제18회 임베디드SW경진대회 개발완료보고서

[webOS 기반 차량용 인포테인먼트 솔루션]

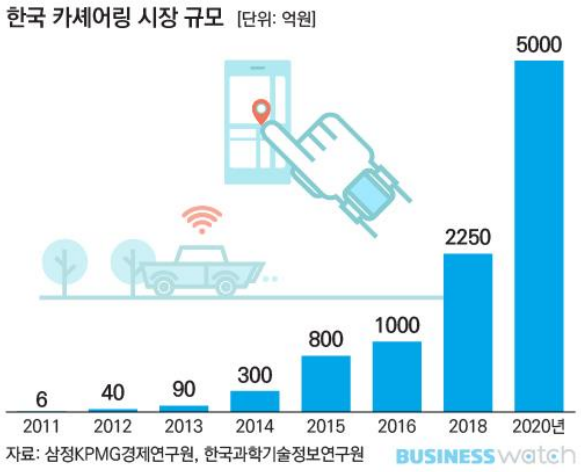
□ 개발 요약

팀 명	RORA
	
작품명	J.A.R.V.I.S(Just A Rather in-Vehicle Infortainment System)
작품설명 (요약)	webOS를 탑재한 커넥티드 카 기반의 차량 공유 모빌리티 서비스다. 대표적인 기능은 차주가 커넥티드 카를 사용하지 않는 시간 동안 차량이 필요한 사람에게 커넥티드 카를 공유할 수 있다. webOS를 통해 어떤 플랫폼에서든 쉽게 차량 공유 서비스 이용이 가능하며, webOS가 탑재된 디바이스간 뛰어난 연동성을 통해 커넥티드 카에서 webOS 플랫폼을 비롯한 다양한 인포테인먼트 서비스 이용이 가능하다.
소스코드	https://github.com/2020ESWContest-webOS-4004
시연동영상	https://youtu.be/abizcG3NliE

□ 개발 개요

○ 개발 작품 개요

경제 패러다임이 소유에서 공유로 이동하고 있는 현재 공유 모빌리티 시장은 지속해서 발전하고 있다. 그중 대표적으로 차를 공유하는 카 셰어링이 있다. 많은 사람이 카 셰어링 서비스를 이용하고 있으며 시장 규모도 커지고 있다.



[그림 1] 대한민국 카셰어링 시장 규모

카 셰어링 시장의 성장으로 인해 대기업부터 스타트업까지 많은 기업들이 카 셰어링 사업에 주목하고 있다. 그 예로 국내 카 셰어링 대표 기업인 쏘카는 SK와 베인캐피탈에서 650억원을 투자받았고, 그린카는 KT렌탈에 인수, KT렌탈이 롯데그룹에 매각되면서 롯데렌탈의 자회사가 되었다. 그리고 최근에는 자동차와 거리가 멀었던 대기업들의 카 셰어 사업 진출도 활발해지고 있다.

[그림 2] 카 셰어링 사업에 뛰어드는 대기업

특히 주목할 점은 완성차 업체들도 카 셰어 사업에 적극 나서고 있다는 점이다. 대표적으로 독일의 BMW와 벤츠가 있다. 이 기업들의 흥미로운 점은 모두 유명 카 셰어 업체와 파트너십을 맺어 자신의 차량을 공급함과 동시에 사업 역량을 키우고 있다는 점이다. 국내의 현대자동차도 현대캐피탈과 함께 카 셰어링 서비스를 런칭해 사업에 뛰어들었고, 일본도 비슷한 상황이다. 또한 현재의 카 셰어링 서비스는 B2B 산업으로써 카 셰어 업체 소유의 차량을 개인이 제공받는 구조다. 그로 인한 높은 비용과 청결 상태 등의 문제가 있으며 완성차 업체 입장에선 카 셰어링 서비스 개발 시 제조사별 독자적인 플랫폼 기반의 카 셰어링 서비스를 개발하게 되

어 파트너십 카 셰어 업체와 호환은 가능하지만 타사 카 셰어 서비스와 호환이 되지 않아 추후에 사업이 확장하게 되면 타 업체와의 카 셰어링 서비스 연동 시 추가 개발 비용이 발생할 수 있다.

하지만 커넥티드 카에 webOS를 탑재하고, 카 셰어링 서비스를 webOS 플랫폼 기반 앱으로 개발 및 배포한다면, 추후 차량 OS에 webOS 점유율이 높아졌을 때 완성차 업체들과 카 셰어링 업체는 webOS라는 통일된 플랫폼을 통해 카 셰어링 서비스 개발비용을 줄이고 호환성을 높일 수 있는 장점을 가질 수 있게 된다. webOS는 리눅스 커널 기반의 임베디드 운영체제로, 5G, 자율주행, 인공지능, IoT와 연동 등 최신기술을 적용할 수 있어 차세대 커넥티드 카 운영체제로 적합하다.

따라서 우리는 webOS를 탑재한 완성차 업체 입장에서 webOS 앱 및 서비스 개발자가 되어 webOS가 탑재된 커넥티드 카에서 차주가 커넥티드 카를 공유하는 C2C 공유 모빌리티 서비스를 만들고자 한다.

이 서비스는 차주가 일하고 있거나 휴가를 떠나 차를 사용하지 않는 잉여 시간동안 차량이 필요한 사람에게 대여료를 받고 차를 공유해 수익을 창출할 수 있다. 이를 통해 차주는 차량 할부금, 리스 금액 등 차량 유지비의 금전적 부담을 감소할 수 있다. 또한, 차량 대여자는 서비스 이용 과정 중 카 셰어링 업체의 개입이 없기 때문에 저렴한 비용으로 공유 서비스를 이용할 수 있다. 그리고 무분별한 자동차 구매를 감소시켜 환경오염 문제를 개선하는 등 개인과 사회에 이롭고 편리한 서비스로 자리잡을 수 있을 것이다.

○ 개발 목표

JARVIS는 다음 3가지 테마를 중심으로 개발 목표를 세웠다.

1. 편의성
   - 인증된 사용자라면 JARVIS를 통해 커넥티드 카에서 다양한 webOS 기반 플랫폼(스마트 홈, 스마트 웨어러블 등 IoT 장비)을 손쉽게 제어할 수 있다. 반대로 스마트 홈과 같은 webOS 기반 플랫폼을 통해 커넥티드 카를 예약하는 등 다양한 편의 기능을 제공한다.

2. 경제성
   - JARVIS는 잉여 시간을 활용해 대여자에게 카 셰어링을 하여 커넥티드 카의 가용성을 최대한으로 끌어 올려 수익 창출이 가능해진다.

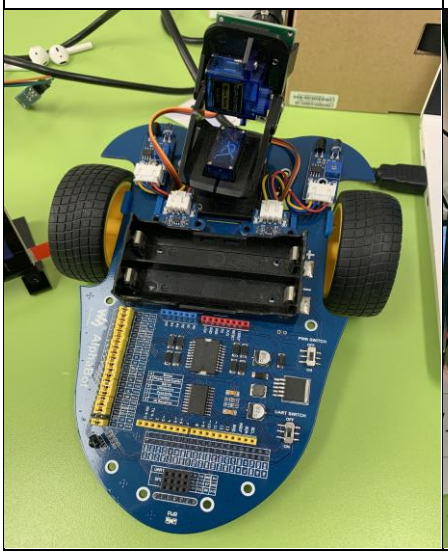
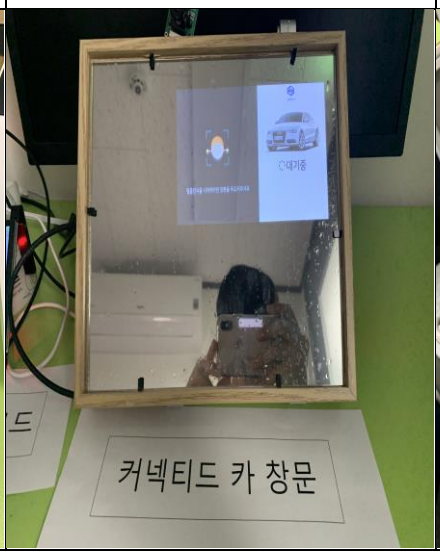
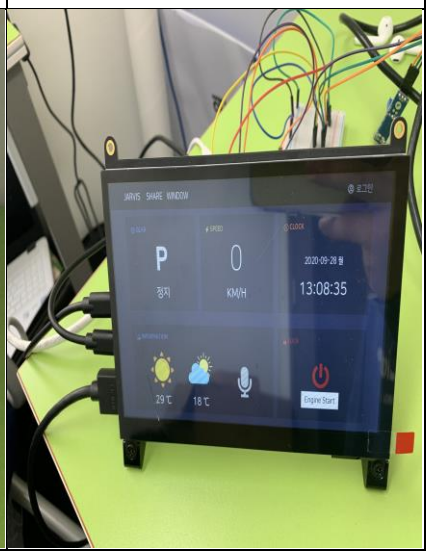
3. 안전성
   - 사고 위험에 노출될 수밖에 없는 자동차의 특성상 커넥티드 카의 안전성을 보장하기 위한 기능을 개발했다. 첫 번째는 응급 구조 서비스다. 인적이 드문 장소에서 갑작스런 대형 교통사고가 발생해 운전자의 의식이 없을 때 webOS에 탑재된 VPA(가상비서)가 사고의 상태를 분석, 운전자에게 대화를 시도하고, 운전자 응답이 없을 경우 자동으로 현재 위치, 운전자 혈액형 정보 등을 구조대로 신고해 빠른 구조를 도와준다. 또한 차량 도난 시도가 감지될 경우 자동으로 경찰과 차주에게 연락을 취해 도난을 사전에 방지할 수 있다.

□ 개발 환경 설명

○ Hardware 구성

프로젝트를 통해 제작되는 하드웨어는 커넥티드 카 프로토 타입(RC카), 사용자 인증을 위한 운전석 쪽에 위치한 스마트 윈도우, 차량 대시보드 등 총 세 가지다.

1. Hardware 실물

커넥티드 카 프로토타입	커넥티드 카 스마트 윈도우	커넥티드 카 대시보드
		

[표 1] Hardware 실물

1.1. 커넥티드 카 프로토 타입(RC카)

- 재료 : AlphaBot - Pi Acce Pack, 라즈베리파이4B, IR센서, 초음파센서
- 설명 : AlphaBot으로 이루어진 커넥티드 카 프로토 타입은 webOS가 설치된 라즈베리파이를 탑재해 webOS를 통해 커넥티드 카의 모든 움직임을 제어할 수 있다.

1.2. 커넥티드 카 스마트 윈도우

- 재료 : 7-inch Display, 라즈베리파이 4B, 카메라, 진동센서(SW-420), 브레드보드, 유리 액자, 하프미러 필름, 합판 4개, 받침핀 3개, 나사 10개, 검정 종이 2장
- 설명 : 유리에 하프미러 필름을 붙인 스마트 미러 형식이며, 유리창 뒤편에 7-inch 디스플레이, 라즈베리파이, 브레드보드 및 센서를 수납할 수 있는 공간을 만들었다. 마찬가지로 webOS가 설치된 라즈베리파이를 탑재했으며, 커넥티드 카와 연결되어 사용자 인증 시 활용된다.

  스마트 윈도우의 사용방법은 하단 [그림 3]과 같다. 별도의 차 키 없이 사용자가 커넥티드 카에 접근해 스마트 윈도우에 노크를 2회 하면 스마트 윈도우에 장착된 센서가 노크를 감지하고 카메라를 동작 시켜 사용자의 얼굴을 인식한다. 그리고 인증과정 및 결과가 스마트 윈도우 디스플레이에 나타나며 커넥티드 카에 권한이 있는 사람이라면 차량 문을 개방해준다.

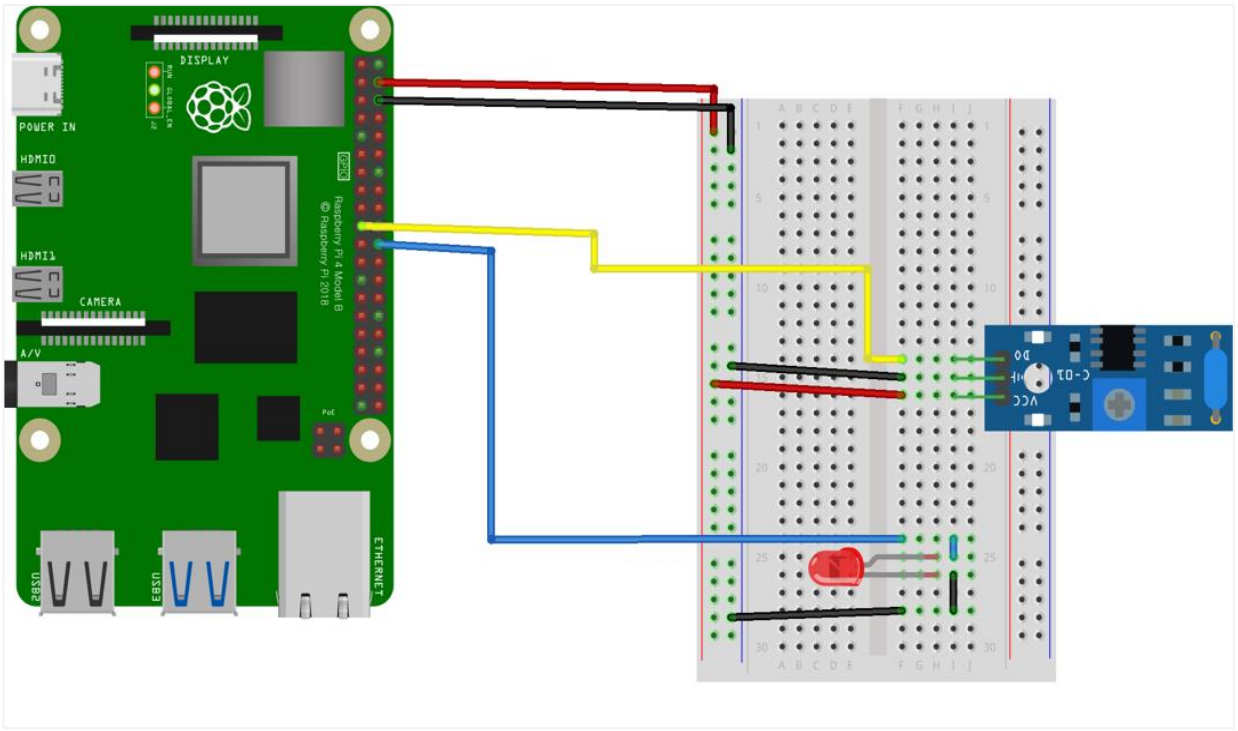
[그림 3] webOS 스마트 윈도우 생체인식 로그인 방법

1.3. 커넥티드 카 대시보드

- 재료 : 7-inch 디스플레이
- 커넥티드 카에 탑승하면 볼 수 있는 터치형 차량 대시보드다. 이 대시보드를 활용해 webOS 커넥티드 카를 쉽게 제어할 수 있다. 차량 기어, 속도, 시간, 날씨 등 다양한 정보가 나타나며 VPA(가상비서), 시동 등의 기능을 사용할 수 있다. 그리고 대시보드를 통해 쉽게 공유 모빌리티 서비스를 이용할 수 있다.

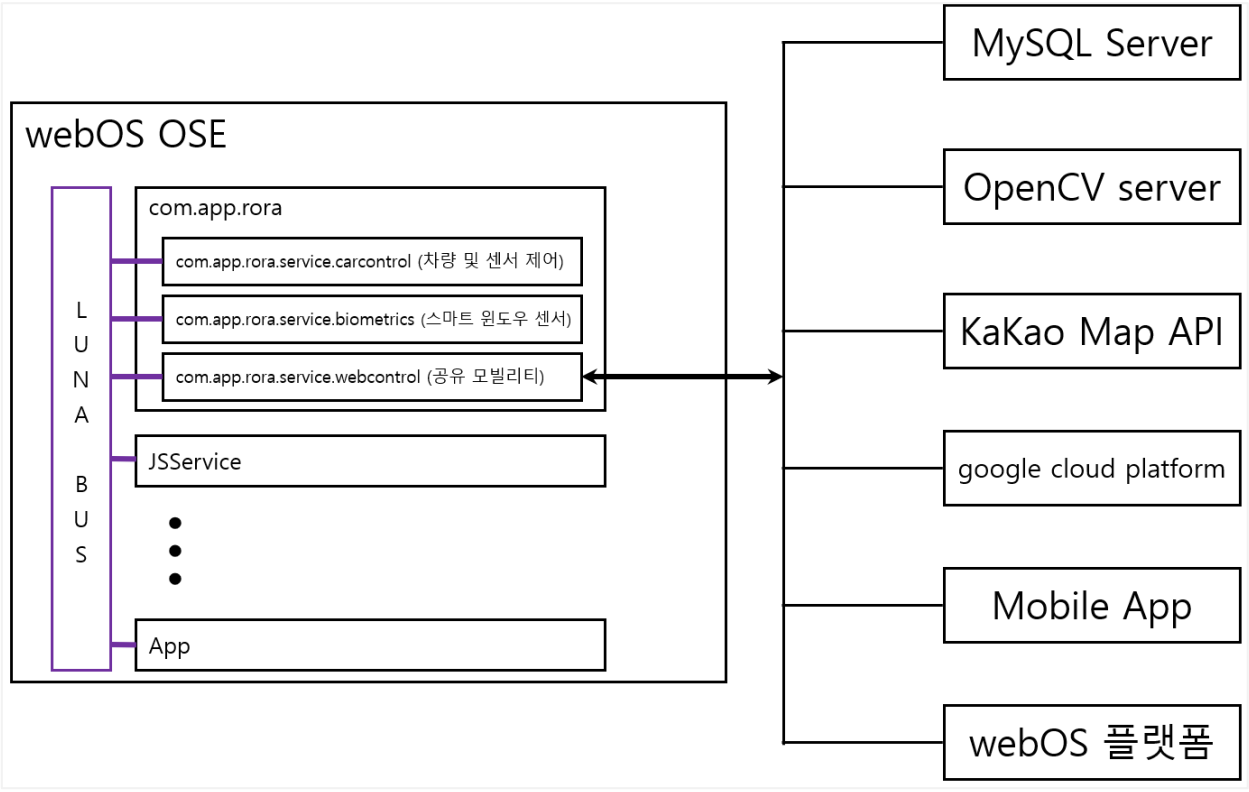
2. Hardware 결선도

2.1. 스마트 미러 결선도



[그림 4] 스마트 미러 진동센서 결선도

○ **Software 구성**



[그림 5] JARVIS software 구성도

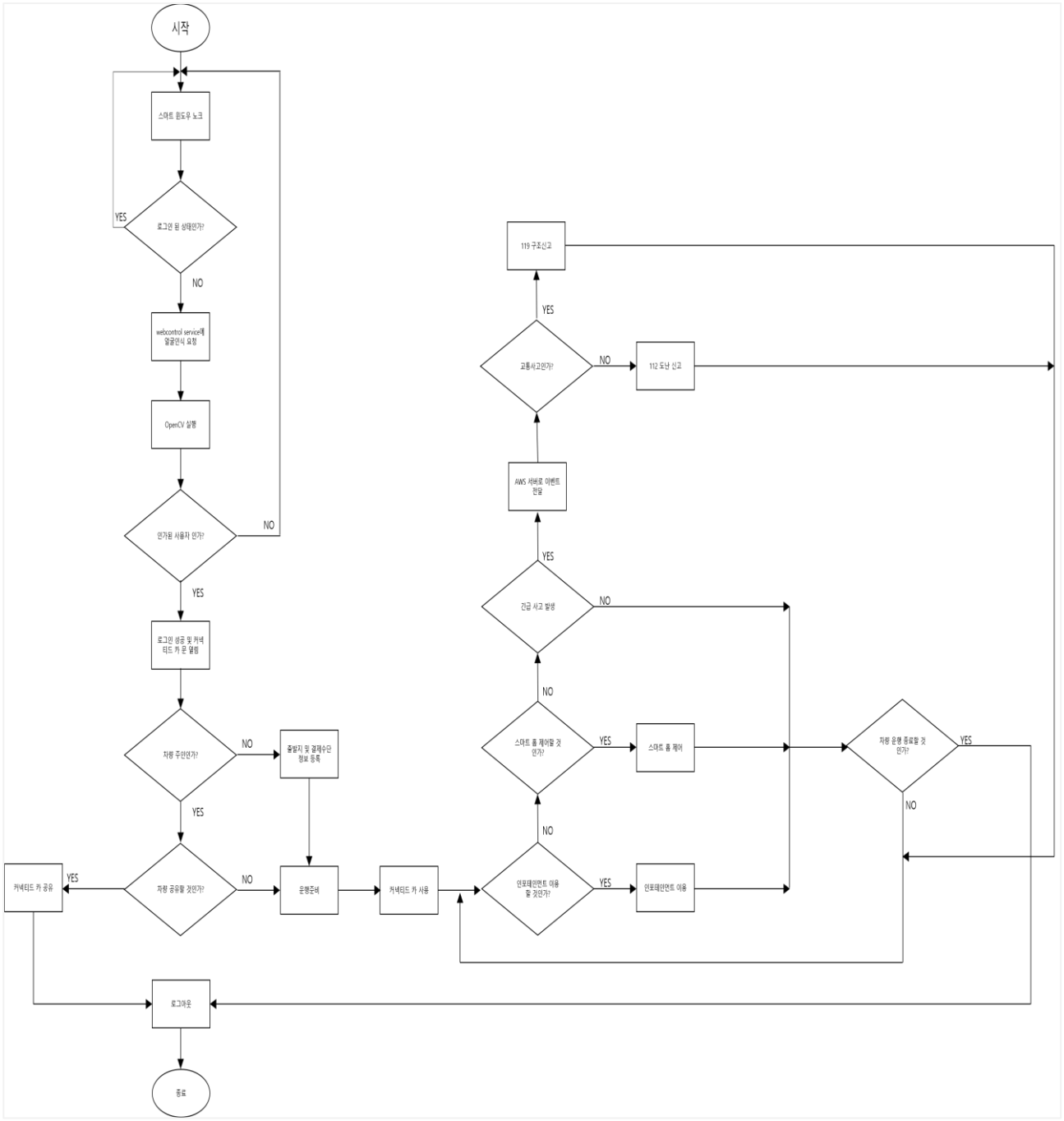
[그림 5]를 보면 JARVIS는 webOS OSE에 com.app.rora 라는 ID로 설치가 된다. com.app.rora 내부에는 JARVIS를 구성하는 총 세 개의 JSService(백그라운드 서비스)가 존재한다. 세 개의 서비스는 LUNA BUS를 통해 서로 필요한 기능 또는 정보를 교환할 수 있다. 차량과 차량에 부착된 센서를 제어하는 com.app.rora.service.carcontrol과 스마트 윈도우의 센서를 제어하는 com.app.rora.service.biometrics가 있다.

두 서비스는 C언어로 개발한 Native 서비스다. 주로 차량 하드웨어를 제어하기 위한 서비스로 라즈베리파이의 GPIO핀을 통해 차량 움직임 및 스마트 윈도우의 센서를 감지할 수 있다. 감지된 데이터는 LUNA BUS를 통해 com.app.rora.service.webcontrol로 전달된다.

webcontrol은 커넥티드 카 공유 모빌리티 서비스를 구현한 JSService다. JSService는 Node.js로 개발되는 특성을 이용해 웹 서버로 동작하도록 개발했다. 웹 서버로 동작하기 때문에 다양한 플랫폼에서 공유 모빌리티 서비스에 접근 및 사용이 가능하다. 그리고 외부에 위치한 DB 서버, 얼굴인식을 위한 OpenCV 서버, 그리고 공유 모빌리티에 지도 데이터를 적용하기 위해 카카오 맵 API를 사용한다. 그리고 webOS 플랫폼(webOS TV, webOS Signage, Smart Watch)와 연동된다.

○ **Software 설계도**

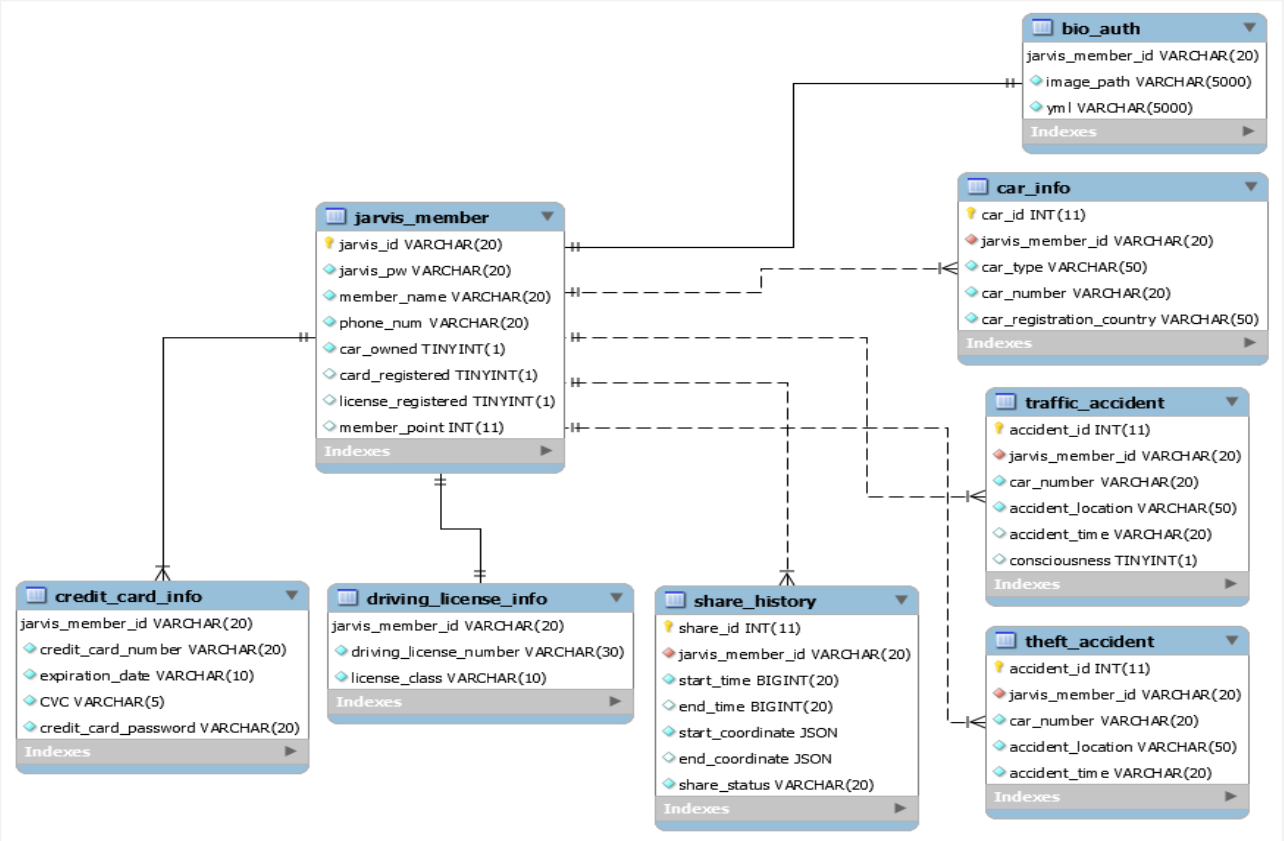
- JARVIS software 흐름도 (Life Cycle)



[그림 6] JARVIS 서비스 흐름도

사용자가 스마트 윈도우에 노크를 하면서 서비스 흐름이 진행된다. 노크 후 얼굴인식을 통해 사용자가 커넥티드 카의 차주인지, 대여자인지 판별하고 사용자 레벨에 맞는 흐름으로 분기해 진행한다. 인증 후 사용자가 커넥티드 카를 사용하면서 다양한 인포테인먼트, 스마트홈 서비스 등을 이용하게 되며 이후 커넥티드 카 사용을 마칠 경우 JARVIS에서 로그아웃 한 후 서비스가 종료되면서 Life Cycle이 끝나게 된다.

- JARVIS 데이터베이스 구조



[그림 7] JARVIS 데이터베이스 구조

- 데이터베이스 구조 설명

1. jarvis_member : JARVIS 사용자의 정보를 저장하는 테이블
2. bio_auth : 사용자의 생체 인증을 위한 정보를 저장하는 테이블
3. car_info : 차주가 커넥티드 카 공유를 위해 차량의 정보를 저장하는 테이블
4. credit_card_info : 공유비용 결제에 필요한 결제 카드 정보를 저장하는 테이블
5. driving_license : 사용자의 면허정보를 등록하는 테이블
6. traffic_accident : 커넥티드 카 이용 중 발생한 교통사고의 정보를 저장하는 테이블
7. theft_accident : 커넥티드 카 도난 기록이 저장되는 테이블
8. share_history : 커넥티드 카를 빌려주고 대여한 기록을 저장하는 테이블

○ Software 기능

- **스마트 윈도우 사용자 인증**

커넥티드 카 이용 전, 사용자가 차량 이용이 가능한 사람인지 확인하는 절차를 거쳐야 한다. 이를 확인하기 위해 스마트 윈도우에 적용된 센서(충격센서)에 노크신호가 들어오면 카메라가 동작해 노크를 한 사람의 얼굴을 인식하도록 한다. 윈도우 노크는 백그라운드에 thread를 생성해 센서를 감지할 수 있도록 Native Service로 개발했다. 노크가 감지되면 카메라에 얼굴 인식 요청을 한다. 카메라가 얼굴인식을 한 후 결과 값을 JSService인 com.app.rora.service.webcontrol로 보내주고 인증에 성공하면 문을 개방하고

로그인 처리한다.

- **차주 및 대여자 구분**

  인증 후 커넥티드 카에 탑승한 사용자는 차주, 대여자 두 경우로 나뉘게 된다. JARVIS 회원은 가입 시 차량 소유 여부를 입력하게 되며 이 때 입력한 차량소유 정보를 통해 차주와 대여자를 구분하며, 차주는 대시보드에 '빌려주기' 버튼이 활성화되고, 대여자는 공유 페이지에 접근 가능하다.

- **차주 커넥티드 카 공유**

  차주는 탑승 후 차량 이용을 하거나, 공유모드로 전환해 일정 시간동안 커넥티드 카를 공유할 수 있다. 대시보드를 통해 '빌려주기' 버튼을 눌러 공유가 가능하며 공유하는 동안 공유비를 받고, 해당 시간동안 공유 받은 사용자 이외에 차량을 이용할 수 없다.

- **사용자의 커넥티드 카 대여**

  대여자가 대시보드의 'SHARE' 버튼을 누르면 카카오맵 API를 통해 개발한 지도와 서비스 버튼들이 화면에 나온다. 여기서 출발지, 대여조건을 확인하고 면허 정보와 결제 카드 정보를 등록한다. 등록이 완료되면 '대여하기' 버튼을 누르고 대여 정보를 최종적으로 확인하는 페이지가 나온다. 다음 페이지의 '대여 시작' 버튼을 누르면 대여가 시작된다. 이 때 사용자가 현재 차량을 대여 중인 상태인 것을 기록하기 위해 DB의 share_history 테이블을 통해 관리하게 된다. 차량 이용을 종료할 경우 결제 정보 확인 페이지에서 대여 관련 정보를 확인하고 포인트 사용 여부를 결정한 다음 결제하기 버튼을 통해 포인트 사용 및 비용 결제가 완료된다. 결제 성공 메세지와 영수증이 출력되면서 대여가 종료된다.

- **커넥티드 카 원격 제어 모듈**

  운전석 출입이 어려운 경우 커넥티드 카를 원격에서 조종할 필요가 있다. 이 때 원격 제어 모듈을 이용해서 탑승 가능한 구역까지 커넥티드 카를 원격으로 이동시킬 수 있다. 원격 모듈은 차량 제어를 위한 Native Service(carcontrol)과 웹서버 형식으로 제작된 JSService(webcontrol)을 이용해 웹을 통해 Native Service를 제어할 수 있다. 브라우저 또는 앱에서 JSService에 접속해 컨트롤러 UI를 통해 커넥티드 카를 원격에서 제어할 수 있다.

- **JARVIS 모바일 앱**

  JARVIS의 기능을 모바일을 통해 이용 가능하도록 개발한 하이브리드 모바일 앱이다. 차주, 대여자가 모바일을 통해 공유 정보 등 현재 위치와 차량 정보, 사고 이력 등을 조회할 수 있으며 커넥티드 카 원격 제어 모듈도 제공한다.

- **긴급 구조 서비스**

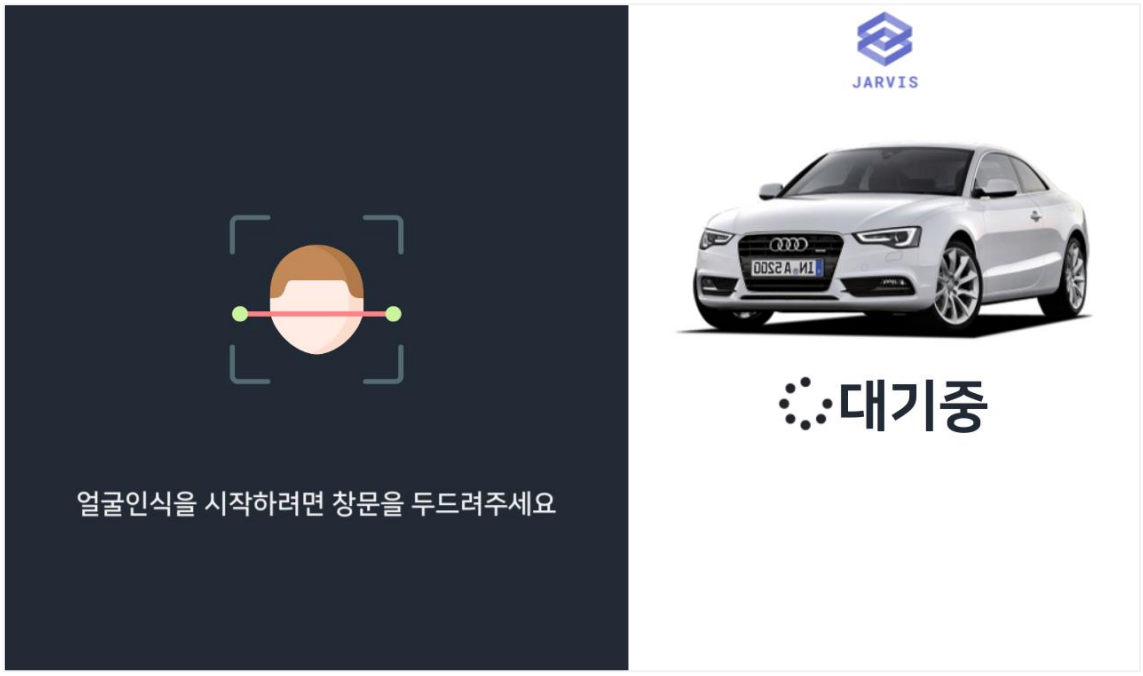
  차량에 부착된 센서들을 통해 실시간으로 충돌 여부를 감지한다. 운행 중 충돌사고가 발생하면 센서를 관리하는 Native Service(carcontrol)에서 JSService(webcontrol)에 사고 데이터를 넘기면 webcontrol은 사용자 이름, 사고 위치, 사고 시간, 사용자 의식 등 관련 데이터를 FCM 서버에 넘기고, FCM 서버는 앱의 키 값 확인 후 해당 스마트폰에 Notificaton을 push하고 응급 구조 기관에 SMS를 통해 자동으로 구조요청을 진행한다.

- **스마트홈**

  커넥티드 카와 연동시켜 커넥티드 카에서 스마트 홈의 기능 이용이 가능하다.

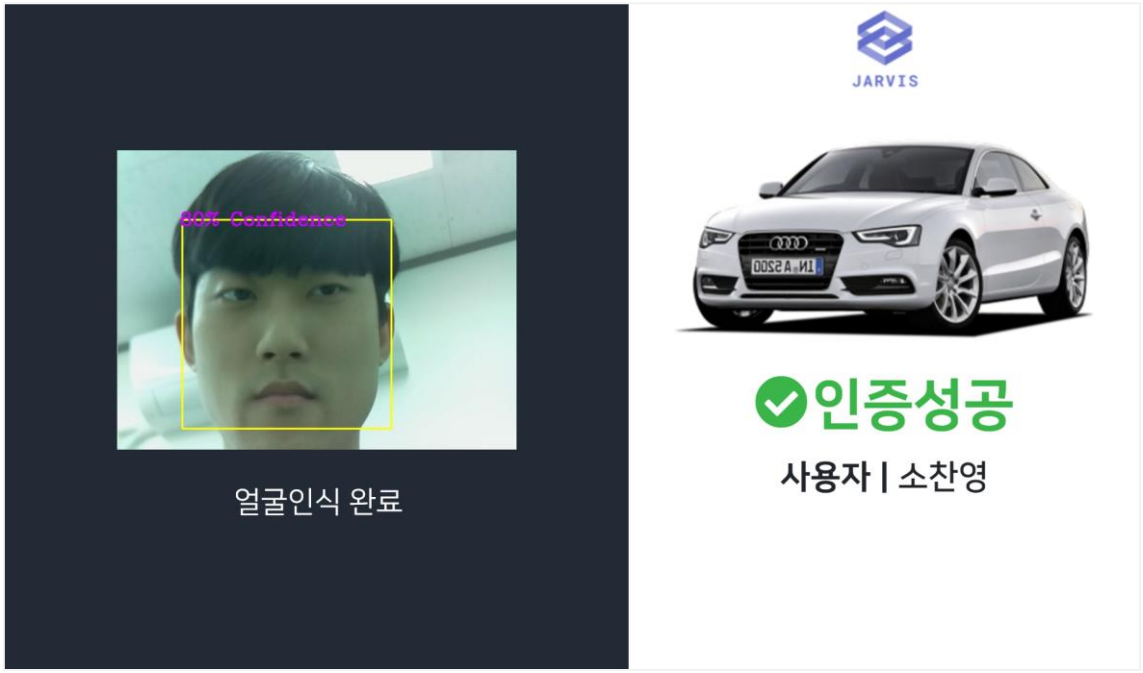
○ **UI 사용법**

1. 사용자 인증과정



[그림 8] webOS 커넥티드 카 스마트 윈도우 대기화면

커넥티드 카의 스마트 윈도우 디스플레이에 출력 되는 대기화면이다. 차량 이용을 위해 스마트 윈도우를 두 번 노크한다.



[그림 9] 스마트 윈도우를 통한 얼굴인식

스마트 윈도우를 노크하면 센서를 감지해 스마트 윈도우에 부착된 카메라가 동작한다. 사용자가 스마트 윈도우의 화면을 3초간 바라보면 얼굴을 판독해 차량에 탑승할 수 있는 사용자인지 구별하고, 권한이 있는 사용자일 경우 차량 문을 개방하고 서비스에 로그인 시켜준다.

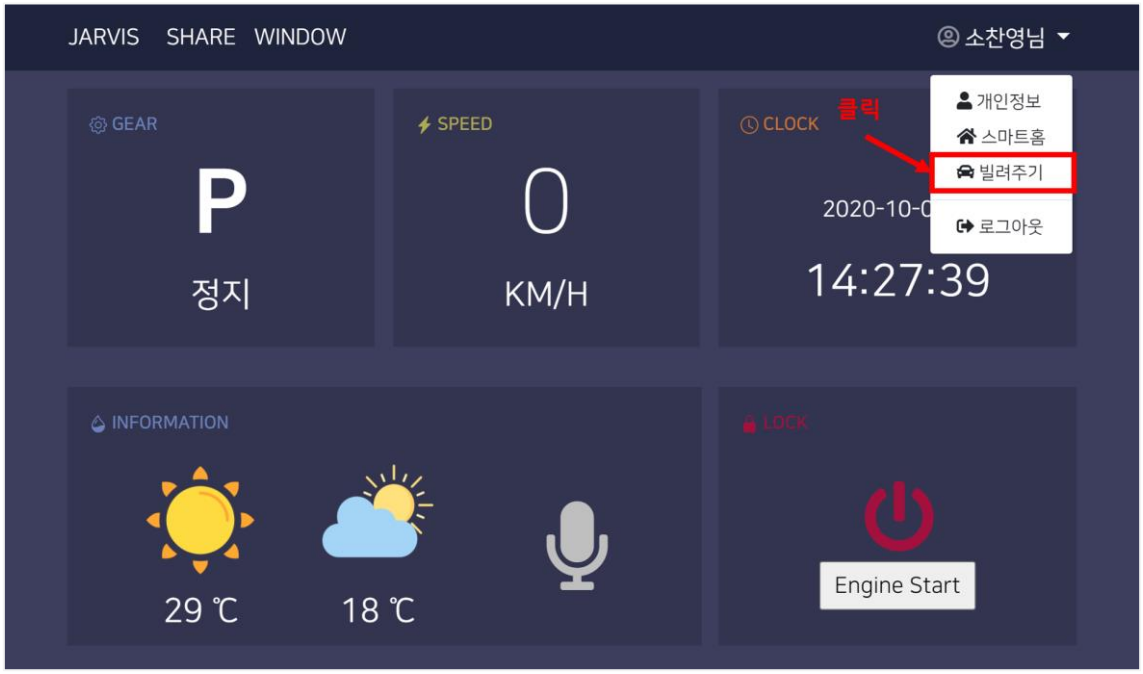
[그림 10] webOS 커넥티드 카 대시보드

커넥티드 카에 탑승하면 나오는 차량용 대시보드로, webOS 커넥티드 카의 컨트롤 타워라고 할 수 있다. 대시보드 구성 설명은 아래와 같다.

① 로그인 한 차량 공유, 스마트 홈 및 개인 페이지에 접속 가능한 버튼 내장
② 차량의 현재 기어 상태를 나타내는 위치로 전진, 우회전, 좌회전, 후진, 정지가 있다.
③ 차량의 속도를 나타낸다.
④ 현재 시간을 나타낸다.
⑤ 차량을 공유 받고자 할 때 접속하는 차량 공유 페이지
⑥ 스마트 윈도우에 접속
⑦ 현재 날씨 출력
⑧ 구글 어시스턴트 VPA(음성 비서)를 실행하고 끄는 버튼
⑨ 차량 시동 버튼

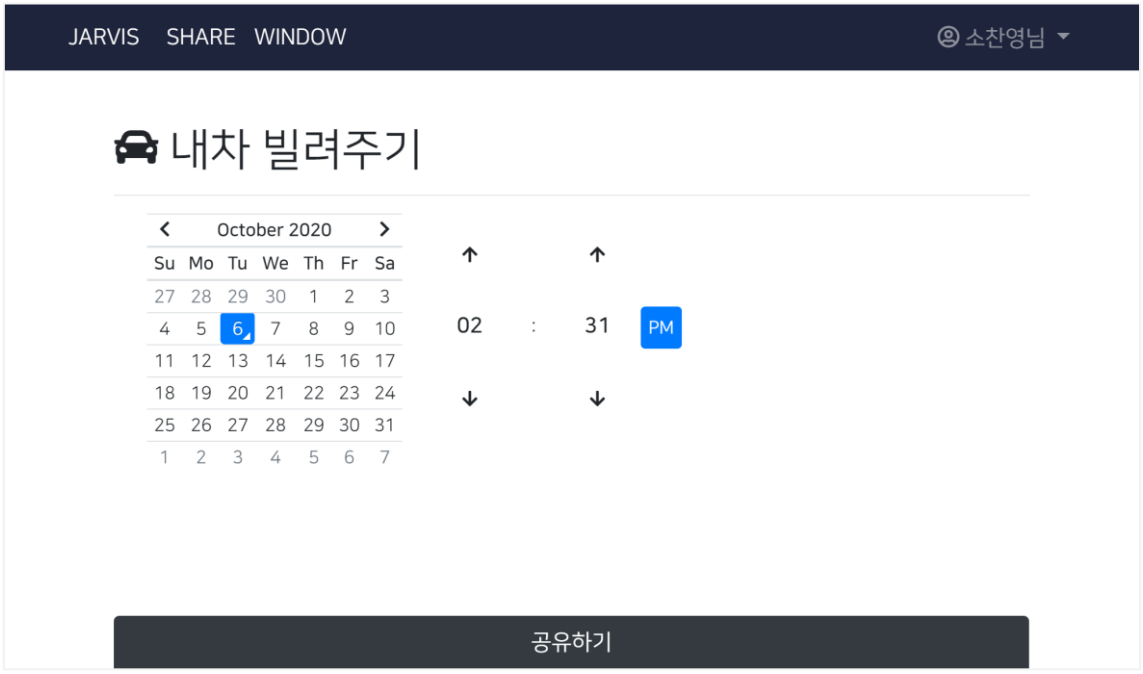
얼굴인식 후 차량에 탑승한 사람은 커넥티드 카의 차주일 수도 있고, 대여를 원하는 대여자일 수 있다. 지금부터 차주가 커넥티드 카를 공유할 경우와 대여자가 커넥티드 카를 대여할 경우의 사용법으로 나눠서 설명한다.

2. 차주 차량 공유



[그림 11] 커넥티드 카 공유 하기 (1)

차주가 차량을 공유할 경우 우측 상단의 자신의 이름을 클릭하면 dropdown 형식으로 메뉴가 나온다. 이 때 '**빌려주기**' 버튼을 클릭한다.



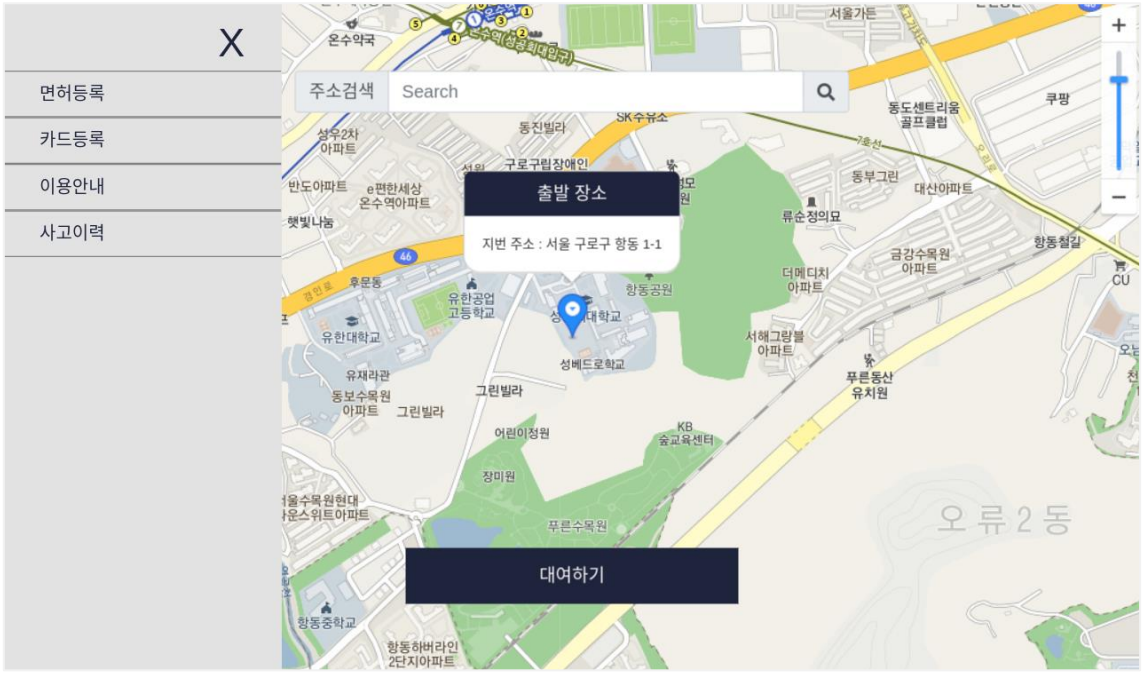
[그림 12] 커넥티드 카 공유 하기 (2)

내 차를 다른 사람에게 공유할 시간을 정한 후 '**공유하기**' 버튼을 클릭하면 해당 차는 지정한 시간부터 JARVIS의 공유 서비스를 통해 필요한 사람에게 공유하게 된다.

3. 대여자 차량 대여

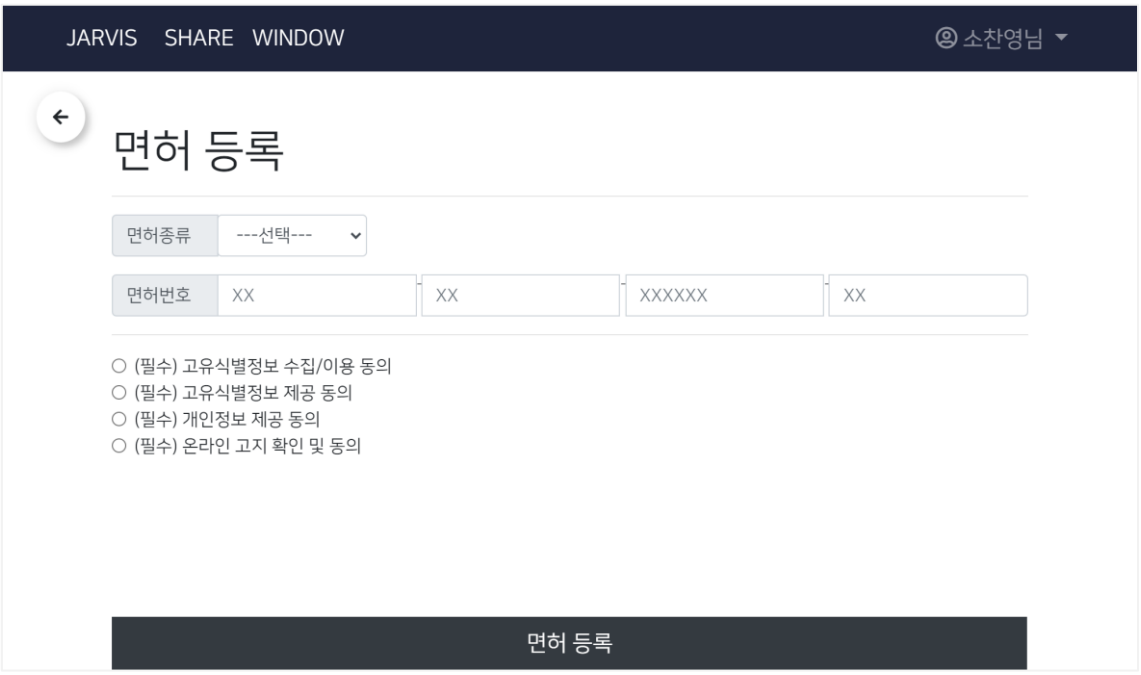
[그림 13] 커넥티드 카 대여 페이지

[그림 13]은 차량을 공유 받으려는 사용자가 인증 과정을 거친 후 커넥티드 카에 탑승해 SHARE 페이지에 접속하면 나오는 화면이다. 여기서 현재 출발지 위치를 지정한 후 왼쪽 상단에 위치한 동그란 버튼을 누른다.



[그림 14] 대여 페이지 사이드 메뉴

차량을 대여하기 전 왼쪽 메뉴를 통해 면허 및 카드를 등록해야 한다.

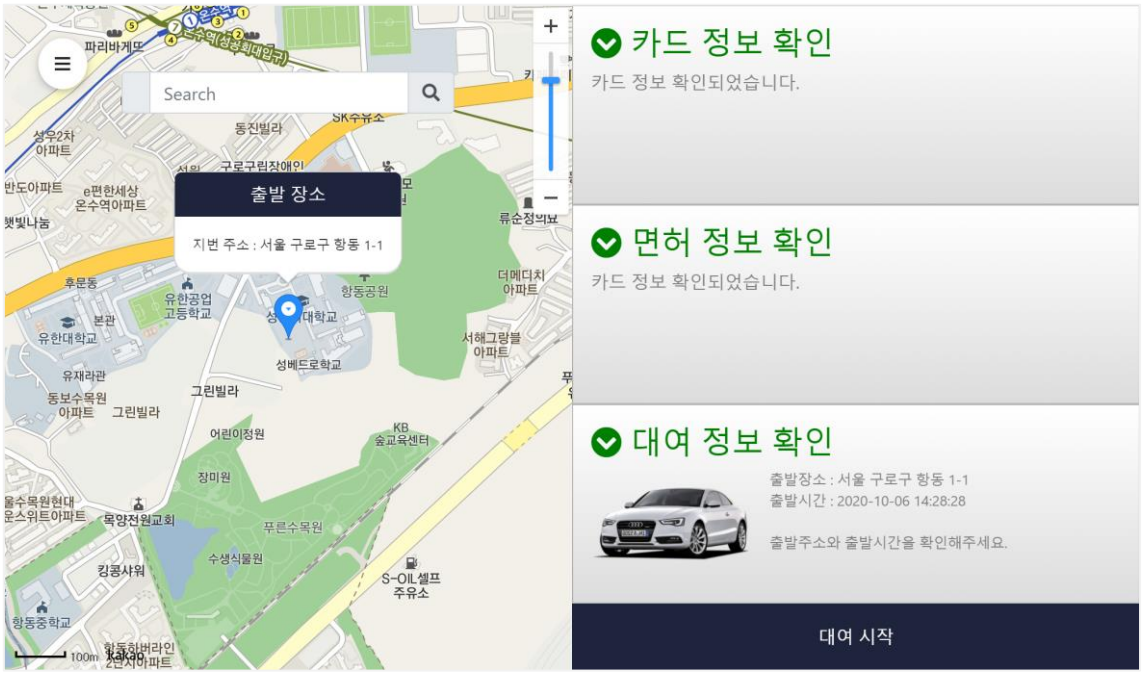


[그림 15] 사용자 면허 등록

먼저 면허 등록 페이지다. 이용자의 면허정보를 등록한다.

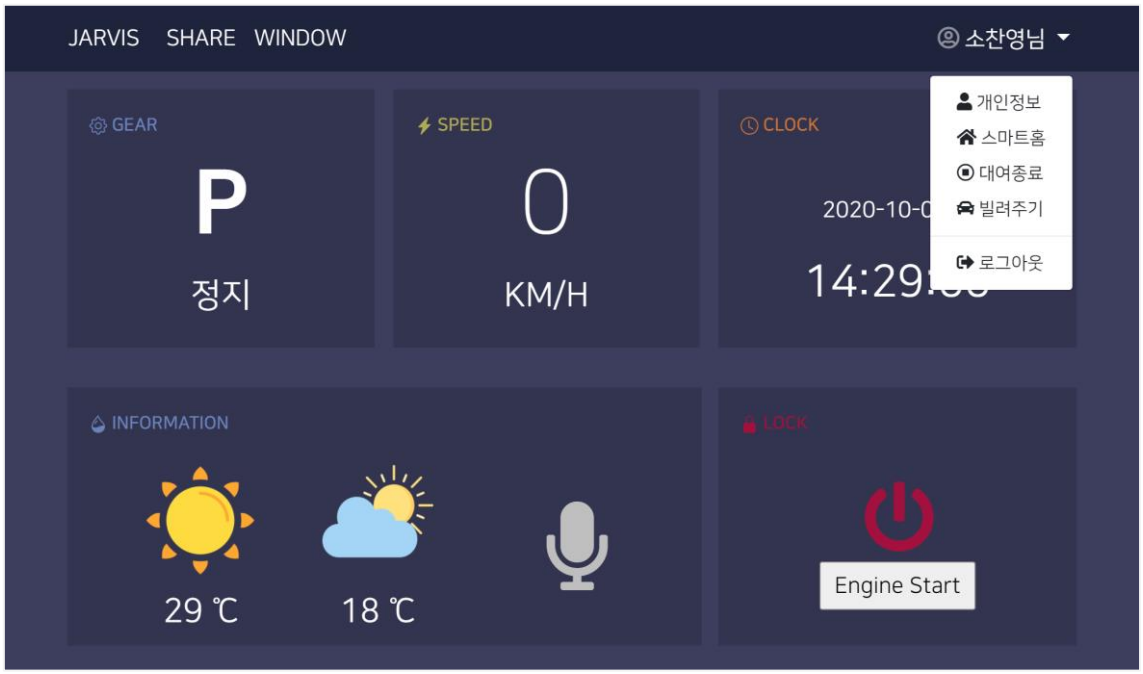
[그림 16] 사용자 결제 카드 등록

다음은 카드 등록 페이지다. 공유 비용을 결제할 카드 정보를 등록한다.



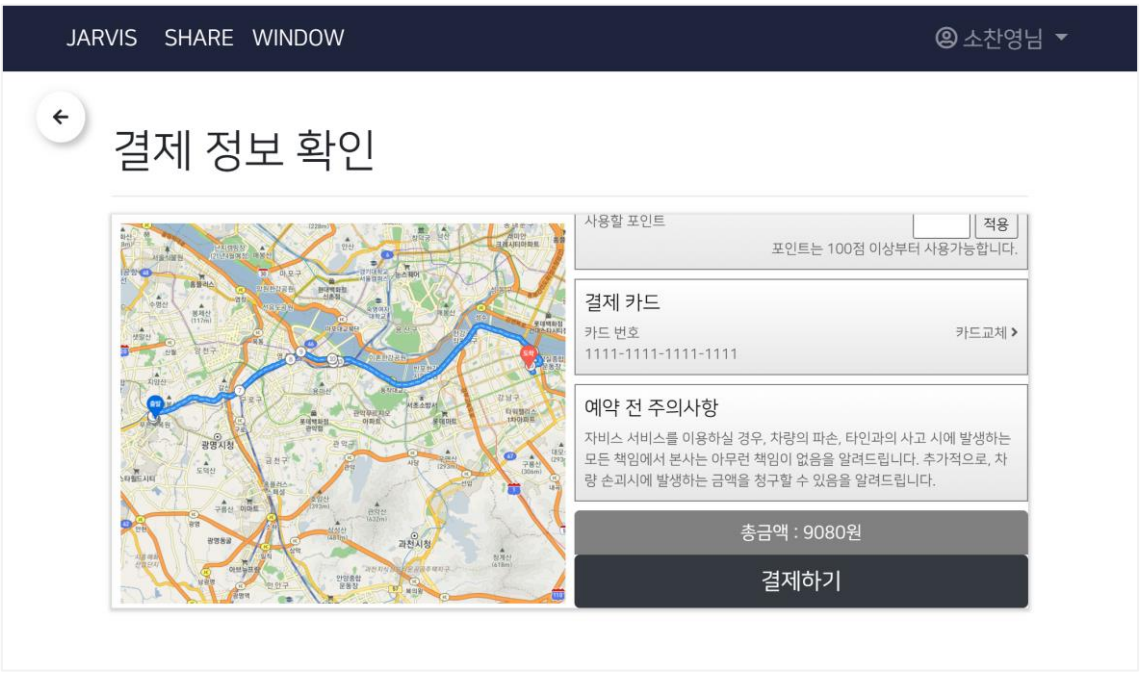
[그림 17] 대여 정보 최종 확인 페이지

면허와 카드 등록을 마친 후 [그림 13]의 '**대여하기**' 버튼을 누르면 [그림 17]과 같은 화면이 나온다. 대여정보를 확인하고 '**대여 시작**' 버튼을 누르면 커넥티드 카를 사용할 수 있다.



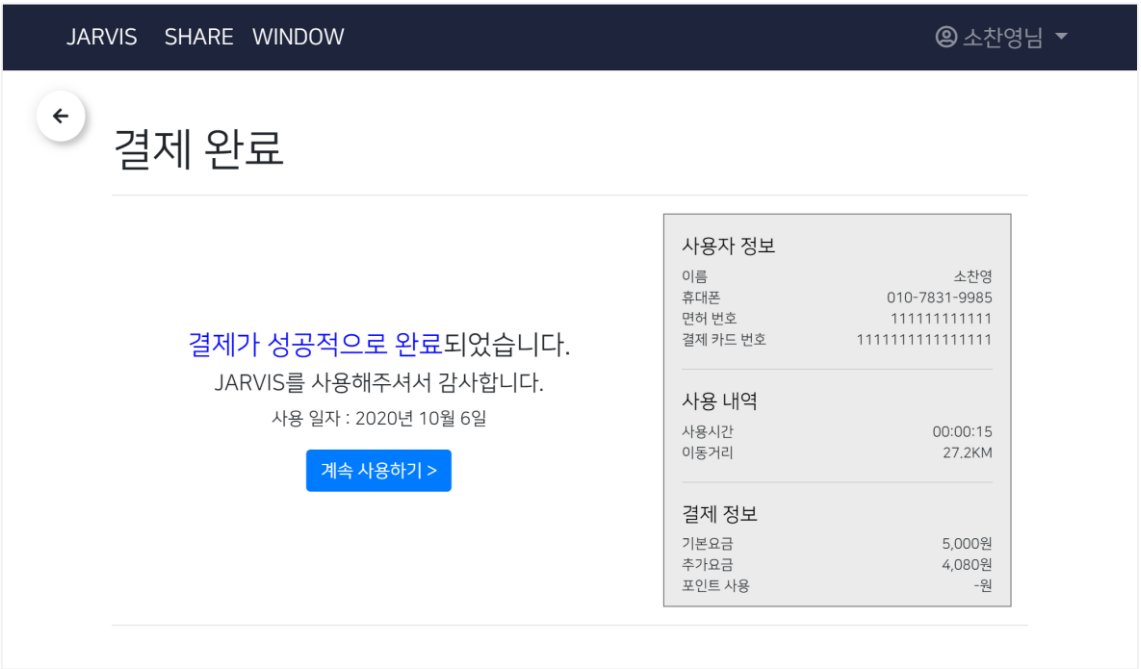
[그림 18] 커넥티드 카 대여 종료

차량을 빌려서 사용중인 대시보드의 모습이다. 처음과 달리 오른쪽 메뉴에 '**대여종료**' 버튼이 활성화된다. 차량 이용을 마친 대여자는 [그림 18]의 '**대여종료**' 버튼을 누른다.



[그림 19] 대여 기록 및 결제 정보 확인 페이지

[그림 18]의 '**대여종료**' 버튼을 누르면 넘어오는 페이지로, 공유 비용을 결제하기 전 사용자가 커넥티드 카를 타고 주행한 기록, 이동거리, 결제 정보 등을 확인할 수 있다. 내용을 확인한 후 '**결제하기**' 버튼을 누르면 결제가 진행된다.



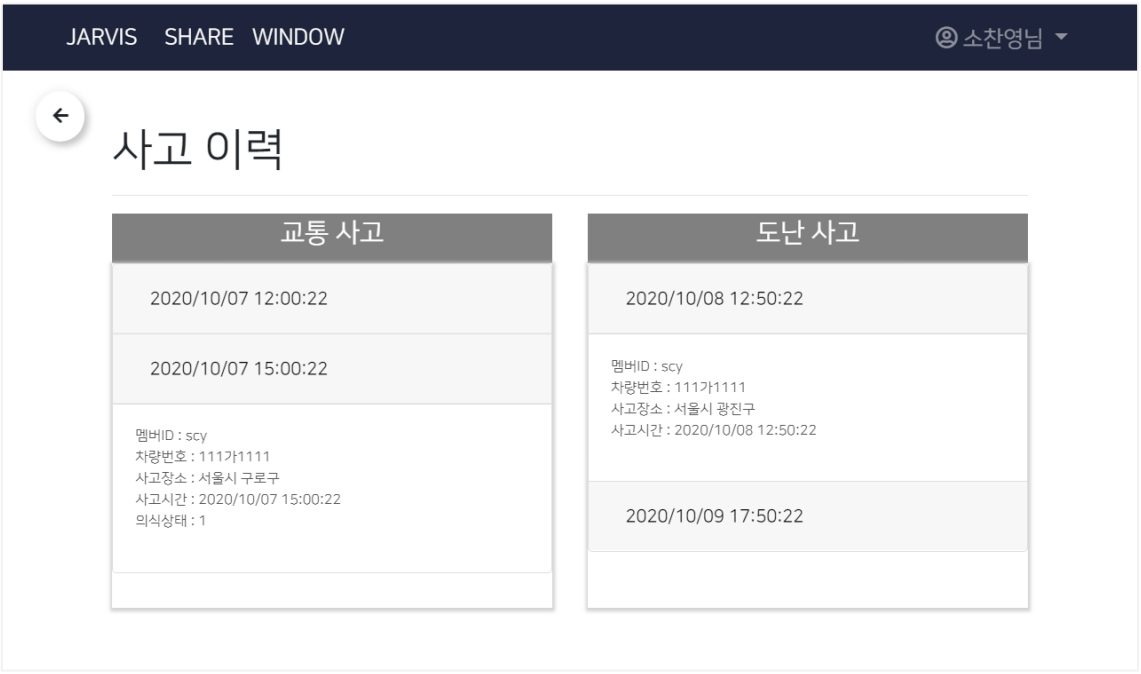
[그림 20] 결제 완료

결제를 완료한 후 결제내역 및 이용정보를 보여준다. 다시 이용하려면 '**계속 사용하기**' 버튼을 누르면 되고 이용을 종료하려면 우측 상단에 이용자 이름을 클릭한 후 '**로그아웃**' 버튼을 누르고 차량에서 하차하면 자동으로 차량이 잠기면서 서비스 이용이 종료된다.

4. 서비스 부가 기능

[그림 21] 커넥티드 카에서 스마트 홈 제어

webOS 기반의 스마트 홈에 거주중인 JARVIS 이용자는 JARVIS 계정을 통해 스마트 홈과 연동할 수 있다. 그로 인해 스마트 홈의 모든 기능을 커넥티드 카에서 쉽게 사용할 수 있다.



[그림 22] 커넥티드 카 교통사고, 도난사고 관련 기록 확인

[그림 14]의 좌측에 있는 '사고이력'을 누르면 접근할 수 있는 페이지로, 커넥티드 카 이용 중 발생한 교통 사고나 도난 사고에 대한 데이터를 수집해 보여준다.

○ **개발환경**

	커넥티드 카 프로토 타입	스마트 윈도우	공유 모빌리티 서비스
분류	Native Service	Native Service	JSService
타켓OS	webOS OSE 2.6.0		
개발PC	Ubuntu 18.04		
개발언어	C	C, python	JavaScript
프레임워크	-	Flask	Node.js, IONIC(6.10.0)
라이브러리	wiringPi	wiringPi, OpenCV, socketio	express, socket.io, ejs, mysql, request, socket.io-client, LS2 API
데이터베이스	MySQL 5.7.31		
디버거	gdb		ares-inspect, ares-server ls-monitor, 개발자 도구
IDE	Visual Studio Code(1.49.3), CMAKE, ares-cli		Visual Studio Code(1.49.3), ares-cli
버전관리	Github		

[표 2] webOS 커넥티드 카 공유 모빌리티 파트별 개발환경

□ 개발 프로그램 설명

○ 파일 구성

1. Web App

1.1. com.app.rora

JARVIS를 구성하는 각종 Native 서비스, JS Service를 사용할 수 있는 어플리케이션이다. html로 작성된 webapp 방식이며, JSService로 구동되는 커넥티드 카 대시보드와 공유 모빌리티 서비스를 webOS app방식으로 실행시키는 역할을 한다.

```
com.app.rora
├── appinfo.json // JARVIS 앱 메타데이터 및 서비스 퍼미션 정의
├── exec-script // webOS 부팅 시 JARVIS 서비스 실행을 위한 부팅 스크립트
├── icon.png // JARVIS 앱 아이콘
├── index.html // JARVIS (JSService) 로딩
└── README.md // README 파일
```

[표 3] com.app.rora 소스코드 구조

2. Native Service

2.1. com.app.rora.service.carcontrol

webOS가 탑재된 커넥티드 카를 OS level에서 제어할 수 있는 기능을 제공하는 Native 서비스다. 우리는 커넥티드 카에 webOS를 탑재해 실제로 운용해보고자 직접 커넥티드 카 프로토 타입(RC카)을 제작했다. 제작한 커넥티드 카를 제어하려면 임베디드 보드(라즈베리파이)의 gpio핀을 이용해 하드웨어를 제어할 수 있다. gpio핀을 사용하려면 webOS에 gpio 모듈 설치가 필요하기 때문에 webOS OSE 패키지를 직접 빌드해 설치했다.

그 다음 gpio 모듈을 활용해 차량을 제어하기 위해 본 native service를 개발하였다. 이 native service는 C언어로 개발되었으며 webOS를 통해 커넥티드 카에 시동, 전진, 좌회전, 우회전, 후진, 정지 등 차량을 움직일 수 있다. 만약 차량 관련 기능을 더 추가하고자 하면 이 서비스에 다양한 하드웨어 제어 코드를 추가하여 서비스 개선이 가능하다.

```
com.app.rora.service.carcontrol/
├── BUILD
│   ├── 빌드관련 파일 생략
├── CMakeLists.txt // CMake 빌드 파일
├── include
│   ├── gpio_handle.h // GPIO pin 및 차량 제어 함수 선언
│   └── sensor_thread.h // 커넥티드 카에서 사용되는 센서 감지 함수 선언
├── pkg_arm
│   ├── 실행 바이너리 리스트 생략
├── README.md // README 파일
```

```
├──── services.json // webOS OSE에서 본 서비스 실행 시 참조하는 메타데이터 파일
└──── src
    ├──── gpio_handle.c // GPIO pin 및 차량 제어 함수 기능 구현
    ├──── main.c // 서비스를 webOS에서 사용할 수 있게 백그라운드에 등록하는 코드
    └──── sensor_thread.c // 커넥티드 카에서 사용되는 센서 감지 함수 기능 구현
```

[표 4] com.app.rora.service.carcontrol 소스코드 구조

2.2. com.app.rora.service.biometrics

생체 인증을 위한 Native 서비스다. 커넥티드 카에 연결된 각종센서 및 카메라로부터 생체 데이터를 실시간으로 수집한다. 대표적인 기능은 사용자가 커넥티드 카 탑승시 운전석 쪽 창문에 노크(knock)를 하면 창문에 연결된 충격센서에서 input을 감지하고 사용자 얼굴인증을 위해 JSService로 동작되는 com.app.rora.service.webcontrol 서비스에 카메라 모듈 실행을 요청한다.

그 후 카메라 모듈과 함께 OpenCV 모듈에서 얼굴인식을 처리하게 된다. 사용자 인증이 완료되면 webOS 커넥티드 카 및 JARVIS를 사용할 수 있다. JARVIS의 라이프 사이클은 com.app.rora.service.biometrics 서비스로부터 시작된다고 할 수 있다.

```
com.app.rora.service.biometrics
├──── BUILD
|    ├──── 빌드관련 파일 생략
├──── CMakeLists.txt // CMake 빌드 파일
├──── include
|    └──── gpio_handle.h // 생체인증 관련 창문 노크 감지 함수 선언
├──── pkg_arm
|    ├──── 실행 바이너리 리스트 생략
├──── README.md // README 파일
├──── services.json // webOS OSE에서 본 서비스 실행 시 참조하는 메타데이터 파일
└──── src
    ├──── gpio_handle.c // 생체인증 관련 창문 노크 감지 함수 기능 구현
    └──── main.c // 서비스를 webOS에서 사용할 수 있게 백그라운드에 등록하는 코드
```

[표 5] com.app.rora.service.biometrics 소스코드 구조

3. JSService

3.1) com.app.rora.service.webcontrol

JARVIS 공유 모빌리티 서비스의 핵심 기능을 구현한 JSService다. webcontrol은 커넥티드 카 제어를 위한 대시보드, 생체인증 로그인, 차량 소유 확인, 대여하기, 결제하기, 컴퓨터 비전, DB 연동, webOS Service 호출 및 스마트홈 연동과 같은 webOS 플랫폼 간 연동 등 공유 서비스에 필요한 주요 기능을 제공한다. webOS OSE 공식 홈페이지에 명시된 Node.js로 개발하는 특징을 이용해 webOS Application을 보조하기 위한 단순 백그라운드

JSService가 아닌 Node.js의 express 모듈을 이용해 웹 서버로 개발했다.

다시 말해 webOS 뿐만 아니라 외부의 브라우저를 통해 원격에서 webcontrol 서비스에 접근이 가능하다. webcontrol을 웹 서버 방식으로 개발한 이유는 커넥티드 카를 webOS를 비롯해 모바일, PC 등 다양한 플랫폼과 호환성을 높이기 위함이다. 그리고 커넥티드 카는 그 특성상 실시간으로 데이터를 빠르게 주고받아 처리해야하기 때문에 웹 소켓 라이브러리 socketIO를 사용해 실시간으로 데이터를 주고받도록 설계했다.

JSService의 이러한 구조를 통해 webOS, 모바일 앱, 태블릿, PC 등 다양한 플랫폼에서 webOS 커넥티드 카 공유 모빌리티 서비스를 이용할 수 있다. 그리고 일반적인 JSService처럼 webOS의 타 어플리케이션과 JSService에서 webcontrol에 정의된 서비스를 사용할 수 있도록 webcontrol의 기능을 Luna Bus를 통해 호출하는 Luna API도 개발했다. 그리고 차량 제어, 인증 관련 기능을 가진 Native Service를 Luna Bus를 통해 호출한다.

```
com.app.rora.service.webcontrol
├─── app.js // webcontrol에서 제공하는 JSService 등록, 공유 서비스 초기화 및 시작
├─── auth.js // 생체인증(얼굴인식) 후 로그인 세션 처리
├─── config
|    └─── mysql.js // 데이터베이스(MySQL) 연결 관련 기능 정의
├─── controller
|    └─── jarvisController.js // 면허 및 카드 정보 출력 메서드 정의
├─── env
|    ├─── dbConfig.json // 데이터베이스 세부 정보 정의
|    └─── env.json // 긴급구조 및 OpenCV 서버 정보 정의
├─── luna.js // luna 서비스 호출 함수 정의
├─── models
|    └─── jarvisModel.js // 서비스에서 사용하는 데이터베이스 쿼리문을 메서드로 정의
├─── node_modules
|    ├─── npm 모듈 생략
├─── README.md // README
├─── routes
|    ├─── iot.js // webOS에서 스마트홈 및 IoT 연동을 위한 router
|    ├─── main.js // express 서버의 메인 router 정의
|    └─── share.js // 공유 서비스를 제공하는 router 정의
├─── services.json // webOS OSE에서 본 서비스 실행 시 참조하는 메타데이터 파일
├─── utils.js // 자주 사용하는 기능이 정의된 유틸리티 라이브러리
└─── views
     ├─── accidenthistory.ejs // 사용자의 사고 이력 확인 페이지
     ├─── cardregistration.ejs // 사용자 신용카드 등록 페이지
     ├─── carshare.ejs // 커넥티드 카 공유 페이지
     ├─── dashboard.ejs // 커넥티드 카 대시보드 페이지
```

```
    ├─── drivinglicense.ejs // 사용자 면허 등록 페이지
    ├─── index.ejs // 커넥티드 카 컨트롤러(핸들) 페이지
    ├─── lend.ejs // 커넥티드 카 대여 페이지
    ├─── lib
    |    └─── navigation_bar.ejs // JARVIS 상단바 분리
    ├─── sharepayment.ejs // 커넥티드 카 대여 종료 및 결제 화면 페이지
    ├─── sharepaymentsuccess.ejs // 커넥티드 카 대여료 결제완료 페이지
    ├─── smarthome.ejs // 스마트홈 제어 페이지
    └─── window.ejs // 생체인증용 창문 디스플레이
```

[표 6] com.app.rora.service.webcontrol 소스코드 구조

4. OpenCV

biometrics에서 사용자의 노크를 감지한 후 webcontrol에 센서 감지를 알려주면 OpenCV server로 얼굴인식을 진행하라는 요청이 들어온다. 그 때 스마트 윈도우에 부착된 카메라가 open되면서 얼굴 인식을 진행한다.

```
openCV_face/
├─── blinkEyes.py // 사용자 눈깜빡임 기능
├─── data
|    ├─── haarcascades // 사람의 신체 인식에 사용되는 데이터 정보
|    └─── shape_predictor_68_face_landmarks.dat // 얼굴 특징점 68개를 찍는 정보 파일
├─── faceEye.py // 사용자 눈을 포착하는 기능
├─── README.md
└─── webstreaming
    ├─── faceGetData.py // 얼굴 데이터를 저장하고, 훈련데이터를 생성하는 기능
    ├─── haarcascade_frontalface_alt2.xml // 얼굴 인식에 사용되는 데이터 정보
    ├─── haarcascade_frontalface_default.xml // 얼굴 인식에 사용되는 데이터 정보
    ├─── main.py // 실시간 사용자 얼굴 인식 기능
    ├─── recognition.py // 훈련 데이터를 기반으로 사용자 얼굴 인식을 하는 기능
    ├─── templates
    |    └─── index.html // 사용자 얼굴 실시간 인식 페이지
```

[표 7] OpenCV 소스코드 구조

5. 모바일 하이브리드 앱

webOS의 공유 모빌리티를 모바일에서도 이용이 가능함을 증명하기 위해 개발한 모바일 하이브리드 앱이다.

```
com.app.rora.mobile/
├─── google-services.json // FCM 사용을 위한 메타데이터 파일
├─── package.json // ionic framework을 구성하는 모듈 및 디펜던시 정의
├─── src
|    ├─── app
|    |    ├─── app.component.ts // 사고 발생 시 notification 정보 수신
|    |    ├─── tab1 // 카카오맵 API를 통해 차량의 현재 위치 표시
|    |    ├─── tab3 // 커넥티드 카 사고 이력 확인
```

[표 8] 모바일 앱 소스코드 구조

○ **함수별 기능**

1. com.app.rora.service.carcontrol : 차량 제어 관련 주요 함수

- gpio_init() : 차량 제어 gpio 핀 초기화
- forward() : 차량 전진
- backward() : 차량 후진
- right() : 차량 우회전
- left() : 차량 좌회전
- stop() : 차량 정지
- get_infrared_value(LSHandle *sh, LSMessage *message, void *data) : 커넥티드 카에 부착된 IR 센서, 초음파 센서를 통해 교통사고를 감지하는 함수
- *check_obstacle_thread(LSHandle *sh) : get_infrared_value 함수를 무한반복해 실시간으로 사고를 감지하는 thread 함수

2. com.app.rora.service.biometrics : 생체인증(얼굴인식) 관련 주요 함수

- gpio_init() : 생체인증 관련 gpio 핀 초기화
- *knock_check_thread(LSHandle *sh) : 스마트 윈도우에 사용자가 창문을 두드리는지 실시간으로 감지하는 무한반복 thread 함수

3. com.app.rora.service.webcontrol : 공유 모빌리티 관련 주요 함수

- service.register() : webcontorl에서 정의한 Luna API를 webOS에 등록
- http.listen(env.port, function(){}) : express 서버 시작
- utils.init(service, http) : SocketIO(웹 소켓) 및 OpenCV 서버에 대한 초기화
- utils.tts(ment) : TTS 호출
- utils.start_assistant(), utils.stop_assistant() : VPA(가상비서) 시작, 종료
- utils.help_request(type) : 긴급상황 발생 시 외부 서버에 구조 요청
- utils.request_camera(socketio, socketio_client) : 스마트 윈도우 센서가 감지되면 실행되

는 함수다. 이 함수가 실행되면 OpenCV 서버로 카메라 동작 및 얼굴인식 요청

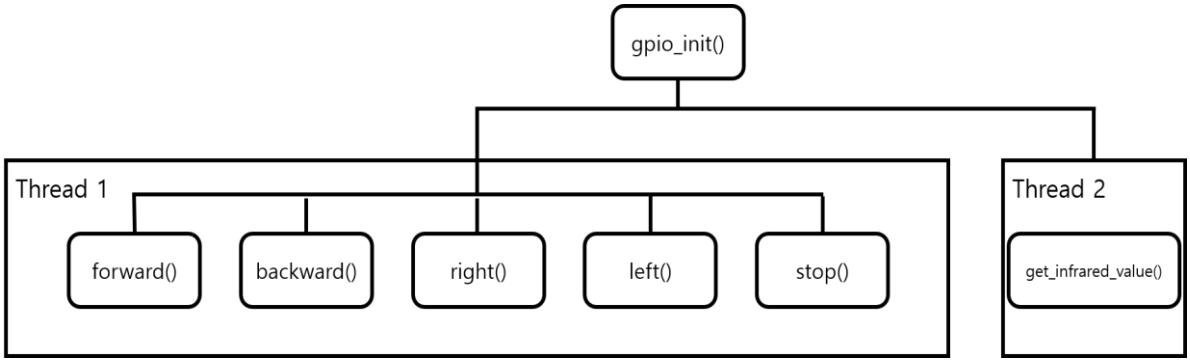
- auth.jarvis_login(auth_data) : 얼굴인증을 통해 정상 사용자로 인증되면 웹 서버 세션을 발급받기 위해 실행되는 로그인 함수
- auth.jarvis_logout() : 서비스 이용을 마치고 로그아웃 하는 함수
- mysql.mysqlConnection : MySQL과 연결하기 위해 정의된 객체로, DB 초기화 메서드, init(), DB 연결 메서드 open(), DB 연결해제 메서드, close() 정의
- share.js의 router.post('/', function(req, res){}) : 출발지, 시간 등 정보 입력 후 차량 대여
- share.js의 router.get('/payment', function(req, res){}) : 커넥티드 카 대여 종료 시 대여 금액 지불 기능

4. OpenCV 함수

- face_extractor(self, img) : 사진에서 얼굴 부분만 crop하여 리턴 처리한다.
- createFolder(self, directory) : 만약 처음 등록하는 사용자라면 입력되는 사용자 ID로 폴더 생성
- saveTrainName (self, name, tpath) : 사용자 이름, 사진이 저장된 경로를 딕셔너리 저장
- saveName(self, name, path) : 사용자 이름, 사용자의 사진 저장 경로를 딕셔너리로 저장
- savePath(self, path) : 사용자의 사진을 저장할 폴더의 경로
- inputName(self) : 사용자 회원 가입시 등록한 ID를 입력
- train(self, name) : 사전에 등록된 사용자 사진 기반으로 사용자 얼굴 학습을 위한 함수
- trains(self) : 여러 사용자들을 학습하기 위해 정의된 train(self, name)을 사용하여 학습을 진행하는 함수
- face_detector(self, frame) : 카메라를 통해 입력되는 영상에서 사용자의 얼굴 부분을 찾기 위한 함수
- get_test(self) : 실시간 영상을 송출하기 위한 함수
- get_frame(self) : 사용자의 사진을 토대로 학습된 데이터를 기반으로 실시간 영상을 통해 입력되는 사용자의 얼굴과 비교하여 confidence를 구하여 등록된 사용자인지 아닌지를 판별하는 함수
- index() : 웹페이지 렌더링하기 위한 함수
- sqlResult(name) : DB 접속 및 사용자 조회를 위함 함수
- gen(recognition) : recognition.py의 클래스를 사용하여 리턴 받는 값인 frame은 웹 페이지에 실시간 스트리밍, label은 판별된 사용자의 아이디를 입력 받아 데이터베이스에 등록된 사용자 조회를 하는 함수, 그리고 등록된 사용자 판별 시 클라이언트에게 결과 전송
- video_feed() : gen(recognition)을 실행하기 위한 함수

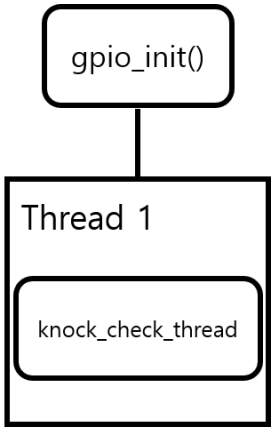
○ **주요 함수의 흐름도**

- com.app.rora.service.carcontrol (차량제어)



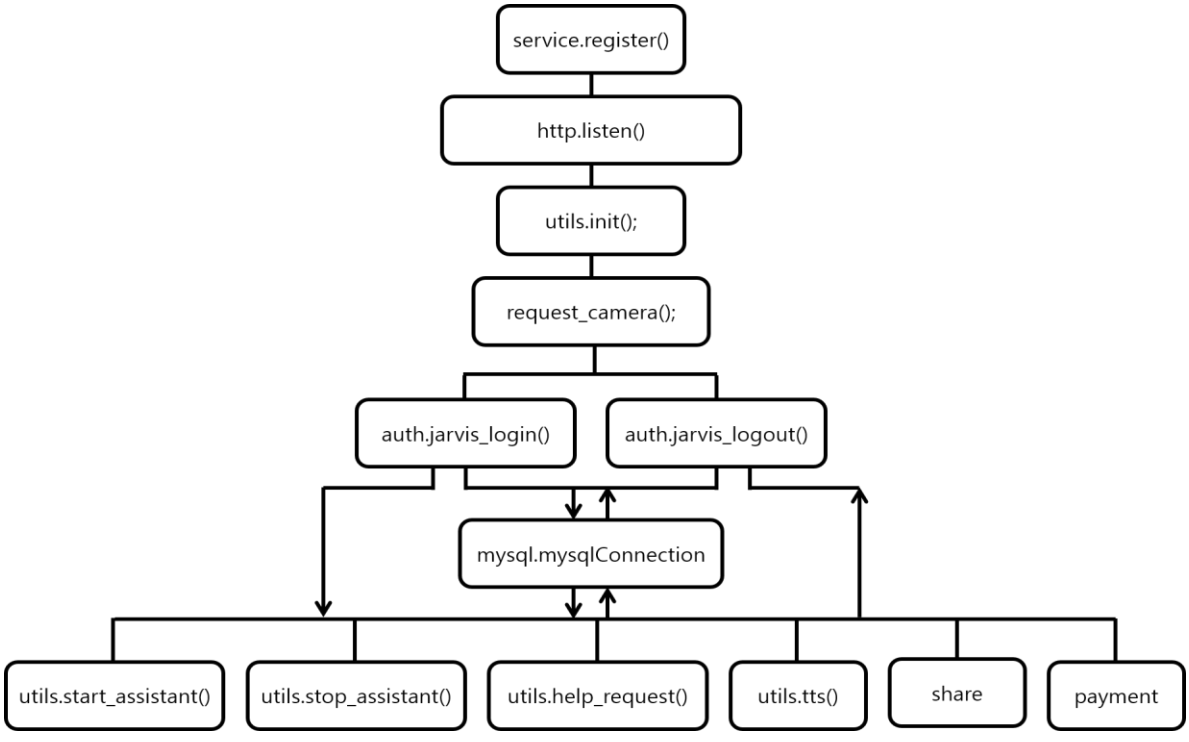
[그림 23] com.app.rora.service.carcontrol 주요 함수 흐름도

- com.app.rora.service.biometrics (생체인증)



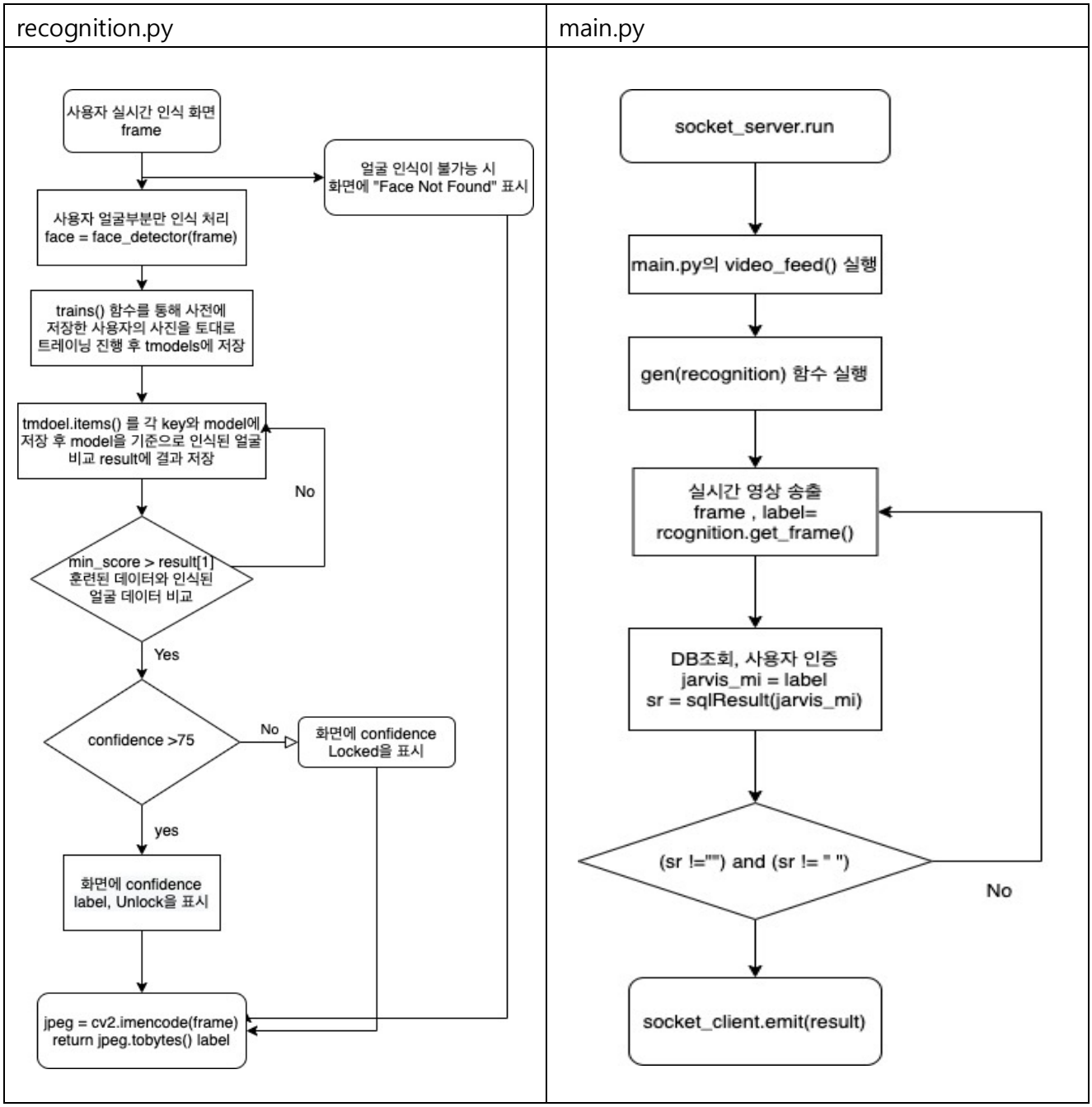
[그림 24] com.app.rora.service.biometrics 주요 함수 흐름도

- com.app.rora.service.webcontrol (공유 모빌리티 서비스)



[그림 25] com.app.rora.service.webcontrol 주요 함수 흐름도

- **OpenCV (얼굴인식)**



[그림 26] OpenCV 주요 함수 흐름도

○ **기술적 차별성**

- Node.js로 개발되는 JSService의 특징을 활용해 webcontrol을 웹 서버 방식으로 개발했다. webOS 플랫폼을 비롯해 모바일, PC 등 다양한 플랫폼에서 공유 모빌리티 서비스를 이용할 수 있다.
- 스마트 윈도우 기술을 적용해 창문 노크 후 얼굴 인식을 통해 간편하게 차량 인증 및 서비스 이용이 가능하다.
- 커넥티드 카에 부착된 다양한 센서를 활용해 실시간으로 차량 및 사용자의 상태를 파악하고 긴급 상황 발생시 자동으로 구조 요청을 보내 사용자의 안전성을 보장한다.

□ 개발 중 장애요인과 해결방안

○ gpio 및 wiringPi 라이브러리 설치

장애요인	webOS OSE 빌드 과정에서 gpio 패키지 빌드 불가
해결방안	구글링을 통해 해결방법을 검색했지만 webOS에 적합한 방법이 나오지 않았다. 그래서 GPIO 패키지 빌드 시 나오는 에러를 분석했고, meta-layer에서 참조하는 wiringPi 라이브러리의 레포지토리(git://git.drogon.net/wiringPi) 연결 문제인 것을 확인했다. 해당 레포지토리는 현재 사이트가 다운되어 있고, 구글링을 통해 찾아낸 비공식 임시 사이트(git://github.com/WiringPi/WiringPi) 주소를 yocto 레시피파일에 작성한 후 정상적으로 빌드되어 webOS에 설치할 수 있었다. 대회 종료 후 해당 이슈관련해 webOS OSE 컨트리뷰트에 도전할 생각이다.

○ OpenCV

장애요인(1)	webOS OSE에 OpenCV 설치 과정 중 dlib 라이브러리 설치에 문제가 발생했다. 해당 라이브러리는 생체 인증에 활용하는 중요한 라이브러리로, 개발일정에 많은 딜레이 발생
해결방안(1)	webOS OSE 커뮤니티를 통해 오픈소스 개발자들과 해결을 논의하고자 이슈 글을 올렸으나, 해결책이 나오지 않았고, 임베디드 소프트웨어 경진대회 8월 기술지원 미팅 때 해당 이슈를 담당자에게 문의한 결과, 개발사(LG)도 인지하고 있는 이슈로 현재 webOS를 통한 해결 방안이 존재하지 않다고 한다. 기술 지원 미팅 때 LG 연구원님과 협의 후 OpenCV의 경우 webOS 내부에 탑재하지 않고, 별도의 하드웨어를 사용해 구현해도 인정한다는 이야기를 들어 별도의 라즈베리파이를 추가하여 카메라 및 OpenCV를 webOS와 연동하여 동작하도록 했다. 대회와 별개로 dlib 라이브러리 관련 이슈 해결 방법을 논의 후 webOS OSE에 컨트리뷰트 할 예정이다.

장애요인(2)	저장된 사용자의 사진을 기반으로 미리 훈련된 데이터를 토대로 얼굴 판별 시 정확도가 떨어지는 현상이 발생
해결방안(2)	미리 훈련된 데이터를 사용하지 않고, 실시간으로 얼굴이 입력이 될 때, 사전에 저장된 사용자들의 사진을 실시간으로 훈련시켜 비슷한 얼굴을 찾아내는 방식으로 바꿔 정확도를 높이는 결과를 가지고 올 수 있었다.

□ 개발결과물의 차별성

○ 스마트 윈도우를 통한 생체 인증 방식 도입

시중에 판매되는 차량의 경우 물리적인 스마트 키 혹은 스마트폰의 전자키를 이용해 차량의 문을 열고 탑승할 수 있다. 이러한 방식은 손에 key를 쥐고 특정 행동을 해야하는 번거로움이 있다. 그리고 현재는 스마트폰이 대중화 되었지만 근미래에는 mobileless를 지향하는 기술이 보편화되어 스마트폰이 필요 없어지는 시대가 도래할 것이다. 그에 걸맞는 기술을 적용하고자 webOS 커넥티드 카에 스마트 윈도우를 도입했다. 스마트 윈도우를 활용해 별도의 키를 사용하지 않고, 스마트 윈도우에 사용자가 노크하면 카메라가 사용자를 판별해 차 문을 열어주는 편리한 기술을 적용했다.

○ 생체인식으로 인한 보안성 강화

보편적인 로그인 방식은 아이디와 비밀번호를 입력 받아 사용자를 구분하는 방법이다. 이는 오래전부터 검증된 로그인 방식이지만 패스워드가 노출되거나 해킹을 통해 계정을 탈취당할 경우 진짜 사용자가 아니더라도 탈취한 계정을 통해 로그인이 가능하다. 그로인해 금전적 피해로 이어질 수 있다. 시중에 판매되는 자동차도 마찬가지다. 얼마전까지 물리적인 key를 소지하고 있어야만 차에 들어갈 수 있었고, 요즘은 스마트키라고 해서 전자키가 나왔지만 이 방식도 무선 신호를 해킹한다면 쉽게 자동차를 탈취할 수 있다. 하지만 생체인증은 유일한 개인의 생체 정보를 토대로 안전하게 인증이 가능하기 때문에 보안성이 향상된 서비스 이용이 가능하다.

○ webOS 플랫폼간 호환성

JARVIS는 webOS OSE 어플리케이션 및 서비스로, webOS 플랫폼인 Smart TV, Digital Signage, Smart Watch 등 webOS가 탑재된 다양한 디바이스에서 언제 어디서든 JARVIS에 접속해 차량을 현재 위치로 호출 및 공유가 가능하다. 예를 들어 강남역에 webOS Signage가 설치되어 있다면 강남역을 지나는 사용자가 차량 공유를 원할 때 스마트폰 이용 없이 바로 앞에 있는 디지털 사이니지에서 JARVIS를 실행시켜 현재 위치로 차량을 호출해 이용할 수 있다. 마찬가지로 webOS TV가 있는 집에서 호출, Smart Watch를 이용해 외부에서 차량 호출 및 이용이 가능하다. 그리고 webOS 커넥티드 카에서 스마트 홈을 제어하는 것도 가능하다. webOS 플랫폼에서 활용범위가 무궁무진한 장점이 있다.

○ 사용자 안전을 보장하는 긴급 구조 시스템

커넥티드 카에 탑재된 다양한 센서들이 실시간으로 수집하는 데이터를 분석 및 종합하 여 현재 차량과 운전자의 상태를 지속적으로 판단한다. 만약 교통사고가 발생하거나 운전자의 상태가 좋지 않을 경우 즉각적으로 차량에서 수집한 데이터를 가시화하여 119에 신고하여 빠른 구조로 인명피해를 막을 수 있다. 특히 야간이나 인적이 드문 곳에서 발생한 사고 시 운전자가 의식불명인 위급한 상황이라면 골든타임을 지키는데 많은 도움이 될 것이다.

□ 개발 일정

No	내용	2020年 6月	7月	8月	9月
1	JARVIS 기능 설계				
2	webOS OSE 분석				
3	개발환경 구축				
4	AWS 서버 구축				
5	JARVIS UI 개발				
6	커넥티드 카 프로토타입 개발				
7	커넥티드 카 웹 컨트롤러 개발				
8	JARVIS 대시보드(시동, 기어조작, VPA) 개발				
9	TTS 모듈화				
10	JARVIS DB 설계				
11	생체인증 모듈 개발				
12	생체인증 모듈 webOS 연동				
13	카 셰어링 서비스(위치파악 대여, 결제) 개발				
14	카 셰어링 서비스(생체인증 로그인, 공유 서비스 유틸리티 모듈)				
15	커넥티드 카 스마트 윈도우 하드웨어 개발				
16	구조요청 시스템 개발				
17	webOS와 스마트 홈 연동				
18	webOS와 디지털 사이니지 연동				

□ 팀 업무 분장

No	구분	성명	참여인원의 업무 분장
1	팀장	소찬영	1. 프로젝트 총괄 (방향성 제시, 일정 및 업무 분할) 2. 소스코드 버전관리 3. 커넥티드 카 프로토타입(RC카) 제작 4. 생체인증 모듈 개발 및 webOS 로그인 세션처리 5. JARVIS 대시보드 개발 (시동, 기어조작, VPA(가상비서)) 6. JARVIS 카 셰어링 서비스(생체인증 로그인, 유틸리티 모듈) 개발 7. Google Assistant를 활용한 VPA(가상비서) 서비스 탑재 8. 가상비서 활용을 위한 TTS 서비스 탑재 9. webOS OSE와 스마트홈 연동
2	팀원	배상준	1. JARVIS UI 개발 2. 소스코드 버전관리 3. 응급구조 및 도난 방지 기능 개발 4. JARVIS 모바일(하이브리드)앱 개발 5. 모바일 앱과 webOS 연동 6. JARVIS DB 설계 7. JARVIS 카 셰어링 서비스(위치파악, 대여, 결제) 개발 8. 카카오 맵 API 적용 9. webOS OSE와 디지털 사이니지 연동
3	팀원	한지훈	1. 커넥티드 카 프로토타입(RC카) 제작 2. 커넥티드 카 스마트 윈도우 하드웨어 개발 3. 커넥티드 카 웹 컨트롤러(핸들) 개발 4. webOS OSE 이미지 및 라이브러리 패키지 빌드 5. AWS 서버 구축 6. 응급구조 및 도난 방지 기능 개발 7. 컴퓨터 비전(OpenCV) 기술을 이용한 얼굴인식 기능 개발 8. 생체인증 모듈 개발 및 webOS 연동 9. JARVIS DB 설계