GOOD MORNING!
早上好!
안녕하세요!

DAY 4

DAY 1

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

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

- 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

DAY 4 (MINI PROJECT)

- System(High Level) Design (Mini Project)
  - System Architectural Diagram
- Detail Design to Acceptance - Agile Development (SPRINTs)
  - Detection
  - AMR Control

DAY 4 (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 5 (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 5 (FINAL PROJECT)

- 비즈니스/System 요구 사항 업데이트
- 시스템 설계 및 프로세스 정립
  - System(High Level) Design
  - Schedule/Time Management
- 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

프로젝트 RULE NUMBER ONE!!!

Have Fun Fun Fun!

EXERCISE : RUN TUTORIAL ON REAL ROBOT

Make copy and Update the ***simulation*** tutorial code provided to successfully execute in the project environment with ***real robot***

Tutorial Codes are found in:
$HOME/turtlebot4_ws/src/turtlebot4_tutorials/turtlebot4_python_tutorials/turtlebot4_python_tutorials

DEPTH/TRANSFORM EXERCISE

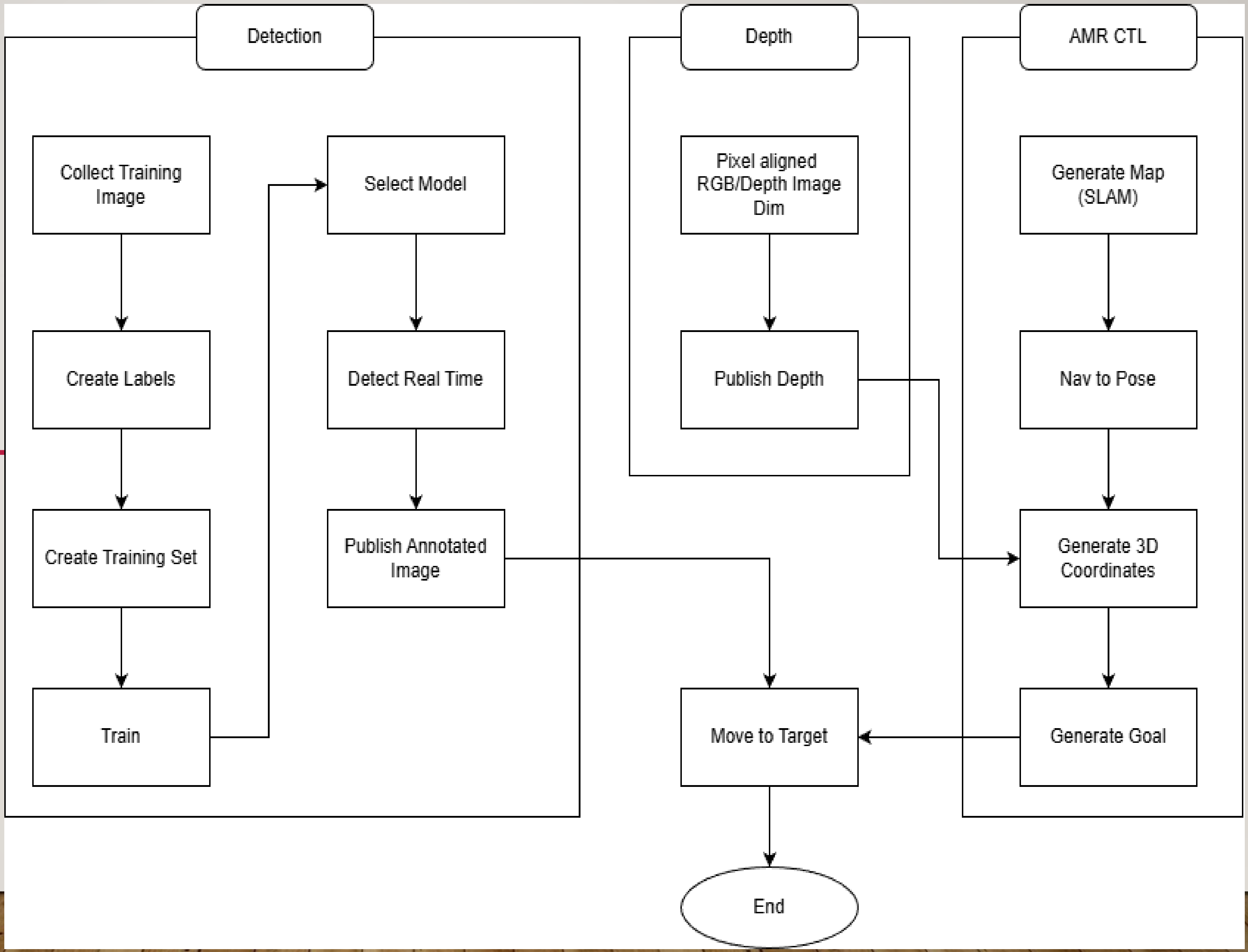
Make copy and Update the ***simulation*** depth code provided to successfully execute in the project environment with ***real robot***

Simulation Depth Codes are found on **Github**

HOMEWORK CHECK

- Create an AMR control code
  - AMR receives an event and undocks
  - Init Pose is set
  - AMR moves to a goal position
  - AMR able to approach a target
    - **Design a way to get information about the target**
    - **Design an approach algorithm**
    - **Test**
- Update System Requirement

TASKS TO BE PERFORMED



TEAM EXERCISE 2-3

Brainstorm Updated Mini Project System Requirement for the project and document

Using the posted notes and flipchart as needed

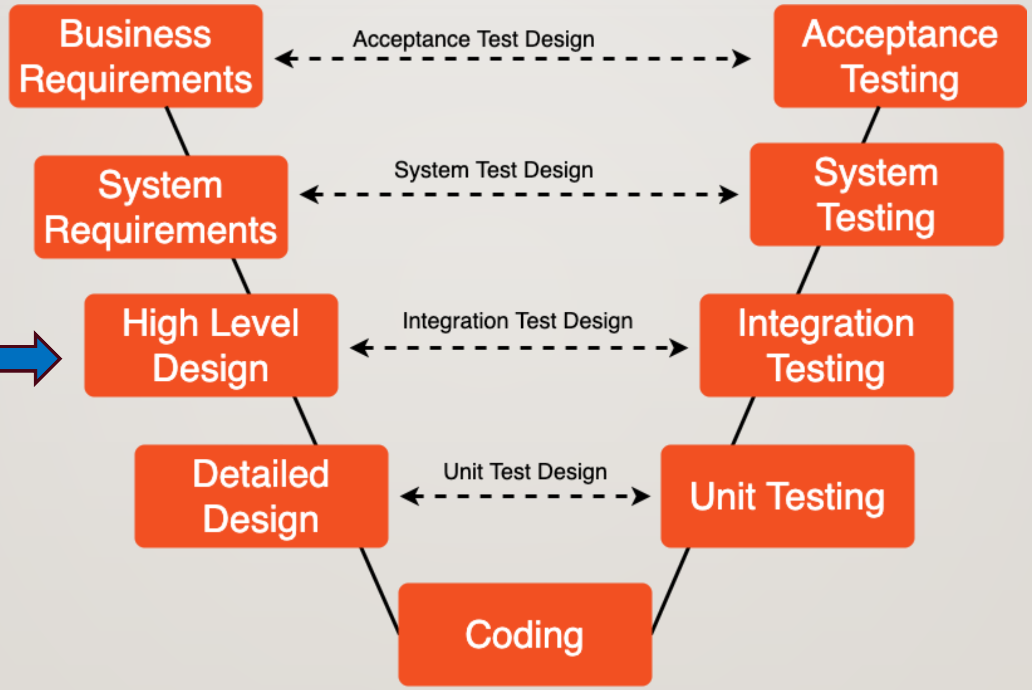
Include where, when, what will be used

SYSTEM REQUIREMENT PRESENTATION BY EACH TEAM (SELECTED)

Using the posted notes and flipchart as needed

SW DEVELOPMENT PROCESS

HOW:
Where, What,
When, How Much



SDLC - V Model - notepub.io

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.

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

TEAM EXERCISE 3

Create 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

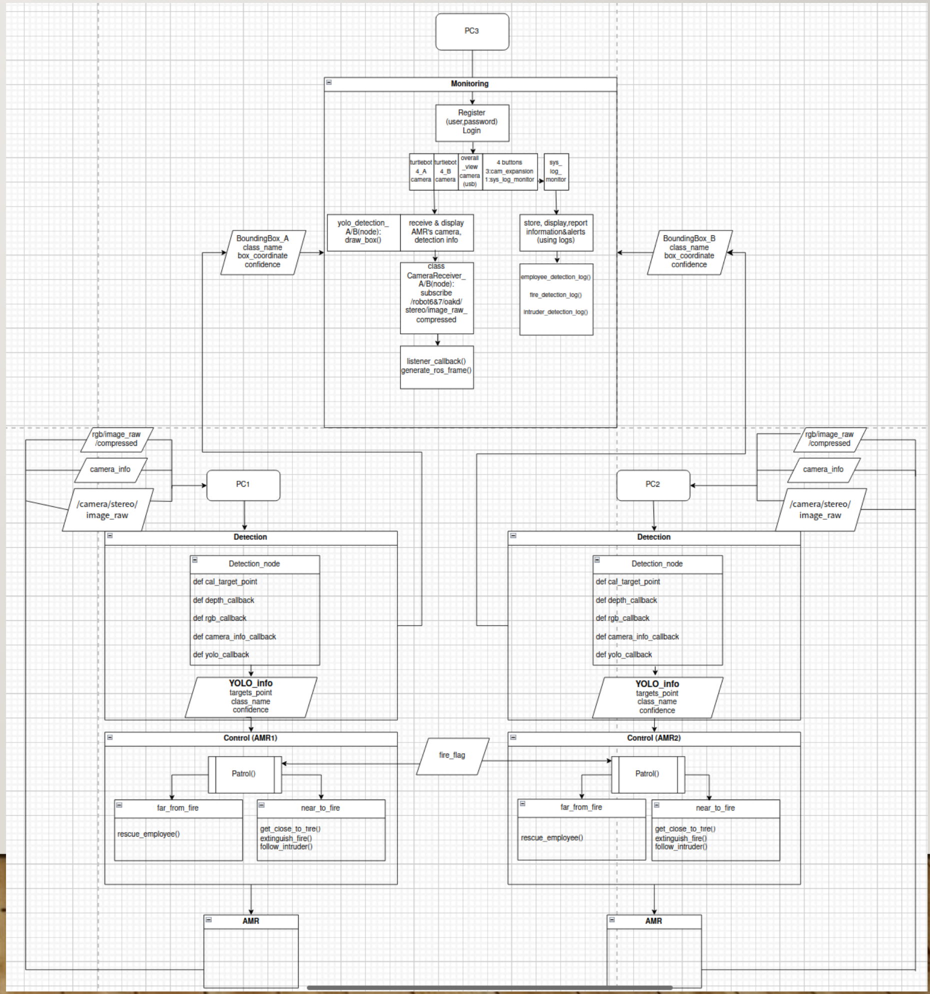
  AMR Controller

- Functions
- Interfaces

  Dataflow

- Testing

  Integration Testing



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 AMR 이 독립적으로 작동할 수 있도록 필요한 하드웨어 및 소프트웨어 구성 요소로 구성되며, 중앙 서버 없이 데이터를 현장에서 처리합니다.

---

2. 시스템 아키텍처

이 시스템은 단일 AMR 으로 구성되므로 데이터 처리, 네비게이션, 위협 탐지 및 경고가 모두 AMR 에서 로컬로 수행됩니다. AMR 은 모니터링, 알림 및 수동 제어를 위해 PC 의 사용자 인터페이스와 로컬 네트워크(Wi-Fi)를 통해 직접 통신합니다.

PROJECT TIMELINE/CRITICAL PATH ITEM MANAGEMENT

EX. IMPLEMENTATION TIMELINE

Function Backlog	Owner	5월 20일	5월 21일	5월 22일	5월 23일	5월 24일	5월 25일
Unloading Module	John						
Input1	John						
Input2	John						
Output 1	John						
Unit Test	John						
Receiving Module	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, 어떻 게를 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 activitie

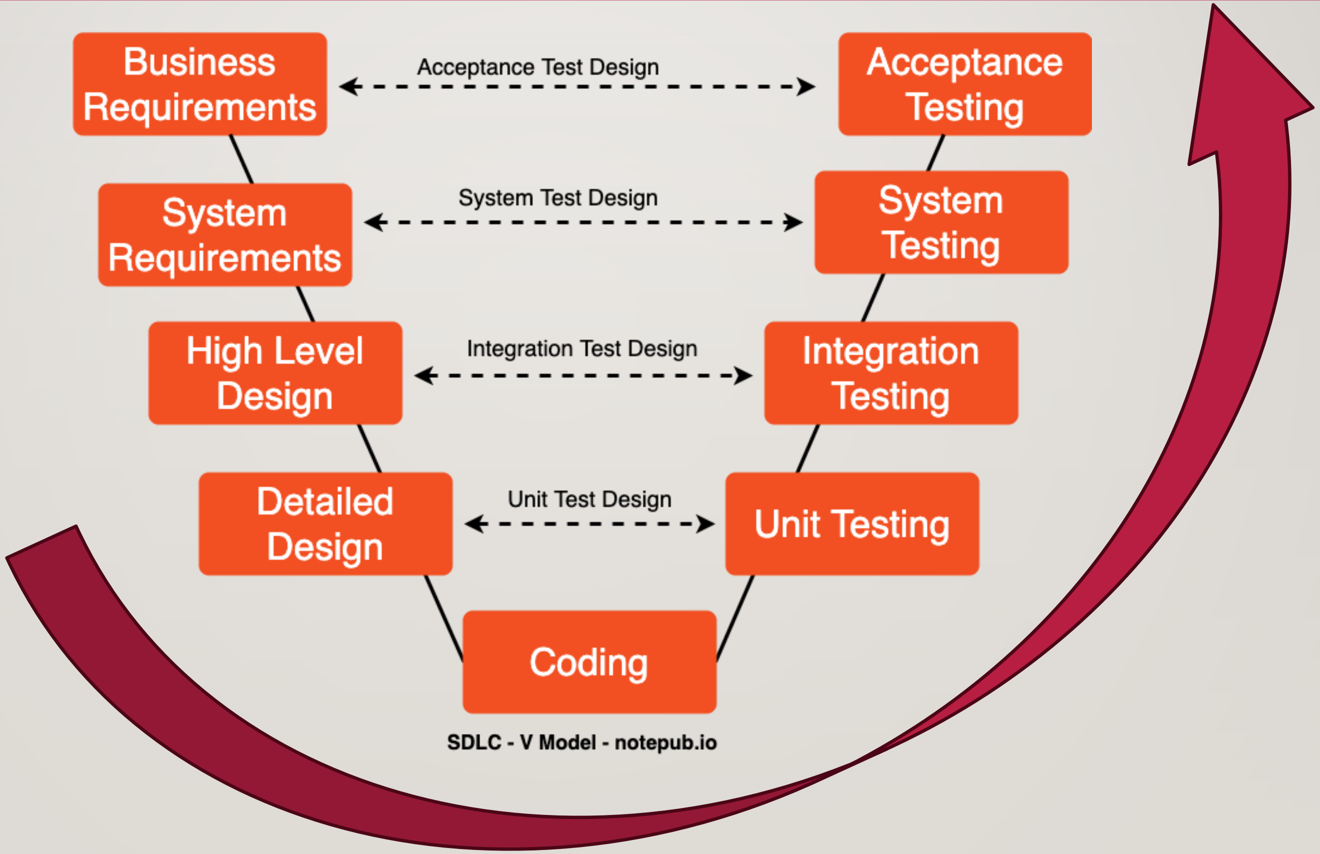
- 프로젝트 타임라인에 직접적인 영향을 주는 작업입니다. 이러한 작업이 지연되면 종속 활동이 가장 길어지기 때문에 프로젝트의 전체 완료가 지연됩니다

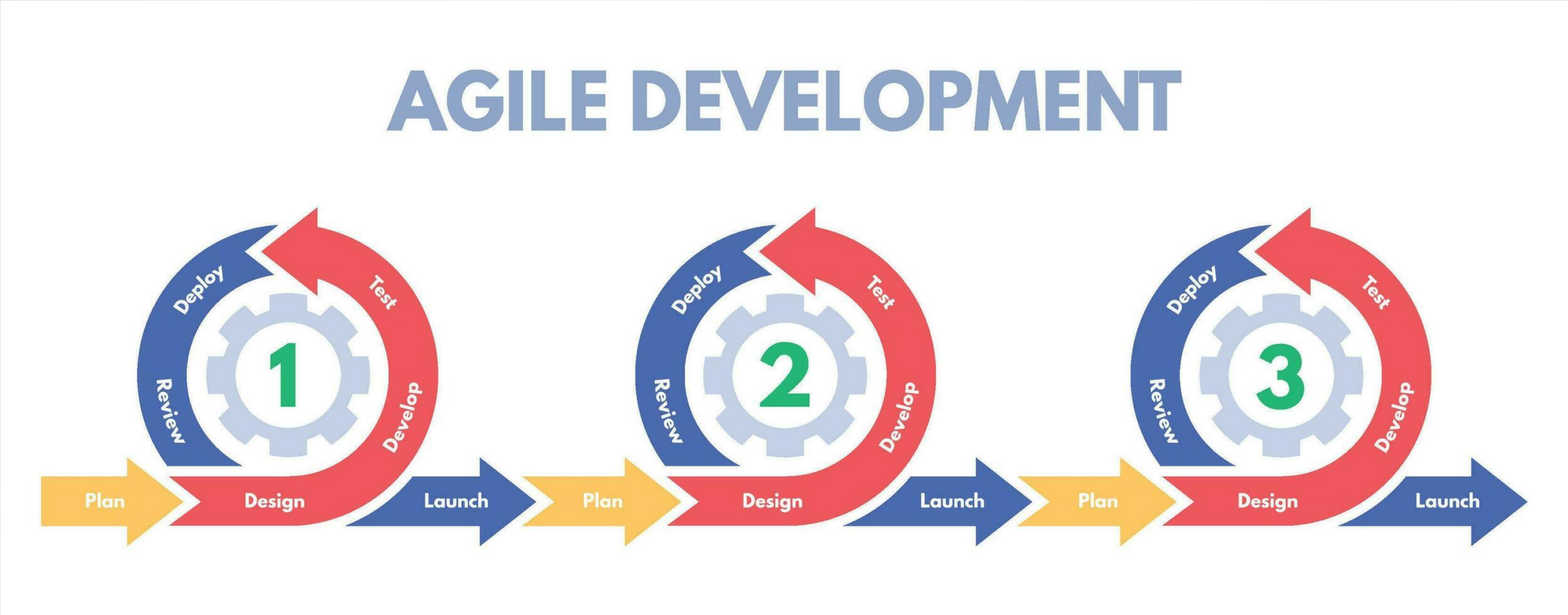
GUIDE TO PROGRESS INDICATORS

- 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)

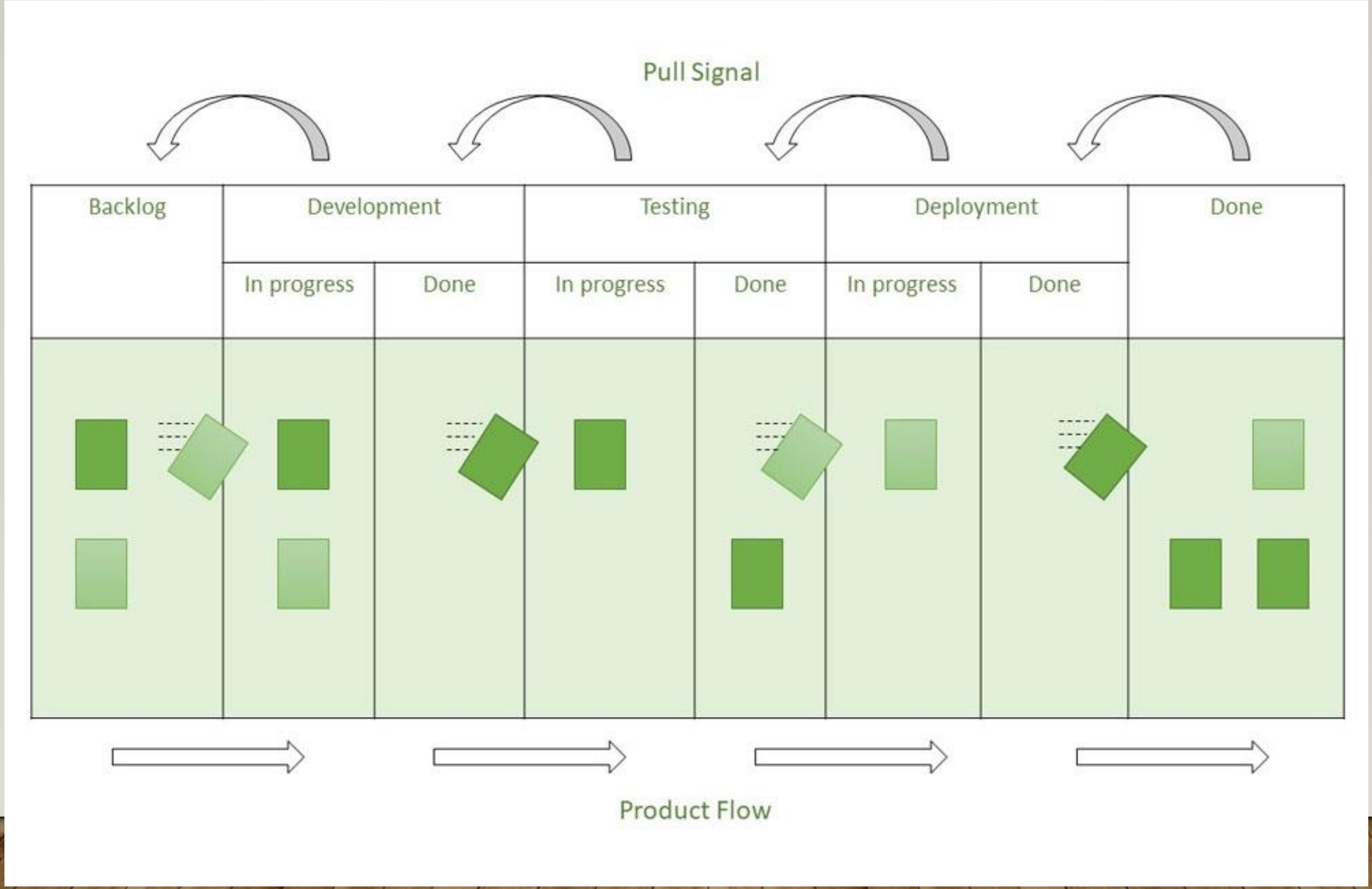
DETAIL DESIGN TO USER ACCEPTANCE

SW DEVELOPMENT PROCESS




AGILE DEVELOPMENT
Deploy
Test
1
Review
Develop
Plan
Design
Launch
Deploy
Test
2
Review
Develop
Plan
Design
Launch
Deploy
Test
3
Review
Develop
Plan
Design
Launch

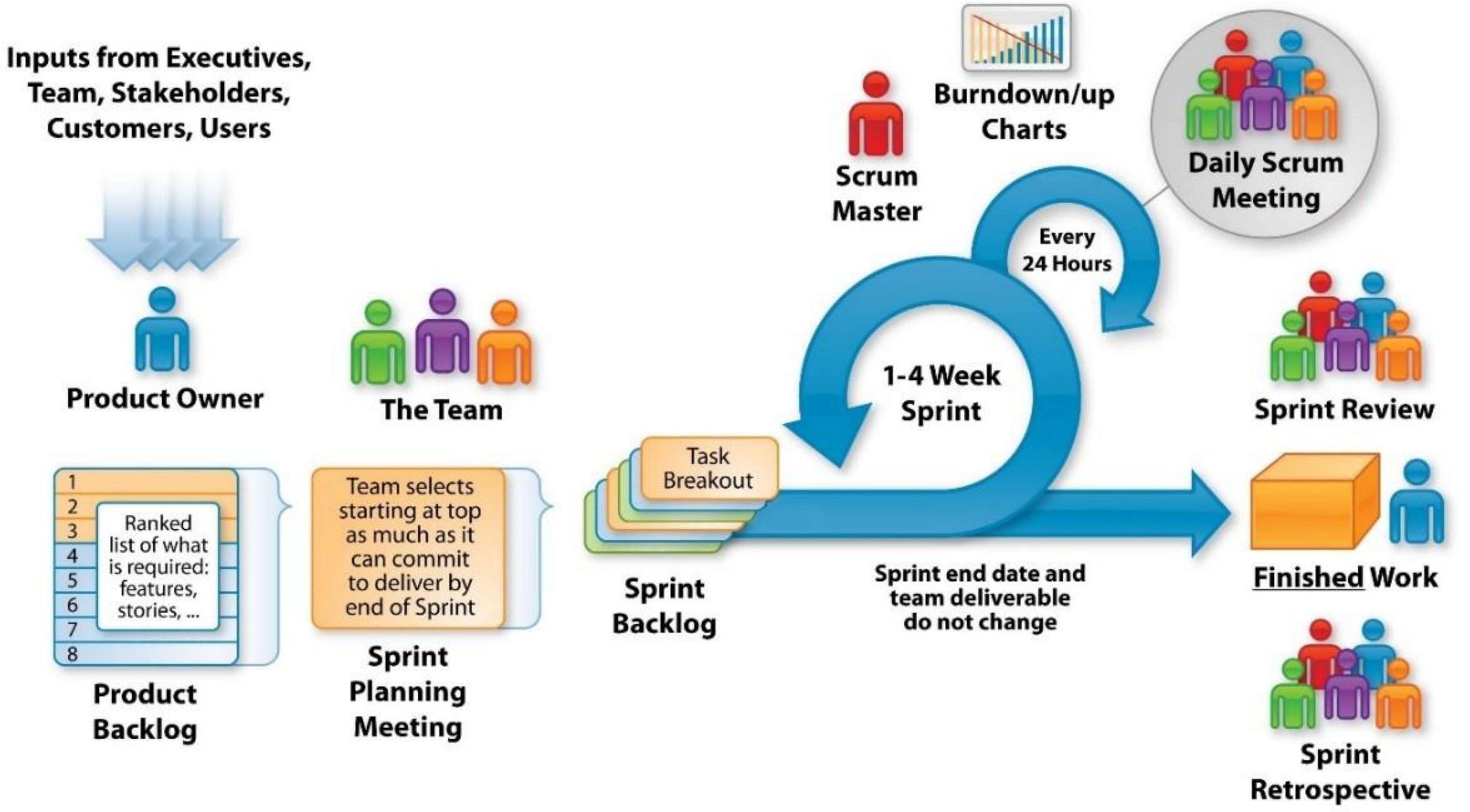
KANBAN METHODOLOGY




TO DO
IN PROGRESS
TESTING


FINANCIAL ANALYTICS
DEBT
PROFIT
NEW VALUE CURVE

The Agile - Scrum Framework



5 Stages of Scrum Sprint

Initiate

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.

Plan and Estimate

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.

Implement

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.

Review and Retrospect

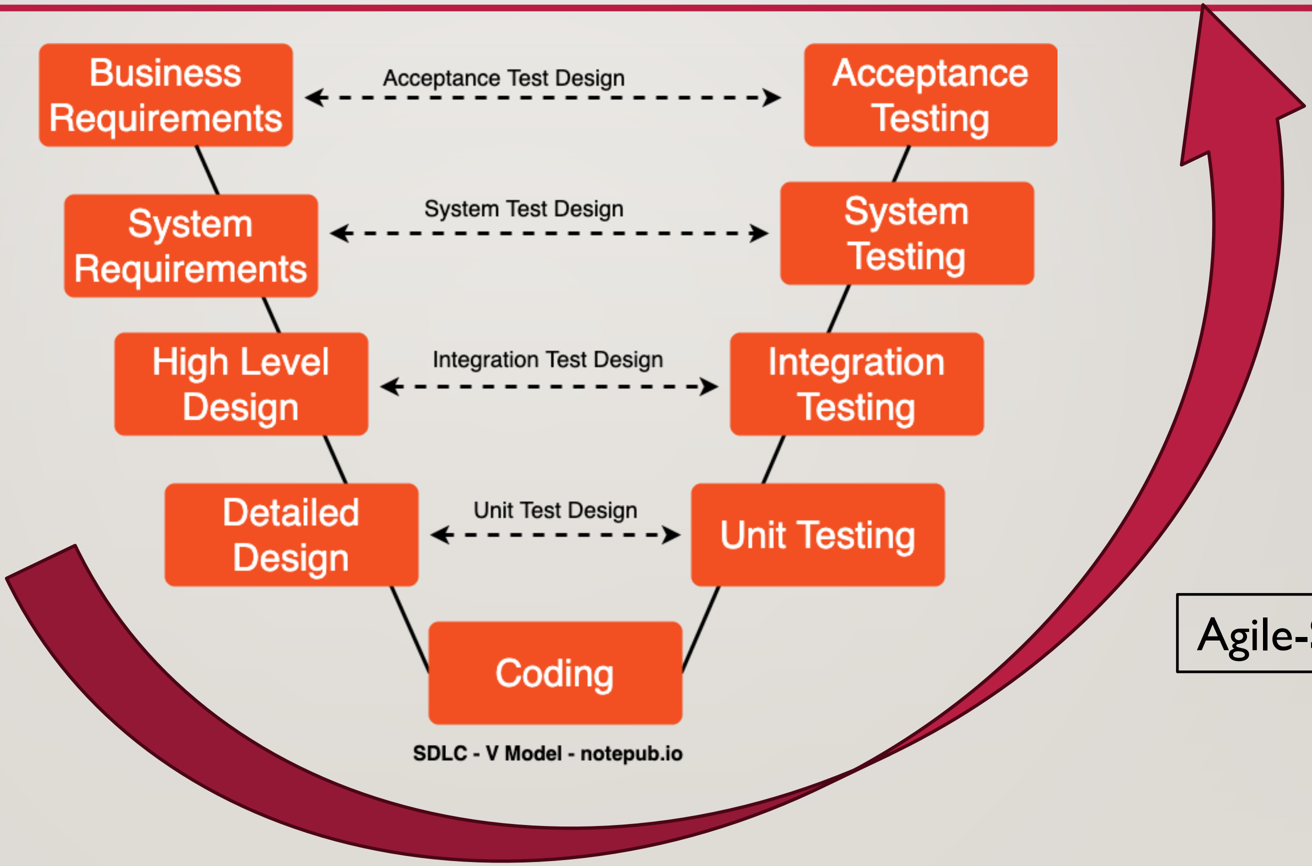
This stage of the project lifecycle is concerned with evaluating what has been accomplished so far, whether the team has worked to plan, and how it can do things better in the future.

Release

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

maruti techlabs

SW DEVELOPMENT PROCESS



Agile-SPRINTs

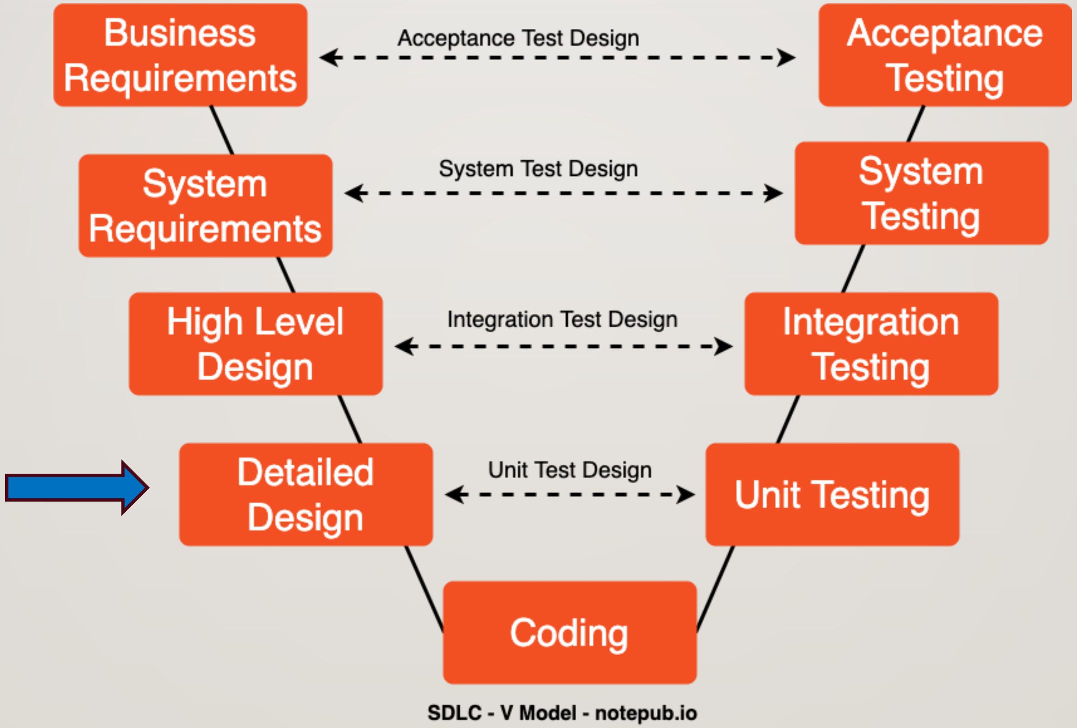
PROJECT SPRINTS

- Detection Alert
  - Camera Capture
  - Object Detection
  - Send messages to other subsystems

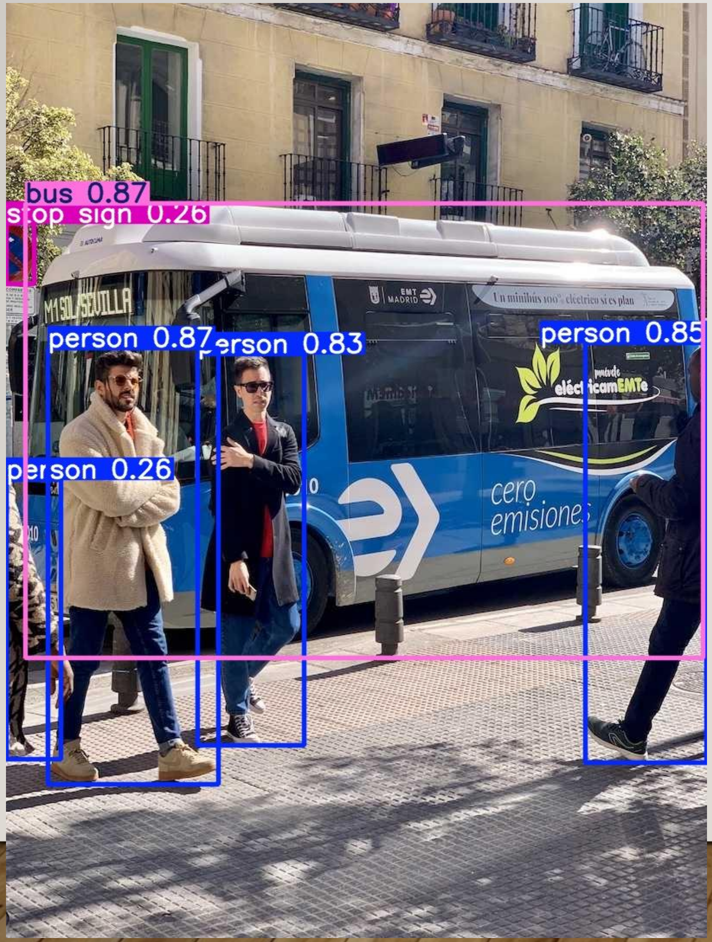
- AMR Controller
  - Receive messages and act accordingly
  - Move using (SLAM) with Obstruction avoidance
  - Target Acquisition (Obj. Det.) and Tracking
  - Follow target using camera and motor control

DETECTION ALERT SPRINT

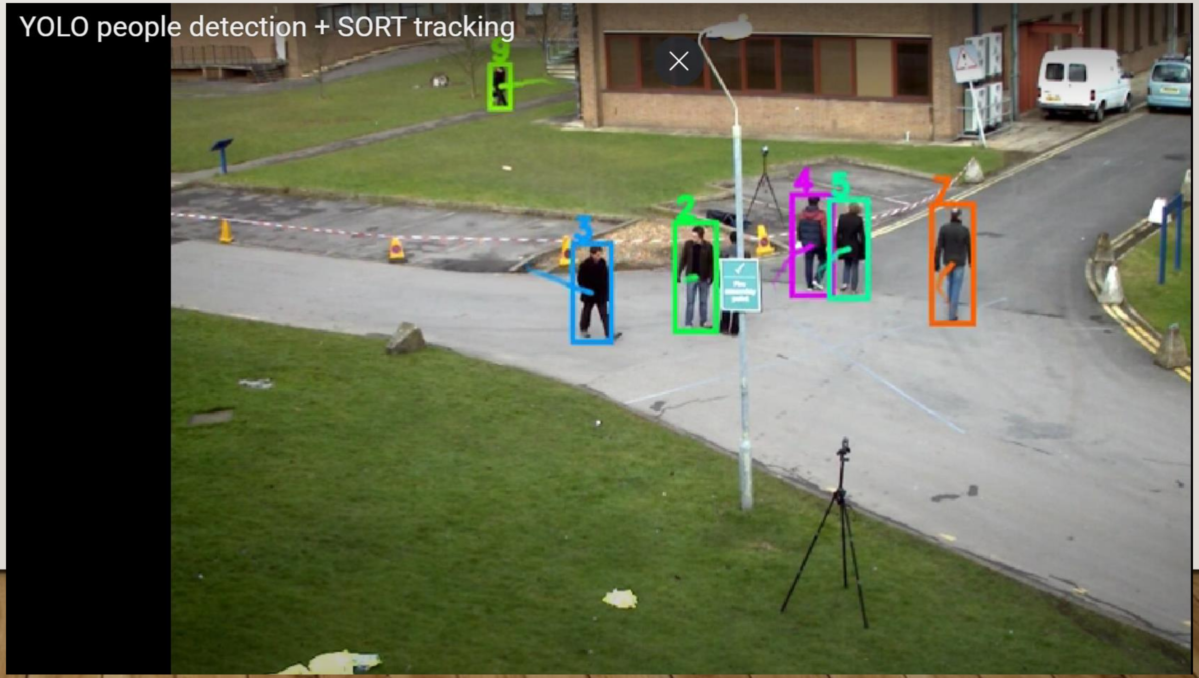
SPRINT 1 - DETECTION ALERT



YOLO OBJ. DET. VS. YOLO TRACKING



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



VISUALIZATION – DETAILED FUNCTIONAL PROCESS (ARCHITECTURE) DIAGRAMS

- To-Be Functional Process Diagram

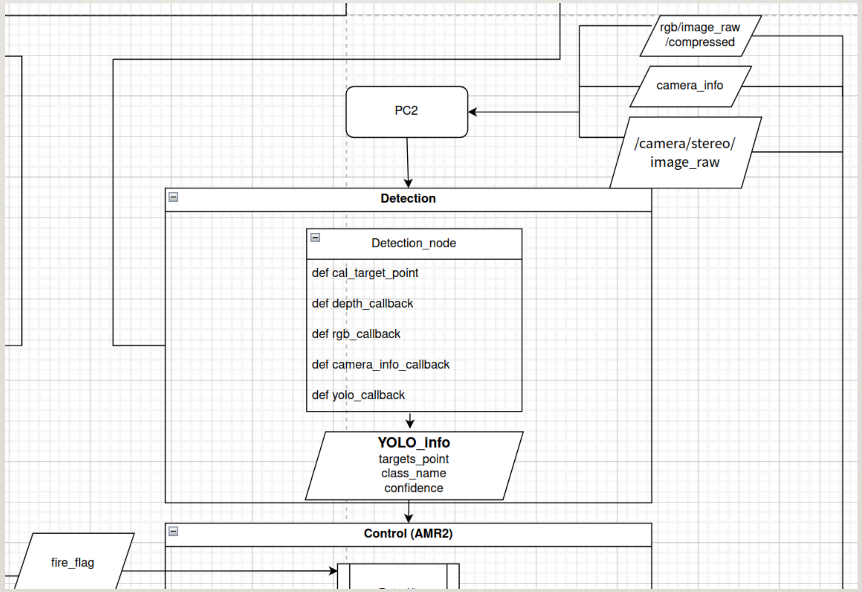
  Detection Alert

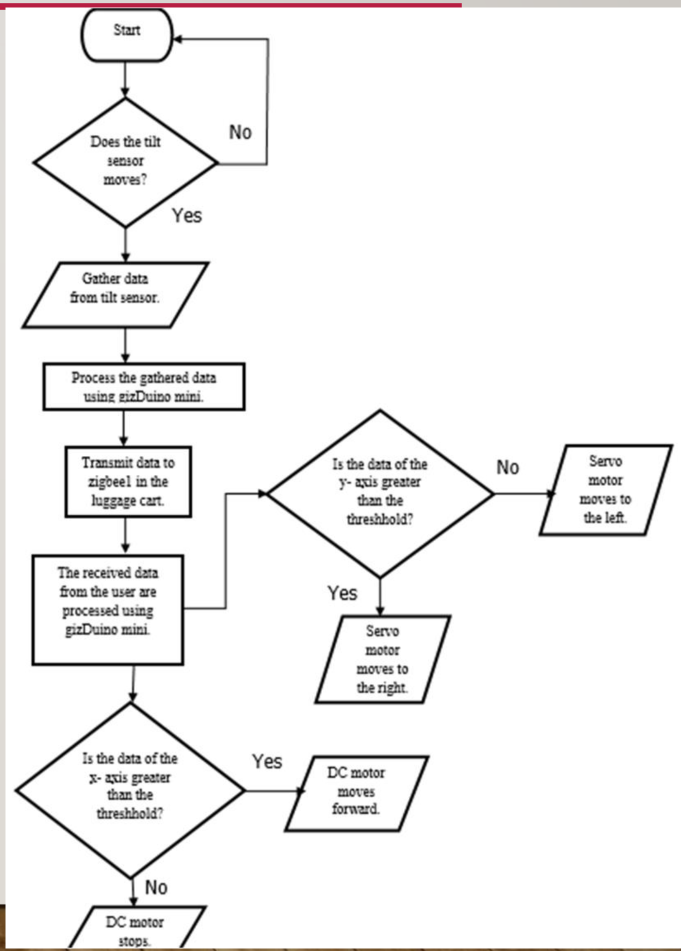
  PC or AMR or Both

  Remember F.I.T.!!

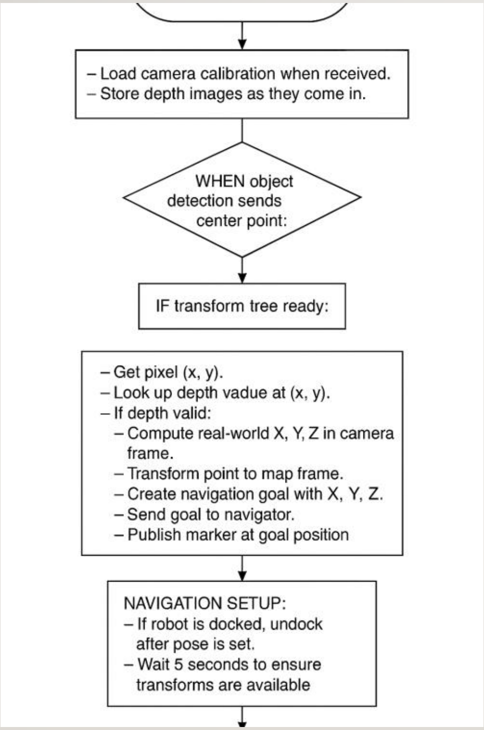
- Testing

  Error & Exception Handling





DETAIL DESIGN EXAMPLE (PSEUDO-CODE)




```
START ROBOT NODE

- Load camera calibration when received.
- Store depth images as they come in.

WHEN object detection sends center point:
    IF transform tree ready:
        - Get pixel (x,y).
        - Look up depth value at (x,y).
        - If depth valid:
            - Compute real-world X,Y,Z in camera frame.
            - Transform point to map frame.
            - Create navigation goal with X,Y,Z.
            - Send goal to navigator.
            - Publish marker at goal position.

NAVIGATION SETUP:
    - If robot is docked, undock after pose is set.
    - Wait 5 seconds to ensure transforms are available.
```

TEAM EXERCISE 4-1

Perform Detail Design of Detection Alert Module using Process Flow Diagram

DETAIL DESIGN REVIEW BY EACH TEAM (SELECTED)

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-Fi 를 통해 모니터링 PC 로 알림을 전송할 수 있습니다.

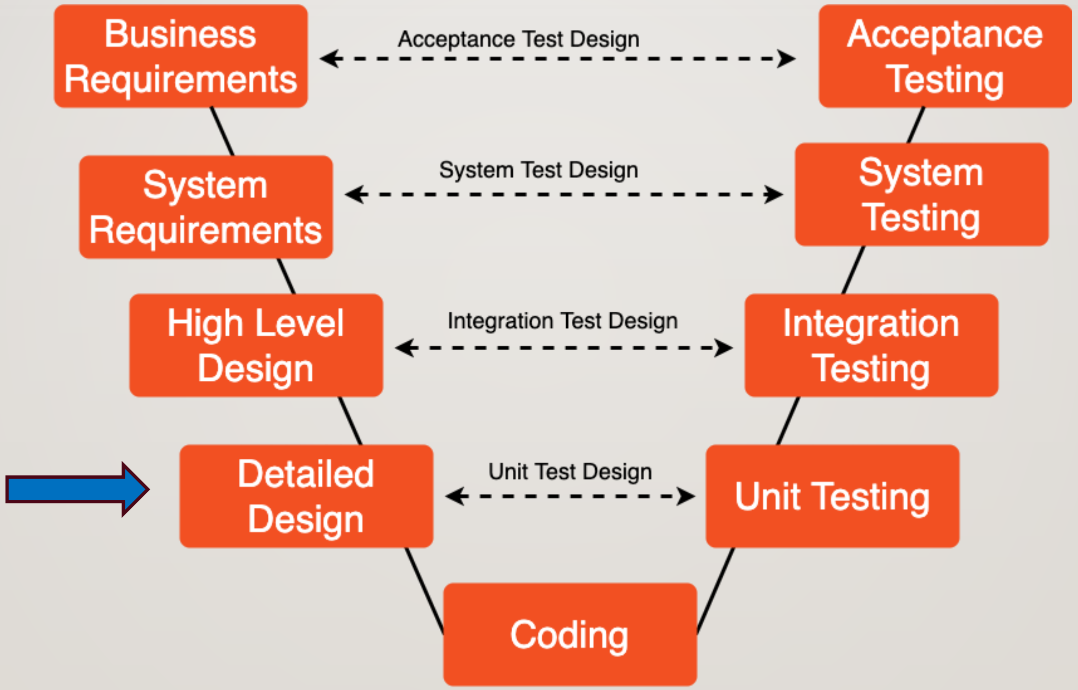
PROJECT SPRINTS

- 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

AMR CONTROLLER SPRINT

SPRINT 2 – AMR CONTROLLER



SDLC - V Model - notepub.io

TEAM EXERCISE 4-2

Perform Detail Design of AMR Controller Module using Process Flow Diagram

DESIGN QUESTIONS:

- How do you start the robot?
  - Initial Position
- How do you find AMR current position and orientation?
- How do you sending Goals?
  - Single goal
  - Multiple goals
- How do you manual control of AMR Odometry?
  - How to move forward, backward, left and right???

DESIGN HINTS:

- How do you manual control of AMR Odometry?
  - Teleops?
  - Navigation?

- Use teleops to move the robots
  - Echo topic: cmd_vel
- geometry_msgs.msg
  - Twist
    - twist.linear.x = <+/-n>
    - twist.angular.z = <+/-n>
  - self.cmd_publisher.publish(twist)

WHAT IS THE FOLLOW ALGORITHM?

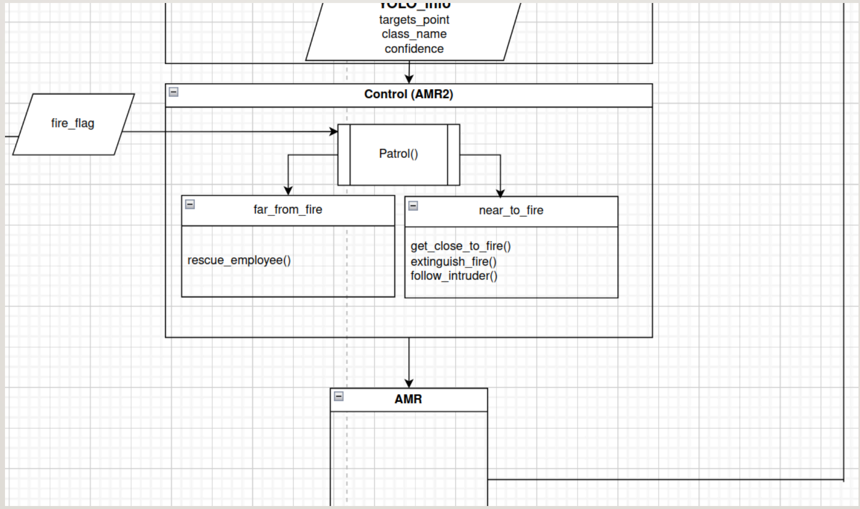
- Left/Right?
- Forward/Backward?
- Velocity?
- Camera position?
- Depth? Local/Global Coordinate transform?

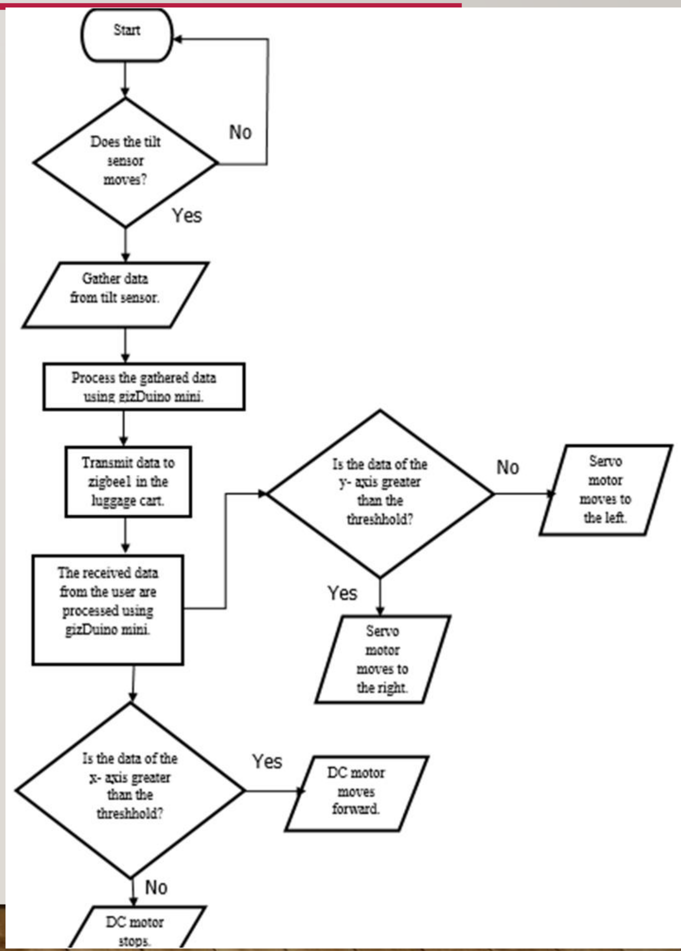
VISUALIZATION – DETAILED FUNCTIONAL PROCESS (ARCHITECTURE) DIAGRAMS

- To-Be Functional Process Diagram

AMR Controller

Remember F.I.T.!!





DETAIL DESIGN REVIEW BY EACH TEAM (SELECTED)

Using the process flow diagram present team’s design

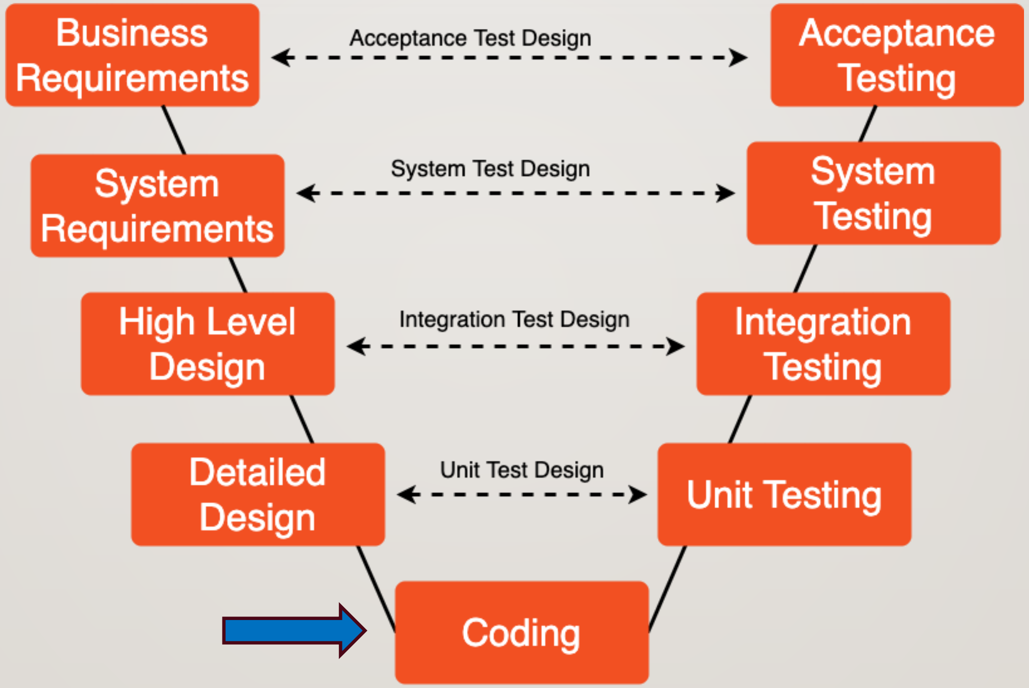
CODING/TESTING

PROJECT SPRINTS

- Detection Alert
  - Camera Capture
  - Object Detection
  - Send messages to other subsystems

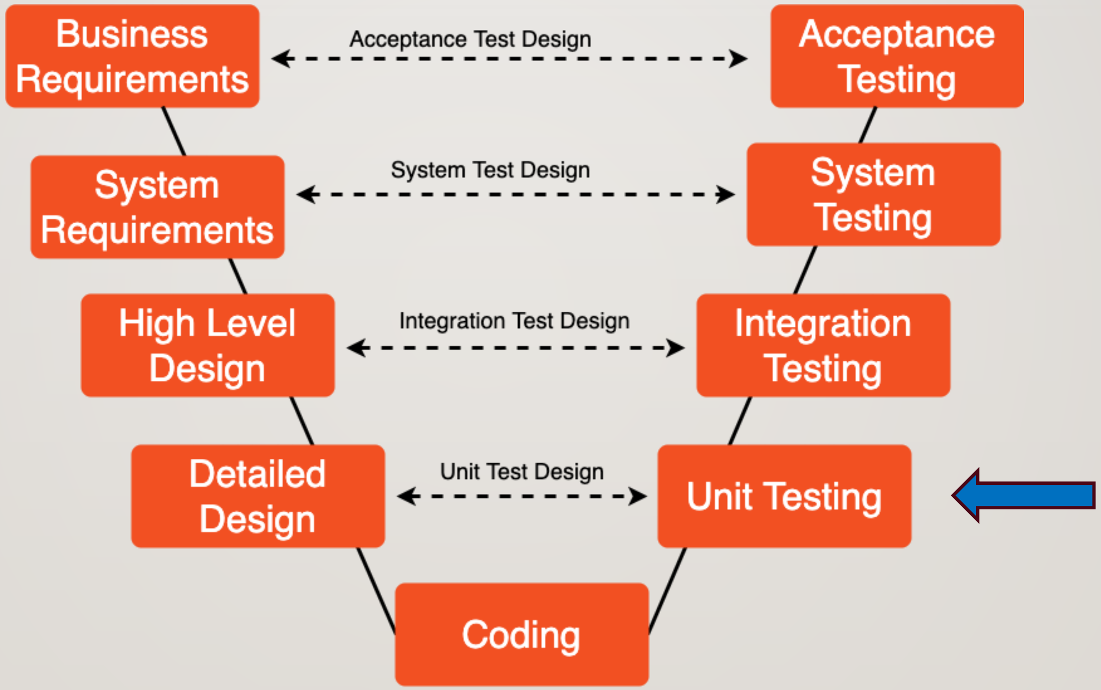
- AMR Controller
  - Receive messages and act accordingly
  - Move using (SLAM) with Obstruction avoidance
  - Target Acquisition (Obj. Det.) and Tracking
  - Follow target using camera and motor control

SPRINT 1 - DETECTION ALERT



SDLC - V Model - notepub.io

SPRINT 1 - DETECTION ALERT



SDLC - V Model - notepub.io

TEAM EXERCISE 5

Perform coding and testing of Detection Alert Module

CODING HINTS:

- How do you initiate transform?

```
        # TF2 buffer for transforms (camera_frame -> base_link -> map)
        self.tf_buffer = Buffer()
        self.tf_listener = TransformListener(self.tf_buffer, self)
```

CODING HINTS:

- How do you time sync RGB and Depth?

```python
        # Time-synchronized subscribers for RGB + Depth images
        self.rgb_sub = Subscriber(self, CompressedImage, self.rgb_topic)
        self.depth_sub = Subscriber(self, ROSImage, self.depth_topic)

        # ApproximateTimeSynchronizer: allows slight timestamp mismatch
        self.ts = ApproximateTimeSynchronizer(
            [self.rgb_sub, self.depth_sub],
            queue_size=10,
            slop=0.1
        )
        self.ts.registerCallback(self.synced_callback)
```

CODING HINTS:

- How do you launch multi-thread?

```python

def main():
    rclpy.init()
    node = DepthToMap()
    executor = MultiThreadedExecutor()
    executor.add_node(node)

```

EXPECTED OUTCOME

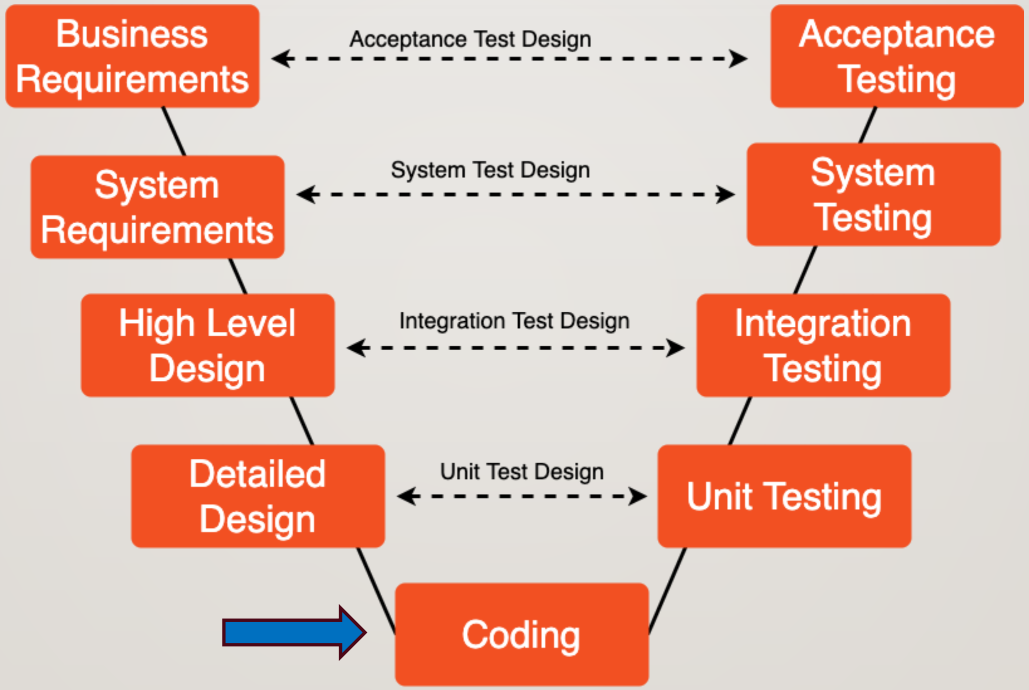
- Successful object detection
- ROS Nodes, and Topics created to send and display images and data

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

SPRINT 2 – AMR CONTROLLER



SDLC - V Model - notepub.io

CODING HINTS:

- How do you start the robot?

```
# Start on dock
if not navigator.getDockedStatus():
    navigator.info('Docking before intialising pose')
    navigator.dock()

# Set initial pose
initial_pose = navigator.getPoseStamped([0.0, 0.0], TurtleBot4Directions.NORTH)
navigator.setInitialPose(initial_pose)
```

CODING HINTS:

- How do you find AMR current position and orientation?
  - Echo topic: amcl_pose
  - Use Rviz Publish Points & Echo clicked_points
  - Etc…

CODING HINTS:

- How do you sending Goals?
  - Single goal

```
# Wait for Nav2
navigator.waitUntilNav2Active()
```

```
    # Set goal poses
    # goal_pose = navigator.getPoseStamped([-13.0, 9.0], TurtleBot4Directions.EAST)
    goal_pose = navigator.getPoseStamped([-1.7, -0.1], TurtleBot4Directions.EAST)

#   x: 2.0924954414367676
#   y: 4.481560230255127
# [x,y]=[-1.707,-0.106]

    # Undock
    navigator.undock()

    # Go to each goal pose
    navigator.startToPose(goal_pose)
```

CODING HINTS:

- How do you sending Goals?
  - Single goal
  - Multiple goals

```python
# Set goal poses
goal_pose = []
goal_pose.append(navigator.getPoseStamped([-1.7, -0.1], TurtleBot4Directions.EAST))
goal_pose.append(navigator.getPoseStamped([-1.1, 1.6], TurtleBot4Directions.NORTH))
goal_pose.append(navigator.getPoseStamped([-1.0, 0.05], TurtleBot4Directions.NORTH_WEST))
```

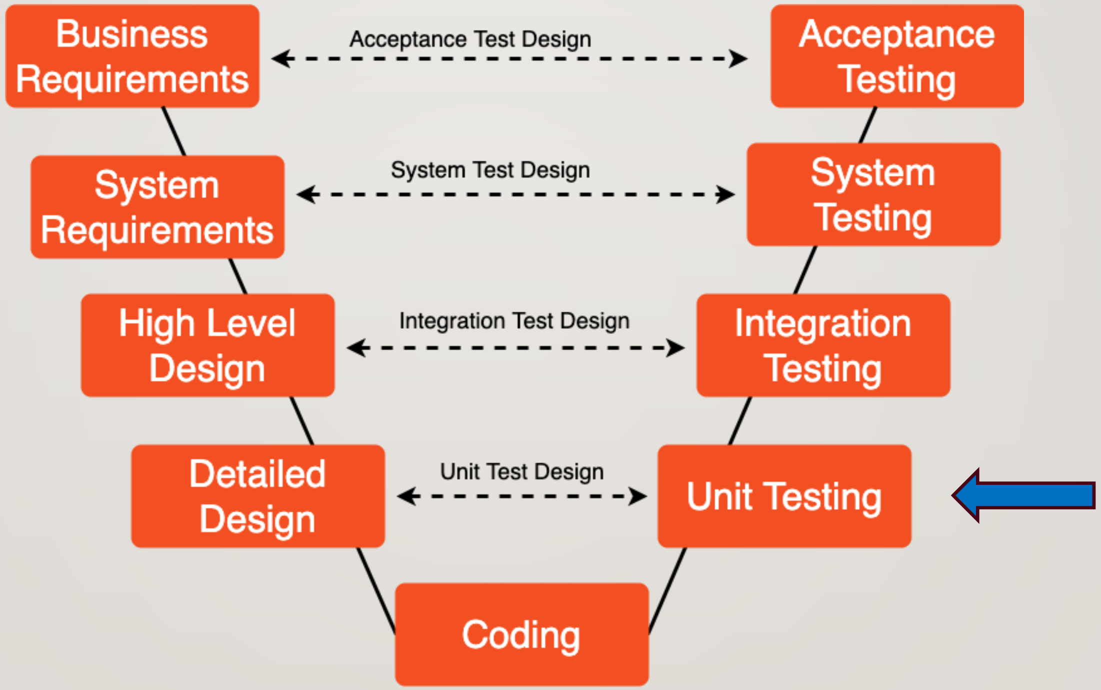
```python
# Undock
navigator.undock()

# Navigate through poses
navigator.startThroughPoses(goal_pose)
```

```python
# Follow Waypoints
navigator.startFollowWaypoints(goal_pose)
```

CODING HINTS:

- How do you reduce the depth error?
  - Bigger sampling area?
  - Multiple images?
  - Kalman Filter or other optimization functions?
  - What about the time sync between RGB and Stereo/Depth?
  - Etc….

SPRINT 2 – AMR CONTROLLER



SDLC - V Model - notepub.io

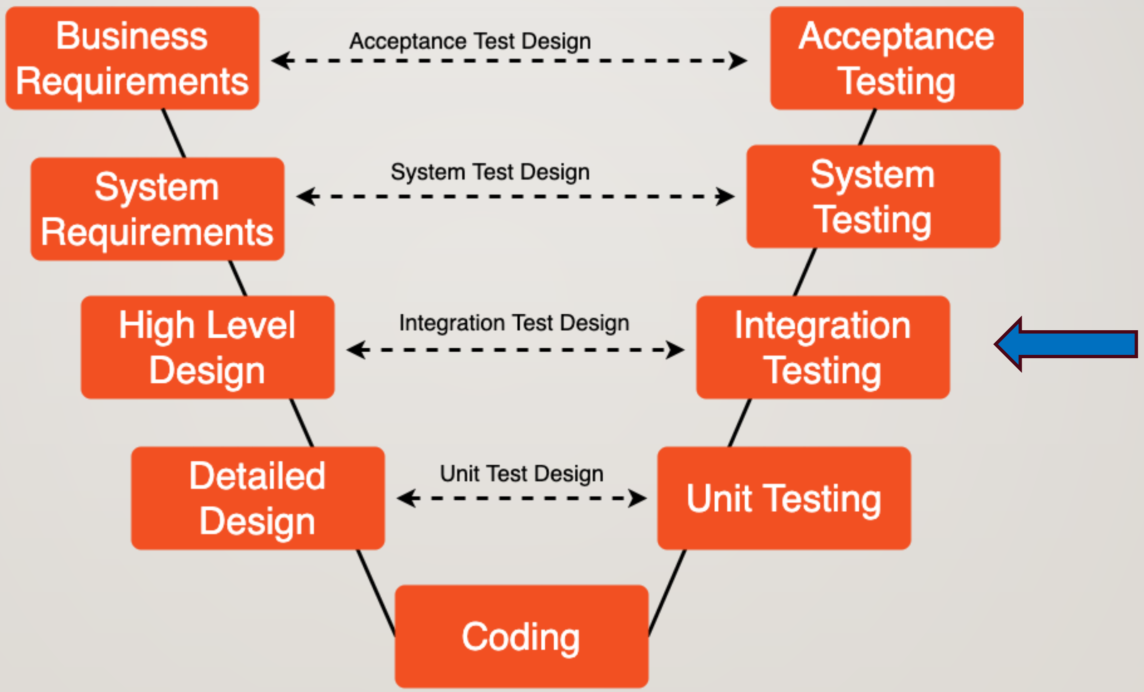
TEAM EXERCISE 6

Perform coding and testing of AMR Controller Module

EXPECTED OUTCOME

- AMR navigates while avoiding obstacles

SPRINT 1&2 – DETECTION ALERT/AMR CONTROLLER INTEGRATION & TEST



SDLC - V Model - notepub.io

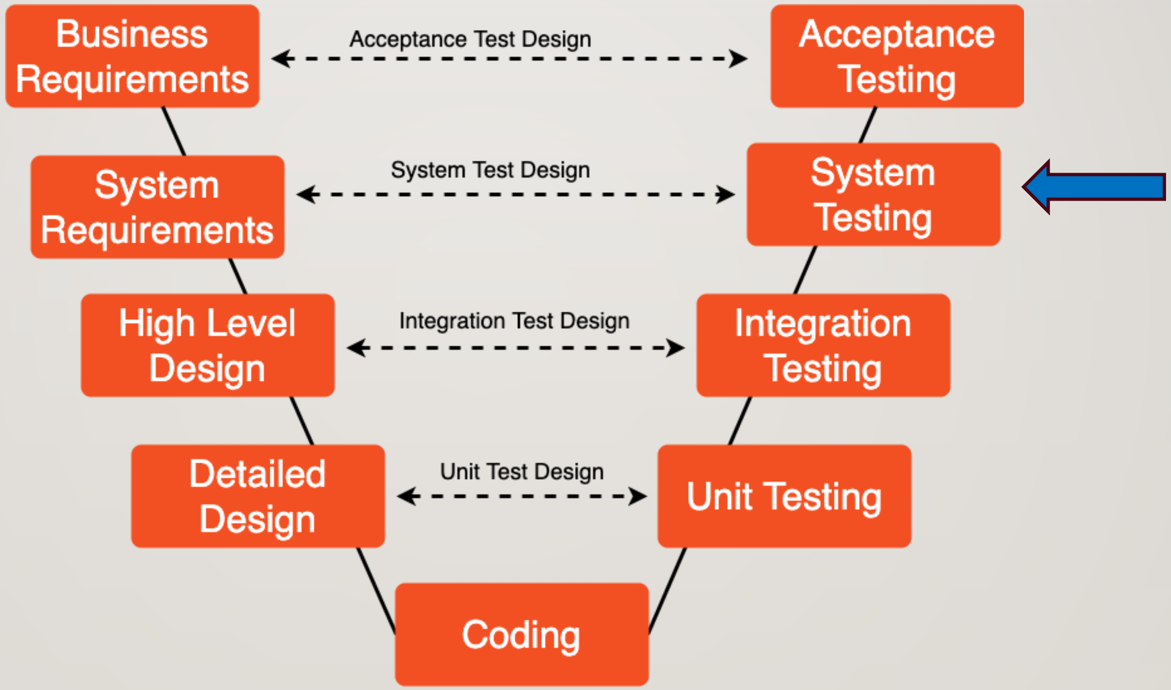
TEAM EXERCISE 7

Perform integrate and test of Detection Alert and AMR Controller Modules

EXPECTED OUTCOME

AMR navigates to avoid obstacles, ignores dummies, track, and approach target

SYSTEM INTEGRATION & TEST



SDLC - V Model - notepub.io

TEAM EXERCISE 8

Perform integration and testing of Detection, and AMR Controller

DEMONSTRATION OF SOLUTION BY EACH TEAM

Show actual results against the expected results and explain

TESTING AND INTEGRATION HINTS

- Test in pieces with a plan
- Do most of testing without using the actual robot, first.
  - Use tools like "ros bag" to record ahead the topics that you need and replay it for testing
- Use published topics instead of imshow(), and rqt or rviz to view
- Put in place plenty of log/print statements to debug
- Use visual aids over text to validate the results; ex: rviz
- Integrate in pieces: module to module; sub-system to sub-system

RECOVERY

AMR

- turtlebot4-service-restart
  - Turtlebot4 bring up service
- turtlebot4-source
  - Similar to source .bashrc
- turtlebot4-daemon-restart
  - ros2 daemon stop and start
- sudo reboot
- sudo shutdown now

PC

- Localization should come up every time if AMR is stable
- Execute in the order of Localization, Rviz (set init_pose using 2d_pose_estimate; then undock), and Nav2
- If Nav2 is not coming up, then execute
  - ros-restart (ros2 daemon stop; ros2 daemon start), first
  - Then try executing navigation again

프로젝트 RULE NUMBER ONE!!!

Are we still having FUN!