
객체지향개발방법론 (OOAD)

로봇 청소기(RVC) VIBE CODING

202211291 김호경

202111282 노을영

202111284 문재현

Agenda

- Prompt
- SRS
- SSD
- Code
- Test
- System Code Analysis
- Trouble Shooting

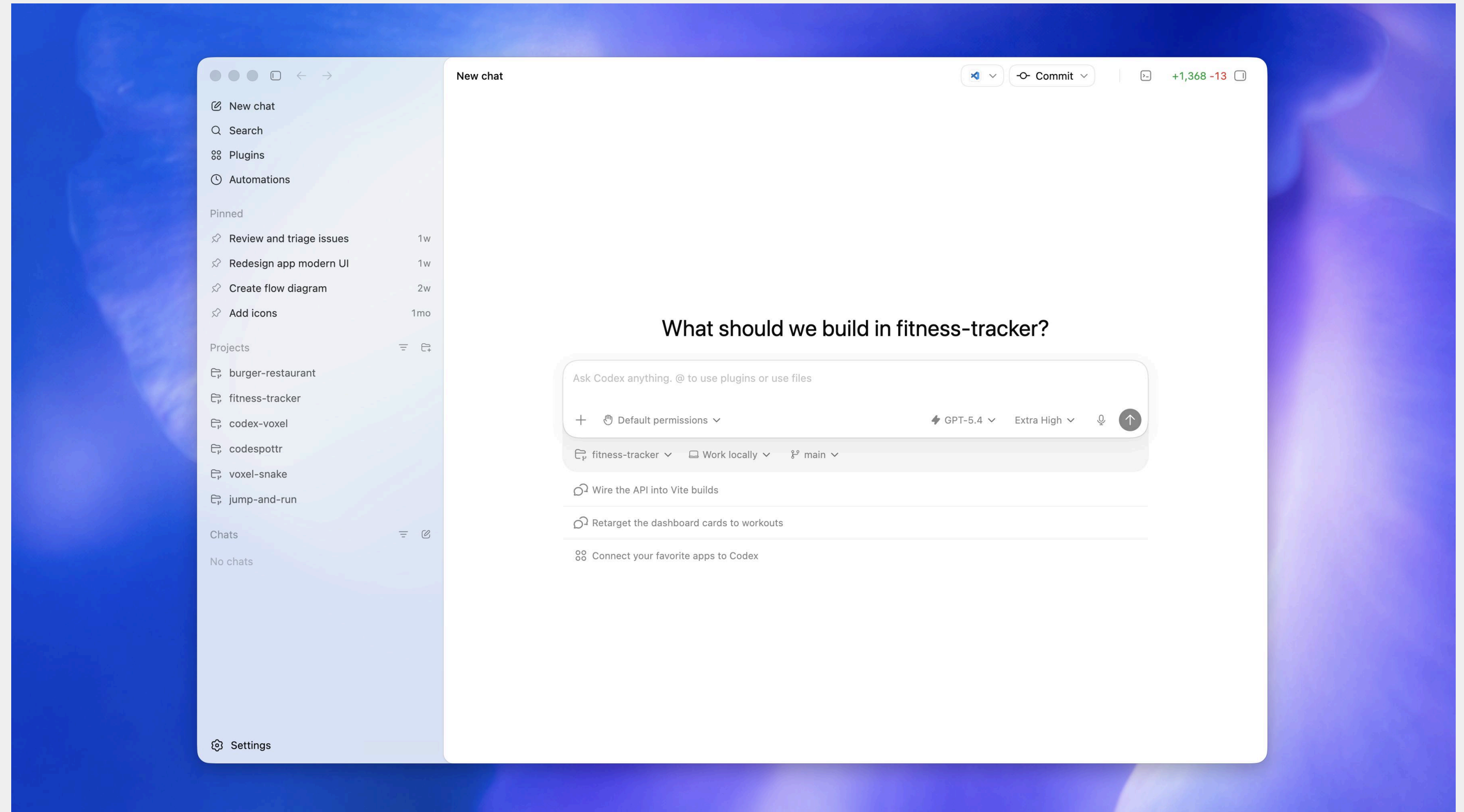
작업 흐름



CODEX

BACKGROUND

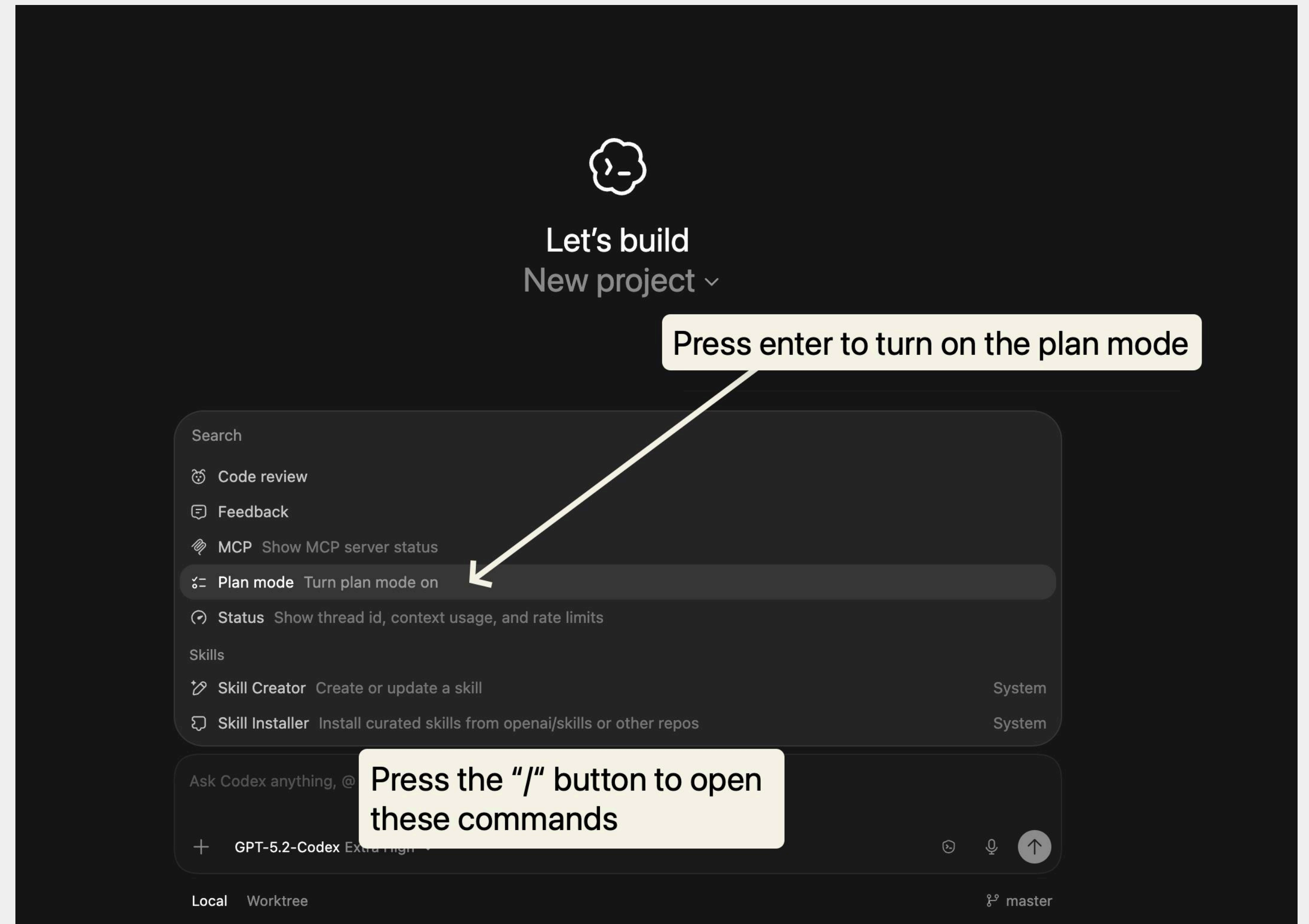
- OpenAI에서 만든 코드 에이전트
- Web, IDE, CLI등 다양한 방식으로 사용 가능
- 다른 도구도 있는데, Codex를 고른 이유
 - 사용량 문제
 - 조원 모두가 사용하기 편했음
- Cursor, Gemini CLI, Claude Code와의 비교
 - Cursor → 너무 비싸고, 코드 편집기와 종속되다보니 무거움
 - Gemini CLI → CLI 기반이어서 결과를 보기 직관적이지 않고, 대화 데이터가 터미널 종료 시 날아감
 - Claude Code → 결제 필요. 우리 모두가 사용하기에는 올바르지 않다고 생각함



PLAN MODE

<https://developers.openai.com/codex/learn/best-practices>

- Codex 공식 문서를 보면, Best Practice로 Plan 모드를 이용하라고 나와 있음
- Plan 모드는 Chain-of-Thought(CoT) Programming 에 기반을 두고 있음
 - 이는 문제 Z로부터, 정답인 Y에 도달하기 전, 중간 과정 Z를 거치는 방식
- 단순히 코드 구현이 아닌, 문서를 만들고 그 문서에 의거해 하나하나 나아가는 방식이므로, Plan 모드로 틀을 잡는 것이 적절하다는 생각을 하였음



PROMPT

docs 폴더안에 있는 pdf를 읽고 ooad 분석하고 소프트웨어랑 시뮬레이터를 c++로 만들어줘

먼저 저장소 구조랑 docs 안의 PDF 후보를 확인해서, 과제가 요구하는 범위와 기존 코드 상태를 잡아볼게요. 지금은 계획 모드라 파일을 바꾸지는 않고 읽고 확인만 하겠습니다.

저장소는 거의 비어 있고 docs/rvc.pdf 하나만 있네요. 이제 PDF 텍스트를 뽑아 요구사항이 무엇인지 확인하겠습니다.

텍스트 추출은 가능하지만 Windows 콘솔 인코딩 때문에 한 번 끊겼습니다. PDF 자체는 65쪽짜리 OOAD 팀 프로젝트 가이드로 보이고, 출력 인코딩만 UTF-8로 맞춰 다시 읽겠습니다.

PDF의 실제 RVC 요구사항은 4~6쪽에 집중되어 있고, 나머지는 산출물 예시와 UP/OOAD 절차 설명입니다. 구현 계획을 현실적으로 만들기 위해 로컬 C++ 빌드 도구도 확인해볼게요.

확인된 요구사항은 "RVC 자동 청소 제어 SW + 테스트용 시뮬레이터"이고, 저장소에는 아직 구현 골격이 없습니다. 이제 구현 형태에서 결정이 필요한 부분만 짧게 잠그겠습니다.

좋습니다. 범위는 전체 산출물, 시뮬레이터는 CLI 격자, 유닛 테스트는 GoogleTest, 시스템 테스트는 시뮬레이터 기반으로 잡겠습니다. GoogleTest를 어떻게 연결할 수 있을지 로컬 힌트만 더 확인해볼게요.

PROMPT

PLAN 모드 응답 →

플랜



RVC 제어 소프트웨어와 CLI 시뮬레이터 구현 계획

요약

- docs/rvc.pdf 의 4~6쪽 요구사항을 기준으로 RVC 자동 청소 제어 소프트웨어와 테스트용 CLI 격자 시뮬레이터를 C++로 구현한다.
- OOAD 산출물은 docs/ood_analysis.md 에 정리하고, 코드에는 제어 로직과 시뮬레이터를 분리해 둔다.
- 유닛 테스트는 구글테스트로 작성하고, 시스템 테스트는 시뮬레이터 시나리오 실행 결과로 검증한다.

주요 구현

- CMake 기반 C++20 프로젝트를 만든다.
- 제어 소프트웨어는 센서, 모터, 청소기, 시계 입력을 추상 인터페이스로 두고 RvcController 가 매 틱마다 동작 명령 결정하게 한다.
- 기본 동작 규칙:
 - 장애물이 없으면 청소하면서 직진한다.
 - 전방 장애물이 있고 좌우 중 한쪽이 열려 있으면 정지 후 회전하고 전진한다.
 - 전방, 좌측, 우측이 모두 막히면 후진 후 회전한다.
 - 먼지를 감지하면 3틱 동안 청소 세기를 높인다.
 - 좌우가 모두 가능한 경우 기본 회전 정책은 우선 방향을 번갈아 선택해 반복 회전을 줄인다.

PROMPT

PLAN 모드 응답 →

OOAD 산출물

- docs/ood_analysis.md 에 다음을 포함한다:
 - 기능 요구사항, 비기능 요구사항, 유스케이스
 - 시스템 시퀀스 다이어그램과 시스템 오퍼레이션
 - 도메인 모델과 주요 클래스 책임
 - 설계 시퀀스 흐름, SOLID 적용 분석
 - 요구사항부터 테스트까지의 추적성 표

테스트 계획

- 구글테스트 유닛 테스트:
 - 장애물 없음: 전진 및 청소 명령
 - 전방 장애물: 회전 후 전진 결정
 - 삼방향 장애물: 후진 후 회전 결정
 - 먼지 감지: 3틱 부스트 유지 후 정상 세기로 복귀
 - 회전 정책: 양쪽 가능 시 결정이 예측 가능함
- 시스템 테스트:
 - 시뮬레이터 시나리오를 실행해 최종 위치, 방향, 청소된 먼지 수, 로그의 핵심 명령을 검증한다.
- CMake는 먼저 설치된 구글테스트를 찾고, 없으면 고정된 구글테스트 버전을 가져오도록 구성한다. 네트워크가 막히면 구현 단계에서 승인 후 내려받는다.

가정

- GUI는 만들지 않고 CLI 격자 시뮬레이터로 구현한다.
- 하드웨어 세부 제어는 PDF 지시에 따라 제외하고, 제어 SIM는 추상 명령을 출력한다.
- 로컬 빌드는 CMake와 Visual Studio 2022 생성기를 기본 기준으로 한다.

PROMPT

PLAN 모드에서 AI가 놓친 부분이 많았기 때문에 PLAN 수정을 위한 상호작용 프롬프트

산출물은 유스케이스+FR+NFR / ooa -SSD,Domain Diagram / ood - SD,CD 로 각각 파일 만들어줘 필요한 경우 더 만들어도됨
후진은 탈출 가능할때까지 후진한다고 명시

전방센서는 interrutp 나머지는 period 센서임

ooa ssd 문서 아래에 시스템 인터페이스도 추가해줘 또 작업은 문서 작업을 우선 끝내고 문서 기반으로 작업해줘

핵심 제어 규칙 도 명시해줘

PROMPT

최종 수정된 PLAN 모드 응답 →

RVC OOAD 문서 우선 작성 및 C++ 구현 계획

요약

- 작업 순서는 문서 산출물 완성 → 문서 기반 C++ 구현 → 테스트/시뮬레이터 검증으로 진행한다.
- docs/rvc.pdf 의 RVC 요구사항에 추가 조건을 반영한다: 전방 센서는 interrupt, 좌/우/먼지 센서는 periodic, 후진은 탈출 가능할 때까지 계속 수행.
- 구현은 완성된 OOAD 문서의 시스템 인터페이스와 핵심 제어 규칙을 기준으로 작성한다.

문서 산출물

- docs/requirements.md
 - Use Cases, Functional Requirements, Non-Functional Requirements
 - 센서 방식과 핵심 제어 규칙 명시
- docs/ooa_ssd.md
 - 자동 청소 시작, 전방 장애물 interrupt, periodic 센서 갱신, 먼지 감지, 후진 탈출 SSD
 - 문서 하단에 System Interface 섹션 추가
- docs/ooa_domain_diagram.md
 - RVC, Controller, Sensors, Motor, Cleaner, Clock, Environment 도메인 다이어그램
- docs/ood_sequence_diagrams.md
 - 제어 루프, interrupt 처리, periodic sampling, 장애물 회피, 후진 탈출 SD
- docs/ood_class_diagram.md
 - Class Diagram, 클래스 책임, SOLID 적용 근거
- 필요 시 docs/traceability.md
 - FR, 유스케이스, SSD, SD, 클래스, 테스트 케이스 연결

PROMPT

최종 수정된 PLAN 모드 응답 →

핵심 제어 규칙

- 청소가 시작되면 RVC는 기본적으로 청소기를 켜 상태로 전진한다.
- 전방 센서는 interrupt 방식이며, 전방 장애물이 감지되면 즉시 전진을 멈추고 회피 판단으로 진입한다.
- 좌측 센서, 우측 센서, 먼지 센서는 periodic 방식이며 제어 tick마다 최신 값을 갱신한다.
- 전방 장애물이 있고 좌/우 중 한쪽만 열려 있으면 열린 방향으로 회전한 뒤 전진 청소를 재개한다.
- 전방 장애물이 있고 좌/우가 모두 열려 있으면 기본 회전 정책에 따라 방향을 선택한다. 기본값은 좌우 번갈아 선택이다.
- 전방, 좌측, 우측이 모두 막혀 있으면 Escaping 상태로 진입한다.
- Escaping 상태에서는 탈출 가능할 때까지 계속 후진한다.
- 탈출 가능 조건은 전방, 좌측, 우측 중 하나 이상이 열린 상태로 정의한다.
- 탈출 가능해지면 열린 방향으로 회전하고 전진 청소를 재개한다.
- 먼지가 감지되면 일정 tick 동안 청소 세기를 높이고, 시간이 끝나면 기본 세기로 복귀한다.

구현 방향

- C++20/CMake 프로젝트로 구성한다.
- 제어 SW는 문서의 System Interface를 기준으로 RvcController API를 만든다.
- System Interface 예:
 - startCleaning()
 - stopCleaning()
 - onFrontObstacleInterrupt()
 - tick()
 - readPeriodicSensors()
 - decideNextCommand()
- CLI 시뮬레이터는 격자 맵에서 로봇, 장애물, 먼지를 표시하고 tick별 센서/event, 명령, 위치, 방향, 청소 세기를 출력한다.

PROMPT

최종 수정된 PLAN 모드 응답 →

테스트 계획

- GoogleTest 유닛 테스트:
 - 정상 직진 청소
 - 전방 interrupt 발생 시 즉시 정지와 회피
 - 좌/우 periodic 센서 값 기반 회전
 - 삼방향 장애물에서 Escaping 진입
 - 탈출 가능할 때까지 후진 유지
 - 탈출 가능 후 회전 및 청소 재개
 - 먼지 감지 시 청소 부스트 유지와 복귀
- 시스템 테스트:
 - 시뮬레이터 시나리오로 전방 interrupt, periodic 센서, 먼지 청소, 후진 탈출 흐름 검증
 - 최종 위치, 방향, 청소된 먼지 수, 핵심 로그 확인

가정

- 다이어그램은 Markdown 내부 Mermaid로 작성한다.
- GUI는 만들지 않고 CLI 시뮬레이터만 구현한다.
- 후방 센서는 제어 SW 판단에는 포함하지 않는다. 시뮬레이터는 격자 충돌 방지를 위해 후방 이동 가능 여부만 내부적으로 검사한다.

RESULT

12m 37s 동안 작업 >

완료했습니다. 문서 작업을 먼저 끝낸 뒤 그 문서의 System Interface 와 핵심 제어 규칙을 기준으로 C++20 구현, CLI 시뮬레이터, GoogleTest 테스트까지 붙였습니다.

주요 산출물

- 문서: [requirements.md](#), [ooa_ssd.md](#), [ooa_domain_diagram.md](#), [ood_sequence_diagrams.md](#), [ood_class_diagram.md](#), [traceability.md](#)
- 구현: [RvcController.hpp](#), [RvcController.cpp](#), [GridSimulator.cpp](#), [main.cpp](#)
- 테스트: [controller_tests.cpp](#), [system_tests.cpp](#)
- 실행 예시: [dust_and_interrupt.rvc](#), [backward_escape.rvc](#)
- 사용법: [README.md](#)

구현에는 전방 센서 interrupt, 좌/우/먼지 periodic sensor, Escaping 상태에서 탈출 가능할 때까지 Backward 반복, 먼지 감지 시 boost 청소를 반영했습니다.


검증도 끝났습니다.

```
powershell  
  
cmake --build build --config Debug  
ctest --test-dir build -C Debug --output-on-failure
```

결과: 11/11 테스트 통과했습니다. 시뮬레이터 예시도 실행 확인했고, dust_and_interrupt 시나리오는 먼지 2개를 모두 청소했습니다.

 **ooa_domain_diagram.md**
문서 · MD

열기 ▾


 **ooa_ssd.md**
문서 · MD

열기 ▾

 **ood_class_diagram.md**
문서 · MD

열기 ▾

4개 더 보기 ▾

 **파일 19개 편집됨**
+1,815 -6

실행 취소 ↶ 리뷰

docs/ooa_domain_diagram.md •

+107 -0 ▾

docs/ooa_ssd.md •

+141 -0 ▾

docs/ood_class_diagram.md •

+111 -0 ▾

16개 파일 더 보기 ▾

SRS

SRS

RVC Control SW Software Requirements Specification

본 문서는 IEEE Std 830-1998의 SRS 구조를 기준으로 작성한다.

1. Introduction

1.1 Purpose

이 문서는 RVC(Robotic Vacuum Cleaner) 자동 청소 제어 소프트웨어의 Software Requirements Specification(SRS)을 정의한다. 요구사항은 `docs/rvc.pdf` 의 원본 문제 정의, OOA 산출물, 현재 C++20 구현 및 테스트 가능한 CLI 시뮬레이터 범위를 기준으로 정리한다.

본 문서는 구현 방법이 아니라 시스템이 제공해야 할 외부 동작, 인터페이스, 기능 요구사항, 비기능 요구사항, 검증 기준을 명세한다.

1.2 Scope

대상 시스템은 RVC 자동 청소 제어 로직과 이를 검증하기 위한 CLI 그리드 시뮬레이터이다. 제어 소프트웨어는 추상화된 센서 입력을 받아 이동 및 청소 명령을 생성한다.

범위에 포함하는 항목은 다음과 같다.

- 자동 청소 시작 및 중지
- 전방 장애물 interrupt 처리
- 좌측, 우측, 먼지 센서의 periodic sampling
- 장애물 회피 및 막힌 영역 탈출
- 먼지 감지 시 boost 청소
- CLI 시뮬레이터를 통한 요구사항 검증

범위에 포함하지 않는 항목은 다음과 같다.

- 실제 모터 드라이버
- 실제 센서 하드웨어
- 배터리, 충전, 전원 관리
- 사용자 앱, 네트워크, 영구 저장소
- 물리적 모터 가속도, 센서 노이즈, 지도 작성 알고리즘

SRS

1.3 Definitions, Acronyms, and Abbreviations

용어	정의
RVC	Robotic Vacuum Cleaner, 자동 청소 로봇
Control SW	센서 입력을 해석해 이동 및 청소 명령을 결정하는 제어 소프트웨어
Controller	<code>RvcController</code> 로 구현되는 핵심 제어 객체
Simulator	<code>GridSimulator</code> 로 구현되는 그리드 기반 검증 환경
Tick	periodic 센서 값을 읽고 다음 명령을 결정하는 제어 주기
Interrupt	전방 장애물처럼 즉시 반응해야 하는 비동기 이벤트
Periodic Sensor	tick마다 주기적으로 샘플링되는 센서
Front Sensor	전방 장애물 감지 센서이며 interrupt 방식으로 동작
Left Sensor	좌측 장애물 감지 센서이며 periodic 방식으로 동작
Right Sensor	우측 장애물 감지 센서이며 periodic 방식으로 동작
Dust Sensor	현재 위치의 먼지 감지 센서이며 periodic 방식으로 동작
Cleaner	일반 청소 또는 강화 청소를 수행하는 청소 장치
Boost	먼지 감지 후 일정 tick 동안 유지되는 강화 청소 세기
Escaping	전방, 좌측, 우측이 모두 막혔을 때 탈출을 위해 후진하는 상태
OOA	Object-Oriented Analysis
SSD	System Sequence Diagram

SRS

1.4 References

문서	설명
IEEE Std 830-1998	Software Requirements Specifications 작성 기준
<code>docs/rvc.pdf</code>	원본 RVC Control SW 요구사항
<code>docs/requirements.md</code>	유스케이스, 기능 요구사항, 비기능 요구사항, 핵심 제어 규칙
<code>docs/ooa_domain_diagram.md</code>	OOA Domain Diagram
<code>docs/ooa_ssd.md</code>	OOA System Sequence Diagram 및 System Interface
<code>docs/sdd.md</code>	Software Design Description
<code>docs/traceability.md</code>	요구사항, 설계, 테스트 추적성
<code>include/rvc/*.hpp</code> , <code>src/*.cpp</code>	현재 C++20 구현
<code>tests/*.cpp</code>	GoogleTest 기반 단위 및 시스템 테스트

1.5 Overview

2장은 제품 관점, 주요 기능, 사용자와 외부 주체, 제약 및 가정을 설명한다. 3장은 외부 인터페이스, 기능 요구사항, 비기능 요구사항, 상태와 데이터, 제어 규칙을 명세한다. 부록은 OOA 분석 산출물, 검증 기준, 요구사항 추적성을 포함한다.

SRS

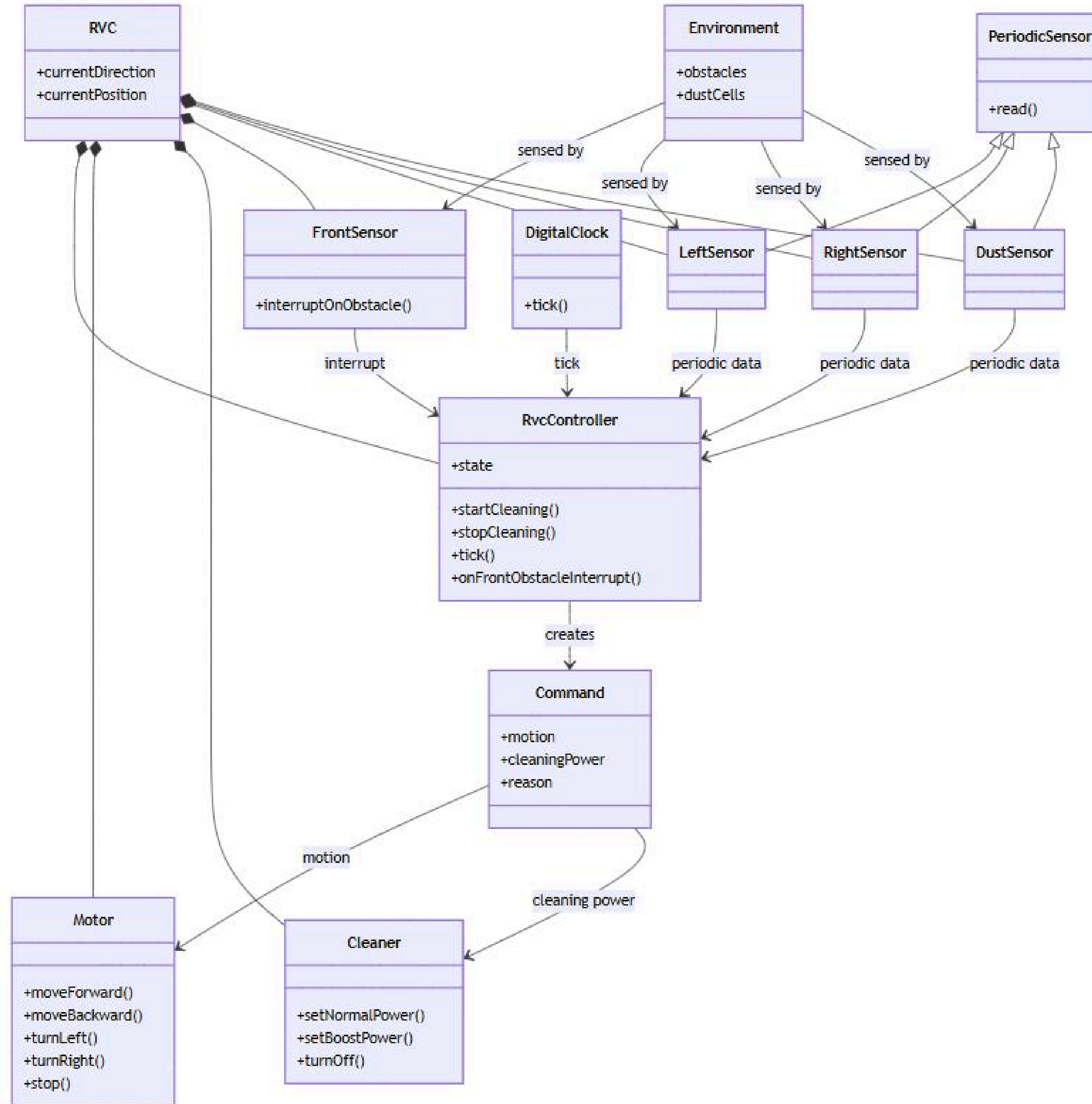
2. Overall Description

2.1 Product Perspective

RVC Control SW는 센서 입력과 사용자 요청을 받아 actuator 명령을 생성하는 제어 소프트웨어이다. 전방 장애물은 interrupt로 전달되고, 좌측/우측/먼지 센서는 제어 tick마다 periodic 데이터로 전달된다. 시스템은 두 입력 흐름을 판단 입력으로 결합하고 motor motion과 cleaner power를 포함한 command를 출력한다.

시뮬레이터는 실제 하드웨어가 아니라 요구사항 검증을 위한 외부 환경이다. 시뮬레이터는 지도에서 센서 값을 생성하고, controller가 반환한 command를 적용해 로그와 실행 결과를 만든다.

SRS



SRS

2.2 Product Functions

시스템은 다음 주요 기능을 제공해야 한다.

- 사용자의 자동 청소 시작 및 중지 요청을 처리한다.
- 전방 장애물 interrupt를 수신하면 즉시 회피 판단으로 전환한다.
- 좌측, 우측, 먼지 센서 값을 tick마다 주기적으로 샘플링한다.
- 전방 장애물 상황에서 열린 좌측 또는 우측 방향으로 회전한다.
- 좌우가 모두 열린 경우 좌우 회전 방향을 번갈아 선택한다.
- 전방, 좌측, 우측이 모두 막힌 경우 `Escaping` 상태로 전환하고 후진 명령을 지속한다.
- 먼지 감지 시 설정된 tick 동안 청소 세기를 `Boost` 로 유지한다.
- CLI 시뮬레이터를 통해 지도, 센서, 명령, 위치, 방향, 청소 결과를 검증한다.

2.3 User Characteristics

주체	특성 및 역할
User	자동 청소 시작 또는 중지를 요청한다. 내부 제어 상태나 센서 구조를 직접 다루지 않는다.
Tester	CLI 시뮬레이터와 테스트 코드를 통해 요구사항 충족 여부를 검증한다.
Front Sensor	전방 장애물을 interrupt 이벤트로 알린다.
Left Sensor	좌측 장애물 상태를 주기적으로 제공한다.
Right Sensor	우측 장애물 상태를 주기적으로 제공한다.
Dust Sensor	먼지 감지 상태를 주기적으로 제공한다.
Digital Clock	tick 기반 제어 주기를 제공한다.
Motor	<code>Forward</code> , <code>Backward</code> , <code>TurnLeft</code> , <code>TurnRight</code> , <code>Stop</code> 명령을 수행한다.
Cleaner	<code>Off</code> , <code>Normal</code> , <code>Boost</code> 청소 세기를 수행한다.

2.4 Constraints

- 전방 센서는 interrupt 방식으로 처리해야 한다.
- 좌측 센서, 우측 센서, 먼지 센서는 periodic 방식으로 처리해야 한다.
- 핵심 제어 로직은 하드웨어와 직접 결합되지 않아야 한다.
- 컨트롤러의 판단은 동일 입력에 대해 결정적이어야 한다.
- 프로젝트는 CMake와 C++20로 빌드되어야 한다.
- 단위 테스트는 GoogleTest 기반이어야 한다.
- 문서와 소스 파일은 UTF-8 인코딩을 사용해야 한다.

2.5 Assumptions and Dependencies

- 사용자의 자동 청소 시작 요청은 `startCleaning()` 호출로 추상화한다.
- 사용자의 자동 청소 중지 요청은 `stopCleaning()` 호출로 추상화한다.
- 전방 장애물 interrupt는 `onFrontObstacleInterrupt()` 호출로 추상화한다.
- tick마다 좌측, 우측, 먼지 센서 값이 `PeriodicSensorData` 로 제공된다.
- 실제 하드웨어의 물리적 지연, 모터 가속도, 센서 노이즈, 배터리 상태는 본 SRS의 범위 밖이다.
- 시뮬레이터에서 지도 밖 영역은 장애물로 간주한다.

2.6 Apportioning of Requirements

향후 실제 하드웨어 연동, 사용자 앱, 배터리 관리, 지도 작성 기능은 본 버전의 요구사항에 포함하지 않는다. 현재 버전은 추상화된 controller API와 CLI 시뮬레이터 검증에 초점을 둔다.

SRS

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

인터페이스	설명	기대 결과
청소 시작	사용자가 자동 청소를 시작한다.	컨트롤러가 <code>Cleaning</code> 상태가 되고 cleaner가 켜진 전진 청소 명령을 생성한다.
청소 중지	사용자가 자동 청소를 중지한다.	컨트롤러가 <code>Idle</code> 상태가 되고 motor는 정지, cleaner는 off가 된다.

3.1.2 Hardware Interfaces

본 SRS는 실제 하드웨어 드라이버를 정의하지 않는다. 다만 다음 추상 hardware interface 의미를 요구한다.

입력 또는 출력	방식	데이터	설명
전방 장애물	Interrupt	<code>frontObstacle</code>	전방 장애물이 감지되면 즉시 컨트롤러에 이벤트를 전달한다.
좌측 장애물	Periodic	<code>leftObstacle</code>	tick마다 좌측 방향의 장애물 여부를 제공한다.
우측 장애물	Periodic	<code>rightObstacle</code>	tick마다 우측 방향의 장애물 여부를 제공한다.
먼지 감지	Periodic	<code>dustDetected</code>	tick마다 현재 위치의 먼지 감지 여부를 제공한다.
Motor command	Actuator output	<code>motion</code>	이동, 회전, 정지 명령을 수행한다.
Cleaner command	Actuator output	<code>cleaningPower</code>	청소 장치 세기를 수행한다.

SRS

3.1.3 Software Interfaces

API	설명
<code>RvcController::startCleaning()</code>	자동 청소를 시작한다.
<code>RvcController::stopCleaning()</code>	자동 청소를 중지하고 boost 잔여 tick을 초기화한다.
<code>RvcController::onFrontObstacleInterrupt()</code>	실행 중 전방 장애물 interrupt를 기록한다.
<code>RvcController::tick(const PeriodicSensorData&)</code>	periodic 센서 값을 받아 다음 명령을 반환한다.
<code>RvcController::readPeriodicSensors(const PeriodicSensorData&)</code>	전방 interrupt 상태와 periodic 센서 값을 하나의 snapshot으로 결합한다.
<code>RvcController::decideNextCommand(const SensorSnapshot&)</code>	센서 snapshot을 기반으로 다음 명령을 결정한다.
<code>GridSimulator::run(int, bool)</code>	지정된 tick 수만큼 시뮬레이션을 실행한다.
<code>GridSimulator::loadScenario(const std::filesystem::path&)</code>	시나리오 파일을 읽어 tick 수와 지도 정보를 구성한다.

컨트롤러는 매 tick마다 다음 형태의 `Command` 를 생성한다.

출력	값	설명
<code>motion</code>	<code>None</code> , <code>Stop</code> , <code>Forward</code> , <code>Backward</code> , <code>TurnLeft</code> , <code>TurnRight</code>	모터 이동 또는 회전 명령
<code>cleaningPower</code>	<code>Off</code> , <code>Normal</code> , <code>Boost</code>	청소 장치 세기
<code>reason</code>	문자열	명령 결정 사유를 설명하는 디버그 및 로그용 메시지

SRS

3.1.4 Communications Interfaces

본 시스템은 네트워크 또는 외부 통신 프로토콜을 요구하지 않는다. CLI 시뮬레이터는 표준 입력이 아니라 command line argument와 scenario file을 사용한다.

실행 형식은 다음과 같다.

```
rvc_simulator [--ticks N] [--scenario FILE] [--quiet-map]
```

옵션	설명
<code>--help</code> , <code>-h</code>	사용법을 출력하고 종료한다.
<code>--ticks N</code>	실행할 제어 tick 수를 지정한다.
<code>--scenario FILE</code>	지정한 시나리오 파일을 로드한다.
<code>--quiet-map</code>	초기/최종 지도와 tick별 frame 출력을 생략하고 로그 중심으로 출력한다.

3.1.5 Scenario File Format

시나리오 파일은 선택적 tick 설정과 지도 섹션으로 구성된다.

```
ticks=10
map:
#####
#..*.....#
#.>..*#...#
#####
```

문자	의미
#	장애물
.	빈 칸
*	먼지
R	북쪽을 바라보는 로봇
^	북쪽을 바라보는 로봇
>	동쪽을 바라보는 로봇
v	남쪽을 바라보는 로봇
<	서쪽을 바라보는 로봇

시나리오 지도에는 로봇이 정확히 하나 있어야 한다. 지도 섹션이 없거나 로봇이 없거나 로봇이 둘 이상이면 오류로 처리한다.

3.1.6 Log Output Format

시뮬레이터는 tick마다 다음 정보를 포함하는 로그를 출력한다.

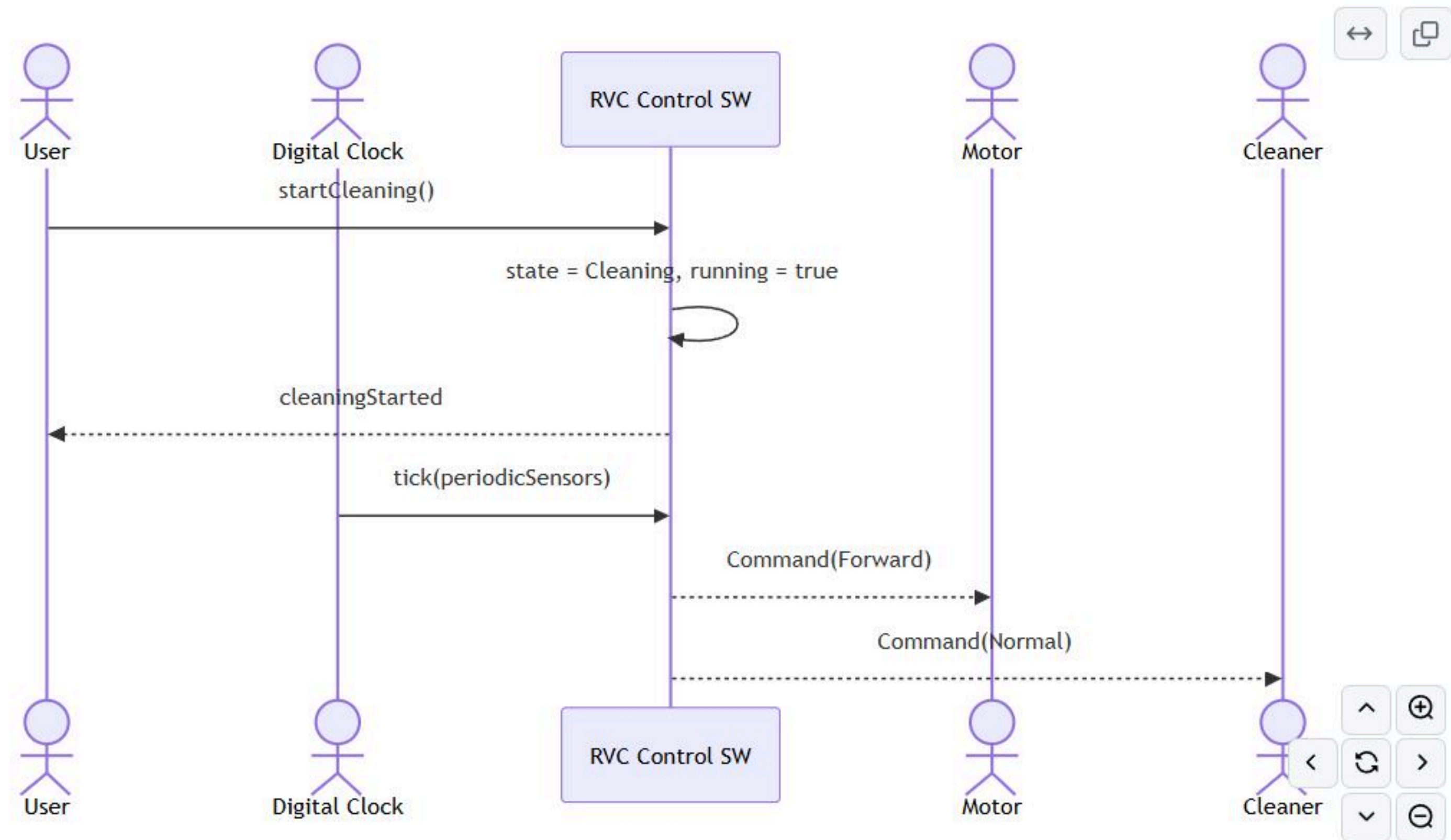
필드	설명
<code>tick</code>	현재 제어 tick 번호
<code>frontInterrupt</code>	전방 장애물 interrupt 발생 여부
<code>leftPeriodic</code>	좌측 periodic 센서 상태
<code>rightPeriodic</code>	우측 periodic 센서 상태
<code>dustPeriodic</code>	먼지 periodic 센서 상태
<code>motion</code>	컨트롤러가 생성한 이동 명령
<code>cleaner</code>	컨트롤러가 생성한 청소 세기
<code>position</code>	명령 적용 후 로봇 위치
<code>direction</code>	명령 적용 후 로봇 방향
<code>cleaned</code>	누적 청소 먼지 수
<code>reason</code>	명령 결정 사유

실행 종료 시 summary는 실행 tick 수, 청소된 먼지 수, 남은 먼지 수, 최종 위치, 최종 방향을 포함해야 한다.

SRS

3.2.1 UC-01 Start Automatic Cleaning / SSD-01

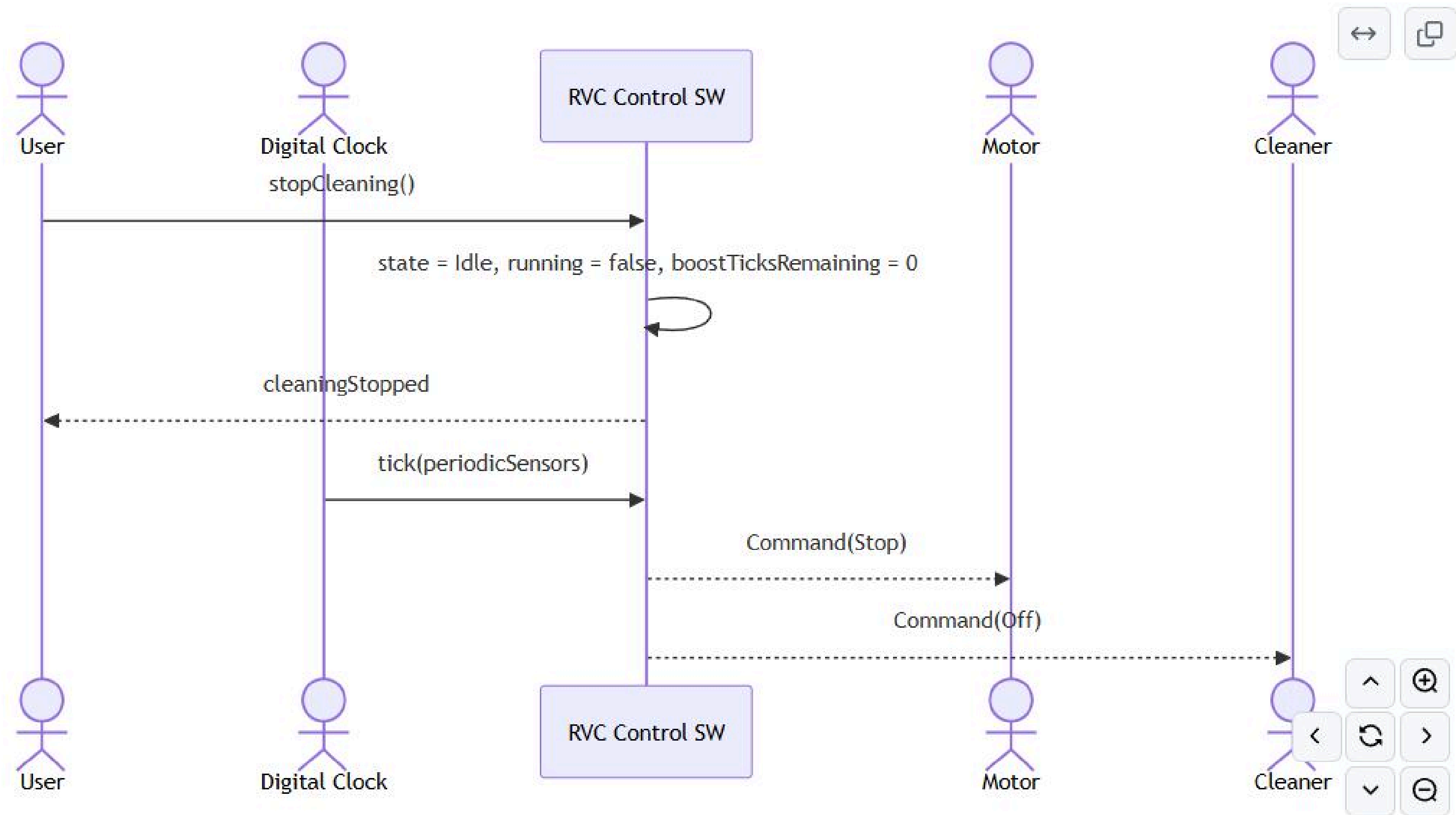
항목	내용
주요 주체	User
목표	RVC가 자동 청소를 시작한다.
사전 조건	제어 소프트웨어가 초기화되어 있다.
기본 흐름	사용자가 청소 시작을 요청하면 시스템은 Cleaning 상태로 전환하고 cleaner를 켜 상태로 전진 청소를 시작한다.
사후 조건	RVC는 자동 청소 상태가 된다.



SRS

3.2.2 UC-02 Stop Automatic Cleaning / SSD-06

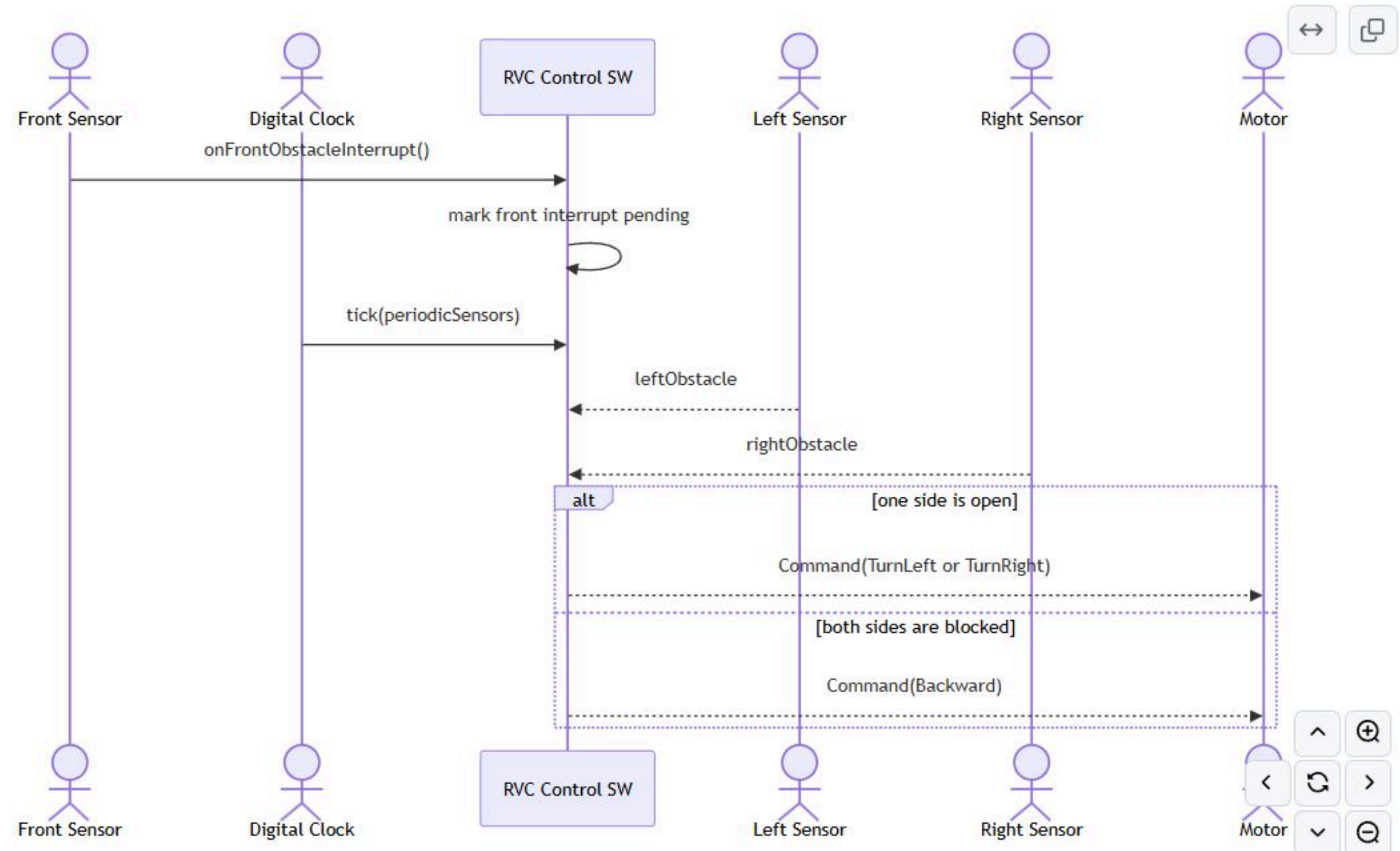
항목	내용
주요 주체	User
목표	RVC가 자동 청소를 중지한다.
사전 조건	RVC가 자동 청소 중이다.
기본 흐름	사용자가 청소 중지를 요청하면 시스템은 motor를 정지하고 cleaner를 끈다.
사후 조건	RVC는 Idle 상태가 된다.



SRS

3.2.3 UC-03 Avoid Front Obstacle / SSD-02

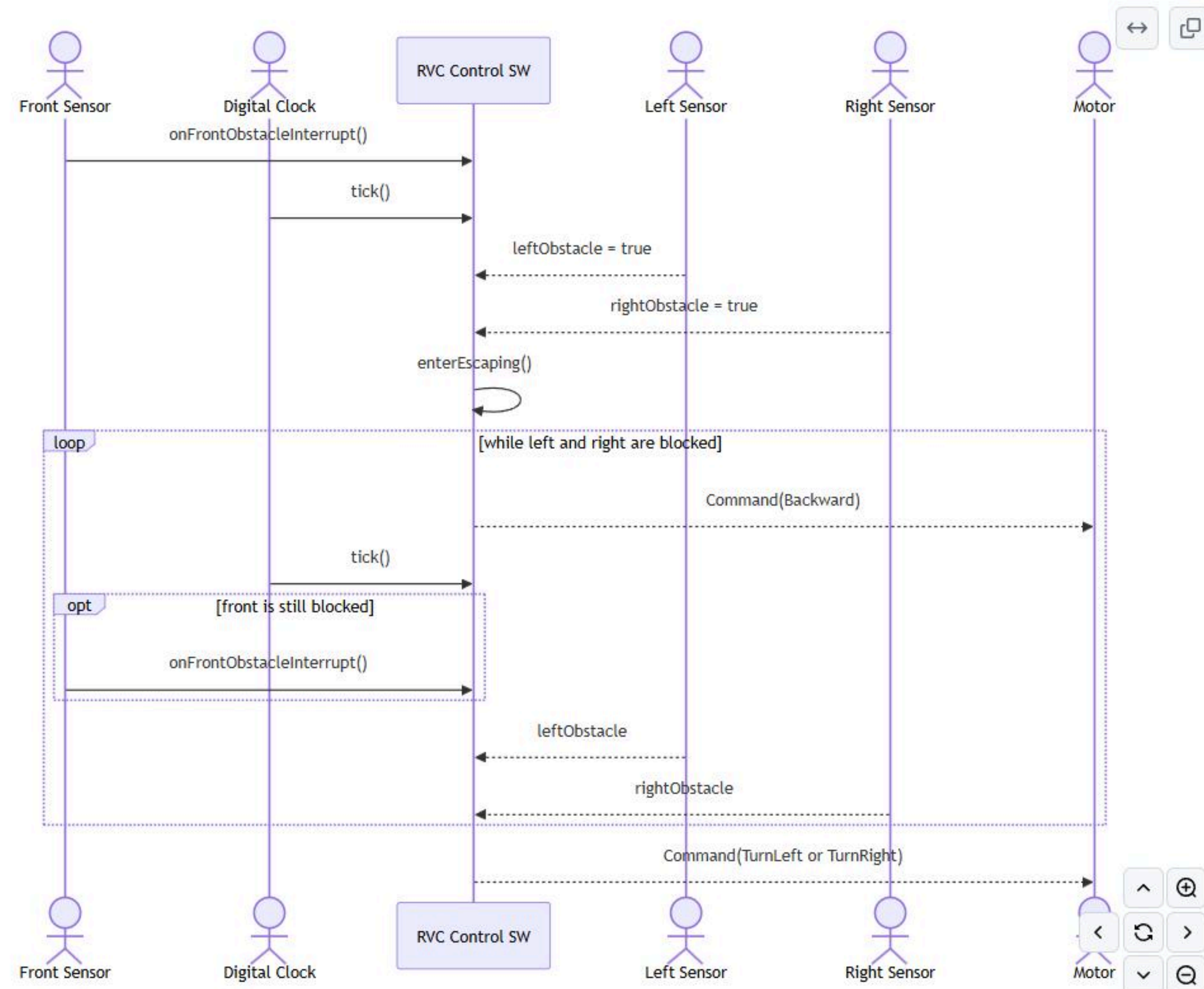
항목	내용
주요 주체	Front Sensor
목표	전방 장애물 감지 시 충돌 없이 회피한다.
사전 조건	RVC가 자동 청소 중이다.
기본 흐름	전방 센서가 interrupt를 발생시키면 시스템은 즉시 전진을 중단하고 좌측/우측 periodic 센서 값을 기준으로 열린 방향을 선택한다.
대안 흐름	좌우가 모두 열려 있으면 이전 선택과 반대 방향을 선택하여 좌우를 번갈아 회전한다.
사후 조건	RVC는 장애물을 회피하고 청소를 계속한다.



SRS

3.2.4 UC-04 Escape From Blocked Area / SSD-05

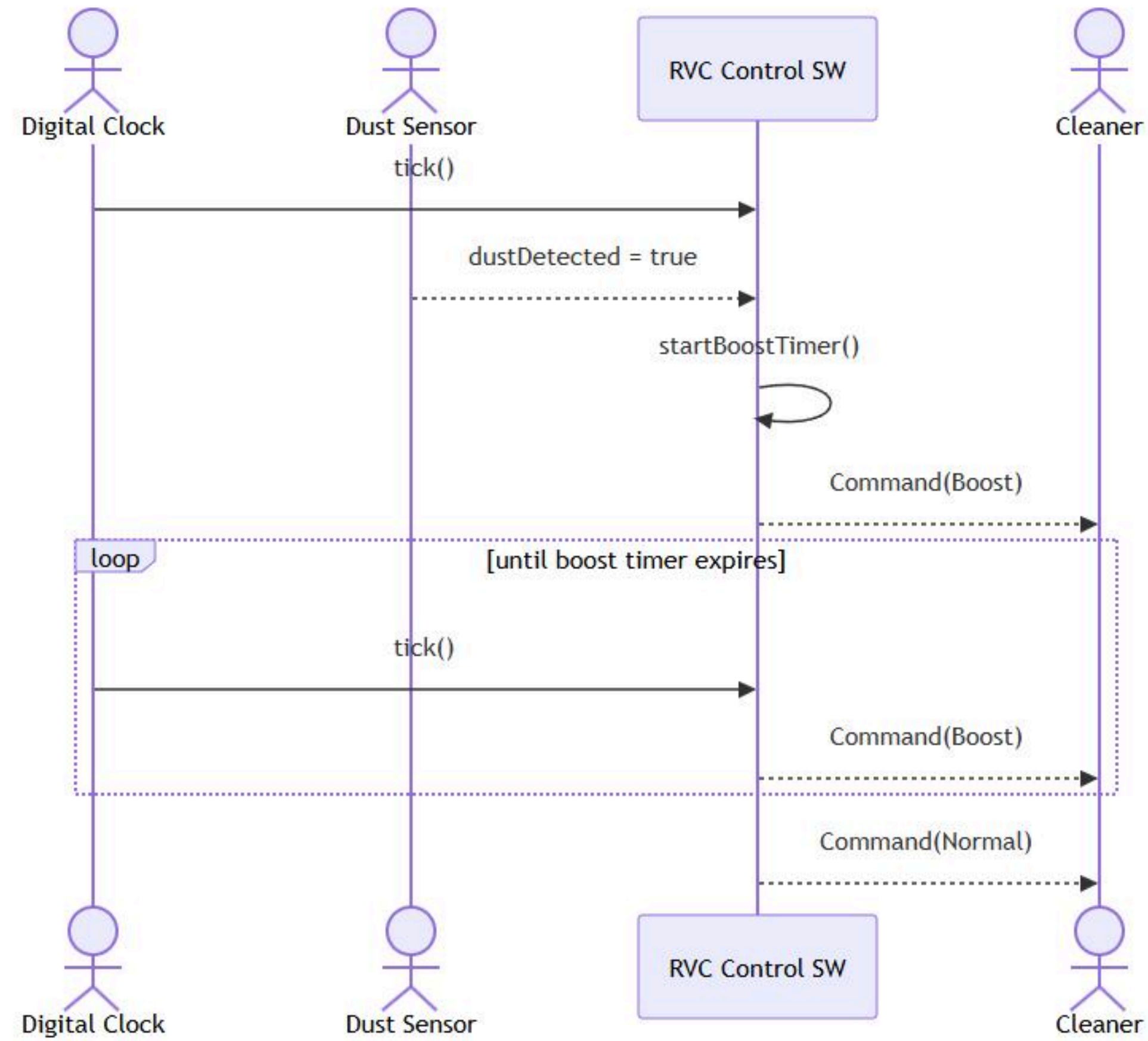
항목	내용
주요 주체	Front Sensor, Left Sensor, Right Sensor
목표	전방, 좌측, 우측이 모두 막힌 상황에서 탈출 가능한 상태까지 이동한다.
사전 조건	RVC가 자동 청소 중이며 전방, 좌측, 우측이 모두 막혀 있다.
기본 흐름	시스템은 Escaping 상태로 전환하고 좌측 또는 우측 측면 탈출구가 열릴 때까지 후진 명령을 지속한다.
사후 조건	탈출 가능해지면 열린 측면 방향으로 회전하고 청소를 계속할 수 있는 상태로 복귀한다.



SRS

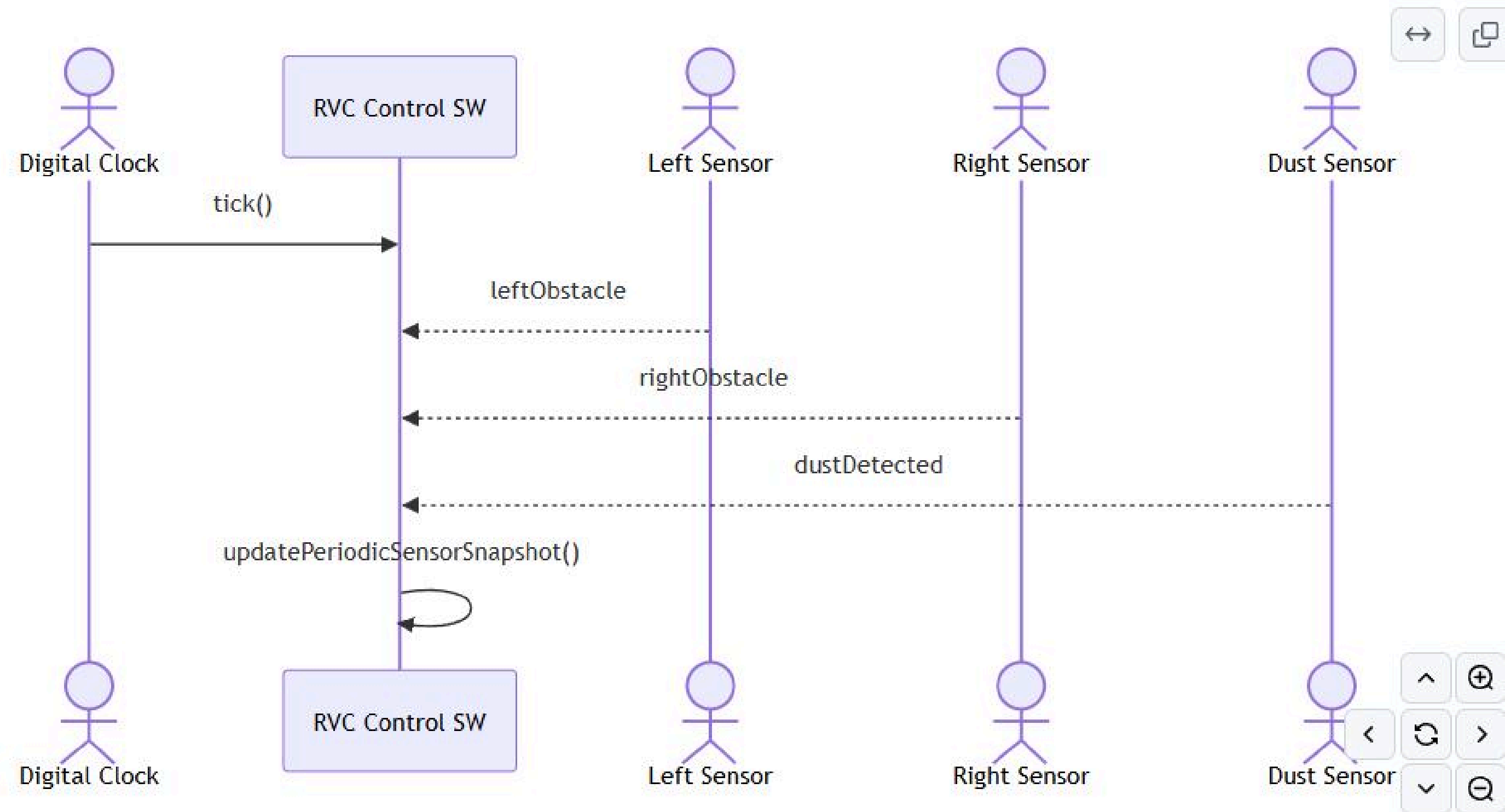
3.2.5 UC-05 Boost Cleaning On Dust / SSD-04

항목	내용
주요 주체	Dust Sensor
목표	먼지 감지 구간에서 청소 세기를 높인다.
사전 조건	RVC가 자동 청소 중이다.
기본 흐름	먼지 센서가 먼지를 감지하면 시스템은 cleaner를 Boost 로 설정하고 설정된 tick 동안 유지한다.
사후 조건	boost 시간이 만료되고 새 먼지가 감지되지 않으면 cleaner는 Normal 로 복귀한다.



SRS

3.2.6 Periodic Sensor Sampling / SSD-03



ID	요구사항	수용 기준	관련 항목
FR-01	System shall start automatic cleaning when requested.	<code>startCleaning()</code> 호출 후 컨트롤러는 실행 중 상태가 되고 다음 tick에서 청소 명령을 생성한다.	UC-01
FR-02	System shall stop motor and cleaner when cleaning is stopped.	<code>stopCleaning()</code> 호출 후 tick 결과는 <code>motion=Stop</code> , <code>cleaningPower=Off</code> 가 된다.	UC-02
FR-03	System shall move forward while cleaning when no obstacle blocks the front direction.	전방 interrupt가 없으면 <code>motion=Forward</code> , <code>cleaningPower=Normal</code> 또는 <code>Boost</code> 를 반환한다.	UC-01
FR-04	Front obstacle detection shall be handled as an interrupt event.	전방 장애물은 periodic 입력이 아니라 <code>onFrontObstacleInterrupt()</code> 이벤트로 컨트롤러에 전달된다.	UC-03
FR-05	System shall stop forward motion immediately when a front obstacle interrupt is received.	interrupt가 pending 상태인 tick에서 컨트롤러는 <code>Forward</code> 대신 회피, 회전, 또는 후진 명령을 반환한다.	UC-03
FR-06	Left, right, and dust sensors shall be sampled periodically on control tick.	<code>tick()</code> 은 <code>PeriodicSensorData</code> 의 좌측, 우측, 먼지 값을 사용하여 명령을 결정한다.	UC-03, UC-05
FR-07	If front is blocked and only left is open, System shall turn left.	<code>frontObstacle=true</code> , <code>leftObstacle=false</code> , <code>rightObstacle=true</code> 이면 <code>motion=TurnLeft</code> 를 반환한다.	UC-03
FR-08	If front is blocked and only right is open, System shall turn right.	<code>frontObstacle=true</code> , <code>leftObstacle=true</code> , <code>rightObstacle=false</code> 이면 <code>motion=TurnRight</code> 를 반환한다.	UC-03
FR-09	If front is blocked and both left and right are open, System shall choose turn direction by alternating left and right.	동일 조건이 반복되면 첫 선택은 좌회전, 다음 선택은 우회전처럼 방향을 번갈아 반환한다.	UC-03
FR-10	If front, left, and right are all blocked, System shall enter <code>Escaping</code> state.	세 방향이 모두 막힌 tick 이후 컨트롤러 상태는 <code>Escaping</code> 이 된다.	UC-04

SRS

FR-11	In <code>Escaping</code> state, System shall keep moving backward until escape is possible.	<code>Escaping</code> 상태에서 좌우 측면이 계속 막혀 있으면 매 tick <code>motion=Backward</code> 를 반환한다.	UC-04
FR-12	Escape shall be considered possible when at least one side direction is open.	<code>Escaping</code> 상태에서 좌측 또는 우측 중 하나 이상이 열리면 탈출 가능 상태로 판단한다.	UC-04
FR-13	After escape becomes possible, System shall turn toward an open side.	열린 측면 방향으로 <code>TurnLeft</code> 또는 <code>TurnRight</code> 명령을 반환한다.	UC-04
FR-14	If dust is detected, System shall set cleaner power to boost for a configured number of ticks.	<code>dustDetected=true</code> 인 tick에서 <code>cleaningPower=Boost</code> 가 되고 <code>dustBoostTicks</code> 동안 유지된다.	UC-05
FR-15	If boost duration expires and no new dust is detected, System shall return cleaner power to normal.	먼지가 새로 감지되지 않고 boost 잔여 tick이 끝나면 <code>cleaningPower=Normal</code> 로 복귀한다.	UC-05
FR-16	Simulator shall render a grid map with robot, obstacle, dust, and empty cells.	지도 출력은 로봇 방향 문자, #, ., * 를 사용해 현재 상태를 표현한다.	VS-01
FR-17	Simulator shall log tick, sensor values, command, robot position, direction, and cleaning power.	각 tick 로그는 센서 상태, motion, cleaner, position, direction, cleaned, reason을 포함한다.	VS-01
FR-18	Simulator shall use the same controller interface as the production control SW.	시뮬레이터는 <code>RvcController</code> 에 interrupt와 periodic 데이터를 전달하고 반환된 <code>Command</code> 를 적용한다.	VS-01

SRS

3.3 Performance Requirements

- 컨트롤러의 tick 판단은 단일 제어 주기 안에서 완료되어야 한다.
- 현재 범위에서는 실시간 deadline, memory limit, CPU limit을 정량적으로 요구하지 않는다.
- 같은 초기 상태와 같은 입력 순서에 대해 같은 command sequence가 생성되어야 한다.

3.4 Logical Database Requirements

본 시스템은 데이터베이스 또는 영구 저장소를 요구하지 않는다. 시나리오 파일과 실행 로그는 검증용 입출력으로만 사용한다.

3.5 Design Constraints

- 구현 언어는 C++20이다.
- 빌드 시스템은 CMake를 사용한다.
- 단위 테스트와 시스템 테스트는 GoogleTest 및 CTest 기반으로 수행한다.
- 센서 입력과 actuator 명령은 구조체와 enum으로 추상화되어야 한다.
- Controller logic은 concrete hardware 및 simulator 구현에 직접 의존하지 않아야 한다.

SRS

3.6 Software System Attributes

ID	요구사항	수용 기준
NFR-01	Controller logic shall be independent from concrete hardware and simulator classes.	<code>RvcController</code> 는 하드웨어 또는 <code>GridSimulator</code> 구체 타입에 직접 의존하지 않는다.
NFR-02	Controller shall be testable through deterministic inputs without real hardware.	단위 테스트에서 <code>PeriodicSensorData</code> 와 <code>interrupt</code> 호출만으로 동작을 검증할 수 있다.
NFR-03	The project shall build with CMake and C++20.	<code>cmake -S . -B build</code> 및 CMake build 구성을 지원한다.
NFR-04	Unit tests shall be written with GoogleTest.	컨트롤러 단위 테스트는 GoogleTest 기반으로 작성된다.
NFR-05	System tests shall exercise the controller through the simulator.	시스템 테스트는 <code>GridSimulator</code> 를 통해 컨트롤러 명령과 결과를 검증한다.
NFR-06	Sensor and actuator abstractions shall allow future sensor changes or additions.	센서 입력과 명령 출력은 구조체와 enum으로 추상화되어 있다.
NFR-07	CLI simulator output shall be human-readable for manual review.	로그는 key-value 형식과 readable reason 문자열을 포함한다.
NFR-08	Core control decisions shall be deterministic for repeatable tests.	같은 초기 상태와 같은 입력 순서는 같은 명령 순서를 생성한다.
NFR-09	Source files and documents shall be encoded in UTF-8.	한글 문서가 깨지지 않아야 하며 UTF-8로 저장되어야 한다.
NFR-10	The design shall follow SOLID principles where applicable.	제어 로직, 시뮬레이션, 타입 정의가 역할별로 분리되어야 한다.

SRS

3.7 State and Data Requirements

3.7.1 Controller States

상태	설명	진입 조건	이탈 조건
Idle	청소하지 않는 대기 상태	초기 상태 또는 청소 중지 요청	청소 시작 요청
Cleaning	기본 자동 청소 상태	청소 시작, 전방이 열린 상태, 탈출 완료	전방 interrupt, 청소 중지
Avoiding	전방 장애물 회피를 위해 회전하는 상태	전방 interrupt 후 열린 좌/우 방향 존재	다음 tick에서 전방 상태에 따라 청소 또는 탈출
Escaping	세 방향 막힘 상황에서 후진하는 상태	전방, 좌측, 우측이 모두 막힘	좌측 또는 우측 중 하나 이상 열림 또는 청소 중지

3.7.2 Core Data Types

타입	필드 또는 값	설명
PeriodicSensorData	leftObstacle, rightObstacle, dustDetected	tick마다 샘플링되는 센서 입력
SensorSnapshot	frontObstacle, leftObstacle, rightObstacle, dustDetected	interrupt와 periodic 센서를 결합한 판단용 snapshot
Command	motion, cleaningPower, reason	컨트롤러가 생성하는 actuator 명령
Motion	None, Stop, Forward, Backward, TurnLeft, TurnRight	이동 또는 회전 명령 종류
CleaningPower	Off, Normal, Boost	cleaner 출력 세기
ControllerState	Idle, Cleaning, Avoiding, Escaping	컨트롤러 내부 상태
Position	row, col	시뮬레이터 내 로봇 좌표
Direction	North, East, South, West	시뮬레이터 내 로봇 방향

SRS

3.7.3 Core Control Rules

1. 청소가 시작되면 RVC는 cleaner를 켜 상태로 전진 청소를 수행한다.
2. 전방 장애물이 감지되면 interrupt 이벤트로 처리하고 해당 tick에서 전진 명령을 내리지 않는다.
3. 좌측, 우측, 먼지 센서는 tick마다 periodic 데이터로 반영한다.
4. 전방이 막히고 좌측만 열려 있으면 좌회전한다.
5. 전방이 막히고 우측만 열려 있으면 우회전한다.
6. 전방이 막히고 좌측과 우측이 모두 열려 있으면 좌우를 번갈아 선택한다.
7. 전방, 좌측, 우측이 모두 막히면 `Escaping` 상태로 전환한다.
8. `Escaping` 상태에서는 측면 탈출구가 열릴 때까지 후진 명령을 지속한다.
9. 탈출 가능 조건은 좌측 또는 우측 중 하나 이상이 열린 상태이다.
10. 탈출 가능해진 뒤 열린 측면 방향으로 회전한다.
11. 먼지가 감지되면 `dustBoostTicks` 만큼 `Boost` 를 유지한다.
12. boost 유지 시간이 끝나고 새 먼지가 감지되지 않으면 `Normal` 로 복귀한다.

SRS

Appendix A. OOA Analysis Summary

A.1 Domain Concepts

Concept	Responsibility
RVC	물리적 로봇 청소기 전체를 의미한다.
RvcController	센서 입력과 interrupt를 기반으로 motor/cleaner 명령을 결정한다.
FrontSensor	전방 장애물을 interrupt로 알린다.
LeftSensor	좌측 장애물 상태를 periodic 방식으로 제공한다.
RightSensor	우측 장애물 상태를 periodic 방식으로 제공한다.
DustSensor	현재 위치의 먼지 감지 상태를 periodic 방식으로 제공한다.
DigitalClock	제어 tick을 발생시킨다.
Motor	전진, 후진, 좌회전, 우회전, 정지 동작을 수행한다.
Cleaner	normal power, boost power, off 상태를 수행한다.
Environment	장애물과 먼지가 있는 청소 공간이다.
Command	controller가 actuator에 전달하는 추상 명령이다.

A.2 Important Domain Rules

- FrontSensor는 polling 대상이 아니라 interrupt source이다.
- LeftSensor, RightSensor, DustSensor는 DigitalClock tick에 맞춰 sampling된다.
- RvcController는 sensor와 actuator의 구체 구현을 알지 않는다.
- Escaping 상태에서는 후방 센서 없이 backward command를 반복한다.
- Simulator의 Environment는 실제 하드웨어가 아니라 테스트용 외부 세계이다.

A.3 System Operations

Operation	Related FR	Notes
startCleaning()	FR-01, FR-03	실행 상태를 시작하며 실제 전진/청소 명령은 다음 tick() 에서 생성된다.
stopCleaning()	FR-02	실행 상태와 boost timer를 초기화하며 다음 tick() 에서 Stop / Off command가 생성된다.
onFrontObstacleInterrupt()	FR-04, FR-05	interrupt는 다음 tick() 보다 먼저 들어올 수 있다.
tick(periodicSensors)	FR-06	Digital Clock의 제어 주기마다 호출된다.
decideNextCommand(snapshot)	FR-07 to FR-15	회피, 탈출, boost 규칙을 포함한다.

SRS

Appendix B. Verification Criteria

B.1 Build and Test Commands

프로젝트 검증은 다음 명령을 기준으로 수행할 수 있다.

```
cmake -S . -B build
cmake --build build --config Debug
ctest --test-dir build -C Debug --output-on-failure
```

B.2 Unit Test Criteria

테스트	검증 요구사항
ControllerMovesForwardWhenPathIsClear	FR-01, FR-03
StopCleaningReturnsStopAndOff	FR-02
FrontInterruptTriggersImmediateAvoidance	FR-04, FR-05
TurnsTowardOpenSide	FR-07, FR-08
AlternatesWhenBothSidesAreOpen	FR-09
AllBlockedEntersEscapingAndKeepsBackingUp	FR-10, FR-11
EscapingIgnoresOpenFrontUntilSideOpens	FR-12, FR-13
DustBoostLastsConfiguredTicks	FR-14, FR-15
AvoidanceOutputStaysOffWhileBoostStateIsMaintained	FR-05, FR-14, FR-15

B.3 System Test Criteria

테스트	검증 요구사항
SimulatorCleansDustAndLogsCommands	FR-14, FR-15, FR-17
SimulatorUsesBackwardEscape	FR-10, FR-11, FR-18
SimulatorKeepsCommandingBackwardWhenBoxedIn	FR-11
SimulatorKeepsBackingUpUntilSideExitOpens	FR-11, FR-12, FR-13
SimulatorKeepsCleanerOffDuringBoostedEscape	FR-10, FR-11, FR-14, FR-15
SimulatorTurnsAfterFrontInterrupt	FR-04, FR-05, FR-08
SimulatorCliDefaultRuns	FR-16, FR-17
SimulatorCliContinuousBackwardScenarioRuns	FR-10, FR-11, FR-16, FR-17

B.4 Manual Review Criteria

- 시뮬레이터 로그에서 전방 장애물 interrupt와 회피 명령이 같은 tick 흐름에서 확인되어야 한다.
- 먼지가 있는 칸에서 `dustPeriodic=detected` 와 `cleaner=Boost` 가 확인되어야 한다.
- 세 방향이 막힌 지도에서는 `motion=Backward` 가 반복되어야 한다.
- `--quiet-map` 옵션을 사용하면 지도 frame 출력 없이 로그와 summary 중심으로 확인할 수 있어야 한다.
- 문서의 요구사항 ID와 테스트 추적성이 `docs/traceability.md` 와 모순되지 않아야 한다.

B.5 Verification Support Item

시뮬레이터 시나리오 실행은 RVC의 사용자 또는 센서 중심 유스케이스가 아니라 제어 소프트웨어 구현을 검증하기 위한 지원 기능으로 분리한다.

ID	목적	관련 요구사항
VS-01	CLI 그리드 시뮬레이터로 컨트롤러 구현을 실행하고 지도 렌더링, tick 로그, production controller interface 사용 여부를 검증한다.	FR-16, FR-17, FR-18

SRS

Appendix C. Requirements Traceability

C.1 Requirement to Design Traceability

Requirement	Use Case / Verification Item	OOA SSD	OOD SD	Class
FR-01	UC-01	SSD-01	SD-01	RvcController
FR-02	UC-02	SSD-06	SD-01	RvcController
FR-03	UC-01	SSD-01	SD-01, SD-04	RvcController, Command
FR-04	UC-03	SSD-02	SD-02	RvcController
FR-05	UC-03	SSD-02	SD-02, SD-04	RvcController, Command
FR-06	UC-03, UC-05	SSD-03	SD-03	PeriodicSensorData, SensorSnapshot
FR-07	UC-03	SSD-02	SD-04	RvcController
FR-08	UC-03	SSD-02	SD-04	RvcController
FR-09	UC-03	SSD-02	SD-04	RvcController
FR-10	UC-04	SSD-05	SD-05	RvcController
FR-11	UC-04	SSD-05	SD-05	RvcController
FR-12	UC-04	SSD-05	SD-05	SensorSnapshot
FR-13	UC-04	SSD-05	SD-05	RvcController
FR-14	UC-05	SSD-04	SD-06	RvcController
FR-15	UC-05	SSD-04	SD-06	RvcController
FR-16	VS-01	SSD-03	SD-01	GridSimulator
FR-17	VS-01	SSD-03	SD-01	GridSimulator, SimulationResult
FR-18	VS-01	SSD-01 to SSD-05	SD-01	RvcController, GridSimulator

C.2 Requirement to Test Traceability

Requirement	Test Case
FR-01, FR-03	ControllerMovesForwardWhenPathIsClear
FR-02	StopCleaningReturnsStopAndOff
FR-04, FR-05	FrontInterruptTriggersImmediateAvoidance, SimulatorTurnsAfterFrontInterrupt
FR-07, FR-08	FrontInterruptTriggersImmediateAvoidance, TurnsTowardOpenSide, SimulatorTurnsAfterFrontInterrupt
FR-09	AlternatesWhenBothSidesAreOpen
FR-10, FR-11	AllBlockedEntersEscapingAndKeepsBackingUp, SimulatorUsesBackwardEscape, SimulatorKeepsCommandingBackwardWhenBoxedIn, SimulatorKeepsCleanerOffDuringBoostedEscape
FR-12, FR-13	EscapingIgnoresOpenFrontUntilSideOpens, SimulatorKeepsBackingUpUntilSideExitOpens
FR-14, FR-15	DustBoostLastsConfiguredTicks, AvoidanceOutputStaysOffWhileBoostStateIsMaintained, SimulatorCleansDustAndLogsCommands, SimulatorKeepsCleanerOffDuringBoostedEscape
FR-16, FR-17, FR-18	SimulatorCleansDustAndLogsCommands, SimulatorUsesBackwardEscape, SimulatorKeepsBackingUpUntilSideExitOpens, SimulatorCliDefaultRuns, SimulatorCliContinuousBackwardScenarioRuns

SDD

SDD

RVC Control SW Software Design Description

본 문서는 IEEE Std 1016-2009의 SDD 구조를 기준으로 작성한다.

1. Scope

1.1 Purpose

이 문서는 RVC(Robotic Vacuum Cleaner) 자동 청소 제어 소프트웨어의 Software Design Description(SDD)이다. docs/srs.md 에 정의된 요구사항을 현재 C++20 구현 구조와 연결하고, 설계 context, stakeholder concern, design viewpoint, design view, design rationale을 정리한다.

1.2 System Scope

설계 대상은 RvcController 기반 제어 로직과 GridSimulator 기반 CLI 시뮬레이터이다. controller는 sensor 입력을 판단해 motor motion과 cleaner power를 포함한 Command 를 생성한다. simulator는 검증 환경으로서 scenario map에서 sensor 값을 만들고 command를 적용한다.

본 SDD의 범위에 포함하지 않는 항목은 실제 모터 드라이버, 실제 센서 하드웨어, 배터리, 네트워크, UI, 영구 저장소 설계이다.

1.3 Design Goals

목표	설명
요구사항 충족	FR-01부터 FR-18까지의 자동 청소, 장애물 회피, 탈출, 먼지 boost, 시뮬레이터 검증 요구사항을 만족한다.
결정적 동작	동일한 초기 상태와 동일한 센서 입력 순서에 대해 동일한 명령을 생성한다.
하드웨어 독립성	controller는 실제 하드웨어나 simulator 구현에 직접 의존하지 않고 값 객체와 command만 사용한다.
테스트 가능성	controller 단위 테스트와 simulator 시스템 테스트로 핵심 흐름을 검증한다.
단순한 확장성	sensor 입력과 actuator 명령을 구조체와 enum으로 분리하여 향후 sensor나 command 확장을 쉽게 한다.

SDD

2. References

문서	설명
IEEE Std 1016-2009	Software Design Descriptions 작성 기준
<code>docs/rvc.pdf</code>	원본 RVC Control SW 요구사항
<code>docs/srs.md</code>	Software Requirements Specification
<code>docs/requirements.md</code>	유스케이스, FR, NFR, 핵심 제어 규칙
<code>docs/ooa_domain_diagram.md</code>	OOA domain model
<code>docs/ooa_ssd.md</code>	System Sequence Diagram과 system operation
<code>docs/ood_class_diagram.md</code>	OOD class diagram과 SOLID 분석
<code>docs/ood_sequence_diagrams.md</code>	OOD sequence diagram
<code>docs/traceability.md</code>	요구사항, 설계, 테스트 추적성
<code>include/rvc/*.hpp</code> , <code>src/*.cpp</code>	현재 C++20 설계와 구현
<code>tests/*.cpp</code>	GoogleTest 기반 단위/시스템 테스트

3. Terms and Definitions

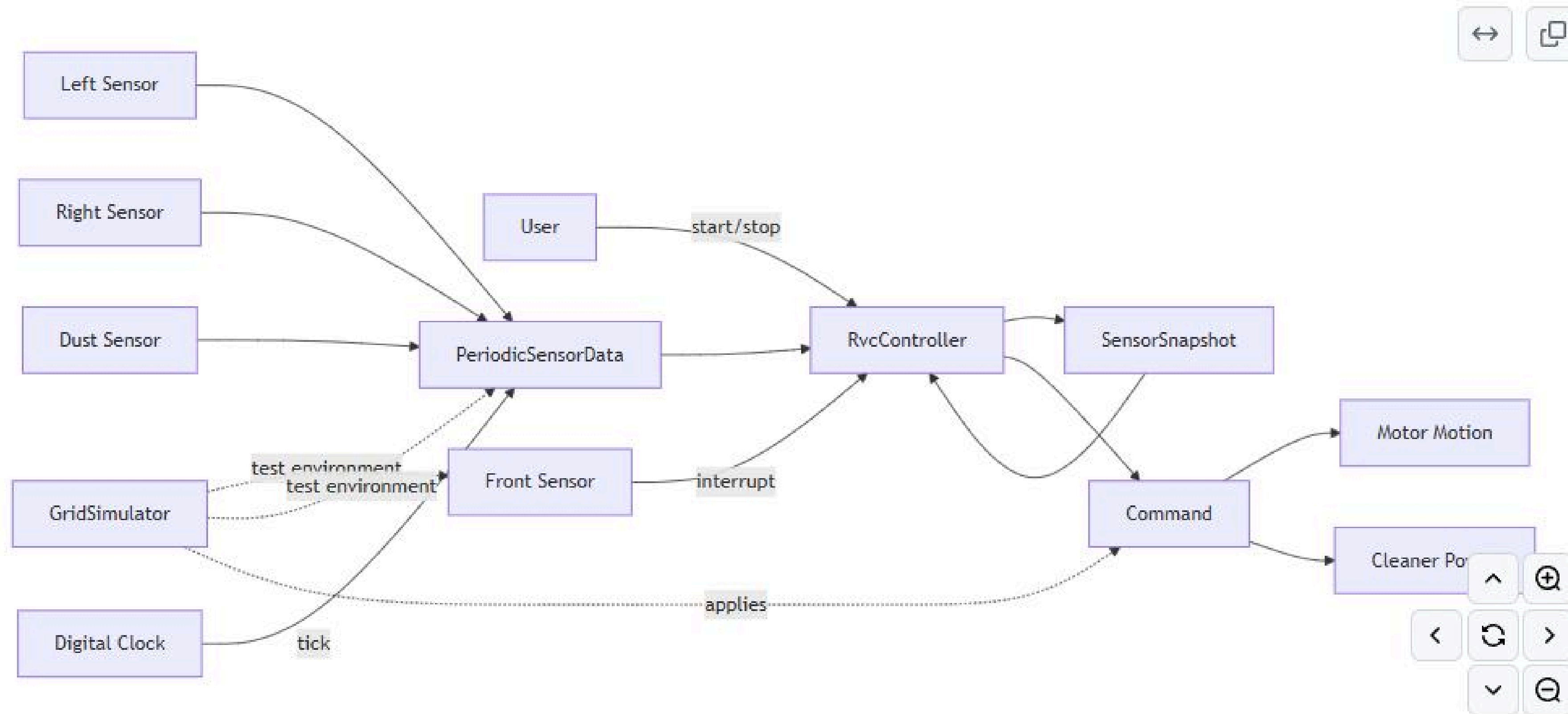
용어	정의
Design Context	설계 대상 시스템의 경계, 외부 주체, 운영 환경을 설명하는 정보
Design Stakeholder	설계 결과에 이해관계가 있는 사용자, 검증자, 구현자
Design Concern	stakeholder가 설계에서 확인해야 하는 관심사
Design Viewpoint	특정 concern을 다루기 위한 표현 규칙과 분석 관점
Design View	viewpoint를 적용해 작성한 실제 설계 표현
Controller	<code>RvcController</code> 로 구현되는 핵심 제어 객체
Simulator	<code>GridSimulator</code> 로 구현되는 검증 환경
Command	controller가 actuator 또는 simulator에 전달하는 추상 명령
SensorSnapshot	pending front interrupt와 periodic sensor 값을 결합한 판단 입력
Escaping	삼방향이 막힌 상황에서 후진 탈출을 수행하는 controller 상태

SDD

4. Design Context

4.1 System Boundary

RVC Control SW는 sensor 입력을 읽고 actuator 명령을 결정하는 controller 중심 구조이다. 전방 장애물은 interrupt로 controller에 기록되고, 좌측/우측/먼지 sensor는 제어 tick마다 periodic 데이터로 전달된다. controller는 두 입력 흐름을 `SensorSnapshot` 으로 결합한 뒤 `Command` 를 생성한다.



SDD

4.2 External Actors and Environment

외부 요소	설계상 관계
User	<code>startCleaning()</code> 과 <code>stopCleaning()</code> 요청으로 controller 상태를 전환한다.
Front Sensor	<code>onFrontObstacleInterrupt()</code> 로 전방 장애물 event를 전달한다.
Left Sensor	<code>PeriodicSensorData::leftObstacle</code> 로 좌측 장애물 상태를 전달한다.
Right Sensor	<code>PeriodicSensorData::rightObstacle</code> 로 우측 장애물 상태를 전달한다.
Dust Sensor	<code>PeriodicSensorData::dustDetected</code> 로 먼지 감지 상태를 전달한다.
Digital Clock	<code>tick()</code> 호출 주기를 제공한다.
Motor	<code>Command::motion</code> 값을 수행한다.
Cleaner	<code>Command::cleaningPower</code> 값을 수행한다.
GridSimulator	실제 하드웨어 대신 <code>sensor/event</code> 를 만들고 <code>command</code> 를 격자 상태에 적용한다.

4.3 Design Constraints

- C++20과 CMake 기반으로 빌드한다.
- Controller는 concrete hardware 및 `GridSimulator` 세부 구현에 의존하지 않는다.
- Front sensor는 interrupt, left/right/dust sensor는 periodic 입력으로 분리한다.
- 출력은 motor motion과 cleaner power를 포함하는 `Command` 로 통일한다.
- 테스트는 deterministic input sequence를 기반으로 반복 가능해야 한다.
- 모든 Markdown 문서는 UTF-8로 작성한다.

SDD

5. Design Stakeholders and Concerns

Stakeholder	Concern	설계 대응
User	청소 시작, 중지, 장애물 회피, 먼지 청소가 요구사항대로 동작해야 한다.	controller 상태와 command 규칙을 명시적으로 정의한다.
Tester	요구사항별 동작을 단위 및 시스템 테스트로 재현할 수 있어야 한다.	sensor 입력과 command 출력을 값 객체로 제공하고 simulator 로 그를 남긴다.
Implementer	제어 로직, simulator, 공용 타입의 책임 경계가 명확해야 한다.	<code>RvcController</code> , <code>GridSimulator</code> , core type을 모듈별로 분리한다.
Maintainer	sensor나 command 확장 시 영향 범위가 제한되어야 한다.	입력 구조체와 enum 기반 command를 사용한다.
Reviewer	SRS 요구사항과 설계 요소, 테스트가 추적 가능해야 한다.	FR별 설계 요소와 테스트 매핑을 유지한다.

6. Design Viewpoints

Viewpoint	Concern	주요 표현
Structural Viewpoint	모듈과 클래스 책임 분리	모듈 표, class diagram, 책임 표
Interface Viewpoint	public API와 입출력 계약	operation table, command table
Data Viewpoint	핵심 데이터 구조와 enum	data type table
Interaction Viewpoint	객체 간 실행 흐름	sequence diagram
State Viewpoint	controller 상태와 전이	state diagram, 상태 표
Algorithm Viewpoint	회피, 탈출, boost 판단 규칙	절차 목록, 조건 표
Error Handling Viewpoint	simulator 입력 오류와 controller 상태 처리	오류 처리 표
Verification Viewpoint	요구사항과 테스트 연결	traceability table

SDD

7. Design Views

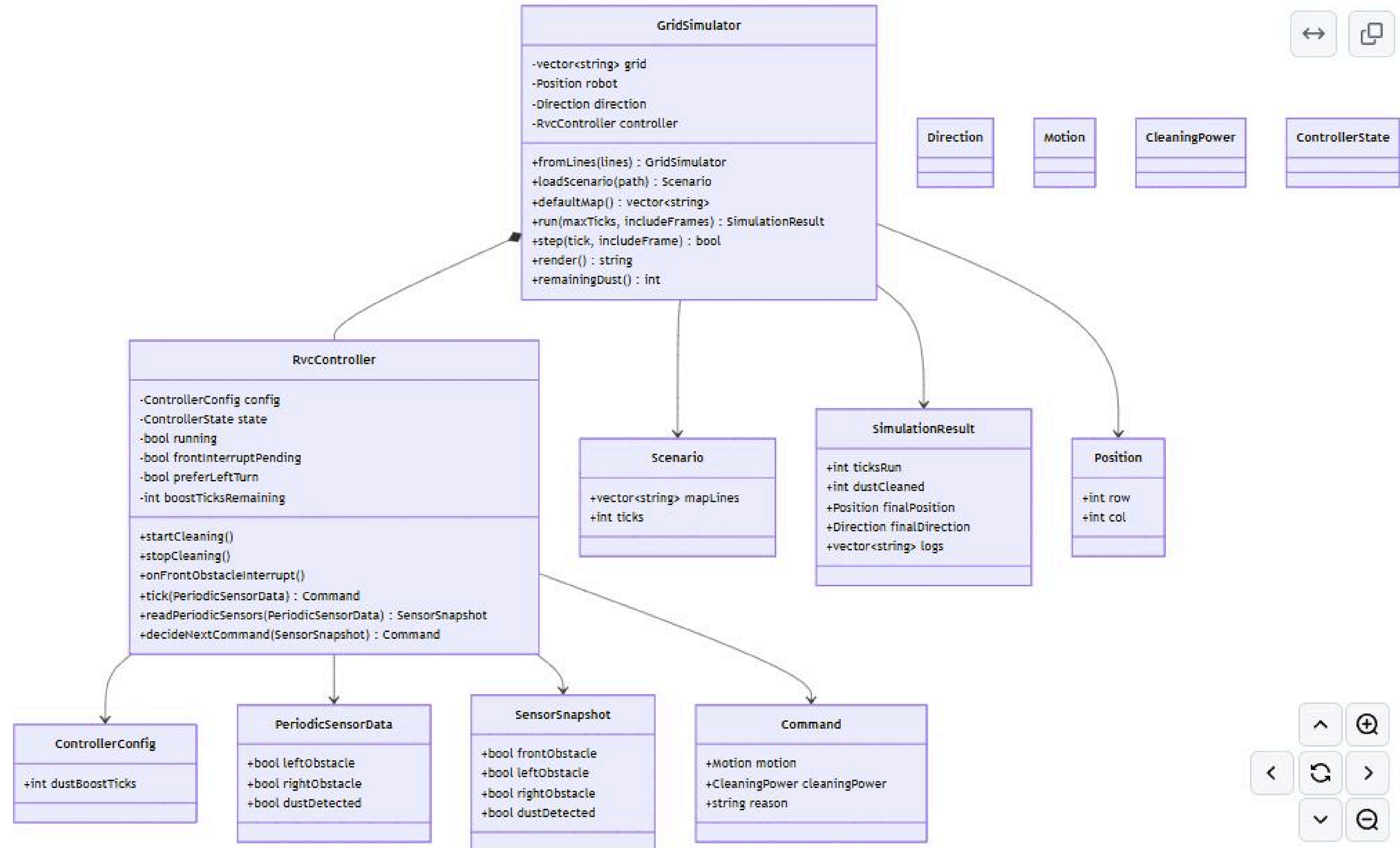
7.1 Structural View

7.1.1 Module Structure

모듈	주요 파일	책임
Core type	<code>include/rvc/Types.hpp</code> , <code>src/Types.cpp</code>	방향, 동작, 청소 세기, 상태, 위치, sensor data, command type을 정의한다.
Controller	<code>include/rvc/RvcController.hpp</code> , <code>src/RvcController.cpp</code>	자동 청소 상태와 sensor snapshot을 기반으로 다음 motion/cleaner command를 결정한다.
Simulator	<code>include/rvc/GridSimulator.hpp</code> , <code>src/GridSimulator.cpp</code>	격자 환경에서 sensor 입력을 생성하고 command를 적용하며 로그와 실행 결과를 만든다.
CLI	<code>src/main.cpp</code>	simulator 실행 옵션을 파싱하고 초기/최종 지도, tick 로그, summary를 출력한다.
Tests	<code>tests/controller_tests.cpp</code> , <code>tests/system_tests.cpp</code>	controller 규칙과 simulator 통합 흐름을 검증한다.

SDD

7.1.2 OOD Class Diagram



SDD

7.1.3 Class Responsibilities

Class	Responsibility
RvcController	문서의 system interface를 구현하고 핵심 제어 규칙에 따라 command를 결정한다.
ControllerConfig	boost duration 같은 제어 정책 값을 제공한다.
PeriodicSensorData	좌측, 우측, 먼지 periodic sensor 값을 전달한다.
SensorSnapshot	pending front interrupt와 periodic sensor 값을 결합한 판단 입력이다.
Command	motor motion과 cleaner power를 함께 표현하는 추상 actuator 명령이다.
GridSimulator	격자 환경에서 sensor/event를 만들고 controller command를 적용한다.
Scenario	시나리오 파일에서 읽은 지도와 기본 tick 수를 담는다.
SimulationResult	시스템 테스트와 CLI 출력에 필요한 실행 결과를 담는다.

SDD

7.2 Interface View

7.2.1 Controller Interface

Operation	Input	Output	책임	관련 요구사 항
<code>RvcController::startCleaning()</code>	없음	없음	자동 청소 실행 상태로 전환하고 전방 interrupt pending 값을 초기화한다.	FR-01, FR-03
<code>RvcController::stopCleaning()</code>	없음	없음	idle 상태로 전환하고 motor stop/cleaner off 상태가 되도록 내부 실행 상태와 boost timer를 초기화한다.	FR-02
<code>RvcController::onFrontObstacleInterrupt()</code>	없음	없음	실행 중일 때 전방 장애물 interrupt를 pending 상태로 기록한다.	FR-04, FR-05
<code>RvcController::tick(const PeriodicSensorData&)</code>	periodic sensor 값	Command	제어 tick마다 sensor 값을 반영하고 다음 command를 반환한다.	FR-06 to FR-15
<code>RvcController::readPeriodicSensors(const PeriodicSensorData&)</code>	periodic sensor 값	SensorSnapshot	pending front interrupt와 periodic 값을 결합한다.	FR-04, FR-06
<code>RvcController::decideNextCommand(const SensorSnapshot&)</code>	sensor snapshot	Command	상태, 장애물, 먼지 정보를 기반으로 motion과 cleaner power를 결정한다.	FR-07 to FR-15

SDD

7.2.2 Simulator Interface

Operation	Input	Output	책임	관련 요구 사항
<code>GridSimulator::loadScenario(const std::filesystem::path&)</code>	scenario path	Scenario	scenario file에서 tick 설정과 map을 읽는다.	FR-16
<code>GridSimulator::run(int, bool)</code>	max tick, frame 포함 여부	SimulationResult	지정 tick 수만큼 simulation을 실행하고 결과를 반환한다.	FR-16, FR-17, FR-18
<code>GridSimulator::step(int, bool)</code>	tick 번호, frame 포함 여부	bool	한 tick의 sensor 생성, controller 호출, command 적용, log 생성을 수행한다.	FR-17, FR-18
<code>GridSimulator::render()</code>	없음	std::string	현재 grid와 robot 방향을 문자열 map으로 렌더링한다.	FR-16

7.2.3 CLI Interface

```
rvc_simulator [--ticks N] [--scenario FILE] [--quiet-map]
```

옵션	설명
<code>--help</code> , <code>-h</code>	사용법을 출력하고 종료한다.
<code>--ticks N</code>	simulation tick 수를 지정한다.
<code>--scenario FILE</code>	scenario file을 읽어 초기 map과 기본 tick 수를 설정한다.
<code>--quiet-map</code>	초기/최종 map과 tick별 frame 출력을 생략하고 log와 summary 중심으로 출력한다.

알 수 없거나 값이 부족한 인자는 오류 메시지와 usage를 출력하고 종료 코드 2를 반환한다. scenario 처리나 실행 중 예외가 발생하면 오류 메시지를 출력하고 종료 코드 1을 반환한다.

SDD

7.3 Data View

7.3.1 Core Data Types

타입	필드/값	설명
ControllerConfig	dustBoostTicks	먼지 감지 후 boost를 유지할 tick 수이다. 기본값은 3이다.
PeriodicSensorData	leftObstacle, rightObstacle, dustDetected	tick마다 sampling되는 좌측/우측/먼지 sensor 값이다.
SensorSnapshot	frontObstacle, leftObstacle, rightObstacle, dustDetected	pending front interrupt와 periodic sensor 값을 결합한 판단 입력이다.
Command	motion, cleaningPower, reason	controller가 actuator 또는 simulator에 전달하는 추상 명령이다.
SimulationResult	ticksRun, dustCleaned, finalPosition, finalDirection, logs	simulator 실행 결과와 검증 로그이다.
Position	row, col	simulator 격자 내 robot 좌표이다.

7.3.2 Enum Values

Enum	값	설명
Direction	North, East, South, West	simulator에서 robot이 바라보는 방향이다.
Motion	None, Stop, Forward, Backward, TurnLeft, TurnRight	motor에 전달할 추상 이동 명령이다.
CleaningPower	Off, Normal, Boost	cleaner 출력 세기이다.
ControllerState	Idle, Cleaning, Avoiding, Escaping	controller의 현재 제어 상태이다.

SDD

7.3.3 Scenario Data

scenario file은 선택적 `ticks=N` 설정과 `map:` 섹션으로 구성된다. map 문자는 다음 의미를 갖는다.

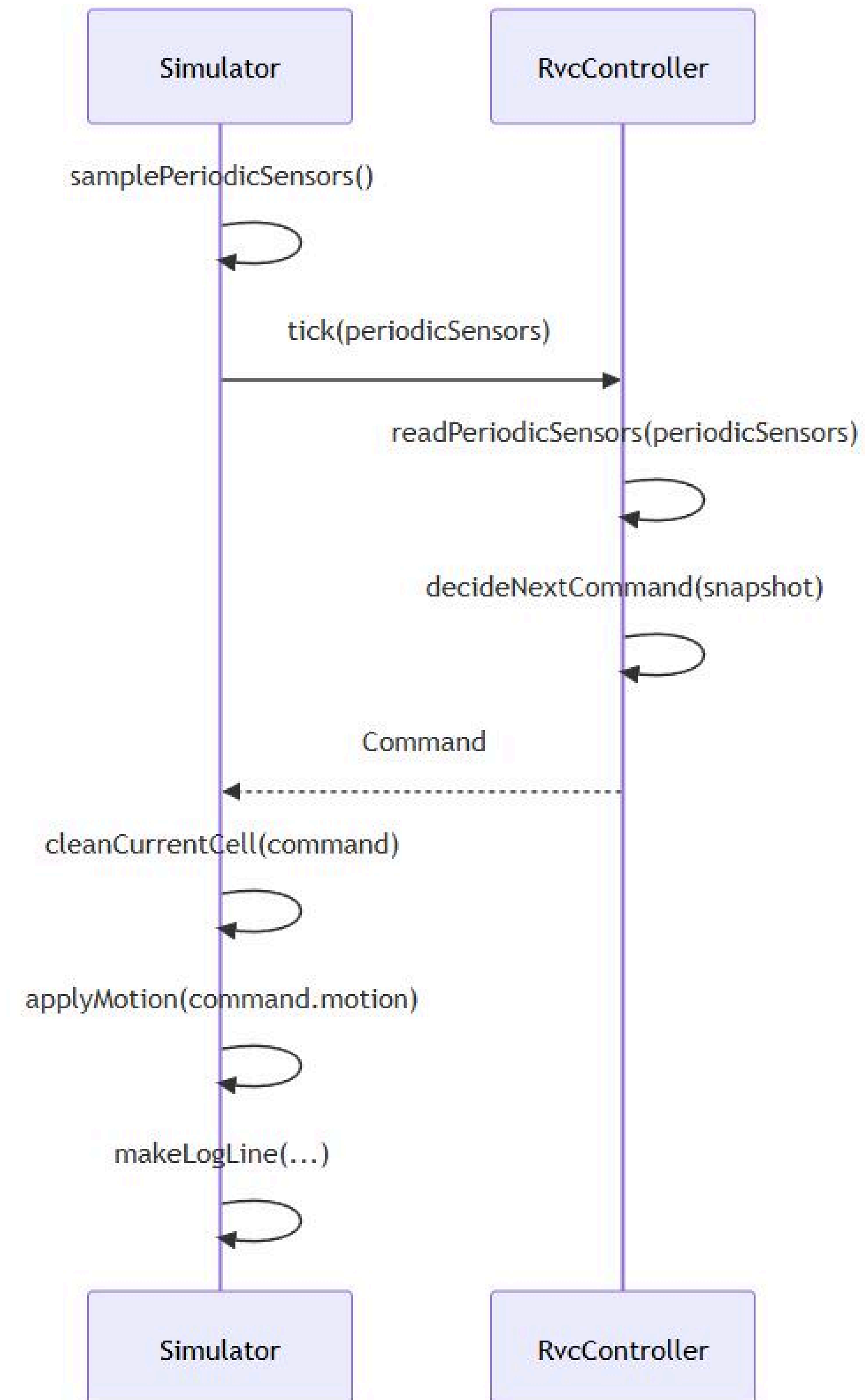
문자	의미
#	장애물
.	빈 칸
*	먼지
R, ^	북쪽을 바라보는 robot
>	동쪽을 바라보는 robot
v	남쪽을 바라보는 robot
<	서쪽을 바라보는 robot

map에는 robot marker가 정확히 하나 있어야 한다. simulator는 robot marker를 읽은 뒤 내부 grid에서는 해당 칸을 빈 칸으로 보관하고, render 시 현재 방향 marker를 다시 표시한다.

SDD

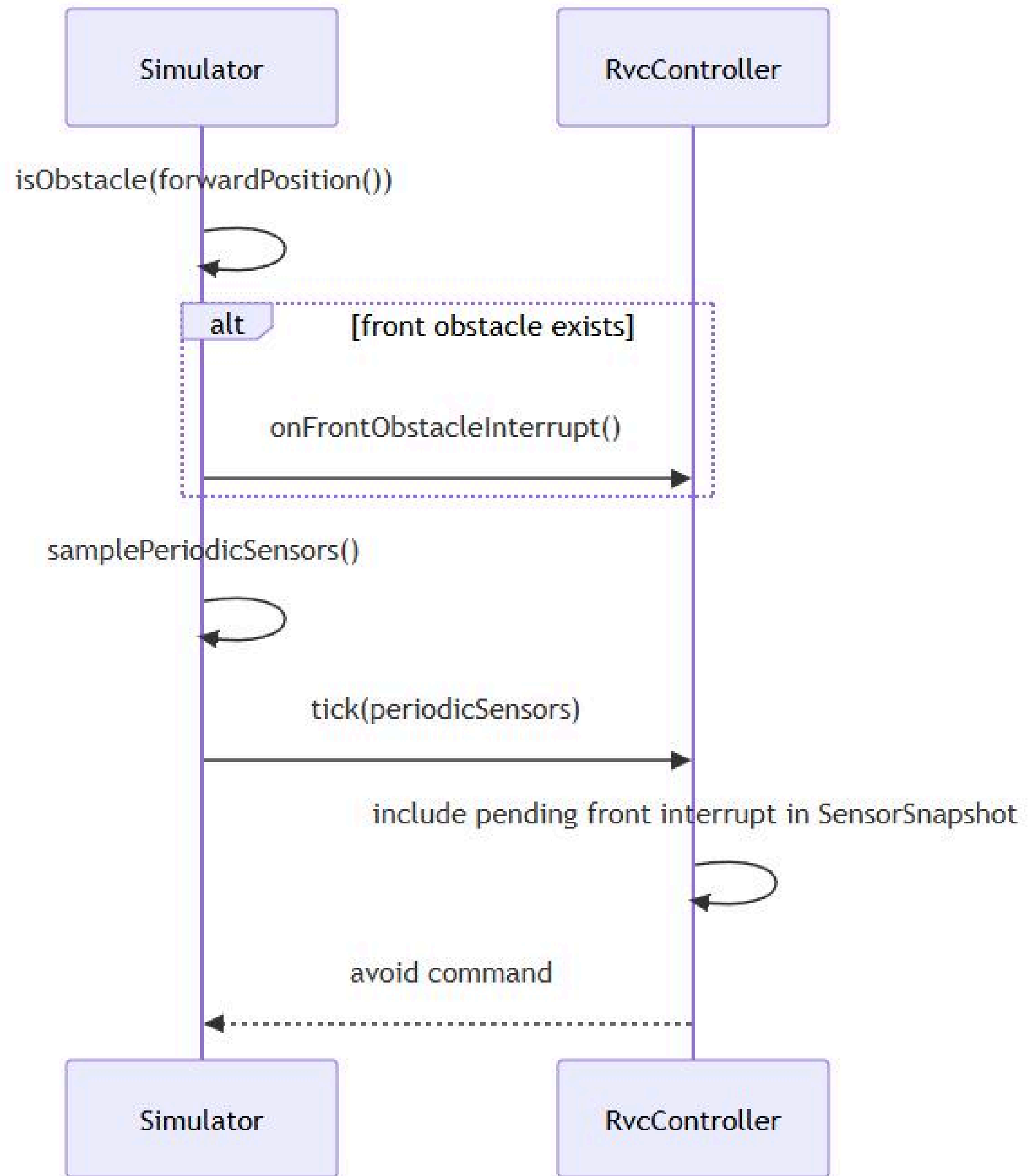
7.4 Interaction View

7.4.1 SD-01 Control Tick Loop



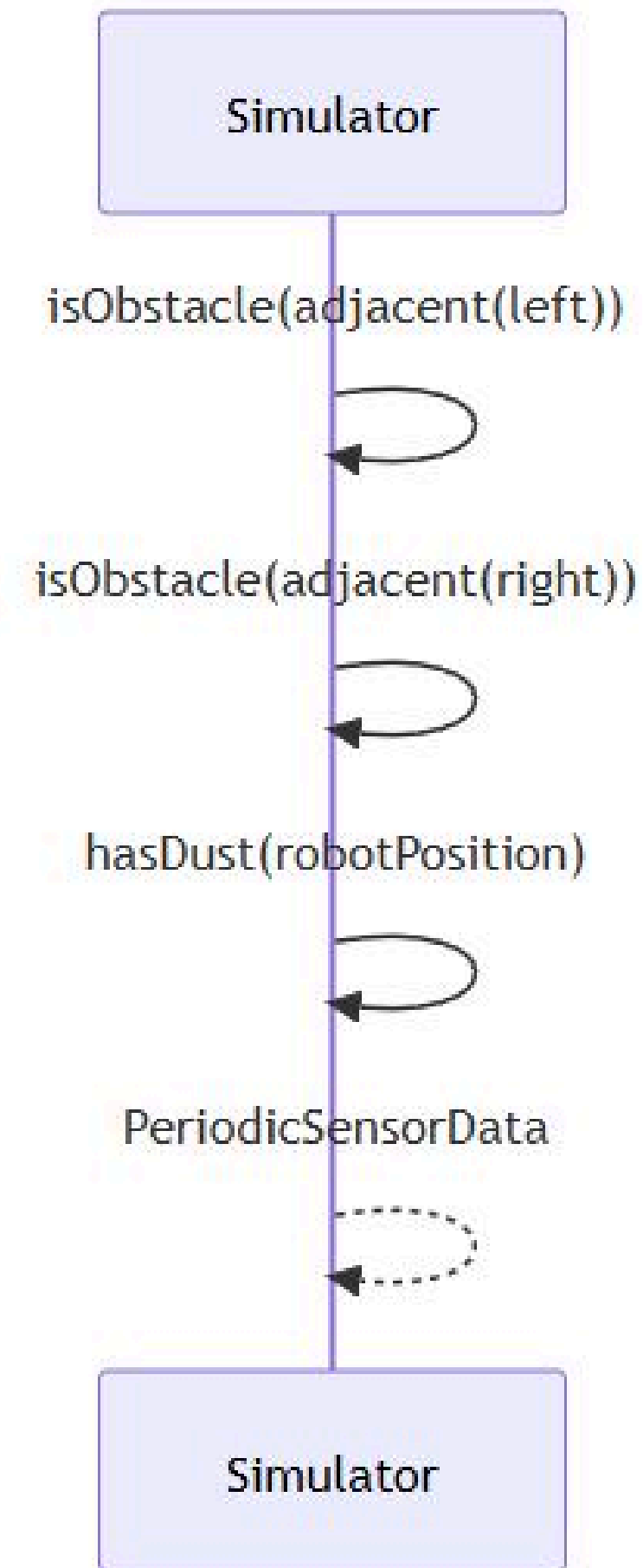
SDD

7.4.2 SD-02 Front Interrupt Handling



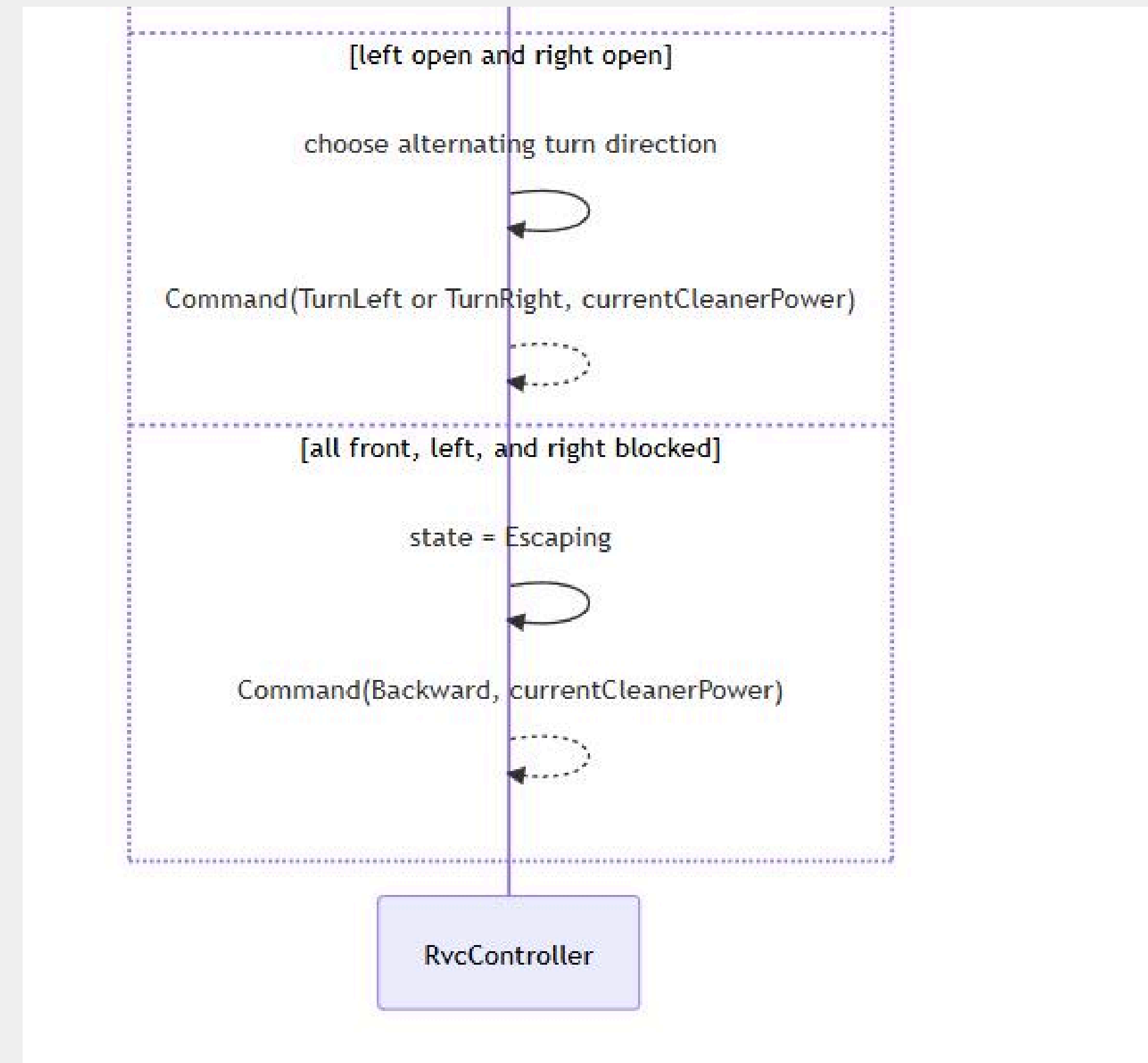
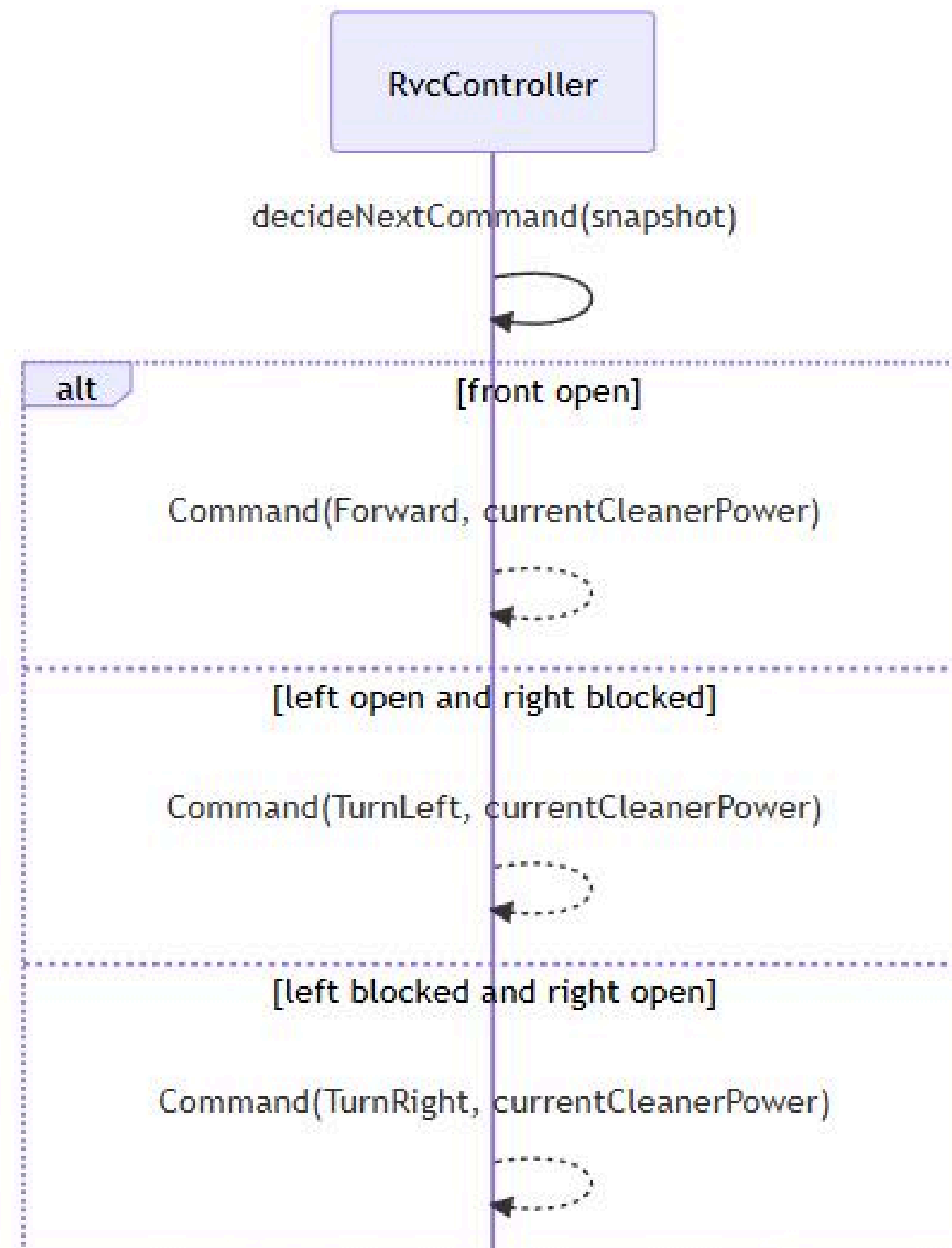
SDD

7.4.3 SD-03 Periodic Sensor Sampling



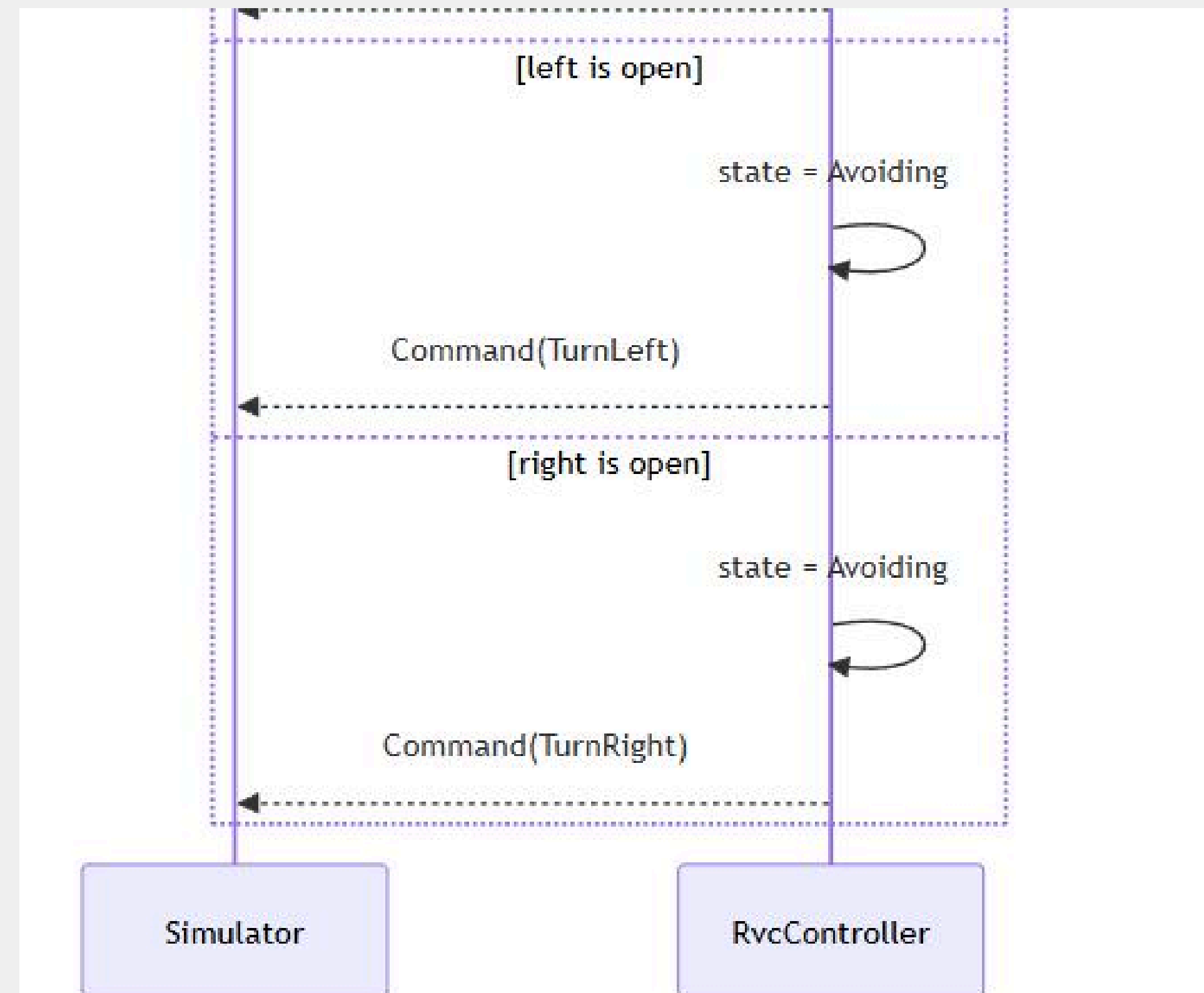
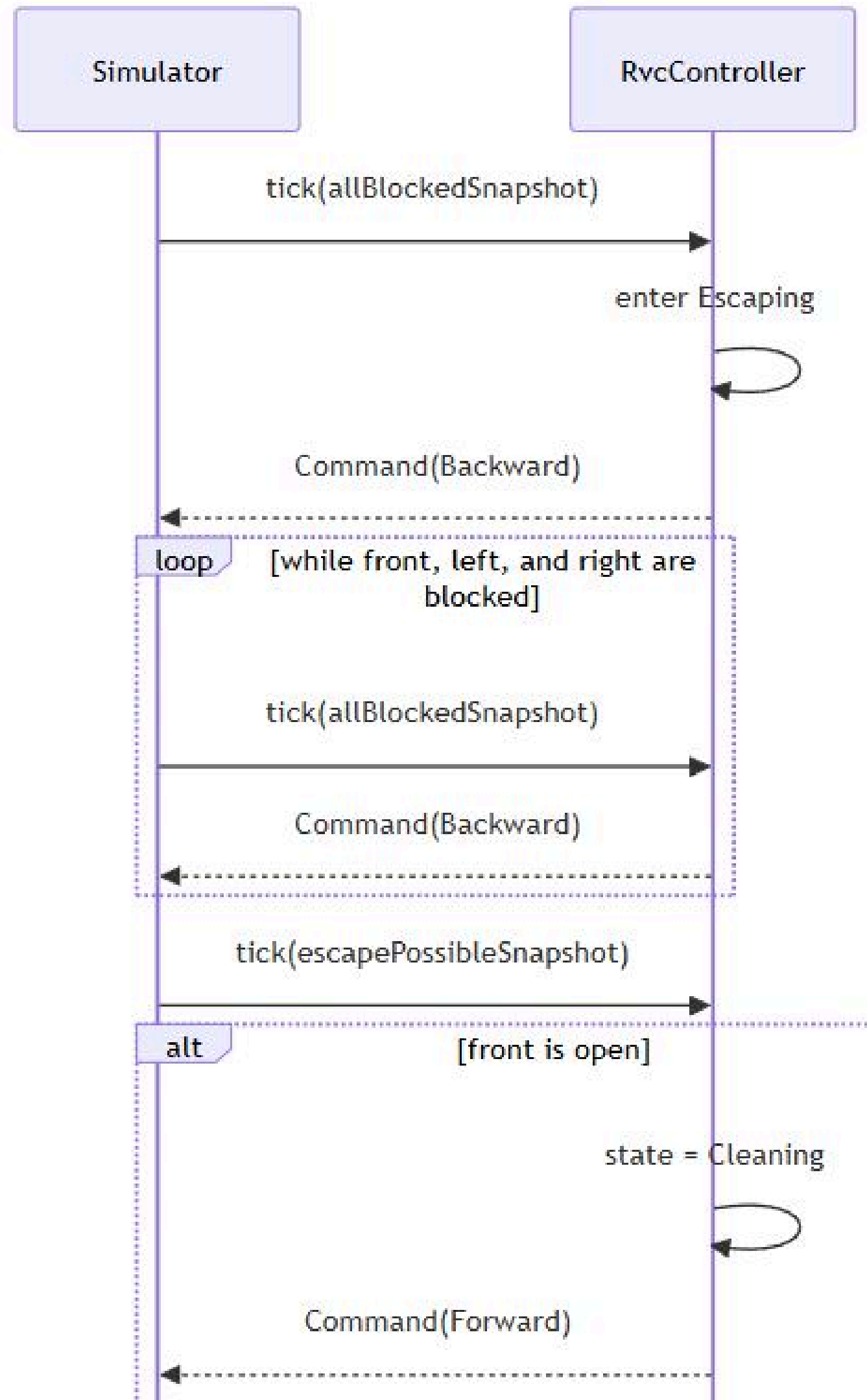
SDD

7.4.4 SD-04 Obstacle Avoidance



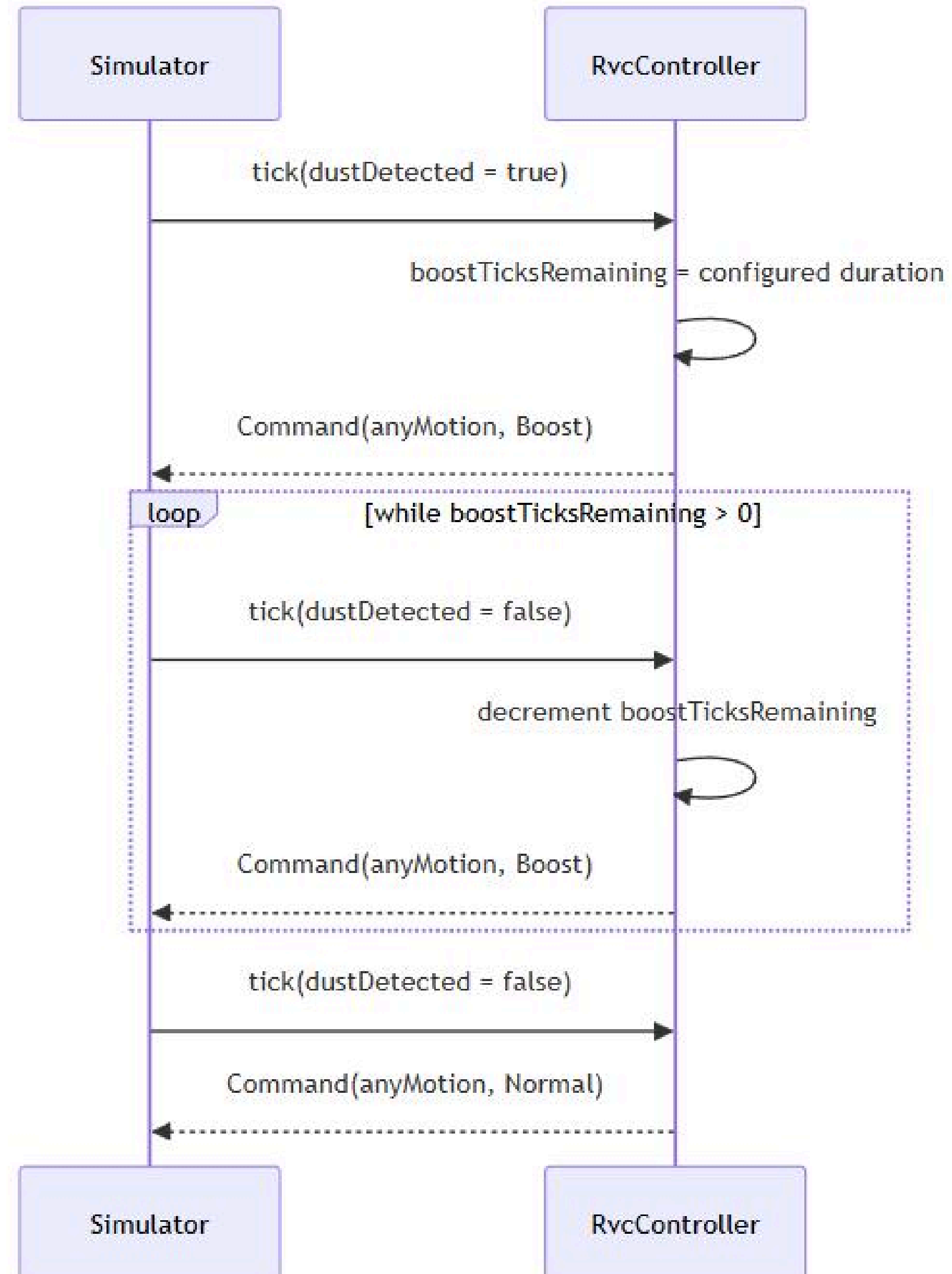
SDD

7.4.5 SD-05 Escape Until Possible



SDD

7.4.6 SD-06 Dust Boost



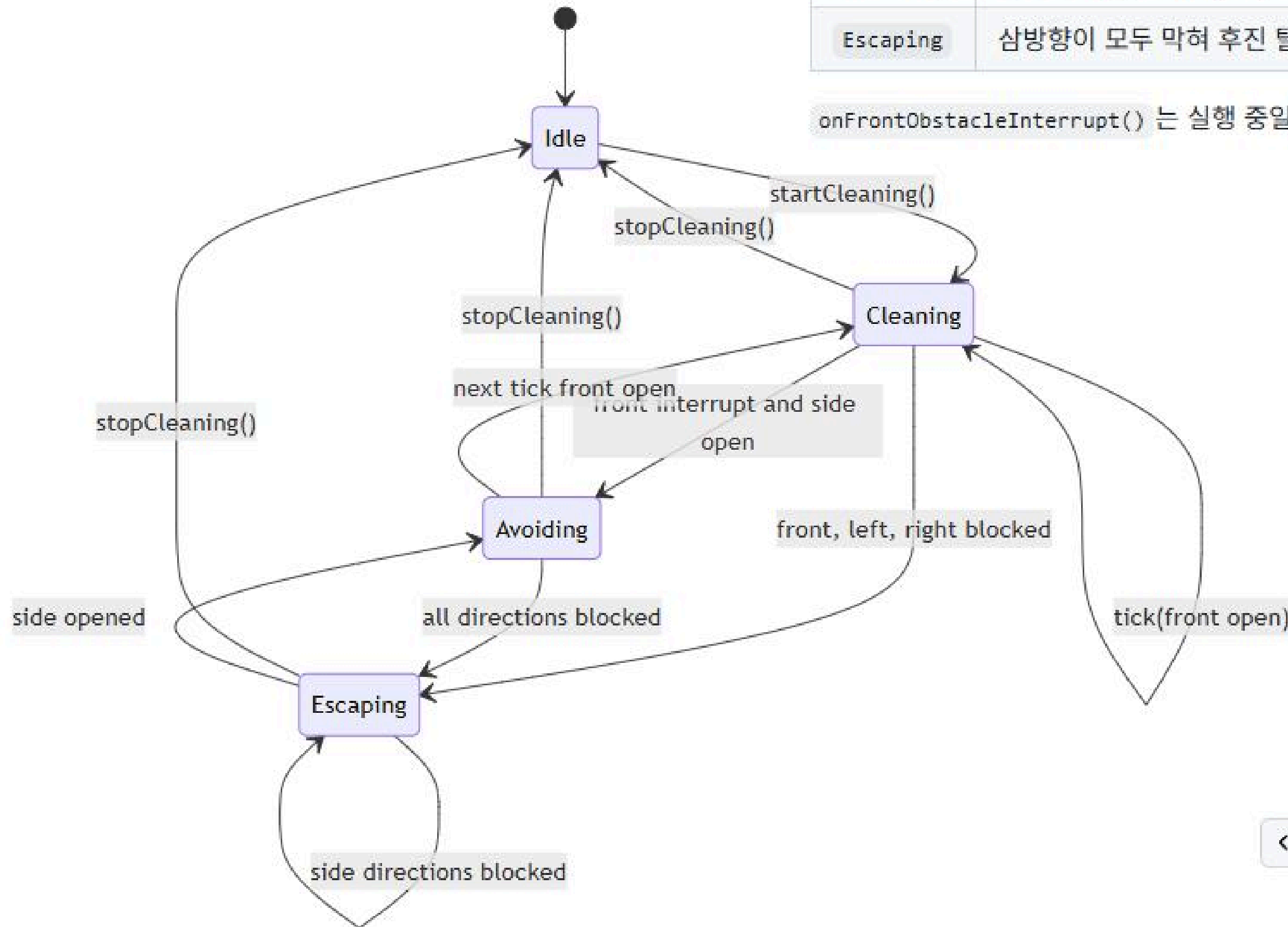
SDD

7.5.2 Controller States

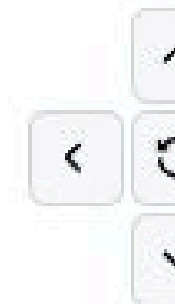
상태	의미	주요 출력
Idle	자동 청소가 실행되지 않는 상태	Stop, Off
Cleaning	기본 자동 청소 상태	전방 open 시 Forward
Avoiding	전방 장애물 interrupt 후 측면 회피를 수행하는 상태	TurnLeft 또는 TurnRight
Escaping	삼방향의 모든 방향이 막혀 후진 탈출을 수행하는 상태	Backward 반복

7.5 State View

7.5.1 Controller State Transition



onFrontObstacleInterrupt() 는 실행 중일 때만 pending interrupt를 기록한다. idle 상태의 interrupt는 다음 tick 판단에 영향을 주지 않는다.



SDD

7.6 Algorithm View

7.6.1 Tick Processing

1. `running_` 이 `false`이면 `Motion::Stop`, `CleaningPower::Off` command를 반환한다.
2. `readPeriodicSensors()` 가 `frontInterruptPending_` 와 `PeriodicSensorData` 를 결합해 `SensorSnapshot` 을 만든다.
3. `decideNextCommand()` 가 snapshot과 현재 state를 기반으로 command를 결정한다.
4. `tick()` 은 command 결정 후 `frontInterruptPending_` 를 `false`로 되돌려 interrupt를 소비한다.

7.6.2 Obstacle Avoidance

조건	상태 변화	Motion
전방이 열림	<code>Cleaning</code>	<code>Forward</code>
전방이 막히고 좌측만 열림	<code>Avoiding</code>	<code>TurnLeft</code>
전방이 막히고 우측만 열림	<code>Avoiding</code>	<code>TurnRight</code>
전방이 막히고 좌/우 모두 열림	<code>Avoiding</code>	<code>TurnLeft</code> 와 <code>TurnRight</code> 를 번갈아 선택
전방, 좌측, 우측 모두 막힘	<code>Escaping</code>	<code>Backward</code>

7.6.3 Escape Rule

`Escaping` 상태에서는 좌측과 우측이 모두 막힌 동안 계속 `Backward` command를 반환한다. 좌측 또는 우측 중 하나 이상이 열리면 탈출 가능 상태로 판단한다. 탈출 가능 상태가 되면 `Avoiding` 으로 전환해 열린 측면 방향으로 회전한다.

7.6.4 Dust Boost Rule

`updateCleaningPower(bool dustDetected)` 는 motion 판단과 독립적으로 cleaner power 후보를 결정한다. 먼지가 감지되면 `boostTicksRemaining_` 을 `config_.dustBoostTicks` 로 재설정한다. 먼지가 감지되지 않고 남은 boost tick이 있으면 1 감소시킨다. 남은 boost tick이 0보다 크면 `Boost`, 아니면 `Normal` 을 반환한다. 최종 `Command` 생성 시에는 `Forward` 동작에서만 해당 power를 출력하고, 회피 회전, 후진, 정지 동작에서는 cleaner power를 `Off` 로 내보낸다.

SDD

7.7 Error Handling View

상황	처리
빈 map으로 simulator 생성	<code>std::invalid_argument</code> 발생
robot marker 없음	<code>std::invalid_argument</code> 발생
robot marker가 둘 이상	<code>std::invalid_argument</code> 발생
scenario file open 실패	<code>std::runtime_error</code> 발생
scenario file에 map section 없음	<code>std::runtime_error</code> 발생
<code>run()</code> 에 음수 tick 입력	<code>std::invalid_argument</code> 발생
grid 범위 밖 좌표	장애물로 간주하여 이동하지 않는다.
알 수 없는 CLI option	usage를 출력하고 종료 코드 2를 반환한다.
실행 중 scenario 예외	오류 메시지를 출력하고 종료 코드 1을 반환한다.

7.8 Verification View

7.8.1 Test Strategy

테스트 수준	대상	검증 내용
Controller unit test	<code>RvcController</code>	전진, 중지, interrupt 회피, 좌/우 회전 선택, 교대 회전, 탈출 유지와 측면 탈출, dust boost duration, 회피 중 cleaner off
Simulator system test	<code>GridSimulator</code>	실제 격자에서 dust 청소, 후진 탈출, boxed-in 반복 후진, 측면 탈출구 확인, boost 중 cleaner off, front interrupt 후 회전
CLI CTest	<code>rvc_simulator</code>	기본 실행과 scenario 기반 실행 가능 여부

SDD

7.8.2 Requirement Traceability

요구 사항	핵심 설계 요소	검증
FR-01	<code>RvcController::startCleaning</code> , <code>ControllerState::Cleaning</code>	<code>ControllerMovesForwardWhenPathIsClear</code>
FR-02	<code>RvcController::stopCleaning</code> , <code>idle</code> command	<code>StopCleaningReturnsStopAndOff</code>
FR-03	<code>decideNextCommand</code> , <code>Motion::Forward</code>	<code>ControllerMovesForwardWhenPathIsClear</code>
FR-04	<code>onFrontObstacleInterrupt</code> , <code>frontInterruptPending_</code>	<code>FrontInterruptTriggersImmediateAvoidance</code> , <code>SimulatorTurnsAfterFrontInterrupt</code>
FR-05	<code>tick</code> , <code>SensorSnapshot::frontObstacle</code> , 회피 command	<code>FrontInterruptTriggersImmediateAvoidance</code> , <code>SimulatorTurnsAfterFrontInterrupt</code>
FR-06	<code>PeriodicSensorData</code> , <code>readPeriodicSensors</code>	controller unit tests
FR-07	<code>chooseOpenSideTurn</code> , <code>Motion::TurnLeft</code>	<code>FrontInterruptTriggersImmediateAvoidance</code>
FR-08	<code>chooseOpenSideTurn</code> , <code>Motion::TurnRight</code>	<code>TurnsTowardOpenSide</code> , <code>SimulatorTurnsAfterFrontInterrupt</code>
FR-09	<code>preferLeftTurn_</code> , 좌우 교대 정책	<code>AlternatesWhenBothSidesAreOpen</code>
FR-10	all-blocked 판단, <code>ControllerState::Escaping</code>	<code>AllBlockedEntersEscapingAndKeepsBackingUp</code> , <code>SimulatorUsesBackwardEscape</code> , <code>SimulatorKeepsCleanerOffDuringBoostedEscape</code>

SDD

8. Design Rationale

8.1 Major Design Decisions

결정	근거
전방 장애물은 <code>onFrontObstacleInterrupt()</code> 로만 controller에 전달한다.	요구사항에서 front sensor가 interrupt 방식으로 동작해야 하기 때문이다.
좌/우/면지 값은 <code>tick(PeriodicSensorData)</code> 호출마다 controller에 전달한다.	periodic sensor sampling 요구사항을 controller API에 직접 반영하기 위해서이다.
<code>readPeriodicSensors()</code> 는 pending front interrupt와 periodic 값을 결합하여 <code>SensorSnapshot</code> 을 만든다.	서로 다른 입력 timing을 단일 판단 입력으로 정리하기 위해서이다.
<code>decideNextCommand()</code> 는 단일 판단 지점으로 둔다.	회피, 탈출, boost 판단을 단위 테스트하기 쉽게 만들기 위해서이다.
<code>Escaping</code> 상태에서 삼방향이 계속 막혀 있으면 반드시 <code>Backward</code> command를 반복한다.	FR-11의 탈출 가능 시점까지 후진 유지 요구사항을 직접 만족하기 위해서이다.
<code>GridSimulator</code> 는 controller command를 실제 하드웨어 대신 격자 상태에 적용한다.	실제 하드웨어 없이도 요구사항을 반복 검증하기 위해서이다.

8.2 SOLID Analysis

Principle	Application
SRP	<code>RvcController</code> 는 제어 결정만 담당하고, <code>GridSimulator</code> 는 환경과 이동 적용만 담당한다.
OCP	sensor 입력은 <code>PeriodicSensorData</code> 와 interrupt API로 추상화되어 새 sensor 추가 시 controller 확장이 가능하다.
LSP	simulator와 실제 하드웨어 어댑터는 같은 <code>Command</code> 의미를 따르므로 대체 가능하다.
ISP	controller의 public interface는 시작, 중지, interrupt, tick, 판단에 필요한 작은 operation으로 분리된다.
DIP	controller는 concrete simulator나 hardware에 의존하지 않고 값 객체와 추상 command에만 의존한다.

SDD

8.3 Constraint Satisfaction

제약	충족 방식
C++20 및 CMake 기반 빌드	<code>CMakeLists.txt</code> 에서 <code>CMAKE_CXX_STANDARD 20</code> 과 <code>rvc_core</code> , <code>rvc_simulator</code> , <code>rvc_tests</code> target을 정의한다.
UTF-8 문서	모든 Markdown 문서는 UTF-8로 작성한다.
하드웨어 독립성	controller가 concrete hardware에 의존하지 않고 sensor data와 command value만 다룬다.
반복 가능한 테스트	controller 입력을 값 객체로 제공하고 random 또는 wall-clock 의존성을 두지 않는다.
요구사항 추적성	FR별 설계 요소와 테스트 이름을 SRS, SDD, traceability 문서에 유지한다.

CODE

RVC Controller

```
1  #pragma once
2
3  #include "rvc/Types.hpp"
4
5  namespace rvc {
6
7  struct ControllerConfig {
8      int dustBoostTicks{3};
9  };
10
11 class RvcController {
12 public:
13     explicit RvcController(ControllerConfig config = {});
14
15     void startCleaning();
16     void stopCleaning();
17     void onFrontObstacleInterrupt();
18
19     [[nodiscard]] Command tick(const PeriodicSensorData& periodicSensors);
20     [[nodiscard]] SensorSnapshot readPeriodicSensors(const PeriodicSensorData& periodicSensors) const;
21     [[nodiscard]] Command decideNextCommand(const SensorSnapshot& snapshot);
22
23     [[nodiscard]] ControllerState state() const;
24     [[nodiscard]] bool isRunning() const;
25     [[nodiscard]] int boostTicksRemaining() const;
26
27 private:
28     [[nodiscard]] CleaningPower updateCleaningPower(bool dustDetected);
29     [[nodiscard]] Motion chooseOpenSideTurn(bool leftObstacle, bool rightObstacle);
30     [[nodiscard]] Command makeCommand(Motion motion, CleaningPower power, std::string reason) const;
31
32     ControllerConfig config_;
33     ControllerState state_{ControllerState::Idle};
34     bool running_{false};
35     bool frontInterruptPending_{false};
36     bool preferLeftTurn_{true};
37     int boostTicksRemaining_{0};
38 };
39
```

```
Command RvcController::decideNextCommand(const SensorSnapshot& snapshot) {
    if (!running_) {
        return makeCommand(Motion::Stop, CleaningPower::Off, "controller idle");
    }

    const CleaningPower power = updateCleaningPower(snapshot.dustDetected);
    const bool sidesBlocked = snapshot.leftObstacle && snapshot.rightObstacle;
    const bool allBlocked = snapshot.frontObstacle && sidesBlocked;

    if (state_ == ControllerState::Escaping) {
        if (sidesBlocked) {
            return makeCommand(Motion::Backward, power, "escaping: keep backing up until a side exit opens");
        }

        state_ = ControllerState::Avoiding;
        return makeCommand(chooseOpenSideTurn(snapshot.leftObstacle, snapshot.rightObstacle), power,
            "escaping: side opened, turn toward exit");
    }

    if (!snapshot.frontObstacle) {
        state_ = ControllerState::Cleaning;
        return makeCommand(Motion::Forward, power, "front clear: forward cleaning");
    }

    if (allBlocked) {
        state_ = ControllerState::Escaping;
        return makeCommand(Motion::Backward, power, "front interrupt and both sides blocked: enter escaping");
    }

    state_ = ControllerState::Avoiding;
    return makeCommand(chooseOpenSideTurn(snapshot.leftObstacle, snapshot.rightObstacle), power,
        "front interrupt: stop forward motion and turn");
}
```

RVC Controller

```
void RvcController::startCleaning() {
    running_ = true;
    state_ = ControllerState::Cleaning;
    frontInterruptPending_ = false;
}

void RvcController::stopCleaning() {
    running_ = false;
    state_ = ControllerState::Idle;
    frontInterruptPending_ = false;
    boostTicksRemaining_ = 0;
}

void RvcController::onFrontObstacleInterrupt() {
    if (running_) {
        frontInterruptPending_ = true;
    }
}

Command RvcController::tick(const PeriodicSensorData& periodicSensors) {
    if (!running_) {
        return makeCommand(Motion::Stop, CleaningPower::Off, "controller idle");
    }

    const SensorSnapshot snapshot = readPeriodicSensors(periodicSensors);
    Command command = decideNextCommand(snapshot);
    frontInterruptPending_ = false;
    return command;
}

SensorSnapshot RvcController::readPeriodicSensors(const PeriodicSensorData& periodicSensors) const {
    return SensorSnapshot{
        .frontObstacle = frontInterruptPending_,
        .leftObstacle = periodicSensors.leftObstacle,
        .rightObstacle = periodicSensors.rightObstacle,
        .dustDetected = periodicSensors.dustDetected,
    };
}
```

```
94  CleaningPower RvcController::updateCleaningPower(bool dustDetected) {
95      if (dustDetected) {
96          boostTicksRemaining_ = config_.dustBoostTicks;
97      } else if (boostTicksRemaining_ > 0) {
98          --boostTicksRemaining_;
99      }
100
101      return boostTicksRemaining_ > 0 ? CleaningPower::Boost : CleaningPower::Normal;
102  }
103
104  Motion RvcController::chooseOpenSideTurn(bool leftObstacle, bool rightObstacle) {
105      if (!leftObstacle && rightObstacle) {
106          return Motion::TurnLeft;
107      }
108
109      if (leftObstacle && !rightObstacle) {
110          return Motion::TurnRight;
111      }
112
113      if (!leftObstacle && !rightObstacle) {
114          const Motion selected = preferLeftTurn_ ? Motion::TurnLeft : Motion::TurnRight;
115          preferLeftTurn_ = !preferLeftTurn_;
116          return selected;
117      }
118
119      return Motion::Backward;
120  }
121
122  Command RvcController::makeCommand(Motion motion, CleaningPower power, std::string reason) const {
123      const CleaningPower outputPower = motion == Motion::Forward ? power : CleaningPower::Off;
124
125      return Command{
126          .motion = motion,
127          .cleaningPower = outputPower,
128          .reason = std::move(reason),
129      };
130  }
131
132  } // namespace rvc
```

Others

```
43  ✓ struct PeriodicSensorData {
44      bool leftObstacle{false};
45      bool rightObstacle{false};
46      bool dustDetected{false};
47  };
48
49  ✓ struct SensorSnapshot {
50      bool frontObstacle{false};
51      bool leftObstacle{false};
52      bool rightObstacle{false};
53      bool dustDetected{false};
54  };
55
56  ✓ struct Command {
57      Motion motion{Motion::None};
58      CleaningPower cleaningPower{CleaningPower::Off};
59      std::string reason;
60  };
61
62  [[nodiscard]] Direction turnLeft(Direction direction);
63  [[nodiscard]] Direction turnRight(Direction direction);
64  [[nodiscard]] Direction opposite(Direction direction);
65  [[nodiscard]] std::string toString(Direction direction);
66  [[nodiscard]] std::string toString(Motion motion);
67  [[nodiscard]] std::string toString(CleaningPower power);
68  [[nodiscard]] std::string toString(ControllerState state);
69
70  } // namespace rvc
```

TEST

Test

RVC 테스트 결과

1. 테스트 실행 요약

항목	값
실행일	2026-05-21
실행 시각	04:41:04 +09:00
빌드 갱신 명령	<code>cmake --build build --config Debug</code>
테스트 실행 명령	<code>ctest --test-dir build -C Debug --output-on-failure</code>
빌드 설정	Debug
테스트 프레임워크	CTest, GoogleTest
전체 결과	57 passed / 0 failed
총 실행 시간	0.94 sec

2. 결과 요약

구분	범위	결과
컨트롤러 유닛 테스트	17건	17 passed / 0 failed
타입 유닛 테스트	4건	4 passed / 0 failed
시뮬레이터 시스템 테스트	9건	9 passed / 0 failed
시나리오 파일 테스트	5건	5 passed / 0 failed
시나리오 회귀 테스트	20건	20 passed / 0 failed
CLI 테스트	2건	2 passed / 0 failed

Unit Test

3. 컨트롤러 유닛 테스트 결과

컨트롤러 유닛 테스트는 결정적인 센서 입력과 인터럽트 입력을 사용해 `RvcController` 의 상태 전이, 회피 판단, 탈출 판단, 먼지 boost 제어 규칙을 검증한다.

번호	테스트 케이스	검증 대상	결과
1	<code>RvcControllerTest.ControllerMovesForwardWhenPathIsClear</code>	전방 경로가 열려 있을 때 청소를 시작하고 전진하는지 검증	Passed
2	<code>RvcControllerTest.IdleControllerReturnsStopAndIgnoresFrontInterrupt</code>	idle 상태에서 전방 인터럽트와 먼지 감지를 무시하고 정지/꺼짐 상태를 유지하는지 검증	Passed
3	<code>RvcControllerTest.StopCleaningReturnsStopAndOff</code>	청소 중지 요청 시 모터 정지와 클리너 끄기 명령을 반환하는지 검증	Passed
4	<code>RvcControllerTest.FrontInterruptTriggersImmediateAvoidance</code>	전방 장애물 인터럽트가 즉시 회피 동작으로 이어지고 클리너가 꺼지는지 검증	Passed
5	<code>RvcControllerTest.FrontInterruptIsConsumedAfterOneTick</code>	전방 인터럽트가 1 tick 후 소비되어 정상 전진으로 복귀하는지 검증	Passed
6	<code>RvcControllerTest.TurnsTowardOpenSide</code>	열린 측면 방향으로 회전하고 회피 중 클리너가 꺼지는지 검증	Passed
7	<code>RvcControllerTest.AlternatesWhenBothSidesAreOpen</code>	좌우가 모두 열려 있을 때 회전 방향을 번갈아 선택하는지 검증	Passed
8	<code>RvcControllerTest.AlternationPersistsAcrossClearTicks</code>	정상 전진 tick을 거쳐도 좌우 회전 교대 순서가 유지되는지 검증	Passed
9	<code>RvcControllerTest.AllBlockedEntersEscapingAndKeepsBackingUp</code>	전방, 좌측, 우측이 모두 막힌 경우 탈출 상태로 진입하고 후진을 유지하는지 검증	Passed

10	<code>RvcControllerTest.AllBlockedWithDustKeepsCleanerOffButPreservesBoostBudget</code>	막힘과 먼지 감지가 동시에 발생해도 클리너를 끄고 boost 예산을 보존하는지 검증	Passed
11	<code>RvcControllerTest.EscapingIgnoresOpenFrontUntilSideOpens</code>	탈출 중 전방이 열려도 측면 탈출구가 열릴 때까지 후진하는지 검증	Passed
12	<code>RvcControllerTest.EscapingTurnsTowardRightSideExit</code>	탈출 중 우측 탈출구가 열리면 우회전하는지 검증	Passed
13	<code>RvcControllerTest.EscapingUsesAlternationWhenBothSidesOpen</code>	탈출 중 양측이 모두 열렸을 때 회전 방향 교대 규칙을 적용하는지 검증	Passed
14	<code>RvcControllerTest.DustBoostLastsConfiguredTicks</code>	먼지 감지 후 설정된 tick 동안 boost 청소가 유지되는지 검증	Passed
15	<code>RvcControllerTest.DustDetectionRefreshesBoostBudget</code>	boost 중 먼지를 다시 감지하면 boost 잔여 tick이 갱신되는지 검증	Passed
16	<code>RvcControllerTest.ZeroTickBoostConfigurationDoesNotBoostOnDust</code>	boost tick이 0인 설정에서 먼지 감지 시에도 normal 청소를 유지하는지 검증	Passed
17	<code>RvcControllerTest.AvoidanceOutputStaysOffWhileBoostStateIsMaintained</code>	boost 상태가 남아 있어도 회피 중에는 클리너가 꺼지고 전진 복귀 후 boost가 이어지는지 검증	Passed

4. 타입 유닛 테스트 결과

타입 유닛 테스트는 방향 전환과 로그용 문자열 변환처럼 다른 모듈이 공유하는 기본 타입 연산의 안정성을 검증한다.

번호	테스트 케이스	검증 대상	결과
1	<code>RvcTypesTest.TurnLeftCyclesThroughAllDirections</code>	좌회전 연산이 네 방향을 올바른 순서로 순환하는지 검증	Passed
2	<code>RvcTypesTest.TurnRightCyclesThroughAllDirections</code>	우회전 연산이 네 방향을 올바른 순서로 순환하는지 검증	Passed
3	<code>RvcTypesTest.OppositeMatchesTwoRightTurns</code>	반대 방향 계산이 우회전 두 번과 일치하는지 검증	Passed
4	<code>RvcTypesTest.StringConversionsExposeStableLogNames</code>	방향, 이동 명령, 청소 세기, 컨트롤러 상태의 로그 문자열이 안정적 인지 검증	Passed

타입 유닛 테스트 소계: 4 passed / 0 failed.

Unit Test

컨트롤러 유닛 테스트 결과

✓ RvcControllerTest (17)	< 1ms
✓ AllBlockedEntersEscapingAndKeepsBackingUp	< 1ms GoogleTes...
✓ AllBlockedWithDustKeepsCleanerOffButPreservesBoostBudget	< 1ms GoogleTes...
✓ AlternatesWhenBothSidesAreOpen	< 1ms GoogleTes...
✓ AlternationPersistsAcrossClearTicks	< 1ms GoogleTes...
✓ AvoidanceOutputStaysOffWhileBoostStateIsMaintained	< 1ms GoogleTes...
✓ ControllerMovesForwardWhenPathIsClear	< 1ms GoogleTes...
✓ DustBoostLastsConfiguredTicks	< 1ms GoogleTes...
✓ DustDetectionRefreshesBoostBudget	< 1ms GoogleTes...
✓ EscapingIgnoresOpenFrontUntilSideOpens	< 1ms GoogleTes...
✓ EscapingTurnsTowardRightSideExit	< 1ms GoogleTes...
✓ EscapingUsesAlternationWhenBothSidesOpen	< 1ms GoogleTes...
✓ FrontInterruptIsConsumedAfterOneTick	< 1ms GoogleTes...
✓ FrontInterruptTriggersImmediateAvoidance	< 1ms GoogleTes...
✓ IdleControllerReturnsStopAndIgnoresFrontInterrupt	< 1ms GoogleTes...
✓ StopCleaningReturnsStopAndOff	< 1ms GoogleTes...
✓ TurnsTowardOpenSide	< 1ms GoogleTes...
✓ ZeroTickBoostConfigurationDoesNotBoostOnDust	< 1ms GoogleTes...

타입 유닛 테스트 결과

✓ RvcTypesTest (4)	< 1ms
✓ OppositeMatchesTwoRightTurns	< 1ms GoogleTes...
✓ StringConversionsExposeStableLogNames	< 1ms GoogleTes...
✓ TurnLeftCyclesThroughAllDirections	< 1ms GoogleTes...
✓ TurnRightCyclesThroughAllDirections	< 1ms GoogleTes...

System Test

시뮬레이터 자체에 대한 테스트

5. 시뮬레이터 시스템 테스트 결과

시뮬레이터 시스템 테스트는 `GridSimulator`, `RvcController`, 맵 렌더링, 로그, 예외 처리가 통합 흐름에서 올바르게 동작하는지 검증한다.

번호	테스트 케이스	검증 대상	결과
1	<code>RvcSystemTest.SimulatorCleansDustAndLogsCommands</code>	시뮬레이터가 먼지를 청소하고 먼지 감지 및 boost 명령 로그를 남기는지 검증	Passed
2	<code>RvcSystemTest.SimulatorUsesBackwardEscape</code>	막힌 맵에서 후진 탈출 동작과 클리너 off가 적용되는지 검증	Passed
3	<code>RvcSystemTest.SimulatorKeepsCommandingBackwardWhenBoxedIn</code>	로봇이 완전히 갇힌 동안 계속 후진 명령을 내리는지 검증	Passed
4	<code>RvcSystemTest.SimulatorKeepsBackingUpUntilSideExitOpens</code>	탈출 중 측면 탈출구가 열릴 때까지 후진하고 열린 측면으로 회전하는지 검증	Passed
5	<code>RvcSystemTest.SimulatorKeepsCleanerOffDuringBoostedEscape</code>	boost 상태에서도 후진 탈출 및 회피 회전 중에는 클리너가 꺼지는지 검증	Passed
6	<code>RvcSystemTest.SimulatorTurnsAfterFrontInterrupt</code>	시뮬레이터가 전방 장애물을 인터럽트로 변환하고 올바르게 회전하는지 검증	Passed
7	<code>RvcSystemTest.SimulatorRunWithZeroTicksLeavesRobotAtInitialPose</code>	0 tick 실행 시 로봇 위치, 방향, 먼지, 로그가 초기 상태로 유지되는지 검증	Passed
8	<code>RvcSystemTest.SimulatorRejectsNegativeTicks</code>	음수 tick 실행 요청을 예외로 거부하는지 검증	Passed
9	<code>RvcSystemTest.SimulatorCanAppendRenderedFrames</code>	맵 렌더링을 포함한 실행 로그가 기대 형식으로 추가되는지 검증	Passed

시뮬레이터 시스템 테스트 소계: 9 passed / 0 failed.

6. 시나리오 파일 테스트 결과

시나리오 파일 테스트는 `.rvc` 파일 로딩, 필수 맵 검증, 로봇 마커 오류, 음수 tick, 현재 알려진 심볼 검증 간극을 문서화한다.

번호	테스트 케이스	검증 대상	결과
1	<code>RvcScenarioFileTest.LoadsScenarioFileWithTicksAndMap</code>	시나리오 파일에서 tick과 map 섹션을 올바르게 로드하는지 검증	Passed
2	<code>RvcScenarioFileTest.RejectsMissingOrEmptyMapFilesDuringLoad</code>	map 섹션이 없거나 비어 있는 시나리오를 로딩 단계에서 거부하는지 검증	Passed
3	<code>RvcScenarioFileTest.RejectsRobotMarkerErrorsAfterLoadingMap</code>	로봇 마커가 없거나 여러 개인 맵을 시뮬레이터 생성 단계에서 거부하는지 검증	Passed
4	<code>RvcScenarioFileTest.RejectsNegativeTickScenarioWhenRun</code>	음수 tick 시나리오가 실행 단계에서 예외 처리되는지 검증	Passed
5	<code>RvcScenarioFileTest.UnknownSymbolScenarioDocumentsCurrentValidationGap</code>	알 수 없는 심볼에 대한 현재 검증 간극과 실제 동작을 회귀 기준으로 문서화	Passed

시나리오 파일 테스트 소계: 5 passed / 0 failed.

8. CLI 테스트 결과

CLI 테스트는 빌드된 `rvc_simulator` 실행 파일이 기본 옵션과 시나리오 입력 옵션으로 정상 종료되는지 검증한다.

번호	테스트 케이스	검증 대상	결과
1	<code>SimulatorCliDefaultRuns</code>	기본 CLI 시뮬레이터 실행이 성공적으로 완료되는지 검증	Passed
2	<code>SimulatorCliContinuousBackwardScenarioRuns</code>	CLI가 continuous backward 시나리오를 성공적으로 실행하는지 검증	Passed

CLI 테스트 소계: 2 passed / 0 failed.

System Test

시뮬레이터가 사용자가 이해할 수 있게 기록을 잘 남기는가?

✓ RvcSystemTest (9)	< 1ms
✓ SimulatorCanAppendRenderedFrames	< 1ms GoogleTes...
✓ SimulatorCleansDustAndLogsCommands	< 1ms GoogleTes...
✓ SimulatorKeepsBackingUpUntilSideExitOpens	< 1ms GoogleTes...
✓ SimulatorKeepsCleanerOffDuringBoostedEscape	< 1ms GoogleTes...
✓ SimulatorKeepsCommandingBackwardWhenBoxedIn	< 1ms GoogleTes...
✓ SimulatorRejectsNegativeTicks	< 1ms GoogleTes...
✓ SimulatorRunWithZeroTicksLeavesRobotAtInitialPose	< 1ms GoogleTes...
✓ SimulatorTurnsAfterFrontInterrupt	< 1ms GoogleTes...
✓ SimulatorUsesBackwardEscape	< 1ms GoogleTes...

시나리오 파일을 잘 불러올 수 있는가?

✓ RvcScenarioFileTest (5)	< 1ms
✓ LoadsScenarioFileWithTicksAndMap	< 1ms GoogleTes...
✓ RejectsMissingOrEmptyMapFilesDuringLoad	< 1ms GoogleTes...
✓ RejectsNegativeTickScenarioWhenRun	< 1ms GoogleTes...
✓ RejectsRobotMarkerErrorsAfterLoadingMap	< 1ms GoogleTes...
✓ UnknownSymbolScenarioDocumentsCurrentValidationGap	< 1ms GoogleTes...

기본 모드와 시나리오 모드를 구분해서 작동시킬 수 있는가?

✓ SimulatorCliContinuousBackwardScenarioRuns	69ms
✓ SimulatorCliDefaultRuns	34ms

7. 시나리오 회귀 테스트 결과

시나리오 회귀 테스트는 `scenarios` 디렉터리의 대표 입력 파일을 실행해 최종 위치, 최종 방향, 청소된 먼지 수, 잔여 먼지 수, 이동 명령 수, 클리너 명령 수, 필수 로그 조각을 함께 검증한다.

번호	시나리오	검증 대상	결과
1	<code>backward_escape.rvc</code>	막힌 구역에서 후진 후 좌측 탈출구로 회전하는 기본 탈출 흐름	Passed
2	<code>backward_escape2.rvc</code>	긴 탈출 경로에서 후진과 우회전 탈출이 기대 횟수로 발생하는지 검증	Passed
3	<code>boundary_without_outer_wall.rvc</code>	외곽 벽이 없는 경계 맵에서 전방 인터럽트와 회전이 안정적으로 동작하는지 검증	Passed
4	<code>clear_corridor_forward.rvc</code>	장애물이 없는 복도에서 계속 전진 청소하는지 검증	Passed
5	<code>continuous_backward.rvc</code>	계속 막힌 상태에서 매 tick 후진 명령과 클리너 off를 유지하는지 검증	Passed
6	<code>dense_dust_maze_extreme.rvc</code>	먼지가 많은 복잡한 미로에서 boost 청소와 회피가 기대 흐름으로 발생하는지 검증	Passed
7	<code>dust_and_interrupt.rvc</code>	먼지 감지와 전방 인터럽트가 함께 있는 시나리오에서 boost와 회피 로그가 남는지 검증	Passed
8	<code>dust_before_dead_end_escape.rvc</code>	먼지 청소 직후 막힌 길에서 탈출하며 클리너를 끄는지 검증	Passed
9	<code>dust_trail_boost_refresh.rvc</code>	먼지 궤적을 따라 이동하며 boost 예산을 갱신하는지 검증	Passed
10	<code>front_both_sides_open.rvc</code>	전방 장애물과 양측 개방 상태에서 좌우 회전을 모두 수행하는지 검증	Passed
11	<code>front_clears_but_sides_still_blocked.rvc</code>	전방이 다시 열려도 측면이 막힌 탈출 흐름을 안정적으로 처리하는지 검증	Passed
12	<code>front_left_only_open.rvc</code>	좌측만 열린 경우 좌회전 회피를 수행하는지 검증	Passed
13	<code>front_right_only_open.rvc</code>	우측만 열린 경우 우회전 회피를 수행하는지 검증	Passed
14	<code>large_open_room_dust_sweep.rvc</code>	큰 열린 방에서 먼지 청소와 일반 전진이 기대 횟수로 발생하는지 검증	Passed
15	<code>long_escape_left_exit_extreme.rvc</code>	긴 후진 탈출 끝에 좌측 출구로 회전하는 극단 사례 검증	Passed
16	<code>long_escape_right_exit_extreme.rvc</code>	긴 후진 탈출 끝에 우측 출구로 회전하는 극단 사례 검증	Passed
17	<code>narrow_tunnel_sides_blocked_front_clear.rvc</code>	좁은 터널에서 측면이 막혀도 전방이 열리면 계속 전진하는지 검증	Passed
18	<code>ragged_map_edge_extreme.rvc</code>	들쭉날쭉한 맵 경계에서 이동과 전방 인터럽트가 안정적으로 처리되는지 검증	Passed
19	<code>repeated_front_interrupts_alternation.rvc</code>	반복되는 전방 인터럽트에서 회전 방향 교대 규칙이 유지되는지 검증	Passed
20	<code>sealed_box_extreme.rvc</code>	완전 봉쇄 맵에서 위치와 방향을 유지하며 후진 명령을 반복하는지 검증	Passed

시나리오 회귀 테스트 소개: 20 passed / 0 failed.

System Test

실질적인 시스템 테스트 수행

```
Windows PowerShell
#####
#.#.#.#.#
#...>.#
#.#.#.#.#
#####
tick=11 frontInterrupt=false leftPeriodic=open rightPeriodic=open dustPeriodic=clean
ion=(2,6) direction=East cleaned=0 reason="front clear: forward cleaning"
#####
#.#.#.#.#
#...>.#
#.#.#.#.#
#####
tick=12 frontInterrupt=false leftPeriodic=blocked rightPeriodic=blocked dustPeriodic=
position=(2,7) direction=East cleaned=0 reason="front clear: forward cleaning"
#####
#.#.#.#.#
#...>.#
#.#.#.#.#
#####
Summary ticks=12 dustCleaned=0 remainingDust=0 finalPosition=(2,7) finalDirection=E
Final map
#####
#.#.#.#.#
#...>.#
#.#.#.#.#
#####
PS D:\OoAD_VIBE\ooad-vibe>
```

ScenarioFiles/RvcScenarioRegressionTest (20)	4ms
ScenarioFileRunsWithExpectedOutcome/backward_escape_rvc [bac...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/backward_escape2_rvc [ba...	1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/boundary_without_outer_...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/clear_corridor_forward_rvc...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/continuous_backward_rvc...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/dense_dust_maze_extreme...	2ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/dust_and_interrupt_rvc [du...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/dust_before_dead_end_esc...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/dust_trail_boost_refresh_rv...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/front_both_sides_open_rvc...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/front_clears_but_sides_still...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/front_left_only_open_rvc [f...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/front_right_only_open_rvc...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/large_open_room_dust_sw...	1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/long_escape_left_exit_extr...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/long_escape_right_exit_ext...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/narrow_tunnel_sides_block...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/ragged_map_edge_extrem...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/repeated_front_interrupts_...	< 1ms GoogleTes...
ScenarioFileRunsWithExpectedOutcome/sealed_box_extreme_rvc [s...	< 1ms GoogleTes...

9. 결론

최신 Debug 빌드 기준으로 전체 57개 테스트가 모두 통과했다. 현재 구현은 시작/중지, idle 상태 처리, 장애물 인터럽트 소비, 좌우 회피, 막힌 구역 탈출, 먼지 boost 청소, boost 갱신, 회피/탈출 중 클리너 off, 타입 변환, 시뮬레이터 로그와 렌더링, 시나리오 파일 로딩 및 오류 처리, 대표 시나리오 회귀 동작, CLI 실행에 대해 검증된 동작을 만족한다.

SYSTEM CODE ANALYSIS

System Code Analysis

CppCheck

```
PS C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe> cppcheck --enable=warning,style,performance,portability --std=c++20 -I include src/ 2>&1
Checking src\GridSimulator.cpp ...
1/4 files checked 49% done
Checking src\RvcController.cpp ...
cppcheck : src\GridSimulator.cpp:239:25: style: The member function 'rvc::GridSimulator::adjacent' can be static. [functionStatic]
At line:1 char:1
+ cppcheck --enable=warning,style,performance,portability --std=c++20 - ...
+ ~~~~~
+ CategoryInfo          : NotSpecified: (src\GridSimulat...functionStatic]:String) [], RemoteException
+ FullyQualifiedErrorId : NativeCommandError

Position GridSimulator::adjacent(Position origin, Direction direction) const {
    ^
2/4 files checked 72% done
include\rvc\GridSimulator.hpp:47:28: note: The member function 'rvc::GridSimulator::adjacent' can be static.Checking src\Types.cpp ...

    [[nodiscard]] Position adjacent(Position origin, Direction direction) const;
    ^
src\GridSimulator.cpp:239:25: note: The member function 'rvc::GridSimulator::adjacent' can be static.
Position GridSimulator::adjacent(Position origin, Direction direction) const {
    ^
src\RvcController.cpp:122:24: style: The member function 'rvc::RvcController::makeCommand' can be static. [functionStatic]
Command RvcController::makeCommand(Motion motion, CleaningPower power, std::string reason) const {
3/4 files checked 85% done
Checking src\main.cpp ...

include\rvc\RvcController.hpp:30:27: note: The member function 'rvc::RvcController::makeCommand' can be static.
    [[nodiscard]] Command makeCommand(Motion motion, CleaningPower power, std::string reason) const;
    ^
src\RvcController.cpp:122:24: note: The member function 'rvc::RvcController::makeCommand' can be static.
Command RvcController::makeCommand(Motion motion, CleaningPower power, std::string reason) const {
    ^
4/4 files checked 100% done
PS C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe>
```

Clang-Tidy

```
--- CLANG TIDY STARTED ---
PROJECT #27: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\ALL_BUILD.vcxproj [Debug|x64]
PROJECT #26: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\Continuous.vcxproj [Debug|x64]
PROJECT #25: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\Experimental.vcxproj [Debug|x64]
PROJECT #24: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\INSTALL.vcxproj [Debug|x64]
PROJECT #23: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\Nightly.vcxproj [Debug|x64]
PROJECT #22: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\NightlyMemoryCheck.vcxproj [Debug|x64]
PROJECT #21: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\RUN_TESTS.vcxproj [Debug|x64]
PROJECT #20: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\ZERO_CHECK.vcxproj [Debug|x64]
PROJECT #19: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\googletest\gmock.vcxproj [Debug|x64]
PROJECT #18: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\googletest\gmock_main.vcxproj [Debug|x64]
PROJECT #17: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\googletest\gtest.vcxproj [Debug|x64]
PROJECT #16: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\googletest\gtest_main.vcxproj [Debug|x64]
PROJECT #15: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\rvc_core.vcxproj [Debug|x64]
PROJECT #14: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\rvc_simulator.vcxproj [Debug|x64]
PROJECT #13: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\rvc_tests.vcxproj [Debug|x64]
PROJECT #12: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\ALL_BUILD.vcxproj [Debug|x64]
PROJECT #11: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\INSTALL.vcxproj [Debug|x64]
PROJECT #10: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\RUN_TESTS.vcxproj [Debug|x64]
PROJECT #9: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\googletest\ALL_BUILD.vcxproj [Debug|x64]
PROJECT #8: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\googletest\INSTALL.vcxproj [Debug|x64]
PROJECT #7: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\googletest\RUN_TESTS.vcxproj [Debug|x64]
PROJECT #6: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\googletest\ALL_BUILD.vcxproj [Debug|x64]
PROJECT #5: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\googletest\INSTALL.vcxproj [Debug|x64]
PROJECT #4: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-build\googletest\RUN_TESTS.vcxproj [Debug|x64]
PROJECT #3: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-subbuild\ALL_BUILD.vcxproj [Debug|x64]
PROJECT #2: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-subbuild\ZERO_CHECK.vcxproj [Debug|x64]

PROJECT: C:\Users\문재현\Desktop\대학\학기\3학년 2학기\객체지향개발방법론\HW_5\ooad-vibe\build\_deps\googletest-subbuild\googletest-populate.vcxproj [Debug|x64]

--- CLANG TIDY FINISHED ---
```

System Code Analysis

CppCheck

1) Cppcheck 결과

Cppcheck 분석 결과, 멤버 변수에 접근하지 않아 `static` 함수로 변경할 수 있는 멤버 함수에 대한 **스타일 경고(style)** 2건이 감지되었습니다.

- **경고 1:** `GridSimulator::adjacent` 함수 `static` 선언 가능
 - **파일 위치:** [GridSimulator.cpp:L239](file:///C:/Users/문재현/Desktop/대학/학기/3학년 2학기/객체지향개발방법론/HW_5/ooad-vibe/src/GridSimulator.cpp#L239)
 - **메시지:** style: The member function 'rvc::GridSimulator::adjacent' can be static. [functionStatic]
 - **설명:** 해당 함수는 `GridSimulator` 클래스의 멤버 변수를 사용하지 않고 독립적으로 동작하는 유틸리티적 성격의 함수이므로 `static` 멤버 함수로 전환하는 것이 권장됩니다.
- **경고 2:** `RvcController::makeCommand` 함수 `static` 선언 가능
 - **파일 위치:** [RvcController.cpp:L122](file:///C:/Users/문재현/Desktop/대학/학기/3학년 2학기/객체지향개발방법론/HW_5/ooad-vibe/src/RvcController.cpp#L122)
 - **메시지:** style: The member function 'rvc::RvcController::makeCommand' can be static. [functionStatic]
 - **설명:** 해당 함수도 클래스 내부의 인스턴스 멤버 변수를 참조하지 않기 때문에 `static` 멤버 함수로 구성되어 불필요한 `this` 포인터 전달을 줄일 수 있습니다.

Clang-Tidy

2) Clang-Tidy 결과

Clang-Tidy 분석 결과, 소스 코드에서 추가로 탐지된 코딩 표준 위반 사례나 메모리 누수, 모호한 표현 등의 잠재적 오류는 발견되지 않았습니다.

4. 권장 조치 사항

Cppcheck에서 검출된 2건의 `functionStatic` 스타일 경고는 프로그램의 동작에 영향을 주는 치명적인 오류는 아닙니다. 그러나 코드의 품질과 최적화를 위해 아래와 같이 변경하는 것을 권장합니다.

1. `GridSimulator::adjacent` 변경:

- [GridSimulator.hpp](file:///C:/Users/문재현/Desktop/대학/학기/3학년 2학기/객체지향개발방법론/HW_5/ooad-vibe/include/rvc/GridSimulator.hpp)에서 `Position adjacent(...)` 선언 앞에 `static` 을 붙이고, [GridSimulator.cpp](file:///C:/Users/문재현/Desktop/대학/학기/3학년 2학기/객체지향개발방법론/HW_5/ooad-vibe/src/GridSimulator.cpp)의 구현부 맨 뒤에 붙은 `const` 키워드를 제거합니다.

2. `RvcController::makeCommand` 변경:

- [RvcController.hpp](file:///C:/Users/문재현/Desktop/대학/학기/3학년 2학기/객체지향개발방법론/HW_5/ooad-vibe/include/rvc/RvcController.hpp)에서 `Command makeCommand(...)` 선언 앞에 `static` 을 붙이고, [RvcController.cpp](file:///C:/Users/문재현/Desktop/대학/학기/3학년 2학기/객체지향개발방법론/HW_5/ooad-vibe/src/RvcController.cpp)의 구현부 맨 뒤에 붙은 `const` 키워드를 제거합니다.

SUMMARY & TROUBLE SHOOTING

Trouble Shooting

후진할 때, 후진 시 앞이 뚫리면 다시 전진
후진 하는 즉시 앞이 뚫리므로, 문제가 해결되지
“backproblemsolve” 브랜치에서 해결

```
tick=6
rd clea
#####
#####
##^##
##.##
##.##
##.##
##.##
##.##
##.##
##.##
##.##
##.##
#####
```

```
tick=7
ed: ent
#####
#####
##.##
##^##
##.##
##.##
##.##
##.##
##.##
##.##
##.##
#####
```

```
tick=8
rd clea
#####
#####
##^##
##.##
##.##
##.##
##.##
##.##
##.##
##.##
##.##
#####
```

SSD-05 번에서 loop를 돌 때 front is open이면 빠져나오는데, 후진 한 번 하면 바로 앞에 장애물이 없는 상태가 되므로 결국 앞으로 갔다 뒤로 가는 상황이 됨. 이 문제를 해결하기 위해선 좌/우만 보고 탈출하도록 해야 함.
위와 더불어 SD-05에서도 후진 로직을 변경해야 함
변경된 로직에 따라 실제 후진하는 코드 변경
후진하는 테스트 케이스 추가. Unit test, Simulator를 이용한 Test
코드가 수정되면 문서도 그에 맞게 함께 수정

3m 34s 동안 작업 >

수정 완료했습니다.




핵심 변경은 [RvcController.cpp \(line 54\)](#)에서 `Escaping` 상태의 탈출 조건을 `front` `left/right` 만 보도록 바꾼 것입니다. 이제 후진 직후 전방이 열려도 좌/우가 모두 막혀 좌/우 중 하나가 열릴 때만 회전합니다.

추가한 검증:


- Unit test: [controller_tests.cpp \(line 102\)](#)
- Simulator test: [system_tests.cpp \(line 71\)](#)
- 문서 갱신: requirements, SSD-05, SD-05, class/domain diagram, traceability

검증 결과:

- `cmake --build build --config Debug` 통과
- `ctest --test-dir build -C Debug --output-on-failure` 통과, 14/14
- `scenarios/backward_escape2.rvc` 수동 실행에서도 tick 2 이후 `frontInterrupt` `open`에서 회전하는 것 확인했습니다.

 requirements.md	문서 · MD	열기 ▾
 ooa_ssd.md	문서 · MD	열기 ▾
 ood_sequence_diagrams.md	문서 · MD	열기 ▾

3개 더 보기 ▾

 파일 9개 편집됨	+60 -36	실행 취소 ↶	리뷰
src/RvcController.cpp	+4 -8	▾	
tests/controller_tests.cpp	+13 -4	▾	
tests/system_tests.cpp	+24 -0	▾	

6개 파일 더 보기 ▾

```
tick=6
exit 0
#####
#####
##.##
##.##
##.##
##.##
##.##
##.##
##.##
##.##
##.##
#####
```

```
tick=7
#####
#####
##.##
##.##
##.##
##.##
##.##
##.##
##.##
##.##
#####
```

```
tick=8
#####
#####
##.##
##.##
##.##
##.##
##.##
##.##
##.##
##.##
#####
```

Trouble Shooting

요구사항에서는 회전 후진할때 청cleaner 꺼지라하지않음?
(일요일) 오후 9:50

28s 동안 작업 >

네, 맞아요. 요구사항 기준으로는 회전하거나 후진할 때 cleaner를 끄라는 말은 없습니다.
문서상 cleaner off가 명시된 건 청소 중지뿐입니다.

- docs/requirements.md: UC-02 / FR-02
stopCleaning() 시 motor stop + cleaner off
- UC-03 회피: 열린 방향으로 회전한 뒤 전진 청소 재개
- UC-04 탈출: 계속 후진, 탈출 가능하면 회전하고 청소 재개
- FR-14/15: dust boost는 motion과 별개로 cleaner power를 normal/boost로 유지

현재 구현도 그 해석대로 되어 있어요. src/RvcController.cpp에서 tick() 이 running 상태면 먼저 updateCleaningPower() 로 Normal / Boost 를 정하고, Backward, TurnLeft, TurnRight 명령에도 그 power를 그대로 붙입니다. off 는 idle일 때만 반환됩니다.

즉 요구사항대로라면 회전/후진 중 cleaner는 꺼지면 안 되고, Normal 또는 Boost 상태가 유지되는 게 맞습니다.

requirements.md
문서 · MD

pdf 내용은?

41s 동안 작업 >

PDF 원문 기준으로 보면, docs/rvc.pdf 4페이지에 RVC 요구사항이 이렇게 적혀 있습니다.
핵심은 이 문장입니다:

"If its sensors found an obstacle, it stops cleaning, turns aside left or right, and goes forward with cleaning."

즉 PDF만 엄격히 읽으면 장애물 감지 시 일단 cleaning을 멈추고, 좌/우 회전한 뒤 다시 cleaning하면서 전진한다고 해석할 수 있습니다.

다만 삼방향 막힘에 대해서는:

"If there are obstacles in both front, left and right, it move backward and turn aside left or right, and goes forward."

라고 되어 있어서 후진 중 cleaner를 끈다/켄다를 명시하진 않습니다.

정리하면:

- 회전 시: PDF에는 "stops cleaning, turns aside..."라고 되어 있어서 cleaner off로 볼 여지가 큼니다.
- 후진 시: PDF에는 cleaner off가 직접 명시되어 있지 않습니다.
- 현재 docs/requirements.md 는 PDF보다 완화해서 "청소를 계속한다" 쪽으로 정리돼 있어, 현재 구현과 맞지만 PDF 원문과는 약간 차이가 있습니다.

그래서 교수님/PDF 기준을 따지면 회전 중 cleaner는 꺼지는 쪽으로 고치는 게 더 안전해 보입니다. 후진 중은 애매하지만, "stops cleaning → 이동/회피 → goes forward with cleaning" 흐름으로 일관되게 보면 후진/회전 동안 Off, 전진 때 Normal/Boost로 잡는 해석도 가능합니다.

- 요구사항에는 장애물 감지 시 Cleaning을 멈춰야 함
- 실제 구현 사항에는 Cleaning을 멈추는 코드가 없었음
- 구현 사항과 다른 점을 발견하고 수정

```
PS C:\Users\문재현\Desktop\대학\학기\3학년_2학기\객체지향개발방법론\HW_5\ood-vibe> build\Debug\rvc_simulator.exe --scenario scenarios\backward_escape.rvc
Initial map
#####
#####
#####
#...#
#####

tick=1 frontInterrupt=true leftPeriodic=blocked rightPeriodic=blocked dustPeriodic=clear motion=Backward cleaner=Off position=(3,2) direction=North cleaned=0 reason="front interrupt and both sides blocked:
enter escaping"
#####
#####
#.#
#.^.#
#####

tick=2 frontInterrupt=false leftPeriodic=open rightPeriodic=open dustPeriodic=clear motion=TurnLeft cleaner=Off position=(3,2) direction=West cleaned=0 reason="escaping: side opened, turn toward exit"
#####
#####
#.#
#.<.#
#####

tick=3 frontInterrupt=false leftPeriodic=blocked rightPeriodic=open dustPeriodic=clear motion=Forward cleaner=Normal position=(3,1) direction=West cleaned=0 reason="front clear: forward cleaning"
#####
#####
#.#
#<.#
#####

tick=4 frontInterrupt=true leftPeriodic=blocked rightPeriodic=blocked dustPeriodic=clear motion=Backward cleaner=Off position=(3,2) direction=West cleaned=0 reason="front interrupt and both sides blocked:
enter escaping"
#####
#####
#.#
#.<.#
#####
```

반영해서 개발문서 반영해줘 코드 구현은 문서 확인하고 할거임

(일요일) 오후 9:53

좋아요. 지금은 계획 모드라서 실제 문서 수정은 하지 않고, PDF 원문 해석을 어떤 개발문서에 어떻게 반영할지 확정 가능한 계획으로 묶겠습니다. 먼저 현재 문서들 사이에 cleaner 동작이 어떻게 퍼져 있는지 한 번 더 좁혀서 확인할게요.

플랜

PDF 기준 Cleaner Off 정책 개발문서 반영 계획

요약

- docs/rvc.pdf 4페이지의 "stops cleaning, turns aside..." 문구를 기준으로, 회피/탈출 이동 중에는 cleaner를 끄고 전진 청소 재개 시 다시 Normal 또는 Boost로 켜는 정책을 개발문서에 명시한다.
- 코드 구현은 하지 않고, 이후 구현자가 문서만 보고 TurnLeft, TurnRight, Backward 시 CleaningPower::Off 를 반환하도록 바꿀 수 있게 요구사항과 설계를 정리한다.
- 기본 해석은 사용자 의도에 맞춰 회전뿐 아니라 후진도 cleaner off 대상으로 둔다.

주요 변경

- docs/requirements.md
 - UC-03: 전방 장애물 감지 시 전진 청소를 멈추고 cleaner를 끈 뒤 회전, 전진 재개 시 cleaner를 다시 켜라고 수정.
 - UC-04: 삼방향 막힘 시 cleaner를 끄고 후진/회전 탈출을 수행한 뒤 전진 재개 시 cleaner를 다시 켜라고 수정.
 - FR 추가 또는 보강:
 - 회피/탈출 이동(Backward, TurnLeft, TurnRight) 중 cleaner power는 Off .
 - 전진 청소(Forward) 중 cleaner power는 먼지 boost 상태에 따라 Normal 또는 Boost .
 - 먼지 boost 타이머는 cleaner가 off인 회피/탈출 중에도 내부 상태로는 유지되며, 전진 재개 시 남은 boost 상태가 반영됨.
 - Core Control Rules 4, 7, 9, 10에 cleaner off/restart 규칙 반영.
- OOA/OOD 문서
 - docs/ooo_ssd.md: 장애물 회피와 탈출 SSD에 cleaner: turnOff() 를 추가하고, 전진 재개 시 setPower(Normal/Boost) 흐름을 명시.
 - docs/ood_sequence_diagrams.md: SD-04/SD-05의 Command(..., currentCleanerPower) 를 회전/후진에서는 Command(..., Off) 로 바꾸고, 전진에서는 activeCleanerPower 로 표현.
 - docs/ood_class_diagram.md: Command.cleaningPower 설명에 "motion에 따라 off가 될 수 있음"을 명시하고, boost 계산값과 실제 actuator power가 분리될 수 있음을 덧붙임.
- 추적성과 시나리오 문서
 - docs/traceability.md: 새 cleaner-off 요구사항을 UC-03/UC-04, SD-04/SD-05, 관련 향후 테스트에 연결.
 - docs/scenario_catalog.md: 전방 회피, 후진 탈출, dust+escape 시나리오에 "회피/탈출 중 cleaner=Off 확인" 목적을 추가.

구현자가 따라야 할 의미

- Motion::Forward : cleaner는 Normal 또는 Boost .
- Motion::TurnLeft, Motion::TurnRight, Motion::Backward, Motion::Stop : cleaner는 Off .
- stopCleaning() 은 기존처럼 motor stop + cleaner off.
- dust boost duration은 "청소 세기 상태"로 유지하되, 실제 cleaner 출력은 전진할 때만 적용한다.

테스트 계획

- 문서 반영 후 코드 구현 단계에서 다음 테스트를 추가/수정한다.
- 전방 장애물 후 좌/우 회전 명령은 cleaningPower == Off .
- 삼방향 막힘으로 후진 명령이 나올 때 cleaningPower == Off .
- Escaping 중 side exit 발견 후 회전 명령도 cleaningPower == Off .
- dust boost 중 장애물을 만나면 회피/후진 동안은 off, 전진 재개 시 boost tick이 남아 있으면 Boost .
- 기존 전진 청소 및 stopCleaning 테스트는 유지하되, 회전/후진 테스트의 기존 Normal 기대값은 Off 로 수정한다.

가정

- PDF의 "stops cleaning"을 cleaner off로 해석한다.
- PDF에서 후진 중 cleaner off가 직접 쓰여 있지는 않지만, 사용자의 "회전 후진할 때" 요청과 회피 이동 일관성을 기준으로 후진도 cleaner off로 문서화한다.
- 이번 작업 범위는 개발문서 수정이며, 소스 코드와 테스트 코드는 이후 별도 단계에서 수정한다.

pdf 에서 또 반영이 안된 내용이 있으면 반영

PDF 원문에서 요구사항으로 볼 수 있는 부분을 문서와 대조해서 빠진 항목을 더 찾겠습니다. 특히 4페이지의 Preliminary/Future 요구사항과 5페이지 DFD가 현재 개발문서에 얼마나 반영됐는지 확인할게요.

플랜

다운로드 공유 댓글

PDF 원문 누락사항까지 포함한 개발문서 반영 계획

요약

- docs/rvc.pdf 4페이지의 RVC 요구사항과 5페이지 DFD를 기준으로, 현재 개발문서에 덜 반영된 내용을 보강한다.
- 코드 구현은 하지 않고, 이후 구현자가 문서 기준으로 cleaner off 정책과 확장 요구사항을 확인할 수 있게 문서만 정리한다.
- 핵심 보강점은 회피/탈출 중 cleaner off, 청소 및 물걸레질 범위 명시, 향후 확장 요구사항 분리, DFD 입출력 용어 정합성이다.

주요 문서 변경

- docs/requirements.md

- Scope에 PDF 원문처럼 RVC가 household surface를 “청소하고 물걸레질한다”는 설명을 추가하되, 현재 제어 SW에 서는 이를 cleaner actuator의 추상 cleaning 기능으로 다룬다고 명시한다.
- UC-03에 “장애물 감지 시 cleaning을 멈추고 cleaner를 끈 뒤 회전, 이후 전진 청소 재개”를 반영한다.
- UC-04에 “삼방향 막힘 시 cleaner를 끄고 후진/회전 탈출, 이후 전진 재개”를 반영한다.
- FR에 다음 규칙을 추가한다:
 - 회피/탈출 이동(Backward , TurnLeft , TurnRight) 중 cleaner power는 Off .
 - 전진 청소(Forward) 중 cleaner power는 dust 상태에 따라 Normal 또는 Boost .
 - dust boost 상태는 회피/탈출 중에도 내부 타이머로 유지될 수 있으나 실제 cleaner 출력은 전진 중에만 적용된다.

- “Future or Extended Requirements” 절을 추가해 센서 추가/변경, 한 지점 순환 청소, 모바일 앱 통신, 머신러닝/추론 기반 효율 개선을 향후 고려사항으로 분리한다.

- OOA/OOD 문서

- docs/ooa_domain_diagram.md: cleaner 설명에 normal/boost/off뿐 아니라 청소/물걸레질을 대표하는 추상 actuator라는 의미를 추가한다.
- docs/ooa_ssd.md: 장애물 회피와 탈출 SSD에 cleaner: turnOff() 를 추가하고, 전진 재개 시 setPower(Normal/Boost) 흐름을 명시한다.
- docs/ood_sequence_diagrams.md: SD-04/SD-05에서 회전/후진 명령의 cleaner power를 off 로 바꾸고, 전진 명령만 active cleaner power를 사용하도록 수정한다.
- docs/ood_class_diagram.md: Command.cleaningPower 설명에 motion별 cleaner 출력 정책을 추가한다.

- 추적성과 시나리오 문서

- docs/traceability.md: 새 cleaner-off 요구사항을 UC-03/UC-04, SD-04/SD-05, 향후 테스트 항목에 연결한다.
- docs/scenario_catalog.md: 회피/후진/탈출 시나리오에 cleaner=off 검증 목적을 추가하고, dust+escape 시 boost와 off의 우선순위를 확인하는 목적을 명시한다.

PDF 누락사항 반영 기준

- “An RVC automatically cleans and mops household surface.”
 - 제품 설명과 scope에는 반영하되, 현재 구현 범위에서는 별도 mop actuator를 만들지 않고 cleaner 로 추상화한다.

- “stops cleaning, turns aside left or right...”
 - 회전 중 cleaner off 요구사항으로 반영한다.

- “move backward and turn aside...”
 - 후진과 회전 모두 탈출 이동으로 보고 cleaner off 요구사항으로 반영한다.

- “Future or Extended Requirements to Consider”
 - 현재 FR로 구현 대상화하지 않고 별도 향후 확장 요구사항으로 문서화한다.

- DFD Level 0의 Direction , Clean , Tick
 - Command.motion , Command.cleaningPower , DigitalClock.tick() 과 대응된다는 설명을 설계 문서에 보강한다.

테스트 계획

- 문서 반영 후 코드 구현 단계에서 다음 테스트를 추가/수정한다.
- 장애물 회피 회전 명령은 cleaningPower == Off .
- 삼방향 막힘 후진 명령은 cleaningPower == Off .
- Escaping에서 side exit 발견 후 회전 명령도 cleaningPower == Off .
- dust boost 중 회피/탈출하면 실제 command는 off , 전진 재개 시 boost tick이 남아 있으면 Boost .
- 기존 전진 청소, stopCleaning, sensor sampling, backward escape 테스트는 새 cleaner 정책에 맞게 기대값만 조정한다.

가정

- PDF의 “stops cleaning”은 cleaner off로 해석한다.
- PDF가 후진 중 cleaner off를 직접 쓰지는 않았지만, 회피 이동 중 cleaning 중지라는 정책 일관성과 사용자 요청에 맞춰 후진도 off로 문서화한다.
- “mops”는 현재 버전에 별도 물걸레 하드웨어나 제어 명령을 추가하지 않고, RVC 기능 설명과 cleaner 추상화에만 반영한다.
- 모바일 앱, 한 지점 순환, 머신러닝/추론은 이번 구현 범위, 시나리오 future requirements로만 기록한다.

3m 7s 동안 작업 >

개발문서에 PDF 원문 누락사항까지 반영했습니다. 코드/테스트 구현 파일은 건드리지 않았고 문서만 수정했습니다.

반영 파일:

- requirements.md
- ooa_domain_diagram.md
- ooa_ssd.md
- ood_sequence_diagrams.md
- ood_class_diagram.md
- traceability.md
- scenario_catalog.md

핵심 변경은 FR-16~FR-18 로 cleaner off/forward boost 우선순위를 새로 명시했고, 기존 simulator 요구사항 21 로 밀었습니다. PDF의 mops , future requirements, DFD의 Direction/Clean/Tick 대응도 설계 문서에 반영했습니다.

검증은 git diff --check -- docs 로 공백 오류 없음을 확인했고, 예전 currentCleanerPower 기반 호환성 이슈가 아 있는지 검색했습니다.

requirements.md	문서 · MD
ooa_domain_diagram.md	문서 · MD
ooa_ssd.md	문서 · MD
4개 더 보기 >	

파일 7개 편집됨	실행 취소
+97 -54	
docs/requirements.md	
docs/ooa_domain_diagram.md	
docs/ooa_ssd.md	
4개 파일 더 보기 >	

공유 댓글

문서화 능력

- AI는 문서화 작업에 뛰어남
- 특히, SRS, SSD를 이미 있는 포맷에 맞춰서 작성해달라 했을 때 강점을 느낄 수 있었음
- 표, 다이어그램 등 다양한 형식을 쉽게 만들 수 있기 때문에 문서화 작업이 많이 간편해짐
- 다만, 표 및 다이어그램 스타일이 미관상 좋지 않거나 난잡할 수 있는데, 사용할 수 있는 도구의 한계로 보임
- 있어빌리티를 높이려면 디자인 부분에서 사람의 개입이 필요하거나, 외부 Tool을 붙여서 발전시키는 것이 좋다고 느낌

RVC Control SW Software Design Description

본 문서는 IEEE Std 1016-2009의 SDD 구조를 기준으로 작성한다.

1. Scope

1.1 Purpose

이 문서는 RVC(Robotic Vacuum Cleaner) 자동 청소 제어 소프트웨어의 Software Design Description(SDD)이다. docs/srs.md 에 정의된 요구사항을 현재 C++20 구현 구조와 연결하고, 설계 context, stakeholder concern, design viewpoint, design view, design rationale을 정리한다.

1.2 System Scope

설계 대상은 RvcController 기반 제어 로직과 GridSimulator 기반 CLI 시뮬레이터이다. controller는 sensor 입력을 판단해 motor motion과 cleaner power를 포함한 Command 를 생성한다. simulator는 검증 환경으로서 scenario map에서 sensor 값을 만들고 command를 적용한다.

본 SDD의 범위에 포함하지 않는 항목은 실제 모터 드라이버, 실제 센서 하드웨어, 배터리, 네트워크, UI, 영구 저장소 설계이다.

1.3 Design Goals

RVC Control SW Software Requirements Specification

본 문서는 IEEE Std 830-1998의 SRS 구조를 기준으로 작성한다.

1. Introduction

1.1 Purpose

이 문서는 RVC(Robotic Vacuum Cleaner) 자동 청소 제어 소프트웨어의 Software Requirements Specification(SRS)을 정의한다. 요구사항은 docs/rvc.pdf 의 원본 문제 정의, OOA 산출물, 현재 C++20 구현 및 테스트 가능한 CLI 시뮬레이터 범위를 기준으로 정리한다.

본 문서는 구현 방법이 아니라 시스템이 제공해야 할 외부 동작, 인터페이스, 기능 요구사항, 비기능 요구사항, 검증 기준을 명세한다.

1.2 Scope

대상 시스템은 RVC 자동 청소 제어 로직과 이를 검증하기 위한 CLI 그리드 명령을 생성한다.

범위에 포함하는 항목은 다음과 같다.

- 자동 청소 시작 및 중지
- 전방 장애물 interrupt 처리
- 좌측, 우측, 먼지 센서의 periodic sampling
- 장애물 회피 및 막힌 영역 탈출
- 먼지 감지 시 boost 청소
- CLI 시뮬레이터를 통한 요구사항 검증

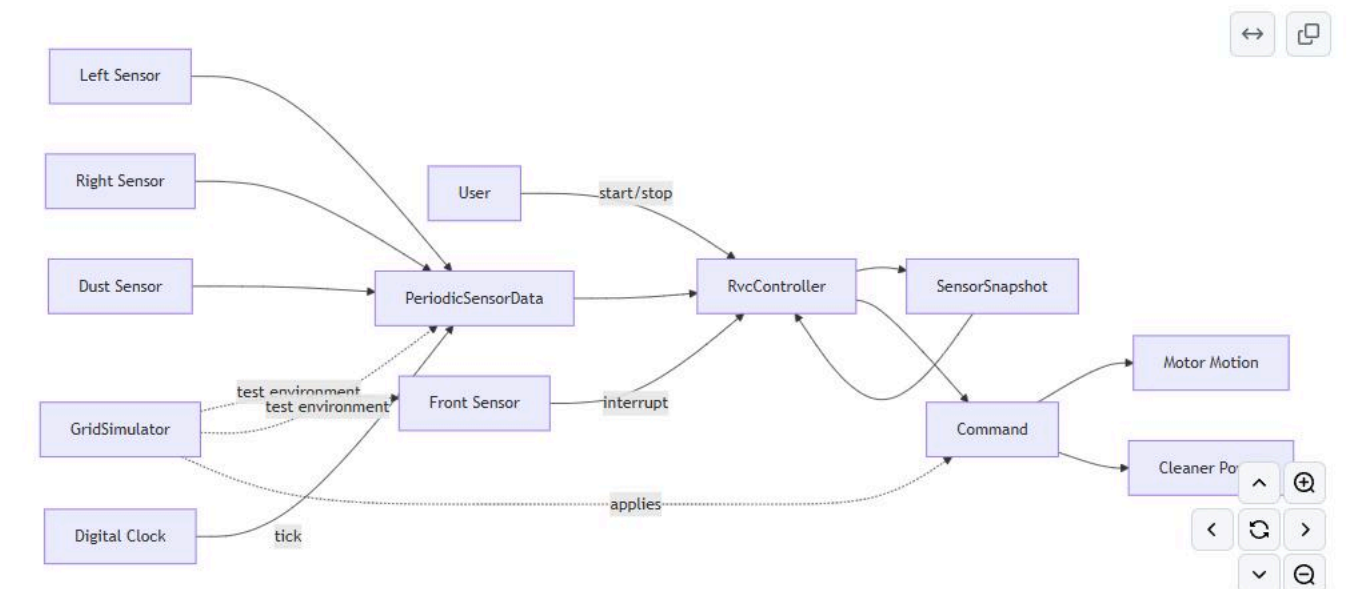
범위에 포함하지 않는 항목은 다음과 같다.

- 실제 모터 드라이버
- 실제 센서 하드웨어
- 배터리, 충전, 전원 관리
- 사용자 앱, 네트워크, 영구 저장소
- 물리적 모터 가속도, 센서 노이즈, 지도 작성 알고리즘

4. Design Context

4.1 System Boundary

RVC Control SW는 sensor 입력을 읽고 actuator 명령을 결정하는 controller 중심 구조이다. 전방 장애물은 interrupt로 controller에 기록되고, 좌측/우측/먼지 sensor는 제어 tick마다 periodic 데이터로 전달된다. controller는 두 입력 흐름을 SensorSnapshot 으로 결합한 뒤 Command 를 생성한다.



코드 작성 능력

- 처음부터 만드는 것 보단, PLAN 모드로 계획을 잡고 시작했을 때 결과물이 훨씬 좋았음
- 코드 작성은 쉽고 빠르지만, 일부 직관적이지 않는 부분도 있었음 → 주석을 달거나 문서를 좀 더 보는 것으로 해결
 - 일반적으로 3 단계에 거쳐서 해결되는 문제 (양 옆 값 받아오고 → 값 검증하고 → 다음 판단 내리고)를 한 줄에 해결하려 하는 경향
- 다만, 앞의 Trouble Shooting과 같이 요구 사항을 빠뜨리는 경향도 있기 때문에 사람의 검증이 필요함

플랜

📄 🔄 🏠 🔍

RVC 제어 소프트웨어와 CLI 시뮬레이터 구현 계획

요약

- docs/rvc.pdf 의 4~6쪽 요구사항을 기준으로 RVC 자동 청소 제어 소프트웨어와 테스트용 CLI 격자 시뮬레이터를 C++로 구현한다.
- OOAD 산출물은 docs/ooad_analysis.md 에 정리하고, 코드에는 제어 로직과 시뮬레이터를 분리해 둔다.
- 유닛 테스트는 구글테스트로 작성하고, 시스템 테스트는 시뮬레이터 시나리오 실행 결과로 검증한다.

주요 구현

- CMake 기반 C++20 프로젝트를 만든다.
- 제어 소프트웨어는 센서, 모터, 청소기, 시계 입력을 추상 인터페이스로 두고 RvcController 가 매 틱마다 동작 명령 결정하게 한다.
- 기보 도자 그치.

SSD-05 번에서 loop를 돌 때 front is open이면 빠져나오는데, 후진 한 번 하면 바로 앞에 장애물이 없는 상태가 되므로 결국 앞으로 갔다 뒤로 가는 상황이 됨. 이 문제를 해결하기 위해선 좌/우만 보고 탈출하도록 해야 함.
위와 더불어 SD-05에서도 후진 로직을 변경해야 함
변경된 로직에 따라 실제 후진하는 코드 변경
후진하는 테스트 케이스 추가. Unit test, Simulator를 이용한 Test
코드가 수정되면 문서도 그에 맞게 함께 수정

3m 34s 동안 작업 >

수정 완료했습니다.

핵심 변경은 `RvcController.cpp (line 54)`에서 Escaping 상태의 탈출 조건을 front/left/right 전체가 아니라 left/right 만 보도록 바꾼 것입니다. 이제 후진 직후 전방이 열려도 좌/우가 모두 막혀 있으면 계속 Backward 를 명령하고, 좌/우 중 하나가 열릴 때만 회전합니다.

추가한 검증:

- Unit test: `controller_tests.cpp (line 102)`
- Simulator test: `system_tests.cpp (line 71)`
- 문서 갱신: requirements, SSD-05, SD-05, class/domain diagram, traceability

검증 결과:

- `cmake --build build --config Debug` 통과
- `ctest --test-dir build -C Debug --output-on-failure` 통과, 14/14
- `scenarios/backward_escape2.rvc` 수동 실행에서도 tick 2 이후 `frontInterrupt=false` 인데 계속 후진하다가 side open에서 회전하는 것 확인했습니다.

코드 내용 분석

- 프롬프트를 통해 SOLID 원칙에 따라 코드를 작성해 달라고 요청했으나, RVC와 Simulator에 대해서만 원칙을 적용하였음.
- RVC 내부에 대해서는 적용하지 않아서 절차지향적인 코드가 작성되었다고 생각함.
- 다음 주차인 SOLID 분석 기간 동안 해당 부분을 수정해서 객체지향적인 코드를 작성해야한다고 느낌.
- Simulator와 RVC를 한번에 같이 만들어달라고 해서 생긴 문제점으로 생각됨.

8.2 SOLID Analysis

Principle	Application
SRP	<code>RvcController</code> 는 제어 결정만 담당하고, <code>Gridsimulator</code> 는 환경과 이동 적용만 담당한다.
OCP	sensor 입력은 <code>PeriodicSensorData</code> 와 interrupt API로 추상화되어 새 sensor 추가 시 controller 확장이 가능하다.
LSP	simulator와 실제 하드웨어 어댑터는 같은 <code>Command</code> 의미를 따르므로 대체 가능하다.
ISP	controller의 public interface는 시작, 중지, interrupt, tick, 판단에 필요한 작은 operation으로 분리된다.
DIP	controller는 concrete simulator나 hardware에 의존하지 않고 값 객체와 추상 command에만 의존한다.

테스트 작성 능력

- 테스트 작성 능력은 월등히 뛰어남
- 짧은 시간 안에 많은 테스트를 만들어 냄
- 다만, 테스트 개수가 많을 뿐이지 실제로 유용한 테스트인지는 역시 사람이 검증해야 하는 문제가 있음
 - 시뮬레이터 검증 테스트 부분이 쓸데 없이 많다고 생각하긴 함
- 시가 테스트에 실패하면, 코드를 고치는 게 아니라 테스트가 통과하도록 고치려는 경향이 있음
- 위 내용에 따라, 테스트 불통 시 인간이 코드를 직접 고치거나, TDD를 수행하는 것도 좋은 방법으로 생각함

3. 컨트롤러 유닛 테스트 결과

컨트롤러 유닛 테스트는 결정적인 센서 입력과 인터럽트 입력을 사용해 RvcController 의 상태 전이, 회피 판단, 탈출 판단, 먼지 boost 제어 규칙을 검증한다.

번호	테스트 케이스	검증 대상	결과
1	RvcControllerTest.ControllerMovesForwardWhenPathIsClear	전방 경로가 열려 있을 때 청소를 시작하고 전진하는지 검증	Passed
2	RvcControllerTest.IdleControllerReturnsStopAndIgnoresFrontInterrupt	idle 상태에서 전방 인터럽트와 먼지 감지를 무시하고 정지/꺼짐 상태를 유지하는지 검증	Passed
3	RvcControllerTest.StopCleaningReturnsStopAndOff	청소 중지 요청 시 모터 정지와 클리너 끄기 명령을 반환하는지 검증	Passed
4	RvcControllerTest.FrontInterruptTriggersImmediateAvoidance	전방 장애물 인터럽트가 즉시 회피 동작으로 이어지고 클리너가 꺼지는지 검증	Passed
5	RvcControllerTest.FrontInterruptIsConsumedAfterOneTick	전방 인터럽트가 1 tick 후 소비되어 정상 전진으로 복귀하는지 검증	Passed
6	RvcControllerTest.TurnsTowardOpenSide	열린 측면 방향으로 회전하고 회피 중 클리너가 꺼지는지 검증	Passed
7	RvcControllerTest.AlternatesWhenBothSidesAreOpen	좌우가 모두 열려 있을 때 회전 방향을 번갈아 선택하는지 검증	Passed
8	RvcControllerTest.AlternationPersistsAcrossClearTicks		
9	RvcControllerTest.AllBlockedEntersEscapingAndKeepsBackingUp		

테스트 케이스	결과
RvcControllerTest (17)	< 1ms
AllBlockedEntersEscapingAndKeepsBackingUp	< 1ms GoogleTes...
AllBlockedWithDustKeepsCleanerOffButPreservesBoostBudget	< 1ms GoogleTes...
AlternatesWhenBothSidesAreOpen	< 1ms GoogleTes...
AlternationPersistsAcrossClearTicks	< 1ms GoogleTes...
AvoidanceOutputStaysOffWhileBoostStateIsMaintained	< 1ms GoogleTes...
ControllerMovesForwardWhenPathIsClear	< 1ms GoogleTes...
DustBoostLastsConfiguredTicks	< 1ms GoogleTes...
DustDetectionRefreshesBoostBudget	< 1ms GoogleTes...
EscapingIgnoresOpenFrontUntilSideOpens	< 1ms GoogleTes...
EscapingTurnsTowardRightSideExit	< 1ms GoogleTes...
EscapingUsesAlternationWhenBothSidesOpen	< 1ms GoogleTes...
FrontInterruptIsConsumedAfterOneTick	< 1ms GoogleTes...
FrontInterruptTriggersImmediateAvoidance	< 1ms GoogleTes...
IdleControllerReturnsStopAndIgnoresFrontInterrupt	< 1ms GoogleTes...
StopCleaningReturnsStopAndOff	< 1ms GoogleTes...
TurnsTowardOpenSide	< 1ms GoogleTes...
ZeroTickBoostConfigurationDoesNotBoostOnDust	< 1ms GoogleTes...