

OOAD: #5 Vibe Coding

202211318 엄정석 / 202211334 이동훈 / 202211384 차민우 / 202211392 최환

Contents

1. Inception

- a. FR/NFR
- b. Use-Case
- c. SRS

2. OOA & OOD

- a. Use-Case Diagram
- b. System Sequence Diagram
- c. Domain Model
- d. Sequence Diagram
- e. Class Diagram

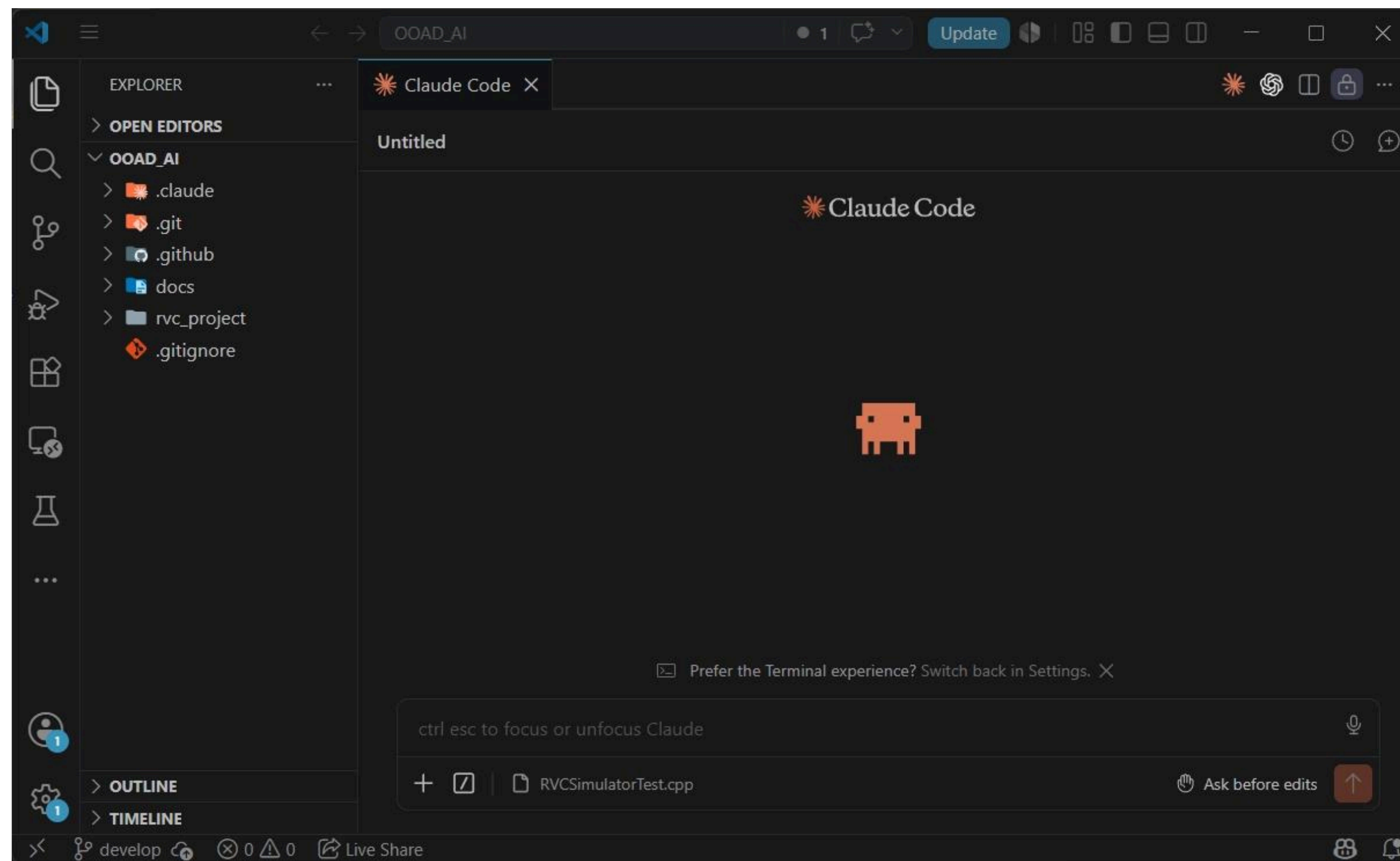
3. OOI

- a. Coding
- b. Unit Test
- c. System Test
- d. Static Analysis
- e. Simulators

OOAD with AI

수행과정

Environment



Model

Claude Code - Sonnet 4.6

사용 환경

Visual Studio Code의 엔트로픽 공식 extension을 사용해

모델이 전체 코드베이스에 접근 가능하도록 함

초기 프롬프트

과제 PPT에 있는 RVC SW의 내용들을 그대로 사용하였고, 추가적으로 OOAD 방식의 코딩임을 명시해줌.

“C++를 이용해 개발하고 Git을 통한 version control, Github Actions를 이용한 CI/CD 파이프라인 구축, Static Analysis 환경 설정, Unit Test와 이를 위한 system case 작성 등이 필요함. 먼저 컴파일/테스팅 환경 구축부터 진행. 어떤 구조를 선택하고, 사용할 것인지 제시.” 라는 프롬프트를 통해 전체 구조를 먼저 구성하고 이를 검증 후 작업을 시작할 수 있도록 함.

개발을 진행하며 이전에 생성한 문서들을 참조하여 진행하도록 하였고, 아들을 따로 디렉토리에 분리하고, 변경사항이 생길 때마다 이전의 문서들 또한 모두 수정되도록 함.

Prompt

OOAD를 적용한 RVC Controller SW를 만들 것.
앞으로 만드는 모든 산출물은 .MD파일로 저장하고 수정사항이 생길 때마다 기록하며 업데이트해줘.

RVC의 조건은 다음과 같음.

Preliminary Requirements for RVC SW Controller

- An RVC automatically cleans and mops household surface.
- It goes straight forward while cleaning.
- If its sensors found an obstacle, it stops cleaning, turns aside left or right, and goes forward with cleaning.
- If there are obstacles in both front, left and right, it move backward and turn aside left or right, and goes forward.
- If it detects dust, power up the cleaning for a while.
- We do not consider the detail design and implementation on HW controls.
- We only focus on the automatic cleaning function.

Future or Extended Requirements to Consider

- The RVC will add or change sensors.
- It will be able to circulate one spot for a while.
- It will have to communicate with a mobile app.
- It can do machine learning and inferring for more efficient cleaning.

우선 FR(Function Requirement), NFR(Non-function Requirement) 문서부터 작성해줘

Show less

● Thought for 2s >

● 메모리 시스템과 현재 디렉토리 구조를 먼저 확인한 후 요구사항 문서를 작성하겠습니다.

Inception

SRS

FR/NFR

Use Case
(Brief Format)

OOA

Use-Case
Diagram

Domain Model

System Sequence
Diagram

OOD

Sequence Diagram

Class Diagram

OOI

Code

Unit Test

System Test

Static Code
Analysis

Simulator

Inception

FR

Functional Requirements

FR-MOV : 이동(Movement) 관련

FR-SEN : 센서(Sensor) 관련

FR-CLN : 청소(Cleaning) 관련

FR-ERR : 오류 처리

FR-SIM : 시뮬레이터

FR-MOV

Functional Requirements

FR-MOV-01

RVC는 청소 시작 후 장애물·이벤트가 없으면 직진하며 청소한다.

FR-MOV-02

전방 장애물 감지 시 이동을 즉시 정지한다.

FR-MOV-03

정지 후 좌측 또는 우측으로 회전한다.
방향은 좌·우 센서 결과에 따라 결정된다.

FR-MOV-04

방향 전환 후 전진을 재개한다.

FR-MOV

Functional Requirements

FR-MOV-05

전방·좌·우 모두 장애물이면 후진한다.

FR-MOV-06

후진 후 후방(BACK) 방향으로 회전한다.

FR-MOV-07

회전 후 전진을 재개한다.

FR-SEN

Functional Requirements

FR-SEN-01

전방 장애물 감지 센서를 사용한다.

FR-SEN-02

좌측 장애물 감지 센서를 사용한다.

FR-SEN-03

우측 장애물 감지 센서를 사용한다.

FR-SEN-04

먼지 감지 센서를 사용한다.

FR-SEN

Functional Requirements

FR-SEN-05

센서 데이터는 매 폴링 사이클마다 Controller에 전달된다.

FR-SEN-06

새로운 센서 유형 추가 시 기존 코드 변경이 최소화되어야 한다.

FR-CLN

Functional Requirements

FR-CLN-01

RVC는 이동 중 지속적으로 청소한다.

FR-CLN-02

먼지 레벨이 `kDustThreshold(0.5)`를 초과하면 청소 파워를 즉시 증가시킨다.

FR-CLN-03

파워 증가 후 즉시 기본 파워로 복귀한다.

FR-CLN-04

장애물 정지 중에도 청소 유닛은 중단되지 않는다.

FR-ERR

Functional Requirements

FR-ERR-01

센서 오류가 `kErrorThreshold(3)`회 미만이면 일시적 오류로 판단하여 이전 유효 상태를 유지한다.

FR-ERR-02

센서 오류가 `kErrorThreshold(3)`회 이상 누적되면 영구 오류로 판단하여 이동을 즉시 정지한다.

FR-ERR-03

오류 카운터는 센서 ID별로 독립적으로 관리된다.

FR-SIM

Functional Requirements

FR-SIM-01

HW 없이 전체 SW 스택을 실행할 수 있는 Simulator를 제공한다.

FR-SIM-02

Simulator는 장애물·먼지·센서 오류 조건을 프로그래밍 또는 대화형으로 주입할 수 있어야 한다.

FR-SIM-03

대화형 CLI를 통해 명령어로 RVC 동작을 조작하고 상태를 확인할 수 있어야 한다.

FR-SIM-04

각 폴링 사이클(tick)은 독립적으로 실행 가능해야 한다.

FR-SIM-05

Simulator는 이동 상태, 마지막 회전 방향, 정지 횟수, 청소 활성 여부, 파워 레벨을 검사할 수 있어야 한다.

Inception

NFR

Non-Functional Requirements

NFR-PER : 성능(Performance)

NFR-REL : 신뢰성(Reliability)

NFR-MNT : 유지보수성(Maintainability)

NFR-EXT : 확장성(Extensibility)

NFR-SAF : 안전성(Safety)

NFR-PRT : 이식성(Portability)

NFR-TST : 테스트 가능성(Testability)

NFR-PER

Non-Functional Requirements

NFR-PER-01

센서 입력 감지 후 제어 명령 출력까지
응답 시간은 100ms 이하이어야 한다.

| 응답 시간 $\leq 100\text{ms}$ |

NFR-PER-02

먼지 감지 후 청소 파워 변경은 50ms 이내이어야 한다.

| 파워 변경 $\leq 50\text{ms}$ |

NFR-PER-03

Controller는 초당 최소 10회 이상 센서를 폴링할 수 있어야 한다.

| 폴링 주기 $\leq 100\text{ms}$ |

NFR-REL

Non-Functional Requirements

NFR-REL-01

센서 오류 발생 시 RVC는 안전한 상태(정지)로 전환되어야 한다.
| 오류 임계값 도달 → 즉시 정지 |

NFR-REL-02

SW 오류로 인한 비정상 종료 없이
연속 8시간 이상 운영 가능해야 한다.
| MTBF \geq 8시간 |

NFR-REL-03

장애물 감지 실패율은 1% 미만이어야 한다.
| 오감지율 $<$ 1% |

NFR-MNT

Non-Functional Requirements

NFR-MNT-01

각 기능 모듈은 독립적으로 수정·교체 가능한 구조로 설계되어야 한다.
| 모듈 간 결합도 Low |

NFR-MNT-02

임계값(kDustThreshold, kErrorThreshold)은
코드 수정 없이 변경 가능해야 한다.
| `static constexpr` 단일 정의 |

NFR-MNT-03

CI 파이프라인에서 정적 분석 위반이 1건이라도 있으면
빌드가 실패해야 한다.
| `WarningsAsErrors: '*'` |

NFR-EXT

Non-Functional Requirements

NFR-EXT-01

새로운 센서 유형 추가 시 기존 모듈 수정 없음.

| OCP — `Sensor` 추상 클래스 상속만으로 추가 |

NFR-EXT-02

Simulator에 새로운 Sim 컴포넌트 추가 시

`RVCSimulator` 배선만 변경.

| DIP — 인터페이스를 통한 주입 |

NFR-SAF

Non-Functional Requirements

NFR-SAF-01

모든 방향에 장애물이 있으면 후진 후 방향 전환을 시도한다.

| UC-05 전 시나리오 테스트 통과 |

NFR-SAF-02

비정상 센서값(범위 초과)은 무시하고 이전 유효 상태를 유지한다.

| `retainLastValidState()` |

NFR-PRT

Non-Functional Requirements

NFR-PRT-01

Controller 핵심 로직은 특정 HW 플랫폼에 종속되지 않아야 한다.
| 추상 인터페이스만 사용 — 직접 HW 접근 없음 |

NFR-PRT-02

HW 제어는 추상화된 인터페이스를 통해서만 접근해야 한다.
| `movementUnit_`, `cleaningUnit_`,
`*Sensor_` 포인터 경유 |

NFR-TST

Non-Functional Requirements

NFR-TST-01

모든 기능 요구사항은 자동화 테스트로 검증 가능해야 한다.
| UC별 테스트 케이스 100% 작성 |

NFR-TST-02

단위·통합·시스템 테스트가 HW 없이 실행 가능해야 한다.
| Mock(GMock) + Simulator 사용 |

NFR-TST-03

CI 파이프라인에서 모든 테스트가 자동으로 실행되어야 한다.
| GitHub Actions `ctest` |

Inception

Use Case (Brief Format)

Use Case (Brief Format) Prompt

```
</> c:\00_Folders\01_UNIV\26-1\...  
  
먼저 Fr/NFR에서 수정할 점을 알려줌  
  
FR의 미래 확장 요구사항, NFR의 범위 외 항목은 없애고 진행해줘.  
  
적은 FR/NFR에 기반하여 UseCase를 Brief format으로 적어줘.  
  
Show less  
  
● Edit c:\00_Folders\01_UNIV\26-1\03_OOAD\OOAD_AI\requirements.md  
Added 1 line  
| v1.0 | 2026-05-19 | 최초 작성 - FR / NFR 도출 | OOAD Team |  
□ | v1.1 | 2026-05-19 | FR-EXT(미래 확장 요구사항) 및 범위 외 항목(Out of Scope) 섹션 삭제 | OOAD Team |  
  
● Edit c:\00_Folders\01_UNIV\26-1\03_OOAD\OOAD_AI\requirements.md  
Added 2 lines  
### FR 분류 체계  
  
- **FR-MOV** : 이동(Movement) 관련  
- **FR-SEN** : 센서(Sensor) 관련  
- **FR-CLN** : 청소(Cleaning) 관련  
□  
□ > **Note:** 미래 확장 요구사항(FR-EXT)은 본 문서 범위에서 제외한다.
```

Use Case: Actor 정의

RVC System

자동 청소를 수행하는 로봇 청소기 시스템(Primary Actor)

Obstacle Sensor

전방·좌측·우측 장애물 감지 센서(Supporting Actor)

Dust Sensor

먼지 감지 센서(Supporting Actor)

**RVC는 완전 자율 시스템으로,
사람(Human Actor)이 직접 개입하는 Use Case는 존재하지 않는다.**

Use Case

Functional Requirements	Use-Case Number	Use-Case Name
FR-MOV-1, FR-CLN-01	UC-01	Perform Normal Cleaning
FR-SEN-01, FR-SEN-05	UC-02	Detect Obstacle (Front)
FR-MOV-02, FR-MOV-03, FR-MOV-04	UC-03	Avoid Front Obstacle
FR-SEN-02, FR-SEN-03, FR-SEN-05	UC-04	Detect Obstacle (Left & Right)
FR-MOV-05, FR-MOV-06, FR-MOV-07	UC-05	Avoid All-Direction Obstacle
FR-SEN-04, FR-SEN-05	UC-06	Detect Dust
FR-CLN-02, FR-CLN-03, FR-CLN-04	UC-07	Boost Cleaning Power
NFR-REL-01, NFR-SAF-02	UC-08	Handle Sensor Error

Use Case

Use-Case	1. Perform Normal Cleaning
Actors	RVC System
Summary	RVC는 시작 후 아무 장애물·먼지 이벤트가 없는 정상 상태에서 직진하며 청소 (흡입·걸레질)를 지속적으로 수행한다.

Use Case

Use-Case	2. Detect Obstacle (Front)
Actors	Obstacle Sensor → RVC System
Summary	전방 센서가 장애물을 감지하면 센서 데이터를 Controller에 즉시 전달하여 장애물 존재를 알린다.

Use Case

Use-Case	3. Avoid Front Obstacle
Actors	RVC System
Summary	전방 장애물 감지 시 RVC는 이동을 정지하고, 좌측 또는 우측으로 방향을 전환한 뒤 전진을 재개하여 청소를 계속한다. 방향 전환 방향(좌/우)은 좌·우 센서 상태에 따라 결정된다.

Use Case

Use-Case	4. Detect Obstacle (Left & Right)
Actors	Obstacle Sensor → RVC System
Summary	좌측 및 우측 센서가 장애물을 감지하면 해당 정보를 Controller에 전달한다. UC-03 실행 중 방향 결정 또는 UC-05 진입 조건 판단에 활용된다.

Use Case

Use-Case	5. Avoid All-Direction Obstacle
Actors	RVC System
Summary	전방·좌측·우측 모두 장애물이 감지된 경우 RVC는 후방으로 이동한 뒤, 좌측 또는 우측으로 방향을 전환하고 전진을 재개하여 청소를 계속한다.

Use Case

Use-Case	6. Detect Dust
Actors	Dust Sensor → RVC System
Summary	먼지 센서가 먼지를 감지하면 해당 데이터를 Controller에 즉시 전달하여 고농도 먼지 존재를 알린다.

Use Case

Use-Case	7. Boost Cleaning Power
Actors	RVC System
Summary	먼지 감지 시 RVC는 청소 파워를 즉시 증가시키고, 설정된 시간이 경과하면 기본 청소 파워로 자동 복귀한다. 파워 부스트 중에도 이동 및 청소는 중단 없이 계속된다.

Use Case

Use-Case	8. Handle Sensor Error
Actors	Obstacle Sensor / Dust Sensor → RVC System
Summary	센서에서 비정상 값(범위 초과, 통신 오류 등)이 수신되면 RVC는 해당 값을 무시하고 이전 유효 상태를 유지한다. 오류가 지속되면 안전을 위해 이동을 정지한다.

Inception

SRS

SRS

Purpose, Scope, Glossary

Purpose

본 문서는 RVC(Robot Vacuum Cleaner) Controller SW의 시스템 요구사항을 정의한다.
기능 요구사항(FR), 비기능 요구사항(NFR), 소프트웨어 설계 제약, 구현 산출물, 검증 요구사항을 포괄하며,
구현이 완료된 시점을 기준으로 작성된 As-Built SRS이다.

Scope

Controller SW 핵심 로직

이동·청소·센서·오류 처리

HW 추상화 인터페이스

Sensor, MovementUnit, CleaningUnit

대화형 Simulator

CLI 기반 수동 조작 및 시스템 테스트

CI/CD 파이프라인

GitHub Actions 기반 빌드·테스트·정적 분석

실제 HW 드라이버 구현, 모바일 앱·네트워크 통신, 지도 생성·경로 계획

범위 외 (HW 추상화만 정의)

Glossary

RVC : Robot Vacuum Cleaner — 자동 청소 로봇

Controller : RVC의 두뇌 역할 SW 클래스. 센서 입력을 받아 이동·청소 유닛을 제어

MovementUnit : RVC 이동을 담당하는 HW 추상화 인터페이스

CleaningUnit : 흡입·걸레질을 담당하는 HW 추상화 인터페이스

ObstacleSensor : 장애물 감지 센서 HW 추상화 인터페이스

DustSensor : 먼지 감지 센서 HW 추상화 인터페이스

RVCSimulator : Sim 컴포넌트로 전체 스택을 구성한 대화형 테스트 도구

pollSensors() : 한 번의 HW 타이머 사이클에 해당하는 Controller의 센서 폴링 메서드

tick : Simulator에서 pollSensors() 한 번을 실행하는 조작 단위

kDustThreshold : 먼지 부스트 발동 임계값 (0.5 / 범위 0.0~1.0)

kErrorThreshold : 영구 센서 오류 판정 누적 횟수 (3회)

SRS

Software Design Constraints

언어 및 표준

Software Design Constraints

언어 : C++17

컴파일러 : GCC 15 (Linux CI), MSVC / MinGW (로컬 개발)

빌드 시스템 : CMake \geq 3.20

테스트 프레임워크 : Google Test / Google Mock v1.14.0
(FetchContent)

아키텍처 제약

Software Design Constraints

HW 추상화 : 모든 HW 접근은 순수 가상 클래스(`= 0`) 경우 필수

소유권 : Controller는 Raw Non-owning 포인터 사용 (소유권 없음)

네임스페이스 : 생산 코드: `namespace rvc`, Sim 코드: `namespace rvc::sim`, 테스트 Mock: `namespace rvc::test`

클래스 설계 : Abstract interface는 `cppcoreguidelines-special-member-functions` 제외 허용

정적 분석 규칙

Software Design Constraints

경고를 오류로 처리

``WarningsAsErrors: '*``

이름 규칙 — 클래스/구조체 : ``CamelCase``

이름 규칙 — 메서드/함수 : ``camelBack` (lowerCamelCase)`

이름 규칙 — Enum 상수 : ``UPPER_CASE``

이름 규칙 — private 멤버 : ``camelBack` + suffix `_``

이름 규칙 — static constexpr : ``CamelCase` + prefix `k``

반환값 무시 금지 : ``modernize-use-nodiscard`` — 모든 non-void
getter에 ``[[nodiscard]]`` 필수

열거형 기반 타입 : ``std::uint8_t`` 사용 (``performance-enum-size``)

비활성화 검사 : ``cppcoreguidelines-special-member-functions``,
``bugprone-easily-swappable-parameters`` 등

cppcheck 규칙

Software Design Constraints

활성화 범주

``warning`, `style`, `performance`, `portability``

오류 시 종료 코드

``--error-exitcode=1``

인라인 억제

``// cppcheck-suppress <id>` 허용`

제외

``unusedFunction` (추상 인터페이스 false-positive)`

OOA & OOD

Use-Case Diagram

Use Case Diagram Prompt

```
</> c:\00_Folders\01_UNIV\26-1\...
```

이를 기반으로 한 UseCase Diagram을 그려줘

- `use_case_diagram.md` 생성 완료. 주요 내용 요약:

다이어그램 구조

Actors (2개)

- `Obstacle Sensor` — 전방·좌·우 장애물 감지 입력
- `Dust Sensor` — 먼지 감지 입력

관계 정의

관계	방향	대상
<code><<extend>></code>	확장 UC → UC-01	UC-03, UC-05, UC-07
<code><<include>></code>	UC-03 →	UC-02, UC-04
<code><<include>></code>	UC-05 →	UC-02, UC-04
<code><<include>></code>	UC-07 →	UC-06
Association	Sensor →	UC-02, UC-04, UC-06, UC-08

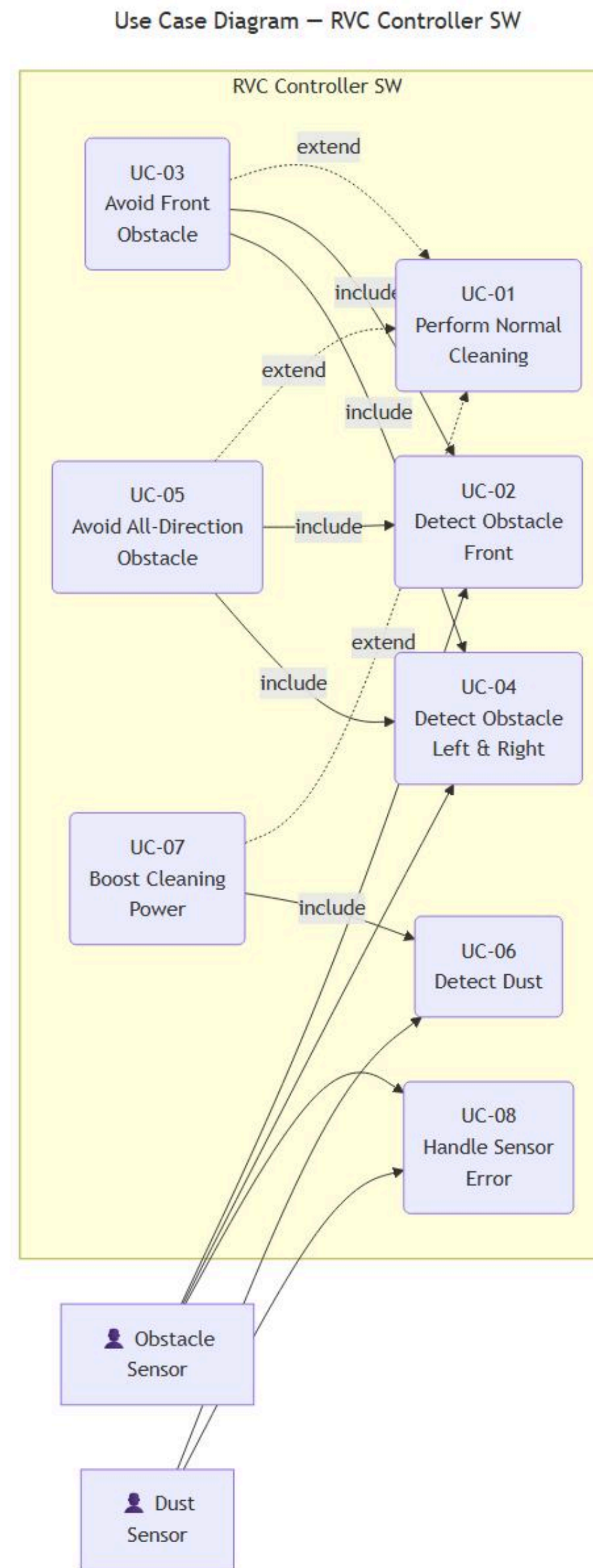
렌더링 방법

- VS Code: PlantUML 확장 설치 후 Alt+D 미리보기
- 온라인: plantuml.com 에 코드 붙여넣기

다음은 **Domain Model** 또는 **Fully-Dressed Use Case** 작성을 진행할 수 있습니다.

[OOA & OOD]

Use-Case Diagram



OOA & OOD

System Sequence Diagram

System Sequence Diagram Prompt

```
</> c:\00_Folders\01_UNIV\26-1\...
이를 기반으로 한 System Sequence Diagram도 그려줘

SSD 5개 시나리오
```

SSD	시나리오	핵심 흐름
SSD-01	Normal Cleaning	startCleaning() → loop: pollSensors() + moveForward() + clean()
SSD-02	Avoid Front Obstacle	reportObstacle(FRONT) → stopMovement() → checkSideObstacles() → turnToFreeDirection() → moveForward()
SSD-03	Avoid All-Direction	reportObstacle(FRONT/LEFT/RIGHT) → stopMovement() → moveBackward() → turnToFreeDirection() → moveForward()
SSD-04	Boost Cleaning Power	reportDust(dustLevel) → boostCleaningPower() → (boostDuration 경과) → restoreNormalPower()
SSD-05	Handle Sensor Error	reportSensorError(sensorId) → validateSensorData() → (일시적: 이전 상태 유지 / 지속: stopMovement())

```
도출된 System Operations (4개)

다음 Design 단계에서 Controller 클래스의 공개 인터페이스 후보로 사용됩니다.

startCleaning()
reportObstacle(position: SensorPosition)
reportDust(dustLevel: float)
reportSensorError(sensorId: String)
```

```
</> c:\00_Folders\01_UNIV\26-1\...
이전에 만든 UseCase를 기반으로 1대1 대응하도록 SSD를 수정해줘

변경 내용

SSD 5개 → 8개 (UC와 1:1 대응)
```

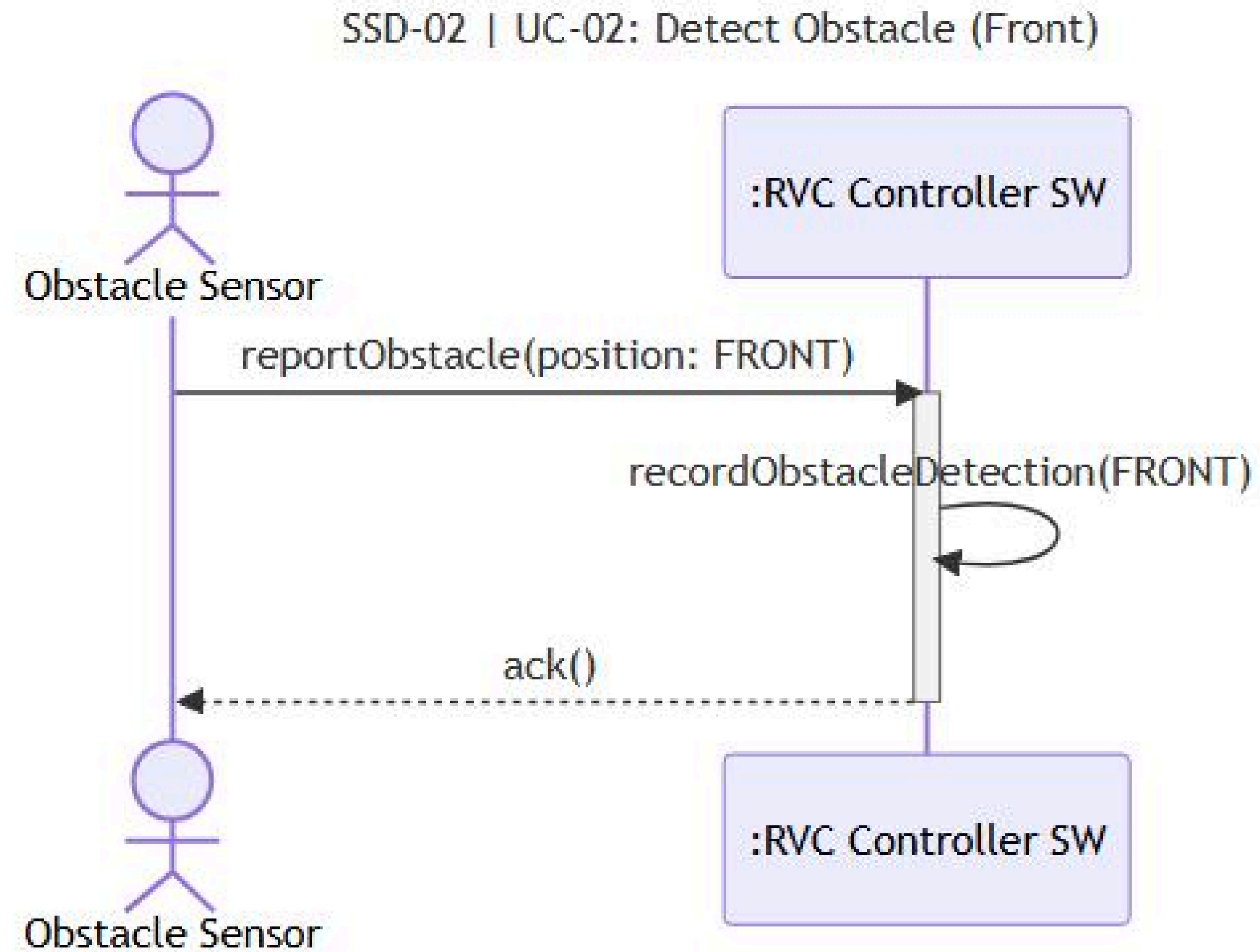
SSD	핵심 변경
SSD-02 (신규 분리)	Obstacle Sensor → reportObstacle(FRONT) — 감지 자체를 독립 SSD로 분리
SSD-03 (재정의)	UC-02 결과를 사전조건으로 받아 avoidFrontObstacle() 내부 흐름만 표현
SSD-04 (신규 분리)	Obstacle Sensor → reportObstacle(LEFT/RIGHT) — 좌우 감지 독립 SSD
SSD-05 (재정의)	UC-02+UC-04 결과를 사전조건으로 받아 avoidAllDirectionObstacle() 흐름 표현
SSD-06 (신규 분리)	Dust Sensor → reportDust(dustLevel) — 먼지 감지 독립 SSD
SSD-07 (재정의)	UC-06 결과를 사전조건으로 받아 boostCleaningPower() 흐름 표현

```
각 SSD에 사전조건(Precondition) / 사후조건(Postcondition) 명시
System Operation 4개 → 7개로 확장

startCleaning()
reportObstacle(position) ← 외부 입력
avoidFrontObstacle() ← 내부 트리거
avoidAllDirectionObstacle() ← 내부 트리거
reportDust(dustLevel) ← 외부 입력
boostCleaningPower() ← 내부 트리거
reportSensorError(sensorId) ← 외부 입력

다음은 Operation Contract 또는 Design Class Diagram 중 어느 것을 진행할까요?
```

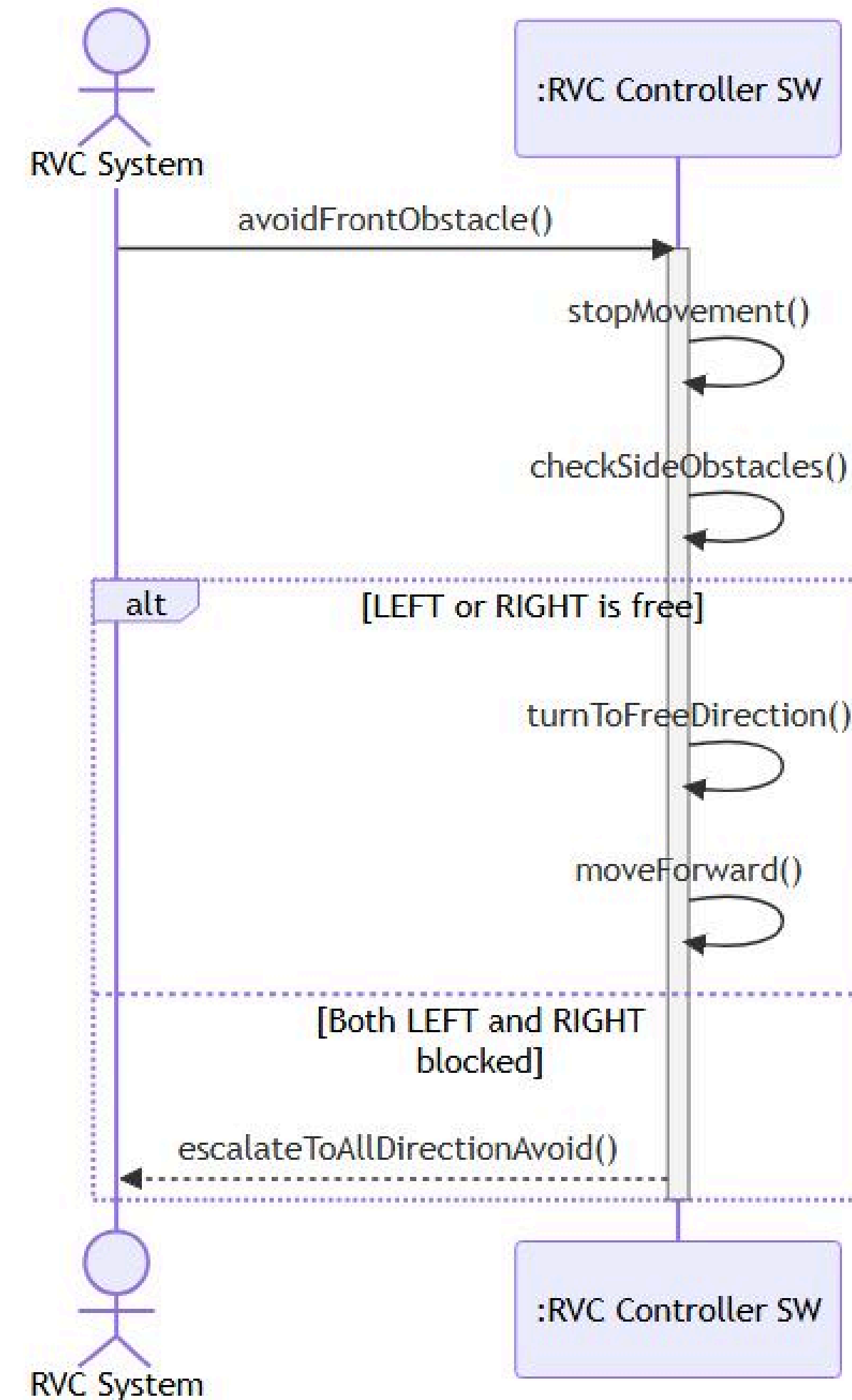

System Sequence Diagram

SSD-02 : UC-02 Detect Obstacle (Front)

System Sequence Diagram

SSD-03 : UC-03 Avoid Front Obstacle

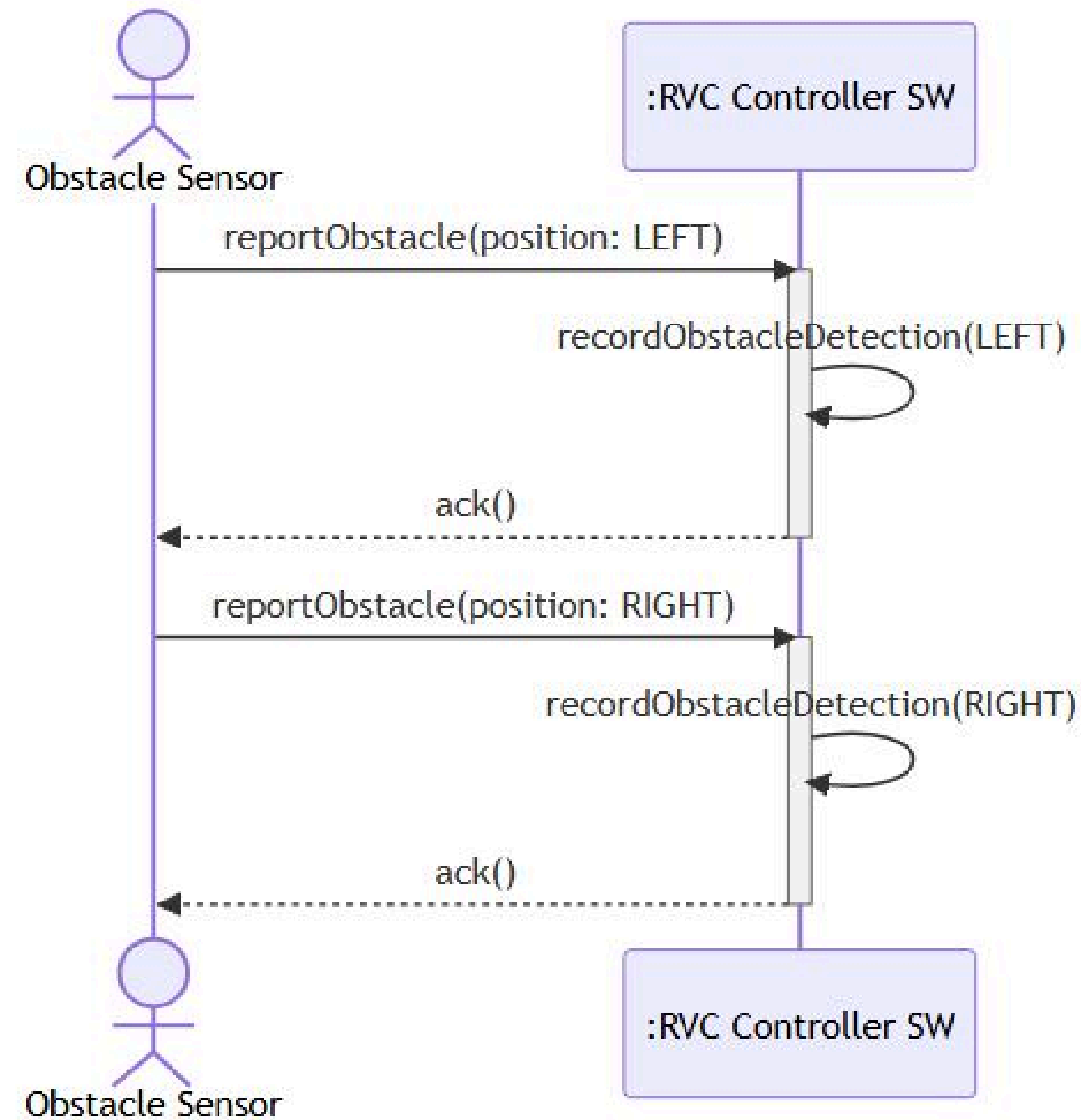
SSD-03 | UC-03: Avoid Front Obstacle



System Sequence Diagram

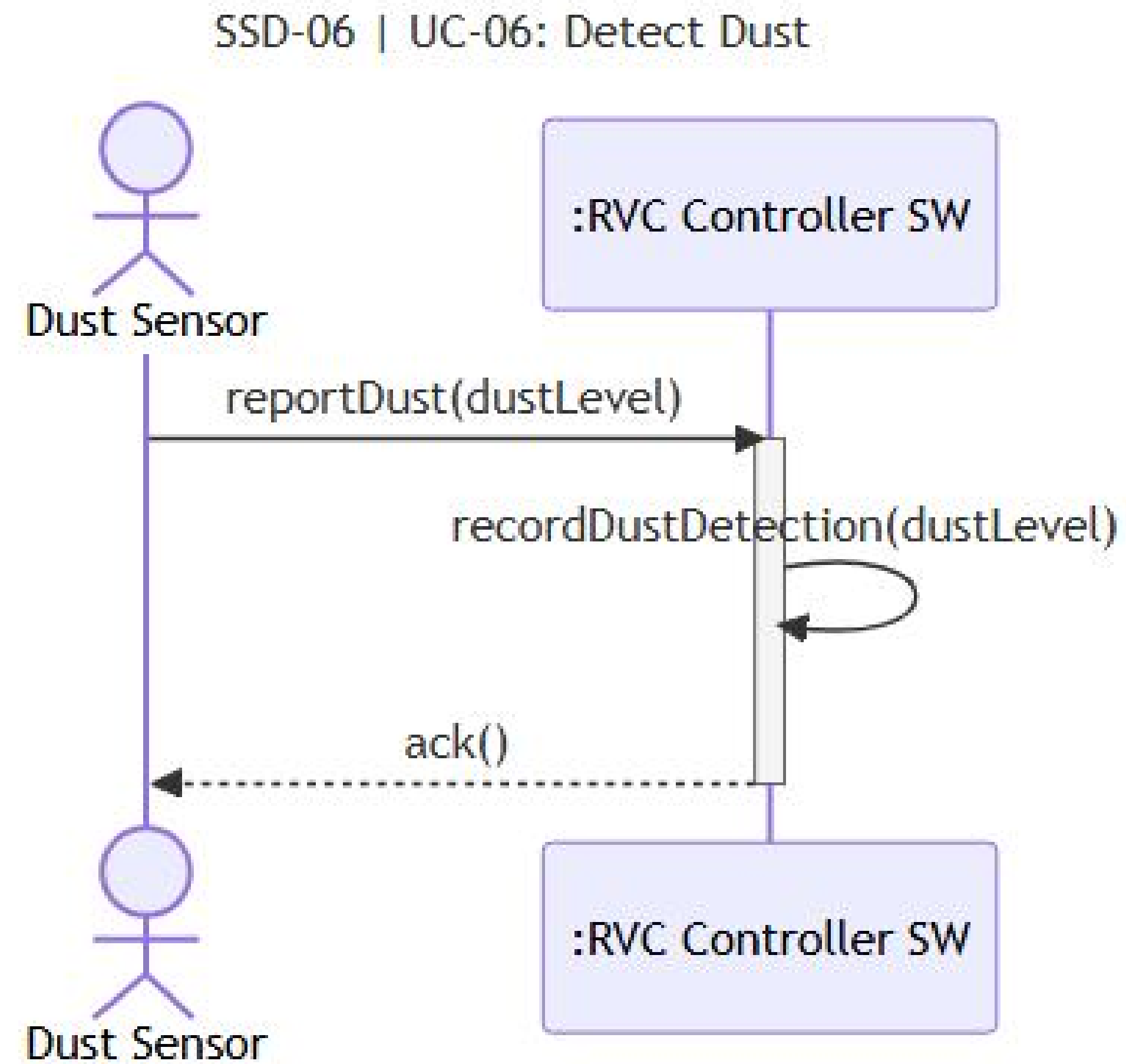
SSD-04 : UC-04 Detect Obstacle (Left & Right)

SSD-04 | UC-04: Detect Obstacle (Left & Right)



System Sequence Diagram

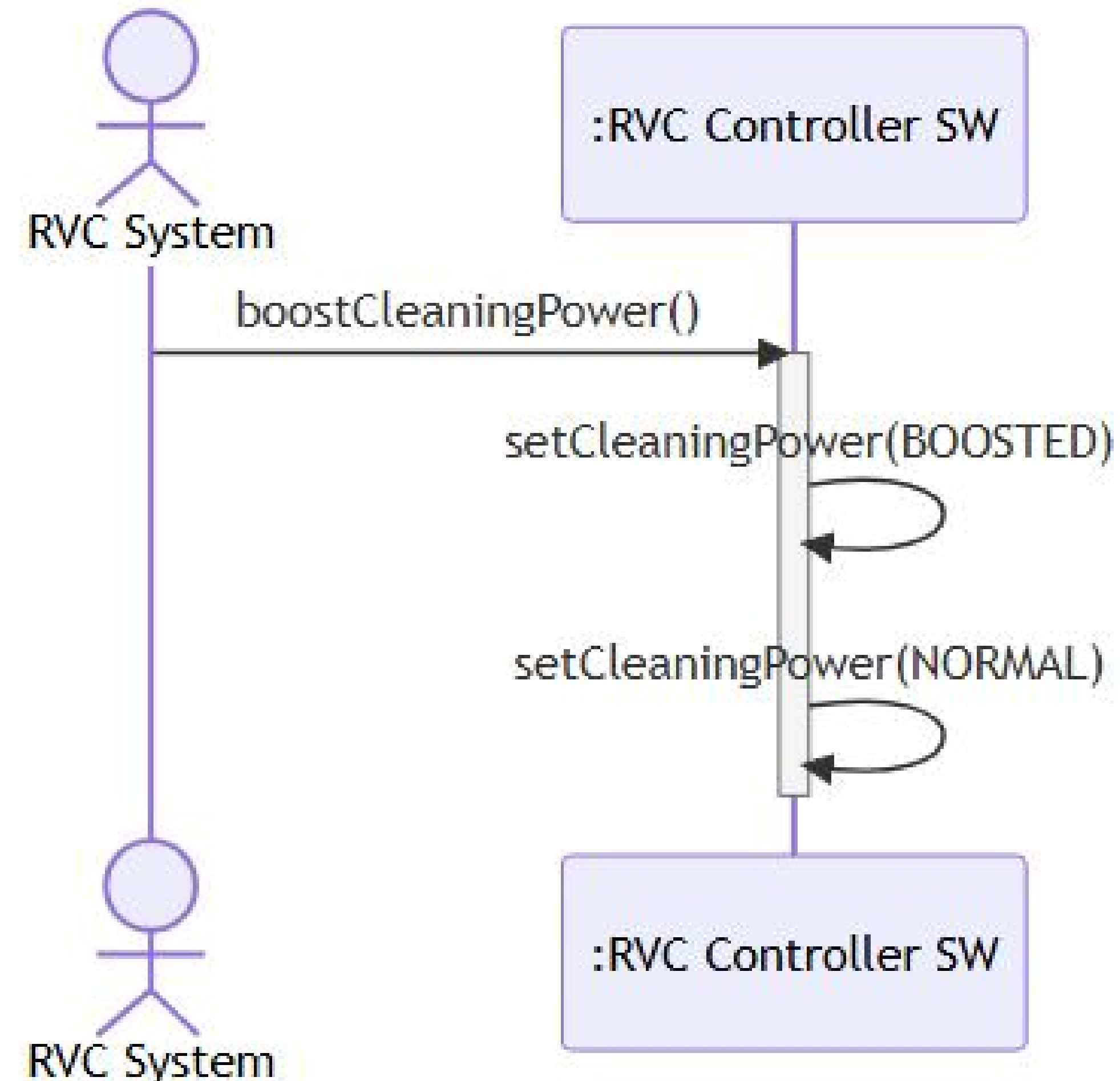
SSD-06 : UC-06 Detect Dust



System Sequence Diagram

SSD-07 : UC-07 Boost Cleaning Power

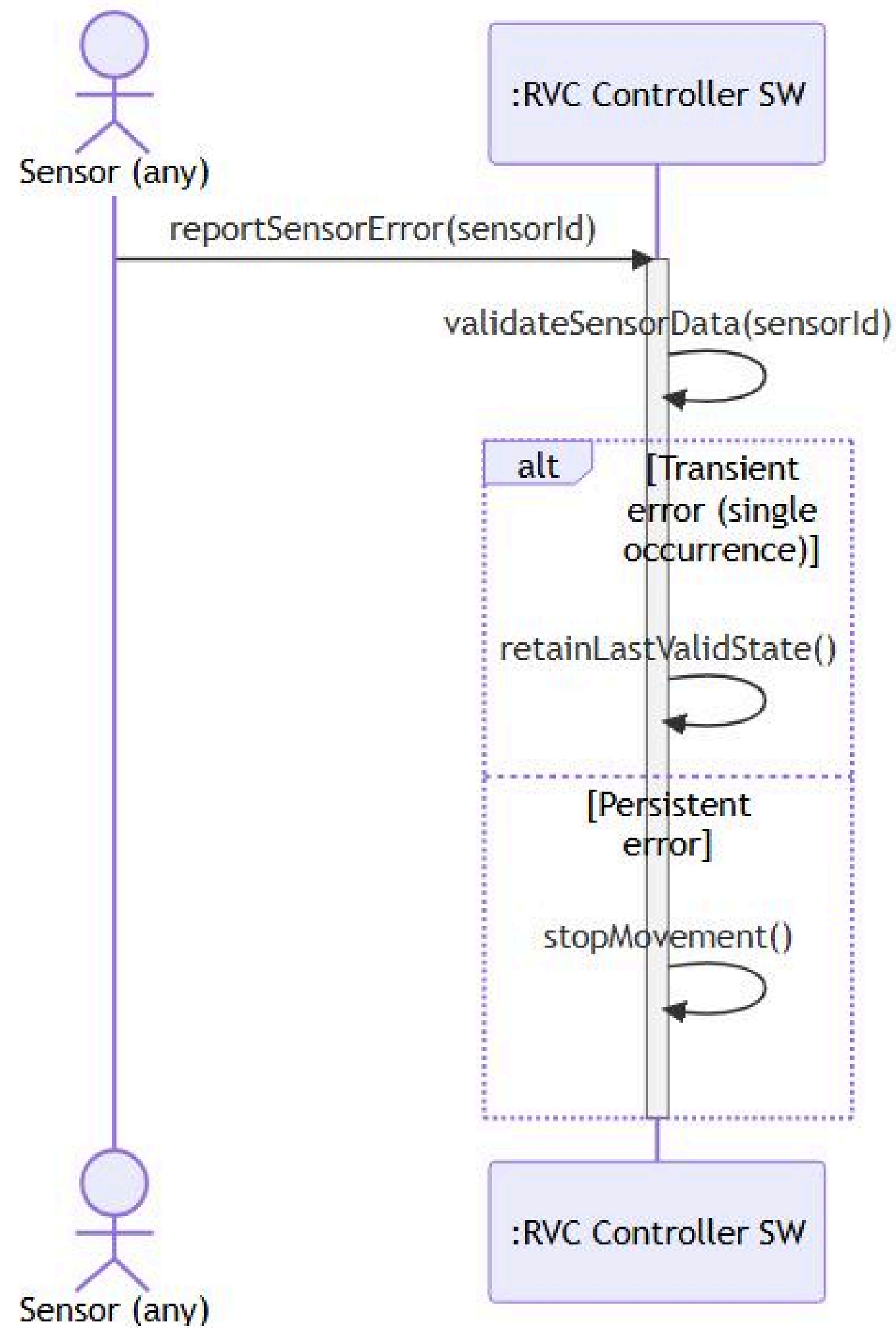
SSD-07 | UC-07: Boost Cleaning Power



System Sequence Diagram

SSD-08 : UC-08 Handle Sensor Error

SSD-08 | UC-08: Handle Sensor Error



OOA & OOD

Domain Model

Domain Model Prompt

```
</> c:\00_Folders\01_UNIV\26-1\...
```

도메인 모델도 그려줘

Domain Model 구성

9개 도메인 클래스

RVC (root)

- ├─ Controller ← 센서 감시 + 이동/청소 명령
- ├─ MovementUnit ← 방향·속도 관리
- └─ CleaningUnit ← 파워 레벨·부스트 관리

Sensor (abstract)

- ├─ ObstacleSensor ← FRONT / LEFT / RIGHT 3개 인스턴스
- └─ DustSensor ← 먼지 농도 감지

Obstacle ← 환경 개념 (ObstacleSensor가 감지)

Dust ← 환경 개념 (DustSensor가 감지)

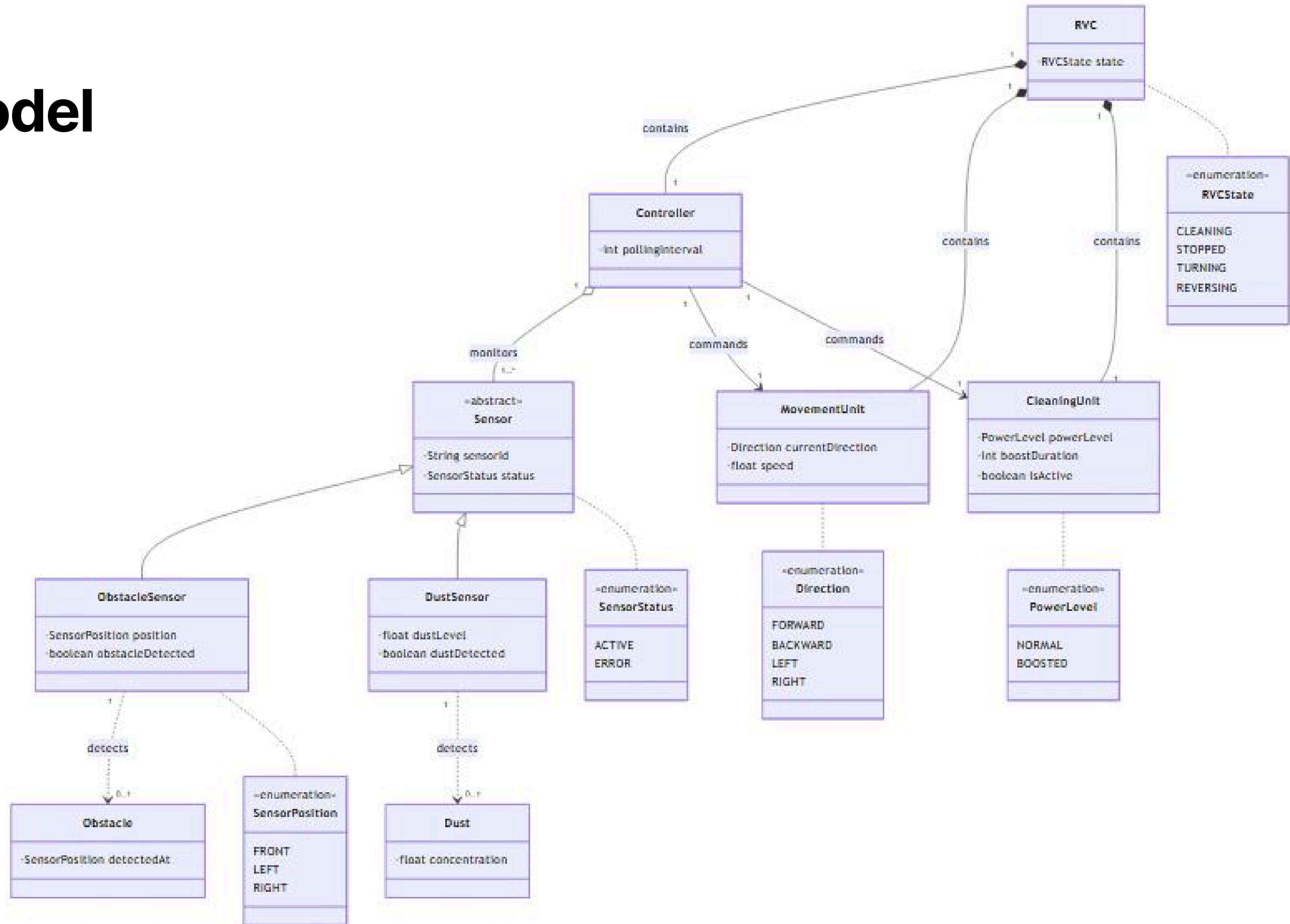
관계 종류 4가지

관계	사용처
Composition (*--)	RVC → Controller / MovementUnit / CleaningUnit
Aggregation (o--)	Controller → Sensor (교체 가능)
Generalization (^< --)	
Dependency (..>)	센서 → 환경 객체 (감지 시 존재)

6개 도메인 규칙 (DR) — 센서 에러 시 강제 정지, 부스트 자동 복귀, 청소 지속 등 핵심 도메인 제약 명세

OOA & OOD Domain Model

Domain Model - RVC Controller SW



OOA & OOD

Sequence Diagram

Sequence Diagram Prompt

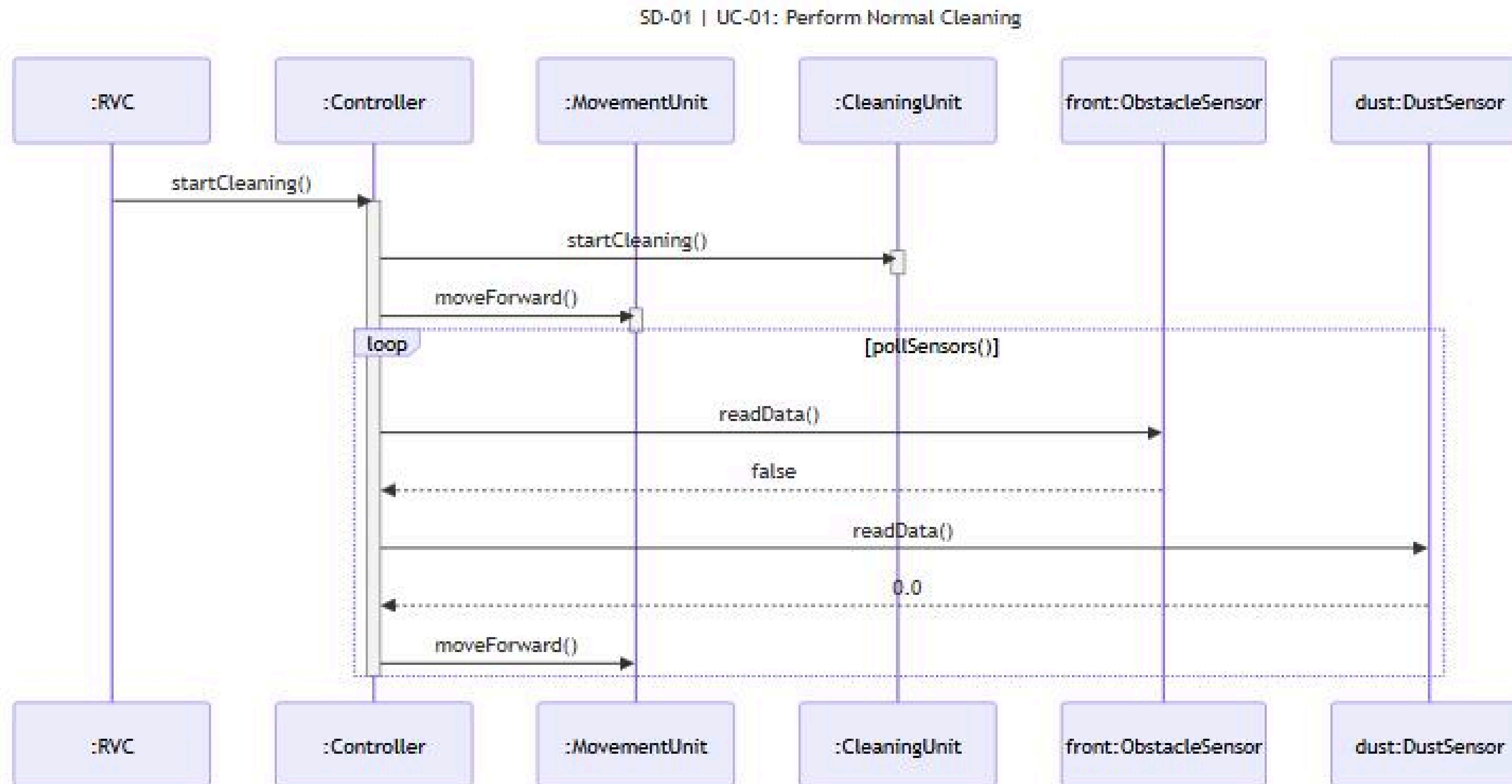
```
</> c:\00_Folders\01_UNIV\26-1\...
```

이를 기반으로 Sequence Diagram을 만들어줘
Note없이

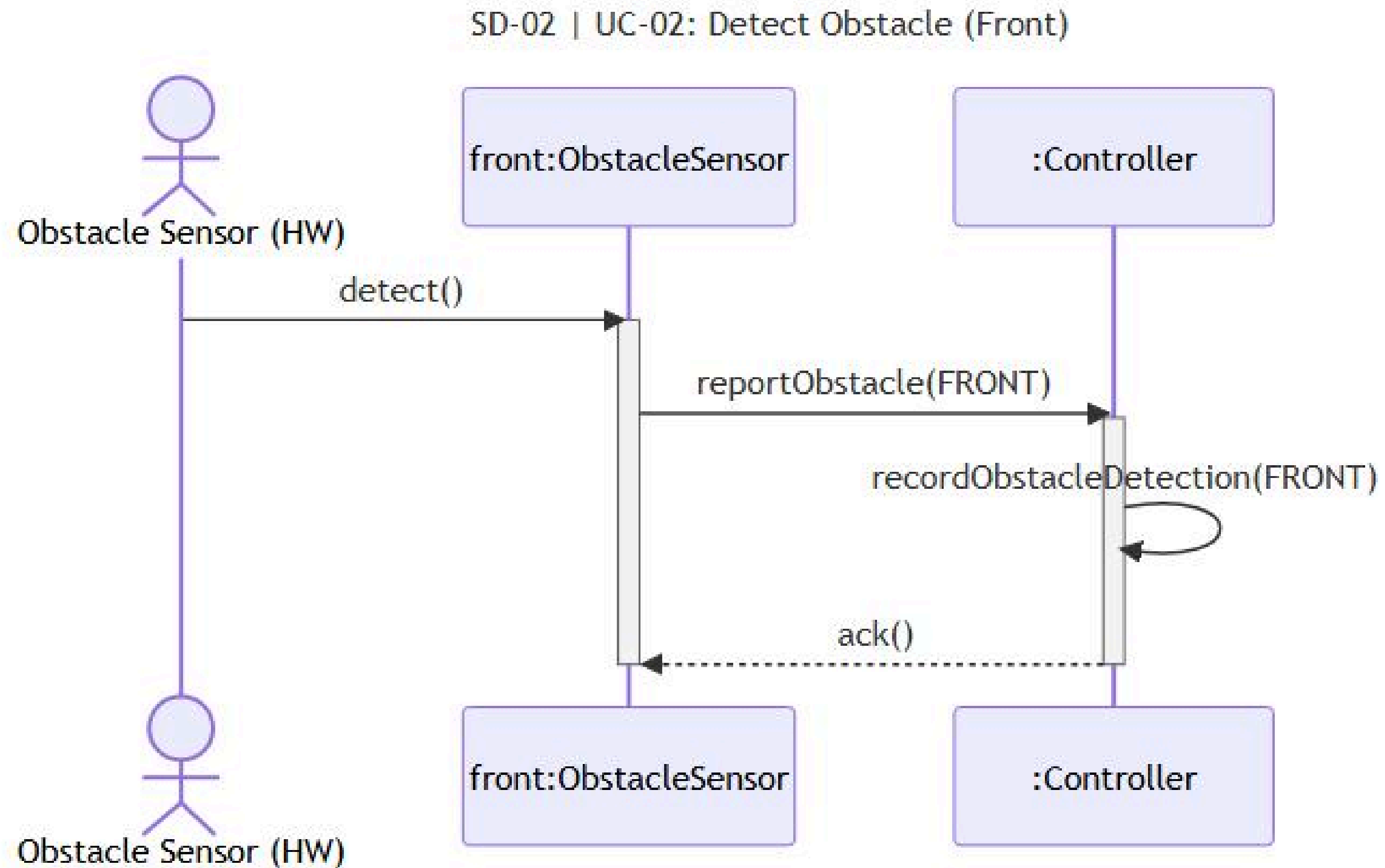
SD별 핵심 흐름

SD	핵심 객체 협력
SD-01	CTR → CU: <code>startCleaning()</code> → MU: <code>moveForward()</code> → loop: sensor <code>readData()</code>
SD-02/04/06	Sensor → CTR: <code>reportX()</code> → CTR: <code>recordX()</code>
SD-03	CTR: <code>stop()</code> → LS/RS: <code>isObstacleDetected()</code> → alt: <code>turn()</code> + <code>moveForward()</code>
SD-05	CTR → MU: <code>stop()</code> → <code>moveBackward()</code> → <code>turn()</code> → <code>moveForward()</code>
SD-07	CTR → CU: <code>boostPower()</code> → <code>restoreNormalPower()</code>
SD-08	SEN → CTR: <code>reportSensorError()</code> → alt: <code>retainLastValidState()</code> / MU: <code>stop()</code>

Sequence Diagram

SD-01 : UC-01 Perform Normal Cleaning

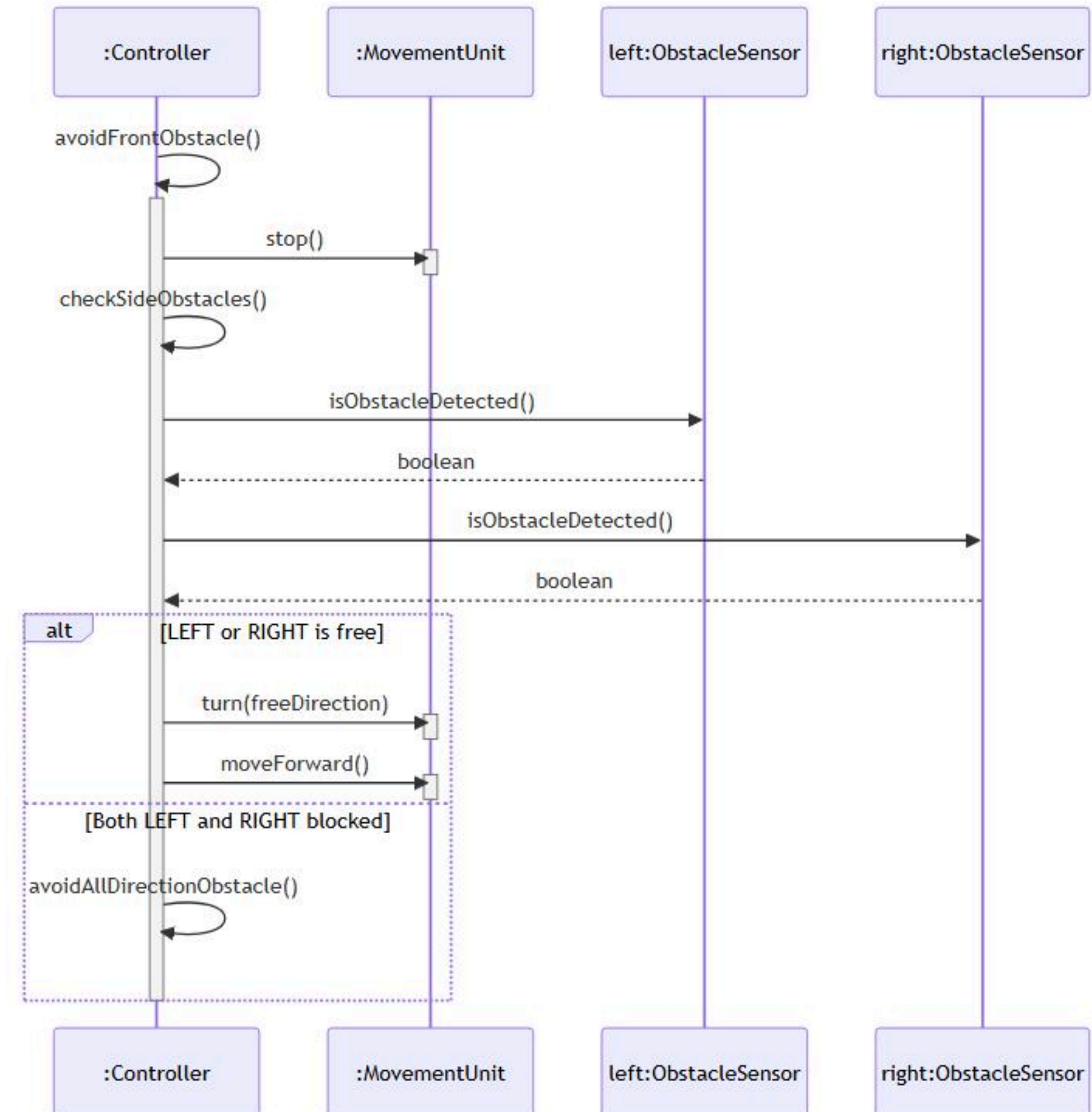
Sequence Diagram

SD-02 : UC-02 Detect Obstacle (Front)

Sequence Diagram

SD-03 : UC-03 Avoid Front Obstacle

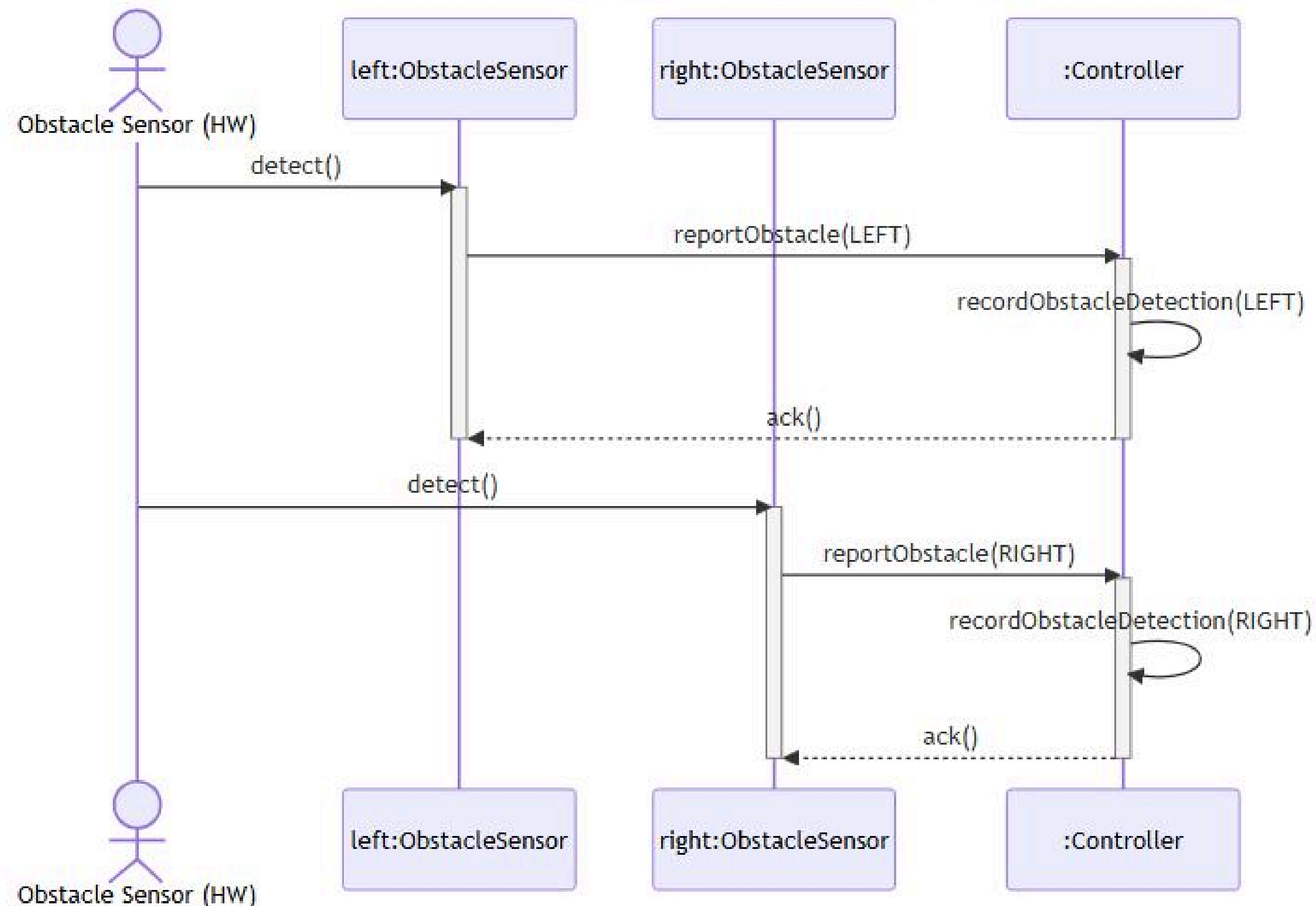
SD-03 | UC-03: Avoid Front Obstacle



Sequence Diagram

SD-04 : UC-04 Detect Obstacle (Left & Right)

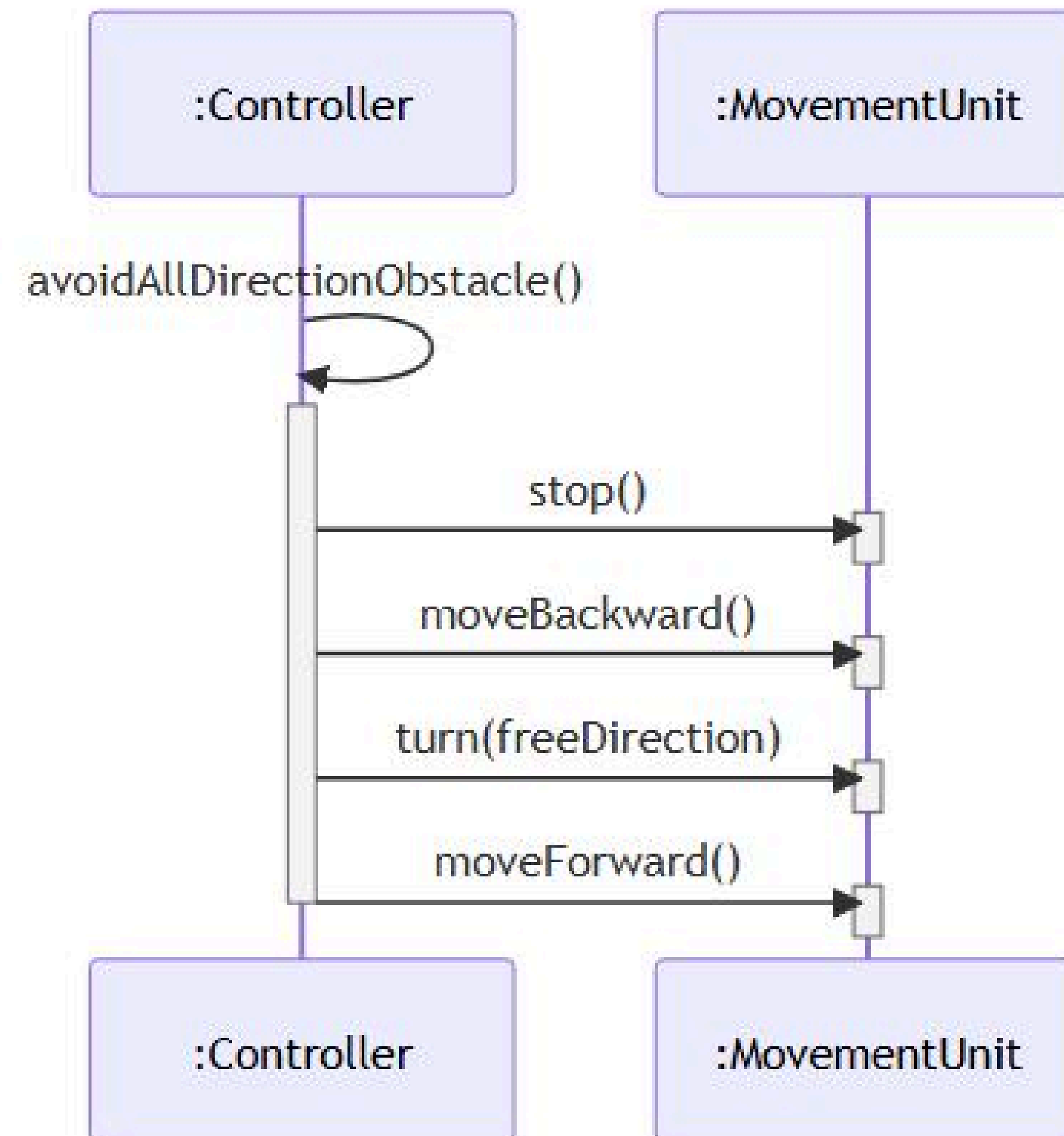
SD-04 | UC-04: Detect Obstacle (Left & Right)



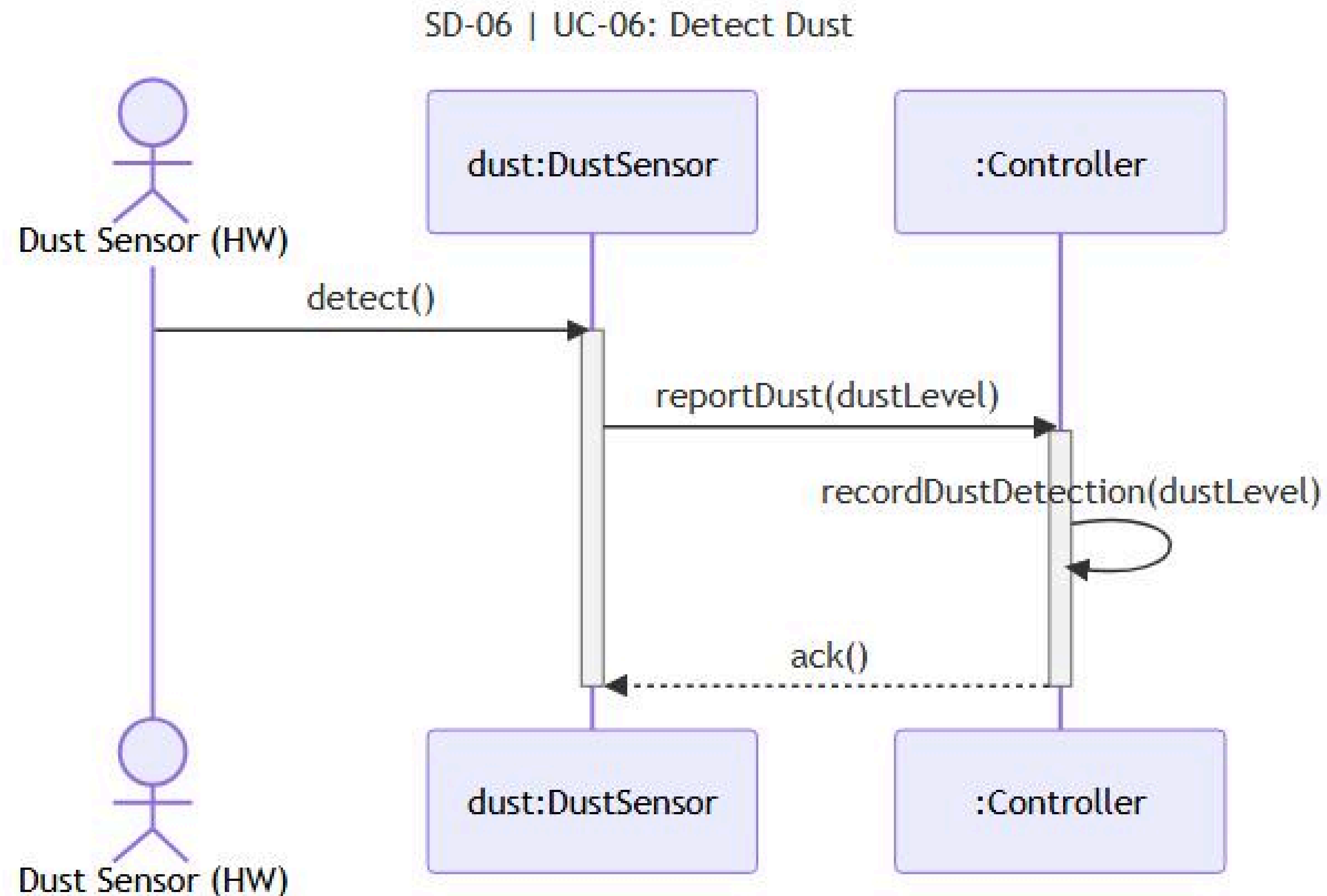
Sequence Diagram

SD-05 : UC-05 Avoid All-Direction Obstacle

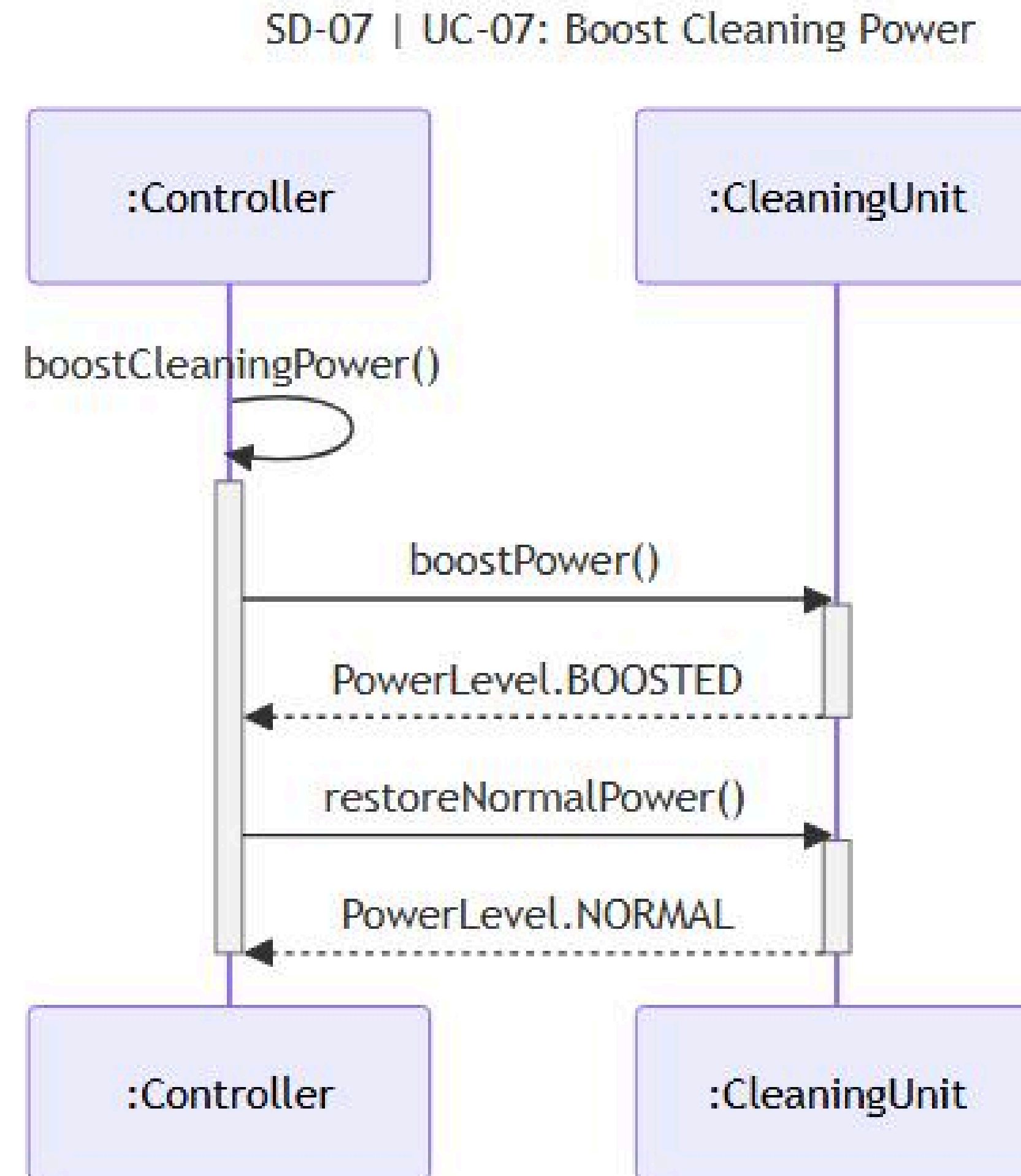
SD-05 | UC-05: Avoid All-Direction Obstacle



Sequence Diagram

SD-06 : UC-06 Detect Dust

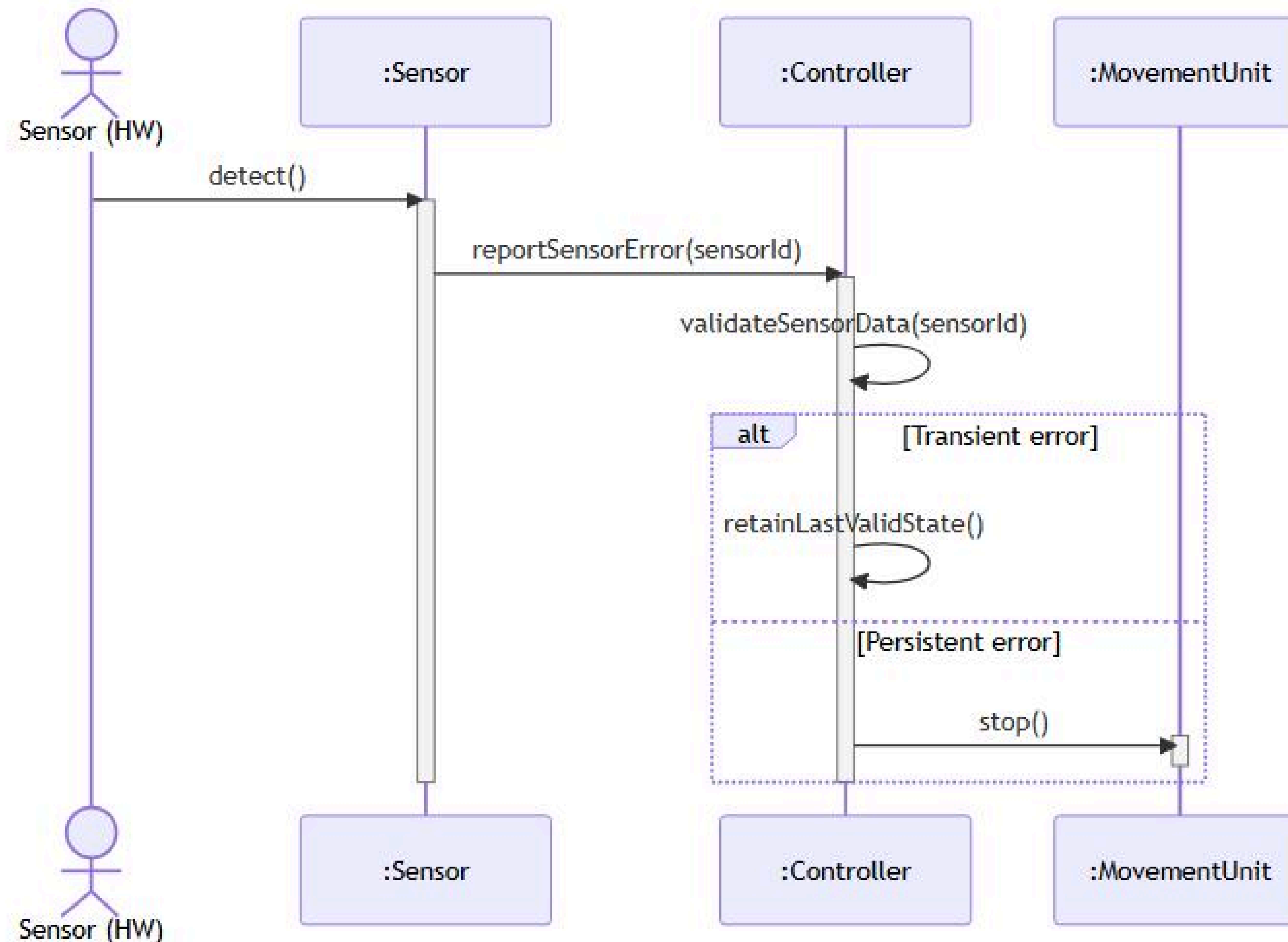
Sequence Diagram

SD-07 : UC-07 Boost Cleaning Power

Sequence Diagram

SD-08 : UC-08 Handle Sensor Error

SD-08 | UC-08: Handle Sensor Error



OOA & OOD

Class Diagram

Class Diagram Prompt

</> c:\00_Folders\01_UNIV\26-1\...

지금까지의 내용을 기반으로 Class Diagram을 그려줘

Design Class Diagram 구성

7개 클래스 + 5개 Enum

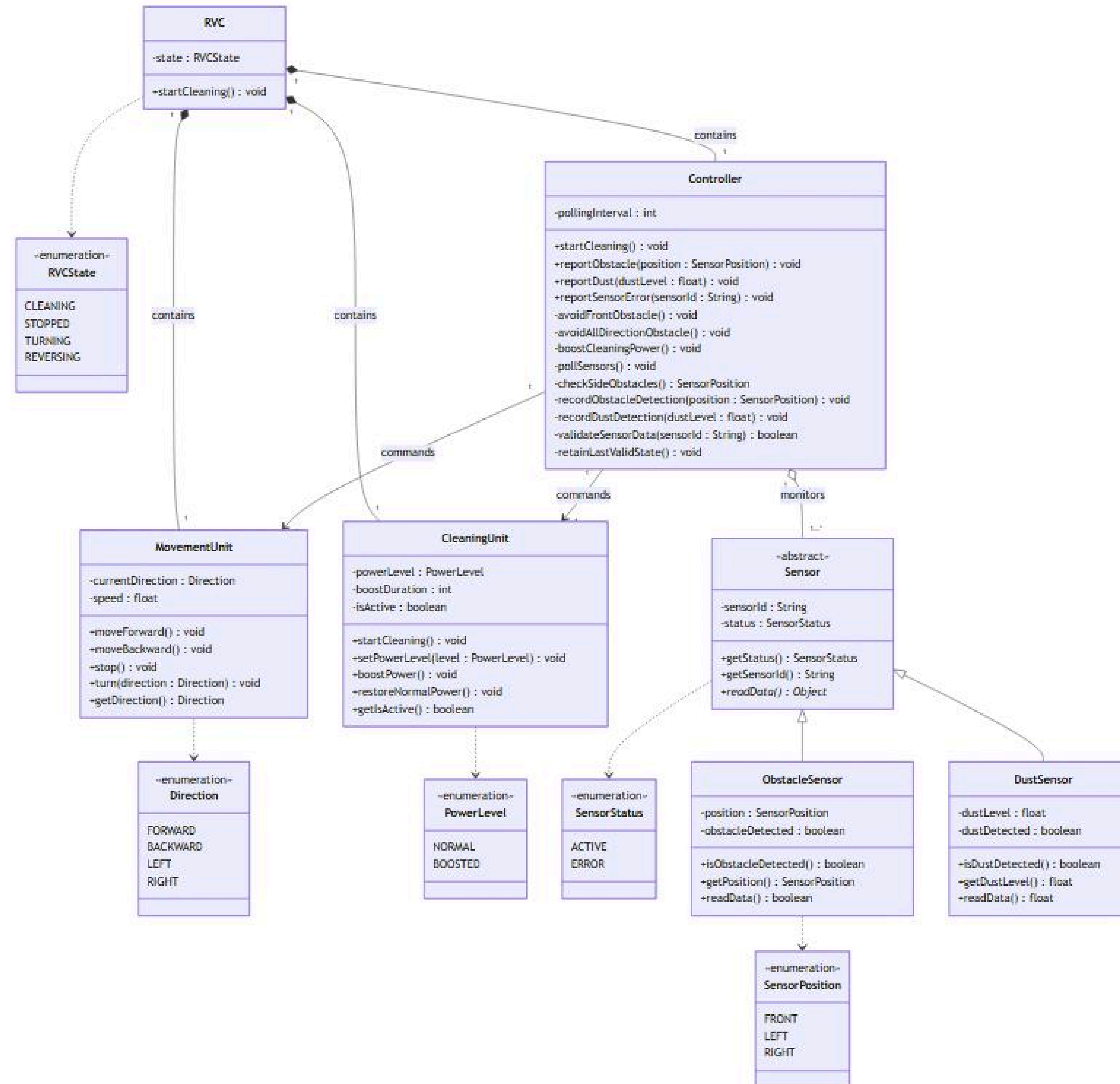
클래스	주요 추가 메서드 (Domain Model 대비)
RVC	startCleaning()
Controller	공개 4개: reportObstacle, reportDust, reportSensorError, startCleaning / 비공개 9개
MovementUnit	moveForward/Backward, stop, turn, getDirection
CleaningUnit	startCleaning, setPowerLevel, boostPower, restoreNormalPower
Sensor (abstract)	getStatus, getSensorId, readData (abstract)
ObstacleSensor	isObstacleDetected, getPosition, readData (구현)
DustSensor	isDustDetected, getDustLevel, readData (구현)

Domain Model 대비 주요 변경

- Obstacle / Dust 클래스 제거 → 센서 내부 상태값으로 통합
- SSD System Operations → Controller 공개 인터페이스로 반영
- Sensor.readData() 추상 메서드로 정의 (다형성 적용)
- 모든 멤버에 가시성(+ / -) 명시

OOA & OOD Class Model

Design Class Diagram - RVC Controller SW



OOI Coding

Coding Prompt

자 이제 이전의 문서들을 바탕으로 RVC와 Simulator를 구현. Simulator는 나중에.

구현 요약

`Controller.cpp` 핵심 로직 (시퀀스 다이어그램 직접 반영)

메서드	참조 SD	구현 내용
<code>startCleaning()</code>	SD-01	CU 활성화 → MU 전진
<code>pollSensors()</code>	SD-01 loop	전방센서 → 장애물 시 회피; 먼지센서 → 임계초과 시 부스트; 정상 시 전진
<code>reportObstacle(FRONT)</code>	SD-02	기록 + 즉시 <code>avoidFrontObstacle()</code>
<code>avoidFrontObstacle()</code>	SD-03	stop → 측면 확인 → 좌/우 회전 or 전방향 회피
<code>avoidAllDirectionObstacle()</code>	SD-05	stop → backward → turn(BACK) → forward
<code>boostCleaningPower()</code>	SD-07	<code>boostPower()</code> → <code>restoreNormalPower()</code> 순서 보장
<code>validateSensorData()</code>	SD-08	per-sensor 에러 카운터, <code>kErrorThreshold(3)</code> 이상 시 stop

발견·수정된 설계 이슈: `avoidFrontObstacle()` → `avoidAllDirectionObstacle()` 경로에서 `stop()` 이 두 번 호출되는 것은 SD-03과 SD-05가 모두 `stop()`으로 시작하기 때문에 의도된 동작. 테스트를 `Times(AtLeast(1))`로 수정하여 반영.

다음 단계: Simulator 구현 (`SimObstacleSensor`, `SimDustSensor`, `SimMovementUnit`, `SimCleaningUnit`).

OOI

Unit Test

Unit Test Prompt

그대로 진행. UseCase-Driven 방식으로 CI/CD, Static Analysis, Test Case 작성부터 해줘.

환경 구축 결과

파일 구조 (22개)

```

rvc_project/
├── .github/workflows/
│   ├── ci.yml          ← Build + Test + Coverage (GCC, Ubuntu)
│   └── analysis.yml    ← clang-tidy + cppcheck
├── cmake/ClangTidy.cmake
├── src/rvc/
│   ├── Types.hpp      ← 5개 enum 정의
│   ├── Sensor.hpp     ← abstract base (pure virtual)
│   ├── ObstacleSensor.hpp ← abstract (isObstacleDetected)
│   ├── DustSensor.hpp ← abstract (getDustLevel)
│   ├── MovementUnit.hpp ← abstract (move/stop/turn)
│   ├── CleaningUnit.hpp ← abstract (boost/restore)
│   ├── Controller.hpp ← 공개 시스템 오퍼레이션 + pollSensors()
│   ├── Controller.cpp ← 스텝 (TDD Red phase)
│   └── RVC.hpp / RVC.cpp
├── tests/
│   ├── unit/mocks/MockSensors.hpp ← gmock 4종
│   ├── unit/ControllerTest.cpp    ← UC-Driven 단위 테스트 22개
│   ├── unit/SensorTest.cpp        ← 인터페이스 계약 테스트 (통과)
│   └── integration/RVCSystemTest.cpp ← 시나리오 6개
├── .clang-tidy / .clang-format

```

UseCase → 테스트 추적성

UC	테스트 ID	핵심 검증
UC-01	TC-UC01-001~004	startCleaning 순서, pollSensors 루프
UC-02	TC-UC02-001~003	전방 장애물 보고 → stop 우선
UC-03	TC-UC03-001~004	좌/우 free 분기, 양측 막힘 시 에스컬레이션
UC-04	TC-UC04-001~003	좌우 장애물 기록 독립성
UC-05	TC-UC05-001~002	stop→backward→turn→forward 순서 강제
UC-06	TC-UC06-001~003	먼지 보고/임계값 미만 부스트 없음
UC-07	TC-UC07-001~004	임계값 초과 boost, 복원, 장애물 우선순위
UC-08	TC-UC08-001~004	일시/지속 에러 카운터, per-sensor 독립

CI/CD 파이프라인 흐름

Push / PR

```

├── ci.yml
│   ├── [build-and-test] GCC Debug + coverage → ctest → lcov artifact
│   └── [build-release] GCC Release 빌드 검증
├── analysis.yml
│   ├── [clang-tidy] .clang-tidy 규칙 적용 (src/rvc/*.cpp)
│   └── [cppcheck] 메모리/UB 정적 분석

```

SensorTest

ObstacleSensorInterfaceTest

```
TEST(ObstacleSensorInterfaceTest, MockSatisfiesInterface_NoObstacle) {
    MockObstacleSensor sensor;
    EXPECT_CALL(sensor, getId()).WillOnce(Return(1));
    EXPECT_CALL(sensor, getStatus()).WillOnce(Return(SensorStatus::OK));
    EXPECT_CALL(sensor, getPosition()).WillOnce(Return(SensorPosition::FRONT));
    EXPECT_CALL(sensor, isObstacleDetected()).WillOnce(Return(false));

    EXPECT_EQ(1, sensor.getId());
    EXPECT_EQ(SensorStatus::OK, sensor.getStatus());
    EXPECT_EQ(SensorPosition::FRONT, sensor.getPosition());
    EXPECT_FALSE(sensor.isObstacleDetected());
}

TEST(ObstacleSensorInterfaceTest, MockSatisfiesInterface_ObstaclePresent) {
    MockObstacleSensor sensor;
    EXPECT_CALL(sensor, isObstacleDetected()).WillOnce(Return(true));
    EXPECT_CALL(sensor, getPosition()).WillOnce(Return(SensorPosition::LEFT));

    EXPECT_TRUE(sensor.isObstacleDetected());
    EXPECT_EQ(SensorPosition::LEFT, sensor.getPosition());
}

TEST(ObstacleSensorInterfaceTest, ReadData_CanBeCalled) {
    MockObstacleSensor sensor;
    EXPECT_CALL(sensor, readData()).Times(1);
    sensor.readData();
}

TEST(ObstacleSensorInterfaceTest, ErrorStatus_ReportedCorrectly) {
    MockObstacleSensor sensor;
    EXPECT_CALL(sensor, getStatus()).WillOnce(Return(SensorStatus::ERROR_TRANSIENT));
    EXPECT_EQ(SensorStatus::ERROR_TRANSIENT, sensor.getStatus());
}
```

SensorTest

DustSensorInterfaceTest

```
TEST(DustSensorInterfaceTest, MockSatisfiesInterface_NoDust) {
    MockDustSensor sensor;
    EXPECT_CALL(sensor, getId()).WillOnce(Return(2));
    EXPECT_CALL(sensor, getStatus()).WillOnce(Return(SensorStatus::OK));
    EXPECT_CALL(sensor, isDustDetected()).WillOnce(Return(false));
    EXPECT_CALL(sensor, getDustLevel()).WillOnce(Return(0.0));

    EXPECT_EQ(2, sensor.getId());
    EXPECT_EQ(SensorStatus::OK, sensor.getStatus());
    EXPECT_FALSE(sensor.isDustDetected());
    EXPECT_DOUBLE_EQ(0.0, sensor.getDustLevel());
}

TEST(DustSensorInterfaceTest, MockSatisfiesInterface_DustDetected) {
    MockDustSensor sensor;
    EXPECT_CALL(sensor, isDustDetected()).WillOnce(Return(true));
    EXPECT_CALL(sensor, getDustLevel()).WillOnce(Return(0.85));

    EXPECT_TRUE(sensor.isDustDetected());
    EXPECT_DOUBLE_EQ(0.85, sensor.getDustLevel());
}

TEST(DustSensorInterfaceTest, ReadData_CanBeCalled) {
    MockDustSensor sensor;
    EXPECT_CALL(sensor, readData()).Times(1);
    sensor.readData();
}
```

SensorTest

MovementUnitInterfaceTest

```
TEST(MovementUnitInterfaceTest, AllMotionCommands_CanBeCalled) {
    MockMovementUnit mu;
    EXPECT_CALL(mu, moveForward()).Times(1);
    EXPECT_CALL(mu, moveBackward()).Times(1);
    EXPECT_CALL(mu, stop()).Times(1);
    EXPECT_CALL(mu, turn(Direction::LEFT)).Times(1);

    mu.moveForward();
    mu.moveBackward();
    mu.stop();
    mu.turn(Direction::LEFT);
}

TEST(MovementUnitInterfaceTest, GetDirection_ReturnsCurrentDirection) {
    MockMovementUnit mu;
    EXPECT_CALL(mu, getDirection()).WillOnce(Return(Direction::FRONT));
    EXPECT_EQ(Direction::FRONT, mu.getDirection());
}
```

SensorTest

CleaningUnitInterfaceTest

```
TEST(CleaningUnitInterfaceTest, BoostAndRestore_CanBeCalled) {
    MockCleaningUnit cu;
    EXPECT_CALL(cu, boostPower()).Times(1);
    EXPECT_CALL(cu, restoreNormalPower()).Times(1);
    cu.boostPower();
    cu.restoreNormalPower();
}

TEST(CleaningUnitInterfaceTest, StartCleaning_CanBeCalled) {
    MockCleaningUnit cu;
    EXPECT_CALL(cu, startCleaning()).Times(1);
    cu.startCleaning();
}
```

ControllerTest

UC01

```
// TC-UC01-001
// startCleaning() must activate the CleaningUnit before any movement.
// Ref: SD-01 – CTR->>CU: startCleaning()
TEST_F(ControllerTest, UC01_TC001_StartCleaning_ActivatesCleaningUnit) {
    EXPECT_CALL(cu_, startCleaning()).Times(1);
    controller_>startCleaning();
}

// TC-UC01-002
// startCleaning() must command forward movement immediately after activation.
// Ref: SD-01 – CTR->>MU: moveForward()
TEST_F(ControllerTest, UC01_TC002_StartCleaning_CommandsForwardMovement) {
    EXPECT_CALL(mu_, moveForward()).Times(AtLeast(1));
    controller_>startCleaning();
}

// TC-UC01-003
// pollSensors() with clear sensors must continue forward – no obstacle avoidance.
// Ref: SD-01 loop – readData() returns false/0.0 → moveForward()
TEST_F(ControllerTest, UC01_TC003_PollSensors_NoObstacle_ContinuesForward) {
    EXPECT_CALL(mu_, moveForward()).Times(AtLeast(1));
    EXPECT_CALL(mu_, stop()).Times(0);
    controller_>pollSensors();
}

// TC-UC01-004
// pollSensors() must call readData() on every sensor each cycle.
// Ref: SD-01 – sensors polled in every loop iteration
TEST_F(ControllerTest, UC01_TC004_PollSensors_ReadsAllSensorsEachCycle) {
    EXPECT_CALL(frontSensor_, readData()).Times(1);
    EXPECT_CALL(dustSensor_, readData()).Times(1);
    controller_>pollSensors();
}
```

ControllerTest UC02

```
// TC-UC02-001
// reportObstacle(FRONT) must trigger stop – front obstacle detected by SW sensor.
// Ref: SD-02 – FS->>CTR: reportObstacle(FRONT) → CTR triggers avoidance
TEST_F(ControllerTest, UC02_TC001_ReportFrontObstacle_TriggersStop) {
    EXPECT_CALL(mu_, stop()).Times(AtLeast(1));
    controller_>reportObstacle(SensorPosition::FRONT);
}

// TC-UC02-002
// reportObstacle() must complete without throwing for any valid position.
TEST_F(ControllerTest, UC02_TC002_ReportObstacle_DoesNotThrow) {
    EXPECT_NO_THROW(controller_>reportObstacle(SensorPosition::FRONT));
    EXPECT_NO_THROW(controller_>reportObstacle(SensorPosition::LEFT));
    EXPECT_NO_THROW(controller_>reportObstacle(SensorPosition::RIGHT));
}

// TC-UC02-003
// pollSensors() with front obstacle + calls stop before any turn or forward.
// Ref: SD-03 – stop() is the first response to a front obstacle
TEST_F(ControllerTest, UC02_TC003_PollSensors_FrontObstacle_StopsFirst) {
    ON_CALL(frontSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(false));

    InSequence seq;
    EXPECT_CALL(mu_, stop());
    EXPECT_CALL(mu_, turn()); // direction determined by side sensors
    controller_>pollSensors();
}
```

ControllerTest

UC03

```
// TC-UC03-001
// Front obstacle + LEFT free + turn LEFT then resume forward.
// Ref: SD-03 alt [LEFT or RIGHT is free] - turn(FreeDirection) + moveForward()
TEST_F(ControllerTest, UC03_TC001_FrontObstacle_LeftFree_TurnsLeftResumesForward) {
    ON_CALL(frontSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(false));
    ON_CALL(rightSensor_, isObstacleDetected()).WillByDefault(Return(true));

    InSequence seq;
    EXPECT_CALL(mu_, stop());
    EXPECT_CALL(mu_, turn(Direction::LEFT));
    EXPECT_CALL(mu_, moveForward());
    controller_->pollSensors();
}

// TC-UC03-002
// Front obstacle + RIGHT free + turn RIGHT then resume forward.
// Ref: SD-03 alt [LEFT or RIGHT is free] - right-side free path
TEST_F(ControllerTest, UC03_TC002_FrontObstacle_RightFree_TurnsRightResumesForward) {
    ON_CALL(frontSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(rightSensor_, isObstacleDetected()).WillByDefault(Return(false));

    InSequence seq;
    EXPECT_CALL(mu_, stop());
    EXPECT_CALL(mu_, turn(Direction::RIGHT));
    EXPECT_CALL(mu_, moveForward());
    controller_->pollSensors();
}

// TC-UC03-003
// Front obstacle + BOTH sides blocked + escalates to all-direction avoidance.
// Ref: SD-03 else [Both LEFT and RIGHT blocked] -> avoidAllDirectionObstacle()
TEST_F(ControllerTest, UC03_TC003_FrontObstacle_BothSidesBlocked_EscalatesAllDirection) {
    ON_CALL(frontSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(rightSensor_, isObstacleDetected()).WillByDefault(Return(true));

    // All-direction avoidance requires backward movement
    EXPECT_CALL(mu_, moveBackward()).Times(AtLeast(1));
    controller_->pollSensors();
}

// TC-UC03-004
// checkSideObstacles() queries both left and right sensors.
// Ref: SD-03 - CTR->>LS/RS: isObstacleDetected()
TEST_F(ControllerTest, UC03_TC004_FrontObstacle_QueriesBothSideSensors) {
    ON_CALL(frontSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(false));

    EXPECT_CALL(leftSensor_, isObstacleDetected()).Times(AtLeast(1));
    EXPECT_CALL(rightSensor_, isObstacleDetected()).Times(AtLeast(1));
    controller_->pollSensors();
}
```

ControllerTest UC04

```
// TC-UC04-001
// reportObstacle(LEFT) is acknowledged without error.
// Ref: SD-04 - LS->>CTR: reportObstacle(LEFT) + CTR: recordObstacleDetection(LEFT)
TEST_F(ControllerTest, UC04_TC001_ReportLeftObstacle_RecordedWithoutError) {
    EXPECT_NO_THROW(controller_>reportObstacle(SensorPosition::LEFT));
}

// TC-UC04-002
// reportObstacle(RIGHT) is acknowledged without error.
// Ref: SD-04 - RS->>CTR: reportObstacle(RIGHT)
TEST_F(ControllerTest, UC04_TC002_ReportRightObstacle_RecordedWithoutError) {
    EXPECT_NO_THROW(controller_>reportObstacle(SensorPosition::RIGHT));
}

// TC-UC04-003
// Left sensor reports obstacle + subsequent poll treats LEFT side as blocked.
// Ref: SD-04 + SD-03 - recorded LEFT obstacle influences avoidFrontObstacle()
TEST_F(ControllerTest, UC04_TC003_LeftObstacleRecorded_AffectsSideCheck) {
    // After LEFT is blocked and FRONT is clear, moving forward is still valid
    ON_CALL(frontSensor_, isObstacleDetected()).WillByDefault(Return(false));
    EXPECT_CALL(mu_, moveForward()).Times(AtLeast(1));
    controller_>reportObstacle(SensorPosition::LEFT);
    controller_>pollSensors();
}
```

ControllerTest

UC05

```
// TC-UC05-001
// All directions blocked + sequence: stop + backward + turn + forward.
// Ref: SD-05 - full avoidAllDirectionObstacle() call sequence
TEST_F(ControllerTest, UC05_TC001_AllDirectionsBlocked_FullAvoidanceSequence) {
    ON_CALL(frontSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(rightSensor_, isObstacleDetected()).WillByDefault(Return(true));

    InSequence seq;
    EXPECT_CALL(mu_, stop()).Times(AtLeast(1)); // stop() in avoidfront + avoidAllDir
    EXPECT_CALL(mu_, moveBackward());
    EXPECT_CALL(mu_, turn(_));
    EXPECT_CALL(mu_, moveForward());
    controller_>pollSensors();
}

// TC-UC05-002
// All-direction avoidance must call stop() before moveBackward().
// Ref: SD-05 - stop() precedes moveBackward() to prevent unsafe reverse
TEST_F(ControllerTest, UC05_TC002_AllDirectionsBlocked_StopBeforeBackward) {
    ON_CALL(frontSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(rightSensor_, isObstacleDetected()).WillByDefault(Return(true));

    bool stopCalled = false;
    bool backwardCalled = false;

    ON_CALL(mu_, stop()).WillByDefault([&stopCalled]{ stopCalled = true; });
    ON_CALL(mu_, moveBackward()).WillByDefault([&]{ backwardCalled = true;
        EXPECT_TRUE(stopCalled) << "stop() must precede moveBackward()"; });

    EXPECT_CALL(mu_, stop()).Times(AtLeast(1));
    EXPECT_CALL(mu_, moveBackward()).Times(AtLeast(1));
    controller_>pollSensors();
}
```

ControllerTest

UC06

```
// TC-UC06-001
// reportDust() records the level without throwing.
// Ref: SD-06 - DS->>CTR: reportDust(dustLevel) + recordDustDetection()
TEST_F(ControllerTest, UC06_TC001_ReportDust_RecordsWithoutError) {
    EXPECT_NO_THROW(controller_>reportDust(0.8));
    EXPECT_NO_THROW(controller_>reportDust(0.0));
}

// TC-UC06-002
// Dust level at zero + no boost triggered.
// Ref: SD-06 - zero dust does not trigger UC-07
TEST_F(ControllerTest, UC06_TC002_ZeroDustLevel_NoPowerBoost) {
    EXPECT_CALL(cu_, boostPower()).Times(0);
    controller_>reportDust(0.0);
}

// TC-UC06-003
// pollSensors() reads DustSensor every cycle.
// Ref: SD-01 - CTR->>DS: readData() in every loop iteration
TEST_F(ControllerTest, UC06_TC003_PollSensors_ReadsDustSensorEachCycle) {
    EXPECT_CALL(dustSensor_, readData()).Times(AtLeast(1));
    controller_>pollSensors();
}
```

ControllerTest

UC07

```
// TC-UC07-001
// Dust level above threshold → boostPower() called on CleaningUnit.
// Ref: SD-07 – CTR->>CU: boostPower()
TEST_F(ControllerTest, UC07_TC001_HighDustLevel_TriggersPowerBoost) {
    ON_CALL(dustSensor_, getDustLevel()).WillByDefault(Return(Controller::kDustThreshold + 0.1));

    EXPECT_CALL(cu_, boostPower()).Times(AtLeast(1));
    controller_>pollSensors();
}

// TC-UC07-002
// Dust level below threshold → boostPower() must NOT be called.
// Ref: SD-07 only triggered when dust > threshold
TEST_F(ControllerTest, UC07_TC002_LowDustLevel_NoPowerBoost) {
    ON_CALL(dustSensor_, getDustLevel()).WillByDefault(Return(Controller::kDustThreshold - 0.1));

    EXPECT_CALL(cu_, boostPower()).Times(0);
    controller_>pollSensors();
}

// TC-UC07-003
// After boostPower(), restoreNormalPower() must be called.
// Ref: SD-07 – CU: boostPower() → restoreNormalPower() in sequence
TEST_F(ControllerTest, UC07_TC003_AfterBoost_PowerIsRestored) {
    ON_CALL(dustSensor_, getDustLevel()).WillByDefault(Return(Controller::kDustThreshold + 0.1));

    InSequence seq;
    EXPECT_CALL(cu_, boostPower());
    EXPECT_CALL(cu_, restoreNormalPower());
    controller_>pollSensors();
}

// TC-UC07-004
// Boost must not interrupt obstacle avoidance.
// When front obstacle is present, avoidance takes priority over boost.
TEST_F(ControllerTest, UC07_TC004_FrontObstaclePresent_BoostDoesNotPreemptAvoidance) {
    ON_CALL(frontSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(false));
    ON_CALL(dustSensor_, getDustLevel()).WillByDefault(Return(0.9));

    // Avoidance (stop/turn) must happen; boost may happen after or be deferred
    EXPECT_CALL(mu_, stop()).Times(AtLeast(1));
    controller_>pollSensors();
}
```

ControllerTest

UC08

```
// TC-UC08-001
// Single sensor error + treated as transient; movement must NOT be stopped.
// Ref: SD-08 alt [Transient error] – retainLastValidState(), MU not stopped
TEST_F(ControllerTest, UC08_TC001_TransientSensorError_MovementNotStopped) {
    EXPECT_CALL(mu_, stop()).Times(0);
    controller_>reportSensorError(1); // first occurrence = transient
}

// TC-UC08-002
// Repeated sensor errors (≥ kErrorThreshold) + persistent; stop() called.
// Ref: SD-08 else [Persistent error] – CTR->MU: stop()
TEST_F(ControllerTest, UC08_TC002_PersistentSensorError_StopsMovement) {
    EXPECT_CALL(mu_, stop()).Times(AtLeast(1));
    for (int i = 0; i < Controller::kErrorThreshold; ++i) {
        controller_>reportSensorError(1);
    }
}

// TC-UC08-003
// Error counter is per-sensor: errors on different sensor IDs are independent.
// Ref: SD-08 – validateSensorData(sensorId) scoped to specific sensor
TEST_F(ControllerTest, UC08_TC003_ErrorCountIsPerSensor_IndependentCounters) {
    // Two errors on sensor 1, two errors on sensor 2 – neither reaches threshold of 3
    EXPECT_CALL(mu_, stop()).Times(0);
    controller_>reportSensorError(1);
    controller_>reportSensorError(2);
    controller_>reportSensorError(1);
    controller_>reportSensorError(2);
}

// TC-UC08-004
// reportSensorError() must not throw for any sensor ID.
TEST_F(ControllerTest, UC08_TC004_ReportSensorError_DoesNotThrow) {
    EXPECT_NO_THROW(controller_>reportSensorError(0));
    EXPECT_NO_THROW(controller_>reportSensorError(99));
}
```

OOI

System Test

SystemScenario 정상 청소

```
// =====  
// Scenario 1 - 정상 청소 세션 (UC-01)  
//  
// 시나리오: RVC가 시작 후 장애물·먼지 없는 공간에서 5틱 동안 청소  
// 검증 포인트:  
// - 청소 내내 FORWARD 상태 유지  
// - CleaningUnit 활성화 상태 유지  
// - 한 번도 STOPPED/TURNING 상태가 되지 않음  
// =====  
TEST(SystemScenario, Scenario1_NormalCleaningSession) {  
    RVCSimulator sim;  
    sim.start();  
  
    ASSERT_TRUE(sim.cleaning().getIsActive());  
    ASSERT_EQ(Motion::FORWARD, sim.movement().getMotion());  
  
    for (int tick = 1; tick <= 5; ++tick) {  
        sim.cleaning().resetBoostFlag();  
        sim.tick();  
  
        EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion())  
            << "Tick " << tick << ": should still be moving forward";  
        EXPECT_TRUE(sim.cleaning().getIsActive())  
            << "Tick " << tick << ": cleaning should remain active";  
        EXPECT_FALSE(sim.cleaning().wasBoosted())  
            << "Tick " << tick << ": no boost expected in clean area";  
        EXPECT_EQ(0, sim.movement().getStopCount())  
            << "Tick " << tick << ": should never have stopped";  
    }  
}
```

SystemScenario 장애물 연속 회피

```
// =====
// Scenario 2 - 장애물 연속 회피 세션 (UC-01, UC-02, UC-03)
//
// 시나리오: 청소 중 두 차례 전방 장애물을 만나 각각 다른 방향으로 회피
// Phase 1) 2회 정상 전진
// Phase 2) 전방+우측 막힘 + 좌회전 회피
// Phase 3) 2회 정상 전진
// Phase 4) 전방+좌측 막힘 + 우회전 회피
// Phase 5) 2회 정상 전진
// =====
TEST(SystemScenario, Scenario2_DoubleObstacleAvoidance) {
    RVCSimulator sim;
    sim.start();

    // --- Phase 1: 정상 전진 ---
    runClearTicks(sim, 2);
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());
    int stopsAfterClear1 = sim.movement().getStopCount();

    // --- Phase 2: 전방+우측 막힘 + 좌회전 ---
    sim.setFrontObstacle(true);
    sim.setRightObstacle(true);
    sim.cleaning().resetBoostFlag();
    sim.tick();

    EXPECT_EQ(Direction::LEFT, sim.movement().getLastTurn())
        << "First obstacle: should turn left (right is blocked)";
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion())
        << "Should resume forward after avoiding";
    EXPECT_GT(sim.movement().getStopCount(), stopsAfterClear1)
        << "Should have stopped at least once during avoidance";

    // --- Phase 3: 정상 전진 재개 ---
    runClearTicks(sim, 2);
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());
    int stopsAfterAvoid1 = sim.movement().getStopCount();
```

```
// --- Phase 4: 전방+좌측 막힘 + 우회전 ---
sim.setFrontObstacle(true);
sim.setLeftObstacle(true);
sim.setRightObstacle(false);
sim.cleaning().resetBoostFlag();
sim.tick();

EXPECT_EQ(Direction::RIGHT, sim.movement().getLastTurn())
    << "Second obstacle: should turn right (left is blocked)";
EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion())
    << "Should resume forward after second avoidance";
EXPECT_GT(sim.movement().getStopCount(), stopsAfterAvoid1)
    << "Additional stop during second avoidance";

// --- Phase 5: 정상 전진 재개 ---
runClearTicks(sim, 2);
EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());
```

}

SystemTest 사방(?) 막힘 탈출

```
// =====  
// Scenario 3 - 사방 막힘 탈출 세션 (UC-01, UC-02, UC-04, UC-05)  
//  
// 시나리오: 청소 중 모든 방향이 막혀 완전 역전 기동 수행  
// Phase 1) 3틱 정상 전진  
// Phase 2) 전방+좌+우 모두 막힘 → 후진+회전+전진  
// Phase 3) 3틱 정상 전진 재개  
// =====  
TEST(SystemScenario, Scenario3_CornerTrapAndEscape) {  
    RVCSimulator sim;  
    sim.start();  
  
    // --- Phase 1: 정상 전진 ---  
    runClearTicks(sim, 3);  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
    EXPECT_EQ(0, sim.movement().getStopCount());  
  
    // --- Phase 2: 사방 막힘 → 완전 역전 기동 ---  
    sim.setFrontObstacle(true);  
    sim.setLeftObstacle(true);  
    sim.setRightObstacle(true);  
    sim.cleaning().resetBoostFlag();  
    sim.tick();  
  
    // 역전 기동 완료 후 전진 재개 확인  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion())  
        << "After full reversal, should be moving forward again";  
    // 역전 기동 중 stop()이 2회 호출됨 (avoidFront + avoidAllDir)  
    EXPECT_GE(sim.movement().getStopCount(), 2)  
        << "Full reversal involves at least 2 stop() calls";  
    // 후방 회전 수행 여부 (avoidAllDirectionObstacle → turn(BACK))  
    EXPECT_EQ(Direction::BACK, sim.movement().getLastTurn())  
        << "Reversal maneuver should turn BACK";  
  
    // --- Phase 3: 장애물 해제 후 정상 전진 재개 ---  
    runClearTicks(sim, 3);  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
    EXPECT_TRUE(sim.cleaning().getIsActive());  
}
```

SystemTest

먼지 구역 통과

```
// =====  
// Scenario 4 - 먼지 구역 통과 세션 (UC-01, UC-06, UC-07)  
//  
// 시나리오: 깨끗한 구역 → 먼지 구역 진입 → 먼지 구역 통과 → 깨끗한 구역  
// Phase 1) 3틱 - 먼지 없음 (부스트 없어야 함)  
// Phase 2) 4틱 - 먼지 레벨 0.8 (매 틱 부스트 발동)  
// Phase 3) 3틱 - 먼지 해소 (부스트 다시 없어야 함)  
// =====  
TEST(SystemScenario, Scenario4_DustyZoneTraversal) {  
    RVCSimulator sim;  
    sim.start();  
  
    // --- Phase 1: 깨끗한 구역 ---  
    for (int t = 0; t < 3; ++t) {  
        sim.cleaning().resetBoostFlag();  
        sim.tick();  
        EXPECT_FALSE(sim.cleaning().wasBoosted())  
            << "Phase 1 tick " << (t+1) << ": no boost in clean area";  
    }  
  
    // --- Phase 2: 먼지 구역 진입 ---  
    sim.setDustLevel(Controller::kDustThreshold + 0.3);  
    int boostCount = 0;  
    for (int t = 0; t < 4; ++t) {  
        sim.cleaning().resetBoostFlag();  
        sim.tick();  
        if (sim.cleaning().wasBoosted()) ++boostCount;  
        EXPECT_EQ(PowerLevel::NORMAL, sim.cleaning().getPowerLevel())  
            << "Phase 2 tick " << (t+1) << ": power should be restored to NORMAL after boost";  
        EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion())  
            << "Phase 2 tick " << (t+1) << ": should keep moving during dust boost";  
    }  
    EXPECT_EQ(4, boostCount) << "Every tick in dusty zone should trigger boost";  
  
    // --- Phase 3: 먼지 해소 ---  
    sim.setDustLevel(0.0);  
    for (int t = 0; t < 3; ++t) {  
        sim.cleaning().resetBoostFlag();  
        sim.tick();  
        EXPECT_FALSE(sim.cleaning().wasBoosted())  
            << "Phase 3 tick " << (t+1) << ": no boost after dust cleared";  
    }  
}
```

SystemTest 장애물 회피 + 먼지 구역 복합

```
// =====  
// Scenario 5 - 장애물 회피 + 먼지 구역 복합 세션 (UC-01~03, UC-06~07)  
//  
// 시나리오: 전방 장애물 회피 직후 먼지 구역에 진입하는 연속 이벤트  
// Phase 1) 2틱 정상 전진  
// Phase 2) 전방 장애물 + 먼지 동시 → 장애물 우선 처리 (이 턱에 부스트 없음)  
// Phase 3) 장애물 해제 + 먼지 유지 → 부스트 발동  
// Phase 4) 먼지 해소 + 2틱 정상 전진  
// =====  
TEST(SystemScenario, Scenario5_ObstacleThenDustyZone) {  
    RVCSimulator sim;  
    sim.start();  
  
    // --- Phase 1: 정상 전진 ---  
    runClearTicks(sim, 2);  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
  
    // --- Phase 2: 전방 장애물 + 먼지 동시 ---  
    // pollSensors: front obstacle + avoidFrontObstacle + return (먼지 체크 건너뛰음)  
    sim.setFrontObstacle(true);  
    sim.setDustLevel(Controller::kDustThreshold + 0.2);  
    sim.cleaning().resetBoostFlag();  
    sim.tick();  
  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion())  
        << "After obstacle avoidance, should be moving forward";  
    EXPECT_FALSE(sim.cleaning().wasBoosted())  
        << "Obstacle avoidance takes priority; dust boost skipped this tick";  
  
    // --- Phase 3: 장애물 해제 + 먼지 유지 → 이번 턱에 부스트 발동 ---  
    sim.setFrontObstacle(false);  
    sim.cleaning().resetBoostFlag();  
    sim.tick();  
  
    EXPECT_TRUE(sim.cleaning().wasBoosted())  
        << "With obstacle cleared, dust boost should now fire";  
    EXPECT_EQ(PowerLevel::NORMAL, sim.cleaning().getPowerLevel())  
        << "Power should be restored to NORMAL after boost";  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
  
    // --- Phase 4: 먼지 해소 + 정상 전진 ---  
    runClearTicks(sim, 2);  
    EXPECT_FALSE(sim.cleaning().wasBoosted());  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
}
```

SystemTest

센서 오류 누적과 비상 정지

```
// =====  
// Scenario 6 - 센서 오류 누적과 비상 정지 세션 (UC-01, UC-08)  
//  
// 시나리오: 청소 중 센서 오류가 점진적으로 누적되어 비상 정지에 이르는 과정  
// Phase 1) 3틱 정상 청소  
// Phase 2) 오류 1회 - 일시적 오류, 청소 계속  
// Phase 3) 오류 2회 - 누적 2회, 아직 청소 계속  
// Phase 4) 오류 3회 - 임계값 도달, 비상 정지  
// =====  
TEST(SystemScenario, Scenario6_SensorDegradationAndEmergencyStop) {  
    RVCSimulator sim;  
    sim.start();  
  
    // --- Phase 1: 정상 청소 ---  
    runClearTicks(sim, 3);  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
    EXPECT_EQ(0, sim.movement().getStopCount());  
  
    // --- Phase 2: 오류 1회 (일시적) ---  
    sim.reportSensorError(1);  
    EXPECT_NE(Motion::STOPPED, sim.movement().getMotion())  
        << "1st error: transient, should not stop";  
  
    // --- Phase 3: 오류 2회 누적 (아직 일시적) ---  
    sim.reportSensorError(1);  
    EXPECT_NE(Motion::STOPPED, sim.movement().getMotion())  
        << "2nd error: still transient, should not stop";  
  
    // 일반 틱은 여전히 정상 동작  
    sim.cleaning().resetBoostFlag();  
    sim.tick();  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion())  
        << "Normal ticks between errors should continue forward";  
  
    // --- Phase 4: 오류 3회 - 임계값 도달 → 비상 정지 ---  
    int stopsBeforeEmergency = sim.movement().getStopCount();  
    sim.reportSensorError(1);  
    EXPECT_EQ(Motion::STOPPED, sim.movement().getMotion())  
        << "3rd error reaches threshold: emergency stop must occur";  
    EXPECT_EQ(stopsBeforeEmergency + 1, sim.movement().getStopCount())  
        << "Emergency stop calls stop() exactly once";  
}
```

SystemTest 전체 청소

```
// =====
// Scenario 7 - 전체 청소 세션 (모든 UC 연속 발생)
//
// 시나리오: 실제 방 청소를 모사한 장시간 복합 시나리오
// Phase 1) 청소 시작 + 3틱 정상 전진
// Phase 2) 전방 장애물 (좌측 여유) → 좌회전 + 전진
// Phase 3) 2틱 정상 전진
// Phase 4) 먼지 구역 3틱 (매 틱 부스트)
// Phase 5) 먼지 해소 + 2틱 전진
// Phase 6) 사방 막힘 → 완전 역전 기동
// Phase 7) 2틱 정상 전진
// Phase 8) 센서 오류 3회 누적 → 비상 정지
// =====
TEST(SystemScenario, Scenario7_FullRoomCleaningSession) {
    RVCSimulator sim;

    // --- Phase 1: 청소 시작 ---
    sim.start();
    ASSERT_TRUE(sim.cleaning().getIsActive());
    ASSERT_EQ(Motion::FORWARD, sim.movement().getMotion());

    runClearTicks(sim, 3);
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());
    EXPECT_EQ(0, sim.movement().getStopCount());

    // --- Phase 2: 전방 장애물, 좌측 여유 → 좌회전 ---
    sim.setFrontObstacle(true);
    sim.setRightObstacle(true);
    sim.cleaning().resetBoostFlag();
    sim.tick();

    EXPECT_EQ(Direction::LEFT, sim.movement().getLastTurn());
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());
    int stopsP2 = sim.movement().getStopCount();
    EXPECT_GT(stopsP2, 0);

    // --- Phase 3: 정상 전진 재개 ---
    runClearTicks(sim, 2);
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());
```

```
// --- Phase 4: 먼지 구역 진입 ---
sim.setDustLevel(Controller::kDustThreshold + 0.25);
int totalBoosts = 0;
for (int t = 0; t < 3; ++t) {
    sim.cleaning().resetBoostFlag();
    sim.tick();
    if (sim.cleaning().wasBoosted()) ++totalBoosts;
}
EXPECT_EQ(3, totalBoosts) << "All 3 dusty ticks should boost";
EXPECT_EQ(PowerLevel::NORMAL, sim.cleaning().getPowerLevel());

// --- Phase 5: 먼지 해소 ---
runClearTicks(sim, 2);
EXPECT_FALSE(sim.cleaning().wasBoosted());
EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());

// --- Phase 6: 사방 막힘 → 완전 역전 기동 ---
sim.setFrontObstacle(true);
sim.setLeftObstacle(true);
sim.setRightObstacle(true);
sim.cleaning().resetBoostFlag();
sim.tick();

EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion())
    << "After full reversal, must resume forward";
EXPECT_EQ(Direction::BACK, sim.movement().getLastTurn())
    << "Full reversal turns BACK";
int stopsP6 = sim.movement().getStopCount();
EXPECT_GT(stopsP6, stopsP2);

// --- Phase 7: 정상 전진 재개 ---
runClearTicks(sim, 2);
EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());

// --- Phase 8: 센서 오류 누적 → 비상 정지 ---
for (int i = 0; i < Controller::kErrorThreshold - 1; ++i) {
    sim.reportSensorError(2);
    EXPECT_NE(Motion::STOPPED, sim.movement().getMotion())
        << "Error " << (i+1) << ": still transient, should not stop";
}
sim.reportSensorError(2);
EXPECT_EQ(Motion::STOPPED, sim.movement().getMotion())
    << "Final error: emergency stop";

// 비상 정지 후 청소 유닛은 여전히 활성 상태 (FR-CLN-04)
EXPECT_TRUE(sim.cleaning().getIsActive())
    << "CleaningUnit stays active even after emergency stop";
}
```

OOI

Static Analysis

Static analysis Prompt

정적분석 툴들도 깃헙 액션에서 실행되도록 수정해줘

- Thought for 1265s >
- 현재 `analysis.yml` 의 문제를 파악합니다.
- **Read** `c:\00_Folders\01_UNIV\26-1\03_OOAD\OOAD_AI\.github\workflows\analysis.yml`
- **Read** `c:\00_Folders\01_UNIV\26-1\03_OOAD\OOAD_AI\rvc_project\.clang-tidy`
- Thought for 1082s >
- 문제 정리:
 - `--target googletest` : FetchContent는 configure 단계에서 다운로드됨 → build target이 없어 항상 실패 후 `|| true` 로 숨김
 - `WarningsAsErrors: ''` : 빈 값 → 위반해도 CI가 통과됨 (의미 없는 분석)
 - `EnumConstantCase: CamelCase` : 실제 코드는 `FRONT`, `IDLE` 등 `UPPER_CASE`
 - `ConstantCase: UPPER_CASE + prefix k` : 모순 (`kErrorThreshold` 는 camelCase)
 - `cppcheck --enable=all` : `all` 은 false positive 다수 발생
- **Write** `c:\00_Folders\01_UNIV\26-1\03_OOAD\OOAD_AI\.github\workflows\analysis.yml`

73 lines

Static Analysis

Clang-tidy

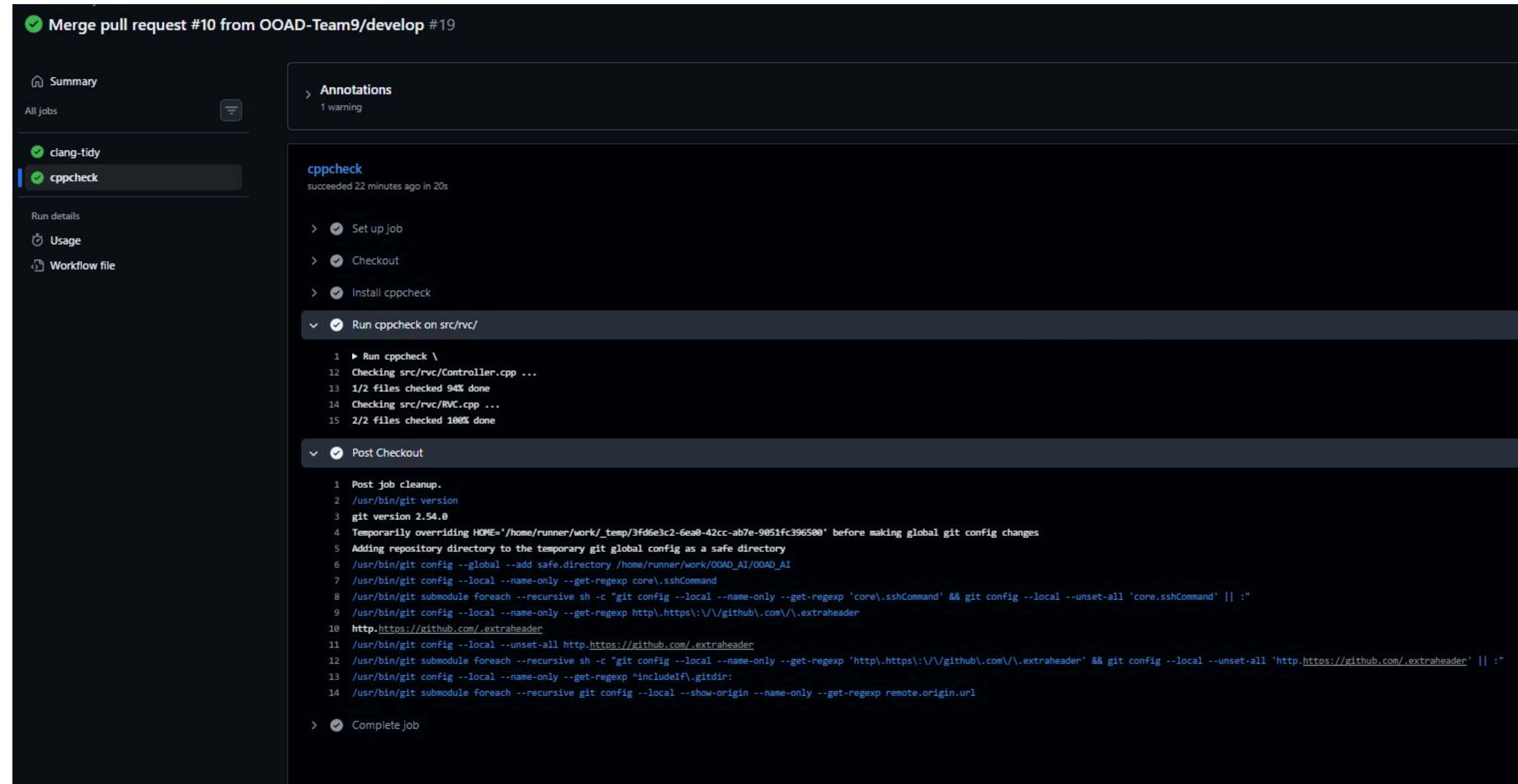
The screenshot shows a GitHub Actions workflow run for a merge pull request. The workflow is titled "Merge pull request #10 from OOAD-Team9/develop #19" and is part of a "Static Analysis" workflow. The run is successful, indicated by a green checkmark. The left sidebar shows the workflow steps: "clang-tidy" (selected) and "cppcheck". The main content area shows the "Annotations" section for the "clang-tidy" job, which succeeded 19 minutes ago in 30s. The job steps are: "Set up job", "Checkout", "Install cmake + clang + clang-tidy", "Configure — generate compile_commands.json", "Run clang-tidy on production sources", "Post Checkout", and "Complete job". The "Run clang-tidy on production sources" step is expanded, showing the following output:

```
1 ▶ Run find src/rvc -name '*.cpp' -print0 | \  
5 [1/2] Processing file /home/runner/work/OOAD_AI/OOAD_AI/rvc_project/src/rvc/RVC.cpp.  
6 11231 warnings generated.  
7 [2/2] Processing file /home/runner/work/OOAD_AI/OOAD_AI/rvc_project/src/rvc/Controller.cpp.  
8 22535 warnings generated.  
9 Suppressed 22535 warnings (22535 in non-user code).  
10 Use -header-filter=. to display errors from all non-system headers. Use -system-headers to display errors from system headers as well.
```

The "Post Checkout" step is also expanded, showing the following output:

```
1 Post job cleanup.  
2 /usr/bin/git version  
3 git version 2.54.0  
4 Temporarily overriding HOME='/home/runner/work/_temp/8a30d6fd-0a07-4598-a004-da04ed7116c' before making global git config changes  
5 Adding repository directory to the temporary git global config as a safe directory  
6 /usr/bin/git config --global --add safe.directory /home/runner/work/OOAD_AI/OOAD_AI  
7 /usr/bin/git config --local --name-only --get-regexp core.sshCommand  
8 /usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp 'core.sshCommand' && git config --local --unset-all 'core.sshCommand' || :"  
9 /usr/bin/git config --local --name-only --get-regexp http\,https:\\\\github\\.com\\\.extraheader  
10 http.https://github.com/.extraheader  
11 /usr/bin/git config --local --unset-all http.https://github.com/.extraheader  
12 /usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp 'http\,https:\\\\github\\.com\\\.extraheader' && git config --local --unset-all 'http.https://github.com/.extraheader' || :"  
13 /usr/bin/git config --local --name-only --get-regexp ^includeIf\,gitdir:  
14 /usr/bin/git submodule foreach --recursive git config --local --show-origin --name-only --get-regexp remote.origin.url
```

Static Analysis Cppcheck



The screenshot shows a GitHub Actions workflow for a pull request. The workflow is titled "Merge pull request #10 from OOAD-Team9/develop #19". It has a sidebar with "Summary", "All jobs", "clang-tidy", and "cppcheck". The "cppcheck" job is selected and expanded, showing a list of steps: "Set up job", "Checkout", "Install cppcheck", "Run cppcheck on src/rvc/", "Post Checkout", and "Complete job". The "Run cppcheck on src/rvc/" step is expanded, showing the following output:

```
1 Run cppcheck \  
12 Checking src/rvc/Controller.cpp ...  
13 1/2 files checked 94% done  
14 Checking src/rvc/RVC.cpp ...  
15 2/2 files checked 100% done
```

The "Post Checkout" step is also expanded, showing the following output:

```
1 Post job cleanup.  
2 /usr/bin/git version  
3 git version 2.54.0  
4 Temporarily overriding HOME='/home/runner/work/_temp/3fd6e3c2-6ea0-42cc-ab7e-9051fc396500' before making global git config changes  
5 Adding repository directory to the temporary git global config as a safe directory  
6 /usr/bin/git config --global --add safe.directory /home/runner/work/OOAD_AI/OOAD_AI  
7 /usr/bin/git config --local --name-only --get-regexp core.sshCommand  
8 /usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp 'core.sshCommand' && git config --local --unset-all 'core.sshCommand' || :"  
9 /usr/bin/git config --local --name-only --get-regexp http.https://github.com/.extraheader  
10 http.https://github.com/.extraheader  
11 /usr/bin/git config --local --unset-all http.https://github.com/.extraheader  
12 /usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp 'http.https://github.com/.extraheader' && git config --local --unset-all 'http.https://github.com/.extraheader' || :"  
13 /usr/bin/git config --local --name-only --get-regexp includeIf.gitdir  
14 /usr/bin/git submodule foreach --recursive git config --local --show-origin --name-only --get-regexp remote.origin.url
```

OOI Simulator

Simulator Prompt

System test를 위한 Simulator를 구현해줘

```
1/53 Test #1: ObstacleSensorInterfaceTest.MockSatisfiesInterface_NoObstacle .....
```

- 53/53 전부 통과합니다. 이제 커밋하면 됩니다:

```
git add src/simulator/ tests/system/ src/CMakeLists.txt tests/CMakeLists.txt
git commit -m "feat: add Simulator and system tests (53 tests total)"
git push origin develop
```

구현 요약

파일	역할
src/simulator/SimObstacleSensor.hpp	장애물 상태를 setObstacle(bool) 로 주입
src/simulator/SimDustSensor.hpp	먼지 레벨을 setDustLevel(double) 로 주입
src/simulator/SimMovementUnit.hpp	마지막 motion/turn/stopCount를 추적
src/simulator/SimCleaningUnit.hpp	active 여부, 파워 레벨, 부스트 이력 추적
src/simulator/RVCSimulator.hpp/.cpp	전체 시스템 래퍼 — start() / tick() / setX()
tests/system/RVCSimulatorTest.cpp	9개 시스템 테스트

Mock 기반 Integration Test와의 차이:

- Integration: 상호작용 횟수 검증 (EXPECT_CALL().Times(...)) → "어떻게 동작했는가"
- System (Simulator): 최종 상태 검증 (EXPECT_EQ(motion, FORWARD)) → "무슨 상태가 됐는가"
- Multi-tick 시나리오: 장애물 해제 후 복귀, 먼지 레벨 변동 등 실제 동작 흐름 검증

Simulator Simulator

```
C:\W00_Folders\W01_UNIVW26-1 x + v
=====
RVC Interactive Simulator
=====
Type 'help' for commands, 'quit' to exit.

> tick
One poll cycle done.
-----
MOVEMENT
  State      : FORWARD
  Last turn  : FRONT
  Stops     : 0
CLEANING
  Active     : NO
  Power      : NORMAL
  Boosted    : no
SENSORS
  Front: clear   Left: clear   Right: clear
  Dust : 0.00
-----
> front on
front obstacle -> BLOCKED
> dust 0.7
Dust level -> 0.70
> |
```

OOAD with AI

장단점 및 소감

OOAD with AI 결과물 검증

- Sequence Diagram을 생성했을 때 Actor가 명확히 명시되지 않았고, UseCase와 1대1 대응되지 않는 문제가 발생
→ Actor를 구체화하고 UseCase 문서를 참조하여 1대1 대응으로 다시 생성하도록 함.
- Static Analysis와 CI pipeline 구축 과정에서 이를 github actions에서 실행하지 않는 문제가 발생
→ yml 파일을 생성해 github action에서도 PR이 있을 때마다 Static analysis를 진행하도록 수정
- System test case를 생성했을 때 Unit test level에 가까운 테스트가 생성된 문제가 발생
→ 시나리오 기반의 system case로 수정하도록 함

OOAD with AI 장단점 및 소감

기존에 혼자 코딩할 때는 설계와 디버깅 단계에서 많은 시간과 에너지를 소모하며 흐름이 자주 끊어졌습니다.

반면, 이번에는 AI가 요구사항을 명확한 객체 모델링과 책임 분배로 즉시 시각화하고 복잡한 시물레이션 로직까지 빈틈없이 채워주었습니다.

덕분에 불필요한 시행착오가 획기적으로 줄어들었고, 전체적인 아키텍처를 조망하며 오롯이 핵심 도메인과 비즈니스 흐름에만 집중할 수 있었습니다.

단순히 텍스트를 기계적으로 입력하던 과거의 방식에서 벗어나, AI와 실시간으로 싱크를 맞추며 거대한 설계를 막힘없이 현실로 구현해내는 개발의 진정한 재미를 느꼈습니다.

감사합니다.