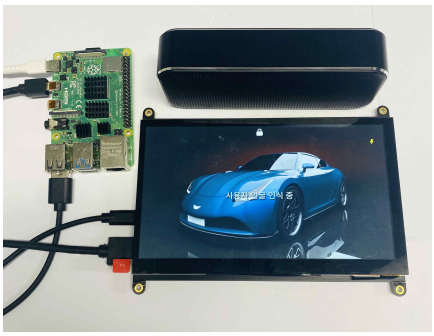
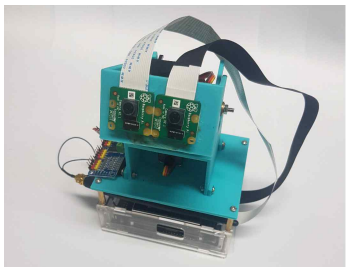
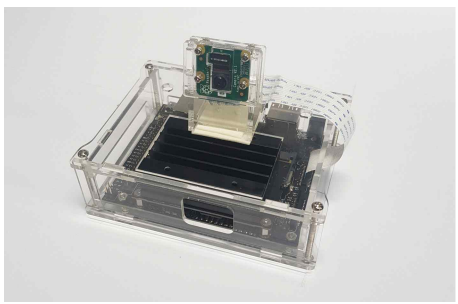
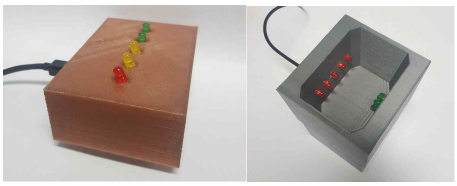
제19회 임베디드SW경진대회 개발완료보고서

[webOS 기반 Car 2 Smart Home 솔루션]

□ 개발 요약

<table>
<tr><td>팀 명</td><td>THINCAR</td></tr>
<tr><td colspan="2">
<table>
<tr><th>차량</th><th>스마트 홈</th></tr>
<tr><td> 
차량 대시보드</td><td> 
현관 CCTV 모듈</td></tr>
<tr><td> 
차량 내 얼굴 인식 모듈</td><td> 
냉장고 모듈</td></tr>
<tr><td></td><td> 
전기장판 및 욕조 모듈</td></tr>
</table>
</td></tr>
<tr><td>작품명</td><td>THINCAR의 움직이는 Smart Home</td></tr>
<tr><td>작품설명
(요약)</td><td>- 1인 가구의 불편함을 개선할 수 있는 운전자 편의기능 제공
- 대시보드에서 스마트 홈 IoT 기기를 제어할 수 있는 다양한 환경 제공
- Drowsiness detection을 이용한 주행 집중도와 졸음 운전지수 제공</td></tr>
<tr><td>소스코드</td><td>https://github.com/THINCAR/2021ESWContest_webOS_3004</td></tr>
<tr><td>시연동영상</td><td>https://youtu.be/7USkTjFd8rU</td></tr>
</table>

□ 개발 개요

○ 개발 작품 개요

[그림1] 현대자동차 카투홈(Car to Home) 서비스

현재 현대자동차에서 지원하는 카투홈(Car to Home) 서비스는 에어컨, 보일러, 가스 차단기 등 스마트 홈 IoT기기를 차 안에서 제어할 수 있다. 그러나 현재까지의 서비스는 단순히 IoT기기를 제어하는데 머물렀다면 제어하는 것뿐만 아니라 스마트홈으로부터 받은 정보를 활용한 다양한 유용하고 편리한 서비스를 제공하고자 한다.

[그림2] 최근 5년간 1인 가구 비중

이와 더불어 최근 5년간 1인 가구 비중의 증가 폭을 살펴보면 해가 갈수록 1인 가구 수는 증가하고 있으며 매년 그 증가 폭이 커지고 있다. 이러한 추이를 고려하면 앞으로도 1인 가구는 지속적으로 증가할 것으로 예상된다. 본 프로젝트는 최근 증가하는 1인 가구들의 편의성을 높일 수 있는 스마트홈 정보를 활용한 Car to Smart Home 솔루션 개발을 목표로 한다.

해당 솔루션을 개발하기 위하여 1인 가구의 불편함 중 1인 가구의 특성상 가사를 온전히 혼자 부담함으로써 생기는 불편함, 치안에 대한 불안감, 거주인 부재 시 방문객에게 생기는 의사전달 제한에 초점을 맞추었다. 따라서 냉장고, 현관, 전기장판, 욕조에 관한 총 4가지의 IoT모듈을 제작하게 되었다.

○ 개발 목표

1인 가구의 불편함 개선을 초점으로 둔 운전자 편의기능을 개발하기로 하였다.

1) 가사를 온전히 혼자 부담함으로써 생기는 불편함에 초점을 둔 기능

- 냉장고 모듈

차량 내부에서 스마트홈의 냉장고 모듈 안에 있는 식재료의 실시간 모습과 Food Detection 된 식재료명, 개수를 알 수 있다. 그리고 냉장고 모듈 내부의 식재료를 활용한 메뉴를 차량에서 추천받을 뿐만 아니라 해당 메뉴에 추가적으로 필요한 식재료 정보를 제공받는다. 추가적으로 메뉴 관련 레시피를 차량에서 냉장고 모듈로 정보를 전송할 수 있다. 따라서 전송된 정보를 냉장고 모듈에서도 볼 수 있다. 이러한 기능을 통해 집으로 가는 퇴근길에 냉장고 내부 식재료 정보를 알 수 있어 집을 들리는 번거로움 없이 장을 보고 집으로 돌아갈 수 있다.

- 전기장판, 욕조 모듈

차량 내부에서 스마트홈에 있는 전기장판, 욕조 모듈의 전원을 제어할 수 있다. 그리고 전기장판 모듈은 온도 조절이 가능하고, 욕조 모듈은 온도 조절뿐만 아니라 수위 조절까지 가능하다. 추가적으로 Gaze Detection을 활용하여 운전자의 주행집중도를 파악하여 Text to Speech기술(이하 TTS)를 통해 운전자에게 모듈 사용을 권해주는 추천 서비스를 제공한다. 또한 음성인식을 활용하여 운전자는 음성으로 추천 서비스를 수락 또는 거절할 수 있다. 그리하여 집에 도착하기 전에 미리 사용할 IoT기기를 제어하여 시간을 효율적으로 활용할 수 있다.

2) 치안에 대한 불안감에 초점을 둔 기능

- 현관(CCTV) 모듈

차량 내부에서 스마트홈의 현관 모듈을 통해 집 앞의 상황을 실시간 영상으로 확인할 수 있을 뿐만 아니라, 집 앞에 외부인이 감지되었을 경우 알림과 함께 차량으로 사진을 전송한다. 전송된 사진은 시간 순으로 나열되어있다. 이를 통해 외부에 있어도 집 앞 상황을 파악할 수 있어 치안에 대한 불안감을 해소할 수 있다.

3) 거주인 부재 시 방문객에게 생기는 의사전달 제한에 초점을 둔 기능

- 현관(음성메세지) 모듈

거주자 부재 시 외부인은 스마트홈의 현관 모듈을 이용하여 본인의 소속과 방문 목적을 음성메세지로 남길 수 있다. 이러한 음성메세지는 실시간으로 알림과 함께 차량으로 전송된다. 운전자는 차량 내부에서 음성메세지를 확인할 수 있다. 이러한 점을 통해 거주인 부재 시 방문객에게 생기는 의사전달 제한을 해소할 수 있다.

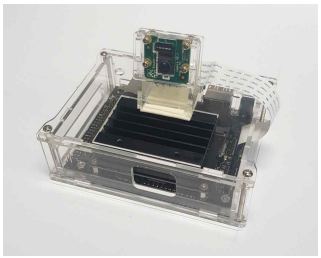
□ 개발 환경 설명

○ Hardware 구성

- ①차량 대시보드

├── Raspberry Pi 4 Model B 4GB
├── 7'inch TouchScreen
├── 마이크
└── 스피커

- ②차량내 얼굴 인식 모듈

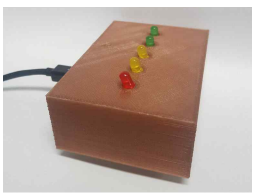


├── NVIDIA Jetson Nano Board
├── 아크릴 케이스 (SZH-JET004)
└── 카메라 (RPI 8MP CAMERA V2)

- ③냉장고 모듈

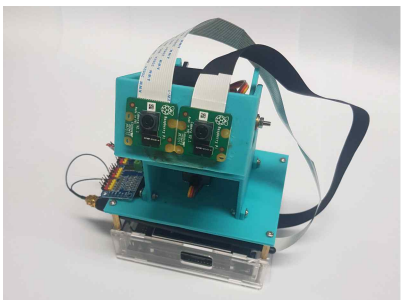
├── Raspberry Pi 4 Model B 4GB
├── TouchScreen (7inch HDMI LCD)
├── 아크릴판
└── 카메라 (RPI 8MP CAMERA V2)

- ④전기장판 모듈 ⑤욕조 모듈



├── Arduino WeMos D1 Mini Board
├── Led, Resistance, breadboard
└── 3D 프린팅 본체 (필라멘트)

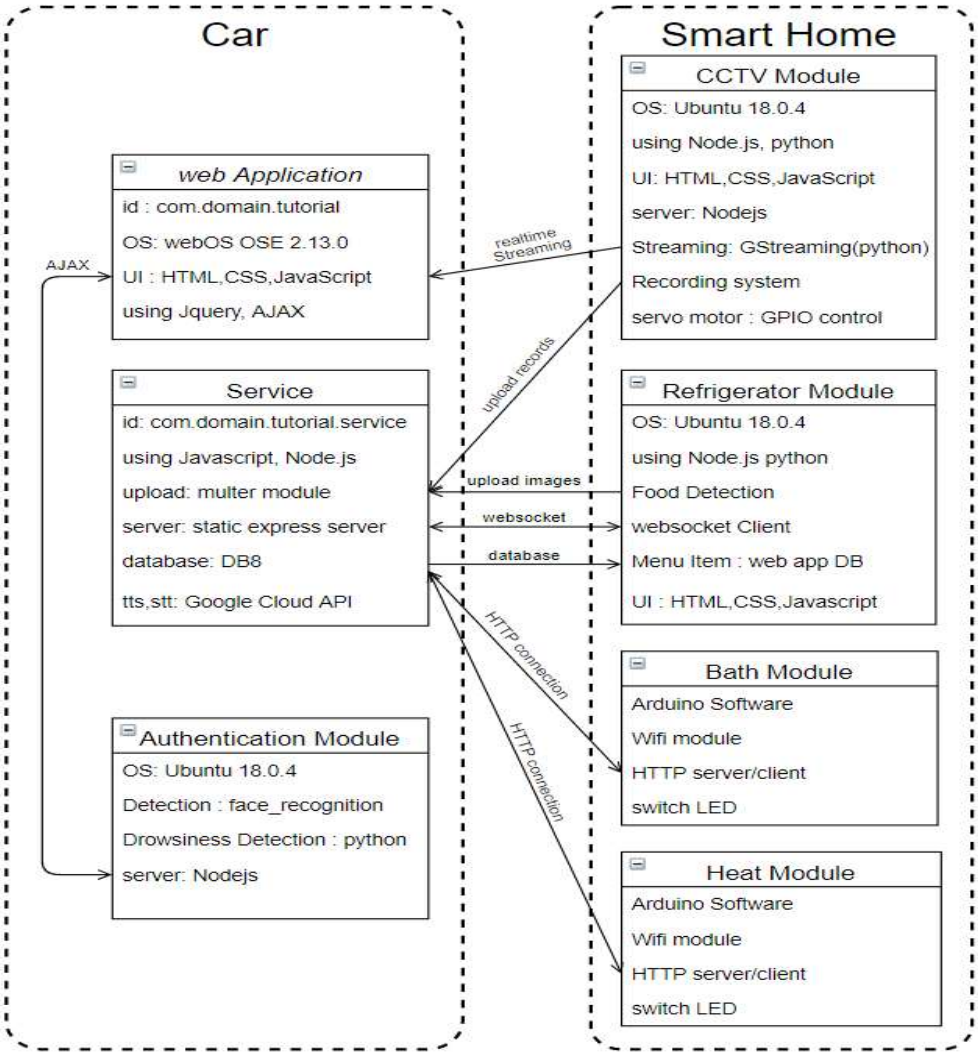
- ⑥현관 CCTV 모듈



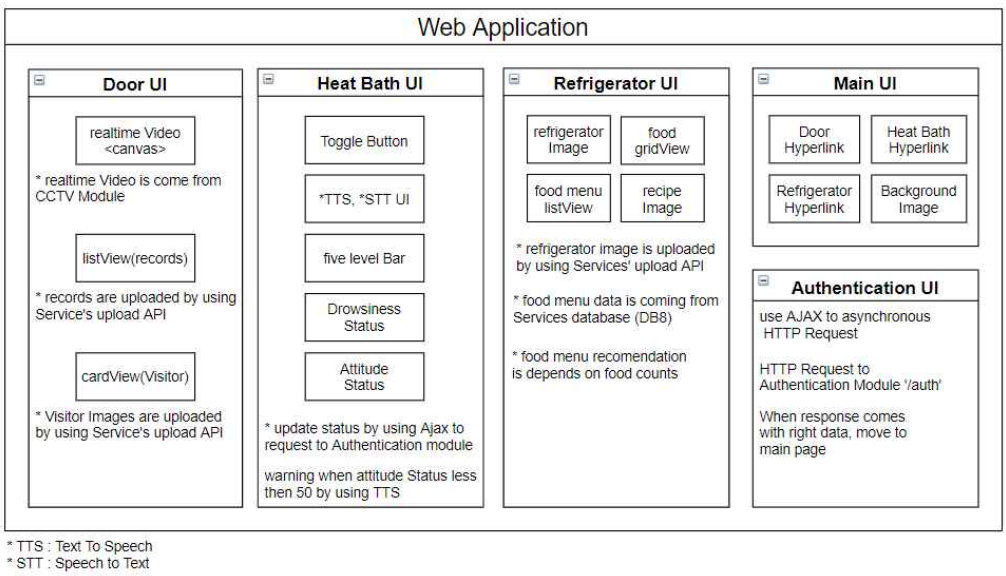
├── NVIDIA Jetson Nano B01 Board
├── 아크릴 케이스 (SZH-JET004)
├── 카메라 (RPI 8MP CAMERA V2) x2
├── 마이크
└── 3D 프린팅 본체 (필라멘트)

○ Software 구성

- 전체 구성도

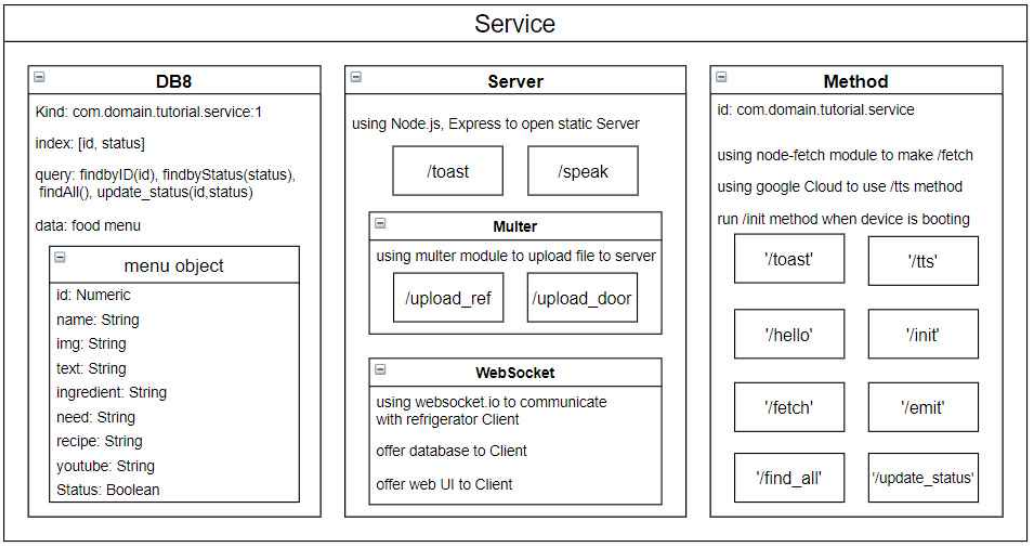


- 웹 어플리케이션 (①차량 대시보드)

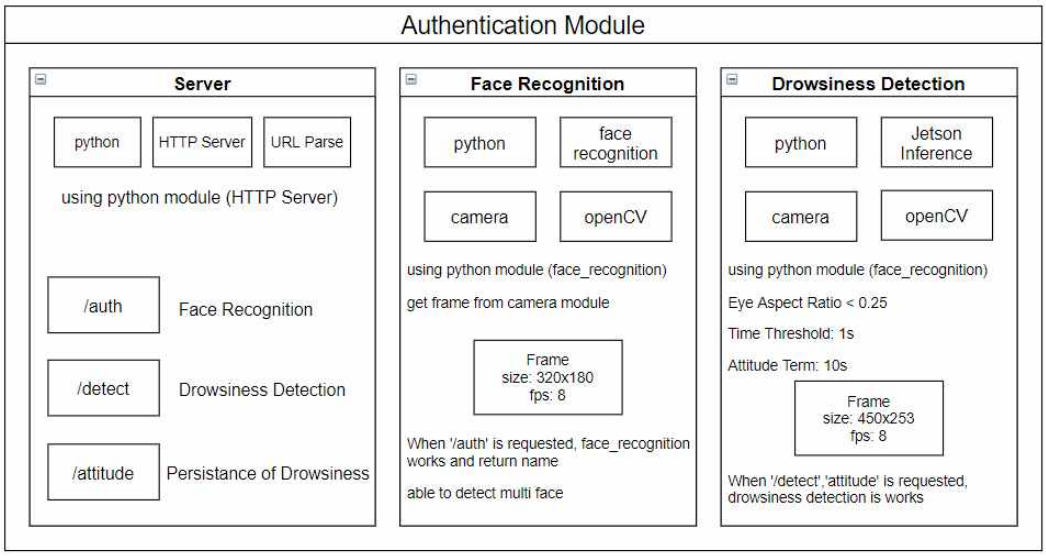


* TTS : Text To Speech
* STT : Speech to Text

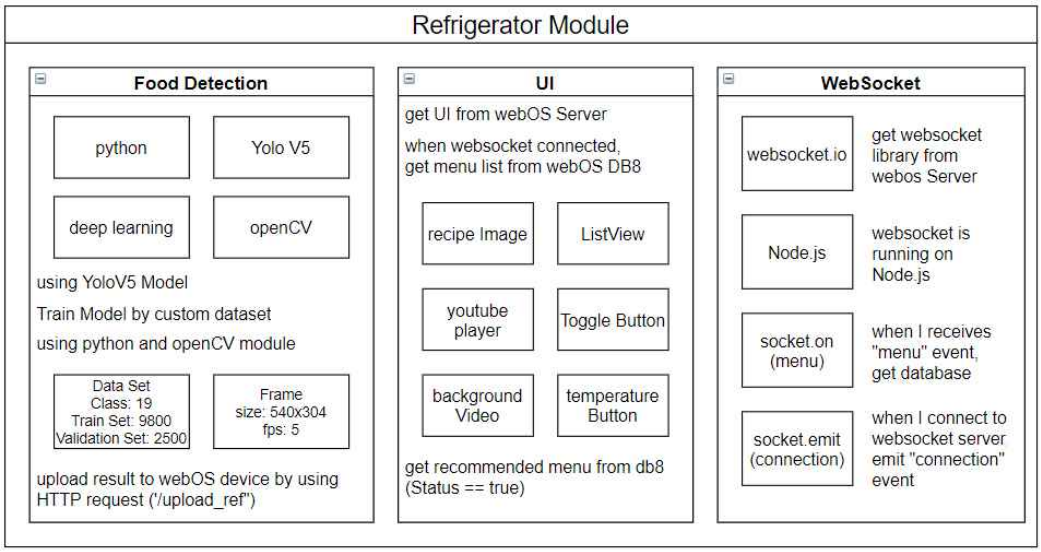
- 서비스 (①차량 대시보드)



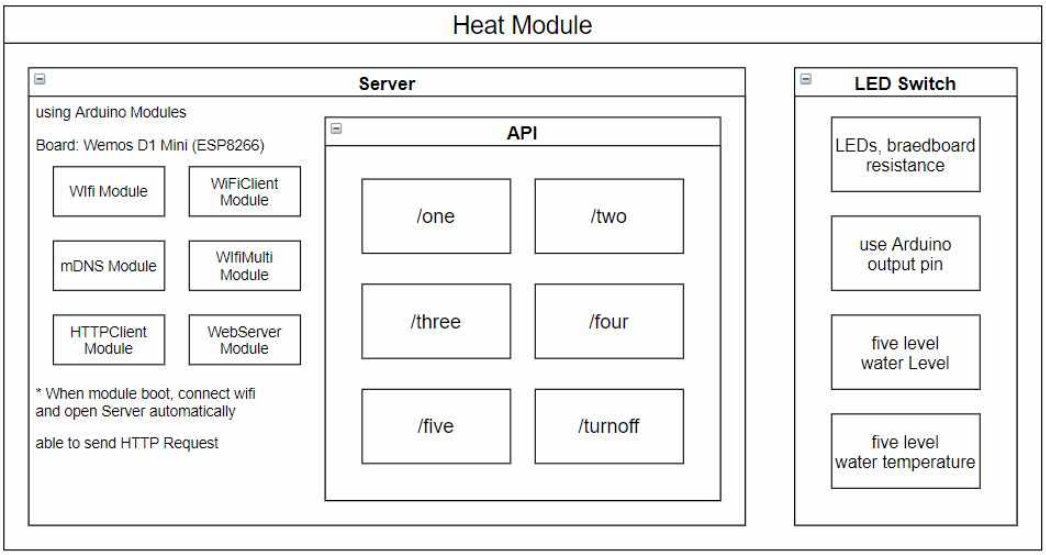
- 인증 모듈 (②차량내 얼굴 인식 모듈)



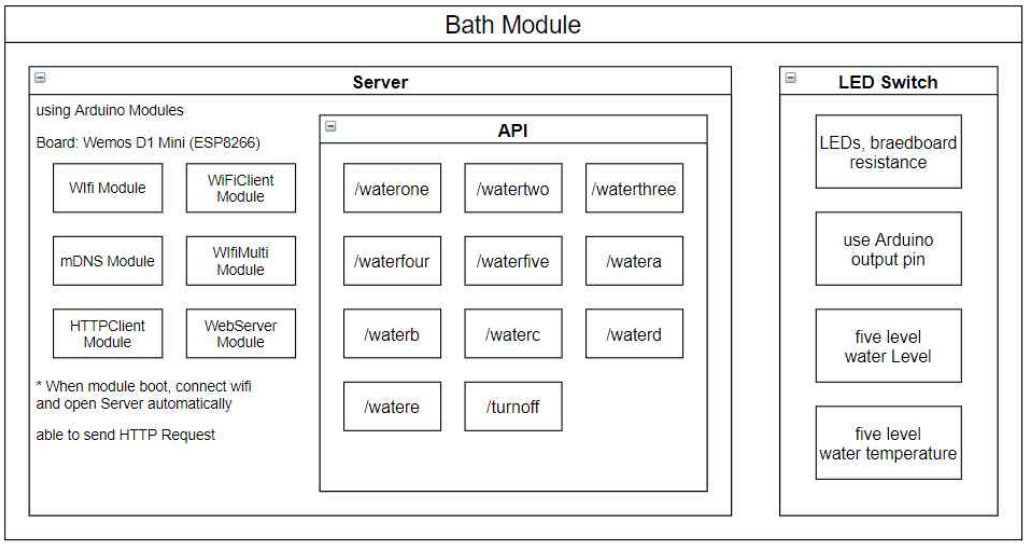
- 냉장고 모듈 (③냉장고 모듈)



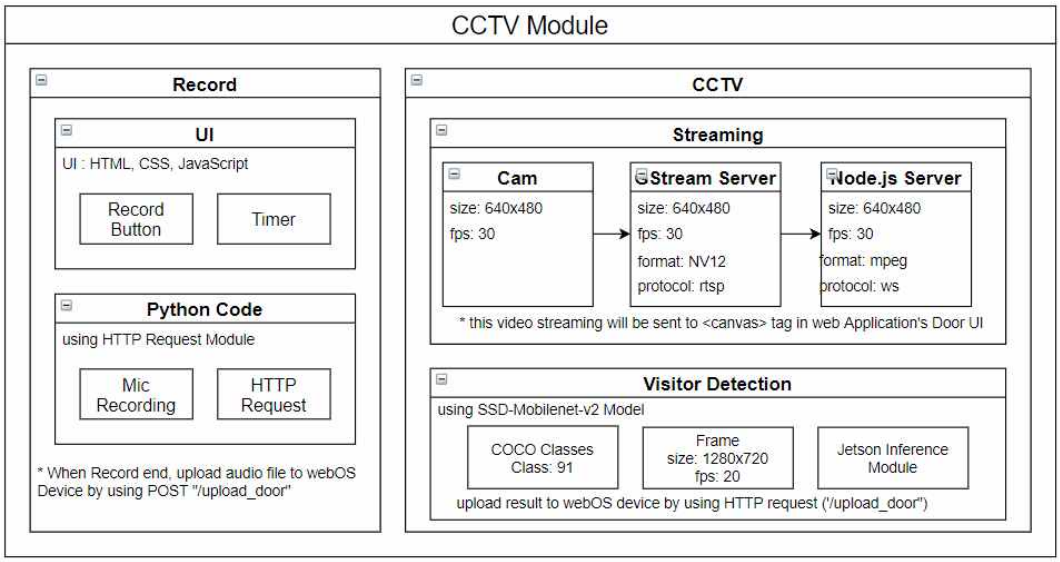
- 전기장판 모듈 (④전기장판 모듈)



- 욕조 모듈 (⑤욕조 모듈)



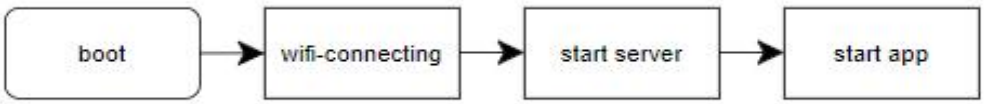
- CCTV 모듈 (⑥현관 CCTV 모듈)



○ Software 설계도 (흐름도 및 클래스 다이어그램 등 / 개발언어에 따라 선택)

- ①차량 대시보드

  ㄴ 부팅시 스크립트 실행

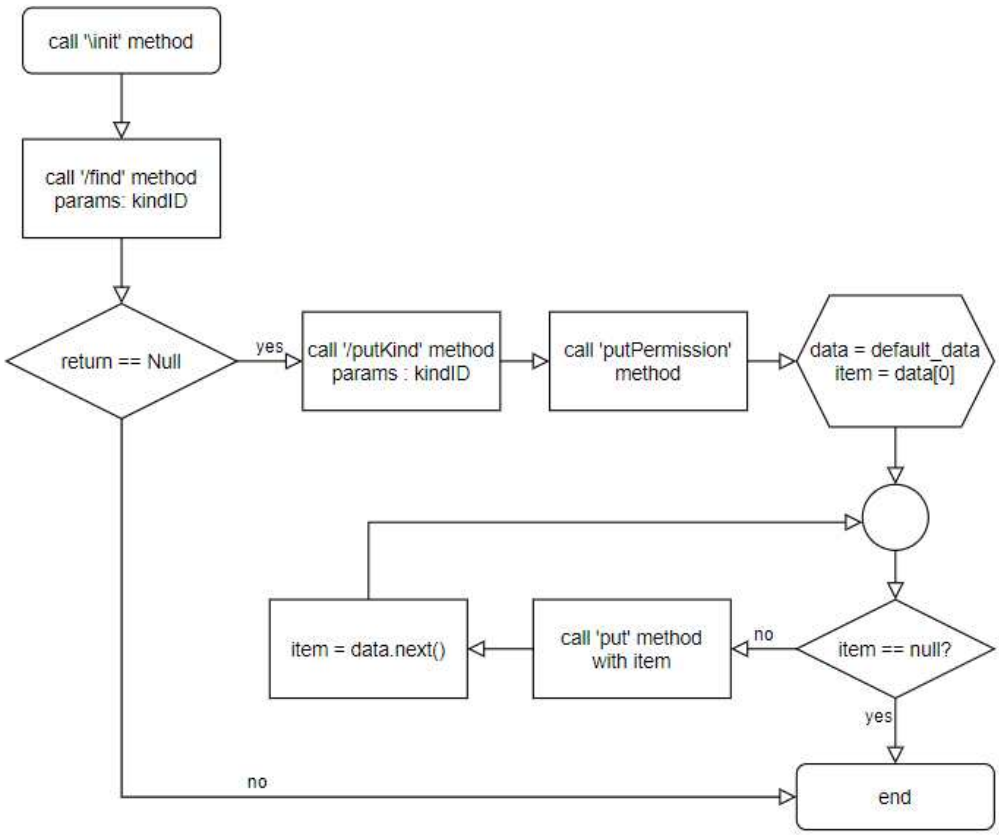


- wifi-connecting : /lib/systemd/system/runscript-wifi.service 파일에서 'com.webos.service.wifi/connect' 메서드를 호출하여 ssid와 password를 매개변수로 넘겨 자동으로 wifi 연결을 시도한다.
- Start server : /lib/systemd/system/runscript-server.service 파일에서 직접 만든 메서

드인 'com.domain.tutorial.service/init' 메서드를 호출하여 Nodejs - Express 서버를 열고 유지한다.

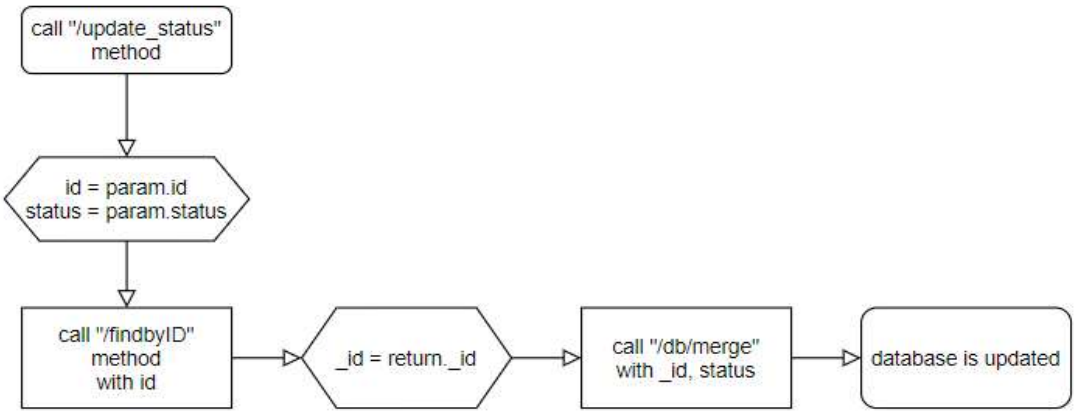
- Start app : /lib/systemd/system/runscript.service 파일에서 'com.webos.service.applicationmanager/launch' 메서드를 호출하여 직접 개발한 어플리케이션을 실행한다.

ㄴ 데이터베이스 초기화 흐름도



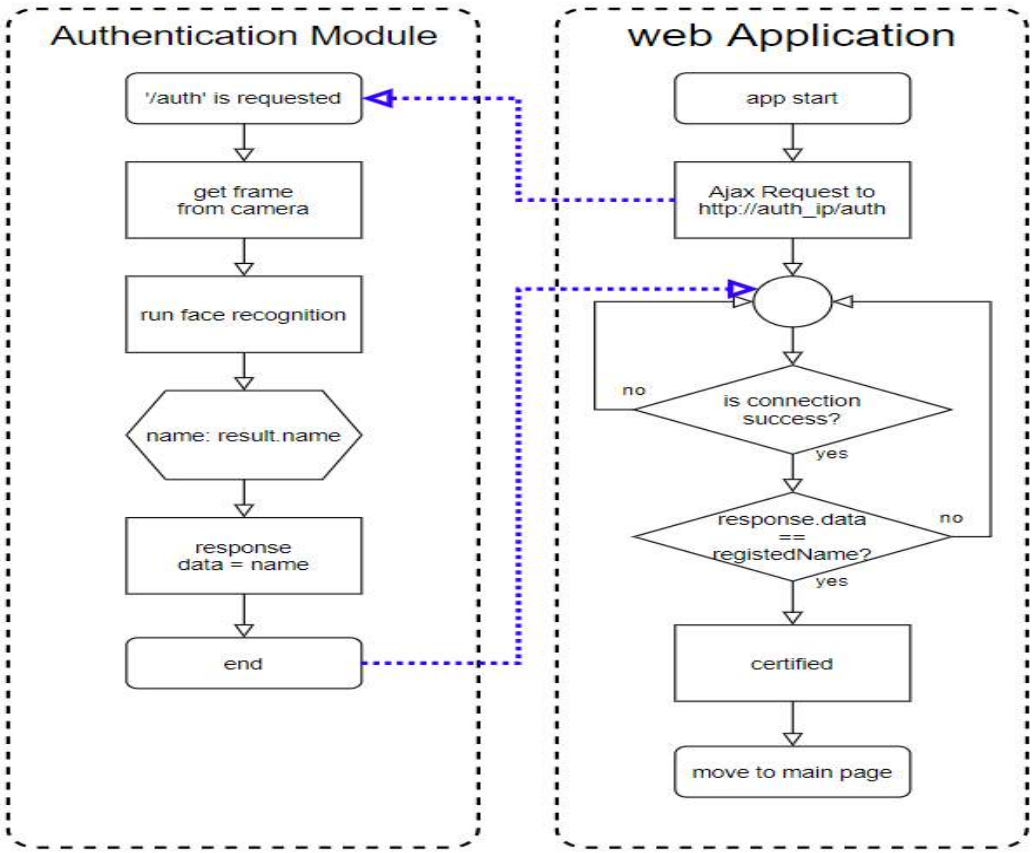
- /find method : kindID에 해당하는 Kind가 존재하는지 확인한다. Kind가 있으면 종료한다. 없으면 계속 진행한다.
- /putKind method : kindID의 id로 새로운 Kind를 생성한다.
- /putPermission method : kind의 read, create, update, delete 권한을 부여한다.
- /put : default_data.json 파일에 있는 모든 데이터 객체를 /put 명령어를 통해 Kind에 넣는다. 완료되면 종료한다.

ㄴ 데이터베이스 상태 업데이트 흐름도



- /update_status : 매개변수로 data의 id와 status를 입력 받는다.
- /findbyID : 해당 id를 갖는 데이터 객체를 query를 통해 반환 받는다.
- /db/merge : 해당 객체의 _id 변수를 이용하여 status 값을 변화시킨다.
- Database에서 id값을 이용한 Status 변경이 완료된다.

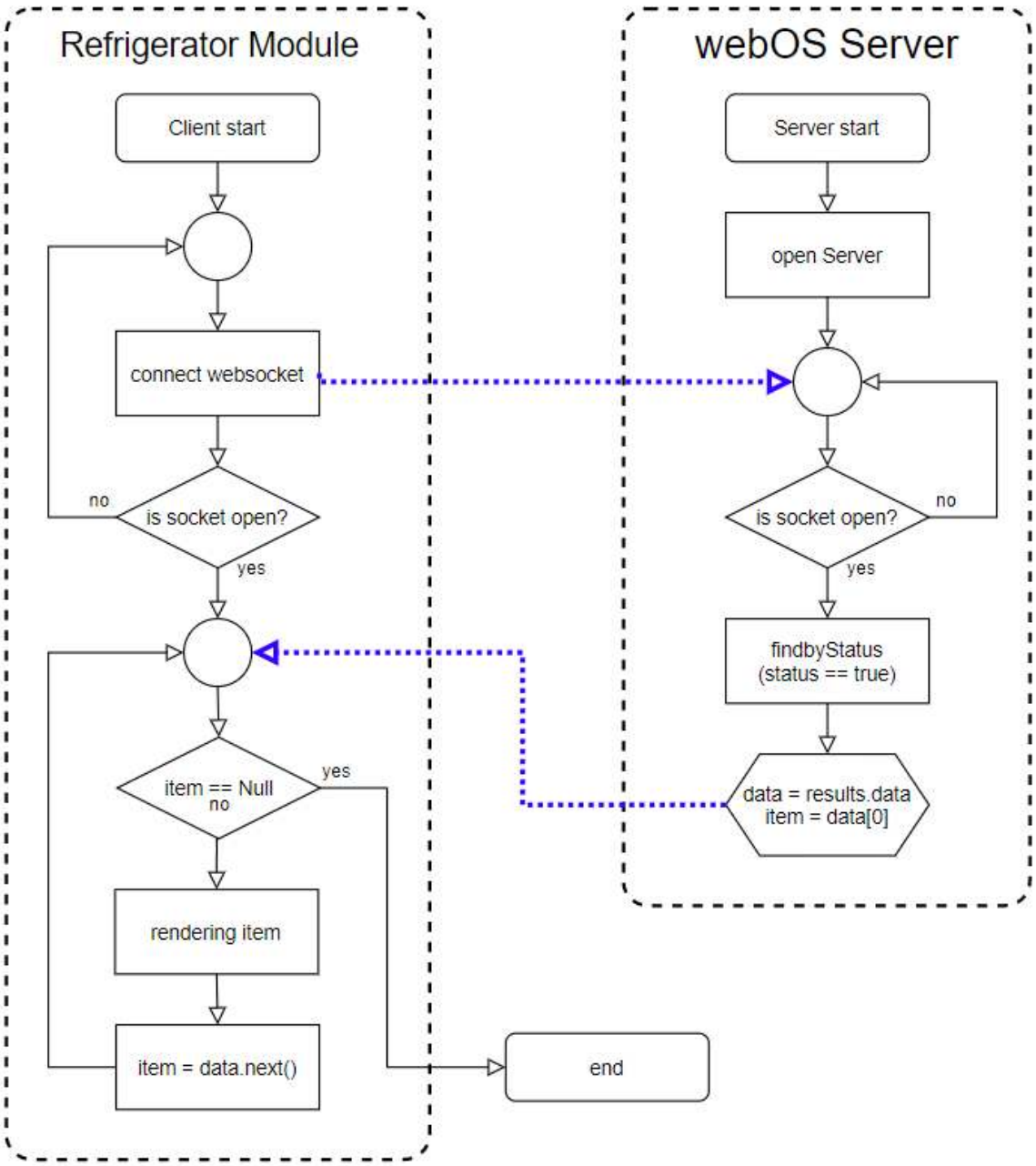
- ②차량내 얼굴 인식 모듈
  ㄴ 사용자 인증 흐름도



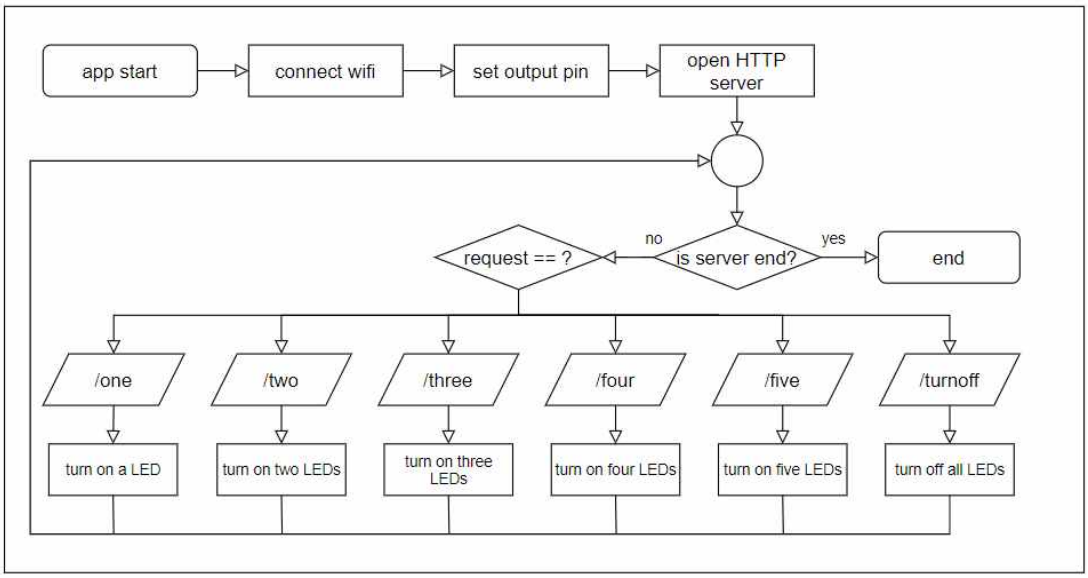
- request(/auth) : ②차량내 얼굴 인식 모듈에 http://ip:port/auth GET request를 비동기 통신으로 보낸다.
- 정상적으로 통신이 완료되면 계속 진행한다. 통신이 완료 되지 않았을 경우 다시 1번으로 돌아간다.

- response.data == registedName? : 받은 데이터가 등록된 사용자일 경우 인증을 완료한다. 미인증 사용자일 경우 인증을 위해 다시 1번으로 돌아간다.
- Certified : 인증이 완료되었으니 어플리케이션의 메인화면(main.html)로 이동한다.

- ③냉장고 모듈
  ㄴ 냉장고 모듈 메뉴 불러오기

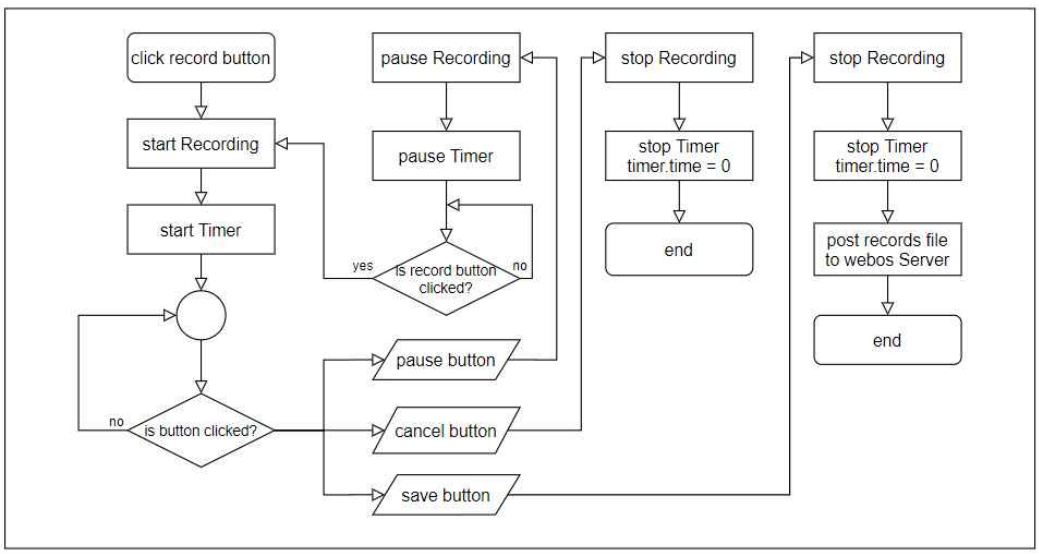


- ④전기장판 모듈, ⑤욕조 모듈
  ㄴ 모듈 흐름도

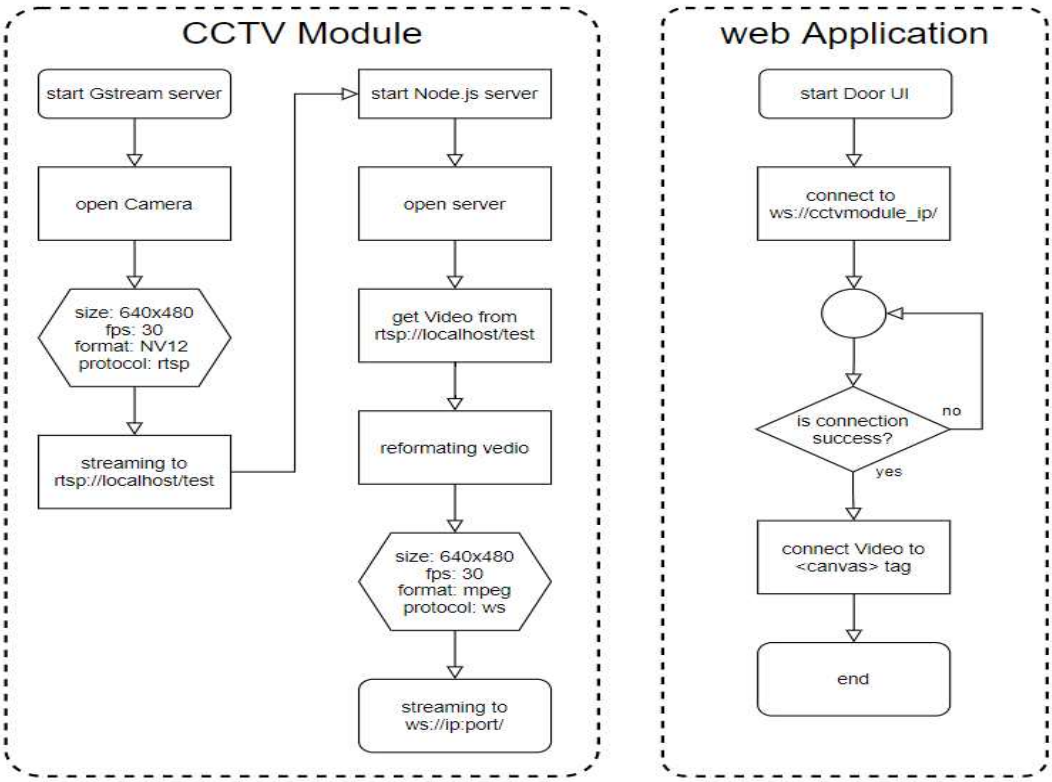


- ⑥현관 CCTV 모듈

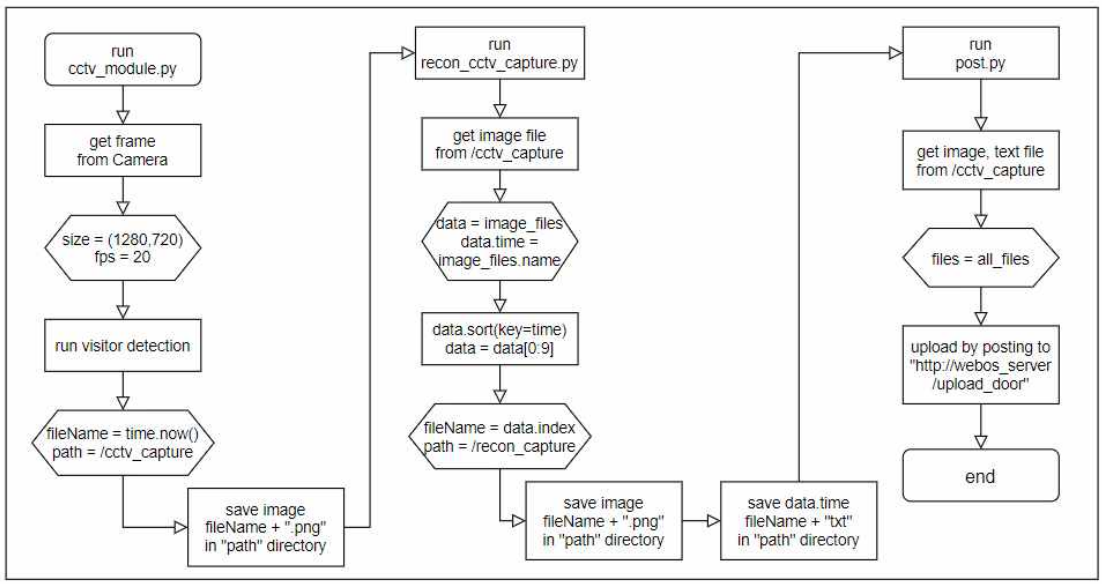
ㄴ CCTV 모듈 녹음 흐름도



ㄴ CCTV 모듈 스트리밍 흐름도



ㄴ CCTV 모듈 보행자 감지 흐름도

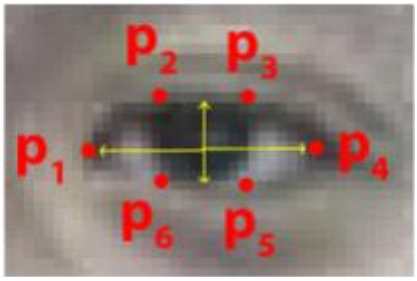


○ Software 기능 (필요 시 알고리즘 설명 포함)

- ① Drowsiness detection Argorithm

Tereza Soukupova & Jan´Cech에 의해 제시된 1)Eyes Aspect Ratio 방식을 사용한다.

1) CECH, Jan; SOUKUPOVA, Tereza. Real-time eye blink detection using facial landmarks. Cent. Mach. Perception, Dep. Cybern. Fac. Electr. Eng. Czech Tech. Univ. Prague, 2016, 1-8.

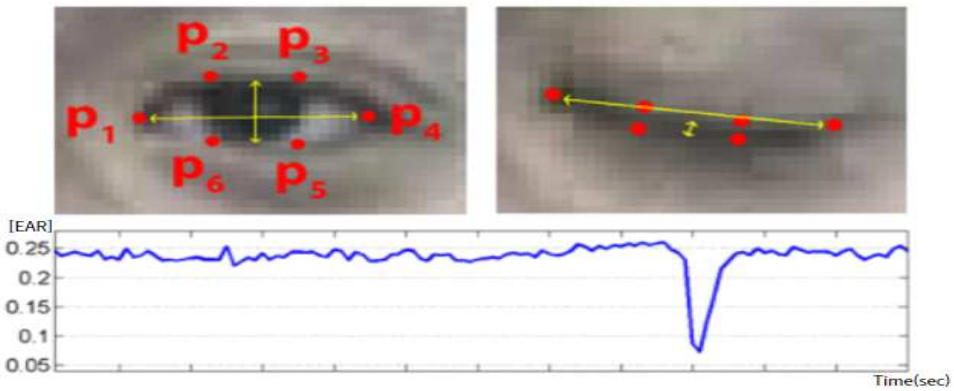


[그림3] EAR 계산을 위한 Point 정의

$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

[그림4] EAR 계산식

EAR은 그림3과 같은 정의를 바탕으로 그림4의 수식으로 계산할 수 있다.



[그림5] Point 변화에 따른 EAR값 변화

EAR은 그림5에서 보이는 것과 같이 눈을 뜨고 있을 땐 0보다 큰 실수 값을 갖게 되고, 눈을 감게 되면 0에 가까운 실수 값을 갖게 된다. 이렇게 구해진 EAR가 설정하는 임계값을 기준으로 눈을 뜨고 감고 있는 지를 판단할 수 있게 된다.
눈 깜박임을 통한 졸음 감지를 위해서는 운전자의 사고 방지를 위한 최대한 빠른 졸음 판단 시간이 필요하다. 그래서 몇 가지의 가정을 통해서 운전자에게 주어지는 대략적인 판단 시간을 계산해야 한다.

1) 고속도로 기준 적용(규정속도:100km/h, 차량 간 유지 거리 : 100m 이상)
2) 사람이 브레이크를 밟는 데 걸리는 시간(0.45초) + 브레이크 밟는 시간(0.2초) + 제동 시작까지 걸리는 시간(0.05초) = 0.7초
3) 100km/h로 달리는 차량의 제동 거리는 50m

위와 같은 가정을 통해서 운전자에게는 앞 차량과의 거리 100m중에 제동 거리를 제외한 약 50m의 여유거리가 있다. 100km/h는 1초에 27m를 이동하고 50m는 약 1.85초가 소요된다. 반응 시간 등을 고려한 시간이 0.7초라 가정 하였기에 운전자에게는 약 1.15초 정도 여유시간이 생기게 된다. 그래서 본 프로젝트에서는 여유있게 1초 정도 눈을 감고 있다면 졸음운전 상태라고 판단한다. 추가로 졸음 운전 상태가 반복되는 주기가 10초 이내라면 단순한 위험 경고 이외에도 주행 집중도가 좋지 않다는 판단을 한다.

- ② Passenger Detection Argorithm

jetson.inference 라이브러리와 COCO classes dataset의 91개의 class가 학습되어 있는

SSD-Mobilenet-v2 Network를 통해서 CCTV에서 사람을 검출하고 그에 대한 신뢰도가 70% 이상이면 사람이라고 판단한다. 이를 통해서 마스크를 쓴 사람이더라도 검출이 가능했다. 새로운 사람이 검출되면 webOS기기로 전송 하기 위해서 영상을 저장하도록 구현했다.

○ UI 사용법 (구성 및 설명)

1) 차량 내 대시보드 UI

[그림6] 차량용 대시보드 초기화면

① webOS를 통해 앱을 실행시켰을 때 처음 화면이다. 사용자 얼굴인식이 이루어지면 화면이 넘어가게 된다.

[그림7] 인증 후 메인화면

① webOS내 차량에서 3가지 모듈의 메인화면이다. 3가지 버튼으로 이루어져 있다.

[그림8] 냉장고 모듈의 냉장고 내부

① 메인화면으로 돌아갈 수 있다.
② 냉장고 내부와 메뉴 추천 화면을 오갈 수 있다.
③ 냉장고 안의 식재료를 파악한다.
④ 파악된 식재료를 글로 표현해 준다.
⑤ 파악된 식재료를 그림과 갯수로 알려준다.

[그림9] 냉장고 모듈의 메뉴 추천

① 냉장고 내부의 파악된 식재료를 통해 추천된 메뉴가 나타난다.
② 추천 메뉴의 조리법을 볼 수 있다.
③ 스마트 홈의 냉장고로 조리법을 전송한다.

[그림10] 그림9의 조리법 보기 클릭 시

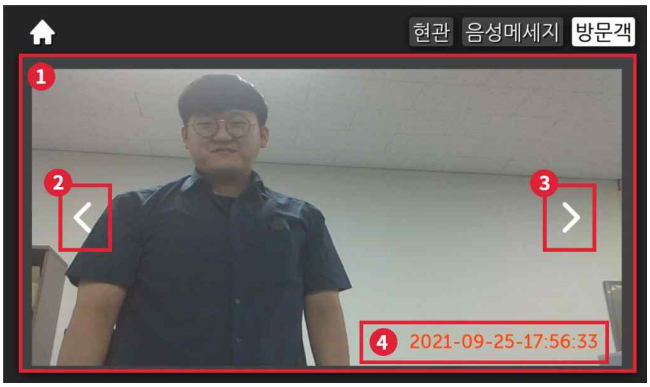
① 조리법이 나타난 모습이다.
② X 버튼을 통해 종료할 수 있다.

[그림11] 현관 모듈의 현관

① 실시간으로 집 앞 영상을 보여준다.

[그림12] 현관 모듈의 음성메세지

① 음성메세지 리스트를 확인할 수 있다.
② 삭제를 할 수 있다.
③ 음성 메세지를 재생할 수 있다.



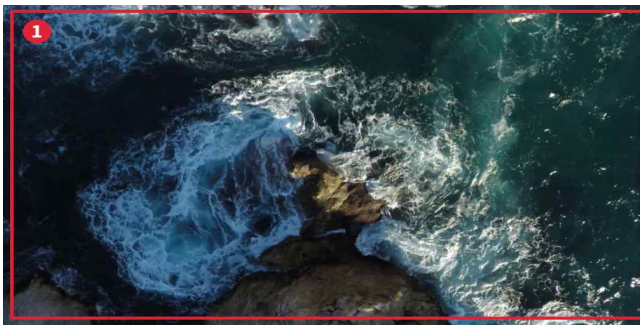
[그림13] 현관 모듈의 방문객 기록

① 방문객 사진이 출력된다.
② 이후 사진을 확인한다.
③ 이전 사진을 확인한다.
④ 방문 날짜와 시간을 확인 할 수 있다.

[그림14] 운전자 상태 맞춤형 서비스 모듈

① 주행 집중도와 졸음운전지수를 알 수 있다.
② 전기장판 모듈의 전원을 켜고 끌 수 있고 온도의 단계를 설정할 수 있다.
③ 욕조 모듈의 전원을 켜고 끌 수 있고 온도와 수위의 단계를 설정할 수 있다.

2) 스마트홈의 IoT 모듈 UI



[그림15] 스마트 홈의 냉장고 모듈 대기화면

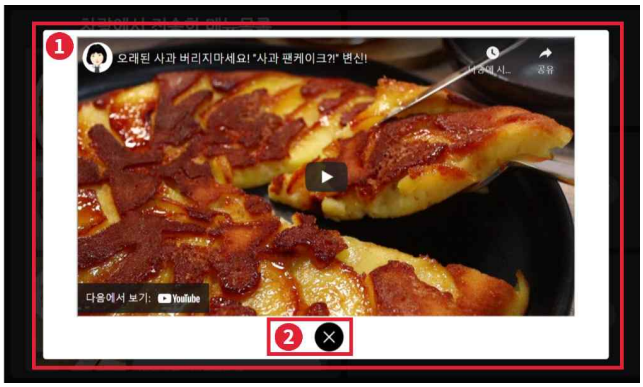
① 스마트 홈의 냉장고 모듈 대기화면이다.

[그림16] 스마트 홈의 냉장고 모듈 실행화면

① 차량의 냉장고 모듈에서 받아온 메뉴의 모습이다.
② 조리법을 확인할 수 있다.
③ 조리법과 관련된 영상을 볼 수 있다.
④ 냉장고의 온도를 조절 할 수 있다.
⑤ 대기화면으로 돌아갈 수 있다.

[그림17] 그림16의 조리법 보기 클릭 시

① 조리법을 보여준다.
② X버튼을 통해 종료할 수 있다.



[그림18] 그림16의 조리법 영상보기 클릭 시

① 조리법 관련 영상을 보여준다.
② X버튼을 통해 종료할 수 있다.

[그림19] 스마트 홈의 현관 모듈

① 초인종을 누를 수 있다.
② 음성메세지를 남길 수 있다.

[그림20] 그림19의 음성메시지 클릭 시

① 이전화면으로 돌아갈 수 있다.
② 방문자의 소속을 선택할 수 있다.

① 이전화면으로 돌아갈 수 있다.
② 녹음 시작과 종료를 할 수 있다.
③ 녹음이 진행되는 시간을 알 수 있다.
④ 녹음을 취소할 수 있다.

[그림21] 그림20의 소속 선택 후

○ 개발환경 (언어, Tool, 사용시스템 등)

- 언어

HTML, CSS, Javascript	Python	C++

- Tools

Visual Studio Code	Github	Source Tree
Notion	MobaXterm	Arduino
Pytorch	YOLO v5	Google Cloud API (TTS, STT)
Node.js	Node.js express	Node.js socket.io
Node.js multer	Node.js fs	webOS CLI
webOS LS2 API	Python OpenCV	Python face_recognition
Python http.server	Python numpy	GStreamer
Jetson.utils	Jetson.inference	

- 사용시스템

macOS Big Sur	Ubuntu 18.04	Xubuntu 20.04
Windows 10	webOS 2.13.1	

□ 개발 프로그램 설명

○ 파일 구성

```
.
├── README.md
├── debug.bat
├── module
│   ├── Authentication_module
│   │   ├── node.js
│   │   │   ├── app.js
│   │   │   └── index.js
│   │   └── python
│   │       ├── camera.py
│   │       ├── knowns
│   │       │   └── Jinwoo.jpg
│   │       ├── server.py
│   │       └── user_recog.py
│   ├── Bath_module
│   │   └── WebOs_bath.ino
│   ├── CCTV_module
│   │   ├── UI
│   │   │   ├── meter_reading.html
│   │   │   ├── neighbor.html
│   │   │   ├── parcel.html
│   │   │   ├── public.html
│   │   │   ├── script.js
│   │   │   └── style.css
│   │   ├── app.js
│   │   ├── index.html
│   │   ├── package-lock.json
│   │   ├── package.json
│   │   ├── python
│   │   │   ├── CCTV_module.py
│   │   │   ├── audio
│   │   │   │   ├── audio_post.py
│   │   │   │   ├── state.txt
│   │   │   │   ├── state2.txt
│   │   │   │   └── yourname.txt
│   │   │   ├── recon_cctv_capture.py
│   │   │   └── server.py
│   │   ├── style.css
│   │   └── voice_message.html
│   ├── Heat_module
│   │   └── webOS_heat.ino
│   └── Refrigerator_module
│       └── food_detection
│           ├── Dockerfile
│           ├── __pycache__
│           │   └── camera.cpython-37.pyc
│           ├── best.pt
│           ├── camera.py
│           ├── data
│           │   ├── frame
│           │   │   └── food.jpg
│           │   └── images
│           │       └── food.jpg
│           ├── detect.py
│           ├── models
│           │   ├── __init__.py
│           │   ├── __pycache__
│           │   │   ├── __init__.cpython-37.pyc
│           │   │   ├── common.cpython-37.pyc
│           │   │   ├── experimental.cpython-37.pyc
│           │   │   └── yolo.cpython-37.pyc
│           │   ├── common.py
│           │   └── experimental.py
│           ├── runs
│           │   └── detect
│           └── utils
│               ├── __init__.py
│               ├── __pycache__
│               │   ├── __init__.cpython-37.pyc
│               │   ├── augmentations.cpython-37.pyc
│               │   ├── autoanchor.cpython-37.pyc
│               │   ├── datasets.cpython-37.pyc
│               │   ├── downloads.cpython-37.pyc
│               │   ├── general.cpython-37.pyc
│               │   ├── metrics.cpython-37.pyc
│               │   ├── plots.cpython-37.pyc
│               │   └── torch_utils.cpython-37.pyc
│               ├── activations.py
│               ├── augmentations.py
│               ├── autoanchor.py
│               ├── aws
│               │   ├── __init__.py
│               │   ├── mime.sh
│               │   ├── resume.py
│               │   └── userdata.sh
│               ├── callbacks.py
│               ├── datasets.py
│               ├── general.py
│               ├── loggers
│               │   ├── __init__.py
│               │   └── wandb
│               │       ├── __init__.py
│               │       ├── log_dataset.py
│               │       ├── sweep.py
│               │       ├── sweep.yaml
│               │       └── wandb_utils.py
│               ├── loss.py
│               ├── metrics.py
│               ├── plots.py
│               └── torch_utils.py
├── service
│   ├── db.js
│   ├── default_db.json
│   ├── luna.js
│   ├── package-lock.json
│   ├── package.json
│   ├── refrigerator_client
│   │   ├── index.html
│   │   ├── main.html
│   │   ├── menu_recommand
│   │   ├── recipe_img
│   │   ├── script.js
│   │   ├── style.css
│   │   └── video.mp4
│   ├── sample.html
│   ├── server.js
│   ├── service.js
│   └── services.json
└── webapp
    ├── appinfo.json
    ├── background.png
    ├── car.jpg
    ├── door1
    │   ├── door.html
    │   ├── left.png
    │   ├── luna.js
    │   ├── profile.jpg
    │   ├── right.png
    │   ├── script.js
    │   ├── style.css
    │   └── uploads
    ├── heatbath1
    │   ├── heatbath.html
    │   ├── luna.js
    │   ├── profile.jpg
    │   ├── script.js
    │   └── style.css
    ├── icon.png
    ├── index.html
    ├── luna.js
    ├── main.html
    ├── refrigerator1
    │   ├── 11.jpg
    │   ├── index.js
    │   ├── luna.js
    │   ├── memo.png
    │   ├── menu_recommand
    │   ├── object_img
    │   ├── profile.jpg
    │   ├── recipe_img
    │   ├── refrigerator.html
    │   ├── script.js
    │   ├── style.css
    │   └── uploads
    ├── script.js
    └── style.css

39 directories, 116 files
```

○ 함수별 기능

- Authentication module

1. user_recog.py

- FaceRecog() : Face ID를 구현하기 위해서 class로써 구현했다.
- face_recog.get_frame() : FaceRecog()의 메소드 중 하나로 영상을 출력하며 해당 영상에서 저장된 사용자인지 추정한 결과도 출력한다.
- camera.get_frame() : 카메라 모듈을 통해 입력받은 frame을 가져온다.
- cv2.resize() : 사용자가 정의한 사이즈만큼 frame의 width, height을 조절한다.
- face_recognition.face_locations() : face_recognition 라이브러리의 메소드 중 하나로 얼굴 영역과 특징을 추출한다.
- face_recognition.face_encodings() : face_recognition 라이브러리의 메소드 중 하나로 얼굴 영역과 특징을 추출한다.
- face_recognition.face_distance() : face_recognition 라이브러리의 메소드 중 하나로 얼굴 특징점 사이의 거리를 구한다.
- eye_aspect_ratio() : 입력은 검출된 랜드마크 중 눈의 랜드마크만을 받고, Eyes Aspect Ratio를 계산하여 출력한다.
- dlib.get_frontal_face_detector(): 영상에서 사람의 정면을 찾아내는 객체를 생성한다.
- dlib.shape_predictor("./shape_predictor_68_face_landmarks.dat") : 입력되는 데이터에 적합한 랜드마크들의 위치를 추정하는 객체를 생성한다.

2. server.py

- SimpleHTTPRequestHandler(BaseHTTPRequestHandler) : BaseHTTPRequestHandler를 상속받는 class를 구현했다.
- SimpleHTTPRequestHandler.do_GET() : 받게 되는 request에 대한 각각의 response를 구현했다.
- HTTPServer(('0.0.0.0', port), SimpleHTTPRequestHandler) : 지정한 port를 통해서 통신하는 HTTP server를 생성한다.
- httpd.serve_forever() : 위에서 생성된 객체를 server로써 유지 시키는 반복 함수이다.

- CCTV module

1. CCTV_module.py

- net = jetson.inference.detectNet("ssd-mobilenet-v2", threshold=0.5) : 사전에 다운로드 받은 ssd-mobilenet-v2 모델의 객체를 생성한다.
- camera = jetson.utils.videoSource(video source,[argv]) : video source를 지정하고 argv를 통해서 입력에 대한 회전, 크기, 프레임 등을 설정하여 객체를 생성한다.
- display = jetson.utils.videoOutput(display source) : display source를 지정하여 영상을 출력하기 위한 객체를 생성한다.

- camera.Capture() : camera 객체로 부터 1 frame의 영상을 출력받는다
- detection = net.Detect() : net 객체에 1 frame의 영상을 입력하고 객체 선언시 지정한 network를 통과시켜 검출 결과를 출력한다.
- detection.ClassID = 검출 결과 중 Class번호를 출력한다.(detection.ClassID = 1 은 사람을 의미한다.)
- detection.Confidence = 검출 결과에 해당하는 Class번호에 대한 확률을 출력한다.(사람에 대한 확률은 얼굴이 보일 수록 높은 경향이 있다.)

2. recon_cctv_capture.py: webOS 어플리케이션에 보내기 위해 방문자 인식된 프레임을 시간순으로 재가공한다.

3. audio_post.py
   - pyaudio.Pyaudio() : pyaudio 라이브러리의 메소드 중 하나로 파이썬에서 오디오 객체를 생성한다.
   - pyaudio.Pyaudio().open() : pyaudio 라이브러리의 메소드인 Pyaudio()를 활용해 오디오 객체를 생성하고 난 후 오디오 객체에 기본셋팅값을 넣어주어 실행하는 메소드이다.
   - requests.post() : requests 라이브러리 메소드 중 하나로 HTTP에 POST를 요청할 때 쓰인다.

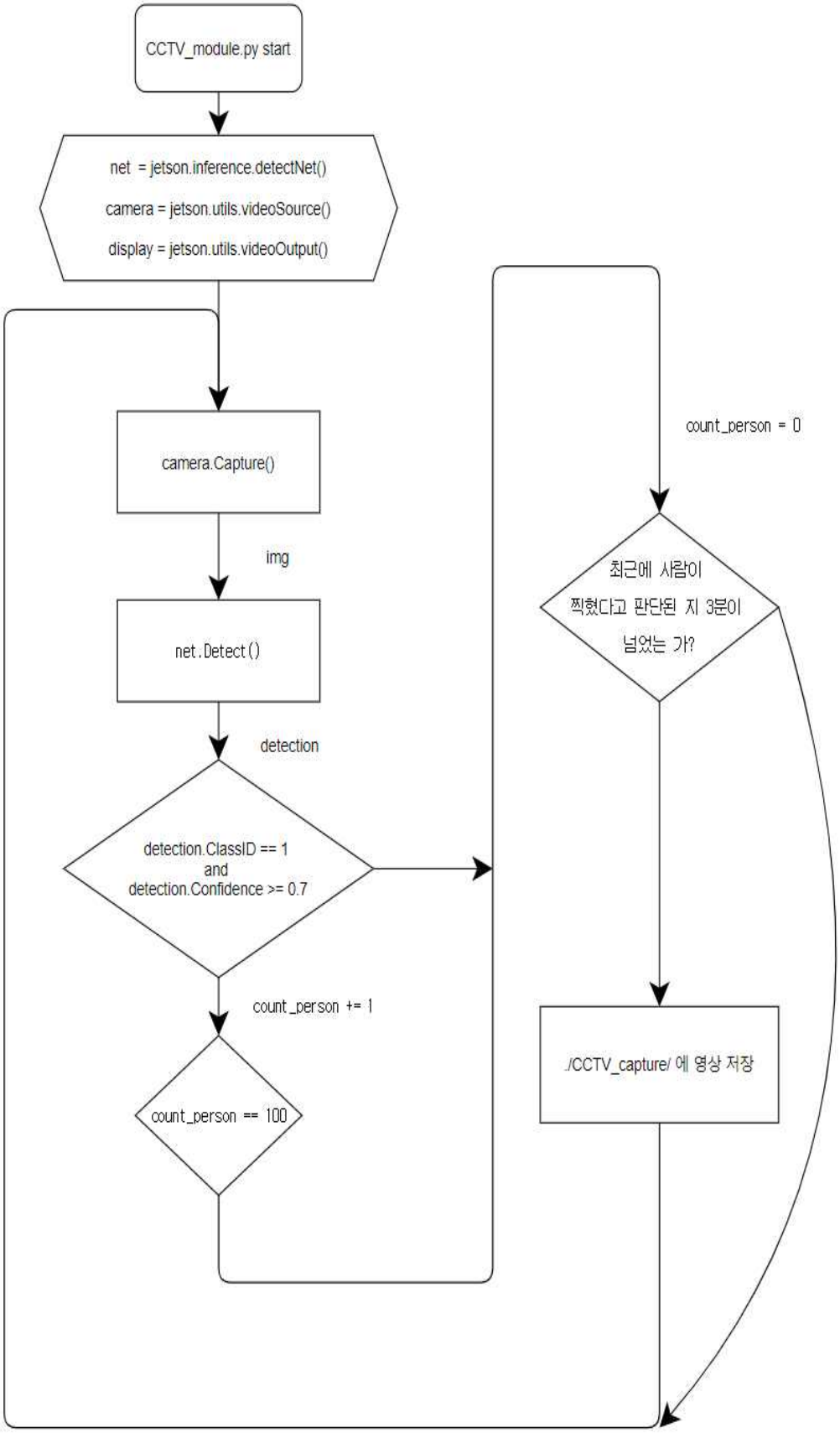
4. server.py
   - SimpleHTTPRequestHandler(BaseHTTPRequestHandler) : BaseHTTPRequestHandler를 상속받는 class를 구현했다.
   - SimpleHTTPRequestHandler.do_GET() : 받게 되는 request에 대한 각각의 response를 구현했다.
   - HTTPServer(('0.0.0.0', port), SimpleHTTPRequestHandler) : 지정한 port를 통해서 통신하는 HTTP server를 생성한다.
   - httpd.serve_forever() : 위에서 생성된 객체를 server로써 유지 시키는 반복 함수이다.

- Refrigerator module

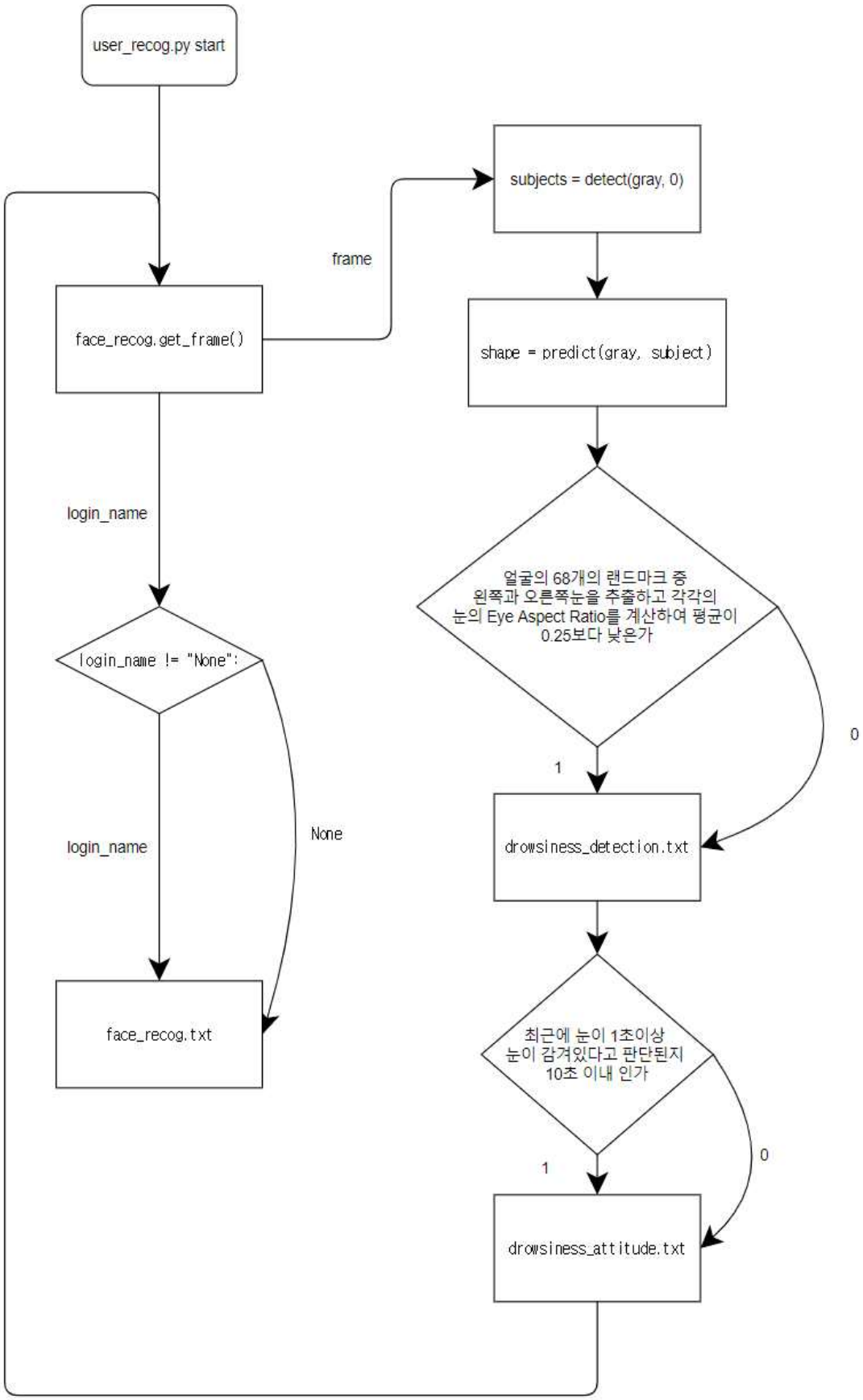
1. detect.py
   - camera.get_frame() : 카메라 모듈을 통해 입력받은 frame을 가져온다.
   - cv2.imwrite() : 특정 경로에 해당 frame을 저장한다.
   - requests.post() : requests 라이브러리 메소드 중 하나로 HTTP에 POST를 요청할 때 쓰인다.
   - parse_opt() : 인식하고자 하는 frame의 경로를 넣어주면 default값에 따라 Object Detection을 수행한다.

○ 주요 함수의 흐름도

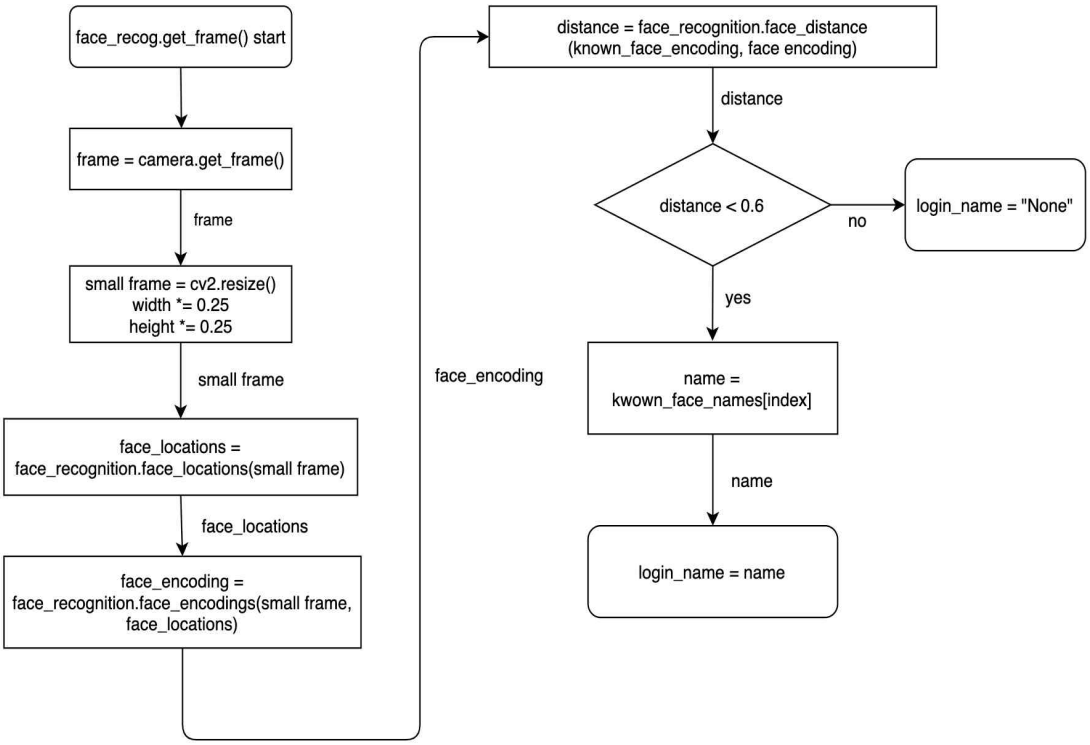
- CCTV_module



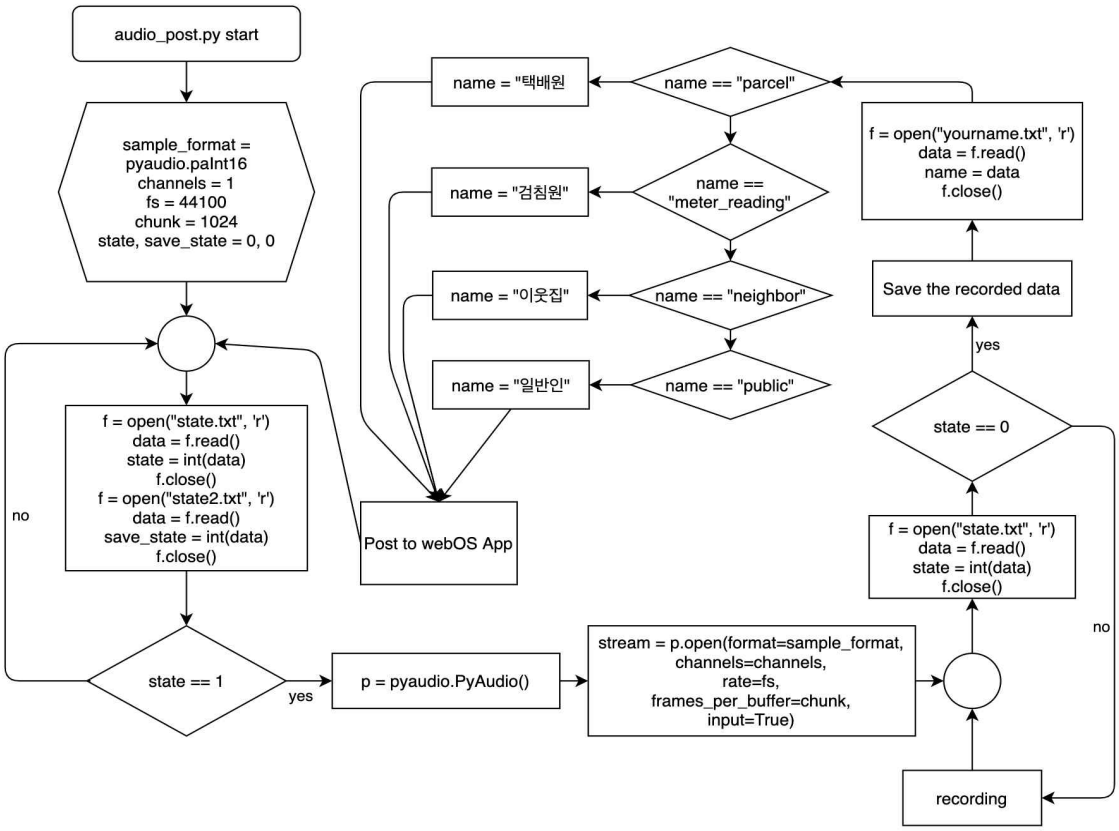
- user_recog.py



- face_recog.get_frame()



- audio_post.py



○ 기술적 차별성

- 다양한 임베디드 활용

  Jetson Nano, Raspberry Pi, Arduino와 같은 다양한 보드를 사용하였다. Jetson Nano 보드의 경우 Nvidia에서 제공하는 TensorRT SDK를 활용하여 GPU를 효율적으로 사용할 수 있기 때문에 영상처리 분야와 인공지능 연산에 장점을 갖고, 이에 따라서 영상 분야 인공지능이 사용된 모듈인 CCTV나 생체인식 모듈을 보다 좋은 성능으로 개발 할 수 있었다. Arduino 보드의 경우 비교적 경제적이고 다양한 하드웨어 부품을 쉽게 연동할 수 있으므로 보다 수월하게 IoT 모듈을 제작할 수 있었다.

- YOLOv5를 활용한 Object Detection

  냉장고 내부 식재료를 인식하기 위해서 YOLOv5를 사용하였다. YOLOv5의 pre-trained model 중에서 임베디드의 저 사양 환경에서 잘 돌아갈 수 있도록 크기가 작은 model인 yolovs.pt을 사용하여 학습시켰다. 또한 Open images Dataset V6+에 있는 식재료 관련 19개 Class를 활용하여 Custom Dataset을 만들어 학습시켰다.

  위의 과정을 거쳐 만들어진 model을 활용하여 파이썬 프로그램인 detect.py을 작동시키면 스마트홈의 냉장고 모듈에 설치된 카메라를 통해서 실시간으로 입력 frame을 받고 3~4초 가량의 연산과정을 거쳐 학습한 19개 식재료 Class에 관한 결과값을 얻을 수 있다. 이러한 결과값은 webOS 어플리케이션에 실시간 전송되어 사용자가 냉장고 내부 식재료를 차량 내부에서도 편리하게 파악할 수 있다.

- Database API DB8을 활용

  냉장고 내부의 식재료에 따른 레시피 추천을 위해서 레시피 데이터 저장을 해야 하는 데 이후에 데이터 확장과 중복방지를 위한 방법으로써 webOS에서 사용하는 Database API DB8을 활용하였다.

- Node.js multer모듈을 활용한 파일 전송시스템 구축

  Node.js express는 사용자가 업로드한 파일을 받아서 저장하는 기본 기능을 제공하지 않는다. 그러므로 이러한 문제점을 해결하기 위해서 Node.js의 multer모듈을 활용했다.

  따라서 현관(CCTV) 모듈에서의 방문자 사진, 현관(음성메시지) 모듈에서의 방문자 음성메시지, 냉장고 모듈에서의 냉장고 내부 식재료 인식결과 사진을 홈 IoT모듈에서 webOS 어플리케이션으로 전송하는 것이 가능했다.

- Drowsiness Detection Detection을 활용한 졸음운전 지수, 주행 집중도 계산

  인공지능과 공학적으로 설계한 Drowsiness Detection을 통해서 운전자의 졸음 지수,집중도를 모니터링할 수 있다. 이를 통해서 운전자는 TTS, STT를 통해서 운전 중에도 손을 사용하지 않고 스마트 홈 서비스를 제공받을 수 있다.

□ 개발 중 장애요인과 해결방안

○ webOS OSE 빌드 문제 발생

장애요인	webOS OSE의 의존성 라이브러리인 Ostree에서 master 브랜치가 main 브랜치로 바뀌면서 빌드 툴인 Yocto에서 문제가 발생함.
해결방안	해당 문제에 대해서 issue를 open하였고, shr-project@036edb4를 cherrypick 하거나 다음 버전의 webOS OSE를 기다리라는 답변을 받았다. 이미 빌드된 prebuilt-image를 다운로드 받아 사용하였다.

○ ares-install 문제 발생

장애요인	ares-install 명령어 실행시 응답 없이 대기 해야 하는 문제가 발생함.
해결방안	webOS OSE가 라즈베리파이 4B 8G 장치에서는 ares-install이 제대로 동작하지 않는 오류가 있어서, 라즈베리파이 4B 4G 장치로 SD카드를 옮겨서 동작하였다.

○ ares-install –r 문제 발생

장애요인	webOS OSE에 설치한 service에 런타임 오류가 있을 경우 앱 설치는 정상적으로 되지만, 앱 삭제시에는 에러와 함께 앱 삭제를 강제로 할 수 없는 문제가 발생함.
해결방안	1. $ journalctl –f 명령어를 통해 Service에서 발생한 문제부분의 코드를 확인하고 해당 폴더(/media/developer/apps/usr/palm/services/[Service])에서 코드를 수정한다. 2. 해당 폴더를 삭제하여 앱을 강제로 삭제한다.

○ Express Server를 실행시 자동 종료되는 문제 발생

장애요인	webOS OSE에서 서비스로 Express 서버를 실행할 경우 곧바로 종료가 되는 문제 발생함.

해결방안	$ journalctl –f 명령어를 통해 로그를 확인한 결과 Express 서버가 오랫동안 대기하면 activitymanager에서 해당 activity를 종료하기 때문에, Express 서버가 자동 종료되는 것으로 확인.
	1. "service.activityManager.idleTimeout = 60*60*24*7;" 코드를 통하여 해당 activity가 동작하지 않더라도 꺼지지 않는 시간을 1주일로 설정한다.
	2. var keepAlive;
	service.activityManager.create("keepAlive", function(activity){
	keepAlive = activity; }); 코드를 사용하여 activity를 명시적으로 실행시키고 계속 유지되도록 설정합니다.
	이와 같이 정적인 Express 서버를 실행할 수 있었고, 추가적으로 systemd를 사용하여 부팅시 스크립트 실행을 구현하여, 라즈베리파이 기기 부팅시 정적인 Express 서버를 자동으로 실행하도록 했습니다.

□ 개발결과물의 차별성

○ 운전자 상태 인식을 통한 추천서비스

현재 상용화된 서비스로는 Drowsiness Detection을 통해 졸음을 판단한 경우 알림음을 내는 것에 불과하다. 그러나 우리가 개발한 운전자 상태 인식을 통한 추천서비스는 운전자의 졸음 운전 지수와 주행 집중도를 0 ~ 100 사이 값으로 판단하고 운전자에게 제공한다. 따라서 주행자는 졸음 운전지수와 주행 집중도 수치를 통해 자신의 졸음 운전정도와 주행 집중도를 실시간으로 알 수 있고, 또한 주행 집중도 수치가 50 이하라면 TTS기능을 이용하여 운전자에게 주의를 주며 운전자가 안전한 주행을 할 수 있도록 한다.

만약 사용자가 퇴근길에 피곤하다면 추천서비스를 이용하여 보다 편리하게 스마트홈 IoT 모듈을 제어할 수 있기 때문에, 사용자가 집으로 돌아가 직접적으로 모듈을 제어할 때 발생하는 불필요한 시간을 단축시킬 수 있다. 그리하여 바쁜 현대인들이 시간을 효율적으로 쓰는데에 기여할 것이다.

○ Car-Smart Home 양방향 정보 전달

기존 카투홈(Car to Home) 서비스는 단순히 차량에서 집에 있는 IoT 기기를 제어하는 것에 그친다. 하지만 실시간으로 수집되는 스마트홈의 정보를 활용한다면 사용자에게 더욱 다양한 유용하고 편리한 서비스를 편리하게 제공할 수 있다.

자체 개발한 스마트 냉장고 모듈을 살펴보면 사용자는 냉장고에 있는 식재료들을 이용한 추천 메뉴와 해당 메뉴에 추가적으로 필요한 식재료 정보를 제공받을 수 있다. 또한, 사용자는 해당 메뉴의 레시피를 차량의 대시보드에서 확인가능하다. 하지만 차량에 있는 사용자의 특성상 대시보드에 집중할 수 없는 상황이기 때문에 이를 가정에 있는 냉장고로 보내는 양방향 정보전달이 가능하다. 이를 통해 사용자가 차량과 스마트홈에서 유기적인 서비스를 제공받을 수 있고 집에 도착하기 전 미리 필요한 정보를 습득함으로써 시간을 효율적으로 활용할 수 있다.

○ 1인 가구 치안 강화 및 유용한 기능 제공

1인 가구가 꾸준하게 증가하고 있고, 경제 활동을 하는 동안 집안 내부에 사람이 없는 것은 1인 가구의 어쩔 수 없는 부분이다. 이에 따라 1인 가구의 치안 강화가 필요해지는 시점에서 CCTV 모듈을 통해 실시간 영상을 전달받음으로써 안전성을 보장하는 것은 물론 일상생활에서 흔히 발생하는 택배 배달원이나 검침원과 같은 부재중 외부인이 자택 방문을 한 상황에 운전자는 CCTV 모듈을 통해 파악된 외부인의 모습을 사진으로 남기고 차량으로 전달한다. 운전자는 이를 통해 외부인의 존재를 확인하고 외부인은 사용자에게 음성 메시지를 남겨 방문객에게 생기는 의사전달의 제한을 해소할 수 있다.

이러한 기능은 1인 가구의 안정성을 보장함은 물론 일상생활 간에 유용한 기능을 통해 사용자에게 편의를 제공한다.

□ 개발 일정

No	내용	2021年															
		6月				7月				8月				9月			
1	webOS 환경설정 및 기본 개념 정리																
2	webOS 기기 및 IoT 모듈 UI 제작																
3	YOLOv5 기반 모델 및 얼굴인식 구현																
4	IoT제품 설계 및 아두이노 실습																
5	LS2 API 활용한 서비스 구현																
6	Jetson Nano 기반 모듈(CCTV, 생체인식)																
7	아두이노 기반 모듈(전기장판, 욕조)																
8	Raspberry Pi 기반 모듈(Refrigerator)																
9	DB8을 활용한 데이터베이스 구축																
10	서비스를 포함한 웹 앱 패키징																
11	시험 평가 및 테스트																

□ 팀 업무 분장

No	구분	성명	참여인원의 업무 분장
1	팀장	송지민	Jetson nano 기반 IoT모듈(CCTV,운전자 인식 모듈) 제작
2	팀원	서현명	DataBase API(DB8),Node.js를 통한 Backend 설계
3	팀원	신지연	HTML5,CSS3,JS를 통한 Frontend(UI/UX) 구현
4	팀원	온민권	Raspberry Pi 기반 IoT모듈(냉장고 모듈) 제작, Face ID 구현
5	팀원	정진우	Arduino 기반 IoT모듈(욕조,전기장판 모듈) 제작