학과 졸업(공학사)
2013년 호남대학교 대학원 소프트웨어 공학과 졸업(공학석사)
2013년 호남대학교 대학원 소프트웨어 공학과 졸업(공학석사)
2014년 ㈜엠텍정보 연구소장
2021년 조선이공대학교 전기과 교수
2023년 에이치엠테크놀로지 연구소장
※ 관심분야 : IoT, 센서 네트워크

문은아(Eun-Ah Moon)

1991년 조선대학교 전기공학과 졸업(공학사)
2005년 조선대학교 대학원 전기공학과 졸업(공학석사)
2014년 조선대학교 대학원 전기공학과 졸업(공학박사)
2014년 조선이공대학교 전기과 교수
※ 관심분야 : 자동제어, 신재생에너지