**201502766 이호찬 HW6 과제**

<Conf.js>

…

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

// build cse

//KETI에서 제공하는 Mobius server Platform 주소

cse.host        = '203.253.128.161';

cse.port        = '7579';

cse.name        = 'Mobius';

cse.id          = '/Mobius2';

cse.mqttport    = '1883';

cse.wsport      = '7577';

// build ae => nCube에 대한 구성 내용이다.

ae.name         = 'iot-lab-holee';

ae.id           = 'S' + ae.name;

ae.parent       = '/' + cse.name;

ae.appid        = 'measure\_co2';

ae.port         = '9727';

ae.bodytype     = 'json'; // select 'json' or 'xml' or 'cbor'

ae.tasport      = '3105';

// build cnt => Container를 생성한다. 컨테이너 종류로는 co2, led, temp, tvoc가있다.

var count = 0;

cnt\_arr[count] = {};

cnt\_arr[count].parent = '/' + cse.name + '/' + ae.name;

cnt\_arr[count++].name = 'co2';

cnt\_arr[count] = {};

cnt\_arr[count].parent = '/' + cse.name + '/' + ae.name;

cnt\_arr[count++].name = 'led';

cnt\_arr[count] = {};

cnt\_arr[count].parent = '/' + cse.name + '/' + ae.name;

cnt\_arr[count++].name = 'temp';

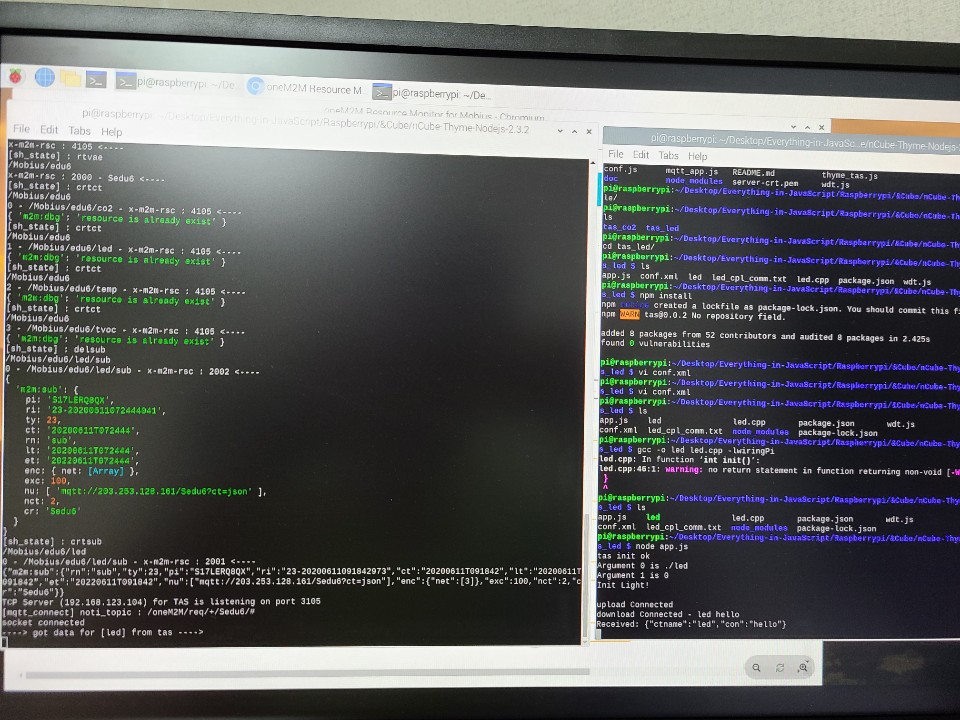
cnt\_arr[count] = {};

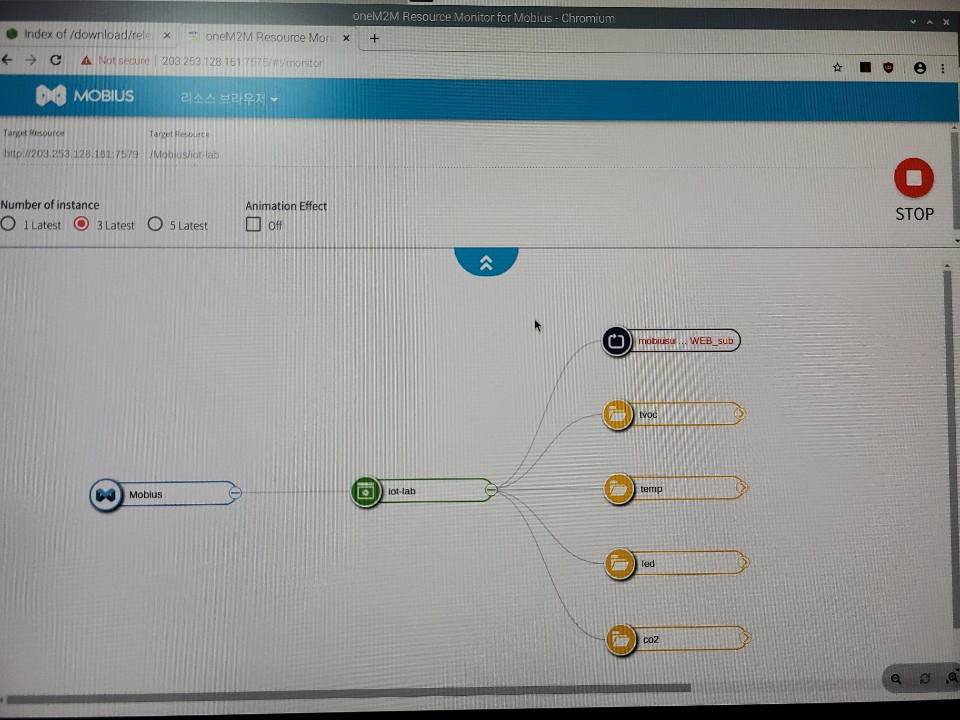
cnt\_arr[count].parent = '/' + cse.name + '/' + ae.name;

cnt\_arr[count++].name = 'tvoc';

…

다음장은 thyme.js 실행화면이며 처음에는 ae.name을 iot-lab으로 진행했지만 이후 실습에는 iot-lab-holee로 ae.name을 바꿔서 진행했다.





<Conf.xml>

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<m2m:conf xmlns:m2m="http://www.onem2m.org/xml/protocols" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <tas>

        <parenthostname>localhost</parenthostname>

        <parentport>3105</parentport>

    </tas>

    <download>

        <ctname>led</ctname>

        <id>led#1</id>

    </download>

</m2m:conf>

<parenthostname>localhost</parenthostname>

<parentport>3105</parentport>

Tas가 연결할 parenthost의 AE의 주소를 나타낸다.

<download>

    <ctname>led</ctname>

    <id>led#1</id>

</download>

Upload, Download option이있다.

<ctname>led</ctname>

상위 태그에서 정한 방식을 진행할 container를 정한다. 우리는 led를 사용하므로 led로 바꿔준다.

<app.js>

/\*\*

 \* Created by ryeubi on 2015-08-31.

 \* Updated 2017.03.06

 \* Made compatible with Thyme v1.7.2

 \*/

var net = require('net');

var util = require('util');

var fs = require('fs');

var xml2js = require('xml2js');

var exec = require("child\_process").exec;

var wdt = require('./wdt');

var useparentport = '';

var useparenthostname = '';

var upload\_arr = [];

var download\_arr = [];

var conf = {};

// This is an async file read

// 아까 위에서 설정한 conf 파일을 읽어서 처리한다.

fs.readFile('conf.xml', 'utf-8', function (err, data) {

    if (err) {

        console.log("FATAL An error occurred trying to read in the file: " + err);

        console.log("error : set to default for configuration")

    }

    else {

// xml 파싱을 진행한다.

        var parser = new xml2js.Parser({explicitArray: false});

        parser.parseString(data, function (err, result) {

            if (err) {

                console.log("Parsing An error occurred trying to read in the file: " + err);

                console.log("error : set to default for configuration")

            }

            else {

                var jsonString = JSON.stringify(result);

                conf = JSON.parse(jsonString)['m2m:conf'];

// 파싱이후 xml파일에서 parenthostname과 port를 가져오는 모습

                useparenthostname = conf.tas.parenthostname;

                useparentport = conf.tas.parentport;

// 만약 xml 파일 중 정한 옵션이 upload일 경우

                if(conf.upload != null) {

                    if (conf.upload['ctname'] != null) {

                        upload\_arr[0] = conf.upload;

                    }

                    else {

                        upload\_arr = conf.upload;

                    }

                }

// 만약 xml 파일 중 정한 옵션이 download일 경우

                if(conf.download != null) {

// xml 파일에서 설정해준 container name을 가져온다.

                    if (conf.download['ctname'] != null) {

                        download\_arr[0] = conf.download;

                    }

                    else {

// conf.download 는 이때 id 값이다.

                        download\_arr = conf.download;

                    }

                }

            }

        });

    }

});

var tas\_state = 'init';

var upload\_client = null;

var t\_count = 0;

var tas\_download\_count = 0;

// AE로부터 input이 있을 때 실행된다.

function on\_receive(data) {

    if (tas\_state == 'connect' || tas\_state == 'reconnect' || tas\_state == 'upload') {

        var data\_arr = data.toString().split('<EOF>');

        if(data\_arr.length >= 2) {

            for (var i = 0; i < data\_arr.length - 1; i++) {

                var line = data\_arr[i];

                var sink\_str = util.format('%s', line.toString());

                var sink\_obj = JSON.parse(sink\_str);

                if (sink\_obj.ctname == null || sink\_obj.con == null) {

                    console.log('Received: data format mismatch');

                }

                else {

                    if (sink\_obj.con == 'hello') {

                        console.log('Received: ' + line);

                        if (++tas\_download\_count >= download\_arr.length) {

                            tas\_state = 'upload';

                        }

                    }

                    else {

                        for (var j = 0; j < upload\_arr.length; j++) {

                            if (upload\_arr[j].ctname == sink\_obj.ctname) {

                                console.log('ACK : ' + line + ' <----');

                                break;

                            }

                        }

// Input값 처리 부분인데 Input은 실질적으로 우리가 브라우져에서 create해서 넘겨주는 즉 모비우스에서 &Cube로 넘겨주는 integer 값이다.

                        for (j = 0; j < download\_arr.length; j++) {

                            if (download\_arr[j].ctname == sink\_obj.ctname) {

                                g\_down\_buf = JSON.stringify({id: download\_arr[i].id, con: sink\_obj.con});

                                console.log(g\_down\_buf + ' <----');

// 받은 값을 control\_led로 넘겨준다.

                                control\_led(sink\_obj.con);

                                break;

                            }

                        }

                    }

                }

            }

        }

    }

}

// 넘겨받은 인자들과 함께led.cpp 파일을 실행한다.

function control\_led(comm\_num){

    var cmd = 'sudo ./led ' + comm\_num;

    exec(cmd, function callback(error, stdout, stderr) {

        console.log(stdout);

    });

}

var Serial = null;

var myPort = null;

// 주기적으로 실행하며 Input을 받으면 처리한다.

function tas\_watchdog() {

    if(tas\_state == 'init') {

// Socker 생성

        upload\_client = new net.Socket();

// on\_receive함수 실행

        upload\_client.on('data', on\_receive);

        upload\_client.on('error', function(err) {

            console.log(err);

            tas\_state = 'reconnect';

        });

// socket이 close이면 destroy 후 tas\_state 을 reconnect로 변경

        upload\_client.on('close', function() {

            console.log('Connection closed');

            upload\_client.destroy();

            tas\_state = 'reconnect';

        });

// socket 통신 후 tas\_state를 init\_thing으로 변경

        if(upload\_client) {

            console.log('tas init ok');

            tas\_state = 'init\_thing';

        }

    }

    else if(tas\_state == 'init\_thing') {

// tas\_state가 init\_thing일 때 초기화 진행 후 tas\_state를 connect로 변경

        control\_led('0');

        tas\_state = 'connect';

    }

// tas\_state가 connect나 reconnect일 경우 AE로부터 데이터를 받음.

    else if(tas\_state == 'connect' || tas\_state == 'reconnect') {

        upload\_client.connect(useparentport, useparenthostname, function() {

            console.log('upload Connected');

            tas\_download\_count = 0;

            for (var i = 0; i < download\_arr.length; i++) {

                console.log('download Connected - ' + download\_arr[i].ctname + ' hello');

                var cin = {ctname: download\_arr[i].ctname, con: 'hello'};

                upload\_client.write(JSON.stringify(cin) + '<EOF>');

            }

            if (tas\_download\_count >= download\_arr.length) {

                tas\_state = 'upload';

            }

        });

    }

}

// watchdog를 주기적으로 실행해주기위한 코드.

wdt.set\_wdt(require('shortid').generate(), 3, tas\_watchdog);

var cur\_c = '';

var pre\_c = '';

var g\_sink\_buf = '';

var g\_sink\_ready = [];

var g\_sink\_buf\_start = 0;

var g\_sink\_buf\_index = 0;

var g\_down\_buf = '';

#include <wiringPi.h>

#include <stdio.h>

#include <string.h>

#define VCC\_PIN 0

#define GREEN\_PIN 1 // GPIO 18

#define BLUE\_PIN 4 // GPIO 23

// green led ON 함수

int turn\_on\_green(){

    printf("%s\n", "Green Light ON!");

    digitalWrite(VCC\_PIN, HIGH);

    digitalWrite(GREEN\_PIN, LOW);

    return 0;

}

// green led OFF 함수

int turn\_off\_green(){

    printf("%s\n", "Green Light OFF!");

    digitalWrite(VCC\_PIN, HIGH);

    digitalWrite(GREEN\_PIN, HIGH);

    return 0;

}

// blue led ON 함수

int turn\_on\_blue(){

    printf("%s\n", "Blue Light ON!");

    digitalWrite(VCC\_PIN, HIGH);

    digitalWrite(BLUE\_PIN, LOW);

    return 0;

}

// blue led OFF 함수

int turn\_off\_blue(){

    printf("%s\n", "Blue Light OFF!");

    digitalWrite(VCC\_PIN, HIGH);

    digitalWrite(BLUE\_PIN, HIGH);

    return 0;

}

// vcc, green led, blue led 모두 OFF하는 Init 함수

int init(){

    printf("%s\n", "Init Light!");

    digitalWrite(VCC\_PIN, HIGH);

    digitalWrite(GREEN\_PIN, HIGH);

    digitalWrite(BLUE\_PIN, HIGH);

}

int main (int argc,char \*argv[])

{

    int i;

    for (i=0; i < argc; i++)

        printf("Argument %d is %s\n", i, argv[i]);

    wiringPiSetup() ;

    pinMode(VCC\_PIN, OUTPUT) ;

    pinMode(GREEN\_PIN, OUTPUT) ;

    pinMode(BLUE\_PIN, OUTPUT) ;

// 받은 인자값으로 각각의 함수를 실행 시켜준다.

    if(argc == 2){

        char\* comm = argv[1];

        if(strcmp(comm, "1") == 0){

            turn\_on\_green();

        } else if(strcmp(comm, "2") == 0){

            turn\_off\_green();

        } else if(strcmp(comm, "3") == 0){

            turn\_on\_blue();

        } else if(strcmp(comm, "4") == 0){

            turn\_off\_blue();

        }else if(strcmp(comm, "0") == 0){

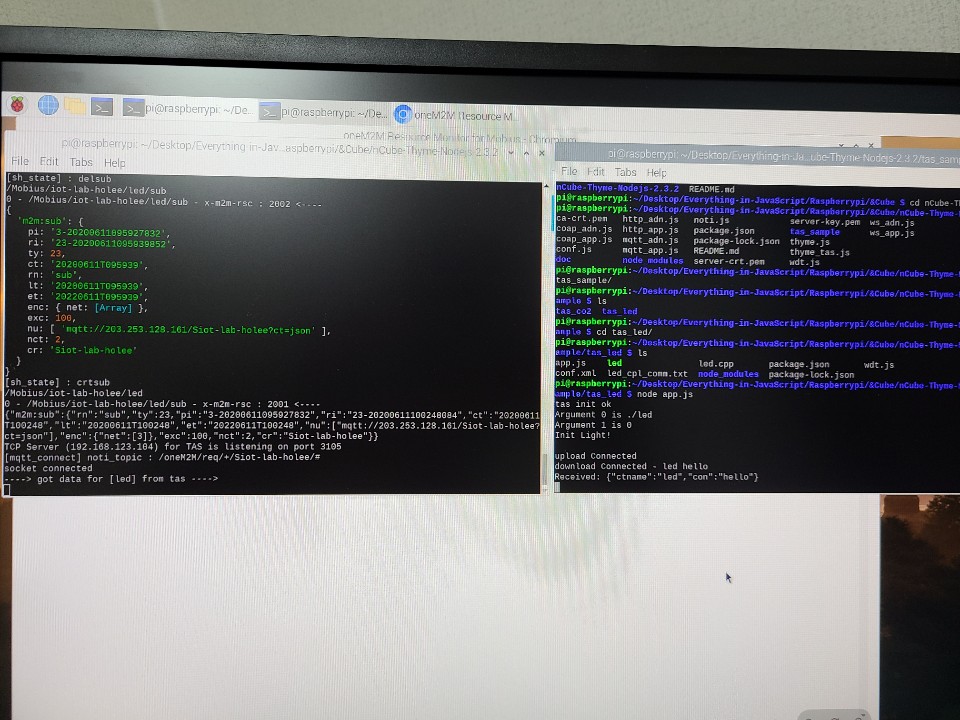
            init();

        }

    }

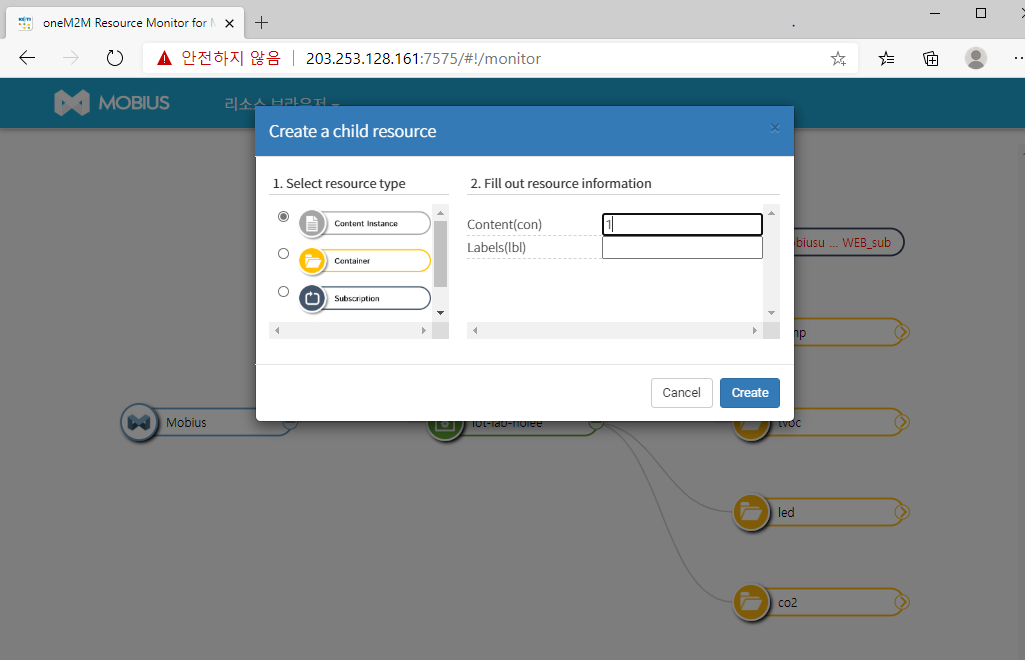
    return 0 ;

}



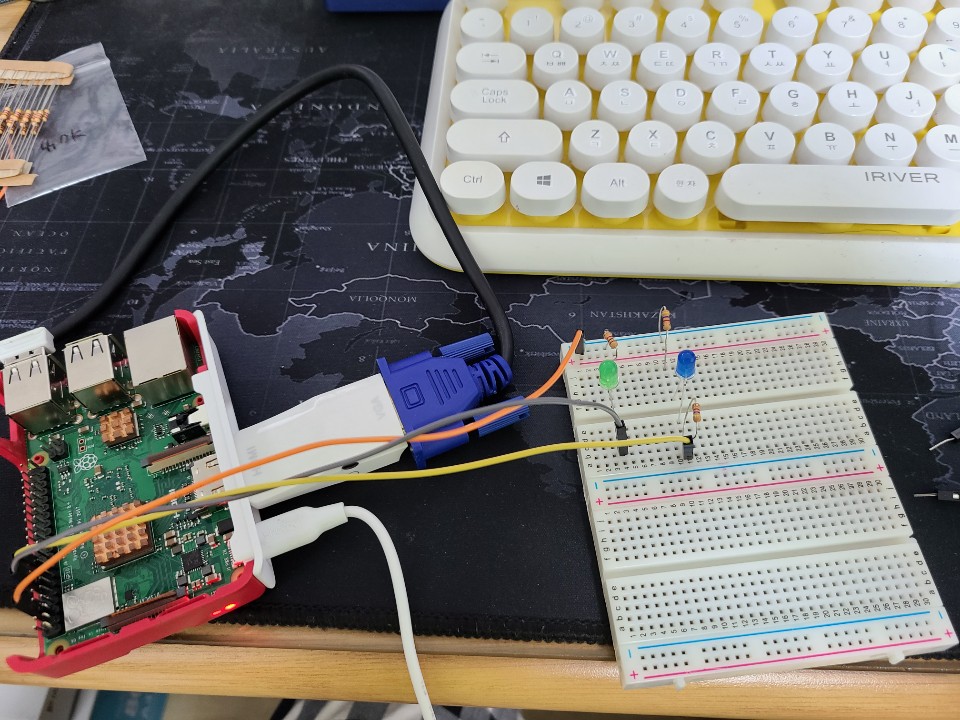
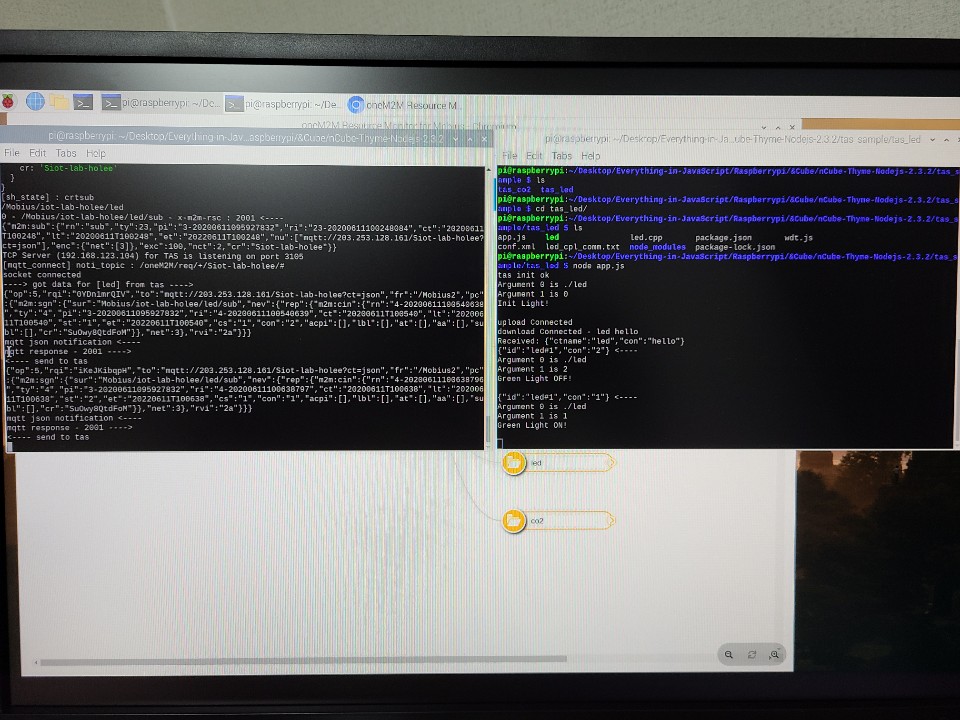
Thyme.js 실행후 app.js 를 실행시킨다.

App.js 실행후 Socket Connected가 뜨는걸로 봐서 연결이 잘 된 것을 확인 할 수 있다.

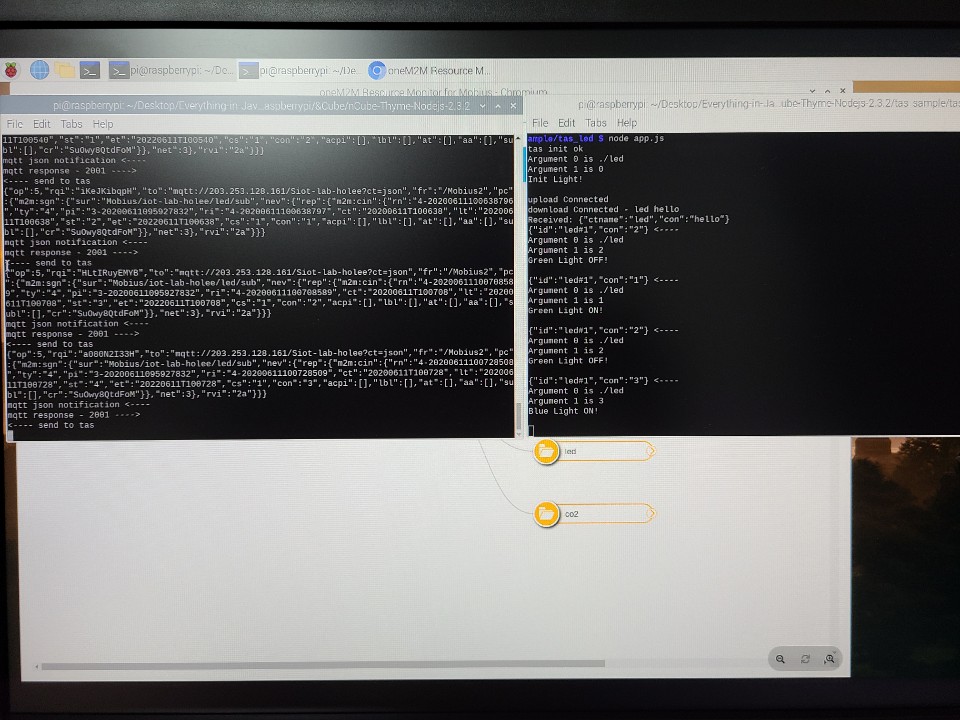


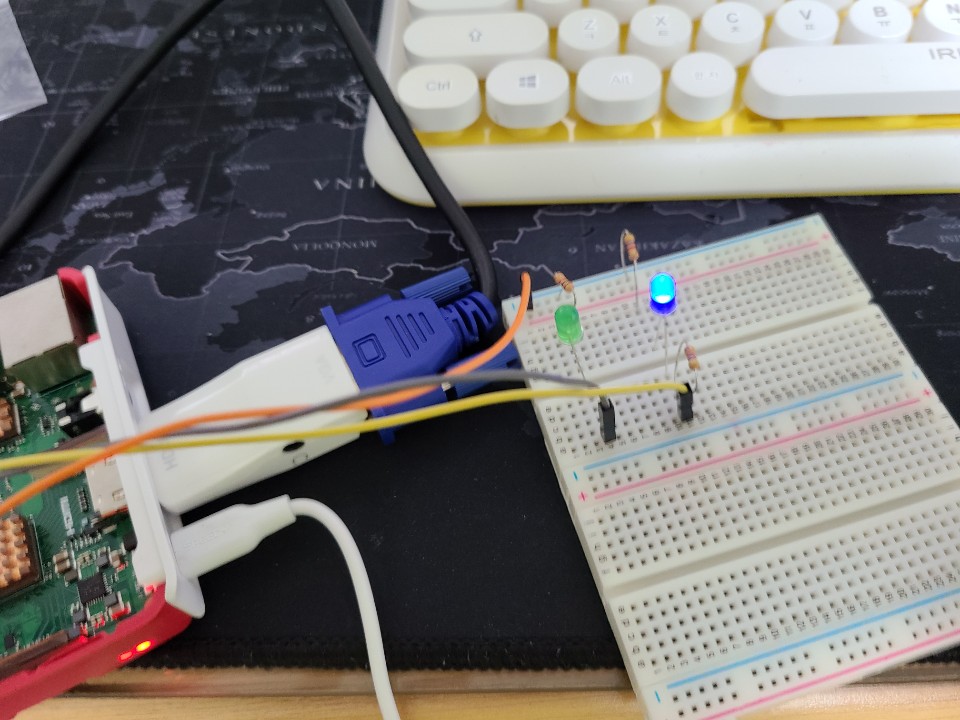
브라우져로 <http://203.253.128.161:7575/#!/monitor>로 제공됨으로 라즈베리파이가아닌 다른 컴퓨터로도 접속이 가능하다. 다음과 같이 content값을 수정하며 integer를 보내준다.

초록 불이 켜진다. 아래 사진은 초록불이 켜진 상태이다. 사진상으로는 잘 안나왔는데 자세히 보면 불이 켜져있다.



초록불을 꺼준다음에 파란불을 켜준다.





모든 불을 끈다.

