당신의 여행에 날개를 달다

NALDA

SSAFY 7th TEAM NALDA

팀장: 박명규
팀원: 곽영후, 김상현, 김정아, 김혜수, 정형진

August 19, 2022

목차

개발 동기 2
기술 스택 2
빌드 상세내용 3
백엔드 빌드 3
빌드 4
프론트 엔드 빌드 5
배포 특이사항 6
DB 설정 7
1. MySQL WorkBench 추가하기 7
2. 계정 정보 입력 7
3. IP 정보 입력 8
프로퍼티 정의 8
스프링부트 HTTPS 설정 10
외부서비스 10
KAKAO 주소 API 10
인천 국제 공항 버스 정보 API 11
Embedded 설정 12
1. 제품명 12
2. 제품 정보 12
3. 기능 설명 12
3. 메인보드 핀 정보 14
4. 가상 키보드 설치 14
Embedded 통신 15
1. Raspberry MySQL RDS 접속하기 15
2. MQTT(Mosquitto) 16
3. ESP32-WROOM-32 Modile MQTT 20
4. MQTT Broker → DB 23

개발 동기

현재 해외에서 국내로 들어오는 모든 여행자는 관세법에 따라 인적사항, 세관 신고 대상 물품 등을 기재한 여행자 휴대품 신고서를 제출 해야 합니다. 이를 통해 면세 범위를 넘는 물품에 대해 관세를 냅니다. 관세청은 매번 입국 시 종이로 된 신고서를 작성하는 불편함과 절차상의 번거러움을 가지고 있습니다.

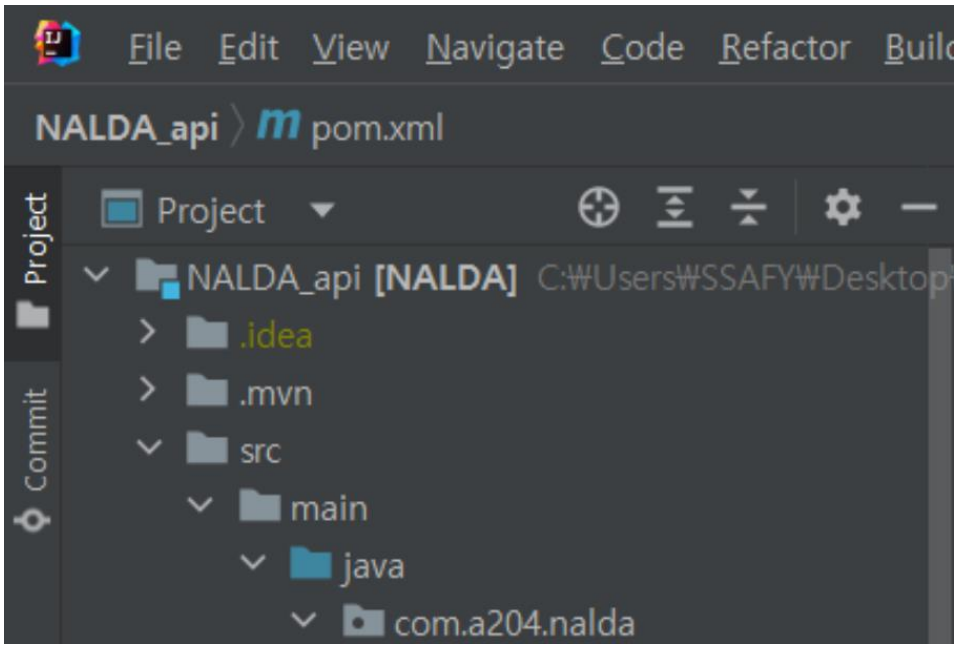
우리는 이러한 불편함이 기대감으로 가득차야 할 여행의 방해 요소가 될 수 있다고 생각하고 불편함을 줄여 줄 방법을 생각해보습니다. 우리는 이제 미리 작성해야 할 인적 사항을 입력할 필요가 없습니다. 지류와 볼펜없이 세관신고서를 손 쉽고 간편하게 작성하고 다양한 기내 서비스 제공으로 행복한 여행에 날개를 달아 보아요.

기술 스택

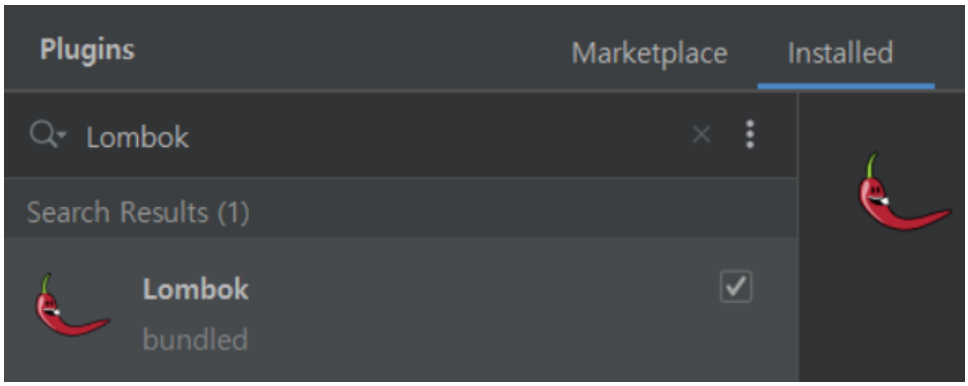
1. 이슈관리 : Jira
2. 형상관리 : Gitlab
3. 커뮤니케이션 : mattermost, notion, webex
4. 개발 환경
    1. OS : window 10
    2. IDE
        1. Intellij IDEA 2022.1.3
        2. Visual Studio Code
        3. UI/UX : Miro
    3. Database : MySQL Workbench
    4. Server : AWS EC2 (MobaXterm)
        1. Ubuntu 20.04.4 LTS
5. 상세 사용
    1. Backend
        1. Java 11
        2. Spring Boot Maven 4.0.0
        3. Spring Boot 2.7.1
        4. Lombok, Jpa
    2. Frontend
        1. Nuxt.js 2.15.8
        2. HTML5, CSS3, JavaScript(ES6)
        3. vuex 4.1.0, vuetify 2.6.8, bootstrap 4.6.1
    3. AWS
        1. RDS(mysql)
        2. EC2
    4. Embedded
        1. Rasberrypi 4

빌드 상세내용

백엔드 빌드

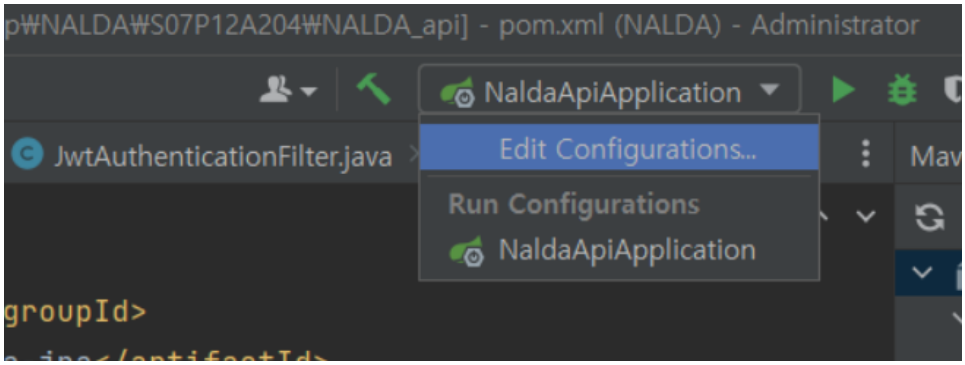


Intellij 에서 NALDA_api 을 maven 으로 import 합니다. import 이후, Lombok 의 플러그인 설치를 해야합니다. Intellij 의 File → Settings → Plugins 로 들어갑니다. Lombok 을 검색한 뒤 설치 해줍시다.

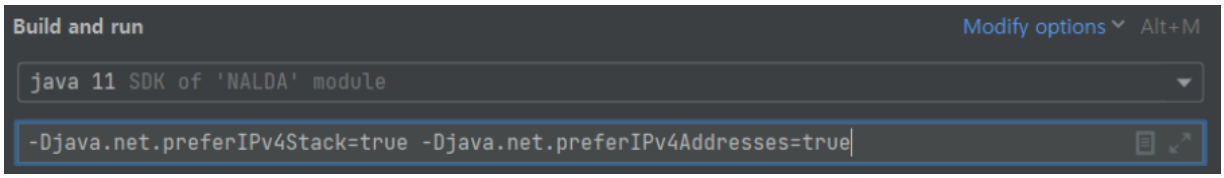


installed 에 Lombok 이 설치되면 intellij 를 재실행 해줍니다.

다음으로는 IPv4 설정을 해주어야 합니다. 그 이유는 좌석정보를 IP 주소로 받아오는데 DB 에 IPv4 형식으로 ip 주소를 저장할 것이기 때문입니다.



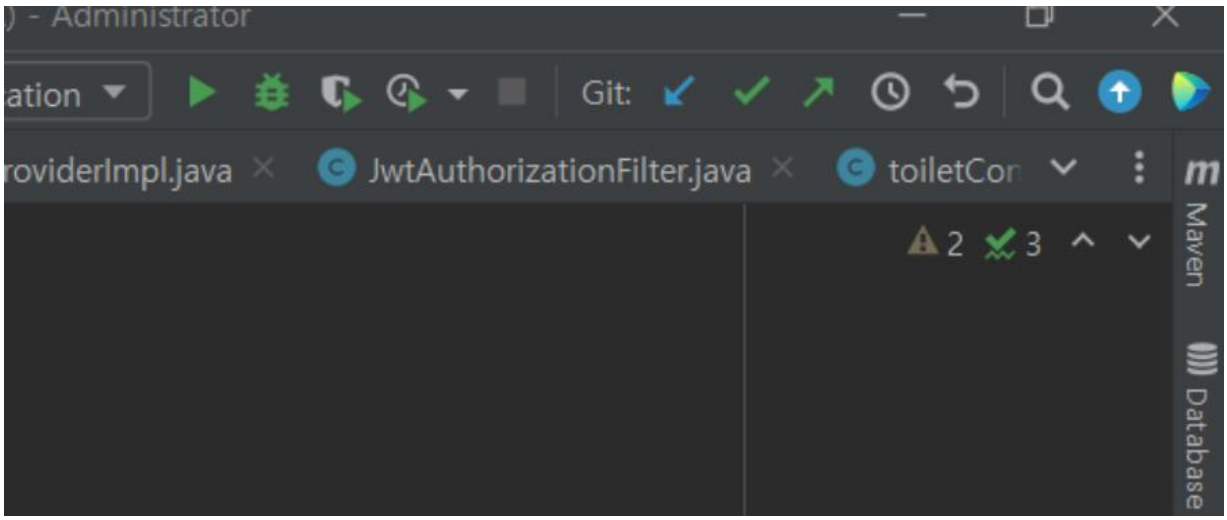
Edit Configurations 를 클릭해줍니다.



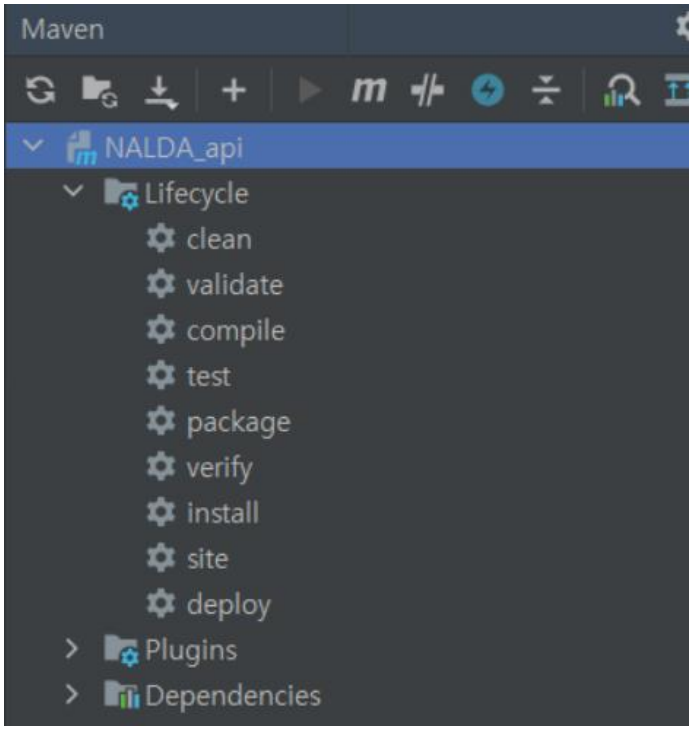
위와 같이 VM option 을 설정해 줍니다.

```
-Djava.net.preferIPv4Stack=true
-Djava.net.preferIPv4Addresses=true
```

빌드



intellij 우측 상단에 Maven 버튼을 클릭합니다.



위와 같이 뜨는 화면에 clean, validate, compile, install 순서대로 클릭을 하고 각각 실행이 완료 될 때까지 기다려 줍니다.

후에 NALDA_api/target 폴더에 들어가면

jar 파일이 생성된 것을 확인할 수 있습니다.

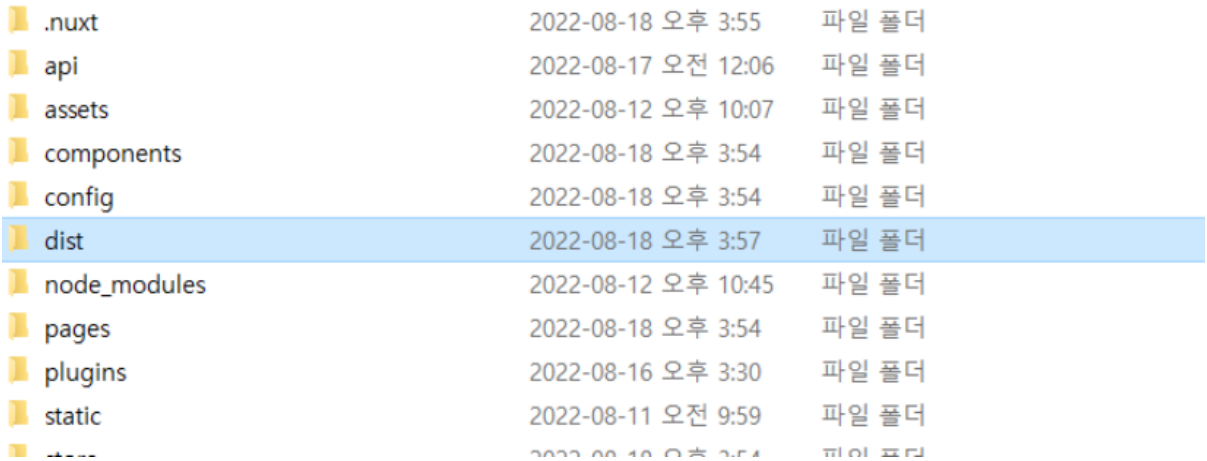
프론트 엔드 빌드

1. node_module 를 위한 기본 install

```
npm i
```

2. 빌드 명령어

```
npm run generate
```



dist 폴더 생성

배포 특이사항

1. AWS EC2 에 만들어진 jar 파일과 dist 폴더를 업로드 후 (수동) 배포를 진행합니다.
2. dist 폴더 안의 200.html 의 이름을 index.html 로 변경해줍니다.
3. 다음과 같은 명령어로 확인합니다.
    i. 구동 중인 jar 파일 확인

```
ps -ef | grep jar
```

다음 명령어를 실행하면 구동중인 jar 프로세스의 PID 가 보입니다.

ii. 구동 중인 프로세스 종료

```
kill -9 <PID>
```

해당 프로세스를 종료하여, 배포서버에서 정상적으로 배포가 중단되었는지 확인합니다.

iii. 새로운 jar 파일 실행

```
nohup java -jar abcdefg.jar &
```

iv. nginx 재실행

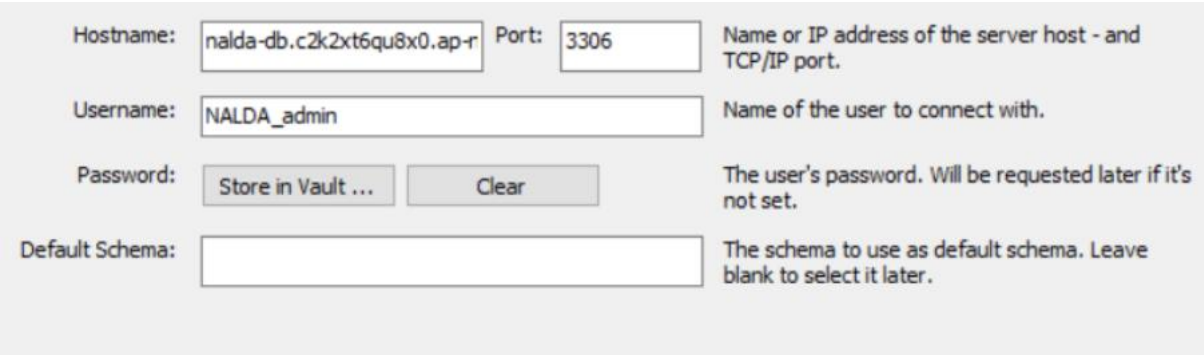
```
sudo service nginx restart
```

DB 설정

1. MySQL WorkBench 추가하기

MySQL WorkBench 를 열어서 + 버튼을 눌러줍니다.

2. 계정 정보 입력

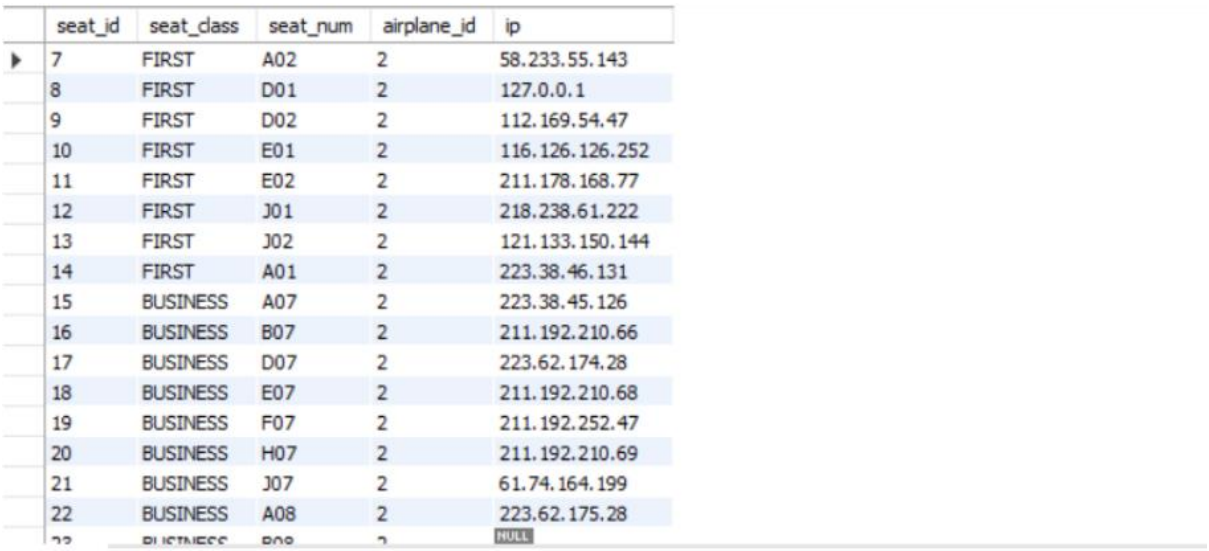


Hostname: nalda-db.c7ku0zhouuvi.us-west-1.rds.amazonaws.com

Username: NALDA_admin

Password: Blueskin741^

3. IP 정보 입력



seat_id	seat_class	seat_num	airplane_id	ip
7	FIRST	A02	2	58.233.55.143
8	FIRST	D01	2	127.0.0.1
9	FIRST	D02	2	112.169.54.47
10	FIRST	E01	2	116.126.126.252
11	FIRST	E02	2	211.178.168.77
12	FIRST	J01	2	218.238.61.222
13	FIRST	J02	2	121.133.150.144
14	FIRST	A01	2	223.38.46.131
15	BUSINESS	A07	2	223.38.45.126
16	BUSINESS	B07	2	211.192.210.66
17	BUSINESS	D07	2	223.62.174.28
18	BUSINESS	E07	2	211.192.210.68
19	BUSINESS	F07	2	211.192.252.47
20	BUSINESS	H07	2	211.192.210.69
21	BUSINESS	J07	2	61.74.164.199
22	BUSINESS	A08	2	223.62.175.28
				NULL

각 기계의 IP 주소에 따라 자리를 구분하기 때문에 SEAT 테이블에 입력해주어야 테스트가 가능합니다.

```
UPDATE seat SET ip = 'ip 주소' WHERE seat_id = 'ip 가 null 인 행'
```

프로퍼티 정의

1. nginx 세팅
    i. ec2 에서 /etc/nginx/sites-available 파일로 접근

```
sudo apt-get update
```

    ii. default 파일 편집

```
sudo vi default
```

```
server {
  listen 80 default_server proxy_protocol;
  listen [::]:80 default_server;

  server_name  i7a204.p.ssafy.io;

  return 301 https://i7a204.p.ssafy.io$request_uri;
}

server {
  listen 443 ssl;
  listen [::]:443;
  server_name i7a204.p.ssafy.io;

  ssl_certificate     /etc/letsencrypt/live/i7a204.p.ssafy.io/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/i7a204.p.ssafy.io/privkey.pem;

  location / {
    root      /home/ubuntu/dist;
    try_files $uri /index.html;
}
  location /api
{
        proxy_pass http://i7a204.p.ssafy.io;
        proxy_redirect off;
        charset utf-8;

        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;
         }
}
```

스프링부트 HTTPS 설정 

SSL 인증서 없이 웹 애플리케이션을 서버에 올리면 http 로 접속이 되고 경고 메시지가 나타납니다.

application.yml 파일에 위의 코드를 넣어주어 해결합니다.

```
ssl:
  key-store: classpath:keystore.p12
  key-store-type: PKCS12
  key-password: ssafy1357
  key-alias: naldakey
  key-store-password: ssafy1357
http2:
  enabled: true
```

```
meta:[
                                {
                'http-equiv': 'Content-Security-Policy',
                content: 'upgrade-insecure-requests',
              }
                 ]
```

다음으로 프론트엔드에서는 nuxt.config.js 에 위의 코드를 넣어주면 보안연결 적용이 완료됩니다.

외부서비스

KAKAO 주소 API

주소 API 의 경우 다른 KAKAO API 와 다르게 KEY 를 발급받지 않아도 되며, 사용량 제한 또한 없다. 기업용/상업용 상관 없이 무료로 사용 가능하다.

코드를 사용하기 위해서 nuxt.config.js 의 head 에 아래코드를 추가하여 전역으로 외부 스크립트를 로드할 수 있도록 도와준다.

https://postcode.map.daum.net/guide#sample 의 예제 **사용자가 선택한 값 이용하기** 의 코드를 활용한다.

```
script: [
      // 다음 주소 검색 API
      { src:
'//t1.daumcdn.net/mapjsapi/bundle/postcode/prod/postcode.v2.js' },
    ],
```

인천 국제 공항 버스 정보 API

인천 국제 공항 여객터미널 T1 과 T2 의 버스 정보로 각 버스의 번호와 시간, 노선도 등의 내용을 담고 있다.

```
async getInfo({ commit }, area) {
        let url = 'http://apis.data.go.kr/B551177/BusInformation/getBusInfo'
            const key =
              '서비스키'
            commit('CLEAR_INFO_LIST')
            url += '?' + encodeURIComponent('serviceKey') + '=' + key
            url += '&' + encodeURIComponent('numOfRows') + '=' +
encodeURIComponent('100')
            url += '&' + encodeURIComponent('pageNo') + '=' +
encodeURIComponent('1')
            url += '&' + encodeURIComponent('area') + '=' +
encodeURIComponent(area)
            url += '&' + encodeURIComponent('type') + '=' +
encodeURIComponent('json')
            await fetch(url)
              .then((res) => res.json())
              .then((data) => {
                commit('SET_INFO_LIST', data.response.body)
              })
}
```

응답(사용데이터)

- area: 지역(1. 서울, 2. 경기, 3. 인천, 4. 강원, 5. 충청, 6. 경상, 7. 전라)
- busnumber: 버스 번호
- t1wdayt: 제 1 터미널 평일 시간표
- t1wt: 제 1 터미널 주말 시간표
- t2wdayt: 제 2 터미널 평일 시간표
- t2wt: 제 2 터미널 주말 시간표
- routeinfo: 노선 정보
- totalCnt: 총 항목 수

Embedded 설정

1. 제품명

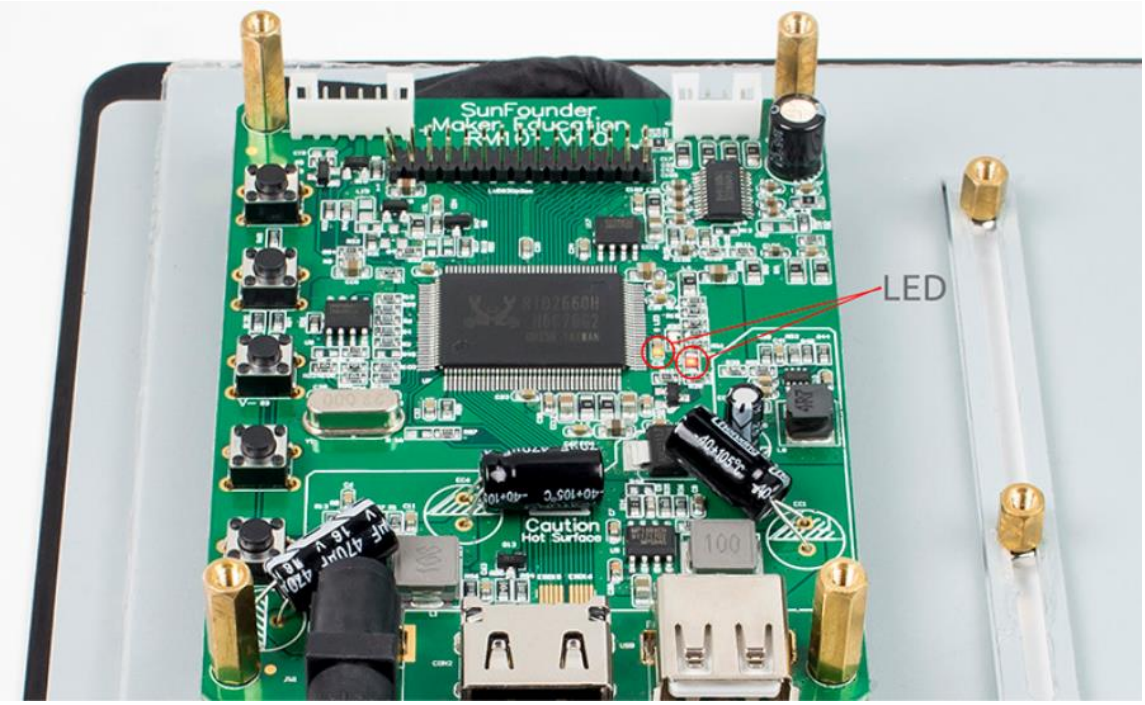
10.1 Inch Touch Screen for Raspberry Pi

2. 제품 정보

- 10.1 인치 모니터 10-point capacitive touch screen
- 1280x800
- 사용 가능 운영 체제 : Raspbian, Ubuntu, Ubuntu Mate, Windows, Android, and Chrome OS(USB 확장 케이블 필요)
- Features
    - Resolution: 1280x800
    - Size: 10.1"
    - LCD Type: IPS
    - Power: DC12V/2A
    - Consumption: 5W
    - Capacitive touch
    - 10-point touch screen

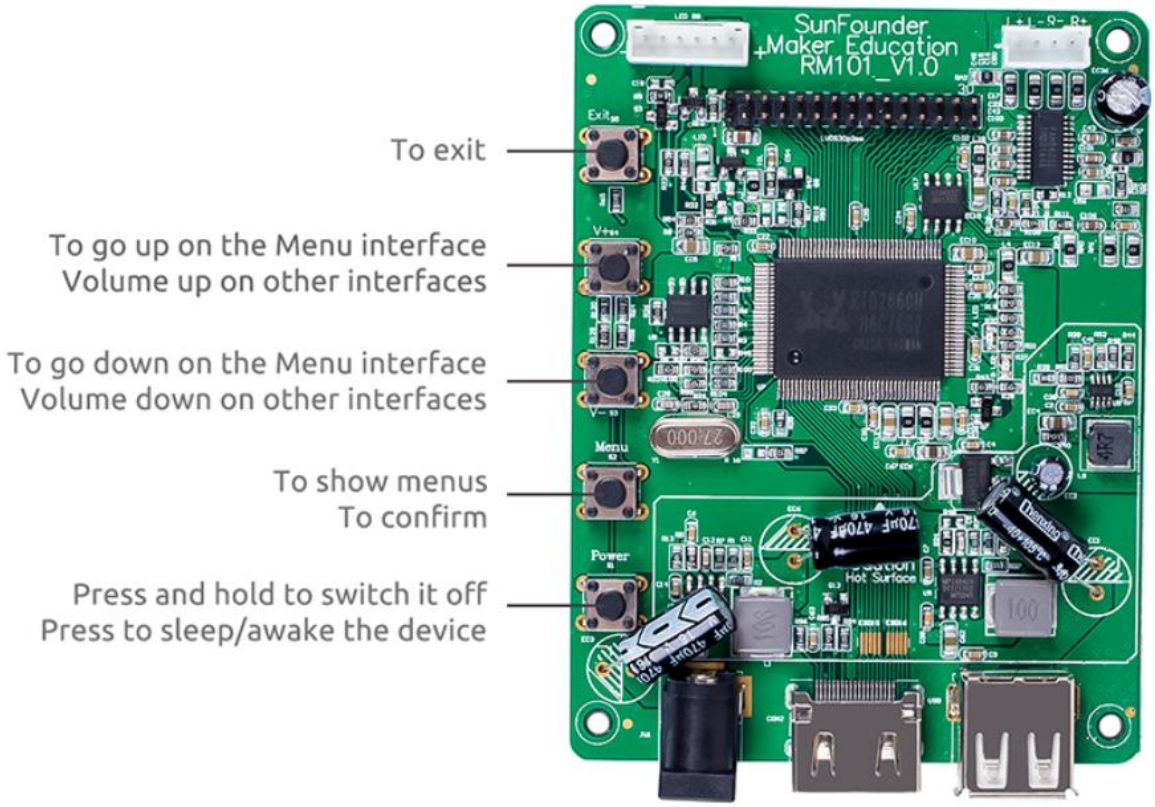
3. 기능 설명

1. 메인보드 정보



- 메인 보드에 2 개 LED 존재
- 스크린 전원이 켜지면 2 개 LED 즉각적으로 켜짐
- 만약 HDMI 신호가 없으면 초록 LED 만 들어온다.
- HDMI 신호 있으면 빨간 LED 도 들어옴
- 버튼 누르면 디스플레이 화면 들어오고, LED 두 개 켜짐
- 만약 디바이스 끄면 붉은색 LED 만 들어온다.

2. 메인보드 버튼



a. Power
   - 기기를 키고 끄는데 사용

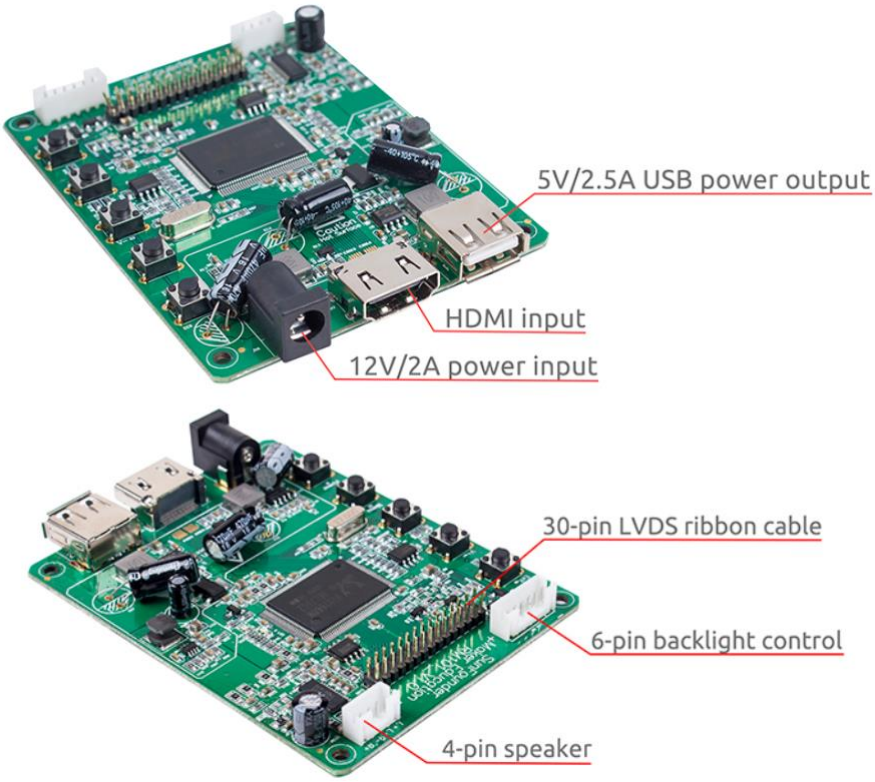
b. Menu
   - 메뉴 보여줌, 확인버튼

c. V−
   - 메뉴 인터페이스로 돌아감
   - 다른 인터페이스의 볼륨 내리기

d. V+
   - 메뉴 인터페이스를 올림
   - 다른 인터페이스의 볼륨 높이기

e. Exit
   - 나가기

3. 메인보드 핀 정보

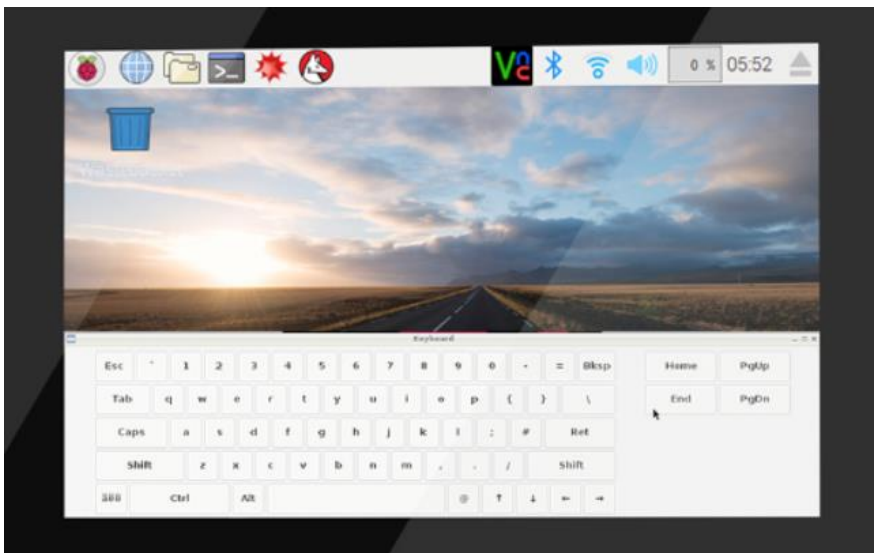


4. 가상 키보드 설치

1. 설치 명령어

```
$sudo apt-get install matchbox-keyboard
```

2. 설치 완료 화면



Embedded 통신

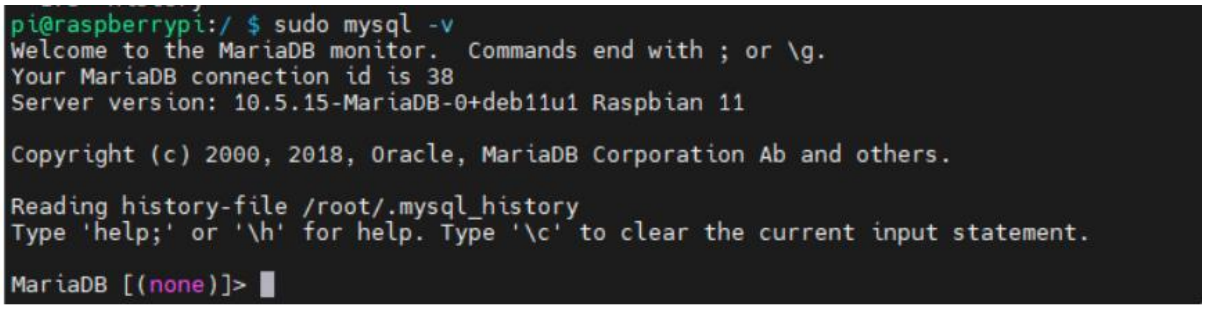
1. Raspberry MySQL RDS 접속하기

1. 환경설정

- raspberry pi 에 mariaDB 설치하기

```
$ sudo apt install mariadb-client mariadb-server
```

- 버전 확인



2. 라즈베리파이에서 MySql 사용하기

```
$ sudo mysql
$ use mysql;#change database
```

3. AWS RDS 접속하기

- RDS 접속 명령어

```
$ sudo mysql -u NALDA_admin -p -h nalda-db.c2k2xt6qu8x0.ap-northeast-
2.rds.amazonaws.com
```

- 데이터베이스 사용하기

```
$ use NALDA_db;
```

2. MQTT(Mosquitto)

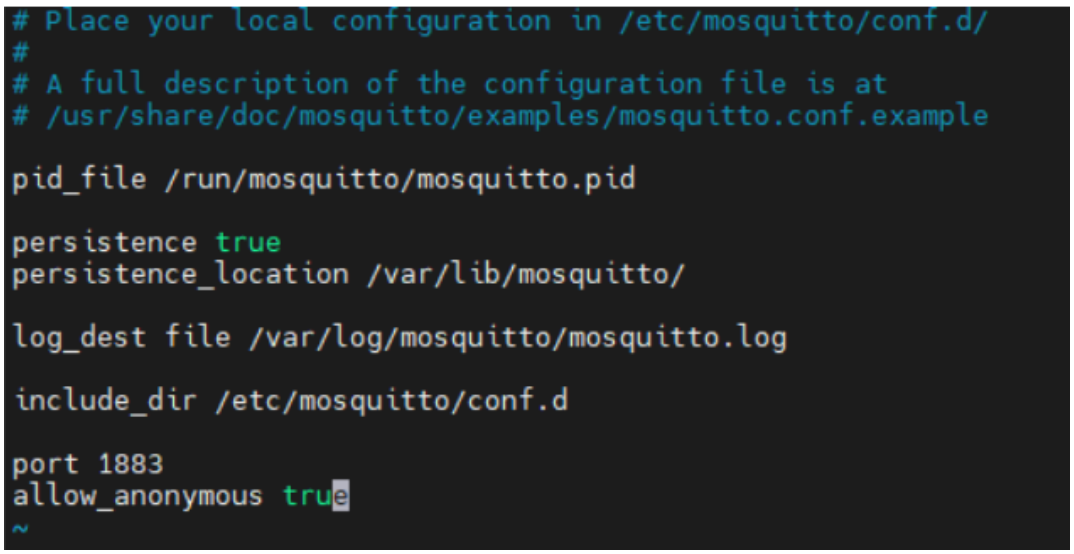
1. 라즈베리파이에 MQTT 설치

```
$ sudo apt-get install mosquitto mosquitto-clients
```

2. 라즈베리파이를 MQTT broker 로 설정

```
$ sudo vi /etc/mosquitto/mosquitto.conf
```

- 위 명령어를 통해 vi 로 mosquitto.conf 파일을 수정한다.



- port 번호를 1883 으로 설정하고 allow_anonymous 값을 true 로 설정해서 접속할 수 있도록 허용한다.

3. MQTT 명령어

- MQTT 를 실행하는 명령어
    - MQTT 를 broker 로 사용하고 싶으면 반드시 먼저 실행시켜준다.

```
$ sudo /etc/init.d/mosquitto start
```

- MQTT 를 멈추는 명령어

```
$ sudo /etc/init.d/mosquitto stop
```

- MQTT 를 Publisher 로 사용하는 경우

```
$ mosquitto_pub -h brokerIP -t "topic"
```

- MQTT 를 Subscriber 로 사용하는경우

```
$ mosquitto_sub -t "topic" -h broker_ip -p port_num
```

4. ESP-32S 보드 Publish 코드

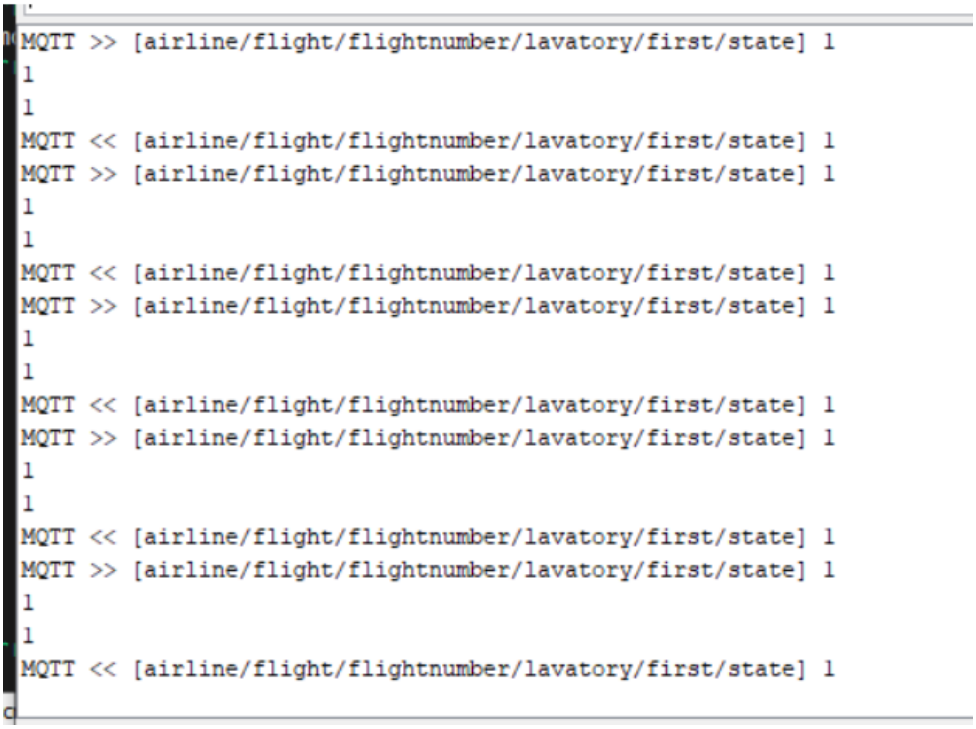
```
#include "EspMQTTClient.h"
#define led_green 23 // 초록 led 핀
#define led_red 22 // 빨강 led 핀
#define button 21 // 버튼 핀
#define BAUD_RATE 9600 // 보드레이트 정의 (추후 적용)

EspMQTTClient client(
  "SK_WiFiGIGA4676_2.4G", //SSID 이름
  "1606051756", //SSID Password
  "192.168.35.242",  // MQTT Broker server ip (Raspberry pi)
  "MQTTUsername",   // Can be omitted if not needed
  "MQTTPassword",   // Can be omitted if not needed
  "TestClient",     // Client name that uniquely identify your device,
두개연결시 변경
  1883              // The MQTT port, default to 1883. this line can be
omitted
);


int button_push=0; //버튼 입력값의 변화를 담을 변수
String buttonString; //payload 에 담기 위해 string 으로 변경
char *topic = "airline/flight/flightnumber/lavatory/first/state"; //topic
지정
void tx(){
  client.publish(topic,buttonString);  //topic 에 payload 를 전송
}
```

```
void setup()
{
  Serial.begin(115200);
  pinMode(led_green,OUTPUT);  // led 출력 설정
  pinMode(led_red,OUTPUT); // led 출력 설정
  pinMode(button,INPUT); // 버튼 입력 설정

  // Optional functionalities of EspMQTTClient
  client.enableDebuggingMessages(); // Enable debugging messages sent to
serial output
  client.enableHTTPWebUpdater(); // Enable the web updater. User and
password default to values of MQTTUsername and MQTTPassword. These can be
overridded with enableHTTPWebUpdater("user", "password").
  client.enableOTA(); // Enable OTA (Over The Air) updates. Password
defaults to MQTTPassword. Port is the default OTA port. Can be overridden
with enableOTA("password", port).
  client.enableLastWillMessage("TestClient/lastwill", "I am going offline");
// You can activate the retain flag by setting the third parameter to true
}

// This function is called once everything is connected (Wifi and MQTT)
// WARNING : YOU MUST IMPLEMENT IT IF YOU USE EspMQTTClient
void onConnectionEstablished()
{
  // Subscribe to "mytopic/test" and display received message to Serial
  client.subscribe(topic, [](const String & payload) { //topic 에 payload
구독
    Serial.println(payload); //payload 값 시리얼 모니터 출력
  });
}
}

void loop()
{
  button_push= digitalRead(button); // button 의 값을 입력받아 변수에 저장
  buttonString = String(button_push); // string 으로 변환
  if(button_push==1){ // 버튼 값에 따른 led 변화
    Serial.println(button_push);
    digitalWrite(led_green,HIGH);
    digitalWrite(led_red,LOW);
  }
  else {
    Serial.println(button_push);
    digitalWrite(led_green,LOW);
    digitalWrite(led_red,HIGH);
  }
  tx();
  delay(500);
  client.loop();
}
```

- Topic Naming
  - airline/flight/flightnumber/lavatory/first/state
  - airline/flight/flightnumber/lavatory/second/state

5. Rasberry Pi MQTT 결과

- Arduino Serial Monitor



- Raspberry Pi MQTT_sub

3. ESP32-WROOM-32 Modile MQTT

1. ESP-32S 보드 Publish 코드

- Module1

```
#include "EspMQTTClient.h"
#define led_green 23 // green led GPIO23
#define led_red 22 // red led GPIO22
#define button 21 // button GPIO21
#define BAUD_RATE 115200

EspMQTTClient client(
  "SK_WiFiGIGA4676", // My wifi
  "1606051756", // My wifi password
  "192.168.35.242",  // MQTT Broker server ip
  "MQTTUsername",   // MQTT user name, Not use
  "MQTTPassword",   // MQTT password, Not use
  "TestClient",     // Client1
  1883              // The MQTT port, default to 1883.
);

int button_push=0; // button push var
String buttonString; // change button value to string, to publish topic
char *topic = "toilet/777-200ER-1/HL01"; // topic1, toilet, plane_id=777-
200ER-1, toilet_location = Head/Left/01
void tx(){
  client.publish(topic,buttonString); // publish topic
}

void setup()
{
  Serial.begin(BAUD_RATE); // Serial Baud rate
  pinMode(led_green,OUTPUT); // set led_green output
  pinMode(led_red,OUTPUT); // set led_red output
  pinMode(button,INPUT); // set button input

  // Optional functionalities of EspMQTTClient
  client.enableDebuggingMessages(); // Enable debugging messages sent to
serial output
  client.enableHTTPWebUpdater(); // Enable the web updater. User and
password default to values of MQTTUsername and MQTTPassword. These can be
overridded with enableHTTPWebUpdater("user", "password").
  client.enableOTA(); // Enable OTA (Over The Air) updates. Password
defaults to MQTTPassword. Port is the default OTA port. Can be overridden
with enableOTA("password", port).
  client.enableLastWillMessage("TestClient/lastwill", "I am going offline");
// You can activate the retain flag by setting the third parameter to true
}
```

```
// This function is called once everything is connected (Wifi and MQTT)
// WARNING : YOU MUST IMPLEMENT IT IF YOU USE EspMQTTClient
void onConnectionEstablished()
{
  // Subscribe to "mytopic/test" and display received message to Serial
  client.subscribe(topic, [](const String & payload) {
    Serial.println(payload);
  });

}

void loop()
{
  button_push= digitalRead(button); // read button input and store
  buttonString = String(button_push); // change int to string
  if(button_push==1){ // it means nobody use toilet (pull up circuit)
    Serial.println(button_push);
    digitalWrite(led_green,HIGH); // green led on
    digitalWrite(led_red,LOW); // red led off
  }
  else { // it means someone use toilet
    Serial.println(button_push);
    digitalWrite(led_green,LOW); // green led off
    digitalWrite(led_red,HIGH); // red led on
  }
  tx(); // public topic
  delay(5000); // delay 5sec
  client.loop();
}
```

- Module2

```
#include "EspMQTTClient.h"
#define led_green 5 // green led GPIO5
#define led_red 17 // red led GPIO7
#define button 19 // button GPIO19
#define BAUD_RATE 115200

EspMQTTClient client(
  "SK_WiFiGIGA4676", // My wifi
  "1606051756", // My wifi password
  "192.168.35.242",  // MQTT Broker server ip
  "MQTTUsername",   // MQTT user name, Not use
  "MQTTPassword",   // MQTT password, Not use
  "TestClient2",     // Client2
  1883              // The MQTT port, default to 1883.
);

int button_push=0; // button push var
String buttonString; // change button value to string, to publish topic
char *topic = "toilet/777-200ER-1/HR01"; // topic1, toilet, plane_id=777-
200ER-1, toilet_location = Head/Right/01
```

```
void tx(){
  client.publish(topic,buttonString); // publish topic
}

void setup()
{
  Serial.begin(BAUD_RATE); // Serial Baud rate
  pinMode(led_green,OUTPUT); // set led_green output
  pinMode(led_red,OUTPUT); // set led_red output
  pinMode(button,INPUT); // set button input

  // Optional functionalities of EspMQTTClient
  client.enableDebuggingMessages(); // Enable debugging messages sent to
serial output
  client.enableHTTPWebUpdater(); // Enable the web updater. User and
password default to values of MQTTUsername and MQTTPassword. These can be
overridded with enableHTTPWebUpdater("user", "password").
  client.enableOTA(); // Enable OTA (Over The Air) updates. Password
defaults to MQTTPassword. Port is the default OTA port. Can be overridden
with enableOTA("password", port).
  client.enableLastWillMessage("TestClient2/lastwill", "I am going
offline");  // You can activate the retain flag by setting the third
parameter to true
}

// This function is called once everything is connected (Wifi and MQTT)
// WARNING : YOU MUST IMPLEMENT IT IF YOU USE EspMQTTClient
void onConnectionEstablished()
{
  // Subscribe to "mytopic/test" and display received message to Serial
  client.subscribe(topic, [](const String & payload) {
    Serial.println(payload);
  });

}

void loop()
{
  button_push= digitalRead(button); // read button input and store
  buttonString = String(button_push); // change int to string
  if(button_push==1){ // it means nobody use toilet (pull up circuit)
    Serial.println(button_push);
    digitalWrite(led_green,HIGH); // green led on
    digitalWrite(led_red,LOW); // red led off
  }
  else { // it means someone use toilet
    Serial.println(button_push);
    digitalWrite(led_green,LOW); // green led off
    digitalWrite(led_red,HIGH); // red led on
  }
  tx(); // public topic
  delay(5000); // delay 5sec
  client.loop();
}
```

4. MQTT Broker → DB

1. 라즈베리파이 python MQTT 환경 구축

- python 설치
  - 기본적으로 설치되어 있지만, 설치되어 있지 않다면, 아래 명령어로 원하는 버전 설치

```
$ sudo apt-get install python3.9
```

- Paho-mqtt 설치
  - paho-mqtt 는 파이썬의 mqtt 라이브러리이다.
  - 참고자료

```
$ pip3 install paho-mqtt
```

- MySQL-connector 설치
  - python 을 mysql DB 와 연결해주는 라이브러리이다.
  - 참고자료

```
$ pip3 install paho-mqtt
```

2. 파이썬 코드를 이용해 DB 에 접속하기

```
import paho.mqtt.client as mqtt # mqtt 라이브러리
import mysql.connector as mariadb # db 연결 라이브러리

NALDA_db = mariadb.connect(host="nalda-db.c2k2xt6qu8x0.ap-northeast-
2.rds.amazonaws.com",user='NALDA_admin',password='Blueskin741^',database='NA
LDA_db') # NALDA_db 와 연결
cur = NALDA_db.cursor() # cursor 지정
#global value
#global my_topic

# 브로커 접속 시도 결과 처리 콜백 함수
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+ str(rc))
    if rc == 0:
        client.subscribe("toilet/#") # 연결 성공시 토픽 구독 신청
    else:
        print('연결 실패 : ', rc)
```

```
import paho.mqtt.client as mqtt # mqtt 라이브러리
import mysql.connector as mariadb # db 연결 라이브러리

NALDA_db = mariadb.connect(host="nalda-db.c2k2xt6qu8x0.ap-northeast-
2.rds.amazonaws.com",user='NALDA_admin',password='Blueskin741^',database='NA
LDA_db') # NALDA_db 와 연결
cur = NALDA_db.cursor() # cursor 지정
#global value
#global my_topic

# 브로커 접속 시도 결과 처리 콜백 함수
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+ str(rc))
    if rc == 0:
        client.subscribe("toilet/#") # 연결 성공시 토픽 구독 신청
    else:
        print('연결 실패 : ', rc)
```