객체지향개발방법론

팀프로젝트#6

2026.05.29

7팀 강희진 김희용 양지윤 윤성진

Traditional Coding

Functional Requirements

Ref.#	Functional Requirements	Use-Case Number & Name	Category	Visibillity
R1.1	전원이 켜지면 로봇 청소기 시스템을 시작한다.	1. Power On System	Secondary	Evident
R2.1	청소 중에는 전진 이동한다. (이동 중 바닥 청소 및 물걸레 청소를 수행한다.)	2. Perform Automatic Cleaning	Primary	Evident
R3.1	전방, 좌측 센서를 통해 장애물을 감지한다.	3. Detect Obstacle	Primary	Hidden
R3.2	장애물 감지 시 전진 이동 및 청소를 중지한다.	3. Detect Obstacle	Primary	Evident
R3.3	전방이 막혀 있고 좌측이 비어 있으면 좌회전한다.	4. Avoid Obstacle – Turn Left	Primary	Evident
R3.4	전방과 좌측이 막혀 있고 우측이 비어 있으면 우회전한다.	5. Avoid Obstacle – Turn Right	Primary	Evident
R3.5	전방, 좌측, 우측이 모두 막혀 있으면 후진 후 방향 전환한다.	6. Avoid Obstacle – Move Backward & Turn	Primary	Evident
R4.1	먼지 센서를 통해 먼지를 감지한다.	2. Perform Automatic Cleaning	Secondary	Hidden
R4.2	먼지 감지 시 청소 강도를 높인다.	7. Perform Boost Cleaning	Primary	Evident
R4.3	강화된 청소 강도를 일정 시간 유지한다.	7. Perform Boost Cleaning	Secondary	Evident
R5.1	전원 종료 시 모든 동작을 중지한다.	8. Power Off System	Secondary	Evident
R6.1	예외 상황 혹은 오류 발생 시 전원을 종료한다.	9. Power Off System – Exceptional	Secondary	Hidden

Non-Functional Requirements
Performance Requirements (Quality)

NFR-P-01	전원 켜짐 시 청소 상태가 되어 첫 이동을 시작하기까지의 초기화 시간은 5초 이내여야 한다.
NFR-P-02	전방 센서로부터 장애물 감지 시 정지 상태로 전환 시간은 50ms 이내여야 한다.
NFR-P-03	좌회전, 우회전 결정 시 판단 및 모터 제어 신호 발생까지의 지연 시간은 500ms 이내여야 한다.
NFR-P-04	전/좌/우 방향이 모두 장애물로 막힌 경우 정지 후 100ms 이내에 후진을 시작하고, 후진 완료 후 회전 방향을 500ms 이내에 결정해야 하며, 결정된 후 500ms 이내에 모터 제어 신호가 발생해야 한다.
NFR-P-05	먼지 센서로부터 먼지 감지 시 터보 청소 모드 상태로 전환 시간은 100ms 이내여야 한다.
NFR-P-06	터보 청소 모드는 5초간 유지되고, 이후 터보 청소 모드를 해제한다.

Non-Functional Requirements
Other requirements (Quality)

NFR-O-01	시스템의 주요 구성 요소는 추상 클래스로 정의하여 구현 간 결합도를 낮추고 변경에 유연하도록 설계되어야 한다.
NFR-O-02	실제 외부 의존성 없이도 단위 테스트가 가능하도록 Stub 객체를 구현해야 한다.
NFR-O-03	시스템은 정상 동작 중 오류가 발생하지 않도록 설계되어야 하며 예외 상황 발생 시 안전한 종료 상태로 전환하는 예외 처리 로직을 포함해야 한다.
NFR-O-04	시스템은 입력 데이터에 대해 유효성 검증을 수행해야 하며 검증되지 않은 입력은 처리하지 않아야 한다.
NFR-O-05	시스템은 민감한 인증 정보를 소스 코드에 하드코딩하지 않아야 하며 안전한 저장 방식을 통해 보호되어야 한다.
NFR-O-06	회전 동작 중 센서 입력이 들어오더라도 시스템은 입력을 무시하고 회전 동작을 완수한다.
NFR-O-07	터보 청소 모드 중 먼지 센서 입력이 들어오더라도 시스템은 입력을 무시하고 터보 청소 모드를 완수한다.

Non-Functional Requirements
Other requirements (Quality)

NFR-O-08	터보 청소 모드 중 전방 및 좌측센서 입력에 따른 후진 및 회전 동작은 실행되어야 한다.
NFR-O-09	장애물 감지와 먼지 감지 중 항상 전자를 우선한다.

Non-Functional Requirements
Operating Environment

NFR-OE-01	시스템은 C++17 표준으로 구현하며 CI 빌드 환경과 로컬 개발 환경에서 동일한 CMake 3.14 버전으로 빌드가 성공해야 한다.

Non-Functional Requirements
Interface Requirements

NFR-I-01	로봇 청소기의 방향 전환, 청소 모드와 같은 상태는 터미널 화면에 실시간 텍스트 로그 형태로 출력되어야 한다.
NFR-I-02	사용자는 터미널 기반 CLI를 통해 전원 ON/OFF 명령을 입력할 수 있어야 한다.

Use Case Details

Use Case	1. Power On System
Actors	User (사용자)
Purpose	로봇청소기를 초기화하기 위함
Overview	사용자가 전원을 켜면 시스템이 초기화한다.
Type	Secondary and Breif
Cross Reference	R1.1 Use Case : Perform Automatic Cleaning, Power Off System – Exceptional
Pre-Requisites	시스템 전원이 꺼져있어야 한다.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) 사용자가 전원을 켠다. 2. (S) 시스템이 초기화된다. 3. (S) 사용자에게 정상 동작되었음을 알린다. 4. (S) 초기화 완료 후 UC-2 Perform Automatic Cleaning를 수행한다.
Alternative Courses of Events	없음.
Exceptional Courses of Events	Line 2: 에러 발생 시 UC-9 Power Off System – Exceptional을 수행한다.

Use Case Details

Use Case	2. Perform Automatic Cleaning
Actors	FrontSensor, DustSensor, Motor(offstage), Cleaner(offstage)
Purpose	자동 청소 모드를 활성화하고 전진 이동하며 바닥 및 물걸레 청소를 수행하기 위함.
Overview	자동 청소 모드가 활성화되어, 로봇 청소기는 전진하면서 바닥 청소를 수행한다.
Type	Primary and Breif
Cross Reference	R 2.1, R4.1 Use Case : Perform Automatic Cleaning, Detect Obstacle, Perform Boost Cleaning, Power Off System – Exceptional
Pre-Requisites	자동 청소 모드가 비활성화되어 있어야 한다.
Typical Courses of Events	(A) : Actor, (S) : System 1. (S) 자동 청소 모드를 활성화한다. 2. (S) 전진 이동, 바닥 청소 및 물걸레 청소를 수행한다. 3. (S) 이동 중 시스템이 전방 센서에 감지 신호를 요구한다. 4. (A) 전방 센서가 시스템에 감지 신호를 보낸다. 5. (S) 이동 중 시스템이 먼지 센서에 감지 신호를 요구한다. 6. (A) 먼지 센서가 시스템에 감지 신호를 보낸다. 7. Line 2~6을 반복한다.
Alternative Courses of Events	Line 4: (S) 장애물이 감지되면 UC-3 Detect Obstacle이 수행된다. Line 6: (S) 먼지가 감지되면 UC-7 Perform Boost Cleaning이 수행된다.
Exceptional Courses of Events	Line 1~6: 에러 발생 시 UC-9 Power Off System – Exceptional을 수행한다.

Use Case Details

Use Case	3. Detect Obstacle
Actors	LeftSensor, RightSensor, Motor(offstage), Cleaner(offstage)
Purpose	좌측 및 우측 센서를 통해 장애물을 감지하기 위함
Overview	좌측 및 우측 센서가 장애물을 감지하면 상황에 따른 Avoid UC를 호출한다.
Type	Primary and Breif
Cross Reference	R3.1, R3.2 Use Case : Perform Automatic Cleaning, Avoid Obstacle–Turn Left, Avoid Obstacle–Turn Right, Avoid Obstacle–Move Backward & Turn, Power Off System – Exceptional
Pre–Requisites	시스템이 자동 청소 모드 중 전방에 장애물을 감지해야 한다.
Typical Courses of Events	(A) : Actor, (S) : System 1. (S) 시스템이 전진 이동 및 청소를 중지한다. 2. (S) 시스템이 좌측 센서에 감지 신호를 요구한다. 3. (A) 좌측 센서가 장애물 감지 신호를 시스템에 보낸다. 4. (S) 시스템이 180도 회전 명령을 발생시킨다. 5. (S) 회전 완료 후 시스템이 좌측 센서에 감지 신호를 요구한다. 6. (A) 좌측 센서가 장애물 감지 신호를 시스템에 보낸다. 7. (S) 시스템이 180도 회전 명령을 발생시켜 원래 방향으로 복귀한다.
Alternative Courses of Events	Line 7: 전방이 막히고 좌측이 비어 있으면 UC–4 (Turn Left)를 수행한다. Line 7: 전방과 좌측이 막히고 우측이 비어 있으면 UC–5 (Turn Right)를 수행한다. Line 7: 전방, 좌측, 우측이 모두 막히면 UC–6 (Move Backward & Turn)을 수행한다.
Exceptional Courses of Events	Line 1~7: 에러 발생 시 UC–9 Power Off System – Exceptional을 수행한다.

Use Case Details

Use Case	4. Avoid Obstacle – Turn Left
Actors	Motor(offstage)
Purpose	좌회전하여 장애물을 회피하기 위함.
Overview	시스템은 좌회전을 수행하고 자동 청소를 재개한다.
Type	Primary and Breif
Cross Reference	R3.3 Use Case: Perform Automatic Cleaning, Power Off System – Exceptional
Pre-Requisites	전방 센서가 장애물을 감지하고 이동이 중지된 상태여야 한다. 좌측 센서가 장애물 없음을 확인한 상태여야 한다.
Typical Courses of Events	(A) : Actor, (S) : System 1. (S) 시스템이 좌회전 명령을 발생시킨다. 2. (S) 좌회전 완료 후 UC-2 Perform Automatic Cleaning를 수행한다.
Alternative Courses of Events	Line 1: 회전 중 장애물 감지 신호 및 먼지 감지 신호가 들어와도 시스템은 이를 무시하고 회전을 완수한다.
Exceptional Courses of Events	Line 1~2: 에러 발생 시 UC-9 Power Off System – Exceptional을 수행한다.

Use Case Details

Use Case	5. Avoid Obstacle – Turn Right
Actors	Motor(offstage)
Purpose	우회전하여 장애물을 회피하기 위함.
Overview	시스템은 우회전을 수행하고 자동 청소를 재개한다.
Type	Primary and Breif
Cross Reference	R3.4 Use Case: Perform Automatic Cleaning, Power Off System – Exceptional
Pre-Requisites	전방 및 좌측 센서가 장애물을 감지하고 이동이 중지된 상태여야 한다. 180도 회전 후 좌측 센서가 장애물 없음을 확인한 상태여야 한다.
Typical Courses of Events	(A) : Actor, (S) : System 1. (S) 시스템이 우회전 명령을 발생시킨다. 2. (S) 우회전 완료 후 UC-2 Perform Automatic Cleaning를 수행한다.
Alternative Courses of Events	Line 1: 회전 중 장애물 감지 신호 및 먼지 감지 신호가 들어와도 시스템은 이를 무시하고 회전을 완수한다.
Exceptional Courses of Events	Line 1~2: 에러 발생 시 UC-9 Power Off System – Exceptional을 수행한다.

Use Case Details

Use Case	6. Avoid Obstacle – Move Backward & Turn
Actors	LeftSensor, RightSensor, Motor(offstage)
Purpose	전방, 좌측, 우측이 모두 막혀 있을 때 후진 후 방향을 전환하여 장애물을 회피하기 위함.
Overview	전방, 좌측, 우측이 모두 막혀 있으면 시스템은 후진한 뒤 회전 가능한 방향을 판단한다.
Type	Primary and Breif
Cross Reference	R3.5 Use Case: Avoid Obstacle – Turn Left, Avoid Obstacle – Turn Right, Power Off System – Exceptional
Pre-Requisites	좌측 , 우측 센서가 장애물을 감지하고 이동이 중지된 상태여야 한다.
Typical Courses of Events	(A) : Actor, (S) : System 1. (S) 시스템이 후진 명령을 발생시킨 후 일정 거리를 후진하고 정지한다. 2. (S) 시스템이 좌측 센서에 장애물 감지 신호를 요구한다. 3. (A) 좌측 센서가 장애물 감지 신호를 시스템에 보낸다. 4. (S) 시스템이 180도 회전 명령을 발생시킨다. 5. (S) 회전 완료 후 시스템이 좌측 센서에 감지 신호를 요구한다. 6. (A) 좌측 센서가 장애물 감지 신호를 시스템에 보낸다. 7. (S) 시스템이 180도 회전 명령을 발생시켜 원래 방향으로 복귀한다.
Alternative Courses of Events	Line 7: 좌측, 우측에 모두 장애물이 감지되지 않았을 경우 UC-4 Avoid Obstacle – Turn Left, 좌측에 장애물이 감지되었을 경우 UC-5 Avoid Obstacle – Turn Right로 전이한다.
Exceptional Courses of Events	Line 1~7: 에러 발생 시 UC-9 Power Off System – Exceptional을 수행한다.

Use Case Details

Use Case	7. Perform Boost Cleaning
Actors	Cleaner(offstage)
Purpose	먼지 센서가 먼지를 감지했을 때 청소 강도를 높여 강화 청소 모드를 수행하기 위함
Overview	먼지가 감지됐을 때 청소 강도를 높이고 일정 시간 동안 강화 청소 모드를 유지한 뒤 기본 모드로 복귀한다.
Type	Primary and Breif
Cross Reference	R4.2, R4.3 Use Case: Perform Automatic Cleaning, Power Off System – Exceptional
Pre-Requisites	시스템이 자동 청소 모드(UC-2)로 동작 중이어야 한다. 먼지가 감지된 상태여야 한다.
Typical Courses of Events	(A) : Actor, (S) : System 1. (S) 강화 청소 모드를 활성화하고 타이머를 시작한다. 2. (S) 일정 시간 경과 후 강화 청소 모드를 종료한다.
Alternative Courses of Events	Line 1: 현재 강화 청소 모드인 경우, 먼지를 감지하더라도 상태를 유지하고 타이머를 재시작하지 않는다. Line 1: 강화 청소 모드 중 전방 및 좌우 장애물 감지에 따른 후진 및 회전 동작은 실행되어야 한다.
Exceptional Courses of Events	Line 1~2: 에러 발생 시 UC-9을 수행한다.

Use Case Details

Use Case	8. Power Off System
Actors	User (사용자), Motor(offstage), Cleaner(offstage)
Purpose	로봇청소기의 모든 동작을 안전하게 중지하고 시스템을 종료하기 위함
Overview	사용자가 전원을 종료하면 시스템은 현재 수행중인 모든 동작을 중지하고 종료 상태로 전환한다.
Type	Secondary and Breif
Cross Reference	R 5.1 Use Case: Power Off System – Exceptional
Pre-Requisites	시스템 전원이 켜져있어야 한다.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) 사용자가 전원 버튼을 눌러 종료 명령을 입력한다. 2. (S) 시스템이 현재 수행 중인 동작을 중지한다. 3. (S) 모든 모터와 청소 장치를 정지시킨다. 4. (S) 정상 종료 단계 수행 중임을 사용자에게 알린다. 5. (S) 시스템을 안전한 종료 상태로 전환한다.
Alternative Courses of Events	없음.
Exceptional Courses of Events	Line 1~5: 에러 발생 시 UC-9을 수행한다.

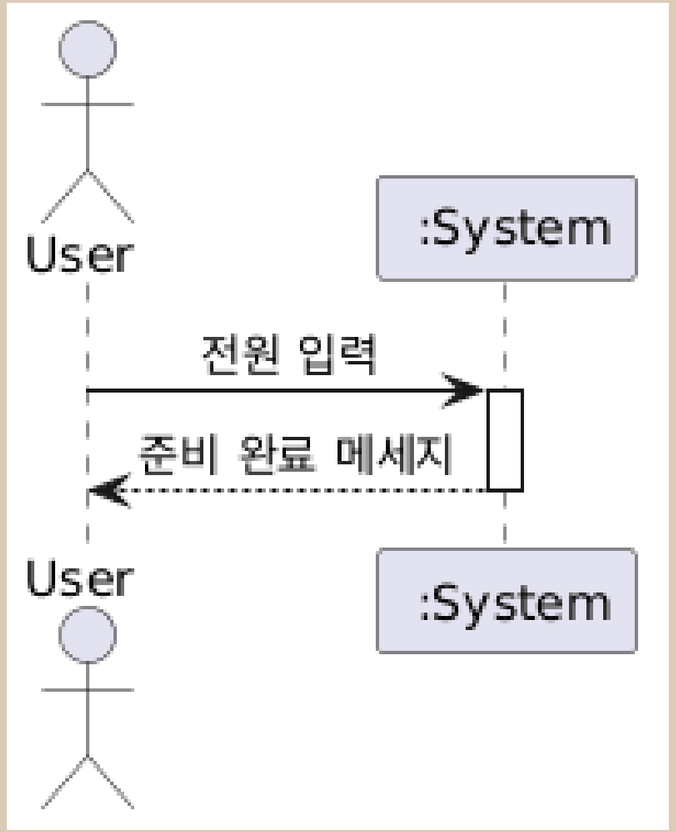
Use Case Details

Use Case	9. Power Off System – Exceptional
Actors	User (사용자: offstage), Motor(offstage), Cleaner(offstage)
Purpose	예외 발생 시 로봇청소기의 모든 동작을 안전하게 중지하고 시스템을 종료하기 위함
Overview	시스템이 예외 감지 시 종료 명령을 통해 현재 수행중인 모든 동작을 중지하고 종료 상태로 전환한다.
Type	Secondary and Breif
Cross Reference	R 6.1
Pre-Requisites	시스템이 켜져 있어야 하고, 시스템이 예외 또는 오류를 감지해야 한다.
Typical Courses of Events	(S) : System 1. (S) 감지된 예외 또는 오류를 사용자에게 알린다. 2. (S) 시스템이 현재 수행 중인 동작을 중지한다. 3. (S) 모든 모터와 청소 장치를 정지시킨다. 4. (S) 시스템을 안전한 종료 상태로 전환한다.
Alternative Courses of Events	없음.
Exceptional Courses of Events	없음.

Define System Sequence Diagrams

UC 01
Power On System

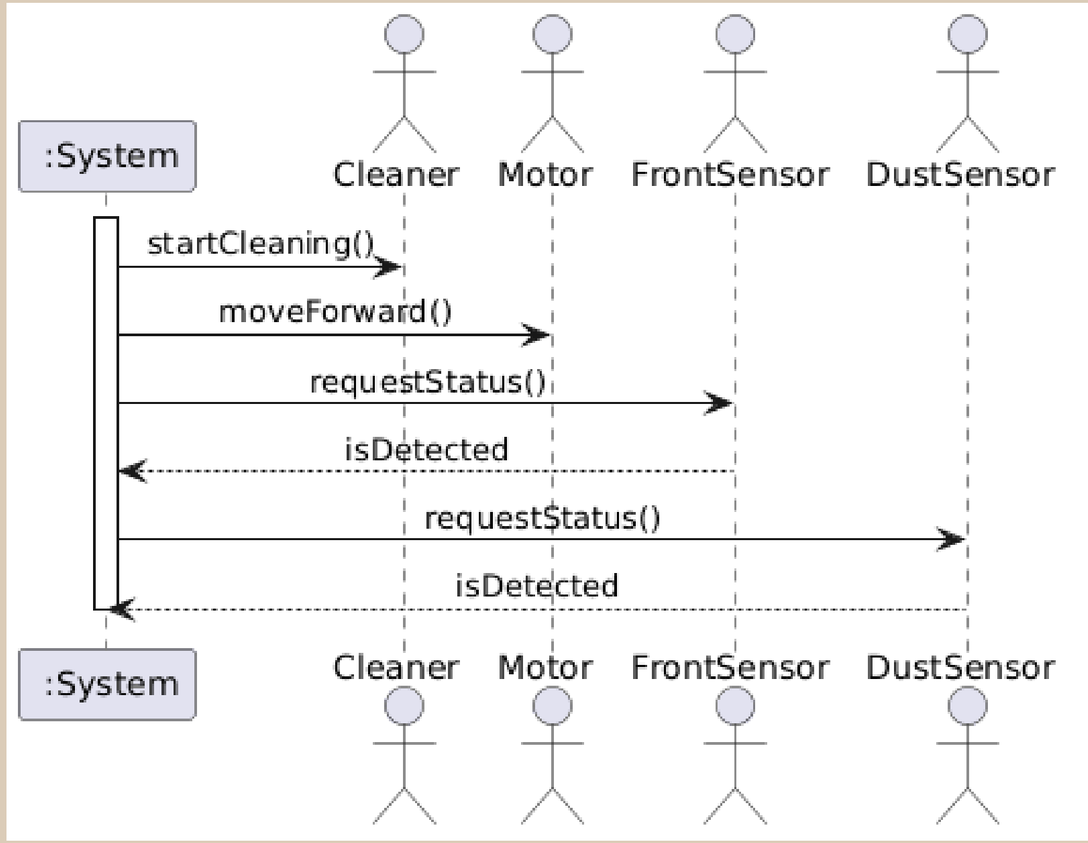
1. (A) 사용자가 전원을 켠다.
2. (S) 시스템이 초기화된다.
3. (S) 사용자에게 정상 동작되었음을 알린다.
4. (S) 초기화 완료 후 UC-2 Perform Automatic Cleaning를 수행한다.



Define System Sequence Diagrams

UC 02
Perform Automatic Cleaning

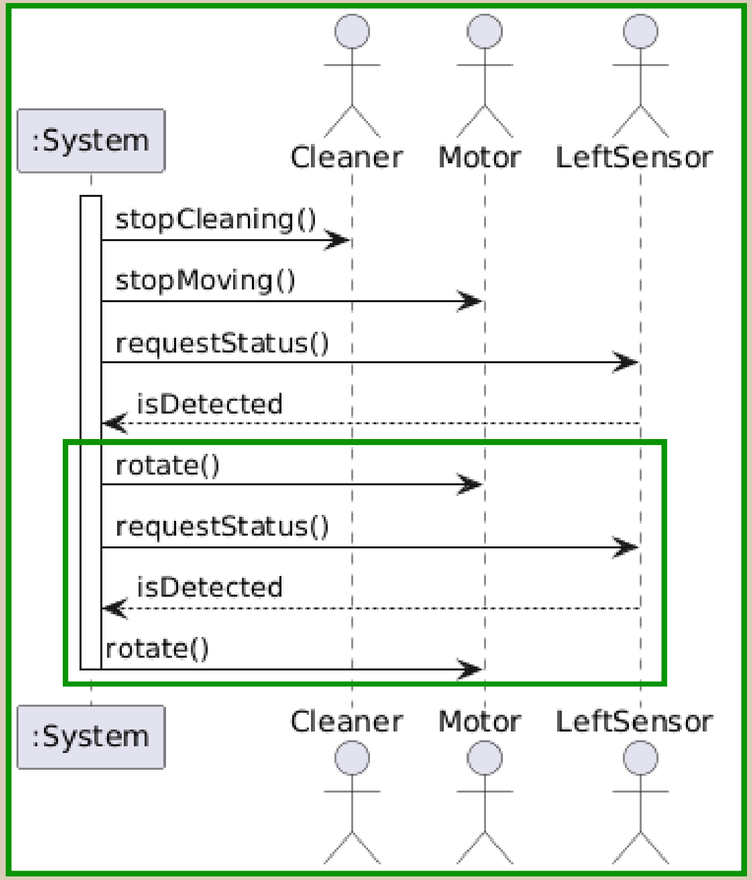
1. (S) 자동 청소 모드를 활성화한다.
2. (S) 전진 이동, 바닥 청소 및 물걸레 청소를 수행한다.
3. (S) 이동 중 시스템이 전방 센서에 감지 신호를 요구한다.
4. (A) 전방 센서가 시스템에 감지 신호를 보낸다.
5. (S) 이동 중 시스템이 먼지 센서에 감지 신호를 요구한다.
6. (A) 먼지 센서가 시스템에 감지 신호를 보낸다.
7. Line 2~6을 반복한다.



Define System Sequence Diagrams

UC 03
Detect Obstacle

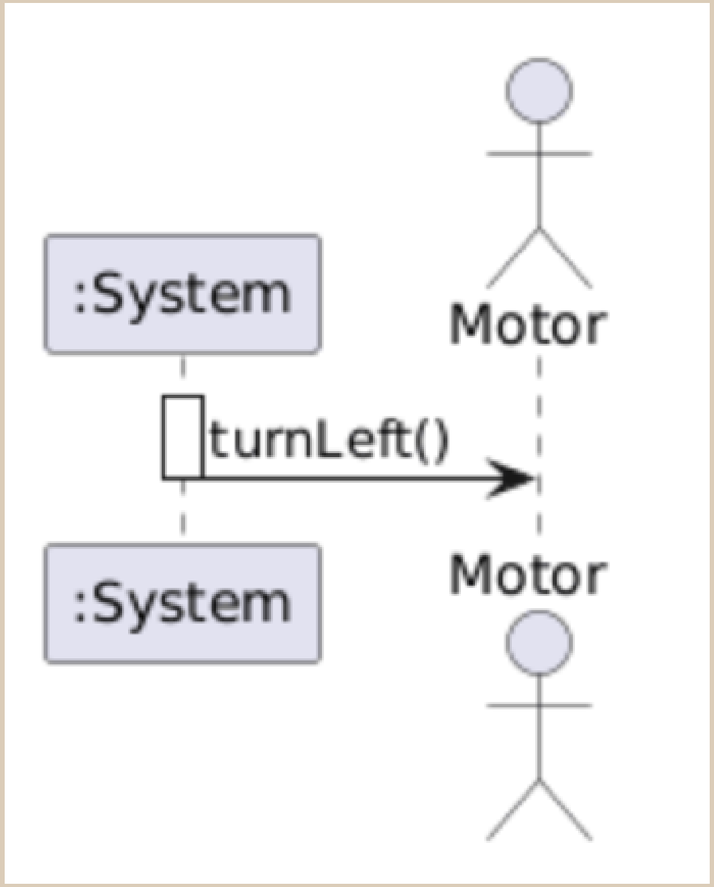
1. (S) 시스템이 전진 이동 및 자동 청소를 중지한다.
2. (S) 시스템이 좌측 센서에 감지 신호를 요구한다.
3. (A) 좌측 센서가 장애물 감지 신호를 시스템에 보낸다.
4. (S) 시스템이 180도 회전 명령을 발생시킨다.
5. (S) 회전 완료 후 시스템이 좌측 센서에 감지 신호를 요구한다.
6. (A) 좌측 센서가 장애물 감지 신호를 시스템에 보낸다.
7. (S) 시스템이 180도 회전 명령을 발생시켜 원래 방향으로 복귀한다.



Define System Sequence Diagrams

UC 04
Avoid Obstacle - Turn Left

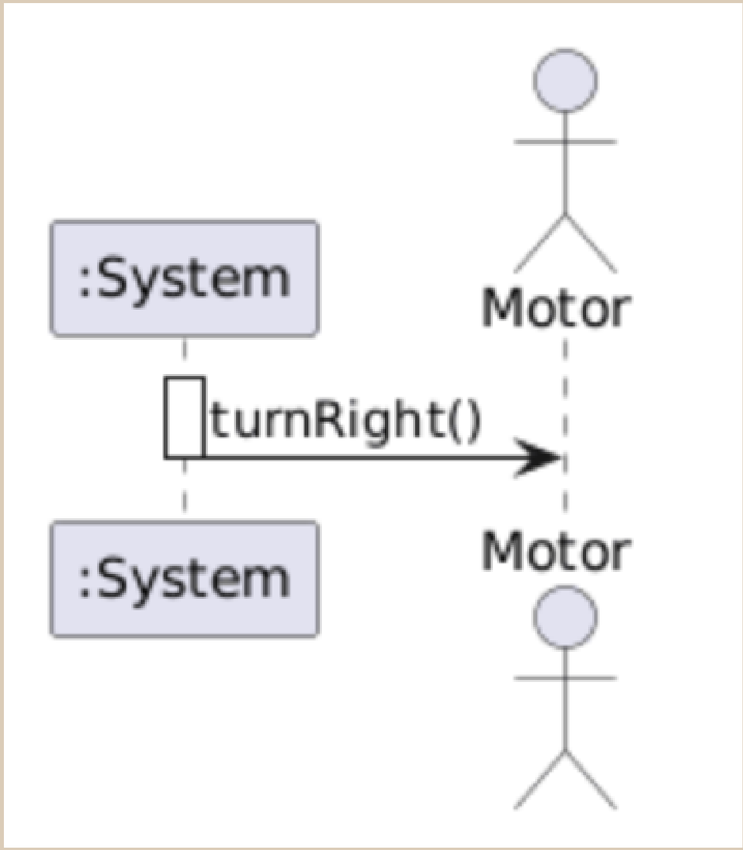
1. (S) 시스템이 좌회전 명령을 발생시킨다.



Define System Sequence Diagrams

UC 05
Avoid Obstacle - Turn Right

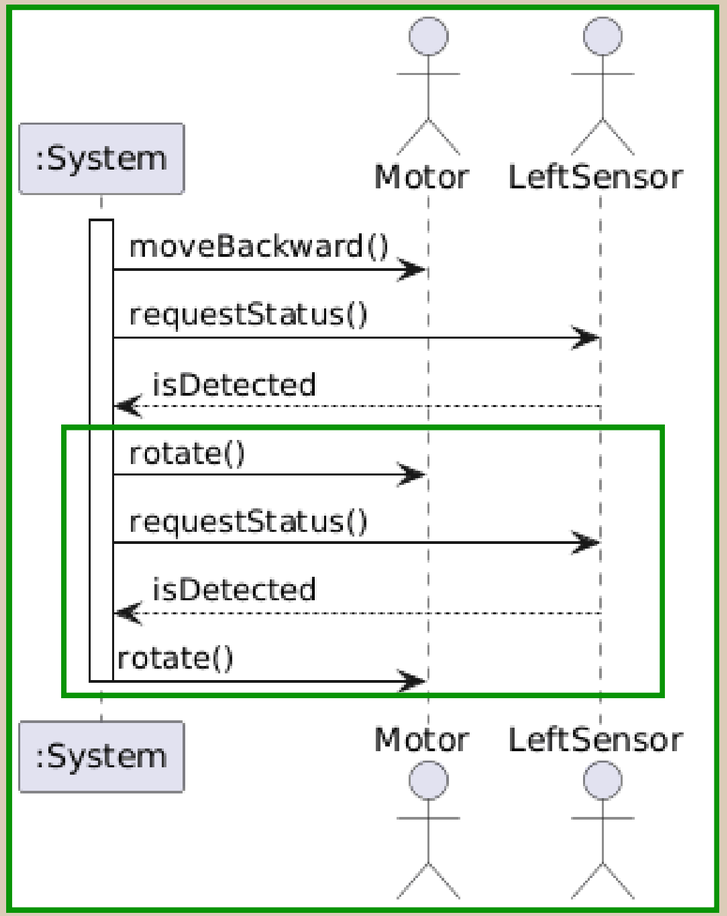
1. (S) 시스템이 우회전 명령을 발생시킨다.



Define System Sequence Diagrams

UC 06
Avoid Obstacle - Move Backward & Turn

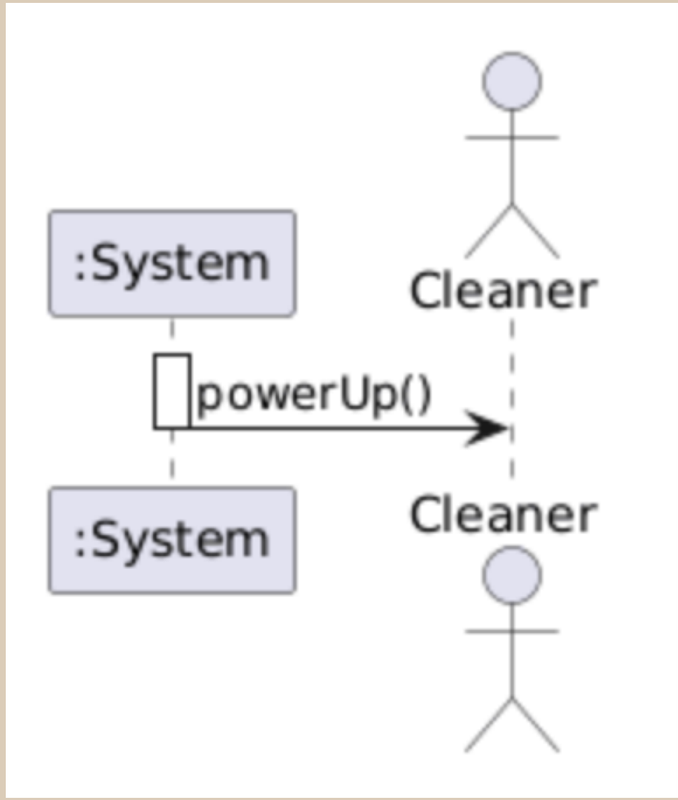
1. (S) 시스템이 후진 명령을 발생시킨 후 일정 거리를 후진하고 정지한다.
2. (S) 시스템이 좌측 센서에 장애물 감지 신호를 요구한다.
3. (A) 좌측 센서가 장애물 감지 신호를 시스템에 보낸다.
4. (S) 시스템이 180도 회전 명령을 발생시킨다.
5. (S) 회전 완료 후 시스템이 좌측 센서에 감지 신호를 요구한다.
6. (A) 좌측 센서가 장애물 감지 신호를 시스템에 보낸다.
7. (S) 시스템이 180도 회전 명령을 발생시켜 원래 방향으로 복귀한다.



Define System Sequence Diagrams

UC 07
Perform Boost Cleaning

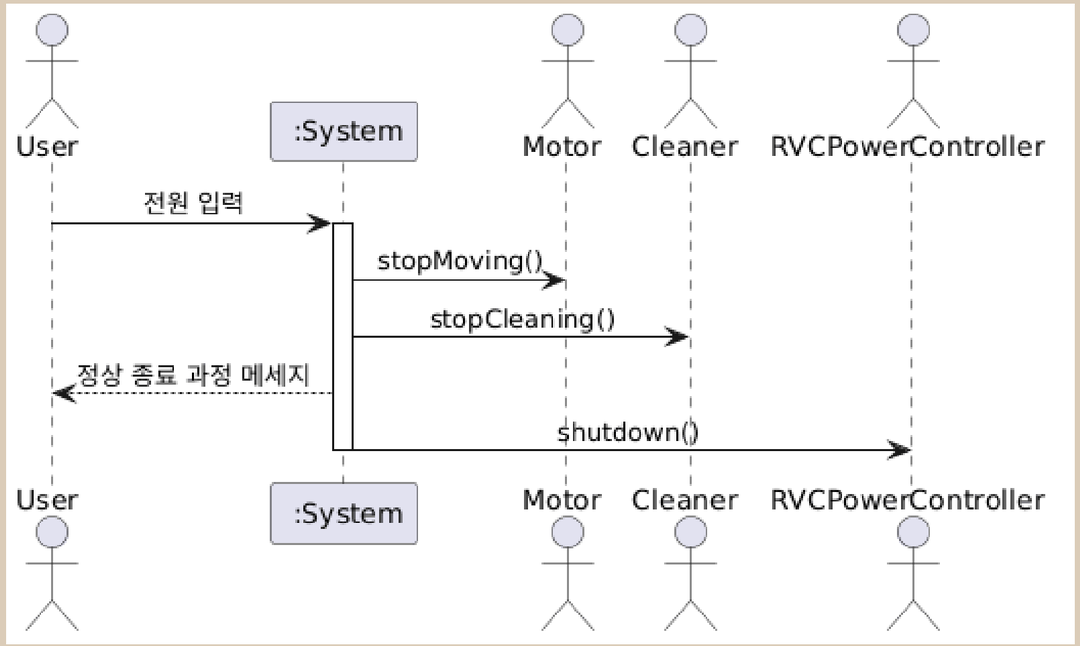
1. (S) 강화 청소 모드를 활성화하고 타이머를 시작한다.
2. (S) 일정 시간 경과 후 강화 청소 모드를 종료한다.



Define System Sequence Diagrams

UC 08
Power Off System

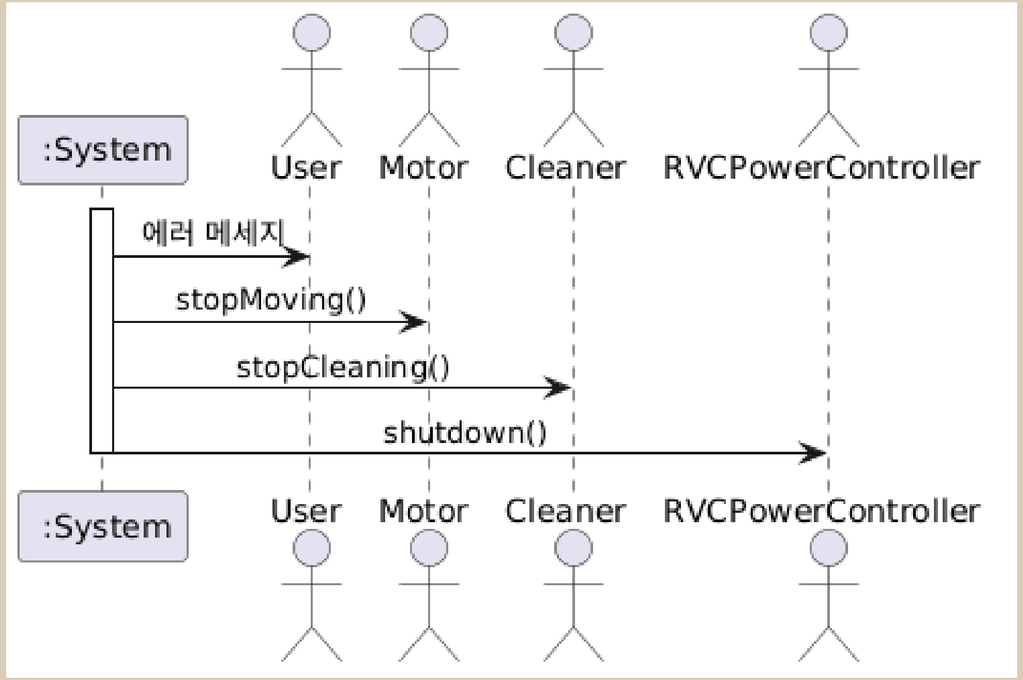
1. (A) 사용자가 전원 버튼을 눌러 종료 명령을 입력한다.
2. (S) 시스템이 현재 수행 중인 동작을 중지한다.
3. (S) 모든 모터와 청소 장치를 정지시킨다.
4. (S) 정상 종료 단계 수행 중임을 사용자에게 알린다.
5. (S) 시스템을 안전한 종료 상태로 전환한다.



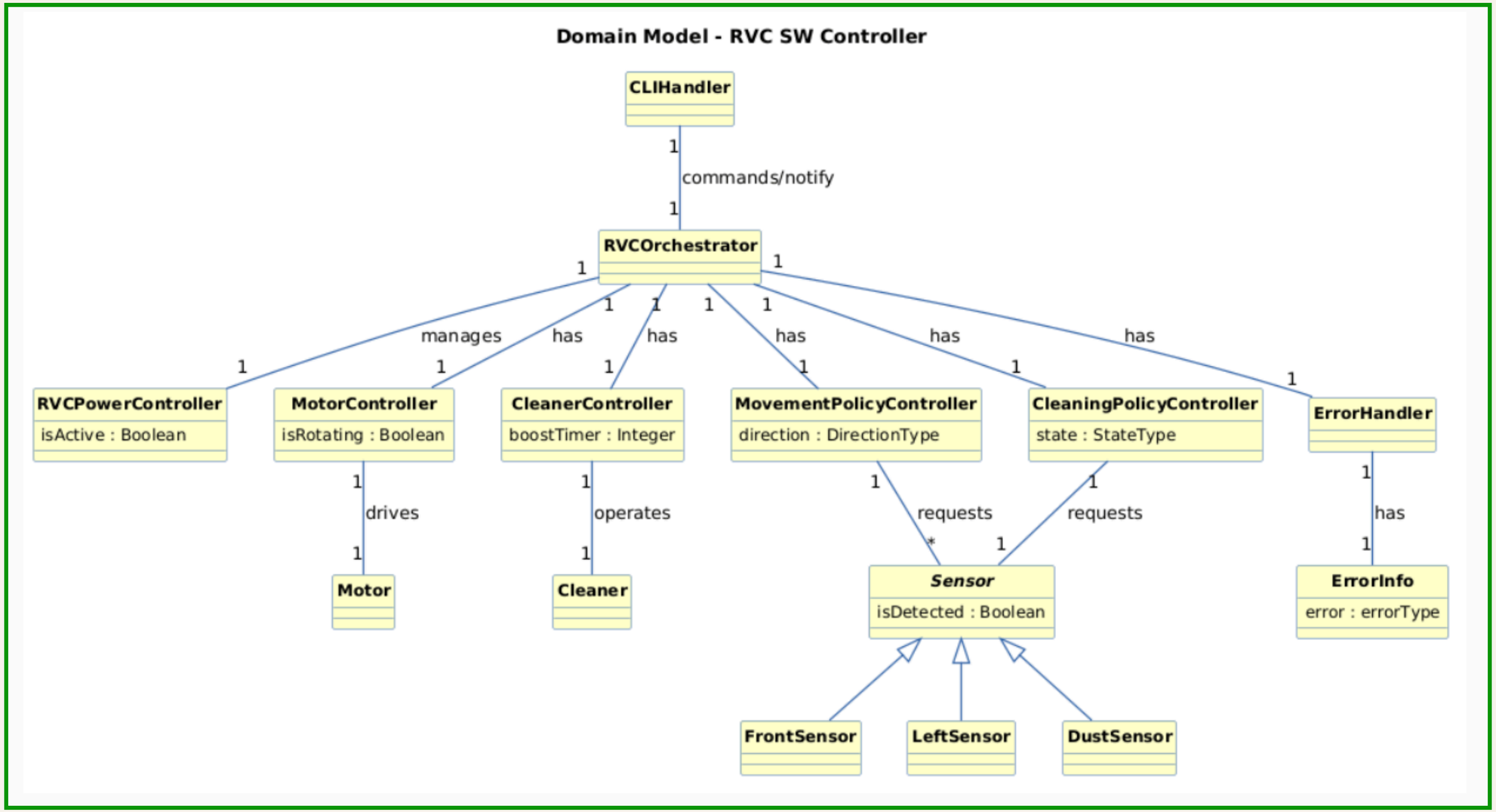
Define System Sequence Diagrams

UC 09
Power Off System - Exceptional

1. (S) 감지된 예외 또는 오류를 사용자에게 알린다.
2. (S) 시스템이 현재 수행 중인 동작을 중지한다.
3. (S) 모든 모터와 청소 장치를 정지시킨다.
4. (S) 시스템을 안전한 종료 상태로 전환한다.



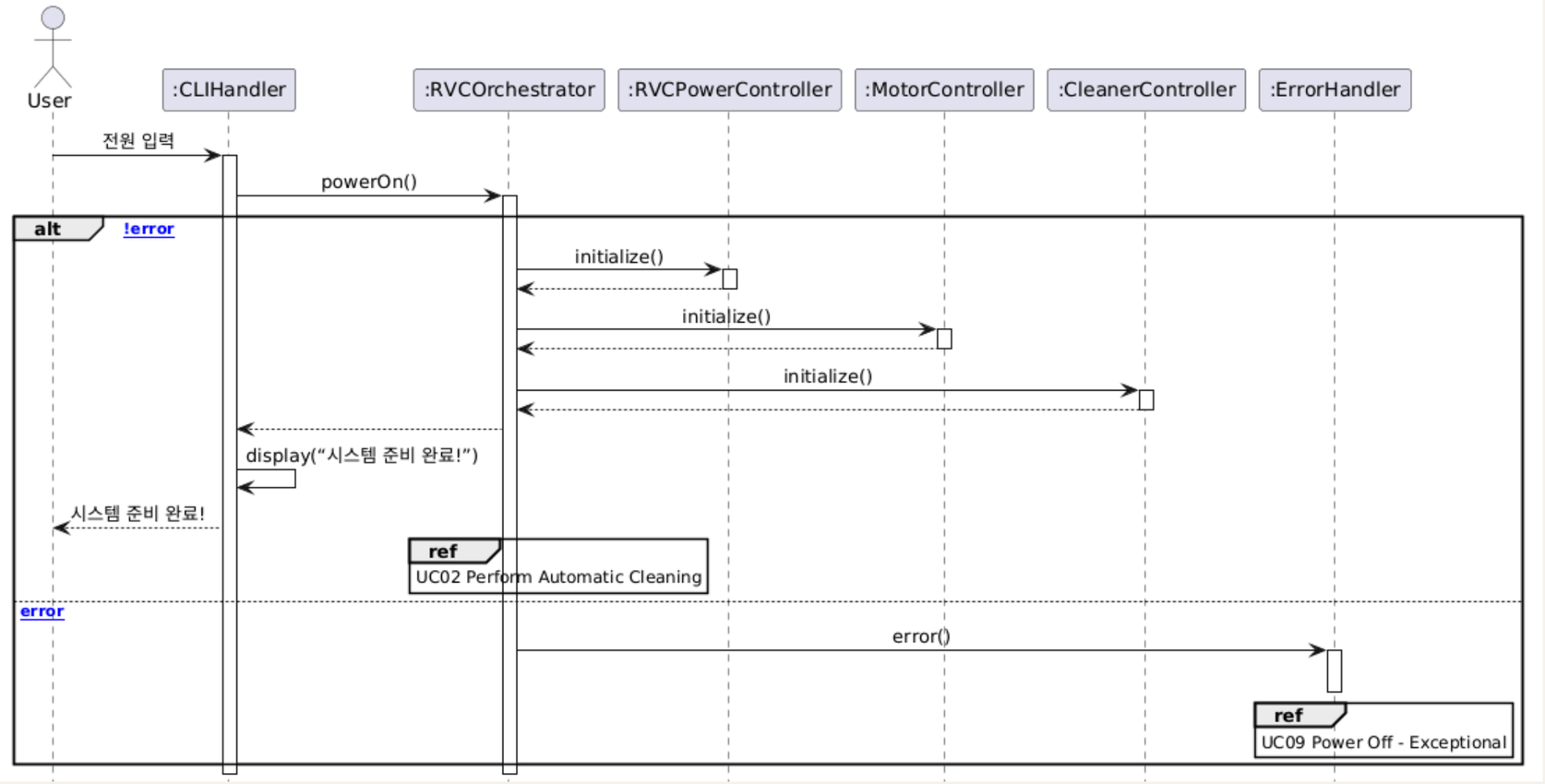
Domain Model



Sequence Diagram - Use Case 1

UseCase 1 - Power On System

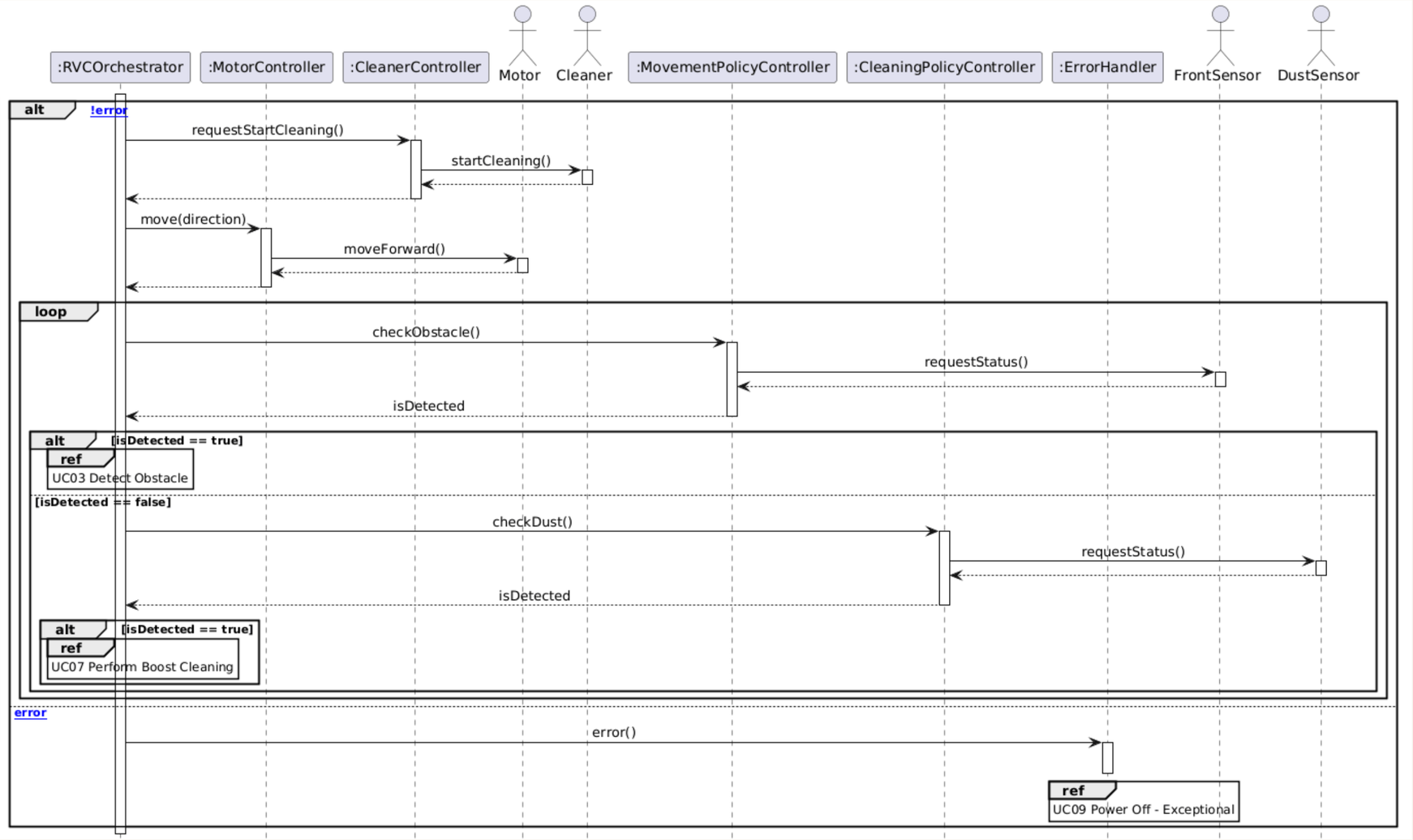
1. 사용자가 전원을 켠다.
2. 시스템이 초기화된다.
3. 사용자에게 정상 동작되었음을 알린다.
4. 초기화 완료 후 UC-2 Perform Automatic Cleaning를 수행한다.



Sequence Diagram - Use Case 2

UseCase 2 - Perform Automatic Cleaning

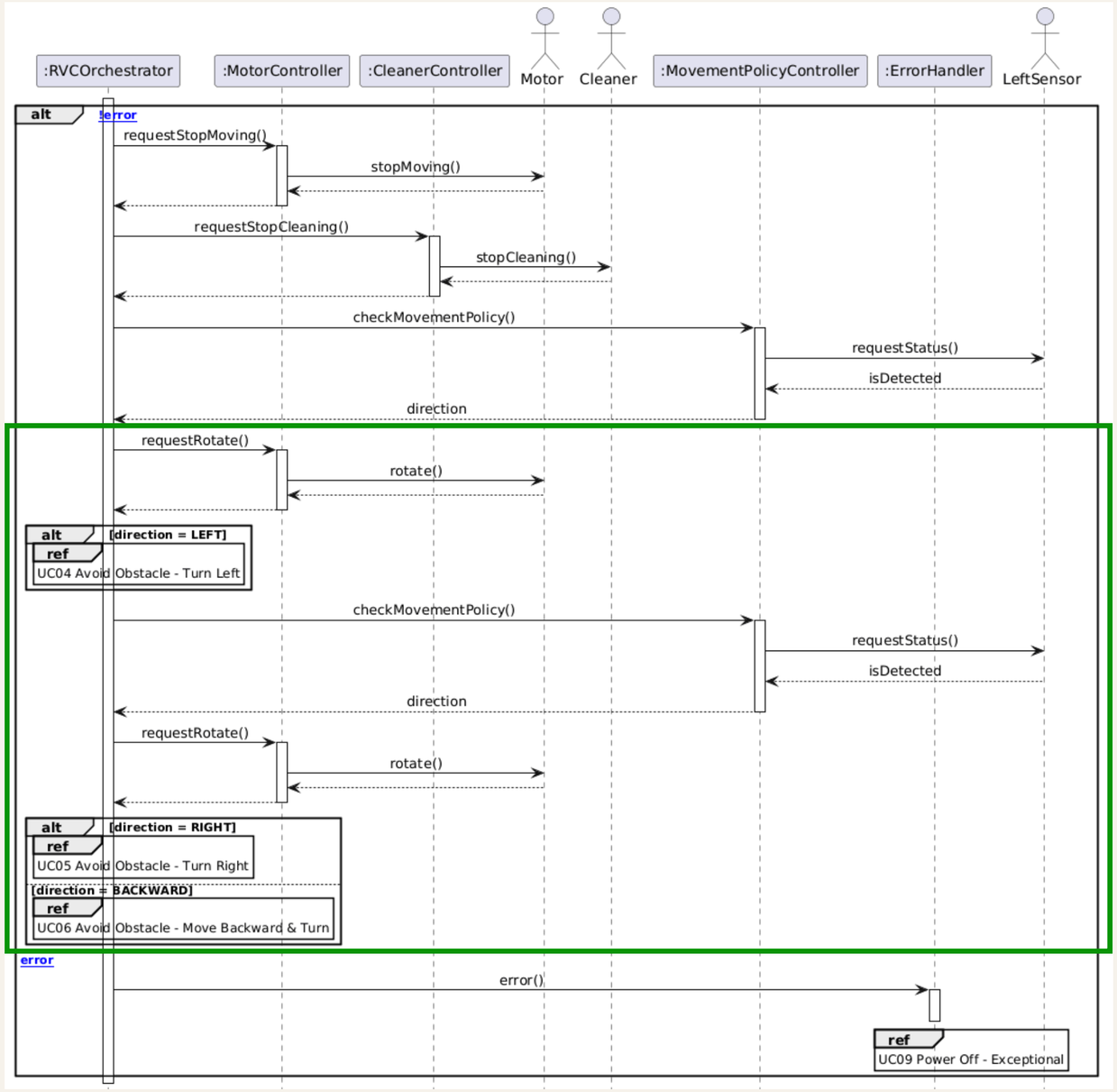
1. 자동 청소 모드를 활성화한다.
2. 전진 이동, 바닥 청소 및 물걸레 청소를 수행한다.
3. 이동 중 시스템이 전방 센서에 감지 신호를 요구한다.
4. 전방 센서가 시스템에 감지 신호를 보낸다.
5. 이동 중 시스템이 먼지 센서에 감지 신호를 요구한다.
6. 먼지 센서가 시스템에 감지 신호를 보낸다.
7. Line 2~6을 반복한다.



Sequence Diagram - Use Case 3

UseCase 3 - Detect Obstacle

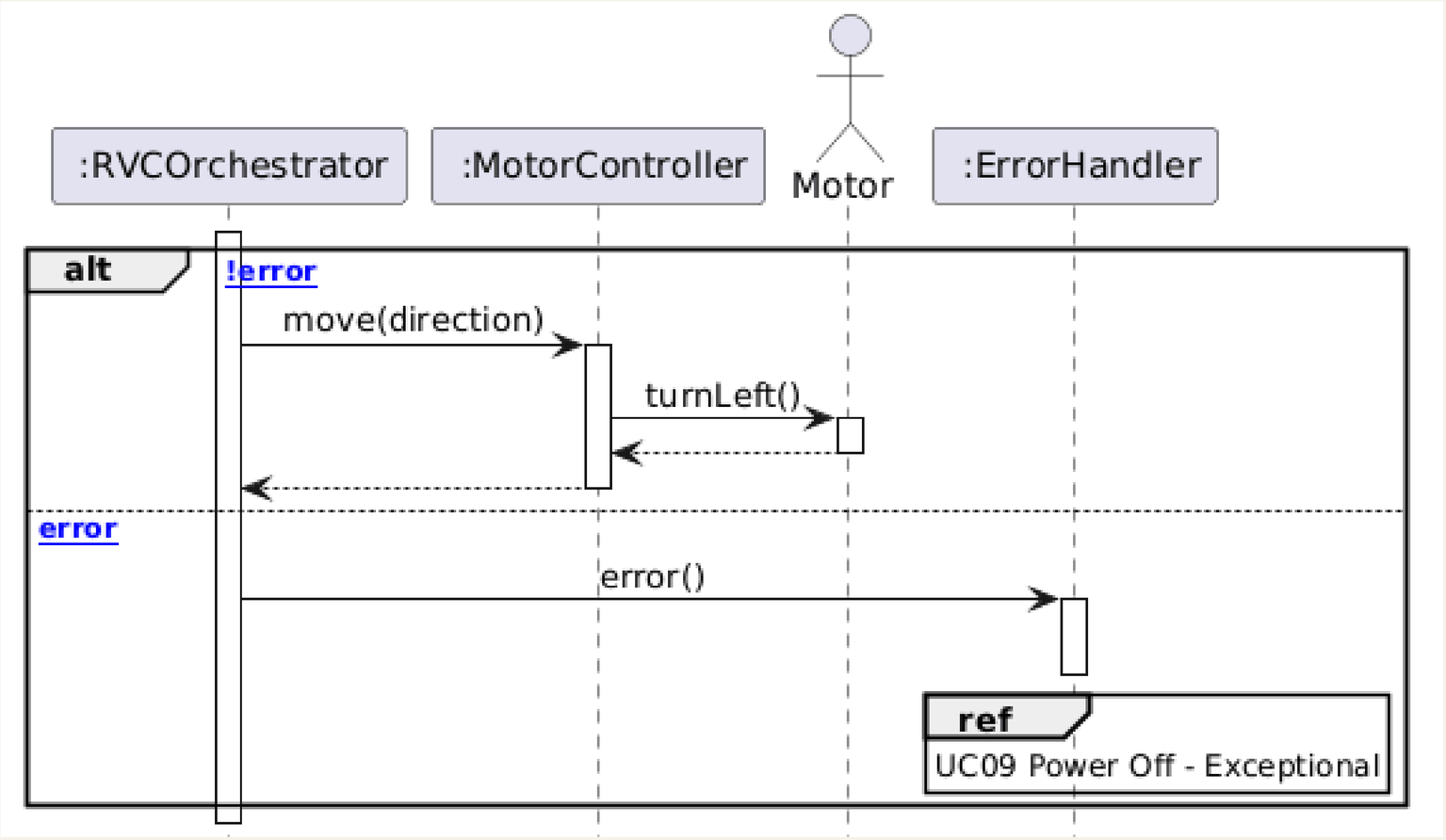
1. 시스템이 전진 이동 및 자동 청소를 중지한다.
2. 시스템이 좌측 센서에 감지 신호를 요구한다.
3. 좌측 센서가 장애물 감지 신호를 시스템에 보낸다.
4. 시스템이 180도 회전 명령을 발생시킨다.
5. 회전 완료 후 시스템이 좌측 센서에 감지 신호를 요구한다.
6. 좌측 센서가 장애물 감지 신호를 시스템에 보낸다.
7. 시스템이 180도 회전 명령을 발생시켜 원래 방향으로 복귀한다.



Sequence Diagram - Use Case 4

UseCase 4 -
Avoid Obstacle - Turn Left

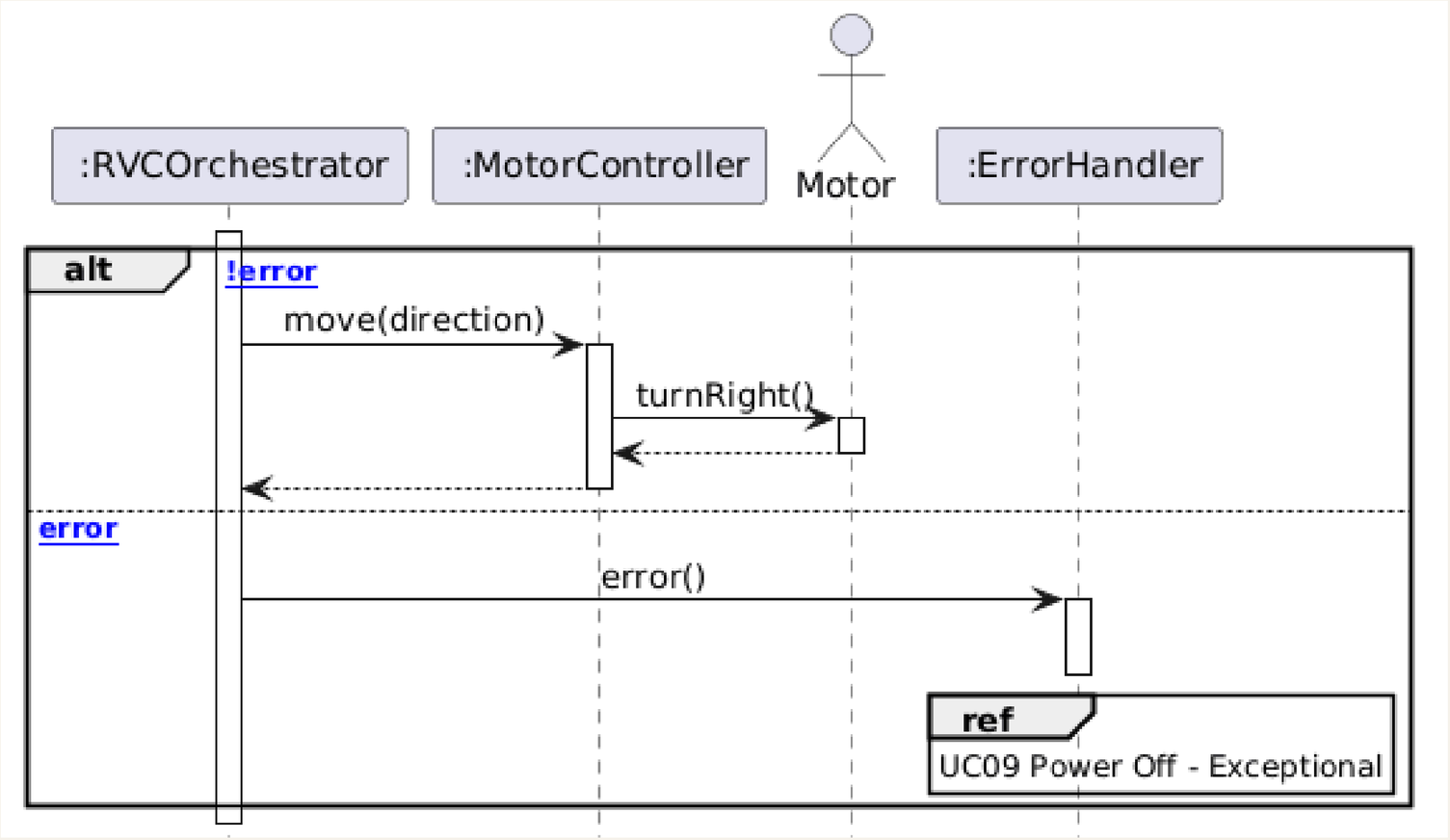
1. 시스템이 좌회전 명령을 발생시킨다.
2. ~~좌회전 완료 후 UC-2 Perform Automatic Cleaning를 수행한다.~~



Sequence Diagram - Use Case 5

UseCase 5 -
Avoid Obstacle - Turn Right

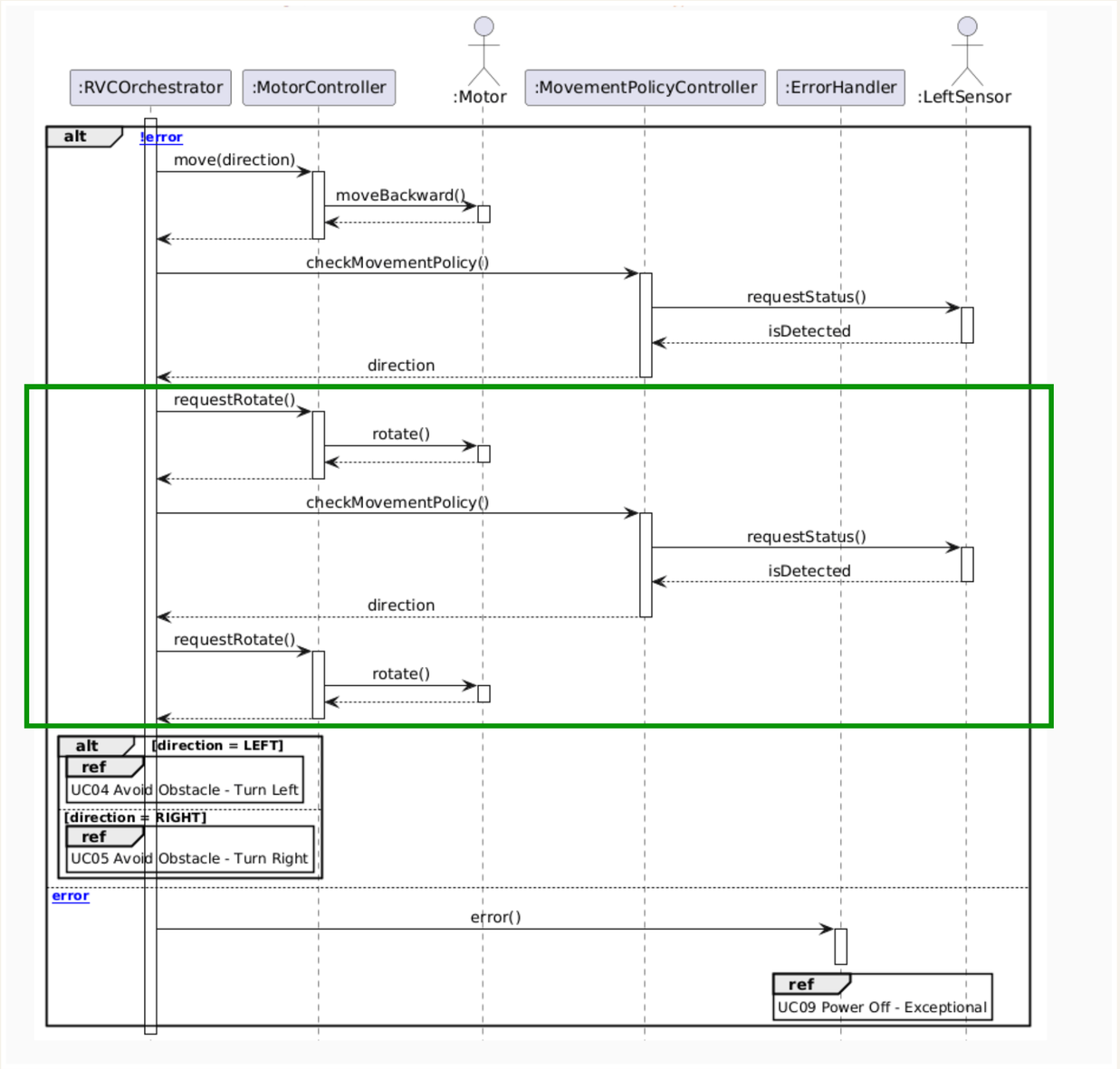
1. 시스템이 우회전 명령을 발생시킨다.
2. ~~우회전 완료 후 UC-2 Perform Automatic Cleaning를 수행한다.~~



Sequence Diagram - Use Case 6

UseCase 6 -
Avoid Obstacle
- Move Backward & Turn

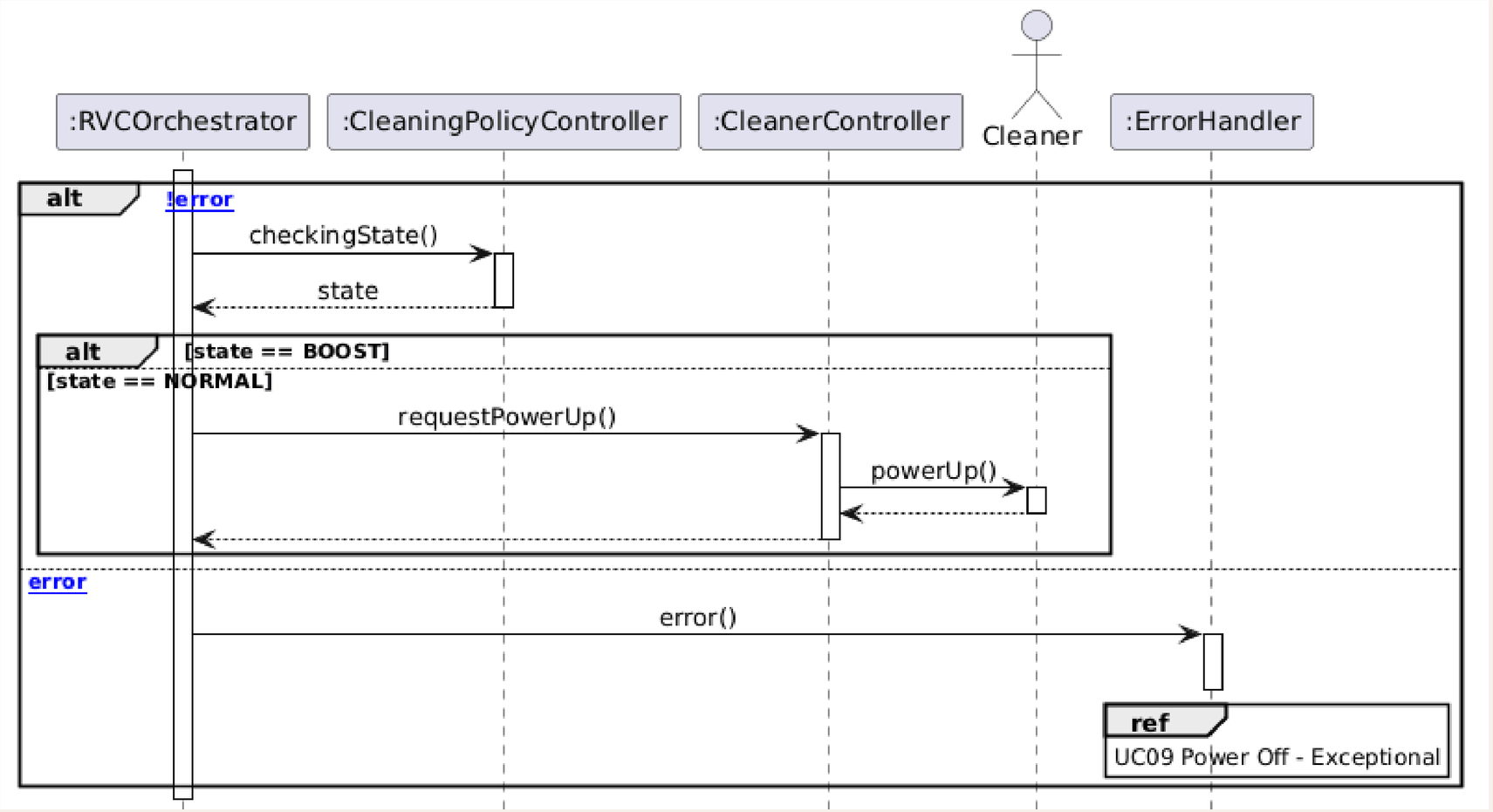
1. 시스템이 후진 명령을 발생시킨 후 일정 거리를 후진하고 정지한다.
2. 시스템이 좌측 센서에 장애물 감지 신호를 요구한다.
3. 좌측 센서가 장애물 감지 신호를 시스템에 보낸다.
4. 시스템이 180도 회전 명령을 발생시킨다.
5. 회전 완료 후 시스템이 좌측 센서에 감지 신호를 요구한다.
6. 좌측 센서가 장애물 감지 신호를 시스템에 보낸다.
7. 시스템이 180도 회전 명령을 발생시켜 원래 방향으로 복귀한다.



Sequence Diagram - Use Case 7

UseCase 7 -
Perform Boost Cleaning

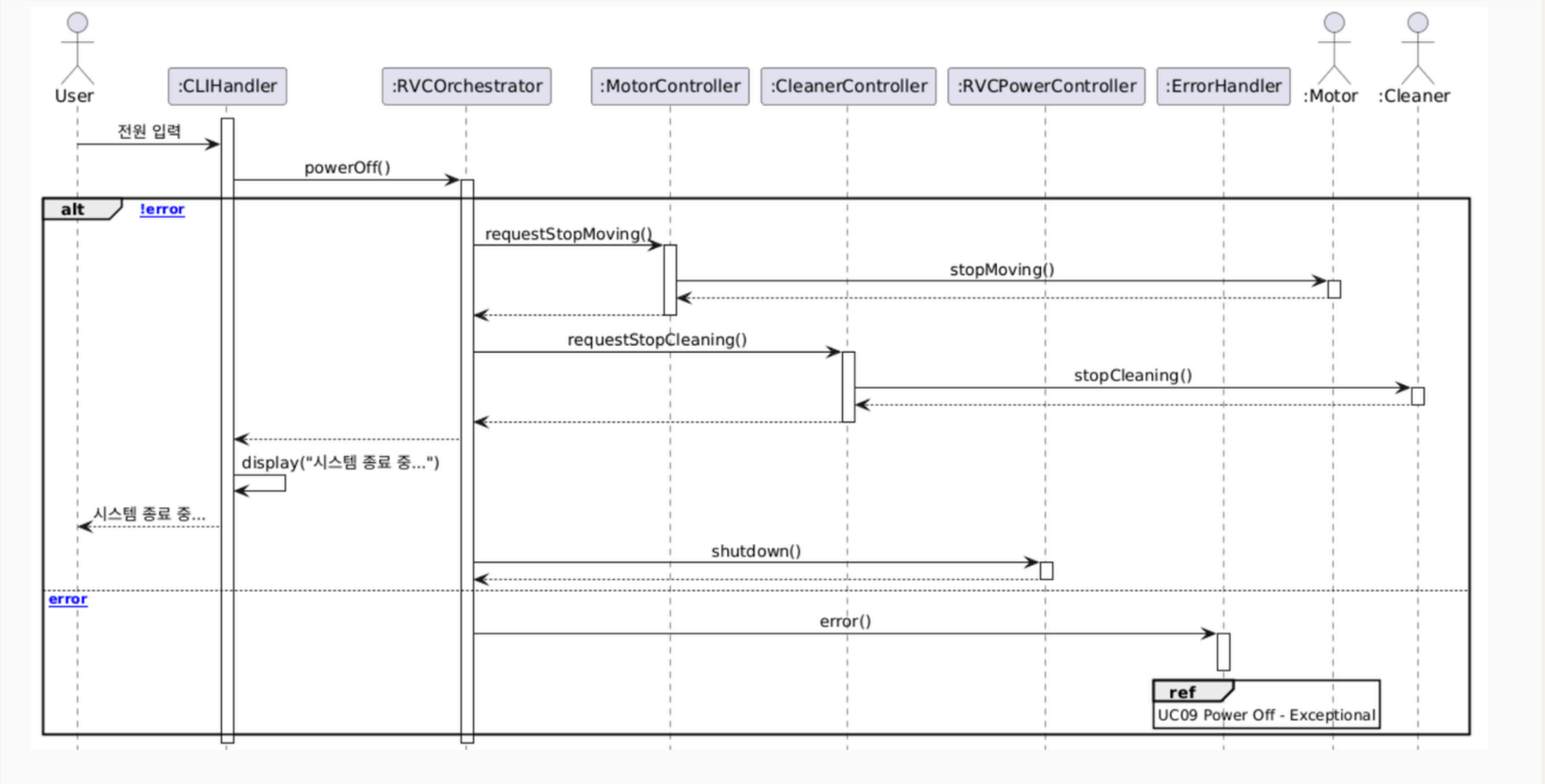
1. 강화 청소 모드를 활성화하고 타이머를 시작한다.
2. 일정 시간 경과 후 강화 청소 모드를 종료한다.



Sequence Diagram - Use Case 8

UseCase 8 - Power Off System

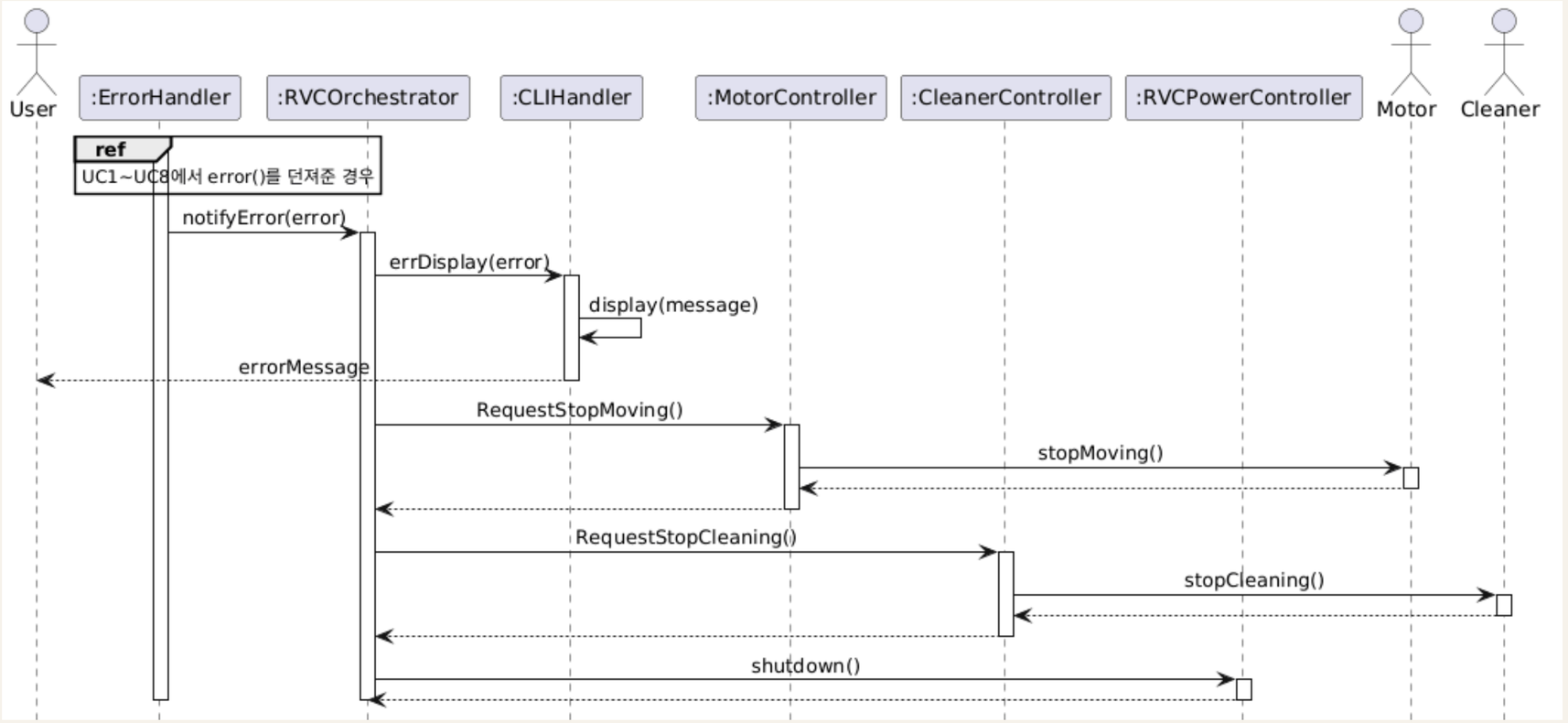
1. 사용자가 전원 버튼을 눌러 종료 명령을 입력한다.
2. 시스템이 현재 수행 중인 동작을 중지한다.
3. 모든 모터와 청소 장치를 정지시킨다.
4. 정상 종료 단계 수행 중임을 사용자에게 알린다.
5. 시스템을 안전한 종료 상태로 전환한다.



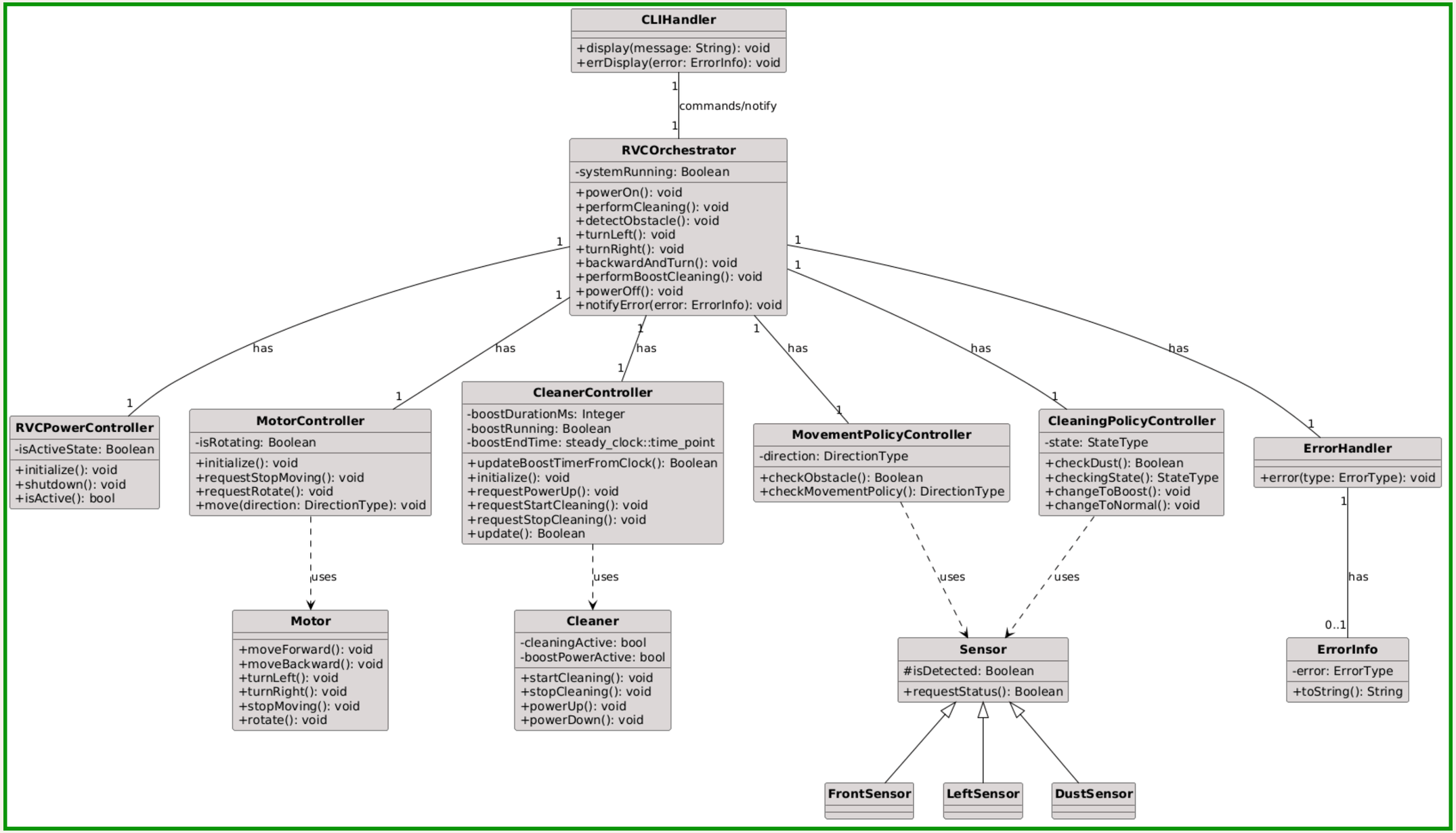
Sequence Diagram - Use Case 9

UseCase 9 -
Power Off System - Exceptional

1. 감지된 예외 또는 오류를 사용자에게 알린다.
2. 시스템이 현재 수행 중인 동작을 중지한다.
3. 모든 모터와 청소 장치를 정지시킨다.
4. 시스템을 안전한 종료 상태로 전환한다.



Class Diagram



구현 (Coding)

RVCOrchestrator.cpp

```cpp
// UC3: Detect Obstacle
void RVCOrchestrator::detectObstacle() {
    cleanerPtr->requestStopCleaning();
    motorPtr->requestStopMoving();

    DirectionType dir = movementPtr->checkMovementPolicy();
    if (dir == DirectionType::LEFT) { turnLeft(); return; }

    // 좌측 막힘: 180도 회전 후 좌측 센서로 원래 우측 방향 확인, 이후 복귀
    motorPtr->requestRotate();
    DirectionType dir2 = movementPtr->checkMovementPolicy();
    motorPtr->requestRotate();

    if (dir2 == DirectionType::LEFT) { turnRight(); return; }
    backwardAndTurn();
}
```

```cpp
// UC6: Backward & Turn
void RVCOrchestrator::backwardAndTurn() {
    motorPtr->move(DirectionType::BACKWARD);

    DirectionType dir = movementPtr->checkMovementPolicy();
    if (dir == DirectionType::LEFT) { turnLeft(); return; }

    // 좌측 막힘: 180도 회전 후 좌측 센서로 원래 우측 방향 확인, 이후 복귀
    motorPtr->requestRotate();
    DirectionType dir2 = movementPtr->checkMovementPolicy();
    motorPtr->requestRotate();

    if (dir2 == DirectionType::LEFT) { turnRight(); return; }
    // all blocked → remain stopped (fail-safe)
}
```

Unit Testing

MovementPolicyControllerTest.cpp

삭제된 테스트

TestName	TestDescription	Pass/Fail
LeftBlockedRightFreeReturnsRight	좌측 막힘·우측 자유 시 RIGHT를 반환하는지 확인	삭제
RightCalledOnlyWhenLeftBlocked	좌측 막힘 시에만 우측 센서가 호출되는지 확인	삭제
PolicyReturnedIsExactlyRight	checkMovementPolicy()가 정확히 RIGHT를 반환하는지 확인	삭제
ObstacleLeadsToPolicyRightPath	전방 장애물·좌측 막힘·우측 자유 시 RIGHT를 반환하는지 확인	삭제

Unit Testing

MovementPolicyControllerTest.cpp

수정된 테스트

TestNum	TestName	TestDescription	Pass/Fail
UT-05	CheckMovementPolicyDoesNotCallFrontSensor	checkMovementPolicy() 호출 시 front 센서를 호출하지 않는지 확인 (수정 전: right 센서 미호출 검증)	Pass
UT-07	LeftBlockedReturnsBackward	좌측 막힘 시 BACKWARD를 반환하는지 확인 (수정 전: 양측 막힘 조건에서 검증)	Pass
UT-08	LeftFreeCallsLeftSensorOnce	좌측 자유 시 left 센서가 1회 호출되는지 확인 (수정 전: right 센서 미호출 검증)	Pass
UT-10	PolicyCalledIndependentlyAfterReset	reset 후 재호출 시 BACKWARD를 반환하는지 확인 (수정 전: RIGHT 반환 검증)	Pass
UT-12	PolicyReturnedIsExactlyBackward	checkMovementPolicy()가 정확히 BACKWARD를 반환하는지 확인 (수정 전: right 센서 설정 포함)	Pass
UT-15	ObstacleLeadsToPolicyBackwardPath	전방 장애물·좌측 막힘 시 BACKWARD를 반환하는지 확인 (수정 전: right 센서 설정 포함)	Pass

Unit Testing

MovementPolicyControllerTest.cpp

추가된 테스트

TestNum	TestName	TestDescription	Pass/Fail
UT-17	CheckMovementPolicyNeverReturnsRight	좌측 자유/막힘 두 경우 모두 RIGHT를 반환하지 않는지 확인	Pass
UT-18	LeftSensorQueueFirstCallReturnsLeft	responseQueue 첫 번째 값이 false일 때 LEFT를 반환하는지 확인	Pass
UT-19	LeftSensorQueueSecondCallReturnsBackward	responseQueue 두 번째 값이 true일 때 BACKWARD를 반환하는지 확인	Pass
UT-20	MultiplePolicyCalls_LeftBlockedAlways_AllReturnBackward	좌측 센서 항상 막힘 시 3회 연속 호출해도 모두 BACKWARD를 반환하는지 확인	Pass

Unit Testing

RVCOrchestratorTest.cpp

수정된 테스트

TestNum	TestName	TestDescription	Pass/Fail
UT-09	UC3_DetectObstacle_LeftBlocked_TurnsRightNotLeft	좌측 막힘 시 180도 회전 후 우측 확인하여 TURN_RIGHT 호출·TURN_LEFT 미호출 확인 (수정 전: rightSensor 직접 사용)	Pass
UT-11	UC3_DetectObstacle_BothBlocked_BackwardOnlyNoTurn	180도 회전 후에도 양방향 막힘 시 BACKWARD만 호출되는지 확인 (수정 전: rightSensor 직접 사용)	Pass
UT-23	UC6_BackwardAndTurn_LeftBlocked_CallsBackwardThenTurnRight	후진 후 좌측 막힘 시 180도 회전으로 우측 확인하여 TURN_RIGHT 호출 확인 (수정 전: rightSensor 직접 사용)	Pass
UT-25	UC6_BackwardAndTurn_BothBlocked_OnlyBackwardCalled	후진 후 180도 회전해도 양방향 막힘 시 BACKWARD만 호출되는지 확인 (수정 전: rightSensor 직접 사용)	Pass

Unit Testing

RVCOrchestratorTest.cpp

추가된 테스트

TestNum	TestName	TestDescription	Pass/Fail
UT-10	UC3_DetectObstacle_LeftBlocked_RotateIsCalled	장애물 감지 시 좌측 막힘이면 우측 확인을 위한 180도 회전(requestRotate)이 실제로 호출되는지 확인	Pass
UT-12	UC3_DetectObstacle_BothBlocked_RotateIsCalled	장애물 감지 시 양방향 막힘이어도 180도 회전(requestRotate)이 호출되는지 확인	Pass
UT-24	UC6_BackwardAndTurn_LeftBlocked_RotateIsCalledAndCountIs4	후진 후 좌측 막힘 시 rotate 호출 여부와 모터 호출 횟수(backward+rotate+rotate+turnRight=4)를 확인	Pass
UT-26	UC6_BackwardAndTurn_BothBlocked_RotateIsCalled 호출되는지 확인	후진 후 양방향 막힘 시에도 180도 회전(requestRotate)이	Pass

Unit Testing

1. Test Case 분석

(1) 전원, 에러 관련 (UC1/UC8/UC9)
- 총 29개 Test Case
- 초기화, 종료, 에러 처리 포함

(2) 청소 모드 관련 (UC2/UC7 등)
- 총 52개 Test Case
- 초기화, 종료, 에러 처리 포함

(3) 이동·장애물 관련 (UC3~6)
- 총 82(+4)개 테스트 케이스
- 방향 전환, 센서 감지, 이동 명령 포함
- 우측 센서 삭제 및 180도 회전 검증 로직 포함

(4) UI/Handler 관련
- 총 17개 Test Case
- 출력 포맷, 에러 메세지 포함

(5) 시스템 통합 (RVCOrchestrator)
- 총 40(+4)개 Test Case
- UC별 Orchestrator 흐름 검증
- 180도 회전(requestRotate) 메커니즘 검증 추가

System Test

RVCOrchestratorTest.cpp

TestNum	TestName	TestDescription	Pass/Fail
ST-33 (추가)	EnvRightFree_RotateIsCalledForRightCheck	좌측 막힘·우측 자유 환경에서 장애물 감지 시 우측 확인을 위한 180도 회전이 실제로 호출되는지 확인	Pass
ST-34 (추가)	EnvBothBlocked_RotateIsCalledForRightCheck	양방향 막힘 환경에서 장애물 감지 시에도 우측 확인을 위한 180도 회전이 호출되는지 확인	Pass
ST-45 (추가)	EnvRightFree_RotateIsCalledForRightCheck	좌측 막힘·우측 자유 환경에서 후진 후 우측 확인을 위한 180도 회전이 실제로 호출되는지 확인	Pass
ST-46 (추가)	EnvBothBlocked_RotateIsCalledForRightCheck	양방향 막힘 환경에서 후진 후에도 우측 확인을 위한 180도 회전이 호출되는지 확인	Pass

System Test

개요

1. Pass Ratio = 72/72(+4) (100%)
2. Positive Test Case (22개)
3. Negative Test Case (50개)
4. Neagtive Test Case Ratio = 50/72 (70%)

Flow	Test Case 개수	Positive Test	Negative Test
Flow1. Power Lifecycle	10	3	7
Flow2. Cleaning Session	10	3	7
Flow3. Obstacle Avoidance	14(+2)	4	10(+2)
Flow4. Backward And Turn	11(+2)	3	8(+2)

Flow	Test Case 개수	Positive Test	Negative Test
Flow5. Boost Cleaning	9	3	6
Flow6. Error Handling	9	4	5
Flow7. Complete Session	9	3	6
합계	72(+4)	22	50(+4)

System Test

(2) 시뮬레이터 테스트 화면

```
heeedragon@heeyongkim-MacBookAir CICD-Team7 % ./build/oop
전원을 켜려면 Enter, 종료하려면 'q'를 입력하세요.

[RVC] 시스템 준비 완료!
q
[RVC] 시스템 종료 중...
```

```
[PASS] Flow2_CleaningSession::Cleaning_CleanerDoesNotEnterBoost
[PASS] Flow2_CleaningSession::Cleaning_MotorDoesNotTurnLeft
[PASS] Flow2_CleaningSession::Cleaning_MotorDoesNotTurnRight
[PASS] Flow2_CleaningSession::Cleaning_MotorDoesNotMoveBackward
[PASS] Flow2_CleaningSession::Cleaning_DoesNotOutputError
[PASS] Flow3_ObstacleAvoidance::EnvLeftFree_MotorEntersTurnLeftState
[PASS] Flow3_ObstacleAvoidance::EnvRightFree_MotorEntersTurnRightState
[PASS] Flow3_ObstacleAvoidance::EnvBothBlocked_MotorEntersBackwardState
[PASS] Flow3_ObstacleAvoidance::Cleaning_ThenObstacle_CleanerBecomesIdle
[PASS] Flow3_ObstacleAvoidance::Obstacle_ThenResumeCleaning_MotorReturnsForward
[PASS] Flow3_ObstacleAvoidance::EnvLeftFree_MotorDoesNotTurnRight
[PASS] Flow3_ObstacleAvoidance::EnvLeftFree_MotorDoesNotMoveBackward
[PASS] Flow3_ObstacleAvoidance::EnvRightFree_MotorDoesNotTurnLeft
[PASS] Flow3_ObstacleAvoidance::EnvRightFree_MotorDoesNotMoveBackward
[PASS] Flow3_ObstacleAvoidance::EnvBothBlocked_MotorDoesNotTurnLeft
[PASS] Flow3_ObstacleAvoidance::EnvBothBlocked_MotorDoesNotTurnRight
[PASS] Flow3_ObstacleAvoidance::Obstacle_CleanerDoesNotEnterBoost
[PASS] Flow3_ObstacleAvoidance::EnvRightFree_RotateIsCalledForRightCheck
[PASS] Flow3_ObstacleAvoidance::EnvBothBlocked_RotateIsCalledForRightCheck
[PASS] Flow4_BackwardAndTurn::EnvLeftFree_MotorFinalStateTurnLeft
[PASS] Flow4_BackwardAndTurn::EnvRightFree_MotorFinalStateTurnRight
[PASS] Flow4_BackwardAndTurn::AfterObstacle_BackwardAndTurn_MotorMovesBackward
[PASS] Flow4_BackwardAndTurn::EnvBothBlocked_MotorStaysBackward_NoTurn
[PASS] Flow4_BackwardAndTurn::BackwardAndTurn_CleanerModeUnchanged
[PASS] Flow4_BackwardAndTurn::EnvLeftFree_MotorDoesNotTurnRight_AfterBackward
[PASS] Flow4_BackwardAndTurn::EnvRightFree_MotorDoesNotTurnLeft_AfterBackward
[PASS] Flow4_BackwardAndTurn::BackwardAndTurn_CleanerDoesNotEnterBoost
[PASS] Flow4_BackwardAndTurn::BackwardAndTurn_CleanerDoesNotStartCleaning
[PASS] Flow4_BackwardAndTurn::EnvRightFree_RotateIsCalledForRightCheck
[PASS] Flow4_BackwardAndTurn::EnvBothBlocked_RotateIsCalledForRightCheck
[PASS] Flow5_BoostCleaning::Boost_CleanerEntersBoostMode
[PASS] Flow5_BoostCleaning::Cleaning_ThenBoost_CleanerUpgradesToBoost
[PASS] Flow5_BoostCleaning::Boost_ThenPressOff_CleanerReturnsToIdle
[PASS] Flow5_BoostCleaning::Boost_CleanerDoesNotBecomeIdle
[PASS] Flow5_BoostCleaning::Boost_MotorDoesNotMoveForward
[PASS] Flow5_BoostCleaning::Boost_MotorDoesNotMoveBackward
[PASS] Flow5_BoostCleaning::Boost_MotorDoesNotTurnLeft
[PASS] Flow5_BoostCleaning::Boost_MotorDoesNotTurnRight
[PASS] Flow5_BoostCleaning::Boost_DoesNotOutputError
[PASS] Flow6_ErrorHandling::Error_MotorEntersStoppedState
[PASS] Flow6_ErrorHandling::Cleaning_ThenError_CleanerSafeStop
[PASS] Flow6_ErrorHandling::Error_ThenPressOff_HardwareRemainsInSafeState
[PASS] Flow6_ErrorHandling::Error_CleanerDoesNotResumeCleaning
[PASS] Flow6_ErrorHandling::Error_CleanerDoesNotEnterBoost
[PASS] Flow6_ErrorHandling::Error_MotorDoesNotMoveForward
[PASS] Flow6_ErrorHandling::Error_OutputDoesNotContainRVCPrefix
[PASS] Flow6_ErrorHandling::Error_OutputContainsErrorType
[PASS] Flow6_ErrorHandling::Error_MotorDoesNotMoveBackward
[PASS] Flow7_CompleteSession::UC1_Avoid_UC8_FinalSafeState
[PASS] Flow7_CompleteSession::UC2_BackwardAvoid_UC2_MotorReturnsForward
[PASS] Flow7_CompleteSession::UC1_Boost_UC8_FinalSafeState
[PASS] Flow7_CompleteSession::UC9_ThenUC8_SafeShutdownMessageDisplayed
[PASS] Flow7_CompleteSession::UC2_Obstacle_UC8_CleanerIsIdle
[PASS] Flow7_CompleteSession::UC1_UC8_NoErrorOutput
[PASS] Flow7_CompleteSession::UC9_MotorDoesNotResumeAfterError
[PASS] Flow7_CompleteSession::UC8_CleanerDoesNotAutoRestart
[PASS] Flow7_CompleteSession::UC2_UC3_MotorNotForwardAfterObstacle

=== System Test Results ===
Passed: 72
Failed: 0
Total : 72
```

System Test

(3) 테스트 수행 과정 및 결과 (CI)

Static Code Analysis

(1) Clang-Tidy, Cppcheck

CICD 알림 앱 오후 7:55

📝 **Cppcheck 분석 결과:** `OOP-konkuk/CICD-Team7` - 20/merge

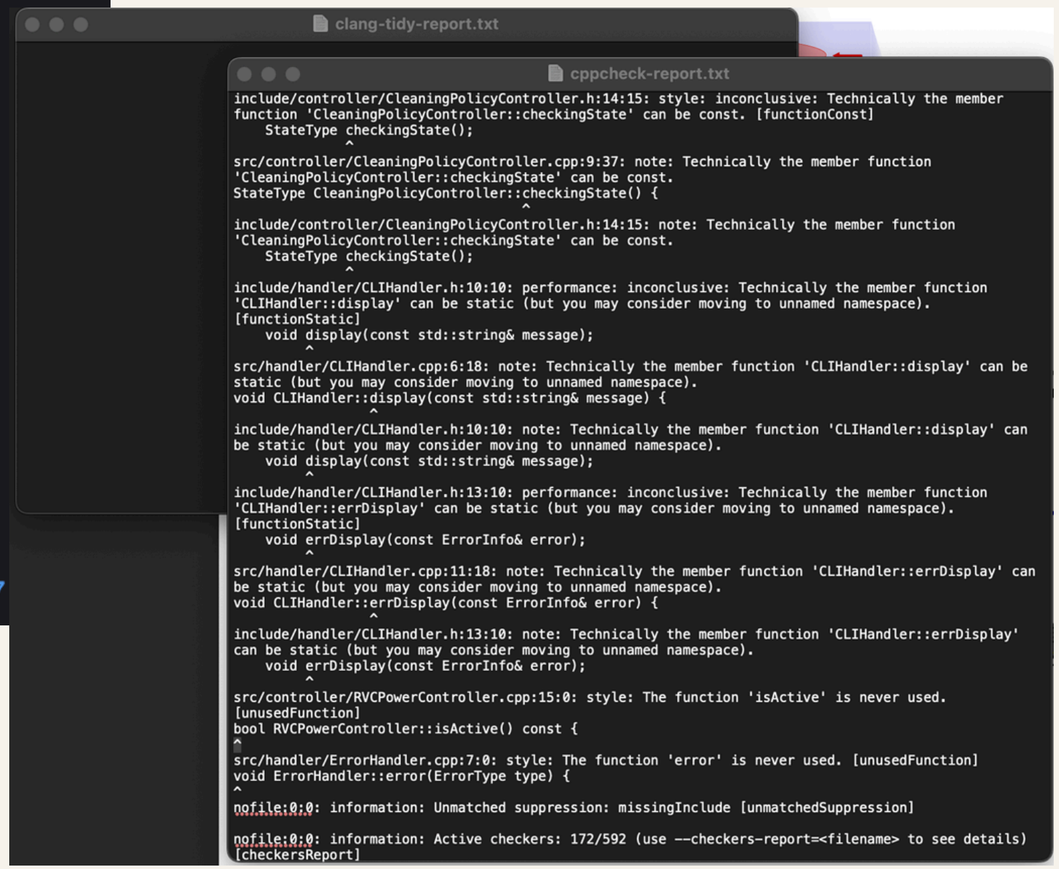
- Errors: 0
- Warnings: 0
- Style issues: 3
- Performance issues: 2
- Total: 37

🔍 **Clang-Tidy 분석 결과:** `OOP-konkuk/CICD-Team7` - 20/merge

- Errors: 0
- Warnings: 0
- Total: 0

✅ 빌드/테스트 성공: `OOP-konkuk/CICD-Team7` - 20/merge

📊 커버리지 리포트: https://app.codecov.io/gh/OOP-konkuk/CICD-Team7



clang-tidy-report.txt

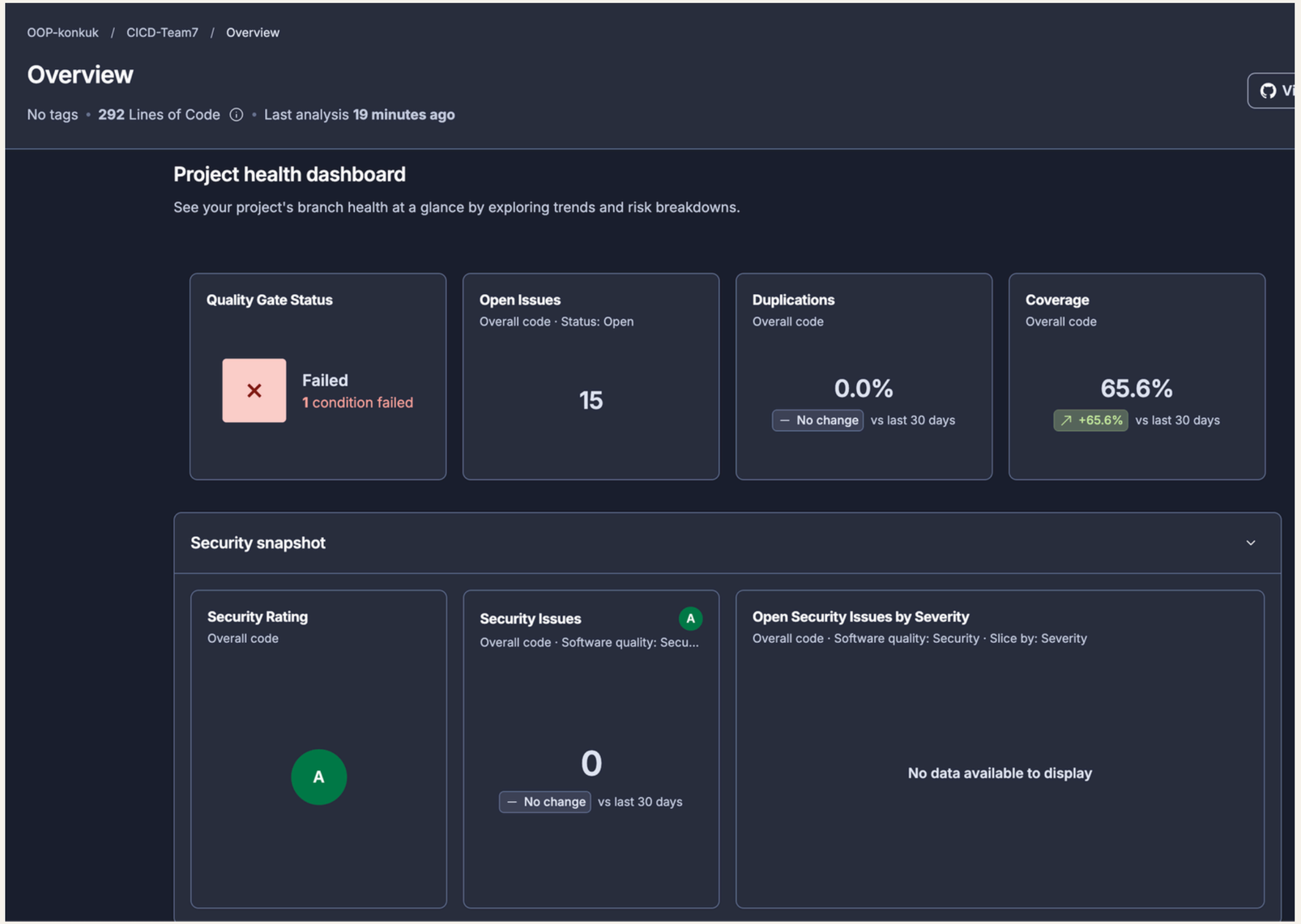
cppcheck-report.txt

```
include/controller/CleaningPolicyController.h:14:15: style: inconclusive: Technically the member
function 'CleaningPolicyController::checkingState' can be const. [functionConst]
    StateType checkingState();
              ^
src/controller/CleaningPolicyController.cpp:9:37: note: Technically the member function
'CleaningPolicyController::checkingState' can be const.
StateType CleaningPolicyController::checkingState() {
                                    ^
include/controller/CleaningPolicyController.h:14:15: note: Technically the member function
'CleaningPolicyController::checkingState' can be const.
    StateType checkingState();
              ^
include/handler/CLIHandler.h:10:10: performance: inconclusive: Technically the member function
'CLIHandler::display' can be static (but you may consider moving to unnamed namespace).
[functionStatic]
    void display(const std::string& message);
         ^
src/handler/CLIHandler.cpp:6:18: note: Technically the member function 'CLIHandler::display' can be
static (but you may consider moving to unnamed namespace).
void CLIHandler::display(const std::string& message) {
                 ^
include/handler/CLIHandler.h:10:10: note: Technically the member function 'CLIHandler::display' can
be static (but you may consider moving to unnamed namespace).
    void display(const std::string& message);
         ^
include/handler/CLIHandler.h:13:10: performance: inconclusive: Technically the member function
'CLIHandler::errDisplay' can be static (but you may consider moving to unnamed namespace).
[functionStatic]
    void errDisplay(const ErrorInfo& error);
         ^
src/handler/CLIHandler.cpp:11:18: note: Technically the member function 'CLIHandler::errDisplay' can
be static (but you may consider moving to unnamed namespace).
void CLIHandler::errDisplay(const ErrorInfo& error) {
                 ^
include/handler/CLIHandler.h:13:10: note: Technically the member function 'CLIHandler::errDisplay'
can be static (but you may consider moving to unnamed namespace).
    void errDisplay(const ErrorInfo& error);
         ^
src/controller/RVCPowerController.cpp:15:0: style: The function 'isActive' is never used.
[unusedFunction]
bool RVCPowerController::isActive() const {
^
src/handler/ErrorHandler.cpp:7:0: style: The function 'error' is never used. [unusedFunction]
void ErrorHandler::error(ErrorType type) {
^
nofile:0:0: information: Unmatched suppression: missingInclude [unmatchedSuppression]

nofile:0:0: information: Active checkers: 172/592 (use --checkers-report=<filename> to see details)
[checkersReport]
```

Static Code Analysis

(2) SonarCloud



Vibe Coding

Vibe Coding 작업환경 설정

- 원본 코드 참조 및 국소 부분 수정으로 스킬 조정

- /change-impact
  - 오른쪽 센서 삭제 영향 범위 확인.
- /srs
  - srs 작성 스킬.
  - legacy 인계 후 legacy srs 수정.
- /sdd
  - sdd 작성 스킬.
  - legacy 인계 후 legacy sdd 수정.
- /code
  - 코드 작성 스킬.
  - legacy 인계 후 legacy code 수정.
- /ut
  - unit test 작성 스킬.
  - legacy 인계 후 unit test 수정.
- /simulator
  - 시뮬레이터 작성 스킬
  - legacy 인계 후 시뮬레이터 수정
- /st
  - system test 작성 스킬.
  - legacy 인계 후 system test 수정.
- /sa
  - 정적 분석 스킬.
- /package
  - 문서 최종화 및 클린 빌드.

FR

1.1 시스템 생명주기 제어 (FR-CTRL)

FR-CTRL-01 – 시스템 기동
사용자가 CLI에서 전원 켜기 명령을 입력하면, 시스템은 5초 이내에 내부 구성 요소를 초기화하고 준비 완료를 표준 출력에 알린 뒤 자동 청소 루프(FR-CLEAN-01)로 진입해야 한다.

검증 기준: 전원 ON 명령 수신 후 5초 이내에 자동 청소 루프 첫 반복이 시작됨을 확인한다.

FR-CTRL-02 – 정상 종료
사용자가 CLI에서 전원 끄기 명령을 입력하면, 시스템은 진행 중인 모든 이동·청소 동작을 중지하고, 구동 모터와 청소 장치를 안전하게 정지한 뒤 종료 메시지를 출력하고 오프라인 상태로 전환해야 한다.

FR-CTRL-03 – 오류·예외 종료
시스템 내 어느 구성 요소에서든 복구 불가능한 예외 또는 오류가 감지되면, 시스템은 오류 내용을 표준 출력에 즉시 알리고 FR-CTRL-02와 동일한 정지 절차를 수행한 뒤 오프라인 상태로 전환해야 한다.

1.2 이동·방향 제어 (FR-MOVE)

FR-MOVE-01 – 전진 이동
자동 청소 루프가 활성화되면, 시스템은 구동 모터를 통해 로봇을 전방으로 이동시켜야 한다. 이동은 장애물 감지, 종료 명령, 또는 오류가 발생하기 전까지 지속된다.

FR-MOVE-02 – 장애물 회피 (3-경로 결정)
전방 장애물이 감지되면 시스템은 전진 이동을 중지하고 좌·우측 센서 상태를 조회하여 아래 결정 표에 따라 이동 방향을 전환해야 한다.

FR-MOVE-03 – 회전 중 입력 마스크
회전 동작(좌회전 또는 우회전)이 진행되는 동안에는 전방·좌·우 근접 센서 및 먼지 센서로부터 유입되는 모든 신호를 무시하고 회전을 완수해야 한다. 회전 완료 후 정상 센서 폴링을 재개한다.

근거: 회전 중 센서 반응 시 무한 방향 재결정 루프가 발생할 수 있으므로 회전 완료를 원자 동작으로 처리한다.

1.3 청소 제어 (FR-CLEAN)

FR-CLEAN-01 – 표준 청소
전진 이동이 활성화된 상태에서 바닥 브러시와 물걸레 장치를 동시에 가동하여 청소를 수행해야 한다. 이동이 정지되면 청소 장치도 함께 정지한다.

FR-CLEAN-02 – 강화 청소 모드
먼지 센서에서 먼지가 감지되면 시스템은 100ms 이내에 청소 장치를 강화 모드(출력 증가)로 전환하고 5초 타이머를 시작해야 한다. 타이머 만료 시 자동으로 표준 모드로 복귀한다.

1.4 센서 입력 처리 (FR-SENSE)

FR-SENSE-01 – 장애물 감지
시스템은 청소 루프 반복마다 전방 근접 센서를 폴링하여 장애물 [illegible] 한다. 전방 장애물 감지 시 FR-MOVE-02를 즉시 트리거한다. 장애[illegible] 먼지 감지 처리보다 항상 우선한다.

FR-SENSE-02 – 먼지 감지
시스템은 청소 루프 반복마다 먼지 센서를 폴링하여 먼지 여부를 [illegible] 먼지 감지 시 FR-CLEAN-02를 트리거하되, 현재 FR-MOVE-02(장애[illegible] 중이면 먼지 이벤트는 보류하지 않고 폐기한다.

FR-MOVE-02 – 장애물 회피 (2-경로 결정)

항목	내용
ID	FR-MOVE-02
요구사항	전방 장애물이 감지되면 시스템은 전진 이동을 즉시 중지하고, 좌측 센서 상태를 조회하여 아래 결정 표에 따라 이동 방향을 전환해야 한다. 방향 전환 완료 후 전진 이동을 재개한다.
사전 조건	M-CLEAN 상태에서 전방 센서 감지 신호 수신
사후 조건	방향 전환 완료 후 M-CLEAN 복귀, 전진 이동 재개
우선순위	필수

방향 결정 표:

전방	좌측	실행 동작
막힘	비어 있음	좌회전 → 전진 재개
막힘	막힘	후진(일정 거리) → 좌측 재조회 → 좌회전(좌측 비면) 또는 우회전[※](좌측 막힘이면) → 전진 재개

> [※] 후진 후 좌측이 여전히 막혀 있을 때의 우회전은 **센서 판독 없이 수행되는 최후 탈출(blind last-resort)** 이다. 우측 근접 센서가 제거되었으므로 우측 공간을 사전 조회하지 않으며, 후진으로 확보된 여유를 이용한 탈출 시도로만 의미를 갖는다.

측정 기준:

구간	제한 시간	NFR 참조
전방 감지 → 전진 이동 완전 정지	≤ 50ms	NFR-TIMING-02
좌회전 결정 → 모터 명령 발생	≤ 500ms	NFR-TIMING-03
전·좌 모두 막힘 → 후진 시작	≤ 100ms	NFR-TIMING-04a
후진 완료 → 회전 방향 결정	≤ 500ms	NFR-TIMING-04b
회전 방향 결정 → 모터 명령 발생	≤ 500ms	NFR-TIMING-04c

NFR

2.1 아키텍처 품질 (NFR-ARCH)

NFR-ARCH-01 – 하드웨어 추상화
구동 모터, 청소 장치, 각 센서(전방·좌·우·먼지) 는 순수 추상 C++ 클래스(인터페이스)로 정의되어야 한다. 애플리케이션 로직은 이 인터페이스만 참조하며 구체 구현에 직접 의존하지 않는다.

NFR-ARCH-02 – Stub 기반 단위 테스트 가능성
물리 하드웨어 없이도 모든 핵심 로직의 단위 테스트를 실행할 수 있어야 한다. 이를 위해 NFR-ARCH-01의 각 인터페이스에 대응하는 Stub 구현을 제공해야 한다.

NFR-ARCH-03 – 예외 처리 및 안전 종료 보장
시스템은 어느 계층에서 예외가 발생하더라도 FR-CTRL-03의 안전 종료 흐름이 실행됨을 보장하는 예외 처리 구조를 가져야 한다. 예외를 전파하지 않고 소멸시키는 코드는 허용하지 않는다.

2.2 응답 시간 (NFR-TIMING)

ID	이벤트	제한 시간	측정 기준
NFR-TIMING-01	전원 ON → 자동 청소 루프 첫 반복 시작	≤ 5,000ms	FR-CTRL-01 완료 시점
NFR-TIMING-02	전방 센서 장애물 감지 → 전진 이동 완전 정지	≤ 50ms	FR-MOVE-02 진입 시점
NFR-TIMING-03	장애물 감지 후 방향 결정 → 모터 명령 발생 (좌/우 회전)	≤ [illegible]	[illegible]
NFR-TIMING-04a	전·좌·우 모두 막힘 → 후진 시작	≤ 100ms	FR-MOVE-02
NFR-TIMING-04b	후진 완료 → 회전 방향 결정	≤ 500ms	후진 정지 시점 기준
NFR-TIMING-04c	회전 방향 결정 → 모터 명령 발생	≤ 500ms	결정 시점 기준
NFR-TIMING-05	먼지 감지 → 강화 모드 활성화	≤ 100ms	FR-CLEAN-02 진입 시점
NFR-TIMING-06	강화 모드 지속 시간	5,000ms	타이머 시작 기준

2.3 빌드 환경 (NFR-BUILD)

NFR-BUILD-01 – 구현 언어·빌드 시스템
- 구현 언어: C++17 표준 (`-std=c++17`)
- 빌드 시스템: CMake ≥ 3.14
- 컴파일러: GCC/G++ on Ubuntu (WSL 포함)
- 테스트 프레임워크: GoogleTest
- 로컬 빌드와 CI 빌드 환경에서 동일한 CMake 버전으로 빌드가 성공해야 한다.

2.4 사용자 인터페이스 (NFR-UI)

NFR-UI-01 – 실시간 상태 로그
시스템의 모드 전환(전진 시작, 회전, 후진, 강화 모드 활성/해제, 종료 등)은 발생 즉시 표준 출력에 텍스트 로그로 기록되어야 한다.

NFR-UI-02 – CLI 전원 제어
사용자는 터미널 CLI를 통해 전원 ON/OFF 명령을 입력할 수 있어야 한다.

2.5 동작 안전 규칙 (NFR-SAFETY)

NFR-SAFETY-01 – 이벤트 처리 우선순위
장애물 감지 이벤트는 먼지 감지 이벤트보다 항상 높은 우선순위를 갖는다. 두 이벤트가 같은 루프 반복에서 발생하면 장애물 처리가 먼저 실행되며, 해당 반복에서 먼지 이벤트는 폐기된다.

4.2 응답 시간 (NFR-TIMING)

ID	이벤트	제한 시간	측정 기준	관련 FR
NFR-TIMING-01	전원 ON → 자동 청소 루프 첫 반복 시작	≤ 5,000ms	FR-CTRL-01 완료 시점	FR-CTRL-01
NFR-TIMING-02	전방 장애물 감지 → 전진 이동 완전 정지	≤ 50ms	FR-MOVE-02 진입 시점	[illegible]
NFR-TIMING-03	장애물 감지 후 좌/우 회전 결정 → 모터 명령 발생	≤ 500ms	결정 시점 기준	FR-MOVE-02
NFR-TIMING-04a	전·좌·우 모두 막힘 → 후진 시작	≤ 100ms	FR-MOVE-02 후진 경로 진입	FR-MOVE-02
NFR-TIMING-04b	후진 완료 → 회전 방향 결정	≤ 500ms	후진 정지 시점 기준	FR-MOVE-02
NFR-TIMING-04c	회전 방향 결정 → 모터 명령 발생	≤ 500ms	결정 시점 기준	FR-MOVE-02
NFR-TIMING-05	먼지 감지 → 강화 모드 활성화	≤ 100ms	FR-CLEAN-02 진입 시점	FR-CLEAN-02
NFR-TIMING-06	강화 모드 지속 시간	5,000ms (±허용 오차)	타이머 시작 기준	FR-CLEAN-02

UC

UC-01: 시스템 기동

항목	내용
트리거	사용자가 CLI에서 전원 켜기 명령 입력
사전 상태	시스템이 오프라인(M-IDLE) 상태
주요 흐름	1. 시스템이 내부 구성 요소를 순차 초기화한다. ② 초기화 완료 후 준비 완료 메시지를 터미널에 출력한다. ③ UC-02 자동 청소로 자동 전이한다.
분기 / 예외	초기화 중 오류 발생 → UC-06 오류 종료로 전이
검증 시그널	전원 ON 명령 수신 후 5초 이내에 "준비 완료" 로그 출력 + UC-02 루프 진입

관련 FR: FR-CTRL-01, NFR-TIMING-01

UC-02: 자동 청소

항목	내용
트리거	UC-01 완료 후 자동 전이; 또는 UC-03 장애물 회피 완료 후 복귀
사전 상태	시스템이 청소 비활성 상태 (M-INIT 또는 M-AVOID 완료)
주요 흐름	① 청소 장치를 표준 강도로 활성화한다. ② 전진 이동을 시작한다. ③ [루프] 전방 센서 폴링 → 먼지 센서 폴링 → 반복.
분기 / 예외	전방 장애물 감지 → UC-03 전이 / 먼지 감지 → UC-04 전이 (단, 장애물 이벤트 우선) / 전원 OFF → UC-05 / 오류 → UC-06
검증 시그널	"전진 이동 시작" 로그 + 청소 장치 활성 상태 확인

관련 FR: FR-MOVE-01, FR-CLEAN-01, FR-SENSE-01, FR-SENSE-02, NFR-SAFETY-01

UC-03: 장애물 회피

항목	내용
트리거	전방 센서에서 장애물 감지 신호 수신 (UC-02 루프 중)
사전 상태	전진 이동 중 (M-CLEAN)
주요 흐름	① 전진 이동 및 청소 장치를 즉시 정지한다. ② 좌·우 센서를 조회하여 방향을 결정한다. ③ 결정된 동작(회전 또는 후진+회전)을 실행한다. ④ UC-02로 복귀한다.

UC-04: 강화 청소

항목	내용
트리거	먼지 센서에서 먼지 감지 신호 수신 (자동 청소 중; 장애물 회피 중이 아닐 것)
사전 상태	M-CLEAN (자동 청소 중), 강화 모드 비활성
주요 흐름	① 청소 장치를 강화 강도로 전환한다. ② 5초 타이머를 시작한다. ③ 타이머 만료 시 표준 강도로 복귀한다.
분기 / 예외	강화 모드 중 먼지 재감지 → 무시 (타이머 재시작 없음) / 강화 모드 중 장애물 감지 → UC-03 수행 후 강화 모드 유지하여 복귀 / 오류 → UC-06
검증 시그널	"강화 청소 활성" 로그 출력 후 5초 경과 시 "표준 청소 복귀" 로그 출력

관련 FR: FR-CLEAN-02, FR-SENSE-02, NFR-TIMING-05, NFR-TIMING-06

UC-05: 정상 종료

항목	내용
트리거	사용자가 CLI에서 전원 끄기 명령 입력
사전 상태	시스템이 활성 상태 (M-CLEAN, M-AVOID, M-BOOST 중 어느 하나)
주요 흐름	① 현재 진행 중인 이동·청소 동작을 중지한다. ② 구동 모터와 청소 장치를 정지한다. ③ "종료 중" 메시지를 출력한다. ④ 오프라인 상태(M-IDLE)로 전환한다.
분기 / 예외	종료 처리 중 오류 → UC-06 전이
검증 시그널	"시스템 종료" 로그 출력 + 모든 구 [illegible] 정지 상태 [illegible]

관련 FR: FR-CTRL-02

UC-06: 오류 종료

항목	내용
트리거	시스템 내부의 어느 단계에서든 복구 불가 예외/오류 발생 (UC-01~UC-05에서 전이 가능)
사전 상태	시스템 활성 상태 임의
주요 흐름	① 오류 내용을 터미널에 즉시 출력한다. ② 이동·청소 동작을 중지한다. ③ 구동 모터와 청소 장치를 정지한다. ④ 오프라인 상태로 전환한다.
분기 / 예외	없음 (최종 폴백 – 추가 예외 처리 없음)
검증 시그널	오류 메시지 로그 출력 + 모든 구동장치 정지 확인

관련 FR: FR-CTRL-03

UC-03: 장애물 회피

항목	내용
UC ID	UC-03
UC 명	장애물 회피
주 액터	전방·좌측 근접 센서
목적	전방 장애물을 피해 방향을 전환하고 청소 루프로 복귀
사전 조건	M-CLEAN 상태에서 전방 센서 감지 신호 수신
주요 흐름	① 전진 이동 및 청소 장치를 즉시 정지한다. ② 좌측 센서를 조회하여 방향을 결정한다(아래 결정 표 참조). ③ 결정된 동작(좌회전 또는 후진+회전)을 실행한다. ④ UC-02 자동 청소로 복귀한다.
대안·예외	회전 실행 중: 모든 센서 입력(장애물·먼지)을 무시하고 회전 완수 / 오류 → UC-06 전이
사후 조건	방향 전환 완료; M-CLEAN 복귀
검증 시그널	방향 전환 후 UC-02 재진입 확인, 해당 방향 로그 출력 확인
관련 FR	FR-MOVE-02, FR-MOVE-03, FR-SENSE-01, NFR-TIMING-02, NFR-TIMING-03, NFR-TIMING-04a~04c

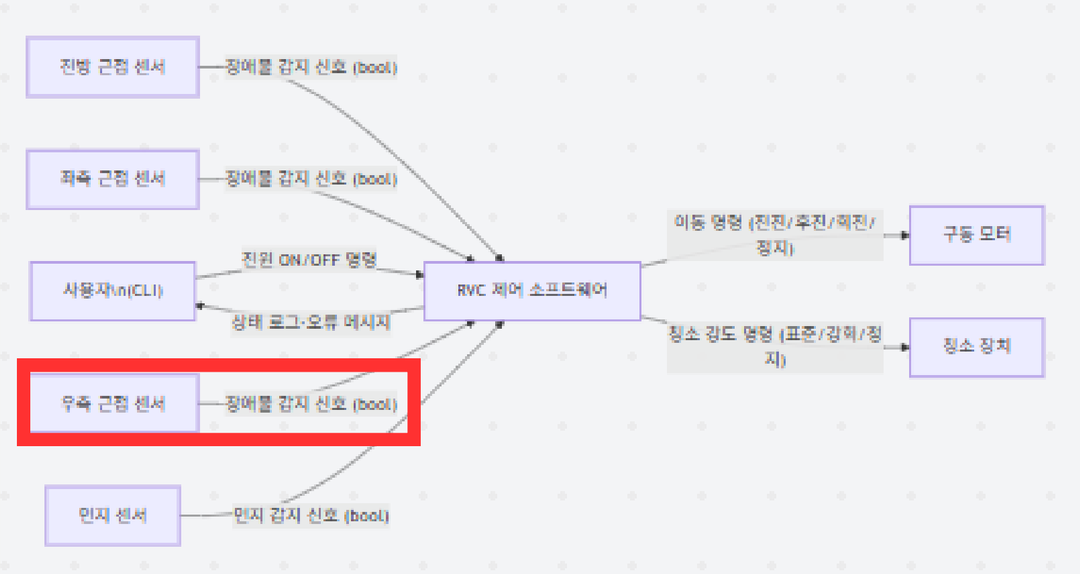
방향 결정 표:

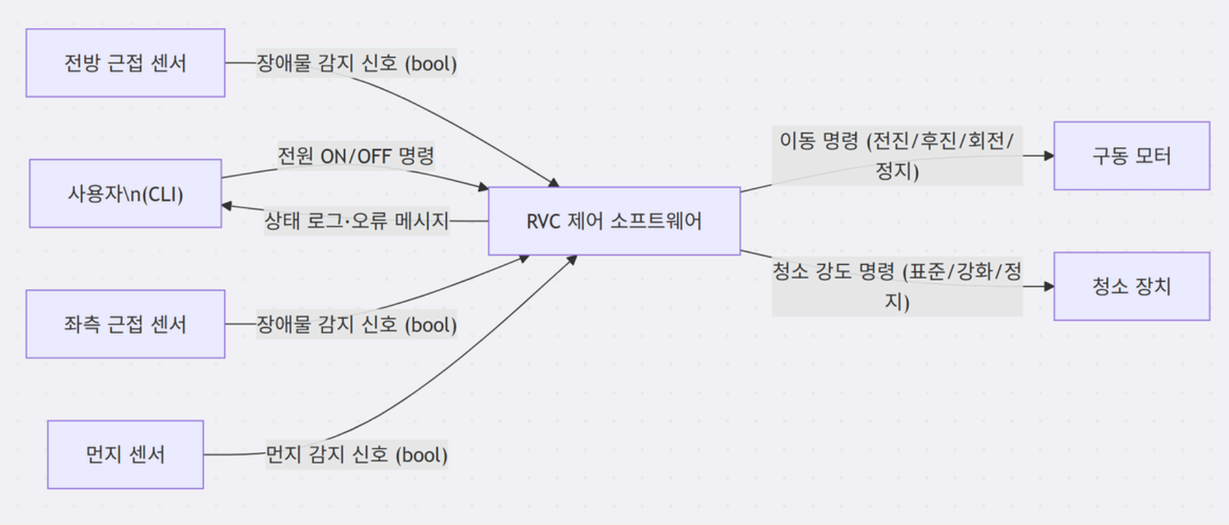
전방	좌측	실행 동작	타이밍 기준
막힘	비어 있음	좌회전	감지→정지 ≤50ms; 결정→모터 ≤500ms
막힘	막힘	후진 → 좌측 재조회 → 좌회전(좌측 비면) 또는 우회전[※](좌측 막힘이면)	정지→후진 ≤100ms; 후진 후 결정 ≤500ms; 결정→모터 ≤500ms

[※] FR-MOVE-02의 주석과 동일: 후진 후 우회전은 우측 센서 판독 없이 수행되는 최후 탈출이다.

SRS

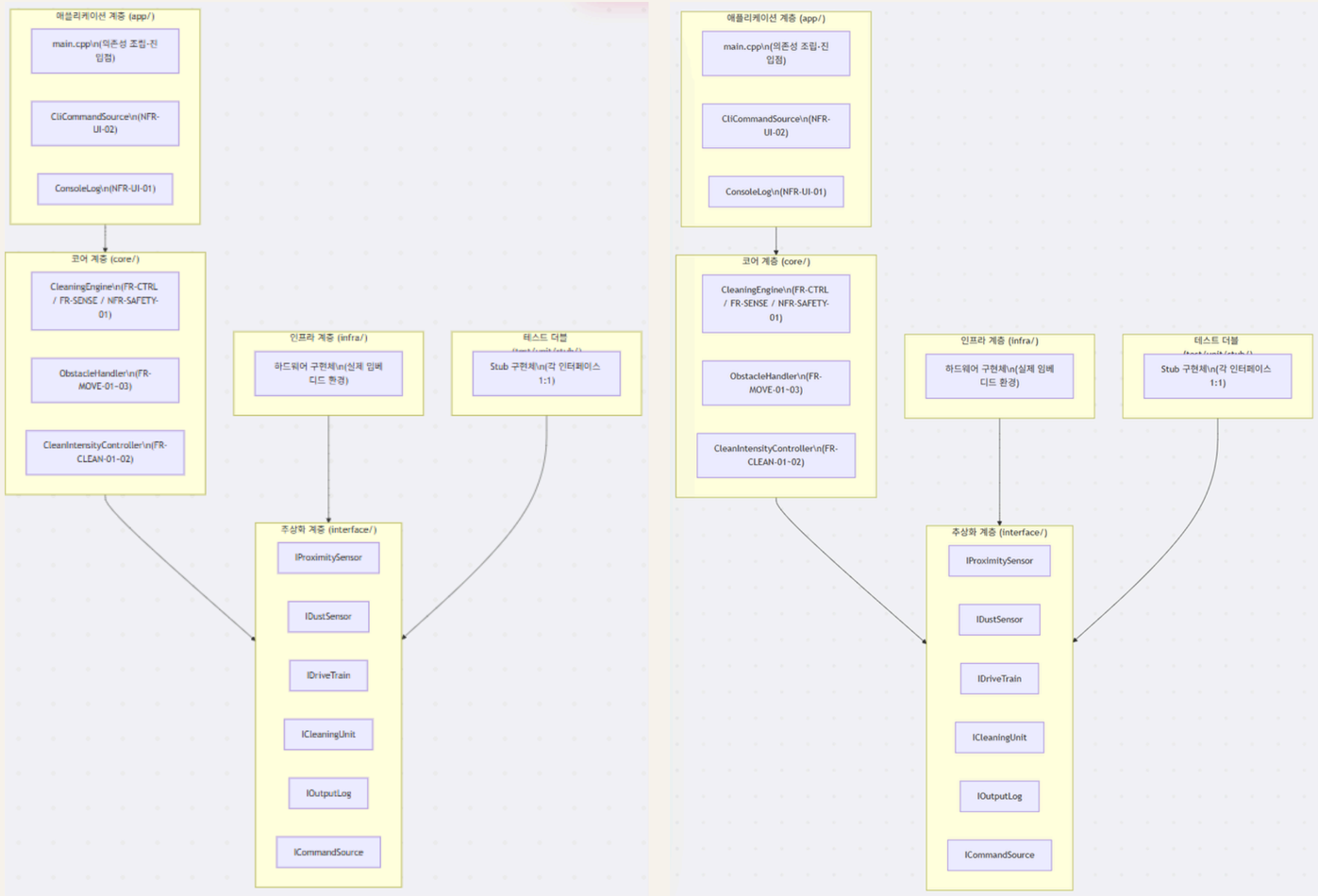
- SRS.md
  - 시스템 컨텍스트





SDD

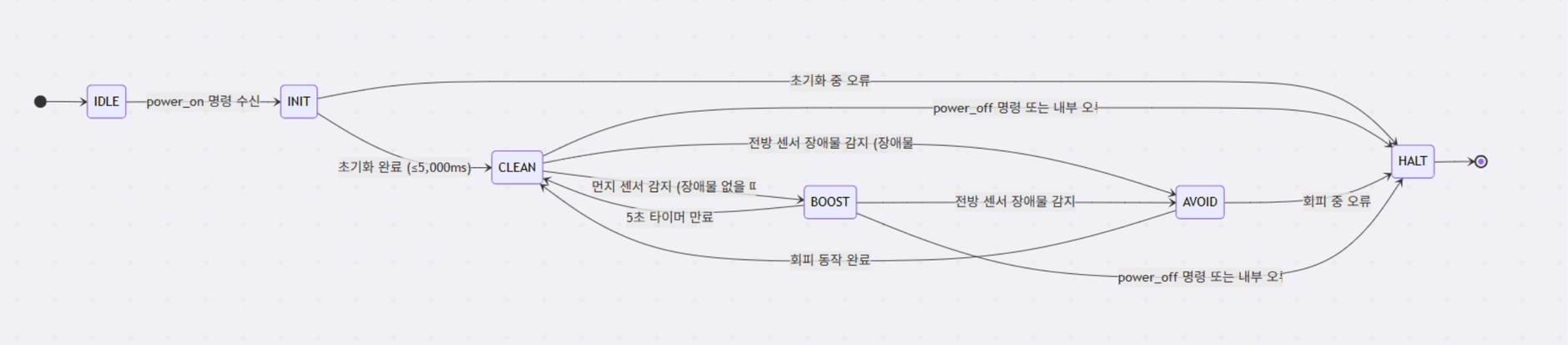
- 계층 구조 다이어그램

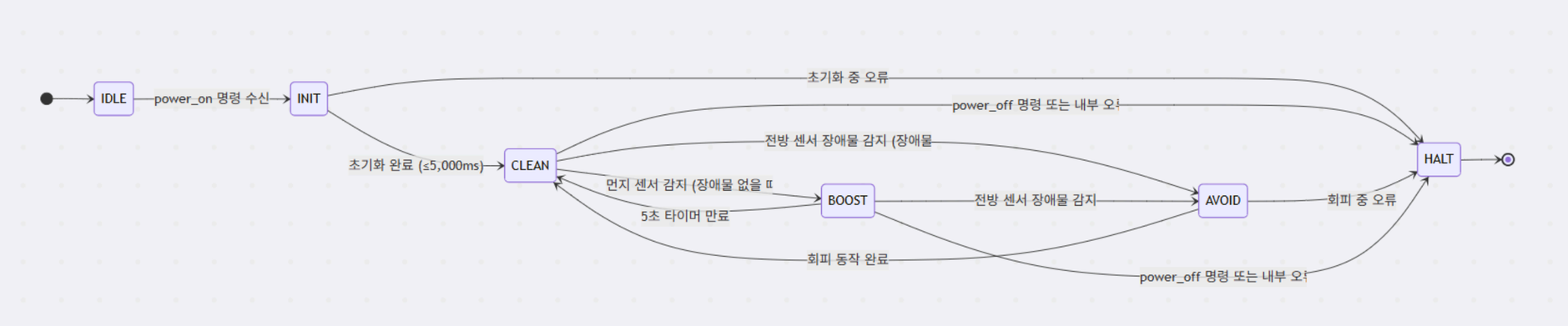


- 변화 없음.

SDD

- 상태 전이 다이어그램

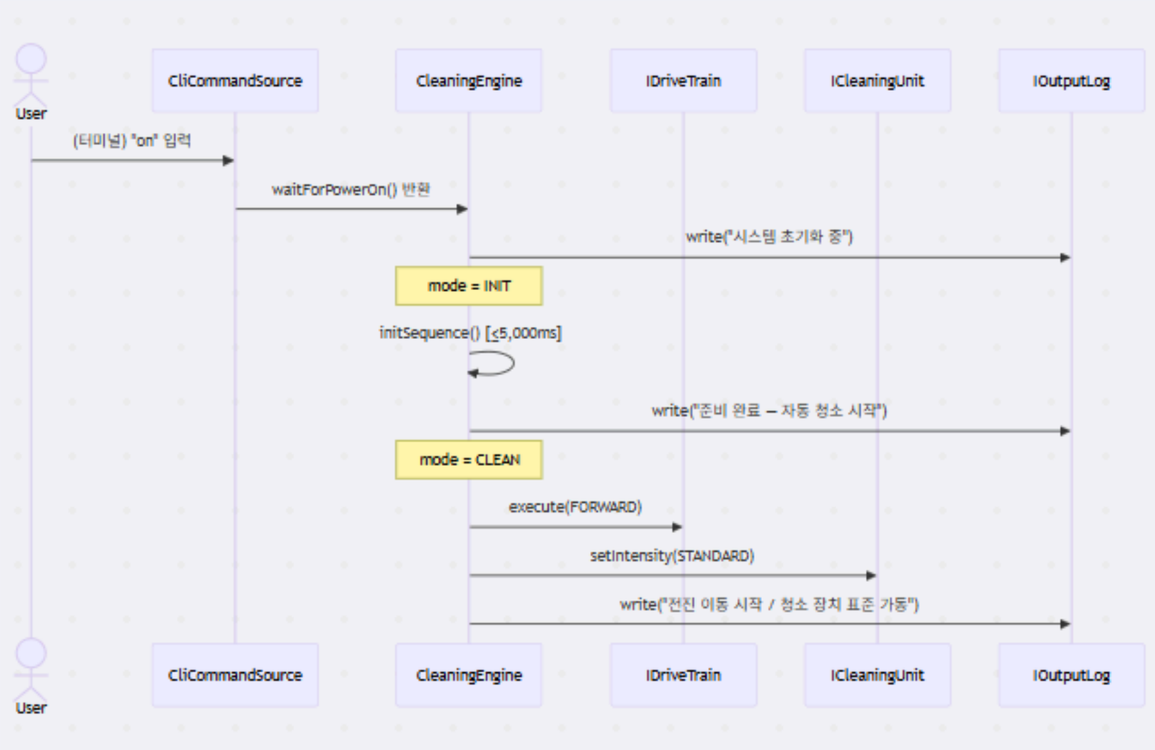


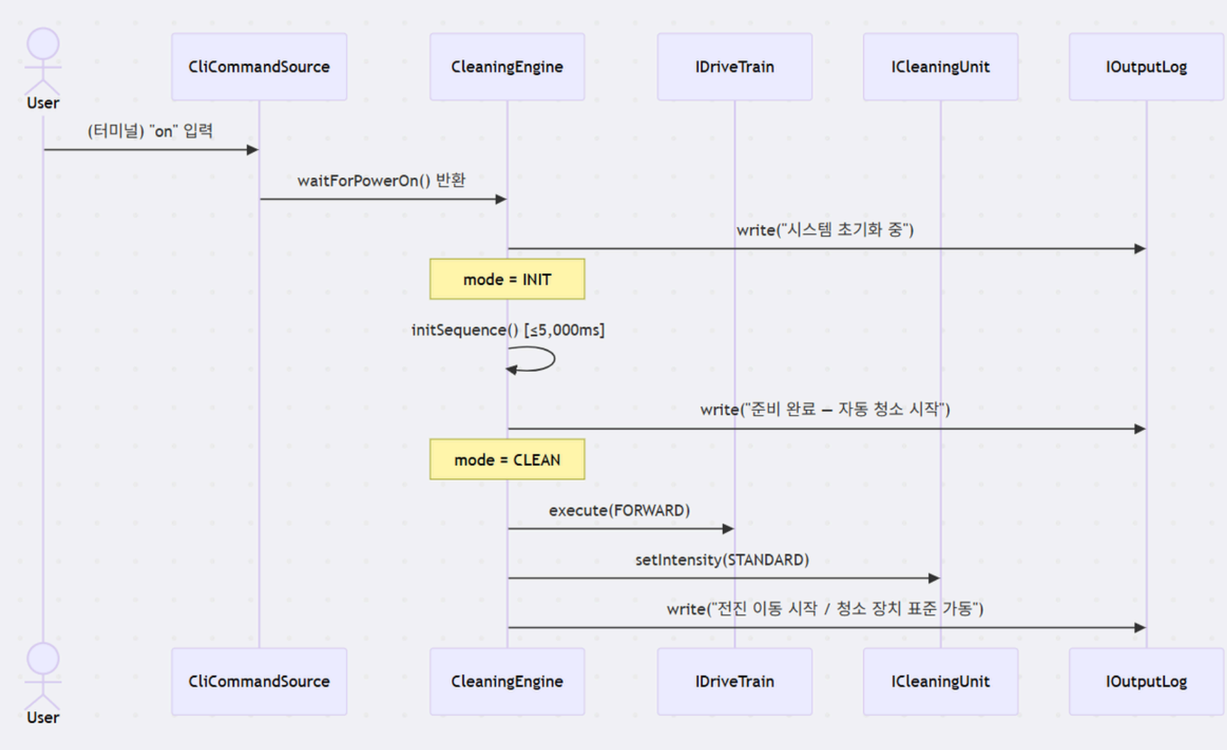


- 변화 없음.

SDD

- UC-01: 시스템 가동

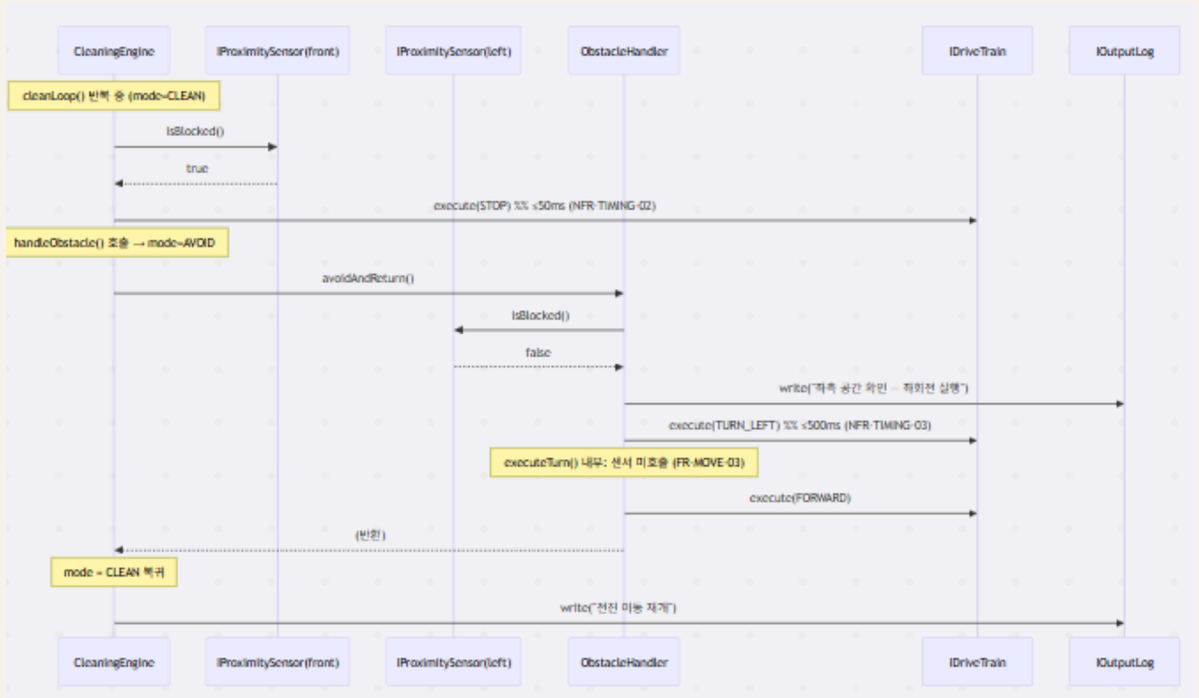




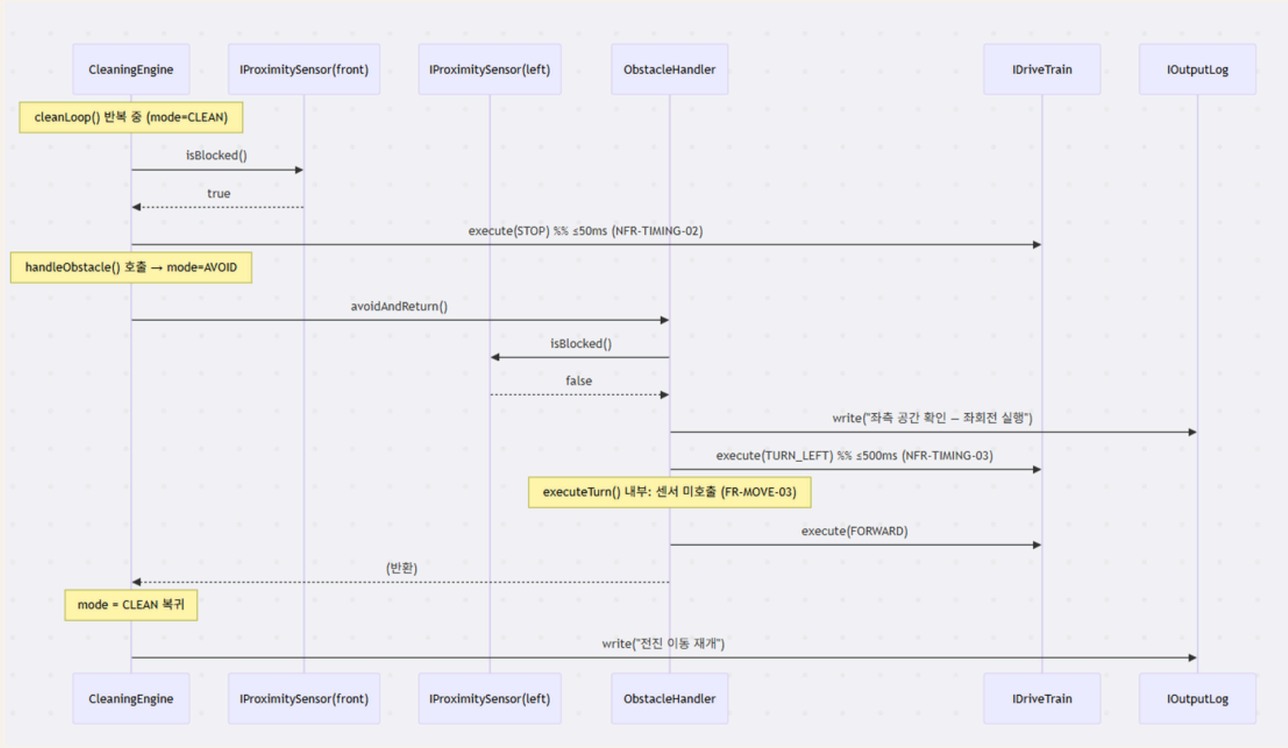
- 변화 없음.

SDD

- UC-03: 장애물 회피 – 좌회전 경로

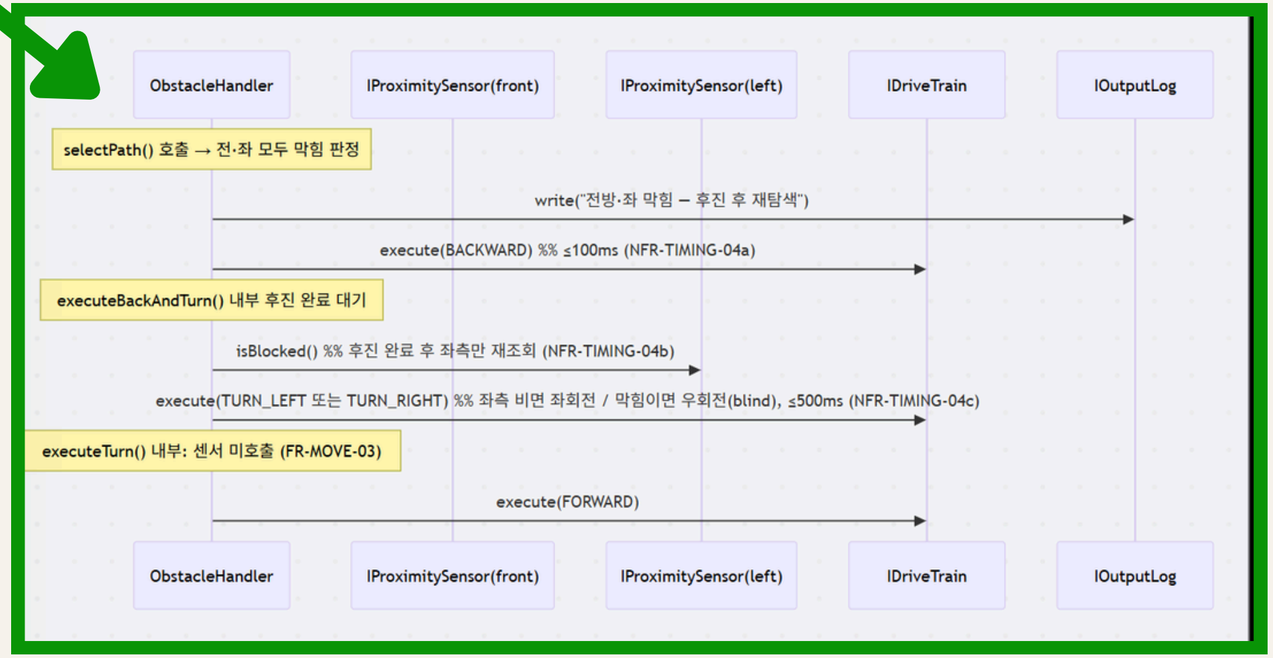


- 변화 없음.



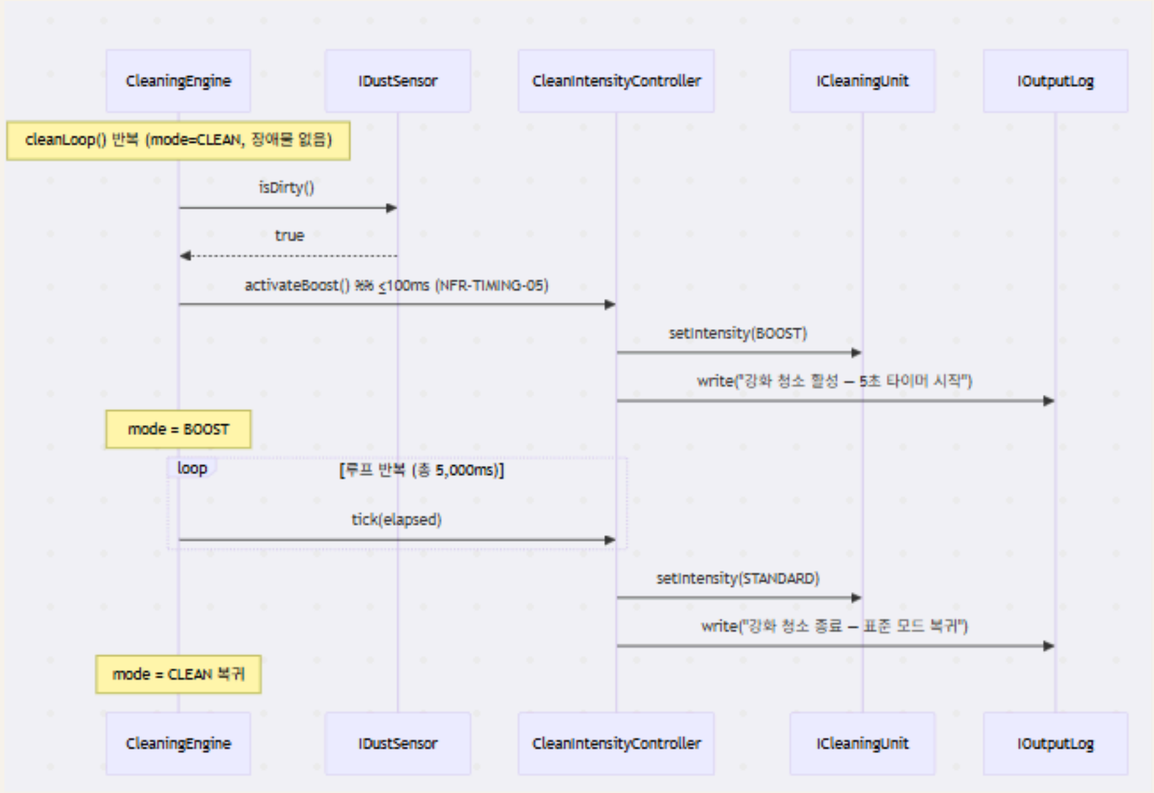
SDD

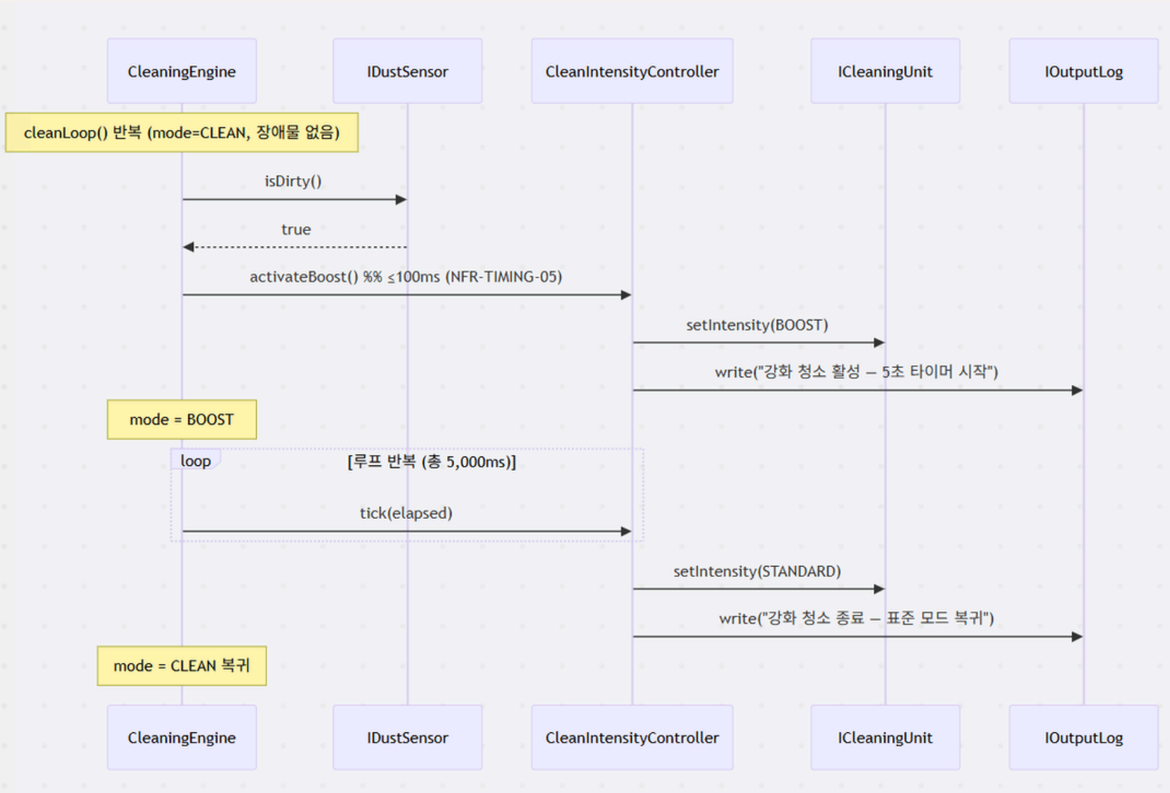
- UC-03: 장애물 회피 – 후진+재조회 경로



SDD

- UC-04: 강화 청소

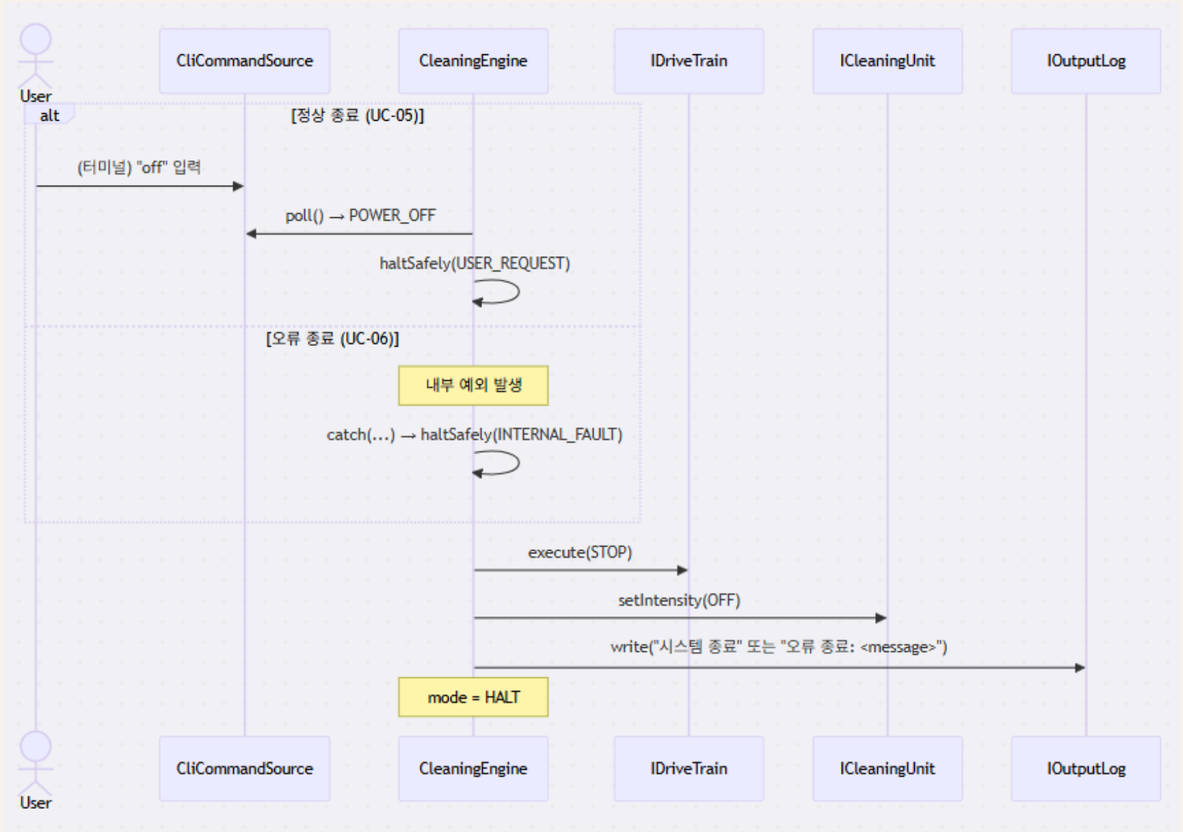


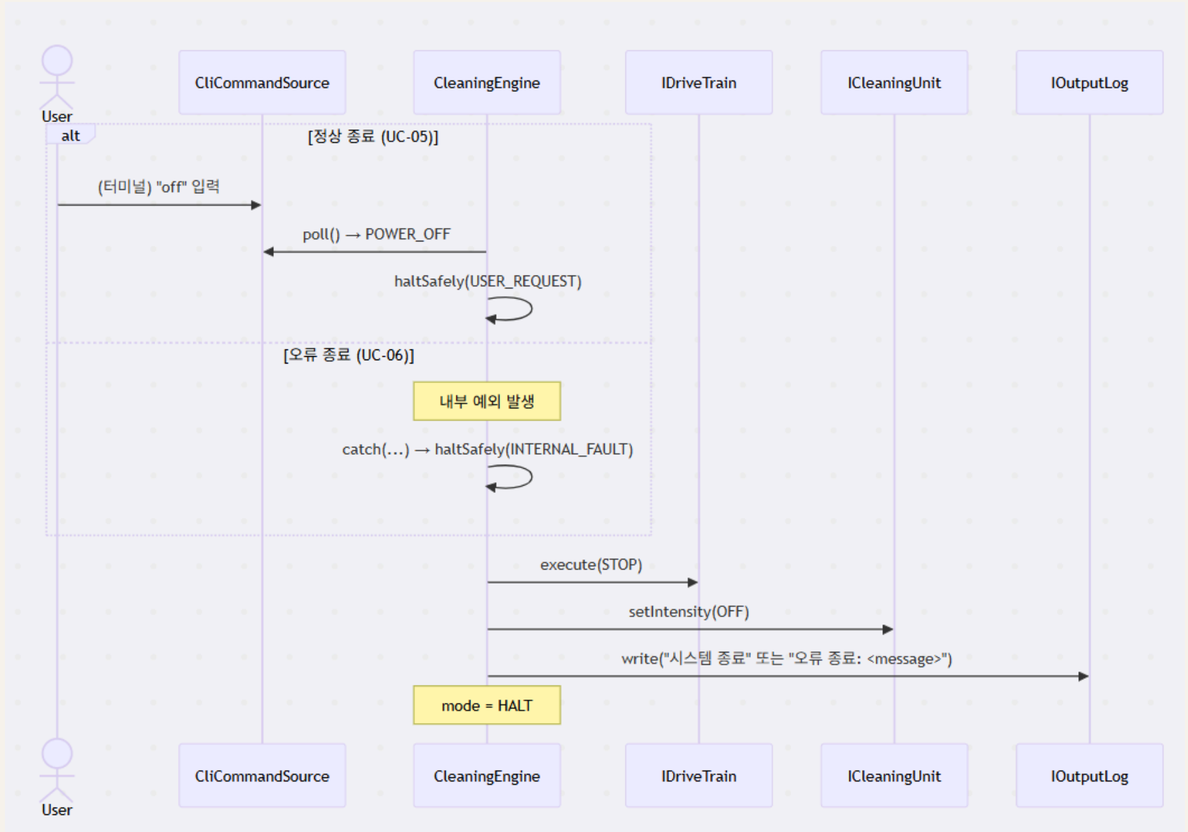


- 변화 없음.

SDD

- UC-05/UC-06: 정상 오류 종료





- 변화 없음.

Unit Test Log

주요 결정
- **legacy `test/unit/`을 그대로 인계 후 우측 센서·우측 분기 관련 케이스만 수정.** 테스트 파일명·구조는 수정 구현(=legacy 구조)을 따른다.
- **`test_obstacle_handler.cpp`**:
  - 픽스처 `StubProximitySensor front, left, right;` → `front, left;`, `handler{left, right, …}` → `handler{left, …}`.
  - **`TurnRightWhenLeftBlockedRightClear` 테스트 삭제** — "좌 막힘·우 비면 우회전"은 우측 센서 판독 경로(`AvoidPath::TURN_RIGHT`)였고 폐지됨. 삭제 사유를 주석으로 명시(추적성).
  - `BackwardThenTurnWhenAllBlocked` → `BackwardThenTurnWhenLeftBlocked`로 개명·조건 단순화 (`right.setBlocked` 제거). 후진 후 좌 막힘 → `TURN_RIGHT`(blind) 기대값은 **유지**(REVIEW 권장안: executeBackAndTurn 존속).
  - `BackwardThenLeftTurnIfLeftClearsAfterReverse`: `right.enqueueResult(true)` 제거 (selectPath는 이제 좌만 검사). 좌 enqueue(true→false)로 후진 후 좌 클리어 시 좌회전 검증.
  - `NoSensorPollingDuringRotation`: `right` 리셋·`right.callCount()==0` 단언 제거, 주석을 "좌만 폴링"으로. `front.callCount()==0`·`left.callCount()==1` 유지.
  - `CommandOrder_BackPath_*`: `right.setBlocked(true)` 제거. BACKWARD→…→TURN_RIGHT(blind)→FORWARD 순서 검증 유지.
- **`test_cleaning_engine.cpp`**: 픽스처 `front, left, right;`→`front, left;`, `makeEngine()`·`ExceptionCausesSafeHalt`의 `CleaningEngine(…, right, …)`에서 `right` 인자 제거. 센서 동작 케이스는 front/left/dust 기반이라 로직 변경 없음.
- **스텁·`test_clean_intensity_controller.cpp`는 일치**: `stub_proximity_sensor.hpp`는 방향 비특정이라 무변경. 나머지 5개 스텁·강도 컨트롤러 테스트는 센서 무관.
- **`TURN_RIGHT` 기대값 유지 근거**: 후진 후 blind 우회전(`MotorCommand::TURN_RIGHT`)은 모터 명령으로 존속하므로 BackPath 테스트의 TURN_RIGHT 단언을 그대로 둔다.

결과 요약
legacy 단위 테스트를 인계받아 우측 센서 픽스처·인자와 우측 분기 검증을 일관 정리했다. 우측 센서 전용 케이스 1개(`TurnRightWhenLeftBlockedRightClear`)를 삭제(사유 주석)하고,
후진-후-blind-우회전 경로 케이스는 기대값 유지. 테스트 수 30→**29**(삭제 1). **29/29 통과 (100%)**, 빌드 경고 0. 스텁 6종·강도 컨트롤러 테스트는 일치.

Simulator Log

주요 결정
- **legacy `test/simulator/`를 그대로 인계 후 `rvc_sim_harness.hpp`에서 우측 센서 배선·주입 API만 제거.** Sim 클래스 이름·하니스 구조·쿼리 API는 legacy 그대로.
- **하니스 변경점**:
  - 엔진 조립 `engine_(front_, left_, right_, dust_, …)` → `engine_(front_, left_, dust_, …)` (수정된 7+1인자 `CleaningEngine` ctor에 정합).
  - 멤버 `SimProximitySensor front_, left_, right_;` → `front_, left_;`.
  - 환경 주입 API에서 우측 경로 제거: `setRightBlocked()`, `injectRightObstacle()` 삭제.
  - 콜카운트 API `rightSensorCalls()` 삭제, `resetSensorCallCounts()`에서 `right_` 항목 제거.
  - 직접 접근자 `rightSensor()` 삭제.
- **`sim_proximity_sensor.hpp`는 일치**: 방향 비특정(`isBlocked()`/`setBlocked()`/`enqueueEvent()`) 클래스라 우측 센서 삭제와 무관 — 클래스 자체 무변경, 하니스의 인스턴스 수(3→2)만 변동.
- **나머지 Sim 5종 일치**: dust/drive/cleaning_unit/output/command 모두 센서 무관.
- **하니스 주석 "six Sim implementations" 유지**: Sim *클래스* 수는 여전히 6종(proximity·dust·drive·cleaning·output·command). 줄어든 것은 proximity 인스턴스 수이므로 문구 일치.
- **빌드 검증 방식**: 현재 `test/system/`이 비어 `oop_system_test` 타겟이 없으므로(다음 `/st`에서 생성), 하니스를 단독 TU로 컴파일+`rvc_core` 링크하여 헤더 정합성을 선검증. 새 8인자 `CleaningEngine` ctor에 정확히 링크됨.

결과 요약
legacy 시뮬레이터를 인계받아 테스트 하니스(`rvc_sim_harness.hpp`)에서 우측 센서 인스턴스·배선·주입/조회 API만 제거했다. Sim 클래스 6종은 모두 일치(방향 비특정 `SimProximitySensor` 포함). 하니스를 단독 컴파일+`rvc_core` 링크로 선검증 — 새 `CleaningEngine` ctor(우측 인자 없음)에 정합하게 링크·실행 성공, 경고 0. 실제 시스템 시나리오 실행은 `/st`에서 수행.

System Test Log

주요 결정

- **기존선 인계**: `legacy/test/system/`의 두 파일(`test_system_scenarios.cpp`, `test_uc_scenarios.cpp`)을 최상위 `test/system/`로 인계받아 우측 근접 센서 삭제가 닿는 부분만 최소 수정. 시나리오를 새로 짜지 않음.
- **회피 시나리오 재구성**: 우측 근접 센서가 사라져 `selectPath()`가 "좌측 가능→좌회전 / 좌측 막힘→후진"으로 단순화됨에 따라(impact.md §5·SRS §3.2 결정 표):
  - legacy의 **우측 경로 시나리오 2건 삭제** — `SystemScenario.ObstacleAvoidanceRightPath`(좌 막힘·우 비면→우회전), `UC03_ObstacleAvoidance.RightPath_WhenLeftBlockedRightClear`. 우측 센서가 없으면 이 경로는 선택될 수 없으므로 시나리오 자체가 성립하지 않음.
  - **후진 시나리오 트리거 단순화** — legacy는 "전·좌·우 모두 막힘"으로 후진을 유발했으나, 이제 "전방 회피 + 좌측 막힘"만으로 후진 경로 진입. `setRightBlocked(true)` 주입 제거, `BackPath_WhenAllBlocked` → `BackPath_WhenLeftBlocked`로 개명.
- **수정 CleaningEngine ctor 정합**: `(front, left, dust, drive, cleaner, log, commands)`로 7-인자화(우측 센서 인자 제거)된 시그니처에 픽스처·오류처리 시나리오의 직접 생성 코드를 맞춤.
- **수정 하니스(RVCSimHarness) API 정합**: `setRightBlocked`/`injectRightObstacle`/`rightSensorCalls` 제거된 API에 맞춰 UC 시나리오의 우측 주입 호출 제거.
- **보존(일치)한 시나리오**: 전원 생명주기(UC-01/UC-02/UC-05), 강화 청소(UC-04), 좌회전 회피(UC-03 Path1), 전방 센서 마스킹(FR-MOVE-03), 명령 시퀀스, 오류 종료(UC-06)의 **검증 의도·기대값**은 legacy 그대로. 우측 센서 인자·주입만 정리. UC 커버리지(UC-01~06) 100% 유지.
- **우회전(모터 명령) 미삭제**: `MotorCommand::TURN_RIGHT`는 모터 출력으로 존속(SRS §6.2). ST에서는 우측 센서 기반 회피 경로 시나리오만 삭제했고, 후진 후 blind 최후 탈출(좌 센서만 사용)은 코드상 유지되므로 별도 신규 시나리오는 추가하지 않음(최소 수정).

결과 요약

- 시스템 테스트 2파일 인계·수정. 우측 근접 센서 기반 회피 경로 시나리오 **2건 삭제**(파일별 1건씩), 후진 시나리오 트리거 단순화, 오류처리·픽스처의 우측 센서 인자 제거.
- 테스트 수: legacy 25건(7+18 등가) → 수정 후 **24건**(System 6 + UC 18). UC-01~UC-06 **전 UC 커버리지 유지**.
- 좌회전/후진/강화청소/전원/오류 등 센서 무관 시나리오는 legacy 검증 의도·기대값 그대로 보존(일치).
- 빌드 성공, **24/24 전부 통과**.

System Test Log

주요 결정

- **방법론을 legacy와 동일하게 유지**: cppcheck·clang-tidy 바이너리가 WSL 환경에 설치되어 있지 않음(legacy 분석 시점과 동일). legacy SA가 GCC `-Wall -Wextra -Wpedantic -Wconversion -Wshadow`(cppcheck 대체) 및 GCC `-fanalyzer`(clang-tidy 대체)를 사용했으므로, **비교의 등가성을 위해 동일한 GCC 명령**을 수정 코드에 재실행. 새 의존성 미도입(CLAUDE.md §3).
- **수정 코드 기준 재분석**: 최상위 `src/`·`include/`(legacy 아님) 6개 .cpp를 대상으로 진단.
- **legacy 대비 등가 비교 보장**: legacy의 -fanalyzer 5건(STL 거짓 양성)이 본 재실행에서 0건이 된 원인을 규명하기 위해, **동일 명령을 legacy/src 에도 재실행** → legacy도 0건임을 확인. 따라서 5→0은 코드 개선이 아니라 toolchain 차이(GCC Bug #107417 거짓 양성의 현재 toolchain 미재현)임을 summary에 명시.
- **억제 판단**: portability POSIX 헤더(`cli_command_source.cpp`)는 WSL/Linux 전용 의도로 legacy 그대로 억제 유지(우측 센서 무관). CWE-457 억제 항목은 현재 미재현으로 불필요해짐.

결과 요약

- 분석 전/후 프로젝트 이슈: **0 → 0** (수정 불요). 수정 코드 6파일 모두 GCC 진단 clean.
- 우측 근접 센서 삭제는 정적 분석 이슈를 **0건 유발**. `ObstacleHandler`의 `right`/`right_`·우측 회피 분기·`AvoidPath::TURN_RIGHT` 케이스를 잔여 없이 제거해 미사용/도달불가 경고 없음.
- legacy 대비 유일한 수치 델타(-fanalyzer 5→0)는 toolchain 차이이며 코드 변경과 무관(legacy 코드도 현재 toolchain에서 0건).

비교분석

작업흐름 비교

- Traditional Coding
  - 변경으로 인해 영향을 미치는 범위를 직접 파악 → 산출물 하나하나 수동 수정
  - SRS → SDD → Code → UT → ST 순서로 각 단계마다 사람이 직접 판단하고 작성
- Vibe Coding
  - 작업환경(스킬 세트)을 메인터넌스용으로 재구성하는 준비 단계가 필요
  - 이후 단계별 실행은 AI가 주도하고 사람은 논리적 오류 검수에 집중
  - 사전 준비만 완료되면 이후 작업 속도가 확연히 빠름

속도 및 검수 품질

- Traditional Coding
  - 작성과 동시에 검수가 이뤄짐
  - 변경 범위가 작아도 모든 산출물을 사람이 일일이 고쳐야 하므로 선형적으로 시간 소요
- Vibe Coding
  - 작성과 검토가 분리됨
  - 스킬 준비 후 단계 실행은 빠르나, 검수를 소홀히 하면 논리 오류가 전 산출물에 전파될 위험이 있음

블랙박스 문제와 문서화

- 바이브 코딩의 맹점: 블랙박스
  - 작업 지시를 내리고 결과물을 받지만, 중간 판단 근거가 기록되지 않으면 "왜 이렇게 됐는지"를 추적할 수 없음
  - 특히 오류 발생 시 원인 파악이 어려움
- 대응 방향
  - 작업환경 설계 시 "기록 파일 생성" 및 "문서화 동시 수행"을 스킬에 내재화해야 함
  - 코드 작성과 코드 명세서(SDD 등) 작성을 동시에 진행하면 블랙박스 상태를 방지할 수 있음
  - 이번 프로젝트의 Unit Test Log, Simulator Log, System Test Log가 그 예시

팀 협업

- Traditional Coding
  - 각자 담당 영역을 나눠 작성하는 방식이 자연스럽게 통용됨
  - 협업 방식이 코드 리뷰, PR 등 업계에서 오래 정립된 구조를 따름
- Vibe Coding
  - 작업 방식(plan mode, 단계별 실행 등)이 팀원마다 다를 수 있음
  - 스킬 세트가 공유되지 않으면 산출물 포맷과 구조가 달라져 상호 호환 불가
  - 결과물 생성 속도가 빠른 만큼 중간에 방향을 바꾸기 어렵고, 초반 방향 정렬이 더 중요

코드레벨 비교 - 구현철학

- Traditional Coding

```
// RVCOrchestrator.cpp
  void RVCOrchestrator::detectObstacle() {
      DirectionType dir = movementPtr->checkMovementPolicy();
      if (dir == DirectionType::LEFT) { turnLeft(); return; }

      // 좌측 막힘 → 180도 회전해서 좌측 센서로 원래 우측 확인
      motorPtr->requestRotate();
      DirectionType dir2 = movementPtr->checkMovementPolicy();
      // 좌측 센서가 우측을 봄
      motorPtr->requestRotate();  // 원래 방향으로 복귀

      if (dir2 == DirectionType::LEFT) { turnRight(); return; }
      backwardAndTurn();
  }
```

  - 우측 센서는 없어졌지만, 우측 방향 확인은 여전히 필요하다 → 우회
    - 우측 정보를 능동적으로 획득하려는 시도. 구조는 복잡해지지만 판단 근거가 명확
  - rotate()로 본체를 180도 돌려 좌측 센서로 우측을 간접 감지

코드레벨 비교 - 구현철학

- Vibe Coding

```
// obstacle_handler.cpp
  void ObstacleHandler::executeBackAndTurn() {
      drive_.execute(MotorCommand::BACKWARD);
      drive_.execute(MotorCommand::STOP);

      // 후진 후 좌측 재확인. 우측은 폴링하지 않음.
      MotorCommand turnCmd = (!left_.isBlocked())
                           ? MotorCommand::TURN_LEFT
                           : MotorCommand::TURN_RIGHT;  // 블라인드 우회전
      executeTurn(turnCmd);
  }
```

  - 우측 센서가 없으면 우측 방향 자체를 알 수 없다 → 단순화
    - 알 수 없는 것은 알려 하지 않는다. 구조는 단순하지만 우측 상황을 모른 채 진입.
  - 우측 확인 경로 자체를 제거, 막히면 블라인드 우회전

코드레벨 비교 - 정답?

관점	Traditional	Vibe-Coding
안정성	우측 상황을 확인하고 판단	단순함 → 예외가 적음
확장성	우측 센서 재도입 시 구조 재활용 가능	재도입 시 구조 전체 재설계 필요
유지보수	체인이 길어 수정 범위 넓음	새로운 매커니즘이 추가된 것이 아님 → 수정 범위 좁고 일관성 높음
요구사항 수행	우측 확인 이라는 동작의 의도를 충분히 파악함	우측 센서 없음이라는 상황을 문자 그대로 받아들임 → “블라인드로 돌아도 되는가"는 요구사항 해석의 문제

느낀점

1. 바이브 코딩은 유지보수성 있는 작업환경 설계가 전제되어야 한다. 스킬을 한 번 잘 만들어두면 이후 반복 변경에서 강력하지만, 사전준비 없이는 오히려 통제하기 어렵다.
2. 기록과 문서화를 작업환경에 내재화해야 한다. 블랙박스를 방지하는 것은 결국 사람이 설계하는 작업환경의 몫이다.
3. 팀 단위 바이브 코딩은 작업방식 통일이 선행되어야 한다. 개인 단위에서는 유연하게 쓸 수 있지만, 팀 단위에서는 오히려 초기 합의 비용이 더 들 수 있다.
4. 두 방식은 대체 관계가 아니라 보완 관계다. 요구사항 분석이나 설계 검수처럼 판단이 필요한 영역에서는 전통 방식의 사고 과정이 여전히 핵심이다.