웹/AI 기반 스마트 주차장 시스템

# 설 계 서

2020.05.31


**가천대학교 컴퓨터공학과**

팀장 ■ 홍기현(201533892)
팀원 ■ 이우석(201433860)
팀원 ■ 이다은(201632230)
팀원 ■ 한혜경(201736058)

# 문서 정보

**본 문서는 웹 기반 지능형 주차관제 모니터링 시스템 개발을 위한 설계서입니다.**

| | |
|---|---|
| **버 전** | 1.0 |
| **작성일** | 2020-05-31 |
| **상 태** | ■완료 □진행 중 □초안 |

| 버전 | 변경한 사람 | 변경한 날짜 | 버전업 변경(또는 추가)내용 |
|---|---|---|---|
| 0.1 | 이병문교수 | 2020-05-18 | 요구규격 및 설계서 양식(템플릿) 정의 |
| 0.2 | 이다은 | 2020-05-25 | 요구사항 정의 작성 |
| 0.3 | 한혜경 | 2020-05-25 | 하드웨어 구성 및 소프트웨어 구성 |
| 0.4 | 이우석 | 2020-05-26 | 화면 UI 구성 및 스토리보드 설계 |
| 0.45 | 홍기현 | 2020-05-27 | REST API 정의 / ERD 작성 |
| 0.5 | 이다은 | 2020-05-27 | 개발 환경 구성 및 작성 |
| 0.55 | 한혜경 | 2020-05-28 | 하드웨어 구성 및 소프트웨어 구성 |
| 0.6 | 이우석 | 2020-05-29 | 기기 외관 디자인 |
| 0.65 | 홍기현 | 2020-05-29 | REST API 정의 및 설계 / 데이터베이스 설계 |
| 0.7 | 이다은 | 2020-05-30 | 프로젝트 개요 및 서비스 요구사항 정의 작성 |
| 0.8 | 한혜경 | 2020-05-30 | 하드웨어 구성 및 소프트웨어 구성 마무리 |
| 0.9 | 이우석 | 2020-05-31 | 운영환경 및 소스 디렉터리 구조 작성 |
| 1.0 | 홍기현 | 2020-05-31 | 설계서 최종 검토 및 보완 |

# 목 차

# 1. 프로젝트 개요

## 1.1. 개발목표와 범위

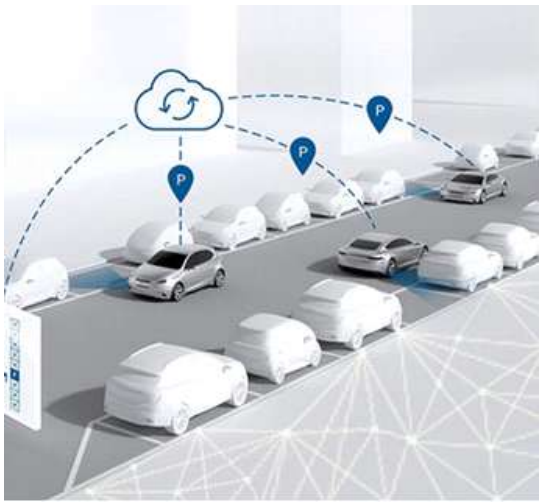
■ 개발 필요성

미국 교통분석업체 INRIX에 따르면 워싱턴에서 주차공간을 찾는데 낭비하는 시간이 연간 65시간에 달한다. LA는 이보다 많은 85시간, 악명 높은 뉴욕에서는 107시간을 허비한다는 조사결과도 있다. 서울도 크게 다르지 않을 것이다. 글로벌 컨설팅 회사 프로스트&설리번 보고서에 따르면 '교통정체의 30%는 운전자가 주차공간을 찾고 있는 시간이나 상황에서 발생' 한다고 한다. 주차장을 찾기 위해 소비하는 시간과 주차장에 진입하기 위해 길게 늘어선 자동차가 교통흐름을 방해하고 정체를 유발하면서 개인과 사회적 비용을 낭비하는 원인이라는 지적도 오래전부터 나왔다.

우리나라 주차장은 상당 부분 스마트해 졌지만 여전히 많은 주차장이 관리직원을 필요로 한다. 앱이 알려준 정보와 실제 정보가 달라 낭패를 보는 일도 적지 않다. 특히 주택가의 거주자 우선주차구역, 도로 인근 공양주차장 등 노상에 마련된 개방형 주차장에선 무인주차시스템을 거의 찾아볼 수 없는 점도 스마트 주차장 보급의 걸림돌로 지적된다.

이런 문제를 해결할 수 있는 스마트 무인주차관리 플랫폼이 최근 주목을 받고 있다. 스마트 무인주차관리 플랫폼은 IoT 기술 기반의 차량 검지센서와 통합 주차관제 센터를 연동해 이용자에게 정확한 주차 공간 정보를 제공하고 노상 주차장까지 무인 관리가 가능한 시스템이다.

본 프로젝트에서는 이러한 무인 관리가 가능한 스마트 주차장을 개발하고자 한다.



<그림 1.1> IoT 기반의 스마트 주차장

■ 개발목표

본 설계서는 임베디드 환경에서 차량 운전자를 위한 IoT 스마트 주차장 서비스를 개발하기 위한 설계서이다. 사용자는 차량 운전자이며, 센서 및 액츄레이터로 데이터 송수신이 자동화되어 사용하기 편리하도록 고안되는 것을 전제로 한다. IoT 스마트 주차장 서비스를 효과적으로 관리하기 위해 관리기능도 구성하였으며, 최대한 편리성과 유연성을 추구할 수 있도록 개발한다. 자동차 번호판 인식 AI로 차량 운전자 식별과 주차 시간 및 요금 정산도 간편하게 수행한다.

아래 그림 1.2는 본 서비스의 전체적인 구성을 나타내며, 차량 운전자인 사용자와 스마트 주차장 관리자가 본 서비스를 이용한다.


<그림 1.2> 스마트 주차장 플랫폼의 설계범위

## 1.2. 개발일정/산출물

본 설계서는 아래 그림 1.3의 개발일정에 따라 진행하며, 단계별로 산출물을 개발한다.

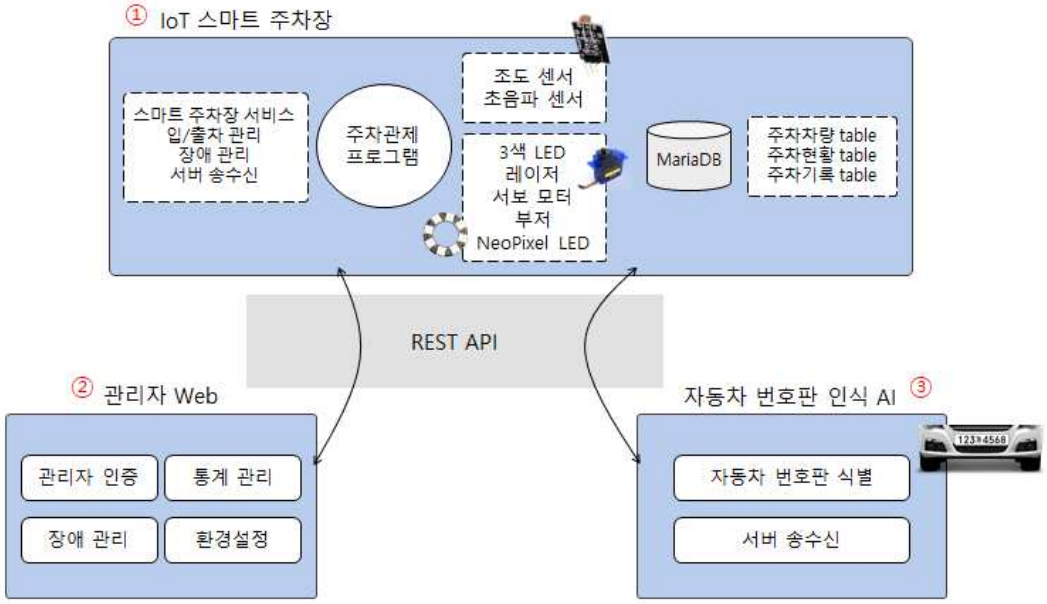

<그림 1.3> 개발일정 및 산출물

## 1.3. 개발조직/역할

| 역할 | 학번 | 이름 | 역할(주요 개발기능) |
|---|---|---|---|
| 팀장 | 201533892 | 홍기현 | 서버 송수신(주차장),<br>통계 관리(관리자),<br>장애 관리(관리자) |
| 팀원 | 201433860 | 이우석 | 입/출차 관리(주차장),<br>관리자 인증(관리자),<br>자동차 번호판 식별(AI) |
| 팀원 | 201632230 | 이다은 | 입/출차 관리(주차장),<br>장애 관리(주차장),<br>통계 관리(관리자) |
| 팀원 | 201736058 | 한혜경 | 스마트 주차장 서비스(주차장),<br>환경설정(관리자),<br>서버 송수신(AI) |

# 2. 서비스기능 정의

## 2.1. 서비스 개요




<그림 2.1> IoT 기반의 스마트 주차장 서비스 모델

① IoT 스마트 주차장은 스마트 주차장 서비스로 빈자리 안내, 주차장 길 안내, 일방통행 경보, 엘리베이터 작동, 주차 차량 감지, 두 자리 주차 차량 감지를 할 수 있다. 주차 차량 감지는 조도 센서로 감지하고 3색 LED로 표현한다. 입/출차 관리로 만차 안내, 차단기 작동, 자동차 번호판 인식, 요금징수기 작동을 할 수 있다. 만차는 NeoPixel LED의 색깔 변화로 안내한다. 또한, 장애 관리로 모듈 동기화, 로그 생성이 가능하다. 서버 송수신으로 입/출차 요청, 자동차 번호판 데이터, 주차 요금 데이터, 장애 로그 데이터를 요청하고 응답받을 수 있다. 각 모듈이 작동될 때 발생하는 모든 데이터는 REST API 방식으로 데이터베이스에 저장된다.

② 관리자는 관리자 인증으로 로그인, 로그아웃, 정보수정을 할 수 있다. 통계 관리에선 IoT 스마트 주차장의 실시간 및 주기적 진행 상황을 확인한다. 장애 관리와 환경설정에서는 IoT 스마트 주차장에 대한 모든 제어를 수행한다.

③ 자동차 번호판 인식 AI는 미리 학습된 모델로 요청받은 자동차 번호판을 식별한다. 자동차 번호판 이미지에서 추출한 차량 고유번호는 서버 송수신으로 전송되고 REST API 방식으로 데이터베이스에 저장된다.

## 2.2. 서비스 구성요소의 정의

○ IoT 스마트 주차장

IoT 스마트 주차장은 2개의 센서(조도 센서, 초음파 센서)와 5개의 액츄레이터(LED, 레이저, 서보 모터, 부저, NeoPixel LED)로 구성되며, 라즈베리파이에서 스마트 주차장 전반의 데이터를 측정하는 기능을 담당하는 기기이다. 수집한 데이터는 스마트 주차장 서버를 통해 전송되어 데이터베이스에 저장된다. 사용자가 주차장을 편리하게 이용할 수 있도록 여러 정보를 제공하고, 사용자를 제어하여 주차장 기능의 정상 작동을 도모한다.

○ IoT 스마트 주차장 사용자

IoT 스마트 주차장 사용자는 IoT 스마트 주차장을 직접 이용하는 주체이다. 사용자는 주차장의 모듈이 제공하는 정보를 통해 주차장을 효과적으로 활용할 수 있다. 사용자로부터 발생하는 모든 데이터는 주차장의 데이터베이스에 기록된다.

○ IoT 스마트 주차장 관리자

IoT 스마트 주차장 관리자는 IoT 스마트 주차장을 직접 관리하는 주체이다. 관리자는 주차장의 모듈이 얻어내는 정보를 통해 주차장을 효과적으로 제어할 수 있다. 관리자는 다양한 통계자료 분석과 모듈 장애 관리로 주차장의 운영을 지속한다.

○ IoT 스마트 주차장 서버

IoT 스마트 주차장 서버는 IoT 스마트 주차장에서 나타나는 모든 데이터의 송수신을 담당하고, 데이터베이스와의 연결을 책임지는 주체이다. 주차장 서버는 주차장, 사용자, 관리자로부터 데이터를 요청받고 응답한다. 필요에 따라 데이터베이스에 저장된 데이터를 가공해서 전달한다.

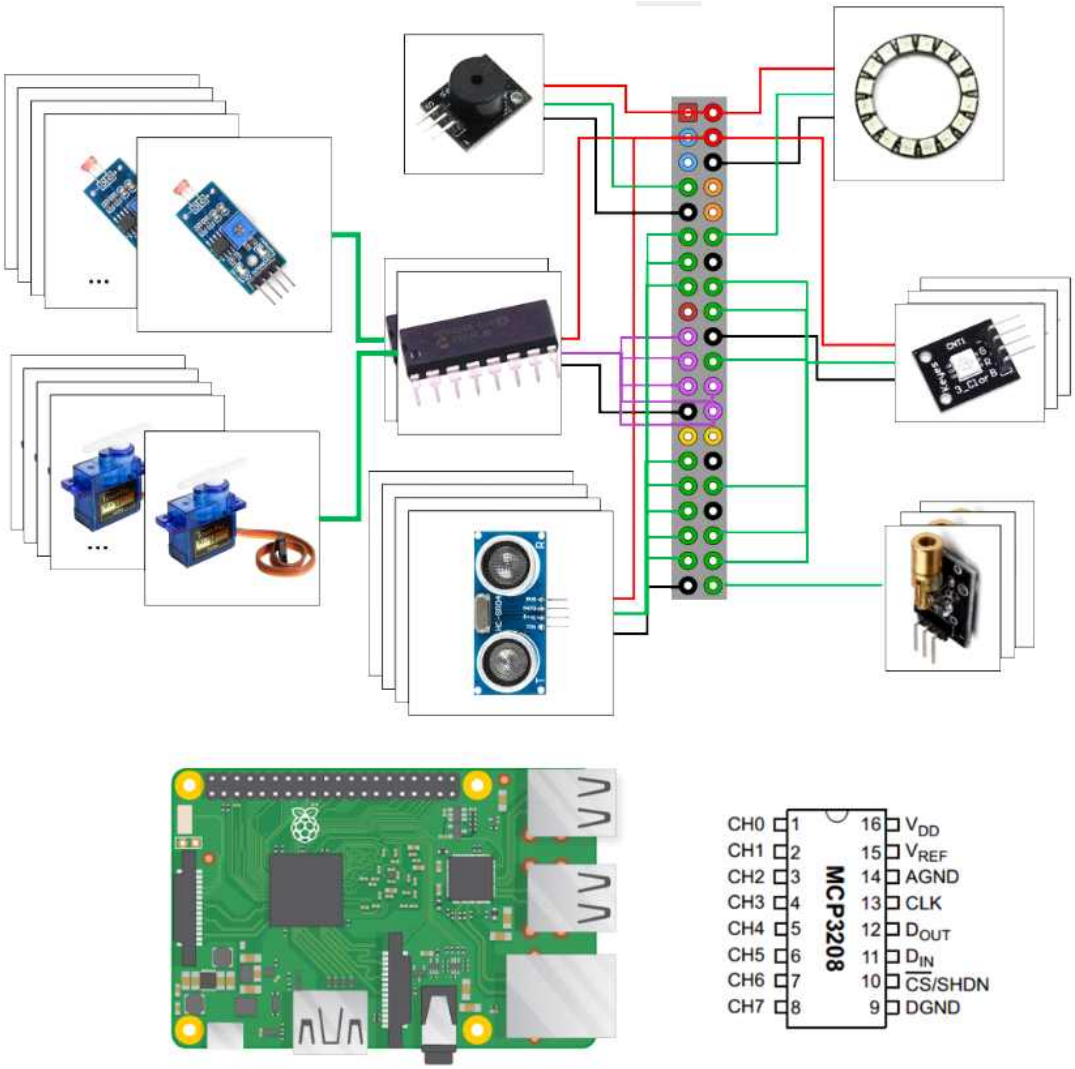
○ 자동차 번호판 인식 AI

자동차 번호판 인식 AI는 IoT 스마트 주차장에서 차량 입/출차를 책임지는 주체이다. 차량 입차 시 자동차 번호판 인식을 통해 자동차 고유번호를 확인하고, 데이터베이스에 PK 값으로 저장한다. 차량 출차 시에도 자동차 번호판 인식을 통해 자동차 고유번호를 확인하여 주차 시간과 요금 정산이 원활하게 이루어지도록 도와준다.

## 2.3. 요구사항 정의(스마트 주차장)

### ○ 하드웨어 요구사항 구성

스마트 주차장 서비스를 제공하기 위해 그림 2.2와 같이 하드웨어 구성도를 작성하였다. 주차장의 주차량을 나타내기 위해 NeoPixel LED를 사용하고 버저로 주차장의 장애 상황을 알린다. MCP3208 2개를 연결하여 9개의 조도센서와 6개의 서보모터를 제어한다. 조도센서를 이용해 자동차를 감지하고 서보모터는 물리적인 차단기나 승강기에 사용된다. 초음파 센서는 거리를 측정하여 차량의 일방통행을 측정하는 데에 사용한다. 레이저 발광 센서는 두 자리를 감지하기 위해 사용한다.



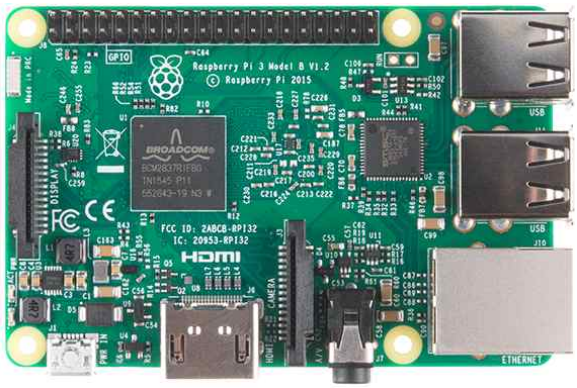

<그림 2.2> 하드웨어 구성도

| 번호 | 센서 | 센서 핀 | 라즈베리 파이 핀(wPi) |
|---|---|---|---|
| 1) | 조도센서 | A0 | ADC 0 Channel 0 ~ADC 0 Channel 7, ADC 1 Channel 0 |
| | | VCC | 3.3v |
| 2) | 서보모터 | SIG | ADC 1 Channel 1~ADC 1 Channel 6 |
| | | GND | GND |
| | | VCC | 5v |
| 3) | 3색 LED | SIG | GPIO 12, 16, 20, 23, 24, 25 |
| | | GND | GND |
| | | VCC | 5v |
| 4) | NeoPixel LED | SIG | GPIO 18 |
| | | GND | GND |
| | | VCC | 5v |
| 5) | 초음파 센서 | TRIG | GPIO 17, 27, 22, 5 |
| | | ECHO | GPIO 6, 13, 19, 26 |
| | | GND | GND |
| | | VCC | 5v |
| 6) | 부저 | SIG | GPIO 4 |
| | | GND | GND |
| | | VCC | 5v |
| 7) | MCP3208 (ADC) | MISO | GPIO 13 (MISO) |
| | | MOSI | GPIO 12 (MOSI) |
| | | SCLK | GPIO 14 (SCLK) |
| | | SS | GPIO 10 (CE0) GPIO 11 (CE1) (2개 사용) |
| | | VCC | 5v |
| | | GND | GND |
| 8) | 레이저 발광 모듈 | SIG | GPIO 40 |
| | | GND | GND |
| | | VCC | 5v |

<표 2.1> 센서/액츄레이션 핀 구성표

■ Raspberry Pi 3

스마트 주차장을 개발하기 위해 Open Source 하드웨어인 Raspberry Pi 3 B+를 사용한다. 64bit ARMv7 Quad Core CPU, 1G RAM을 지원하고 다양한 센서와 다중 센서를 동시 처리하는 컴퓨팅 파워를 제공한다. 네트워크 통신을 위한 WiFi모듈과 GPIO 인터페이스를 지원한다. 하드웨어의 사양은 표 2.2과 같다.



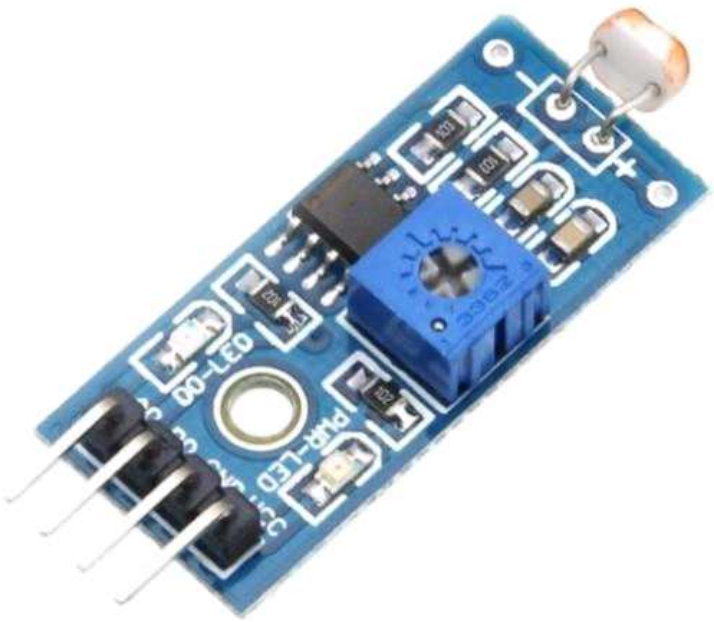
<그림 2.3> Raspberry Pi 3

| 항목 | 사양 |
|---|---|
| Processor Chipset | Boardcom BCM2837 64bit Quad Core Processor Powered Single Board Compute running at 1.2GHz |
| Processor Speed | Quad Core @ 1.2 GHz |
| 1RAM | 1GB SDRAM @ 400MHz |
| Storage | MicroSD |
| USB 2.0 | 4x USB Port |
| Max Power/voltage | 2.5A @ 5V |
| GPIO | 40Pin |
| Ethernet Port | 지원 |
| Wi-Fi | Built in |
| Bluetooth 4.1 | Built in |
| Media port | HDMI |

<표 2.2> Raspbeery Pi 3 Spec

■ **조도 센서**

스마트 주차장 서비스를 개발하기 위해서 조도센서를 사용한다. 조도센서는 주차장 바닥에 설치하여 자동차가 특정 위치에 도착한 것을 감지한다. 조도센서는 아날로그 데이터를 받을 수 있기 때문에 ADC를 이용하여 세밀한 값을 측정한다.



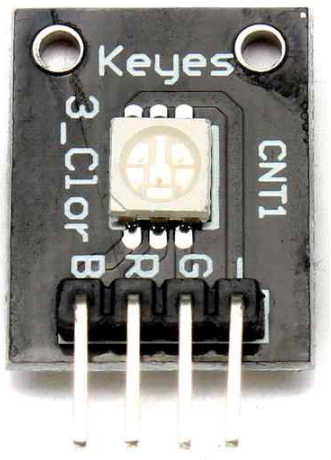

<그림 2.4> 조도(광량) 센서

| 항목 | 사양 |
| --- | --- |
| Dimension | 2.0cm X 3.2cm X 1.1cm |
| Working Voltage Range | DC 3.3V ~ 5V |
| Work Temperature | −30°C ~ 70°C |

<표 2.3> 조도 센서 사양 표

■ 3색 LED 모듈

스마트 주차장에서 운전자에게 주차장 상태를 안내하기 위해 사용한다. 주차 자리를 표시하거나 주차길을 안내한다.



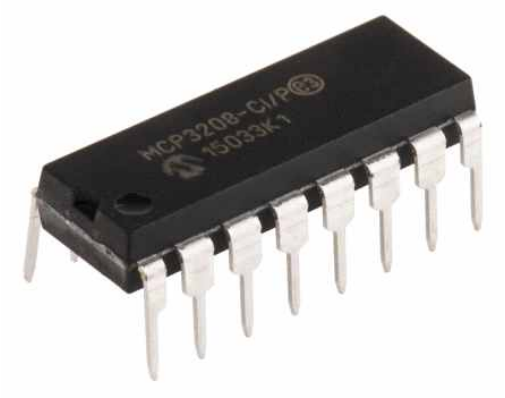

<그림 2.5> Keyes CNT1 SMD RGB LED

| 항목 | 사양 |
|---|---|
| Dimension | 20mm x 10mm x 7mm |
| Working Voltage | 5V |
| Forward Current | 20mA ~ 30mA |
| Operating Temperature | −25°C ~ 85°C |

<표 2.4> 3색 LED 모듈 사양 표

■ MCP3208

MCP3208은 아날로그 신호를(전압) 디지털 신호로 변환하는 ADC이다. 조도센서에서 측정한 아날로그 값을 12-bits(0~4095) 전기신호로 변환하여 라즈베리 파이의 SPI 인터페이스를 통해 값을 전달한다. MCP3208은 CH0부터 CH7까지 총 8개의 독립적인 채널로 구성된다. 본 프로젝트에서는 9개의 조도센서와 6개의 서보모터 제어에 사용된다.



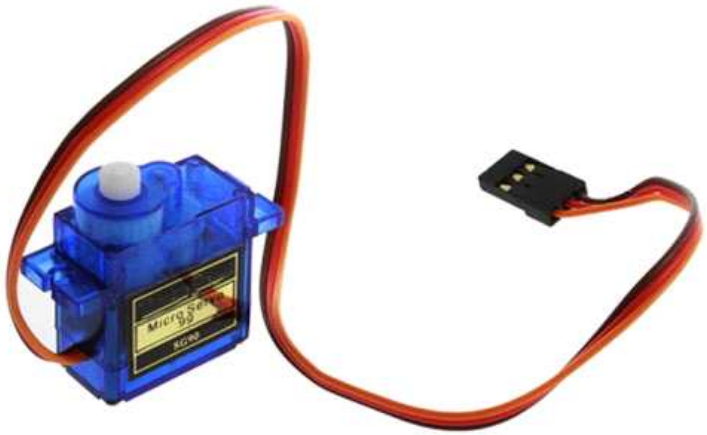

<그림 2.6> MCP3208

| 항목 | 사양 |
|---|---|
| Resolution | 12 bits |
| Interface | Serial, 4-Wire, SPI |
| Dimensions | 3.3mm x 19.05mm |
| Voltage Input | 2.7 V to 5.5 V |

<표 2.5> MCP3208 사양 표

■ **서보 모터**

서보 모터는 입력한 각도만큼 움직이는 모터이다. 주차장 차단기를 구현하기 위해 서보 모터를 사용한다.



<그림 2.7> 서보 모터

| 항목 | 사양 |
| --- | --- |
| Dimensions | 23×12.2x29mm |
| Voltage Input | 3.5V to 6V |

<표 2.6> 서보모터 사양 표

■ NeoPixel LED

NeoPixel LED는 한 칸마다 각각 다른 색을 출력할 수 있다. 스마트 주차장 밖에서 차량을 쉽게 파악하기 위해 NeoPixel LED의 색을 한 칸씩 변경하여 주차량을 보여준다. 만차인 경우 모든 LED를 붉은 색으로 보여 운전자에게 안내한다.

<그림 2.8> NeoPixel LED

| 항목 | 사양 |
| --- | --- |
| Dimensions | 36.8mm / 1.5 " |
| Voltage Input | 4V to 7V |

<표 2.7> NeoPixel LED 사양 표

■ **초음파 센서**

초음파 센서는 전방의 사물과 거리를 감지할 수 있다. 초음파 센서를 이용하여 일방통행을 무시하고 접근하는 차량을 감지해 스마트 주차장 서비스를 제공할 수 있다.


<그림 2.9> 초음파 센서

| 항목 | 사양 |
|---|---|
| Dimensions | 20mm x 45mm x 15mm |
| Voltage Input | 5V |

<표 2.8> 초음파 센서 사양 표

■ **버저**

버저는 신호를 받으면 특정한 신호음을 출력한다. 스마트 주차장의 긴급상황이나 장애가 발생했을 때 버저를 이용하여 운전자에게 알릴 수 있다.


<그림 2.10> 버저

| 항목 | 사양 |
| --- | --- |
| Dimensions | 24mm × 21mm |
| Voltage Input | 3.3V to 5V |

<표 2.9> 버저 사양 표

■ **레이저 발광 모듈**

레이저 발광 모듈은 신호를 받으면 레이저를 출력한다. 이를 통하여 조도 센서로 감지해 그 사이에 특정 물체가 가리고 있는지 파악 가능하다.

<그림 2.11> 레이저 발광 모듈

| 항목 | 사양 |
|---|---|
| Dimensions | 24mm × 21mm |
| Voltage Input | 3.3V to 5V |

<표 2.10> 레이저 발광 모듈 사양 표

○ 소프트웨어 요구사항 기능정의 (스마트 주차장 server)

| 번호 | 주요객체 | 상세기능 | 개발담당자 |
|---|---|---|---|
| 1) | 스마트 주차장 서비스 | 빈자리 안내 | 한혜경 |
| 2) | | 주차장 길 안내 | 한혜경 |
| 3) | | 일방통행 경보 | 한혜경 |
| 4) | | 엘리베이터 작동 | 한혜경 |
| 5) | | 주차 차량 감지 | 이다은 |
| 6) | | 두 자리 주차 차량 감지 | 이다은 |
| 7) | 입/출차 관리 | 만차 안내 | 이다은 |
| 8) | | 차단기 작동 | 이다은 |
| 9) | | 자동차 번호판 인식 | 이우석 |
| 10) | | 요금징수기 작동 | 이우석 |
| 11) | 장애 관리 | 모듈 동기화 | 이다은 |
| 12) | | 로그 생성 | 이다은 |
| 13) | 서버 송수신 | 입/출차 요청 | 홍기현 |
| 14) | | 자동차 번호판 데이터 | 홍기현 |
| 15) | | 주차 요금 데이터 | 홍기현 |
| 16) | | 스마트 주차장 서비스 데이터 | 홍기현 |
| 17) | | 장애 로그 데이터 | 홍기현 |

<표 2.11> 지능형 주차관제 모니터링 시스템 (스마트 주차장) 기능 정의 표

1) 빈자리 안내
- 사용자에게 3색 LED의 색으로 빈자리 정보를 제공한다.
- 빈자리면 초록색을, 아니라면 빨간색을 출력한다.

2) 주차장 길 안내
- 사용자에게 3색 LED로 주차장의 길을 안내한다.
- LED의 켜짐에 시간 변화를 주어 주차장 통행 방향을 알려준다.

3) 일방통행 경보
- 자동차를 초음파 센서로 측정하여 올바른 방향으로 이동하는지 확인한다.
- 자동차가 올바르지 않은 방향으로 이동하면 부저에서 소리가 난다.

4) 엘리베이터 작동
- 자동차가 엘리베이터 앞에 서 있으면 엘리베이터가 작동한다.
- 자동차의 유무는 조도 센서로 확인한다.

5) 주차 차량 감지
- 자동차의 유무는 각 자리에 있는 조도 센서로 확인한다.
- 자동차가 있다면 측정 센서값이 크고, 없다면 측정 센서값이 작다.

6) 두 자리 주차 차량 감지
- 각 주차 자리의 경계선에 레이저와 조도 센서를 설치한다.
- 자동차의 유무는 두 모듈을 통해 확인한다.

7) 만차 안내
- 입차기 앞에 NeoPixel LED를 두어 만차 여부를 보여준다.
- NeoPixel LED의 색에 따라 만차 여부가 달라진다.

8) 차단기 작동
- 차량의 입/출차 시 차단기를 작동시켜 차량을 제어한다.
- 차단기는 서보 모터를 통해 움직일 수 있다.

9) 자동차 번호판 인식
- AI를 통한 자동차 번호판 인식으로 각 차량 정보를 구분한다.
- 자동차 번호판 식별 정보는 DB에서 PK 값으로 활용된다.

10) 요금징수기 작동
- 사용자가 주차한 시간만큼 요금을 계산하여 징수한다.
- 요금징수기는 터치스크린 모니터로 구현된다.

11) 모듈 동기화
- 스마트 주차장의 원활한 진행을 위해 모듈이 동기화된다.
- 스마트 주차장에 문제가 생기는 경우, 모듈은 한 번에 재동기화될 수 있다.

12) 로그 생성
- 주차장 시스템에서 발생하는 모든 로그는 DB에서 조회할 수 있어야 한다.
- 주차장 시스템의 모든 로그는 DB에 기록된다.

13) 입/출차 요청
- 관리자 및 AI 서버와의 통신으로 차량 입/출차 가능 여부를 확인한다.
- 주차장 만차 시 차량 입차 요청이 거부된다.
- 자동차 번호판 데이터 부재 시 차량 출차 요청이 거부된다.

14) 자동차 번호판 데이터
- 차량 입차 시 자동차 번호판 이미지를 AI 서버에 전달한다.
- 차량 출차 시 자동차 번호판 데이터를 AI 서버로부터 전달받는다.

15) 주차 요금 데이터
- 사용자의 주차 시간에 비례하여 주차 요금을 산정한다.
- 주차 요금과 정산 여부는 DB에서 조회할 수 있다.

16) 스마트 주차장 서비스 데이터
- 스마트 주차장의 모듈에서 발생하는 모든 데이터는 DB에 저장된다.
- 모듈에 관한 정보는 DB에서 조회할 수 있다.

17) 장애 로그 데이터
- 스마트 주차장의 작동에서 발생하는 모든 장애 로그는 DB에 저장된다.
- 장애에 관한 정보는 DB에서 조회할 수 있다.

## 2.4. 소프트웨어 요구사항 정의 (관리자 server)

| 번호 | 주요객체 | 상세기능 | 개발담당자 |
|---|---|---|---|
| 1) | 관리자 인증 | 로그인 | 이우석 |
| 2) | | 로그아웃 | 이우석 |
| 3) | | 관리자 정보수정 | 이우석 |
| 4) | 통계 관리 | 전체 주차상황 파악 | 이다은 |
| 5) | | 구역별 주차상황 파악 | 이다은 |
| 6) | | 기간별 주차량 분석 | 홍기현 |
| 7) | | 분기별 매출 확인 | 홍기현 |
| 8) | 장애 관리 | 모듈 초기화 | 홍기현 |
| 9) | | 응급상황 모듈 작동 | 홍기현 |
| 10) | 환경설정 | 스마트 주차장 관리 | 한혜경 |
| 11) | | 결제기능 관리 | 한혜경 |
| 12) | | 모듈 로컬파일 설정 | 한혜경 |
| 13) | | 메인 컨트롤러 측정주기 변경 | 이우석 |

<표 2.12> 지능형 주차관제 모니터링 시스템 (관리자) 기능 정의 표

1) 로그인
- ■ 관리자는 로그인을 통해 관리자 페이지에 연결된다.

2) 로그아웃
- ■ 관리자는 로그아웃을 통해 관리자 페이지 연결이 끊긴다.

3) 관리자 정보수정
- ■ 관리자는 비밀번호, 전화번호 등의 정보를 수정할 수 있다.

4) 전체 주차상황 파악
- ■ 관리자는 통계자료로 스마트 주차장의 전체 주차상황을 파악할 수 있다.
- ■ 전체 주차상황은 실시간으로 DB에서 업데이트된다.

5) 구역별 주차상황 파악
- ■ 관리자는 통계자료로 스마트 주차장의 구역별 주차상황을 파악할 수 있다.
- ■ 구역별 주차상황은 실시간으로 DB에서 업데이트된다.

6) 기간별 주차량 분석
- 관리자는 통계자료로 스마트 주차장의 기간별 주차량을 분석할 수 있다.
- 기간별 주차량은 정기적으로 DB에서 업데이트된다.

7) 분기별 매출 확인
- 관리자는 통계자료로 스마트 주차장의 분기별 매출을 확인할 수 있다.
- 분기별 매출은 정기적으로 DB에서 업데이트된다.

8) 모듈 초기화
- 스마트 주차장의 원활한 진행을 위해 모듈은 초기화된다.
- 스마트 주차장에 문제가 생기는 경우, 모듈은 한 번에 초기화될 수 있다.

9) 응급상황 모듈 작동
- 관리자는 스마트 주차장의 상황에 따라 응급상황 모듈을 작동할 수 있다.

10) 스마트 주차장 관리
- 관리자는 스마트 주차장 통계자료 및 모듈 로그를 통해 주차장을 관리한다.
- 관리자는 필요에 따라 모듈 전체를 초기화할 수 있다.

11) 결제기능 관리
- 관리자는 주차 요금에 모든 권한을 가진다.
- 관리자는 주차 요금 산정 방식을 변경할 수 있다.

12) 모듈 로컬파일 설정
- 관리자는 모듈 로컬파일로 모듈의 작동을 제어한다.

13) 메인 컨트롤러 측정주기 변경
- 관리자는 메인 컨트롤러 제어 권한을 가진다.
- 관리자는 필요에 따라 메인 컨트롤러의 측정주기를 변경할 수 있다.

## 2.5. 소프트웨어 요구사항 정의 (번호판 인식 AI server)

| 번호 | 주요객체 | 상세기능 | 개발담당자 |
|---|---|---|---|
| 1) | 자동차 번호판 식별 | 자동차 번호판 인식 | 이우석 |
| 2) | | 자동차 번호판 분석 | 이우석 |
| 3) | 서버 송수신 | 자동차 번호판 식별 응답 | 한혜경 |
| 4) | | 데이터베이스 저장 | 이다은 |

<표 2.13> 지능형 주차관제 모니터링 시스템 (번호판 인식 AI) 기능 정의 표

1) 자동차 번호판 인식
- ■ AI 학습 모델을 기반으로 요청받은 자동차 번호판 데이터를 테스트한다.

2) 자동차 번호판 분석
- ■ AI 학습 모델을 기반으로 요청받은 데이터에서 자동차 번호를 추출한다.

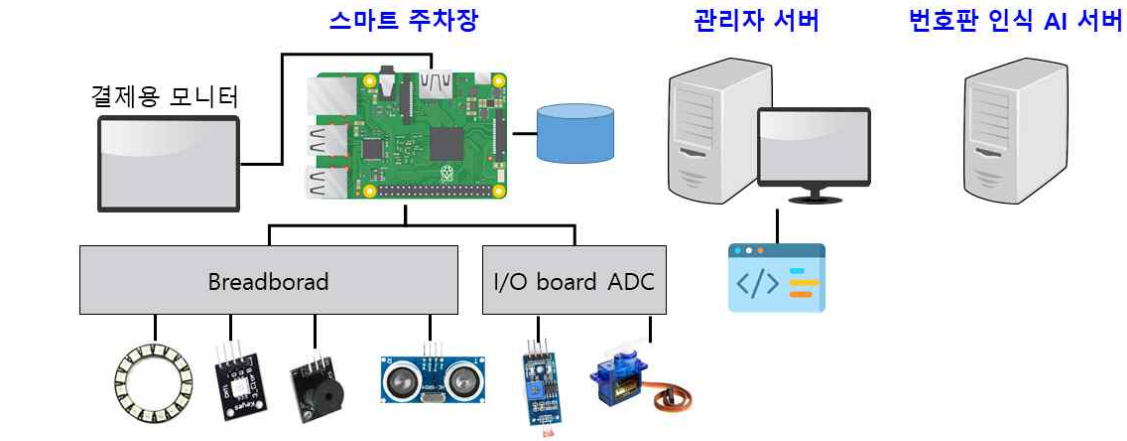
3) 자동차 번호판 식별 응답
- ■ AI 서버에서 스마트 주차장 서버로 최종 식별 결과를 전송한다.

4) 데이터베이스 저장
- ■ AI 서버에서 처리되는 모든 데이터는 DB에 저장된다.
- ■ AI 학습 모델의 모든 변동 사항은 DB에서 업데이트된다.

# 3. 프로세스(기능) 설계

## 3.1. 하드웨어 구성

스마트 주차장 서비스를 개발하기 위해 아래와 같이 하드웨어를 구성한다. 주차장을 구현하기 위한 센서, 액츄레이터 모듈과 관리자 Web UI를 위한 서버, 번호판 인식 AI 서버로 구분할 수 있다.




| | | |
|---|---|---|
| 3색 LED | SIG | GPIO 12, 16, 20, 23, 24, 25 |
| | GND | GND |
| | VCC | 5v |

| | | |
|---|---|---|
| NeoPixel LED | SIG | GPIO 18 |
| | GND | GND |
| | VCC | 5v |

| | | |
|---|---|---|
| 조도 센서 | A0 | ADC 0 Channel 0 ~ADC 0 Channel 7, ADC 1 Channel 0 |
| | GND | GND |
| | VCC | 3.3v |

| | | |
|---|---|---|
| 초음파 센서 | TRIG | GPIO 17, 27, 22, 5 |
| | ECHO | GPIO 6, 13, 19, 26 |
| | GND | GND |
| | VCC | 5v |

| | | |
|---|---|---|
| 부저 | SIG | GPIO 4 |
| | GND | GND |
| | VCC | 5v |

| | | |
|---|---|---|
| 서보 모터 | SIG | ADC 1 Channel 1~ADC 1 Channel 6 |
| | GND | GND |
| | VCC | 5v |

<그림 3.1> 하드웨어 구성도

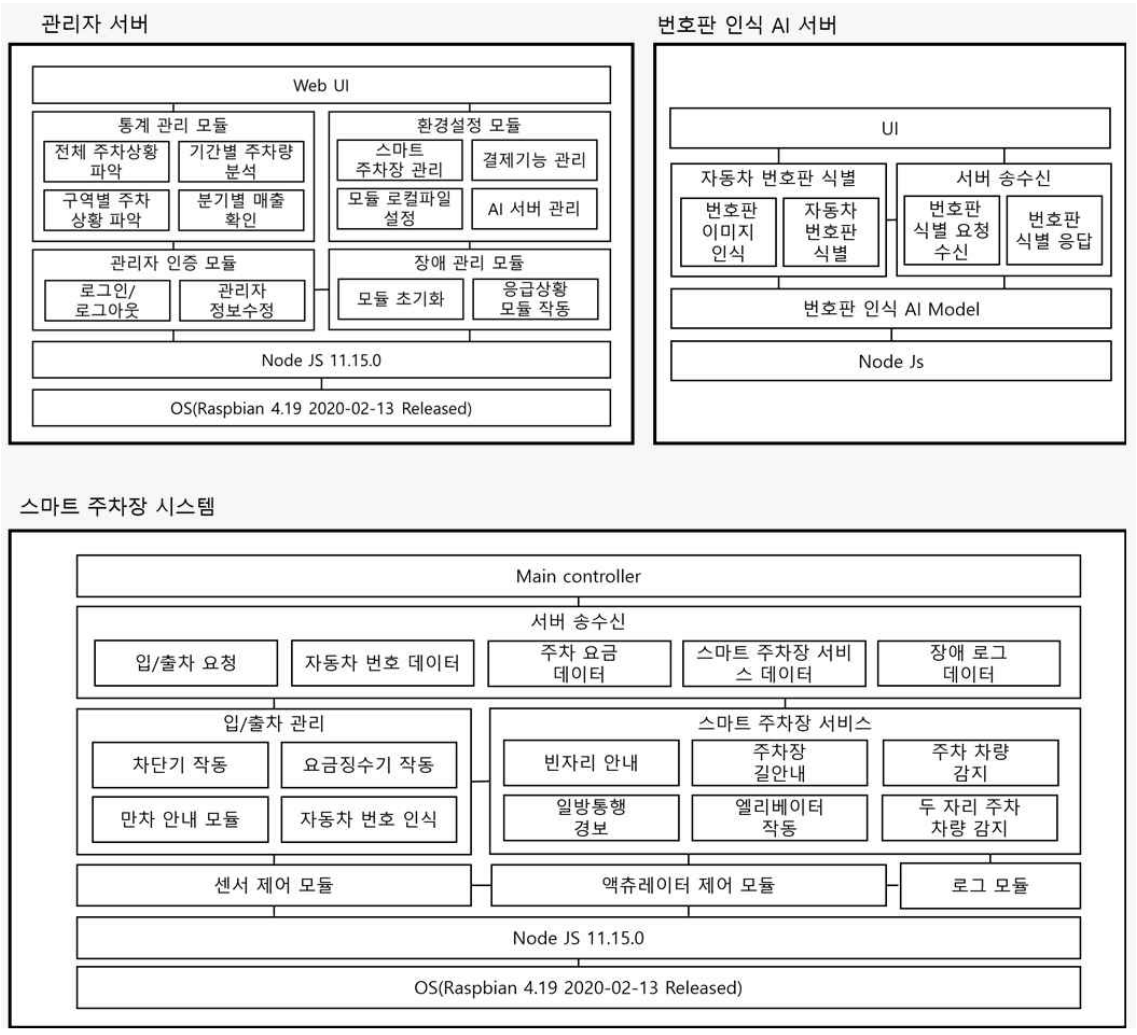
그림 3.1을 보면 스마트 주차장은 라즈베리파이와 다양한 센서, 액츄레이터 모듈로 구성된다. 스마트 주차장은 관리자 서버를 통해 웹으로 관리자에게 UI를 제공한다. 번호판 인식 AI 서버는 번호판 이미지를 등록하면 인공지능 서버에서 이미지를 분석하여 결과를 반환한다.

스마트 주차장에는 다양한 센서와 액츄레이터가 연결되어 있다. 주차장 서비스를 제공하면서 측정한 정보를 데이터베이스에 저장하여 관리자가 해당 정보를 Web UI로 열어볼 수 있다.

서보모터 입출차를 제한하는 차단기 역할을 수행하고 조도센서는 자동차가 특정 위치에 접근해서 빛이 가려지는지 판단한다. 초음파 센서는 전방 사물의 거리를 파악하여 일방통행을 무시하고 다가오는 차량을 감지한다. LED 모듈을 이용하여 사용자에게 주차 가능 여부를 간단하게 안내할 수 있다.

## 3.2. 소프트웨어 구성

작성한 요구사항을 만족하는 스마트 주차장 시스템을 개발하기 위해 아래와 같이 소프트웨어 구성도를 작성한다.




<그림 3.2> 전체 시스템 소프트웨어 구성도

그림 3.2은 전체 시스템에 대한 소프트웨어 구성도이다. 스마트 주차장 시스템은 하드웨어 센서 제어와 액츄레이터 제어를 기반으로 다양한 주차장 서비스 기능을 제공한다. 사용자 조작을 최소화하기 위해 번호판을 AI로 인식하기 위해 AI 서버와 송수신한다. 주차장에서 측정하고 분석한 내용을 데이터베이스에 저장하여 관리자 서버에 제공한다.

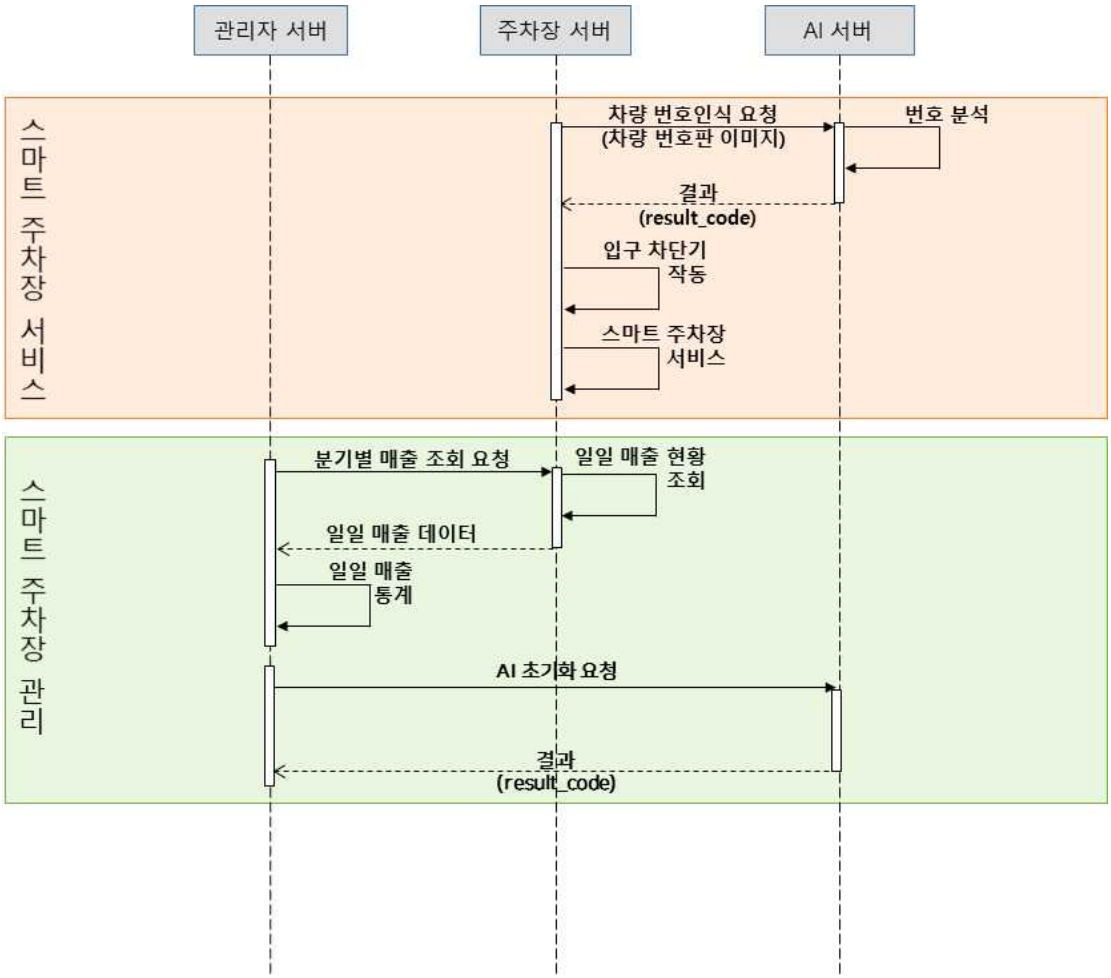
관리자 서버는 관리자에게 주차장의 상황을 Web UI를 통해 효과적으로 제공한다. 관리자 서버는 주차장 시스템과 Rest API로 데이터를 주고 받으며 정보를 분석하여

시각적으로 제공한다. 관리자 서버는 스마트 주차장 시스템과 AI서버를 관리할 수 있다. 스마트 주차장 서비스 제공 여부를 관리하고 초기화 실행 명령을 보내서 장애 상황에 능동적으로 대처할 수 있다.

번호판 인식 AI 서버는 스마트 주차장 시스템에서 보낸 이미지를 분석하여 번호 정보를 제공한다. 번호판 분석을 진행하는 모듈과 데이터를 가공하여 정해진 양식으로 송수신하는 모듈로 구성된다.

## 3.3. 기기 간 설계 (Sequence Diagram)

기기 간 상호작용을 아래와 같이 설계한다.



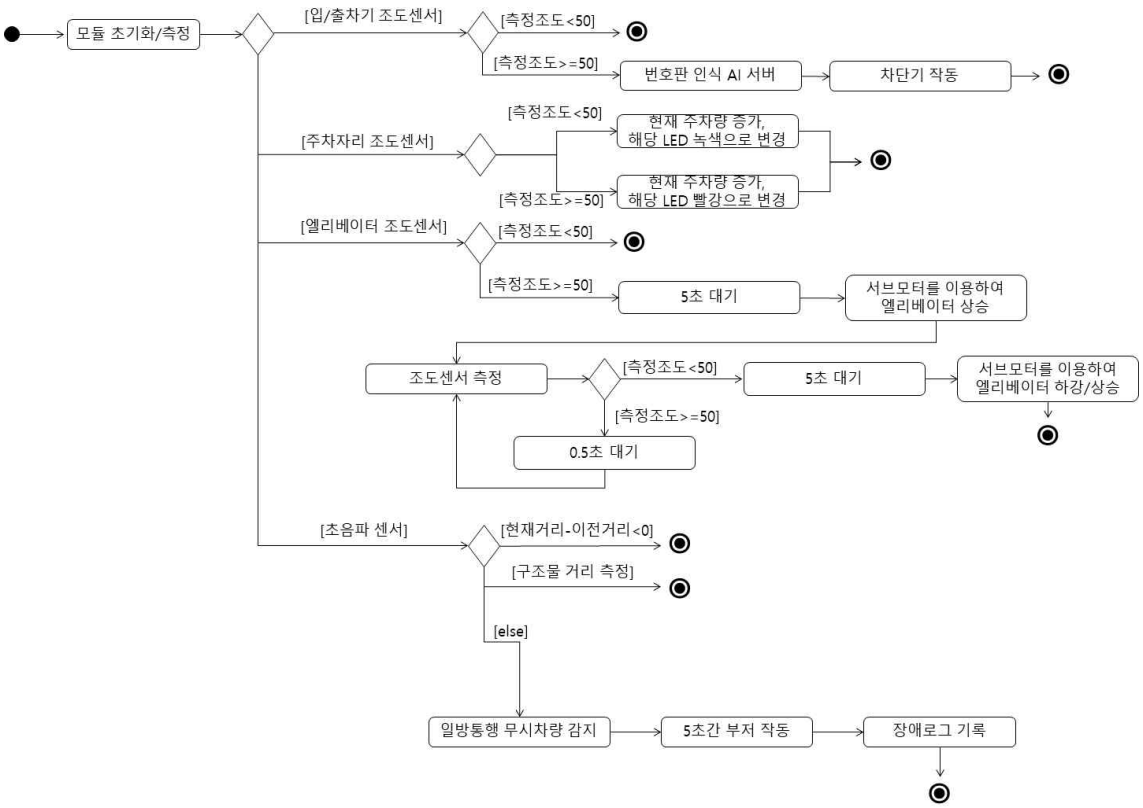

<그림 3.3> 기기 간의 시퀀스 다이어그램

그림 3.3를 보면 각 서버 간에 데이터를 주고받는 과정을 나타낸다. 기기 간 데이터를 주고받는 과정은 크게 스마트 주차장 시스템에서 서비스를 제공하는 과정과 관리자 서버에서 주차장을 관리할 때로 구분된다. 스마트 주차장 서비스가 동작하는 중에는 AI 서버에서 번호판 인식을 요청하고 결과를 받아온다. 관리자 서버는 주차장 시스템에서 데이터를 제공받는다. 관리자 서버는 각각 스마트 주차장과 번호판 인식 AI 서버에 관리 명령을 보내거나 정보를 받아올 수 있다.

## 3.4. 스마트 주차장 (Activity Diagram)

### ■ 스마트 주차장 시스템

스마트 주차장 프로그램에서 자동화 서비스를 제공하기 위해 센서로 데이터를 측정하고 처리하는 부분을 아래와 같이 설계한다.



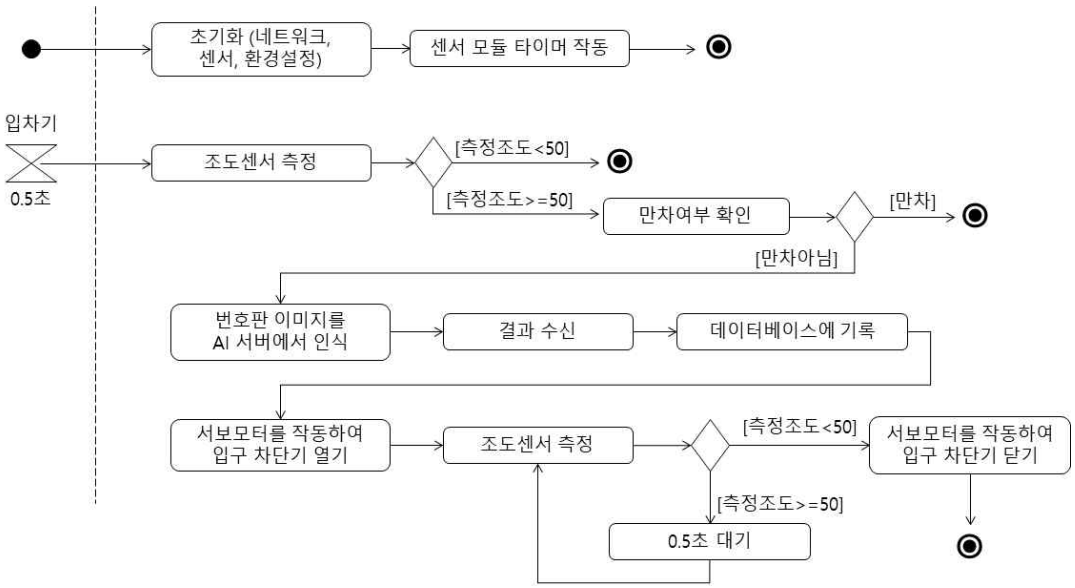

<그림 3.4> 스마트 주차장 시스템 Activity diagram

스마트 주차장은 조도센서와 초음파 센서로 차량을 인식하여 서비스를 제공한다. 그림 3.4는 스마트 주차장 시스템에서 조도센서와 초음파 센서가 설치된 위치에 따라 서비스를 제공하는 간략한 과정이다. 상세 과정은 기능별로 구분하여 그림 3.5부터 그림 3.11에 설계한다.

조도센서는 차량을 인식하여 각종 서비스를 제공한다. 입출차기 앞에 차량이 도착하면 번호판 인식 AI 서버와 통신하여 차량을 인식하고 서비스를 제공한다. 차량이 조도센서로 주차자리에 인식되면 LED를 변경하여 다른 사용자에게 알린다. 엘리베이터 앞에 차량이 대기하는 경우 인식하여 엘리베이터를 동작시킨다. 초음파 센서는 물체와 거리를 감지할 수 있기 때문에 거리를 측정하여 일방통행을 무시하고 다가오는 차량을 감지한다.

■ **초기화 입차**

시스템이 시작되고 자동차가 들어오는 입차 과정을 아래와 같이 설계한다.



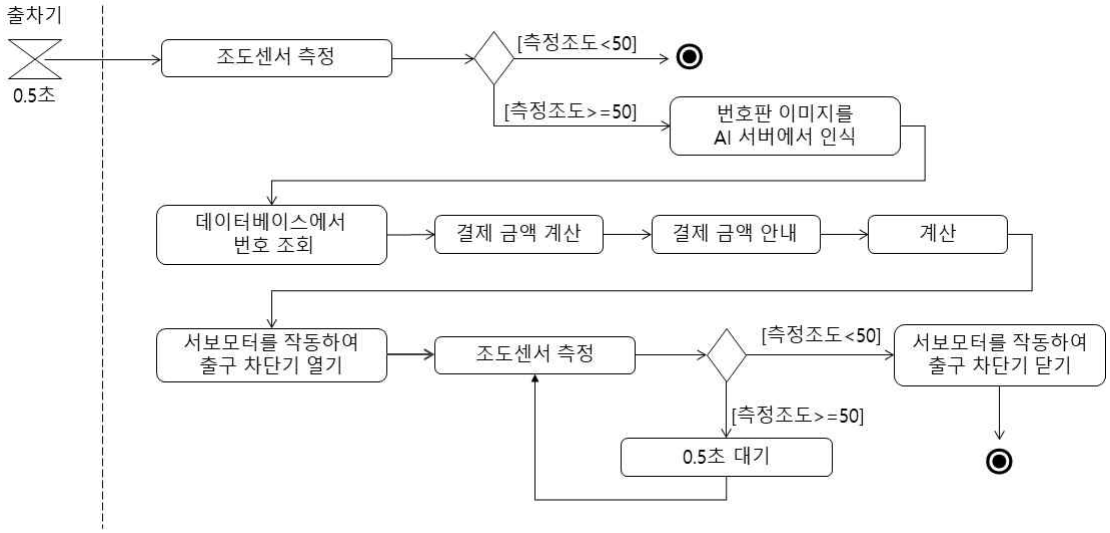

<그림 3.5> 초기화 입차 Activity diagram

그림 3.5를 보면 기기가 부팅되면 네트워크, 센서를 초기화하고 환경설정을 받아온다. 설정에 맞춰 각 서비스의 타이머를 작동시킨다. 입차기는 0.5초마다 조도센서를 측정하여 차가 근접해 가려지는 경우에 입차 과정을 수행한다.

입차기 근처에 차량이 근접하면 입차기는 만차 여부를 조회한다. 주차장이 만차인 경우에는 동작하지 않는다. 주차장에 자리가 있을 경우에 번호판을 인식하고 입장 시각과 번호를 데이터베이스에 저장한다. 서보 모터를 이용해 입구 차단기를 열고 조도센서를 측정하여 차가 지나가면 서보모터를 작동해 차단기를 닫는다.

■ **출차**

주차된 차가 나가는 출차 과정을 아래와 같이 설계한다.

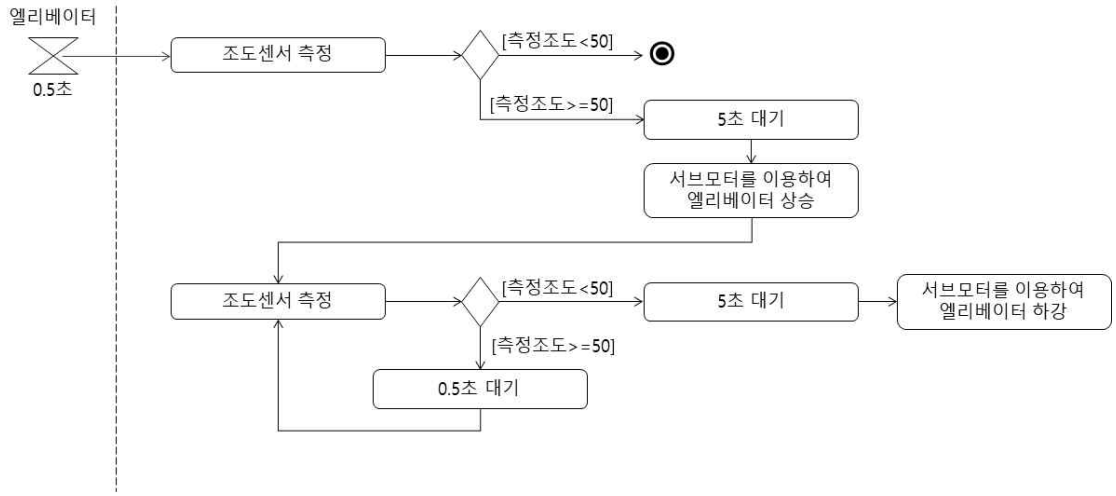



<그림 3.6> 출차 과정 Activity diagram

그림 3.6을 보면 출차기 앞에 설치된 조도센서 위에 차량이 감지되면 동작한다. 번호판을 인식하고 데이터베이스에서 해당 번호의 차량이 들어온 시각을 조회한다. 관리자가 지정한 가격에 따라 요금을 계산하고 나면 서보모터를 이용해 출구 차단기를 연다. 차가 지나가고 나면 서보 모터를 이용해 다시 출구 차단기를 닫는다.

■ **엘리베이터**

2층으로 올라가는 엘리베이터를 구현하기 위해 아래와 같이 설계한다.

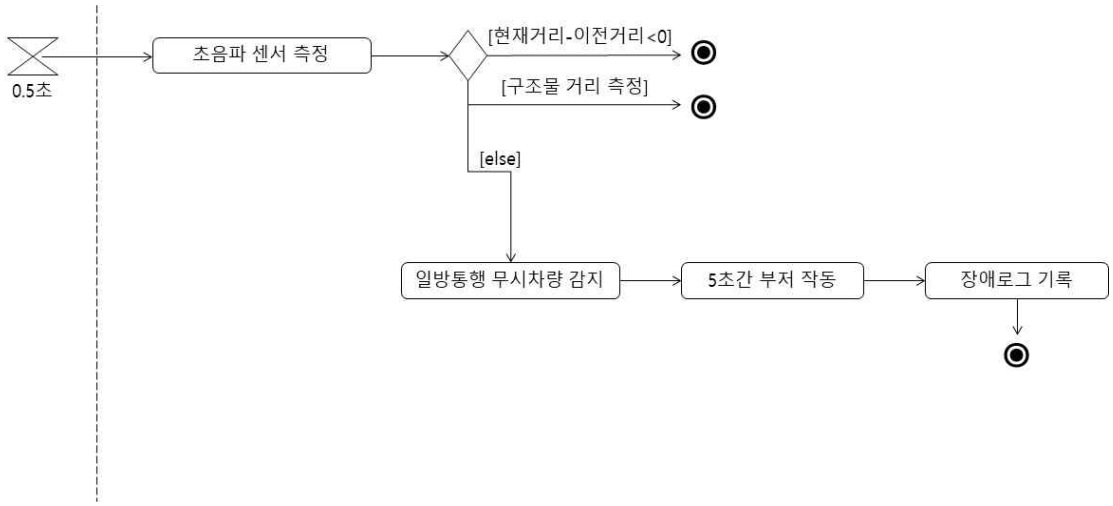



<그림 3.7> 엘리베이터 Activity diagram

그림 3.7을 보면 엘리베이터가 동작하는 과정을 나타내고 있다. 엘리베이터 안에 차량이 감지되면 승강기가 작동하여 차를 2층으로 올려준다. 반대로 2층에서 엘리베이터 앞에 차량이 감지되면 승강기가 작동하여 차를 1층으로 내려갈 수 있게 움직인다. 차량의 감지는 조도센서를 이용한다.

■ **일방통행 감지**

스마트 주차장에서 차량의 일방통행 무시를 감지하기 위해 아래와 같이 설계한다.

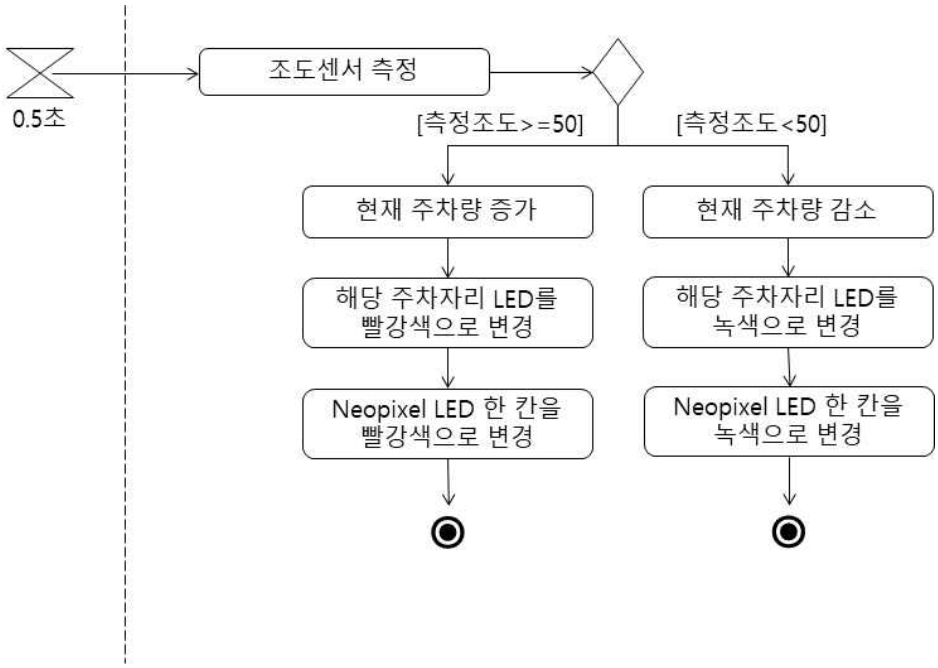



<그림 3.8> 일방통행 감지 Activity diagram

그림 3.8을 보면 초음파 센서를 이용하여 일방통행을 무시하고 지나가는 차를 감지한다. 초음파 센서로 거리를 측정하여 차가 점점 다가오는 경우 일방통행에 맞춰 진행하고 있다고 파악하고 과정을 종료한다. 이동하는 차가 없어 길 끝에 있는 구조물이 측정되는 경우에도 과정을 종료한다. 감지된 차의 거리가 멀어지면 일방통행을 반대로 가고 있다고 파악하여 부저를 작동시킨다. 일방통행 무시 차량의 시간을 장애로그로 남긴다.

■ **주차량 감지**

주차량을 감지하고 안내하는 서비스를 제공하기 위해 아래와 같이 설계한다.

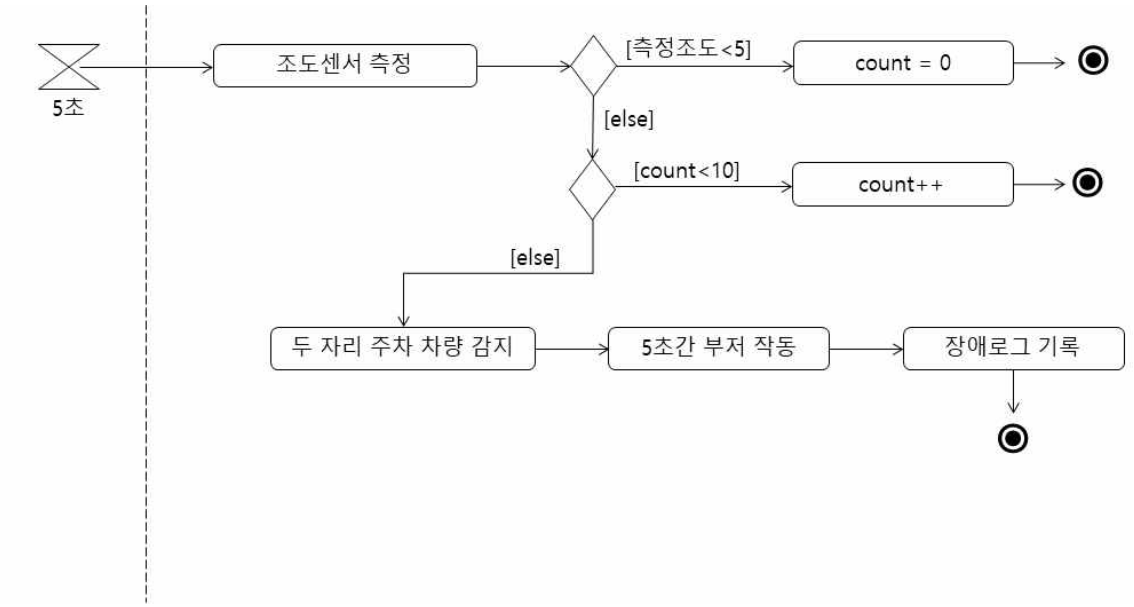



<그림 3.9> 주차량 감지 Activity diagram

그림 3.9을 보면 주차장 자리마다 설치된 조도센서를 측정한다. 조도센서 위에 차가 자리하여 주차된 경우 현재 자리를 주차된 자리로 변경한다. 주차 자리에 같이 부착된 LED를 빨간색으로 변경하고 주차장 밖에 설치된 NeoPixel LED에도 한 칸을 빨간색으로 변경한다. 조도센서에서 차량이 없다고 감지하는 경우 반대로 현재 자리의 LED를 녹색으로 변경하고 NeoPixel도 녹색 불로 유지한다.

■ **두 자리 주차 차량 감지**

두 자리에 주차하는 차량을 감지하기 위해 아래와 같이 설계한다.

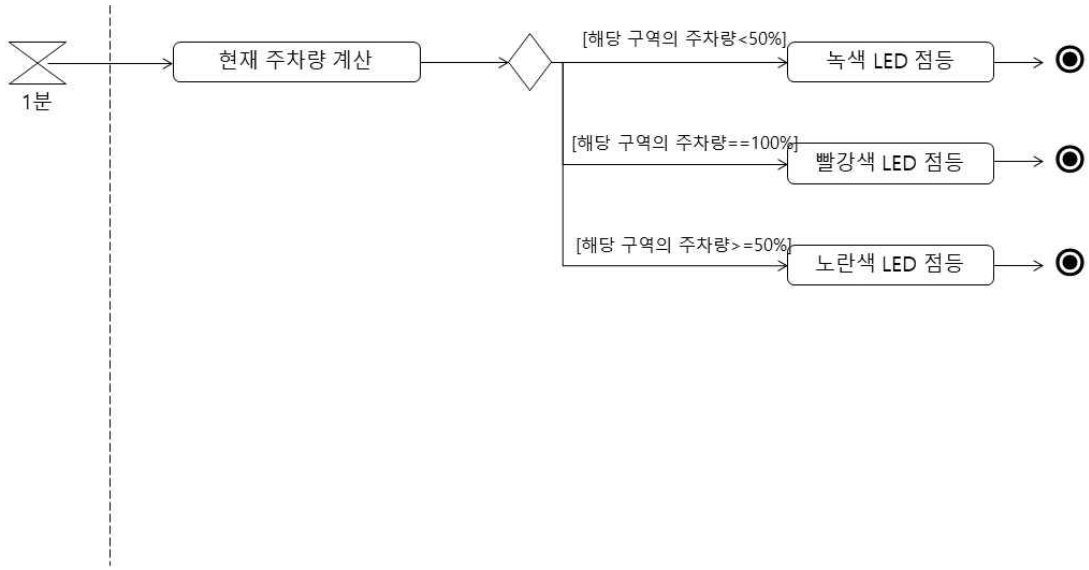



<그림 3.10> 두 자리 주차 차량 감지 Activity diagram

그림 3.10를 보면 주차 자리를 침범한 차량을 감지하기 위한 과정이다. 주차 선 위에 레이저 포인터와 조도 센서를 배치한다. 차량이 레이저 포인터를 가려 다른 값이 측정되면 조도센서는 카운트를 시작한다. 주차하는 과정에서 일시적으로 발생한 게 아니라 50초 이상 유지되는 경우 자리를 침범하여 주차한 것으로 판단한다. 장애 상황에서 부저를 작동시키고 장애 로그로 기록한다.

■ **길안내**

빈 자리를 안내하기 위해 아래와 같이 설계한다.



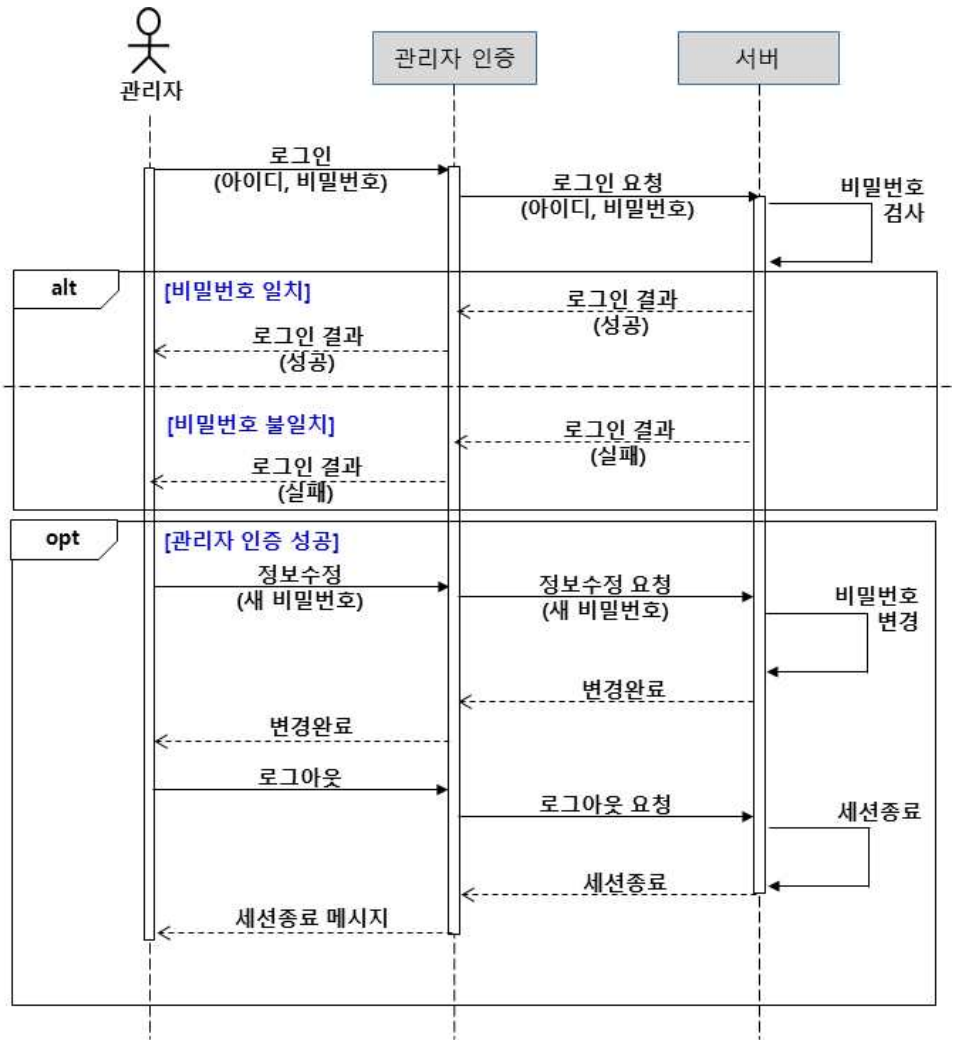

<그림 3.11> 길안내 Activity diagram

그림 3.11을 보면 현재 주차량과 해당 구역의 주차량을 계산하여 길안내를 수행한다. 갈림길, 주차장의 꼭지점에 설치한 LED 모듈은 주변 주차량을 인식하여 각각 다른 색으로 LED를 점등한다. 주차량이 반 이하인 경우에는 원활하다는 뜻으로 녹색 LED를 점등하고 주차 자리가 아예 없으면 빨간색 LED를 켠다. 그 사이의 경우에는 빨간색과 녹색을 둘 다 점등하여 노란색 LED를 점등한다.

## 3.5. 관리자 서버 (Sequence Diagram)

■ 관리자 인증

관리자 인증 과정을 구현하기 위해 아래와 같이 설계한다.

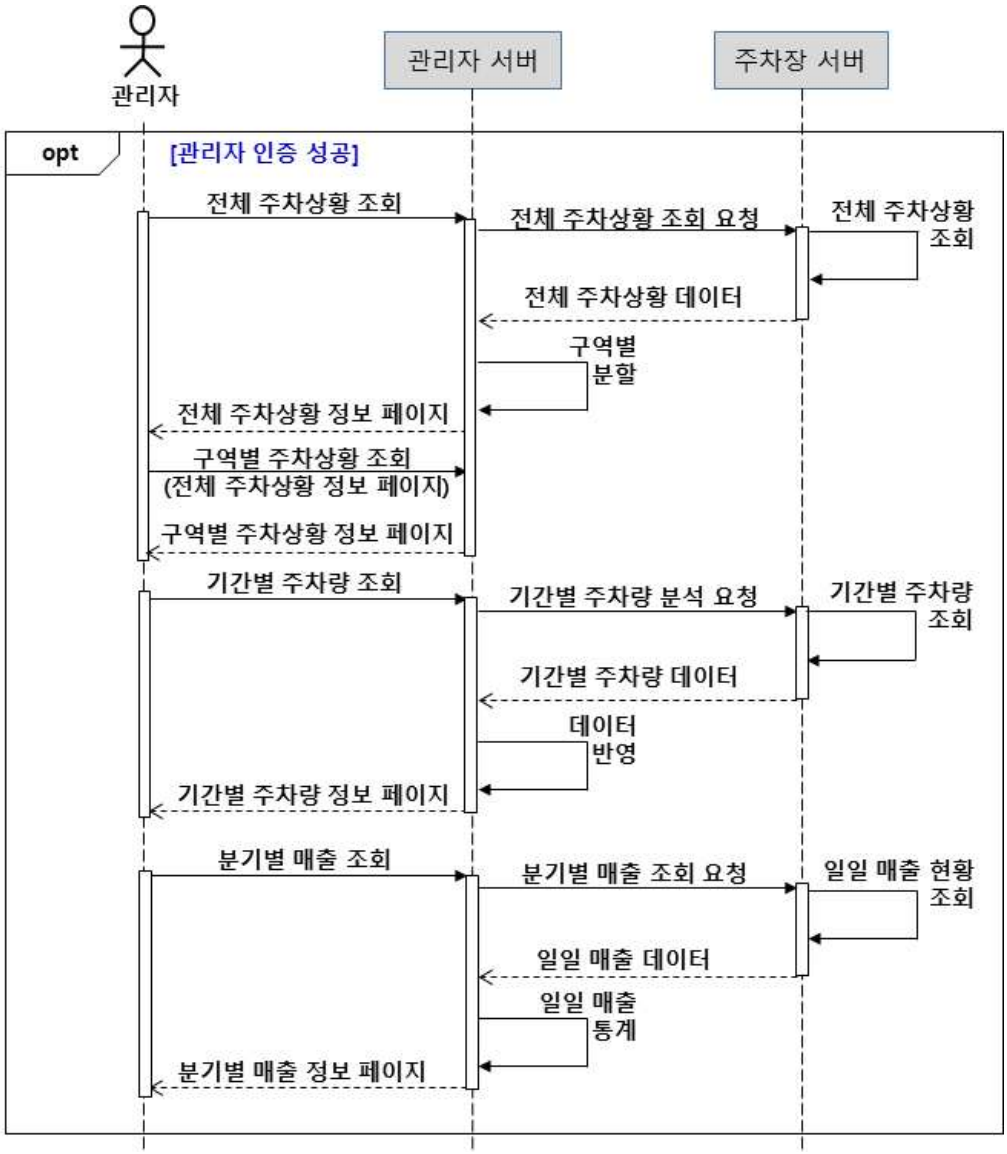



<그림 3.12> 관리자 인증 Sequence diagram

그림 3.12를 보면 관리자는 관리자 서버를 이용하기 위해서 관리자 로그인을 수행해야 한다. 관리자는 아이디와 비밀번호를 입력하여 관리자 서버에 로그인을 요청할 수 있다. 관리자는 로그인으로 관리자 인증을 성공한 다음 관리자 기능을 이용할 수 있다. 관리자는 관리자 인증에 성공한 다음 관리자 비밀번호를 변경할 수 있다. 새로 사용할 비밀번호를 입력하면 서버에서는 새로운 비밀번호를 저장하여 새 비밀번호로 로그인 할 수 있도록 지원한다. 로그아웃을 이용해 관리자 인증 정보가 담긴 세션을 종료할 수 있다.

■ **통계 관리**

스마트 주차장 정보를 제공하기 위해 아래와 같이 설계한다.

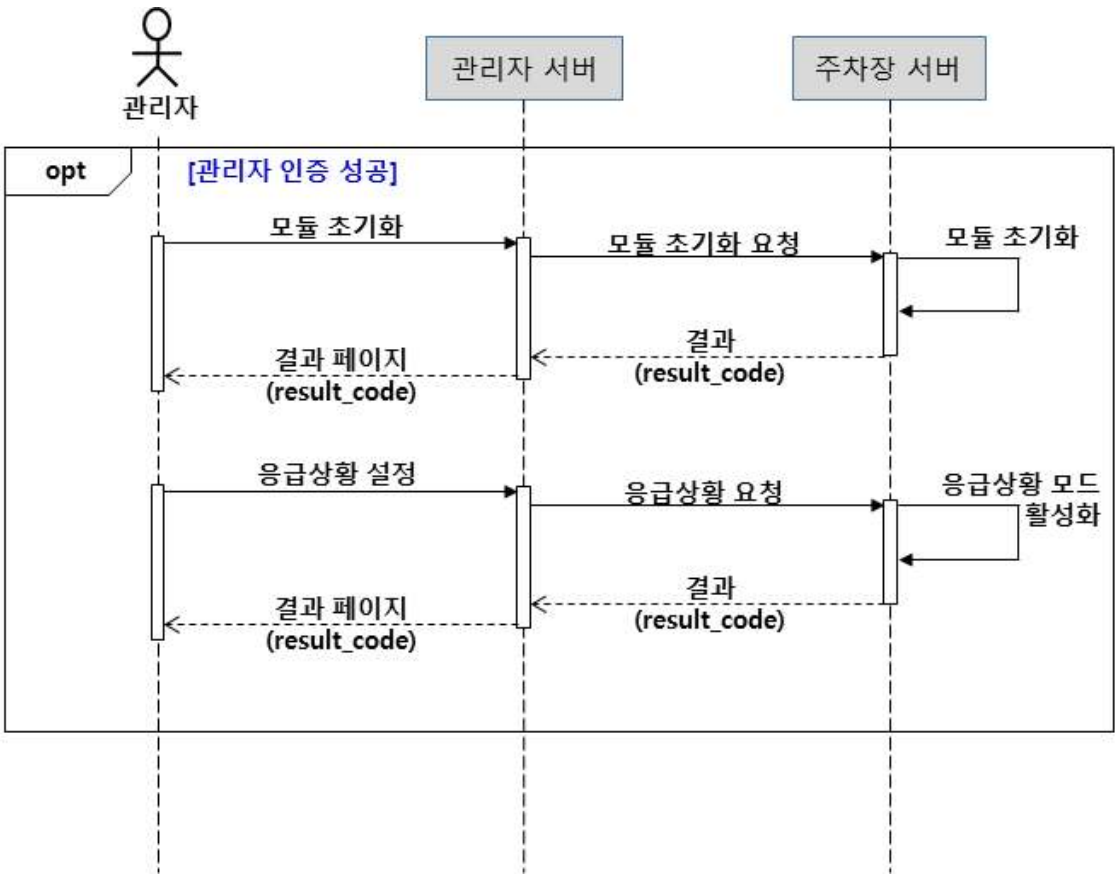



<그림 3.13> 통계관리 Sequence diagram

그림 3.13을 보면 관리자는 통계 관리 기능을 사용하기 위해 관리자 인증에 성공해야 한다. 서버는 관리자 요청에 따라 관리자 서버는 주차장 시스템 데이터베이스에 있는 정보를 받아온다. 받아온 정보를 가공하여 웹으로 관리자에게 제공한다.

■ 장애 관리

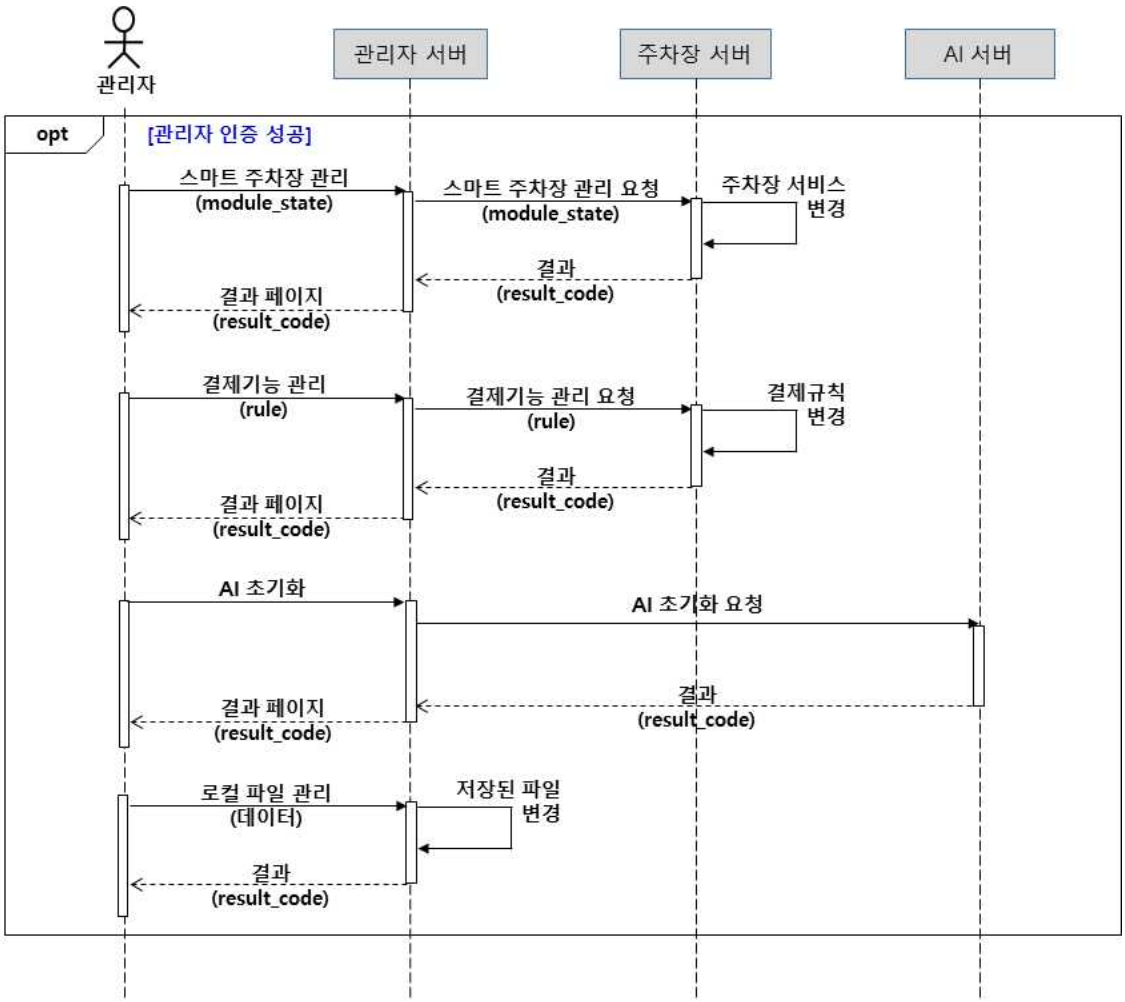
관리자가 주차장 서버를 관리하기 위해 아래와 같이 설계한다.




<그림 3.14> 장애 관리 Sequence diagram

그림 3.14를 보면 장애가 발생한 상황에서 관리자가 수동으로 주차장 시스템에 간섭하기 위해 위와 같은 기능을 제공한다. 관리자는 스마트 주차장의 각 센서 모듈을 초기화할 수 있다. 또 응급한 상황에 부저를 울려서 응급 상황을 주차장에 알릴 수 있다.

■ 환경 설정

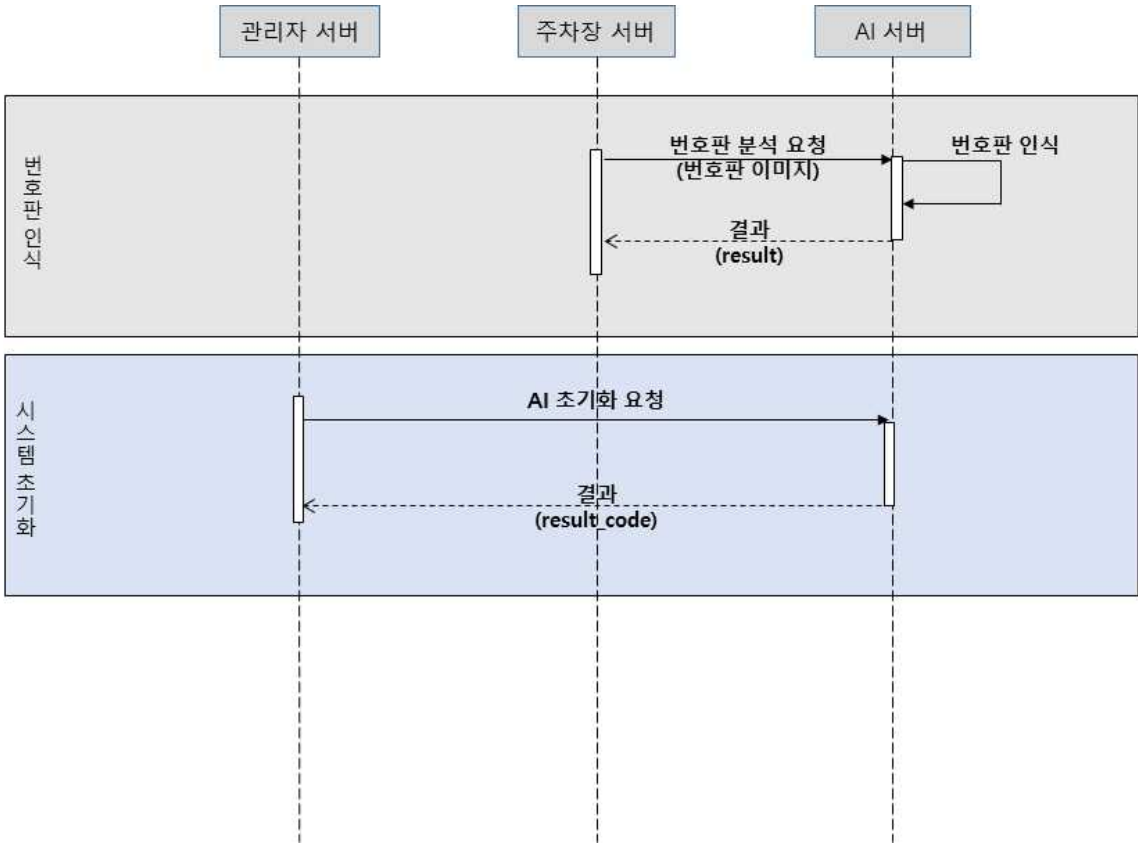



<그림 3.15> 환경 설정 Sequence diagram

그림 3.15를 보면 관리자는 관리자 서버에 등록된 내용을 비롯하여 주차장 서버, AI 서버의 설정을 변경할 수 있다. 관리자는 관리자 서버를 통해 주차장에서 제공하는 서비스와 결제 가격을 변경한다. AI 서버에도 문제가 발생했을 때 초기화 재시작 명령을 보내서 결과를 확인 가능하다.

## 3.6. 번호판 인식 AI 서버 (Sequence Diagram)

번호판 인식 AI 서버를 개발하기 위해 아래와 같이 설계한다.




<그림 3.16> 번호판 인식 AI 서버 Sequence diagram

그림 3.16을 보면 번호판 인식과 관리자 서버의 명령을 받아 시스템을 초기화 재시작하는 기능을 제공한다. 주차장 서버는 자동차 번호판 이미지를 AI 서버로 전달하여 결과로 자동차 번호를 받는다. 번호판 인식 AI는 모델 함수 형태로 서버에서 호출하여 판독한다. 또 관리자의 명령을 받으면 서버를 재시작하고 결과를 반환한다.

# 4. 화면(UI) 설계

## 4.1. 스토리보드(메뉴) 구성

### ■ 사용자

사용자(키오스크)


<그림 4.1> 사용자용 메뉴구조도

### ■ 관리자

관리자(웹)


<그림 4.2> 관리자용 메뉴 구조도

## 4.2. 관리자 웹 페이지 초기화면 설계


- 사용자 키오스크의 대기 화면이다.
- 차량이 가까이 다가올 때까지 대기한다.


- 관리자 웹페이지의 초기 화면이다.
- 아이디와 패스워드를 입력하여 로그인 한다.

## 4.3. 사용자 UI 설계

■ 결제 요구 화면

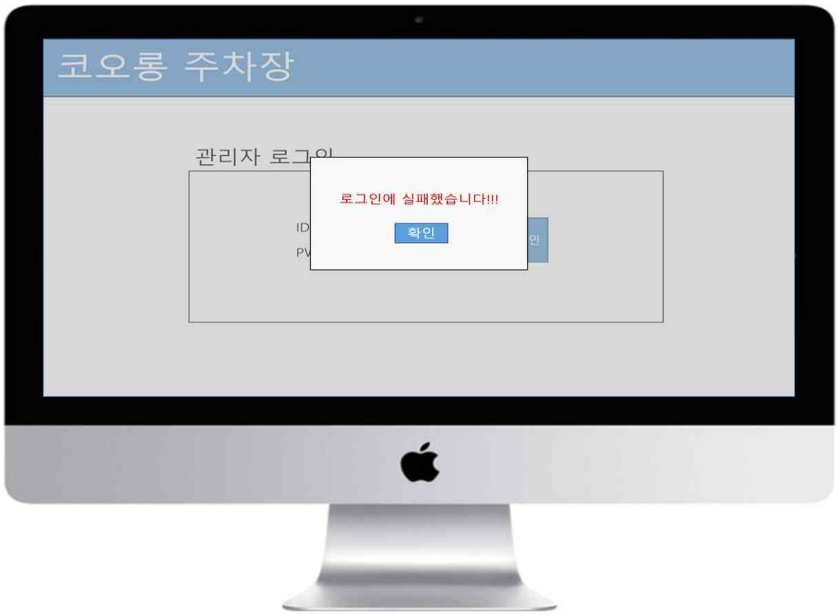

- 차량이 인식기에 다가오면 자동차 번호를 인식 후 체류 시간을 계산한다.
- 요금 결제 요구 화면을 띄워준다.

■ 결제 완료 화면


- 사용자가 결제를 완료하고 난 후 화면이다.

## 4.4. 관리자 UI 설계

■ 로그인 실패 화면




- 관리자가 로그인 실패했을 때 화면이다.

■ 기간별 조회 화면


- 관리자가 로그인했을 때 가장 처음에 보이는 화면이다.
- 기간별로 매출액, 사용률 등을 조회할 수 있다.

■ 시간 당 요금 수정 화면


- 관리자가 시간 당 요금을 수정 할 수 있는 화면이다.

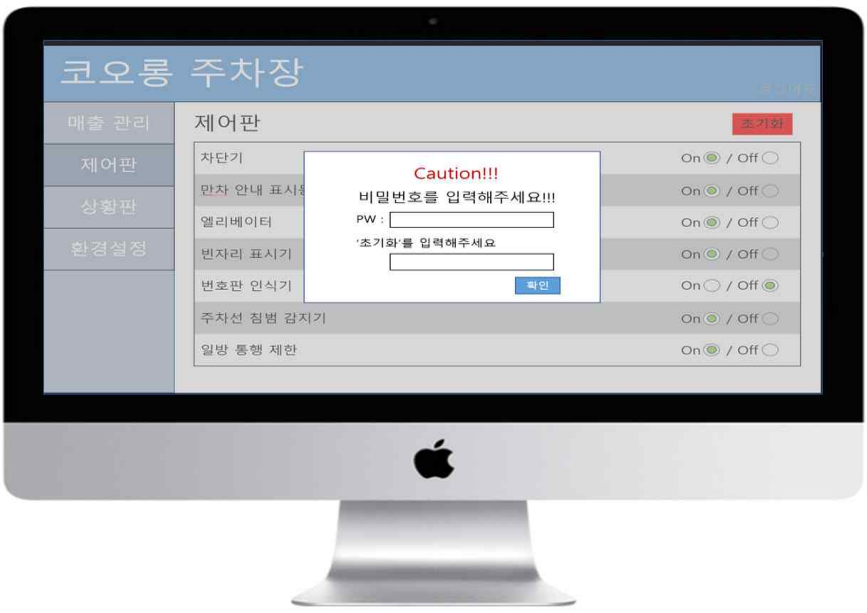
■ 요금 수정 확인 화면


- 요금을 수정하기 전 다시한 번 관리자 인증을 받는 화면이다.

■ 주차장 제어판


- 주차장의 각종 기능들을 제어(On/Off) 할 수 있는 화면이다.

■ 초기화 확인 화면




- 주차장의 모든 기능들을 초기화 하기 전에 관리자 재인증을 받는 화면이다.

■ 주차장 상황판


- 주차장의 현 상태를 볼 수 있다.
- 빈자리 여부, 만차 여부를 알 수 있다.

■ 관리자 비밀번호 변경 화면

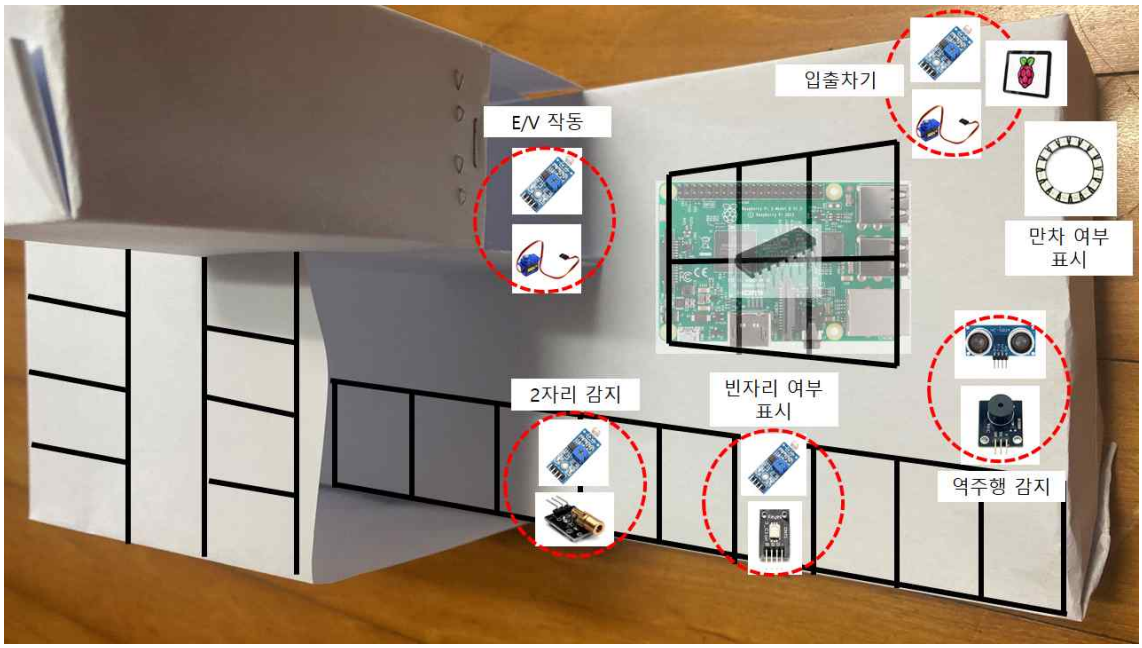

- 관리자의 비밀번호를 변경 할 수 있는 화면이다.

■ 측정 주기 변경 화면


- 각 기능 별 센서들의 측정주기를 변경 할 수 있는 화면이다.

## 4.5. 기기 외관 설계

스마트 주차장의 전체적인 외관은 다음과 같다.

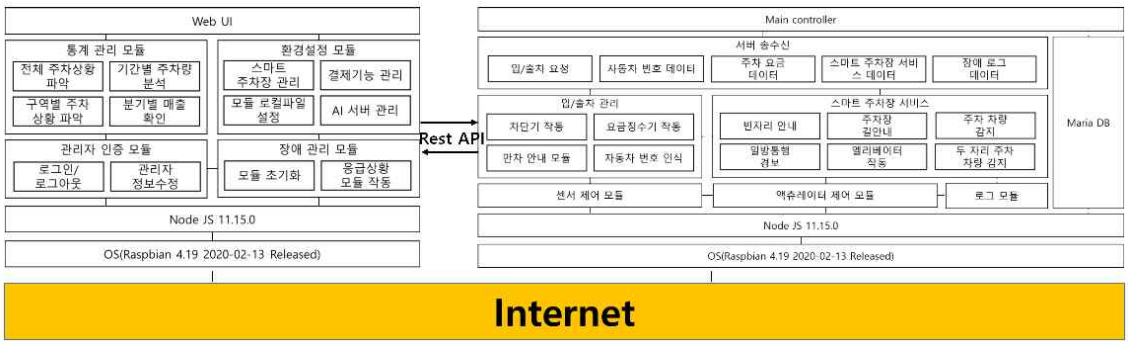



<그림 4.3> 스마트 주차장 외관 구성도

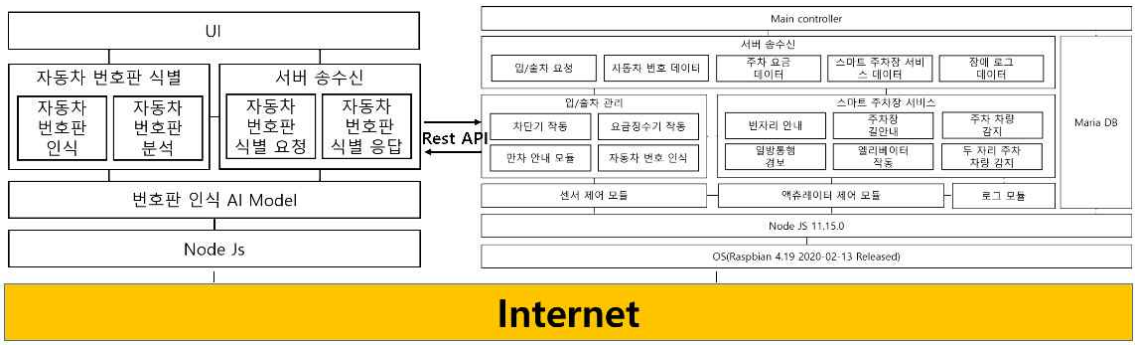
그림 4.3은 스마트 주차장의 전체적인 외관 구성도이다. 가로는 약 60cm * 세로 약 30 cm * 높이 약 30cm로 구성한다.

# 5. REST API 인터페이스 설계

## 5.1. 서버/클라이언트 구조




<그림 5.1> 전송 인터페이스 구조




<그림 5.2> 전송 인터페이스 구조

본 설계는 그림 5.1과 그림 5.2에서처럼 서버와 클라이언트간에 메시지를 전송하여 필요한 정보를 송수신하는 연동용 인터페이스 전송 구조이다. 즉, 클라이언트가 서버에게 정보를 요청하거나 명령을 내리는 방식으로 RESTful API 방식을 이용하여 설계한다.

클라이언트가 서버에게 메시지를 요청하는 방식은 크게 4가지로 구분한다. 정보조회(GET), 생성(POST), 수정 또는 변경(PUT), 삭제(DELETE)가 그것이며, 각 요청별로 REST API 방식으로 메시지를 설계한다. 본 시스템은 다음과 같 서버별로 연동하기 때문에 각 연동별로 구분하여 REST API는 정의하고 설계된다.

- 관리자 server - 주차장 관리 server와 연동

- 주차장 관리 server - 번호판 식별 AI server와 연동
- 주차장 관리 server - 관리자 server와 연동
- 번호판 식별 AI server - 주차장 관리 server와 연동

다음에서 각 연동별로 REST API를 정의하고 설계한다.

## 5.2. REST API 정의 및 설계 (관리자 server)

○ REST API 정의

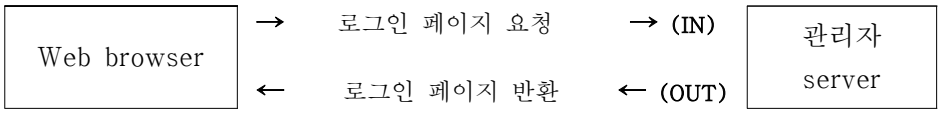
| Method | URI | Description |
|---|---|---|
| GET | /admin/login | 로그인 페이지 요청 |
| | /admin/main | 메인화면 페이지 요청 |
| | /admin/parkAnalysis/index | 기간별 주차량 분석 화면 요청 |
| | /admin/sales/index | 매출 확인 화면 요청 |
| | /admin/skils/index | 주차장 기능 초기화 화면 요청 |
| | /admin/skils/emergency | 응급 상황 요청 |
| | /admin/skils/init | 주차장 기능 초기화 요청 |
| | /admin/config/index | 환경 설정 화면 요청 |
| | /admin/config/adminConfig | 관리자 정보 수정 화면 요청 |
| | /admin/config/payment/index | 결제 기능 관리 페이지 요청 |
| | /admin/config/aiInit/index | AI 서버 초기화 화면 요청 |
| | /admin/config/parkInit/index | 주차장 설정 화면 요청 |
| | /admin/config/timer | 주차장 서비스 주기 설정 화면 요청 |

| Method | URI | Description |
|---|---|---|
| POST | /admin/login | 로그인 요청 |
| | /admin/logout | 로그아웃 요청 |
| | /admin/config/aiInit/index | AI 서버 초기화 |
| | /admin/config/parkInit/index | 주차장 설정 기능 |
| | /admin/config/timer | 주차장 서비스 주기 설정 기능 |

| Method | URI | Description |
|---|---|---|
| PUT | /admin/config/adminConfig/index | 관리자 정보 수정 |
| | /admin/config/payment/index | 결제 기능 수정 요청 |

○ REST API 상세설계

**GET /admin/login**




**Parameter**

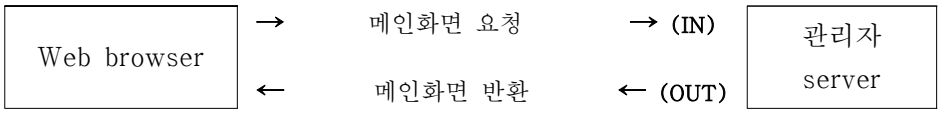
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)**

```
전송방향 : IN
{

 }
전송방향 : OUT
{
   result_code : 200
}
```

GET /admin/main




Parameter

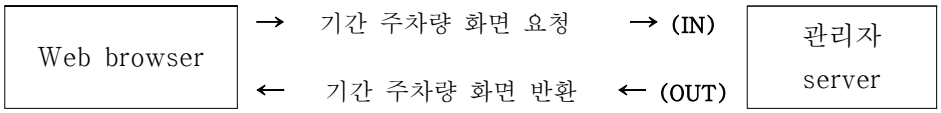
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| park_state | | O | json | 현재 주차장에 주차된 차량 정보와 위치 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   park_state : {
      {
         parking_time : Thu May 16 2019 17:16:13 GMT+0900,
         position : 3
      },
      {
         parking_time : Thu May 16 2019 07:15:31 GMT+0900,
         position : 5
      }
   }
   result_code : 200
}
```

GET /admin/parkAnalysis/index




Parameter

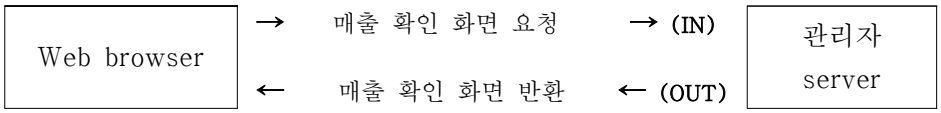
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| park_state | | O | json | 기간별 주차장에 주차된 차량 정보와 위치 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   park_state : {
      {
         parking_start_time : Thu May 16 2019 17:16:13 GMT+0900,
         parking_end_time : Thu May 18 2019 17:16:13 GMT+0900,
         position : 3
      },
      {
         parking_start_time : Thu May 16 2019 07:15:31 GMT+0900,
         parking_end_time : Thu May 16 2019 17:16:13 GMT+0900,
         position : 5
      }
   },
   result_code : 200
}
```

GET /admin/sales/index




Parameter

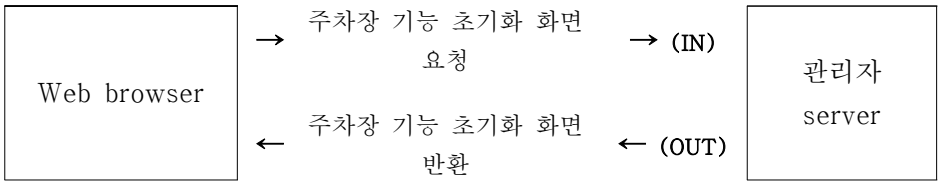
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| sales | | O | json | 일일 매출 현황 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   sales : {
      {
         day : Thu May 16 2019 00:00:00 GMT+0900,
         money : 102900
      },
      {
         day : Thu May 17 2019 00:00:00 GMT+0900,
         money : 101900
      }
   },
   result_code : 200
}
```

GET /admin/skils/index




Parameter

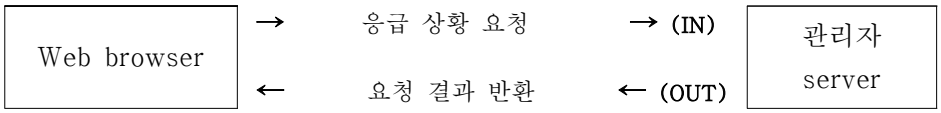
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   result_code : 200
}
```

GET /admin/skils/emergency




Parameter

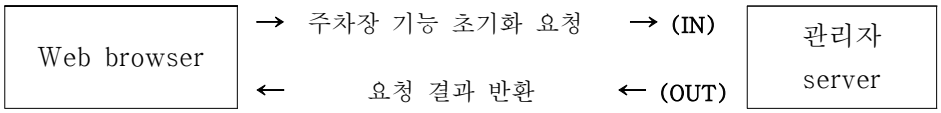
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   result_code : 200
}
```

GET /admin/skils/init




Parameter

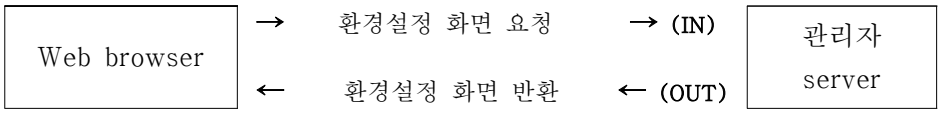
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

 }
전송방향 : OUT
{
   result_code : 200
}
```

GET /admin/config/index




Parameter

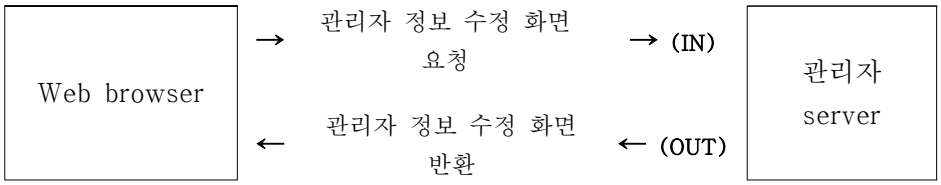
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

 }
전송방향 : OUT
{
  result_code : 200
}
```

GET /admin/config/adminConfig




Parameter

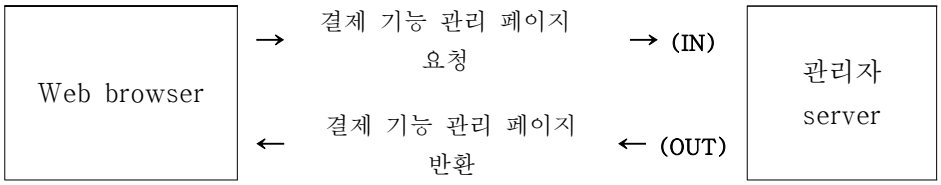
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   result_code : 200
}
```

GET /admin/config/payment/index




Parameter

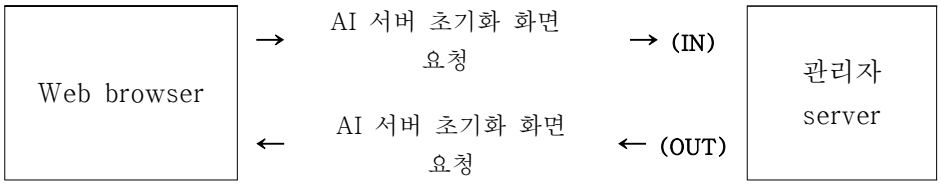
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| rule | | O | json | 현재 요금 산정 방식 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   rule : {
      name : '요금 산정 규칙',
      rule : '주차 시간 * 1500'
   },
   result_code : 200
}
```

GET /admin/config/aiInit/index




Parameter

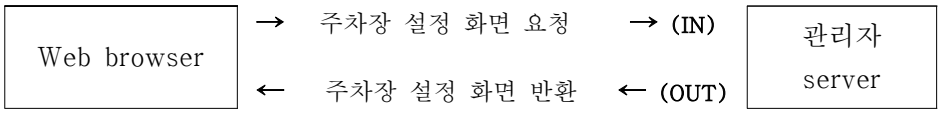
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   result_code : 200
}
```

GET /admin/config/parkInit/index




Parameter

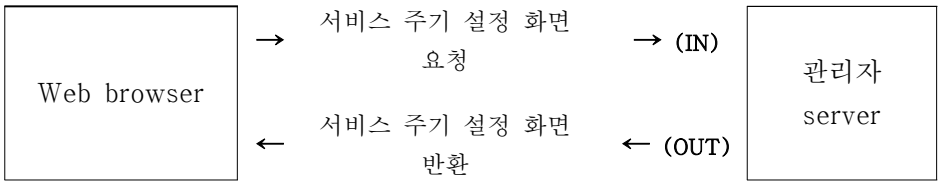
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| module_state | | O | | 현재 제공 기능 on/off 여부 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   module_state : {
      {
         name : '주차장 길 안내 기능',
         status : True
      },
      {
         name : '일방통행 감지 기능',
         status : False
      },
      {
         name : '엘리베이터 설정 기능',
         status : True
      },
   },
   result_code : 200
}
```

GET /admin/config/timer




Parameter

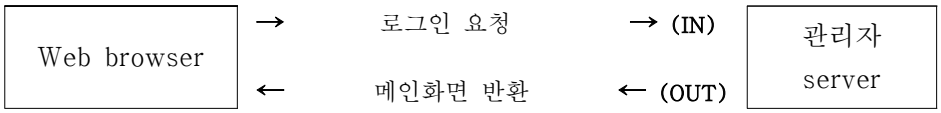
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| timer | | O | int | 주차장 주기값 (기본값: 500) |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   timer : 500,
   result_code : 200
}
```

POST /admin/login



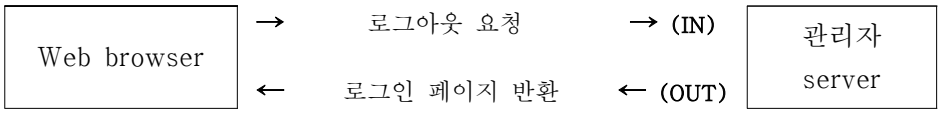

Parameter

| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| admin_id | O | | string | 관리자 id |
| admin_pw | O | | string | 관리자 pw |
| park_state | | O | json | 현재 주차장에 주차된 차량 정보와 위치 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)**

```
전송방향 : IN
{
   admin_id : 'admin',
   admin_pw : 'gachon654321'
}
전송방향 : OUT
{
   park_state : {
      {
         parking_time : Thu May 16 2019 17:16:13 GMT+0900,
         position : 3
      },
      {
         parking_time : Thu May 16 2019 07:15:31 GMT+0900,
         position : 5
      }
   }
   result_code : 200
}
```

POST /admin/logout




Parameter

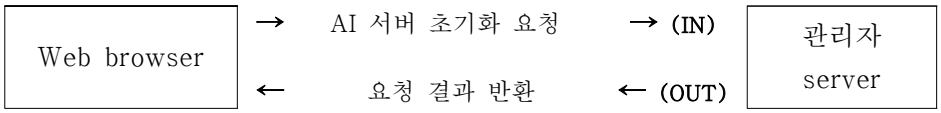
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   result_code : 200
}
```

POST /admin/config/aiInit/index




Parameter

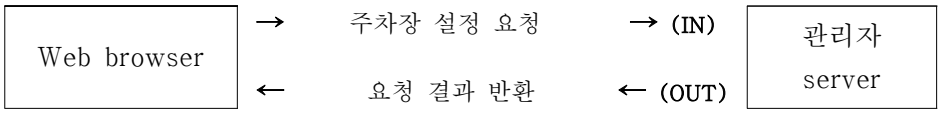
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

 }
전송방향 : OUT
{
   result_code : 200
}
```

POST /admin/config/parkInit/index



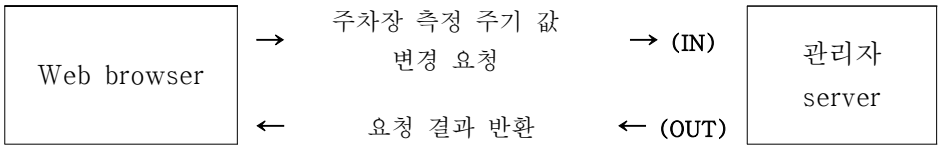

Parameter

| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| module_state | O | | json | 현재 제공 기능 on/off 여부 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{
   module_state : {
      {
         name : '주차장 길 안내 기능',
         status : True
      },
      {
         name : '일방통행 감지 기능',
         status : False
      },
      {
         name : '엘리베이터 설정 기능',
         status : True
      },
   }
 }
전송방향 : OUT
{
   result_code : 200
}
```

POST /admin/config/timer



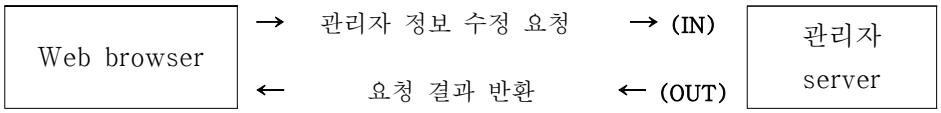

Parameter

| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| module_state | O | | int | 주차장 측정 주기 값 변경 요청 값 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{
   int : 500
 }
전송방향 : OUT
{
   result_code : 200
}
```

PUT /admin/config/adminConfig/index



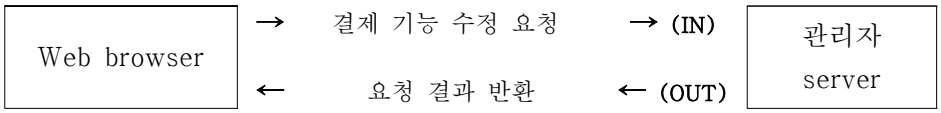

Parameter

| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| admin_id | O | | string | 관리자 id |
| admin_pw | O | | string | 관리자 패스워드 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{
   admin_id : 'admin',
   admin_pw : 'gachon654321'
}
전송방향 : OUT
{
   result_code : 200
}
```

PUT /admin/config/payment/index




Parameter

| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| rule | O | | json | 바꿀 요금 산정 방식 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{
   rule : {
      name : '요금 산정 규칙',
      rule : '주차 시간 * 2500'
   }
}
전송방향 : OUT
{
   result_code : 200
}
```

## 5.3. REST API 정의 및 설계 (스마트 주차장 server)

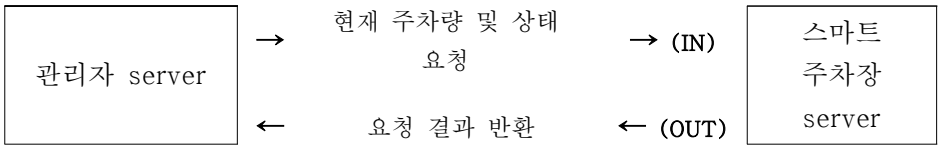
○ REST API 정의

| Method | URI | Description |
|---|---|---|
| GET | /park/nowState | 현재 주차량 및 상태 요청 |
| | /park/dateState | 기간별 주차 정보 요청 |
| | /park/sales | 주차장 매출 정보 |
| | /park/moduleState | 현재 주차장 모듈 상태 요청 |
| | /park/emergency | 응급 상황 발생 요청 |
| | /park/reset | 모듈 초기화 기능 요청 |
| | /park/payment | 주차장 요금 계산 방법 요청 |
| | /park/parkInit | 주차장 설정 값 요청 |
| | /park/timer | 주차장 측정 주기 설정 값 요청 |

| Method | URI | Description |
|---|---|---|
| POST | /park/payment | 주차장 요금 계산 방법 설정 요청 |
| | /park/parkInit | 주차장 모듈 설정 요청 |
| | /park/timer | 주차장 측정 주기 설정 요청 |

○ REST API 상세설계

**GET /park/nowState**




**Parameter**

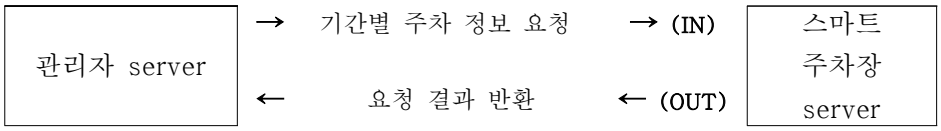
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| park_state | | O | json | 현재 주차장에 주차된 차량 정보와 위치 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)**

```
전송방향 : IN
{

}
전송방향 : OUT
{
   park_state : {
      {
         parking_time : Thu May 16 2019 17:16:13 GMT+0900,
         position : 3
      },
      {
         parking_time : Thu May 16 2019 07:15:31 GMT+0900,
         position : 5
      }
   }.
   result_code : 200
}
```

GET /park/dateState




Parameter

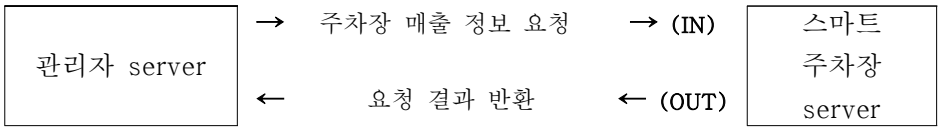
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| park_state | | O | json | 기간별 주차장에 주차된 차량 정보와 위치 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)**

```
전송방향 : IN
{

 }
전송방향 : OUT
{
   park_state : {
      {
         parking_start_time : Thu May 16 2019 17:16:13 GMT+0900,
         parking_end_time : Thu May 18 2019 17:16:13 GMT+0900,
         position : 3
      },
      {
         parking_start_time : Thu May 16 2019 07:15:31 GMT+0900,
         parking_end_time : Thu May 16 2019 17:16:13 GMT+0900,
         position : 5
      }
   }.
   result_code : 200
}
```

GET /park/sales




Parameter

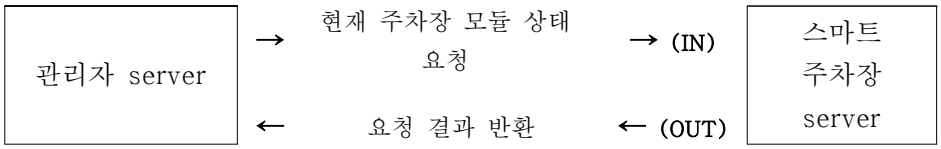
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| sales | | O | json | 일일 매출 현황 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   sales : {
      {
         day : Thu May 16 2019 00:00:00 GMT+0900,
         money : 102900
      },
      {
         day : Thu May 17 2019 00:00:00 GMT+0900,
         money : 101900
      }
   },
   result_code : 200
}
```

GET /park/moduleState




Parameter

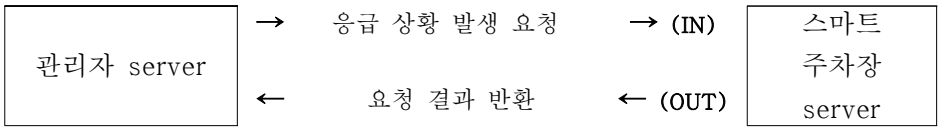
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| module_state | | O | json | 현재 정상 작동 여부 데이터 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)**

```
전송방향 : IN
{

}
전송방향 : OUT
{
   module_state : {
      {
         name : '주차장 길 안내 기능',
         status : True
      },
      {
         name : '일방통행 감지 기능',
         status : False
      },
      {
         name : '엘리베이터 설정 기능',
         status : True
      },
   },
   result_code : 200
}
```

GET /park/emergency




Parameter

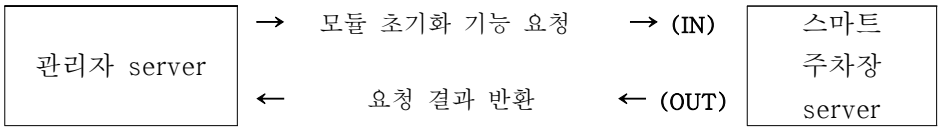
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

 }
전송방향 : OUT
{
   result_code : 200
}
```

GET /park/reset




Parameter

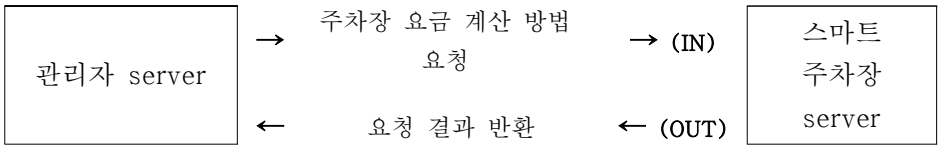
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

 }
전송방향 : OUT
{
   result_code: 200
}
```

GET /park/payment




Parameter

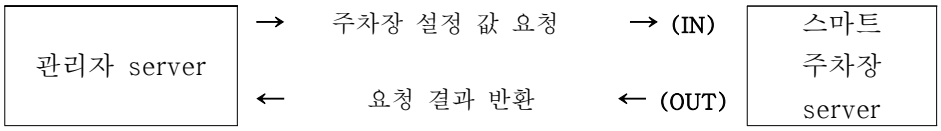
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| rule | | O | json | 현재 요금 산정 방식 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)**

```
전송방향 : IN
{

}
전송방향 : OUT
{
   rule : {
      name : '요금 산정 규칙',
      rule : '주차 시간 * 1500'
   },
   result_code : 200
}
```

GET /park/parkInit




Parameter

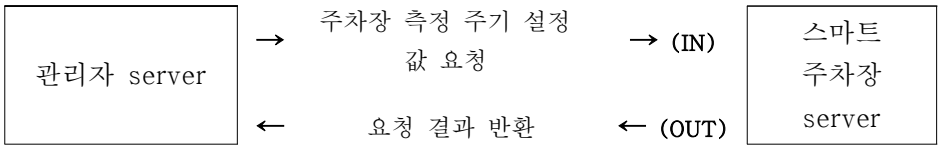
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| module_state | | O | json | 현재 제공 기능 on/off 여부 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)**

```
전송방향 : IN
{

}
전송방향 : OUT
{
   module_state : {
      {
         name : '주차장 길 안내 기능',
         status : True
      },
      {
         name : '일방통행 감지 기능',
         status : False
      },
      {
         name : '엘리베이터 설정 기능',
         status : True
      },
   },
   result_code : 200
}
```

GET /park/timer




Parameter

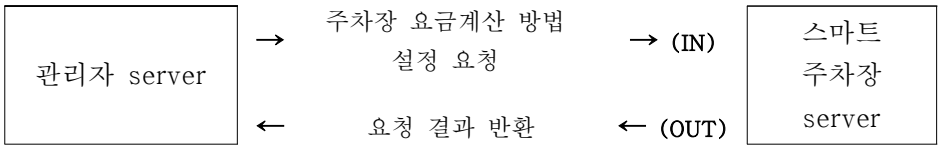
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| timer | | O | int | 주차장 측정 주기 설정 값 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{

}
전송방향 : OUT
{
   timer : 500
   result_code : 200
}
```

POST /park/payment



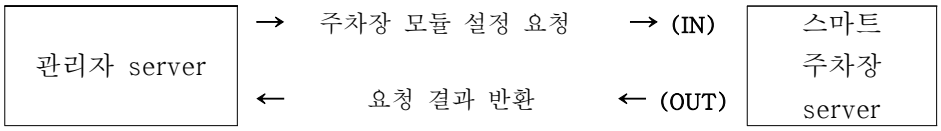

Parameter

| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| rule | O | | json | 바꿀 요금 산정 방식 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)**

```
전송방향 : IN
{
   rule : {
      name : '요금 산정 규칙',
      rule : '주차 시간 * 2500'
   }
}
전송방향 : OUT
{
   result_code : 200
}
```

POST /park/parkInit



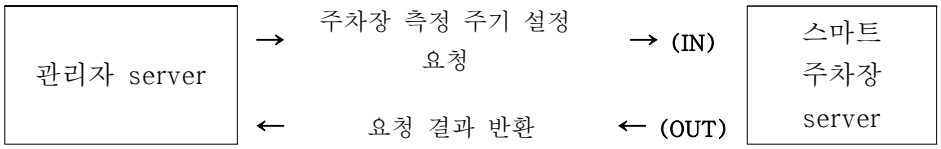

Parameter

| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| module_state | O | | json | 현재 제공 기능 on/off 여부 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)**

```
전송방향 : IN
{
   module_state : {
      {
         name : '주차장 길 안내 기능',
         status : True
      },
      {
         name : '일방통행 감지 기능',
         status : False
      },
      {
         name : '엘리베이터 설정 기능',
         status : True
      },
   },
 }
전송방향 : OUT
{
   result_code : 200
}
```

POST /park/timer




Parameter

| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| timer | O | | int | 주차장 측정 주기 값 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)

```
전송방향 : IN
{
   timer : 500
 }
전송방향 : OUT
{
   result_code : 200
}
```

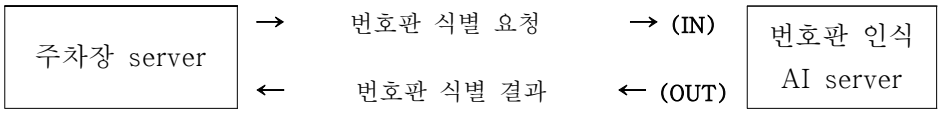
## 5.4. REST API 정의 및 설계 (번호판 인식 AI server)

○ REST API 정의

| Method | URI | Description |
| --- | --- | --- |
| GET | /ai/identification | 번호판 식별 요청 |

○ REST API 상세설계

**GET /ai/identification**




**Parameter**

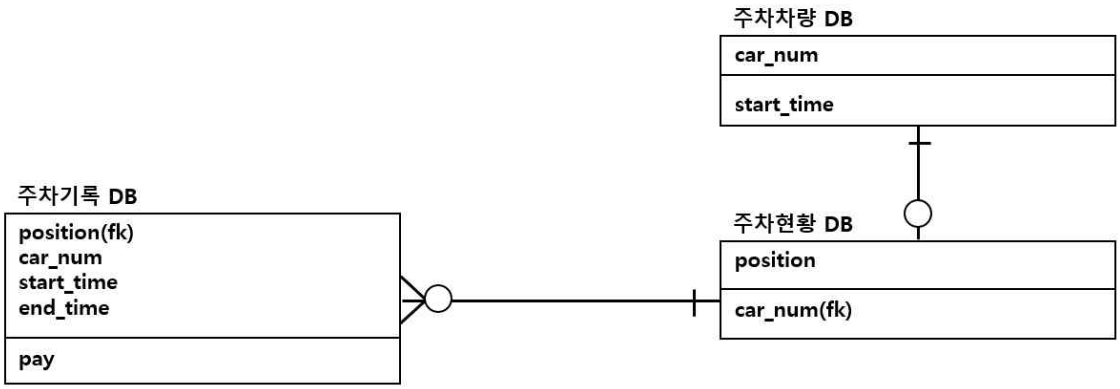
| 속성 | 전송방향 | | Type | Description |
|---|---|---|---|---|
| | IN | OUT | | |
| image | O | | Object | 번호판 이미지 |
| text | | O | string | 식별된 번호판의 값 |
| result_code | | O | int | 처리결과 코드(200 : 성공, 600: 오류 코드) |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**실제 전송내용 (JSON 의 경우라면, JSON이 아닐경우에는 전송내용을 작성한다.)**

```
전송방향 : IN
{
   image : '31아4123.jpg'
}
전송방향 : OUT
{
   text : '31아4123',
   result_code : 200
}
```

# 6. 데이터베이스 설계

## 6.1. ERD




<그림 6.1> 스마트 주차장 ERD

## 6.2. 논리적 DB설계 (테이블명세서)

| 테이블명(주차차량) : | | | SENTENCE | | | |
|---|---|---|---|---|---|---|
| NO | Attribute | Data Type | NN | PK | FK | Description |
| 1 | car_num | VARCHAR(20) | O | O | O | 차량번호 |
| 2 | start_time | DATETIME | O | | | 주차 시작시간 |

| 테이블명(주차현황) : | | | SENTENCE | | | |
|---|---|---|---|---|---|---|
| NO | Attribute | Data Type | NN | PK | FK | Description |
| 1 | position | INTEGER | O | O | O | 주차구역 고유번호 |
| 2 | car_num | VARCHAR(20) | | | O | 차량번호 |

| 테이블명(주차기록) : | | | SENTENCE | | | |
|---|---|---|---|---|---|---|
| NO | Attribute | Data Type | NN | PK | FK | Description |
| 1 | position | VARCHAR(20) | O | O | O | 주차구역 고유번호 |
| 2 | car_num | VARCHAR(20) | O | O | | 차량번호 |
| 3 | start_time | DATETIME | O | O | | 주차 시작시간 |
| 4 | end_time | DATETIME | O | O | | 주차 종료시간 |
| 5 | pay | INTEGER | O | | | 요금 |

## 6.3. 물리적 DB설계 (SQL스크립트)

| 테이블명 | PARKCAR |
|---|---|

```
CREATE TABLE PARKCAR
(
  car_num varchar(20),
  start_time datatime default sysdate
  primary key(car_num)
);
```

| 테이블명 | PARKCAR |
|---|---|

```
CREATE TABLE PARKNOW
(
  position int(10),
  car_num varchar(20),
  primary key(position),
  foreign key(car_num) references PARKCAR(car_num)
);
```

| 테이블명 | PARKCAR |
|---|---|

```
CREATE TABLE PARKLOG
(
  position int(10),
  car_num varchar(20),
  start_time datatime,
  end_time datatime,
  pay int(10) not null,
  primary key(position, car_num, start_time, end_time),
  foreign key(position) references PARKNOW(position)
);
```

# 7. 환경구성

## 7.1. 개발환경 및 운영환경

○ 개발환경(IoT 기기)

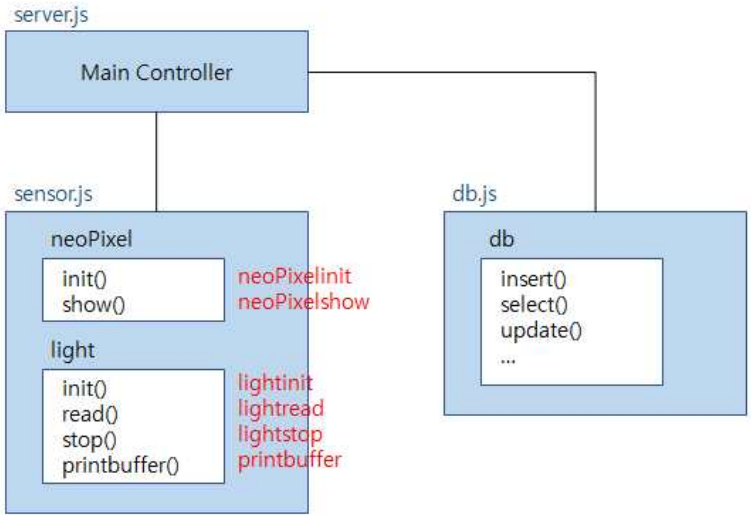
| 구분 | 개발환경 | 규격 | 비고 |
|---|---|---|---|
| 개발툴 | 편집툴 | ATOM | |
| | 개발IDE | VS Code | |
| | 프레임워크 | Node.js | |
| | 사용할 미들웨어(모듈) | Express<br>Router<br>WiringPi<br>winston | |
| 서버 | OS | Raspbian | |
| | DB | MariaDB | |
| 서버 하드웨어 | CPU | ARM Cortex-A53 | |
| | RAM | 1GB | |
| | DISK | MicroSD | |
| | Network | 100Mbps | |

○ 개발환경(서버)

| 구분 | 개발환경 | 규격 | 비고 |
|---|---|---|---|
| 개발툴 | 편집툴 | ATOM | |
| | 개발IDE | VS Code | |
| | 프레임워크 | Node.js | |
| | 사용할 미들웨어(모듈) | Express<br>Router<br>React | |
| 서버 | OS | Windows | |
| | DB | MariaDB | |
| 서버 하드웨어 | CPU | Intel Core i7-7700HQ | |
| | RAM | 16GB | |
| | DISK | 512GB(SSD) | |
| | Network | 100Mbps | |

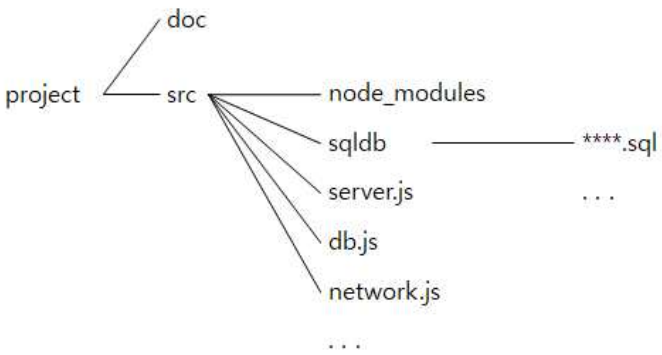
## 7.2. 소스디렉터리 구조

○ 개발환경 (IoT 기기)




<그림 7.1> MVC 모델 구조 (IoT 기기)

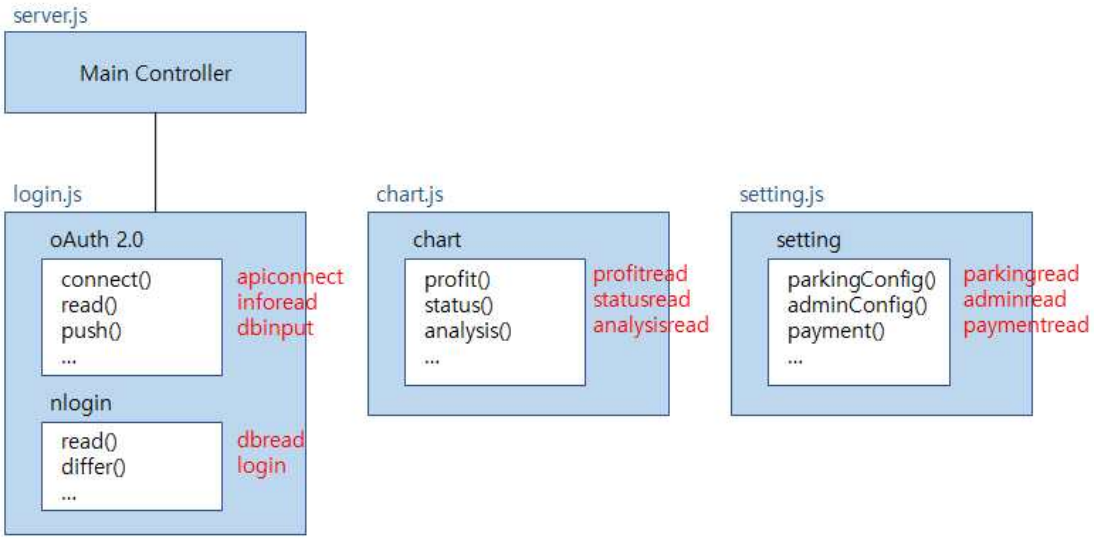
구현에 앞서, 그림 7.1과 같은 구조로 MVC 모델 구조를 정의한다.




<그림 7.2> 디렉터리 및 파일환경

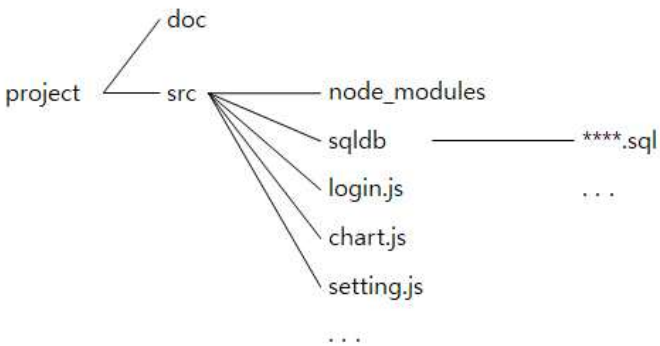
그림 7.2에서처럼 개발 소스와 디렉터리의 구조를 정의한다.

○ 개발환경 (서버)




<그림 7.3> MVC 모델 구조 (서버)

구현에 앞서, 그림 7.3과 같은 구조로 MVC 모델 구조를 정의한다.




<그림 7.4> 디렉터리 및 파일환경

그림 7.4에서처럼 개발 소스와 디렉터리의 구조를 정의한다.