객체지향개발방법론
Practice #4

202211287 김태인

202311252 곽수호

202111368 정선민

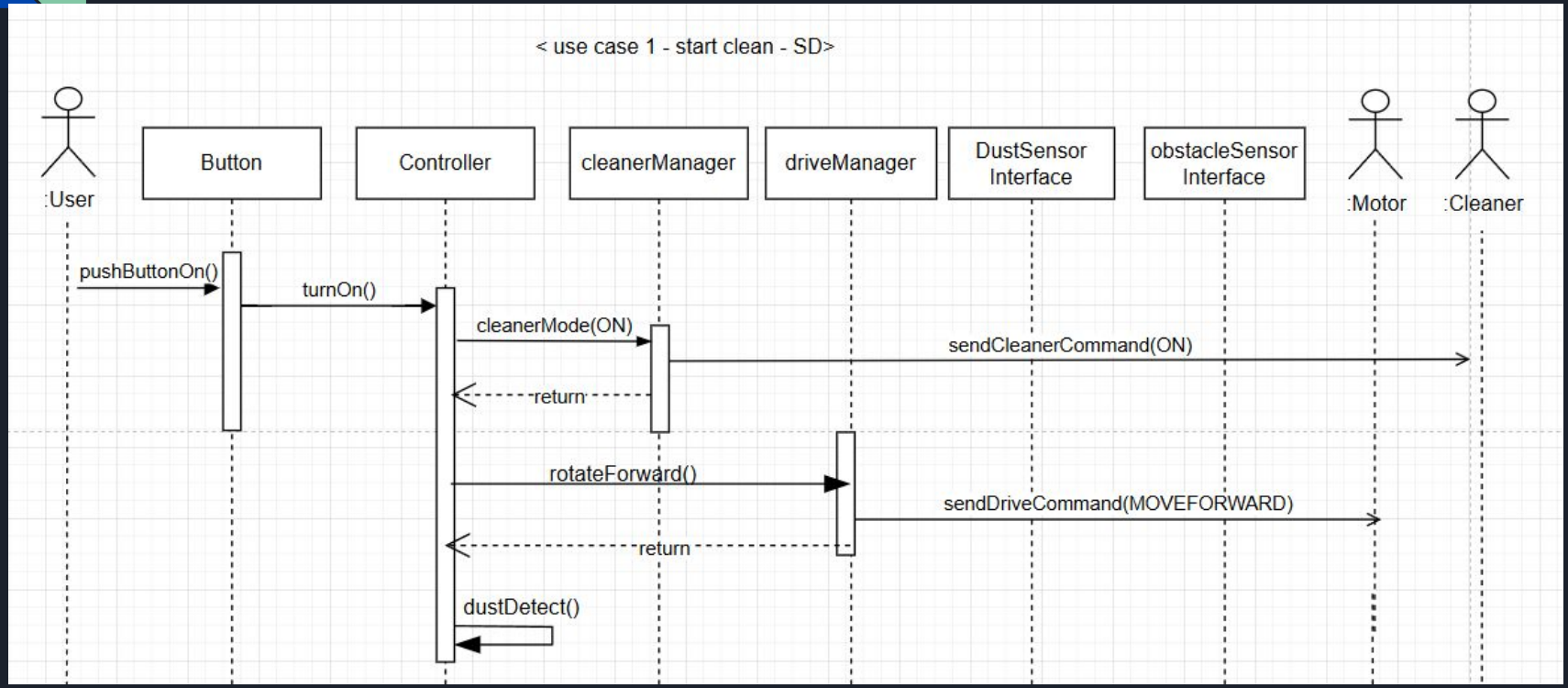
202211378 조성원

목차

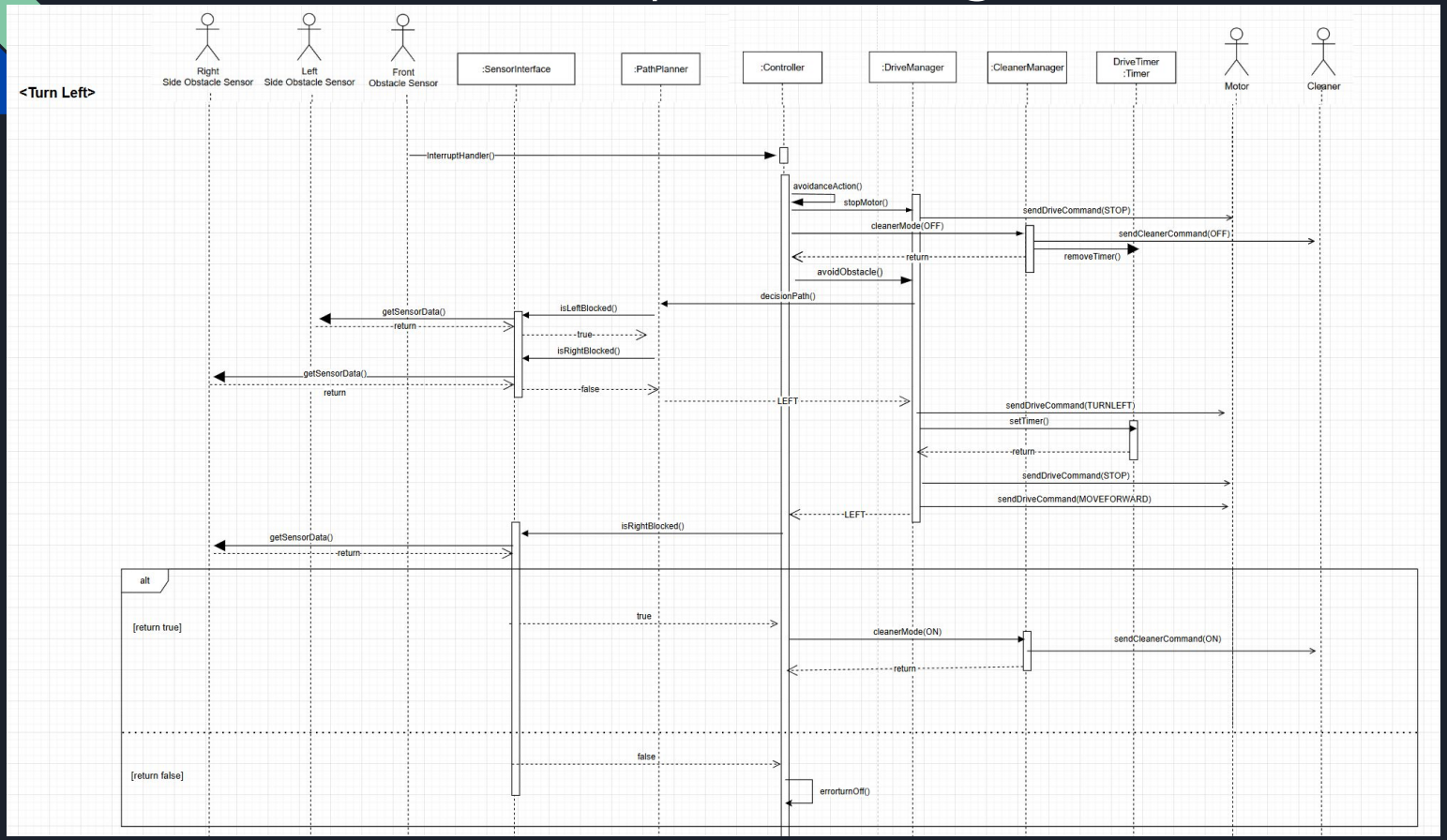
1. 변경사항 - (Refined FR)

R1.2Check Sensor Inputs	시작하고 obstacle과 dust sensor 값이 정상적으로 잘 들어오는 확인한다. 오류가 없을 시 클리너와 모터를 동작시킨다.
R4.1Pause motor	전방에 장애물을 만났을 때 motor를 정지한다.
R5.1Check Obstacle Sensor Error	앞에 장애물을 만나고 왼쪽, 오른쪽에 장애물이 없어서 어느 한 방향으로 회전할 경우, 회전 이후 회전 방향 반대 센서의 입력을 확인해서 sensor값이 잘 인식하는지 검증한다. 오류가 없을 시 클리너와 모터를 동작시킨다.

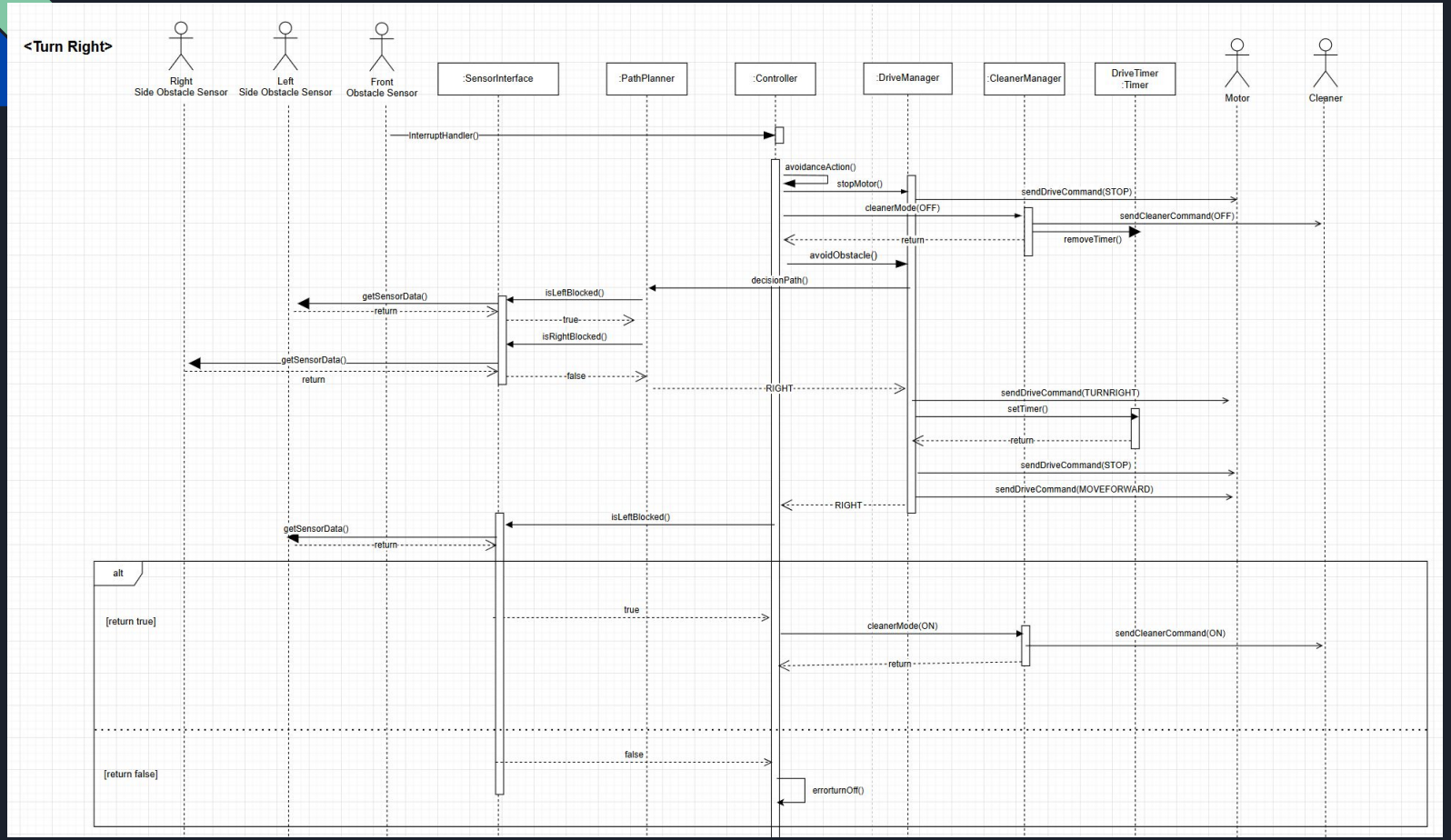
1. 변경사항 - Sequence diagram #1



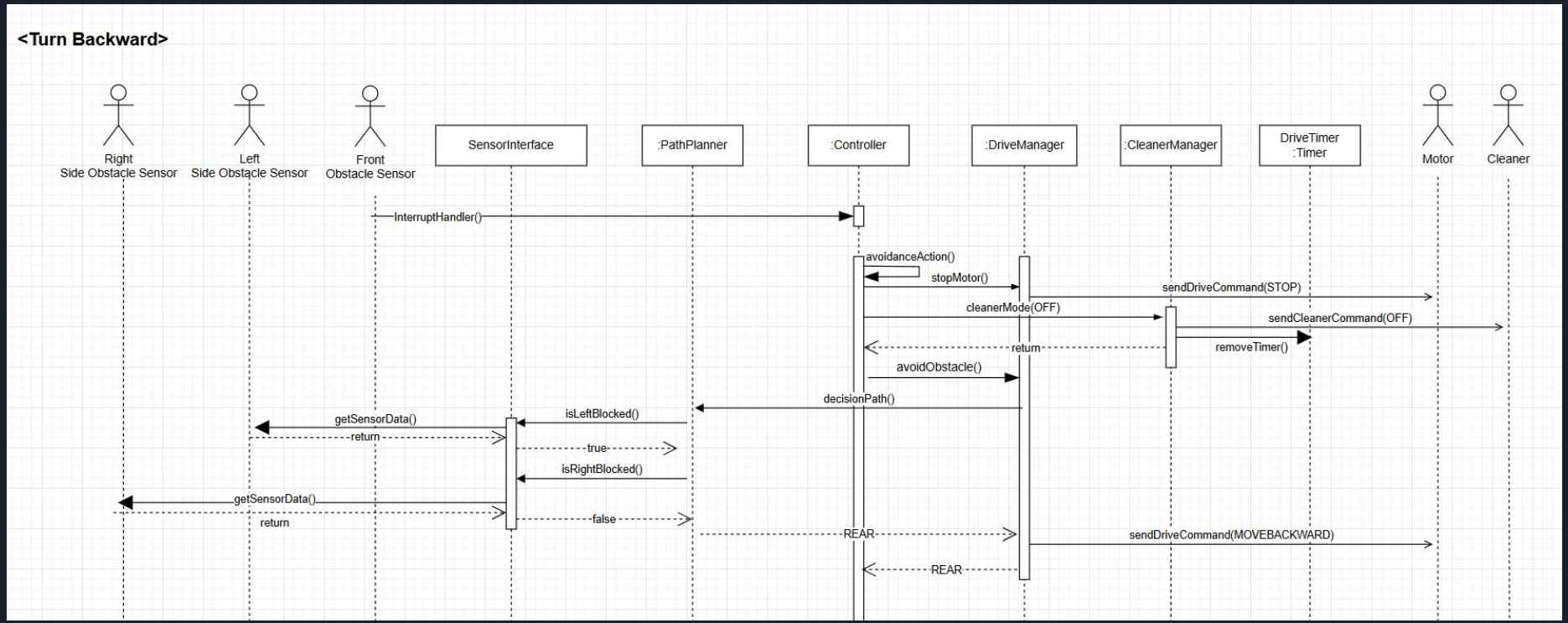
1. 변경사항 - Sequence diagram #2-1



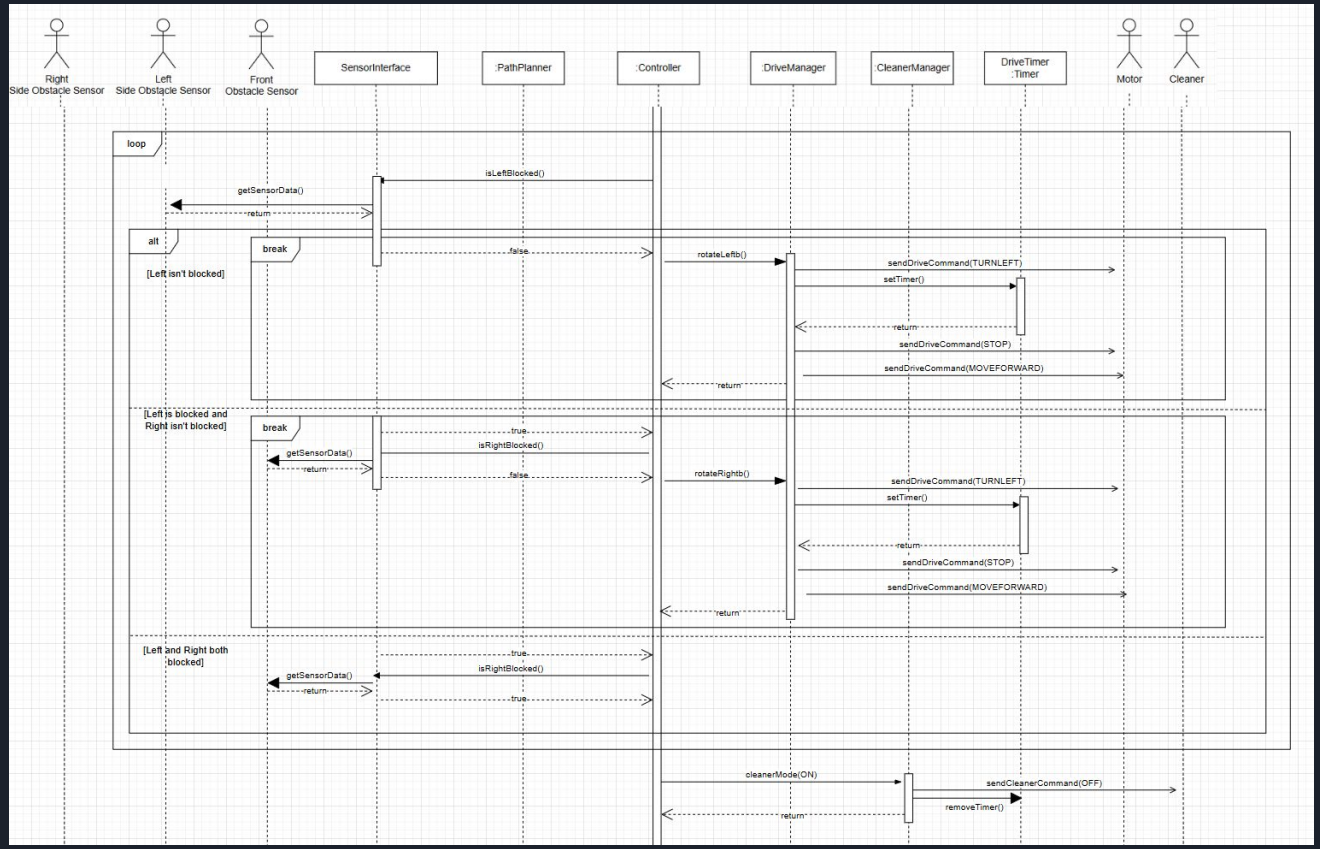
1. 변경사항 - Sequence diagram #2-2



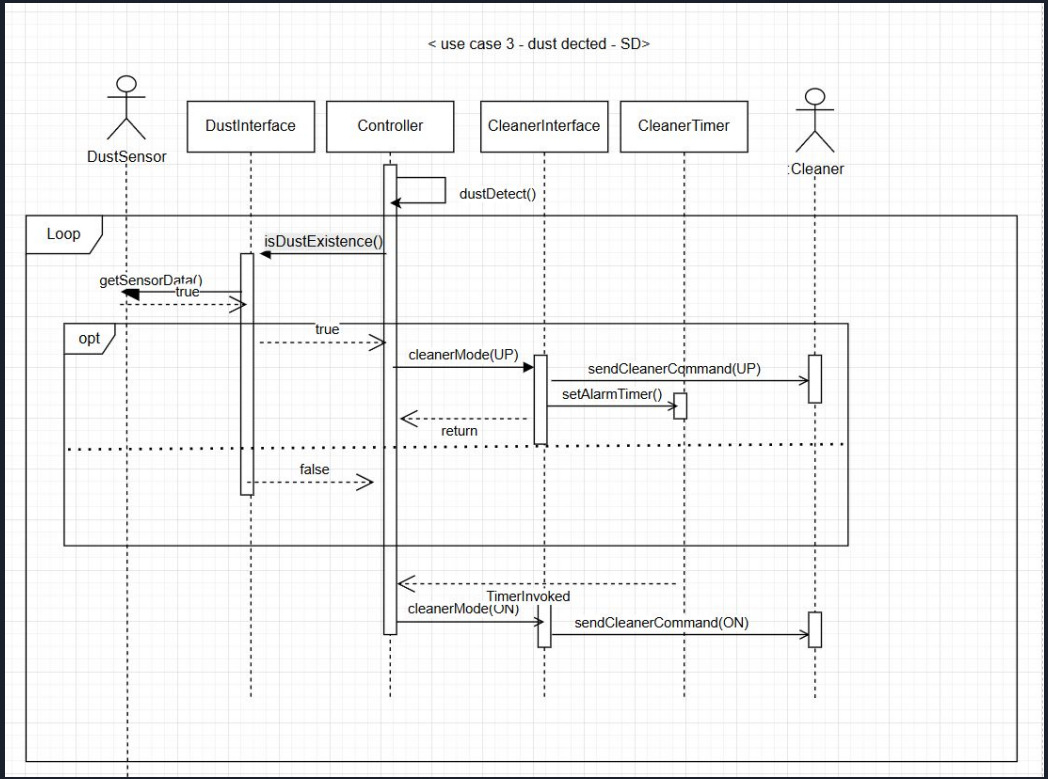
1. 변경사항 - Sequence diagram #2-3(1)



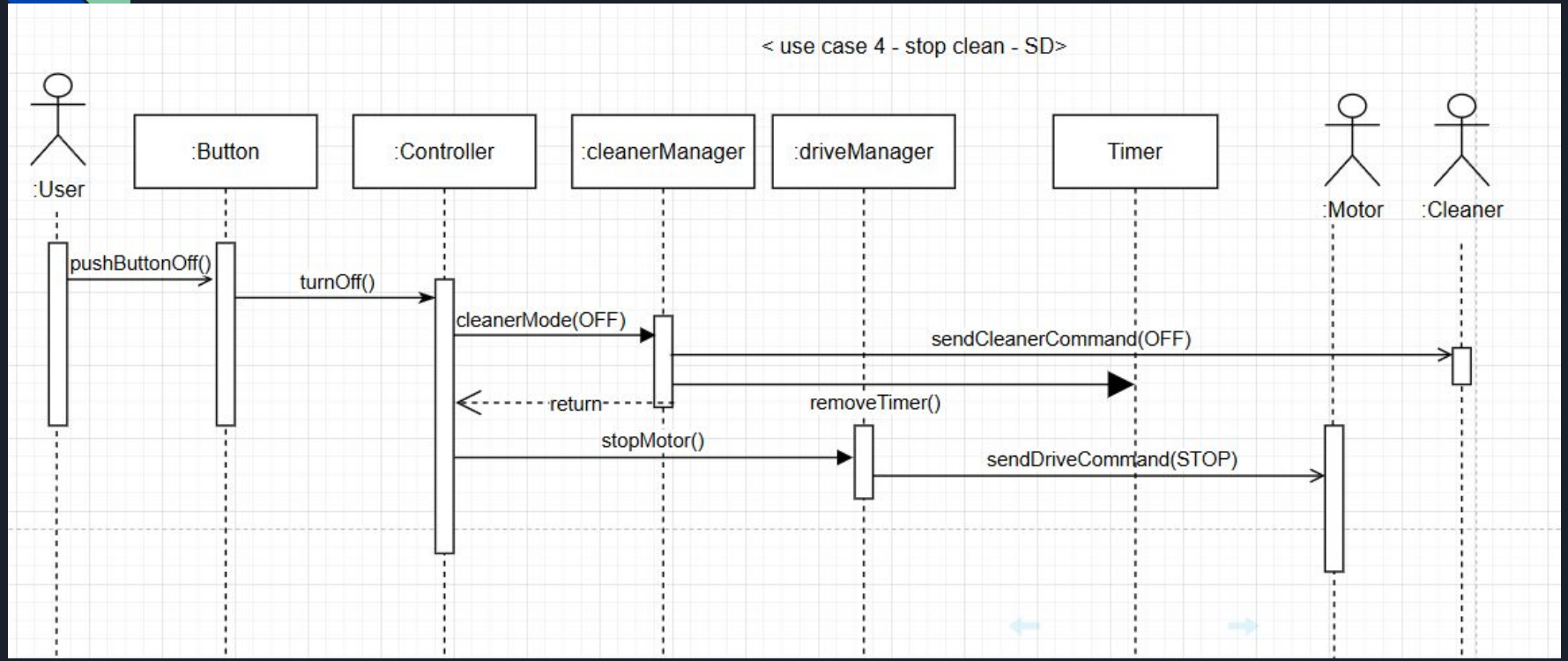
1. 변경사항 - Sequence diagram #2-3(2)



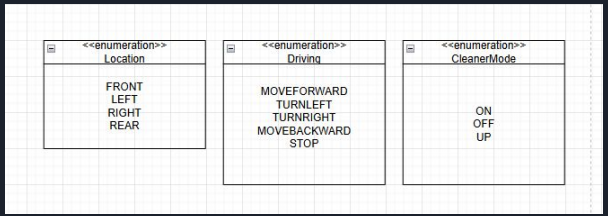
1. 변경사항 - Sequence diagram #3

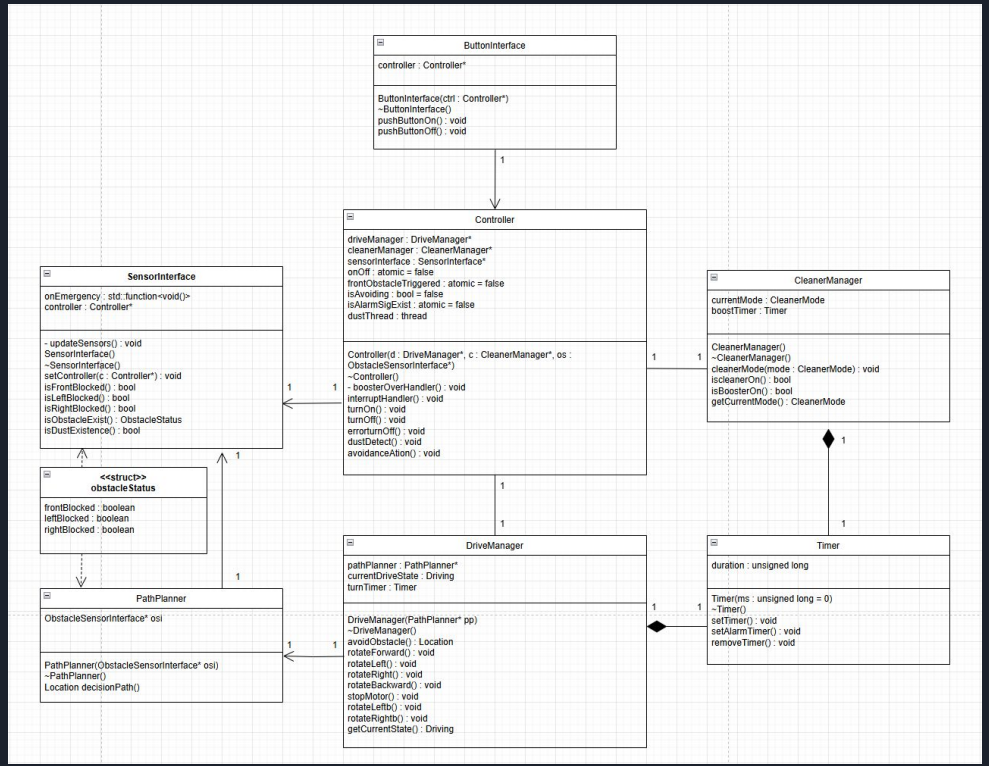


1. 변경사항 - Sequence diagram #4

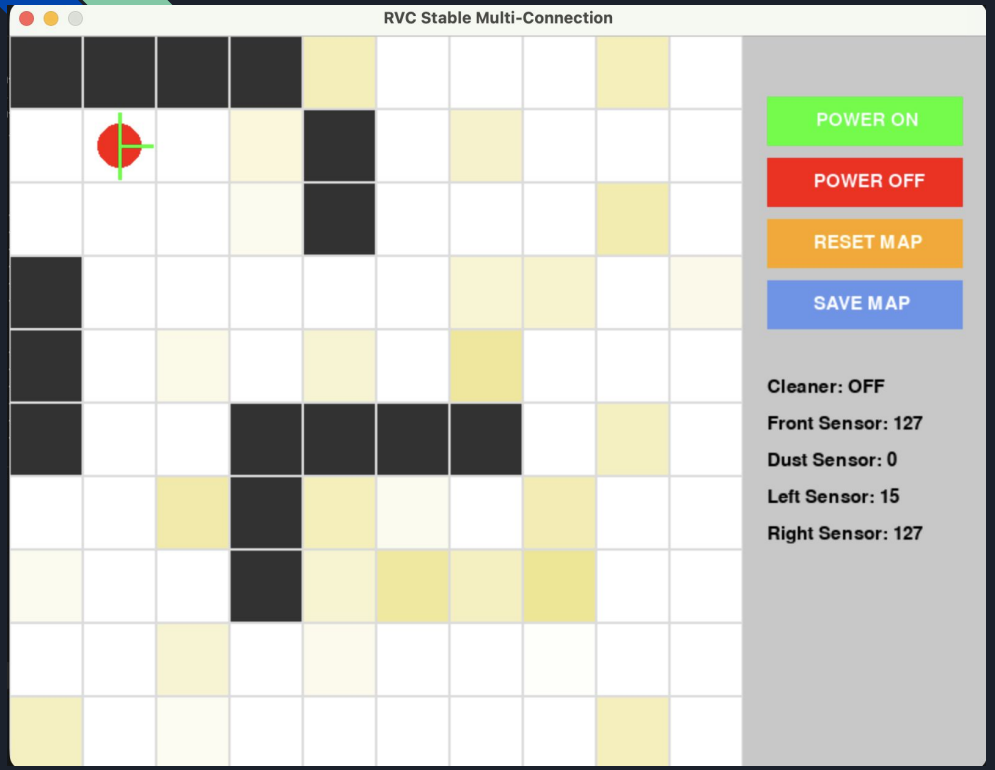


1. 변경사항 - class diagram





2. 시뮬레이터 환경



- 실제 하드웨어 없이 RVC의 제어 로직을 검증하기 위한 Virtual Hardware 환경 구축
- 로봇 중심으로 3 방향(전방, 좌, 우)에 대해 초록색으로 보이는 가상의 Ray를 투사하여 벽과의 거리 계산
- 맵 바닥의 색깔을 통해 Dust 여부를 반영
- 전방 센서가 측정한 거리가 특정 임계값 아래로 떨어지면 시뮬레이터가 클라이언트에 INTERUPT 이벤트를 즉시 PUSH
- 우측에 user가 버튼을 눌러주는 경우를 버튼으로 대체. 센서값, cleaner 상태를 인터페이스에서 실시간으로 확인 가능

2. 시뮬레이터 환경

Core Structure

- **Pygame GUI**: 10×10 grid map에서 로봇 위치, 방향, 장애물, 먼지 상태를 시각화
- **TCP Socket Communication**: C++ Controller와 Python Simulator를 socket으로 연결
  - Controller → Simulator: 이동, 회전, 청소 모드 제어 명령 전송
  - Simulator → Controller: front/left/right sensor, dust sensor 값 전달
- **Sensor Simulation**: raycasting 방식으로 장애물 거리 계산
- **Event Handling**: 장애물이 threshold 이하로 감지되면 interrupt event 발생
- **SAVE MAP**: 장애물·먼지 배치를 저장하여 동일 테스트 환경 재구성 가능

3. Unit-test (CleanerManager)

```
TEST_F(CleanerManagerTest, Test_iscleanerOn) {
    EXPECT_FALSE(cleanerManager->iscleanerOn());
    cleanerManager->cleanerMode(CleanerMode::ON);
    EXPECT_TRUE(cleanerManager->iscleanerOn());
    cleanerManager->cleanerMode(CleanerMode::OFF);
    EXPECT_FALSE(cleanerManager->iscleanerOn());
}
```

```
TEST_F(CleanerManagerTest, Test_getCurrentMode_iscleanerOn) {
    EXPECT_EQ(cleanerManager->getCurrentMode(), CleanerMode::OFF);
    cleanerManager->cleanerMode(CleanerMode::ON);
    EXPECT_EQ(cleanerManager->getCurrentMode(), CleanerMode::ON);
    cleanerManager->cleanerMode(CleanerMode::OFF);
    EXPECT_EQ(cleanerManager->getCurrentMode(), CleanerMode::OFF);
}
```

```
TEST_F(CleanerManagerTest, Test_cleanerMode_UP) {
    cleanerManager->cleanerMode(CleanerMode::UP);
    EXPECT_EQ(cleanerManager->getCurrentMode(), CleanerMode::UP);

    std::this_thread::sleep_for( d: std::chrono::milliseconds( r: 100));
    EXPECT_EQ(cleanerManager->getCurrentMode(), CleanerMode::UP);

    cleanerManager->cleanerMode(CleanerMode::OFF);
}
```

3. Unit-test (ButtonInterface)

```
// 1. 꺼진 상황에서 버튼을 눌렀을 때 turnOn()이 호출되는지 확인
TEST_F(ButtonInterfaceTest, PushButtonOn_WhenOff) {
    EXPECT_CALL(*mockController, turnOn()).Times(1);
    button->pushButtonOn();
}
```

```
// 2. 켜진 상황에서 버튼을 눌렀을 때 turnOff()가 호출되는지 확인
TEST_F(ButtonInterfaceTest, PushButtonOff_WhenOn) {
    EXPECT_CALL(*mockController, turnOff()).Times(1);
    button->pushButtonOff();
}
```

3. Unit-test (DriveManager)

```
// 1. 기본 주행 상태 변경 테스트
TEST_F(DriveManagerTest, DrivingTest_rotateXXX_getCurrentState) {
    driveManager->rotateForward();
    EXPECT_EQ(driveManager->getCurrentState(), Driving::MOVEFORWARD);

    driveManager->rotateLeft();
    EXPECT_EQ(driveManager->getCurrentState(), Driving::TURNLEFT);

    driveManager->rotateRight();
    EXPECT_EQ(driveManager->getCurrentState(), Driving::TURNRIGHT);

    driveManager->rotateBackward();
    EXPECT_EQ(driveManager->getCurrentState(), Driving::MOVEBACKWARD);

    driveManager->stopMotor();
    EXPECT_EQ(driveManager->getCurrentState(), Driving::STOP);
}
```

```
// 2. 회피 로직 호출 및 상태 복구 테스트
TEST_F(DriveManagerTest, AvoidObstacle_LogicTest) {
    // PathPlanner가 왼쪽으로 가라고 명령할 것임을 설정
    EXPECT_CALL(*mockPP, decisionPath()).WillOnce( Return(value: Location::LEFT))

    Location result = driveManager->avoidObstacle();

    EXPECT_EQ(result, Location::LEFT);
    // 회피 후 다시 전진 상태로 복구되었는지 확인
    EXPECT_EQ(driveManager->getCurrentState(), Driving::MOVEFORWARD);
}
```

```
// 3. 특수 회전 로직 테스트
TEST_F(DriveManagerTest, RotateRightb_ExecutionTest) {
    driveManager->rotateRightb();
    EXPECT_EQ(driveManager->getCurrentState(), Driving::MOVEFORWARD);
}
```

3. Unit-test (Controller)

```
// 1. turnOn() 시 내부 컴포넌트 호출 검증
TEST_F(ControllerTest, TurnOn_CallsComponents) {
    EXPECT_CALL(*mockCM, cleanerMode( gmock_a0: CleanerMode::ON)).Times(1);
    EXPECT_CALL(*mockDM, rotateForward()).Times(1);

    controller->turnOn();
}
```

```
// 2. turnOff() 시 내부 컴포넌트 호출 검증
TEST_F(ControllerTest, TurnOff_CallsComponents) {
    controller->turnOn();

    EXPECT_CALL(*mockCM, cleanerMode( gmock_a0: CleanerMode::OFF)).Times(1);
    EXPECT_CALL(*mockDM, stopMotor()).Times(1);

    controller->turnOff();
}
```

```
// 3. 장애물 감지 시 회피 로직 호출 검증
TEST_F(ControllerTest, avoidanceAction_TriggersAvoidance) {
    struct TestController : public Controller {
        using Controller::Controller;
    };
    TestController* testCtrl = new TestController(mockDM, mockCM, mockOS);

    // 전방 장애물 감지 시나리오
    EXPECT_CALL(*mockDM, stopMotor()).Times(AtLeast( n: 1));
    EXPECT_CALL(*mockCM, cleanerMode( gmock_a0: CleanerMode::OFF)).Times(AtLeast(
    EXPECT_CALL(*mockDM, avoidObstacle()).WillOnce( Return( value: Location::LEFT))

    // 회피 후 정면 체크 (성공했다고 가정하여 false 리턴)=

    EXPECT_CALL(*mockOS, isRightBlocked()).WillOnce( Return( value: true));

    // 회피 성공 후 청소기를 다시 ON으로 돌리는 호출 추가
    EXPECT_CALL(*mockCM, cleanerMode( gmock_a0: CleanerMode::ON)).Times(AtLeast(

    testCtrl->avoidanceAction();

    delete testCtrl;
}
```

3. Unit-test (SensorInterface)

```
// 1-1-1 전방센서 장애물 여부 반환값 확인 (경계값 테스트)
TEST_F(ObstacleSensorInterfaceTest, Check_Front_Blocked) {
    const int values_true[] = {2, 10, 19, 20, 21, 52, 76};
    for (int v : values_true) {
        if (values_true[v]> osi->threshold) {
            EXPECT_CALL(*mockOBS, isFrontBlocked()).WillOnce( Return( value: true))
            EXPECT_EQ(mockOBS->isFrontBlocked(), true);
        }
        else {
            EXPECT_CALL(*mockOBS, isFrontBlocked()).WillOnce( Return( value: false)
            EXPECT_EQ(mockOBS->isFrontBlocked(), false);
        }

    }

}
```

```
// 1-1-3 우측센서 장애물 여부 반환값 확인 (경계값 테스트)
TEST_F(ObstacleSensorInterfaceTest, Check_Right_Blocked) {
    const int values[] = {5, 33, 59, 60, 61, 91, 125};
    for (int v : values) {
        if (values[v]> osi->thresholdside) {
            EXPECT_CALL(*mockOBS, isRightBlocked()).WillOnce( Return( value: true))
            EXPECT_EQ(mockOBS->isRightBlocked(), true);
        }
        else {
            EXPECT_CALL(*mockOBS, isRightBlocked()).WillOnce( Return( value: false)
            EXPECT_EQ(mockOBS->isRightBlocked(), false);
        }
    }
```

```
// 1-1-2 좌측센서 장애물 여부 반환값 확인 (경계값 테스트)
TEST_F(ObstacleSensorInterfaceTest, Check_Left_Blocked) {
    const int values[] = {18, 44, 59, 60, 61,73, 115};
    for (int v : values) {
        if (values[v]> osi->thresholdside) {
            EXPECT_CALL(*mockOBS, isLeftBlocked()).WillOnce( Return( value: true));
            EXPECT_EQ(mockOBS->isLeftBlocked(), true);
        }
        else {
            EXPECT_CALL(*mockOBS, isLeftBlocked()).WillOnce( Return( value: false))
            EXPECT_EQ(mockOBS->isLeftBlocked(), false);
        }
    }
}
```

```
// 2-1 먼지 여부 반환값 확인
TEST_F(ObstacleSensorInterfaceTest, Check_Dust_Existence) {
    // EXPECT_CALL(*mockOBS, isDustExistence()).WillOnce(Return(false));
    // EXPECT_EQ(mockOBS->isDustExistence(), false);

    const int dust_values[] = {0, 1, 27, 101};
    for (int v : dust_values) {
    if (dust_values[v]>0) {
        EXPECT_CALL(*mockOBS, isDustExistence()).WillOnce( Return( value: true));
        EXPECT_EQ(mockOBS->isDustExistence(), true);
    }else {
        EXPECT_CALL(*mockOBS, isDustExistence()).WillOnce( Return( value: false));
        EXPECT_EQ(mockOBS->isDustExistence(), false);
        }
    }
}
```

3. Unit-test (PathPlanner)

```
// 1. 왼쪽이 비어있을 때 왼쪽으로 결정을 내리는지 확인
TEST_F(PathPlannerTest, decisionPath_TurnLeft) {
    // Expectation: isLeftBlocked() 호출 시 false 리턴
    EXPECT_CALL(*mockOSI, isLeftBlocked()).WillOnce( Return( value: false));

    EXPECT_EQ(pathPlanner->decisionPath(), Location::LEFT);
}
```

```
// 2. 왼쪽이 막히고 오른쪽이 비어있을 때 오른쪽으로 결정을 내리는지 확인
TEST_F(PathPlannerTest, decisionPath_TurnRight) {
    // Expectation: 왼쪽은 막히고, 오른쪽은 뚫림
    EXPECT_CALL(*mockOSI, isLeftBlocked()).WillOnce( Return( value: true));
    EXPECT_CALL(*mockOSI, isRightBlocked()).WillOnce( Return( value: false));

    EXPECT_EQ(pathPlanner->decisionPath(), Location::RIGHT);
}
```

```
// 3. 양쪽이 모두 막혔을 때 뒤로 결정을 내리는지 확인
TEST_F(PathPlannerTest, decisionPath_MoveBackward) {
    // Expectation: 양쪽 모두 막힘
    EXPECT_CALL(*mockOSI, isLeftBlocked()).WillOnce( Return( value: true));
    EXPECT_CALL(*mockOSI, isRightBlocked()).WillOnce( Return( value: true));

    EXPECT_EQ(pathPlanner->decisionPath(), Location::REAR);
}
```

3. Unit-test (Timer)

```cpp
// 1. setTimer()가 설정된 시간(ms)만큼 대기하는지 확인
TEST_F(TimerTest, SetTimer_WaitCorrectly) {
    unsigned long waitTime = 100;
    Timer timer(waitTime);

    auto start :time_point<steady_clock> = std::chrono::high_resolution_clock::now();
    timer.setTimer(); // 실제 std::this_thread::sleep_for 실행
    auto end :time_point<steady_clock> = std::chrono::high_resolution_clock::now();

    auto elapsed :long long = std::chrono::duration_cast<std::chrono::milliseconds>( fd

    // OS 스케줄링 오차를 고려하여 검증 (보통 5~10ms 내외)
    EXPECT_GE(elapsed, waitTime - 5);
}
```

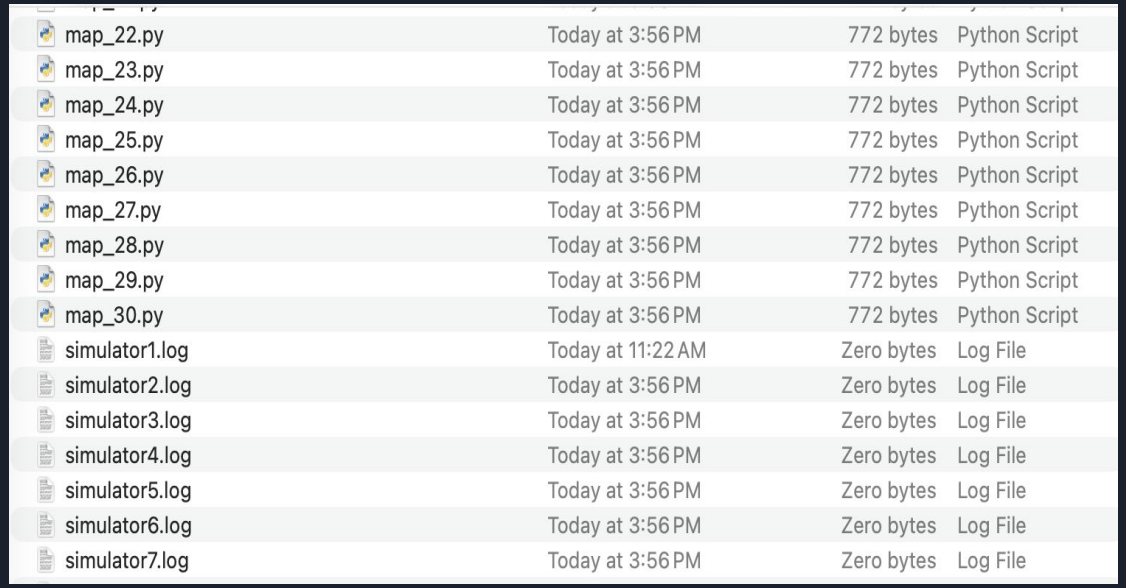
```cpp
// 2. setAlarmTimer() 호출 시 예외가 발생하지 않는지 확인
TEST_F(TimerTest, SetAlarmTimer_Execution) {
    Timer timer( ms: 1000);

    // 시스템 호출(setitimer)이 성공적으로 수행되는지 확인
    EXPECT_NO_THROW(timer.setAlarmTimer());

    // 테스트용 알람이 실제 SIGALRM을 발생시키기 전에 바로 제거
    timer.removeTimer();
}
```

```cpp
// 3. removeTimer() 호출 시 예외가 발생하지 않는지 확인
TEST_F(TimerTest, RemoveTimer_Execution) {
    Timer timer( ms: 0);
    EXPECT_NO_THROW(timer.removeTimer());
}
```

4. System test

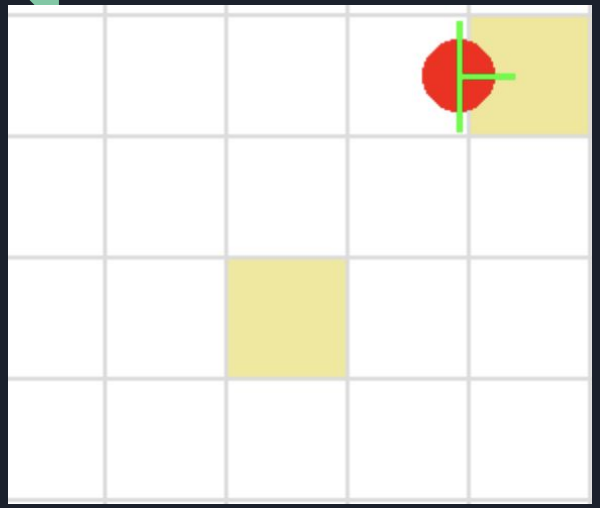


map_22.py	Today at 3:56 PM	772 bytes	Python Script
map_23.py	Today at 3:56 PM	772 bytes	Python Script
map_24.py	Today at 3:56 PM	772 bytes	Python Script
map_25.py	Today at 3:56 PM	772 bytes	Python Script
map_26.py	Today at 3:56 PM	772 bytes	Python Script
map_27.py	Today at 3:56 PM	772 bytes	Python Script
map_28.py	Today at 3:56 PM	772 bytes	Python Script
map_29.py	Today at 3:56 PM	772 bytes	Python Script
map_30.py	Today at 3:56 PM	772 bytes	Python Script
simulator1.log	Today at 11:22 AM	Zero bytes	Log File
simulator2.log	Today at 3:56 PM	Zero bytes	Log File
simulator3.log	Today at 3:56 PM	Zero bytes	Log File
simulator4.log	Today at 3:56 PM	Zero bytes	Log File
simulator5.log	Today at 3:56 PM	Zero bytes	Log File
simulator6.log	Today at 3:56 PM	Zero bytes	Log File
simulator7.log	Today at 3:56 PM	Zero bytes	Log File

저희팀은 Redmine과 같은 관리 툴을 사용하는 대신

1. 실제 로봇의 동작을 가장 명확하게 입증할 수 있는 로깅 시스템을 구축
2. 'SAVEMAP' 기능을 통해 복잡한 장애물 상황도 언제든 재현할 수 있게 설계

4. System test ST-3

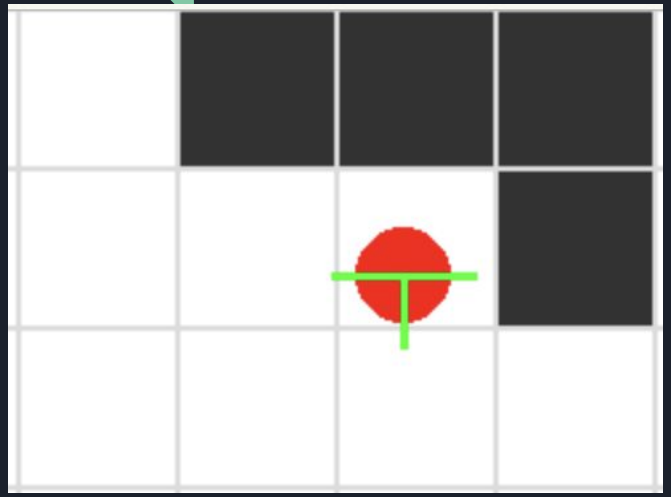


```
[Interface] Received Event: BUTTON_ON
[System] POWER ON
start moveforward
Power On Clicked
[Interface] Received Event: BUTTON_ON
Power On Clicked
[Interface] Received Event: BUTTON_ON
Power On Clicked
[Interface] Received Event: BUTTON_ON
Power On Clicked
[Interface] Received Event: BUTTON_ON
Power On Clicked
[Interface] Received Event: BUTTON_ON
Power On Clicked
[Interface] Received Event: BUTTON_ON
[Interface] Received Event: BUTTON_ON
```

ST-03 Power ON 중복 입력 방지

시스템이 이미 ON 상태, pushButtonOn()을 여러 번 호출, 중복 동작 없이 기존 ON 상태 유지, 불필요한 thread 재생성 없음 onOff guard, Controller 상태 관리

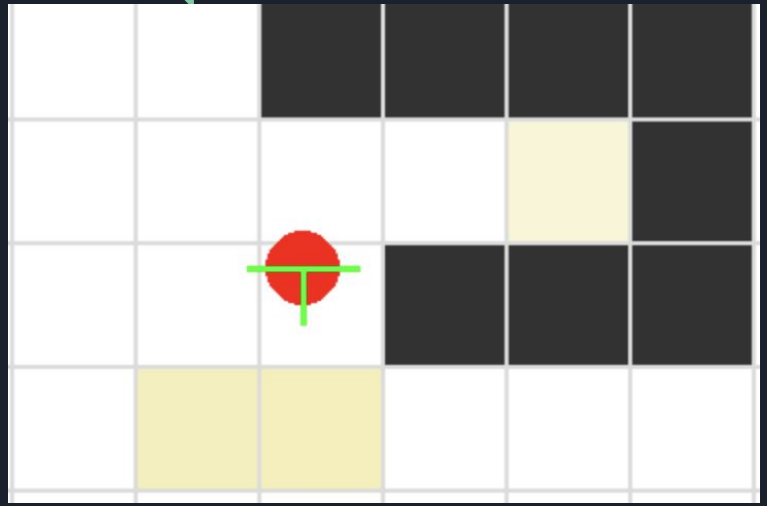
4. System test ST-9



```
[System] POWER ON
start moveforward
[Simulator] INTERRUPT Triggere
d! Distance: 19
front sensor inerrupt occured
front sensor interrupt
[System] Obstacle Detected! St
arting Avoidance...
```

ST-09 전방 장애물 Interrupt 처리
시스템 ON, 주행 중 Controller::interruptHandler() 호출되어 rvc가 회전경로를 탐색

4. System test ST-14

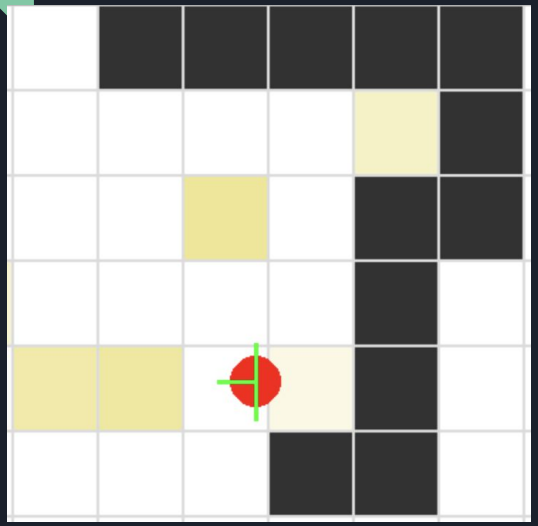


```
[Simulator] INTERRUPT Triggere
d! Distance: 19
front sensor inerrupt occured
front sensor interrupt
[System] Obstacle Detected! St
arting Avoidance...
trun backward start
[System] Still Blocked! Mainta
ining Flag...
front sensor interrupt
[System] Obstacle Detected! St
arting Avoidance...
trun backward start
[System] Rear blocked path - S
earching for exit...
[System] Right side cleared! E
scaping to Right.
start moveforward
```

ST-14 후방회피 감지

시스템이 이미 ON 상태, 전진중 장애물을 만나 회피경로를 결정할 때 양옆에 장애물이 있을 경우 후진해서 경로 재탐색하다 왼쪽이 먼지 비게되면 왼쪽으로 회전

4. System test ST-19,20

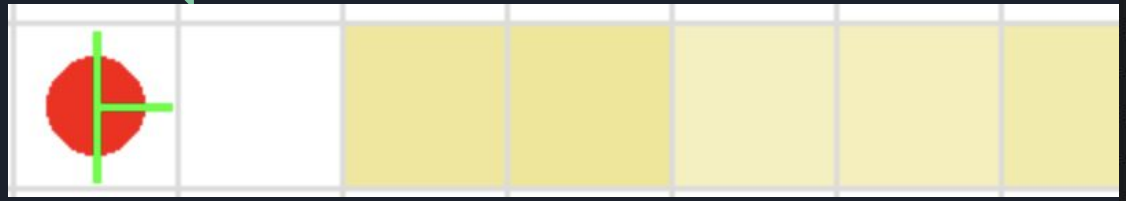


ST-19 회피 성공 후 flag 초기화, ST-20 회피 성공 후 cleaner 복귀
시스템이 전방 장애물 인터룹트가 발생하여 회피경로를 찾아 회피 성공, 그 이후 다음 인터룹트까지 cleaner가 켜지고 다음 회피도 잘 수행하는지

```
front sensor interrupt
[System] Obstacle Detected! St
arting Avoidance...
trun backward start
[System] Rear blocked path - S
earching for exit...
[System] Right side cleared! E
scaping to Right.
start moveforward
[System] Starting 1000ms timer
Waiting for SIGALRM
[Wait-Thread] SIGALRM Caught!
Routing to Controller...
[System] Starting 1000ms timer
Waiting for SIGALRM
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
```

```
[Simulator] INTERRUPT Triggere
d! Distance: 19
front sensor inerrupt occured
front sensor interrupt
[System] Obstacle Detected! St
arting Avoidance...
[System] Right side cleared! E
scaping to Right.
start moveforward
```

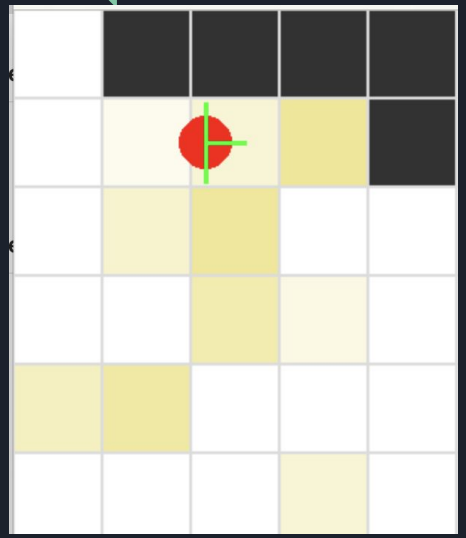
4. System test ST-22

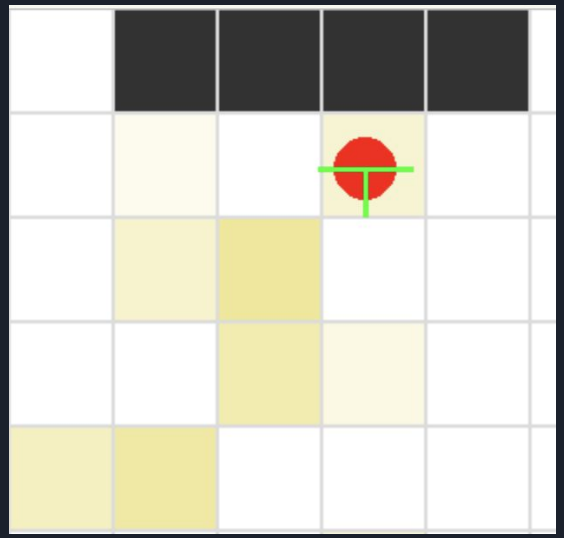


ST-22 Booster 중 추가 먼지 감지
이미 CleanerMode::UP 상태 먼지 감지 true 유지 ,
시스템 오류 없이 UP 상태 유지 -> booster timer 재설정

```
start moveforward
[System] Starting 1000ms timer
Waiting for SIGALRM
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
[System] Starting 1000ms timer
```

4. System test ST-32





```
[System] Obstacle Detected! St
arting Avoidance...
[System] Right side cleared! E
scaping to Right.
Wall toggled at: 8, 1
start moveforward
[System] POWER OFF
```

ST-32: 우회피 후 Error TurnOff()

전방 장애물 발생, PathPlanner가 RIGHT 반환, 회피 후 왼쪽 센서가 clear 상태
avoidanceAction() 실행, Controller가 errorturnOff()를 호출하고 시스템이 OFF 상태로 전환됨 errorturnOff(), turnOff(), Cleaner OFF, Drive STOP

서버에서 테스트

Test Results

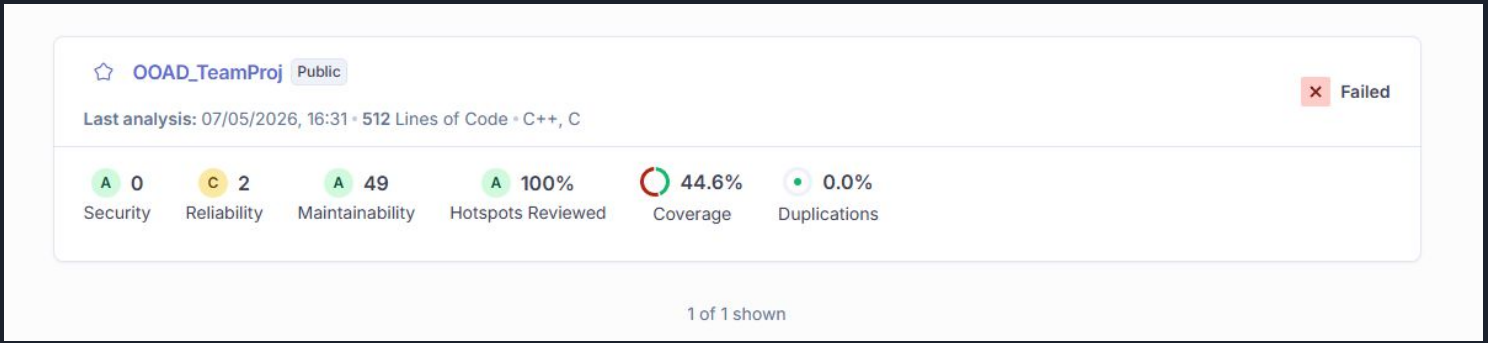
23 tests -1 | 23 ✅ -1 | 4s ⏱️ -1
1 suites ±0 | 0 💤 ±0
1 files ±0 | 0 ❌ ±0

Results for commit d7929da4. ± Com
9dbbd448.

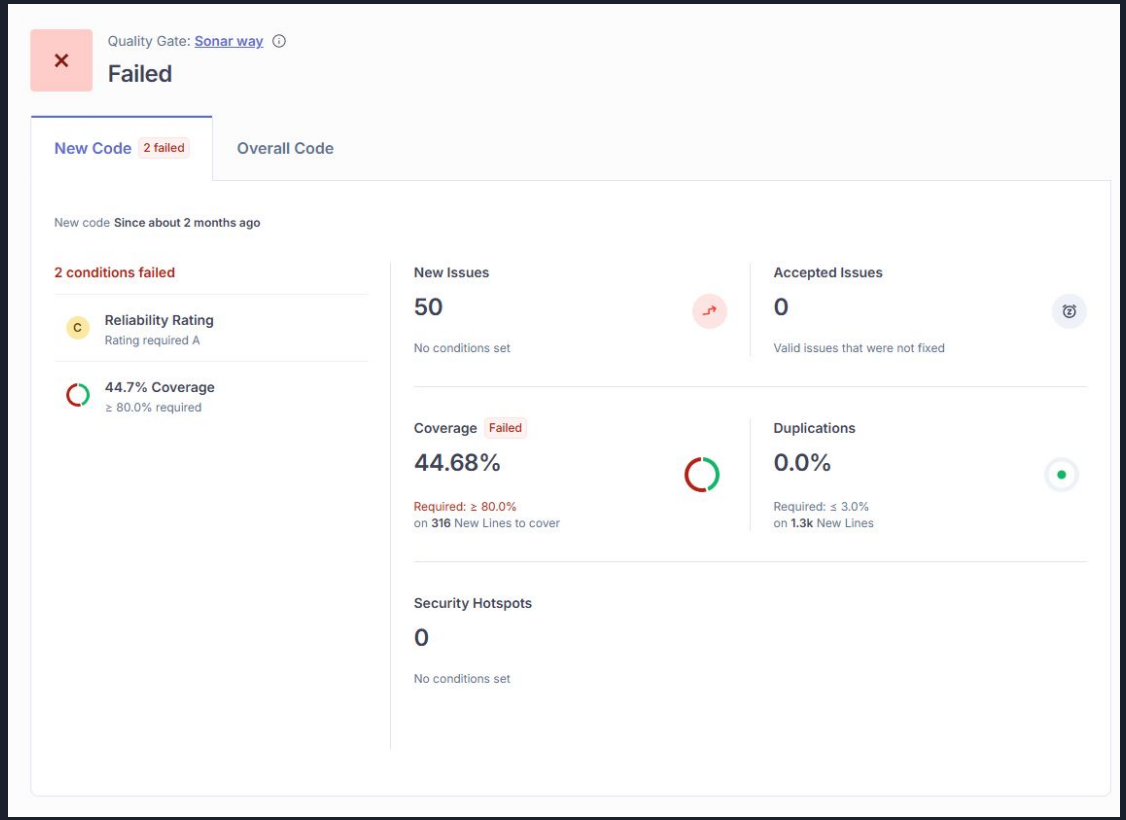
Job summary generated at run-time

```
▶ Run # 1. ctest 실행 시 결과를 XML로 출력하도록 옵션(--output-junit) 추가
Internal ctest changing into directory:
/home/runner/work/OOAD_TeamProj/OOAD_TeamProj/build
Test project /home/runner/work/OOAD_TeamProj/OOAD_TeamProj/build
      Start  1: ButtonInterfaceTest.PushButtonOn_WhenOff
 1/24 Test  #1: ButtonInterfaceTest.PushButtonOn_WhenOff
.........................   Passed    1.01 sec
      Start  2: ButtonInterfaceTest.PushButtonOff_WhenOn
 2/24 Test  #2: ButtonInterfaceTest.PushButtonOff_WhenOn
.........................   Passed    1.01 sec
      Start  3: ControllerTest.TurnOn_CallsComponents
 3/24 Test  #3: ControllerTest.TurnOn_CallsComponents
.............................   Passed    1.01 sec
      Start  4: ControllerTest.TurnOff_CallsComponents
 4/24 Test  #4: ControllerTest.TurnOff_CallsComponents
.............................   Passed    1.01 sec
      Start  5: ControllerTest.avoidanceAction_TriggersAvoidance
 5/24 Test  #5: ControllerTest.avoidanceAction_TriggersAvoidance
..................   Passed    1.01 sec
      Start  6:
DriveManagerTest.DrivingTest_rotateXXX_getCurrentState
 6/24 Test  #6:
DriveManagerTest.DrivingTest_rotateXXX_getCurrentState .....
```

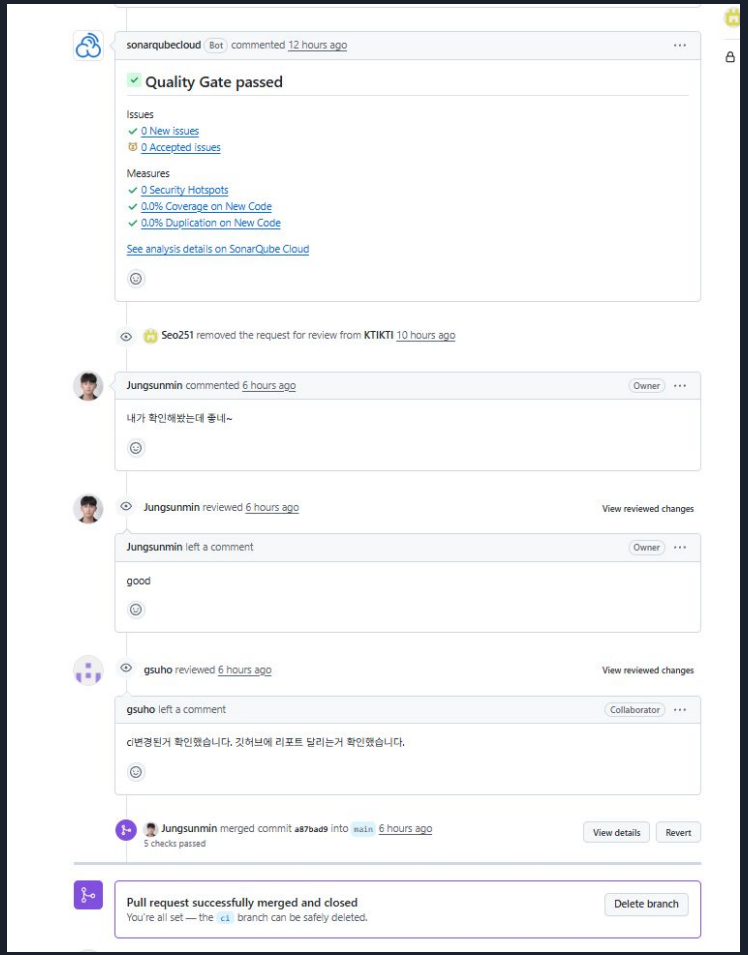
Static Code Analysis



Static Code Analysis



Code Review


sonarqubecloud Bot commented 12 hours ago
Quality Gate passed
Issues
0 New issues
0 Accepted Issues
Measures
0 Security Hotspots
0.0% Coverage on New Code
0.0% Duplication on New Code
See analysis details on SonarQube Cloud
Seo251 removed the request for review from KTIKTI 10 hours ago
Jungsunmin commented 6 hours ago
Owner
내가 확인해봤는데 좋네~
Jungsunmin reviewed 6 hours ago
View reviewed changes
Jungsunmin left a comment
Owner
good
gsuho reviewed 6 hours ago
View reviewed changes
gsuho left a comment
Collaborator
ci변경된거 확인했습니다. 깃허브에 리포트 달리는거 확인했습니다.
Jungsunmin merged commit a87bad9 into main 6 hours ago
5 checks passed
View details
Revert
Pull request successfully merged and closed
You're all set — the ci branch can be safely deleted.
Delete branch

감사합니다.