**설계 요건:**

**라인 감지 센서**: 라인을 신속하고 정확하게 감지하는 센서가 필요합니다. 일반적으로 광센서, 적외선 센서 또는 레이저 센서를 사용합니다. 여기서 저희가 사용할 센서는 적외선 센서입니다. 적외선 센서를 2개를 앞뒤에 두고 만약 라인을 이탈하면 특정한 알고리즘을 동작하도록 구성하게 됩니다.

**모터 및 구동 장치**: 라인 트레이서가 움직일 수 있도록 모터와 관련 구동 장치가 필요합니다. 이를 통해 로봇은 라인을 따라 이동하거나 조향할 수 있습니다. 여기서 저희 팀은 스텝 모터를 사용할 예정입니다.

**제어 시스템**: 라인트레이서의 동작을 제어하는 아두이노가 필요합니다. 이를 통해 센서 데이터를 처리하고 모터를 제어합니다. 저희 팀은 아두이노 우노를 사용합니다

**프로그래밍 환경**: 라인트레이서 동작을 프로그래밍하고 수정할 수 있는 프로그래밍 환경이 필요합니다. 저희 팀은 스케치 ide를 사용합니다.

**전원 공급**: 라인트레이서를 구동하기 위한 전원 공급 방법이 필요하며, 배터리 또는 전원 어댑터를 사용할 수 있습니다. 저희 팀은 건전지를 사용할 예정입니다.

**안전 기능**: 라인트레이서기 다른 물체나 사람과 충돌하지 않도록 안전 기능이 필요하며, 이를 통해 충돌을 감지하고 피하는 기능이 있어

야 합니다. 이는 보통 초음파 센서로 구현됩니다.

**제약점:**

* **환경 제약**: 라인 트레이서는 특정 환경에서만 작동하므로 환경 변화에 대처할 수 있는 유연성이 제한됩니다. 즉 라인이 없거나 끊길시 문제가 생깁니다.
* **라인 굴곡과 곡선**: 라인이 굴곡 또는 곡선을 가질 경우, 로봇이 정확하게 따라가는 것이 어려울 수 있으며, 이로 인해 정확성에 제약이 생길 수 있습니다.
* **센서 및 하드웨어 제약**: 사용하는 센서의 해상도, 성능, 감도, 모터의 속도, 회전 반경 및 정확성 등이 성능에 영향을 미칩니다. 이러한 하드웨어의 제약은 라인 추적의 정확성과 속도에 영향을 미칩니다.
* **전원 공급 및 배터리 수명**: 로봇을 구동하기 위한 전원 공급은 배터리 수명, 충전 시간, 용량 등에 대한 제약을 가집니다.
* **프로그래밍 복잡성**: 라인 트레이서를 프로그래밍하고 조정하는 것은 복잡할 수 있으며, 신중한 조정과 테스트가 필요합니다.

라인트레이서 구현사례

* **아두이노 기반 라인 트레이서:**
* **하드웨어:** 아두이노 (예: Arduino Uno 또는 Arduino Nano), 적외선센서 모듈 (IR Sensor Module), DC 모터, 바퀴, 배터리 팩 등
* **소프트웨어:** 아두이노 IDE (C/C++ 언어)
* **설명:** 아두이노와 적외선센서를 사용하여 라인 트레이서를 만드는 프로젝트입니다. 적외선센서로 라인을 감지하고 모터를 제어하여 라인을 따라 움직입니다. 아두이노 IDE를 사용하여 프로그램을 개발하고 업로드합니다.
* **고급 라인 트레이서 프로젝트:**
* **하드웨어:** 아두이노, 적외선 라인 트레이서 센서 배열 (Line Follower Sensor Array), 모터 및 드라이버, 휠, 라즈베리 파이 (옵션)
* **소프트웨어:** 아두이노 IDE, 라즈베리 파이 (옵션)
* **설명:** 이 프로젝트는 고급 라인 트레이서로, 더 많은 센서를 사용하여 정교한 라인 추적을 수행합니다. 또한, 라즈베리 파이를 사용하여 라인 트레이서와 통신하거나 추가적인 기능을 제공할 수 있습니다.
* **무선 제어 라인 트레이서:**
* **하드웨어:** 아두이노, 라즈베리 파이, 무선 통신 모듈 (예: Bluetooth 또는 Wi-Fi 모듈), 적외선센서, 모터 및 드라이버, 휠
* **소프트웨어:** 아두이노 IDE, 라즈베리 파이, 무선 통신 라이브러리
* **설명:** 이 프로젝트에서는 아두이노를 사용하여 무선으로 제어되는 라인 트레이서를 구현합니다. 무선 통신 모듈을 사용하여 아두이노와 라즈베리 파이 사이의 통신을 설정하고 스마트폰 앱 또는 컴퓨터를 통해 로봇을 제어할 수 있습니다.

-자세한 사례

적외선 센서를 앞에 2개 달고 만약 라인을 이탈하는 즉 왼쪽센서가 하얀색을 감지하면 오른쪽으로 꺾고 반대의 경우는 왼쪽으로 꺾는게 보통의 구현 사례였습니다. 이 센서의 방향에 따른 개수를 늘릴경우 정밀도는 향상될 것입니다. 만일 라인의 갈림길이 많을 경우의 사례를 고려하였을때 경로를 먼저 dfs알고리즘으로 탐색한후 이를 맵핑하여 다익스트라 알고리즘으로 구현한 사례가 있었습니다. (남서울대 랩실)