

임베디드 시스템 소프트웨어

IoT 플랫폼과 서비스

2023.12

(주)다인시스

목 차

◆ IoT 표준화

Application Layer Protocol

IoT 시각화 툴 Node-RED

IoT 서비스

Machine Learning

Internet-of-Things (IoT, 사물인터넷)

IoT?

Internet of Things

사물인터넷

Internet of Everything

Web of Things

Embedded Web

Machine to Machine

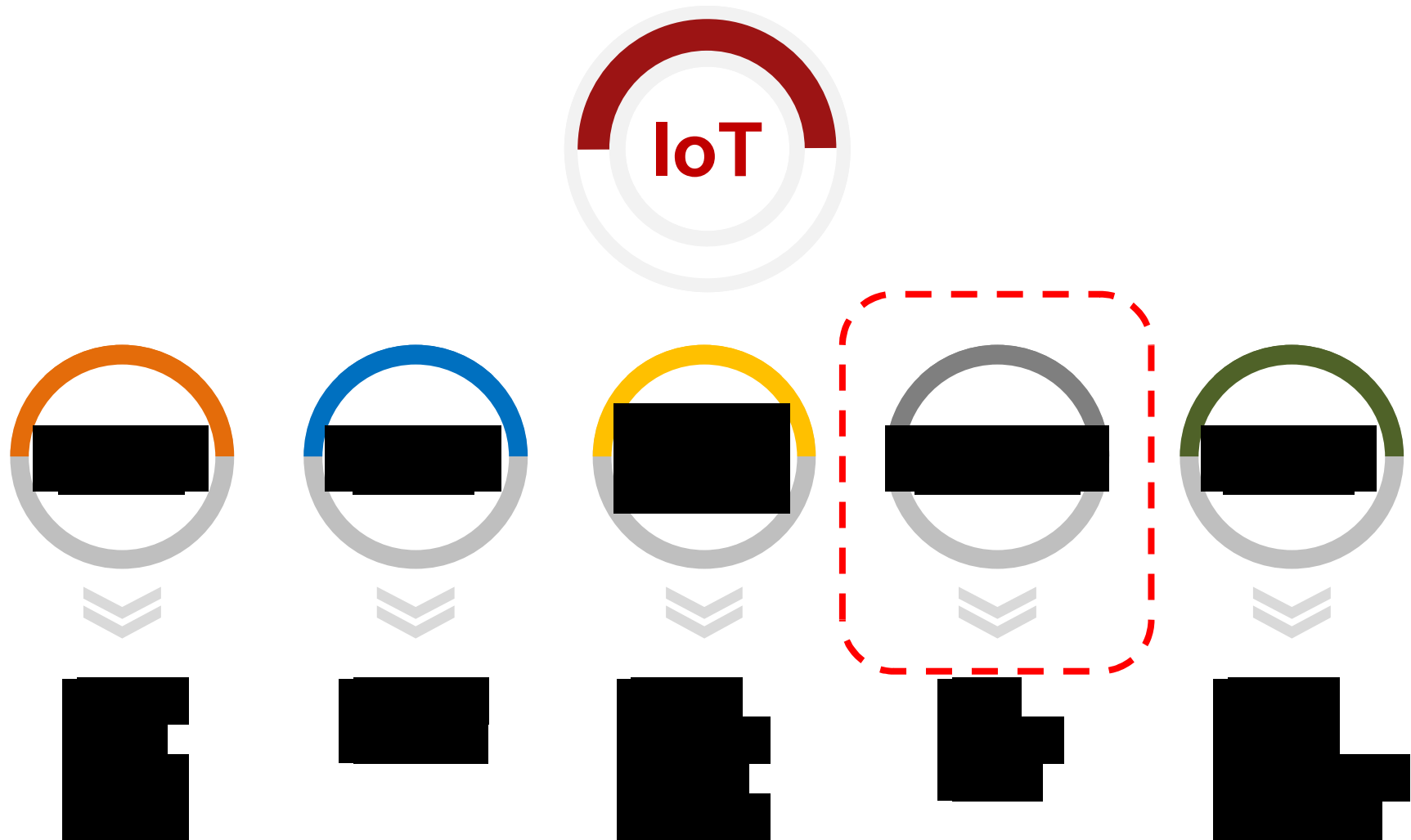
Industry 4.0

인터넷을 기반으로 모든 사물을 연결하여 사람과 사물,
사물과 사물간의 정보를 상호 소통하는

지능형 기술 및 서비스

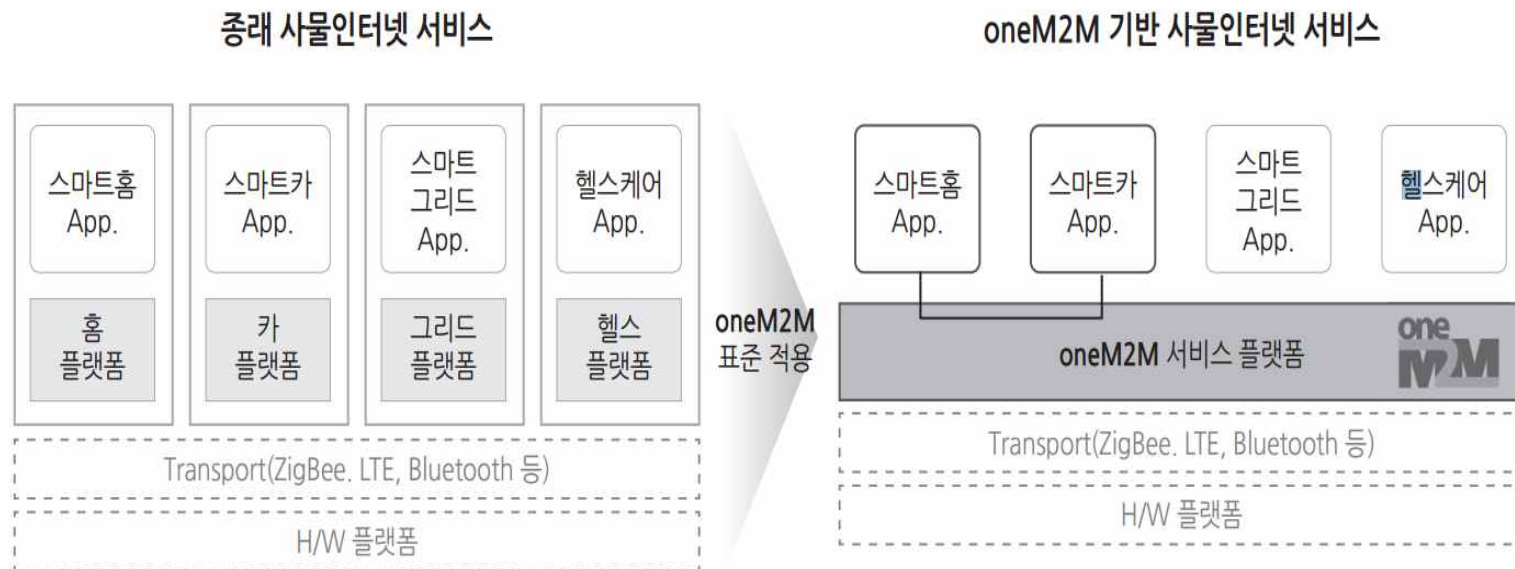
기능화 + 상호연결 + 지능화 = IoT

Internet-of-Things 요소 기술



사물인터넷 표준화

- ◆ 국제표준기구인 ITU-T, ISO/IEC에서 사물인터넷 관련 표준 개발이 활발하며, 글로벌표준기구(de facto) IETF, IEEE, OGC, OMA, ETSI, oneM2M 등에서 사물인터넷 관련 정의, 기술 분류 및 필요한 요소 기술들에 대한 표준들을 활발히 개발



OCF : Open Connectivity Foundation



◆ An industry organization to

- develop standards,
- promote a set of interoperability guidelines, and
- provide a certification program for devices involved in the IoT.

◆ Major membership

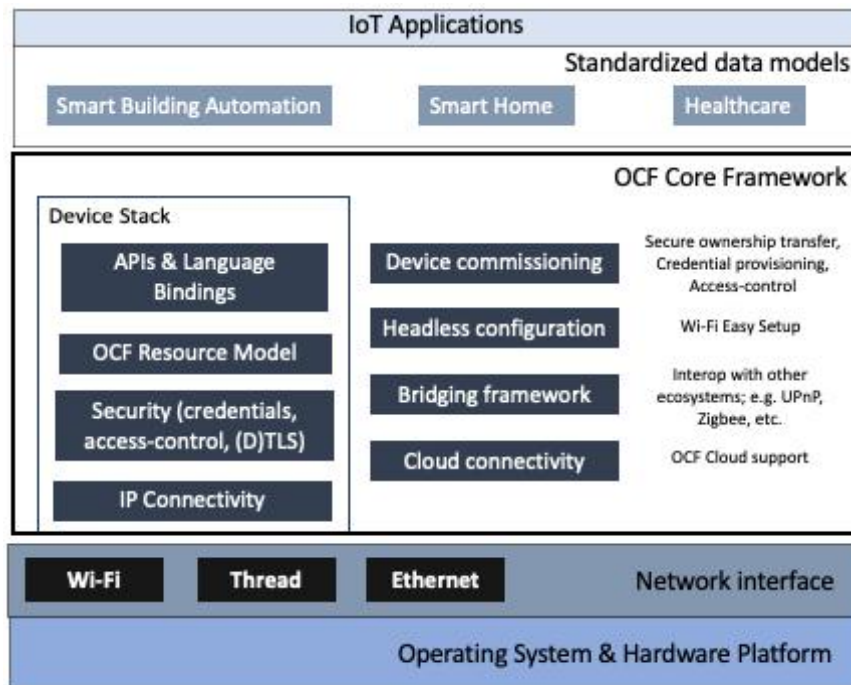
- Samsung Electronics, Intel, Microsoft, Qualcomm and Electrolux.

◆ The OCF delivers a framework that enables these requirements via a specification, a reference implementation and a certification program

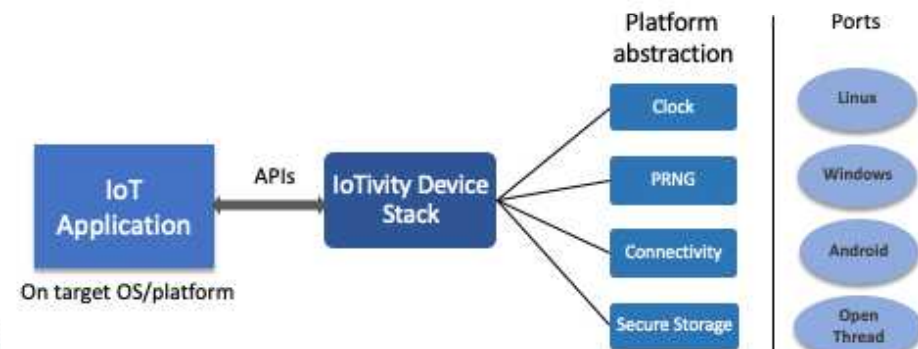
- IoTivity: open source reference implementation developed by different members of the OCF.

IoTivity

- ◆ Open source framework that implements the Open Connectivity Foundation (OCF) standards providing easy and secure communications for IoT devices



IoTivity device stack and modules



IoTivity was designed for rapid portability to any deployment target

목 차

IoT 표준화

◆ **Application Layer Protocol**

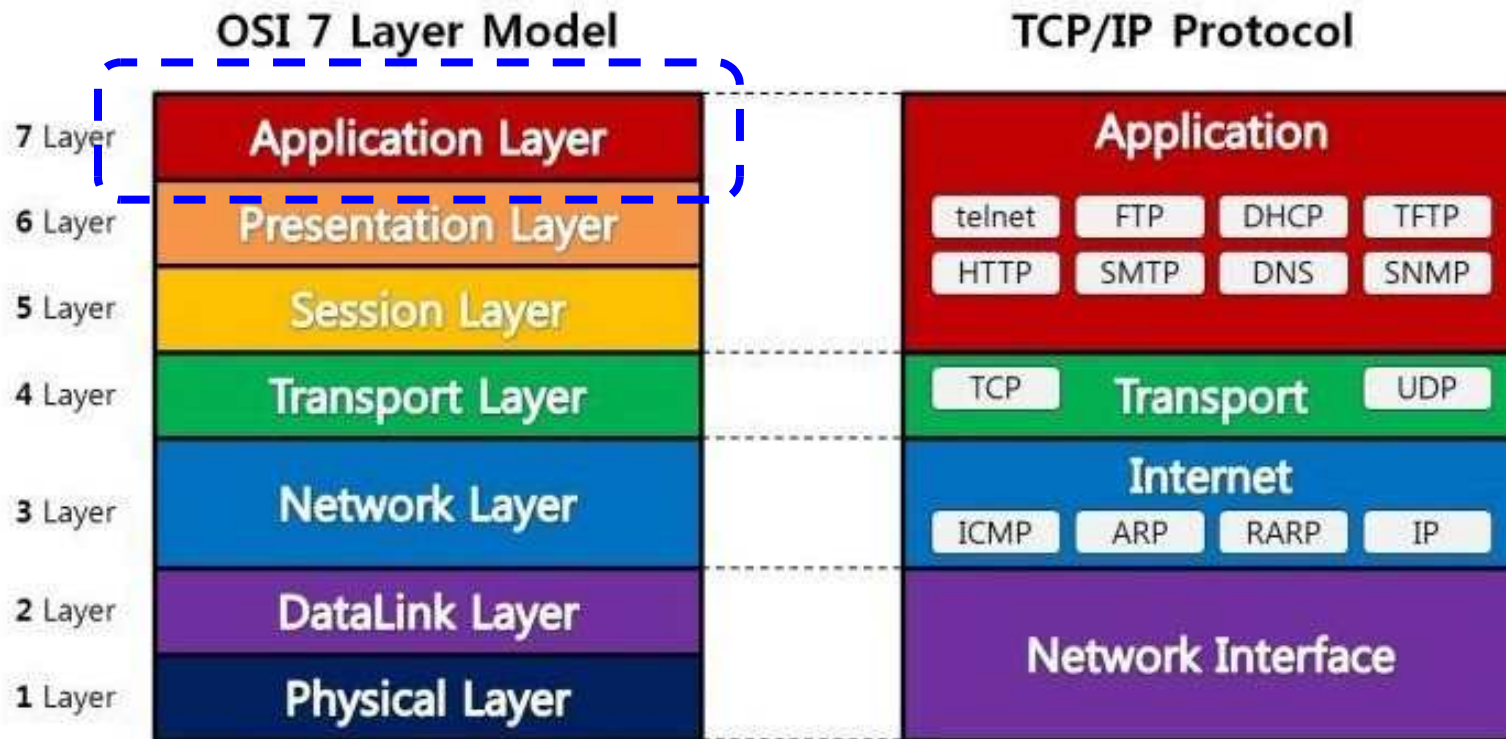
IoT 시각화 툴 Node-RED

IoT 서비스

Machine Learning

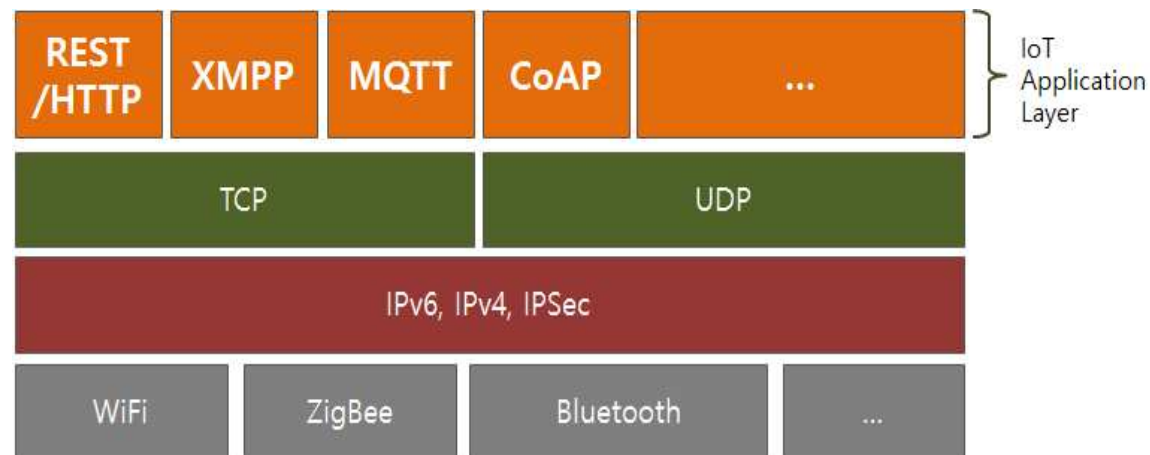
OSI 7 Layer

- ◆ 네트워크 프로토콜이 통신하는 구조를 7개로 분리하여, 각 계층간 상호 작동하는 방식을 정의해 놓은 계층 구조



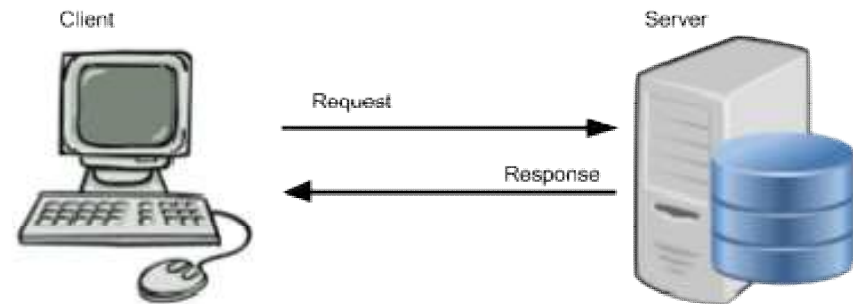
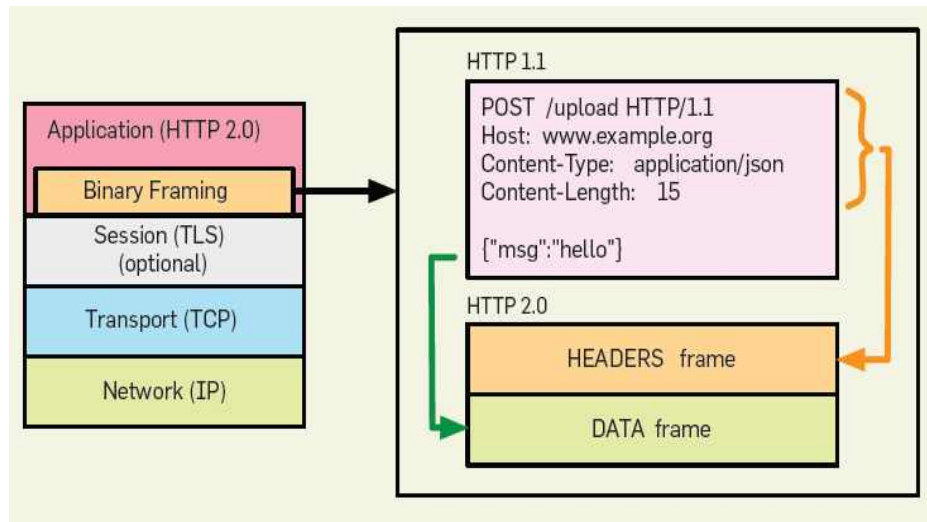
IoT를 위한 Application Layer Protocol

- ◆ IoT 환경에서는 다양한 종류의 사물기기들이 존재하며 빈번한 데이터 전달 및 교환이 발생하고, 네트워크 환경의 경우 메시지 전달의 손실이 자주 발생
- ◆ IoT 특성 상 저사양 하드웨어와 저전력 요구사항을 가짐에도 안정적인 네트워크 동작이 요구됨. 이에 따라 IoT를 위한 메시지 교환 어플리케이션 프로토콜이 개발됨
 - REST (Representational State Transfer), XMPP (Extensible Message and Presence Protocol), MQTT (Message Queuing Telemetry Transport, CoAP (Constrained Environments Application Protocol) 등



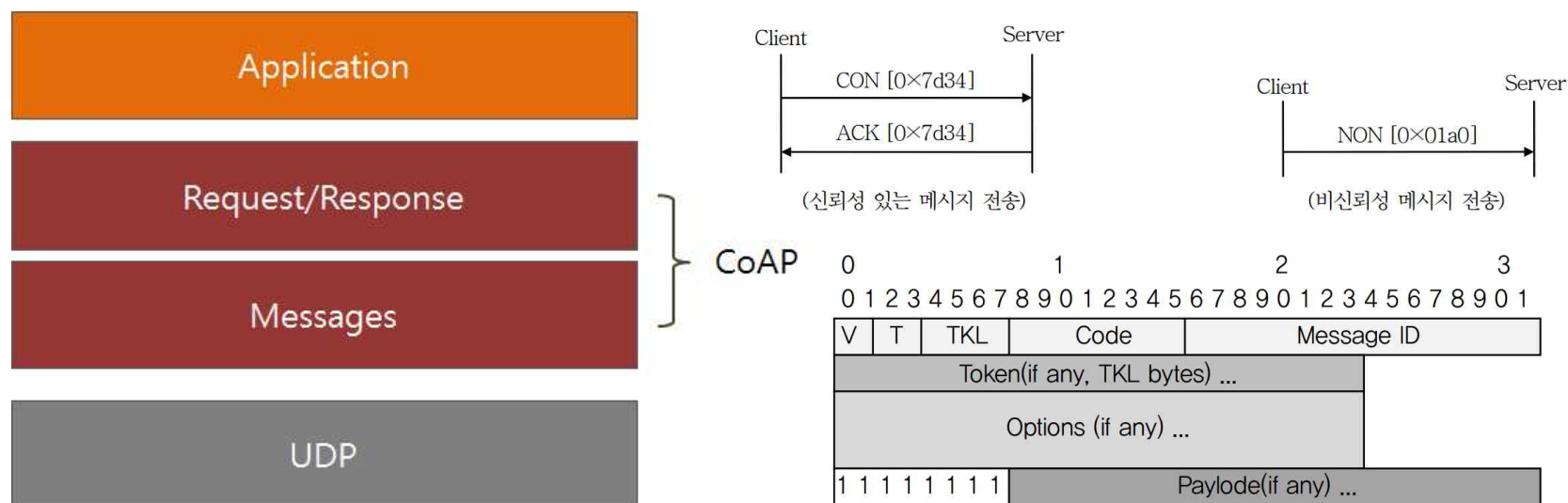
HTTP (Hypertext Transfer Protocol)

- ◆ WWW(World Wide Web)에서 사용되는 통신규약으로 클라이언트와 서버 간의 정보를 주고받을 수 있는 어플리케이션 계층 프로토콜.
- ◆ 주로 HTML 문서를 주고 받는 데에 쓰이며 TCP와 UDP를 전송계층으로 사용.
- ◆ 클라이언트와 서버 사이에 요청/응답(Request/Response) 기반 데이터 교환 방식



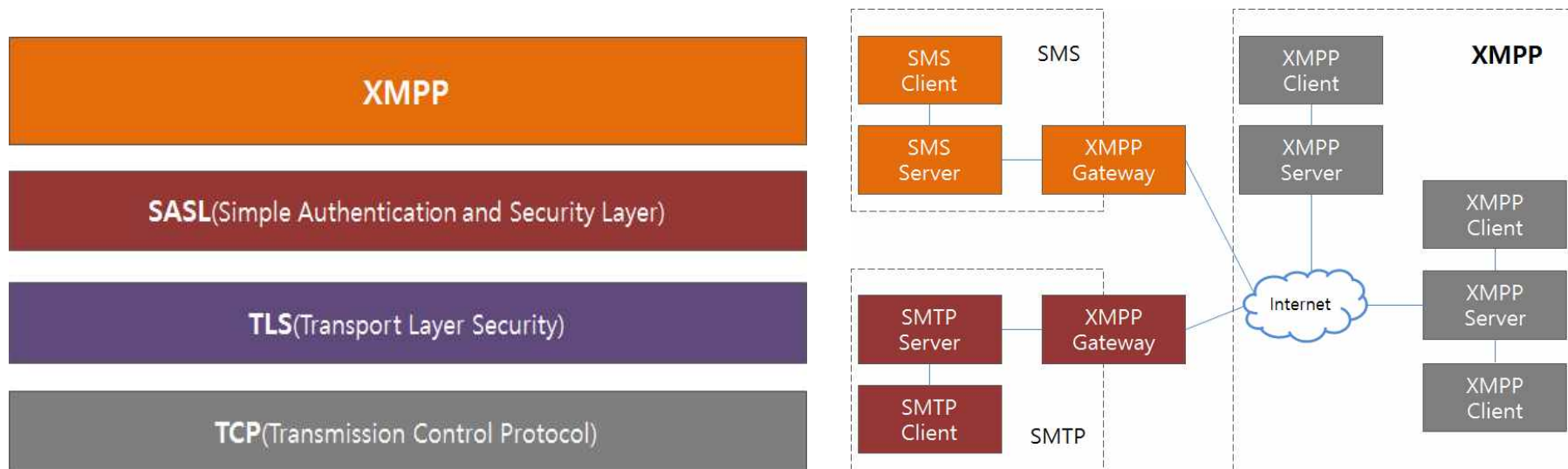
CoAP (Constrained Application Protocol)

- ◆ 저전력, 고손실 네트워크 및 소용량 그리고 소형 노드와 같은 데이터 손실 가능성이 큰 제약적인 환경에서 사용될 수 있도록 특화된 웹 전송 프로토콜
- ◆ TCP 대신 UDP를 사용 (리소스 제약이 있는 기기 고려), UDP기반의 Request/Response 모델로 동작하며 멀티캐스트를 지원
- ◆ 신뢰성 있는 전달을 위해서 재전송 및 타이머 관리를 옵션으로 포함
- ◆ 보안을 위해서 UDP와 CoAP 계층 사이에 DTLS(Datagram Transport Layer Security)계층 사용
- ◆ 네이티브 UDP, 멀티캐스트지원, DTLS보안, 리소스/서비스 검색, 비동기식 통신



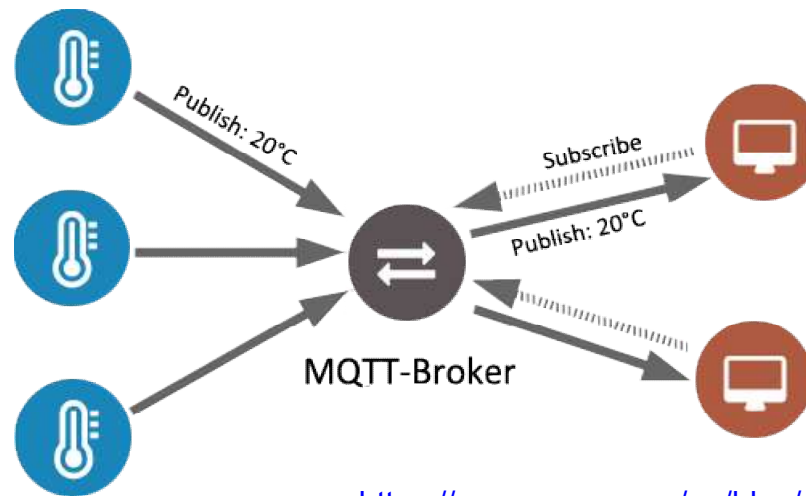
XMPP (Extensible Messaging and Presence Protocol)

- ◆ 2인 이상의 참여자 간에 구조적 데이터의 실시간 교환을 위한 프로토콜
- ◆ 다양한 메신저들에서 사용
 - Google Talk, Facebook, AOL, MSN 메신저 등
- ◆ TCP를 이용하여 동작하고 클라이언트는 도메인 서버에게 데이터를 전송하는 서버 클라이언트 방식으로 동작하기 때문에 서버 기반으로 상대기기간의 인증 및 허가와 관련한 보안요소가 지원되고 있으며 실시간 메시지 전달과 확장성을 고려하여 프로토콜이 설계됨



MQTT (Message, Queuing, Telemetry, Transport)

- ◆ 지연/손실이 심한 네트워크 환경에서 검침기, 센서 등 작은 기기들의 원격제어, 원격측정을 위한 프로토콜로서, 신뢰성/저전력 특징 때문에 IoT에 적용하기 적합
- ◆ ‘publish/subscribe’ 모델을 사용하며, MQTT 네트워크 노드 간에 메시지를 관리하고 라우팅하기 위해 중앙 MQTT 브로커(broker)를 필요로 함
- ◆ TCP를 사용하여 ‘고신뢰성, 정렬, 에러 검사’를 특징으로 하는 전송 계층 구현



- <https://www.emqx.com/en/blog/mqtt-client-tools>
- <https://www.catchpoint.com/network-admin-guide/mqtt-broker>
- <http://www.steves-internet-guide.com/mosquitto-bridge-configuration/>
- <https://tinkerman.cat/post/mqtt-topic-naming-convention>

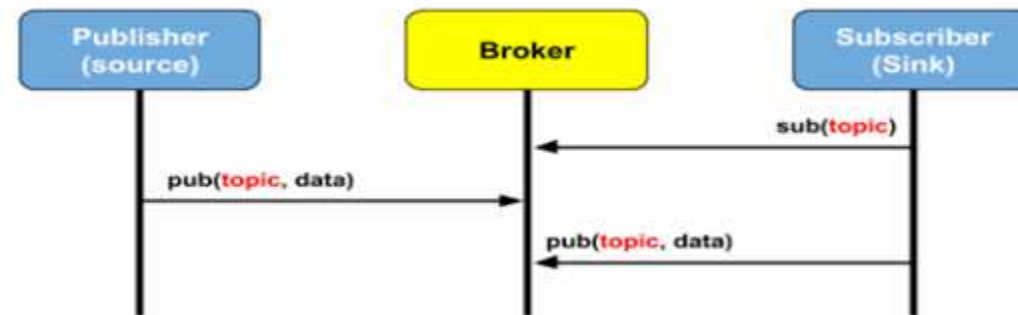
MQTT Broker/Publisher/Subscriber

◆ MQTT 프로토콜

- 푸시 기술(push technology)에서 일반적으로 사용되는 클라이언트/서버 방식 대신, 메시지 매개자(broker)를 통해 송신자가 특정 메시지를 발행(publish)하고 수신자가 메시지를 구독(subscribe)하는 방식 사용
- 즉, 매개자(broker)를 통해 메시지가 송수신된다.

◆ MQTT 구성 요소

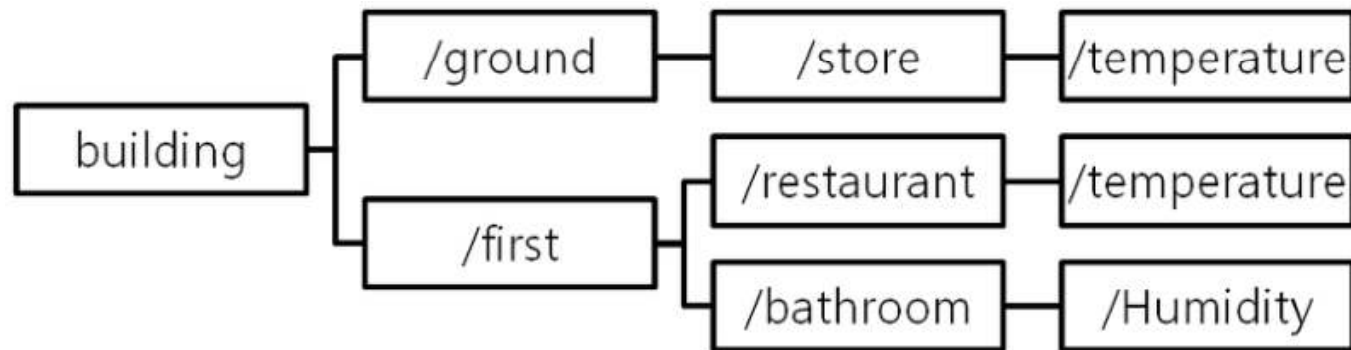
- Broker(브로커)
 - ❖ 브로커는 서버에 연결된 구독자 클라이언트와 발행자 클라이언트 간에 메시지를 전송하는 역할 담당
- Publisher (발행자)
 - ❖ 특정 Topic(화제)을 통해 Broker(중개인)에 메시지를 전송
- Subscriber (구독자)
 - ❖ Topic 기준으로 Broker에 구독을 요청
 - ❖ Subscriber의 polling(주기적인 체크) 방식을 이용하여 Broker에 있는 Topic을 조회



MQTT Topic

◆ Topic(화제)

- Publish와 Subscriber가 발행하고 구독할 수 있는 채널
- Publisher와 Subscriber는 Topic을 기준으로 메시지를 발행하거나 구독
- Topic은 문자열로 구성되어 있기 때문에 / 를 이용하여 계층적으로 구성할 수 있어서 대량의 센서 기기들을 효율적으로 관리 가능



MQTT QoS

◆ QoS(서비스 품질: Quality of Service)

- 각 연결(connection)은 0 ~ 2 사이의 정수 값을 지정하여 브로커에 대한 서비스 품질 지정
- QoS는 TCP 데이터 전송 처리에 영향을 주지 않고 MQTT 클라이언트 사이에서만 영향이 있음

◆ Level 0 (At most Once)

- 메시지는 한번만 전달되며 전달의 성공여부는 확인 하지 않는 레벨

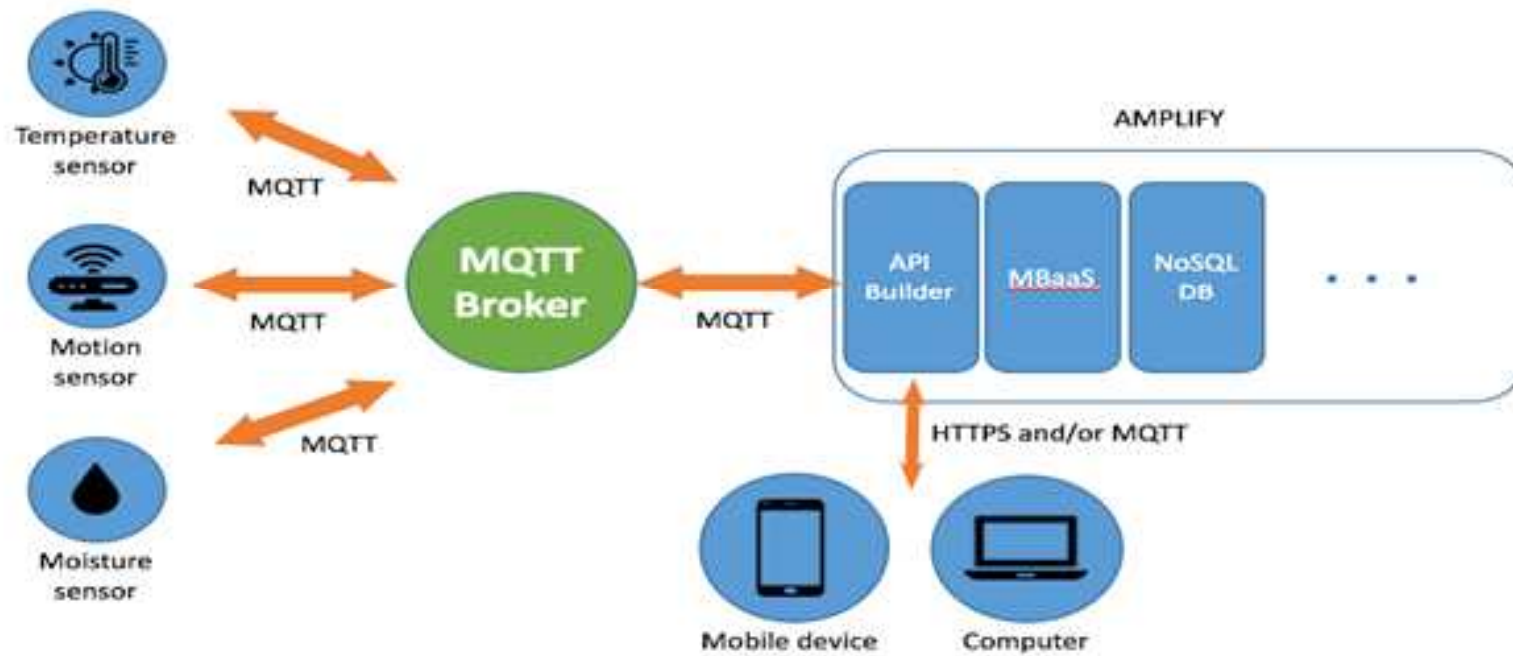
◆ Level 1 (At least Once)

- 메시지는 최소 한번 이상 전달되며 Publisher에게 PUBACK을 전달하여 성공 여부를 알린다.
- 하지만 Publisher가 PUBACK을 성공적으로 받지 못하면 Subscriber에게 중복메시지를 보내는 경우 발생

◆ Level 2 (At Exactly Once)

- 메시지는 반드시 한번만 전달
- PUBACK방식을 PUBREC으로 핸드셰이킹 함으로써 Broker가 PUBACK을 받지 못하더라도 Broker에게 메시지를 보냈다는 사실을 알고 있기 때문에 중복메시지를 보내지 않는다.

MQTT 응용



MQTT Client Tools

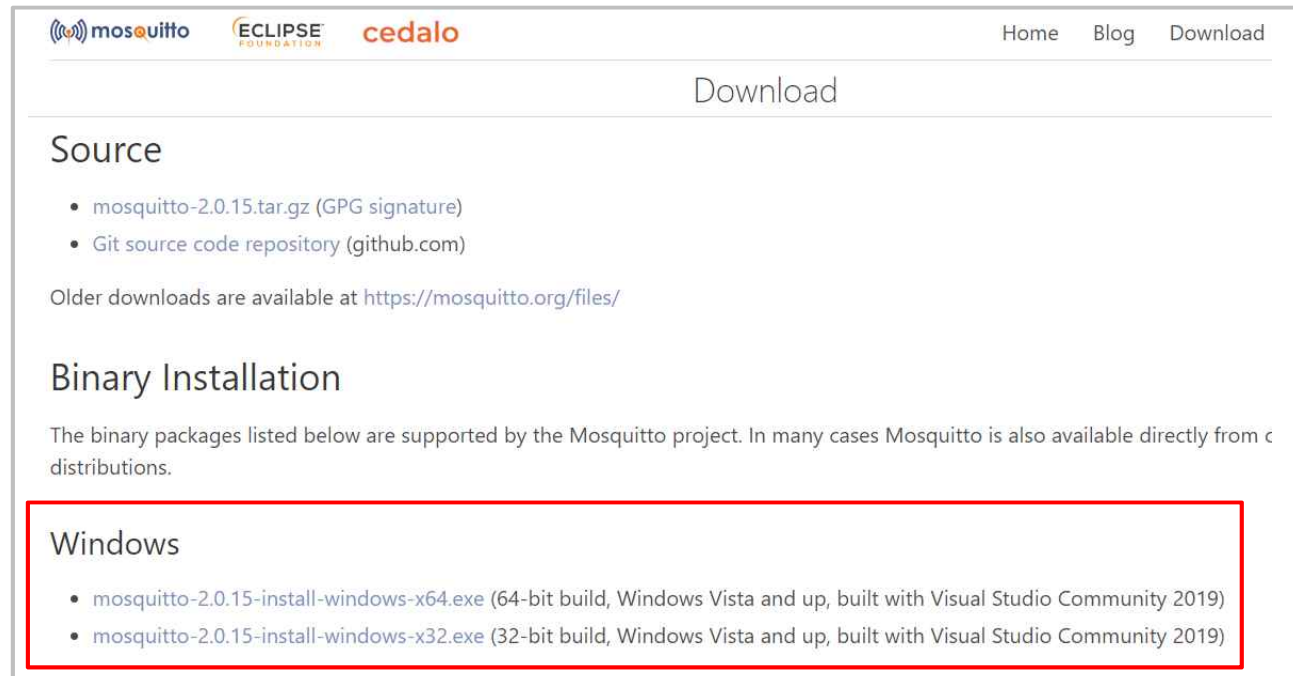
- ◆ MQTT Client Tools 란?
 - MQTT Broker, Publisher와 Subscriber가 서로 연결하여 메시지를 보내고 받을 수 있는 툴

- ◆ 사용 가능한 MQTT Client Tool
 - MQTTX
 - ❖ <https://mqttx.app/>
 - ❖ <https://github.com/emqx/MQTTX/releases>
 - MQTT Explorer
 - ❖ <https://github.com/thomasnordquist/MQTT-Explorer>
 - MQTT.fx
 - ❖ <https://github.com/topics/mqttx>
 - MQTTX Web / MQTTX Cli
 - NanoMQ Cli
 - Eclipse Mosquitto
 - ❖ Lightweight and easy to use, support debug mode
 - ❖ SSL/TLS encryption/authentication support
 - ❖ <https://github.com/eclipse/mosquitto>

[실습1] MQTT Broker Mosquitto 설치 (1)

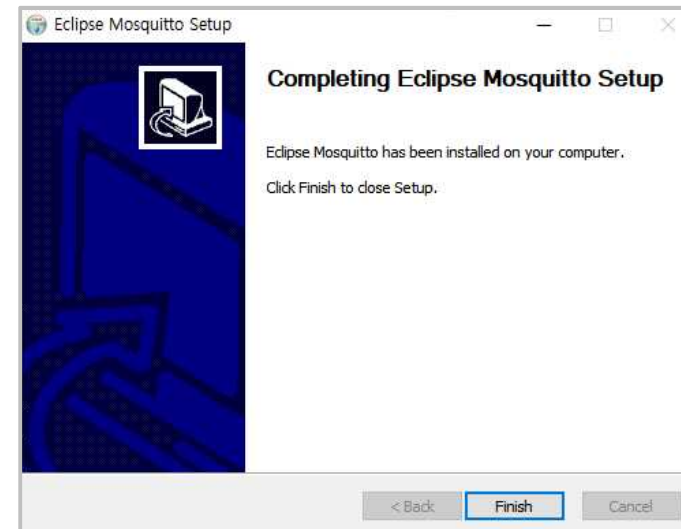
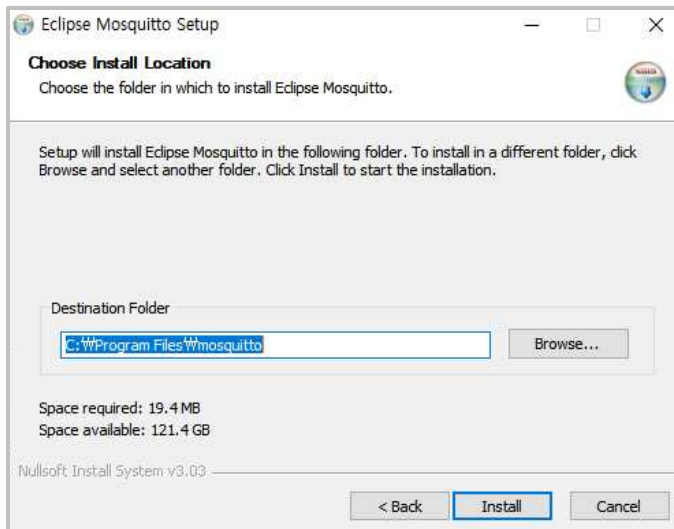
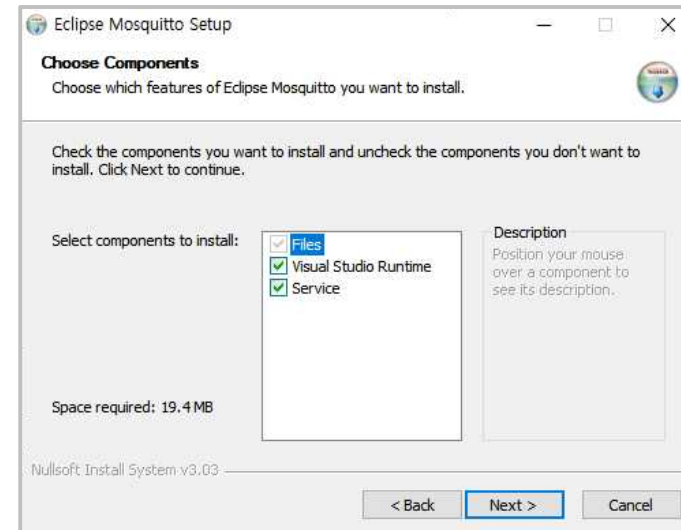
◆ Mosquitto 다운로드 및 설치

- <https://mosquitto.org/download/>



- 참고 : 이전버전에서는 OpenSni 등 의존성 프로그램을 별도로 설치 했으나 현재 버전은 Mosquitto 바이너리만 설치하면 됨

[실습1] MQTT Broker Mosquitto 설치 (2)



[실습1] MQTT Broker Mosquitto 설치 (3)

- ◆ 설치 완료 후 2개의 명령어 창을 열고 설치 디렉토리로 이동
 - 기본 설치 디렉토리 : C:\Programs Files\mosquitto

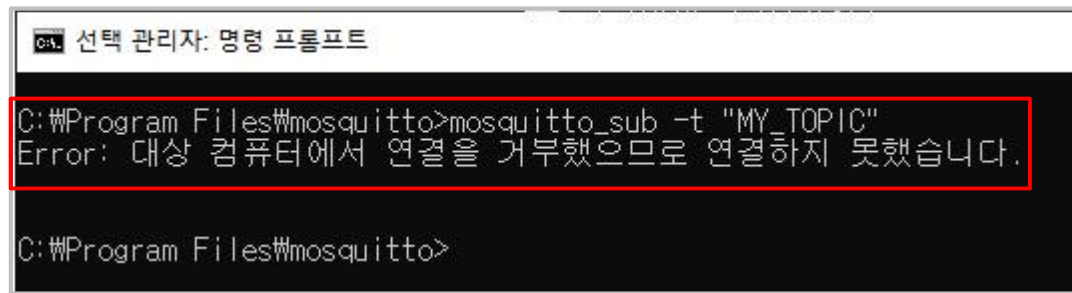
```
선택 관리자: 명령 프롬프트
2023-04-30 오후 04:48 <DIR> .
2023-04-30 오후 04:48 <DIR> ..
2022-08-16 오후 10:34      230 aclfile.example
2022-08-16 오후 10:34 135,368 ChangeLog.txt
2023-04-30 오후 04:47 <DIR> devel
2022-08-16 오후 10:34      1,568 edl-v10
2022-08-16 오후 10:34      14,197 epi-v20
2022-07-06 오전 06:43 3,415,552 libcrypto-1_1-x64.dll
2022-07-06 오전 06:43 685,056 libssl-1_1-x64.dll
2022-08-16 오후 10:34      40,449 mosquitto.conf
2022-08-16 오후 10:35      87,040 mosquitto.dll
2022-08-16 오후 10:41 382,464 mosquitto.exe
2022-08-16 오후 10:35      18,432 mosquitto_topp.dll
2022-08-16 오후 10:35      76,288 mosquitto_ctrl.exe
2022-08-16 오후 10:37 122,880 mosquitto_dynamic_security.dll
2022-08-16 오후 10:34      22,528 mosquitto_passwd.exe
2022-08-16 오후 10:35      51,712 mosquitto_pub.exe
2022-08-16 오후 10:35      79,872 mosquitto_rr.exe
2022-08-16 오후 10:35      81,920 mosquitto_sub.exe
2022-08-16 오후 10:34      1,888 NOTICE.md
2022-08-16 오후 10:34      355 pwwfile.example
2022-08-16 오후 10:34      939 README-letsencrypt.md
2022-08-16 오후 10:34      2,453 README-windows.txt
2022-08-16 오후 10:34      3,768 README.md
2022-08-16 오후 10:34      66,085 Uninstall.exe
2023-04-30 오후 04:47      22개 파일      5,291,042 바이트
                3개 디렉터리 130,320,101,376 바이트 남음

C:\Program Files\mosquitto>
C:\Program Files\mosquitto>
```

[실습1] MQTT Broker Mosquitto 설치 (4)

◆ Mosquitto Subscriber 실행

- 명령 실행 : `mosquitto_sub -t "MY_TOPIC"`

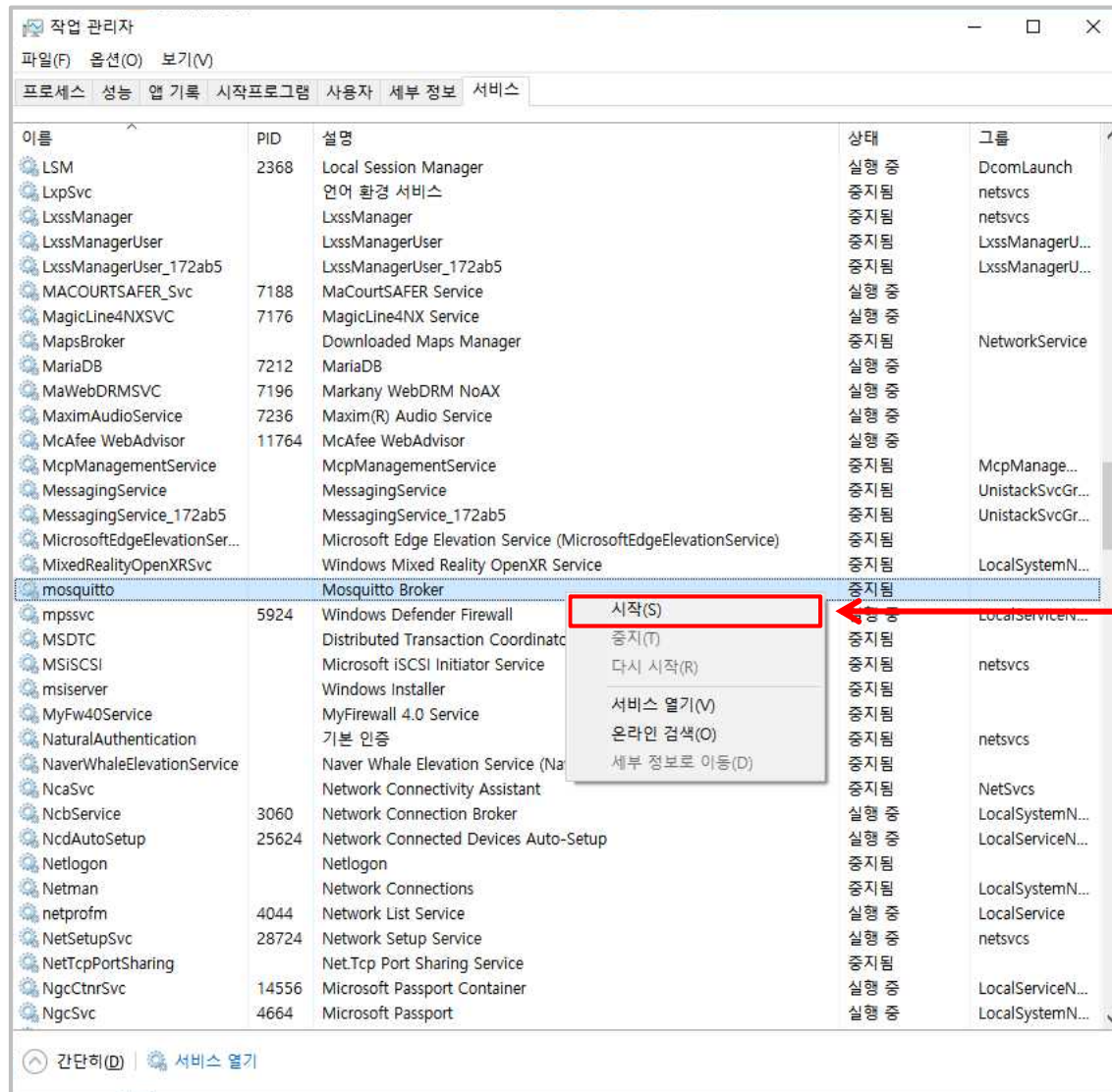


A screenshot of a Windows command prompt window. The title bar reads "선택 관리자: 명령 프롬프트". The command prompt shows the following text:
C:\Program Files\mosquitto>mosquitto_sub -t "MY_TOPIC"
Error: 대상 컴퓨터에서 연결을 거부했으므로 연결하지 못했습니다.
C:\Program Files\mosquitto>

- 명령실행 후 위와 같은 에러 발생
 - ❖ mosquitto 서비스 실행이 안되어 있기 때문에 발생한 에러임
 - ❖ “작업관리자 -> 서비스” 탭에서 mosquitto 서비스를 실행해야 함



[실습1] MQTT Broker Mosquitto 설치 (5)

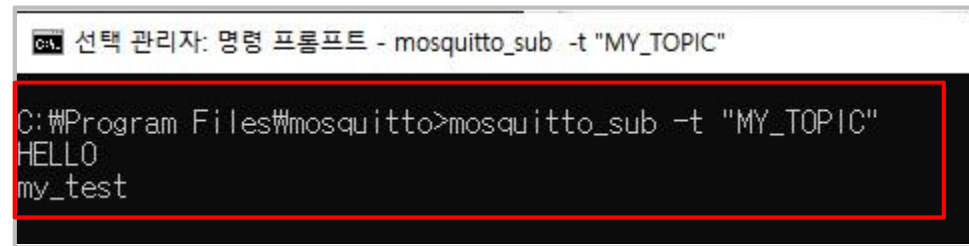


서비스 시작

[실습1] MQTT Broker Mosquitto 설치 (4)

◆ Mosquitto Subscriber 실행

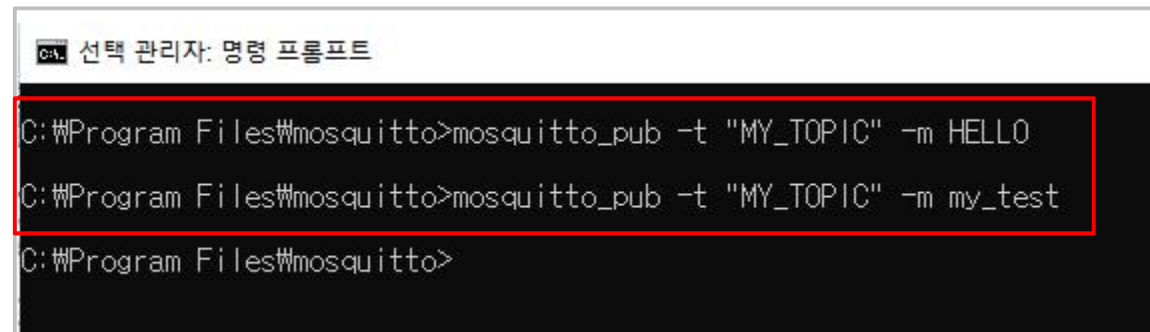
- 명령 실행 : `mosquitto_sub -t "MY_TOPIC"`
- 여기서 "HELLO"와 "my_test"는 Publisher에서 보낸 메시지 임



```
C:\Program Files\mosquitto>mosquitto_sub -t "MY_TOPIC"
HELLO
my_test
```

◆ Mosquitto Publisher 실행

- 명령 실행 : `mosquitto_pub -t "MY_TOPIC" -m "HELLO"`
- 여기서 "HELLO"와 "my_test"는 Broker로 보내는 메시지 임



```
C:\Program Files\mosquitto>mosquitto_pub -t "MY_TOPIC" -m HELLO
C:\Program Files\mosquitto>mosquitto_pub -t "MY_TOPIC" -m my_test
C:\Program Files\mosquitto>
```

목 차

IoT 표준화

Application Layer Protocol

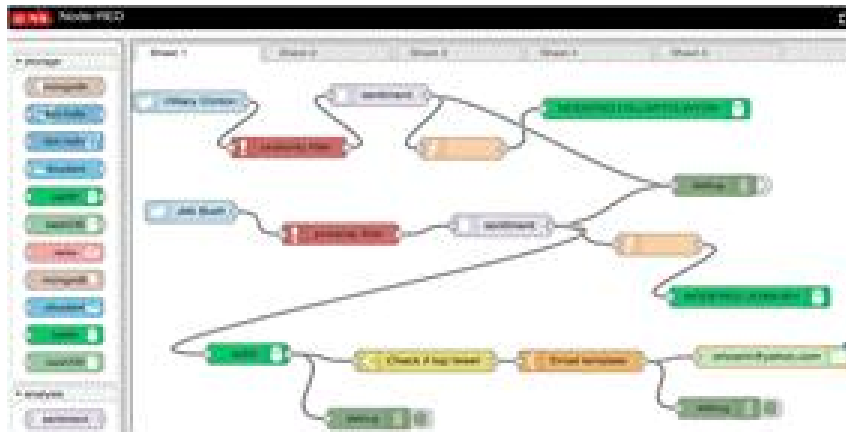
◆ **IoT 시각화 툴 Node-RED**

IoT 서비스

Machine Learning

IoT 시각화 도구 (인포그래픽)

- ◆ 사물 인터넷 장치의 센서 데이터를 수집 및 원격 제어 하기 위해서는 다양한 기술을 필요로 하고, 많은 개발 시간 소요
 - 데이터베이스, 웹서버, 웹프론트엔드(시각화), 프로토콜, 개발언어 등 다양한 기술이 필요 → 많은 개발 시간 소요
- ◆ IoT 시각화 도구(툴)은 사물인터넷 서비스 개발에 필요한 통신 프로토콜과 다양한 정보를 이용하여 플로우(Flow)를 프로그래밍 할 수 있는 인포그래픽 환경 제공



Node-RED

◆ Node-RED(노드 레드)

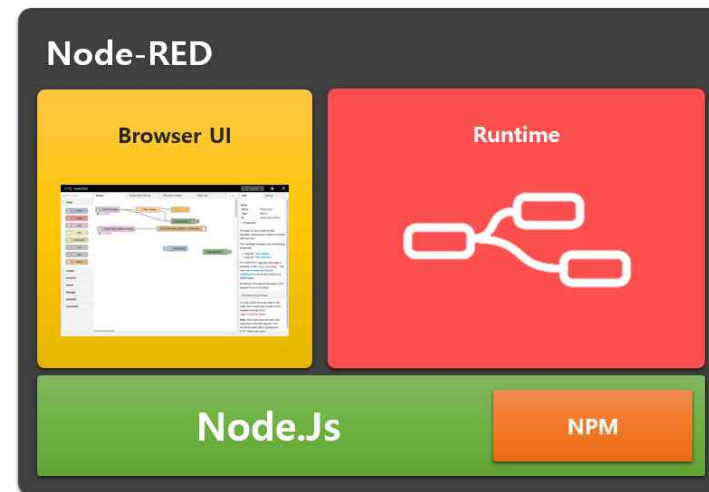
- 하드웨어 장치들, API, 온라인 서비스를 사물인터넷의 일부로 와이어링(배선화)시키기 위해 IBM이 개발한 시각 프로그래밍을 위한 플로우(Flow) 기반 개발 도구
- 브라우저 기반 플로우 편집기를 제공
- 자바스크립트 함수를 개발하는데 사용할 수 있으며 애플리케이션의 요소들은 재사용을 위해 저장하거나 공유 가능
- 런타임은 Node.js 위에서 개발되었으며 Node-RED에서 만든 플로는 JSON을 사용하여 저장

◆ Node-RED 구조

- 노드(Node)라고 불리는 블록을 이용하여 네트워크 응용 프로그램의 동작 구조를 쉽게 설계
- 강력한 비동기 런타임인 Node.js 위에서 구동
- 사용자에게 웹 기반의 플로우 에디터를 제공하기 위한 서버와 만든 플로우를 실제로 구동시키는 Runtime 이 하나의 소프트웨어에서 작동

Node-RED

- ◆ Node-RED(노드 레드) : <https://nodered.org/>
 - 하드웨어 장치들, API, 온라인 서비스를 사물인터넷의 일부로 와이어링(배선화)시키기 위해 IBM이 개발한 시각 프로그래밍을 위한 플로우(Flow) 기반 개발 도구
 - 브라우저 기반으로 각 노드(Node)간 플로우 배선을 쉽게 설계 가능
 - 한번의 click로 runtime으로 배포(deploy)와 runtime 구동 가능
- ◆ 주요 특징
 - Browser-based flow editing
 - Built on Node.js
 - Social development



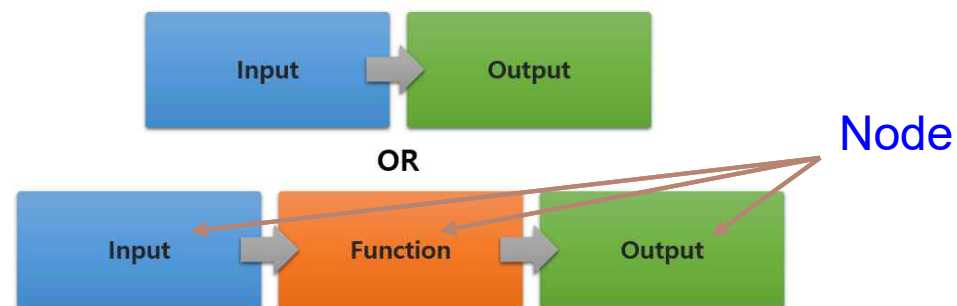
Node-RED 를 활용한 IoT 응용 명세/구현

◆ Node-Red 로 Application을 구현하는 방식은 크게 세 가지로 구분

- Design: 노드를 조합하여 하나의 플로우를 설계
- Deploy: 설계한 플로우를 배포
- Runtime: 설계된 플로우 Node.js로 실행

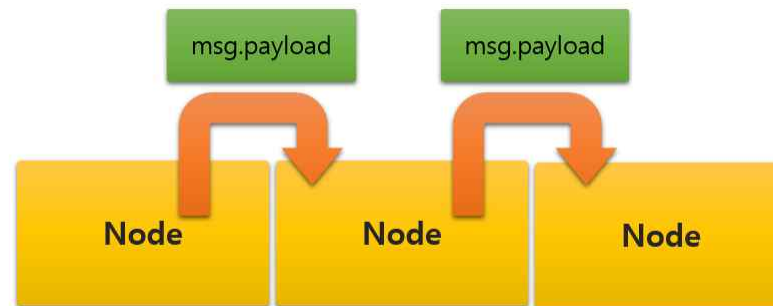


◆ 설계 플로우 예시



Node-RED 를 활용한 IoT 응용 명세/구현 (cont'd)

- ◆ Input, Output, Function 세 가지 종류의 Node
- ◆ 하나의 flow는 Input Node에서 시작하여 Output Node로 결과 전달
- ◆ Input과 output node 사이에 data 가공을 위해 Function Node가 사용될 수 있음
- ◆ 하나의 Node는 다음 Node로 JSON 형식 데이터를 전달
 - 기본값: Message 오브젝트에 payload 속성



Node-RED 환경 구성

◆ Node-RED 환경 구성 절차

1. Node.js(LTS 8.x이상) 설치
2. 윈도우 명령 프롬프트 또는 Powershell 실행
3. 명령 프롬프트에서 Node.js설치 확인 및 *npm정보 확인
4. Node-RED 설치 (by npm)
5. 서버와 플로우(Flows)를 실행
6. 크롬 웹 브라우저를 실행
 - ❖ 주소 입력창에 해당 서버 주소를 입력
 - ❖ 기본 로컬 서버 주소는 'http://localhost:1880'으로 접속

Node-RED 환경 구성 : Node.js 설치 (1)

◆ Node.js 설치

- 다운로드 경로 : <https://nodejs.org>
- 설치파일 : node-v16.x.x-x64.msi



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for Windows (x64)

16.17.0 LTS

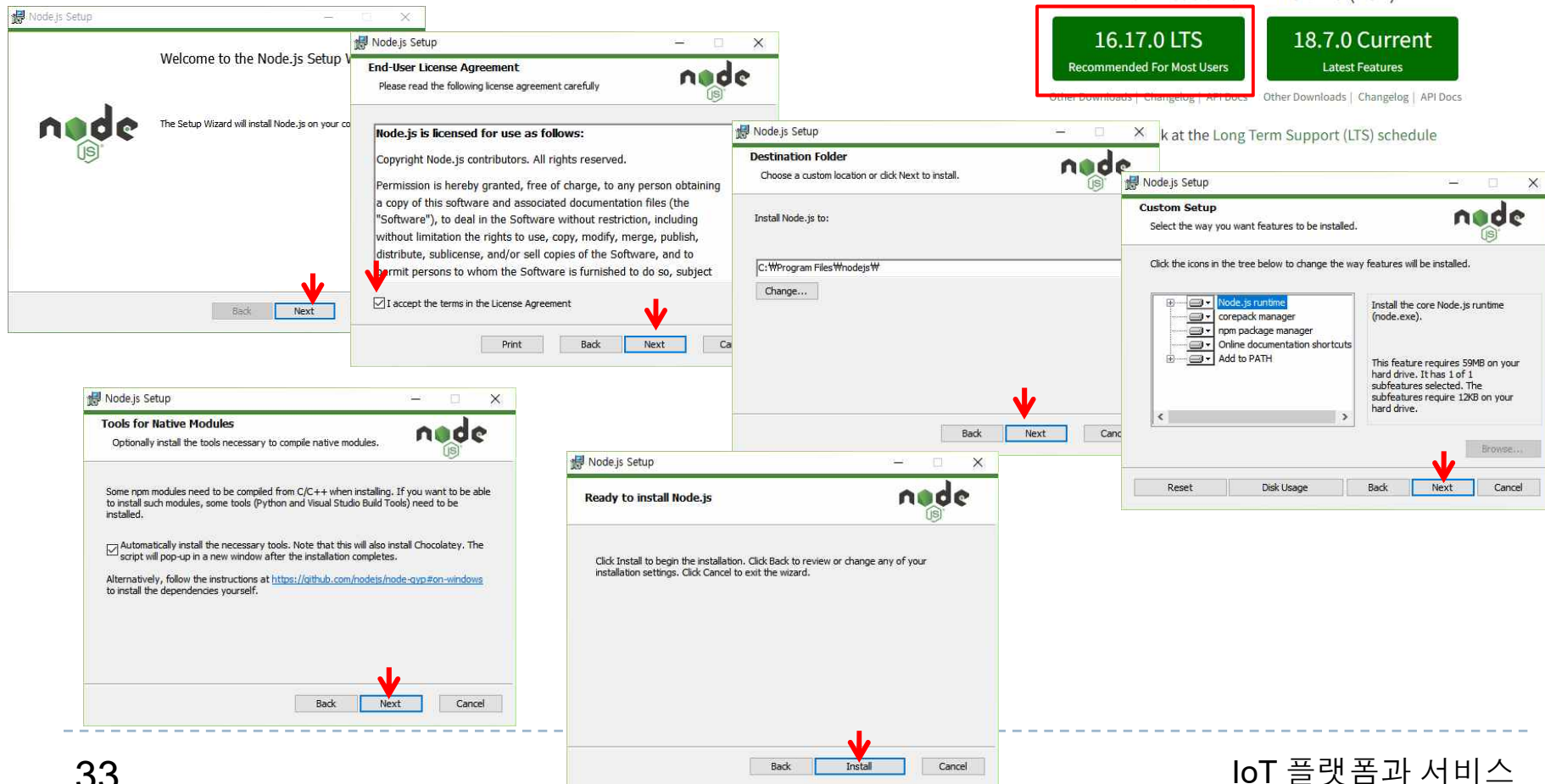
Recommended For Most Users

18.7.0 Current

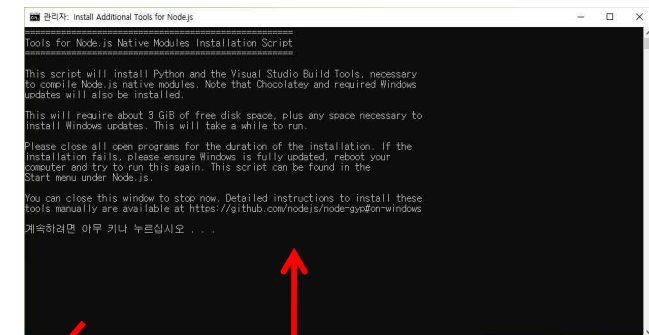
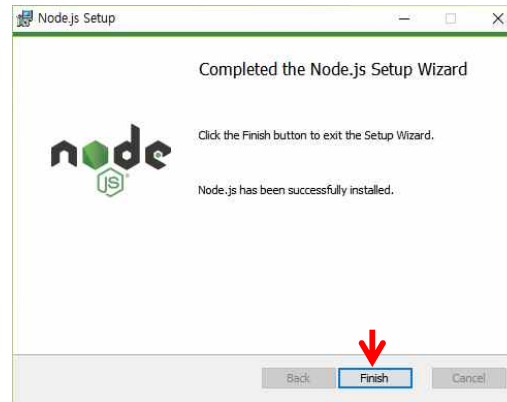
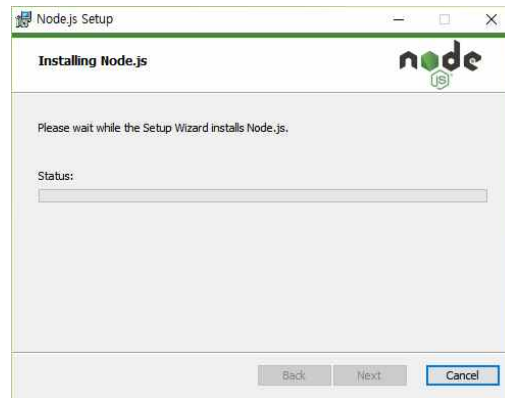
Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

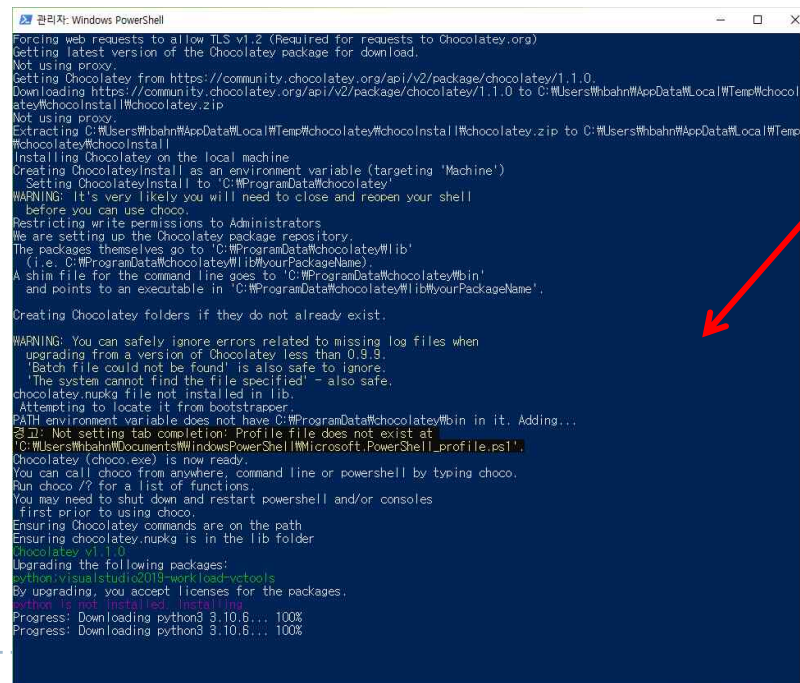
[Learn more about the Long Term Support \(LTS\) schedule](#)



Node-RED 환경 구성 : Node.js 설치 (2)



Press Any Key



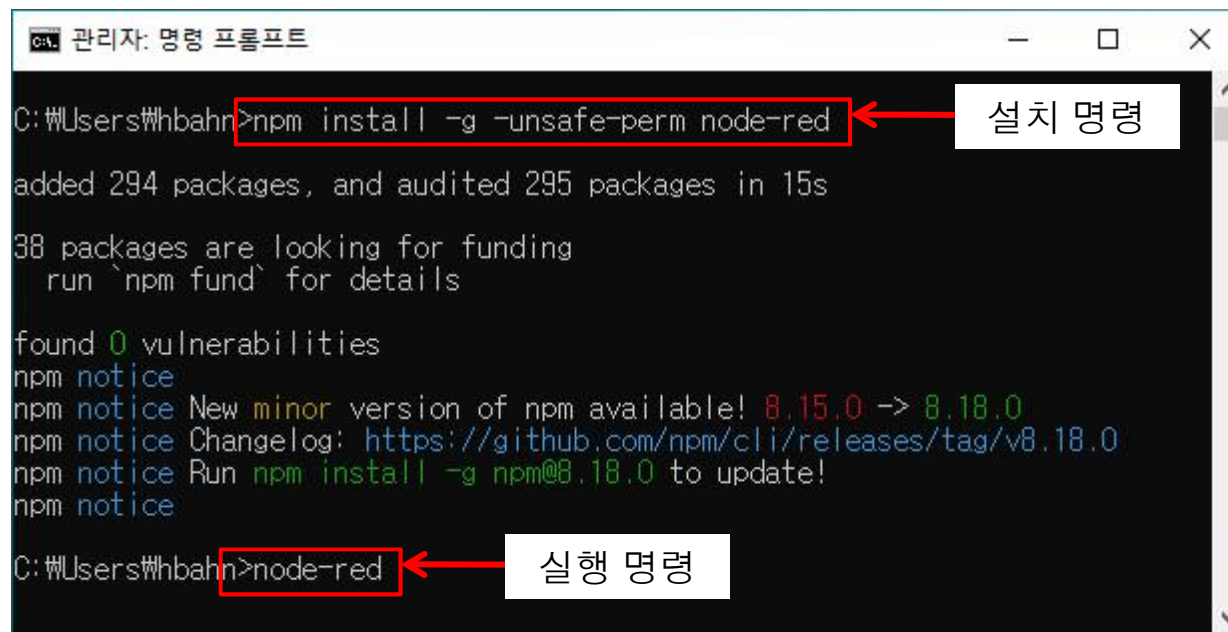
Install Additional Packages

설치 완료되고 Powershell 종료 될 때까지 대기

Node-RED 환경 구성 : Node-RED 설치

◆ Node-RED 설치

- 윈도우 명령 프롬프트 실행
- 윈도우 실행 > 'cmd' 입력 > 오른쪽 마우스 > '관리자 권한' 으로 실행



The screenshot shows a Windows Command Prompt window titled "관리자: 명령 프롬프트". The command prompt shows the following output:

```
C:\Users\hbbahn>npm install -g -unsafe-perm node-red
added 294 packages, and audited 295 packages in 15s

38 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 8.15.0 -> 8.18.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.18.0
npm notice Run `npm install -g npm@8.18.0` to update!
npm notice
C:\Users\hbbahn>node-red
```

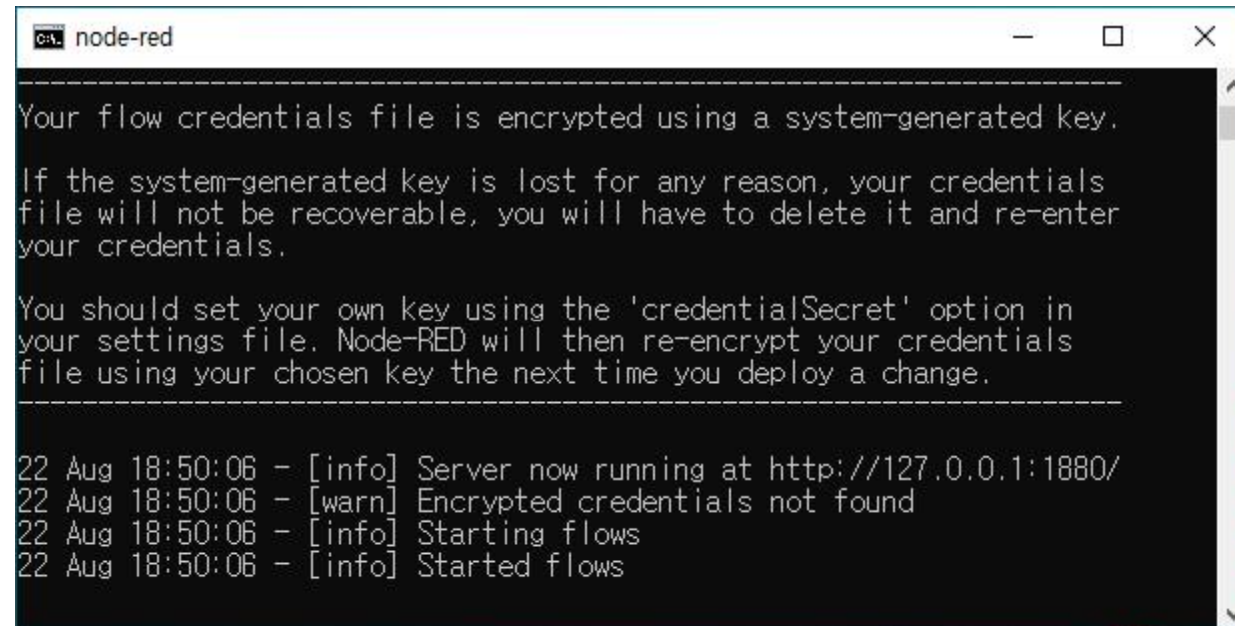
Two red boxes highlight the commands entered in the prompt. The first box, around the command `npm install -g -unsafe-perm node-red`, is pointed to by a red arrow from a white box labeled "설치 명령". The second box, around the command `node-red`, is pointed to by a red arrow from a white box labeled "실행 명령".

- ▲ node-red 실행 : node-red 설치 후 실행,
설치 명령어: `npm install -g -unsafe-perm node-red`

Node-RED 환경 구성 : Node-RED 실행

◆ Start flows

- 실행 명령 : node-red

A terminal window titled 'node-red' with standard window controls. The terminal output shows a warning about encrypted credentials and a confirmation that flows have started.

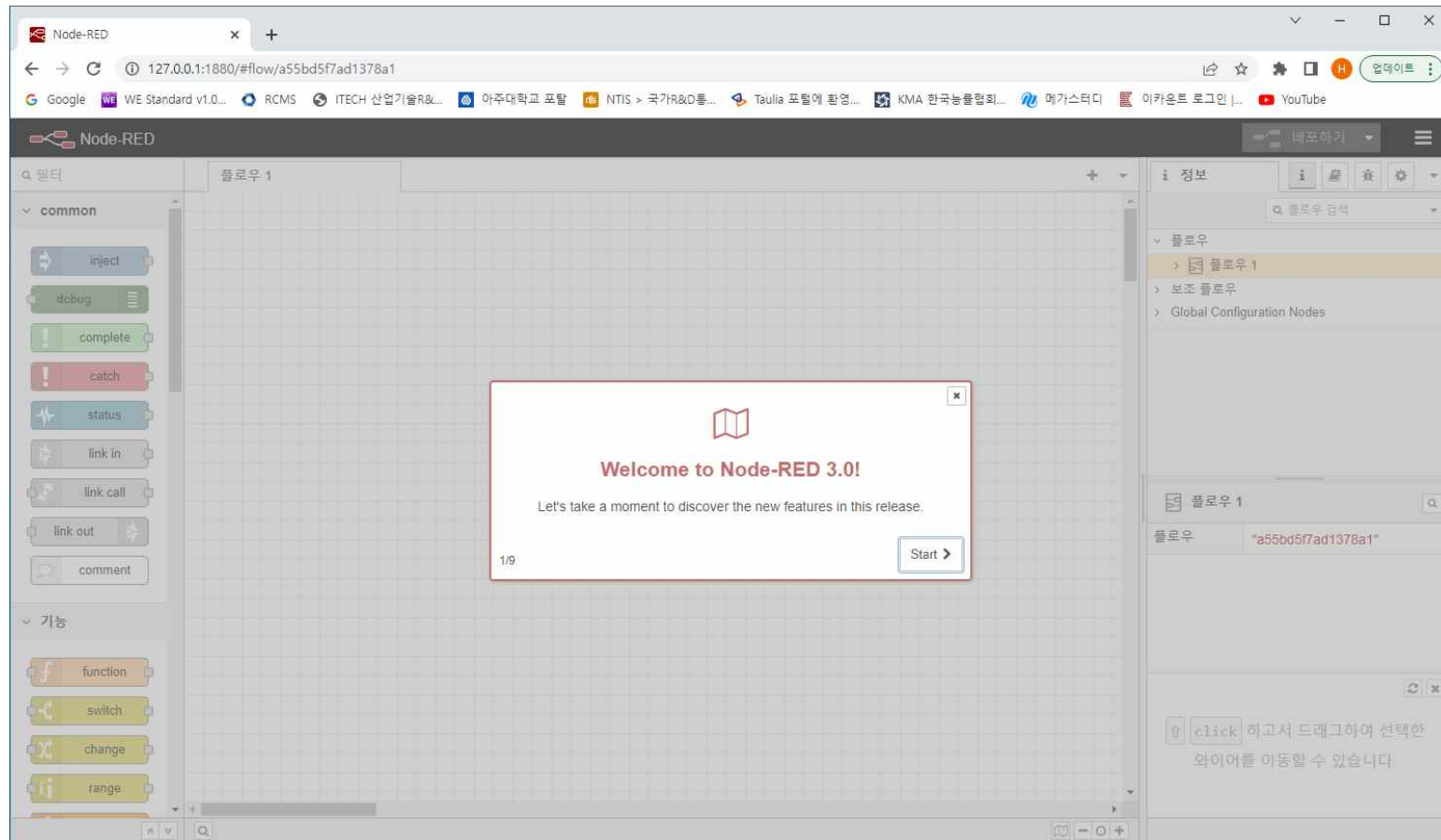
```
node-red
Your flow credentials file is encrypted using a system-generated key.
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----
22 Aug 18:50:06 - [info] Server now running at http://127.0.0.1:1880/
22 Aug 18:50:06 - [warn] Encrypted credentials not found
22 Aug 18:50:06 - [info] Starting flows
22 Aug 18:50:06 - [info] Started flows
```

- Node-RED 정상 실행 확인
“Started flows”

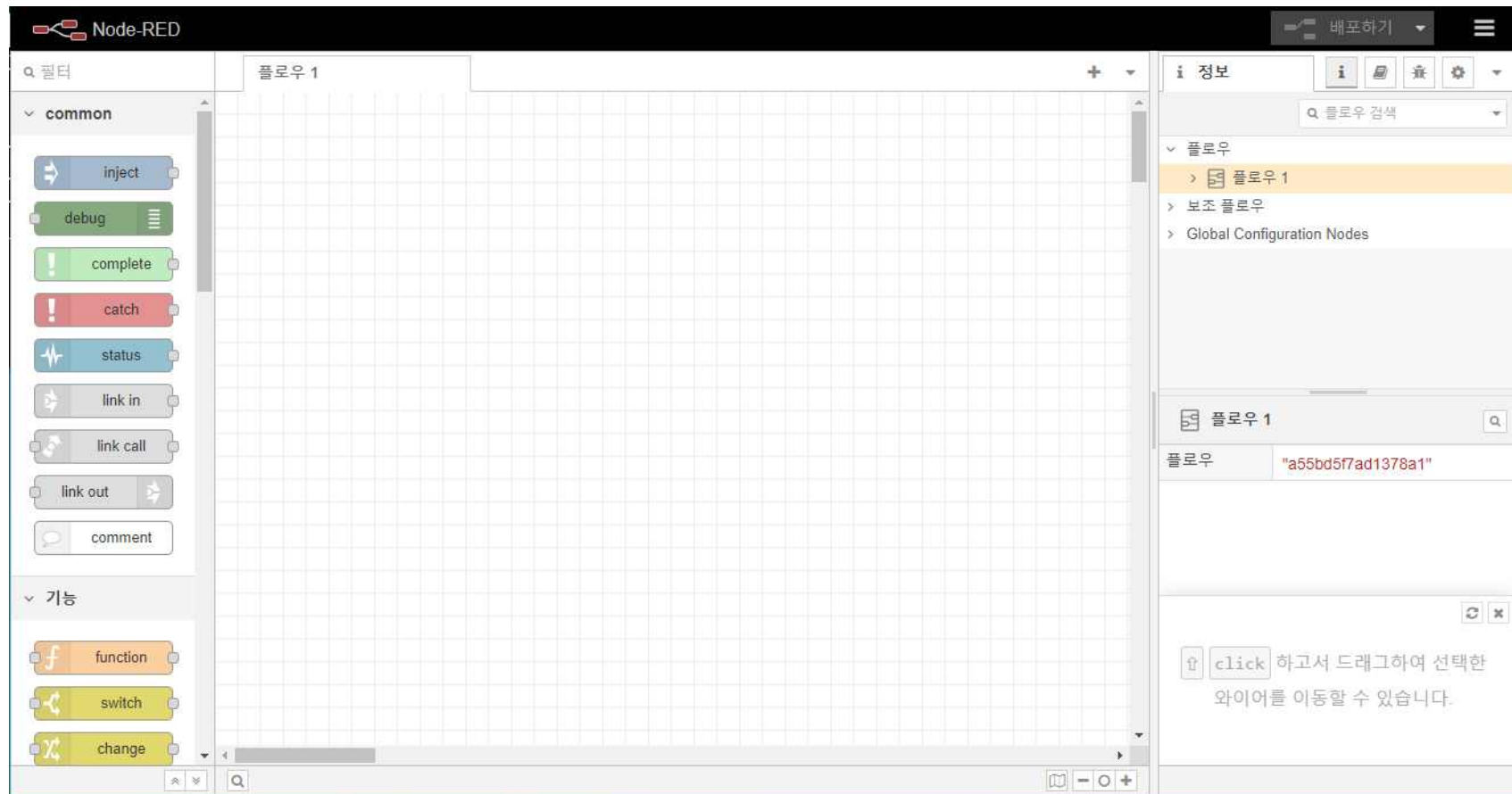
Node-RED 환경 구성 : Node-RED 접속

◆ Chrome 브라우저를 이용하여 Node-RED에 접속

- 접속 주소 : <http://127.0.0.1:1880/>



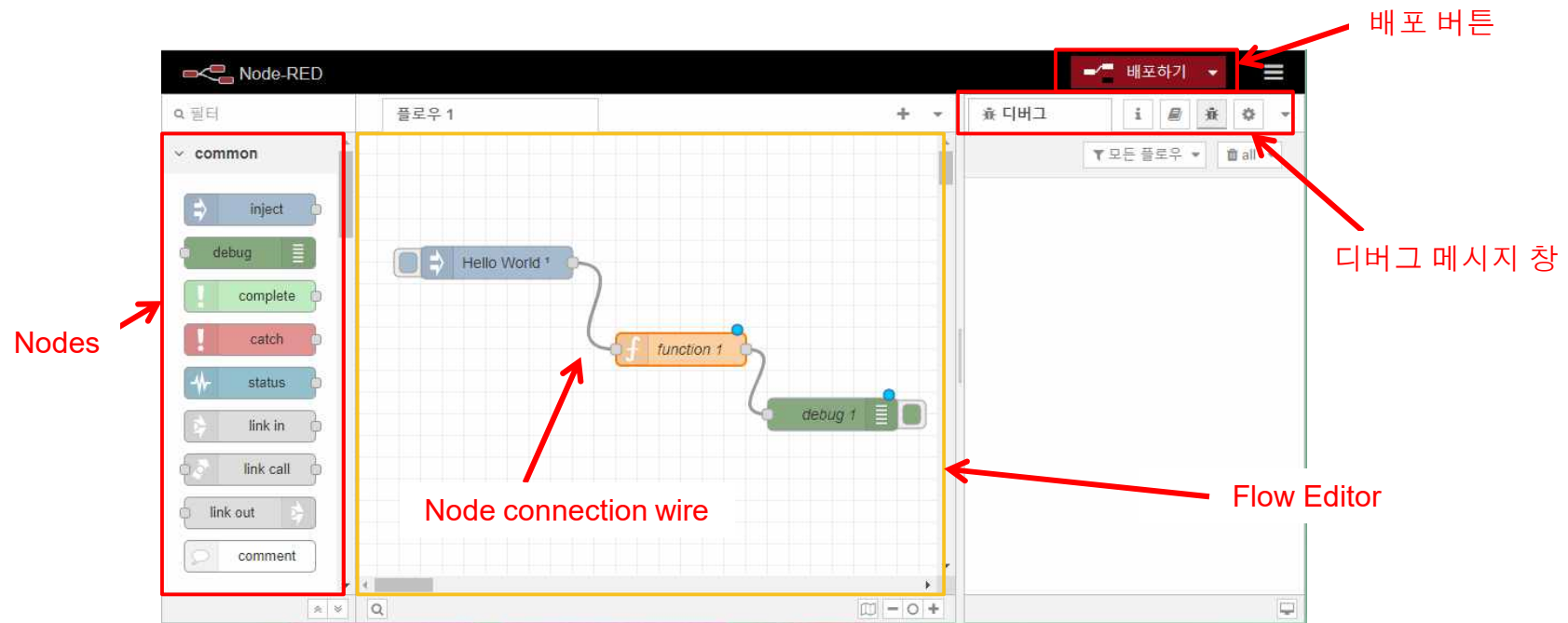
Node-RED 환경 구성 : Node-RED 환경 완료



Node-RED 구성

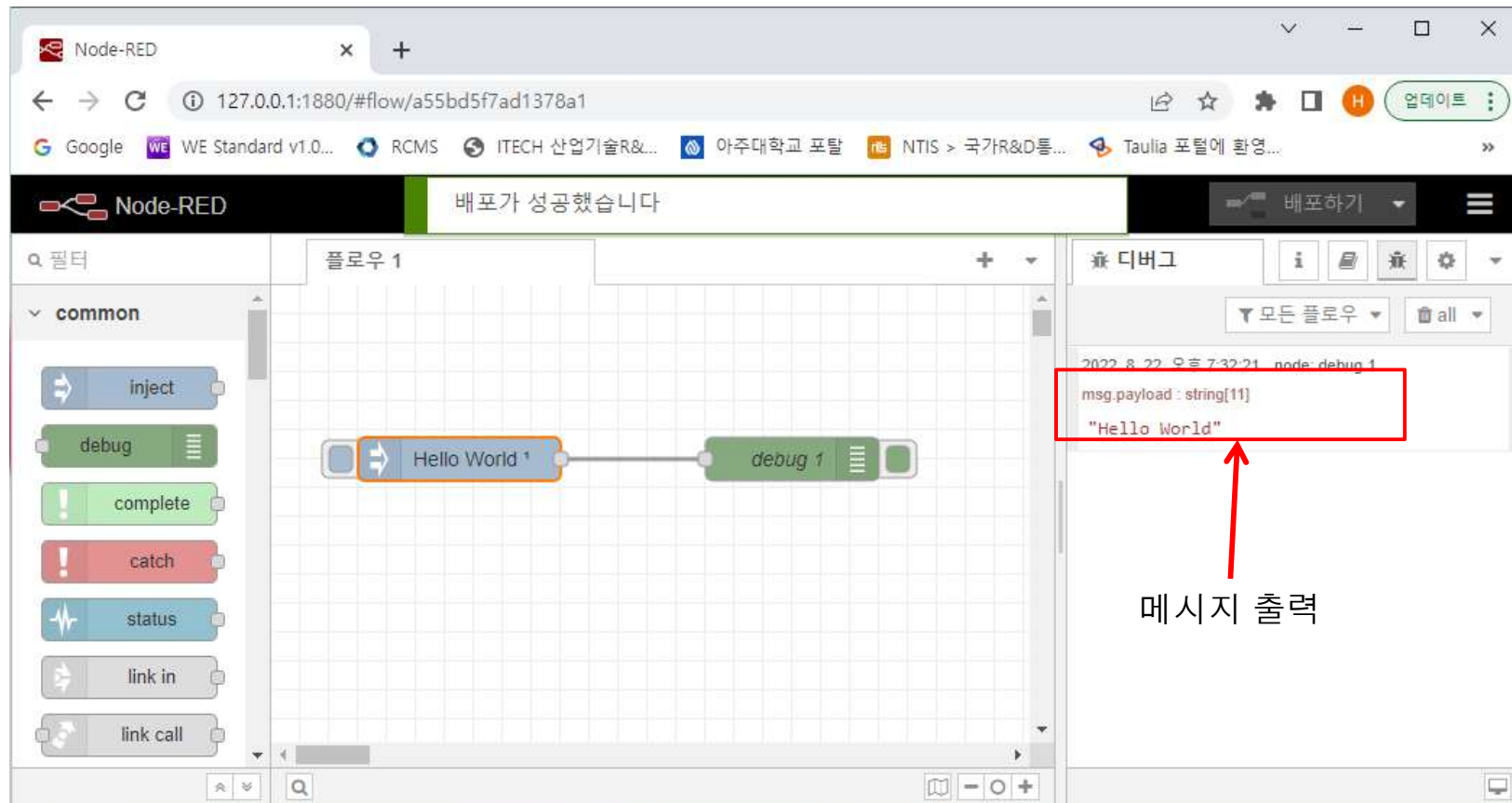
◆ Node-RED는 아래와 같이 4가지로 분류

- Flow Editor : 노드를 이용하여 플로우를 설계할 수 있는 에디터
- Nodes : 배치할 수 있는 노드들의 리스트
- Service Start : 플로우를 배포(deploy) 할 수 있는 버튼
- Debug Message : 배포(Deploy)된 플로우의 디버그 메시지를 확인 가능



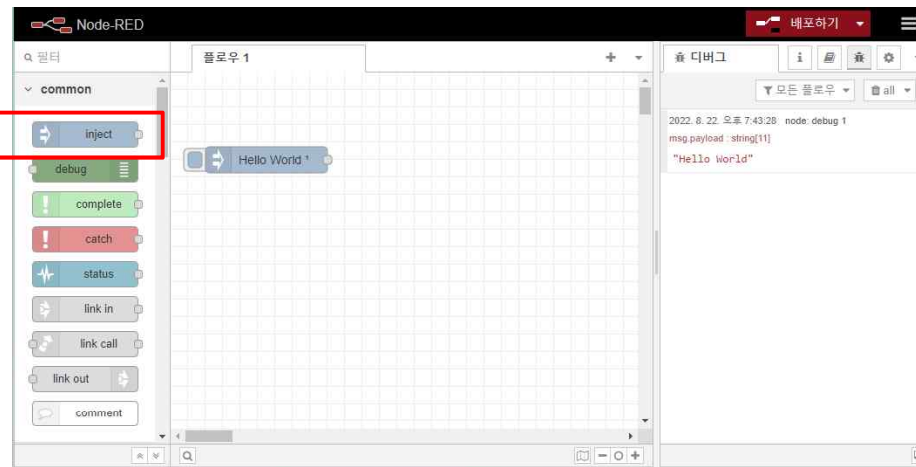
[실습2] Hello World 출력

- ◆ Node-RED에서 Payload String “Hello World” 출력

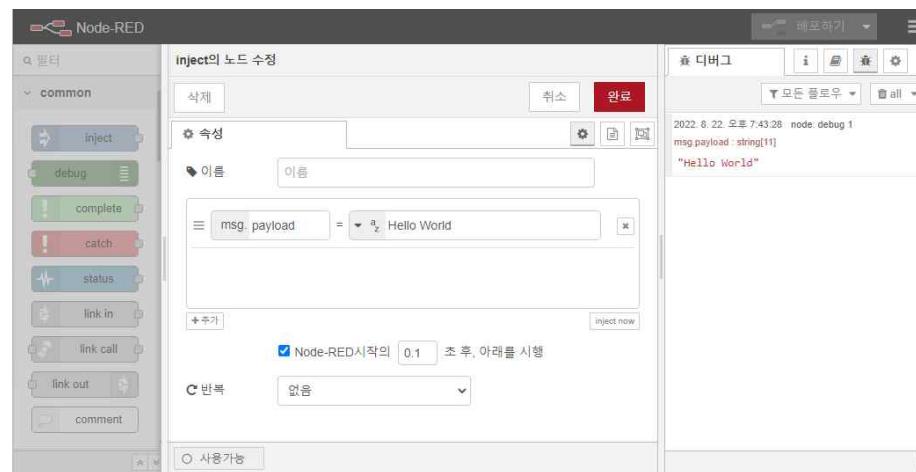


[실습2] Hello World 출력 : Flow 편집(1)

① 왼쪽 메뉴(노드탭)에서 Inject 노드를 드래그 앤 드롭하여 플로우 에디터로 사용

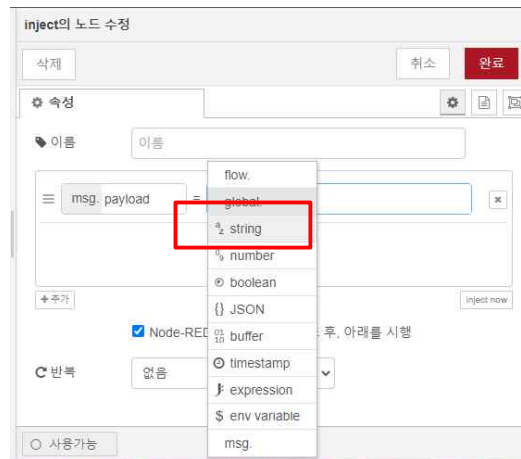


② 가져온 Inject 노드를 더블 클릭하여 아래와 같이 Edit inject node 탭으로 이동



[실습2] Hello World 출력 : Flow 편집(2)

③ Payload의 timestamp를 클릭한 후 string으로 변경

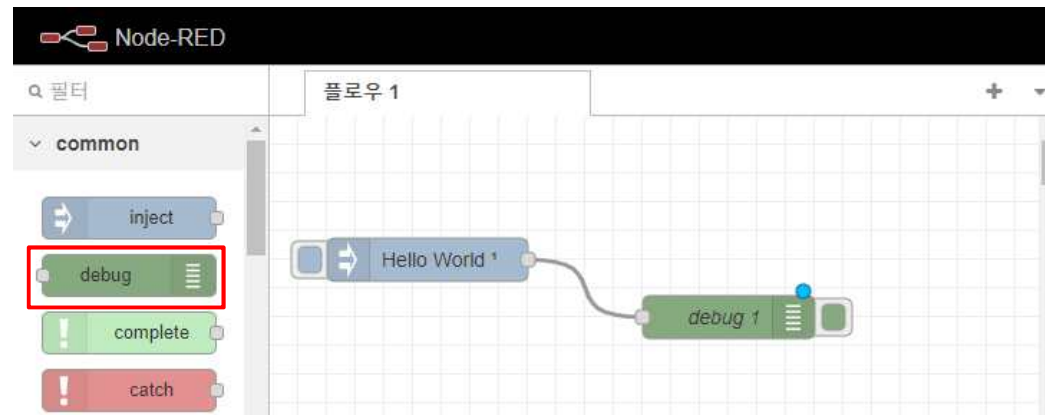


④ 텍스트 박스에 출력하고자 하는 문자를 입력 Payload(String) : “Hello world”

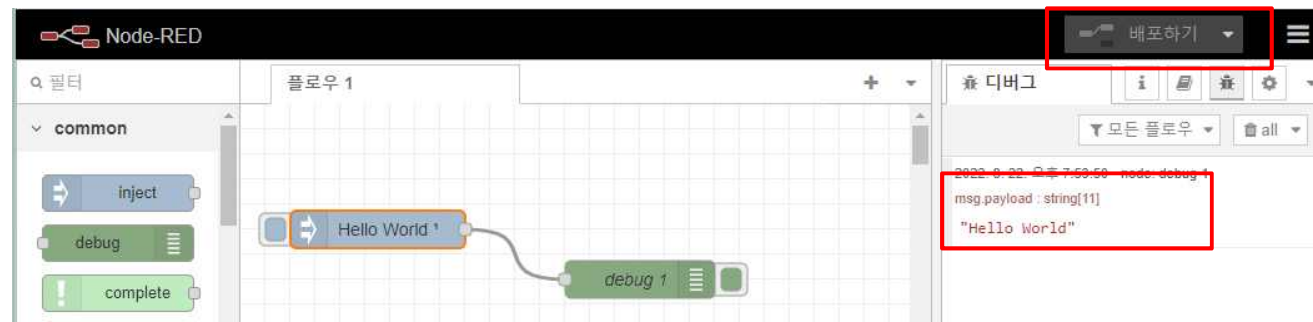


[실습2] Hello World 출력 : Flow 편집(2)

- ⑤ 왼쪽 노드 탭에서 debug 노드를 드래그 앤 드롭하여 플로우 에디터로 가져와 각 노드를 연결

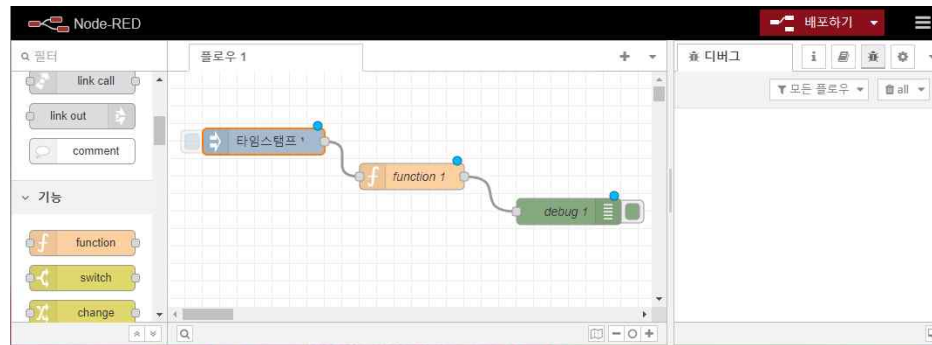


- ⑥ 상단 탭의 배포(Deploy) 버튼을 클릭하고 디버그 내용을 확인하기 위하여 오른쪽 상단의 Debug(디버그 메시지) 탭을 클릭(수정 및 업데이트 시 항상 배포하기 버튼을 클릭) 마지막으로 Inject 노드(Hello World)의 왼쪽 실행 버튼을 클릭하면 디버그 창에 메시지가 출력 되는 것을 확인 가능

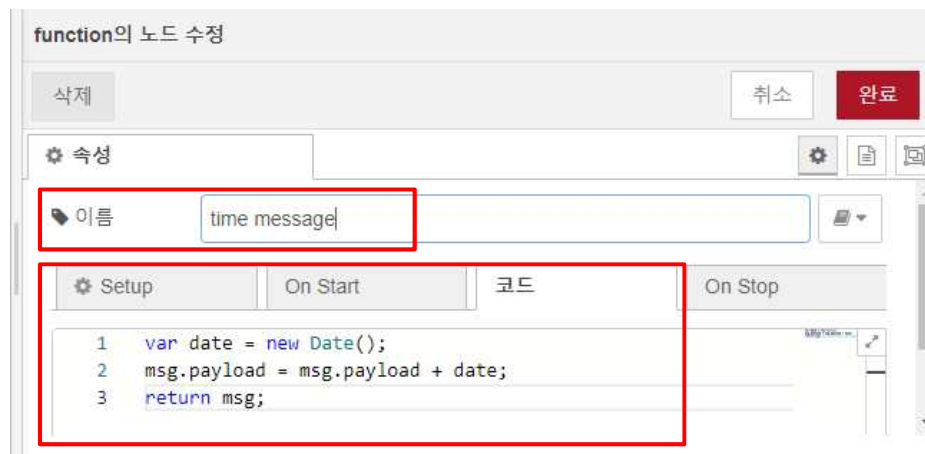


[실습3] 현재 시간출력하기 (1)

- ① Inject 노드와 디버그 노드를 추가하고 아래와 같이 function(기능)항목의 function 노드 추가 및 각 노드를 연결



- ② function 노드를 더블 클릭하여 아래와 같이 작성한 후 완료(Done) 버튼을 클릭



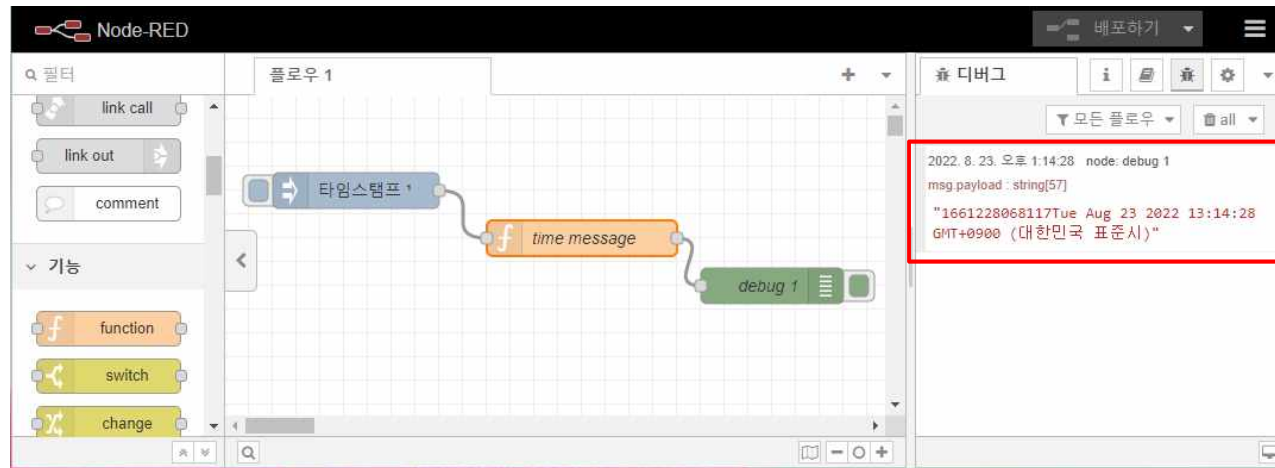
Name : time message

Function :

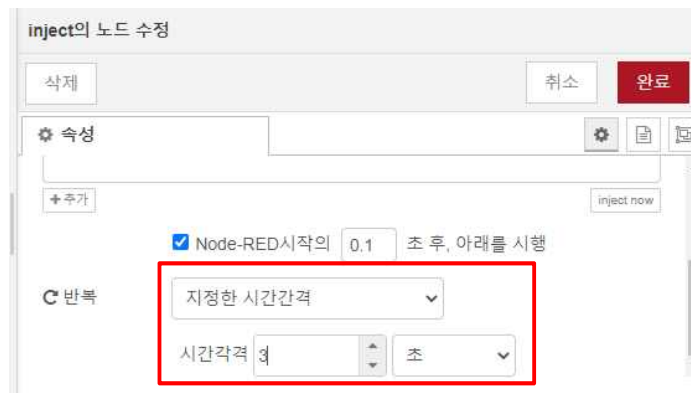
```
var date = new Date();
msg.payload = msg.payload + date;
return msg;
```

[실습3] 현재 시간출력하기(2)

- ③ 배포(Deploy) 버튼을 클릭하고 실행하면(Inject노드) Inject에서 넘어 온 payload데이터와 Date() 정보가 합쳐져 return된 메시지가 Debug 노드를 통해 출력이 되는 것을 확인

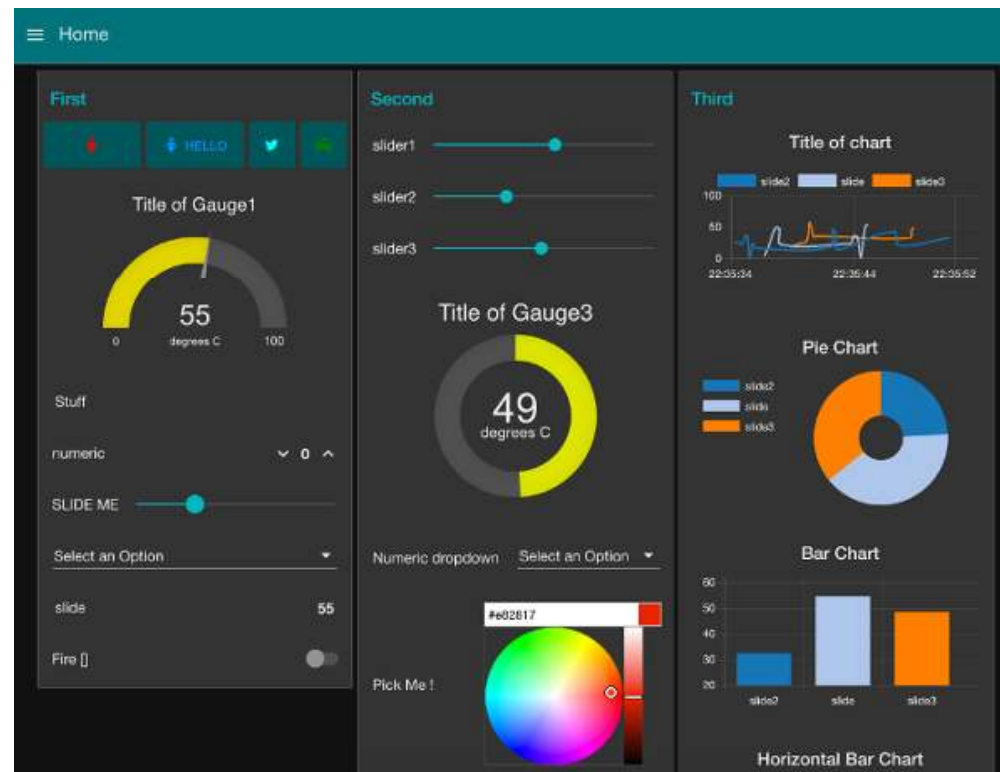


- ④ 사용자가 특정시간 루프를 지정하여 연속적으로 데이터를 출력, Inject 노드의 속성에서 Repeat(반복)를 Interval(지정한 시간 간격)로 설정하고 시간주기는 3초로 설정



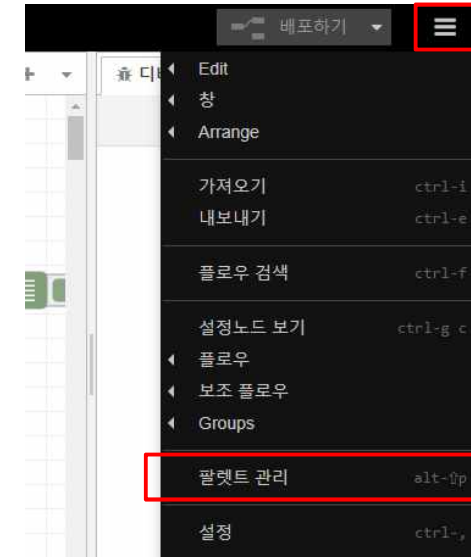
[실습4] 그래픽 사용자 인터페이스 사용하기 (1)

- ◆ Dashboard는 중요한 정보나 측정항목을 다양한 위젯을 통해 나타내주는 도구이며 다수의 측정데이터 항목을 한 번에 모니터링 할 수 있으며, 이를 통해 데이터의 변화 분석이 가능

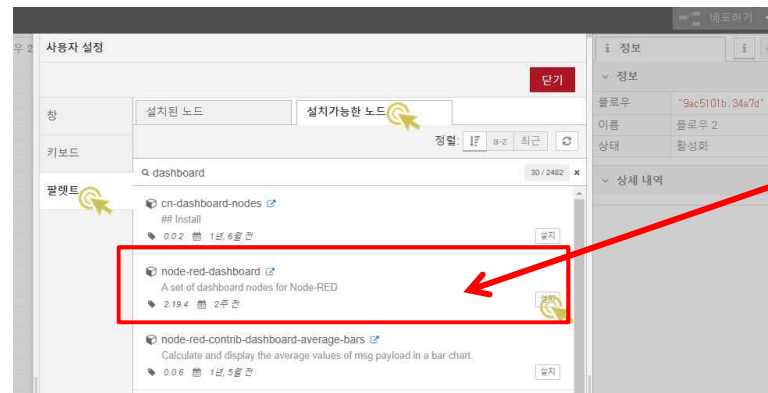


[실습4] 그래픽 사용자 인터페이스 사용하기 (2)

- ① Dashboard를 사용하기 위해서는 추가 Node 설치가 필요
Node-red 오른쪽 상단 메뉴의
“Manage Palette(팔레트 관리)” 를 클릭



- ② 사용자 설정 (User settings) 화면으로 이동하여
“팔레트(Palette)” > “설치 가능한 노드(Install)” 탭을 선택하고,
필터에 ‘dashboard’ 라고 검색 후 해당 팔레트 설치 화면이 나타나면 설치(install)



node-red-dashboard

[실습4] 그래픽 사용자 인터페이스 사용하기 (3)

- ※ “Manage Palette(팔레트 관리)” 를 클릭시 아래 오류 발생하면
명령 프롬프트 창에서 아래 명령으로 별도 설치
(설치 후 node-red 재실행)

```
npm i node-red-dashboard
```

- ※ node-red-dashboard 관련 정보

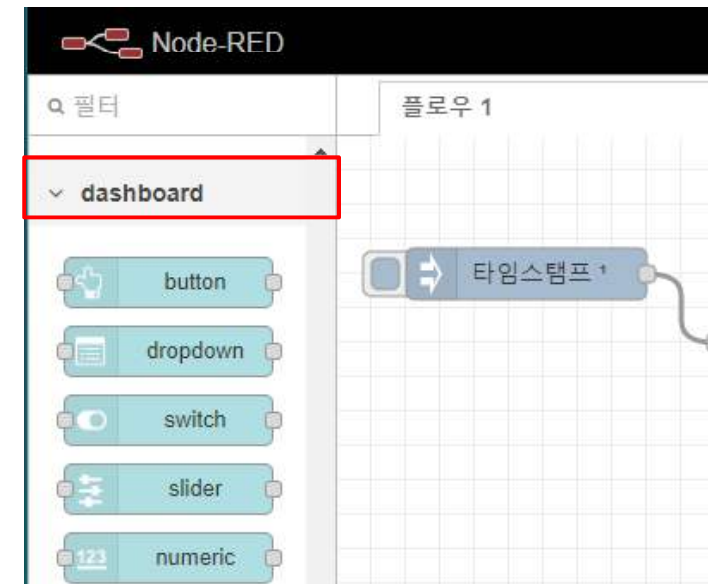
<https://flows.nodered.org/node/node-red-dashboard>

<https://github.com/node-red/node-red-dashboard>

- ③ 배포Dashboard 팔레트 설치가 완료되면
화면 왼쪽 팔레트에 설치된 ‘dashboard’
항목 표시 확인

Failed to load node catalogue.
Check the browser console for more information

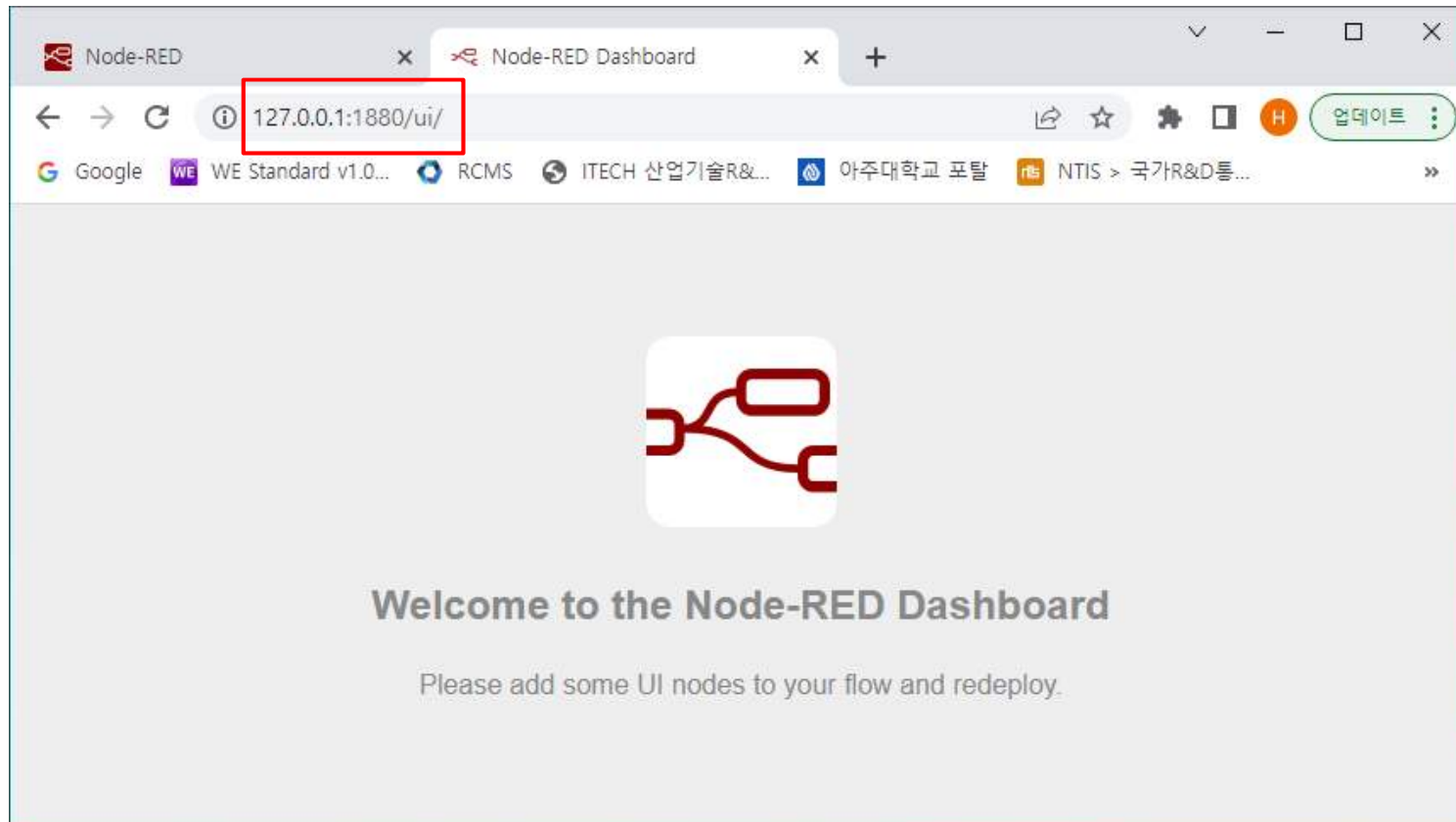
노드 카탈로그를 설치하지 못했습니다.
브라우저 콘솔로그를 참고하세요.



[실습4] 그래픽 사용자 인터페이스 사용하기 (4)

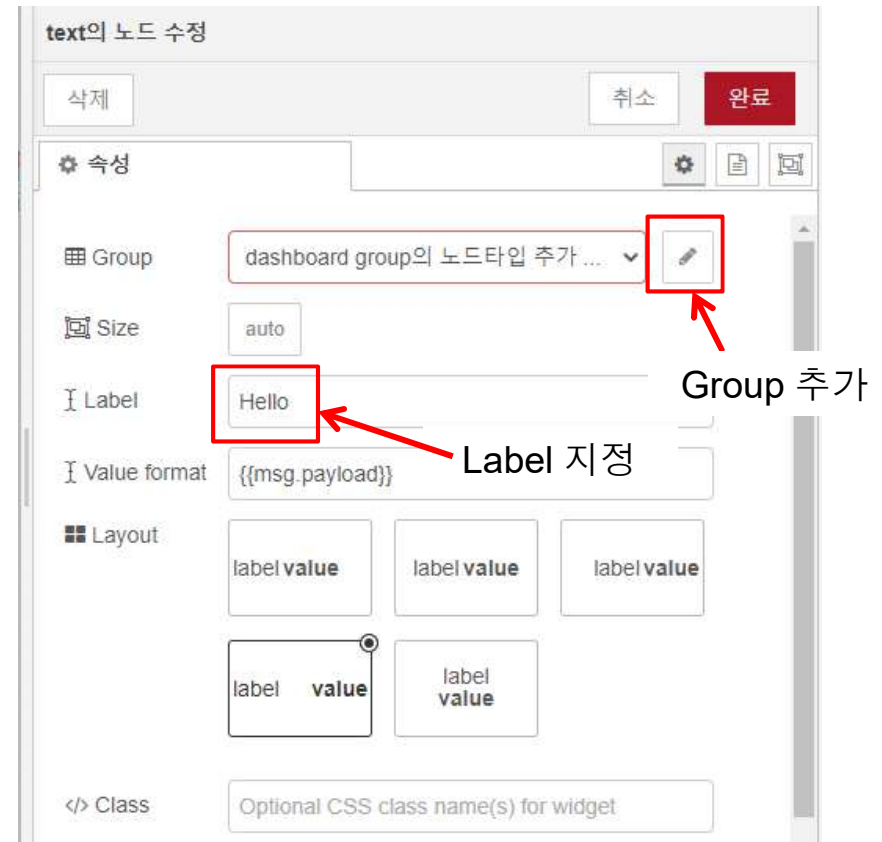
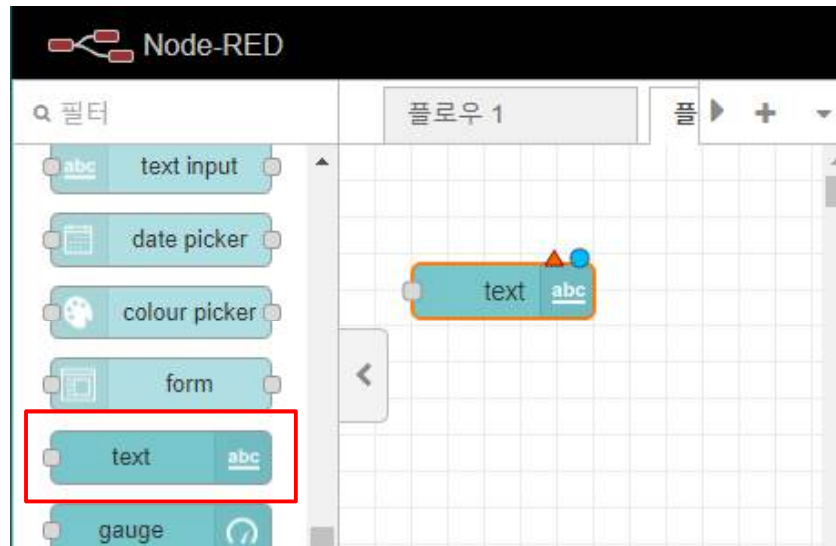
④ dashboard 팔레트가 생성되면 그래픽 사용자 인터페이스를 살펴볼 수 있는 별도의 UI페이지가 활성화 된다.

- 접속 주소는 기존 접속주소 뒤에 /ui를 추가 UI주소 : <http://localhost:1880/ui>



[실습4] 그래픽 사용자 인터페이스 사용하기 (5)

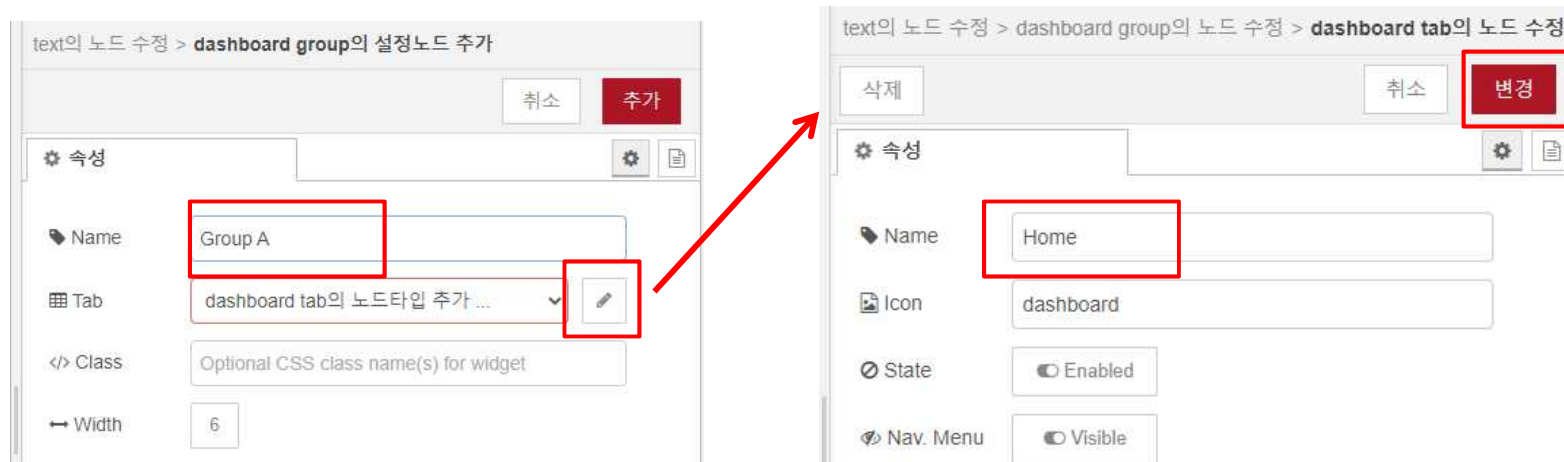
- ⑤ Node-red 개발 페이지로 이동하여 Dashboard 기본 노드를 사용해 보도록 한다.
- dashboard의 'text' 노드를 추가하고 dashboard의 속성을 확인



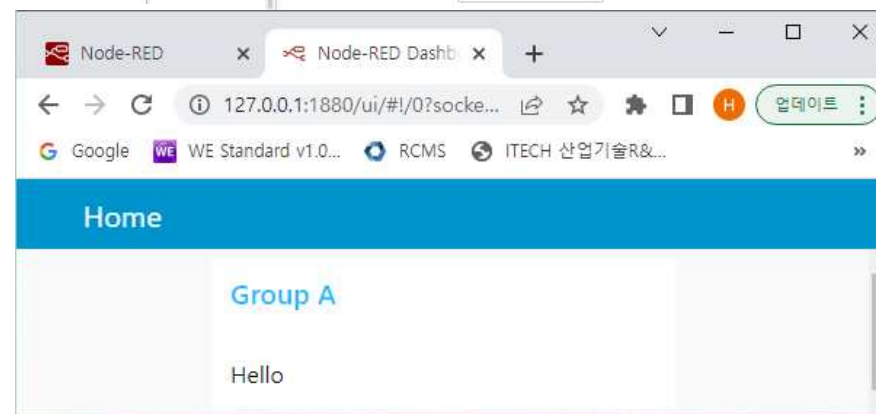
[실습4] 그래픽 사용자 인터페이스 사용하기 (6)

⑥ Dashboard의 위젯을 사용하기 위해서는 몇 가지 속성 지정이 필요 함.

- 기본 UI는 각각의 메뉴 (Home)를 의미하는 Tab화면과 한 개의 Tab에 다수의 Group 존재
- Text 노드의 Group을 지정(Default)하고 완료 버튼 클릭 및 배포하기(Deploy) 클릭



- UI화면으로 이동하여 확인

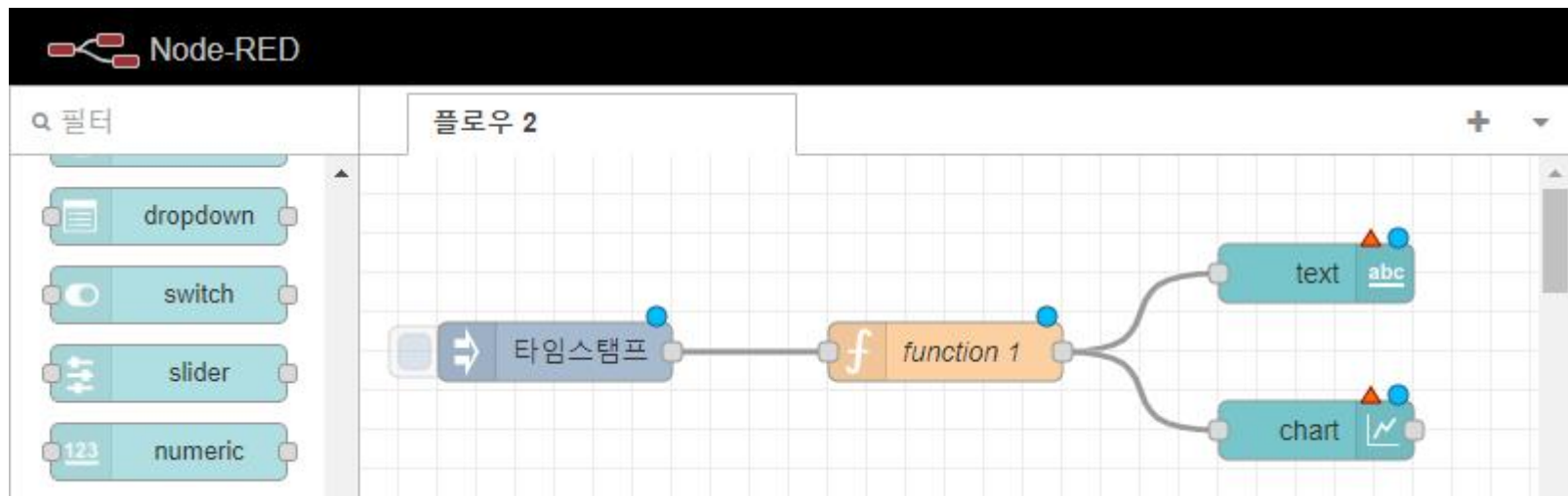


[실습5] 그래픽 UI 차트 활용 (1)

◆ Dashboard를 이용하여 1부터 100까지 임의의 값 출력 실습

① 노드 추가

- 주기적으로 데이터를 반복하기 위한 inject 노드 추가
- 임의의 값을 주기 위한 함수 노드 추가
- dashboard에 출력하기 위한 text, chart 노드 추가

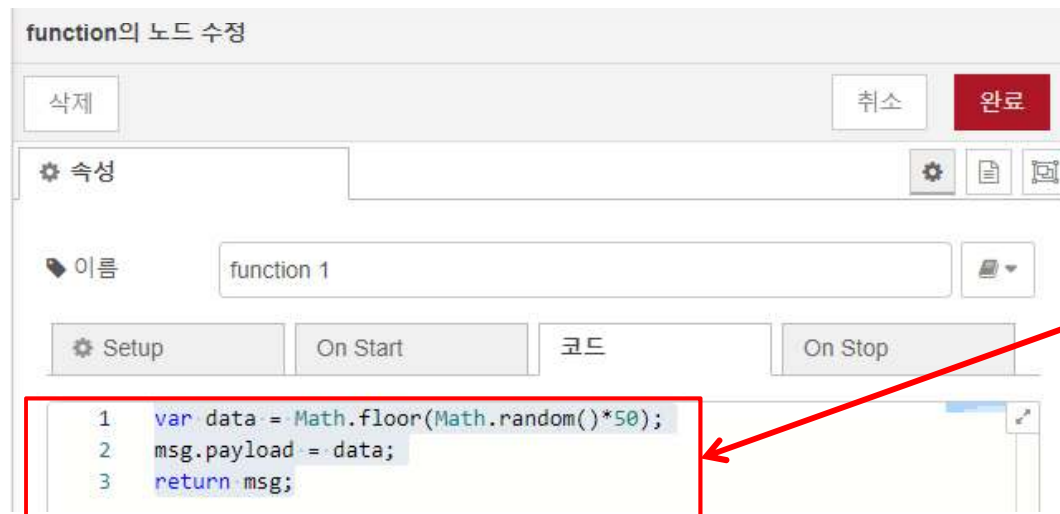


[실습5] 그래픽 UI 차트 활용 (2)

- ② Inject노드(타임스탬프)에서 주기적으로 데이터를 반복하기 위해서 지정한 시간간격반복 설정 (2초 간격 반복)



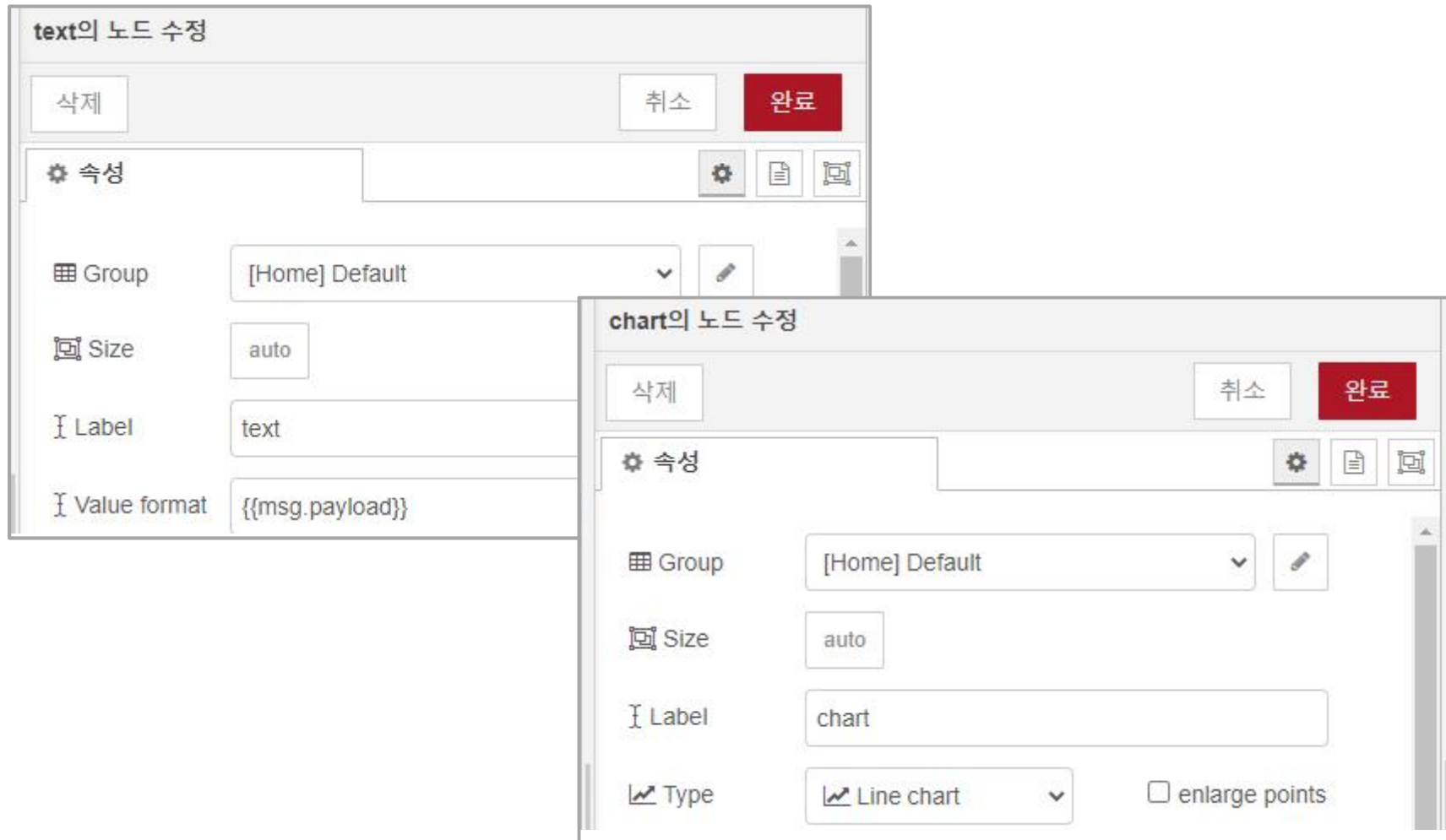
- ③ Dashboard를 이용하여 1부터 100까지 임의의 값을 출력하기 위해서 Function노드에 코드를 추가



```
var data = Math.floor(Math.random()*50);  
msg.payload = data;  
return msg;
```

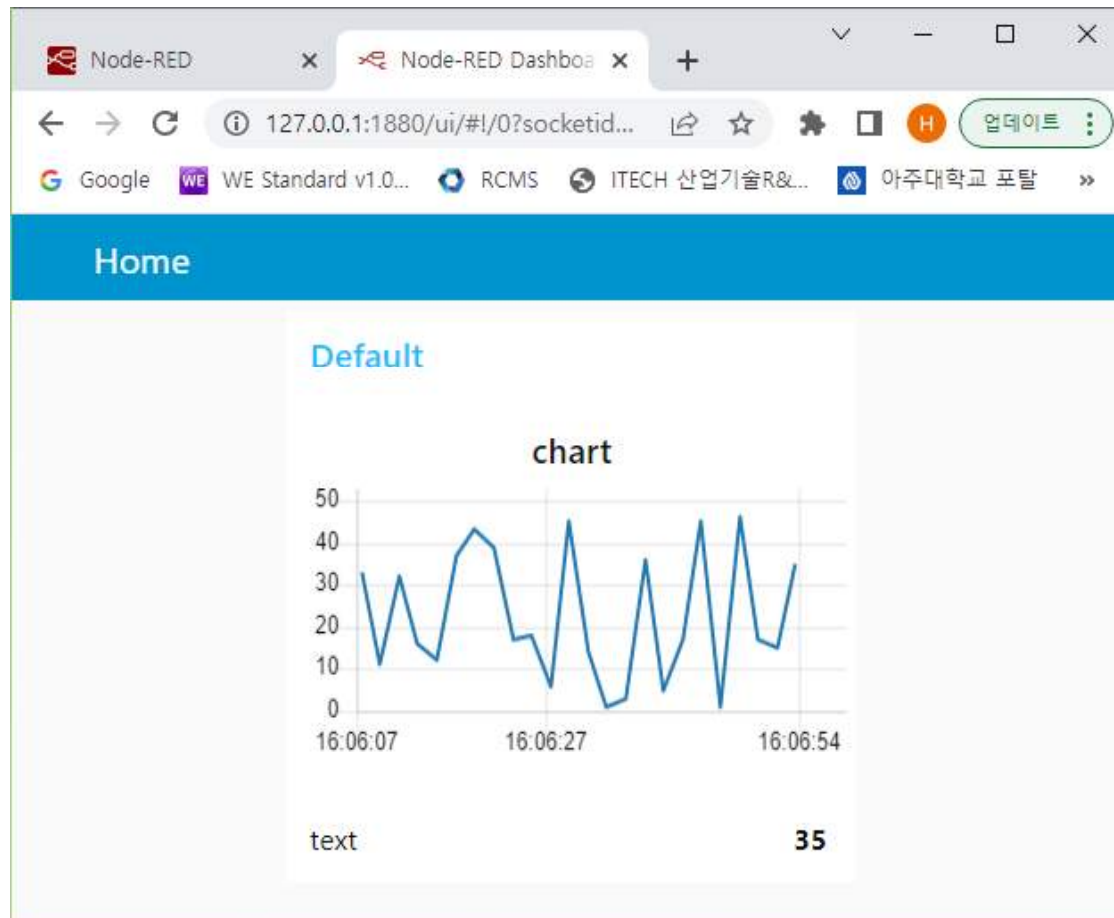
[실습5] 그래픽 UI 차트 활용 (3)

④ 데이터를 시각화하기 위한 text노드와 chart노드를 추가하고 속성에서 Group을 지정



[실습5] 그래픽 UI 차트 활용 (4)

- ⑤ 모든 설정이 완료되면 배포를 클릭하고 UI페이지로 이동하여 화면을 확인



목 차

IoT 표준화

Application Layer Protocol

IoT 시각화 툴 Node-RED

◆ **IoT 서비스**

Machine Learning

Internet-of-Things (IoT, 사물인터넷)

IoT?

Internet of Things

사물인터넷

Internet of Everything

Web of Things

Embedded Web

Machine to Machine

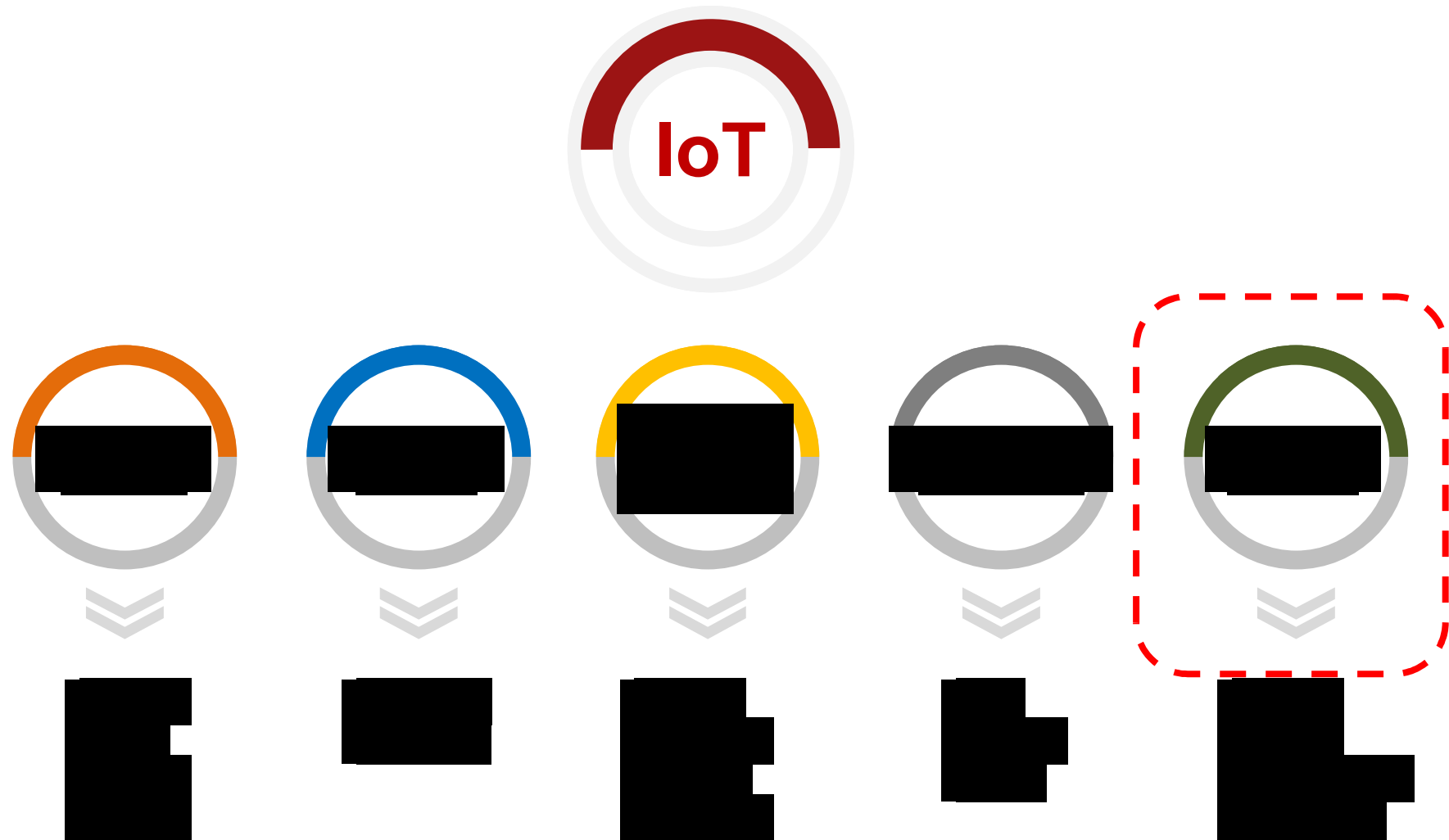
Industry 4.0

인터넷을 기반으로 모든 사물을 연결하여 사람과 사물,
사물과 사물간의 정보를 상호 소통하는

지능형 기술 및 서비스

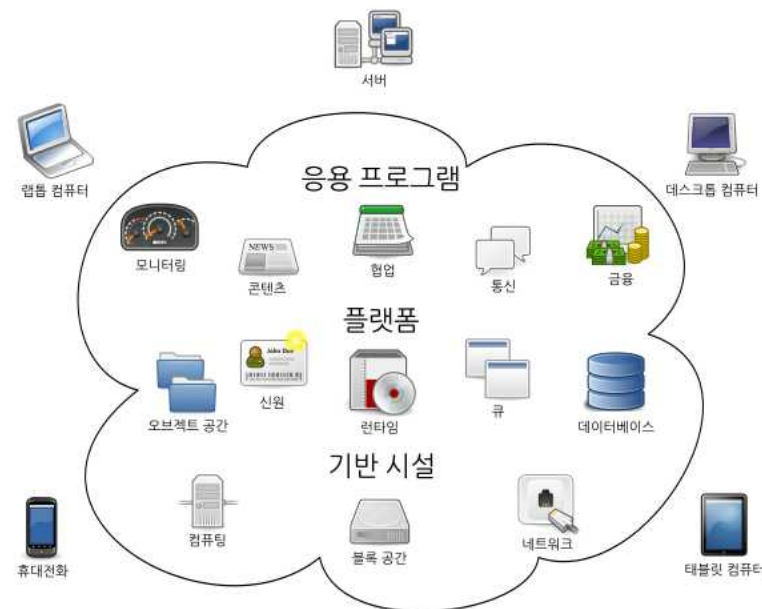
기능화 + 상호연결 + 지능화 = IoT

Internet-of-Things 요소 기술



Cloud Computing

- ◆ 네트워크 환경(구름)에서 원하는 작업을 요청하여 실행한다는 데서 기원
 - 인터넷 기술을 활용하여 IT자원을 서비스로 제공하는 컴퓨팅을 뜻
- ◆ 네트워크, 스토리지, 서버, 서비스 및 애플리케이션 등의 IT 자원을 소유하던 전통적인 방식(On-premises)에서 벗어나, 인터넷에 연결된 다양한 모바일 디바이스를 활용하여 서비스 형태(On-demand)로 이용하는 컴퓨팅 방식



Cloud Computing - Service

- IT자원의 서비스 종류에 따라 인프라 서비스(IaaS, Infrastructure as a Service), 플랫폼 서비스(PaaS, Platform as a Service), 응용 소프트웨어 서비스(SaaS, Software as a Service)로 구분



Cloud Computing – Service 종류

- ◆ 컴퓨팅, 스토리지, 데이터베이스, 분석, 네트워킹, 모바일, 개발자 도구, 관리 도구, IoT, 보안 및 엔터프라이즈 애플리케이션을 비롯하여 광범위한 글로벌 클라우드 기반 제품을 제공



컴퓨팅



스토리지



데이터베이스



마이그레이션



네트워킹 및 콘텐츠 전송



개발자 도구



관리 도구



미디어 서비스



보안, 자격 증명 및 규정 준수



분석



Machine Learning



모바일 서비스



증강현실 및 가상현실



애플리케이션 통합



고객 인게이지먼트



비즈니스 생산성



데스크톱 및 앱 스트리밍



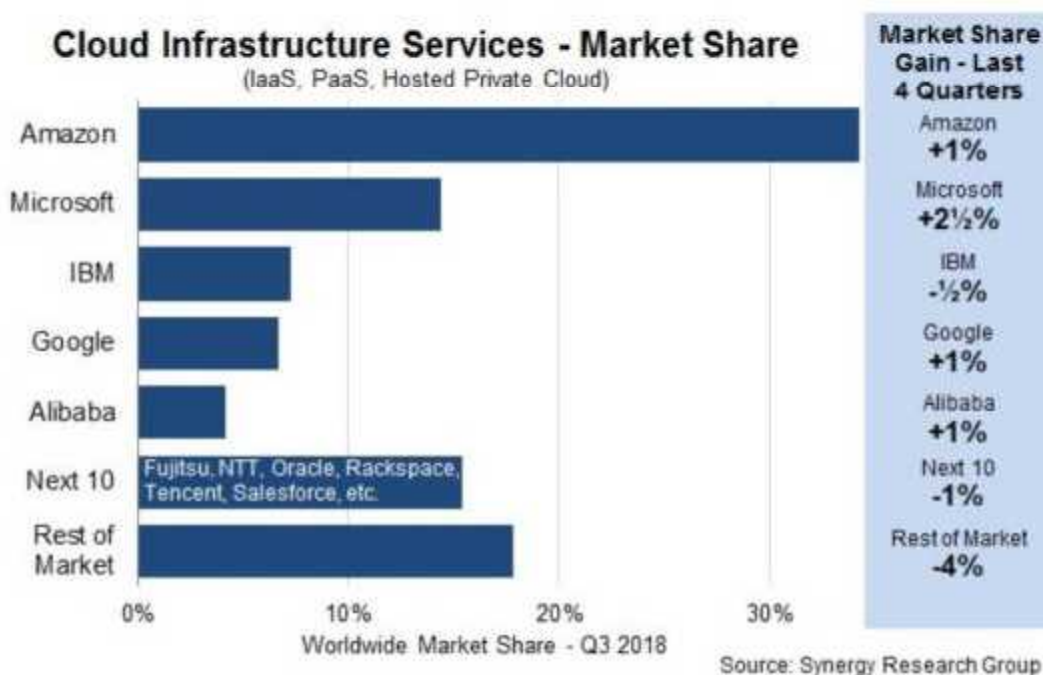
사물 인터넷



게임 개발

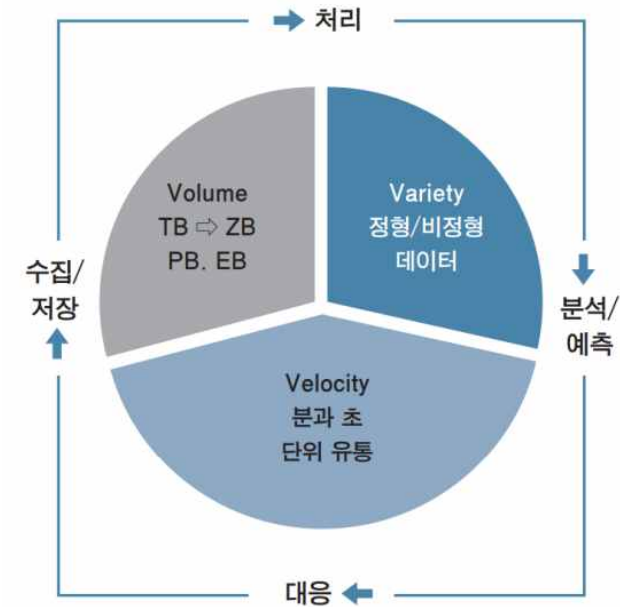
Cloud Computing – Service 종류

- ◆ 전 세계적으로 ICT 자원 절감, 업무혁신을 위해 정보시스템을 자체 구축, 사용하는 방식에서 서비스 형태로 이용하는 클라우드 컴퓨팅이 확산
- ◆ 세계 클라우드 시장은 아마존, MS 등 세계 주요선도 클라우드 선도기업이 위치하고 있는 미국 시장 및 기업을 중심으로 시장을 주도 중이며, 적극적인 정부육성 의지와 기업의 투자를 추진하고 있는 중국 클라우드 시장이 급성장 중



BigData

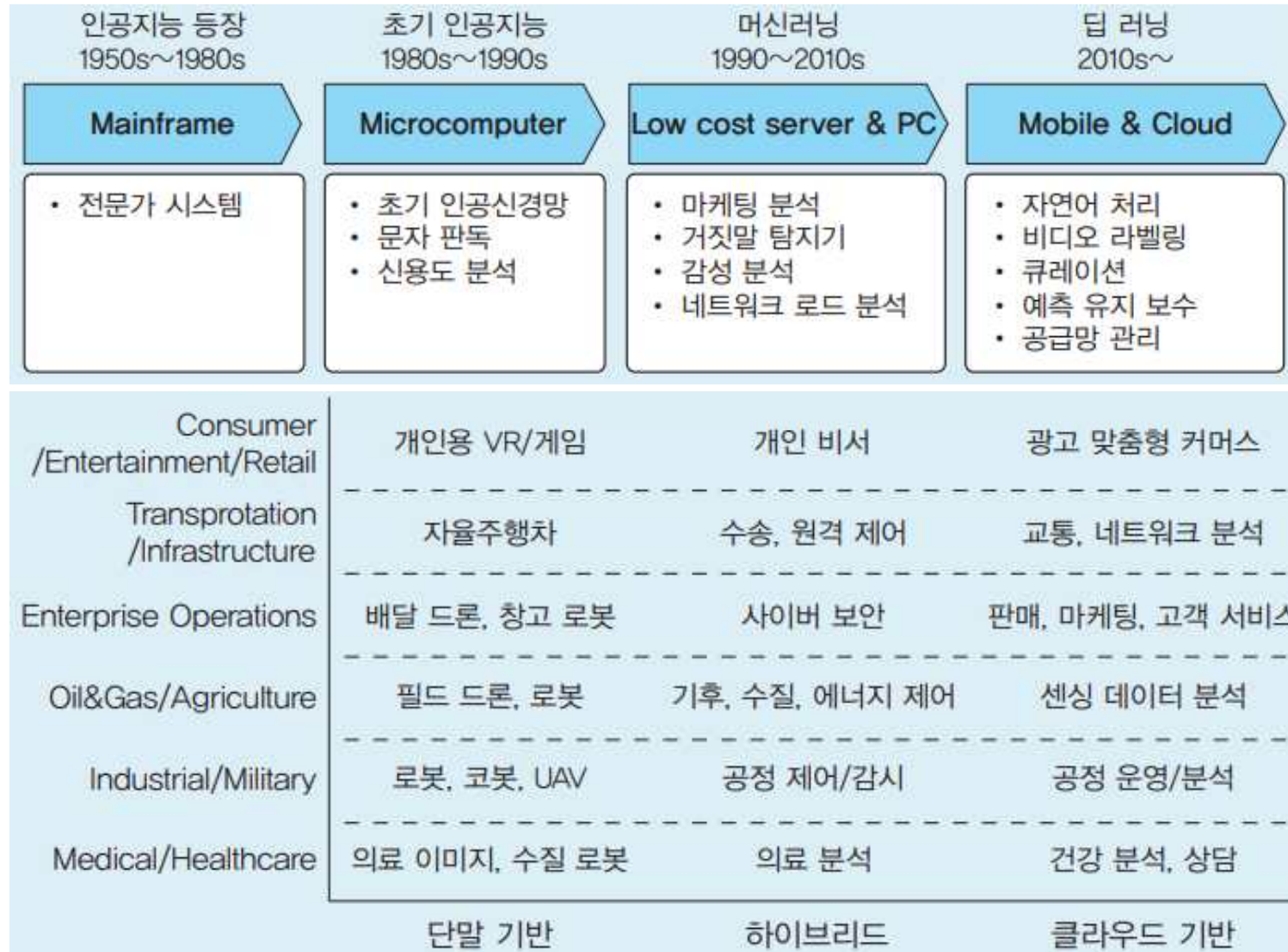
- ◆ 디지털 환경에서 생성되는 자원으로 그 규모가 방대하고, 생성 주기도 짧고, 형태도 수치 데이터뿐 아니라 문자와 영상 데이터를 포함하는 대규모 데이터
- ◆ 빅데이터는 데이터의 양(Volume), 데이터 생성 속도(Velocity), 다양성(Variety) 등 세 가지로 정의한
- ◆ 데이터를 분석하고 처리함으로써 기존의 데이터에서 볼 수 없었던 새로운 의미를 산출하는데 목표
- ◆ 이를 위해 텍스트마이닝과 웹마이닝, 소셜마이닝 작업이 이루어짐



데이터셋 활용사례 참여마당 정보공유



인공지능 활용



On-Device Learning

◆ AI 응용 구동 하드웨어

- 클라우드
- Desktop
- Mobile phone
- IoT

◆ On-Device learning의 필요성

- Low latency
- Privacy (keep data on-device)
- Works offline

◆ <https://developers.google.com/learn/topics/on-device-ml>

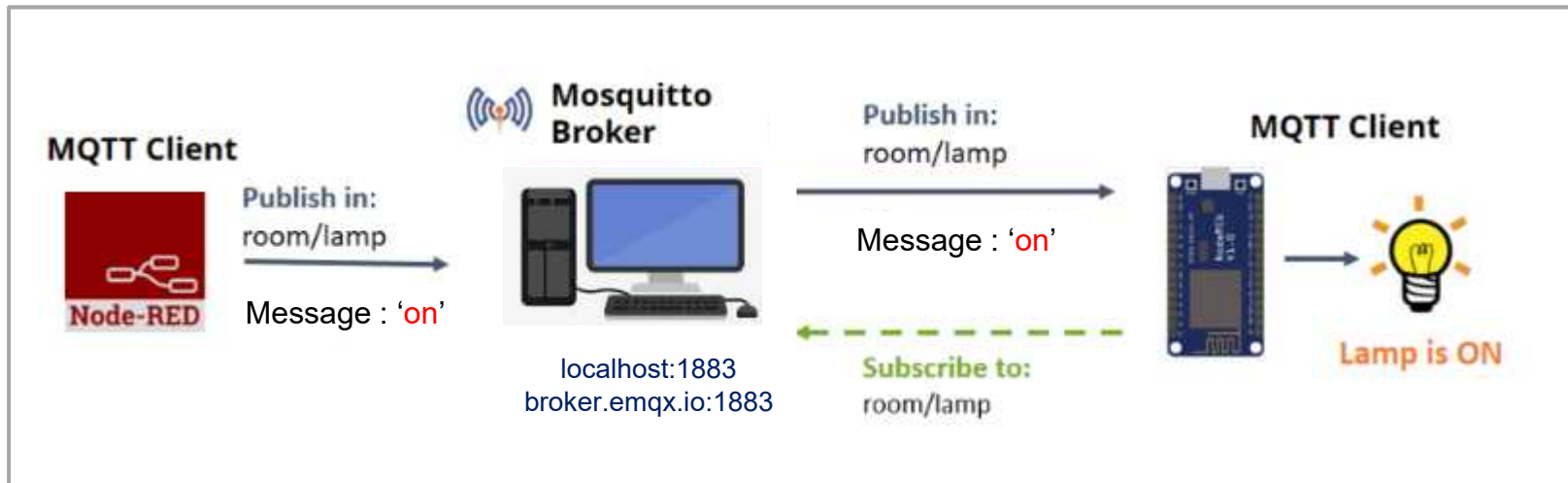


[실습6] LED On/Off with Node-RED & MQTT

1. 작업관리자에서 mosquitto 서비스 실행
2. 명령어 창에서 mosquitto subscriber 실행
 - 명령 : `mosquitto_sub -h localhost -p 1883 -t "room/lamp"`
 - ❖ IP 주소 : localhost, 포트번호 : 1883, topic : room/lamp
3. 명령어 창에서 Node-RED 실행
4. Chrome Browser 실행
5. Node-RED 워크스페이스에서 Publisher node와 on/off inject node 추가
 - 배포 실행 후 명령어(CMD) 창의 mosquitto subscriber 메시지 확인
6. Node-RED 워크스페이스에서 Subscriber node와 Debug node 추가
 - 배포 실행 후 Node-RED 디버그 창의 mosquitto subscriber 메시지 확인
 - 배포 실행 후 명령어(CMD) 창의 mosquitto subscriber 메시지 확인
7. LED 구동 ESP32 아두이노 하드웨어 동작 시험
 - ESP32 기반의 LED 구동 회로 구성
 - MQTT 기반 LED 제어 아두이노 SW 개발 및 배포

[실습6] 실습 환경 구성

- ◆ Node-RED와 MQTT를 이용한 LED ON/OFF 제어 환경
 - Node-RED
 - Broker
 - ❖ localhost:1883 (Windows PC Mosquitto Broker)
 - ❖ broker.emqx.io:1883 (Public Broker)
 - ESP32 Target



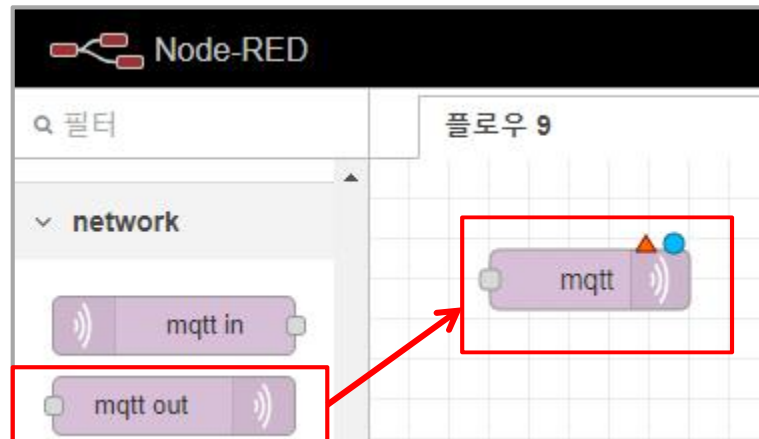
[실습6] Node-RED를 이용한 MQTT Publishing (1)

◆ Node-RED에서 MQTT Publish node를 이용한 Publishing 구현

- 클라이언트에서 Broker로 Message(메시지)를 Publish
- Publish를 위해 필요한 전송 정보
 - ❖ Message Topic
 - ❖ Message QoS
 - ❖ Retain Flag : Message 유지 여부
 - False : Broker가 Message를 유지하지 않음
 - True : Retained flag가 설정된 Message를 유지

◆ Node-RED에서 MQTT Publish node 추가

- Node-RED의 “node palette”에서 MQTT publish(mqtt out) node를 Workspace로 드래그하여 설치



[실습6] Node-RED를 이용한 MQTT Publishing (2)

◆ MQTT Publish node 수정

- Publish Node를 더블클릭 후 수정
- 수정 항목
 - ❖ 서버 : MQTT Broker의 주소, (예 : localhost)
 - ❖ 토픽 : Publish와 Subscriber가 사용할 채널 (예: root/lamp)
 - ❖ QoS : QoS 레벨 설정 (예 : 2)
 - ❖ 보존(Retain Flag) : 메시지 보존 여부
 - ❖ 이름 : 기입하지 않음

mqtt out의 노드 수정

삭제 취소 완료

속성

서버 localhost:1883

토픽 room/lamp

QoS 2 보존 하지않는

이름 이름

주석: 토픽이나 QoS를 메시지의 프로퍼티를 사용하여 설정하는 경우에는, 기입하지 않습니다.

mqtt out의 노드 수정 > mqtt-broker의 노드 수정

삭제 취소 변경

속성

이름 이름

접속

서버 localhost 포트 1883

☒ Connect automatically

☐ 사용TLS

Protocol MQTT V3.1.1

클라이언트 ID를 자동생성하는 경우에는, 기입하지 않습니다

keep alive 시간 60

Session ☒ 세션 초기화

[실습6] Node-RED를 이용한 MQTT Publishing (3)

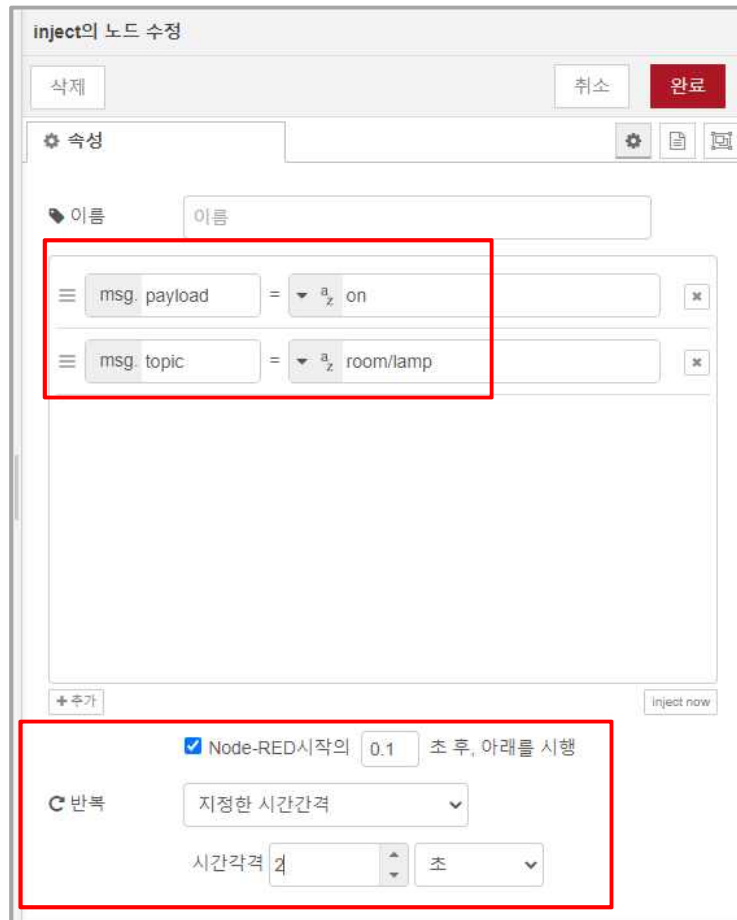
◆ 2개의 “inject node” 추가

- “on” 메시지를 MQTT topic “room/lamp”에 전송
 - ❖ Node-RED 시작 0.1초 후부터 2초 간격으로 on 메시지 전송
- “off” 메시지를 MQTT topic “room/amp”에 전송
 - ❖ Node-RED 시작 1.1초 후부터 2초 간격으로 off 메시지 전송



[실습6] Node-RED를 이용한 MQTT Publishing (4)

- ◆ 2개의 “inject node” 를 더블 클릭하여 topic과 payload(메시지) 지정
 - 주기적으로 payload on/off를 room/lamp topic으로 전송



inject의 노드 수정

삭제 취소 완료

속성

이름 이름

msg. payload = a_z on

msg. topic = a_z room/lamp

+ 추가 inject now

☒ Node-RED시작의 0.1 초 후, 아래를 시행

반복 지정한 시간간격

시간각격 2 초



inject의 노드 수정

삭제 취소 완료

속성

이름 이름

msg. payload = a_z off

msg. topic = a_z room/lamp

+ 추가 inject now

☒ Node-RED시작의 1.1 초 후, 아래를 시행

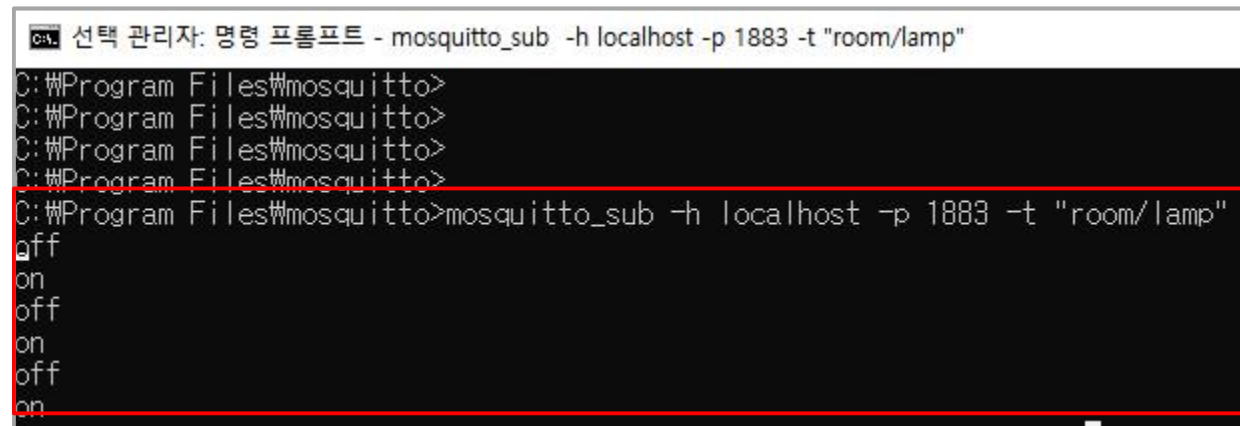
반복 지정한 시간간격

시간각격 2 초

[실습6] Node-RED MQTT Subscribing (1)

◆ 명령어 창에서 Mosquitto subscribe 명령 실행하여 확인

- 명령 : `mosquitto_sub -h localhost -p 1883 -t "room/lamp"`
 - ❖ IP 주소 : localhost
 - ❖ 포트번호 : 1883
 - ❖ topic : room/lamp
- 'room/lamp' topic에서 주기적으로 payload 'on/off' 수신 확인

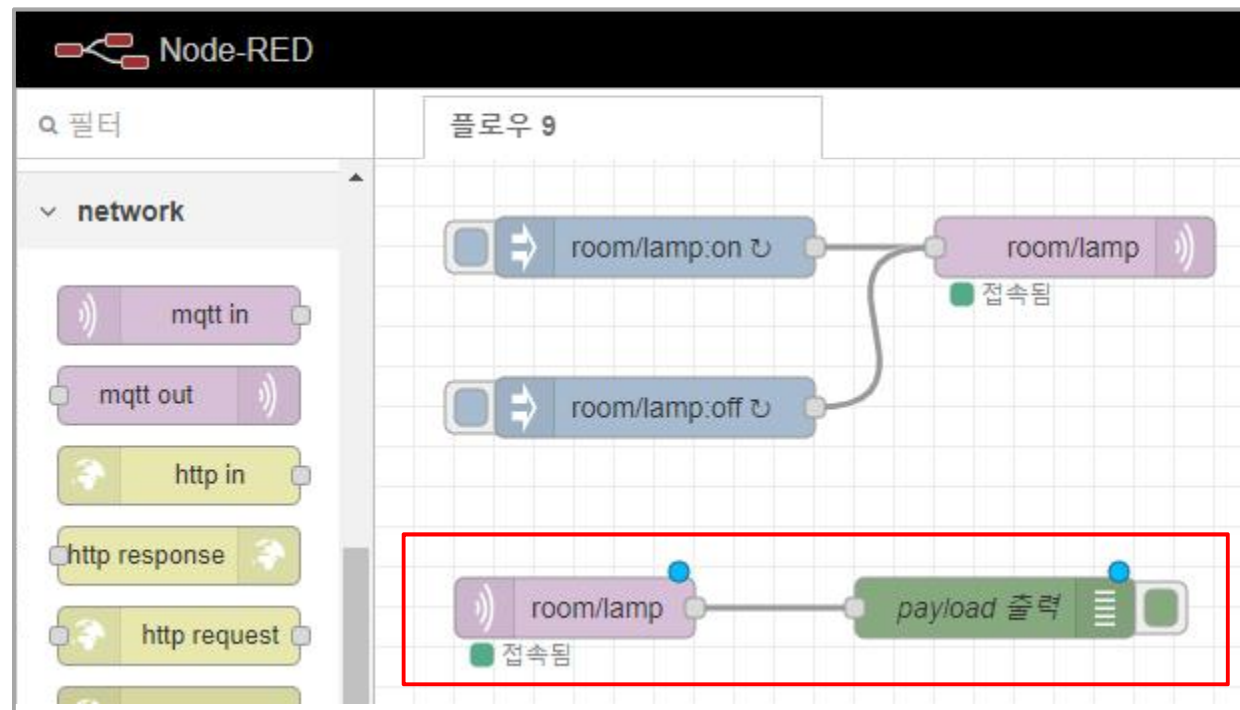


```
CA. 선택 관리자: 명령 프롬프트 - mosquitto_sub -h localhost -p 1883 -t "room/lamp"
C:\Program Files\mosquitto>
C:\Program Files\mosquitto>
C:\Program Files\mosquitto>
C:\Program Files\mosquitto>
C:\Program Files\mosquitto>mosquitto_sub -h localhost -p 1883 -t "room/lamp"
off
on
off
on
off
on
```

[실습6] Node-RED MQTT Subscribing (2)

◆ Node-RED에서 MQTT Scribe node 추가

- Node-RED의 “node palette”에서 MQTT subscribe(mqtt in) node를 Workspace로 드래그하여 설치
- “debug node”를 드래그하여 설치 후 subscribe 노드와 연결



[실습6] Node-RED MQTT Subscribing (3)

◆ MQTT Subscribe node 수정

- Subscribe Node를 더블클릭 후 수정
 - ❖ 서버 : MQTT Broker의 주소, (예 : localhost)
 - ❖ Action : Subscribe to single topic
 - ❖ 토픽 : Publish와 Subscriber가 사용할 채널 (예: root/lamp)
 - ❖ QoS : QoS 레벨 설정 (예 : 2)
 - ❖ 이름 : 기입하지 않음

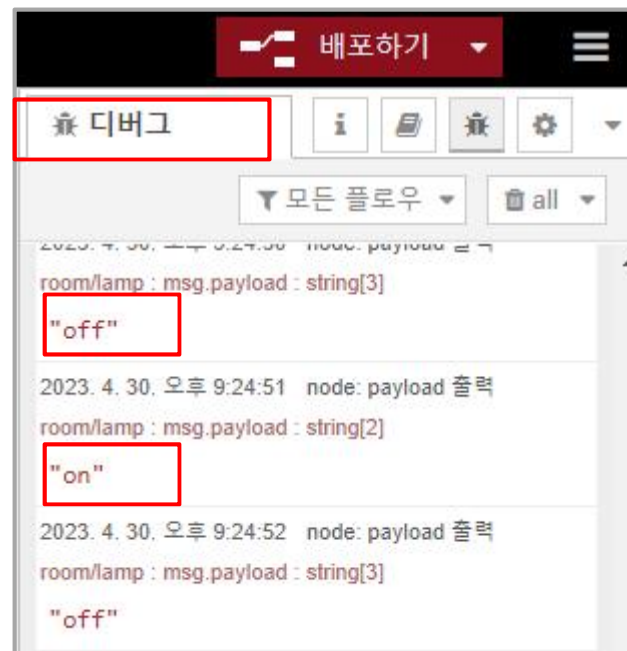
◆ Debug Node 수정

The screenshot shows the 'debug' node configuration panel. At the top, there are buttons for '삭제' (Delete), '취소' (Cancel), and '완료' (Done). Below these is a '속성' (Properties) tab. Under '속성', there is a '대상' (Target) dropdown set to 'msg. payload'. Below that, there is a '출력대상' (Output target) section with a checked checkbox for '디버그 창' (Debug window) and unchecked checkboxes for '시스템 콘솔' (System console) and '노드 상태(32자)' (Node status (32 characters)). At the bottom, there is an '이름' (Name) field containing the text 'payload 출력'.

The screenshot shows the 'mqtt in' node configuration panel. At the top, there are buttons for '삭제' (Delete), '취소' (Cancel), and '완료' (Done). Below these is a '속성' (Properties) tab. Under '속성', there are several fields: '서버' (Server) set to 'localhost:1883', 'Action' set to 'Subscribe to single topic', '토픽' (Topic) set to 'room/lamp', 'QoS' set to '2', '출력' (Output) set to '자동판정(JSON오브젝트, 문자열혹은 바이너리)' (Automatic judgment (JSON object, string or binary)), and '이름' (Name) set to '이름'.

[실습6] Node-RED MQTT Subscribing (4)

- ◆ 배포(Deploy) 후 Node-RED 디버그 메시지 확인
 - “배포하기” 실행 후 디버그 창 확인



[실습6] ESP32 MQTT 동작 구현(1)

◆ Wokwi 환경에서 MQTT 접속 시험

- MQTT room/lamp 토픽에서
on 메시지 수신 시 LED ON
off 메시지 수신 시 LED OFF
- MQTT on/off 메시지 Publish는
Node-RED 환경에서 구동
- MQTT Broker 변경
 - ❖ Wokwi 사용시 내부망을 사용할 수 없기 때문에
Open MQTT Broker 사용(broker.emqx.io)
 - ❖ Node-RED의 MQTT Broker도 변경

◆ Node-RED에서 room/lamp 토픽으로 2초마다 on/off message를 publish

- 단, 시작 시점은 1초 차이가 있기 때문에
LED blink 동작



참고 : Wokwi ESP32 WiFi Network

- ◆ Wokwi WiFi 네트워크는 모든 인터넷 액세스 지원 가능
- ◆ 주요 사용 예
 - MQTT 서버 접속 및 publish/subscribe
 - HTTP, HTTPD 및 웹 socket 지원
 - HTTP 서버 구동 및 웹접속 (별도 Wokwi 필요)
- ◆ 주의 사항
 - Wokwi 사용시 기본적으로 Public Gateway 사용 됨 (IP :10.10.0.2로 설정)
 - 로컬 네트워크 사용시 별도 Private Wokwi IoT Gateway 필요함 (IP:10.13.37.2로 설정)
 - ❖ <https://github.com/wokwi/wokwigw/releases/tag/v1.1.0>
 - Private Wokwi Gateway를 사용하기 위해서는 Club 멤버이어야 함

[실습6] ESP32 MQTT 동작 구현(2)

```
#include <WiFi.h>
#include "PubSubClient.h"
```

```
const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* mqttServer = "broker.emqx.io";
int port = 1883;
String stMac;
char mac[50];
char clientId[50];
```

```
WiFiClient espClient;
PubSubClient client(espClient);
```

```
const int ledPin = 2;
```

```
void setup() {
  Serial.begin(115200);
  randomSeed(analogRead(0));
```

```
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
```

```
  wifiConnect();
```

```
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println(WiFi.macAddress());
  stMac = WiFi.macAddress();
  stMac.replace(":", "_");
  Serial.println(stMac);
```

```
  client.setServer(mqttServer, port);
  client.setCallback(callback);
  pinMode(ledPin, OUTPUT);
}
```

```
void wifiConnect() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
}
```

```
void mqttReconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    long r = random(1000);
    sprintf(clientId, "clientId-%ld", r);
    if (client.connect(clientId)) {
      Serial.print(clientId);
      Serial.println(" connected");
      client.subscribe("room/lamp");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

[실습6] ESP32 MQTT 동작 구현(3)

```
void callback(char* topic, byte* message, unsigned int length) {
    Serial.print("Message arrived on topic: ");
    Serial.print(topic);
    Serial.print(". Message: ");
    String stMessage;

    for (int i = 0; i < length; i++) {
        Serial.print((char)message[i]);
        stMessage += (char)message[i];
    }
    Serial.println();

    if (String(topic) == "room/lamp") {
        Serial.print("Changing output to ");
        if(stMessage == "on"){
            Serial.println("on");
            digitalWrite(ledPin, HIGH);
        }
        else if(stMessage == "off"){
            Serial.println("off");
            digitalWrite(ledPin, LOW);
        }
    }
}
```

```
void loop() {
    delay(10);
    if (!client.connected()) {
        mqttReconnect();
    }
    client.loop();
}
```

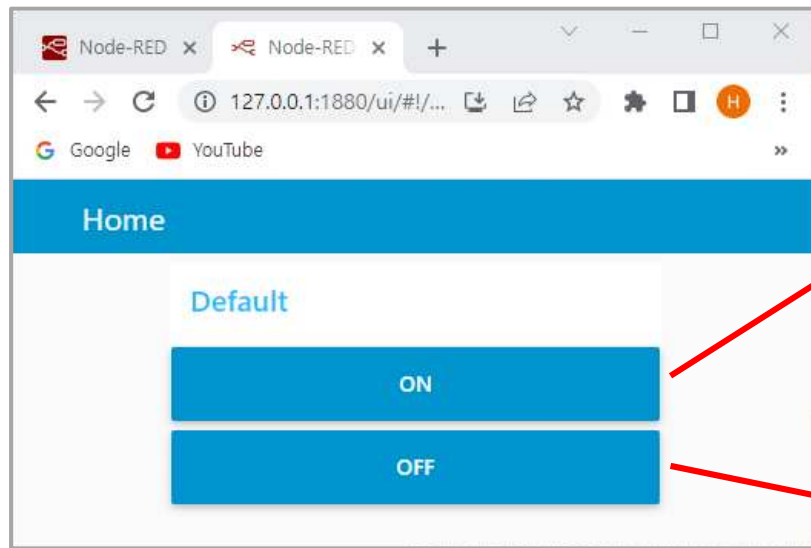
※ Reference :

<https://reference.arduino.cc/reference/en/libraries/pubsubclient/>

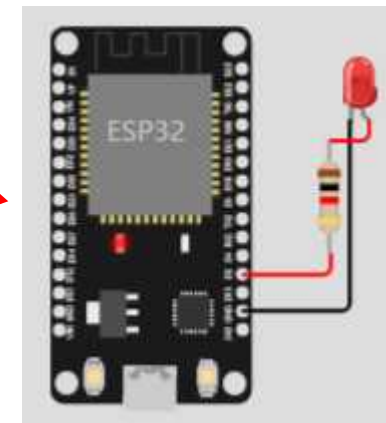
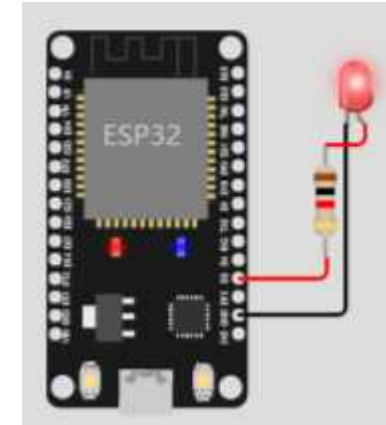
<https://pubsubclient.knolleary.net/api>

Mini Project #1 : LED On/Off by Button

◆ Node-RED 대시보드에서 LED 제어



UI : 127.0.0.1:1880/ui

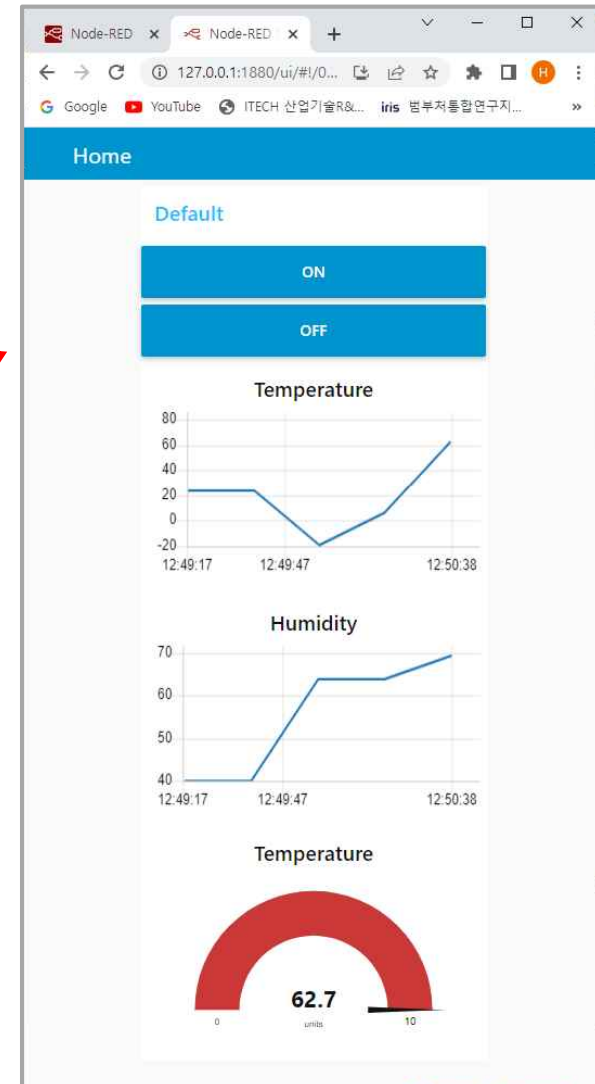
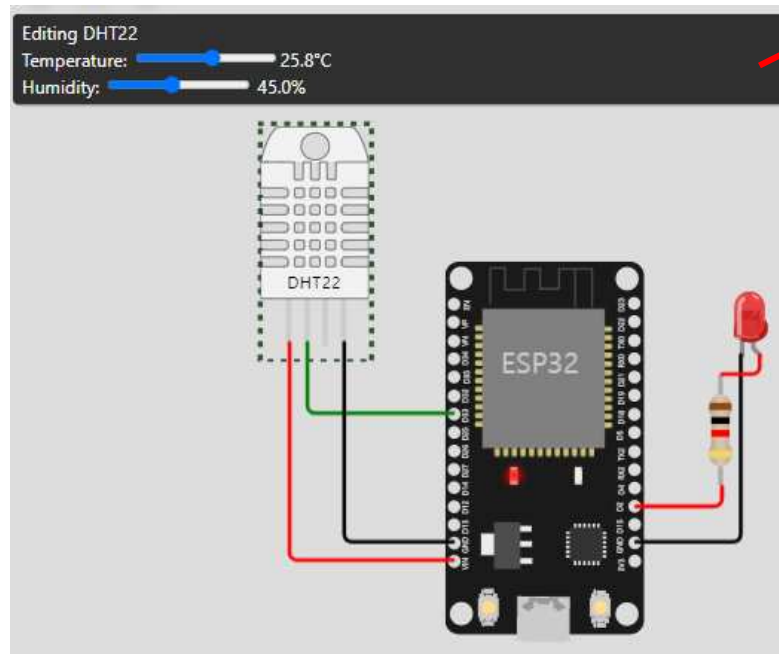


Mini Project #2 : 온도/습도 읽기

◆ 30초마다 온도와 습도를 읽어서 Node-RED 대시보드에 표시

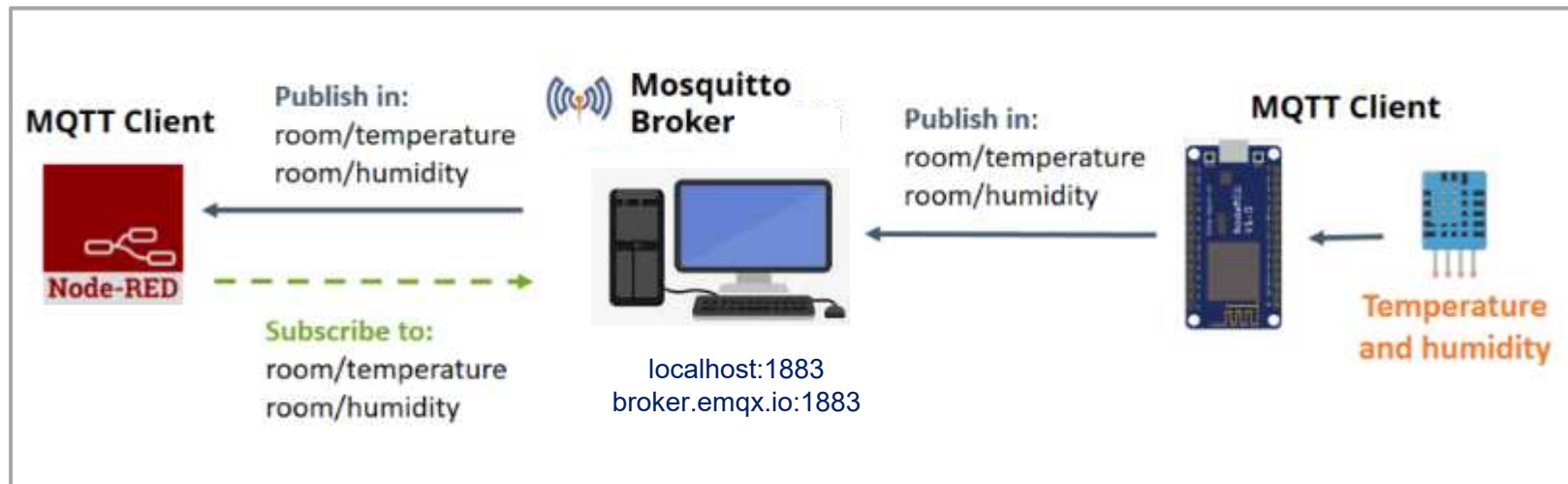
- 온도 : Chart, Gauge
- 습도 : Chart

온도/습도 변화에 따른 차트와 게이지 변경 확인



Mini Project #2. 환경 구성

- ◆ ESP32, 온/습도센서, MQTT와 Node-RED를 이용한 대시보드 표출 실습 환경 구성
 - Node-RED
 - ❖ 30초마다 온도와 습도 값을 읽어서 대시보드에 표시
 - Broker
 - ❖ localhost:1883 (Windows PC Mosquitto Broker)
 - ❖ broker.emqx.io:1883 (Public Broker)
 - ESP32 Target with 온도/습도 센서



목 차

IoT 표준화

Application Layer Protocol

IoT 시각화 툴 Node-RED

IoT 서비스

◆ **Machine Learning**

Machine Learning with Node-RED + Arduino

◆ 주요 학습 내용 : BigML과 Node-RED 활용 방법

- TensorFlow와 Python 환경 설치
- Node-RED에 Machine Learning Module 설치
- 온/습도 센서 데이터 분석



◆ TensorFlow & Node-RED

- 텐서플로(TensorFlow)
 - ❖ 구글(Google)에서 만든 딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능을 제공하는 라이브러리
 - ❖ 텐서플로 자체는 기본적으로 C++로 구현 되어 있음
 - ❖ Python, Java, Go 등 다양한 언어 지원
 - ❖ Python을 최우선으로 지원하며 대부분의 편한 기능들이 Python 라이브러리로만 구현
 - ❖ Python에서 개발하는 것이 가장 편리
- Node-RED의 TensorFlow 지원
 - ❖ Node-RED에서 TensorFlow와 같이 사용 가능한 인터페이스 제공
 - ❖ <https://flows.nodered.org/node/node-red-contrib-tensorflow>

Node-RED에서 ML & TensorFlow 설치 (1)

- ◆ Node-RED에서 설치 또는 npm 명령을 이용하여 설치 가능
- ◆ Node-RED에서 설치하는 방법
 - ◆ 우측 상단 “메뉴” > “팔렛트 관리” > “설치가능한 노드” 에서 검색하여 설치
 - ◆ “node-red-contrib-machine-learning”, “node-red-contrib-tensorflow” 설치
 - ◆ 설치 후 machine learning 팔렛트 확인

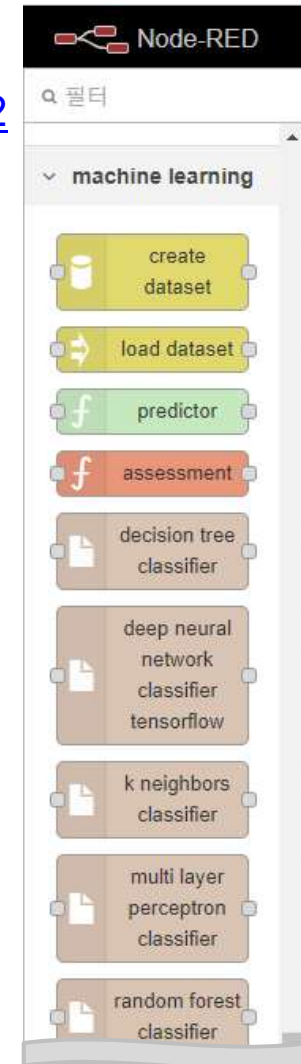
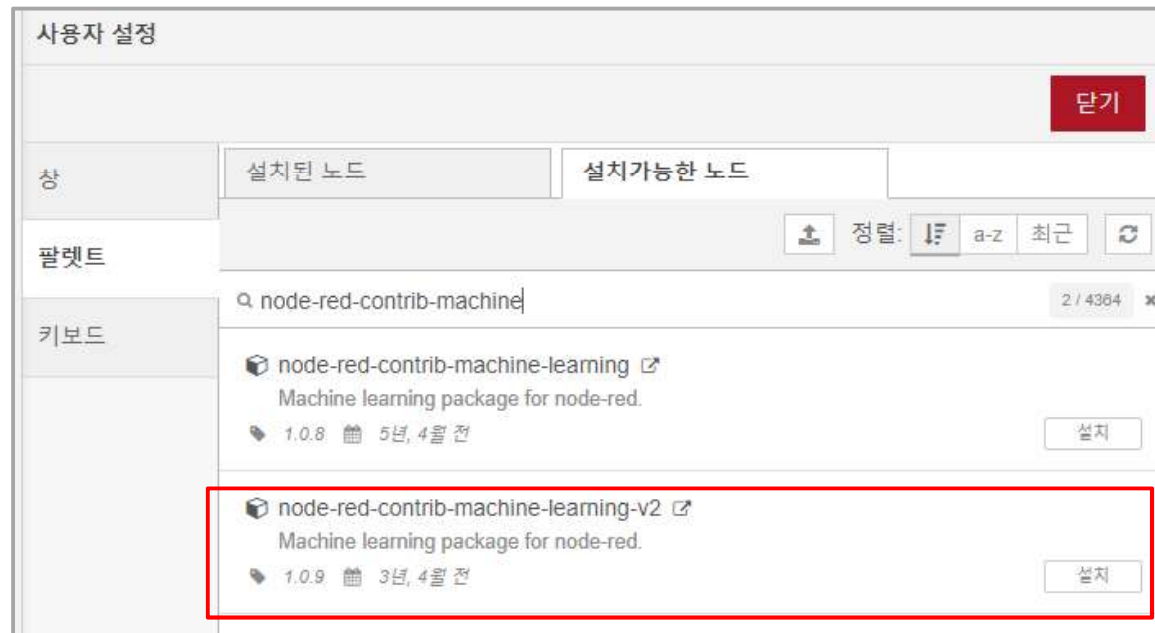


- ◆ 명령으로 설치하는 방법
 - ◆ `npm install node-red-contrib-machine-learning`
 - ◆ `npm install node-red-contrib-tensorflow`

Node-RED에서 ML & TensorFlow 설치 (2)

◆ Library : node-red-contrib-machine-learning

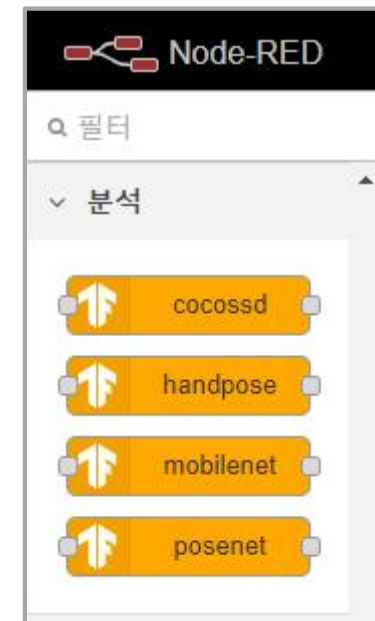
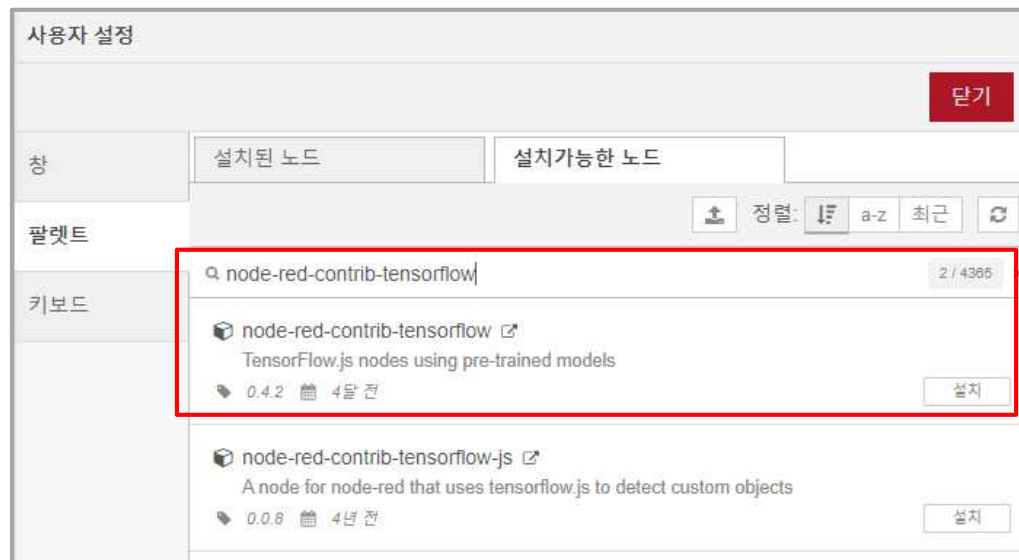
- <https://flows.nodered.org/node/node-red-contrib-machine-learning-v2>
- 설치 : npm install node-red-contrib-machine-learning



Node-RED에서 ML & TensorFlow 설치 (3)

◆ Library : node-red-contrib-tensorflow

- <https://flows.nodered.org/node/node-red-contrib-tensorflow>
- 설치 : npm install node-red-contrib-tensorflow



#	Name	Description
1	cocossd	A node that returns the name of the object in the image
2	handpose	A node that estimates the positions of fingers and joints from a hand image
3	mobilenet	A node that returns the name of the object in the image
4	posenet	A node that estimates the positions of arms, head, and legs from the image of a person

Node-RED에서 ML & TensorFlow 설치 (4)

- ◆ Library : bigml-nodered
 - <https://flows.nodered.org/node/bigml-nodered>
 - 설치 : npm install bigml-nodered



목 차

IoT 표준화

Application Layer Protocol

IoT 시각화 툴 Node-RED

Node-RED 활용

IoT 서비스

Machine Learning

◆ 참고 : 용어정리

용어정리

용어	설명	응용분야
Java	Java는 웹 애플리케이션 코딩에 널리 사용되는 프로그래밍 언어 다중플랫폼, 객체지향 및 네트워크 중심 언어 Java는 썬 마이크로 시스템즈에서 개발 발표	모바일 앱, 클라우드 컴퓨팅, 빅데이터, 인공지능, 사물인 터넷 등
JavaScript	JavaScript는 웹페이지에서 복잡한 기능을 쉽고 단순하게 구현할 수 있도록 하는 스크립트 언어 또는 프로그래밍 언어 Netscape Inc.에서 개발, Java 플랫폼과는 별개임	웹페이지(서버, 클라이언트)
Node.js	JavaScript 엔진으로 빌드된 JavaScript 런타임(프로그램 실행 환경) 노드를 통해 다양한 JavaScript 애플리케이션 실행 Event 기반, Non-blocking I/O 모델 사용	네트워크 어플리케이션 웹 어플리케이션
Node-RED	브라우저 기반 플로우(Flow) 편집기 JavaScript 함수를 개발하는데 사용 가능 런타임은 Node.js 위에서 개발됨	사물인터넷
npm	Node Packaged Manager Node.js로 만들어진 패키지(Package, 모듈:module)을 웹에서 받아서 설치하고 관리하는 툴	
JSON	JavaScript Object Notation JavaScript에서 데이터를 저장하고 전송할 때 사용하는 경량의 데이 터 교환 형식 서버와 클라이언트 간 데이터 전송에 사용	

Java와 JavaScript

구분	Java	Java Script
객체지향	객체지향 프로그래밍 언어	객체지향 스크립트 언어
실행환경	JAVA 애플리케이션은 JVM 위에서 실행 JRE와 JDK 설치	웹브라우저 위에서 실행 추가적인 환경설정 필요 없음
컴파일	프로그래밍 언어로 컴파일 필요	텍스트 기반 스크립트로 컴파일 없이 바로 실행
모바일 애플리케이션	휴대전화 APP 대부분은 JAVA 사용 Android 스마트폰도 JAVA 사용	주로 웹 애플리케이션 용
변수 지정	변수 자료형은 반드시 선언 되어야 함 (정적 형지정, static typing) 블록 밖에서 사용 불가	변수 자료형이 선언되지 않음 (종적 형지정, dynamic typing) 함수 안에서만 사용
적용 범위	다양한 운영체제에서 실행 가능한 독립적인 언어	웹브라우저 HTML, CSS 와 같은 환경 필요

질의 응답