사물인터넷 오픈소스 기술 – IoTivity

표준 + 오픈소스 개념을 활용하여 최고의 사물인터넷 플랫폼에 도전한다

이원석, 차홍기, 전종홍
한국전자통신연구원

Ⅰ. 서 론

사물인터넷 시장의 주도권 확보를 위한 치열한 플랫폼 경쟁이 진행 중에 있으며 IoTivity는 OIC 표준 기반 오픈소스 프로젝트로 상호운용성 확보와 기술 파급력 극대화를 통한 주도권 확보에 도전하고 있다.

사물인터넷(Internet of Things, 약어로 IoT)은 각종 사물에 센서와 통신 기능을 내장하여 인터넷에 연결하는 기술을 의미하며, 여기서 사물이란 가전제품, 모바일 장비, 웨어러블 컴퓨터 등 다양한 임베디드 시스템이 될 수 있다[1]. 오픈소스 소프트웨어는 소스코드가 공개되어 있는 소프트웨어를 말하며, 일반적으로 자유롭게 사용 · 복제 · 배포 · 수정할 수 있다. 오픈소스 소프트웨어의 대표적인 예로는 리눅스 커널 및 관련 GNU 소프트웨어, 아파치 웹서버, 파이어폭스 웹 브라우저, MySQL 데이터베이스 시스템, 파이썬/펄 언어, 이클립스 툴 등을 들 수 있으며, 그 외에도 많은 오픈소스 소프트웨어들이 전세계에 걸쳐 수많은 개발자들에 의해 개발되고 있다[2]. 본 논문에서는 소개하는 IoTivity[3]는 사물인터넷과 관련된 오픈소스 소프트웨어이다.

아래의 〈그림 1〉은 사물인터넷의 가장 초보적인 단계의 두가지 유즈케이스를 보여준다. 좌측 유즈케이스는 스마트폰 앱을 이용하여 실내 온도 등 에어컨의 상태정보를 확인하거나 외부에서 집안의 에어컨을 실시간으로 제어 상황을 보여준다. 이와 같은 유즈케이스는 사용자가 여름에 퇴근 전 집안의 온도를 확인하고 필요한 경우 원하는 온도를 설정하여 집에 도착하였을 때 사용자가 원하는 최적의 상태를 만들어 사용자 환경을 윤택하게 만들 수 있다. 오른쪽의 유즈케이스는 집안에 카메라를 설치하여 스마트폰 앱을 이용하여 언제 어디서나 실시간으로 집안의 상황을 모니터링 서비스를 제공하는 것을 보여준다. 보안 관점에서 또는 반려 동물의 상황을 모니터링하는데 유용하게 활용될 수 있다. 이와 같은 사물인터넷 기반 서비스는 지속적으로 다양해지고 확대되어 실생활에서 사람들에게 보다 안전하고 편리한 생활 환경을 제공해 줄 것으로 기대하고 있다.

그러나 사물인터넷 생태계를 만드는데 있어서 해결이 필요한 다양한 이슈들이 있으며 이들 중 가장 큰 이슈 중 하나는 상호호환성 확보이다. 현재 스마트폰 생태계를 주도하고 있는 애플과 구글은 홈킷(Homekit)[4]과 브릴로(Brillo)[5] 등 독자적인 기술 기반으로 향후 엄청난 시장을 형성할 사물인터넷 시장의 주도권 확보를 위해 치열한 경쟁 중에 있다. 또한 퀄컴 주도하에 2013년

그림 1. 스마트폰을 이용한 에어컨 제어 및 집안 실시간 모니터링 유즈케이스(사진 출처: Belkin)

12월에 결성되어 마이크로소프트, LG전자 등 180개 이상의 회원를 확보하고 있는 올씬 얼라이언스(AllSeen Alliance)[6]는 올조인(Alljoyn) 오픈소스를 중심으로 사물인터넷을 위한 미들웨어 기술 개발 및 확산에 노력 중에 있다. 삼성전자와 인텔 주도로 2014년 7월에 결성된 OIC(Open Interconnect Consortium)[7]는 시스코, HP, 미디어텍 등 100개 이상의 회원사를 확보하고 있으며 사물인터넷 표준과 표준을 구현한 오픈 소스 프로젝트인 IoTivity를 동시에 개발하는 방식을 채택하여 사물인터넷 시장에서 상호호환성 확보와 기술 파급력을 극대화하여 사물인터넷 시장에서의 주도권 확보 경쟁을 버리고 있다. 특히 IoTivity 오픈소스 프로젝트는 로열티 프리 정책을 기반으로 하고 있어 어느 회사든 로열티 없이 본 오픈소스를 적용한 제품 출시가 가능하다.

본고에서는 OIC 표준 기반한 사물인터넷 오픈소스 기술인 IoTivity에 대해 알아본다. 본고의 구성으로 IoTivity오픈소스 프로젝트의 개념 및 아키텍처를 살펴보고, 세부적인 기술인 IoTivity 스택과 프로토콜에 대해서 설명한다. 또한 IoTivity 서비스에 대해서 설명하고, 마지막으로 향후 개발될 주요 개발 계획에 대해서 설명한다.

II. 본 론

1. IoTivity 개념 및 구성

가. IoTivity 개념

IoTivity의 목적은 미래 사물인터넷 세상에 출현할 수십억 개의 디바이스를 다양한 운영체제와 네트워크 프로토콜에 상관없이 자연스럽게 연결할 수 있는 오픈소스 소프트웨어 프레임워크를 개발하는데 있다.

사물인터넷 생태계 활성화를 위해 가장 중요한 부분 중의 하나는 다양한 사물을 제조하는 중소기업들이 자신들의 제품에 인터넷 연결 기능을 추가하고 이에 대한 스마트폰 앱이나 서비스 얼마나 쉽게 함께 제공할 수 있는 환경을 제공할 수 있는가에 있다. IoTivity는 이러한 요구사항을 만족시키기 위한 프레임워크로 뛰어난 사물인터넷 기기간 상호운영성 보장과 빠른 사물인터넷 제품 개발을 가능하게 한다. 또한 새롭게 개발되는 최신 사물인터넷 기술을 지속적으로 확대 적용하며, 다양한 오픈소스 하드웨어(e.g. 라즈베리파이, 에디슨 등)와 소프트웨어 플랫폼(e.g. 안드로이드, iOS, 윈도우, 리눅스 등) 지원 범위를 확대해 나아가고 있다. 현재 IoTivity는 우분투(Ubuntu), 타이젠(Tizen), 안드로이드를 지원하고 있으며, iOS는 향후 지원 예정이다. 오픈소스 하드웨어 플랫폼의 경우 현재 아두이노(Arduino), 에디슨(Edison)을 지원하며 지속적으로 지원

IoTivity는 쉽고 빠르게 상호운용성이 뛰어난 사물인터넷 제품 개발을 가능하게 하는 오픈소스 소프트웨어 프레임워크이다.

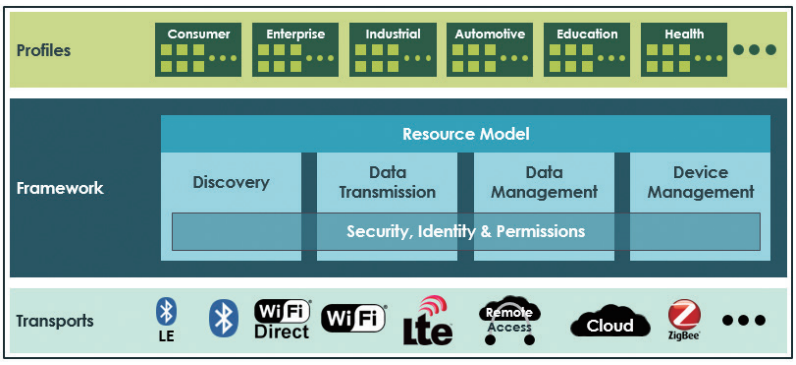

그림 2. IoTivity 개념 아키텍처

하드웨어 플랫폼을 확대할 예정이다.

또한 기본적으로 IoTivity는 OIC 표준 기반의 사물인터넷 미들웨어 오픈소스 기술로 뛰어난 상호운용성을 지원한다.

〈그림 2〉는 IoTivity의 개념적인 아키텍처를 보여준다. 블루투스, 와이파이, Zigbee 등 사물인터넷에서 활용되는 다양한 연결 기술에 대한 부분인 전송(Transports) 레이어, 프로파일 레이어의 소비자, 기업, 자동차, 헬스 등 다양한 사물인터넷 응용들이 전송 레이어의 연결 기술과는 독립적으로 개발될 수 있도록 리소스 발견, 데이터 전송, 디바이스 관리, 데이터 관리 등의 기능을 지원하는 프레임워크 (Framework) 레이어, 사물인터넷 응용의 각각의 버티컬 분야를 의미하는 프로파일(Profiles) 레이어와 같이 크게 세개의 레이어로 구성된다. 전송 레이어의 경우 지속적으로 신기술들이 확장될 수 있으며, 이러한 확장이 되더라도 프레임워크 레이어의 지원으로 프로파일 레이어의 응용들은 변경없이 실행될 수 있다.

참고로 라이선스 정책은 아파치 2.0을 따르고 있으며 리눅스 재단에 의해 운영되고 있다.

나. IoTivity 구성

IoTivity는 리소스 기반 RESTful 아키텍처 모델을 기반으로 하고, 따라서 모든 사물을 리소스로 표현하고 CRUDN(Create, Read, Update, Delete and Notify) 오퍼레이션을 제공한다. 또한 데몬(Deamon) 없이 CoAP(Constrained Application Protocol) 기반으로 설계되어 저사양, 저전력 기기 지원이 용이한 장점이 있다.

IoTivity 프레임워크에 대한 세부적인 구성은 〈그림 3〉과 같다.

IoTivity는 크게 OIC 핵심 표준을 구현한 리소스 블록과 편리한 스마트폰 앱/서비스 개발 지원하는 서비스 블록으로 구성된다

〈그림 3〉과 같이 IoTivity 프레임워크는 크게 IoTivity 서비스와 관련된 기본 서비스(Basic Service)블록과 추가 서비스(Additional Service) 블록 그리고 OIC 표준 기반 구현 부분인 자원(Resource) 블록으로 구성된다.

자원 블록은 OIC 표준 기반한 부분으로 일반 리소스 모델, 리소스 발견, 메시징, 식별자 및 주소표현, CRUDN 오퍼레이션, 보안 등 IoTivity 프레임워크의 근간을 이루는 핵심적인 부분이다. IoTivity의 필수 메시징 프로토콜은 IETF CoAP이며 향후 선택적인 메시징 프로토콜로 HTTP(HyperText Transfer Protocol), MQTT(MQ Telemetry Transport)도 지원 예정이다.

IoTivity 서비스 부분은 IoTivity 기반의 확장 기능과 스마트폰 앱/서비스 개발시 빠른 개발을 지원하는 프레임워크 기능들로 구성된다. 즉, 프로토콜 플러그인 관리자(Protocol Plugin Manger)는 IoTivity 프레임워크에서 지원하지 않는 특정 프로토콜을 추가할 수 있는 기능이며, 소프트웨어 센서 관리자(Software Sensor Manger)는 센서들을 조합하여 새로운 가상 센서를 만들

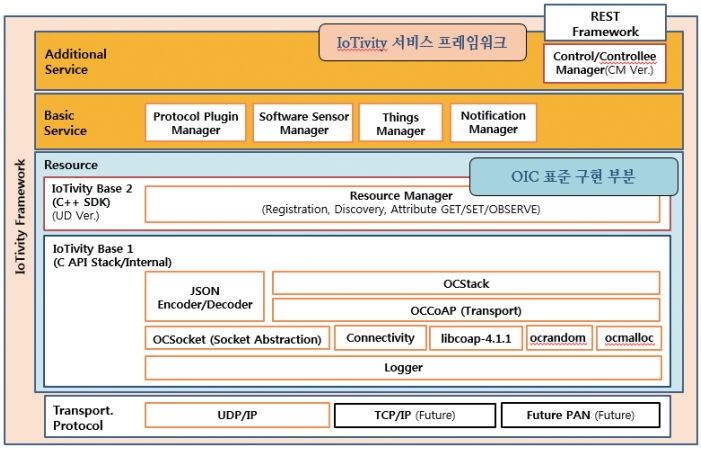

그림 3. IoTivity 프레임워크 구성

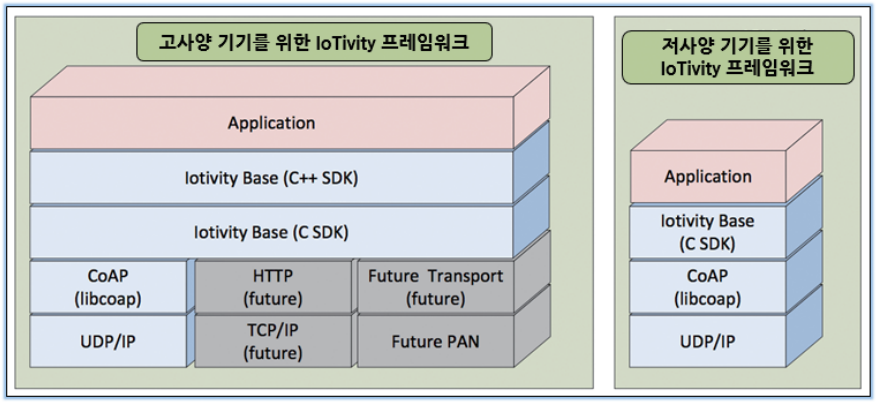

그림 4. IoTivity 프레임워크 분류

수 있는 유연한 기능을 지원한다. 또한 사물 관리자(Things Manager)는 여러 사물들을 그룹화하여 한번에 제어할 수 있는 기능을 지원하며, 알림 관리자(Notification Manager)는 사물의 특정 상황에 대한 알림 기능을 지원한다.

2. IoTivity 프레임워크

가. 프레임워크 분류

IoTivity 프레임워크는 〈그림 4〉와 같이 크게 고사양 기기를 위한 프레임워크와 저사양 기기를 위한 프레임워크로 나눌 수 있다. 저사양 기기를 위한 프레임워크는 컴퓨팅 파워가 약한 사물에 적용하기 위한 것으로 IoTivity 서비스 기능 없이 최소한의 기본 기능만 지원하며 바이너리 크기가 100KB 정도된다. 반면에 고사양 기기를 위한 프레임워크는 IoTivity 서비스를 포함하고 있으며 안드로이드, iOS, 윈도우 등에서 사물과 연동되는 애플리케이션 개발에 사용하기 위한 것이다.

IoTivity는 기기의 컴퓨팅 파워에 따라 고사양 기기를 위한 프레임워크와 저사양 기기를 위한 프레임워크를 지원한다.

나.. IoTivity 프레임워크 실행 모드

IoTivity 프레임워크는 클라이언트, 서버 그리고 모두(클라이언트 및 서버) 형태로 설정하여 사용할 수 있다. 일반적으로 사물은 자신의 리소스와 관련된 서비스를 제공하므로 서버 모드로 설정된다. 반면에 스마트폰 앱 또는 클라우드의 서비스는 사물의 상태를 요청하여 읽어오거나 사물의 설정 변경을 요청하는 형태로 사용되므로 클라이언트 모드로 설정되어 사용된다. 아래의 〈그림 5〉는 IoTivity를 활용하여 소스코드 레벨에서 서버 모드와 클라이언트 모드를 설정하는 예

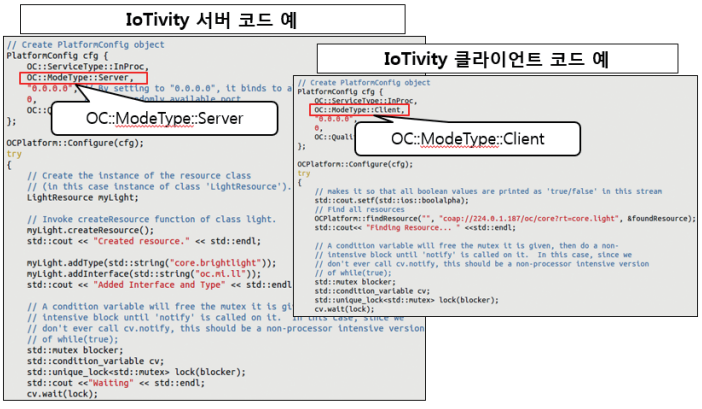

그림 5. IoTivity 서버/클라이언트 설정 코드 예

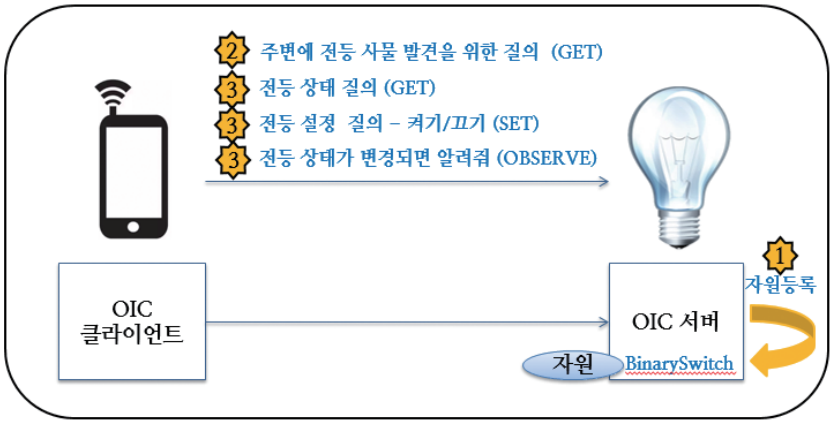

그림 6. IoTivity 프레임워크 기반 사물간 상호 동작의 예

를 보여준다.

다. IoTivity 리소스 블록 주요기능

리소스 블록은 OIC 표준을 구현한 부분으로 IoTivity의 기반이 되는 핵심기능을 제공한다. 특히 리소스 블록은 사물간의 연동에 필수적인 주변 사물이나 리소스 발견과 그 후 특정 사물이나 리소스에 대해서 상태를 질의나 사물이나 리소스의 설정을 변경하나 상태 변경을 모니터링 하는 기능을 제공한다.

IoTivity에서 사물들 간의 핵심적인 연동 기능은 리소스 블록에 구현되어 있다.

〈그림 6〉은 IoTivity를 기반으로 스마트폰과 스마트 전등이 상호 연동 하는 예와 이때 활용되는 주요 동작 기능을 보여준다. 스마트 전등을 제어하는 스마트폰은 OIC 클라이언트 역할을 하며, 스마트 전등은 OIC 서버 역할을 하며 전등 모니터링과 설정 변경 기능을 서비스로 제공한다. 〈그림 6〉의 동작시나리오를 순차적으로 설명하면 먼저 스마트 전등 제품이 켜지는 시점에 (1) OIC 서버에 "BinarySwitch" 자원을 등록하여 서비스 제공을 위한 기본 준비를 한다. 스마트폰은 OIC 클라이언트 역할을 하며 (2) 주변에 스마트 전등을 찾는 메시지(e.g. "GET /oc/core?rt=light")를 CoAP 멀티캐스트 형태로 보낸다. 이에 해당되는 스마트 전등은 유니캐스트로 스마트폰에 사물에 대한 기본적인 정보를 보낸다. 스마트폰은 스마트 전등이 보내준 정보를 보고 (3) 전등이 상태(켜져있는지 또는 꺼져있는지)를 질의할 수 있고, 또는 (3) 전등이 켜져 있을 경우 끄고, 꺼져있을 경우 켜도록 설정 요청을 진행할 수도 있다. 또한 (3) 전등의 상태가 변경되는 시점에 스마트폰에 알려달라고 스마트 전등에 요청을 할 수도 있다.

이러한 세부적인 동작들이 IoTivity 프레임워크 내부에서는 어떻게 동작하는지 아래에서 설명한다.

① 리소스 등록

사물 디바이스와 스마트폰 디바이스가 연동하는 구조에서는 사물은 OIC 서버 역할을 하며 스마트폰은 OIC 클라이언트 역할을 한다.

리소스 등록은 일반적으로 사물에서 실행되는 서버가 서비스로 제공할 리소스를 등록하는 것을 의미한다. 예를 들면 다른 디바이스에서 리소스 발견 요청을 받으면 등록된 리소스에 대한 정보를 제공하게 된다. 또한 등록된 리소스는 IoTivity 클라이언트 디바이스로부터의 리소스 상태 정보 요청이나 리소스의 상태 변경 요청을 받는 대상이 된다.

〈그림 7〉은 리소스 등록 절차를 보여준다. 애플리케이션에서 리소스 등록을 위해 (1)에서 (6)까지의 처리를 진행하게 된다. 즉, C++ SDK의 platform.RegisterResource()를 호출하고 이 메소드는 C API 스택의 InProcServer.RegisterResource() 호출하고 이 함수가 OCStack의 OCCreateResource()를 호출한다. 그 후 각 함수 실행 결과를 받아서 처리하는 구조로 구현되어 있다.

② 디바이스/리소스 발견

디바이스/리소스 발견은 주변 디바이스에 멀티캐스트로 디바이스 또는 원하는 리소스에 대한

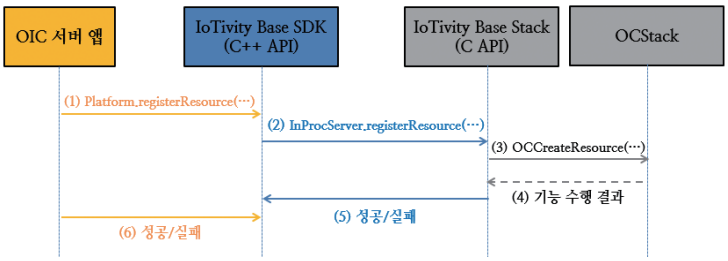

그림 7. 리소스 등록 절차

질의를 하고 결과를 받는 절차를 통해서 이루어진다. 즉, 주변에 원하는 디바이스가 있는지 또는 원하는 리소스가 있는지 알 수 없기 때문에 주변의 모든 디바이스를 대상으로 질의하는 것이 필요하며 따라서 멀티캐스트 프로토콜을 사용하여 질의를 수행한다. 그 후 모든 디바이스는 질의를 받게 되고 질의에 응답이 필요한 디바이스는 질의한 디바이스에게만 응답을 하면 되는데 따라서 유니캐스트를 사용하여 응답을 수행한다.

〈그림 8〉은 이러한 전체적인 절차를 이해하기 쉽도록 그림으로 표현한다.

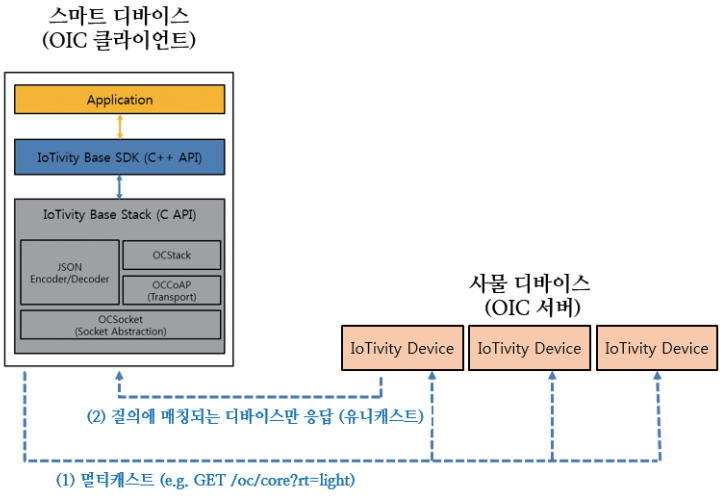

그림 8. 디바이스/리소스 발견 절차

③ 리소스 상태 질의

리소스 상태 질의는 일반적으로 리소스 발견 후 리소스 상태 정보를 요청하는 것을 의미한다. 예를 들면 스마트 전등의 경우 "BinarySwitch"라는 리소스를 갖고 있고 현재 상태 확인을 원하는 디바이스의 "BinarySwitch" 상태를 요청할 수 있다. 이러한 기능을 이용해서 외출 중에 집안에 있는 모든 전등이 켜져있는지 혹은 꺼져있는지 확인하는데 사용할 수 있다.

④ 리소스 상태 설정

리소스 상태 설정은 일반적으로 리소스 발견 후 리소스 상태에 대한 변경 요청을 하는 것을 의미한다. 즉, 외부에서 집안의 모든 전등의 상태 질의 후 거실의 등이 켜져 있는 상태라면 거실 등의 "BinarySwitch" 리소스 상태 설정 변경 요청을 통하여 끌 수 있다.

⑤ 리소스 상태 모니터링

리소스 상태 모니터링은 IoTivity 클라이언트가 특정 사물의 리소스 상태의 변경시 마다 변경된 상태를 모니터링 하고 싶을 때 사용하는 기능이다. 원하는 리소스를 관찰 대상으로 등록

IoTivity는 기본적으로 디바이스/리소스 발견 기능을 지원하며 발견 후에 해당 리소스에 대한 상태 확인이나 상태 설정을 가능하게 하는 기능도 지원한다.

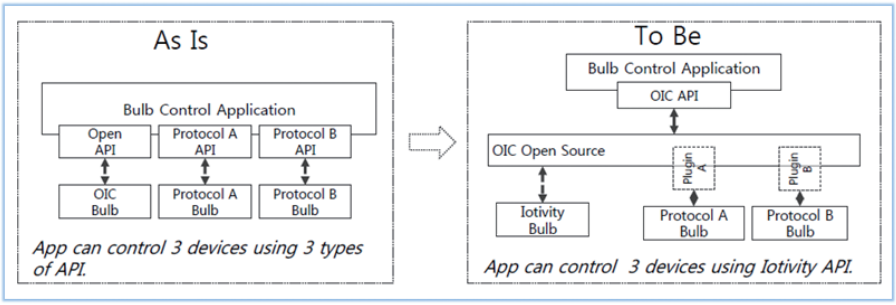

그림 9. PPM 지원 방향

하면 사물의 IoTivity 서버 기능 등록된 리소스의 모니터링을 시작하고 변경이 감지되면 이의 상태 정보를 IoTivity 클라이언트에게 전달한다. 본 기능의 장점은 이와 같이 지속적으로 리소스의 상태를 모니터링 하기 위해서 IoTivity 클라이언트가 지속적으로 상태정보 요청을 하지 않고도 효율적으로 같은 기능을 수행할 수 있다는데 있다. 예를 들면, 휴가를 가면서 아파트 현관문과 외부 유리창의 스마트 도어락을 잠김으로 설정 후 리소스 상태 모니터링을 설정하여 상태변경이 있을 때에 사용자에게 알림 메시지를 보내도록 할 수 있다.

3. IoTivity 서비스

IoTivity는 프로토콜 확장과 편리한 스마트폰 앱/서비스 개발 지원을 위한 서비스 프레임워크를 제공한다.

IoTivity 서비스는 기본적인 사물간의 연동을 위한 근본적인 기능이라기 보다는 IoTivity 기반 상용화를 보다 쉽게 지원하기 위한 추가적인 서비스 프레임워크이다. IoTivity 서비스는 크게 기본 서비스 블록과 추가 서비스 블록으로 구분되며, 기본 서비스 블록은 프로토콜 플러그인 관리자, 소프트웨어 센서 관리자, 사물 관리자, 알림 관리자 기능을 지원한다.

가. 프로토콜 플러그인 관리자(PPM)

PPM은 IoTivity에서 지원하지 못하는 추가적인 프로토콜이 필요한 경우 정형화된 플러그인 기반 프로토콜 확장하여 사용할 수 있는 기능을 지원한다. 확장 프로토콜은 공유 라이브러리 형태인 '.so' 형태가 되며 런타임에 필요한 프로토콜에 대한 .so 로딩을 하여 실행한다.

또한 플러그인으로 지원하는 프로토콜의 단위는 특정 프로토콜 (e.g. MQTT or AllSeen) 지원 형태나 특정한 프로토콜에서 활용되는 특정한 기기(e.g. AllSeen FAN or Philip Hue) 지원 형태 모두 가능하며 이는 플러그인 개발자가 선택하여 개발하면 된다.

〈그림 9〉는 PPM이 없는 상황에서 프로토콜을 확장하여 사용할 경우 IoTivity 애플리케이션 개발자가 프로토콜 별로 특정 API를 사용하여 개발이 필요했지만, PPM을 사용하면서는 확장되는 API와 독립적으로 정형화된 API를 사용하여 대응이 가능하기 때문에 개발 효율성이 크게 확대되는 장점을 가질 수 있다.

나. 소프트웨어 센서 관리자(SSM)

SSM은 물리적/논리적 센서 정보 등을 조합하여 새로운 가상의 센서 정보를 정의할 수 있는 소프트웨어 서비스 기능을 지원한다.

다. 사물 관리자(TM)

스마트홈에서 에너지 절약에 대한 유즈케이스 중 하나는 집 주인이 외출을 할 때 집의 모든 등을 끄거나 전기를 아끼기 위해서 외출시 전기를 차단해도 문제가 없는 모든 전자제품의 플러그를 끈다. 이와 같은 기능을 수행하기 위해서는 스마트폰 애플리케이션 개발시 집안의 각 사물들을 하나씩 제어해야 하는 비효율적인 문제가 있다. 이러한 유즈케이스를 효과적으로 대응할 수 있도록 지원하기 위한 기능이 TM 기능이다.

TM은 사물을 원하는 형태로 그룹화하여 효과적으로 관리할 수 있는 기능을 지원한다. 이를 위해 기본적으로 사물에 대한 그룹 생성, 조회, 수정, 삭제 기능을 지원하고 그룹 상태나 디바이스 변경을 확인할 수 있는 기능을 제공한다.

라. 알림 관리자(NM)

주인이 외출 중일 때 집의 창문이나 문이 열리면 사용자에게 상태를 알려주는 유즈케이스는 가정 집 보안과 관련된 가장 일반적인 케이스이다. IoTivity에서는 특정 사물의 상태변화가 발생한 경우 이를 특정 디바이스로 알려주는 기능을 쉽게 개발할 수 있도록 지원하는 NM 기능을 지원한다.

Ⅲ. 결 론

본고에서는 IoTivity 오픈소스 기술에 대한 기본 개념 및 아키텍처를 살펴보고, 리소스 스택의 분류 및 OIC 표준 기반으로 개발된 핵심적인 기능들에 대한 세부적인 동작 방식을 소개하였다. 또한 IoTivity에서 개발의 편의성을 위해서 추가적으로 지원하는 IoTivity 서비스들에 대해서도 알아보았다.

사물인터넷 생태계 구축을 위해 초기에 가장 중요한 역할은 사물을 상품화하는 다양한 제조사에게 있다. 그러나 많은 중소 제조사들은 IT기술에 대한 전문지식 및 인력들의 부족으로 빠르게 진화하는 IT 기술을 활용하여 경쟁력있는 상품을 개발하는 것은 매우 어려운 일이다. 따라서 빠른 사물인터넷 제품 개발을 가능하게 하고, 뛰어난 사물인터넷 기기간 상호운용성 보장과 최신의 사물인터넷 기술을 지속적으로 적용하며 다양한 하드웨어와 소프트웨어 플랫폼을 지원하는 사물인터넷 미들웨어 플랫폼 기술 필요성이 매우 높은 상황이다. IoTivity는 이런한 요구사항을 만족시켜가는 로열티 프리(Royalty Fee) 오픈소스 기술로 사물인터넷 생태계를 확산하는데 핵심적인 역할을 할 것으로 기대된다.

참고로 2015년 10월 IoTivity 1.0.0 버전[8]이 공식적으로 릴리즈 되었다.

올해 10월 공식적인 IoTivity 1.0.0이 릴리즈 되어 2016년부터 IoTivity를 적용한 상용제품 출시가 시작될 것으로 예상된다.

Acknowledgment

이 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No. R0127-15-1042, 라이프케어를 위한 스마트 웨어러블 표준 개발).

참 고 문 헌

[1] J. Höller, V. Tsiatsis, C. Mulligan, S. Karnouskos, S. Avesand, D. Boyle: From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence. Elsevier, 2014.

[2] "오픈소스SW 라이선스 가이드1.0(초급편)," 한국저작권 협회, Feb. 2012.

[3] "IoTivity," 2015, from https://www.iotivity.org/.

[4] "Homekit," 2015, from https://developer.apple.com/homekit/.

[5] "Project Brillo," September, 23, 2015, from https://developers.google.com/brillo/.

[6] “Allseen Alliance,”2015, from https://allseenalliance.org/.
[7] “Open Interconnect Consortium,”2015, from http://openinterconnect.org/.
[8] “IoTivity 1.0.0,”2015, from https://www.iotivity.org/downloads/iotivity-1.0.0 .