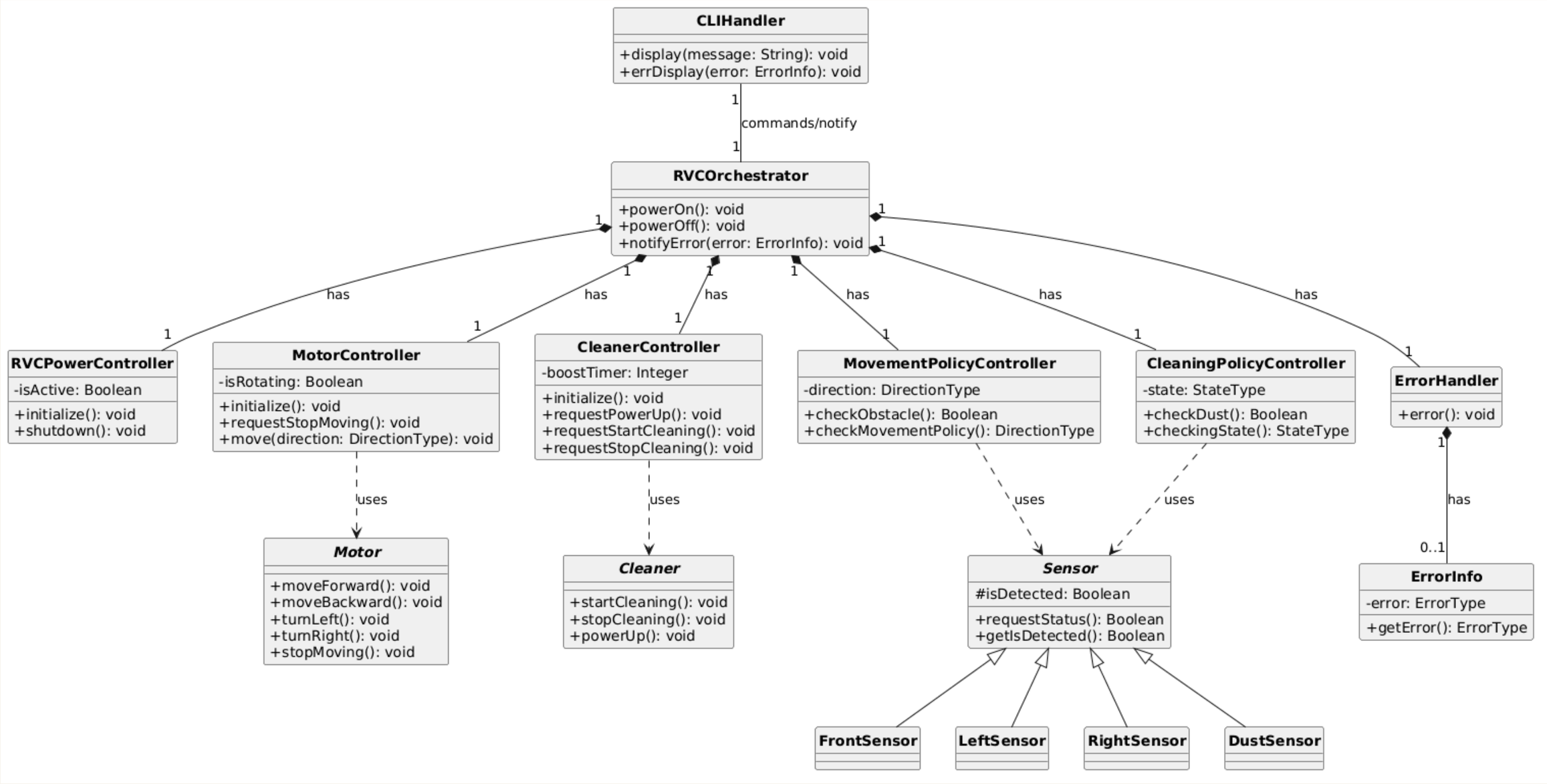
객체지향개발방법론
팀프로젝트#4

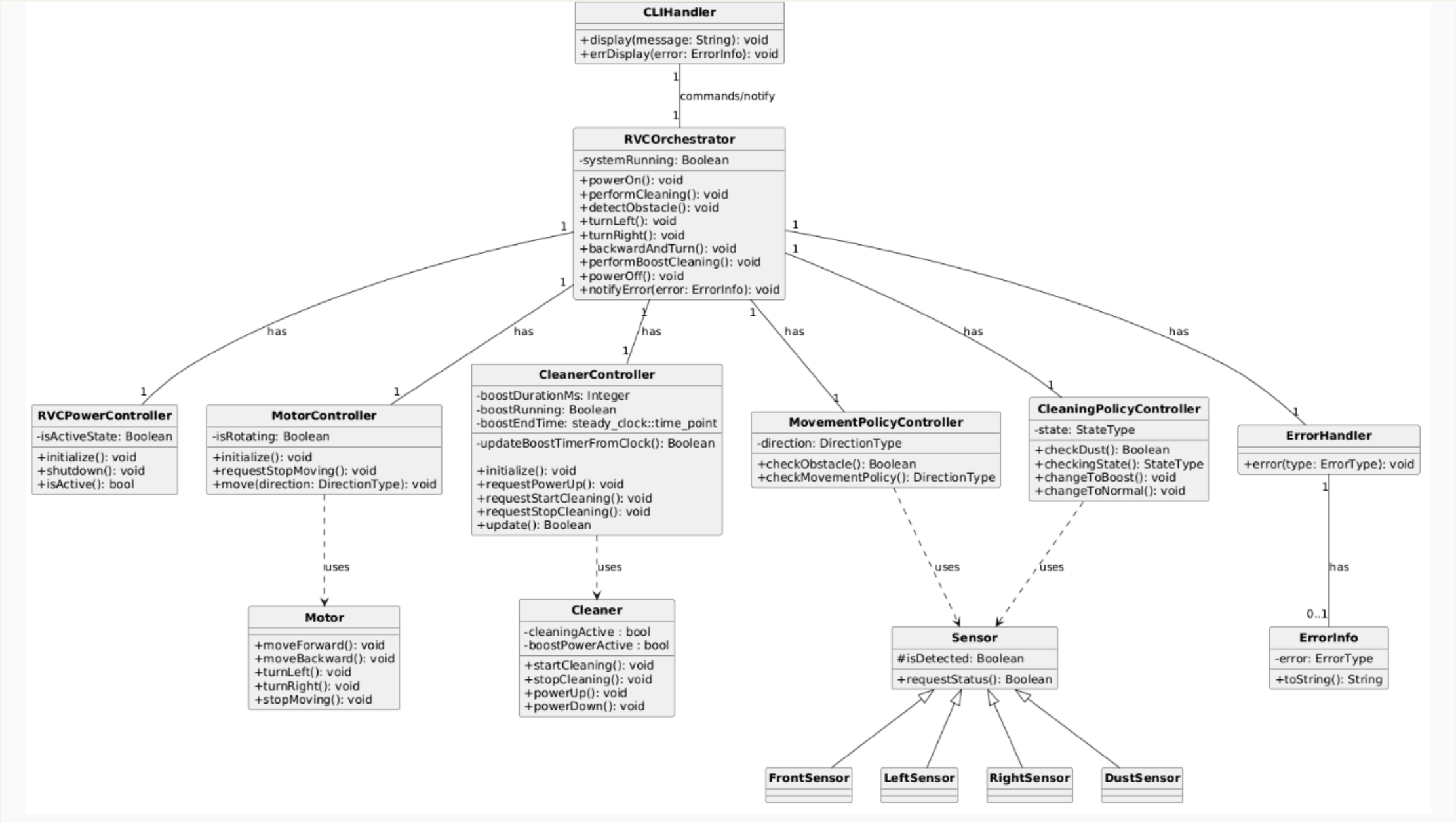
2026.05.07

7팀 강희진 김희용 양지윤 윤성진

Class Diagram – Before



Class Diagram – After



구현 (Coding)

CleanerController.cpp

```cpp
#include "controller/CleanerController.h"

CleanerController::CleanerController(int durationMs)
    : boostDurationMs(durationMs), cleanerPtr(&cleaner) {}

CleanerController::CleanerController(ICleaner* c)
    : boostDurationMs(0), cleanerPtr(c) {}

void CleanerController::initialize() {
    boostRunning = false;
    boostEndTime = {};
}

void CleanerController::requestStartCleaning() {
    cleanerPtr->startCleaning();
}

void CleanerController::requestStopCleaning() {
    cleanerPtr->stopCleaning();

    boostRunning = false;
    boostEndTime = {};
}
```

```cpp
void CleanerController::requestPowerUp() {
    cleanerPtr->powerUp();

    const auto now :time_point<steady_clock> = std::chrono::steady_clock::now();
    boostEndTime = now + std::chrono::milliseconds(boostDurationMs);

    boostRunning = true;
}

bool CleanerController::update() {
    return updateBoostTimerFromClock();
}

bool CleanerController::updateBoostTimerFromClock() {
    if (!boostRunning) {
        return false;
    }

    const auto now :time_point<steady_clock> = std::chrono::steady_clock::now();

    if (now >= boostEndTime) {
        cleaner.powerDown();

        boostRunning = false;

        return true;
    }

    return false;
}
```

- CleanerController
  - 테스트용 생성자
  - 실제 사용 생성자
  - 시작 및 초기화
  - 청소 시작 요청
  - 청소 정지 요청
  - 강화 청소 모드 요청
  - 강화 청소 시간 확인
    - 만료 여부 확인

구현 (Coding)

CleaningPolicyController.cpp

```cpp
CleaningPolicyController::CleaningPolicyController() : state(StateType::NORMAL) {}

bool CleaningPolicyController::checkDust() {
    return dustSensor.requestStatus();
}

StateType CleaningPolicyController::checkingState() {
    return state;
}

void CleaningPolicyController::changeToBoost() {
    state = StateType::BOOST;
}

void CleaningPolicyController::changeToNormal() {
    state = StateType::NORMAL;
```

- CleaningPolicyController
  - 생성자
  - 먼지 감지
  - 현재 청소 모드 반환
  - 강화 청소 모드 변환
  - 일반 청소 모드 변환

MotorController.cpp

```cpp
#include "controller/MotorController.h"

MotorController::MotorController() : isRotating(false), motorPtr(&motor) {}

MotorController::MotorController(IMotor* m) : isRotating(false), motorPtr(m) {}

void MotorController::initialize() { isRotating = false; }

void MotorController::requestStopMoving() {
    motorPtr->stopMoving();
    isRotating = false;
}

void MotorController::move(DirectionType direction) {
    switch (direction) {
        case DirectionType::FORWARD:
            motorPtr->moveForward();
            break;
        case DirectionType::BACKWARD:
            motorPtr->moveBackward();
            break;
        case DirectionType::LEFT:
            motorPtr->turnLeft();
            break;
        case DirectionType::RIGHT:
            motorPtr->turnRight();
            break;
    }
    isRotating = (direction == DirectionType::LEFT || direction == DirectionType::RIGHT);
}
```

- MotorController
  - 테스트용 생성자
  - 실제 사용 생성자
  - 시작 및 초기화
  - 이동 정지 요청
  - 이동(매개변수: 방향)

구현 (Coding)

MovementPolicyController.cpp

```cpp
#include "controller/MovementPolicyController.h"

MovementPolicyController::MovementPolicyController()
    : frontSensor( u: std::make_unique<FrontSensor>()),
      leftSensor( u: std::make_unique<LeftSensor>()),
      rightSensor( u: std::make_unique<RightSensor>()) {}

MovementPolicyController::MovementPolicyController(
    std::unique_ptr<Sensor> front,
    std::unique_ptr<Sensor> left,
    std::unique_ptr<Sensor> right)
    : frontSensor( u: std::move( [>>] front)),
      leftSensor( u: std::move( [>>] left)),
      rightSensor( u: std::move( [>>] right)) {}

bool MovementPolicyController::checkObstacle() {
    return frontSensor->requestStatus();
}

DirectionType MovementPolicyController::checkMovementPolicy() {
    if (!leftSensor->requestStatus())  return DirectionType::LEFT;
    if (!rightSensor->requestStatus()) return DirectionType::RIGHT;
    return DirectionType::BACKWARD;
}
```

RVCPowerController.cpp

```cpp
#include "controller/RVCPowerController.h"

RVCPowerController::RVCPowerController() : isActiveState(fal

// UC1: 하드웨어 초기 상태 설정, isActive = true
void RVCPowerController::initialize() {
    isActiveState = true;
}

// UC8, UC9: 시스템을 안전한 종료 상태로 전환, isActive = false
void RVCPowerController::shutdown() {
    isActiveState = false;
}

bool RVCPowerController::isActive() const {
    return isActiveState;
}
```

- MovementPolicyController
  - 테스트용 생성자
  - 실제 사용 생성자
  - 전방 장애물 감지
  - 이동 정책 결정
- RVCPowerController
  - 생성자
  - 시작 및 초기화
  - 시스템 종료
  - 활성화 여부 반환

구현 (Coding)

CLIHandler.cpp

```cpp
#include "handler/CLIHandler.h"
#include "common/types.h"
#include <iostream>

// UC1, UC8: "[RVC] 시스템 준비 완료!" / "[RVC] 시스템 종료 중..." 등 출력
void CLIHandler::display(const std::string& message) {
    std::cout << "[RVC] " << message << std::endl;
}

// UC9: 에러 타입을 문자열로 변환해서 출력
void CLIHandler::errDisplay(const ErrorInfo& error) {
    std::cout << "[ERROR] " << errorTypeToString(error.getError()) << std::endl;
}
```

- CLIHandler
  - 일반 출력
  - 에러 출력
- ErrorHandler
  - 생성자
  - 에러 객체 생성 및 전달
- ErrorInfo
  - 생성자
  - 에러 반환

ErrorHandler.cpp

```cpp
ErrorHandler::ErrorHandler(IErrorNotifiable* notifiable)
    : notifiable(notifiable) {}

// UC9 진입점: 에러 정보 생성 후 notifyError 전달
void ErrorHandler::error(ErrorType type) {
    errorInfo = std::make_unique<ErrorInfo>(type);
    if (notifiable != nullptr) {
        notifiable->notifyError(*errorInfo);
    }
}
```

ErrorInfo.cpp

```cpp
#include "handler/ErrorInfo.h"

ErrorInfo::ErrorInfo(ErrorType error) : error(error) {}

std::string ErrorInfo::toString() const {
    switch (error) {
        case ErrorType::MOTOR_ERROR:    return "MOTOR_ERROR";
        case ErrorType::SENSOR_ERROR:   return "SENSOR_ERROR";
        case ErrorType::CLEANER_ERROR:  return "CLEANER_ERROR";
        case ErrorType::UNKNOWN_ERROR:  return "UNKNOWN_ERROR";
        default:                        return "NONE";
    }
}
```

구현 (Coding)

Cleaner.cpp

```cpp
#include "hardware/Cleaner.h"

void Cleaner::startCleaning() {
    // 기본 청소 모드 시작 (UC2)
    cleaningActive = true;
}

void Cleaner::stopCleaning() {
    // 기본 청소 모드 종료 (UC2)
    cleaningActive = false;
    boostPowerActive = false;
}

void Cleaner::powerUp() {
    // 강화 청소 모드 시작 (UC7)
    boostPowerActive = true;
}

void Cleaner::powerDown() {
    // 강화 청소 모드 종료 (UC7)
    boostPowerActive = false;
}
```

ICleaner.h

```cpp
class ICleaner {
public:
    virtual ~ICleaner() = default;

    virtual void startCleaning() = 0;
    virtual void stopCleaning() = 0;
    virtual void powerUp() = 0;
    virtual void powerDown() = 0;
};
```

- 주요 필드
  - 청소 활성화 여부
  - 강화 청소 모드 활성화 여부
- Cleaner
  - 기본청소모드 시작
  - 기본청소모드 종료
  - 강화청소모드 시작
  - 강화청소모드 종료

Cleaner.h

```cpp
#pragma once
#include "hardware/ICleaner.h"

class Cleaner : public ICleaner {
private:
    bool cleaningActive{false};
    bool boostPowerActive{false};

public:
    Cleaner() = default;

    void startCleaning() override;
    void stopCleaning() override;
    void powerUp() override;
    void powerDown() override;
};
```

구현 (Coding)

IMotor.h

```
class IMotor {
public:
    virtual ~IMotor() = default;

    virtual void moveForward() = 0;
    virtual void moveBackward() = 0;
    virtual void turnLeft() = 0;
    virtual void turnRight() = 0;
    virtual void stopMoving() = 0;
};
```

Motor.h

```
class Motor : public IMotor {
public:
    Motor() = default;

    void moveForward() override;
    void moveBackward() override;
    void turnLeft() override;
    void turnRight() override;
    void stopMoving() override;
};
```

구현 (Coding)

types.h

```h
#pragma once
#include <string>

enum class DirectionType {
    FORWARD,
    BACKWARD,
    LEFT,
    RIGHT
};

enum class StateType {
    NORMAL,
    BOOST
};

enum class ErrorType {
    NONE,
    MOTOR_ERROR,
    SENSOR_ERROR,
    CLEANER_ERROR,
    UNKNOWN_ERROR
};
```

- enum
  - 방향 (사방)
  - 청소 모드 (일반/강화)
  - 에러 종류 (없음/모터/센서/클리너/불명)

구현 (Coding)

CleanerController.h

```h
#pragma once
#include <chrono>
#include "hardware/Cleaner.h"

class CleanerController {
private:
    int boostDurationMs;
    bool boostRunning{false};
    std::chrono::steady_clock::time_point boostEndTime{};

    Cleaner cleaner;
    ICleaner* cleanerPtr;

    bool updateBoostTimerFromClock();

public:
    explicit CleanerController(int durationMs = 5 * 60 * 1000);
    explicit CleanerController(ICleaner* c);
    CleanerController(const CleanerController&) = delete;

    void initialize();
    void requestPowerUp();
    void requestStartCleaning();
    void requestStopCleaning();
    bool update();
};
```

- 주요 필드
  - 강화 청소 모드 지속시간
  - 강화 청소 모드 여부

CleaningPolicyController.h

```h
#pragma once
#include "common/types.h"
#include "hardware/DustSensor.h"

class CleaningPolicyController {
private:
    StateType state;
    DustSensor dustSensor;

public:
    CleaningPolicyController();

    bool checkDust();
    StateType checkingState();
    void changeToBoost();
    void changeToNormal();
};
```

- 주요 필드
  - 현재 상태
  - 먼지 센서 객체

MotorController.h

```h
#pragma once
#include "common/types.h"
#include "hardware/Motor.h"

class MotorController {
private:
    bool isRotating;
    Motor motor;
    IMotor* motorPtr;

public:
    MotorController();
    explicit MotorController(IMotor* m);
    MotorController(const MotorController&) = delete;

    void initialize();
    void requestStopMoving();
    void move(DirectionType direction);
};
```

- 주요 필드
  - 회전 여부
  - 모터, 모터 인터페이스 객체

구현 (Coding)

MovementPolicyController.h

```h
#pragma once
#include "common/types.h"
#include "hardware/Sensor.h"
#include <memory>

class MovementPolicyController {
private:
    std::unique_ptr<Sensor> frontSensor;
    std::unique_ptr<Sensor> leftSensor;
    std::unique_ptr<Sensor> rightSensor;

public:
    MovementPolicyController();
    MovementPolicyController(std::unique_ptr<Sensor> front,
                             std::unique_ptr<Sensor> left,
                             std::unique_ptr<Sensor> right);
    MovementPolicyController(const MovementPolicyController&) = delete;

    bool checkObstacle();
    DirectionType checkMovementPolicy();
};
```

- 주요 필드
  - 전방, 좌측, 우측 센서 포인터

RVCPowerController.h

```h
#pragma once

class RVCPowerController {
private:
    bool isActive;

public:
    RVCPowerController();
    // UC1: 시스템 초기화 - isActive = true
    void initialize();
    // UC8, UC9: 시스템 안전 종료 - isActive = false
    void shutdown();
    bool getIsActive() const;
};
```

- 주요 필드
  - 활성화 여부

구현 (Coding)

CLIHandler.h

```h
#pragma once
#include <string>
#include "handler/ErrorInfo.h"

class CLIHandler {
public:
    CLIHandler() = default;

    // UC1, UC8: 일반 상태 메시지 출력
    void display(const std::string& message);

    // UC9: 에러 정보 출력
    void errDisplay(const ErrorInfo& error);
};
```

ErrorHandler.h

```h
#pragma once
#include "handler/ErrorInfo.h"
#include "handler/IErrorNotifiable.h"
#include <memory>

class ErrorHandler {
private:
    std::unique_ptr<ErrorInfo> errorInfo;
    IErrorNotifiable* notifiable;  // RVCOrchestrator 대신 인터페이스에 의존

public:
    // 생성자 주입: 항상 유효한 상태로 시작 (nullptr 허용 - 테스트용)
    explicit ErrorHandler(IErrorNotifiable* notifiable);

    // UC1~UC8에서 예외 발생 시 호출 → UC9 흐름 시작
    void error(ErrorType type = ErrorType::UNKNOWN_ERROR);
};
```

- 주요 필드
  - 에러 객체
  - 에러 인터페이스

구현 (Coding)

ErrorInfo.h

```h
#pragma once
#include "common/types.h"

class ErrorInfo {
private:
    ErrorType error;

public:
    explicit ErrorInfo(ErrorType error = ErrorType::NONE);

    ErrorType getError() const;
};
```

- 주요 필드
  - 에러 종류 객체

Motor.cpp

```cpp
#include "hardware/Motor.h"

void Motor::moveForward() {}

void Motor::moveBackward() {}

void Motor::turnLeft() {}

void Motor::turnRight() {}

void Motor::stopMoving() {}
```

IErrorNotifiable.h

```h
#pragma once
#include "handler/ErrorInfo.h"

// ErrorHandler가 RVCOrchestrator에 직접 의존하지 않도록 분리한 인터페이스
class IErrorNotifiable {
public:
    virtual ~IErrorNotifiable() = default;
    virtual void notifyError(const ErrorInfo& error) = 0;
};
```

구현 (Coding)

Sensor.h 및 클래스 계층 구조

```cpp
#pragma once

class Sensor {
protected:
    bool isDetected;

public:
    Sensor();
    virtual ~Sensor() = default;

    virtual bool requestStatus();
};
```

```cpp
#pragma once
#include "Sensor.h"

class FrontSensor : public Sensor {
public:
    FrontSensor();
};
```

```cpp
#pragma once
#include "Sensor.h"

class DustSensor : public Sensor {
public:
    DustSensor();
};
```

```cpp
#include "hardware/LeftSensor.h"

LeftSensor::LeftSensor() {}
```

```cpp
#include "hardware/FrontSensor.h"

FrontSensor::FrontSensor() {}
```

```cpp
#include "hardware/DustSensor.h"

DustSensor::DustSensor() {}
```

```cpp
#include "hardware/RightSensor.h"

RightSensor::RightSensor() {}
```

Sensor.cpp

```cpp
#include "hardware/Sensor.h"

Sensor::Sensor() : isDetected(false) {}

bool Sensor::requestStatus() { return isDetected; }
```

구현 (Coding)

RVCOrchestrator.h

```
#pragma once
#include "handler/IErrorNotifiable.h"
#include "handler/ErrorInfo.h"
#include "handler/CLIHandler.h"
#include "handler/ErrorHandler.h"
#include "controller/RVCPowerController.h"
#include "controller/MotorController.h"
#include "controller/CleanerController.h"
#include "controller/MovementPolicyController.h"
#include "controller/CleaningPolicyController.h"

class RVCOrchestrator : public IErrorNotifiable {
private:
    CLIHandler& cliHandler;

    RVCPowerController* powerPtr;
    MotorController* motorPtr;
    CleanerController* cleanerPtr;
    MovementPolicyController* movementPtr;
    CleaningPolicyController* cleaningPolicyPtr;

    ErrorHandler errorHandler;

    bool systemRunning{false};
```

```
public:

    RVCOrchestrator(CLIHandler& cliHandler,
                    RVCPowerController* power,
                    MotorController* motor,
                    CleanerController* cleaner,
                    MovementPolicyController* movement,
                    CleaningPolicyController* cleaningPolicy);

    void powerOn();
    void performCleaning();
    void detectObstacle();
    void turnLeft();
    void turnRight();
    void backwardAndTurn();
    void performBoostCleaning();
    void powerOff();
    void notifyError(const ErrorInfo& error) override;

};
```

구현 (Coding)

RVCOrchestrator.cpp

```cpp
#include "RVCOrchestrator.h"

RVCOrchestrator::RVCOrchestrator(
    CLIHandler& cliHandler,
    RVCPowerController* power,
    MotorController* motor,
    CleanerController* cleaner,
    MovementPolicyController* movement,
    CleaningPolicyController* cleaningPolicy)
    : cliHandler(cliHandler),
      powerPtr(power),
      motorPtr(motor),
      cleanerPtr(cleaner),
      movementPtr(movement),
      cleaningPolicyPtr(cleaningPolicy),
      errorHandler( notifiable: this)
{}

// UC1: Power On System
void RVCOrchestrator::powerOn() {
    systemRunning = true;

    powerPtr->initialize();
    motorPtr->initialize();
    cleanerPtr->initialize();
    cliHandler.display( message: "시스템 준비 완료!");
}
```

```cpp
// UC2: Perform Cleaning
void RVCOrchestrator::performCleaning() {
    cleanerPtr->requestStartCleaning();
    motorPtr->move(DirectionType::FORWARD);

    while (systemRunning) {
        bool isDetected = movementPtr->checkObstacle();
        if (isDetected) {
            detectObstacle();
            motorPtr->move(DirectionType::FORWARD);
        }

        bool dustDetected = cleaningPolicyPtr->checkDust();
        if (dustDetected) {
            performBoostCleaning();
        }
    }
}

// UC3: Detect Obstacle
void RVCOrchestrator::detectObstacle() {
    cleanerPtr->requestStopCleaning();
    motorPtr->requestStopMoving();
    DirectionType dir = movementPtr->checkMovementPolicy();
    if (dir == DirectionType::LEFT)     { turnLeft();        return; }
    if (dir == DirectionType::RIGHT)    { turnRight();       return; }
    if (dir == DirectionType::BACKWARD) { backwardAndTurn(); return; }
}

// UC4: Turn Left
void RVCOrchestrator::turnLeft() {
    motorPtr->move(DirectionType::LEFT);
}
```

```cpp
// UC5: Turn Right
void RVCOrchestrator::turnRight() {
    motorPtr->move(DirectionType::RIGHT);
}

// UC6: Backward & Turn
void RVCOrchestrator::backwardAndTurn() {
    motorPtr->move(DirectionType::BACKWARD);
    DirectionType dir = movementPtr->checkMovementPolicy();
    if (dir == DirectionType::LEFT)  { turnLeft();  return; }
    if (dir == DirectionType::RIGHT) { turnRight(); return; }
    // all blocked → remain stopped (fail-safe)
}

//UC7: perform boost cleaning
void RVCOrchestrator::performBoostCleaning() {
    StateType state = cleaningPolicyPtr->checkingState();

    if (state == StateType::NORMAL) {
        cleaningPolicyPtr->changeToBoost();
        cleanerPtr->requestPowerUp();

        while (true) {
            bool isBoostExpired = cleanerPtr->update();

            if (isBoostExpired) {
                cleaningPolicyPtr->changeToNormal();
                break;
            }
        }
    }

}
```

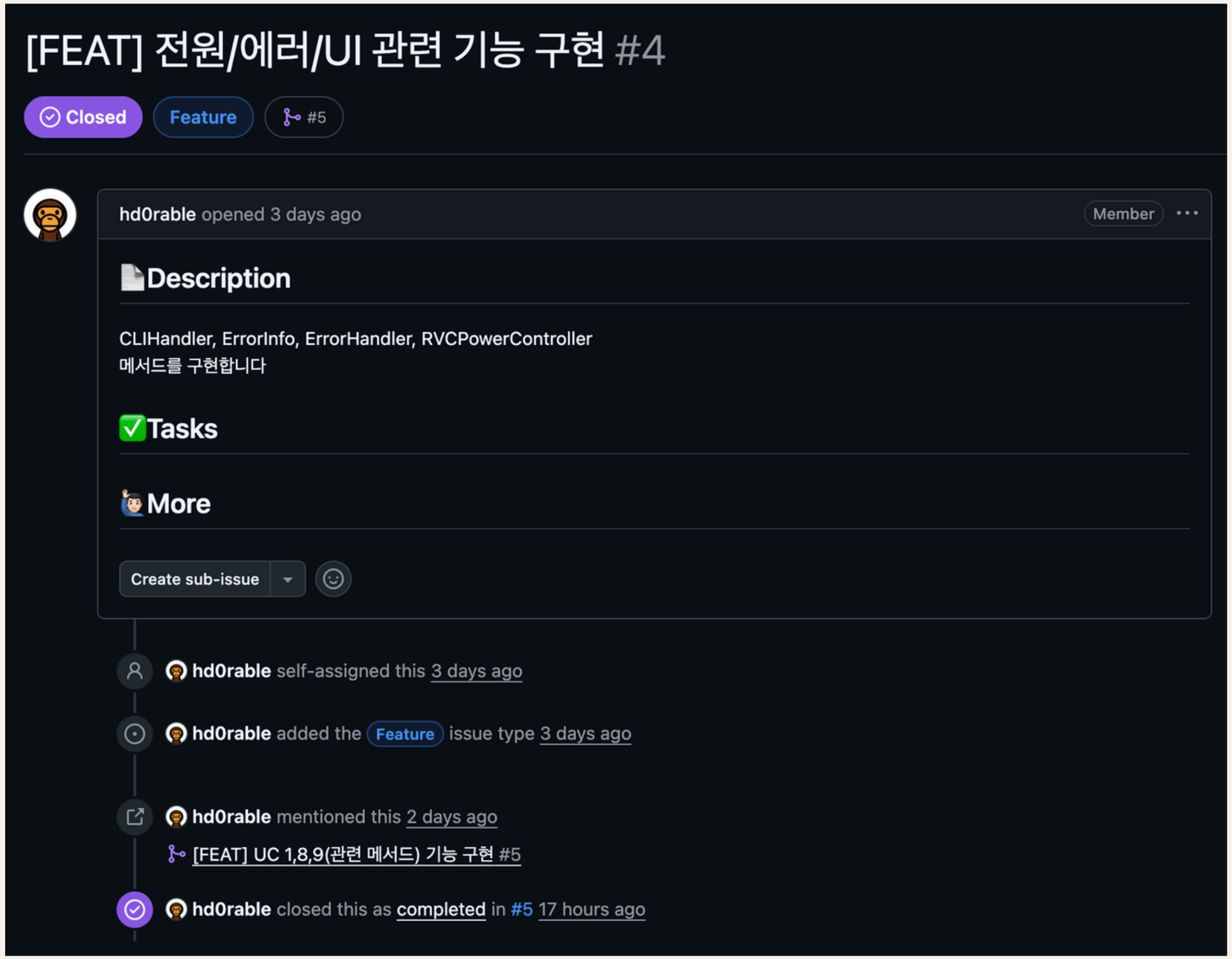
구현 (Coding)

RVCOrchestrator.cpp

```cpp
// UC8: Power Off System
void RVCOrchestrator::powerOff() {
    systemRunning = false;

    motorPtr->requestStopMoving();
    cleanerPtr->requestStopCleaning();
    cliHandler.display( message: "시스템 종료 중...");
    powerPtr->shutdown();
}

// UC9: Power Off System - Exceptional
void RVCOrchestrator::notifyError(const ErrorInfo& error) {
    systemRunning = false;

    cliHandler.errDisplay(error);
    motorPtr->requestStopMoving();
    cleanerPtr->requestStopCleaning();
    powerPtr->shutdown();
}
```

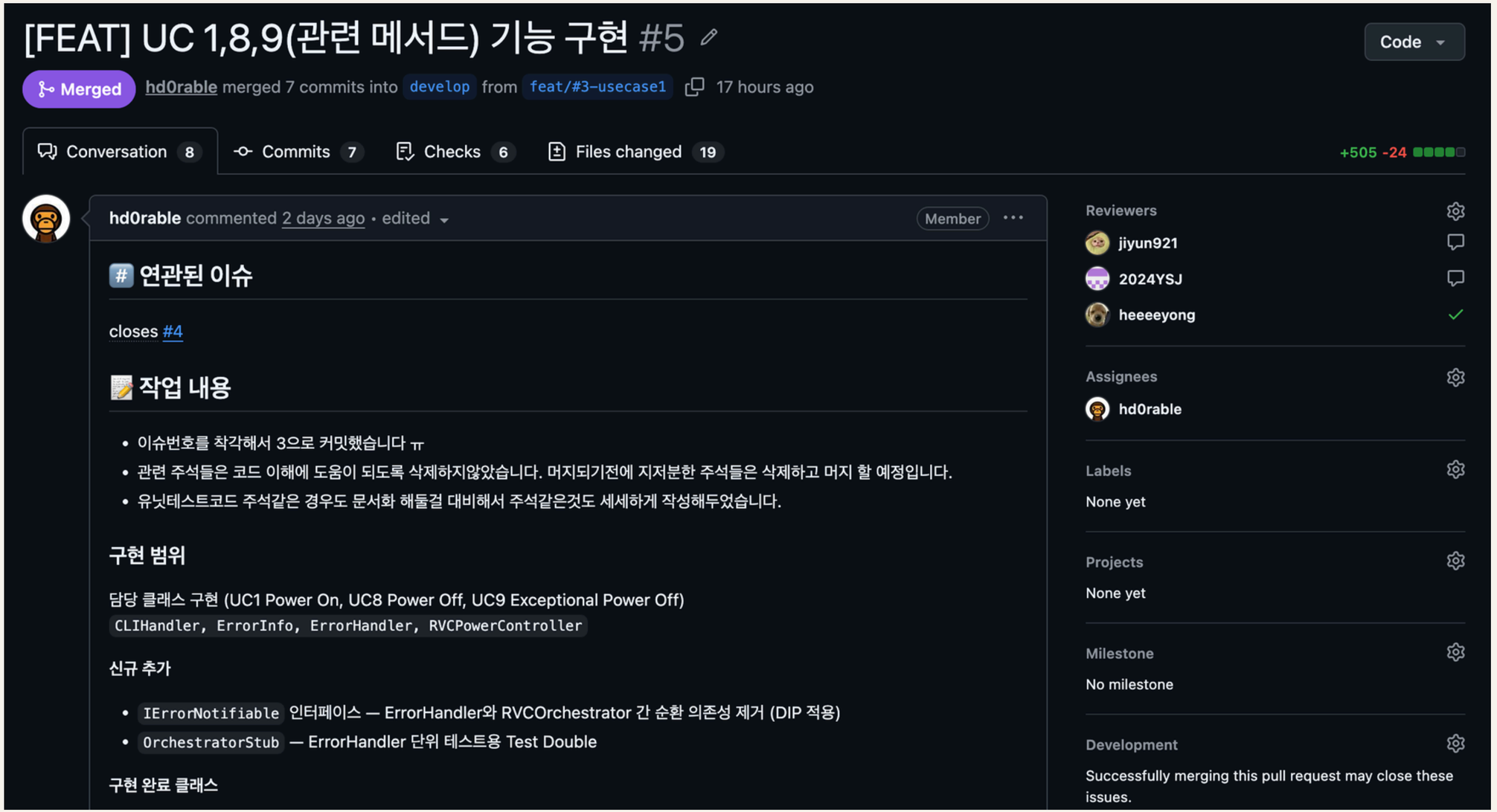
Code Review Process

1. 이슈 생성 및 구현 → [Local] Unit Test 수행



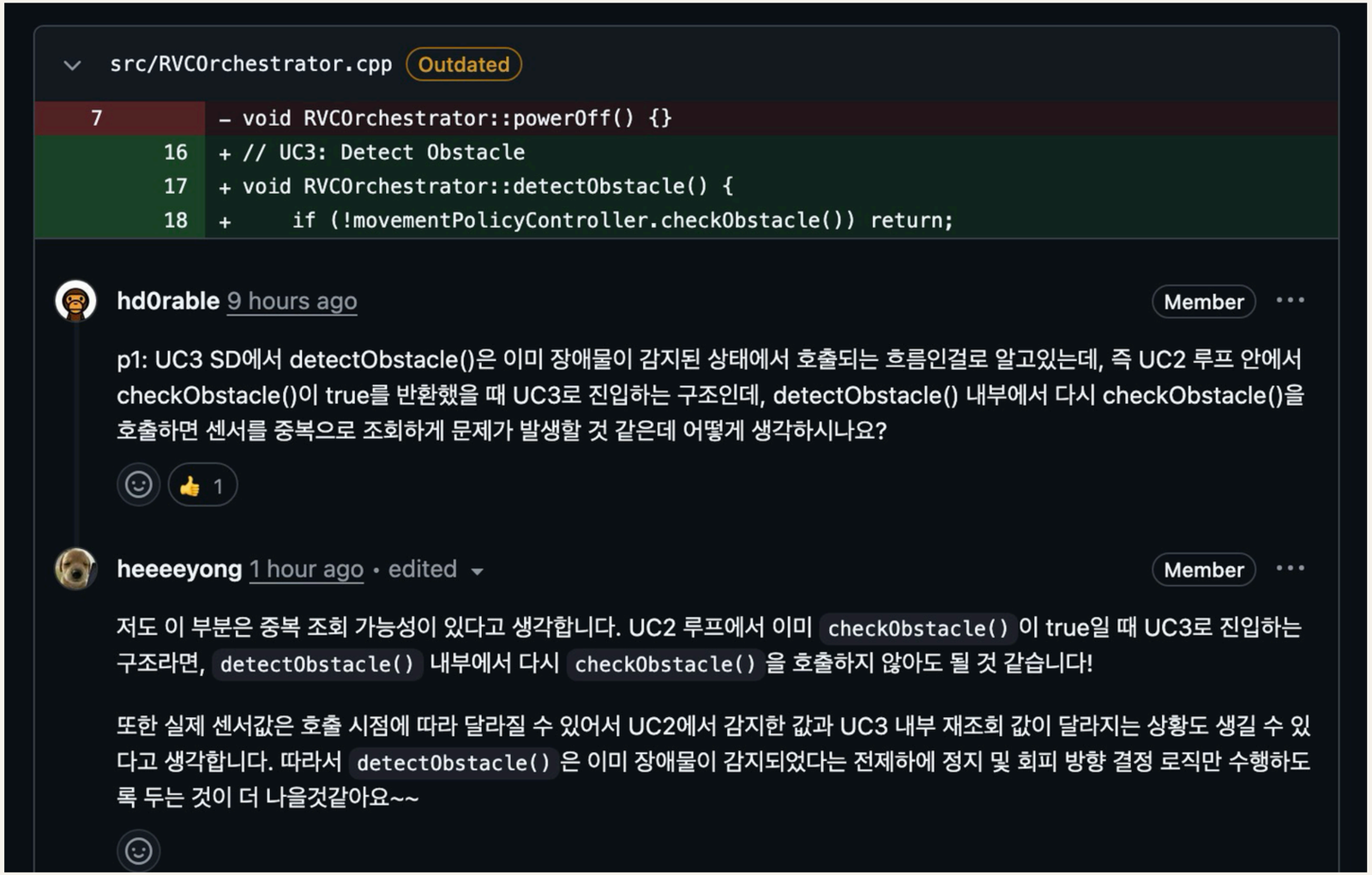
Code Review Process

2. Pull Request 생성



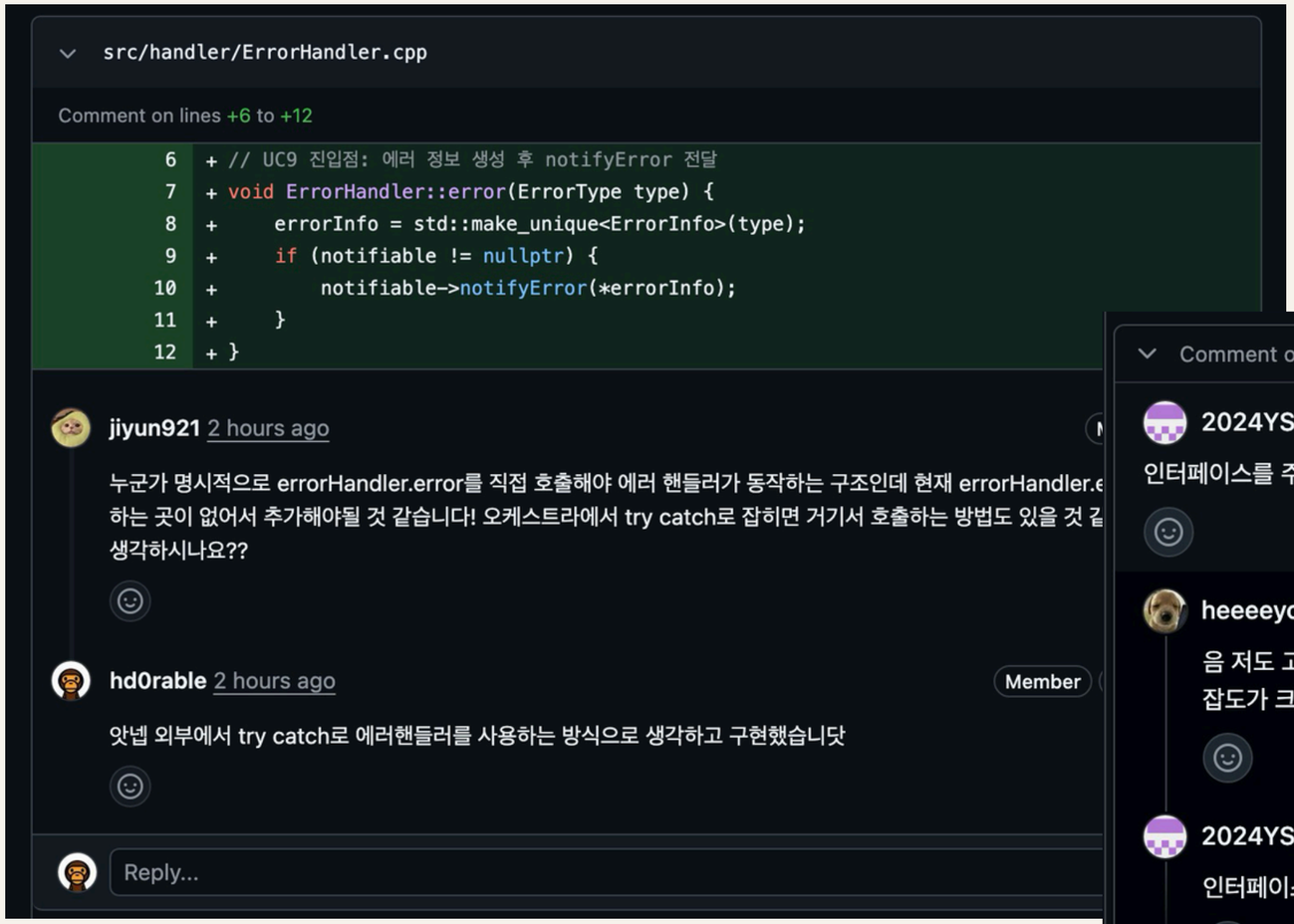
Code Review Process

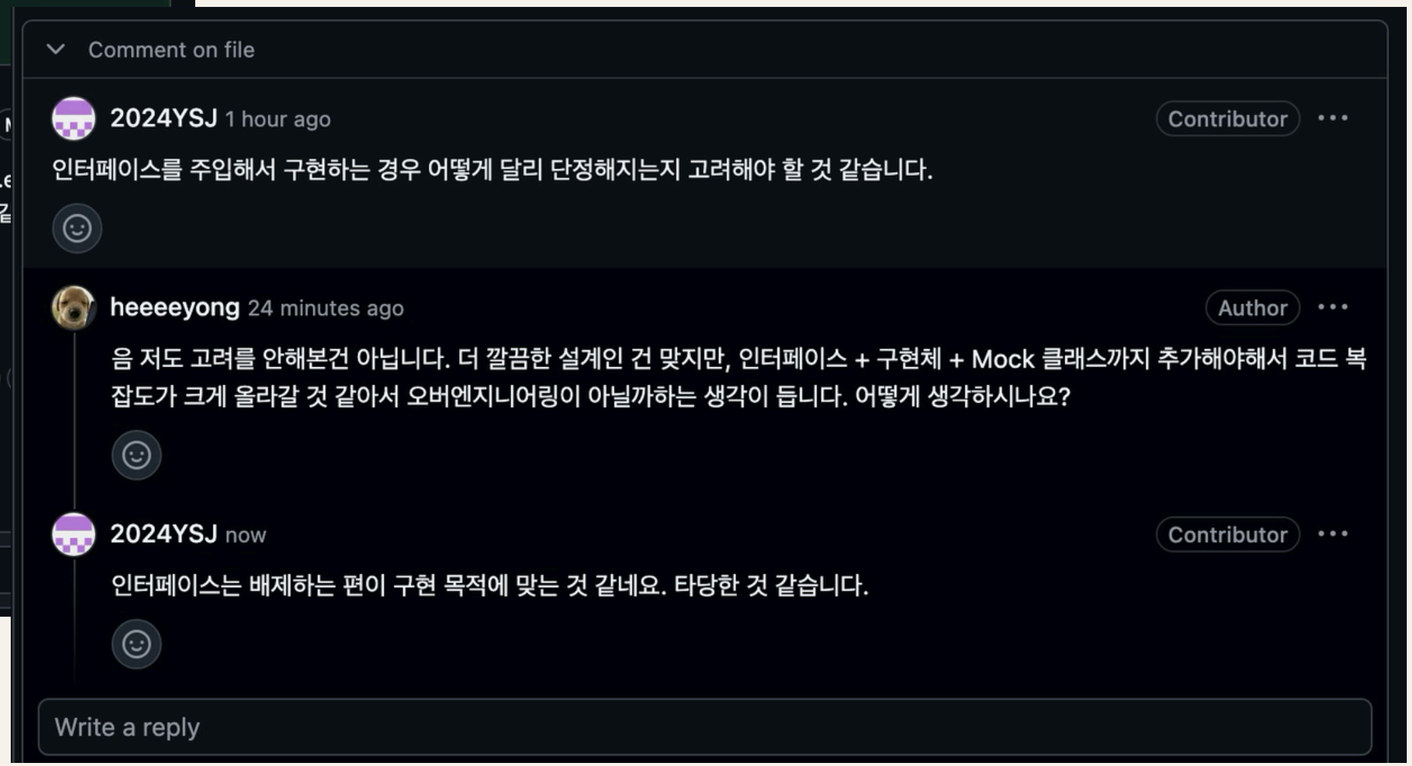
3. 팀원 전체 코드 리뷰 및 스레드 기반 논의



Code Review Process

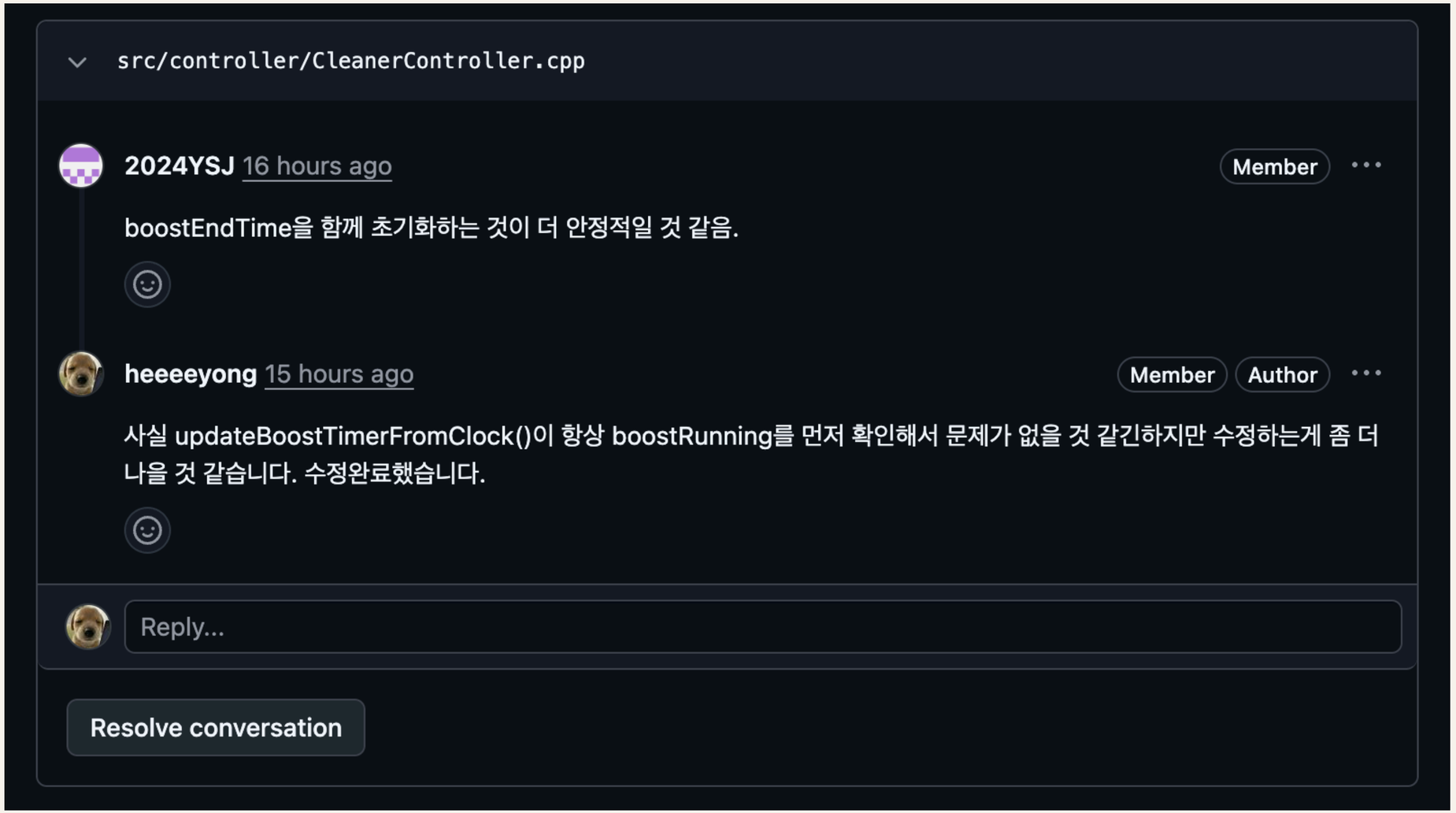
3. 팀원 전체 코드 리뷰 및 스레드 기반 논의





Code Review Process

4. 리팩토링 및 수정사항 Commit



Code Review Process

5. 재검토 및 승인 → Develop 브랜치 Merge

Unit Testing

RVCPowerControllerTest.cpp

TestNum	TestName	TestDescription	Pass/Fail
UT-01	InitialStateIsInactive	생성 직후 isActive가 false(비활성)인지 확인	Pass
UT-02	InitializeActivatesSystem	initialize() 호출 후 isActive가 true로 전환되는지 확인 (UC1)	Pass
UT-03	InitializeCalledTwiceRemainsActive	initialize()를 2회 연속 호출해도 active 상태가 유지되는지 확인	Pass
UT-04	InitializeAfterShutdownReactivates	shutdown 후 initialize() 재호출 시 active 상태로 복귀하는지 확인	Pass
UT-05	ShutdownDeactivatesSystem	initialize() 후 shutdown() 호출 시 isActive가 false로 전환되는지 확인 (UC8, UC9)	Pass
UT-06	ShutdownWithoutInitializeRemainsInactive	initialize() 없이 shutdown() 호출해도 크래시 없이 inactive 상태를 유지하는지 확인	Pass
UT-07	ShutdownCalledTwiceRemainsInactive	shutdown()을 2회 연속 호출해도 inactive 상태가 유지되는지 확인	Pass
UT-08	InitializeThenShutdownCycleUC1ToUC8	UC1 → UC8 순서로 initialize → shutdown 시 상태가 올바르게 전환되는지 확인	Pass
UT-09	MultipleOnOffCycles	initialize → shutdown 사이클을 반복해도 매 사이클마다 상태가 정확한지 확인	Pass

Unit Testing

CleanerControllerTest.cpp

TestNum	TestName	TestDescription	Pass/Fail
UT-01	RequestStartCleaningCallsHardware	requestStartCleaning() 호출 시 하드웨어 startCleaning이 호출되는지 확인	Pass
UT-02	StartCleaningDoesNotCallStop	requestStartCleaning() 호출 시 stopCleaning은 호출되지 않는지 확인	Pass
UT-03	StartCleaningDoesNotCallPowerUp	requestStartCleaning() 호출 시 powerUp은 호출되지 않는지 확인	Pass
UT-04	StartCleaningCallCountIsOne	requestStartCleaning() 1회 호출 시 callCount가 정확히 1인지 확인	Pass
UT-05	StartCleaningCalledTwiceCountIsTwo	requestStartCleaning() 2회 호출 시 callCount가 누적되어 2인지 확인	Pass
UT-06	RequestStopCleaningCallsHardware	requestStopCleaning() 호출 시 하드웨어 stopCleaning이 호출되는지 확인	Pass
UT-07	StopCleaningDoesNotCallStart	requestStopCleaning() 호출 시 startCleaning은 호출되지 않는지 확인	Pass
UT-08	StopCleaningDoesNotCallPowerUp	requestStopCleaning() 호출 시 powerUp은 호출되지 않는지 확인	Pass
UT-09	StopCleaningCallCountIsOne	requestStopCleaning() 1회 호출 시 callCount가 정확히 1인지 확인	Pass
UT-10	RequestPowerUpCallsHardware	requestPowerUp() 호출 시 하드웨어 powerUp이 호출되는지 확인	Pass

Unit Testing

CleanerControllerTest.cpp

UT-11	PowerUpDoesNotCallStart	requestPowerUp() 호출 시 startCleaning은 호출되지 않는지 확인	Pass
UT-12	PowerUpDoesNotCallStop	requestPowerUp() 호출 시 stopCleaning은 호출되지 않는지 확인	Pass
UT-13	PowerUpCallCountIsOne	requestPowerUp() 1회 호출 시 callCount가 정확히 1인지 확인	Pass
UT-14	InitializeDoesNotCallAnyHardware	initialize() 호출 시 하드웨어 메서드가 일체 호출되지 않는지 확인	Pass
UT-15	StopAfterStartBothCalled	start → stop 순서 호출 시 두 메서드 모두 호출되고 callCount가 2인지 확인	Pass
UT-16	StartAfterStopBothCalled	stop → start 순서 호출 시 두 메서드 모두 호출되고 callCount가 2인지 확인	Pass
UT-17	PowerUpThenStartBothCalled	powerUp → start 순서 호출 시 두 메서드 모두 호출되고 callCount가 2인지 확인	Pass
UT-18	StopCleaningCalledThreeTimesCountIsThree	requestStopCleaning() 3회 반복 호출 시 callCount가 3인지 확인	Pass
UT-19	StartThenResetThenStopOnlyStopCalled	reset() 이후 stop만 호출 시 start는 미호출이고 stop만 기록되는지 확인	Pass
UT-20	AllMethodsAccumulateCallCount	start / stop / powerUp 각각 1회 호출 시 callCount 합계가 3인지 확인	Pass

Unit Testing

MotorControllerTest.cpp
SD04: Turn Left

TestNum	TestName	TestDescription	Pass/Fail
UT-01	TurnLeftCallsTurnLeft	move(LEFT) 호출 시 하드웨어 turnLeft가 호출되는지 확인	Pass
UT-02	TurnLeftDoesNotCallMoveForward	move(LEFT) 호출 시 moveForward는 호출되지 않는지 확인	Pass
UT-03	TurnLeftLogHasOneEntry	move(LEFT) 1회 호출 시 callLog 크기가 1인지 확인	Pass
UT-04	TurnLeftLogFirstEntryIsTurnLeft	callLog 첫 번째 항목이 "turnLeft"인지 확인	Pass
UT-05	TurnLeftTotalCallCountIsOne	move(LEFT) 1회 호출 시 callCount가 1인지 확인	Pass
UT-06	TurnLeftDoesNotCallTurnRight	move(LEFT) 호출 시 turnRight는 호출되지 않는지 확인	Pass
UT-07	TurnLeftDoesNotCallMoveBackward	move(LEFT) 호출 시 moveBackward는 호출되지 않는지 확인	Pass
UT-08	TurnLeftDoesNotCallStopMoving	move(LEFT) 호출 시 stopMoving은 호출되지 않는지 확인	Pass
UT-09	TurnLeftCalledTwiceCallCountIsTwo	move(LEFT) 2회 호출 시 callCount가 2인지 확인	Pass
UT-10	StopMovingCallsHardwareStop	requestStopMoving() 호출 시 하드웨어 stop이 호출되는지 확인	Pass

Unit Testing

MotorControllerTest.cpp
SD04: Turn Left

UT-11	StopMovingDoesNotCallTurnLeft	requestStopMoving() 호출 시 turnLeft는 호출되지 않는지 확인	Pass
UT-12	StopMovingDoesNotCallMoveForward	requestStopMoving() 호출 시 moveForward는 호출되지 않는지 확인	Pass
UT-13	StopMovingCallCountIsOne	requestStopMoving() 1회 호출 시 callCount가 1인지 확인	Pass
UT-14	InitializeDoesNotCallAnyHardware	initialize() 호출 시 하드웨어 메서드가 호출되지 않는지 확인	Pass
UT-15	TurnLeftStateBecomesRotating	initialize() 후 move(LEFT) 호출 시 turnLeft가 호출되는지 확인	Pass
UT-16	TurnLeftAfterStopResetsLog	stop 후 reset, 다시 move(LEFT) 시 callLog 첫 항목이 "turnLeft"인지 확인	Pass
UT-17	TurnLeftOnlyCallsTurnLeft	move(LEFT) 호출 시 callLog 첫 항목이 오직 "turnLeft"인지 확인	Pass
UT-18	MultipleMovesAccumulateCallLog	move(LEFT) 2회 호출 시 callLog 크기가 2인지 확인	Pass
UT-19	TurnLeftRequestStatusNotCalledOnMotor	move(LEFT) 호출 시 backward/stop/right가 모두 미호출인지 확인	Pass
UT-20	ForwardMoveDoesNotCallTurnLeft	move(FORWARD) 호출 시 turnLeft는 호출되지 않는지 확인	Pass

Unit Testing

MotorControllerTest.cpp
SD05: Turn Right

TestNum	TestName	TestDescription	Pass/Fail
UT-01	TurnRightCallsTurnRight	move(RIGHT) 호출 시 하드웨어 turnRight가 호출되는지 확인	Pass
UT-02	TurnRightDoesNotCallMoveForward	move(RIGHT) 호출 시 moveForward는 호출되지 않는지 확인	Pass
UT-03	TurnRightLogHasOneEntry	move(RIGHT) 1회 호출 시 callLog 크기가 1인지 확인	Pass
UT-04	TurnRightLogFirstEntryIsTurnRight	callLog 첫 번째 항목이 "turnRight"인지 확인	Pass
UT-05	TurnRightTotalCallCountIsOne	move(RIGHT) 1회 호출 시 callCount가 1인지 확인	Pass
UT-06	TurnRightDoesNotCallTurnLeft	move(RIGHT) 호출 시 turnLeft는 호출되지 않는지 확인	Pass
UT-07	TurnRightDoesNotCallMoveBackward	move(RIGHT) 호출 시 moveBackward는 호출되지 않는지 확인	Pass
UT-08	TurnRightDoesNotCallStopMoving	move(RIGHT) 호출 시 stopMoving은 호출되지 않는지 확인	Pass
UT-09	TurnRightCalledTwiceCallCountIsTwo	move(RIGHT) 2회 호출 시 callCount가 2인지 확인	Pass
UT-10	StopMovingCallsHardwareStop	requestStopMoving() 호출 시 하드웨어 stop이 호출되는지 확인	Pass

Unit Testing

MotorControllerTest.cpp
SD05: Turn Right

UT-11	StopMovingDoesNotCallTurnRight	requestStopMoving() 호출 시 turnRight는 호출되지 않는지 확인	Pass
UT-12	StopMovingDoesNotCallMoveForward	requestStopMoving() 호출 시 moveForward는 호출되지 않는지 확인	Pass
UT-13	StopMovingCallCountIsOne	requestStopMoving() 1회 호출 시 callCount가 1인지 확인	Pass
UT-14	InitializeDoesNotCallAnyHardware	initialize() 호출 시 하드웨어 메서드가 호출되지 않는지 확인	Pass
UT-15	TurnRightStateBecomesRotating	initialize() 후 move(RIGHT) 호출 시 turnRight가 호출되는지 확인	Pass
UT-16	TurnRightAfterStopResetsLog	stop 후 reset, 다시 move(RIGHT) 시 callLog 첫 항목이 "turnRight"인지 확인	Pass
UT-17	TurnRightOnlyCallsTurnRight	move(RIGHT) 호출 시 callLog 첫 항목이 오직 "turnRight"인지 확인	Pass
UT-18	MultipleMovesAccumulateCallLog	move(RIGHT) 2회 호출 시 callLog 크기가 2인지 확인	Pass
UT-19	ForwardMoveDoesNotCallTurnRight	move(FORWARD) 호출 시 turnRight는 호출되지 않는지 확인	Pass
UT-20	LeftMoveDoesNotCallTurnRight	move(LEFT) 호출 시 turnRight는 호출되지 않는지 확인	Pass

Unit Testing

MotorControllerTest.cpp
SD06: Backward

TestNum	TestName	TestDescription	Pass/Fail
UT-01	MoveBackwardCallsHardwareBackward	move(BACKWARD) 호출 시 하드웨어 backward가 호출되는지 확인	Pass
UT-02	MoveBackwardCallCountIsOne	move(BACKWARD) 1회 호출 시 callCount가 1인지 확인	Pass
UT-03	MoveBackwardLogHasOneEntry	move(BACKWARD) 시 callLog 크기가 1이고 항목이 "backward"인지 확인	Pass
UT-04	BackwardCalledBeforeAnyTurn	callLog 첫 항목이 "backward"인지 확인	Pass
UT-05	BackwardThenLeftSequenceIsCorrect	backward → reset → move(LEFT) 시 callLog 첫 항목이 "turnLeft"인지 확인	Pass
UT-06	MoveBackwardDoesNotCallMoveForward	move(BACKWARD) 호출 시 moveForward는 호출되지 않는지 확인	Pass
UT-07	MoveBackwardDoesNotCallTurnLeft	move(BACKWARD) 호출 시 turnLeft는 호출되지 않는지 확인	Pass
UT-08	MoveBackwardDoesNotCallTurnRight	move(BACKWARD) 호출 시 turnRight는 호출되지 않는지 확인	Pass
UT-09	MoveBackwardDoesNotCallStopMoving	move(BACKWARD) 호출 시 stopMoving은 호출되지 않는지 확인	Pass
UT-10	BackwardCalledTwiceCallCountIsTwo	move(BACKWARD) 2회 호출 시 callCount가 2인지 확인	Pass

Unit Testing

MotorControllerTest.cpp
SD06: Backward

UT-11	BackwardAfterTurnRightDoesNotAccumulateTurn	turnRight → reset → backward 시 turnRight가 미호출인지 확인	Pass
UT-12	BackwardAfterTurnLeftDoesNotAccumulateTurn	turnLeft → reset → backward 시 turnLeft가 미호출인지 확인	Pass
UT-13	ForwardMoveDoesNotCallBackward	move(FORWARD) 호출 시 backward는 호출되지 않는지 확인	Pass
UT-14	LeftMoveDoesNotCallBackward	move(LEFT) 호출 시 backward는 호출되지 않는지 확인	Pass
UT-15	RightMoveDoesNotCallBackward	move(RIGHT) 호출 시 backward는 호출되지 않는지 확인	Pass
UT-16	StopMovingAfterBackwardWorks	backward 후 requestStopMoving() 호출 시 stop만 1회 기록되는지 확인	Pass
UT-17	BackwardThenRightSequenceIsCorrect	backward → reset → move(RIGHT) 시 callLog 첫 항목이 "turnRight"인지 확인	Pass
UT-18	InitializeBeforeBackwardDoesNotCallHardware	initialize() 후 callCount 0, move(BACKWARD) 후 1인지 확인	Pass
UT-19	BackwardThreeTimesCallCountIsThree	move(BACKWARD) 3회 호출 시 callCount가 3인지 확인	Pass

Unit Testing

MovementPolicyControllerTest.cpp

TestNum	TestName	TestDescription	Pass/Fail
UT-01	NoObstacleReturnsFalse	전방 센서 미감지 시 checkObstacle()이 false를 반환하는지 확인	Pass
UT-02	ObstacleDetectedReturnsTrue	전방 센서 감지 시 checkObstacle()이 true를 반환하는지 확인	Pass
UT-03	CheckObstacleCallsFrontSensorOnce	checkObstacle() 1회 호출 시 frontSensor callCount가 1인지 확인	Pass
UT-04	CheckObstacleDoesNotCallLeftSensor	checkObstacle() 호출 시 leftSensor는 호출되지 않는지 확인	Pass
UT-05	CheckObstacleDoesNotCallRightSensor	checkObstacle() 호출 시 rightSensor는 호출되지 않는지 확인	Pass
UT-06	LeftFreeReturnsLeft	left 미감지 시 checkMovementPolicy()가 LEFT를 반환하는지 확인	Pass
UT-07	LeftBlockedRightFreeReturnsRight	left 감지 + right 미감지 시 RIGHT를 반환하는지 확인	Pass

Unit Testing

MovementPolicyControllerTest.cpp

UT-08	BothBlockedReturnsBackward	left/right 모두 감지 시 BACKWARD를 반환하는지 확인	Pass
UT-09	LeftFreeShortCircuitsRightSensor	left 미감지 시 rightSensor를 호출하지 않는지 확인 (short-circuit)	Pass
UT-10	RightCalledOnlyWhenLeftBlocked	left 감지 시에만 rightSensor가 호출되는지 확인	Pass
UT-11	CheckObstacleCalledTwiceCountIsTwo	checkObstacle() 2회 호출 시 frontSensor callCount가 2인지 확인	Pass
UT-12	PolicyCalledIndependentlyAfterReset	reset 후 센서 상태 변경 시 새 상태로 올바른 방향을 반환하는지 확인	Pass
UT-13	PolicyReturnedIsExactlyLeft	LEFT 반환 시 RIGHT, BACKWARD와 다름을 확인	Pass
UT-14	PolicyReturnedIsExactlyRight	RIGHT 반환 시 LEFT, BACKWARD와 다름을 확인	Pass
UT-15	PolicyReturnedIsExactlyBackward	BACKWARD 반환 시 LEFT, RIGHT와 다름을 확인	Pass

Unit Testing

MovementPolicyControllerTest.cpp

UT-16	NoObstacleCheckObstacleStillReturnsCorrectly	장애물 없을 때 front callCount 1, left callCount 0인지 확인	Pass
UT-17	ObstacleLeadsToPolicyLeftPath	전방 장애물 + left 미감지 시 policy가 LEFT를 반환하는지 확인	Pass
UT-18	ObstacleLeadsToPolicyRightPath	전방 장애물 + left 감지 + right 미감지 시 RIGHT를 반환하는지 확인	Pass
UT-19	ObstacleLeadsToPolicyBackwardPath	전방 장애물 + left/right 모두 감지 시 BACKWARD를 반환하는지 확인	Pass
UT-20	CheckMovementPolicyCalledThreeTimesCallsLeftThreeTimes	checkMovementPolicy() 3회 호출 시 leftSensor callCount가 3인지 확인	Pass

Unit Testing

CleaningPolicyControllerTest.cpp

TestNum	TestName	TestDescription	Pass/Fail
UT-01	InitialState_IsNormal	초기 상태가 NORMAL인지 확인 (UC7)	Pass
UT-02	ChangeToBoost_SetsBoostState	changeToBoost() 호출 후 checkingState()가 BOOST인지 확인 (UC7)	Pass
UT-03	ChangeToNormal_AfterBoost_SetsNormalState	BOOST 상태에서 changeToNormal() 호출 후 NORMAL로 전환되는지 확인	Pass
UT-04	CheckingState_MultipleCallsDoNotModifyState_WhenNormal	NORMAL 상태에서 checkingState() 반복 호출 시 상태가 변하지 않는지 확인	Pass
UT-05	CheckingState_MultipleCallsDoNotModifyState_WhenBoost	BOOST 상태에서 checkingState() 반복 호출 시 상태가 변하지 않는지 확인	Pass
UT-06	ChangeToNormal_WhenAlreadyNormal_IsIdempotent	이미 NORMAL 상태에서 changeToNormal() 호출해도 NORMAL을 유지하는지 확인	Pass
UT-07	ChangeToBoost_Twice_IsIdempotent	BOOST 상태에서 changeToBoost() 재호출 시 BOOST를 유지하는지 확인	Pass
UT-08	ChangeToNormal_Multiple_IsIdempotent	changeToNormal() 연속 호출해도 NORMAL을 유지하는지 확인	Pass
UT-09	StateTransition_NormalBoostNormalBoost_IsCorrect	NORMAL → BOOST → NORMAL → BOOST 상태 전이가 올바르게 동작하는지 확인	Pass

Unit Testing

CleanerTest.cpp

TestNum	TestName	TestDescription	Pass/Fail
UT-01	StartCleaning_ActivatesCleaning	startCleaning() 호출 시 cleaningActive가 true인지 확인 (UC2)	Pass
UT-02	PowerUp_ActivatesBoost	powerUp() 호출 시 boostActive가 true인지 확인 (UC7)	Pass
UT-03	StopCleaning_WithoutStart_ShouldNotCrash	startCleaning 없이 stopCleaning() 호출 시 크래시가 발생하지 않는지 확인	Pass
UT-04	StopCleaning_AfterStart_ClearsCleaningActive	stopCleaning() 호출 후 cleaningActive가 false인지 확인	Pass
UT-05	StopCleaning_AlsoClearsBoostActive	stopCleaning() 호출 시 boostActive도 함께 해제되는지 확인 (UC7)	Pass
UT-06	PowerDown_WithoutPowerUp_ShouldNotCrash	powerUp 없이 powerDown() 호출 시 크래시가 발생하지 않는지 확인	Pass
UT-07	StartCleaning_Twice_IsIdempotent	startCleaning() 2회 호출해도 cleaningActive가 유지되는지 확인	Pass
UT-08	StopCleaning_Twice_IsIdempotent	stopCleaning() 2회 호출해도 크래시 없고 cleaningActive가 false인지 확인	Pass
UT-09	PowerUp_DoesNotActivateCleaning	powerUp() 호출 시 cleaningActive는 활성화되지 않는지 확인	Pass
UT-10	PowerDown_AfterPowerUp_ClearsBoostActive	powerUp() 후 powerDown() 호출 시 boostActive가 false인지 확인	Pass

Unit Testing

CLIHandlerTest.cpp

TestNum	TestName	TestDescription	Pass/Fail
UT-01	DisplayIncludesRVCPrefix	출력 결과에 "[RVC]" 접두사가 포함되는지 확인	Pass
UT-02	DisplayIncludesGivenMessage	출력 결과에 전달한 메시지 내용이 포함되는지 확인	Pass
UT-03	DisplayExactFormatUC1	UC1 시나리오: 전원 ON 후 출력 형식이 정확한지 확인	Pass
UT-04	DisplayExactFormatUC8	UC8 시나리오: 정상 종료 시 출력 형식이 정확한지 확인	Pass
UT-05	DisplayEmptyMessageStillHasPrefix	빈 문자열 전달 시에도 "[RVC] " 접두사가 출력되는지 확인	Pass
UT-06	DisplayCalledTwiceOutputsBothLines	display()를 연속 두 번 호출했을 때 두 메시지가 모두 출력되는지 확인	Pass
UT-07	DisplaySpecialCharacters	특수문자가 포함된 메시지도 그대로 출력되는지 확인	Pass
UT-08	ErrDisplayIncludesErrorPrefix	에러 출력 결과에 "[ERROR]" 접두사가 포함되는지 확인	Pass
UT-09	ErrDisplayMotorError	MOTOR_ERROR 출력 형식이 정확한지 확인	Pass
UT-10	ErrDisplaySensorError	SENSOR_ERROR 출력 형식이 정확한지 확인	Pass
UT-11	ErrDisplayCleanerError	CLEANER_ERROR 출력 형식이 정확한지 확인	Pass
UT-12	ErrDisplayUnknownError	UNKNOWN_ERROR 출력 형식이 정확한지 확인	Pass
UT-13	ErrDisplayNoneError	NONE 타입 에러도 정상 출력되는지 확인 (Negative)	Pass

Unit Testing

ErrorHandlerTest.cpp

TestNum	TestName	TestDescription	Pass/Fail
UT-01	ConstructWithNullDoesNotCrash	nullptr 전달 시 생성자가 예외 없이 정상 동작하는지 확인	Pass
UT-02	ConstructWithStubDoesNotCrash	Stub 전달 시 생성자가 예외 없이 정상 동작하는지 확인	Pass
UT-03	ErrorWithNullNotifiableDoesNotCrash	notifiable이 nullptr일 때 error() 호출 시 크래시 없이 안전하게 처리되는지 확인	Pass
UT-04	ErrorWithNullDoesNotCallNotify	notifiable이 nullptr일 때 notifyError가 호출되지 않는지 확인	Pass
UT-05	ErrorCallsNotifyOnStub	error() 호출 시 Stub의 notifyError가 실제로 호출되는지 확인	Pass
UT-06	ErrorPassesMotorErrorType	error() 호출 시 MOTOR_ERROR 타입이 정확하게 전달되는지 확인	Pass
UT-07	ErrorPassesSensorErrorType	error() 호출 시 SENSOR_ERROR 타입이 정확하게 전달되는지 확인	Pass
UT-08	ErrorPassesCleanerErrorType	error() 호출 시 CLEANER_ERROR 타입이 정확하게 전달되는지 확인	Pass
UT-09	ErrorDefaultTypeIsUnknown	인자 없이 error() 호출 시 기본값 UNKNOWN_ERROR가 전달되는지 확인	Pass
UT-10	ErrorCalledOnceNotifyCalledOnce	error() 1회 호출 시 notifyError도 정확히 1회 호출되는지 확인	Pass
UT-11	ErrorCalledMultipleTimesNotifyCalledMultipleTimes	error() 2회 호출 시 notifyError도 2회 호출되는지 확인	Pass
UT-12	ErrorUpdatesLastErrorTypeOnRepeatCall	error() 반복 호출 시 마지막 에러 타입으로 갱신되는지 확인	Pass

Unit Testing

ErrorInfoTest.cpp

TestNum	TestName	TestDescription	Pass/Fail
UT-01	DefaultConstructorSetsNone	인자 없이 생성 시 기본값이 NONE인지 확인	Pass
UT-02	ConstructWithMotorError	MOTOR_ERROR로 생성 시 해당 타입이 저장되는지 확인	Pass
UT-03	ConstructWithSensorError	SENSOR_ERROR로 생성 시 해당 타입이 저장되는지 확인	Pass
UT-04	ConstructWithCleanerError	CLEANER_ERROR로 생성 시 해당 타입이 저장되는지 확인	Pass
UT-05	ConstructWithUnknownError	UNKNOWN_ERROR로 생성 시 해당 타입이 저장되는지 확인	Pass
UT-06	GetErrorReturnsExactTypeMotor	getError()가 지정한 타입만 반환하고 다른 타입과 구별되는지 확인	Pass
UT-07	GetErrorIsIdempotent	getError()를 여러 번 호출해도 동일한 값을 반환하는지 확인 (멱등성)	Pass
UT-08	TwoInstancesAreIndependent	서로 다른 타입으로 생성된 두 인스턴스가 독립적인지 확인	Pass

Unit Testing

CleanerTest.cpp

TestNum	TestName	TestDescription	Pass/Fail
UT-01	StartCleaning_ActivatesCleaning	startCleaning() 호출 시 cleaningActive가 true인지 확인 (UC2)	Pass
UT-02	PowerUp_ActivatesBoost	powerUp() 호출 시 boostActive가 true인지 확인 (UC7)	Pass
UT-03	StopCleaning_WithoutStart_ShouldNotCrash	startCleaning 없이 stopCleaning() 호출 시 크래시가 발생하지 않는지 확인	Pass
UT-04	StopCleaning_AfterStart_ClearsCleaningActive	stopCleaning() 호출 후 cleaningActive가 false인지 확인	Pass
UT-05	StopCleaning_AlsoClearsBoostActive	stopCleaning() 호출 시 boostActive도 함께 해제되는지 확인 (UC7)	Pass
UT-06	PowerDown_WithoutPowerUp_ShouldNotCrash	powerUp 없이 powerDown() 호출 시 크래시가 발생하지 않는지 확인	Pass
UT-07	StartCleaning_Twice_IsIdempotent	startCleaning() 2회 호출해도 cleaningActive가 유지되는지 확인	Pass
UT-08	StopCleaning_Twice_IsIdempotent	stopCleaning() 2회 호출해도 크래시 없고 cleaningActive가 false인지 확인	Pass
UT-09	PowerUp_DoesNotActivateCleaning	powerUp() 호출 시 cleaningActive는 활성화되지 않는지 확인	Pass
UT-10	PowerDown_AfterPowerUp_ClearsBoostActive	powerUp() 후 powerDown() 호출 시 boostActive가 false인지 확인	Pass

Unit Testing

SensorTest.cpp

TestNum	TestName	TestDescription	Pass/Fail
UT-01	Constructor_DefaultIsDetectedFalse	생성 직후 isDetected 기본값이 false인지 확인	Pass
UT-02	RequestStatus_ReturnsCurrentState	requestStatus()가 현재 상태를 반환하는지 확인	Pass
UT-03	RequestStatus_DoesNotModifyState	requestStatus() 호출 후 상태가 변경되지 않는지 확인	Pass
UT-04	GetIsDetected_ReturnsCurrentState	getIsDetected()가 현재 상태를 반환하는지 확인	Pass

Unit Testing

RVCOrchestratorTest.cpp

UC1: powerOn()

TestNum	TestName	TestDescription	Pass/Fail
UT-01	UC1_PowerOn_ExactOutputFormat	powerOn() 출력 형식이 "[RVC] 시스템 준비 완료!\n"인지 확인	Pass
UT-02	UC1_PowerOn_MotorHardwareNotCalled	powerOn() 시 motor 하드웨어가 호출되지 않는지 확인	Pass
UT-03	UC1_PowerOn_CleanerHardwareNotCalled	powerOn() 시 cleaner 하드웨어가 호출되지 않는지 확인	Pass
UT-04	UC1_PowerOn_NoErrorOutput	powerOn() 시 에러 출력이 발생하지 않는지 확인	Pass

UC2: performCleaning()

TestNum	TestName	TestDescription	Pass/Fail
UT-05	UC2_PerformCleaning_StartsCleaningAndMovesForward	performCleaning() 호출 시 startCleaning과 moveForward가 호출되는지 확인	Pass
UT-06	UC2_PerformCleaning_DoesNotImmediatelyStopMotor	performCleaning() 호출 시 stopMoving이 즉시 호출되지 않는지 확인	Pass
UT-07	UC2_PerformCleaning_DoesNotImmediatelyStopCleaning	performCleaning() 호출 시 stopCleaning이 즉시 호출되지 않는지 확인	Pass

Unit Testing

RVCOrchestratorTest.cpp

UC3: detectObstacle()

TestNum	TestName	TestDescription	Pass/Fail
UT-08	UC3_DetectObstacle_LeftFree_StopsAndTurnsLeft	왼쪽 비어 있을 때 stopCleaning·stopMoving·turnLeft가 모두 호출되는지 확인	Pass
UT-09	UC3_DetectObstacle_LeftBlocked_TurnsRightNotLeft	왼쪽 막힘·오른쪽 비어 있을 때 turnRight가 호출되고 turnLeft는 미호출인지 확인	Pass
UT-10	UC3_DetectObstacle_BothBlocked_BackwardOnlyNoTurn	양쪽 모두 막혔을 때 backward만 호출되고 turn은 호출되지 않는지 확인	Pass
UT-11	UC3_DetectObstacle_ProducesNoDisplayOutput	detectObstacle() 호출 시 display 출력이 없는지 확인	Pass

UC4: turnLeft()

TestNum	TestName	TestDescription	Pass/Fail
UT-12	UC4_TurnLeft_CallsMotorTurnLeft	turnLeft() 호출 시 motor.turnLeft가 호출되는지 확인	Pass
UT-13	UC4_TurnLeft_DoesNotCallTurnRight	turnLeft() 호출 시 turnRight는 호출되지 않는지 확인	Pass
UT-14	UC4_TurnLeft_DoesNotCallBackward	turnLeft() 호출 시 backward는 호출되지 않는지 확인	Pass
UT-15	UC4_TurnLeft_DoesNotCallStopMoving	turnLeft() 호출 시 stopMoving은 호출되지 않는지 확인	Pass

Unit Testing

RVCOrchestratorTest.cpp

UC5: turnRight()

TestNum	TestName	TestDescription	Pass/Fail
UT-16	UC5_TurnRight_CallsMotorTurnRight	turnRight() 호출 시 motor.turnRight가 호출되는지 확인	Pass
UT-17	UC5_TurnRight_DoesNotCallTurnLeft	turnRight() 호출 시 turnLeft는 호출되지 않는지 확인	Pass
UT-18	UC5_TurnRight_DoesNotCallBackward	turnRight() 호출 시 backward는 호출되지 않는지 확인	Pass
UT-19	UC5_TurnRight_DoesNotCallStopMoving	turnRight() 호출 시 stopMoving은 호출되지 않는지 확인	Pass

UC7: performBoostCleaning()

TestNum	TestName	TestDescription	Pass/Fail
UT-25	UC7_PerformBoostCleaning_NormalState_CallsPowerUp	NORMAL 상태에서 performBoostCleaning() 호출 시 powerUp이 호출되는지 확인	Pass
UT-26	UC7_PerformBoostCleaning_DoesNotCallStopCleaning	performBoostCleaning() 호출 시 stopCleaning이 호출되지 않는지 확인	Pass
UT-27	UC7_PerformBoostCleaning_DoesNotMoveMotor	performBoostCleaning() 호출 시 motor 동작이 발생하지 않는지 확인	Pass

Unit Testing

RVCOrchestratorTest.cpp

UC6: backwardAndTurn()

TestNum	TestName	TestDescription	Pass/Fail
UT-20	UC6_BackwardAndTurn_LeftFree_BackwardThenTurnLeft	왼쪽 비어 있을 때 backward·turnLeft 순서로 호출되고 callCount가 2인지 확인	Pass
UT-21	UC6_BackwardAndTurn_LeftBlocked_CallsBackwardThenTurnRight	왼쪽 막힘·오른쪽 비어 있을 때 backward·turnRight가 호출되는지 확인	Pass
UT-22	UC6_BackwardAndTurn_BothBlocked_OnlyBackwardCalled	양쪽 모두 막혔을 때 backward만 호출되고 turn은 미호출인지 확인 (fail-safe)	Pass
UT-23	UC6_BackwardAndTurn_CleanerNotCalled	backwardAndTurn() 호출 시 cleaner가 호출되지 않는지 확인	Pass
UT-24	UC6_BackwardAndTurn_ProducesNoDisplayOutput	backwardAndTurn() 호출 시 display 출력이 없는지 확인	Pass

Unit Testing

RVCOrchestratorTest.cpp

UC8: powerOff()

TestNum	TestName	TestDescription	Pass/Fail
UT-28	UC8_PowerOff_ExactOutputFormat	powerOff() 출력 형식이 "[RVC] 시스템 종료 중...\n"인지 확인	Pass
UT-29	UC8_PowerOff_CallsStopMoving	powerOff() 호출 시 motor.stopMoving이 호출되는지 확인	Pass
UT-30	UC8_PowerOff_CallsStopCleaning	powerOff() 호출 시 cleaner.stopCleaning이 호출되는지 확인	Pass
UT-31	UC8_PowerOff_DoesNotCallStartCleaning	powerOff() 호출 시 startCleaning은 호출되지 않는지 확인	Pass

UC9: notifyError()

TestNum	TestName	TestDescription	Pass/Fail
UT-32	UC9_NotifyError_MotorError_ExactOutput	MOTOR_ERROR 전달 시 출력 형식이 "[ERROR] MOTOR_ERROR\n"인지 확인	Pass
UT-33	UC9_NotifyError_CallsStopMoving	notifyError() 호출 시 motor.stopMoving이 호출되는지 확인	Pass
UT-34	UC9_NotifyError_CallsStopCleaning	notifyError() 호출 시 cleaner.stopCleaning이 호출되는지 확인	Pass
UT-35	UC9_NotifyError_OutputDoesNotContainRVCPrefix	notifyError() 출력에 "[RVC]" 접두사가 포함되지 않는지 확인	Pass
UT-36	UC9_NotifyError_DoesNotCallStartCleaning	notifyError() 호출 시 startCleaning은 호출되지 않는지 확인	Pass

Unit Testing

1. Test Case 분석

(1) 전원, 에러 관련 (UC1/UC8/UC9)
- 총 29개 Test Case
- 초기화, 종료, 에러 처리 포함

(2) 청소 모드 관련 (UC2/UC7 등)
- 총 52개 Test Case
- 초기화, 종료, 에러 처리 포함

(3) 이동·장애물 관련 (UC3~6)
- 총 79개 테스트 케이스
- 방향 전환, 센서 감지, 이동 명령 포함

(4) UI/Handler 관련
- 총 17개 Test Case
- 출력 포맷, 에러 메세지 포함

(5) 시스템 통합 (RVCOrchestrator)
- 총 36개 Test Case
- UC별 Orchestrator 흐름 검증

Unit Testing

2. Driver / Stub / Spy 적용

(1) Stub (ex. StubCleaner)

```cpp
#pragma once
#include "hardware/ICleaner.h"

class StubCleaner : public ICleaner {
public:
    bool startCleaningCalled = false;
    bool stopCleaningCalled = false;
    bool powerUpCalled = false;
    int callCount = 0;

    void startCleaning() override { startCleaningCalled = true; ++callCount; }
    void stopCleaning() override { stopCleaningCalled = true; ++callCount; }
    void powerUp() override { powerUpCalled = true; ++callCount; }
    void powerDown() override {}

    void reset() {
        startCleaningCalled = stopCleaningCalled = powerUpCalled = false;
        callCount = 0;
    }
};
```

```cpp
class CleanerControllerStubTest : public ::testing::Test {
protected:
    StubCleaner stub;
    CleanerController ctrl{&stub};

    void SetUp() override { stub.reset(); }
};
```

```cpp
// requestStartCleaning() 호출 시 하드웨어 startCleaning이 호출되는지 확인
TEST_F(CleanerControllerStubTest, RequestStartCleaningCallsHardware) {
    ctrl.requestStartCleaning();
    EXPECT_TRUE(stub.startCleaningCalled);
}
```

Unit Testing

2. Driver / Stub / Spy 적용

(2) Driver (ex. RVCOrchestratorTest)

```
class RVCOrchestratorTest : public ::testing::Test {
protected:
    CLIHandler cliHandler;

    StubMotor stubMotor;
    StubCleaner stubCleaner;
    StubSensor* frontSensor;
    StubSensor* leftSensor;
    StubSensor* rightSensor;

    MotorController motorCtrl{&stubMotor};
    CleanerController cleanerCtrl{&stubCleaner};
    std::unique_ptr<MovementPolicyController> movCtrl;
    RVCPowerController powerCtrl;
    CleaningPolicyController cleaningPolicyCtrl;


    std::unique_ptr<RVCOrchestrator> orchestrator;

    std::ostringstream captured;
    std::streambuf* originalBuf{};
```

```
void SetUp() override {
    stubMotor.reset();
    stubCleaner.reset();

    auto f = std::make_unique<StubSensor>();
    auto l = std::make_unique<StubSensor>();
    auto r = std::make_unique<StubSensor>();
    frontSensor = f.get();
    leftSensor  = l.get();
    rightSensor = r.get();
    movCtrl = std::make_unique<MovementPolicyController>(
        std::move(f), std::move(l), std::move(r));

    orchestrator = std::make_unique<RVCOrchestrator>(
        cliHandler,
        &powerCtrl,
        &motorCtrl,
        &cleanerCtrl,
        movCtrl.get(),
        &cleaningPolicyCtrl
    );

    originalBuf = std::cout.rdbuf(captured.rdbuf());
}
```

```
// [Positive] powerOn() 출력 형식이 정확한지 확인
TEST_F(RVCOrchestratorTest, UC1_PowerOn_ExactOutputFormat) {
    orchestrator->powerOn();
    EXPECT_EQ(output(), "[RVC] 시스템 준비 완료!\n");
}
```

Unit Testing

2. Driver / Stub / Spy 적용

(3) Spy (ex. SpyCleaner)

```cpp
class SpyCleaner : public Cleaner {
public:
    bool cleaningActive{false};
    bool boostActive{false};

    void startCleaning() override {
        cleaningActive = true;
        Cleaner::startCleaning();
    }
    void stopCleaning() override {
        cleaningActive = false;
        boostActive = false;
        Cleaner::stopCleaning();
    }
    void powerUp() override {
        boostActive = true;
        Cleaner::powerUp();
    }
    void powerDown() override {
        boostActive = false;
        Cleaner::powerDown();
    }
};
```

```cpp
// [Positive] UC7: powerUp 호출 시 부스트 활성화
TEST(CleanerTest, PowerUp_ActivatesBoost) {
    SpyCleaner c;
    c.powerUp();
    EXPECT_TRUE(c.boostActive);
}
```

Unit Testing

3. 테스트 품질 측정

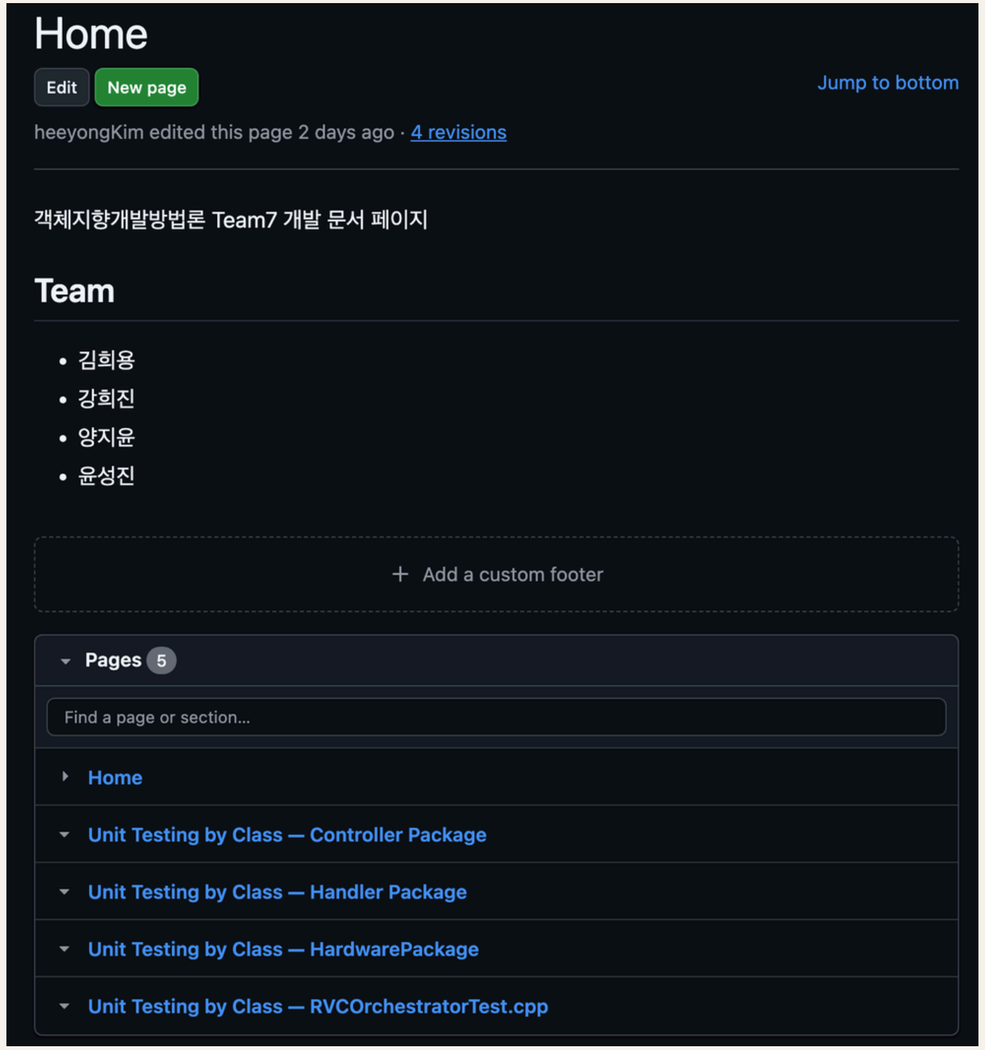
(1) 테스트 통과율

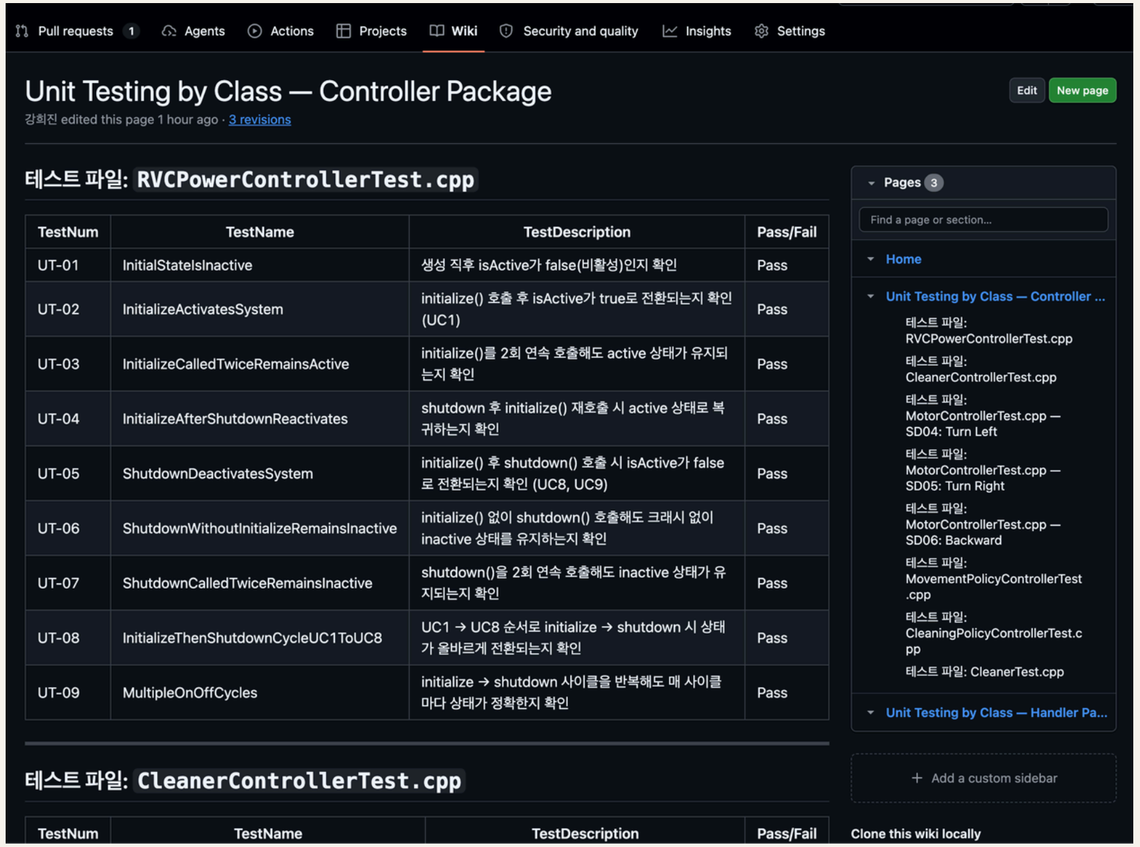
전체 Unit Test 213개 중 213개 Pass => Pass Ratio 100%

(2) 코드 커버리지 (main.cpp제외)

Unit Testing

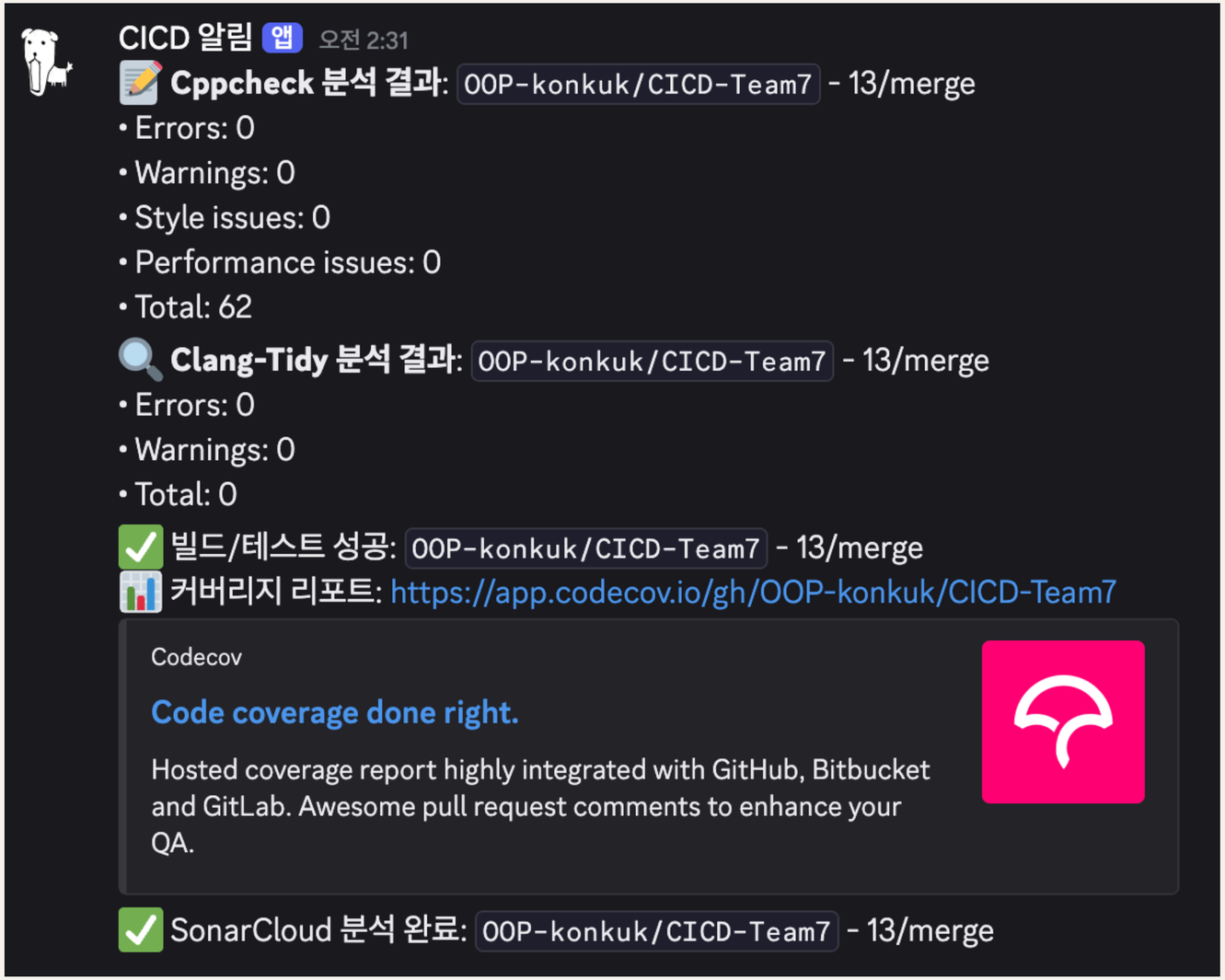
4. Unit Test 문서화 (GitHub Wiki)





Unit Testing

5. Discord Webhook 연동을 통해 CI/CD 결과를 실시간 알림 형태로 공유



System Test

Flow 1: Power Lifecycle (UC1 · UC8)

TestNum	TestName	TestDescription	Type
ST-01	PressOn_ThenOff_MessagesInOrder	전원 켜고 끄면 준비 완료 → 종료 메시지가 순서대로 출력되는지 확인	Positive
ST-02	PressOn_ThenOff_MotorIsStopped	전원을 끄면 모터가 STOPPED 상태가 되는지 확인	Positive
ST-03	PressOn_ThenOff_CleanerIsIdle	전원을 끄면 청소기가 IDLE 모드가 되는지 확인	Positive
ST-04	PressOn_Only_MotorRemainsStoped	전원만 켜면 모터가 STOPPED 상태를 유지하는지 확인	Negative
ST-05	PressOn_Only_CleanerRemainsIdle	전원만 켜면 청소기가 IDLE 모드를 유지하는지 확인	Negative
ST-06	PressOn_DoesNotOutputError	전원을 켤 때 에러 메시지가 출력되지 않는지 확인	Negative
ST-07	PressOn_MotorDoesNotMoveForward	전원을 켜는 것만으로 모터가 FORWARD 상태가 되지 않는지 확인	Negative
ST-08	PressOn_CleanerDoesNotEnterBoost	전원을 켜는 것만으로 청소기가 BOOST 모드가 되지 않는지 확인	Negative
ST-09	PressOff_WithoutPressOn_NoCrash	pressOn 없이 pressOff를 호출해도 크래시가 없는지 확인	Negative
ST-10	PressOff_DoesNotOutputError	정상 종료 시 에러 메시지가 출력되지 않는지 확인	Negative

System Test

Flow 2: Cleaning Session (UC2)

TestNum	TestName	TestDescription	Type
ST-01	Cleaning_CleanerEntersNormalMode	청소 시작 시 청소기가 NORMAL 모드로 전환되는지 확인	Positive
ST-02	Cleaning_MotorEntersForwardState	청소 시작 시 모터가 FORWARD 상태로 전환되는지 확인	Positive
ST-03	Cleaning_ThenPressOff_BothHardwareStopped	청소 후 전원을 끄면 모터 STOPPED·청소기 IDLE 상태가 되는지 확인	Positive
ST-04	Cleaning_MotorIsNotStopped	청소 중 모터가 STOPPED 상태가 아닌지 확인	Negative
ST-05	Cleaning_CleanerIsNotIdle	청소 중 청소기가 IDLE 모드가 아닌지 확인	Negative
ST-06	Cleaning_CleanerDoesNotEnterBoost	기본 청소 시작 시 청소기가 BOOST 모드가 되지 않는지 확인	Negative
ST-07	Cleaning_MotorDoesNotTurnLeft	청소 중 모터가 TURN_LEFT 상태가 되지 않는지 확인	Negative
ST-08	Cleaning_MotorDoesNotTurnRight	청소 중 모터가 TURN_RIGHT 상태가 되지 않는지 확인	Negative
ST-09	Cleaning_MotorDoesNotMoveBackward	청소 중 모터가 BACKWARD 상태가 되지 않는지 확인	Negative
ST-10	Cleaning_DoesNotOutputError	정상 청소 시작 시 에러 메시지가 출력되지 않는지 확인	Negative

System Test

Flow 3: Obstacle Avoidance (UC3)

TestNum	TestName	TestDescription	Type
ST-01	EnvLeftFree_MotorEntersTurnLeftState	왼쪽 장애물 없음 환경에서 장애물 감지 시 모터가 TURN_LEFT 상태가 되는지 확인	Positive
ST-02	EnvRightFree_MotorEntersTurnRightState	오른쪽만 열린 환경에서 장애물 감지 시 모터가 TURN_RIGHT 상태가 되는지 확인	Positive
ST-03	EnvBothBlocked_MotorEntersBackwardState	양방향 막힌 환경에서 장애물 감지 시 모터가 BACKWARD 상태가 되는지 확인	Positive
ST-04	Cleaning_ThenObstacle_CleanerBecomesIdle	청소 중 장애물 감지 시 청소기가 IDLE 모드로 전환되는지 확인	Positive
ST-05	Obstacle_ThenResumeCleaning_MotorReturnsForward	장애물 회피 후 재청소 시 모터가 FORWARD 상태로 복귀하는지 확인	Negative
ST-06	EnvLeftFree_MotorDoesNotTurnRight	왼쪽 열린 환경에서 장애물 감지 시 모터가 TURN_RIGHT 상태가 되지 않는지 확인	Negative
ST-07	EnvLeftFree_MotorDoesNotMoveBackward	왼쪽 열린 환경에서 장애물 감지 시 모터가 BACKWARD 상태가 되지 않는지 확인	Negative

System Test

ST-08	EnvRightFree_MotorDoesNotTurnLeft	오른쪽만 열린 환경에서 장애물 감지 시 모터가 TURN_LEFT 상태가 되지 않는지 확인	Negative
ST-09	EnvRightFree_MotorDoesNotMoveBackward	오른쪽만 열린 환경에서 장애물 감지 시 모터가 BACKWARD 상태가 되지 않는지 확인	Negative
ST-10	EnvBothBlocked_MotorDoesNotTurnLeft	양방향 막힌 환경에서 모터가 TURN_LEFT 상태가 되지 않는지 확인	Negative
ST-11	EnvBothBlocked_MotorDoesNotTurnRight	양방향 막힌 환경에서 모터가 TURN_RIGHT 상태가 되지 않는지 확인	Negative
ST-12	Obstacle_CleanerDoesNotEnterBoost	장애물 감지 시 청소기가 BOOST 모드로 전환되지 않는지 확인	Negative

System Test

Flow 4: Backward And Turn (UC6)

TestNum	TestName	TestDescription	Type
ST-01	EnvLeftFree_MotorFinalStateTurnLeft	왼쪽 열린 환경에서 backwardAndTurn 후 모터가 TURN_LEFT 상태가 되는지 확인	Positive
ST-02	EnvRightFree_MotorFinalStateTurnRight	오른쪽만 열린 환경에서 backwardAndTurn 후 모터가 TURN_RIGHT 상태가 되는지 확인	Positive
ST-03	AfterObstacle_BackwardAndTurn_MotorMovesBackward	장애물 감지 후 backwardAndTurn 시 모터가 BACKWARD 상태를 거치는지 확인	Positive
ST-04	EnvBothBlocked_MotorStaysBackward_NoTurn	양방향 막힌 환경에서 backwardAndTurn 후 회전 없이 BACKWARD 상태인지 확인	Negative
ST-05	BackwardAndTurn_CleanerModeUnchanged	backwardAndTurn은 청소기 모드를 변경하지 않는지 확인	Negative
ST-06	EnvLeftFree_MotorDoesNotTurnRight_AfterBackward	왼쪽 열린 환경에서 backwardAndTurn 후 모터가 TURN_RIGHT가 되지 않는지 확인	Negative
ST-07	EnvRightFree_MotorDoesNotTurnLeft_AfterBackward	오른쪽만 열린 환경에서 backwardAndTurn 후 모터가 TURN_LEFT가 되지 않는지 확인	Negative

System Test

ST-08	BackwardAndTurn_CleanerDoesNotEnterBoost	backwardAndTurn 호출 시 청소기가 BOOST 모드가 되지 않는지 확인	Negative
ST-09	BackwardAndTurn_CleanerDoesNotStartCleaning	backwardAndTurn 호출 시 청소기가 NORMAL 청소 모드로 전환되지 않는지 확인	Negative

System Test

Flow 5: Boost Cleaning (UC7)

TestNum	TestName	TestDescription	Type
ST-01	Boost_CleanerEntersBoostMode	부스트 청소 요청 시 청소기가 BOOST 모드로 전환되는지 확인	Positive
ST-02	Cleaning_ThenBoost_CleanerUpgradesToBoost	NORMAL 청소 후 부스트 요청 시 청소기가 BOOST 모드로 업그레이드되는지 확인	Positive
ST-03	Boost_ThenPressOff_CleanerReturnsToIdle	부스트 청소 후 전원을 끄면 청소기가 IDLE 모드로 복귀하는지 확인	Positive
ST-04	Boost_CleanerDoesNotBecomeIdle	부스트 청소는 청소기를 IDLE로 전환하지 않는지 확인	Negative
ST-05	Boost_MotorDoesNotMoveForward	부스트 청소 요청 시 모터가 FORWARD 상태가 되지 않는지 확인	Negative
ST-06	Boost_MotorDoesNotMoveBackward	부스트 청소 요청 시 모터가 BACKWARD 상태가 되지 않는지 확인	Negative
ST-07	Boost_MotorDoesNotTurnLeft	부스트 청소 요청 시 모터가 TURN_LEFT 상태가 되지 않는지 확인	Negative
ST-08	Boost_MotorDoesNotTurnRight	부스트 청소 요청 시 모터가 TURN_RIGHT 상태가 되지 않는지 확인	Negative
ST-09	Boost_DoesNotOutputError	부스트 청소 요청 시 에러 메시지가 출력되지 않는지 확인	Negative

System Test

Flow 6: Error Handling (UC9)

TestNum	TestName	TestDescription	Type
ST-01	Error_MotorEntersStoppedState	오류 발생 시 모터가 STOPPED 상태로 전환되는지 확인	Positive
ST-02	Cleaning_ThenError_CleanerSafeStop	청소 중 오류 발생 시 청소기가 IDLE 모드로 안전 정지하는지 확인	Positive
ST-03	Error_ThenPressOff_HardwareRemainsInSafeState	오류 후 전원을 끄면 모터와 청소기가 안전 상태를 유지하는지 확인	Positive
ST-04	Error_CleanerDoesNotResumeCleaning	오류 발생 후 청소기가 NORMAL 모드가 되지 않는지 확인	Negative
ST-05	Error_MotorDoesNotMoveForward	오류 발생 후 모터가 FORWARD 상태가 되지 않는지 확인	Negative
ST-06	Error_CleanerDoesNotEnterBoost	오류 발생 후 청소기가 BOOST 모드가 되지 않는지 확인	Negative
ST-08	Error_OutputDoesNotContainRVCPrefix	에러 출력 메시지에 [RVC] 접두사가 포함되지 않는지 확인	Negative

System Test

Flow 7: Complete Sessions (전체 UC 흐름)

TestNum	TestName	TestDescription	Type
ST-01	UC1_Avoid_UC8_FinalSafeState	UC1→UC3→UC8: 전원 켜기→장애물 회피(좌회전)→전원 끄기 후 STOPPED/IDLE 상태인지 확인	Positive
ST-02	UC2_BackwardAvoid_UC2_MotorReturnsForward	UC2→UC6→UC2: 청소→후진 회피→재청소 시 모터가 FORWARD로 복귀하는지 확인	Positive
ST-03	UC1_Boost_UC8_FinalSafeState	UC1→UC7→UC8: 전원 켜기→부스트 청소→전원 끄기 후 STOPPED/IDLE 상태인지 확인	Positive
ST-04	UC9_ThenUC8_SafeShutdownMessageDisplayed	UC9→UC8: 오류 발생 후 전원 끄기 시 [ERROR]·종료 메시지가 모두 출력되는지 확인	Negative
ST-05	UC2_Obstacle_UC8_CleanerIsIdle	UC2→UC3→UC8: 청소→장애물 감지→전원 끄기 후 청소기가 IDLE 상태인지 확인	Negative
ST-06	UC1_UC8_NoErrorOutput	UC1→UC2→UC8: 정상 흐름에서 에러 메시지가 출력되지 않는지 확인	Negative
ST-07	UC9_MotorDoesNotResumeAfterError	UC9 이후 별도 재시작 없이 모터가 자동으로 FORWARD 상태가 되지 않는지 확인	Negative
ST-08	UC8_CleanerDoesNotAutoRestart	UC8 이후 청소기가 자동으로 재시작되지 않고 IDLE을 유지하는지 확인	Negative
ST-09	UC2_UC3_MotorNotForwardAfterObstacle	UC2→UC3: 청소 중 장애물 감지 후 모터가 FORWARD 상태가 되지 않는지 확인	Negative

System Test

개요

1. Pass Ratio = 66/66 (100%)
2. Positive Test Case (22개)
3. Negative Test Case (44개)
4. Neagtive Test Case Ratio = 44/66 (66.7%)

Flow	Test Case 개수	Positive Test	Negative Test
Flow1. Power Lifecycle	10	3	7
Flow2. Cleaning Session	10	3	7
Flow3. Obstacle Avoidance	12	4	8
Flow4. Backward And Turn	9	3	6

Flow	Test Case 개수	Positive Test	Negative Test
Flow5. Boost Cleaning	9	3	6
Flow6. Error Handling	7	3	4
Flow7. Complete Session	9	3	6
합계	66	22	44

System Test

(1) 시뮬레이터 구현코드

```cpp
#pragma once
#include <chrono>
#include <fstream>
#include <functional>
#include <iostream>
#include <sstream>
#include <string>
#include <vector>

struct TestResult {
    std::string suiteName;
    std::string testName;
    bool        passed{true};
    std::string failMessage;
    double      elapsedMs{0.0};
};

class TestRunner {
public:
    struct Registration {
        std::string           suite;
        std::string           name;
        std::function<void()> fn;
    };

    static TestRunner& instance() {
        static TestRunner inst;
        return inst;
    }

    void registerTest(const std::string& suite,
                      const std::string& name,
                      std::function<void()> fn) {
        registrations_.push_back({suite, name, std::move(fn)});
    }
```

```cpp
int runAll(const std::string& xmlOutputPath = "") {
    int passed = 0, failed = 0;

    for (auto& reg : registrations_) {
        TestResult result;
        result.suiteName = reg.suite;
        result.testName  = reg.name;
        current_         = &result;

        auto t0 = std::chrono::steady_clock::now();
        try {
            reg.fn();
        } catch (const std::exception& ex) {
            recordFailure(std::string("EXCEPTION: ") + ex.what());
        } catch (...) {
            recordFailure("UNKNOWN EXCEPTION");
        }
        auto t1        = std::chrono::steady_clock::now();
        result.elapsedMs =
            std::chrono::duration<double, std::milli>(t1 - t0).count();
        current_ = nullptr;

        if (result.passed) {
            std::cout << "[PASS] " << reg.suite << "::" << reg.name << "\n";
            ++passed;
        } else {
            std::cout << "[FAIL] " << reg.suite << "::" << reg.name
                      << "\n       " << result.failMessage << "\n";
            ++failed;
        }
        results_.push_back(std::move(result));
    }

    std::cout << "\n=== System Test Results ===\n"
              << "Passed: " << passed << "\n"
              << "Failed: " << failed << "\n"
              << "Total : " << (passed + failed) << "\n";

    if (!xmlOutputPath.empty())
        writeJUnit(xmlOutputPath, failed);

    return (failed > 0) ? 1 : 0;
}
```

System Test

(1) 시뮬레이터 구현코드

```
private:
    TestRunner() = default;
    std::vector<Registration> registrations_;
    std::vector<TestResult>   results_;
    TestResult*               current_{nullptr};

    static std::string xmlEscape(const std::string& s) {
        std::string out;
        for (char c : s) {
            switch (c) {
                case '<': out += "&lt;";   break;
                case '>': out += "&gt;";   break;
                case '&': out += "&amp;";  break;
                case '"': out += "&quot;"; break;
                default:  out += c;
            }
        }
        return out;
    }
```

```
    void writeJUnit(const std::string& path, int totalFailed) {
        std::ofstream f(path);
        if (!f) return;

        double totalMs = 0.0;
        for (auto& r : results_) totalMs += r.elapsedMs;

        f << "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n"
          << "<testsuites tests=\"" << results_.size()
          << "\" failures=\"" << totalFailed
          << "\" time=\"" << (totalMs / 1000.0) << "\">\n";

        std::string cur;
        for (auto& r : results_) {
            if (r.suiteName != cur) {
                if (!cur.empty()) f << "  </testsuite>\n";
                cur = r.suiteName;
                int sf = 0, st = 0; double sm = 0.0;
                for (auto& s : results_) {
                    if (s.suiteName != cur) continue;
                    ++st; sm += s.elapsedMs;
                    if (!s.passed) ++sf;
                }
                f << "  <testsuite name=\"" << xmlEscape(cur)
                  << "\" tests=\"" << st
                  << "\" failures=\"" << sf
                  << "\" time=\"" << (sm / 1000.0) << "\">\n";
            }
            f << "    <testcase name=\"" << xmlEscape(r.testName)
              << "\" classname=\"" << xmlEscape(r.suiteName)
              << "\" time=\"" << (r.elapsedMs / 1000.0) << "\"";
            if (r.passed) {
                f << "/>\n";
            } else {
                f << ">\n      <failure message=\""
                  << xmlEscape(r.failMessage) << "\"/>\n    </testcase>\n";
            }
        }
        if (!cur.empty()) f << "  </testsuite>\n";
        f << "</testsuites>\n";
    }
};
```

```
#define STEST_FAIL(msg) \
    do { \
        std::ostringstream _oss_; \
        _oss_ << __FILE__ << ":" << __LINE__ << " | " << (msg); \
        TestRunner::instance().recordFailure(_oss_.str()); \
    } while (0)

#define STEST_EXPECT_TRUE(expr) \
    do { if (!(expr)) STEST_FAIL("Expected TRUE: " #expr); } while (0)

#define STEST_EXPECT_FALSE(expr) \
    do { if ((expr))  STEST_FAIL("Expected FALSE: " #expr); } while (0)

#define STEST_EXPECT_EQ(a, b) \
    do { \
        if (!((a) == (b))) { \
            std::ostringstream _oss_; \
            _oss_ << "Expected EQ: " #a " == " #b \
                  << " (got " << (a) << " vs " << (b) << ")"; \
            STEST_FAIL(_oss_.str()); \
        } \
    } while (0)

#define STEST_EXPECT_CONTAINS(str, sub) \
    do { \
        if ((str).find(sub) == std::string::npos) \
            STEST_FAIL("Expected \"" + std::string(sub) + \
                       "\" in \"" + (str) + "\""); \
    } while (0)

#define STEST_EXPECT_NOT_CONTAINS(str, sub) \
    do { \
        if ((str).find(sub) != std::string::npos) \
            STEST_FAIL("Did NOT expect \"" + std::string(sub) + \
                       "\" in \"" + (str) + "\""); \
    } while (0)

// ── Registration macro ──────────────────────────────────────
#define STEST_REGISTER(suite, name, fn) \
    namespace { \
        const bool _sreg_##suite##_##name = []{ \
            TestRunner::instance().registerTest(#suite, #name, fn); \
            return true; \
        }(); \
    }

// ── Place exactly once at the bottom of the test .cpp file ──
#define STEST_MAIN() \
    int main(int argc, char** argv) { \
        std::string xmlPath; \
        for (int i = 1; i < argc; ++i) { \
            std::string a(argv[i]); \
            if (a.rfind("--xml=", 0) == 0) xmlPath = a.substr(6); \
        } \
        return TestRunner::instance().runAll(xmlPath); \
    }
```

System Test

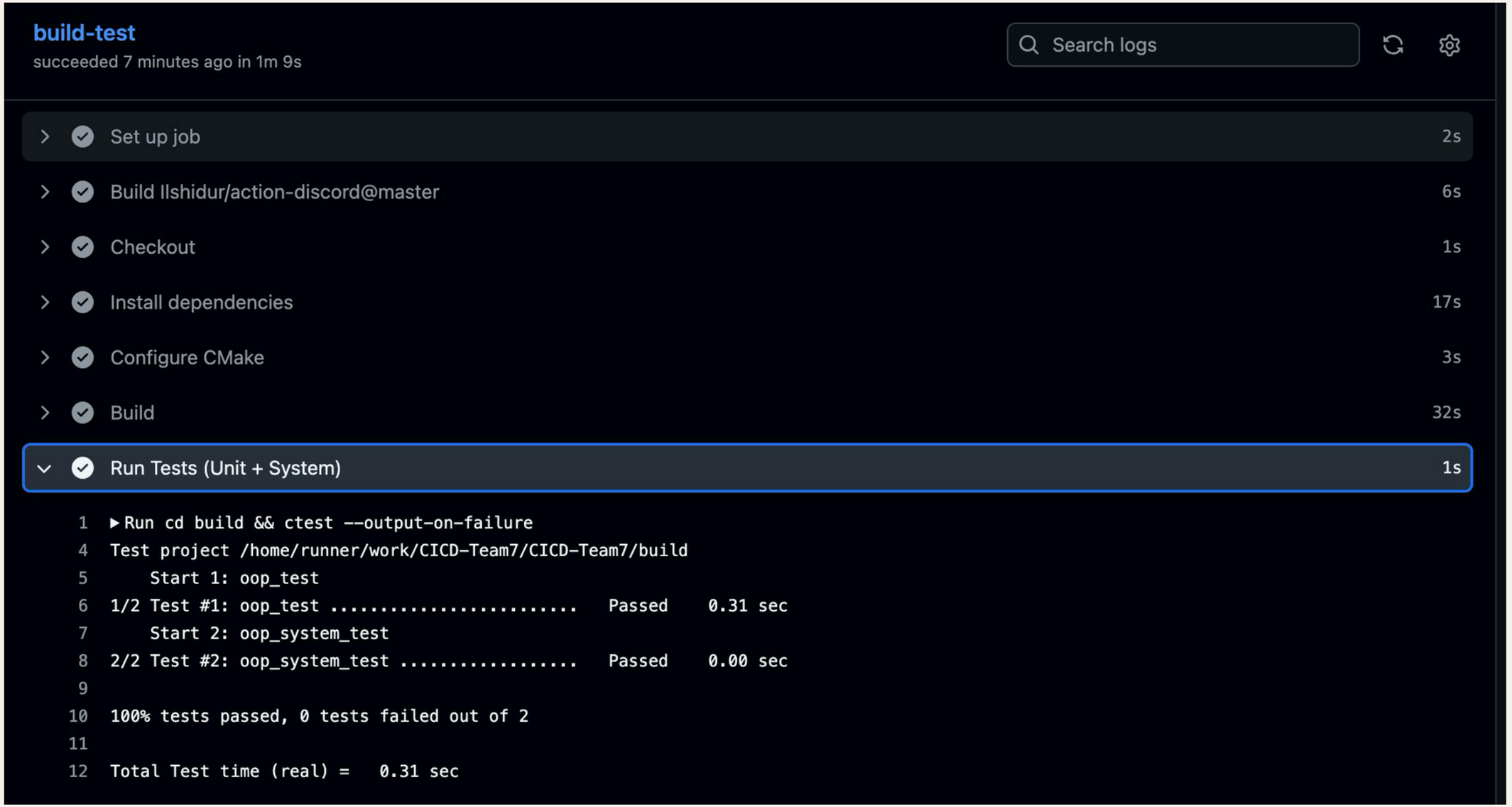
(2) 시뮬레이터 테스트 화면

```
heeedragon@heeyongkim-MacBookAir CICD-Team7 % ./build/oop
전원을 켜려면 Enter, 종료하려면 'q'를 입력하세요.

[RVC] 시스템 준비 완료!
q
[RVC] 시스템 종료 중...
```

```
heeedragon@heeyongkim-MacBookAir CICD-Team7 % ./build/oop_system_test --xml=build/system_test_results.xml

[PASS] Flow1_PowerLifecycle::PressOn_ThenOff_MessagesInOrder
[PASS] Flow1_PowerLifecycle::PressOn_ThenOff_MotorIsStopped
[PASS] Flow1_PowerLifecycle::PressOn_ThenOff_CleanerIsIdle
[PASS] Flow1_PowerLifecycle::PressOn_Only_MotorRemainsStoped
[PASS] Flow1_PowerLifecycle::PressOn_Only_CleanerRemainsIdle
[PASS] Flow1_PowerLifecycle::PressOn_MotorDoesNotMoveForward
[PASS] Flow1_PowerLifecycle::PressOn_CleanerDoesNotEnterBoost
[PASS] Flow1_PowerLifecycle::PressOn_DoesNotOutputError
[PASS] Flow1_PowerLifecycle::PressOff_DoesNotOutputError
[PASS] Flow1_PowerLifecycle::PressOff_WithoutPressOn_NoCrash
[PASS] Flow2_CleaningSession::Cleaning_CleanerEntersNormalMode
[PASS] Flow2_CleaningSession::Cleaning_MotorEntersForwardState
[PASS] Flow2_CleaningSession::Cleaning_ThenPressOff_BothHardwareStopped
[PASS] Flow2_CleaningSession::Cleaning_MotorIsNotStopped
[PASS] Flow2_CleaningSession::Cleaning_CleanerIsNotIdle
[PASS] Flow2_CleaningSession::Cleaning_CleanerDoesNotEnterBoost
[PASS] Flow2_CleaningSession::Cleaning_MotorDoesNotTurnLeft
[PASS] Flow2_CleaningSession::Cleaning_MotorDoesNotTurnRight
[PASS] Flow2_CleaningSession::Cleaning_MotorDoesNotMoveBackward
[PASS] Flow2_CleaningSession::Cleaning_DoesNotOutputError
[PASS] Flow3_ObstacleAvoidance::EnvLeftFree_MotorEntersTurnLeftState
[PASS] Flow3_ObstacleAvoidance::EnvRightFree_MotorEntersTurnRightState
[PASS] Flow3_ObstacleAvoidance::EnvBothBlocked_MotorEntersBackwardState
[PASS] Flow3_ObstacleAvoidance::Cleaning_ThenObstacle_CleanerBecomesIdle
[PASS] Flow3_ObstacleAvoidance::Obstacle_ThenResumeCleaning_MotorReturnsForward
[PASS] Flow3_ObstacleAvoidance::EnvLeftFree_MotorDoesNotTurnRight
[PASS] Flow3_ObstacleAvoidance::EnvLeftFree_MotorDoesNotMoveBackward
[PASS] Flow3_ObstacleAvoidance::EnvRightFree_MotorDoesNotTurnLeft
[PASS] Flow3_ObstacleAvoidance::EnvRightFree_MotorDoesNotMoveBackward
[PASS] Flow3_ObstacleAvoidance::EnvBothBlocked_MotorDoesNotTurnLeft
[PASS] Flow3_ObstacleAvoidance::EnvBothBlocked_MotorDoesNotTurnRight
[PASS] Flow3_ObstacleAvoidance::Obstacle_CleanerDoesNotEnterBoost
[PASS] Flow4_BackwardAndTurn::EnvLeftFree_MotorFinalStateTurnLeft
[PASS] Flow4_BackwardAndTurn::EnvRightFree_MotorFinalStateTurnRight
[PASS] Flow4_BackwardAndTurn::AfterObstacle_BackwardAndTurn_MotorMovesBackward
[PASS] Flow4_BackwardAndTurn::EnvBothBlocked_MotorStaysBackward_NoTurn
[PASS] Flow4_BackwardAndTurn::BackwardAndTurn_CleanerModeUnchanged
[PASS] Flow4_BackwardAndTurn::EnvLeftFree_MotorDoesNotTurnRight_AfterBackward
[PASS] Flow4_BackwardAndTurn::EnvRightFree_MotorDoesNotTurnLeft_AfterBackward
[PASS] Flow4_BackwardAndTurn::BackwardAndTurn_CleanerDoesNotEnterBoost
[PASS] Flow4_BackwardAndTurn::BackwardAndTurn_CleanerDoesNotStartCleaning
[PASS] Flow5_BoostCleaning::Boost_CleanerEntersBoostMode
[PASS] Flow5_BoostCleaning::Cleaning_ThenBoost_CleanerUpgradesToBoost
[PASS] Flow5_BoostCleaning::Boost_ThenPressOff_CleanerReturnsToIdle
[PASS] Flow5_BoostCleaning::Boost_CleanerDoesNotBecomeIdle
[PASS] Flow5_BoostCleaning::Boost_MotorDoesNotMoveForward
[PASS] Flow5_BoostCleaning::Boost_MotorDoesNotMoveBackward
[PASS] Flow5_BoostCleaning::Boost_MotorDoesNotTurnLeft
[PASS] Flow5_BoostCleaning::Boost_MotorDoesNotTurnRight
[PASS] Flow5_BoostCleaning::Boost_DoesNotOutputError
[PASS] Flow6_ErrorHandling::Error_MotorEntersStoppedState
[PASS] Flow6_ErrorHandling::Cleaning_ThenError_CleanerSafeStop
[PASS] Flow6_ErrorHandling::Error_ThenPressOff_HardwareRemainsInSafeState
[PASS] Flow6_ErrorHandling::Error_CleanerDoesNotResumeCleaning
[PASS] Flow6_ErrorHandling::Error_CleanerDoesNotEnterBoost
[PASS] Flow6_ErrorHandling::Error_MotorDoesNotMoveForward
[PASS] Flow6_ErrorHandling::Error_OutputDoesNotContainRVCPrefix
[PASS] Flow7_CompleteSession::UC1_Avoid_UC8_FinalSafeState
[PASS] Flow7_CompleteSession::UC2_BackwardAvoid_UC2_MotorReturnsForward
[PASS] Flow7_CompleteSession::UC1_Boost_UC8_FinalSafeState
[PASS] Flow7_CompleteSession::UC9_ThenUC8_SafeShutdownMessageDisplayed
[PASS] Flow7_CompleteSession::UC2_Obstacle_UC8_CleanerIsIdle
[PASS] Flow7_CompleteSession::UC1_UC8_NoErrorOutput
[PASS] Flow7_CompleteSession::UC9_MotorDoesNotResumeAfterError
[PASS] Flow7_CompleteSession::UC8_CleanerDoesNotAutoRestart
[PASS] Flow7_CompleteSession::UC2_UC3_MotorNotForwardAfterObstacle

=== System Test Results ===
Passed: 66
Failed: 0
Total : 66
```

System Test

(3) 테스트 수행 과정 및 결과 (CI)



Static Code Analysis

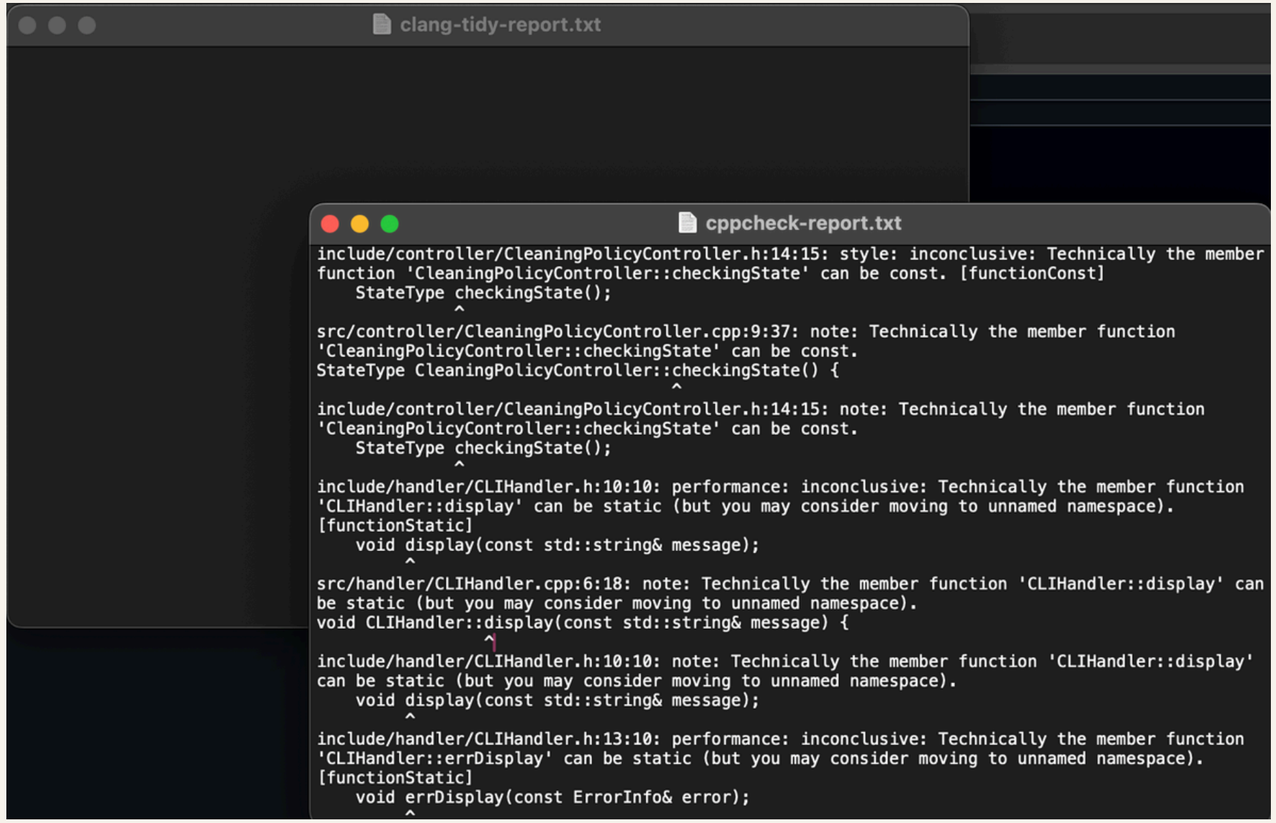
(1) Clang-Tidy, Cppcheck

📝 **Cppcheck 분석 결과:** `OOP-konkuk/CICD-Team7` - main
- Errors: 0
- Warnings: 0
- Style issues: 3
- Performance issues: 2
- Total: 37

🔍 **Clang-Tidy 분석 결과:** `OOP-konkuk/CICD-Team7` - main
- Errors: 0
- Warnings: 0
- Total: 0

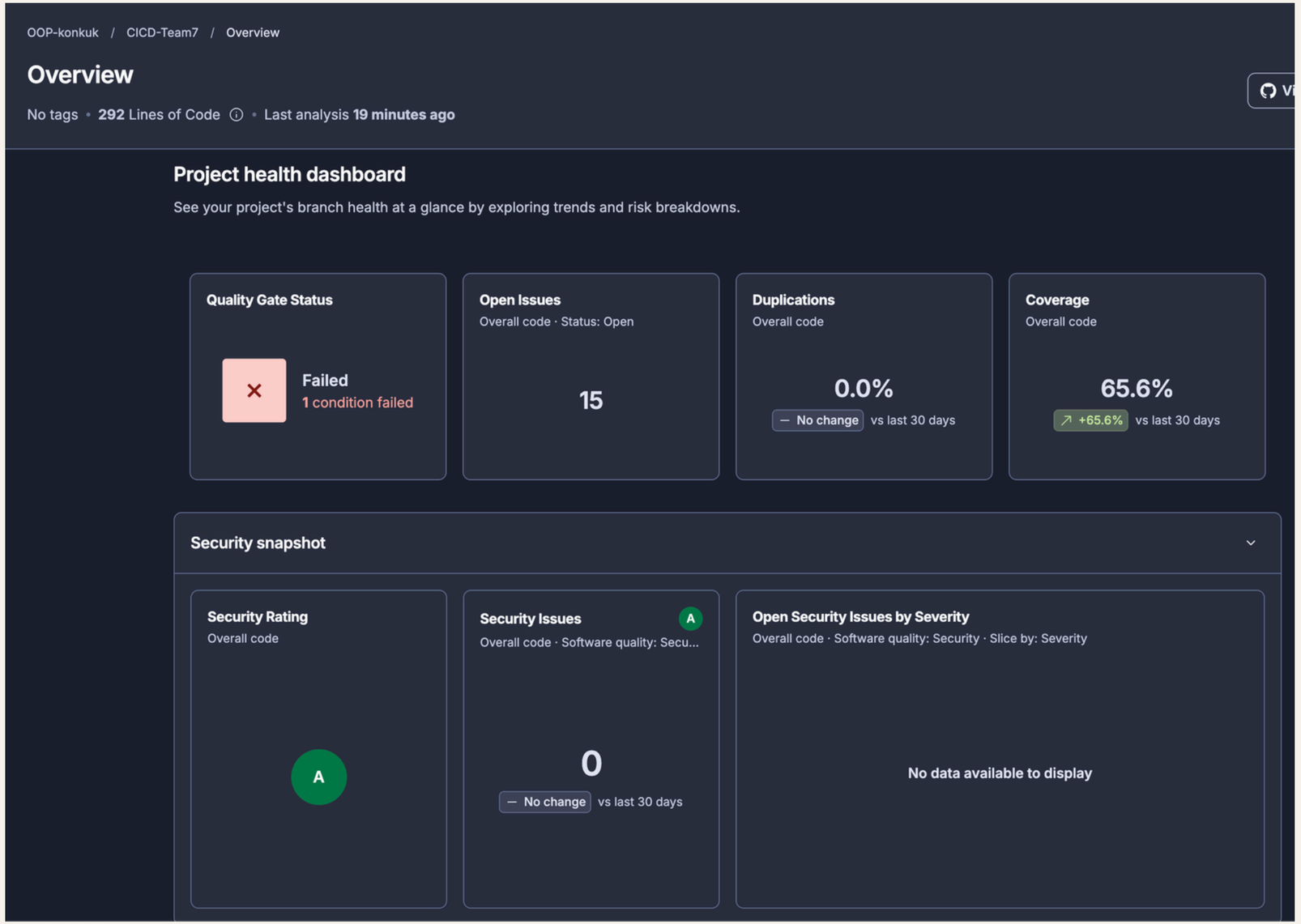
✅ 빌드/테스트/배포 성공: OOP-konkuk/CICD-Team7 - main

✅ SonarCloud 분석 완료: `OOP-konkuk/CICD-Team7` - main



Static Code Analysis

(2) SonarCloud



감사합니다.