시스템 설계 문서 (SDD)

프로젝트 제목: **<YOLO 기반 화재 감지와 협동 로봇 Fire Guard AMR>**
버전: 1.2
날짜: 2025.12.05
팀명: Double F (F1&F3)

1. 개요

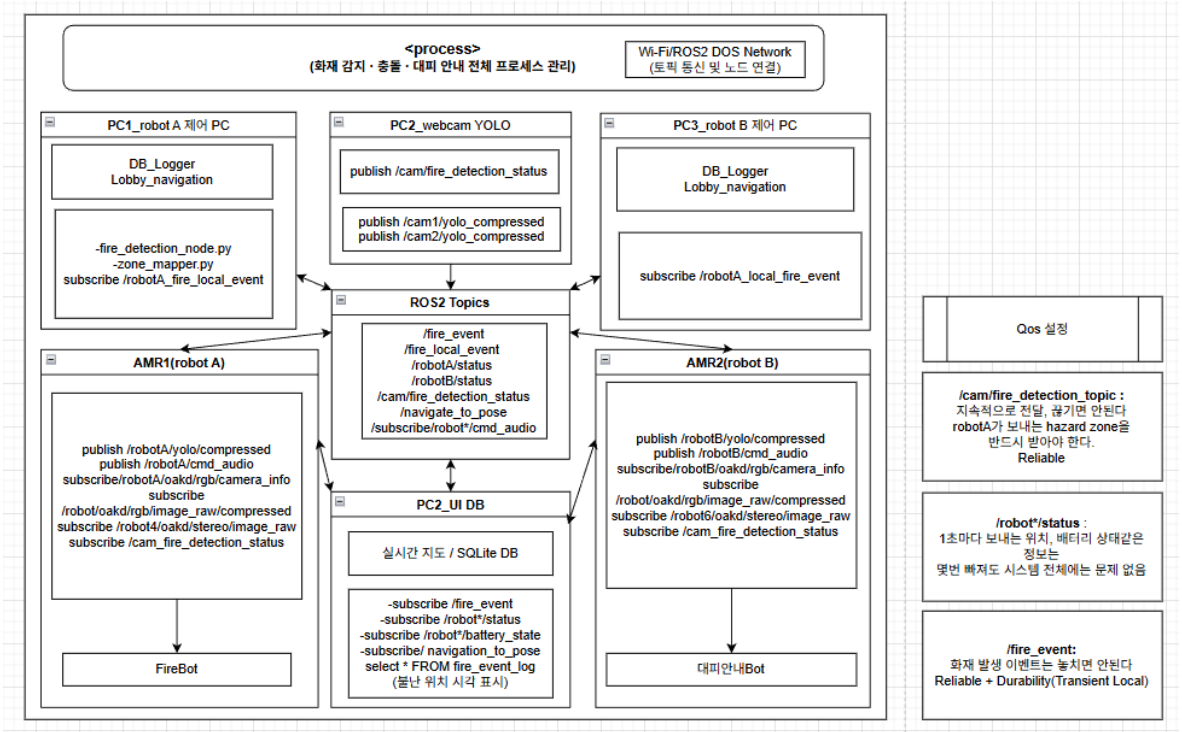
FireGuard Duo 시스템은 실내 환경에서 화재를 조기에 감지하고 신속한 초기 대응을 수행하기 위해 설계된 AI 기반 AMR(Autonomous Mobile Robot) 협동 솔루션입니다.

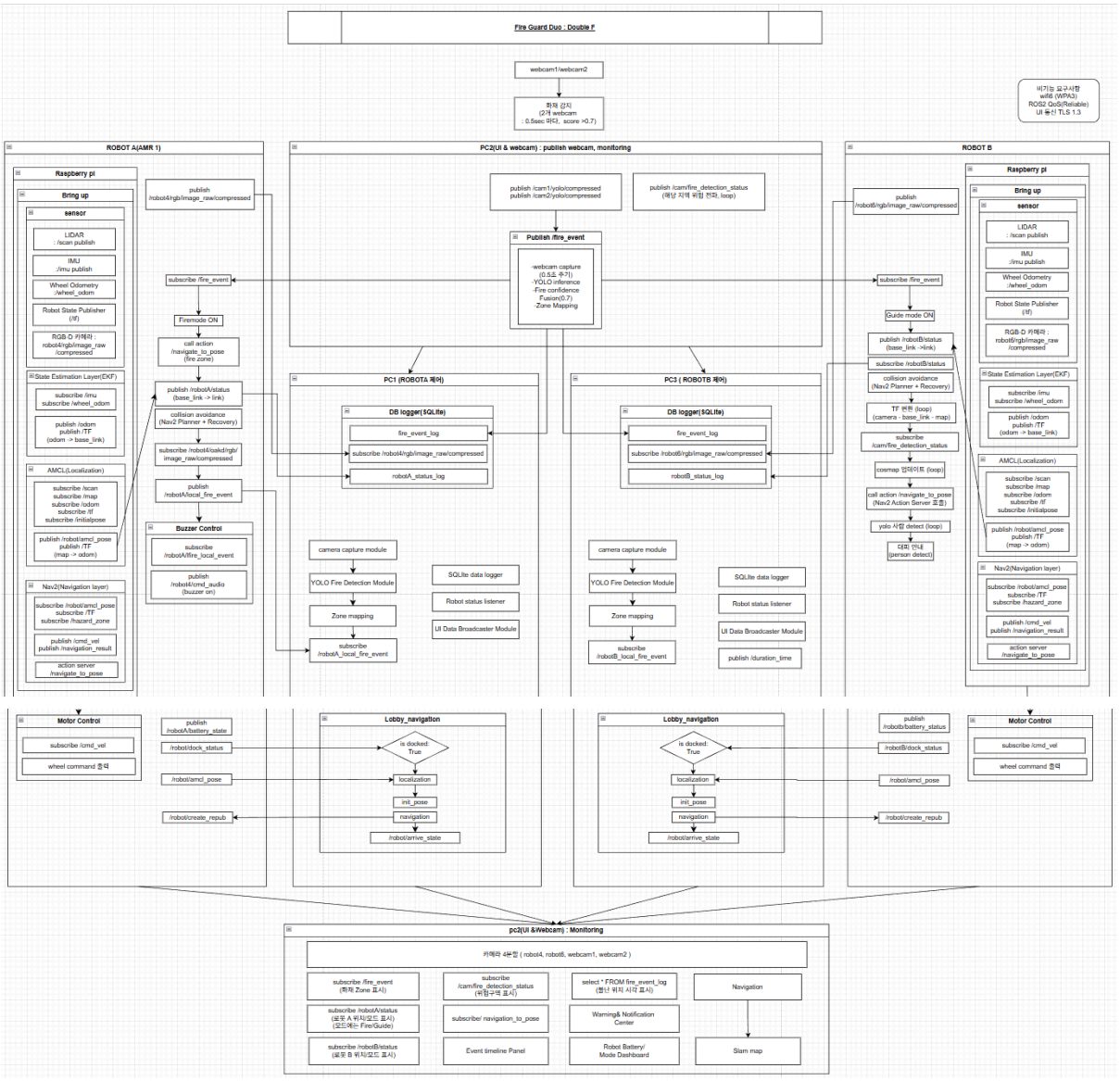
본 시스템은 YOLO 기반 실시간 화재·연기 감지 기술과 두 대의 TurtleBot4 AMR을 결합하여, 위험 구역 경보·진입 제한·대피 안내 등 다양한 대응 기능을 자동화합니다. 화재 발생 시 중앙 서버에서 이벤트를 처리하고 두 로봇 간 협력 동작을 조정함으로써, 초기 3~5분 내의 골든타임 확보, 인명 피해 최소화, 지능형 안전 관리를 목표로 합니다.

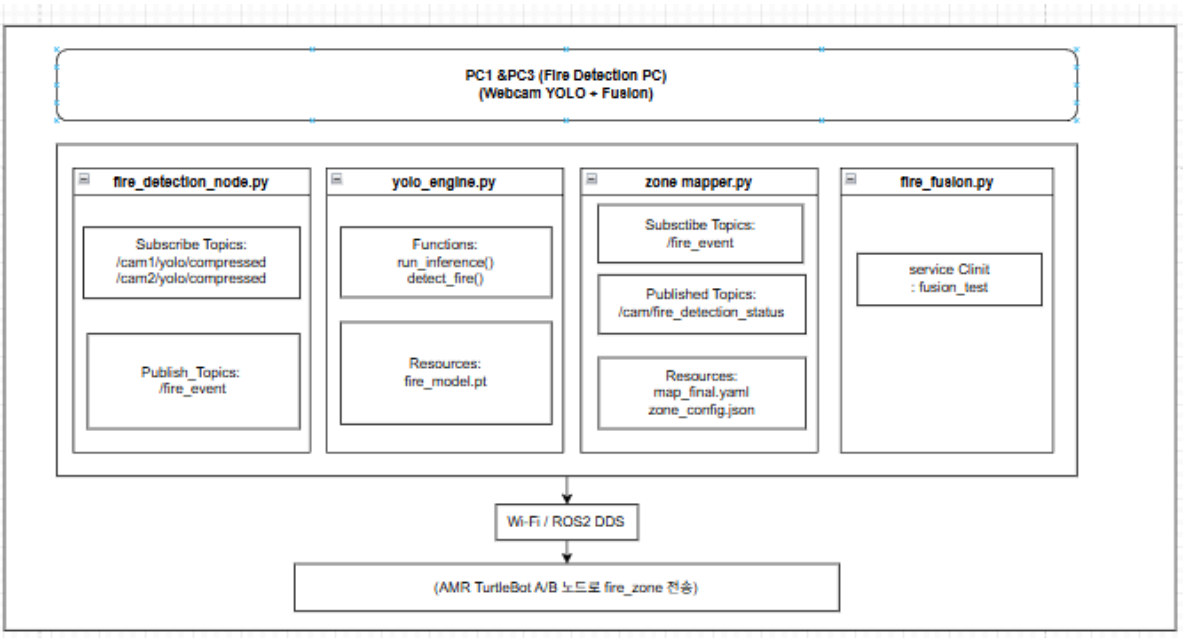
2. 시스템 아키텍처

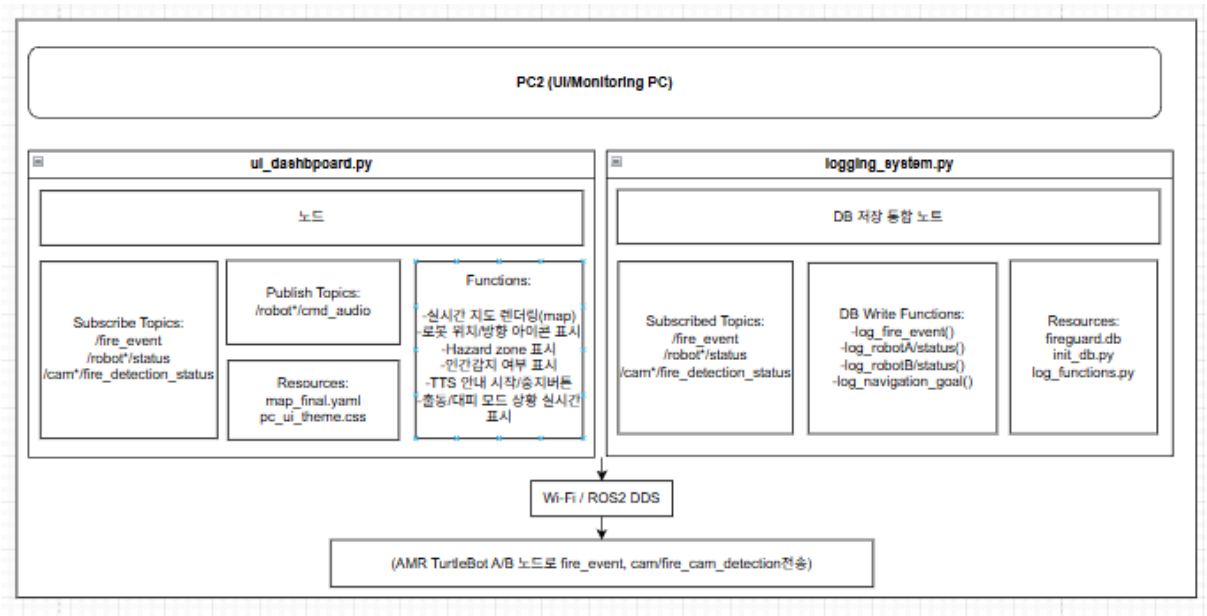
이 시스템은 2대의 AMR TurtleBot4로 구성되며, 화재 감지·데이터 처리·경로 계획·대피 안내 등 주요 기능은 중앙 서버 PC에서 처리됩니다. 서버는 YOLO 기반 영상 분석을 통해 WebCam으로부터 화재 여부를 실시간 판단하고, ROS2 DDS 통신을 통해 각 AMR에 화재 이벤트 및 동작 모드를 전송합니다. 각 TurtleBot4는 온보드 Raspberry Pi 및 내장 센서를 활용해 자율 주행을 수행하며, 서버로부터 수신한 명령에 따라 화재 지역 경보 출력(robotA), 사람 인식 및 대피 안내(robotB) 등의 기능을 실행합니다. AMR은 Wi-Fi를 통해 중앙 사용자 대시보드(UI PC)와 연결되며, 로봇의 위치·배터리 상태·동작 모드 등의 정보는 UI 화면에 실시간으로 시각화 되어 시스템 운영자가 즉시 파악할 수 있습니다.

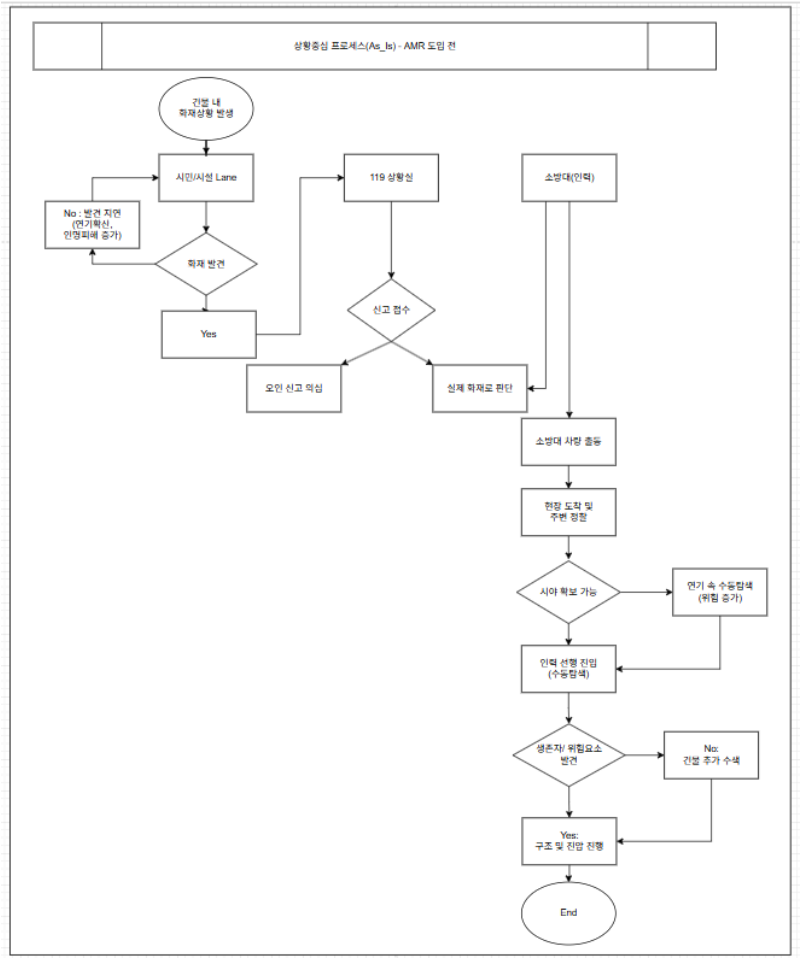
2.1 고레벨 아키텍처 다이어그램

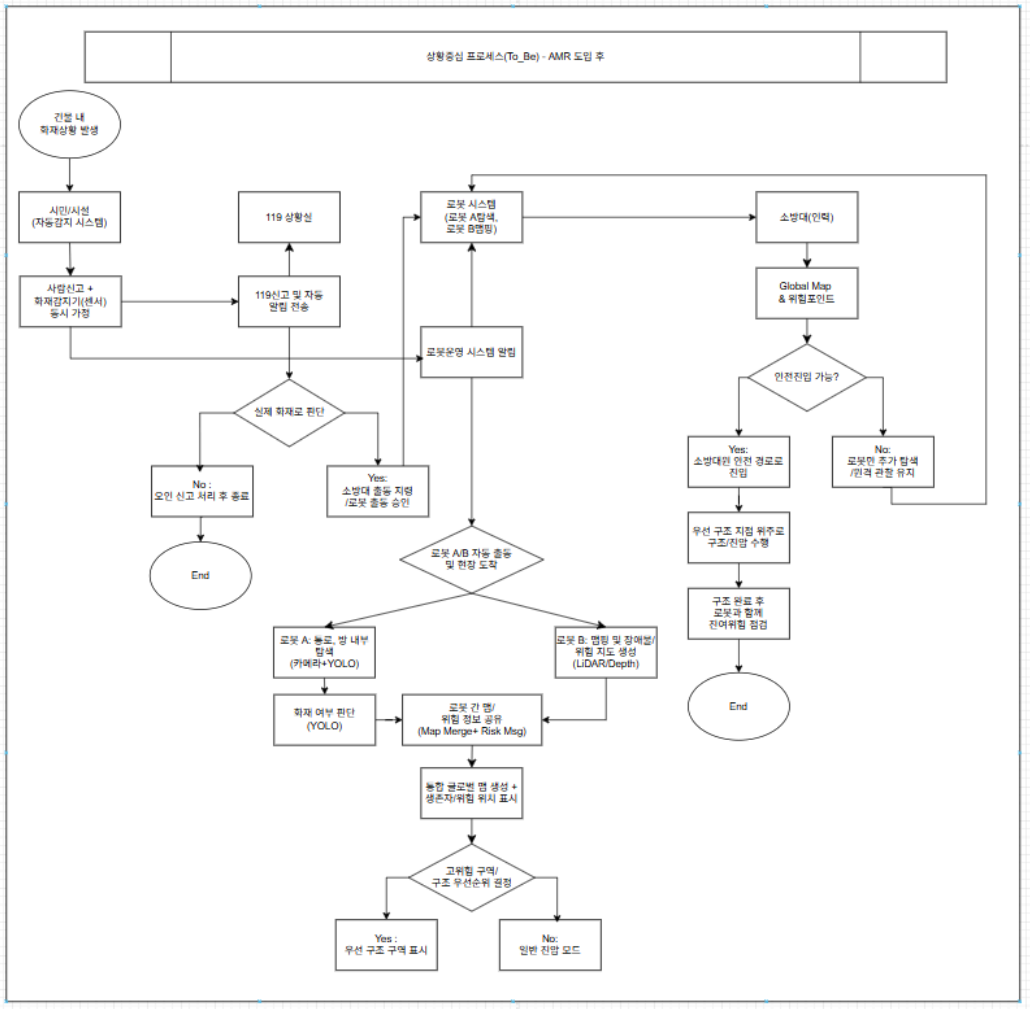



Fire Guard Duo : Double F
webcam1/webcam2
화재 감지
(2개 webcam
: 0.5sec 마다, score >0.7)
비기능 요구사항
wifi6 (WPA3)
ROS2 QoS(Reliable)
UI 통신 TLS 1.3
ROBOT A(AMR 1)
Raspberry pi
Bring up
sensor
LIDAR
: /scan publish
IMU
:/imu publish
Wheel Odometry
:/wheel_odom
Robot State Publisher
(tf)
RGB-D 카메라
robot4/rgb/image_raw
/compressed
State Estimation Layer(EKF)
subscribe /imu
subscribe /wheel_odom
publish /odom
publish /TF
(odom -> base_link)
AMCL(Localization)
subscribe /scan
subscribe /map
subscribe /odom
subscribe /tf
subscribe /initialpose
publish /robot/amcl_pose
publish /TF
(map -> odom)
Nav2(Navigation layer)
subscribe /robot/amcl_pose
subscribe /TF
subscribe /hazard_zone
publish /cmd_vel
publish /navigation_result
action server
/navigate_to_pose
publish
/robot4/rgb/image_raw/compressed
subscribe /fire_event
Firemode ON
call action
/navigate_to_pose
(fire zone)
publish /robotA/status
(base_link -> link)
collision avoidance
(Nav2 Planner + Recovery)
subscribe /robot4/oakd/rgb/
image_raw/compressed
publish
/robotA/local_fire_event
Buzzer Control
subscribe
/robotA/fire_local_event
publish
/robot4/cmd_audio
(buzzer on)
PC2(UI & webcam) : publish webcam, monitoring
publish /cam1/yolo/compressed
publish /cam2/yolo/compressed
publish /cam/fire_detection_status
(해당 지역 위험 전파, loop)
Publish /fire_event
·webcam capture
(0.5초 주기)
·YOLO inference
·Fire confidence
Fusion(0.7)
·Zone Mapping
PC1 (ROBOTA 제어)
DB logger(SQLite)
fire_event_log
subscribe /robot4/rgb/image_raw/compressed
robotA_status_log
camera capture module
YOLO Fire Detection Module
Zone mapping
subscribe
/robotA_local_fire_event
SQLite data logger
Robot status listener
UI Data Broadcaster Module
PC3 (ROBOTB 제어)
DB logger(SQLite)
fire_event_log
subscribe /robot6/rgb/image_raw/compressed
robotB_status_log
camera capture module
YOLO Fire Detection Module
Zone mapping
subscribe
/robotB_local_fire_event
SQLite data logger
Robot status listener
UI Data Broadcaster Module
publish /duration_time
ROBOT B
Raspberry pi
Bring up
sensor
LIDAR
: /scan publish
IMU
:/imu publish
Wheel Odometry
:/wheel_odom
Robot State Publisher
(tf)
RGB-D 카메라 :
robot6/rgb/image_raw
/compressed
State Estimation Layer(EKF)
subscribe /imu
subscribe /wheel_odom
publish /odom
publish /TF
(odom -> base_link)
AMCL(Localization)
subscribe /scan
subscribe /map
subscribe /odom
subscribe /tf
subscribe /initialpose
publish /robot/amcl_pose
publish /TF
(map -> odom)
Nav2(Navigation layer)
subscribe /robot/amcl_pose
subscribe /TF
subscribe /hazard_zone
publish /cmd_vel
publish /navigation_result
action server
/navigate_to_pose
publish
/robot6/rgb/image_raw/compressed
subscribe /fire_event
Guide mode ON
publish /robotB/status
(base_link -Hink)
subscribe /robotB/status
collision avoidance
(Nav2 Planner + Recovery)
TF 변환 (loop)
(camera - base_link - map)
subscribe
/cam/fire_detection_status
cosmap 업데이트 (loop)
call action /navigate_to_pose
(Nav2 Action Server 호출)
yolo 사람 detect (loop)
대피 안내
(person detect)
Motor Control
subscribe /cmd_vel
wheel command 출력
publish
/robotA/battery_state
/robot/dock_status
/robot/amcl_pose
/robot/create_repub
Lobby_navigation
is docked:
True
localization
init_pose
navigation
/robot/arrive_state
Lobby_navigation
is docked:
True
localization
init_pose
navigation
/robot/arrive_state
publish
/robotb/battery_state
/robotB/dock_status
/robot/amcl_pose
/robot/create_repub
Motor Control
subscribe /cmd_vel
wheel command 출력
pc2(UI &Webcam) : Monitoring
카메라 4분할 (robot4, robot6, webcam1, webcam2)
subscribe /fire_event
(화재 Zone 표시)
subscribe /robotA/status
(로봇 A 위치/모드 표시)
(모드에는 Fire/Guide)
subscribe /robotB/status
(로봇 B 위치/모드 표시)
subscribe
/cam/fire_detection_status
(위험구역 표시)
subscribe/ navigation_to_pose
Event timeline Panel
select * FROM fire_event_log
(최난 위치 시각 표시)
Warning& Notification
Center
Robot Battery/
Mode Dashboard
Navigation
Slam map

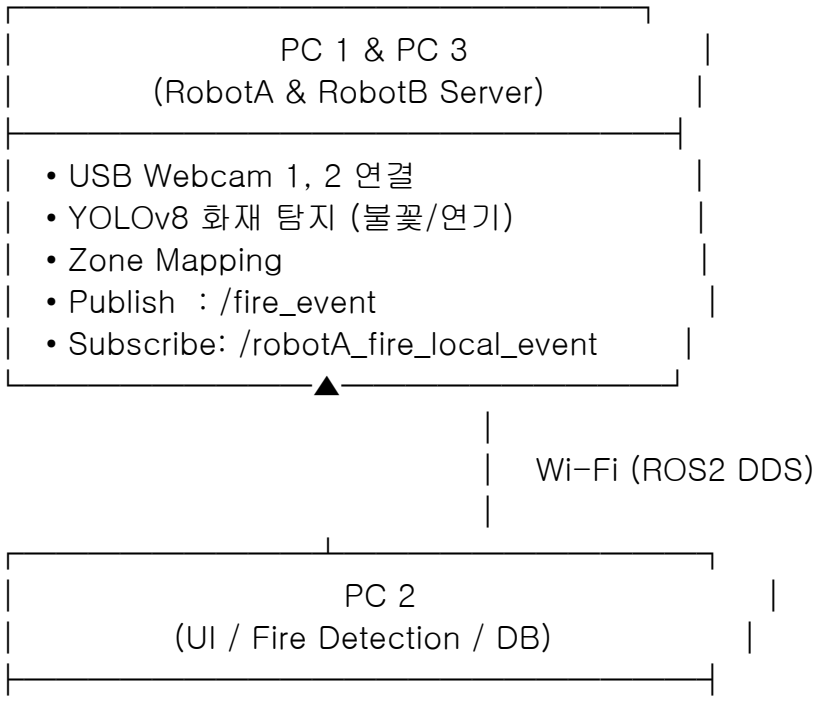

PC1 &PC3 (Fire Detection PC)
(Webcam YOLO + Fusion)
fire_detection_node.py
Subscribe Topics:
/cam1/yolo/compressed
/cam2/yolo/compressed
Publish_Topics:
/fire_event
yolo_engine.py
Functions:
run_inference()
detect_fire()
Resources:
fire_model.pt
zone mapper.py
Subsctibe Topics:
/fire_event
Published Topics:
/cam/fire_detection_status
Resources:
map_final.yaml
zone_config.json
fire_fusion.py
service Clinit
: fusion_test
Wi-Fi / ROS2 DDS
(AMR TurtleBot A/B 노드로 fire_zone 전송)

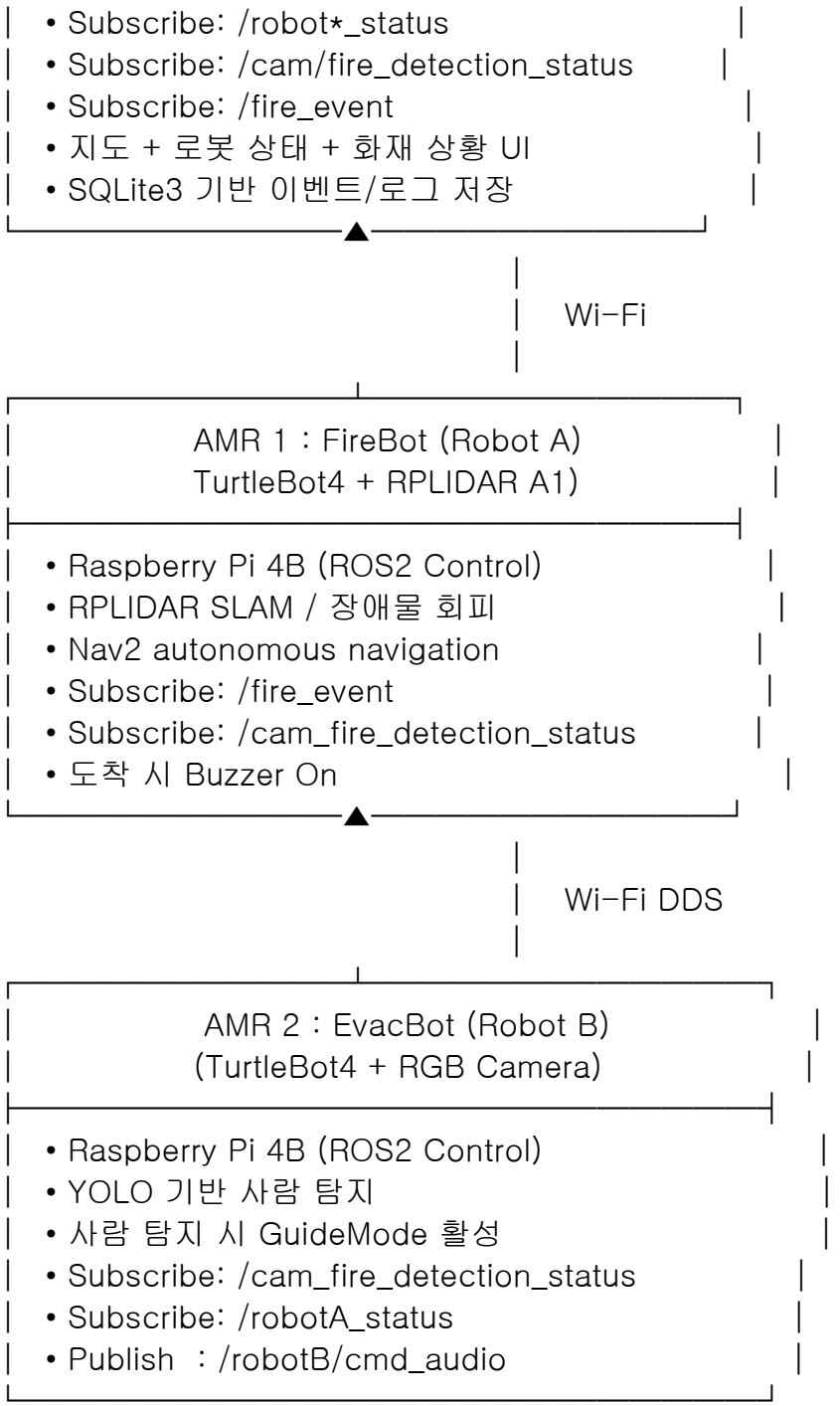

PC2 (UI/Monitoring PC)
ui_dashbpoard.py
노드
Subscribe Topics:
/fire_event
/robot*/status
/cam*/fire_detection_status
Publish Topics:
/robot*/cmd_audio
Resources:
map_final.yaml
pc_ui_theme.css
Functions:
-실시간 지도 렌더링(map)
-로봇 위치/방향 아이콘 표시
-Hazard zone 표시
-인간감지 여부 표시
-TTS 안내 시작/송지버튼
출동/대피 모드 상황 실시간
표시
logging_system.py
DB 저장 통합 노드
Subscribed Topics:
/fire_event
/robot*/status
/cam*/fire_detection_status
DB Write Functions:
-log_fire_event()
-log_robotA/status()
-log_robotB/status()
-log_navigation_goal()
Resources:
fireguard.db
init_db.py
log_functions.py
Wi-Fi / ROS2 DDS
(AMR TurtleBot A/B 노드로 fire_event, cam/fire_cam_detection전송)


상황중심 프로세스(As_Is) - AMR 도입 전
긴보 내
화재상황 발생
시민/시설 Lane
No : 발견 지연
(연기확산,
인명피해 증가)
화재 발견
Yes
119 상황실
신고 접수
오인 신고 의심
실제 화재로 판단
소방대(인력)
소방대 사항 출동
현장 도착 및
주변 정찰
시야 확보 가능
연기 속 수동탐색
(위험 증가)
인력 선행 진입
(수동탐색)
생존자/ 위험요소
발견
No:
긴보 추가 수색
Yes:
구조 및 진압 진행
End



2.2 상황중심 프로세스 (As-it, To-be)




• Subscribe: /robot*_status
• Subscribe: /cam/fire_detection_status
• Subscribe: /fire_event
• 지도 + 로봇 상태 + 화재 상황 UI
• SQLite3 기반 이벤트/로그 저장
Wi-Fi
AMR 1 : FireBot (Robot A)
TurtleBot4 + RPLIDAR A1)
• Raspberry Pi 4B (ROS2 Control)
• RPLIDAR SLAM / 장애물 회피
• Nav2 autonomous navigation
• Subscribe: /fire_event
• Subscribe: /cam_fire_detection_status
• 도착 시 Buzzer On
Wi-Fi DDS
AMR 2 : EvacBot (Robot B)
(TurtleBot4 + RGB Camera)
• Raspberry Pi 4B (ROS2 Control)
• YOLO 기반 사람 탐지
• 사람 탐지 시 GuideMode 활성
• Subscribe: /cam_fire_detection_status
• Subscribe: /robotA_status
• Publish : /robotB/cmd_audio

3. 구성 요소 설계

3.1 하드웨어 구성 요소

1. PC (화재 감지 및 중앙 제어 서버)

- 운영 체제 : Ubuntu 22.04
- 카메라 : 실내 화재 감지를 위한 USB WebCam 2대
- 네트워크: AMR과의 안정적인 Wi-Fi 6 기반 ROS2 DDS 통신 지원
- GPU : YOLO 추론 성능 향상을 위한 NVIDIA RTX GPU 권장

2. 자율 이동 로봇 (AMR)
   - TurtleBot4 × 2 (robotA, robotB)
   - 프로세서: Raspberry Pi 4B (온보드 AI 처리 및 데이터 처리)
   - 운영 체제: Ubuntu 22.04 및 ROS2 (로봇 제어)
   - 센서:
     - RPLIDAR: SLAM 및 공간 인식, 장애물 감지
     - 카메라(OAK-D Pro): 동적객체와 정적객체 탐지용
     - 배터리: 2 시간 30 분 ~ 4 시간

3. 충전 스테이션
   - 수동 개입 없이 AMR 의 충전을 위한 자율 도킹 스테이션

3.2 소프트웨어 구성 요소

1. PC 소프트웨어 (화재 감지 서버 & UI 서버)
   - Python3: 화재 감지, 이벤트 처리, 데이터 로깅 등 시스템 핵심 로직 구현
   - ROS2 Humble: AMR(robotA·robotB)과의 통신 및 토픽 기반 메시지 전달
   - OpenCV: WebCam 영상 캡처 및 전처리를 위한 기본 이미지 처리 라이브러리
   - YOLO (Ultralytics): 불꽃·연기 감지를 위한 실시간 객체 인식 엔진
   - Flask / FastAPI: 실시간 모니터링 UI 및 대시보드 구현 (PC2에서 동작)
   - SQLite3: 화재 이벤트·로봇 상태 정보를 저장하는 로컬 데이터베이스

2. AMR 소프트웨어
   - ROS2 Humble: 네비게이션, SLAM, 로봇 제어 및 서버-PC와의 메시지 통신을 위한 핵심 프레임워크
   - Navigation2 (Nav2): 글로벌/로컬 경로 계획, 장애물 회피, 목표 지점

도달을 위한 AMR 이동 제어
- Python3: 로봇 모드 전환(FireMode/GuideMode), TTS 제어, 상태 Publish 등 노드 구성
- OpenCV (선택): robotB가 사람 탐지를 수행할 경우 On-board 영상 처리 지원
- gTTS 또는 pyttsx3: robotB 대피 안내 음성 생성 및 출력
- irobot_create_msgs: TurtleBot4의 버저 제어 및 로봇 기본 기능 연동

4. 데이터 흐름 설계

4.1 데이터 수집

• 카메라(WebCam 1 & WebCam 2)

- 두 개의 USB WebCam을 이용하여 실내 환경을 0.5초 간격으로 캡처
- YOLO 기반 불꽃 및 연기 감지를 위해 640×480 해상도로
- 프레임 전처리
- 감지된 이미지 정보는 PC(화재 감지 서버)로 전달되어 화재 신뢰도(Confidence) 계산에 활용됨

• AMR 센서:
- AMR은 내장 LiDAR 및 IMU를 사용하여 실내 자율 네비게이션 수행
- robotA는 화재 구역 도착 후 버저 경고를 위한 위치 정보 수집
- robotB는 사람 감지를 위한 온보드 카메라(Optional) 사용 가능
- 두 로봇 모두 ROS2를 통해 실시간 위치, 배터리 상태, 모드 정보를 PC로 전송

4.2 데이터 처리 및 범위 및 논리

• yolo_detect.py (카메라 영상 처리 모듈)

- WebCam에서 수집된 raw 프레임을 OpenCV 포맷으로 변환
- 영상 밝기·노이즈 등 기본 전처리 수행
- 처리된 영상은 YOLO 추론 엔진(YoloEngine.py)의 입력으로 사용

• fire_detection_node.py (화재 감지 관리 모듈)

- YOLO 기반 불꽃/연기 감지 모델 수행
- WebCam1과 WebCam2에서 입력된 감지 점수를 합산
- 신뢰도(score ≥ 0.7)일 경우 화재 이벤트 생성 후 /fire_event 토픽으로 Publish

- 동시에 화재 위치를 ZoneMapper.py로 전달하여 건물 내 Zone ID로 매핑
- 감지된 Confidence는 /fire_confidence로 주기적으로 Publish

• ZoneMapper.py (Zone 매핑 모듈)
- 감지된 화재 픽셀 위치를 사전에 정의된 실내 지도로 매핑
- 각 WebCam의 설치 위치·FOV(시야각) 정보를 기반으로
- 화재 위치를 Zone ID 형태로 변환
- robotA와 robotB가 (Nav2 costmap)에 등록하여
- 충돌·진입 제한 기능 수행 가능

• amr_controller.py (AMR 이동 처리 모듈)
- /fire_event를 수신 후 robotA는 Nav2 Action Server를 사용하여
- 화재 Zone으로 이동할 경로를 계산
- robotB는 사람 탐지 시 해당 위치를 TF2를 사용해
- camera → base_link → map 프레임으로 변환
- 변환된 좌표를 기반으로 안전한 Exit 방향으로
- /navigate_to_pose 액션 호출
- 두 로봇은 이동 중 Nav2 Planner + Recovery Behavior로
- 충돌 회피 및 우회 경로 자동 수행

4.3 데이터 저장

• AMR 로컬 저장: 각 AMR(robotA , robotB)은 로컬 시스템에서 다음 데이터를 일정 기간 버퍼링할 수 있음 :

- 이동 중 실시간 위치(x, y, θ)
- 현재 모드(FireMode / GuideMode)
- 배터리 잔량
- 장애물 감지 및 경로 재계획 정보

• PC 로그 저장(SQLite 기반 중앙 저장소):

- YOLO 기반 화재 감지 이력 (timestamp / webcam_id / confidence)
- 화재 발생 이벤트 로그 (zone_id, 신뢰도, 시간)
- Robot A/B 상태 정보 (위치, 각도, 배터리, 모드)
- 위험 구역(hazard zone) 갱신 이력

4.4 경고 및 알림 전달

• fire_detection_node는 화재 신뢰도(confidence ≥ 0.7) 판단 시

- /fire_event 토픽을 AMR 두 대(robotA , robotB)에 즉시 전송
- robotA은 화재 지점 도착 시 /cmd_buzzer 또는 irobot_create_msgs 기반 버저 신호를 출력
- robotB은 GuideMode 진입 후 /cmd_tts 명령을 주기적(10초 간격)으로

실행하여 '이쪽으로 대피하세요' 음성 안내 송출
- UI PC는 /fire_event, /robotA_status, /robotB_status, /fire_zone을 구독하여 사용자 모니터링 화면에 실시간 알림을 표시
- ROS2 DDS 통신 기반으로 구성되어, Wi-Fi 환경에서도 안정적인 이벤트·경고 메시지 전송 가능

5. 상세 설계

5.1 AMR 네비게이션 및 사람 탐지 / 화재 대응 동작

• 네비게이션 (SLAM + Nav2 기반)
- TurtleBot4는 RPLIDAR와 IMU 데이터를 기반으로
- SLAM(동시 위치추정 및 지도작성) 을 수행
- Navigation2(Nav2) 프레임워크를 활용해 전역 경로 계획(Global Planner), 장애물 회피(Local Planner), 경로 재계산을 수행하여 지연 없이 화재 지점 또는 대피 경로를 따라 이동함
- robotA는 화재 Zone으로 이동하여 안전거리 경고 및 접근 제한 역할을 수행
- robotB는 위험 구역을 우회하여 사람을 출구 방향으로 유도함

• 사람 탐지 및 대피 안내(GuideMode)
- robotB는 설치된 카메라 또는 입력된 위치 정보를 기반으로 YOLO 기반 사람(person) 감지를 수행
- 사람 위치는 TF2 변환을 통해 camera → base_link → map 프레임으로 변환
- 변환된 좌표를 바탕으로 Nav2에 /navigate_to_pose 액션 요청 → 사람이 있는 방향으로 이동 후 → 최종적으로 출구(Exit) 방향으로 유도
- 탐지된 사람이 사라지면 로봇은 안전 탐색 모드 또는 대기 모드로 자동 전환

• 통신 프로토콜:
- 모든 노드 간 통신은 ROS2 DDS(Fast-DDS) 기반
- AMR은 Wi-Fi6를 통해 PC1(화재 감지 서버) 및 PC2(UI 서버)에 실시간 상태를 전송
- /robotA_status, /robotB_status, /fire_event, /fire_zone 등 주요 토픽은 QoS Reliability 모드에 따라 전달 보장
- Wi-fi 환경은 끊김/지연이 발생하기 쉬워서 Qos를 제대로 설정해야 한다. (실시간+안정성+안전성 보장 위함)
- Ros2 Qos는 주고받는 메시지 성격에 따라 통신방식을 골라야 한다.
- /fire_event 화재 발생이벤트는 절대 놓치면 안되기 때문에 Reliable+Durability 필요
- /fire_zone은 지속적으로 전달해야 한다. robotB가 위험지역을

우회하려면 robotA가 보내는 hazard zone을 반드시 받아야 하므로 Reliable사용
- /robotA/status, /robotB/status에서 1초마다 보내는 위치, 배터리 정보는 몇번 빠져도 시스템 전체에는 문제가 없어 Best_effort 사용
- /fire_confidence는 실시간성이 정확도보다 중요하기 때문에 Bests_effort 사용한다.

• 작업 종료 및 상태 판단
- robotA는 화재 Zone 도착 및 Boozer 작동 후
- FireMode 유지 또는 “Safe” 상태 복귀
- robotB는 사람 인식 종료 또는 출구 안내 완료 시
- GuideMode FSM(Finite State Machine)을 종료하고 Idle 모드 복귀
- UI에서는 다음 상태가 실시간으로 보여짐 (Fire Detected, robotA Arrived, robotB Guiding, Safe Mode)
- 화재 신뢰도가 일정 threshold 이하로 떨어지면
- 시스템은 자동으로 종료 이벤트를 기록하고 로그 DB에 저장됨

5.2 Tracking 노드
- Tracking 모듈은 화재 감지 서버에서 전달된 화재 Zone ID 및 감지 신뢰도(fire confidence) 를 기반으로 FireBot의 목표 위치(goal)를 결정함
- robotA가 화재 Zone으로 이동하는 중 Nav2의 local planner가 주기적으로 장애물 및 경로 상태를 갱신하여 최신 목표점으로 재계산함
- 내부적으로 로봇 위치, hazard zone(위험 구역), 충돌 위험을 고려하여 필요 시 동적으로 경로 업데이트
- robotA가 지정된 Zone에 도착하면 Tracking Node는 /fire_arrival 메시지를 publish하여 UI 및 robotB에 전달
- 최종 도착 상태는 /fire_tracking_success 로 송출되며, robotA는 버저를 작동하여 “위험 구역” 알림을 수행

5.3 Detection Pipeline
- WebCam1, WebCam2에서 0.5초 주기로 수집된 영상이 ImageProcessor.py로 전달됨
- ImageProcessor는 프레임 전처리 후 YOLO 엔진(YoloEngine.py)의 입력으로 제공
- FireDetectionManager.py는 YOLO 기반 Flame/Smoke 감지를 수행하고, confidence score 계산, zone 맵핑 수행, score >=0.7일 때 , /fire_event publish

6. 보안 설계
• 데이터 암호화:
- PC ↔ AMR 간 모든 ROS2 DDS 통신은 Wi-Fi6 + WPA3 환경에서 이루어지며 필요 시 TLS 1.3 기반 보안 통신 적용 가능

• 접근 제어:
- UI 대시보드는 관리자/사용자 권한 기반 로그인 시스템(RBAC)

적용
- 화재 감지 서버는 인증된 내부 네트워크에서만 접근 가능

• 로그보안:
- SQLite에 저장되는 화재 이벤트·로봇 상태 로그는 AES-256 기반 파일 암호화 옵션 적용 가능
- 로그 파일은 관리자 권한으로만 열람 가능

7. 성능 요구사항

• 반응 시간
- 화재 발생 시 WebCam → YOLO 감지 → /fire_event Publish까지 1초 이내 반영되어야 함

• 로봇 출동 시간:
- robotA가 화재 Zone으로 이동하는 시간: 약 3~10초 (거리/장애물 배치에 따라 변동)

• robotB 안내 반응:
- /fire_event 수신 후 GuideMode 전환까지 0.5초 이하, 사람 감지 후 TTS 출력은 10초 간격 유지

• 배터리 지속시간:
- 실측 기준 약 2.5~4 시간 연속 운영 가능

• 확장성:
- ROS2 기반 구조로 로봇 수를 4대 이상으로 확장 가능, W
- wi-Fi namespace 및 ROS2 domain 분리로 다중 로봇 확장 용이

• 다른 장치 도입 여부:
- ROS2 기반 구조로 추가 센서 및 노드 유입이 유연함

8. 오류 처리 및 복구

• 네트워크 오류:

- Wi-Fi 끊김 시 AMR은 마지막 정상 명령을 유지한 채 안전 상태로 전환
- 연결이 복구되면 ROS2 DDS의 자동 재연결 기능을 통해 동작 재개

• 하드웨어 오류:

- 센서(RPLIDAR/IMU) 이상 감지 시 PC UI에 즉시 오류 표시
- AMR 자가진단 모듈이 주기적으로 상태 점검 수행
- 장애 발생 시 robotB는 즉시 정지, robotA는 경고음 출력 유지

• 배터리 부족:
    - /robotA_status, /robotB_status 통해 UI에 표시
    - 배터리 임계값(20% 이하) 도달 시 자동으로 Docking Station으로 복귀

9. 테스트 및 검증

1. 단위 테스트:
    - ROS2 노드(fire_detection, robotA_controller, robotB_controller, UI node) 각각에 대해 기능 단위 테스트 수행
    - YOLO 추론 정확성 테스트 (불꽃/연기 데이터셋)
    - CameraReader, ZoneMapper, DB 로깅 기능(unit test) 개별 검증
    - AMR 내장 센서(LiDAR, IMU) 데이터 수신 정상 여부 확인
2. 통합 테스트:
    - WebCam → YOLO 감지 → /fire_event Publish → robotA/robotB 동작 전체 파이프라인 테스트
    - robotA가 화재 Zone으로 정확히 이동하는지 Nav2 경로 계획 검증
    - robotB가 사람을 감지 후 대피 안내로 정상적으로 전환되는지 확인
    - 모든 노드 간 ROS2 DDS 통신이 끊김 없이 동작하는지 검증
3. 사용자 승인 테스트(UAT): 관리자(UI 운영자)와 함께
    - 화재 감지 알림 확인
    - UI 표시(로봇 위치, hazard zone, fire score) 검증
    - 로봇 자동 대응 동작(FireMode/GuideMode) 검증
    - 실제 실내 환경에서 장애물, 조명 변화 등 비정형 조건에서 테스트 수행
4. 현장 테스트:
    - WebCam을 통한 Flame/Smoke 감지 테스트
    - FireBot 현장 도달 및 경보음 출력 테스트
    - robotB 대피 안내(TTS) 기능 테스트
    - 다양한 거리·조도·연기량에 따른 감지 정확도 수집
    - Wi-Fi 환경에서 데이터 지연 및 패킷 손실 검증

10. 배포 및 유지보수 계획

• 배포 단계:
    - 서버 설치 :
        A. PC1(화재 감지 서버)에 ROS2 Humble
        B. YOLO 환경 (Ultralytics)
        C. SQLite3 구성

- UI 구성 :
    A. PC2(UI PC)에 Flask/FastAPI 기반 대시보드 설치,
    B. /fire_event /robot_status /fire_zone 구독하도록 설정
- AMR 설정 :
    A. TurtleBot4(robotA, robotB)에 ROS2 네트워크 설정 및 Nav2 환경 구성
    B. wi-Fi6 기반 통신 확인
- 최종 테스트 후 현장 배포 :
    A. 초기 네비게이션 맵 로딩
    B. 화재 이벤트에 대한 양 로봇의 실제 반응 검증

• 유지보수 일정:
- 매월 소프트웨어 업데이트
    A. YOLO 모델 업데이트
    B. 보안 패치(ROS2 보안 설정)
    C. UI 개선
- 정기 하드웨어 점검:
    A. AMR 배터리 점검
    B. LiDAR 센서 청소 및 캘리브레이션
    C. WebCam 렌즈 상태 점검
- 로그 분석 기반 유지보수:
    A. SQLite DB에 저장된 화재 감지 기록·로봇 이동 기록 분석
    B. 감지 false positive/negative 개선

11. 부록

• 라이브러리 및 종속성: ROS2, OpenCV, YOLO, Flask, SQLite3 등 사용된 소프트웨어 종속성 목록
• 용어 사전:
- YOLO: 실시간 객체 탐지 모델로 Flame/Smoke 감지에 사용
- SLAM: Simultaneous Localization and Mapping – 로봇이 이동하면서 지도 작성 및 위치 추정
- Nav2: ROS2 기반 경로 계획 및 장애물 회피 프레임워크
- FireMode: 화재 이벤트 수신 후 FireBot이 화재 Zone으로 출동하는 모드
- GuideMode: robotB가 사람을 탐지 후 대피 안내를 수행하는 모드
- Hazard Zone: robotA가 Publish하는 위험 구역 정보 (robotB는 이를 회피함)

• 참고 문헌:
- ROS2 공식 문서
- TurtleBot4 Technical Guide
- Ultralytics YOLO Documentation
- Nav2 Behavior Tree & Planner Docs

12. 버그 수정

- 2025.11.30(Sun) … Fire video monitor 빛 반사시 화재감지(사람의 피부가 빛에 반사되면 화재 감지로 인식하는 문제)의 어려움을 HSV post-processing으로 해결
- 2025.12.01(Mon) … AMR 4 & 6 불 아예 안들어오는 문제 해결(server문제)
- 2025.12.02(Tue) … 로봇B기능 추가 (대피소 안내기능)
- 2025.12.03(Wed) … PC 사용 대 수 결정
- 2025.12.03(Wed) … 하나의 맵& 두개의 로봇 문제 해결, depth 값 불러오기
- 2025.12.04(Thu) … AMR 통신오류