

OOAD with AI

The RVC Control SW

6팀 정상훈, 정수혁, 하재아, 한지훈

| 목차

Chapter 0. Inception

Chapter 1. OOA

Chapter 2. OOD

Chapter 3. OOI

Chapter 4. 장단점 및 소감

Inception - CLAUDE.md



```
Claude Code v2.1.143
Sonnet 4.6 · Claude Pro
~/Desktop/rvc-controller

Opus 4.7 xhigh is now available! · /model to switch

> Try "create a util logging.py that..."
```

```
# RVC SW Controller 프로젝트

## 프로젝트 개요
로봇 청소기(RVC) SW Controller를 00AD 프로세스로 개발한다.

## 요구사항
- RVC는 가정의 표면을 자동으로 청소하고 닦는다
- 청소하면서 직진한다
- 센서가 장애물을 감지하면 청소를 멈추고, 좌/우로 방향을 틀어 청소하며 전진한다
- 앞/좌/우 모두 장애물이 있으면 후진 후 좌/우로 방향을 틀어 전진한다
- 먼지를 감지하면 잠시 청소 출력을 높인다
- HW 제어의 상세 설계 및 구현은 고려하지 않는다
- 자동 청소 기능에만 집중한다

## 폴더 구조
arch/requirements/      - 요구사항 문서
arch/usecases/          - Use Case 문서
arch/analysis/ssd/      - SSD 다이어그램
arch/analysis/domain/   - 도메인 모델
arch/design/SD/         - 시퀀스 다이어그램
arch/design/class/      - 클래스 다이어그램
arch/system-tests/     - 시스템 테스트
src/                   - 소스 코드
tests/                 - 테스트 코드
simulator/             - RVC 동작 시뮬레이터 (유일하게 객체지향으로 개발하지 않아도 됨)

## 진행 단계
1. 요구사항 분석: arch/system.md, arch/requirements/fr-nfr.md
2. 00A: arch/usecases/usecases.md → arch/usecases/UC-nnn.md → arch/analysis/ssd/ssd.md → arch/analysis/domain/domain.md
3. 00D: arch/design/SD/SD.md → arch/design/class/class-diagram.md
4. 00I: src/ → tests/ → arch/system-tests/system-tests.md → simulator/

## 설계 방향
- 구동 모터, 청소 모터는 외부 액터로 취급한다
- 시스템 내부에는 모터를 제어하는 별도 컨트롤러 객체를 둔다 (예: MotorController, CleaningController)

## 규칙
- 모든 다이어그램은 Mermaid 형식으로 작성한다
- 모든 코드는 객체지향 원칙을 철저히 준수한다 (simulator 제외)
- simulator를 통하여 system-tests를 테스트한다
- 장애물 회피 시 좌/우 방향은 랜덤으로 선택한다
- usecases, ssd, domain-model, ssd, class-diagram 은 반드시 통일성을 가질 것
- 구현 언어: C++
```

Inception - system.md

폴더 구조를 생성하고 arch/system.md를 작성해줘

시스템 정의 (System Definition)

1. 시스템 목적

RVC(Robot Vacuum Cleaner) SW Controller는 로봇 청소기가 가정의 바닥을 자동으로 청소하고 닦을 수 있도록 제어하는 소프트웨어 시스템이다.

2. 시스템 범위

포함 범위

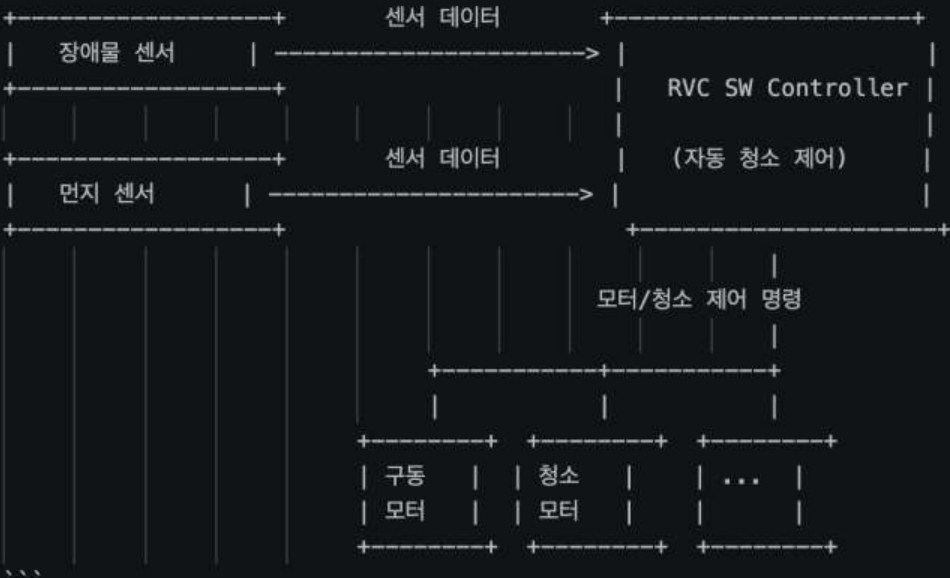
- 자동 주행 및 청소 제어 로직
- 장애물 감지에 따른 방향 전환 로직
- 먼지 감지에 따른 청소 출력 조절 로직
- 센서 데이터 수신 및 처리
- HW 제어 인터페이스 (추상화 수준)

제외 범위

- HW 드라이버 상세 설계 및 구현
- 수동 조작(리모컨, 앱 제어) 기능
- 배터리 관리 및 충전 복귀 기능
- 맵핑 및 경로 계획 기능

3. 시스템 컨텍스트

...



...

4. 주요 요구사항 요약

ID	요구사항
SR-01	RVC는 가정의 표면을 자동으로 청소하고 닦는다
SR-02	청소하면서 직진한다
SR-03	센서가 장애물을 감지하면 청소를 멈추고, 좌/우로 방향을 틀어 청소하며 전진한다
SR-04	앞/좌/우 모두 장애물이 있으면 후진 후 좌/우로 방향을 틀어 전진한다
SR-05	먼지를 감지하면 잠시 청소 출력을 높인다

5. 시스템 외부 인터페이스

인터페이스	방향	설명
장애물 센서	입력	전방/좌/우 장애물 감지 신호
먼지 센서	입력	먼지 감지 신호
구동 모터	출력	전진/후진/정지/방향 전환 제어
청소 모터	출력	청소 출력 조절 (Normal / High)

6. 핵심 동작 시나리오

- **정상 청소****: RVC가 전원을 켜면 직진하며 청소 및 닦기를 수행한다.
- **장애물 회피 (측면 여유 있음)****: 전방 장애물 감지 → 청소 정지 → 좌 또는 우로 회전 → 청소 재개 및 전진.
- **장애물 회피 (삼면 막힘)****: 전/좌/우 장애물 모두 감지 → 후진(이때 좌 또는 우에 막히지 않는 방향이 나올 때까지 후진한다) → 좌 또는 우로 회전 → 전진.
- **고출력 청소****: 먼지 감지 → 청소 출력 High 전환 → 일정 시간 경과 후 Normal 복귀.

7. 개발 단계

단계	산출물
요구사항 분석	arch/system.md, arch/requirements/fr-nfr.md
00A	arch/usecases/usecases.md, arch/usecases/UC-nnn.md, arch/analysis/ssd/ssd.md, arch/analysis/domain/domain.md
00D	arch/design/sd/sd.md, arch/design/class/class-diagram.md
00I	src/, tests/, simulator/, arch/system-tests/system-tests.md

Inception - fr-nfr.md

› arch/system.md가 수정 됐 어 . 다시 읽고 fr-nfr.md 작성 해 주

인터페이스	방향	설명
장애물 센서	입력	전방/좌/우 장애물 감지 신호
먼지 센서	입력	먼지 감지 신호
구동 모터	출력	전진/후진/정지/방향 전환 제어
청소 모터	출력	청소 출력 조절 (Normal / High)
걸레 모터	출력	걸레 동작 제어

```
# 기능/비기능 요구사항 (Functional / Non-Functional Requirements)

## 1. 기능 요구사항 (Functional Requirements)

### FR-01: 자동 청소 및 닦기
- **설명**: RVC는 전원이 켜지면 바닥을 자동으로 청소(흡입)하고 닦으면서 직진한다.
- **출처**: SR-01, SR-02
- **입력**: 전원 ON 이벤트
- **출력**: 구동 모터 전진, 청소 모터 Normal 출력 동작
- **조건**: -

---

### FR-02: 전방 장애물 감지 시 방향 전환
- **설명**: 전방에 장애물이 감지되고 좌 또는 우 중 하나라도 여유 공간이 있으면, 청소를 멈추고 여유 있는 방향으로 회전한 뒤 청소를 재개하며 전진한다.
- **출처**: SR-03
- **입력**: 전방 장애물 감지 신호 (장애물 센서)
- **출력**: 청소 모터 정지 → 구동 모터 회전(좌 또는 우) → 청소 모터 재개 및 구동 모터 전진
- **조건**: 좌/우 중 적어도 한 방향에 장애물이 없어야 한다
- **선택 정책**: 좌/우 모두 여유 있으면 랜덤으로 방향을 선택한다

---

### FR-03: 삼면 장애물 감지 시 후진 후 방향 전환
- **설명**: 전방·좌·우 모두 장애물이 감지되면 후진하고, 좌 또는 우에 막히지 않는 방향이 나올 때까지 후진한 뒤 해당 방향으로 회전하여 전진한다.
- **출처**: SR-04
- **입력**: 전방·좌·우 장애물 감지 신호 동시 수신
- **출력**: 구동 모터 후진 → (좌 또는 우 여유 공간 감지 시) 구동 모터 회전 → 전진
- **조건**: 후진은 좌 또는 우 중 하나의 장애물이 해소될 때까지 계속한다
- **선택 정책**: 후진 중 좌/우 모두 동시에 여유 공간이 확보되면 랜덤으로 방향을 선택한다

---

### FR-04: 먼지 감지 시 청소 출력 강화
- **설명**: 먼지 센서가 먼지를 감지하면 청소 모터 출력을 High로 높이고, 일정 시간이 경과하면 Normal 출력으로 복귀한다.
- **출처**: SR-05
- **입력**: 먼지 감지 신호 (먼지 센서)
- **출력**: 청소 모터 출력 Normal → High → (일정 시간 후) Normal
- **조건**: 출력 강화 지속 시간은 별도 파라미터로 관리한다

---
```

Inception - fr-nfr.md

```
## 2. 비기능 요구사항 (Non-Functional Requirements)

### NFR-01: 실시간 응답성
- **설명**: 센서 이벤트(장애물, 먼지 감지) 수신 후 제어 명령 발행까지의 지연이 허용 범위 이내여야 한다.
- **기준**: 센서 이벤트 발생 후 100ms 이내에 제어 명령 발행

---

### NFR-02: 안전성
- **설명**: 알 수 없는 상태나 예외 상황에서 RVC는 안전한 상태(정지)로 전환되어야 한다.
- **기준**: 예외 발생 시 구동 모터 및 청소 모터를 즉시 정지한다

---

### NFR-03: 테스트 가능성
- **설명**: 모든 기능 요구사항은 단위 테스트 및 시스템 테스트로 검증 가능해야 한다.
- **기준**: FR-01 ~ FR-04 각각에 대한 시스템 테스트 시나리오가 존재한다 (arch/system-tests/system-tests.md)

---

### NFR-04: 유지보수성
- **설명**: HW 인터페이스(센서, 모터)는 추상화 계층으로 분리되어 SW 로직 변경 없이 교체 가능해야 한다.
- **기준**: 센서 및 모터 제어는 인터페이스(추상 클래스 또는 프로토콜)를 통해서만 접근한다

---

### NFR-05: 시뮬레이션 지원
- **설명**: 실제 HW 없이 SW 로직을 검증할 수 있는 시뮬레이터를 지원한다.
- **기준**: simulator/ 디렉터리에 HW 인터페이스 스텝(Stub) 구현이 존재한다

---

## 3. 요구사항 추적 매트릭스

| 시스템 요구사항 | 기능 요구사항 | 비기능 요구사항 |
|-----|-----|-----|
| SR-01 (청소 및 닦기) | FR-01 | NFR-01, NFR-02 |
| SR-02 (직진) | FR-01 | NFR-01 |
| SR-03 (장애물 → 방향 전환) | FR-02 | NFR-01, NFR-02 |
| SR-04 (삼면 장애물 → 후진) | FR-03 | NFR-01, NFR-02 |
| SR-05 (먼지 → 출력 강화) | FR-04 | NFR-01 |
```

Inception - usecases.md

› arch/requirements/requirements.md를 읽고 Use Case를 도출해서 arch/usecases/usecases.md를 작성해줘

1. 액터 (Actors)

액터	종류	설명
User	Primary Actor	시스템을 시작시키는 주체
ObstacleSensor	Primary Actor	전방/좌/우 장애물 감지 신호를 시스템에 전달한다
DustSensor	Primary Actor	먼지 감지 신호를 시스템에 전달한다
DriveMotor	Secondary Actor	시스템으로부터 주행 명령(전진/후진/회전/정지)을 수신한다
CleaningMotor	Secondary Actor	시스템으로부터 청소 출력 명령(Normal/High/정지)을 수신한다

3. Use Case 목록

Use Case ID	이름	관련 액터	관련 요구사항	상세 문서
UC-001	자동 청소 수행	User, DriveMotor, CleaningMotor	FR-01	UC-001.md
UC-002	측면 장애물 회피	ObstacleSensor, DriveMotor, CleaningMotor	FR-02	UC-002.md
UC-003	삼면 장애물 후진 회피	ObstacleSensor, DriveMotor	FR-03	UC-003.md
UC-004	고출력 청소	DustSensor, CleaningMotor	FR-04	UC-004.md

4. Use Case 요약

UC-001: 자동 청소 수행

User가 시스템을 시작하면 CleaningMotor를 Normal 출력으로 가동하고 DriveMotor로 직진하며 바닥을 청소하고 닦는다. 이 동작은 시스템의 기본(Base) 흐름이며, 나머지 UC는 이를 확장(extend)한다.

UC-002: 측면 장애물 회피

전방에 장애물이 감지되고 좌 또는 우 중 여유 공간이 있을 때, 청소를 일시 정지하고 여유 있는 방향으로 회전한 뒤 청소를 재개하며 전진한다.

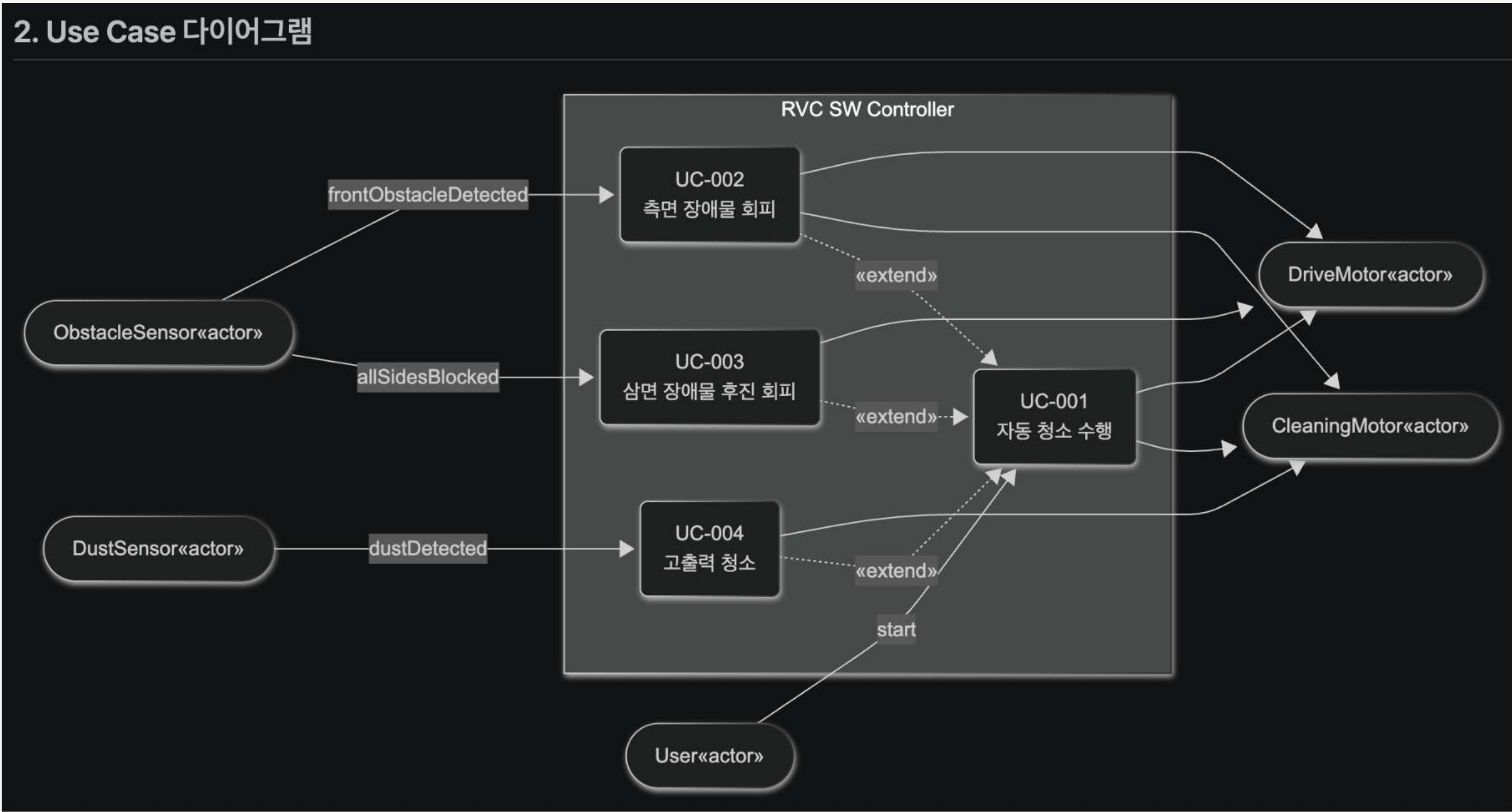
UC-003: 삼면 장애물 후진 회피

전방·좌·우 모두 장애물이 감지될 때, 후진하며 좌 또는 우 중 하나의 장애물이 해소될 때까지 후진을 유지한다. 여유 공간이 확보되면 해당 방향으로 회전하여 전진한다.

UC-004: 고출력 청소

먼지가 감지되면 CleaningMotor 출력을 High로 높이고, 일정 시간이 경과하면 자동으로 Normal 출력으로 복귀한다. 주행은 중단 없이 계속한다.

Inception - usecases.md



OOA - UC-001.md

› arch/usecases/usecases.md를 읽고 각 UC별로 상세 Use Case 문서를 arch/usecases/UC-001.md, UC-002.md, UC-003.md, UC-004.md로 작성해줘. 장애물 회피 시 좌/우 방향은 랜덤으로 선택한다.

› UC-001.md 기본 흐름 테이블에 액터/시스템 칸을 추가해줘. UC-002.md 형식과 동일하게 맞춰줘

UC-001: 자동 청소 수행

기본 정보

항목	내용
Use Case ID	UC-001
이름	자동 청소 수행
액터	User (Primary), DriveMotor (Secondary), CleaningMotor (Secondary)
관련 요구사항	FR-01, SR-01, SR-02
종류	Base Use Case

사전 조건 (Preconditions)

- User가 RVC를 시작시킨다.

사후 조건 (Postconditions)

- RVC가 CleaningMotor Normal 출력으로 직진하며 청소 및 닦기를 수행하고 있다.

기본 흐름 (Main Flow)

단계	액터/시스템	행위
1	User	RVC를 시작시킨다.
2	System	CleaningMotor를 Normal 출력으로 가동한다.
3	System	DriveMotor를 전진 방향으로 구동한다.
4	System	직진하며 바닥을 청소하고 닦는다.
5	System	단계 4를 계속 반복한다.

확장 포인트 (Extension Points)

확장 포인트	조건	확장 UC
EP-1	frontObstacleDetected AND (좌 OR 우 여유 있음)	UC-002
EP-2	allSidesBlocked	UC-003
EP-3	dustDetected	UC-004

OOA - UC-002.md

UC-002: 측면 장애물 회피

기본 정보

항목	내용
Use Case ID	UC-002
이름	측면 장애물 회피
액터	ObstacleSensor (Primary), DriveMotor (Secondary), CleaningMotor (Secondary)
관련 요구사항	FR-02, SR-03
종류	Extension of UC-001 (EP-1)

사전 조건 (Preconditions)

- UC-001이 실행 중이다.
- 전방에 장애물이 감지되었다.
- 좌 또는 우 중 하나 이상의 방향에 여유 공간이 있다.

사후 조건 (Postconditions)

- RVC가 장애물이 없는 방향(좌 또는 우)으로 회전한 뒤, 청소를 재개하며 전진하고 있다.

기본 흐름 (Main Flow)

단계	액터/시스템	행위
1	ObstacleSensor	전방 장애물 감지 신호를 시스템에 전달한다.
2	System	CleaningMotor를 정지한다.
3	System	DriveMotor를 정지한다.
4	ObstacleSensor	좌/우 방향 장애물 유무 신호를 시스템에 전달한다.
5	System	여유 있는 방향(좌 또는 우)으로 DriveMotor를 회전시킨다.
6	System	CleaningMotor를 Normal 출력으로 재가동한다.
7	System	DriveMotor를 전진 방향으로 구동한다.
8		UC-001 기본 흐름 단계 4로 복귀한다.

대안 흐름 (Alternative Flow)

A1: 좌·우 모두 여유 있을 때

- 단계 5에서 좌·우 모두 여유 있으면 랜덤으로 방향을 선택하여 회전한다.

예외 흐름 (Exception Flow)

E1: 회피 도중 전/좌/우 모두 막힘

- 단계 5 이전에 삼면이 모두 막힌 것이 감지되면 UC-003으로 전환한다.

OOA - UC-003.md

UC-003: 삼면 장애물 후진 회피

기본 정보

항목	내용
Use Case ID	UC-003
이름	삼면 장애물 후진 회피
액터	ObstacleSensor (Primary), DriveMotor (Secondary)
관련 요구사항	FR-03, SR-04
종류	Extension of UC-001 (EP-2)

사전 조건 (Preconditions)

- UC-001이 실행 중이다.
- 전방·좌·우 방향 모두에 장애물이 감지되었다.

사후 조건 (Postconditions)

- RVC가 후진 중 확보된 여유 방향(좌 또는 우)으로 회전한 뒤 전진하고 있다.

기본 흐름 (Main Flow)

단계	액터/시스템	행위
1	ObstacleSensor	전방·좌·우 장애물 감지 신호를 시스템에 전달한다.
2	System	CleaningMotor를 정지한다.
3	System	DriveMotor를 후진 방향으로 구동한다.
4	ObstacleSensor	좌/우 방향 장애물 유무 신호를 계속 시스템에 전달한다.
5	System	좌 또는 우에 여유 공간이 확보될 때까지 단계 3~4를 반복한다.
6	System	DriveMotor를 정지한다.
7	System	여유 공간이 확보된 방향(좌 또는 우)으로 DriveMotor를 회전시킨다.
8	System	DriveMotor를 전진 방향으로 구동한다.
9	System	CleaningMotor를 Normal 출력으로 재가동한다.
10		UC-001 기본 흐름 단계 4로 복귀한다.

대안 흐름 (Alternative Flow)

A1: 후진 중 좌·우 동시에 여유 공간 확보

- 단계 5에서 좌·우 모두 동시에 해소되면 랜덤으로 방향을 선택하여 회전한다.

OOA - UC-004.md

UC-004: 고출력 청소

기본 정보

항목	내용
Use Case ID	UC-004
이름	고출력 청소
액터	DustSensor (Primary), CleaningMotor (Secondary)
관련 요구사항	FR-04, SR-05
종류	Extension of UC-001 (EP-3)

사전 조건 (Preconditions)

- UC-001이 실행 중이다 (CleaningMotor Normal 출력, DriveMotor 전진 중).
- DustSensor가 먼지를 감지하였다.

사후 조건 (Postconditions)

- CleaningMotor가 Normal 출력으로 복귀하였다.
- DriveMotor 전진은 UC-004 실행 중에도 중단 없이 유지되었다.

기본 흐름 (Main Flow)

단계	액터/시스템	행위
1	DustSensor	먼지 감지 신호를 시스템에 전달한다.
2	System	CleaningMotor 출력을 High로 전환한다.
3	System	고출력 지속 타이머를 시작한다.
4	System	DriveMotor 전진을 유지한다 (주행 중단 없음).
5	System	타이머가 만료될 때까지 단계 4를 반복한다.
6	System	CleaningMotor 출력을 Normal로 복귀시킨다.
7		UC-001 기본 흐름 단계 4로 복귀한다.

대안 흐름 (Alternative Flow)

A1: 고출력 청소 중 장애물 감지

- 단계 4 중 장애물이 감지되면 UC-002 또는 UC-003을 수행한다.
- 장애물 회피 완료 후 남은 타이머 시간만큼 고출력 청소를 계속하고 단계 6으로 진행한다.

A2: 고출력 청소 중 추가 먼지 감지

- 단계 5 중 먼지가 재감지되면 타이머를 초기화하여 단계 3부터 재시작한다.

OOA - System Sequence Diagram

› arch/usecases/ 폴더의 모든 UC 문서를 읽고 각 UC별로 SSD를 Mermaid 형식으로 arch/analysis/ssd/ 폴더에 작성해줘

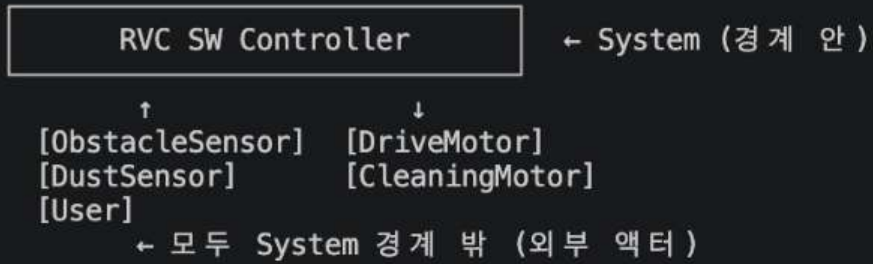
› arch/analysis/ssd/ 의 모든 SSD를 수정해줘. SSD는 외부 액터와 시스템 (RVC SW Controller) 사이의 메시지만 표현해야 해. 청소 모터, 구동 모터 등 시스템 내부 객체는 SSD에서 제거해줘.



› 현재 SSD에서 CleaningMotor와 DriveMotor를 외부 액터로 설계하고 있는 게 맞아? 설계 의도를 설명해줘

- 좋은 질문입니다. 현재 설계의 의도와 trade-off를 설명합니다.

현재 설계의 시스템 경계



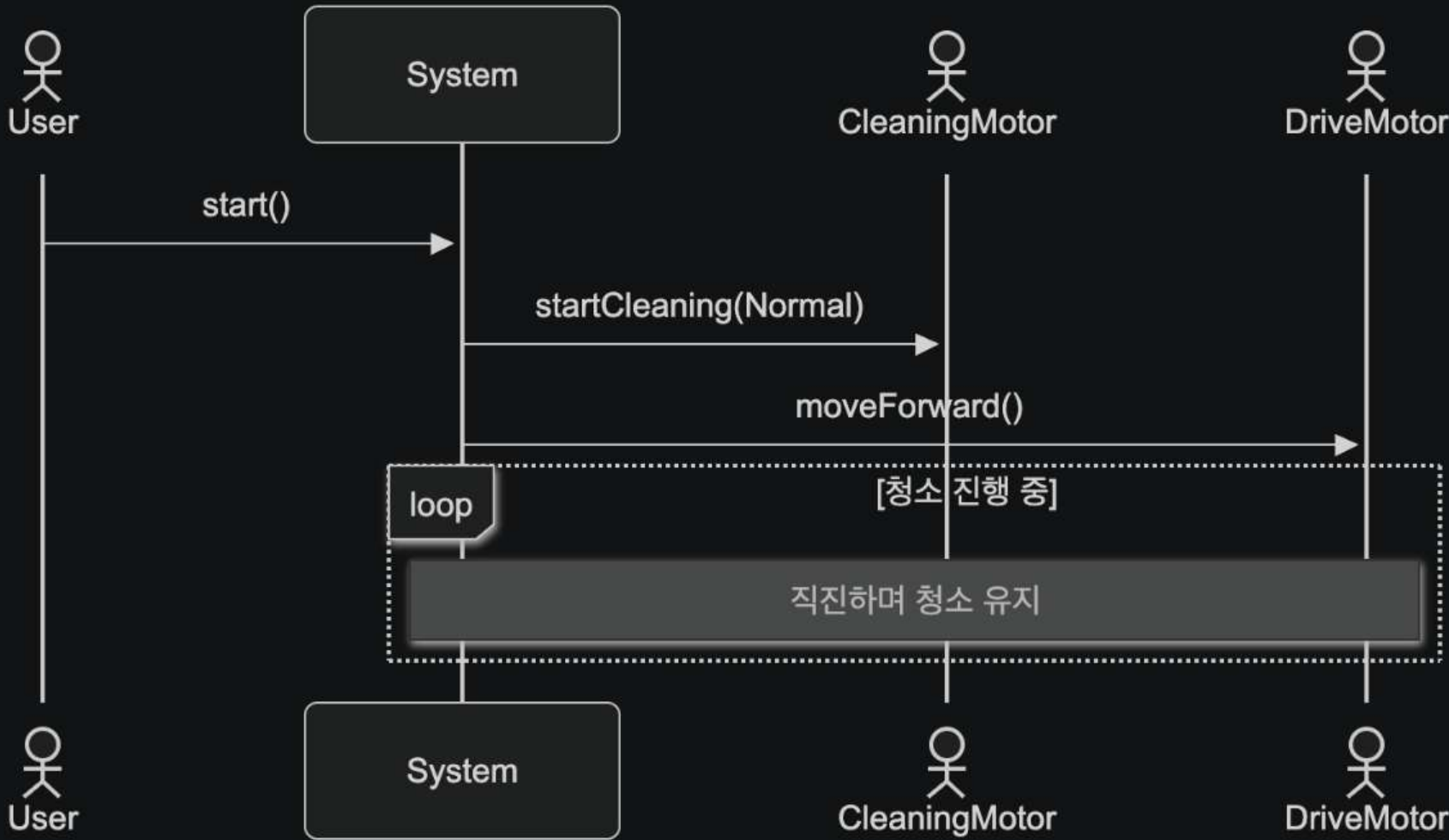
의도: SW Controller만을 System으로 보고, 모터는 SW가 명령을 보내는 HW 인터페이스 대상으로 취급합니다. 이렇게 하면 SSD에서 도출되는 시스템 오퍼레이션 (start(), frontObstacleDetected() 등)과 더불어, System이 모터에 보내는 메시지 (stopCleaning(), moveForward() 등)가 HW 추상화 인터페이스로 명확히 정의됩니다. CLAUDE.md의 "HW 제어의 상세 설계 및 구현은 고려하지 않는다"는 원칙과도 일치합니다 - HW는 블랙박스 액터로 두고, SW는 그 인터페이스에만 의존합니다.

OOA - SSD-001.md

SSD-001: 자동 청소 수행

- 연관 UC: UC-001
- 관련 요구사항: FR-01, SR-01, SR-02

시스템 시퀀스 다이어그램



시스템 오퍼레이션 목록

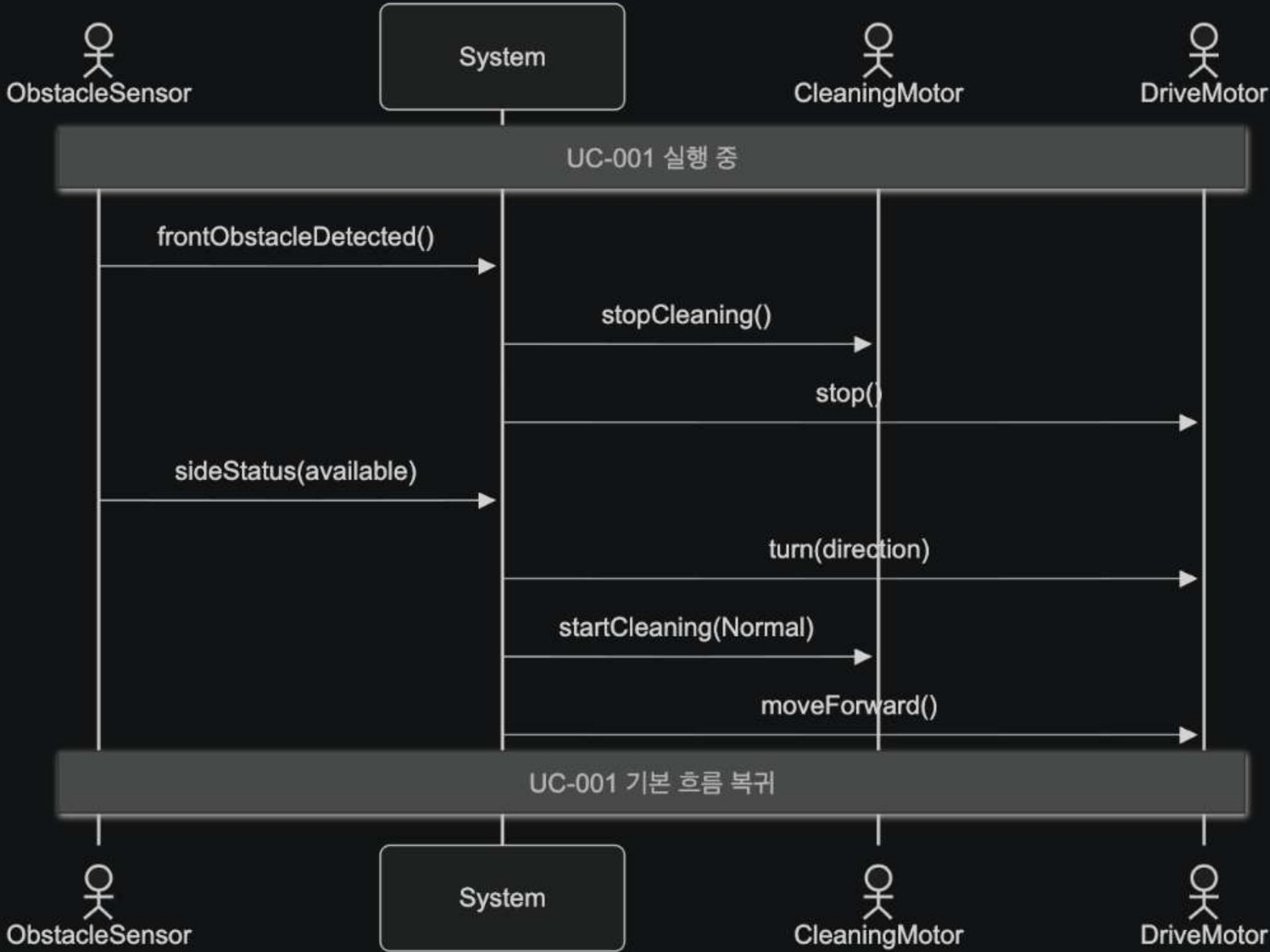
오퍼레이션	설명
start()	User가 시스템을 시작시킨다

OOA - SSD-002.md

SSD-002: 측면 장애물 회피

- 연관 UC: UC-002
- 관련 요구사항: FR-02, SR-03

시스템 시퀀스 다이어그램



시스템 오퍼레이션 목록

오퍼레이션	설명
frontObstacleDetected()	전방 장애물 감지 신호를 수신한다
sideStatus(available)	여유 공간이 있는 방향(Direction)의 목록을 수신한다

OOA - SSD-003.md

SSD-003: 삼면 장애물 후진 회피

- 연관 UC: UC-003
- 관련 요구사항: FR-03, SR-04

시스템 시퀀스 다이어그램

```
sequenceDiagram
    participant OS as ObstacleSensor
    participant S as System
    participant CM as CleaningMotor
    participant DM as DriveMotor

    Note over S: UC-001 실행 중
    OS->>S: allSidesBlocked()
    S->>CM: stopCleaning()
    S->>DM: moveBackward()
    loop [좌 또는 우 여유 공간 확보 시까지]
        S->>OS: sideStatus(available)
    end
    S->>DM: stop()
    S->>DM: turn(direction)
    S->>DM: moveForward()
    S->>CM: startCleaning(Normal)
    Note over S: UC-001 기본 흐름 복귀
```

시스템 오퍼레이션 목록

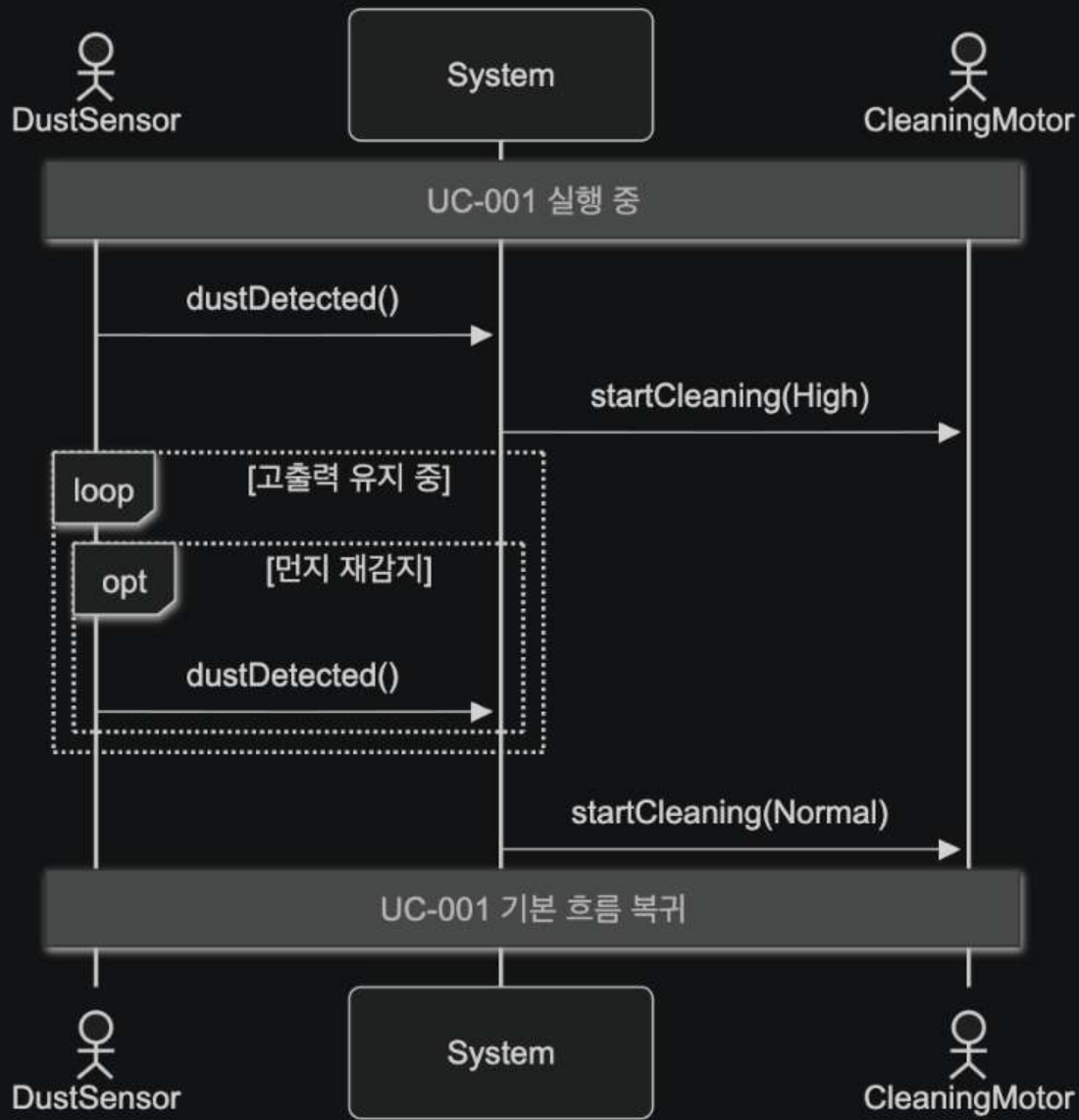
오퍼레이션	설명
allSidesBlocked()	전방·좌·우 모두 장애물 감지 신호를 수신한다
sideStatus(available)	여유 공간이 있는 방향(Direction)의 목록을 수신한다. 후진 중 반복 수신

OOA - SSD-004.md

SSD-004: 고출력 청소

- 연관 UC: UC-004
- 관련 요구사항: FR-04, SR-05

시스템 시퀀스 다이어그램

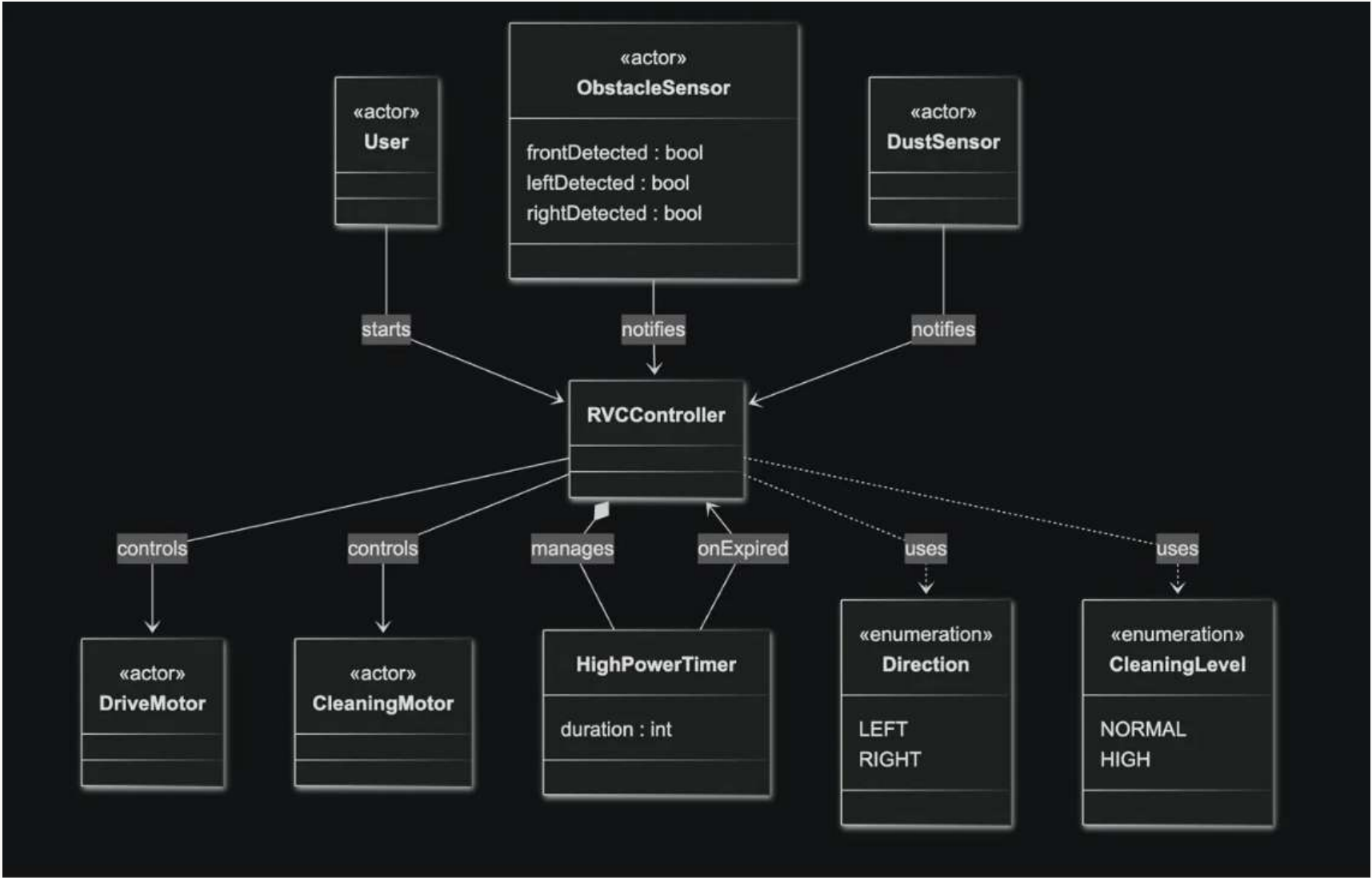


시스템 오퍼레이션 목록

오퍼레이션	설명
dustDetected()	먼지 감지 신호를 수신한다. 재감지 시 동일 오퍼레이션으로 타이머 초기화

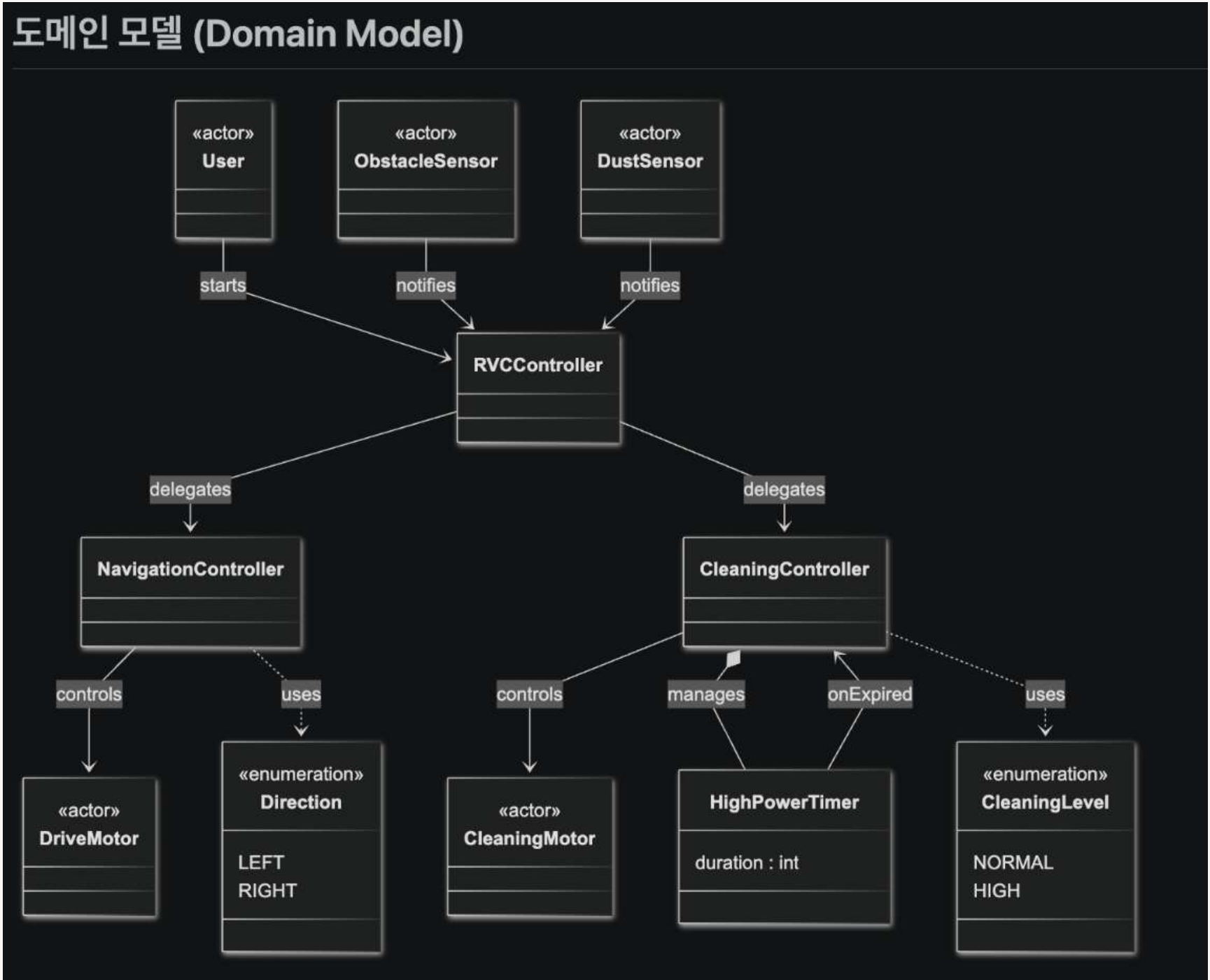
OOA - domain.md

> arch/analysis/ssd/ 의 모든 SSD와 arch/usecases/ 의 모든 UC 문서를 읽고 도메인 모델을 Mermaid 형식으로 arch/analysis/domain/model.md에 작성해줘



OOA - domain.md

> 도메인 모델을 검토해줘. RVController가 너무 많은 책임을 가지고 있는 것 같아. 단일 책임 원칙에 맞게 내부 객체들을 적절히 분리해줘.



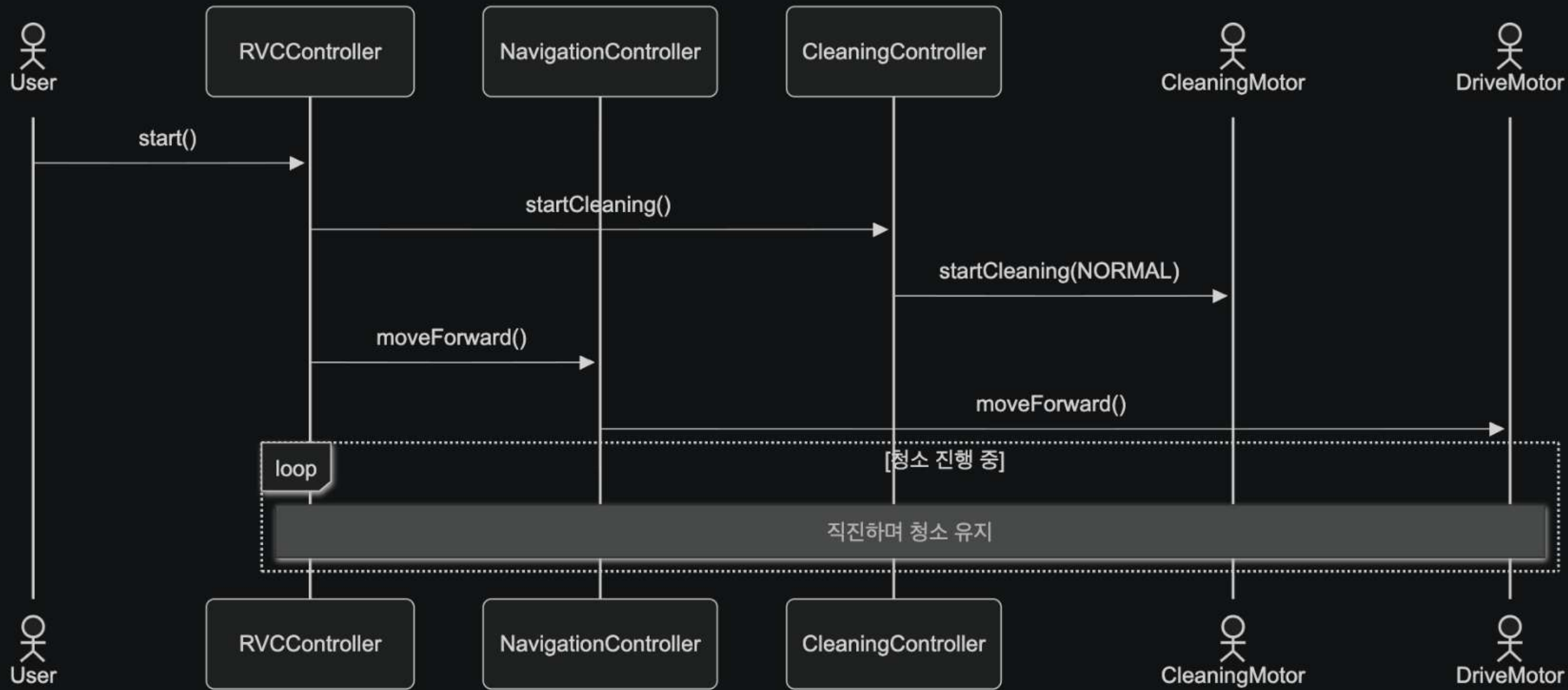
OOD - SD-001.md

> arch/analysis/ 의 모든 문서를 읽고 시퀀스 다이어그램을 Mermaid 형식으로 arch/design/sd/ 폴더에 UC별로 작성해줘

SD-001: 자동 청소 수행

- 연관 UC: UC-001
- 연관 SSD: SSD-001

시퀀스 다이어그램

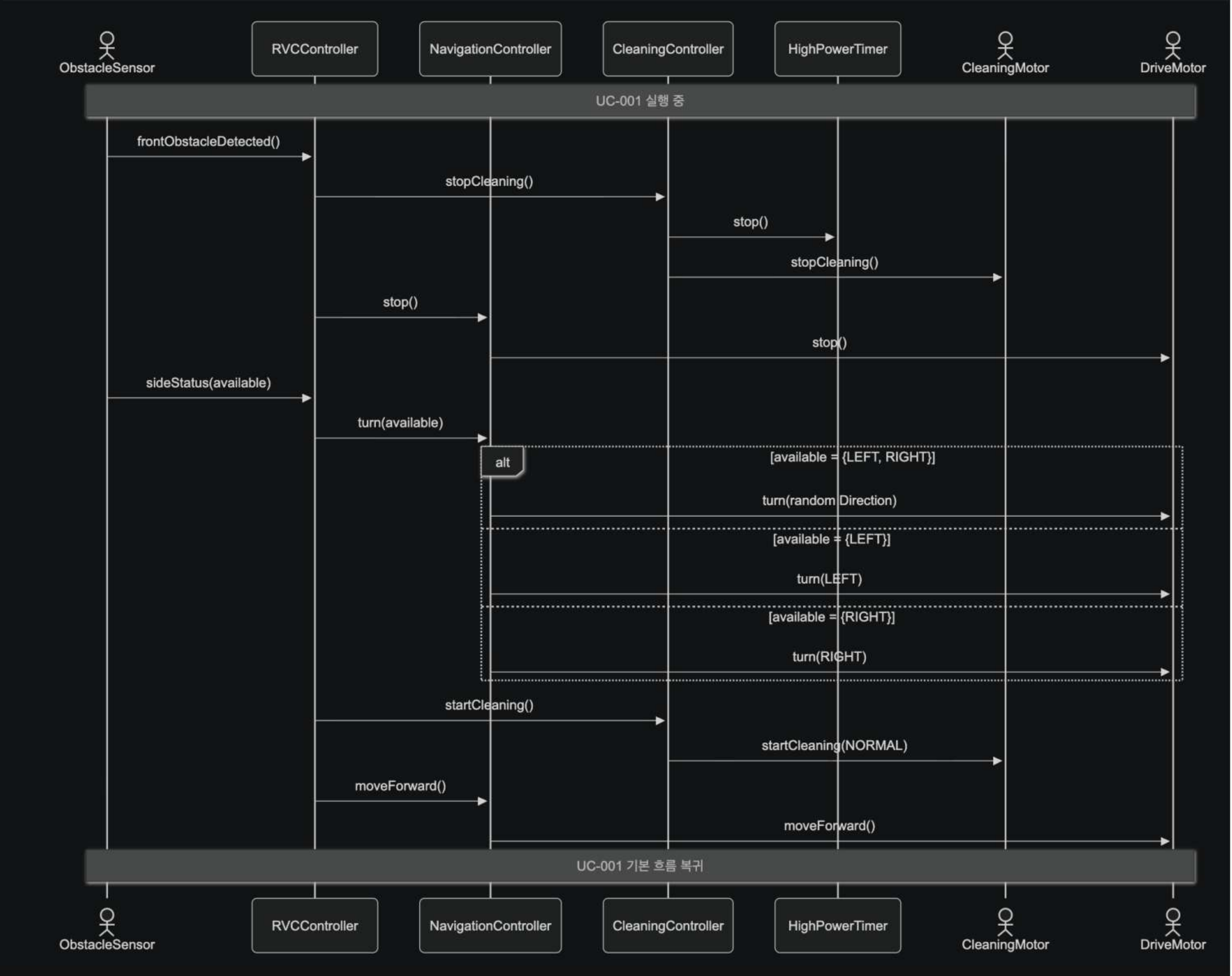


OOD - SD-002.md

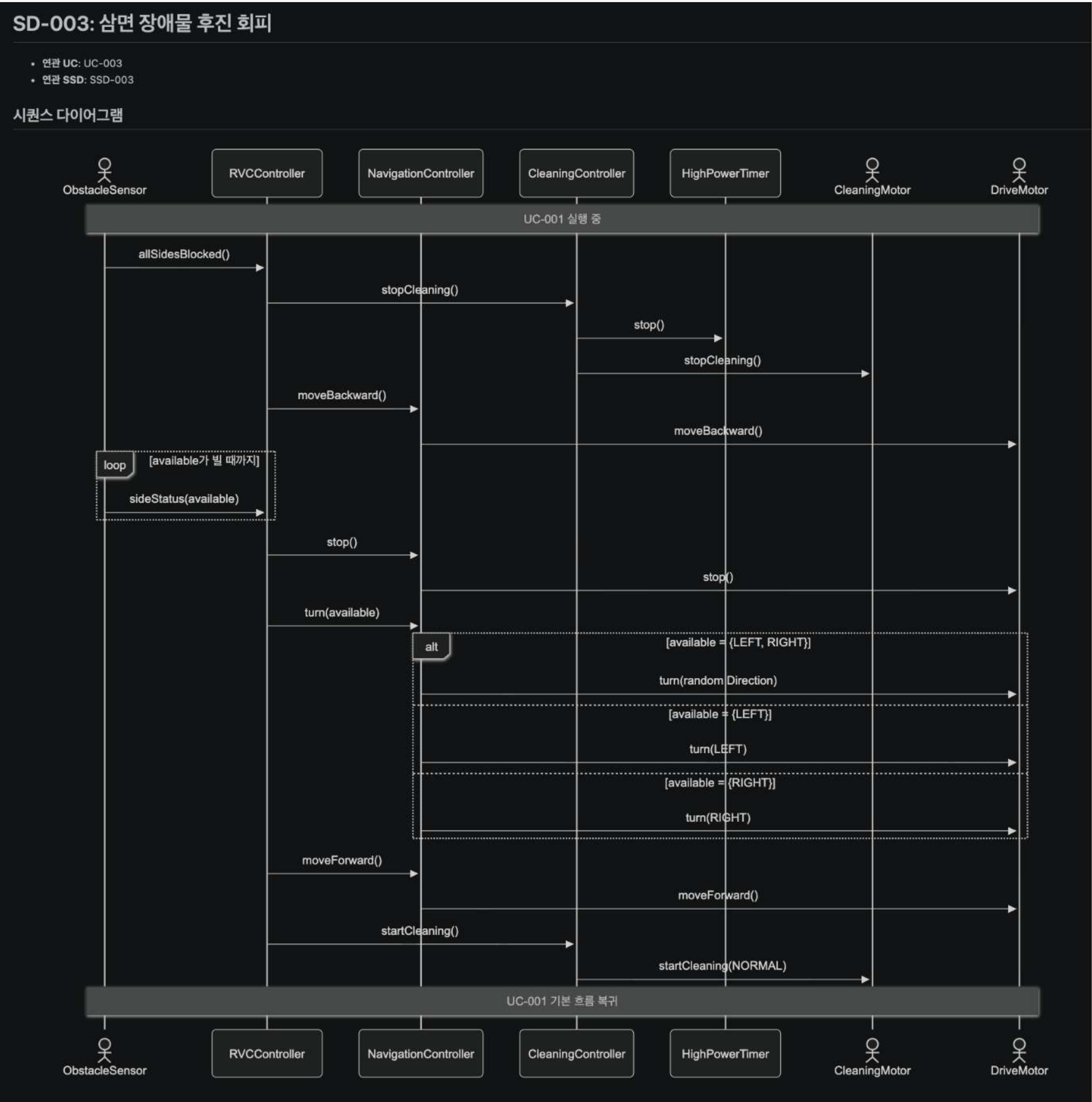
SD-002: 측면 장애물 회피

- 연관 UC: UC-002
- 연관 SSD: SSD-002

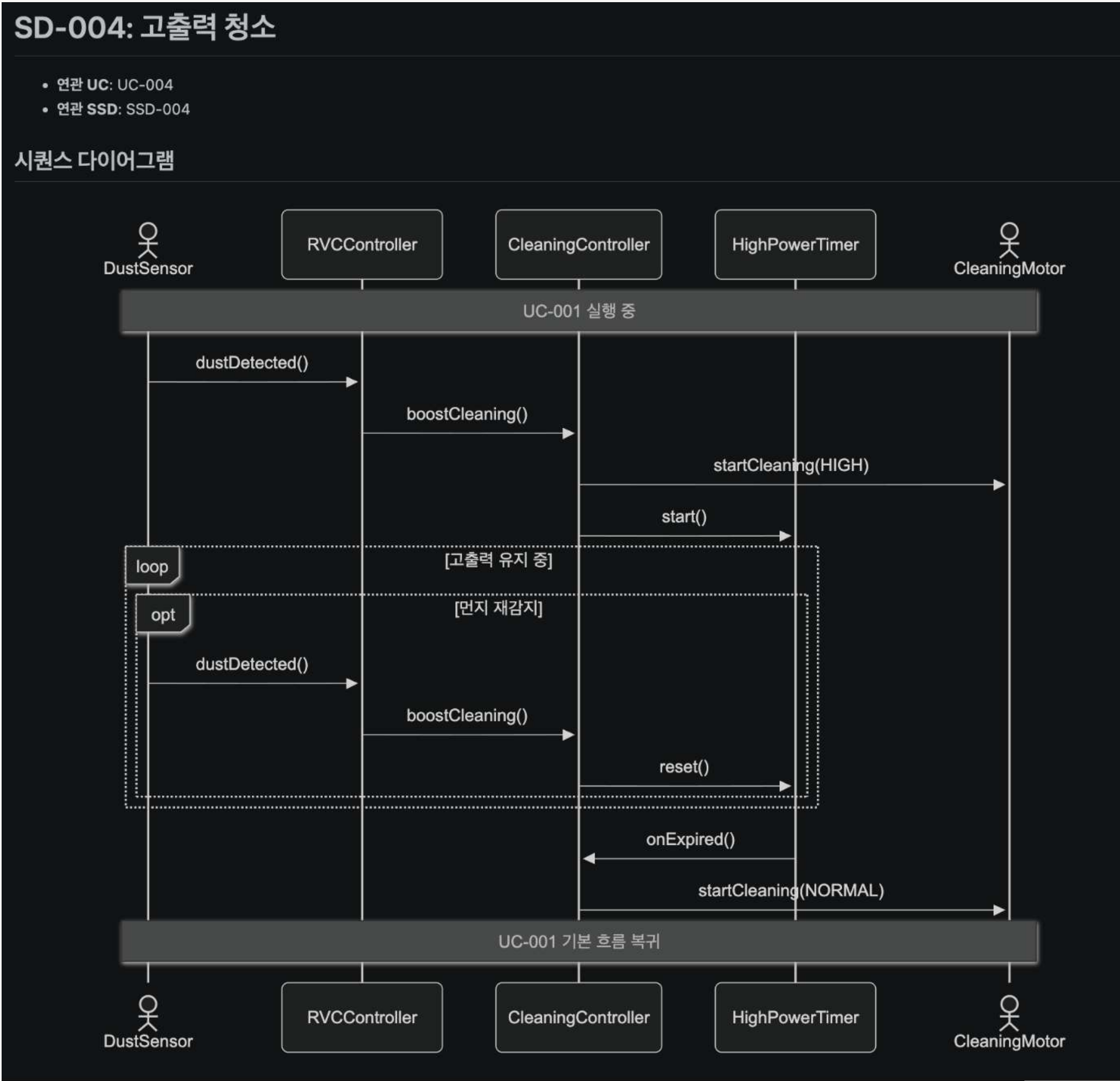
시퀀스 다이어그램



OOD - SD-003.md

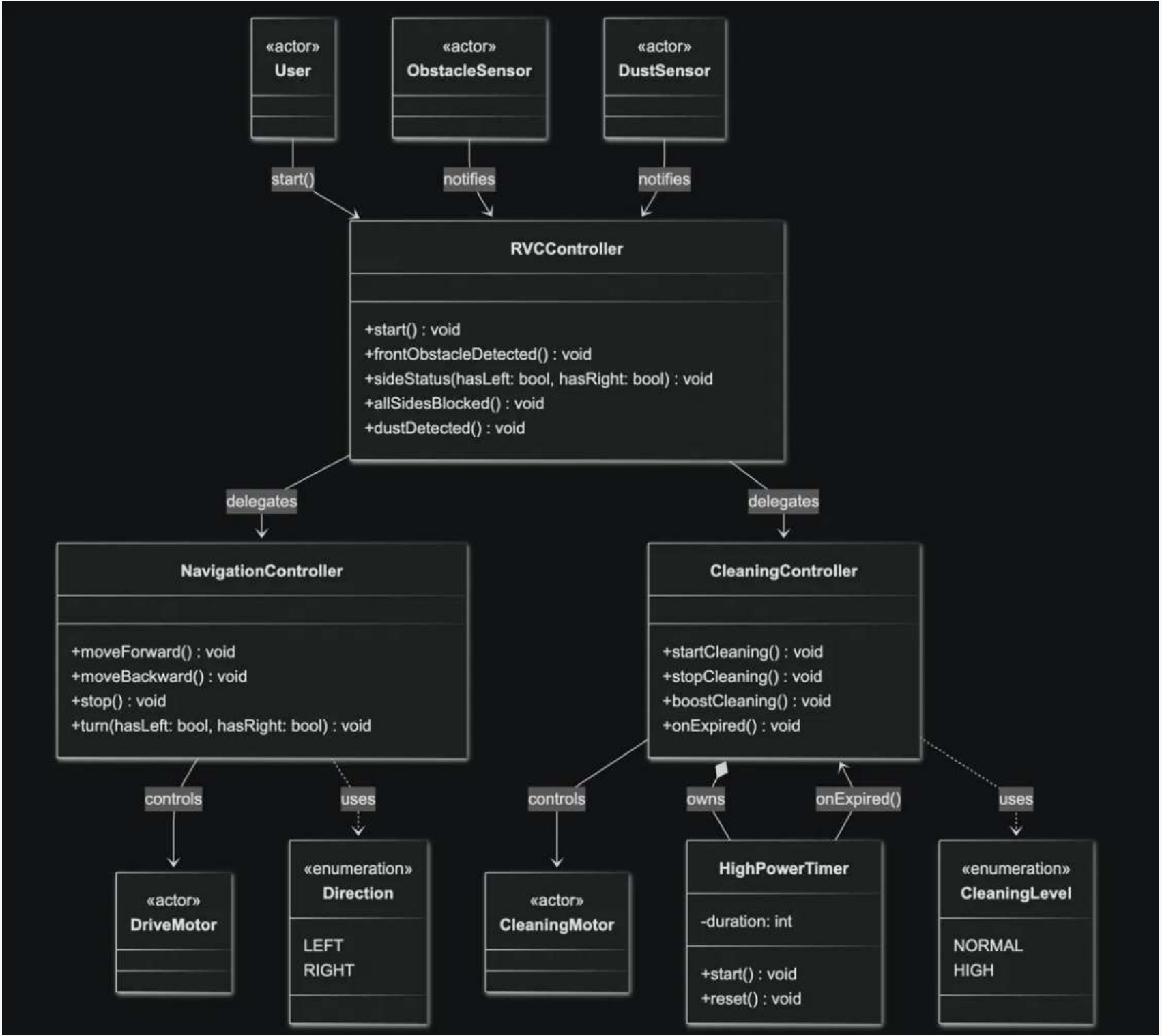


OOD - SD-004.md



OOD - class-diagram.md

arch/analysis/ 와 arch/design/sd/ 의 모든 문서를 읽고 클래스 다이어그램을 Mermaid 형식으로 arch/design/class/ 에 작성해줘



OOD - class-diagram.md (최종)

› 클래스 다이어그램에서 bool 파라미터로 방향을 표현하는 부분을 Direction enum을 활용하는 방식으로 개선해줘

› arch/design/class/ 의 클래스 다이어그램을 기반으로 C++ 코드를 src/ 폴더에 생성해줘. 객체지향 원칙을 철저히 준수해줘.

› src/ 폴더의 구현 코드와 arch/design/class/ 의 클래스 다이어그램을 비교해서 일관성을 확인하고 불일치하는 부분이 있으면 알려줘

=====

결론

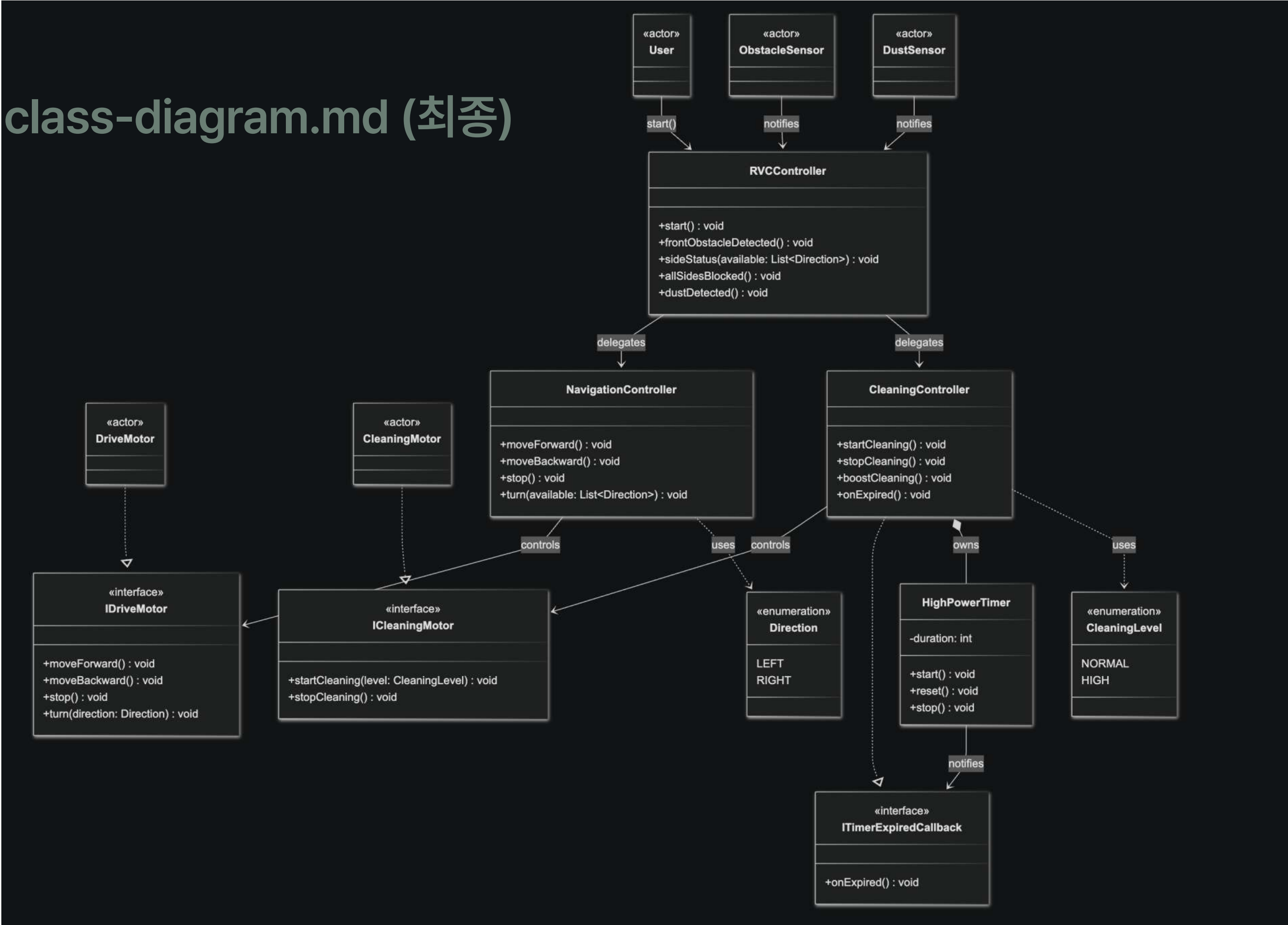
구조적 불일치는 2건으로, 코드가 DIP를 올바르게 적용한 결과입니다. 코드가 잘못된 게 아니라 클래스 다이어그램이 이 설계 결정을 반영하지 못한 상태입니다.

클래스 다이어그램을 수정하시겠습니까, 아니면 현재 상태 (다이어그램은 개념 수준, 코드는 DIP 적용)로 유지하시겠습니까?

* Cooked for 55s

› arch/design/class/ 의 클래스 다이어그램을 기준으로 arch/design/sd/, arch/analysis/ssd/, arch/analysis/domain/model.md 를 전부 일관성 있게 수정해줘

OOD - class-diagram.md (최종)



OOI - Code, Unit Test

› arch/design/class/ 의 클래스 다이어그램을 기반으로 C++ 코드를 src/ 폴더에 생성해줘. 객체지향 원칙을 철저히 준수해줘.

› src/ 폴더의 코드를 읽고 Stub을 활용한 단위 테스트를 tests/ 폴더에 C++로 작성해줘. 놓치는 코드 없이 모든 코드를 커버할 수 있도록 작성해줬으면 좋겠어

```
• Bash(build/rvc_tests 2>&1)
  └─ Error: Exit code 134
    Running main() from /private/tmp/googletest-20250910-5249-qce8ci/googletest-1.17.0/googletest/src/gtest_main.cc
    [=====] Running 35 tests from 4 test suites.
    [-----] Global test environment set-up.
    [-----] 6 tests from CleaningControllerTest
    [ RUN      ] CleaningControllerTest.startCleaning_callsMotorWithNormalLevel
    [       OK ] CleaningControllerTest.startCleaning_callsMotorWithNormalLevel (0 ms)
    [ RUN      ] CleaningControllerTest.stopCleaning_callsMotorStop
    [       OK ] CleaningControllerTest.stopCleaning_callsMotorStop (0 ms)
    [ RUN      ] CleaningControllerTest.boostCleaning_whenIdle_switchesToHighLevel
    ... +7 lines (ctrl+o to expand)
```

```
* Tempering... (4m 13s · ↓ 15.1k tokens · almost done thinking)
  └─ Tip: Use /btw to ask a quick side question without interrupting Claude's current work
```

› █

▸▸ accept edits on (shift+tab to cycle) · esc to interrupt

OOI - Code, Unit Test

● 35개 테스트 전부 통과했습니다.

작성된 테스트 구조 요약

Stub 파일 (tests/)

파일	목적
StubCleaningMotor.hpp	ICleaningMotor 구현체 - startCalls, stopCount 기록
StubDriveMotor.hpp	IDriveMotor 구현체 - forwardCount, backwardCount, stopCount, turnCalls 기록
StubTimerCallback.hpp	ITimerExpiredCallback 구현체 - callCount 기록

테스트 파일 (35개 케이스)

파일	케이스 수	주요 커버
CleaningControllerTest.cpp	6	start/stop/boost(idle/already boosting)/onExpired
NavigationControllerTest.cpp	8	moveForward/Backward/stop, turn(빈 목록 /단일 /복수 /중복 호출)
RVControllerTest.cpp	16	모든 상태 전환 (IDLE→CLEANING→STOPPED→REVERSING→CLEANING)
HighPowerTimerTest.cpp	5	미시작 /만료 /소멸자정지 /단발성 /reset 재시작

참고 - CleaningController 타이머 제약

HighPowerTimer가 CleaningController 내부에 직접 생성(duration 3000ms 고정)되므로, "boostCleaning → onExpired() 직접호출 → boostCleaning" 시나리오는 타이머 스레드가 살아있는 상태에서 timer_.start()를 재호출해 크래시가 발생합니다. 타이머 duration을 주입 가능하게 개선하면 이 시나리오도 단위 테스트로 커버할 수 있습니다.

› HighPowerTimer의 duration을 외부에서 주입 가능하게 개선해 줘

```
void HighPowerTimer::start() {
    // 이전 타이머가 자연 만료된 경우 스레드가 아직 join되지 않았을 수 있다.
    // 재대입 전에 join해야 std::terminate를 방지할 수 있다.
    if (timerThread_.joinable()) {
        timerThread_.join();
    }
    active_ = true;
    timerThread_ = std::thread(&HighPowerTimer::run, this);
}
```


OOI - Coverage

› gcov나 lcov를 사용해서 테스트 커버리지를 측정하고 결과를 보여줘

LCOV - code coverage report

Current view: [top level](#)

Test: [RVC Controller — Test Coverage](#)

Test Date: [2026-05-17 11:43:27](#)

Legend: Rating:

low: < 75 %

medium: >= 75 %

high: >= 90 %

Coverage

Total

Hit

Lines:

100.0 %

115

115

Functions:

85.0 %

40

34

Directory	Line Coverage			Function Coverage		
	Rate	Total	Hit	Rate	Total	Hit
src/	<div></div> 100.0 %	115	115	<div>85.0 %</div>	40	34

Generated by: [LCOV version 2.4-0](#)

함수 커버리지 85%의 원인

인터페이스 헤더 3개 (ICleaningMotor.hpp, IDriveMotor.hpp, ITimerExpiredCallback.hpp)에서 각각 50%로 집계된 것이 원인입니다. 이 파일들의 "미달" 함수는 가상 소멸자의 숨겨진 변형 (destructors like D0Ev, D1Ev, D2Ev)으로, 컴파일러가 자동 생성하는 코드입니다. 실제 비즈니스 로직 (startCleaning, moveForward 등)은 100% 커버되어 있습니다.

OOI - System Test, Simulator

1. simulator/ 폴더에 RVC 동작 시뮬레이터 구현 (실제 src/ 코드 활용 , 터미널에서 시각적으로 동작 확인 가능)
2. arch/system-tests/에 시뮬레이터 기반으로 Google Test 없이 시스템 테스트 케이스 30개 작성 (Positive 10개 , Negative 20개)
3. 테스트 실행 시 각 케이스가 커버되었는지 결과 리포트 출력

RVC 시스템 테스트 리포트

— Positive [UC 정상 동작 검증] —

✓ ST-P01 자동 청소 시작 - NORMAL + forward

✓ ST-P02 전방 장애물 감지 - stopCleaning + stop

✓ ST-P03 장애물 회피 우측 - 완전 시퀀스

✓ ST-P04 장애물 회피 좌측 - 완전 시퀀스

✓ ST-P05 장애물 회피 좌·우 모두 여유 - turn 1회 , 유효 방향

✓ ST-P06 삼면 장애물 후 우측 전진 - 순서 포함

✓ ST-P07 삼면 장애물 후 좌측 전진 - 순서 포함

✓ ST-P08 삼면 장애물 - 빈 sideStatus 반복 후 방향 확보

✓ ST-P09 먼지 감지 - HIGH 전환

✓ ST-P10 먼지 감지 후 타이머 만료 - NORMAL 자동 복귀

— Negative [상태·순서·격리 위반 검증] —

✓ ST-N01 IDLE에서 sideStatus({LEFT}) - turn 미호출

✓ ST-N02 IDLE에서 sideStatus({}) - 모든 모터 미호출

✓ ST-N03 CLEANING에서 sideStatus({LEFT}) - turn 미호출

✓ ST-N04 CLEANING에서 sideStatus({}) - 모터 추가 호출 없음

✓ ST-N05 REVERSING에서 sideStatus({}) - stop/turn/forward 없음

✓ ST-N06 REVERSING에서 sideStatus({}) 3회 - forward 0회

✓ ST-N07 dustDetected 2회 - startCleaning(HIGH) 1회만

✓ ST-N08 dustDetected - DriveMotor 호출 전혀 없음

✓ ST-N09 frontObstacleDetected - startCleaning 미호출

✓ ST-N10 allSidesBlocked - startCleaning 미호출

✓ ST-N11 sideStatus({L,R}) - turn 정확히 1회

✓ ST-N12 UC-002 순서 : stopCleaning → stop

✓ ST-N13 UC-002 순서 : turn → startCleaning(NORMAL)

✓ ST-N14 UC-003 순서 : forward → startCleaning(NORMAL)

✓ ST-N15 start() 후 DriveMotor.stop 미호출

✓ ST-N16 start() 후 moveForward 정확히 1회

✓ ST-N17 STOPPED → allSidesBlocked - stopCleaning 2회

✓ ST-N18 REVERSING → frontObstacleDetected → sideStatus 처리

✓ ST-N19 장애물 2회 연속 - turn 총 2회

✓ ST-N20 고출력 중 장애물 - stop 1회 , 마지막 호출 NORMAL

결과 전체 : 30 통과 : 30 실패 : 0

RVC SW Controller 시뮬레이터

시나리오 1: 자동 청소 시작 (UC-001)

▶ [User] start()
 ← [CleaningMotor] startCleaning(NORMAL)
 ← [DriveMotor] moveForward()

 시스템 상태
 상태 : **CLEANING** 구동 : 전진 ▶ 청소 : **NORMAL** Ⓞ

시나리오 2: 전방 장애물 - 우측 방향 전환 (UC-002)

(UC-001 실행 중)

▶ [User] start()
 ← [CleaningMotor] startCleaning(NORMAL)
 ← [DriveMotor] moveForward()

 (전방 장애물 감지)
▶ [ObstacleSensor] frontObstacleDetected()
 ← [CleaningMotor] stopCleaning()
 ← [DriveMotor] stop()

 (우측 여유 공간 확인 → 우측 전환)
▶ [ObstacleSensor] sideStatus({RIGHT})
 ← [DriveMotor] turn(RIGHT)
 ← [CleaningMotor] startCleaning(NORMAL)
 ← [DriveMotor] moveForward()

 시스템 상태
 상태 : **CLEANING** 구동 : 전진 ▶ 청소 : **NORMAL** Ⓞ

시나리오 3: 전방 장애물 - 좌측 방향 전환 (UC-002)

(UC-001 실행 중)

▶ [User] start()
 ← [CleaningMotor] startCleaning(NORMAL)
 ← [DriveMotor] moveForward()

 (전방 장애물 감지)
▶ [ObstacleSensor] frontObstacleDetected()
 ← [CleaningMotor] stopCleaning()
 ← [DriveMotor] stop()

OOI - System Test, Simulator

› simulator/ 폴더에 기존 시스템 테스트와 별개로 인터랙티브 터미널 시뮬레이터를 만들어줘 .

- 그리드 맵에서 RVC가 실제로 움직이는 걸 시각적으로 보여줘
- 키보드로 장애물 위치, 먼지 위치를 직접 설정할 수 있게
- 실제 src/ 코드를 연결해서 RVC가 스스로 판단하고 움직여야 해
- RVC가 자율적으로 청소하는 걸 실시간으로 볼 수 있어야 해

› 인터랙티브 시뮬레이터가 무한 루프로 화면이 계속 출력되는 버그가 있어 . 수정해줘

1. 먼지가 연속으로 감지될 때 처음만 HIGH 출력이 되고 이후엔 NORMAL로 돌아오는 문제 - 먼지가 계속 감지되는 동안은 HIGH 출력이 유지되어야 함
2. 회전 후 바로 앞에 있는 먼지를 청소하지 못하고 건너뛰는 문제
3. 청소 알고리즘 부재로 모서리에서 벽만 따라가는 문제 - 전체 공간을 효율적으로 커버할 수 있는 청소 알고리즘 추가

› simulator/interactive_simulator 에서 모서리에서 벽만 따라가는 청소 알고리즘 문제가 아직 해결 안 됐어 . 전체 공간을 효율적으로 커버할 수 있는 청소 알고리즘으로 수정하고 빌드까지 해줘

› 청소 알고리즘이 장애물로 둘러싸인 구역에 갇혀서 탈출 못하는 문제가 있어 . 방문한 셀을 기록해서 이미 청소한 구역은 다시 가지 않고 미청소 구역을 우선적으로 탐색하는 알고리즘으로 개선해줘

OOI - System Test, Simulator

RVC 인터랙티브 시뮬레이터 - 시뮬레이션 모드

상태

상태 : **CLEANING** 방향 : **EAST** 위치 : (11,7) 청소출력 : **HIGH** ★
청소 : 34/197 (17%)

P 일시정지/재개 E 편집모드 Q 종료

RVC 인터랙티브 시뮬레이터 - 시뮬레이션 모드

상태

상태 : **CLEANING** 방향 : **EAST** 위치 : (1,18) 청소출력 : **NORMAL** ◎
청소 : 67/197 (34%)

P 일시정지/재개 E 편집모드 Q 종료

RVC 인터랙티브 시뮬레이터 - 시뮬레이션 모드

상태

상태 : **CLEANING** 방향 : **WEST** 위치 : (3,9) 청소출력 : **HIGH** ★
청소 : 128/197 (64%)

P 일시정지/재개 E 편집모드 Q 종료

RVC 인터랙티브 시뮬레이터 - 시뮬레이션 모드

상태

상태 : **DONE** 방향 : **EAST** 위치 : (3,6) 청소출력 : **HIGH** ★
청소 : 197/197 (100%)

P 일시정지/재개 E 편집모드 Q 종료

OOI - CI 연동 및 Static Analysis

› GitHub Actions CI/CD yaml 파일을 만들어줘 . 빌드 , 단위 테스트 , 시스템 테스트 , cppcheck, clang-tidy 정적분석이 전부 포함되어야 해 . 그 다음 <https://github.com/Jae-ah/RVC-CONTROLLER> 에 push해 줘

- ✓ Build
- ✗ cppcheck
- ✓ clang-tidy
- ✓ Unit Tests
- ✓ System Tests

cppcheck

failed 1 minute ago in 15s

Search logs

Explain error

0s

Run cppcheck

```
1 ▶ Run cppcheck \
14 Checking src/CleaningController.cpp ...
15 1/7 files checked 1% done
16 Checking src/HighPowerTimer.cpp ...
17 2/7 files checked 3% done
18 Checking src/NavigationController.cpp ...
19 3/7 files checked 4% done
20 Checking src/RVCController.cpp ...
21 4/7 files checked 6% done
22 Checking simulator/interactive.cpp ...
23 simulator/interactive.cpp:827:20: style: Variable 'line' can be declared as reference to const [constVariableReference]
24     for (auto& line : log_.lines) {
25         ^
26 simulator/interactive.cpp:338:0: style: Consider using std::any_of algorithm instead of a raw loop. [useStlAlgorithm]
27         if (c == Cell::DUST || c == Cell::EMPTY) return;
28     ^
29 5/7 files checked 58% done
30 Checking simulator/main.cpp ...
31 6/7 files checked 69% done
32 Checking arch/system-tests/SystemTests.cpp ...
33 arch/system-tests/TestRunner.hpp:80:24: style: Variable 'r' can be declared as reference to const [constVariableReference]
34     for (auto& r : results_)
35         ^
36 7/7 files checked 100% done
37 simulator/interactive.cpp:70:0: style: The function 'moveCursor' is never used. [unusedFunction]
38 static void moveCursor(int row, int col) {
39     ^
40 simulator/interactive.cpp:207:0: style: The function 'motion' is never used. [unusedFunction]
41     Motion    motion()    const { return motion_; }
42     ^
43 simulator/interactive.cpp:208:0: style: The function 'lastTurn' is never used. [unusedFunction]
44     Direction lastTurn() const { return lastTurn_; }
45     ^
46 simulator/interactive.cpp:243:0: style: The function 'running' is never used. [unusedFunction]
47     bool        running() const { return running_; }
48     ^
49 nofile:0:0: information: Active checkers: 161/592 (use --checkers-report=<filename> to see details) [checkersReport]
50
51 Error: Process completed with exit code 1.
```

Chapter 3.

OOI - cppcheck

- › **cppcheck가 style 경고 때문에 CI에서 실패하고 있어.**
 1. 사용하지 않는 함수들 제거하거나
 2. cppcheck 옵션에서 style 체크를 제외하거나 둘 중 더 나은 방법으로 수정하고 push해줘

```
• 먼저 로컬에서 cppcheck를 실행해 어떤 경고가 나오는지 확인하겠습니다.

• Bash(cppcheck --enable=all --std=c++17 --suppress=missingIncludeSystem --suppress=unmatchedSuppression --error-exitcode=1 --inline-suppr -I src
  └ Error: Exit code 1
    Checking src/CleaningController.cpp ...
    1/7 files checked 1% done
    Checking src/HighPowerTimer.cpp ...
    2/7 files checked 3% done
    Checking src/NavigationController.cpp ...
    3/7 files checked 4% done
    Checking src/RVCController.cpp ...
    4/7 files checked 6% done
    Checking simulator/interactive.cpp ...
    ... +42 lines (ctrl+o to expand)

• 경고 내용이 명확합니다. 미사용 함수 제거 + 나머지 스타일 수정이 더 좋은 방법입니다. 코드를 직접 고치겠습니다.
```

```
• 이제 빌드 확인 후 cppcheck 재실행합니다.

• Bash(cmake --build build --target rvc_interactive rvc_system_tests --parallel 2>&1)
  └ Error: Exit code 2
    [ 16%] Building CXX object CMakeFiles/rvc_interactive.dir/simulator/interactive.cpp.o
    /Users/hajaea/Desktop/rvc-controller/simulator/interactive.cpp:216:9: error: use of undeclared identifier 'running_'
    216 |         running_ = true;
        |         ~~~~~
    1 error generated.
    make[3]: *** [CMakeFiles/rvc_interactive.dir/simulator/interactive.cpp.o] Error 1
    make[2]: *** [CMakeFiles/rvc_interactive.dir/all] Error 2
    make[1]: *** [CMakeFiles/rvc_interactive.dir/rule] Error 2
    make: *** [rvc_interactive] Error 2

    Read 1 file (ctrl+o to expand)

• Update(simulator/interactive.cpp)
  └ Removed 1 line
    213
    214     void startCleaning(CleaningLevel level) override {
    215         level_ = level;
    216 -         running_ = true;
    216         if (level == CleaningLevel::HIGH)
    217             log_.push(MAGENTA " [Clean] startCleaning(HIGH)" RESET);
    218     else
```

```
• 빌드 통과. cppcheck 최종 확인합니다.

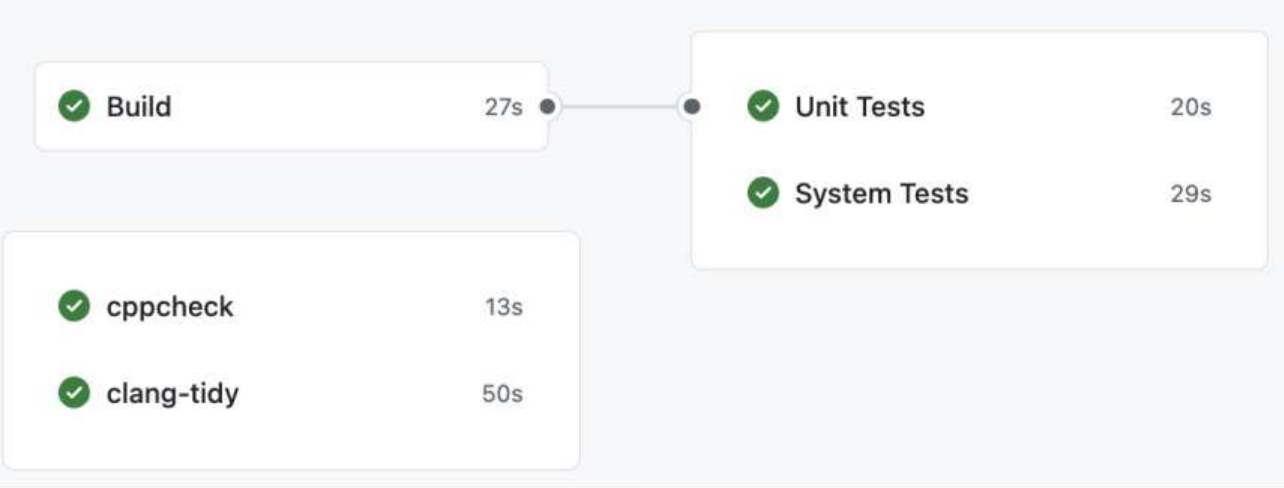
• Bash(cppcheck --enable=all --std=c++17 --suppress=missingIncludeSystem --suppress=unmatchedSuppression --error-exitcode=1 --inline-suppr -I src
  └ Error: Exit code 1
    Checking src/CleaningController.cpp ...
    1/7 files checked 1% done
    Checking src/HighPowerTimer.cpp ...
    2/7 files checked 3% done
    Checking src/NavigationController.cpp ...
    3/7 files checked 4% done
    Checking src/RVCController.cpp ...
    4/7 files checked 6% done
    Checking simulator/interactive.cpp ...
    ... +12 lines (ctrl+o to expand)

• useStlAlgorithm 경고가 남아있습니다. 중첩 루프라 인라인 suppression 위치 문제입니다. CI yaml에서 해당 경고를 전역 suppress하는 것이 깔끔합니다.
```

```
• 통과. CI yaml에 반영합니다.

• Update(.github/workflows/ci.yml)
  └ Added 1 line
    98         --suppress=missingIncludeSystem \
    99         --suppress=unmatchedSuppression \
    100        --suppress=useStlAlgorithm \
    101 +      --suppress=normalCheckLevelMaxBranches \
    102        --error-exitcode=1 \
    103        -I src \
    104        src/ \

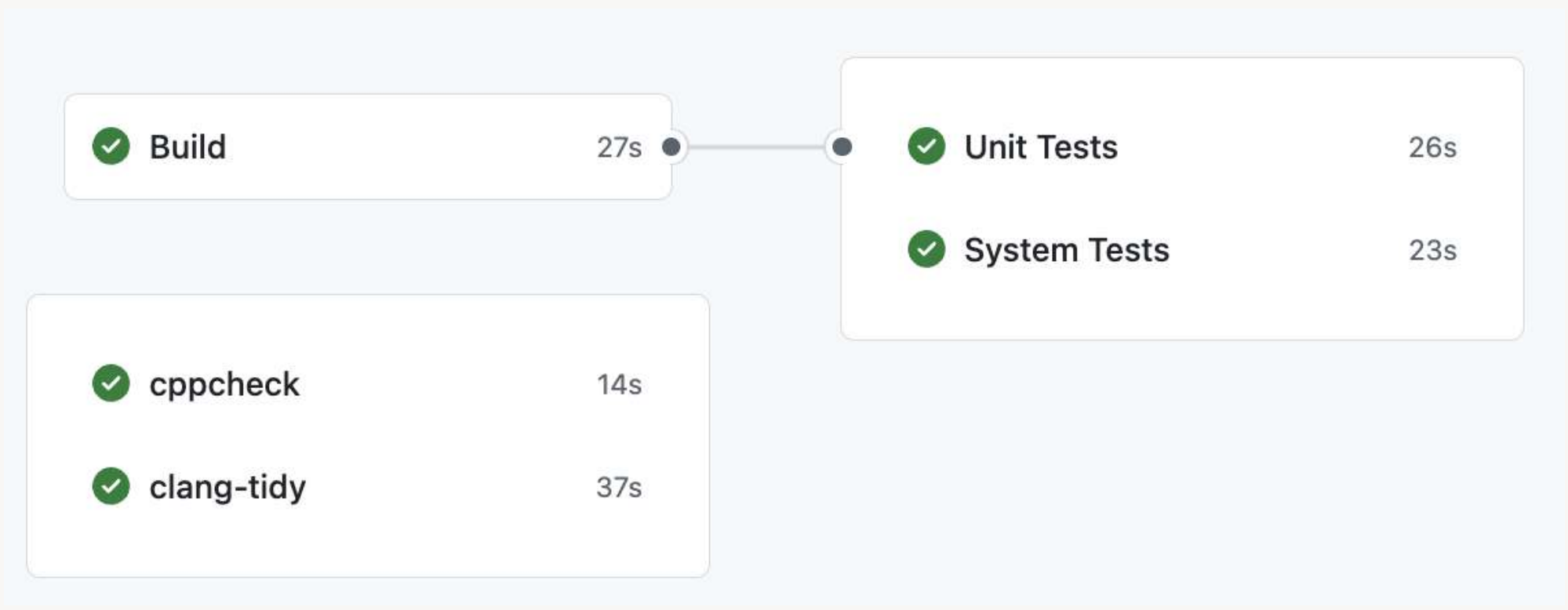
• 이제 commit하고 push합니다.
```



OOI - clang-tidy

```
/home/runner/work/RVC-CONTROLLER/RVC-CONTROLLER/src/NavigationController.cpp:26:15: warning:  
variable 'direction' is not initialized [cppcoreguidelines-init-variables]  
26 |     Direction direction;  
    |           ^
```

> NavigationController.cpp에서 Direction direction이 초기화 없이 선언된 문제를 수정하고 push해 줘



장단점 및 소감

[장점]

- 요구사항만 주면 OOAD 전 단계를 빠르게 진행 가능
- 문서(SRS, SDD)와 코드가 일관성 있게 자동 생성됨
- 테스트 코드, CI/CD까지 자동화로 개발 시간 단축
- 설계 오류를 대화하면서 즉시 발견하고 수정 가능

[단점]

- 컨텍스트가 길어지면 토큰 부족으로 내용이 날아감
- 완전히 믿으면 안 됨 - 직접 검토하지 않으면 잘못된 설계가 그대로 반영됨
- 세부적인 의도 전달이 어려울 때가 있음
- SSD에서 외부 액터를 내부 객체로 잘못 표현하는 등 OOAD 개념을 틀리게 적용하는 경우 있음

감사합니다

THANK YOU