****

**August 19, 2022**

**SSAFY 7th TEAM NALDA**

팀장: 박명규

팀원: 곽영후, 김상현, 김정아, 김혜수, 정형진

**NALDA**

당신의 여행에 날개를 달다

목차

[개발 동기 2](#_Toc111773023)

[기술 스택 2](#_Toc111773024)

[빌드 상세내용 3](#_Toc111773025)

[백엔드 빌드 3](#_Toc111773026)

[빌드 4](#_Toc111773027)

[프론트 엔드 빌드 5](#_Toc111773028)

[배포 특이사항 6](#_Toc111773029)

[DB설정 7](#_Toc111773030)

[1. MySQL WorkBench 추가하기 7](#_Toc111773031)

[2. 계정 정보 입력 7](#_Toc111773032)

[3. IP 정보 입력 8](#_Toc111773033)

[프로퍼티 정의 8](#_Toc111773034)

[스프링부트 HTTPS 설정 10](#_Toc111773035)

[외부서비스 10](#_Toc111773036)

[KAKAO 주소 API 10](#_Toc111773037)

[인천 국제 공항 버스 정보 API 11](#_Toc111773038)

[Embedded 설정 12](#_Toc111773039)

[1. 제품명 12](#_Toc111773040)

[2. 제품 정보 12](#_Toc111773041)

[3. 기능 설명 12](#_Toc111773042)

[3. 메인보드 핀 정보 14](#_Toc111773043)

[4. 가상 키보드 설치 14](#_Toc111773044)

[Embedded 통신 15](#_Toc111773045)

[1. Raspberry MySQL RDS 접속하기 15](#_Toc111773046)

[2. MQTT(Mosquitto) 16](#_Toc111773047)

[3. ESP32-WROOM-32 Modile MQTT 20](#_Toc111773048)

[4. MQTT Broker → DB 23](#_Toc111773049)

# 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

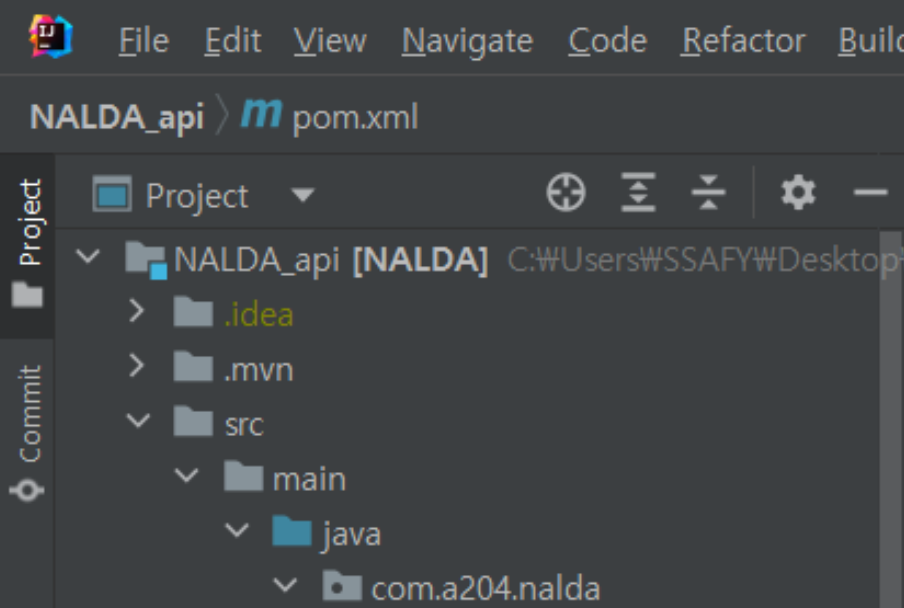
4. Embedded

1. Rasberrypi 4

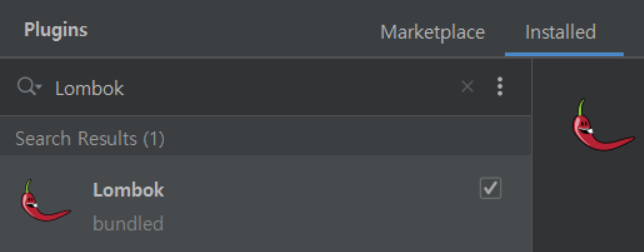
2. ESP32 wifi module

# 빌드 상세내용

## 백엔드 빌드

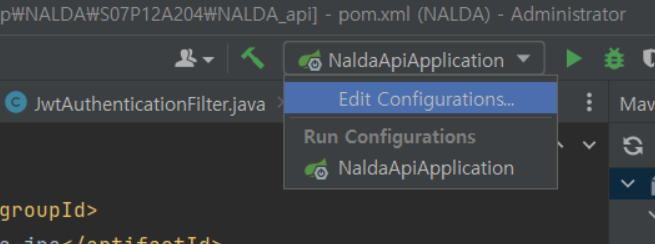


Intellij 에서 NALDA\_api을 maven으로 import 합니다. import 이후, Lombok의 플러그인 설치를 해야합니다. Intellij의 File → Settings → Plugins 로 들어갑니다. Lombok을 검색한 뒤 설치 해줍시다.

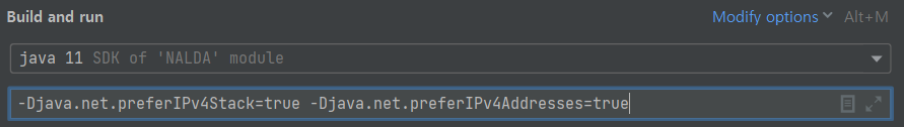


installed에 Lombok이 설치되면 intellij를 재실행 해줍니다.

다음으로는 IPv4 설정을 해주어야 합니다. 그 이유는 좌석정보를 IP주소로 받아오는데 DB에 IPv4형식으로 ip 주소를 저장할 것이기 때문입니다.



Edit Configurations를 클릭해줍니다.

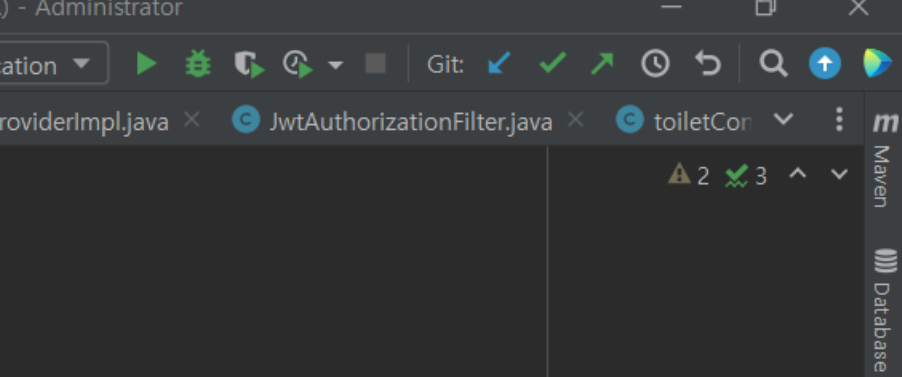


위와 같이 VM option을 설정해 줍니다.

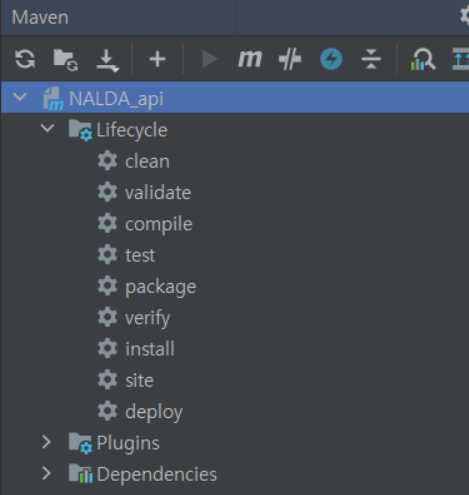
-Djava.net.preferIPv4Stack=true

-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.

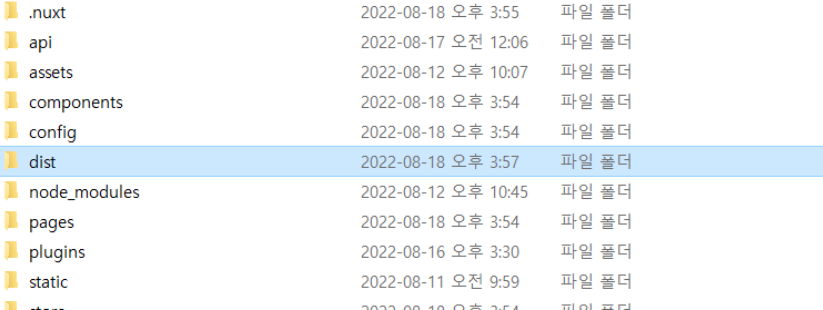
## 프론트 엔드 빌드

1. node\_module를 위한 기본 install

npm i

1. 빌드 명령어

npm run generate



dist 폴더 생성

# 배포 특이사항

1. AWS EC2에 만들어진 jar 파일과 dist폴더를 업로드 후 (수동) 배포를 진행합니다.
2. dist 폴더 안의 200.html의 이름을 index.html로 변경해줍니다.
3. 다음과 같은 명령어로 확인합니다.
   1. 구동 중인 jar 파일 확인

ps -ef | grep jar

다음 명령어를 실행하면 구동중인 jar 프로세스의 PID가 보입니다.

* 1. 구동 중인 프로세스 종료

kill -9 <PID>

해당 프로세스를 종료하여, 배포서버에서 정상적으로 배포가 중단되었는지 확인합니다.

* 1. 새로운 jar 파일 실행

nohup java -jar abcdefg.jar &

* 1. nginx 재실행

sudo service nginx restart

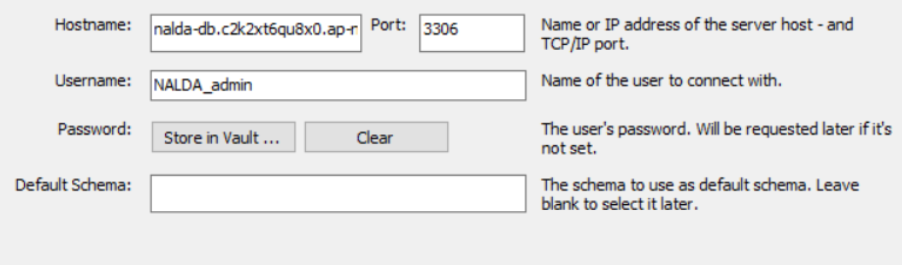
## DB설정

### 1. MySQL WorkBench 추가하기



MySQL WorkBench를 열어서 + 버튼을 눌러줍니다.

### 2. 계정 정보 입력

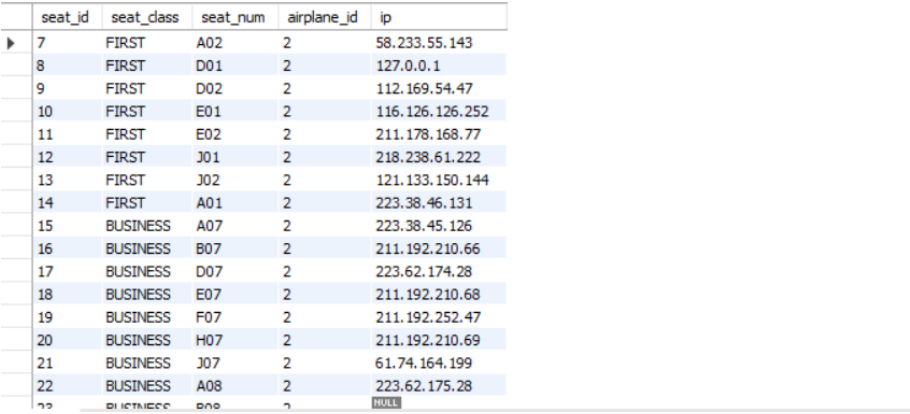


Hostname: [nalda-db.c7ku0zhouuvi.us-west-1.rds.amazonaws.com](http://nalda-db.c7ku0zhouuvi.us-west-1.rds.amazonaws.com/)

Username: NALDA\_admin

Password: Blueskin741^

### 3. IP 정보 입력



각 기계의 IP주소에 따라 자리를 구분하기 때문에 SEAT테이블에 입력해주어야 테스트가 가능합니다.

UPDATE seat SET ip = 'ip주소' WHERE seat\_id = 'ip가 null인 행'

# 프로퍼티 정의

1. nginx 세팅
   1. ec2에서 /etc/nginx/sites-available 파일로 접근

sudo apt-get update

* 1. 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 http로 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.js에 위의 코드를 넣어주면 보안연결 적용이 완료됩니다.

# 외부서비스

### KAKAO 주소 API

주소 API의 경우 다른 KAKAO API와 다르게 KEY를 발급받지 않아도 되며, 사용량 제한 또한 없다. 기업용/상업용 상관 없이 무료로 사용 가능하다.

코드를 사용하기 위해서 nuxt.config.js의 head에 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 T1과 T2의 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

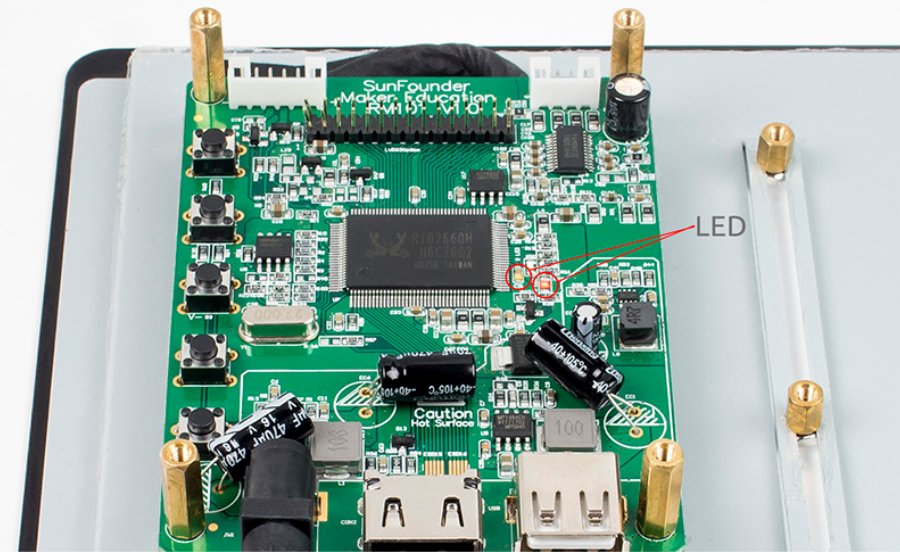
10.1 Inch Touch Screen for Raspberry Pi

### 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

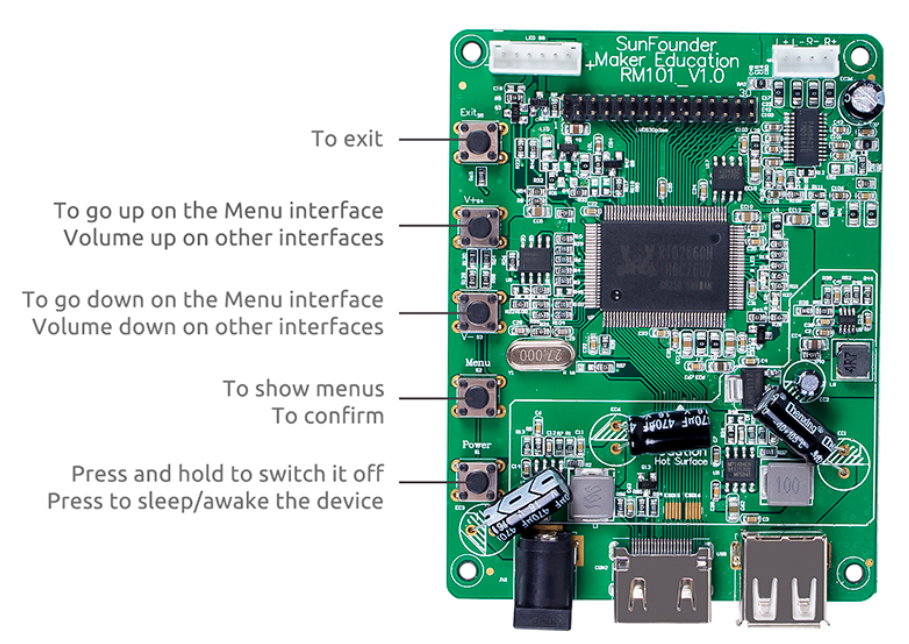
### 3. 기능 설명

1. 메인보드 정보



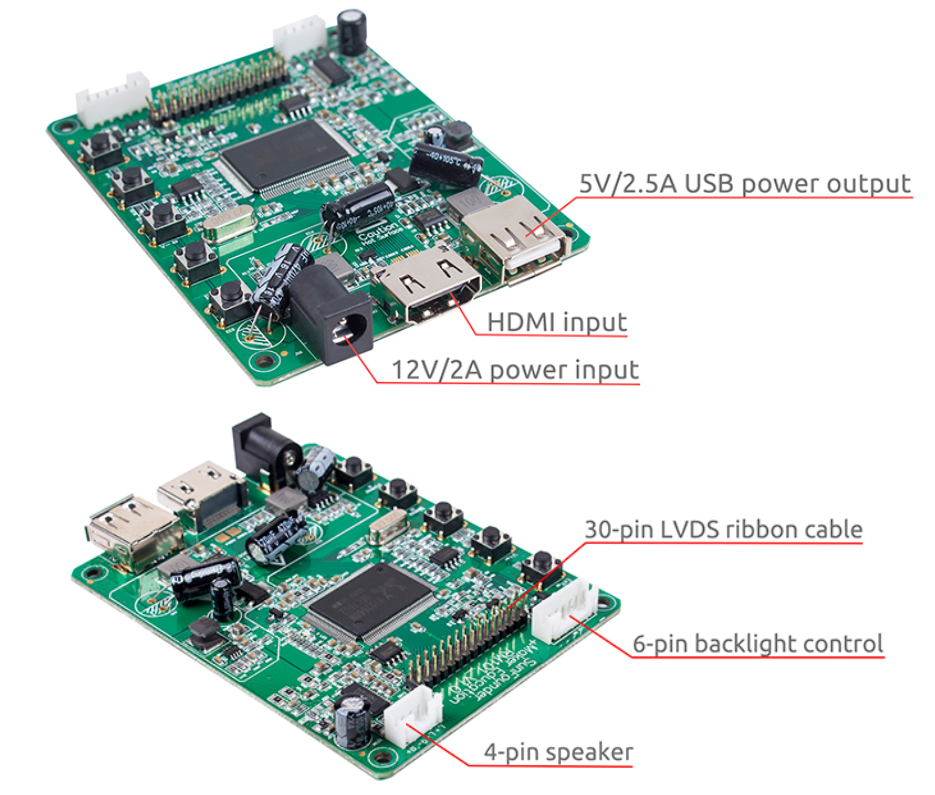
* 메인 보드에 2개 LED 존재
* 스크린 전원이 켜지면 2개 LED 즉각적으로 켜짐
* 만약 HDMI 신호가 없으면 초록 LED만 들어온다.
* HDMI 신호 있으면 빨간 LED도 들어옴
* 버튼 누르면 디스플레이 화면 들어오고, LED 두 개 켜짐
* 만약 디바이스 끄면 붉은색 LED만 들어온다.

1. 메인보드 버튼



1. Power
   * 기기를 키고 끄는데 사용
2. Menu
   * 메뉴 보여줌, 확인버튼
3. V-
   * 메뉴 인터페이스로 돌아감
   * 다른 인터페이스의 볼륨 내리기
4. V+
   * 메뉴 인터페이스를 올림
   * 다른 인터페이스의 볼륨 높이기
5. Exit
   * 나가기

### 3. 메인보드 핀 정보



### 4. 가상 키보드 설치

1. 설치 명령어

$sudo apt-get install matchbox-keyboard

1. 설치 완료 화면



# Embedded 통신

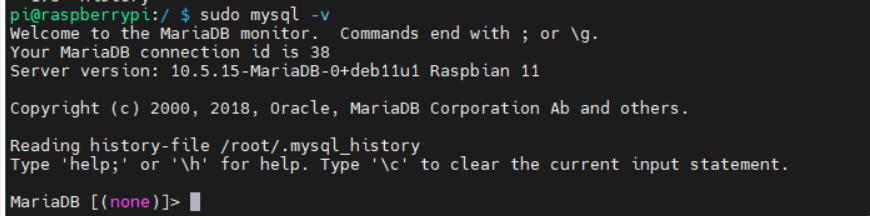
### 1. Raspberry MySQL RDS 접속하기

1. 환경설정

* raspberry pi에 mariaDB 설치하기

$ sudo apt install mariadb-client mariadb-server

* 버전 확인



1. 라즈베리파이에서 MySql 사용하기

$ sudo mysql

$ use mysql;*#change database*

1. 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)

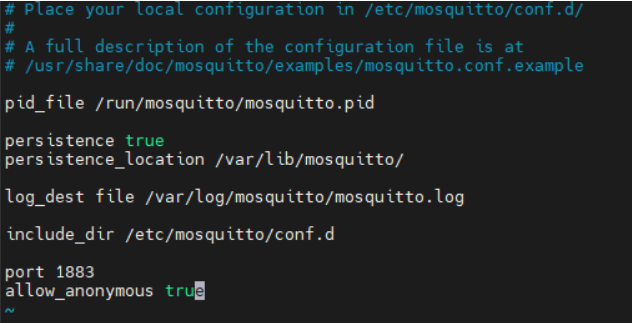
1. 라즈베리파이에 MQTT 설치

$ sudo apt-get install mosquitto mosquitto-clients

1. 라즈베리파이를 MQTT broker로 설정

$ sudo vi /etc/mosquitto/mosquitto.conf

* 위 명령어를 통해 vi로 mosquitto.conf 파일을 수정한다.



* port 번호를 1883으로 설정하고 allow\_anonymous 값을 true로 설정해서 접속할 수 있도록 허용한다.

1. MQTT명령어

* MQTT를 실행하는 명령어
  + MQTT를 broker로 사용하고 싶으면 반드시 먼저 실행시켜준다.

$ sudo /etc/init.d/mosquitto start

* MQTT를 멈추는 명령어

$ sudo /etc/init.d/mosquitto stop

* MQTT를 Publisher로 사용하는 경우

$ mosquitto\_pub -h brokerIP -t "topic"

* MQTT를 Subscriber로 사용하는경우

$ mosquitto\_sub -t "topic" -h broker\_ip -p port\_num

1. 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; *//payload에 담기 위해 string으로 변경*

**char** \*topic = "airline/flight/flightnumber/lavatory/first/state"; *//topic 지정*

**void** tx(){

client.publish(topic,buttonString); *//topic에 payload를 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) { *//topic에 payload 구독*

Serial.println(payload); *//payload 값 시리얼 모니터 출력*

});

}

}

**void** loop()

{

button\_push= digitalRead(button); *// button의 값을 입력받아 변수에 저장*

buttonString = **String**(button\_push); *// string으로 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);

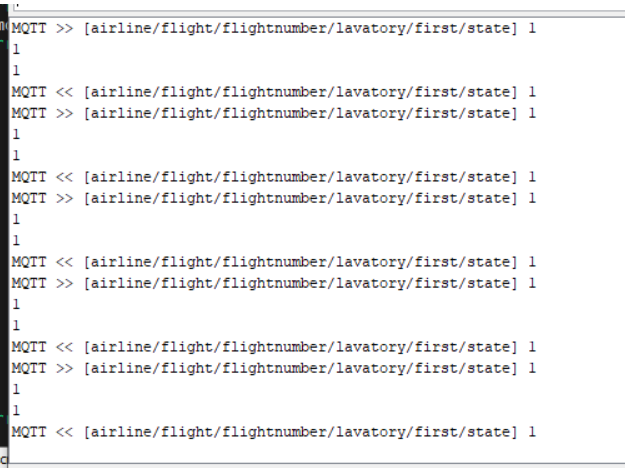
client.loop();

}

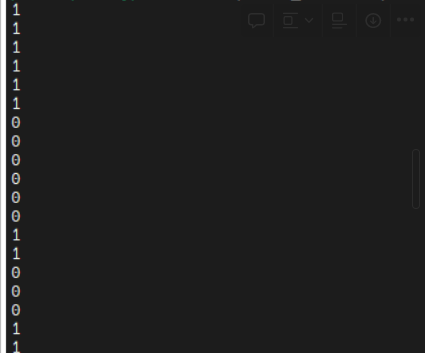
* Topic Naming
  + airline/flight/flightnumber/lavatory/first/state
  + airline/flight/flightnumber/lavatory/second/state

1. 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-mqtt는 파이썬의 mqtt 라이브러리이다.
  + [참고자료](https://www.eclipse.org/paho/index.php?page=clients/python/index.php)

$ pip3 install paho-mqtt

* MySQL-connector 설치
  + python을 mysql DB와 연결해주는 라이브러리이다.
  + [참고자료](https://www.mysql.com/products/connector/)

$ pip3 install paho-mqtt

1. 파이썬 코드를 이용해 DB에 접속하기

**import** **paho.mqtt.client** **as** **mqtt** *# mqtt 라이브러리*

**import** **mysql.connector** **as** **mariadb** *# db연결 라이브러리*

NALDA\_db = mariadb.connect(host="nalda-db.c2k2xt6qu8x0.ap-northeast-2.rds.amazonaws.com",user='NALDA\_admin',password='Blueskin741^',database='NALDA\_db') *# NALDA\_db와 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** *# db연결 라이브러리*

NALDA\_db = mariadb.connect(host="nalda-db.c2k2xt6qu8x0.ap-northeast-2.rds.amazonaws.com",user='NALDA\_admin',password='Blueskin741^',database='NALDA\_db') *# NALDA\_db와 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)