객체지향개발방법론
Practice #5

202211287 김태인

202311252 곽수호

202111368 정선민

202211378 조성원

목차

Cursor

사용자 정의

Project Rule: 프로젝트 단위로 AI에게 적용되는 지속 규칙

Skills: 특정 작업을 잘 수행하기 위한 재사용 가능한 능력/절차

Plugins: rules, skills, agents, hooks, MCP 등을 묶어 설치하는 확장 패키지

하위 에이전트: 특정 역할을 맡는 별도 AI 에이전트

훅: 특정 이벤트가 발생할 때 자동으로 실행되는 동작

MCP: Cursor가 외부 도구/데이터와 연결되는 표준 방식

rules, skills, agents, hooks, MCP
등을 묶어 설치하는 확장 패키지

Cursor

Models

모델의 모든 속성은 모델 및 요금제 페이지에서 확인하세요.

이름	기본 컨텍스트	최대 모드	기능
Claude 4 Sonnet	200k	-	
Claude 4 Sonnet 1M	-	1M	
Claude 4.5 Haiku	200k	-	
Claude 4.5 Opus	200k	200k	
Claude 4.5 Sonnet	200k	1M	
Claude 4.6 Opus	200k	1M	
Claude 4.6 Opus (Fast mode)	200k	1M	
Claude 4.6 Sonnet	200k	1M	

Markdown files - rules

```
---
description: RVC OOAD project C++ build, test, CI/CD conventions
alwaysApply: true
---

### CI/CD

CI/CD: Github Actions
IDE : VS Code
Version Control: Github
Target Language: C++
Build : CMake
Unit-test: G-test로 테스트 자동화
System Test: 파이썬으로 제작한 시뮬레이터 환경에서 테스트
Static Analysis: SonarCloud, Cppcheck, Clang-tidy
Dynamic Analysis: Gcov
Output & Feedback: Github Pages, Discord Webhook
```

개발, 빌드환경, 테스팅 환경과 CI/CD환경을 미리 지정해 사용할 환경을 확정함.

alwaysApply: true 를 이용해 매번 참고하도록 설정

Markdown files - skill

name	skill-database
description	Provides RVC software controller requirements and domain constraints for OOAD work. Use when working on RVC requirements, design, controller behavior, or project documentation.

Preliminary Requirements for RVC SW Controller

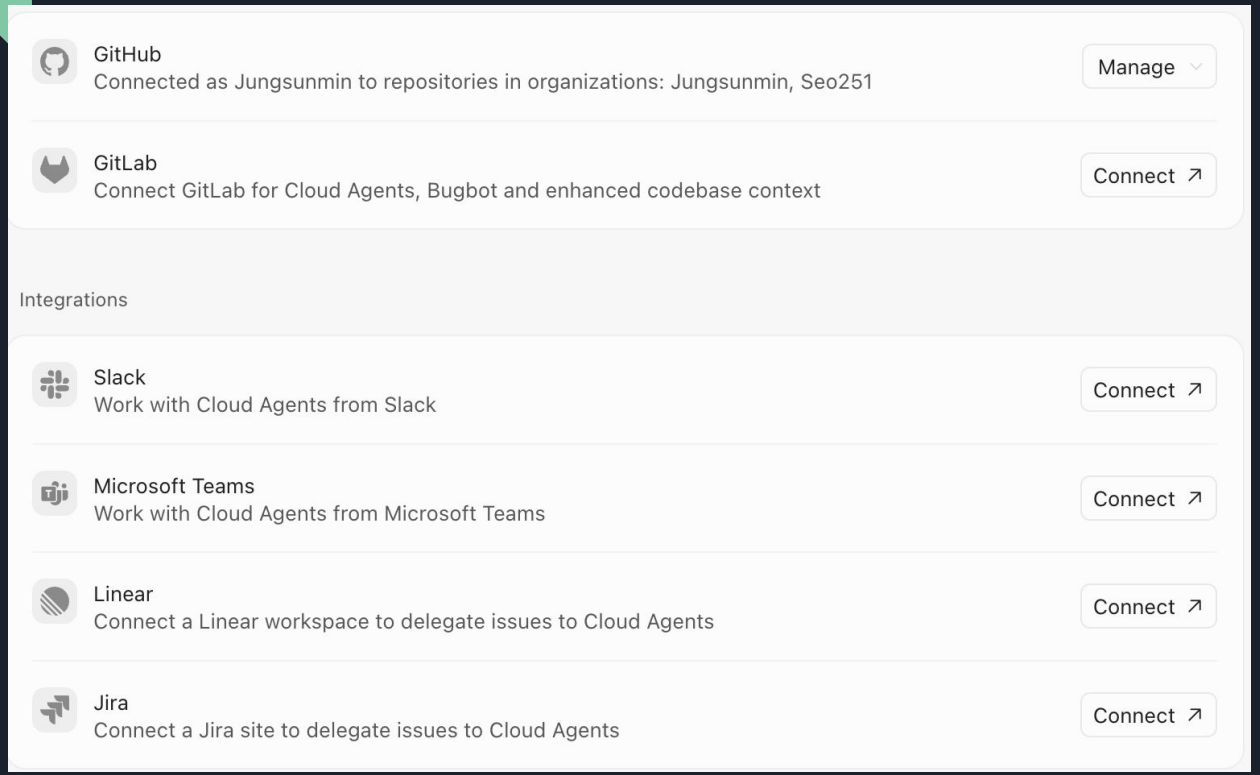
• An RVC automatically cleans and mops household surface. • It goes straight forward while cleaning. • If its sensors found an obstacle, it stops cleaning, turns aside left or right, and goes forward with cleaning. • If there are obstacles in both front, left and right, it move backward and turn aside 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.

RVC Input/Output Event Definitions

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

개발할 프로그램의 전제조건과 예비 요구사항, 주의사항과 입력/출력 되는 이벤트의 포맷을 정의

Cursor - github repo 연결



Step1. Inception

Prompt - Step1. Inception

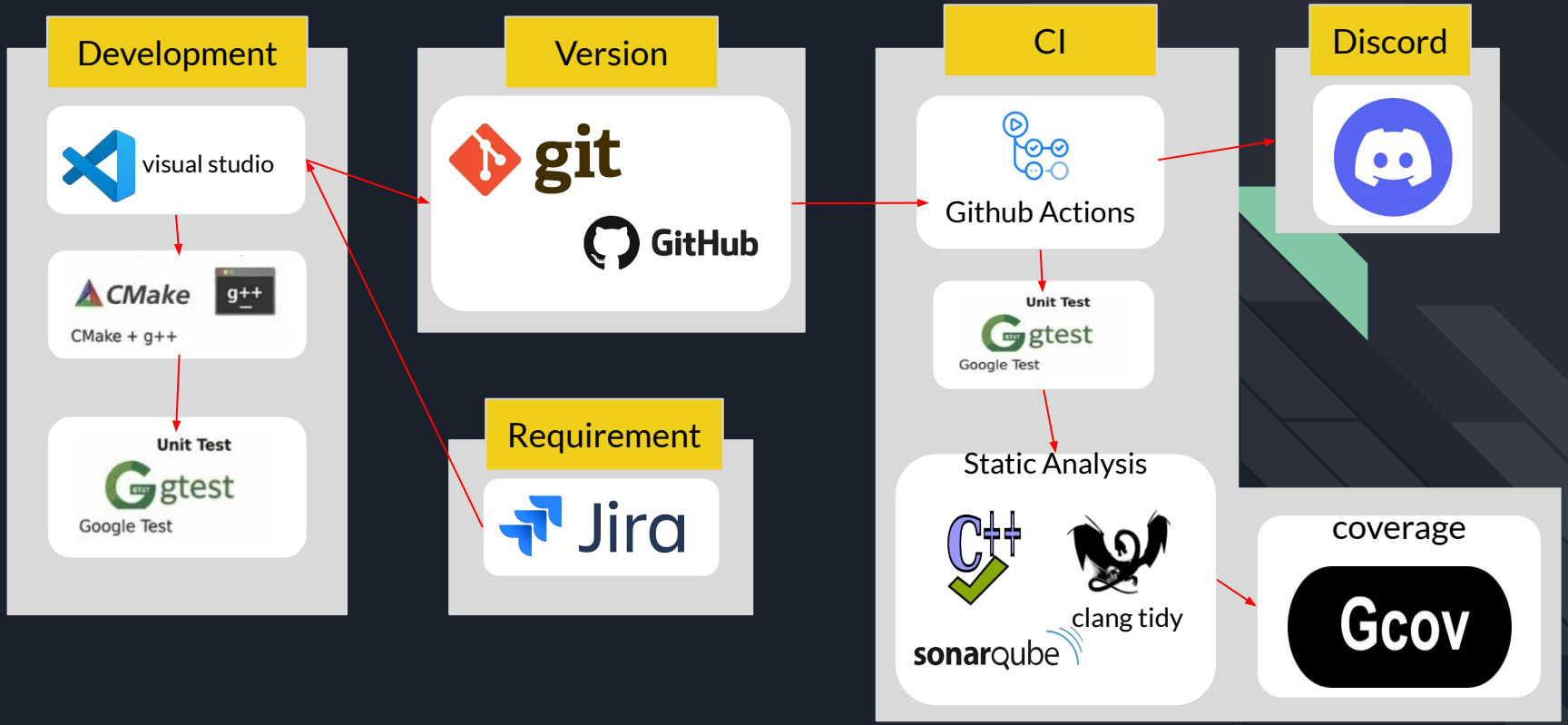
프롬프트 1 (The Inception (UP))

1. 우리는 rvc_controller sw를 OOAD (UP) 개발방식에 따라 c++ 코드로 개발하려고 해. 우선 CI/CD 환경을 github Action 으로 CI 서버를 구축하고 google test로 testing 진행할거야. test는 pull request할 때마다 돌아가게 하려고 해. pr이 오면 review가 3명 이상이어야 pr이 가능하고 static code analysis도 sonarqube 포함 3개 이용하고 싶어. Code version control은 git으로 하고 cmake 써서 빌드해. discord를 통해 CI 서버에서 test, static analysis 결과, build 성공 여부 등을 정보로 계속 받아보고 싶어. gcov를 통해 coverage 분석도 있었으면 좋겠어.

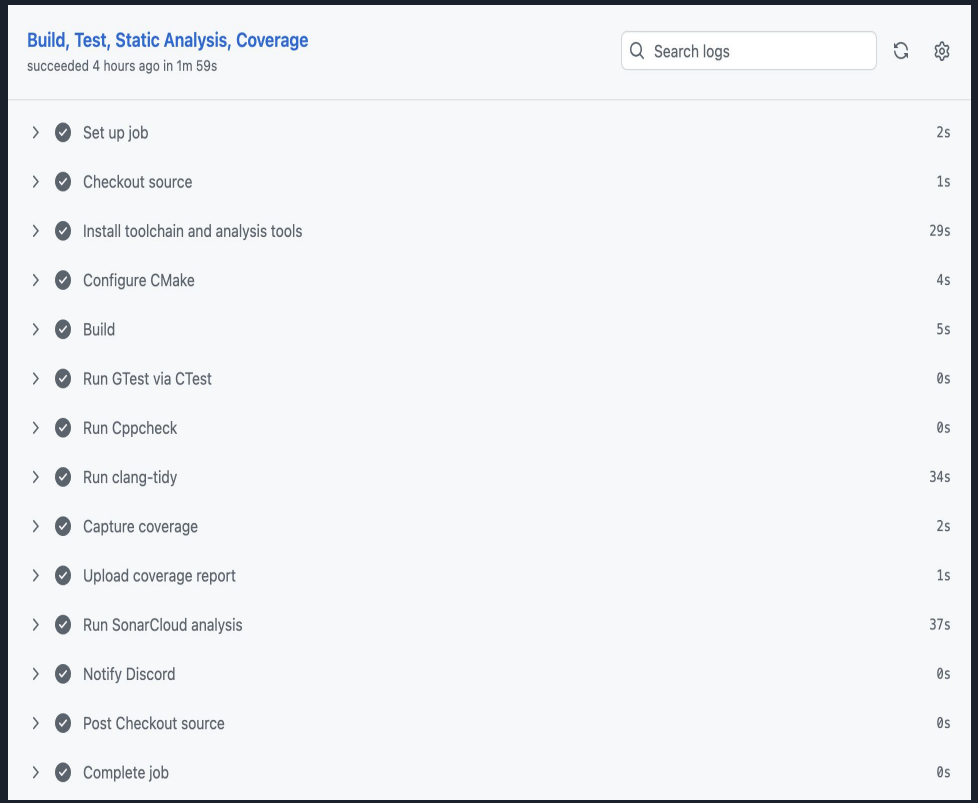
Step1-1. Inception 수정

내가 원한건 먼지를 감지 했을때, 전진하면서 power up모드가 되어서 3초동안 청소를 하고, booster up 모드 중 다시 먼지를 감지할 경우 그 시점부터 다시 3초를 리셋해서 booster up 모드를 유지시키는 기능을 원했어, fr, useCase 단계부터 검사해서 틀린부분이 있으면 고쳐주고 소스코드, 유닛 테스트 코드까지 반영해서 수정해줘.

CI/CD 환경 구성


Development
visual studio
CMake
g++
CMake + g++
Unit Test
gtest
Google Test
Version
git
GitHub
Requirement
Jira
CI
Github Actions
Unit Test
gtest
Google Test
Static Analysis
C++
clang tidy
sonarqube
coverage
Gcov
Discord

Github Action



Jira

Discord

RVC CI Bot 앱 어제 오후 11:28

RVC Controller CI Result

Build, GTest/CTest, static analysis, and coverage completed with status: success

Repository	Branch	Commit
Seo251/RVC_OOAD	main	1654e7f

Run
https://github.com/Seo251/RVC_OOAD/actions/runs/26039741068

2026년 5월 19일

RVC CI Bot 앱 오전 1:15

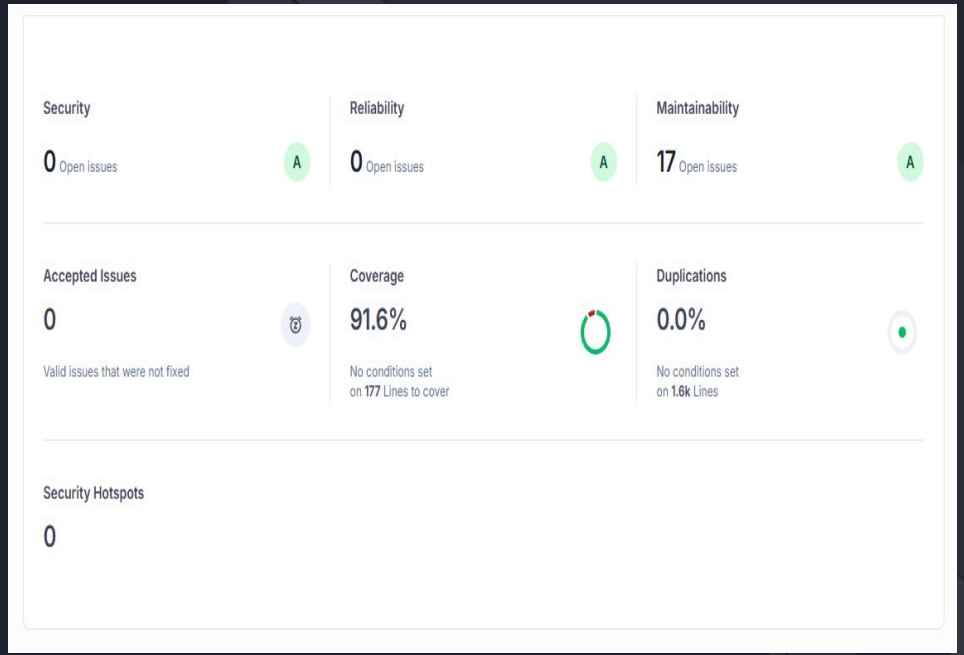
RVC Controller CI Result

Build, GTest/CTest, static analysis, and coverage completed with status: success

Repository	Branch	Commit
Seo251/RVC_OOAD	main	d5c335b

Run
https://github.com/Seo251/RVC_OOAD/actions/runs/26045581348

Sonar Cloud



Use case

UC-001 Start Cleaning

Field	Detail
Trigger	User invokes `PressPowerButton()` while `PoweredOff`.
Main Flow	Controller turns power on, enters `CheckingFront`, requests front path state. If clear, motor forward and cleaner on.
Alternative	If front obstacle is detected during startup, controller does not turn cleaner on and enters UC-003.
Exception	If user presses the button again before cleaning starts, UC-002 priority applies and the controller powers off safely.

Use case

UC-002 Stop Cleaning

Field	Detail
Trigger	User invokes `PressPowerButton()` while powered on.
Main Flow	Controller cancels timer, stops motor, turns cleaner off, and sets `PoweredOff`.
Priority	This use case overrides cleaning, obstacle avoidance, and dust power-up.
Exception	No active process exists: still output motor stop and cleaner off to keep the controller safe.

Use case

UC-003 Avoid Obstacle

Field	Detail
Trigger	`FrontObstacleDetected()` while powered on.
Main Flow	Controller cancels power-up timer, turns cleaner off, enters `AvoidingObstacle`, checks left/right sensors, turns toward clear side, then resumes forward cleaning.
Alternative A	Left blocked and right clear: turn right.
Alternative B	Left and right blocked: move backward repeatedly until one side is clear, then turn toward clear side.
Exception	Dust detected during avoidance is ignored. Cleaner remains off.
Exception	User button during avoidance immediately stops motor, turns cleaner off, cancels timer, and powers off.

Use case

UC-004 Power Up Cleaning On Dust

Field	Detail
Trigger	`DustDetected()` while `CleaningForward`.
Main Flow	Controller sets cleaner to power-up and starts a 3-second timer. **No motor command is issued at any point during this use case** (NFR-011); the forward motion that started before the dust event continues uninterrupted, and after 3 seconds the cleaner returns to normal on while motor still keeps moving forward.
Alternative	Dust detected again while already in `PowerUpCleaning`: discard the previous timer and start a new 3-second timer from that moment. Cleaner stays at power-up. **No motor command is issued**; motor keeps moving forward.
Exception	Front obstacle detected during power-up: cancel timer, turn cleaner off, enter obstacle avoidance. Existing power-up elapsed time is discarded. Motor commands from this exception come from the avoidance flow (a motor-side event), not from the cleaner.
Exception	Dust detected while turning/backward/avoiding: ignored because cleaner must remain off during avoidance. Motor is also not touched.

FR - 21개

ID	Requirement
FR-001	`rvc_controller` 는 초기 상태에서 전원이 꺼져 있어야 한다.
FR-002	전원이 꺼진 상태에서 사용자가 버튼을 누르면 `rvc_controller` 는 전원을 켜고 청소 시작 절차를 수행해야 한다.
FR-003	전원이 켜진 상태에서 사용자가 버튼을 누르면 `rvc_controller` 는 motor를 정지시키고 cleaner를 끈 뒤 청소를 종료해야 한다.
FR-004	전원이 켜지거나 청소 중일 때 `rvc_controller` 는 전방 장애물 여부를 확인해야 한다.
FR-005	전방 장애물이 없으면 `rvc_controller` 는 motor를 전진시키고 cleaner를 켜야 한다.
FR-006	전방 장애물이 감지되면 `rvc_controller` 는 장애물 회피 프로세스를 시작해야 한다.
FR-007	장애물 회피 중에는 cleaner를 꺼야 한다.
FR-008	장애물 회피 프로세스는 먼저 좌측 장애물 여부를 확인해야 한다.
FR-009	좌측 장애물이 없으면 `rvc_controller` 는 motor에 좌회전 명령을 내려야 한다.

FR

FR-010	좌측 장애물이 있으면 `rvc_controller`는 우측 장애물 여부를 확인해야 한다.
FR-011	우측 장애물이 없으면 `rvc_controller`는 motor에 우회전 명령을 내려야 한다.
FR-012	좌측과 우측 모두 장애물이 있으면 `rvc_controller`는 좌측 또는 우측 중 하나가 비어 있을 때까지 motor에 후진 명령을 내려야 한다.
FR-013	후진 중 좌측 또는 우측 중 하나가 비어 있으면 `rvc_controller`는 장애물이 없는 방향으로 회전해야 한다.
FR-014	장애물 회피 프로세스가 종료되면 `rvc_controller`는 다시 motor를 전진시키고 cleaner를 켜야 한다.
FR-015	전진 중 전방 먼지가 감지되면 `rvc_controller`는 motor의 전진 동작을 멈추거나 지연시키지 않은 채 cleaner의 흡입력만 3초 동안 증가시켜야 한다.
FR-016	흡입력 증가 후 3초가 지나면 `rvc_controller`는 cleaner의 흡입력을 정상화해야 하며, 이 변경은 motor의 전진 동작에 영향을 주지 않아야 한다.
FR-017	흡입력 증가 중 전방 장애물이 감지되면 `rvc_controller`는 증가된 시간을 저장하지 않고 즉시 cleaner를 끄고 장애물 회피 프로세스를 시작해야 한다.
FR-021	흡입력 증가 중 다시 먼지가 감지되면 `rvc_controller`는 이전 타이머를 폐기하고 그 시점부터 다시 3초 타이머를 시작해야 한다. 이 재시작은 motor 동작에 영향을 주지 않는다.

FR

FR-018	`rvc_controller`는 motor 명령을 `Forward`, `Backward`, `TurnLeft`, `TurnRight`, `Stop` 중 하나로 출력해야 한다.
FR-019	`rvc_controller`는 cleaner 명령을 `Off`, `On`, `PowerUp` 중 하나로 출력해야 한다.
FR-020	`rvc_controller`는 전방 센서, 좌측 센서, 우측 센서, 먼지 센서, 사용자 버튼 입력을 처리해야 한다.

NFR - 11개

ID	Requirement
NFR-001	장애물 감지 후 cleaner를 끄고 회피 명령을 내리는 반응은 사용자가 체감할 수 없을 정도로 즉시 수행되어야 한다.
NFR-002	먼지 감지 후 흡입력 증가 시간은 3초를 기준으로 하며, 타이머 오차는 구현 단계에서 명시적으로 관리되어야 한다.
NFR-003	전원 종료 명령은 청소, 회피, 흡입력 증가 상태보다 우선해야 한다.
NFR-004	컨트롤러 로직은 하드웨어 상세 제어와 분리되어 단위 테스트가 가능해야 한다.
NFR-005	모든 핵심 상태 전이는 GTest 기반 단위 테스트로 검증 가능해야 한다.
NFR-006	CI에서는 CMake build, GTest, Cppcheck, clang-tidy, SonarCloud 분석, coverage 수집이 자동으로 수행되어야 한다.

NFR

NFR-007	회피 정책은 좌측 우선 정책을 기본으로 하되, 향후 정책 변경이 가능하도록 controller 내부 의사결정과 actuator 출력이 분리되어야 한다.
NFR-008	전원 상태, 청소 상태, 회피 상태, 흡입력 증가 상태는 명확한 상태 모델로 표현되어야 한다.
NFR-009	잘못된 센서 조합이나 반복 입력에도 controller는 정의되지 않은 motor/cleaner 명령을 출력하지 않아야 한다.
NFR-010	본 Inception 범위에서는 모터 각도, 속도, 흡입력 세기 등 하드웨어 상세 파라미터를 결정하지 않는다.
NFR-011	Motor 출력(`Forward` , `Backward` , `TurnLeft` , `TurnRight` , `Stop`)과 Cleaner 출력(`Off` , `On` , `PowerUp`)은 서로 독립적으로 결정되어야 한다. Motor 상태 변화는 Cleaner 상태에 영향을 줄 수 있으나(예: 전방 장애물 → cleaner off, 전진 시작 → cleaner on), 반대로 Cleaner 상태 변화는 Motor 상태나 진행을 변경하지 않아야 한다(예: 먼지 감지로 cleaner가 `PowerUp` 이 되어도 motor의 전진은 멈추거나 느려지지 않는다).

Step2. OOA

Prompt - Step2. OOA

2. CI/CD 구축한 걸 가지고
프롬프트 2 (OOA (UP): FR + Use cases + SSD + Domain model)

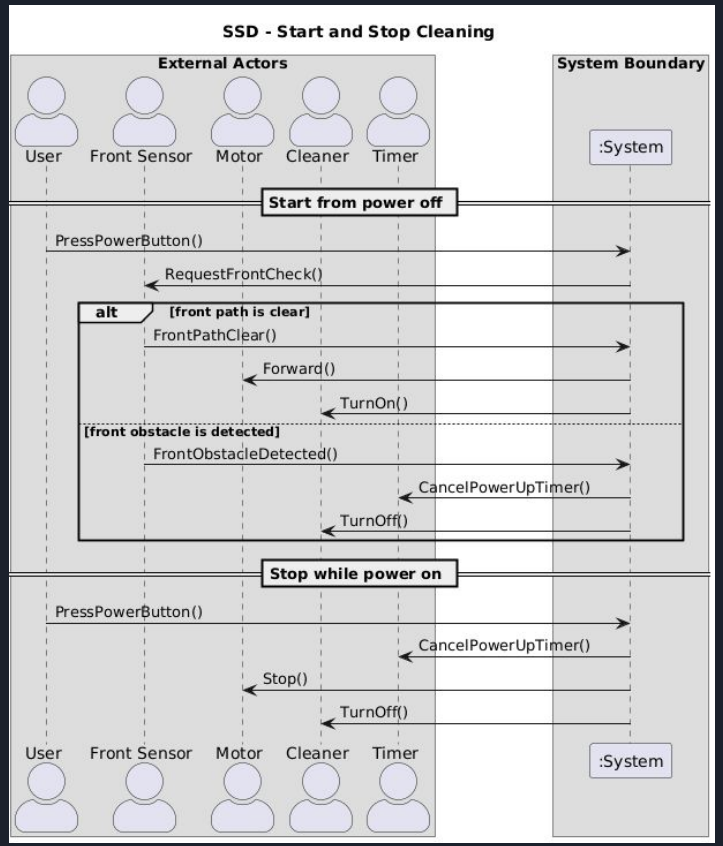
1. 우선 기능 요구사항을 정의하고, 그걸 기반으로 Use Cases들을 만들어 볼 생각이야.
   처음에는 rvc_controller의 전원이 꺼져 있고 사용자가 버튼을 누르면 rvc_controller의 작동이 시작돼. 전원이 켜진 rvc_controller는 전방에 장애물이 있는지 확인하고 장애물이 없다면 motor가 전진을 하고, cleaner가 켜져 청소가 시작될거야. 반대로 전원이 켜진 상태에서 사용자가 버튼을 누르면 motor가 움직임을 멈추고, cleaner도 꺼져 청소를 종료할거야
   청소도중, 혹은 전원이 켜졌을 때 전방에 장애물이 감지되면, 장애물 회피 프로세스가 시작될건데, 우선 좌측에 장애물이 없는지 확인하고 좌측에 장애물이 없다면 바로 좌회전을 할거야. 좌측에 장애물이 있는 경우 우측의 장애물도 확인할건데, 우측의 장애물이 없는 경우 우회전을 할거고 좌우측 둘다 장애물이 있다면 왼쪽이든 오른쪽이든 장애물이 없을 때까지 후진을 하고, 장애물이 없는 방향으로 회전할거야. 회전 혹은 후진하는 상황에서는 cleaner가 꺼져 청소를 수행하지는 않을 거고, 장애물 회피 프로세스가 종료되면 다시 전진을 하면서 cleaner가 켜질거야
   전진 도중 전방에 먼지가 감지되면 5초동안 cleaner의 흡입력을 증가시키고, 5초가 지나면 다시 cleaner의 흡입력을 정상화시킬거야. cleaner의 흡입력이 증가되었을 때 전방에 장애물이 감지되면 흡입력이 증가된 시간을 저장하지 않고 바로 cleaner를 끌거야.
   이를 토대로 FR, NFR, Use case를 뽑아주고 usecase diagram, System Sequence Diagrams, Domain Model을 만들어서 시각화해줘. plantuml 코드로 저장해줘도 돼.

2. 로봇 청소기 프로젝트를 위한 간단한 Jira 요구사항 관리 구조를 만들어줘.

조건:

- Jira는 요구사항 추적 및 상태 관리 용도로만 사용할 것
- workflow는 최대한 단순하게 구성할 것
- Requirement, Task, Bug 정도의 이슈 타입만 사용할 것
- 요구사항 네이밍 규칙도 포함할 것
- GitHub 브랜치 네이밍 예시도 포함할 것
- 학생 프로젝트에 적합한 가벼운 agile 방식으로 설계할 것

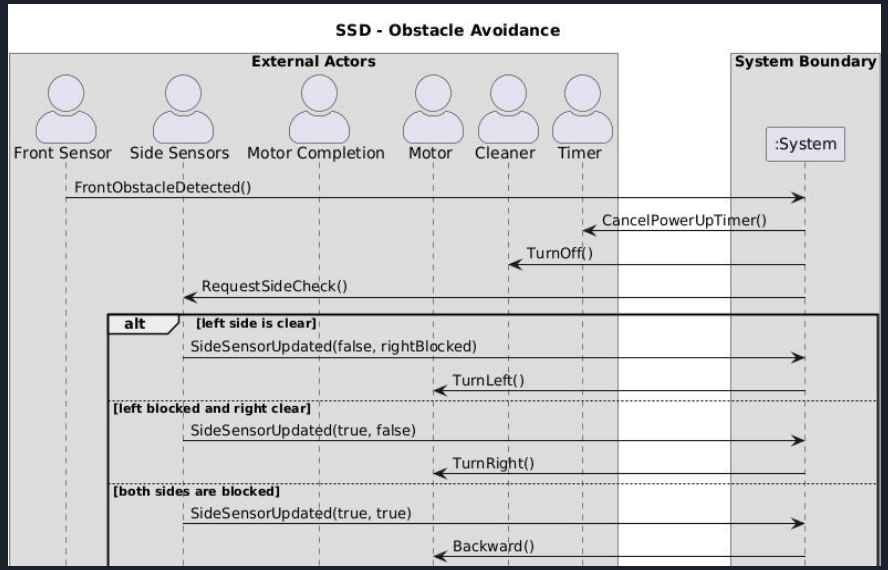
OOA - System Sequence Diagram

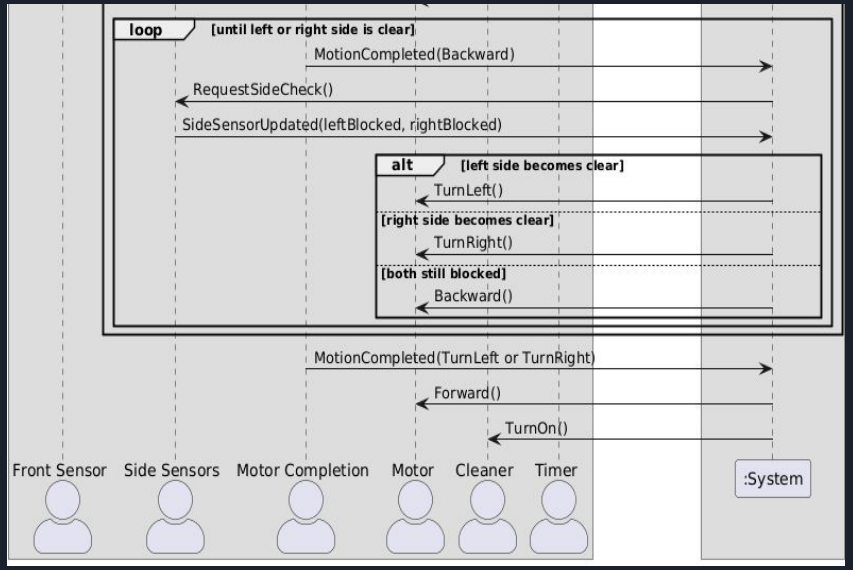


UC-001 & UC-002

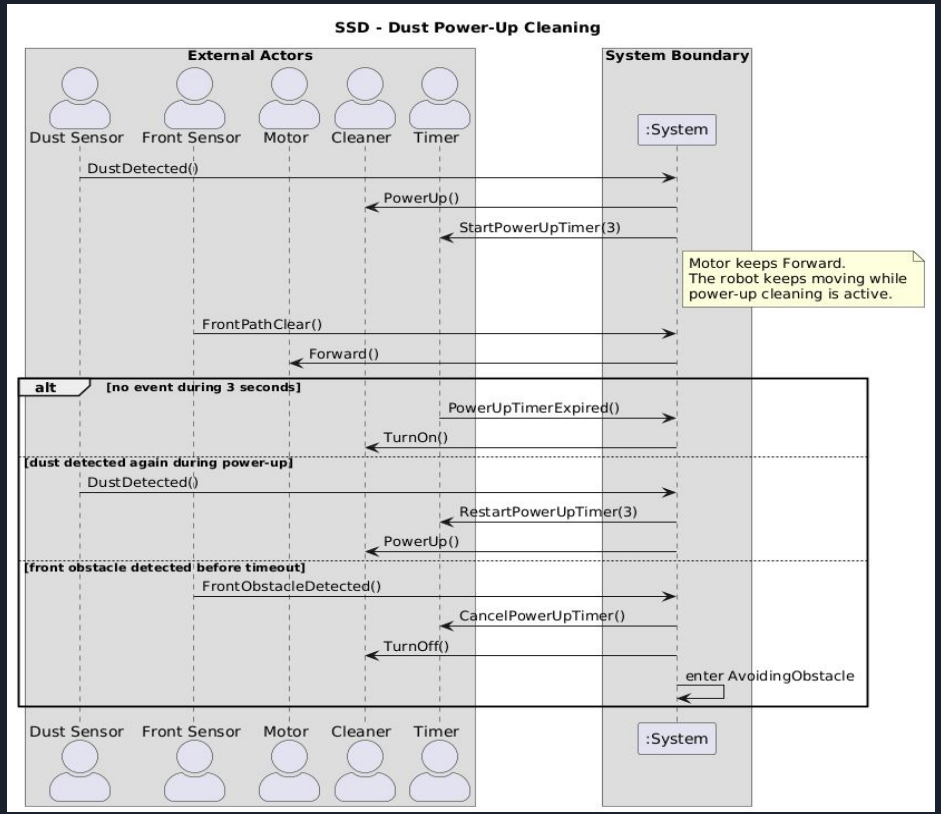
start cleaning & stop cleaning

OOA - System Sequence Diagram

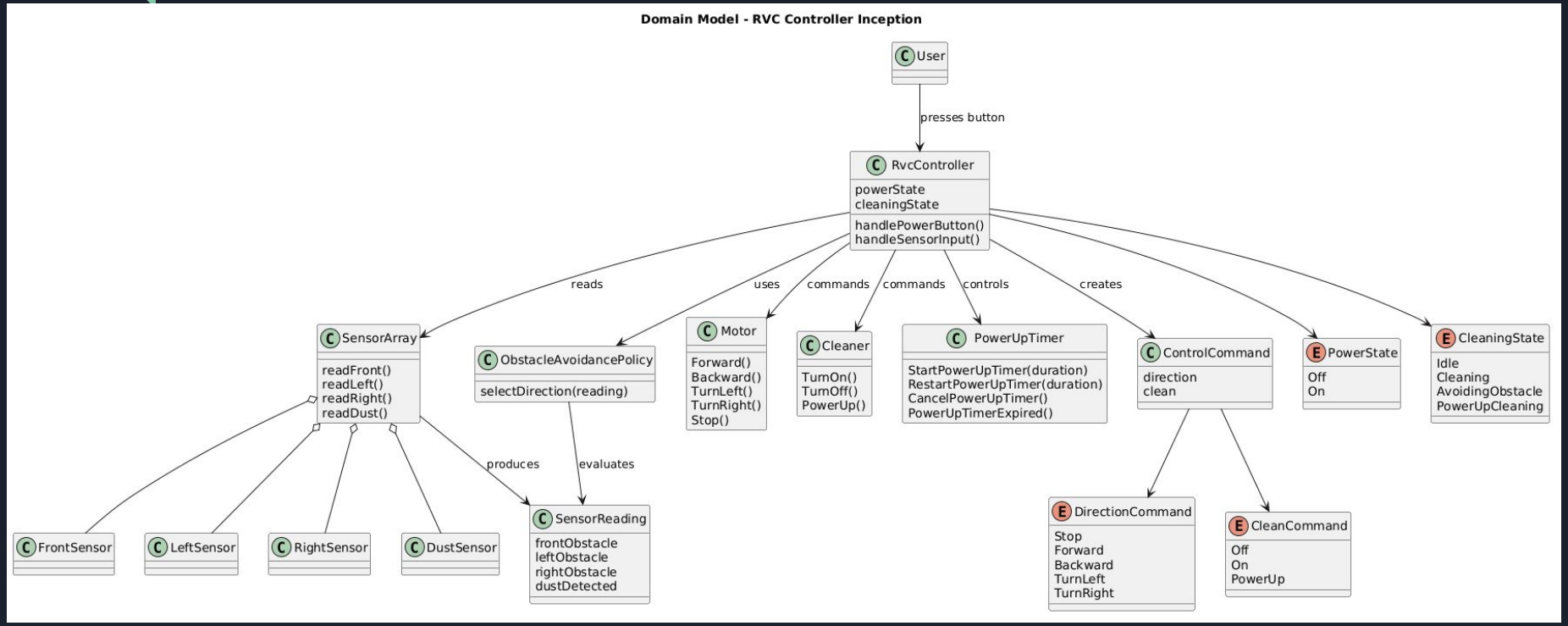




OOA - System Sequence Diagram



OOA - Domain model

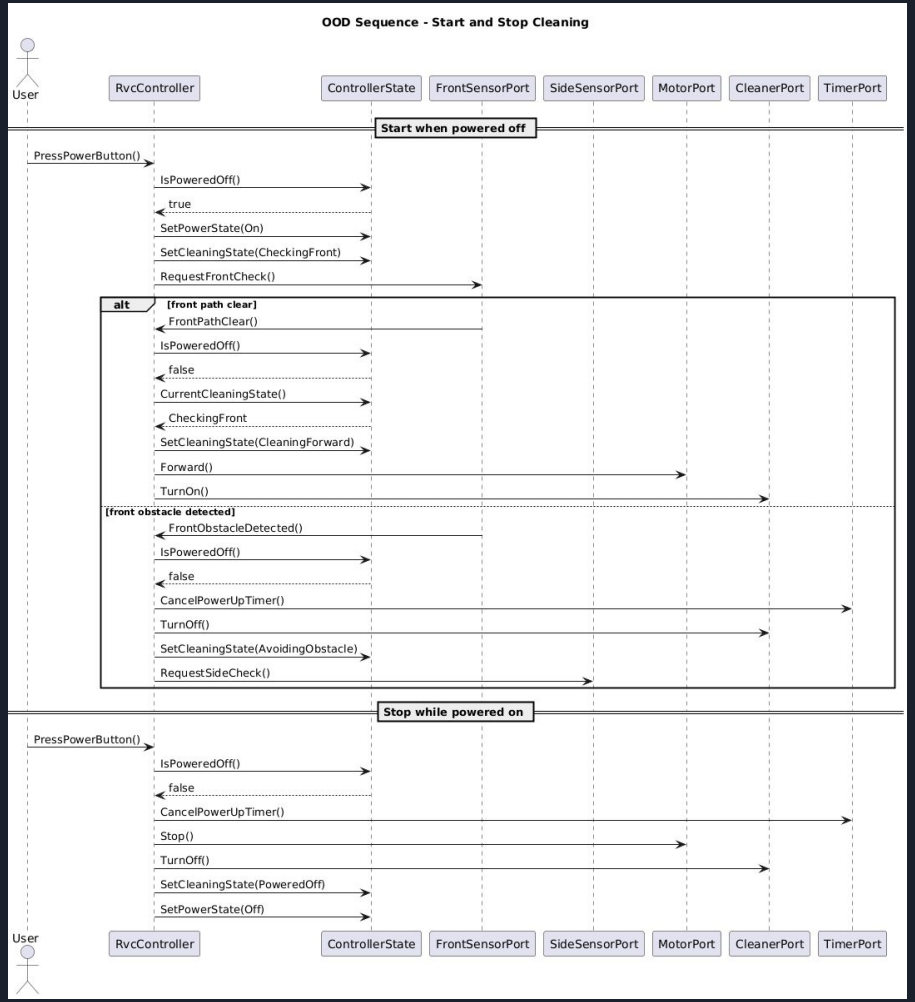


Step3. OOD

Prompt - Step3. OOD

프롬프트 3 (Use cases 리파인 + Sequence diagrams + Class Diagram + ~~SOLID 분석 및 적용~~)

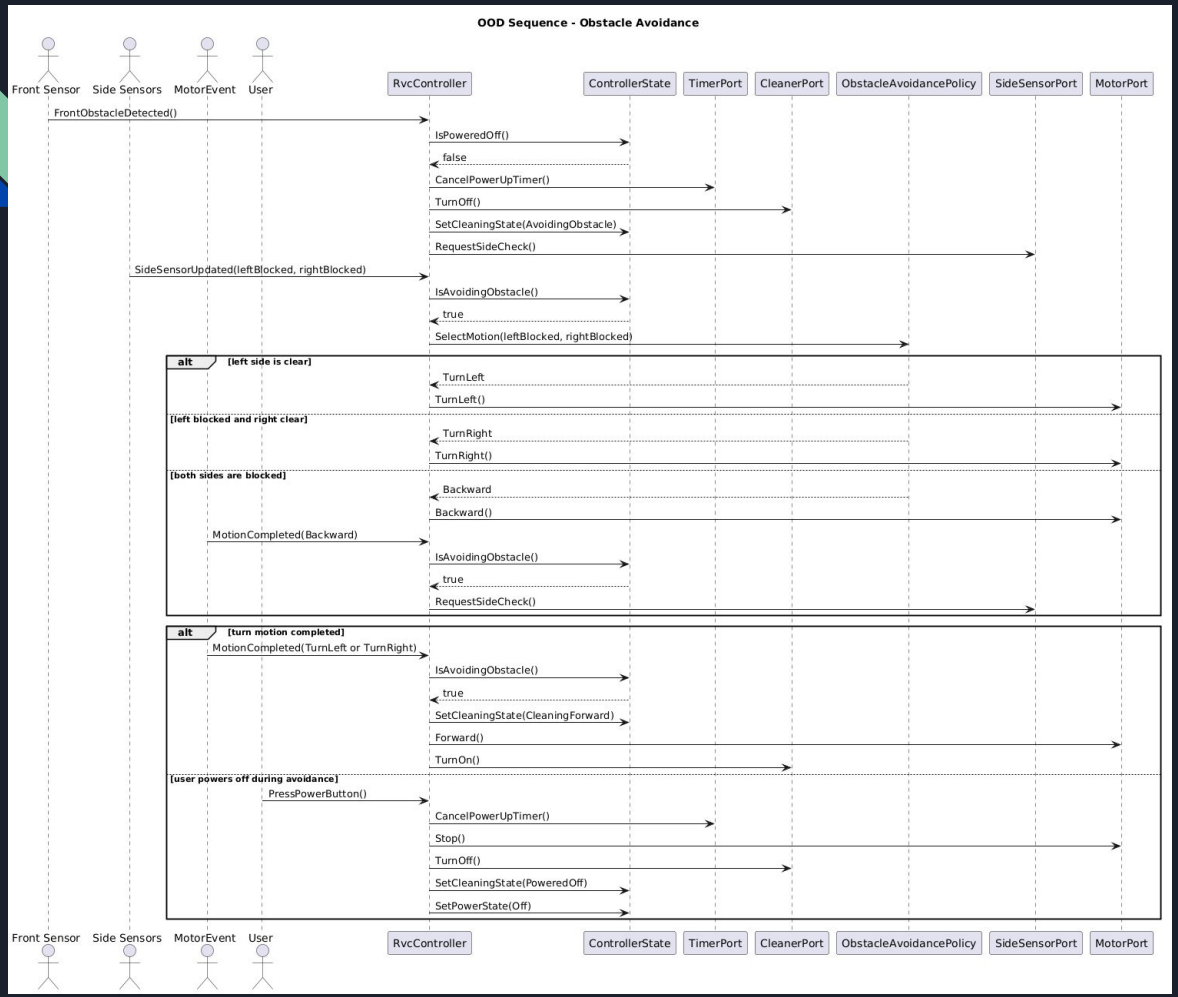
이전에 만들어 놓은 유즈케이스와 시스템 시퀀스 다이어그램 을 기반으로 각 유즈케이스를 구체화 할거야. 예를 들면, 먼지가 감지되어 청소기가 이미 파워업 되어있는데 또 먼지를 발견되는 중복되는 상황에서는 타이머를 초기화하고 새로 카운트 하는등의 예외 사항만 기반으로 전제해놓거나 추가하는 등으로 구체화 하면 되고, 또 회전중에는 먼지를 발견해도 청소기가 켜지지 않는다, 장애물을 발견하면 바로 청소기를 오프 상태로 전환하고 장애물 대응 과정으로 넘어가야 한다 등의 실행중 우선순위 정의를 제대로 하는 등의 구체화를 하는 작업이야. 만약 논리상 허점이나 오류가 발견되면 알려줘. 어떻게 시스템 시퀀스 다이어그램을 기반으로 유스케이스 구체화 하면서 ood단계의 시퀀스 다이어그램과 클래스 다이어그램을 완성시킬거야. 필요한 시스템오퍼레이션을 생각해내면서 객체지향 개발의 조건과 어긋나지 않게 설계해줘. 시스템 오퍼레이션은 controller에 들어오는 operation만을 일컫는 용어야. 이전에 분석했던 기능 요구사항이 녹아들도록 만들어줘. 모든 다이어그램은 plantuml 코드로 저장해줘.



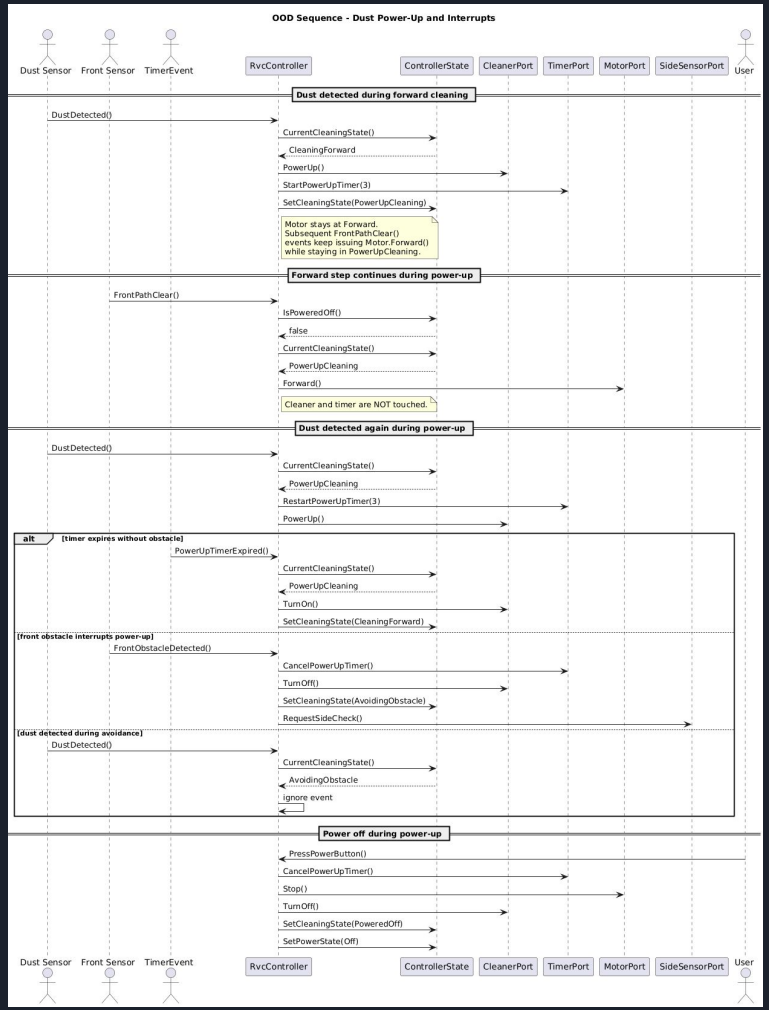
UC-001 & UC-002

SSD와 마찬가지로 UC-001와 UC-002을 통합

UC-001 : Start CLeaning
UC-002 : Stop Cleaning

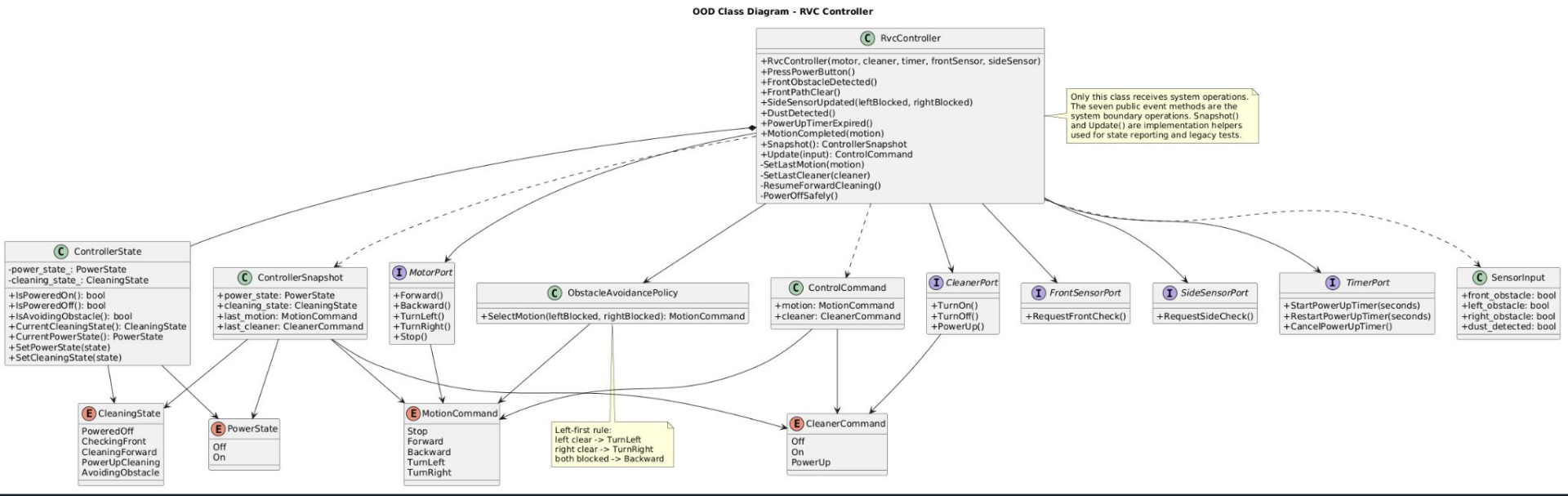

OOD Sequence - Obstacle Avoidance
Front Sensor
Side Sensors
MotorEvent
User
RvcController
ControllerState
TimerPort
CleanerPort
ObstacleAvoidancePolicy
SideSensorPort
MotorPort
FrontObstacleDetected()
IsPoweredOff()
false
CancelPowerUpTimer()
TurnOff()
SetCleaningState(AvoidingObstacle)
RequestSideCheck()
SideSensorUpdated(leftBlocked, rightBlocked)
IsAvoidingObstacle()
true
SelectMotion(leftBlocked, rightBlocked)
alt
[left side is clear]
TurnLeft
TurnLeft()
[left blocked and right clear]
TurnRight
TurnRight()
[both sides are blocked]
Backward
Backward()
MotionCompleted(Backward)
IsAvoidingObstacle()
true
RequestSideCheck()
alt
[turn motion completed]
MotionCompleted(TurnLeft or TurnRight)
IsAvoidingObstacle()
true
SetCleaningState(CleaningForward)
Forward()
TurnOn()
[user powers off during avoidance]
PressPowerButton()
CancelPowerUpTimer()
Stop()
TurnOff()
SetCleaningState(PoweredOff)
SetPowerState(Off)

UC-003


OOD Sequence - Dust Power-Up and Interrupts
Dust Sensor
Front Sensor
TimerEvent
RvcController
ControllerState
CleanerPort
TimerPort
MotorPort
SideSensorPort
User
Dust detected during forward cleaning
DustDetected()
CurrentCleaningState()
CleaningForward
PowerUp()
StartPowerUpTimer(3)
SetCleaningState(PowerUpCleaning)
Motor stays at Forward.
Subsequent FrontPathClear()
events keep issuing Motor.Forward()
while staying in PowerUpCleaning.
Forward step continues during power-up
FrontPathClear()
IsPoweredOff()
false
CurrentCleaningState()
PowerUpCleaning
Forward()
Cleaner and timer are NOT touched.
Dust detected again during power-up
DustDetected()
CurrentCleaningState()
PowerUpCleaning
RestartPowerUpTimer(3)
PowerUp()
alt
[timer expires without obstacle]
PowerUpTimerExpired()
CurrentCleaningState()
PowerUpCleaning
TurnOn()
SetCleaningState(CleaningForward)
[front obstacle interrupts power-up]
FrontObstacleDetected()
CancelPowerUpTimer()
TurnOff()
SetCleaningState(AvoidingObstacle)
RequestSideCheck()
[dust detected during avoidance]
DustDetected()
CurrentCleaningState()
AvoidingObstacle
ignore event
Power off during power-up
PressPowerButton()
CancelPowerUpTimer()
Stop()
TurnOff()
SetCleaningState(PoweredOff)
SetPowerState(Off)

UC-004

OOD - Class Diagram



Step4. OOI

Prompt - Step4. OOI

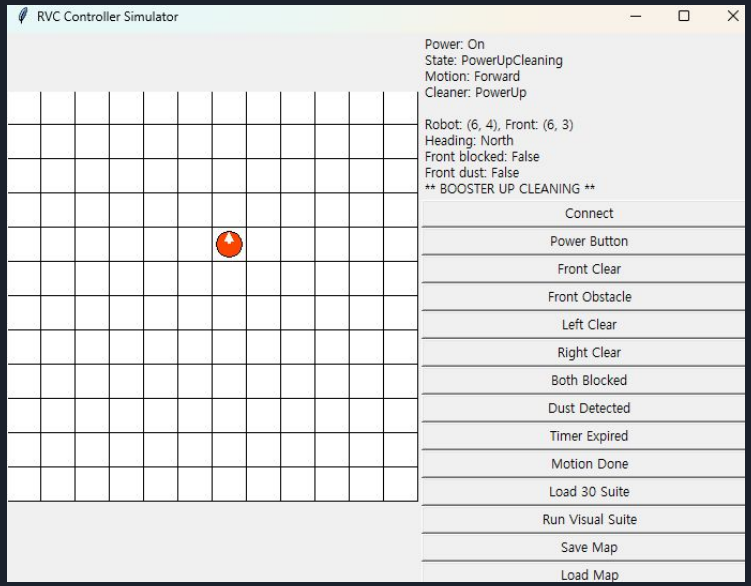
프롬프트 4 (시뮬레이터 + 코드 구현 + Unit / System Test 생성 및 실행 (Google Test) + Code Review (수동) + Static Code Analysis(Cppcheck + clang-tidy) + sonarcloud code analysis로 결과 시각화)

이 로봇 청소기 소프트웨어의 코드를 구현할거야. 시스템 오퍼레이션의 구성과 구조는 시퀀스 다이어그램과 클래스 다이어그램을 기반으로 다이어그램들과 어긋나는점이 없도록 만들거고, 시뮬레이터는 파이썬기반으로 ui가 보이게 만들어서 먼지, 장애물을 좌클릭, 우클릭으로 배치해서 수동으로 조합해서 직접 맵을 만들 수 있도록 하는 기능을 포함하고 실제 진행 상황을 우리가 직접 볼 수 있도록 c++소스 코드와 클라이언트 서버 구조로 통신하면서 상황이 업데이트 되도록 만들어줘. 유닛테스트는 구글테스트 기반으로 유닛테스트를 만들거고, 유닛테스트에 포함될 대상은 시뮬레이터를 제외한 로봇청소기 소프트웨어의 소스코드야. 라인커버리지로 90%이상이 커버되도록 만들어 주고, 객체 속 함수가 하나하나 잘 작동되는지, 그 함수에서 다른 함수나 객체 호출을 의도한 대로 하는지 체크하는 테스트가 있으면 좋겠어. 정적분석은 cppcheck, clang-tidy를 로컬에서 먼저 작동해줘서 코드 스멜과 잠재적 오류 요소를 찾아줘. 시스템 테스트는 구글테스트 기반으로 하지 않아도 되고, 각 테스트 케이스와 해당 케이스를 테스트하기 적절한 맵 파일을 만들어줘. Cmakelist파일도 유닛 테스트용, 실제 프로그램 실행용에 해당하는 파일들을 모아서 구성해줘. 마지막으로 유닛테스트 결과와 정적분석 결과를 종합해서 보고서를 만들어주는 소나클라우드 서비스를 이용해서 결과 정리해줘. 그리고 코드 파일들의 구조는 시뮬레이터 디렉터리 안에 시뮬레이터 관련 파이썬 코드들과 파이썬과 로봇청소기 소스 코드와 통신하는 함수를 담은 c++파일이 들어가면 되고 소스코드 디렉터리에는 소스코드와 헤더파일이, 그리고 테스트 디렉터리는 유닛테스트에 필요한 테스트코드와 테스트에 필요한 헤더와 인클루드, 모킹해야할 함수들을 모아놓은 헤더파일을 만들어줘.

Step4-1. OOI 수정

문제 발생

- 시뮬레이터가 맵 환경과 상호작용을 하는것이 아닌 버튼클리으로 주어지는 입력에만 반응
- 위치 초기화, 버튼 클릭에 대한 여러 버그 발생
- 의도와 다르게 청소기가 먼지 발견 타이밍 등에서 1틱씩 멈추는 상황 발생



Step4-1. OOI 수정

- 스텝에서 인풋값을 주입하는 방식이 아닌 맵/환경기반 시뮬레이션 테스트도 가능하도록 수정

runners로 시스템 테스트를 테스트 해볼때는 지금처럼 하는게 좋을 것 같은데, 시각화해서 시뮬레이터에서 돌려볼때에는 map-driven system test로 바꿔줄 수 있어?

- 시뮬레이터 버그수정

우리는 시뮬레이터 관련 코드만 수정할거야

처음에 power button을 누르면 RVC가 이상한 위치로 이동하는 버그가 있어, 제자리에서 On이 되도록 고쳐줘

전진할 때와 후진할 때 다음칸으로 넘어가는 시간을 1초로 잡아주고, 좌회전 우회전할 때 걸리는 시간도 1초로 잡아줘, 그리고 클리너가 booster up 모드가 되는건 3초 동안으로 해주고 중요한건 booster up 모드에서 정면 장애물을 만나서 회피상태가 되지 않는 한 전진상태가 멈추면 안되. 지금은 RVC가 먼지를 감지하면 멈춰버려.

Implementation

```cpp
class MotorPort {
 public:
  virtual ~MotorPort() = default;
  virtual void Forward() = 0;
  virtual void Backward() = 0;
  virtual void TurnLeft() = 0;
  virtual void TurnRight() = 0;
  virtual void Stop() = 0;
};

class CleanerPort {
 public:
  virtual ~CleanerPort() = default;
  virtual void TurnOn() = 0;
  virtual void TurnOff() = 0;
  virtual void PowerUp() = 0;
};

class TimerPort {
 public:
  virtual ~TimerPort() = default;
  virtual void StartPowerUpTimer(int seconds) = 0;
  virtual void RestartPowerUpTimer(int seconds) = 0;
  virtual void CancelPowerUpTimer() = 0;
};

class FrontSensorPort {
 public:
  virtual ~FrontSensorPort() = default;
  virtual void RequestFrontCheck() = 0;
};

class SideSensorPort {
 public:
  virtual ~SideSensorPort() = default;
  virtual void RequestSideCheck() = 0;
};
```

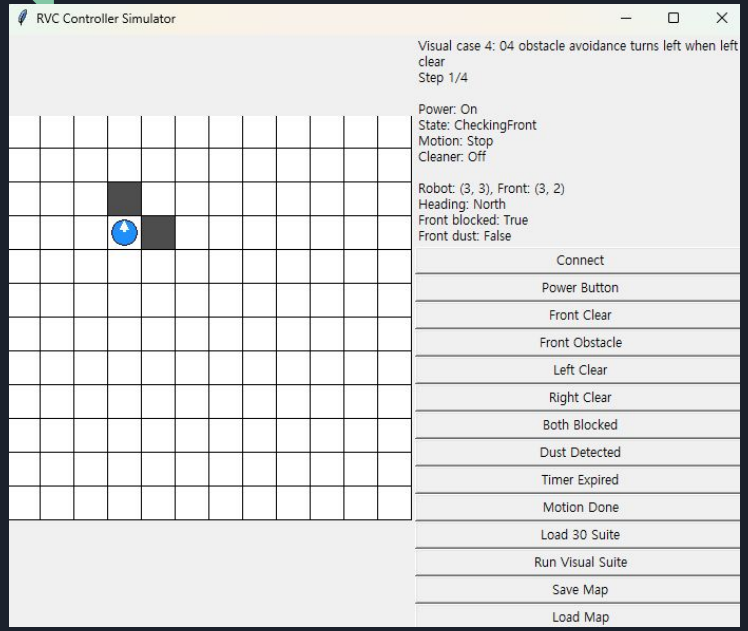
```cpp
class ControllerState {
 public:
  bool IsPoweredOn() const;
  bool IsPoweredOff() const;
  bool IsAvoidingObstacle() const;
  CleaningState CurrentCleaningState() const;
  PowerState CurrentPowerState() const;
  void SetPowerState(PowerState state);
  void SetCleaningState(CleaningState state);

 private:
  PowerState power_state_{PowerState::Off};
  CleaningState cleaning_state_{CleaningState::PoweredOff};
};

class ObstacleAvoidancePolicy {
 public:
  MotionCommand SelectMotion(bool left_blocked, bool right_blocked) const;
};

class RvcController {
 public:
  RvcController(MotorPort& motor, CleanerPort& cleaner, TimerPort& timer,
                FrontSensorPort& front_sensor, SideSensorPort& side_sensor);

  void PressPowerButton();
  void FrontObstacleDetected();
  void FrontPathClear();
  void SideSensorUpdated(bool left_blocked, bool right_blocked);
  void DustDetected();
  void PowerUpTimerExpired();
  void MotionCompleted(MotionCommand motion);

  ControllerSnapshot Snapshot() const;

  // Compatibility helper for simple one-shot decisions used by early tests.
  ControlCommand Update(const SensorInput& input) const;

 private:
  void SetLastMotion(MotionCommand motion);
  void SetLastCleaner(CleanerCommand cleaner);
  void ResumeForwardCleaning();
  void PowerOffSafely();

  MotorPort& motor_;
  CleanerPort& cleaner_;
  TimerPort& timer_;
  FrontSensorPort& front_sensor_;
  SideSensorPort& side_sensor_;
  ControllerState state_;
  ObstacleAvoidancePolicy avoidance_policy_;
  MotionCommand last_motion_{MotionCommand::Stop};
  CleanerCommand last_cleaner_{CleanerCommand::Off};
};
```

OOD 산출물(Class Diagram, SD)을 기반으로 C++ 구현

OOD 시스템 오퍼레이션과 코드 매핑
- 7개 system operation이 RvcController에 1:1 구현

OOD 단계에서 정의한 RvcController, 상태 객체, 정책 객체, Port Interface를 C++ 클래스와 추상 인터페이스로 그대로 구현

Simulator / Execution Environment



Python Tkinter UI로 RVC 동작 시각화

grid map 기반:

obstacle 표시, dust 표시,
robot position, heading arrow

상태별 색상:

Blue: normal cleaning
Orange-red: booster up
Purple: obstacle avoidance
Gray: powered off

30개 시스템 테스트 케이스를 시각적으로 replay

Unit Testing

```
PS C:\RVC_OOAD> ctest --test-dir build --output-on-failure
Internal ctest changing into directory: C:/RVC_OOAD/build
Test project C:/RVC_OOAD/build
      Start  1: RvcControllerTest.StartsByRequestingFrontCheck
 1/24 Test  #1: RvcControllerTest.StartsByRequestingFrontCheck ....................................   Passed    0.01 sec
      Start  2: RvcControllerTest.StartsForwardCleaningWhenFrontPathIsClear
 2/24 Test  #2: RvcControllerTest.StartsForwardCleaningWhenFrontPathIsClear .......................   Passed    0.01 sec
      Start  3: RvcControllerTest.PowerButtonStopsEverythingWhenPoweredOn
 3/24 Test  #3: RvcControllerTest.PowerButtonStopsEverythingWhenPoweredOn ..........................   Passed    0.01 sec
      Start  4: RvcControllerTest.FrontObstacleInterruptsPowerUpAndRequestsSideCheck
 4/24 Test  #4: RvcControllerTest.FrontObstacleInterruptsPowerUpAndRequestsSideCheck ..............   Passed    0.01 sec
      Start  5: RvcControllerTest.AvoidanceTurnsLeftWhenLeftIsClear
 5/24 Test  #5: RvcControllerTest.AvoidanceTurnsLeftWhenLeftIsClear ................................   Passed    0.01 sec
      Start  6: RvcControllerTest.AvoidanceTurnsRightWhenLeftBlockedAndRightClear
 6/24 Test  #6: RvcControllerTest.AvoidanceTurnsRightWhenLeftBlockedAndRightClear .................   Passed    0.01 sec
      Start  7: RvcControllerTest.AvoidanceMovesBackwardWhenBothSidesBlocked
 7/24 Test  #7: RvcControllerTest.AvoidanceMovesBackwardWhenBothSidesBlocked ......................   Passed    0.01 sec
      Start  8: RvcControllerTest.BackwardCompletionRequestsSideCheckAgain
 8/24 Test  #8: RvcControllerTest.BackwardCompletionRequestsSideCheckAgain ........................   Passed    0.01 sec
      Start  9: RvcControllerTest.TurnCompletionResumesForwardCleaning
 9/24 Test  #9: RvcControllerTest.TurnCompletionResumesForwardCleaning ............................   Passed    0.01 sec
      Start 10: RvcControllerTest.DustPowerUpStartsThreeSecondTimerDuringForwardCleaning
10/24 Test #10: RvcControllerTest.DustPowerUpStartsThreeSecondTimerDuringForwardCleaning ..........   Passed    0.01 sec
      Start 11: RvcControllerTest.DuplicateDustRestartsThreeSecondTimerAndKeepsState
11/24 Test #11: RvcControllerTest.DuplicateDustRestartsThreeSecondTimerAndKeepsState ..............   Passed    0.01 sec
      Start 12: RvcControllerTest.FrontPathClearDuringPowerUpKeepsStateAndDrivesForward
12/24 Test #12: RvcControllerTest.FrontPathClearDuringPowerUpKeepsStateAndDrivesForward ...........   Passed    0.01 sec
      Start 13: RvcControllerTest.CleanerOnlyEventsDoNotIssueMotorCommandsDuringPowerUpFlow
13/24 Test #13: RvcControllerTest.CleanerOnlyEventsDoNotIssueMotorCommandsDuringPowerUpFlow .......   Passed    0.01 sec
      Start 14: RvcControllerTest.DustIsIgnoredDuringAvoidance
14/24 Test #14: RvcControllerTest.DustIsIgnoredDuringAvoidance .....................................   Passed    0.01 sec
      Start 15: RvcControllerTest.TimerExpiryRestoresNormalCleaning
15/24 Test #15: RvcControllerTest.TimerExpiryRestoresNormalCleaning ................................   Passed    0.01 sec
      Start 16: RvcControllerTest.EventsAreIgnoredWhenPoweredOff
16/24 Test #16: RvcControllerTest.EventsAreIgnoredWhenPoweredOff ...................................   Passed    0.01 sec
      Start 17: RvcControllerTest.NonMatchingStateEventsAreIgnored
17/24 Test #17: RvcControllerTest.NonMatchingStateEventsAreIgnored .................................   Passed    0.01 sec
      Start 18: RvcControllerTest.MotionCompletionIgnoresUnsupportedAvoidanceMotions
18/24 Test #18: RvcControllerTest.MotionCompletionIgnoresUnsupportedAvoidanceMotions ...............   Passed    0.01 sec
      Start 19: RvcControllerTest.TurnRightCompletionAlsoResumesForwardCleaning
19/24 Test #19: RvcControllerTest.TurnRightCompletionAlsoResumesForwardCleaning ....................   Passed    0.01 sec
      Start 20: RvcControllerTest.CompatibilityUpdateStillMapsSensorInput
20/24 Test #20: RvcControllerTest.CompatibilityUpdateStillMapsSensorInput ..........................   Passed    0.01 sec
      Start 21: RvcControllerTest.CompatibilityUpdateCoversAllBasicMotionChoices
21/24 Test #21: RvcControllerTest.CompatibilityUpdateCoversAllBasicMotionChoices ...................   Passed    0.01 sec
      Start 22: RvcControllerTest.ControllerStateAccessorsAndMutators
22/24 Test #22: RvcControllerTest.ControllerStateAccessorsAndMutators .............................   Passed    0.01 sec
      Start 23: RvcControllerTest.ObstacleAvoidancePolicySelectsLeftFirstThenRightThenBackward
23/24 Test #23: RvcControllerTest.ObstacleAvoidancePolicySelectsLeftFirstThenRightThenBackward ...   Passed    0.01 sec
      Start 24: RvcControllerTest.ToStringCoversAllEnumValues
24/24 Test #24: RvcControllerTest.ToStringCoversAllEnumValues ......................................   Passed    0.01 sec

100% tests passed, 0 tests failed out of 24

Total Test time (real) =   0.17 sec
PS C:\RVC_OOAD>
```

총 24개의 unit test를 작성하여 주요 기능과 분기 조건을 검증

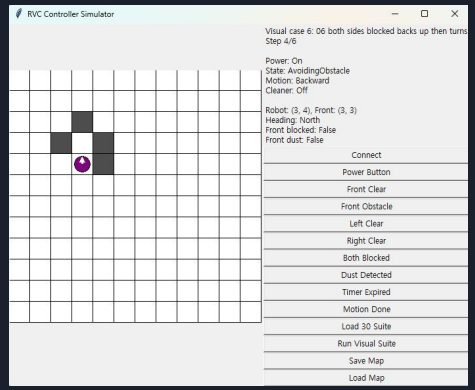
- GTest 기반으로 RVC 소프트웨어의 로직을 검증

- 실제 하드웨어 없이 테스트하기 위해 Motor, Cleaner, Timer, Sensor Port를 Fake 객체로 대체

- 단위 테스트는 시스템 오퍼레이션, 상태 전이, 예외 상황, 포트 호출 여부를 중심으로 구성

System Testing

```
PS C:\RVC_COAD> python system_tests\run_system_tests.py --suite system_tests\suites\rvc_30_system_tests.json --port 5050
Running 01 power on starts front check then forward cleaning
Running 02 power button stops active forward cleaning
Running 03 startup front obstacle enters avoidance
Running 04 obstacle avoidance turns left when left is clear
Running 05 obstacle avoidance turns right when only right is clear
Running 06 both sides blocked backs up then turns left
Running 07 both sides blocked backs up then turns right
Running 08 repeated backward while both sides remain blocked
Running 09 dust starts power-up timer
Running 10 power-up timer expiry restores normal cleaner
Running 11 repeated dust restarts power-up timer
Running 12 front obstacle interrupts active power-up
Running 13 dust is ignored while checking front
Running 14 dust is ignored during obstacle avoidance
Running 15 timer expiry is ignored outside power-up
Running 16 side sensor update ignored outside avoidance
Running 17 motion completion ignored outside avoidance
Running 18 front clear during power-up keeps power-up and moves forward
Running 19 power button stops active power-up safely
Running 20 power button stops obstacle avoidance safely
Running 21 front obstacle while cleaning enters avoidance
Running 22 front obstacle while checking enters avoidance
Running 23 front clear from avoidance resumes cleaning
Running 24 stop motion completion ignored during avoidance
Running 25 forward motion completion ignored during avoidance
Running 26 both side clear still prefers left
Running 27 repeated front clear keeps forward cleaning
Running 28 timer expiry after power off is ignored
Running 29 duplicate dust then timeout returns to normal
Running 30 combined obstacle recovery then dust power-up
Passed 30 system test cases
PS C:\RVC_COAD>
```



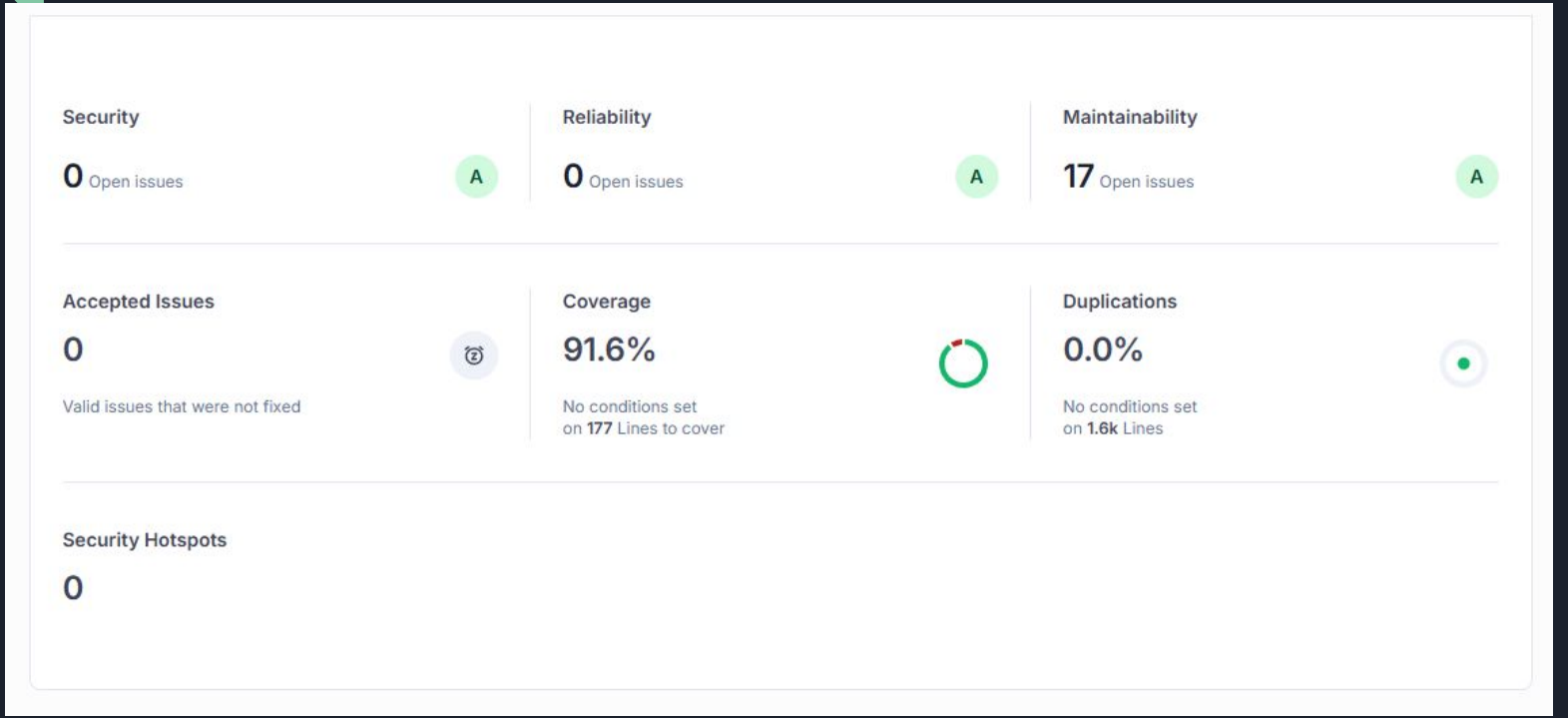
총 30개 system test case

두 가지 검증 방식:

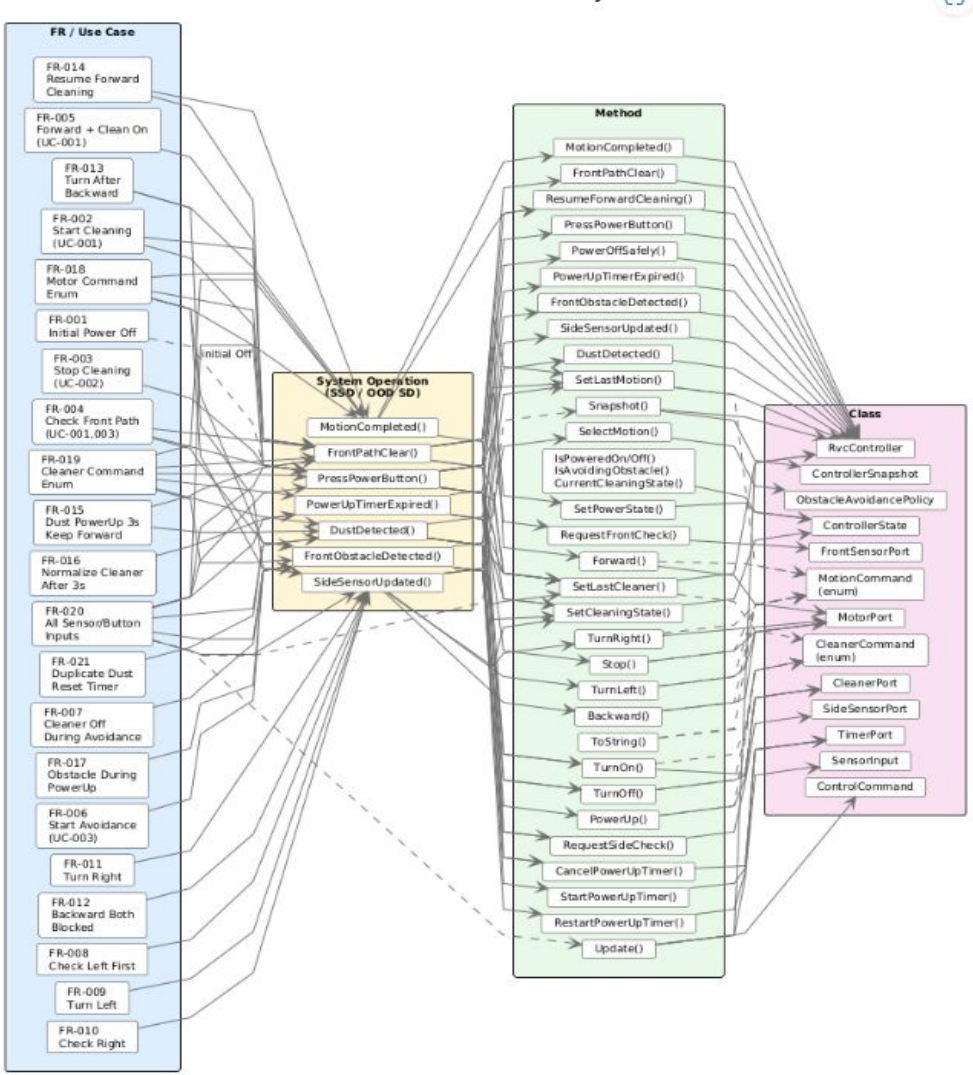
1. 자동 검증 방식
Python test runner가 C++ controller와 TCP로 통신하면서 각 테스트 케이스의 system operation을 순서대로 전송하고, controller가 반환한 snapshot을 기대 결과와 비교하여 pass/fail을 자동으로 판단

2.시각화 검증 방식
동일한 30개 테스트 케이스를 Python simulator UI에서 재생하여, RVC의 이동, 회전, 장애물 회피, booster up cleaning 상태를 눈으로 확인.
이 방식에서는 map 정보를 기반으로 sensor input을 생성하여 실제 시뮬레이션 환경과 유사하게 동작을 확인

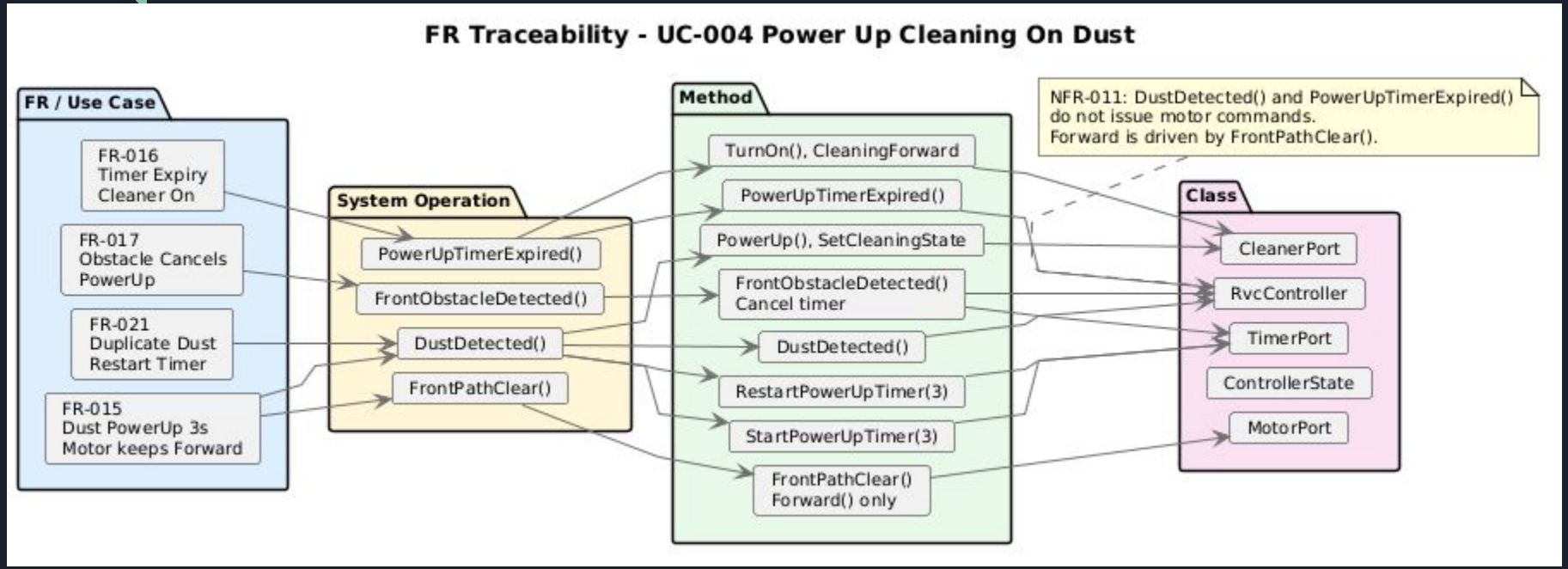
Static Code Analysis



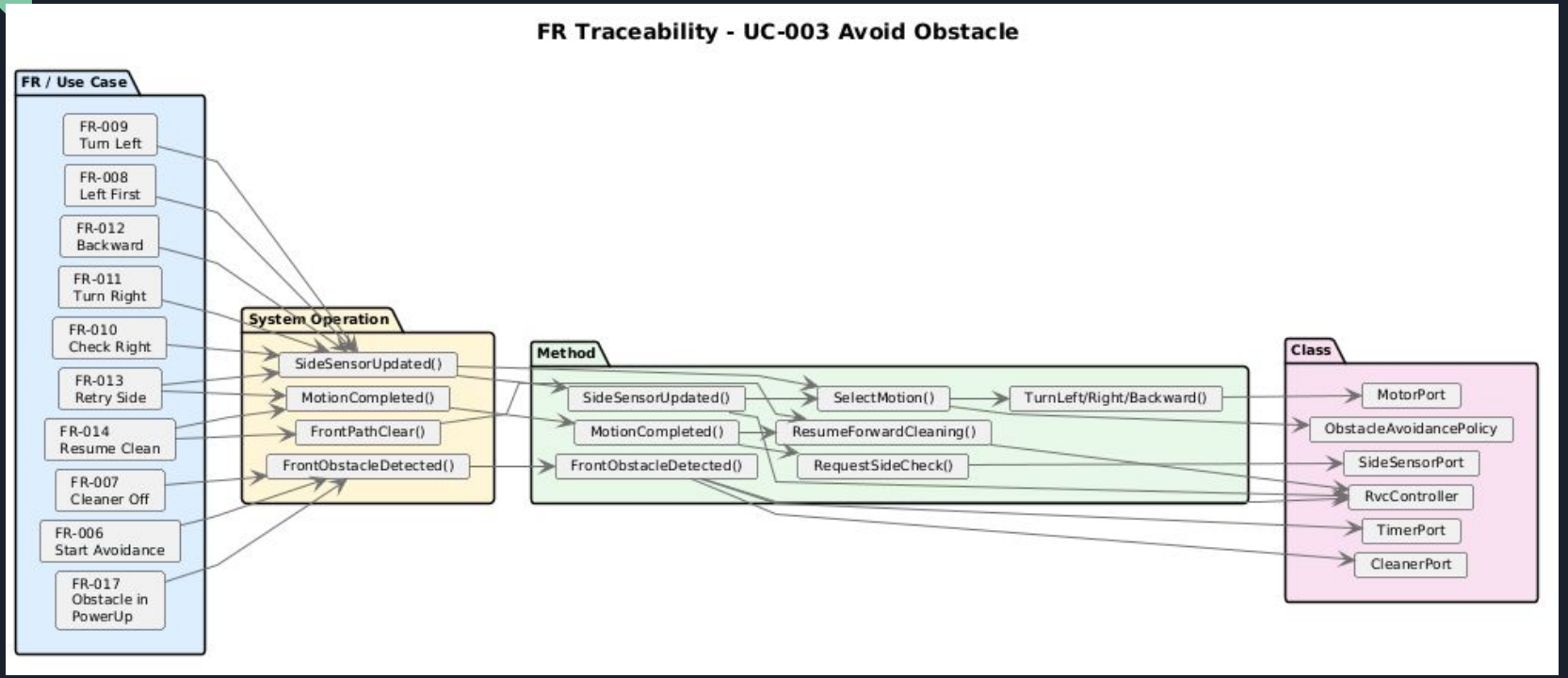
Traceability table



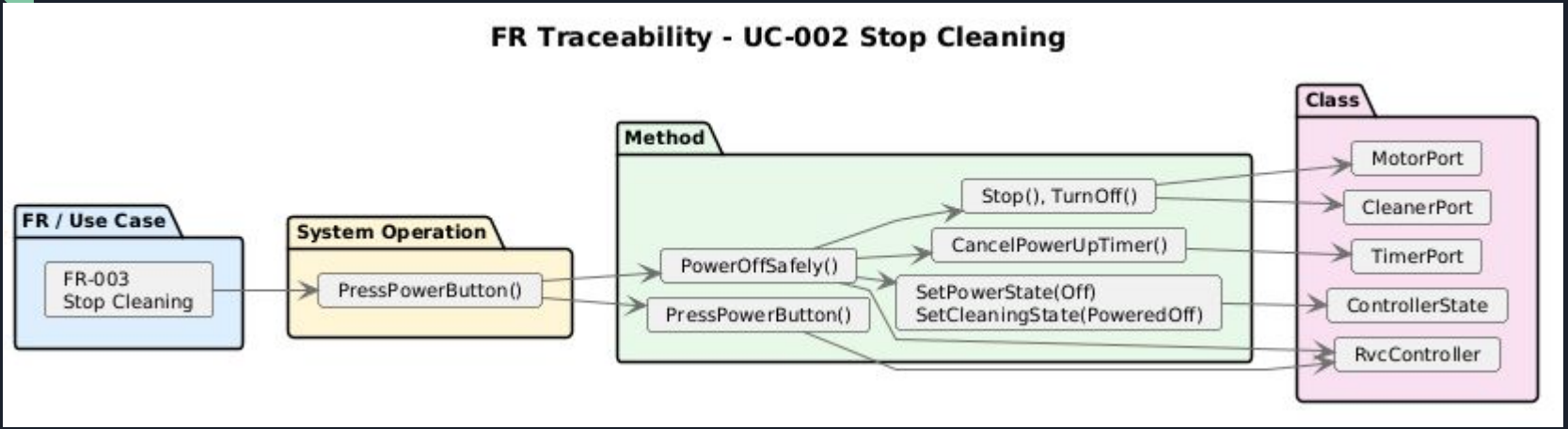
Traceability table



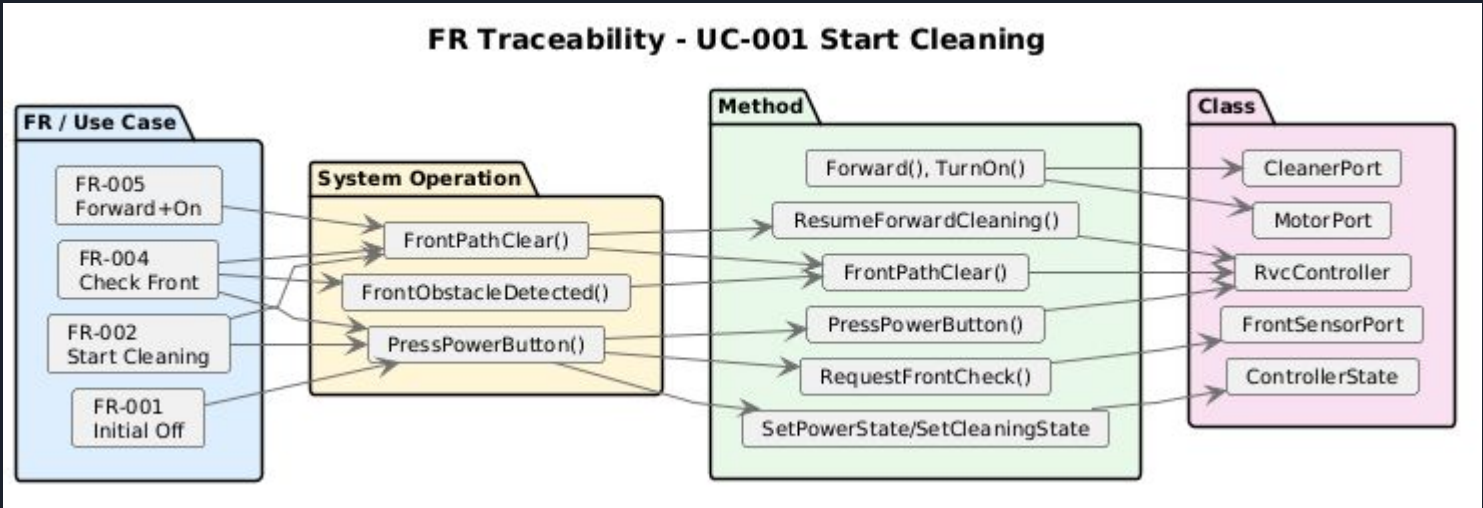
Traceability table



Traceability table



Traceability table



소감

요구사항을 명확히 정의하고 OOAD 과정에 맞춰 설계한 부분은 Cursor가 의도를 잘 반영해 코드를 작성해 주었고, 오류도 거의 없이 금방 끝냈습니다.

반면 simulator는 구체적인 요구사항 없이 바로 구현을 요청했기 때문에, 결과물이 저희 의도와 다르게 나와 수정 과정에서 어려움이 있었습니다.

감사합니다.