**시스템 설계 문서 (SDD)**

**프로젝트 제목:** 멸종 위기동물 탐지 및 추적 관찰 시스템  
**버전:** 1.0  
**날짜:** 24.12.30

**1. 개요**

자율 이동 로봇(AMR) 동물 추적 관찰 시스템은 단일 AI 기반 로봇을 사용하여 광범위한 구역에서 멸종위기동물 탐지 및 추적 관찰을 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.

**2.1 고레벨 아키텍처 다이어그램**

포스트잇 노트, 텍스트, 친필, 종이 제품이(가) 표시된 사진

자동 생성된 설명

**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:** AMR과 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:** 동물 탐지용 비디오 피드 캡처
     + **초음파 센서:** 근거리 장애물 회피
   * **배터리:** 최소 8시간 배터리 수명, 자율 도킹 및 충전 지원
3. **충전 스테이션**
   * 수동 개입 없이 AMR의 충전을 위한 자율 도킹 스테이션

**3.2 소프트웨어 구성 요소**

1. **PC 소프트웨어**
   * **Python3:** 인터페이스 관련 작업을 위한 기본 언어
   * **ROS2:** AMR과의 통신을 위한 플랫폼
   * **Flask:** 모니터링 대시보드를 위한 로컬 웹 서버
   * **SQLite3:** 경고 및 로그 기록을 위한 경량 데이터베이스
   * **OpenCV 및 Ultralytics (YOLO):** PC의 관측카메라에서 동물 포착 및 좌표 계산 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-Orin에서 실행되는 객체 감지 및 목표와의 거리 추정

**4. 데이터 흐름 설계**

1. **데이터 수집**
   * **PC 카메라:** PC는 카메라를 동물을 감지하고 해당 동물의 좌표를 계산합니다.
   * **AMR 센서:** AMR은 LiDAR, 카메라, 초음파 센서를 사용하여 자율 네비게이션을 수행하고 목표 동물을 추적, 관찰합니다.
2. **데이터 처리 및 분석**
   * **온보드 처리:** 비디오 및 센서 데이터는 pc와 AMR에서 YOLO를 사용해 로컬로 처리되어 객체 감지 및 이미지 분석이 이루어집니다. PC프로세서와 Jetson-Orin이 AI 추론을 수행하여 실시간으로 객체나 이상 상태를 식별합니다.
3. **데이터 저장**
   * **AMR 로컬 저장:** 각 AMR은 중요 데이터를 로컬에 48시간 동안 저장합니다.
   * **PC 로그 저장:** 데이터 및 로그가 모니터링 PC의 SQLite3에 저장되어 검토할 수 있습니다.

**5. 상세 설계**

**5.1 AMR 네비게이션 및 동물 탐지**

* **네비게이션:** ROS2와 SLAM(동시 위치 추정 및 매핑) 및 LiDAR를 사용하여 AMR이 관측 구역 내에서 자율적으로 이동하고 장애물을 회피합니다.
* **동물 탐지:** Jetson-Orin에서 실행되는 YOLO 기반 객체 감지를 통해 목표 동물을 식별하여 추적관찰합니다.
* **통신 프로토콜:** AMR과 PC는 로컬 Wi-Fi 네트워크를 통해 WebSocket과 같은 보안 프로토콜로 데이터 전송을 수행합니다.

**5.2 사용자 인터페이스 (대시보드)**

* **모니터링 대시보드 (Flask 기반):**
  + AMR의 실시간 상태, 라이브 비디오 피드를 표시합니다.
  + Flask로 구축되었으며 PC의 브라우저에서 접근할 수 있어 보안 담당자가 AMR을 모니터링하고 제어할 수 있습니다.
  + **수동 제어:** 관측 담당자는 필요시 ROS2 명령을 통해 수동으로 AMR을 제어할 수 있습니다.

**7. 성능 요구사항**

* **지연 시간:** 경고는 1초 미만의 지연 시간으로 모니터링 PC에 도달해야 합니다.
* **시스템 가동률:** AMR은 최소 8시간 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 AMR은 제한된 기능으로 자율적으로 작동을 유지하며, 연결이 복원되면 데이터 전송을 재개합니다.
* **하드웨어 오류:** AMR은 정기적으로 센서 및 하드웨어 상태를 점검합니다. 문제가 감지되면 모니터링 PC에 오류를 보고합니다.
* **배터리 부족:** 배터리가 낮을 경우 AMR은 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:** AMR과 모니터링 PC를 통합하여 원활한 데이터 흐름과 명령 실행을 보장합니다.
3. **사용자 승인 테스트(UAT):** 관측 담당자와 함께 대시보드, 탐지 및 수동 제어 기능을 실제 상황에서 검증합니다.
4. **현장 테스트:** 목표 관측 구역에서 AMR을 테스트하여 자율 네비게이션, 동물 탐지 및 추적 기능을 실제 조건에서 검증합니다.

**10. 배포 및 유지보수 계획**

* **배포 단계:**
  + PC에 ROS2, Flask, SQLite3를 설치합니다.
  + 모니터링 대시보드를 구성하고 AMR을 Wi-Fi로 연결합니다.
  + 초기 네비게이션 경로를 설정하고 추적 기능을 테스트합니다.
* **유지보수 일정:**
  + 분기마다 AMR의 하드웨어 점검, 센서 보정 및 배터리 검사
  + 매년 AMR과 PC의 소프트웨어 업데이트로 안정성 패치 적용

**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