KIPS Tr. Software and Data Eng.
Vol.7, No.1 pp.19~24 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2018.7.1.19

A Generic Interface for Internet of Things (IoT) Platforms

Mi Kim† · Nam-Yong Lee†† · Jin-Ho Park†††

ABSTRACT

This paper presents an IoT platform of common interfaces that are flexible IoT applications and Connect the smart devices. The IoT platform includes automatic collaboration discovery the smart Device. It is different things case with connection each device through IoT Platforms are each device and application service. Due to these heterogeneities, it is quite advantages to develop applications working with derived IoT services. This circumstance needs the generic interface and results in a range of IoT architectures by not only the environments settings and discovery resource but also varied uniqe to implementation services working with IoT applications. Therefore, this suggestion of solve the problems and make it possible independent platforms always alive to connection with each devices based on the generic interface. The generic interface is heterogeneity-driven solve the problems and effectively contributions a platform that could be operated in diverse IoT Platforms.

Keywords : IoT Platform, Internet of Things, Generic Interface, Platform, Application

IoT 플랫폼을 위한 범용 인터페이스

김 미† · 이 남 용†† · 박 진 호†††

요 약

IoT 플랫폼은 다양한 사람과 사물의 가상세계를 객체로 연결하는 확장 가능한 IoT 애플리케이션 및 서비스의 개발을 위한 핵심 요소들이다. 그러나 IoT 플랫폼 시장은 매우 복잡하고 빠르게 변하고 있다. 이러한 IoT 플랫폼은 유용한 정보를 제공하고 단순한 기능을 수행하기 위해 IoT 디바이스와의 협업을 통해 서비스를 제공한다. 공통 서비스를 수행하기 위해 범용 서비스 인터페이스가 필요하며 IoT 장치 및 리소스 설정뿐 아니라 다양한 장치의 협업 환경에 따라 다양한 IoT 아키텍처가 구성된다. 기기의 이질성으로 인해 매번 다양한 IoT 서비스로 작업하는 응용 프로그램을 개발하는 것은 상당히 어려우며 이러한 응용 프로그램을 유지 관리하기가 훨씬 어렵다. 이러한 모든 문제는 IoT 장치 간의 이식성 및 이동성으로 인한 결과로 본 논문에서는 공통의 특징을 갖는 모든 IoT platform에서 수행될 수 있는 범용 인터페이스를 정의한다. 기존의 디자인 패턴을 채택하여 IoT 플랫폼의 공통화된 연결성을 제공하는 범용 인터페이스는 이질성 문제를 해결하고 다양한 IoT 플랫폼에서 수행 가능한 플랫폼 독립적인 Generic Interface가 수행될 수 있음을 확인했다.

키워드 : Iot 플랫폼, 사물인터넷, 범용인터페이스, 플랫폼, 어플리케이션

1. 서 론

Internet of Things (IoT) 패러다임을 통해 응용 프로그램은 IoT 플랫폼에 연결된 서비스와 상호 작용을 통해서 유용한 서비스와 사용 편의를 제공한다. 기존 IoT 플랫폼은 유용한 정보를 제공하는 등의 단순한 기능을 수행하고 IoT 디바이스와의 협업을 통해 서비스를 제공한다. 따라서 모든 플랫폼에 적용할 수 있는 범용 서비스 인터페이스가 필요하며 IoT 디바이스 및 리소스의 구성뿐 아니라 각 디바이스에 대한 다양한 협업 환경에 따라 다양한 IoT 아키텍처가 구성된다. 기종간의 이질성으로 인해 다양한 IoT 서비스로 작업하는 응용 프로그램을 개발하는 것은 상당히 어려우며 이러한 응용 프로그램을 유지 관리하기가 훨씬 더 어렵다. 이러한 모든 문제는 IoT 장치 간의 이식성과 이동성으로 인해 발생한다. 따라서, 본 논문은 범용 인터페이스를 제공하여 효과적으로 인터페이스의 집합을 정의하고 잘 알려진 디자인 패턴을 채택하여 IoT 플랫폼의 공통적 인 특징을 수용하는 범용 인터페이스는 이질성 문제를 해결하고 다양한 IoT 플랫폼에 효과적으로 채택 할 수 있다.

범용 인터페이스는 IoT 기반으로 리소스를 자동으로 구성하며 배터리 소모 실험을 통해서 범용의 서비스 인터페이스를 제안한다. 이기종 디바이스를 연결하는 다양한 환경이 데이터의 흐름을 제어하는 공통적인 서비스와 특징을 유연

† 준 회 원 : 리감 기술연구소 소장
†† 비 회 원 : 숭실대학교 컴퓨터학부 교수
††† 종신회원 : 숭실대학교 소프트웨어학부 교수
Manuscript Received : August 14, 2017
Accepted : September 13, 2017
* Corresponding Author : Jin-Ho Park(gomalove@daum.net)

하게 반영하는 범용 인터페이스를 제안한다. 논문의 구성은 2장에서 관련연구에 대해 논의하고, 3장은 범용 인터페이스를 제시하고, 4장에서 사례 연구 및 평가를 제시하며, 마지막 5장에서 결론을 내린다.

2. 관련 연구

IoT를 관리하는 플랫폼은 필립스 휴[1]와 IoT.EST[2]로 가정용 자동화 또는 센서 네트워크에 중점을 둔 플랫폼 연구가 대부분을 차지한다. 2011년 말까지 6LoWPAN에 API를 도입하기 위해 45억 달러 이상을 투자하였다. 이 솔루션은 매우 유연하고 견고하다. 소프트웨어 제약에 대한 축소 버전의 RESTFul을 도입한 COAP (Constrained Application Protocol)이 제안되었다. IoT 구현 솔루션은 uniform naming [4]과 어드레싱에 중점을 두고 있으며 유비쿼터스 스마트 객체를 위한 공통 프로토콜은 공통 통신 플랫폼을 개발하기 위해 물리적 객체의 가상현실을 소개한 연구이다. 이와 같이 하는 SENSEI프로젝트[5]는 실세계의 물리적 객체와 IoT에서 실현되는 관리적 대응성에 대한 개념을 밝히고 있다.

또한 자원은 물리적 계층의 자원을 나타내고 인터페이스 집합을 정의하여 소프웨어 프로세스 구성, 즉 리소스 종단과 연결한다.

IoT.EST(서비스 창출 및 테스팅을 위한 사물인터넷 환경) [6]에서는 서비스에 접근하기 위해 동적으로 구성을 할 수 있는 환경으로 구성 되었다 이 프로젝트를 통해서 다양한 통신 기술을 사용하는 센서 및 액추에이터의 데이터를 수집하고 활용하는 동적 서비스를 생성하는 환경을 구성한다. 이러한 아키텍처는 재사용이 가능하며 IoT 서비스 구성 요소에 기반한 비즈니스 서비스 구성, “사물”에 대한 서비스 구성 및 테스트 자동화, 상호 운용성 등을 보장하며 기술의 이질성으로 인한 문제를 해결하였다.

NOS는 분산처리 방식으로 배포된 네트워크 스마트 노드이며 리소스를 계산하는데 엄격한 제약 조건을 제시한다. IoT 디바이스와 인터페이스 할 수 있으며 수집된 데이터를 중앙 집중식 플랫폼보다 하단의 단말에서 처리한다.

NOS 계층은 노드에 의해 생성된 데이터 흐름을 관장하고 처리하는 역할을 담당한다[7].

또한 NOS HTTP 프로토콜을 기반으로 하는 연결지향 인터페이스로 MQTT (Message Queue Telemetry Transport) 프로토콜을 기반으로 한다. MQTT는 경량으로 자원을 공개하여 등록하는 프로토콜로써 역할을 한다[8].

NOS는 HTTP GET 요청을 통해 자원을 검색한다.

MQTT 및 NOS와의 본 논문에서 제안한 범용 인터페이스와의 차별점은 이기종간의 중계 브로커 없이 내부에서 정의된 에이전트를 통해서 드라이버 인터페이스를 직접 검색하고 호출한다. 따라서 브로커 메시지를 통한 별도의 수행 프로세스에 대한 리소스 낭비를 줄일 수 있다.

요약하면, 이질적인 환경이 플랫폼을 위한 특화된 구조에서 제공하는 방식으로 여러 논문에서 언급되었다. 하지만 기존 IoT 서비스의 공통 기능에 대한 통신프로토콜은 없는 실정이다. 관련연구에서 논한 IoT 플랫폼에 대한 연구는 모두 특화된 상황에서 데이터 흐름을 관장하는 IoT 응용단의 문제이다. 따라서 본 논문은 모든 IoT 플랫폼에 적용하여 특화된 서비스를 공통의 기능을 자동으로 연결한 범용 인터페이스를 제안하여 플랫폼 인터페이스 구축에 기여하고자 한다.

3. A Generic Interface Architecture

3.1 Design of Generic Architecture

본 절에서는 IoT Platform Architecture는 Service Layer, Configuration Layer, Device Layer 등으로 구성된다.

범용 API 드라이버는 메타 모델, 장치 관리자, 환경구성 관리자 및 보안 관리자와 같은 IoT 플랫폼과 연결된 범용 인터페이스이다.

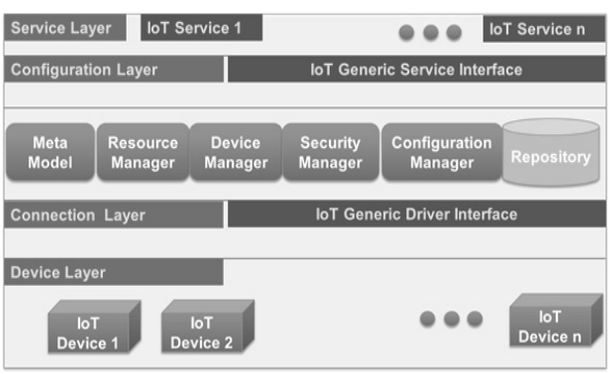

Fig. 1. Generic Architecture Platform

3.2 Framework for Collaborative Environments in IoT Platforms

IoT 플랫폼의 메타 모델[14]이 다양한 협업 환경구성과 관련하여 IoT에 대한 상세한 조사를 통해 Entity Objects, IoT Device, IoT Framework, IoT Service 등 4 가지 주요 핵심요소가 도출된다[9, 10]. 다음은 범용 인터페이스의 몇가지 설계상의 이점을 나열하였다.

첫째, IoT 기반의 서비스가 자동으로 구성된다. 이 서비스는 다른 서비스 환경을 구성하는 데 사용될 수 있다.

둘째, 서비스 자동발견으로 서비스를 효과적으로 등록하고 발견하여 메타데이터를 통해서 데이터 서비스를 등록한다.

마지막으로, 서비스 재사용성으로 코드의 재사용을 촉진하기 위해 다양한 서비스로 구성된다.

3.3 Meta - Model of IoT Framework Environments

기존의 IoT 플랫폼을 통해 객체, IoT 디바이스, IoT 프레임 워크, IoT 서비스 등의 핵심 요소가 요구된다. IoT 플랫폼의 메타 모델은 Fig. 2와 같이 이들 핵심 요소와 이들 간의 관계를 사용하여 기술된다.

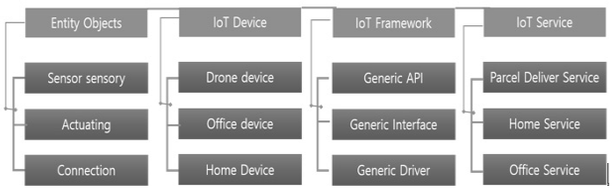

Fig. 2. Meta Model of IoT Framework

Object, People, Sensor, Actuator는 모든 환경을 구성하는 단위이다. Object는 사용자의 획득, 데이터 수집, 사용자 위치 등을 나타낸다. IoT 디바이스는 Drone, AR, 모션 캡쳐 등을 작동시키는 물리적 객체이다. 소프트웨어의 Leap Motion 컨트롤러는 3D 위치 정보를 생성하기 위해 2D 프레임 데이터를 비교하여. IoT 디바이스가 특정 엔티티를 모니터하도록 설정된 경우, 그 중 하나 이상의 센서는 원시 객체의 상태로 대상 객체의 상태를 캡처한다.

IoT 프레임워크는 IoT 디바이스 간에 연결을 관리하고 IoT 디바이스, 저장소에서 센서 데이터를 수집하여 저장소에 저장하고 이러한 서비스를 분석한다. IoT 프레임워크는 IoT 기기 및 원격 미터기, 아마존의 택배시스템 등의 용용 서비스와 협업한다.

IoT 서비스는 능동적이고 개인화된 지능형 서비스를 기반으로 사용자에게 서비스를 제공한다.

각 IoT 서비스는 IoT 디바이스에서 수집된 데이터를 가져와 사람과 서비스 메타 모델에 편리한 생활을 제공하며 원격 난방 시스템 및 택배 서비스로 서비스 환경을 구성하는 스키마이다.

3.4 Framework for Collaborative Environments

IoT 객체에 대한 프레임워크 설계는 초기 개발 단계에서 최종 개발단계까지 IoT구성과 협업 환경을 갖춘다. IoT 플랫폼의 프로세스는 데이터 수집과 연결의 정확도가 높고 제한된 배터리를 관리하며 IoT 서비스와 디바이스간의 일치성을 가져야 한다.

3.5 IoT Generic Service Interface

이기종 인터페이스를 위한 IoT 범용 인터페이스는 서로 다른 방식으로 서비스를 호출하며 범용 인터페이스를 호출하는 Get 방식을 통해 상위 레이어를 호출한다. 이 알고리즘은 이기종 IoT 서비스를 위한 내부에이전트를 통해서 매개변수를 전달하여 호출되는 방식이다. Drone, Wearable Watch 및 SmartCar Service에 대한 다양한 응용 프로그램을 프레임워크에서 적용하여 수행하며 Fig. 3은 범용 인터페이스 호출방식을 보여주는 알고리즘이다.

3.6 IoT Generic Driver Interface

메타 데이터를 통해서 드라이버를 검색하고 드라이버 레지스트리에 등록해 놓은 최종 드라이버를 찾는 방식의 알고리즘을 Table 1에서 확인 할 수 있다.

Table 1. Generic Driver Interface Algorithm

```
1   legacy device settings
2   ---------------------------------------
3   input data param dev_ID1, dev_ID2
4
5   Initializing Generic Driver Interface
6   Search Generic driver connection
7   Set Generic driver connection equal to discovery
8   dev_ID1, dev_ID2
9   Mapping param dev_ID1, dev_ID2 to interface
10  parameter
11
12  For each driver do
13   assignment to driver interface
14      If   driver discovery to dev_ID1, dev_ID2
15        then   success connection to device
16          Return;
17   Search    next device driver
18       For   each next driver
19            If   next device is discovery
20                 Then next driver assignment to device
21         Connection   registry device driver
    close connection
```

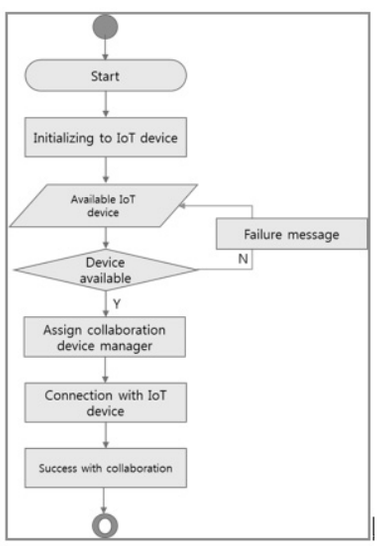

Fig. 3. Generic Interface Implements Algorithm

3.7 Configuration Layer

환경구성 매니저의 역할은 각 디바이스, 자원 등의 자동화된 환경 구성을 의미하며, 디바이스명, ID 등의 정보를 저장소에 실시간으로 저장하여 협업할 디바이스와 연결하고 현재 사용이 가능한지 메시지를 보내고 실패한 경우 다른 사용 가능한 자원을 찾는 방식으로 구성된다.

3.8 Connection Layer and Device Layer

커넥션 레이어와 디바이스 레이어는 연결할 디바이스를 디바이스풀에 등록하고 찾고자 하는 디바이스 드라이버를 조회하는 방식이다.

사용할 수 없는 경우 장치는 오류 메시지를 보내고 장치가 비활성 상태이면 다른 사용 가능한 장치를 실시간으로 찾는 방식이다. 디바이스 매니저(Fig. 4)의 내부 에이전트는

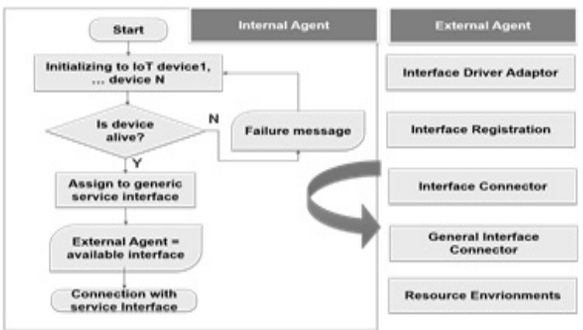

Fig. 4. Device Manager

내부 알고리즘을 수행하면서 디바이스 드라이버를 찾고 외부 에이전트는 이를 받아 디바이스를 등록한다.

4. A Case Study and Evaluation

4.1 A Case Study of IoT Platform

제안된 Generic Interface를 플랫폼에 실험 시나리오를 적용하여 IoT Platform 아키텍처의 유용성을 입증하고자 한다.

우리는 스마트폰을 통해서 IoT 모니터링 서비스에 따라 제어 될 수 있는 시나리오를 가정했다. 최종 사용자에게 발견 된 특수 디바이스 드라이버에는 광센서 및 온도 센서가 있음을 가정한다. 이 센서는 IoT 플랫폼에 연결되고 연결된 장비를 통해서 할당하기 위한 드라이버 셋을 쿼리 한다. 적합한 장치를 발견하면 IoT 모니터링 서비스가 광센서와 온도 센서를 할당하고 센서 모두가 IoT 플랫폼에 연결되어 있음을 확인할 수 있다. 사용자가 온도 센서와 날씨 도메인의 메뉴를 선택하여 IoT 모니터링을 통해서 IoT 저장소에 적재된 정보를 서비스하여 사용 가능한 장치 드라이버 목록을 획득한다. 본 실험에서 설계된 디바이스 레이어를 통해서 광드라이버 센서를 조회하여 최종 환경구성 레이어를 거쳐서 서비스 레이어로 서비스 된다. 사용자가 온도 센서 측정에 따라 마지막 옵션을 선택하면 IoT 모니터링 서비스는 구성장치가 정확한지 여부를 판단하여 실제 서비스를 할당할 수 있는지 추론한다.

4.2 실험환경 및 평가

본 실험에 따르면, 사용자는 서로 다른 환경구성 중에서 인증된 기기를 통해 해당 서비스를 제공받는다.

범용 인터페이스는 기능에 대한 공통 명세서를 제공한다.

실험을 위해 Generic Interface는 이기종 인터페이스로 구현되며 Service Layer, Configuration Layer, Connection Layer, Device Layer와 같이 4 개의 다른 레이어를 배포한다.

첫째, Service Layer는 가정용 서비스 및 사무실 서비스를 원격 제어 패널(예 : 집안일 켜기 / 끄기)에서 기능을 식별하고 사용자의 활동을 분석하고 분석 결과를 표시한다. 전원 스위치를 측정하여 사용자의 난방 상태를 관리하며 이러한 방식의 서비스 레이어를 통해서 Fig. 5와 같이 프레임워크를 구성한다.

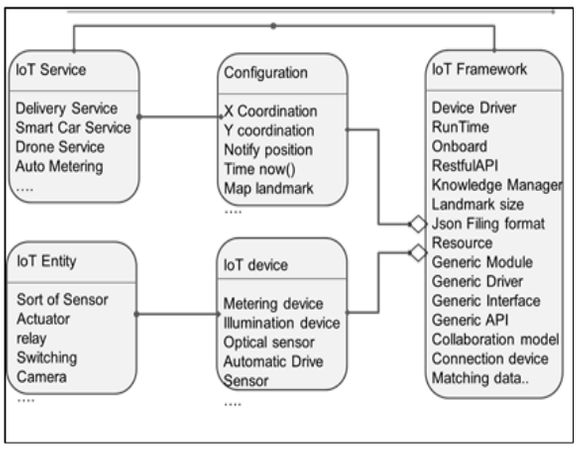

Fig. 5. Framework for Collaborative Environments

IoT 플랫폼의 데이터 흐름은 각 레이어 간의 다양한 센서데이터의 흐름을 연결하여 IoT 서비스의 매개 변수 사양에 따라 IoT 서비스에 제공한다. 각 IoT 서비스는 아래와 같이 서로 다른 계층 구조의 요구 사항을 갖고 있다.

ConfigSET_IoT Service Layer = {서비스, Monitoring}

ConfigSET_IoT Configuration Layer = {리소스, Wi-Fi, GPS

ConfigSET_IoT Connection Layer = {연결, 매개 변수}

ConfigSET_IoT Device Layer = {발견, 등록}

각 데이터 유형은 Table 2와 같이 하나 이상의 장치 센서에서 요구 사항을 제공하며 응답시간별 가중치를 계산하여 개발비용을 산정하였다.

Table 2. IoT Requirement of IoT Experiments Spec

Layer	Configugation	IoT Parameter	Cost Price Weight
Service Layer	DevSET_IoTService Layer	Cognitive WIFI	0.5
Configuration Layer	DevSET_IoTConfiguration Layer	Resource, GPS	0.4
Connection Layer	DevSET_IoTConnection Layer	Connection Value1	0.3
Device Layer	DevSET_IoTDevice Layer	Discovery Dev1	0.5

각 계층은 비용을 측정한 값, IoT 매개 변수를 가중치로 지정된 값으로 실행하도록 설정하고 높음 (0.5), 중간 (0.4) 및 낮음 (0.2)과 같고 응답시간별로 비용 산출 가중치를 산정하였다.

실험은 Table 3과 Fig. 7과 같이 IoT 각 계층에 대한 쿼리의 응답 시간별 효율성과 정확성을 측정하였다.

응답 시간별 효율성은 A/B 로 A는 실제 처리된 데이터 응답 시간이고, B는 데이터의 전체응답시간을 합으로 나눈 값이

Table 3. Results of Efficiency and Accuracy Spec

Settings Layer	IoT query	Efficiency	Accuracy
DevSET_IoTService Layer	Service WIFI	95%	90%
DevSET_IoTConfiguration Layer	Resource, GPS	86%	80%
DevSET_IoTConnection Layer	IoT Parameter	88%	70%
Device Layer	DevSET_IoTDevice Layer	92%	90%

다. 정확도는 범용인터페이스를 통해서 처리된 정확한 데이터를 전체 데이터로 나눈 정확도를 백분율로 나타낸 것이다[11].

실험결과 제안된 플랫폼과 알고리즘은 CPU 사용에 대한 자원소비를 크게 줄이고 IoT 플랫폼에 대한 데이터 전송 속도를 크게 향상시킨다. 광범위한 실험은 제안된 플랫폼의 다음과 같은 이점을 제공한다. 제안 된 플랫폼의 범용 인터페이스 시뮬레이션은 다중 서버 큐잉 시스템을 기반으로 설계된 프레임워크로 자바 기반 시뮬레이터를 사용하여 수행된다. 시뮬레이션에서, 4개의 수신 요청은 데이터 패킷의 응답시간을 시간당 비용 가중치를 계산하여 데이터 트랜스미션과 평균 LED 디스플레이 밝기 레벨을 표시하여 전원의 에너지 소모를 측정하였다. 각 요청의 서비스 기간은 무작위로 분배되며 자원 요구는 주기적으로 지정됩니다. 요청된 메모리 요구 사항은 2MB~1024MB이며 각 요청의 CPU 요구 사항은 1~2 코어로 각 요청의 저장소 요구 사항은 10MB~10GB가 된다.

Table 4는 Eclipse ADT (Android Development Tools (ADT))의 DDMS (Dalvik Debug Monitor Server, 특정 시간 동안의 에너지 소비 분석) 툴로 측정한 CPU로드 결과를 보여준다.

Table 4. Power Consumption of Platform (mW)

Division	Power Consumption(mW)		
	Generic IoT	EVRTYHNG	THINGWORX
Sensor data	2970	3710	3920
WIFI	3050	5210	4550

IoT 서비스 운영 및 CPU 측정은 최소한의 전력만 소모하며, 디스플레이에서의 소실은 LED 디스플레이 장치의 밝기 값을 낮춤으로써 줄일수 있다. Fig. 6은 범용 IoT와 ThingWorx의 에너지 소모 비교를 보여준다.

우리는 결과를 시각화 하기 위해 두 가지 그림에서 전력 소비 분석 도구를 사용하였으며, Fig. 6은 ThingWorx와 비교하여 레이어드 아키텍처 당 범용 IoT 플랫폼의 응답 시간 결과를 보여준다. 응답 시간이 ThingWorx보다 더 빠르다는 것을 확인했다. Fig. 7은 데이터 전송률의 효율성과 정확도를 측정하여 비교한 그림이다.

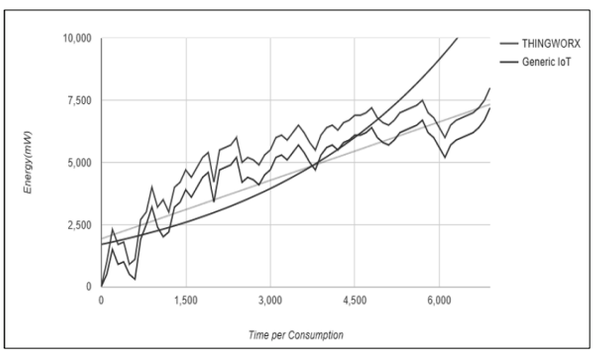

Fig. 6. Power Consumption of Generic IoT (mW)

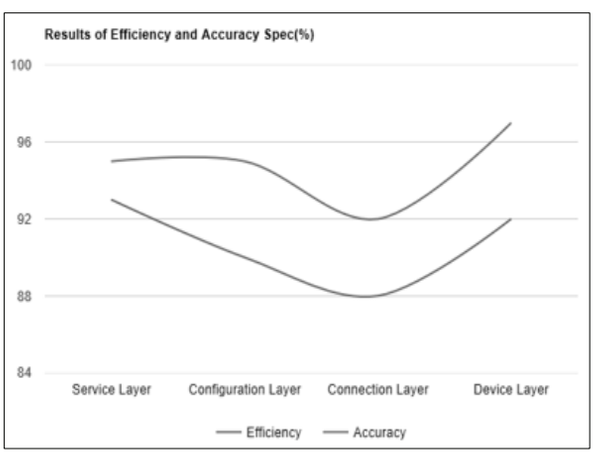

Fig. 7. Results of Efficiency and Accuracy Spec

5. 결 론

이질성으로 인한 기기 간 모든 문제는 IoT 디바이스간의 이식성과 이동성으로 인해 발생한다. 따라서 본 논문은 범용 인터페이스를 제안하여 효율적으로 범용 인터페이스에 대한 공통적인 특징을 정의하고 잘 알려진 디자인 패턴을 채택하여 IoT 플랫폼의 공통적인 특징 별 유형에 대해 실험을 통해서 다른 플랫폼보다 높은 데이터전송률을 얻을 수 있었다.

효율적인 알고리즘으로 제안된 플랫폼과 이를 사용하여 범용IoT 플랫폼에서 폭넓게 활용 될 수 있음을 확인하기 위해 에너지 사용량에 대한 자원 소비를 줄였다. IoT 플랫폼에 대한 높은 활용도를 확보하기 위해 응답시간 당 데이터 전송 속도의 통신비용 원가를 측정하여 본 논문에서 제안한 플랫폼의 이점을 제공하였다. IoT 플랫폼의 이동성과 이식성은 빈번하게 사용하는 재사용성에 기인하며 컴팩트하고 단순화 되어 이식성이 높아야 한다. 따라서 자주 사용하는 기능은 통신비용을 낮추어 실험하였다. 제안한 범용 인터페

이스는 다양한 기기간의 이질성으로 인한 문제를 해결하기 위해 공통의 인터페이스를 통해서 다양한 IoT 플랫폼의 협업을 위한 플랫폼 독립적인 연결 지향성을 제공한다.

또한 에너지 소모별 CPU로드를 측정하여 Generic Interface가 통신비용을 최소화하기 위해 효율적으로 데이터를 전송함으로써 여러 플랫폼에서 채택할 수 있음을 확인했다.

References

[1] Philips Hue. (2014). “Meet hue,” [Internet], http://www.developers.meethue.com/.

[2] S. De, F. Carrez, E. Reetz, R. Tonjes, and W. Wang, “Test-enabled Architecture for IoT service creation and provisioning,” *Future Internet Lect. Notes Comput. Sci.*, Vol.7858, pp.233-245, 2013.

[3] Voxeo Labs Tropo Whitepaper, Make the Shift from Telco Power to Telco Powered with the Tropo API 2013.

[4] J. MongayBatalla, P. Krawiec, M. Gajewski, K. Sienkiewicz, “ID layer for internet of things based on name-oriented networking,” *J. Telecomm. Inf. Technol.*, Vol.2, pp.40-48, 2013.

[5] EU FP7 SENSEI Project Consortium: Final SENSEI Architecture Framework. SENSEI Project Deliverable D3.

[6] IOT-ESTProject [Internet], http://ict-iotest.eu/iotest/.

[7] S. Sicari, A. Rizzardi, D. Miorandi, C. Cappiello, and A. Coen-Porisini, “A secure and quality-aware prototypical architecture for Internet of Things,” *Information Systems*, Vol.58, pp.43-55.

[8] IBM and Eurotech, MQTT V3.1 Protocol Specification, http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html.

[9] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things(IoT): A Vision, architectural elements, and future directions,” *Future Generation Computer Systems*, Vol.29, No.7, pp.1645-1660, 2013.

[10] D. Miorandi, S. Sicari, F. Pellegrini, and I. Chamtac, “Internet of Things: Vision, Applications and Research Challenges,” *Ad Hoc Networks*, Vol.10, No.7, pp.1497-1516, 2012.

[11] M. Kim, N.-Y. Lee, and J.-H. Park, “A Quality Evaluation model for IoT Services,” *KIPS Tr. Comp. and Comm. Sys.*, Vol.5. No.9, pp.269-274, 2016.

김 미

http://orcid.org/0000-0001-9649-8365
e-mail : pytwoori@gmail.com
2000년 호남대학교 정보통신공학과(석사)
1996년~2000년 LG전자 연구원
2015년~현 재 숭실대학교 컴퓨터학과 박사수료
2017년~현 재 리감 기술연구소 소장
관심분야 : IoT 서비스, Software Engineering, Software Quality

이 남 용

http://orcid.org/0000-0003-3688-0145
e-mail : nylee@ssu.ac.kr
1979년 숭실대학교 전자계산학과(학사)
1983년 고려대학교 경영정보학(MIS) (경영학석사)
1993년 미시시피 주립대학교(MSU) 경영정보학(MIS)(경영학박사)
현 재 숭실대학교 컴퓨터학부 교수
관심분야 : 소프트웨어 테스팅, 품질보증, MIS, IT정책경영

박 진 호

http://orcid.org/0000-0003-1961-6983
e-mail : gomalove@daum.net
1998년 숭실대학교 소프트웨어공학(학사)
2001년 숭실대학교 컴퓨터학과 소프트웨어공학전공(석사)
2011년 숭실대학교 컴퓨터학과(박사)
2018년~현 재 숭실대학교 소프트웨어학부 교수
2018년~현 재 숭실대학교 SW교육연구소 SW융합센터 센터장
2018년~현 재 한국SW.ICT총연합회 사무총장
2018년~현 재 한국IT정책경영학회 수석부회장
2018년~현 재 한국정보처리학회 이사
2018년~현 재 대통령 직속 지역발전위원회 위원
2018년~현 재 국회 4차산업혁명 SW안전포럼 부의장, 스마트라이프IoT포럼 / 초융합포럼 위원
관심분야 : SW Reuse, SW Maintenance, IT Service, IT기술평가전략