SmartThings 스마트 앱을 위한 실시간 모니터 구현

고가영 O1 창병모 1 최광훈 2
1 숙명여자대학교 컴퓨터과학과
2 전남대학교 전자컴퓨터공학부
dianakoh@sm.ac.kr, chang@sm.ac.kr, kwanghoon.choi@jnu.ac.kr

Implementation of Real-Time Monitor for SmartThings Smart App

Gayoung Koh O1 Byeongmo Chang 1 Kwanghoon Choi 2
1 Dept. of Computer Science, Sookmyung Women's University
2 Dept. of Electronics and Computer Engineering, Chonnam National University

요 약

SmartThings는 사물인터넷(IoT) 플랫폼으로 IoT 디바이스 및 디바이스를 자동화하는 스마트 앱(Smart App)을 개발, 관리하는 환경을 제공한다. 그러나 SmartThings 클라우드 내에서 실행되는 스마트 앱은 그 앱의 동작 과정을 확인하는 것이 어렵다. 본 연구에서는 이러한 점에 주목하여 스마트 앱 사용자가 웹에서 스마트 앱의 동작 과정을 실시간으로 확인할 뿐 아니라 스마트 앱의 프로파일 정보를 파악할 수 있도록 스마트 앱 모니터를 설계 구현하였다.

1. 서 론

사물인터넷(Internet of Things:IoT)은 사물에 센서와 통신 기능을 탑재하여 무선 통신을 이용해 여러 사물을 연결하는 기술이다. SmartThings는 스마트 홈을 주 타겟으로 하는 사물인터넷(IoT) 플랫폼으로 IoT 디바이스 및 디바이스를 자동화하는 스마트 앱(Smart App)을 개발, 관리하는 환경을 제공한다. [1].

스마트 앱은 SmartThings 클라우드에서 실행되며 디바이스, app, location 등의 이벤트 발생 시 이벤트와 연결된 이벤트 핸들러가 실행되고 이벤트 핸들러 내에서 디바이스 액션 및 메시지 전송 액션이 실행되는 구조이다. 이러한 스마트 앱의 특성 상 사용자는 IoT 디바이스의 액션 동작 외 이벤트 발생 및 이벤트 핸들러 메소드 등의 앱의 동작 과정을 시각적으로 확인하는 것이 어렵다. 사용자가 앱의 실행과정을 실시간으로 모니터링하면 이벤트, 이벤트 핸들러, 액션 등이 실행되는 과정을 확인할 수 있고 이를 통해 스마트 앱이 의도에 맞게 실행되었는지 확인할 수 있으며, 그렇지 않은 경우에는 그 원인을 파악할 수 있다.

IoT 모니터링에 대한 기존 연구는 디바이스(CPU 사용률, firmware 버전)나, 네트워크(백앤드 시스템 부하, 디바이스와의 통신 상태)를 주로 다루었다[3,4]. SmartThings용 스마트 앱을 관리하는 모바일 앱에서는 허브와 연결된 IoT 디바이스를 기준으로 각 디바이스에서 실행되는 액션은 확인할 수 있지만 스마트 앱의 실행과정은 확인할 수 없다.

본 연구에서는 이러한 점에 주목하여 스마트 앱이 실제로 동작하는 과정을 웹을 통해 실시간으로 확인하고 스마트 앱의 프로파일 정보를 파악할 수 있는 스마트 앱 모니터를 설계 구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 논문의 기술적 배경에 대해 기술한다. 3장에서는 본 연구에서 구현한 주요 기능 및 시스템 구조에 대해 기술하고 4장에서는 구현에 대해 기술한다. 5장에서 결론을 맺는다.

2. 배 경

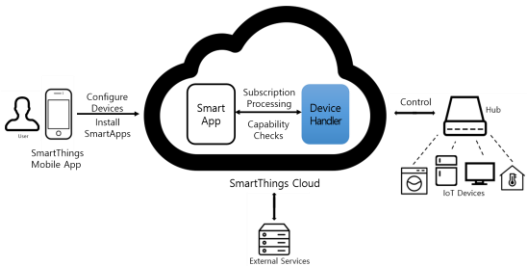

그림 1. SmartThings 플랫폼 구조 [2]

SmartThings는 스마트 홈을 위한 사물인터넷(IoT) 플랫폼으로서 그림 1과 같이 허브를 통하여 IoT 디바이스들을 연결하고 관리한다. SmartThings 모바일 앱은 IoT 디바이스와 스마트 앱을 관리하는 모바일 애플리케이션이다.

스마트 앱(Smart App)은 IoT 디바이스의 기능(capability)을 사용하여 디바이스를 자동화하는 SmartThings 클라우드에서 실행되는 작은 앱으로 Groovy 언어로 작성한다 [1].

```
// definition
definition(
    name: "Control the light according to movement",
    namespace: "yournamespace",
    author: "your name",
    description: "Turn on/off the light according to movement",
    category: "Convenience")
// preferences
preferences {
    section("Select devices") {
        input "motion1", "capability.motionSensor"
        input "light1", "capability.switch"
    }
}
//predefined callbacks
def installed() {
    subscribe(motion1, "motion", motionHandler)
}
def updated() {
    unsubscribe()
    subscribe(motion1, "motion", motionHandler)
}
// event handler
def motionHandler(evt) {
    if (evt.value == "active") {
        light1.on()
    } else if (evt.value == "inactive") {
        light1.off()
    }
}
```

그림 2. 스마트 앱 코드[1]

스마트 앱은 그림 2와 같이 definition, preferences, predefined callbacks, event handler로 구성된다.
- definition: 스마트 앱의 이름 및 앱의 세부 정보들을 정의한다.
- preferences: 스마트 앱의 설치 및 업데이트 시 나타나는 화면 페이지를 정의하는 부분으로서 스마트 앱에서 사용할 디바이스를 선택할 수 있다.
- predefined callbacks: installed(), updated() 를 포함하며 스마트 앱의 설치 및 업데이트 시 스마트 앱에서 사용되는 event subscription을 정의한다.
- event handler: 스마트 앱에서 사용되는 event subscription 및 다른 메소드들을 구현한다.

3. 주요 기능 및 시스템 구조

3.1 주요 기능

스마트 앱 모니터의 목적은 SmartThings 클라우드에서 실행되고 있는 스마트 앱의 실제 동작 과정을 사용자가 확인할 수 있도록 하는 것이다. 이를 위한 스마트 앱 모니터의 주요 기능은 다음과 같다.

- 실시간 모니터: 실시간 스마트 앱의 동작 과정을 확인할 수 있다.
- 프로파일: 과거 발생한 스마트 앱의 동작 과정을 발생 시간 정보와 함께 확인할 수 있다.
- 웹 기반 모니터: 실시간 모니터 및 프로파일은 웹 페이지를 통해 편리하게 확인할 수 있다.
- 이벤트 처리 플로우: 스마트 앱 모니터는 스마트 앱의 이벤트 처리 과정을 보여주는 것으로 이벤트 발생, 이벤트 핸들러 실행, 메소드 실행, 액션 실행의 플로우 형태로 나타낸다.

3.2 시스템 구조

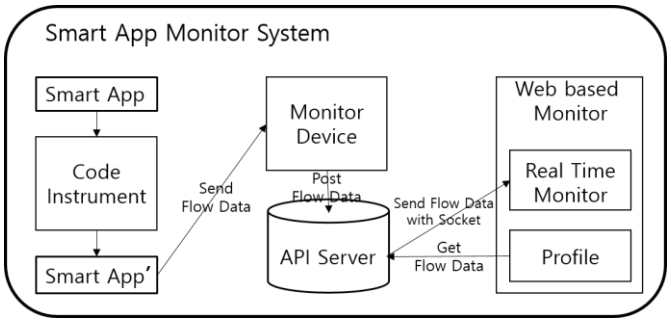

그림 3. 스마트 앱 모니터 시스템 구조

스마트 앱 모니터는 그림 3과 같이 Code Instrument, 모니터 디바이스, API 서버, 실시간 모니터, 프로파일로 구성된다.
- Code Instrument: 스마트 앱을 모니터하기 위해 필요한 코드를 기존 스마트 앱에 추가한다. 변형된 스마트 앱을 설치하면 스마트 앱은 모니터 디바이스에 앱의 플로우 데이터를 전달한다.
- 모니터 디바이스: 스마트 앱으로부터 받은 플로우 데이터를 API서버에 전송한다.
- API 서버: 모니터 디바이스에서 전달된 플로우 데이터를 저장하고 실시간 모니터로 데이터를 보낸다.
- 실시간 모니터: 스마트 앱의 플로우 정보를 실시간으로 볼 수 있으며 한 개 이상의 스마트 앱의 플로우를 동시에 확인할 수 있다.
- 프로파일: 설치된 스마트 앱의 과거 플로우를 파악할 수 있다.

4. 구 현

본 연구에서는 Code Instrument를 이용해 기존 스마트 앱에 모니터를 위한 코드를 추가하고 스마트 앱이 실행되는 동안 스마트 앱의 플로우를 실시간으로 확인하며 프로파일을 통해 스마트 앱의 과거 플로우를 파악할 수 있는 모니터 기능을 구현하였다.

4.1 Code Instrument

Code Instrument는 스마트 앱 모니터링을 위해 기존 스마트 앱 코드에 필요한 코드를 추가하는 툴이다. Groovy 언어로 작성된 스마트 앱 코드에 대한 AST를 분석하여 그림 5와 같이 input으로 모니터 디바이스를 추가하고 이벤트 발생, 이벤트 핸들러 실행, 메소드 실행, 액션 실행 및 메시지 전송 부분에 각 정보를 모니터 디바이스로 전달하는 코드를 추가한다.

```
preferences {
    section("When I arrive..."){
        input "presence1", "capability.presenceSensor", title: "Who?", multiple: true
    }
    section("Unlock the lock..."){
        input "lock1", "capability.lock", multiple: true
    }
}
.....
def presence(evt)
{
    def anyLocked = lock1.count{it.currentLock == "unlocked"} != lock1.size()
    if (anyLocked) {
        sendPush "Unlocked door due to arrival of $evt.displayName"
        lock1.unlock()
    }
}
```

```
preferences {
    //Inserted Code
    section("Select SmartAppMonitor") {
        input "smartAppMonitor", "capability.execute"
    }
    ......
}
.....
def presence(evt)
{
    //Inserted Code
    smartAppMonitor.setData(app.getName(), "${evt.value}", "event", "${evt.getDevice()}", "event")
    //Inserted Code
    smartAppMonitor.setData(app.getName(), "presence", "handlerMethod", "this", "handlerMethod")
    def anyLocked = lock1.count{it.currentLock == "unlocked"} != lock1.size()
    if (anyLocked) {
    //Inserted Code
    smartAppMonitor.setData(app.getName(), "sendPush", "send", "this", "action")
        sendPush "Unlocked door due to arrival of $evt.displayName"
    //Inserted Code
    smartAppMonitor.setData(app.getName(), "unlock", "lock", "${lock1.getName()}", "action")
        lock1.unlock()
    }
}
```

그림4. 기존 스마트 앱 코드(위)와 변환된 코드(아래)[5]

4.2 모니터 디바이스 및 API 서버

모니터 디바이스는 SmartThings 디바이스 핸들러로 작성하였고 스마트 앱으로부터 플로우 정보를 받아 API 서버에 전달한다. SmartThings에서 제공하는 httpPost 라이브러리를 사용하여 API 서버에 각 정보를 보내면 API 서버는 데이터베이스에 정보를 저장한다.

4.3 실시간 모니터

실시간 모니터를 통해 스마트 앱의 플로우를 웹에서 확인할 수 있다. API 서버와 웹 클라이언트가 웹 소켓으로 통신하여 모니터 디바이스로부터 서버에 정보가 전달되면 서버는 클라이언트로 정보를 보낸다. 웹 클라이언트는 서버로부터 받은 정보를 시각적 형태로 변환하여 웹 페이지에 보여준다.

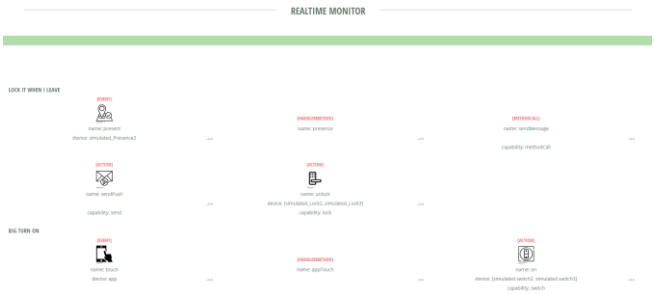

그림 5. 실시간 모니터

그림 5의 Lock It When I Leave 스마트 앱[5]은 사용자가 집에서 나갈 때 문을 자동으로 잠그는 앱이다. Lock It When I Leave 스마트 앱 예시에서 실시간 모니터를 통해 보여주는 플로우 정보는 다음과 같다.

- 이벤트: 이벤트가 발생한 디바이스 아이콘[6], ‘present’와 같은 이벤트 이름, ‘simulated_Presence’ 와 같이 이벤트가 발생한 디바이스 이름
- 이벤트 핸들러: ‘presence’와 같은 핸들러 이름
- 액션: 액션이 발생한 디바이스 아이콘[6], ‘sendPush’, ‘unlock’과 같은 액션 이름, ‘simulated_Lock’과 같은 디바이스 이름과 디바이스의 capability 정보

4.4 프로파일

프로파일은 과거 스마트 앱의 플로우 정보를 보여준다. 스마트 앱 리스트에서 스마트 앱을 선택하면 해당 스마트 앱의 과거 플로우 정보를 확인할 수 있다. 웹의 http request 기능을 이용하여 API 서버로 httpGet request 를 보내 스마트 앱 리스트를 가져와 보여준다. 스마트 앱 리스트를 선택하면 httpGet request의 파라미터로 스마트 앱의 이름을 전달하여 그 이름에 해당하는 스마트 앱의 플로우 정보를 가져와 보여준다. API 서버는 플로우 정보를 JSON 포맷으로 보내며 웹 페이지에서는 받은 정보를 파싱(Parsing)하여 그림 6과 같이 이를 시각적 형태로 변환한 후 웹 페이지에 보여준다.

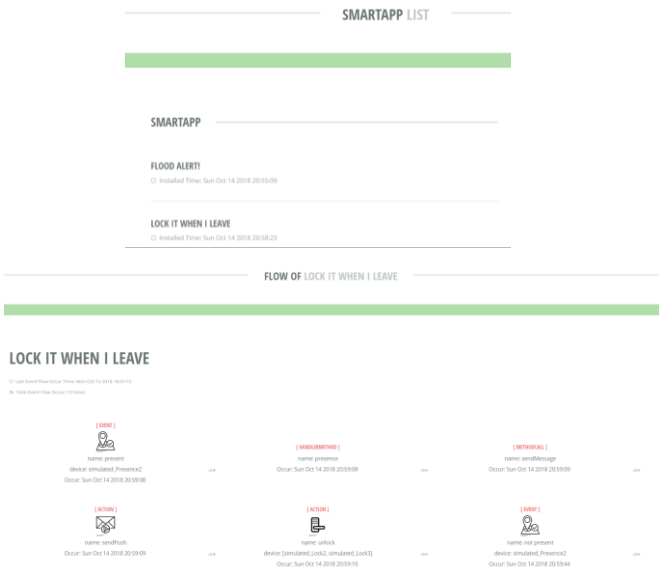

그림 6. 프로파일

프로파일에서는 실시간 모니터를 통해 확인할 수 있는 정보에 더해 시간 정보를 함께 보여주어 플로우 발생 시점을 파악할 수 있도록 구현하였다.

5. 결론 및 향후 연구

본 연구에서는 스마트 앱의 실제 동작 플로우를 시각적으로 확인할 수 있도록 code Instrument 및 스마트 앱 모니터를 구현하였다.

code Instrument 툴을 이용해 86개의 public 스마트 앱 중 59개에 대해서 정확히 code Instrument를 할 수 있었다. 향후 code Instrument 툴을 보완하여 더 많은 스마트 앱에 적용할 것이다. 또한, 스마트 앱 모니터의 시각적 효과를 높이기 위해 연관성이 높은 아이콘 정보를 제공하도록 구현할 것이다.

참고문헌

[1] SmartThings, SmartThings Developer Documentation, 2018
[2]. Y. Tian, N. Zhang, Y.-H. Lin, X. Wang, B. Ur, X. Guo, P. Tague, Smartauth: User-centered authorization for the internet of things, 2017
[3] DATADOG
https://www.datadoghq.com/
[4] Dynatrace
https://www.dynatrace.com/
[5] public 스마트 앱
https://github.com/SmartThingsCommunity/SmartThingsPublic/tree/master/smartapps/smartthings
[6] 디바이스 아이콘
https://thenounproject.com/