자원 관리 계획 및 품질 관리 계획

# 1. 자원 관리 계획

|  |  |
| --- | --- |
| 자원 항목 | 계획 및 관리 내용 |
| 개발자 인력 | 주요 기능을 구현할 수 있는 Python 및 ROS 숙련자 확보 |
| 테스터 인력 | 시스템을 실제 환경에서 검증할 인력 배치 |
| 로봇 하드웨어 | XYCAR 및 USB 카메라, 모터 제어 장치 확보 및 점검 |
| 소프트웨어 라이브러리 | OpenCV, ROS, Matplotlib 등 라이브러리 설치 및 의존성 관리 |
| 작업 공간 | 안전하고 충분한 공간에서 테스트 수행 |
| 컴퓨팅 자원 | 고성능 노트북 또는 워크스테이션 확보 |
| 네트워크 연결 | ROS 노드 통신을 위한 안정적인 로컬 네트워크 환경 유지 |
| 전원 공급 장치 | 지속적인 테스트를 위한 안정적 전원 확보 |
| 백업 시스템 | 코드 및 데이터의 정기적 백업 수행 |
| 문서화 도구 | Sphinx, Google Docs, Word 등 문서화 도구 활용 |
| 프로젝트 관리 도구 | Trello, Jira 등을 통한 작업 분배 및 일정 관리 |
| 시뮬레이션 도구 | Gazebo 또는 RViz와 같은 시뮬레이터 활용 |
| 교통 표지판 모델 | 테스트용 apriltag 또는 시각 표지판 확보 |
| 데이터 저장 장치 | 로그, 이미지, 센서 데이터를 위한 저장 공간 확보 |
| 디버깅 도구 | 실시간 로깅 및 디버깅을 위한 rqt, rostopic 등 도구 사용 |
| 안전 장비 | 테스트 중 안전 확보를 위한 보호 장비 구비 |
| 스페어 부품 | 모터, 휠, 센서 등 소모성 부품 예비 확보 |
| 교육 자료 | 참여자 교육을 위한 매뉴얼 및 가이드 문서 준비 |
| 외부 컨설턴트 | 필요 시 외부 전문가 자문 확보 |
| 보안 관리 | 코드 접근 제어 및 장비 보안 조치 수행 |

# 2. 품질 관리 계획

|  |  |
| --- | --- |
| 품질 항목 | 관리 방안 |
| 코드 정합성 | PEP8 등 코딩 표준 준수 |
| 기능 정확성 | 요구사항에 따른 테스트 케이스 통과 여부 확인 |
| 성능 최적화 | 주행 지연 최소화를 위한 연산 최적화 |
| 에러 처리 | 예외 발생 시 시스템 안전 정지 및 로그 출력 |
| 시각화 품질 | 플롯 갱신 및 직관적 표현 확인 |
| 센서 신뢰도 | 태그 인식 정확도 및 거리 추정 정확도 점검 |
| 시스템 반응성 | 실시간 반응 시간 측정 및 튜닝 |
| UI/UX | 터미널 및 시각 피드백 구성 점검 |
| 테스트 범위 | 다양한 환경 및 각도에서의 인식 정확성 평가 |
| 회귀 테스트 | 업데이트 시 기존 기능 재검증 |
| 버전 관리 | git 등을 활용한 안정적 코드 관리 |
| 문서화 | 개발/설치/운영 문서 제공 |
| 라이브러리 검증 | 사용된 오픈소스 라이브러리의 안정성 확인 |
| 하드웨어 연동 검증 | 모터 제어, 센서 연결 테스트 |
| 시나리오 기반 검증 | 실제 주행 조건 기반 테스트 |
| 환경 적응성 | 조도, 위치 변화 등에 따른 안정성 검증 |
| 결함 추적 | 버그 발생 시 즉시 기록 및 해결 추적 |
| 정기 리뷰 | 주간 코드 리뷰 및 품질 회의 |
| 사용자 피드백 | 사용자 관찰 및 의견 수집 |
| 자동화 테스트 | 가능한 테스트는 스크립트화하여 반복 수행 |