

# OOAD:

## #6 Traditional Coding vs. Vibe Coding

202211318 엄정석 / 202211334 이동훈 / 202211384 차민우 / 202211392 최환

# Contents

## 1. Traditional Coding

- a. OOAD
- b. OOI
- c. Test & Static Analysis

## 2. Vibe Coding

- a. OOAD
- b. OOI
- c. Test & Static Analysis
- d. 비교 및 수정

# Traditional Coding

OOAD: FR/NFR

# Functional Requirements

## FR-01. RVC 전원 - 전원 ON/OFF

- 사용자가 RVC의 전원을 키면(Power ON) RVC SW를 시작한다.
- 사용자가 RVC의 전원을 끄면(Power OFF) RVC SW를 종료한다.

# Functional Requirements

## FR-02. 기본 주행 - 청소 및 직진

- 시스템은 구동 시 직진 방향으로 이동하며 흡입(clean) 및 물걸레(mop) 청소 기능을 동시에 수행해야 한다.
- 모든 주행 및 청소 시나리오에서 '장애물 감지 및 회피(FR-02, FR-03)'는 '먼지 감지 및 강화 청소(FR-04)'보다 우선한다. 회피 동작이 시작되는 즉시 진행 중인 청소 모드 및 관련 타이머는 중단 및 초기화된다.

# Functional Requirements

## FR-03. 장애물 회피 - 일반

- 시스템은 전방 센서에 장애물이 감지된 경우, 탈출 경로를 확보하기 위해 판단 결과에 따라 아래와 같이 동작한다.
  - **[조건 A]** 양쪽이 비어있는 경우:  
좌측으로 회전한 후, 전진하며 청소를 재개한다.
  - **[조건 B]** 왼쪽이 장애물로 막힌 경우:  
우측으로 회전한 후, 전진하며 청소를 재개한다.
  - **[조건 C]** 오른쪽이 장애물로 막힌 경우:  
좌측으로 회전한 후, 전진하며 청소를 재개한다.
- 모든 회전 동작은 시작 시 정해진 각도를 완전히 수행할 때까지 중단되지 않는다. 회전 동작 중에 추가적인 장애물이 감지되더라도 현재의 회전 동작을 끝까지 완료한 후 다음 상태(전진 또는 재판단)로 전이한다.

# Functional Requirements

## FR-04. 장애물 회피 - 3면 고립 상황 탈출

- 시스템은 전방, 좌측, 우측 센서에 모두 장애물이 감지되어 고립된 경우, 탈출 경로를 확보하기 위해 다음의 절차를 수행해야 한다.
  - a. 초기 회피 (후진): 3면 장애물 감지 즉시 3초 동안 후진한다.
  - b. 측면 공간 탐색: 3초 후진 완료 후, 좌측 및 우측 센서를 확인하여 탈출 가능 여부를 판단한다.
  - c. 조건별 탈출 동작: 판단 결과에 따라 아래와 같이 동작한다.
    - **[조건 A]** 양쪽 모두 빈 공간인 경우:  
~~좌측으로 회전한 후, 전진하며 청소를 재개한다.~~  
후진 전 전면 기준 좌측으로 90° 회전한 후, 전진하며 청소를 재개한다.
    - **[조건 B]** 한쪽만 빈 공간인 경우:  
~~장애물이 없는 방향으로 회전한 후, 전진하며 청소를 재개한다.~~  
장애물이 없는 방향으로 후진 전 전면 기준으로 90° 회전한 후, 전진하며 청소를 재개한다.
    - **[조건 C]** 양쪽 모두 장애물로 막힌 경우:  
최대 10회까지 NFR-04의 안전 절차에 따라 위 절차를 반복한다.

# Functional Requirements

## FR-05. 먼지 감지 및 강화 청소 - Power Up 모드

- 시스템은 먼지 감지 시 즉시 흡입력을 강화(Power Up 모드)하며, 감지 시점으로부터 정확히 5초간 해당 상태를 유지한 후 일반 상태로 자동 전환해야 한다.
- Power Up 모드 유지 중 먼지가 재감지될 경우, 진행 중인 타이머를 유지한다.

# Non-Functional Requirements

## NFR-01. 반응 및 회전 시간

장애물 감지 후 판단 지연 시간은 0.5초 이내여야 하며,  
회전 명령 수행 시 0.5초 이내에 제자리에서 130도 90도 회전을 완료해야 한다.  
시간은 1초 이내여야 한다.

## NFR-02. 타이머 정밀도

Power Up 모드 유지 시간(5초)의 오차 범위는 0.1초 이내여야 한다.

## NFR-03. 충돌 방지

장애물 감지 센서의 정확도는 99% 이상이어야 하며,  
물리적 충격을 최소화하도록 제어되어야 한다.

# Non-Functional Requirements

## NFR-04. 후진 동작 제약 및 시스템 보호

시스템은 고립 상황 탈출을 위해 반복적인 후진을 수행할 수 있으나, 후방 안전 확보를 위해 1회 후진 시간은 3초를 초과할 수 없으며, 각 후진 주기 사이에는 센서 재스캔을 위해 0.5초의 정지 시간을 가져야 한다. 또한, 총 10회 이상의 반복 후진 후에도 경로 미확보 시 시스템은 안전을 위해 에러를 출력하며 동작을 정지한다.

## NFR-05. 하드웨어 독립성

제어 소프트웨어(RVC SW)는 특정 모터 드라이버나 센서 모델에 종속되지 않도록 추상화된 인터페이스를 사용해야 한다.

## NFR-06. 범위 제한

본 설계는 '자동 청소 기능'에 한정하며, 수동 조작이나 배터리 관리 루틴은 포함하지 않는다.

# Traditional Coding

OOAD: Use-Case

<b>Use Case</b>	UC-01. 전원 켜기
<b>Actor</b>	User
<b>Purpose</b>	RVC를 시작한다.
<b>Overview</b>	RVC의 전원이 꺼져있는 상태에서 User가 RVC의 전원을 켤다.
<b>Type</b>	Primary & Essential
<b>Cross Reference</b>	Functional Requirement FR-01 Use-Case UC-03
<b>Pre-Requests</b>	시스템이 현재 전원 OFF상태여야 한다.
<b>Typical Courses of Events</b>	<ol style="list-style-type: none"><li><b>1. Actor:</b> RVC의 전원 버튼을 누른다.</li><li><b>2. System:</b> HW에 SW를 로딩한다.</li><li><b>3. System:</b> UC-03으로 이동한다.</li></ol>
<b>Alternative Courses of Events</b>	-
<b>Exceptional Courses of Events</b>	-

<b>Use Case</b>	UC-02. 전원 끄기
<b>Actor</b>	User
<b>Purpose</b>	RVC를 종료한다.
<b>Overview</b>	RVC의 전원이 켜져있는 상태에서 User가 RVC의 전원을 끈다.
<b>Type</b>	Primary & Essential
<b>Cross Reference</b>	Functional Requirement FR-01
<b>Pre-Requests</b>	RVC의 전원이 ON상태여야 한다.
<b>Typical Courses of Events</b>	<ol style="list-style-type: none"><li><b>1. Actor:</b> RVC의 전원 버튼을 누른다.</li><li><b>2. System:</b> 청소를 멈춘다.</li><li><b>3. System:</b> 전원을 종료한다.</li></ol>
<b>Alternative Courses of Events</b>	-
<b>Exceptional Courses of Events</b>	Line 1~2 : RVC SW의 전원을 꺼도 청소를 멈추지 않으면 유저에게 error메시지를 보여준다. Line 2~3 : 청소를 멈춰도 전원이 꺼지지 않으면 유저에게 error메시지를 보여준다.

<b>Use Case</b>	UC-03. 기본 주행
<b>Actor</b>	RVC
<b>Purpose</b>	RVC가 바닥의 모든 구역 청소를 진행한다.
<b>Overview</b>	RVC가 직진하며 청소를 진행한다.
<b>Type</b>	Primary & Essential
<b>Cross Reference</b>	Function Requirement FR-02 Use-Case UC-04, UC-05, UC-06, UC-08
<b>Pre-Requests</b>	시스템 현재 전원 ON상태여야 한다.
<b>Typical Courses of Events</b>	<ol style="list-style-type: none"> <li><b>System</b> : 바퀴 모터를 가동해서 전진을 시작한다.</li> <li><b>System</b> : 클리너 모터를 가동해서 청소를 시작한다.</li> </ol>
<b>Alternative Courses of Events</b>	<p>Line 1 : (System) 전면과 좌측 장애물 감지 센서를 통해 전방에 장애물을 탐지할 경우 UC-4를 실행한다.</p> <p>Line 1 : (System) 전면과 좌측 장애물 감지 센서를 통해 전방 및 좌측에 장애물을 탐지할 경우 UC-5를 실행한다.</p> <p>Line 1 : (System) 전면과 좌측 장애물 감지 센서를 통해 전방 및 좌측, 우측에 장애물을 탐지할 경우 UC-6을 실행한다.</p> <p>Line 2 : (System) 전면과 좌측 장애물 감지 센서를 통해 먼지를 감지할 경우 UC-8을 실행한다.</p>
<b>Exceptional Courses of Events</b>	Line 1~2 : 기본 주행 명령을 수신하고도 Moter 또는 Cleaner를 가동시키지 않으면 유저에게 error메시지를 보여준다.

<b>Use Case</b>	UC-04. 장애물 감지 - 좌회전
<b>Actor</b>	RVC
<b>Purpose</b>	RVC가 기본 주행 상태일 때 전면 혹은 전면과 우측에 장애물이 있다면 좌회전하여 회피한다.
<b>Overview</b>	<ul style="list-style-type: none"> <li>• RVC가 직진하며 전면과 좌측 장애물 감지 센서를 통해 장애물을 감지한다.</li> <li>• 전면 장애물을 감지하면, RVC는 기본 주행을 멈추고 좌측으로 회전하여 장애물을 피해 경로를 확보한다. 회전 완료 이후 기본 주행을 재개한다.</li> <li>• 회전 중 추가 장애물이 감지되더라도 중단 없이 회전을 완수한다.</li> </ul>
<b>Type</b>	Primary & Essential
<b>Cross Reference</b>	Function Requirement FR-03 Use-Case UC-03
<b>Pre-Requests</b>	시스템이 현재 기본 주행 상태여야한다.
<b>Typical Courses of Events</b>	<ol style="list-style-type: none"> <li><b>1. System:</b> 전면 장애물 감지 신호를 수신한다.</li> <li><b>2. System:</b> 전면에 장애물을 감지하여 청소를 중단한다.</li> <li><b>3. System:</b> 제자리에서 130도 좌회전 한다.</li> <li><b>4. System:</b> UC-03으로 이동한다.</li> </ol>
<b>Alternative Courses of Events</b>	-
<b>Exceptional Courses of Events</b>	Line 1~3 : 장애물 감지 신호를 수신하고도 모터를 가동시키지 않으면 유저에게 error메시지를 보여준다.

<b>Use Case</b>	UC-05. 장애물 감지 - 우회전
<b>Actor</b>	RVC
<b>Purpose</b>	RVC가 기본 주행 상태일 때 전면 혹은 전면과 좌측에 장애물이 있다면 우회전하여 회피한다.
<b>Overview</b>	<ul style="list-style-type: none"> <li>• RVC가 직진하며 전면과 좌측 장애물 감지 센서를 통해 장애물을 감지한다.</li> <li>• 전면과 좌측에 장애물이 감지되면, RVC는 기본 주행을 멈추고 우측으로 회전하여 전면 센서를 통해 우측 장애물 여부를 판단한다.</li> <li>• 장애물이 감지되지 않으면 기본 주행을 재개한다.</li> <li>• 회전 중 추가 장애물이 감지되더라도 중단 없이 회전을 완수한다.</li> </ul>
<b>Type</b>	Primary & Essential
<b>Cross Reference</b>	Function Requirement FR-03 Use-Case UC-03, UC-06
<b>Pre-Requests</b>	시스템이 현재 기본 주행 상태여야한다.
<b>Typical Courses of Events</b>	<ol style="list-style-type: none"> <li><b>1. System:</b> 전면 및 좌측 센서로부터 장애물 감지 신호를 수신한다.</li> <li><b>2. System :</b> 제자리에서 90도 우회전 한다.</li> <li><b>3. System:</b> 전면 센서를 통해 우측 장애물 감지 신호를 수신한다.</li> <li><b>4. System:</b> UC-03으로 이동한다.</li> </ol>
<b>Alternative Courses of Events</b>	Line 3 : (System) 우측 장애물이 있을 시 좌측으로 90도 회전 후 UC-06 - Line 5 로 이동한다
<b>Exceptional Courses of Events</b>	Line 1~3 : 장애물 감지 신호를 수신하고도 모터를 가동시키지 않으면 유저에게 error메시지를 보여준다.

<b>Use Case</b>	UC-06. 장애물 감지 - 후진
<b>Actor</b>	RVC
<b>Purpose</b>	RVC가 기본 주행 상태일 때 3면(전면/좌측/우측)에 장애물이 있다면 후진하여 회피한다.
<b>Overview</b>	<ul style="list-style-type: none"> <li>• RVC가 직진하며 전면과 좌측의 장애물 감지 센서를 통해 3면의 장애물을 감지한다.</li> <li>• 전면과 좌측 센서가 모두 3면의 장애물을 감지하면, RVC는 기본 주행을 멈추고 3초간 후진한다.</li> <li>• 3초 후진 완료 후, 좌측의 장애물 감지 센서를 확인하여 탈출 가능 여부를 파악한다. 판단 결과에 따라 FR-04.c와 같이 동작한다.</li> </ul>
<b>Type</b>	Primary & Essential
<b>Cross Reference</b>	Function Requirement FR-04 Use-Case UC-03, UC-04, UC-07
<b>Pre-Requests</b>	시스템이 현재 기본 주행 상태여야한다.
<b>Typical Courses of Events</b>	<ol style="list-style-type: none"> <li>1. <b>System:</b> 전면, 좌측로부터 장애물 감지 신호를 수신한다.</li> <li>2. <b>System:</b> 제자리에서 90도 우회전 한다.</li> <li>3. <b>System:</b> 전면 센서를 통해 우측 장애물 감지 신호를 수신한다.</li> <li>4. <b>System:</b> 제자리에서 90도 좌회전 한다.</li> <li>5. <b>System:</b> 뒤로 3초간 후진한다.</li> <li>6. <b>System:</b> 양쪽에 장애물이 없는 경우 좌회전한다.</li> <li>7. <b>System:</b> UC-03으로 이동한다.</li> </ol>
<b>Alternative Courses of Events</b>	<p>Line 6 : (System) 좌측에만 장애물이 있는 경우 UC-05로 이동한다.</p> <p>Line 6 : (System) 양쪽에 장애물이 있는 경우, Line 5로 이동한다.</p> <p>Line 6 : (System) 연속으로 10번 반복한 후에도 회전을 통한 경로를 확보하지 못한 경우 UC-07로 이동한다.</p>
<b>Exceptional Courses of Events</b>	Line 1~3 : 장애물 감지 신호를 수신하고도 모터를 가동시키지 않으면 유저에게 error메시지를 보여준다.

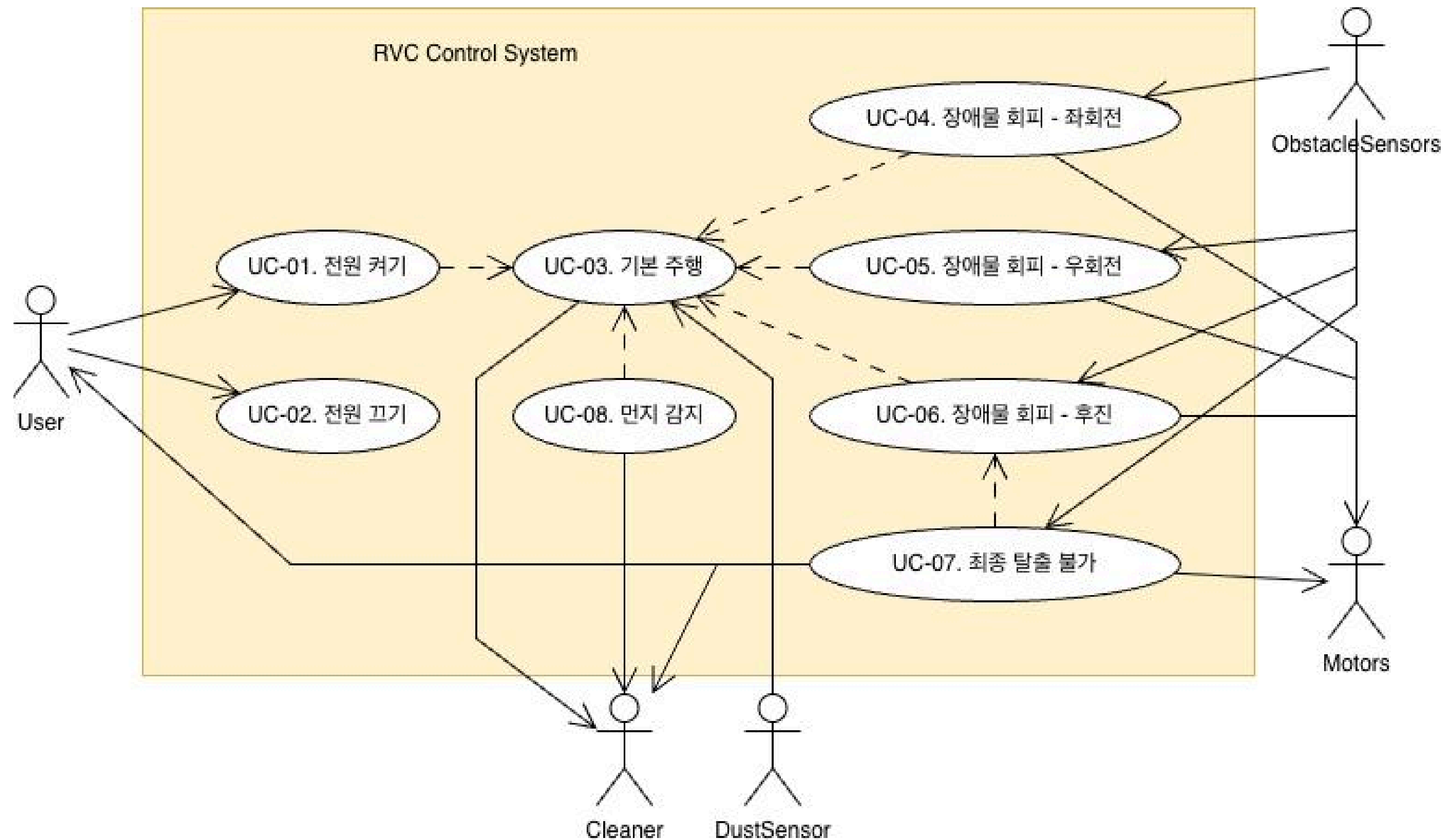
<b>Use Case</b>	UC-07. 최종 탈출 불가
<b>Actor</b>	RVC
<b>Purpose</b>	RVC가 10번 후진한 후에도 좌측 및 우측에 장애물이 있다면 시스템을 종료한다.
<b>Overview</b>	<ul style="list-style-type: none"> <li>RVC가 '4. 장애물 감지 - 후진' 상황을 연속으로 10번 반복한 후에도 회전을 통한 경로를 확보하지 못한 경우 하드웨어 과부하 및 무한 루프 방지를 위해 시스템을 정지한다. 이후 사용자에게 에러 메시지를 출력하고 시스템을 종료한다.</li> </ul>
<b>Type</b>	Secondary & Essential
<b>Cross Reference</b>	Function Requirement FR-04 Use-Case UC-06
<b>Pre-Requests</b>	시스템이 전면, 좌측 장애물 감지 센서로부터 장애물 감지 신호를 수신한 상태여야 한다.
<b>Typical Courses of Events</b>	<ol style="list-style-type: none"> <li><b>System</b> : RVC는 후진 후 좌측 장애물 감지 센서로부터 감지 신호를 수신한다.</li> <li><b>System</b> : 제자리에서 90도 우회전 한다.</li> <li><b>System</b> : 전면 센서를 통해 우측 장애물 감지 신호를 수신한다.</li> <li><b>System</b> : 제자리에서 90도 좌회전 한다.</li> <li><b>System</b> : 좌측 및 우측에 장애물을 감지하여 후진을 수행한다.</li> <li><b>System</b> : 10번까지 반복한다.</li> <li><b>System</b> : 좌측 장애물 감지 센서로부터 감지 신호를 수신한다.</li> <li><b>System</b> : 제자리에서 90도 우회전 한다.</li> <li><b>System</b> : 전면 센서를 통해 우측 장애물 감지 신호를 수신한다.</li> <li><b>System</b> : 제자리에서 90도 좌회전 한다.</li> <li><b>System</b> : 시스템을 정지한다.</li> </ol>
<b>Alternative Courses of Events</b>	Line 1~11 : (System) 좌측에 장애물이 없는 경우 UC-04로 이동한다. Line 1~11 : (System) 우측에 장애물이 없는 경우 UC-05로 이동한다. Line 1~11 : (System) 좌측, 우측에 장애물이 없는 경우 UC-04로 이동한다.
<b>Exceptional Courses of Events</b>	-

<b>Use Case</b>	UC-08. 먼지 탐지
<b>Actor</b>	RVC
<b>Purpose</b>	RVC가 청소 중 먼지를 감지하면 Cleaner의 출력을 높인다.
<b>Overview</b>	<ul style="list-style-type: none"> <li>• RVC가 청소 중 먼지 감지 센서를 통해 바닥의 먼지를 감지한다.</li> <li>• 먼지가 감지되면 Power Up 모드를 5초간 실행한 후 청소 동작으로 복귀한다.</li> </ul>
<b>Type</b>	Primary & Essential
<b>Cross Reference</b>	Functional Requirement FR-05 Use-Case UC-03
<b>Pre-Requests</b>	시스템이 현재 기본 주행 상태여야한다.
<b>Typical Courses of Events</b>	<ol style="list-style-type: none"> <li><b>1. System:</b> 기본 주행 중 먼지 감지 센서에서 먼지 감지 신호를 수신한다.</li> <li><b>2. System:</b> 5초간 Cleaner의 출력을 높인다.</li> </ol>
<b>Alternative Courses of Events</b>	-
<b>Exceptional Courses of Events</b>	Line 1~2 : 먼지 감지 신호를 수신하고도 클리너 출력을 높이지 않으면 유저에게 error메시지를 보여준다.

# **Traditional Coding**

OOAD: Use-Case Diagram

# Use-Cases Diagram

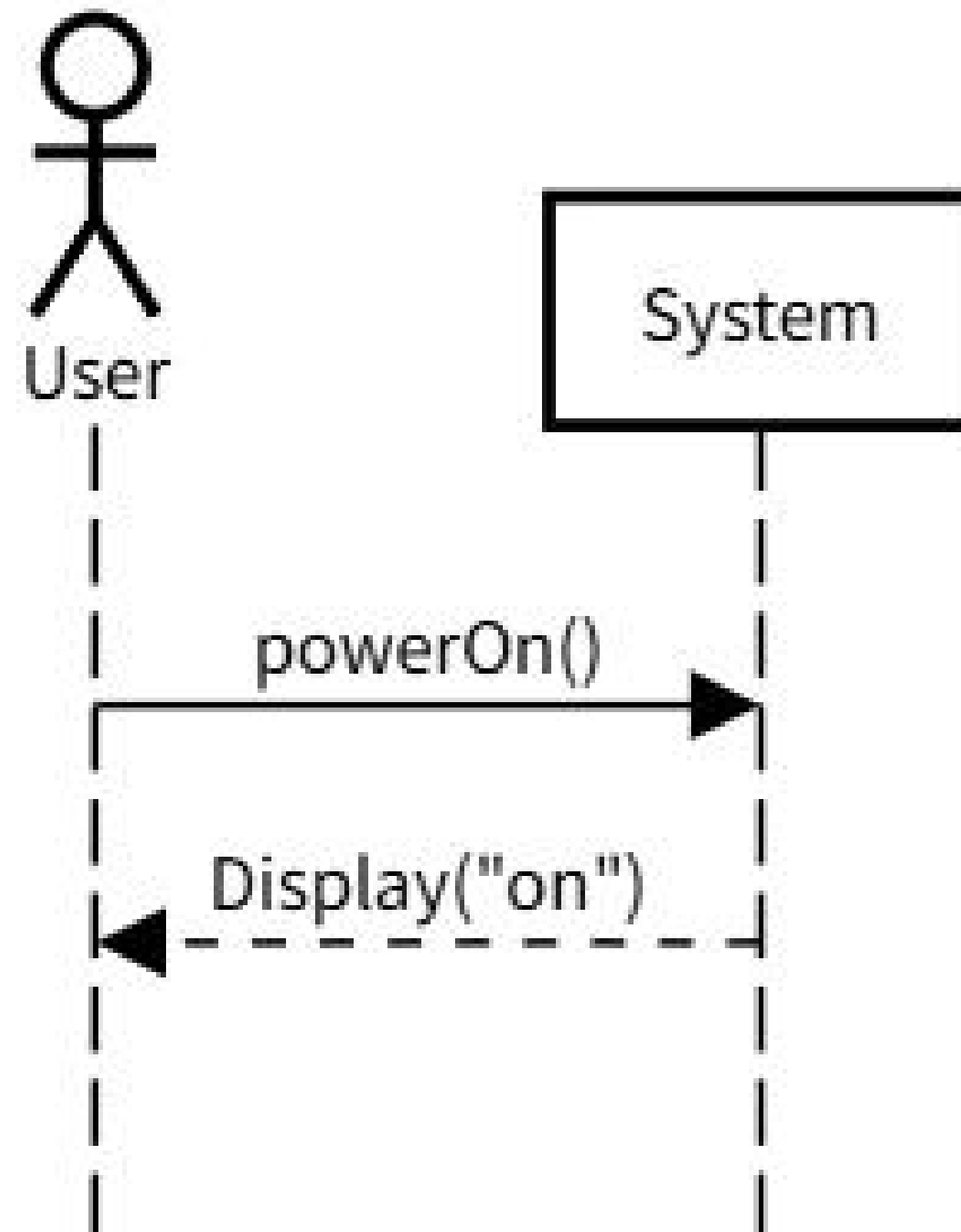


# **Traditional Coding**

OOAD: System Sequence Diagram

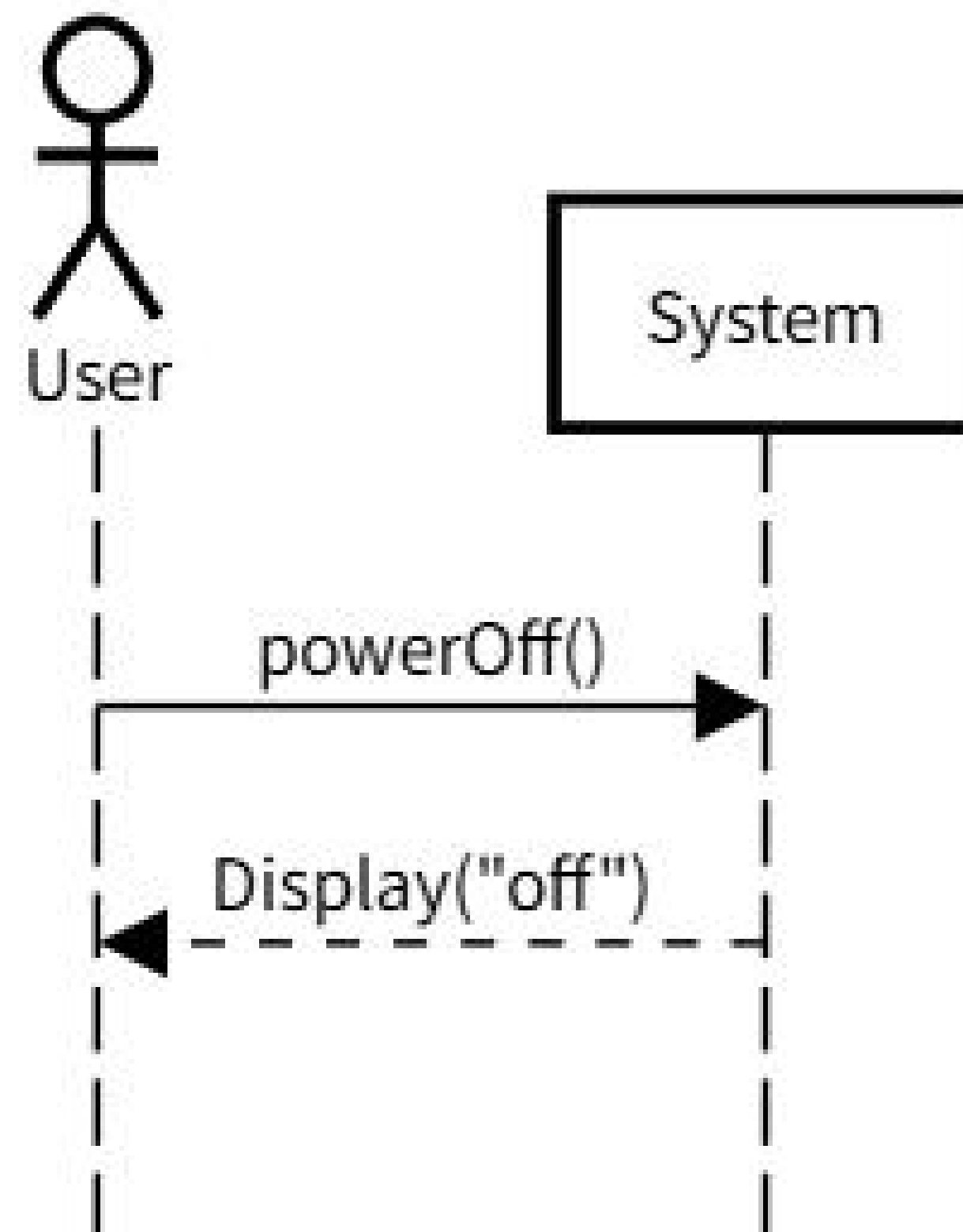
# System Sequence Diagrams: UC-01

전원 켜기

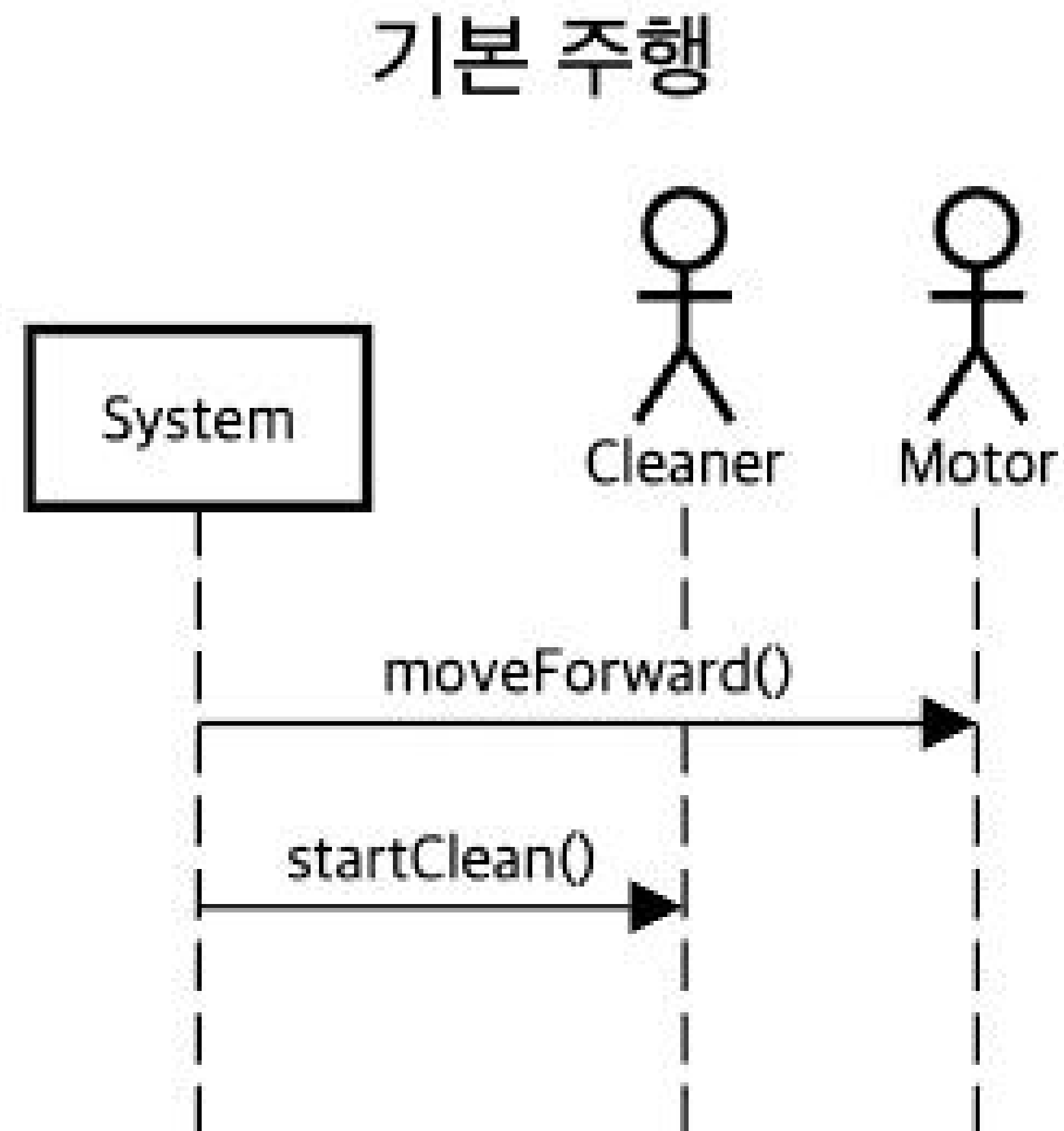


# System Sequence Diagrams: UC-02

전원 끄기

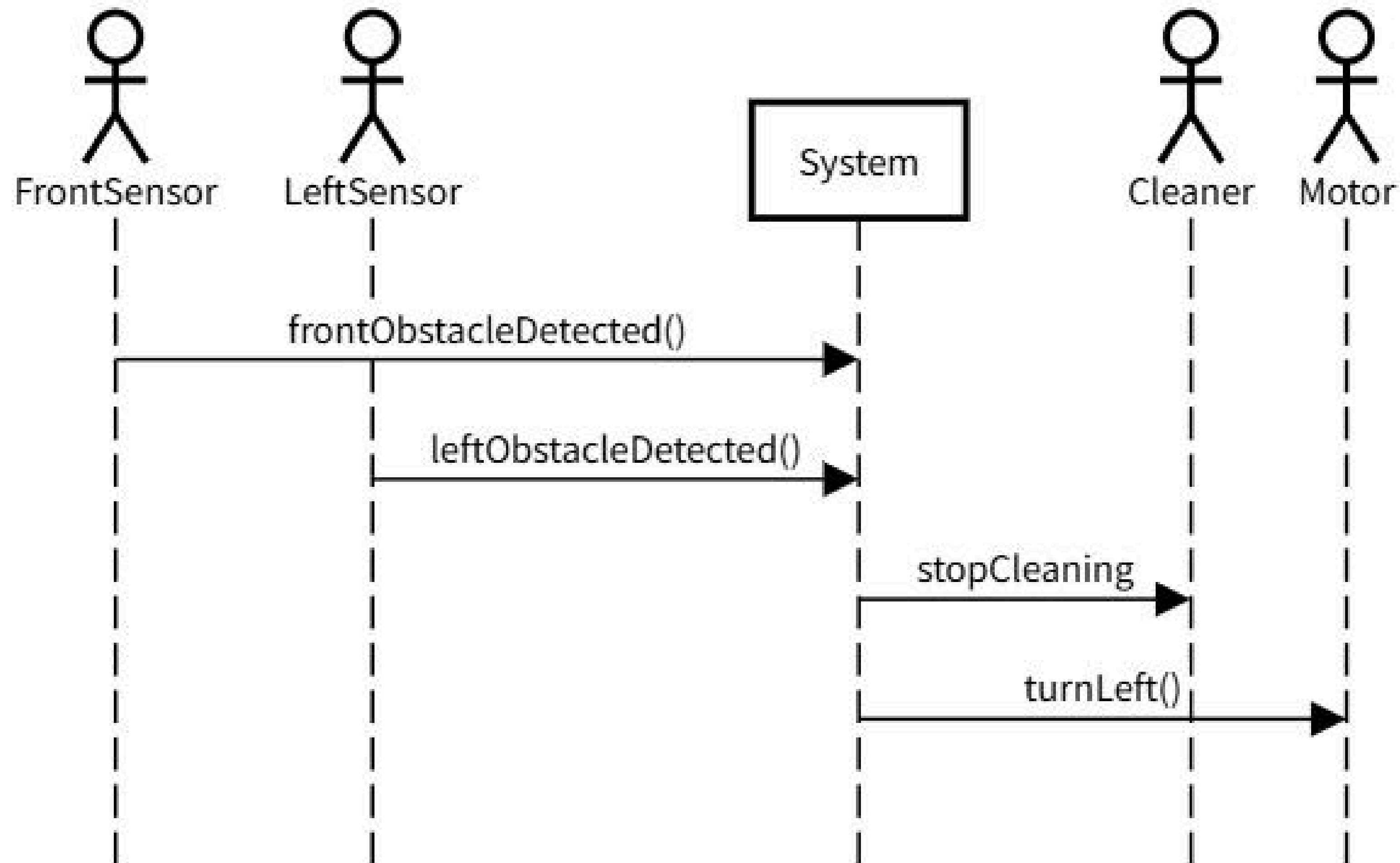


# System Sequence Diagrams: UC-03



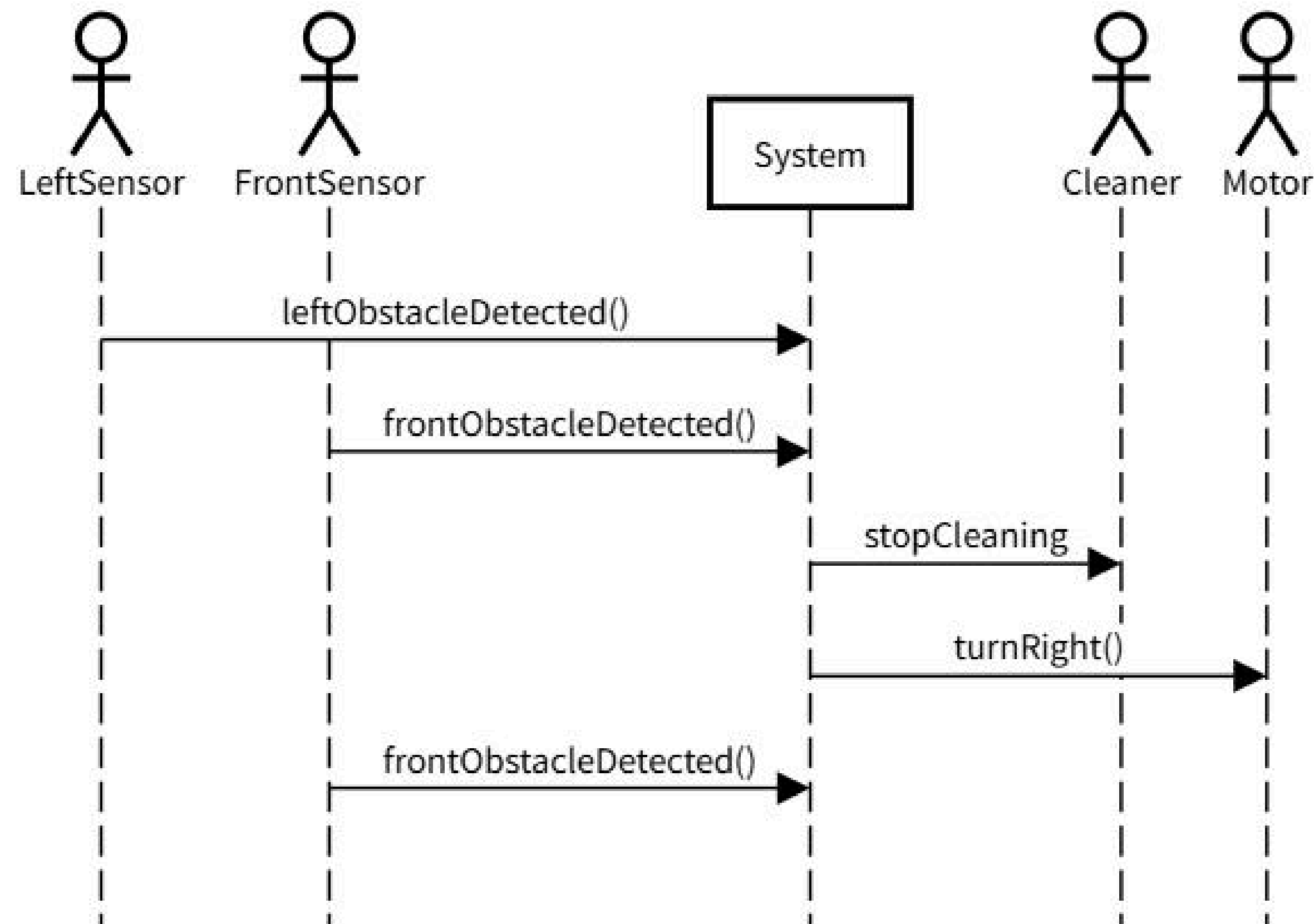
# System Sequence Diagrams: UC-04

장애물 감지 - 좌회전



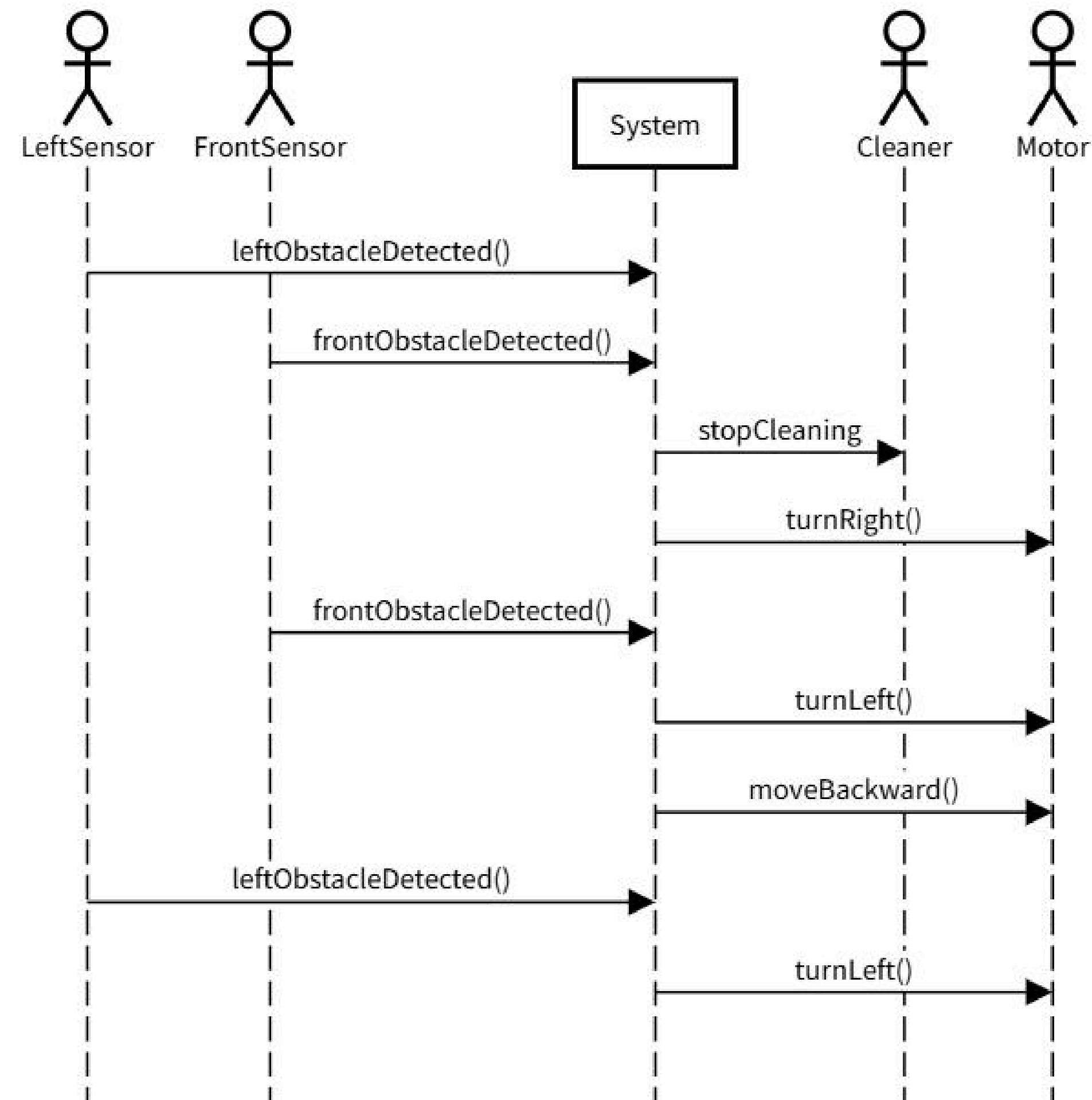
# System Sequence Diagrams: UC-05

장애물 감지 - 우회전

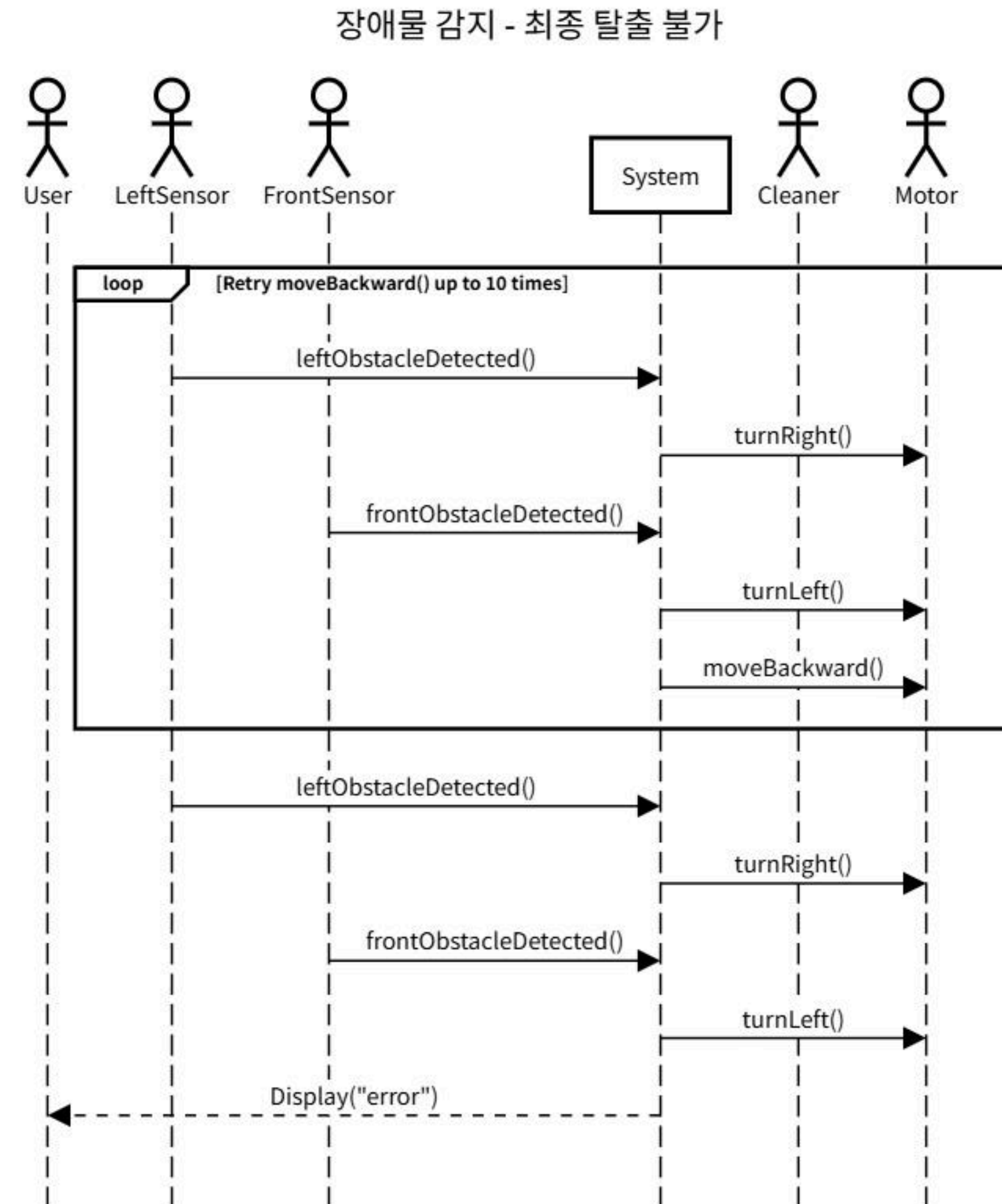


# System Sequence Diagrams: UC-06

장애물 감지 - 후진

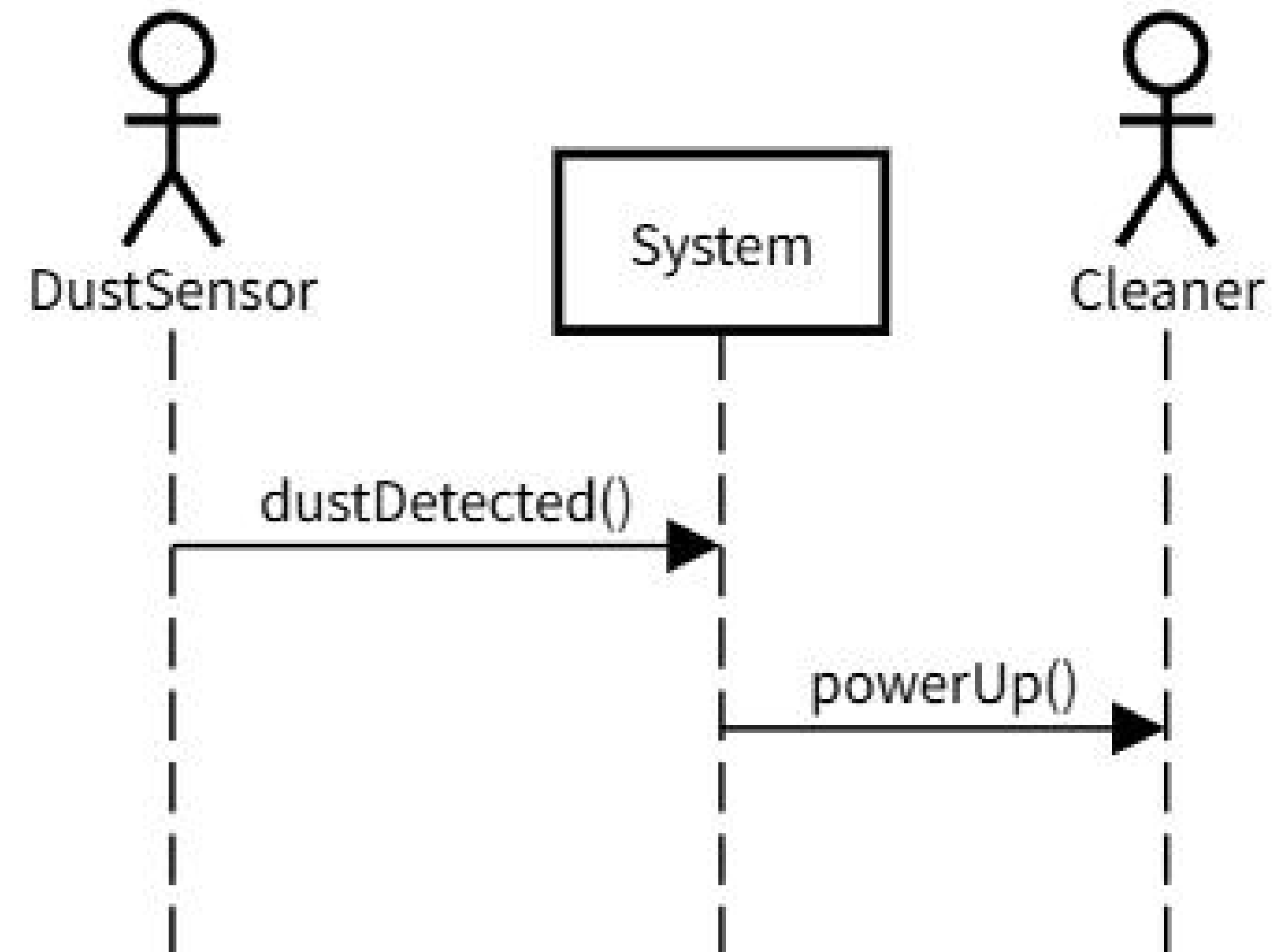


# System Sequence Diagrams: UC-07



# System Sequence Diagrams: UC-08

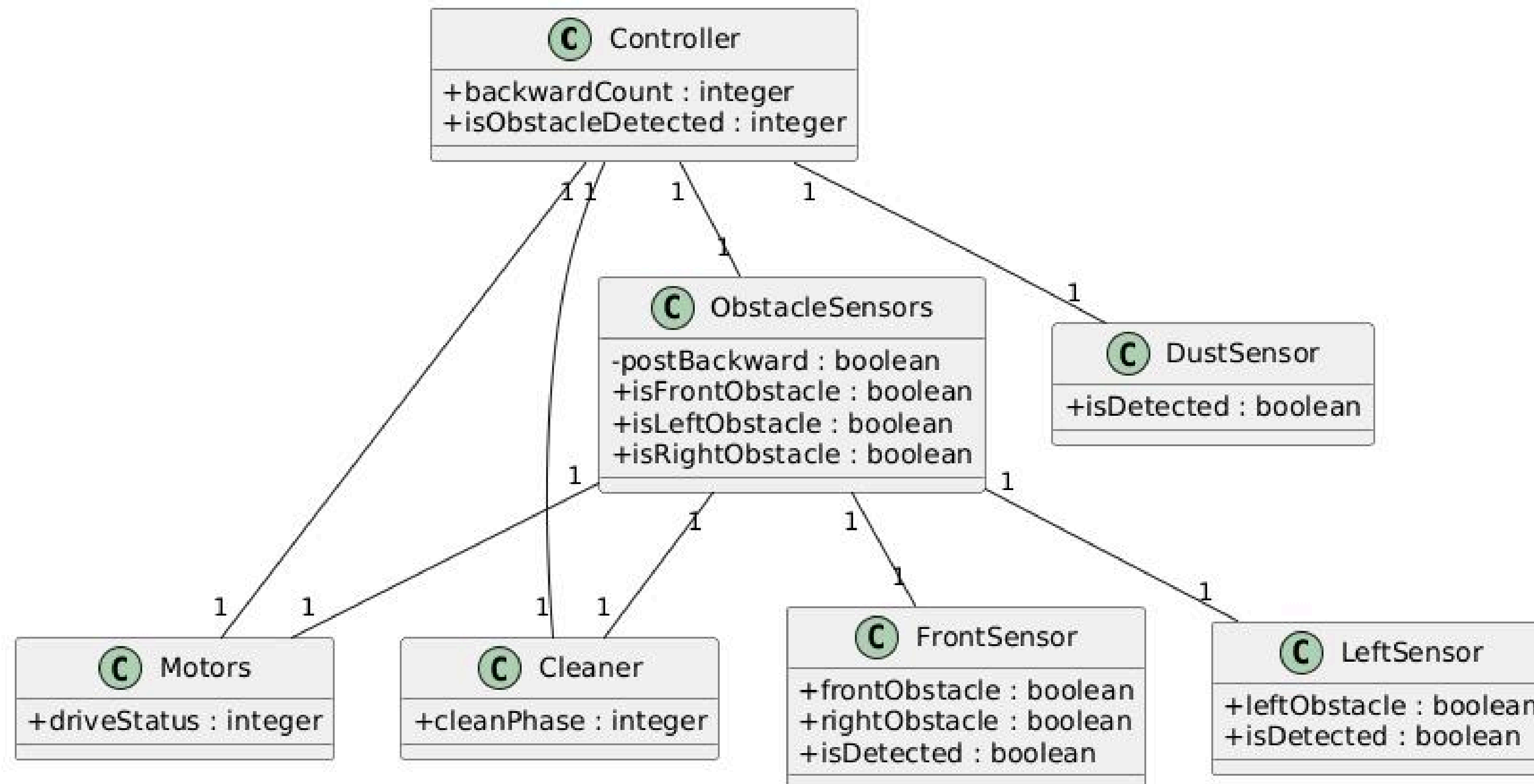
먼지 감지



# Traditional Coding

OOAD: Domain Model

# Domain Model

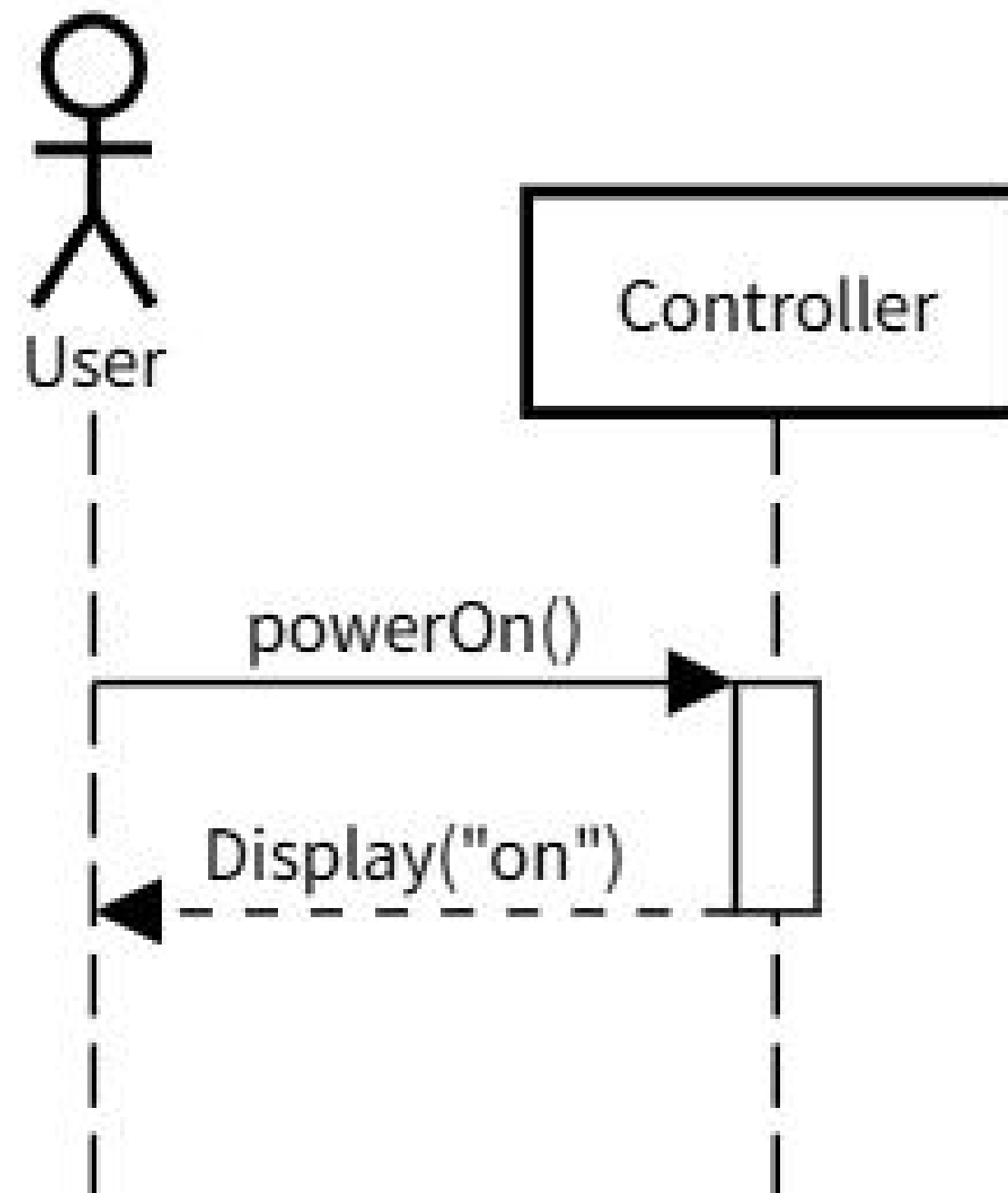


# **Traditional Coding**

OOAD: Sequence Diagram

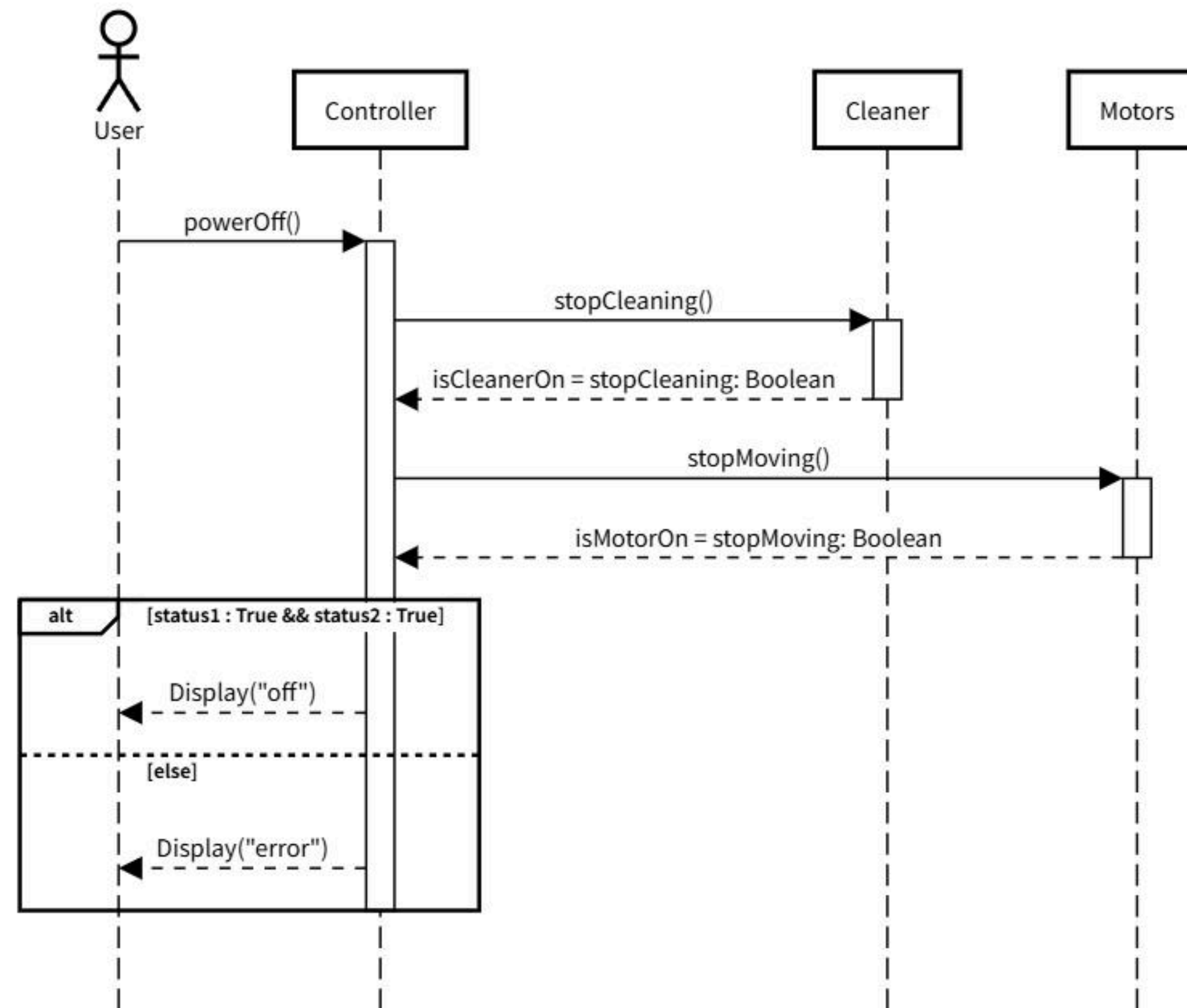
# Sequence Diagrams: UC-01

전원 켜기

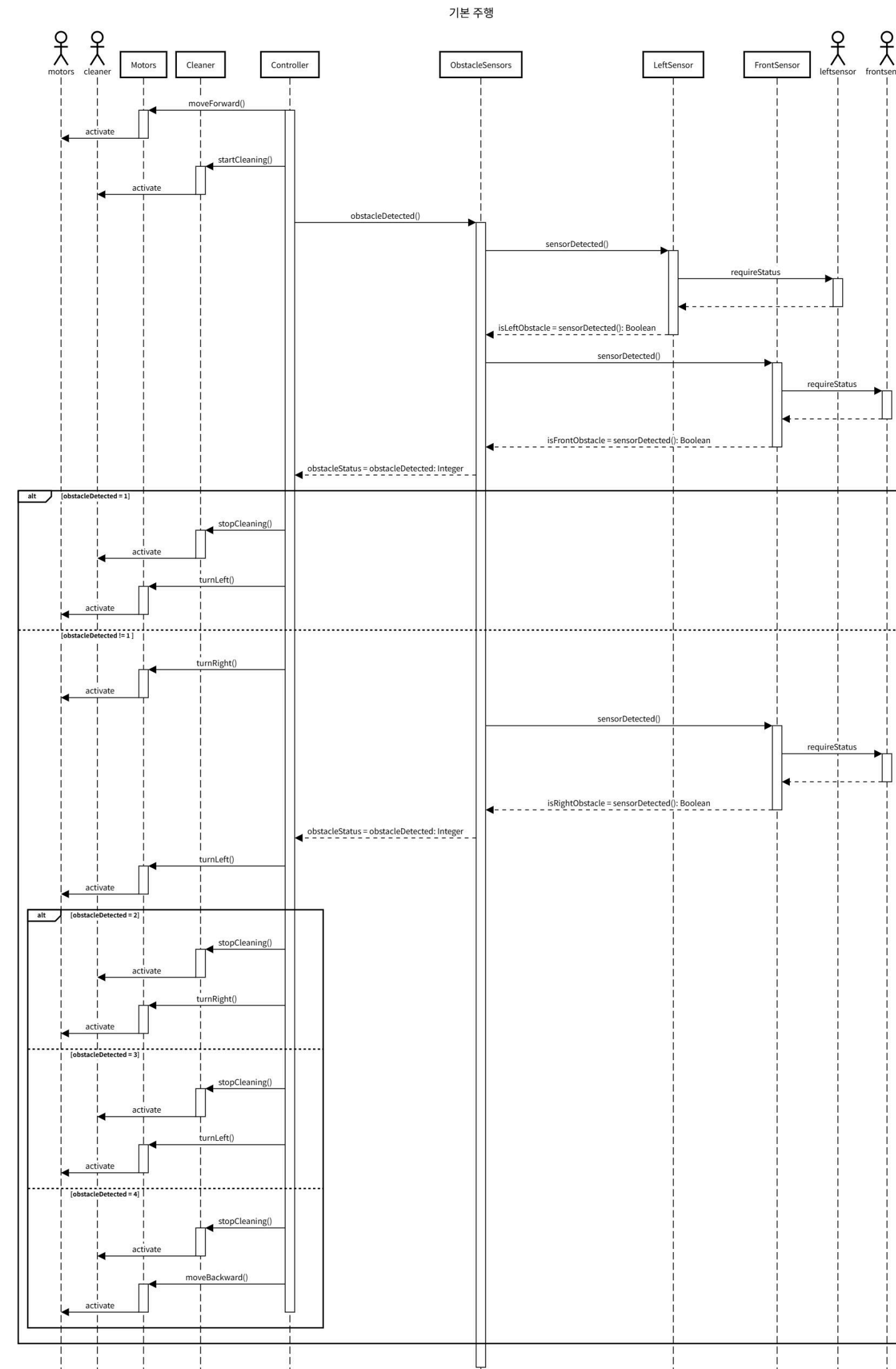


# Sequence Diagrams: UC-02

전원 끄기

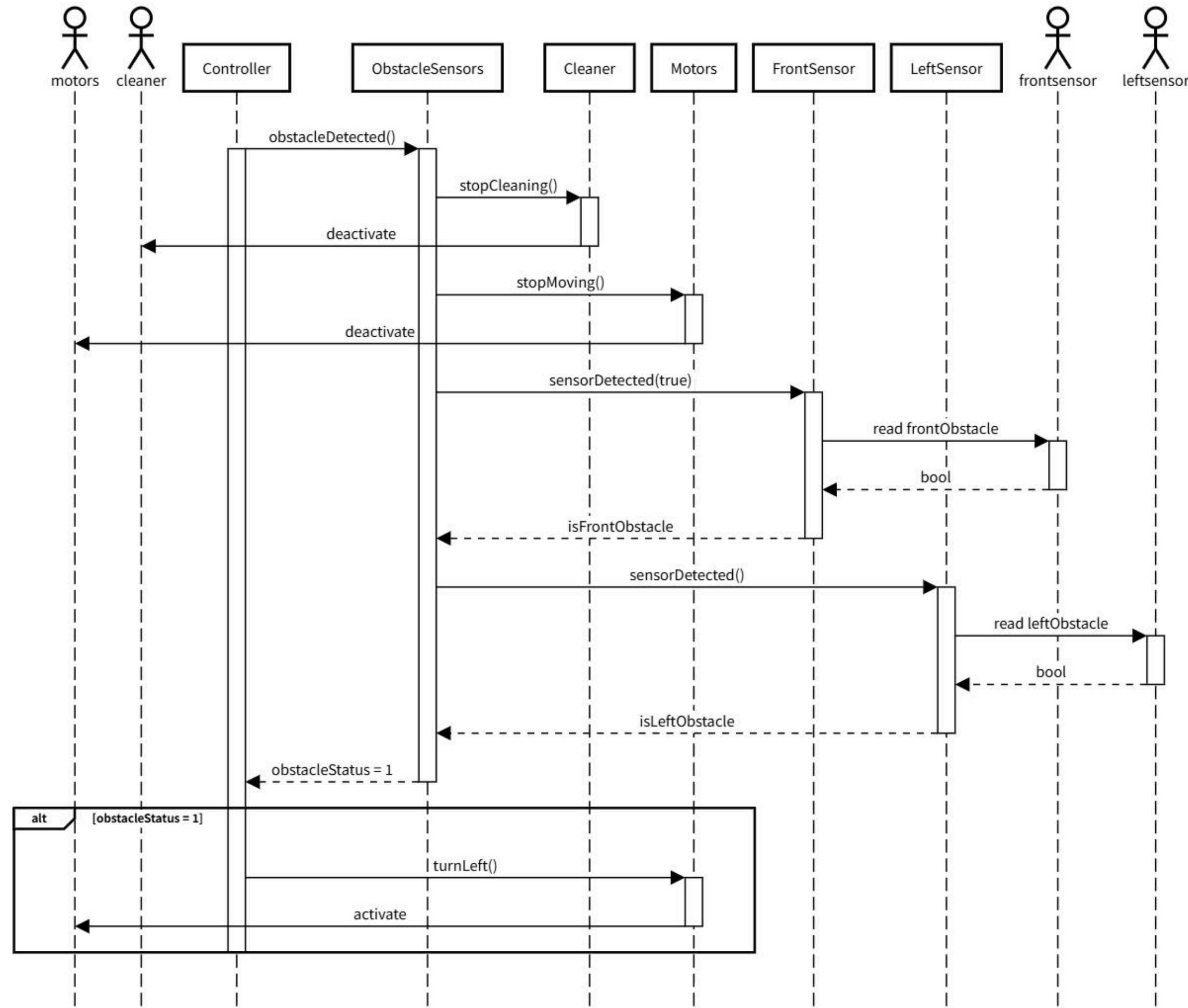


# Sequence Diagrams: UC-03

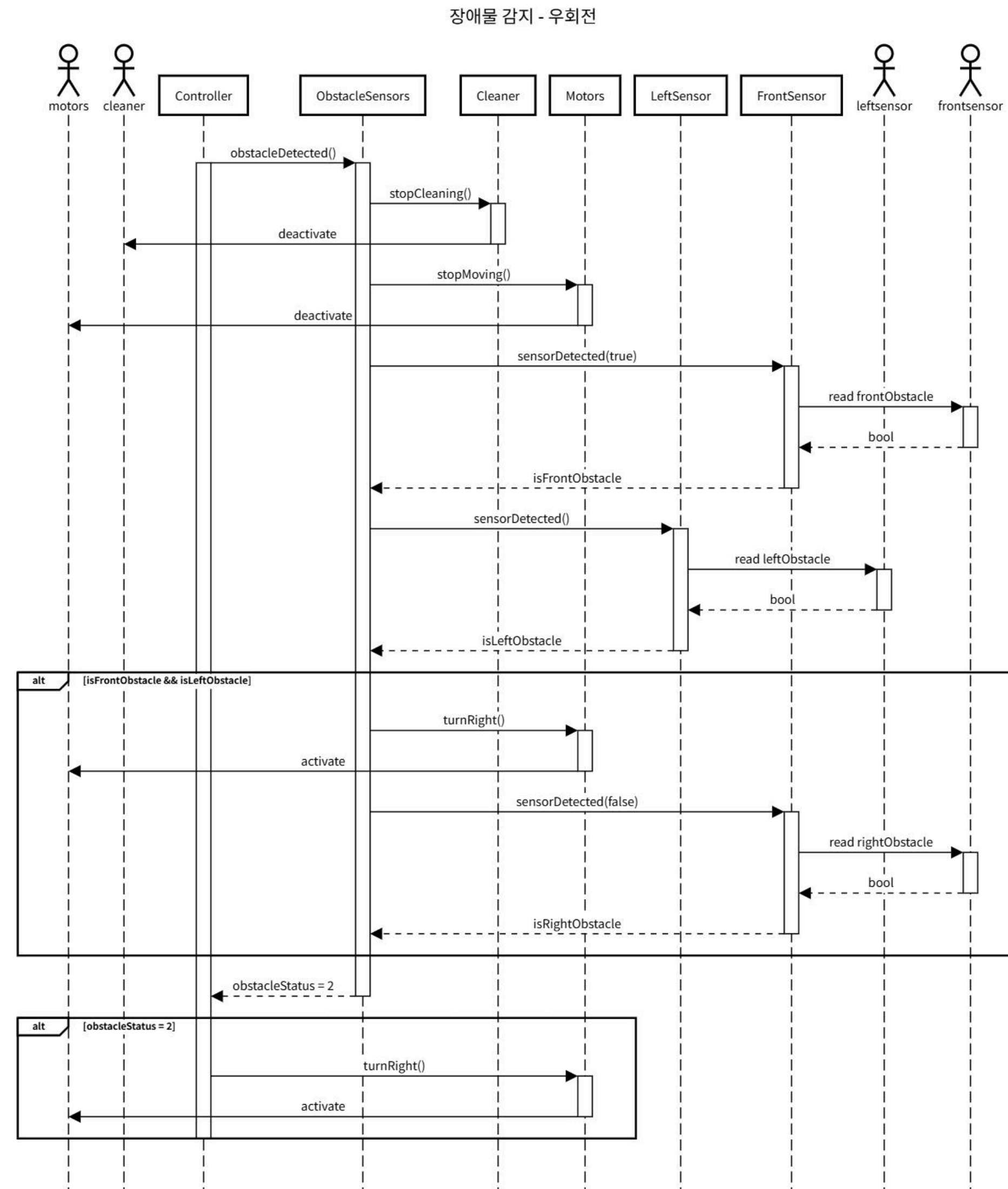


# Sequence Diagrams: UC-04

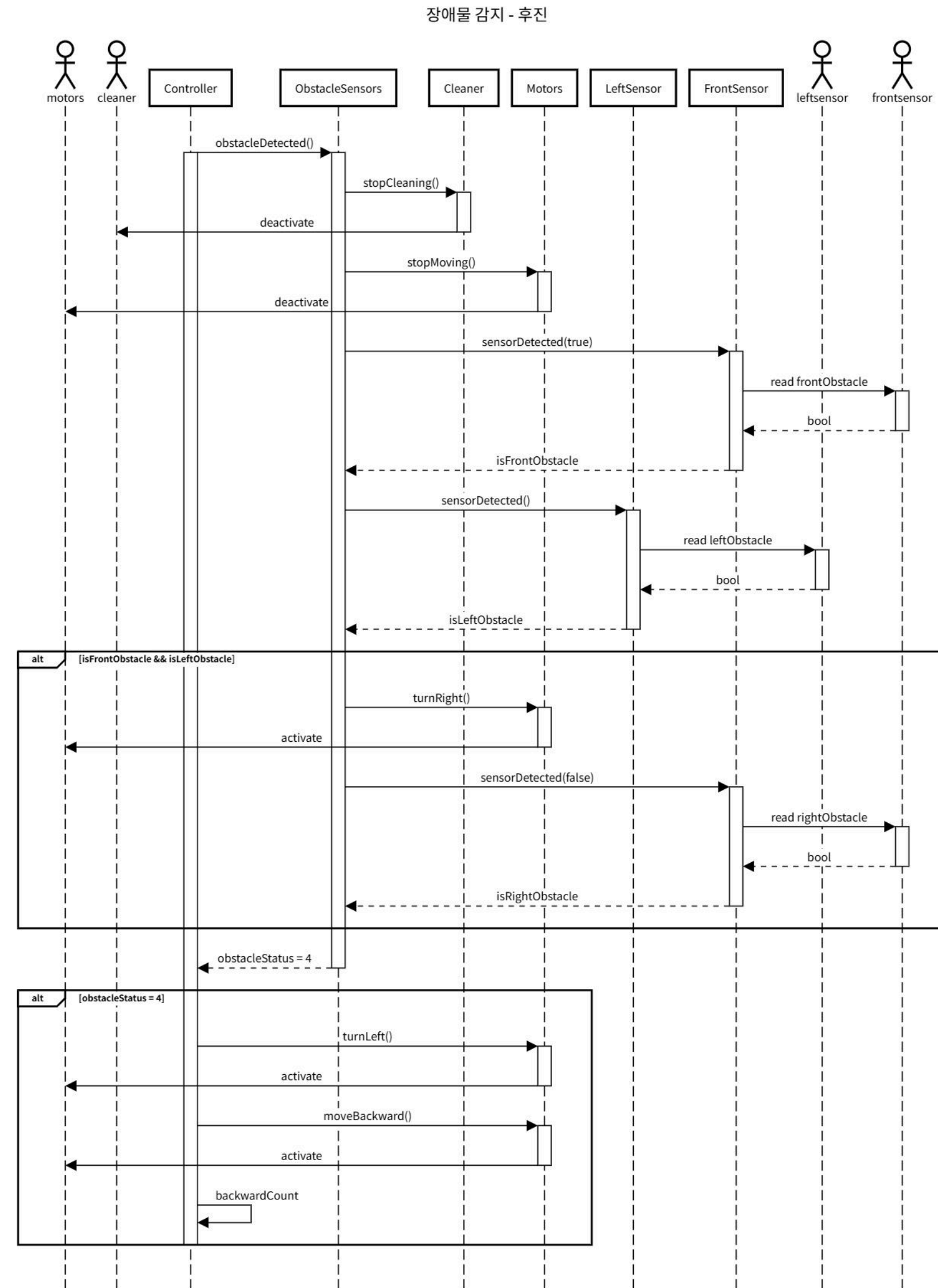
장애물 감지 - 좌회전



# Sequence Diagrams: UC-05

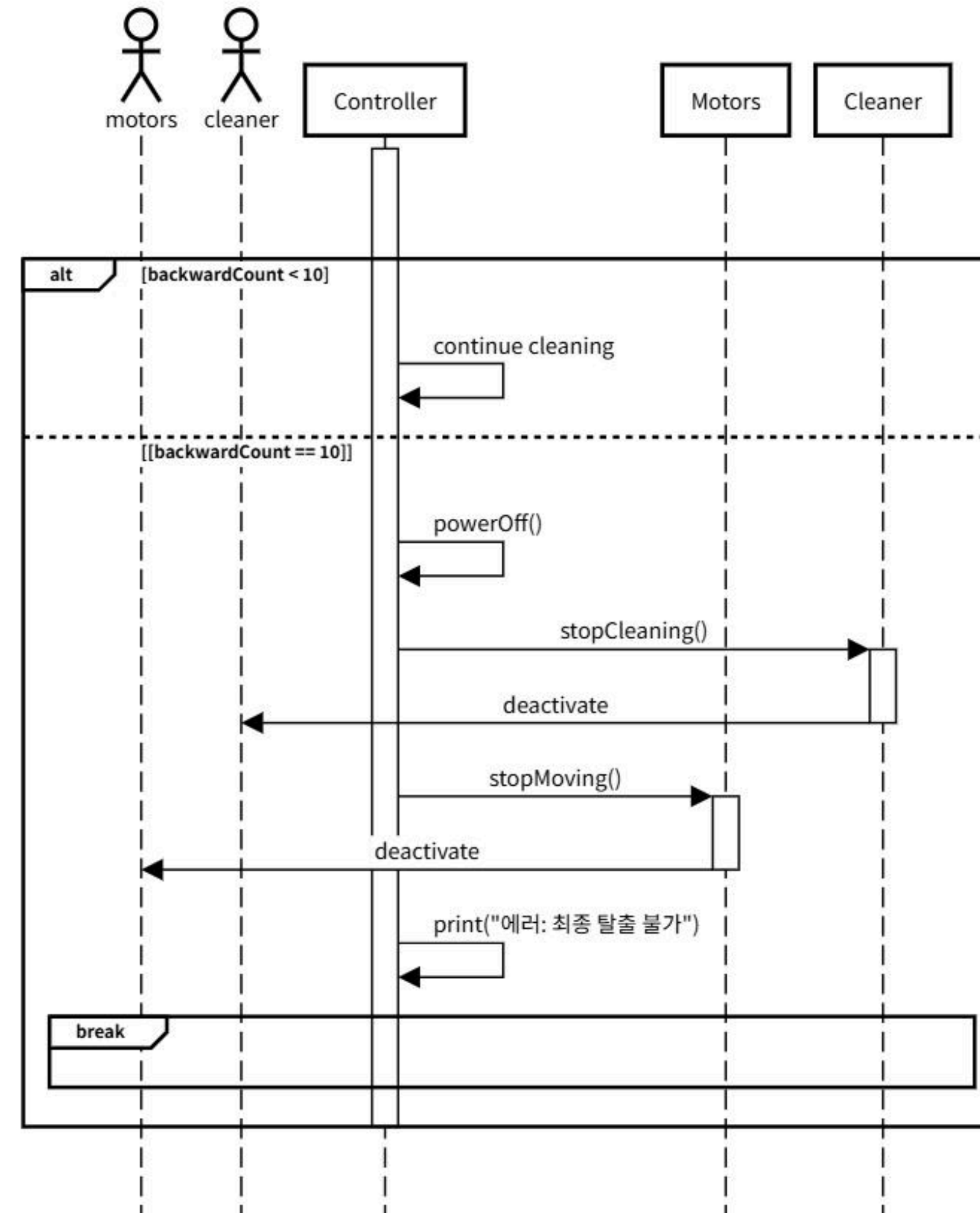


# Sequence Diagrams: UC-06

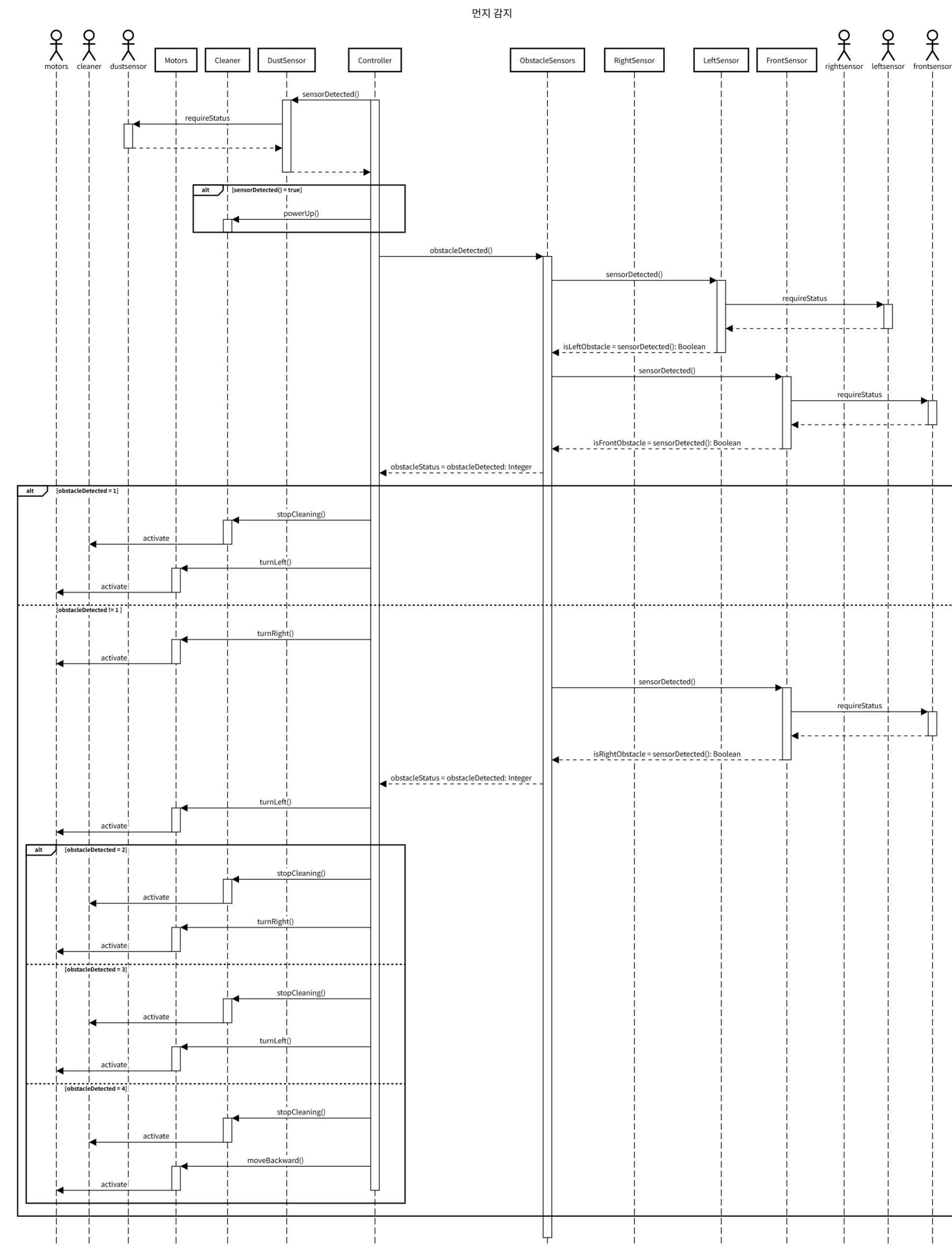


# Sequence Diagrams: UC-07

최종 탈출 불가



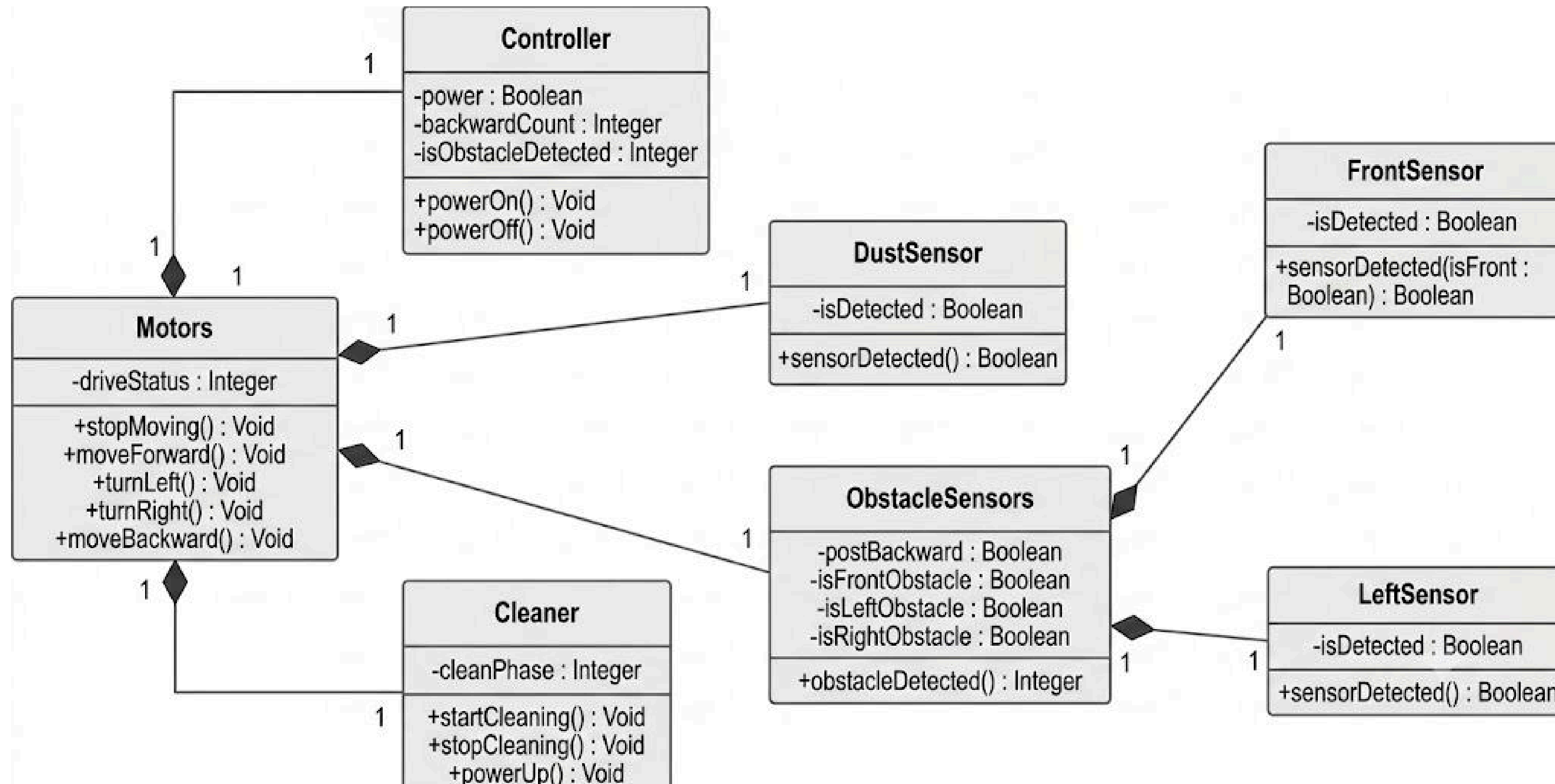
# Sequence Diagrams: UC-08



# Traditional Coding

OOAD: Class Diagram

# Class Diagram



# Traditional Coding

OOI

# Code

The screenshot displays a GitHub repository interface for 'OOAD-Team9 / RVC\_Controller'. The main content area shows a commit merge for commit 916b752, authored by DONGL3XD last month. The commit message is 'Merge pull request #55 from 00AD-Team9/develop' and it indicates that 480 files were changed with a total of +230,280 lines of code. A yellow warning banner states 'Some content is hidden' due to the large commit size. Below this, a search box for 'Search within code' is visible. The code diff for 'SystemTest/Report01/index.html' is shown, with line 15 containing the jQuery ready function: `$(document).ready(function ()`. The left sidebar shows a file tree with folders like 'SystemTest', 'Report01', 'third-party', 'JQuery', 'RGraph', 'libraries', 'css', and 'table-images', along with various files including 'index.html', 'jquery-3.4.1.min.js', and 'style.css'.

OOAD-Team9 / RVC\_Controller

Code Issues Pull requests 1 Agents Actions Projects Security and quality 1 Insights Settings

Commit 916b752 [Browse files](#)

DONGL3XD authored last month · 4/4 · Verified

Merge pull request #55 from 00AD-Team9/develop

Develop

main (#55) 2 parents 53d2242 + 500dc1f commit 916b752

480 files changed +230,280

Filter files...

SystemTest

Report01

- index.html

third-party

JQuery

- jquery-3.4.1.min.js

RGraph

libraries

- RGraph.common.core...
- RGraph.common.dyn...
- RGraph.common.tool...
- RGraph.pie.js
- license.txt

css

- style.css

table-images

- bottleft.png

Some content is hidden

Large Commits have some content hidden by default. Use the searchbox below for content that may be hidden.

Search within code

SystemTest/Report01/index.html +82

```
... @@ -0,0 +1,82 @@
1 + <!DOCTYPE html>
2 + <html xmlns="http://www.w3.org/1999/xhtml">
3 +   <head>
4 +     <meta charset="utf-8"/>
5 +     <title>RVC.exe</title>
6 +     <link rel="stylesheet" type="text/css" href="third-party/css/style.css"/>
7 +     <script src="third-party/RGraph/libraries/RGraph.common.core.js" type="text/javascript"></script>
8 +     <script src="third-party/RGraph/libraries/RGraph.common.dynamic.js" type="text/javascript"></script>
9 +     <script src="third-party/RGraph/libraries/RGraph.common.tooltips.js" type="text/javascript"></script>
10 +     <script src="third-party/RGraph/libraries/RGraph.pie.js" type="text/javascript"></script>
11 +     <script src="third-party/JQuery/jquery-3.4.1.min.js"></script>
12 +   </head>
13 +   <body>
14 +     <script type="text/javascript">
15 +       $(document).ready(function ()
```

# **Traditional Coding**

Test & Static Analysis: Unit Test

# Unit Test

🏠 Summary
☰

All jobs ☰

- ✔ Style Check (cppLint)
- ✔ Static Analysis (cppCheck)
- ✔ Build and Scan (gTest, SonaQubeCloud)

Run details

- 🕒 Usage
- 📄 Workflow file

**Build and Scan (gTest, SonaQubeCloud)**  
succeeded last month in 1m 28s

▼ ✔ Test and Generate Coverage

```

92 32/48 Test #32: SensorTest.LeftSensorTest_Default ..... Passed 0.00 sec
93      Start 32: SensorTest.LeftSensorTest_Detected
94 32/48 Test #32: SensorTest.LeftSensorTest_Detected ..... Passed 0.00 sec
95      Start 33: SensorTest.LeftSensorTest_Sequential
96 33/48 Test #33: SensorTest.LeftSensorTest_Sequential ..... Passed 0.00 sec
97      Start 34: SensorTest.RightSensorTest_Default
98 34/48 Test #34: SensorTest.RightSensorTest_Default ..... Passed 0.00 sec
99      Start 35: SensorTest.RightSensorTest_Detected
100 35/48 Test #35: SensorTest.RightSensorTest_Detected ..... Passed 0.00 sec
101      Start 36: SensorTest.RightSensorTest_Sequential
102 36/48 Test #36: SensorTest.RightSensorTest_Sequential ..... Passed 0.00 sec
103      Start 37: SensorTest.DustSensor_Error
104 37/48 Test #37: SensorTest.DustSensor_Error ..... Passed 0.00 sec
105      Start 38: SensorTest.FrontSensor_Error
106 38/48 Test #38: SensorTest.FrontSensor_Error ..... Passed 0.00 sec
107      Start 39: SensorTest.LeftSensor_Error
108 39/48 Test #39: SensorTest.LeftSensor_Error ..... Passed 0.00 sec
109      Start 40: SensorTest.RightSensor_Error
110 40/48 Test #40: SensorTest.RightSensor_Error ..... Passed 0.00 sec
111      Start 41: ObstacleSensorsTest.Default
112 41/48 Test #41: ObstacleSensorsTest.Default ..... Passed 0.00 sec
113      Start 42: ObstacleSensorsTest.Front_test
114 42/48 Test #42: ObstacleSensorsTest.Front_test ..... Passed 0.00 sec
115      Start 43: ObstacleSensorsTest.Left_test
116 43/48 Test #43: ObstacleSensorsTest.Left_test ..... Passed 0.00 sec
117      Start 44: ObstacleSensorsTest.Right_test
118 44/48 Test #44: ObstacleSensorsTest.Right_test ..... Passed 0.00 sec
119      Start 45: ObstacleSensorsTest.Front_Left_test
120 45/48 Test #45: ObstacleSensorsTest.Front_Left_test ..... Passed 0.00 sec
121      Start 46: ObstacleSensorsTest.Front_Right_test
122 46/48 Test #46: ObstacleSensorsTest.Front_Right_test ..... Passed 0.00 sec
123      Start 47: ObstacleSensorsTest.Front_Left_Right_test
124 47/48 Test #47: ObstacleSensorsTest.Front_Left_Right_test ... Passed 0.00 sec
125      Start 48: ObstacleSensorsTest.Left_Right_test
126 48/48 Test #48: ObstacleSensorsTest.Left_Right_test ..... Passed 0.00 sec
127
128 100% tests passed, 0 tests failed out of 48
129
130 Total Test time (real) = 30.15 sec
131 (INFO) Reading coverage data...
132
133 (INFO) Writing coverage report...
                    
```

# Code Review

Code review interface showing a change in `include/Controller.h`. The change is a new line of code: `motors->turnLeft();` at line 102. A comment from **eomright** (16 minutes ago) asks: "만약에 후진을 하다가 왼쪽에만 장애물이 있을 경우 우회전을 할지 직진을 할 지에 대한 로직이 존재하나요? 직진을 하면 안될것 같습니다." (If you go back while there is an obstacle only on the left, will you do a U-turn or go straight? There is no logic for this, right? It might not be good if you go straight.)

```
include/Controller.h
95 99          break;
96 100
97 101      default:
102 +      motors->turnLeft();
```

**eomright** 16 minutes ago

만약에 후진을 하다가 왼쪽에만 장애물이 있을 경우 우회전을 할지 직진을 할 지에 대한 로직이 존재하나요? 직진을 하면 안될것 같습니다.

**Chaminwoo** 3 minutes ago

수정 완료했습니다.

Code review interface showing a change in `src/main.cpp`. The change is a new line of code: `auto obstacleSensors = std::make_shared<ObstacleSensors>(frontSensor, leftSensor, motors);` at line 17. A comment from **eomright** (15 minutes ago) asks: "sensor에 대한 함수인데 motors가 있는 이유가 있을까요??" (It's a function for sensor, but why is there a motor?)

```
src/main.cpp Outdated
16 15      auto leftSensor = std::make_shared<LeftSensor>(data);
17 16      auto dustSensor = std::make_shared<DustSensor>(data);
18 -      auto obstacleSensors = std::make_shared<ObstacleSensors>(frontSensor, leftSensor, rightSensor);
17 +      auto obstacleSensors = std::make_shared<ObstacleSensors>(frontSensor, leftSensor, motors);
```

**eomright** 15 minutes ago

sensor에 대한 함수인데 motors가 있는 이유가 있을까요??

**Chaminwoo** 1 minute ago

우측 센서를 제거하기 위해 회전 후 탐지를 컨트롤러에서 모두 담당할 수 없었기에 ObstacleSensors에서 관리하도록 변경하는 과정에서 모터가 들어갔습니다.

# **Traditional Coding**

Test & Static Analysis: System Test

# Simulator

```
----- [Simulator] RVC Status -----  
  
          OBSTL  
          |RVC|  
  
----- [Simulator] 장애물/면지 상태 입력 -----  
- 0: 아무것도 감지되지 않음  
- 1: 정면만 감지  
- 2: 정면, 좌측 감지  
- 3: 정면, 우측 감지  
- 4: 정면, 좌측, 우측 감지  
- 5: 좌측, 우측 감지 (후진용)  
- 6: 면지 감지되지 않음  
- 7: 면지 감지  
- 8: 전원 켜기/끄기  
- 9: 시뮬레이터 종료  
  
- 11: 정면 장애물 센서 에러  
- 12: 좌측 장애물 센서 에러  
- 13: 우측 장애물 센서 에러  
- 14: 면지 센서 에러  
- 15: 클리너 에러  
- 16: 모터 에러  
|
```

# System Test Cases

positive	negative
전원 켜기	전원 켜는 중 클리너 오작동
전원 끄기	전원 켜는 중 전체 장애물 센서 고장
기본 주행	먼지 흡입 후 power up 유지
전방 장애물 감지	최종 탈출 불가
전방, 좌측 장애물 감지	전방 장애물 감지 실패
전방, 우측 장애물 감지	좌측 장애물 감지 실패 (논리상 우회전 불가로 좌회전)
전방, 좌우측 장애물 감지	모터 에러 (장애물 감지에도 이동 불능)

# System Test Cases

positive	negative
먼지 감지	우회전 시작 후 cleaner 고장으로 청소 재개 불가
후진 중 탈출	주행 중 청소 불가
전방 장애물 감지 및 먼지 감지	회전 후 청소 불가
전방, 좌측 장애물 감지 및 먼지 감지	
전방, 우측 장애물 감지 및 먼지 감지	
전방, 좌우측 장애물 감지 및 먼지 감지	
후진 후 우측 장애물 감지	

# System Test Cases

Positive	Negative
후진 후 좌측 장애물 감지	
후진 후 장애물 감지하지 않을 경우	
좌회전 시작 후 센서 값이 “전방+좌측”으로 바뀌어도 중단 없이 회전	
우회전 시작 후 센서 값이 “전방+우측”으로 바뀌어도 중단 없이 회전	
파워업 중 클리너 에러	
후진 이후 전면 막힘	













# Test 수행 과정 및 결과

## 13. 전방, 좌우측 장애물 감지 및 먼지 감지

```

C:\Users\User\WRVC_Conf x + v
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Cleaner]청소 종료
[Motors]정지
[Motors]우회전
[Motors]좌회전
[Motors]3초간 후진
[Cleaner]청소 LV-2
[Motors]우회전
[Motors]전진
[Cleaner]청소 LV-2
[Motors]전진
[Cleaner]청소 LV-2
[Motors]전진
[Cleaner]청소 LV-2
[Motors]전진

C:\Users\User\Download x + v
0
----- [Simulator] RVC Status -----
|RVC|
----- [Simulator] 장애물/먼지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 먼지 감지되지 않음
- 7: 먼지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료
- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
    
```

## 14. 후진 후 우측 장애물 감지

```

C:\Users\User\WRVC_Conf x + v
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Cleaner]청소 종료
[Motors]정지
[Motors]우회전
[Motors]좌회전
[Motors]3초간 후진
[Cleaner]청소 LV-1
[Cleaner]청소 종료
[Motors]정지
[Motors]우회전
[Motors]좌회전
[Cleaner]청소 LV-1
[Cleaner]청소 종료
[Motors]정지
[Motors]좌회전
[Cleaner]청소 LV-1

C:\Users\User\Download x + v
- 16: 모터 에러
0
----- [Simulator] RVC Status -----
|RVC|
----- [Simulator] 장애물/먼지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 먼지 감지되지 않음
- 7: 먼지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료
- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
    
```

# Test 수행 과정 및 결과

## 15. 후진 후 좌측 장애물 감지

```

C:\Users\User\RV_Comb x + v
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 종료
[Motors]정지
[Motors]우회전
[Motors]좌회전
[Motors]3초간 후진
[Cleaner]청소 LV-1
[Motors]우회전
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1

C:\Users\User\Download x + v
0
----- [Simulator] RVC Status -----

|RVC|

----- [Simulator] 장애물/먼지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 먼지 감지되지 않음
- 7: 먼지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료

- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
    
```

## 16. 후진 후 장애물 감지하지 않을 경우

```

C:\Users\User\RV_Comb x + v
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Cleaner]청소 종료
[Motors]정지
[Motors]우회전
[Motors]좌회전
[Motors]3초간 후진
[Cleaner]청소 LV-1
[Cleaner]청소 종료
[Motors]정지
[Motors]우회전
[Motors]좌회전
[Cleaner]청소 LV-1
[Cleaner]청소 종료
[Motors]정지
[Motors]좌회전
[Cleaner]청소 LV-1

C:\Users\User\Download x + v
- 16: 모터 에러
0
----- [Simulator] RVC Status -----

|RVC|

----- [Simulator] 장애물/먼지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 먼지 감지되지 않음
- 7: 먼지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료

- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
    
```



# Test 수행 과정 및 결과

## 19. 전원 켜는 중 클리너 오작동으로 전진만 수행

```

C:\Users\User\WRVC_Conb x +
[Cleaner]FAIL: 에러 발생
[Motors]전진
[Cleaner]FAIL: 에러로 인한 작동 불가
[Cleaner]FAIL: 에러 발생
[Motors]전진
[Cleaner]FAIL: 에러로 인한 작동 불가
[Cleaner]FAIL: 에러 발생
[Motors]전진
[Cleaner]FAIL: 에러로 인한 작동 불가
[Cleaner]FAIL: 에러 발생
[Motors]전진
[Cleaner]FAIL: 에러로 인한 작동 불가

C:\Users\User\Download x +
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
15
[Simulator] Cleaner FAIL!
----- [Simulator] 장애물/먼지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 먼지 감지되지 않음
- 7: 먼지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료

- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
  
```

## 20. 전원 켜는 중 전체 장애물 센서 고장으로 전진만 수행

```

C:\Users\User\WRVC_Conb x +
[FrontSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1
[FrontSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1
[FrontSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1
[FrontSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1

C:\Users\User\Download x +
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
11
[Simulator] FrontSensor FAIL!
----- [Simulator] 장애물/먼지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 먼지 감지되지 않음
- 7: 먼지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료

- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
  
```



# Test 수행 과정 및 결과

## 23. 전방 장애물 감지 실패

```

C:\Users\User\WRVC_Conb x + v
[FrontSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1
[FrontSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1
[FrontSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1
[FrontSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1
[FrontSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1
[FrontSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1

C:\Users\User\Download x + v
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
11
[Simulator] FrontSensor FAIL!

----- [Simulator] 장애물/먼지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 먼지 감지되지 않음
- 7: 먼지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료

- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
  
```

## 24. 좌측 장애물 감지 실패 (논리 상 우회전 불가로 좌회전)

```

C:\Users\User\WRVC_Conb x + v
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[LeftSensor]FAIL: 에러 발생
[LeftSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1
[LeftSensor]FAIL: 에러 발생
[LeftSensor]FAIL: 에러로 인한 작동 불가

C:\Users\User\Download x + v
- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
12
[Simulator] LeftSensor FAIL!

----- [Simulator] 장애물/먼지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 먼지 감지되지 않음
- 7: 먼지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료

- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
  
```

# Test 수행 과정 및 결과

## 25. 모터 에러 (장애물 감지에도 이동 불능)

```

C:\Users\User\WRVC_Comb x + v
C:\Users\User\Download x + v

[Motors]FAIL: 에러로 인한 작동 불가
[Cleaner]청소 LV-1
[Motors]FAIL: 에러 발생
[Cleaner]청소 종료
[Motors]FAIL: 에러로 인한 작동 불가
[Motors]FAIL: 에러로 인한 작동 불가
[Cleaner]청소 LV-1
[Motors]FAIL: 에러 발생
[Cleaner]청소 종료
[Motors]FAIL: 에러로 인한 작동 불가
[Motors]FAIL: 에러로 인한 작동 불가
[Cleaner]청소 LV-1
[Motors]FAIL: 에러 발생
[Cleaner]청소 종료
[Motors]FAIL: 에러로 인한 작동 불가
[Motors]FAIL: 에러로 인한 작동 불가
[Cleaner]청소 LV-1
[Motors]FAIL: 에러 발생
[Cleaner]청소 종료
[Motors]FAIL: 에러로 인한 작동 불가

3
----- [Simulator] RVC Status -----
                                OBSTL
                                |RVC|OBSTL

----- [Simulator] 장애물/먼지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 먼지 감지되지 않음
- 7: 먼지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료

- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
  
```

## 26. 우회전 시작 후 cleaner 고장으로 청소 재개 불가

```

C:\Users\User\WRVC_Comb x + v
C:\Users\User\Download x + v

[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]전진
[Cleaner]청소 LV-1
[Motors]정지
[Motors]우회전
[Cleaner]청소 LV-1
[Cleaner]FAIL: 에러 발생
[Cleaner]FAIL: 에러로 인한 작동 불가

- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러

[Simulator] Cleaner FAIL!

----- [Simulator] 장애물/먼지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 먼지 감지되지 않음
- 7: 먼지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료

- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 먼지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러
  
```



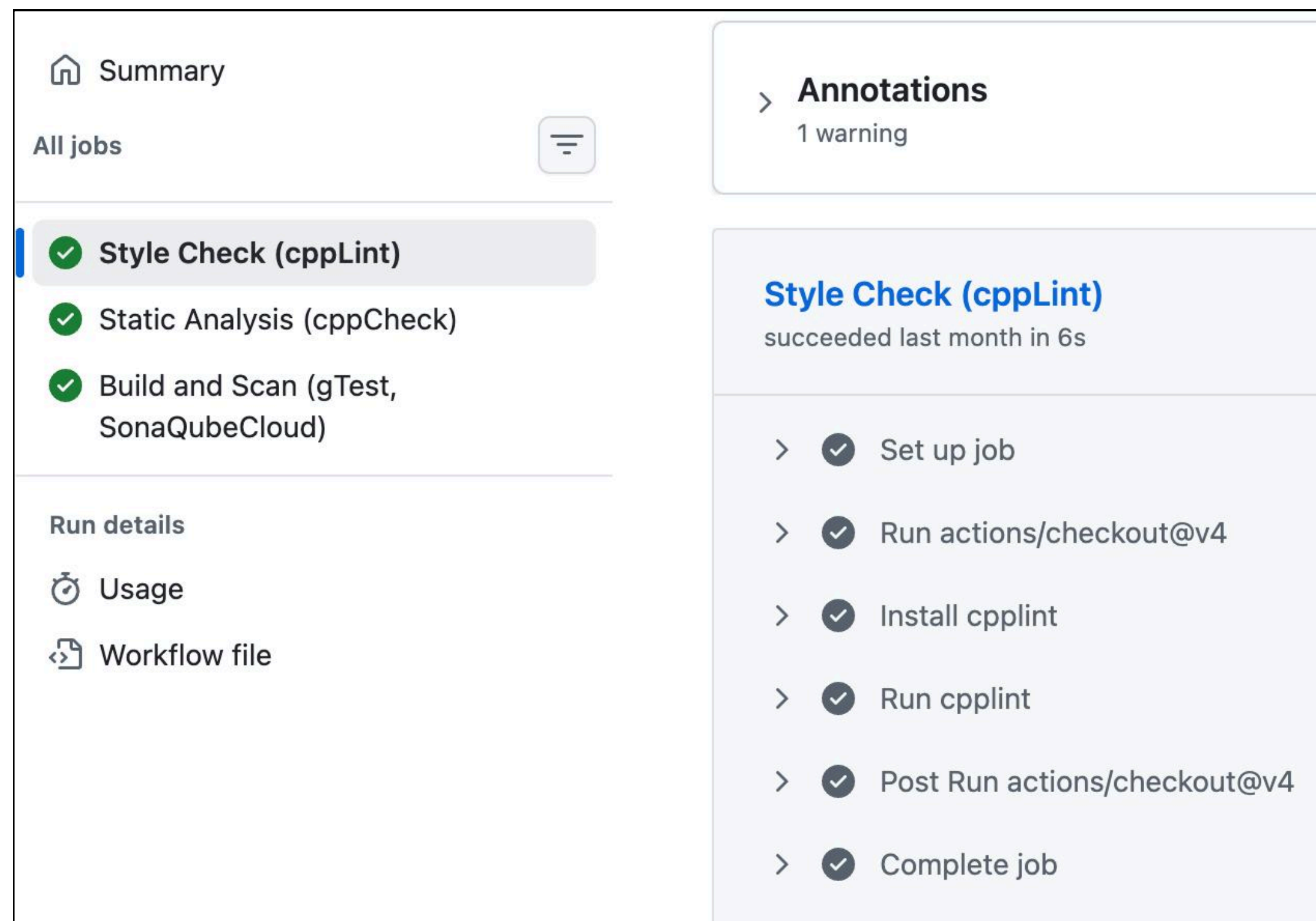


# **Traditional Coding**

Test & Static Analysis: Static Analysis

# Static Analysis

## 1. cppLint

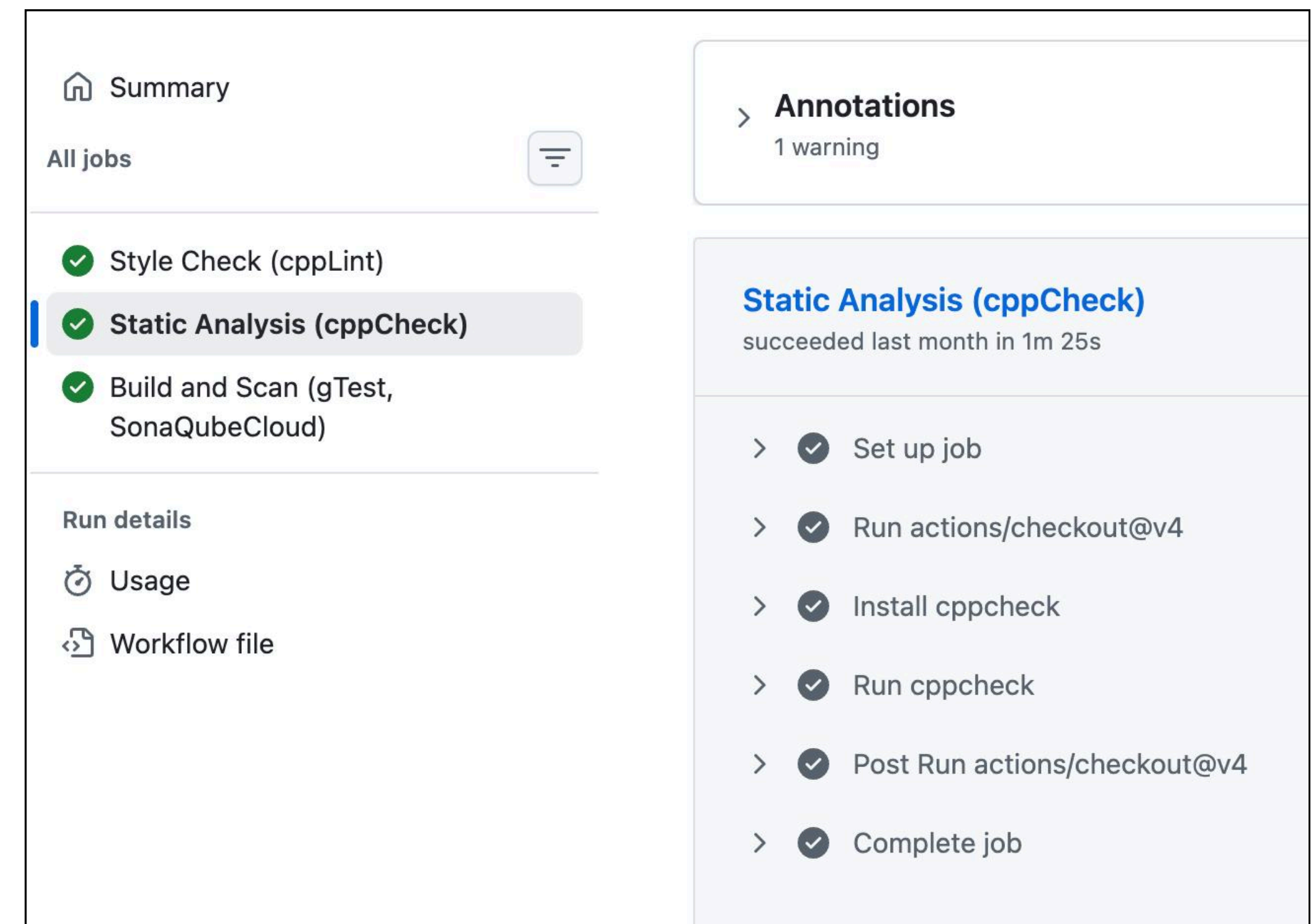


The screenshot shows a GitHub Actions workflow summary for a job named 'Style Check (cppLint)'. The job is marked as successful with a green checkmark. The summary indicates that the job succeeded last month in 6 seconds. The workflow steps are listed as follows:

- Set up job
- Run actions/checkout@v4
- Install cpplint
- Run cpplint
- Post Run actions/checkout@v4
- Complete job

The left sidebar shows the job's status as 'Style Check (cppLint)' and lists other jobs: 'Static Analysis (cppCheck)' and 'Build and Scan (gTest, SonaQubeCloud)'. The top right corner shows 'Annotations' with 1 warning.

## 2. cppCheck



The screenshot shows a GitHub Actions workflow summary for a job named 'Static Analysis (cppCheck)'. The job is marked as successful with a green checkmark. The summary indicates that the job succeeded last month in 1m 25s. The workflow steps are listed as follows:

- Set up job
- Run actions/checkout@v4
- Install cppcheck
- Run cppcheck
- Post Run actions/checkout@v4
- Complete job

The left sidebar shows the job's status as 'Static Analysis (cppCheck)' and lists other jobs: 'Style Check (cppLint)' and 'Build and Scan (gTest, SonaQubeCloud)'. The top right corner shows 'Annotations' with 1 warning.

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

# Vibe Coding

초기 설정

# 프롬프트 설정

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

지금까지 진행한 V2 OOA/OOD를 바탕으로 기존의 RVC SW와 Simulator를 수정. CI/CD 파이프라인, 테스트 등은 이전에 사용한 것과 동일한 기술을 사용. 기존의 Unit Test, System Test 또한 오른쪽 센서가 삭제됨에 따라 수정이 필요함. 이후 전체 시스템을 설명하는 문서를 만들고 V1(Right sensor 삭제 이전)에서 추가/변경/삭제 된 부분을 명시할 것.

Show less

## 전체 변경 요약

### SW 구현 변경

구분	파일	내용
[삭제]	Types.hp p	SensorPosition::RIGHT 제거
[변경]	Controlle r.hpp/cpp	생성자 파라미터 6→5개, checkSideObstacles() 좌측만, avoidFrontObstacle() turn(RIGHT)+front재조회, avoidAllDirectionObstacle() turn(BACK) 제거
[추가]	SimFrontS ensor.hpp	방향 인식 전방 센서 — isFacingRight() 기반 이중 상태 반환
[변경]	SimMoveme ntUnit.hp p	facingRight_ 플래그 + isFacingRight() 추가

[변경]	RVCSimula tor.hpp/cp p	rightSensor_ 삭제, SimFrontSensor 채택, 멤버 순서 변경, setRightObstacle() → setRightSideObstacle()
[변경]	main.cpp	right 명령 → rightside 명령

## 테스트 변경

구분	파일	내용
[삭제]	ControllerTest.cp p	TC-UC04-002(RIGHT) 삭제, rightSensor mock 제거
[변경]	ControllerTest.cp p	TC-UC03-002/003/004, TC-UC05-001/002 v2 로직 반영
[변경]	RVCSystemTest.cp p	rightSensor mock 제거, Scenario2/3 업데이트
[변경]	RVCSimulatorTest. cpp	setRightObstacle → setRightSideObstacle, Scenario3/7 BACK → RIGHT

## 문서

파일	내용
docs/v2/SDD.md	v1→v2 전체 변경 명시, 흐름도, 테스트 변경표

# Vibe Coding

OOAD: FR/NFR

# FR/NFR

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

RVC의 하드웨어 조건이 변경되어 오른쪽 센서 없이 전방/좌측 센서로만 장애물을 탐지해야 하며 이외 기능들은 동일하게 구현되어야 함. OOAD 단계들을 처음부터 다시 진행해 기존의 개발을 수정하며 docs/v2 디렉토리 내에 v1과 동일한 문서들을 제작. 이때 우측 센서가 사라진 수정사항에 대해 [추가/변경/삭제] 를 문서 내에 명시할 것. 우선 FR/NFR 문서부터 작성. 우측 센서가 삭제된 것 외의 변경사항은 없음.

Show less

● Thought for 4s >

● 기존 docs 구조와 요구사항 파일을 확인합니다.



우측 센서가 삭제되어  
우회전이라는 기동 자체를 삭제함

# FR/NFR

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

우측 센서가 사라졌다고 해서 기존에 존재하던 기능을 삭제할 수 없고 유지해야 함. 로직을 추가/변경함으로써 구현해야 함.

● Thought for 44s >

● 이해했습니다. 우측 회전, 전방향 역전 등 기능 자체는 그대로 유지하되, 우측 센서 없이 어떻게 해당 기능을 구현하는지 로직을 변경/추가하는 방향으로 작성합니다.

재분석:

- 우측 회전 기능 → 유지. 단 트리거 조건 변경: 좌측이 막히면 우측 여부 확인 없이 우측 회전 시도
- 전방향 역전 기동 → 유지. 단 발동 조건 변경: 전방+좌 막힘 시 (우측 확인 불가 → 조건 완화)
- FR-SEN-03 → [삭제] (HW 자체가 없음)
- 우측 장애물 감지로 인한 방향 판단 로직 → [변경] (좌측 센서만으로 판단, 좌측 막힘 시 우측 방향 시도)



우측 센서가 없으니  
우측 장애물에 대한 감지 자체를 삭제하고  
**역전 기동**(전방/좌측에 장애물 감지 시  
후진 후 좌측으로 탈출)이라는 새로운 기동을 추가함

# FR/NFR

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

우회전을 삭제하거나 새로운 기동을 만들어서는 안됨. 기존 존재하는 기능들은 유지한 채로 우측 센서가 사라졌을 때 로직만 변경.

- Thought for 35s >
- 이해했습니다. 기능/기동은 전부 그대로이고, 우측 센서가 없어서 조건 판단 로직만 달라지는 것입니다. 최소한의 변경만 반영합니다:
  - FR-SEN-03 → [삭제] (HW 없음)
  - FR-MOV-05 → [변경] 발동 조건만: "전방+좌+우 모두 막힘" → "전방+좌 막힘"
  - NFR-SAF-01 → [변경] 조건만: "모든 방향" → "전방+좌측"
  - 나머지는 전부 동일



우측 센서가 없기에  
우측 장애물 감지를 아예 고려하지 않고 계속해서 진행함

# FR/NFR

역전기동의 내용은 V1에 존재하지 않았음. 우측 센서가 없으므로 우측 장애물의 데이터를 활용하지 못하는 것이 아닌 RVC를 우측으로 90도 회전 후 전방 센서를 이용해 우측 장애물을 감지하는 식으로 우측 장애물을 감지할 수 있음. 이를 이용해 역전기동을 삭제하고 전진/좌회전/우회전/후진 등의 기동을 가능하도록 하는 FR/NFR문서를 업데이트해 작성할 것.

Show less

Thought for 115s >

이제 의도를 정확히 이해했습니다.

### 핵심 원리:

- 우측 센서 제거 → 우측을 우회전 90° + 전방 센서로 감지
- 역전기동( `turn(BACK)` 180° 회전 패턴)은 삭제
- 사용 가능한 기동: 전진/좌회전/우회전/후진 (기본 4가지만)



결국 직접 우측 장애물 감지 로직을 프롬프트로 입력해 올바른 로직을 구현하기 시작함.

# Functional Requirements

**FR-MOV : 이동(Movement) 관련**

**FR-SEN : 센서(Sensor) 관련**

**FR-CLN : 청소(Cleaning) 관련**

FR-MOV

# Functional Requirements

## FR-MOV-01

RVC는 기본 상태에서 직진(전방)하며 청소한다.

## FR-MOV-02

전방에 장애물이 감지되면 이동을 즉시 정지한다.

## FR-MOV-03

전방 장애물 감지 후, 좌측 센서를 확인하여 좌측이 비어 있으면 좌회전하고, 좌측이 막혀 있으면 우회전 후 전방 센서로 우측 장애물 유무를 확인하여 방향을 결정한다.

## FR-MOV-04

방향 전환 완료 후 전진을 재개하며 청소를 계속한다.

FR-MOV

# Functional Requirements

## FR-MOV-05

전방·좌측·우측(우회전 후 전방 센서로 확인) 모두에 장애물이 감지되면 후방으로 이동한다.

## FR-MOV-06

후진 완료 후 다시 좌측 및 우측 방향을 판단하여 좌측 또는 우측으로 회전(방향 전환)한다.

## FR-MOV-07

방향 전환 완료 후 전진을 재개하며 청소를 계속한다.

## FR-MOV

# Functional Requirements

## FR-MOV-03 변경 상세

v1 (기준): 전방 장애물 감지 후, 좌측 또는 우측으로 회전(방향 전환)한다.  
→ 좌측 센서와 우측 센서로 각 방향을 직접 확인하여 방향을 결정하였음.

v2 [변경]: 우측 장애물 센서가 없으므로 아래 순서로 방향을 결정한다.

1. 좌측 센서 확인: 좌측이 비어 있으면 좌회전 후 전진 재개 (v1 동작 유지)
2. 좌측이 막혀 있으면: 우측으로 90° 회전 후 전방 센서로 우측 방향 장애물 유무 확인
  - 우측이 비어 있으면: 우회전 완료 후 전진 재개 (우회전 기동 유지)
  - 우측도 막혀 있으면: FR-MOV-05 조건으로 진입

## FR-MOV-05 변경 상세

v1 (기준): 전방·좌측·우측 모두 장애물이 감지되면 후방으로 이동한다.  
→ 우측 센서로 우측 장애물을 직접 감지하여 전방향 막힘을 판단하였음.

v2 [변경]: 우측 장애물 감지를 우회전 후 전방 센서로 수행한다.

전방·좌측이 막혀 있고, 우회전 후 전방 센서가 우측도 막혀 있음을 확인하면 후방으로 이동한다.  
후방 이동 자체는 v1과 동일하게 유지된다.

## FR-MOV-06 변경 상세

v1 (기준): 후진 완료 후 좌측 또는 우측으로 회전(방향 전환)한다.

→ 후진 후 고정된 방향(BACK, 즉 현재 기준 후방 방향으로 180° 회전)으로 전환하였음.

v2 [변경]: 고정 후방 방향(BACK) 회전을 삭제한다.

후진 완료 후 FR-MOV-03의 방향 판단 로직(좌측 센서 확인 → 좌측 막힘 시 우회전 후 전방 센서 확인)을 동일하게 적용하여 좌측 또는 우측으로 회전한다.

**FR-SEN**

# Functional Requirements

## **FR-SEN-01**

RVC는 전방 장애물을 감지하는 센서를 사용한다.

## **FR-SEN-02**

RVC는 좌측 장애물을 감지하는 센서를 사용한다.

## **FR-SEN-03**

RVC는 우측으로 90° 회전한 후 전방 센서를 이용하여 우측 방향의 장애물을 감지한다.

**FR-SEN**

# Functional Requirements

## **FR-SEN-04**

RVC는 먼지를 감지하는 센서를 사용한다.

## **FR-SEN-05**

센서 데이터는 실시간으로 Controller에 전달되어야 한다.

## **FR-SEN-06**

시스템은 센서 추가 또는 교체에 대응할 수 있도록 설계되어야 한다.

FR-SEN

# Functional Requirements

## FR-SEN-03 변경 상세

### [삭제] 구 FR-SEN-03

우측 장애물 감지 센서는 HW에서 제거되었다.

### [추가] 신 FR-SEN-03

우측 방향 감지는 물리적 센서 대신 기동을 이용한다.

- RVC를 우측으로 90° 회전시킨 후,  
기존의 전방 센서를 이용해 해당 방향의 장애물 유무를 확인한다.
- 이를 통해 HW 센서 추가 없이 우측 방향 감지 기능을 유지한다.

**FR-CLN**

# Functional Requirements

**FR-CLN-01**

RVC는 이동 중 지속적으로 청소(흡입·걸레질)를 수행한다.

**FR-CLN-02**

먼지가 감지되면 일정 시간 동안 청소 파워를 증가시킨다.

**FR-CLN-03**

청소 파워 증가 시간이 경과하면 기본 청소 파워로 복귀한다.

**FR-CLN-04**

장애물로 인해 정지 중에도 청소는 중단되지 않는다. (정지는 이동만 정지)

# 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-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**

- 코드 변경 없이 설정 값(파워 증가 시간, 회전 각도 등)을 수정할 수 있어야 한다.
- 설정 파일 분리

**NFR-MNT-03**

- 시스템은 동작 이벤트 로그를 기록하여 디버깅에 활용할 수 있어야 한다.
- 로그 수준 지원

**NFR-EXT**

# Non-Functional Requirements

**NFR-EXT-01**

- 새로운 센서 유형을 추가할 때 기존 코드 수정을 최소화해야 한다. (OCP 준수)
- 기존 모듈 수정 없음

**NFR-EXT-02**

- 새로운 모듈을 추가할 때 핵심 로직 변경이 없어야 한다.
- 인터페이스 분리

**NFR-EXT-03**

- 새로운 모듈을 플러그인 형태로 통합할 수 있어야 한다.
- 의존성 역전 적용

NFR-SAF

# Non-Functional Requirements

## NFR-SAF-01

- 전방·좌측·우측(우회전 후 전방 센서로 확인) 모두에 장애물이 감지되는 경우 RVC는 후방으로 이동하여 탈출을 시도해야 한다.
- 후방 이동 동작 확인

## NFR-SAF-02

- 비정상 센서값(범위 초과 등)은 무시하고 이전 유효 상태를 유지해야 한다.
- 입력 유효성 검사

## [변경] NFR-SAF-01

v1 (기존): "모든 방향(전방·좌·우)에 장애물이 있는 경우"

→ 우측 센서로 우측 장애물을 직접 확인하였음.

v2 [변경]: 우측 장애물은 우회전 후 전방 센서로 확인한다.

→ 전방 막힘, 좌측 막힘, 우회전 후 전방 센서가 우측도 막힘을 확인한 경우 후방 이동을 수행한다.

→ 후방 이동 후 재방향 판단을 통해 탈출하는 전체 흐름은 v1과 동일하게 유지된다.

**NFR-PRT**

# **Non-Functional Requirements**

**NFR-PRT-01**

- Controller 핵심 로직은 특정 HW 플랫폼에 종속되지 않아야 한다.
- HW 추상화 레이어 분리

**NFR-PRT-02**

- HW 제어 인터페이스는 추상화된 API를 통해서만 접근해야 한다.
- 직접 HW 접근 금지

# Vibe Coding

OOAD: Use-Case

# Use-Case

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

이를 기반으로 v1의 usecase를 수정해서 v2 디렉토리에 문서를 생성. 추가/변경/삭제 태그 필수로 삽입.

- docs/v2/use\_cases.md 생성 완료했습니다.

## v1 → v2 변경 내용 요약:

구분	대상	변경 내용
[변경]	Actor: Obstacle Sensor	전방·좌측만 HW 센서. 우측은 기동으로 대체
[변경]	UC-03 Avoid Front Obstacle	방향 결정 순서 변경 — ① 좌측 센서 확인 → ② 좌측 막힘 시 우회전+전방 센서로 우측 확인 → ③ 우측도 막히면 UC-05
[변경]	UC-04 Detect Side Obstacles	좌측은 HW 직접 감지, 우측은 우회전+전방 센서 능동 감지로 변경
[변경]	UC-05 Avoid All-Direction Obstacle	① 트리거 조건 변경 (우측 확인이 UC-03 흐름 내에서 발생), ② 후진 후 고정 BACK 180° 회전 삭제 → 재판단 후 좌/우 회전
—	UC-01, 02, 06, 07, 08	변경 없음

# 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 Side Obstacles
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

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

# Use Case

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

# Use Case

<b>Use-Case</b>	<b>3. Avoid Front Obstacle</b>
<b>Actors</b>	<b>RVC System</b>
<b>Summary</b>	<p>전방 장애물 감지 시 RVC는 이동을 정지하고, 좌측 또는 우측으로 방향을 전환한 뒤 전진을 재개하여 청소를 계속한다. 방향 전환 방향(좌/우)은 다음 순서로 결정된다.</p> <ol style="list-style-type: none"><li>1. 좌측 센서를 확인하여 좌측이 비어 있으면 좌회전한다.</li><li>2. 좌측이 막혀 있으면 우회전 후 전방 센서로 우측 방향 장애물 유무를 확인한다.<ul style="list-style-type: none"><li>• 우측이 비어 있으면 우회전을 완료하고 전진을 재개한다.</li><li>• 우측도 막혀 있으면 UC-05로 진입한다.</li></ul></li></ol>

## v1과의 차이:

- > v1에서는 좌측 센서와 우측 센서를 동시에 조회하여 방향을 결정하였다.
- > v2에서는 우측 HW 센서가 없으므로, 우측 방향 확인이 필요한 경우  
우회전 후 전방 센서를 이용하여 우측 장애물 유무를 판단한다. (FR-MOV-03 변경, FR-SEN-03 추가)

# Use Case

<b>Use-Case</b>	<b>4. Detect Side Obstacles</b>
<b>Actors</b>	<b>Obstacle Sensor → RVC System</b>
<b>Summary</b>	<p>측면 장애물 정보를 Controller에 전달한다. UC-03 실행 중 방향 결정 및 UC-05 진입 조건 판단에 활용된다. 좌측과 우측 감지 방식이 서로 다르다.</p> <ul style="list-style-type: none"> <li>• 좌측 감지: 좌측 장애물 센서(HW)가 장애물을 직접 감지하여 Controller에 전달한다.</li> <li>• 우측 감지: 우측 HW 센서가 없으므로, RVC가 우측으로 90° 회전한 후 전방 센서를 이용하여 우측 방향의 장애물 유무를 확인한다.</li> </ul>

## v1과의 차이

- > v1에서는 좌측·우측 두 HW 센서가 각각 독립적으로 장애물을 감지하였다. (FR-SEN-02, FR-SEN-03)
- > v2에서는 우측 HW 센서(구 FR-SEN-03)가 삭제되고, 우회전 후 전방 센서를 이용한 우측 감지 방식(신 FR-SEN-03)으로 대체된다.
- > UC의 이름은 유지되나, 우측 감지 방식이 수동 센서 조회에서 능동 기동(우회전 + 전방 센서 읽기)으로 변경된다.

# Use Case

<b>Use-Case</b>	<b>5. Avoid All-Direction Obstacle</b>
<b>Actors</b>	<b>RVC System</b>
<b>Summary</b>	<b>전방·좌측·우측 모두 장애물이 감지된 경우 RVC는 후방으로 이동한 뒤, 좌측 및 우측 방향을 다시 판단하여 회전하고 전진을 재개하여 청소를 계속한다.</b>

## v1과의 차이 (두 가지)

- > 1. 트리거 조건 변경: v1에서는 전방·좌·우 HW 센서가 동시에 장애물을 감지하여 진입하였다. v2에서는 우측 확인이 우회전 후 전방 센서로 이루어지므로, UC-03의 방향 판단 흐름 내에서 우측도 막힘이 확인된 경우 본 UC로 진입한다.
- > 2. 후진 후 회전 방향 변경: v1에서는 후진 완료 후 고정된 후방 방향(`BACK`, 180° 회전)으로 전환하였다. v2에서는 고정 BACK 방향 회전을 삭제하고, 후진 완료 후 UC-03의 방향 판단 로직(좌측 센서 확인 → 좌측 막힘 시 우회전 후 전방 센서 확인)을 동일하게 적용하여 좌측 또는 우측으로 회전한다. (FR-MOV-06 변경)

# Use Case

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

# Use Case

<b>Use-Case</b>	<b>7. Boost Cleaning Power</b>
<b>Actors</b>	<b>RVC System</b>
<b>Summary</b>	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

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

# Vibe Coding

OOAD: Use-Case Diagram

# Use-Case Diagram

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

이를 기반으로 한 use case diagram을 출력. v2 디렉토리 내 diagrams라는 디렉토리를 만들고 보관.

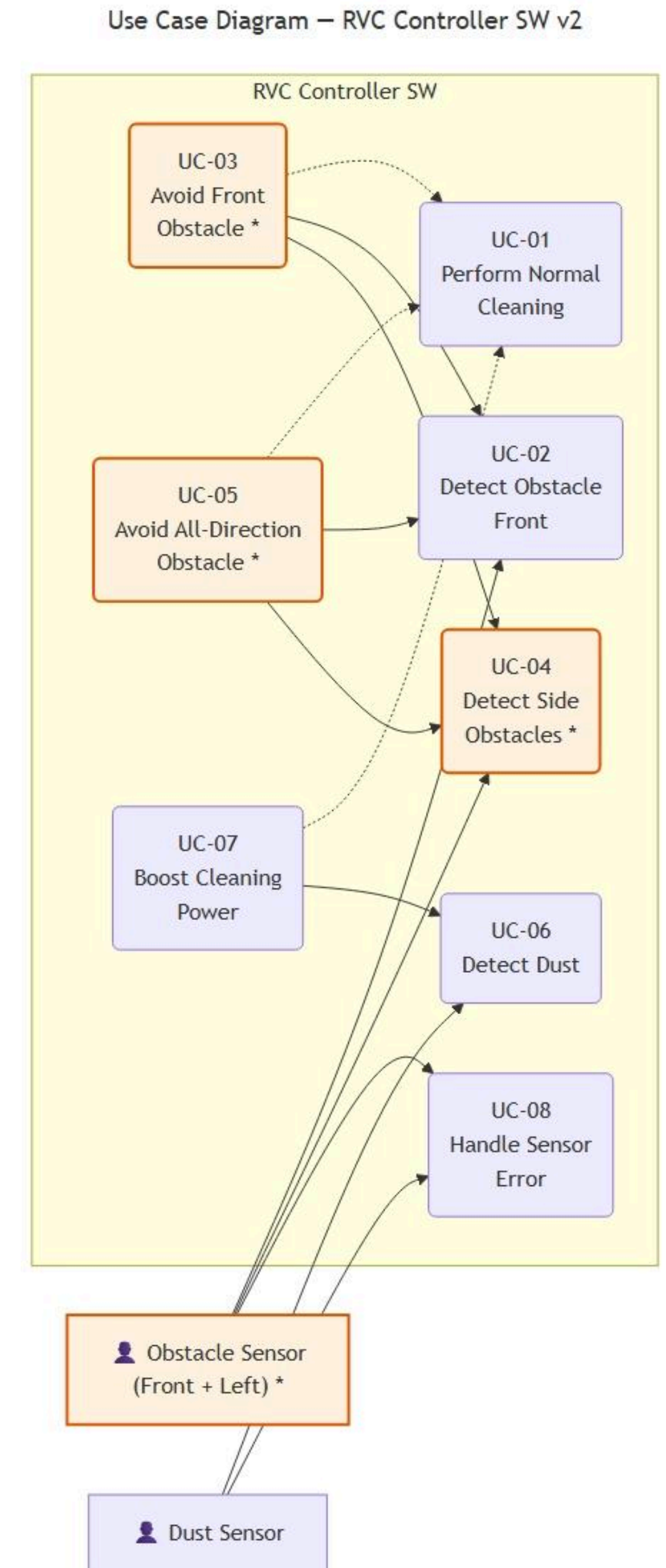
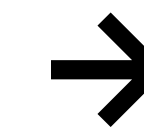
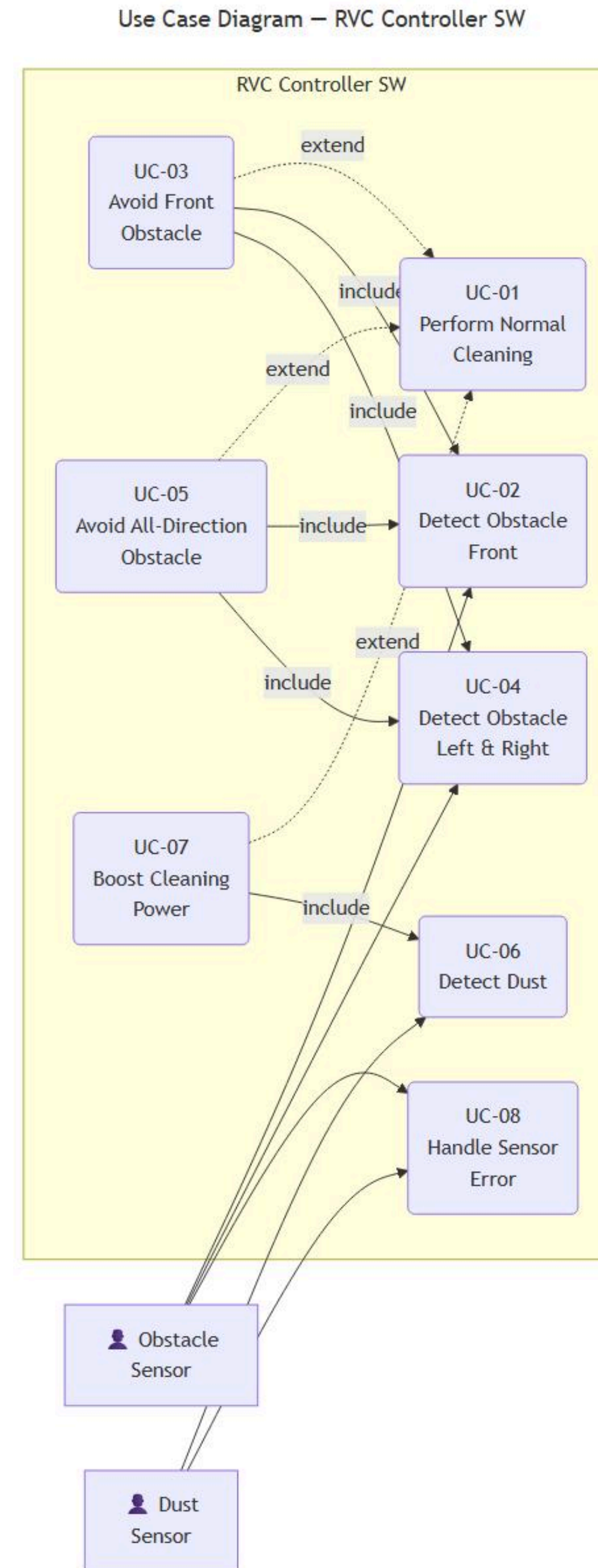
- docs/v2/diagrams/use\_case\_diagram.mmd 생성 완료했습니다.

v1 대비 다이어그램 변경 사항:

구분	대상	변경 내용
[변경]	<b>Obstacle Sensor</b> (Actor)	레이블에 (전방·좌측) 명시, 우측 HW 센서 제거 반영, 주황 테두리 강조
[변경]	<b>UC-03</b> Avoid Front Obstacle	* 마커 + 주황 강조 — 방향 결정 로직 변경
[변경]	<b>UC-04</b> Detect Obstacle Left & Right → <b>Detect Side Obstacles</b>	우측 HW 감지 → 우회전+전방 센서 감지 방식 반영, 명칭 변경
[변경]	<b>UC-05</b> Avoid All-Direction Obstacle	* 마커 + 주황 강조 — 트리거 조건 변경 + BACK 고정 회전 삭제
—	UC-01, 02, 06, 07, 08	변경 없음, 기본 스타일 유지

구조적 관계(include/extend)는 v1과 동일하며, 변경은 UC 내부 로직과 Actor 범위에 국한됩니다.

# Use-Case Diagram



# Vibe Coding

OOAD: System Sequence Diagram

# System Sequence Diagram

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

오른쪽 센서가 사라진 것을 반영해 지금까지의 V1, V2 문서들을 기반으로 기존의 V2 SSD는 무시하고 System Sequence Diagram을 생성. 이때 V1의 SSD에서 추가/변경/삭제 된 부분은 명시할 것.

● 생성 완료했습니다.

생성 파일:

- [docs/v2/system\\_sequence\\_diagram.md](#)
- docs/v2/diagrams/ssd\_01.mmd~08.mmd + .png (8개)

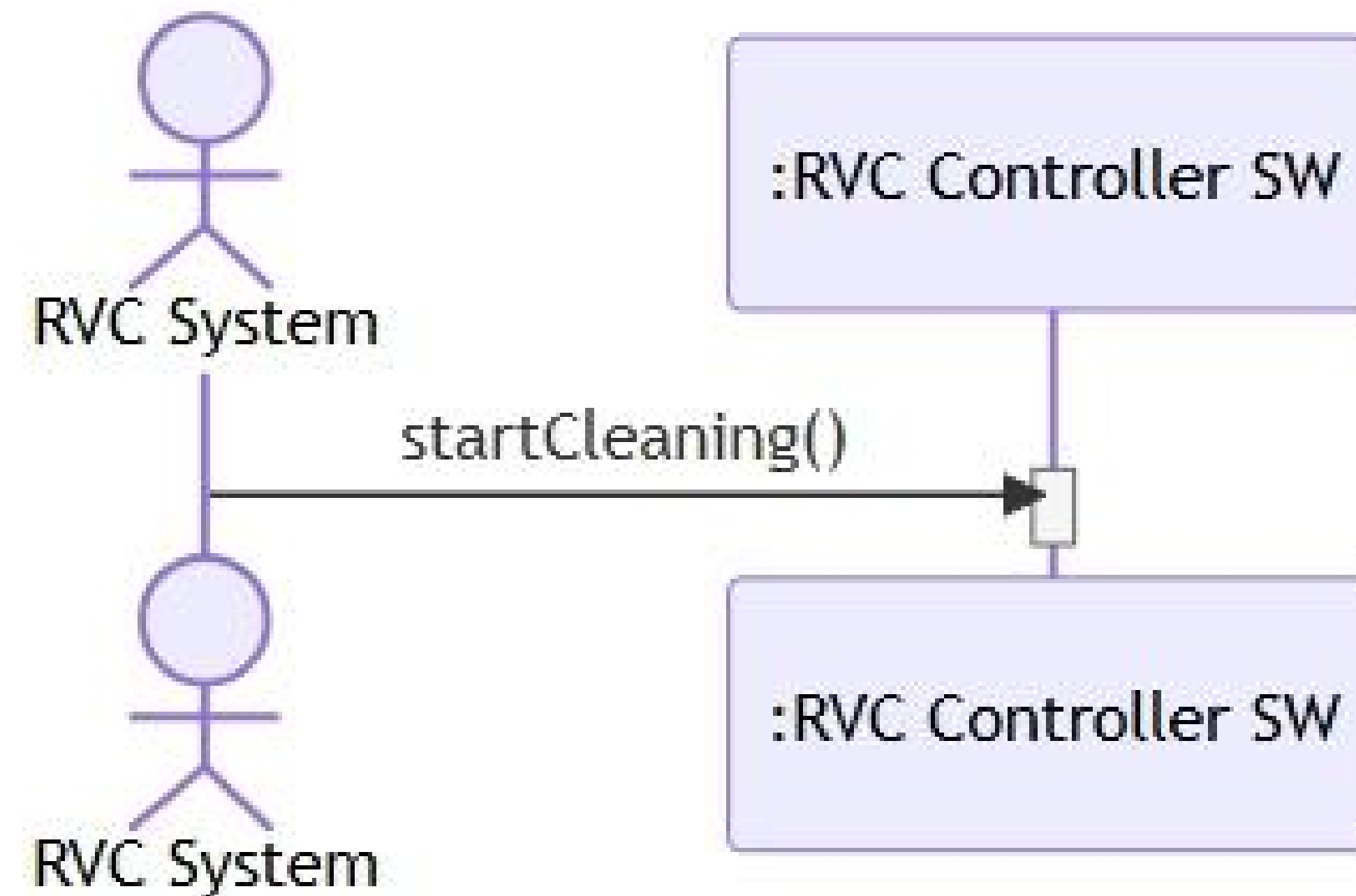
v1 → v2 핵심 변경 요약:

SSD	구분	변경 내용
SSD-03	[변경]	checkSideObstacles(L+R) → ① checkLeftSensor() → ② 좌측 막힘 시 turnRight90() + checkFrontSensor() 2단계로 분리. 주황 rect 로 추가 로직 시각화
SSD-04	[삭제]	reportObstacle(position: RIGHT) 삭제. Note로 삭제 이유 및 대체 방법 명시
SSD-05	[변경]	turnToFreeDirection(BACK) 삭제. 후진 후 UC-03와 동일한 2단계 방향 판단 (checkLeftSensor → turnRight90+checkFrontSensor) 적용
SSD-01,02,06,07,08	—	변경 없음, v1과 동일

## System Sequence Diagram

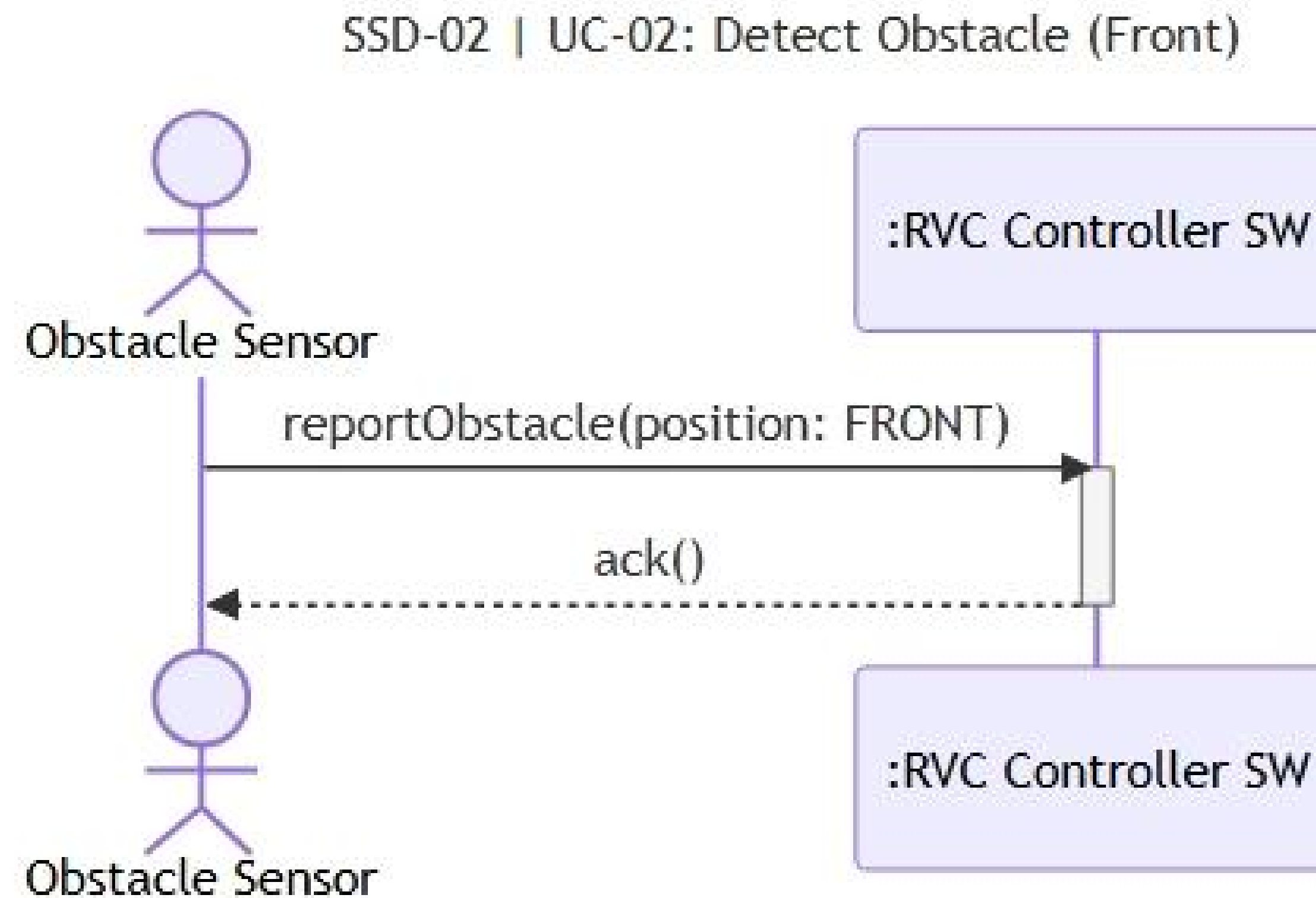
# SSD-01 : UC-01 Perform Normal Cleaning

SSD-01 | UC-01: Perform Normal Cleaning



## System Sequence Diagram

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

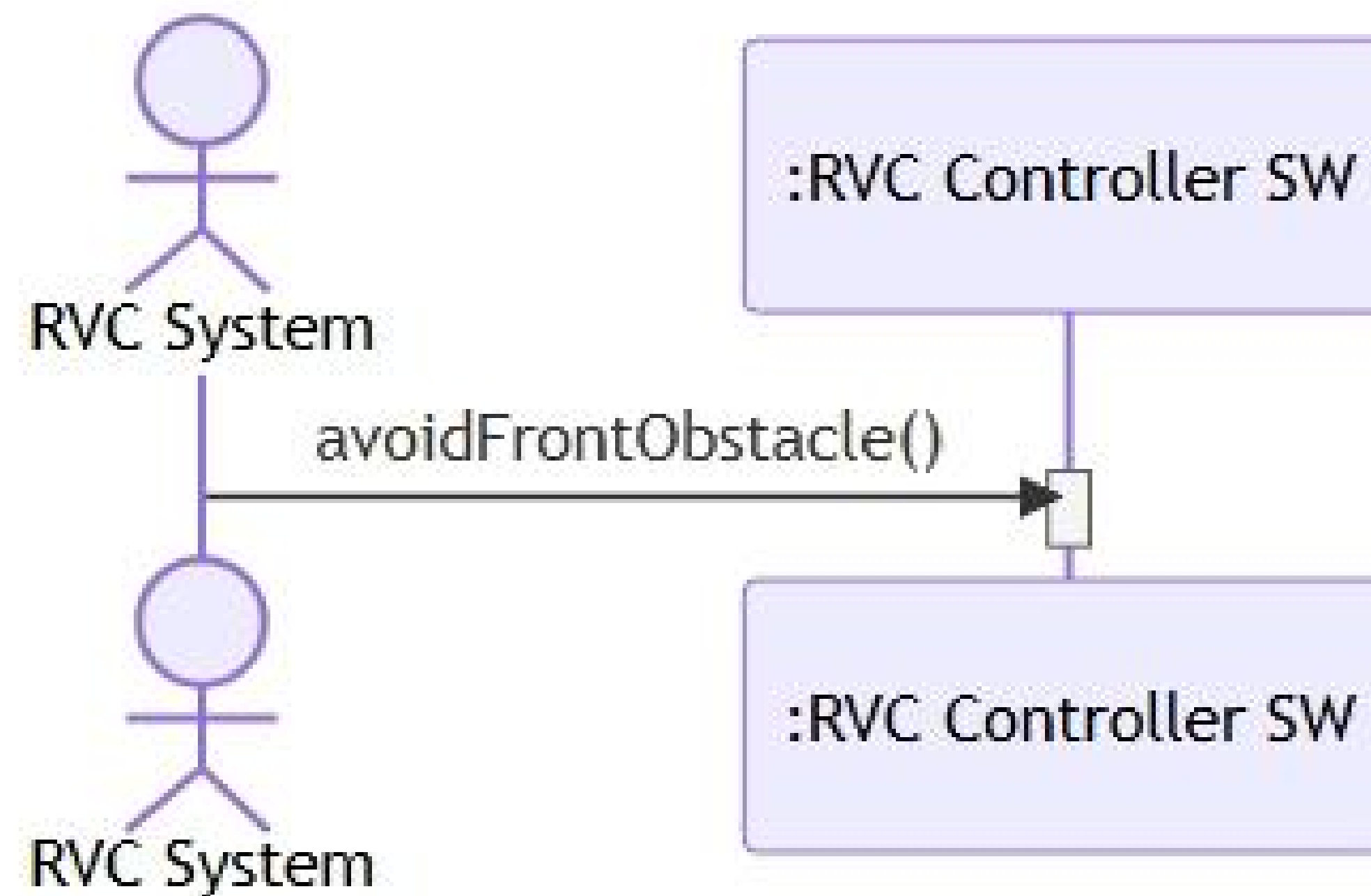


## System Sequence Diagram

# SSD-03 : UC-03 Avoid Front Obstacle

- ``checkSideObstacles(L+R)`` → ``checkLeftSensor()`` + ``turnRight90()`` + ``checkFrontSensor()``

SSD-03 | UC-03: Avoid Front Obstacle

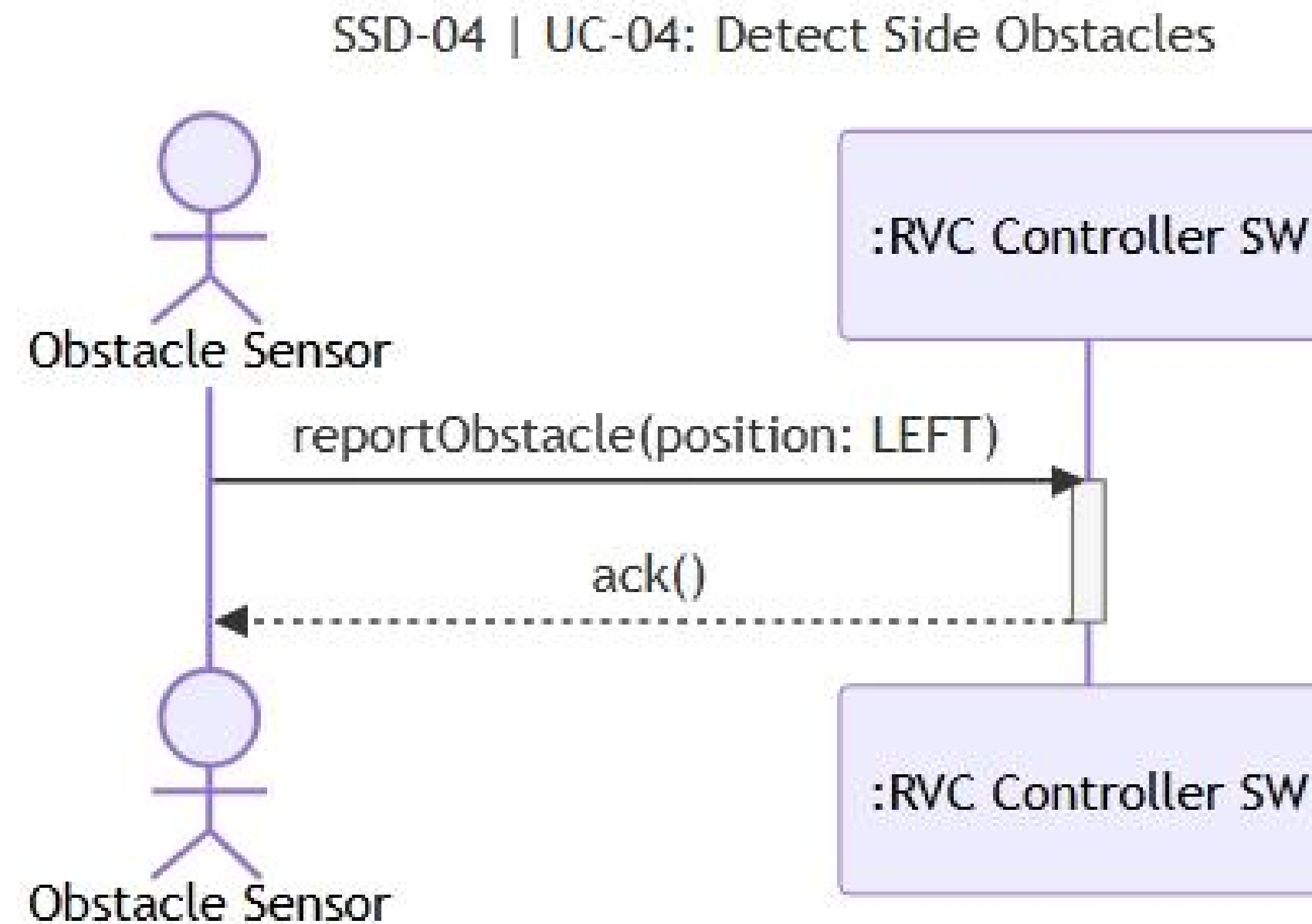


## System Sequence Diagram

# SSD-04 : UC-04 Detect Side Obstacles

## ~~Detect Obstacle (Left & Right)~~

- `reportObstacle(RIGHT)` [삭제] — 우측 HW 센서 제거

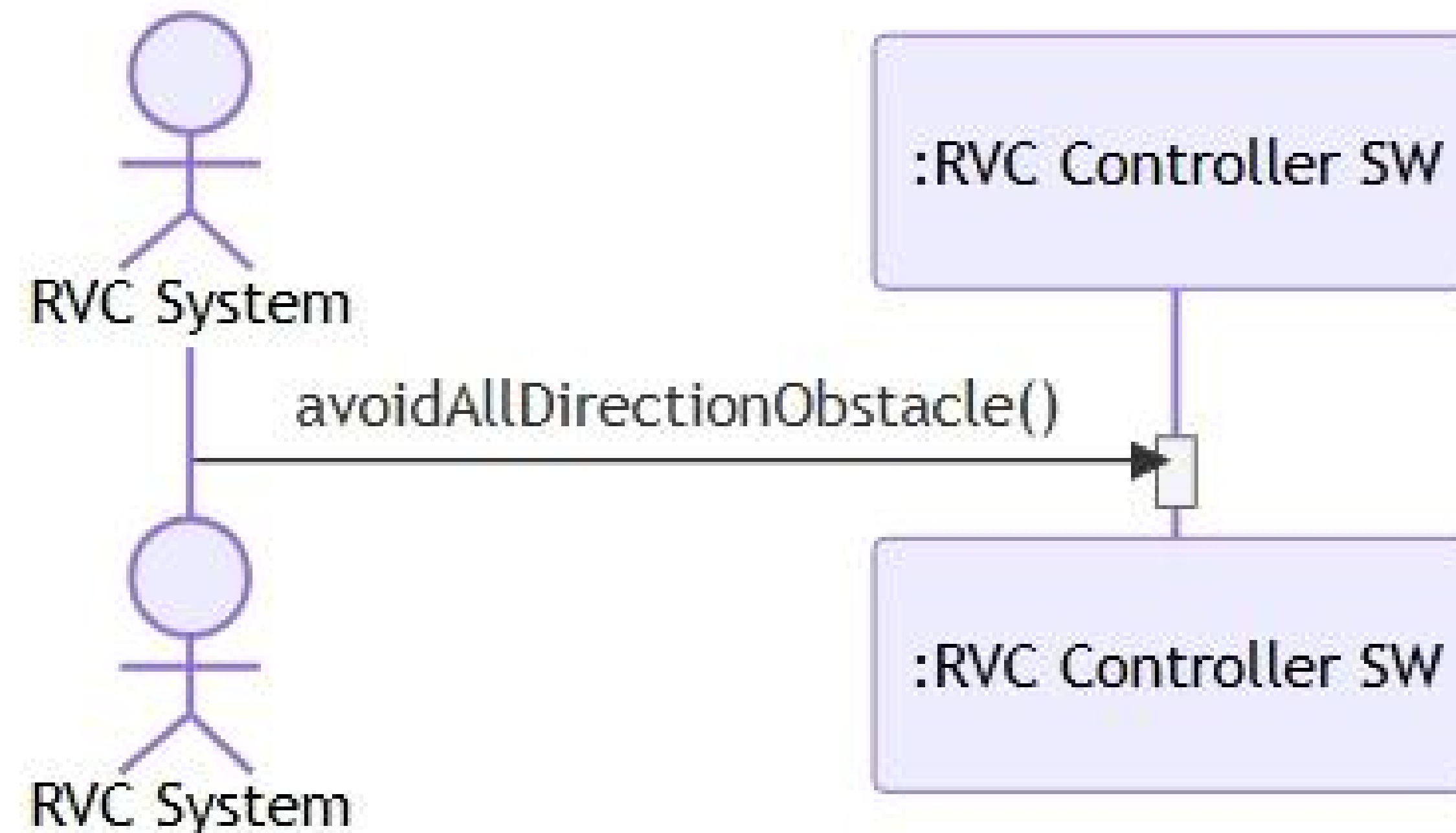


## System Sequence Diagram

# SSD-05 : UC-05 Avoid All-Direction Obstacle

- `turnToFreeDirection(BACK)` [삭제] → 후진 후 UC-03 방향 판단 로직 재적용

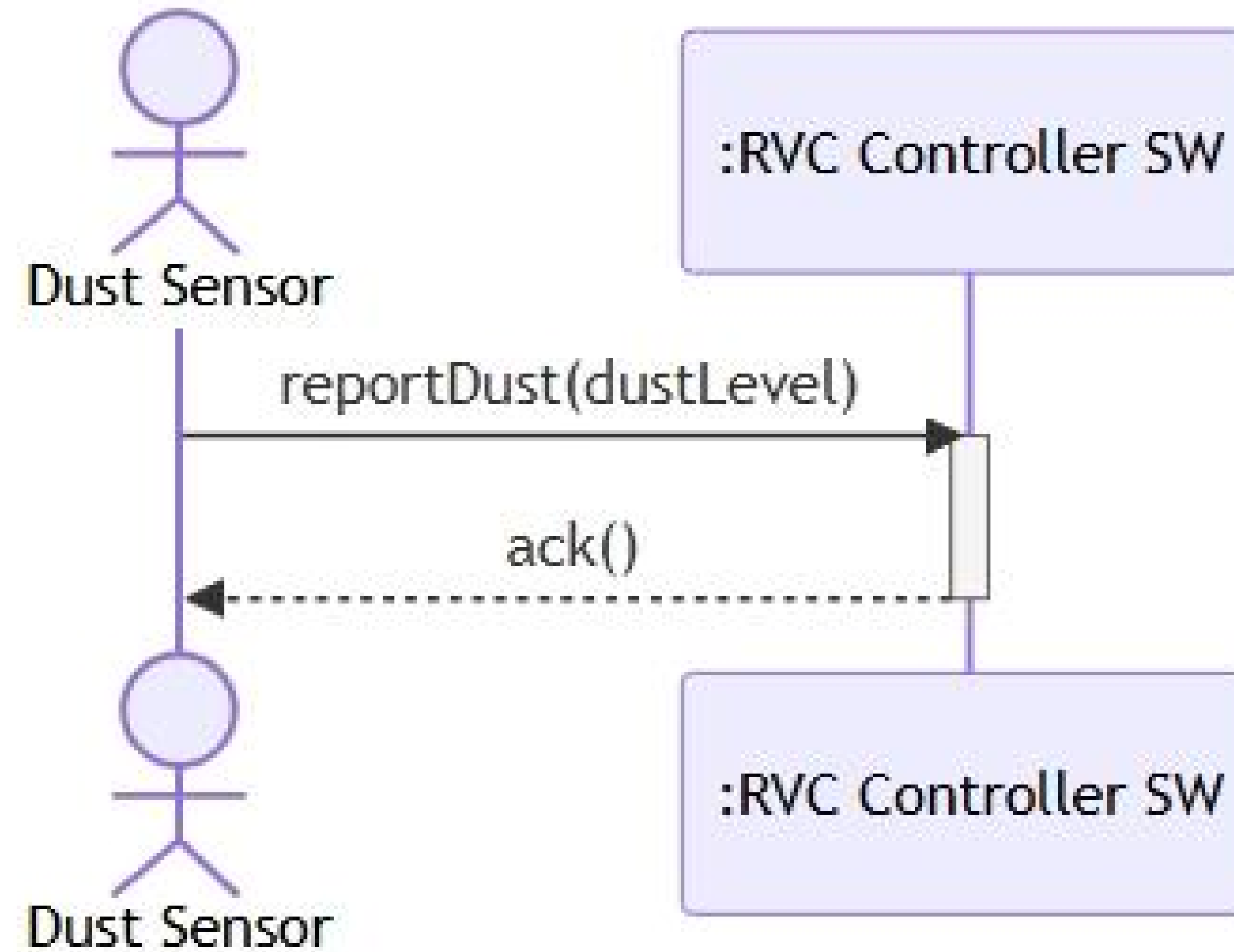
SSD-05 | UC-05: Avoid All-Direction Obstacle



## System Sequence Diagram

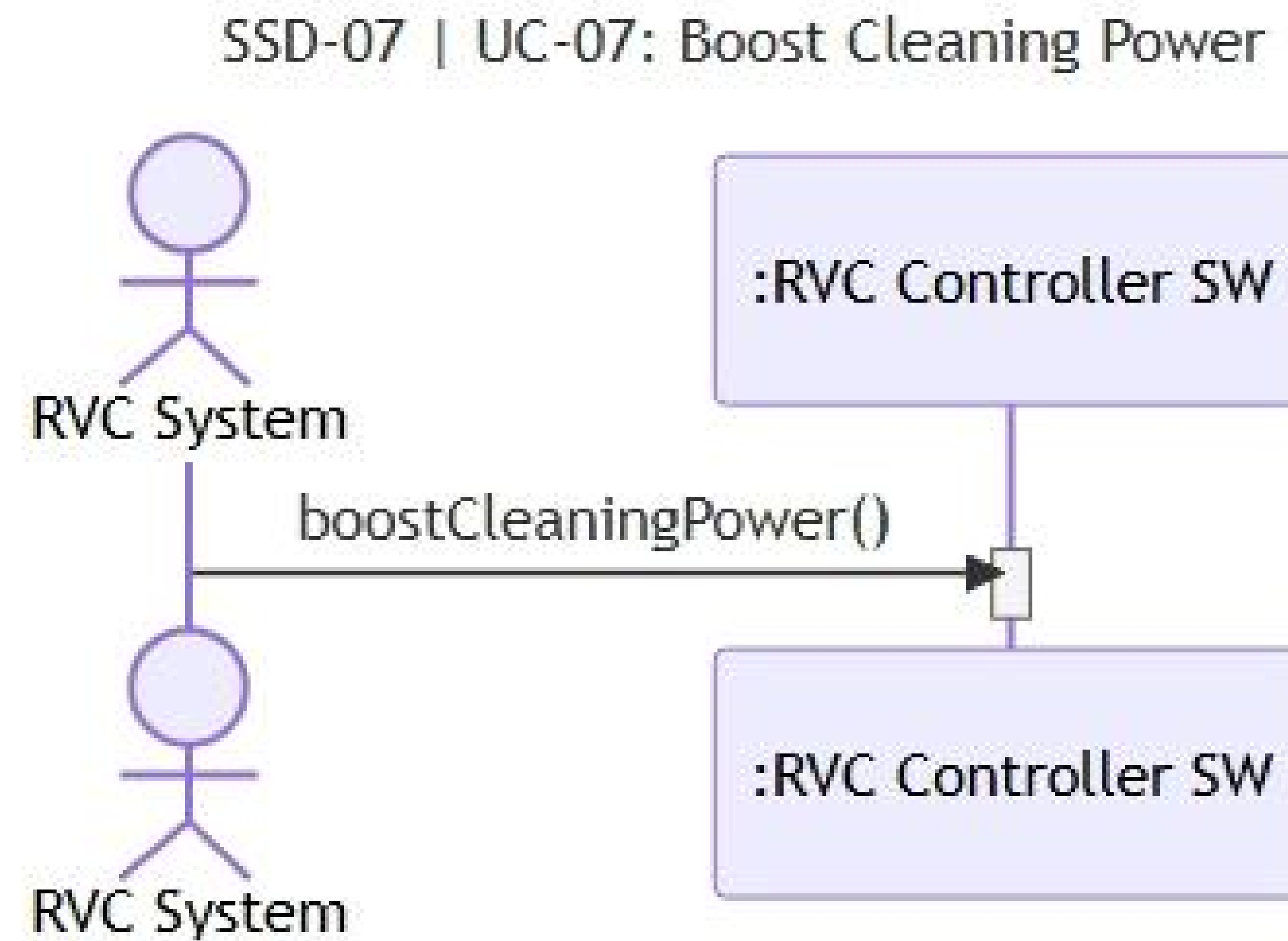
# SSD-06 : UC-06 Detect Dust

SSD-06 | UC-06: Detect Dust



## System Sequence Diagram

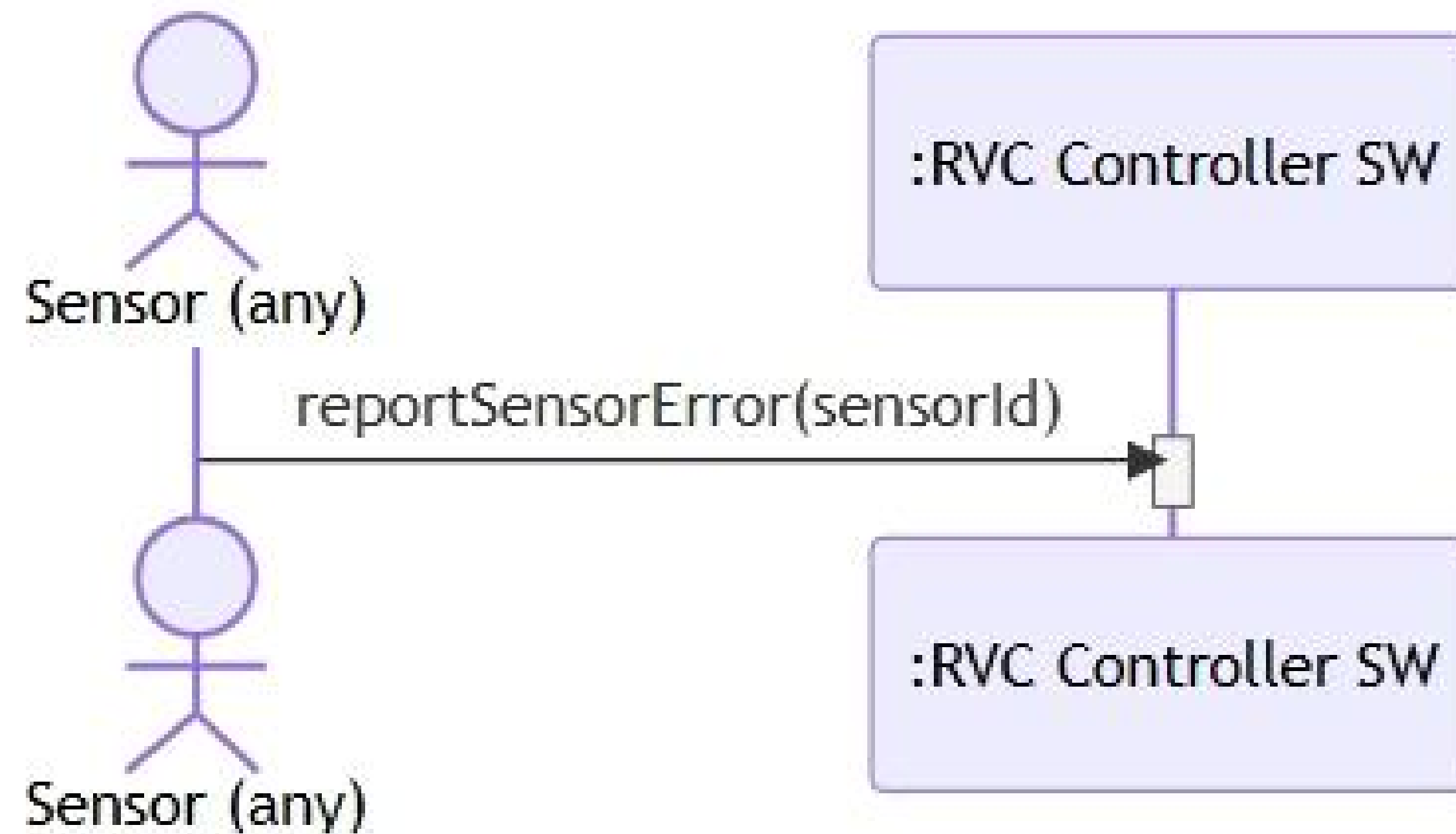
# SSD-07 : UC-07 Boost Cleaning Power



## System Sequence Diagram

# SSD-08 : UC-08 Handle Sensor Error

SSD-08 | UC-08: Handle Sensor Error



# Vibe Coding

OOAD: Domain Model

# Domain Model

오른쪽 센서가 사라진 것을 반영해 지금까지의 V1, V2 문서들(SSD 제외)을 기반으로 Domain Model을 생성. 이때 V1의 domain Model에서 추가/변경/삭제 된 부분은 명시할 것.

| 항목 | 내용 |

- 생성 완료했습니다.

생성 파일:

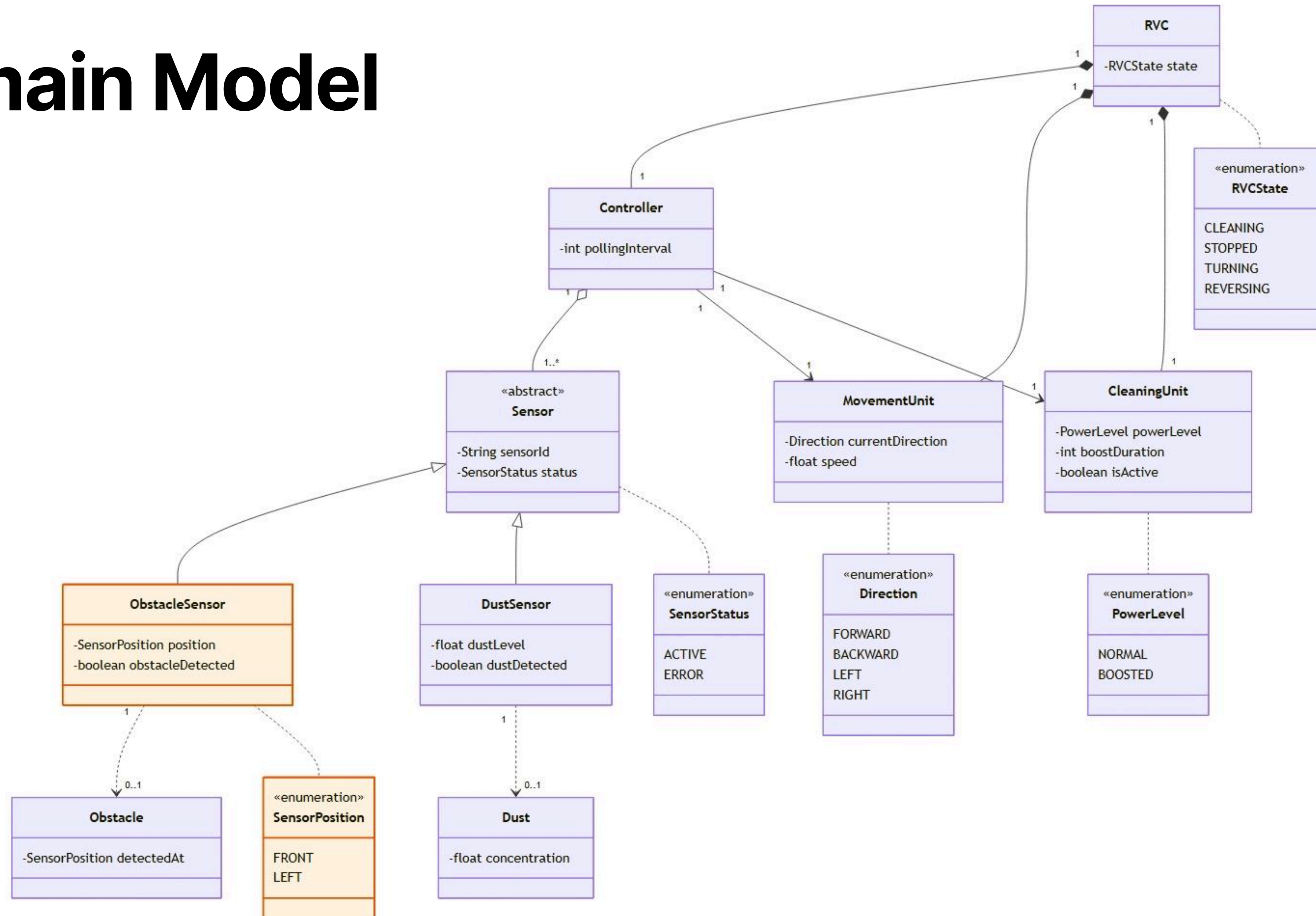
- [docs/v2/domain\\_model.md](#)
- [docs/v2/diagrams/domain\\_model.mmd](#)
- [docs/v2/diagrams/domain\\_model.png](#)

v1 → v2 변경 내용:

구분	대상	내용
[삭제]	SensorPosition .RIGHT	우측 HW 센서 없음 → 열거값 제거. 다이어그램에서 주황 강조
[변경]	ObstacleSensor	인스턴스 3개(FRONT·LEFT·RIGHT) → 2개(FRONT·LEFT). 주황 테두리로 표시
[변경]	DR-01	인스턴스 수 3 → 2 반영
[변경]	DR-06	전방향 막힘 조건: 우측 센서 직접 감지 → 우회전+전방 센서 확인 후 REVERSING 전환
[추가]	DR-07	Controller가 MovementUnit(우회전 90°) + FRONT ObstacleSensor 재조회로 우측 감지하는 도메인 규칙 신규 도출

Domain Model – RVC Controller SW v2

# Domain Model



# Vibe Coding

OOAD: Sequence Diagram

# Sequence Diagram

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

오른쪽 센서가 사라진 것을 반영해 지금까지의 V1, V2 문서들을 기반으로 Sequence Diagram을 생성. 이때 V1의 SSD에서 추가/변경/삭제 된 부분은 명시할 것.

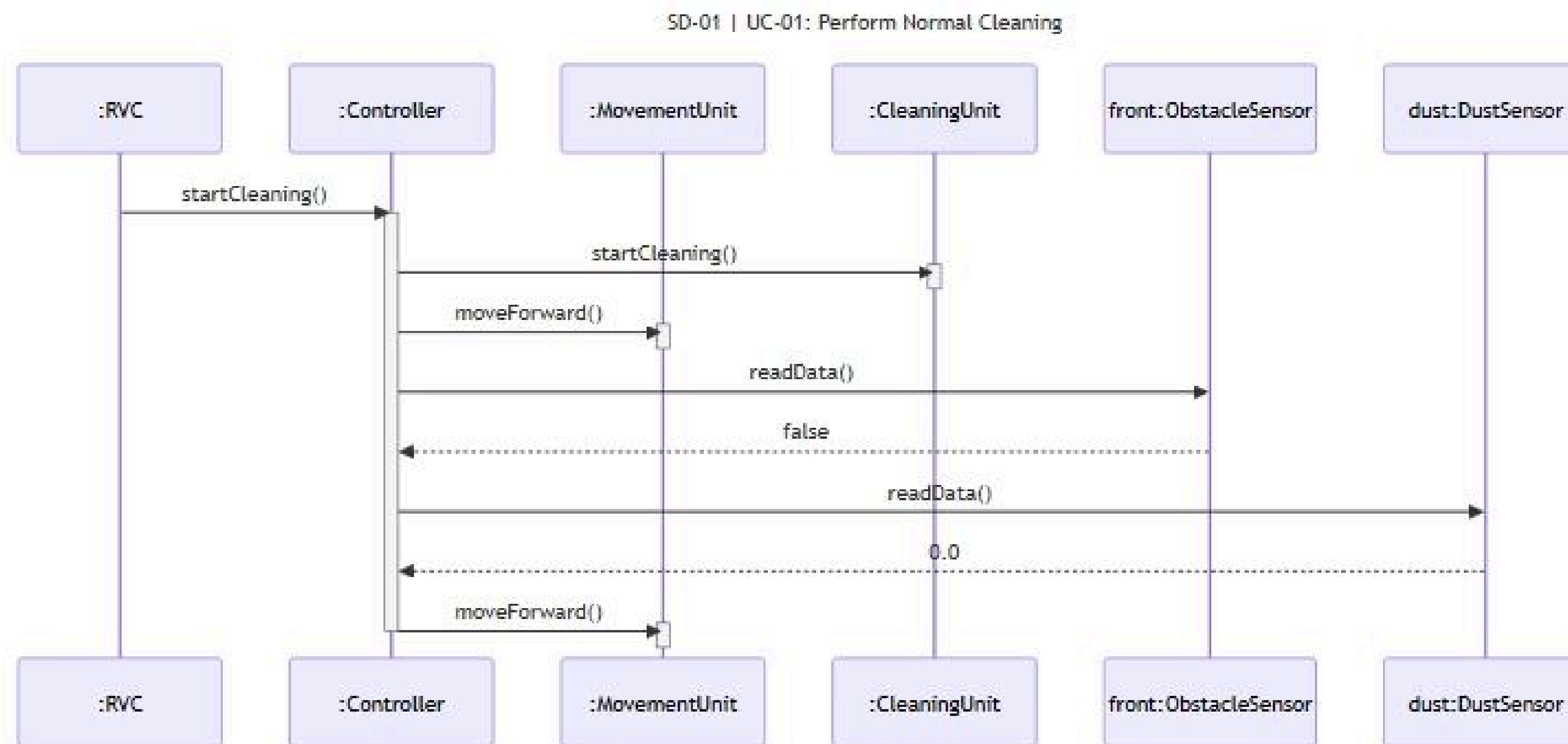
항목	내용
•	생성 완료했습니다.
	생성 파일:
	<ul style="list-style-type: none"> <li>• <a href="#">docs/v2/sequence_diagram.md</a></li> <li>• docs/v2/diagrams/sd_01.mmd~08.mmd + .png (8개)</li> </ul>

---

**v1 → v2 핵심 변경 요약:**

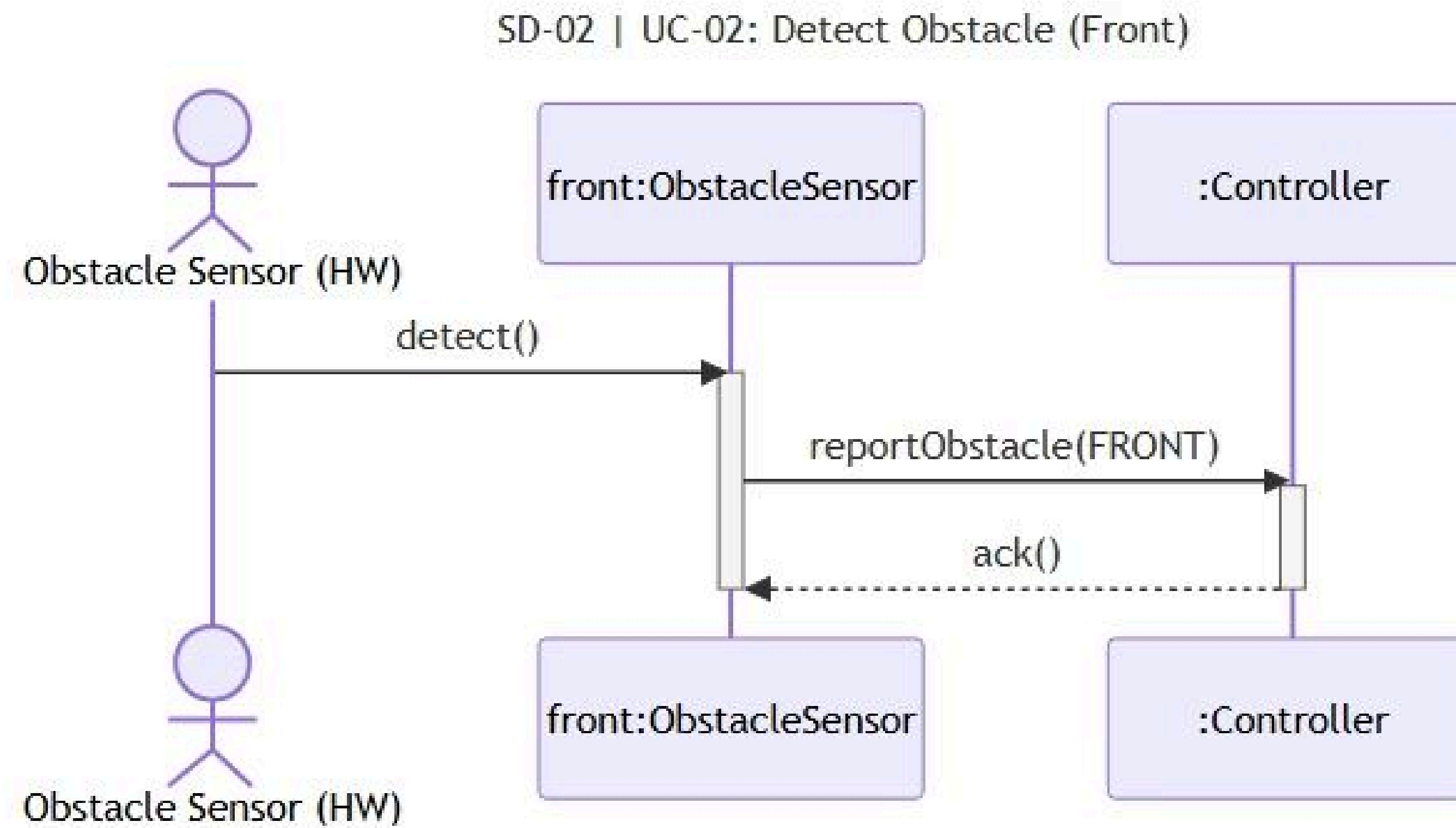
SD	구분	변경 내용
SD-03	[변경]	right:ObstacleSensor [삭제] → front:ObstacleSensor [추가] / 방향 결정을 ① 좌측 확인 → ② 좌막힘 시 turn(RIGHT) + front:FS.isObstacleDetected() 2단계로 변경
SD-04	[변경]	right:ObstacleSensor 참여자 전면 삭제 (detect, reportObstacle(RIGHT), recordObstacleDetection(RIGHT), ack 5개 메시지 모두 삭제)
SD-05	[변경]	turn(BACK) [삭제] → left:ObstacleSensor + front:ObstacleSensor [추가] / 후진 후 UC-03과 동일한 2단계 방향 재판단 로직
SD-01,02,06,07,08	—	변경 없음

## Sequence Diagram

**SD-01 : UC-01 Perform Normal Cleaning**

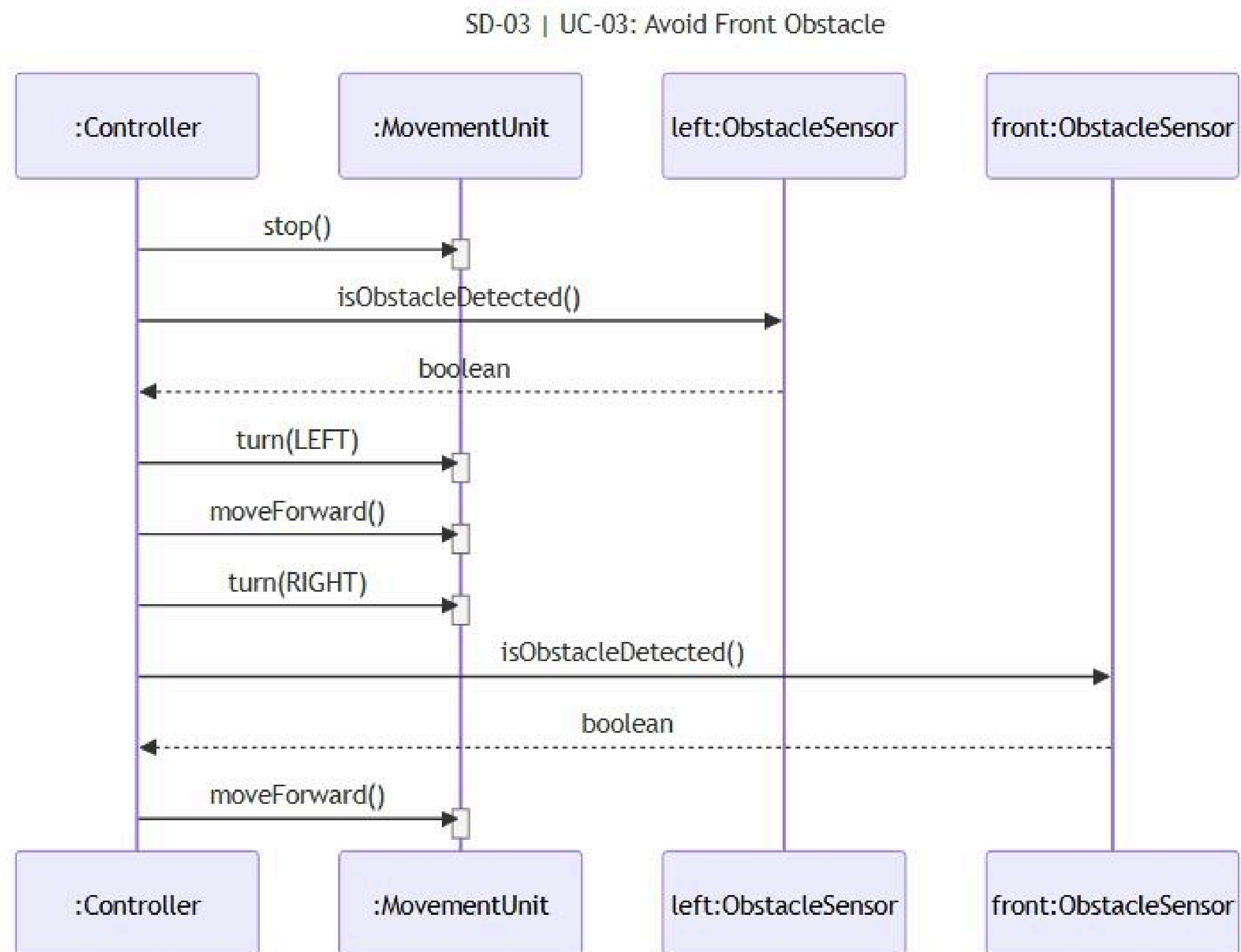
## Sequence Diagram

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



## Sequence Diagram

## SD-03 : UC-03 Avoid Front Obstacle

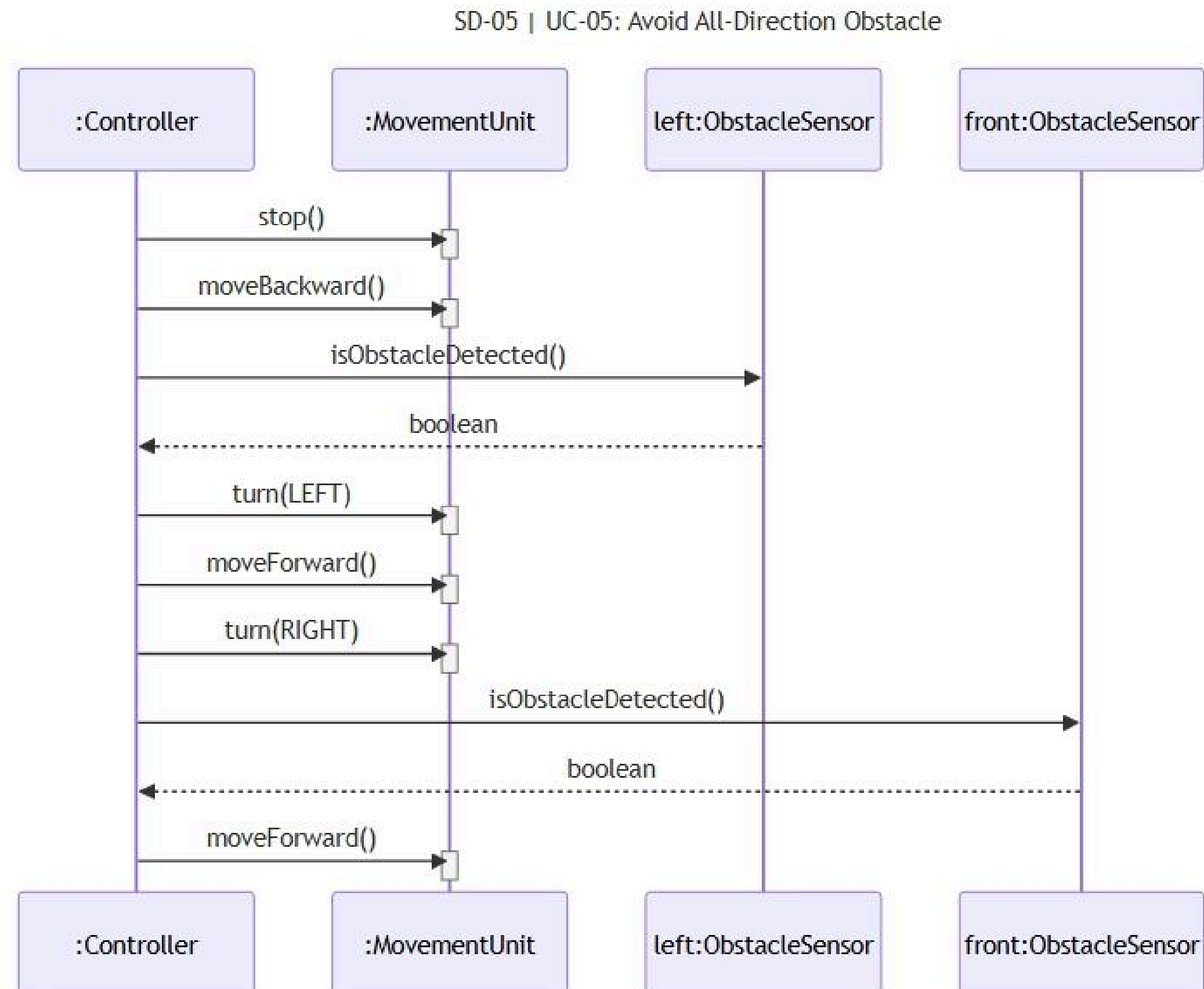


- 'right:ObstacleSensor' 삭제
- 'front:ObstacleSensor' 추가
- 방향 결정 2단계 로직으로 변경



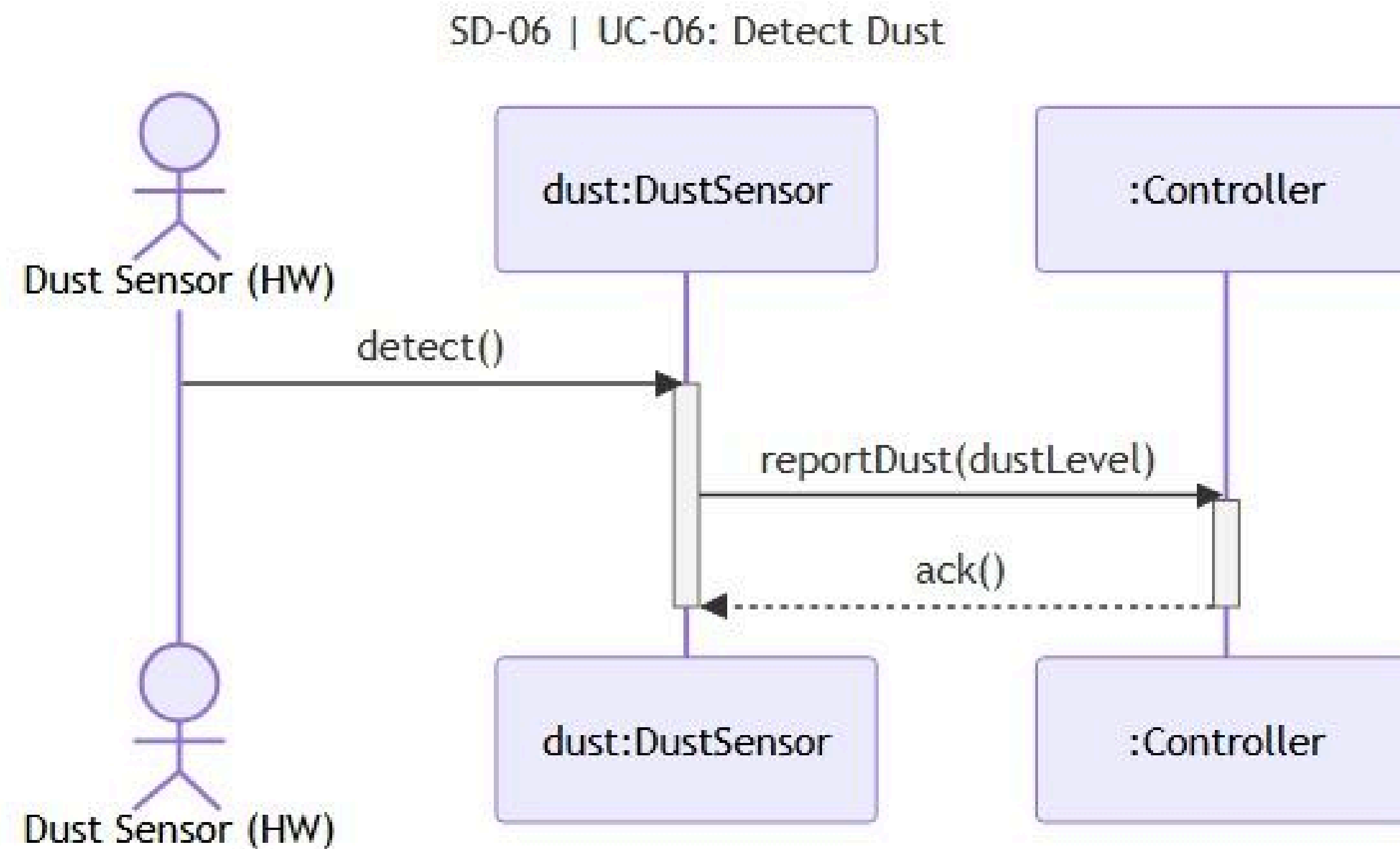
## Sequence Diagram

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



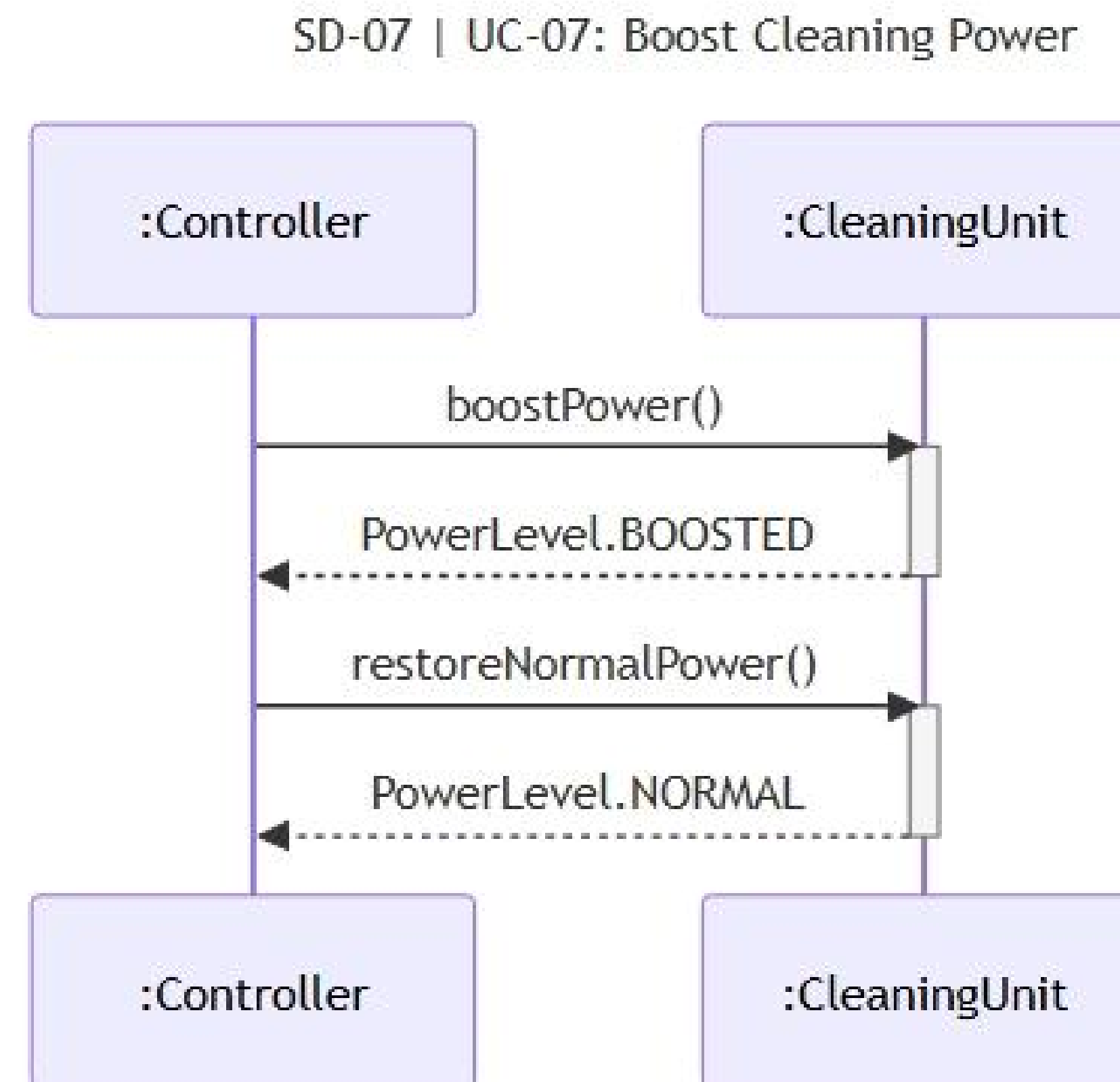
- `turn(BACK)` 삭제
- `left:ObstacleSensor` + `front:ObstacleSensor` 추가
- 방향 재판단 로직 추가

## Sequence Diagram

**SD-06 : UC-06 Detect Dust**

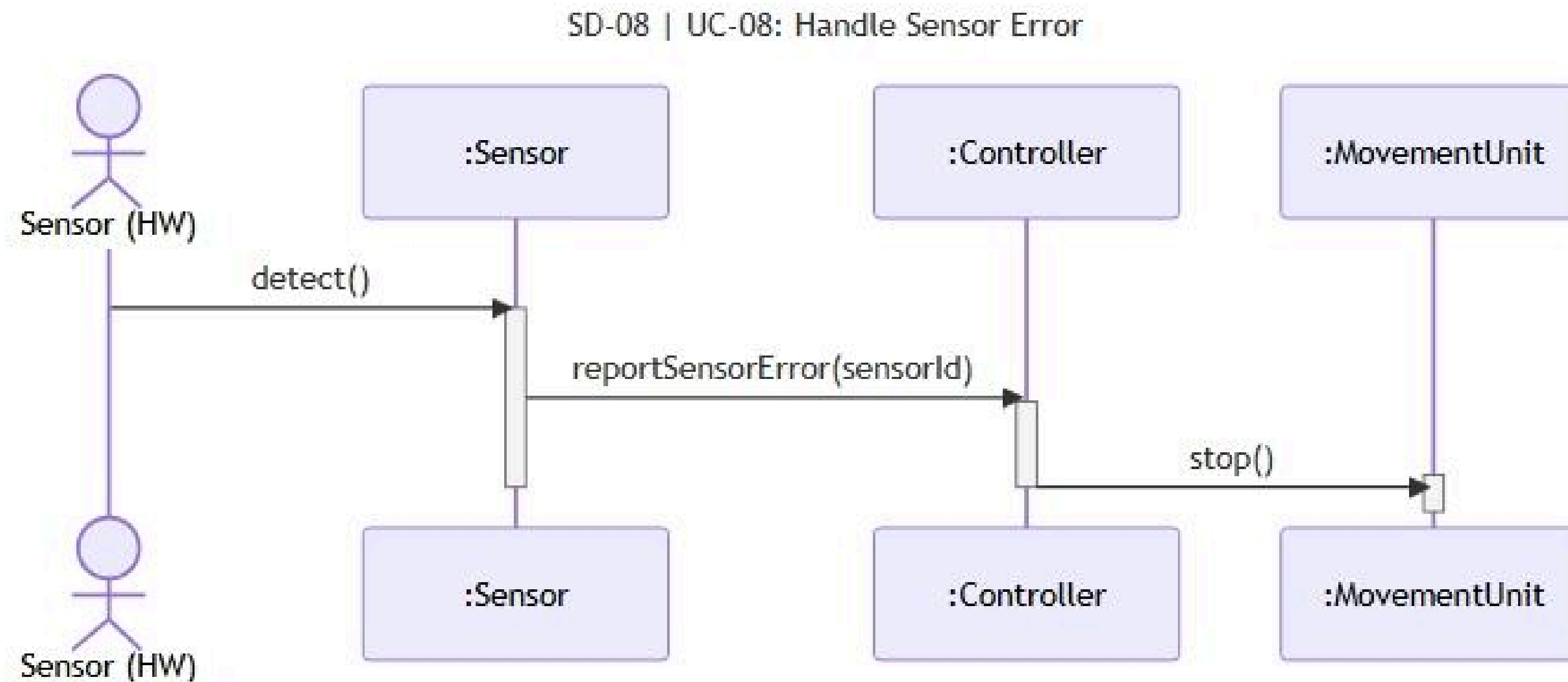
## Sequence Diagram

# SD-07 : UC-07 Boost Cleaning Power



## Sequence Diagram

# SD-08 : UC-08 Handle Sensor Error



# Vibe Coding

OOAD: Class Diagram

# Class Diagram

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

오른쪽 센서가 사라진 것을 반영해 지금까지의 V1, V2 문서들을 기반으로 Class Diagram을 생성. 이때 V1의 Class Diagram에서 추가/변경/삭제 된 부분은 명시할 것.

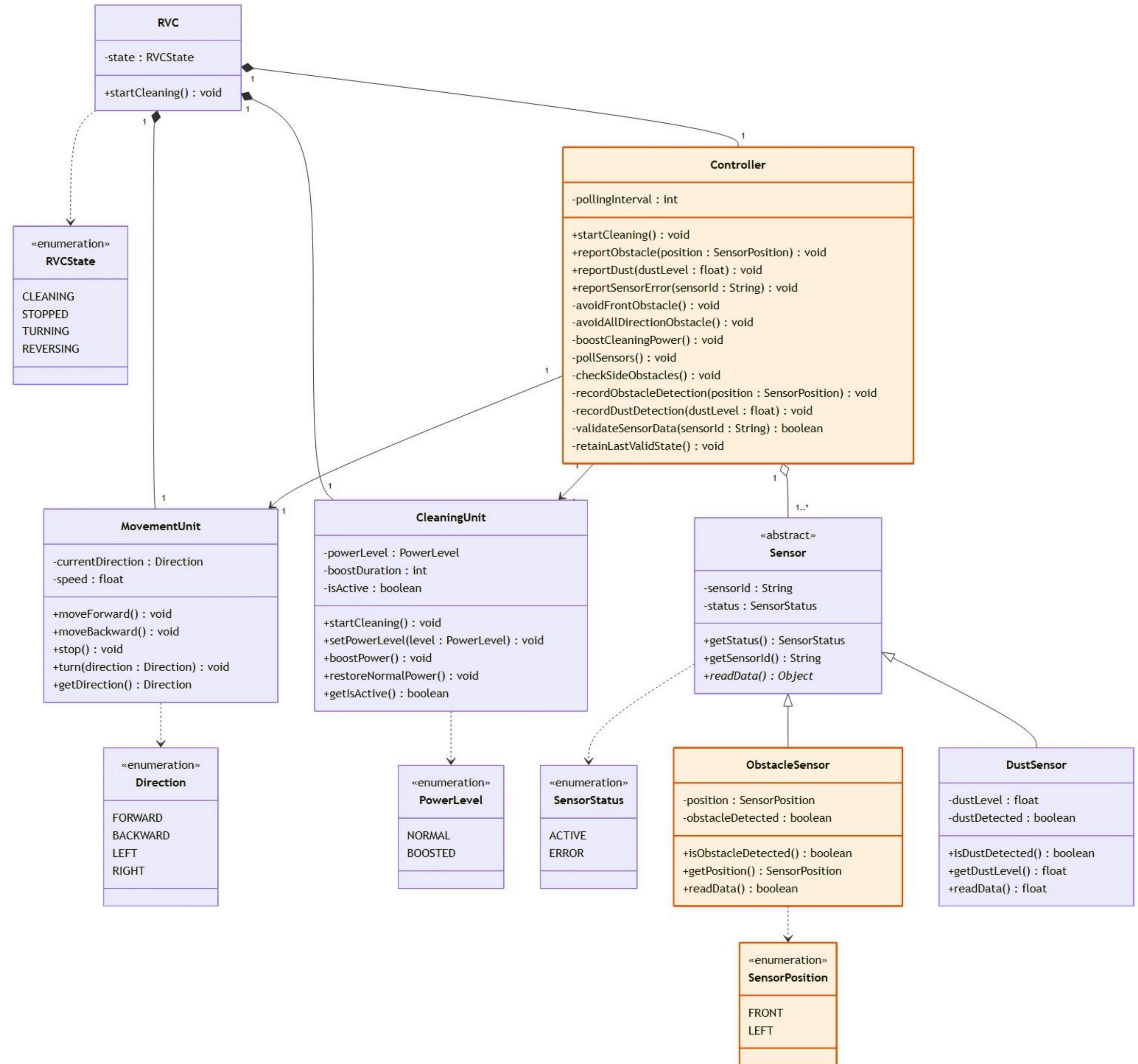
- [docs/v2/class\\_diagram.md](#)
- [docs/v2/diagrams/class\\_diagram.mmd](#)
- [docs/v2/diagrams/class\\_diagram.png](#)

v1 → v2 변경 요약 (주황 강조로 시각화):

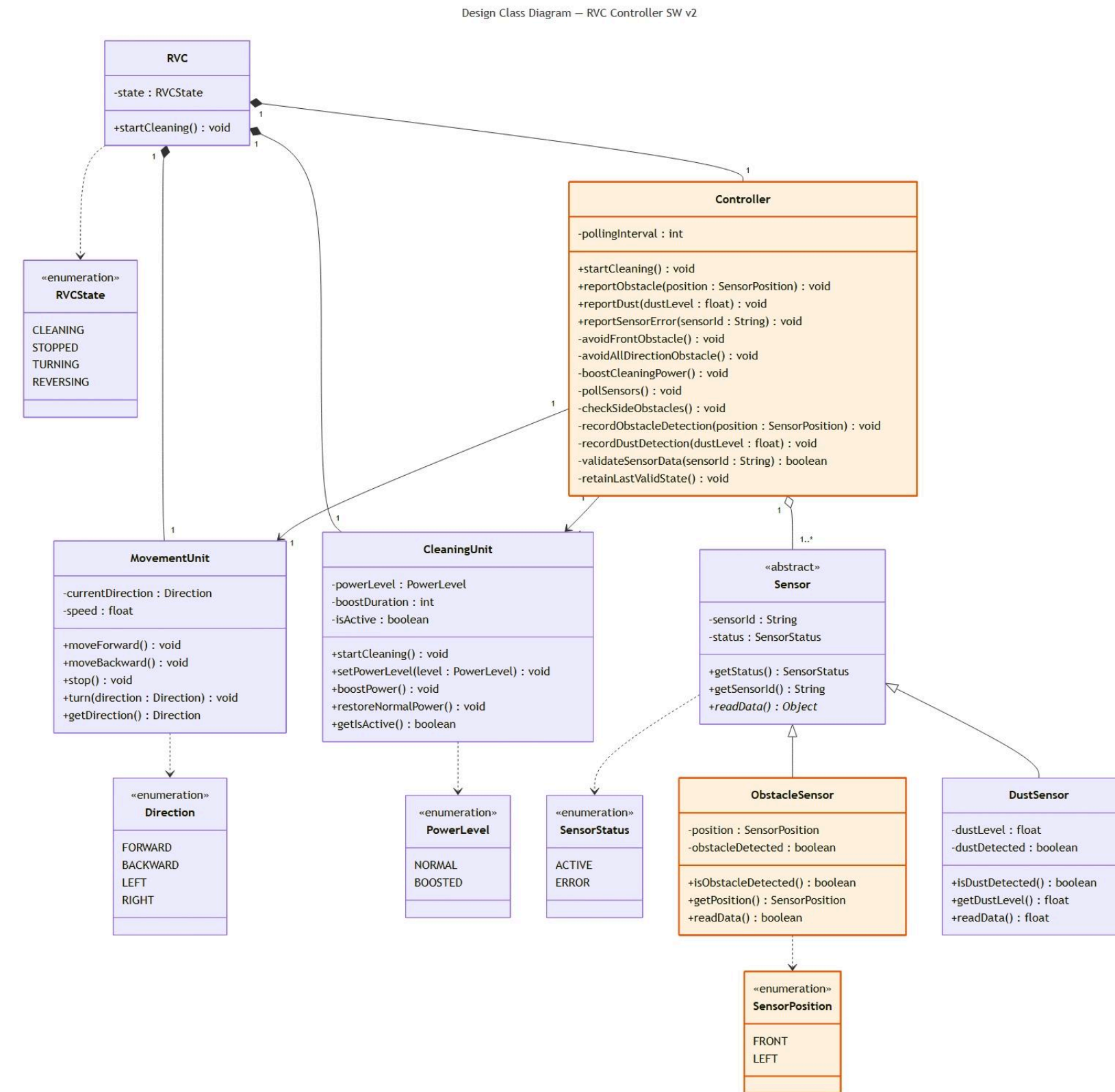
구분	대상	변경 내용
[삭제]	SensorPosition.RIGHT	우측 HW 센서 없음 → enum 값 제거
[변경]	Controller.- checkSideObstacles()	반환 타입 SensorPosition → void. 좌측만 직접 확인, 우측은 avoidFrontObstacle() 내부에서 turn(RIGHT) + front:ObstacleSensor 재활용
[변경]	Controller.- avoidAllDirectionObstacle()	서명 유지, 동작 변경. turn(BACK) 제거 → 후진 후 방향 재판단
[변경]	ObstacleSensor 인스턴스	3개 → 2개(FRONT-LEFT). 클래스 구조는 동일, front 인스턴스가 우측 감지에도 재활용됨
[변경]	Controller↔Sensor 실제 인스턴스 수	4개(3+1) → 3개(2+1)

Design Class Diagram – RVC Controller SW v2

# Class Diagram



# Class Diagram



[삭제]	<code>`SensorPosition.RIGHT`</code>	우측 HW 센서 제거에 따라 열거값 삭제
[변경]	<code>`Controller.-checkSideObstacles()`</code>	반환 타입 <code>`SensorPosition`</code> → <code>`void`</code> . 좌측 센서만 직접 확인; 우측은 <code>`avoidFrontObstacle()`</code> 내부에서 <code>`turn(RIGHT)`</code> + <code>`front:ObstacleSensor`</code> 재활용으로 간접 확인
[변경]	<code>`Controller.-avoidAllDirectionObstacle()`</code>	메서드 서명 유지, 동작 변경. <code>`turn(BACK)`</code> 제거 → 후진 후 <code>`checkSideObstacles()`</code> 와 동일한 방향 재판단 로직 적용
[변경]	ObstacleSensor 인스턴스 수	3개(FRONT·LEFT·RIGHT) → 2개(FRONT·LEFT). 클래스 자체는 유지

# Vibe Coding

OOI

# Code

The screenshot displays a GitHub interface for a repository named 'OOAD-Team9 / OOAD\_AI'. At the top, navigation links include Code, Issues, Pull requests, Agents, Actions, Projects, Security and quality, Insights, and Settings. A search bar is present with the text 'Type to search'. Below the navigation, a commit summary for 'Commit c8289c9' is shown, indicating it was authored by 'DONGL3XD' 5 hours ago and is verified. The commit message is 'Merge pull request #15 from OOAD-Team9/develop', and it shows '12 files changed' with a net change of '+361 -119' lines.

The main content area shows a file diff for 'docs/v2/AI-edit-log.md'. The diff highlights changes in green, showing the addition of a new log entry. The code content is as follows:

```

@@ -0,0 +1,89 @@
1 + # AI Edit Log - 장애물 회피 로직 개선
2 +
3 + 작성일: 2026-06-04
4 + 대상 브랜치: `feat/no-right-sensor`
5 +
6 + ## 1. 작업 목표
7 +
8 + 다음 요구사항에 맞춰 RVC의 장애물 회피 로직과 단위/시스템 테스트를 수정한다.
9 +
10 + 1. **전방에 장애물이 있는 경우**
11 +   - 1.1 좌측을 확인 후 없으면 좌회전 후 진행
12 +   - 1.2 우측을 확인 후 없으면 우회전 후 진행
13 +   - 1.3 좌/우측이 모두 막혀있으면 후진
14 + 2. **후진의 경우**
15 +   - 2.1 3면이 모두 막혀있다면 후진
16 +   - 2.2 한 번 후진 후에는 바로 탈출이 아닌 좌/우측을 탐지 후 비어있는 방향으로 탈출(비어있는 방향으로 회전 후 전진)
17 +   - 2.3 좌/우측이 모두 막혀있는 경우에는 후진을 반복
18 +   - 2.4 10번 후진 후에도 좌/우측에 장애물이 있다면 최종 탈출 불가로 판정해 RVC의 전원 OFF
19 +
20 + ## 2. 기존 구조 파악 요약

```

The left sidebar shows a file explorer with the following structure:

- docs/v2
  - AI-edit-log.md
- diagrams
  - domain\_model.mmd
  - domain\_model.png
- rvc\_project
- src
  - rvc
    - CleaningUnit.hpp
    - Controller.cpp
    - Controller.hpp
    - Types.hpp
  - simulator
    - RVCSimulator.hpp
    - SimCleaningUnit.hpp
  - tests
  - system

# Vibe Coding

Test & Static Analysis: Unit Test

# Unit Test

## : Obstacle Sensor Test

```
// -----  
// ObstacleSensor interface contract  
// -----  
  
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());  
}
```

# Unit Test

## : Dust Sensor Test

```
// -----  
// DustSensor interface contract  
// -----  
  
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();  
}
```

# Unit Test

## : Movement & Cleaning Test

```
// -----  
// MovementUnit interface contract  
// -----  
  
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());  
}  
  
// -----  
// CleaningUnit interface contract  
// -----  
  
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();  
}
```

# Unit Test

(Controller Test)

## : UC-01. Perform Normal Cleaning

```
// =====  
// UC-01 : Perform Normal Cleaning  
// =====  
  
TEST_F(ControllerTest, UC01_TC001_StartCleaning_ActivatesCleaningUnit) {  
    EXPECT_CALL(cu_, startCleaning()).Times(1);  
    controller_>startCleaning();  
}  
  
TEST_F(ControllerTest, UC01_TC002_StartCleaning_CommandsForwardMovement) {  
    EXPECT_CALL(mu_, moveForward()).Times(AtLeast(1));  
    controller_>startCleaning();  
}  
  
TEST_F(ControllerTest, UC01_TC003_PollSensors_NoObstacle_ContinuesForward) {  
    EXPECT_CALL(mu_, moveForward()).Times(AtLeast(1));  
    EXPECT_CALL(mu_, stop()).Times(0);  
    controller_>pollSensors();  
}  
  
TEST_F(ControllerTest, UC01_TC004_PollSensors_ReadsAllSensorsEachCycle) {  
    EXPECT_CALL(frontSensor_, readData()).Times(1);  
    EXPECT_CALL(dustSensor_, readData()).Times(1);  
    controller_>pollSensors();  
}
```

# Unit Test

(Controller Test)

## : UC-02. Detect Obstacle (Front)

```
// =====  
// UC-02 : Detect Obstacle (Front)  
// =====  
  
TEST_F(ControllerTest, UC02_TC001_ReportFrontObstacle_TriggersStop) {  
    EXPECT_CALL(mu_, stop()).Times(AtLeast(1));  
    controller_>reportObstacle(SensorPosition::FRONT);  
}  
  
TEST_F(ControllerTest, UC02_TC002_ReportObstacle_DoesNotThrow) {  
    EXPECT_NO_THROW(controller_>reportObstacle(SensorPosition::FRONT));  
    EXPECT_NO_THROW(controller_>reportObstacle(SensorPosition::LEFT));  
}  
  
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());  
    controller_>pollSensors();  
}
```

# Unit Test (Controller Test)

## : UC-03. Avoid Front Obstacle

```
// =====
// UC-03 : Avoid Front Obstacle
// =====

TEST_F(ControllerTest, UC03_TC001_FrontObstacle_LeftFree_TurnsLeftResumesForward) {
    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::LEFT));
    EXPECT_CALL(mu_, moveForward());
    controller_>pollSensors();
}

// [변경] 좌측 막힘 → turn(RIGHT) + 전방 센서로 우측 확인 → 우측 여유 → 전진
TEST_F(ControllerTest, UC03_TC002_LeftBlocked_TurnsRight_FrontSensorClearsRight) {
    EXPECT_CALL(frontSensor_, isObstacleDetected())
        .WillOnce(Return(true)) // poll
        .WillOnce(Return(false)); // tryEscapeToFreeSide: turn(RIGHT) → 우측 여유
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(true));

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

// [변경] 양측 모두 막힘 → avoidAllDirectionObstacle 진입 → moveBackward 호출
TEST_F(ControllerTest, UC03_TC003_FrontAndLeftAndRightBlocked_EscalatesAllDirection) {
    EXPECT_CALL(frontSensor_, isObstacleDetected())
        .WillOnce(Return(true)) // poll
        .WillOnce(Return(true)) // tryEscapeToFreeSide turn(RIGHT) → 우측도 막힘
        .WillRepeatedly(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(true));

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

// [변경] checkSideObstacles() 제거 → tryEscapeToFreeSide()가 좌측 센서 직접 조회
TEST_F(ControllerTest, UC03_TC004_FrontObstacle_QueriesLeftSensorFirst) {
    ON_CALL(frontSensor_, isObstacleDetected()).WillByDefault(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(false));

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

# Unit Test

(Controller Test)

## : UC-01. Detect Side Obstacles

```
// =====  
// UC-04 : Detect Side Obstacles  
// =====  
  
TEST_F(ControllerTest, UC04_TC001_ReportLeftObstacle_RecordedWithoutError) {  
    EXPECT_NO_THROW(controller_>reportObstacle(SensorPosition::LEFT));  
}  
  
TEST_F(ControllerTest, UC04_TC002_LeftObstacleRecorded_AffectsSideCheck) {  
    ON_CALL(frontSensor_, isObstacleDetected()).WillByDefault(Return(false));  
    EXPECT_CALL(mu_, moveForward()).Times(AtLeast(1));  
    controller_>reportObstacle(SensorPosition::LEFT);  
    controller_>pollSensors();  
}
```

# Unit Test (Controller Test)

## : UC-05. Avoid All-Direction Obstacle

```
// [추가] 후진 1회 후 좌측 여유 → 좌회전 탈출 (backwardAttempts_ 리셋)
TEST_F(ControllerTest, UC05_TC003_OneBackward_ThenLeftFree_EscapesLeft) {
    // poll: front blocked / avoidFront tryEscape: right also blocked
    EXPECT_CALL(frontSensor_, isObstacleDetected())
        .WillOnce(Return(true)) // poll
        .WillOnce(Return(true)) // avoidFront tryEscape: turn(RIGHT) blocked
        .WillRepeatedly(Return(false));
    // avoidFront tryEscape: left blocked / avoidAllDir tryEscape: left clear
    EXPECT_CALL(leftSensor_, isObstacleDetected())
        .WillOnce(Return(true)) // avoidFront tryEscape
        .WillOnce(Return(false)); // avoidAllDir tryEscape: left clear → escape

    EXPECT_CALL(mu_, moveBackward()).Times(1);
}
```

```
// =====
// UC-05 : Avoid All-Direction Obstacle
// =====

// [변경] 단일 후진 확인 - stop, moveBackward, stop 순서 검증
TEST_F(ControllerTest, UC05_TC001_AllDirectionsBlocked_BackwardCalledOnce) {
    EXPECT_CALL(frontSensor_, isObstacleDetected())
        .WillOnce(Return(true))
        .WillOnce(Return(true))
        .WillRepeatedly(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(true));

    EXPECT_CALL(mu_, stop()).Times(AtLeast(2));
    EXPECT_CALL(mu_, moveBackward()).Times(1);
    controller_>pollSensors();
    EXPECT_FALSE(controller_>isPoweredOff());
}

// [변경] stop() precedes moveBackward()
TEST_F(ControllerTest, UC05_TC002_AllDirectionsBlocked_StopBeforeBackward) {
    EXPECT_CALL(frontSensor_, isObstacleDetected())
        .WillOnce(Return(true))
        .WillOnce(Return(true))
        .WillRepeatedly(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(true));

    bool stopCalled = false;
    bool backwardCalled = false;

    ON_CALL(mu_, stop()).WillByDefault([&stopCalled]{ stopCalled = true; });
    ON_CALL(mu_, moveBackward()).WillByDefault([&backwardCalled, &stopCalled]{
        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();
}
```

```
    // 탈출은 turn(LEFT)
    EXPECT_CALL(mu_, turn(LEFT)).Times(1);
};

// [변경] 후진 1회 후 좌측 여유 → 좌회전 탈출 (backwardAttempts_ 리셋)
TEST_F(ControllerTest, UC05_TC003_OneBackward_ThenLeftFree_EscapesLeft) {
    // poll: front blocked / avoidFront tryEscape: right also blocked
    EXPECT_CALL(frontSensor_, isObstacleDetected())
        .WillOnce(Return(true)) // poll
        .WillOnce(Return(true)) // avoidFront tryEscape: turn(RIGHT) blocked
        .WillRepeatedly(Return(false));
    // avoidFront tryEscape: left blocked / avoidAllDir tryEscape: left clear
    EXPECT_CALL(leftSensor_, isObstacleDetected())
        .WillOnce(Return(true)) // avoidFront tryEscape
        .WillOnce(Return(false)); // avoidAllDir tryEscape: left clear → escape

    EXPECT_CALL(mu_, moveBackward()).Times(1);
};

// [변경] stop() precedes moveBackward()
TEST_F(ControllerTest, UC05_TC002_AllDirectionsBlocked_StopBeforeBackward) {
    EXPECT_CALL(frontSensor_, isObstacleDetected())
        .WillOnce(Return(true))
        .WillOnce(Return(true))
        .WillRepeatedly(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(true));

    bool stopCalled = false;
    bool backwardCalled = false;

    ON_CALL(mu_, stop()).WillByDefault([&stopCalled]{ stopCalled = true; });
    ON_CALL(mu_, moveBackward()).WillByDefault([&backwardCalled, &stopCalled]{
        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();
}
```

```
g) 호출 (요구사항 2.4)
stopCleaningUnit) {
    }).WillByDefault(Return(true));
    }).WillByDefault(Return(true));
};

// [변경] 후진 1회 후 좌측 여유 → 좌회전 탈출 (backwardAttempts_ 리셋)
TEST_F(ControllerTest, UC05_TC003_OneBackward_ThenLeftFree_EscapesLeft) {
    // poll: front blocked / avoidFront tryEscape: right also blocked
    EXPECT_CALL(frontSensor_, isObstacleDetected())
        .WillOnce(Return(true)) // poll
        .WillOnce(Return(true)) // avoidFront tryEscape: turn(RIGHT) blocked
        .WillRepeatedly(Return(false));
    // avoidFront tryEscape: left blocked / avoidAllDir tryEscape: left clear
    EXPECT_CALL(leftSensor_, isObstacleDetected())
        .WillOnce(Return(true)) // avoidFront tryEscape
        .WillOnce(Return(false)); // avoidAllDir tryEscape: left clear → escape

    EXPECT_CALL(mu_, moveBackward()).Times(1);
};

// [변경] stop() precedes moveBackward()
TEST_F(ControllerTest, UC05_TC002_AllDirectionsBlocked_StopBeforeBackward) {
    EXPECT_CALL(frontSensor_, isObstacleDetected())
        .WillOnce(Return(true))
        .WillOnce(Return(true))
        .WillRepeatedly(Return(true));
    ON_CALL(leftSensor_, isObstacleDetected()).WillByDefault(Return(true));

    bool stopCalled = false;
    bool backwardCalled = false;

    ON_CALL(mu_, stop()).WillByDefault([&stopCalled]{ stopCalled = true; });
    ON_CALL(mu_, moveBackward()).WillByDefault([&backwardCalled, &stopCalled]{
        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();
}
```

# Unit Test

(Controller Test)  
: UC-06. Detect Dust

```
// =====  
// UC-06 : Detect Dust  
// =====  
  
TEST_F(ControllerTest, UC06_TC001_ReportDust_RecordsWithoutError) {  
    EXPECT_NO_THROW(controller_>reportDust(0.8));  
    EXPECT_NO_THROW(controller_>reportDust(0.0));  
}  
  
TEST_F(ControllerTest, UC06_TC002_ZeroDustLevel_NoPowerBoost) {  
    EXPECT_CALL(cu_, boostPower()).Times(0);  
    controller_>reportDust(0.0);  
}  
  
TEST_F(ControllerTest, UC06_TC003_PollSensors_ReadsDustSensorEachCycle) {  
    EXPECT_CALL(dustSensor_, readData()).Times(AtLeast(1));  
    controller_>pollSensors();  
}
```

# Unit Test (Controller Test)

## : UC-07. Boost Cleaning Power

```
// =====
// UC-07 : Boost Cleaning Power
// =====

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();
}

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();
}

// [변경] 부스트는 즉시 복귀가 아니라 kBoostDuration(5초) 경과 후 복귀한다.
TEST_F(ControllerTest, UC07_TC003_AfterBoostDuration_PowerIsRestored) {
    auto fakeNow = std::make_shared<Controller::Clock::time_point>();
    controller_>setClock([fakeNow] { return *fakeNow; });

    InSequence seq;
    EXPECT_CALL(cu_, boostPower()); // 1st poll: 먼지 감지 → 부스트 시작
    EXPECT_CALL(cu_, restoreNormalPower()); // 5초 경과 후 poll: 기본 파워 복귀

    // 1st poll: 먼지 높음 → 부스트 시작
    ON_CALL(dustSensor_, getDustLevel()).WillByDefault(Return(Controller::kDustThreshold + 0.1));
    controller_>pollSensors();

    // 5초 경과 + 먼지 해소 → 다음 poll에서 자동 복귀(재부스트 없음)
    *fakeNow += Controller::kBoostDuration;
    ON_CALL(dustSensor_, getDustLevel()).WillByDefault(Return(Controller::kDustThreshold - 0.1));
    controller_>pollSensors();
}

// [추가] 5초가 지나기 전에는 부스트를 유지하며 재부스트하지 않는다.
TEST_F(ControllerTest, UC07_TC005_WithinBoostDuration_StaysBoostedNoReboost) {
    auto fakeNow = std::make_shared<Controller::Clock::time_point>();
    controller_>setClock([fakeNow] { return *fakeNow; });
    ON_CALL(dustSensor_, getDustLevel()).WillByDefault(Return(Controller::kDustThreshold + 0.1));

    // 부스트는 1회만 시작, 5초 이내에는 복귀하지 않음
    EXPECT_CALL(cu_, boostPower()).Times(1);
    EXPECT_CALL(cu_, restoreNormalPower()).Times(0);

    controller_>pollSensors(); // 부스트 시작
    *fakeNow += Controller::kBoostDuration / 2; // 2.5초 경과
    controller_>pollSensors(); // 아직 유지 (재부스트 X)
}

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));

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

# Unit Test

(Controller Test)

## : UC-08. Handle Sensor Error

```
// =====  
// UC-08 : Handle Sensor Error  
// =====  
  
TEST_F(ControllerTest, UC08_TC001_TransientSensorError_MovementNotStopped) {  
    EXPECT_CALL(mu_, stop()).Times(0);  
    controller_>reportSensorError(1);  
}  
  
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);  
}  
  
TEST_F(ControllerTest, UC08_TC003_ErrorCountIsPerSensor_IndependentCounters) {  
    EXPECT_CALL(mu_, stop()).Times(0);  
    controller_>reportSensorError(1);  
    controller_>reportSensorError(2);  
    controller_>reportSensorError(1);  
    controller_>reportSensorError(2);  
}  
  
TEST_F(ControllerTest, UC08_TC004_ReportSensorError_DoesNotThrow) {  
    EXPECT_NO_THROW(controller_>reportSensorError(0));  
    EXPECT_NO_THROW(controller_>reportSensorError(99));  
}
```

# Vibe Coding

Test & Static Analysis: System Test

# System Test

## Scenario 1

: 정상 청소 세션

```
// =====  
// Scenario 1 - 정상 청소 세션 (변경 없음)  
// =====  
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());  
        EXPECT_FALSE(sim.cleaning().wasBoosted());  
        EXPECT_EQ(0, sim.movement().getStopCount());  
    }  
}
```

# System Test

## Scenario 2

### : 장애물 연속 회피

```
// =====
// Scenario 2 - 장애물 연속 회피 (UC-01, UC-02, UC-03)
//
// [변경] v1: setRightObstacle로 우측 막힘 설정
//       v2: setRightSideObstacle로 우측 방향 상태 설정
// Phase 2: front blocked, left free → turn LEFT (동일)
// Phase 4: front blocked, left blocked, rightSide free → turn RIGHT
// =====
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: front blocked, left free → turn LEFT —————
    sim.setFrontObstacle(true);
    sim.setRightSideObstacle(true); // [변경] setRightObstacle → setRightSideObstacle
    sim.cleaning().resetBoostFlag();
    sim.tick();

    EXPECT_EQ(Direction::LEFT, sim.movement().getLastTurn())
        << "First obstacle: left free → should turn left";
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());
    EXPECT_GT(sim.movement().getStopCount(), stopsAfterClear1);

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

    // — Phase 4: front blocked, left blocked, rightSide free → turn RIGHT —
    // [변경] setRightSideObstacle(false): turn(RIGHT) 후 frontSensor가 clear 반환
    sim.setFrontObstacle(true);
    sim.setLeftObstacle(true);
    sim.setRightSideObstacle(false);
    sim.cleaning().resetBoostFlag();
    sim.tick();

    EXPECT_EQ(Direction::RIGHT, sim.movement().getLastTurn())
        << "Second obstacle: left blocked, right side clear → should turn right";
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());
    EXPECT_GT(sim.movement().getStopCount(), stopsAfterAvoid1);

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

# System Test

## Scenario 3

: 사방 막힘 후진 → 측면 개방 시 탈출

```
// =====
// Scenario 3 - 사방 막힘 후진 → 측면 개방 시 탈출 (UC-01, UC-02, UC-04, UC-05)
//
// [변경] v2 로직(요구사항 2.3): 전방·좌·우가 모두 막히면 즉시 탈출하지 않고
// 후진을 반복한다. 한쪽이 열린 뒤에야 그 방향으로 회전 후 전진 탈출한다.
// =====
TEST(SystemScenario, Scenario3_CornerTrapAndEscape) {
    RVCSimulator sim;
    sim.start();

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

    // — Phase 2: 전방+좌+우측 모두 막힘 → 후진(즉시 탈출 X) —————
    sim.setFrontObstacle(true);
    sim.setLeftObstacle(true);
    sim.setRightSideObstacle(true);
    sim.cleaning().resetBoostFlag();
    sim.tick();

    EXPECT_NE(Motion::FORWARD, sim.movement().getMotion())
        << "All sides blocked: must reverse and retry, not resume forward";
    EXPECT_FALSE(sim.isPoweredOff()) << "Single reversal must not power off";
    EXPECT_GE(sim.movement().getStopCount(), 2)
        << "Two stop() calls: avoidFrontObstacle + avoidAllDirectionObstacle";

    // — Phase 3: 좌측이 열림 → 좌회전 후 전진 탈출 —————
    sim.setLeftObstacle(false);
    sim.cleaning().resetBoostFlag();
    sim.tick();

    EXPECT_EQ(Direction::LEFT, sim.movement().getLastTurn())
        << "Left side opened → should turn left and escape";
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());

    runClearTicks(sim, 3);
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());
    EXPECT_TRUE(sim.cleaning().getIsActive());
}
```

# System Test

## Scenario 4

### : 먼지 구역 통과

```
// =====  
// Scenario 4 - 먼지 구역 통과 (UC-01, UC-06, UC-07)  
//  
// [변경] 부스트는 5초간 유지 후 복귀(FR-CLN-02/03). 가상 시간을 주입하여  
// 시작 → 유지(재부스트 X) → 5초 만료 후 복귀 라이프사이클을 검증한다.  
// =====  
TEST(SystemScenario, Scenario4_DustyZoneTraversal) {  
    RVCSimulator sim;  
    auto fakeNow = std::make_shared<Controller::Clock::time_point>();  
    sim.setClock([fakeNow] { return *fakeNow; });  
    sim.start();  
  
    // --- Phase 1: 청정 구역 - 부스트 없음 ---  
    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";  
        EXPECT_EQ(PowerLevel::NORMAL, sim.cleaning().getPowerLevel());  
    }  
  
    // --- Phase 2: 먼지 구역 진입 - 첫 틱에 부스트 시작 ---  
    sim.setDustLevel(Controller::kDustThreshold + 0.3);  
    sim.cleaning().resetBoostFlag();  
    sim.tick();  
    EXPECT_TRUE(sim.cleaning().wasBoosted()) << "Dust detected → boost starts";  
    EXPECT_EQ(PowerLevel::BOOSTED, sim.cleaning().getPowerLevel());  
  
    // --- Phase 3: 5초 이내에는 BOOSTED 유지, 재부스트 없음 ---  
    for (int t = 0; t < 3; ++t) {  
        *fakeNow += std::chrono::seconds(1); // 누적 3초 (< 5초)  
        sim.cleaning().resetBoostFlag();  
        sim.tick();  
        EXPECT_FALSE(sim.cleaning().wasBoosted()) << "Within 5s: no re-boost";  
        EXPECT_EQ(PowerLevel::BOOSTED, sim.cleaning().getPowerLevel())  
            << "Within 5s: power stays boosted";  
        EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
    }  
  
    // --- Phase 4: 5초 경과 + 먼지 잔존 → 복귀 후 동일 틱에 재부스트 ---  
    *fakeNow += std::chrono::seconds(3); // 누적 6초 (≥ 5초)  
    sim.cleaning().resetBoostFlag();  
    sim.tick();  
    EXPECT_TRUE(sim.cleaning().wasBoosted())  
        << "Boost expired but dust remains → re-boost";  
    EXPECT_EQ(PowerLevel::BOOSTED, sim.cleaning().getPowerLevel());  
  
    // --- Phase 5: 먼지 해소 + 5초 경과 → 기본 파워 복귀 후 유지 ---  
    sim.setDustLevel(0.0);  
    *fakeNow += Controller::kBoostDuration;  
    for (int t = 0; t < 2; ++t) {  
        sim.cleaning().resetBoostFlag();  
        sim.tick();  
        EXPECT_FALSE(sim.cleaning().wasBoosted());  
        EXPECT_EQ(PowerLevel::NORMAL, sim.cleaning().getPowerLevel());  
    }  
}
```

# System Test

## Scenario 5

: 장애물+먼지 복합

```
// =====  
// Scenario 5 - 장애물+먼지 복합 (UC-01~03, UC-06~07)  
// [변경] setRightObstacle 제거 (좌측 free이므로 right side 설정 불필요)  
// =====  
TEST(SystemScenario, Scenario5_ObstacleThenDustyZone) {  
    RVCSimulator sim;  
    sim.start();  
  
    runClearTicks(sim, 2);  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
  
    // front blocked + dust: 장애물 우선 처리 → 부스트 없음  
    sim.setFrontObstacle(true);  
    sim.setDustLevel(Controller::kDustThreshold + 0.2);  
    sim.cleaning().resetBoostFlag();  
    sim.tick();  
  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
    EXPECT_FALSE(sim.cleaning().wasBoosted())  
        << "Obstacle avoidance takes priority; dust boost skipped";  
  
    // 장애물 해제 + 먼지 유지 → 부스트 발동  
    sim.setFrontObstacle(false);  
    sim.cleaning().resetBoostFlag();  
    sim.tick();  
  
    EXPECT_TRUE(sim.cleaning().wasBoosted())  
        << "Obstacle cleared; dust boost should fire";  
    // [변경] 부스트 직후에는 BOOSTED 유지 (5초 경과 전 복귀하지 않음)  
    EXPECT_EQ(PowerLevel::BOOSTED, sim.cleaning().getPowerLevel());  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
  
    runClearTicks(sim, 2);  
    EXPECT_FALSE(sim.cleaning().wasBoosted());  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
}
```

# System Test

## Scenario 6

: 센서 오류 누적 → 비상 정지

```
// =====  
// Scenario 6 - 센서 오류 누적 → 비상 정지 (변경 없음)  
// =====  
TEST(SystemScenario, Scenario6_SensorDegradationAndEmergencyStop) {  
    RVCSimulator sim;  
    sim.start();  
  
    runClearTicks(sim, 3);  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
    EXPECT_EQ(0, sim.movement().getStopCount());  
  
    sim.reportSensorError(1);  
    EXPECT_NE(Motion::STOPPED, sim.movement().getMotion());  
  
    sim.reportSensorError(1);  
    EXPECT_NE(Motion::STOPPED, sim.movement().getMotion());  
  
    sim.cleaning().resetBoostFlag();  
    sim.tick();  
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());  
  
    int stopsBeforeEmergency = sim.movement().getStopCount();  
    sim.reportSensorError(1);  
    EXPECT_EQ(Motion::STOPPED, sim.movement().getMotion());  
    EXPECT_EQ(stopsBeforeEmergency + 1, sim.movement().getStopCount());  
}
```

# System Test

## Scenario 7

### : 전체 청소 세션

```
// =====
// Scenario 7 - 전체 청소 세션 (모든 UC 연속)
//
// [변경] Phase 2: setRightObstacle → setRightSideObstacle
//         Phase 6: setRightObstacle → setRightSideObstacle, getLastTurn BACK → RIGHT
// =====
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: front blocked, left free → turn LEFT —————
    // [변경] setRightSideObstacle 사용
    sim.setFrontObstacle(true);
    sim.setRightSideObstacle(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 —————
    // [변경] 부스트는 첫 틱에 1회 시작되어 5초간 BOOSTED 유지(재부스트 X)
    sim.setDustLevel(Controller::kDustThreshold + 0.25);
    sim.cleaning().resetBoostFlag();
    sim.tick();
    EXPECT_TRUE(sim.cleaning().wasBoosted()) << "Dust detected → boost starts";
    EXPECT_EQ(PowerLevel::BOOSTED, sim.cleaning().getPowerLevel());
}
```

```
for (int t = 0; t < 2; ++t) {
    sim.cleaning().resetBoostFlag();
    sim.tick();
    EXPECT_FALSE(sim.cleaning().wasBoosted()) << "Within 5s: no re-boost";
    EXPECT_EQ(PowerLevel::BOOSTED, sim.cleaning().getPowerLevel());
}

// — Phase 5: 먼지 해소 —————
runClearTicks(sim, 2);
EXPECT_FALSE(sim.cleaning().wasBoosted());

// — Phase 6: 사방 막힘 → 후진(즉시 탈출 X) —————
// [변경] v2 요구사항 2.3: 모두 막히면 후진 반복, 즉시 전진 탈출 안 함
sim.setFrontObstacle(true);
sim.setLeftObstacle(true);
sim.setRightSideObstacle(true);
sim.cleaning().resetBoostFlag();
sim.tick();

EXPECT_NE(Motion::FORWARD, sim.movement().getMotion())
    << "All sides blocked: reverse and retry, not forward";
EXPECT_FALSE(sim.isPoweredOff());
int stopsP6 = sim.movement().getStopCount();
EXPECT_GT(stopsP6, stopsP2);

// — Phase 7: 우측이 열림 → 우회전 후 전진 탈출 —————
sim.setRightSideObstacle(false);
sim.cleaning().resetBoostFlag();
sim.tick();

EXPECT_EQ(Direction::RIGHT, sim.movement().getLastTurn())
    << "Right side opened (left still blocked) → turn RIGHT and escape";
EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());

runClearTicks(sim, 2);
EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());

// — Phase 8: 센서 오류 → 비상 정지 —————
for (int i = 0; i < Controller::kErrorThreshold - 1; ++i) {
    sim.reportSensorError(2);
    EXPECT_NE(Motion::STOPPED, sim.movement().getMotion());
}
sim.reportSensorError(2);
EXPECT_EQ(Motion::STOPPED, sim.movement().getMotion());

EXPECT_TRUE(sim.cleaning().getIsActive());
}
```

# System Test

## Scenario 8

: 완전 포위 (10회 후진 후 탈출 불가 → 전원 OFF)

```
// =====
// Scenario 8 - 완전 포위: 10회 후진 후 탈출 불가 → 전원 OFF (UC-05, 요구사항 2.4)
//
// 전방·좌·우측이 모두 막힌 상태가 지속되면, RVC는 매 틱 후진을 반복하다가
// kMaxBackwardAttempts(10)회 후에도 탈출하지 못하면 최종 탈출 불가로 판정하여
// 전원을 차단한다(이동 정지 + 청소 정지).
// =====
TEST(SystemScenario, Scenario8_FullyTrapped_PowersOffAfterMaxReversals) {
    RVCSimulator sim;
    sim.start();
    ASSERT_TRUE(sim.cleaning().getIsActive());

    // — Phase 1: 정상 전진 —————
    runClearTicks(sim, 2);
    EXPECT_EQ(Motion::FORWARD, sim.movement().getMotion());

    // — Phase 2: 전방·좌·우측 모두 막힘 - 매 틱 후진 반복 —————
    sim.setFrontObstacle(true);
    sim.setLeftObstacle(true);
    sim.setRightSideObstacle(true);

    // 9번째 틱까지는 아직 전원 OFF가 아니어야 한다.
    for (int i = 0; i < Controller::kMaxBackwardAttempts - 1; ++i) {
        sim.cleaning().resetBoostFlag();
        sim.tick();
        EXPECT_FALSE(sim.isPoweredOff())
            << "Reversal " << (i + 1) << ": should still be retrying, not powered off";
        EXPECT_TRUE(sim.cleaning().getIsActive());
    }

    // — Phase 3: 10번째 후진 후에도 탈출 불가 → 전원 OFF —————
    sim.tick();
    EXPECT_TRUE(sim.isPoweredOff())
        << "After 10 reversals still trapped → RVC must power off";
    EXPECT_EQ(Motion::STOPPED, sim.movement().getMotion());
    EXPECT_FALSE(sim.cleaning().getIsActive())
        << "Power off must stop the cleaning unit";

    // — Phase 4: 전원 OFF 이후에는 더 이상 동작하지 않는다 —————
    int stopsAfterOff = sim.movement().getStopCount();
    sim.tick();
    EXPECT_EQ(stopsAfterOff, sim.movement().getStopCount())
        << "No further actuation after power off";
    EXPECT_TRUE(sim.isPoweredOff());
}
```

# Vibe Coding

Test & Static Analysis: Static Analysis

# Static Analysis

## 1. Clang-tidy

The screenshot shows the details of a Clang-tidy job. On the left, a sidebar contains a 'Summary' section with 'All jobs' and a list of jobs: 'clang-tidy' (selected) and 'cppcheck'. Below this is a 'Run details' section with links for 'Usage' and 'Workflow file'. The main content area shows 'Annotations' with '1 warning'. Below that, the job name 'clang-tidy' is displayed with the status 'succeeded 3 days ago in 37s'. A list of steps follows, all marked with checkmarks: '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'.

## 2. cppCheck

The screenshot shows the details of a cppcheck job. On the left, a sidebar contains a 'Summary' section with 'All jobs' and a list of jobs: 'clang-tidy' and 'cppcheck' (selected). Below this is a 'Run details' section with links for 'Usage' and 'Workflow file'. The main content area shows 'Annotations' with '1 warning'. Below that, the job name 'cppcheck' is displayed with the status 'succeeded 3 days ago in 19s'. A list of steps follows, all marked with checkmarks: 'Set up job', 'Checkout', 'Install cppcheck', 'Run cppcheck on src/rvc/', 'Post Checkout', and 'Complete job'.

# Vibe Coding

비교 및 수정

# 장애물 회피 로직 수정

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

OOAD를 이용한 RVC SW 개발을 진행 중. docs/v2 안에 있는 문서들과 코드를 통해 소프트웨어의 구조를 파악하고 아래의 수정사항을 반영하여 코드를 수정할 것.

장애물 회피 로직을 수정

1. 전방에 장애물이 있는 경우

1.1 좌측을 확인 후 없으면 좌회전 후 진행

1.2 우측을 확인 후 없으면 우회전 후 진행

1.3 좌/우측이 모두 막혀있으면 후진

2. 후진의 경우

2.1 3면이 모두 박혀있다면 후진

2.2 한번 후진 후에는 바로 탈출이 아닌 좌/우측을 탐지 후 비어있는 방향으로 탈출(비어있는 방향으로 회전 후 전진)

2.3 좌/우측이 모두 막혀있는 경우에는 후진을 반복

2.4. 10번 후진 후에도 좌/우측에 장애물이 있다면 최종 탈출 불가로 판정해 RVC의 전원 off

위 로직을 반영하여 RVC의 코드와 Unit test/System test를 수정. 이후 어떤 부분을 어떻게 수정하였는지를 .md파일로 만들어 docs/v2/AI-edit-log.md로 저장

Show less

```
cleaningUnit_2stopcleaning());  
state_ = RVCState::POWER_OFF;  
poweredOff_ = true;  
backwardAttempts_ = 0;
```

# 장애물 회피 로직 수정

```

14 + static constexpr int kErrorThreshold = 3;
15 + static constexpr double kDustThreshold = 0.5;
16 + static constexpr int kMaxBackwardAttempts = 10; // [추가] 후진 재시도 최대 횟수

```

```

30 + // [변경] 전원 OFF 상태 조회 (10회 후진 후에도 좌/우 모두 막혀 탈출 불가 시 true)
31 + [[nodiscard]] bool isPoweredOff() const { return poweredOff_; }
32 +

```

```

34 - void checkSideObstacles();
37 + // [추가] 좌→우 순서로 탈출 방향을 탐색하고 탈출 성공 시 true 반환
38 + bool tryEscapeToFreeSide();
39 + // [추가] 최종 탈출 불가 판정 - 이동/청소 정지 후 전원 OFF 상태로 전환
40 + void powerOff();
41 + // [삭제] checkSideObstacles() - tryEscapeToFreeSide()로 통합

```

```

58 + bool poweredOff_{false}; // [변경] 전원 OFF(최종 탈출 불가) 상태 플래그
59 + int backwardAttempts_{0}; // [추가] 연속 후진 시도 횟수
60 +

```

- 좌/우측 장애물 감지를 tryToEscapeFreeSide() 함수로 일원화하여 이를 전/측방 장애물 감지와 후진의 경우에 사용할 수 있도록 구현
- 기존에 존재하지 않던 10회 후진 기능 및 최초 후진 이후 바로 좌측 탈출이 아닌 좌/우측 탐지 후 탈출 기능을 구현

# Cleaner 'PowerUp' 수정

boostCleaningPower는 5초간 power up후 원상복귀로 변경  
AI-cleaner-log.md로 정리

## 변경 요약

boostCleaningPower 를 "즉시 올렸다 복구" → "5초간 부스트 후 자동 복구" 로 변경했습니다.

## 핵심 설계

"5초"는 실제 경과 시간이므로 `std::chrono::steady_clock` 기반으로 구현하되, 시뮬레이터가 틱(폴링) 구동이라 매 폴링마다 만료를 확인하는 방식으로 처리했습니다. 테스트에서 실제로 5초를 기다리지 않도록 시간 소스(clock)를 주입 가능하게 만들었습니다.

- [Controller.hpp](#): `kBoostDuration{5000}ms`, `clock`, `setClock()`, `updateCleaningBoost()`, 부스트 상태 멤버 추가
- [Controller.cpp](#): `boostCleaningPower()` 는 부스트 시작만(중복 방지), `updateCleaningBoost()` 가 5초 경과 시 복구. `pollSensors()` 최상단에서 타이머 갱신 → 장애물 회피로 조기 반환해도 정확히 5초 후 복구
- [RVCSimulator.hpp](#): 시스템 테스트용 `setClock()` 패스스루

## 동작

- 먼지 감지 → 부스트 시작(BOOSTED) → **5초간 유지**(이 동안 재부스트 없음) → 5초 경과 시 다음 폴링에서 NORMAL 복구. 먼지가 계속 있으면 복구 직후 새 5초 창으로 재부스트.

## 테스트

즉시 복귀를 전제하던 기존 테스트들(단위 `UC07_TC003`, 통합 `Scenario4/6`, 시스템 `Scenario4/5/7`)을 가상 시간 기반 라이프사이클 검증으로 수정하고, "5초 이내 유지·재부스트 없음" 단위 테스트(`UC07_TC005`)를 추가했습니다.

# Cleaner 'PowerUp' 수정

```
143 151 void Controller::boostCleaningPower() {  
152 +   if (boostActive_) return;  
144 153   cleaningUnit_>boostPower();  
145 -   cleaningUnit_>restoreNormalPower();
```

## [기존 코드]

- powerUp 후 곧바로 원 상태로 복귀

```
149 + // [변경] 먼지 감지 시 부스트 시작 - 5초간 유지 (즉시 복귀하지 않음).  
150 + //   이미 부스트 중이면 시작 시각을 유지하여 중복 부스트를 방지한다.  
143 151 void Controller::boostCleaningPower() {  
152 +   if (boostActive_) return;  
144 153   cleaningUnit_>boostPower();  
145 -   cleaningUnit_>restoreNormalPower();  
154 +   boostActive_ = true;  
155 +   boostStartedAt_ = now();  
156 + }  
157 +  
158 + // [추가] 부스트 시작 후 kBoostDuration(5초)이 지나면 기본 청소 파워로 복귀.  
159 + void Controller::updateCleaningBoost() {  
160 +   if (!boostActive_) return;  
161 +   if (now() - boostStartedAt_ >= kBoostDuration) {  
162 +     cleaningUnit_>restoreNormalPower();  
163 +     boostActive_ = false;  
164 +   }
```

## [변경된 코드]

- 5초의 시간을 유지하도록 코드를 수정
- 5초가 유지되는 동안 다른 동작이 불가능한 deadlock이 발생하는지도 검토했으나 일어나지 않도록 잘 구현됨

**감사합니다.**