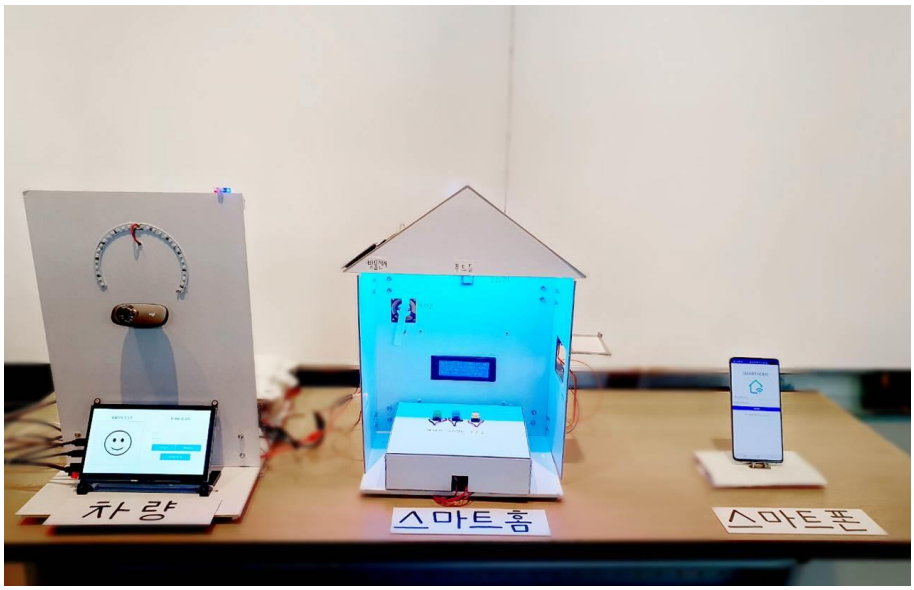
제19회 임베디드SW경진대회 개발완료보고서

[webOS 기반 Car 2 Smart Home 솔루션]

메모 포함[W1]: ※ '파란색 안내 문구'는 삭제하고 검정색 글씨로 총 30page 이내로 작성하여 PDF로 변환하여 제출. (폰트 : 맑은 고딕 / 폰트 크기 : 11pt / 줄 간격 : 160%)

□ 개발 요약

팀 명	삼아아



작품명	SMILE : Smart Mobility Improving Life Experience
작품설명 (요약)	webOS와 스마트폰의 어플리케이션으로 집의 상황을 확인하고 가전을 제어할 수 있는 서비스를 제공한다. 음성과 메시지를 통해 현재 제어되고 있는 가전과 외출 시 주의가 필요한 가스밸브, 창문 등에 대한 사항을 안내 받을 수 있다. 사용자는 다양한 상황에 맞는 가전의 동작스케줄을 일시적으로 혹은 반복적으로 만들어 사용할 수 있다. 또한, 가전과 모드의 실행 여부를 가전 목록에서 확인하여 조회할 수 있다.
소스코드	https://github.com/ThreeAmericano
시연동영상	https://www.youtube.com/watch?v=jM3Lb4GanMw

□ 개발 개요

메모 포함[W2]: ※ 프로젝트 개발 배경, 동기, 목표 등 / 2page 이내로 작성

○ 개발 배경 및 동기

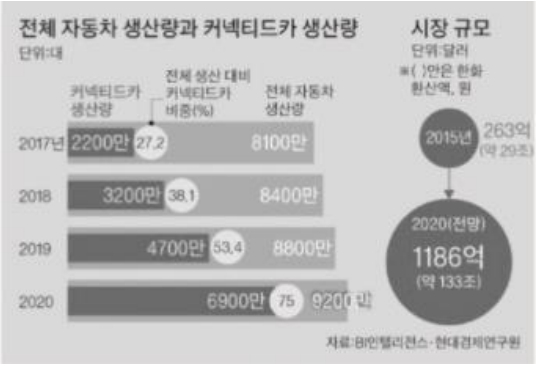
1. 최근 고령화로 인한 독거노인 증가, 비혼과 비자발적 독신 층 증가, 맞벌이와 자녀교육으로 인한 기러기 가족 증가, 타지역으로의 진학과 취업 등 다양한 요인으로 인하여 1인 가구의 증가하고 있다. 행정안전부의 주민등록 인구 통계에 따르면 2020년 12월 기준 1인 가구의 비중은 906만 가구로 지난 20년새에 2배 가까이 증가하였다.

이러한 흐름에 맞춰 소형 가전에 IOT 기술을 접목한 스마트 소형 가전제품의 수요가 꾸준하게 증가하고 있다. 2020년 기준 스마트 가전 보급률은 한국이 15.7%로 세계 1위이며, 뒤를 이어 미국, 영국, 독일, 중국이 위치하고 있다. 특히나 1인 가구의 경우 외출을 하게 되면 집이 비게 되는데 가전이 꺼져 있는지 확인하거나 귀가 전 미리 가전을 실행하는 등의 작업을 수행할 수 있는 스마트 가전의 수요가 높아지고 있다고 한다.

[그림1] 1인가구 규모 추이

2. 무선 네트워크를 통해 언제 어디서든 통신을 할 수 있는 커넥티드 카의 수요도 늘어나고 있다. 2020년 7월까지의 무선 통신서비스 가입 수는 약 300만대에 육박하며, 현재도 계속 늘어나고 있는 추세라고 한다. 이렇게 점점 증가하고 있는 스마트 가전과 커넥티드 카의 보급률에 맞춰 차량 인포테인먼트 시스템의 플랫폼으로 webOS를 탑재하고 편리한 스마트 홈 어플리케이션을 제공한다면 많은 사용자가 사용할 것이라는 생각이 들었다. 아직까지 스마트폰과 연동되어 제공되는 인포테인먼트의 스마트 홈 어플리케이션이 없다. 그러므로 인포테인먼트와 호환성이 높은 스마트폰 어플을 함께 제공하면 사용자의 편의를 더 높일 수 있을 것이라 생각하였다.

따라서 우리 팀은 webOS와 모바일의 스마트 홈 어플리케이션을 함께 개발하여 사용자가 안심하고 외출할 수 있는 스마트 홈 시스템 구축을 목표로 하였다. 대부분의 사람들은 외출 후 "내가 가스밸브를 껐나?" 등의 경험을 겪은 적이 있을 것이다. 때문에 외출해서 어디서든 가전을 확인 및 제어할 수 있는 환경을 제공하면 바깥에서도 안심하고 다른 일을 할 수 있을 것이다. 그리고 1인 가구의 경우에는 귀가 시 불이 꺼지고 차가운 집보다는 스케줄 등의 제어를 통해 귀가 전 미리 환경을 조성해 놓으면 더 안락한 환경을 제공할 수 있을 것이다.



[그림2] 전체 자동차 생산량과 커넥티드카 생산량

○ **개발 목표**

1. 편리성

- 우리 팀은 한눈에 집의 상황을 확인하고 가전을 제어할 수 있는 UI 구성과 기능 제공을 목표로 하였다. 복잡하고 제어하기 불편하다면 사용자가 차량에서 사용하기 꺼려할 수 있기 때문이다. 그래서 TTS 기능을 사용하여 여러 이벤트를 안내하도록 하였고, 커스텀이 가능한 모드 명령을 제공하여 사용자가 쉽게 여러 가전을 한꺼번에 제어할 수 있도록 하였다.

2. 신속성

- 가전의 상황이 즉각적으로 UI에 반영되도록 설계하였다. 사용자가 오프라인이나 스마트폰을 통하여 제어하였는데 그것이 즉각적으로 반영되지 않는다면 문제가 있다고 생각하였기 때문이다. 그래서 집의 변동사항이 UI에 즉각적으로 반영되도록 설계하여 사용자가 빠르게 인지할 수 있게 되었다.

3. 안정성

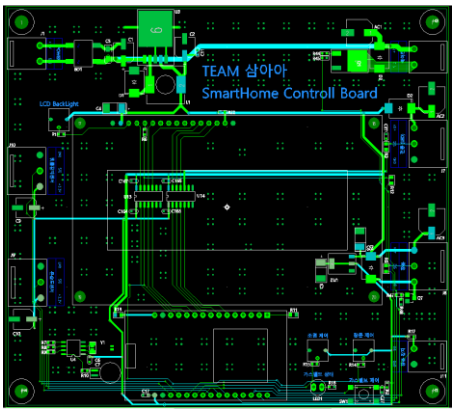
- 차량은 빠른 속도로 달리거나 터널에 들어가는 경우 네트워크가 불안정해지는 경우가 발생할 수 있다. 그래서 웹 소켓을 통해 스마트 홈과 직접 연결하는 방식보다 낮은 대역폭, 낮은 성능의 환경에서도 잘 사용할 수 있는 MQTT를 사용하였다. 차량은 송신할 메시가 발생할 때 이를 MQTT Broker로 송신하고, 스마트 홈은 비동기로 MQTT Broker에 접근하고 메시지를 수신하여 제어 명령을 수행하도록 설계하였다.

□ 개발 환경 설명

메모 포함[W3]: ※ 최대한 자세하게 기술 / 15page 이내로 작성

○ Hardware 구성

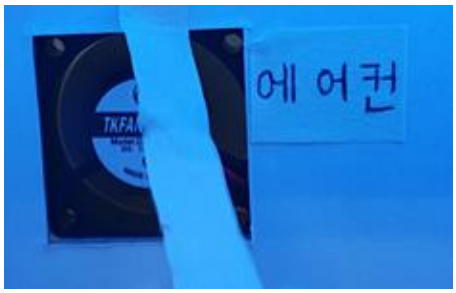
1. Smarthome Control Board



[그림 3] Smarthome Control Board PCB 파일

- 스마트 홈은 컨트롤 보드를 설계하여 제어하도록 구성하였다. MCU는 와이파이 모듈이 내장되어 있는 ESP32를 선정하였고 가전은 스마트 홈에서 보편적으로 많이 제어되는 에어컨, 무드등, 가스밸브, 창문으로 구성하였다. 센서는 온·습도 센서로 집 내부 온습도를 측정하고, 빗물감지센서를 통해 갑작스러운 소나기 등의 날씨 변화를 감지하여 사용자에게 알려줄 수 있도록 구성하였다.

- Power는 각 기기들의 다양한 입력 전압을 위해 12V -> 5V -> 3.3V로 구성하였다. 12V 입력은 보편적으로 많이 사용하는 12V 어댑터를 통해 입력을 받았고, 극성에 관계없이 사용하기 위해 브릿지 다이오드를 실장하였다. 입력 받은 12V는 5V로 강압하여야 하는데 전등, 서보 모터 등 소모전류가 많아서 큰 부하전류를 견딜 수 있고 비교적 발열이 적게 발생하는 스위칭 레귤레이터(LM2576)로 구성하여 강압하였다. 또, ESP32와 센서들의 전원 공급을 위해 5V를 3.3V로 강압을 하여야 하는데, 정확한 센싱을 위해 리플이 적은 전원을 공급할 수 있고 적은 부하 전류를 사용하기 때문에 LDO (LM1117-3.3V)를 통하여 강압하였다.

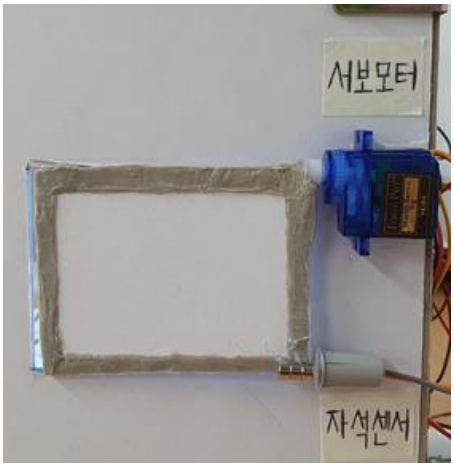


- 에어컨은 12V 쿨링 팬을 사용하여 구성하였다. MOSFET을 통해 스위칭 회로를 구성하였고, 기능 중 바람 세기 조절이 있어서 PWM을 통해 해당 기능을 구현하였다.

[그림 4] 쿨링 팬

[그림 5] WS2813

- 무드등은 WS2813로 구성된 LED Strip Bar를 사용하였다. WS2813은 LED 내에 내장된 드라이버칩에 8비트의 PWM을 주어 R,G,B 색 제어가 가능하며 다음 LED에 대한 제어 값을 주어 색을 줄 수 있어 각 LED에 대한 제어를 할 수 있다.



[그림 6] 서보 모터, 자석 센서

- 창문은 SG90 이라는 서보모터를 사용하여 구성하였다. 0~180도까지 제어가 가능하며, PWM 제어를 통해 각도를 제어할 수 있다. 모터 자체는 창문이 열렸는지, 닫혔는지 확인할 수 있는 방법이 없다. 그래서 사람이 손으로 창문을 열게 되면 문제가 발생할 수 있어서 창문을 감지할 수 있도록 자석센서를 추가하였다. 자석이 감지되면 창문이 닫힌 것으로 Firebase에 업로드하고 아니면 열린 상태로 Firebase에 업로드 할 수 있도록 하였다.

[그림 7] LCD

- LCD를 통해 현재 가전들의 상태, 집의 온/습도를 체크할 수 있도록 구성하였다.

[그림 8] 빗물감지 센서

- 빗물감지는 NS-RSM 센서의 ADC 값을 읽어 감지하도록 하였다. 타이머 인터럽트를 통해 3초마다 빗물을 감지하도록 하였고, 빗물이 감지되었거나 비가 그치면 새로운 값이 Firebase에 업로드 되도록 하였다.

- 가스밸브는 Red LED를 통하여 가스밸브 열림 / 닫힘 상태를 표시할 수 있도록 하였다.

- 집 내부에서도 가전들의 제어가 가능해야 하므로, TACT 스위치를 통해 제어하도록 구성하였다. 1초 이상 누르면 각 가전에 해당하는 동작이 Toggle 되도록 구성하였다.

- 집의 온・습도 체크는 DHT-11을 사용하여 센싱 값을 입력 받았다. 타이머 인터럽트를 통해 60초 마다 온/습도를 읽고 해당 값을 Firebase에 업로드하도록 하였다.

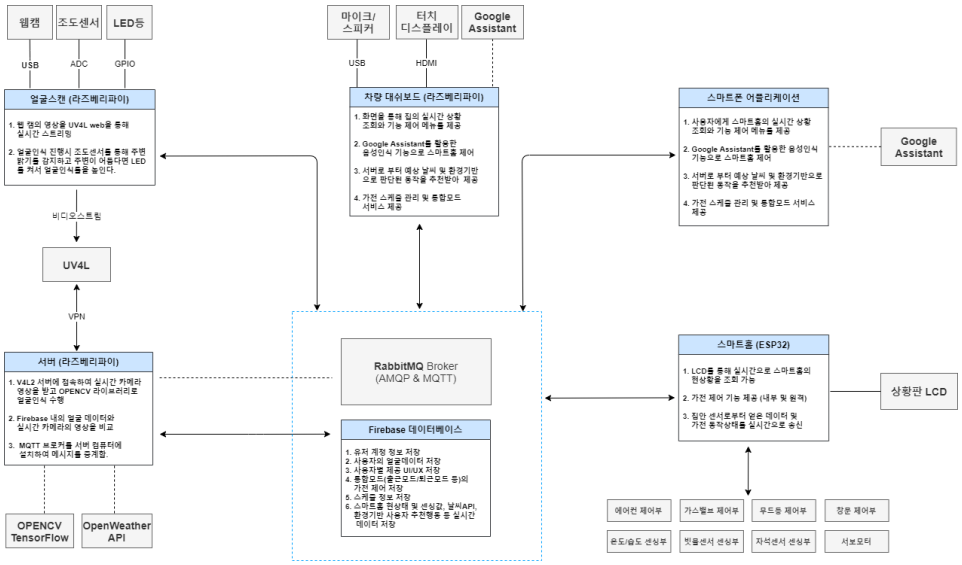
2. 얼굴 촬영 라즈베리파이

- 얼굴 인식 로그인을 위해 영상을 촬영하여 스트리밍을 담당하는 부분이다. 라즈베리파이3B+를 사용하였고, 카메라는 로지텍의 C310을 사용하였다. 라즈베리파이가 조도센서를 통해 얼굴 촬영 시 밝기가 어두우면 조명을 켜서 얼굴 인식률을 향상시킬 수 있도록 구성하였다.

[그림 9] 얼굴 촬영 부 카메라, 조명, 조도 센서

○ **Software 구성**

1. 전체 소프트웨어 구성도



[그림 10] 전체 소프트웨어 구성도

- 전체 소프트웨어는 얼굴스캔 부분, 차량 대시 보드, 스마트폰 어플리케이션, 서버, 데이터베이스, 스마트 홈으로 크게 6가지의 파트로 구성 되어있다.
- 얼굴 스캔 부분에는 얼굴인식용 카메라가 부착되어 있어 사용자의 얼굴을 실시간으로 촬영하고 UV4L을 통하여 스트리밍한다. 또한 얼굴 인식 시 주변 밝기가 어두운 경우 이를 감지하여 LED 조명을 제어해 얼굴 인식률을 높인다.
- 차량에서 사용자에게 입력을 받고 각종 정보를 출력해주는 대시 보드는 webOS를 사용 화면을 통해 집의 실시간 상황을 조회하고 가전을 제어할 수 있는 메뉴를 제공한다. 실시간

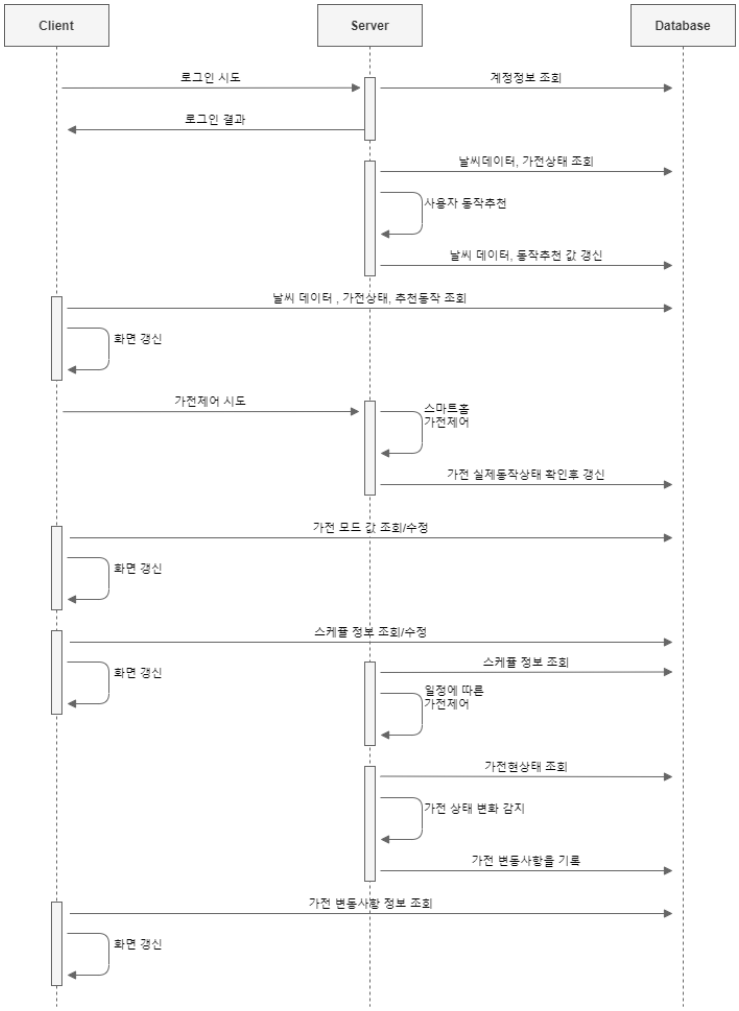
현황은 실시간 데이터베이스에 업로드 된 스마트 홈 상태와 날씨 API의 관제값과 현재 환경에 맞는 행동 추천을 사용자에게 제공한다. 가전제어는 개별가전제어, 모드제어(일괄 프리셋) 그리고 스케줄제어로 다양하게 제어가 가능해 사용자 편의성을 늘렸다. 또한 Google Assistant를 활용하여 음성인식을 통해서도 가전제어가 가능하다.

- 차량에서뿐만 아니라 스마트 홈 내부와 일반 외출시에도 가전제어 기능을 사용할 수 있도록 스마트폰 어플리케이션을 구성하였다. 전반적인 기능은 차량 대시 보드와 동일하며, 유사한 UI로 구성하여 사용자가 사용에 편의성을 더 하였다.
- 스마트 홈은 여러 가전기기와 센서로 구성하였으며, 이를 실시간으로 확인할 수 있는 LCD를 부착하였다. 각 가전은 차량 대시 보드와 스마트폰 어플리케이션으로 원격으로 제어가 가능하며, 스마트 홈 내부에서도 직접 제어가 가능하다. 스마트 홈은 센서로부터 얻은 데이터와 가전 동작 상태를 실시간으로 데이터베이스에 업데이트한다.

- 서버는 얼굴스캔 부분에서 제공하는 스트리밍 영상을 받고, 이를 openCV를 이용하여 얼굴인식을 수행한다. 또한 해당 서버에 AMQP(MQTT) Broker를 구성하였으며, rabbitMQ를 채택하였다. 스마트 홈 센서를 통해 받아들인 값과 현재 가전상태, 날씨API(OpenWeatherMap)를 통해 받아온 값을 정제, 가공, 업데이트하는 역할을 수행한다.
- 데이터베이스는 파이어베이스를 사용하였으며 인증 기능과 실시간 데이터베이스 기능을 활용하여 개발/사용 과정에서의 자원 효율을 높였다.

○ **Software 설계도**

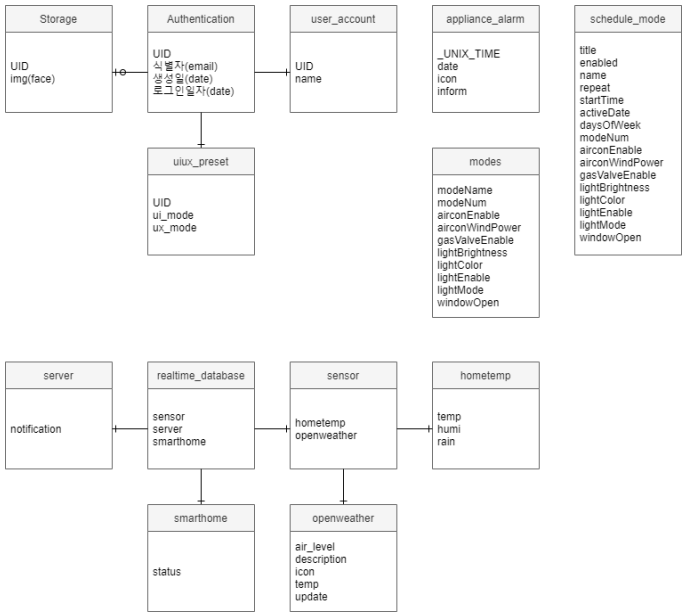
1. 소프트웨어 흐름도 작성



[그림 11 소프트웨어 흐름도]

- 사용자가 클라이언트 기기(차량 터치패널 / 스마트폰 어플리케이션)를 통해 로그인 (계정 또는 얼굴 인식)을 시도하면서 시퀀스가 시작된다. 입력된 정보를 통해 사용자를 판단하고 인증정보를 반환한다. 서버는 주기적으로 날씨 및 가전 변동사항을 확인하여 데이터베이스에 업데이트하고, 클라이언트는 이를 DB로부터 불러와서 출력하며, 사용자로부터 수정사항을 받아 이를 서버에 전달한다. 추가로 서버는 스케줄에 따라서 가전을 자동 제어한다.

2. 데이터베이스 구조



[그림 11 데이터베이스 구조]

- 데이터베이스는 Firebase를 채택하였고 nosql 방식이다. 스마트 홈 사용자를 기준으로 하여, 연결된 모든 클라이언트 기기들이 위 그림과 같은 데이터베이스 세트를 사용하게 된다. 각 데이터베이스 컬렉션에 관련된 설명은 아래와 같다.

1. authentication : 각 사용자의 계정정보(Email, Password, UID)를 암호화하여 관리
2. user_account : 각 사용자의 부가정보(이름)을 저장
3. uiux_preset : 각 사용자별로 사용할 UI/UX 정보를 저장
4. modes : 각 모드에 따른 가전정보 프리셋 값을 저장
5. schedule_mode : 사용자가 추가한 스케줄 정보를 저장
6. storage : 얼굴 인식을 위하여 각 사용자의 얼굴 사진을 저장
7. realtime database : 실시간으로 변동되는 값을 저장 (센싱값, 가전상태 등)

○ **Software 기능**

1. 사용자 인식 (얼굴 인식 및 이메일 사용자 인증)

스마트 홈 서비스를 이용하기 위해 사용자 인증 로그인을 해야 한다. 이를 위해 사용자는 얼굴 인식 혹은 이메일을 사용한다. 얼굴 인식 버튼을 누르면 얼굴 인식 중입니다. 토스트가 실행되며 얼굴 인식까지 약 1분 ~ 2분 정도 소요된다. 카메라의 얼굴 인식 결과를 반환 받으면, 사용자의 이름을 가지고 홈 페이지로 넘어간다. 이메일 로그인은 사용자가 회원가입을 할 수 있다. 생성된 계정을 사용자가 직접 이메일, 비밀번호를 기입하여 로그인하고 로그인 성공 시 계정 정보를 webOS DB에 저장하여 인터넷이 끊긴 상황에는 오프라인 로그인을 할 수 있도록 한다.

2. 다크 모드

UI모드로 다크 모드를 사용할 수 있다. 사용 시 배경색이 흰색에서 검정색으로 변경되어 야간에 사용 시에 무리를 줄여줄 수 있다. 설정 메뉴에서 사용자 별로 다크 모드 사용 여부를 지정할 수 있다. 이를 통해 로그인 시 사용자의 모드 사용 여부를 식별하여 자동으로 다크 모드를 ON/OFF 한다.

3. 날씨 확인 + 가전 상태 : 이슈사항

스마트 홈의 센서로 집안의 온습도 및 Open Weather Map API를 활용하여 날씨를 확인한다. 홈 페이지에서 현재 기온, 미세먼지 상태, 날씨를 확인할 수 있다. 가전 상태가 알림이 필요한 상황이거나 날씨와 가전 상태와 같이 판단하여 TTS + Toast로 추천하는 동작을 안내한다.

예) 가스밸브 열림 : "가스 밸브가 열려있어요"

미세먼지 나쁨 + 창문 열림 : "대기질이 좋지 않아 창문을 닫아주세요"

4. 가전, 모드 제어

웹 앱과 안드로이드 어플을 이용하여 실외에서도 가전 상태를 확인하고 제어할 수 있다. 에어컨, 무드등, 가스 밸브, 창문을 제어할 수 있다. 또한 실내, 실외, 슬립, 에코 모드를 제공하여 개별적인 가전 제어가 아닌 한번의 동작으로 여러 가전을 제어할 수 있다. 이를 통해 미처 집 안 가전을 끄지 못하고 나온 상황, 출장을 간 상황 등에서 집 안의 가전을 손쉽게 제어할 수 있다.

5. 스케줄 제어

스케줄 시스템을 이용하여 사용자가 원하는 가전의 상태를 단발성 혹은 반복성 있게 제어할 수 있다. 단발성 스케줄의 경우 사용자가 원하는 일시에 집 안 가전 상태를 설정하면 해당 시각에 동작을 수행한다. 반복성 스케줄의 경우, 반복될 요일, 시간을 선택하여 가전의 상태를 선택하면 주기적으로 해당 스케줄을 수행한다. 이를 통해 출근시간, 퇴근시간의 스케줄을 등록해두면 사용자가 동작 시키지 않아도 알아서 가전을 켜고 끄는 제어를 할 수 있다.

6. 가전 동작 정보

가전 동작 정보를 확인할 수 있다. 이를 통해 가전이 어느 일시에 동작했는지 내역을 파악할 수 있으며, 어느 모드가 사용되었는지 확인할 수 있다.

7. 안드로이드 어플

동일한 동작을 하는 안드로이드 어플로 차를 벗어난 공간에서도 스마트 홈을 확인, 제어할 수 있다. 통일성 있는 UI와 기능으로 웹 앱의 사용자는 어려움 없이 앱을 사용하여 제어할 수 있다.

8. 음성비서

WebOS에는 TTS와 구글 어시스턴트 기능을 제공하여 사용자에게 음성으로 정보 제공을 할 수 있도록 하였다. 구글 어시스턴트와 문답은 DialogFlow를 통해 기초적인 가전과 모드제어 명령을 인식하여 답변할 수 있도록 하였다.

○ UI 사용법 (구성 및 설명)

1. 로그인 페이지

[그림 13 로그인 페이지]

- 얼굴 인식 로그인

1. [webOS] 얼굴 인식을 사용하여 로그인을 수행한다.

> 얼굴 인식 버튼, JS 서비스를 이용하여 서버에 MQTT로 얼굴 인식 시작 명령 전송

> 시작 명령을 받은 서버는 대시보드 카메라 측으로 명령을 전달

> 카메라 파이는 얼굴 인식을 하고 성공 시, 이름과 UID / 실패 시, 실패 사유 반환

> 서버는 webOS 측 웹 앱에 해당 결과를 전달한다.

> webOS 웹 앱은 받은 결과로 로그인, 혹은 실패를 띄운다.

- 이메일 로그인

1. 이메일 로그인을 사용하여 로그인을 수행한다.

> 이메일을 기입하고 로그인 버튼을 누르면 JS 서비스를 이용하여 Firebase의 Authentication에 저장된 계정 정보와 비교하여 같은 계정 정보로 UID 값을 받는다.

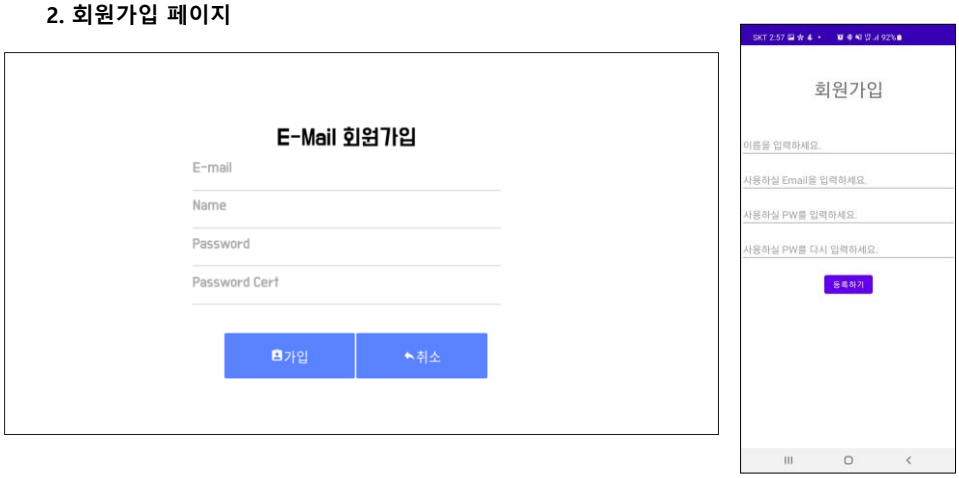
> JS 서비스를 이용하여 MQTT를 이용하여 UID값을 서버로 전송하면 서버는 UID 값에 맞는 계정 주인의 이름을 반환한다.

> 이름을 반환 받으면 로그인 성공으로 다음 페이지로 넘어간다.

2. [webOS] 인터넷이 끊긴 상황에서는 오프라인 로그인을 사용한다.

> 이메일 로그인을 한 번 수행하면 DB8(webOS DB Service)에 이메일, 비밀번호, 이름, UID 값이 저장된다.
> 이메일을 작성하여 오프라인 로그인을 누르면 DB8에 저장된 계정 정보로 로그인을 수행한다.

2. 회원가입 페이지



[그림 14 회원가입 페이지]

1. 이메일 회원가입을 진행한다.
> 이메일, 이름, 비밀번호, 비밀번호 확인을 기입하고 가입 버튼을 누른다.
> 빈 칸이 있는가, 비밀번호는 같은가 등 예외 상황을 확인하고 회원가입을 진행한다.
> JS 서비스를 이용하여 Firebase의 Authentication에 이메일, 비밀번호로 회원가입을 하고 UID 값을 반환 받는다.
> MQTT로 이름과 UID를 서버로 보내 사용자 등록을 한다.

3. 홈 페이지

[그림 15 홈 페이지]

① 사용자 이름

> 홈 페이지 진입 시 TTS 서비스를 통해 "000님 안녕하세요." 를 말한다. (webOS)

> 홈 페이지 진입 시 "000님 안녕하세요."을 표시한다. (공통)

② 스마트 홈의 온/습도 값

> Firebase Realtime DB에 저장된 온/습도 값을 가져와 Progress Bar와 함께 보여준다.

③ Open Weather Map API에서 제공하는 날씨 정보 (서울)

> 서버가 Firebase Realtime DB에 업로드하는 실시간 날씨 정보를 읽어와 UI에 표시

④ 모드 편집 페이지 이동 / ⑤ 스케줄 편집 페이지 이동

⑥ 가전 상세 제어 페이지 이동 / ⑦ 모드 토글 버튼

> 모드에 저장된 개별가전의 ON/OFF 동작을 실행시킨다. 모드에 해당하는 개별가전의 ON/OFF 여부가 UI에 반영되며, MQTT 메시지를 스마트 홈 컨트롤 보드로 송신한다.

> 모드는 하나만 실행될 수 있다. 다른 모드 실행 시 이전 모드는 OFF 상태가 된다.

> Firebase Listener를 통해 현재 모드 실행상태가 UI(토글버튼)에 반영된다. 사용자는 이를 통해 현재 실행되고 있는 모드를 알 수 있다.

⑧ 개별가전 토글 버튼

> 개별가전의 ON/OFF 동작을 수행한다. 버튼 클릭 시 해당하는 동작의 MQTT 메시지를 스마트 홈 컨트롤 보드로 송신한다.

> Firebase Listener를 통해 현재 개별 가전의 상태가 실행상태가 UI(토글 버튼)에 반영된다. 사용자는 이를 통해 현재 실행되고 있는 개별가전들을 알 수 있다.

⑨ 실시간 가이드 문구

> 서버가 날씨, 가전 상태을 복합적으로 체크하여 Realtime DB에 업로드하는 문구이다.

> 특별한 이상이 없으면 "오늘도 좋은 하루가 되세요!"가 출력되며, 비가 오는데 창문이 열려 있는 등의 이상이 발생하면 "비가 오니 창문을 닫아주세요"의 문구를 띄운다.

[webOS] Toast와 TTS를 통해 실시간 가이드를 사용자에게 알려준다.

[Android] Text를 통해 사용자에게 실시간 가이드를 사용자에게 알려준다.

4. 모드 상세 설정 페이지

[그림 16 모드 설정 페이지]

① 모드 리스트

> Firestore에 저장된 모드를 불러오고 해당 모드의 가전 동작 상태를 UI에 표시한다.

② 에어컨 ON/OFF

③ 에어컨의 바람세기 조절 Progress Bar

④ 무드등 ON/OFF

⑤ 무드등 밝기 Progress Bar

⑥ 무드등 색상

⑦ 무드등 모드

⑧ 가스밸브 ON/OFF

⑨ 창문 OPEN/CLOSE

⑩ 모드의 가전 제어 값 저장

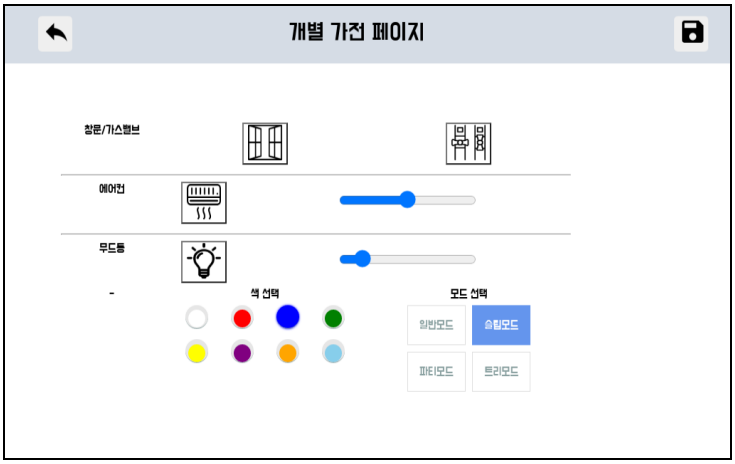
> 현재 UI에 지정된 상태를 가져와 Firestore의 해당 모드에 값들을 저장한다.

5. 스케줄 페이지

[그림 17 스케줄 설정 페이지]

① 스케줄 리스트 : Firestore에 등록된 스케줄를 불러와 표시한다.
② 스케줄 추가 : Firestroe에 새로운 스케줄을 등록한다.
③ 스케줄 사용 활성화
④ 스케줄 가전 제어를 모드로 할 것 인지 체크
⑤ 반복 여부 확인 : 반복 시 요일을 선택하여 반복, 해제 시 특정 일시에 스케줄 실행
⑥ 시간 설정 : 원하는 시각에 스케줄 실행
⑦ 가전 제어

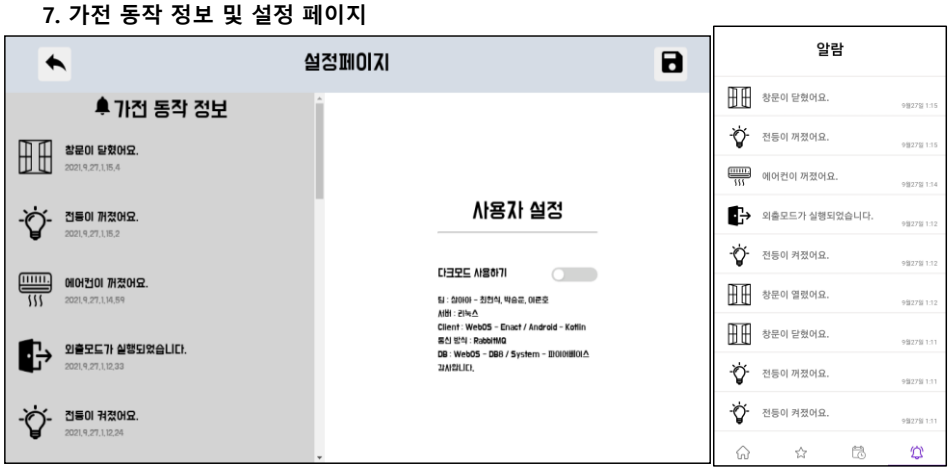
6. 가전 상세 제어 페이지



[그림 18] 가전 상세 제어 페이지

- 가전 상세 제어 : MQTT를 이용하여 스마트 홈으로 가전 제어 메시지를 보낸다.

7. 가전 동작 정보 및 설정 페이지



[그림 19] 가전 동작 정보 페이지 + [webOS] UI 모드 변경

- 가전 동작 정보 : Firebase의 Fire store에 저장된 모드, 가전 동작 이력을 확인한다.
- [webOS] UI 모드 변경 : 다크 모드를 선택하여 CSS를 수정하고, 다크 모드를 실행한다.

○ 개발환경 (언어, Tool, 사용시스템 등)

1. 차량 대쉬보드

타겟 OS	webOS OSE-2.13.g	**프레임워크**	Enact, node.js
개발언어	Javascript, html, css	**라이브러리**	React-dom, react-dom-route

2. 스마트 홈

타겟 OS	Arduino (ESP **)	**프레임워크**	Arduino
개발언어	C	**라이브러리**	ESP

3. 스마트폰 어플리케이션

타겟 OS	Android	**프레임워크**	
개발언어	Kotlin, java	**라이브러리**	

4. 서버

타겟 OS	Raspbian	**프레임워크**	
개발언어	Python, bash script	**라이브러리**	Firebase_admin, re, pika, tensorflow, scipy, opencv-python, requests

5. 차량 카메라 기기

타겟 OS	Raspbian	**프레임워크**	
개발언어	Python, bash script	**라이브러리**	RPi.GPIO, pika

□ 개발 프로그램 설명

메모 포함[W4]: ※ 최대한 자세하게 기술 / 8page 이내로 작성

○ 파일 구성

1. Web App

- com.ta.car2smarthome

대시보드를 구성하는 웹 앱으로 Enact을 사용하여 카 대시보드를 구현하였다.
사용자가 사용할 로그인, 회원가입, 홈, 가전, 모드, 스케줄, 가전 동작 정보+UI 페이지로 구성되어 있으며, WebOSServiceBridge API 를 이용하여 JS 서비스와 통신하여 기능을 구현했다.

```
Car2Smarthome
 ㄴ node_modules // 설치한 모듈
 ㄴ resources // 웹 앱에서 사용하는 리소스 파일
  ㄴ css // 공통적으로 적용되는 css 파일이 담김 폴더
   ㄴ smarthome_icon // 스마트 홈 제어 부분에 들어가는 아이콘 폴더
   ㄴ weather_icon // 날씨 아이콘 폴더
 ㄴ src
  ㄴ App
   ㄴ App.js // App 컴포넌트
   ㄴ package.json // main 명시
  ㄴ views // 페이지
   ㄴ SignIn // 로그인 페이지
   ㄴ SignUp // 회원가입 페이지
   ㄴ Home // 홈 페이지
   ㄴ Appliance // 가전 상세 제어 페이지
   ㄴ Mode // 모드 설정 페이지
   ㄴ Schedule // 스케줄 설정 페이지
   ㄴ Alarm // 알람 목록 + UI 모드 제어 페이지
  ㄴ firebase.js // firebase reatime DB, store 정보 저장
  ㄴ index.js
 ㄴ appinfo.json // 웹 앱 정보 및 권한 설정
 ㄴ icon.png
 ㄴ package.json // 모듈 정보
 ㄴ package-lock.json // 모듈 정보
```

- JS Service

2.1 com.ta.car2smarthome.service

대시보드의 기능을 담당하는 JS Service이다. 사용자 회원가입, 로그인, 스마트 홈 상태 확인, 가전 제어 명령 등 기능을 담당하며 스마트 홈, 서버와 MQTT 송·수신 서비스와 데이터베이스인 Firebase의 Authentication, Cloud Firestore, Realtime Database의 데이터를 생성, 읽기, 갱신, 삭제한다.

```
ㄴ service
  ㄴ node_modules // 설치한 모듈
  ㄴ package.json // 모듈 정보
  ㄴ package-lock.json // 모듈 정보
  ㄴ service.json // JS Service 정보
  ㄴ webos_service.js // 작성한 JS Service
```

2. Smarthome Control

2.1 ESP32

Smarthome의 가전 제어를 담당하는 부분이다. ESP32가 WIFI를 통해 MQTT의 메시지를 수신하고 온/습도, 가전의 상태를 Firebase에 업로드 하도록 한다. 무드등, 쿨링 팬등의 가전제어와 온/습도, 택트버튼 감지 등의 센싱작업들 여러가지 작업들이 통신과 함께 딜레이 없이 이루어져야 하므로 타이머 인터럽트를 통해 시분할 시스템처럼 작동하도록 구성하였다. 센싱작업은 60초마다 체크하여 Firebase에 업로드하였고, 무드등처럼 지속적으로 값을 보내줘야 하는 것은 타이머에서 주기적으로 Drive하도록 하였다. 일반적으로 while문에서는 MQTT 서버 끊김과 메시지가 수신 여부를 체크하여 수신시 작업을 수행하도록 진행하였다.

```
SmartHomeControl
 ㄴ applianceControl // 무드등, 에어컨 등 가전 제어 함수
 ㄴ chk_interrupt    // 타이머 인터럽트 실행 함수
 ㄴ IRAM_ATTR onTimer // 타이머 인터럽트 서비스 루틴
 ㄴ Communication // WIFI 설정 및 Firebase, MQTT 연결 함수
```

3. Android Application

3.1 Android Application

스마트 홈 가전 제어 및 상태를 조회할 수 있는 안드로이드 어플리케이션이다. WebOS의 웹앱과 동일한 기능과 유사한 UI로 제작하였다.

```
com.psw9999.car2smarthome
 ㄴ Adapter // recycler view 를 생성하기 위한 Adapter 폴더
 ㄴ data // Firebase 에서 불러온 데이터를 저장하기 위한 dataclass 파일 폴더
 ㄴ loginActivity.kt // 로그인 Activity
 ㄴ SignupActivity.kt // 회원가입 Activity
 ㄴ MQTT_Thread.kt // 로그인 성공시 실행하는 MQTT Thread
 ㄴ MainFragment.kt  // 메인화면 Fragment
 ㄴ ModeFragment.kt  // 모드 설정 Fragment
 ㄴ ScheduleFragment.kt  // 스케줄 리스트 Fragment
 ㄴ ScheduleActivity.kt  // 스케줄 설정 Activity
 ㄴ AlarmFragment.kt  // 알람 리스트 Fragment
```

4. 서버

MQ 송수신, 파이어스토어 접근/관리, 각 데이터 정제 및 처리, 얼굴인식을 담당한다.

```
Backend-Server
 ㄴ firebase-python-sdk-key //파이어베이스 접속을 위한 KEY 를 보관
   ㄴ <firebase-sdk-key>.json
   ㄴ module //각 모듈라이브러리를 저장
 ㄴ programstart_manager.py //각 프로그램 총괄관리
 ㄴ backend_process.py //MQTT 명령 처리 프로그램
 ㄴ realtimedb_connect.py //파이어베이스 RealTimeDB 를 접속관리하는 프로그램
 ㄴ realtimedb.json //RealTimeDB 에 현황을 파일로 실시간 업데이트
 ㄴ server_notification //RealTimeDB 에 업로드할 사용자 추천 행동 값

Face-Recognition
 ㄴ 20180408-102900
 ㄴ npy
 ㄴ facenet.py
 ㄴ detect_face.py
 ㄴ face.result //얼굴 인식 결과를 공유하는 파이프라인
 ㄴ realtime_facenet_git.py //얼굴 인식을 진행하는 프로그램
```

4.1. 얼굴 학습 및 인식 OpenCV, TensorFlow, FaceNet

> 얼굴 인식을 위하여 사용자의 얼굴을 학습해야 한다.

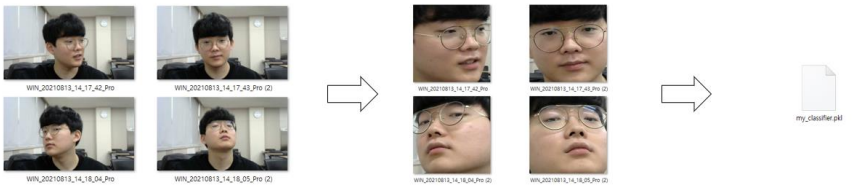


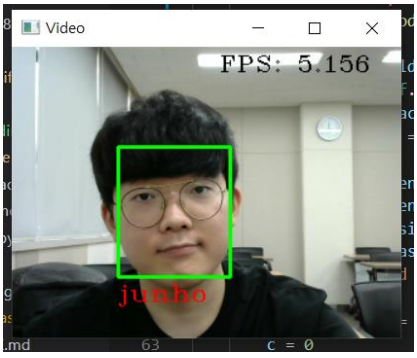
사진 촬영 사진 조정 사진 학습

[그림 20] 얼굴 학습 과정

> 이미지 처리를 위하여 OpenCV 라이브러리를 활용했다.
학습할 사용자의 사진을 촬영한 후, Firebase의 Storage에 저장한다.
서버는 MTCNN 기반의 얼굴 탐지하여, 얼굴을 182x182 크기로 정렬한다.

> 얼굴 학습
얼굴을 인식하는 얼굴 인식 모듈을 위하여 2015년 구글이 발표한 FaceNet 논문을 활용한 얼굴 인식을 구현하였다. 얼굴을 학습시킬 때는 160x160로 리사이즈하고, 미리 학습된 FaceNet 모델과 통합하여 새로운 Crown 모델을 생성한다.

> 얼굴 인식
FaceNet 특징 추출, 얼굴 매칭으로 얼굴 인식을 진행한다.
이는 각 하는 모델로 얼굴 사진에서 사람의 대한 특징 값을 구해주는 모델로 이 값을 활용하여 사용자를 구분할 수 있다. 특징 추출의 단계에서는 얼굴 이미지를 128차원으로 임베딩하여 유클리드 공간에서 이미지간 거리를 통해 분류하는 triplet loss 기법을 사용한다. 현재 이미지와 모델과의 얼굴 벡터와 거리를 계산하는데 닮을수록 0에 가까워지고 닮지 않을수록 1에 가까운 값이 나온다.
> 실시간 인식
웹 캠으로 촬영하는 영상을 UV4L 스트리밍 서버로 송신하여 처리 서버에서 실시간 인식을 시작한다. 얼굴 사진으로 학습한 crown 모델과 미리 학습된 모델을 사용하여 사용자를 인식한다.해당 프로젝트에서는 사용자 배열을 만들고 촬영 프레임마다 인식한 사용자의 count 값을 증가하는 방식으로 count가 15가 넘어가는(약 1분) 사람을 인식하는 방법을 사용하였다.



[그림 21] 실시간 얼굴 인식

- 5. 차량카메라

얼굴인식 카메라(웹캠)를 사용하여 UV4L로 스트리밍하는 기기이다. AMQP를 통해 얼굴인식 시작 메시지를 받게되면 관련 동작이 시작되며 이때 조도센서를 통해 주변이 어두운 경우 연결되어있는 조명(LED)을 켜 얼굴인식률을 높인다.

```
CameraClient
 ㄴ camera_client.py //Message Queue 통신을 수행하고, GPIO 포트를 제어하는 프로그램
 ㄴ module
   ㄴ rabbitmq_client.py //rabbitMQ 를 사용하기위한 모듈
 ㄴ uv4l
   ㄴ run_uv4l.sh //uv4l 스트리밍을 가동시키는 스크립트
   ㄴ uv4l-uvc.conf //uv4l 설정파일에 사본
 ㄴ vpn
   ㄴ auth.txt //vpn 접속용 계정정보를 저장해둔 파일
   ㄴ camera.ovpn //vpn 서버 관련 정보를 저장해둔 oepnvpn 설정 파일
   ㄴ openvpn_start.sh //vpn 서버와 연결을 진행하는 스크립트

.config
 ㄴ autostart
   ㄴ RUN_OPENVPN.desktop //부팅시 VPN 스크립트를 실행시키는 파일
   ㄴ CAMERA_MQTT_CLIENT.desktop //부팅시 camera_client.py 를 실행시키는 파일
```

○ 함수별 기능

1. 웹 앱 - com.ta.car2smarthome (페이지 별 중복 함수는 작성 x)

1.1 SignIn 페이지

- onSignIn() : 이메일 로그인
- onFaceSignIn() : 얼굴 인식 로그인
- onDBSignIn() : Native Service인 DB8을 이용한 로그인
- tts(ment) : Native Service인 tts를 활용, ment 음성으로 얘기
- createToast(ment) : Native Service인 notification을 활용, ment 토스트로 알림
- putKind() : DB8 시작
- putPermissions() : DB8 권한 설정
- putUserData(email, password, name, db, uid) : DB8 계정정보 추가
- findUserData(email, password) : DB8에서 이메일 정보에 맞는 계정 찾기

1.2 Home 페이지

- startAssistant() : Native Service인 ai voice 시작
- getState() : ai voice 설정
- stopAssistant() : ai voice 종료

- setUI : UI 표시
- onDoMode(num) : num에 맞는 모드 실행 (실내, 외출, 슬립, 에코)
- onDoApplience(num) : num에 맞는 가전 실행 (에어컨, 무드등, 가스밸브, 창문)
- sendMqtt(exchange, routingKey, msg) : MQTT 메시지 송신
- getRTDB() : Firebase Realtime Database 수신
- getStoreDB() : Firebase Firestore Database 수신
- getUIDB() : Firebase Firestore Database 계정별 UI 프리셋 수신

1.3 Mode 페이지
- onSelectApplience() : 선택한 모드의 가전을 변경

1.4 Schedule 페이지
- setScheduleUI() : getStoreDB로 수신한 스케줄 정보를 그림
- onAddDoc() : 새로운 스케줄을 추가
- onDeleteDoc() : 스케줄 삭제
- onSelectSchedule(num) : num에 맞는 스케줄 선택

1.5 Appliance 페이지
- setUI() : getRTDB로 수신 받은 가전 상태를 표시

1.6 Alarm 페이지 (가전 동작 정보 + UI모드)
- displayIcon() : 가전 동작 정보의 아이콘을 표시
- setAlarmUI() : 반환 받은 동작 정보를 표시
- setDarkMode(boolean) : 다크모드 실행

2. JS Service - com.ta.car2smarthome.service

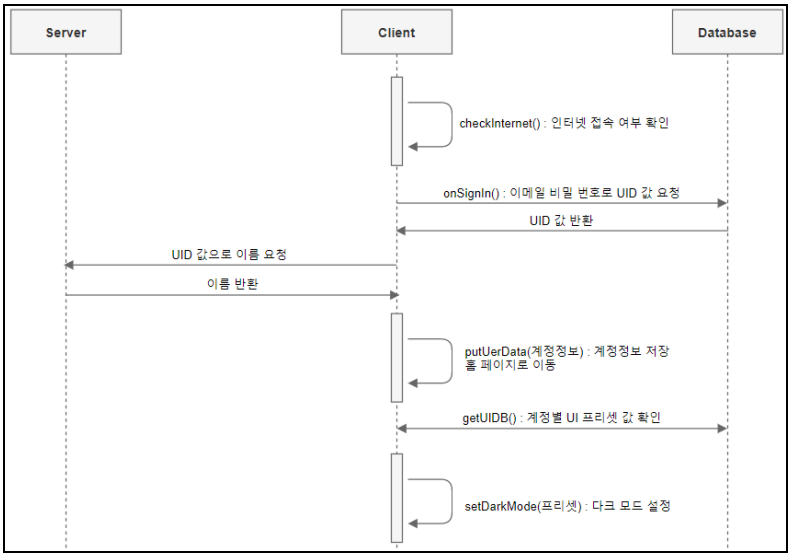
2.1 MQTT 서비스
- sendMqttFunc(exchange, routingKey, msg) : Rabbit MQ MQTT 송신 함수
- getMqtt(queue) : Rabbit MQ MQTT 수신 함수

2.2 Firebase 서비스
- signIn 서비스 : 이메일 비밀번호 받아서 로그인
- firebaseLogin(email, password) : Firebase authentication 로그인하여 UID값 반환
- signUp 서비스 : 이메일, 비밀번호 받아서 회원가입
- firebaseSignup(email, password) : authentication 회원가입하여 UID값 반환
- firebaseRTdb() : Firebase Realtime DB 불러오기
- getDB() 서비스 : Realtime DB 불러와서 반환
- facerSignIn 서비스 : 얼굴 인식 시작 신호 메세지 서버로 송신하고 성공 시 반환 받은 이름으로 로그인
- facerGetMqtt(queue) : MQTT queue(webos.car)에 올라온 이름 값 반환

○ **주요 함수의 흐름도**

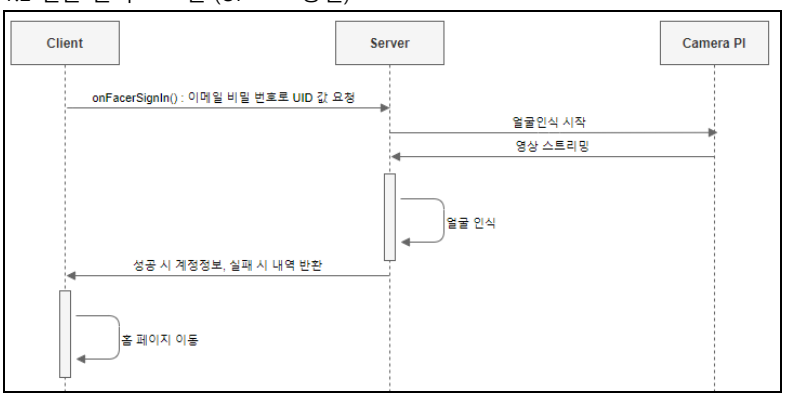
1. 로그인

- 이메일 로그인



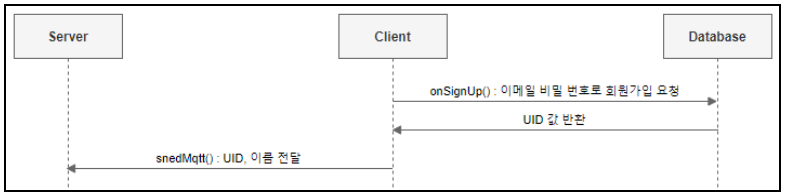
[그림 22] 이메일 로그인 흐름도

1.2 얼굴 인식 로그인 (UI 모드 동일)



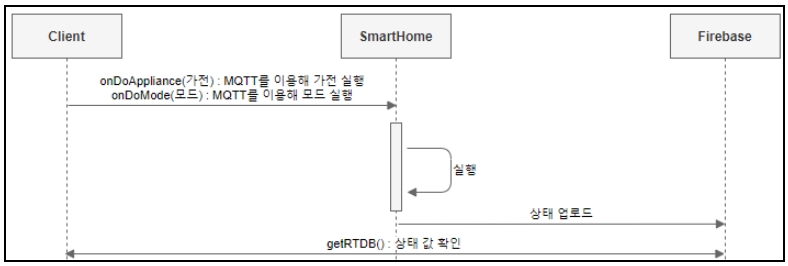
[그림 23] 얼굴 인식 로그인 흐름도

2. 이메일 회원 가입



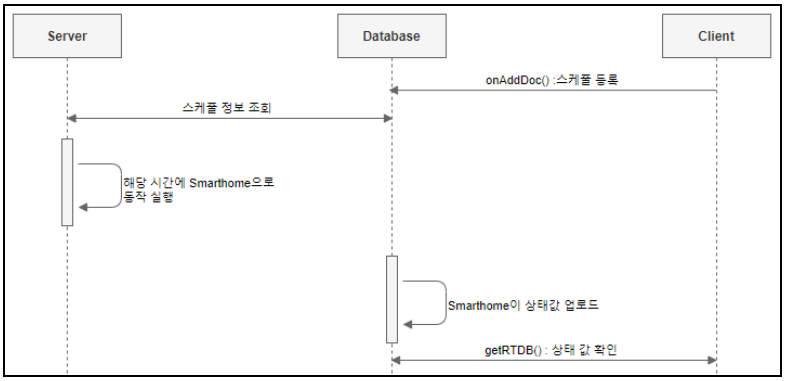
[그림 24] 이메일 회원가입 흐름도

3. 가전, 모드 제어



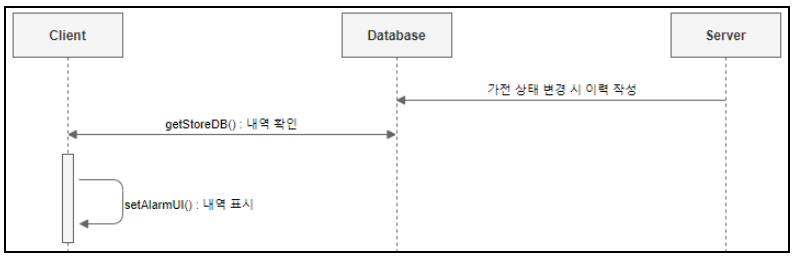
[그림 25] 가전, 모드 제어 흐름도

4. 스케줄 제어



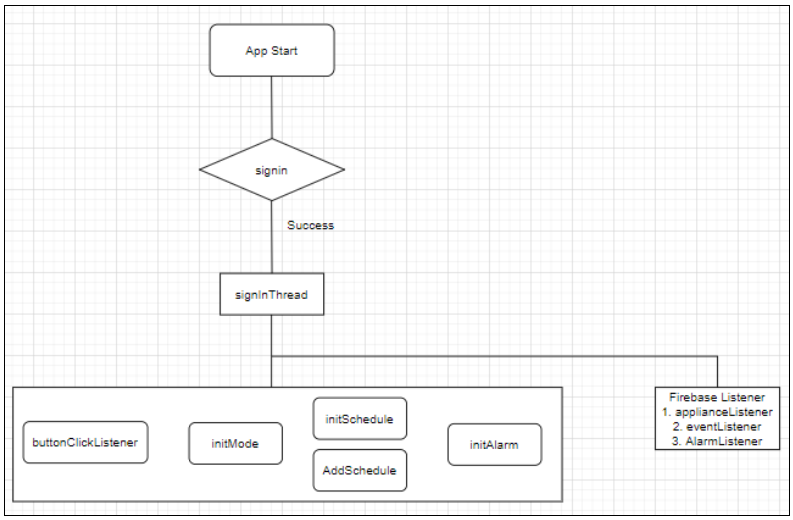
[그림 26] 스케줄 제어 흐름도

5. 가전 동작 정보



[그림 27] 가전 동작 정보(내역) 흐름도

6. 안드로이드 애플리케이션



[그림 28] 안드로이드 애플리케이션 흐름도

○ 기술적 차별성

- Message Queue 통신 방식 사용

각 기기간의 통신을 일반적인 Socket통신이 아닌 MQ방식을 채택하였다. 이는 항시 통신과 작업처리를 염두해 두어야 하는 네트워크 임베디드 기기에 적절하다고 볼 수 있다. MQTT는 단순하고 가벼운 메세징 프로토콜이며, 구독/발행이라는 비동기 처리방식은 각 기기에 자원 소모를 최소화 하여준다. 또한 큐를 보관하는 브로커 서버가 존재하며, rabbitMQ의 메시지는 Exchange를 통해 Queue에 전달되므로, 다자간 통신이나 에러 처리 등 개발에서도 유리하다.

- Firebase 사용

데이터베이스는 구글이 제공하는 파이어베이스를 채택하였다. NOSQL DB로 개발단계에서의 발생하는 데이터 필드 규격의 변화를 유연하게 처리할 수 있었다. 파이어베이스 인증(Firebase Authentication)을 통해 이미 검증된 로그인 기능을 사용하였으며, 유저의 민감한 정보를 안정적으로 보관하였다. 또한 리스너를 통해 각 기기와 데이터베이스의 자원낭비를 방지하였다.

□ 개발 중 장애요인과 해결방안

메모 포함[W5]: ※ 개발 과정에서 나타났던 모든 장애요인(Risk)들을 나열하고, 이러한 장애요인들이 발생했던 경우 어떻게 해결했는지 구체적으로 제시 / 1page 이내로 작성

○ 차량 측 UV4L 스트리밍시 다른 네트워크에서 접근 불가능

- 얼굴인식을 진행하기위하여 차량측에 카메라를 부착하고 이를 UV4L을 통하여 web으로 스트리밍을 진행하였으나, IP로 접근이 필요하여 다른 네트워크에 있고 해당 네트워크로부터 포트포워딩을 받을 수 없는 경우 접근이 제한되어 문제가 발생.
- 서버를 통해 가상 네트워크(VPN)를 구축하고, IP를 배정하여 얼굴인식 서버와 차량 측 카메라 간에 원활한 통신이 가능하도록 함.

○ 스마트 홈(ESP32) 측 성능이슈로 AMQP 사용 불가능.

- 스마트 홈 측 ESP32의 성능 이슈로 AMQP를 통한 통신이 연결 단계에서부터 비정상적으로 진행되는 문제가 발생.
- rabbitMQ Broker에 기본 프로토콜인 AMQP와 더불어 추가로 MQTT 를 사용할 수 있도록 플러그인(rabbitmq_mqtt)을 설치하고, Broker측에서 MQTT통신에 사용할 계정정보와 접근권한 exchange 및 listener 등을 설정.
- 스마트 홈(ESP32)는 다른 기기들과 다르게 AMQP가 아닌 MQTT로 통신을 진행하며, 그로 인해 발생하는 차이는 Broker가 보완함.

○ 산발적 데이터베이스 조회로 인한 자원 낭비

- 여러 기기에서 산발적으로 데이터베이스를 조회하여 기기 및 데이터베이스 모두 필요 이상의 자원낭비가 발생함. (변경된 내용이 없으나 새로 읽어오는 경우)
- Firebase SDK에서 제공하는 Listener를 이용하여 Thread를 만들어 두면, 데이터베이스가 변경이 되었을 때 이를 감지하고, callback 함수가 실행되어 처리로직을 자동으로 실행할 수 있게 됨.
- 필요한 데이터를 polling 할 필요가 없어 기기의 자원 낭비를 방지할 수 있으며, 어플리케이션은 매시점에서 언제나 최신 데이터를 가지고 있게 됨.

○ ESP 8266 + WiFi 모듈 펌웨어 업로드 이슈

- 펌웨어 업로드 시 8266과 WiFi 모듈 사이에 오류가 발생하고 테스트 시 매번 배선을 하는 장애 및 번거로움 발생
- WiFi 일체형 모델인 ESP 32로 대체하여 장애요인을 개선하였다.

○ FaceNet 윈도우 테스트 버전 -> 리눅스 서버

- 윈도우 콘다 가상환경으로 테스트했던 얼굴 인식 FaceNet 모델을 리눅스 서버로 이식하는 과정에서 윈도우에서 사용했던 가상환경이 리눅스 환경에서는 바로 이식되지 않음
- 테스트 코드는 수정하지 않고 리눅스 환경에서 새로 환경을 꾸려 이상 없이 이식하였다

□ 개발결과물의 차별성

메모 포함[W6]: ※ 개발한 결과물의 차별성 및 우수성 설명 / 1page 이내로 작성

○ 모드 가전 제어

- 실내, 실외, 슬립, 에코 모드를 제공하여 개별적인 가전 제어가 아닌 한번의 동작으로 여러 가전을 제어할 수 있다. 이를 통해 가전들을 다양한 상황에 알맞은 동작을 하는 제어를 한 번에 할 수 있다. 또한 사용자가 모드 동작을 직접 설정할 수 있어 사용자가 자신에 맞는 상황으로 제어할 수 있다.

예) 외출 모드 : 에어컨 OFF, 무드등 OFF, 가스 밸브 OFF, 창문 OFF
에코 모드 : 에어컨 OFF, 무드등 OFF, 가스 밸브 OFF, 창문 ON

○ 예약 스케쥴 작업

- 사용자가 원하는 가전의 상태를 단발성 혹은 반복성 있게 제어할 수 있다.
  > 단발성 스케줄의 경우 사용자가 원하는 일시에 집 안 가전 상태를 설정하면 해당 시각에 동작을 수행한다.
  > 반복성 스케줄의 경우, 반복될 요일, 시간을 선택하여 가전의 상태를 선택하면 주기적으로 해당 스케줄을 수행한다.
- 출장, 여행 등 반복적이지 않은 상황에서 미리 가전의 상태를 예약하여 사용할 수 있다.
  - 출근시간, 퇴근시간의 스케줄을 등록해두면 사용자가 동작 시키지 않아도 알아서 가전을 켜고 끄는 제어를 할 수 있다.

○ 환경기반 자동 안내 (비올 때 창문닫기, 가스벨브 열리면 경고)

- 스마트 홈의 센서로 집안의 온습도 및 Open Weather Map API를 활용하여 날씨를 확인
  - 가전 상태가 알림이 필요한 상황, 날씨와 가전 상태와 같이 판단하여 자동 안내
  - TTS + Toast로 추천 동작 내역을 안내한다.

예) 가스밸브 열림 : "가스 밸브가 열려있어요"
미세먼지 나쁨 + 창문 열림 : "대기질이 좋지 않아 창문을 닫아주세요"

○ 음성 알림

- 사용자가 운전 중에는 대시보드를 직접 동작 할 수 없다. 따라서 가전 상태의 변화 및 안내사항을 TTS 서비스로 안내하여 사용자가 안전하게 정보를 전달받을 수 있도록 했다.

○ 다크모드

- 밝은 색상을 제어하여 눈의 피로를 줄여준다.
- 주변 조명이 낮은 상황에서도 잘 보인다.

□ 개발 일정

No	내용	2021年															
		6月				7月				8月				9月			
1	계획 수립																
2	자료 검토 및 기술 조사																
3	webOS OSE 분석																
4	기능 설계																
5	개발환경 구축																
6	서버(MQ, UV4L, VPN) 구축																
7	스케쥴 서비스 개발																
8	스마트 홈 제어 서비스(스케줄,알람,모드) 개발																
9	Firebase 구축																
10	스마트 홈 기기 개발																
11	얼굴 **Software 기능** 인식 모듈화																
12	스마트폰 앱 개발																
13	PCB 설계/제작																
14	TTS 모듈화																
15	음성인식 개발																
16	웹 앱(대시보드) 개발																
17	UI 개발																
18	webOSE, 스마트 홈, 스마트폰, 서버 연동																
19	하드웨어 제작/가공																
20	시험 평가 및 테스트																
21	보고서 및 시연 영상 제작																

□ 팀 업무 분장

No	구분	성명	참여인원의 업무 분장
1	팀장	최현식	• 서버 구축 / 개발 • DB(파이어베이스) 설계 • 차량 측 영상 스트리밍 및 VPN 연결 구현 • webOSE 웹앱 UI 개발 • 하드웨어 제작/가공
2	팀원	박승운	• 안드로이드 앱 개발 • 음성인식(Google Assistant)를 활용한 서비스 구현 • PCB 설계/제작 • 스마트 홈 개발 • 하드웨어 제작/가공
3	팀원	이준호	• 얼굴 인식 기능 개발 (OpenCV) • webOSE 웹앱 및 서비스 개발 • 하드웨어 제작/가공