Software Design Document

물류창고 내 자동 운반 로봇 협업 시스템

0cha0cha

Version 1.0
Printed: February 8th, 2025
Team 0cha0cha :
Moon HongIL, Son Gyeong-Man, Park Ji-Hoon,
Choi Dea-hyuk, Lee Seung-Min, Lee Ju-Hak
Supervisor : Andreas Kim
Mentor : Jung Eun-Young, Ji Ye-Eun

rokey bootcamp

▼ 1. Introduction

1.1 Purpose

본 문서는 **물류 창고 내 자동 운반 로봇 협업 시스템**의 소프트웨어 설계를 기술하는 **Software Design Document (SDD)**이다.

본 SDD의 목적은 다음과 같다.

- **Business Requirement Document (BRD)** 및 System Requirement Document (SRD)에서 정의된 요구사항을 기반으로,
- 시스템을 구성하는 주요 **소프트웨어 컴포넌트, 데이터 구조, 인터페이스, 동작 흐름**을 설계 관점에서 명확히 기술하며,
- 향후 구현, 통합, 테스트 단계에서 설계 기준(reference)으로 활용될 수 있는 문서를 제공하는 것이다.

본 문서는 구현에 사용되는 세부 코드나 라이브러리 내부 동작을 설명하지 않으며,

시스템이 요구사항을 충족하기 위해 **어떻게 구조화되고 상호작용해야 하는지**에 초점을 둔다.

1.2 Scope

본 프로젝트는 물류 창고 환경에서 두 대의 TurtleBot 기반 AMR(Autonomous Mobile Robot)이 협업하여 물품을 자동으로 운반하는 **데모 수준의 자동화 시스템**을 설계·구현하는 것을 목표로 한다.

본 SDD에서 다루는 시스템 범위는 다음을 포함한다.

- 웹캠 기반 물품 감지 및 QR 코드 인식
- YOLO 기반 파손 물품 여부 확인
- 로봇 출동 및 목적지 할당 로직
- 작업 효율 향상을 위한 로봇 간 협업 로직
- 다중 로봇 환경에서의 충돌 방지 및 우선순위 제어
- ArUco Marker를 이용한 목적지 도착 후 정밀 정렬(Alignment)

본 프로젝트는 대규모 상용 물류 자동화 시스템 구축을 목표로 하지 않으며,

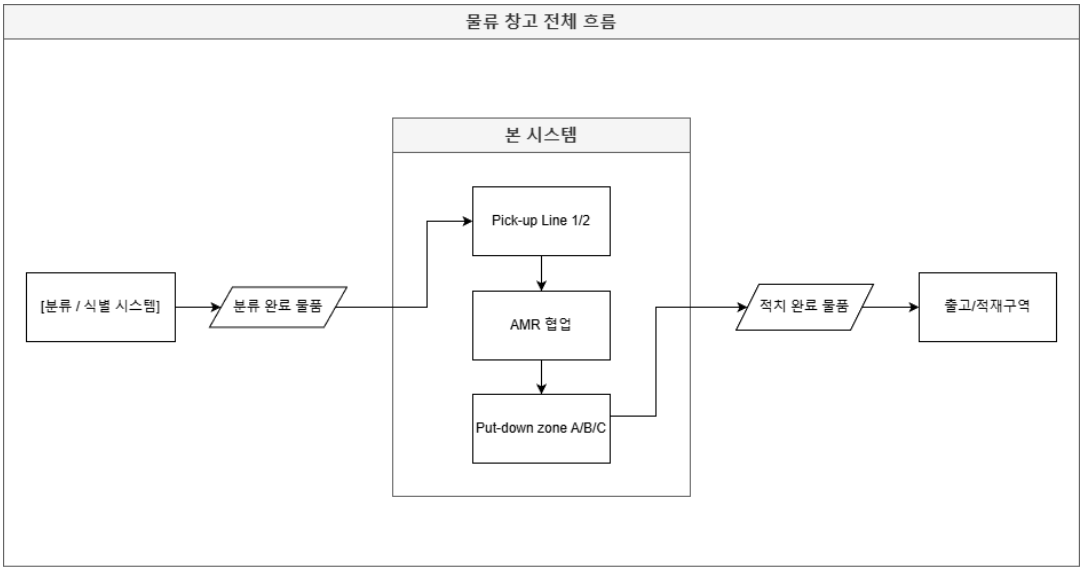
개념 검증(Proof of Concept)을 위한 소프트웨어 설계에 중점을 둔다.

1.3 Definitions and Abbreviations

본 절에서는 본 문서를 이해하는 데 필요한 주요 용어 및 약어를 정의한다.

용어	정의
SDD	Software Design Document
BRD	Business Requirement Document
SRD	System Requirement Document
AMR	Autonomous Mobile Robot
Pick-up Line	물품이 적재되어 로봇이 출동하는 대기 라인
Put-down Location	로봇이 물품을 운반하여 하역하는 목표 위치
Alignment	ArUco Marker 등을 이용해 로봇의 위치를 미세 조정하는 과정

▼ 2. System Overview



본 장에서는 본 프로젝트에서 설계한 시스템의 **기능적 범위, 적용 배경, 운영 환경**에 대한 개괄적인 설명을 제공한다.

본 시스템은 물류 창고 내부에서 발생하는 반복적인 운반 작업을 대상으로 하며,

다중 자동 운반 로봇(AMR)을 활용한 협업 기반 자동 운반 구조의 타당성을 검증하기 위한 데모 시스템이다.

2.1 System Context

본 시스템은 물류 창고 환경에서 **분류 및 식별이 완료된 물품을 내부적으로 운반하는 과정**을 대상으로 한다.

해당 구간은 픽업 라인에서 출고(적치) 위치까지의 반복적인 이동 작업으로 구성되며, 기존에는 인력 중심으로 수행되어 왔다.

Business Requirement Document에서 정의한 바와 같이, 이 내부 운반 구간은 작업량 변동에 따라 인력 투입이 급격히 증가하고, 운반 지연이 누적될 경우 전체 출고 및 배송 일정에 직접적인 영향을 미치는 **병목 구간**으로 작용한다.

본 시스템은 이러한 병목 구간을 대상으로 다중 자동 운반 로봇(AMR)을 활용한 협업 기반 자동 운반 구조를 설계하고, 데모 환경에서 그 기술적·운영적 타당성을 검증하는 것을 목적으로 한다.

2.2 System Objectives

본 시스템의 주요 목적은 다음과 같다.

- 분류 완료 이후 내부 운반 구간의 자동화를 통해 반복적 수작업을 최소화
- 다중 로봇 환경에서의 작업 분담 및 협업 제어 구조 검증
- 동일 목적지 요청 시 발생할 수 있는 충돌 및 정체 상황에 대한 제어 가능성 검증
- 제한된 하드웨어 및 개발 기간 내에서 구현 가능한 현실적인 자동화 구조 제시

본 시스템은 상용 물류 자동화 시스템 구축이 아닌, **Proof of Concept 수준의 설계 및 구현 검증**을 목표로 한다.

2.3 System Functional Overview

시스템은 **두 개의 물품 Pick-up 라인**과 **세 개의 Put-down 위치(A, B, C)**를 중심으로 동작한다.

각 Pick-up 라인에는 고정된 웹캠이 설치되어 있으며, 웹캠은 물품 상단의 QR 코드를 인식하여 파손 여부 및 목표 적치 위치 정보를 시스템에 전달한다.

물품이 감지되면 해당 라인을 담당하는 로봇이 자동으로 출동하며, 로봇은 Navigation 기능을 이용해 지정된 Put-down 위치로 이동한다.

목표 위치에 도착한 이후에는 ArUco Mark를 활용한 정밀 정렬(Alignment)을 수행하여 보다 정확한 위치 조정을 진행한다.

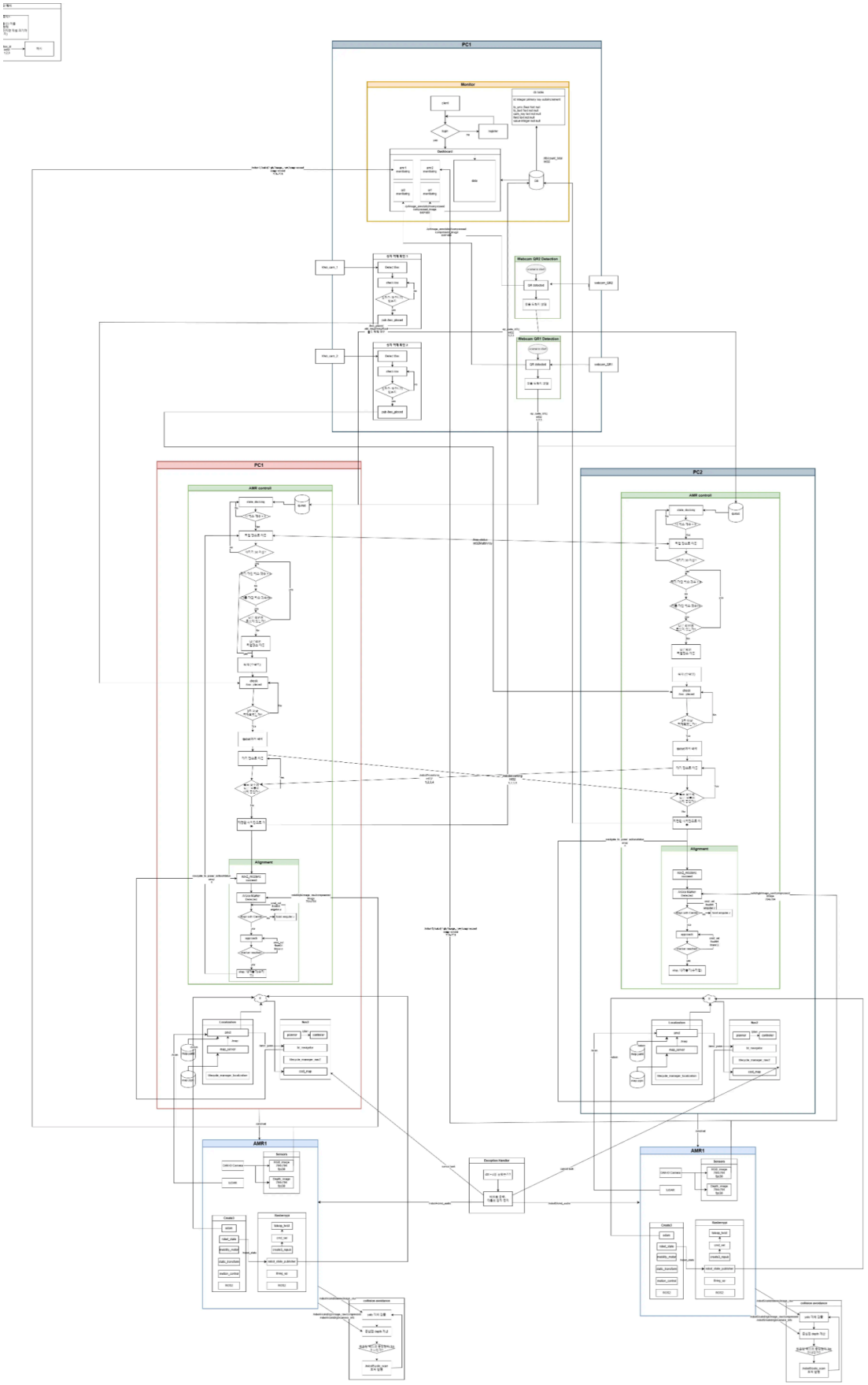
동일한 목적지 요청이 중복될 경우, 사전에 정의된 로봇 간 우선순위 규칙에 따라 대기 또는 재경로 설정이 이루어진다.

2.4 Operating Environment

본 시스템의 운영 환경은 다음과 같다.

- **자동 운반 로봇(AMR)**: 2대
- **Pick-up 라인**: 2개 (Line 1, Line 2)
- **Put-down 위치**: 3개 (A, B, C)
- **컴퓨팅 환경**: 노트북 2대
- **시각 입력 장치**: Pick-up 라인별 고정 카메라 2대

▼ 3. System Architecture


PC1
Monitor
Dashboard
Webcam QR2 Detection
Webcam QR1 Detection
PC1
AMR control
Alignment
PC2
AMR control
Alignment
AMR1
Sensors
Exception Handler
AMR1
Sensors

3.1 Architectural Design

본 시스템은 **다중 AMR 기반 물류 자동 운반 데모 시스템**으로서, 두 대의 TurtleBot 기반 AMR과 이를 제어하는 두 대의 PC, 그리고 총 네 대의 비전 입력 장치로 구성된 **분산형 소프트웨어 아키텍처**를 따른다.

각 로봇은 독립적인 제어 환경에서 동작하며, 시스템 전반의 협업 및 작업 관리는 논리적으로 분리된 소프트웨어 서브시스템 간의 메시지 기반 상호작용을 통해 이루어진다.

본 아키텍처는 다음과 같은 주요 서브시스템으로 구성된다.

3.1.1 Vision Subsystem

Vision Subsystem은 물류 처리 과정에서 발생하는 시각 정보를 수집하고, 이를 시스템이 활용 가능한 정보로 변환하는 역할을 수행한다.

주요 기능은 다음과 같다.

- 픽업 라인에 배치된 웹캠을 이용한 물품 존재 여부 감지
- QR 코드를 통한 물품 식별 정보 획득
- 웹캠 기반 물품 적재 여부 확인

Vision Subsystem은 판단 결과만을 상위 제어 로직에 전달하며, 로봇 제어나 이동 결정에는 직접 개입하지 않는다.

3.1.2 Robot Control Subsystem

Robot Control Subsystem은 본 시스템의 **중앙 제어 역할**을 수행하는 핵심 서브시스템이다.

각 로봇의 상태를 관리하고, 작업(Task)을 생성·할당·관리한다.

주요 기능은 다음과 같다.

- 로봇 상태 관리 (배터리 상태, 도킹 여부, 작업 상태)
- 물품 정보 관리 (목표 위치, 수량)
- 작업 큐(Task Queue) 관리 및 로봇 출동 제어
- 다중 로봇 협업 및 역할 재할당 로직 수행
- 동일 목표 지점에 대한 우선순위 결정 및 충돌 회피 로직 처리
- ArUco Mark를 활용한 도착 지점 미세 정렬 제어

Robot Control Subsystem은 Navigation Subsystem과 분리되어 있으며, 이동 수행 자체보다는 **작업 의사결정과 상태 전이**에 집중한다.

3.1.3 Navigation Subsystem

Navigation Subsystem은 로봇의 물리적 이동을 담당하는 서브시스템으로, 지정된 목적지까지의 경로 계획 및 이동 수행을 담당한다.

주요 기능은 다음과 같다.

- 목적지(픽업 위치, 적치 위치, 대기 위치, 도킹 위치)로의 이동 수행
- 이동 중 장애물 회피 및 경로 추종
- 이동 완료 여부 판단 및 상태 보고

본 서브시스템은 상위 제어 로직으로부터 목표 위치를 전달받아

독립적으로 이동을 수행하며, 이동 결과만을 상위에 보고한다.

3.1.4 Monitoring System

Monitoring System은 시스템의 운영 상태를 관찰하기 위한 보조 서브시스템으로, 직접적인 제어 로직에는 개입하지 않는다.

주요 기능은 다음과 같다.

- 웹캠 영상 및 AMR 카메라 영상 실시간 모니터링
- 물품 정보 및 운송 상태 시각화

- 관리자 계정 기반 로그인(Admin) 접근 제어

Monitoring System은 시스템의 안정성과 운영 가시성을 높이기 위한 용도로 사용된다.

3.2 Subsystem Decomposition Description

본 시스템은 **기능 단위로 명확히 분리된 서브시스템 간 협업 구조**를 따른다.

- Vision Subsystem은 **환경 인식 전담**
- Robot Control Subsystem은 **의사결정 및 이동 실행 전담**
- Monitoring System은 **관측 및 기록 전담**

각 서브시스템은 독립적으로 동작하며, 명확히 정의된 인터페이스를 통해서만 상호작용한다.

이러한 분해 구조를 통해 특정 서브시스템의 수정이나 확장이 다른 서브시스템에 미치는 영향을 최소화하도록 설계되었다.

3.3 설계 이론적 해석

본 시스템에서 제안한 분산형 아키텍처는 다음과 같은 이유로 선택되었다.

- 기능별 모듈화 구조를 통해 개발 및 디버깅이 용이함
- 로봇 수 증가 시 협업 로직 확장이 가능함
- 비전 처리와 로봇 제어를 분리하여 시스템 안정성을 확보할 수 있음
- 제한된 개발 기간(약 5일)과 하드웨어 자원 내에서 구현 가능한 현실적인 구조임

대안으로 중앙 집중식 제어 아키텍처도 고려되었으나, 다중 로봇의 Nav2 및 Localization 동시 운용 시 발생할 수 있는 충돌 문제와 컴퓨팅 자원 제한으로 인해 채택하지 않았다.

본 아키텍처는 데모 환경에서의 구현 가능성과 확장 가능성 간의 균형을 고려한 결과이다.

▼ 4. Data design

4.1 데이터 설명

4.1.1 정보 도메인 → 데이터 구조 변환

본 시스템의 정보 도메인은 **물품 단위의 작업(Task)** 으로 모델링된다. 각 Pick-up 라인에서 발생하는 감지 이벤트(물품 감지/QR 인식/파손 판정)를 입력으로 하여 Task를 생성하고, 이를 라인별 작업 큐에 적재한 뒤, 로봇의 디스패치·내비게이션 목표·우선순위 제어로 변환한다.

특히 SRD에서는 Task가 최소한 다음 정보를 포함해야 함을 요구한다:

source_line(P1/P2), target_putdown(A/B/C), quantity, timestamp

또한 **손상(Damaged) 물품은 작업 생성에서 제외**되어야 한다.

4.1.2 핵심 데이터 흐름(End-to-End)

(1) 라인 비전 입력 → 인식 결과 생성

- 입력: 라인별 카메라 이미지 프레임
- 처리 결과(논리):
  - QR 디코딩 결과(목표 Put-down A/B/C 및 라인 정보)
  - Damaged 여부 판정(데모 규칙 기반)

(2) Task 생성 및 라인별 큐 적재(Queueing)

- QR 결과를 기반으로 Task를 생성하고, Line1/Line2 Queue에 적재한다.
- Task는 기본적으로 해당 라인의 담당 로봇에 할당되며(예: P1→robot4, P2→robot5), 필요 시 협업 규칙에 따라 라인 지원이 발생할 수 있다.

(3) 적재 확인 → 목표 전송 → Nav 수행

- SRD 요구사항상 목표 전달 시점은 **"적재 확인 완료 후"** 로 정의된다.
- 적재 확인은 데모 기준으로 **Pick-up 도착 후 최소 5초 정지**(필수) + (선택) 카메라 확인 단계로 구성된다.
- 이후 Task의 target_putdown(A/B/C)가 로봇 내비게이션 목표로 변환되어 전달된다.

(4) 목표 중복 시 우선순위/대기 상태 관리

- 두 로봇이 동일 Put-down을 목표로 할 경우,
  - 목표를 먼저 선언한 로봇이 1순위
  - 2순위는 Put-down별 대기 위치로 이동
  - 1순위 작업 완료 신호 수신 후 2순위가 진입하여 수행
- 이 로직을 위해 Put-down별 "점유 로봇 / 대기 목록 / 완료 신호" 상태가 필요하다.

(5) 모니터링 데이터 흐름(표시용)

- 입력: 웹캠 영상(최대 4개) + AMR 카메라 영상
- 처리: 별도의 제어 개입 없이 그대로 표시(관측/확인 목적)

4.1.3 데이터 저장/관리(논리)

본 시스템은 대규모 DB가 아니라 **런타임 메모리 기반의 큐/상태 저장소** 중심으로 설계한다.

- **Line별 Task Queue**
  - Line1 Queue, Line2 Queue
  - 기본적으로 라인 담당 로봇이 소비(consume)
- **우선순위/충돌 관리 상태**
  - Put-down 위치별 점유/대기/완료 플래그
- **Robot State**
  - SoC, 도킹 여부, 현재 task, 현재 목표 및 상태 머신 상태

4.2. data dic

4.2.1 QR, Monitoring

인터페이스 네임	형식	pub/sub	내용	설명
Vision_QR				
`/qr_code_id1`	`int32`	pub	1,2,3	qr코드 주소를 읽어 put down(1,2,3) 위치 발행
`/qr_code_id2`	`int32`	pub	1,2,3	
Monitoring 입력				
`/qr1/image_annotated/compressed`	`compressed`	sub	640*480*	QR1 webcam에서 입력
`/qr2/image_annotated/compressed`	`compressed`	sub	640*480*	QR2 webcam에서 입력
/yolo_webcam1/image/compressed	`compressed`	sub	640*480*	placed webcam에서 입력
/yolo_webcam2/image/compressed	`compressed`	sub	640*480*	placed webcam에서 입력
`/robot4/oakd/rgb/image_raw/compressed`	`compressed`	sub	640*480*	AMR04에서 입력
`/robot5/oakd/rgb/image_raw/compressed`	`compressed`	sub	640*480*	AMR05에서 입력
Vision_stack				

`/box_placed1`	`bool`	pub	True/False	AMR control logic으로 발행하는 출발 트리거
`/box_placed2`	`bool`	pub	True/False	

4.2.2 AMR 충전 상태와 출발 트리거 협업 (/robot_working이 없어)

인터페이스 네임	형식	pub/sub	내용	설명
`main_controller.py`				
`/battery_state`	`BatteryState`	sub	0.0 ~ 1.0	현재 터틀봇의 배터리 상태를 받는다 0.1마다 바뀜
`/line_status`	`Int32MultiArray`	sub	[0,0,0]	라인 1과 라인 2에 각 로봇이 박스를 받고 있는 중인지 체크
`/box_placed1`	`Bool`	sub	True/False	라인 1 AMR 출발 트리거
`/box_placed2`	`Bool`	sub	True/False	라인 2 AMR 출발 트리거
`/pop_queue1`	`Int32`	pub	1	0 초과 숫자를 pub를 하면 robot4 큐 삭제
`/pop_queue2`	`Int32`	pub	1	0 초과 숫자를 pub를 하면 robot5 큐 삭제
`/pop_queue1`	`Int32`	sub	1	0 초과 숫자를 sub 하면 robot4 큐 삭제
`/pop_queue2`	`Int32`	sub	1	0 초과 숫자를 sub 하면 robot5 큐 삭제
`/robot_working4`	`Int32`	pub/sub	1,2,3,4	1,2,3 : 중복라인 들어갈시 정지 / 4 : 작업완료
`/robot_working5`	`Int32`	pub/sub	1,2,3,4	1,2,3 : 중복라인 들어갈시 정지 / 4 : 작업완료

4.2.3 Yolo 장애물 검출

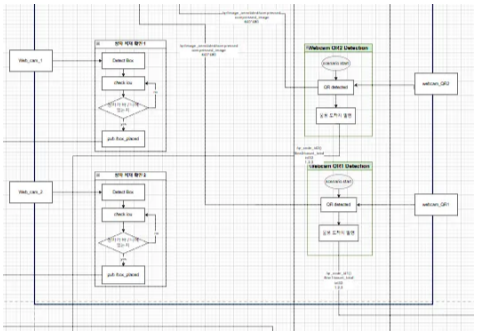
인터페이스 네임	형식	pub/sub	내용	설명
avoid.py				
`/robot4/oakd/rgb/image_raw/compressed`	`Image`	sub	rgb Image	
`/robot4/oakd/rgb/camera_info`	`CameraInfo`	sub	camera parameter	
`/robot4/oakd/stereo/image_raw`	`Image`	sub	depth Image	
`/robot4/yolo_scan`	`LaserScan`	pub	scan	rviz에 장애물을 표시하기 위한 레이저 스캔 토

▼ 5. Component Design

본 장에서는 3.0에서 정의한 서브시스템을 실제 구현 단위(컴포넌트)로 분해하고, 각 요소의 책임, 입력/출력, 내부 데이터, 주요 알고리즘(의사코드)을 기술한다.

이 프로젝트의 실제 구현 단위(컴포넌트)는 **각 단계별 기능하나를 기준**으로 기술한다.

5.1. Vision Subsystem : 정보 인식 전담



5.1.2 Function: Decode QR (Pick-up Line Camera)

Purpose

Pick-up 라인에 설치된 고정 웹캠 영상에서 QR을 디코딩하여 목적지(A/B/C) 및 Robot Control에 전달한다.

Inputs

- Line camera frames (Line1, Line2)

Outputs

- `/qr_code_id1`, `/qr_code_id2`

Local Data

- 1, 2, 3

5.1.3 Function: Stack Check (YOLO-based)

Purpose

YOLO 기반으로 물품이 정확히 올려져 있는지를 판단하고, 출발신호를 보낸다

Inputs

- Frame (물품 이미지)

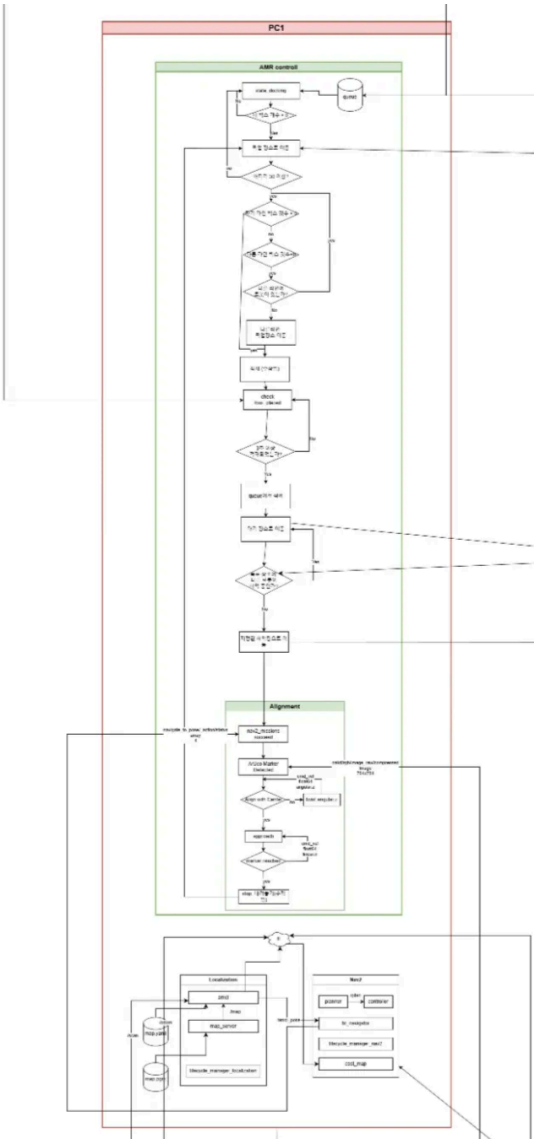
Outputs

- `box_placed1` , `box_placed2`

Local Data

- ROI_MODEL_PATH = 관심영역 학습파일(.pt) 경로
- BOX_MODEL_PATH = 상자 학습파일(.pt) 경로
- ROI_CONF = 관심영역 신뢰도 임계값
- BOX_CONF = 박스 신뢰도 임계값
- IOU_TH = IOU 임계값
- START_SEC = True 토픽 퍼블리싱 시작 시간
- END_SEC = True 토픽 퍼블리싱 종료 시간

5.2. Robot Control Subsystem (의사결정 및 이동 실행 전담)



5.2.1 Function: Create Task & Enqueue

Purpose

VisionEvent를 기반으로 Task를 생성하고 라인별 큐에 적재한다. 파손품은 큐에 적재하지 않는다.

Inputs

- `/qr_code_id1` (std_msgs/Int32): 1번 라인 비전 센서에서 인식된 QR 데이터
- `/qr_code_id2` (std_msgs/Int32): 2번 라인 비전 센서에서 인식된 QR 데이터

Outputs

- `self.queue1` (deque): 1번 라인 작업 대기열에 Task ID 추가
- `self.queue2` (deque): 2번 라인 작업 대기열에 Task ID 추가

Local Data

- `self.queue1` (deque): 1번 라인 작업 대기열에 1, 2, 3 번 추가
- `self.queue2` (deque): 2번 라인 작업 대기열에 1, 2, 3 번 추가

5.2.2 Function: undock Robot & go to Pick-up

Purpose

Task가 큐에 적재되면, 해당 라인의 기본 담당 로봇에게 Pick-up 이동을 지시한다.

Inputs

- `self.queue1` , `self.queue2`

Outputs

- `self.state` : `CHARGING` → `MOVE_TO_PICKUP` → `WAITTING` 으로 변경
- `Undock` : 로봇을 충전 스테이션에서 분리
- `MapsToPose` : 픽업 위치 좌표로 이동시키는 주행 명령 (`follow_move_and_wait`)

Local Data

- `self.my_robot_id` (int): 로봇 식별자 (4 or 5)
- `self.current_working_line` (int): 작업 라인 초기화 (Robot 4는 1, Robot 5는 2로 설정)
- `self.is_helping` (bool): 지원 모드 플래그 초기화 (False)

5.2.4 Function: Stack Check (input Vision)

Purpose

Pick-up 도착 후 적재 완료를 확인한다. stackcheck webcam에서 적재 여부를 확인 받는다.

Inputs

- `box_placed1` , `box_placed2`

Outputs

- `self.state` : `LOADING` → `MOVE_TO_DEST` 로 변경

Local Data

- `self.start` (bool): 1번 라인 적재 플래그 (Default: False)
- `self.start2` (bool): 2번 라인 적재 플래그 (Default: False)
- `self.is_helping` (bool): 현재 지원 모드인지 확인 (자신의 라인 신호를 볼지, 상대방 신호를 볼지 결정)

5.2.5 Function: Priority & Conflict Handling (same Put-down)

Purpose

동일 Put-down(A/B/C) 목표 중복 시 우선순위 규칙에 따라 진행/대기 및 대기 위치 이동을 처리한다.

Inputs

- `/robot5/working` (robot4에서 실행 시)

Outputs

- `/robot4/working`

5.2.6 Function: Put-down Execution + Alignment (ArUco + time stop)

Purpose

Put-down 도착 후 ArUco marker 기반 정밀 정렬을 수행하고, 5초 정지(하역 데모) 후 작업 완료를 선언한다

Inputs

- `navigate_to_pose/_action/status`
- `oakd/rgb/image_raw/compressed`

Outputs

- `cmd_vel`

5.2.7 Function: Support other line

Purpose

본인 라인이 비어 있고 상대 라인에 물품이 있으며 상대 로봇이 비가용 상태일 경우 지원한다.

Inputs

- `/line_status` (std_msgs/Int32MultiArray): 각 라인 픽업 구역의 점유 상태 (로봇 유무)

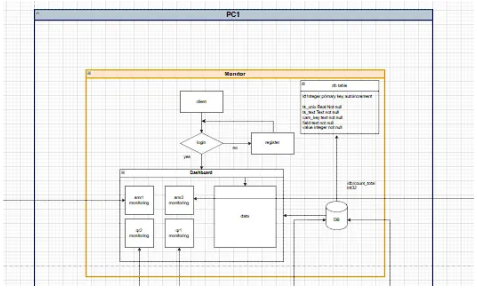
Outputs

- `self.state` : `WAITTING` → `GO_TO_OTHER` 로 변경

Local Data

- `self.is_helping` (bool): 지원 모드 활성화 (Set to **True**)
- `self.current_working_line` (int): 현재 작업 라인 변경 (Robot 4: 1→2 / Robot 5: 2→1)

5.3 Monitoring System : 관측 및 기록 전담



5.3.1 Function: Display Streams & Status (read-only)

Purpose

웹캠 및 AMR 카메라 영상, 로봇 상태/작업 상태를 표시한다. 제어 개입 로직은 포함하지 않는다.

Inputs

- `/qr1/image_annotated/compressed`
- `/qr2/image_annotated/compressed`
- `/yolo_webcam2/image/compressed`
- `/robot4/oakd/rgb/image_raw/compressed`
- `/robot5/oakd/rgb/image_raw/compressed`

Outputs

- UI display (read-only)

▼ 6. Human-Interface Design

본 장에서는 물류 창고 자동 운반 로봇 협업 시스템의 운영 및 모니터링을 위해 제공되는 인간–시스템 인터페이스(Human Interface)에 대해 기술한다.

6.1 Interface Design Principles

본 시스템의 인간 인터페이스는 다음 설계 원칙을 따른다.

- 가시성(Visibility)시스템의 현재 상태(로봇 상태, 작업 상태)는 운영자가 즉시 인지할 수 있어야 한다.
- 단순성(Simplicity)데모 환경을 고려하여 불필요한 입력 요소를 최소화하고, 핵심 정보만을 표시한다.
- 비개입성(Non-intrusiveness)정상 동작 중에는 운영자의 수동 개입이 필요하지 않도록 설계한다.

6.2 User Roles

본 시스템에서 정의되는 사용자 역할은 다음과 같다.

- **Operator (운영자)**
  - 시스템 전반의 상태를 모니터링
  - 예외 상황 발생 시 인지 및 대응
  - 데모 시나리오 진행 여부 확인

본 프로젝트에서는 **단일 사용자 역할**만을 가정하며, 관리자/권한 분리 등의 고급 사용자 모델은 범위에서 제외한다.

6.3 Monitoring Interface Overview

운영자는 모니터링 인터페이스를 통해 다음 정보를 확인할 수 있어야 한다.

6.3.1 webcam, AMR camera mirroring

각 비전장치의 화면을 입력받아 출력한다.

- AMR camera1,2
- webcam_QR1,2
- wbcam_stack1,2

6.3.2 Robot Status Panel

각 로봇에 대해 다음 상태 정보를 표시한다.

- Queue
- 현재 작업(Task) ID

6.4 Human Interface Constraints

- 인터페이스는 데스크톱 환경(노트북 화면)을 기준으로 설계한다.
- 모바일 또는 다중 사용자 인터페이스는 범위에서 제외한다.
- UI는 기능적 요구사항을 충족하는 수준으로만 구현한다.

▼ 7. Performace Requirements

본 장에서는 물류 창고 자동 운반 로봇 협업 시스템이 정상적으로 동작하고 요구된 목적을 달성하기 위해 충족해야 하는 성능(Performance) 관련 요구사항을 정의한다.

본 성능 요구사항은 **데모(Proof of Concept) 환경을 기준**으로 설정되었으며, 절대적인 산업 표준 수치가 아닌 **상대적 성능 검증과 시스템 안정성 확보**를 목표로 한다.

7.1 Task Processing Performance

7.1.1 작업(Task) 생성 지연 시간

- 물품이 Pick-up 라인에서 감지된 시점부터 해당 물품에 대한 작업(Task)이 생성되어 Queue에 반영되기까지의 지연 시간은 **2초 이내**여야 한다.

의의

물품 감지 이후 작업 큐 반영 지연을 최소화하여 전체 운반 처리 흐름의 정체를 방지한다.

7.1.2 작업 할당 반응 시간

- 작업(Task)이 생성된 시점부터
- 로봇에게 해당 작업이 할당(Dispatch)되기까지의 시간은 **5초 이내**여야 한다.

의의

로봇의 유휴 시간을 줄이고 내부 운반 병목 구간의 체류 시간을 완화한다.

7.2 Navigation and Execution Performance

7.2.1 목표 도착 정확도

- 로봇은 지정된 Put-down 위치에 대해 목표 좌표 기준 **반경 ±0.25 m 이내**로 도착해야 한다.

의의

출고 위치 중복 처리 시 발생할 수 있는 물리적 충돌 및 작업 간섭을 방지하기 위한 최소 정밀도 기준이다.

7.3 Multi-Robot Cooperation Performance

7.3.1 목표 중복 처리 안정성

- 두 대 이상의 로봇이 동일한 Put-down 위치를 요청하는 상황에서 다음 조건을 모두 만족해야 한다.
  - 물리적 충돌 발생: **0회**
  - 데드락(상호 무한 대기) 발생: **0회**

의의

우선순위 및 대기 제어 로직이 다중 로봇 환경에서도 안정적으로 동작함을 검증한다.

7.3.2 협업 전환 반응 시간

- 한 로봇이 다른 로봇의 작업을 지원하기까지의 시간은 **10초 이내**여야 한다.

의의

단일 로봇 의존도를 줄이고 시스템 전체 처리량 저하를 방지한다.

7.4 System Reliability and Stability

7.4.1 연속 운용 안정성

- 시스템은 **20분 이상 연속 운용** 중 다음 상황이 발생하지 않아야 한다.
  - 시스템 중단
  - 강제 재시작
  - 제어 불능 상태

의의

데모 환경에서 요구되는 기본적인 시스템 신뢰성과 안정성을 보장한다.

7.5 Perception Performance

7.5.1 물품 식별 성공률

- 정상 조건(조명 안정, 가림 없음) 하에서 물품 식별(QR 코드 인식 포함) 성공률은 **90% 이상**을 목표로 한다.

의의

작업(Task) 생성의 신뢰도를 확보하기 위한

최소 성능 기준이다.

7.6 Performance Requirement Summary

항목	성능 기준
Task 생성 지연	≤ 2초
Task 할당 지연	≤ 5초
목표 도착 정확도	±0.25 m
협업 충돌 / 데드락	0회
연속 운용 시간	≥ 30분
인식 성공률	≥ 90%

▼ 8. Error Handling and Recovery

본 장에서는 물류 창고 자동 운반 로봇 협업 시스템에서 발생 가능한 오류 상황을 식별하고, 각 오류에 대해 시스템이 어떻게 인지하며 어떠한 방식으로 복구 또는 안전 상태로 전이되는지를 설계 관점에서 정의한다.

본 오류 처리 및 복구 설계는 **완전한 자율 복구를 목표로 하지 않으며**, 데모(Proof of Concept) 환경에서의 **시스템 안정성 유지와 오류 확산 방지**를 주요 목표로 한다.

---

8. Error Handling

상황

- 사용자가 **Ctrl + C 입력 시**
- 사용자가 **q 키 입력 시**

처리 절차 (설계 관점)

- **Ctrl + C 입력 감지**
  - 시스템 종료 신호(SIGINT)를 감지한다.
  - 로봇의 현재 주행 명령을 즉시 중단하고 속도 명령을 0으로 설정하여 로봇을 정지시킨다.
  - 동시에 경고음을 발생시켜 사용자에게 긴급 정지 상태임을 알린다.
  - 이후 ROS2 노드를 안전하게 종료하여 시스템 리소스 누수를 방지한다.
- **q 키 입력 감지**
  - 사용자 수동 종료 명령으로 인식한다.
  - 기존의 Ctrl + C와 동일한 정지 절차를 수행하여 로봇을 안전 상태로 전환한다.
  - 정상적인 프로그램 종료 루틴을 실행하여 시스템을 안정적으로 종료한다.

의의

- 긴급 상황에서 사용자가 즉시 로봇을 정지시킬 수 있도록 하여 **작업 환경 내 안전성을 확보**하였다.
- 단순 프로세스 종료가 아닌 **주행 정지 → 경고 알림 → 안전 종료 절차**를 수행함으로써 로봇 시스템의 신뢰성을 향상시켰다.

- 사용자 수동 개입 기능을 제공하여 자율 시스템 동작 중 발생할 수 있는 예기치 못한 상황에 대응할 수 있도록 설계하였다.

▼ 9. Deployment and Maintenance Plan

본 시스템은 **데모(Proof of Concept) 환경**을 기준으로 설계된 자동 운반 로봇 협업 시스템이다.

따라서 복잡한 배포 자동화나 장기적인 유지보수 체계는 본 문서의 범위에 포함하지 않는다.

9.1 Deployment Plan

- 시스템 배포는 사전에 구성된 개발 환경에서 수행된다.
- 각 소프트웨어 컴포넌트는 ROS 노드 단위로 실행된다.
- 배포 절차는 다음과 같이 단순화된다.
  - 시스템 전원 인가
  - ROS 환경 설정 로드
  - 각 노드 실행

본 배포 방식은 데모 환경에서의 반복 실행과 검증을 목적으로 한다.

9.2 Maintenance Plan

- 유지보수는 데모 환경에서 필요한 최소 수준으로 제한한다.
- 주요 유지보수 항목은 다음과 같다.
  - 설정 파라미터 수정
  - 노드 재시작
  - 경미한 로직 수정

장기 운영을 위한 자동 업데이트, 원격 배포, 장애 예측 시스템은

본 프로젝트의 범위에 포함하지 않는다.