

# GOOD MORNING!

早上好!

안녕하세요!

---

DAY 2

# NEW NOTION LINK

---



# DAY I RECAP

---

# 2 PROJECTS

---

- Mini Project (Individual Team)
  - For learning techniques

차시	구분	세부사항
1	프로젝트 계획 및 환경 구축	시스템 개발 프로세스의 이해, 개발 환경 구축
2	기술 탐색 및 검증	AI VISION 기술 탐색 및 검증
3	기술 탐색 및 검증	AMR 제어 기술 탐색 및 검증
4	기술 탐색 및 검증	Mini project 완성 및 발표

# 2 PROJECTS

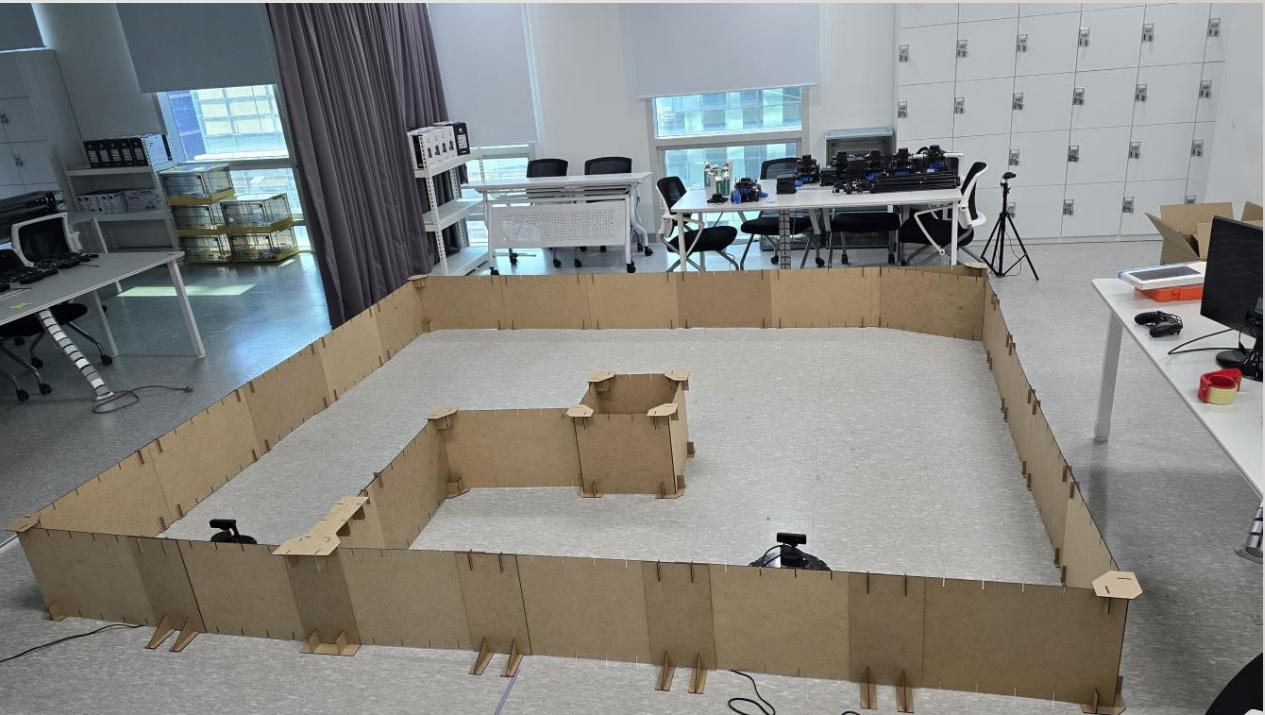
---

- Final Project (2 Teams in One)

차시	구분	세부사항
5	프로젝트 설계	외부 시스템 모니터 기술 탐색 및 검증 파이널 프로젝트 시스템 요구사항 설계 및 프로세스 정립
6	개발	기능 구현 및 Unit Test
7	개발	기능 구현 및 Unit Test
8	개발	통합 시스템 구축 및 테스트
9	개발	통합 시스템 구축 및 테스트
10	최종 프레젠테이션 및 시연	프로젝트 발표 및 시연, 산출물 정리, 기술 컨퍼런스

# 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

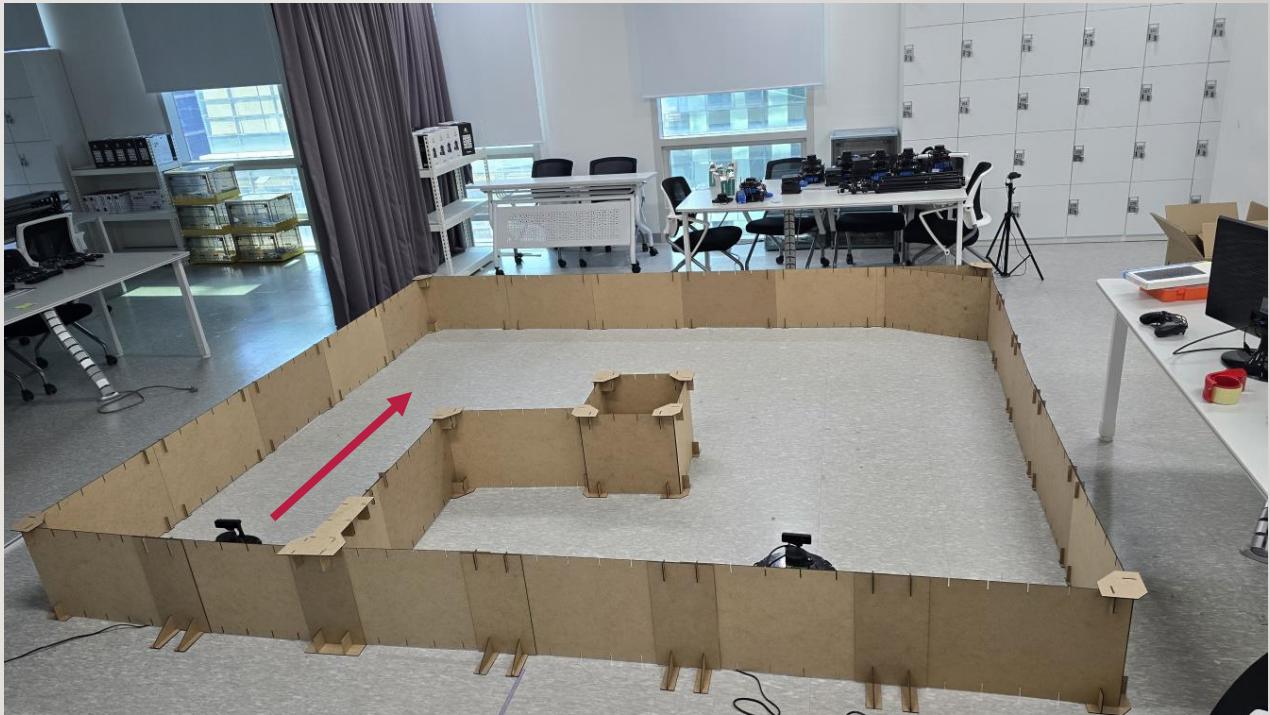
# DETECTION ALERT

---



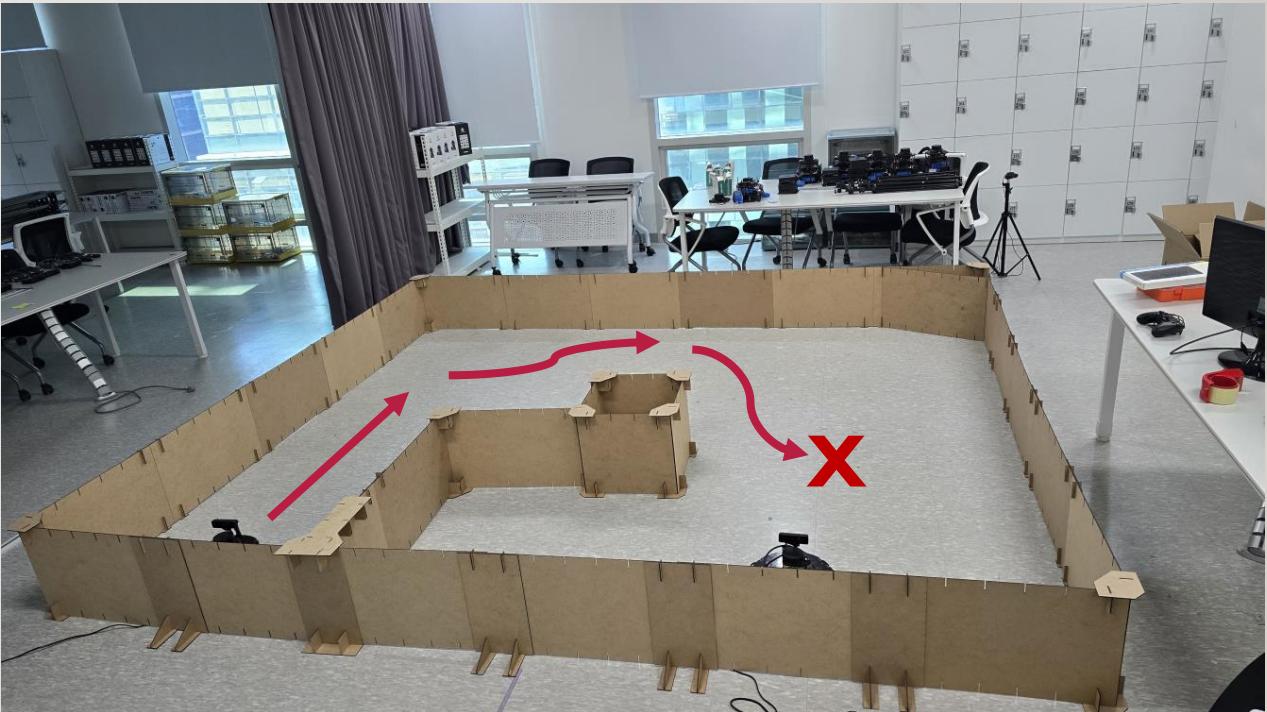
**START**

---



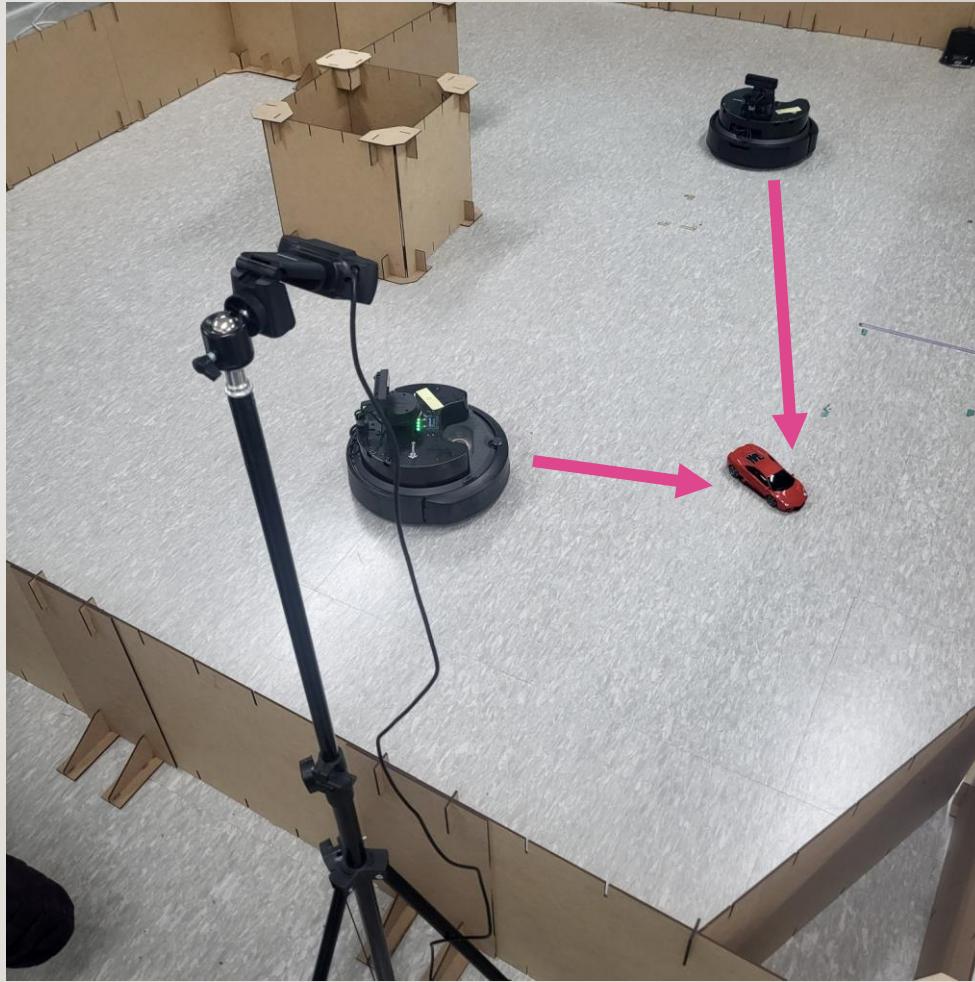
## NAVIGATE TO A POSITION

---



## TRACK & APPROACH

---



# DAY I

---

- Welcome
- Project Introduction
- Introduction to Project Development Process
- Business Requirement Development
- System Requirement Development
- System and Development environment Setup

# DAY 2 (MINI PROJECT)

---

- Yolo 객체 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

- 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 DEMO

- 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

---

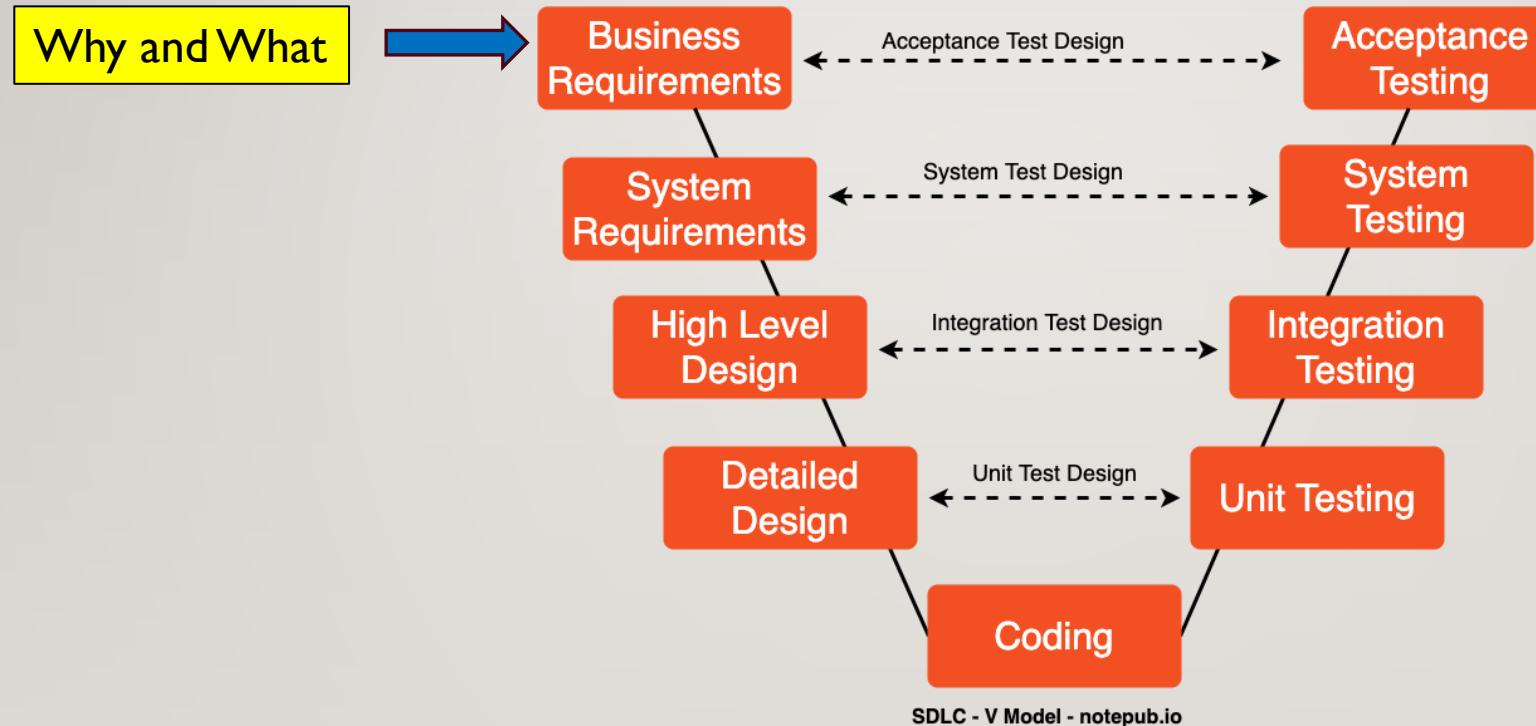
- Every input is good input
- Do not critique inputs only seek to understand
- Organize inputs into logical groupings
- Sequence or show relationships as needed
- Use Posted Notes on Flip Chart



PROJECT DEVELOPMENT IS A PROCESS

---

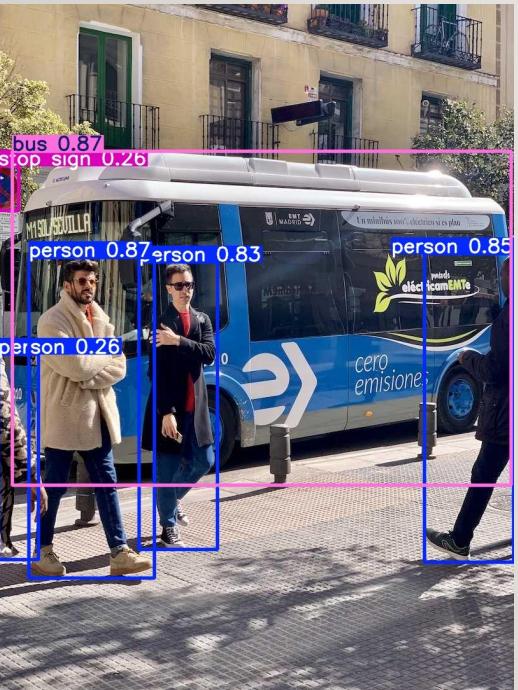
# SW DEVELOPMENT PROCESS



# ADVANCED TECHNIQUES THAT WE HAVE

---

- AI Object Detection

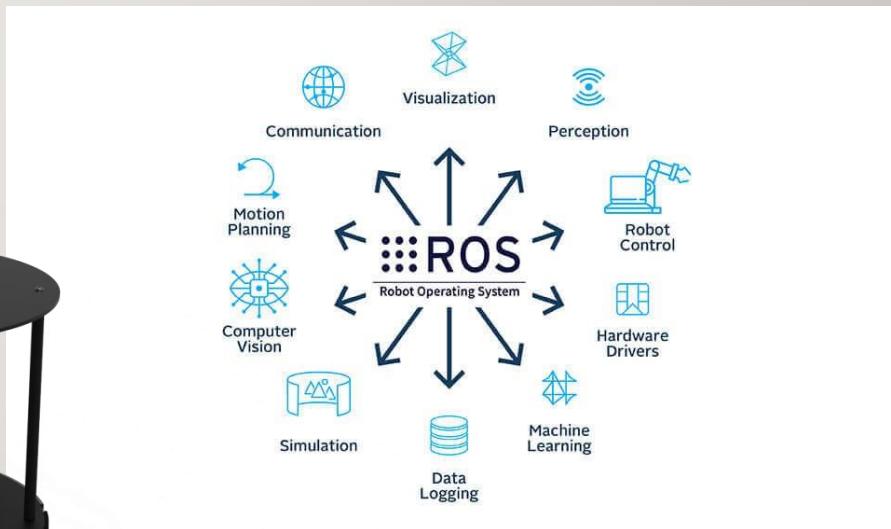


- AMR

- Navigation with obstruction avoidance
- Sensors



- ROS2



# TEAM EXERCISE I

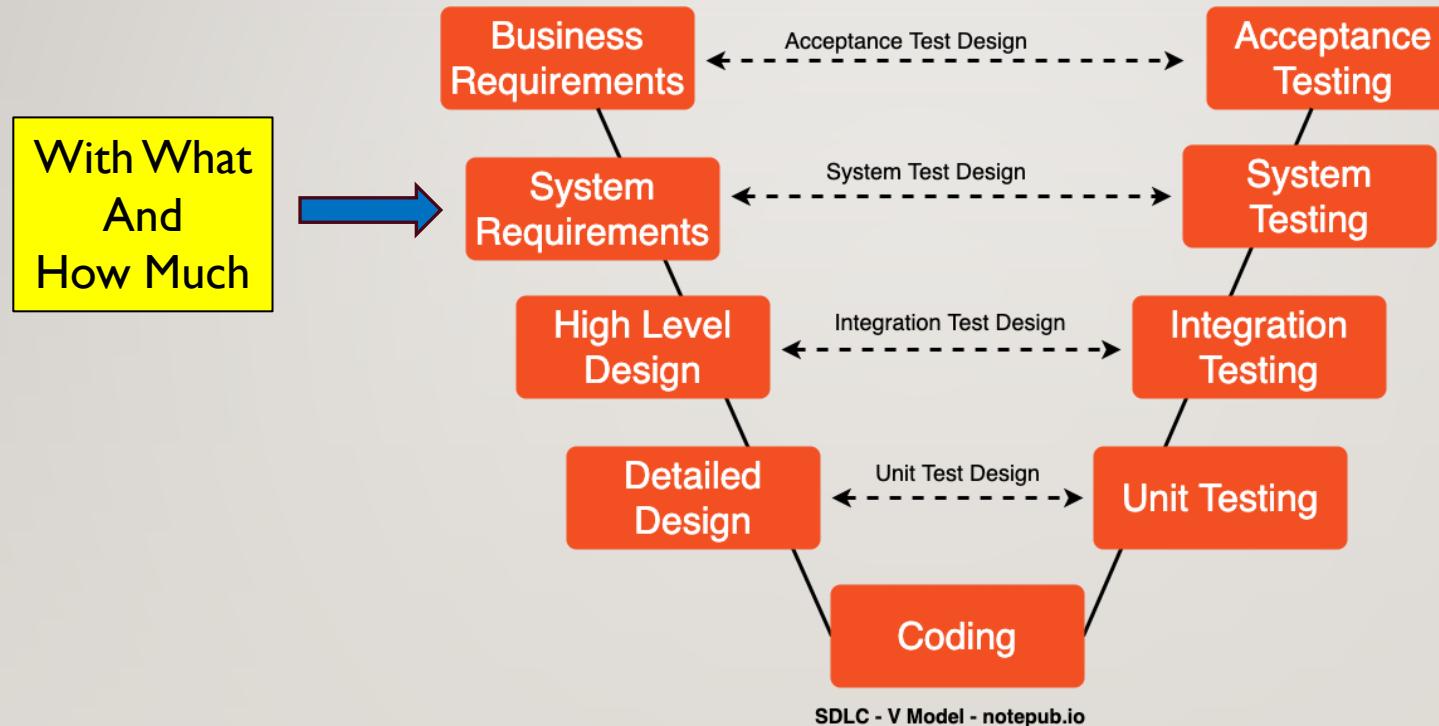
---

Brainstorm Business/Solution Requirement for **Your** Solution and write business requirement statement

Using the posted notes and flipchart as needed

# SW DEVELOPMENT PROCESS

---



# TEAM EXERCISE 2- I

---

Brainstorm System/Solution Requirement for **Your** solution and document

Using the posted notes and flipchart as needed

# YOUR PROJECT ENVIRONMENT

---



# BASE HW/OS

---

- PC
  - Ubuntu 22.04
  - USB Camera



- Network
  - Wifi

- AMR
  - TurtleBot4
  - Ubuntu 22.04



## TEAM EXERCISE 2-2

---

Brainstorm **mini-project** System Requirement for the project and document

Using the posted notes and flipchart as needed

Include where, when, what will be used

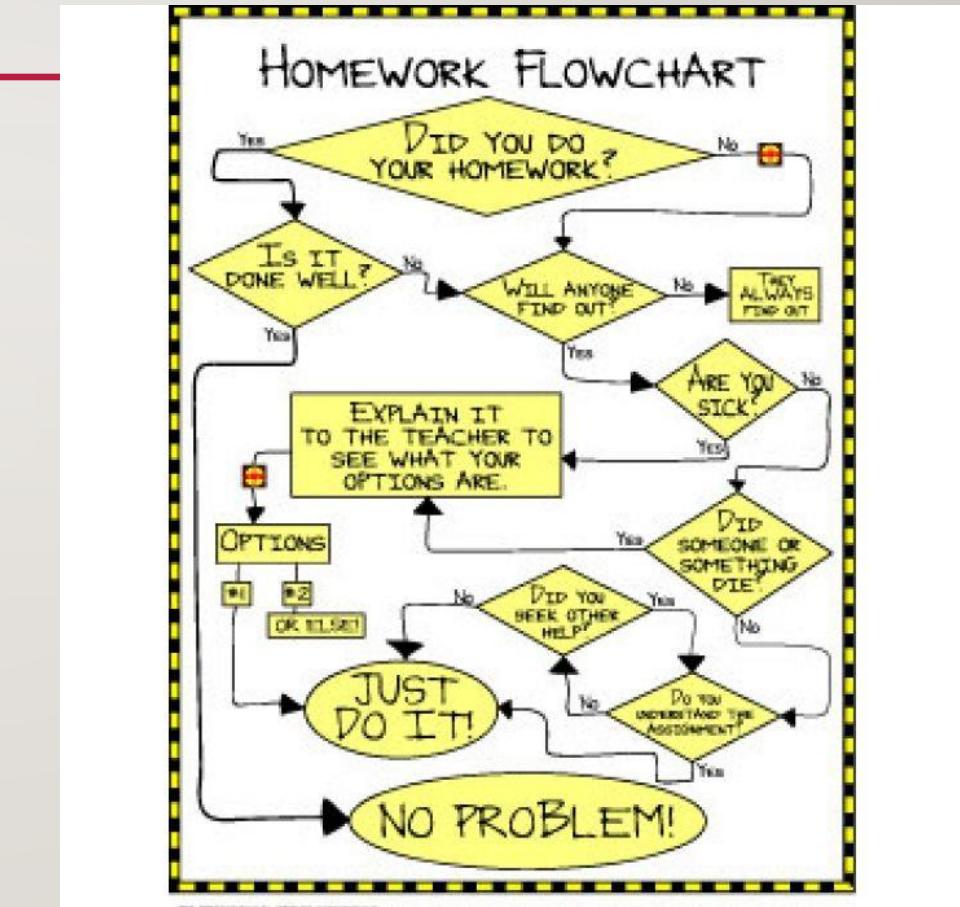
# DEVELOP YOUR BUSINESS SCENARIO (USE-CASE) PROCESS DIAGRAM

---

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/>



# SYSTEM AND DEVELOPMENT ENVIRONMENT SETUP

---

# HOMEWORK CHECK

---

# AMR (TURTLEBOT4)

---

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



# PLEASE REVIEW YOU 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: LabelImg/roboflow)
  - 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

# ROS EXERCISE I

---

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>
```

# ROS EXERCISE 2

## 강의 보조 문서

### DAY 1 - Setup/Development Process

Aa 이름	상태
평가 지표	● 시작 전
Turtlebot4 robot 만나기	● 시작 전
수업 환경 구성	● 시작 전
.bashrc 구성	● 시작 전
Turtlebot4 환경 구성_source	● 시작 전
Turtlebot4 Setup	● 시작 전
User PC Network Setup(Single Robot Setup)	● 시작 전
Move robot CLI	● 시작 전
ROS Workspace Example	● 시작 전
YOLO Setup	● 시작 전
Homework_삐뽀삐뽀 소리내기 노드 만들기	● 시작 전

## 미션 수행

CLI로 동작 확인

```
ros2 topic pub --once /robot<n>/cmd_audio irobot_create
{frequency: 880.0, max_runtime: {sec: 0, nanosec: 300
{frequency: 440.0, max_runtime: {sec: 0, nanosec: 300
{frequency: 880.0, max_runtime: {sec: 0, nanosec: 300
{frequency: 440.0, max_runtime: {sec: 0, nanosec: 300
}]}"
```

- 각 음이 0.3초(300ms) 재생됨
- 880Hz 와 440Hz 를 번갈아 재생 → 경고음 같은 효과

### ▼ 삐뽀삐뽀 소리내기 노드 만들기

- 코드 작성하기

```
#homework
```
- setup.py 설정

# 프로젝트 RULE NUMBER ONE!!!

---

Are we having  
Fun???



# DAY 2

---

# 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
  - Approach target using camera and motor control

**PERFORM DATA COLLECTION FOR  
DETECTION ALERT**

---

# COLLECTION IMAGES FROM WEBCAM

---

- Image Capture (WEBCAM)

[  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

---



- Undock to see camera topics
- Are all camera topics available?
- Which image topic to use?
- Which dimensions and resolution?
- Are Depth and RGB pixel aligned?
- ...

# 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 “{}”
```

# WHICH IMAGE TOPIC TO USE?

---

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

# RGB CAMERA

A screenshot of a digital notebook interface titled "DAY 2 - AI VISION (YOLO)". The list contains the following nodes:

- Homework\_삐뽀삐뽀 소리내기 노드 만들기 (Status: 시작 전)
- RGB Camera 이해** (Status: 시작 전) — This node is highlighted with a blue border.
- Depth Preview (Status: 시작 전)
- Depth Camera 이해 (Status: 시작 전)
- Robot\_Depth (Status: 시작 전)

At the bottom left is a "New page" button, and at the top right is a "New" button.

A screenshot of a ROS package configuration interface for the "RGB Camera 이해" node. The interface includes the following fields:

- 환경: ubuntu22.04 humble
- 키워드: Empty
- status: ● 시작 전
- 순서: 2-1
- 주제: Empty

Below this is a section titled "RGB 카메라 토픽 확인" containing the command:

```
ros2 topic list | grep oakd
```

# 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	<input checked="" type="checkbox"/> 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

# USING DEPTH

---

# DEPTH INTRO

+ New page

---

## DAY 2 - AI VISION (YOLO)

Aa 이름 status +

Homework_삐뽀삐뽀 소리내기 노드 만들기	● 시작 전
RGB Camera 이해	● 시작 전
Depth Preview	● 시작 전
Depth Camera 이해	● 시작 전
Robot_Depth	● 시작 전

온영 + New page

## Depth Preview

환경	ubuntu22.04 humble
키워드	Empty
status	● 시작 전
순서	2-3
주제	Empty
+ Add a property	

🔍 어떤 원리로 동작하나?

1. 스테레오 카메라 (왼쪽/오른쪽) 간의 시차(disparity)를 계산

# RGB AND DEPTH ALIGNMENT

+ New page

---

## DAY 2 - AI VISION (YOLO)

Aa 이름	status
🤖 Homework_삐뽀삐뽀 소리내기 노드 만들기	● 시작 전
🤖 RGB Camera 이해	● 시작 전
🤖 Depth Preview	● 시작 전
🤖 Depth Camera 이해	● 시작 전
🤖 Robot_Depth	● 시작 전

+ New page

Depth Camera 이해

환경: ubuntu22.04, humble

키워드: Empty

status: ● 시작 전

순서: 2-4

주제: Empty

+ Add a property

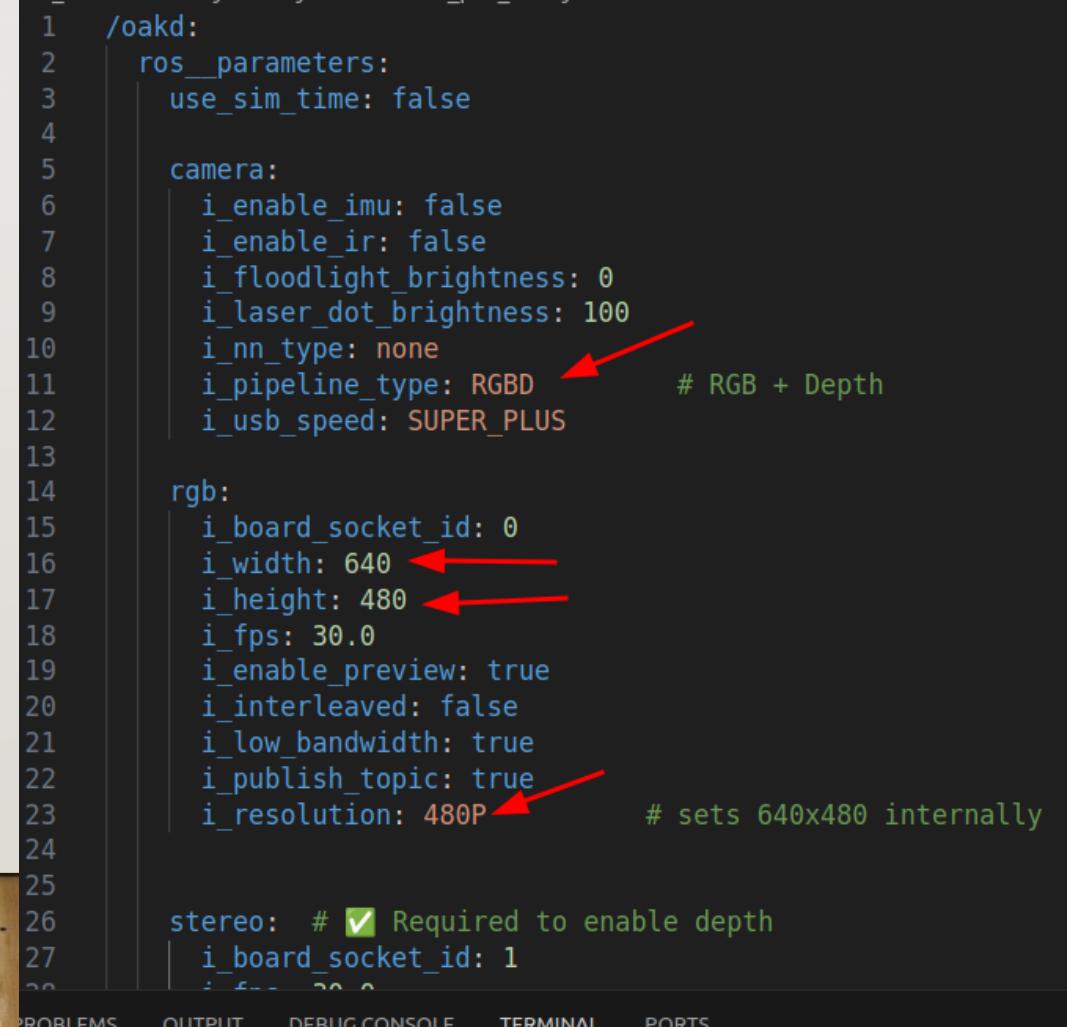
Depth 카메라 토픽 확인

# 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: #  Required to enable depth  
27     i_board_socket_id: 1  
28
```

# DEPTH VALUE FROM IMAGE

Aa 이름 상태 +

+ New page

## DAY 2 - AI VISION (YOLO)

Aa 이름 status +

Homework_삐뽀삐뽀 소리내기 노드 만들기	● 시작 전
RGB Camera 이해	● 시작 전
Depth Preview	● 시작 전
Depth Camera 이해	● 시작 전
Robot_Depth	● 시작 전

+ New page

Robot\_Depth

: 환경 ubuntu22.04 humble

: 키워드 Empty

: status ● 시작 전

: 순서 2-5

: 주제 Empty

+ Add a property

### 학습 목표

Depth 토픽 활성화

# AIM OF THE GOOD OBJ. DET. TRAINING SET

---

- Training images matches the real inference images as much as possible
  - Lighting
  - Dimensions
  - FOV
  - Backgrounds
  - Objects
  - ...
- For RGB and Depth and ...

# CODING HINTS

---

- Image Capture
- Image Capture (AMR)

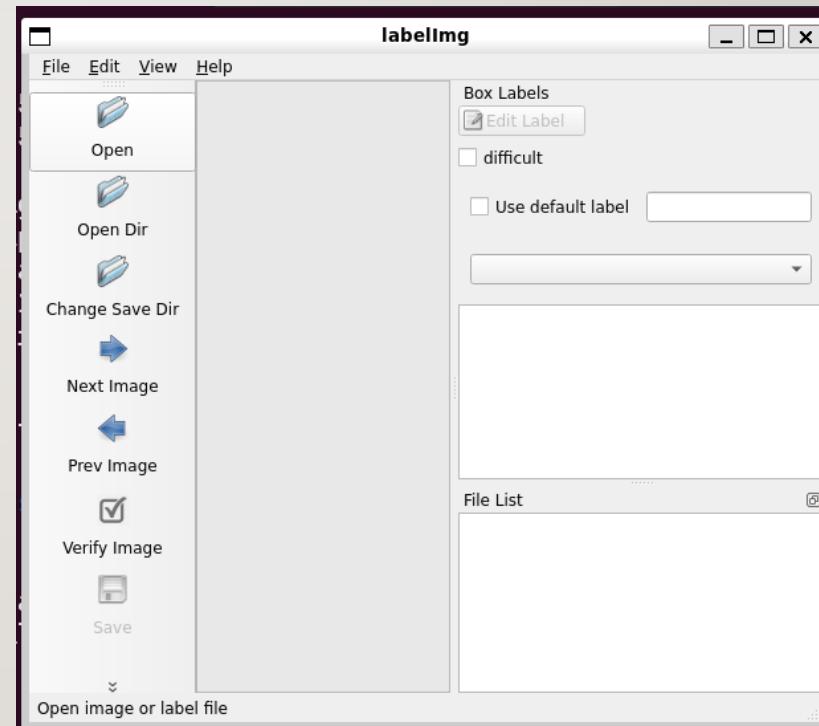


- 2\_1\_a\_capture\_wc\_image.py
- 2\_1\_b\_cont\_capture\_wc\_image.py
- 2\_1\_c\_capture\_wc\_thread.py
- 2\_1\_d\_capture\_image.py
- 2\_1\_e\_cont\_capture\_image.py

# CODING HINTS

---

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



# CODING HINTS

---

- Data Labelling : LabelImg

## 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

---

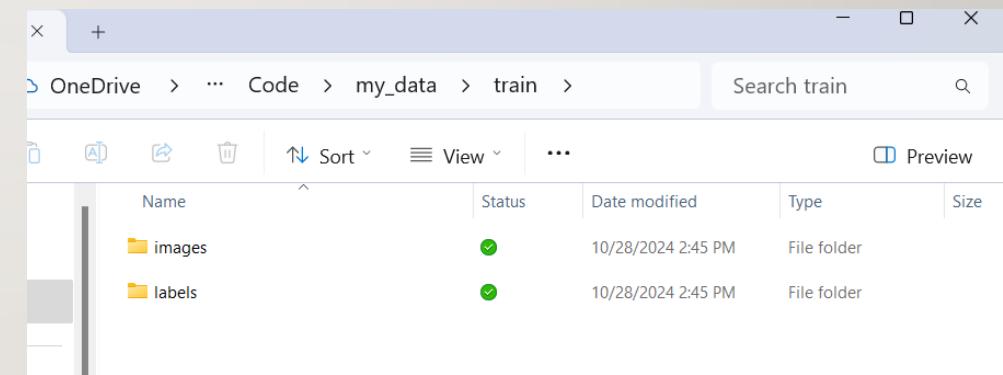
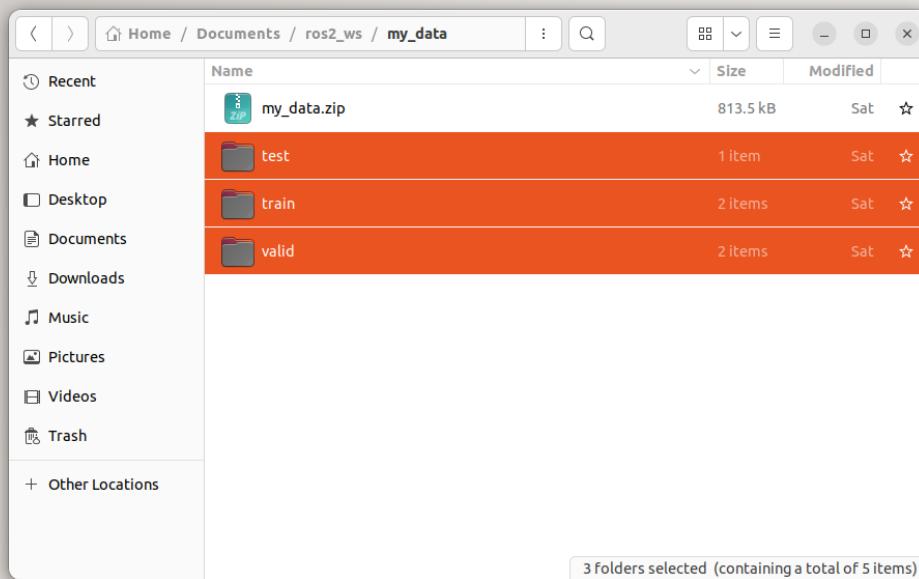
- Image Capture
- Data Labelling
- Data Preprocessing



- 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)

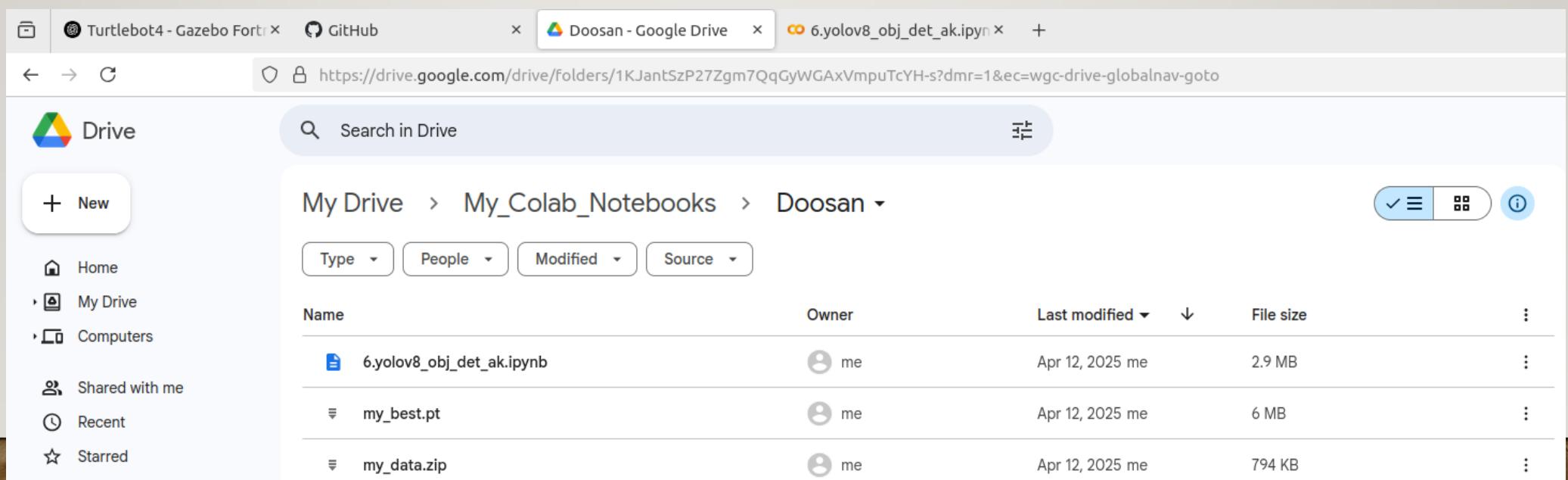


- 2\_3\_a\_create\_data\_dirs.py
- 2\_3\_b\_move\_image.py
- 2\_3\_c\_move\_labels.py
- 2\_4\_a\_yolov8\_obj\_det\_ak.ipynb
- 2\_4\_b\_gpu\_test.py
- 2\_4\_c\_compare\_yolo.py

# USING GOOGLE COLAB TO CREATE CUSTOM MODEL

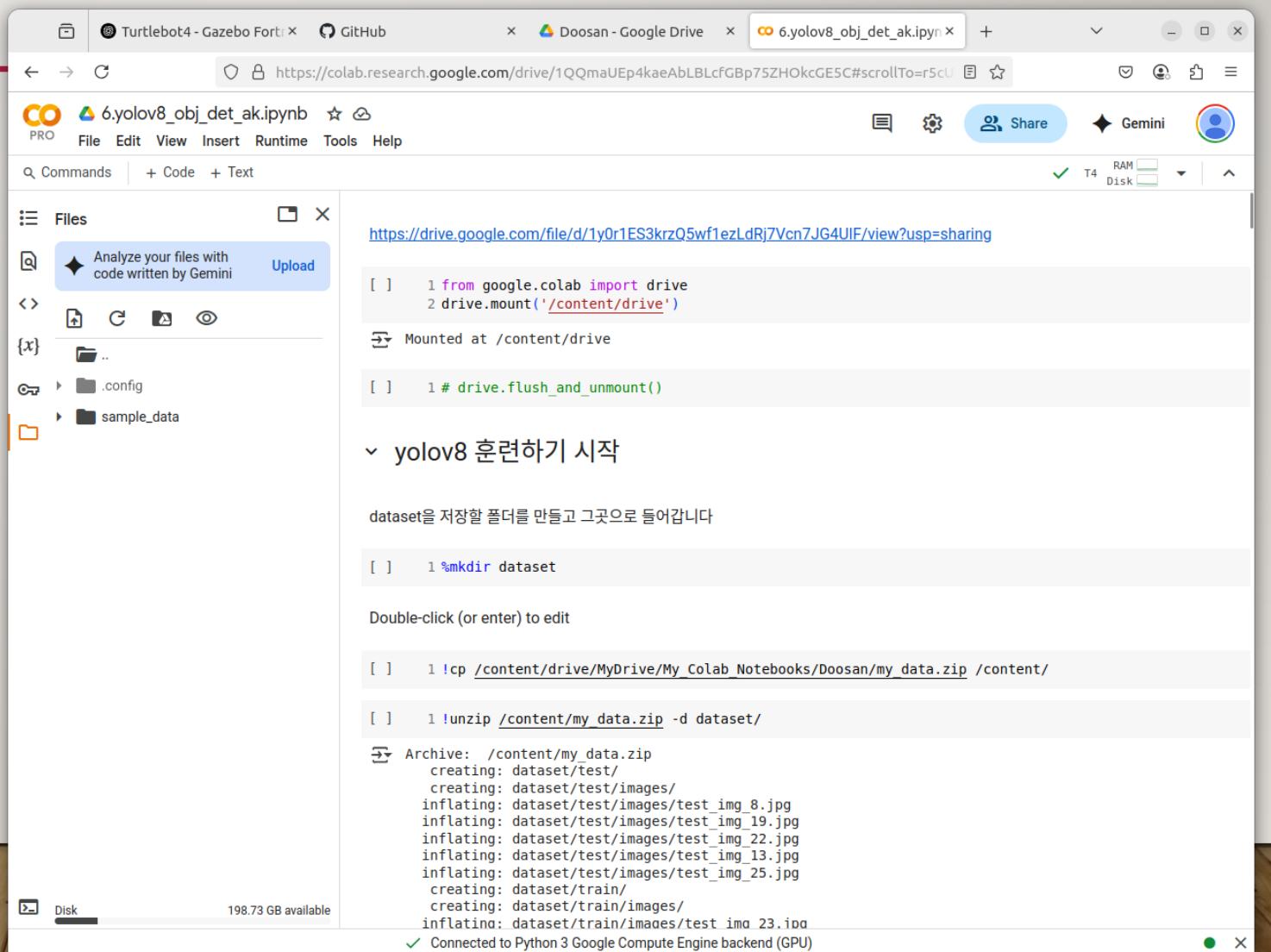
---

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



# USING GOOGLE COLAB TO CREATE CUSTOM MODEL

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



The screenshot shows a Google Colab notebook titled "6.yolov8\_obj\_det\_ak.ipynb". The sidebar displays a file tree with ".config" and "sample\_data" folders. The main area shows the following code execution:

```
https://drive.google.com/file/d/1y0r1ES3krzQ5wf1ezLdRj7Vcn7JG4UIF/view?usp=sharing

[ ] 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)



- ➊ 2\_3\_a\_create\_data\_dirs.py
- ➋ 2\_3\_b\_move\_image.py
- ➌ 2\_3\_c\_move\_labels.py
- ➍ 2\_4\_a\_yolov8\_obj\_det\_ak.ipynb
- ➎ 2\_4\_b\_gpu\_test.py
- ➏ 2\_4\_c\_compare\_yolo.py

# CODING HINTS

---

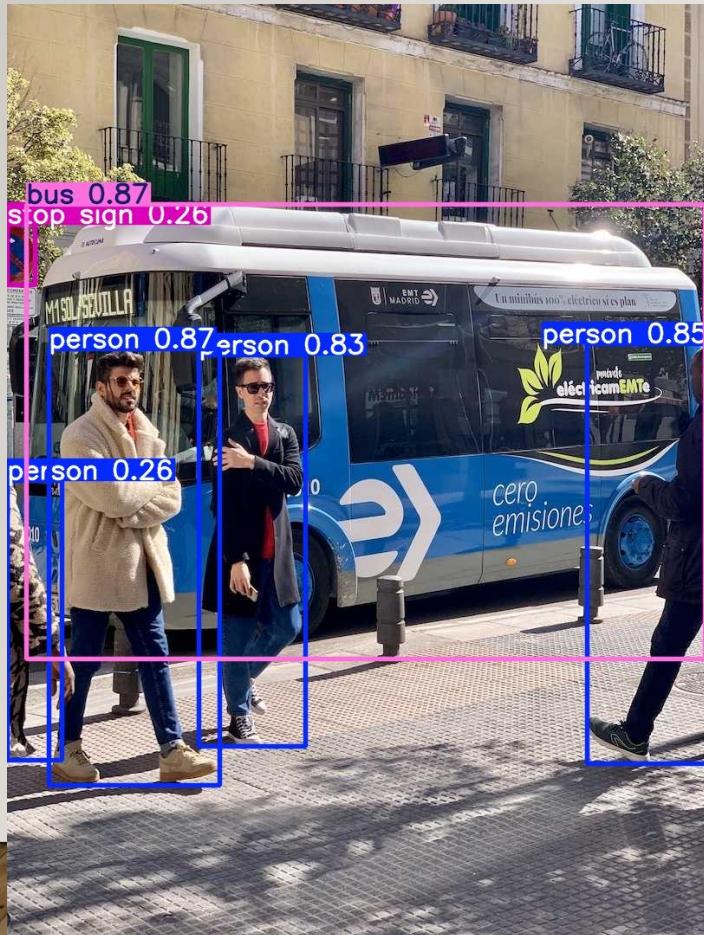
- Image Capture
- Data Labelling
- Preprocessing
- Yolo8 Object Det (WEBCAM)



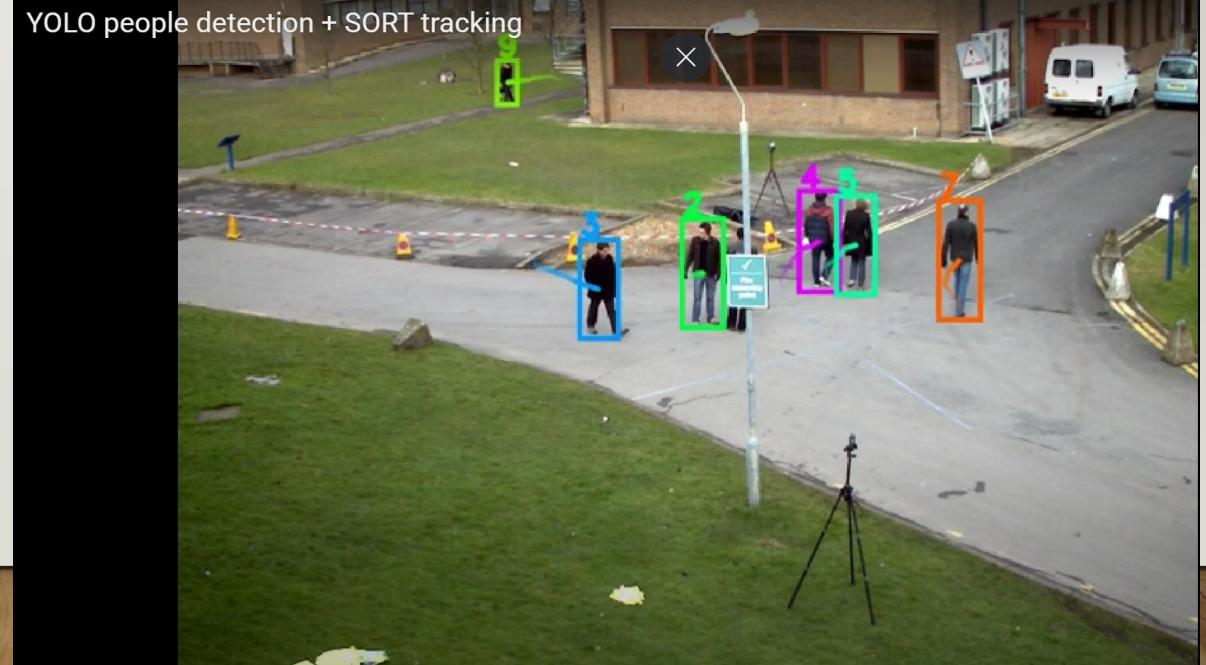
```
 2_4_a_yolov8_obj_det_ak.ipynb  
 2_4_b_gpu_test.py  
 2_4_c_compare_yolo.py  
 2_4_d_yolov8_obj_det_wc.py  
 2_4_e_yolo_publisher_wc.py  
 2_4_f_yolo_subscriber_wc.py
```

# 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

# REQUIRED RESEARCH

---

1. 왜 yolov8n.pt 모델을 선정하였습니까?

- yolo 다른 버전과 비교 분석
- mAPVS Inference speed

2. 객체 탐지 속도를 높이기 위한 최선의 전략은?

- 데이터 사이즈
- processing 방식
- 노드 구조

3. 다른 Pre-trained model(Huggingface)을 사용한다면?

4. object detection이 아닌 segmentation, pose, obb 등을 활용할 수 없을까?

# 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
  - Using .pt file to predict/inference on pc
- Depth
  - Find and display the distance to the center of the detected objects
- Update System Requirement

# 프로젝트 RULE NUMBER ONE!!!

---

Are we still having  
FUN!

