

TEAM 2

OOAD *with AI*

강병완 202211248

강현준 202211251

박 완 202211301

정민수 202211365

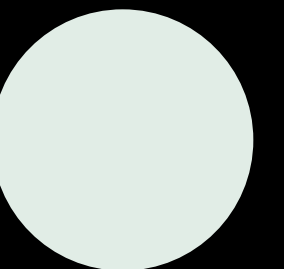
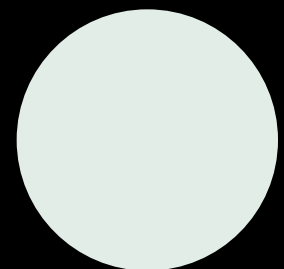












Table of Content







- 프로젝트 전체 개요 및 진행
- Inception
- Elaboration (OOA, OOD)
- Construction (OOI)

프로젝트 전체 개요

SRS

-  [예비 요구사항](#)
-  [용어집 \(Glossary\)](#)
-  [Vision](#)
-  [Use-Case Model](#)
-  [Use-Case](#)
-  [System Sequence Diagram](#)
-  [Supplementary Specification](#)
-  [Functional Requirements](#)
-  [Non-Functional Requirements](#)
-  [Traceability Table](#)

SDD

-  [Domain Model](#)
-  [Domain Model \(Mermaid\)](#)
-  [Design Model](#)
-  [SW Architecture Document](#)
-  [Sequence Diagram](#)
-  [Class Diagram](#)




Unit Test

-  [Unit Test](#)

System Test

-  [System Test](#)

기타

-  [Architecture Decisions \(2026-05-17\)](#)
-  [Failures and Resolutions \(2026-05-17\)](#)
-  [프로젝트 설명](#)

프로젝트 진행

Inception

- Glossary
- Vision
- Use-Case Casual Format
- Functional Requirements
- Non-Functional Requirements



Elaboration

OOA

- System Sequence Diagram
- System Operation
- Domain Model

OOD

- Sequence Diagram
- Class Diagram
- Traceability Table

Construction

OOI

- Program (UI + Simulator + RVC Control SW)
- Unit Test
- System Test
- Static Code Analysis



- 각 단계의 Output은 프로젝트 하위 docs 디렉토리에 저장
- Claude가 필요한 docs를 읽고 작성 및 구현
- Claude가 Mermaid를 이용해 Diagram을 작성
- 프로젝트 마무리에 Notion MCP를 활용하여 docs의 문서를 모두 Notion에 아카이빙

Claude Code

Vibe Coding Tool

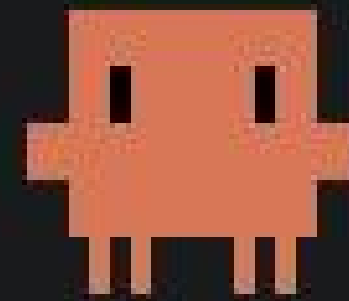
- Claude Code CLI

사용 모델 / effort

- Claude Sonnet 4.6 with medium effort

시간

- 요구사항 분석 ~ Static Analysis까지 약 3시간 소요



Claude Code v2.1.146

Sonnet 4.6 with medium effort · Claude Pro
~/OOADProject

Initial Setup

The RVC Control SW

Preliminary Requirements for RVC SW Controller

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

Future pr Extended Requirements to Consider

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

DFD Level 0 from SASD

- '<-' means input direction.
- '->' means output direction.

```
RVC Control SW
<- Front Sensor Input - Front Sensor
<- Left Sensor Input  - Left Sensor
<- Right Sensor Input - Right Sensor
<- Dust Sensor Input  - Dust Sensor
<- Tick                - Digital Clock
-> Direction           - Motor
-> Clean               - Cleaner
```

Input/Output Event	Description	Format / Type
Front Sensor Input	Detects obstacles in front of the RVC	True / False, interrupt
Left Sensor Input	Detects obstacles in the left side of the RVC periodically.	True / False, Periodic
Right Sensor Input	Detects obstacles in the right side of the RVC periodically.	True / False, Periodic
Dust Sensor Input	Detects dust on the floor periodically.	True / False, Periodic
Direction	Direction commands to the motor (go forward / turn left with an angle / turn right with an angle)	Forward / Backward / Left / Right
Clean	Turn off / Turn on / Power Up	On / Off / Up

Initial Setup

AGENTS.md

Project Outline

- Goal : Use OOAD based on UP and Make RVC Control SW, RVC UI and RVC Simulator
- Outputs : SRS, SDD, Code, UT, ST, SA, Simulator
- Language : C++17
- Environment : Ubuntu 24.04 WSL2
- Development Method : OOAD based on UP(Unified Process), V&V and TDD

CI / CD

Development

- IDE : Visual Studio Code
- Build Management : CMake
- Unit Testing : Google Test
- Version Control : Github
- Requirements Management : Notion, Jira

CI

- Continuous Integration : Github Action
- Build Management : CMake
- Unit Testing : Google Test
- Static Code Analysis : clang-tidy
- Coverage : gcovr

etc.

- Integrated Testing : Using Simulator

Directory

```
1 00ADProject/  
2 |  
3 |— AGENTS.md  
4 |— .clang-format  
5 |— .clang-tidy  
6 |— .gitignore  
7 |— docs/  
8 | |— requirements/  
9 | |— design/  
10 |— skills/  
11 | |— conventions/  
12 |— src/  
13 | |— CMakeList.txt  
14 |— test/  
15
```

Strict Prohibition (Highest Priority)

- Adding external packages or libraries
- Deleting any file or directory
- Changing any

프롬프트

- Initialize your settings based on AGENTS.md

Initial Setup

CLAUDE.md

This file provides guidance to Claude Code (claude.ai/code) when working with code in this repository.

Project Overview

This project builds an RVC (Robot Vacuum Cleaner) system using OOAD based on the Unified Process (UP). It produces three components: **RVC Control SW**, **RVC UI**, and **RVC Simulator**. Development follows UP phases sequentially: Inception → Elaboration → Construction.

- Language: C++17
- Environment: Ubuntu 24.04 WSL2
- Methodology: OOAD / UP, V&V, TDD

Build Commands

```
# Configure (from project root)
cmake -S src -B build

# Build
cmake --build build

# Run all tests
cd build && ctest --output-on-failure

# Run a single test binary
./build/<test-binary-name>

# Static analysis
clang-tidy src/**/*.*.cpp -- -std=c++17

# Coverage (after building with coverage flags)
gcovr -r . --html --html-details -o coverage/index.html
```

Strict Prohibitions (Highest Priority)

- **Do not add external packages or libraries** beyond what is already present.
- **Do not delete any file or directory.**

UP Phase Outputs

Each phase must produce specific artifacts before advancing:

Phase	Artifacts
Inception	Use-Case Model, Vision, Supplementary Specification, Non-Functional Requirements, Glossary
Elaboration	Domain Model, Design Model, SW Architecture Document, refined Inception outputs
Construction	RVC Control SW, RVC UI, RVC Simulator

Each iteration must implement all three components (Control SW, UI, Simulator) and pass integration testing via the Simulator.

Initial Setup

Architecture

The RVC Control SW uses a **4-layer architecture**: Application → Domain → Interface → HAL. Full detail is in [docs/design/sw-architecture.md](#).

Key design decisions:

- `RvcController` is the central orchestrator with an explicit `RvcState` state machine (IDLE / CLEANING / AVOIDING_OBSTACLE / ESCAPING / INTENSIFYING).
- Navigation logic lives in `INavigationStrategy` / `DefaultNavigationStrategy` — injected into `RvcController` so it can be swapped (e.g., ML-based) without touching the controller.
- `FrontSensor` is interrupt-driven (`onInterrupt()` → `onFrontObstacleDetected()`); Left, Right, Dust sensors are polled each Tick via `detect()`.
- All hardware dependencies are injected via constructor (`ISensor*`, `IMotorController*`, `ICleanerController*`) — enables full Google Test isolation with mocks.

Enums: `Direction` {FORWARD, BACKWARD, LEFT, RIGHT, STOP} · `CleanPower` {OFF, ON, POWER_UP} · `RvcState`

Source Layout

```
src/
├─ interfaces/    # ISensor, IMotorController, ICleanerController, INavigationStrategy
├─ domain/       # SensorData, enums, DefaultNavigationStrategy
├─ hal/         # FrontSensor, LeftSensor, RightSensor, DustSensor
└─ app/         # RvcController, main.cpp
test/
├─ domain/      # DefaultNavigationStrategyTest
└─ app/        # RvcControllerTest
docs/
├─ requirements/ # Vision, Use-Case Model, Supplementary Spec, Glossary
└─ design/     # Domain Model, Design Model, SW Architecture
```

Naming

Construct	Style	Example
Class	PascalCase	<code>SensorController</code>
Interface	PascalCase with <code>I</code> prefix	<code>ISensor</code>
Abstract Class	PascalCase with <code>Abstract</code> prefix	<code>AbstractCleaner</code>
Method	camelCase	<code>detectObstacle()</code>
Variable (public)	snake_case	<code>front_sensor</code>
Variable (private)	snake_case with <code>_</code> prefix	<code>_speed</code>
Constant	UPPER_SNAKE_CASE	<code>MAX_SPEED</code>
Directory	kebab-case	<code>sensor-drivers/</code>

Rules

- Apply SOLID principles throughout.
- Boolean variables/methods use `is` or `has` prefix (e.g., `is_blocked`, `has_dust`).
- Use standard English plurals for collections: `sensors` not `sensor_list`.

Inception phase

프롬프트

- We will start this project based on UP. Let's begin Inception phase first. outputs of this phase will be properly saved in docs directory

결과물

- Glossary
- Vision
- Use-Case Casual Format
- Functional Requirements
- Non-Functional Requirements

Functional Requirement

FR ID	Description	Feature	Use Case
F-01	기본적으로 직진하며 청소. 장애물이 없으면 RVC가 Forward 상태를 유지하며 Cleaner는 On 상태를 유지한다.	자율 전진 Navigation	UC-01, UC-02
F-04	Dust Sensor가 먼지를 감지하면 Cleaner를 일시적으로 Power Up한다. INTENSIFY_DURATION 이후 정상 On 상태로 복귀한다. Navigation은 중단하지 않는다.	적응형 청소 강도	UC-05
F-03	전방, 좌측, 우측 모두 차단된 경우(Surrounded State), RVC가 후진 후 가능한 측면으로 회전하여 전진을 재개한다.	포워 장애물 탈출	UC-04
F-02	전방만 차단되고 좌측 또는 우측이 개방된 경우, RVC가 정지 후 개방된 측면으로 회전하여 전진 navigation을 재개한다.	단일 측면 장애물 회피	UC-03

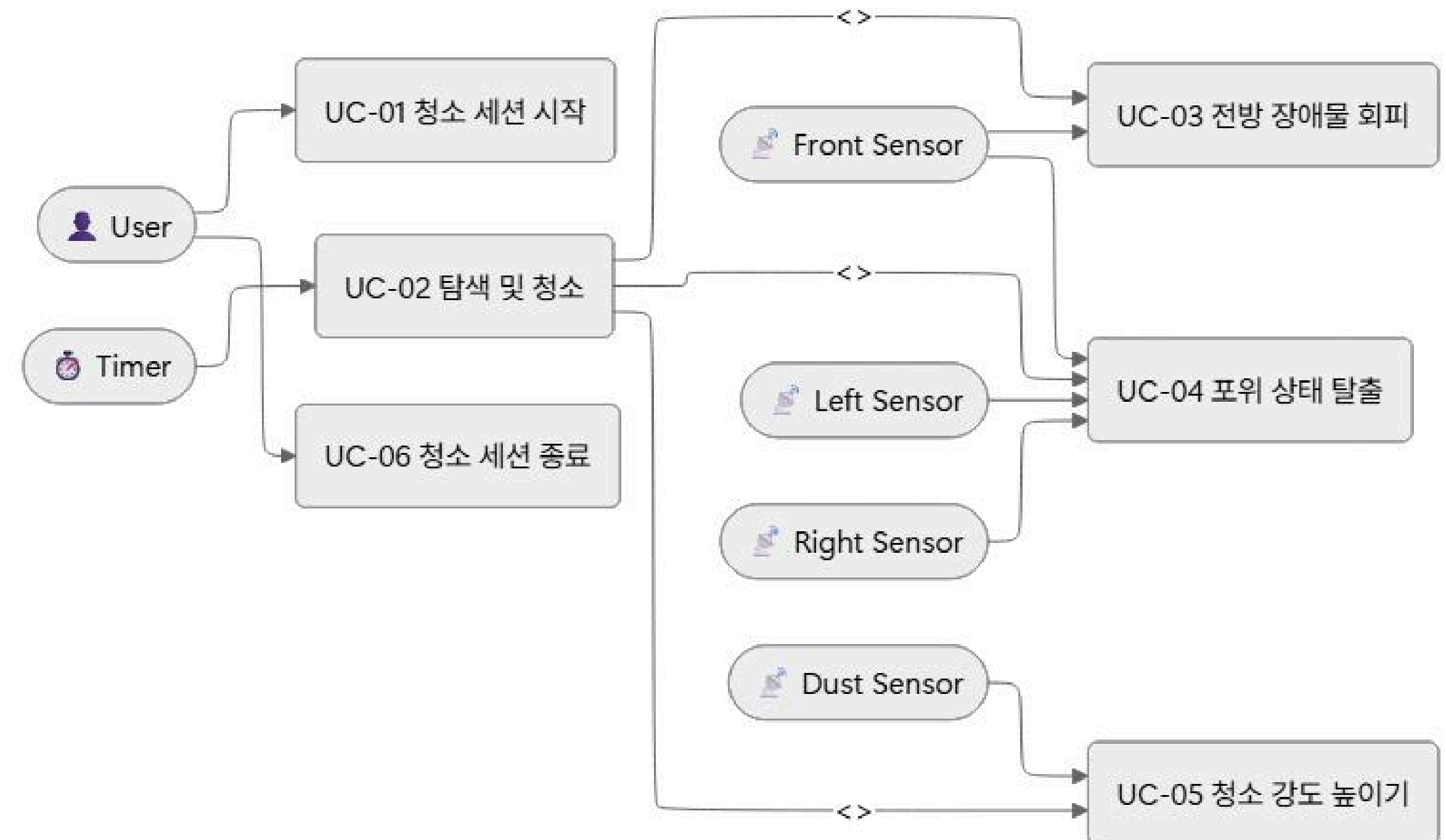
Non-Functional Requirement

Aa Req ID	Category	Requirement
DC-01	Design Constraint	구현 언어: C++17.
DC-02	Design Constraint	외부 라이브러리나 패키지를 도입할 수 없다.
DC-03	Design Constraint	모든 모듈은 대응하는 Google Test 단위 테스트를 가져야 한다.
DC-04	Design Constraint	코드는 프로젝트 정의 규칙 세트로 clang-tidy 정적 분석을 통과해야 한다.
DC-05	Design Constraint	HW 수준 제어(전기 신호, 레지스터 접근)는 범위 밖이며, SW는 추상화된 I/O 이벤트에서만 동작한다.
FUNC-01	Functionality	시스템은 Front Sensor 입력을 interrupt로 처리해야 한다 (polling 불가). 전방 장애물에 즉각 응답을 보장한다.
FUNC-02	Functionality	Left, Right, Dust Sensor 입력은 매 Timer Tick마다 평가된다 (주기적 polling).
FUNC-03	Functionality	Motor direction 명령은 상호 배타적이다; 한 번에 하나의 방향만 활성화된다.
FUNC-04	Functionality	Cleaner 전력 상태는 상호 배타적이다: Off, On, 또는 Power Up.
PERF-01	Performance	주기적 센서 평가(Left, Right, Dust)는 단일 Tick 간격 내에 완료되어야 한다.
PERF-02	Performance	강화 청소 지속 시간(UC-05)은 인라인 하드코딩이 아닌 시스템 상수로 설정 가능해야 한다.
REL-01	Reliability	시스템은 하나의 처리 사이클 내에 Front Sensor interrupt에 응답해야 한다.
REL-02	Reliability	시스템은 충돌하는 motor 명령(예: Forward와 Backward 동시)을 발행해서는 안 된다.
REL-03	Reliability	센서 상태를 결정할 수 없으면, 시스템은 safe state(Motor: Stop, Cleaner: Off)로 기본 설정해야 한다.
SUPP-01	Supportability	설계는 기존 센서 처리 로직을 수정하지 않고도 추가 센서 타입을 통합할 수 있어야 한다 (Open/Closed Principle).
SUPP-02	Supportability	Navigation 로직은 향후 navigation 알고리즘 교체(예: ML 기반)를 위해 센서 읽기 로직과 분리되어야 한다.
SUPP-03	Supportability	시스템은 핵심 제어 로직 변경 없이 향후 모바일 앱 통신을 위한 정의된 인터페이스 경계를 노출해야 한다.

Use case Diagram

1. Actor

Actor	타입	설명
User	Primary	청소 세션을 시작하고 종료한다
Front Sensor	External System	정면의 장애물을 감지; interrupt 방식
Left Sensor	External System	좌측의 장애물을 감지; 주기적 (periodic)
Right Sensor	External System	우측의 장애물을 감지; 주기적 (periodic)
Dust Sensor	External System	바닥의 먼지를 감지; 주기적(periodic)
Timer	External System	주기적 동작을 구동하는 주기적 Tick 신호를 제공



Use case

UC-01: 청소 세션 시작

항목	내용
ID	UC-01
이름	청소 세션 시작
Primary Actor	User
간략 설명	사용자가 청소 세션을 시작한다; RVC가 전진하며 청소를 시작한다.
사전 조건	RVC 전원이 켜져 있고 idle 상태다.
사후 조건	RVC가 전진하며 청소 중인 활성 상태다.

주요 성공 시나리오:

1. User가 RVC에 시작 명령을 내린다.
2. 시스템이 Cleaner를 활성화한다 (On).
3. 시스템이 Motor에 Forward 명령을 내린다.
4. 시스템이 활성 청소 루프(UC-02)에 진입한다.

UC-02: 탐색 및 청소

항목	내용
ID	UC-02
이름	탐색 및 청소
Primary Actor	Timer
간략 설명	매 Tick마다 시스템이 센서 상태를 평가하고 다음 navigation 및 청소 동작을 결정한다.
사전 조건	청소 세션이 활성 상태다 (UC-01 완료).
사후 조건	현재 Tick에 대한 Motor direction과 cleaner 상태가 업데이트된다.

주요 성공 시나리오 (장애물 없음, 먼지 없음):

1. Timer가 Tick을 발생시킨다.
2. 시스템이 Left Sensor, Right Sensor, Dust Sensor 상태를 읽는다.
3. 모든 센서가 False를 보고한다 (장애물 없음, 먼지 없음).
4. 시스템이 Motor 명령을 Forward로 유지한다.
5. 시스템이 Cleaner 명령을 On으로 유지한다.

Use case

UC-03: 전방 장애물 회피

항목	내용
ID	UC-03
이름	전방 장애물 회피
Primary Actor	Front Sensor
간략 설명	전방 장애물이 감지되고 RVC가 완전히 포위되지 않은 경우, RVC가 정지하고 개방된 측면으로 회전한 뒤 전진 navigation을 재개한다.
사전 조건	청소 세션이 활성 상태. Front Sensor가 True를 발생시킴. 좌측 또는 우측이 차단되지 않음.
사후 조건	RVC가 새로운 방향으로 전진 중이며, Cleaner는 On 상태를 유지한다.

주요 성공 시나리오:

1. Front Sensor가 True와 함께 interrupt를 트리거한다.
2. 시스템이 Motor에 Stop 명령을 내린다.
3. 시스템이 Left Sensor와 Right Sensor를 읽는다.
4. 최소 한 측면이 개방되어 있다 (Left = False 또는 Right = False).
5. 시스템이 개방된 측면(Left 또는 Right)을 선택하고 Motor에 해당 방향으로 회전 명령을 내린다.
6. 시스템이 Motor에 Forward 명령을 내린다.
7. Cleaner는 On 상태를 유지한다.

대안 흐름 — 양쪽 모두 차단:

- 4단계: Left = True AND Right = True → UC-04로 이전.

UC-04: 포위 상태 탈출

항목	내용
ID	UC-04
이름	포위 상태 탈출
Primary Actor	Front Sensor, Left Sensor, Right Sensor
간략 설명	세 방향(전방, 좌측, 우측) 모두에서 장애물이 감지되면, RVC가 후진하고 가능한 측면으로 회전한 뒤 재개한다.
사전 조건	청소 세션이 활성 상태. Front = True AND Left = True AND Right = True.
사후 조건	RVC가 새로운 방향으로 전진 중이며, Cleaner는 On 상태를 유지한다.

주요 성공 시나리오:

1. 시스템이 Front = True AND Left = True AND Right = True를 감지한다.
2. 시스템이 Motor에 Backward 명령을 내린다.
3. 시스템이 회전 방향(Left 또는 Right)을 선택한다.
4. 시스템이 Motor에 선택한 방향으로 회전 명령을 내린다.
5. 시스템이 Motor에 Forward 명령을 내린다.
6. Cleaner는 On 상태를 유지한다.

Use case

UC-05: 청소 강도 높이기

항목	내용
ID	UC-05
이름	청소 강도 높이기
Primary Actor	Dust Sensor
간략 설명	Dust Sensor가 먼지를 감지하면 Cleaner 전력이 일시적으로 증가한다.
사전 조건	청소 세션이 활성화 상태. Dust Sensor = True.
사후 조건	강화 기간이 끝나면 Cleaner가 정상 전력(On)으로 복귀한다.

주요 성공 시나리오:

1. Dust Sensor가 True를 보고한다.
2. 시스템이 Cleaner에 Power Up 명령을 내린다.
3. 시스템이 UC-02에 따라 navigation을 계속한다.
4. 강화 지속 시간이 경과하면, 시스템이 Cleaner에 On(정상) 명령을 내린다.

UC-06: 청소 세션 종료

항목	내용
ID	UC-06
이름	청소 세션 종료
Primary Actor	User
간략 설명	사용자가 청소 세션을 종료한다; RVC가 이동을 멈추고 Cleaner를 끄다.
사전 조건	청소 세션이 활성화 상태.
사후 조건	RVC가 idle 상태; Motor 정지; Cleaner Off.

주요 성공 시나리오:

1. User가 RVC에 정지 명령을 내린다.
2. 시스템이 Motor에 Stop 명령을 내린다.
3. 시스템이 Cleaner에 Off 명령을 내린다.
4. 시스템이 활성화 청소 루프에서 나온다.

Elaboration phase

프롬프트

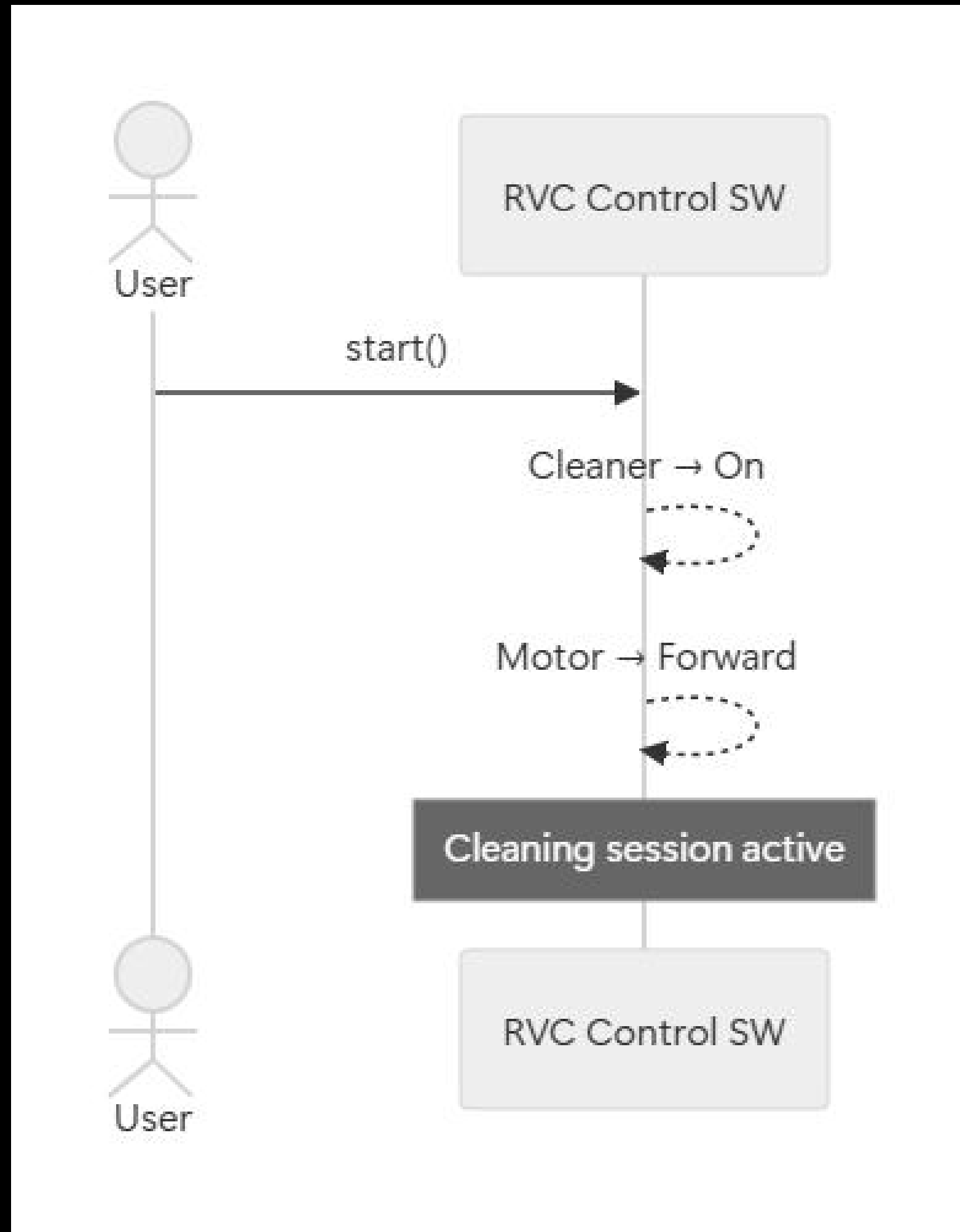
- continue Elaboration phase

결과물

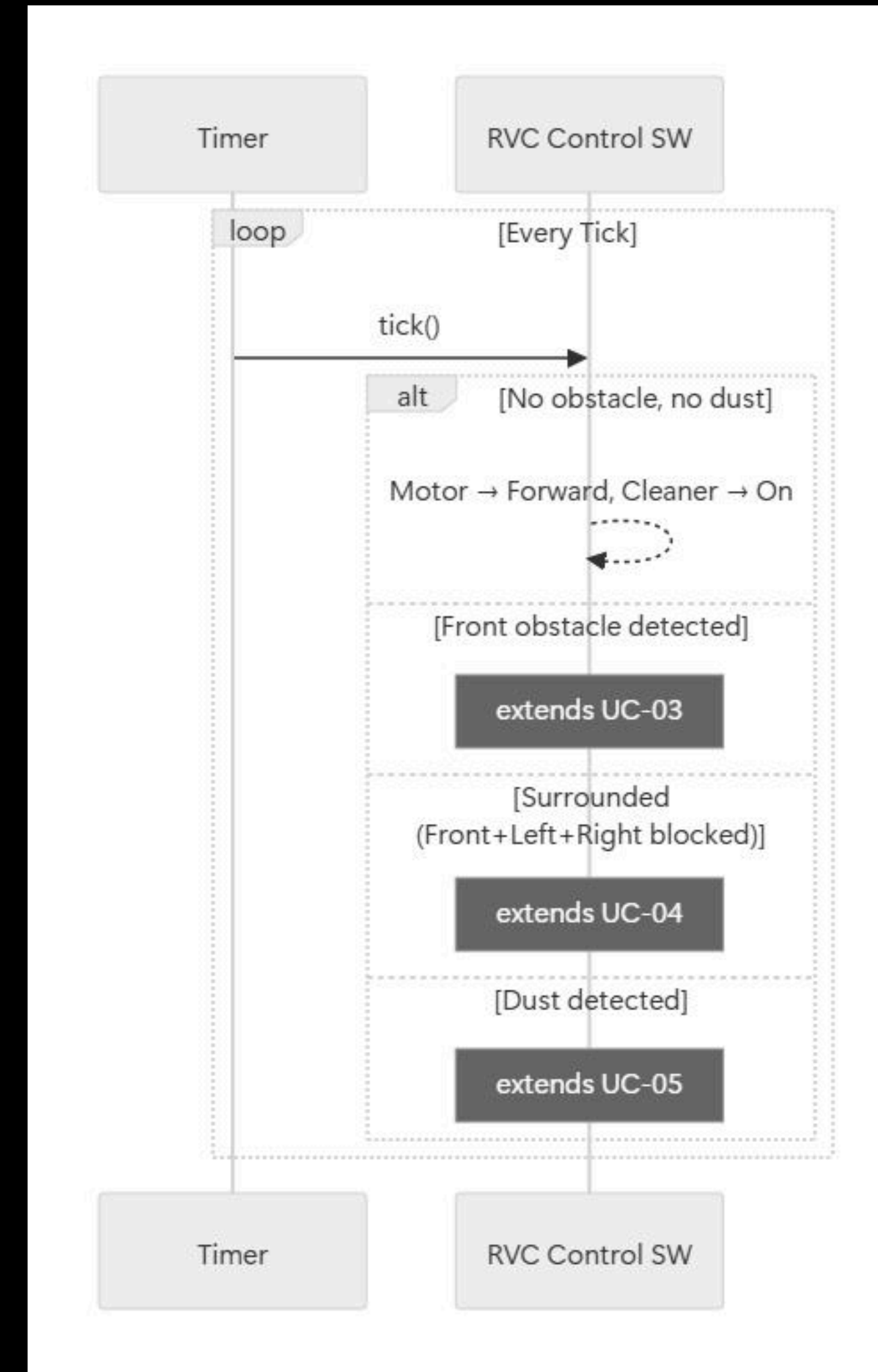
- OOA
 - System Sequence Diagram
 - System Operation
 - Domain Model
- OOD
 - Sequence Diagram
 - Class Diagram

System Sequence Diagram

UC - 01

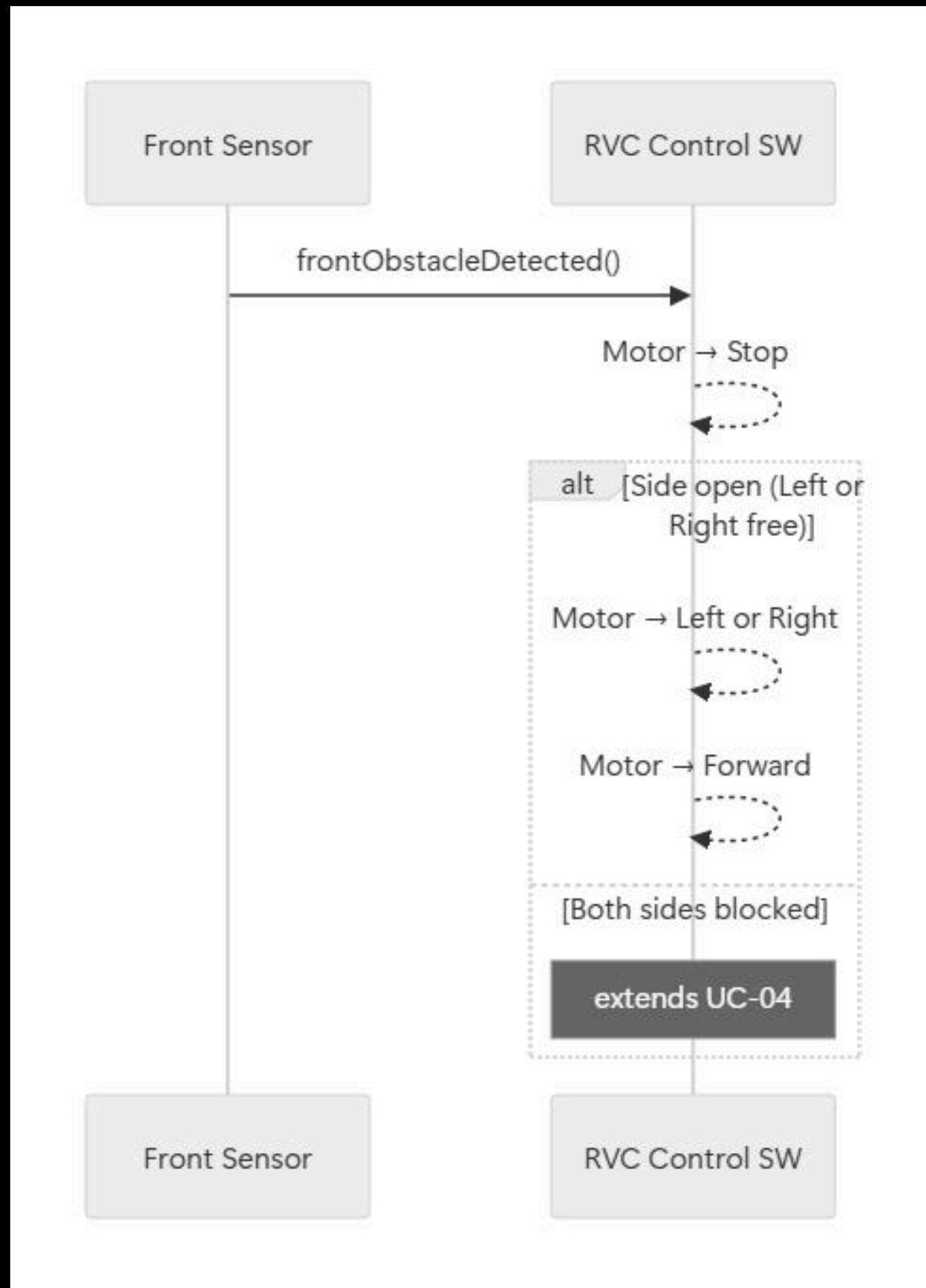


UC - 02

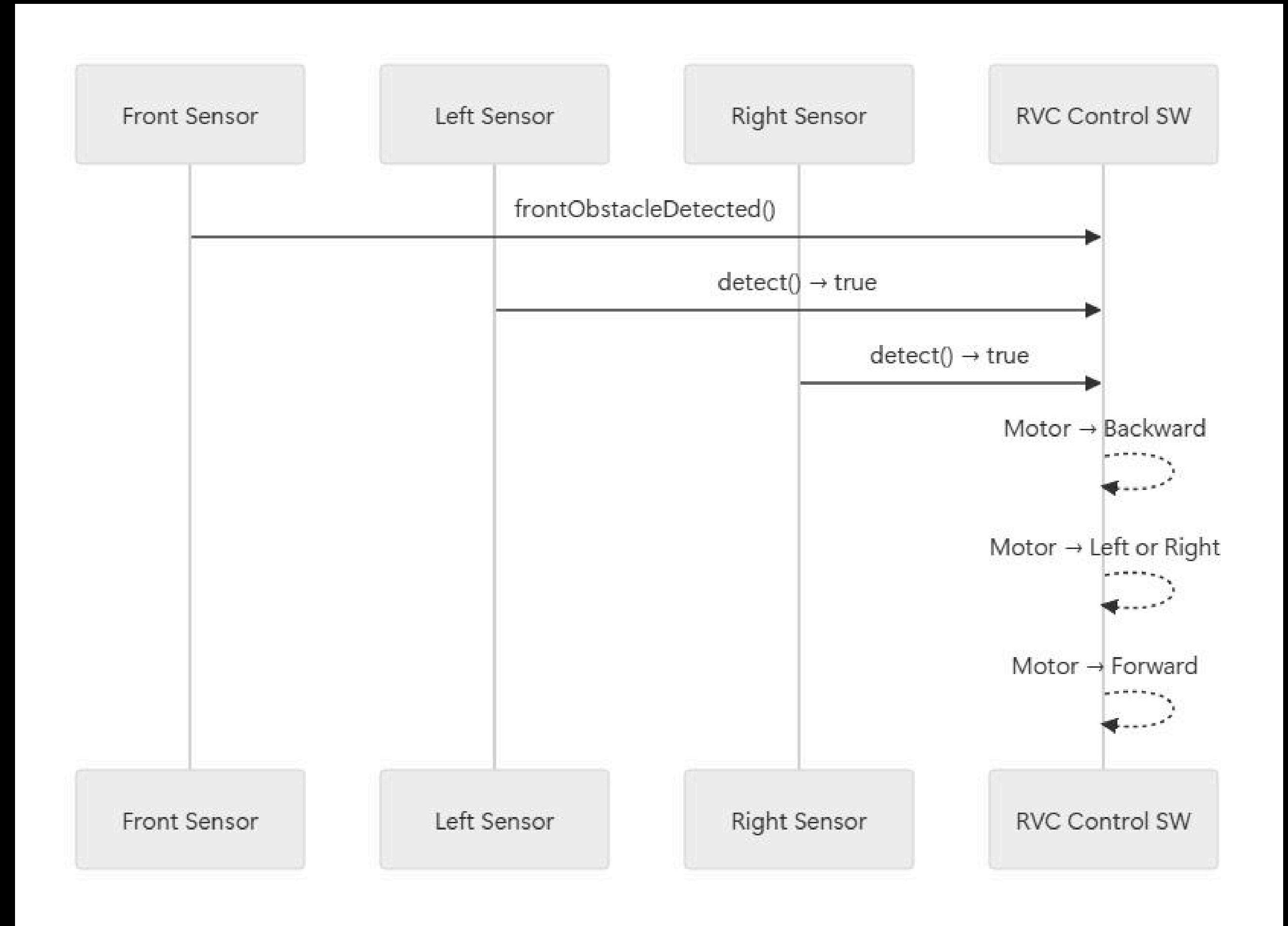


System Sequence Diagram

UC - 03

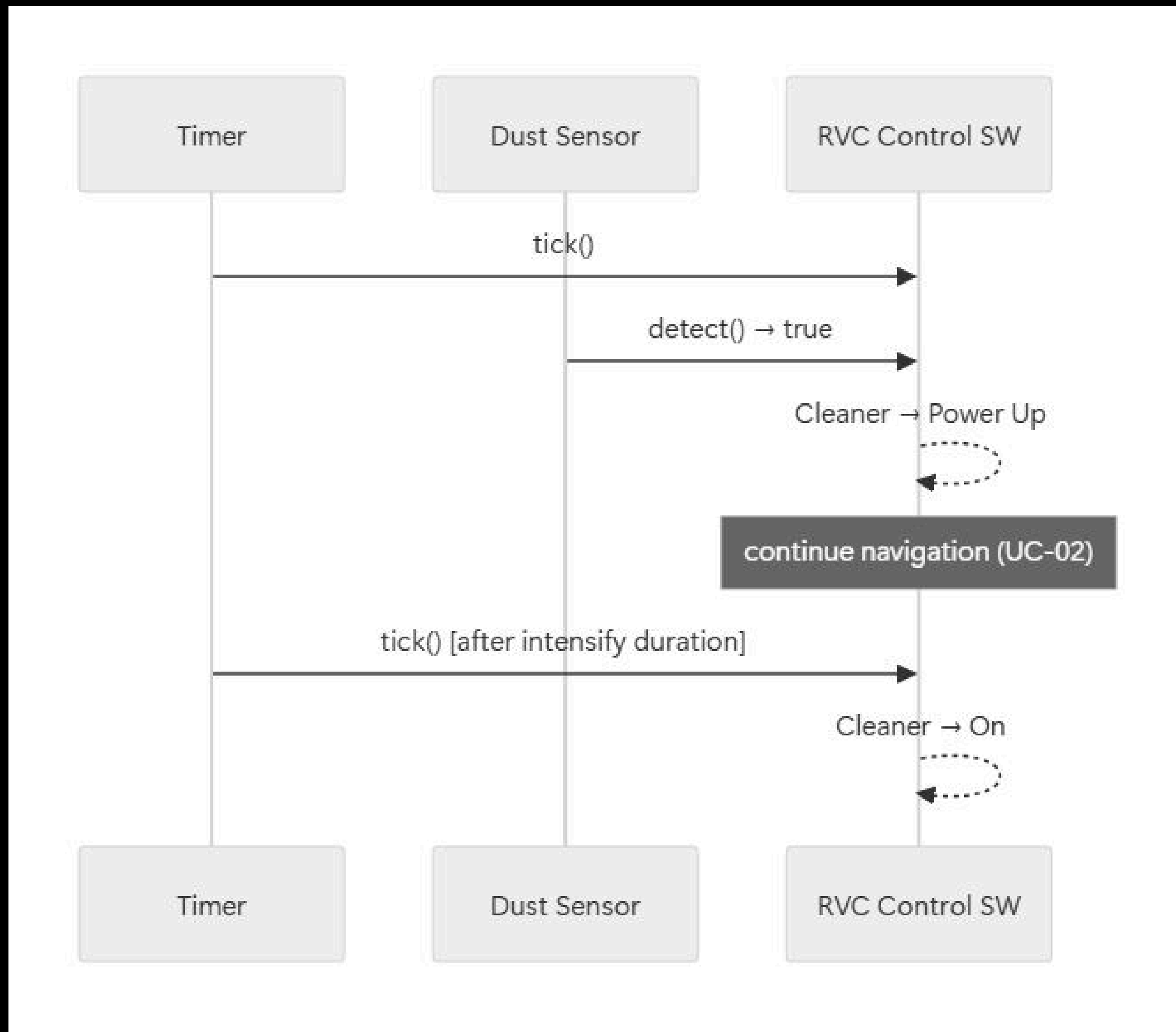


UC - 04

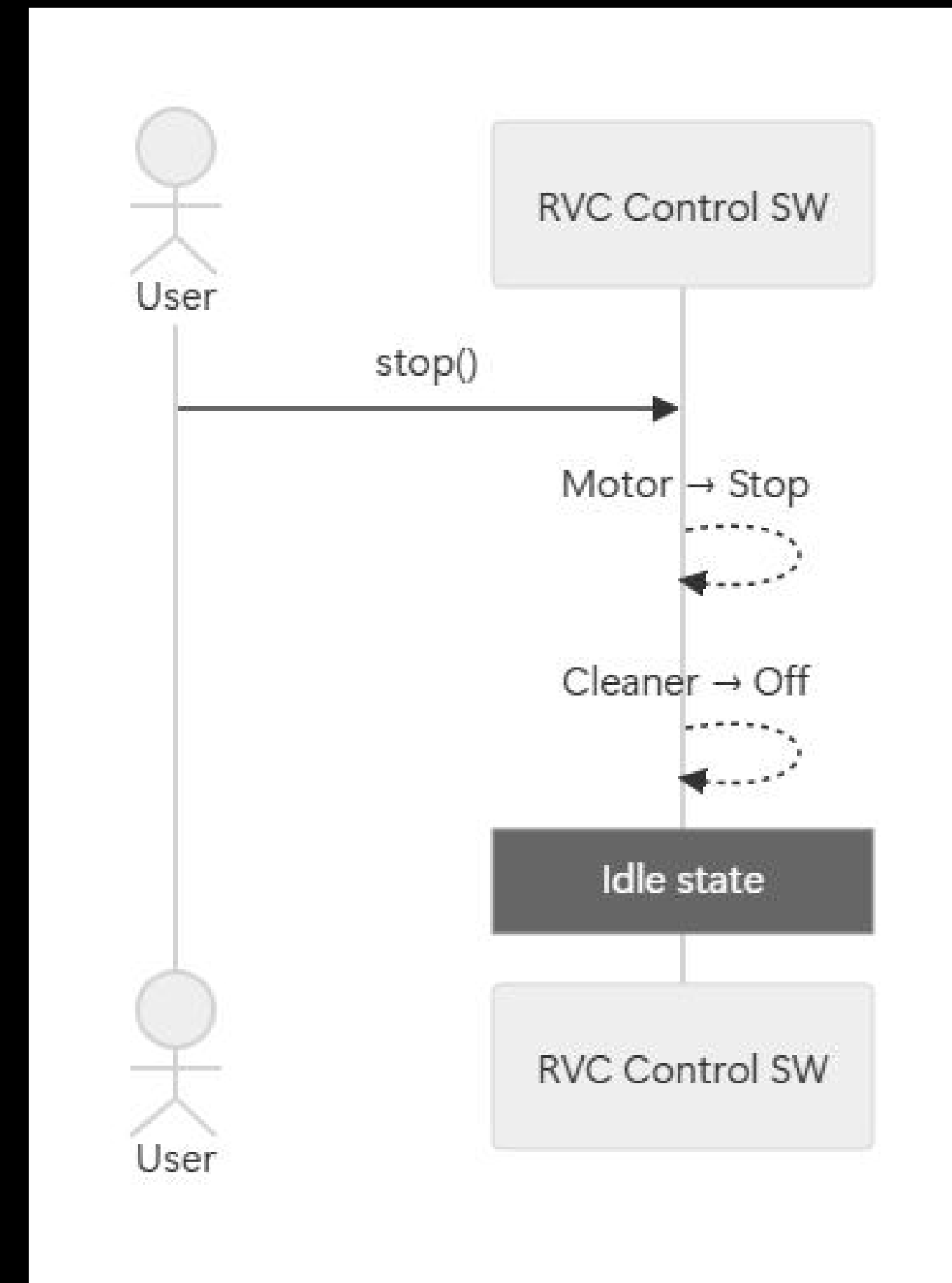


System Sequence Diagram

UC – 05



UC – 06

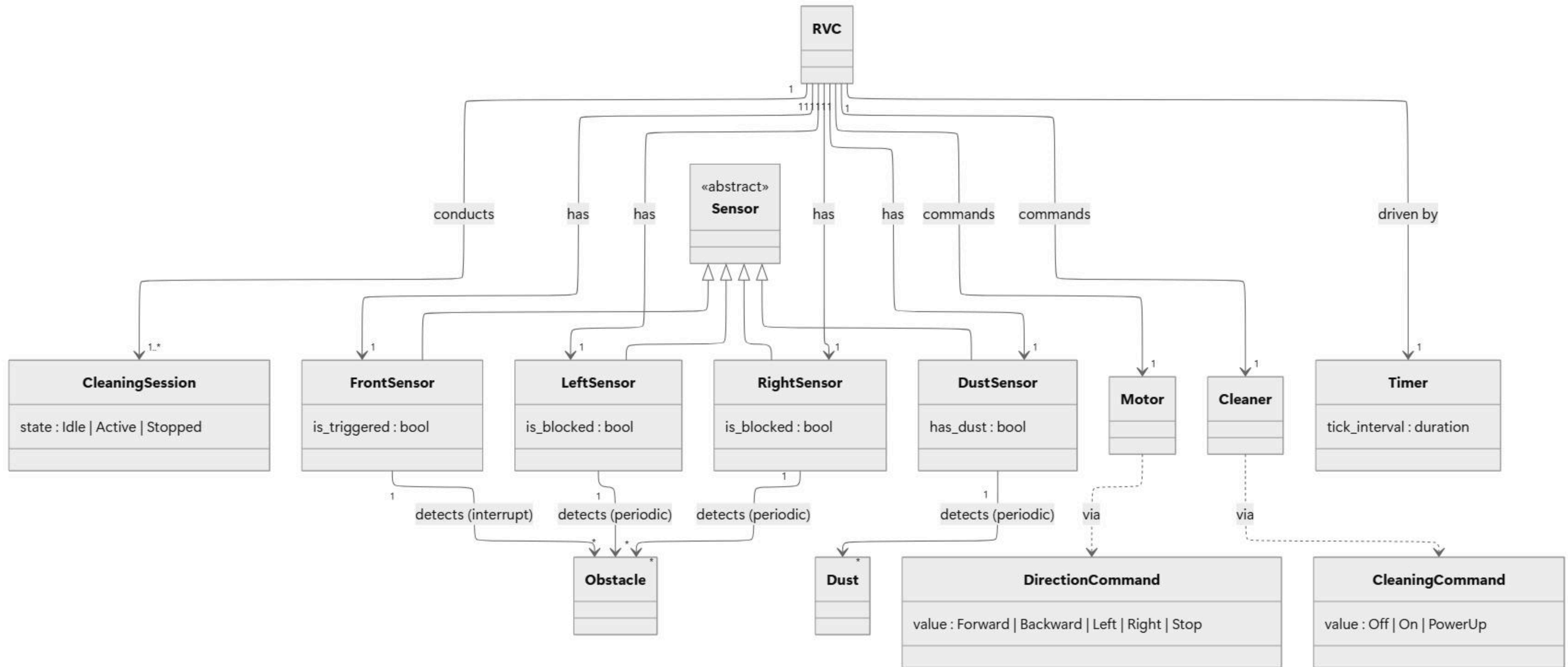


System Operation

{interface}
RVCSystem

+ startCleaning()
+ tick()
+ onFrontObstacleDetected()
+ stopCleaning()

Domain model

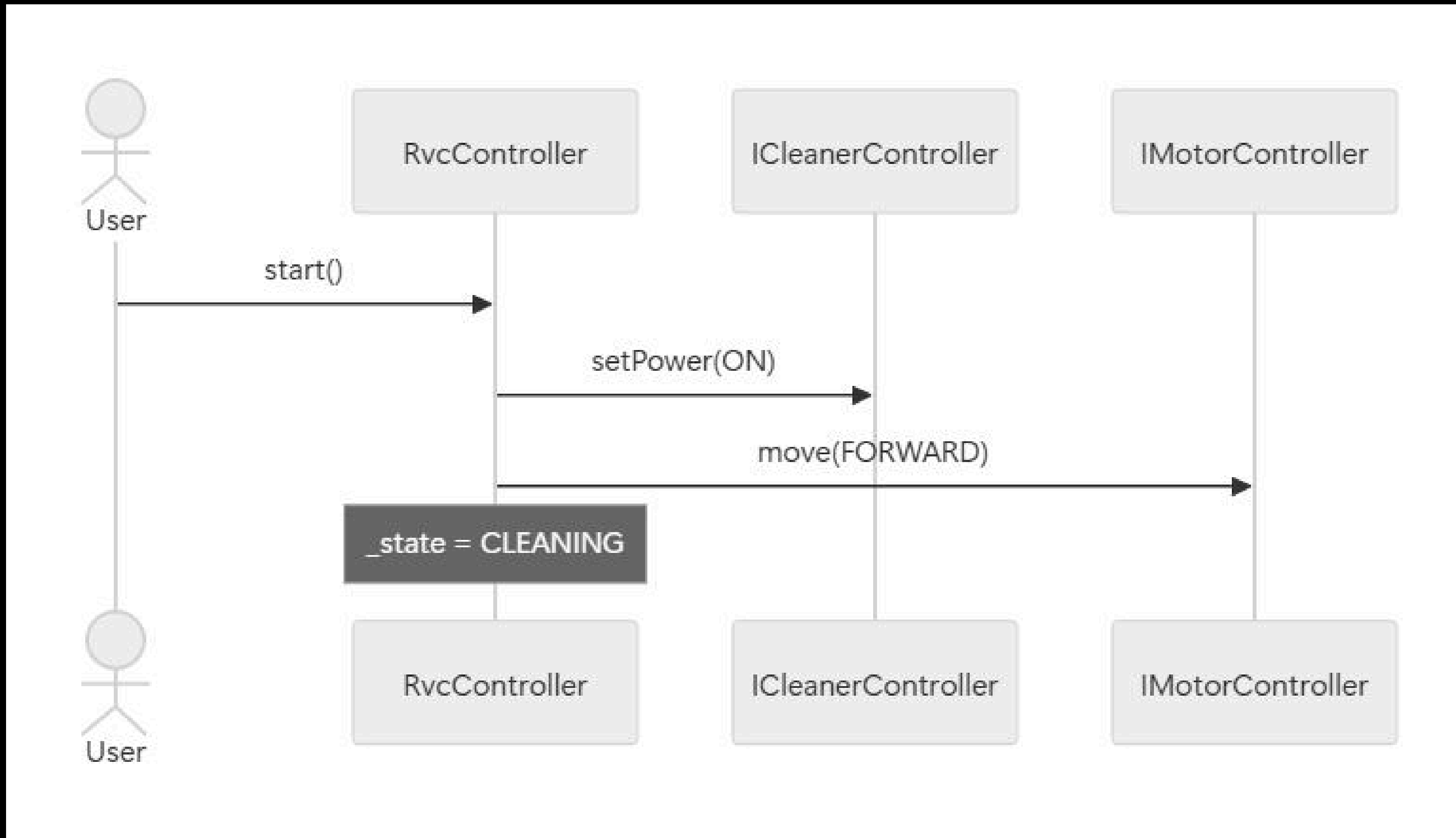


Sequence Diagram

Aa Name	ID	Primary Actor
☰ 청소 세션 시작	UC-01	User
☰ 탐색 및 청소	UC-02	Timer
☰ 전방 장애물 회피	UC-03	Front Sensor
☰ 포위 상태 탈출	UC-04	Left-Right Sensor
☰ 청소 강도 높이기	UC-05	Dust Sensor
☰ 청소 세션 종료	UC-06	User

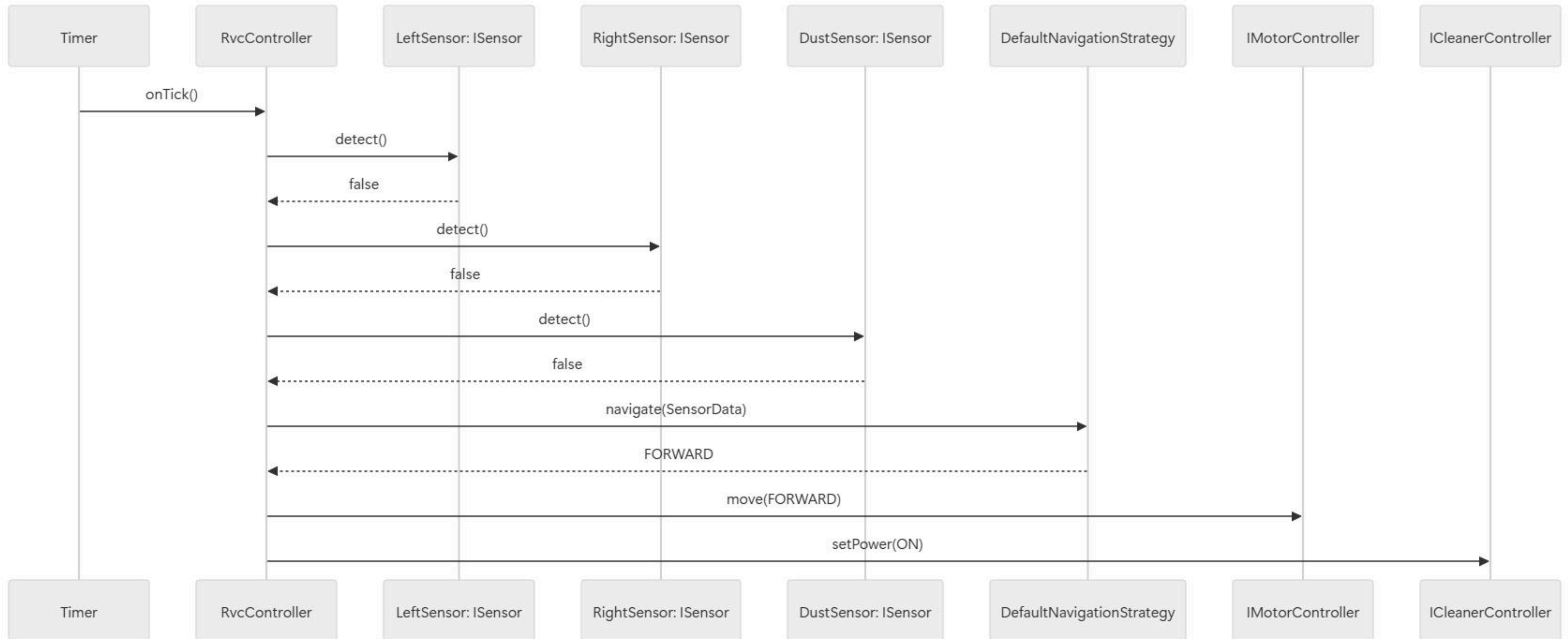
Sequence Diagram

UC#01 - 청소 세션 시작



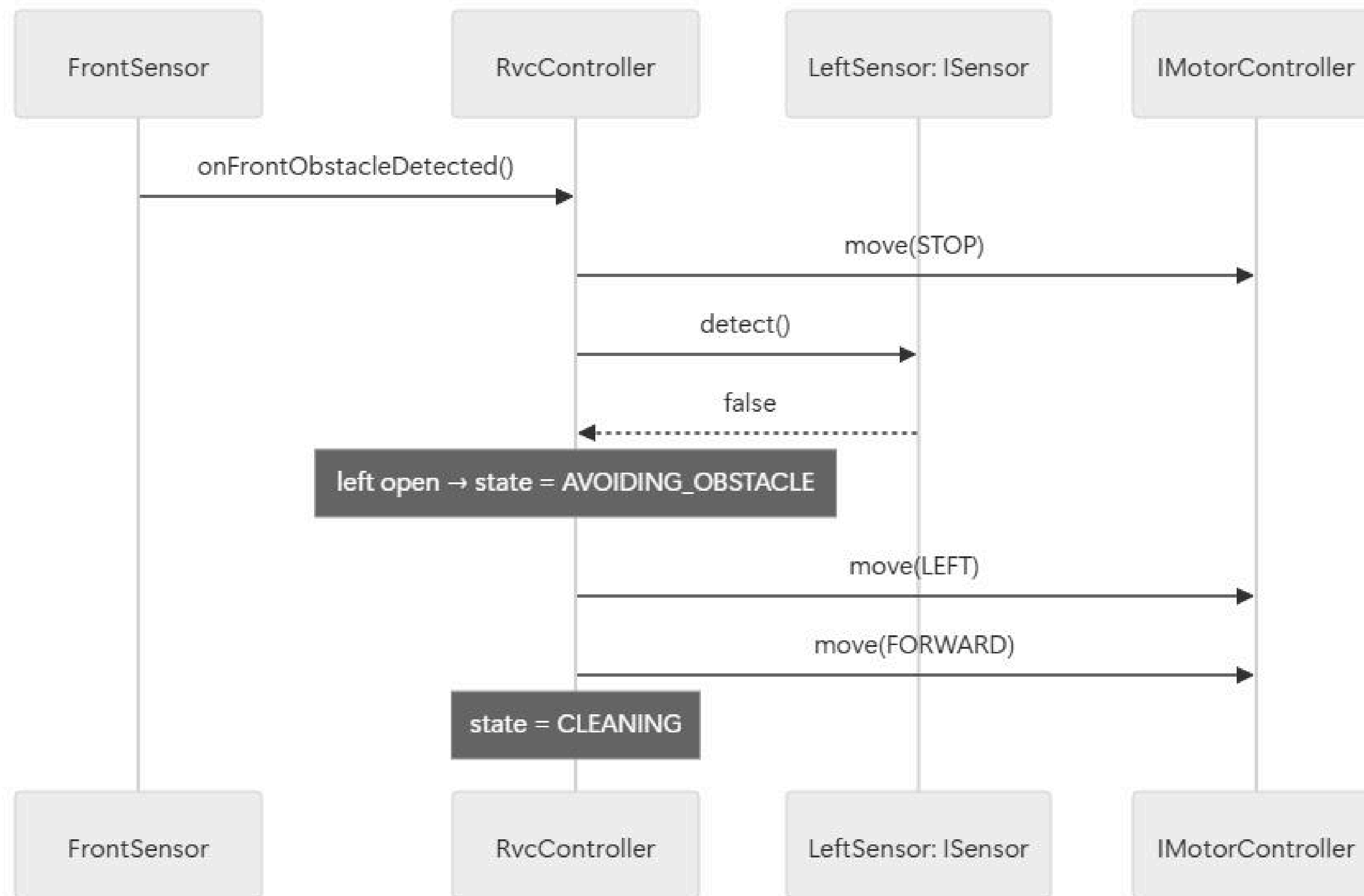
Sequence Diagram

UC#02 - 탐색 및 청소



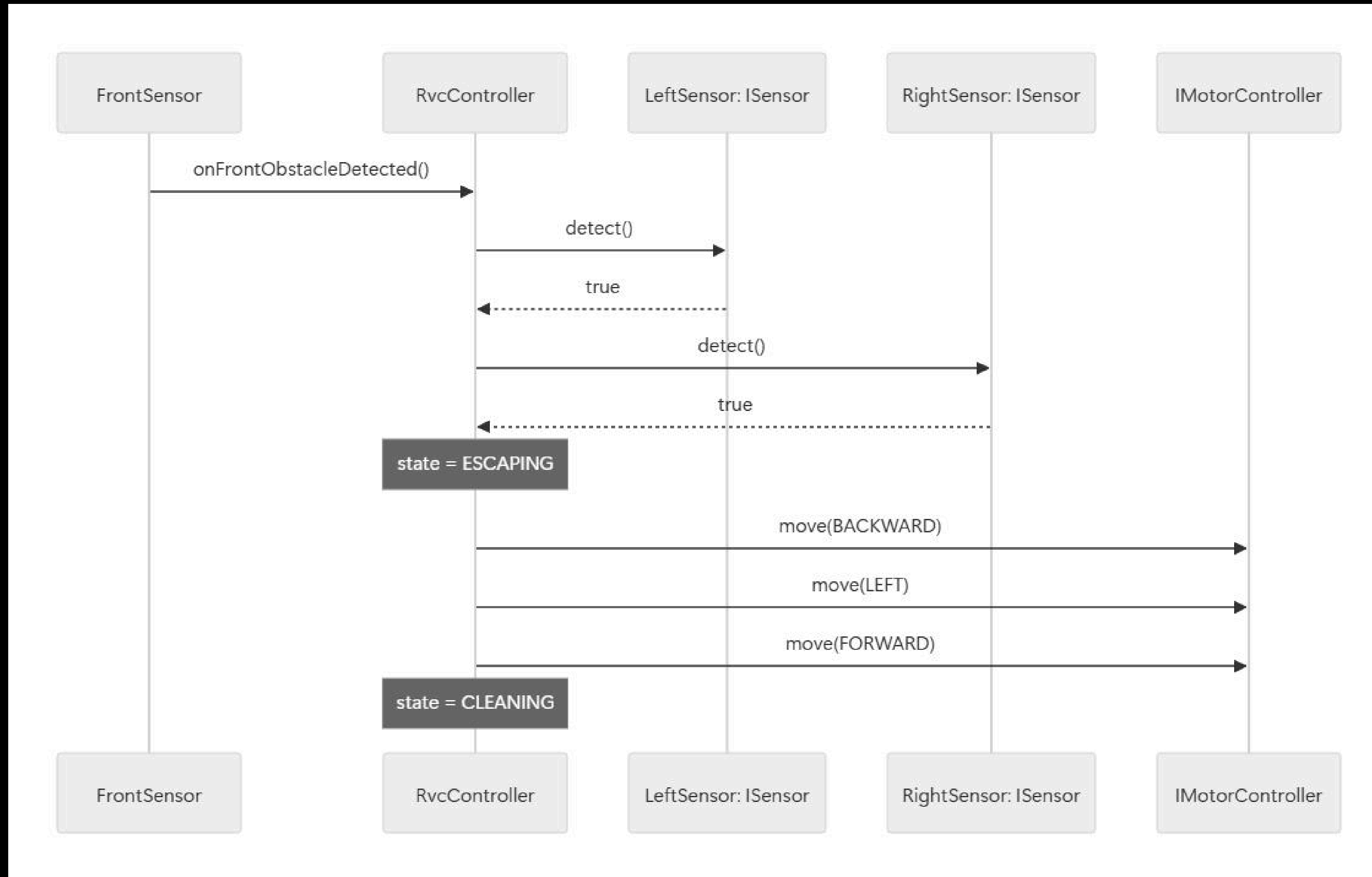
Sequence Diagram

UC#03 - 전방 장애물 회피



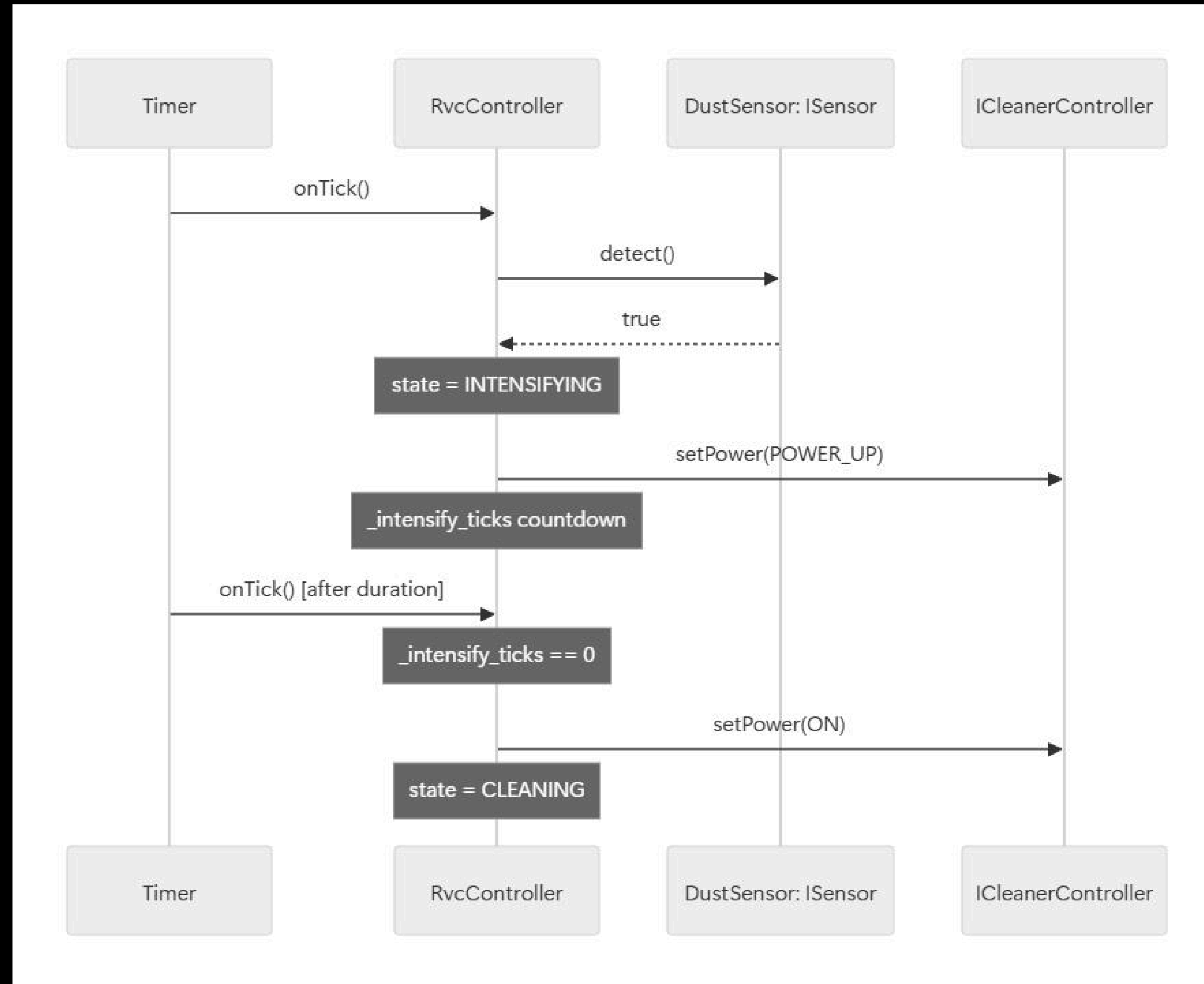
Sequence Diagram

UC#04 - 포위 상태 탈출



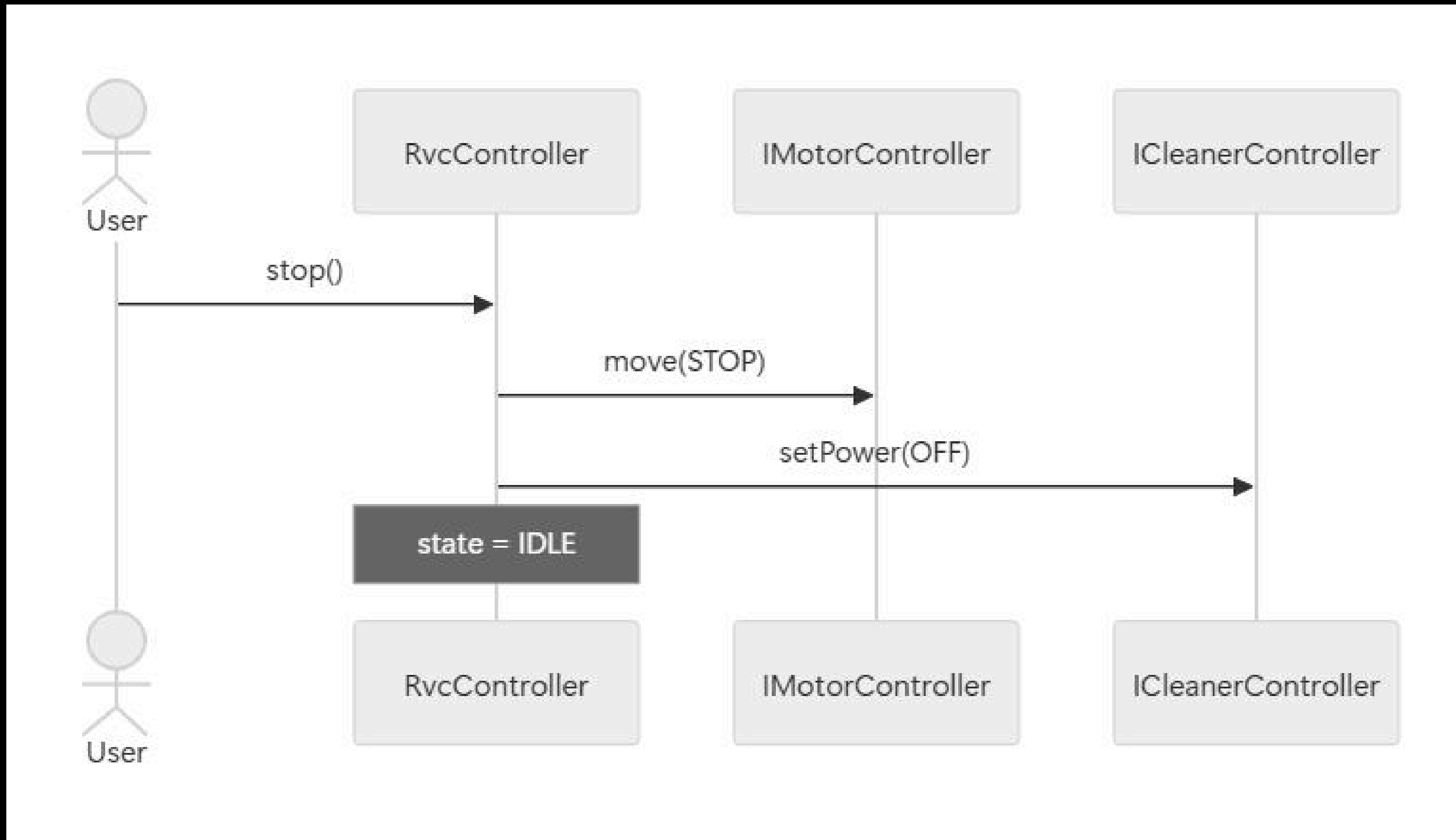
Sequence Diagram

UC#05 - 청소 강도 높이기

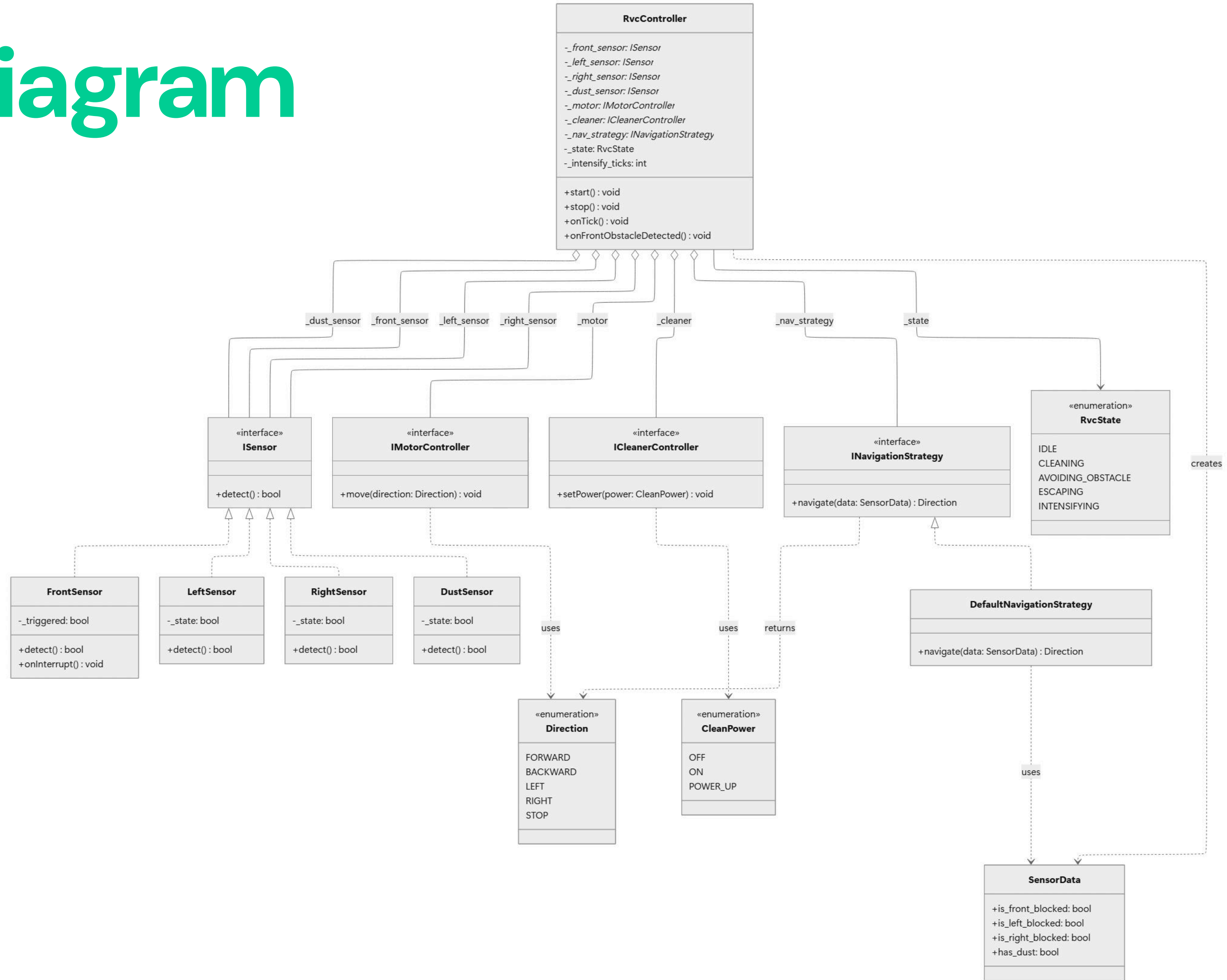


Sequence Diagram

UC#06 - 청소 세션 종료



Class Diagram



Construction phase – OOI

프롬프트

- continue Construction phase

결과물

- Program (RVC Control SW)
- Unit Test

Construction phase – OOI

프롬프트

- "UI는 어떻게 쓸 수 있음?" (How can the UI be used?)
- "프로그램에서 직접 필요한 System Operation을 호출 못하는거야? User가 직접 호출해야 할 수 있어야 하는데?" (Can't system operations be called directly? User should be able to call them.)
- "tick 자동으로 돌아가게 하는건 어려운가?" (Is it difficult to make tick run automatically?)
- "이거 CLI에서 밀림 현상은 어쩔 수 없는건가?" (Is the display pushing/shifting unavoidable?)
- "혹시 dust는 따로 표시되는게 있나?" (Is dust displayed separately?)
- "dust 셀을 *로 표시하고, obstacle 배치하는 거 처럼 dust도 배치하는 sysop 만들어줘" (Display dust as *, add dust placement sysop like obstacle)
- "수정된거 커밋하고 푸시해줘" (Commit and push the changes)

결과물

- Program (UI + Simulator + RVC Control SW)
- Unit Test
- System Test

Claude – Compact Summary

- Compact summary

└ This session is being continued from a previous conversation that ran out of context. The summary below covers the earlier portion of the conversation.

Summary:

1. Primary Request and Intent:

The user is working on an OOAD project building an RVC (Robot Vacuum Cleaner) system in C++17 following the Unified Process (UP). During this session, the user requested:

- Diagnosing why the program only showed Tick incrementing (nothing else moving)
- Making System Operations callable interactively from the running program
- Making tick run automatically in the background
- Fixing terminal display/input collision ("밀리姆 현상")
- Adding dust cell placement (`dust x y`) and visual display (cyan `*`) like obstacles
- Committing and pushing all changes
- Updating all docs based on code changes, writing decisions (AD) and failures (F)
- ****Critically****: Ensuring ALL Control SW docs contain ONLY RVC Control SW and hardware content – Simulator content moved to separate files
- Comparing and updating Notion "KONKUK 2026-1 OOAD / Vibe Coding Result" with latest content
- Updating Unit Test and System Test databases on Notion
- Creating a Use Case diagram in Mermaid and adding to Notion

2. Key Technical Concepts:

- C++17, CMake, Google Test (FetchContent v1.14.0)
- OOAD / Unified Process (UP): Inception → Elaboration → Construction
- 4-layer architecture: Application → Domain → Interface → HAL
- State machine: `RvcState` (IDLE, CLEANING, AVOIDING_OBSTACLE, ESCAPING, INTENSIFYING)
- Strategy pattern: `INavigationStrategy` / `DefaultNavigationStrategy`
- Constructor injection (DI) for all hardware dependencies
- Interrupt-driven FrontSensor vs. polled Left/Right/DustSensor
- `termios` raw mode (`~ECHO | ~ICANON`) for split display/input in terminal
- `std::thread` + `std::mutex` for concurrent auto-tick and user input
- Grid-based dust cell placement with auto-detection and consumption
- ANSI escape codes for terminal rendering (`\033[H`, `\033[2J`, `\033[?25l`, color codes)
- Notion MCP integration for page creation/update
- Mermaid flowchart diagrams for Use Case representation

Result (Simulator)

```
+-----+
| . . . . . < . . . . . |
| . . . . . | . . . . . |
| . . . . . x x x x x . . . . . |
| . . . . . x . . . . . x |
| . . . . . x . . . . . x |
| . . . . . . . . . . x |
| . . . . . . . . . . x |
| . . . . . . . . . . x |
| . . . . . x . . . . . x |
| . . . . . x x x x x x x x x x . . . . . |
| . . . . . | . . . . . |
| . . . . . | . . . . . |
+-----+
Tick:33 | Dir:FORWARD | Clean:ON | Pos:(14,0) | Heading:WEST
>
=== RVC System Operations ===
start      startCleaning()
stop       stopCleaning()
speed <ms> set tick interval (default 300)
dust x y   placeDust(x, y) – shown as cyan *
obstacle x y placeObstacle(x, y)
help       show this menu
quit       exit
```

Unit Test

Name	Expected Result	PorNP	Test #	TestGroup
ForwardWhenAllClear	navigate() returns FORWARD	P	UT-01	DefaultNavigationStrategy
LeftWhenFrontBlockedAndBothSidesOpen	navigate() returns LEFT	P	UT-02	DefaultNavigationStrategy
RightWhenFrontAndLeftBlocked	navigate() returns RIGHT	P	UT-03	DefaultNavigationStrategy
LeftWhenFrontAndRightBlocked	navigate() returns LEFT	P	UT-04	DefaultNavigationStrategy
BackwardWhenSurrounded	navigate() returns BACKWARD	P	UT-05	DefaultNavigationStrategy
DustAloneDoesNotAffectDirection	navigate() returns FORWARD (dust 무시)	P	UT-06	DefaultNavigationStrategy
StartMovesForwardAndCleansOn	motor=FORWARD, cleaner=ON	P	UT-07	RvcController
StopHaltsMotorAndTurnsOffCleaner	motor=STOP, cleaner=OFF	P	UT-08	RvcController
TickDoesNothingWhenIdle	motor.log 비어있음, cleaner.log 비어있음	P	UT-09	RvcController
DustDetectionPowersUpCleaner	cleaner=POWER_UP	P	UT-10	RvcController
CleaningPowerRestoresAfterIntensifyDust	INTENSIFY_DURATION tick 경과 후 cleaner=ON	P	UT-11	RvcController
FrontObstacleOpenSidesTurnsLeft	motor 순서: STOP → LEFT → FORWARD	P	UT-12	RvcController
FrontObstacleLeftBlockedTurnsRight	motor 순서: STOP → RIGHT → FORWARD	P	UT-13	RvcController
SurroundedEscapesBackwardThenLeftThenRight	motor 순서: STOP → BACKWARD → LEFT → FORWARD	P	UT-14	RvcController
FrontObstacleIgnoredWhenIdle	motor.log 비어있음 (Idle 중 인터럽트 무시)	P	UT-15	RvcController

System Test

Name	Expected Result	Reference #	Test #	Test Scenario	Test Type
ST-01: StartMovesForwardAndCleansOn	motor lastDirection = FORWARD, cleaner lastPower = ON	UC-01	ST-01	RVC 시작 후 Motor가 FORWARD, Cleaner가 ON 상태인지 확인	Positive
ST-07: TickIgnoredWhenIdle	motor log 및 cleaner log 모두 비어있음 (명령 없음)	UC-02	ST-07	Idle 상태에서 Timer Tick 발생 — 명령 무시 확인	Negative
ST-02: NormalTickReissuesForward	motor log 크기 1 증가 (FORWARD 재발행), cleaner log 크기 불변	UC-02	ST-02	start() 후 장애물/먼지 없는 정상 Tick 발생 — onTick()이 매 사이클 FORWARD를 재발행하는지 확인 (AD-07)	Positive
ST-04: FrontObstacleAvoidanceResumesForward	lastDirection = FORWARD (회피 후 전진 복귀)	UC-03	ST-04	전방 장애물 감지, 측면 개방 — 회피 후 전진 재개	Positive
ST-05: SurroundedEscapeResumesForward	BACKWARD → LEFT(또는 RIGHT) 회전 → FORWARD 순서로 명령 발행, lastDirection = FORWARD	UC-04	ST-05	전방 + 좌측 + 우측 모두 차단 — 포위 탈출 시퀀스 수행	Positive
ST-03: DustPowersUpCleanerThenRestores	Cleaner POWER_UP 명령 발행 → 지속 시간 후 ON(정상)으로 복귀	UC-05	ST-03	현재 위치에 먼지 셀 배치(placeDust) 후 Tick → 자동 감지 및 셀 소비 → INTENSIFY_DURATION Tick 경과 (AD-10)	Positive
ST-06: StopHaltsMotorAndCleaner	lastDirection = STOP, lastPower = OFF	UC-06	ST-06	정지 명령 발행 — Motor 및 Cleaner 정지 확인	Positive

CI/CD SetUp

프롬프트

- 현재 CI 파이프라인을 반영할 수 있도록 ci.yml 파일을 수정해줘

결과물

작업 내용을 정리합니다.

ci.yml 에 반영된 내용:

단계	내용
Install dependencies	cmake, g++, clang-tidy, gcovr 설치
Configure	-DENABLE_COVERAGE=ON 으로 커버리지 플래그 활성화
Build	--parallel 로 병렬 빌드
Run tests	ctest --output-on-failure
Coverage report	gcovr로 HTML 리포트 생성 (_deps 제외)
Upload artifact	커버리지 리포트를 Actions 아티팩트로 업로드

















CI/CD

Build & Test

succeeded 9 hours ago in 1m 10s

- > Set up job
- > Checkout
- > Install dependencies
- > Cache FetchContent dependencies
- > Configure (with coverage)
- > Build
- > Run tests
- > Static analysis
- > Coverage report
- > Upload coverage report
- > Post Cache FetchContent dependencies
- > Post Checkout
- > Complete job

CI/CD

All workflows		Filter workflow runs				
Showing runs from all workflows						
8 workflow runs		Workflow	Event	Status	Branch	Actor
	fix: resolve clang-tidy errors in main.cpp and Simulator.cpp CI #8: Commit d72d75c pushed by bakbakwanwan	main			29 minutes ago	...
	test: update SimulatorTest to match current Simulator/RvcController b... CI #7: Commit 87023a1 pushed by bakbakwanwan	main			37 minutes ago	...
	docs: update Control SW docs and add simulator-separated decision/fai... CI #6: Commit ab7c084 pushed by bakbakwanwan	main			39 minutes ago	...
	feat: interactive CLI with auto-tick, dust placement, and grid display CI #5: Commit a0f5f54 pushed by bakbakwanwan	main			Today at 9:30 AM	...
	fix: resolve all clang-tidy errors blocking CI static analysis CI #4: Commit 03cbda3 pushed by bakbakwanwan	main			Today at 12:55 AM	...
	docs refinement CI #3: Commit 7cdcdbf pushed by bakbakwanwan	main			Today at 12:46 AM	...
	additional CI setting CI #2: Commit 26249e1 pushed by bakbakwanwan	main			May 18, 1:00 AM GMT+9	...
	ci setting CI #1: Commit dba32b7 pushed by bakbakwanwan	main			May 17, 4:49 PM GMT+9	...

Static Analysis

```
Static analysis
[12/13] Processing file /home/runner/work/vibeCoding/vibeCoding/test/simulator/SimulatorTest.cpp.
158892 warnings generated.
[13/13] Processing file /home/runner/work/vibeCoding/vibeCoding/test/app/RvcControllerTest.cpp.
196391 warnings generated.
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:17:23: error: variable 'g_orig_termios' is non-const and globally accessible, consider making it const [cppcoreguidelines-avoid-non-const-global-variables,-warnings-as-errors]
17 | static struct termios g_orig_termios;
    |                       ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:78:9: error: multiple declarations in a single statement reduces readability [readability-isolate-declaration,-warnings-as-errors]
78 |     int x = 0, y = 0;
    |     ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:81:9: error: multiple declarations in a single statement reduces readability [readability-isolate-declaration,-warnings-as-errors]
81 |     int x = 0, y = 0;
    |     ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:92:5: error: function 'main' has cognitive complexity of 26 (threshold 25) [readability-function-cognitive-complexity,-warnings-as-errors]
92 | int main() {
    | ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:99:5: note: +1, including nesting penalty of 0, nesting level increased to 1
99 |     for (int x = 5; x <= 9; ++x) { sim.placeObstacle(x, 2); }
    |     ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:100:5: note: +1, including nesting penalty of 0, nesting level increased to 1
100 |     for (int y = 3; y <= 4; ++y) { sim.placeObstacle(0, y); }
    |     ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:101:5: note: +1, including nesting penalty of 0, nesting level increased to 1
101 |     for (int y = 7; y <= 8; ++y) { sim.placeObstacle(0, y); }
    |     ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:102:5: note: +1, including nesting penalty of 0, nesting level increased to 1
102 |     for (int x = 5; x <= 14; ++x) { sim.placeObstacle(x, 8); }
    |     ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:103:5: note: +1, including nesting penalty of 0, nesting level increased to 1
103 |     for (int y = 3; y <= 7; ++y) { sim.placeObstacle(16, y); }
    |     ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:114:29: note: nesting level increased to 1
114 |     std::thread tick_thread([&]() {
    |                             ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:115:9: note: +2, including nesting penalty of 1, nesting level increased to 2
115 |         while (running) {
    |         ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:140:5: note: +1, including nesting penalty of 0, nesting level increased to 1
140 |     while (!quit) {
    |     ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:142:9: note: +2, including nesting penalty of 1, nesting level increased to 2
142 |         if (read(STDIN_FILENO, &c, 1) <= 0) { break; }
    |         ^
/home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp:146:9: note: +2, including nesting penalty of 1, nesting level increased to 2
146 |         if (c == '\n' || c == '\r') {
```

```
Static analysis
1 ▶ Run find src test -name '*.cpp' \
5 [1/13] Processing file /home/runner/work/vibeCoding/vibeCoding/src/hal/DustSensor.cpp.
6 [2/13] Processing file /home/runner/work/vibeCoding/vibeCoding/src/hal/FrontSensor.cpp.
7 [3/13] Processing file /home/runner/work/vibeCoding/vibeCoding/src/hal/RightSensor.cpp.
8 [4/13] Processing file /home/runner/work/vibeCoding/vibeCoding/src/hal/LeftSensor.cpp.
9 [5/13] Processing file /home/runner/work/vibeCoding/vibeCoding/src/ui/ConsoleDisplay.cpp.
10 17551 warnings generated.
11 [6/13] Processing file /home/runner/work/vibeCoding/vibeCoding/src/ui/GridDisplay.cpp.
12 37009 warnings generated.
13 [7/13] Processing file /home/runner/work/vibeCoding/vibeCoding/src/domain/DefaultNavigationStrategy.cpp.
14 38373 warnings generated.
15 [8/13] Processing file /home/runner/work/vibeCoding/vibeCoding/src/simulator/Simulator.cpp.
16 50493 warnings generated.
17 [9/13] Processing file /home/runner/work/vibeCoding/vibeCoding/src/app/main.cpp.
18 76950 warnings generated.
19 [10/13] Processing file /home/runner/work/vibeCoding/vibeCoding/src/app/RvcController.cpp.
20 78314 warnings generated.
21 [11/13] Processing file /home/runner/work/vibeCoding/vibeCoding/test/domain/DefaultNavigationStrategyTest.cpp.
22 116581 warnings generated.
23 [12/13] Processing file /home/runner/work/vibeCoding/vibeCoding/test/simulator/SimulatorTest.cpp.
24 158086 warnings generated.
25 [13/13] Processing file /home/runner/work/vibeCoding/vibeCoding/test/app/RvcControllerTest.cpp.
26 196385 warnings generated.
27 Suppressed 196428 warnings (196385 in non-user code, 43 NOLINT).
28 Use -header-filter=. to display errors from all non-system headers. Use -system-headers to display errors from system headers as well.
```

Static Code Analysis with Claude

- 테스트는 통과했는데 clang-tidy가 실패했어요. 에러 내용 확인할게요.

• **Bash** Show failed clang-tidy log

```
IN gh run view 26199133014 --log-failed 2>&1
OUT Build & Test Static analysis 2026-05-21T01:03:16.0110247Z ##[group]Run find src test -name '*.cpp' #
Build & Test Static analysis 2026-05-21T01:03:16.0110630Z ^[[36;1mfind src test -name '*.cpp' #^[[0m
Build & Test Static analysis 2026-05-21T01:03:16.0110986Z ^[[36;1m | xargs clang-tidy -p build --warnings-as-errors='+'^[[0m
```

• Thinking

- 에러가 5가지예요. `src/app/main.cpp` 와 `src/simulator/Simulator.cpp` 를 수정할게요.

• **Read** main.cpp

• 에러 목록:

1. `main.cpp:17` — `g_orig_termios` non-const global
2. `main.cpp:78,81` — `int x = 0, y = 0;` 두 줄 모두 선언 분리 필요
3. `main.cpp:92` — `main()` cognitive complexity 26 (한계 25)
4. `main.cpp:141` — `char c;` 미초기화
5. `Simulator.cpp:28` — `count()` implicit bool 변환

이제 빌드해서 테스트 통과 확인 후 push할게요.

• **Bash** Rebuild and check for errors

```
IN cmake --build build --parallel 2>&1 | grep -E "error:|warning:|Built target" | head -20
OUT [ 12%] Built target gtest
[ 20%] Built target gtest_main
[ 29%] Built target gmock
[ 37%] Built target gmock_main
```

• **Bash** Run all tests

```
IN cd /home/wan/00ADProject/build && ctest --output-on-failure 2>&1 | tail -5
OUT 22/22 Test #22: SimulatorTest.TickIgnoredWhenIdle ..... Passed 0.02 sec

100% tests passed, 0 tests failed out of 22
```

Thank

you!

