

# GOOD MORNING!

早上好!

안녕하세요!

---

PROJECT INTRODUCTION



# WELCOME TO:

## [지능 - 1 A/B] Computer Vision and Automoous Driving 을 활용한 협동 로봇 구현

차시	내용	목표
1	로봇 AI 시스템 개발 프로세스 이해	시스템 개발 프로세스의 이해 개발 환경 구축
2	프로젝트에 필요 기술 검증	AI VISION 기술 탐색 및 검증
3/4	프로젝트에 필요 기술 검증	로봇 AMR 제어 기술 탐색 및 검증
5	프로젝트에 필요 기술 검증	웹 시스템 모니터 기술 탐색 및 검증
6-9	파이널 프로젝트	통합 시스템 설계 및 개발
10	최종 프레젠테이션 및 시연	시스템 발표 및 시연

# Who Am I?

## Andreas (Andy) A. Kim

Born in Korea; Immigrated to US in 1976

### Education:

BS Math, CS.; MS CSE; MSM(MBA);

### Work Experience: 35+ yrs.

Recent Positions: VP/MD Asia, CTO, CSO, 고문

### Worked For:



### Mentored/Mentoring:



### Instructor:

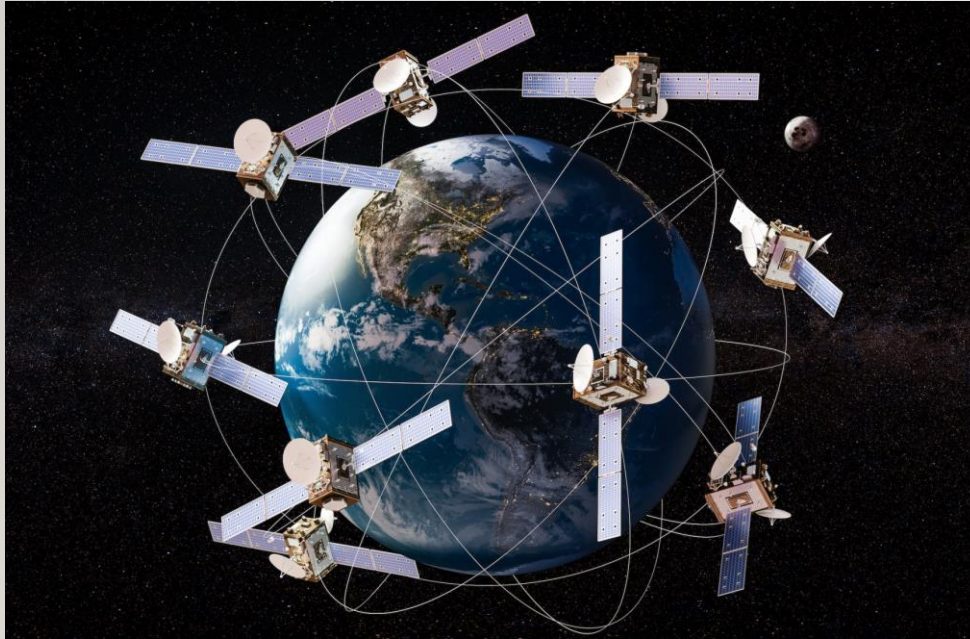
Doosan KG KAIROS 17; Rokey Boot Camp 1,2,3 7|





# EXAMPLE LARGE PROJECT EXPERIENCE (SW DEVELOPMENT)

---



# EXAMPLE SOLUTION PROJECT EXPERIENCE (APPLICATION)

---





# WHAT KIND OF LEADER I WANT TO BE “DEAD POET SOCIETY”

---

Andy  
Or  
“Captain”



NOW,

TELL US WHO YOU ARE,

WHAT IS YOUR OBJECTIVE OF JOINING THIS  
PROGRAM?

---

Interview each other and share to the team

# HOW TO WORK TOGETHER

---

- Participate, Participate, Participate!!!
- No long emails or Kakaotalk, prefer face to face
- Be open to suggestions and idea
- Be proactive (적극적), take initiative (주도적)
- HOW is as important as WHAT



# 일정 및 진행 내용

---

- 운영기간 : 10일간 진행
- 운영시간 : 9:30~18:30 (점심 12:30 ~13:30)
- 세부운영내용
  - 기본 개념부터 step by step으로 기능을 구현하여 쉽게 따라하면서 원리를 습득할 수 있도록 구성
  - 실습을 통해 직접 구현해보며 기술을 습득 - 모듈 단위로 완성된 코드 제공
  - 각 단계별 학습 내용을 통합하여 최종 프로젝트 완성

## 일정 및 진행 내용

구분	시간	내용	담당	교육방법
정규 교육	09:30 ~ 09:40	오전 출석 확인	각 담임조교	대면 교육
	09:40 ~ 11:00	프로젝트 - 1교시	교과강사 / 기술조교	
	11:10 ~ 12:30	프로젝트 - 2교시		
	12:30 ~ 13:30	점심시간		
	13:30 ~ 13:40	오후 출석 확인	각 담임조교	대면 교육
	13:40 ~ 15:00	프로젝트 - 3교시	기술조교	
	15:10 ~ 16:30	프로젝트 - 4교시		
	16:40 ~ 18:20	프로젝트 - 5교시		
	18:20 ~ 18:30	마무리 및 안내사항	각 담임조교	
비정규 교육*	18:30 ~ 19:30	저녁시간		
	19:30 ~ 21:00	자율 훈련		

# MATERIAL & ITEMS CHECK

---





# 프로젝트 주재, 계획, 및 평가

---



## 2 PROJECTS

---

- Mini Project (Individual Team)
  - For learning techniques

1	로봇 AI 시스템 개발 프로세스 이해	시스템 개발 프로세스의 이해 개발 환경 구축
2	프로젝트에 필요 기술 검증	AI VISION 기술 탐색 및 검증
3	프로젝트에 필요 기술 검증	로봇 AMR 제어 기술 탐색 및 검증
4/5	MINI 프로젝트	통합 시스템 설계 및 개발

## 2 PROJECTS

---

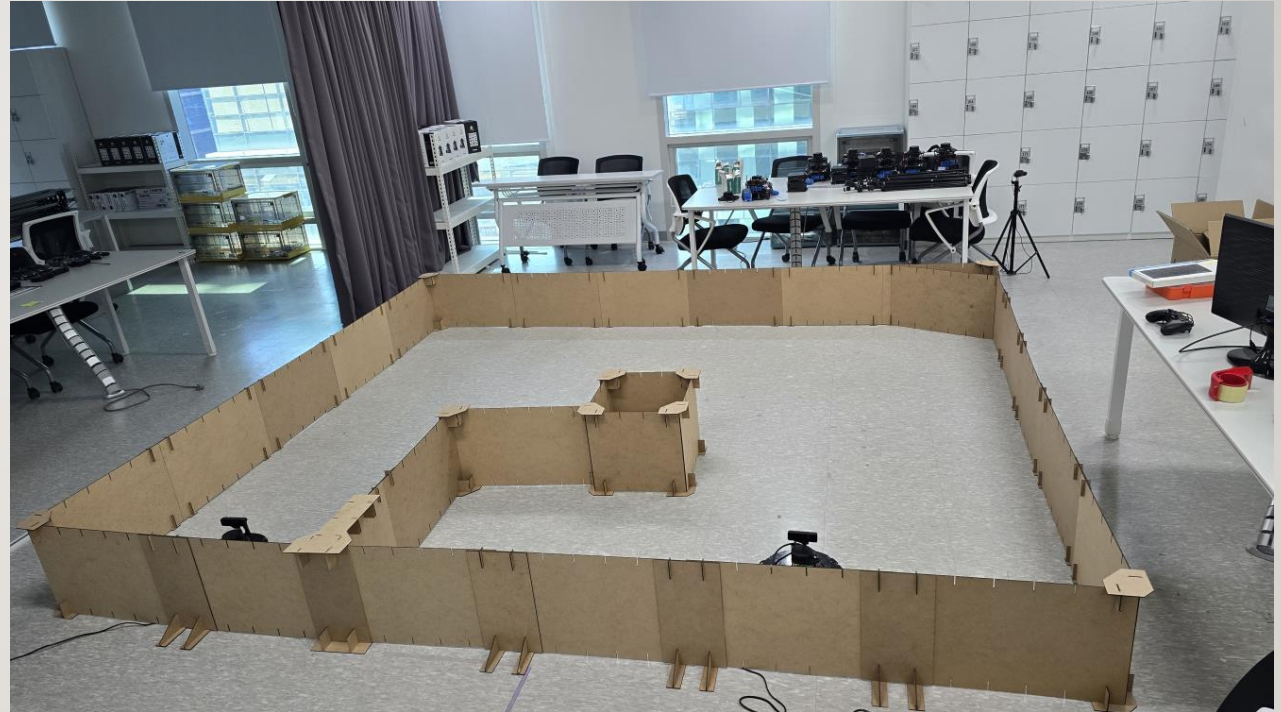
- Final Project (2 Teams in One)

6	프로젝트에 필요 기술 검증	웹 시스템 모니터 기술 탐색 및 검증
6-9	파이널 프로젝트	통합 시스템 설계 및 개발
10	최종 프레젠테이션 및 시연	시스템 발표 및 시연



## MINI PROJECT DESCRIPTION

---



DETECTION ALERT

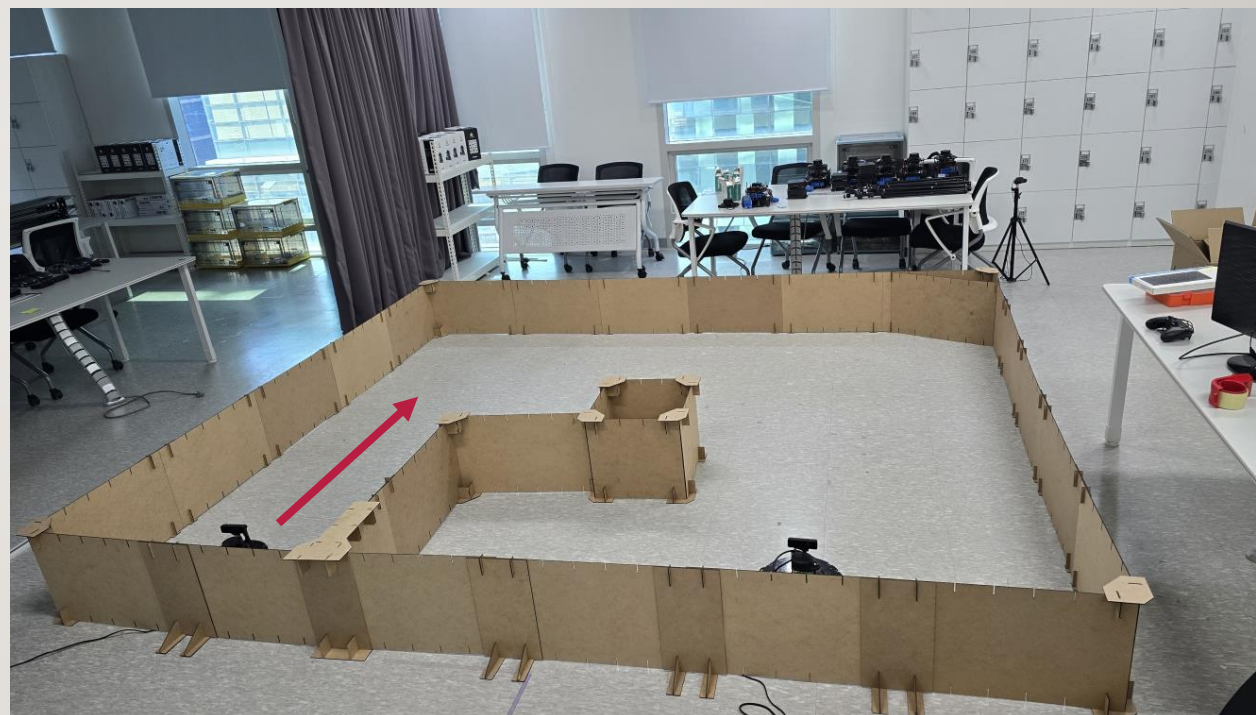
---





START

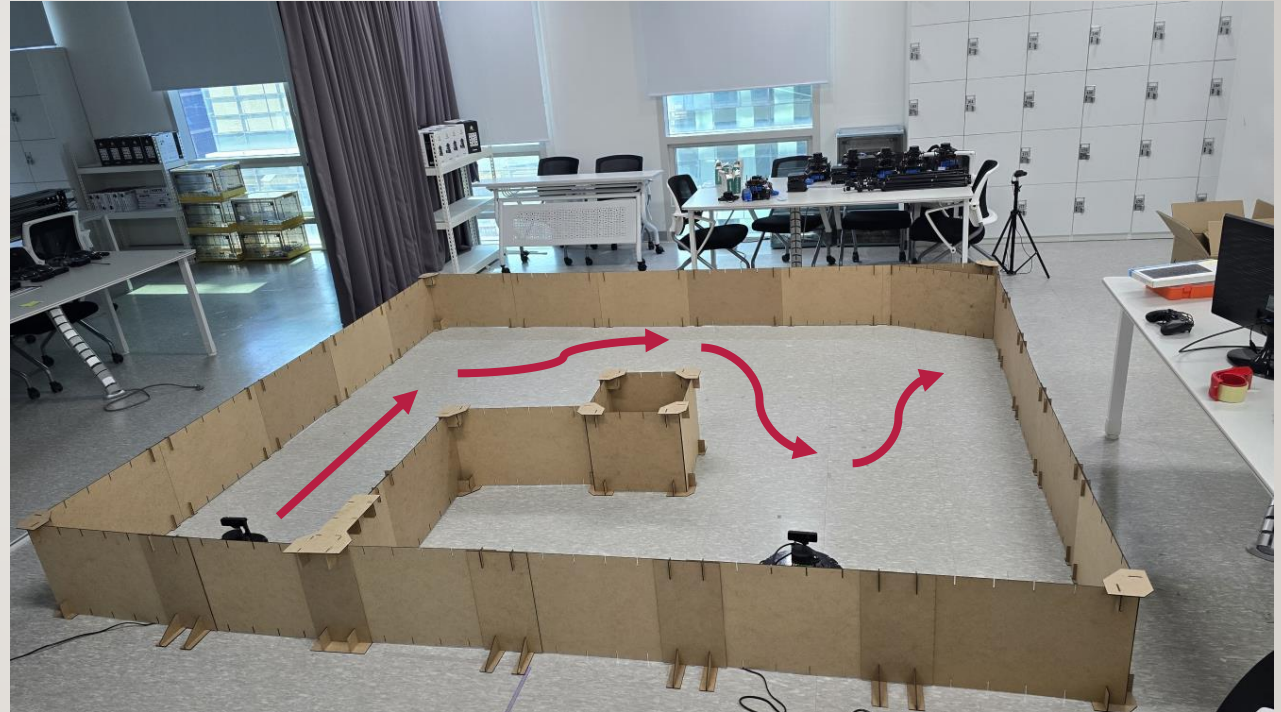
---





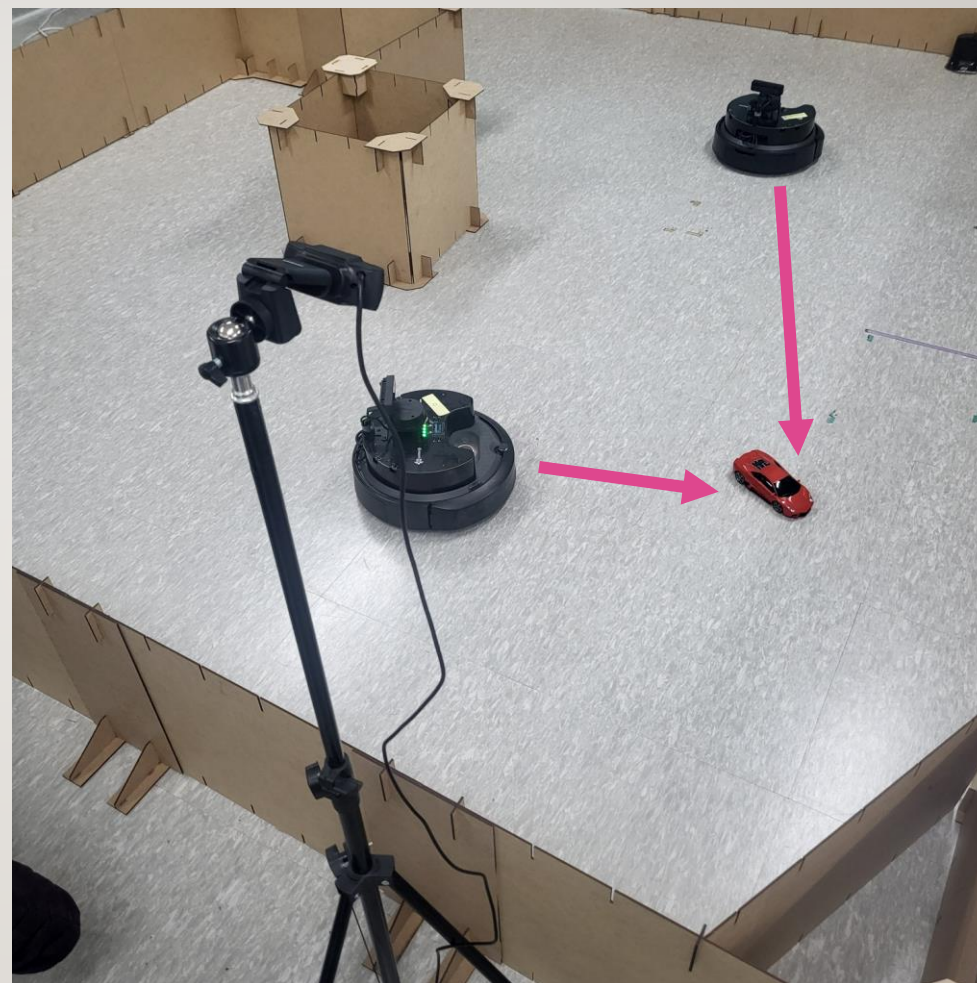
# NAVIGATE

---



# TRACK & FOLLOW

---



# ASSUMPTION ABOUT YOUR KNOWLEDGE

---

- OS
  - Linux
- Language
  - Python3
- Packages
  - ROS2
  - OpenCV
  - Yolo8
  - Flask
  - SQLite3
- Tools
  - Labellmg
  - VSCode
  - Git/Github



# DAY I

---

- Welcome
- Project Introduction
- Introduction to Project Development Process
- Business Requirement Development
- System Requirement Development
- Time Management
- System(High Level) Design

# DAY 2 (MINI PROJECT)

---

- Yolo객체 인식 모델 활용과 성능 평가 방법 이해
  - Custom Dataset과 Fine Tuning으로 자체 객체 인식 모델 구현 및 평가
  - (Optional)경량화 모델 등 개별 요구사항에 적합한 모델 탐색 및 성능 검증

# DAY 2 (MINI PROJECT)

---

## WEB-CAM 기반 객체 인식

### (IF NEEDED)

- YOLOv8 기반 데이터 수집/학습/deploy (Detection Alert)
  - 감시용 데이터 수집(rc\_car, dummy, 등)
  - 감시용 데이터 라벨링
  - YOLOv8 기반 학습
  - YOLOv8 Object Detection

## AMR-CAM 기반 객체 인식

- AMR(Autonomous Mobile Robot) Turtlebot4 개발 환경 구축
- 로봇 개발 환경에 완성 모델 서빙 및 테스트 / 로봇 H/W, 제반 환경의 한계점 도출
  - Tracking 데이터 수집((rc\_car, dummy, 등)
  - Tracking 데이터 라벨링
  - YOLOv8 기반 학습
  - YOLOv8 Object **Tracking**

# DAY 3 (MINI PROJECT)

---

- Auto. Driving 시스템 학습
  - Digital Mapping of environment
  - Operate AMR (Sim. & Real)
  - Tutorial 실행
  - Detection, Depth and AMR 주행
  - 로봇 개발 환경에 적용 및 테스트 / 로봇 H/W, 제반 환경의 한계점 도출

## TURTLEBOT4 시뮬레이션

- 환경 구축
- SLAM과 AutoSLAM으로 맵 생성
- Sim.Tutorial 실행
- Detection, Depth and AMR 주행 example



# DAY 3 (MINI PROJECT)

---

## REAL ROBOT

- Manually operating the AMR (Teleops)
- autonomous driving 시스템 with obstacle avoidance
  - Digital Mapping of environment
  - Launching Localization, Nav2, and using Rviz to operate a robot
  - Goal Setting and Obstacle Avoidance using Navigation

## TUTORIAL

- Turtlebot4 API를 활용한 Initial Pose Navigate\_to Pose 구현
- Turtlebot4 API를 활용한 Navigate\_Through\_pose, Follow Waypoints 구현

# DAY 4

---

- Business Requirement Development
- System Requirement Development
- Time Management
- System(High Level) Design
- Begin Detail Design to Acceptance - Agile Development (SPRINTs)

# DAY 4 (MINI PROJECT)

---

## SYSTEM DESIGN

- Mini Project

## DETAIL DESIGN

- Detection
- AMR Control



# DAY 5 (MINI PROJECT)

---

## CODING, TEST & INTEGRATION

- Coding and Test all modules
- Porting to ROS
- And finally, Integration and Test of Detection Alert & AMR Controller

## MINI PROJECT DEMO

- Prepare and demo completed project

# DAY 6 (FINAL PROJECT)

---

- 시스템 설계 및 프로세스 정립
- 비즈니스 요구 사항 업데이트
- 역할 분담 및 일정 조율
- 개발 환경 구축(맵 디자인, SW 개발, 문서 통합 관리)
- 멀티 로봇 환경 구축 및 네비게이션
- 멀티 로봇 개별 업무 수행
- 멀티 로봇 협동 업무 수행
- (Optional) Turtlebot4 각종 센서 데이터의 이해와 적용

# DAY 6 (FINAL PROJECT)

---

- Flask 를 이용한 웹 서버 구축 (System Monitor)
  - Flask/HTML Intro
  - Deploy YOLOv8 Obj. Det results to web
  - Log in 기능 구현
  - Sysmon 웹기능 구현
- SQLite3를 이용한 데이터베이스 구축 및 연동 (System Monitor)
  - SQLite3 기본 기능 구현
  - DB 기능 구축
  - 저장된 내용 검색하는 기능 구현



# DAY 7 (FINAL PROJECT)

---

- 시스템 설계에 기반한 객체 감지 모델 구현
  - 로봇 환경에 적용 및 Unit Test
  - 모듈로 제작하고 launch파일로 구현
  - code 정리 및 버전관리, 문서 작성 및 영상 촬영, 팀 내 기술 브리핑
- 시스템 설계에 기반한 SysMon 설계 구현
  - 로봇 환경에 적용 및 Unit Test
  - 모듈로 제작하고 launch파일로 구현
  - code 정리 및 버전관리, 문서 작성 및 영상 촬영, 팀 내 기술 브리핑

# DAY 8 (FINAL PROJECT)

---

- 시스템 설계에 기반한 AMR 제어 구현
- 로봇 환경에 적용 및 Unit Test
- 모듈로 제작하고 launch 파일로 구현
- code 정리 및 버전관리, 문서 작성 및 영상 촬영, 팀 내 기술 브리핑

# DAY 9 (FINAL PROJECT)

---

- 개별 기능 통합 구현 및 Integration 테스트
- 통합 Launch 파일로 구현
- Robust한 시스템 구축을 위한 예외 처리 및 Code Refactoring
- code 정리 및 버전관리, 문서 작성 및 영상 촬영, 팀 내 기술 브리핑



# DAY 10 (FINAL PROJECT)

---

- 프로젝트 발표 및 시연
- 최종 산출문 정리(소스코드, 발표 PPT, 동작 영상)
- 팀 간 기술 컨퍼런스를 통한 기술 극복 경험담, 노하우 교류(채점 대상X)

# PROJECT 평가

## 6. 평가 방법(★★★ 매우중요)

### ① 평가 구성(실무 프로젝트 교육)

평가 항목	평가 방법	평가 시기	평가 횟수	평가 주체
기술 역량 평가	<ul style="list-style-type: none"><li>· 별도 제공된 양식에 따라 각 그룹별 평가</li><li>· 조 단위 평가(개인 평가X)</li><li>· 교강사가 직접 평가표 작성 후 운영팀에 제출</li></ul>	각 그룹별 수업 종료 후	6회	교강사
조직 역량 평가	<ul style="list-style-type: none"><li>· 별도 구글 설문 시트를 통한 평가</li><li>· 동료/개인 평가</li></ul>	실무 프로젝트 교육 전체 종료 후	1회	교육생

※ 『조직 역량 평가』 운영은 "운영팀"에서 별도 운영 합니다.

# PROJECT 평가

## ▶ 평가 항목

평가 항목	평가기준
기능 구현 완전성	* 교육에서 요구하는 기능이 모두 구현되어 있는지 여부를 평가한다.
기능 구현 정확성	* 구현된 모든 기능들이 정상적으로 동작하는지 여부를 평가한다.
동작 및 운용 안정성	* 산출물 운용시 안정적으로 동작하는지 여부를 평가한다.
입출력 데이터 이해도	* 데이터 입출력 방법 및 절차가 편리하고 기능 요구 내용에 적합한 지 여부를 평가한다.
기능 동작 지속성	* 장애/오류 발생 시에도 지속적인 동작/운영이 가능한지 여부를 평가한다.



# PROJECT 평가

---

Assessment/Group	
Mini Project	
1. 비즈니스 요구 사항 작성	5
2. 시스템 요구 사항 작성	5
3. Process Flow Diagram을 사용하여 시스템 설계를 생성	5
4. Detection Alert Module 상세설계 수행	5
5. Detection Alert Module의 코딩 및 테스트 수행	5
6. AMR Controller Module 상세설계 수행	5
7. AMR Controller Module의 코딩 및 테스트 수행	5
8. Detection Alert 및 AMR Controller Module의 통합 및 테스트 수행	5
Mini Project Sub Total	40

9. 최종 프로젝트	
System Monitor Module 상세설계 수행	5
System Monitor Module의 코딩 및 테스트 수행	5
completeness	10
accuracy (object Det. / Goal arrival)	10
safety (obj. avoidance)	10
10. 모든 모듈의 시스템 통합 및 테스트 수행	10
11. 최종 프로젝트 발표	10
12. System Design Doc	10
Final Project Sub Total	70

# 프로젝트 RULE

---

80/20 → 20/80

# TEAMWORK AND PROJECT MANAGEMENT

---





프로젝트 RULE NUMBER ONE!!!

---

Have Fun Fun Fun!



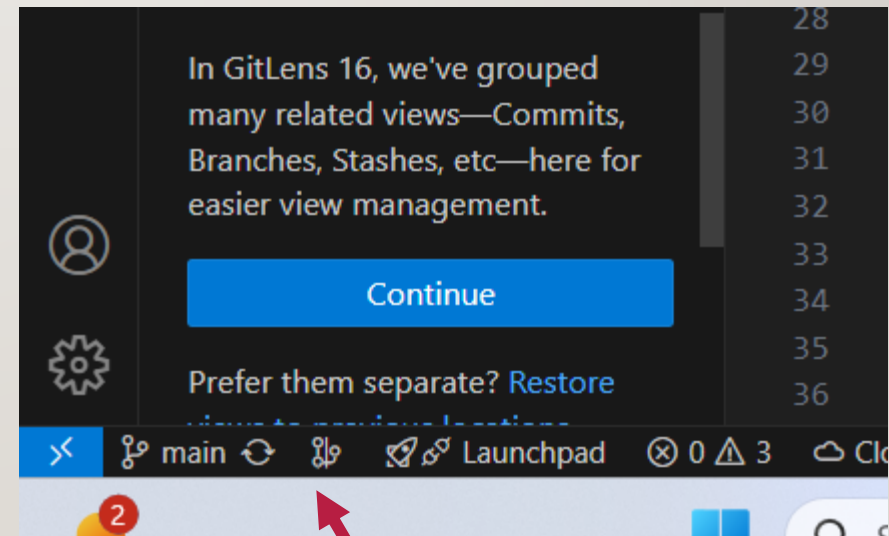
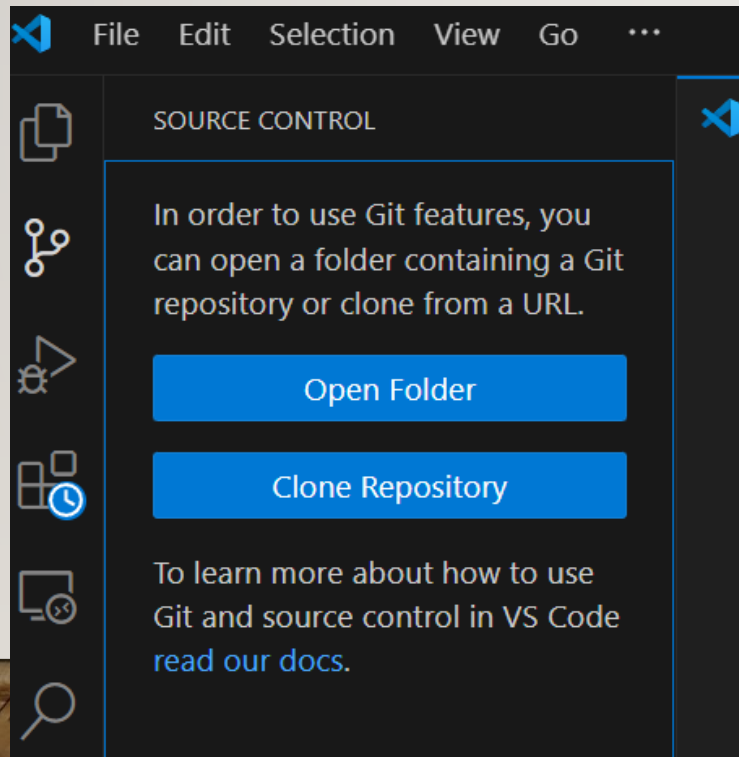


Let's Get Started!!!!

# PUBLIC REPOSITORY FOR CLASS

---

\$ Git Clone [https://github.com/kimandreas/to\\_students](https://github.com/kimandreas/to_students)









# NOTION FOR SETUP SCRIPTS

---

- [Doosan Rokey 37| project](https://indecisive-freedom-6e8.notion.site/Doosan-Rokey-3-project-1d98e215779c80e4a835f4b307f19273)
- <https://indecisive-freedom-6e8.notion.site/Doosan-Rokey-3-project-1d98e215779c80e4a835f4b307f19273>

## Doosan Rokey 37| project

-  [PC System Setup](#)
-  [YOLO Setup](#)
-  [Turtlebot4 PC Setup](#)
-  [Turtlebot4 Setup](#)
-  [Multi Robot Setup](#)
-  [Transform Explained](#)

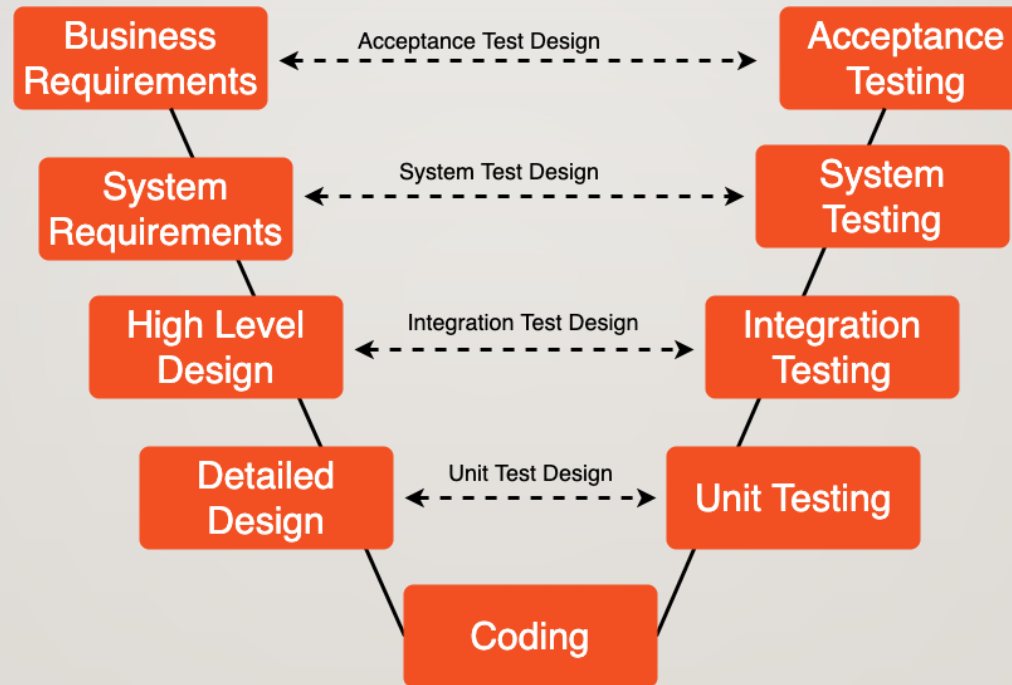
PROJECT DEVELOPMENT  
IS A PROCESS

---



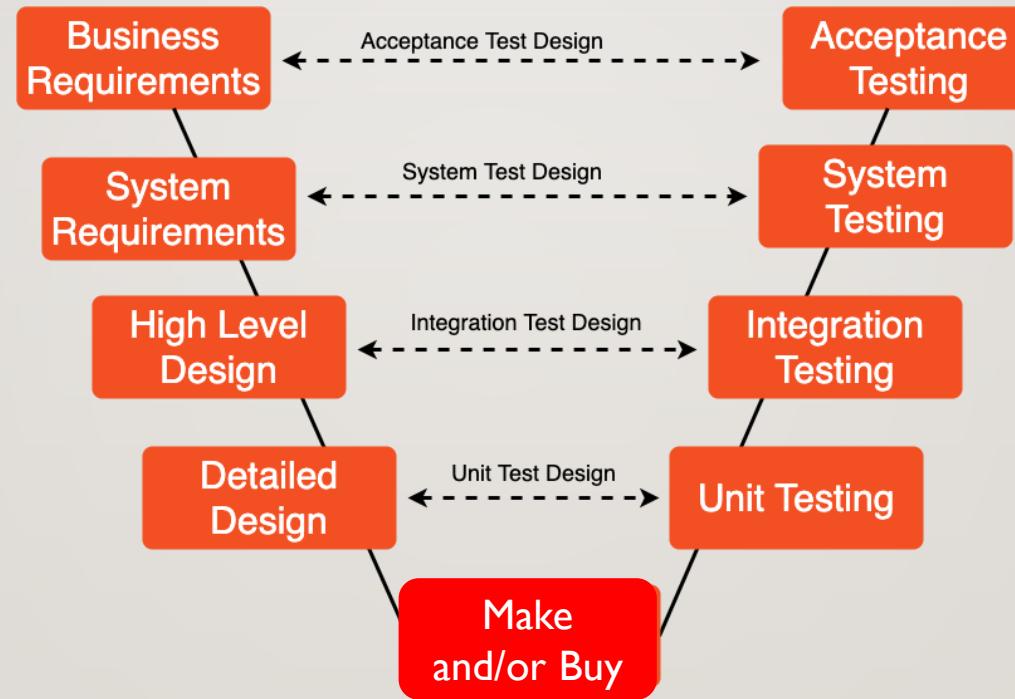


# SW DEVELOPMENT PROCESS



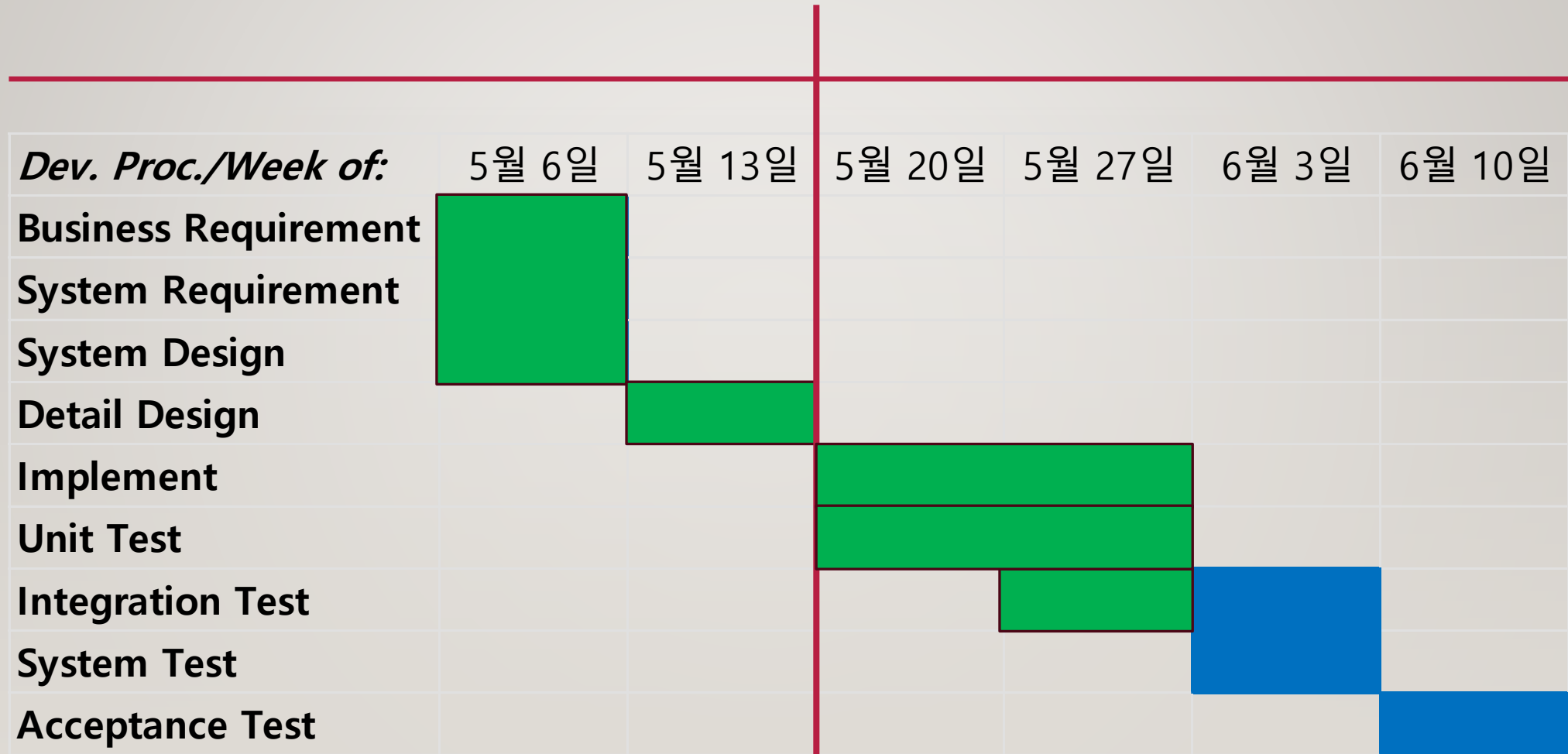
SDLC - V Model - notepub.io

# HW DEVELOPMENT PROCESS



SDLC - V Model - notepub.io

# EXAMPLE TIMELINE



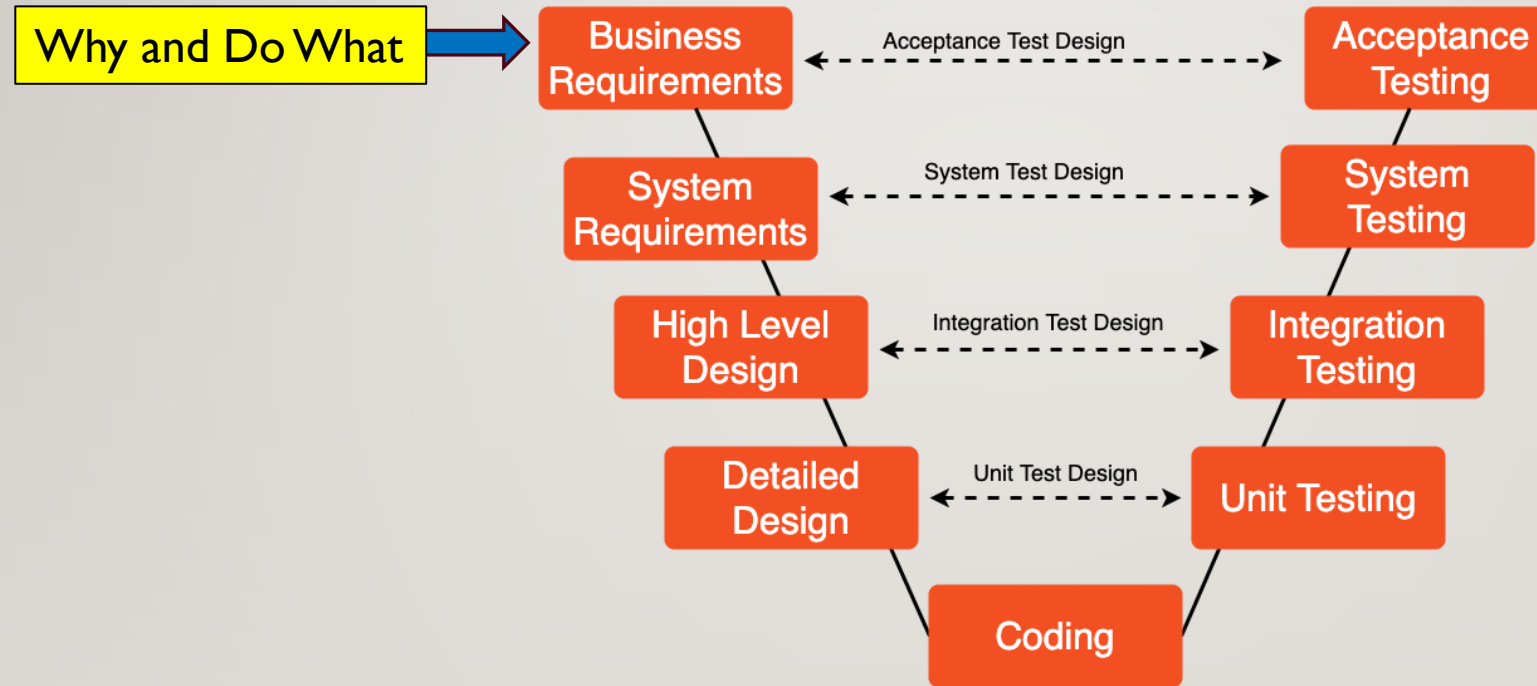
# MINI PROJECT DEVELOPMENT PROCESS

---





# SW DEVELOPMENT PROCESS



SDLC - V Model - notepub.io

# PROJECT/SOLUTION JUSTIFICATION (WHY)

---

- Situation Analysis
  - evaluates both external and internal factors to determine the necessity and feasibility of a project. It helps justify resource allocation by outlining how the project aligns with strategic goals, identifying potential challenges and opportunities, and providing a detailed understanding of the project's context for informed decision-making.
- 상황 분석
  - 프로젝트의 필요성과 타당성을 결정하기 위해 외부 및 내부 요인을 모두 평가합니다. 프로젝트가 전략적 목표에 어떻게 부합하는지 설명하고, 잠재적인 과제와 기회를 식별하고, 정보에 입각한 의사 결정을 위해 프로젝트의 컨텍스트에 대한 자세한 이해를 제공하여 리소스 할당을 정당화하는 데 도움이 됩니다.

# PROJECT/SOLUTION JUSTIFICATION (WHY)

---

- Business Needs/Pain Point Analysis

- identifies and assesses the problems and unmet needs of customers. This process helps businesses tailor their solutions to enhance customer satisfaction and loyalty by directly addressing these issues.

- 비즈니스 니즈/문제점 분석

- 문제와 충족되지 않은 요구를 식별하고 평가합니다. 이 프로세스는 기업이 이러한 문제를 직접 해결하여 고객 만족도와 충성도를 높일 수 있도록 솔루션을 맞춤화하는 데 도움이 됩니다.

# ADVANCED TECHNIQUES THAT WE HAVE

---

- AI Object Detection
- AMR
  - Navigation with obstruction avoidance
- ROS2



BRAINSTORM A SITUATION THAT WILL  
BENEFIT FROM THIS  
SOLUTION/TECHNIQUES

---



# BRAINSTORMING RULES

---

- Every input is good input
- Do not critique inputs only seek to understand
- Organize inputs into logical groupings
- Sequence or show relationships as needed
- Use Posted Notes on Flip Chart



# BUSINESS REQUIREMENT (WHAT EXAMPLE)

---

- **Business Requirements with Metrics:** The company aims to deploy a robotic system integrated with a deep learning model to automate quality inspection in manufacturing. The goal is to reduce human error by achieving 98% accuracy in defect detection and increase production efficiency by minimizing inspection time to under 2 seconds per item.
- 이 회사는 딥 러닝 모델과 통합된 로봇 시스템을 배포하여 제조 시 품질 검사를 자동화하는 것을 목표로 합니다. 목표는 결함 감지에서 98%의 정확도를 달성하여 인적 오류를 줄이고 검사 시간을 품목당 2초 미만으로 최소화하여 생산 효율성을 높이는 것입니다.

# TEAM EXERCISE I

---

Brainstorm Business Requirement for the project and write business requirement statement

Using the posted notes and flipchart as needed



# BUSINESS REQUIREMENT PRESENTATION BY EACH TEAM

---

Using the posted notes and flipchart as needed

# PUBLIC REPOSITORY FOR CLASS

---

\$ Git Clone [https://github.com/kimandreas/to\\_students](https://github.com/kimandreas/to_students)

# EXAMPLE BUSINESS REQUIREMENT DOCUMENT

## Business Requirements Document (BRD)

**Project Title:** Autonomous Mobile Robot (AMR) Security System

**Project Owner:** Kim Andreas

**Date:** [Insert Date]

**Version:** 1.0

### 1. Business Objectives

Implement an AI-powered robotic security solution using Autonomous Mobile Robots (AMRs) to monitor and safeguard a secure area, reducing reliance on human security personnel, increasing operational efficiency, and minimizing security risks.

### 2. Project Scope

Develop and deploy an AMR-based security system to perform real-time surveillance, threat detection, and autonomous response in restricted or

프로젝트 제목: 자율 이동 로봇(AMR) 보안 시스템

↓

프로젝트 소유자: 김 안드레아스

날짜: [날짜 삽입]

버전: 1.0

### 1. 비즈니스 목표

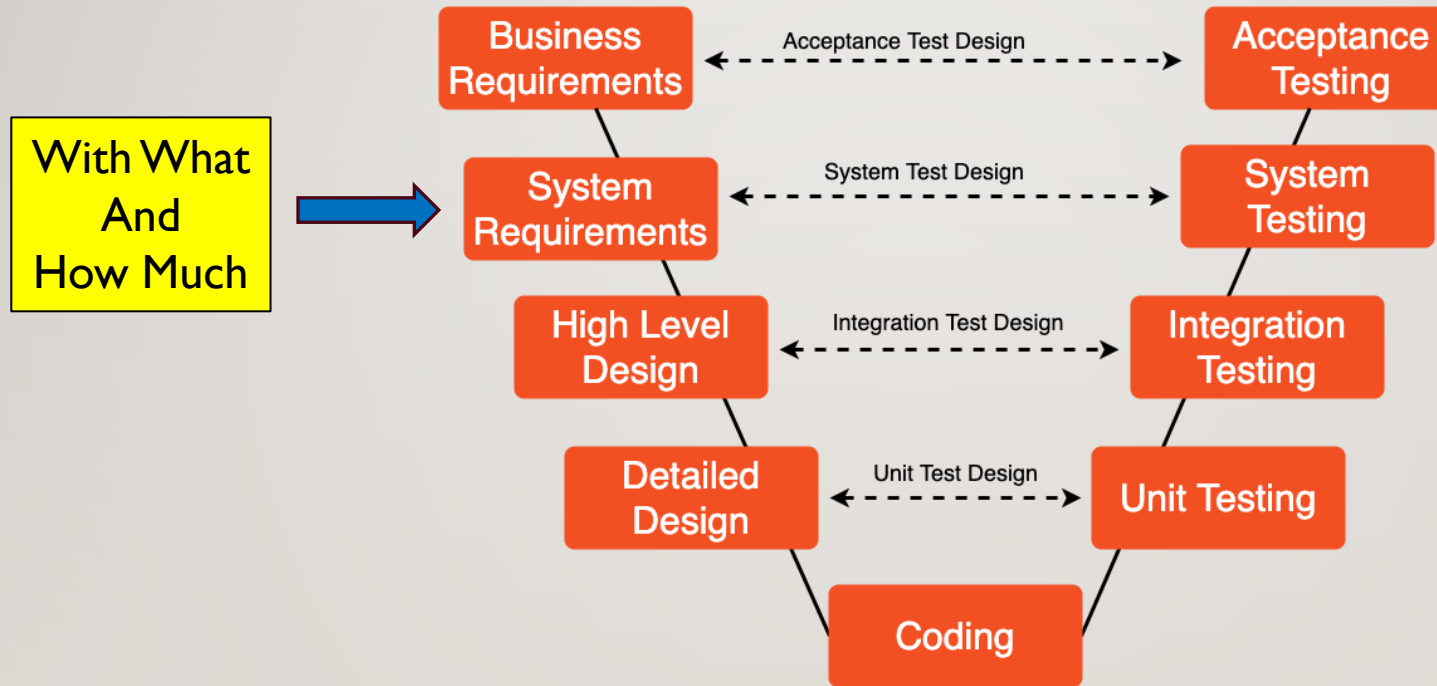
인공지능 기반 자율 이동 로봇(AMR)을 활용하여 보안 지역을 감시하고 보호하는 로봇 보안 솔루션을 구현하여, 인력 보안 의존도를 줄이고 운영 효율성을 높이며 보안 위험을 최소화합니다.

### 2. 프로젝트 범위

제한된 또는 민감한 지역에서 실시간 감시, 위험 탐지 및 자율 대응을 수행하는 AMR 기반 보안 시스템을 개발하고 배포합니다. 주요 기능에는 순찰, 이상 탐지, 위험 대응, 경고 상승이 포함됩니다.

포함 범위:

# SW DEVELOPMENT PROCESS



SDLC - V Model - notepub.io



# SYSTEM/TECHNICAL REQUIREMENT (EXAMPLE)

---

## Technical Requirements and Metrics:

- 1. Deep Learning Models:** Train a CNN to achieve at least 98% accuracy on defect detection in validation datasets.

검증 데이터 세트에서 결함 감지에 대해 최소 98%의 정확도를 달성하도록 CNN을 훈련시킵니다.

- 2. Robotics Hardware:** Ensure the robot processes images and delivers results within 2 seconds per item, with 99.9% system uptime.

로봇이 99.9%의 시스템 가동 시간으로 항목당 2초 이내에 이미지를 처리하고 결과를 제공하도록 보장합니다.



# SYSTEM/TECHNICAL REQUIREMENT (EXAMPLE)

---

## Technical Requirements and Metrics:

- 3. Interface and Control System:** Design for less than 0.1% downtime and a response time under 1 second for user interactions.

다운타임이 0.1% 미만이고 사용자 상호 작용에 대한 응답 시간이 1초 미만으로 설계됩니다.

# TEAM EXERCISE

---

Brainstorm System Requirement for the project and document

Using the posted notes and flipchart as needed

# HOW MUCH

---





# YOUR PROJECT ENVIRONMENT

---



DETECTION ALERT

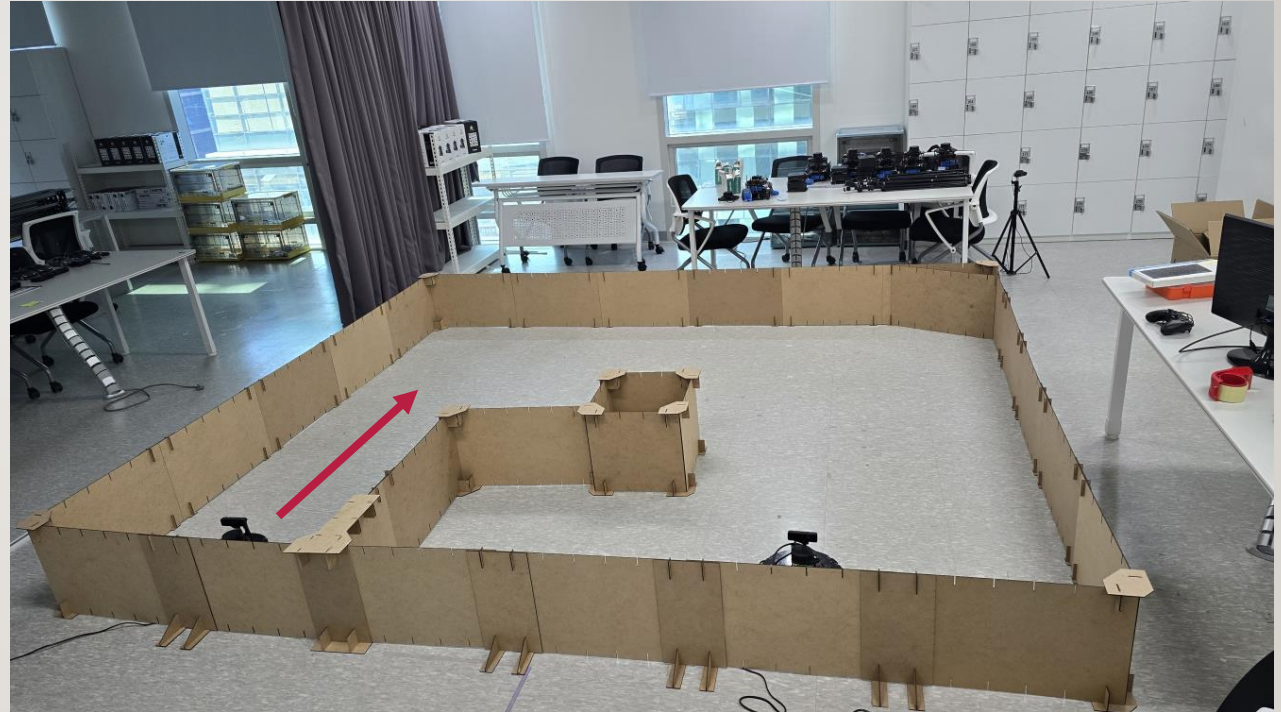
---





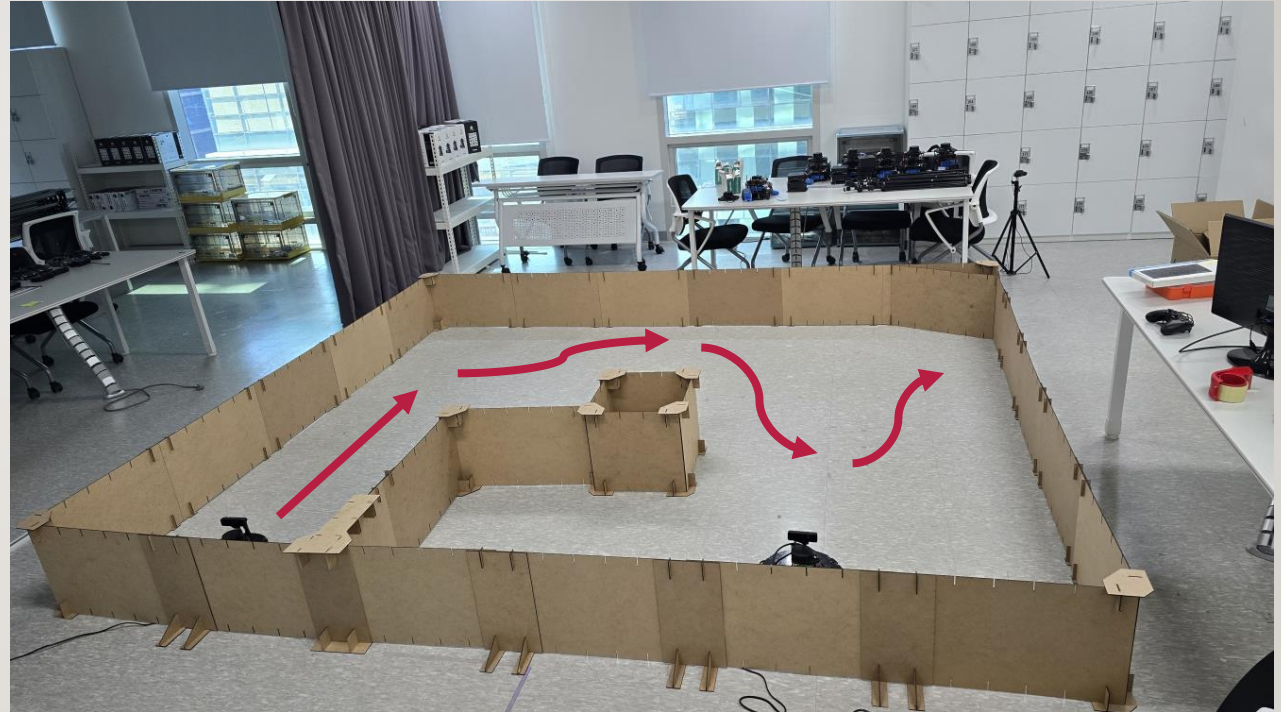
START

---



# NAVIGATE

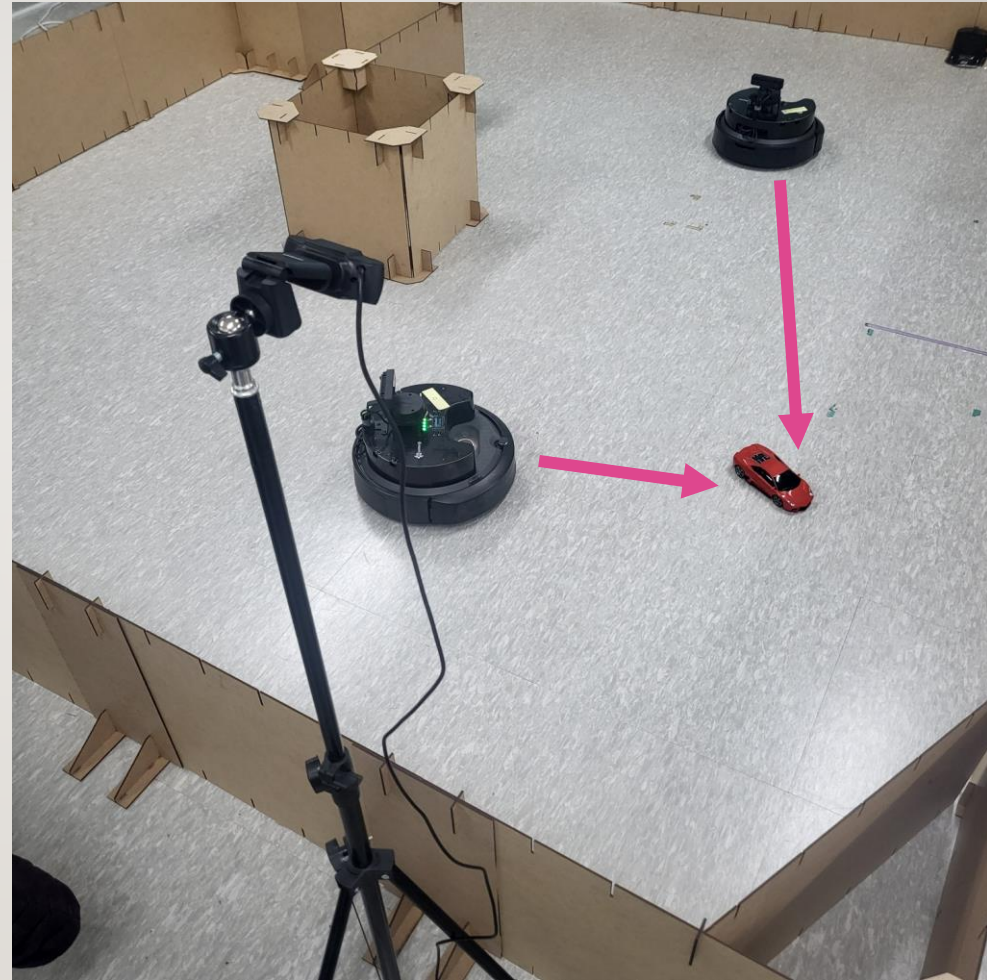
---





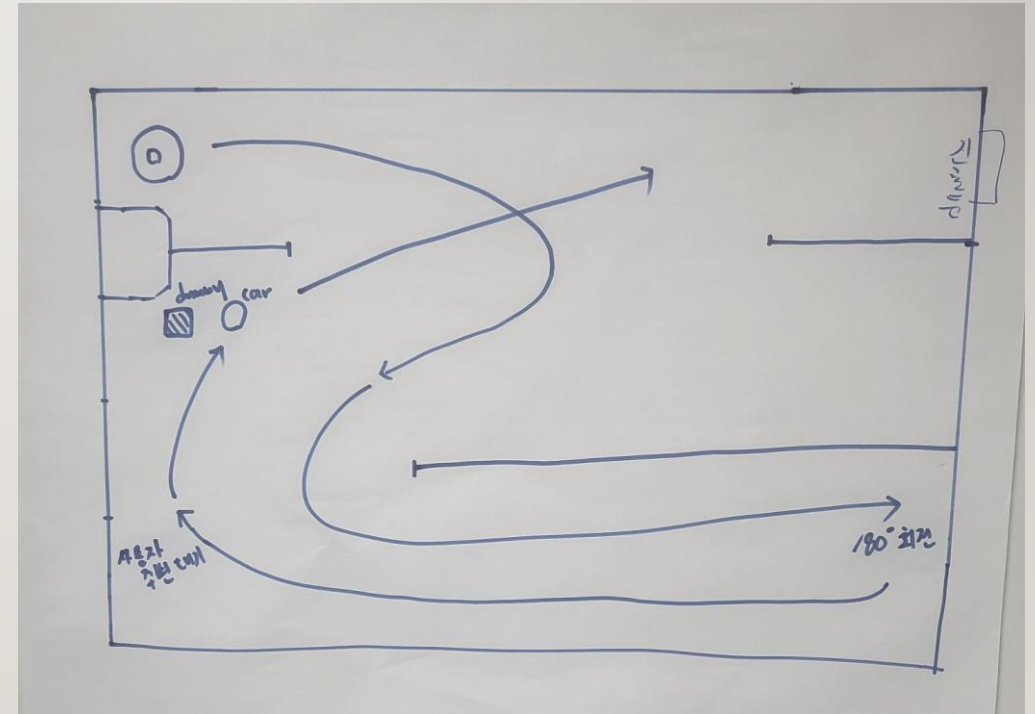
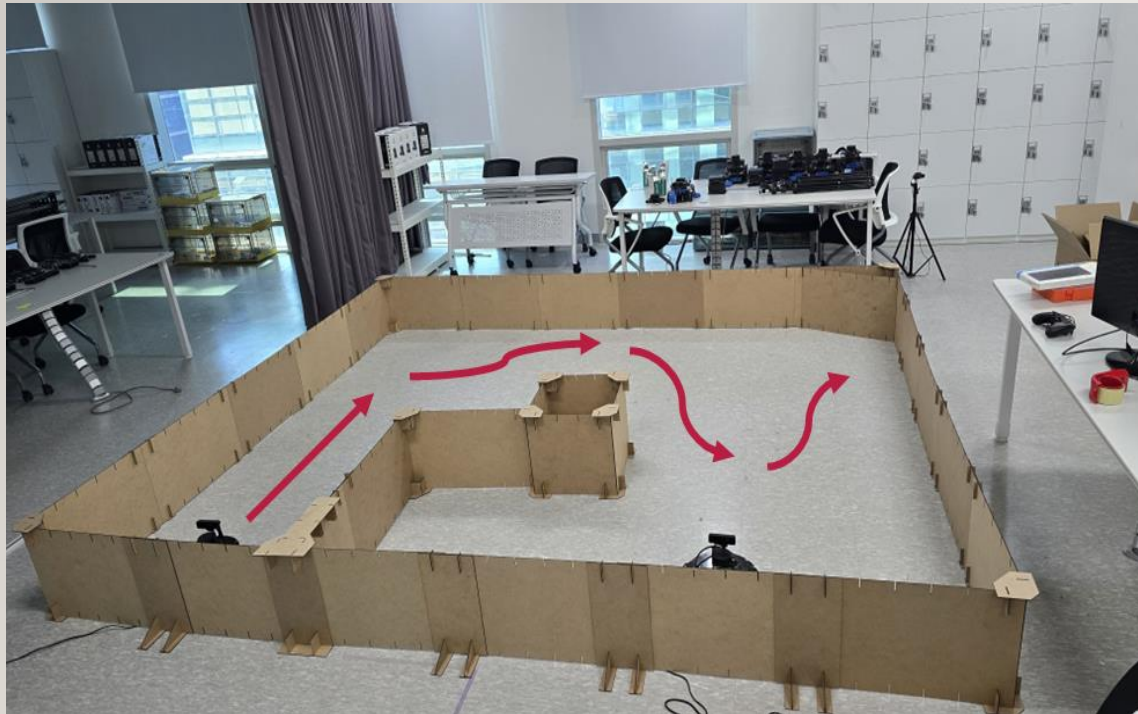
# TRACK & FOLLOW

---



# EXERCISE: SKETCH YOUR SCENARIO

---



# DEVELOP YOUR BUSINESS SCENARIO (USE-CASE) PROCESS DIAGRAM

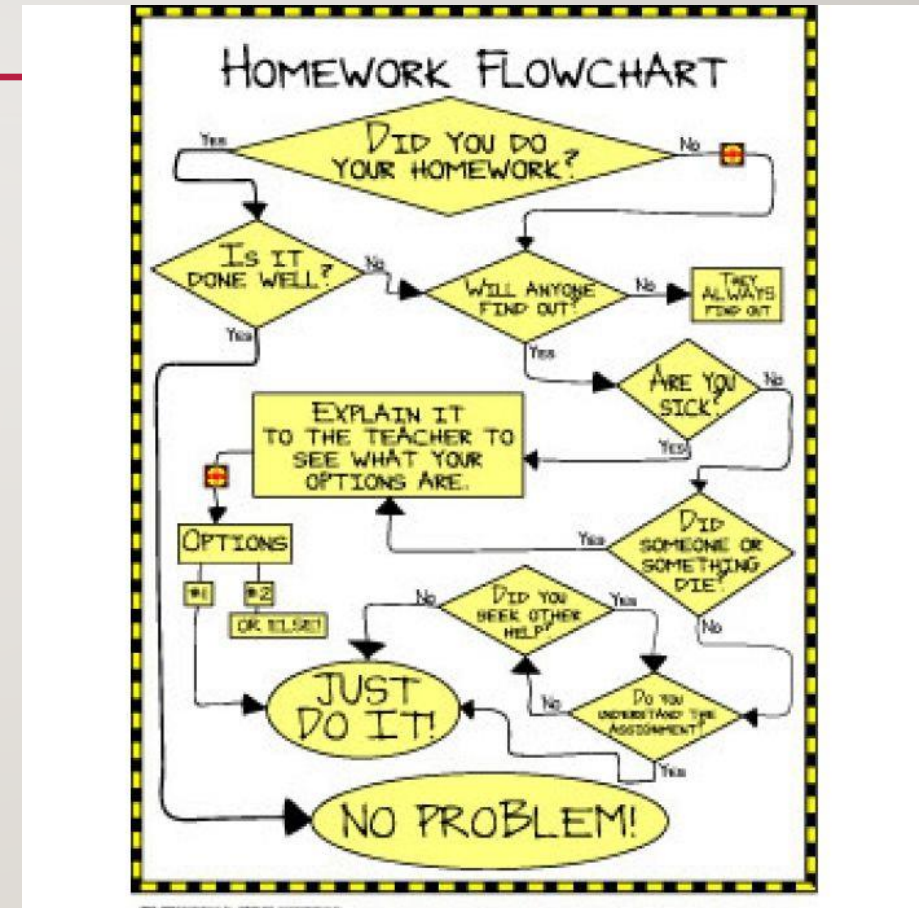
---

Using the posted notes and flipchart as needed



# VISUALIZATION – SCENARIO PROCESS DIAGRAMS

- As-Is Functional Process Diagram
  - Current states
- To-Be Functional Process Diagram
  - Future states
- [Untitled Diagram - draw.io](#)
- <https://app.diagrams.net/>





Process

Decision

Document

Data

Sub  
Process

Start

Data

Card

Manual  
Operation

Preparation

Loop Limit

\_\_\_\_\_

\_\_\_\_\_

paper Tape

Terminator

Display

Sequential  
Data

WITH WHAT

---



# BASE HW/OS

---

- PC

- Ubuntu 22.04
- USB Camera



- Network
  - Wifi



- AMR

- TurtleBot4
- Ubuntu 22.04



# OBJ. DET.

---

TARGET



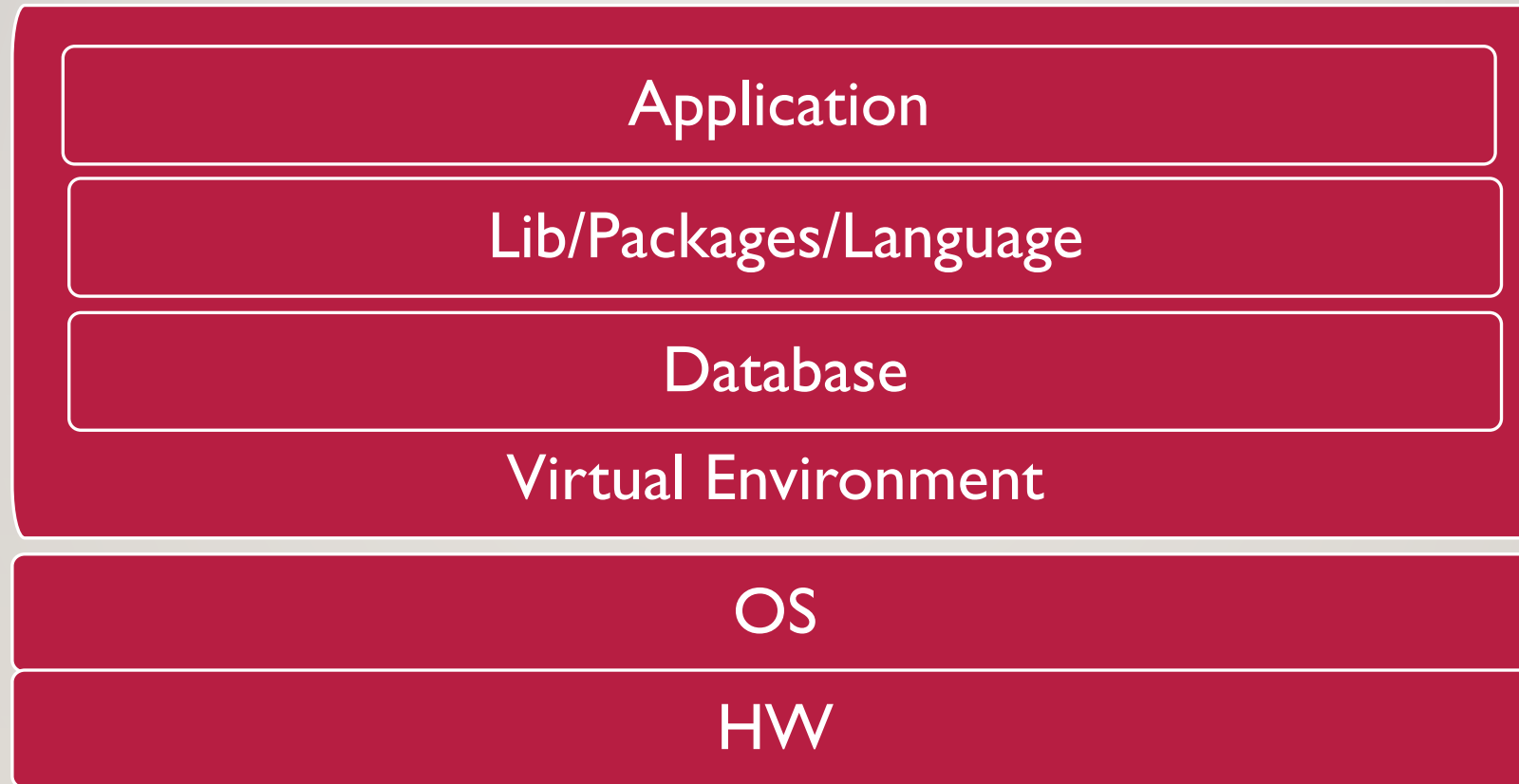
DUMMY





# EXAMPLE SYSTEM STACK

---



# BASE SYSTEM PACKAGES

---

## PC

- Python3
- ROS2
- Opencv
- Ultralytics
- Flask
- SQLite3

## AMR

- Python3
- ROS2

# AMR (TURTLEBOT4)

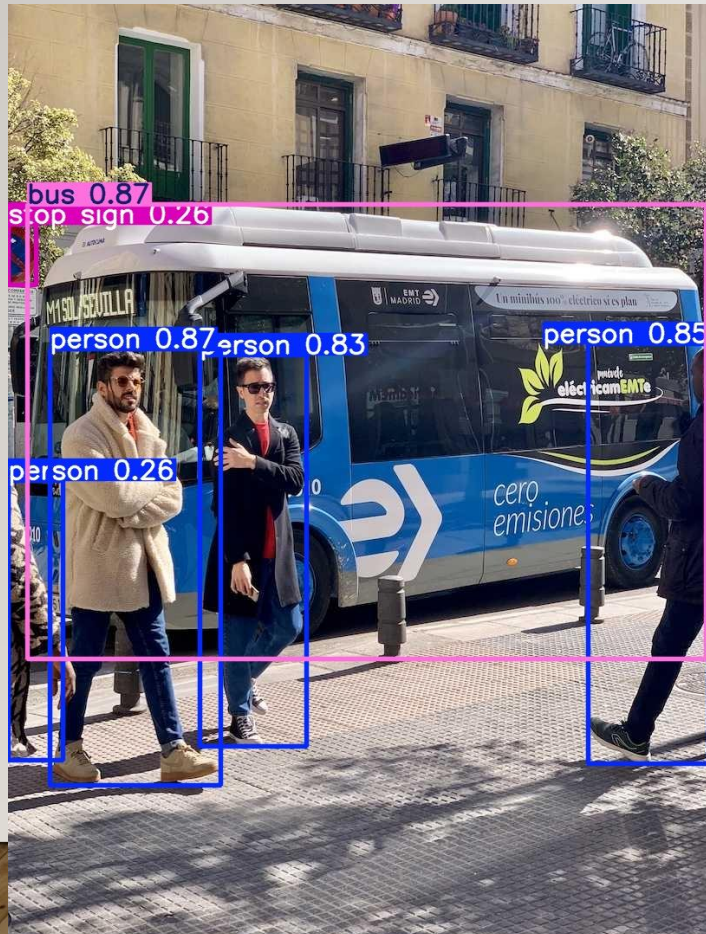
---

- [Features](#) · [User Manual](#)
- <https://turtlebot.github.io/turtlebot4-user-manual/overview/features.html>
- Review the content

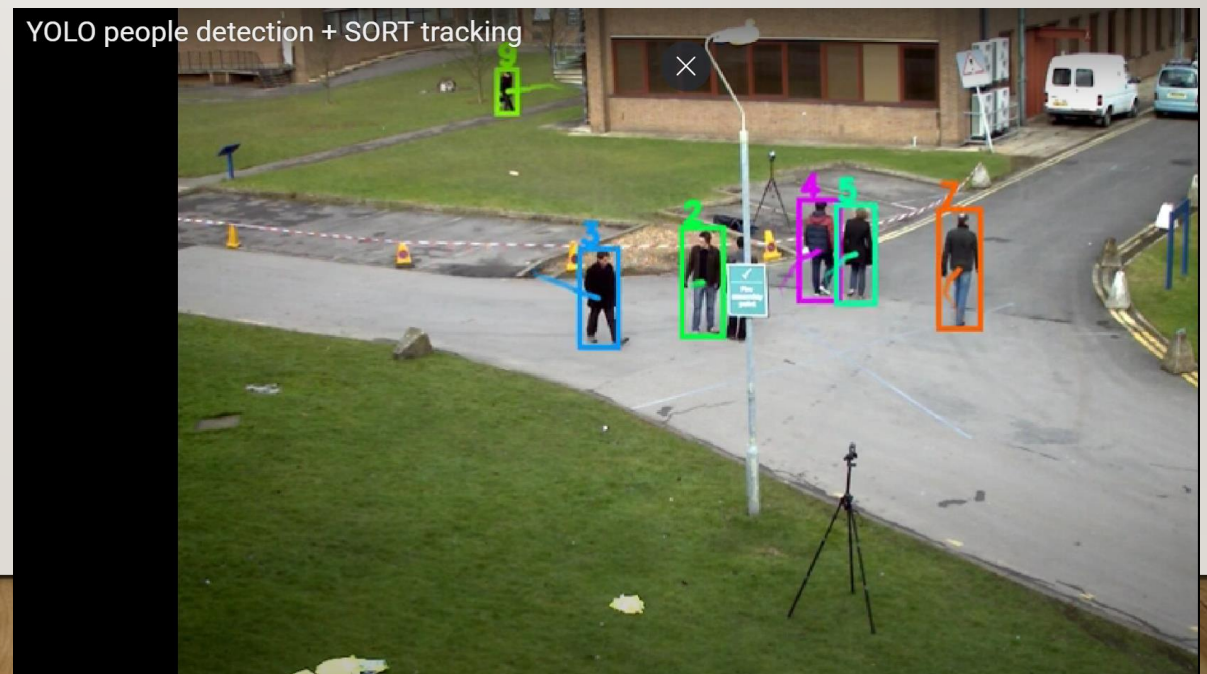




# YOLO OBJ. DET. VS. YOLO TRACKING



- [Track - Ultralytics YOLO Docs](#)
  - [\(469\) YOLO people detection + SORT tracking – YouTube](#)
  - [Bing Videos](#)





# TEAM EXERCISE 2

---

Brainstorm **Updated** System Requirement for the project and document

Using the posted notes and flipchart as needed

Include where, when, what will be used

# SYSTEM REQUIREMENT PRESENTATION BY EACH TEAM

---

Using the posted notes and flipchart as needed

# EXAMPLE SYSTEM REQUIREMENT DOCUMENT

## System Requirements Document (SRD)

↓

**Project Title:** Autonomous Mobile Robot (AMR) Security System↓

**Version:** 1.0↓

**Date:** [Insert Date]

### 1. Introduction

The system requirements define the technical specifications for developing and implementing the AMR-based security solution. This includes hardware, software, networking, and integration requirements.

### 2. System Overview

This system will provide autonomous patrolling, threat detection, and reporting for secure areas using AI-enabled Autonomous Mobile Robots (AMRs). It integrates navigation, sensor data processing, real-time alerts, and user interface management.

## 시스템 요구사항 문서 (SRD)

**프로젝트 제목:** 자율 이동 로봇(AMR) 보안 시스템↓

**버전:** 1.0↓

**날짜:** [날짜 삽입]

### 1. 소개

이 시스템 요구사항 문서는 AMR 기반 보안 솔루션의 개발 및 구현을 위한 기술 사양을 정의합니다. 여기에는 하드웨어, 소프트웨어, 네트워킹, 통합 요구사항이 포함됩니다.

### 2. 시스템 개요

이 시스템은 AI 기반 자율 이동 로봇(AMR)을 사용하여 보안 구역의 자율 순찰, 위험 탐지 및 보고를 제공합니다. 네비게이션, 센서 데이터 처리, 실시간 경고 및 사용자 인터페이스 관리를 통합합니다.

# PROJECT TIMELINE/CRITICAL PATH ITEM MANAGEMENT

---





# EX. IMPLEMENTATION TIMELINE

Function Backlog	Owner	5월 20일	5월 21일	5월 22일	5월 23일	5월 24일	5월 25일
<b>Unloading Module</b>	John						
Input1	John						
Input2	John						
Output 1	John						
Unit Test	John						
<b>Receiving Module</b>	Jan						
Input1	Feb						
Input2	Mar						
Output 1	Apr						
Unit Test	John						
Integration Test	John/Jan						

이 타임라인을 생성할 때  
먼저 시스템 및 시스템  
설계의 기능 프로세스  
다이어그램(To-Be)을  
완료해야 합니다.

그런 다음 각 기능(하위  
함수/모듈 및  
입력/출력)에 대해 누가,  
무엇을, 언제, 어떻게  
정의합니다. 표에 설명  
타임라인 형식의 무엇을,  
누가, 언제를 입력합니다.

# CRITICAL PATH ITEMS LIST

---

- tasks that directly impact the project timeline. Delays in these tasks would delay the project's overall completion because they represent the longest stretch of dependent activities
- 프로젝트 타임라인에 직접적인 영향을 주는 작업입니다. 이러한 작업이 지연되면 종속 활동이 가장 길어지기 때문에 프로젝트의 전체 완료가 지연됩니다

# CRITICAL PATH ITEMS LIST

---

- Examples:
  1. **AI Model Development:** Fine-tuning deep learning models like CNNs for precise item recognition and sorting, requiring data gathering, model training, and testing.
  2. **Robot Integration:** Embedding AI software into robots to enable precise task execution, focusing on software-hardware compatibility and function tests.
  3. **System Testing:** Comprehensive testing of AI and robot performance in simulated environments to ensure operational reliability.
- AI 모델 개발: 정확한 항목 인식 및 정렬을 위해 CNN과 같은 딥 러닝 모델을 미세 조정하여 데이터 수집, 모델 학습 및 테스트가 필요합니다.
- 로봇 통합: AI 소프트웨어를 로봇에 내장하여 소프트웨어-하드웨어 호환성 및 기능 테스트에 중점을 두고 정확한 작업 실행을 가능하게 합니다.
- 시스템 테스트: 시뮬레이션 환경에서 AI 및 로봇 성능을 종합적으로 테스트하여 운영 안정성을 보장합니다.

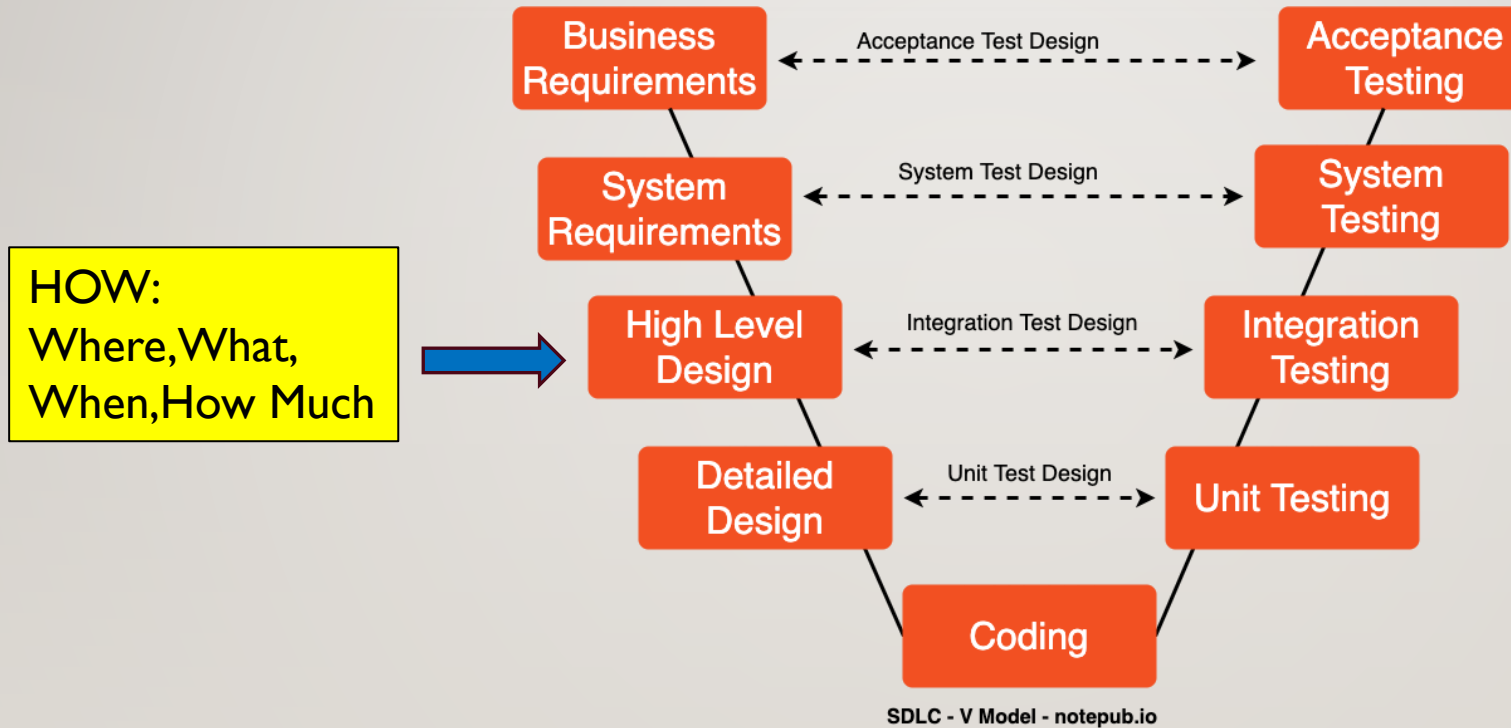
# GUIDE TO PROGRESS INDICATORS

---

- Project Timeline
  - **Green** – less 10% of the listed items are delayed
  - **Yellow** – more than 10% but less than 20% of listed items are delayed
  - **Red** – more than 20% of listed items are delayed
- Critical Path Items
  - **Green** – reduced number of item(s)
  - **Yellow** – no new item(s)
  - **Red** – additional item(s)



# SW DEVELOPMENT PROCESS



# OVERALL SYSTEM/HIGH LEVEL DESIGN(EXAMPLE)

---

- Detailed System Design with Specific Technologies and Models:

1. Deep Learning Model:

1. **Model Type:** Use a Convolutional Neural Network (CNN) based on the EfficientNet architecture for efficient and scalable image processing.

효율적이고 확장 가능한 이미지 처리를 위해 EfficientNet 아키텍처를 기반으로 하는 CNN(Convolutional Neural Network)을 사용합니다.

2. **Training:** Leverage transfer learning with pre-trained ImageNet weights as a starting point to reduce training time and improve accuracy.

사전 훈련된 ImageNet 가중치를 시작점으로 하는 전이 학습을 활용하여 훈련 시간을 단축하고 정확도를 높일 수 있습니다.

# OVERALL SYSTEM DESIGN(EXAMPLE)

---

- Detailed System Design with Specific Technologies and Models:

## 2. Hardware Specifications:

1. **Robotics Arm:** Integrate a high-precision KUKA robotic arm for stable and accurate product handling.

안정적이고 정확한 제품 핸들링을 위해 고정밀 KUKA 로봇 암을 통합합니다.

2. **Cameras:** Use Sony Industrial Cameras with high frame rates and resolution to capture detailed images for defect detection.

높은 프레임 속도와 해상도의 소니 산업용 카메라를 사용하여 결함 감지를 위한 디테일한 이미지를 캡처할 수 있습니다.



# OVERALL SYSTEM DESIGN(EXAMPLE)

---

- Detailed System Design with Specific Technologies and Models:

## 3. Software Technologies:

1. **Framework:** Develop the deep learning model using TensorFlow and Keras for their extensive support and community.

광범위한 지원과 커뮤니티를 위해 TensorFlow 및 Keras를 사용하여 딥 러닝 모델을 개발하세요.

2. **Server Technology:** Utilize NVIDIA DGX systems for high-throughput and low-latency processing, crucial for real-time applications.

실시간 애플리케이션에 중요한 높은 처리량과 짧은 대기 시간 처리를 위해 NVIDIA DGX 시스템을 활용하세요.



# OVERALL SYSTEM DESIGN(EXAMPLE)

---

- Detailed System Design with Specific Technologies and Models:

## 4. Interface and Control System:

1. **Dashboard:** Build the user interface using React.js for its efficient rendering performance, supported by a Node.js backend for handling API requests.

효율적인 렌더링 성능을 위해 React.js를 사용하여 사용자 인터페이스를 구축하고, API 요청을 처리하기 위한 Node.js 백엔드에서 지원합니다.

2. **Communication:** Implement MQTT for lightweight, real-time messaging between the robotic components and the server.

로봇 구성 요소와 서버 간의 경량 실시간 메시징을 위해 MQTT를 구현합니다.

# OVERALL SYSTEM DESIGN(EXAMPLE)

---

- Detailed System Design with Specific Technologies and Models:

## 5. Integration and Testing Techniques:

1. **Simulation Software:** Use ROS (Robot Operating System) for simulation and to prototype interactions between components.

ROS(Robot Operating System)를 사용하여 구성요소 간의 상호 작용을 시뮬레이션하고 프로토타이핑할 수 있습니다.

2. **Automated Testing:** Integrate Jenkins for continuous integration, ensuring every code commit is built, tested, and errors are addressed promptly.

지속적인 통합을 위해 Jenkins를 통합하여 모든 코드 커밋을 빌드, 테스트하고 오류를 신속하게 해결할 수 있습니다.

# BASE HW/OS

---

- PC

- Ubuntu 22.04
- USB Camera



- Network
  - Wifi



- AMR

- TurtleBot4
- Ubuntu 22.04





# OBJ. DET.

---

TARGET



DUMMY





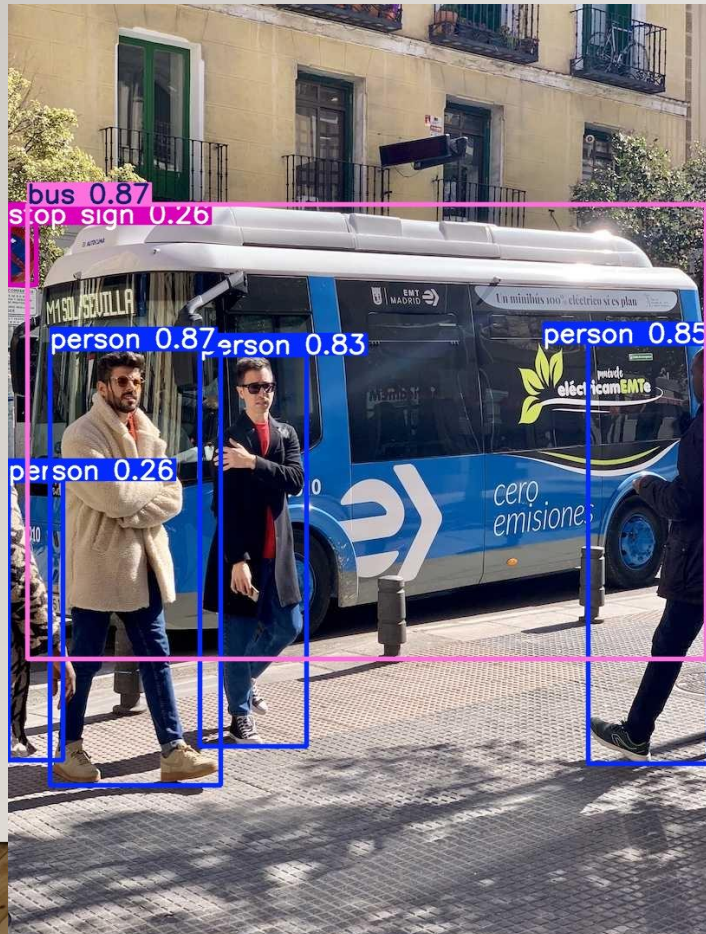
# AMR (DEMO)

---

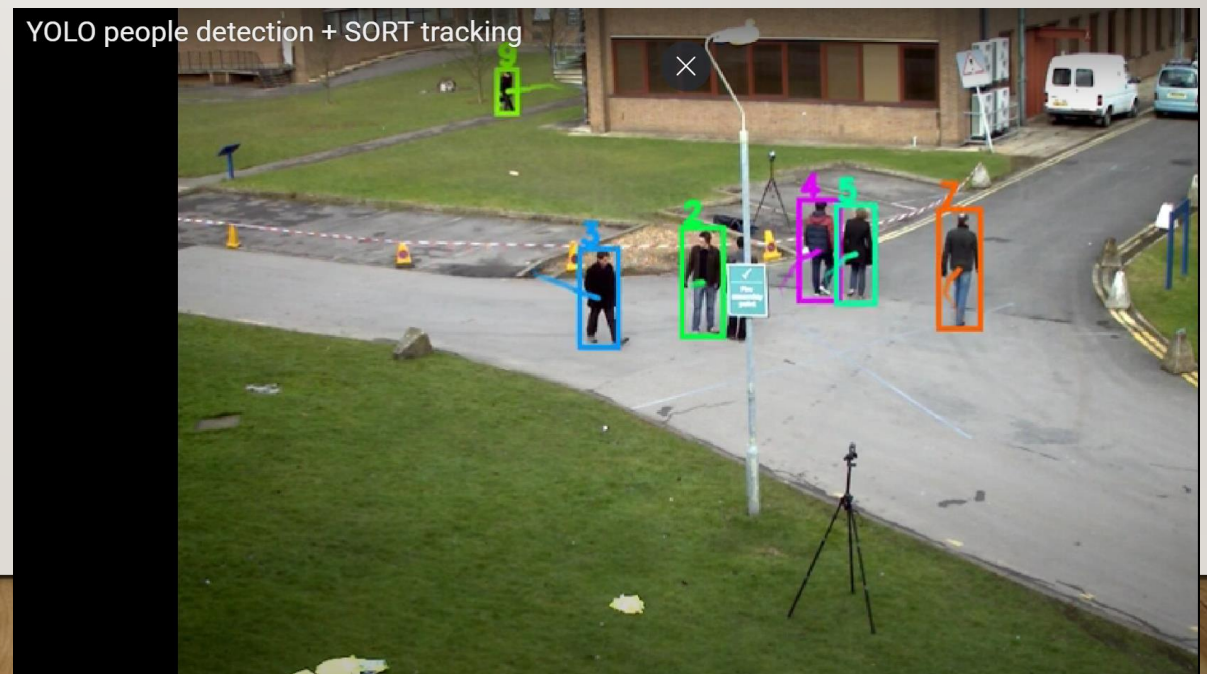
- [User Manual · Turtlebot4 User Manual](#)
- <https://turtlebot.github.io/turtlebot4-user-manual/>
- Navigation with SLAM
- Teleop with keyboard



# YOLO OBJ. DET. VS. YOLO TRACKING



- [Track - Ultralytics YOLO Docs](#)
  - [\(469\) YOLO people detection + SORT tracking – YouTube](#)
  - [Bing Videos](#)



# EXERCISE: FUNCTIONAL GROUPING

---

Using the functional process flow diagram created before, group the functions logically and by HW and SW that will be used to implement them



# KEY SUBSYSTEM (MODULES) TO DEVELOP

---

- Detection Alert
  - Camera Capture
  - Object Detection
  - Send messages to other subsystems
- AMR Controller
  - Receive messages and act accordingly
  - Move using (SLAM) with Obstruction avoidance
  - Target Acquisition (Obj. Det.) and Tracking
  - Follow target using camera and motor control

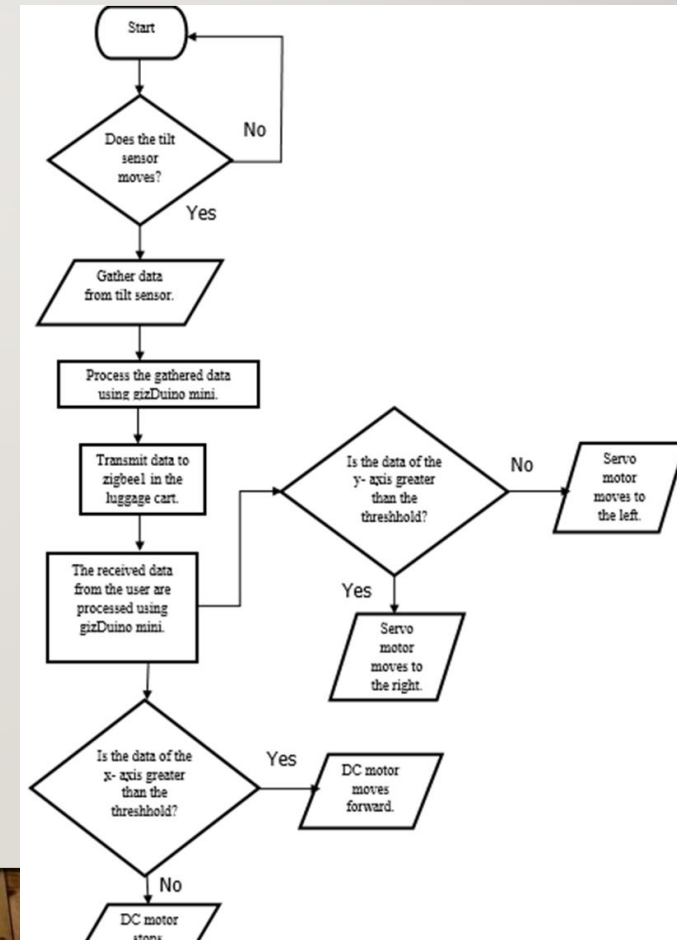


# VISUALIZATION – SYSTEM FUNCTIONAL PROCESS FLOW DIAGRAMS

- To-Be Functional Process Flow Diagram

Detection Alert  
AMR Controller

- **F**unctions
  - **I**nterfaces
  - **T**esting
- Dataflow
- Error and Exception Handling



# SYSTEM DESIGN PRESENTATION BY EACH TEAM

---



# EXAMPLE SYSTEM DESIGN DOCUMENT

## System Design Document (SDD)❏

Project Title: Autonomous Mobile Robot (AMR) Security System↓

Version: 1.1↓

Date: [Insert Date]❏

### 1. Overview❏

The Autonomous Mobile Robot (AMR) Security System is designed to provide autonomous patrolling, threat detection, and alerting within a secure area using a single AI-enabled robot. The system consists of one AMR equipped with necessary hardware and software components to operate independently, processing data on-board without the need for a central server.❏

### 2. System Architecture❏

Since the system consists of a single AMR, data processing, navigation, threat detection, and alerting are all performed locally on the AMR itself. The AMR communicates directly with a user interface on a PC via a local network (Wi-Fi) for monitoring, alerts, and manual override if required.❏

## 시스템 설계 문서 (SDD)❏

프로젝트 제목: 자율 이동 로봇(AMR) 보안 시스템↓

버전: 1.1↓

날짜: [날짜 삽입]❏

### 1. 개요❏

자율 이동 로봇(AMR) 보안 시스템은 단일 AI 기반 로봇을 사용하여 보안 구역 내에서 자율 순찰, 위협 탐지 및 경고를 제공하도록 설계되었습니다. 시스템은 단일 AMR이 독립적으로 작동할 수 있도록 필요한 하드웨어 및 소프트웨어 구성 요소로 구성되며, 중앙 서버 없이 데이터를 현장에서 처리합니다.❏

### 2. 시스템 아키텍처❏

이 시스템은 단일 AMR으로 구성되므로 데이터 처리, 네비게이션, 위협 탐지 및 경고가 모두 AMR에서 로컬로 수행됩니다. AMR은 모니터링, 알림 및 수동 제어를 위해 PC의 사용자 인터페이스와 로컬 네트워크(Wi-Fi)를 통해 직접 통신합니다.❏

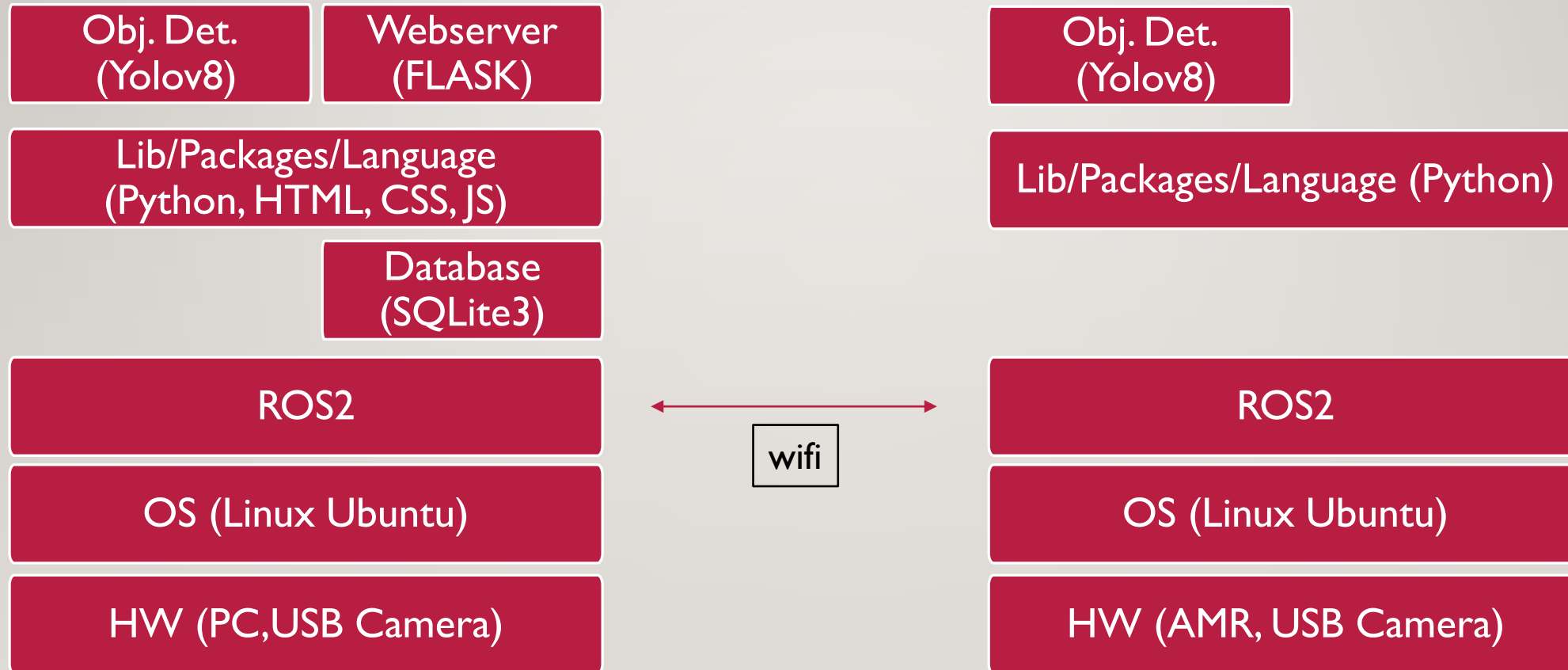
# SYSTEM AND DEVELOPMENT ENVIRONMENT SETUP

---





# PROJECT SW STACK



# USEFUL COMMANDS

---

\$ lsb\_release -a

- Linux distribution info

\$ echo \$ROS\_DISTRO

- ROS: Humble

\$ code --version

- Vscode

\$ python3 --version

- Python

\$ sudo apt update

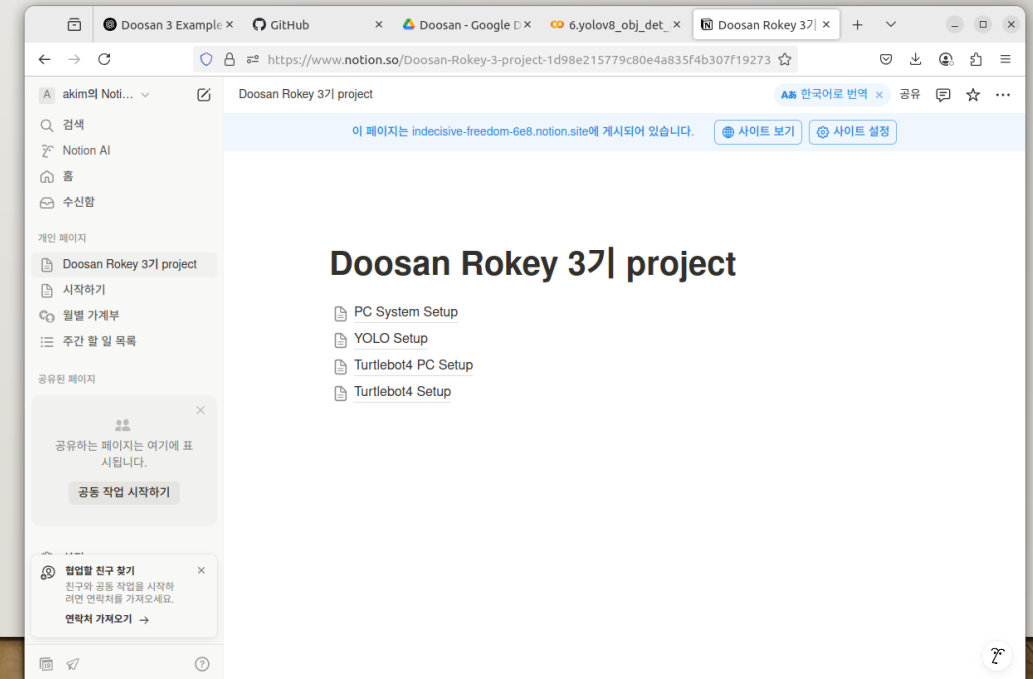
\$ sudo apt upgrade

\$ python -m ensurepip --upgrade

- Assumes Linux (Ubuntu 22.04), ROS Humble, VScode, and Python are already installed globally

# SYSTEM ENVIRONMENT SETUP SHELL SCRIPT

- PC System Setup
- <https://indecisive-freedom-6e8.notion.site/PC-System-Setup-1d98e215779c806080bbd1014d63a406>



# ROS2 DEVELOPMENT WORKSPACE

---

```
$ cat ~/.bashrc
```

```
$ echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc #check path
```

```
$ sudo apt install python3-colcon-common-extensions
```

```
$ echo "source /usr/share/colcon_argcomplete/hook/colcon_argcomplete.bash" >> ~/.bashrc  
#check path
```

```
$ source ~/.bashrc
```





# PREPARING FOR YOLO LABELLING

---

Data Labelling : use previously installed  
Labellmg









Or

[YOLO Setup](#)

<https://indecisive-freedom-6e8.notion.site/YOLO-Setup-1d98e215779c80f389eefbe0d86ee0ec>

\$ labellmg

## Doosan Rokey 371 project

-  [PC System Setup](#)
-  [YOLO Setup](#) 
-  [Turtlebot4 Setup](#)
-  [Turtlebot4 PC Setup](#)
-  [Single Robot Setup](#)
-  [Multi Robot Setup](#)
-  [Transform Explained](#)

# SETUP BASH

---

Add in ~/.bashrc:

- source turtlebot4\_ws/install/setup.bash
- ROS\_DOMAIN\_ID = 0

\$ source ~/.bashrc

# USING ROS\_WS

---



# ROS2 DEVELOPMENT WORKSPACE

---

## CREATE WORKSPACE

```
$ mkdir -p  
  ~/rokey3_<grp_letter><grp_num>_ws/src  
  • (i.e. mkdir -p ~/rokey3_A2_ws/src)
```

```
$ cd ~/rokey3_A2_ws
```

```
$ rosdep install --from-paths src --ignore-src  
-r -y
```

If not installed...

```
$ sudo rosdep init
```

```
$ rosdep update
```

## \*NOT CREATED UNTIL COLCON

```
workspace/      # Root of the workspace  
├─ src/         # Source code (ROS packages)  
├─ build/       # Build files (generated by colcon)  
├─ install/     # Installed packages and setup scripts  
└─ log/         # Build logs
```

```
$ colcon build
```

```
$ source install/setup.bash
```



# ROS2 DEVELOPMENT WORKSPACE

---

```
$ cd ~/rokey3_A2_ws/src
```

```
$ ros2 pkg create --build-type  
ament_python <my_package>
```

```
my_package/  
├─ package.xml          # Package metadata and dependencies  
├─ setup.py             # Build instructions for Python packages  
├─ setup.cfg            # Optional, configures metadata for setuptools  
├─ launch/              # Launch files for starting nodes (optional)  
├─ config/              # Configuration files (optional)  
├─ resource/            # Empty file matching package name for ament index  
├─ my_package/          # Python package directory (contains code)  
│   └─ __init__.py      # Makes this directory a Python package  
│   └─ my_node.py       # Example Python node  
└─ msg/                 # Message definitions (optional)
```

# ROS2 DEVELOPMENT WORKSPACE

---

Write your code below the `my_package/` directory under `my_package/ package` directory

```
my_package/
├─ package.xml          # Package metadata and dependencies
├─ setup.py             # Build instructions for Python packages
├─ setup.cfg            # Optional, configures metadata for setuptools
├─ launch/              # Launch files for starting nodes (optional)
├─ config/              # Configuration files (optional)
├─ resource/            # Empty file matching package name forament inc
├─ my_package/          # Python package directory (contains code)
│   └─ __init__.py      # Makes this directory a Python package
│   └─ my_node.py       # Example Python node
└─ msg/                 # Message definitions (optional)
```

# ROS2 DEVELOPMENT WORKSPACE

---

```
$ cd ~/ rokey3_A2_ws
```

```
$ colcon build
```

```
workspace/      # Root of the workspace
├─ src/         # Source code (ROS packages)
├─ build/       # Build files (generated by colcon)
├─ install/     # Installed packages and setup scripts
└─ log/        # Build logs
```

```
$ echo "source ~/ rokey3_A2_ws/install/setup.bash" >> ~/.bashrc #check path
```

```
$ source ~/.bashrc
```

# PLEASE REVIEW YOUR WORK FROM EARLIER ONLINE CLASS

---

- Yolo
  - Data Labelling (ex: Labellmg)
  - Data pre-processing for YoloV8 Training
  - YoloV8 training to create .pt file
  - Using .pt file to predict/inference
- ROS
  - colcon build
  - Node, Topic, Service, Action, Interface, etc. coding



# 프로젝트 RULE NUMBER ONE!!!

# Are we having Fun???

