

# GOOD MORNING!

早上好!

안녕하세요!

---

DAY 2



# DAY 1 RECAP

---



## 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

---

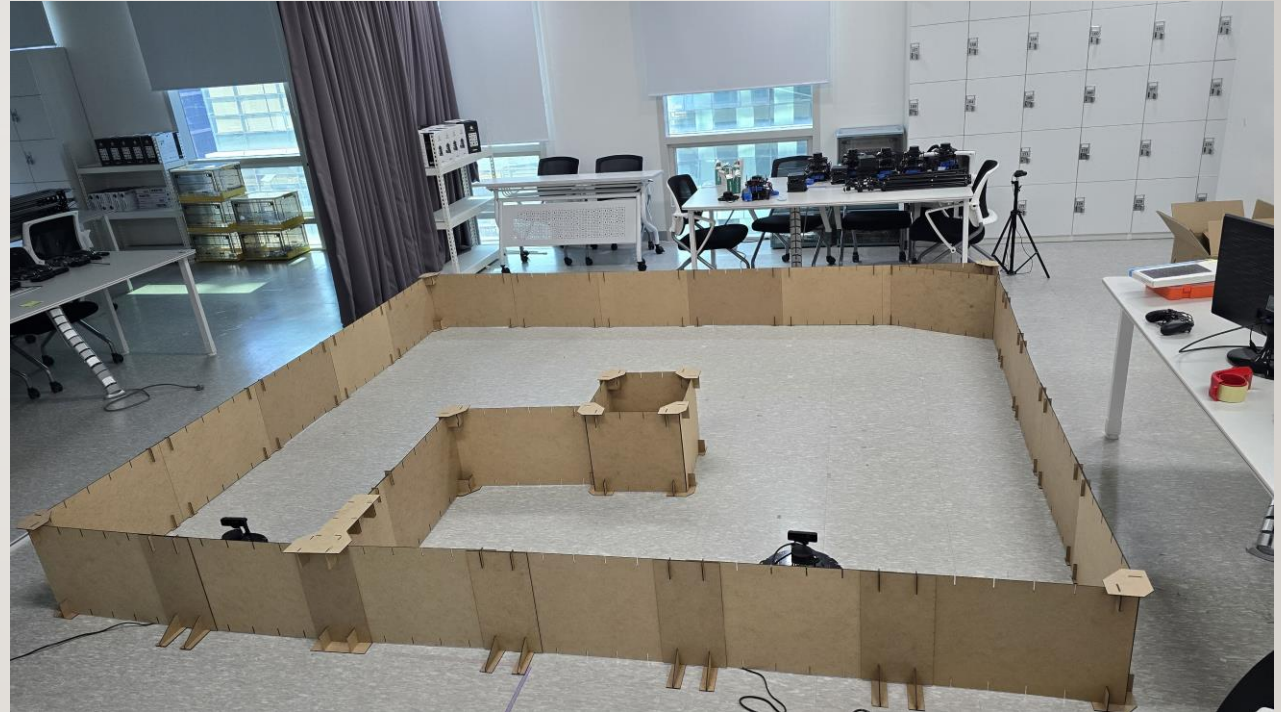
- Final Project (2 Teams in One)

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

---



# 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

DETECTION ALERT

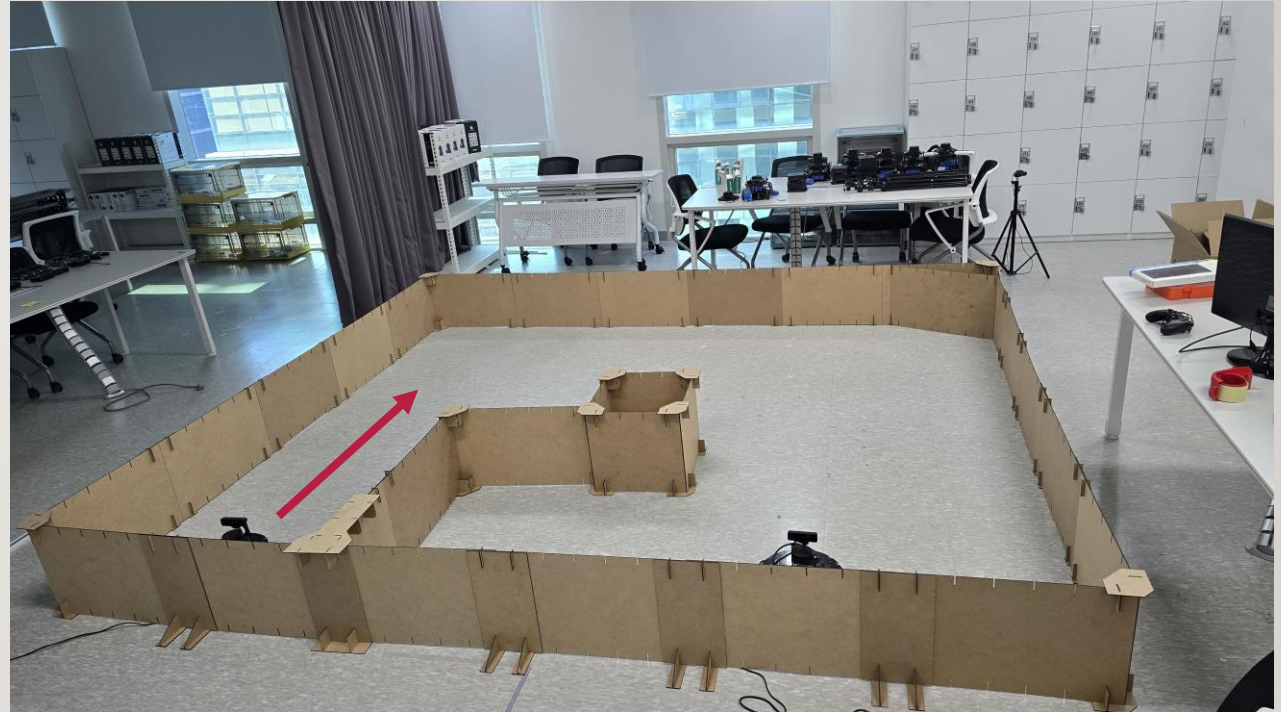
---





START

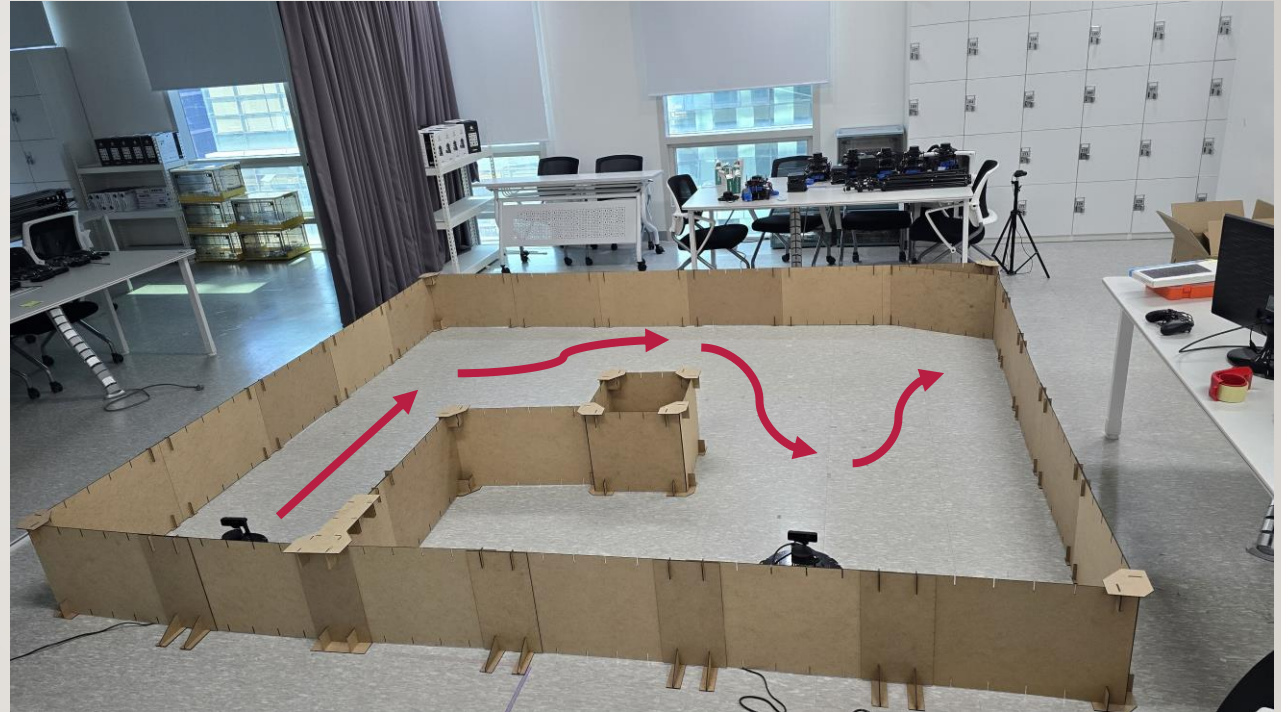
---





# NAVIGATE

---



# TRACK & FOLLOW

---



# DAY I

---

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



# 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

# 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
- Ask the right questions? (to **YOU, team** and me)
- Investigate/Research/Analyze

# 프로젝트 RULE

---

80/20 → 20/80



# TEAMWORK AND PROJECT MANAGEMENT

---



# 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





프로젝트 RULE NUMBER ONE!!!

---

Have Fun Fun Fun!



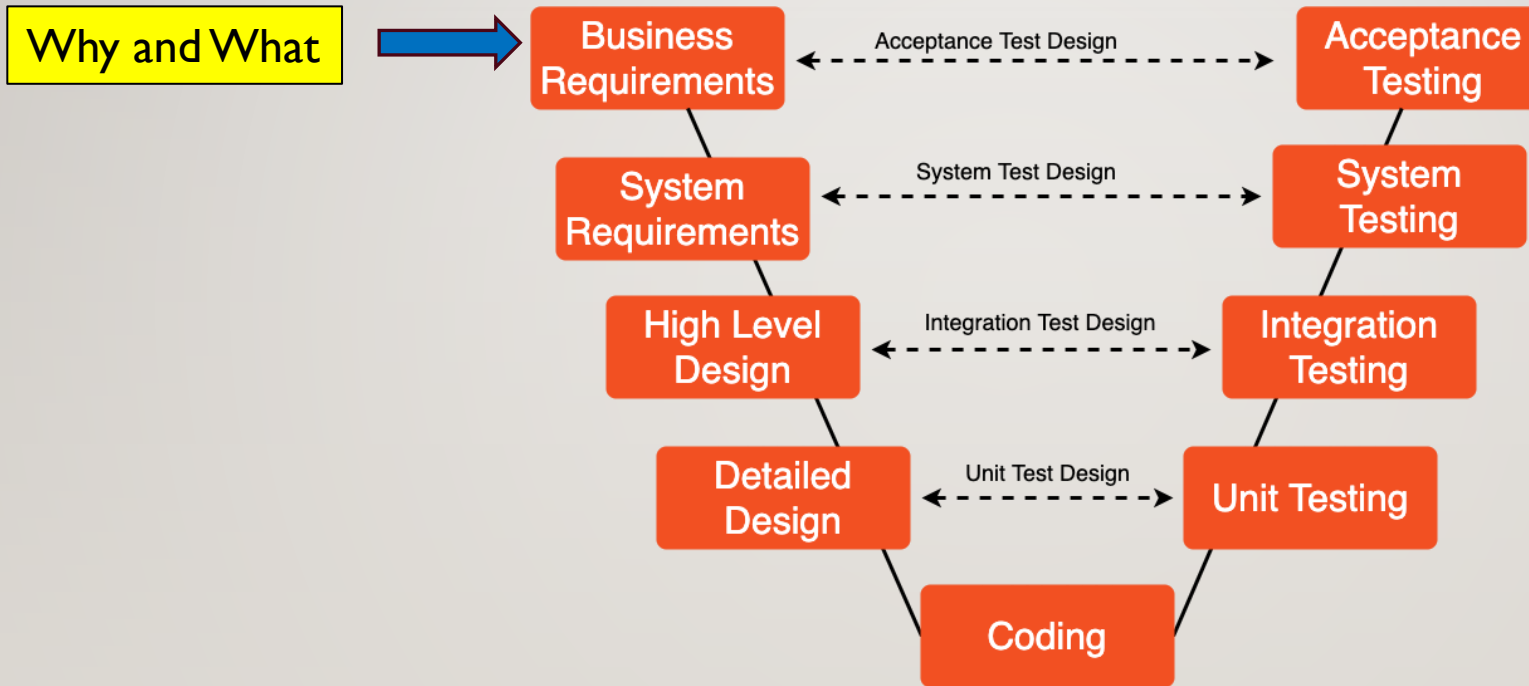


PROJECT DEVELOPMENT  
IS A PROCESS

---



# SW DEVELOPMENT PROCESS



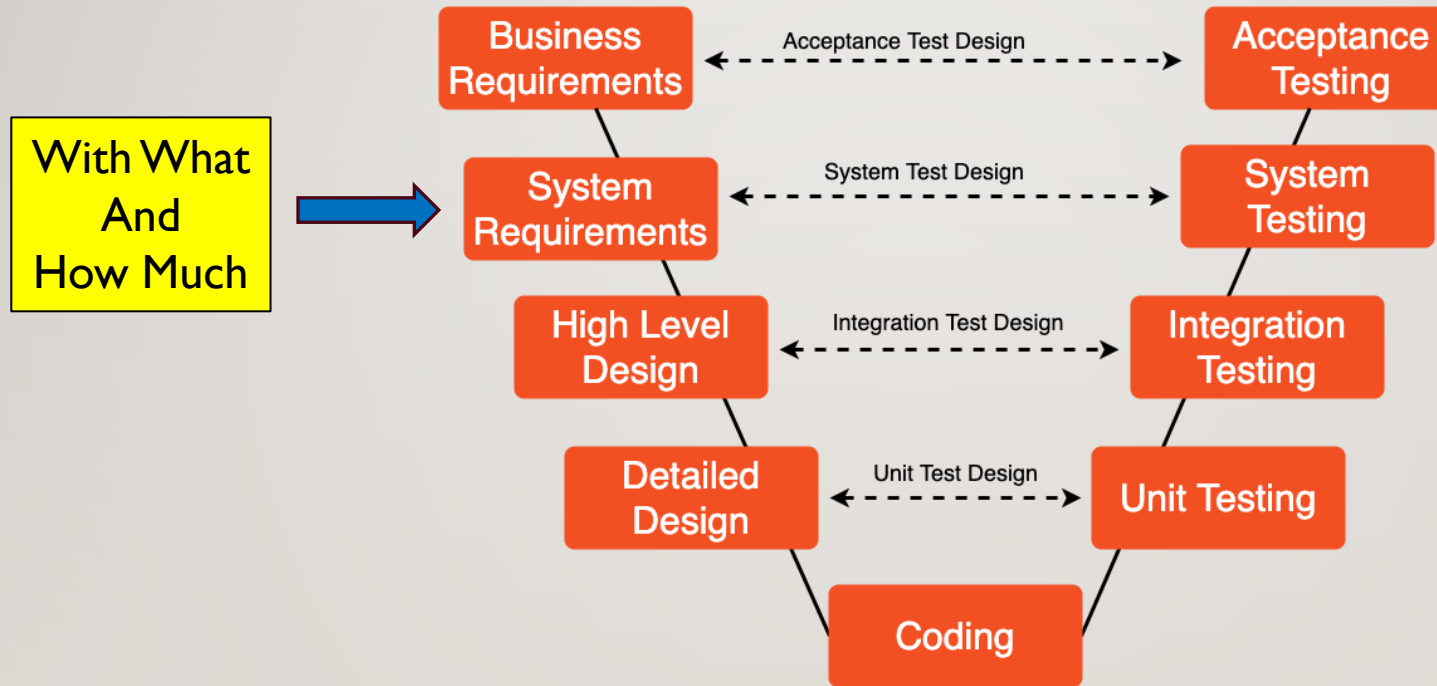
SDLC - V Model - notepub.io

# TEAM EXERCISE I

---

Brainstorm Business Requirement for the project and write business requirement statement

# SW DEVELOPMENT PROCESS



SDLC - V Model - notepub.io



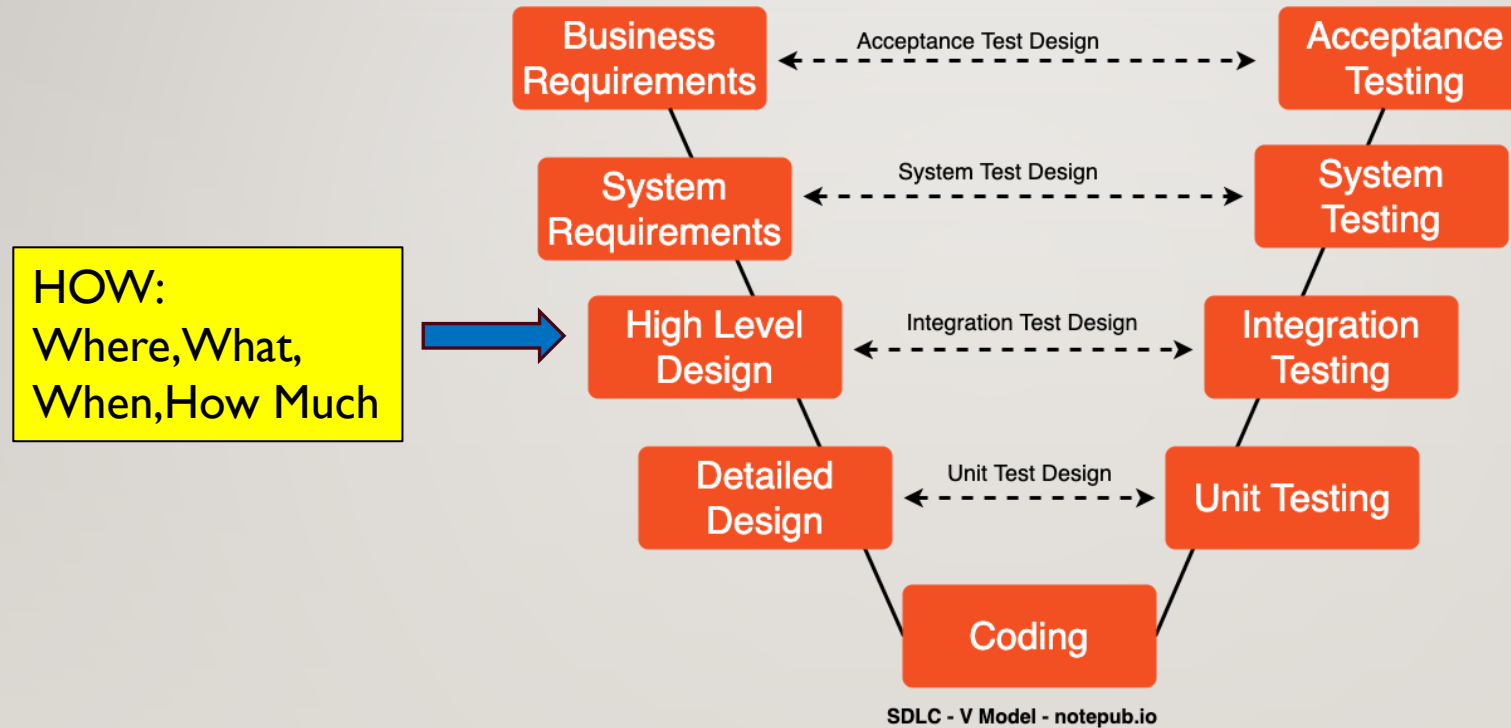
# TEAM EXERCISE 2

---

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

Using the posted notes and flipchart as needed

# SW DEVELOPMENT PROCESS



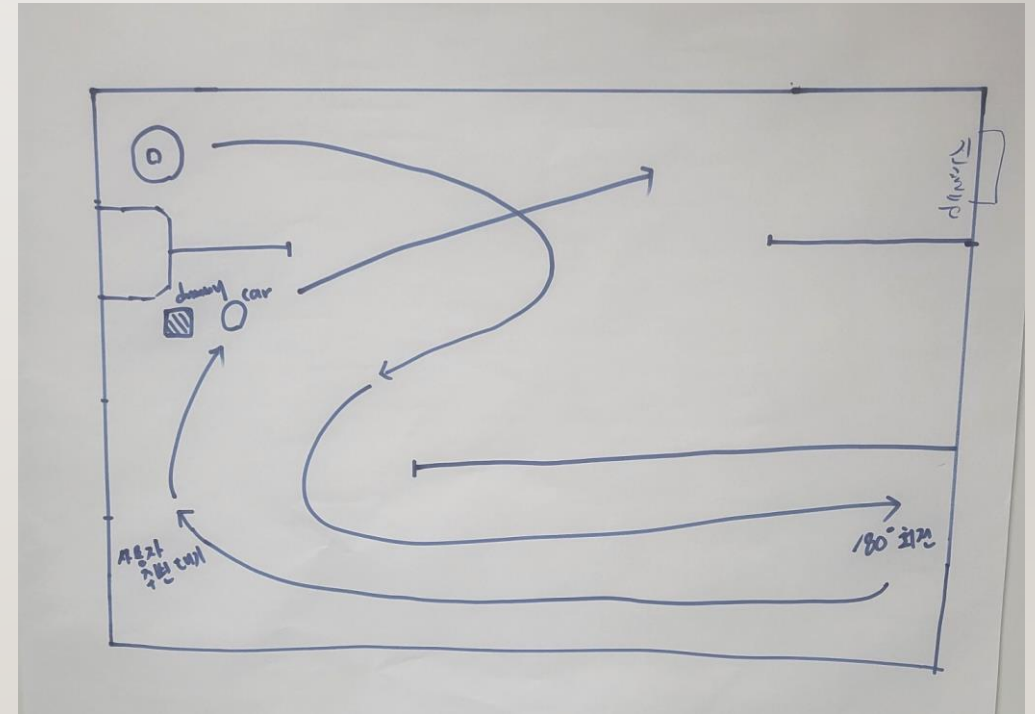
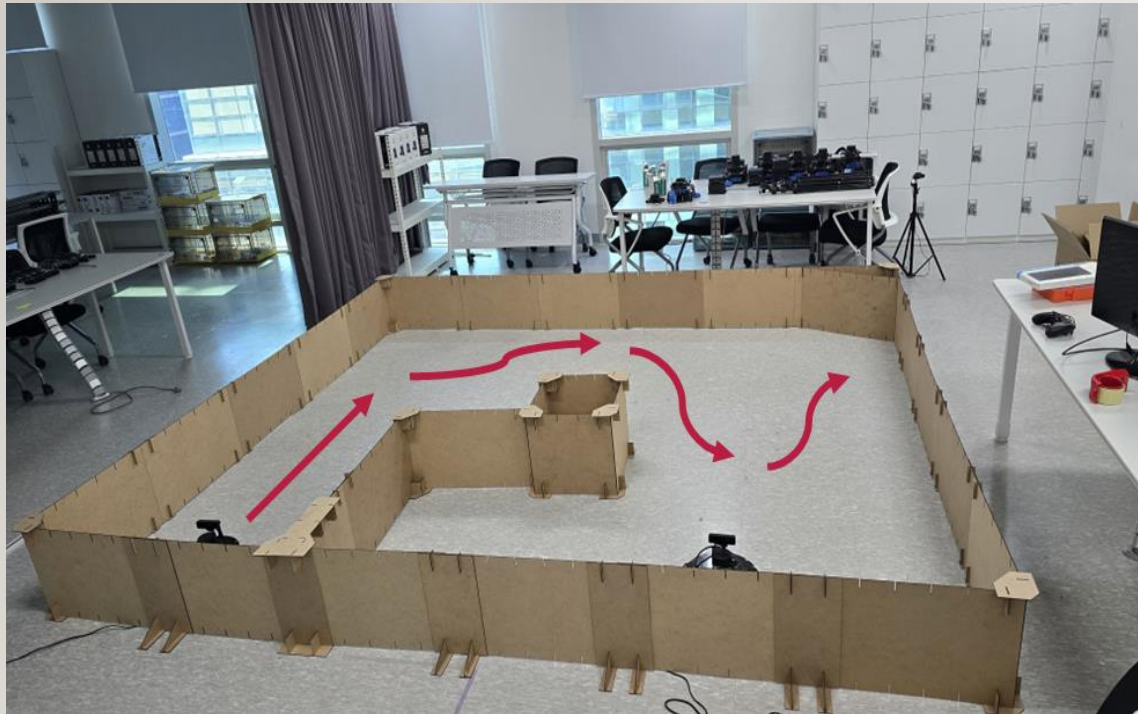
# DEVELOP YOUR BUSINESS SCENARIO (USE-CASE) PROCESS DIAGRAM

---

Using the posted notes and flipchart as needed

# EXERCISE: SKETCH YOUR SCENARIO

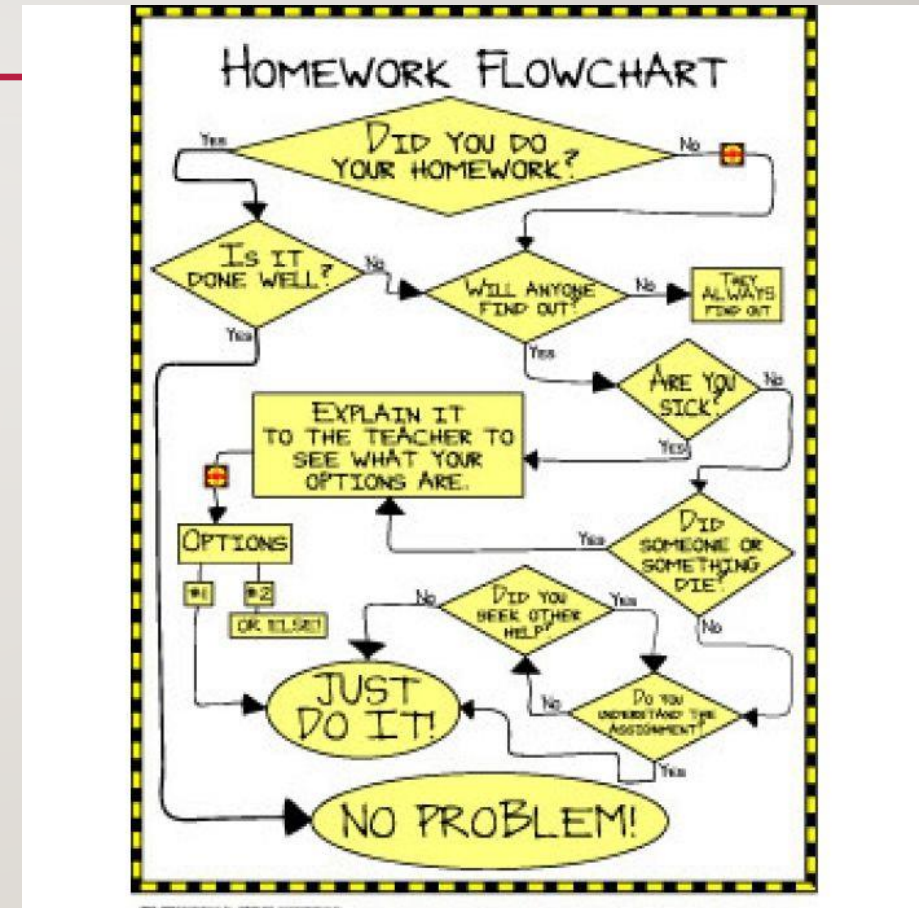
---





# 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/>



# HOMEWORK CHECK

---

# PLEASE REVIEW YOUR WORK FROM EARLIER ONLINE CLASS

---

- Yolo obj. Det. Vs. Yolo Tracking
  - [Object Detection - Ultralytics YOLO Docs](#)
  - [Track - Ultralytics YOLO Docs](#)
  - [Model Training with Ultralytics YOLO - Ultralytics YOLO Docs](#)
- 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

# AMR (TURTLEBOT4)

---

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





# ROS EXERCISE

---

Create a ROS2 Package with these publisher and subscribers

```
2_0_a_image_publisher.py
2_0_b_image_subscriber.py
2_0_c_data_publisher.py
2_0_d_data_subscriber.py
```

```
$ ros2 run rqt_graph rqt_graph
$ ros2 node list
$ ros2 node info <node_name>
$ ros2 topic list
$ ros2 topic info <topic_name>
$ ros2 topic echo /chatter
$ ros2 interface list
$ ros2 interface show
  <package_name>/msg/<MessageName>
```

[Tutorials — ROS 2 Documentation: Humble documentation](#)

# SYSTEM REQUIREMENT PRESENTATION BY EACH TEAM

---

Using the posted notes and flipchart as needed

# DAY 2

---



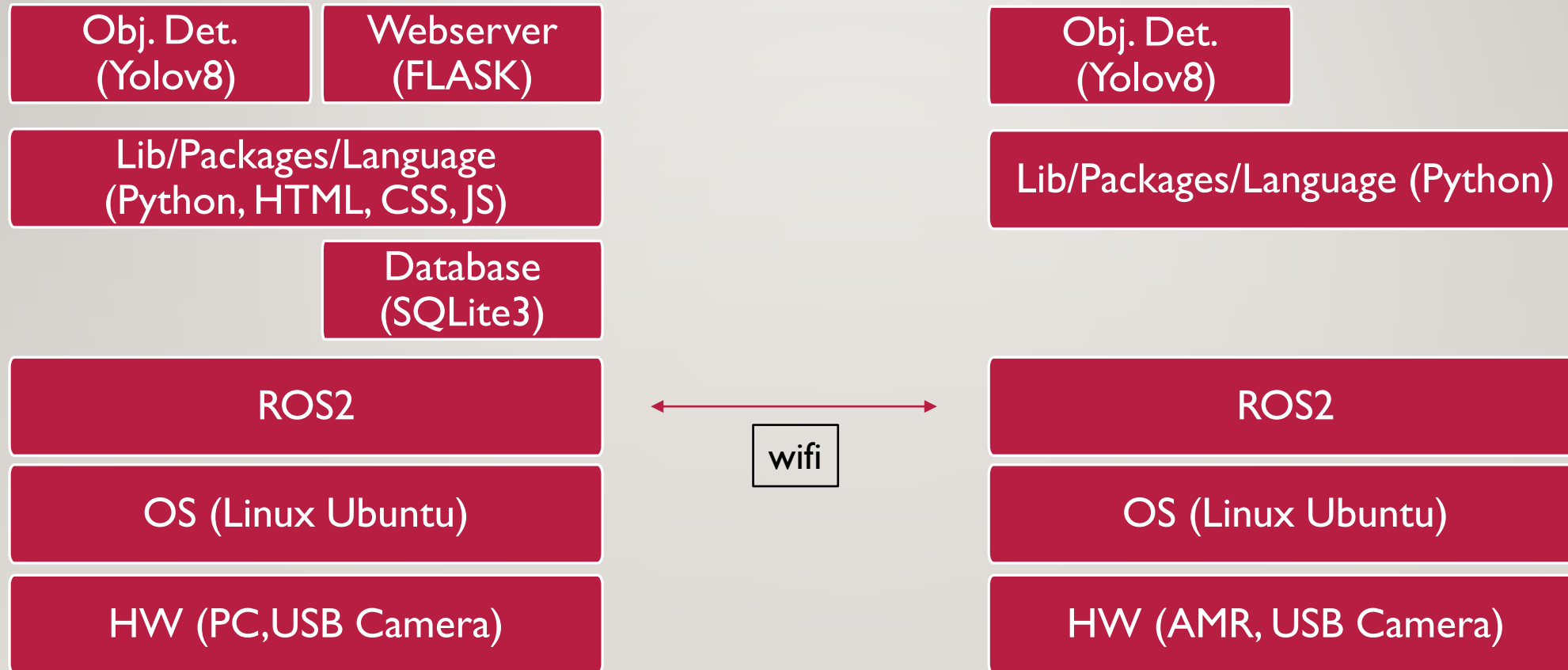


# SYSTEM AND DEVELOPMENT ENVIRONMENT SETUP

---



# 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



# USING ROS\_WS

---



# ROS2 DEVELOPMENT WORKSPACE

---

## CREATE WORKSPACE

```
$ mkdir -p  
  ~/rokey4_<grp_letter><grp_num>_ws/src  
  • (i.e. mkdir -p ~/rokey3_A2_ws/src)  
  
$ cd ~/rokey4_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 ~/rokey4_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 ~/ rokey4_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 ~/ rokey4_A2_ws/install/setup.bash" >>  #check path
~/.bashrc
```

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

# ROS EXERCISE

---

- 🔗 2\_0\_a\_image\_publisher.py
- 🔗 2\_0\_b\_image\_subscriber.py
- 🔗 2\_0\_c\_data\_publisher.py
- 🔗 2\_0\_d\_data\_subscriber.py

# ROS HINTS

---

- Edit setup.py under <package\_name> directory add entry for each node

```
entry_points={ 'console_scripts':  
[ '<command_name> =  
<package_name>.<code_filename>:main', },
```

<command\_name> is used when ros2 run is executed i.e. data\_publisher

```
tests_require=['pytest'],  
entry_points={  
    'console_scripts': [  
        'pub_image = day2.2_0_a_image_publisher:main',  
        'show_image = day2.2_0_b_image_subscriber:main',  
        'pub_data = day2.2_0_c_data_publisher:main',  
        'show_data = day2.2_0_d_data_subscriber:main',
```

# ROS HINTS

---

\$ cd ~/rokey4\_A2\_ws

\$ colcon build

\$ source

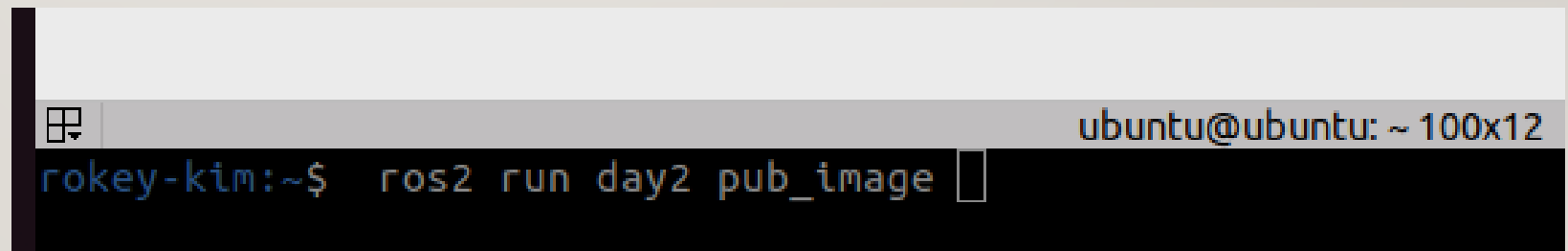
~/rokey4\_A2\_ws/install/setup.bash

\$ ros2 run <package\_name>

<command\_name>

\$ sudo apt update

\$ sudo apt install terminator

A screenshot of a terminal window. The title bar at the top reads 'ubuntu@ubuntu: ~ 100x12'. The terminal content shows a prompt 'rokey-kim:~\$' followed by the command 'ros2 run day2 pub\_image' and a cursor. The terminal window is overlaid on a background of a wooden floor.

```
rokey-kim:~$ ros2 run day2 pub_image
```



# ROS HINTS

---

```
2_0_a_image_publisher.py
2_0_b_image_subscriber.py
2_0_c_data_publisher.py
2_0_d_data_subscriber.py
```

```
$ ros2 run rqt_graph rqt_graph
```

```
$ ros2 node list
```

```
$ ros2 node info <node_name>
```

```
$ ros2 topic list
```

```
$ ros2 topic info <topic_name>
```

```
$ ros2 topic echo /chatter
```

```
$ ros2 interface list
```

```
$ ros2 interface show
<package_name>/msg/<MessageName>
```

# SYSTEM ENVIRONMENT SETUP SHELL SCRIPT


---

## Doosan Rokey 4 7| Project

 [PC System Setup \(1\)](#)



 [YOLO Setup \(1\)](#)

 [Turtlebot4 Setup \(1\)](#)

 [Turtlebot4 PC Setup \(1\)](#)

 [Single Robot Setup \(1\)](#)

# 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


## Doosan Rokey 4 7| Project

 [PC System Setup \(1\)](#)

 [YOLO Setup \(1\)](#)



 [Turtlebot4 Setup \(1\)](#)

 [Turtlebot4 PC Setup \(1\)](#)

 [Single Robot Setup \(1\)](#)






 labellmg



# HOW TO SETUP AMR

---

## Doosan Rokey 4 7| Project

-  [PC System Setup \(1\)](#)
-  [YOLO Setup \(1\)](#)
-  [Turtlebot4 Setup \(1\)](#)
-  [Turtlebot4 PC Setup \(1\)](#)
-  [Single Robot Setup \(1\)](#)



# AMR (TURTLEBOT4)

---

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





# AMR (DEMO)

---

- Power On and Off
- Connecting to Robot
  - Wifi Router
  - SSH
- Docking and Undocking (using HMI panel/buttons)
- Teleop with keyboard





# CONNECTING TO AMR -- SSH

---

Connect your PC to WiFi router that your AMR is connected

Ex: turtle09

Obtain the ip address shown on the OLED display of the Turtlebot4

EX: 192.168.10.16

Open a terminal window

```
$ dpkg -l | grep openssh
```

If not installed...

```
$ sudo apt install openssh-server -y
```

Connect to AMR via SSH

```
$ ssh ubuntu@192.168.10.16
```

PW: rokey1234

Do **NOT INSTALL** any packages to AMR **WITHOUT** speaking with me first!!!

# UNDOCK/DOCK AMR (ON AMR)

---

## UNDOCK

\$ ros2 topic list

Check the list

\$ ros2 action send\_goal  
/robot<n>/undock  
irobot\_create\_msgs/action/Undock  
“{”

## DOCK

\$ ros2 topic list






Check the list

\$ ros2 action send\_goal /robot<n>/dock  
irobot\_create\_msgs/action/Dock “{”

# SETUP PC FOR AMR

---

## Doosan Rokey 4 7| Project

-  [PC System Setup \(1\)](#)
-  [YOLO Setup \(1\)](#)
-  [Turtlebot4 Setup \(1\)](#)
-  [Turtlebot4 PC Setup \(1\)](#)
-  [Single Robot Setup \(1\)](#)



# SETUP BASH

---

Add in ~/.bashrc:

- `source ~/turtlebot4_ws/install/setup.bash`
- `ROS_DOMAIN_ID = 0`






`$ source ~/.bashrc`



# SETUP PC FOR AMR

---

## Doosan Rokey 4 7| Project

-  [PC System Setup \(1\)](#)
-  [YOLO Setup \(1\)](#)
-  [Turtlebot4 Setup \(1\)](#)
-  [Turtlebot4 PC Setup \(1\)](#)
-  [Single Robot Setup \(1\)](#)



# HOW TO TEST PC – AMR CONNECTION

---

## PC TERMINAL

\$ ros2 topic list (will need to execute it twice)

#Check the list

\$ ros2 run teleop\_twist\_keyboard  
teleop\_twist\_keyboard --ros-args -r  
/cmd\_vel:=/robot<n>/cmd\_vel

\$ ro2 topic echo /robot<n>/oakd/rgb/image\_raw  
--once

## AMR TERMINAL

\$ ros2 topic list (will need to execute it twice)

#Check the list

- Undock your robot

\$ ro2 topic echo /robot<n>/oakd/rgb/image\_raw  
--once

# 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

# PERFORM DATA COLLECTION FOR DETECTION ALERT

---






# COLLECTION IMAGES FROM WEBCAM

---

- Image Capture (WEBCAM)



```
2_0_a_image_publisher.py
2_0_b_image_subscriber.py
2_0_c_data_publisher.py
2_0_d_data_subscriber.py
2_1_a_capture_wc_image.py
2_1_b_cont_capture_wc_image.py
2_1_c_capture_wc_thread.py
```

# COLLECTION IMAGES FROM AMR CAMERA

---

- Which image topic to use?



# UNDOCK/DOCK AMR

---

## UNDOCK

\$ ros2 topic list

Check the list

\$ ros2 action send\_goal  
/robot<n>/undock  
irobot\_create\_msgs/action/Undock  
“{”

\$ ros2 topic list

Check the list and compare

## DOCK

\$ ros2 action send\_goal /robot<n>/dock  
irobot\_create\_msgs/action/Dock “{”

# UPDATING THE OAKD CONFIG (ROBOT)

## ON TURTLEBOT4:

```
$ cd
/opt/ros/humble/share/turtlebot4_bringup/co
nfig

$ sudo cp oakd_pro.yaml oakd_pro_orig.yaml

$ sudo cp oakd_pro_new.yaml oakd_pro.yaml

$ sudo reboot
```

```
1 /oakd:
2   ros_parameters:
3     use_sim_time: false
4
5   camera:
6     i_enable_imu: false
7     i_enable_ir: false
8     i_floodlight_brightness: 0
9     i_laser_dot_brightness: 100
10    i_nn_type: none
11    i_pipeline_type: RGBD # RGB + Depth
12    i_usb_speed: SUPER_PLUS
13
14    rgb:
15      i_board_socket_id: 0
16      i_width: 640
17      i_height: 480
18      i_fps: 30.0
19      i_enable_preview: true
20      i_interleaved: false
21      i_low_bandwidth: true
22      i_publish_topic: true
23      i_resolution: 480P # sets 640x480 internally
24
25
26    stereo: # [x] Required to enable depth
27      i_board_socket_id: 1
28      i_fps: 30.0
```



# WHICH IMAGE TOPIC TO USE?

---

- /oakd/rgb/preview/image\_raw
  - /oakd/rgb/image\_raw
  - /oakd/rgb/image\_raw/compressed
  - /oakd/stereo/image\_raw
  - ...
- 
- **\*\*\* not all of the topics are visible, initially**
- EXERCISE
    - Create a script to display and compare

# DIMENSIONS AND RESOLUTION

## Supported `i_resolution` values (RGB):

Resolution Keyword	Width × Height	Notes
1080P	1920 × 1080	Default, high-res
720P	1280 × 720	Medium-res
800P	1280 × 800	Slightly taller
480P	640 × 480	✓ Ideal for alignment with stereo
400P	640 × 400	Wide, cropped top/bottom
320P	640 × 360	Lower-res
240P	320 × 240	Very low-res, fast

Use **rqt** to check and compare the different image topics

# CODING HINTS

---





- Image Capture

```
2_1_a_capture_wc_image.py  
2_1_b_cont_capture_wc_image.py  
2_1_c_capture_wc_thread.py
```

# CODING HINTS

---

- Image Capture
- Image Capture (AMR)

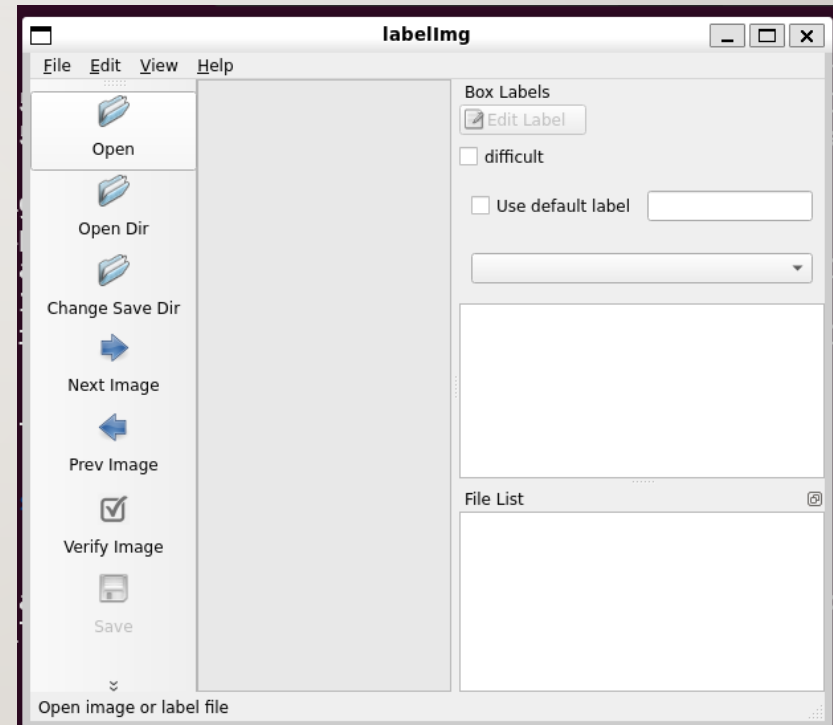
 2\_1\_a\_capture\_wc\_image.py 2\_1\_b\_cont\_capture\_wc\_image.py 2\_1\_c\_capture\_wc\_thread.py 2\_1\_d\_capture\_image.py 2\_1\_e\_cont\_capture\_image.py



# CODING HINTS

---

- Image Capture
- Data Labelling
  - Goto the /labellmg/data/ directory
  - Rename the predefined\_classes.txt



# CODING HINTS

---

- Data Labelling : Labellmg

## 라벨링 순서

1. 이미지파일 불러오기 (Open Dir)
2. 저장형식 변경 (PascalVOC, YOLO)
3. 이미지 선택
4. 바운딩 박스 그리기(create rectbox)
5. Class 지정
6. 저장경로 생성 및 변경(Change Save Dir)
7. 저장(Save)


## 단축키

Ctrl + u	Load all of the images from a directory
Ctrl + r	Change the default annotation target dir
Ctrl + s	Save
Ctrl + d	Copy the current label and rect box
Ctrl + Shift + d	Delete the current image
Space	Flag the current image as verified
w	Create a rect box
d	Next image
a	Previous image
del	Delete the selected rect box
Ctrl++	Zoom in
Ctrl--	Zoom out
↑→↓←	Keyboard arrows to move selected rect box

# CODING HINTS

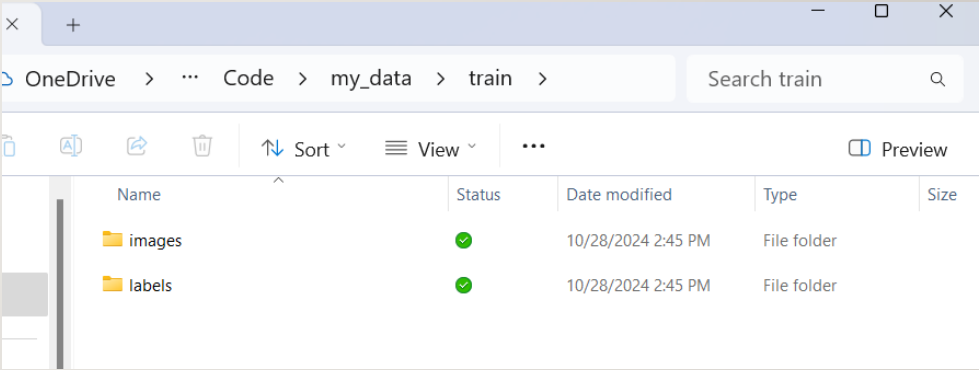
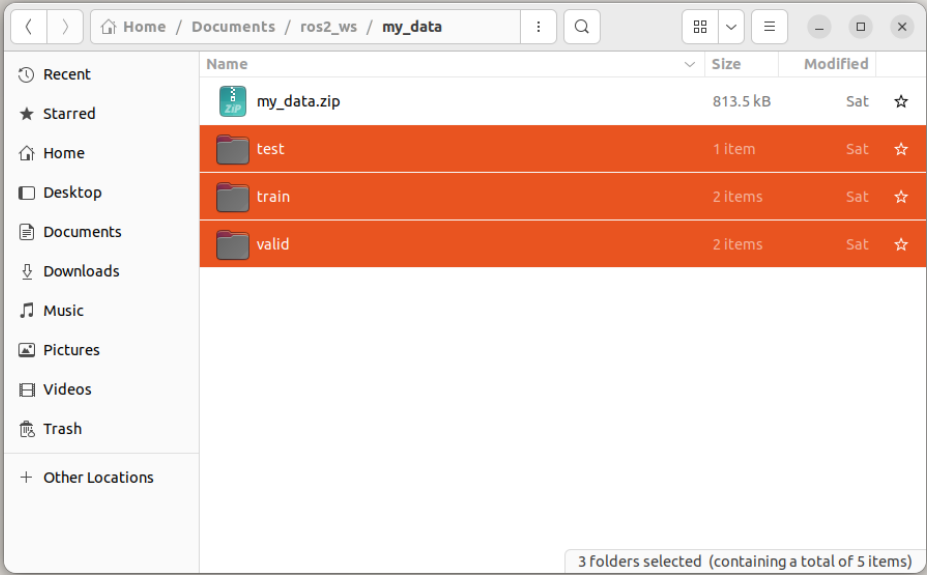
---

- Image Capture
- Data Labelling
- Data Preprocessing



```
2_1_a_capture_wc_image.py
2_1_b_cont_capture_wc_image.py
2_1_c_capture_wc_thread.py
2_1_d_capture_image.py
2_1_e_cont_capture_image.py
2_3_a_create_data_dirs.py
2_3_b_move_image.py
2_3_c_move_labels.py
```

# ZIP TRAIN DATA SET





# PERFORM YOLO TRAINING & INFERENCE

---



# CODING HINTS

---

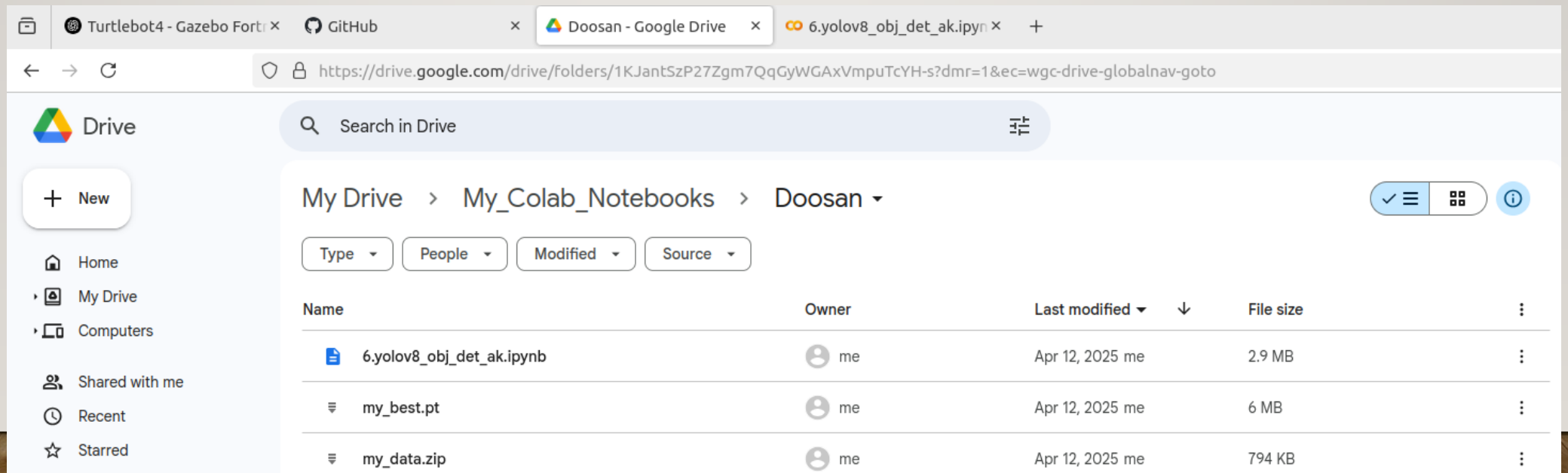
- Image Capture
- Data Labelling
- Preprocessing
- Yolo8 Object Det (Training)



```
2_3_a_create_data_dirs.py
2_3_b_move_image.py
2_3_c_move_labels.py
2_4_a_yolov8_obj_det_ak.ipynb
2_4_b_gpu_test.py
2_4_c_compare_yolo.py
```

# USING GOOGLE COLLAB.TO CREATE CUSTOM MODEL

- Move the files to google drive
  - my\_data.zip
  - yolov8.obj.det.ak.ipynb



# USING GOOGLE COLLAB.TO CREATE CUSTOM MODEL

- Move the training script to google collab. and execute line by line

```
[ ] 1 from google.colab import drive
2 drive.mount('/content/drive')

Mounted at /content/drive

[ ] 1 # drive.flush_and_unmount()

▼ yolov8 훈련하기 시작

dataset을 저장할 폴더를 만들고 그곳으로 들어갑니다

[ ] 1 %mkdir dataset

Double-click (or enter) to edit

[ ] 1 !cp /content/drive/MyDrive/My_Colab_Notebooks/Doosan/my_data.zip /content/

[ ] 1 !unzip /content/my_data.zip -d dataset/

Archive: /content/my_data.zip
creating: dataset/test/
creating: dataset/test/images/
inflating: dataset/test/images/test_img_8.jpg
inflating: dataset/test/images/test_img_19.jpg
inflating: dataset/test/images/test_img_22.jpg
inflating: dataset/test/images/test_img_13.jpg
inflating: dataset/test/images/test_img_25.jpg
creating: dataset/train/
creating: dataset/train/images/
inflating: dataset/train/images/test_img_23.jpg

✓ Connected to Python 3 Google Compute Engine backend (GPU)
```



# CODING HINTS

---

- Image Capture
- Data Labelling
- Preprocessing
- Yolo8 Object Det (Model)



```
2_3_a_create_data_dirs.py
2_3_b_move_image.py
2_3_c_move_labels.py
2_4_a_yolov8_obj_det_ak.ipynb
2_4_b_gpu_test.py
2_4_c_compare_yolo.py
```

# CODING HINTS

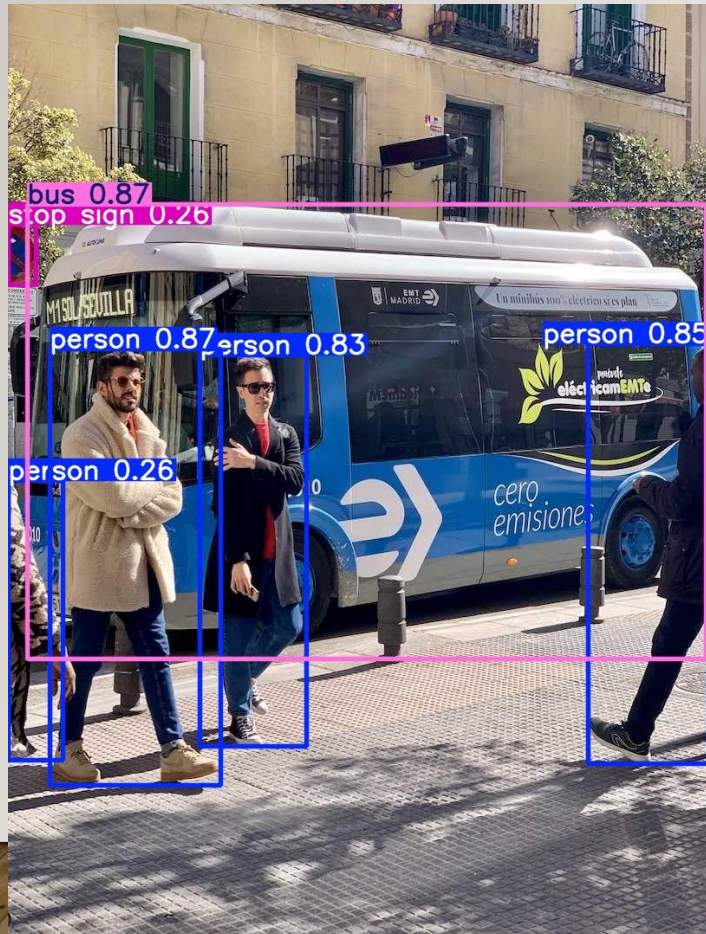
---

- Image Capture
- Data Labelling
- Preprocessing
- Yolo8 Object Det (WEBCAM)

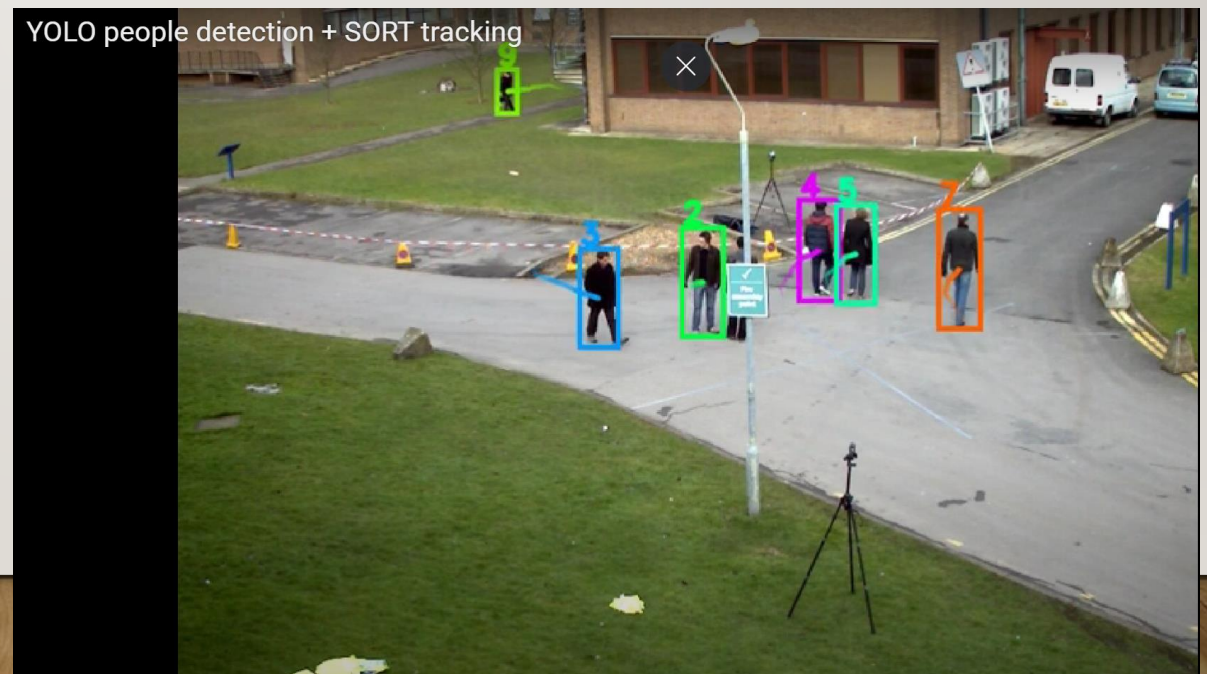


```
2_4_a_yolov8_obj_det_ak.ipynb
2_4_b_gpu_test.py
2_4_c_compare_yolo.py
2_4_d_yolov8_obj_det_wc.py
2_4_e_yolo_publisher_wc.py
2_4_f_yolo_subscriber_wc.py
2_4_g_yolov8_obj_det.py
2_4_h_yolov8_obj_det_thread.py
2_4_i_yolov8_obj_det_track.py
```

# YOLO OBJ. DET. VS. YOLO TRACKING



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






# CODING HINTS

---

- Image Capture
- Data Labelling
- Preprocessing
- Yolo8 Object Det (AMR)



```
2_4_a_yolov8_obj_det_ak.ipynb
2_4_b_gpu_test.py
2_4_c_compare_yolo.py
2_4_d_yolov8_obj_det_wc.py
2_4_e_yolo_publisher_wc.py
2_4_f_yolo_subscriber_wc.py
2_4_g_yolov8_obj_det.py
2_4_h_yolov8_obj_det_thread.py
2_4_i_yolov8_obj_det_track.py
```



# USING DEPTH (ROBOT)

---



# SETUP BASH(ROBOT)

---

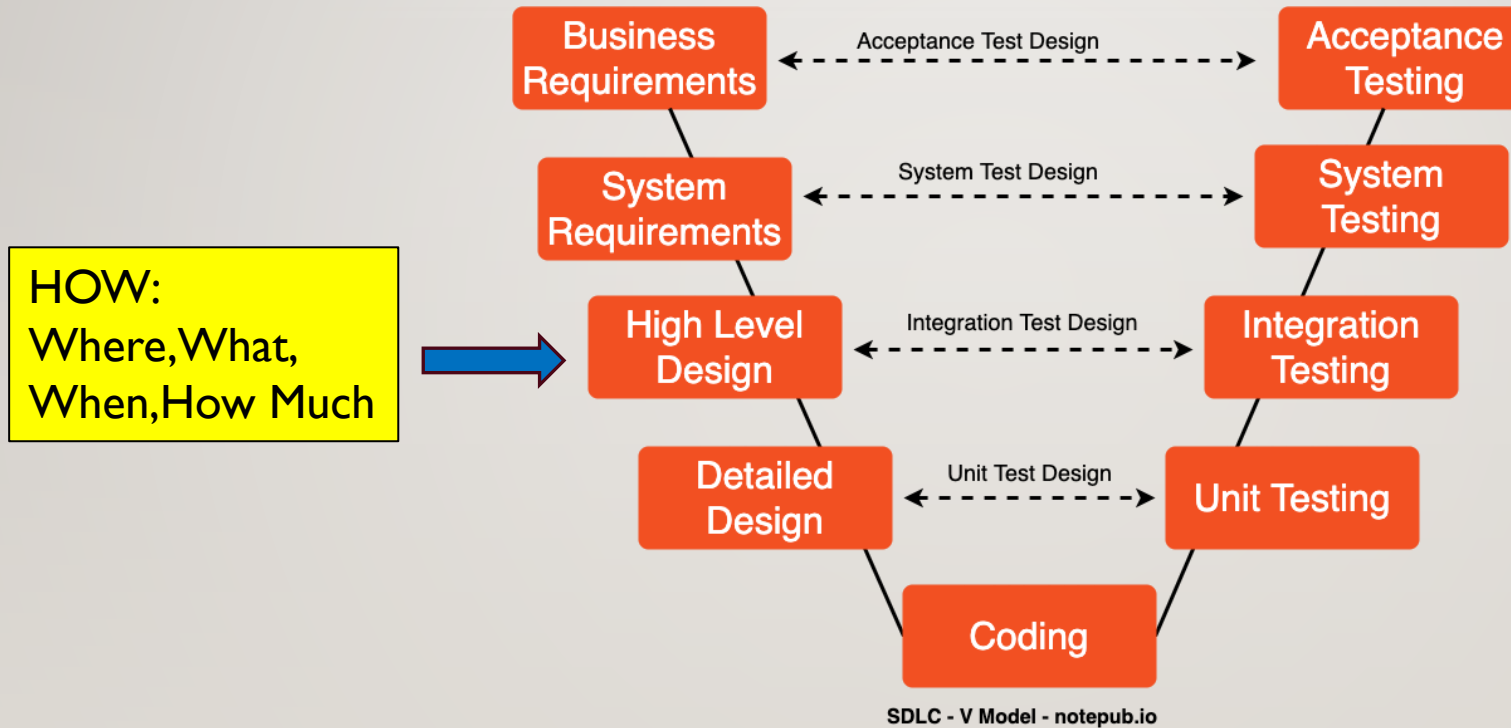
- Make sure bashrc has:
  - `ROS_DOMAIN_ID = 0`
- Make sure discovery setup.bash **is** sourced!
- `source ~/.bashrc`

# EXERCISE

---

- Get depth information from AMR and show a distant to any point on the image
- Hint:
  - Identify the depth topic aligned with rgb image
  - Write a script to subscribe to rgb and depth topics
  - Display the images
  - Select a point in a rgb display
  - Distance to the point is shown on the depth display

# SW DEVELOPMENT PROCESS





# TEAM EXERCISE 3

---

**Update** System Design using Process Flow Diagram.

Use the posted notes and flipchart as needed

# VISUALIZATION – SYSTEM FUNCTIONAL PROCESS FLOW (ARCHITECTURE) DIAGRAMS

- To-Be Functional Process Flow Diagram

Detection Alert  
AMR Controller

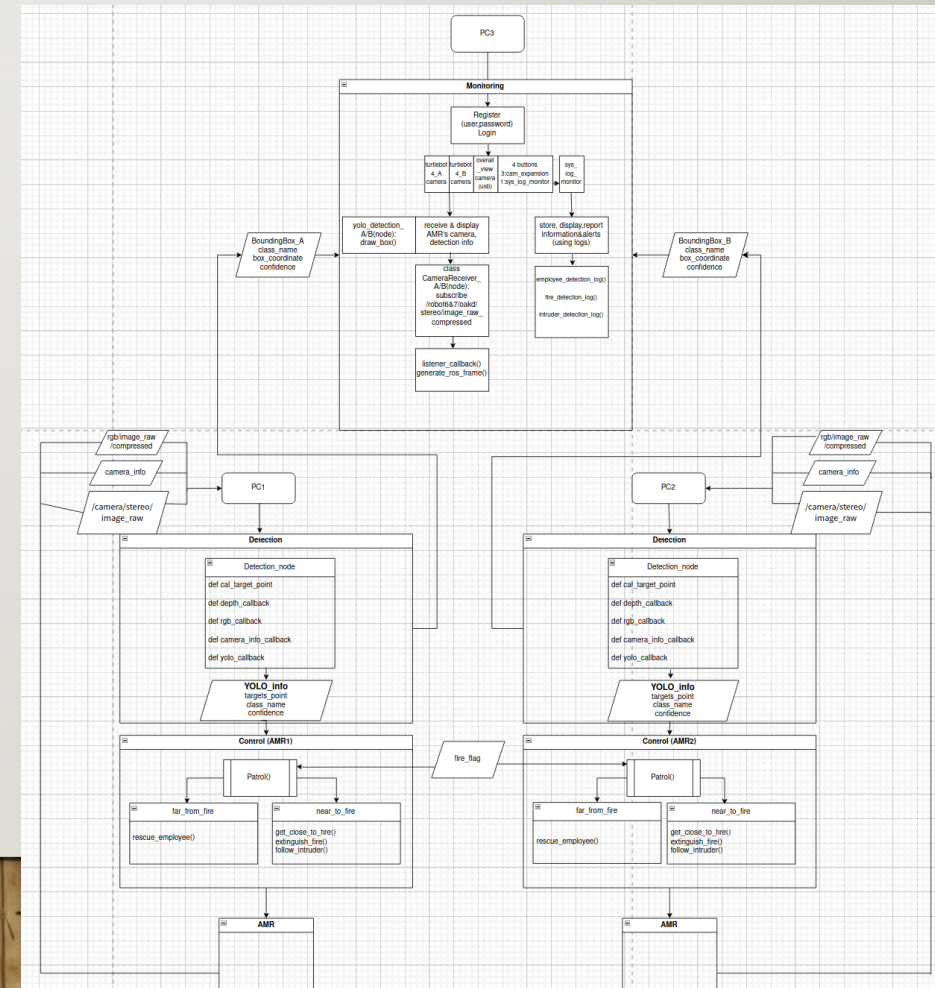
- Functions

- Interfaces

Dataflow

- Testing

Error and Exception Handling



# HOMEWORK

---

- Object Detection
  - Collect various datasets (i.e. different topics/images sizes)
  - Create various models (i.e. v5, v8, v11, etc; arg: Epoch, Batch, Imgsz, augmentation, etc)
  - Analyze the results
  - Determine using key metrics which model best fit your solution
- Depth
  - See exercise sheet above
- Update System Requirement
- Create draft System Design (Architecture Diagram)

# 프로젝트 RULE NUMBER ONE!!!

---

Are we still having  
**FUN!**

