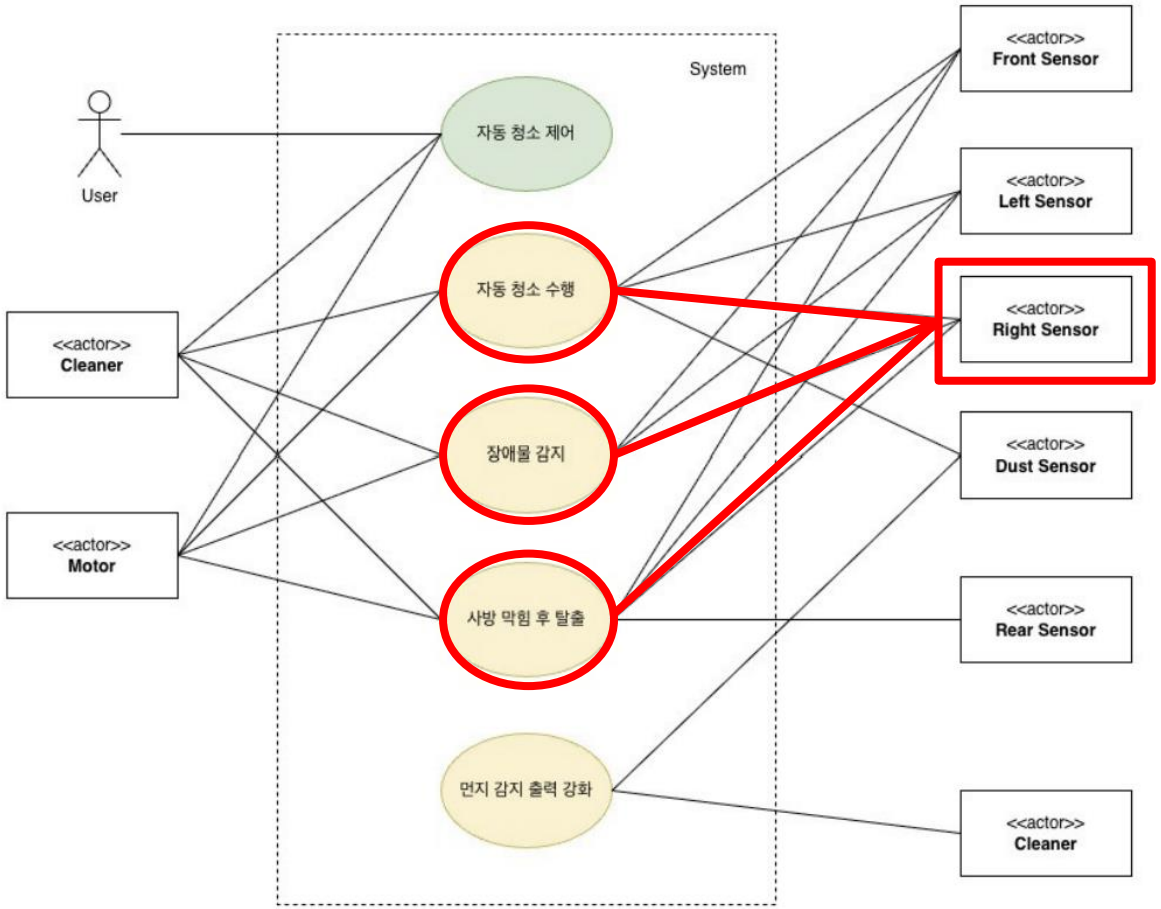
Robot Vacuum Cleaner

RightSensor 제거 변경 발표

3팀
202213351 김태성
202111382 최성준
202011434 최원탁
202011380 최용근

Use Case Diagram



CHANGE SUMMARY

RightSensor 제거와 우측 판단 재설계

CHANGE SCOPE

21개 파일 변경 · 403 insertions · 351 deletions

핵심 변경

1. RightSensor 클래스와 rightBlocked 상태 제거
2. 장애물 감지는 Front / Left / Rear 3개 센서로 진행
3. 우측 개방 여부는 우회전 후 FrontSensor 재확인으로 추론
4. UC-03 회피 실패 시 UC-04 코너 탈출 흐름으로 연결

설계 원칙

센서를 추가하지 않고 기존 전방 센서를 재사용
회전-확인-복구 순서를 Controller 책임으로 정리

RESULT

모델 단순화
회피 흐름 명시화

USE CASE · UC-03

우측 센서 없는 회피 방향 선택

PRECONDITION

UC-02에서 전달받은 obstacleState가 FRONT_LEFT_BLOCKED

MAIN SCENARIO

1. 최초 obstacleState를 재폴링하지 않고 그대로 사용한다.
2. FRONT_BLOCKED는 정지 후 좌회전, 전진, 청소 재개로 처리한다.
3. FRONT_LEFT_BLOCKED는 우회전 후 센서를 다시 폴링한다.
4. 우회전 후 전방이 열려 있으면 우측 회피 가능으로 판단한다.
5. 우회전 후 전방도 막혀 있으면 좌회전으로 복구하고 UC-04를 시작한다.

ALTERNATIVE

우측 확인은 RightSensor 호출이 아니라 Motor 회전 + FrontSensor 재사용으로 수행

RESULT

회피 가능 방향을 결정하거나
UC-04로 전환한다.

USE CASE · UC-04

코너 탈출과 우측 개방 추론

PRECONDITION

UC-03 우측 확인 실패 이후 코너 탈출이 필요한 상태

MAIN SCENARIO

1. systemState를 REVERSING으로 전환하고 먼저 후진한다.
2. 후방이 막혀 있으면 WAIT_REAR_CLEAR 단계에서 정지한다.
3. 좌측 개방 여부는 현재 obstacleState로 판단한다.
4. 좌측이 열려 있으면 우측 확인 없이 좌회전 후 전진하여 탈출한다.
5. 좌측이 막혀 있으면 우회전 후 FrontSensor 결과로 우측 개방 여부를 판단한다.
6. 우측 확인 후 좌회전으로 원방향을 복구한다.
7. 우측이 열려 있으면 우회전 후 전진하여 탈출한다.
8. 좌측과 우측이 모두 막혀 있으면 SEARCH_ESCAPE_DIRECTION 단계를 유지하며 후진 및 방향 탐색을 반복한다.

ALTERNATIVE

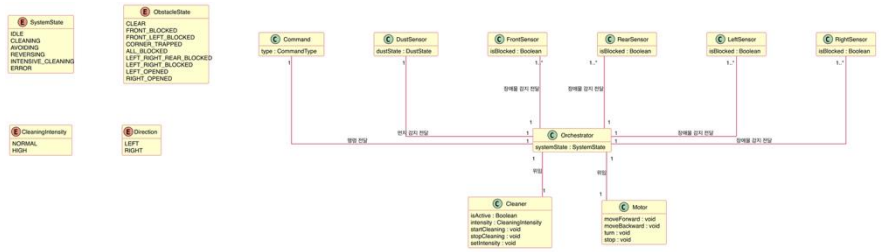
좌측이 열려 있으면 우측 확인 없이 좌측으로 즉시 탈출한다.
좌측이 막힌 경우에만 우측을 확인하며, 우측이 열려 있으면 우측으로 탈출한다.
양쪽이 모두 막혀 있으면 후방 대기 또는 추가 탐색을 유지한다.

RESULT

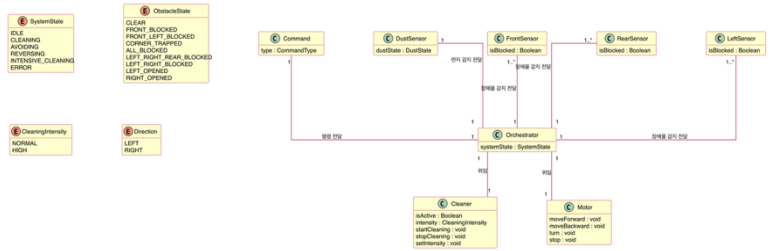
후진, 좌측 우선 탈출 판단, 우측 확인, 회전 복구가 단계별로 관리된다.

Domain Model 비교

Before



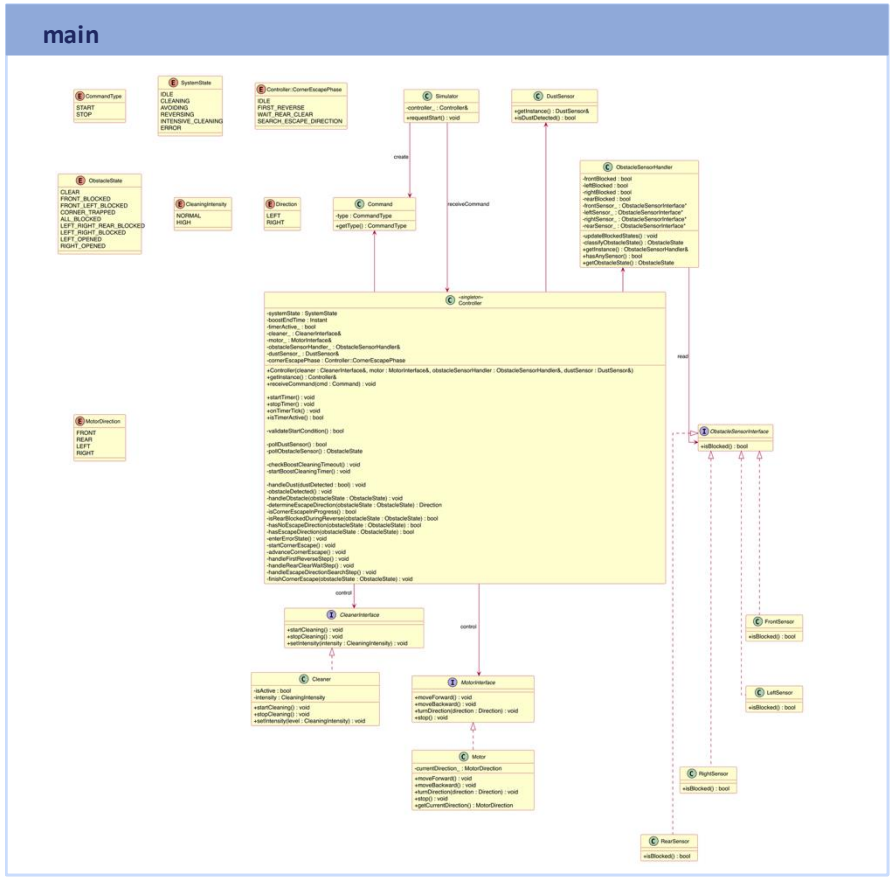
After



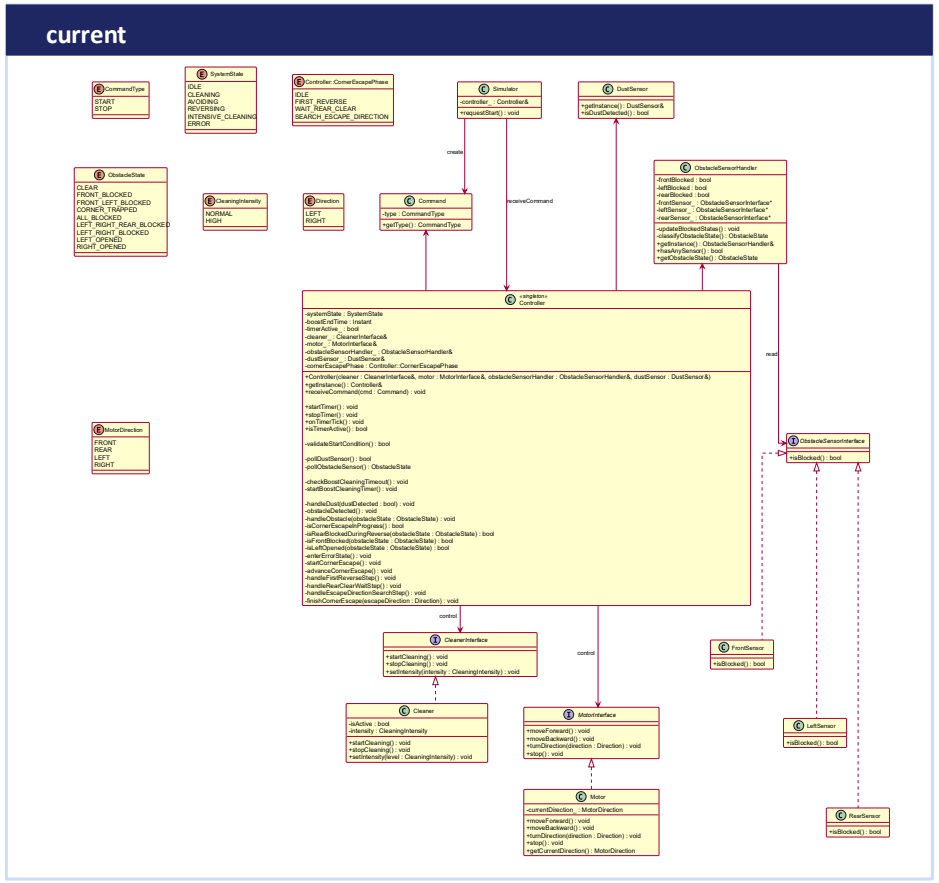
차이: RightSensor 객체와 Orchestrator 전달 관계 제거 · Front / Left / RearSensor 중심으로 단순화 · rightOpened는 파생 판단으로 이동

Class Diagram 비교

main



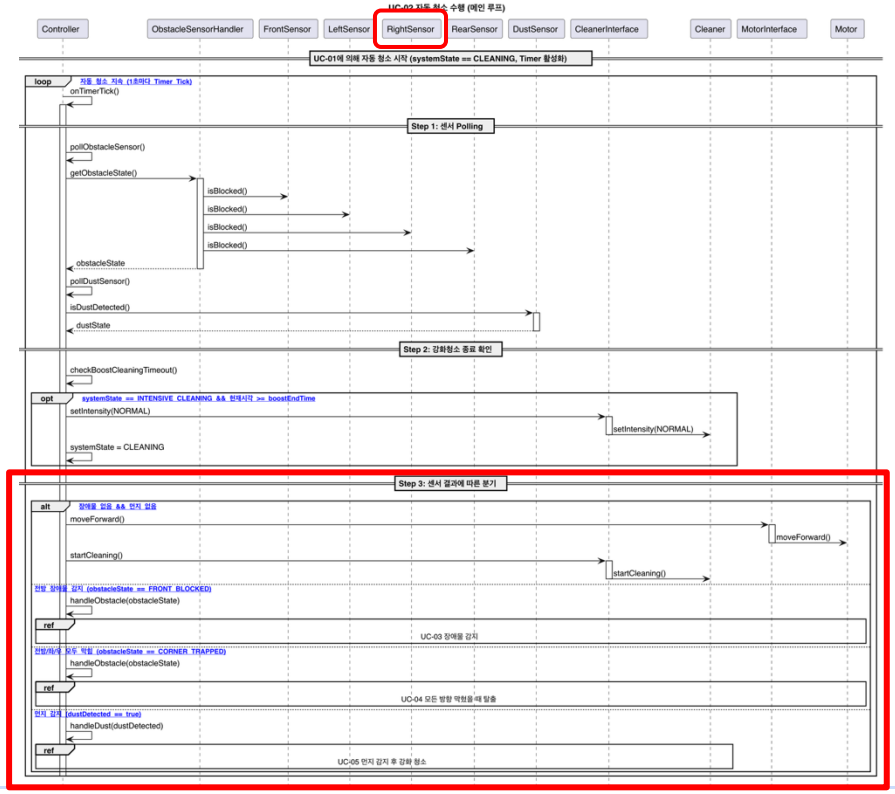
current



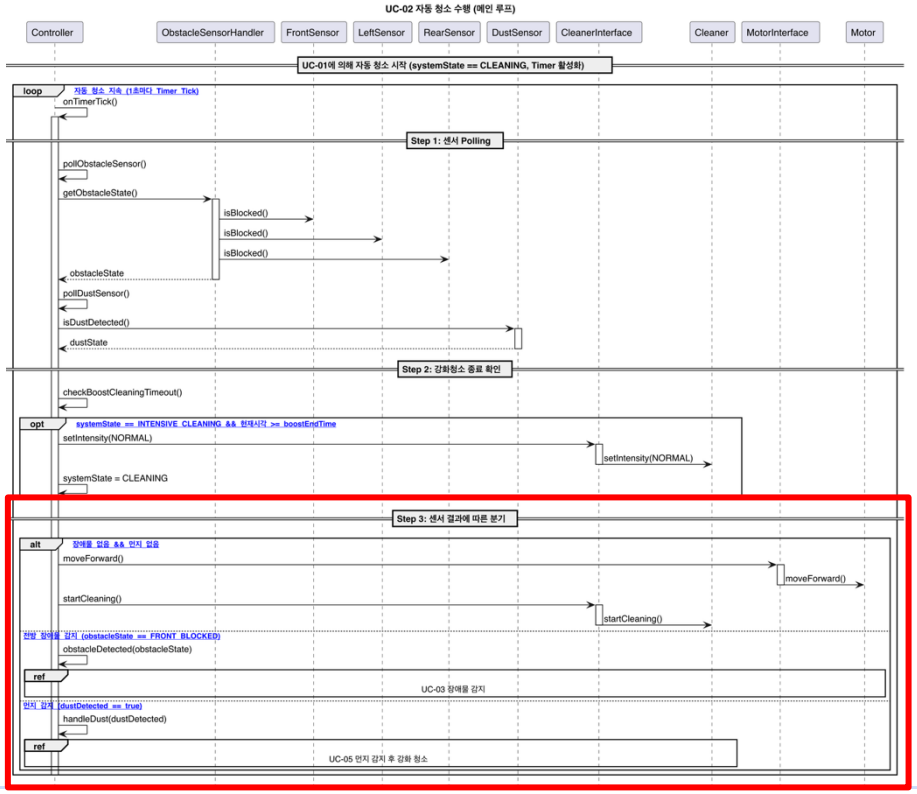
차이: rightSensor_ / rightBlocked 제거 · determineEscapeDirection(ObstacleState) → determineEscapeDirection(leftOpened, rightOpened)

UC-02 Sequence Diagram 비교

main

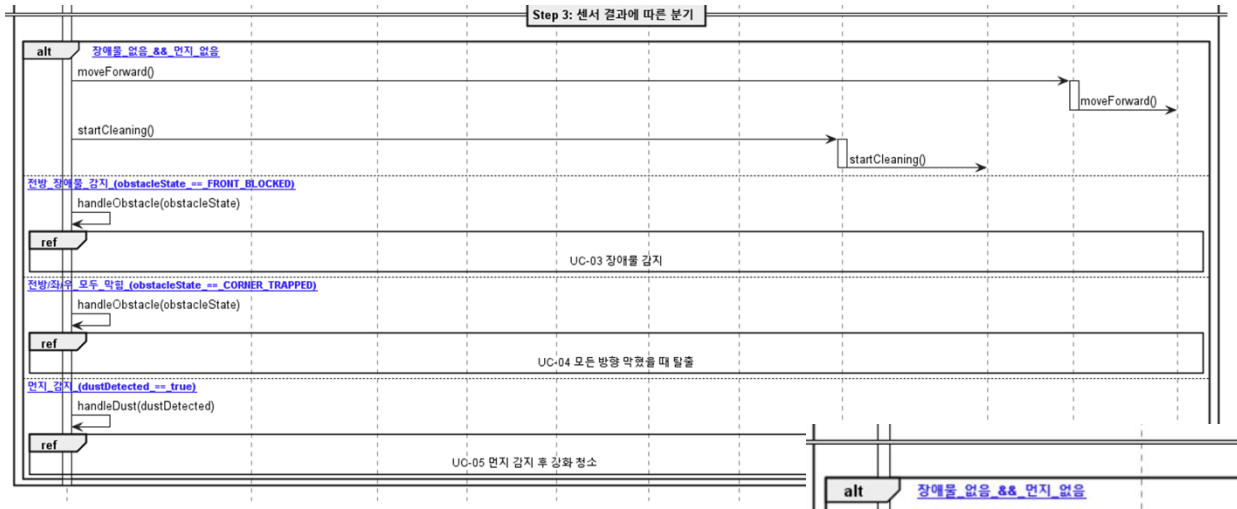


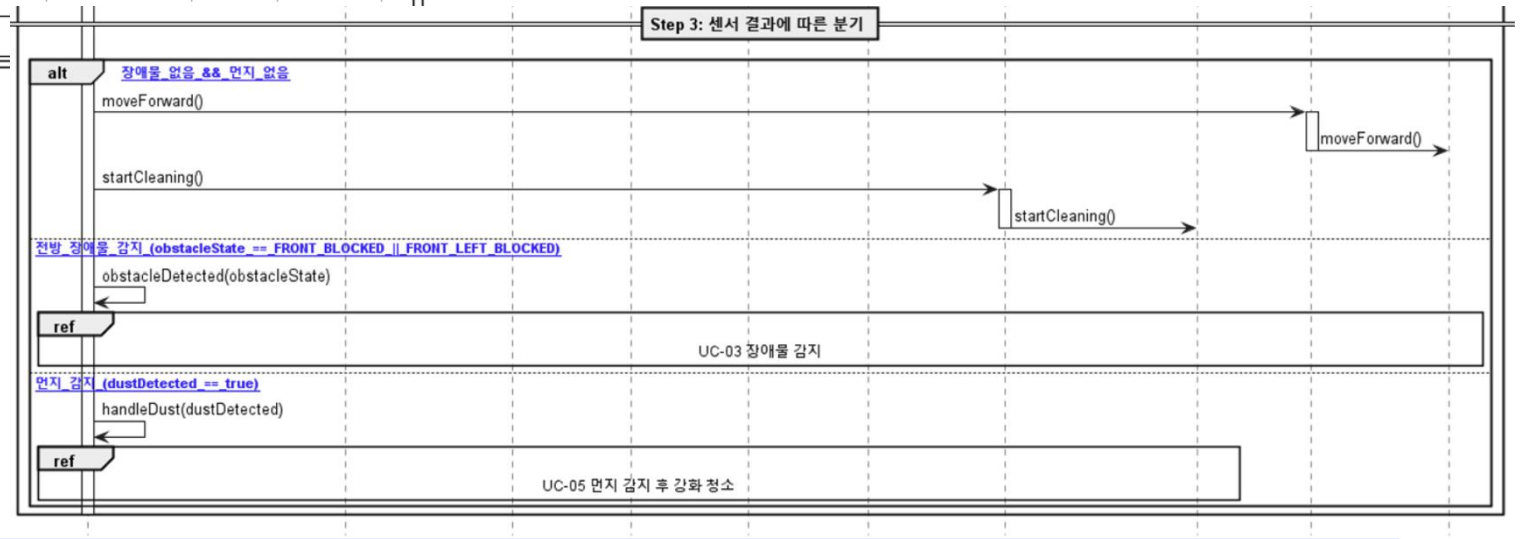
current



차이: Timer Tick에서 polling한 obstacleState를 obstacleDetected(obstacleState)로 직접 전달. UC-03 진입 조건이 obstacleState == CORNER_TRAPPED에서 obstacleState == FRONT_BLOCKED || FRONT_LEFT_BLOCKED로 변경

UC-02 Sequence Diagram 비교

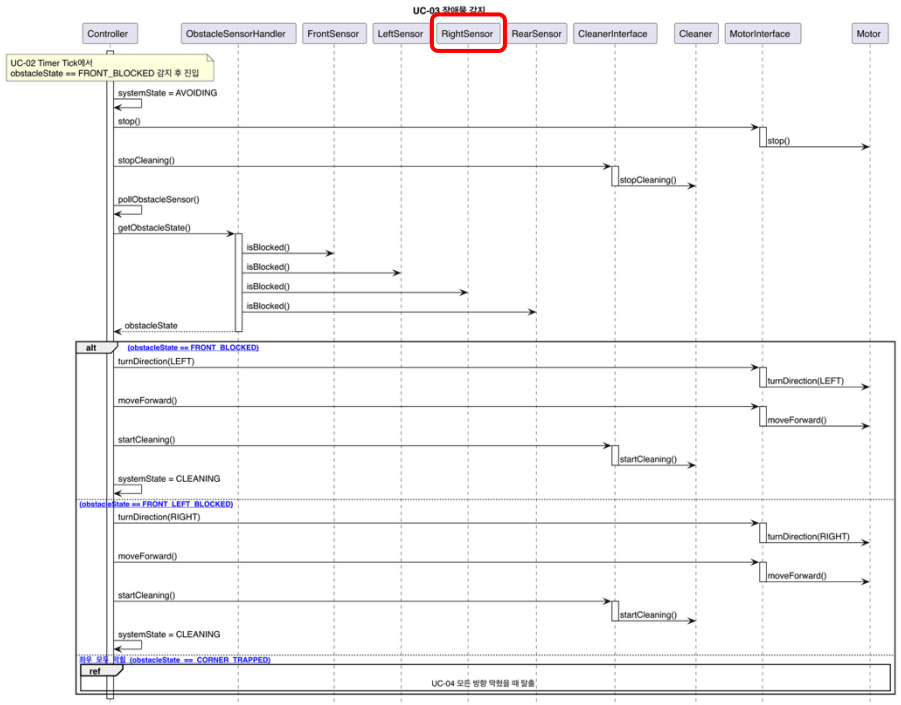




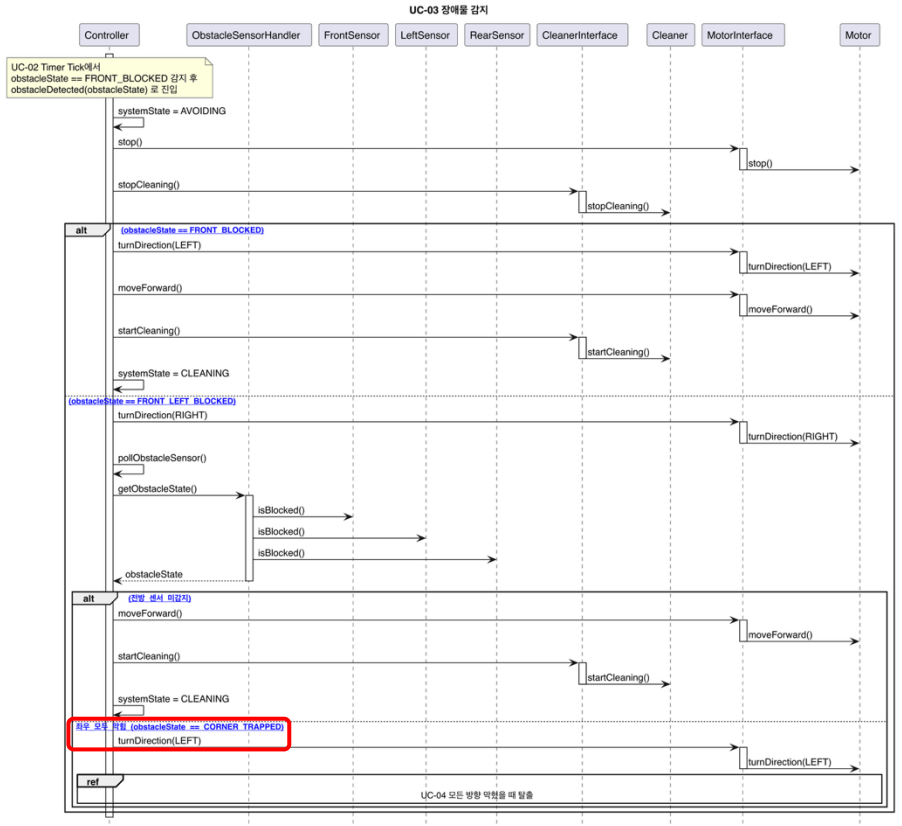
차이: Timer Tick에서 polling한 obstacleState를 obstacleDetected(obstacleState)로 직접 전달. UC-03 진입 조건이 obstacleState == CORNER_TRAPPED에서 obstacleState == FRONT_BLOCKED || FRONT_LEFT_BLOCKED로 변경

UC-03 Sequence 비교

main



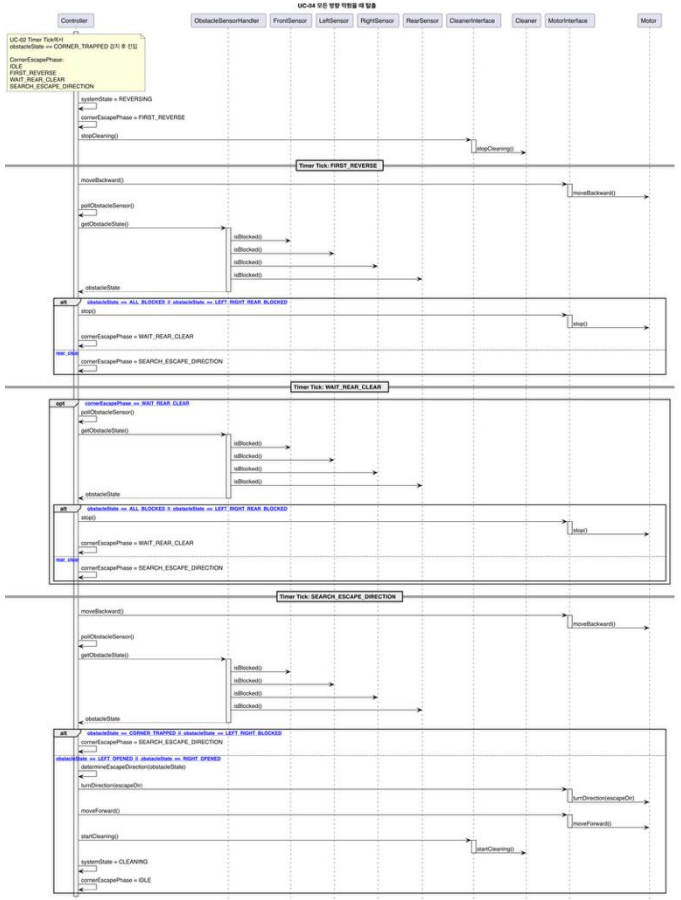
current



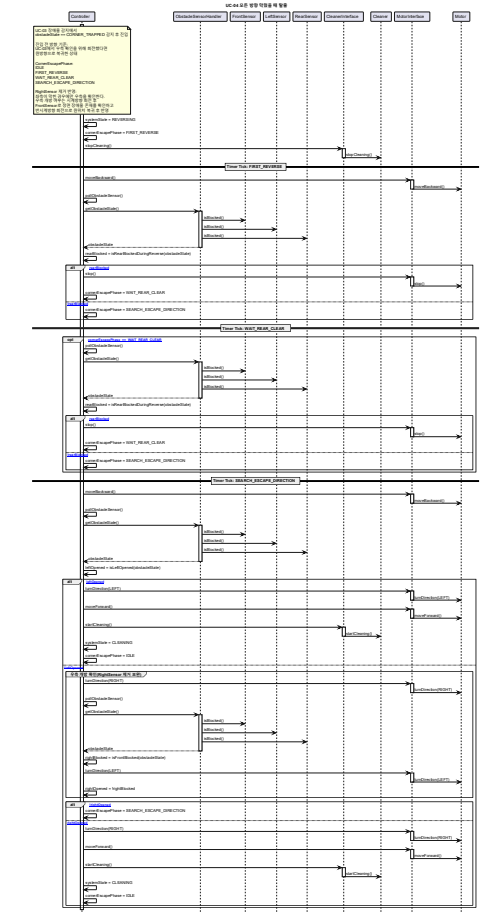
차이: 진입 직후 재폴링 제거 · FRONT_LEFT_BLOCKED에서 turnRight 후 FrontSensor로 우측 방향 확인

UC-04 Sequence 비교

main



current



차이: RightSensor 제거 · 좌측 우선 즉시 탈출 분기 추가 · 좌측 막힘 시에만 우회전 후 FrontSensor로 우측 확인하고 원방향 복구

IMPLEMENTATION

구현 변경 지점

Controller

obstacleDetected(ObstacleState)
UC-03/UC-04 연결
REVERSING 상태 관리

ObstacleSensorHandler

front / left / rear 센서만 보유
rightBlocked 멤버 제거
분류 로직 3방향 기준 단순화

Deleted

include/RightSensor.h
src/RightSensor.cpp
defaultRightSensor 구성

Helpers

isFrontBlocked()
isLeftOpened()
determineEscapeDirection(leftOpened, rightOpened)

State

SystemState::CLEANING
SystemState::REVERSING
rightOpened는 파생 판단으로 문서화

IMPLEMENTATION

구현 변경 지점 UC-02

include/ObstacleSensorHandler.h -3 Viewed

```
@@ -9,12 +9,10 @@ class ObstacleSensorHandler
 private:
     bool frontBlocked;
     bool leftBlocked;
-    bool rightBlocked;
     bool rearBlocked;

     ObstacleSensorInterface* frontSensor_;
     ObstacleSensorInterface* leftSensor_;
-    ObstacleSensorInterface* rightSensor_;
     ObstacleSensorInterface* rearSensor_;

     void updateBlockedStates();
@@ -24,7 +22,6 @@ class ObstacleSensorHandler
     ObstacleSensorHandler(
         ObstacleSensorInterface* frontSensor,
         ObstacleSensorInterface* leftSensor,
-        ObstacleSensorInterface* rightSensor,
         ObstacleSensorInterface* rearSensor
     );
```

```
-           obstacleDetected();
-       }
-       else if (obstacleState == ObstacleState::CORNER_TRAPPED)
-       {
-           handleObstacle(obstacleState);
+           obstacleDetected(obstacleState);
```

```
else if (obstacleState == ObstacleState::FRONT_BLOCKED ||
         obstacleState == ObstacleState::FRONT_LEFT_BLOCKED)
{
```

IMPLEMENTATION

구현 변경 지점 UC-03

```
+ void Controller::obstacleDetected(ObstacleState obstacleState)
  {
      systemState = SystemState::AVOIDING;
      motor_.stop();
      cleaner_.stopCleaning();

-     ObstacleState obstacleState = pollObstacleSensor();
```

```
motor_.turnDirection(Direction::RIGHT);
motor_.moveForward();
cleaner_.startCleaning();
systemState = SystemState::CLEANING;

ObstacleState rightCheckState = pollObstacleSensor();

const bool rightBlocked =
    rightCheckState == ObstacleState::FRONT_BLOCKED ||
    rightCheckState == ObstacleState::FRONT_LEFT_BLOCKED ||
    rightCheckState == ObstacleState::CORNER_TRAPPED;

if (!rightBlocked)
{
    motor_.moveForward();
    cleaner_.startCleaning();
    systemState = SystemState::CLEANING;
}
else
{
    motor_.turnDirection(Direction::LEFT);
    handleObstacle(ObstacleState::CORNER_TRAPPED);
}
```

IMPLEMENTATION

구현 변경 지점 UC-04

```
void Controller::obstacleDetected(ObstacleState obstacleState)
@@ -195,10 +195,7 @@ void Controller::obstacleDetected(ObstacleState obstacleState)

        ObstacleState rightCheckState = pollObstacleSensor();

-       const bool rightBlocked =
-           rightCheckState == ObstacleState::FRONT_BLOCKED ||
-           rightCheckState == ObstacleState::FRONT_LEFT_BLOCKED ||
-           rightCheckState == ObstacleState::CORNER_TRAPPED;
+       const bool rightBlocked = isFrontBlocked(rightCheckState);

        if (!rightBlocked)
        {
```

- 기존 네 방향 센서를 동시에 polling 하는 로직에서
  우측 막힘 여부를 우회전 + 전면 센서로 감지로 변경

```
+ void Controller::obstacleDetected(ObstacleState obstacleState)
  {
      systemState = SystemState::AVOIDING;
      motor_.stop();
      cleaner_.stopCleaning();

-     ObstacleState obstacleState = pollObstacleSensor();
      if (obstacleState == ObstacleState::FRONT_BLOCKED)
      {
          motor_.turnDirection(Direction::LEFT);
          motor_.moveForward();
          cleaner_.startCleaning();
          systemState = SystemState::CLEANING;
      }
-     else if (obstacleState == ObstacleState::FRONT_LEFT_BLOCKED)
+     else if (obstacleState == ObstacleState::FRONT_LEFT_BLOCKED ||
+              obstacleState == ObstacleState::CORNER_TRAPPED)
      {
          motor_.turnDirection(Direction::RIGHT);
-         motor_.moveForward();
-         cleaner_.startCleaning();
-         systemState = SystemState::CLEANING;
+
+         ObstacleState rightCheckState = pollObstacleSensor();
+
+         const bool rightBlocked = isFrontBlocked(rightCheckState);
+
+         if (!rightBlocked)
+         {
+             motor_.moveForward();
+             cleaner_.startCleaning();
+             systemState = SystemState::CLEANING;
+         }
+         else
+         {
+             motor_.turnDirection(Direction::LEFT);
+             handleObstacle(ObstacleState::CORNER_TRAPPED);
+         }
```

- 후진 중 좌측이 열린 경우 우측 확인을 위해 우회전 하는 것은 비효율
  → 좌측이 열리면 바로 좌회전, 우측 확인은 좌측 막혔을 때만

TEST STRATEGY

RightSensor 제거 후 회피/탈출 테스트 재정렬

ObstacleSensorHandlerTest

front/left/rear fixture로 변경
우측 직접 감지 케이스 제거
FRONT_BLOCKED / FRONT_LEFT_BLOCKED 재분류 검증

System / Build

CMake 테스트 타깃에 UC-03 재포함
macOS/Linux/Windows 실행 파일명 처리
corner scenario 데이터 갱신

UC-03 ControllerTest

stop → turnLeft → moveForward 순서 검증
turnRight 후 front open이면 우측 회피
turnRight 후 front blocked이면 UC-04 시작

UC-04 ControllerTest

moveBackward → turnRight → turnLeft 흐름 검증
좌측/우측/양쪽 개방 방향 선택 검증
탈출 완료 뒤 phase reset 검증

CMake/CTest 전체 통과 · 회귀 테스트는 명령 순서 중심으로 강화

테스트 변경 지점 UC-04

```
-     EXPECT_NO_THROW(controller.obstacleDetected());
- }
+     ASSERT_EQ(motor.calls.size(), 4);
+     EXPECT_EQ(motor.calls[0], "stop");
+     EXPECT_EQ(motor.calls[1], "turnDirection");
+     EXPECT_EQ(motor.calls[2], "turnDirection");
+     EXPECT_EQ(motor.calls[3], "moveBackward");
+     ASSERT_EQ(motor.turnDirections.size(), 2);
+     EXPECT_EQ(motor.turnDirections[0], Direction::RIGHT);
+     EXPECT_EQ(motor.turnDirections[1], Direction::LEFT);
+     EXPECT_EQ(cleaner.stopCleaningCount, 2);
+     EXPECT_EQ(cleaner.startCleaningCount, 0);
+ }
```

- UC-04 NO_THROW 대신 모터 및 클리너를 직접 검증하도록 테스트 보강

```
+ TEST(UseCase03ControllerTest, FrontLeftBlockedWithRightBlockedRestoresDirectionAndStartsCornerEscape)
  {
-     frontSensor.setBlocked(true);
-     leftSensor.setBlocked(true);
-     rightSensor.setBlocked(false);
-     rearSensor.setBlocked(false);
+     TextGridReader gridReader{"test.txt"};
+     ScriptedObstacleSensor frontSensor{true, true, true};
+     ScriptedObstacleSensor leftSensor{true, true, true};
+     ScriptedObstacleSensor rearSensor{false, false, false};
+     ObstacleSensorHandler obstacleHandler{
+         &frontSensor,
+         &leftSensor,
+         &rearSensor
+     };
+     NoDustSensor dustSensor{gridReader};
+     SpyCleaner cleaner;
+     SpyMotor motor;
+     Controller controller{cleaner, motor, obstacleHandler, dustSensor};
+
+     controller.onTimerTick();
```

```
-     EXPECT_NO_THROW(controller.obstacleDetected());
- }
+     ASSERT_EQ(motor.calls.size(), 4);
+     EXPECT_EQ(motor.calls[0], "stop");
+     EXPECT_EQ(motor.calls[1], "turnDirection");
+     EXPECT_EQ(motor.calls[2], "turnDirection");
+     EXPECT_EQ(motor.calls[3], "moveBackward");
+     ASSERT_EQ(motor.turnDirections.size(), 2);
+     EXPECT_EQ(motor.turnDirections[0], Direction::RIGHT);
+     EXPECT_EQ(motor.turnDirections[1], Direction::LEFT);
+     EXPECT_EQ(cleaner.stopCleaningCount, 2);
+     EXPECT_EQ(cleaner.startCleaningCount, 0);
+ }
```

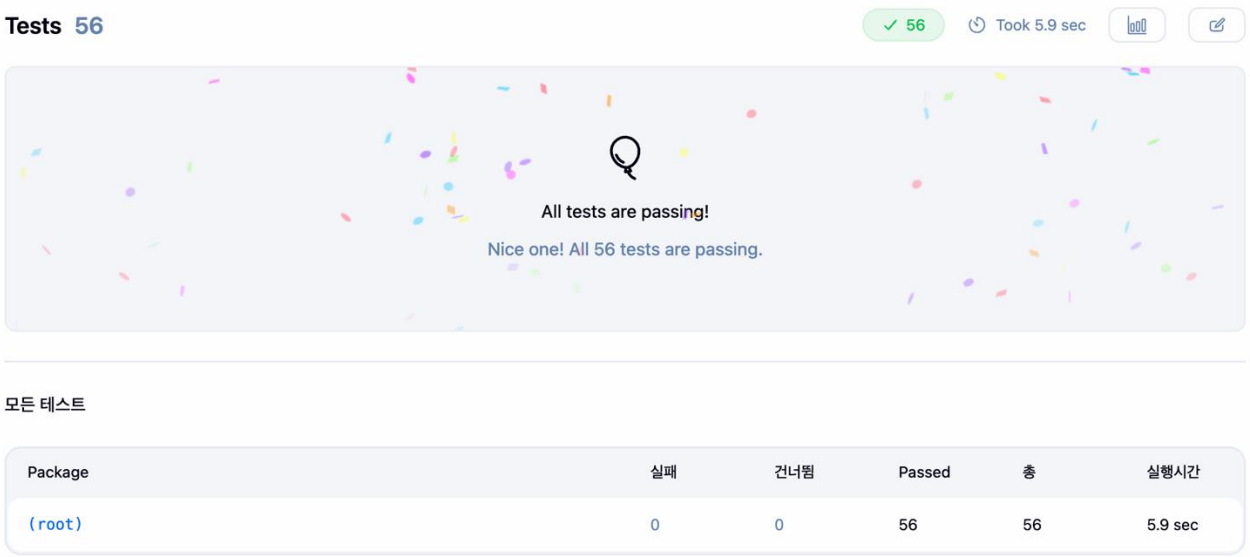
- 우측 막힌 경우 UC-03 → UC-04 전이 테스트

TEST RESULT

Google Test 전체 통과

RVC SW Controller · Regression Test

56	0	0
Tests Passed	Failed	Skipped



의미

- RightSensor 제거 후 회피/탈출 회귀 흐름을 자동 검증
명령 순서 spy 테스트로 실제 제어 흐름을 확인

Clang-Tidy / Cppcheck

RVC SW Controller · Code Quality

Clang-Tidy

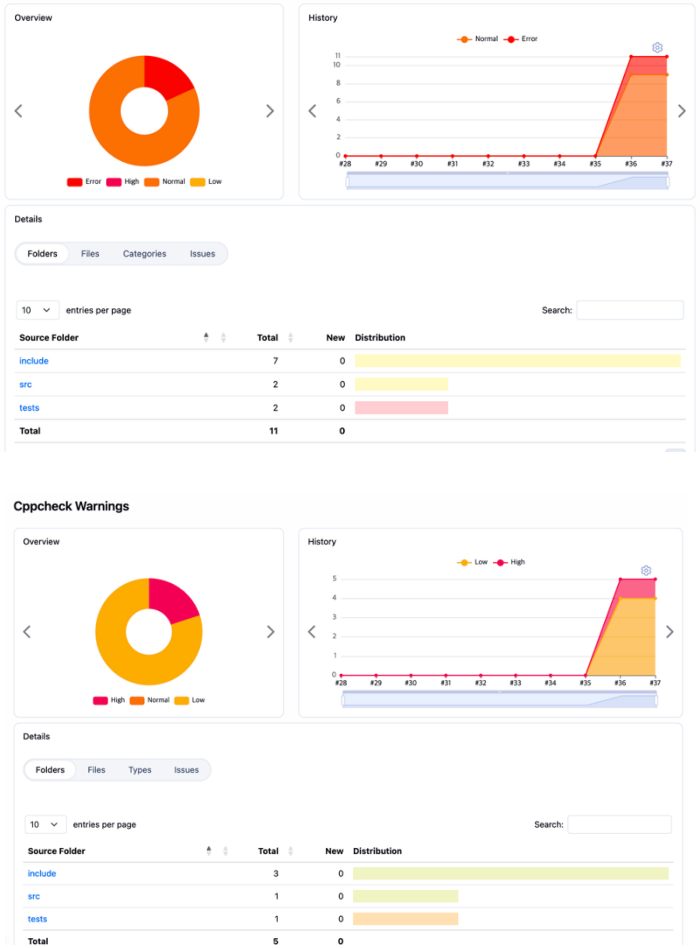
- Clang-Tidy: total 11 warnings

Cppcheck: total 5 warnings / new 0
warning trend를 통해 신규 품질 이슈 추적

의 미

- 빌드와 테스트가 통과한 뒤에도 정적 품질 이슈를 별도로 가시화

RightSensor 제거 이후 남은 참조와 code smell을 후속 정리 대상으로 추적



GCC Code Coverage

RVC SW Controller · Test Coverage

80.4%	89.3%	25.1%
Lines	**Functions**	**Branches**
874 / 1087	268 / 300	593 / 2359

주요 관찰

- Controller.cpp 90.1%, ObstacleSensorHandler.cpp 98.3% line coverage

우측 센서 제거 후 핵심 회피 로직은 높은 line coverage로 보호

GCC Code Coverage Report

Directory: ./
Date: 2026-06-03 10:57:35
Coverage: low: ≥ 0% medium: ≥ 75.0% high: ≥ 90.0%

	Exec	Total	Coverage
Lines:	874	1087	80.4%
Functions:	268	300	89.3%
Branches:	593	2359	25.1%

List of functions

File	Lines	Functions	Branches
include/CleanerInterface.h	100.0% 1 / 1	100.0% 1 / 1	-% 0 / 0
include/DustSensor.h	100.0% 1 / 1	100.0% 1 / 1	-% 0 / 0
include/MotorInterface.h	100.0% 1 / 1	100.0% 1 / 1	-% 0 / 0
include/ObstacleSensorInterface.h	100.0% 1 / 1	100.0% 1 / 1	-% 0 / 0
src/Cleaner.cpp	95.2% 20 / 21	100.0% 5 / 5	66.7% 2 / 3
src/Command.cpp	100.0% 3 / 3	100.0% 2 / 2	-% 0 / 0
src/Controller.cpp	90.1% 146 / 162	92.6% 25 / 27	78.2% 68 / 87
src/DustSensor.cpp	100.0% 10 / 10	100.0% 3 / 3	37.5% 3 / 8
src/FrontSensor.cpp	100.0% 7 / 7	100.0% 2 / 2	-% 0 / 0
src/LeftSensor.cpp	100.0% 7 / 7	100.0% 2 / 2	-% 0 / 0
src/main.cpp	0.0% 0 / 139	0.0% 0 / 16	0.0% 0 / 198
src/Motor.cpp	51.0% 25 / 49	50.0% 3 / 6	47.1% 8 / 17
src/ObstacleSensorHandler.cpp	98.3% 59 / 60	100.0% 6 / 6	78.9% 90 / 114
src/RearSensor.cpp	100.0% 7 / 7	100.0% 2 / 2	-% 0 / 0
src/RightSensor.cpp	100.0% 7 / 7	100.0% 2 / 2	-% 0 / 0
src/Simulator.cpp	0.0% 0 / 7	0.0% 0 / 2	0.0% 0 / 4
src/TextGridReader.cpp	87.0% 20 / 23	100.0% 3 / 3	48.1% 25 / 52
tests/motor_test.cpp	100.0% 23 / 23	100.0% 9 / 9	32.0% 16 / 50
tests/uc01_controller_start_test.cpp	70.4% 19 / 27	63.6% 7 / 11	30.0% 9 / 30
tests/uc02_controller_test.cpp	100.0% 37 / 37	100.0% 21 / 21	16.3% 42 / 258
tests/uc02_obstacle_sensor_handler_test.cpp	100.0% 77 / 77	100.0% 29 / 29	28.6% 36 / 126
tests/uc04_controller_test.cpp	98.9% 272 / 275	98.8% 80 / 81	24.0% 184 / 766

SonarQube Quality Gate

RVC SW Controller · Quality Analysis

Passed ✓
Quality Gate

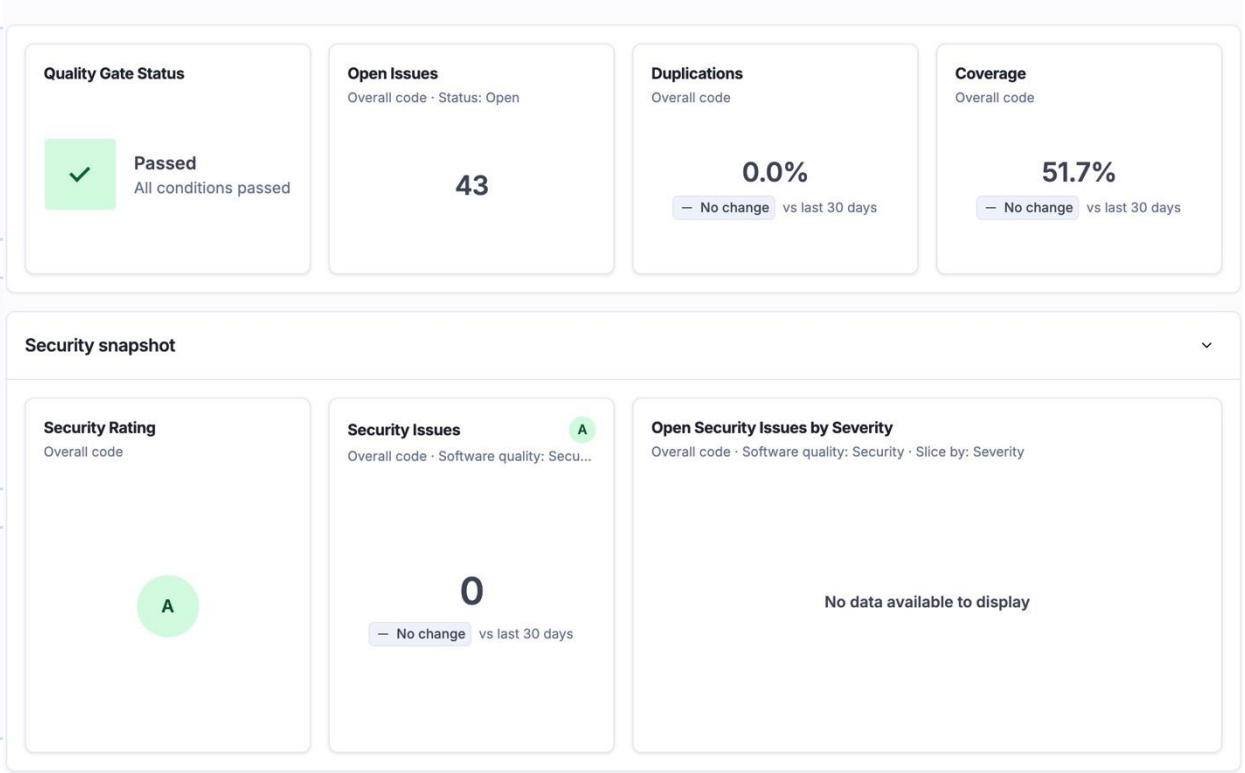
A
Security Rating

43
Open Issues

0
Security Issues

0.0%
Duplications

51.7%
Overall Coverage



의 미

Quality Gate Passed, Security Rating A 유지
Open Issues 43건은 정적 분석 결과와 함께 후속 정리 대상으로 추적

Robot Vacuum Cleaner

요구사항 분석 및 OOAD

3팀
202213351 김태성
202111382 최성준
202011434 최원탁
202011380 최용근

PROMPT

UC 작성 프롬프트

첨부 자료

x

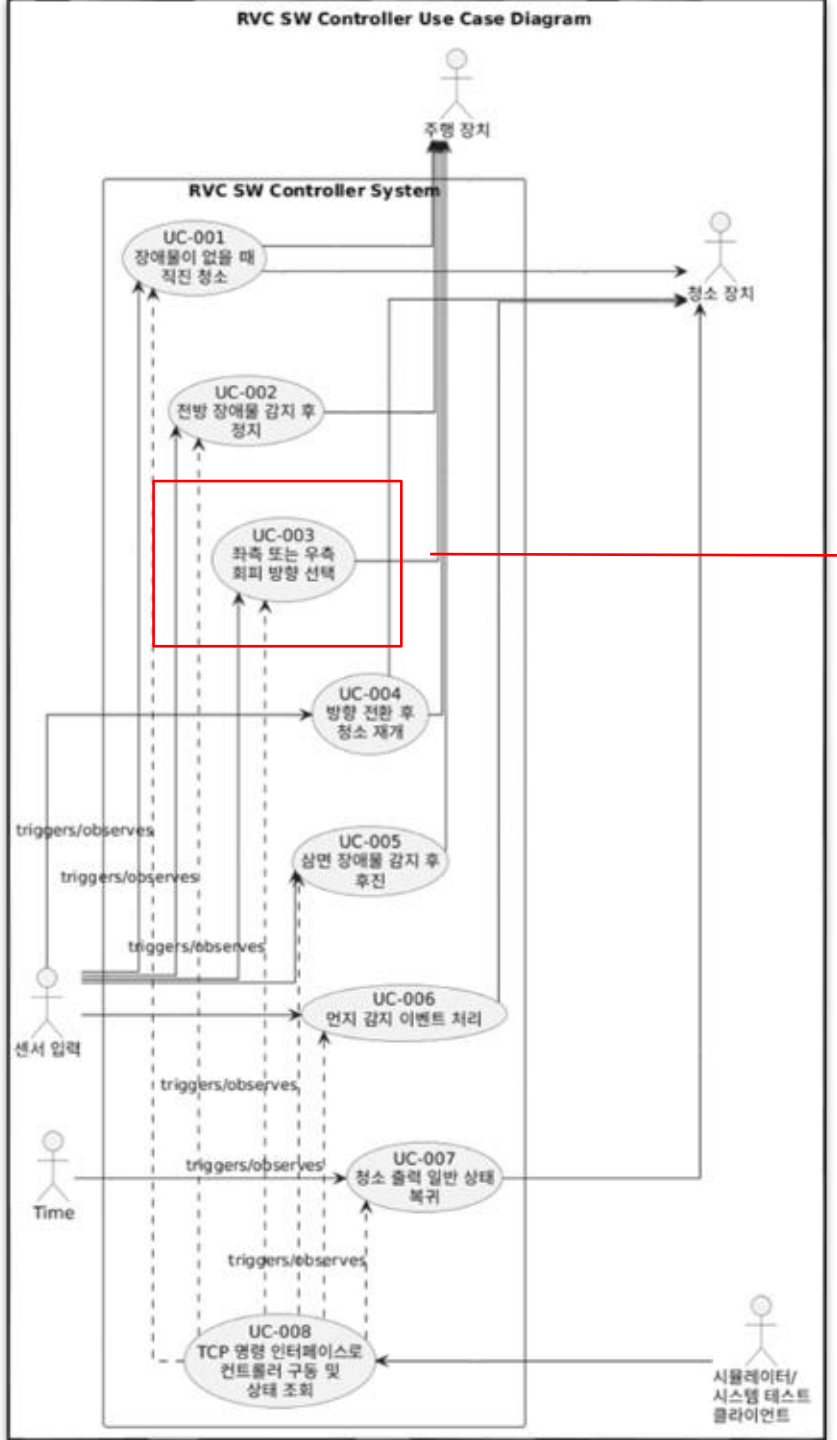
Prompt

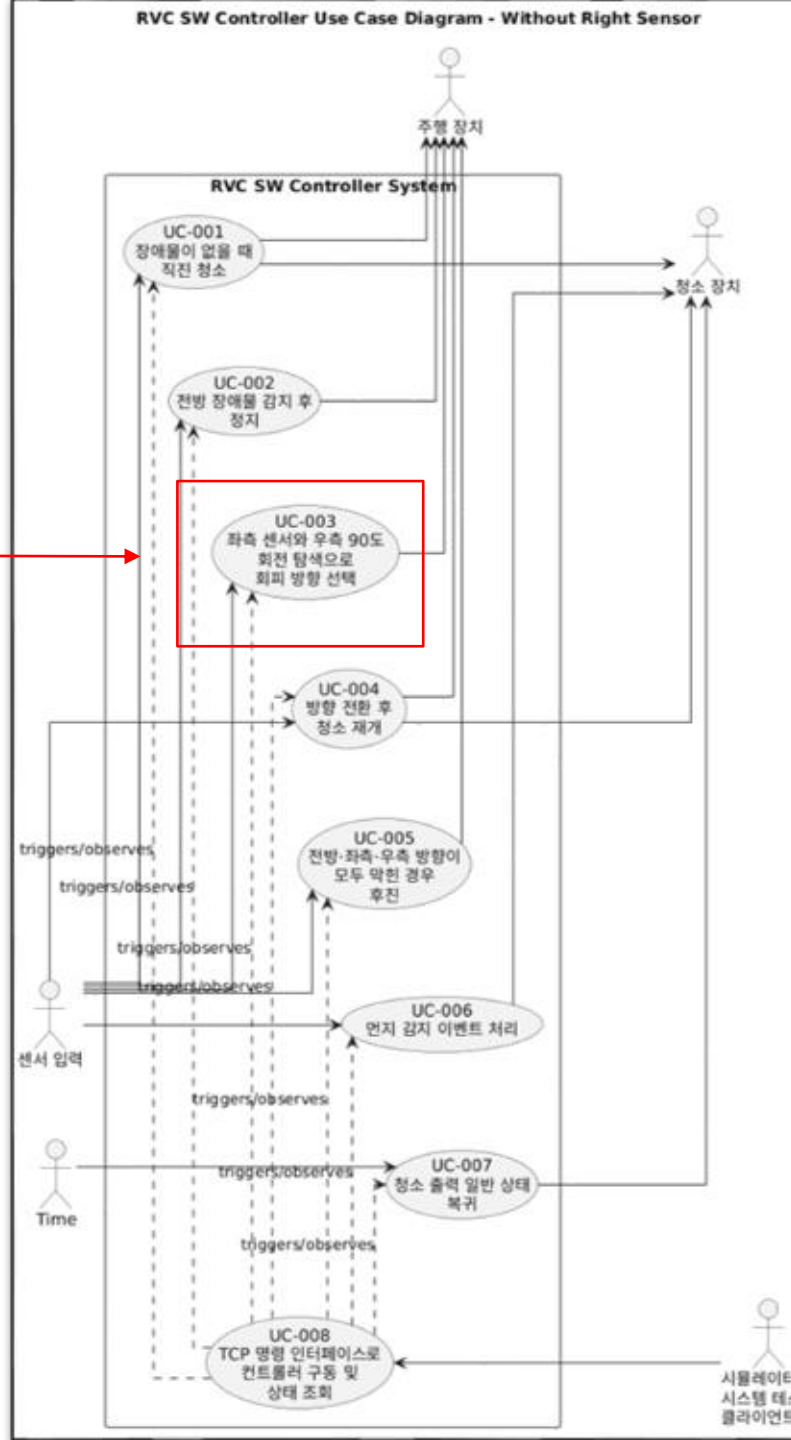
현재 프로젝트의 usecases.md 를 분석해. 이제 오른쪽 센서를 없앨 거야. 이 상태에서도 기존의 UC 동작이 모두 동작할 수 있게 usecase 를 변경해야해. 우측을 어떻게 감지할지 생각하고, 어떤 부분을 수정해야할지 분석해서 수정 계획을 나한테 알려줘

아쉬운 점

우측을 탐지하기 위해 우측으로 회전하는 과정을 생각한 것은 좋으나, 만일 우측으로 이동할 수 없다면 다시 좌측으로 회전하여 탐색 전과 같은 방향을
보게 해야한다는 것을 놓침.

Use Case Diagram





USE CASE · UC-002

전방 장애물 감지 후 정지

PRECONDITION

RVC가 전진 청소 중

MAIN SCENARIO

1. 전방 장애물 감지 확인
2. 전진 명령 중단
3. 주행 장치에 정지 명령
4. 방향 전환 판단 단계로 이전

ALTERNATIVE

전방 장애물 감지 시 → 좌측 회피 가능하면 좌측 회피 우선 (UC-003)
→ 우측 90도 회전 후 장애물 판단 → 우측 회피 or 후진 (UC-005)

RESULT

RVC는 전방 장애물 앞에서 정지하고
회피 판단을 준비한다.

USE CASE · UC-003

좌측 또는 우측 회피 방향 선택

PRECONDITION

방향 전환이 필요한 상태

MAIN SCENARIO

1. 좌측 장애물 감지 상태를 확인한다.
2. 좌측이 이동 가능하면 좌측 회피 방향을 우선 선택한다.
3. 좌측이 막혀 있어 우측 이동 가능 여부 확인이 필요하면 주행 장치에 우측으로 90도 회전 명령을 전달한다.
4. 회전 후 전방 센서로 기존 우측 방향의 장애물 여부를 확인한다.
5. 기존 우측 방향이 이동 가능하면 원래 정면 방향으로 복귀하지 않고 현재 자세를 우측 회피 완료 자세로 사용한다.
6. 기존 우측 방향이 막혀 있으면 주행 장치에 좌측으로 90도 회전 명령을 전달해 원래 정면 방향으로 복귀한다.
7. 좌측과 기존 우측 방향이 모두 막혀 있으면 UC-005로 전환한다.
8. 선택된 좌측 회피 방향 또는 현재 우측 회피 완료 자세를 청소 재개 단계로 전달한다.

ALTERNATIVE

좌측 이동 가능 시 우측 탐색 없이 이동 → 좌측 이동 불가능 & 우측 이동 가능이면 자세 전환 없이 UC-004 전환 → 좌/우 모두 이동 불가능이면 UC-005 전환

RESULT

SW Controller는 좌회전 또는 우회전 방향을 결정한다.

USE CASE · UC-005

삼면 장애물 감지 후 후진

PRECONDITION

전방·좌측·우측 장애물 모두 활성화

MAIN SCENARIO

1. SW Controller는 전방, 좌측, 우측 방향이 모두 막힌 상태와 후방 장애물 상태를 확인한다.
2. 주행 장치에 정지 명령을 전달한다.
3. 후방 장애물이 없어 후진할 수 있으면 주행 장치에 후진 명령을 전달한다.
4. 후진 동작이 끝난 뒤 좌측 장애물 상태를 먼저 확인한다.
5. 후진 후 좌측이 이동 가능하면 우측 탐색을 수행하지 않고 좌측 회피 방향을 우선 선택한다.
6. 후진 후 좌측이 막혀 있어 우측 방향 확인이 필요하면 UC-003과 동일하게 우측으로 90도 회전한 뒤 전방 센서로 기존 우측 방향 장애물 여부를 확인한다.
7. 후진 후 우측 탐색 결과가 이동 가능하면 원래 정면 방향으로 복귀하지 않고 현재 자세를 우측 회피 완료 자세로 사용해 UC-004로 전환한다.

ALTERNATIVE

후진 후 좌측 이동 가능하면 UC-003 전환 → 우측으로만 이동 가능하면 자세 전환 없이 UC-004 → 둘 다 불가능하면 후진

이

RESULT

RVC는 막힌 상태에서 후진하거나, 불가능하면 정지한다.

PROMPT

ssd 작성 프롬프트

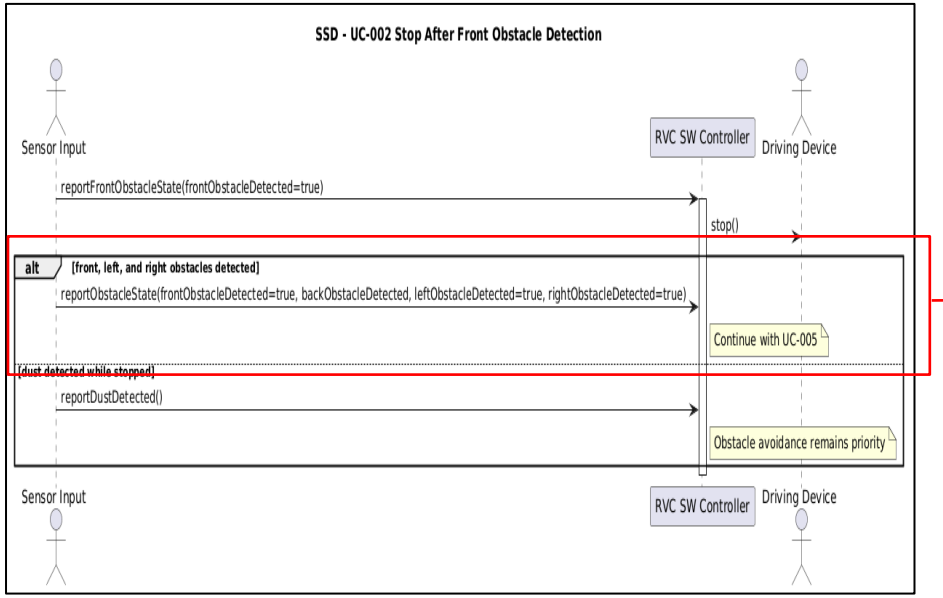
첨부 자료

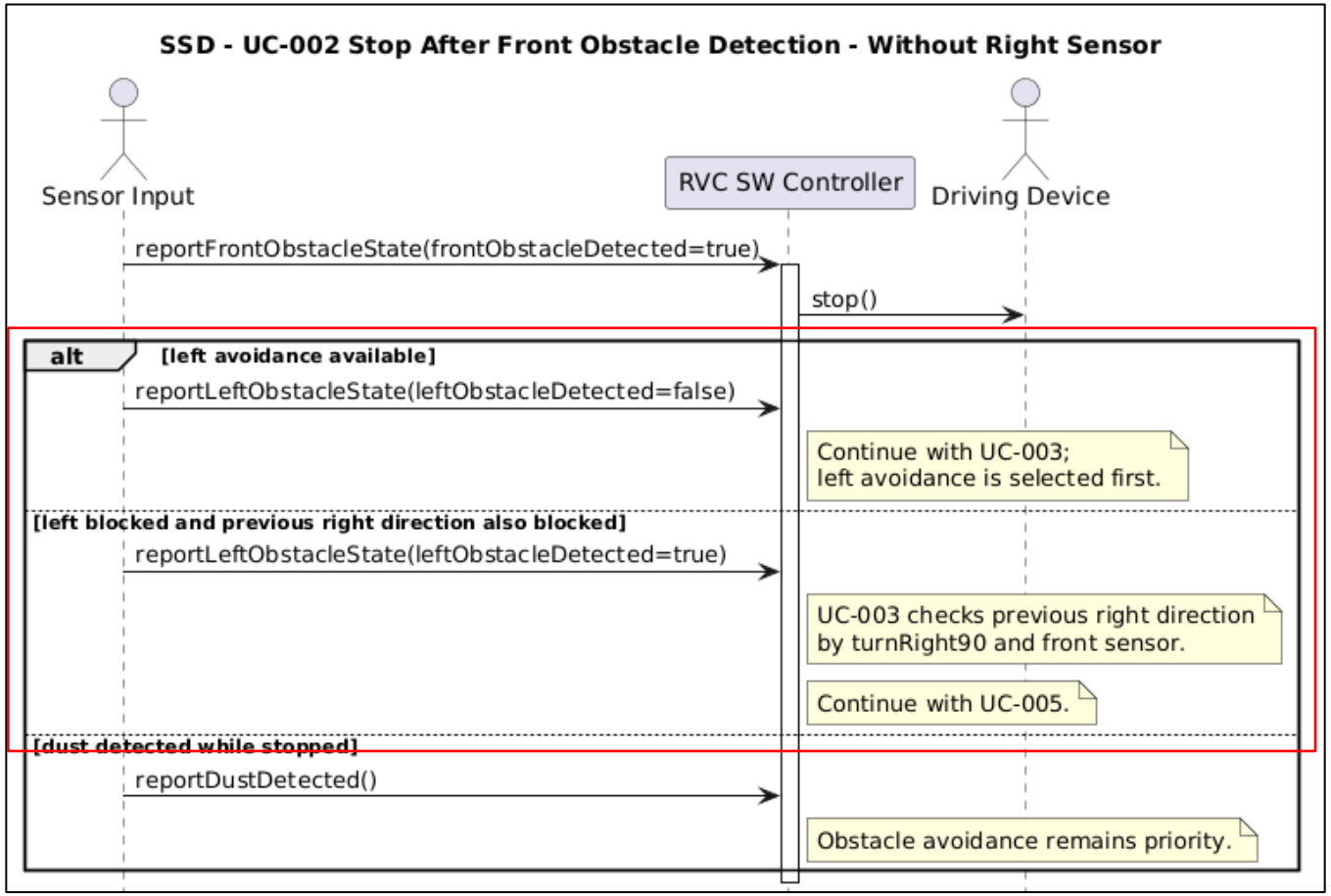
 # TEXT 1
usecase-without-right-sensor.md

Prompt

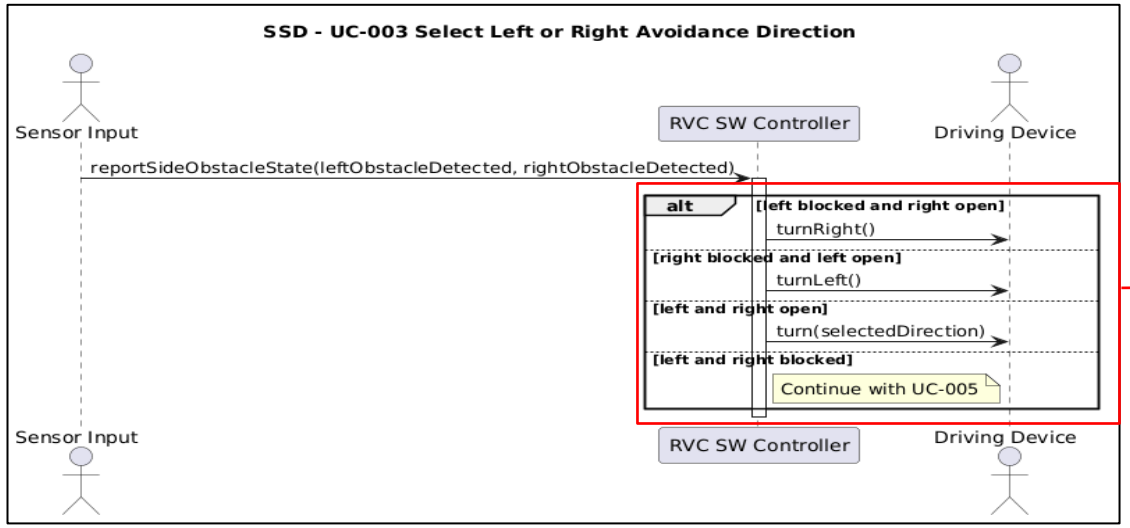
TEXT 1 을 바탕으로 기존 SSD 를 분석해봐. 변경된 UC 흐름에 따라서 수정할 거야. 먼저 변경이 필요한 범위를 특정하고, 해당 범위에서 어떻게 변경을 해야하는지 생각해. 변경이 필요없는 부분은 건들지마

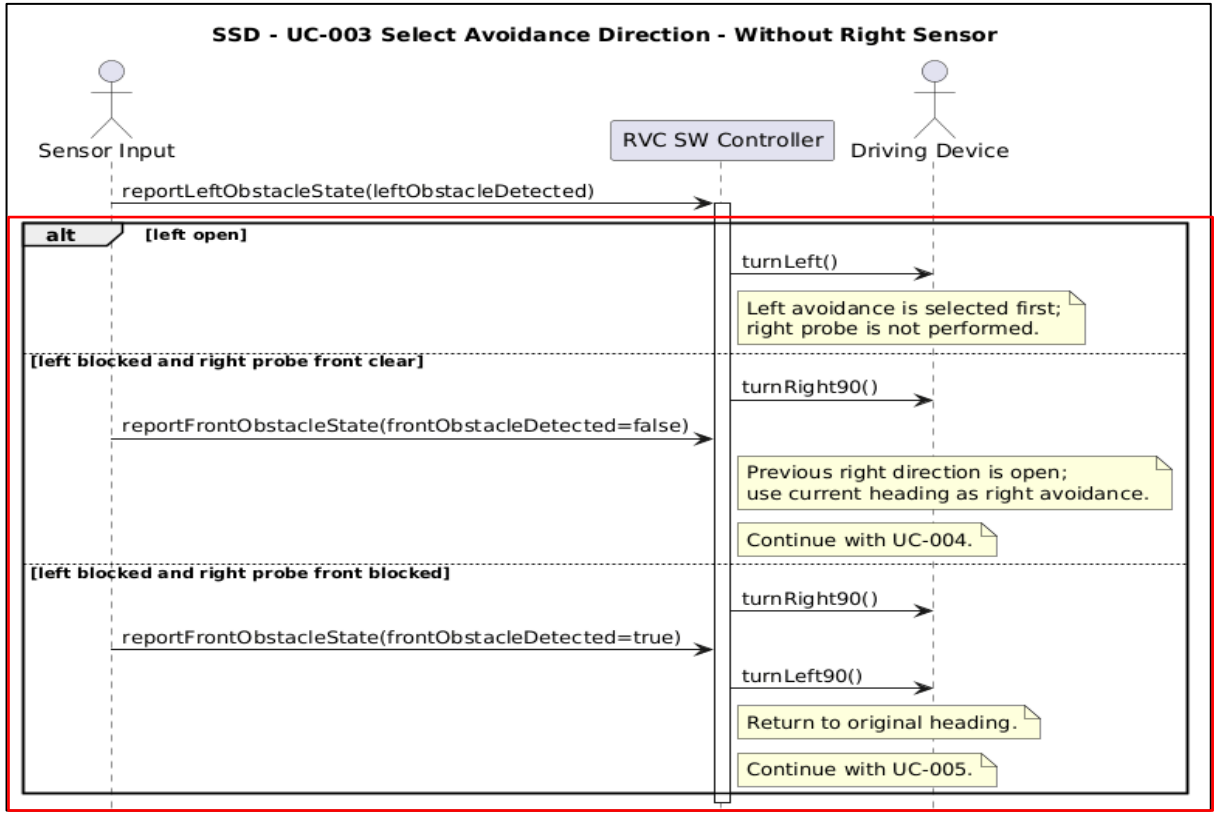
Stop After Front Obstacle Detection



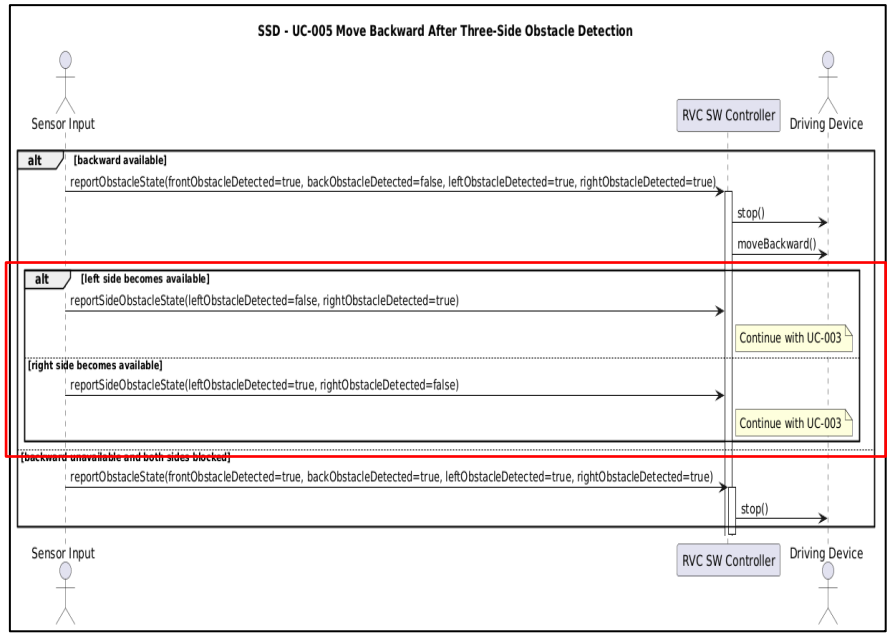


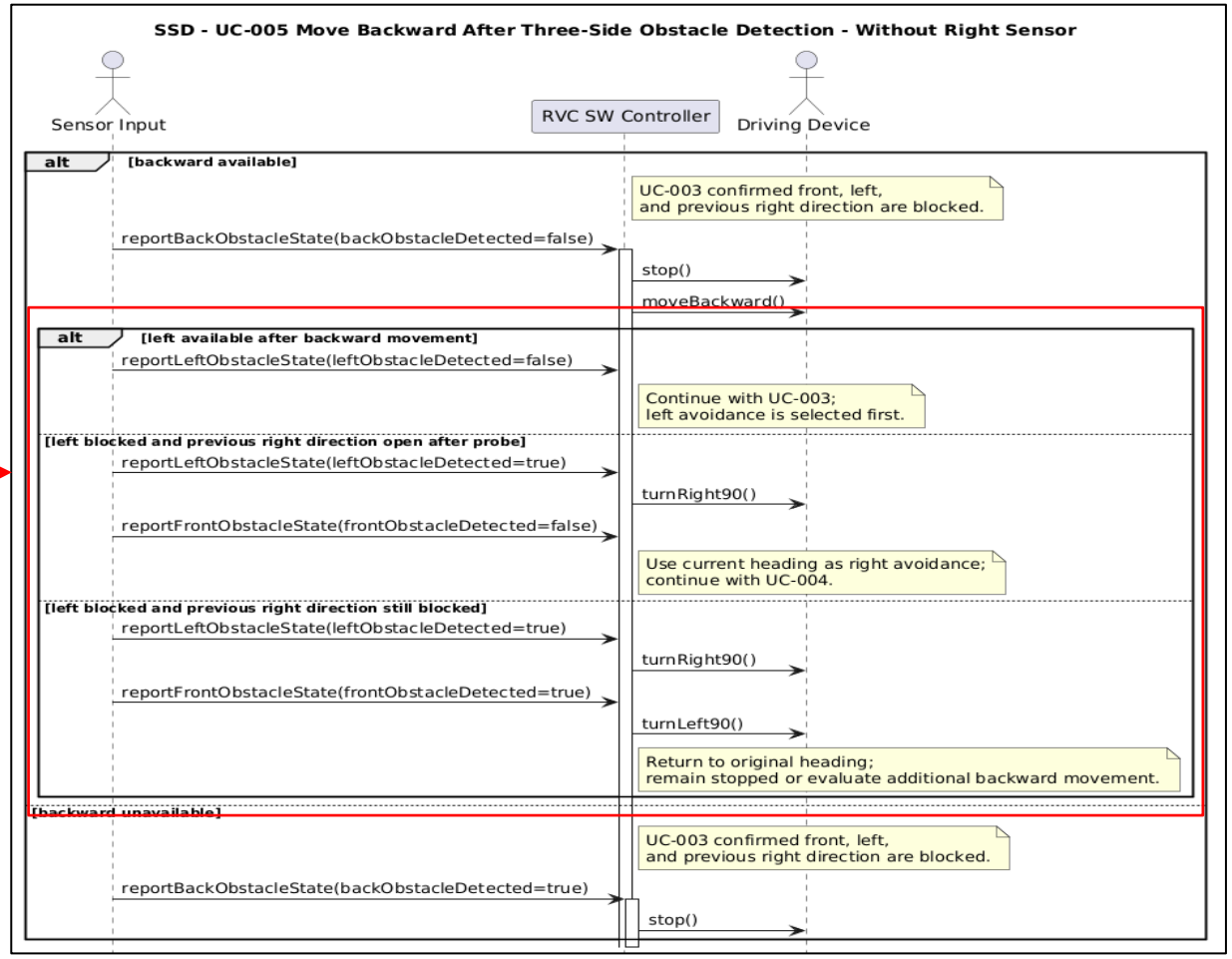
Select Left or Right Avoidance Direction





Move Backward After Three-Side Obstacle Detection





PROMPT

Domain model 작성 프롬프트

첨부 자료

 # TEXT 1
usecase-without-right-sensor.md

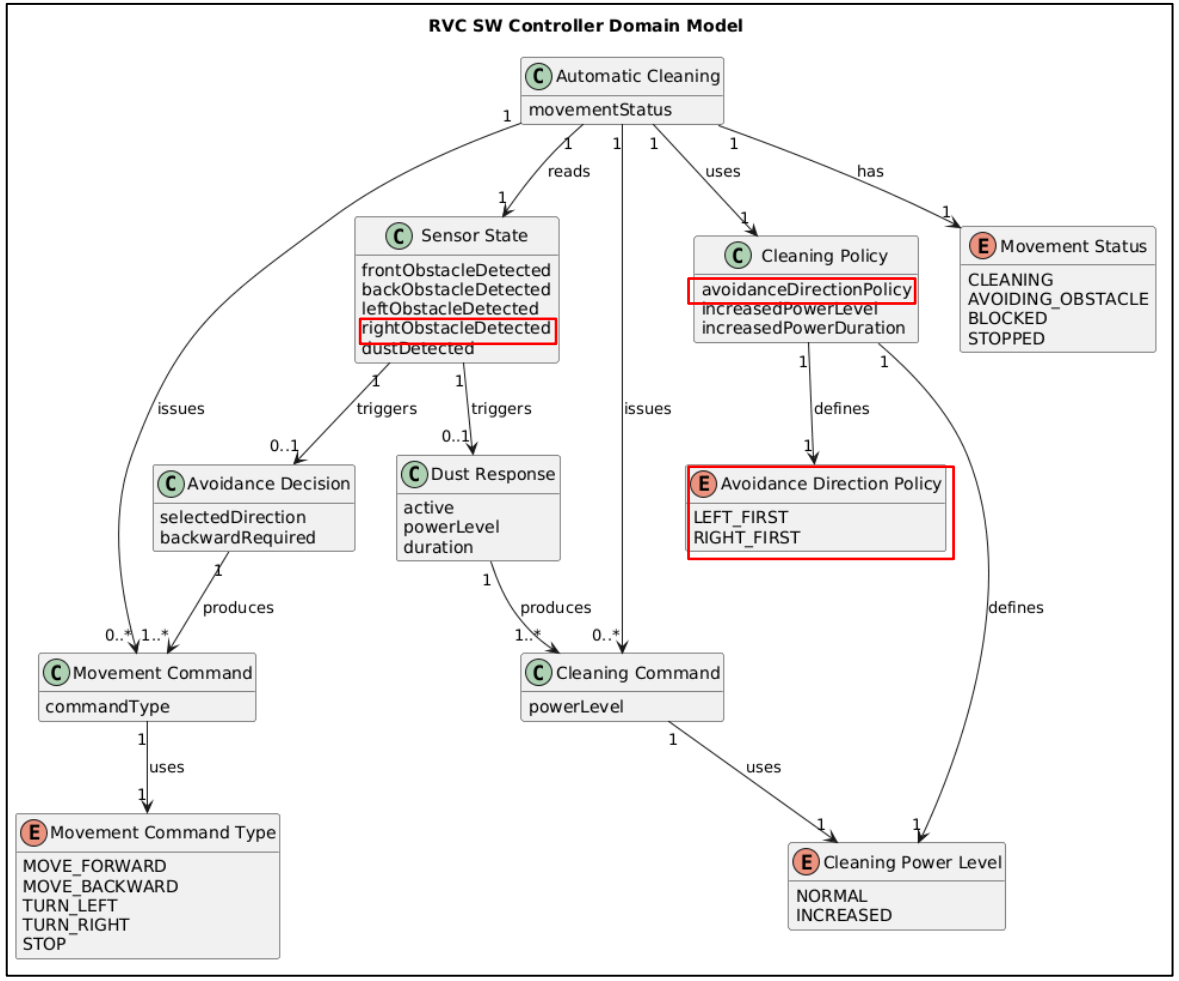
 # TEXT 2
ssd-without-right-sensor.md (2, 3, 5)

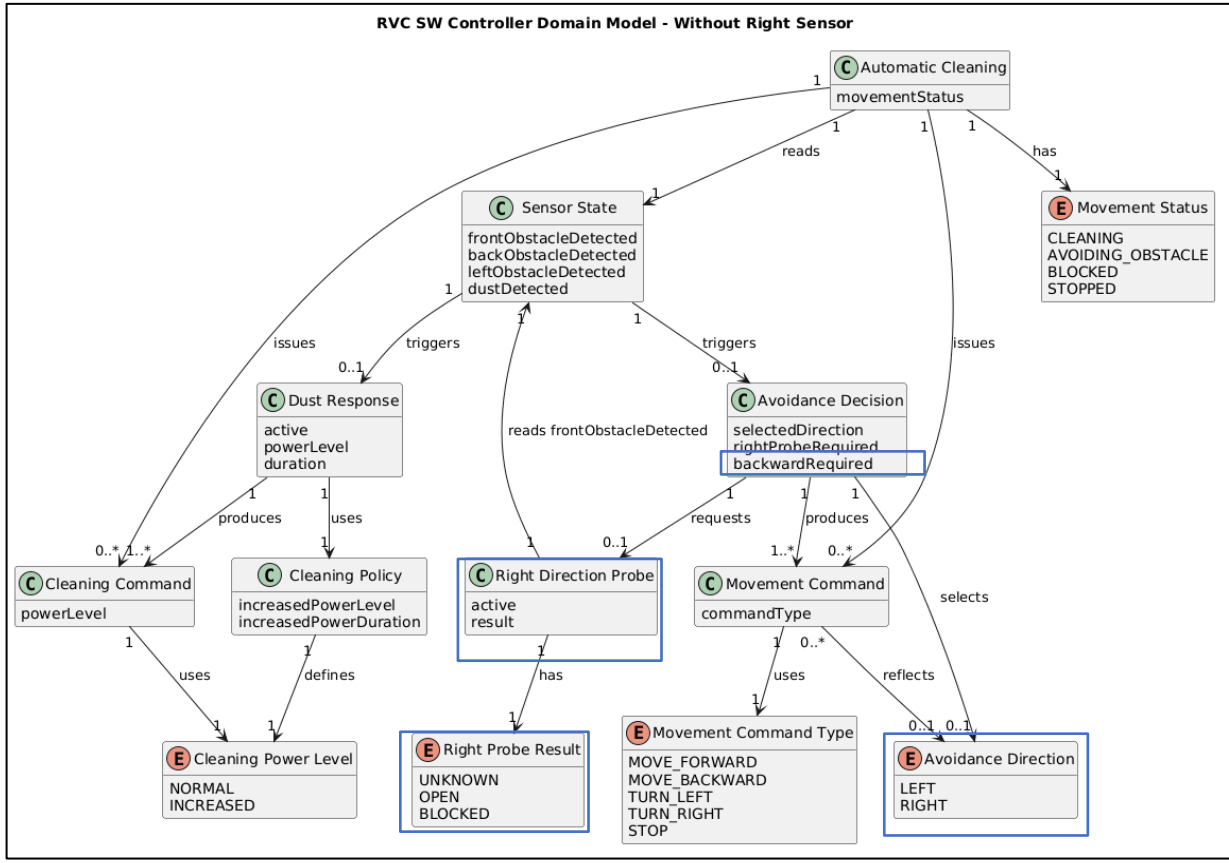
Prompt

TEXT 1 과 # TEXTS 2 를 바탕으로, 기존 도메인 모델을 어떻게 수정해야할지 생각해봐. UC 002, 003, 005 만 변경이 크게 되었으니까 이거 중점으로 고려하면 돼

아쉬운 점

불필요한 클래스를 추가하려고 계획함





1. Sensor State 의 rightObstacleDetected 제거
2. Avoidance Decision 에 rightProbeRequired 추가
3. Right Direction Probe 도메인 클래스가 추
4. Right Probe Result enum 추가 (UNKNOWN, OPEN, BLOCKED)

PROMPT

Sequence Diagram 작성 프롬프트

첨부 자료

 # TEXT 1
usecase-without-right-sensor.md

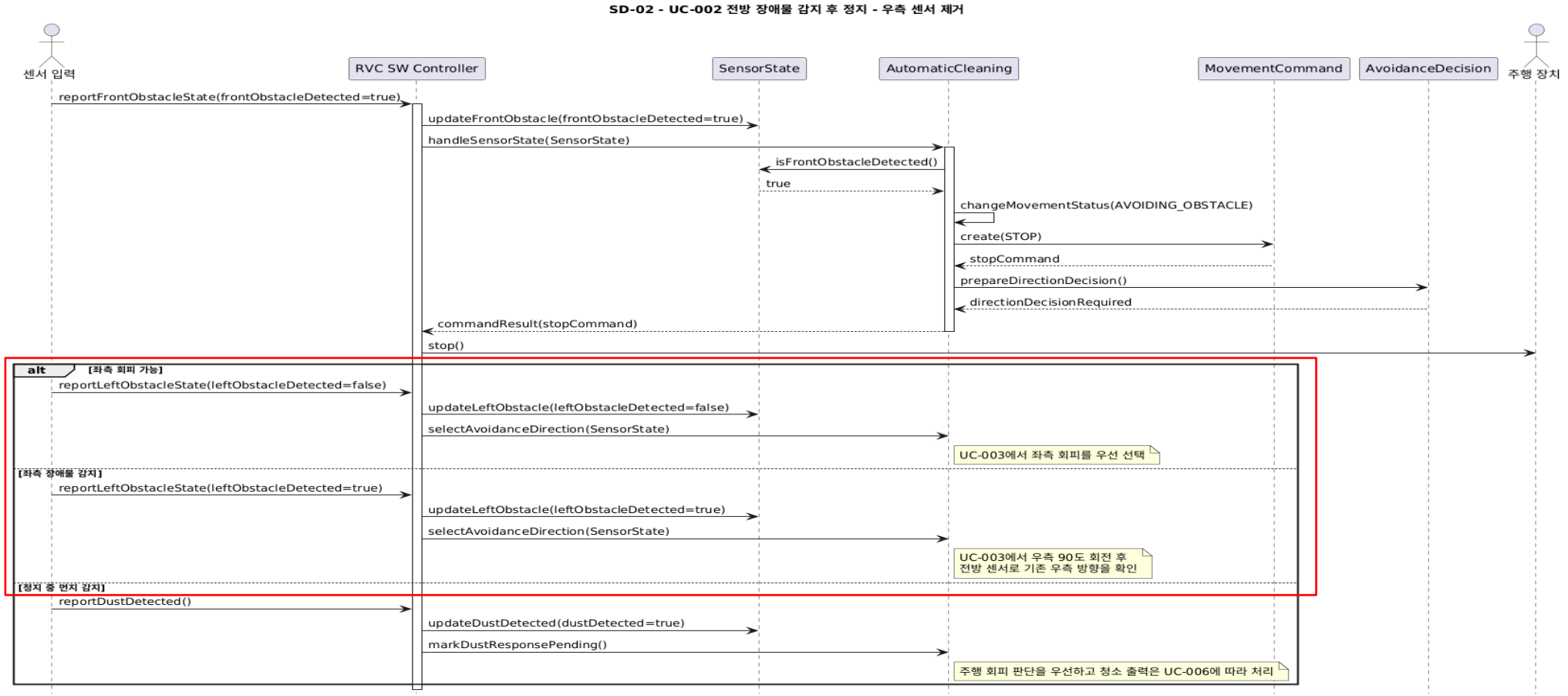
 # TEXT 2
ssd-without-right-sensor.md (2, 3, 5)

 # TEXT 3
domain-model-without-right-sensor.md (2, 3, 5)

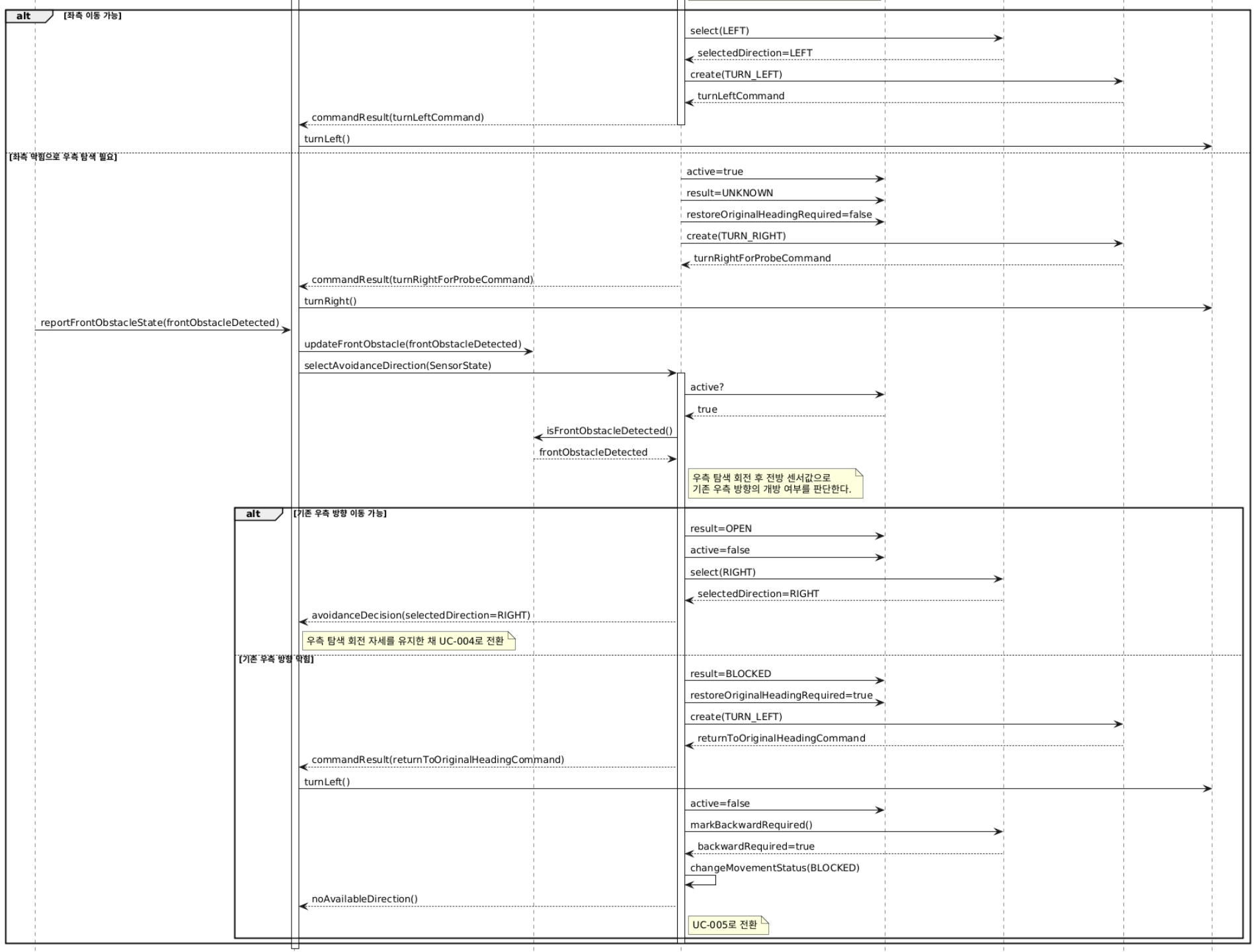
Prompt

TEXT 1과 # TEXT 2, #TEXT 3 을 바탕으로 기존 domain model 을 변경할 계획을 작성해. 주요 변경은 우측 센서 제거로 인한 탐색 과정 추가 및 좌측 우선 탐색으로의 변경이야. 수정될 필요 없는 부분은 건들지 마

전방 장애물 감지 후 정지

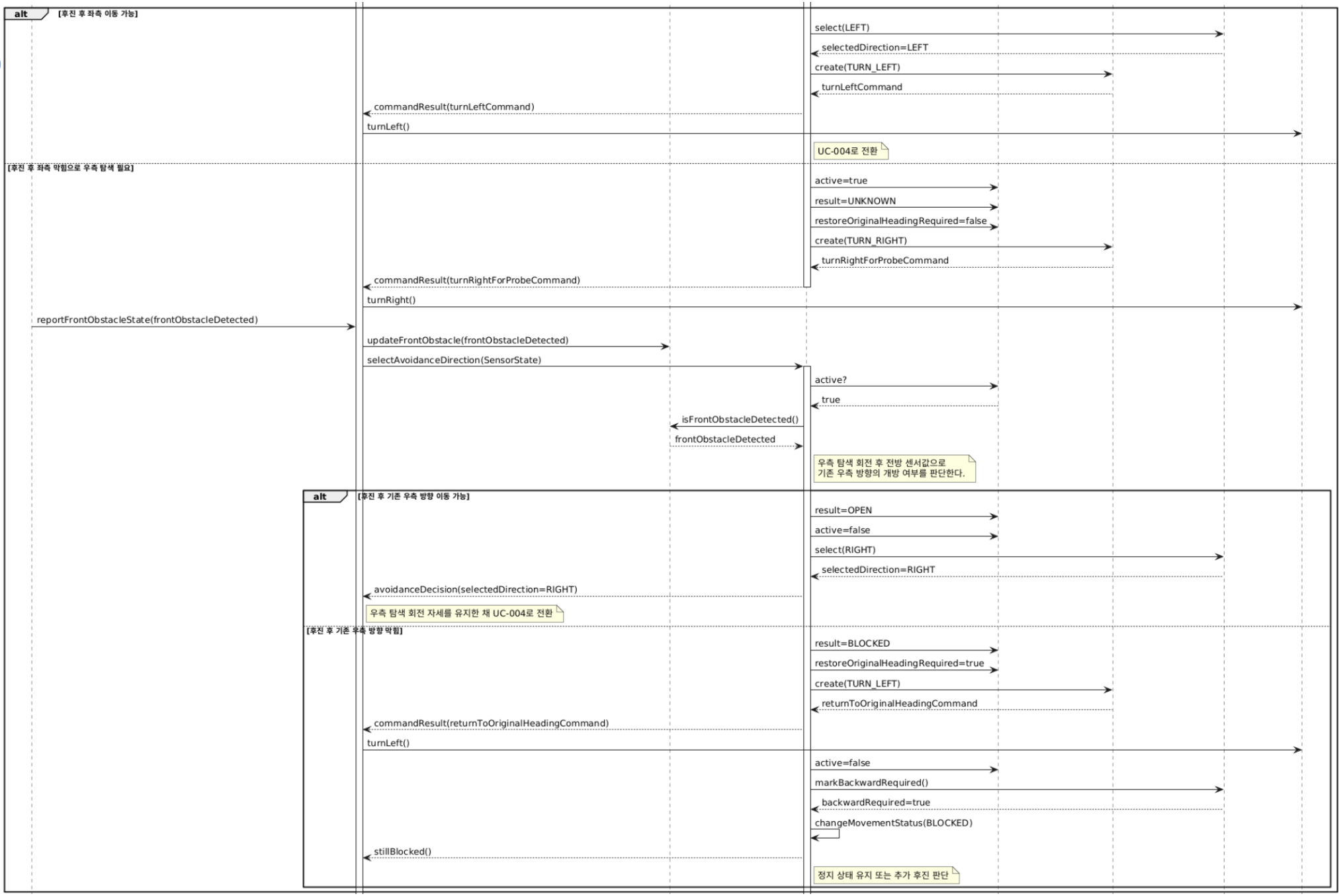

SD-02 - UC-002 전방 장애물 감지 후 정지 - 우측 센서 제거
센서 입력
RVC SW Controller
SensorState
AutomaticCleaning
MovementCommand
AvoidanceDecision
주행 장치
reportFrontObstacleState(frontObstacleDetected=true)
updateFrontObstacle(frontObstacleDetected=true)
handleSensorState(SensorState)
isFrontObstacleDetected()
true
changeMovementStatus(AVOIDING_OBSTACLE)
create(STOP)
stopCommand
prepareDirectionDecision()
directionDecisionRequired
commandResult(stopCommand)
stop()
alt
[좌측 회피 가능]
reportLeftObstacleState(leftObstacleDetected=false)
updateLeftObstacle(leftObstacleDetected=false)
selectAvoidanceDirection(SensorState)
UC-003에서 좌측 회피를 우선 선택
[좌측 장애물 감지]
reportLeftObstacleState(leftObstacleDetected=true)
updateLeftObstacle(leftObstacleDetected=true)
selectAvoidanceDirection(SensorState)
UC-003에서 우측 90도 회전 후
전방 센서로 기존 우측 방향을 확인
[정지 중 먼지 감지]
reportDustDetected()
updateDustDetected(dustDetected=true)
markDustResponsePending()
주행 회피 판단을 우선하고 청소 출력은 UC-006에 따라 처리

좌측 또는 우측 회피 방향 선택



SEQUENCE DIAGRAM

삼면 장애물 감지 후 후진



PROMPT

Class Diagram 작성 프롬프트

첨부 자료

 # TEXT 1

ssd-without-right-sensor.md (2, 3, 5)

 # TEXT 2

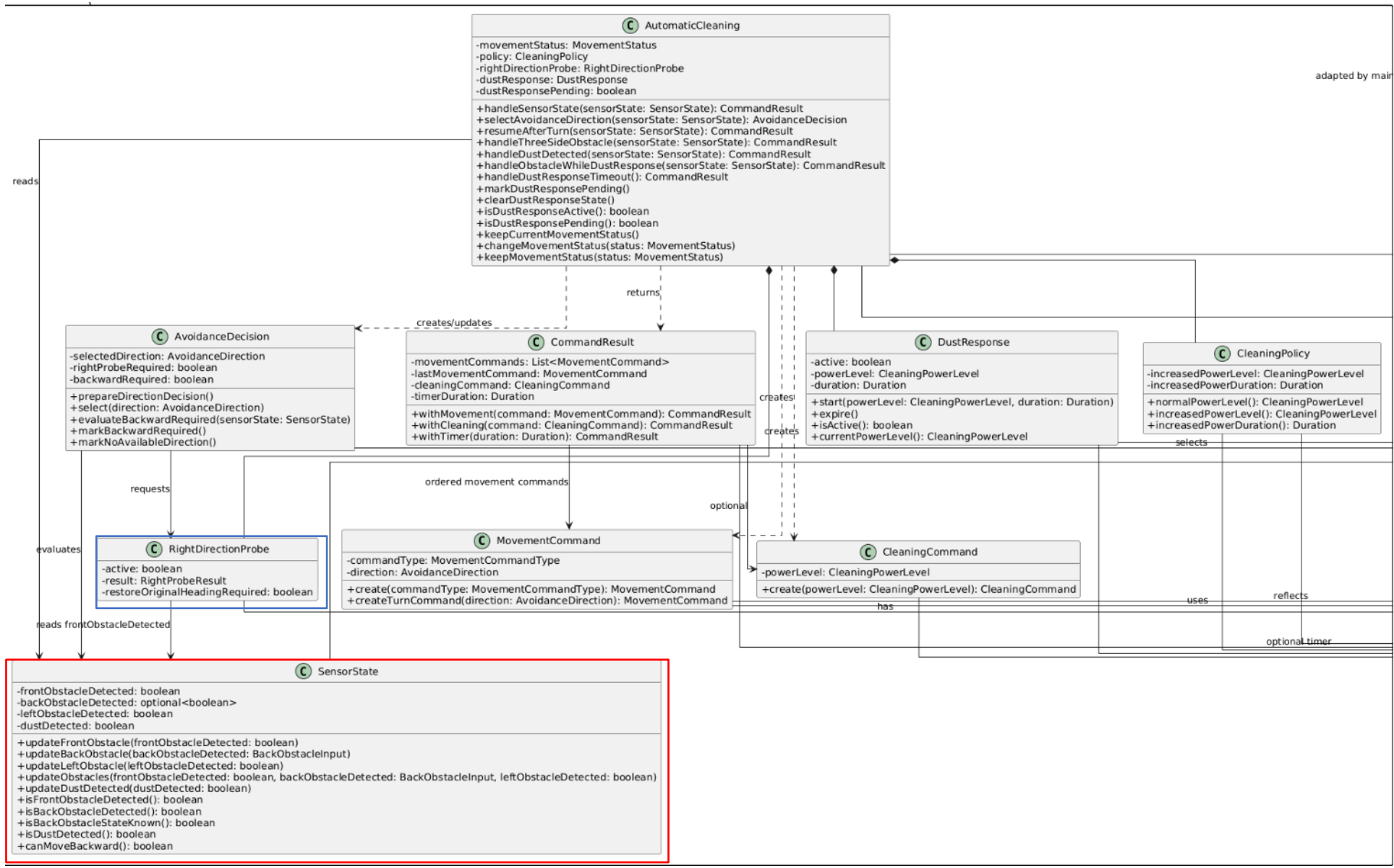
domain-model-without-right-sensor.md (2, 3, 5)

Prompt

TEXT 1과 # TEXT 2 를 바탕으로 기존 class diagram 을 변경할 계획을 작성해.
주요 변경은 우측 센서 제거로 인한 탐색 과정 추가 및 좌측 우선
탐색으로의 변경이야. 수정될 필요 없는 부분은 건들지 마

CLASS DIAGRAM

클래스 다이어그램



핵심 RightDirectionProbe 클래스 추가 및 SensorState, SensorInput 변경

No Right Sensor OOI 구현 프롬프트

RVC SW Controller · 우측 센서 제거

프롬프트

```
우측 센서 없는 상황의 use case, SSD, domain model,
SD, class diagram puml을 먼저 확인하고,
UC-001, UC-006, UC-007 등 우측 센서 무관 케이스는
기존 OOI 코드를 참고해서 진행해줘.

기존 OOI 코드를 최대한 활용해서 수정하는 방향으로
가고, 새 파일 생성은 지양해줘.

오른쪽 장애물은 직접 RIGHT 센서 입력으로 받지 않고,
좌측이 막힌 경우 우회전 90도 후 전방 센서로
기존 우측 방향의 장애물 유무를 확인하도록 구현해줘.
```

프롬프트 결과 요약

- SensorState에서 rightObstacleDetected 상태 직접 제거
- RightDirectionProbe로 우측 탐색 상태를 추적
- BackObstacleInput { Clear, Blocked, Unknown }을 core 타입으로 정리
- SensorInput 인터페이스 추가, RVCSWController가 구현
- 기존 public sensor method는 외부 입력 포트 역할 유지
- 새 소스 파일을 남발하지 않고 기존 OOI 클래스 중심으로 수정

설계 메시지

UC-001 / UC-006 / UC-007은 기존 설계 유지. UC-003 / UC-005만 변경.

구현 요약

RVC SW Controller · No Right Sensor

Controller

- RVCSWController
- SensorInput 구현
- public method = 입력 포트
- CommandResult를 apply()에서 장치 명령으로 변환

Domain Logic

- AutomaticCleaning
- RightDirectionProbe
- DustResponse
- CleaningPolicy

State / Command

- SensorState
- front / optional back / left / dust
- no direct right sensor
- MovementCommand / CleaningCommand
- CommandResult / AvoidanceDecision

External Interface

- DrivingDevice
- CleaningDevice
- Time
- SensorInput

Value Types

- BackObstacleInput
- RightProbeResult
- MovementStatus
- MovementCommandType
- CleaningPowerLevel
- AvoidanceDirection

핵심 변경 포인트

- SensorInput 인터페이스 → RVCSWController가 구현, sensor method를 외부 포트로 추상화
- RightDirectionProbe → 우측 센서 없는 환경에서 회전-확인 상태를 도메인 레벨로 추적
- BackObstacleInput { Clear / Blocked / Unknown } → 후방 상태 미확정 케이스를 타입으로 표현
- 기존 UC-001/UC-006/UC-007 클래스 변경 없이 UC-003/UC-005 로직만 조정

Unit Test 변경 요약

79개 전체 통과 · 오른쪽 센서 제거 흐름 추가

79	0	+26
Tests Passed	Failed	vs 기존 53개

변경/추가된 검증 포인트

- SET_RIGHT / 직접 right sensor 입력 대신 SET_LEFT + SET_FRONT probe 흐름 검증
- 좌측 blocked 시 TURN_RIGHT로 right probe 시작 검증
- right probe open 시 원래 방향 복귀 없이 moveForward 검증
- right probe blocked 시 TURN_LEFT로 원래 방향 복구 후 back sensor 대기 검증
- active probe 중 left-only 입력이 오래된 front 값으로 probe를 확정하지 않는 guard 검증
- BackObstacleInput::Clear / Blocked / Unknown 분기 검증
- combined snapshot reportObstacleState(front, backInput, left) 검증
- 먼지 응답 중 right probe / three-side / back sensor 흐름에서도 keepIncreased 유지 검증
- AvoidanceDecision::markRightProbeRequired() / markBackwardRequired() 등 decision branch 검증

Unit Test 상세 - UC-003 / UC-005 변경

회피 방향 선택 및 삼면 처리 로직

UC-003 · 회피 방향 선택

- 좌측 open이면 우측 probe 없이 TURN_LEFT
- 좌측 blocked이면 TURN_RIGHT로 probe 시작
- probe 후 front clear → 기존 우측 방향 open으로 판단
- probe 후 front blocked → 기존 우측 방향 blocked로 판단

UC-005 · 삼면 장애물 후방 처리

- 전방/좌측/기존 우측 방향이 모두 blocked로 확인된 뒤에만 three-side 처리
- back clear → STOP → MOVE_BACKWARD
- back blocked → STOP 후 STOPPED
- back unknown → STOP 후 BLOCKED
- 후진 이후에도 좌측 먼저, 필요 시 다시 right probe

유지된 흐름 (변경 없음)

- UC-001: 전방 clear → normal cleaning + moveForward
- UC-006: 먼지 감지 → increased power + timer
- UC-007: timeout → normal power 복귀

변경된 흐름

- UC-003: 회피 방향 선택 방식 변경
- UC-005: three-side 판단 선행조건 변경
- protocol/snapshot 입력 형태 변경
- branch coverage 보강: known back snapshot, right probe result, back unknown/blocked 분기 추가

SYSTEM TEST · 변경

System Test 변경 요약

34개 케이스 · RIGHT 제거 반영

34	22	12
Total Cases	Positive	Negative

변경된 테스트 포인트

- SET_OBSTACLES에서 RIGHT key 제거
- SET_LEFT 0|1로 좌측 센서 입력 검증
- SET_BACK 0|1|UNKNOWN으로 후방 unknown 입력 검증
- RIGHT=0 key 전송 시 ERR INVALID_ARGUMENT negative case 유지

three-side 검증 시퀀스

```
# RIGHT=1 직접 전송 없이 three-side 검증
SET_FRONT 1        # 전방 장애물
SET_LEFT 1         # 좌측 장애물 (우측 probe 유도)
SET_FRONT 1        # probe 결과: 우측도 blocked
SET_BACK 0|1|UNKNOWN  # 후방 상태 입력
```

System Test 의미

- C++ class를 직접 호출하지 않고 rvc_app 실행 파일과 TCP protocol을 검증
- simulator와 같은 외부 클라이언트 관점에서 no-right-sensor protocol 확인
- 오른쪽 센서가 없어졌다는 사실이 protocol surface에 반영됐는지 검증
- 기존 TCP runner 구조는 변경 없이 유지

변경/추가 케이스 요약

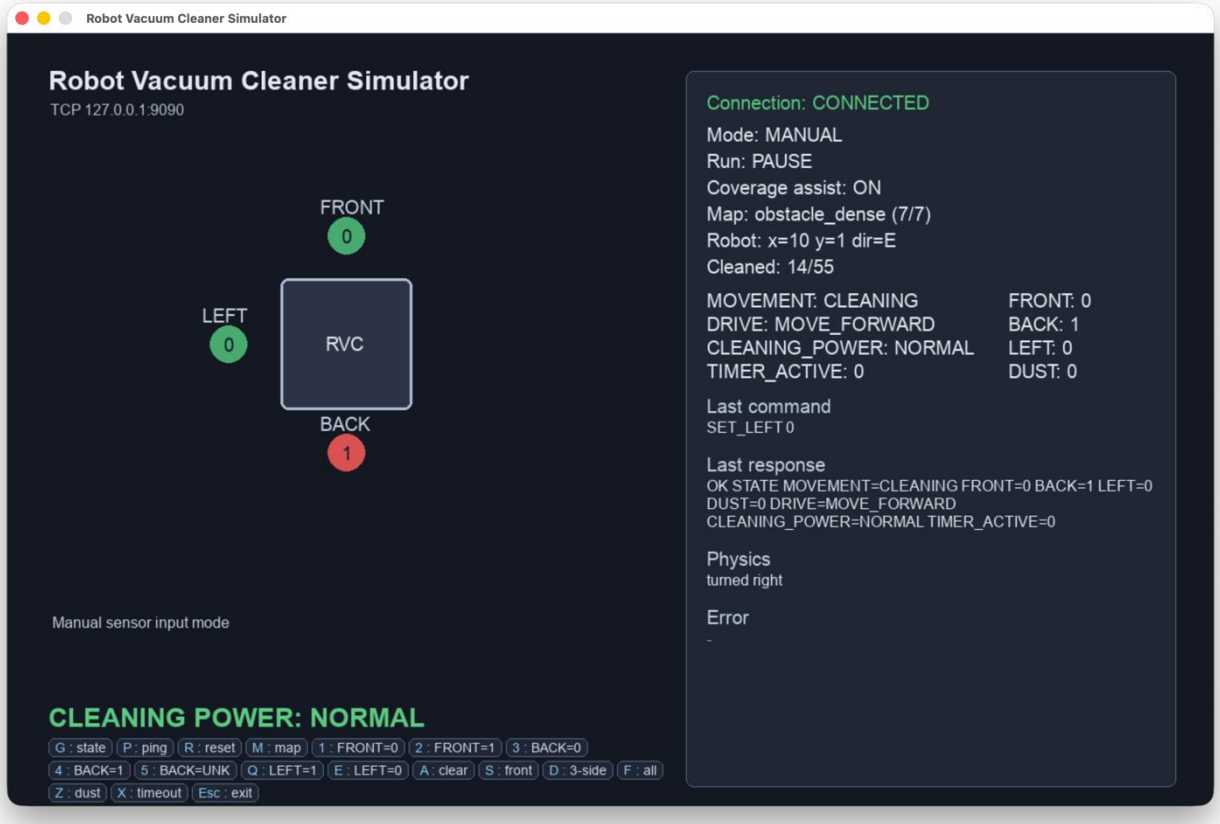
positive	SET_LEFT 0/1 검증
positive	SET_BACK UNKNOWN 검증
positive	three-side 시퀀스 검증
negative	RIGHT=0 → ERR INVALID_ARGUMENT
negative	SET_OBSTACLES에 RIGHT key → 오류

Simulator 변경 요약

RIGHT 센서 제거 · Manual Mode

변경 핵심

- Manual mode와 Map mode 모두 화면에서 RIGHT 직접 센서 입력 제거
- sensor command: FRONT / BACK / LEFT만 전송
- Map mode에서 Python world는 지도/좌표/물리/coverage assist 담당
- C++ controller는 DRIVE 값을 반환, Python이 world 이동으로 적용
- TURN_RIGHT는 우측 탐색 또는 우측 회피를 위한 주행 명령으로 표시 유지



키보드 변경 (Manual Mode)

제거: `Q: L=1/R=0 / W: L=0/R=1 / E: L=0/R=0`

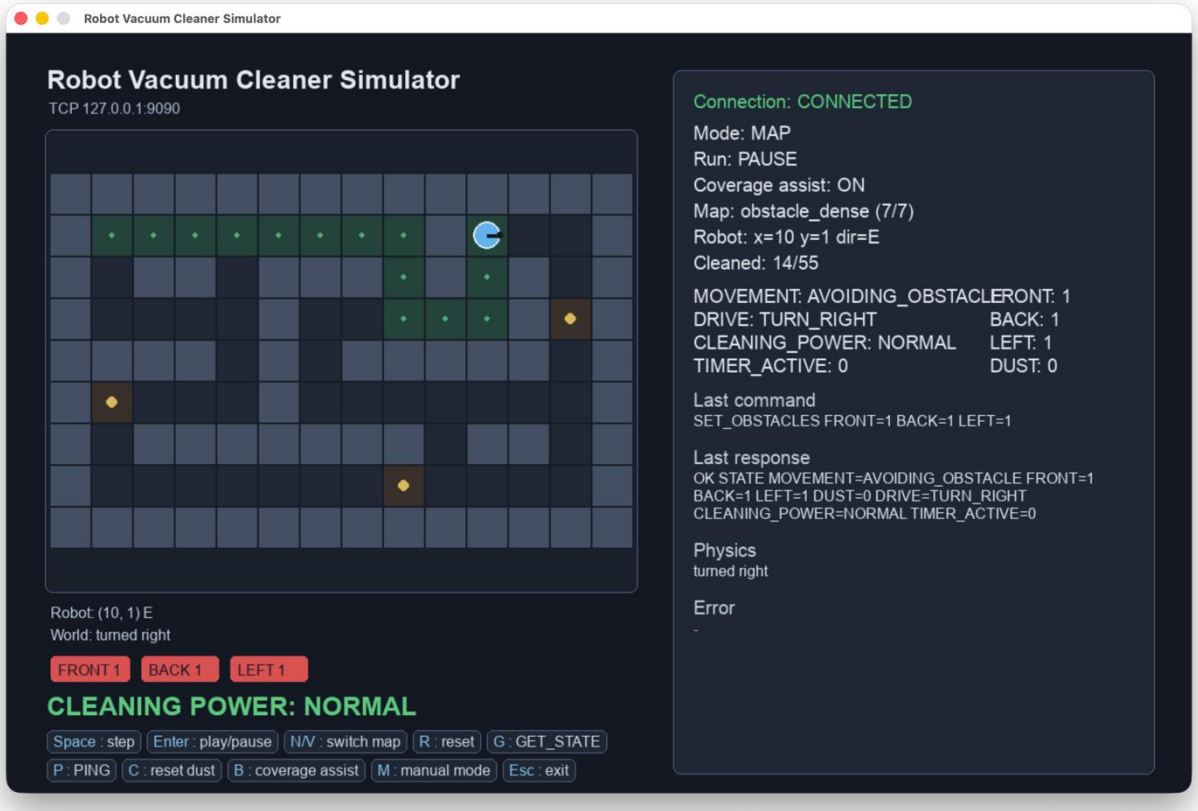
추가/유지: `Q: LEFT=1 / E: LEFT=0 / F: all`

Simulator Map Mode 결과

no-right-sensor protocol 반영 확인

Map Mode 핵심

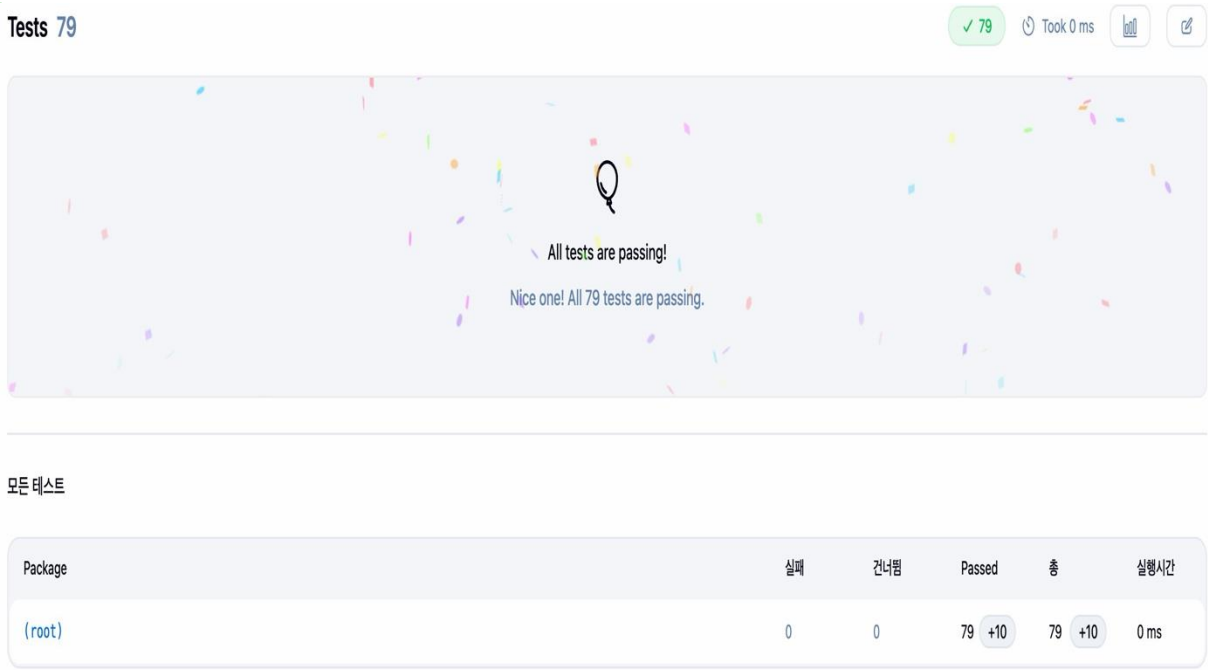
- SET_OBSTACLES FRONT=.. BACK=.. LEFT=.. (RIGHT 없음) 전송
- 상태 패널에 FRONT / BACK / LEFT / DUST만 표시
- C++ 응답의 DRIVE=TURN_RIGHT가 Python world에서 회전 동작으로 적용
- coverage assist와 built-in preset map은 Python 책임으로 유지
- obstacle_dense map에서 AVOIDING_OBSTACLE + TURN_RIGHT 동작 확인



Google Test 결과

79 tests passed · 오른쪽 센서 제거 흐름 포함

79	0	0
Tests Passed	Failed	Skipped



의 미

- 기존 53개 기준에서 우측 센서 제거 흐름과 branch coverage 보강 테스트가 추가됨
- 오른쪽 센서 제거 흐름: right probe, BackObstacleInput 분기, three-side 시퀀스
- branch coverage 보강: guard 조건, AvoidanceDecision branch, combined snapshot 검증

CI · COVERAGE

Coverage 결과

핵심 OOI 도메인 기준 · 고 branch coverage 달성

98.9%	99.1%	93.4%
Lines	Functions	Branches
453 / 458	106 / 107	268 / 287

주요 파일 coverage

파일	Lines	Func	Branches
AutomaticCleaning.cpp	98.6%	96.4%	99.0% (96/97)
RVCSWController.cpp	98.1%	100.0%	90.2% (166/184)
SensorState.cpp	100.0%	100.0%	100.0% (4/4)
Commands.cpp	100.0%	100.0%	100.0% (2/2)

개선 포인트

test code / simulator / app / protocol 영역 제외 → 핵심 OOI 코드 기준 branch coverage 크게 개선. right probe와 back unknown/blocked 분기를 실제 테스트로 검증.

GCC Code Coverage Report

Directory: ./
Date: 2026-06-04 04:07:30
Coverage: low: ≥ 0% medium: ≥ 75.0% high: ≥ 90.0%

	Exec	Total	Coverage
Lines:	453	458	98.9%
Functions:	106	107	99.1%
Branches:	268	287	93.4%

List of functions

File	Lines		Functions		Branches	
include/rvc/CleaningPolicy.hpp	100.0%	1 / 1	100.0%	1 / 1	-%	0 / 0
include/rvc/Devices.hpp	100.0%	3 / 3	100.0%	3 / 3	-%	0 / 0
include/rvc/SensorInput.hpp	100.0%	1 / 1	100.0%	1 / 1	-%	0 / 0
include/rvc/Types.hpp	100.0%	6 / 6	100.0%	6 / 6	-%	0 / 0
src/AutomaticCleaning.cpp	98.6%	146 / 148	96.4%	27 / 28	99.0%	96 / 97
src/CleaningPolicy.cpp	100.0%	9 / 9	100.0%	4 / 4	-%	0 / 0
src/Commands.cpp	100.0%	77 / 77	100.0%	27 / 27	100.0%	2 / 2
src/DustResponse.cpp	100.0%	16 / 16	100.0%	5 / 5	-%	0 / 0
src/RVCSWController.cpp	98.1%	157 / 160	100.0%	19 / 19	90.2%	166 / 184
src/SensorState.cpp	100.0%	37 / 37	100.0%	13 / 13	100.0%	4 / 4

Generated by: GCOVR (Version 7.0)

SonarQube Quality Gate 결과

Quality Gate Passed · Coverage 96.8%

Passed ✓
Quality Gate

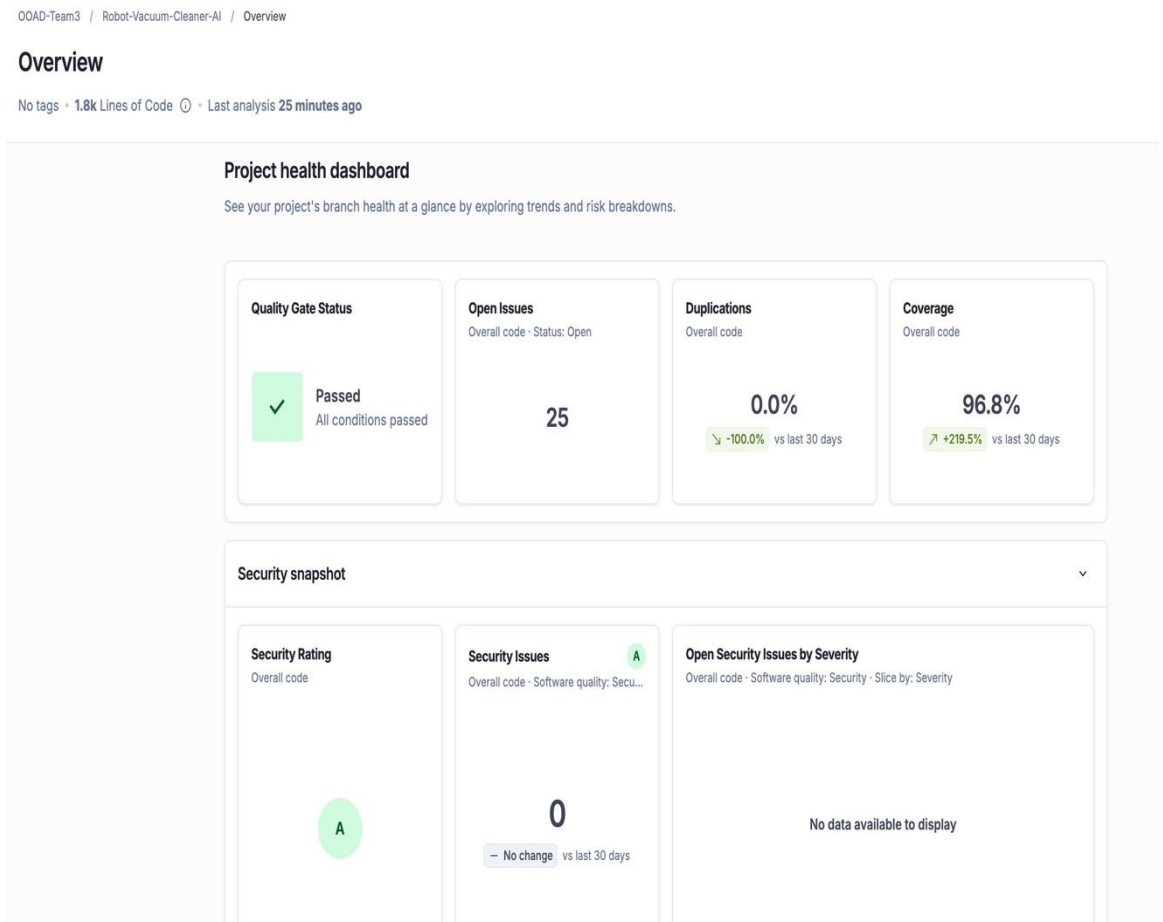
A
Security Rating

25
Open Issues

0
Security Issues

0.0%
Duplications

96.8%
Coverage



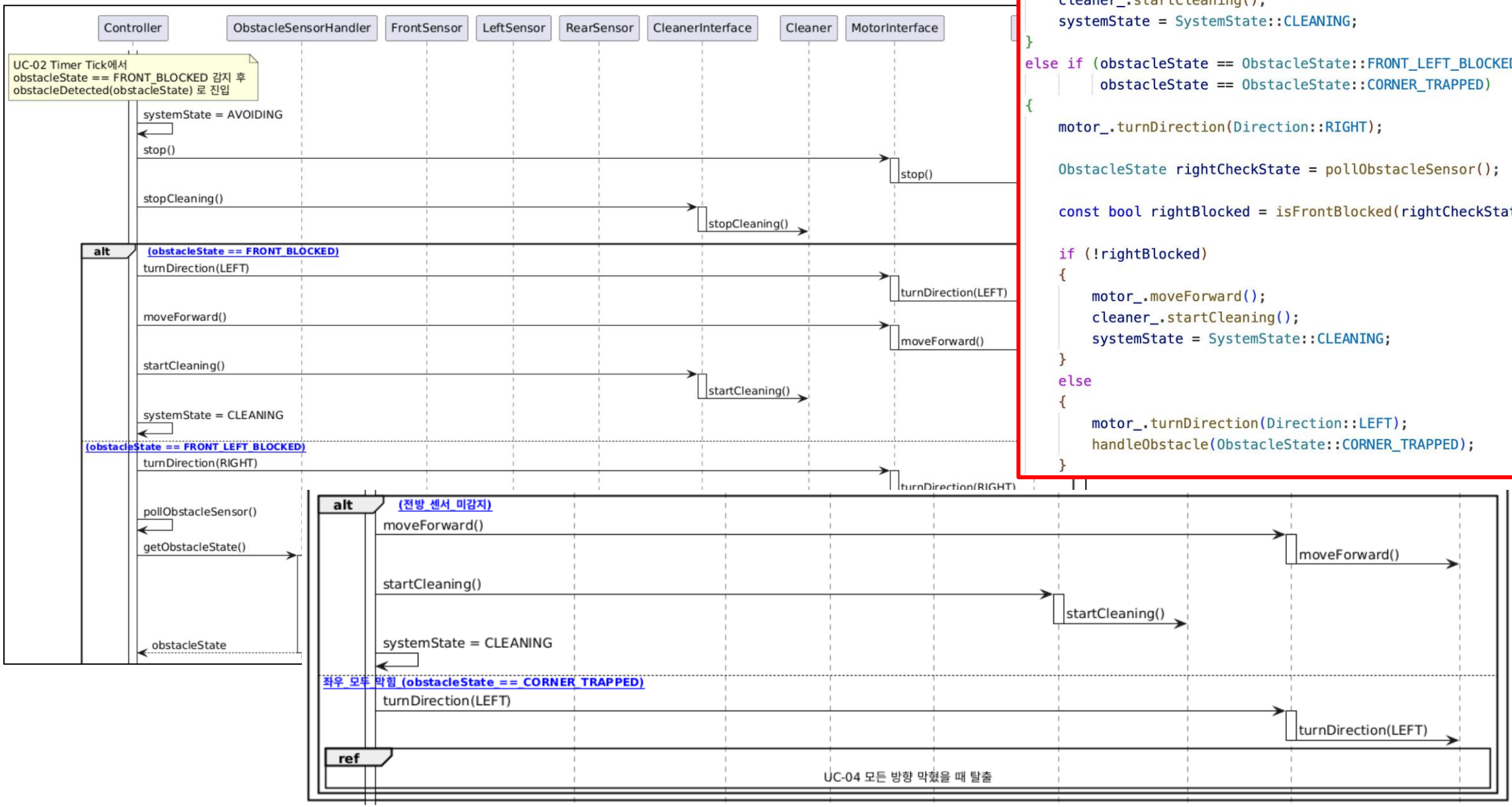
의 미

Jenkins coverage와 별도로 SonarQube에서 프로젝트 품질 추적. Quality Gate 통과로 변경 후 품질 기준 만족.

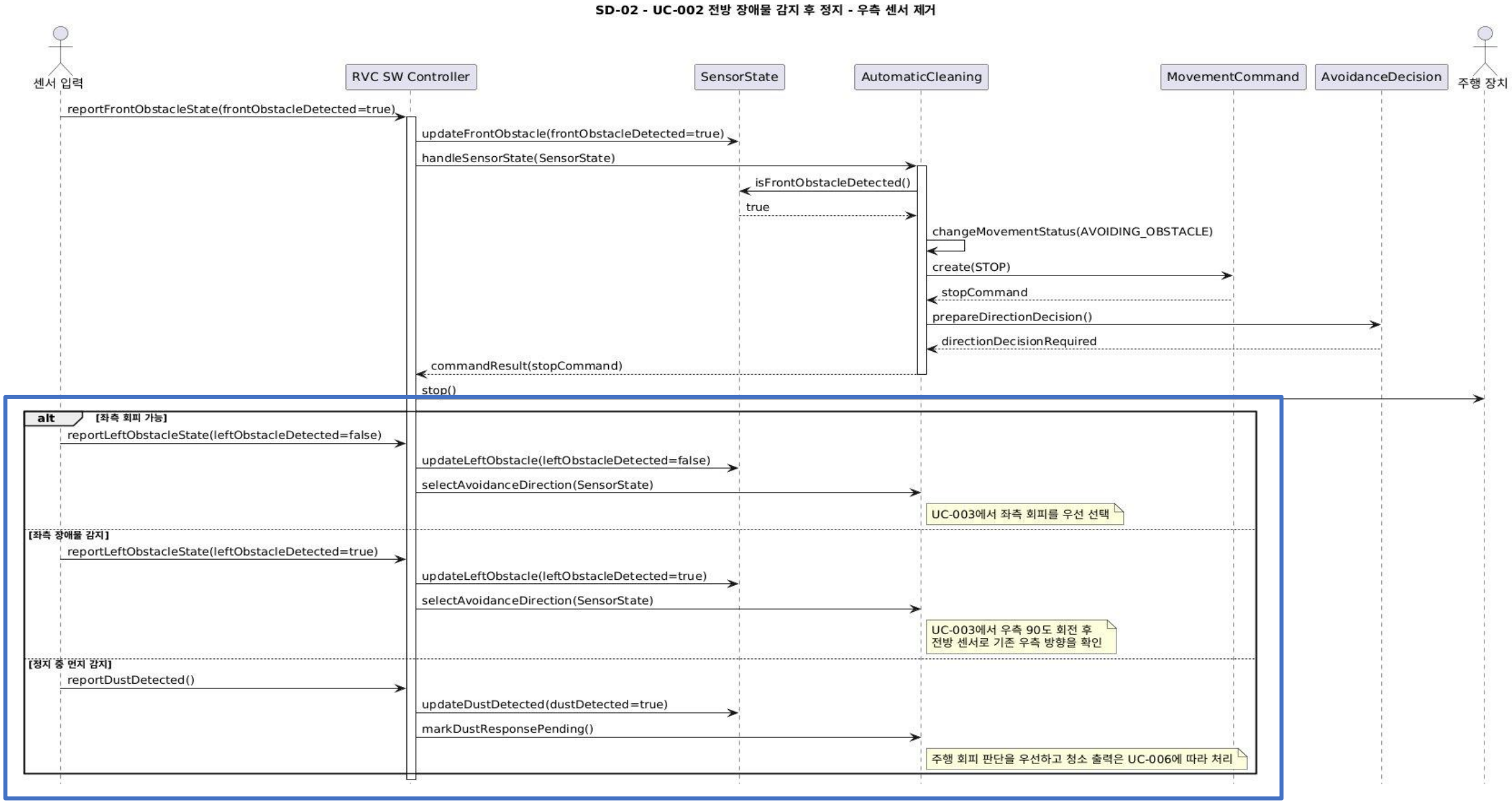
Robot Vacuum Cleaner

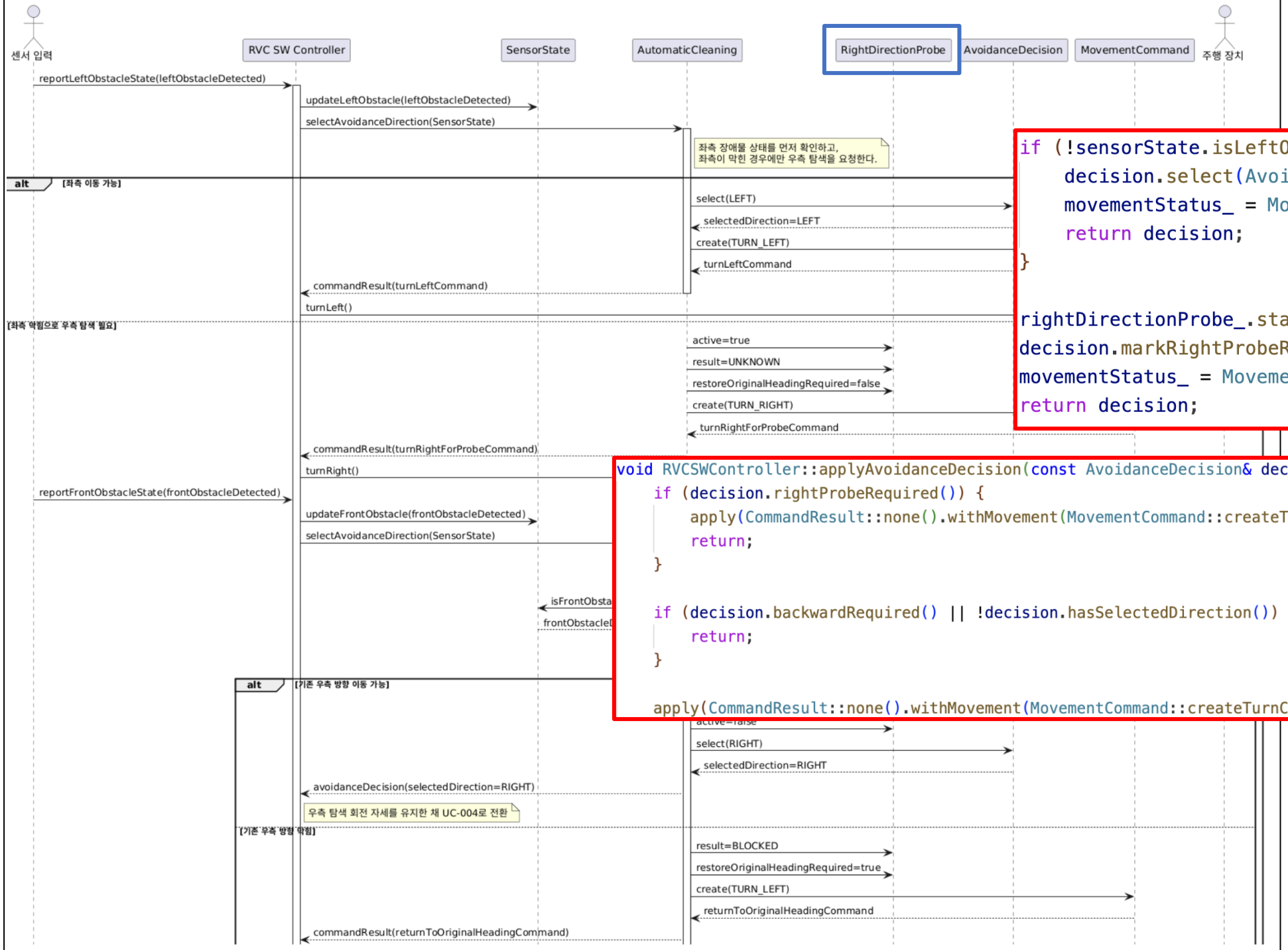
변경 사항 비교

시퀀스 다이어그램 비교




```cpp
if (obstacleState == ObstacleState::FRONT_BLOCKED)
{
    motor_.turnDirection(Direction::LEFT);
    motor_.moveForward();
    cleaner_.startCleaning();
    systemState = SystemState::CLEANING;
}
else if (obstacleState == ObstacleState::FRONT_LEFT_BLOCKED ||
         obstacleState == ObstacleState::CORNER_TRAPPED)
{
    motor_.turnDirection(Direction::RIGHT);

    ObstacleState rightCheckState = pollObstacleSensor();

    const bool rightBlocked = isFrontBlocked(rightCheckState);

    if (!rightBlocked)
    {
        motor_.moveForward();
        cleaner_.startCleaning();
        systemState = SystemState::CLEANING;
    }
    else
    {
        motor_.turnDirection(Direction::LEFT);
        handleObstacle(ObstacleState::CORNER_TRAPPED);
    }
```

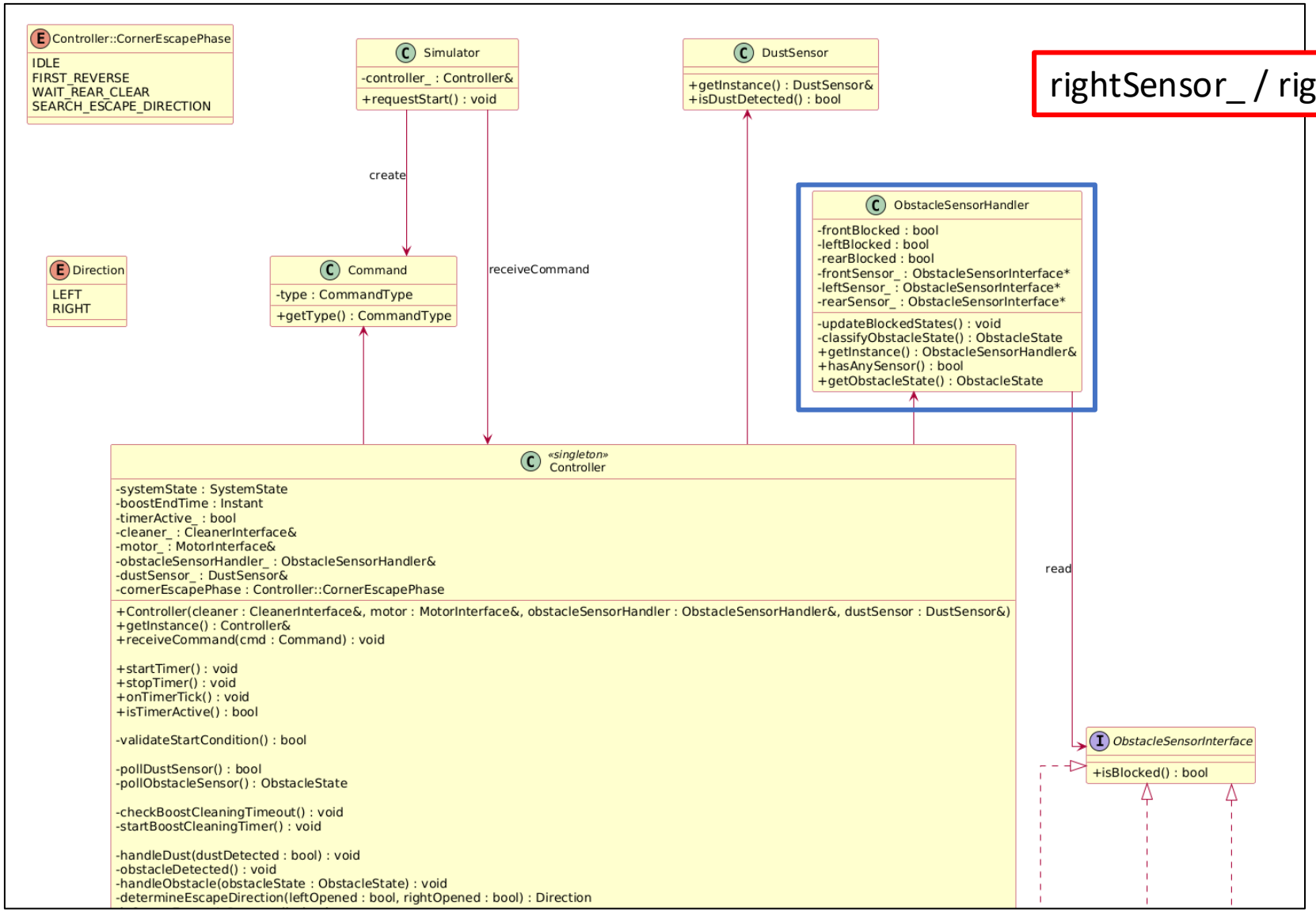

SD-02 - UC-002 전방 장애물 감지 후 정지 - 우측 센서 제거
센서 입력
RVC SW Controller
SensorState
AutomaticCleaning
MovementCommand
AvoidanceDecision
주행 장치
reportFrontObstacleState(frontObstacleDetected=true)
updateFrontObstacle(frontObstacleDetected=true)
handleSensorState(SensorState)
isFrontObstacleDetected()
true
changeMovementStatus(AVOIDING_OBSTACLE)
create(STOP)
stopCommand
prepareDirectionDecision()
directionDecisionRequired
commandResult(stopCommand)
stop()
alt
[좌측 회피 가능]
reportLeftObstacleState(leftObstacleDetected=false)
updateLeftObstacle(leftObstacleDetected=false)
selectAvoidanceDirection(SensorState)
UC-003에서 좌측 회피를 우선 선택
[좌측 장애물 감지]
reportLeftObstacleState(leftObstacleDetected=true)
updateLeftObstacle(leftObstacleDetected=true)
selectAvoidanceDirection(SensorState)
UC-003에서 우측 90도 회전 후
전방 센서로 기존 우측 방향을 확인
[청지 중 먼지 감지]
reportDustDetected()
updateDustDetected(dustDetected=true)
markDustResponsePending()
주행 회피 판단을 우선하고 청소 출력은 UC-006에 따라 처리




```
if (!sensorState.isLeftObstacleDetected()) {
    decision.select(AvoidanceDirection::Left);
    movementStatus_ = MovementStatus::AvoidingObstacle;
    return decision;
}

rightDirectionProbe_.start();
decision.markRightProbeRequired();
movementStatus_ = MovementStatus::AvoidingObstacle;
return decision;
```

```
void RVCSWController::applyAvoidanceDecision(const AvoidanceDecision& decision) {
    if (decision.rightProbeRequired()) {
        apply(CommandResult::none().withMovement(MovementCommand::createTurnCommand(AvoidanceDirection::Right)));
        return;
    }

    if (decision.backwardRequired() || !decision.hasSelectedDirection()) {
        return;
    }

    apply(CommandResult::none().withMovement(MovementCommand::createTurnCommand(*decision.selectedDirection())));
```

클래스 다이어그램 비교



rightSensor_ / rightBlocked 제거

RightDirectionProbe 클래스 추가
및
탈출 방향 판단 흐름 변경