```
1. Caller가 콜을 함 (call action by caller)
2. Robot이 콜을 받고 받았다고 Caller에게 답장을 줌
3. Robot이 Caller의 위치로 따라감
4. Caller가 안전구역(집 등)으로 들어갈 때 까지 동행(임무 수행)
5. Caller가 종료를 선언함
6. Robot이 초기 위치로 돌아감
```

0. Camera

- **사용 환경**
    - YOLOv8 해상도: `1280 x 720`
    - PC: Ubuntu 22.04
    - USB 연결
- **입력**
    - 카메라 영상 이미지(최소 30 fps)
    - Topic: `/camera/image_raw/compressed`
    - Message Type: `sensor_msgs/CompressedImage`
- **출력**
    - 이미지와 검출(인식)된 객체 정보(좌표)
    - 바운딩 박스의 중앙값(높이, 너비 정보 포함 가능)
- **통신 방식**
    - ROS2 Topic (발행)
    - 토픽 이름: `/camera/image_raw/compressed`
    - 메시지 타입: `sensor_msgs/CompressedImage`

---

1. Caller의 Call

- **사용 환경**
    - PC: Ubuntu 22.04 / ROS2 / Python
    - GUI: Flask
    - 노드: `Service_call_node(Node)`
- **입력**

    - 카메라 이미지(최소 30 fps)
    - 검출(인식)된 객체 정보(JSON 형식, 실시간 업데이트)
- **출력**
    - Caller의 위치 좌표(`/caller_position`)
- **통신 방식**
    - ROS2 Service (클라이언트)
    - 구독 토픽: `/camera/image_raw/compressed`, `/detected_objects`
    - 발행 토픽: `/caller_position`
    - 메시지 타입(발행): `geometry_msgs/PoseStamped`

---

2. Robot이 콜을 받고, Caller에게 답장(ACK) 전송

- **사용 환경**
    - PC: Ubuntu 22.04 / ROS2 / Python
    - 노드: `Service_callback_node(Node)`
- **입력**
    - Caller의 Call Sign 토픽
    - Caller(객체)의 위치 정보
- **출력**
    - Call sign reaction 서비스
- **통신 방식**
    - ROS2 Service (서버)
    - 서비스 이름: `/call_acknowledgment`
    - 서비스 타입: 예) `CallAcknowledgment.srv` (사용자 정의)

---

3. Robot이 Caller의 위치로 이동 (추적 시작)

- **사용 환경**
    - AMR(Autonomous Mobile Robot)
    - 노드: `Robot_following_node(Node)`
- **입력**
    - Caller의 위치 정보(지속적 업데이트)
    - 로봇의 현재 위치 정보(odometry)

- **출력**
    - 로봇 이동 명령(velocity commands)
- **통신 방식**
    - ROS2 Topic (구독): `/caller_position`, `/odom`
    - ROS2 Topic (발행): `/cmd_vel`
    - 메시지 타입
        - 입력: `geometry_msgs/PoseStamped`, `nav_msgs/Odometry`
        - 출력: `geometry_msgs/Twist`

---

4. Robot이 Caller를 따라가며 임무 수행

- **사용 환경**
    - AMR(Autonomous Mobile Robot)
    - 노드: `Mission_execution_node(Node)`
- **입력**
    - Caller의 위치 정보
    - 로봇의 현재 위치 정보
- **출력**
    - 로봇 이동 명령
    - 미션 상태 정보(진행 중, 완료 등)
- **통신 방식**
    - ROS2 Topic (구독): `/caller_position`, `/odom`
    - ROS2 Topic (발행): `/cmd_vel`, `/mission_status`
    - 메시지 타입
        - 입력: `geometry_msgs/PoseStamped`, `nav_msgs/Odometry`
        - 출력: `geometry_msgs/Twist`, (사용자 정의) `MissionStatus.msg` 등

---

5. Caller가 종료를 선언함

- **사용 환경**
    - PC: Ubuntu 22.04 / ROS2 / Python
    - GUI: Flask (종료 버튼 포함)
    - 노드: `Mission_termination_node(Node)`
- **입력**

    - Caller의 종료 선언 신호 (GUI 버튼 입력)
- **출력**
    - 미션 종료 신호 (ROS2 Topic)
- **통신 방식**
    - ROS2 Topic (발행)
    - 토픽 이름: `/mission_terminate`
    - 메시지 타입: `std_msgs/Bool`

---

6. Robot이 초기 위치로 복귀

- **사용 환경**
    - AMR(Autonomous Mobile Robot)
    - 노드: `Robot_return_node(Node)`
- **입력**
    - 미션 종료 신호
    - 로봇의 현재 위치 정보(odometry)
    - 초기 위치 좌표
- **출력**
    - 로봇 이동 명령
    - 복귀 완료 상태 정보
- **통신 방식**
    - ROS2 Topic (구독): `/mission_terminate`, `/odom`
    - ROS2 Action (클라이언트): `/navigate_to_pose`
    - 메시지/액션 타입
        - 입력: `std_msgs/Bool`, `nav_msgs/Odometry`
        - 출력: `nav2_msgs/NavigateToPose`

위 구조를 통해 Caller(rc카 등)가 Call을 보내면, Robot은 Caller의 위치로 이동해 추적하고, Caller가 종료(terminate)를 선언하면 Robot은 다시 초기 위치로 복귀하는 전 과정을 ROS2 환경에 구성하였음

각 노드별 기능 및 통신 방식을 정리한 ROS2 시스템 구성 요약

1. 스탠딩 카메라 노드 (Standing_Camera_Node)

- **입력**
    - 카메라 영상 이미지(최소 30 fps)
    - 토픽: `/camera/image_raw/compressed` (예시)
- **출력**
    - 이미지 + 검출(인식)된 객체 정보(좌표)
    - 토픽: `/detected_objects`
    - 메시지 타입: 사용자 정의(`DetectedObjects.msg`)
- **통신 방식**
    - ROS2 Topic (발행)

---

2. AMR 카메라 및 객체 감지 노드 (AMR_Camera_YOLO_Node)

- **입력**
    - 카메라 영상 이미지(최소 30 fps)
    - 토픽: `/amr_camera/image_raw/compressed` (예시)
- **출력**
    - 이미지 + 바운딩 박스 정보
    - 토픽: `/amr_camera/detected_objects`
    - 메시지 타입: 사용자 정의(`DetectedObjects.msg`)
- **통신 방식**
    - ROS2 Topic (발행)

---

3. 로봇 제어 노드 (Robot_Control_Node)

- **입력**
    - 미션 시작 요청: ROS2 Service (서버) `"/start_mission"`
    - 객체(Caller)의 바운딩 박스 정보: 토픽 `"/amr_camera/detected_objects"` (메시지: `DetectedObjects.msg`)
    - 로봇의 현재 위치 정보: 토픽 `"/odom"` (메시지: `nav_msgs/Odometry`)

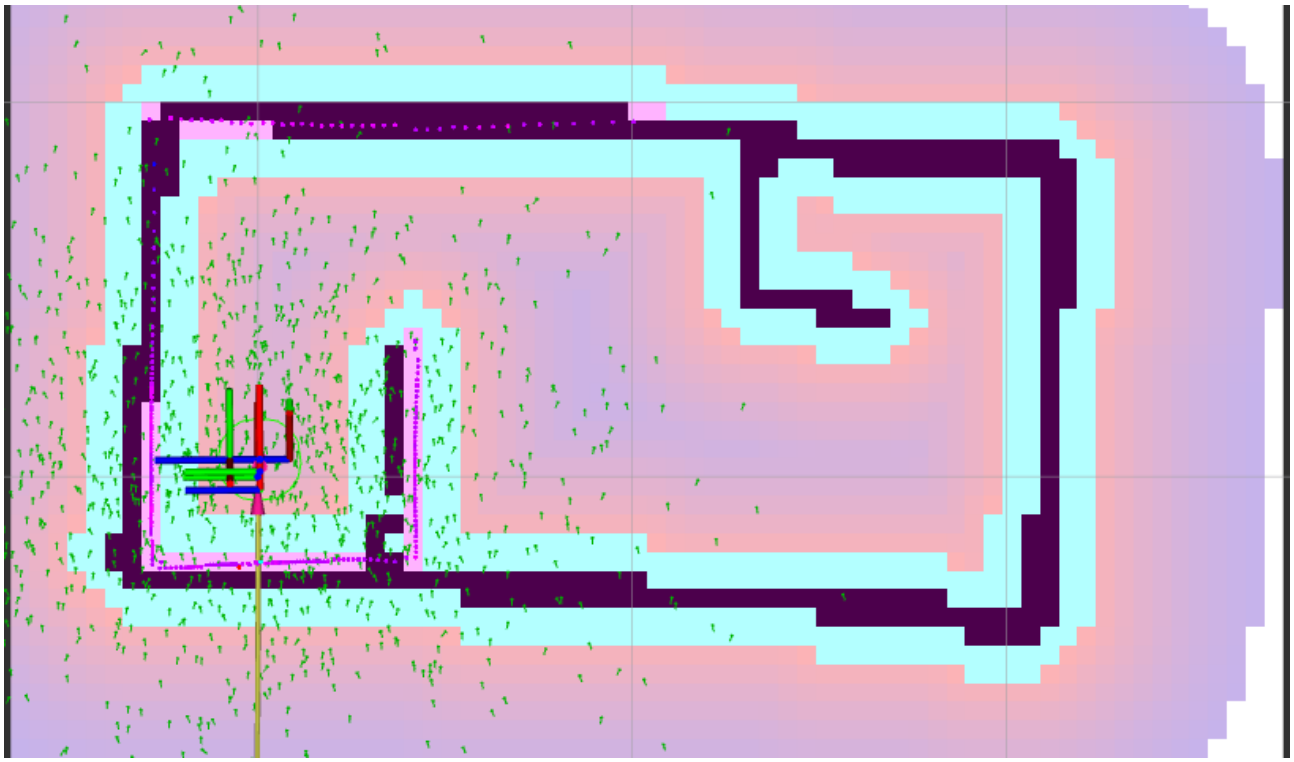
    - 미션 종료 신호: 토픽 `"/mission_terminate"` (메시지: `std_msgs/Bool`)
- **출력**
    - 미션 시작 응답(서비스 응답)
    - 로봇 이동 명령: 토픽 `"/cmd_vel"` (메시지: `geometry_msgs/Twist`)
    - 미션 상태 정보: 토픽 `"/mission_status"` (메시지: `MissionStatus.msg`)
    - 로봇의 현재 위치 정보: 토픽 `"/robot_current_position"` (메시지: `geometry_msgs/PoseStamped`)
- **통신 방식**
    - ROS2 Service (서버): `"/start_mission"` (타입: `StartMission.srv`)
    - ROS2 Topic (구독): `"/amr_camera/detected_objects"`, `"/odom"`, `"/mission_terminate"`
    - ROS2 Topic (발행): `"/cmd_vel"`, `"/mission_status"`, `"/robot_current_position"`

---

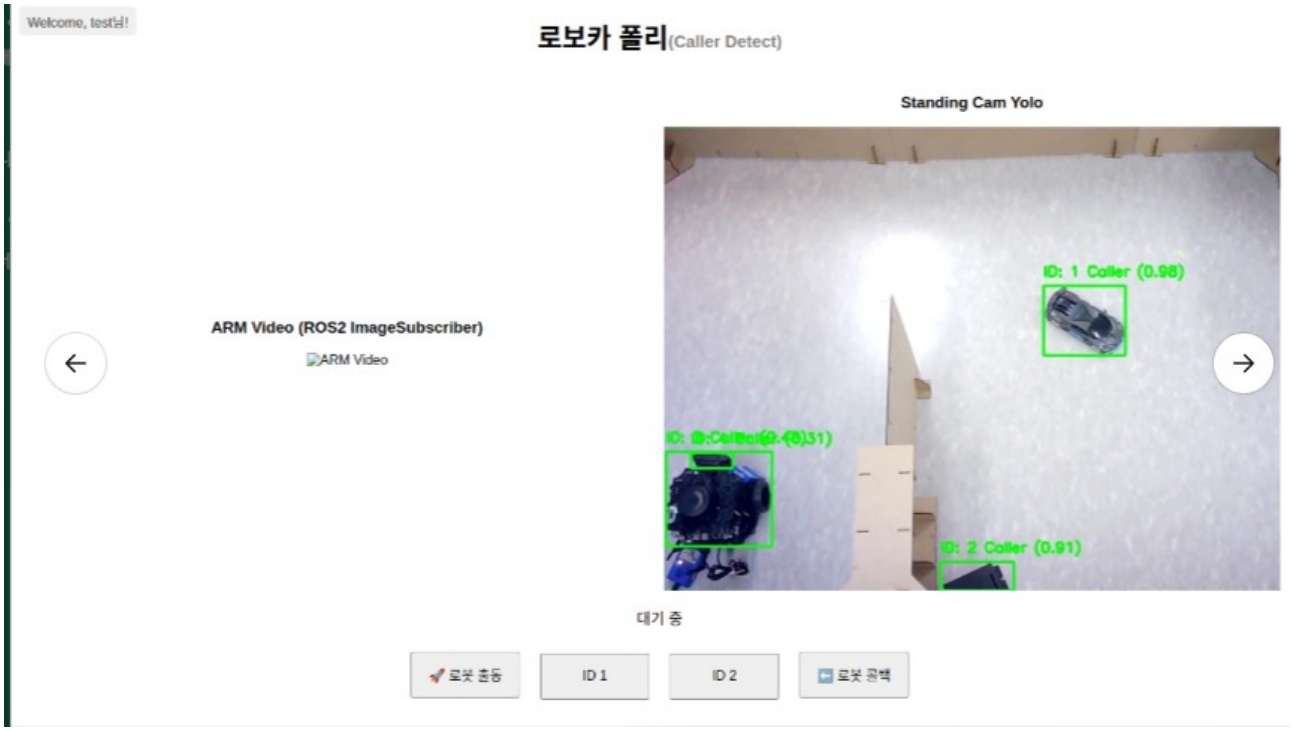
4. 미션 관리 노드 (Mission_Management_Node)

- **입력**
    - 사용자의 호출 요청 (GUI 버튼 입력)
    - Caller의 종료 선언 신호 (GUI 버튼 입력)
    - 로봇의 현재 위치 정보: 토픽 `"/robot_current_position"` (메시지: `geometry_msgs/PoseStamped`)
- **출력**
    - 미션 시작 요청 (ROS2 Service 클라이언트 호출)
    - 미션 종료 신호: 토픽 `"/mission_terminate"` (메시지: `std_msgs/Bool`)
    - 최종 요금 정보: 토픽 `"/final_fare"` (메시지: `FinalFare.msg`)
- **통신 방식**
    - ROS2 Service (클라이언트): `"/start_mission"` (타입: `StartMission.srv`)
    - ROS2 Topic (발행): `"/mission_terminate"`, `"/final_fare"`
    - ROS2 Topic (구독): `"/robot_current_position"`

위 구성을 통해 스탠딩 카메라 노드와 AMR 카메라 노드에서 인식된 객체 정보를 로봇 제어 노드로 전달하고, 미션 관리 노드에서 사용자 요청을 받아 로봇을 제어·관리하는 전체 워크플로우를 완성



로봇의 init-pose



< standing_camera와 gui 연동>