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.

---

3. Navigation Subsystem Design

3.1 Navigation Requirements

- **Autonomous Path Planning:** The AMR should autonomously navigate through predefined or dynamically generated paths, adjusting for obstacles.
- **Real-Time Obstacle Avoidance:** The AMR must detect and avoid obstacles using LiDAR and ultrasonic sensors.
- **Localization:** The AMR must maintain accurate knowledge of its position within the environment using SLAM (Simultaneous Localization and Mapping).
- **Safe Return to Charging Station:** The AMR should autonomously return to the charging dock when battery levels are low.

3.2 Architecture

1. SLAM Module: Uses ROS2's SLAM (Simultaneous Localization and Mapping) library with LiDAR and wheel encoders for real-time map creation and localization.
2. Path Planning Module: Utilizes ROS2 Navigation Stack for path planning, integrating map data and sensor inputs for optimal pathfinding.
3. Obstacle Detection and Avoidance Module: Integrates data from LiDAR and ultrasonic sensors to detect and navigate around obstacles.

3.3 Data Flow for Navigation

1. Input Data: LiDAR, ultrasonic sensors, and odometry data from wheel encoders.
2. Process Flow:
    - SLAM Processing: LiDAR data is fed into the SLAM algorithm to create a real-time map and update the robot's position.
    - Path Planning: The updated map and localization data are used by the Path Planning Module to generate a safe, optimized path.
    - Obstacle Detection: LiDAR and ultrasonic sensor data are continuously analyzed to detect obstacles.
    - Obstacle Avoidance: If an obstacle is detected, the Path Planning Module recalculates the path to avoid the obstruction.
3. Output Data: Navigation commands (velocity and direction) sent to the AMR's drive system.

---

4. Threat Detection Subsystem Design

4.1 Threat Detection Requirements

- Object Detection: Detect and identify unauthorized personnel or suspicious objects within the AMR's field of view.
- Real-Time Processing: Detect threats with minimal latency to provide real-time alerts.
- Anomaly Logging: Log any detected anomalies for future analysis, including image frames and timestamps.
- Immediate Alert Transmission: Send alerts to the monitoring PC when a potential threat is detected.

4.2 Architecture

1. Image Capture Module: Uses a USB camera to capture live video, feeding frames to the Threat Detection Module.

2. **Object Detection Module:** Runs on Jetson-Orin, leveraging YOLO (You Only Look Once) for real-time object detection.
3. **Anomaly Detection Module:** Analyzes detection data for predefined threat types (e.g., unauthorized personnel), logging any identified anomalies and initiating an alert when a threat is detected.

4.3 Data Flow for Threat Detection

1. **Input Data:** Live video frames captured from the USB camera.
2. **Process Flow:**
    - **Image Preprocessing:** Frames from the USB camera are resized and prepared for input into the YOLO model.
    - **Object Detection (YOLO):** YOLO model analyzes frames to identify objects of interest, such as people or suspicious objects.
    - **Anomaly Evaluation:** Identified objects are cross-referenced with predefined threat categories.
    - **Logging:** If an anomaly is detected, the system logs the frame, location, and timestamp for future analysis.
    - **Alert Transmission:** An alert is sent to the monitoring PC with details of the threat.
3. **Output Data:** Detected objects, anomaly logs, and alert notifications.

---

5. Component Descriptions

5.1 SLAM Module

- **Library Used:** ROS2 SLAM Toolbox
- **Inputs:** LiDAR data, wheel encoder odometry
- **Outputs:** Real-time map, robot localization data
- **Functionality:** Processes sensor data to create a map of the environment while estimating the AMR's current position.

5.2 Path Planning Module

- **Library Used:** ROS2 Navigation Stack
- **Inputs:** SLAM map, current localization data, sensor data
- **Outputs:** Navigation commands (linear and angular velocities)
- **Functionality:** Calculates optimal path to reach the destination while avoiding obstacles, updating the path in real-time based on sensor input.

5.3 Obstacle Detection and Avoidance Module

- **Sensors Used:** LiDAR, ultrasonic sensors
- **Inputs:** Real-time sensor data
- **Outputs:** Navigation adjustments to avoid detected obstacles
- **Functionality:** Continuously monitors for obstacles in the robot's path, triggering the Path Planning Module to adjust the route as needed.

5.4 Image Capture Module

- **Device Used:** USB Camera
- **Inputs:** Live video feed
- **Outputs:** Frames for object detection
- **Functionality:** Continuously captures images and sends them to the Threat Detection Module for analysis.

5.5 Object Detection Module

- **Model Used:** YOLO (You Only Look Once) running on Jetson-Orin
- **Inputs:** Image frames from the Image Capture Module
- **Outputs:** Detected objects and classifications
- **Functionality:** Analyzes each frame to detect and classify objects, flagging any predefined threat categories.

5.6 Anomaly Detection Module

- **Library Used:** Custom script with YOLO integration
- **Inputs:** Object classifications from the Object Detection Module
- **Outputs:** Anomaly logs, alert signals
- **Functionality:** Cross-references detected objects with threat categories, logging and sending alerts if an anomaly is identified.

---

6. Data Storage and Logging

- **Navigation Logs:** Log AMR paths, detected obstacles, and battery status.
- **Threat Detection Logs:** Store frames of detected threats, including timestamps and location.
- **Log Retention:** Navigation and threat detection logs are retained for 48 hours on the AMR, with critical events stored longer if needed.

---

7. User Interface Design (Dashboard)

- **Real-Time Monitoring:** Dashboard displays the AMR's current position, live video feed, and alerts.
- **Navigation Status:** Shows route details, current speed, and obstacle status.
- **Threat Detection Alerts:** Provides real-time notifications and access to logs of detected threats, allowing review and manual override if necessary.
- **Manual Override:** Enables security personnel to take manual control of the AMR, adjusting its navigation or stopping it as needed.

---

8. Testing and Validation

8.1 Navigation Testing

- **Path Planning Test:** Validate path planning under various obstacles and environmental layouts.
- **Obstacle Avoidance Test:** Test AMR's reaction to static and dynamic obstacles.
- **SLAM Accuracy Test:** Verify the accuracy of map generation and localization against known reference points.

8.2 Threat Detection Testing

- **Object Detection Test:** Test YOLO's ability to detect and classify predefined threat objects.
- **Latency Test:** Measure detection-to-alert latency to ensure real-time response.
- **False Positive/Negative Test:** Verify system's reliability in identifying true threats while minimizing false alerts.

---

9. Error Handling and Recovery

1. **Navigation Errors:** If localization fails, the AMR pauses and attempts to re-establish position using SLAM. If unresolved, an alert is sent to the monitoring PC.
2. **Threat Detection Errors:** If the camera or object detection model fails, the system notifies the PC dashboard, prompting maintenance.
3. **Network Loss:** The AMR continues autonomous navigation, storing logs locally until the Wi-Fi connection is restored.

10. Performance Requirements

- **Navigation Latency:** Path recalculations and obstacle avoidance updates within 100 ms.
- **Threat Detection Latency:** Object detection and alert notification within 1 second.
- **Battery Life:** Minimum 8 hours of continuous operation.

---

11. Appendix

- **Software Libraries:** ROS2, OpenCV, YOLO, and supporting Python libraries.
- **Glossary:** Definitions of key terms like SLAM, YOLO, localization, and path planning.
- **References:** Documentation links for ROS2, YOLO, and other libraries used in the design.

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.

---

3. 네비게이션 서브시스템 설계

3.1 네비게이션 요구사항

- **자율 경로 계획:** AMR 은 장애물에 따라 조정하며 사전 정의된 경로 또는 동적으로 생성된 경로를 따라 자율적으로 이동해야 합니다.
- **실시간 장애물 회피:** AMR 은 LiDAR 및 초음파 센서를 사용하여 실시간으로 장애물을 감지하고 회피해야 합니다.
- **위치 파악:** AMR 은 SLAM(동시 위치추정 및 매핑)을 사용하여 환경 내에서 정확한 위치를 유지해야 합니다.
- **충전소로 안전 복귀:** 배터리 수준이 낮을 때 AMR 은 자율적으로 충전 도크로 돌아가야 합니다.

3.2 아키텍처

1. **SLAM 모듈:** ROS2 의 SLAM(동시 위치추정 및 매핑) 라이브러리를 사용하여 LiDAR 및 휠 인코더 데이터를 처리해 실시간 맵을 생성하고 위치를 추정합니다.

2. **경로 계획 모듈:** ROS2 네비게이션 스택을 사용하여 경로 계획을 수행하며, 맵 데이터와 센서 입력을 통합해 최적의 경로를 탐색합니다.
3. **장애물 감지 및 회피 모듈:** LiDAR 와 초음파 센서 데이터를 통합하여 장애물을 감지하고 이를 회피합니다.

3.3 네비게이션 데이터 흐름

1. **입력 데이터:** LiDAR, 초음파 센서, 휠 인코더의 위치 데이터
2. **프로세스 흐름:**
    - **SLAM 처리:** LiDAR 데이터를 SLAM 알고리즘에 입력하여 실시간 맵을 생성하고 로봇의 위치를 업데이트합니다.
    - **경로 계획:** 업데이트된 맵과 위치 데이터를 경로 계획 모듈에 전달하여 안전하고 최적화된 경로를 생성합니다.
    - **장애물 감지:** LiDAR 와 초음파 센서 데이터를 지속적으로 분석하여 장애물을 감지합니다.
    - **장애물 회피:** 장애물이 감지되면 경로 계획 모듈에서 회피를 위한 경로를 재계산합니다.
3. **출력 데이터:** AMR 의 구동 시스템으로 전달되는 네비게이션 명령(속도 및 방향).

---

4. 위협 탐지 서브시스템 설계

4.1 위협 탐지 요구사항

- **객체 탐지:** AMR 의 시야 내에서 무단 접근자나 의심스러운 물체를 감지하고 식별해야 합니다.
- **실시간 처리:** 최소 지연 시간으로 위협을 감지하여 실시간 경고를 제공해야 합니다.
- **이상 로그 기록:** 감지된 이상 현상을 기록하고 이미지 프레임 및 타임스탬프를 포함해 저장해야 합니다.
- **즉각적인 알림 전송:** 잠재적인 위협이 감지되면 모니터링 PC 로 경고를 전송해야 합니다.

4.2 아키텍처

1. **이미지 캡처 모듈:** USB 카메라를 사용하여 라이브 비디오를 캡처하고, 프레임을 위협 탐지 모듈로 전달합니다.
2. **객체 탐지 모듈:** Jetson-Orin 에서 YOLO(You Only Look Once)를 활용해 실시간 객체 탐지를 수행합니다.
3. **이상 탐지 모듈:** 탐지된 데이터를 분석하여 미리 정의된 위협 유형(예: 무단 접근자)에 따라 이상 현상을 기록하고 위협이 감지되면 알림을 전송합니다.

4.3 위협 탐지 데이터 흐름

1. **입력 데이터:** USB 카메라로부터 캡처된 라이브 비디오 프레임
2. **프로세스 흐름:**
    - **이미지 전처리:** USB 카메라에서 수집된 프레임을 YOLO 모델에 입력하기 위해 크기 조정 및 전처리 수행
    - **객체 탐지(YOLO):** YOLO 모델이 프레임을 분석하여 사람이나 의심스러운 물체와 같은 객체를 식별
    - **이상 평가:** 감지된 객체를 미리 정의된 위협 카테고리와 대조하여 평가
    - **로그 기록:** 이상이 감지되면 시스템이 프레임, 위치, 타임스탬프를 기록하여 분석을 위해 저장
    - **알림 전송:** 위협이 확인되면 위협 세부 정보를 포함한 알림이 모니터링 PC 로 전송
3. **출력 데이터:** 감지된 객체, 이상 로그 및 알림

---

5. 구성 요소 설명

5.1 SLAM 모듈

- **사용 라이브러리:** ROS2 SLAM Toolbox
- **입력:** LiDAR 데이터, 휠 인코더 위치 데이터
- **출력:** 실시간 맵, 로봇 위치 데이터
- **기능:** 센서 데이터를 처리하여 환경의 맵을 생성하고 로봇의 현재 위치를 추정

5.2 경로 계획 모듈

- **사용 라이브러리:** ROS2 네비게이션 스택
- **입력:** SLAM 맵, 현재 위치 데이터, 센서 데이터
- **출력:** 네비게이션 명령(직선 및 각속도)
- **기능:** 목적지에 도달하기 위해 최적의 경로를 계산하고, 장애물 회피가 필요한 경우 실시간으로 경로를 업데이트

5.3 장애물 감지 및 회피 모듈

- **사용 센서:** LiDAR, 초음파 센서
- **입력:** 실시간 센서 데이터
- **출력:** 감지된 장애물을 회피하는 네비게이션 조정
- **기능:** 로봇의 경로 내 장애물을 지속적으로 모니터링하여, 경로 계획 모듈이 필요시 회피 경로를 재설정하도록 유도

5.4 이미지 캡처 모듈

- **사용 장치:** USB 카메라
- **입력:** 라이브 비디오 피드
- **출력:** 객체 탐지를 위한 프레임
- **기능:** 이미지를 지속적으로 캡처하여 위협 탐지 모듈에 분석을 위해 전달

5.5 객체 탐지 모듈

- **사용 모델:** Jetson-Orin 에서 실행되는 YOLO
- **입력:** 이미지 캡처 모듈에서 받은 프레임
- **출력:** 감지된 객체와 분류
- **기능:** 각 프레임을 분석하여 객체를 감지 및 분류하고, 미리 정의된 위협 카테고리에 해당되는 경우 플래그 지정

5.6 이상 탐지 모듈

- **사용 라이브러리:** YOLO 통합된 커스텀 스크립트
- **입력:** 객체 탐지 모듈의 객체 분류 결과
- **출력:** 이상 로그 및 알림 신호
- **기능:** 감지된 객체를 위협 카테고리와 비교하여 이상이 확인되면 로그를 기록하고 알림을 전송

---

6. 데이터 저장 및 로그 기록

- **네비게이션 로그:** AMR 경로, 감지된 장애물 및 배터리 상태를 기록
- **위협 탐지 로그:** 감지된 위협 프레임, 타임스탬프 및 위치를 저장
- **로그 보존:** 네비게이션 및 위협 탐지 로그는 AMR 에 48 시간 동안 보관되며, 중요한 이벤트는 더 오랫동안 저장 가능

---

7. 사용자 인터페이스 설계 (대시보드)

- **실시간 모니터링:** 대시보드에 AMR 의 현재 위치, 라이브 비디오 피드 및 경고 표시
- **네비게이션 상태:** 경로 세부 사항, 현재 속도 및 장애물 상태 표시
- **위협 탐지 경고:** 감지된 위협의 로그에 실시간 알림과 접근, 수동 제어 기능 제공
- **수동 제어:** 보안 담당자가 필요시 AMR 을 수동으로 제어할 수 있도록 기능 제공

---

8. 테스트 및 검증

8.1 네비게이션 테스트

- **경로 계획 테스트:** 다양한 장애물 및 환경 레이아웃에서 경로 계획을 검증
- **장애물 회피 테스트:** 정적 및 동적 장애물에 대한 AMR 반응 테스트
- **SLAM 정확도 테스트:** 생성된 맵과 로봇 위치가 정확한지 확인

8.2 위협 탐지 테스트

- **객체 탐지 테스트:** YOLO 가 미리 정의된 위협 객체를 감지하고 분류할 수 있는지 검증
- **지연 테스트:** 감지에서 알림까지의 지연 시간을 측정하여 실시간 반응을 보장
- **오탐 및 누락 테스트:** 실제 위협을 정확히 식별하고 오탐을 최소화하는 신뢰성 검증

---

9. 오류 처리 및 복구

1. **네비게이션 오류:** 위치 추정이 실패할 경우, AMR 은 멈추고 SLAM 을 통해 위치를 다시 설정 시도. 해결되지 않으면 모니터링 PC 로 알림 전송.
2. **위협 탐지 오류:** 카메라나 객체 탐지 모델에 오류가 발생하면 PC 대시보드에 알림을 전송해 유지보수를 요청.
3. **네트워크 손실:** AMR 은 자율 네비게이션을 계속하며, 연결이 복구될 때까지 로그를 로컬에 저장.

---

10. 성능 요구사항

- **네비게이션 지연:** 경로 재계산 및 장애물 회피 업데이트는 100ms 이내로 처리
- **위협 탐지 지연:** 객체 감지 및 알림 전송은 1 초 이내로 처리
- **배터리 수명:** 최소 8 시간 연속 운영 가능

---

11. 부록

- **소프트웨어 라이브러리:** ROS2, OpenCV, YOLO, 및 지원되는 Python 라이브러리
- **용어 사전:** SLAM, YOLO, 위치 추정, 경로 계획과 같은 주요 용어 정의
- **참고 문헌:** 설계에 사용된 ROS2, YOLO 및 기타 라이브러리 문서 링크