사물인터넷을 위한 개방형 소프트웨어 플랫폼 설계

Open Software Platform Architecture for Internet of Things

저자
(Authors)

이혜민, 신동군
Hyemin Lee, Dongkun Shin

출처
(Source)

한국정보과학회 학술발표논문집 , 2015.6, 1492-1494 (3 pages)

발행처
(Publisher)

한국정보과학회
KOREA INFORMATION SCIENCE SOCIETY

URL

http://www.dbpia.co.kr/Article/NODE06394453

APA Style

이혜민, 신동군 (2015). 사물인터넷을 위한 개방형 소프트웨어 플랫폼 설계. 한국정보과학회 학술발표논문집, 1492-1494.

사물인터넷을 위한 개방형 소프트웨어 플랫폼 설계

이혜민, 신동군
성균관대학교 정보통신공학부
gudbooy@skku.edu , dongkun@skku.edu

Open Software Platform Architecture for Internet of Things

Hyemin Lee, Dongkun Shin
Sungkyunkwan University

요 약

사물인터넷(Internet of Things, IoT)은 사람과 사물 등 모든 것이 인터넷을 통해 연결되어, 정보의 수집, 생성, 처리, 공유가 가능해 지는 것을 의미한다. 사물인터넷에서는 여러 가지 센서 장치가 스마트 단말기와 연결되어 사용자 주변의 다양한 센서 정보를 제공하고, 센서 데이터를 가공하여 사용자에게 서비스를 제공하는 응용이 확산될 것이다. 기존의 센서 장치가 고정된 기능을 가진 반면에, 본 연구에서는 프로그래밍 가능하고 스마트단말과 연동이 가능한 OPEL(Open Platform Event Logger)이라고 불리는 개방형 플랫폼 이벤트 수집장치를 대상으로 한 소프트웨어 플랫폼을 제안한다. 제안하는 플랫폼은 다양한 사물인터넷 장치에서 사용가능하며, 사용자가 쉽게 프로그래밍 할 수 있고, 센서 장치의 전력 관리 및 안정성을 제공하는 할 수 있는 장점이 있다.

1. 서 론

인터넷 기술의 발전에 따라 사용자에게 언제 어디서나 인터넷 정보를 검색하고 이를 활용할 수 있도록 많은 서비스들이 제공되고 있다. 이러한 시도 중에 하나인 사물인터넷 (Internet of Things)은 1999년 케빈 애쉬튼 MIT 오토아이디 센터 소장이 처음 고안해 낸 용어로 알려져 있다. 사물인터넷은 사이버 환경에서 존재하는 사물들이 인터넷을 통하여 서로 연결되고 이러한 사이버 공간과 물리공간의 사물들을 서로 연동하여 다양한 서비스를 제공하고 있다. 가트너의 보고에 의하면, 2020년에 260억 이상의 디바이스들이 인터넷에 연결될 것이며(PC, 스마트폰, 테블릿 제외)[1], Cisco 역시 향후 2020년까지 500억 이상의 사물이 연결되는 환경이 도래한다고 예상하였다.[2] 국내외 기업들은 사물인터넷 관련 제품 및 서비스 비즈니스 규모 또한 빠르게 성장할 것으로 예상한다.

이러한 흐름에 맞추어 사물인터넷을 위한 플랫폼의 구축이 요구된다. 그 결과 많은 국내외 기업들이 앞다투어 사물인터넷을 위한 플랫폼 시장에 뛰어들고 있다. 그 대표적인 예로 OIC(Open Interconnection Consortium)의 IOTivity[3]가 있으며 AllSeen Alliance에 Alljoyn[4]이 있다.

사물인터넷 시대의 주요 장치로 최근 많은 발전이 있는 다양한 센서를 탑재한 이벤트 기록장치를 고려해 볼 수 있다. 예를 들어, 차량에 탑재되는 블랙박스나 가정용 감시카메라도 영상, 위치, 온도 등의 데이터를 수집하여 사용자에게 제공하는 이벤트 기록장치로 분류할 수 있다. 하지만, 이들 장치의 한계는 기능이 고정되어 있는 폐쇄형 장치로 스마트 단말처럼 사용자가 임의로 프로그램밍할 수 없는 단점이 있다. 이에 본 연구에서는 OPEL(Open-Platform Event Logger)이라고 불리는 개방형 플랫폼을 사용한 이벤트 기록장치를 제안하고, 또한, 이를 위한 소프트웨어 플랫폼 구조를 제안한다.

OPEL은 Javascript 언어로 프로그램밍 가능하도록 Node.js 기반으로 설계되었다. Javascript는 스크립트 언어로 코드 재사용성 및 확장성이 매우 뛰어나다는 장점이 있다. Node.js는 Ryan Dahl에 의해 2009년 처음 시작되었는데, 가장 큰 장점으로는 오픈소스이며, 크로스플랫폼(Cross-Platform)이다. 게다가 이벤트 드리븐 (Event-Driven)방식의 개발이 가능하여 네트워크 애플리케이션 (특히 서버 사이드)개발이 용이하다.

결과적으로 본 연구가 제안하는 OPEL은 기존의 블랙박스나 가정용 감시카메라와 같은 폐쇄형 단말기가 아닌 개발자들이 프로그래밍 가능한 환경을 제공한다. 또한 OPEL은 원시적으로 제어해야 하는 다양한 센서들을 고수준의 인터페이스로 추상화하여 개발자에게 제공한다. 사용자는 고수준의 인터페이스를 사용하여 스크립터 언어로 원하는 기능을 쉽게 구현할 수 있는 장점이 있다.

2. 본 론

OPEL은 개발자에게 고수준 자바스크립트 인터페이스를 제공함으로서 센서들간의 관계를 보다 쉽게 정의 할 수 있으며, Node.js의 NPM(Node Packaged Modules)을 이용하여 센서를 이용한 웹 어플리케이션 및 실시간 어플리케이션을 만들 수 있는 환경을 제공한다. 사용자는 다양한 어플리케이션을 설치하여 센서들을 활용할 수 있으며, 사용자의 스마트폰을 통하여 OPEL 장치를 통제할 수 있다. 시스템 개발자들은 센서들의 대한 저수준 작업을 모듈로 만들어 배포할 수 있으며, 응용 프로그램 개발자는 고수준 인터페이스를 통해서 쉽게 응용 프로그램을 구현할 수 있다. 예를 들어, 카메라와 같은 영상 센서에 대해서는 단순히 영상을 촬영하는 기능 외에 차량 번호판 인식, 차선 인식, 얼굴 인식 등과 같은 고수준 서비스들을 제공하여 개발자가 쉽게

이 논문은 2013 년 정부(미래창조과학부)의 재원으로 (재)스마트 IT 융합시스템 연구단(글로벌프론티어사업)의 지원을 받아 수행된 연구임 ((재)스마트 IT 융합시스템 연구단-2011-0031863).

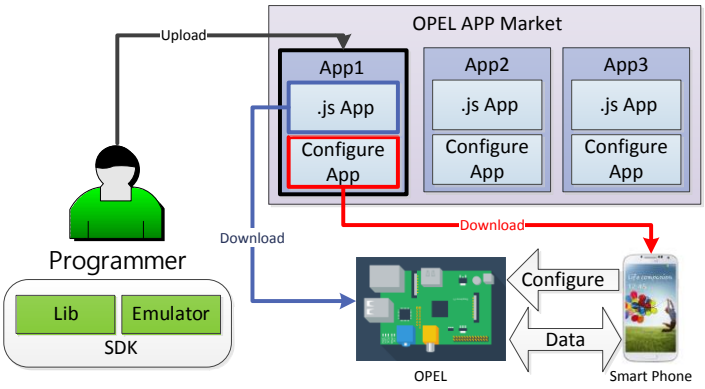

그림 1. OPEL 어플리케이션 마켓

블랙박스의 ADAS(Advanced Driver Assistance System)와 같은 기능을 쉽게 구현 가능하다. 본론에서는 OPEL 장치가 갖추어야 할 요구 사항에 대해 언급하며, 시스템 구조에 대하여 설명한다.

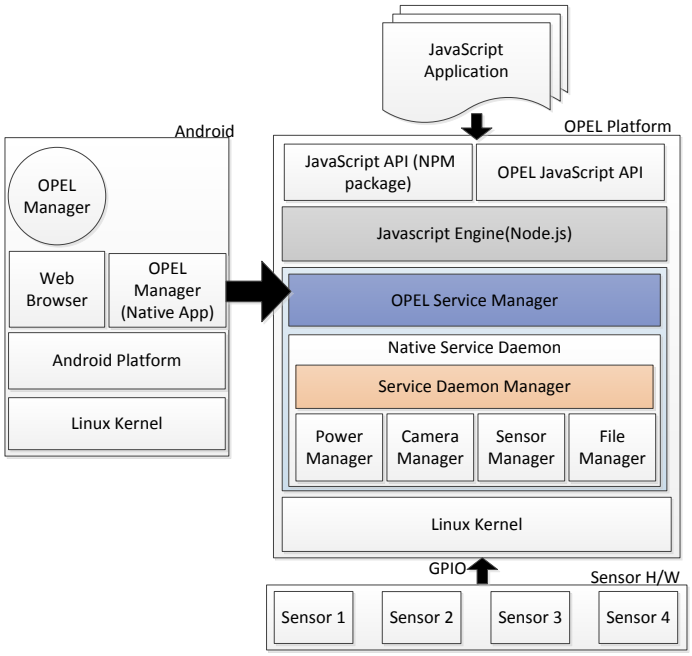

그림 2. OPEL 시스템 구조

2.1 시스템 요구 사항

OPEL 장치는 원격으로 제어되기 때문에 반드시 갖추어져야 되는 요소들이 존재한다.

첫 번째, 제한된 전원공급 상황에서도 사용자 응답에 항시 반응해야 한다. 두 번째, 응용 프로그램 및 시스템 오류로 인한 전체 서비스의 중단을 방지해야 한다.

OPEL은 이러한 문제를 해결하기 위해 다양한 요소들을 가진다. 제한된 전원공급 상황을 대비하기 위해 내부의 전력 관리자가 플랫폼 및 기기 상황을 파악하여 전력손실을 최소화 한다. 어플리케이션 및 시스템의 오류가 발생하는 경우 각각이 개별 모듈로 동작하기 때문에 오류가 발생해도 다른 모듈에게 영향을 미치지 않는다. 더욱이 자바스크립트 구현된 응용프로그램이 오류를 발생하더라도 기본적인 서비스 프로세스들은 독립적으로 작동하기 때문에 오류의 영향에서 벗어나면서 사용자의 작업을 안전하게 진행한다.

2.2 사용자 요구 사항

사용자의 스마트폰을 통해서 언제든지 OPEL 장치에 접속이 가능해야 하며, 각 어플리케이션, 플랫폼, 그리고 OPEL 하드웨어에 대한 통합적인 관리가 필요하다. 따라서 이러한 관리를 스마트폰의 어플리케이션인 OPEL Manager로 할 수 있다. 사용자는 어플리케이션을 쉽게 다운로드 받아야 하며 개발자는 어플리케이션을 하드웨어의 구성없이 개발 할 수 있어야 한다. 그림 1은 사용자가 어플리케이션을 OPEL 마켓을 통해 설치하는 방법을 보여준다. OPEL 개발툴킷(SDK)의 에뮬레이터를 통하여 개발자는 실제 플랫폼과 동일한 환경에서 어플리케이션을 테스트 하고 개발할 수 있다. 응용 프로그램을 마켓에서 내려받아 OPEL 장치에 설치 시, OPEL내부에 자바스크립트 응용 어플리케이션이 설치되며, 사용자의 스마트폰에도 OPEL장치 내에 자바스크립트 어플리케이션을 제어할 수 있는 환경이 함께 설치된다.

2.3 OPEL 시스템 디자인

개발자 및 사용자에게 편의성을 보장하기 위해서는 저수준의 센서 인터페이스를 고수준의 인터페이스로 추상화 하는 과정이 필요하다. 이러한 조건을 충족시키기 위해서 각각의 저수준의 서비스를 처리하는 프로세스들과 고수준 인터페이스가 통신하는 과정이 요구되어진다.

그림 2는 OPEL의 시스템 구조를 보여준다. OPEL은 리눅스 커널위에서 작동하는 플랫폼이다. 리눅스 커널위 네이티브 서비스 데몬은 저수준 작업을 처리하는 데몬 프로세스들의 집합이다. 데몬 매니저는 센서 데이터를 가공하는 프로세스들을 모듈별로 추가, 삭제, 관리하는 서비스를 제공한다. 데몬 매니저가 관리하는 서비스 프로세스는 다음과 같다.

Power Manager 는 장치의 실시간 상태를 파악하여 최소한의 전력으로 OPEL장치가 운전되도록 하는 서비스이다.

Camera Manager 는 OpenCV라이브러리를 탑재하여 카메라를 통해 받은 영상 및 이미지를 처리하는 서비스이다.

Sensor Manager는 OPEL장치에 설치된 Sensor를 GPIO(General Purpose I/O)를 이용하여 데이터를 읽어 들이고 가공하는 서비스이다.

File Manager는 Camera와 Sensor Manager를 통해 얻은 데이터를 플래시 저장장치에 저장하고 관리하는 서비스이다.

OPEL Service Manager는 데몬 매니저와 통신하여 프로세스들의 상태를 사용자에게 제공한다. 각각의 프로세스들은 고수준 인터페이스에게 프로세스 간 통신(IPC)을 이용하여 데이터를 송수신 한다. OPEL Service Manager는 스마트폰과 직접적인 통신을 하며 플랫폼 및 기기 상황을 파악하고 사용자에게 OPEL장치를 제어할 수 있는 권한을 제공한다. 또한, 어플리케이션의 설치/삭제를 담당하여, 어플리케이션의 실행 또는 중단 등의 동작을 관리한다.

V8 engine 기반의 Javascript Framework인 Node.js는

자바스크립트로 작성된 어플리케이션을 실행시키는 역할을 한다. 자바스크립트 기반 어플리케이션은 node.js를 거쳐 OPEL의 다양한 서비스 데몬과 통신이 가능하다.

안드로이드 기반 사용자 스마트 단말기에는 OPEL Service Manager와 직접적으로 통신하는 네이티브 어플리케이션(OPEL Manager)이 탑재된다. OPELManager는 OPEL장치에 대한 전반적인 상태 및 응용프로그램들을 통제할 수 있다. 더욱이 다양한 플랫폼에서 OPELManager를 구동하기 위해서 Node.js에서 실행되는 웹서버에 접속을 하여 네이티브 어플리케이션인 OPELManager와 동일한 환경으로 OPEL장치를 제어할 수 있다.

3. 구 현

3.1 구현 환경

본 연구에서는 Raspberry Pi B+(700MHz ARM1176JZF-S 싱글 코어, RAM 512MB)에 다양한 센서 모듈을 설치하여 OPEL을 구현했으며, 운영체제는 Raspbian과 리눅스 커널3.18을 사용했다. 센서 모듈로는 라즈베리 파이 카메라, 동작 감지센서, 온도센서를 GPIO 를 이용하여 임베디드 보드와 연결 하였다.

3.2 구현 내용

OPEL의 데몬 서비스들은 C/C++언어를 이용하여 작성이 되었으며 Raspberry Pi B+에서 제공하는 GPIO 라이브러리 및 라즈베리파이 카메라를 OpenCV에서 사용 가능하도록 추가적인 오픈소스 라이브러리(raspicam_cv)를 설치했다. 각각의 데몬 서비스들은 Glib의 이벤트 드리븐 콜백 함수 방식으로 설계하였으며, D-Bus[5]를 이용해 고수준 인터페이스와 데이터 통신을 하게 된다. 하지만 고수준 인터페이스는 단순히 자바스크립트 API로서 추가적인 작업이 필요하다. 따라서 C/C++코드를 Node.JS 패키지 Add-on으로 바인딩 하였다. 따라서 고수준의 인터페이스는 단순히 정의된 API만을 보여주고 실제 데이터를 처리하는 부분은 서비스 데몬이다. 사용자의 환경을 셋팅하기 위해 OPEL Manager라는 안드로이드 어플리케이션을 개발하여 OPEL 장치내에 있는 OPEL Service Manager와 통신하여 어플리케이션을 OPEL 장치에 다운로드 하고 실행 및 종료 그리고 제어할 수 있는 환경을 사용자에게 제공한다.

3.3 구현 예제

OPEL장치의 센서들을 이용해 실제 사용자가 다음과 같은 응용프로그램을 사용할 수 있다.

1) 감시카메라로 이용

적외선 동작 감지 센서를 통해 물체가 감지되면 해당 물체의 동영상을 찍으면서 LED불빛과 스피커를 통해 소리를 낼 수 있도록 정의한다. 게다가 카메라로부터 이미지를 받아 이미지프로세싱을 하여 물체의 정보를 OPEL Manager앱을 통해 사용자에게 제공한다.

2) 운전 보조 장비로 이용

카메라로 차선의 이미지를 얻어 차량이 차선을 넘게되면 스피커를 통해 소리를 낼 수 있도록 정의한다. 따라서 운전자가 부주의하게 차선을 넘게 되면 경고음을 발생 시켜 운전자를 보조한다.

3) 주차용 블랙박스로 이용

자이로 센서를 이용하여 주차 후 충격이 가해지면 카메라를 작동시켜 주변에 있는 자동차 번호를 이미지 프로세싱하여 사용자에게 경고 이벤트와 함께 자동차 번호를 핸드폰에 전송하게 된다.

OPEL장비는 이러한 다양한 케이스에서 사용될 수 있다. 사용자는 간단하게 OPEL마켓에서 어플리케이션을 다운로드 내려받아서 자신에 환경에 맞게 장비를 운용할 수 있으며, 어플리케이션 개발자는 손쉽게 고수준 인터페이스를 이용해 빠르고 다양한 어플리케이션을 생산할 수 있다.

3.3 구현 결과

모션감지센서, LED, 온도 센서, 그리고 라즈베리파이 카메라를 라즈베리파이와 GPIO를 통해 구성했다. 모션감지센서를 통해 모션이 감지되는 경우 카메라를 통해 영상을 30초간 녹화하도록 데모 코드를 테스트했다 모든 컴포넌트들이 실행될 때 top명령어를 통해 OPEL 서비스 전체 메모리 사용량은 전체 시스템 메모리의 8.7%(카메라 및 센서 프로세스 작동 시)사용량을 보였다. 따라서 해당 작동을 하는데 약44MB정도의 메모리가 사용되었다.

4. 결 론

최근 사물인터넷은 인터넷 업계에서 화제의 단어이다. 기업들은 자사 플랫폼에 사물인터넷을 이식하여 사용자에게 더 나은 인터넷 환경을 제공하려고 노력하고 있다. 따라서 이런 흐름에 맞추어 본 연구는 개발자 및 사용자에게 센서에 대한 직접적인 제어 권한을 부여하여 직관적인 사용자 경험을 제공하는 플랫폼의 개발의 요구에 따라 OPEL 이라는 사물인터넷용 플랫폼을 디자인 하였다. OPEL은 기존의 프로그램이 불가능한 폐쇄형 장치와 다르게 개발자에게 고수준 언어를 지원하여 더 쉽고 빠르게 개발하도록 제공하는 사물인터넷 플랫폼 기법이다. 본 연구의 연장선으로서 자바스크립트 엔진의 최적화 및 Node.js구조의 경량화의 필요성이 구현 과정에 대두되었으며 저비용 임베디드 장비에 최적화된 OPEL을 만들 필요성이 있다.

참고문헌

[1]Gartner Inc., “The Internet of Things, Worldwide,” 2013.

[2]Bank of America Merrill Lynch, “2015 Technology Year Ahead,” 2014.

[3]IoTviity, https://www.iotivity.org

[4]AllJoyn, https://allseenalliance.org

[5]Robert, L., “Get on the D-BUS,” Linux Journal, Vol. 130, No.3, 2005