

# GOOD MORNING!

早上好!

안녕하세요!

---

DAY 3



# DAY I

---

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

# 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
- YOLOv8 기반 데이터 수집/학습/deploy (Detection Alert)
  - 감시용 데이터 수집(bus, truck, tank 등)
  - 감시용 데이터 라벨링
  - YOLOv8 기반 학습
  - YOLOv8 Object Detection
- Porting to ROS
  - Create Detection Alert Node
  - Generate Topics to send image and Obj. Det. results
  - Create Subscriber node and display image and print data from the Topic

# DAY 3 (MINI PROJECT)

---

- 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(bus, truck, tank 등)
  - Tracking 데이터 라벨링
  - YOLOv8 기반 학습
  - YOLOv8 Object **Tracking**
- Turtlebot4 시뮬레이션 환경 구축
  - SLAM과 Map 생성 및 파라미터 튜닝 (Localization, AMCL)
  - AutoSLAM으로 맵 생성

# DAY 4 (MINI PROJECT)

---

- Turtlebot4 API를 활용한 Initial Pose Navigate\_to Pose 구현
- Turtlebot4 API를 활용한 Navigate\_Through\_pose, Follow Waypoints 구현
- 로봇 개발 환경에 적용 및 테스트 / 로봇 H/W, 제반 환경의 한계점 도출
- AMR기반 카메라 인식 autonomous driving 시스템 with obstacle avoidance 구축 (AMR Controller)
  - Digital Mapping of environment
  - Goal Setting and Obstacle Avoidance using Navigation
  - Object Tracking w/ AMR camera
  - Control logic between navigation/obj. tracking/ obj. following (teleop)
- Porting to ROS
  - Create AMR Controller Node
  - Create and send Obj.Tracking Image and data to Sysmon
- Integrate and test with Detection



프로젝트 RULE NUMBER ONE!!!

---

Have Fun Fun Fun!



# TEAM EXERCISE 3

---

Create System Design using Process Flow Diagram.

Use the posted notes and flipchart as needed

# 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]❧

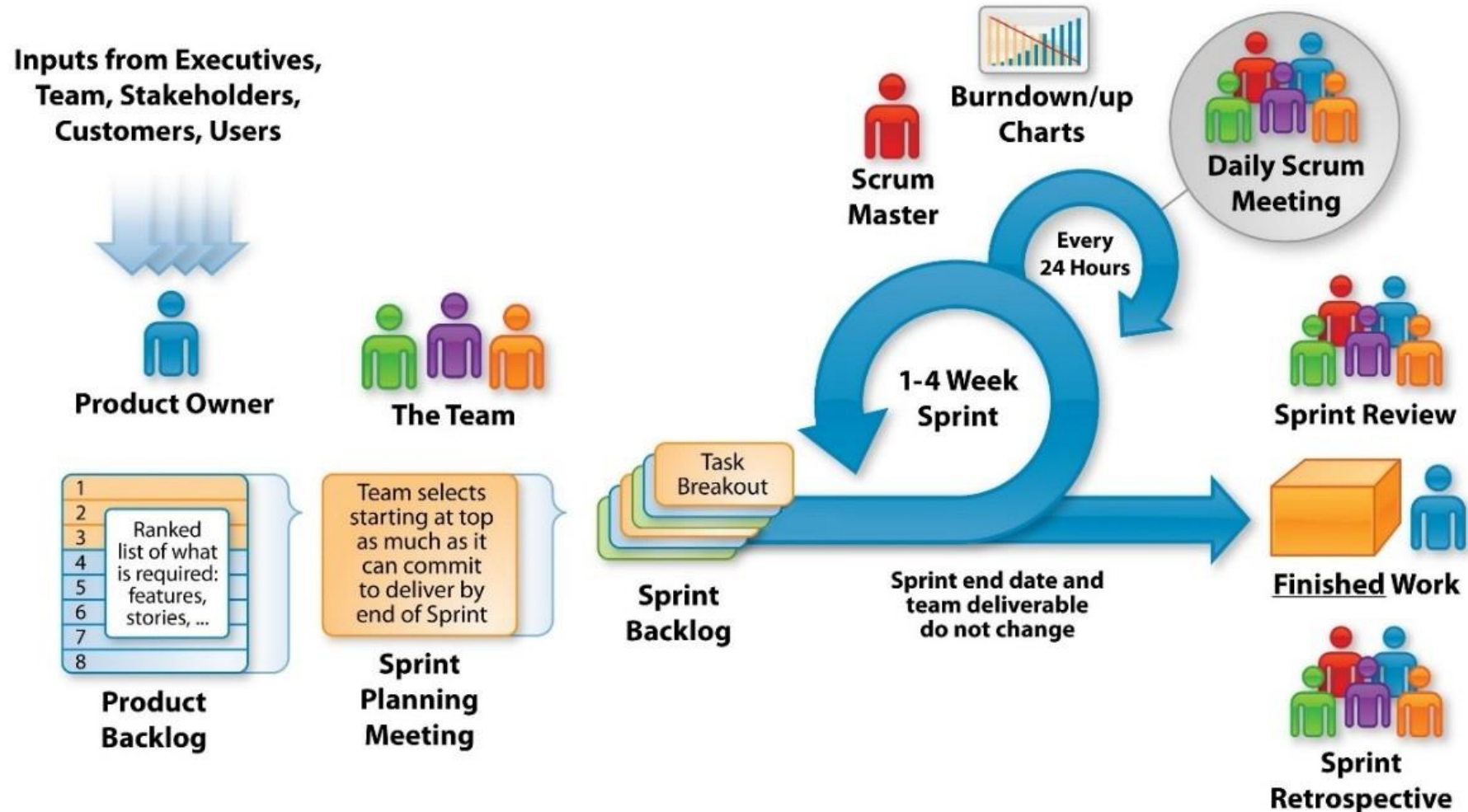
### 1. 개요❧

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.❧

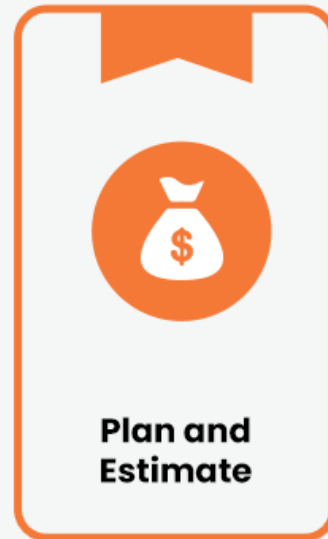
# The Agile - Scrum Framework



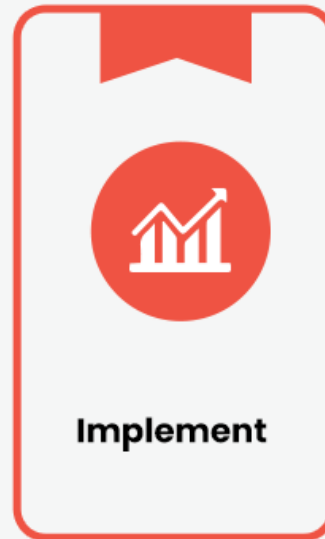
# 5 Stages of Scrum Sprint



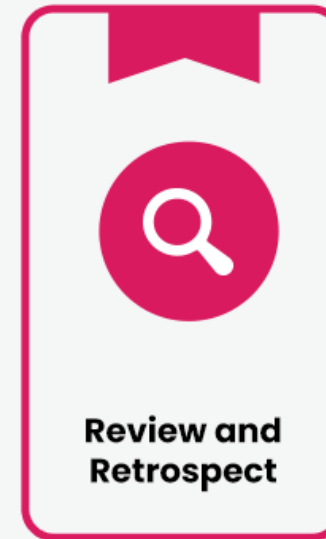
This phase includes the processes related to the commencement of a project, such as a scope and objectives, creating and distributing its charter, and taking other steps to guarantee success.



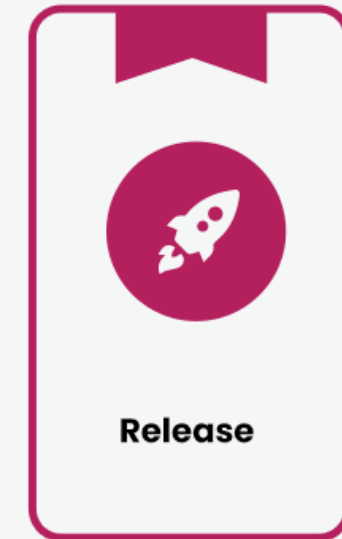
This phase involves planning and estimating processes, including creating user stories, approving, assessing, committing user stories, creating tasks, evaluating tasks, and creating a Sprint backlog.



This phase is about executing the tasks and activities to create a product. These activities include building the various outputs, conducting daily standup meetings, and grooming the product backlog.

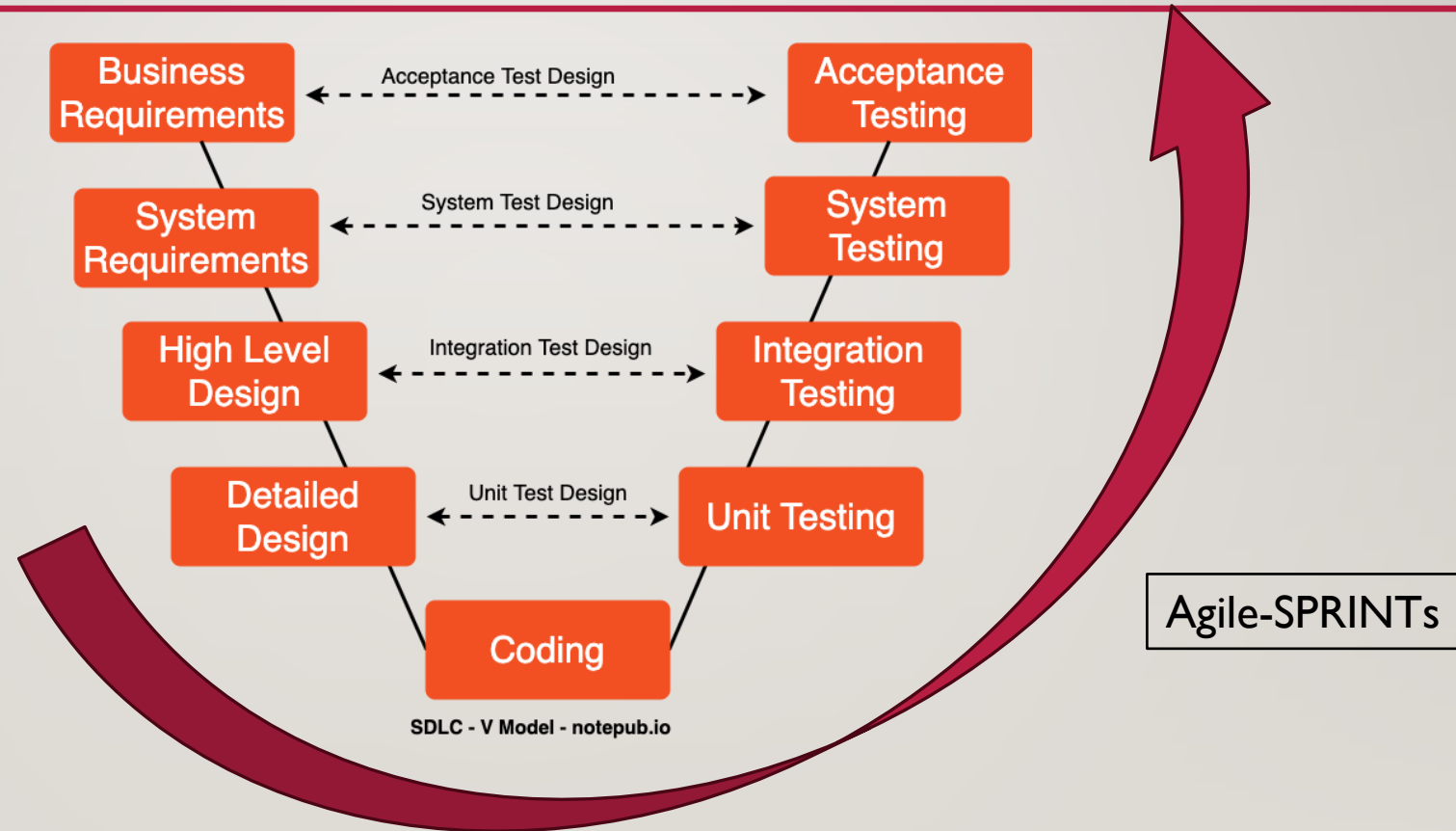


This stage of the project lifecycle is concerned with evaluating what has been accomplished so far, whether the team has worked to plan, and how it can do things better in the future.



This stage highlights delivering the accepted deliverables to the customer and determining, documenting, and absorbing the lessons learned during the project.

# SW DEVELOPMENT PROCESS





# PROJECT SPRINTS

- 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

- System Monitor

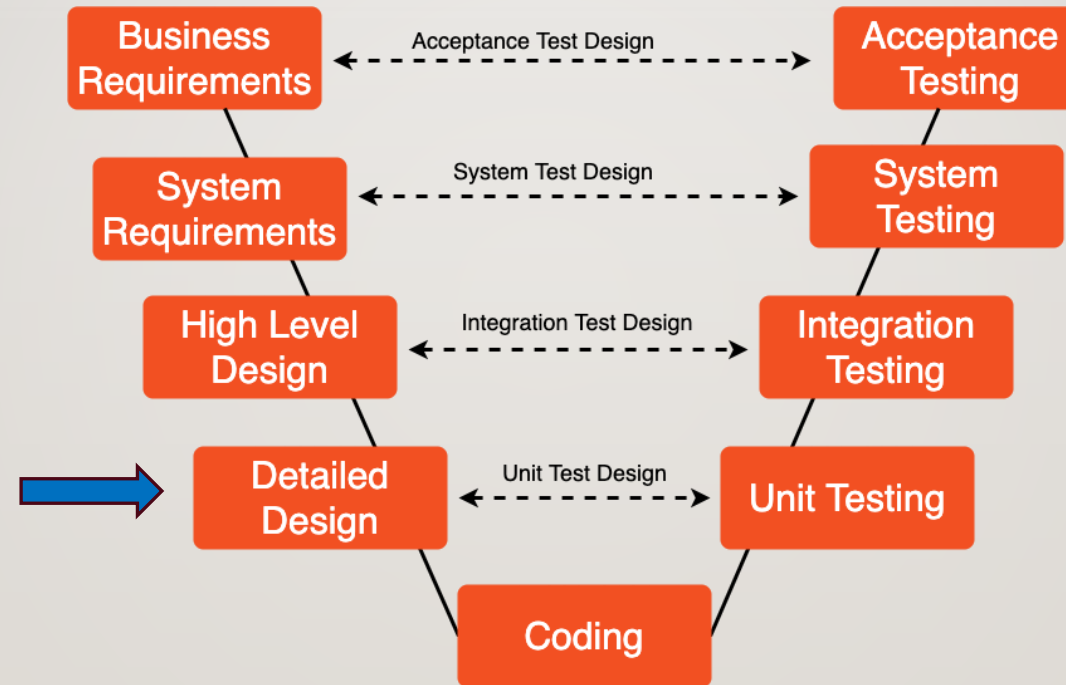
- Receive and Display Detection Camera and info
- Receive and Display AMR Camera and info
- Store, display, and report Information and Alerts

# DETECTION ALERT SPRINT

---



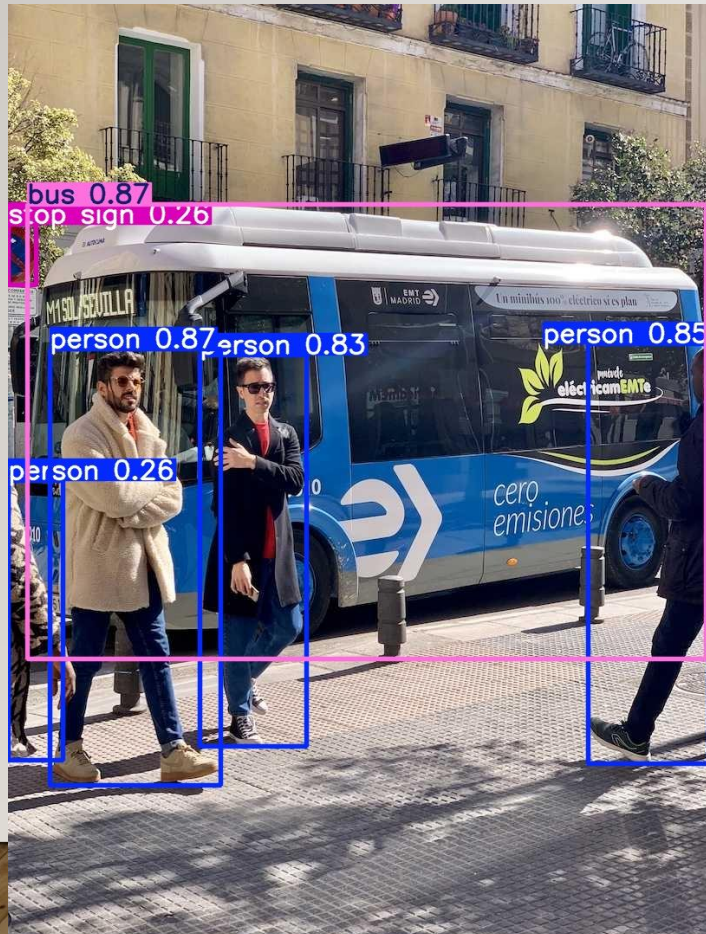
# SPRINT I - DETECTION ALERT



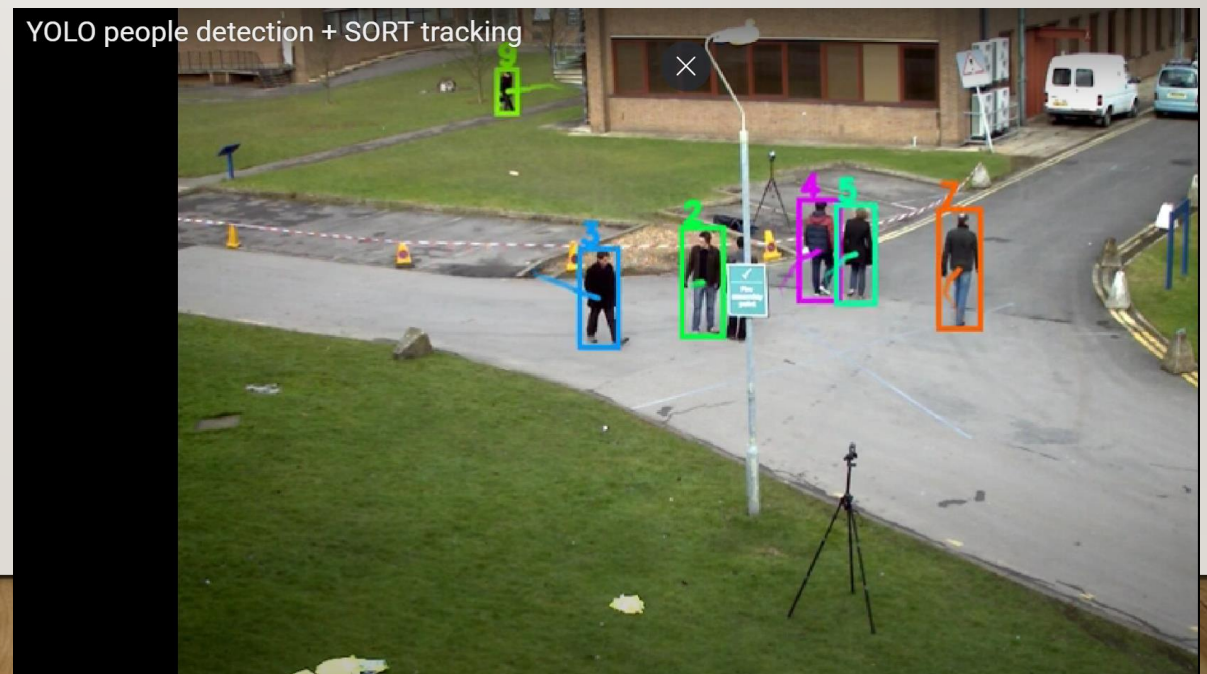
SDLC - V Model - notepub.io



# YOLO OBJ. DET. VS. YOLO TRACKING



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





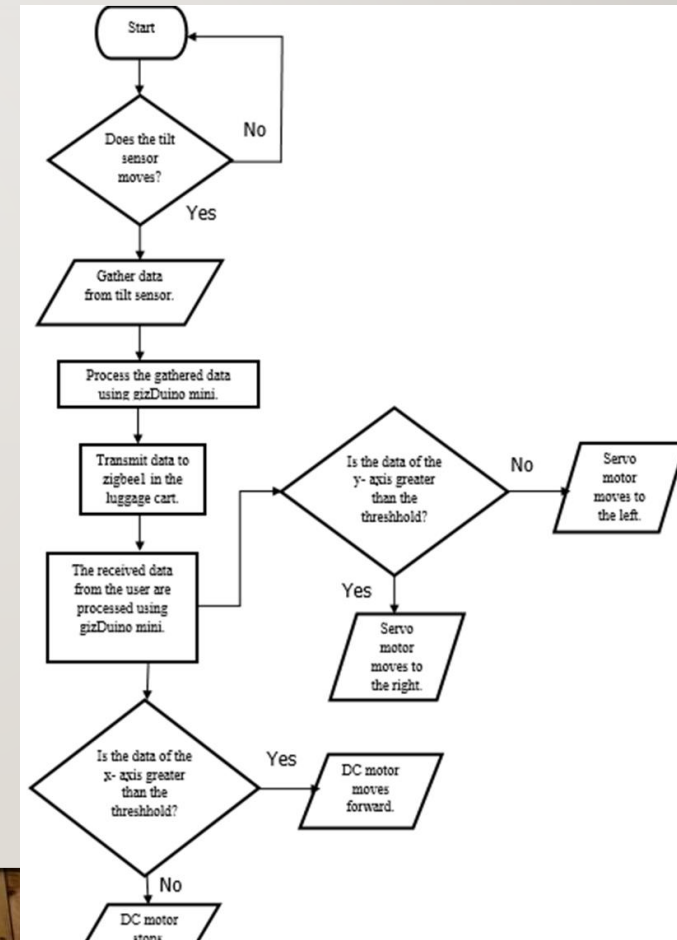
# VISUALIZATION – DETAILED FUNCTIONAL PROCESS DIAGRAMS

- To-Be Functional Process Diagram

Detection Alert

PC or AMR or Both

Remember **F.I.T!!!**



# TEAM EXERCISE 4

---

Perform Detail Design of Detection Alert Module using Process Flow Diagram

# DETAIL DESIGN REVIEW BY EACH TEAM

---

Using the process flow diagram present team's design

# EXAMPLE DETAILED DESIGN DOCUMENT

## Detailed Design Document: AMR Navigation and Threat Detection

Project Title: Autonomous Mobile Robot (AMR) Security System

Version: 1.0

Date: [Insert Date]

### 1. Overview

This document outlines the detailed design for the Autonomous Mobile Robot (AMR) navigation and threat detection components. It covers the architecture, algorithms, data processing, and system interactions necessary to enable autonomous navigation within a secure area and real-time threat detection using onboard sensors.

### 2. System Architecture

The AMR system relies on onboard hardware (e.g., sensors, cameras, Jetson-Orin processor) and software (ROS2, OpenCV, YOLO) for autonomous navigation and real-time threat detection. All processing occurs locally on the AMR, with the capability to transmit alerts to a monitoring PC via Wi-Fi.

## 상세 설계 문서: AMR 네비게이션 및 위협 탐지

프로젝트 제목: 자율 이동 로봇(AMR) 보안 시스템

버전: 1.0

날짜: [날짜 삽입]

### 1. 개요

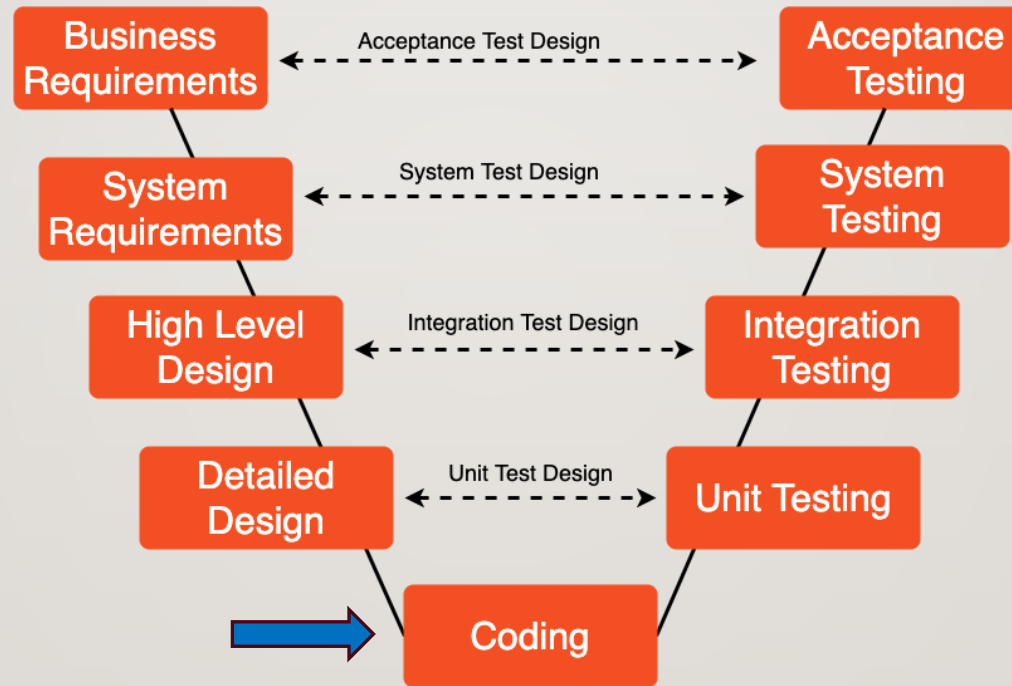
이 문서는 자율 이동 로봇(AMR)의 네비게이션 및 위협 탐지 구성 요소에 대한 상세 설계를 다룹니다. 자율 네비게이션과 실시간 위협 탐지를 위해 온보드 센서를 사용하는 데 필요한 아키텍처, 알고리즘, 데이터 처리 및 시스템 상호작용이 포함되어 있습니다.

### 2. 시스템 아키텍처

AMR 시스템은 자율 네비게이션 및 실시간 위협 탐지를 위해 온보드 하드웨어(예: 센서, 카메라, Jetson-Orin 프로세서)와 소프트웨어(ROS2, OpenCV, YOLO)를 활용합니다. 모든 처리는 AMR 내에서 로컬로 수행되며, 잠재적인 위협이 감지되면 Wi-Fi를 통해 모니터링 PC로 알림을 전송할 수 있습니다.



# SPRINT I - DETECTION ALERT



SDLC - V Model - notepub.io

# PERFORM DATA COLLECTION FOR DETECTION ALERT

---



# CODING HINTS

---

- Image Capture

▼ day2

🔗 `__init__.py`

🔗 `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

- Data Labelling
  - labellmg

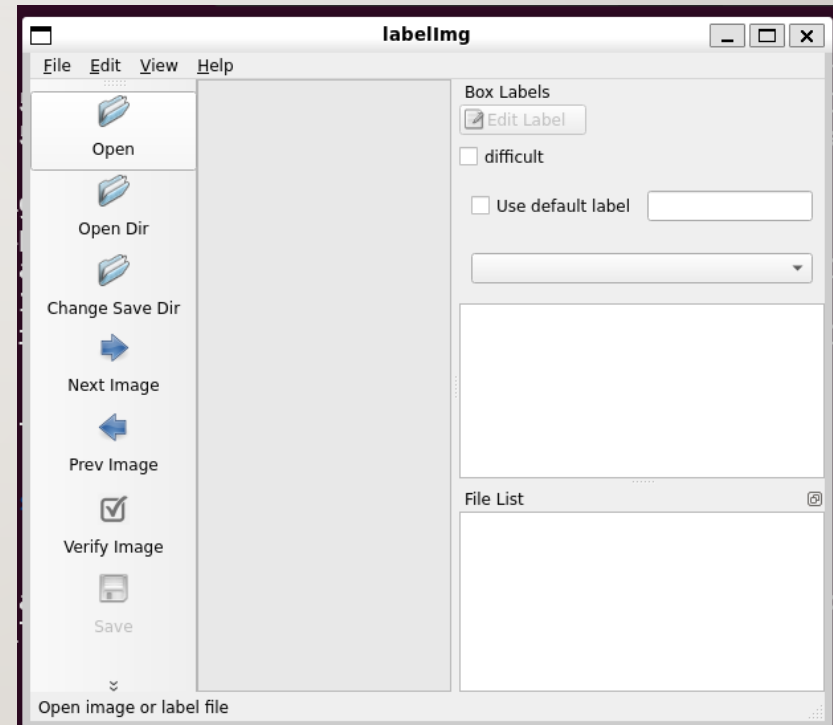
```
▼ day2
  ├── __init__.py
  ├── 2_1_a_capture_wc_image.py
  ├── 2_1_b_cont_capture_wc_image.py
  └── 2_1_c_capture_wc_thread.py
```



# CODING HINTS

---

- Data Labelling : use previously installed Labellmg
- Or
- pip3 install labellmg
  - May also need “pip3 install PyQt5 lxml”
- labellmg



# 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

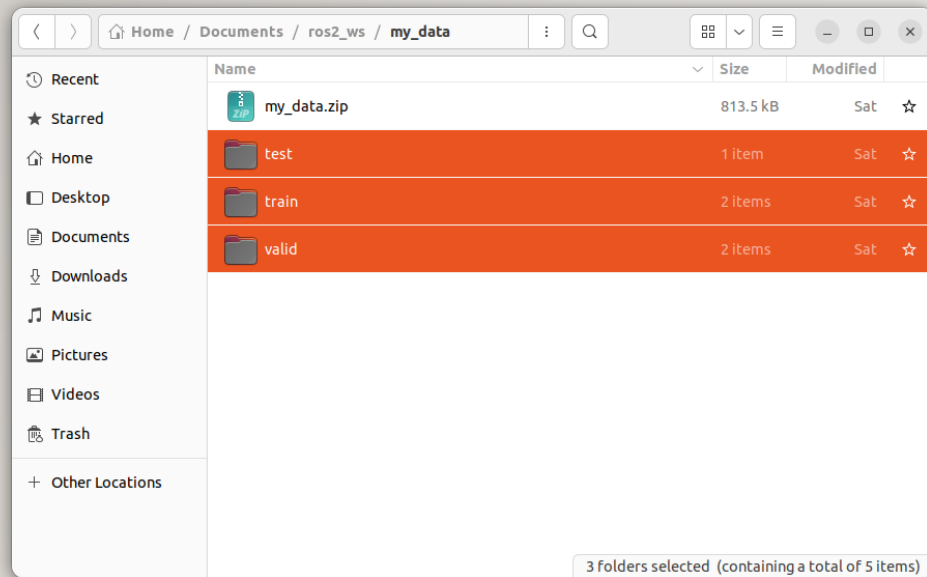
▼ day2

• [\\_\\_init\\_\\_.py](#)  
• [2\\_1\\_a\\_capture\\_wc\\_image.py](#)  
• [2\\_1\\_b\\_cont\\_capture\\_wc\\_image.py](#)  
• [2\\_1\\_c\\_capture\\_wc\\_thread.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 FOR DETECTION ALERT

---



# CODING HINTS

---

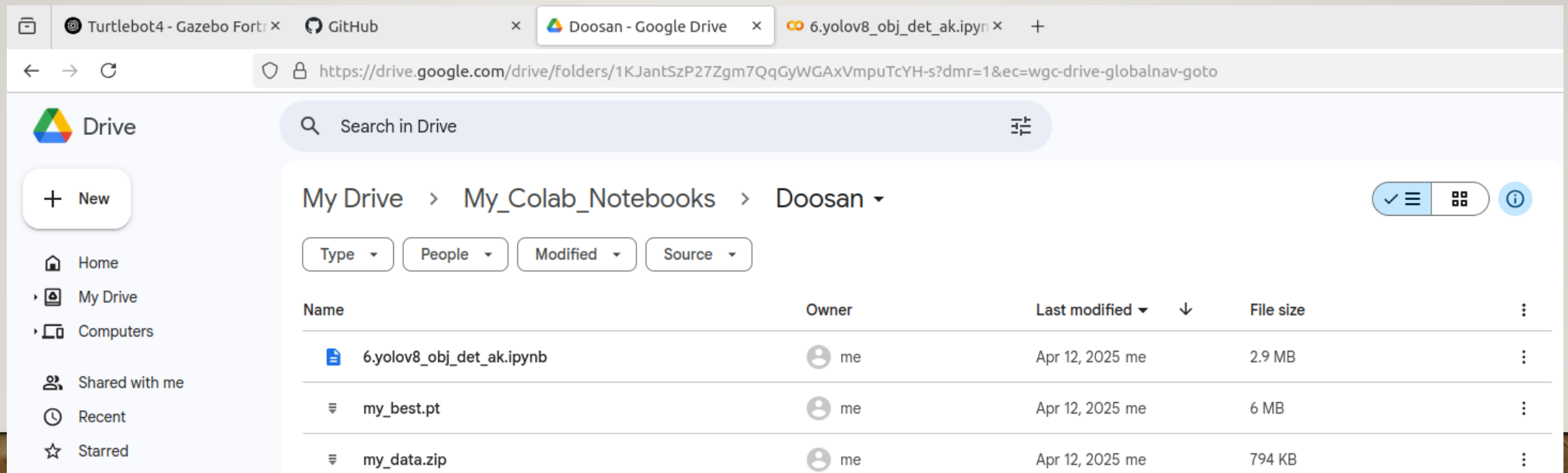
- Image Capture
- Data Labelling
- Preprocessing
- Yolo8 Object Det



```
▼ day2
  + __init__.py
  + 2_1_a_capture_wc_image.py
  + 2_1_b_cont_capture_wc_image.py
  + 2_1_c_capture_wc_thread.py
  + 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
  + 2_4_d_yolov8_obj_det_wc.py
```

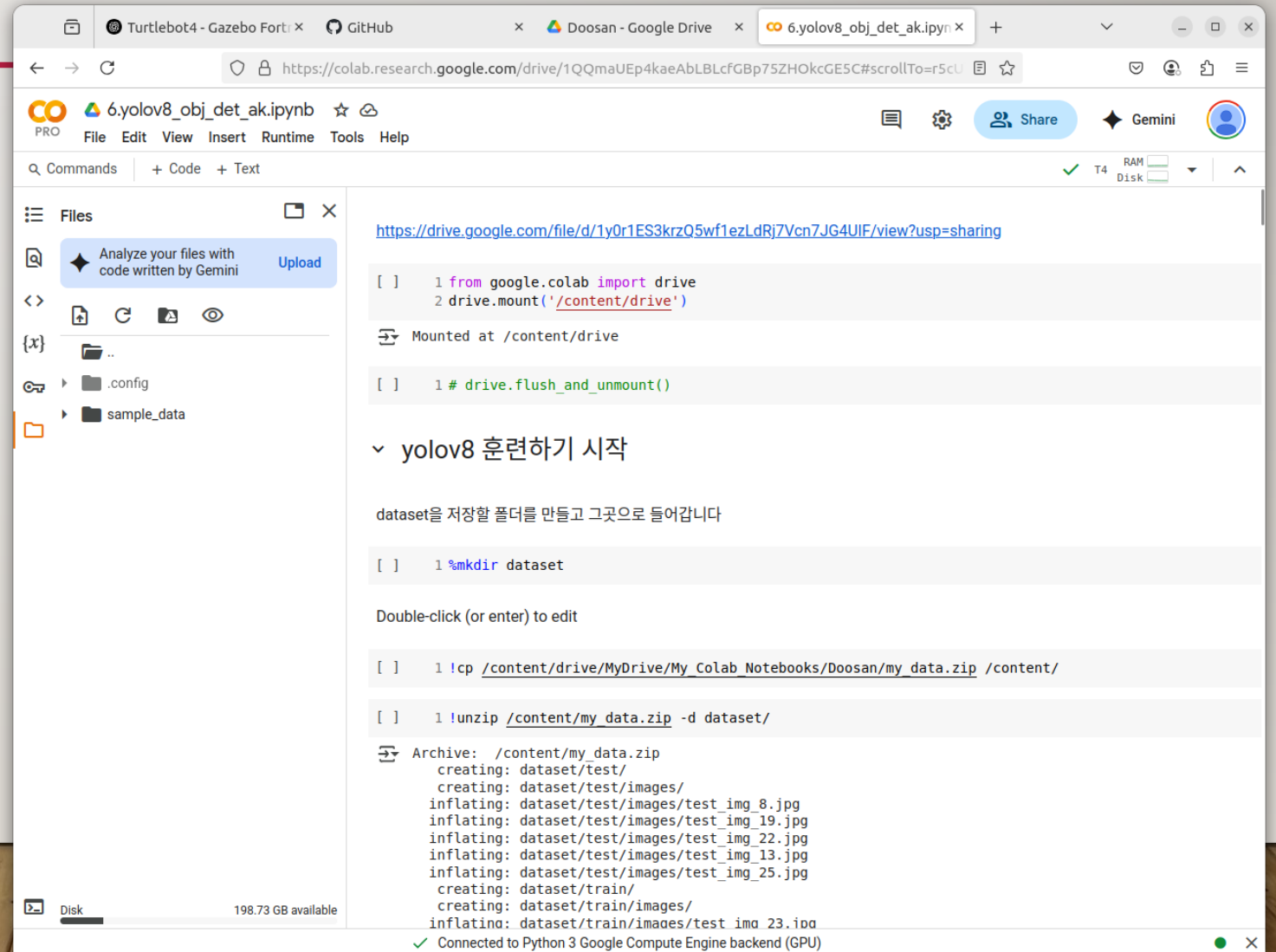
# USING GOOGLE COLLAB.TO CREATE CUSTOM MODEL

- 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)



# SETTING UP YOLO

---

\$ pip install --upgrade pip

\$ nvidia-smi # Check if the GPU is available

# If you have a GPU, install the CUDA version of PyTorch

# \$ pip install torch torchvision torchaudio --extra-index-url <https://download.pytorch.org/whl/cu124>

\$ pip install ultralytics

\$ pip install "numpy<1.25.0"

# REQUIRED PACKAGES SETUP

---

\$ pip list | grep opencv

If doesn't exist....

\$ pip3 install opencv-python

\$ pip3 install opencv-contrib-python

\$ pip list | grep ultra

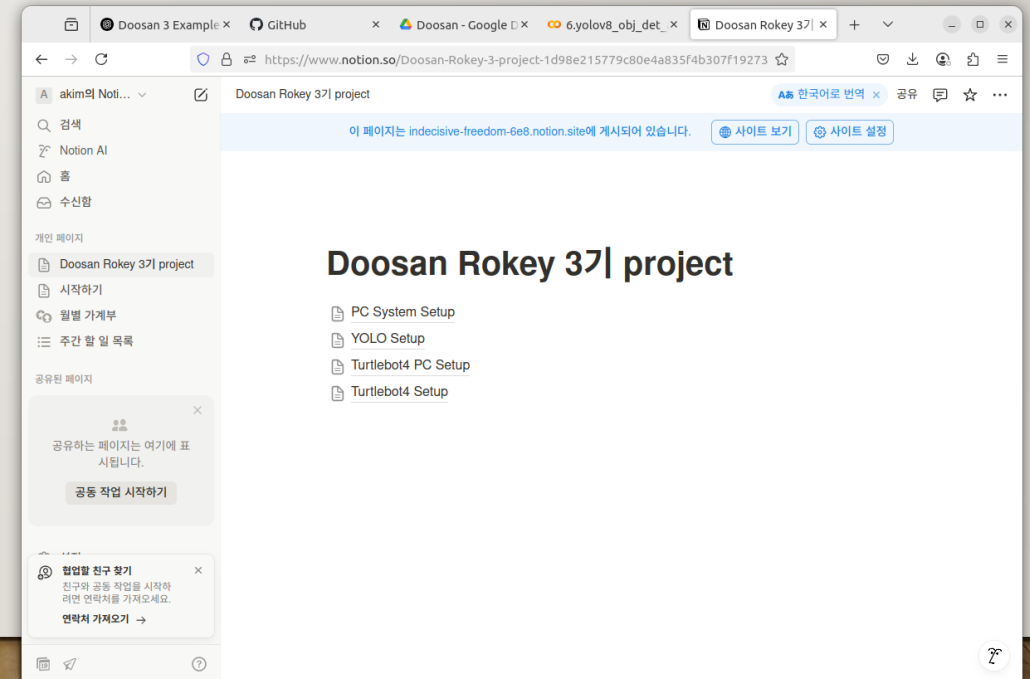
If doesn't exist....

\$ pip install ultralytics

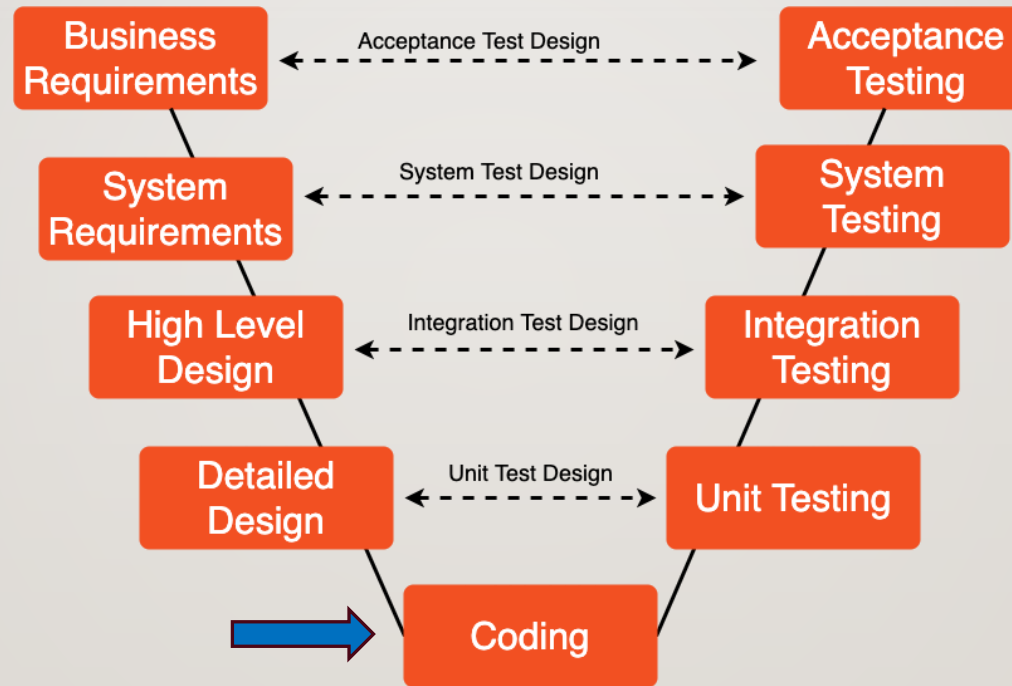


# INSTALLING YOLO

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



# SPRINT I - DETECTION ALERT



SDLC - V Model - notepub.io



HOW DID YOU/DO YOU NEED TO  
DEFINE A DETECTION AREA?

---



# CODING HINTS

- Image Capture
- Data Labelling
- Preprocessing
- Yolo8 Object Det
- Create your detection/alert condition



```
▼ day2
  🔗 __init__.py
  🔗 2_1_a_capture_wc_image.py
  🔗 2_1_b_cont_capture_wc_image.py
  🔗 2_1_c_capture_wc_thread.py
  🔗 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
  🔗 2_4_d_yolov8_obj_det_wc.py
  🔗 2_5_a_Draw_Box_wc.py
  🔗 2_5_b_Draw_Polygon_wc.py
  🔗 2_5_c_Security_Alert_wc.py
```

# PORTING TO ROS

---



# 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
```

# SETUP BASH

---

- `ROS_DOMAIN_ID = 0` in `.bashrc`
- `source ~/.bashrc`
- `ros2 daemon stop; ros2 daemon start`
- Make sure `discovery setup.bash` is not sourced!



# CODING HINTS

---

create your team working folder and place  
files and work from here

```
$ mkdir  
~/rokey3_<grp_letter><grp_num>_ws
```

(i.e. mkdir ~/rokey2\_A2\_ws)

# ROS HINTS

---

```
▼ day2
  🔗 __init__.py
  🔗 2_1_a_capture_wc_image.py
  🔗 2_1_b_cont_capture_wc_image.py
  🔗 2_1_c_capture_wc_thread.py
  🔗 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
  🔗 2_4_d_yolov8_obj_det_wc.py
  🔗 2_5_a_Draw_Box_wc.py
  🔗 2_5_b_Draw_Polygon_wc.py
  🔗 2_5_c_Security_Alert_wc.py
  🔗 2_6_a_image_publisher.py
  🔗 2_6_b_image_subscriber.py
  🔗 2_6_c_data_publisher.py
  🔗 2_6_d_data_subscriber.py
  🔗 2_6_e_yolo_publisher.py
  🔗 2_6_f_yolo_subscriber.py
  🔗 bus.ino
```

```
🔗 2_6_a_image_publisher.py
🔗 2_6_b_image_subscriber.py
🔗 2_6_c_data_publisher.py
🔗 2_6_d_data_subscriber.py
🔗 2_6_e_yolo_publisher.py
🔗 2_6_f_yolo_subscriber.py
🔗 bus.ino
```

# 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

```
entry_points={  
    'console_scripts': [  
        'pub_image = day2.2_6_a_image_publisher:main',  
        'show_image = day2.2_6_b_image_subscriber:main',  
        'pub_data = day2.2_6_c_data_publisher:main',  
        'show_data = day2.2_6_d_data_subscriber:main',  
        'pub_yolo = day2.2_6_e_yolo_publisher:main',  
        'show_yolo = day2.2_6_f_yolo_subscriber:main',  
    ],  
}
```

# ROS HINTS

---

\$ cd ~/rokey3\_A2\_ws

\$ sudo apt update

\$ source

\$ sudo apt install terminator

~/rokey1\_A2\_ws/install/setup.bash

\$ ros2 run <package\_name>

<command\_name>

```
1998  ros2 run day2 show_yolo
```

```
1999  ros2 run day2 pub_yolo
```

```
2000  1
```



# ROS HINTS

---

```
2_6_a_image_publisher.py
2_6_b_image_subscriber.py
2_6_c_data_publisher.py
2_6_d_data_subscriber.py
2_6_e_yolo_publisher.py
2_6_f_yolo_subscriber.py
bus.ino
```

```
$ 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>
```

# TEAM EXERCISE 5

---

Perform coding and testing of Detection Alert Module

# RESULTS & CODE REVIEW BY EACH TEAM

---

Show actual results against the expected results and explain the code written

# PROJECT SPRINTS

- 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

- System Monitor

- Receive and Display Detection Camera and info
- Receive and Display AMR Camera and info
- Store, display, and report Information and Alerts



# RUNNING OBJ. DET ON A REAL ROBOT

---



# AMR INTRODUCTION

---

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

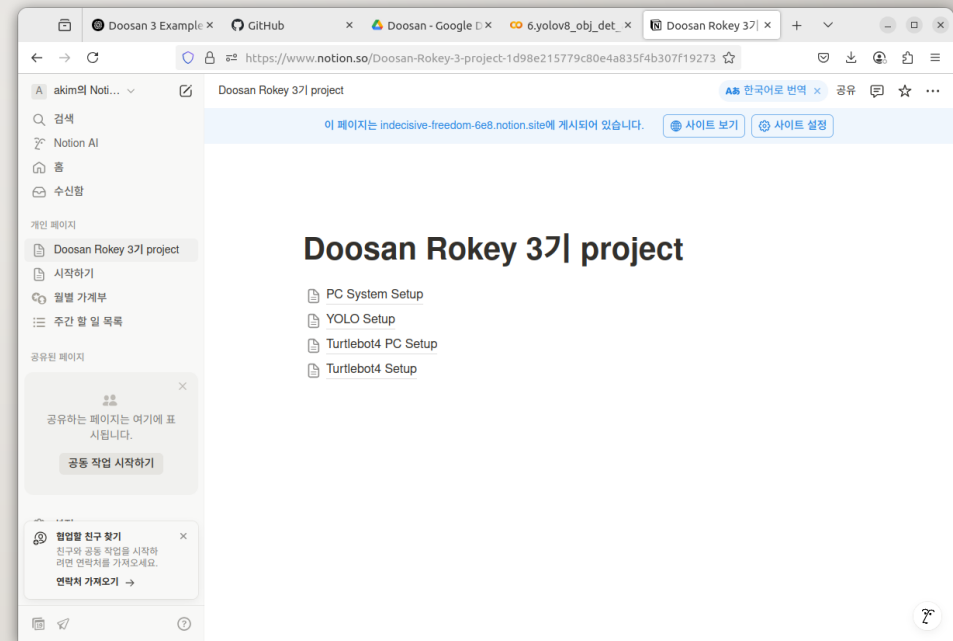




# SETUP PC FOR AMR

## Turtlebot4 PC Setup

<https://indecisive-freedom-6e8.notion.site/Turtlebot4-PC-Setup-1d98e215779c80e887b6c1dff4f151be>





# POWER ON AND OFF AMR

---

## 로봇 전원 켜기

- TurtleBot 4를 도크에 놓기
- 도크의 녹색 LED가 몇 초 동안 켜지고 TurtleBot 4가 켜집니다. 로봇이 부팅될 때까지 잠시 기다리세요.
- 로봇 버튼과 표시등에 대한 자세한 내용은 [Create®3 설명서를 참조하세요.](#)

## POWER OFF

- Turtlebot4 를 도크에서 분리 하기
- 신호음이 들릴 때까지 Create3의 전원 버튼을 5초 동안 길게 누릅니다.
- 전원 버튼을 놓습니다.

# CONNECTING TO AMR -- SSH

---

Connect your PC to WiFi router that your AMR is connected

Ex: RokeyAP\_Test

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
```

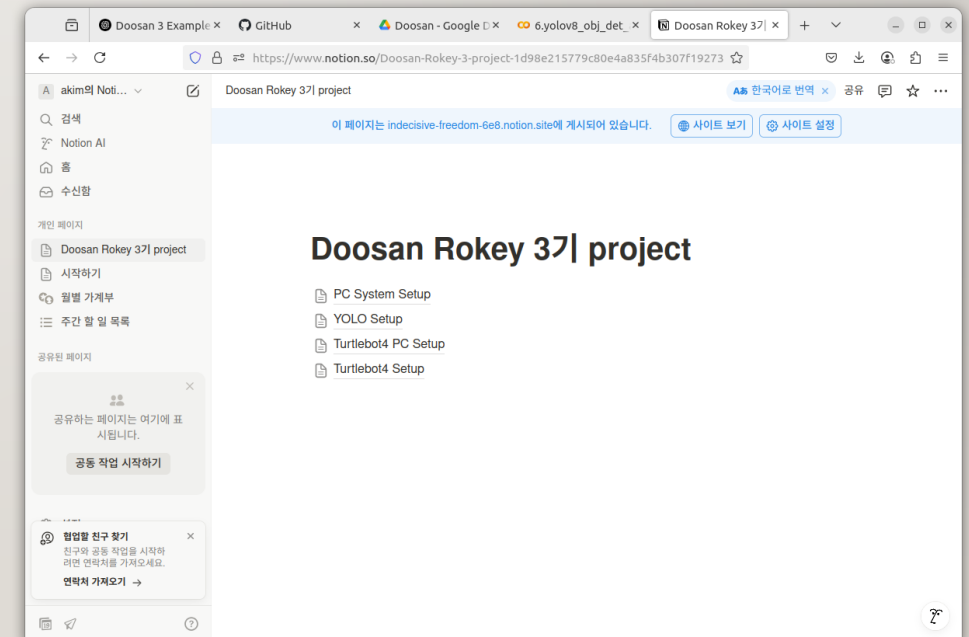
PW: rokey1234

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

# HOW TO SETUP AMR

## Turtlebot4 Setup

<https://indecisive-freedom-6e8.notion.site/Turtlebot4-Setup-1d98e215779c801b8e74eff67eb6305e>



# HOW TO TEST PC – AMR CONNECTION

---

\$ 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

# OBJECT TRACKING WITH AMR CAMERA

---





# CODING HINTS

---

- Image Capture

```
▼ day3
  + __init__.py
  + 3_1_a_capture_image.py
  + 3_1_b_cont_capture_image.py
```

# CODING HINTS

---

- Image Capture

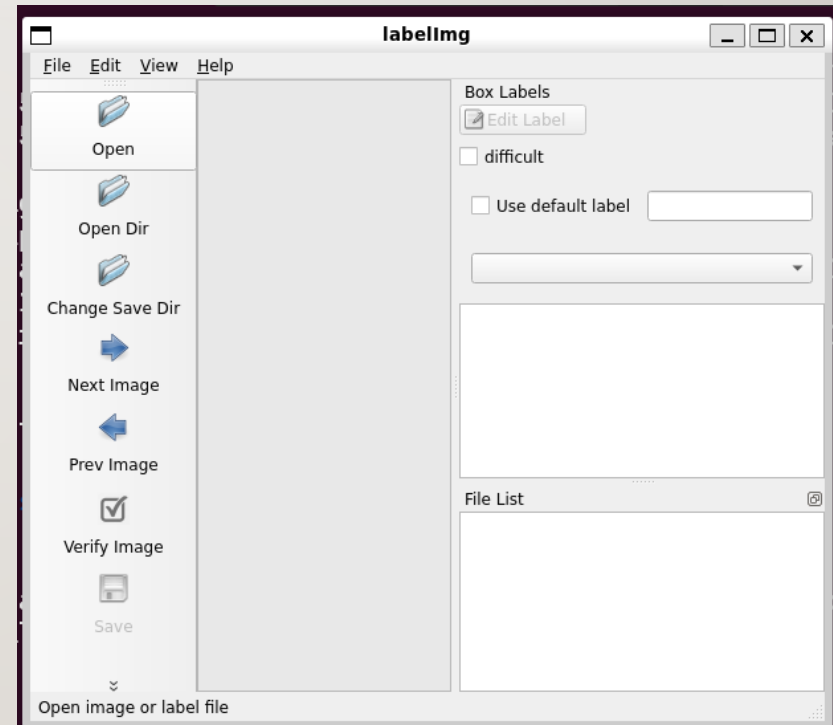
```
▼ day3
  + __init__.py
  + 3_1_a_capture_image.py
  + 3_1_b_cont_capture_image.py
```

- Data Labelling
  - labellmg

# CODING HINTS

---

- Data Labelling : use previously installed Labellmg
- Or
- pip3 install labellmg
  - May also need “pip3 install PyQt5 lxml”
- labellmg



# 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

▼ day2

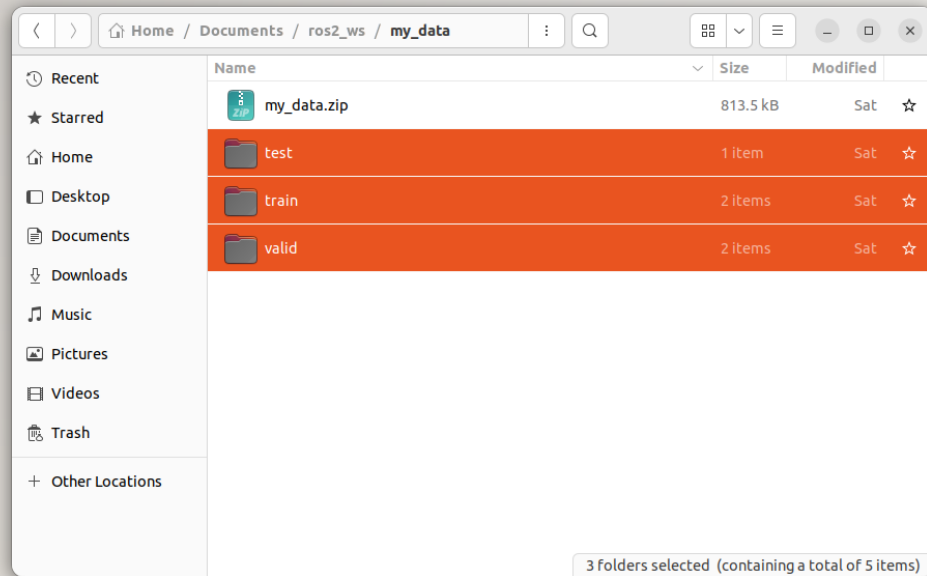


- `__init__.py`
- `2_1_a_capture_wc_image.py`
- `2_1_b_cont_capture_wc_image.py`
- `2_1_c_capture_wc_thread.py`
- `2_3_a_create_data_dirs.py`
- `2_3_b_move_image.py`
- `2_3_c_move_labels.py`

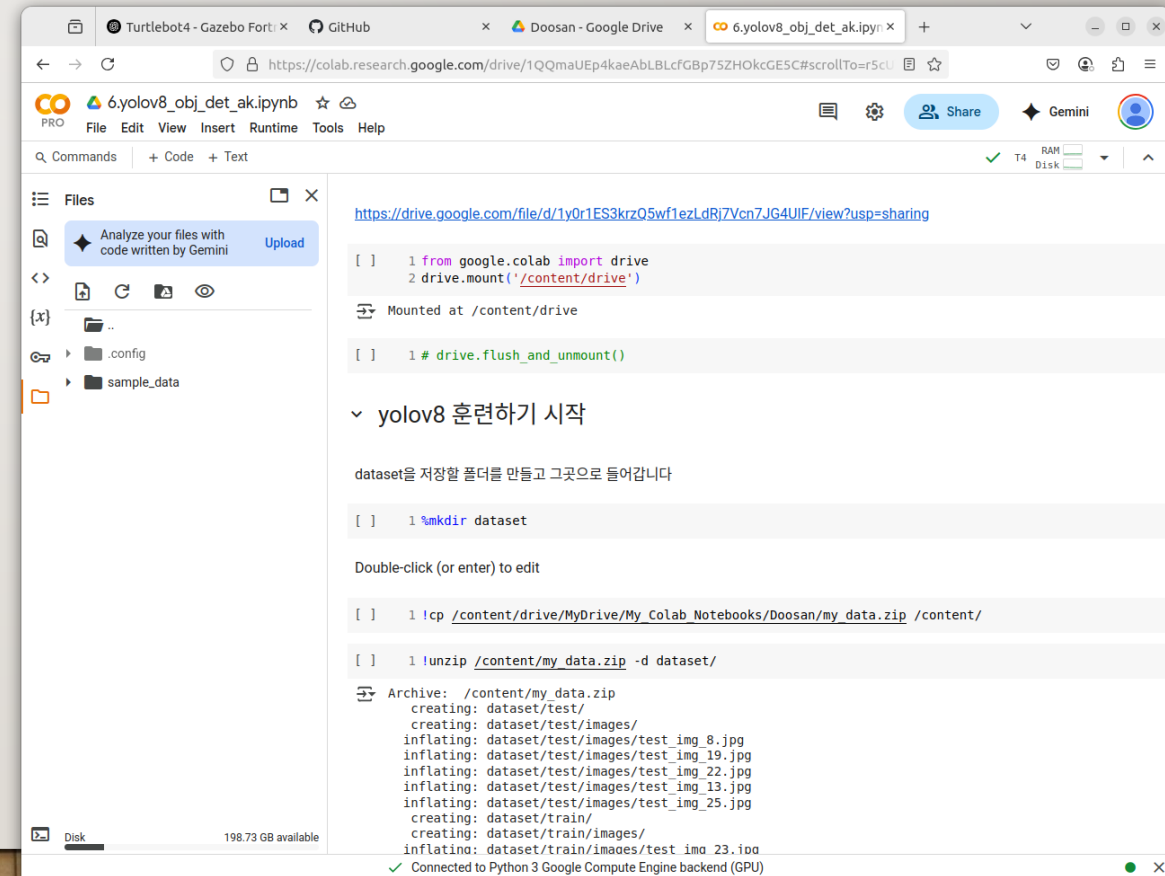
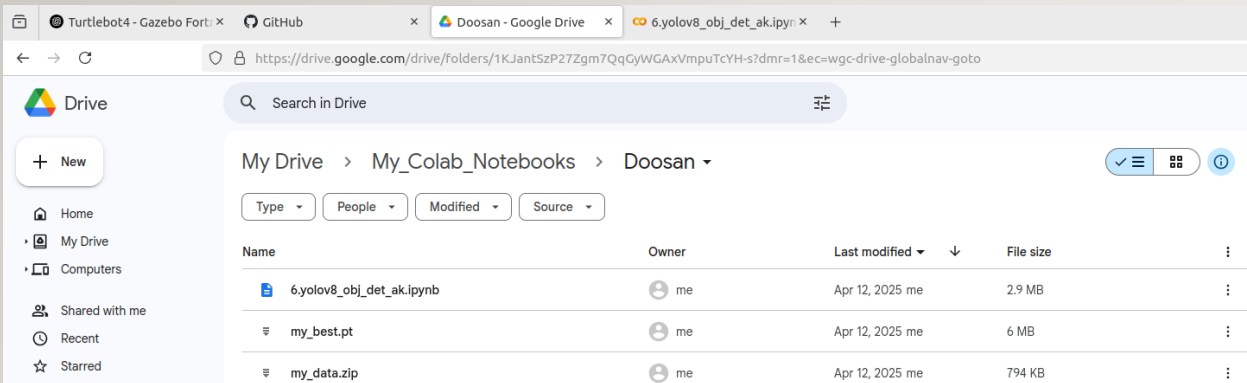


# ZIP TRAIN DATA SET

---



# USING GOOGLE COLLAB.TO CREATE CUSTOM MODEL



# PERFORM YOLO TRAINING & INFERENCE FOR AMR CONTROLLER

---



# UNDOCK/DOCK 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 “{”

# CODING HINTS

---

- Image Capture
- Data Labelling
- Preprocessing
- Yolo8 Object Det



```
▼ day3
  + __init__.py
  + 3_1_a_capture_image.py
  + 3_1_b_cont_capture_image.py
  + 3_4_a_yolov8_obj_det.py
  + 3_4_b_yolov8_obj_det_thread.py
  + 3_4_c_yolov8_obj_det_track.py
```



# RUNNING IN SIMULATION

---



# OPERATING A ROBOT(SIM)

---

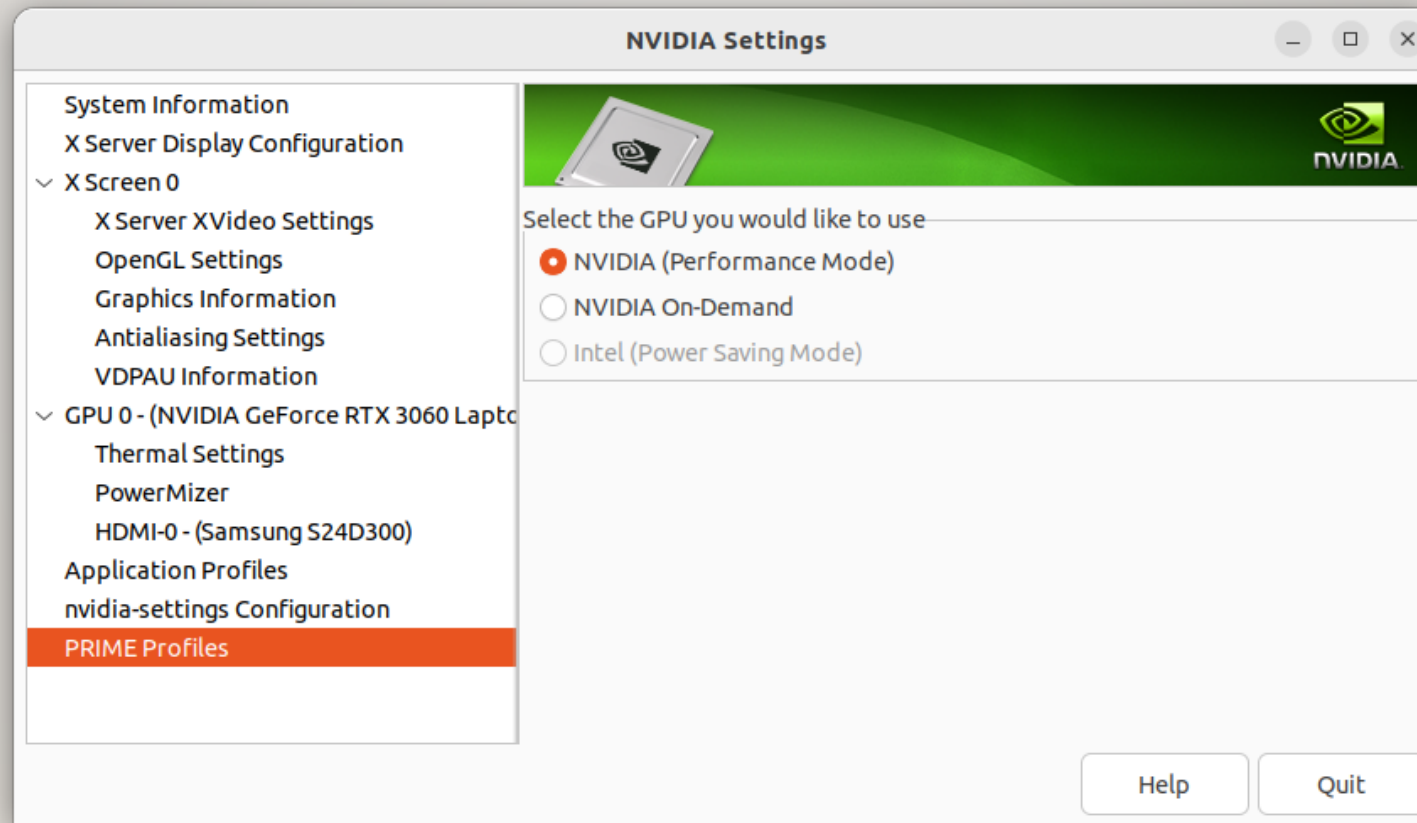
## TERM1

- `ros2 launch turtlebot4_ignition_bringup turtlebot4_ignition.launch.py rviz:=true`

## TERM2

- `ros2 topic list`
- `ros2 topic echo <topic> --once`
  - `/oakd/rgb/preview/image_raw`
  - `/oakd/rgb/preview/depth`
  - ....

# SETUP NVIDIA GPU FOR SLAM



# DIGITAL MAPPING USING SLAM (SIM)

---

## TERM1

- `ros2 launch turtlebot4_ignition_bringup  
turtlebot4_ignition.launch.py nav2:=true slam:=true rviz:=true`

## TERM2 (SAVE MAP AFTER MAPPING FINISHES)

- `ros2 service call /slam_toolbox/save_map slam_toolbox/srv/SaveMap "name: data: 'map_name'"`

Ex:: `ros2 service call /slam_toolbox/save_map slam_toolbox/srv/SaveMap "name: data: 'my_map'"`

# DIGITAL MAPPING WITH AUTO – SLAM (SIM)

---

## TERM1

- `ros2 launch turtlebot4_ignition_bringup turtlebot4_ignition.launch.py`  
`nav2:=true slam:=true rviz:=true`
- Undock the robot

## TERM2

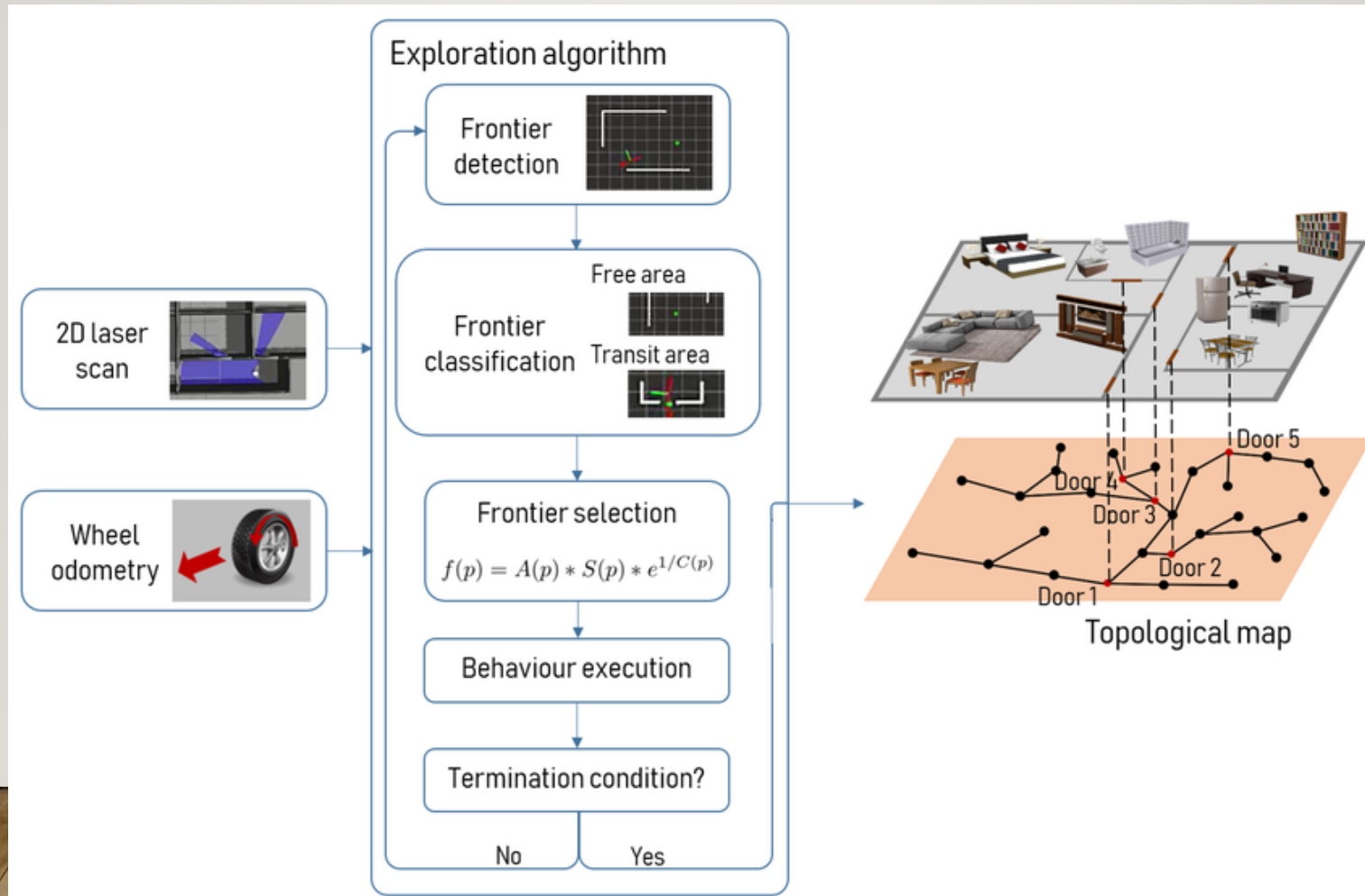
- `ros2 launch explore_lite explore.launch.py`

## TERM3 (SAVE MAP AFTER MAPPING FINISHES)

- `ros2 service call /slam_toolbox/save_map slam_toolbox/srv/SaveMap "name: data: 'map_name'"`  
Ex.: `ros2 service call /slam_toolbox/save_map slam_toolbox/srv/SaveMap "name: data: 'my_map'"`



# AUTO SLAM CONCEPT/ALGORITHM



# ALGORITHM DETAIL

- Map Subscription

explore\_lite subscribes to the SLAM-generated occupancy grid (/map topic) and identifies:

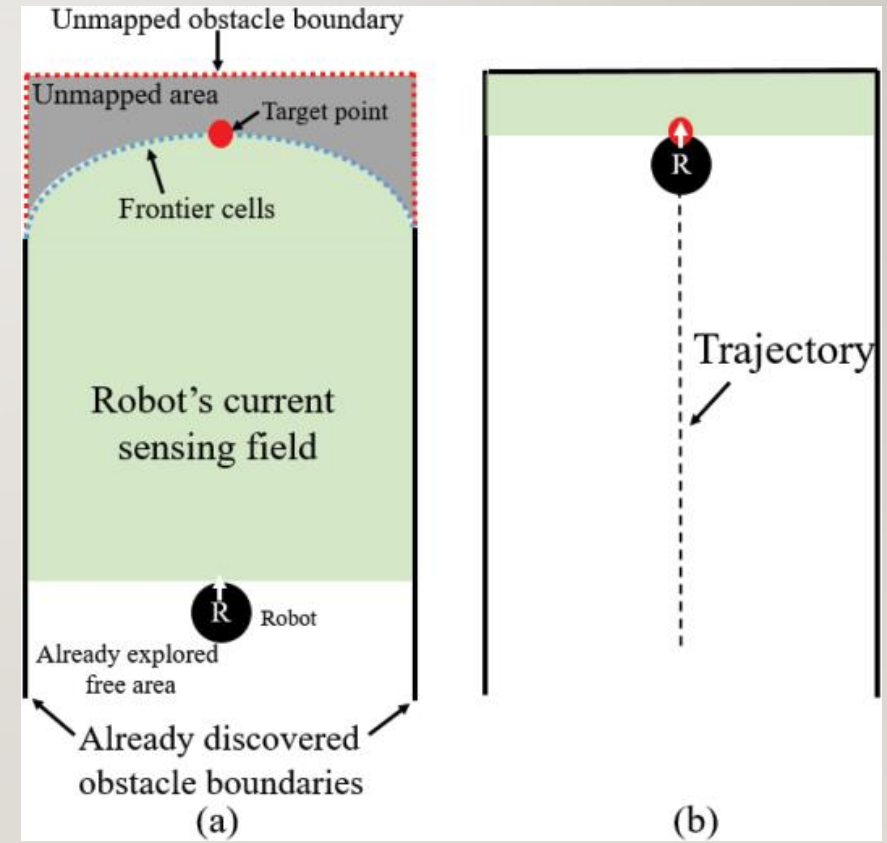
- Free space: known, unoccupied areas
- Occupied space: obstacles
- Unknown space: unexplored

- Frontier Detection

The map is scanned for cells that:

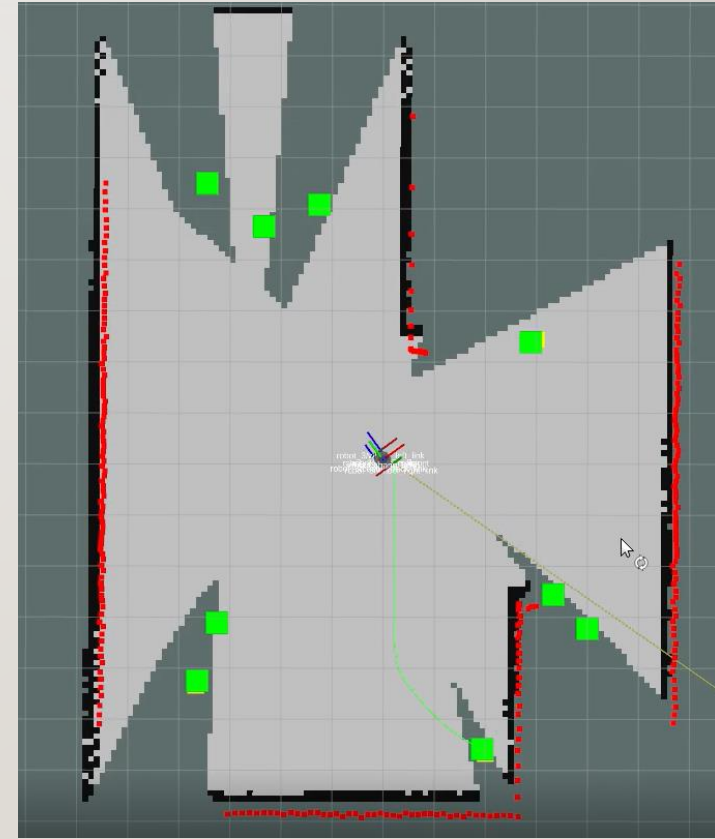
- Are free, and
- Are adjacent to at least one unknown cell.

These are marked as frontier cells.



# ALGORITHM DETAIL

- Frontier Grouping
  - Frontier cells are clustered into connected regions.
  - Each group represents a potential exploration target.
- Goal Selection
  - For each frontier group, a representative point (typically the centroid or closest point) is selected.
  - The robot scores each group based on:
    - Distance from the robot
    - Information gain (how much new area might be revealed)
  - The best-scoring frontier is chosen as the next goal.



# ALGORITHM DETAIL

---

- Termination

- While (frontiers exist and reachable)

- Select best frontier

- Send as goal

- If goal fails → blacklist

- If (no frontiers or all blacklisted)

- Terminate exploration



# CHECKING DIGITAL MAP (SIM)

---

## CHECK IF CORRECT

\$ xdg-open <map-path>/map.pgm

Or,

\$ eog <map-path>/map.pgm



# NAVIGATION W/ DIGITAL MAP (SIM)

---

## TERMI

- `ros2 launch turtlebot4_ignition_bringup turtlebot4_ignition.launch.py nav2:=true  
slam:=false localization:=true rviz:=true`

# TUTORIAL(SIM)

---

- [TurtleBot 4 Navigator · User Manual](https://turtlebot.github.io/turtlebot4-user-manual/tutorials/turtlebot4_navigator.html)

[https://turtlebot.github.io/turtlebot4-user-manual/tutorials/turtlebot4\\_navigator.html](https://turtlebot.github.io/turtlebot4-user-manual/tutorials/turtlebot4_navigator.html)

- Terminal 1

```
$ ros2 launch turtlebot4_ignition_bringup  
  turtlebot4_ignition.launch.py nav2:=true  
  slam:=false localization:=true rviz:=true
```

- Terminal 2

```
$ ros2 run turtlebot4_python_tutorials nav_to_pose  
  
$ ros2 run turtlebot4_python_tutorials  
  nav_through_poses  
  
$ ros2 run turtlebot4_python_tutorials  
  follow_waypoints  
  
$ ros2 run turtlebot4_python_tutorials create_path  
  
$ ros2 run turtlebot4_python_tutorials  
  mail_delivery  
  
$ ros2 run turtlebot4_python_tutorials patrol_loop
```

# USING DEPTH (SIM)

---

```
▼ day3
  ▼ day3
    > __pycache__
    • __init__.py
    • 3_1_a_capture_image.py
    • 3_1_b_cont_capture_image.py
    • 3_4_a_yolov8_obj_det.py
    • 3_4_b_yolov8_obj_det_thread.py
    • 3_4_c_yolov8_obj_det_track.py
    • 3_5_a_depth_checker.py
    • 3_5_b_depth_to_3d.py
    • 3_5_c_depth_to_nav_goal.py
    • 3_5_d_nav_to_person.py
```

```
• 3_5_a_depth_checker.py
• 3_5_b_depth_to_3d.py
• 3_5_c_depth_to_nav_goal.py
• 3_5_d_nav_to_person.py
```

# 프로젝트 RULE NUMBER ONE!!!

---

Are we still having  
**FUN!**

