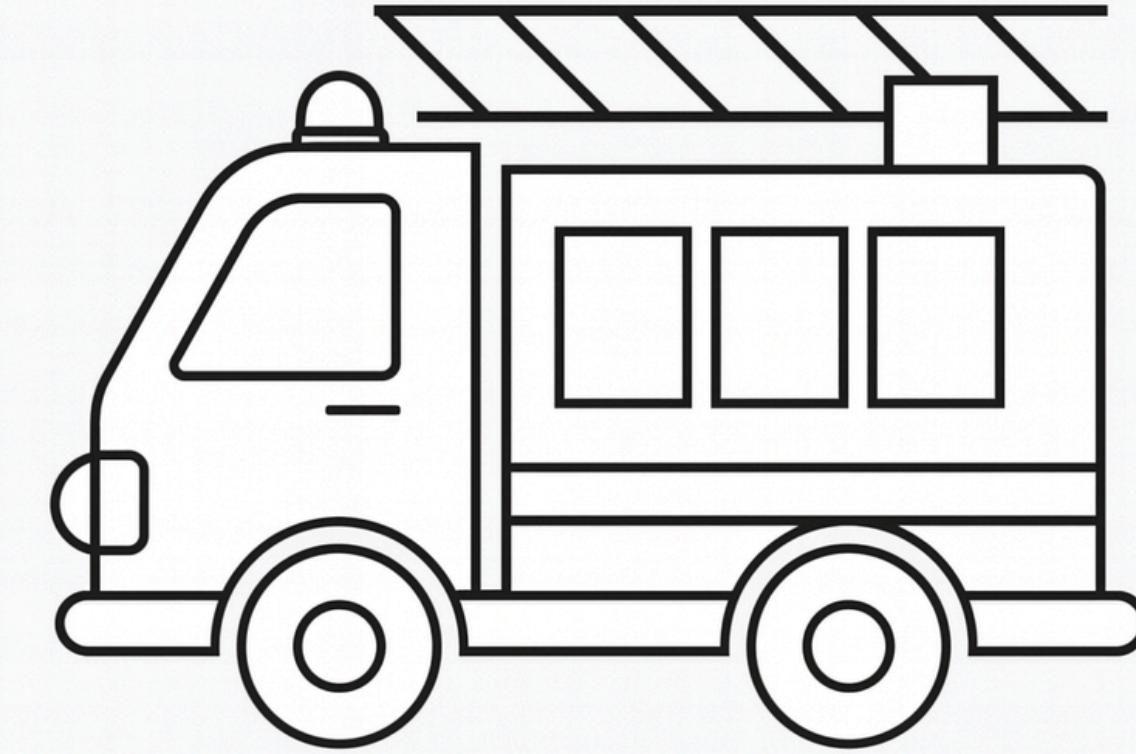


화재 감지와 협동 로봇 Fire Guard AMR

SLAM기반 자율주행 로봇 시스템 구현



F1&F3조_FF

강동혁 | 김갑민 | 김다빈 | 김정욱 | 김효원 | 이용우 | 이효원 | 황혜인

Contents

01 프로젝트 개요

02 프로젝트 팀 구성 및 역할

03 프로젝트 수행 절차 및 방법

04 프로젝트 개발 목차

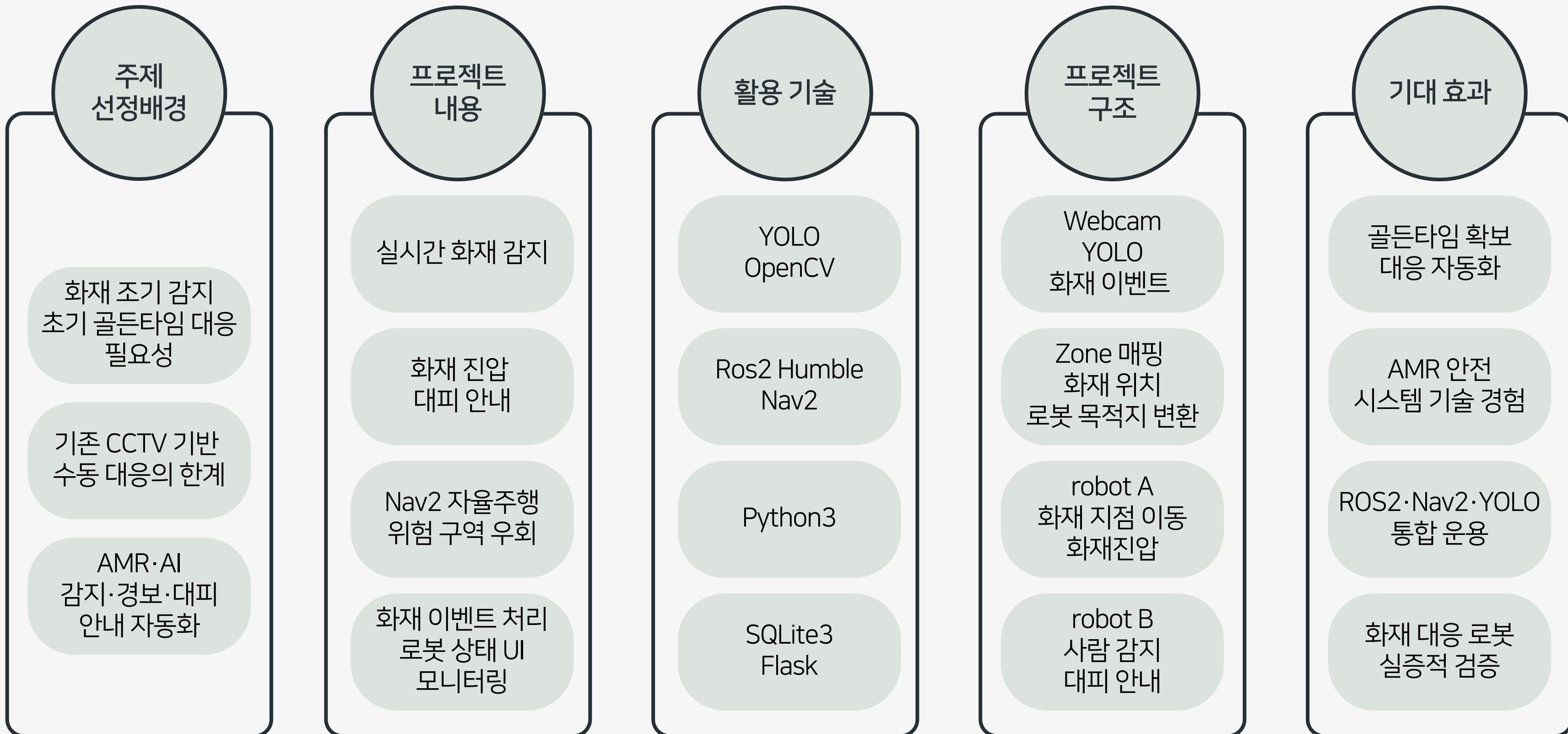
05 프로젝트 수행 경과

06 프로젝트 결과 및 분석

07 자체 평가 의견

08 질문 및 답변

01 프로젝트 개요



02 프로젝트 팀 구성 및 역할

| 조원 | 역할 | 담당 업무 | |
|------------|----|---|---|
| 강동혁 김정욱 | 팀원 | 실제 환경 구성 및 SLAM 기반 지도 생성 YOLO 기반 Alert Sound 노드 패키지 개발 | 로봇 참조 프레임 구성 및 원점 교정 동영상 편집 |
| 김갑민 김효원 | 팀장 | Custom Map 생성 로봇B 사람 Detect 및 Approach 기능 구현 | 웹캠 - DB - 모니터링 시스템 통합 로봇B Patrol 기능 구현 |
| 김다빈 이효원 | 팀원 | YOLO fine tuning system diagram 작성 | Webcam 화재감지 구현 로봇A와 통합 |
| 이용우 황혜인 | 팀원 | Flask 서버 구현 DB 로그 기록 | UI 디자인 UI를 활용한 시스템 통합 |

03 프로젝트 수행 절차 및 방법

| 구분 | 기간 | 활동 | 비고 |
|------------|--------------------|---|----------------------|
| 화재감지 (웹캠) | 11/30(일) ~ 12/3(수) | <ul style="list-style-type: none">Fire dataset 확보 및 YOLO 모델 생성WebCam 영상 입력 테스트 <ul style="list-style-type: none">웹캠 기반 화재 감지 topic publish 구현YOLO 학습 모델 테스트 및 검증 | PC1 화재 감지 서버 구축 |
| DB | 12/1(월) ~ 12/3(수) | <ul style="list-style-type: none">SQLite DB 구조 설계Topic 기반 DB update 기능 구현 <ul style="list-style-type: none">Topic → DB 저장 기능 test | UI·로그 시스템 구축 |
| 화재진압 (로봇A) | 12/1(월) ~ 12/4(목) | <ul style="list-style-type: none">Custom Map 생성 및 LocalizationNav2 기반 Zone A/B 이동 테스트 <ul style="list-style-type: none">Detect 기능 + Buzzer 출력 기능 구현Zone goal 이동 후 detect 검증 | Robot A FireMode 구축 |
| 화재대피 (로봇B) | 12/1(월) ~ 12/4(목) | <ul style="list-style-type: none">Custom Map 생성 및 LocalizationLoop 주행 기능 구현 및 test <ul style="list-style-type: none">Person detect → 안내 기능 연동대피 안내(TTS) 기능 test | Robot B GuideMode 구축 |
| 모니터링 | 12/1(월) ~ 12/3(수) | <ul style="list-style-type: none">Flask 기반 UI 서버 구현WebCam·DB·로봇 상태 통합 시각화 <ul style="list-style-type: none">fire_zone 시각화 기능 testDB 로그 기능 점검 | 실시간 관제 시스템 완성 |
| 통합 | 12/4(목) ~ 12/5(금) | <ul style="list-style-type: none">WebCam → FireEvent → Robot A/B 전체 기능 통합Persom detect → Robot B 동작 통합 <ul style="list-style-type: none">Robot A/B 상태 + UI 모듈 전체 연동전체 시나리오 검증 (감지 → 출동 → 재확인 → 안내 → UI) | 최종 점검 및 발표 준비 |

04 프로젝트 개발 목차

1

Map 생성

2

화재감지 (웹캠)

3

화재진압 로봇 A

4

화재대피 로봇 B

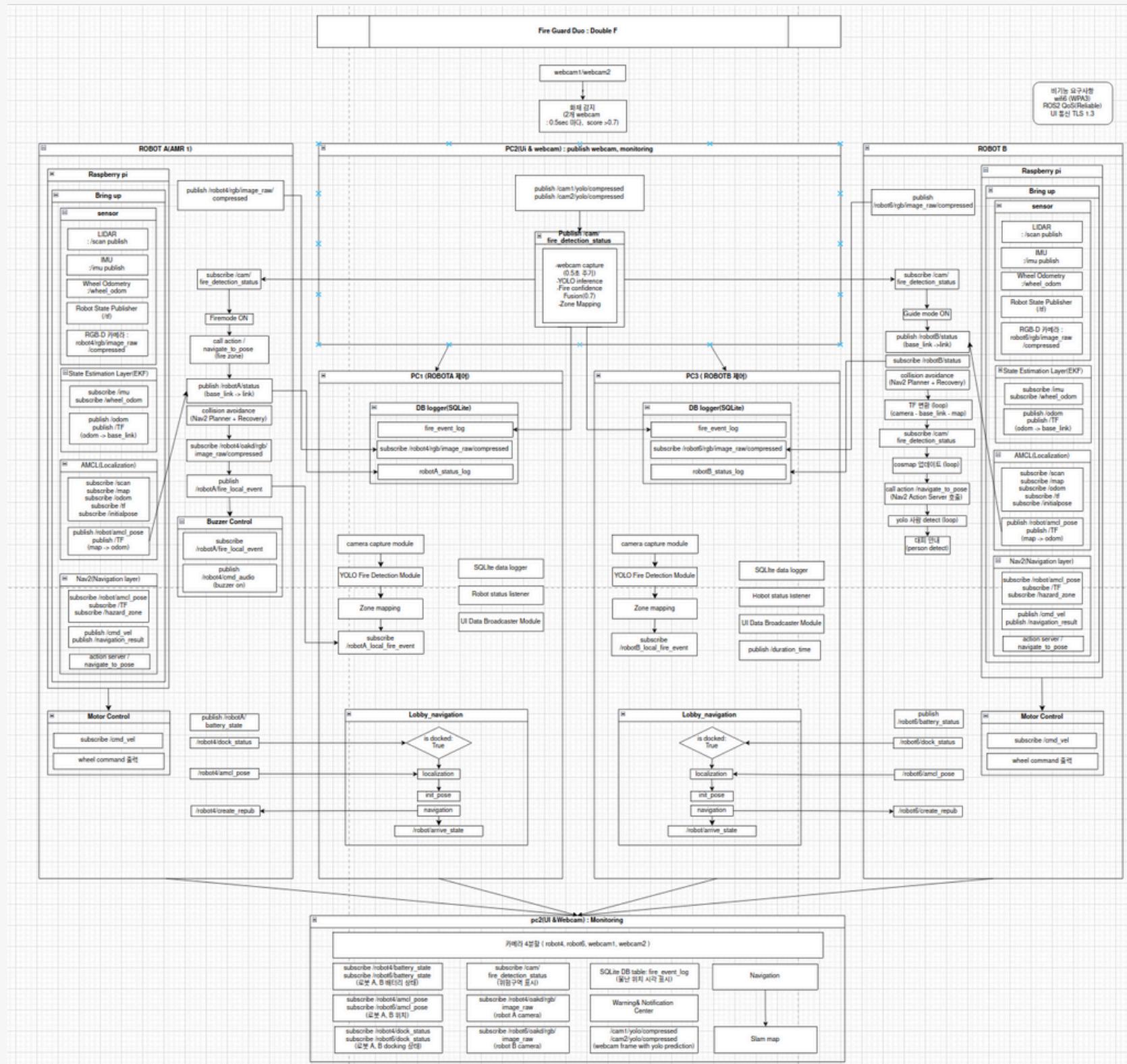
5

모니터링

6

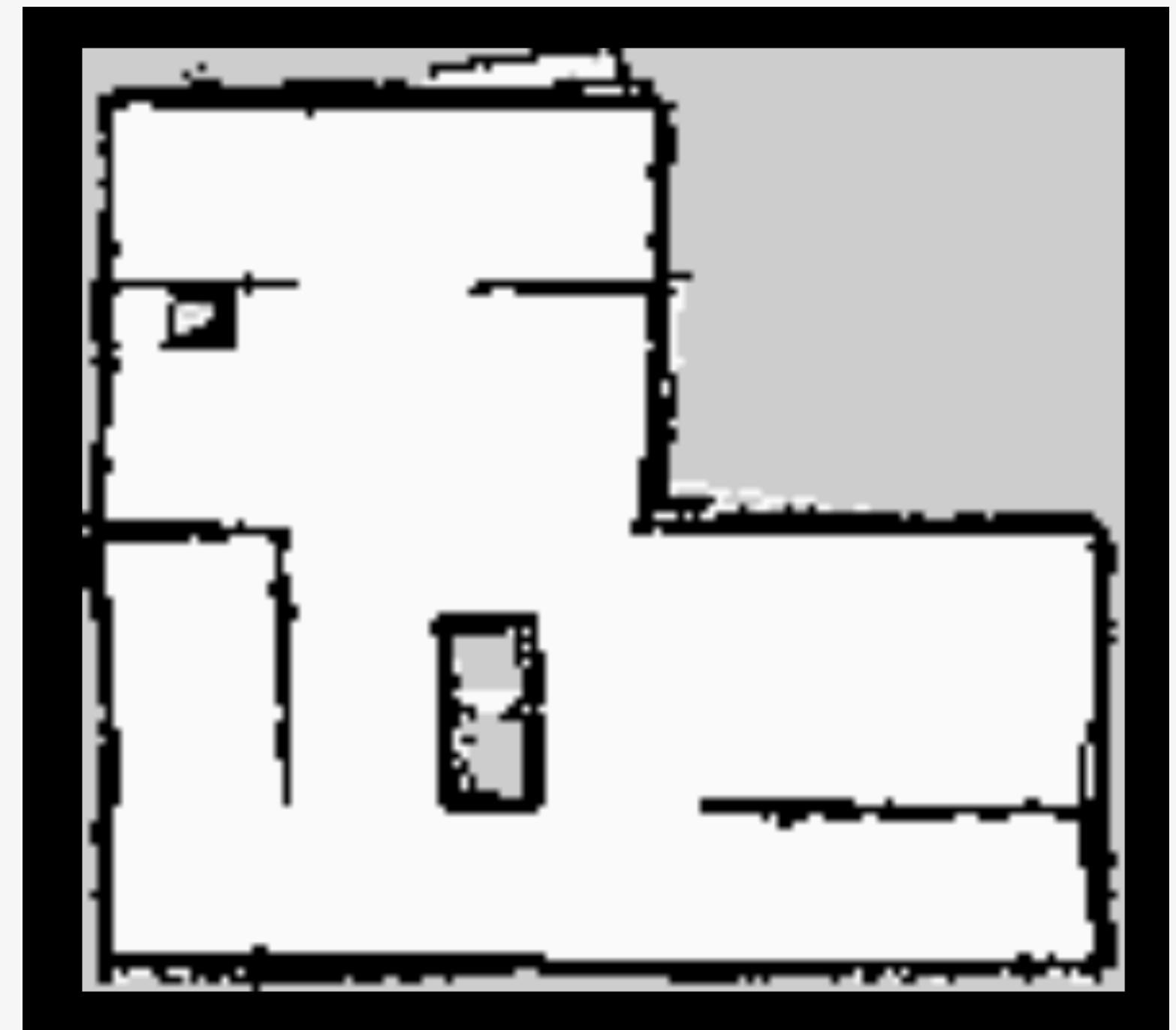
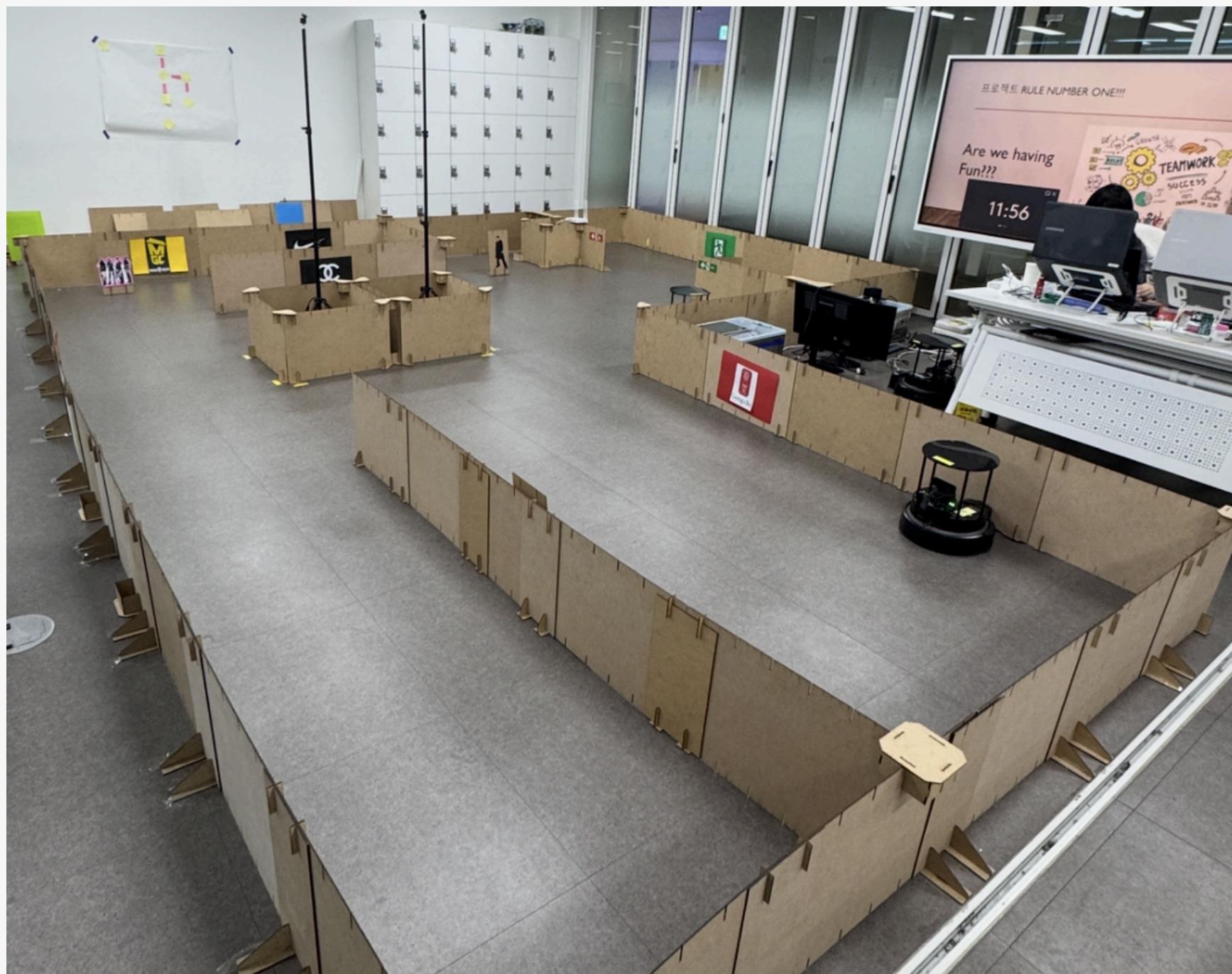
통합

04 프로젝트 개발 목차



05 프로젝트 수행 경과

(1) Map 생성



05 프로젝트 수행 경과

(2) 화재감지 (웹캠)

Fire & Smoke dataset

| | | |
|-------|--------------------|-------|
| hsv_h | 색조 (Hue) | 0.015 |
| hsv_s | 채도 (Saturation) | 0.7 |
| hsv_v | 명도 (Value) | 0.4 |



" Indoor fire & smoke dataset "

train: 3500장

test: 750장

valid: 750장

hsv_h=0.02,
 hsv_s=0.8,
 hsv_v=0.6,

모니터의 불을 인지해야하므로
hsv augmentation 증가

05 프로젝트 수행 경과

(2) 화재감지 (웹캠)

When comparing the confidence scores of YOLOv8 and Detectron2, **Detectron2 comes out to be better in terms of the confidence scores.**

Detectron2 consistently surpasses YOLO in accuracy metrics. On the other hand, YOLO has been observed to detect a larger number of objects in some cases **due to its single-shot detection approach, which allows it to process images faster than Detectron2.**

M. Wadhwa, T. Choudhury, G. Raj and J. C. Patni, "Comparison of YOLOv8 and Detectron2 on Crowd Counting techniques," 2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS), Istanbul, Turkiye, 2023

05 프로젝트 수행 경과

(2) 화재감지 (웹캠)

rt-detr

| 모델 유형 ↓↑ | 사전 훈련된 가중 치 | 지원되는 작업 | 추론 | 검증 | 훈련 | 내보내기 |
|---------------------|----------------|---------|----|----|----|------|
| RT-DETR Large | rtdetr-l.pt | 객체 감지 | ✓ | ✓ | ✓ | ✓ |
| RT-DETR Extra-Large | rtdetr-x.pt | 객체 감지 | ✓ | ✓ | ✓ | ✓ |

Table 7. Performance comparison on the NEU-DET dataset.

| Model | Precision (%) | mAP@0.5 (%) | mAP@0.5:0.95 (%) | GFLOPs | FPS |
|-------------|---------------|-------------|------------------|--------|-------|
| YOLOv5s | 72.14 | 70.87 | 35.02 | 15.8 | 94.5 |
| YOLOv6n | 70.36 | 69.45 | 33.94 | 11.8 | 99.6 |
| YOLOv7 | 74.28 | 71.92 | 37.20 | 13.0 | 96.8 |
| YOLOv8n | 71.45 | 70.38 | 35.71 | 8.1 | 105.3 |
| RT-DETR-R18 | 73.01 | 72.41 | 36.85 | 58.6 | 49.7 |
| YOLOv9s | 76.34 | 73.26 | 37.92 | 26.7 | 65.4 |
| YOLOv10s | 77.18 | 74.08 | 38.36 | 24.4 | 68.2 |
| YOLOv11s | 78.03 | 74.62 | 38.74 | 21.3 | 70.1 |
| Ours | 79.46 | 76.32 | 39.17 | 12.4 | 85.7 |

05 프로젝트 수행 경과

(2) 화재감지 (웹캠)

YOLOv11n

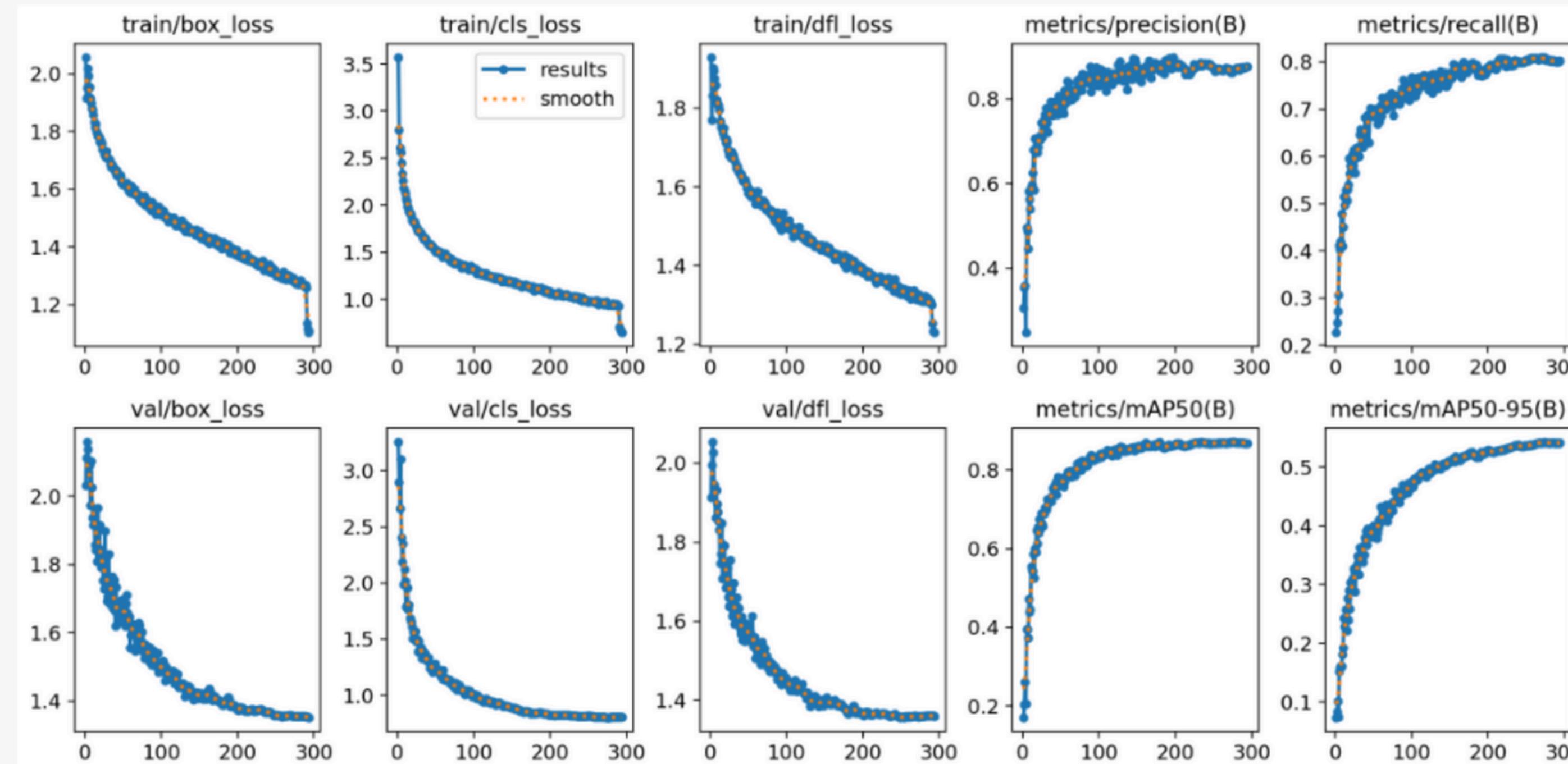
: mAP50-95 = 0.606, inference = (10번) 19 - 31

```
Ultralytics 8.3.232 Python-3.13.5 torch-2.8.0+cu129 CUDA:0 (NVIDIA GeForce RTX 3050 Ti Laptop GPU, 4096MiB)
YOLOv11n summary (fused): 100 layers, 2,582,542 parameters, 0 gradients, 6.3 GFLOPs
val: Fast image access (ping: 0.00.0 ms, read: 319.125.8 MB/s, size: 28.3 KB)
val: Scanning D:\rokey\roskey_turtlebot4\tracking\dataset\valid\labels.cache... 10 images, 0 backgrounds, 0 corrupt: 100% ----- 10/10 13.9Kit/s 0.0s
      Class   Images Instances   Box(P)      R    mAP50    mAP50-95): 100% ----- 1/1 2.4s/it 2.4s
          all       10        19     0.984      1    0.995    0.607
          car        9         9     0.979      1    0.995    0.567
          dummy      10        10     0.99      1    0.995    0.647
Speed: 3.4ms preprocess, 32.3ms inference, 0.0ms loss, 2.9ms postprocess per image
Results saved to D:\rokey\roskey_turtlebot4\tracking\ultralytics_git\runs\detect\val4
mAP@50: 0.995
mAP@50-95: 0.6069901599443608
```

```
save_dir: WindowsPath('D:/rokey/roskey_turtlebot4/tracking/ultralytics_git/runs/detect/val11')
speed: {'preprocess': 3.3363900030963123, 'inference': 19.82459999853745, 'loss': 0.0007400056347250938, 'postprocess': 2.948989998549223} <
```

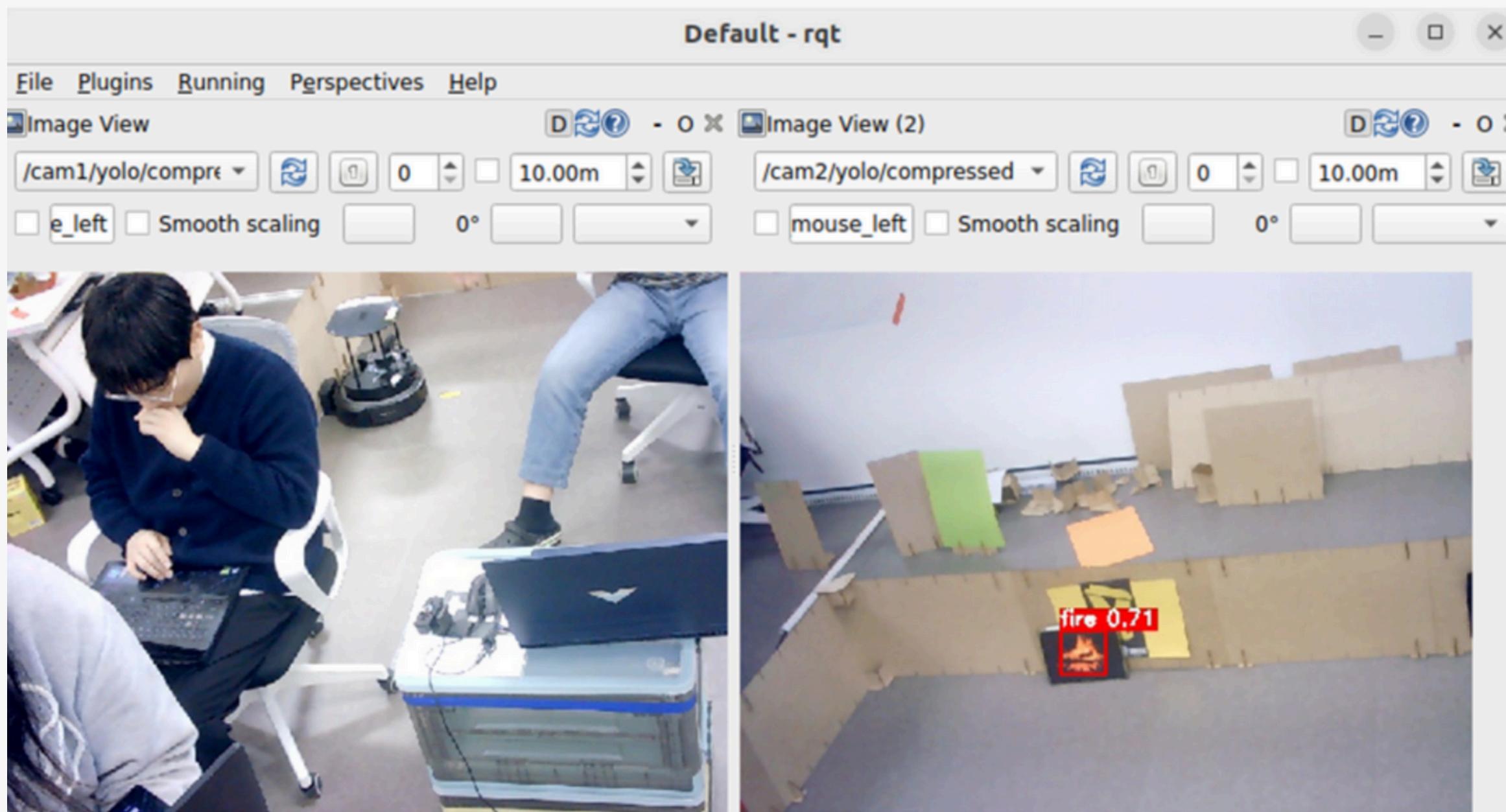
05 프로젝트 수행 경과

(2) 화재감지 (웹캠)



05 프로젝트 수행 경과

(2) 화재감지 (웹캠)



cam1, cam2 detection publish

05 프로젝트 수행 경과

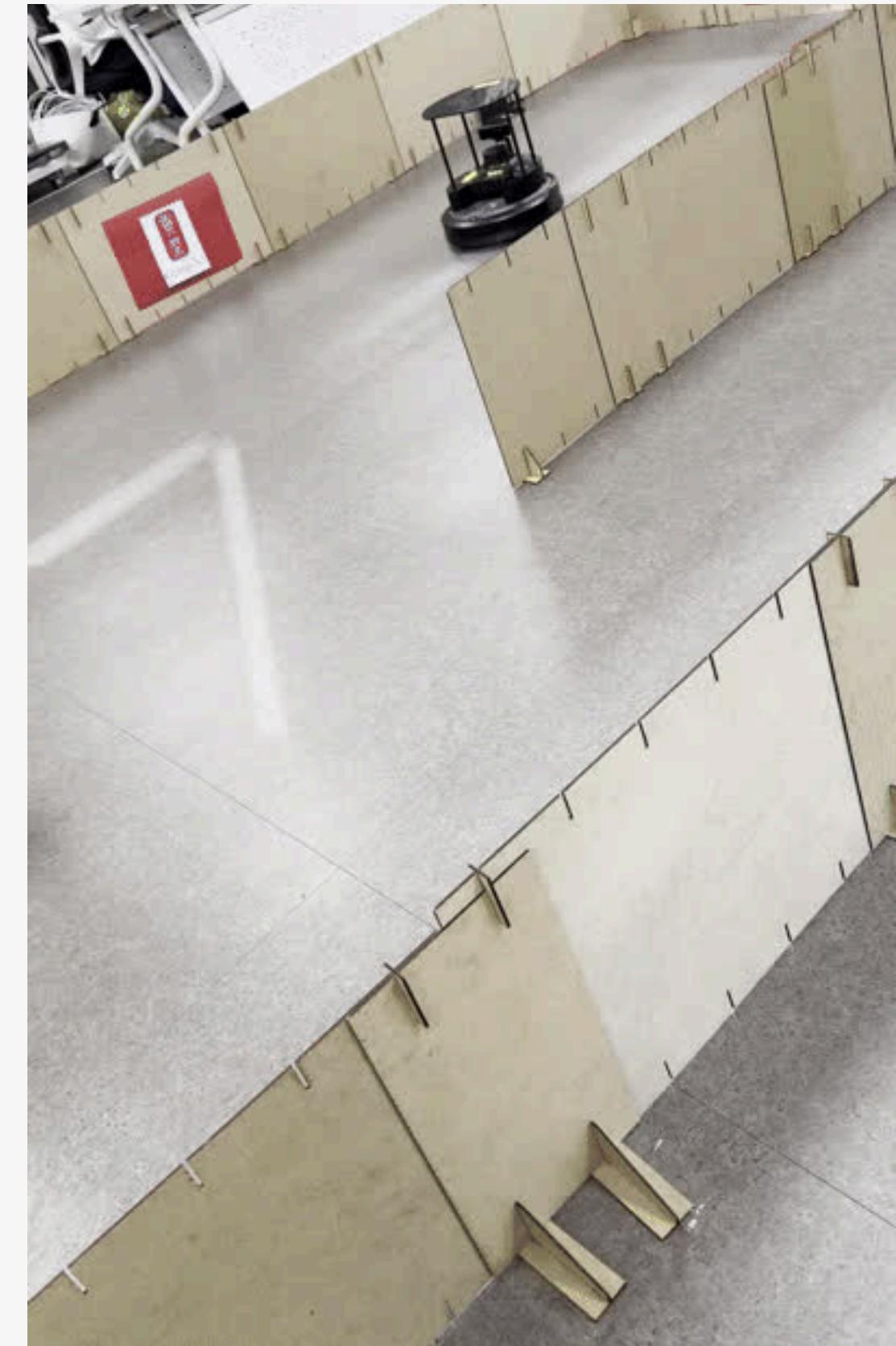
(2) 화재감지 (웹캠)



05 프로젝트 수행 경과

(3) 화재진압 로봇 A

화재경보 발생시
즉시 화재발생 지역으로 이동하여
AMR 카메라로 화재를 2차 검증하는 모습



05 프로젝트 수행 경과

(4) 화재대피 로봇 B의 loop 주행



05 프로젝트 수행 경과

(5) 모니터링

<로봇 A>



상태 패널

화재 감지 상태: 화재 발생

출동 로봇: ID: TB4A / 종류: TB4
상태: Connected
카메라: OK
배터리: 55%

ID: TB4B / 종류: TB4
상태: Connected
카메라: OK
배터리: 20%

전방 장애물 최소 거리
TB4A: 0.64 m
TB4B: 0.64 m

[로그 조회](#)

Transferring data from 127.0.0.1...

<로봇 B>



화면 전환

TB4A 확대

TB4B 확대

CAM1 확대

CAM2 확대

4분할 복귀

SLAM MAP



[로그아웃](#)

로봇 출동 로그

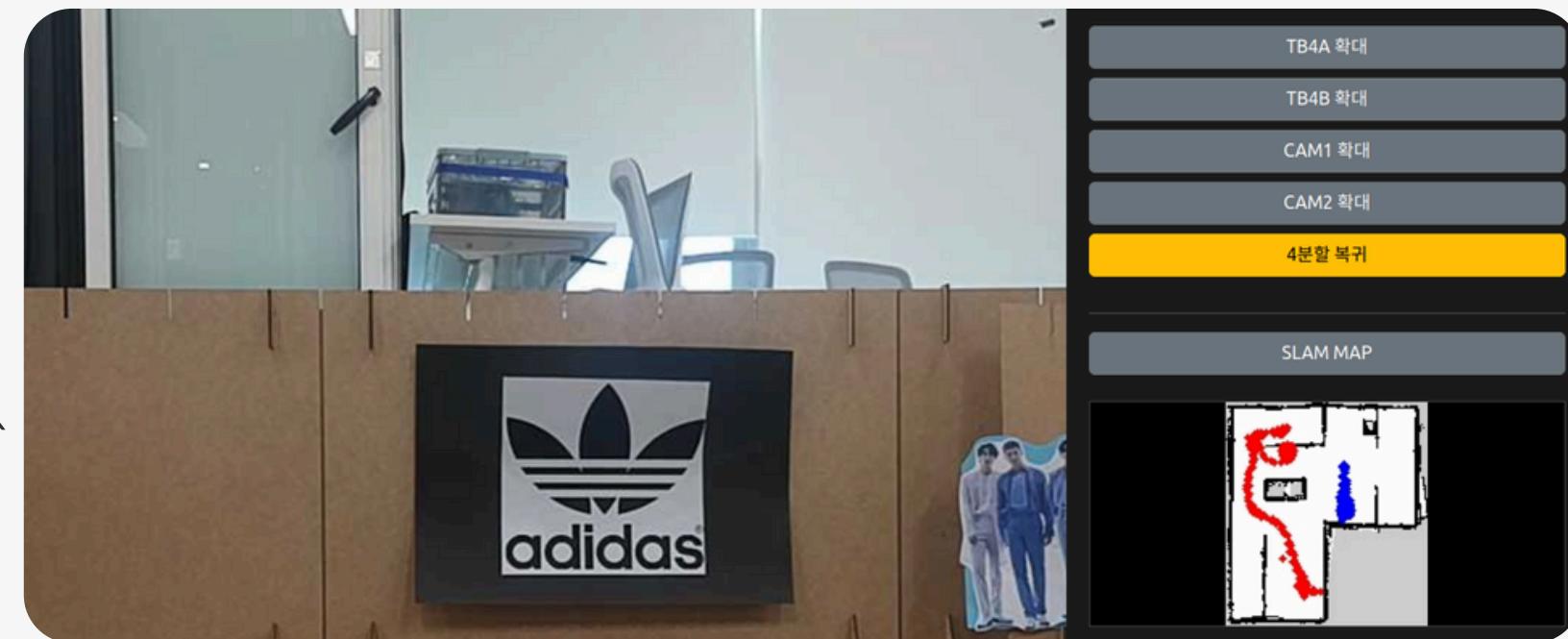
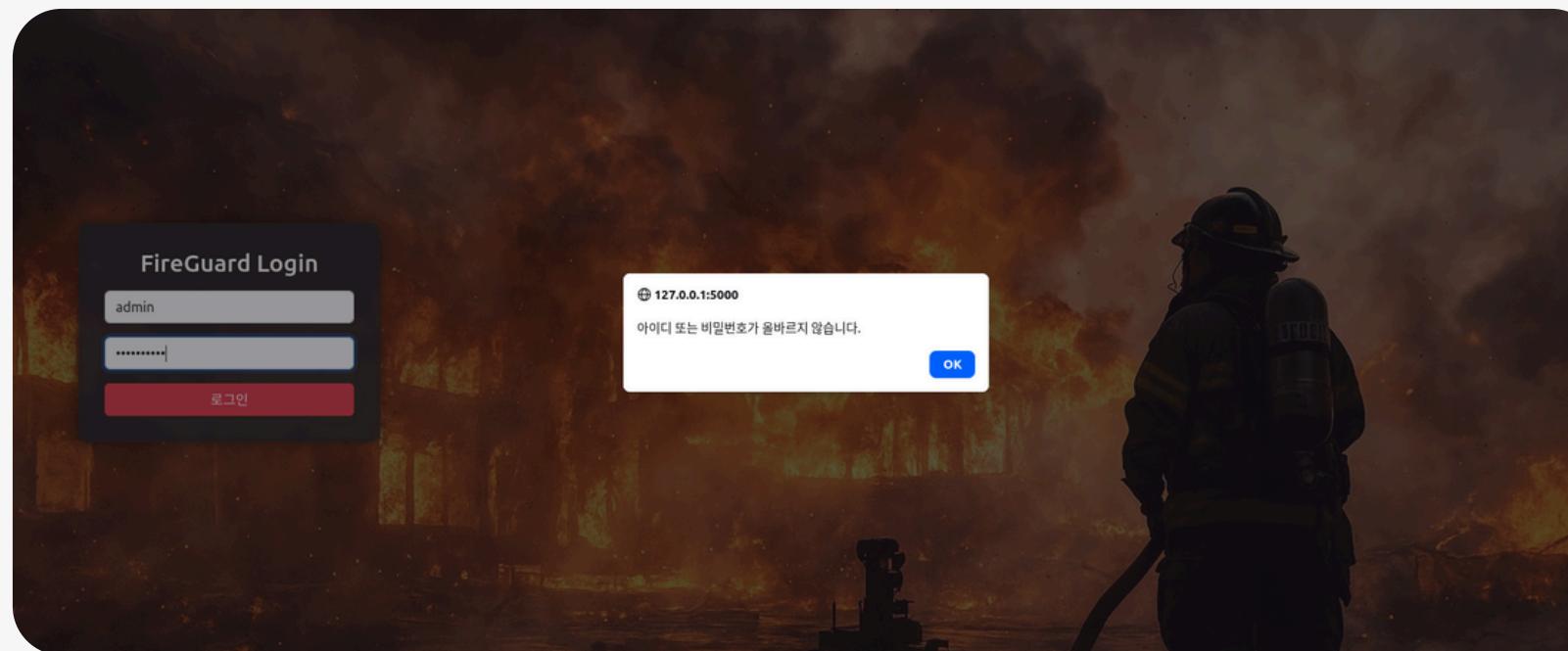
[출동/복귀 기록 보기](#)

<웹캠 1>

<웹캠 2>

05 프로젝트 수행 경과

(5) 모니터링



로봇 출동 로그

출동/복귀 기록 보기

| Robot | 출발 | 도착 | 상태 | 소요 |
|--------|---------------------|----|---------|----|
| robot4 | 2025-12-05 01:12:38 | - | 119신고완료 | - |
| robot6 | 2025-12-05 01:17:38 | - | 119신고완료 | - |

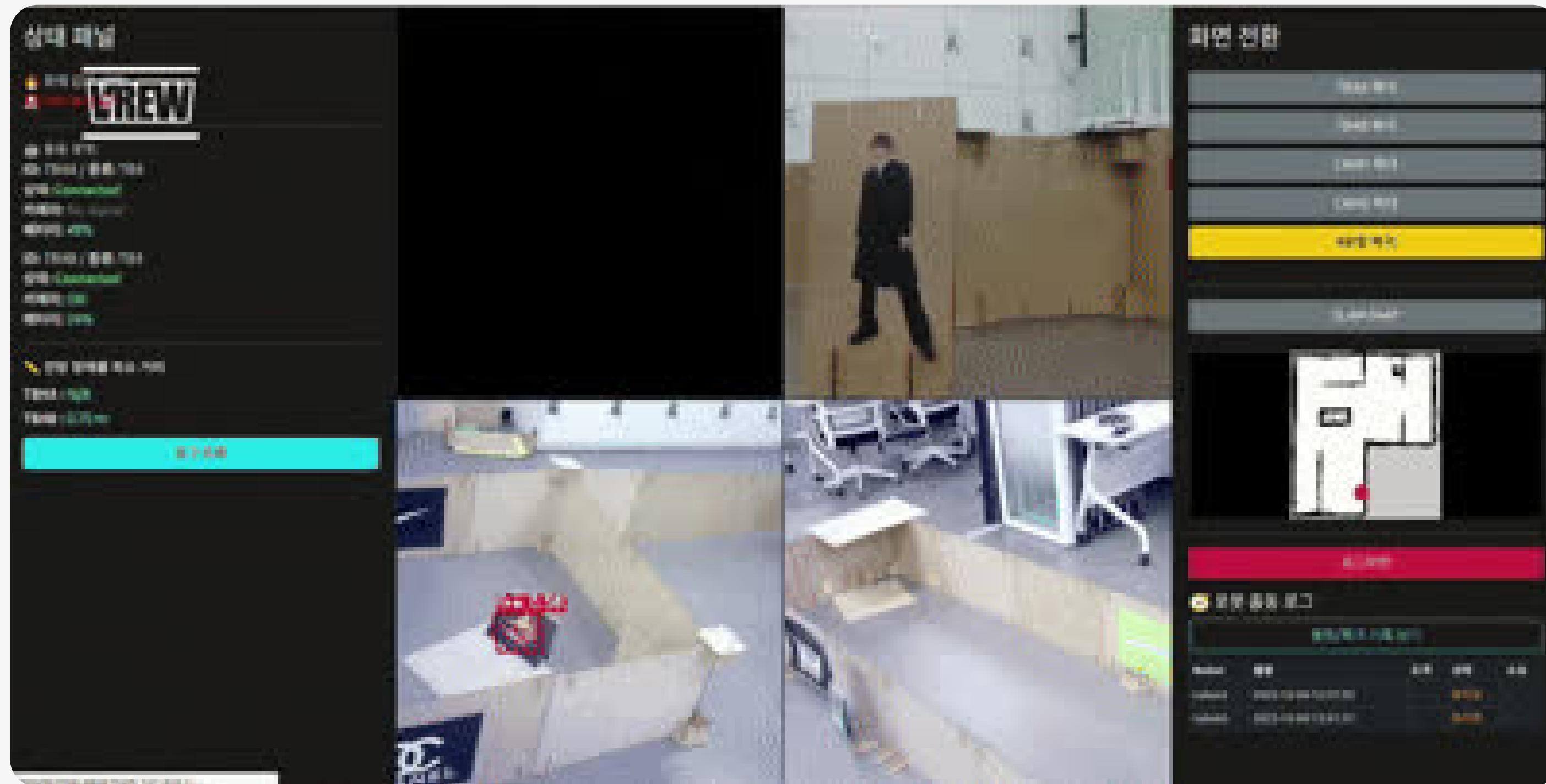
로봇 출동 로그

출동/복귀 기록 보기

| Robot | 출발 | 도착 | 상태 |
|--------|---------------------|---------------------|----|
| robot4 | 2025-12-04 12:41:44 | 2025-12-04 21:44:15 | 완료 |
| robot6 | 2025-12-04 12:41:31 | 2025-12-04 21:43:54 | 완료 |

05 프로젝트 수행 경과

(5) 모니터링



05 프로젝트 수행 경과

(6) 통합

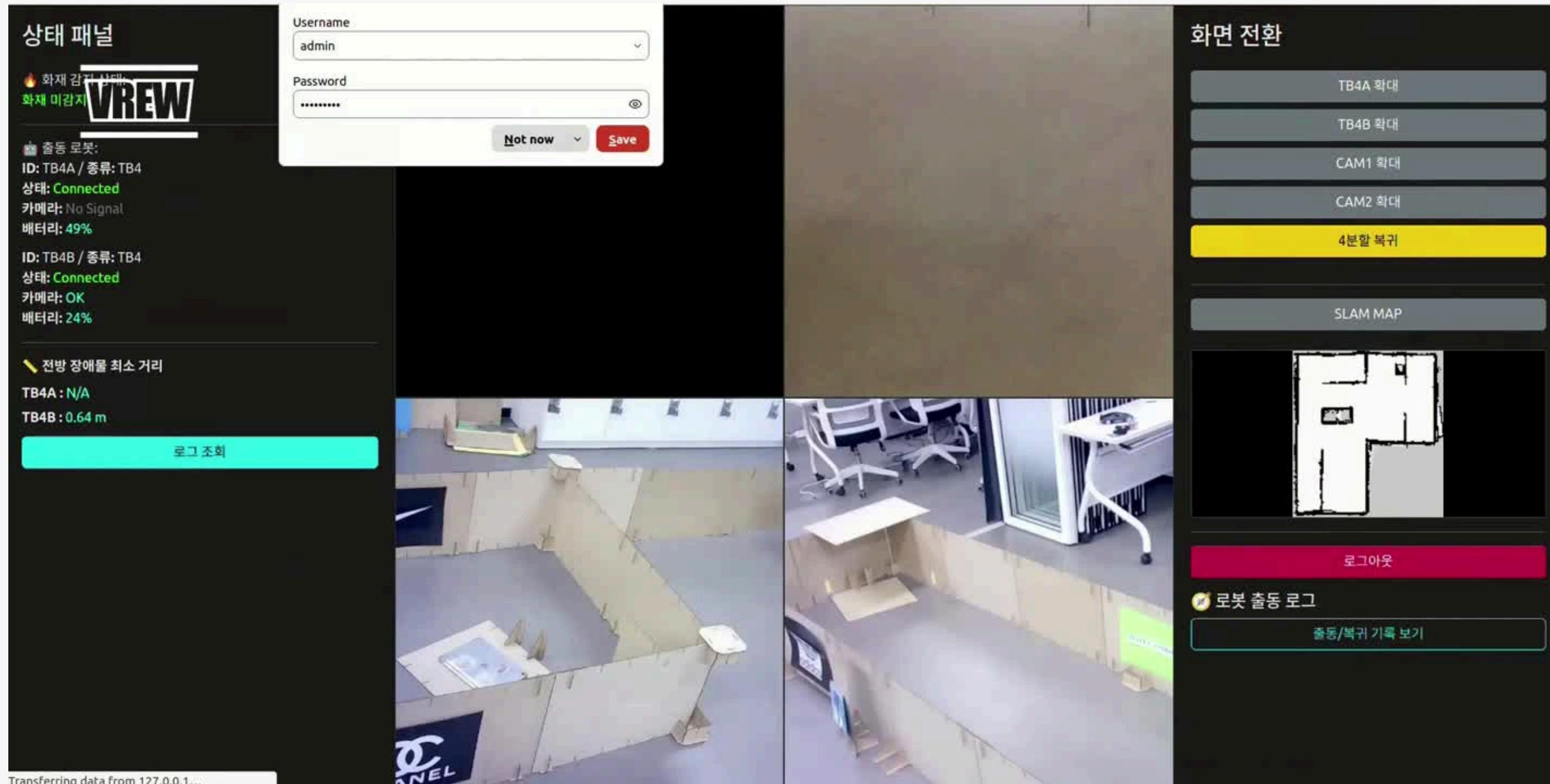


05 프로젝트 수행 경과

(6) 통합



05 프로젝트 수행 경과



<최종 영상>

07 자체 평가 의견

완성도 평가

8 / 10

- 계획했던 핵심 기능 (화재 감지 - 로봇 대응 - 모니터링) 실제로 구현
- 일부 최적화 및 안정화 과정 부족했지만, 전체 시스템의 흐름을 완성도 있게 구성

우리 팀이 잘한 부분 / 아쉬운 점

< 잘한 부분 >

- 체계적인 역할 분담 & 각자 맡은 기능을 끝까지 책임지고 구현
- 모르는 부분은 적극적으로 질문하고 협력

< 아쉬운 점 >

- 장비의 반복적인 오류로 인해 개발 일정이 지연

추후 개선점 및 보완할 점

- 화재 감지 및 사람 감지 알고리즘 정확도 향상 (조명 반사, 오탐 개선)
- 시스템 전체를 자동화된 테스트 환경에서 반복적으로 검증할 수 있는 구조 추가

느낀 점 및 경험한 성과

- YOLO 기반 화재 감지, ROS2 DDS 통신을 실제 로봇과 연동하며, 실시간 이벤트 기반 협동 시스템 구축 과정을 경험
- Nav2 경로 계획·SLAM·TF 구조를 다루며 다중 로봇의 자율주행 파이프라인과 로봇 시스템 아키텍쳐 흐름을 체계적으로 이해

08 기술적 이슈

Webcam

Technical Issue

- 모니터 화재 감지 환경
- 확장으로 인한 피부 오탐

Trouble Shooting

- HSV Agumentation 확장
- HSV Post Processing 으로 피부 반사영역 제거

Future Enhancements

- 불꽃 깜빡임 패턴 분석 추가

Monitoring

Technical Issue

- 각 토픽 전송시 서버 과부화
- 화재 감지 상태 확인 방법

Trouble Shooting

- Launch.py로 한번에 전송
- DB의 message 체크 방식

Future Enhancements

- 실제 119 전화연결
- Voice를 로봇에서 출력
- 화재 위치 표시

08 기술적 이슈 → AMR

Robot

Technical Issue

- 테스트 대상 외 주변 인물까지 함께 인식
- 사람 감지 시 순찰 중단

Trouble Shooting

- 상단 영역은 원거리 객체로 설정하여 하단 영역만 Detection
- startToPose()에서 goToPose()전환
비동기 Action 기반으로 토픽 콜백의 실시간 처리

Future Enhancements

- Depth + Bounding Box 기반 다양한 객체 거리 필터링 적용
- AMR에 스피커를 추가하여 관제실 음성 출력 추가

질문 및 답변

궁금한 사항을 질문해 주세요.

Thank you

감사합니다.