



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
- 브레드보드
- LED 1개
- 저항 1개(220 Ohm)
- 점퍼케이블
- 전원 케이블(USB)

### 디렉토리 만들기

C:\WMQTTProject

source

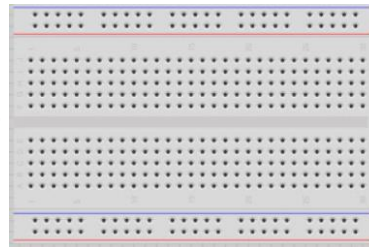
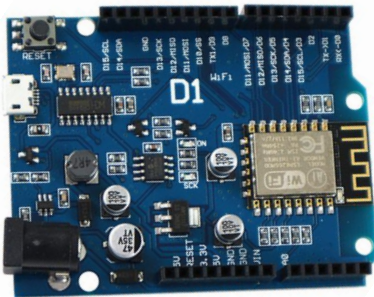
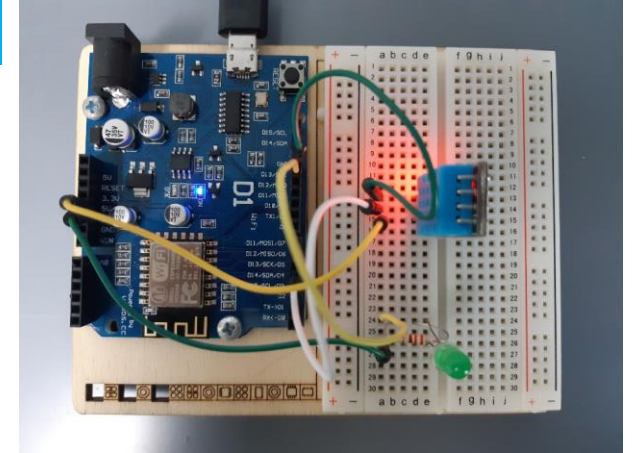
SW

### 설치 프로그램

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

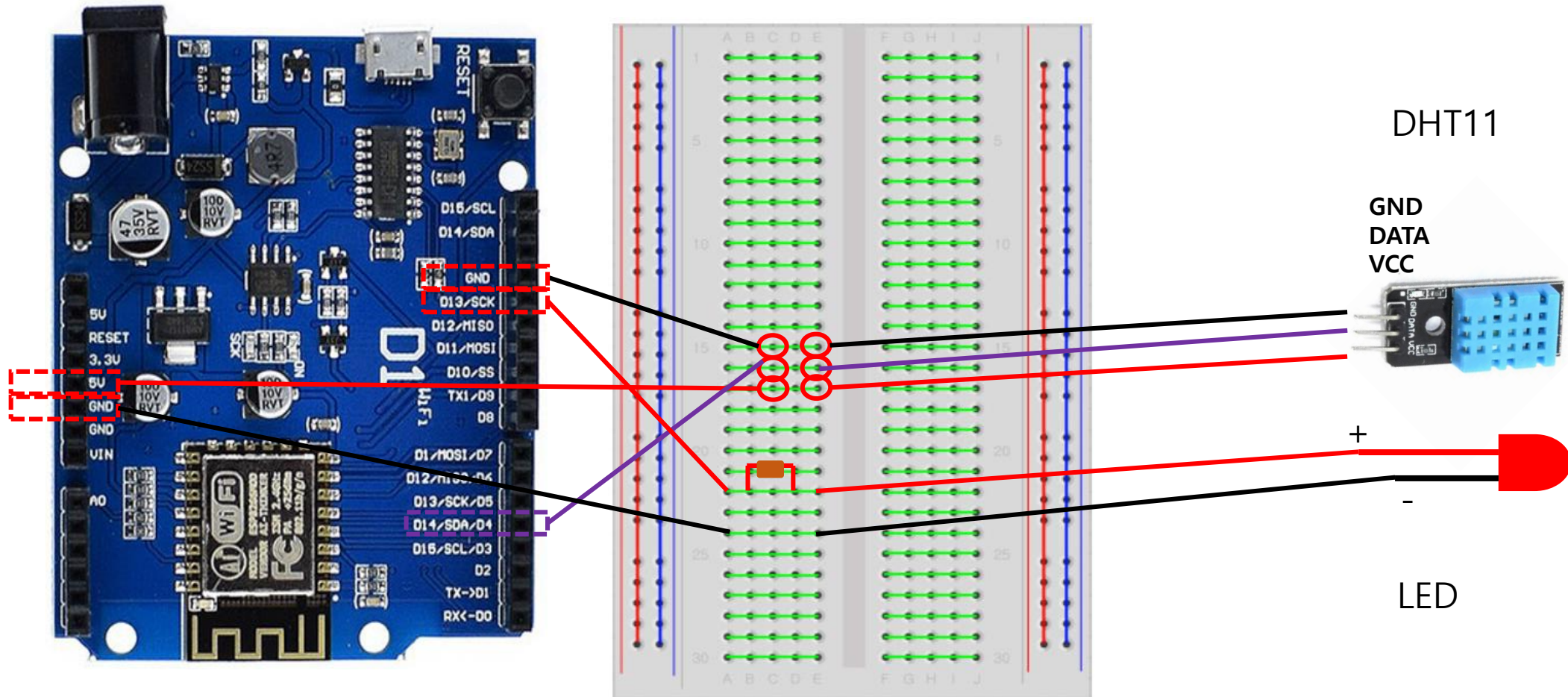
### 통신연결 주소

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



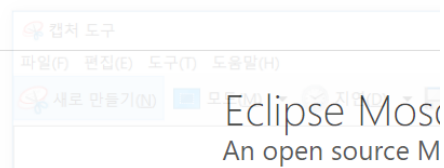
인터넷에서 조회 후 구매

## 2. 회로도 구성



## 1. Eclipse Mosquitto™ An open source MQTT broker Download

→ <https://mosquitto.org/>



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

The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe 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 popular

Mosquitto is part of the Eclipse Foundation, is an [iot.eclipse.org](https://www.eclipse.org/iot) project and is sponsored by [ced.com](https://www.ced.com).

### 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](https://test.mosquitto.org) where you can test your clients in a variety of



## 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/>

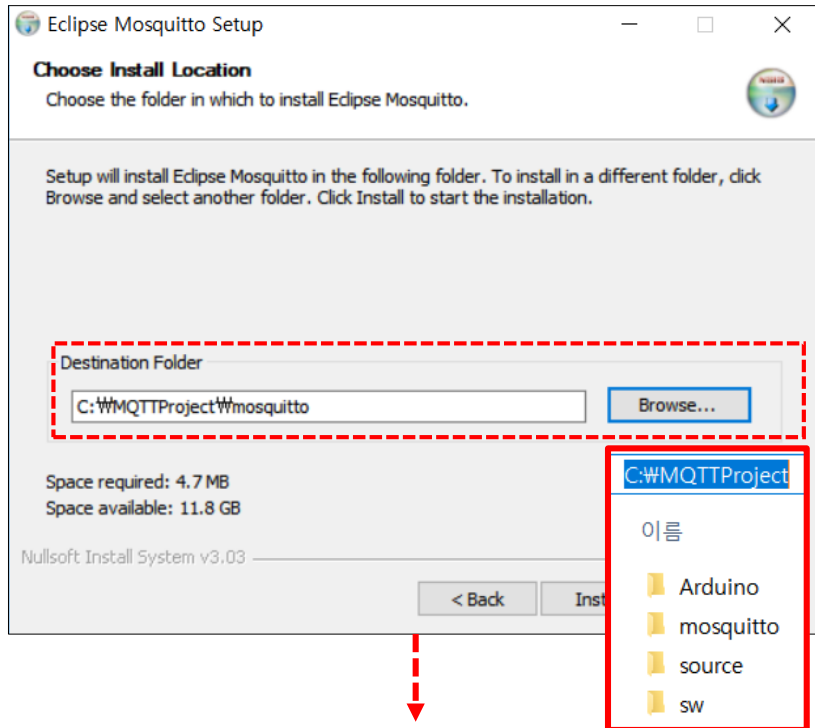
## Binary Installation

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

## 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.
```

## mosquitto 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

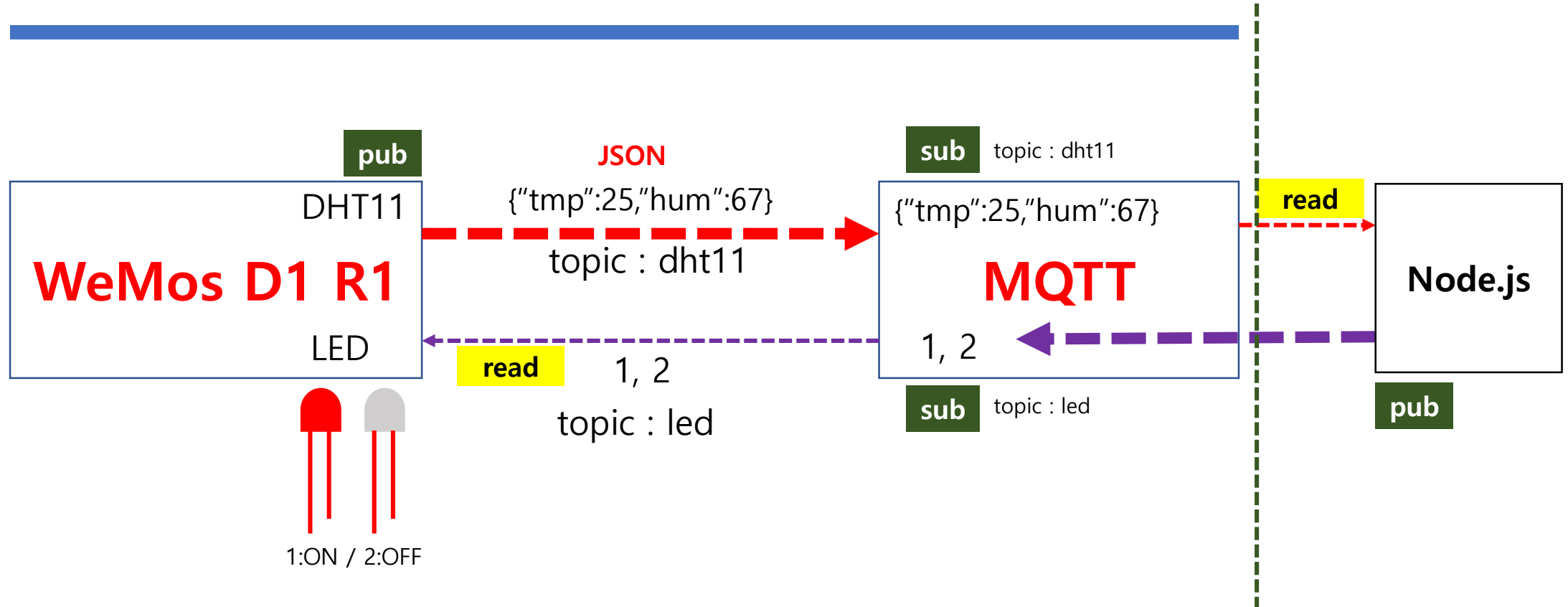
```
C:\>관리자: 명령 프롬프트 - mosquitto -v
C:\MQTTProject>mosquitto -v
1569209814: mosquitto version 1.6.6 starting
1569209814: Using default config.
1569209814: Opening ipv6 listen socket on port 1883.
1569209814: Opening ipv4 listen socket on port 1883.
1569210272: New connection from ::1 on port 1883.
1569210272: New client connected from ::1 as mosq/cl4VD7MdHjrh05rT1s (p2, c1, k60).
1569210272: No will message specified.
1569210272: Sending CONNACK to mosq/cl4VD7MdHjrh05rT1s (0, 0)
1569210272: Received SUBSCRIBE from mosq/cl4VD7MdHjrh05rT1s
1569210272:      iot (QoS 0)
1569210272: mosq/cl4VD7MdHjrh05rT1s 0 iot
1569210272: Sending SUBACK to mosq/cl4VD7MdHjrh05rT1s
1569210332: Received PINGREQ from mosq/cl4VD7MdHjrh05rT1s
1569210332: Sending PINGRESP to mosq/cl4VD7MdHjrh05rT1s
1569210392: Received PINGREQ from mosq/cl4VD7MdHjrh05rT1s

C:\>관리자: 명령 프롬프트 - mosquitto_sub -t iot -p 1883
C:\>cd MQTTProject>mosquitto
C:\MQTTProject>mosquitto_sub -t iot -p 1883
hello
{"tmp":25,"hum":70}
{"tmp":25,"hum":70}

C:\>관리자: 명령 프롬프트
C:\>cd MQTTProject>mosquitto
C:\MQTTProject>mosquitto_pub -t iot -m "hello"
C:\MQTTProject>mosquitto_pub -t iot -m '{"tmp":25,"hum":70}'
C:\MQTTProject>mosquitto_pub -t iot -m '{"tmp":25,"hum":70}'
C:\MQTTProject>
```

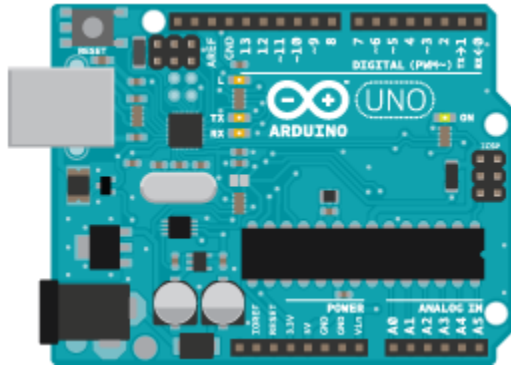
Arduino(WeMos)쪽에서 WiFi로 DHT11 Sensor 데이터(온도/습도) 발행

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



## 2. Download the Arduino IDE

→ <http://arduino.cc>



HOME STORE **SOFTWARE** EDU RESOURCES COMMUNITY HELP

DOWNLOADS

# Download the Arduino IDE

**ARDUINO 1.8.10**  
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, macOS X, and Linux. The environment is based on Java and based on Processing and other open-source libraries. It can be used with any Arduino board. See the [Getting Started](#) page for Installation

**Windows** Installer, for Windows XP and up  
**Windows** ZIP file for non admin install

**Windows app** Requires Win 8.1 or 10  
 **Get**

**Mac OS X** 10.8 Mountain Lion or newer

**Linux** 32 bits  
**Linux** 64 bits  
**Linux** ARM 32 bits  
**Linux** ARM 64 bits

[Release Notes](#)  
[Source Code](#)  
[Checksums \(sha512\)](#)

[JUST DOWNLOAD](#) [CONTRIBUTE & DOWNLOAD](#)

### Contribute to the Arduino Software

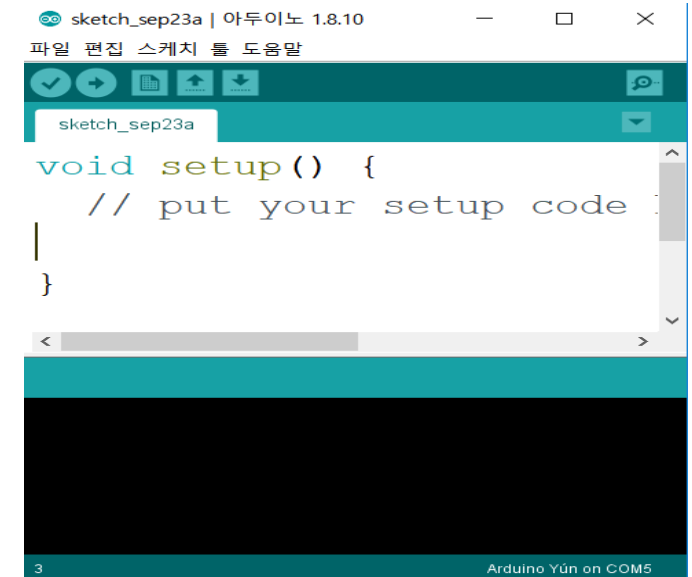
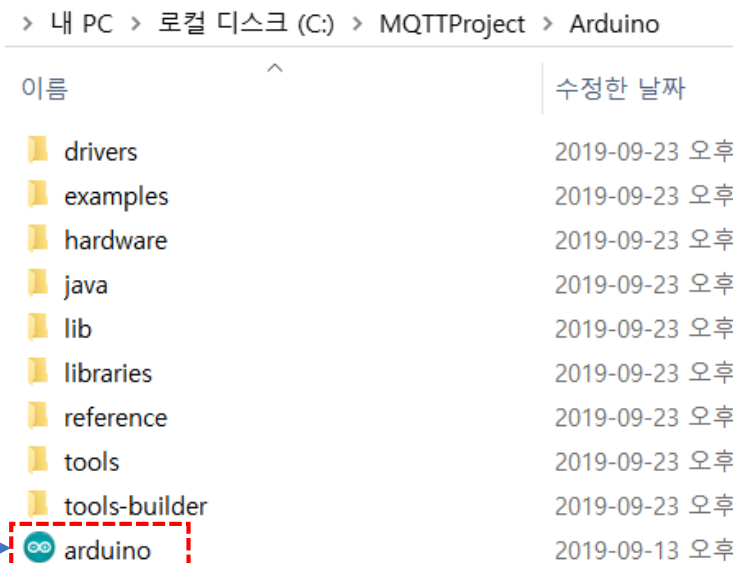
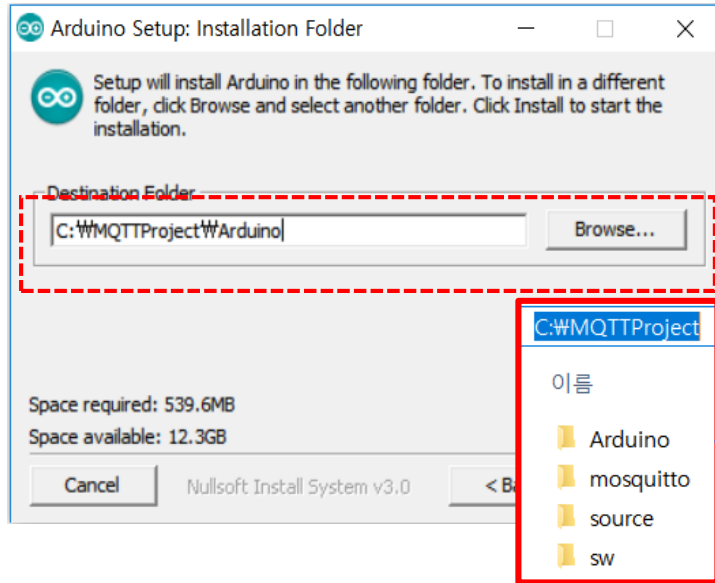
Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). [Learn more on how your contribution will be used.](#)

SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED **35,483,068** TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

**\$3** **\$5** **\$10** **\$25** **\$50** **OTHER**

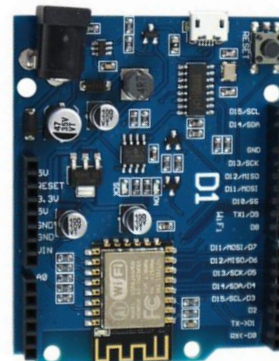


### 3. Arduino 설치



#### 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)

5V	D15	SCL	GPIO5
RST	D14	SDA	GPIO4
3V3			
5V	D13	SCK	GPIO14
GND	D12	MISO	GPIO12
GND	D11	MOSI	GPIO13
VIN	D10	SS	GPIO15
	D9	TX1	GPIO2
	D8		GPIO0
	D7	MOSI	GPIO13
	D6	MISO	GPIO12
	D5	SCK	GPIO14
	D4	SDA	GPIO4
	D3	SLC	GPIO5
	D2		GPIO16
	D1	Tx	GPIO1
	D0	Rx	GPIO3

Rx0\*  
Tx0\*  
10K Pull down  
10K Pull up  
10K Pull up  
Built in led

## 5. 아두이노 WeMos 환경 셋팅

환경설정 참고 SITE

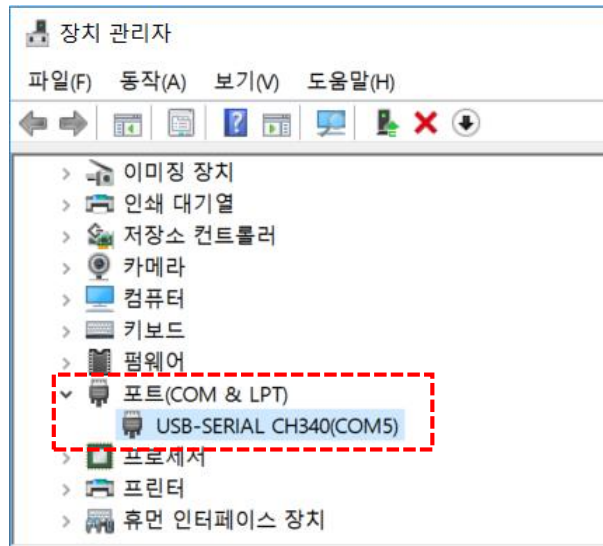
<http://www.makewith.co/page/project/1004/story/2478/>

### 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.

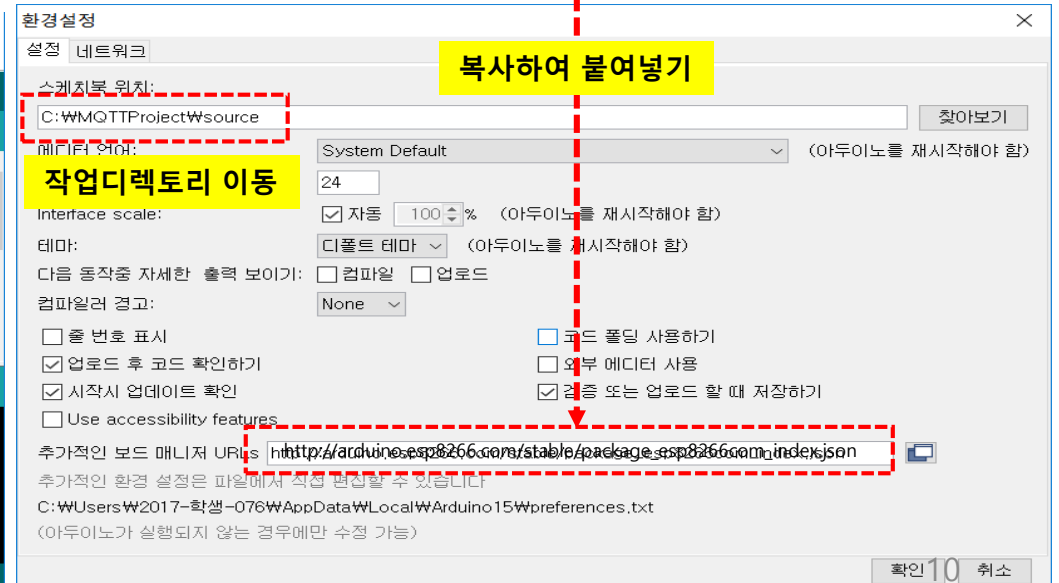
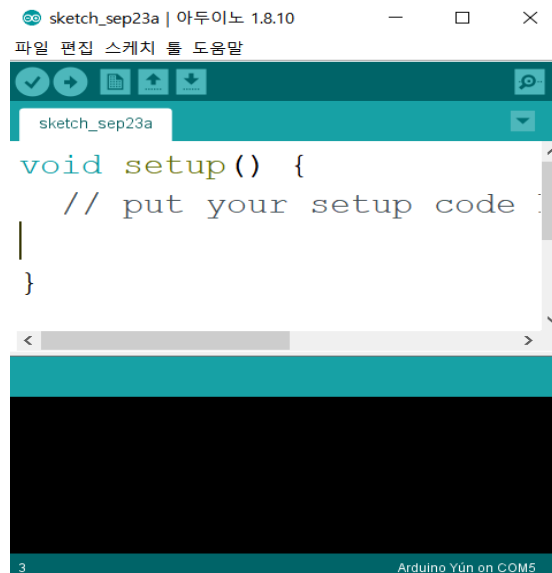


### STEP 3. 아두이노 IDE에 WeMos 보드 라이브러리 설치하기

→아래 URL을 클릭하지 말고 복사합니다.

[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

#### ① 파일->환경설정

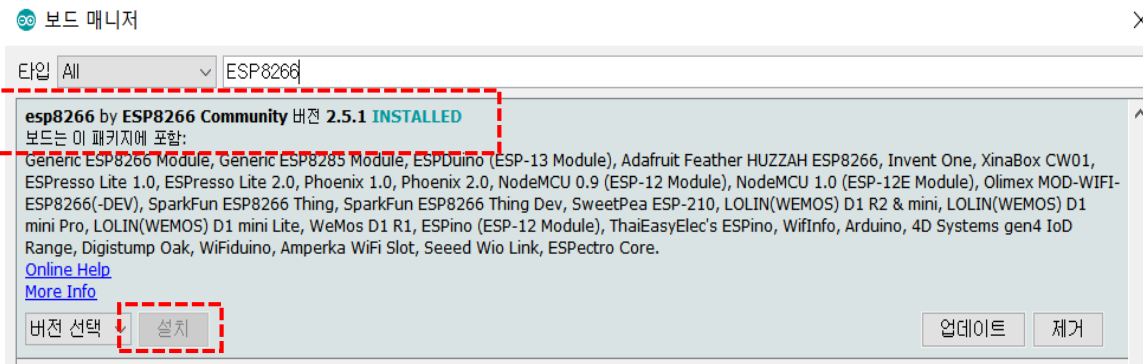


## STEP 4. Board Manager로 WeMos보드 설치하기

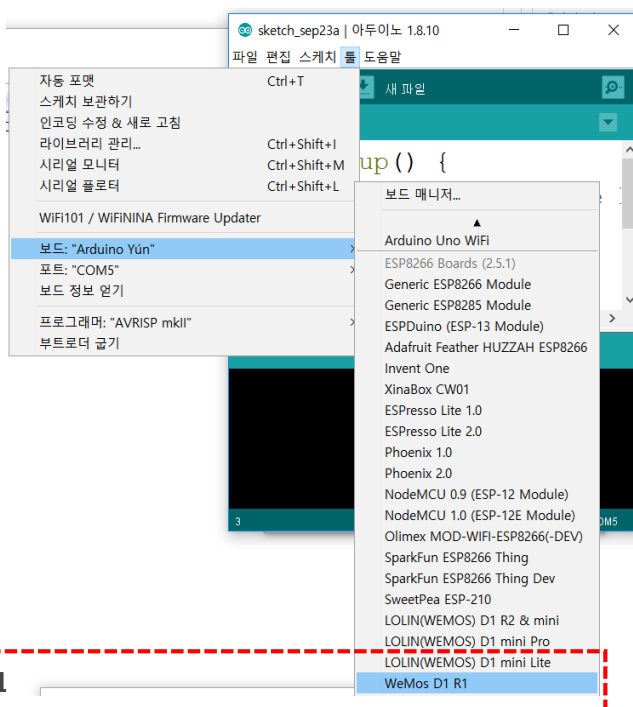
## ② 툴-&gt; 보드-&gt; 보드 매니저...

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

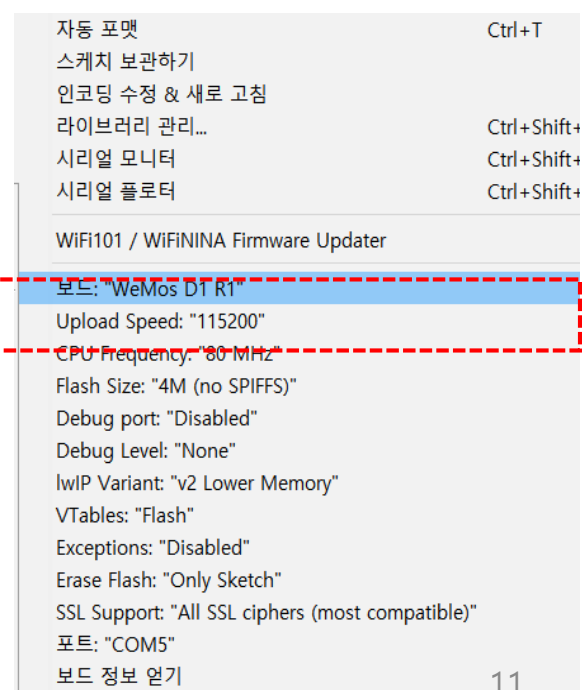
→ 설치 후 아두이노 IDE를 껐다 켜주세요.



## STEP 5. WeMos 보드 선택하기 &amp; speed 설정하기



## ④ 툴-&gt; 보드-&gt; Upload Speed "115200" 선택



## ③ 툴-&gt; 보드-&gt; WeMos D1

## 7. PubSubClient 라이브러리 설치

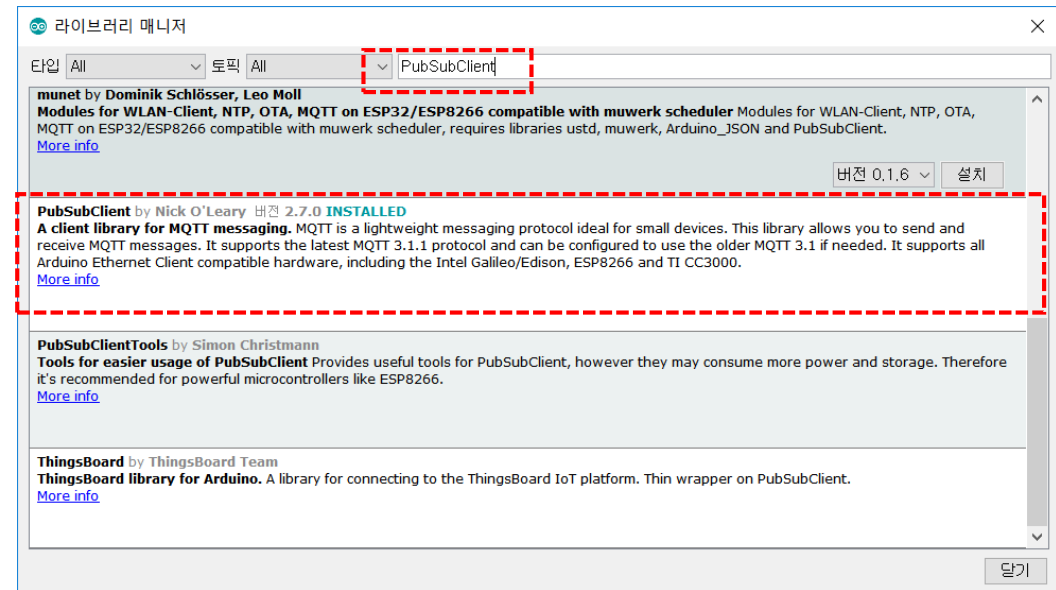
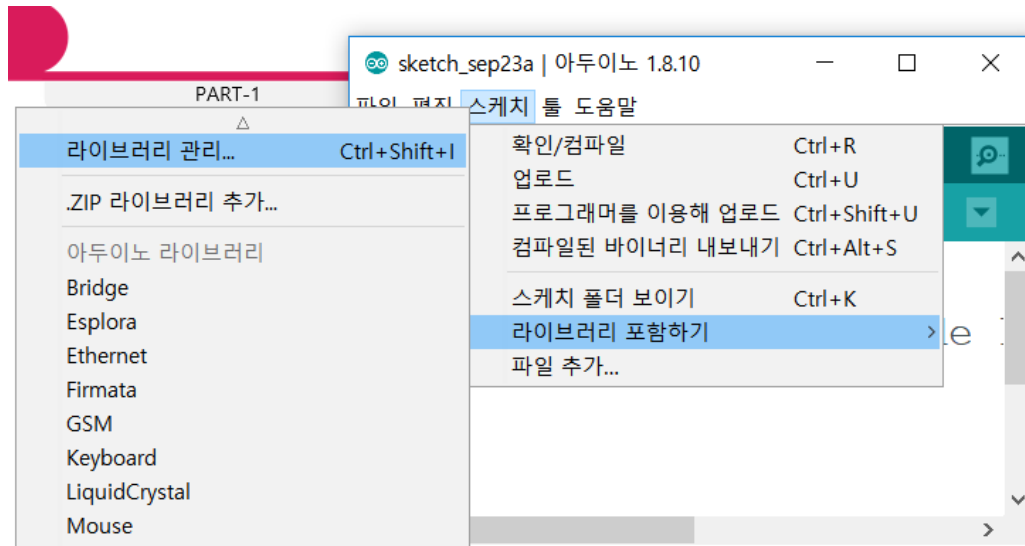
ESP8266 모듈에 MQTT 기능을 구현하기 위해서는 해당 라이브러리가 필요합니다. 아두이노 개발환경에서 사용할 수 있도록 개발된 라이브러리가 여러 종류가 있는데 여기서는 **PubSubClient** 라이브러리를 사용하도록 하겠습니다. 일단 아두이노 개발환경을 실행하고 아래 순서대로 라이브러리를 설치하면 됩니다.

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

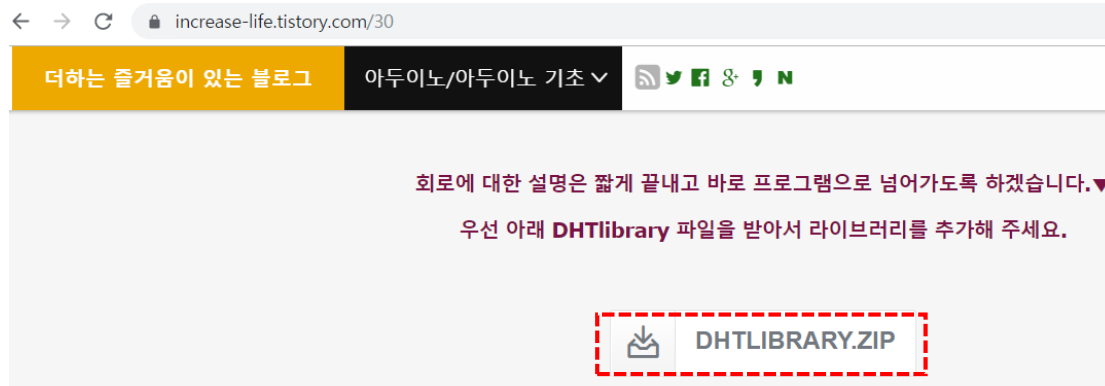
C:\MQTTProject\source\libraries

이름

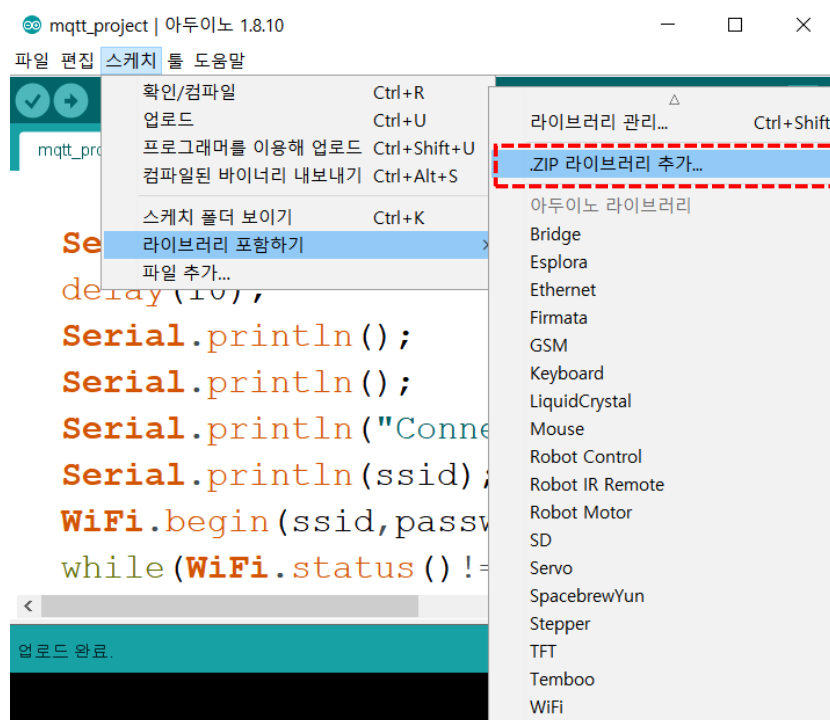
DHTlibrary  
PubSubClient  
readme



## 8. DHT11 라이브러리 설치



다운로드 SITE



C:\MQTTProject\sw

이름

arduino-1.8.10-windows  
DHTlibrary  
mosquitto-1.6.6-install-windows-x64

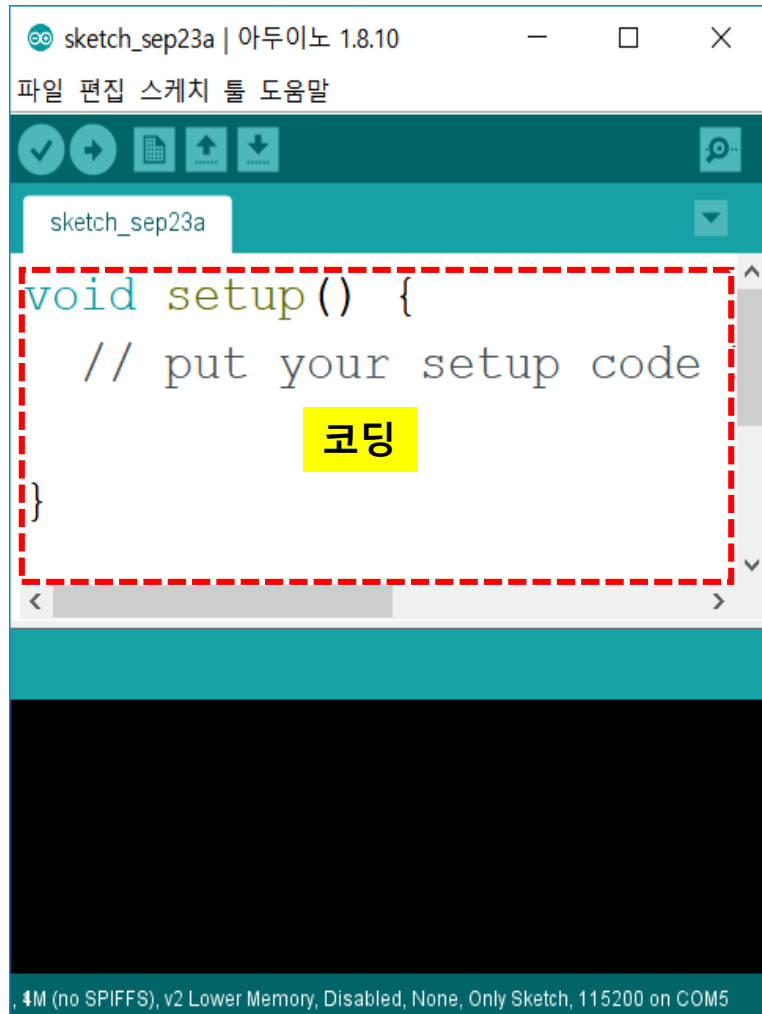
C:\MQTTProject\source\libraries

이름

DHTlibrary  
PubSubClient  
readme

.zip 라이브러리 추가

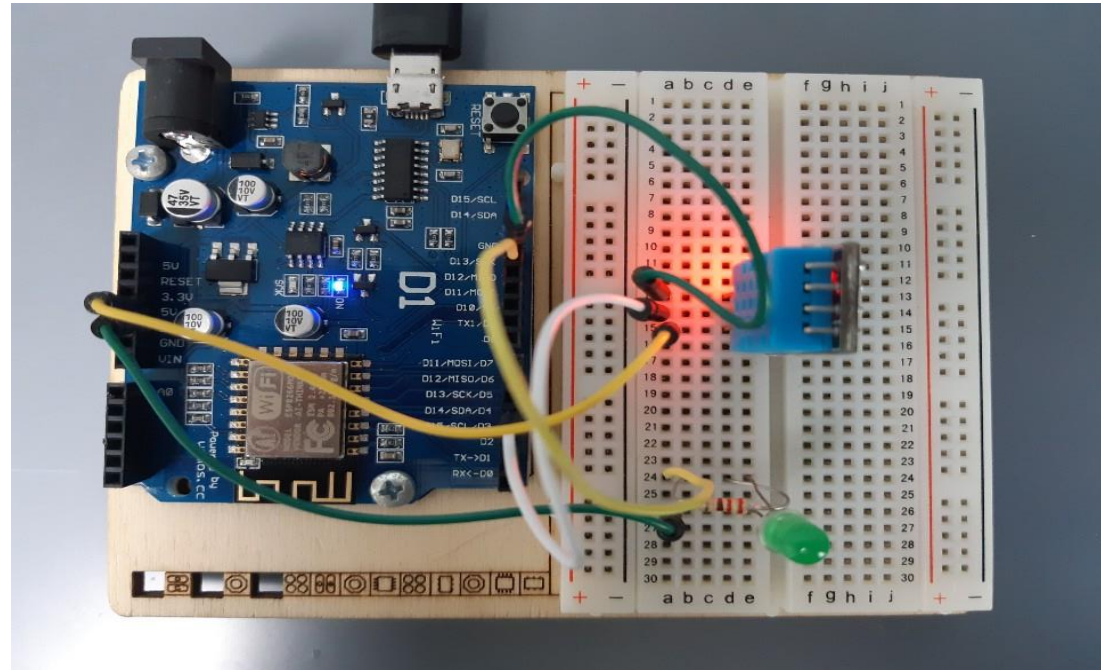
## 9. 소스코딩(소스저장)



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"+W0(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");
    }
}
```

```
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
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,{W"tmpW":%s,W"humW":%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":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

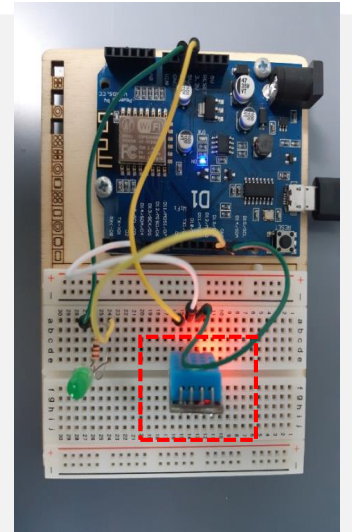
```
1569239495: Received PUBLISH from Arduino (d0, q0, r0, m0, 'dht11', ... (25 bytes))
1569239495: Sending PUBLISH to mosq/acT0za1WP16Ind5fc3 (d0, q0, r0, m0, 'dht11', ... (25 bytes))
1569239498: Received PUBLISH from Arduino (d0, q0, r0, m0, 'dht11', ... (25 bytes))
1569239498: Sending PUBLISH to mosq/acT0za1WP16Ind5fc3 (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/acT0za1WP16Ind5fc3 (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



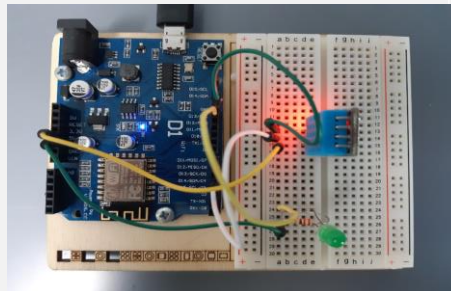
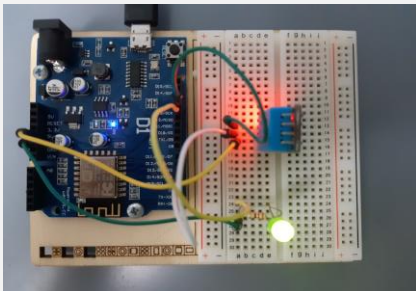
관리자: 명령 프롬프트 - mosquitto\_sub -t led -p 1883

C:\MQTTProject\mosquitto>mosquitto\_sub -t led -p 1883

1  
2

1->ON

2->OFF



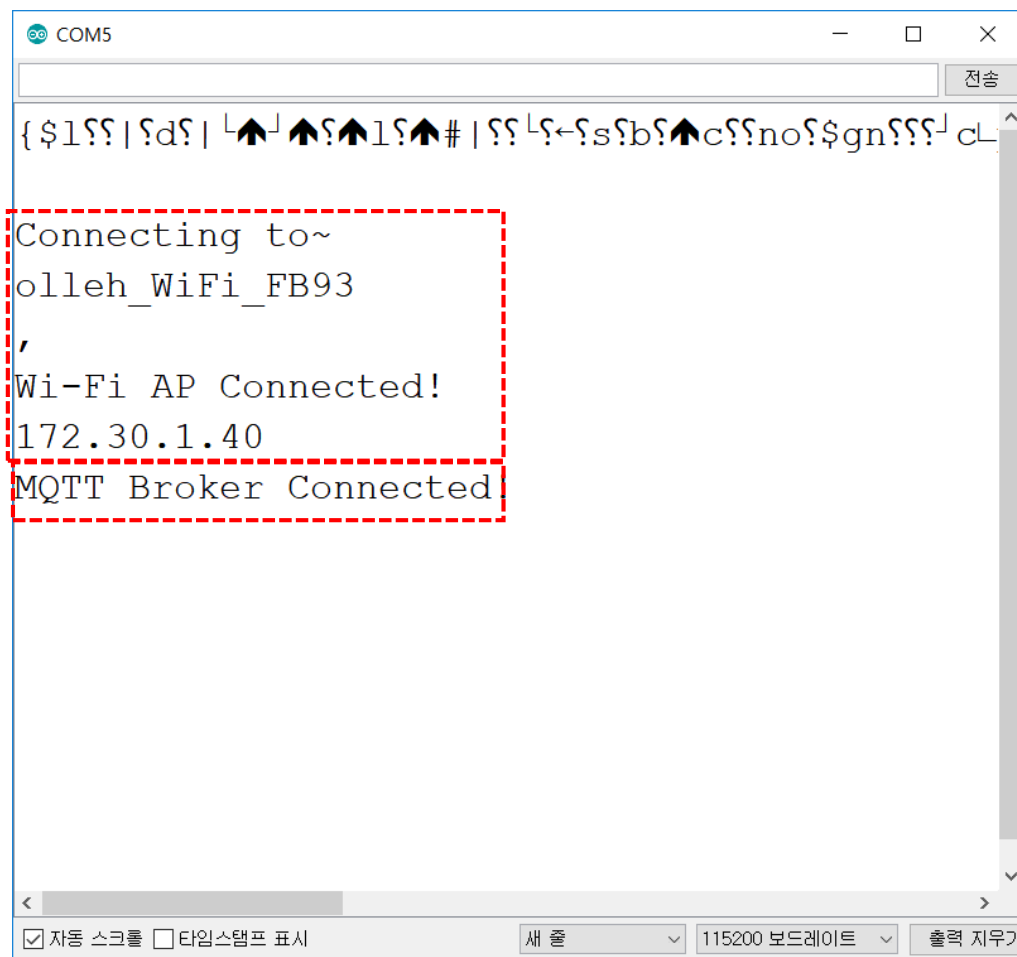
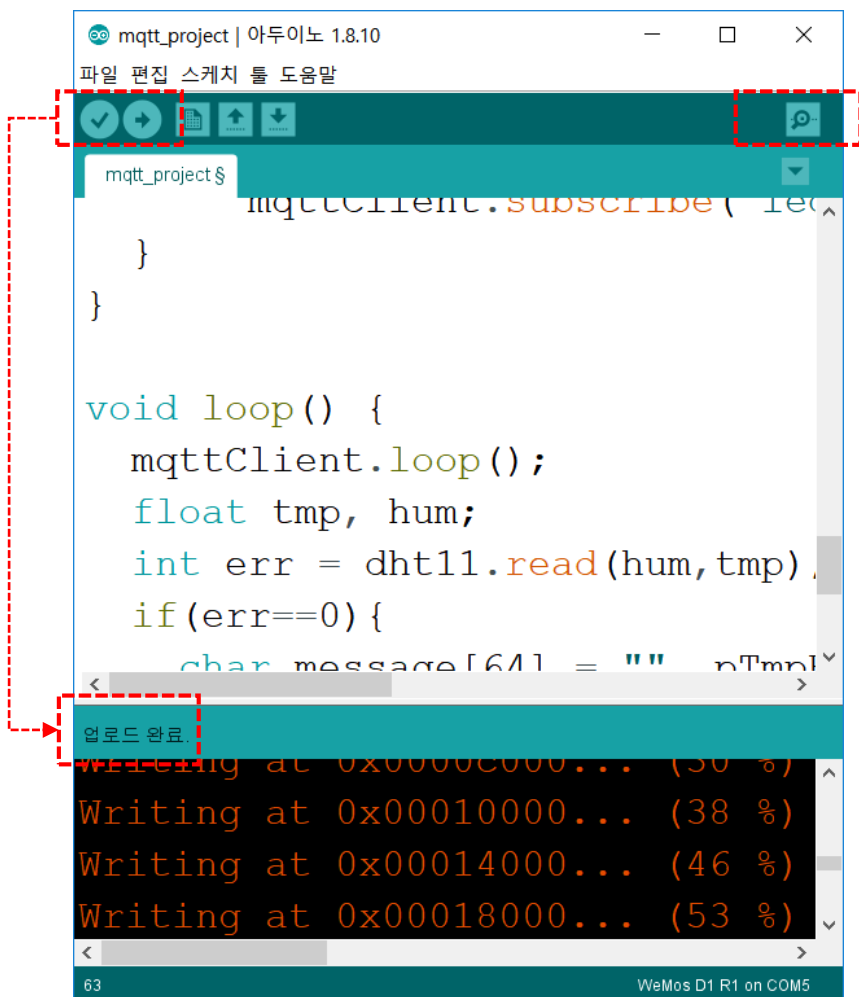
관리자: 명령 프롬프트

C:\MQTTProject\mosquitto>mosquitto\_pub -t led -m 1

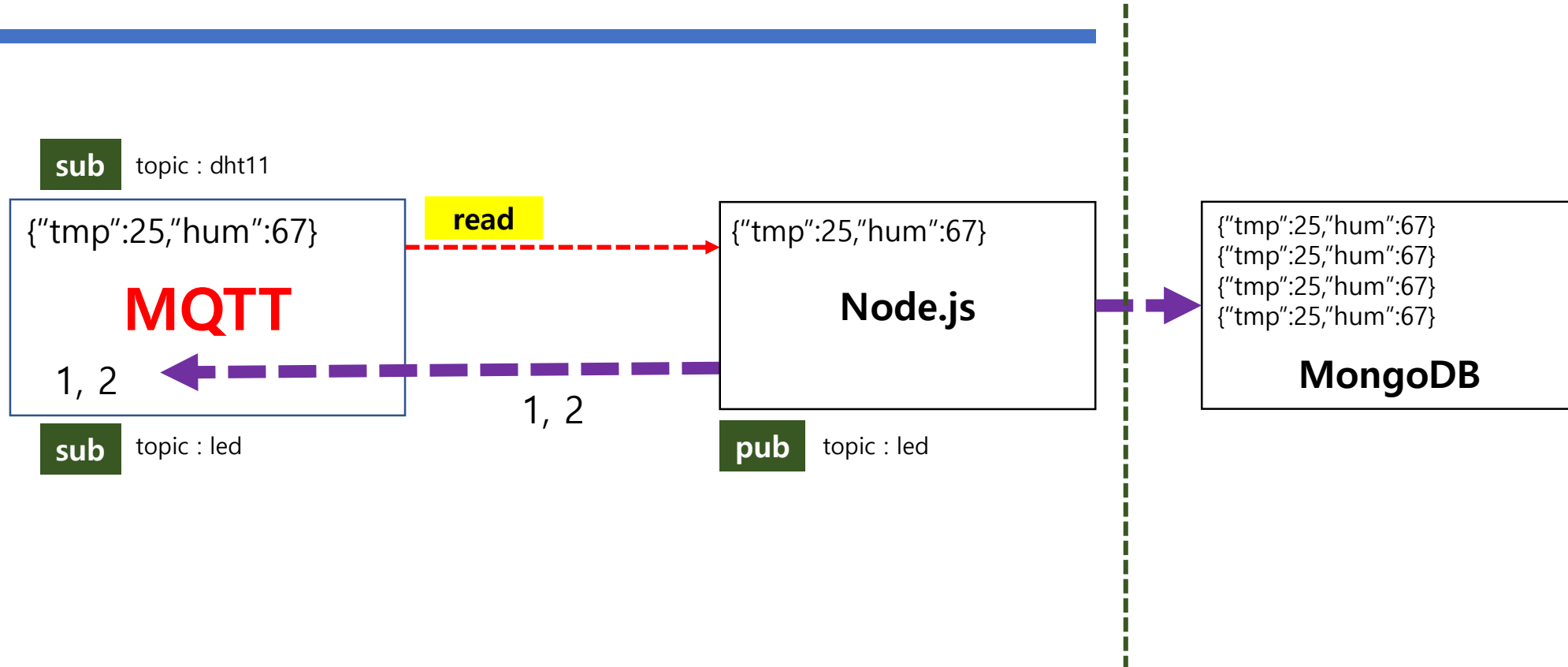
C:\MQTTProject\mosquitto>mosquitto\_pub -t led -m 2

C:\MQTTProject\mosquitto>

## 9. 컴파일, 업로드, 구동 TEST



## 1. 구현내용



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

nodejs.org/ko/download/

### 다운로드

최신 LTS 버전: 10.16.3 (includes npm 6.9.0)

플랫폼에 맞게 미리 빌드된 Node.js 인스톨러나 소스코드를 다운받아서 바로 개발을 시작하세요.

**LTS**  
대다수 사용자에게 추천

**현재 버전**  
최신 기능

Windows Installer  
node-v10.16.3-x64.msi

macOS Installer  
node-v10.16.3.pkg

Source Code  
node-v10.16.3.tar.gz

	32-bit	64-bit
Windows Installer (.msi)		
Windows Binary (.zip)		

C:\MQTTProject\sw

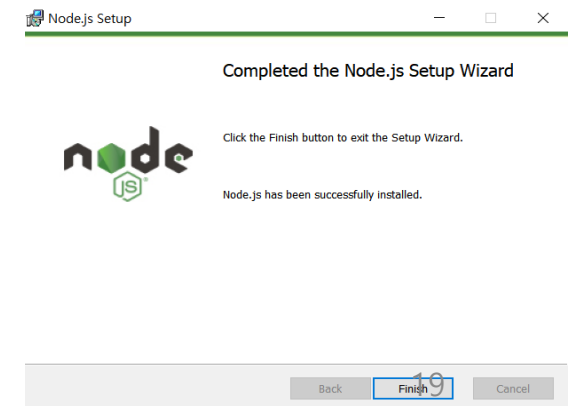
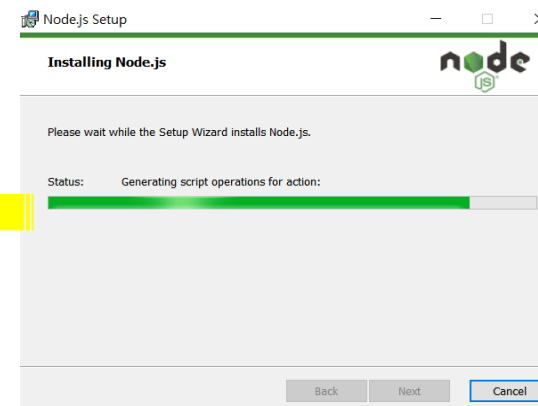
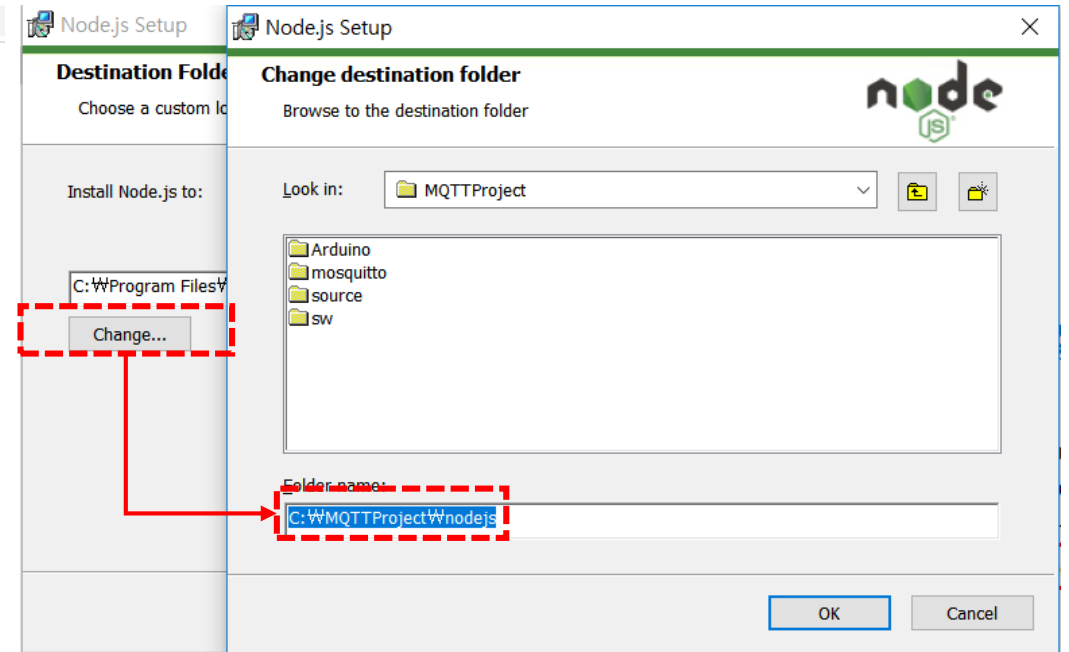
이름

- arduino-1.8.10-windows
- DHTlibrary
- mosquitto-1.6.6-install-windows-x64
- node-v10.16.3-x64

C:\MQTTProject

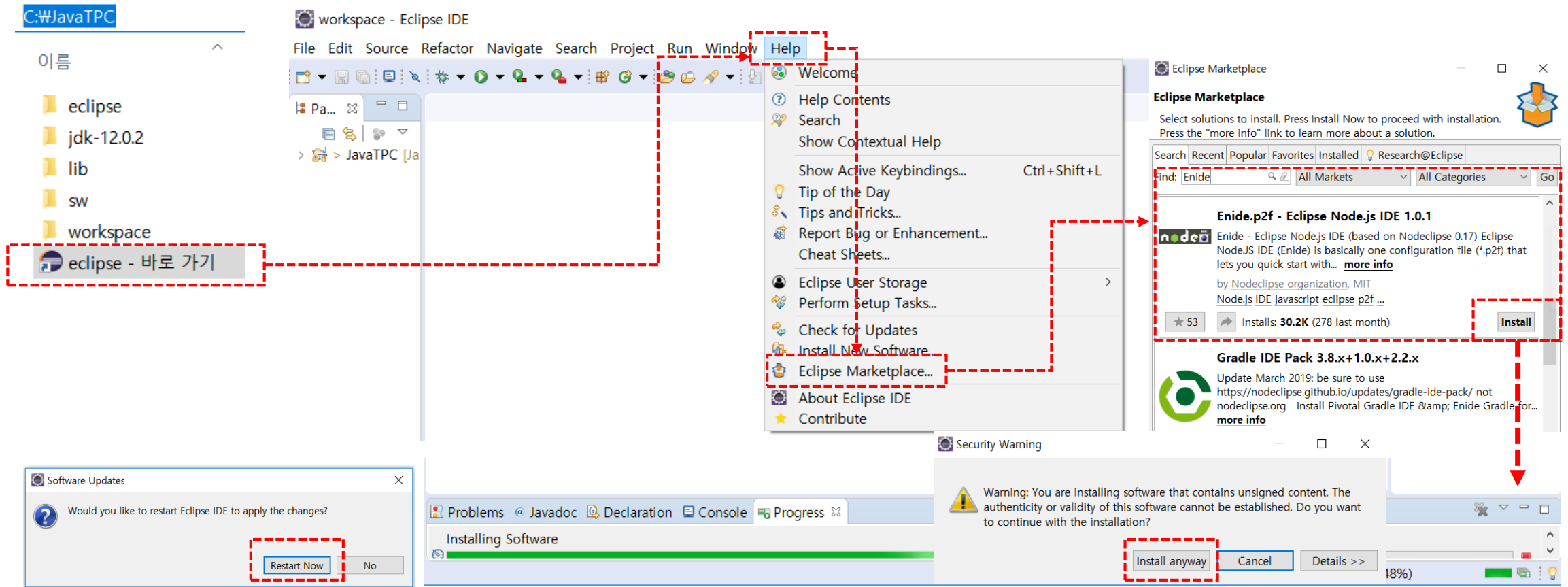
이름

- Arduino
- mosquitto
- nodejs
- source
- sw



### 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

workspace - Eclipse IDE

File Edit Navigate Search Project Run Emmet Window Help

New Window  
Editor  
Appearance  
Show View  
Perspective  
Navigation  
Preferences

Preferences

type filter text

Nodeclipse

Nodeclipse:1.0.2.201509250223 Eclipse:4.12.0.v20190605-1800 Java:12.0.2 OS:win32,x86\_64

Node.js, Express, CoffeeScript, TypeScript settings  
Visit [www.nodeclipse.org](http://www.nodeclipse.org) for news and history. [GitHub](#)

☒ enable Nodeclipse Console  
☒ find node on PATH. Otherwise use Node.js instance in location below

Node.js path: C:\MQTTProject\nodejs\node.exe [Browse...](#)

Node options (node -h):

Node Application arguments:

☒ allow many Node.js instances running  
☐ pass all environment variables of Eclipse to launched Node.js app  
☒ add Tern nature to newly created projects

Node sources directory path: [Browse...](#)

☒ use Node.js base module definitions (changed after restart)  
☒ use Orion IndexFiles (changed after restart)  
☒ use completion.json (changed after restart)

Alternative completions.json path: [Browse...](#)

☐ Node debug without -brk (disable interruption of Node.js app start)

Node debug port: 5858

Express path: C:\JavaTPC\eclipse\configuration\org.eclipse.osgi\989\0\cp\express\bin\express [Browse...](#)

Selected Express version: 3.2.6

Coffee path: C:\JavaTPC\eclipse\configuration\org.eclipse.osgi\988\0\cp\coffee-script\bin\coffee [Browse...](#)

Coffee compile options: --watch

Coffee output folder #76

TypeScript compiler path: [Browse...](#)

TypeScript compiler options:

[Restore Defaults](#) [Apply](#)

[Apply and Close](#) [Cancel](#)

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

선택 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

## 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

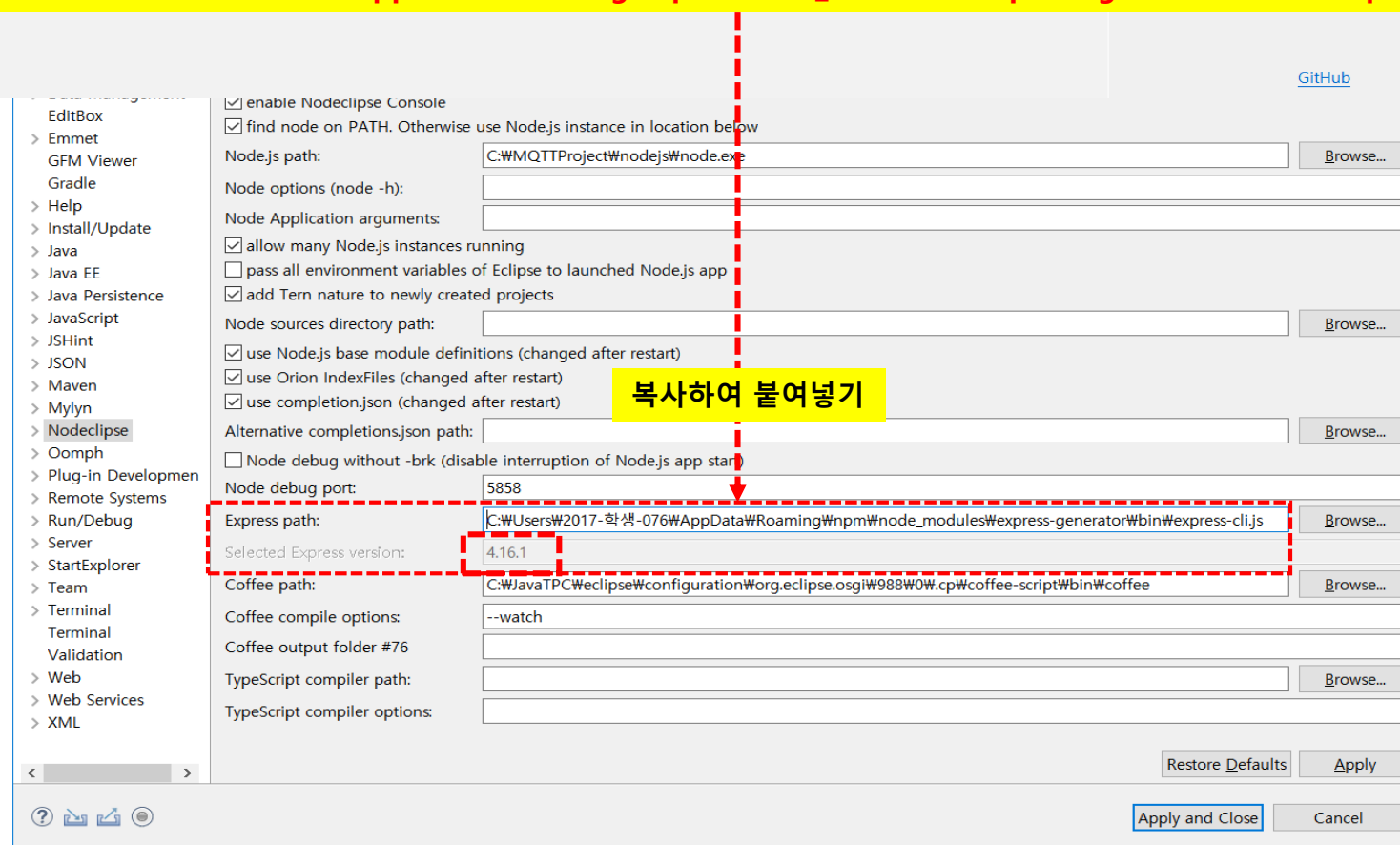
C:\Users\2017-학생-076\AppData\Roaming\npm\express -> C:\Users\2017-학생-076\AppData\Roaming\npm\node\_modules\express-g

enerator\bin\express-cli.js

+ express-generator@4.16.1  
updated 3 packages in 4.362s

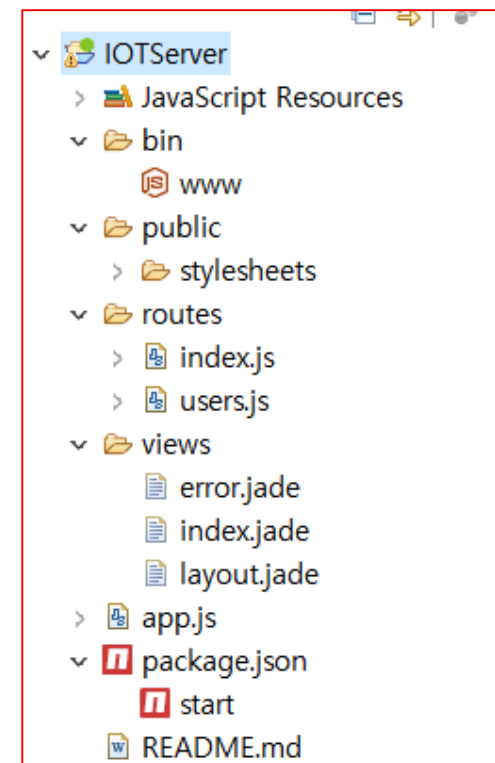
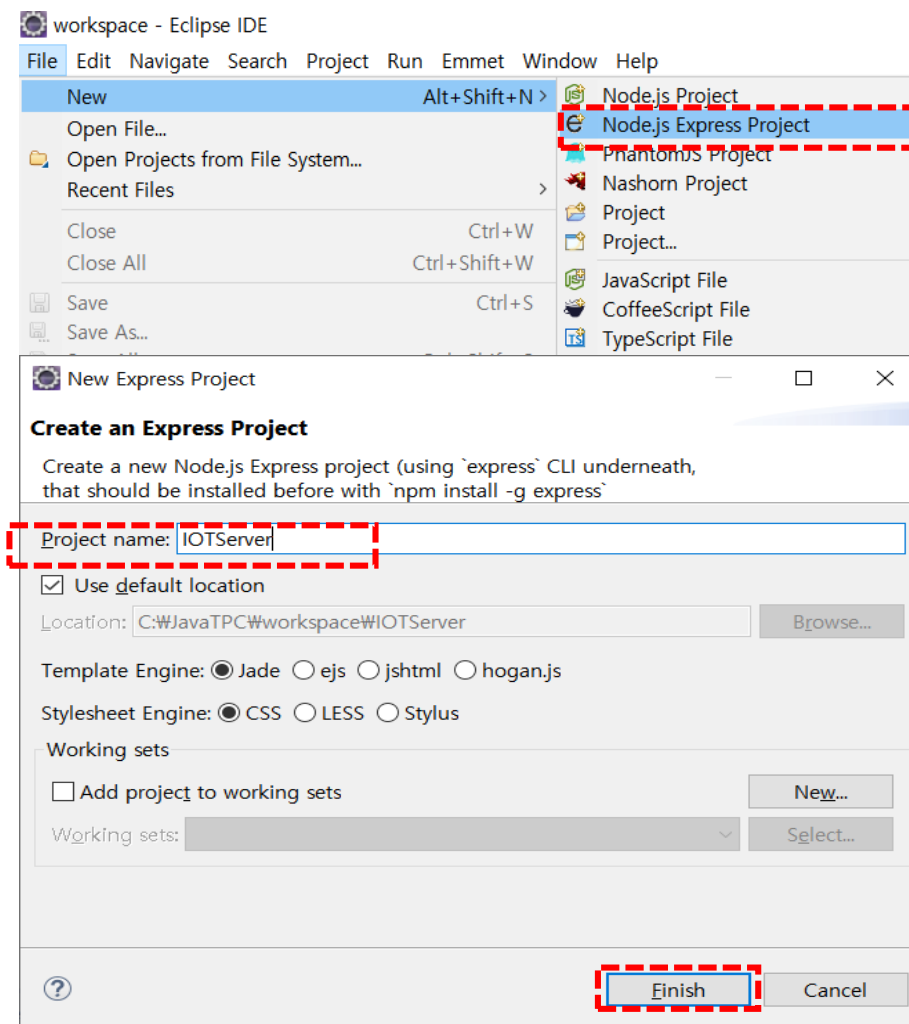
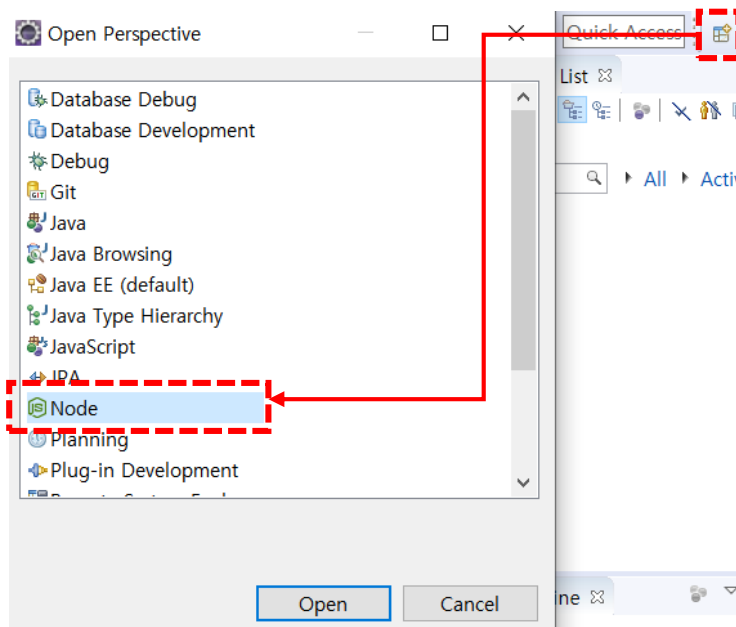
C:\Users\2017-학생-076>

C:\Users\2017-학생-076\AppData\Roaming\npm\node\_modules\express-generator\bin\express-cli.js



## 5. Node.js IOT Server 프로그래밍

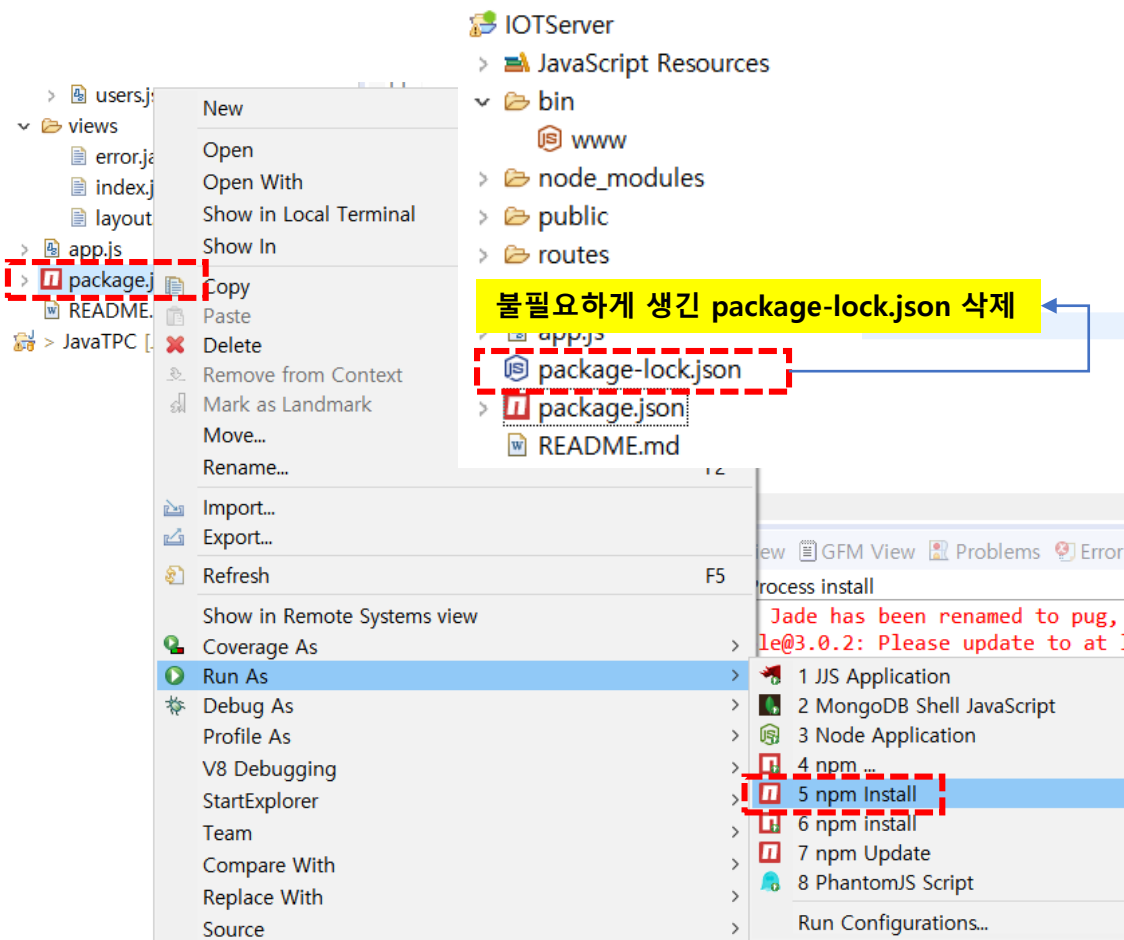
### → Node.js Express Project 만들기(WEB)



## 5. Node.js IOT Server 프로그래밍

### → package 설치하기

```
package.json ✕  
1 {  
2   "name": "iotserver",  
3   "version": "0.0.0",  
4   "private": true,  
5   "scripts": {  
6     "start": "node ./bin/www"  
7   },  
8   "dependencies": {  
9     "cookie-parser": "~1.4.4",  
10    "debug": "~2.6.9",  
11    "express": "~4.16.1",  
12    "http-errors": "~1.6.3",  
13    "jade": "~1.11.0",  
14    "morgan": "~1.9.1",  
15    "mqtt": "^2.14.0",  
16    "mongodb": "^2.2.33",  
17    "socket.io": "^2.0.4"  
18  }  
19 }
```

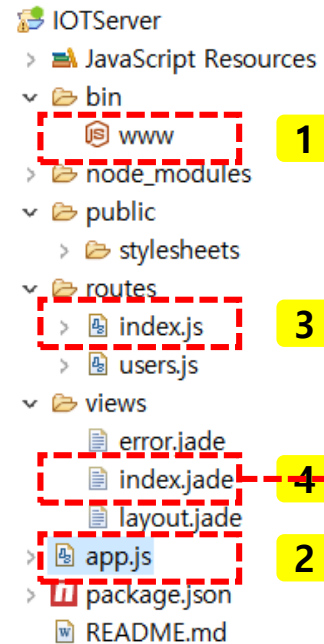


## 5. Node.js IOT Server 프로그래밍

### → Server 구동 및 TEST

The screenshot shows the VS Code interface with the IOTServer project. The file explorer on the left shows the project structure: IOTServer > JavaScript Resources > bin > www. A context menu is open over the 'www' folder, with 'Run As' highlighted. The 'Run As' menu item is circled in red. Below the menu, the console output shows the following logs:

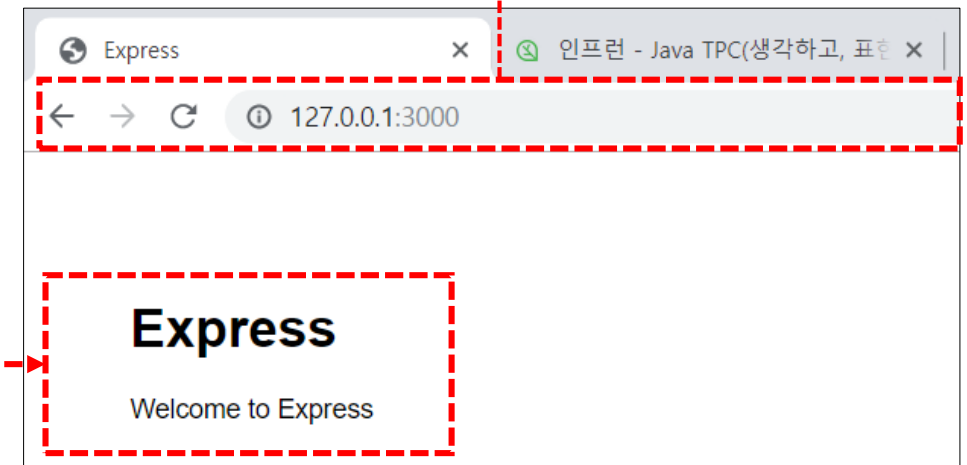
```
IOTServer-bin-www [Node Application] Node.js Process
GET / 200 3009.025 ms - 170
GET /stylesheets/style.css 200 32.279 ms - 111
GET /favicon.ico 404 44.453 ms - 1122
```



3000port 가 개방된다(즉 Web Server 실행된다)

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

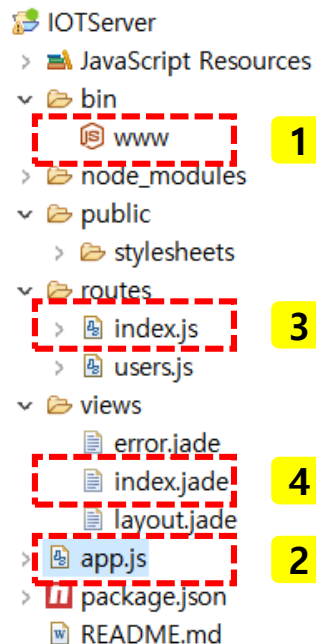
```
app.use('/', indexRouter);
app.use('/users', usersRouter);
```



## 5. Node.js IOT Server 프로그래밍

### → Express Project 구동 절차

#### Web Server 구동(3000 PORT)

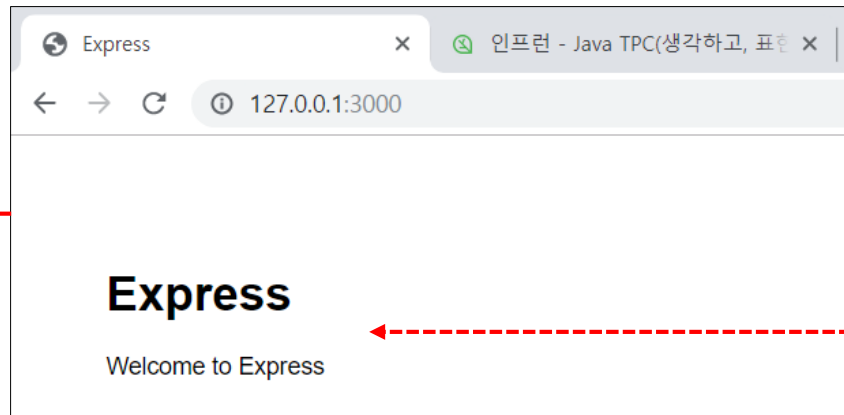


```
var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

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");
```

```
});
```

connect event 발생

dht11 구독자 등록

message event 발생

```
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));
```

```
    });
```

mongodb에 저장하는 부분

```
});
```

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.
```

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}
```

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 }
```

## 1. MongoDB 설치 : <https://mongodb.com>

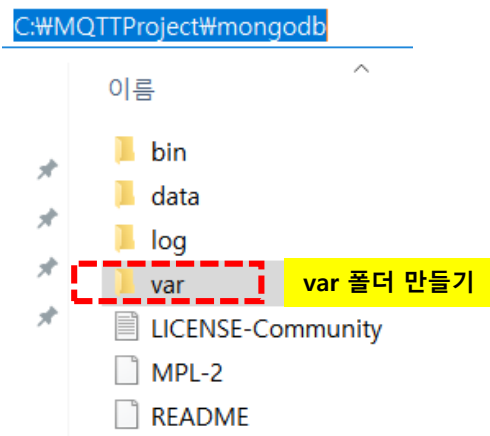
The image shows the MongoDB download center and the installation process. The top part is a screenshot of the MongoDB website, where the 'Try Free' button is highlighted with a red dashed box. Below the website, there are four screenshots of the installation process:

- MongoDB Community Server Selection:** The 'MongoDB Community Server' is selected, and the 'Download' button is highlighted with a red dashed box. The version is 4.0.12 (previous release) and the OS is Windows 64-bit x64.
- Service Configuration:** The 'Service Configuration' window shows the 'Install MongoDB as a Service' option selected. The 'Service Name' is 'MongoDB', and the 'Data Directory' and 'Log Directory' are both set to 'C:\MQTTProject\mongodb\data'. These fields are highlighted with a red dashed box.
- Change destination folder:** The 'Change destination folder' window shows the 'Look in' field set to 'mongodb' and the 'Folder name' field set to 'C:\MQTTProject\mongodb'. A yellow box highlights the folder name, and a red arrow points to it from the 'Custom' option in the 'Choose Setup Type' window.
- Install MongoDB Compass:** The 'Install MongoDB Compass' window shows the 'Install MongoDB Compass' checkbox checked. A yellow box highlights the checkbox, and a red dashed box highlights the 'Install MongoDB Compass' button.

Additional details from the screenshots include:

- The 'Try Free' button on the MongoDB website.
- The 'MongoDB Enterprise Server' section, which is partially visible.
- The 'Release notes', 'Changelog', 'All version binaries', and 'Installation instructions' links.
- The 'Choose Setup Type' window, where the 'Custom' option is selected.
- The 'MongoDB Compass' window, which includes a link to the MongoDB Compass website.

## 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
```

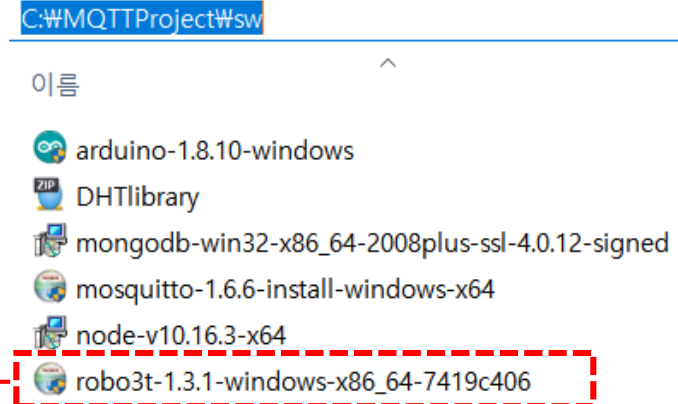
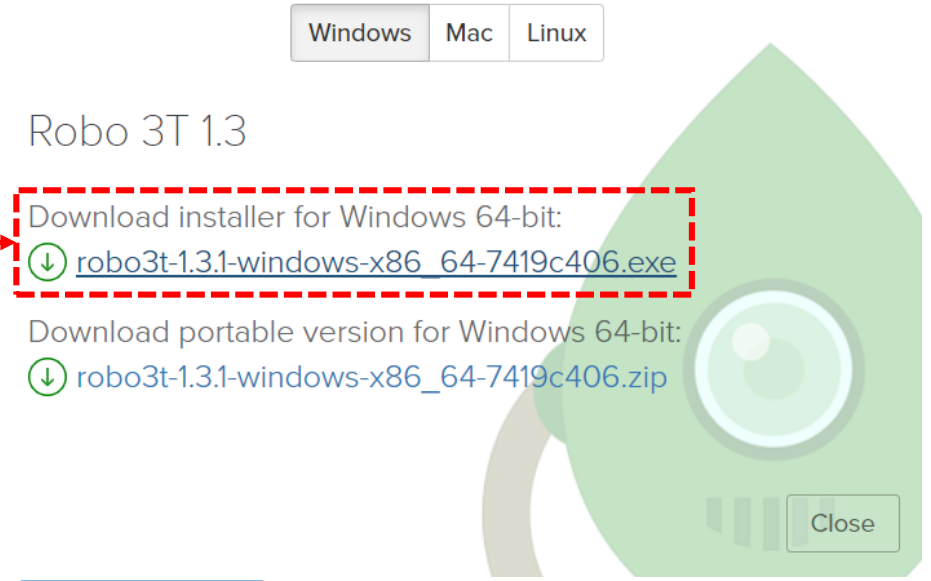
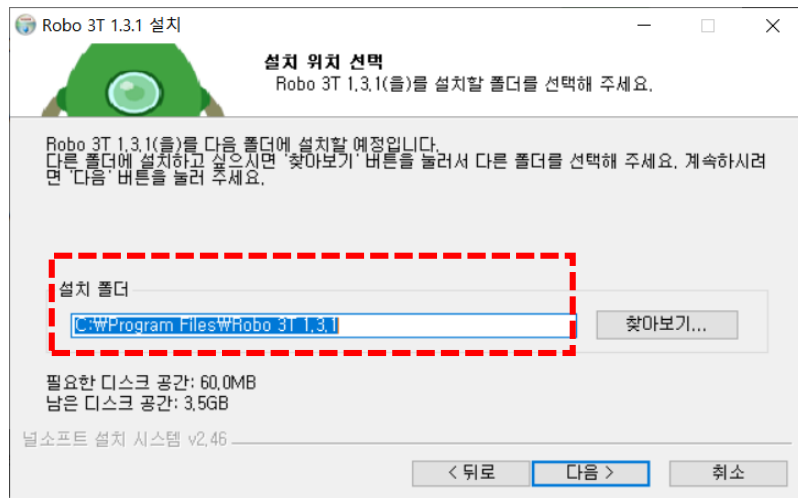
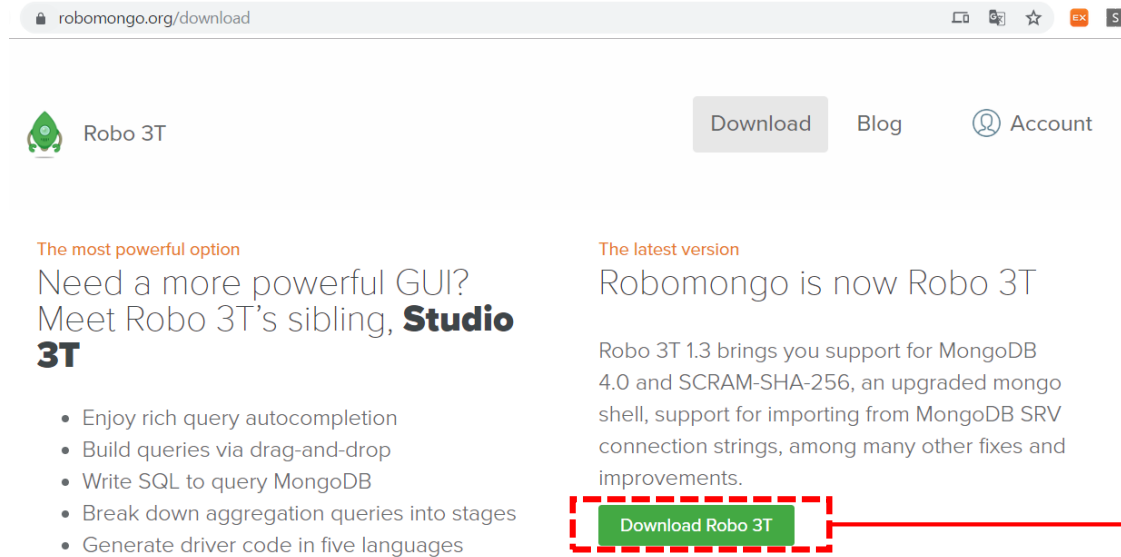
```
MongoDB shell version v4.0.12  
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb  
Implicit session: session { "id" : UUID("b54feb1a-fa86-4d55-bf09-70403af6f8e9") }  
MongoDB server version: 4.0.12
```

```
>
```

### 4. MongoDB 종료하기

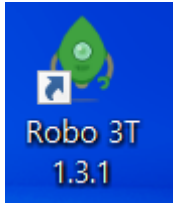
- > **use admin**  
switched to db admin
- > **db.shutdownServer()**

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



## 4. Robo 3T 구동 및 DataBase 만들기

먼저 서버를 구동



```
C:\WMQTTProject\mongodb\bin>mongod --dbpath C:\WMQTTProject\mongodb\var
```

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

The screenshots illustrate the steps to create a database and collection in Robo 3T:

- MongoDB Connections:** The 'Create' button is highlighted in the top-left window.
- Connection Settings:** The 'Name' field is set to 'IOTDataBase' and the 'Address' is 'localhost:27017'. The 'Save' button is highlighted in the bottom-right window.
- MongoDB Connections:** The 'IOTDataBase' connection is listed in the top-right window.
- IOTDataBase (2):** The 'Create Database' option is highlighted in the context menu in the bottom-left window.
- Create Database:** The 'Database Name' is set to 'iot' in the bottom-middle window.
- IOTDataBase (4):** The 'Collections (1)' folder is expanded in the bottom-middle window.
- collections Statistics:** The 'Create Collection...' button is highlighted in the bottom-middle window.
- Create Collection:** The 'Collection Name' is set to 'dht11' in the bottom-right window.

Annotations in red dashed boxes highlight the 'Create', 'Name', 'Address', 'Save', 'Create Database', 'iot', 'Collections (1)', 'Create Collection...', and 'Collection Name' fields.

Annotations in blue boxes highlight the 'DHT11 Sensor 온도, 습도가 저장(JSON)' and 'dht11 table에 온습도 정보가 저장'.

Annotations in red dashed boxes highlight the JSON data: `{"tmp":24,"hum":67}`.

WeMos 보드에 전원이 켜져 있어야 됨

## 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));
    });
});
```

The screenshot displays the Node.js IOT Server code and the MongoDB interface. The code in the editor shows the MongoDB connection and MQTT client setup. The console output shows the successful connection to the MongoDB database and the receipt of a message from the MQTT client. The MongoDB interface shows the 'dht11' collection with the following data:

Key	Value	Type
(1) ObjectId("5d8b2d2a0e1408459850a567")	{ 4 fields }	Object
_id	ObjectId("5d8b2d2a0e1408459850a567")	ObjectId
tmp	25	Int32
hum	69	Int32
created_at	2019-09-25 09:02:34.551Z	Date
(2) ObjectId("5d8b2e9b03a4ba59143c73c1")	{ 4 fields }	Object
(3) ObjectId("5d8b2e9ed3a4ba59143c73c2")	{ 4 fields }	Object
(4) ObjectId("5d8b2ea1d3a4ba59143c73c3")	{ 4 fields }	Object
(5) ObjectId("5d8b2ea4d3a4ba59143c73c4")	{ 4 fields }	Object
(6) ObjectId("5d8b2ea7d3a4ba59143c73c5")	{ 4 fields }	Object



## 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]));
        데이터(온도,습도)를 보냅니다.
      }
    });
  });
  socket.on("socket_evt_led",function(data){
    var obj=JSON.parse(data);
    client.publish("led", obj.led+");
  });
});
```

DHT11

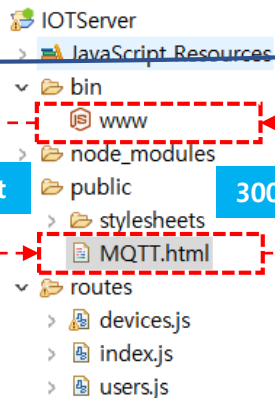
LED

socket\_evt\_mqtt

3000port

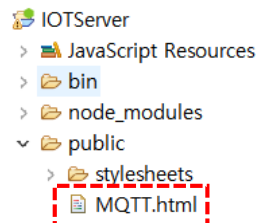
3000port

socket\_evt\_mqtt



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+'<li>'+data.hum+'%'+</li>');
    });
    if(timer==null){
      timer=window.setInterval("timer1()", 1000);
    }
  });
  function timer1(){
    socket.emit("socket_evt_mqtt", JSON.stringify({}));
    console.log("-----");
    데이터(온도,습도)를 주세요
  }
}</script>
</head>
<body>
MQTT 모니터링 서비스
<div id="msg">
  <div id="mqtt_logs">
    <ul class="mqttlist"></ul>
  </div>
</div>
</body>
</html>
```



127.0.0.1:3000/MQTT.html

MQTT 모니터링 서비스

• 26(88%)

## 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]));
      }
    });
  });
});
```

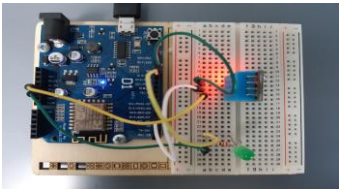
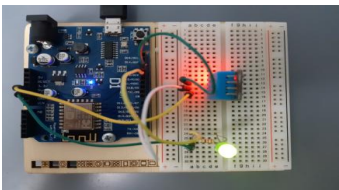
데이터(온도,습도)를 보냅니다.

DHT11

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

MQTT led topic 으로 발행

LED



```
관리자: 명령 프롬프트 - mosquitto_sub -t led -p ...
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd..

C:\Windows>cd..

C:\>cd MQTTProject\mosquitto

C:\MQTTProject\mosquitto>mosquitto_sub -t led -p 1883

1
2
1
2
1
```

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>
```

127.0.0.1:3000/MQTT.html

MQTT 모니터링 서비스

- 27(70%)

socket 통신 방식(LED제어)

LED\_ON LED\_OFF

## 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

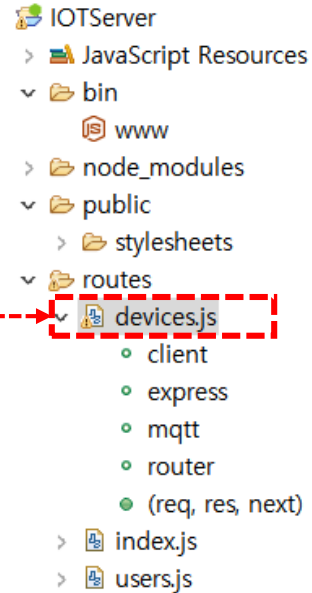
var app = express();

// 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 IP주소 요청할 것

```
관리자: 명령 프롬프트 - mosquitto_sub -t led -p ...
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\WINDOWS\system32>cd..
C:\Windows>cd..
C:\#>cd MQTTProject\mosquitto
C:\MQTTProject\mosquitto>mosquitto_sub -t led -p 1883
1
2
1
2
1
```

MQTT 모니터링 서비스

- 25(73%)

**socket 통신 방식(LED제어)**

LED\_ON LED\_OFF

**REST full Service 통신 방식(LED제어)**

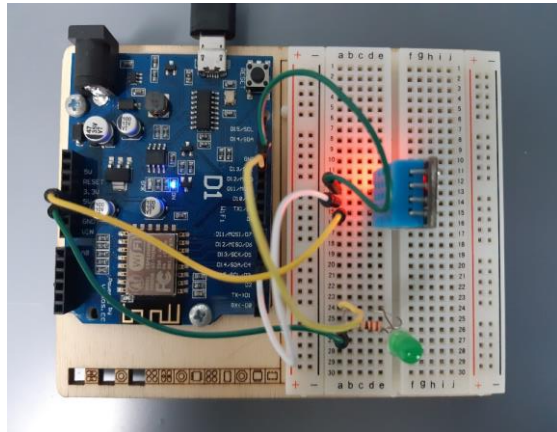
LED\_ON LED\_OFF

LED STATUS

LED\_ON LED\_OFF LED\_ON LED\_OFF  
 on 되었습니다. off 되었습니다.

## IoT Project

## 9. 시연하기



7 WeMos 구동

관리자: 명령 프롬프트 - mosquitto -v

```
C:\MQTTProject\mosquitto>mosquitto -v
1569420036: mosquitto version 1.6.6 starting
1569420036: Using default configuration
1569420036: Opening ipv6 listen socket on port 1883.
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 }
{ "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 }
```

**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
1
1
1
2
1
2
1
2
```

**led sub 구동**

작업 관리자

파일(F) 옵션(O) 보기(V)

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

IOTServer

```
16 */
17 var server = http.createServer(app);
18
19 //MongoDB연결
20 var mongoDB=require("mongodb").MongoClient;
21 var url="mongodb://127.0.0.1:27017/iot";
22 var dbObj=null;
23 mongoDB.connect(url, function(err, db){
24   dbObj=db;
25   console.log("DB Connect .....");
26 });
27
28 var mqtt=require("mqtt");
29 var client=mqtt.connect("mqtt://172.30.1.15");
30 client.on("connect", function(){
31   client.subscribe("dht11");
32 });
33
34 //mqtt 연결
35 client.on("message", function(topic, message){
```

Console 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}
```

IODataBase (4)

System

config

IODataBase localhost:27017 iot

db.getCollection('dht11').find({})

Collections (1)

dht11 0.003 sec.

Key	Value
(1) ObjectId("5d8b2d2a0e1408459850a567")	{ 4 fields }
_id	ObjectId("5d8b2d2a0e1408459850a567")
tmp	25
hum	68
created_at	2019-09-25 09:02:34.551Z
(2) ObjectId("5d8b2e9ebd3a4ba59143c73c1")	{ 4 fields }
(3) ObjectId("5d8b2e9ebd3a4ba59143c73c2")	{ 4 fields }
(4) ObjectId("5d8b2ea1d3a4ba59143c73c3")	{ 4 fields }

4 DBServer 구동

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

```
2019-09-25T16:35:12.247+0900 | STORAGE | [main] Max cache overflow file size
2019-09-25T16:35:12.685+0900 | CONTROL | [main] Automatically disabling TLS
2019-09-25T16:35:12.688+0900 | CONTROL | [initandlisten] MongoDB starting :
2019-09-25T16:35:12.688+0900 | CONTROL | [initandlisten] targetMinOS: Windows
2019-09-25T16:35:12.688+0900 | CONTROL | [initandlisten] db version v4.0.12
2019-09-25T16:35:12.688+0900 | CONTROL | [initandlisten] git version: 5776e3
2019-09-25T16:35:12.688+0900 | CONTROL | [initandlisten] allocator: tcmalloc
2019-09-25T16:35:12.688+0900 | CONTROL | [initandlisten] modules: none
2019-09-25T16:35:12.688+0900 | CONTROL | [initandlisten] build environment:
2019-09-25T16:35:12.688+0900 | CONTROL | [initandlisten] distmod: 200801
2019-09-25T16:35:12.688+0900 | CONTROL | [initandlisten] distarch: x86_6
2019-09-25T16:35:12.688+0900 | CONTROL | [initandlisten] target_arch: x86
```

http://127.0.0.1:3000/MQTT.html

http://172.30.1.15:3000/MQTT.html

← → ↻ 주의 요람 | 172.30.1.15:3000/MQTT.html ☆

MQTT 모니터링 서비스

• 27(72%)

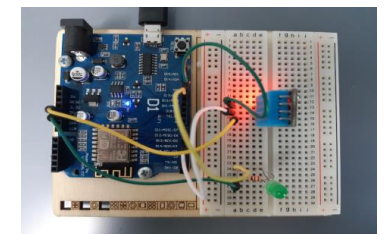
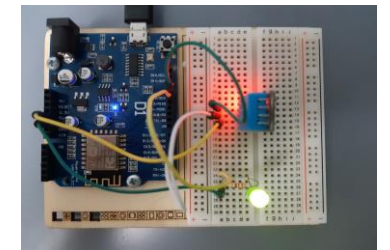
socket 통신 방식(LED제어)

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.