OneM2M 디바이스(&Cube) 플랫폼:

TAS 설치 및 실행 가이드

2015년 9월

경북대학교 통신프로토콜연구실

정중화, 강형우

godopu16@gmail.com, hwkang0621@gmail.com

요 약

사물인터넷 (Internet of Things: IoT)의 시대가 도래하고 있다. 다양한 분야에서 사물인터넷을 활용한 서비스가 개발되고 있다. 본 문서에서는 사물인터넷 국제표준인 oneM2M을 기반으로 만들어진 사물인터넷 디바이스 플랫폼인 &Cube에 센서 값을 전달하기 위한 TAS (Thing Adaptation S/W) 프로그램에 대한 설명 및 실행방법을 알아보도록 한다.

목 차

첨부:

Keti_tas_sample.zip

nCube_Lavender.zip

nCube_LavenderForWindow.zip

1. 서론

사물인터넷에 대한 관심이 증가하면서 다양한 분야에서 사물인터넷의 개념을 활용한 서비스들을 연구 및 개발하고 있다. 사물인터넷 환경에서는 각 센서의 정보를 서버로 전달하는 기능이 중요하다. 본 문서에서는 oneM2M기반의 사물인터넷 디바이스 플랫폼인 &Cube에서 센서의 데이터를 측정하여 서버 플랫폼인 OpenMobius로 전달하는 방법을 알아보도록 하겠다.

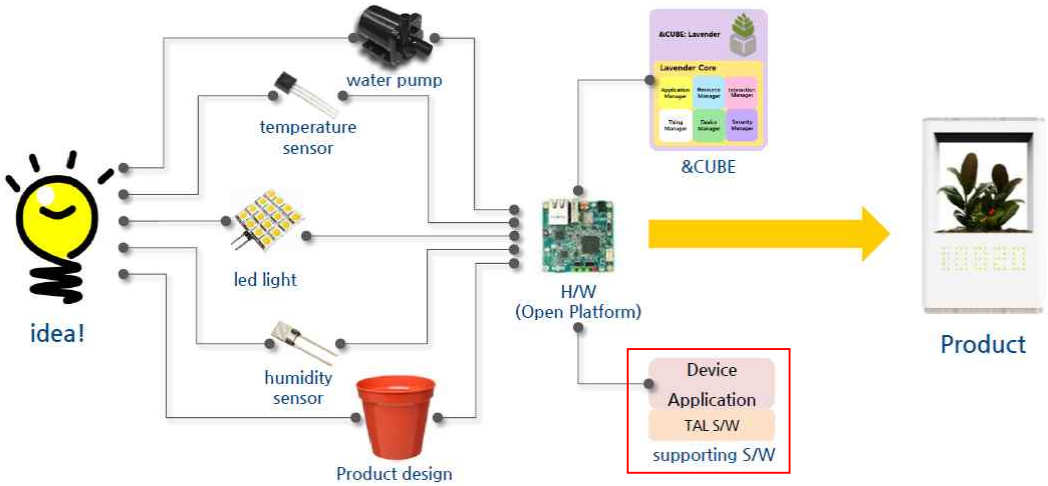

Figure 1. &Cube 플랫폼을 활용한 디바이스 개발 구조

&Cube에서는 TAS (Thing Adaptation S/W)라는 S/W를 활용하여 센서에서 값을 지속적으로 측정하고, 서버 플랫폼인 OpenMobius로 측정된 값을 전달, 사물을 제어하는 기능을 한다. TAS란 실제 사물을 디바이스에 연결하기 위한 S/W로써 사물과 &Cube간의 연결통로를 만드는 역할을 한다. 사용하고자 하는 사물의 API를 TAS에서 결정을 한다. TAS는 Figure 2와 같이 Thing Interworking Unit, Data Interworking Unit, &Cube Interworking Unit 이렇게 총 3가지 unit으로 구성되어 있다.

Thing Interworking Unit은 연결된 사물로부터 데이터를 수신하거나 사물로 데이터를 전송하는 기능을 한다. 사물 데이터 수신 시, Data Adaptation Unit을 호출하고 사물 제어를 위한 커맨드를 수신한다.

Data Adaptation Unit은 사물과 &Cube간 데이터를 인식 가능한 형태로 변환하는 기능을 수행한다. Thing Adaptation Unit으로부터 사물 데이터를 수신, 형태를 변환하고 &Cube Interworking Unit을 호출한다.

마지막으로 &Cube Interworking Unit은 &Cube로 사물 데이터를 전송하고 &Cube로부터

사물 제어 커맨드를 수신하는 기능을 수행한다. 사물 제어 커맨드 수신 시, Data Adaptation Unit으로 사물 제어 커맨드를 전송한다.

본 문서에서는 TAS를 어떻게 구성하고 개발할지에 대하여 설명을 하도록 하겠다. (&Cube 구동 및 설치방법은 oneM2M 기반의 디바이스 플랫폼 &Cube 설치 및 실행 가이드를 참고하길 바란다.)

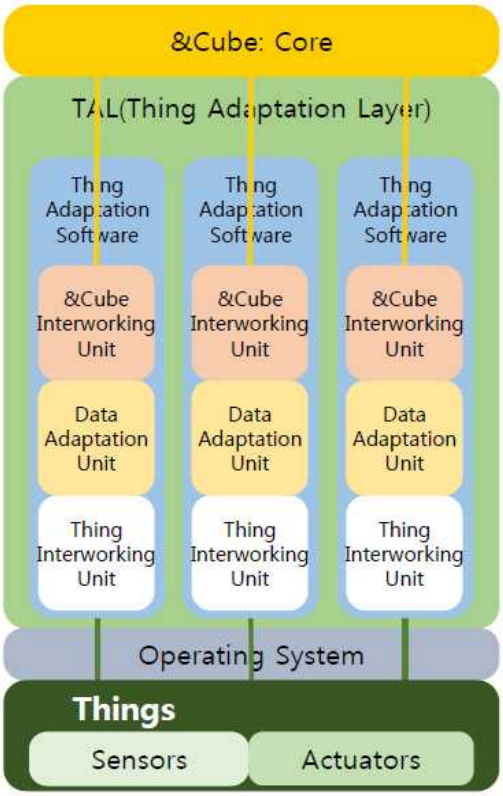

Figure 2. &Cube – TAS 연동 구조

2. TAS(Thing Adaptation S/W) 개발 실습

2.1 개발 환경 구성하기

2.1.1 Eclipse EE 다운로드

TAS를 개발하기 위해서 먼저 Eclipse를 설치해야 한다. http://www.eclipse.org/downloads/로 이동하여 Eclipse EE버전을 다운받는다.

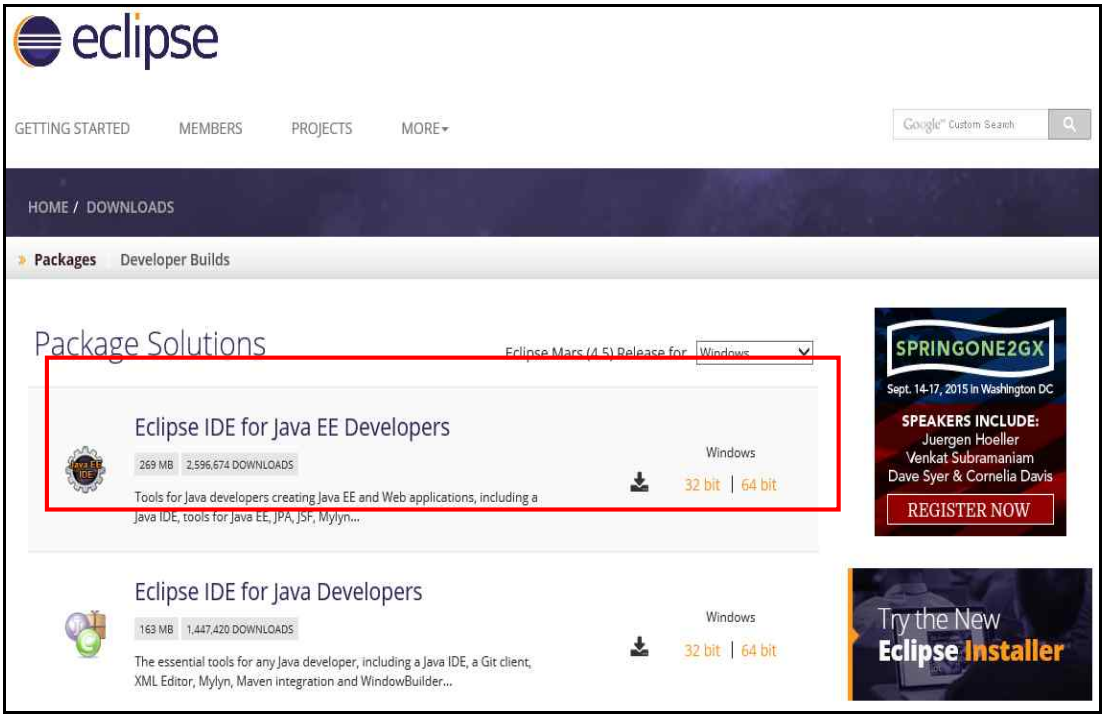

Figure 3. Eclipse EE 다운로드

2.1.2 Example source import

본 저자가 속해있는 연구실 홈페이지인 http://protocol.knu.ac.kr에 접속하여 keti_tas_sample.zip을 다운받는다. 다운받은 파일의 압축을 풀고 Eclipse 에서 [File -> Import -> General -> Existing Projects into Workspace]를 눌러 해당 파일을 import 한다.

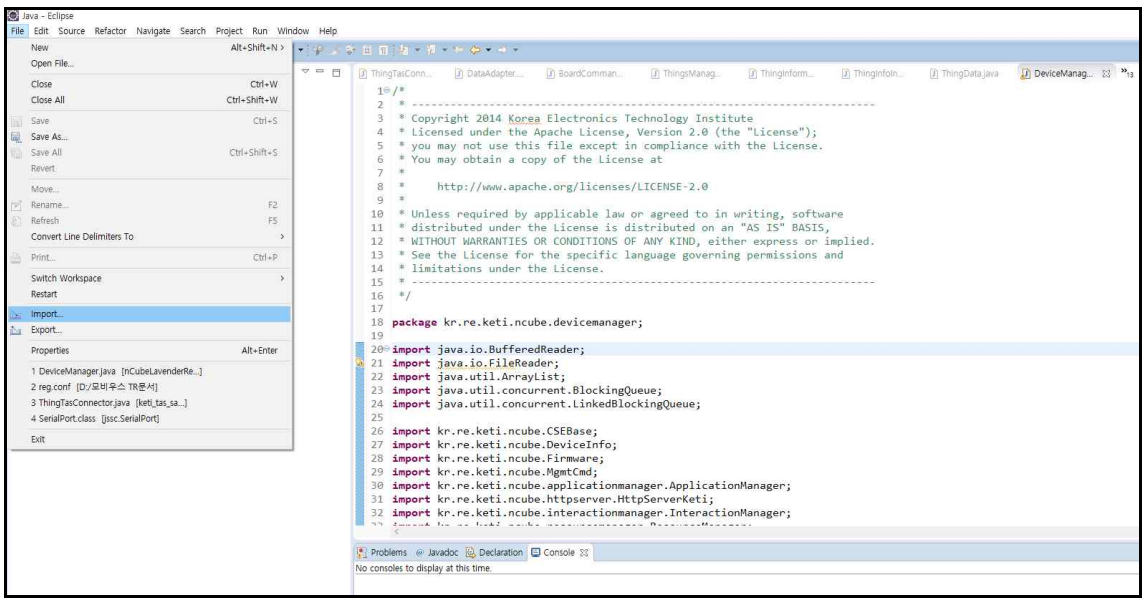

Figure 4. Import keti_tas_sample #1

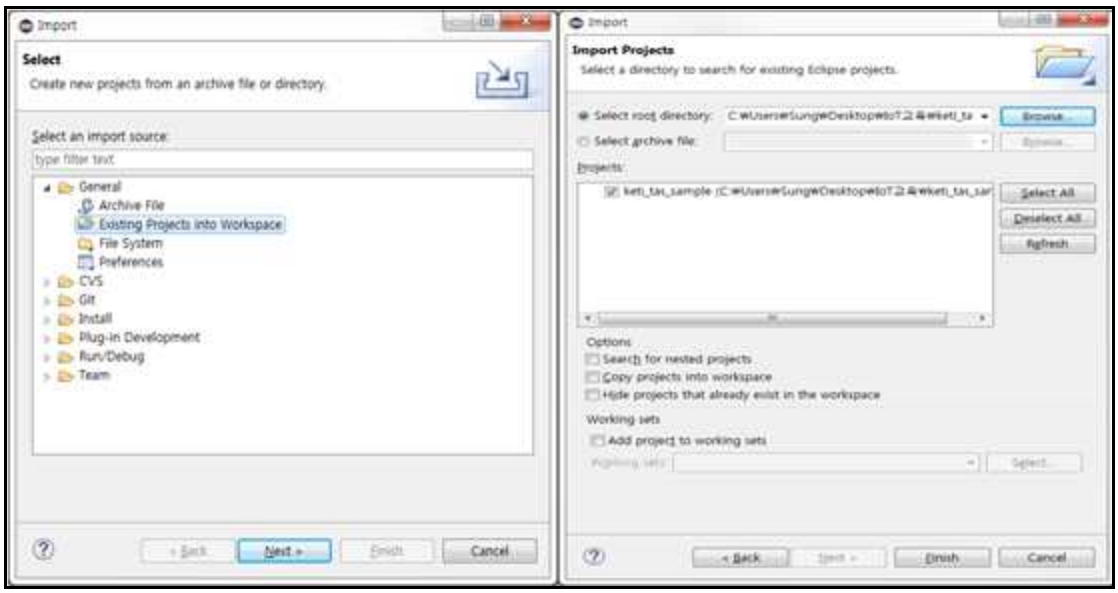

Figure 5. Import keti_tas_sample #2

2.1.3 Import TAS Library

TAS를 Import할 때, 에러가 발생한다면 아래 Figure 6과 같이 jssc.jar 파일을 같이 import하도록 한다.

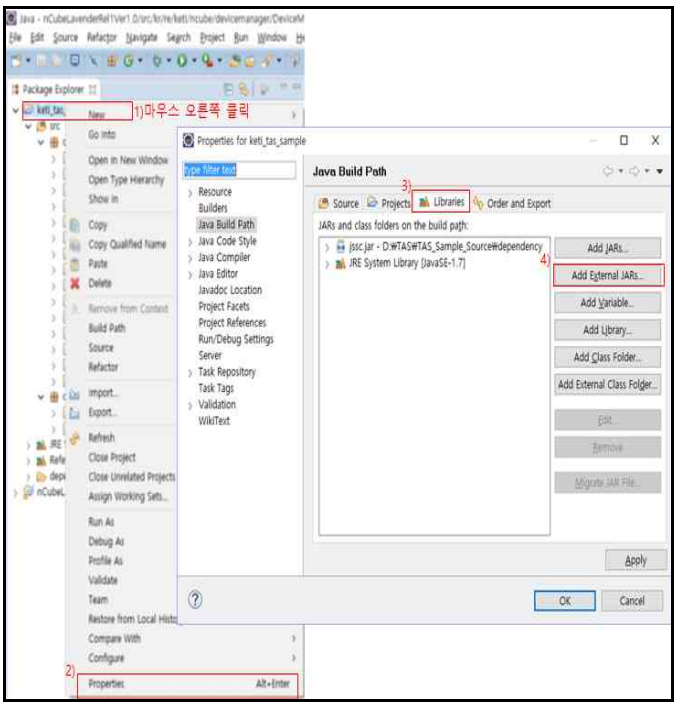

Figure 6. Import jssc Library

library 파일은 keti_tas_sample.zip파일을 압축해제를 한 폴더 내의 dependency 폴더에 있다.

2.2 TAS 구조 살펴보기

2.2.1 Main

모든 java 프로그램에는 main 에서 시작하므로 TAS 또한 main 메소드가 존재한다. Figure7, 8 에서 보이는 것과 같이 DataAdapter.java 에 가면 main 메소드를 볼 수 있다.

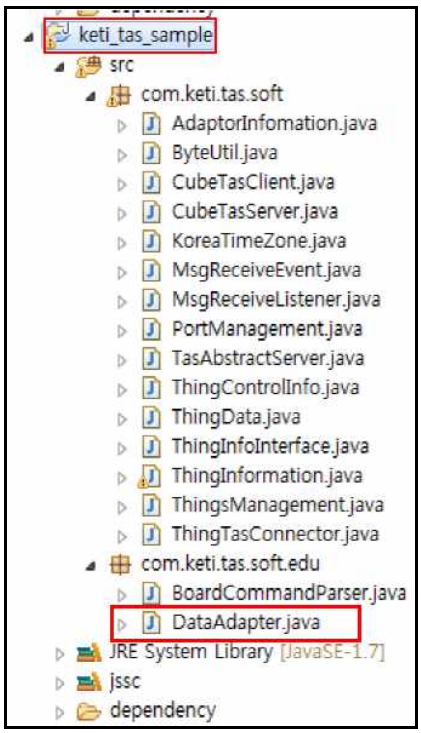

Figure 7. TAS main 메소드 #1

```
public class DataAdapter {

    public static void main(String argString[]) throws IOException, InterruptedException{

        final CubeTasServer cubeServer = new CubeTasServer();
        final ThingTasConnector thingServer = new ThingTasConnector();

        cubeServer.start();
        thingServer.start();

        AdaptorInfomation adaptor = new AdaptorInfomation(cubeServer.getServerPort());
        ThingsManagement manager = new ThingsManagement(adaptor);

        manager.addThing("ledstate", "sensoractor" ,"This is a control board!");
        manager.addControlThing("ledcontrol", "sensoractor", "This is a control board!", cubeServer.getServerPort());
```

Figure 8. TAS main 메소드 #2

2.2.2 Thing Adaptation S/W Registration

TAS를 실행시키게 되면 TAS 등록 절차를 거치게 됩니다. 이 절차에서는 AdaptorInformation, DataAdapter, ThingManagement 클래스가 관여하며, 이 클래스들은 최초 TAS를 &Cube 플랫폼에 등록한다. Figure 9와 같이 AdaptorInformation 클래스에 TAS 초기 정보를 입력하여 TAS 등록을 진행한다.

```
public class AdaptorInfomation {
        ……
        public AdaptorInfomation(int port){
                this.name = "Sample TAS";
                this.port = port;
                this.description = "Sample TAS for Test";
        }
        ……
}

public class DataAdapter {
        public static void main(String argString[])
                                        throws IOException, InterruptedException{
                ……
                AdaptorInfomation adaptor = new AdaptorInfomation(cubeServer.getServerPort());
                ThingsManagement manager = new ThingsManagement(adaptor);
                ……
        }
}

public class ThingsManagement{
        ……
        private void registAdaptor() {
                try {
                        adaptorInfomation.checkRegistMsg(client.requestToCube
                                                (adaptorInfomation.createRegistMsg()));
                }
                ……
        }
        ……
}
```

Figure 9. TAS Registration

TAS 초기 정보를 입력 후 Figure 10과 같이 등록 요청 메시지를 작성하여 &Cube 플랫폼으로 등록 메시지를 전송한다. (&Cube Port: 7591 (고정))

```
String tasRegist =
        "requestTASRegistration;" +
        "<TAS>" +
        "<TASProfile>" +
        "<Poc>" + port + "</Poc>" +
        "<Name>" + name + "</Name>" +
        "<Description>" + description + "</Description>" +
        "</TASProfile>" +
        "</TAS>“;
```

```
requestTASRegistration;
<TAS>
    <TASProfile>
        <Poc>45543</Poc>
        <Name>Sample Tas</Name>
        <Description>Sample Tas for Education</Description>
    </TASProfile>
</TAS>

Poc : Thing Adaptation S/W에서 요청 수신 대기중인 Socket port number
Name : Thing Adaptation S/W의 이름
Description : Thing Adaptation S/W의 간단한 설명
```

Figure 10. TAS Registration 요청 메시지

2.2.3 Thing Container & mgmtCmd Registration

TAS를 &Cube에 등록했다면, 사물의 데이터 저장용 Container와 사물을 컨트롤하기 위한 mgmtCmd를 &Cube에 등록해야 한다. 이와 같은 절차는 DataAdapter와 ThingManagement 클래스가 관여하고 있다.

```
public class DataAdapter {
        public static void main(String argString[])
                                throws IOException, InterruptedException{
                —
                manager.addThing("ledstate", "sensoractor" ,"This is a control board!");
                manager.addControlThing("ledcontrol", "sensoractor", "This is a control board!", cubeServer.getServerPort());
                —
        }
        —
}
public class ThingsManagement{
        —
        public void addThing(final String name, final String type, final String desc ){
                ThingInformation thing = new ThingInformation(name, type, desc);
                for(int i = 0; i < thingMasterTable.size(); i++) {
                        if(thingMasterTable.get(i).equals(thing))    return;
                }
                thingMasterTable.add(thing);
        }
        public void addControlThing(final String name, final String type, final String desc, final int port){
                ThingControlInfo thing = new ThingControlInfo(name, type, desc, port);
                for(int i = 0; i < thingControlMasterTable.size(); i++) {
                        if(thingControlMasterTable.get(i).equals(thing))        return;
                }
                thingControlMasterTable.add(thing);
        }
        —
}
```

Figure 11. Container 와 mgmtCmd 등록하기

Figure 11 과 같이 등록 메시지를 전송하게 되면 Figure 12, Figure 13 과 같은 메시지가 &Cube로 전송된다.

```
String thingRegist =
        "requestThingRegistration;" +
        "<TAS>" +
        "<ThingProfile>" +
        "<name>" + thingName + "</name>" +
        "<containerType>" + containerType + "</containerType>" +
        "<uploadCondition>no</uploadCondition>" +
        "<uploadConditionValue>no</uploadConditionValue>" +
        "</TAS>";
```

```
requestThingRegistration;
<TAS>
    <ThingProfile>
        <name>ledstate</name>
        <containerType>sensor</containerType>
        <uploadCondition>n</uploadCondition>
        <uploadConditionValue>n</uploadConditionValue>
    </ThingProfile>
</TAS>

name : Mobius에 등록할 사물의 이름
containerType : Mobius에 등록할 사물의 종류 예)sensor, actuator, …
uploadCondition&value : 차후 지원
** 등록 완료 후 containerId가 리턴됨
```

Figure 12. Thing Container Registration 요청 메시지

```
String msg =
          "requestThingControlRegistration;" +
          "<TAS>" +
          "<ThingControl>" +
          "<tasPoc>" + tasPort + "</tasPoc>" +
          "<name>" + thingName + "</name>" +
          "<description>" + thingDesc + "</description>" +
          "<cmdType>" + thingType + "</cmdType>" +
          "<execReqArgs>1</execReqArgs>" +
          "<execMode>1</execMode>" +
          "<execFrequency>0</execFrequency>" +
          "<execDelay>0</execDelay>" +
          "<execNumber>0</execNumber>" +
          "</ThingControl>" +
          "</TAS>";
```

```
requestThingControlRegistration;
<TAS>
    <ThingControl>
        <tasPoc>45543</tasPoc>
        <name>ledcontrol</name>
        <description>led on/off control</description>
        <cmdType>actuator</cmdType>
          ……
    </ThingControl>
</TAS>
```

name : Mobius에 등록할 제어 command의 이름
description : Mobius에 등록할 제어 command의 간단한 설명
cmdType : Mobius에 등록할 제어 command의 종류 예) actuator
exec**** : 차후 지원

Figure 13. Thing mgmtCmd Registration 요청 메시지

2.2.4 &Cube로 전송할 데이터 추출

지금까지 TAS, Thing(ThingContainer), ThingController(mgmtCmd)를 등록하였습니다. 이제 실제로 &Cube로 전송할 Data를 센서로부터 추출하여야 한다. ThingTasConnector.java 파일에서 Figure 14와 같은 소스들을 확인할 수 있다. 기본 예제의 경우 라즈베리 파이에 릴레이 보드를 연결하여 Serial 통신으로 데이터를 받은 후 그 데이터를 &Cube로 전송하도록 코드가 작성되어 있다. 예제코드를 수정하여 서버로 보낼 데이터를 추출 및 가공하도록 한다.

```
new Thread(new Runnable() {

                @Override
                public void run() {
                    try {

                        while(isActive){
                            String strData = "Hello-nCube";

                            activeReceiveEvent(strData);

                            Thread.sleep(5000);
                        }
                    } catch (Exception ex) {
                        Logger.getLogger(ThingTasConnector.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
            }).start();
```

Figure 14. ThingTasConnector.java source code

Figure 14의 소스코드는 5초에 1번씩 &Cube로 “Hello-nCube” 라는 문자열을 전송하는 코

드이다. 파일 안에 주석처리 해놓은 부분이 있는데, 해당 소스는 라즈베리 확장 쉴드인 RPino 쉴드를 이용하여, 센서로부터 값을 읽어 들인 후 &Cube로 센서로부터 받은 값을 전송하는 코드이다.

Figure 14의 가장 중요한 부분은 "activeReceiveEvent(strData)" 부분이며 해당 메소드에 &Cube로 전송할 문자열을 전달하면 이벤트 형식으로 문자열을 전달한다.

2.2.5 &Cube로 Data 전송

activeReceiveEvent() 메소드에 전달된 문자열은 MsgReceiveEvent 객체의 인스턴스 생성에 이용된다. 또 만들어진 Event는 MsgReceiveListener 에 전달된다.

Figure 15는 activeReceiveEvent 메소드로 전달된 문자열을 이용하여 MsgReceiveEvent를 만들고 event를 MsgReceiveListener로 전달하는 코드이다.

```
protected void activeReceiveEvent(String msg){
    if(listeners == null) {
        return;
    }
    MsgReceiveEvent event = new MsgReceiveEvent(this, msg);
    notifyListeners(event);
}

private void notifyListeners(MsgReceiveEvent event){
    Iterator<MsgReceiveListener> iter = listeners.iterator();
    while(iter.hasNext()){
        MsgReceiveListener listener = (MsgReceiveListener) iter.next();
        listener.receiveMsgEvent(event);
    }
}
```

Figure 15. MsgReceiveEvent 생성 및 전달

Figure 15의 notifyListeners 메소드를 보면 listeners Collection 에 있는 모든 Listener들에게 해당 Event를 전달하며 이벤트를 발생시키고 있는 것을 확인할 수 있다. MsgReceiveListener 클래스는 receiveMsgEvent() 메소드를 가진 인터페이스이기 때문에 직접 receiveMsgEvent()를 구현하고, listeners Collection에 넣어주어야 한다.

```
thingServer.addReceiveListener(new MsgReceiveListener() {

    @Override
    public void receiveMsgEvent(MsgReceiveEvent event) {
        String strData = event.getMessage();

        System.out.println(strData);

        ThingData mData = new ThingData();

        try {
            System.out.println("receive data : " + strData);
            ThingInformation thing = ThingsManagement.getThingIDByName("test_Thing");
            if (thing != null && thing.getRegistStatus())
                new CubeTasClient()
                        .upload(mData.makeThingDataMsg(thing.getThingName(), "test_Thing", strData));

        } catch (IOException ex) {
            Logger.getLogger(DataAdapter.class.getName()).log(Level.SEVERE, null, ex);
        }

    }
});
```

Figure 16. MsgReceiveListener 구현 및 등록

Figure 16은 main 메소드가 존재하는 DataAdapter 코드의 일부를 캡처하였다. Figure16을 보면 receiveMsgEvent 메소드를 구현한 후 thingServer 객체의 addReceiveListener() 메소드를 통해 해당 listener를 등록하고 있다. 여기서 thingServer는 ThingTasConnector 클래스의 객체이다.

또한, new CubeTasClient().upload()메소드를 통해 Figure 14에서 봤던 "Hello-nCube" 문자열을 전송하는 것을 확인할 수 있다 (센서의 측정 값을 &Cube로 전송).

Upload 메소드에 전달되는 파라미터로 mData.makeThingDataMsg() 메소드의 리턴 객체가 전달된다. 이 객체는 nCube로 보낼 데이터 형식을 지정해 주는 메소드다.

mData 객체는 ThingData 클래스의 객체이며 Figure17에서 확인할 수 있듯이 &Cube로 전송할 메시지 형식은 xml형식을 지니고 있다.

```
public String makeThingDataMsg(String name, String type, String content){
        String thingDataUpload =
            "requestThingDataUpload;" +
            "<TAS>" +
            "<ThingData>" +
            "<containerName>" + name + "</containerName>" +
            "<typeOfContent>" + type + "</typeOfContent>" +
            "<content>" + content + "</content>" +
            "<linkType>no</linkType>" +
            "</ThingData>" +
            "</TAS>";

        return thingDataUpload;
}
```

Figure 17. 전송 메시지 타입

3. TAS 실행하기

TAS를 테스트하기 위해 Runnable Jar 파일 (실행 파일)을 만들어야 한다. File -> export -> java -> Runnable Jar 를 눌러 Runnable Jar 파일을 생성하도록 한다. (Figure 18, 19을 참고)

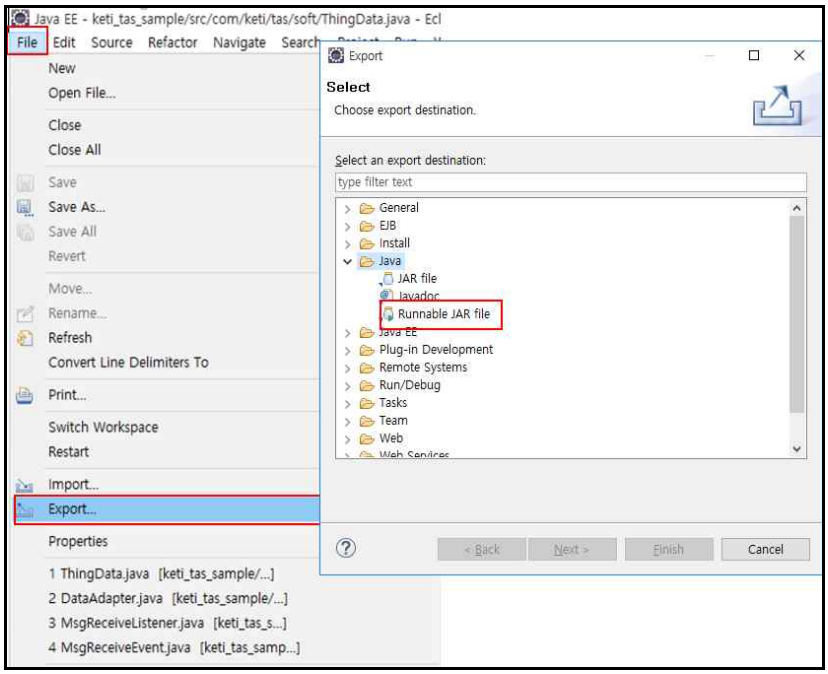

Figure 18. Runnable Jar 생성 #1

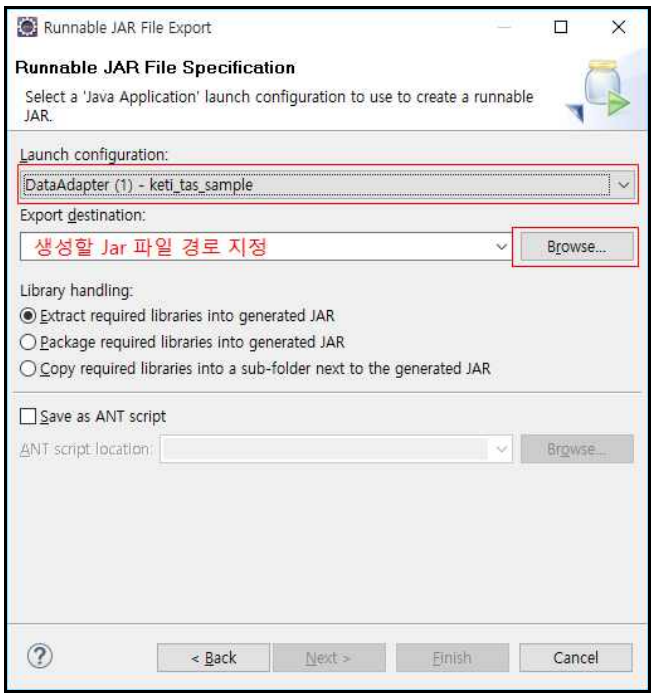

Figure 19. Runnable Jar 생성 #2

만들어진 jar 파일을 라즈베리 파이로 전송한 후 nCube_Lavender.jar 파일과 함께 /nCube 폴더에 넣는다. 이렇게 되면 /nCube 폴더에 reg.conf, nCube_Lavender.jar, tas_sample.jar 이렇게 총 3개의 파일이 존재하게 된다. 그 후 sudo java –jar nCube_Lavender.jar 명령어를 이용하여 nCube를 먼저 실행 한 후 sudo java –jar tas_sample.jar 명령어를 이용하여 TAS를 실행시킬 수 있다.

4. TAS 테스트

4.1 Robomongo 설치

TAS 를 통해 전송한 메시지가 서버로 잘 전달 되었는지 확인하기 위해 robomongo 프로그램을 사용한다. http://www.robomongo.org/download.html 로 이동하여 robomongo 프로그램을 다운로드 받아 설치하도록 한다. (Figure 20, 21 참조)

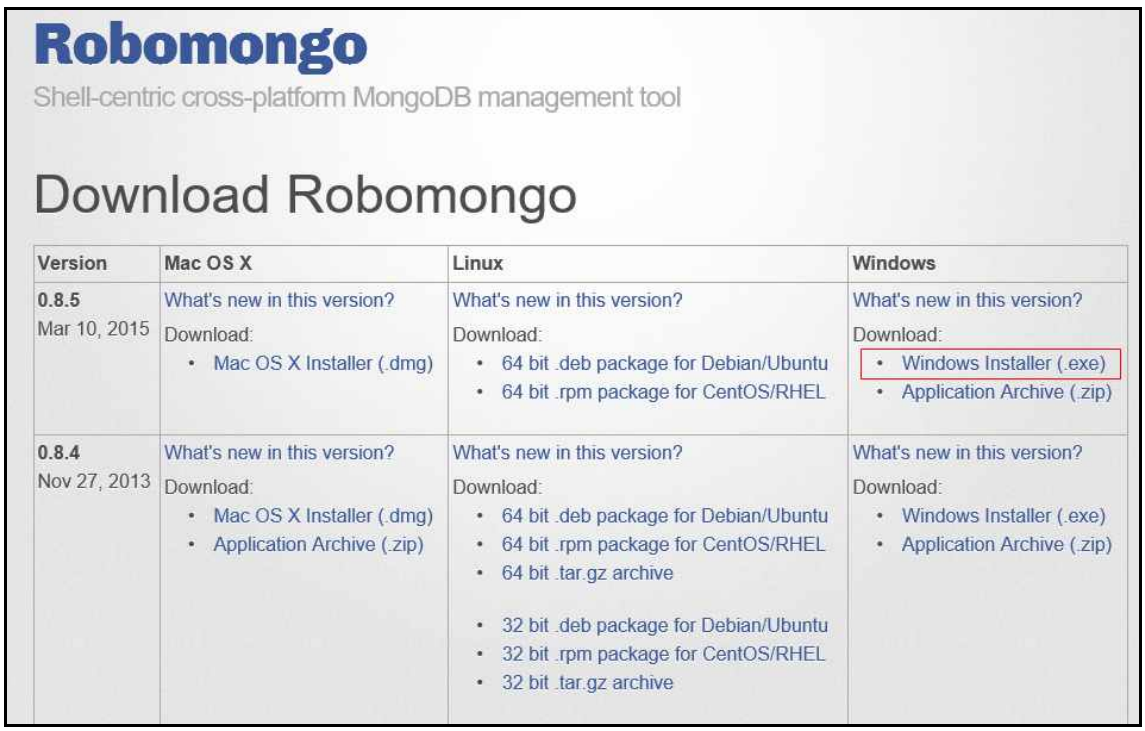

Figure 20. Robomongo 설치 #1

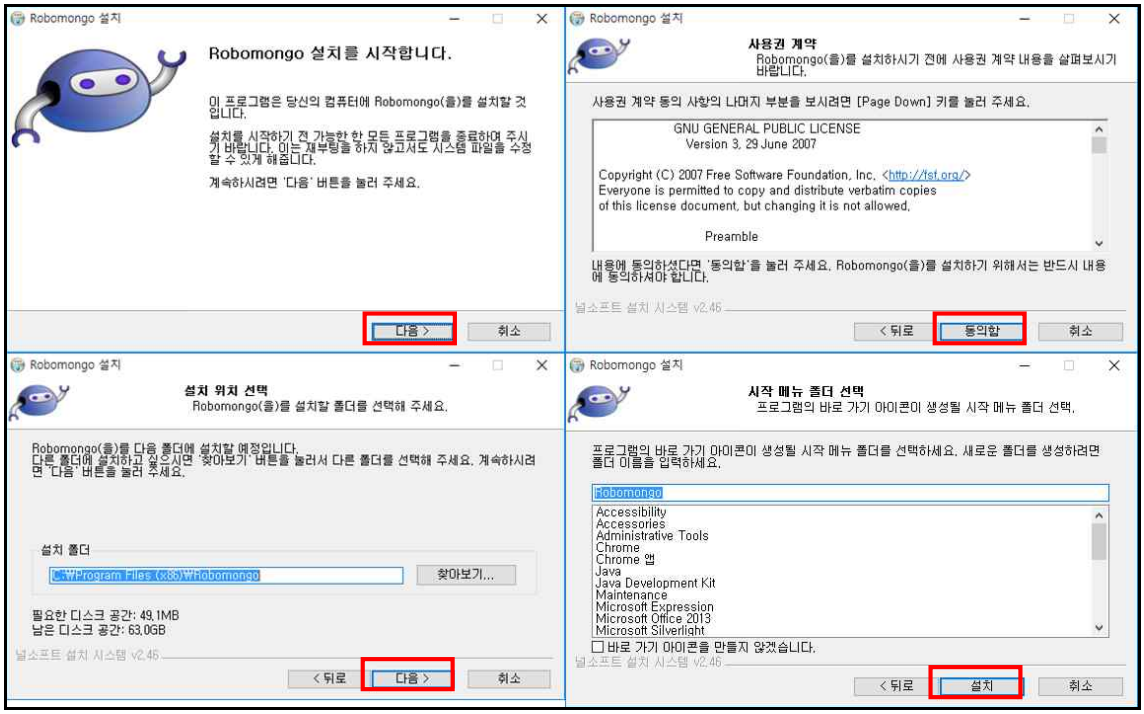

Figure 21. Robomongo 설치 #2

설치를 완료하셨으면 Robomongo 를 실행시키고 서버의 DB로 접속한다.

4.2 Robomongo 실행

Robomongo를 실행시키면 Figure 22와 같은 화면을 볼 수 있다. Figure 23을 참조하여 OpenMobius 서버 플랫폼으로 접속하도록 한다. (Figure 23의 IP주소는 OpenMobius 서버의 IP 주소이다.)

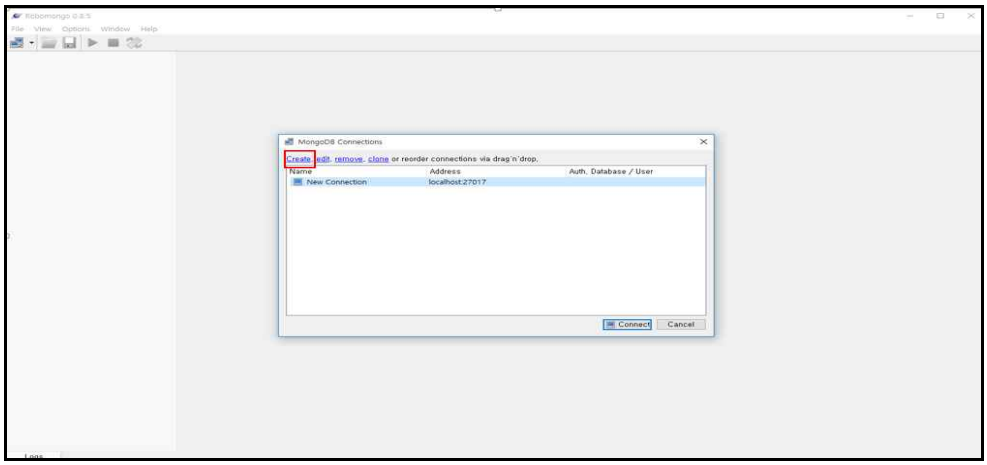

Figure 22. Robomongo 실행 #1

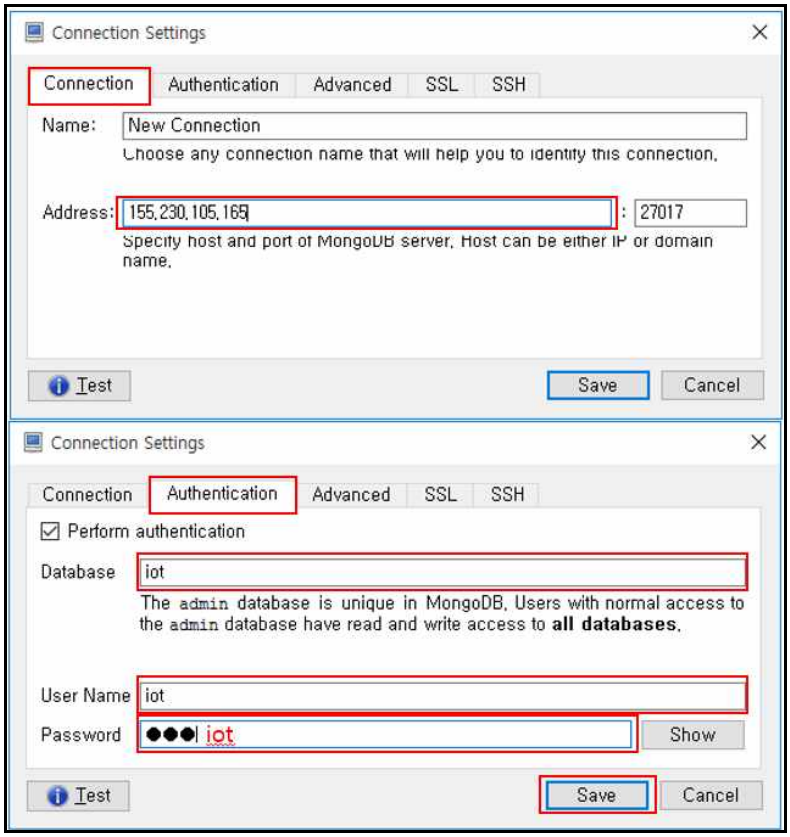

Figure 23. Robomongo 실행 #2

OpenMobius의 MongoDB에 접속하면 Figure 24와 같은 화면을 볼 수 있으며, contentInstance Collection을 보면 전송했던 TAS를 통해 &Cube로 전송했던 메시지들이 OpenMobius 서버에 잘 저장되어 있는 것을 확인할 수 있다.

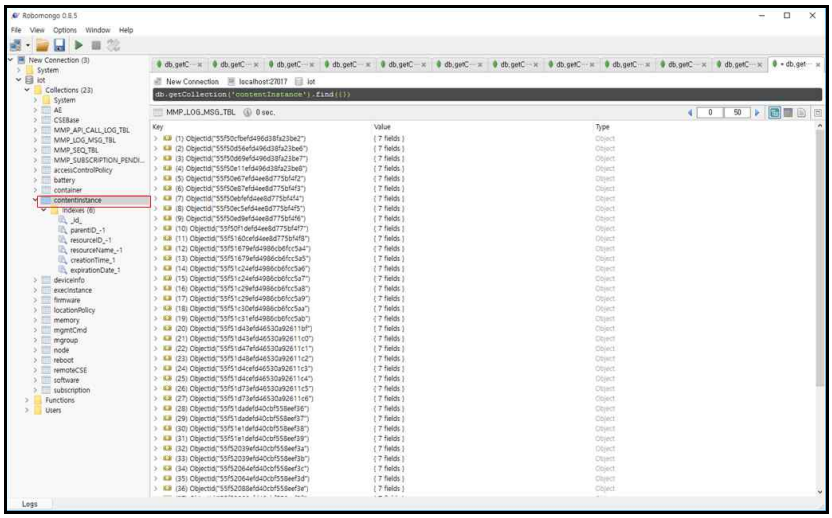

Figure 24. 전송 메시지 확인(contentInstance 확인)

참고로, OpenMobius에 대한 정보 및 설치방법은 oneM2M 기반의 서버 플랫폼 OpenMobius 설치 및 실행 가이드 문서에서 확인할 수 있다.

참고 문헌

[1] 사물인터넷 Alliance Ocean 홈페이지, http://www.iotocean.org/main/

[2] 사물인터넷 디바이스 개발 가이드 문서, &Cube User Guide-1.0.pdf