



## nCube-Thyme-nodejs

**Software Version: 2.5.0**

---

## TAS(led, sensor) Guide v1.0.0

Document Release Date: Mar 2023

Software Release Date: Mar 2023



## Revision History

버전	변경일자	제·개정 내역
1.0.0	2023-03-10	<p>초안</p> <p>wonseok@keti.re.kr / 정원석 hyeonseo0128@keti.re.kr / 손현서 <a href="mailto:hdkyon96@keti.re.kr">hdkyon96@keti.re.kr</a> / 허대근</p>

# 1. nCube-Thyme-Nodejs 소개

## 1.1. 개요

oneM2M 기반 플랫폼에서 AE(Application Entity) 이하의 리소스들을 만들고, 디바이스단의 장치들에 대한 데이터 수집 및 제어를 nCube-Thyme-Nodejs 및 TAS(Thing Adaptation Software)를 통해 수행한다.

본 가이드의 목적과 과정은 다음과 같다.

- 아래의 nCube-Thyme-Nodejs와 TAS를 이용한 실습을 통해 센서데이터를 Mobius(CSE)에 수집하며, led 와 같은 장치를 제어할 수 있다.
  - TAS에 연결된 센서가 MQTT 프로토콜을 사용하여 nCube-Thyme-Nodejs와 통신, 데이터를 주고받는 실습
  - nCube-Thyme-Nodejs를 통해 Subscription을 생성하고 notification 기능을 통해 LED를 제어하는 실습
  - 위의 두 실습을 통합하여 연결된 센서의 값에 따라 LED를 제어하는 실습

## 1.2. nCube-Thyme-Nodejs 구조

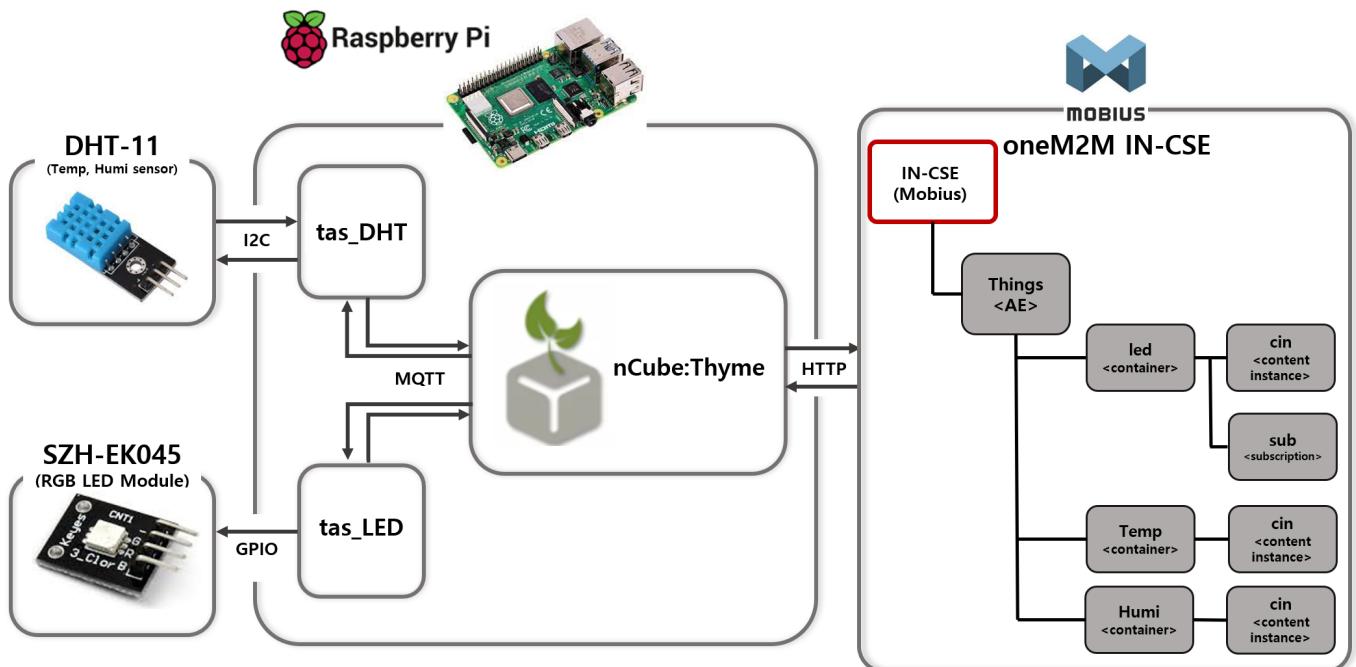


그림 1. nCube-Thyme-Nodejs & TAS를 이용한 oneM2M 플랫폼 서비스 구조도

## 2. 준비물

---

nCube-Thyme-Nodejs & TAS 실습을 위한 준비물은 아래와 같다.

### 2.1. H/W

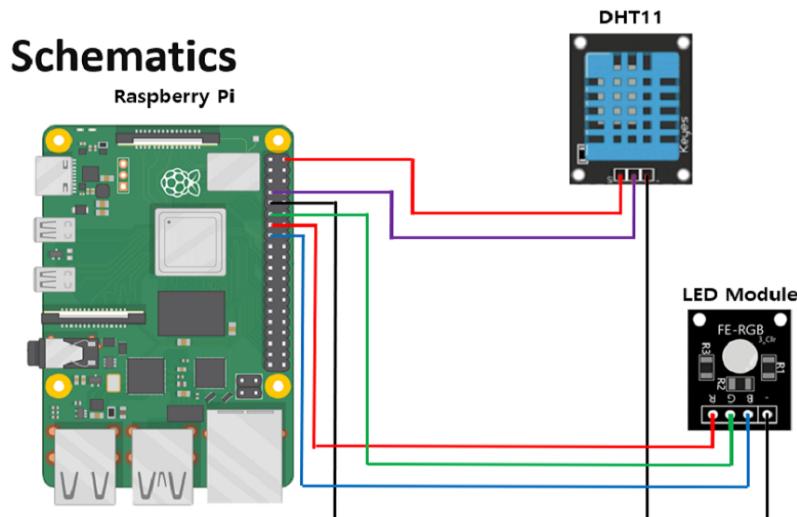
- Raspberry Pi (3 시리즈 이상)
- Micro SD Card (8GB 이상)
- KY-009 (RGB LED Module)
- DHT-11 (Temp, Humi sensor)

### 2.2. S/W

- Raspberry Pi Raspbian (buster 이상)
- Node.js (16.x 버전 이상)
- [Mobius](#)
- [nCube-Thyme-Nodejs](#)
- MQTT-broker

## 3. 장치 연결

---



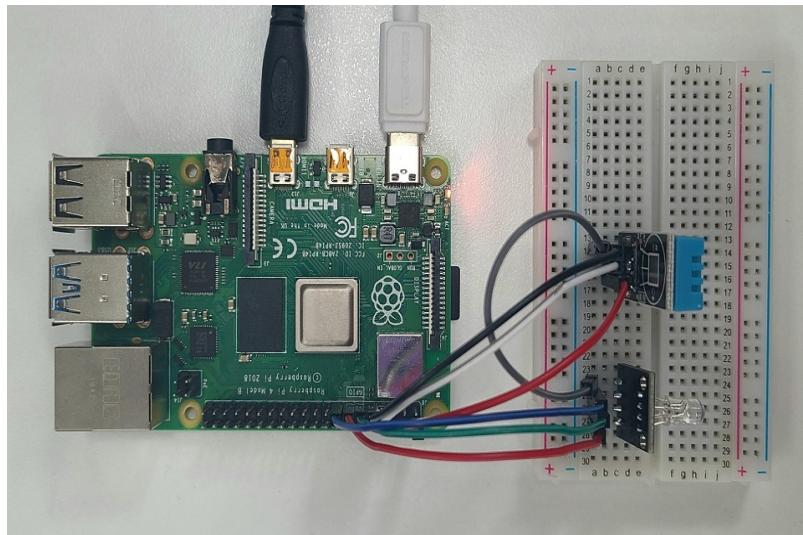


그림 2. 장치 연결

## 4. 실습 환경 구축

nCube-Thyme-Nodejs, TAS 구동에 필요한 구성요소들을 설치하여 실습 환경을 구축한다.

### 4.1. Node.js 설치

아래의 명령어를 통해 패키지 저장소(Personal Package Archive)에 Node.js에 대한 패키지를 업데이트합니다.

```
$ curl -sL https://deb.nodesource.com/setup_16.x | sudo -E bash -
```

위 명령어의 16.x를 원하는 버전으로 변경하여 설치 가능합니다.

```

pi@raspberrypi:~ $ curl -sL https://deb.nodesource.com/setup_16.x | sudo -E bash -
## Installing the NodeSource Node.js 16.x repo ...

## Populating apt-get cache ...
+ apt-get update
Hit:1 http://raspbian.raspberrypi.org/raspbian bullseye InRelease
Hit:2 http://archive.raspberrypi.org/debian bullseye InRelease
Reading package lists... Done

## Confirming "bullseye" is supported ...
+ curl -sLF -o /dev/null 'https://deb.nodesource.com/node_16.x/dists/bullseye/Release'
## Adding the NodeSource signing key to your keyring ...
+ curl -s https://deb.nodesource.com/gpgkey/nodesource.gpg.key | gpg --dearmor | tee /usr/share/keyrings/nodesource.gpg >/dev/null
## Creating apt sources list file for the NodeSource Node.js 16.x repo ...
+ echo 'deb [signed-by=/usr/share/keyrings/nodesource.gpg] https://deb.nodesource.com/node_16.x bullseye main' > /etc/apt/sources.list.d/nodesource.list
+ echo 'deb-src [signed-by=/usr/share/keyrings/nodesource.gpg] https://deb.nodesource.com/node_16.x bullseye main' >> /etc/apt/sources.list.d/nodesource.list

## Running `apt-get update` for you ...
+ apt-get update
Hit:1 http://archive.raspberrypi.org/debian bullseye InRelease
Hit:2 http://raspbian.raspberrypi.org/raspbian bullseye InRelease
Get:3 https://deb.nodesource.com/node_16.x bullseye InRelease [4,586 B]
Get:4 https://deb.nodesource.com/node_16.x bullseye/main armhf Packages [783 B]
Fetched 5,369 B in 2s (3,013 B/s)
Reading package lists... Done

## Run `sudo apt-get install -y nodejs` to install Node.js 16.x and npm
## You may also need development tools to build native addons:
    sudo apt-get install gcc g++ make
## To install the Yarn package manager, run:
curl -sL https://dl.yarnpkg.com/debian/pubkey.gpg | gpg --dearmor | sudo tee /usr/share/keyrings/yarnkey.gpg >/dev/null
echo "deb [signed-by=/usr/share/keyrings/yarnkey.gpg] https://dl.yarnpkg.com/debian stable main" | sudo tee /etc/apt/sources.list.d/yarn.list
sudo apt-get update & sudo apt-get install yarn

```

업데이트 된 패키지 버전의 Node.js를 설치합니다.

```
$ sudo apt-get install -y nodejs
```

```

pi@raspberrypi:~ $ sudo apt-get install -y nodejs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  libfuse2
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 24.2 MB of archives.
After this operation, 120 MB of additional disk space will be used.
Get:1 https://deb.nodesource.com/node_16.x bullseye/main armhf nodejs armhf 16.19.1-deb-1nodesource1 [24.2 MB]
Fetched 24.2 MB in 11s (2,107 kB/s)
Selecting previously unselected package nodejs.
(Reading database ... 106404 files and directories currently installed.)
Preparing to unpack .../nodejs_16.19.1-deb-1nodesource1_armhf.deb ...
Unpacking nodejs (16.19.1-deb-1nodesource1) ...
Setting up nodejs (16.19.1-deb-1nodesource1) ...
Processing triggers for man-db (2.9.4-2) ...

```

## 4.2. nCube-Thyme-Nodejs 다운로드 및 패키지 설치

GitHub에 배포된 nCube-Thyme-Nodejs를 git 명령어를 통해 다운로드합니다.

```
$ git clone https://github.com/IoTKETI/nCube-Thyme-Nodejs
```

```
pi@raspberrypi:~ $ git clone https://github.com/IoTKETI/nCube-Thyme-Nodejs
Cloning into 'nCube-Thyme-Nodejs' ...
remote: Enumerating objects: 4354, done.
remote: Counting objects: 100% (51/51), done.
remote: Compressing objects: 100% (38/38), done.
remote: Total 4354 (delta 26), reused 35 (delta 13), pack-reused 4303
Receiving objects: 100% (4354/4354), 12.97 MiB | 4.05 MiB/s, done.
Resolving deltas: 100% (1031/1031), done.
```

다운로드한 nCube-Thyme-Nodejs 폴더로 이동하여 실행을 위한 패키지를 설치합니다.

```
$ cd nCube-Thyme-Nodejs
```

```
$ npm install
```

```
pi@raspberrypi:~ $ cd nCube-Thyme-Nodejs/
pi@raspberrypi:~/nCube-Thyme-Nodejs $ npm install
added 139 packages, and audited 140 packages in 18s
13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New major version of npm available! 8.19.3 → 9.5.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.5.1
npm notice Run npm install -g npm@9.5.1 to update!
npm notice
```

### 4.3. Mosquitto MQTT broker 설치

Mosquitto 서명키(인증키)를 다운로드하고 추가합니다.

```
$ wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
$ sudo apt-key add mosquitto-repo.gpg.key
```

Mosquitto 저장소 패키지를 등록합니다. 아래 명령어에서 **buster**에 해당하는 부분은 Raspbian의 OS 버전에 따라 변경해야 합니다.

```
$ cd /etc/apt/sources.list.d/
$ sudo wget http://repo.mosquitto.org/debian/mosquitto-buster.list
```

Mosquitto MQTT 브로커를 설치합니다.

```
$ sudo apt-get update  
$ sudo apt-get install -y mosquitto
```

## 5. 환경 설정

TAS - nCube-Thyme-Nodejs - Mobius 간의 연결 및 리소스 생성을 위해 raspberry pi 4에 설치한 nCube-Thyme-Nodejs 구성 파일들의 환경변수 값을 변경해주어야한다. 아래의 각 파일, 변수 설명 및 가이드를 통해 환경변수 값을 수정한다.

raspberry pi의 raspbian은 기본적으로 Linux 기본 편집기 명령인 nano를 지원한다. 이에 가이드 도중 파일 수정을 위한 명령어는 다음과 같다.

```
$ nano [파일 명]
```

```
ex) $ nano conf.js
```

### 5.1. nCube-Thyme-Nodejs 환경 변수 설정

#### 5.1.1 conf.js

: oneM2M Mobius와의 연결을 위한 프로토콜 설정 및 nCube-Thyme-Nodejs를 통해 만들어줄 AE 이하의 리소스 정보를 설정하는 파일

nCube-Thyme-Nodejs에서 연결하고자 하는 oneM2M Mobius에 대한 정보와 Mobius아래 만들고자하는 AE에 대한 정보를 설정한다.

변수명	설명
<b>conf.useprotocol</b>	사용할 프로토콜 선택
<b>conf.sim</b>	가상 데이터 사용여부 선택

- cse object

변수명	설명
host	실행중인 Mobius의 주소
port	Mobius의 포트번호(Mobius 기본값 is 7579)
name	Mobius 이름
id	Mobius-id
mqttport	mqtt를 사용하기 위한 포트 번호
wsport	웹소켓을 사용하기 위한 포트 번호

- ae object

변수명	설명
ae_name	사용자가 설정할 ae의 이름
name	ae리소스의 이름
id	ae리소스의 id
parent	ae를 생성할 Mobius의 경로
appid	애플리케이션 식별자
port	notification을 받기 위한 포트
bodytype	리소스의 body 형식(json or xml)
tasport	TAS와 통신하기 위한 포트

```

conf.useprotocol = 'http'; // select one for 'http' or 'mqtt' or 'coap' or 'ws'

conf.sim = 'disable'; // enable or disable

// build cse
cse = {
    host      : 'localhost', // CSE host ip ex) 192.168.1.1
    port      : '7579',      // CSE http hosting port
    name      : 'Mobius',
    id        : '/Mobius2',
    mqttport: '1883',       //CSE mqtt broaker port
    wsport   : '7577',
};

// build ae
let ae_name = 'device_DEMO';

ae = {
    name      : ae_name,      // AE name to create
    id        : 'S'+ae_name,  // AE-ID
    parent    : '/' + cse.name,
    appid    : 'measure_co2',
    port      : '9727',
    bodytype: 'json',
    tasport  : '3105',
};

```

AE 리소스 생성 후 해당 AE 아래에 만들어주고자하는 장치의 cnt와 sub 설정 값을 설정한다.

- cnt\_arr array

변수명	설명
-----	----

변수명	설명
parent	생성할 cnt의 경로를 작성
name	생성할 cnt의 이름을 작성

- sub\_arr array

변수명	설명
parent	생성할 sub의 경로를 작성
name	생성할 sub의 이름을 작성
nu	알림을 받을 주소와 bodytype 작성

```

cnt_arr = [
  {
    parent: '/' + cse.name + '/' + ae.name,
    name: 'temp',
  },
  {
    parent: '/' + cse.name + '/' + ae.name,
    name: 'humi',
  },
  {
    parent: '/' + cse.name + '/' + ae.name,
    name: 'led',
  },
];
// build sub

sub_arr = [
  {
    parent: cnt_arr[2].parent + '/' + cnt_arr[2].name,
    name: 'sub1',
    nu: 'mqtt://' + cse.host + ':' + cse.mqttport + '/' + ae.id + '?ct=json',
    // 'http://' + ip.address() + ':' + ae.port + '/noti?ct=json',
  },
];

```

nCube-Thyme-Nodejs 모듈에서 구동되어 MQTT broker에 nCube를 연결해줄 TAS에 대한 설정 값을 설정한다.

- tas object

변수명	설명
client	현재 nCube-Thyme-Nodejs의 TAS에 대한 연결 상태, 기본값=false
host	nCube-Thyme-Nodejs와 연결할 MQTT broker의 주소

변수명	설명
port	nCube-Thyme-Nodejs와 연결할 MQTT broker의 구동 포트

```
let tas = {
  client: {
    connected: false,
  },
  connection: {
    host: 'localhost', // nCube MQTT broker ip
    port: 1883, // MQTT broker port
    endpoint: '',
    clean: true,
    connectTimeout: 4000,
    reconnectPeriod: 4000,
    clientId: 'thyme_' + nanoid(15),
    username: 'keti_thyme',
    password: 'keti_thyme',
  },
};
```

### 5.1.2 thyme\_tas.js

: nCube-Thyme-Nodejs 모듈에서 mqtt broker를 통해 주고받는 message에 따라 nCube-Thyme-Nodejs의 동작을 지정하는 파일

변수명	설명
getDataTopic	MQTT Client 생성 후 subscription으로 사용할 MQTT Topic을 작성
setDataTopic	MQTT Client 생성 후 publication으로 사용할 MQTT Topic을 작성

```
let getDataTopic = { // sub topic
  temp: '/thyme/temp',
  humi: '/thyme/humi'
};

let setDataTopic = { // pub topic
  led: '/led/set',
};
```

nCube-Thyme-Nodejs가 MQTT broker를 통해 구독, 발행하는 message에 대해 해당 message를 어떻게 처리할 것인지 설정한다.

다음의 예시는 설정한 "getDataTopic", "setDataTopic" 변수에 따라 해당 Topic에 맞춰 mqtt message를 발행, 구독하도록 설정한 다음, "humi" 데이터에 따라 led cnt의 cin 생성을 하여 led를 제어한다.

```

conf.tas.client.on('message', (topic, message) => {
    let content = null;
    let parent = null;
    /* USER CODES */
    let act = null;
    if(topic === getDataTopic.temp) {
        parent = conf.cnt[1].parent + '/' + conf.cnt[0].name;
        let curTime = moment().format();
        let curVal = parseFloat(message.toString()).toFixed(1);
        content = {
            t: curTime,
            v: curVal
        };
    }
    else if(topic === getDataTopic.humi) {
        parent = conf.cnt[1].parent + '/' + conf.cnt[1].name;
        let curTime = moment().format();
        let curVal = parseFloat(message.toString()).toFixed(1);
        content = {
            t: curTime,
            v: curVal
        };
    }
}

if((content !== null) && (topic === getDataTopic.humi) && (content.v > 50)){
    act = 1;
    console.log("light type set humi :", content.v);
}
else if((content !== null) && (topic === getDataTopic.humi) && (content.v < 50)){
    act = 3;
    console.log("light type set humi :", content.v);
}

if(content !== null) {
    onem2m_client.create_cin(parent, 1, JSON.stringify(content), this,
function (status, res_body, to, socket) {
        console.log('x-m2m-rsc : ' + status + ' -----');
    });
    if(act !== null){
        parent = conf.cnt[1].parent + '/' + conf.cnt[2].name;
        let curTime = moment().format();
        let act_content ={
            t: curTime,
            v: act
        };
        onem2m_client.create_cin(parent, 1, JSON.stringify(act_content), this,
function (status, res_body, to, socket) {
            console.log('x-m2m-rsc : ' + status + ' -----');
        });
    }
}
}

```

```
/* */  
});
```

### 5.1.3. app.js

: nCube-Thyme-Nodejs 모듈에서 onem2m\_client를 통해 주고받는 HTTP 명령에 따라 nCube-Thyme-Nodejs의 동작을 지정하는 파일

Mobius의 cnt에 sub를 생성하여 해당 sub를 통한 notification 동작이 수행 될 시 nCube-Thyme-Nodejs의 동작을 설정한다.

아래의 예시는 5.1.2.의 예시를 통해 생성된 led cnt의 cin이 있을 시 led cnt의 sub가 생성하는 notification을 통해 thyme\_tas 단으로 led의 제어 명령을 전달한다.

```
onem2m_client.on('notification', function (source_uri, cinObj) {  
  
    console.log(source_uri, cinObj);  
  
    var path_arr = source_uri.split('/')  
    var event_cnt_name = path_arr[path_arr.length-2];  
    var content = cinObj.con;  
  
    /* ***** USER CODE ***** */  
    if(event_cnt_name === 'led') {  
        // send to tas  
        thyme_tas.send_to_tas(event_cnt_name, content.v);  
    }  
    /* */  
});
```

## 5.2. TAS 환경 변수 설정

### 5.2.1. tas\_LED.js

: raspberry pi에 연결된 led의 pin 값 및 동작을 설정하여 nCube-Thyme-Nodejs에서 발행하여 MQTT broker를 통해 들어오는 명령과 설정 환경 값에 따라 led를 제어하는 파일

Gpio pin으로 제어중인 led에 대한 연결 및 led 작동 타입(R, G, B)에 대해 설정한다.

```
let LED_G = new Gpio(17, 'out');           // led Gpio pin num set R=27, G=17,  
B=18  
let LED_B = new Gpio(18, 'out');  
let LED_R = new Gpio(27, 'out');  
  
function led_control(light_type){  
    if(light_type === 1){  
        LED_R.writeSync(1);  
        LED_G.writeSync(0);  
    }
```

```

        LED_B.writeSync(0);
    }
    else if(light_type === 2){
        LED_R.writeSync(0);
        LED_G.writeSync(1);
        LED_B.writeSync(0);
    }
    else if(light_type === 3){
        LED_R.writeSync(0);
        LED_G.writeSync(0);
        LED_B.writeSync(1);
    }
    else{
        all_Gpio_off();
        console.log("not defined type - led off")
    }
}

function all_Gpio_off(){
    LED_R.writeSync(0);
    LED_G.writeSync(0);
    LED_B.writeSync(0);
}

```

## 5.2.2 tas\_DHT.js

: raspberry pi에 연결된 DHT-11센서의 pin 값 및 해당 센서를 통해 얻는 데이터가 보내질 MQTT Topic을 설정하여 nCube-Thyme-Nodejs로 센서 데이터를 전송하는 파일

사용하고자 하는 DHT센서의 타입과 raspberry pi에 연결된 Gpio pin 번호를 설정하고, 해당 센서로 부터 들어오는 "temp", "humi" 데이터를 어떤 MQTT topic을 사용하여 nCube-Thyme-Nodejs로 보낼지 설정한다.

```

let DHT_sensor = {
    type: 11,
    GPIO_PIN: 4
}

let sendDataTopic = {
    temp: '/thyme/temp',
    humi: '/thyme/humi'
};

```

아래는 설정한 DHT-11 센서 및 MQTT Topic에 기반하여 3초마다 수집한 DHT-11 센서의 수집 데이터를 nCube-Thyme-Nodejs에 전달하기 위해 message화 하여 MQTT broker로 publish하는 동작을 예시로 작성하였다.

```

/* USER CODE */
/* DHT sensing interval */
setInterval(() => {

```

```
sensor.read(DHT_sensor["type"], DHT_sensor["GPIO_PIN"], function(err,  
temperature, humidity) {  
    if(!err) {  
        con_body = {  
            "temp": temperature,  
            "humidity": humidity  
        }  
        console.log(`DHT sensing result [temp: ${temperature}°C, humidity:  
${humidity}%]`);  
        doPublish(sendDataTopic['temp'], temperature.toString());  
        doPublish(sendDataTopic['humi'], humidity.toString());  
    }  
    else {  
        console.log("DHT_sensing err :", err);  
    }  
});  
, 3000);  
  
/* */
```

# 6. 실습

위의 환경 및 설정 사항들을 모두 수정, 충족하였다면 아래의 실습을 통해 nCube-Thyme-Nodejs와 TAS 연동을 통한 oneM2M 플랫폼에서의 장치 데이터 수집 및 제어 실습을 수행한다.

## 6.1. 실습 1

: MQTT 프로토콜을 사용하여 TAS와 nCube-Thyme-Nodejs 간 DHT-11 센서의 센싱 데이터를 주고받는 실습

nCube-Thyme-Nodejs를 설치한 디렉토리로 이동 후 nCube-Thyme-Nodejs를 구동하기 위해 thyme.js 파일을 실행한다.

```
$ node thyme.js
```

설정한 ae-cnt-sub를 Mobius에 구축해주는 과정을 거친 후 nCube-Thyme-Nodejs의 정상 작동 시 다음과 같은 출력을 확인한다.

```
keti@raspberrypi:/home/nCube-Thyme-Nodejs $ sudo node thyme.js
[status] : crtac
/Mobius
{ "m2m:rsc": "resource is already exist" }
x-m2m-rsc : 4105 <----
[status] : rtvae
/Mobius/device_DEMO
x-m2m-rsc : 2000 - Sdevice_DEMO <----
[status] : crtct
{"m2m:cnt": {"rn": "temp", "lbl": ["temp"]}}
/Mobius/device_DEMO
0 - /Mobius/device_DEMO/temp - x-m2m-rsc : 4105 <----
{ "m2m:rsc": "resource is already exist" }
{"m2m:cnt": {"rn": "humi", "lbl": ["humi"]}}
/Mobius/device_DEMO
1 - /Mobius/device_DEMO/humi - x-m2m-rsc : 4105 <----
{ "m2m:rsc": "resource is already exist" }
{"m2m:cnt": {"rn": "led", "lbl": ["led"]}}
/Mobius/device_DEMO
2 - /Mobius/device_DEMO/led - x-m2m-rsc : 4105 <----
{ "m2m:rsc": "resource is already exist" }
[status] : delsub
/Mobius/device_DEMO/led/sub1
0 - /Mobius/device_DEMO/led/sub1 - x-m2m-rsc : 2002 <----
{
  "m2m:sub": {
    "pi": "3-20230303043715836435",
    "rl": "23-20230303080356388806",
    "ty": 23,
    "ct": "20230303T0800356",
    "rn": "sub1",
    "lt": "20230303T0800356",
    "et": "20250303T0800356",
    "enc": { "net": [Array] },
    "exc": 100,
    "nu": [ "mqtt://[REDACTED]:1883/Sdevice_DEMO?ct=json" ],
    "nct": 2,
    "cr": "Sdevice_DEMO"
  }
}
[status] : crtsub
{"m2m:sub": {"rn": "sub1", "enc": {"net": [1,2,3,4]}, "nu": ["mqtt://
/Mobius/device_DEMO/led
[mqtt_connect]-noti_topic : /oneM2M/reg/+/$device_DEMO/#
0 - /Mobius/device_DEMO/led/sub1 - x-m2m-rsc : 2001 <----
{"m2m:sub": {"rn": "sub1", "ty": 23, "pi": "3-20230303043715836435",
,"et": "20250303T081135", "nu": ["mqtt://[REDACTED]:1883/Sdevice_DEMO"]}}
localhost Connection succeeded!
Subscribe to topics ( /thyme/temp )
{"op": 5, "rq": "VILVoAe72M", "to": "mqtt://[REDACTED]:1883/Sdevice_DEMO/led/sub1", "cr": "Sdevice_DEMO"})
[mqtt_noti_action] received notification of verification
Subscribe to topics ( /thyme/humi )
[status] : crtcl
```

이후 tas\_DHT.js를 실행하여 DHT-11 Sensor로부터 온습도 데이터를 센싱하고, 이를 nCube-Thyme-Nodejs에 전달하여 Mobius의 cnt-cin으로 데이터를 적측한다.

```
$ node tas_DHT.js
```

```

keti@raspberrypi:/home/nCube-Thyme-Nodejs/tas_sample $ node tas_DHT.js
localhost Connection succeeded!
DHT sensing result [temp: 26°C, humidity: 18%]

```

Mobius 모니터링을 통해 각 센서 값 CNT에 데이터가 cin으로 적측됨을 확인한다.

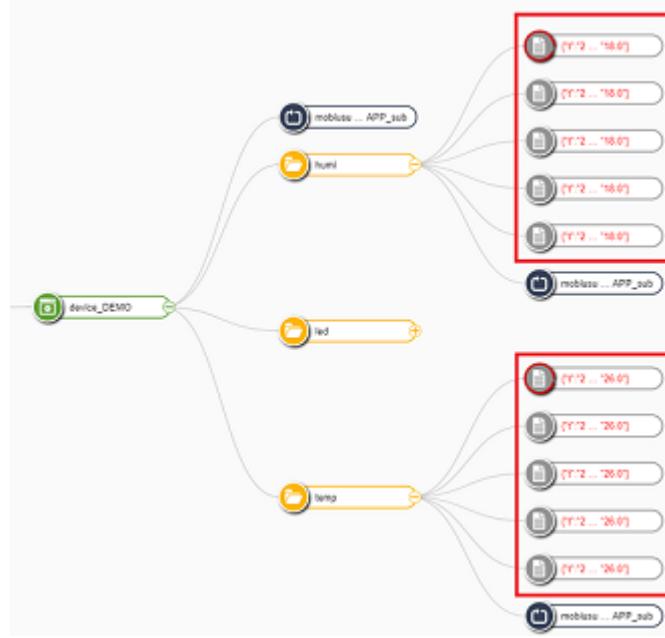


그림 3. oneM2M 리소스브라우저를 통해 확인한 cin 생성

## 6.2. 실습 2

: nCube-Thyme-Nodejs를 통해 sub를 생성하고 notification 기능을 통해 LED를 제어하는 실습

실습 1과 동일하게 thyme.js 파일의 실행을 통해 nCube-Thyme-Nodejs가 실행 중임을 가정한다.

tas\_LED.js를 실행, led의 cnt에 cin이 생성될 때 led cnt-sub의 notification 동작을 통해 MQTT message를 읽어들여 각 밝기 타입 1(R), 2(G), 3(B)에 따라 실제 LED의 동작을 확인한다.

```
$ node tas_LED.js
```

```

keti@raspberrypi:/home/nCube-Thyme-Nodejs/tas_sample $ node tas_LED.js
localhost Connection succeeded!
Subscribe to topics ( /led/set )
led light type : 1
led light type : 2
led light type : 3

```

```

1 {
2   ...
3   "m2m:cin": {
4     ...
5     "con": {"t": "20230227T043336", "v": 1}
6   }
7 }

```

그림 4. postman을 통해 led cnt-cin에 원하는 led 동작 데이터를 담은 con을 post

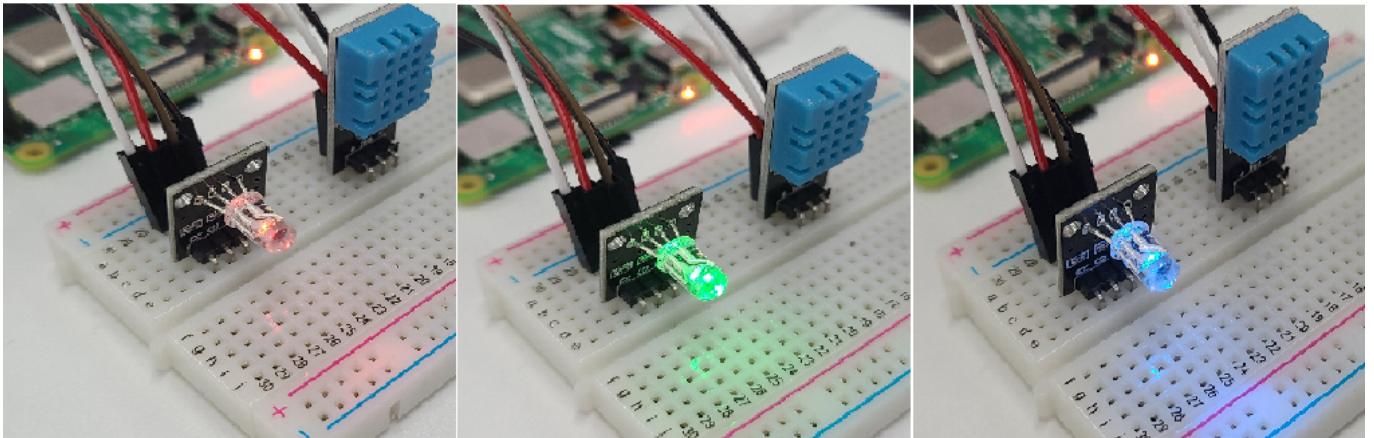


그림 5. 그림 4 과정을 통해 led 동작 경우마다 달라지는 led의 동작

### 6.3. 실습 3

: 위의 두 실습을 통합하여 연결된 센서의 센싱 데이터에 따라 led를 제어하는 실습

- nCube-Thyme.js
- tas\_LED.js
- tas\_DHT.js

를 모두 실행하여 DHT를 통해 읽은 습도 데이터가 50 보다 클 경우 led를 red로, 50 보다 작을 경우 led를 blue로 제어한다.

```
$ node Thyme.js
$ node tas_LED.js
$ node tas_DHT.js
```

습도가 50 작을 경우 분류되는 light type과 적측 humi 데이터 그리고 sub를 통한 con 값(=1)

```

localhost Connection succeeded!
Subscribe to topics ( /led/set )
led light type : 3
led light type : 3
led light type : 3

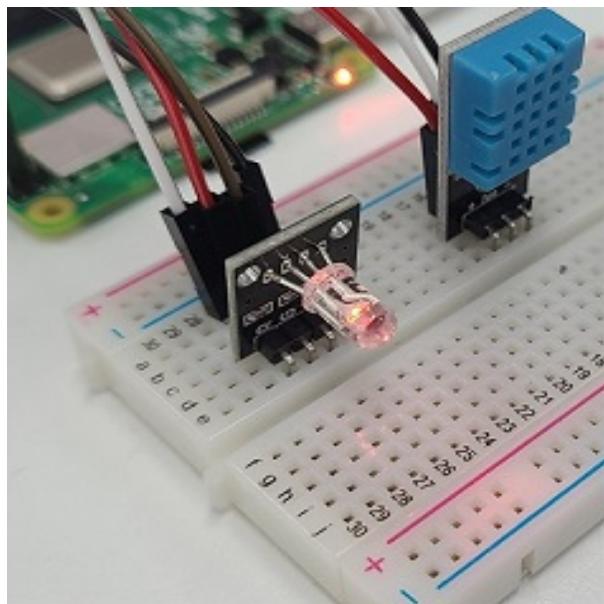
```

```

localhost Connection succeeded!
DHT sensing result [temp: 26°C, humidity: 18%]
DHT sensing result [temp: 26°C, humidity: 18%]
DHT sensing result [temp: 26°C, humidity: 18%]

```

```
{
  "op": 5,
  "rqi": "pnwQ18LBCE",
  "to": "mqtt://[REDACTED]:1883/SDevice_DEMO?ct=json",
  "fr": "/Mobius2",
  "pc": {
    "m2m:sgn": {
      "sur": "Mobius/Device_DEMO/led/sub1",
      "nev": {
        "rep": {
          "m2m:cin": {
            "rn": "4-20230303093019781",
            "ty": 4,
            "pi": "3-20230303092458614092",
            "ri": "4-20230303093019781724",
            "ct": "20230303T093019",
            "lt": "20230303T093019",
            "st": 3,
            "et": "20250303T093019",
            "cs": 1,
            "con": "3",
            "cr": "SS-ehqpdwfl"
          }
        },
        "net": 3
      },
      "rvi": "2a"
    }
  }
}
```

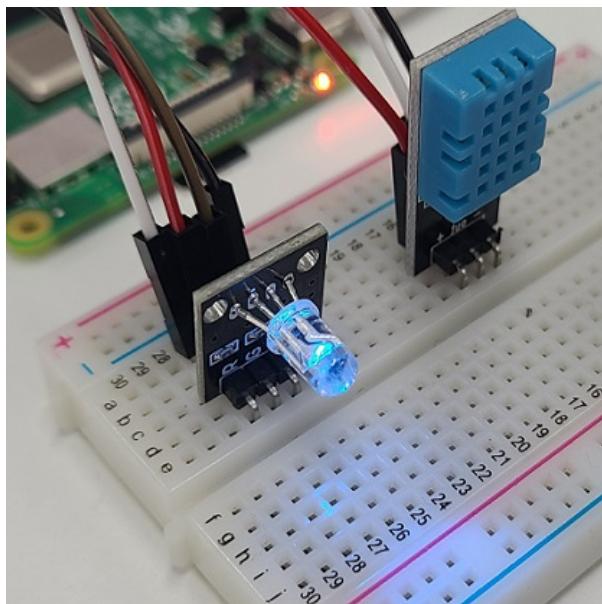


습도가 50 를 경우 분류되는 light type과 적측 humi 데이터 그리고 sub를 통한 con 값(=3)

```
led light type : 1
led light type : 1
led light type : 1
```

```
DHT sensing result [temp: 27°C, humidity: 95%]
DHT sensing result [temp: 28°C, humidity: 58%]
DHT sensing result [temp: 28°C, humidity: 58%]
```

```
{
  "op" : 5,
  "rqi" : "c1WB7IBL440",
  "to" : "mqtt://[REDACTED]:1883/SDevice_DEMO?ct=json",
  "fr" : "/Mobius2",
  "pc" : {
    "m2m:sgn" : {
      "sur" : "Mobius/Device_DEMO/led/sub1",
      "nev" : {
        "rep" : {
          "m2m:cin" : {
            "rn" : "4-20230303093056849",
            "ty" : 4,
            "pi" : "3-20230303092458614092",
            "ri" : "4-20230303093056849072",
            "ct" : "20230303T093050",
            "lt" : "20230303T093050",
            "st" : 4,
            "et" : "20250303T093050",
            "cs" : 1,
            "con" : "1",
            "cr" : "55-ehqpdwFL"
          }
        },
        "net" : 3
      },
      "rvi" : "2a"
    }
  }
}
```



두 개의 Device TAS와 nCube-Thyme을 이용하여 다음과 같이 oneM2M AE-CNT-cin의 갱신을 통하여 센서 데이터의 확보와 LED의 제어가 가능함을 알 수 있다.

## 7. 결론

---

본 가이드를 통해 실제 DHT-11 센서와 RGB LED 모듈을 각각의 TAS와 연결하였고, 제어 동작 및 데이터별로 정의한 Topic에 맞춰 MQTT 프로토콜을 통해 nCube-Thyme-Nodejs를 거친 뒤 Mobius(IN-CSE)에 데이터를 적측하거나 sub-notification 동작을 통해 Mobius로 부터의 RGB LED 모듈 제어 명령을 내려보았습니다. 이를 통해 nCube-Thyme-Nodejs와 TAS의 동작, 연결을 통해 oneM2M 기반 Mobius(IN-CSE)에서 AE 이하 리소스 트리 생성과 해당 리소스들을 이용하여 IoT 디바이스 간 연동이 가능함을 알 수 있습니다.