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.

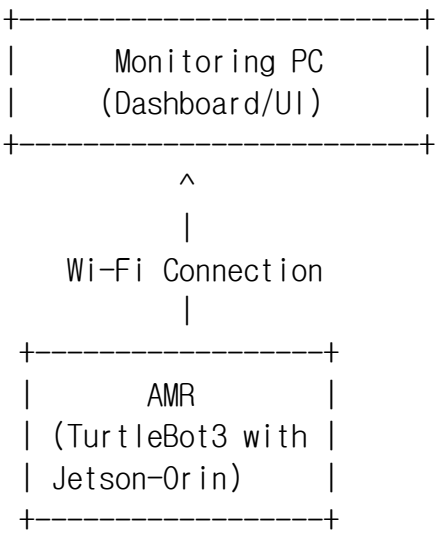
---

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.

2.1 High-Level Architecture Diagram

sql
Copy code



---

3. Component Design

3.1 Hardware Components

1. PC (Monitoring System)
    - Operating System: Ubuntu 22.04
    - Camera: USB Camera for local visual monitoring, if required
    - Network: Wi-Fi 6 for reliable communication with the AMR
2. Autonomous Mobile Robot (AMR) - TurtleBot3
    - Processor: Jetson-Orin for onboard AI processing and data handling
    - Operating System: Ubuntu 22.04 with ROS2 for robotics control
    - Sensors:
        - LiDAR: For navigation and obstacle detection
        - USB Camera: For capturing video feeds for threat detection
        - Ultrasonic Sensors: For close-range obstacle avoidance
    - Battery: Minimum 8-hour battery life with autonomous docking for recharging
3. Charging Station
    - Autonomous docking station for recharging the AMR without manual intervention

---

3.2 Software Components

1. PC Software
    - Python3: Primary language for interface-related tasks
    - ROS2: Facilitates communication with the AMR
    - Flask: Provides a local web server for the monitoring dashboard
    - SQLite3: Used for lightweight data storage of alerts and logs
    - OpenCV and Ultralytics (YOLO): Optional setup for image processing if additional object detection is required on the PC
2. AMR Software
    - ROS2 (Robot Operating System 2): Core platform for navigation, control, and data integration
    - Python3: Main programming language for AMR operations and AI processing
    - OpenCV and Ultralytics (YOLO): Used for onboard object detection and threat analysis on the Jetson-Orin

---

4. Data Flow Design

1. Data Collection
    - AMR Sensors: The AMR collects data from LiDAR, camera, and ultrasonic sensors to perform autonomous navigation and detect potential threats.
2. Data Processing and Analysis
    - Onboard Processing: Video and sensor data are processed directly on the AMR using YOLO for object detection and OpenCV for image processing. The Jetson-Orin handles AI inference, identifying objects or anomalies in real time.
3. Data Storage
    - Local Storage on AMR: The AMR stores critical data locally for up to 48 hours.
    - PC Log Storage: Alerts and logs are stored on the monitoring PC using SQLite3 for review.
4. Alerting and Notification
    - Real-Time Alerts to PC: When a potential threat is detected, the AMR sends an alert to the monitoring PC via Wi-Fi.
    - Dashboard Display: The PC dashboard shows real-time video feed, alerts, and AMR status for security personnel to review.

---

5. Detailed Design

5.1 AMR Navigation and Threat Detection

- Navigation: Using ROS2 with SLAM (Simultaneous Localization and Mapping) and LiDAR, the AMR autonomously navigates the secure area, avoiding obstacles and following predefined routes.
- Threat Detection: YOLO-based object detection running on Jetson-Orin identifies unauthorized personnel, suspicious objects, or unusual movements.
- Communication Protocol: The AMR and PC communicate over a local Wi-Fi network, using a secure protocol (e.g., WebSocket) for low-latency data transmission.

5.2 User Interface (Dashboard)

- Monitoring Dashboard (Flask-based):
    - Displays real-time status of the AMR, live video feed, and any alerts.
    - Built with Flask and accessible via a browser on the PC, allowing security personnel to monitor and control the AMR.

    - **Manual Override:** Security personnel can take manual control if needed, using ROS2 commands sent from the PC.

---

6. Security Design

- **Data Encryption:** All data transmitted between the AMR and the PC uses TLS 1.3 to ensure secure communication.
- **Access Control:** The dashboard is secured with multi-factor authentication to restrict access to authorized personnel.
- **Local Storage Security:** Logs and alert data stored on the PC are encrypted with AES-256 to protect sensitive information.

---

7. Performance Requirements

- **Latency:** Alerts should reach the monitoring PC with less than 1 second of delay.
- **System Uptime:** The AMR should operate continuously with an 8-hour minimum battery life, returning to the charging dock when needed.
- **Scalability:** Designed for single-AMR operation; however, additional units can be added if required, with minimal adjustments to the architecture.

---

8. Error Handling and Recovery

- **Network Loss:** If Wi-Fi connectivity is lost, the AMR continues operating autonomously with limited functionality, storing any alerts locally until connectivity is restored.
- **Hardware Failure:** The AMR runs regular health checks on its sensors and hardware. If an issue is detected, it reports the error to the monitoring PC.
- **Battery Low:** The AMR autonomously returns to the docking station when battery levels are low.

---

9. Testing and Validation

1. **Unit Testing:** All software components, including ROS2 nodes and Flask server, are tested individually to verify functionality.

2. Integration Testing: The AMR and monitoring PC are tested as a cohesive system to ensure smooth data flow and command execution.
3. User Acceptance Testing (UAT): Conducted with security personnel to validate the dashboard, alerts, and manual control features.
4. Field Testing: Test the AMR in the target secure area to validate autonomous navigation, threat detection, and alerting under real conditions.

---

10. Deployment and Maintenance Plan

- Deployment Steps:
  - Install ROS2, Flask, and SQLite3 on the PC.
  - Configure the monitoring dashboard and connect the AMR via Wi-Fi.
  - Set up initial navigation routes and test alert functionality.
- Maintenance Schedule:
  - Monthly software updates on the AMR and PC for security patches.
  - Quarterly hardware checks on the AMR, including sensor calibration and battery testing.

---

11. Appendix

- Libraries and Dependencies: Detailed list of software dependencies, including ROS2, OpenCV, YOLO, Flask, and SQLite3.
- Glossary: Definitions of key terms such as AMR, SLAM, YOLO, etc.
- References: Documentation for ROS2, Jetson-Orin, TurtleBot3, and related technologies.

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.

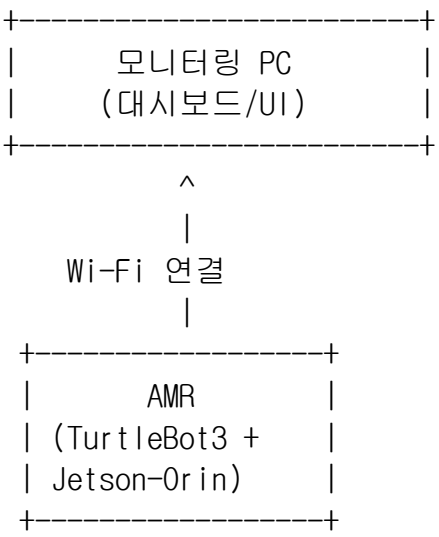
---

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.

2.1 고레벨 아키텍처 다이어그램

lua
Copy code



---

3. 구성 요소 설계

3.1 하드웨어 구성 요소

1. **PC (모니터링 시스템)**
    - **운영 체제:** Ubuntu 22.04
    - **카메라:** 로컬 시각적 모니터링을 위한 USB 카메라 (필요한 경우)
    - **네트워크:** AMR 과 안정적인 Wi-Fi 6 연결 지원
2. **자율 이동 로봇 (AMR) - TurtleBot3**
    - **프로세서:** Jetson-Orin (온보드 AI 처리 및 데이터 처리)
    - **운영 체제:** Ubuntu 22.04 및 ROS2 (로봇 제어)
    - **센서:**
        - **LiDAR:** 네비게이션 및 장애물 감지
        - **USB 카메라:** 위협 탐지용 비디오 피드 캡처
        - **초음파 센서:** 근거리 장애물 회피
    - **배터리:** 최소 8 시간 배터리 수명, 자율 도킹 및 충전 지원
3. **충전 스테이션**
    - 수동 개입 없이 AMR 의 충전을 위한 자율 도킹 스테이션

3.2 소프트웨어 구성 요소

1. **PC 소프트웨어**
    - **Python3:** 인터페이스 관련 작업을 위한 기본 언어
    - **ROS2:** AMR 과의 통신을 위한 플랫폼
    - **Flask:** 모니터링 대시보드를 위한 로컬 웹 서버
    - **SQLite3:** 경고 및 로그 기록을 위한 경량 데이터베이스
    - **OpenCV 및 Ultralytics (YOLO):** PC 에서 추가 객체 탐지가 필요한 경우의 이미지 처리 용도
2. **AMR 소프트웨어**
    - **ROS2 (로봇 운영 시스템 2):** 네비게이션, 제어 및 데이터 통합의 핵심 플랫폼
    - **Python3:** AMR 작동 및 AI 처리에 사용하는 기본 프로그래밍 언어
    - **OpenCV 및 Ultralytics (YOLO):** Jetson-Orin 에서 실행되는 객체 감지 및 위협 분석

---

4. 데이터 흐름 설계

1. **데이터 수집**
    - **AMR 센서:** AMR 은 LiDAR, 카메라, 초음파 센서를 사용하여 자율 네비게이션을 수행하고 잠재적 위협을 감지합니다.
2. **데이터 처리 및 분석**

    - **온보드 처리:** 비디오 및 센서 데이터는 AMR 에서 YOLO 를 사용해 로컬로 처리되어 객체 감지 및 이미지 분석이 이루어집니다. Jetson-Orin 이 AI 추론을 수행하여 실시간으로 객체나 이상 상태를 식별합니다.
3. **데이터 저장**
    - **AMR 로컬 저장:** 각 AMR 은 중요 데이터를 로컬에 48 시간 동안 저장합니다.
    - **PC 로그 저장:** 경고 및 로그가 모니터링 PC 의 SQLite3 에 저장되어 검토할 수 있습니다.
4. **경고 및 알림**
    - **실시간 경고 전송:** 잠재적 위협이 감지되면 AMR 은 Wi-Fi 를 통해 모니터링 PC 로 경고를 전송합니다.
    - **대시보드 표시:** PC 대시보드에서 보안 담당자는 실시간 비디오 피드, 경고 및 AMR 상태를 확인할 수 있습니다.

---

5. 상세 설계

5.1 AMR 네비게이션 및 위협 탐지

- **네비게이션:** ROS2 와 SLAM(동시 위치 추정 및 매핑) 및 LiDAR 를 사용하여 AMR 이 보안 구역 내에서 자율적으로 이동하고 장애물을 회피합니다.
- **위협 탐지:** Jetson-Orin 에서 실행되는 YOLO 기반 객체 감지를 통해 무단 접근자, 의심스러운 물체 또는 이상 행동을 식별합니다.
- **통신 프로토콜:** AMR 과 PC 는 로컬 Wi-Fi 네트워크를 통해 WebSocket 과 같은 보안 프로토콜로 데이터 전송을 수행합니다.

5.2 사용자 인터페이스 (대시보드)

- **모니터링 대시보드 (Flask 기반):**
    - AMR 의 실시간 상태, 라이브 비디오 피드, 경고를 표시합니다.
    - Flask 로 구축되었으며 PC 의 브라우저에서 접근할 수 있어 보안 담당자가 AMR 을 모니터링하고 제어할 수 있습니다.
    - **수동 제어:** 보안 담당자는 필요시 ROS2 명령을 통해 수동으로 AMR 을 제어할 수 있습니다.

---

6. 보안 설계

- **데이터 암호화:** AMR 과 PC 간 전송되는 모든 데이터는 TLS 1.3 을 사용해 암호화됩니다.

- **접근 제어:** 대시보드는 다중 요소 인증을 통해 접근이 제한되어 있습니다.
- **로컬 저장 보안:** PC 에 저장된 로그 및 경고 데이터는 AES-256 을 사용해 암호화하여 민감한 정보를 보호합니다.

---

7. 성능 요구사항

- **지연 시간:** 경고는 1 초 미만의 지연 시간으로 모니터링 PC 에 도달해야 합니다.
- **시스템 가동률:** AMR 은 최소 8 시간 동안 연속 운영이 가능해야 하며, 필요시 충전 독으로 복귀할 수 있어야 합니다.
- **확장성:** 단일 AMR 운영에 맞춰 설계되었으나, 추가 유닛이 필요할 경우 최소한의 아키텍처 변경으로 지원 가능합니다.

---

8. 오류 처리 및 복구

- **네트워크 손실:** Wi-Fi 연결이 끊기면 AMR 은 제한된 기능으로 자율적으로 작동을 유지하며, 연결이 복원되면 데이터 전송을 재개합니다.
- **하드웨어 오류:** AMR 은 정기적으로 센서 및 하드웨어 상태를 점검합니다. 문제가 감지되면 모니터링 PC 에 오류를 보고합니다.
- **배터리 부족:** 배터리가 낮을 경우 AMR 은 자율적으로 도킹 스테이션으로 복귀합니다.

---

9. 테스트 및 검증

1. **단위 테스트:** ROS2 노드 및 Flask 서버를 포함한 모든 소프트웨어 구성 요소를 개별적으로 테스트하여 기능을 검증합니다.
2. **통합 테스트:** AMR 과 모니터링 PC 를 통합하여 원활한 데이터 흐름과 명령 실행을 보장합니다.
3. **사용자 승인 테스트(UAT):** 보안 담당자와 함께 대시보드, 경고 및 수동 제어 기능을 실제 상황에서 검증합니다.
4. **현장 테스트:** 목표 보안 구역에서 AMR 을 테스트하여 자율 네비게이션, 위협 탐지 및 경고 기능을 실제 조건에서 검증합니다.

---

10. 배포 및 유지보수 계획

- **배포 단계:**

    - PC 에 ROS2, Flask, SQLite3 를 설치합니다.
    - 모니터링 대시보드를 구성하고 AMR 을 Wi-Fi 로 연결합니다.
    - 초기 네비게이션 경로를 설정하고 경고 기능을 테스트합니다.
- **유지보수 일정:**
    - 매월 AMR 과 PC 의 소프트웨어 업데이트로 보안 패치 적용
    - 분기마다 AMR 의 하드웨어 점검, 센서 보정 및 배터리 검사

---

11. 부록

- **라이브러리 및 종속성:** ROS2, OpenCV, YOLO, Flask, SQLite3 등 사용된 소프트웨어 종속성 목록
- **용어 사전:** AMR, SLAM, YOLO 등 주요 용어의 정의
- **참고 문헌:** ROS2, Jetson-Orin, TurtleBot3 및 관련 기술에 대한 문서