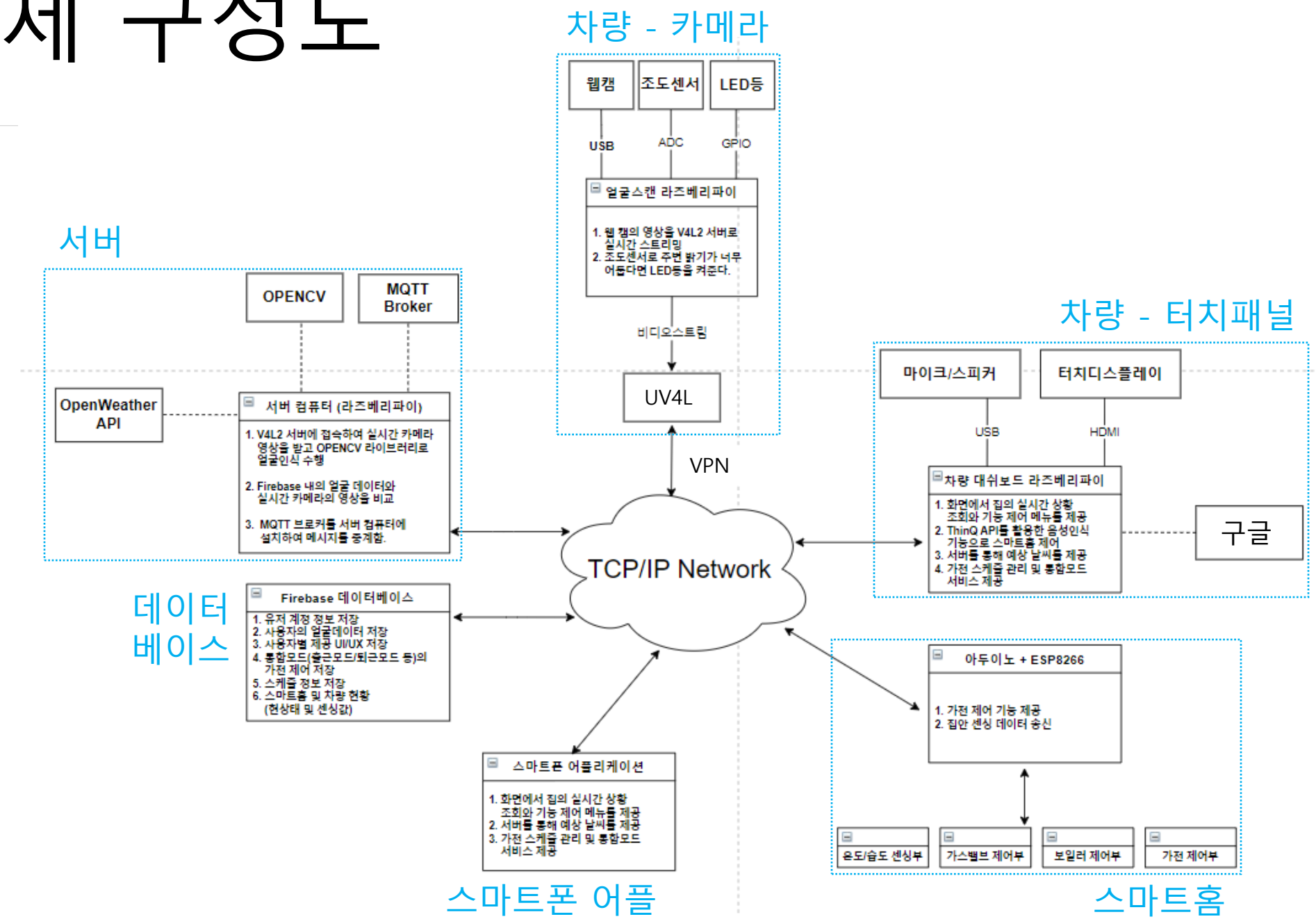
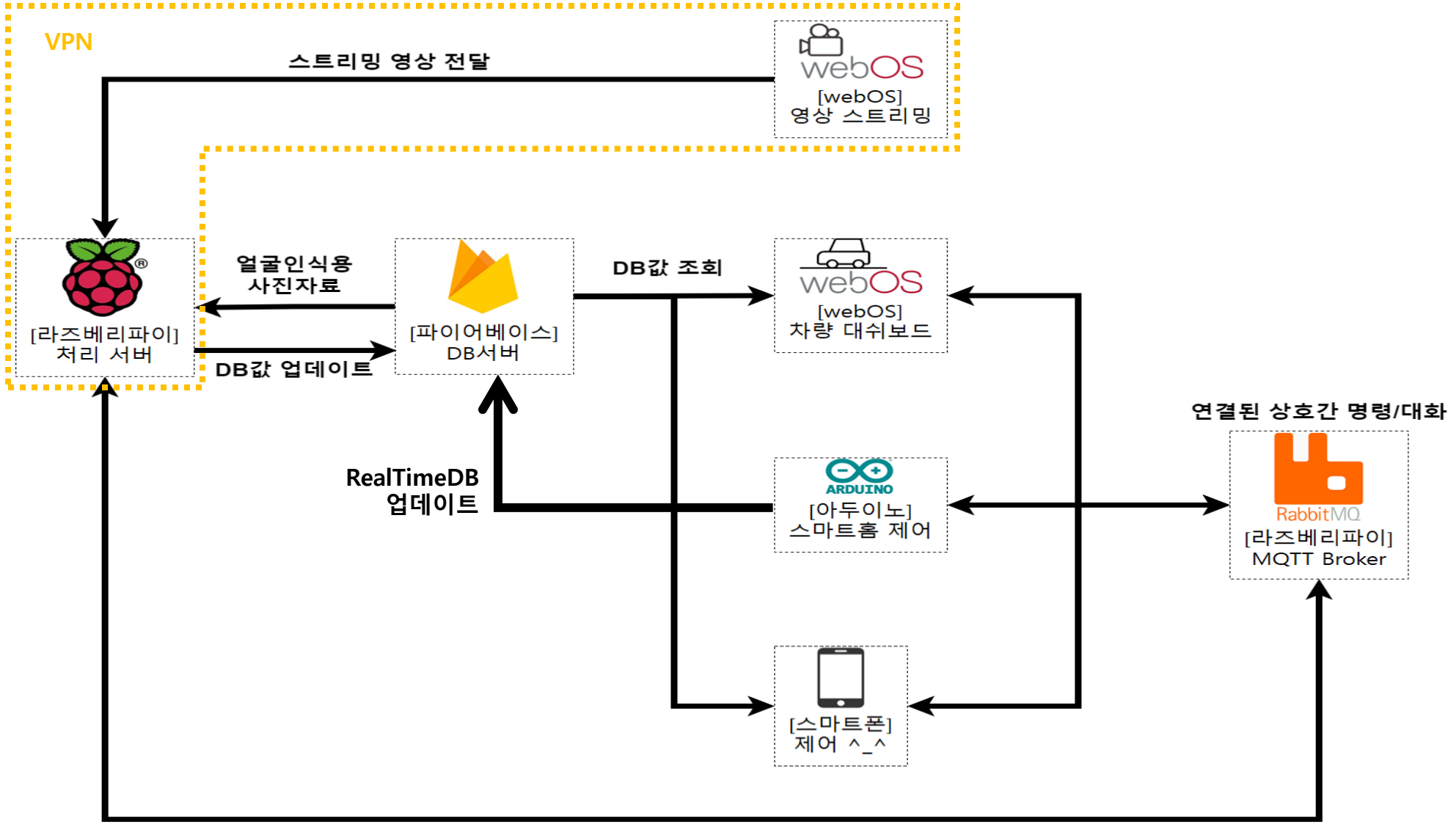
임베디드SW경진대회
WebOS Car2SmartHome

삼아아팀 – 최현식, 박승운, 이준호

전체 구성도


차량 - 카메라
웹캠
조도센서
LED등
USB
ADC
GPIO
얼굴스캔 라즈베리파이
1. 웹 캠의 영상을 V4L2 서버로 실시간 스트리밍
2. 조도센서로 주변 밝기가 너무 어둡다면 LED등을 켜준다.
비디오스트림
UV4L
VPN
서버
OPENCV
MQTT Broker
OpenWeather API
서버 컴퓨터 (라즈베리파이)
1. V4L2 서버에 접속하여 실시간 카메라 영상을 받고 OPENCV 라이브러리로 얼굴인식 수행
2. Firebase 내의 얼굴 데이터와 실시간 카메라의 영상을 비교
3. MQTT 브로커를 서버 컴퓨터에 설치하여 메시지를 중계함.
차량 - 터치패널
마이크/스피커
터치디스플레이
USB
HDMI
차량 대쉬보드 라즈베리파이
1. 화면에서 집의 실시간 상황 조회와 기능 제어 메뉴를 제공
2. ThinQ API를 활용한 음성인식 기능으로 스마트홈 제어
3. 서버를 통해 예상 날씨를 제공
4. 가전 스케줄 관리 및 통합모드 서비스 제공
구글
TCP/IP Network
데이터 베이스
Firebase 데이터베이스
1. 유저 계정 정보 저장
2. 사용자의 얼굴데이터 저장
3. 사용자별 제공 UI/UX 저장
4. 통합모드(출근모드/퇴근모드 등)의 가전 제어 저장
5. 스케줄 정보 저장
6. 스마트홈 및 차량 현황 (현상태 및 센싱값)
아두이노 + ESP8266
1. 가전 제어 기능 제공
2. 집안 센싱 데이터 송신
온도/습도 센싱부
가스밸브 제어부
보일러 제어부
가전 제어부
스마트폰 어플리케이션
1. 화면에서 집의 실시간 상황 조회와 기능 제어 메뉴를 제공
2. 서버를 통해 예상 날씨를 제공
3. 가전 스케줄 관리 및 통합모드 서비스 제공
스마트폰 어플
스마트홈

네트워크 구성도


VPN
스트리밍 영상 전달
[webOS]
영상 스트리밍
[라즈베리파이]
처리 서버
얼굴인식용
사진자료
DB값 업데이트
[파이어베이스]
DB서버
DB값 조회
[webOS]
차량 대쉬보드
RealTimeDB
업데이트
[아두이노]
스마트홈 제어
[스마트폰]
제어 ^_^
연결된 상호간 명령/대화
[라즈베리파이]
MQTT Broker

업무분담

현식

- 서버 구축 / 개발
- 데이터베이스 구축
- 얼굴인식용 영상 스트리밍
- 얼굴 인식 개발 보조

승윤

- 음성인식 구현
- 안드로이드 앱 개발
- 하드웨어 개발

준호

- 얼굴 인식 기능 개발 (OpenCV)
- 웹 프론트앤드 개발

서버 구축

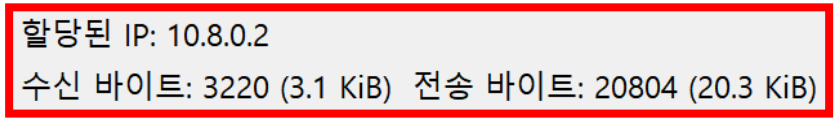
- 처리서버 : 라즈비안 (라즈베리파이4 8GB)
- MQTT 브로커 : RabbitMQ
- DataBase : 파이어베이스

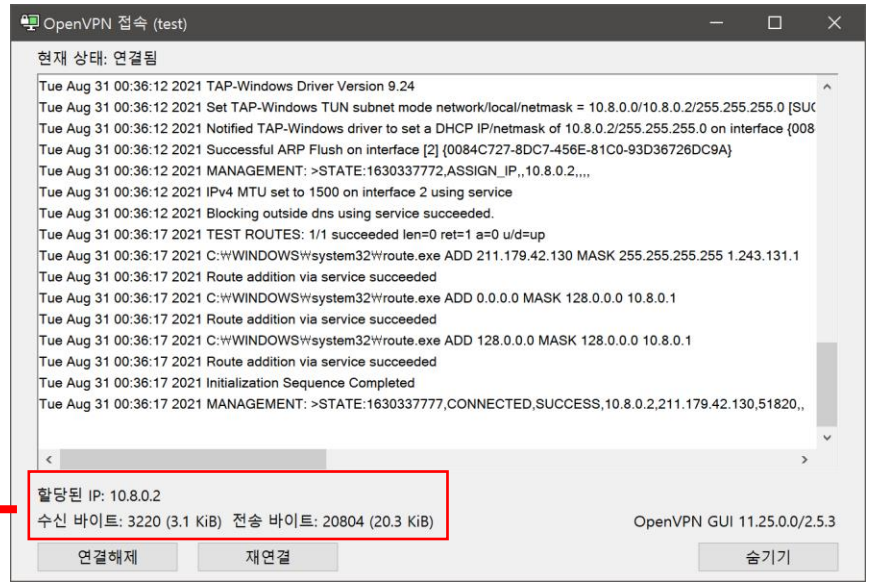
처리 서버

- 기기: 라즈베리파이4 8GB (라즈비안)
- openVPN 서버 구성 (얼굴인식용 카메라측과 직접연결을 위함)
- 각 상황별 처리 프로그램 (Python 및 Shell Script)
  - 날씨 API를 통한 실시간 날씨를 확인하는 로직
  - MQTT 메세지별 처리 로직
  - 파이어베이스 값 확인 및 업데이트
  - oepnCV를 통한 얼굴인식 진행 (→ 다른 장에서 다룸)

openVPN 서버

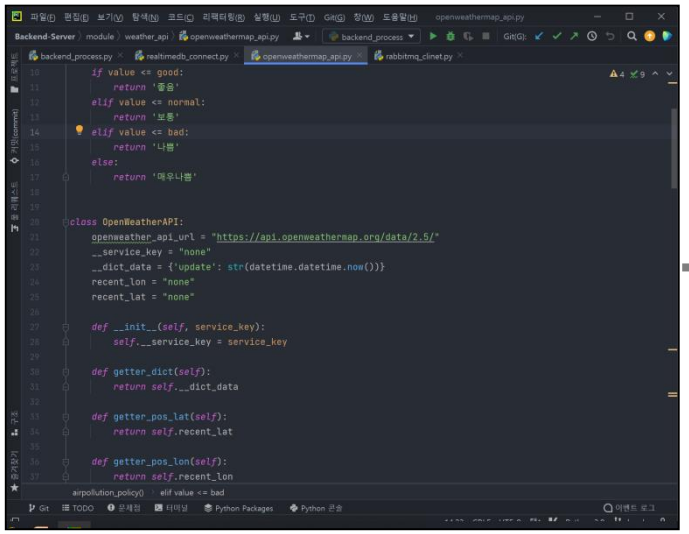
- 얼굴인식용 카메라측과 안전한 직접연결을 위하여 VPN 구성
- 10.8.0.n 대역에 각 기기별 고정된 IP를 부여 (인증서 기준)
  - 10.8.0.1 : Server
  - 10.8.0.2 : Camera Device





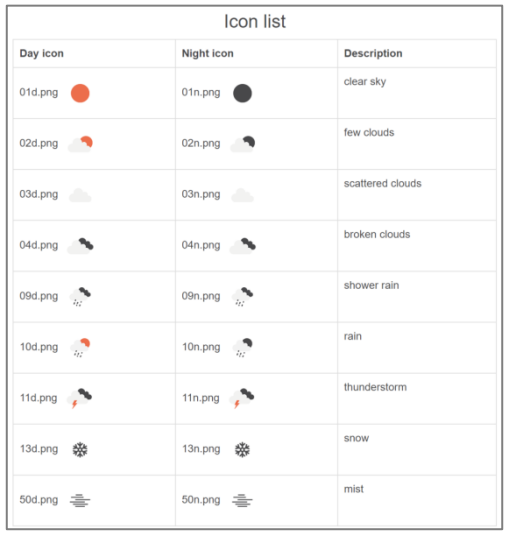
날씨 API

- Open Weather Map의 API를 통하여 실시간 날씨 데이터 수집
- 수집한 데이터를 실시간 DB에 업데이트하여 다른 기기에 공유



```
if value <= good:
    return '좋음'
elif value <= normal:
    return '보통'
elif value <= bad:
    return '나쁨'
else:
    return '매우나쁨'


class OpenWeatherAPI:
    openweather_api_url = "https://api.openweathermap.org/data/2.5/"
    __service_key = "none"
    __dict_data = {'update': str(datetime.datetime.now())}
    recent_lon = "none"
    recent_lat = "none"

    def __init__(self, service_key):
        self.__service_key = service_key

    def getter_dict(self):
        return self.__dict_data

    def getter_pos_lat(self):
        return self.recent_lat

    def getter_pos_lon(self):
        return self.recent_lon
```

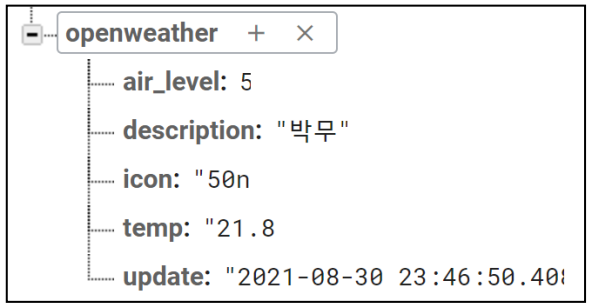
[API 요청 프로그램]



Icon list

Day icon	Night icon	Description
01d.png	01n.png	clear sky
02d.png	02n.png	few clouds
03d.png	03n.png	scattered clouds
04d.png	04n.png	broken clouds
09d.png	09n.png	shower rain
10d.png	10n.png	rain
11d.png	11n.png	thunderstorm
13d.png	13n.png	snow
50d.png	50n.png	mist

[openweathermap Server]



```
openweather
    air_level: 5
    description: "박무"
    icon: "50n
    temp: "21.8
    update: "2021-08-30 23:46:50.40
```

[DataBase]

MQTT 메시지 송수신 및 동작 처리

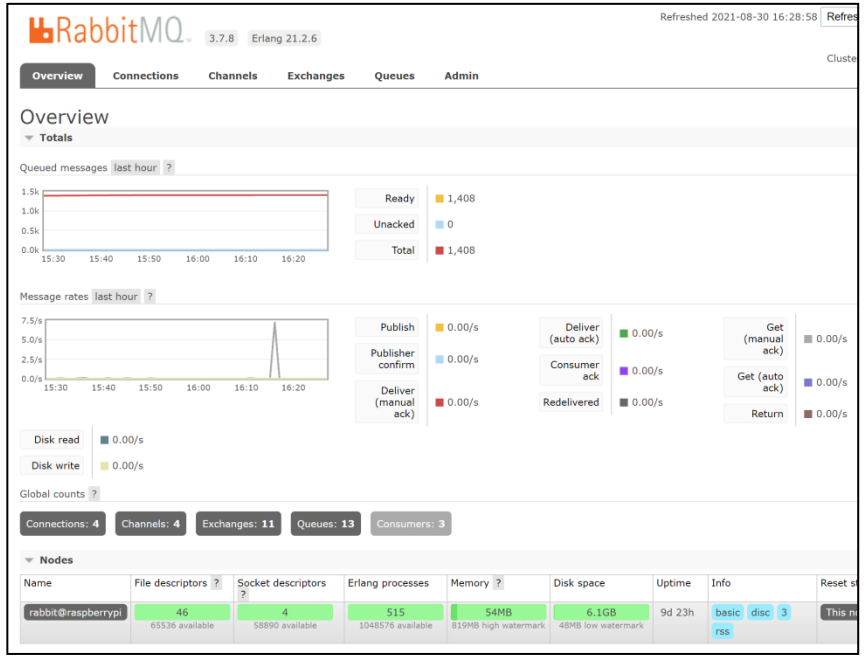
- Server측으로 오는 MQTT 메세지를 처리
  - 회원가입, 로그인, 얼굴인식, 스케쥴 관리 등
- 각 기기 및 API를 통하여 얻은 값을 정제하여 DB에 업데이트

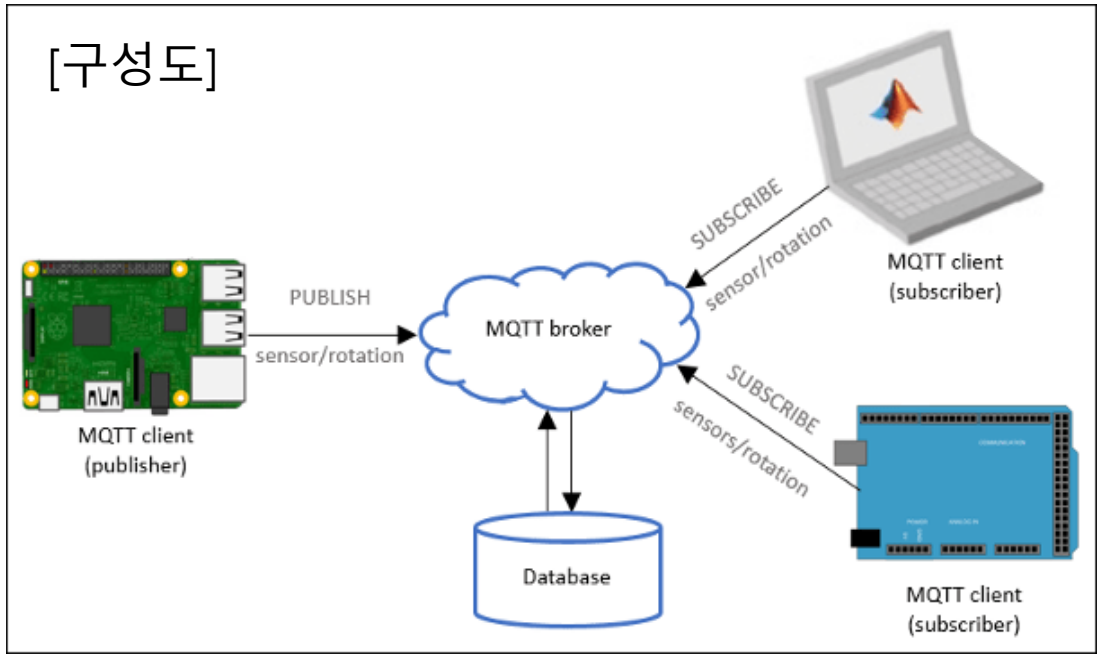
계획 및 개선방향

- 아두이노(스마트홈)측에 전달되는 명령을 확인하여, 해당 동작들이 정상적으로 수행되었는지 체크
- 각 기기(webOS, Android)로 부터 스케쥴 관련 내용을 전달받아 추가/수정하는 프로그램 작성

MQTT Broker

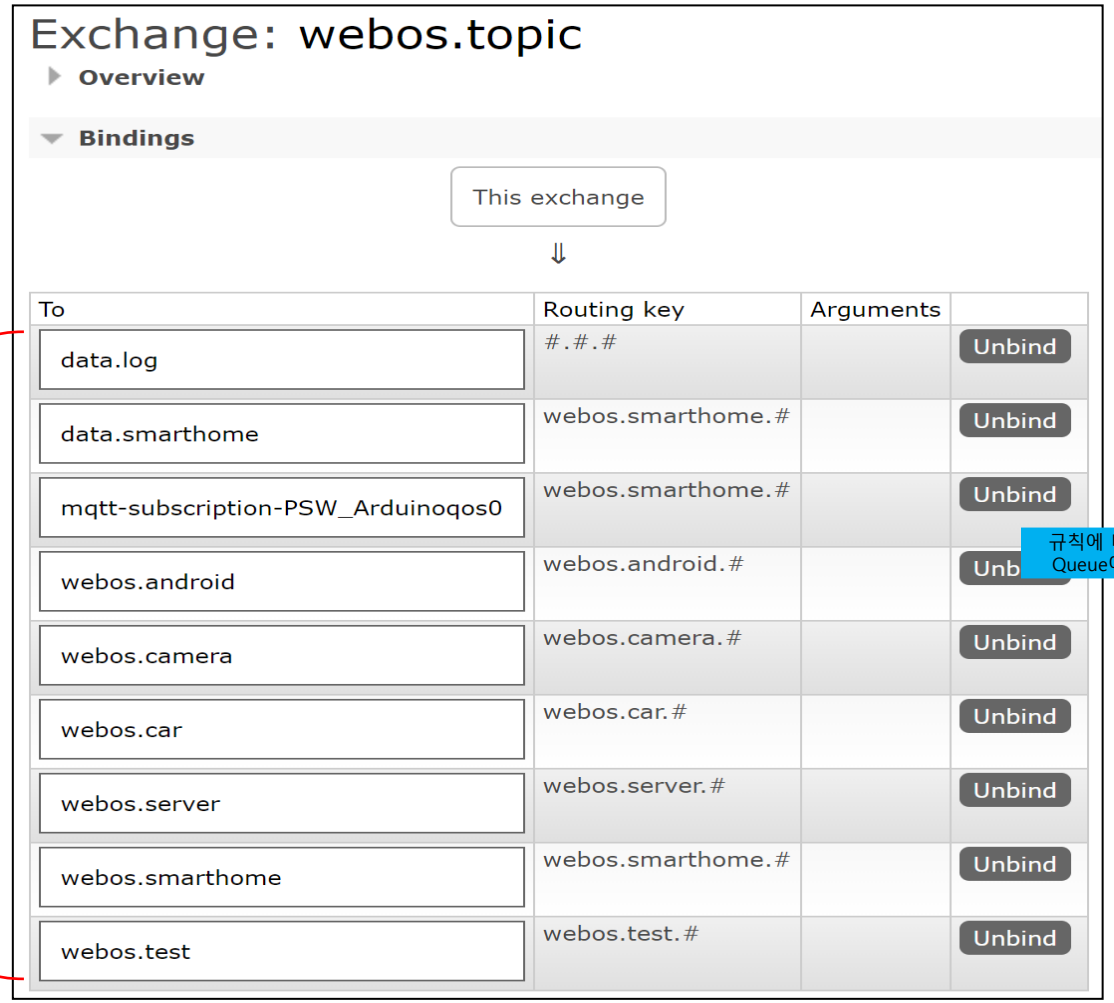
- RabbitMQ (AMQP 0-9-1 / MQTT 3.1.1)
- 사용되는 전체기기(서버(Python), webOS(서비스), 아두이노, 안드로이드)와 통신되는 것을 확인함.





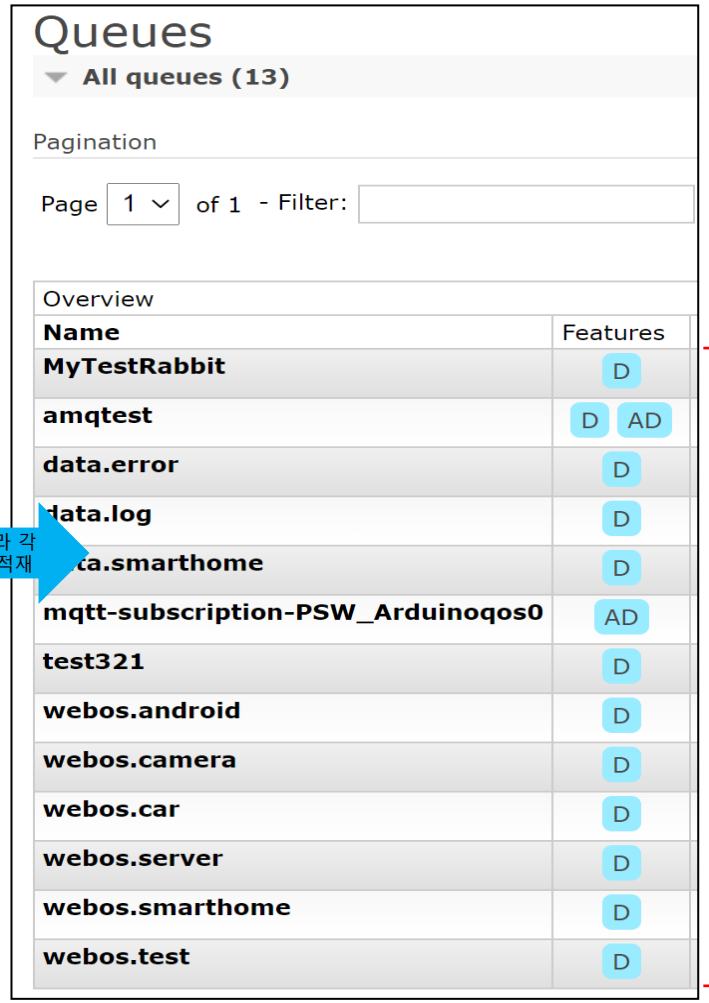
항목구성

메시지
자동분류 규칙



[RabbitMQ Exchange 정보]

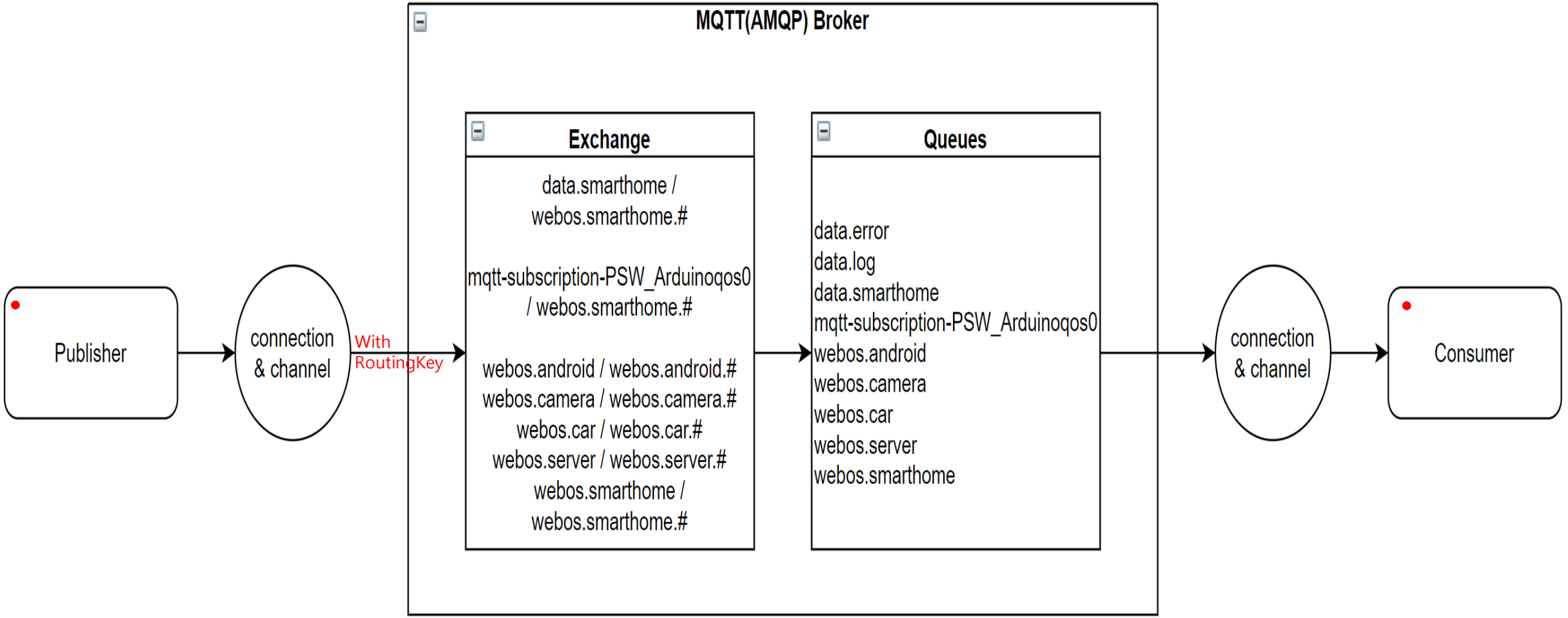
규칙에 따라 각 Queue에 적재



[RabbitMQ Queue 정보]

실제 데이터가
적재되는 Queue

동작구조



- 모든 Client는 Publisher이자 Consumer가 될 수 있음.
  (서버, webOS, 아두이노, 안드로이드)

계획 및 개선방향

- 아두이노측에서 Consume 진행종료시 설정내용이 변동되는 문제 수정 필요 (MQTT-Plugin 옵션 설정)
- QoS(서비스품질) level에 따른 동작속도 및 안정성 비교 진행

파이어베이스(DB)

참조용 데이터 → 계정명, 스케쥴, 사용자별 UI/UX 정보

자주 변경되는 데이터 저장 → 스마트홈 현상태, 센서 정보

얼굴인식용 사진 데이터 저장 → 해당 기반으로 모델 구성

Authentication

각 사용자의 ID(Email)와 PW, UID를 암호화하여 관리

식별자	제공업체	생성한 날짜 ↓	로그인한 날짜	사용자 UID
kim@test.com	✉	2021. 8. 29.	2021. 8. 29.	dcsE1Q8BPXSpqiFfWFp1WKb2GZ...
t@t.com	✉	2021. 8. 24.	2021. 8. 30.	uL5JDoER2BUqEaeTM2jZn7zLOKr2
ccc@test.com	✉	2021. 8. 16.	2021. 8. 16.	VHoW88CMTIWHolPPt4PAu8BRo...
lee@test.com	✉	2021. 8. 16.	2021. 8. 30.	aw0liyvHuUXW1d3AZqzoP9vcsV92
choi@test.com	✉	2021. 8. 13.	2021. 8. 26.	EgPP6gziDVUKGYuZQTHejRXw94...
psw@psw.com	✉	2021. 8. 9.	2021. 8. 10.	NaZowOulHzTjqu0l2PZ3SS3FZTo2
psw1234@naver.com	✉	2021. 8. 9.	2021. 8. 30.	TUwFlZvrnaNsBFB5mDmZdShFd3...
test123@test123.com	✉	2021. 8. 8.	2021. 8. 9.	mCfkGiC85veCIseCzaiBs6VQ41P2
hello@man.hi	✉	2021. 8. 5.	2021. 8. 5.	C3NZCUve8HUH61llXFRrOruqgro1
psw999@test.com	✉	2021. 8. 5.	2021. 8. 5.	dwyWUmIxQMOfkZ0tTXGh6NHr1...
12@12.com	✉	2021. 8. 4.	2021. 8. 4.	HfuN9a9b6rhC2b1i4O5khMZSwS22
park@test.com	✉	2021. 8. 4.	2021. 8. 4.	RLJ7mEoaLZObzVwY7jYbVa2eHh...

FireStore - user_account

각 운전자 정보를 계정 단위로 관리.

Document ID →
(Collection Name)

필드명	타입 (암시적)	설명
UID	**String**	사용자의 고유코드를 저장 (파이어베이스에서 부여) (해당 코드를 기반으로 프리셋 및 사진 탐색)
이름	String	사용자명을 저장
PIN PW	int(10)	앱을 인식을 다시 요청했을 때 사용할 비밀번호 저장

Android 및 webOS는 FireBase를 통해 직접로그인 하므로
비밀번호를 별도 관리하지 않아도됨

FireStore - uiux_preset

각 운전자별로 사용할 UI/UX를 미리 지정.

Document ID →
(Collection Name)

필드명	타입 (암시적)	설명
UID	**String**	**각 사용자의 고유코드**
ui_mode	String	다크모드 / 노안모드 / 유아모드
ux_mode	String	사용자의 사용패턴에 따른 ux 모드를 저장

FireStore - schedule_mode

가전동작에 대한 모드들을 기억함. (기존 홈모드 + 스케쥴 DB 통합)

Document ID → (Collection Name)

Type이 once인 경우

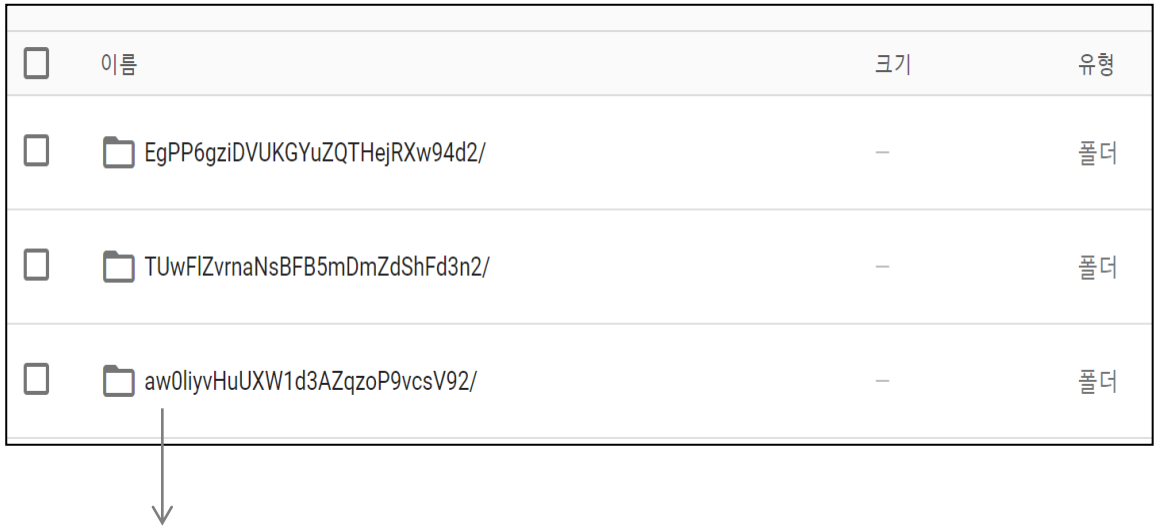
type이 repet인 경우

필드명	타입 (암시적)	설명
Title	String	이름 지정
UID	String(-)	각 사용자의 고유코드
type	String	단발성 작업인지 구분 (once/repet/mode)
Active_date	Timestamp	단발성 작업의 동작시간을 설정함
Enabled	Bool	사용여부 (예약됨)
Daysofweek	Array (Bool)	반복작업의 요일지정 (bool: 일월화수목금토)
Start_time	String	반복작업의 시작 시간 지정 ex)12시05분시작 -> 1205
Devcie_aircon	Array (Bool, number)	동작여부(Bool), ~~설정온도(number)~~, 풍량(number:0~100)
Device_light	Array(Bool, number, String)	동작여부(Bool), 밝기(number:0~100), 색상(string), 동작모드(String)
Device_gas	Array (Bool)	사용여부(Bool)
Device_window	Array (Bool)	열림닫힘여부(Bool)

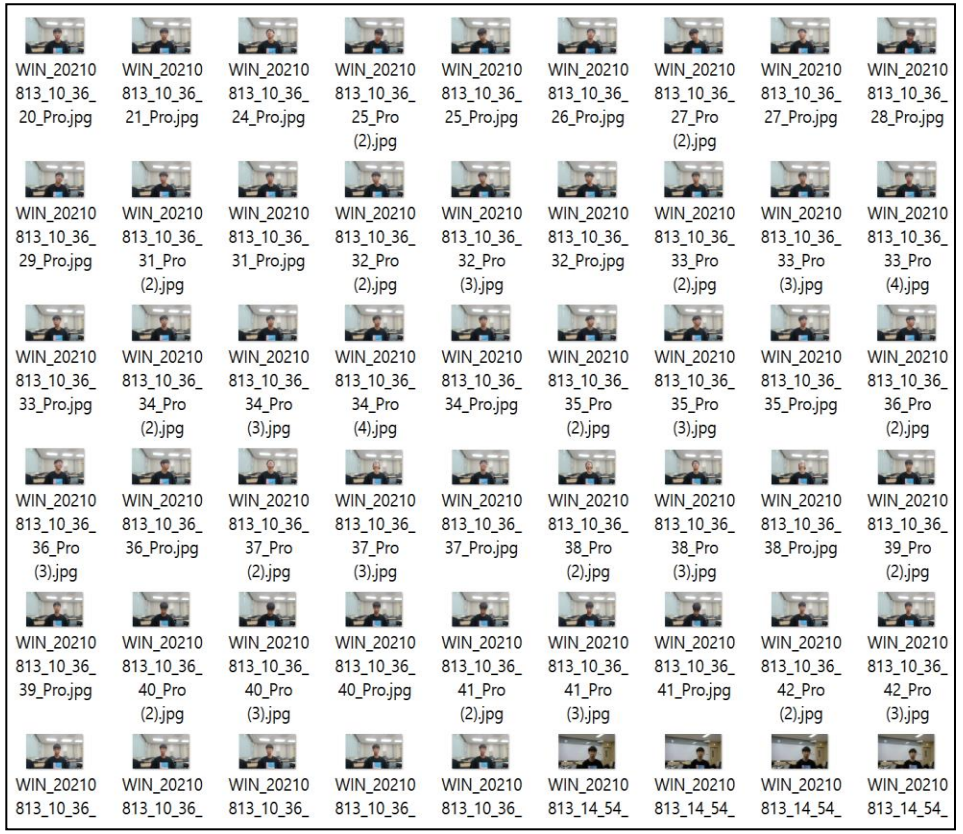
디바이스 동작시 상관없음. 은 그냥 값을 안넘겨주면됨.

Storage

각 사용자의 얼굴인식을 위한 사진을 보관하는 저장소



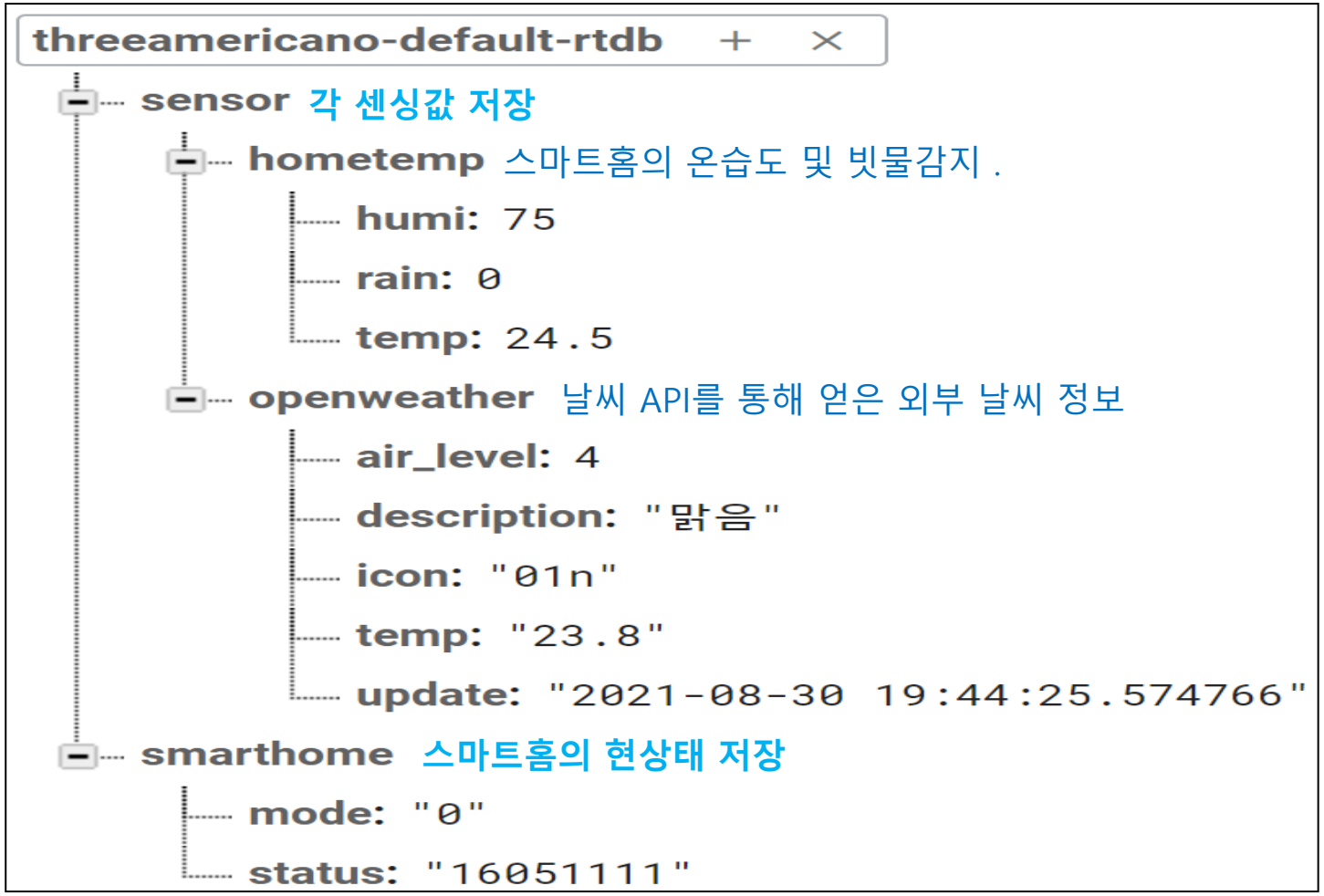
각 사용자의 UID와 동일한 폴더에 사진을 저장



※ 해당 사진들을 기반으로 학습시킨 모델을 얼굴인식시 사용함

RealTime DB

실시간으로 변화하는 데이터 값들을 저장. (센싱값, 스마트홈 현상태, 차량 현상태)



계획 및 개선방향

- DB 변경점 발생시 수정 (각 사용자별 UX 기능 구현 등)

카메라 측 기기

- 기기 : 라즈베리파이3 B+ (라즈비안)
- 웹캠 :로지텍 C310
- 스트리밍 방식 : UV4L을 이용한 웹 스트리밍
- openVPN을 이용하여 서버에서 IP관계없이 접속가능
- 조도센서와 LED 라이트를 사용하여 사용자 환경(어두운 밤)에 자동으로 LED를 점멸함 (얼굴인식률 개선을 위한 밝기조절)

카메라 측 기기 구성

[RabbitMQ]

MQTT 메시지 전달
(얼굴인식 시작/종료)

[카메라측 라즈비안]

현재 밝기값 전달

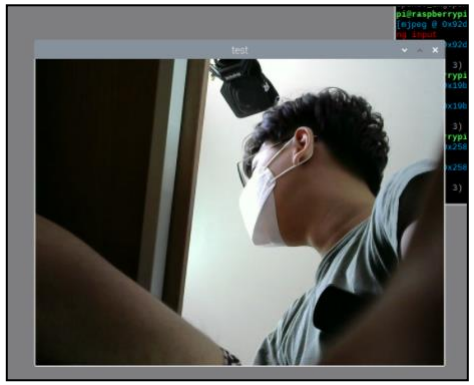
[조도센서]

LED ON/OFF

[LED 스트랩]

[웹캠 C310]

영상 스트리밍



[WEB 스트리밍]

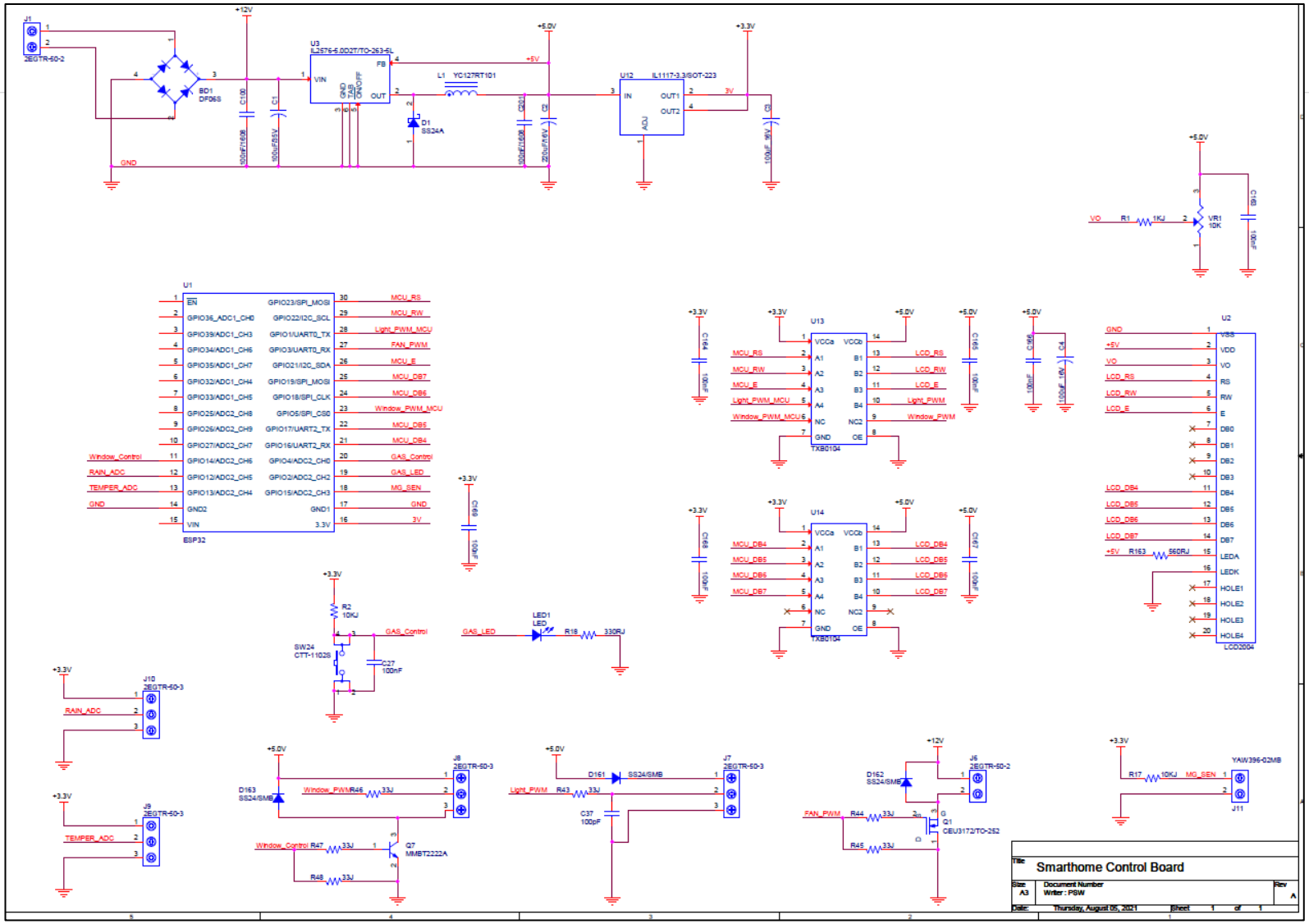
서버에서
영상접근

[openVPN]

계획 및 개선방향

- 인증서를 통해 자동으로 VPN에 접속하도록 프로그램 변경
- 부팅 시 자동으로 Camera 측 Queue에 접근하도록 수정

1. Hardware – Control Board 회로도 설계



0. MCU는 WIFI 통신이 가능한 ESP32로 선정

1. 전원부 설계 (12V -> 5V -> 3.3V)

2. 에어컨 제어 (Cooling PAN)

- PWM 제어, 강도 조절 가능
- 가변저항으로 외부에서 제어 가능

3. 전등 제어 (LED Strip Bar)

- PWM 신호를 다음 LED로 넘겨주는 방식
- 밝기 조절과 슬립 모드, 무드등 모드 등의 여러 모드 설정이 가능함.

4. 가스 밸브 (LED, Switch로 구성)

- 바깥 Switch를 누르면 가스밸브 (LED)가 ON/OFF

5. 창문 (서보모터)

- PWM을 통해 제어
- 각도를 알 수 없으므로 자석감지센서로 자석이 감지되면 닫힌상태, 미감지시 열린상태로 인식하여 서보모터를 제어함.

6. 온습도센서

- 주기적으로 집의 온습도를 체크하여 파이어베이스에 업로드

7. 빗물감지센서

- 센서에서 일정 이상의 ADC가 감지되면, 비가 온다고 감지하여 창문을 닫도록 함.

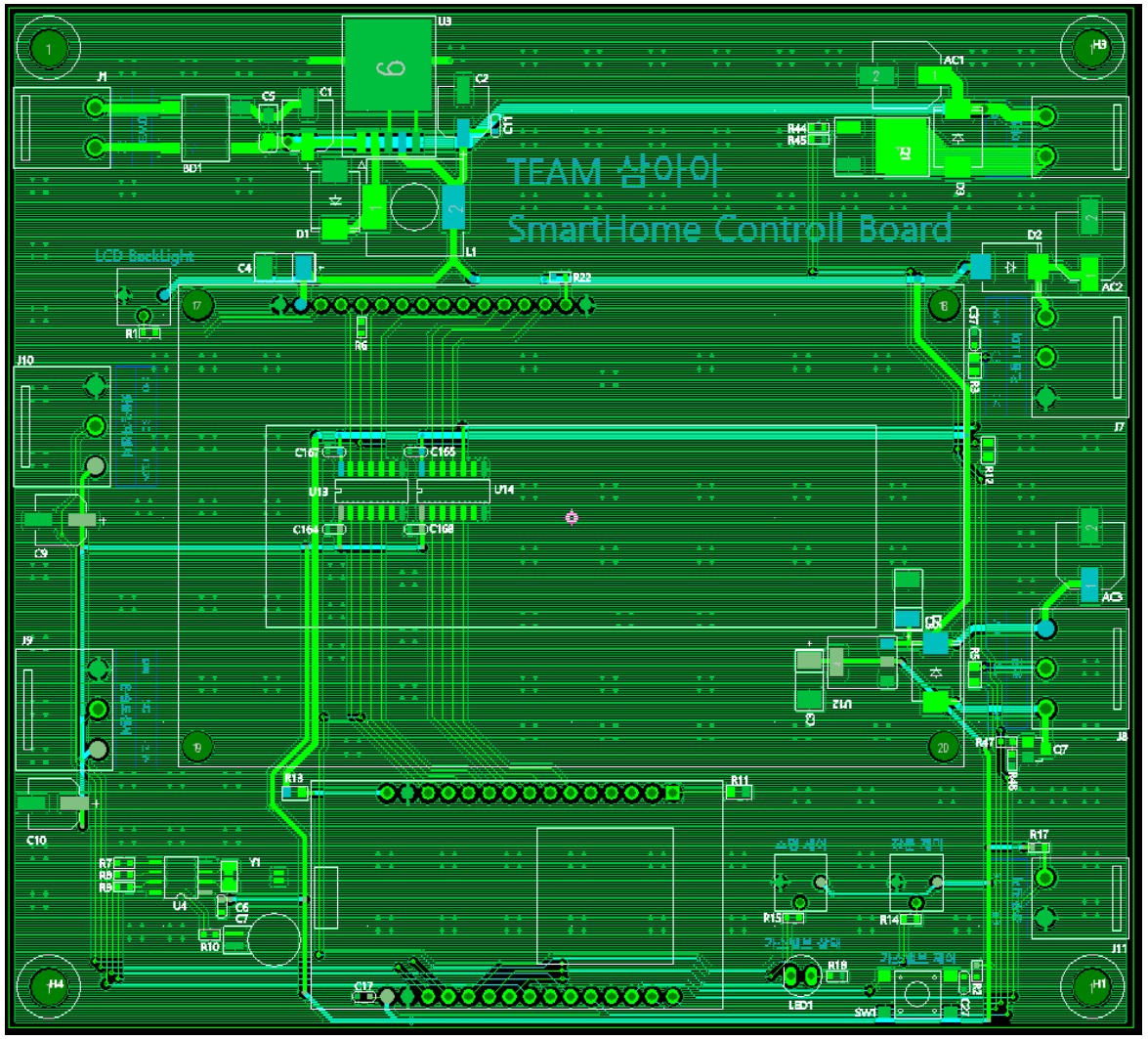
8. RTC IC

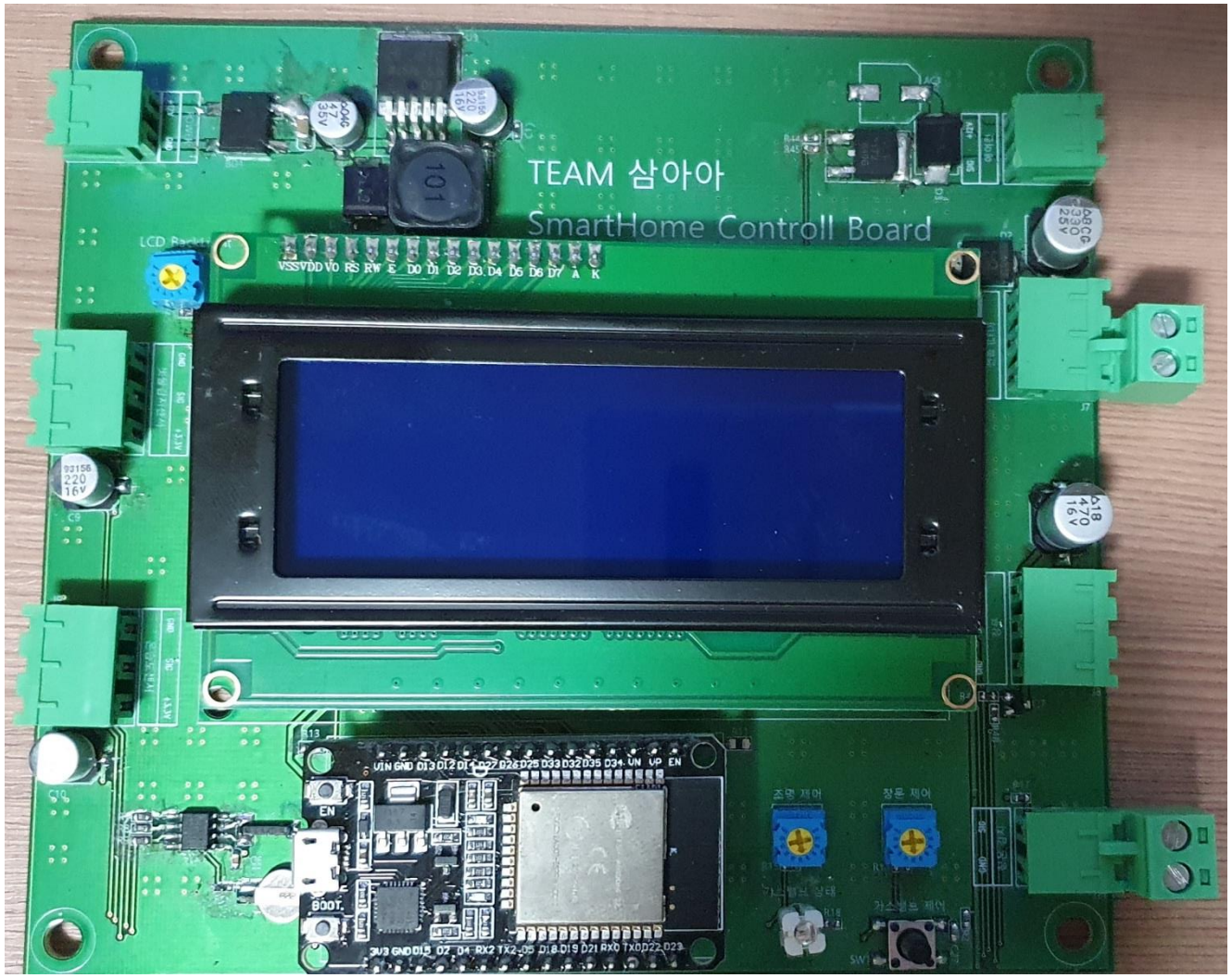
- 시간 별 스케줄 관리를 위해 실장

9. LCD

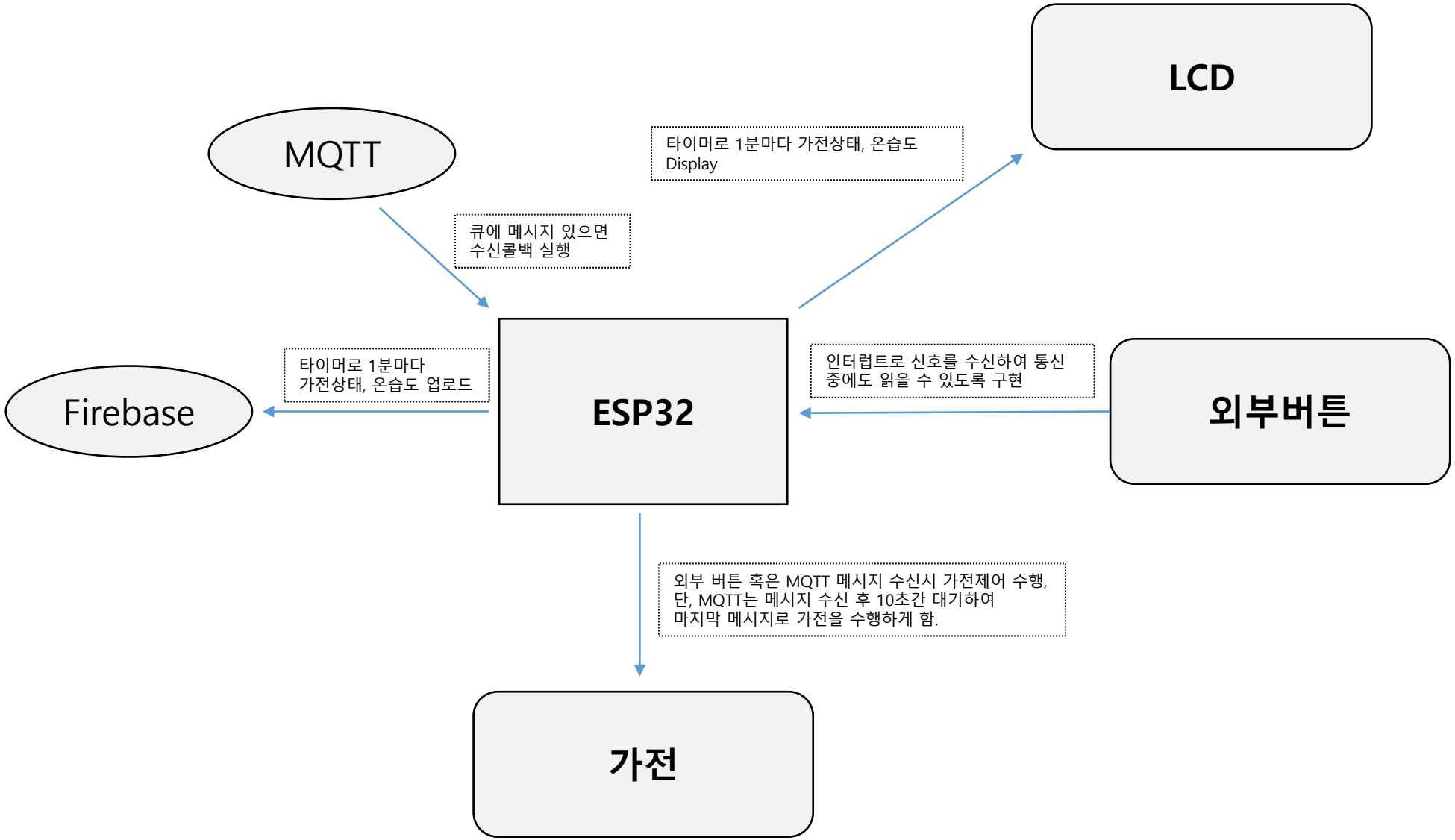
- 현재 가전 상태와 센서 값을 Display

1. Hardware – Control Board PCB 설계





1. Hardware – Control Board 구성도



1. Hardware – 현재 진행 상황

☒ PCB 설계 및 발주 완료

☒ Firebase에 Home 센싱데이터 및 가전상태 업로드

☒ Rabbit MQ의 MQTT 메시지 수신

☒ 에어컨 (쿨링팬) 제어 기능 (PWM)

☒ 가스밸브 제어 (Interrupt)

☒ 전등 색상 및 밝기 제어

☐ 전등 모드 생성 및 제어

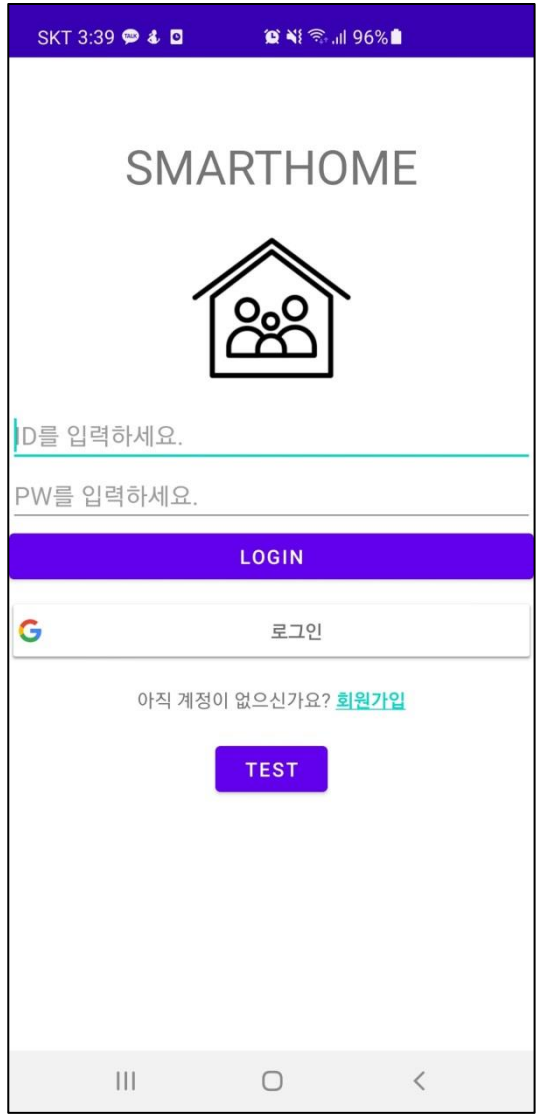
☐ 창문제어

☐ 창문 하드웨어 구성

☐ LCD에 가전상태, 센싱데이터 디스플레이

진행률 : 80% 완료

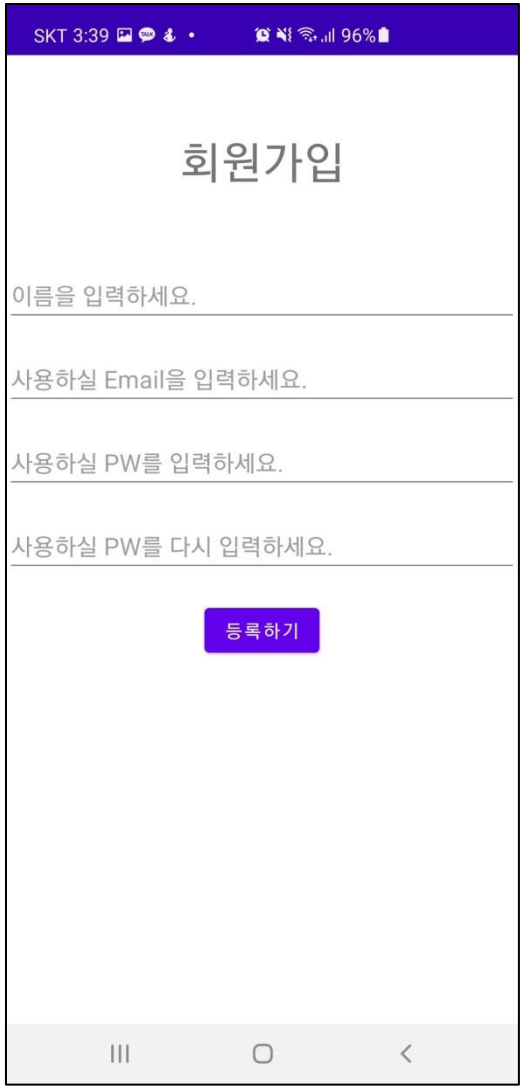
2. Android App – 구현 완료



1. Login 화면

- ID와 Password를 입력하여 Firebase에 인증 요청
- 가입된 사용자일시 Firebase에서 UID를 수신하고, 해당 값을 Rabbit MQ로 송신하여 이름을 수신받음.
- 가입되지 않은 사용자라면, 회원가입을 눌러 회원가입이 가능함.

2. Android App – 구현 완료



2. Signup 화면

- 사용자의 이름과 사용할 Email, Password를 입력하여 회원가입 진행
- 정상적으로 등록 완료시 계정은 Firebase에서 관리되며, 이름 데이터는 서버로 송신하여 관리하도록 함.
- 회원가입 된 데이터는 webOS 앱과 android 앱 모두 로그인이 가능함.

2. Android App – 구현 완료

3. Main 화면

ㄱ. 로그인 완료시 수신한 이름으로 “~~님 안녕하세요“ 표시

ㄴ.서버가 주기적으로 Firebase에 업로드하는 날씨데이터 화면에 디스플레이

ㄷ. 스마트홈 보드가 Firebase에 업로드하는 센싱데이터를 수신하여 디스플레이

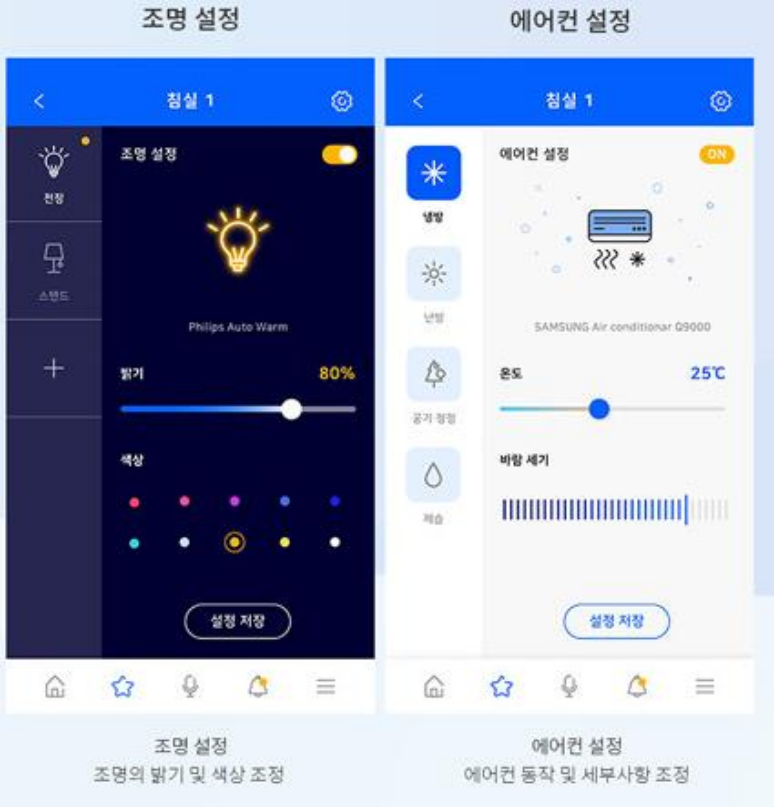
ㄹ. 가전을 한 번에 쉽게 제어 할 수 있는 모드 제어

ㅁ. 가전을 개별로 각각 제어 할 수 있는 개별 제어

- 모드나 가전제어를 클릭하면 해당 제어 프로토콜을 MQTT 서버로 송신하고, 스마트홈 보드가 이를 수신하여 해당하는 동작을 수행한다. (구현 완료)

- 파이어베이스 리스너를 통해 온습도, 날씨, 가전상태등이 새로 업로드 되면, 해당 정보에 맞게 UI / UX를 업로드 한다. (구현 완료)

2. Android App – 구현 예정



4. 가전 개별 제어 설정

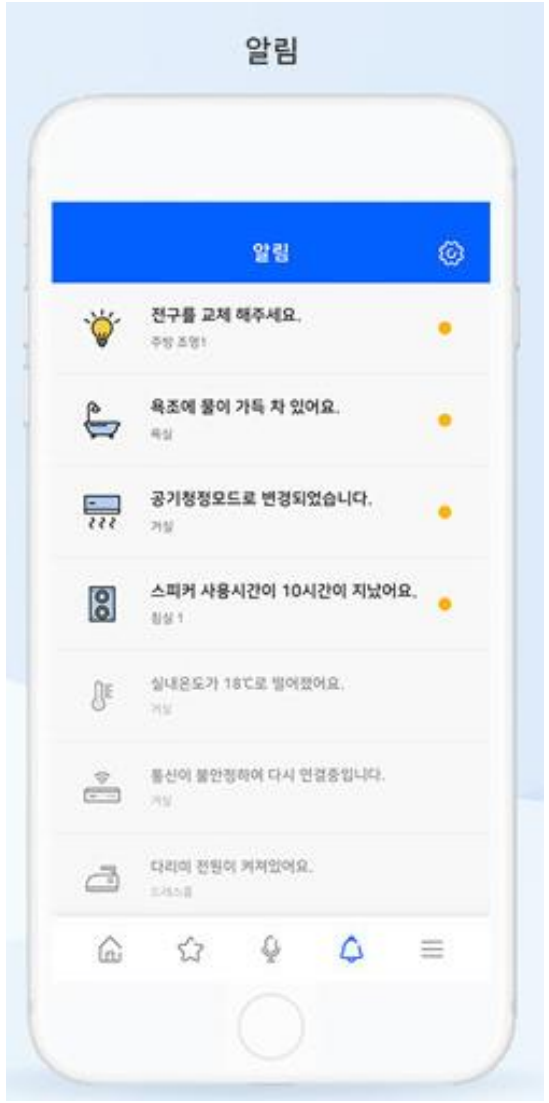
- 에어컨, 조명등 밝기, 색상, 모드 등 자세하게 제어가 가능하도록 세부 제어 메뉴 추가 예정

2. Android App – 구현 예정

5. 스케줄 및 모드 설정

- 스케줄 기능은 알람처럼 원하는 시간에 실행할 모드 혹은 가전제어를 수행할 수 있도록 구현 예정
- 스케줄 반복 여부를 체크하여 단발성인지 반복성인지 체크가 가능함.
- 모드 설정에서 해당 모드에서 가전을 어떤 것을 제어할지 설정 가능

2. Android App – 구현 예정



6. 실행 완료 알림

- 스마트홈 보드에서 명령을 정상적으로 수신하여 동작을 수행하면 알림을 MQTT로 송신한다.
- 안드로이드 앱은 해당 데이터를 수신하여 알림창에서 보여준다.

2. Android App – 진행률

☒ Firebase Authentication 기능을 활용한 로그인 기능 구현

☒ 회원가입 기능 구현

☒ 로그인 후 서버와 통신하여 사용자의 이름 수신하기

☒ Firebase의 데이터 수신하기 (날씨, 가전상태, 온습도)

☒ Firebase 리스너 구현

☒ MQTT Publish로 스마트홈 가전 제어 구현

☐ 자세한 개별제어 메뉴

☐ 모드 / 스케줄 메뉴

☐ 동작 완료 명령 수신 및 디스플레이

진행률 : 70% 완료

3. 음성인식

- ThinQ API를 이용하여 구현 – Stream 형태로 mp3가 재생되지 않아 실패

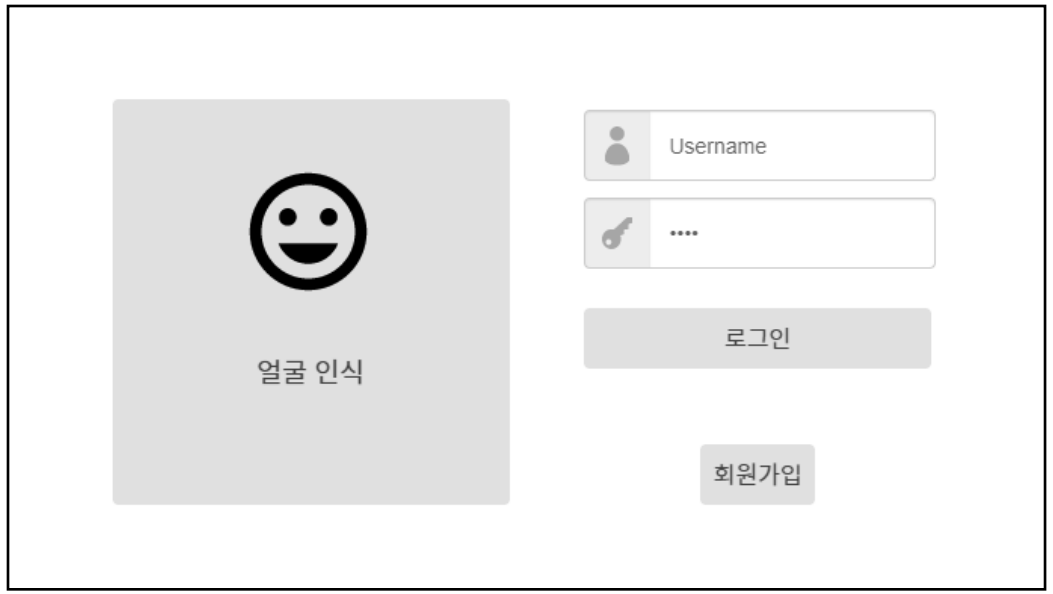
- Google Assistant를 사용하여 구현완료 – 간단한 문답 가능

- 현재 키워드를 이용하여 가전제어 구현 중..

- 개별 가전 제어 -> 모드 제어 -> 스케줄 제어로 구현 예정

진행률 : 30% 완료

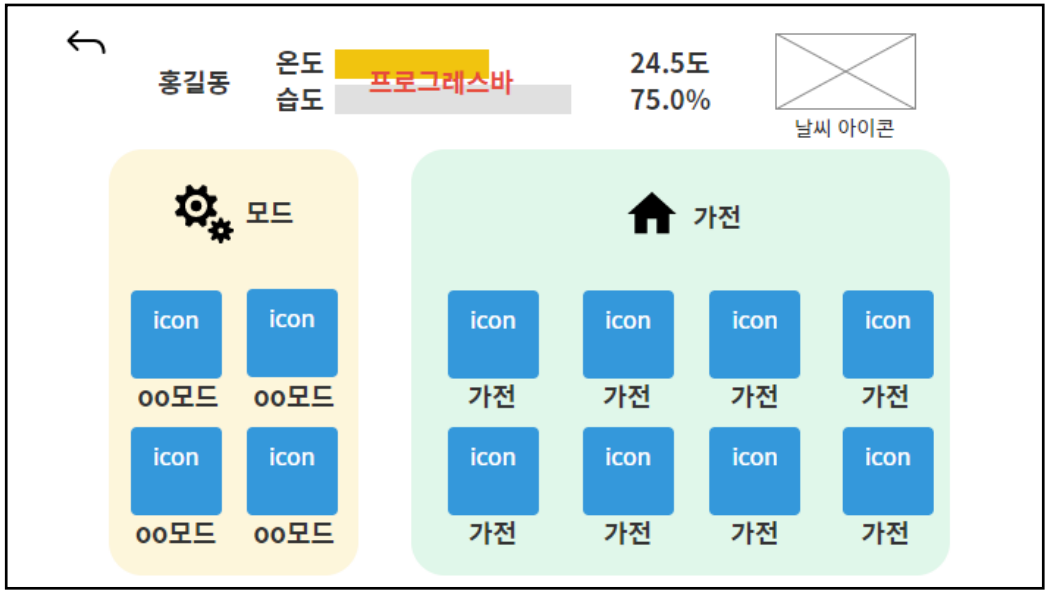
Web UI/UX 설계



1. 로그인 페이지 (구현 완료)
- 웹 앱 처음 실행 시 얼굴인식 로그인 요청
- 얼굴인식 실패 시 해당 페이지로 넘어온다.
- 왼쪽 얼굴인식 버튼 선택 시 MQTT를 사용하여 서버에 얼굴인식 시작을 요청한다.
- 오른쪽 이메일 로그인으로 사용자 인증을 진행할 수 있다.

2. 회원가입 페이지 (구현 예정)
- 회원가입 버튼 선택 시 이메일 회원가입을 진행할 수 있다.
- 이메일 비밀번호로 firebase 인증 계정을 생성할 수 있다.

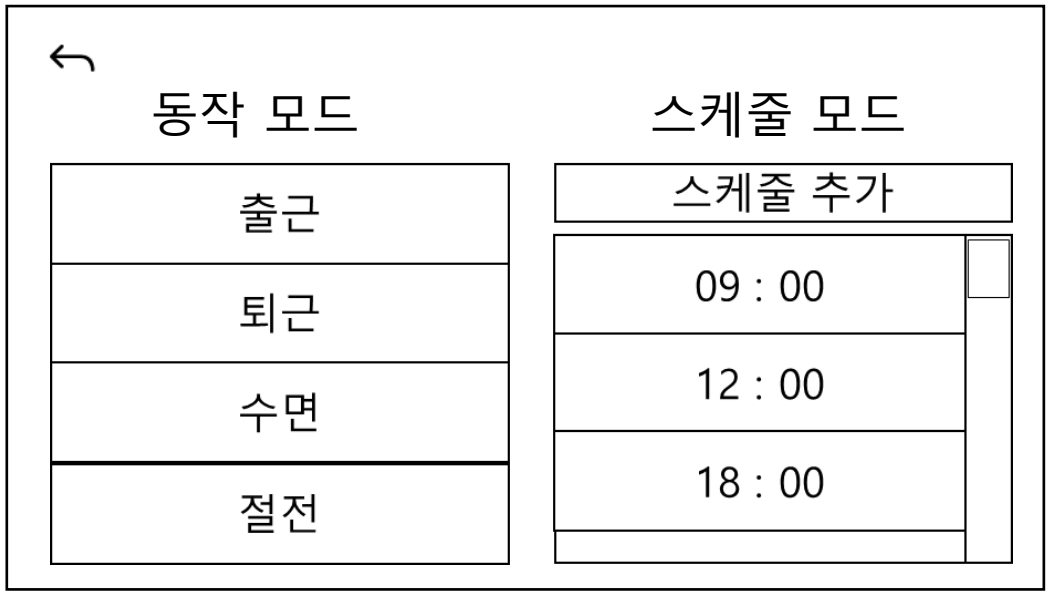
Web UI/UX 설계



1. 홈 페이지 (구현 예정)

- 로그인한 사용자 및 온,습도 날씨를 표시한다.
- 모드 제어 버튼을 두어 특정모드로 가전을 제어할 수 있도록 한다.
- 가전 제어 버튼을 두어 개별적으로 가전을 제어할 수 있도록 한다.
- 모드 혹은 가전의 아이콘을 선택하면 해당 설정으로 들어간다.

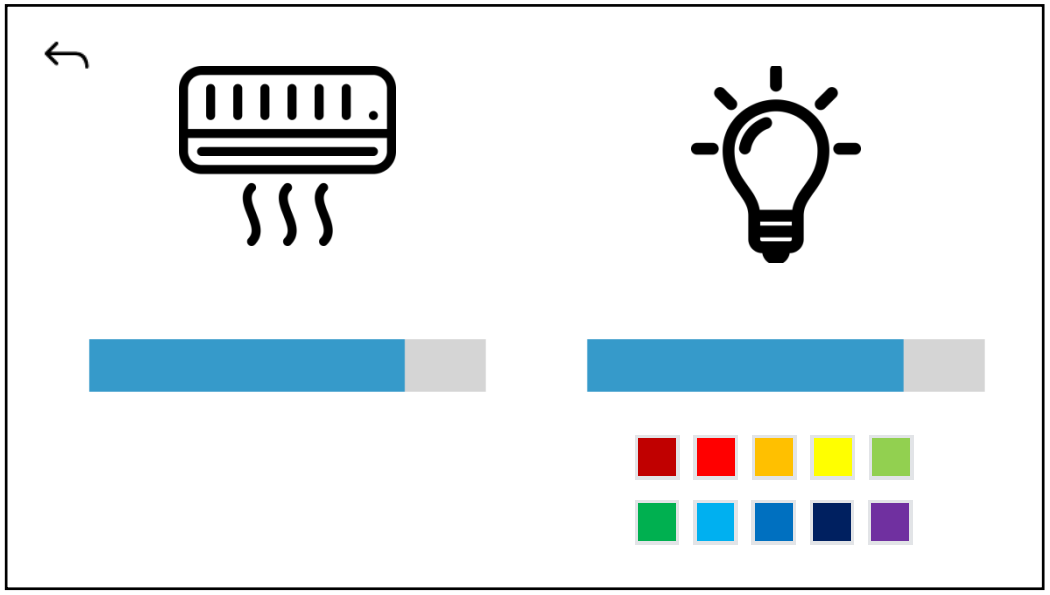
Web UI/UX 설계



1. 모드 설정 페이지 (구현 예정)

- 모드 및 스케줄을 설정할 수 있다.
- 모드는 단발성으로 여러가지 가전의 동작을 한번에 제어할 수 있다.
- 스케줄 기능으로 원하는 시각에 가전을 원하는 동작을 실행시키도록 한다.

Web UI/UX 설계



1. 개별 가전 설정 페이지 (구현 예정)

- 가전의 개별 제어 및 에어컨 세기, 조명의 밝기, 색상 등의 자세한 설정을 가능하게 함

Web UI/UX 설계

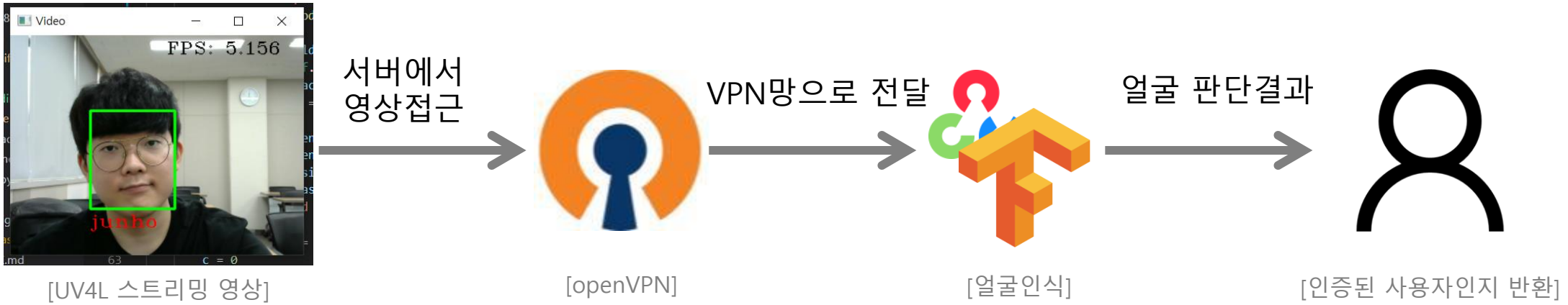
진행률

- ▣ Firebase Authentication 기능을 활용한 로그인 기능 구현
- ▣ 로그인 후 서버와 통신하여 사용자의 이름 수신하기
- ▣ MQTT Publish로 스마트홈 가전 제어 구현
- ▣ JS 서비스 연동 구현
- ▣ 페이지 구현
- ☐ 회원가입 기능 구현
- ☐ Firebase의 데이터 수신하기 (날씨, 가전상태, 온습도)
- ☐ Firebase 리스너 구현
- ☐ 자세한 개별제어 메뉴
- ☐ 모드 / 스케줄 메뉴
- ☐ 동작 완료 명령 수신 및 디스플레이

진행률 : 45%

얼굴인식 구현

- openCV와 Tensorflow를 이용하여 얼굴인식 진행
- UV4L을 통해 스트리밍 되는 영상을 불러와서 이용
- VPN 구성을 통해 UV4L 스트리밍 기기로 직접 연결



얼굴인식 구현

1. 얼굴 학습 (구현 완료)

사진 촬영

다양한 각도, 많은 사진(최소 30장 이상)
으로 정확도를 올릴 수 있다.

사진 조정

사진에서 얼굴을 추출한다.

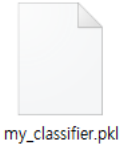
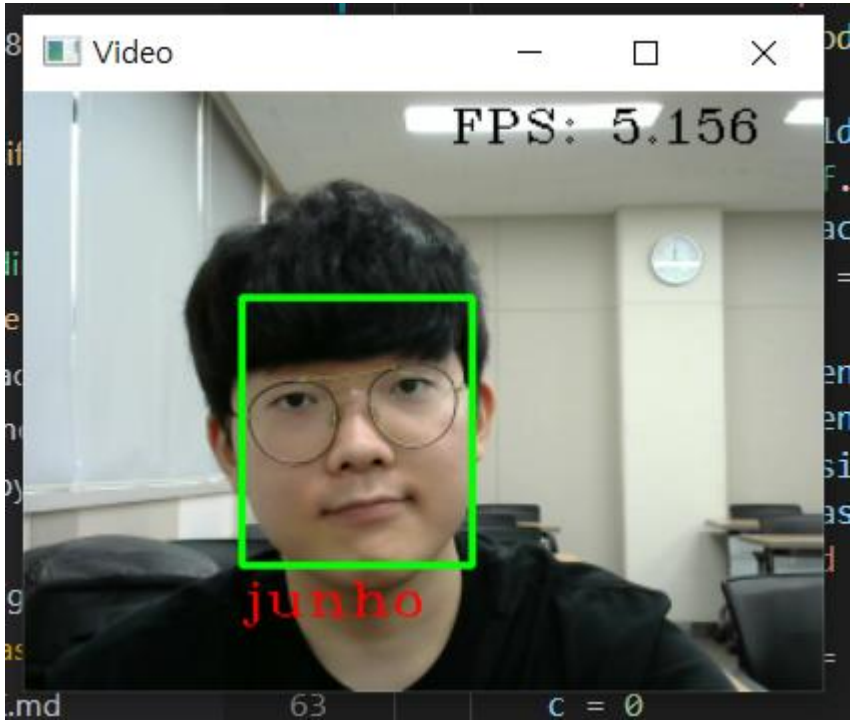


사진 학습

추출한 사진과 미리 학습된
FaceNet 모델과 함께
Crown 모델을 생성한다.

얼굴인식 구현

2. 얼굴 인식 (구현 완료)



- 얼굴 인식 모델은 tensorflow 기반의 FaceNet 모델을 사용
- FaceNet 모델은 탐지와 선처리, 특징 추출, 얼굴 매칭 4단계로 얼굴 인식을 진행
- 웹 캠으로 촬영중인 사용자를 인식해 화면에 이름을 띄움
- 프레임별로 인물을 추측하고 count 변수를 증가
- 학습된 인물의 count 값이 설정한 시간을 넘어가면 얼굴인식이 완료

얼굴인식 구현

진행률

- ▣ 사진 조정
- ▣ 사진 학습
- ▣ 학습 모델링 생성
- ▣ 사용자 인식
- ☐ 서버 연동

진행률 : 80%

문의사항

- 본선 제출이후 추가 개선 진행 가능 여부
- 파이어베이스 연동 및 MQTT 송수신을 구현하였는데, 이것도 서비스 개발인지
- 얼굴인식 속도가 매우느림 (30초 이상소요)