
객체지향개발방법론 (OOAD)

로봇 청소기(RVC) VIBE CODING

202211291 김호경

202111282 노을영

202111284 문재현

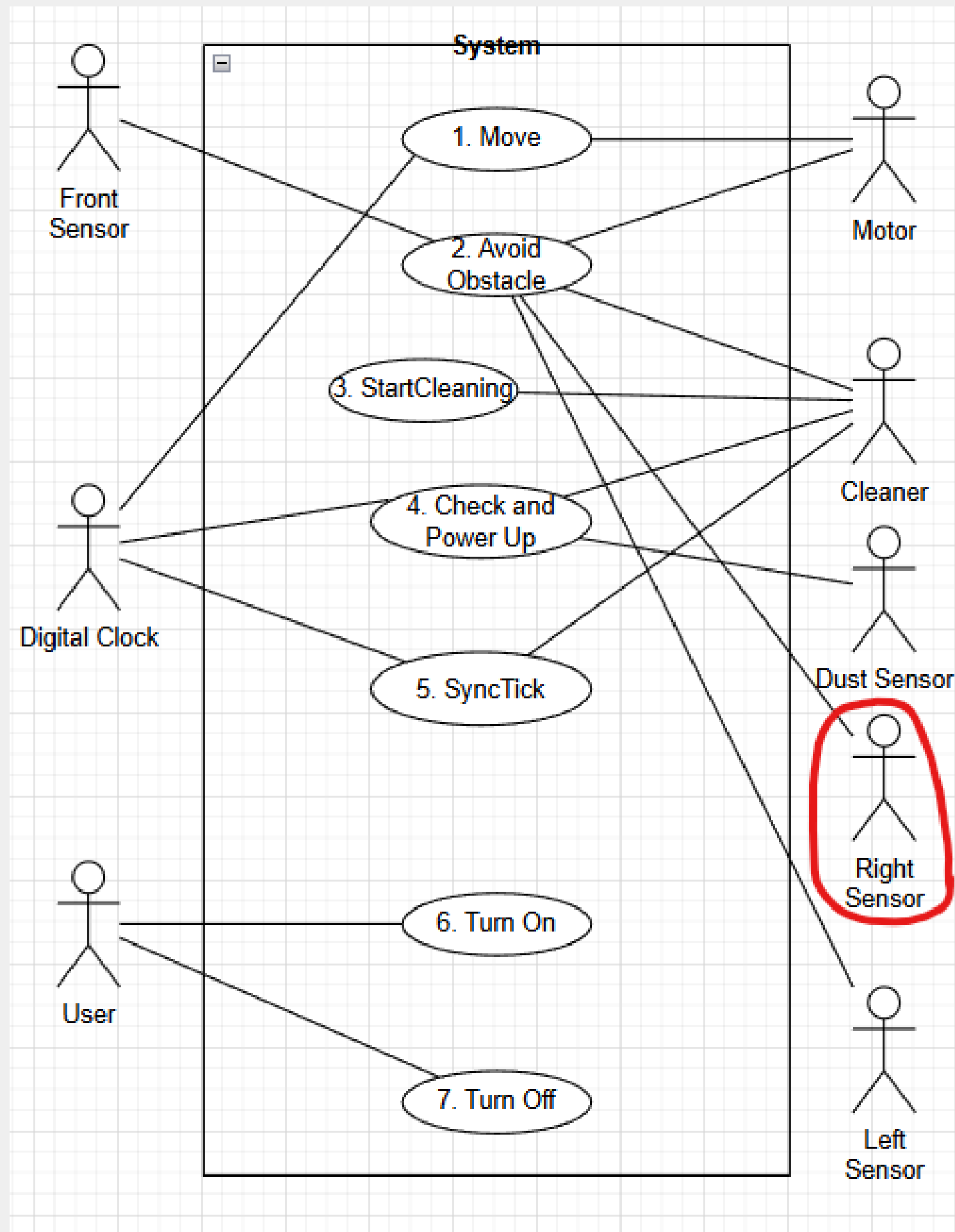
Agenda

- Oracle
 - UseCase
 - OOA
 - OOD
 - Code
 - Test
- Vibe Coding
 - UseCase
 - OOA
 - OOD
 - Code
 - Test
- V&V
- Summary

Oracle

Use Case Refined

Before

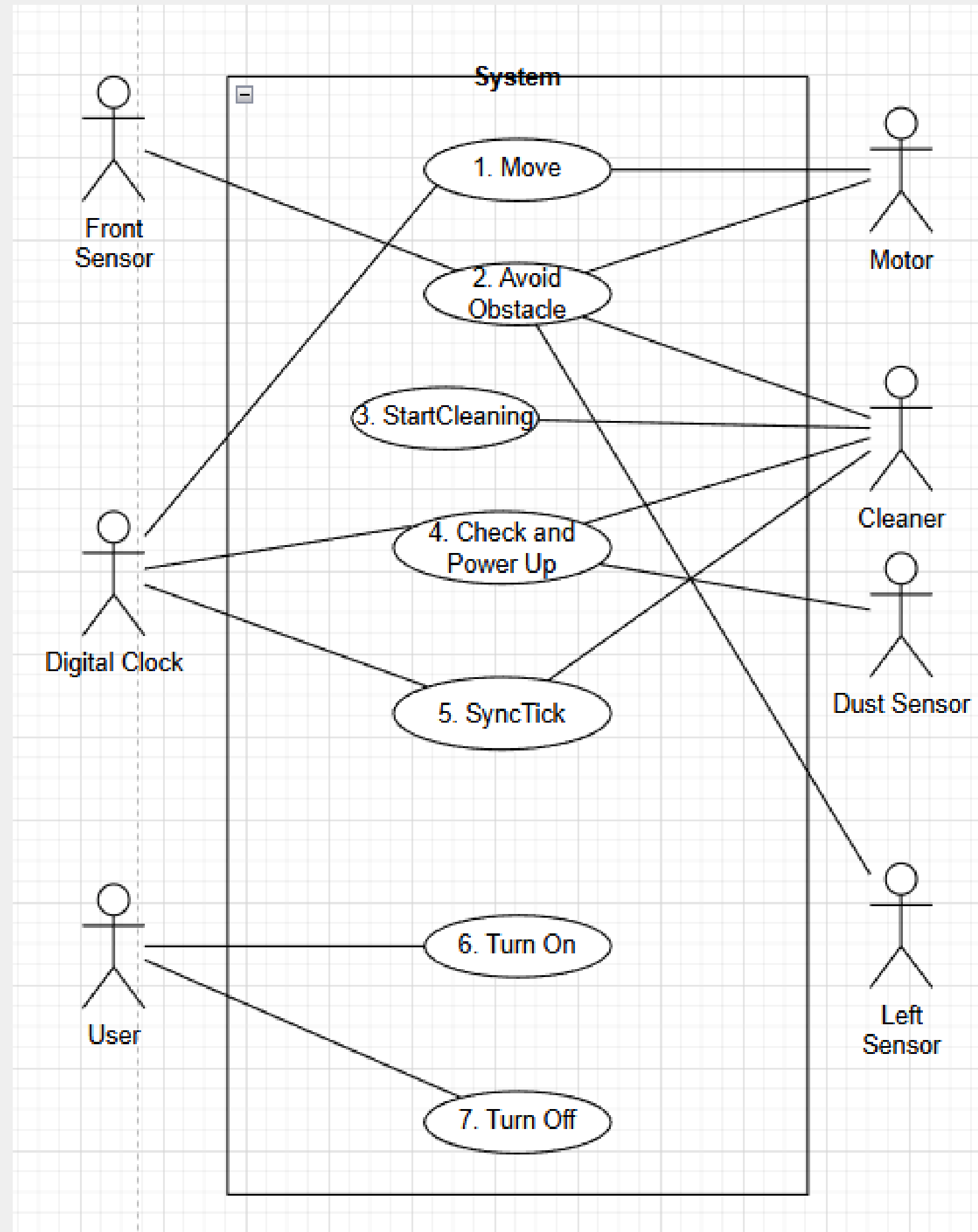


2. Avoid Obstacle

분류	내용
Use Case	2. RVC Turn Right
Actor	Front Sensor, Motor, Cleaner, Right Sensor
Pre-Requisites	(1) RVC의 전원이 켜져 있어야한다.
Typical Courses of Events	<ol style="list-style-type: none"> (A) Front Sensor는 앞의 장애물이 있음을 알린다. (S) RVC는 Cleaner를 꺼 청소를 중지한다. (S) RVC는 정지하도록 Motor를 조작한다. (S) RVC는 Right Sensor, Left Sensor에게 장애물 유무를 받아온다. (S) RVC는 받아온 장애물 유무를 기반으로 방향을 설정하고 이동한다. (S) 4번으로 돌아간다.
Alternative Courses of Events	<p>Line5.</p> <ol style="list-style-type: none"> 오른쪽에 장애물이 없다면 오른쪽으로 회전한다. 오른쪽에 장애물이 있고, 왼쪽에 장애물이 없다면 왼쪽으로 회전한다. 오른쪽에 장애물이 있고, 왼쪽에 장애물이 없다면 후진한다. <p>Line6. 후진하지 않았다면 4번으로 돌아가지 않고 UC-01을 수행한다.</p>
Exceptional Courses of Events	N/A

Use Case Refined

After

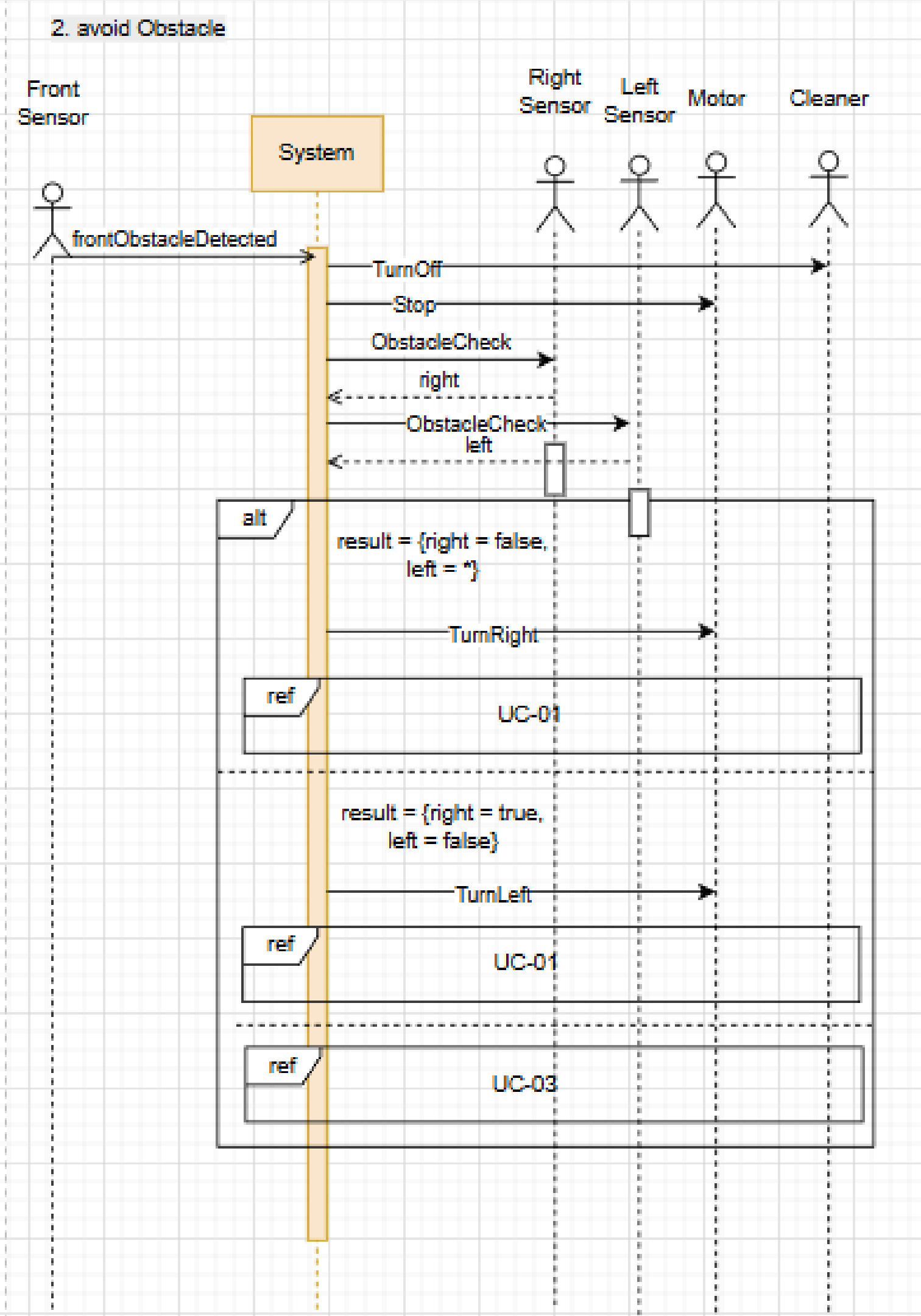
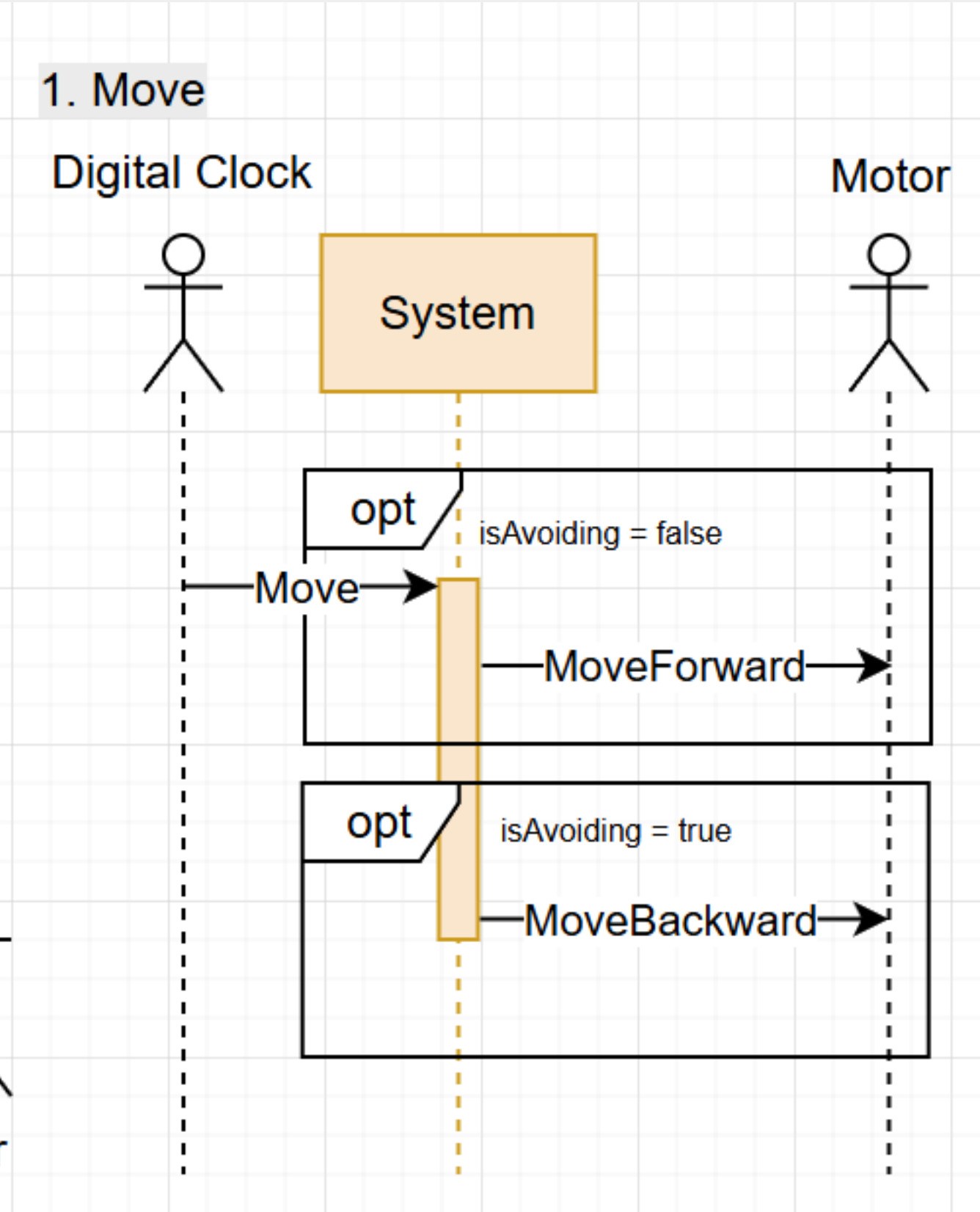


2. Avoid Obstacle

분류	내용
Use Case	2. Avoid Obstacle
Actor	Front Sensor, Motor, Cleaner, Left Sensor
Pre-Requisites	(1) RVC의 전원이 켜져 있어야한다.
Typical Courses of Events	<ol style="list-style-type: none"> (A) Front Sensor는 앞의 장애물이 있음을 알린다. (S) RVC는 Cleaner를 꺼 청소를 중지한다. (S) RVC는 정지하도록 Motor를 조작한다. (S) RVC는 Left Sensor에게 장애물 유무를 받아온다. (S) RVC는 받아들 장애물 유무를 기반으로 방향을 설정하고 이동한다. (S) 4번으로 돌아간다.
Alternative Courses of Events	<p>Line5.</p> <ol style="list-style-type: none"> 왼쪽에 장애물이 없다면 왼쪽으로 회전한다. 왼쪽에 장애물이 있다면 오른쪽으로 회전한다. 오른쪽 회전한 후 정면에 장애물이 있다면 왼쪽으로 다시 돈 후 후진한다. 위 과정을 Tick마다 반복한 후 장애물이 없는 방향이 발견되면 그 방향으로 회전 후 UC1로 전환한다. <p>Line6. 후진하지 않았다면 4번으로 돌아가지 않고 UC-01을 수행한다.</p>
Exceptional Courses of Events	N/A

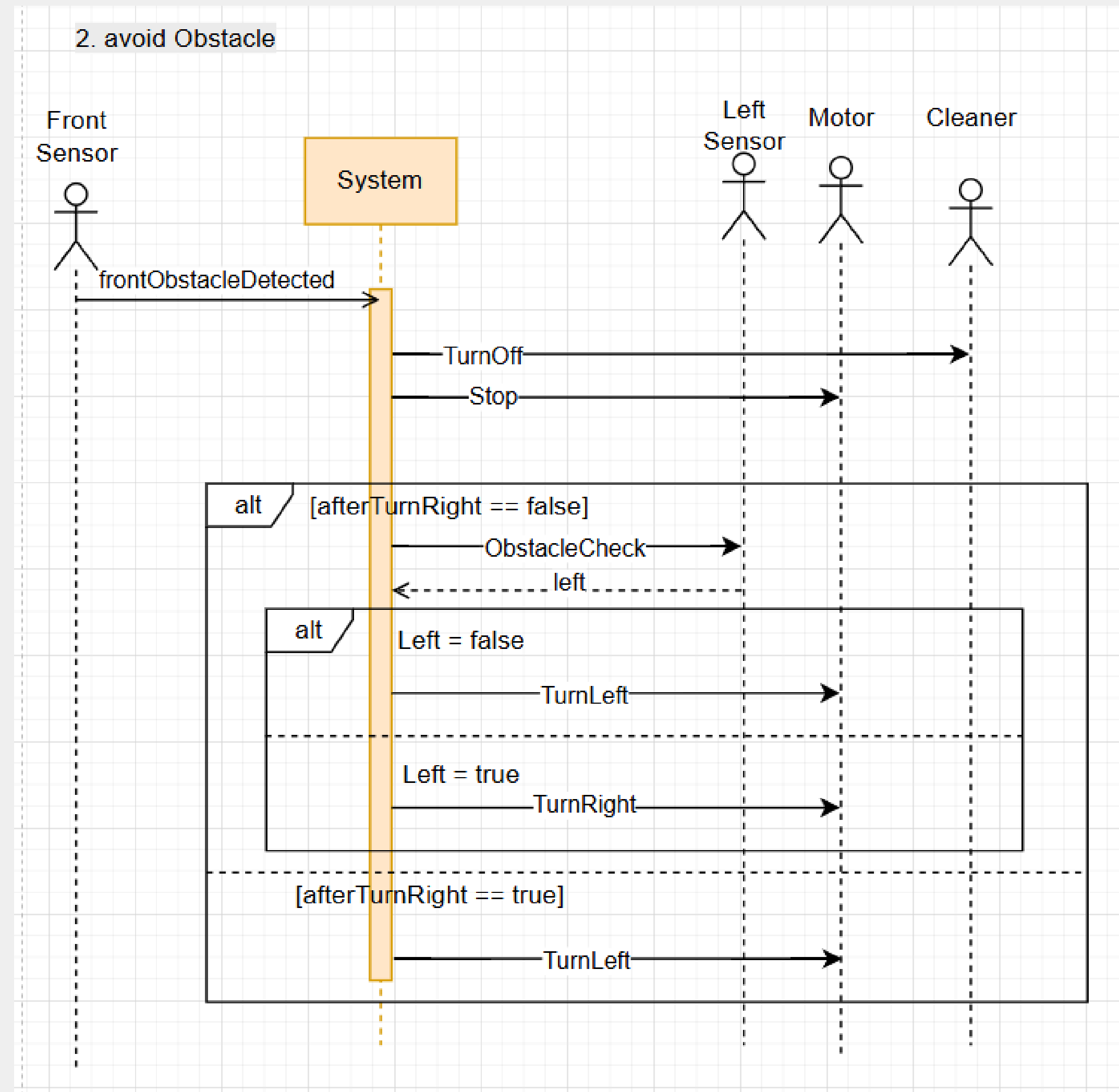
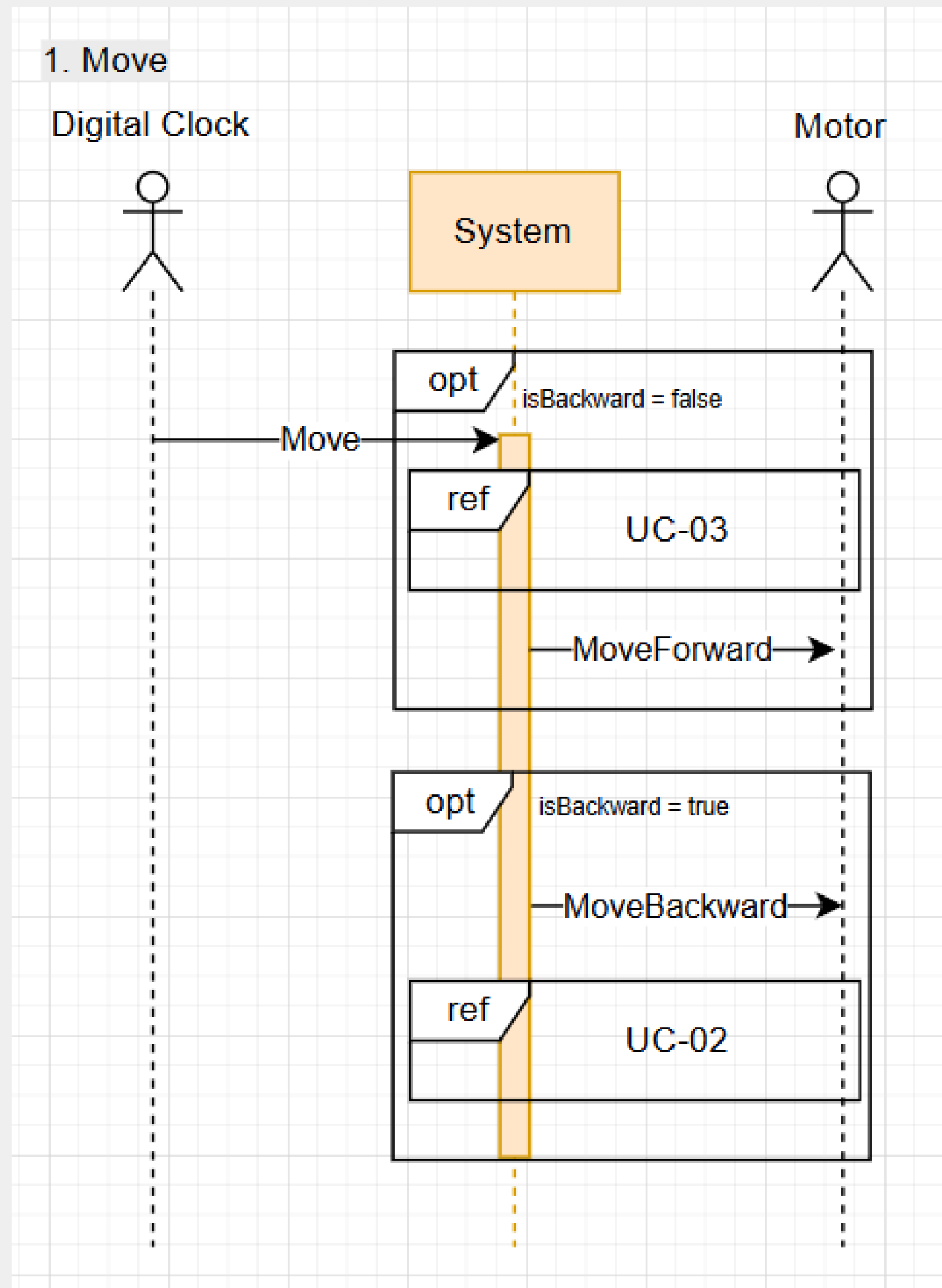
System Sequence Diagram

Before



System Sequence Diagram

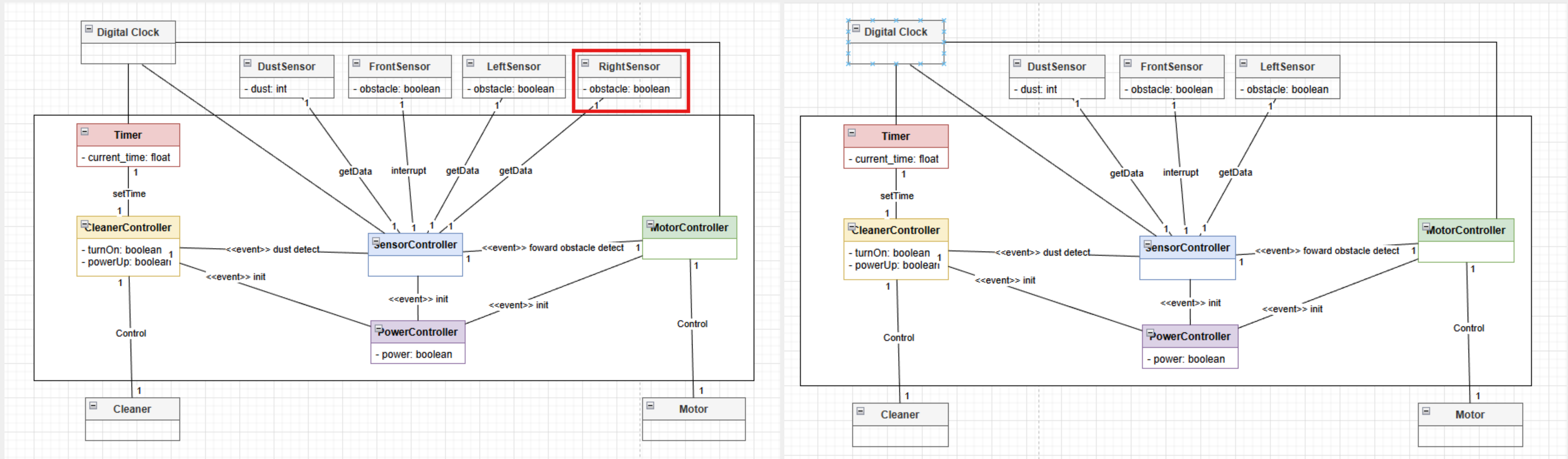
After



Domain Diagram

Before

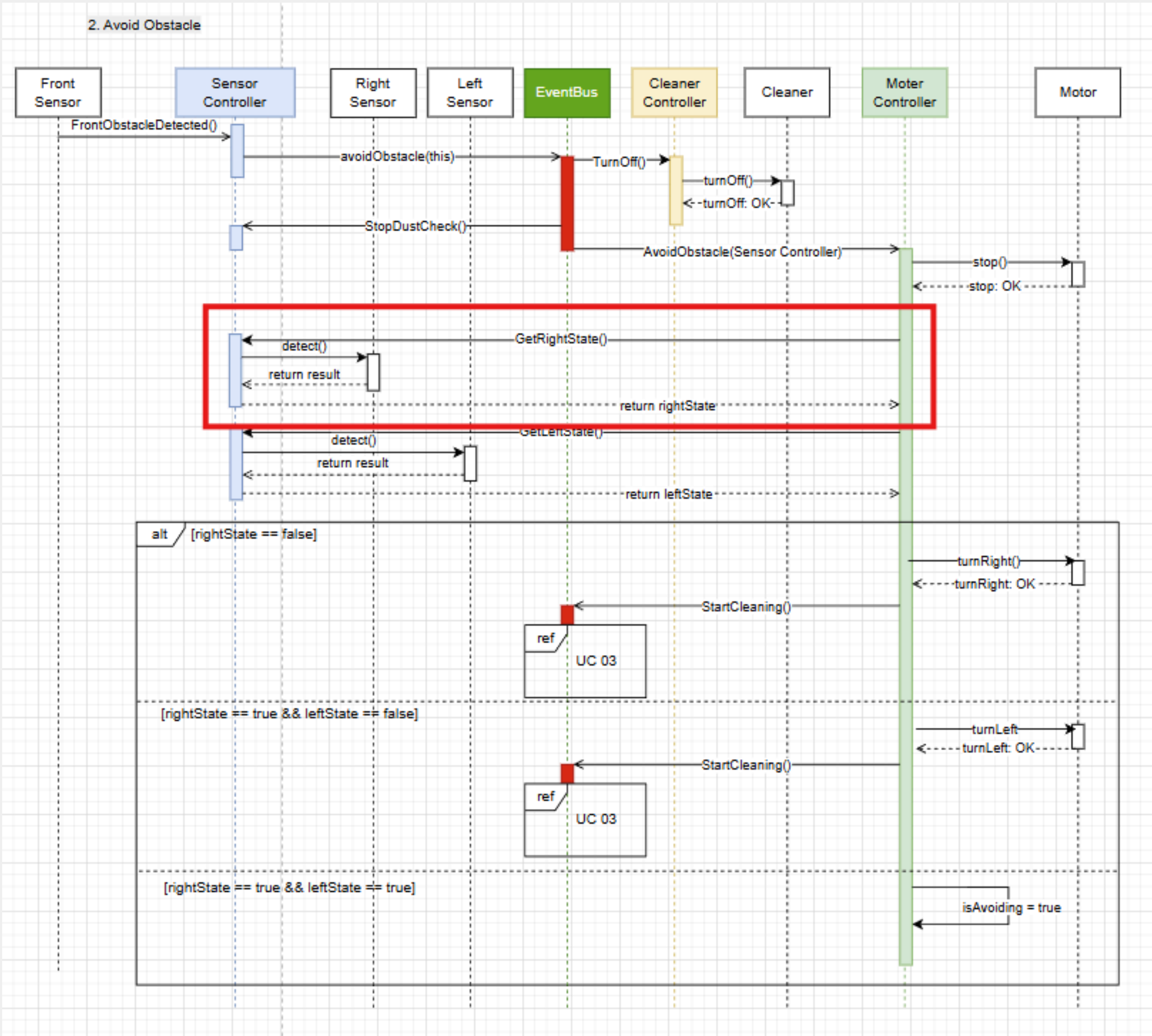
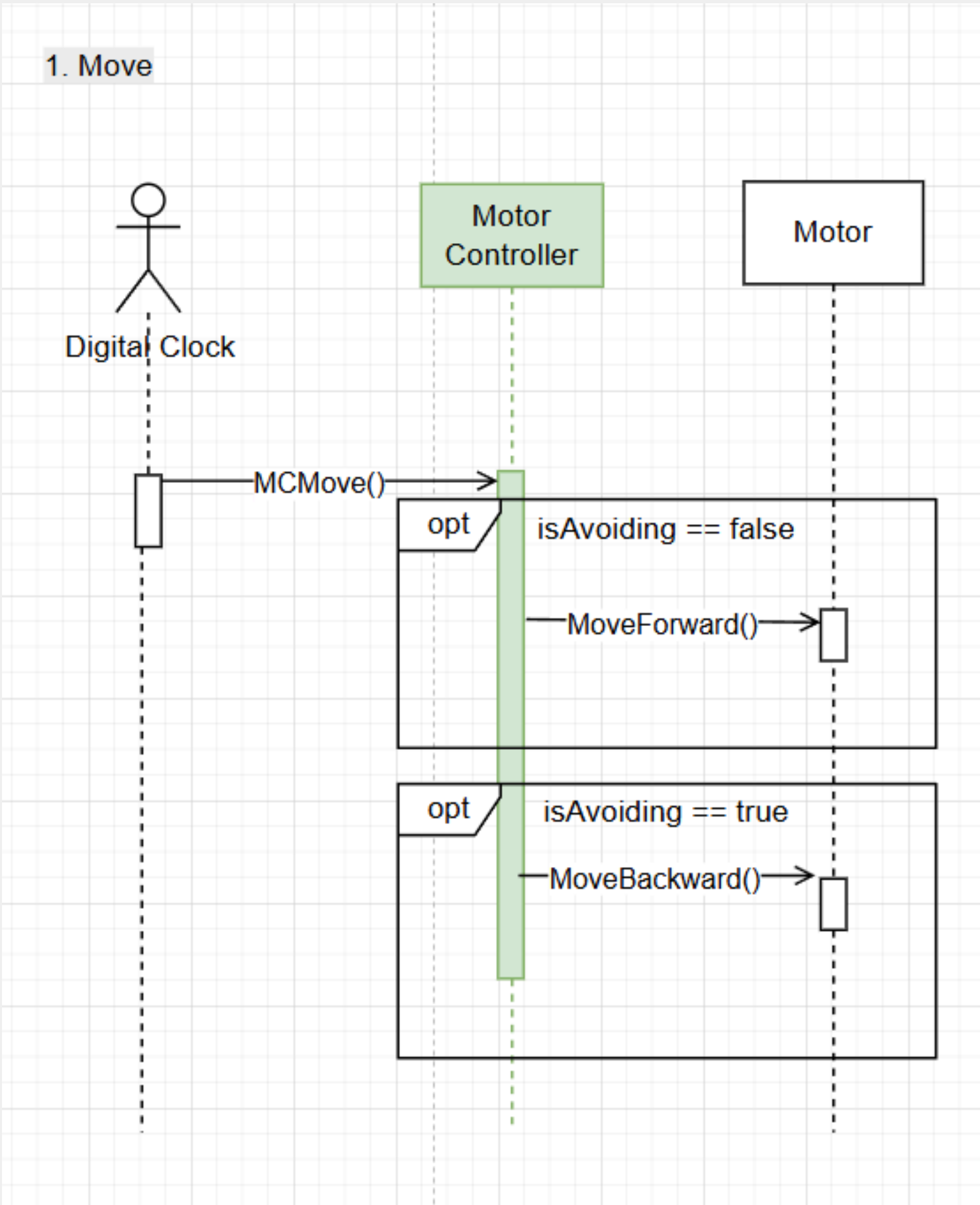
After



Right Sensor 삭제

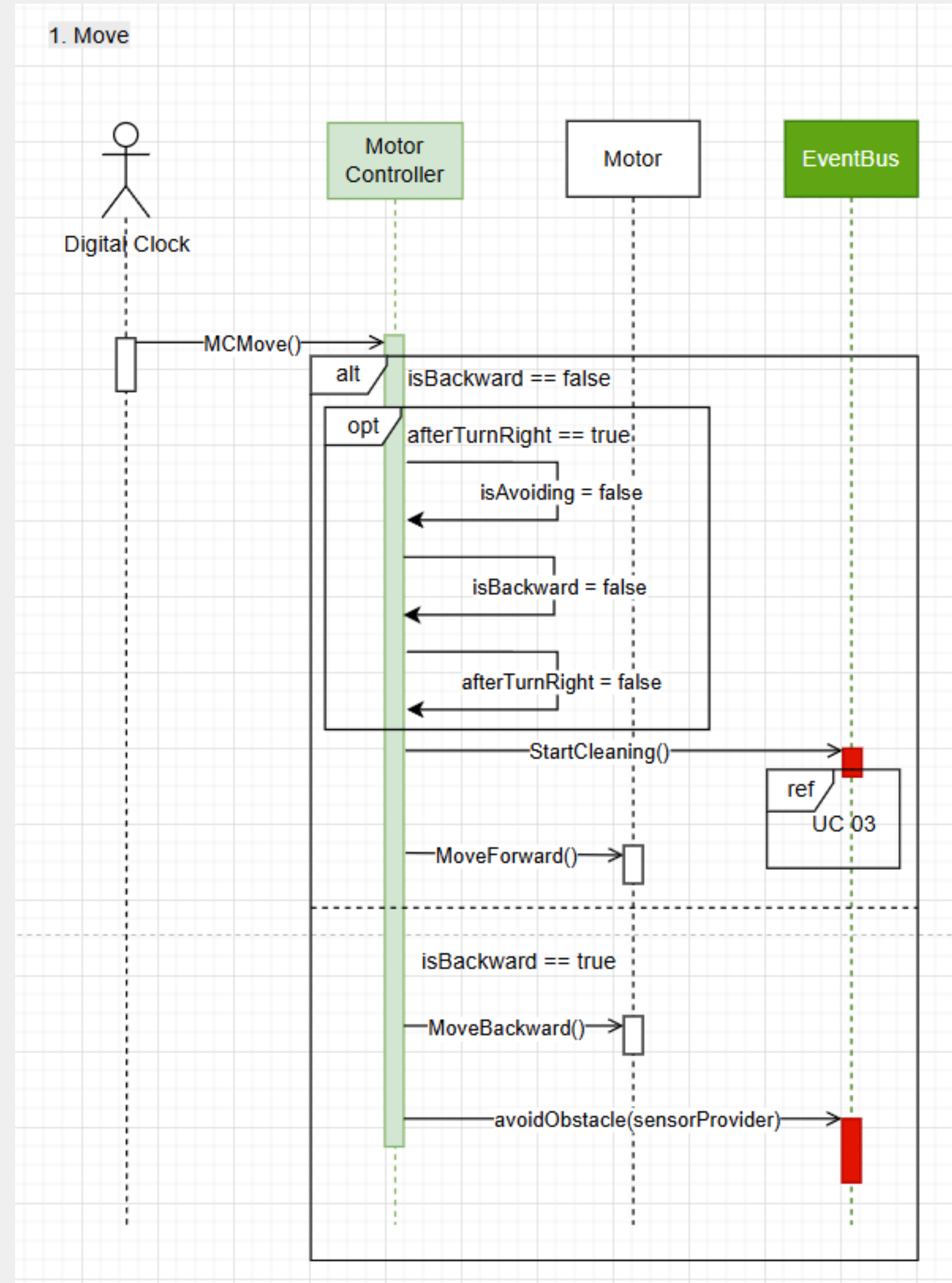
Sequence Diagram

Before



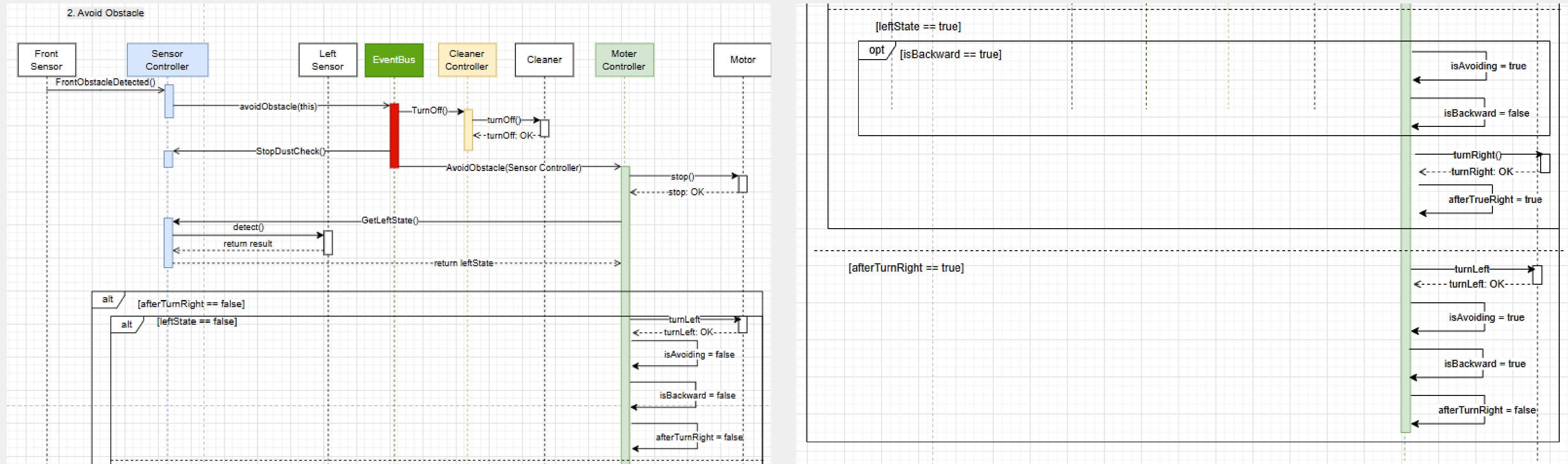
Sequence Diagram

After



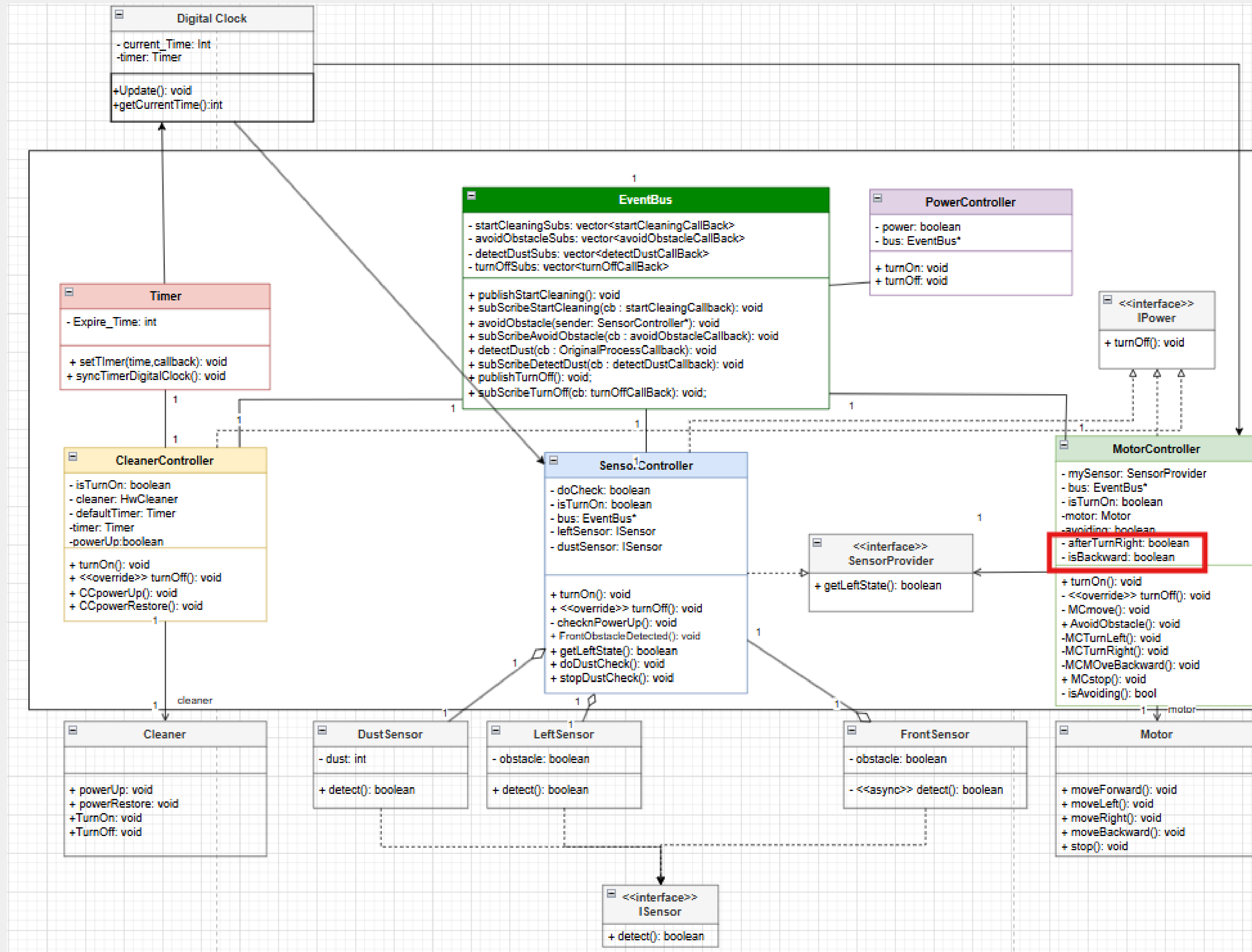
Sequence Diagram

After



Class Diagram

After



Code

Before

```
void MotorController::AvoidObstacle(SensorProvider& provider) {
    mySensor = &provider;
    if(isTurnOn == false) {
        return;
    }

    if(provider.getRightState() == false) {
        MCTurnRight();
        avoiding = false;
        bus->publishStartCleaning();
    }else if(provider.getLeftState() == false){
        MCTurnLeft();
        avoiding = false;
        bus->publishStartCleaning();
    }else{
        avoiding = true;
        // No exit yet: next tick should be another backward tick
    }
}

void MotorController::MCMove() {
    if(!avoiding) {
        motor.moveForward();
        return;
    }else{
        if (mySensor != nullptr) {
            motor.moveBackward();
            bus->publishAvoidObstacle(mySensor);
        }
        return;
    }
}
```

After

```
void MotorController::AvoidObstacle(SensorProvider& provider) {
    mySensor = &provider;
    if(isTurnOn == false) {
        return;
    }

    if (afterTurnRight == true) {
        MCTurnLeft();
        avoiding = true;
        isBackward = true;
        afterTurnRight = false;
    }else if (provider.getLeftState() == false) {
        MCTurnLeft();
        avoiding = false;
        isBackward = false;
        afterTurnRight = false;
    }else {
        if (isBackward == true) {
            avoiding = true;
            isBackward = false;
        }
        MCTurnRight();
        afterTurnRight = true;
    }
}

void MotorController::MCMove() {
    if(isTurnOn == false) {
        return;
    }

    if(isBackward == false) {
        if(afterTurnRight == true) {
            avoiding = false;
            isBackward = false;
            afterTurnRight = false;
        }
        bus->publishStartCleaning();
        motor.moveForward();
    }else{
        MCMoveBackward();
        if (mySensor != nullptr) {
            bus->publishAvoidObstacle(mySensor);
        }
    }
}
```

Unit Test

Before

테스트	기간	특성	오류 메시지
unit_tests (23)	474ms		
<빈 네임스페이스> (15)	474ms		
CleanerTest (4)	< 1ms		
CleanerControllerBusSubscribe...	< 1ms	GoogleTes...	
CleanerHwConnectCheck	< 1ms	GoogleTes...	
TurnOnTurnOffDoesNotThrow	< 1ms	GoogleTes...	
WhenPowerUpTimerCallingCh...	< 1ms	GoogleTes...	
RvcSimulatorTest (3)	474ms		
AutoStepRunsRequestedTicks	474ms	GoogleTes...	
DustTriggersPowerUpAndTime...	< 1ms	GoogleTes...	
StepMovesForwardWhenFrontl...	< 1ms	GoogleTes...	
SensorControllerTest (8)	< 1ms		
TestChecknPowerUp_WhenNo...	< 1ms	GoogleTes...	
TestFrontObstacleDetectedTrue	< 1ms	GoogleTes...	
TestGetLeftStateFalse	< 1ms	GoogleTes...	
TestGetLeftStateTrue	< 1ms	GoogleTes...	
TestGetRightStateFalse	< 1ms	GoogleTes...	
TestGetRightStateTrue	< 1ms	GoogleTes...	
TestTurnOff	< 1ms	GoogleTes...	
TestTurnOn	< 1ms	GoogleTes...	
MotorControllerTest (8)	< 1ms		
MotorControllerAvoidTest (6)	< 1ms		
BothKeep	< 1ms	GoogleTes...	
FrontObstacle	< 1ms	GoogleTes...	
LeftAndRightObstacle	< 1ms	GoogleTes...	
LeftObstacle	< 1ms	GoogleTes...	
RightObstacle	< 1ms	GoogleTes...	
SeqLeftOpen	< 1ms	GoogleTes...	

After

unit_tests (24)			
<빈 네임스페이스> (16)			
CleanerTest (4)			
CleanerControllerBusSubscribeCheck			
CleanerHwConnectCheck			
TurnOnTurnOffDoesNotThrow			
WhenPowerUpTimerCallingCheck			
RvcSimulatorTest (4)			
AutoStepRunsRequestedTicks			
DustTriggersPowerUpAndTimerRestoresPower			
StepMovesForwardWhenFrontIsEmpty			
TurnOffStopsCleaner			
SensorControllerTest (8)			
TestAvoidObstacleStopsDustCheckWithoutRightSensor			
TestChecknPowerUp_WhenNoDust			
TestFrontObstacleDetectedProvidesLeftStateOnly			
TestFrontObstacleDetectedTrue			
TestGetLeftStateFalse			
TestGetLeftStateTrue			
TestTurnOff			
TestTurnOn			
MotorControllerTest (8)			
MotorControllerAvoidTest (6)			
BackwardWithLeftStillBlockedChecksRightAgain			
FrontObstacle			
LeftBlockedTurnsRightWithoutRightSensor			
LeftOpenEscapesWithoutRightSensor			
RightCheckBlockedStartsBackwardWithoutRightSensor			
SeqLeftOpen			
MotorControllerPowerTest (2)			
TurnOff			
TurnOn			

- Right Sensor 삭제로 인해 Sensor, Motor 테스트 강화
 - 특히 Sensor는 좌측 센서의 중요성이 더 커졌으므로 좌측 센서에 대한 테스트를 더 많이 함
- Motor는 원래 우회전이었지만, 로직이 좌회전으로 바뀌면서 많은 부분 수정

Test Coverage

Coverage on New Code 89.8%

New code: since previous version

	Coverage on New Code	Uncovered Lines on New Code	Uncovered Conditions on New Code
OOAD/Class/IPower.cpp_1_2026	0.0%	1	0
OOAD/Class/Timer.cpp_1_2026	86.4%	2	1
OOAD/Class/CleanerController.cpp_1_2026	87.8%	0	6
OOAD/Class/MotorController.cpp_1_2026	88.3%	4	7
OOAD/Class/EventBus.cpp_1_2026	90.0%	0	4
OOAD/Class/SensorController.cpp_1_2026	95.6%	0	2

- 이전에 안 했던 테스트 커버리지 수행
- 전반적으로 높은 테스트 커버리지를 보여줌

System Test

Before

```
#####
# #
# #
# # #
# #^# #
# #
#####
전원 : OFF | 청소기 : OFF | 파워업 : OFF
위치 : (3, 2) | 방향 : 위 (0, 1)
센서 - 전방벽 : 1, 좌측벽 : 1, 우측벽 : 1, 현재면지 : 0
범례 : # 벽, . 빈칸, * 먼지, x 청소완료, ^v<> RVC

#####
# #
# #
# # #
# # #
# > #
#####
전원 : ON | 청소기 : ON | 파워업 : OFF
위치 : (3, 1) | 방향 : 오른쪽 (1, 0)
센서 - 전방벽 : 0, 좌측벽 : 0, 우측벽 : 1, 현재면지 : 0
범례 : # 벽, . 빈칸, * 먼지, x 청소완료, ^v<> RVC

#####
# #
# #
# # #
# # #
# > #
#####
전원 : ON | 청소기 : ON | 파워업 : OFF
위치 : (4, 1) | 방향 : 오른쪽 (1, 0)
센서 - 전방벽 : 0, 좌측벽 : 1, 우측벽 : 1, 현재면지 : 0
범례 : # 벽, . 빈칸, * 먼지, x 청소완료, ^v<> RVC
---- Case 7 End ----
[Case 7][Positive] PASS - 후진 중 우측 길 발견 시 회전
Total: 1, Pass: 1, Fail: 0
test> |
```

After

```
---- Case 8 Start ----
[Positive] 우회전 후 전방 막힘 시 후진 진입

#####
# #
# #
# # #
# #^# #
# #
#####
전원 : OFF | 청소기 : OFF | 파워업 : OFF
위치 : (3, 2) | 방향 : 위 (0, 1)
센서 - 전방벽 : 1, 좌측벽 : 1, 우측벽 : 1, 현재면지 : 0
범례 : # 벽, . 빈칸, * 먼지, x 청소완료, ^v<> RVC

#####
# #
# #
# # #
# # #
# < #
#####
전원 : ON | 청소기 : OFF | 파워업 : OFF
위치 : (3, 1) | 방향 : 왼쪽 (-1, 0)
센서 - 전방벽 : 0, 좌측벽 : 1, 우측벽 : 모르는 상태, 현재면지 : 0
범례 : # 벽, . 빈칸, * 먼지, x 청소완료, ^v<> RVC

#####
# #
# #
# # #
# # #
# < #
#####
전원 : ON | 청소기 : ON | 파워업 : OFF
위치 : (3, 2) | 방향 : 위 (0, 1)
센서 - 전방벽 : 1, 좌측벽 : 1, 우측벽 : 1, 현재면지 : 0
범례 : # 벽, . 빈칸, * 먼지, x 청소완료, ^v<> RVC

#####
# #
# #
# # #
# # #
# < #
#####
전원 : ON | 청소기 : ON | 파워업 : OFF
위치 : (2, 1) | 방향 : 왼쪽 (-1, 0)
센서 - 전방벽 : 0, 좌측벽 : 1, 우측벽 : 모르는 상태, 현재면지 : 0
범례 : # 벽, . 빈칸, * 먼지, x 청소완료, ^v<> RVC

#####
# #
# #
# # #
# # #
# < #
#####
전원 : ON | 청소기 : ON | 파워업 : OFF
위치 : (3, 2) | 방향 : 위 (0, 1)
센서 - 전방벽 : 1, 좌측벽 : 1, 우측벽 : 1, 현재면지 : 0
범례 : # 벽, . 빈칸, * 먼지, x 청소완료, ^v<> RVC

#####
# #
# #
# # #
# # #
# < #
#####
전원 : ON | 청소기 : ON | 파워업 : OFF
위치 : (1, 1) | 방향 : 왼쪽 (-1, 0)
센서 - 전방벽 : 1, 좌측벽 : 1, 우측벽 : 모르는 상태, 현재면지 : 0
범례 : # 벽, . 빈칸, * 먼지, x 청소완료, ^v<> RVC
```

Vibe Coding

최초 프롬프트

```
▼ docs  
| rvc.pdf.pdf
```

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

발견한 문제점

문제 정의:

현재 구현은 `GridSimulator` 가 `RvcController` 를 직접 소유하고 센서 생성, 명령 적용, 실행 루프를 모두 담당한다. 이 때문에 설계상 주 시스템인 RVC보다 테스트용 시뮬레이터가 중심 객체처럼 보인다. 수업의 의도는 RVC를 메인 시스템으로 설계하고 시뮬레이터는 실제 하드웨어를 대체하는 테스트 환경으로 두는 것이므로, `Rvc` 상위 객체와 하드웨어 어댑터 추상화를 추가해 책임 방향을 정리해야 한다.

리팩토링

RVC 구조 문서 변경 ...

현재 GridSimulator가 RvcController를 직접 소유해서 시뮬레이터가 중다. 수업 의도에 맞게 RVC를 메인 시스템으로 드러내고 싶습니다. Rvc는 RvcController와 RvcHardwareAdapter를 소유하게 하고, GridSimulator는 SimulatedHardwareAdapter를 통해 테스트용 하드웨어 환경만 제공해주세요

문서는 [추가] [삭제] [변경] 등의 태그를 달아 수정사항을 쉽게 볼수있게는 태그를 달고 취소선 표시를 해주세요

계획 모드라 실제 파일 수정은 아직 하지 않고, 문서 구조와 현재 RVC/시뮬레이터 설명이 어디에 있는지 그 방식까지 포함해서 바로 실행 가능한 문서 수정 계획으로 정리할게요.

검색 결과상 핵심 충돌은 docs/srs.md, docs/sdd.md, docs/ood_class_diagram.md 에 모여 있는 다이어그램이 GridSimulator *-- RvcController 로 되어 있어 사용자가 말한 "시뮬레이터가 중심"으로 드러납니다.

문서 내용이 일부 깨져 보이는데, 파일 자체보다는 PowerShell 출력 인코딩 문제일 가능성이 큼니다. 기존 한글 문장 보존을 명시하고, 수정 시에는 필요한 줄만 조심적으로 바꾸는 방향으로 잡겠습니다.

이제 변경된 문서로 코드 리팩토링 해줘

5m 27s 동안 작업 >

리팩토링 완료했습니다. 문서 구조에 맞춰 Rvc 상위 객체를 추가하고, RvcHardwareAdapter / SimulatedHardwareAdapter 를 분리했습니다. 이제 GridSimulator.hpp 는 RvcController 를 직접 소유하지 않고, SimulatedHardwareAdapter 를 만든 뒤 Rvc 를 통해 tick을 실행합니다.

주요 변경:

- Rvc.hpp, Rvc.cpp 추가
- RvcHardwareAdapter.hpp 추가
- SimulatedHardwareAdapter.hpp 추가
- GridSimulator.cpp
- rvc_tests.cpp 로 RvcController 테스트
- CMakeLists.txt 갱신

이제 rvc가 하드웨어 어댑터 까지 가지고 시뮬은 하드웨어랑 환경만주는거 맞지

검증:

- Rvc 가 RvcController 와 RvcHardwareAdapter 를 소유합니다.
- Rvc::tick() 이 adapter에서 센서/interrupt를 읽고, controller가 만든 Command 를 다시 adapter에 적용합니다.
- GridSimulator 는 RvcController 를 직접 갖지 않습니다.
- GridSimulator 는 SimulatedHardwareAdapter 를 만들고, 그 adapter를 Rvc 에 넘겨서 테스트용 하드웨어/격자 환경만 제공합니다.

즉 수업 의도대로 메인 시스템은 Rvc, 시뮬레이터는 검증용 환경 제공자 역할입니다.

요구사항 분석

요구사항이 생겼어 우측센서를 제거해야함
우선 문서 부터 수정할건데 우측센서가 없어지면서 바뀌는 정책이 뭘까?

1m 37s 동안 작업 >

핵심은 우측은 더 이상 "열림"으로 판단할 수 없는 방향이 된다는 점이에요. 그래서 문서 정책은 **우측 미관측 = 안전하게 선택 할 수 없음** 으로 잡는 게 맞아 보입니다.

바뀌는 정책은 이렇게 정리하면 됩니다

영역	기존 정책	우측 센서 제거 후 정책
Periodic sensor	좌측, 우측, 먼지 샘플링	좌측, 먼지만 샘플링
회피 판단	좌/우 중 열린 방향으로 회전	좌측이 열려 있으면 TurnLeft, 좌측이 막혀 있으면 검증된 측면 출구가 없으므로 Backward
우회전 판단	우측만 열리면 TurnRight	우측 개방 여부를 알 수 없으므로 센서 기반 TurnRight 정책 제거
양쪽 열림	좌/우 번갈아 회전	양쪽 열림을 판정할 수 없으므로 교대 회전 정책 제거
Escaping 진입	전방, 좌측, 우측 모두 막힘	전방 막힘 + 좌측 막힘, 즉 확인 가능한 측면 출구 없음
Escaping 유지	좌우가 계속 막혀 있으면 후진	좌측이 계속 막혀 있으면 후진
Escaping 종료	좌측 또는 우측 중 하나가 열리면 탈출 가능	좌측이 열릴 때만 탈출 가능
탈출 후 회전	열린 측면 방향으로 회전	좌측이 열리면 TurnLeft

요구사항 분석

아니야 우측센서 역할은 rvc가 우측으로 90도 돌아서 전방센서로 체크할거야 이거 기준으로 다시 정리

56s 동안 작업 >

맞아, 이 기준이면 정책은 **“우측을 모르는 방향으로 버리는 것”**이 아니라 **“우측 판단을 능동 탐색으로 바꾸는 것”**이 핵심이야. 정리하면 이렇게 바뀌어야 해.

영역	기존 정책	변경 정책
센서 구성	전방 interrupt, 좌측/우측/먼지 periodic	전방 interrupt, 좌측/먼지 periodic
우측 장애물 판단	<code>rightObstacle</code> 값으로 즉시 판단	RVC가 <code>TurnRight</code> 로 90도 회전한 뒤, 전방 센서 interrupt로 판단
우측 상태	tick 입력에 항상 포함	<code>Unknown</code> 상태이며, 우측 확인 동작 후에만 <code>Open/Blocked</code> 판단
우회전 의미	열린 우측으로 회피	우측 방향을 향하도록 회전하고, 동시에 우측 탐색 절차에 진입
회피 정책	좌/우 센서로 열린 방향 선택	좌측이 열려 있으면 좌회전, 좌측이 막히면 우측 탐색
Escaping 조건	전방, 좌측, 우측 모두 막힘	전방 막힘 + 좌측 막힘 + 우측 탐색 결과 막힘
Escaping 유지	좌/우 중 하나가 열릴 때까지 후진	좌측이 열리거나, 우측 탐색으로 열린 방향이 확인될 때까지 후진/탐색
교대 회전	좌우 모두 열리면 좌/우 번갈아 선택	즉시 판단 불가. 유지하려면 “우측 선호 차례에는 우측 탐색” 절차가 필요



요구사항 반영

플랜

다운로드 공유 댓글

우측 센서 제거 및 우측 탐색 문서 반영 계획

요약

- 기존 [변경], [삭제], [추가] 와 구분하기 위해 이번 요구사항 변경분은 [R2-변경], [R2-삭제], [R2-추가] 태그를 사용한다.
- 우측 센서는 삭제하고, 우측 장애물 판단은 TurnRight -> Front Sensor 확인 절차로 대체한다.
- 문서 범위는 핵심 명세, OOA/OOD, 추적성, 시나리오 카탈로그, 테스트 결과 영향 표시까지 반영한다.

핵심 정책

- 좌측 센서와 먼지 센서만 periodic 입력으로 유지한다. RightSensor, rightObstacle, rightPeriodic 는 [R2-삭제] 처리한다.
- 전방 장애물 발생 시 좌측이 열려 있으면 즉시 TurnLeft 한다. 좌측이 막혀 있으면 TurnRight 로 우측 탐색을 시작한다.
- 우측 탐색 후 전방 interrupt가 없으면 기존 우측 방향이 열린 것으로 보고 전진 청소를 재개한다.
- 우측 탐색 후 전방 interrupt가 있으면 TurnLeft 로 원래 진행 방향을 복귀한 뒤 Backward 후진 탈출에 진입한다.
- Escaping 중에는 Backward 1회 후 원래 방향 기준 좌측이 막혀 있으면 우측 탐색을 수행한다. 흐름은 Backward -> TurnRight -> 전방센서 확인 -> TurnLeft 또는 Forward 이다.

문서 변경

- SRS/requirements: FR-06은 좌측/먼지 periodic sampling으로 변경하고, FR-08은 “우측 센서값 기반 우회전”에서 “좌측 막힘 시 우측 탐색을 위한 우회전”으로 재정의한다. FR-09의 좌우 교대 정책은 삭제한다.
- SDD/OOD: PeriodicSensorData 와 SensorsSnapshot 에서 rightobstacle 를 제거하고, 우측 탐색 상태와 원래 방향 복구 시퀀스를 상태/알고리즘 뷰에 추가한다.
- OOA/SSD: RightSensor actor를 삭제하고, 우측 탐색은 Motor TurnRight 후 FrontSensor 가 기존 우측 방향을 확인하는 상호작용으로 표현한다.
- 로그/검증 문서: rightPeriodic 필드는 삭제하고 필드 rightProbe=none|checking|open|blocked 형식으로 우측 탐색 결과를 기록하도록 문서화한다.

플랜

다운로드 공유 댓글

R2 우측 탐색 리팩토링 계획

요약

- 변경된 최종 문서 기준으로 우측 periodic 센서를 제거하고, TurnRight 후 전방 interrupt로 기존 우측 방향을 확인하는 흐름으로 컨트롤러를 리팩토링한다.
- 현재 기준선 테스트는 ctest --test-dir build -C Debug --output-on-failure 에서 59개 모두 통과했지만, 구 정책 기준이므로 R2 테스트로 교체한다.

핵심 변경

- Types.hpp 에 RightProbeState { None, Checking, Open, Blocked } 를 추가하고, ControllerState 에 RightProbing, EscapeAligning 을 추가한다.
- PeriodicSensorData 에서 rightObstacle 을 제거하고, SensorSnapshot 은 frontObstacle, leftObstacle, dustDetected, rightProbe 만 갖게 한다.
- RvcController.cpp 의 좌우 센서 기반 chooseOpenSideTurn / preferLeftTurn_ 정책을 제거한다.
- 새 상태 전이는 다음으로 고정한다:
 - Cleaning + front blocked + left open: Avoiding, TurnLeft
 - Cleaning + front blocked + left blocked: RightProbing, TurnRight
 - RightProbing + front clear: Cleaning, rightProbe=Open, Forward
 - RightProbing + front blocked: EscapeAligning, rightProbe=Blocked, TurnLeft
 - EscapeAligning : Escaping, Backward
 - Escaping + left open: Avoiding, TurnLeft
 - Escaping + left blocked: RightProbing, TurnRight
- makeCommand 의 cleaner 출력 규칙은 유지한다. Forward 에서만 Normal/Boost, 그 외 motion은 항상 off.

시뮬레이터와 로그



- SimulatedHardwareAdapter.cpp 는 좌측 obstacle과 dust만 periodic으로 반환하고, 우측 obstacle 계산은 제거

변경사항 분석

R2 변경중에서 꼭 발표해야할 중요 변화 어느파일에 어느 부분임?

1m 58s 동안 작업 >

꼭 발표할 R2 핵심은 우측 센서 제거 → 우회전 후 전방 센서로 우측 탐색입니다. 파일 기준으로는 아래만 잡으면 충분해요.

1순위: 요구사항 변경

- docs/requirements.md (line 10): 좌측/먼지 센서만 periodic, 우측 센서 periodic 삭제
- docs/requirements.md (line 63): UC-03 흐름 변경, 좌측 막히면 TurnRight 로 우측 탐색
- docs/requirements.md (line 104): FR-06, FR-08~FR-13 변경 및 FR-09 좌우 교대 정책 삭제
- docs/srs.md (line 513): 같은 내용을 정식 SRS 요구사항 표로 정리한 부분

2순위: 설계/타입 변경

- include/rvc/Types.hpp (line 29): ControllerState 에 RightProbing, EscapeAligning 추가
- include/rvc/Types.hpp (line 38): RightProbeState 추가
- include/rvc/Types.hpp (line 52): PeriodicSensorData 에서 rightObstacle 제거, SensorSnapshot 에 rightProbe 반영
- docs/sdd.md (line 600): 설계 문서의 상태 정의와 알고리즘 설명

3순위: 실제 제어 로직

- src/RvcController.cpp (line 41): SensorSnapshot 이 전방 interrupt, 좌측/먼지 periodic, 우측 탐색 상태를 합침
- src/RvcController.cpp (line 57): RightProbing 상태에서 전방이 막히면 EscapeAligning, 열리면 전진 재개
- src/RvcController.cpp (line 69): 우측 탐색 실패 후 TurnLeft 로 원래 방향 복구하고 Backward
- src/RvcController.cpp (line 75): Escaping 중 좌측 막히면 다시 우측 탐색 반복
- src/RvcController.cpp (line 93): 전방 interrupt 시 좌측 open이면 TurnLeft, 좌측 blocked면 TurnRight

4순위: 검증 포인트

Changed

1. 핵심 변경 요약

구분	변경 전	변경 후
우측 장애물 판단	Right Sensor 가 우측 장애물을 periodic 방식으로 제공	별도 우측 센서 없이 TurnRight 후 Front Sensor interrupt로 기존 우측 방향을 확인
periodic sensor 입력	좌측, 우측, 먼지 센서 값을 tick마다 전달	좌측, 먼지 센서 값만 tick마다 전달
회피 정책	전방 장애물 발생 후 좌/우 센서값으로 열린 방향 선택	좌측이 열려 있으면 TurnLeft, 좌측이 막혀 있으면 TurnRight 로 우측 탐색
좌우 모두 open 정책	좌우가 모두 열려 있으면 좌/우를 번갈아 선택	우측 센서가 제거되어 좌우 동시 open 판단 및 교대 정책 삭제
삼방향 blocked 처리	전방, 좌측, 우측이 모두 막히면 바로 Escaping 에서 후진	전방과 좌측이 막히면 우측 탐색을 먼저 수행하고, 우측도 막히면 원래 방향 복구 후 후진
탈출 조건	좌측 또는 우측 센서 중 하나가 open이면 탈출 가능	좌측이 open이거나 우측 탐색 결과가 open이면 탈출 가능

Changed

2. 요구사항 변경

R2에서 요구사항의 중심이 "우측 센서값을 읽는다"에서 "전방 센서를 우측 탐색에도 재사용한다"로 바뀌었다.

파일	위치	주요 내용
docs/requirements.md	10-13행	좌측/먼지 센서만 periodic으로 남고, 우측 센서 periodic 요구사항은 삭제됨
docs/requirements.md	63-64행	UC-03에서 좌측 blocked 시 TurnRight 로 우측 탐색 후 전방 interrupt로 판단
docs/requirements.md	104-111행	FR-06, FR-08~FR-13이 R2 정책 기준으로 변경되고 FR-09 좌우 교대 정책 삭제
docs/srs.md	513-520행	SRS 요구사항 표에서 같은 변경을 정식 요구사항으로 반영

상세 설명:

R2에서는 하드웨어 조건 변경으로 우측 센서가 제거되었다. 따라서 우측 장애물은 주기 센서값이 아니라, RVC가 우측으로 90도 회전한 뒤 전방 센서 interrupt를 통해 확인하도록 요구사항이 바뀌었다.

Changed

3. 설계 변경

우측 센서가 사라지면서 controller 상태와 판단 입력 구조도 바뀌었다.

파일	위치	변경 전	변경 후
<code>include/rvc/Types.hpp</code>	29-35 행	<code>Idle</code> , <code>Cleaning</code> , <code>Avoiding</code> , <code>Escaping</code> 중심 상태	<code>RightProbing</code> , <code>EscapeAligning</code> 상태 추가
<code>include/rvc/Types.hpp</code>	38-43 행	우측 탐색 결과를 표현하는 enum 없음	<code>RightProbeState</code> 추가: <code>None</code> , <code>Checking</code> , <code>Open</code> , <code>Blocked</code>
<code>include/rvc/Types.hpp</code>	52-61 행	<code>PeriodicSensorData.rightObstacle</code> , <code>SensorSnapshot.rightObstacle</code> 사용	<code>rightObstacle</code> 제거, <code>SensorSnapshot.rightProbe</code> 추가
<code>docs/sdd.md</code>	600- 633행	측면 센서값으로 회피/탈출 판단	우측 탐색, 원래 방향 복구, 후진 후 재탐색 알고리즘 명시

중요 설계 의미:

- `PeriodicSensorData` 는 이제 좌측 장애물과 먼지만 가진다.
- `SensorSnapshot` 은 전방 interrupt, 좌측 periodic 값, 먼지 periodic 값, 우측 탐색 상태를 결합한다.
- `RightProbing` 은 `TurnRight` 후 전방 센서로 기존 우측 방향을 확인하는 상태다.
- `EscapeAligning` 은 우측 탐색 실패 후 `TurnLeft` 로 원래 진행 방향을 복구하는 상태다.

Changed

4. 구현 변경

실제 제어 로직은 `RvcController` 의 상태 머신에서 가장 크게 바뀌었다.

파일	위치	주요 내용
<code>src/RvcController.cpp</code>	41-47행	<code>readPeriodicSensors()</code> 가 전방 interrupt, 좌측/먼지 periodic 값, 우측 탐색 상태를 <code>SensorSnapshot</code> 으로 결합
<code>src/RvcController.cpp</code>	57-67행	<code>RightProbing</code> 상태에서 전방 interrupt가 있으면 <code>blocked</code> , 없으면 <code>open</code> 으로 판단
<code>src/RvcController.cpp</code>	69-72행	우측 탐색 <code>blocked</code> 시 <code>TurnLeft</code> 로 원래 진행 방향을 복구한 뒤 <code>Backward</code>
<code>src/RvcController.cpp</code>	75-84행	<code>Escaping</code> 중 좌측이 계속 막히면 다시 <code>TurnRight</code> 로 우측 탐색 반복
<code>src/RvcController.cpp</code>	93-101행	전방 interrupt 발생 시 좌측 <code>open</code> 이면 <code>TurnLeft</code> , 좌측 <code>blocked</code> 면 <code>TurnRight</code>

변경 전 흐름:

```
Front blocked
-> Left/Right periodic sensor 확인
-> 열린 측면 방향으로 회전
-> 좌우 모두 blocked이면 Escaping에서 계속 Backward
```

변경 후 흐름:

```
Front blocked
-> Left periodic sensor 확인
-> Left open이면 TurnLeft
-> Left blocked이면 TurnRight로 우측 탐색
-> 우측 탐색 후 Front clear이면 Forward
-> 우측 탐색 후 Front blocked이면 TurnLeft로 원래 방향 복구
-> Backward
-> Left blocked이면 다시 TurnRight로 우측 탐색 반복
```

Changed

5. 하드웨어/시뮬레이터 구조 변경

R2 변경은 센서 정책뿐 아니라 RVC와 시뮬레이터 연결 구조에도 반영되었다.

파일	위치	주요 내용
<code>include/rvc/RvcHardwareAdapter.hpp</code>	7-13행	전방 interrupt, periodic sensor 읽기, command 적용을 추상화
<code>include/rvc/Rvc.hpp</code>	10-27행	<code>Rvc</code> 가 <code>RvcController</code> 와 <code>RvcHardwareAdapter</code> 를 소유
<code>src/Rvc.cpp</code>	23-32행	tick마다 interrupt 확인, periodic sensor 읽기, controller 호출, command 적용 순서 수행
<code>src/SimulatedHardwareAdapter.cpp</code>	68-76행	시뮬레이터 periodic sensor도 좌측 장애물과 먼지만 제공

상세 설명:

R2에서는 controller가 실제 하드웨어나 시뮬레이터를 직접 알지 않도록 `RvcHardwareAdapter` 를 두었고, `Rvc` 가 센서 입력 수집과 command 적용 순서를 조율한다. 이 구조 덕분에 우측 센서 제거 같은 하드웨어 변경을 adapter와 상태 모델로 흡수할 수 있다.

Changed

6. 테스트 변경

기존 테스트는 우측 센서값과 좌우 교대 정책을 검증했다. R2 이후에는 우측 탐색 성공/실패와 후진 후 재탐색을 검증한다.

파일	위치	변경 내용
tests/controller_tests.cpp	110-123행	좌측 blocked 시 <code>RightProbing</code> 진입과 <code>TurnRight</code> 확인
tests/controller_tests.cpp	126-148행	우측 탐색 open 시 <code>Forward</code> 로 청소 재개 확인
tests/controller_tests.cpp	151-176행	우측 탐색 blocked 시 원래 방향 복구 후 <code>Backward</code> 확인
tests/controller_tests.cpp	179-206행	<code>Escaping</code> 중 후진 후 우측 재탐색 확인
tests/system_tests.cpp	90-108행	시뮬레이터에서 <code>TurnRight -> TurnLeft -> Backward</code> 흐름 검증
tests/system_tests.cpp	131-150행	후진 중 우측 탐색 반복 후 open 시 전진 재개 검증
docs/traceability.md	12-19행	FR-06, FR-08~FR-13 변경 추적
docs/traceability.md	33-36행	변경된 요구사항과 테스트 연결

Changed

7. 최종 정리

R2에서 가장 중요한 변화는 다음과 같이 정리할 수 있다.

우측 센서를 제거하면서, 우측 장애물 판단을 periodic sensor 입력이 아니라 `TurnRight` 후 전방 센서 interrupt로 확인하는 탐색 절차로 바꾸었다.

이 변화 때문에 요구사항, 타입 구조, controller 상태 머신, 시뮬레이터 adapter, 테스트 케이스가 모두 함께 변경되었다.

설명 순서는 다음이 적절하다.

1. 하드웨어 변경: 우측 센서 제거
2. 요구사항 변경: FR-06, FR-08~FR-13, FR-09 삭제
3. 설계 변경: `RightProbing`, `EscapeAligning`, `RightProbeState`
4. 구현 변경: `RvcController` 상태 전이
5. 검증 변경: 우측 탐색 성공/실패와 후진 후 재탐색 테스트

피드백 반영

시뮬레이터 폴더 새로 들어온거봐바 시뮬레이터 폴더 안의 내용은 절대 건드리지 말고 기존 동작이 잘 동작하도록 RVC의 인터페이스만 고치려면 얼마나 바뀔? 기존시뮬레이터랑 호환안되도 좋음

5m 4s 동안 작업 >

확인해보니 변경량은 중간 정도예요. 기존 RVC 알고리즘은 그대로 살릴 수 있고, 새 Simulator 가 요구하는 Class/* , HDWARE/* 인터페이스 업데이트를 RVC 쪽에 추가하는 작업이 핵심입니다.

플랜



새 Simulator용 RVC 인터페이스 적용 계획

요약

- Simulator/ 내부는 수정하지 않는다.
- 기존 rvc::RvcController 동작은 유지하고, 새 시뮬레이터가 기대하는 외부 인터페이스만 추가한다.
- 기존 시뮬레이터 호환성은 고려하지 않는다.
- 단, 현재 새 Simulator 자체에 달하지 않은 문자열이 있어, 이 상태 그대로는 RVC 인터페이스만으로 전체 빌드 성공이 불가능하다.

변경량

- 새 헤더/소스 약 10~14개 추가 예상:

피드백 반영

컨트롤러가 너무 많은 역할을 가지고 있음 부스트나 방향정하기? 이거 solid위반인거 같으니 수정ㄱㄱ

오전 10:50 

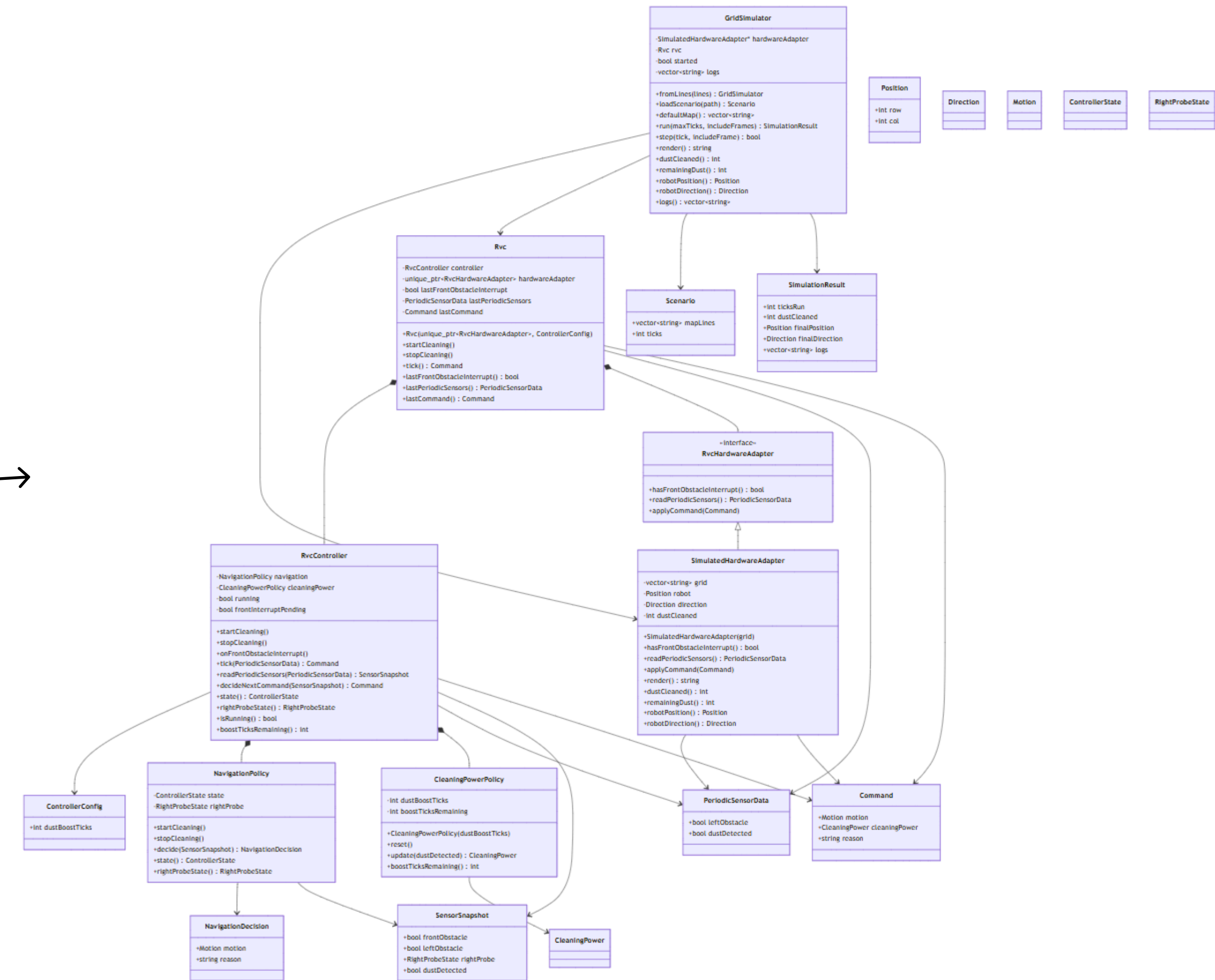
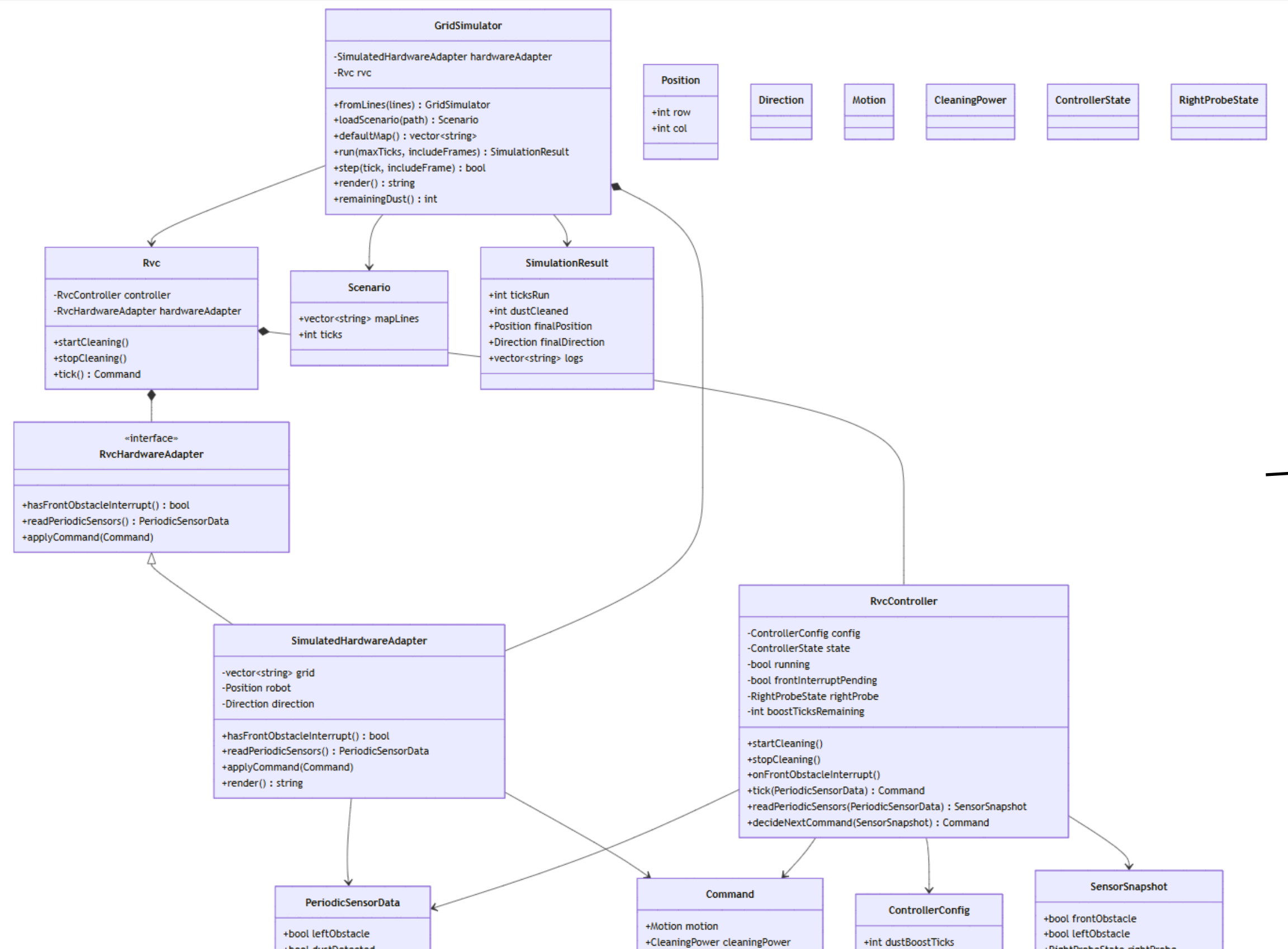
22m 47s 동안 작업 >

수정 완료했습니다. 컨트롤러에서 방향/회피 판단과 먼지 boost 관리를 분리했어요.

핵심 변경:

- `NavigationPolicy` 추가: 방향 결정, 우측 탐색, 회피/탈출 상태 전이 담당
[NavigationPolicy.hpp \(line 14\)](#), [NavigationPolicy.cpp \(line 15\)](#)
- `CleaningPowerPolicy` 추가: dust boost tick 예산과 power 후보 결정 담당
[CleaningPowerPolicy.hpp \(line 7\)](#), [CleaningPowerPolicy.cpp \(line 11\)](#)
- `RvcController` 는 이제 실행 상태, interrupt 소비, policy 결과 조합만 담당
[RvcController.hpp \(line 31\)](#), [RvcController.cpp \(line 62\)](#)
- 분리된 정책 단위 테스트 추가
[policy_tests.cpp \(line 17\)](#)
- OOD/SDD 문서도 새 책임 구조로 동기화했습니다.

class diagram 변화



Coverage

커버리지 분석해서 html 분석지 만들어줘

13m 25s 동안 작업 >

완료했습니다. HTML 커버리지 분석지는 여기 있습니다:

[</> coverage_analysis.html](#)

실제 수집 기준:

- GoogleTest 61 건 실행, 실패 0
- 프로젝트 소스 기준 라인 커버리지: 89.2% (406 / 455)
- 원본 Cobertura 전체 수치 41.1% 는 GoogleTest/표준 라이브러리 코드가 섞인 값이라, HTML에서는 제품 코드만 필터링했습니다.
- 낮은 커버리지 파일과 미커버 라인, 보강 테스트 제안까지 HTML에 포함했습니다.

함께 생성된 주요 산출물은 docs/coverage.merged.cobertura.xml , docs/coverage-test-results.xml , docs/generate_coverage_analysis.ps1 입니다. 참고로 기존에 수정되어 있던 docs/ood_class_diagram.md , docs/sdd.md 는 건드리지 않았습니다.

 generate_coverage_analysis.ps1 수정됨  실행 취소 ↶ 리뷰

RVC 커버리지 분석지

생성 시각: 2026-06-04 12:37:41 +09:00 · 기준 XML: docs/coverage.merged.cobertura.xml · 범위: 프로젝트 소스(src/include/simulator_rvc_interface), 외부 라이브러리 제외

89.2%

프로젝트 라인 커버리지

406 / 455

커버된 라인 / 실행 가능 라인

61

GoogleTest 실행 건수

0

실패 테스트

원본 Cobertura 전체 수치는 GoogleTest 및 표준 라이브러리 코드까지 포함되어 41.1%로 표시됩니다. 이 분석지는 제품 코드만 필터링한 89.2%를 주 지표로 사용합니다.

파일별 요약

파일	커버리지	분포	커버	미커버	미커버 라인
src/main.cpp	66.7%	<div style="width: 66.7%;"></div>	28 / 42	14	9-11, 25-26, 45-47, 63, 79
src/GridSimulator.cpp	83.8%	<div style="width: 83.8%;"></div>	67 / 80	13	26, 107-109, 115-117, 119
src/Rvc.cpp	85.7%	<div style="width: 85.7%;"></div>	24 / 28	4	11, 43-45
src/Types.cpp	88.7%	<div style="width: 88.7%;"></div>	55 / 62	7	16, 30, 48, 66, 78, 96, 110
src/SimulatedHardwareAdapter.cpp	91.6%	<div style="width: 91.6%;"></div>	109 / 119	10	14, 18, 20, 22, 32, 36, 44,
src/RvcController.cpp	98%	<div style="width: 98%;"></div>	49 / 50	1	64
include/rvc/CleaningPowerPolicy.hpp	100%	<div style="width: 100%;"></div>	1 / 1	0	-
include/rvc/GridSimulator					

미커버 라인 상세

src/main.cpp 66.7% · 미커버 9-11, 25-26, 45-47, 63, 79, 83-86

```
7 namespace {
8
9 0 void printUsage(const char* executable) {
10 0     std::cout << "Usage: " << executable << " [--ticks N] [--scenario FILE] [--quiet-map]\n";
11 0 }
12
13 } // namespace
...
23 1     const std::string arg = argv[index];
24 1     if (arg == "--help" || arg == "-h") {
25 0         printUsage(argv[0]);
26 0         return 0;
27 }
```

SUMMARY & TROUBLE SHOOTING

Oracle

FrontSensor를 통해 오른쪽 장애물 여부를 알 수 있도록 방향 결정 로직을 변경함.

기존 EventBus 구조에서 RightSensor가 사용된 부분을 삭제하고, 방향 결정 로직만을 변경하는 것으로 해결함.

Cleaner, Sensor, Motor로 각각 분리를 해두었기 때문에 기존 코드의 여러 함수들 중 실질적으로 변경이 적용된 함수는 AvoidObstacle과 Move로 2개만 변화됨.

CleanerController.cpp	feat: enhance CleanerController to use Timer dependency and update Rv...	last month
CleanerController.h	feat: enhance CleanerController to use Timer dependency and update Rv...	last month
EventBus.cpp	refactor: replace SensorController with SensorProvider in EventBus an...	last month
EventBus.h	refactor: replace SensorController with SensorProvider in EventBus an...	last month
IPower.cpp	test: 예시 테스트 케이스 추가	last month
IPower.h	feat: add tick method to CleanerController and integrate with RvcSimu...	last month
ISensor.h	feat: implement ISensor interface and add sensor classes for dust, fr...	last month
MotorController.cpp	feat: 개편	5 days ago
MotorController.h	feat: 개편	5 days ago
PowerController.cpp	refactor: rename moveForward to startCleaning in EventBus and related...	last month
PowerController.h	chore: move folder	last month
SensorController.cpp	chore: 센서 싹먹으로 인해서 오른쪽 센서 사라짐	last week
SensorController.h	chore: 센서 싹먹으로 인해서 오른쪽 센서 사라짐	last week
SensorProvider.cpp	feat: add tick method to CleanerController and integrate with RvcSimu...	last month
SensorProvider.h	chore: 센서 싹먹으로 인해서 오른쪽 센서 사라짐	last week
Timer.cpp	Tick Operation Edit	last month
Timer.h	Tick Operation Edit	last month

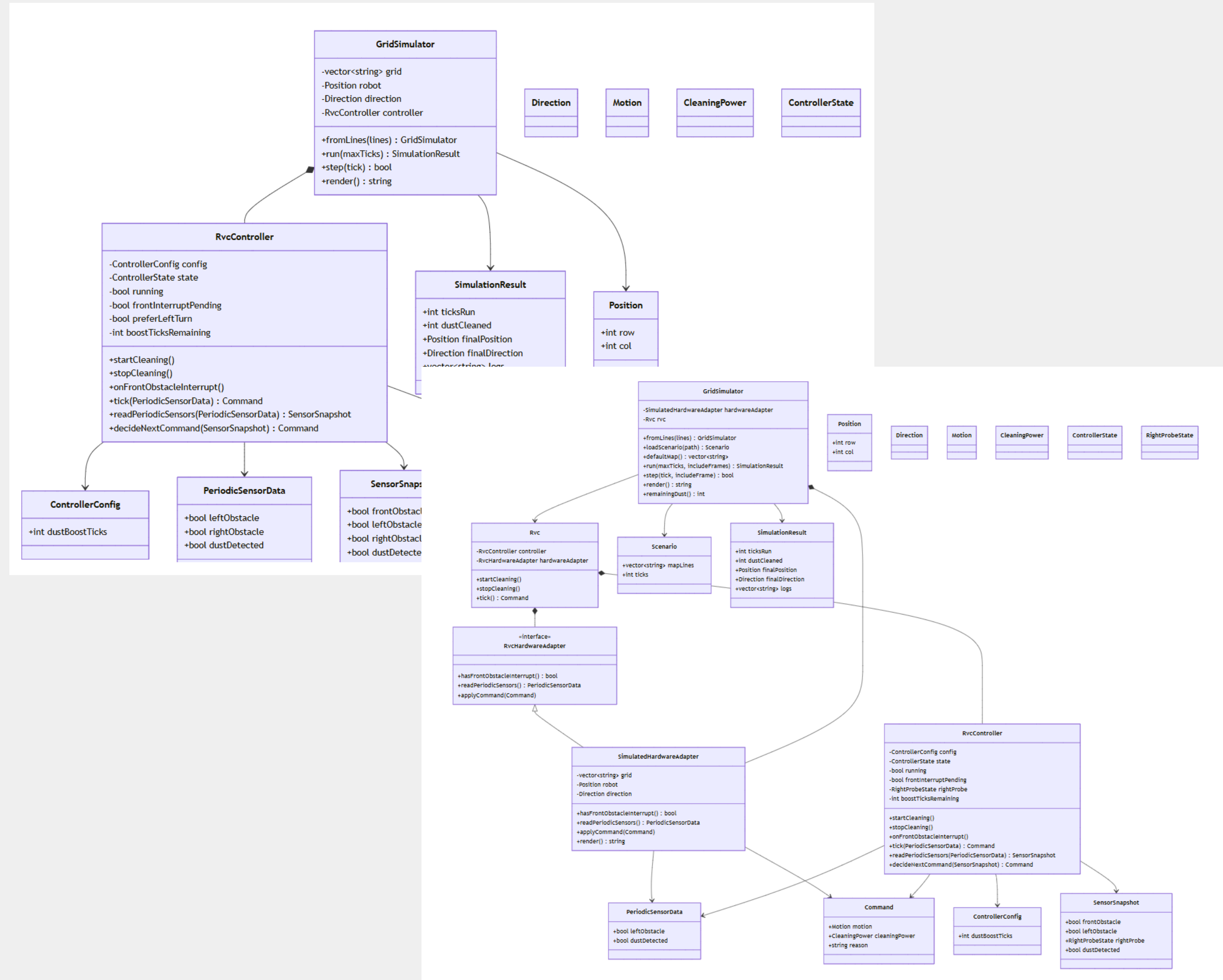
src/GridSimulator.cpp	+22 -188	Viewed
src/Rvc.cpp	+47	Viewed
src/RvcController.cpp	+40 -32	Viewed
src/SimulatedHardwareAdapter.cpp	+201	Viewed
src/Types.cpp	+18	Viewed

Vibe Coding 변화량

Vibe coding

기존 Vibe coding을 진행한 산출물이 Simulator와 RVC간의 결합성이 너무 높아서 RightSensor 제거가 원활하게 수행되지 못했음.

그래서 Simulator와 RVC간 분리를 하는 Refactoring을 먼저 진행한 후에 RightSensor를 제거한 Avoid로직 수정을 진행함.

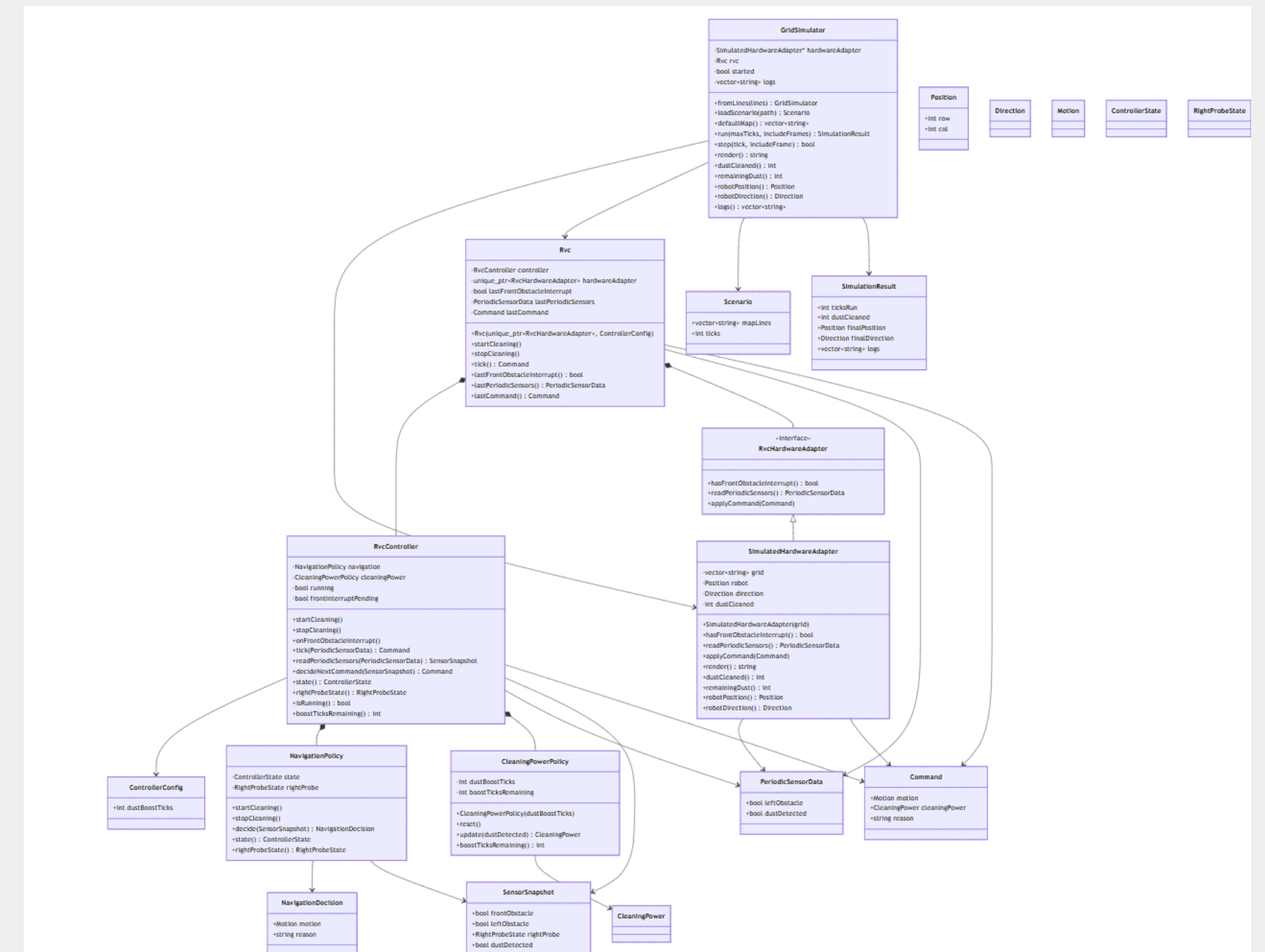
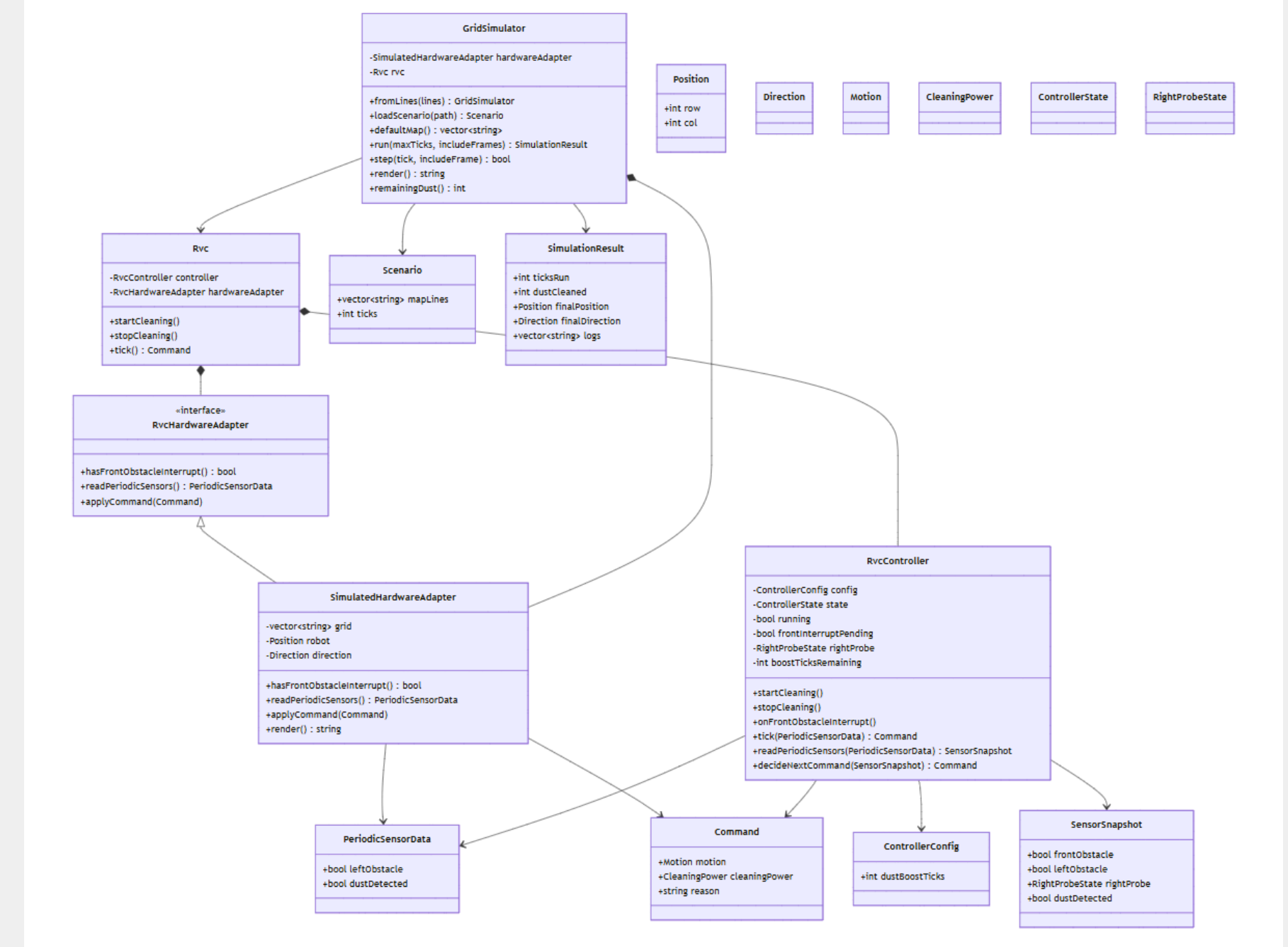


두 코드 내용 비교

Oracle와 Vibe Coding간의 차이점과 수정 사항을 파악하여 더 나은 구조를 모색함

이를 통해 나온 문제점으로 Vibe Coding을 통해 나온 RVCcontroller는 Oracle와 달리 Cleaner와 Motor가 분리되어 있지 않다는 점을 꼽고 해당 사항에 대해 수정을 진행함.

결론적으로 이전 작업물에서 지켜지지 않던 SOLID의 단일 책임 원칙이 어느 정도 지켜지고 보다 직관적인 코드가 되었음



Oracle vs Vibe

Oracle 코드는 오랜 기간 고민하면서 나온 구조이기 때문에 조원 모두가 구조를 명확하게 인지하고 있었고, 어떤 부분을 고쳐야할지 바로 알고 수정하였음.

좌우 확인 후 뒤로가는 요구사항에 대해서 어떻게 해야 올바르게 동작할지 로직을 짜는 과정에 대부분의 시간을 소요함.

이후 코드 및 테스트에서는 별 다른 시간 소요 없이 마무리할 수 있었음

변경사항이 크게 없다는 것이 제일 중요함

Vibe Coding의 코드를 AI가 짜주고 결과물에 대해서 컨펌만 진행해왔기 때문에 요구사항에 대해 어떤 부분을 변화시켜야할지 인지하고 있지 못했고, AI에게 모든 일을 일임시키니 수정된 부분이 여러 있어 확인하는데 시간이 오래 소요되었음.

변경된 요구사항을 전달하는 것만으로 짧은 기간만에 수정이 완료되었으나, 코드가 직관적이지 않고 사람이 파악하기 힘든 부분이 다소 있었음

이에 따라 코드를 쉽게 풀어서 표현한 문서나 보조 도구가 있는 것이 중요하다는 것을 깨달음

Oracle vs Vibe

Oracle와 Vibe coding 둘 다 State 변수를 통해 RightSensor 제거 상황을 해결했다는 점은 동일하나

Oracle 코드보다 훨씬 단순하게 Vibe coding 코드가 작성되어 있어서 문제가 분명 있을 것이라 생각하고 코드를 전부 확인함.

그 과정에서 FrontSensor가 Interrupt로써 처리되고 있지 않음을 확인함.

FrontSensor 값을 받아오는 함수의 이름을 FrontObstacleInterrupt라고 명명하여 발견이 어려웠음.

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

현재 구조에서 "인터럽트 감지 즉시 모터 정지"만 추가한다면 범위는 작습니다.

- `RvcHardwareAdapter.hpp`에 `emergencyStop()` 같은 메서드 추가: 1~2줄
- `Rvc.cpp (line 24)`에서 front interrupt 감지 직후 `hardwareAdapter_->emergencyStop()` 호출: 1~3줄
- `SimulatedHardwareAdapter.cpp`에 구현 추가: 5~15줄
- 실제 하드웨어 adapter가 있다면 거기에 모터 stop 호출 추가: 5~20줄
- 테스트 추가/수정: 20~60줄

다만 시뮬레이터 기준으로는 지금도 interrupt가 있는 tick에서는 `Forward`가 아니라 `TurnLeft`, `TurnRight`, `Backward` 등으로 바뀌어서 "전진 명령을 계속 내리지 않는다"는 요구는 이미 만족해요.

진짜 차이는 controller 판단 전에 actuator 레벨에서 먼저 Stop을 박느냐입니다.

그래서 내가 구현한다면 최소 변경은 이런 느낌입니다.

```
cpp  
  
if (lastFrontObstacleInterrupt_) {  
    hardwareAdapter_->emergencyStop();  
    controller_.onFrontObstacleInterrupt();  
}
```

큰 설계 변경은 아니고, "하드웨어 adapter에 긴급정지 통로 하나 추가"하는 정도예요. 단, 로그에 emergency stop을 남기거나 ISR-thread/atomic/debounce까지 진짜 임베디드답게 넣으면 100줄 안팎까지는 금방 갑니다.