Arduino(WeMos D1) / MQTT / Node.js / MongoDB / Eclipse IDE

MQTT 프로토콜을 이용한 사물인터넷 통신 프로젝트

[프로젝트 세부주제]

→ WEB을 이용한 LED 제어

→ WEB을 이용한 DHT11 온도,습도 모니터링

박매일

1. 사전준비사항

준비물

- Arduino(WeMos D1)
- DHT11 Sensor
- 브래드보드
- LED 1개
- 저항 1개(220 Ohm)
- 점퍼케이블
- 전원 케이블(USB)

디렉토리 만들기

C:₩MQTTProject

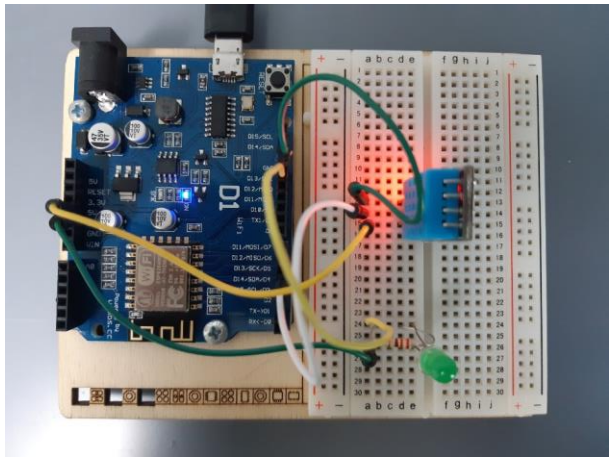
- **source**
- **sw**

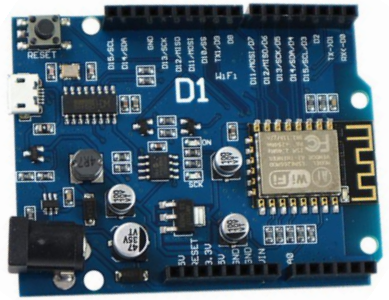
설치 프로그램

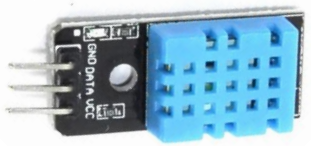
- **mosquitto**
- **Arduino IDE**
- **Node.js**
- **MongoDB**

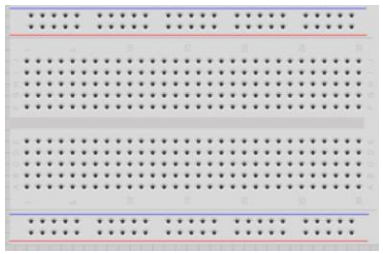
통신연결 주소

- WiFi AP 이름
- WiFi AP 비밀번호
- PC IP주소 (ipconfig)



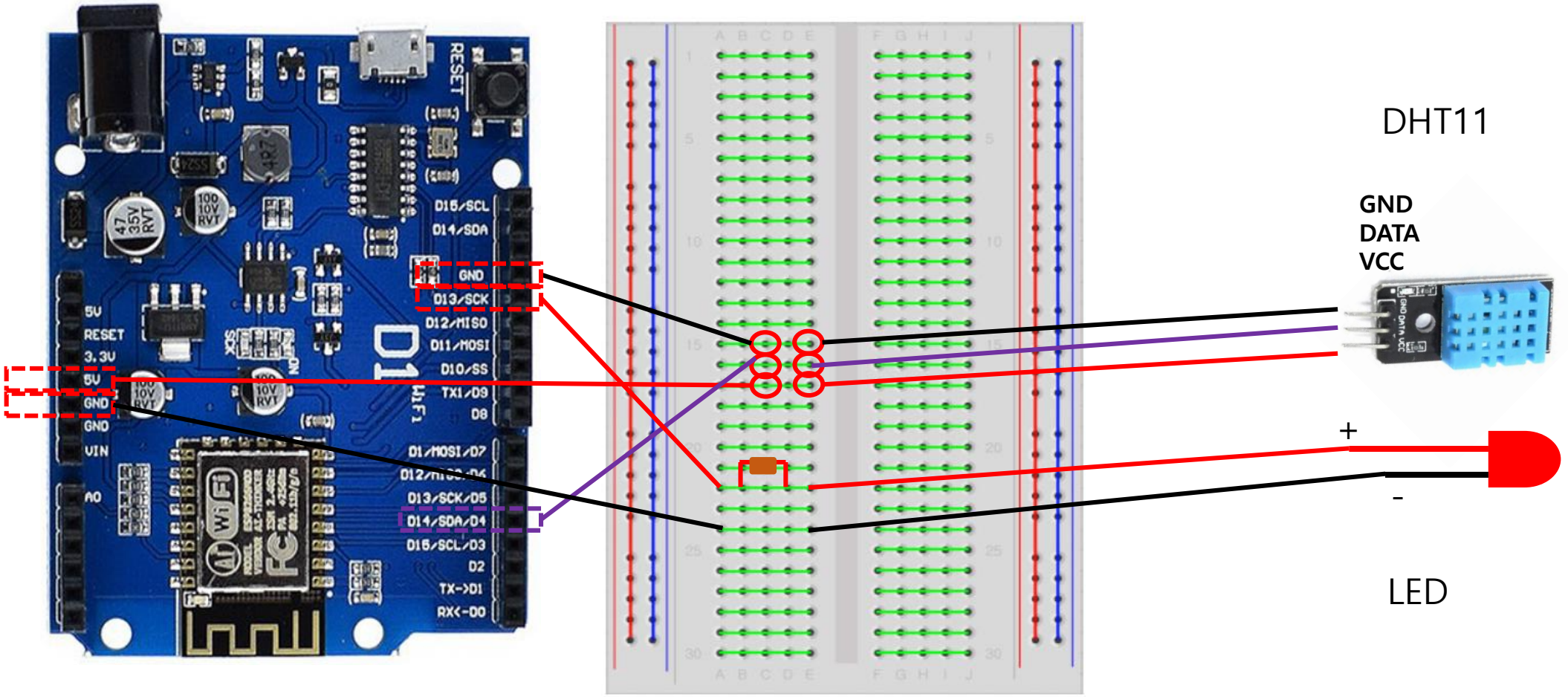






인터넷에서 조회 후 구매

2. 회로도 구성


RESET
D15/SCL
D14/SDA
GND
D13/SCK
D12/MISO
D11/MOSI
D10/SS
TX1/D9
D8
D1/MOSI/D7
D13/SCK/D5
D14/SDA/D4
D15/SCL/D3
D2
TX->D1
RX<-D0
5V
RESET
3.3V
5V
GND
GND
VIN
A0
DHT11
GND
DATA
VCC
+
-
LED

1. Eclipse Mosquitto™ An open source MQTT broker Download

→ https://mosquitto.org/

mosquitto

mosquitto ECLIPSE FOUNDATION

Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQ devices from low power single board computers to full servers.

The MQTT protocol provides a lightweight method of carrying out messaging using a publish/su sensors or mobile devices such as phones, embedded computers or microcontrollers.

The Mosquitto project also provides a C library for implementing MQTT clients, and the very pop

Mosquitto is part of the Eclipse Foundation, is an iot.eclipse.org project and is sponsored by ced

Download

Mosquitto is highly portable and available for a wide range of platforms. Go to the dedicated download page to find the source or binaries for your platform.

Test

You can have your own instance of Mosquitto running in minutes, but to make testing even easier, the Mosquitto Project runs a test server at test.mosquitto.org where you can test your clients in a variety of

mosquitto ECLIPSE FOUNDATION

Source

- mosquitto-1.6.6.tar.gz (319kB) (GPG signature)
- Git source code repository (github.com)

Older downloads are available at https://mosquitto.org/files/

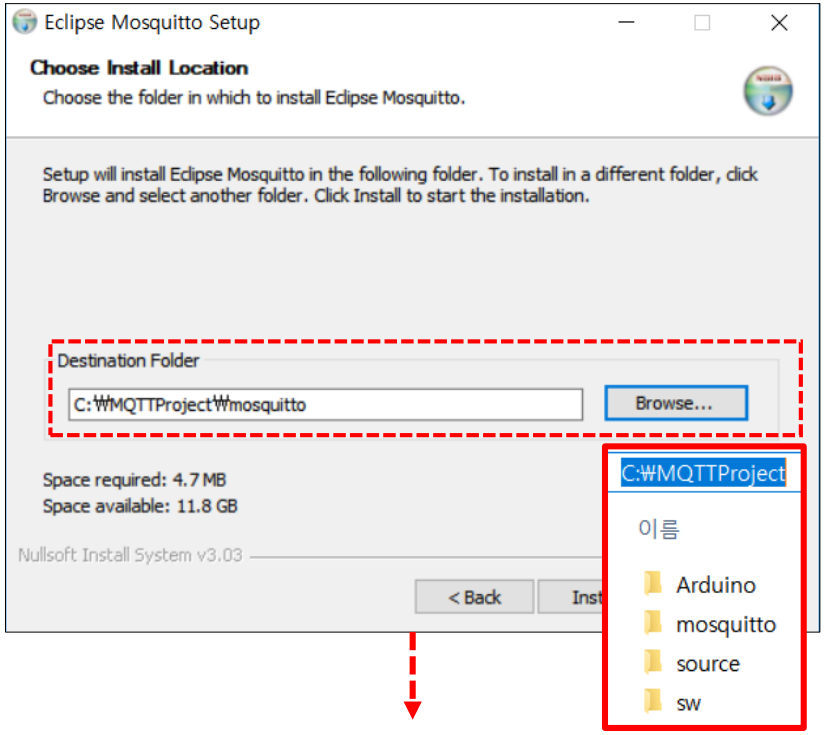
Binary Installation

The binary packages listed below are supported by the Mosquitto project. In ma

Windows

- mosquitto-1.6.6-install-windows-x64.exe (~1.4 MB) (64-bit build, Windows
- mosquitto-1.6.6-install-windows-x32.exe (~1.4 MB) (32-bit build, Windows

2. Mosquitto™ MQTT broker 설치 및 서버구동



① cmd(관리자 권한으로 실행)에서 서버 구동 →서버 구동 창

```
C:₩MQTTProject₩mosquitto>mosquitto -v
1569209814: mosquitto version 1.6.6 starting
1569209814: Using default config.
1569209814: Opening ipv6 listen socket on port 1883.
1569209814: Opening ipv4 listen socket on port 1883.
```

mosquito server 구동 및 구독자, 발행자 실행

② subscriber(구독자) 실행→수신대기 창

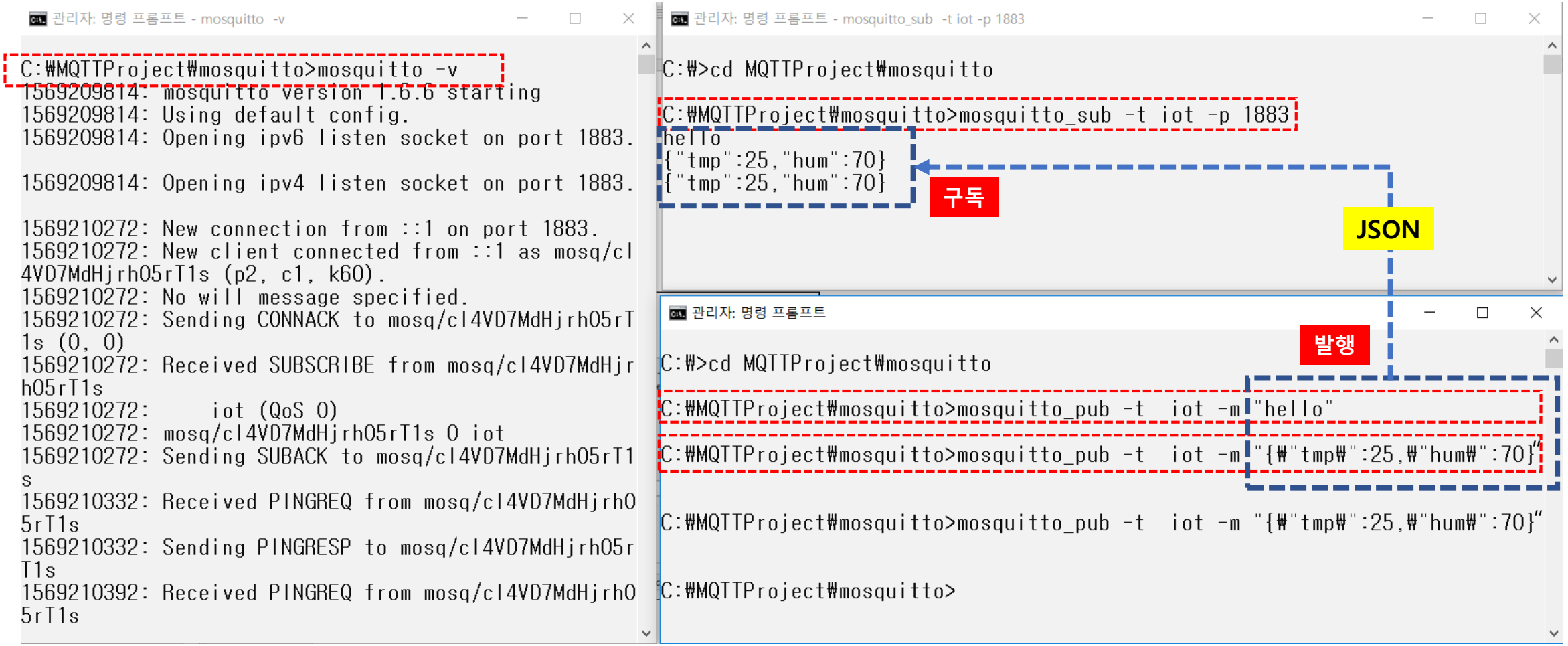
```
C:₩MQTTProject₩mosquitto>mosquitto_sub –t iot –p 1883
hello
{"tmp":25,"hum":70}
```

(외부에서 연결하는 방법)
C:₩MQTTProject₩mosquitto>**mosquitto_sub –h MQTT서버ip주소 –t iot –p 1883**

③ publisher(발행자) 실행→메시지(토픽) 발행 창

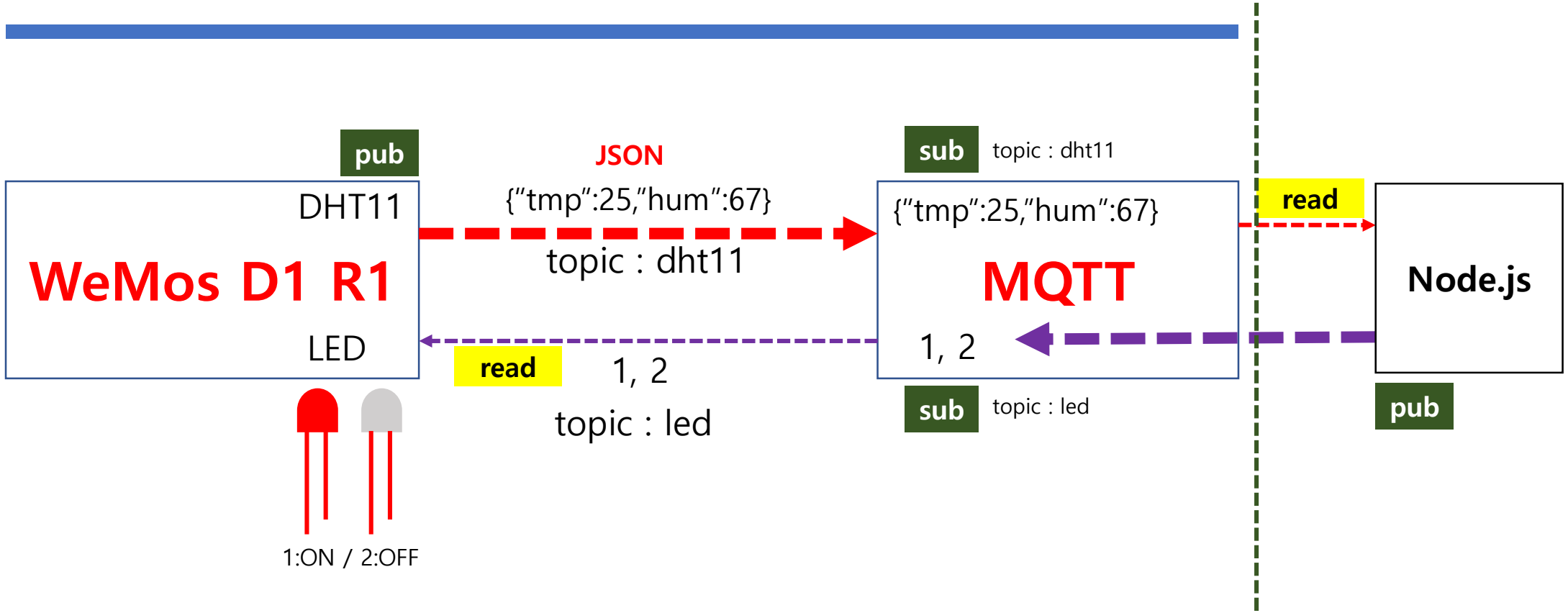
```
C:₩MQTTProject₩mosquitto>mosquitto_pub –t  iot –m “hello”
C:₩MQTTProject₩mosquitto>mosquitto_pub -t  iot -m "{₩"tmp₩":25,₩"hum₩":70}
```

3. MQTT broker 메시지 중개 테스트



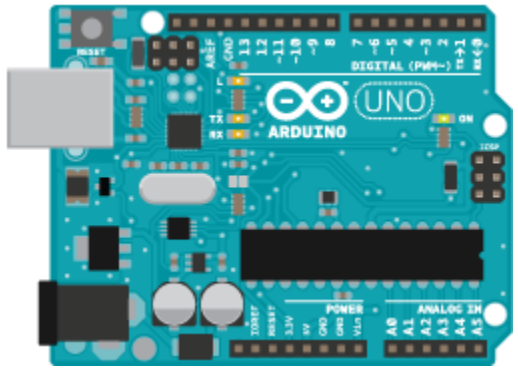
Arduino(WeMos)쪽에서 WiFi로 DHT11 Sensor 데이터(온도/습도) 발행

1. 구현내용(WeMos에서 DHT11 Sensor 데이터를 MQTT 서버로 발행하는 부분)

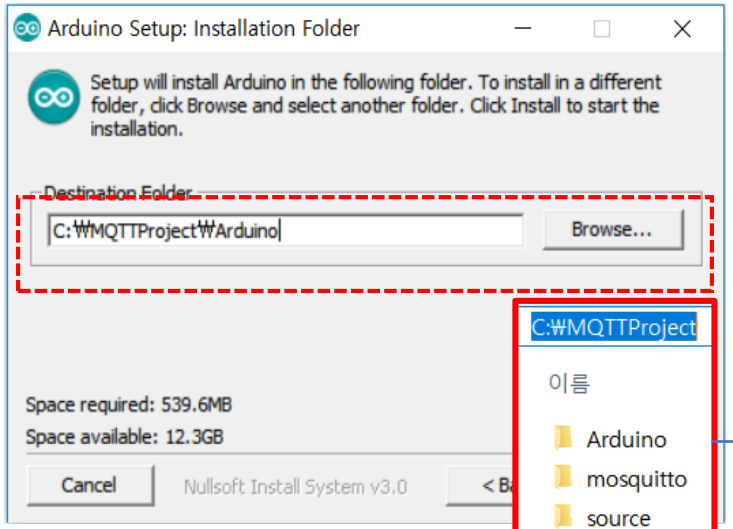


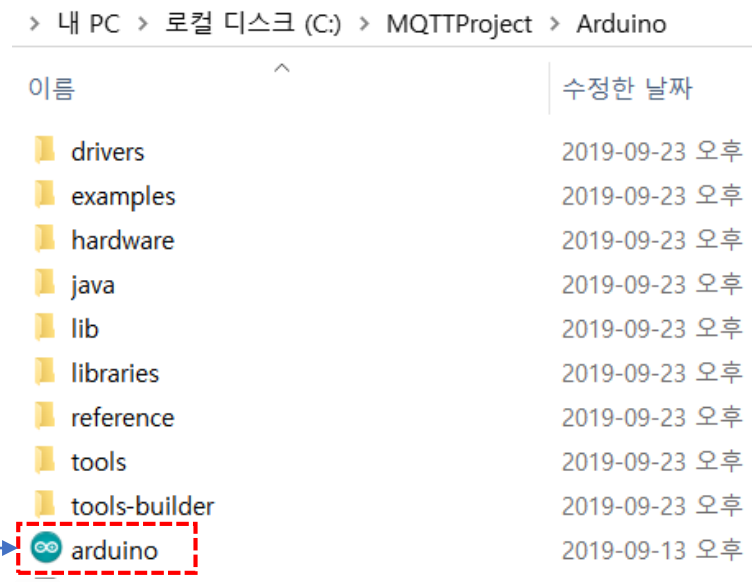
2. Download the Arduino IDE

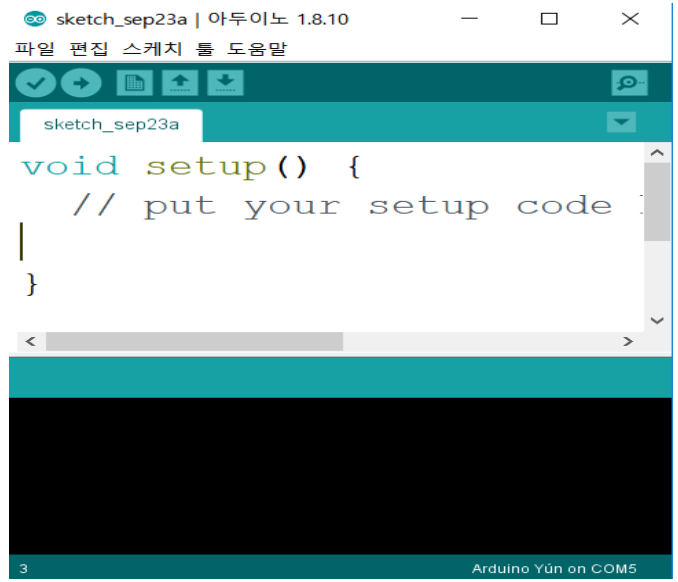
→ http://arduino.cc



3. Arduino 설치

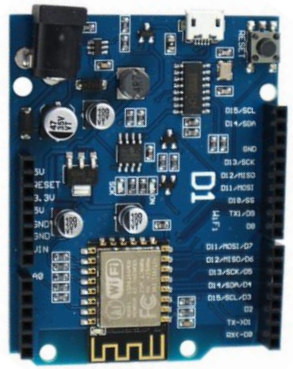




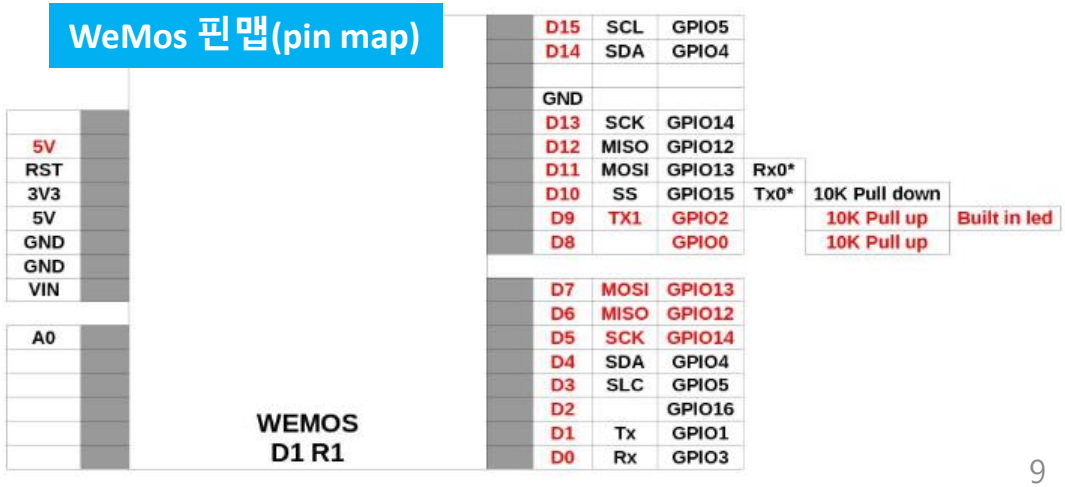


WeMos 보드의 장점

- 아두이노 UNO와 비슷
- 펌웨어 개발 없이 wifi를 다룰 수 있음
- 많은 사람들이 이용하는 ESP8266 모듈
- 가격이 저렴한 편



WeMos 핀맵(pin map)

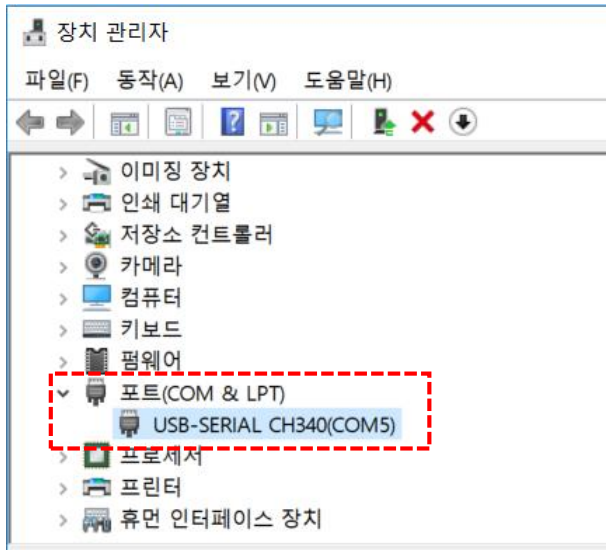


5. 아두이노 WeMos 환경 셋팅

환경설정 참고 SITE
http://www.makewith.co/page/project/1004/story/2478/

STEP 1. WeMos를 컴퓨터에 연결하기
1. WeMos-D1 R1 보드를 usb케이블로 컴퓨터와 연결합니다.
2. Windows OS가 반응합니다. 새 하드웨어의 부착을 알리고 USB 드라이버를 설치합니다.
3. 시스템에서 드라이버를 찾지 못할 경우 USB 드라이버를 다운로드 해야합니다.
   USB 인터페이스 칩은 CH340G입니다.

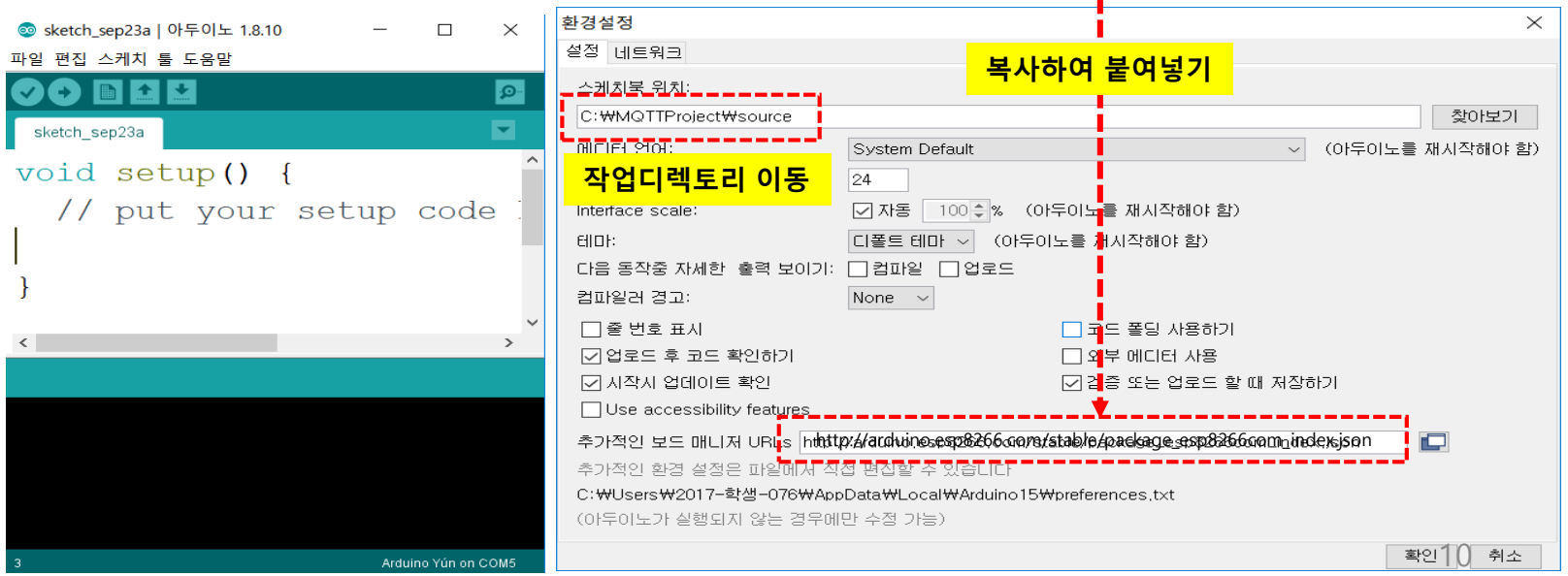
STEP 2. 연결 확인하기
장치관리자(Device Manager)에서 확인합니다.



STEP 3. 아두이노 IDE에 WeMos 보드 라이브러리 설치하기
→아래 URL을 클릭하지 말고 복사합니다.

http://arduino.esp8266.com/stable/package_esp8266com_index.json

① 파일->환경설정

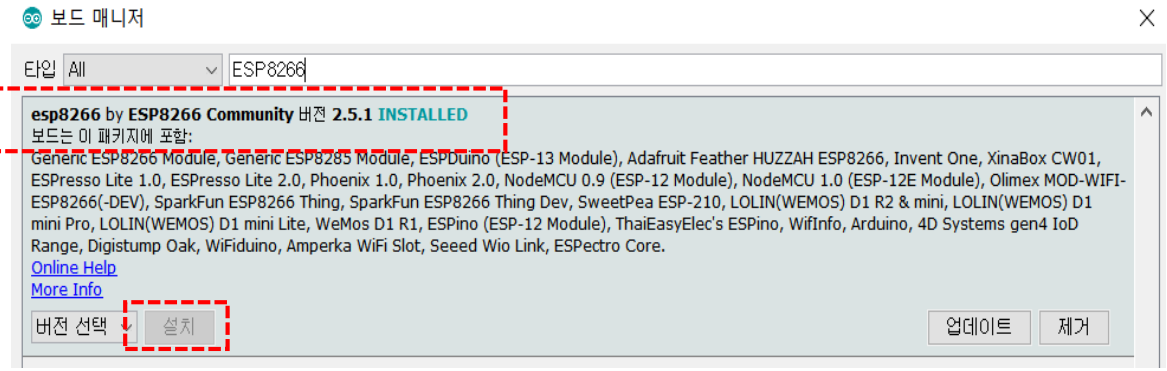


STEP 4. Board Manager로 WeMos보드 설치하기

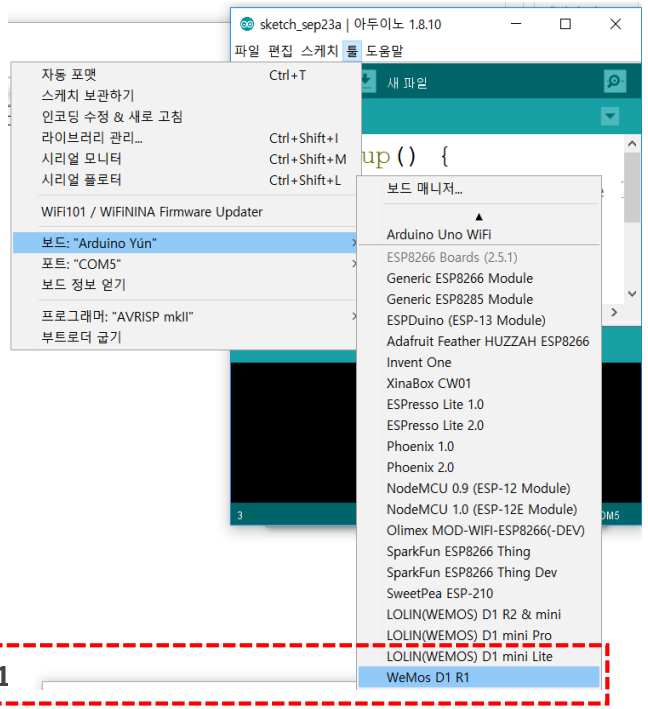
② 툴-> 보드 -> 보드 매니저...

→"esp8266 of ESP8266 Community"를 검색해서 설치해 주세요.

→ 설치 후 아두이노 IDE를 껐다 켜주세요.

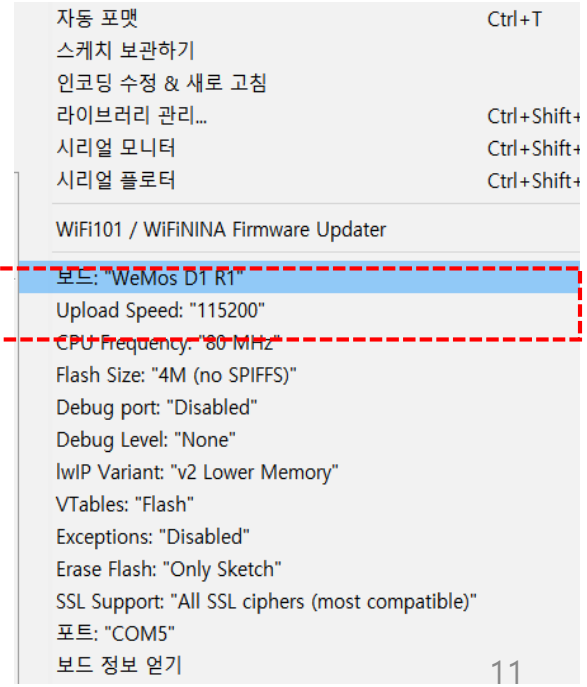


STEP 5. WeMos 보드 선택하기 & speed 설정하기



③ 툴-> 보드 -> WeMos D1

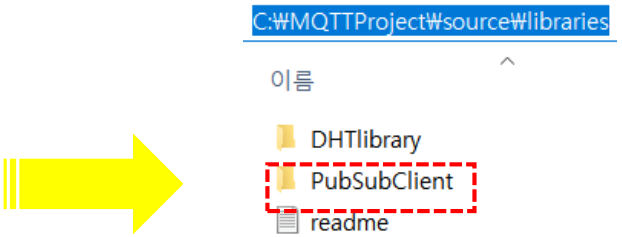
④ 툴-> 보드 -> Upload Speed “115200” 선택

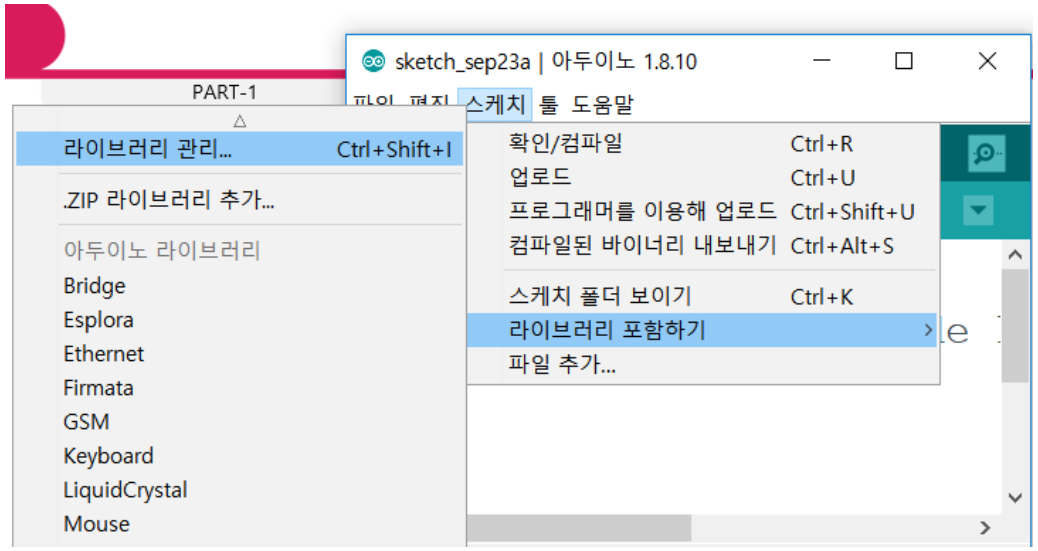


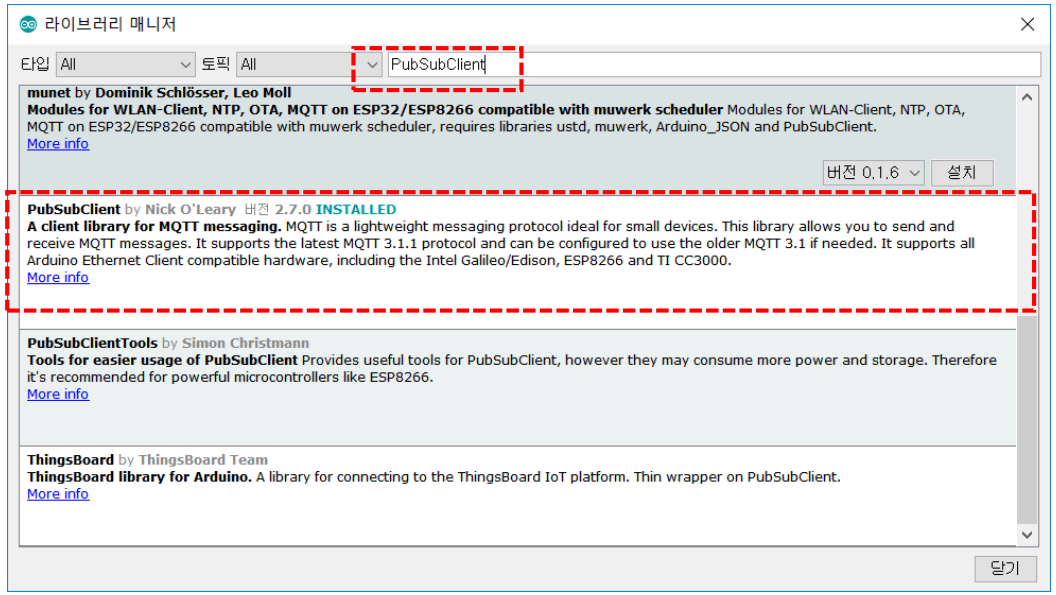
7. PubSubClient 라이브러리 설치

ESP8266 모듈에 MQTT 기능을 구현하기 위해서는 해당 라이브러리가 필요합니다. 아두이노 개발환경에서 사용할 수 있도록 개발된 라이브러리가 여러 종류가 있는데 여기서는 **PubSubClient** 라이브러리를 사용하도록 하겠습니다. 일단 아두이노 개발환경을 실행하고 아래 순서대로 라이브러리를 설치하면 됩니다.

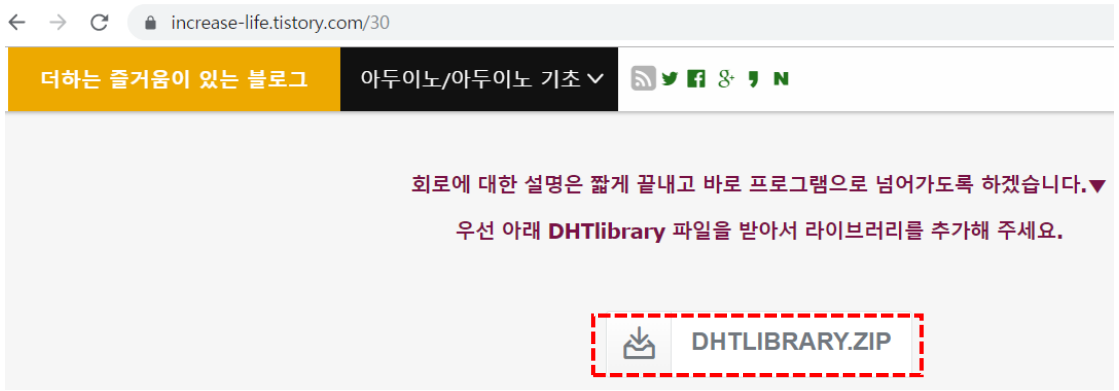
① 스케치 → 라이브러리 포함하기 --> 라이브러리 관리
② PubSubClient 검색
③ 클릭 후 최신 버전 설치



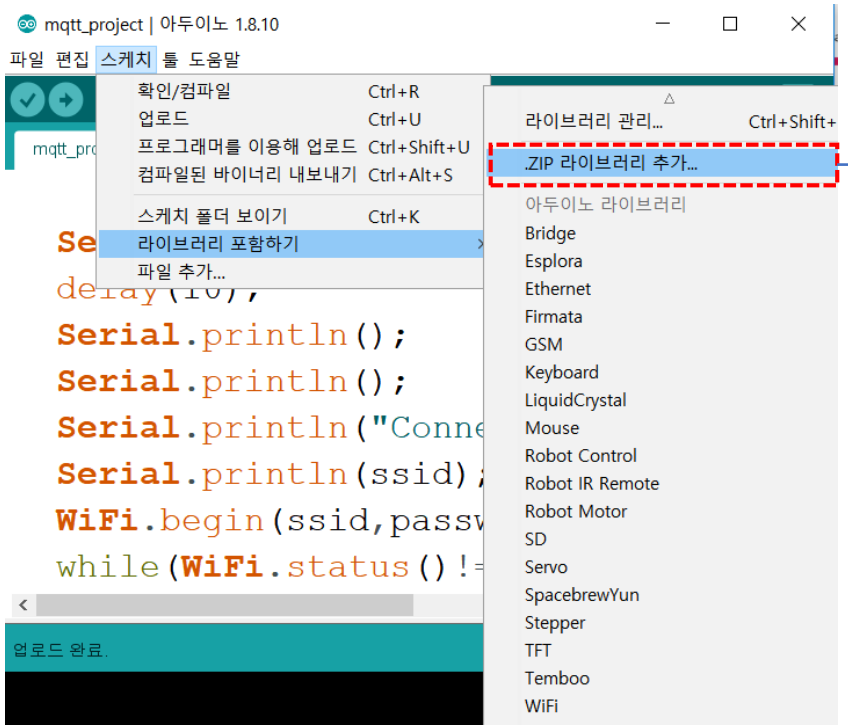


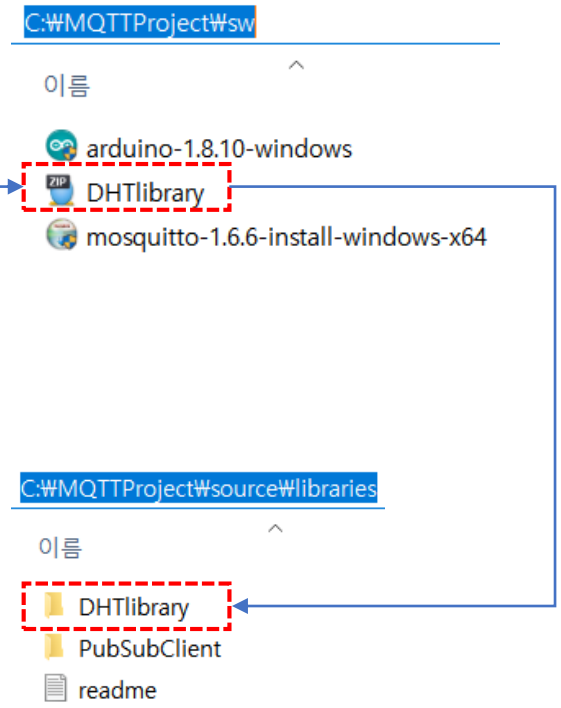


8. DHT11 라이브러리 설치



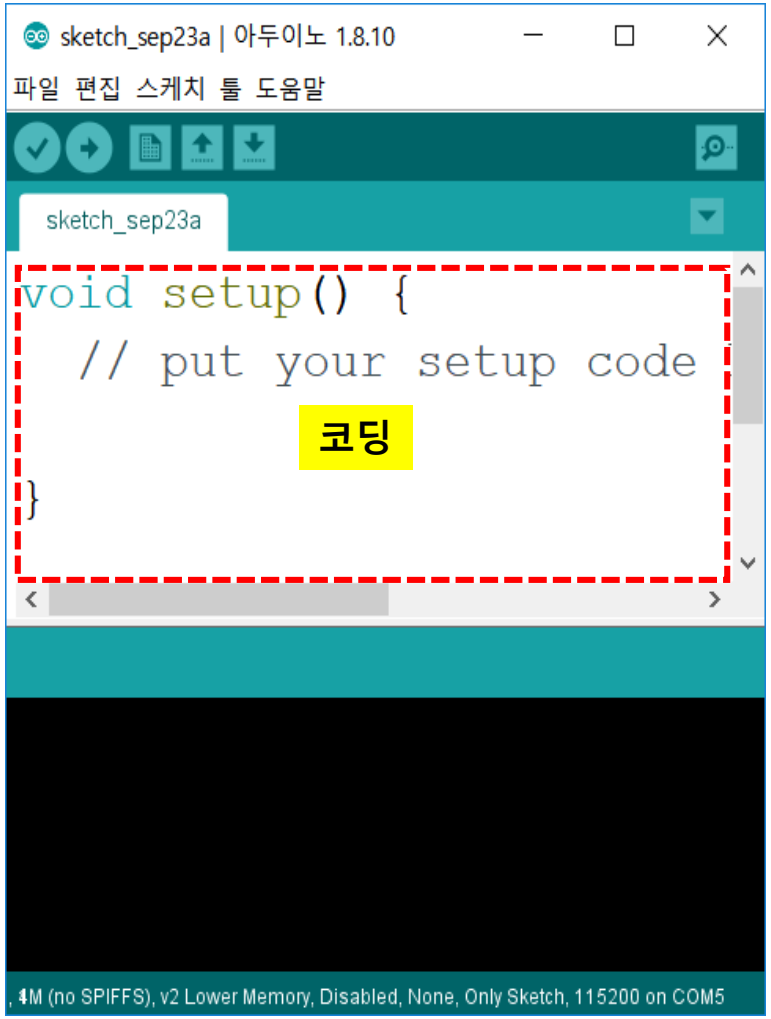
다운로드 SITE

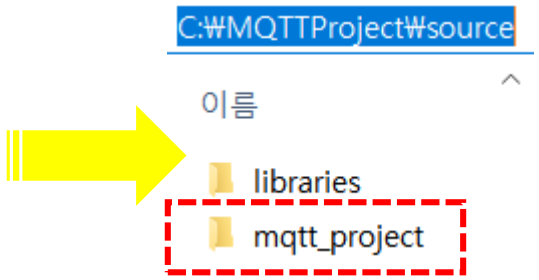


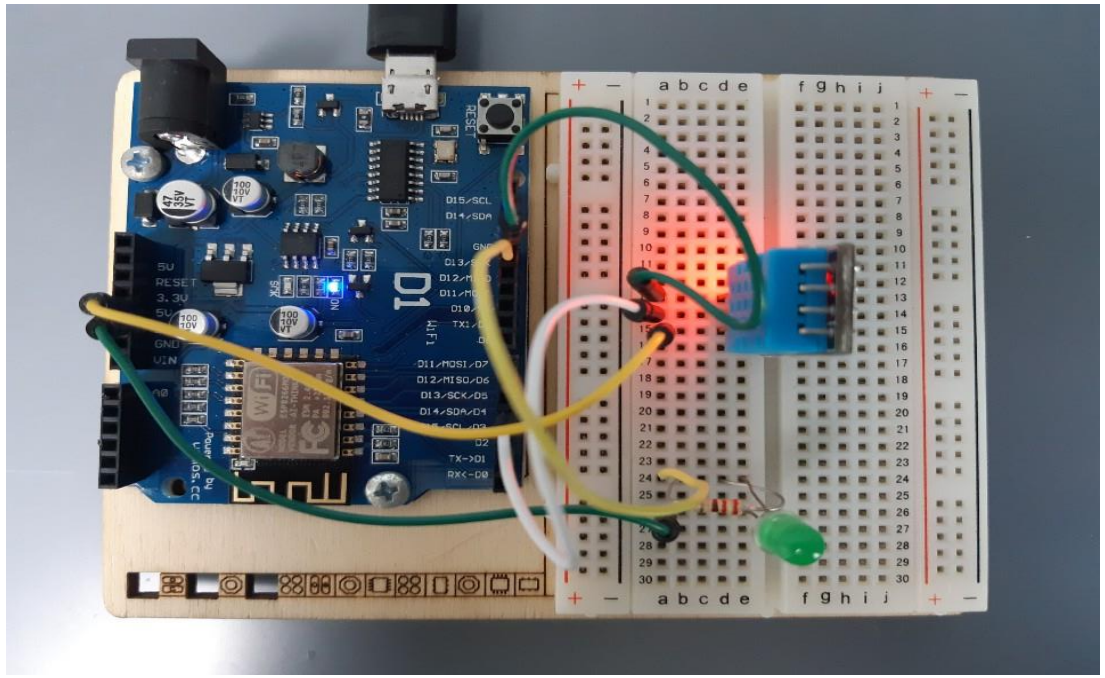


.zip 라이브러리 추가

9. 소스코딩(소스저장)


sketch_sep23a | 아두이노 1.8.10
파일 편집 스케치 툴 도움말
sketch_sep23a
void setup() {
// put your setup code
코딩
}
, 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM5


C:₩MQTTProject₩source
이름
libraries
mqtt_project



10. 소스코드(mqtt_project.ino)

```
char ssid[]="WiFi AP이름";
char password[]="비밀번호";
byte server1[] = {ip주소};//MQTT 브로커IP
```

```
#include <ESP8266WiFi.h>
#include "PubSubClient.h"
#include <DHT11.h>
char ssid[]="";
char password[]="";
byte server1[] = {};//MQTT 브로커IP
int port=1883;
DHT11 dht11(4);
WiFiClient client;
void msgReceived(char *topic,byte *payload,unsigned int uLen){
  char pBuffer[uLen+1];//데이터가 문자열로 날라오는데 데이터를 담을 수 있는 배열선언
                       //c언어 : "1"+₩0(문자열 끝) 문자열은 배열에 저장되어야 한다
  int i;
  for(i=0; i<uLen; i++){
    pBuffer[i]=payload[i];
  }
  pBuffer[i]='₩0';//끝을 의미
  Serial.println(pBuffer);
  if(pBuffer[0]=='1'){
    digitalWrite(14,HIGH);
  }else if(pBuffer[0]=='2'){
    digitalWrite(14,LOW);
  }
}
PubSubClient mqttClient(server1,port,msgReceived,client);
```

관리자: 명령 프롬프트 - mosquitto -v

```
C:₩MQTTProject₩mosquitto>mosquitto -v
1569420036: mosquitto version 1.6.6 starting
1569420036: Using default config.
1569420036: Opening ipv6 listen socket on port 1883.
1569420036: Opening ipv4 listen socket on port 1883.
1569420036: New connection from ::1 on port 1883.
```

```
void setup() {
  pinMode(14,OUTPUT);
  Serial.begin(115200);
  delay(10);
  Serial.println();
  Serial.println();
  Serial.println("Connecting to~");
  Serial.println(ssid);
  WiFi.begin(ssid,password);
  while(WiFi.status()!=WL_CONNECTED){
    delay(500);
    Serial.print(",");
  }
  Serial.println("");
  Serial.println("Wi-Fi AP Connected!");

  Serial.println(WiFi.localIP());
  if(mqttClient.connect("Arduino")){ //MQTT브로커에 접속을 시도하는 것
    Serial.println("MQTT Broker Connected!");
    mqttClient.subscribe("led");
  }
}
```

선택 관리자: 명령 프롬프트 - mosquitto_sub -t led -p 1883

```
C:₩Windows>cd..

C:₩>cd MQTTProject₩mosquitto

C:₩MQTTProject₩mosquitto>mosquitto_sub -t led -p 1883
2
1
1
1
1
2
1
2
1
2
```

```
void loop() {
  mqttClient.loop();
  float tmp, hum;
  int err = dht11.read(hum,tmp);
  if(err==0){
    char message[64] = "", pTmpBuf[50], pHumBuf[50];
    dtostrf(tmp,5,2,pTmpBuf);
    dtostrf(hum,5,2,pHumBuf);
    sprintf(message,"{₩"tmp₩":%s,₩"hum₩":%s}",pTmpBuf,pHumBuf);
    mqttClient.publish("dht11",message);
  }
  delay(3000);
}
```

관리자: 명령 프롬프트 - mosquitto_sub -t dht11 -p 1883

```
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":73.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
```

11. TEST 서버 구동(MQTT Server)

관리자: 명령 프롬프트 - mosquitto -v

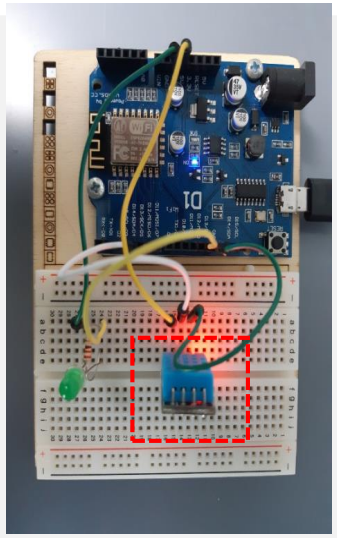
```
1569239495: Received PUBLISH from Arduino (d0, q0, r0,
 m0, 'dht11', ... (25 bytes))
1569239495: Sending PUBLISH to mosq/acTOza1WP16Ind5fc3
 (d0, q0, r0, m0, 'dht11', ... (25 bytes))
1569239498: Received PUBLISH from Arduino (d0, q0, r0,
 m0, 'dht11', ... (25 bytes))
1569239498: Sending PUBLISH to mosq/acTOza1WP16Ind5fc3
 (d0, q0, r0, m0, 'dht11', ... (25 bytes))
1569239501: Received PINGREQ from Arduino
1569239501: Sending PINGRESP to Arduino
1569239501: Received PUBLISH from Arduino (d0, q0, r0,
 m0, 'dht11', ... (25 bytes))
1569239501: Sending PUBLISH to mosq/acTOza1WP16Ind5fc3
 (d0, q0, r0, m0, 'dht11', ... (25 bytes))
```

관리자: 명령 프롬프트 - mosquitto_sub -t dht11 -p 1883

```
{"tmp":24.00,"hum":78.00}
{"tmp":24.00,"hum":76.00}
{"tmp":24.00,"hum":76.00}
{"tmp":24.00,"hum":76.00}
{"tmp":24.00,"hum":76.00}
{"tmp":24.00,"hum":75.00}
{"tmp":24.00,"hum":76.00}
{"tmp":24.00,"hum":77.00}
{"tmp":24.00,"hum":77.00}
{"tmp":24.00,"hum":77.00}
{"tmp":30.00,"hum":77.00}
{"tmp":24.00,"hum":76.00}
{"tmp":23.00,"hum":77.00}
```

WiFi

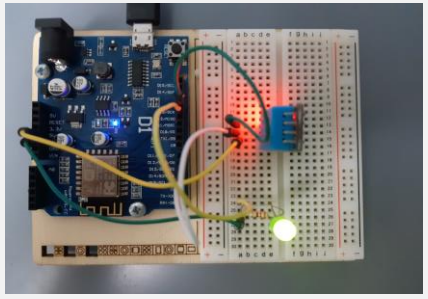
JSON



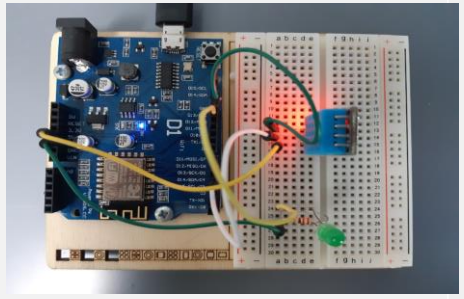
관리자: 명령 프롬프트 - mosquitto_sub -t led -p 1883

```
C:₩MQTTProject₩mosquitto>mosquitto_sub -t led -p 1883
1
2
```

1->ON



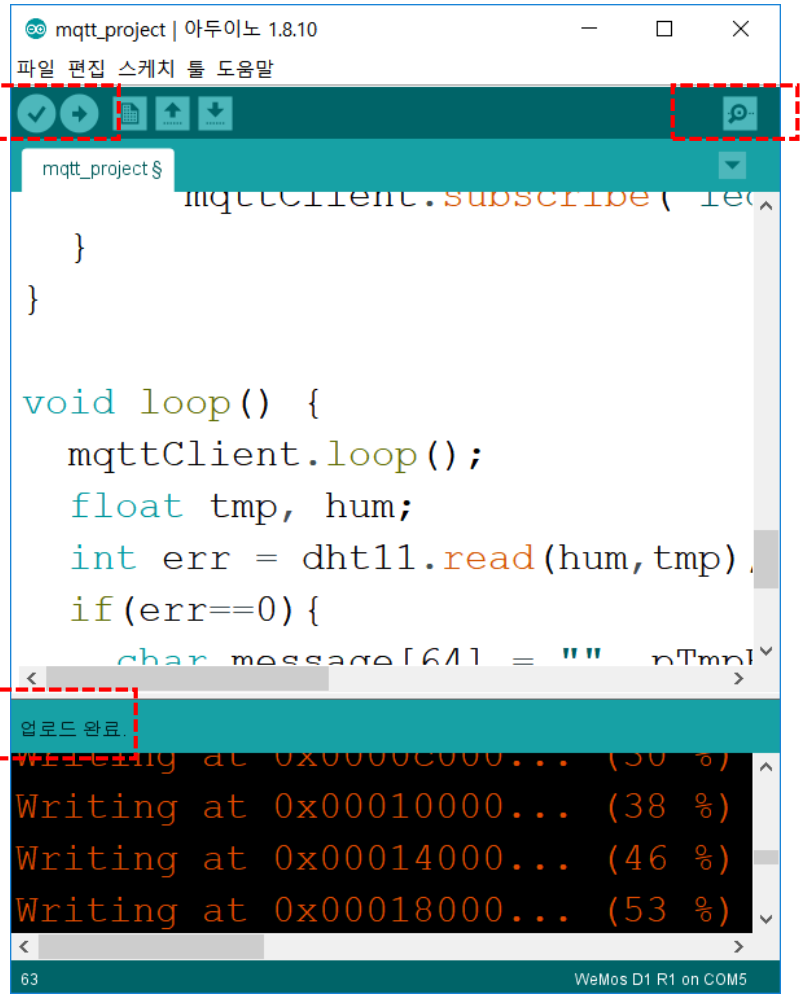
2->OFF

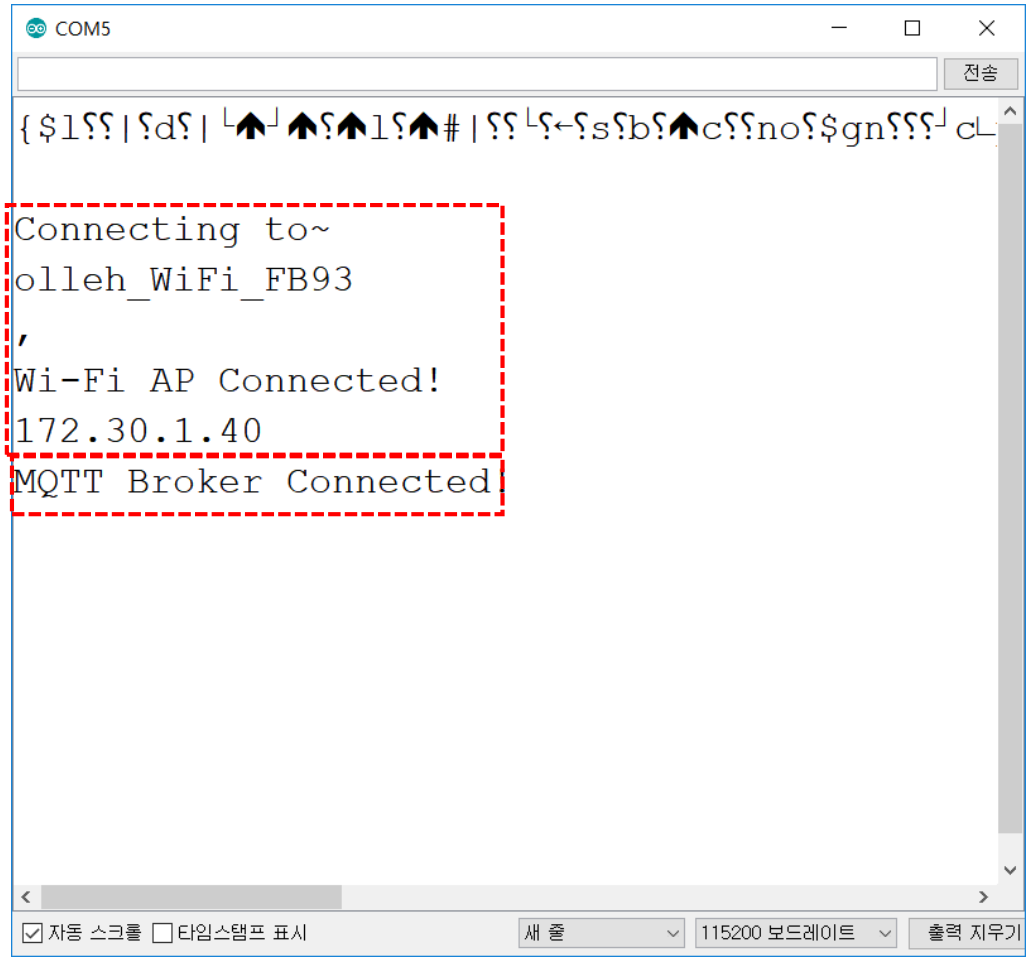


관리자: 명령 프롬프트

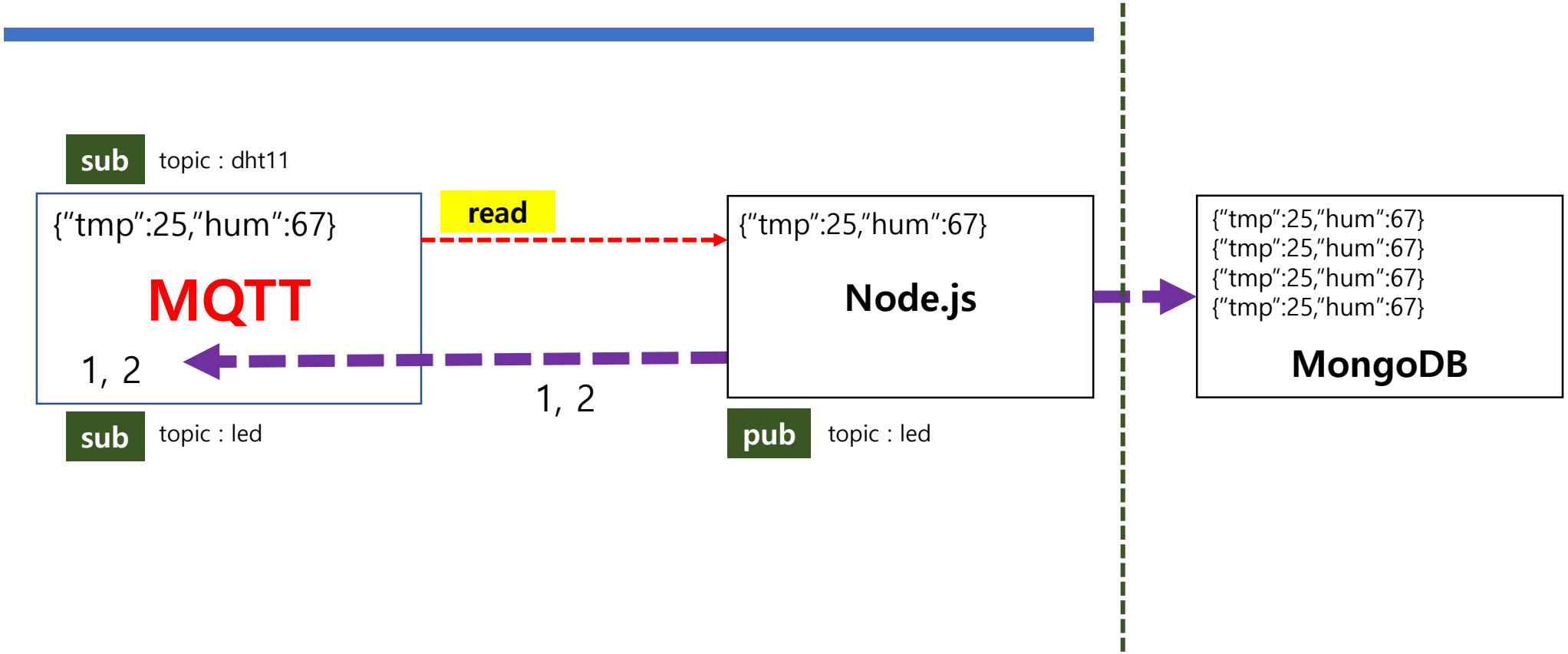
```
C:₩MQTTProject₩mosquitto>mosquitto_pub -t led -m 1

C:₩MQTTProject₩mosquitto>mosquitto_pub -t led -m 2

C:₩MQTTProject₩mosquitto>
```

9. 컴파일, 업로드, 구동 TEST

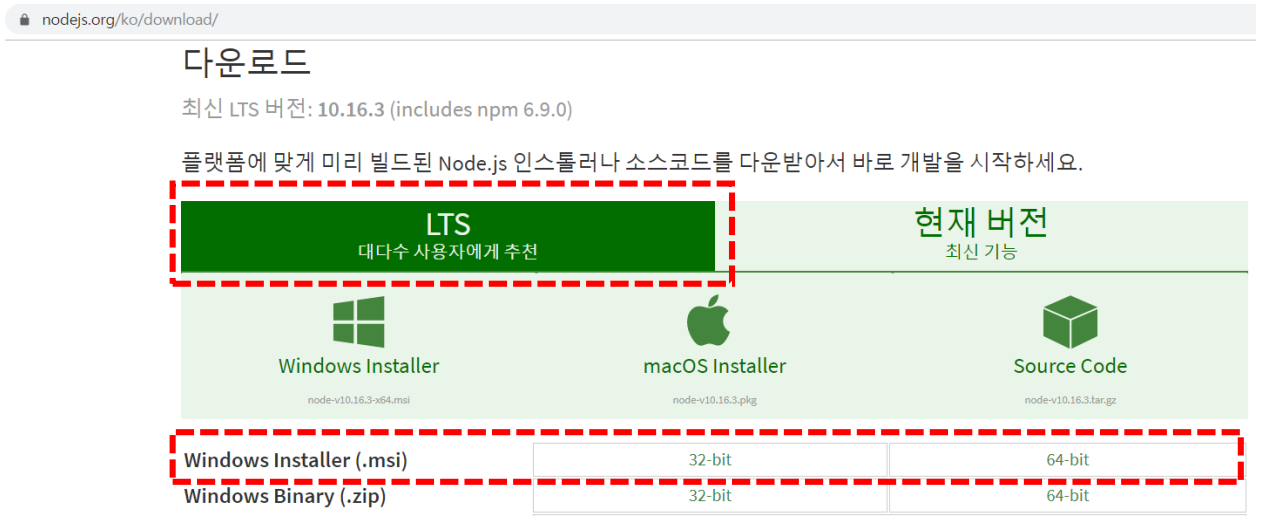


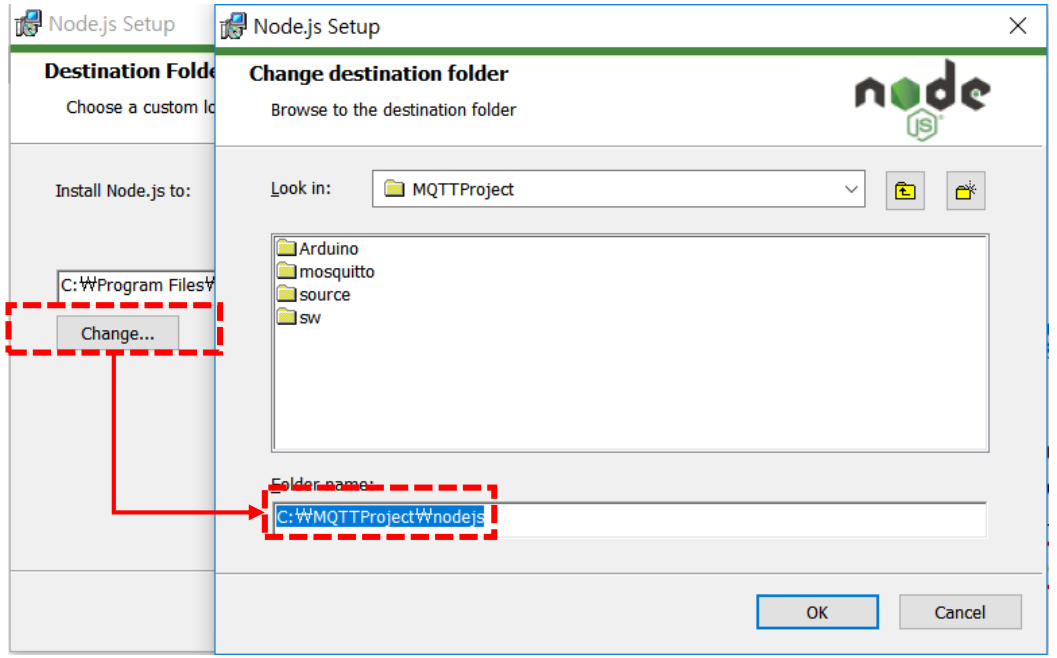


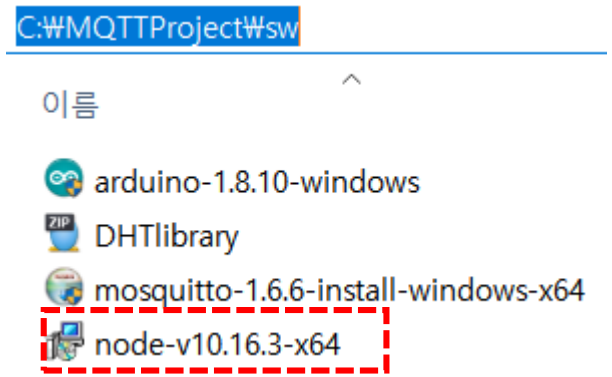
1. 구현내용

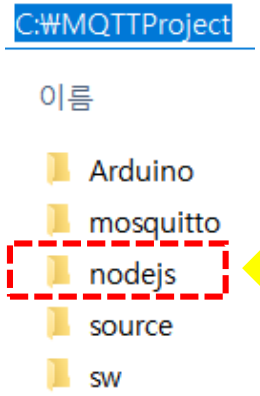

sub
topic : dht11
{"tmp":25,"hum":67}
MQTT
1, 2
sub
topic : led
read
1, 2
{"tmp":25,"hum":67}
Node.js
pub
topic : led
{"tmp":25,"hum":67}
{"tmp":25,"hum":67}
{"tmp":25,"hum":67}
{"tmp":25,"hum":67}
MongoDB

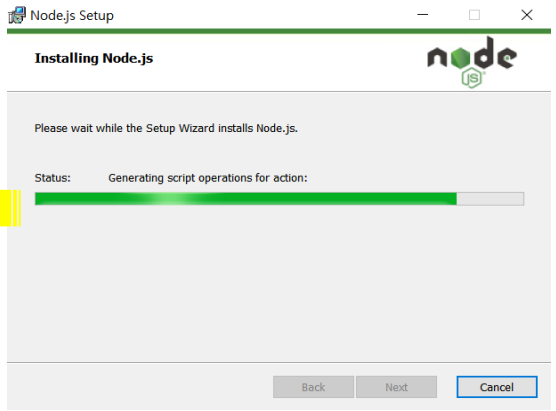
2. Node.js 다운로드 및 설치 → https://nodes.org/ko/download

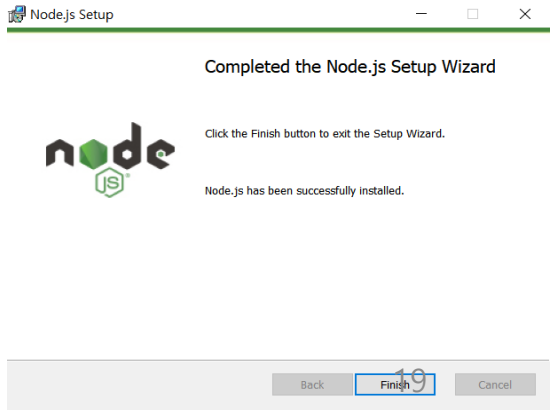






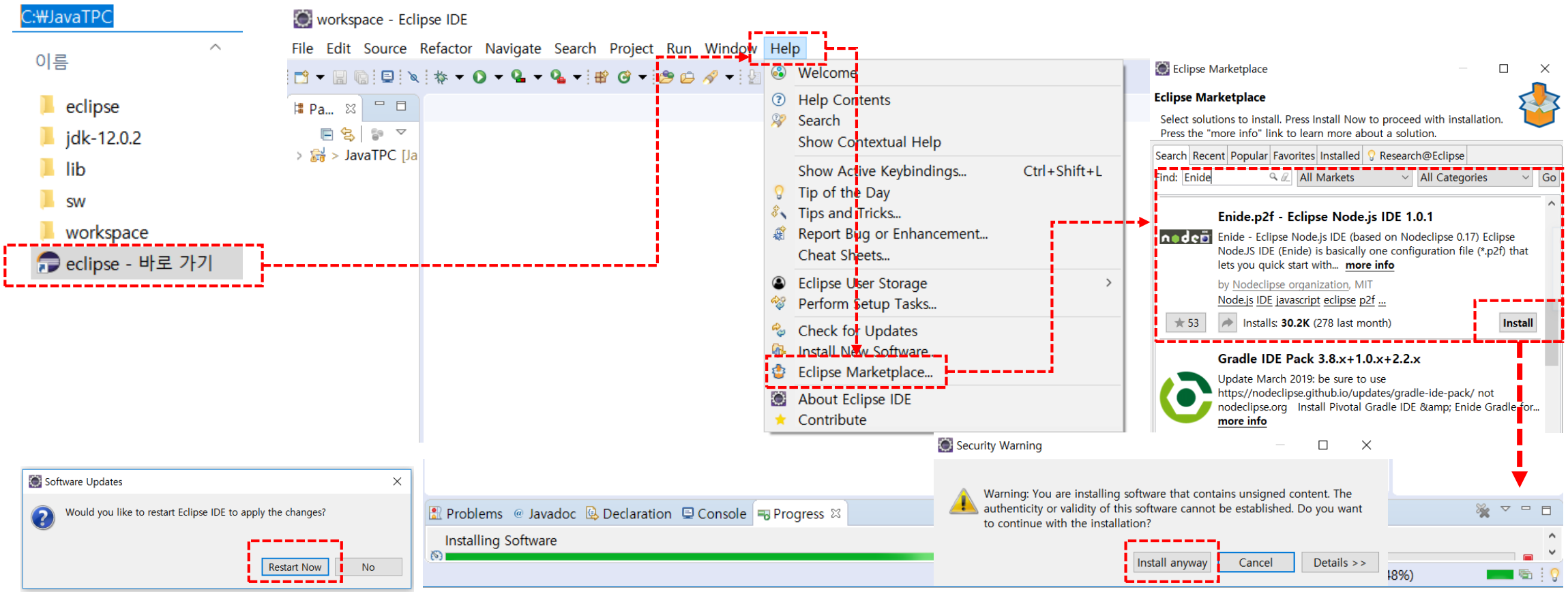




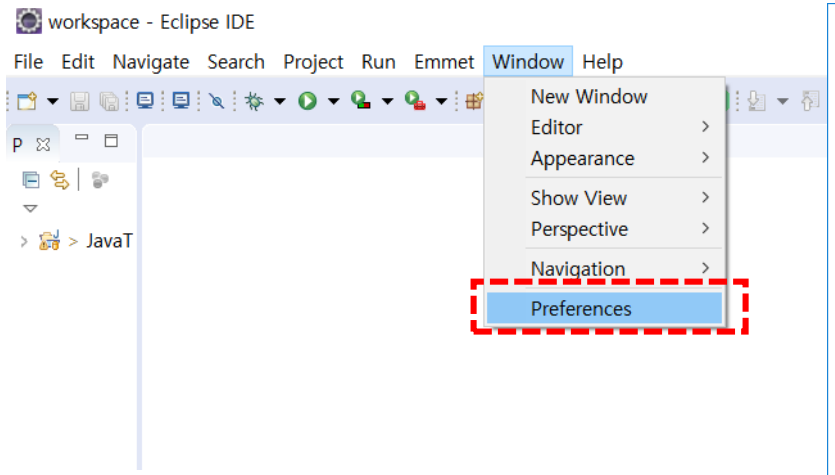


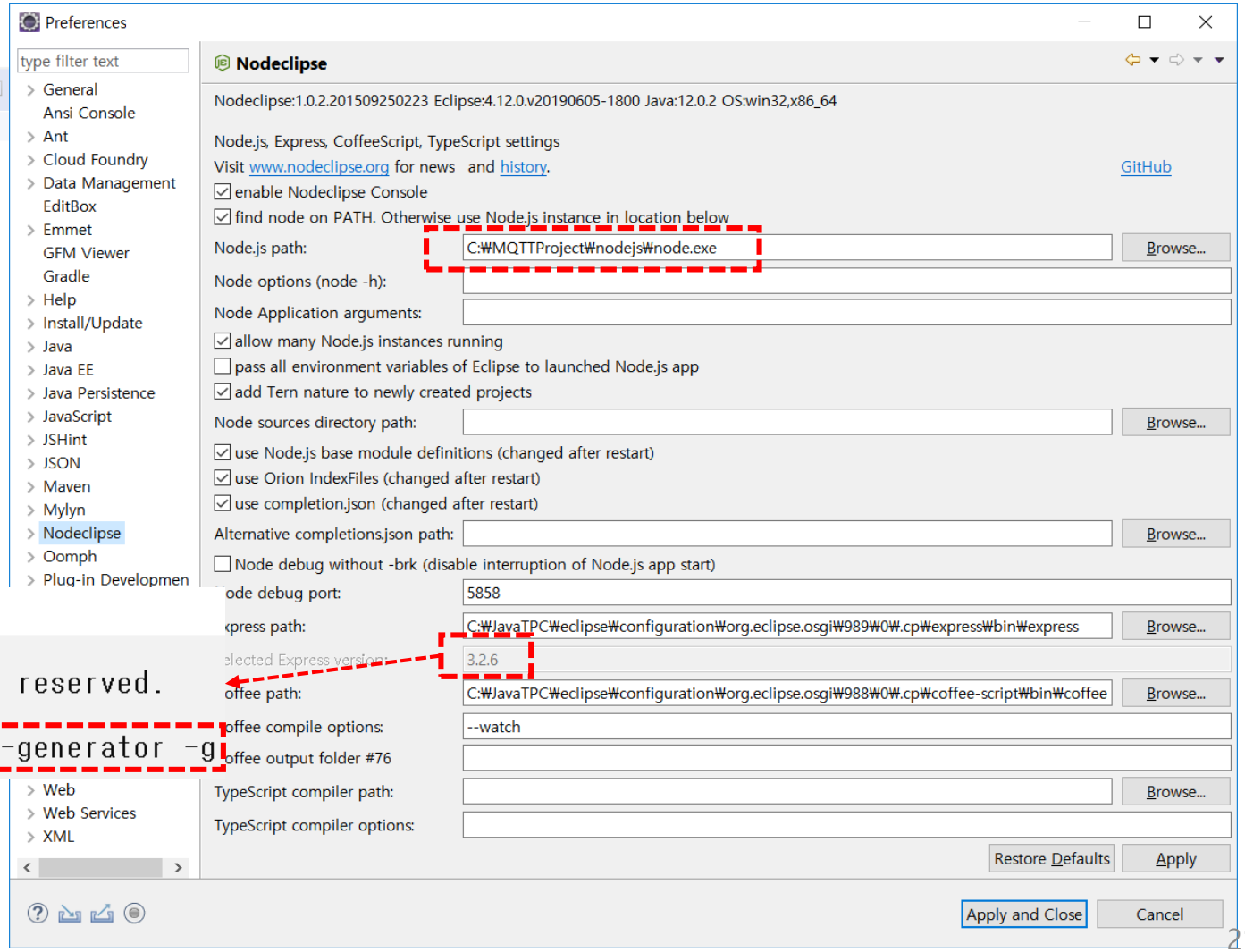
3. Node.js 개발 Eclipse IDE 설치 : Java TPC 1강 참고

: Eclipse 구동 -> Help -> Eclipse Marketplace -> Find : Enide.p2f 검색 -> Install



4. Node.js 개발 Eclipse IDE 설치 : Java TPC 1강 참고



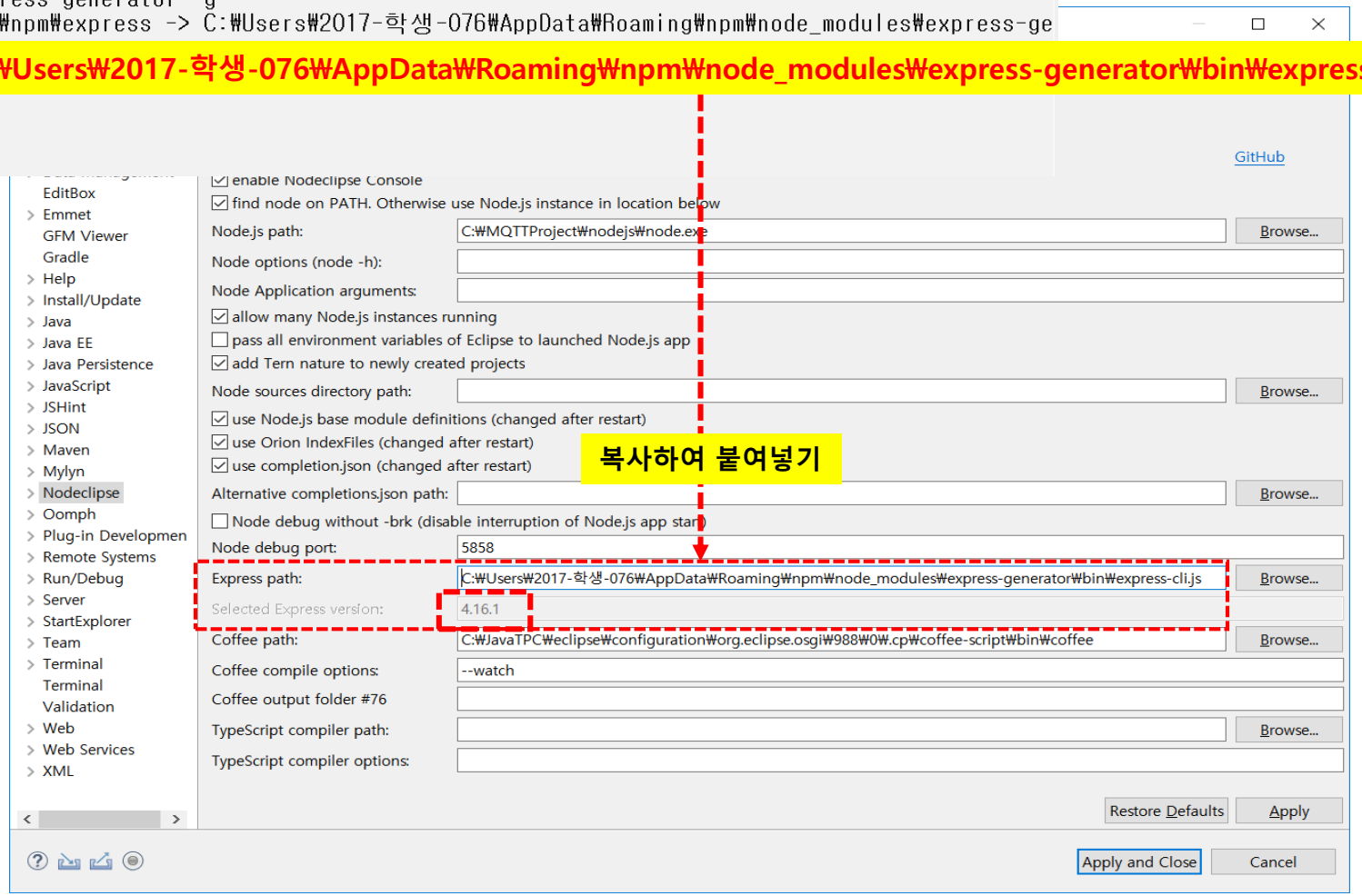


- Express 버전 확인
  (3.X -> 4.X 버전으로 변경)

5. Node.js 개발 Eclipse IDE 설치

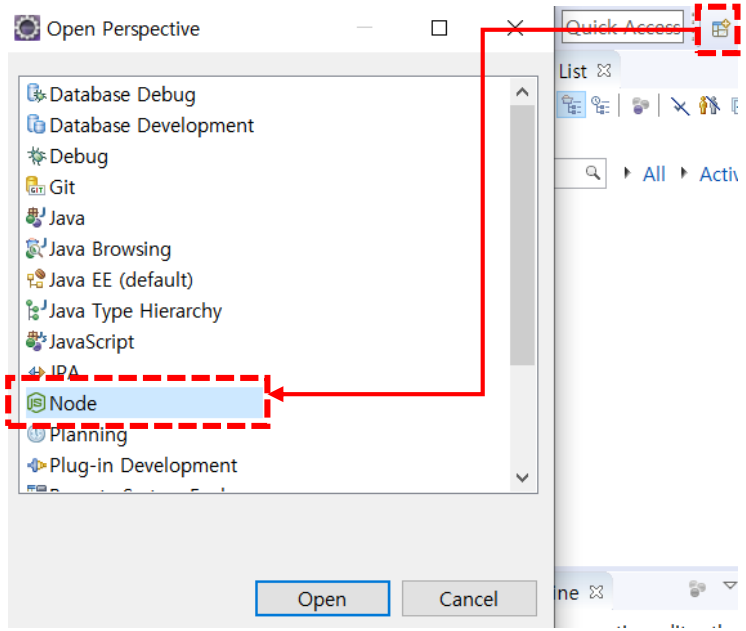
```
명령 프롬프트
Microsoft Windows [Version 10.0.17134.950]
(c) 2018 Microsoft Corporation. All rights reserved.

C:₩Users₩2017-학생-076>npm install express-generator -g
C:₩Users₩2017-학생-076₩AppData₩Roaming₩npm₩express -> C:₩Users₩2017-학생-076₩AppData₩Roaming₩npm₩node_modules₩express-ge
nerator₩bin₩express-cli.js
+ express-generator@4.16.1
updated 3 packages in 4.362s

C:₩Users₩2017-학생-076>
```

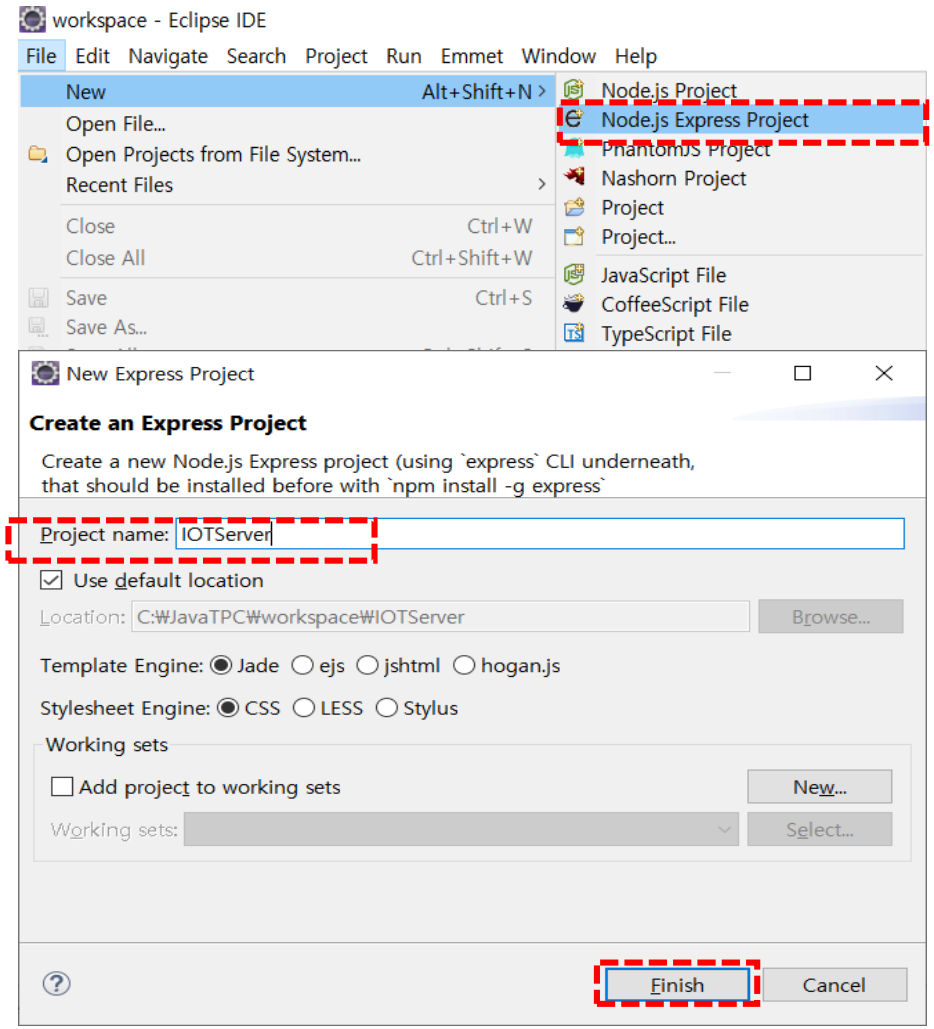
C:₩Users₩2017-학생-076₩AppData₩Roaming₩npm₩node_modules₩express-generator₩bin₩express-cli.js

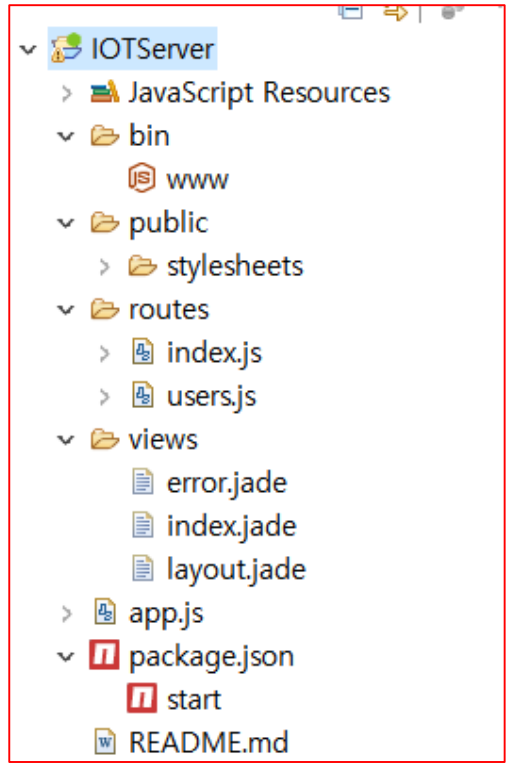


5. Node.js IOT Server 프로그래밍

→ Node.js Express Project 만들기(WEB)





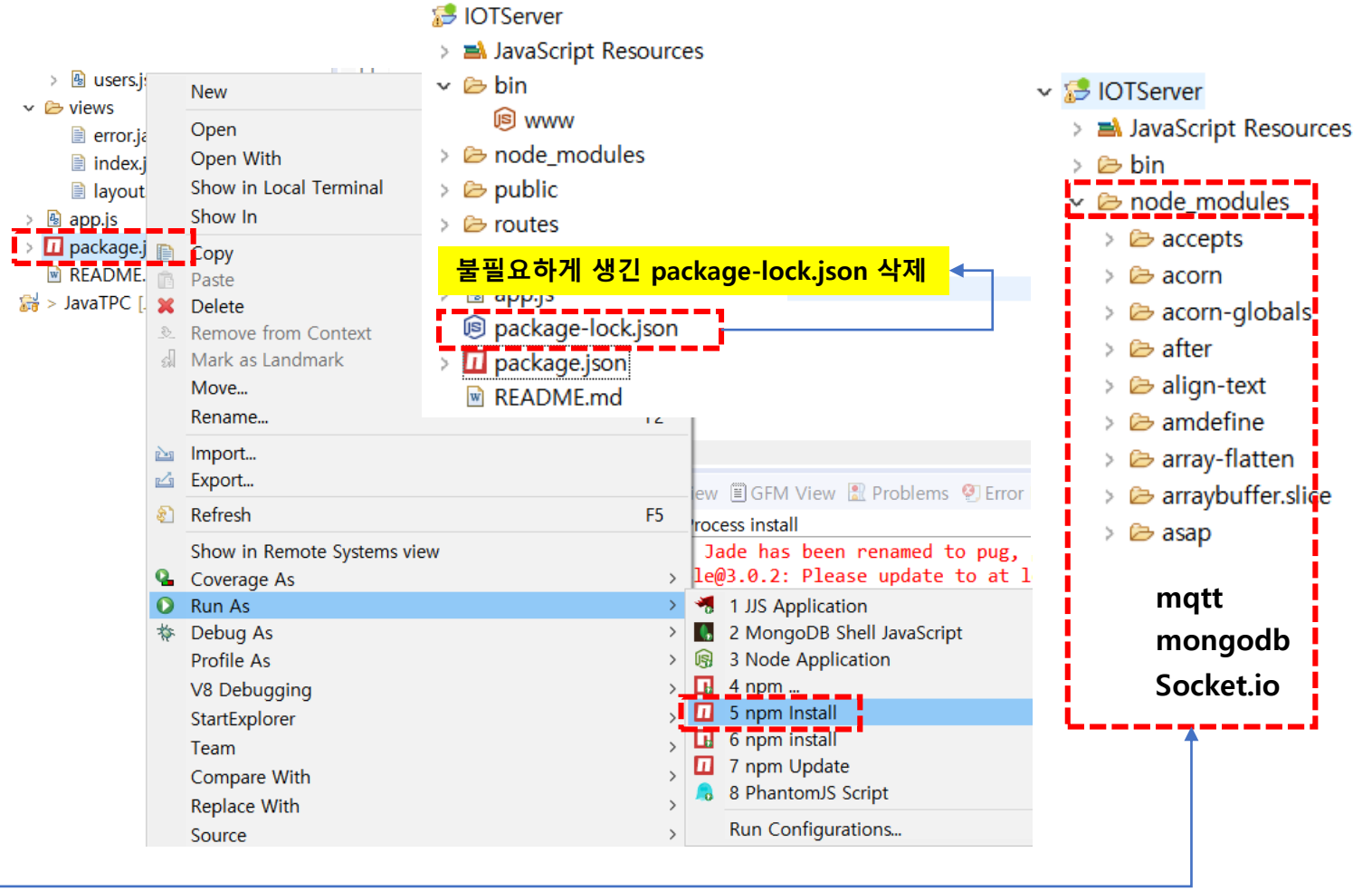


5. Node.js IOT Server 프로그래밍

→ package 설치하기

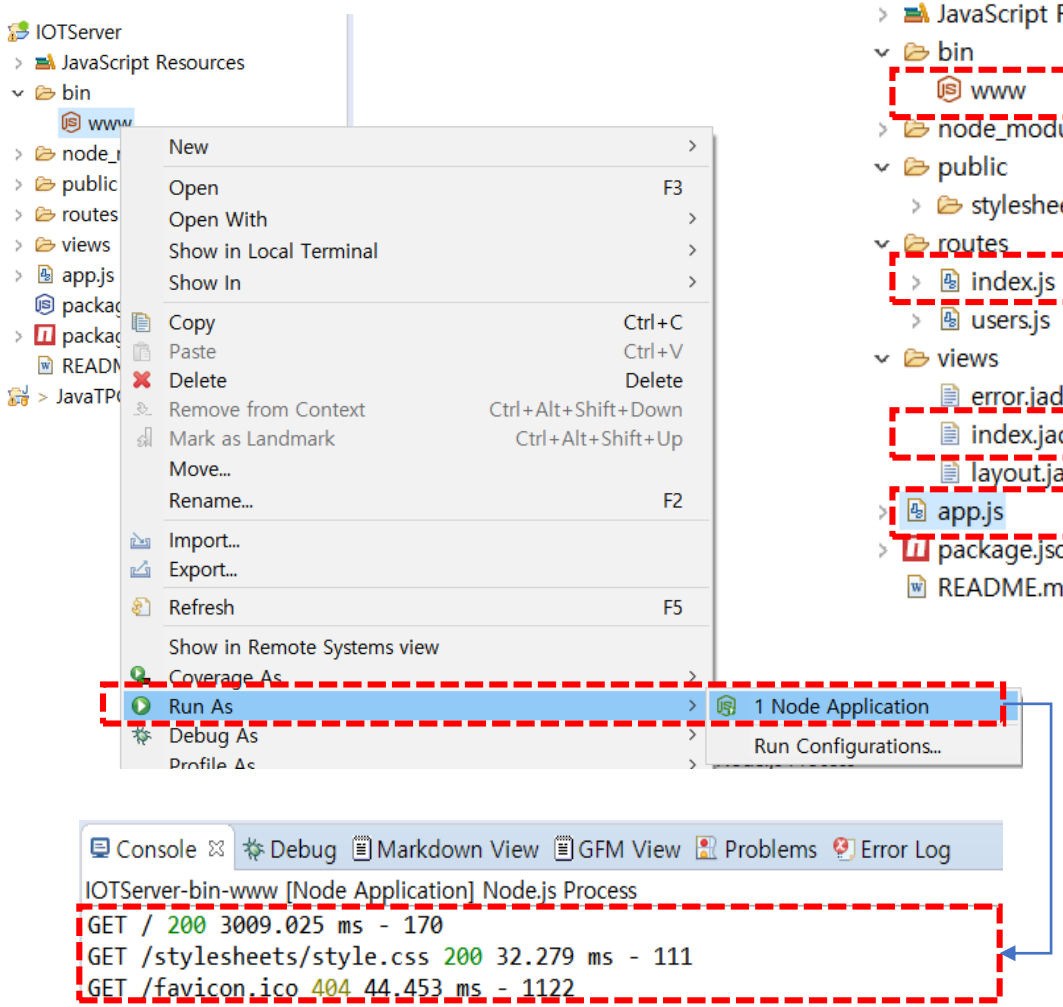
package.json

```
{
  "name": "iotserver",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "jade": "~1.11.0",
    "morgan": "~1.9.1",
    "mqtt" : "^2.14.0",
    "mongodb" : "^2.2.33",
    "socket.io" : "^2.0.4"
  }
}
```



5. Node.js IOT Server 프로그래밍

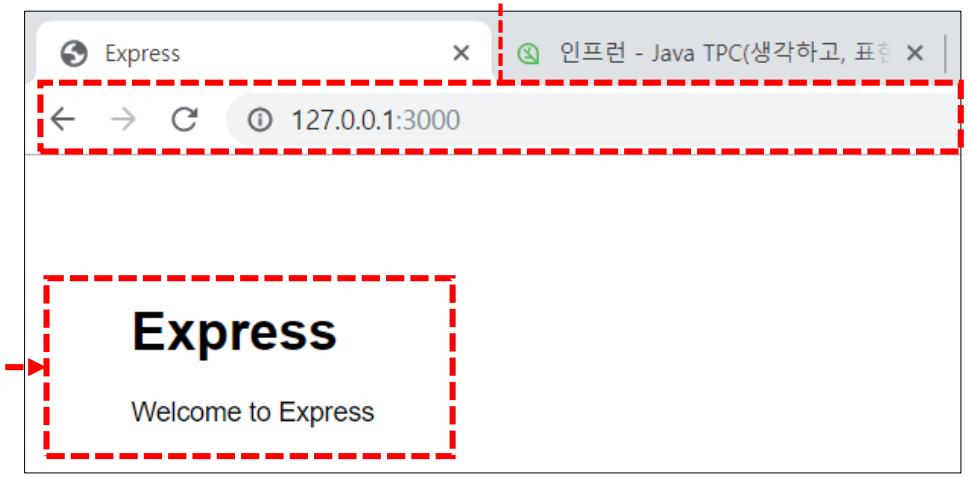
→ Server 구동 및 TEST



3000port 가 개방된다(즉 Web Server 실행된다)

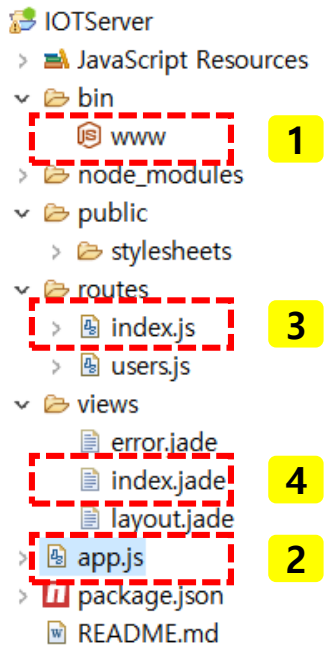
```
var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');
```

```
app.use('/', indexRouter);
app.use('/users', usersRouter);
```



5. Node.js IOT Server 프로그래밍

→ Express Project 구동 절차

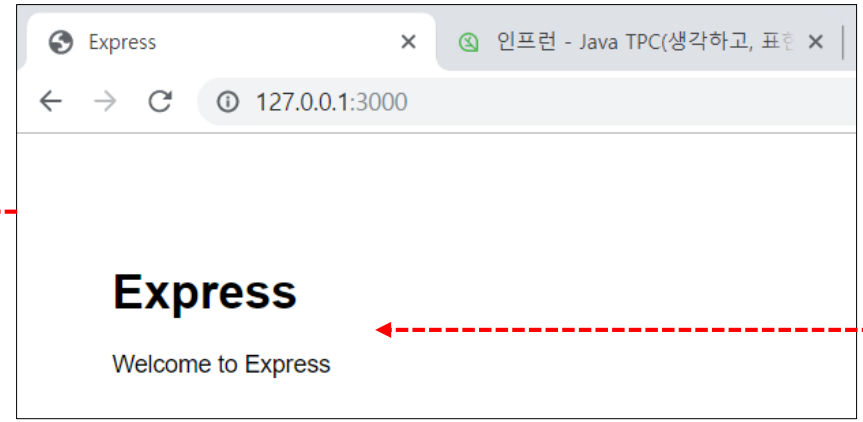


Web Server 구동(3000 PORT)

```
var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);
```

요청

http://127.0.0.1:3000/



app.js

```
var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
app.use('/users', usersRouter);
```

index.js

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

module.exports = router;
```

index.jade

```
extends layout

block content
  h1= title
  p Welcome to #{title}
```

5. Node.js IOT Server 프로그래밍

→ MQTT Server에 연결 하여 DHT11 센서 데이터 읽어 오기 구현(www 파일 수정)

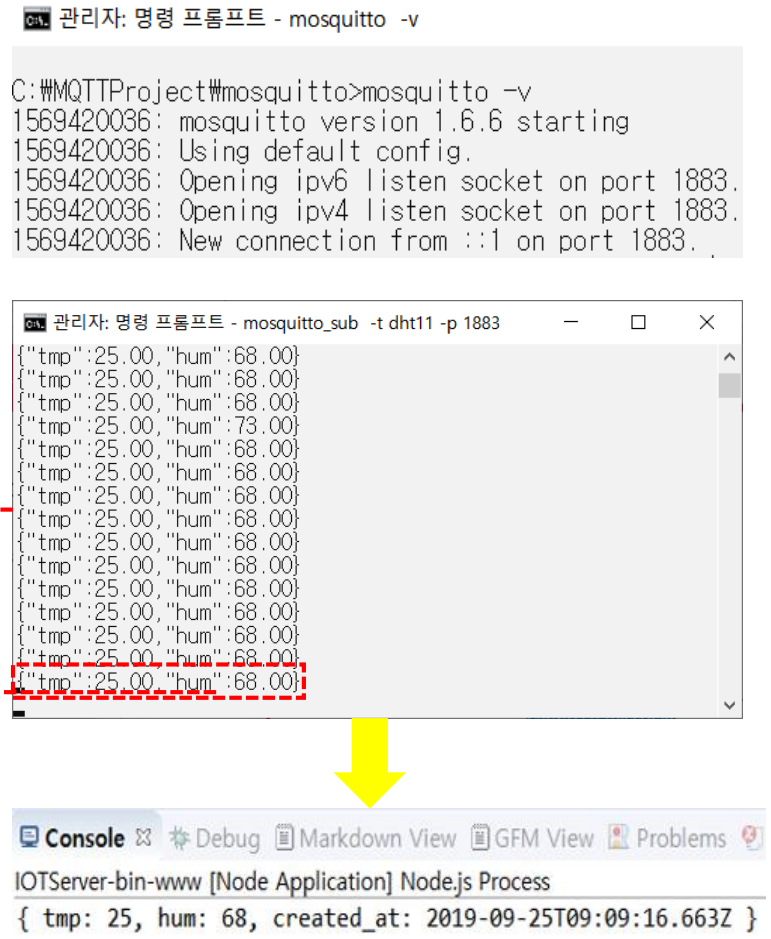
```
//mqtt 연결
var mqtt=require("mqtt");
var client=mqtt.connect("mqtt://172.30.1.15");

client.on("connect", function(){
     client.subscribe("dht11");
});

client.on("message", function(topic, message){
     //console.log(topic+":"+message.toString()
     );
     var obj=JSON.parse(message);
     obj.created_at=new Date();
     console.log(obj);
     var dht11=dbObj.collection("dht11");
     dht11.save(obj, function(err,result){
          if(err) console.log(err);
          else
          console.log(JSON.stringify(result));
     });
});
```

connect event 발생

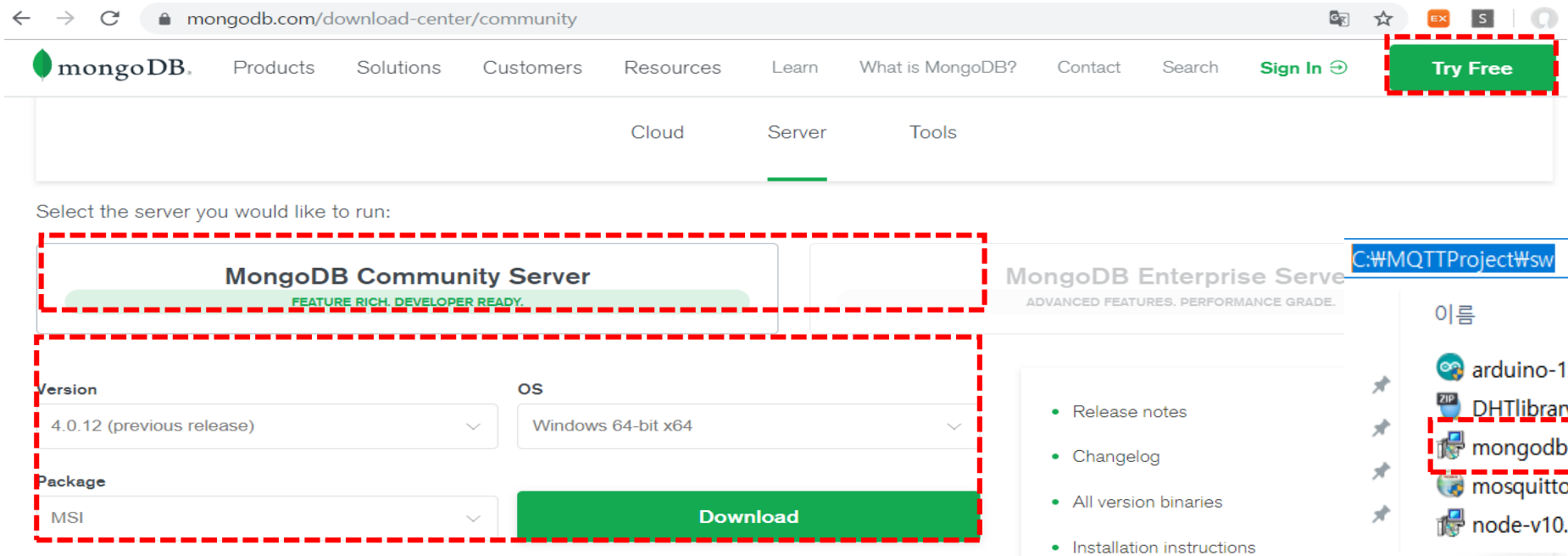
dht11 구독자 등록

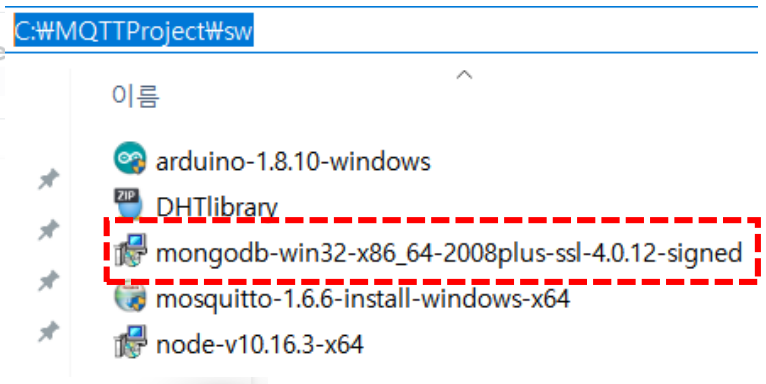
message event 발생

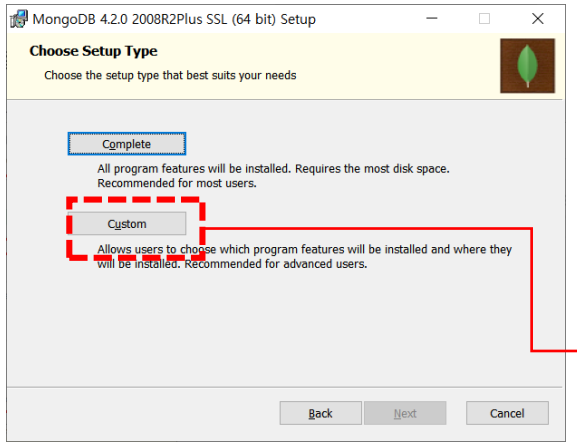
mongodb에 저장하는 부분

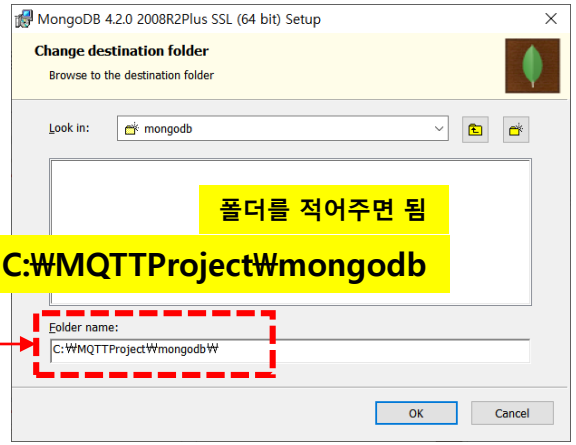


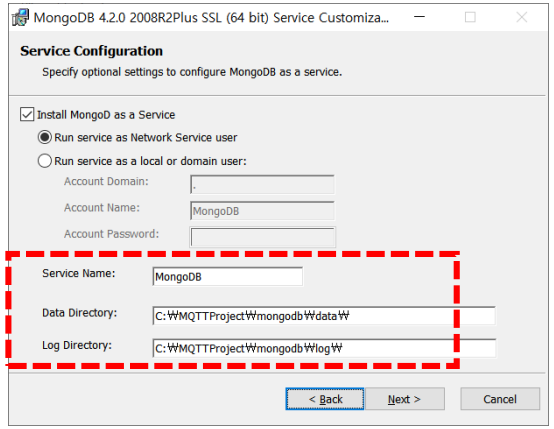
1. MongoDB 설치 : https://mongodb.com

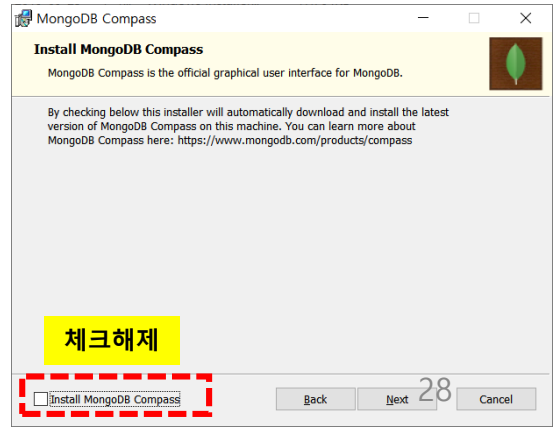




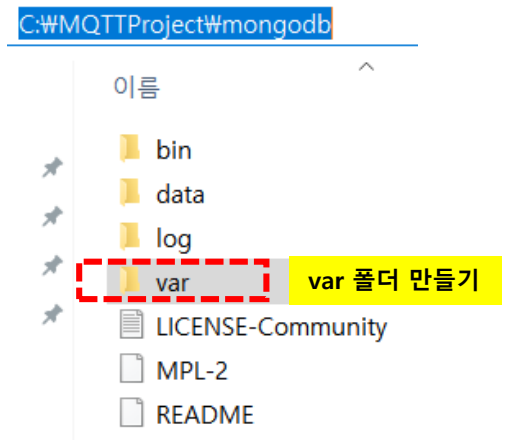








2. MongoDB 구동하기



1. **var 폴더 만들기**
2. **MongoDB Server 구동하기**

C:₩MQTTProject₩mongodb₩bin>**mongod --dbpath C:₩MQTTProject₩mongodb₩var**

2019-09-25T16:35:13.670+0900 I NETWORK [initandlisten] **waiting for connections on port 27017**

3. MongoDB 접속하기

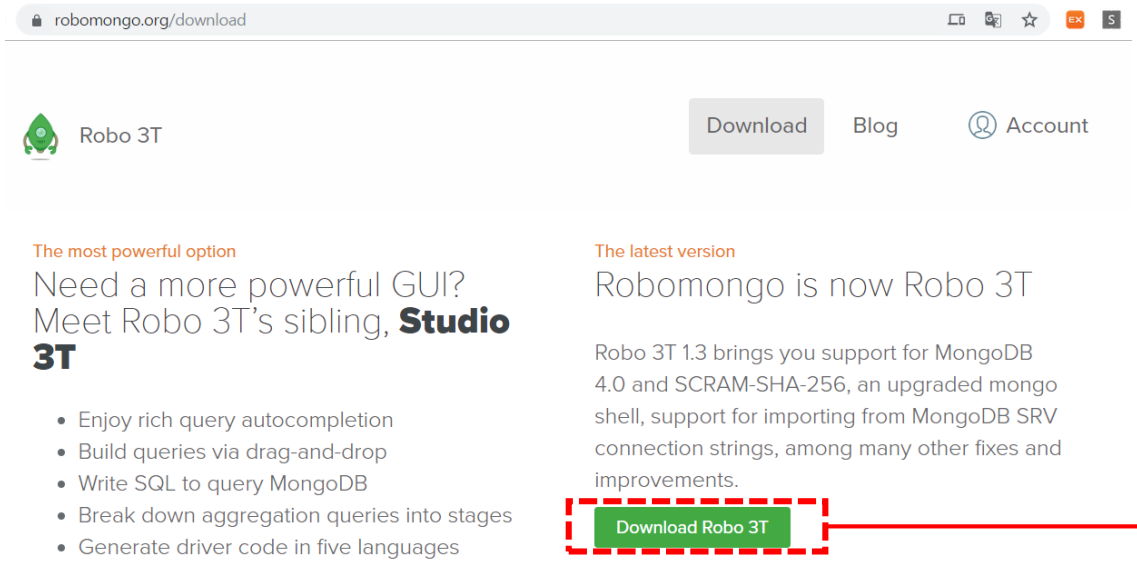
C:₩MQTTProject₩mongodb₩bin>**mongo**

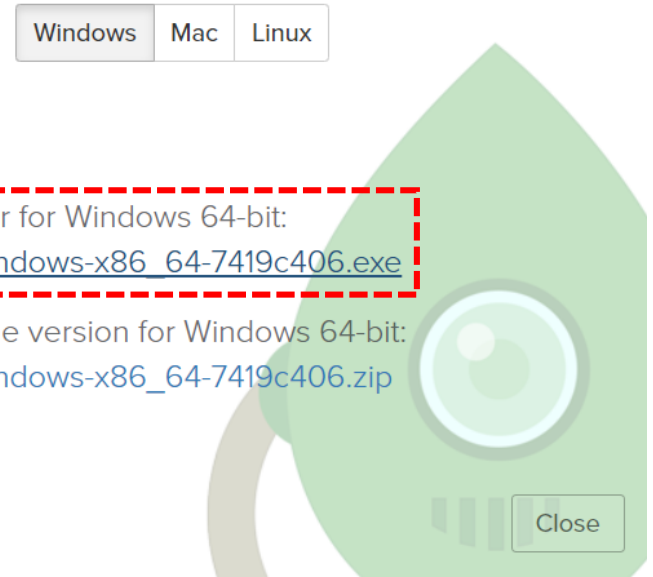
```
MongoDB shell version v4.0.12
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b54feb1a-fa86-4d55-bf09-70403af6f8e9") }
MongoDB server version: 4.0.12

>
```

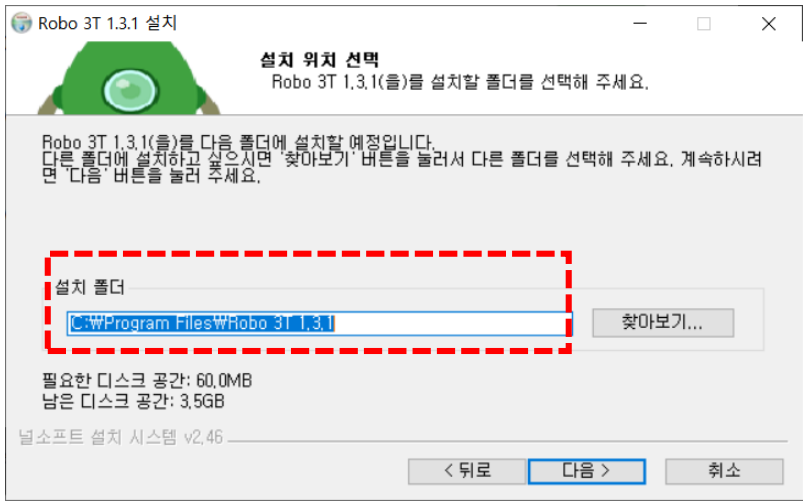
4. MongoDB 종료하기

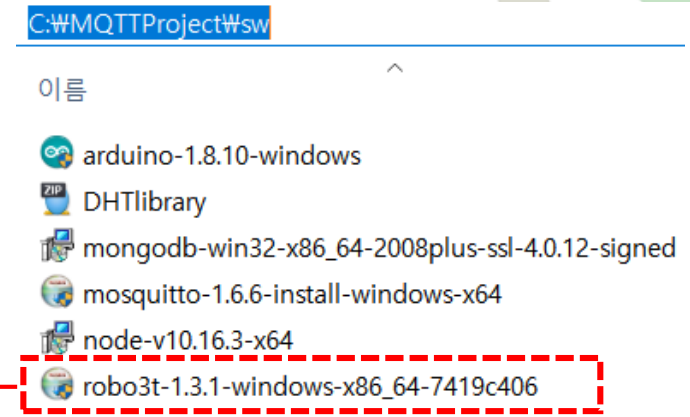
```
> use admin
  switched to db admin
> db.shutdownServer()
```

3. Robomongo 설치하기(Robo 3T) https://robomongo.org







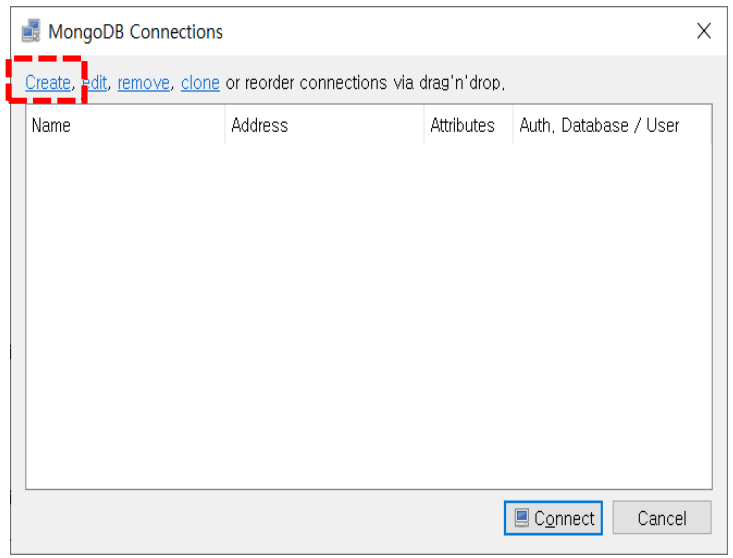


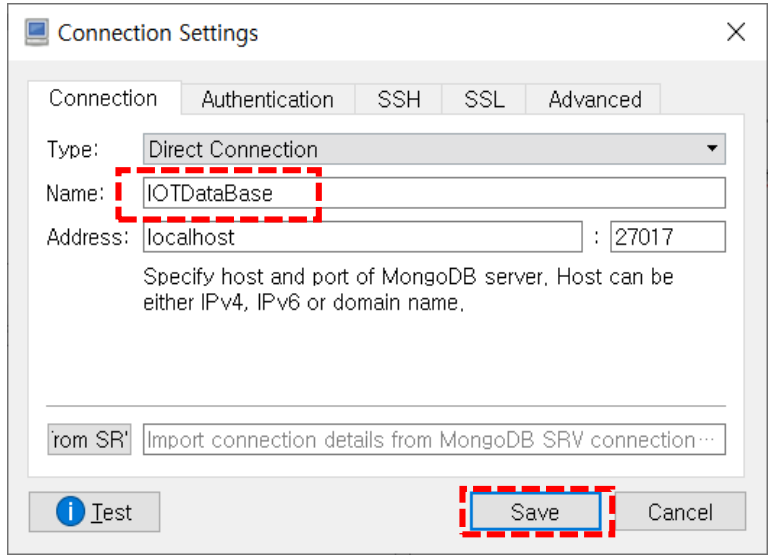
4. Robo 3T 구동 및 DataBase 만들기

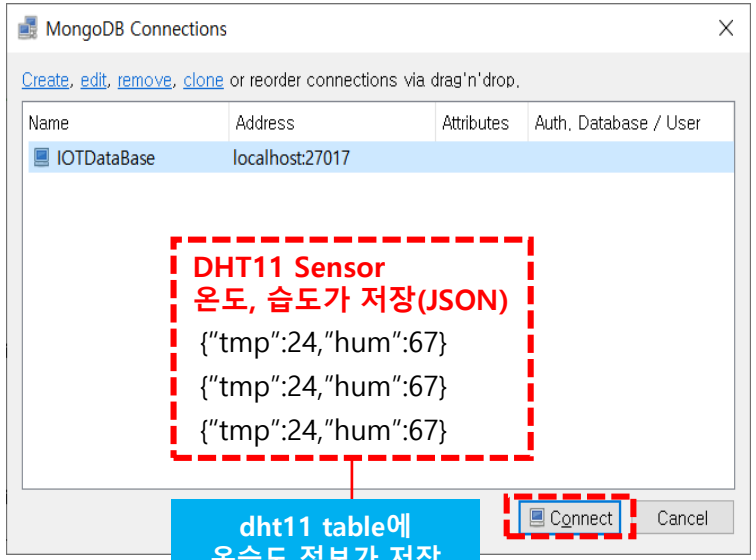
먼저 서버를 구동

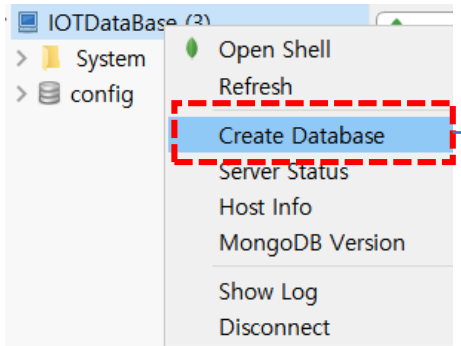
C:₩MQTTProject₩mongodb₩bin>mongod --dbpath C:₩MQTTProject₩mongodb₩var

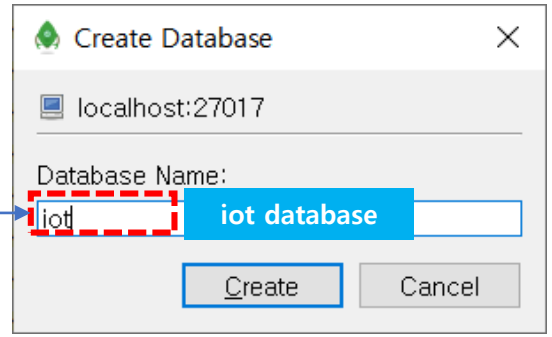
2019-09-25T17:35:13.616+0900 I NETWORK [initandlisten] waiting for connections on port 27017

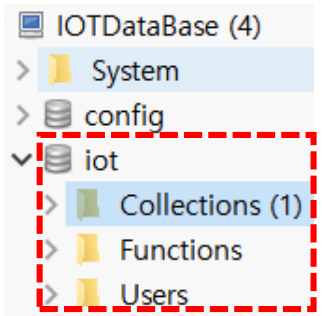


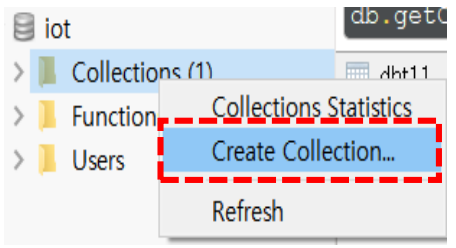


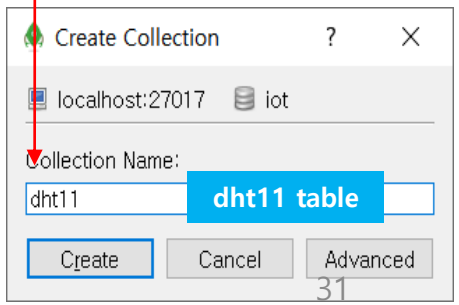










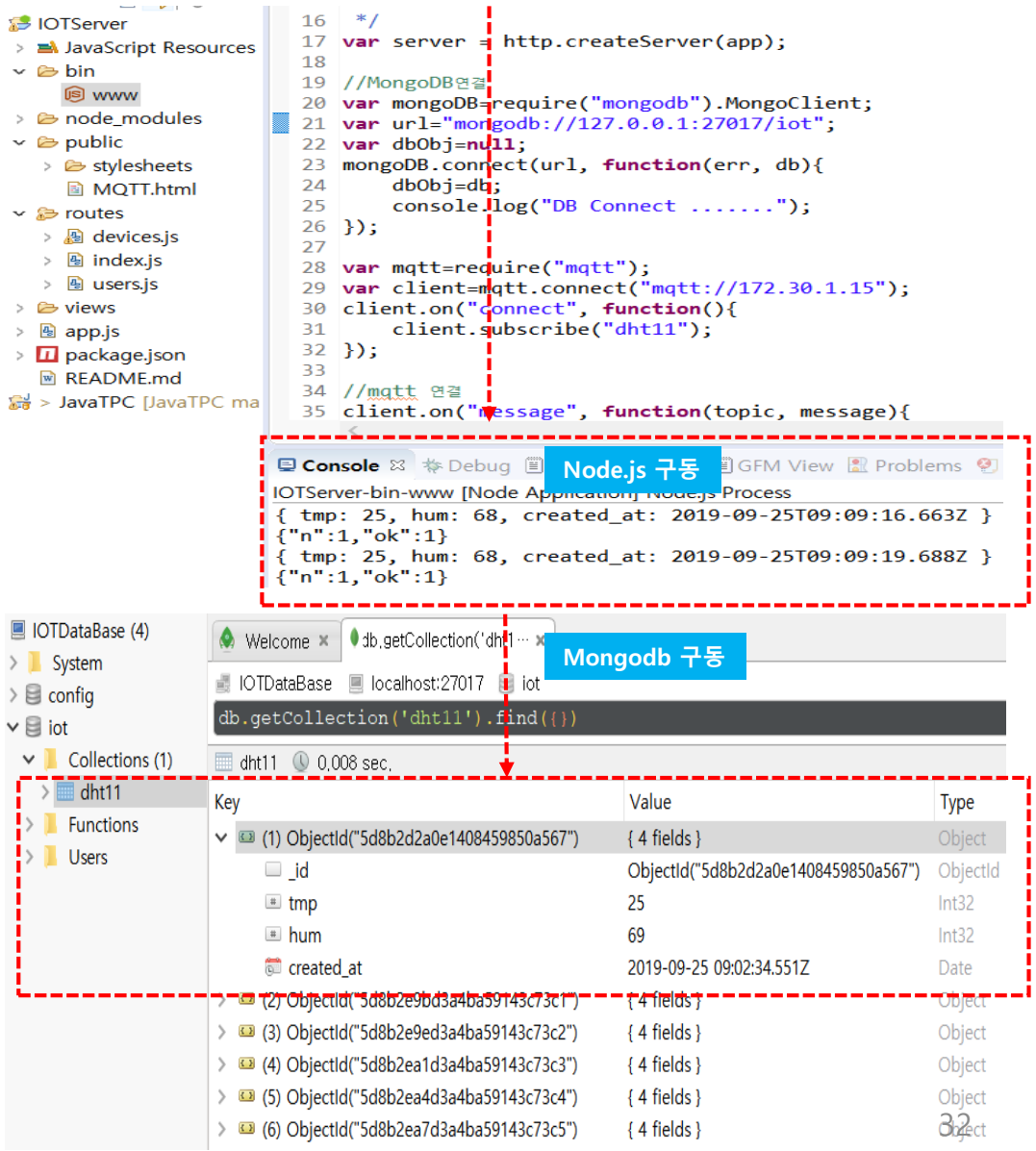


5. Node.js IOT Server 프로그래밍

→ DHT11 센서 데이터를 mongodb에 저장하기(www 파일 수정)

```
// MongoDB 연결
var mongoDB=require("mongodb").MongoClient;
var url="mongodb://127.0.0.1:27017/iot";
var dbObj=null;
mongoDB.connect(url, function(err, db){
      dbObj=db;
      console.log("DB Connect .......");
});
//mqtt 연결
var mqtt=require("mqtt");
var client=mqtt.connect("mqtt://172.30.1.15");

client.on("connect", function(){
      client.subscribe("dht11");
});

client.on("message", function(topic, message){
      //console.log(topic+":"+message.toString());
      var obj=JSON.parse(message);
      obj.created_at=new Date();
      console.log(obj);
      var dht11=dbObj.collection("dht11");
      dht11.save(obj, function(err,result){
            if(err) console.log(err);
            else console.log(JSON.stringify(result));
      });
});
```

WeMos 보드에 전원이 켜져 있어야 됨



5. Node.js IOT Server 프로그래밍

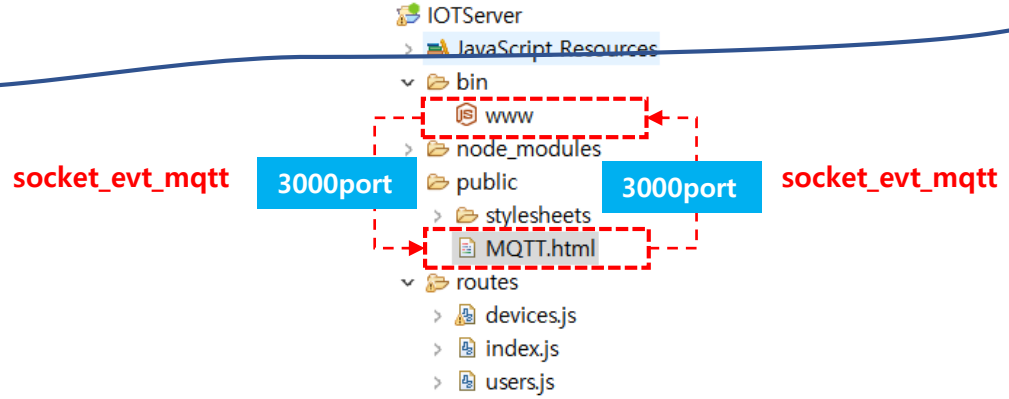
→ 소켓(socket)을 이용하여 JavaScript와 통신하기(www 파일 수정)

```
//web과 socket 통신
var io=require('socket.io')(server);
io.on("connection", function(socket){
    socket.on("socket_evt_mqtt", function(data){
            var dht11=dbObj.collection("dht11");
            dht11.find({}).sort({_id:-1}).limit(1).toArray(function(err, results){
                if(!err){
                  socket.emit("socket_evt_mqtt", JSON.stringify(results[0]));
                }
            });
    });

    socket.on("socket_evt_led",function(data){
        var obj=JSON.parse(data);
         client.publish("led", obj.led+'');
    });
});
```

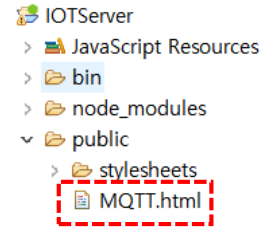
DHT11

데이터(온도,습도)를 보냅니다.

LED

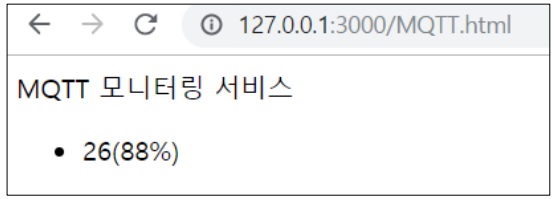


MQTT.html




```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Insert title here</title>
<script src="/socket.io/socket.io.js"></script>
<script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
<script type="text/javascript">
  var socket=null;
  var timer=null;
  $(document).ready(function(){
      socket=io.connect(); // 3000port
      // Node.js보낸 데이터를 수신하는 부분
      socket.on("socket_evt_mqtt", function(data){
        data=JSON.parse(data);
        $(".mqttlist").html('<li>'+data.tmp+'('+data.hum+'%)'+'</li>');
      });
      if(timer==null){
          timer=window.setInterval("timer1()", 1000);
      }
  });
  function timer1(){
      socket.emit("socket_evt_mqtt", JSON.stringify({}));
      console.log("---------");
  }
</script>
</head>
<body>
MQTT 모니터링 서비스
<div id="msg">
   <div id="mqtt_logs">
      <ul class="mqttlist"></ul>
   </div>
</div>
</body>
</html>
```

데이터(온도,습도)를 주세요



5. Node.js IOT Server 프로그래밍

→ 소켓(socket)을 이용하여 LED 제어하기(www 파일 수정)

```
//web과 socket 통신
var io=require('socket.io')(server);
io.on("connection", function(socket){
        socket.on("socket_evt_mqtt", function(data){
                var dht11=dbObj.collection("dht11");
                dht11.find({}).sort({_id:-1}).limit(1).toArray(function(err, results){
                    if(!err){
                      socket.emit("socket_evt_mqtt", JSON.stringify(results[0]));
                    }
                });
    });
```

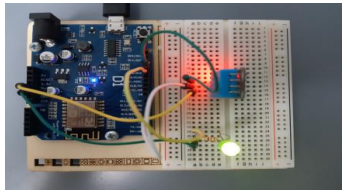
DHT11

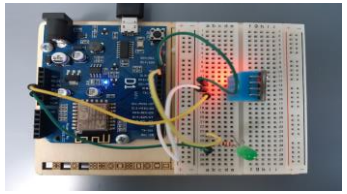
데이터(온도,습도)를 보냅니다.

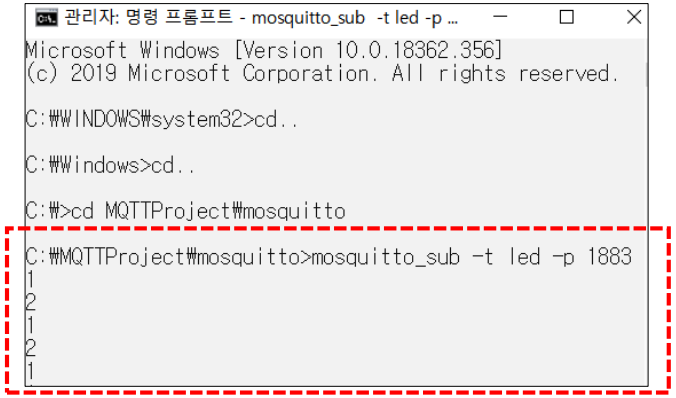
```
socket.on("socket_evt_led",function(data){
        var obj=JSON.parse(data);
        client.publish("led", obj.led+'');
    });
});
```

LED

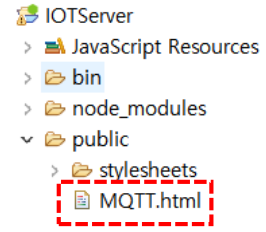
MQTT led topic 으로 발행



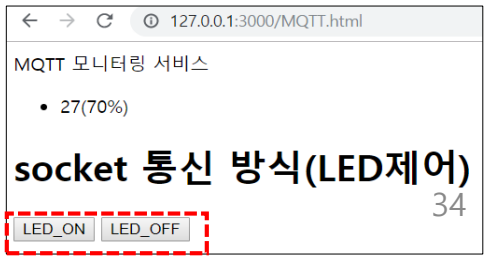




MQTT.html




```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Insert title here</title>
<script src="/socket.io/socket.io.js"></script>
<script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
<script type="text/javascript">
  var socket=null;
  var timer=null;
  $(document).ready(function(){
        socket=io.connect(); // 3000port
                // Node.js보낸 데이터를 수신하는 부분

  });
  function ledOnOff(value){
     // {"led":1}, {"led":2}
     socket.emit("socket_evt_led", JSON.stringify({led:Number(value)}));
  }
</script>
</head>
<body>
MQTT 모니터링 서비스
<div id="msg">
   <div id="mqtt_logs">
      <ul class="mqttlist"></ul>
   </div>
   <h1>socket 방식통신</h1>
   <button onclick="ledOnOff(1)">LED_ON</button>
   <button onclick="ledOnOff(2)">LED_OFF</button>

</div>
</body>
</html>
```

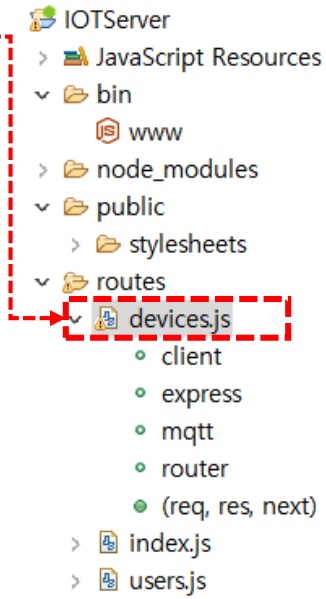


5. Node.js IOT Server 프로그래밍

→ RESTfull 서비스를 이용한 LED 제어하기(app.js 파일 수정)

app.js

```
var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');
var devicesRouter = require('./routes/devices'); //devices.js

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
app.use('/users', usersRouter);
app.use('/devices', devicesRouter);
```



MQTT.html

```
<script type="text/javascript" src="/socket.io/socket.io.js"></script>
<script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
<script type="text/javascript">
  function ajaxledOnOff(value){
      if(value=='1') var value="on";
      else if(value=="2") var value="off";
      $.ajax({
         url:"http://172.30.1.15:3000/devices/led/"+value,
         type:"post",
         success : ledStatus,
         error : function(){ alert("error");}
      });
  }
  function ledStatus(obj){
    $("#led").html("<font color='red'>"+obj.led+"</font> 되었습니다.");
  }
</script>
</head>
<body>
MQTT 모니터링 서비스
<div id="msg">
   <div id="mqtt_logs">
      <ul class="mqttlist"></ul>
   </div>
   <h1>REST full Service 통신 방식(LED제어)</h1>
   <button onclick="ajaxledOnOff(1)">LED_ON</button>
   <button onclick="ajaxledOnOff(2)">LED_OFF</button>
   <div id="led">LED STATUS</div>
</div>
</body>
</html>
```

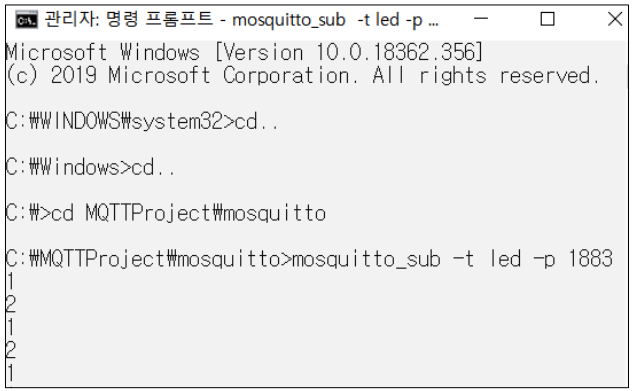
5. Node.js IOT Server 프로그래밍

→ **RESTfull 서비스를 이용한 LED 제어하기(devices.js 파일 만들기)**

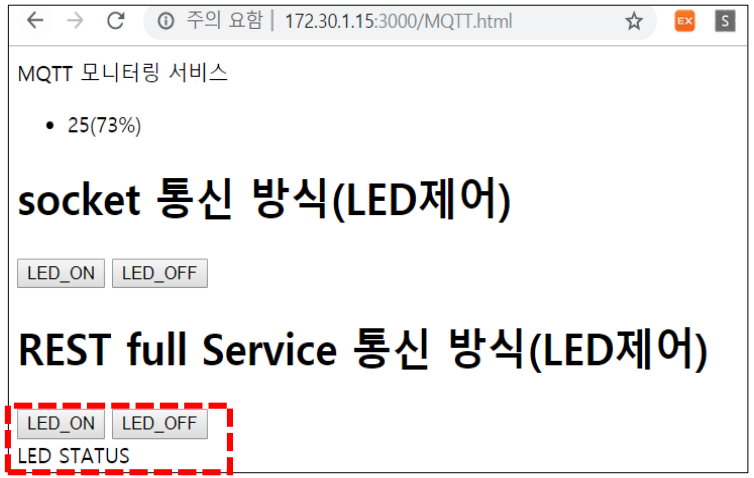
devices.js

```
var express = require('express');
var router = express.Router();

var mqtt=require("mqtt");
var client=mqtt.connect("mqtt://172.30.1.15");

/* GET home page. */
router.post('/led/:flag', function(req, res, next) {
   res.set('Content-Type', 'text/json');
   if(req.params.flag=="on"){
       // MQTT->led : 1
       client.publish("led", '1');
       res.send(JSON.stringify({led:'on'}));
   }else{
       client.publish("led", '2');
       res.send(JSON.stringify({led:'off'}));
   }
});

module.exports = router;
```

http://172.30.1.15:3000/devices/led/on
http://172.30.1.15:3000/devices/led/off



반드시 자신의 IP조 요청할 것

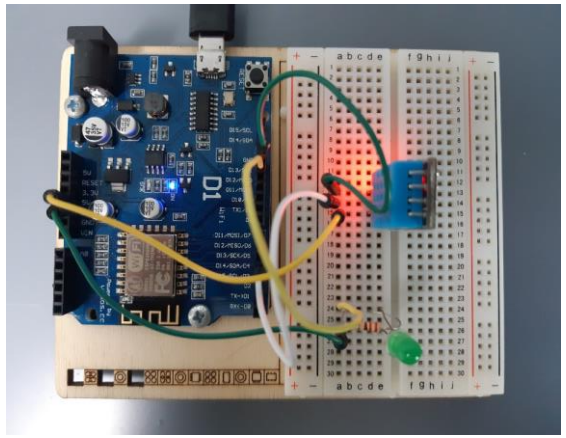


LED_ON LED_OFF
on 되었습니다.

LED_ON LED_OFF
off 되었습니다.

9. 시연하기

7 WeMos 구동



관리자: 명령 프롬프트 - mosquitto -v

1

```
C:₩MQTTProject₩mosquitto>mosquitto -v
1569420036: mosquitto version 1.6.6 starting
1569420036: Using default
1569420036: Opening ipv6
1569420036: Opening ipv4 listen socket on port 1883.
1569420036: New connection from ::1 on port 1883.
```

MQTT Server구동

관리자: 명령 프롬프트 - mosquitto_sub -t dht11 -p 1883

```
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":73.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
{"tmp":25.00,"hum":68.00}
```

2 dht11 sub 구동

선택 관리자: 명령 프롬프트 - mosquitto_sub -t led -p 1883

```
C:₩Windows>cd..

C:₩>cd MQTTProject₩mosquitto

C:₩MQTTProject₩mosquitto>mosquitto_sub -t led -p 1883
0
1
1
1
2
1
2
1
2
```

3 led sub 구동

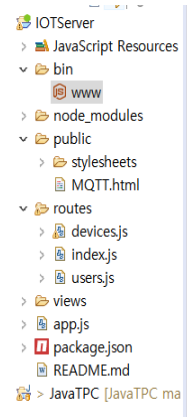
작업 관리자

파일(F) 옵션(O) 보기(V)

프로세스 | 성능 | 앱 기록 | 시작프로그램 | 사용자 | 세부 정보

이름	PID	상태
Microsoft.Photos.exe	14552	일시 중단됨
MicrosoftEdge.exe	24916	일시 중단됨
MicrosoftEdgeCP.exe	16512	일시 중단됨
MicrosoftEdgeSH.exe	2440	일시 중단됨
ModuleCoreService....	5420	실행 중
mongod.exe	12004	실행 중
mosquitto.exe	5464	실행 중
mosquitto_sub.exe	3440	실행 중
mosquitto_sub.exe	17684	실행 중

5 Node.js 구동

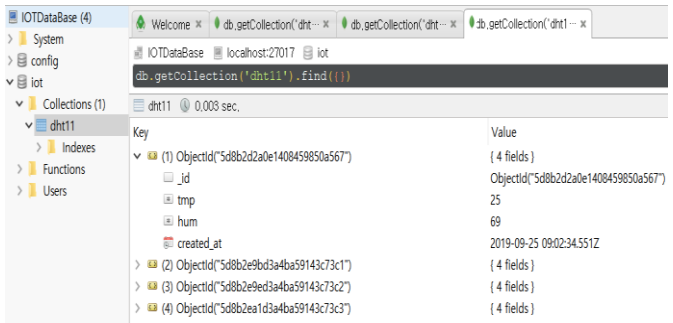



```
 */
var server = http.createServer(app);

//MongoDB연결
var mongoDB=require("mongodb").MongoClient;
var url="mongodb://127.0.0.1:27017/iot";
var dbObj=null;
mongoDB.connect(url, function(err, db){
    dbObj=db;
    console.log("DB Connect .......");
});

var mqtt=require("mqtt");
var client=mqtt.connect("mqtt://172.30.1.15");
client.on("connect", function(){
    client.subscribe("dht11");
});

//mqtt 연결
client.on("message", function(topic, message){
```

Console | Debug | Markdown View | GFM View | Problems

IOTServer-bin-www [Node Application] Node.js Process

```
{ tmp: 25, hum: 68, created_at: 2019-09-25T09:09:16.663Z }
{"n":1,"ok":1}
{ tmp: 25, hum: 68, created_at: 2019-09-25T09:09:19.688Z }
{"n":1,"ok":1}
```

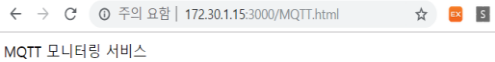



```
C:₩MQTTProject₩mongodb₩bin>mongod --dbpath C:₩MQTTProject₩mongodb₩var
2019-09-25T16:35:12.247+0900 I STORAGE  [main] Max cache overflow file size
2019-09-25T16:35:12.685+0900 I CONTROL  [main] Automatically disabling TLS
abledProtocols 'none'
2019-09-25T16:35:12.688+0900 I CONTROL  [initandlisten] MongoDB starting :
godb₩var 64-bit host=
2019-09-25T16:35:12.6                                  targetMinOS: Windows
2019-09-25T16:35:12.6                                  db version v4.0.12
2019-09-25T16:35:12.6                                  git version: 5776e3
2019-09-25T16:35:12.689+0900 I CONTROL  [initandlisten] allocator: tcmalloc
2019-09-25T16:35:12.689+0900 I CONTROL  [initandlisten] modules: none
2019-09-25T16:35:12.689+0900 I CONTROL  [initandlisten] build environment:
2019-09-25T16:35:12.689+0900 I CONTROL  [initandlisten]     distmod: 2008pl
2019-09-25T16:35:12.689+0900 I CONTROL  [initandlisten]     distarch: x86_6
2019-09-25T16:35:12.689+0900 I CONTROL  [initandlisten]     target_arch: x8
```

4 DBServer 구동

http://127.0.0.1:3000/MQTT.html

http://172.30.1.15:3000/MQTT.html



MQTT 모니터링 서비스

- 27(72%)

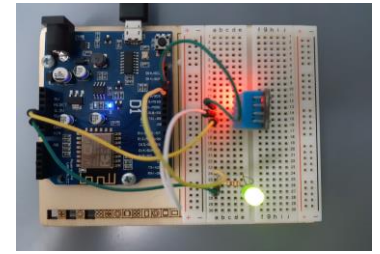
socket 통신 방식(LED제어) 6

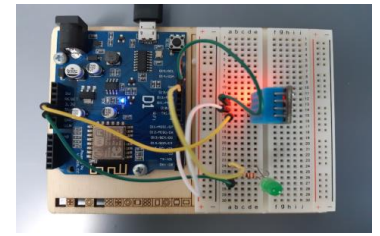
LED_ON LED_OFF

REST full Service 통신 방식(LED제어)

LED_ON LED_OFF

LED STATUS





Arduino(WeMos D1) / MQTT / Node.js / MongoDB / Eclipse IDE

MQTT 프로토콜을 이용한 사물인터넷 통신 프로젝트

수고하셨습니다.