전상환 교수님

10376 공학입문설계

Arduino를 활용한 Line Tracer에 대한 이해와 고찰

6조 종박사님을 아세요

[목차]

[PLAN]

1. 제작 계획

2. 아두이노 Uno에 의해 작동하는 H/W 제작 계획

3. 아두이노 우노와 센서, 액츄에이터를 구조물에 설치하는 방법

4. H/W를 작동시키기 위한 Sketch 작성 계획

[CREATE]

1. 기능성과 심미성의 관점에서 제작 방향에 대한 설명

2. Plan 단계에서 수립한 제작 계획을 바탕으로 성과물 제작 과정에 대한 설명

[IMPROVE]

1. 설계물 시험 평가 결과
   가. 결함/오류 유발 요인에 대한 분석, 수정 보완 과정
2. 최종 성과물에 반영된 수정 보완 사항의 설명
   가. H/W 설명
   - 설계물 사진, 사진에 대한 설명

- 설계물 작동 동영상(스트리밍 서버 주소)과 설명

나. S/W 설명

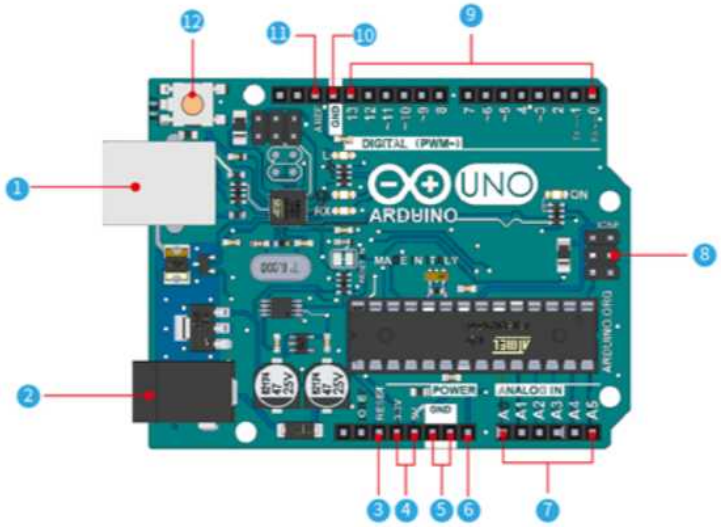
- 스케치에 대한 수정 과정의 상세한 설명

[ASK]

1. 설계 요건 및 제약점 정리

설계 요건 : 주어진 교재에 있는 재료(센서, 액츄에이터 등)을 사용하여 제시된 라인을 따라 장애물을 피하는 Line Tracer을 제작한다.

1. 라인 감지 센서 선택 : 라인을 신속하고 정확하게 감지하는 센서가 필요하다. 일반적으로 광 센서, 적외선 센서 또는 레이저 센서를 사용합니다. 이번 프로젝트에서 우리 팀이 사용할 센서는 적외선 센서와 초음파 센서이다. 적외선 센서를 2개를 앞뒤에 두고 만약 라인을 이탈하면 특정한 알고리즘을 동작하도록 구성한다.
2. 모터 및 구동 장치 선택 : Line tracer가 움직일 수 있도록 모터와 관련 구동 장치가 필요하다. 이를 통해 로봇은 라인을 따라 이동하거나 조향할 수 있습니다. 이번 프로젝트에서 우리 팀이 사용할 모터 및 구동 장치는 스텝 모터이다.
3. 액츄에이터 선택 : 센서에서 받아들인 정보를 통해 적절히 반응할 액츄에이터를 선택한다. 이번 프로젝트에서 우리 팀이 사용할 액츄에이터는 피에조 스피커, LED, 텍스트 LCD 등이다.
4. 로봇 구조 설계 : Line Tracer 로봇의 물리적 및 외적 구조를 설계한다. 모터, 바퀴, 프레임 등을 고려하여 로봇의 모양과 크기를 결정한다.
5. Arduino 보드 선택 : Line Tracer의 센서에서 받아들인 정보에 어떻게 반응할지 결정하여 액츄에이터에 정보를 전달하고, Line Tracer의 동작을 제어하는 Arduino 보드를 선택한다. 이번 프로젝트에서 우리 팀이 사용할 Arduino 보드는Arduino Uno이다.
6. 프로그래밍 환경 : Line Tracer의 동작을 제어하는 프로그램을 작성하고 수정할 수 있는 프로그래밍 환경이 필요하다. 이번 프로젝트에서 우리가 사용할 프로그래밍은 라인을 따라가는 알고리즘, 최적화 알고리즘 등이고, 프로그래밍 환경은 스케치 IDE이다.
7. 전원 공급 : Line Tracer를 구동하기 위한 전원 공급 방법이 필요하며, 배터리 또는 전원 어댑터를 사용한다. 우리 팀이 사용할 전원 공급 방법은 18650 배터리 및 배터리 홀더이다.



제약점 : 주어진 교재의 제약점과 교재 내의 부품의 제약점이 있다.

주어진 교재의 제약점을 먼저 살펴보면 다음과 같다.

1. 모든 조에 동일하게 주어진 교재에 있는 재료만을 사용해야 한다. Arduino Uno, 적외선 센서, 피에조 스피커 등 이외의 재료를 사용해서는 안 된다.

2. 모든 프로젝트가 끝난 후, 사용한 교재는 다시 분해가 가능해야 한다. 부품을 접합하기 위하여 강한 접착제를 사용하거나 납땜을 해서는 안 된다.

3. 만약 교재에 납땜이나 접착 등이 필요할 경우, 교수님과 사전에 협의가 있어야 한다. 조 내에서 독단적으로 결론을 내려 실행해서는 안 된다.

두 번째 제약점은 교재 내의 부품 및 프로그래밍의 제약점으로, 이는 다음과 같다.

1. 환경 제약 : Line Tracer는 특정 환경에서만 작동하므로 환경 변화에 대처할 수 있는 유연성이 제한된다. 즉 라인이 없거나 끊길 시 문제가 생긴다. 또한 라인이 굴곡 또는 곡선을 가질 경우 정확하게 따라가는 것이 어려울 수 있어 이로 인해 정확성에 제약이 생길 수 있다.
2. 센서 및 하드웨어 제약 : 사용하는 센서의 해상도, 성능, 감도, 모터의 속도, 회전반경 및 정확성 등이 성능에 영향을 미친다. 이러한 하드웨어의 제약은 라인 추적의 정확성과 속도에 영향을 미친다.
3. 전원 공급 및 배터리 수명 : 로봇을 구동하기 위한 전원 공급은 배터리 수명, 충전 시간, 용량 등에 제약이 있다.
4. 프로그래밍 복잡성 : Line Tracer를 프로그래밍하고 조정하는 것은 복잡할 수 있으며, 신중한 조정과 테스트가 필요하다.

2. 자료 조사 분석

가. Arduino를 활용한 Line Tracer 제작 사례 및 부품 조사 분석

- 사례 조사 및 주행 특성 분석

1. Arduino 기반 Line tracer :

하드웨어 : Arduino (예: Arduino Uno 또는 Arduino Nano), 적외선 센서 모듈 (IR Sensor Module), DC 모터, 바퀴, 배터리 팩 등

소프트웨어 : Arduino IDE (C/C++ 언어)

설명 : Arduino와 적외선 센서를 사용하여 Line tracer를 만드는 프로젝트입니다. 적외선 센서로 라인을 감지하고 모터를 제어하여 라인을 따라 움직입니다. Arduino IDE를 사용하여 프로그램을 개발하고 업로드한다.

2. 고급 Line tracer 프로젝트 :

하드웨어 : Arduino, 적외선 Line tracer 센서 배열 (Line Follower Sensor Array), 모터 및 드라이버, 휠, 라즈베리 파이 (옵션)

소프트웨어 : Arduino IDE, 라즈베리 파이 (옵션)

설명 : 이 프로젝트는 고급 Line tracer로, 더 많은 센서를 사용하여 정교한 라인 추적을 수행한다. 또한, 라즈베리 파이를 사용하여 Line tracer와 통신하거나 추가적인 기능을 제공할 수 있다.

3. 무선 제어 Line tracer :

하드웨어 : Arduino, 라즈베리 파이, 무선 통신 모듈 (예: Bluetooth 또는 Wi-Fi 모듈), 적외선센서, 모터 및 드라이버, 휠

소프트웨어 : Arduino IDE, 라즈베리 파이, 무선 통신 라이브러리

설명 : 이 프로젝트에서는 Arduino를 사용하여 무선으로 제어되는 Line tracer를 구현합니다. 무선 통신 모듈을 사용하여 Arduino와 라즈베리 파이 사이의 통신을 설정하고 스마트폰 앱 또는 컴퓨터를 통해 로봇을 제어할 수 있다.

이외의 논문 사례 :

적외선 센서를 앞에 2개 달고 만약 라인을 이탈하는, 즉 왼쪽 센서가 하얀색을 감지하면 오른쪽으로 꺾고 반대의 경우는 왼쪽으로 꺾는 게 보통의 구현 사례였다. 이 센서의 방향에 따른 개수를 늘릴 경우 정밀도는 향상될 것이다. 만일 라인의 갈림길이 많을 경우의 사례를 고려하였을 때 경로를 먼저 DFS알고리즘으로 탐색한후 이를 맵핑하여 다익스트라 알고리즘으로 구현한 사례가 있다.

주행 특성 분석 : Line tracer 주행 특성 및 센서 Line tracer는 바닥에 그려진 선을 따라서 움직이는 로봇이다. Line tracer는 적외선과 초음파를 이용하여 선의 유무를 감지하고, 주변의 장애물들을 인식하여 피해가며 모터를 제어하여 알맞은 방향으로 주행한다. Line tracer의 구성은 크게 센서부, 제어부, 모터부로 나눌 수 있다.

우선, 센서부는 적외선, 초음파 발신부와 수신부로 이루어져 있으며, 적외선을 바닥으로 쏘아 반사되는 빛의 양으로 선의 색상을 구분한다. 적외선 라인 추적 센서는 바닥에 그려진 선을 감지하는 역할을 한다. 센서의 개수와 간격에 따라 선의 곡률과 각도를 판단할 수 있다. Line tracer는 적외선 센서가 흰색과 검은색을 구분할 수 있도록 바닥과 선의 색상이 눈에 띄게 대비되게 해야 한다. 초음파 라인 추적 센서는 사람이 들을 수 있는 20Hz에서 20KHz(20,000Hz)까지의 가청주파수를 벗어난 40KHz의 사운드를 출력하고 그것이 어떤 물체에 부딪혀서 돌아오는 시간을 재서 거리를 감지한다. 초음파 센서는 들어있는 부품마다 조금씩 다르지만 3~40도 정도의 센싱 각도를 가지고 있어 초음파를 발사하면 음파이기 때문에 소리가 옆으로 퍼지게 되어 옆에 있는 물체도 센싱이 가능하다. 반면 적외선 센서는 적외선이 반사돼서 돌아오는 각도를 재다 보니 정확하게 앞에 있는 오브젝트만 센싱을 할 수 있다. 그리고 센서의 감도를 가변저항으로 조절할 수 있다. 참고로 감도가 너무 낮으면 선을 인식하지 못하고, 너무 높으면 주변의 잡음에 영향을 받을 수 있으므로 적절하게 조절하는 것이 중요하다. 다음으로, 제어부는 센서부로부터 입력된 값을 연산하고, 이를 바탕으로 모터부에게 명령을 내리는 역할을 한다. 제어부에는 Arduino와 같은 마이크로 컨트롤러가 사용되는데, 이는 센서와 모터를 연결하고 제어하는 역할을 한다. 프로그래밍을 통해 로봇의 동작을 결정할 수 있다. 프로그래밍에서는 센서의 신호를 읽고, 모터에게 명령을 내리고, 로봇의 상태를 확인하는 등의 작업을 할 수 있다. 마지막으로 모터부는 제어부가 내린 명령에 따라 모터의 속도와 방향을 조절하여 로봇의 움직임을 결정한다. DC 모터와 스텝핑 모터를 사용하는데, 이들은 로봇의 바퀴를 구동하는 역할을 한다. 모터의 속도와 방향을 제어하여 로봇의 회전과 직진을 할 수 있다. DC모터는 간단하게 회전하고, 효율이 좋고, 출력에 비해 소형 경량이라는 장점이 있지만 속도를 제어해야 하고 위치제어가 필요하며, 잡음이 발생하는 단점이 있다. 스텝핑 모터는 디지털 신호로 직접 오픈루프 제어를 할 수 있고, 시스템 전체가 간단하며 기동, 정지, 정-역회전 변속이 용이하며 응답특성도 좋다는 장점이 있다. 하지만 특정 주파수에

서 진동, 공진 현상이 발생하기 쉽고, 관성이 있는 부하에 약하며 고속 운전시 탈조하기 쉽다는 단점이 있다. 한편 모터의 속도는 PWM 신호로 조절할 수 있다. PWM 신호는 아날로그 신호처럼 연속적으로 변하지 않고, 디지털 신호처럼 0과 1만 가지는 신호다. PWM 신호는 0과 1이 반복되는 주기와 1이 차지하는 비율에 따라 평균 전압을 조절할 수 있다.

Line tracer를 제작하기 위해서는 다음과 같은 과정이 필요하다. 첫째, 바디를 설계하고 제작한다. 바디는 로봇의 크기와 무게, 안정성 등을 고려하여 만들어야 한다. 둘째, 센서부, 제어부, 모터부를 바디에 연결하고 배선한다. 연결 방법은 각 부품의 사양과 회로도를 참고하여 정확하게 해야 한다. 셋째, Arduino에 Line tracer를 제어하는 프로그램을 작성하고 업로드한다. 프로그램은 센서값을 읽고, 모터를 제어하는 알고리즘을 포함해야 한다. 마지막으로 Line tracer를 테스트하고 성능을 개선해야 한다. 테스트를 위해 Line tracer 트랙을 준비하고, 감도나 모터의 속도 등을 조절하면 제작 과정이 끝나게 된다.

- Sensor, Actuator 분석

액츄에이터 : 디지털 또는 아날로그 신호를 통해 특정 기능을 동작시키는 것

액츄에이터의 종류 : LED, 삼색 LED, 피에조 스피커, 7세그먼트, LCD, 서보 모터, 릴레이

1. LED : 전원을 연결하면 한 가지 색으로 빛난다.
2. 삼색 LED : RGB 값을 설정해 다양한 색으로 빛난다.
3. 피에조 스피커 : 간단한 노래나 효과음 등을 재생할 수 있다.
4. 7 세그먼트 : 7개로 나뉘어진 LED로 숫자 또는 글자를 나타낼 수 있다.
5. LCD : 글자나 숫자, 또는 그림 등 다양한 것을 표시할 수 있다.
6. 서보 모터 : 각도를 조절할 수 있는 모터, 로봇 팔과 같은 것을 만들 수 있다.
7. 릴레이 : 전기를 연결하거나 끊을 수 있다.

액츄에이터(모터) 사용 방법:

1. 모터 선택 : 모터 선택 시, 토크, RPM, 전압 등을 고려하여 프로젝트에 맞는 모터를 선택해야 한다.
2. 전원 공급 : 모터에 충분한 전원을 공급해야 한다. 일반적으로 사용 전압 범위를 확인하고 배터리 또는 전원 공급 장치를 선택해야 한다.

3. 모터 드라이버 연결 : 모터 드라이버를 사용하여 모터를 제어하므로, 모터 드라이버와 Arduino 간에 연결을 설정해야 한다.
4. PWM 제어 : 모터 드라이버를 통해 모터의 속도와 방향을 제어하는 데 PWM (펄스 폭 변조) 신호를 사용해야 한다.
5. 모터 속도 및 방향 제어 : Arduino 코드를 통해 모터의 속도와 방향을 제어해야 한다. 액츄에이터가 라인 추적 작업을 수행하도록 프로그래밍해야 한다.

액츄에이터(모터) 사용 시 유의사항 :

1. 전원 공급 안정성 : 충분한 전원을 제공하여 모터가 정상적으로 작동하도록 해야 한다. 전류 공급 능력을 고려할 필요가 있다.
2. 과열 방지 : 모터가 과열되지 않도록 모터의 작동 시간과 부하를 감안하여 적절한 냉각을 제공해야 한다.
3. 모터 드라이버 보호 : 모터 드라이버를 사용하여 모터를 제어할 때, 과전류 및 과전압 보호를 구현해야 한다.
4. 모터지지 : 모터가 로봇의 구조에 안전하게 고정되었는지 확인해야 한다. 진동 및 충격을 완화할 수 있는 적절한 지지 구조를 고려해야 한다.
5. 소음과 진동 관리 : 모터는 소음과 진동을 발생시킬 수 있으므로 이를 관리하기 위한 조치가 필요하다.
6. 모터 드라이버 설정 : 모터 드라이버의 설정을 확인하여 모터와 드라이버가 일치하는지 확인해야 한다. 모터의 전압 및 전류 요구 사항을 파악해야 한다.
7. 작동 시 안전 : 로봇이 장애물을 피하고 사용자나 환경을 피해 안전하게 작동하도록 프로그래밍해야 한다.

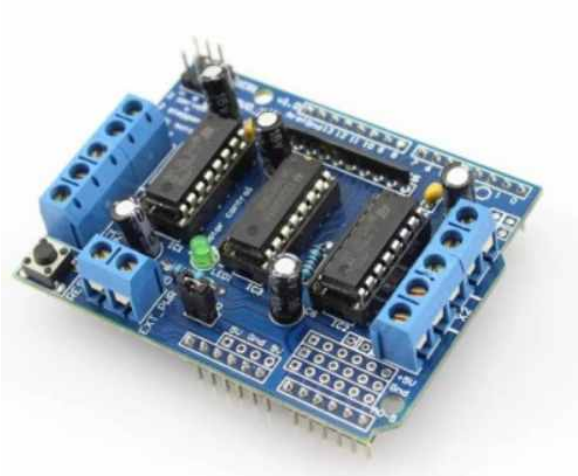
나. Motor shield 사용 방법

- 사용 방법 및 유의점, 스케치의 적용, 결선 방법

Motor shield 사용 방법 :

1. 모터 쉴드 선택: 프로젝트에 적합한 모터 쉴드를 선택해야 한다. 모터 쉴드에는 다양한 모델이 있으며, 모터의 수 및 전압 요구 사항에 따라 선택을 한다.
2. 모터 쉴드 연결: Arduino 보드와 모터 쉴드를 연결한다. 모터 쉴드는 Arduino의 핀을 덮어주고, 보드 위에 쌓을 수 있도록 설계되어 있다.
3. 모터 연결: 모터를 모터 쉴드의 모터 드라이버에 연결한다. 일반적으로 각 모터 드라이버에는 두 개의 모터 핀이 있다. 모터의 전원 및 접지도 연결해야 한다.

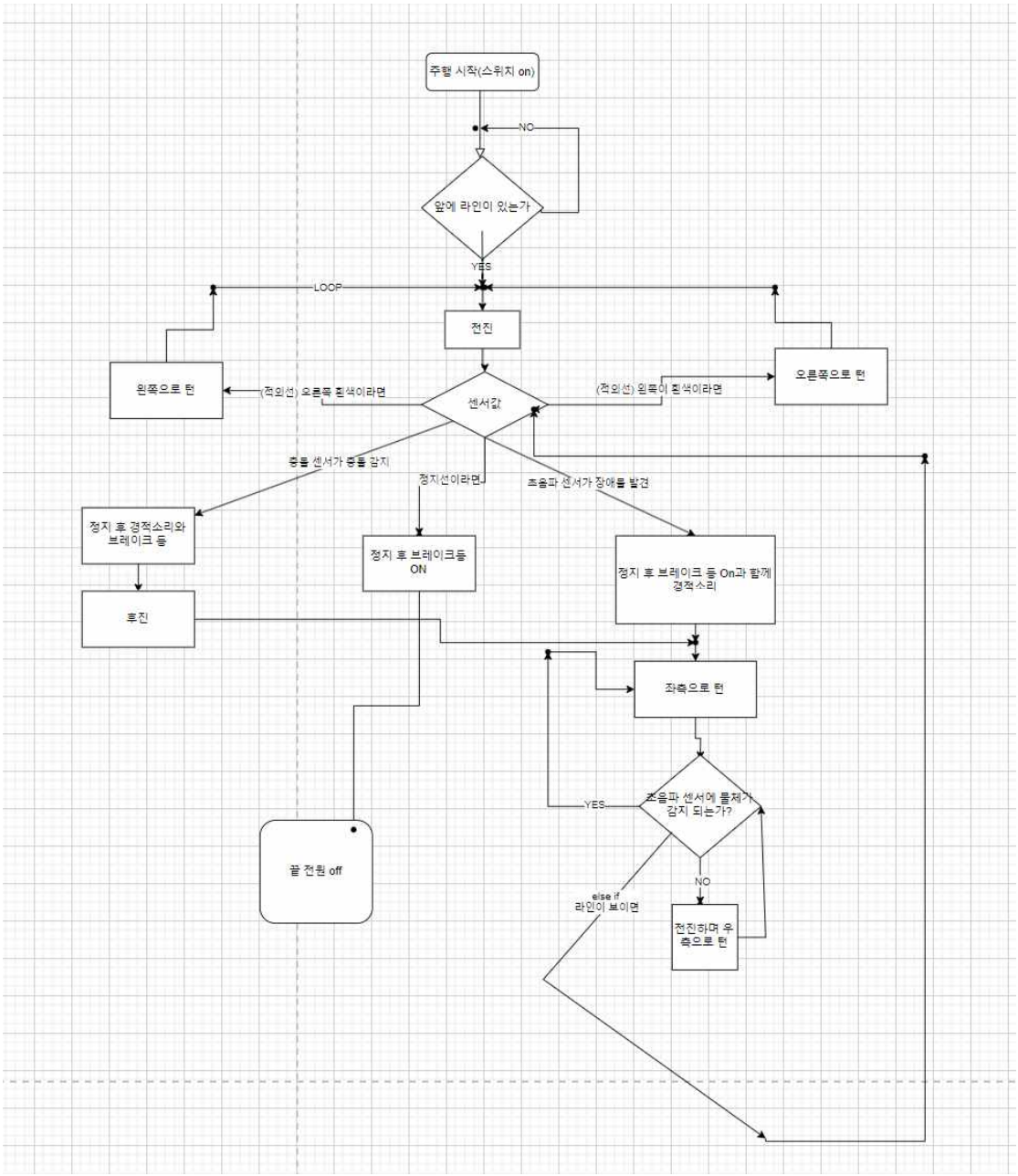
4. 전원 공급: 모터 쉴드에 충분한 전원을 공급한다. 모터와 Arduino 모터 쉴드가 필요로 전압과 전류를 고려해야 한다.
5. 코드 작성: Arduino IDE를 사용하여 모터 쉴드를 제어하기 위한 코드를 작성한다. 모터 드라이버를 통해 모터의 속도와 방향을 설정하는 코드를 포함해야 한다.
6. 모터 드라이버 설정: 모터 드라이버에 대한 설정을 확인하고, 모터와 드라이버가 일치하는지 확인한다. 이로써 모터가 제대로 동작하게 된다.



Motor shield 유의점 :

1. 전원 안정성: 충분한 전원을 제공하여 모터와 모터 쉴드가 정상적으로 작동할 수 있도록 한다. 전류 공급 능력을 고려해야 한다.
2. 온도 관리: 모터 쉴드 및 모터는 높은 온도로 가열될 수 있으므로, 과열을 방지하기 위한 냉각 장치나 열 방출을 고려해야 한다.
3. 모터 드라이버 설정: 모터 드라이버의 스펙과 모터의 스펙을 일치시켜야 한다. 모터 드라이버의 기능과 모터의 요구 사항을 이해할 필요가 있다.
4. 안전 주의사항: 로봇이 작동할 때 안전을 고려해야 한다. 장애물 회피 및 사용자 안전을 위한 알고리즘을 프로그래밍해야 한다.

스케치의 적용 :



결선 방법 : 이번 프로젝트에서 사용하는 Motor shield는 L293D로, 이 모터 드라이브의 결선 방법은 다음과 같다.

1. 모터 드라이브 IC와 모터 연결 : L293D IC의 핀 3, 6, 11, 14는 모터 1의 입력 및 출력핀이고, 2, 7, 10, 15는 모터 2의 입력 및 출력핀이다. 각 모터의 출력 단자를 L293D의 해당 출력 핀에 연결한다.
2. 전원 공급 및 GND 연결 : 이번 프로젝트에서 사용하는 Motor shield는 L293D로, Arduino Uno와 적층하여 사용하기에 모터 드라이브와 Arduino Uno 보드 모두에 전원을 연결해주어야 한다. 또한 외부 입력 전원과 USB를 사용하므로 점퍼를 제거하여 한다. L293D IC에 전원을 공급하기 위해 DC Jack과 핀8(VCC1)와 핀

16(VCC2)를 사용하고, 그라운드(GND)를 핀 1, 9, 12, 16에 연결하여 전원 및 제어 신호의 기준을 제공한다.

3. Arduino와 L293D 연결 : L293D의 핀 2, 7, 10, 15는 Arduino의 디지털 핀에 연결하여 모터의 방향을 제어한다. 이는 모터를 전진시키거나 후진, 정지시키고 속도를 조절하는 데 사용된다.
4. 모터 제어 시그널 연결 : Arduino에서 L293D로 모터를 제어하는 디지털 핀을 연결하기 위해 Arduino의 디지털 핀을 통해 L293D에 연결된 핀 2, 7, 10, 15를 제어한다.

- 18650 배터리와 배터리 홀더를 이용한 전원 공급 방법

배터리와 홀더를 이용한 전원 공급 방법 : 전압 및 전력 요구에 따라 적절한 배터리를 선택하고, 배터리 홀더를 사용해야 안전하고 효율적으로 전원을 공급할 수 있다. 이번 프로그램에서는 18650 배터리와 배터리 홀더를 사용하고, 이를 통해 전원을 공급하는 방법은 다음과 같다.

1. 18650 배터리 선택 및 확인 : 적절한 전압 및 용량을 가진 18650 배터리를 선택한다. 이번 프로젝트에서는 3.7V의 배터리가 사용된다.
2. 배터리 홀더 선택 : 선택한 18650 배터리와 호환되는 배터리 홀더를 선택한다. 배터리 홀더는 배터리를 안전하게 보관하고 연결하는 역할을 한다.
3. 배터리 홀더와 배터리 연결 : 배터리 홀더의 스프링 연결 핀을 사용하여 18650 배터리의 양극과 음극에 연결한다. 배터리 홀더에는 연결 필의 극성이 표시되어 있으며 이 점에 유의한다.
4. 전원 출력 연결 : 배터리 홀더에는 전원 출력 핀이 있으며, 이 출력을 사용하여 다른 장치나 회로에 전원을 공급한다. 전원 출력 핀은 보통 VCC(+)와 GND(-)로 표시되며, 이를 사용하여 외부 장치에 전원을 연결한다.

[IMAGINE]

1. ZK-4WD를 이용한 Line tracer 제작 방향에 대한 창의적 아이디어(기능성/심미성), 브레인스토밍

<기능성>

1. 스텝 모터 : 서보 모터는 출력이 뛰어난 대신 정교성이 부족하고, 스텝 모터는

출력은 부족하지만 정교성이 뛰어남, 이번 프로젝트는 주어진 라인을 얼마나 잘 따라가는지를 확인하는 것이므로 서보 모터보다는 스텝 모터를 사용

2. 초음파 센서 : 전방에 40KHz의 사운드를 출력하고, 초음파가 물체에 부딪혀서 돌아오는 시간을 재서 거리를 감지한다. 이를 통해 전방에 장애물이 있을 시 거리를 감지하여 정지 및 방향을 조절한다.
3. 조도 센서 : 눈의 감도를 기준으로 측정한 광사 속(광속)의 밀도를 측정하여 휴대폰 플래시를 사용하여 주변의 빛을 조절하였을 때 속도 및 방향을 제어할 수 있도록 한다.
4. 진동 충격 센서 : 장애물과 충돌했을 시 발생하는 진동을 감지하여 텍스트 LCD에 충돌 감지 텍스트를 띄운다.
5. 적외선 센서 : 바닥에 그려진 선을 감지하는 역할을 하는 것으로, 흰색과 검은색이 구분되는 곳에서 선의 적외선을 감지하여 라인을 따라 움직일 수 있도록 한다.

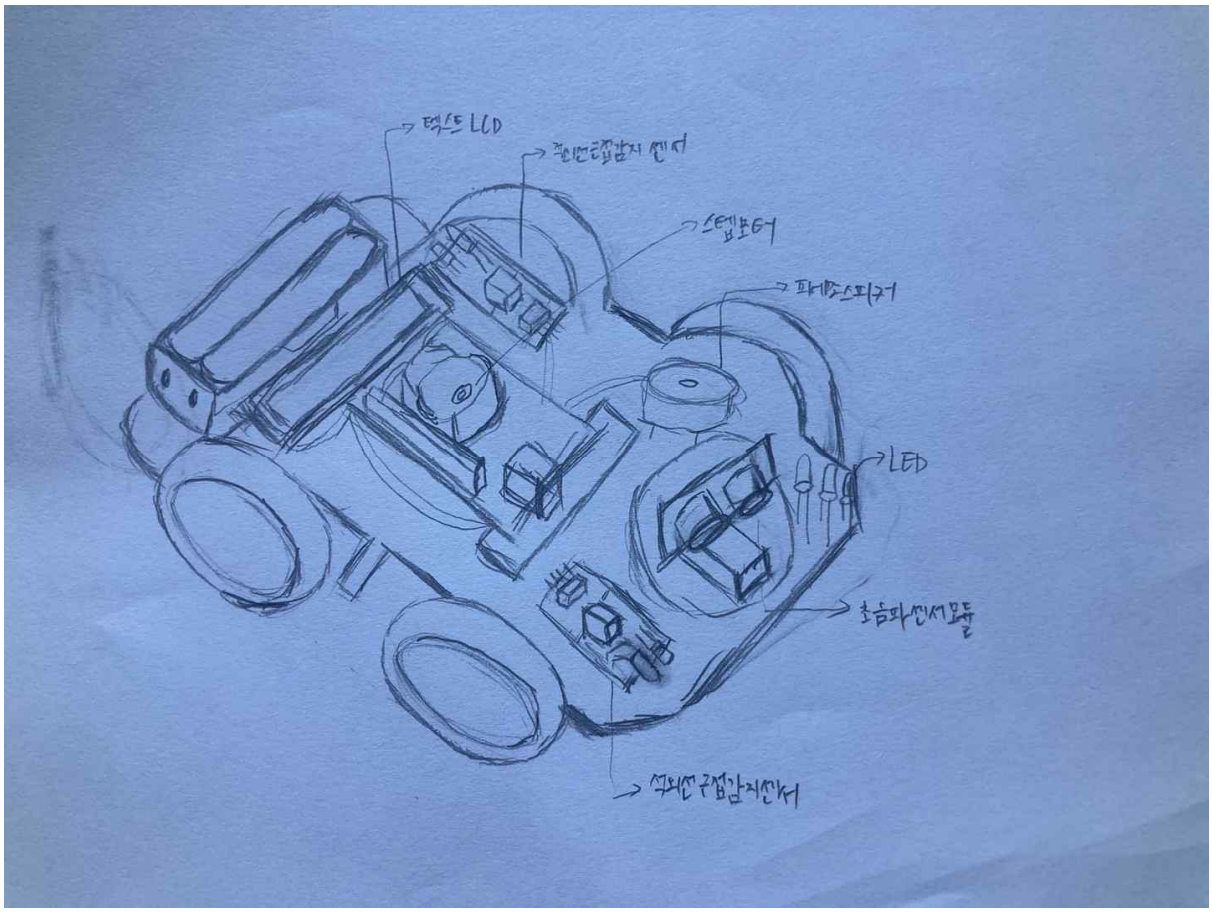
<심미성>

1. 5파이 그린 LED : 구급차의 초록색 불빛을 구현하기 위하여 그린 LED를 사용하여 시간 간격을 두고 깜빡일 수 있도록 한다.
2. 10mm RGB 3색 캐소드(애소드) LED : 1번과 비슷하게 경찰차의 불빛을 구현하기 위하여 파란색과 빨간색을 시간 간격을 두고 깜빡일 수 있도록 한다.
3. 텍스트 LED : 전방에 장애물이 있으면 초음파 센서나 진동 충격 센서에서 받아들인 정보를 사용하여 텍스트를 띄울 수 있도록 한다.
4. 피에조 스피커 : 전방에 장애물이 있으면 초음파 센서나 진동 충격 센서에서 받아들인 정보를 사용하여 경적 소리나 다양한 음성이 나올 수 있도록 한다.

2. **Decision Matrix**

	실현 가능성	심미성	기능성	창의성	계
스텝 모터	20	-	18	16	54
초음파 센서	18	-	18	18	54
조도 센서	15	-	17	18	50
진동 충격 센서	14	-	18	19	51
적외선 센서	18	-	19	17	54
5파이 그린 LED	20	16	-	18	54
10mm RGB 3색 캐소드 LED	20	18	-	18	56
텍스트 LED	18	20	-	18	56
피에조 스피커	20	18	-	18	56

3. **적용하고자 하는 아이디어에 대한 Drawing**



[PLAN]

1. 제작 계획

앞선 IMAGINE 과정에서 제시한 Drawing과 같이 몸체의 아랫판의 하단 부분에 모터를 연결하여 바퀴가 지면과 최대한의 지면 접촉을 확보하고, 몸체의 윗판과 아랫판을 M6 볼트와 3mm 너트를 이용하여 간격을 두고 사이에 배터리를 연결하면 효율적으로 배터리와 전선을 간결하게 배치할 수 있고, 최대한 아두이노 Uno와 다른 센서 및 액츄에이터와의 합선이 없도록 배치할 수 있을 것이다. 또한 몸체의 윗판의 상단 부분에는 아두이노 Uno를 설치하여 쉽게 업로드 및 전원 공급을 쉽게 할 수 있을 것이다.

사용할 센서로는 적외선 센서와 초음파 센서이다. 적외선 센서는 적외선을 이용한 센서로 외부 물질로부터 방사된 적외선이 센서 내의 자발 분극을 갖는 물질의 분극을 변화시켜 외부 자유 전하를 발생시킴으로써 외부 물질을 감지한다. 이 프로젝트에서는 흰색과 검은색이 구분되는 곳에서 선의 적외선을 감지하여 라인을 감지했을 때 1의 신호를, 감지하지 않을 때는 0의 신호를 전달하여 라인 트레이서가 올바른 경로로 주행할 수 있도록 만드는 센서로 사용할 것이다. 초음파 센서는 초음파를 내보내 초음파가 어떤 물체에 부딪혀 다시 돌아오는 시간을 이용하여 거리를 측정하는 센서이다. 이 프로젝트에서는 윗판의 전방에 설치하여 라인 트레이서가 전진 주행 시 전방에 장애물이 있을 시 거리를 감지하여 정지 및 방향(후진 주행)을 조절하는 센서로 사용할 것이다.

또한 사용할 액츄에이터로는 5파이 그린 LED와 피에조 스피커이다. LED는 전극으로부터 반도체에 주입된 전자와 양공이 다른 에너지띠 (전도띠나 원자가띠)를 흘러 P-N 접합부 부근에서 띠틈을 넘어 재결합하는데, 이때 띠틈에서 상당한 에너지가 광자, 즉 빛으로 방출됨으로써 작동한다. 이 프로젝트에서는 심미성을 위하여 구급차의 초록색 빛을 표현하여 시각적 효과를 부여하기 위해 사용할 것이다. 피에조 스피커는 피에조 효과를 사용하여 소리를 발생시키는 액츄에어터로, 결정에 압력을 가할 때 전기 분극에 의해서 전압이 발생하는 현상을 피에조 효과라고 한다. 이 프로젝트에서는 전방에 장애물이 있으면 초음파 센서에서 받아들인 정보를 사용하여 경적 소리나 다양한 음성이 나와 주변 환경을 알릴 수 있도록 할 것이다.

앞서 제시한 Drawing과의 차이점으로는 출력이 부족한 스텝 모터 대신 서보 모터를 사용할 것이고, 기존 2개였던 적외선 센서를 추가하여 전후방 좌우에 두 개씩 설치해 총 4개의 적외선 센서를 설치할 예정이다. 스텝 모터는 출력이 부족하지만 정교성이 뛰어나고, 서보 모터는 출력을 뛰어나지만 정교성이 부족하다. 스텝 모터를 외부 전원과 연결하였을 때 예상한 것보다 출력이 저조하여 코딩을 통하여 정교성을 높일 수 있는 서보 모터를 사용하기로 결정하였다.

- **프로젝트 수행 일정 ;**

  10월 넷째주(10월 24일, 25일) : 교재 수령 및 정리, 간단한 Drawing 및

Sketch 작성

11월 첫째주(10월 31일, 11월 1일) : 라인 트레이서(H/W) 몸체 제작

11월 둘째주(11월 7일, 8일) : 센서 및 액츄에이터 배치

11월 셋째주(11월 14일, 15일) : 코드 수정 및 시험 주행, 문제점 보완

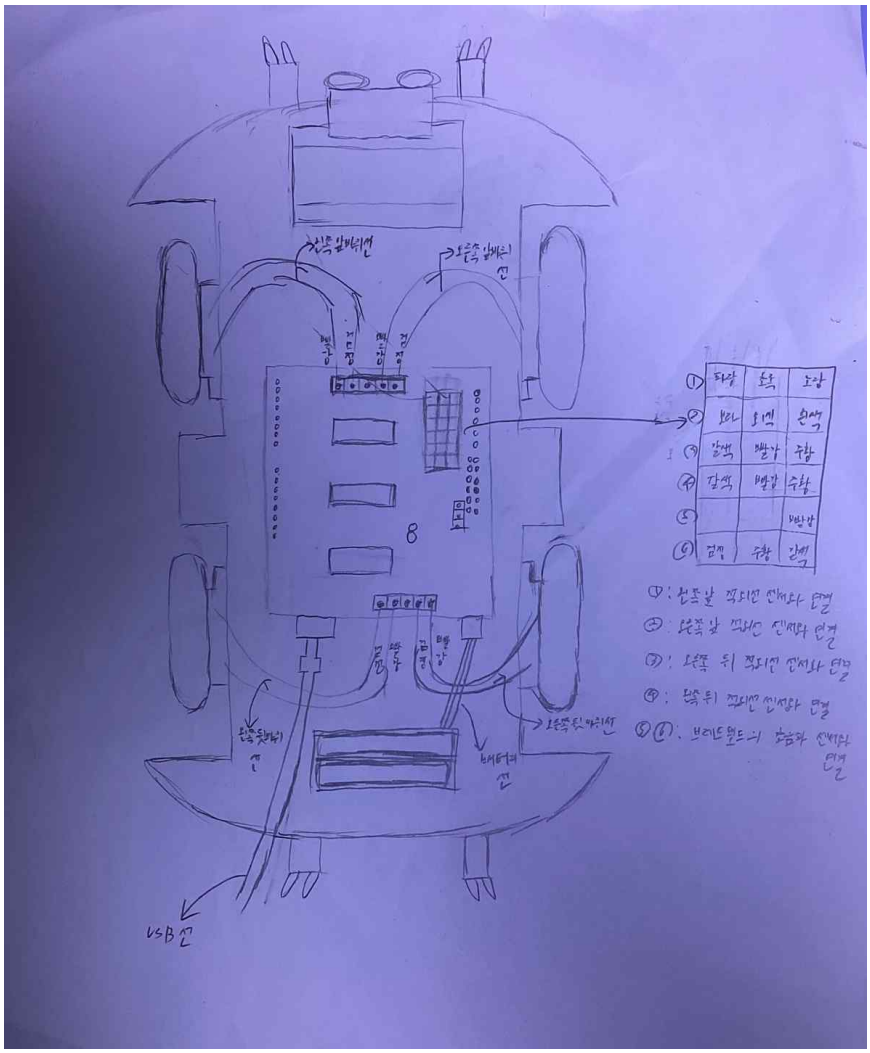
11월 넷째주(11월 21일, 22일) : 코드 수정 및 시험 주행, 외관(심미성) 조형, 문제점 보완

11월 다섯째주(11월 28일, 29일) : 코드 수정 및 시험 주행, 발표 자료 준비, 문제점 보완

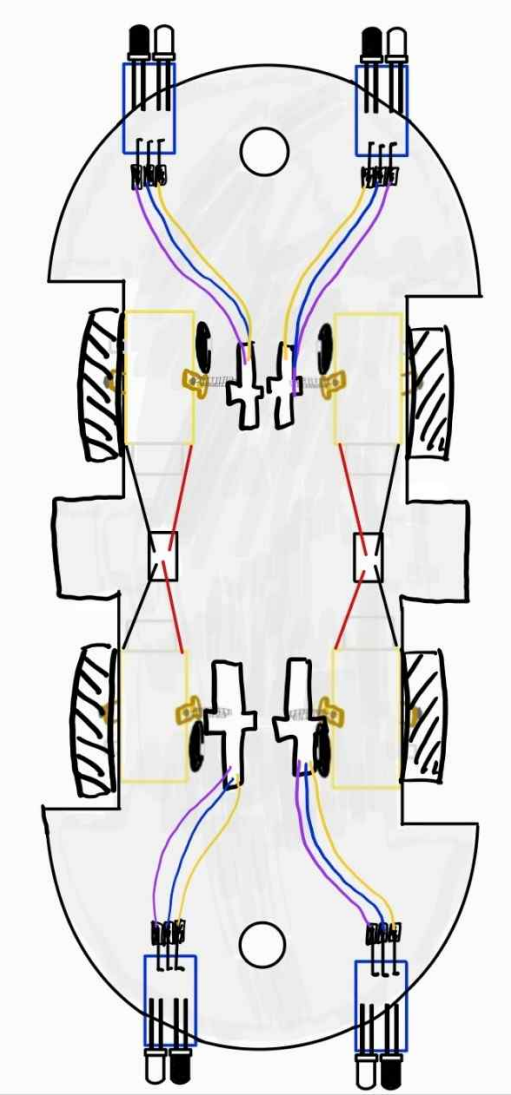
12월 첫째주(12월 5일, 6일) : 실제 주행 및 발표

- **도면 ;**

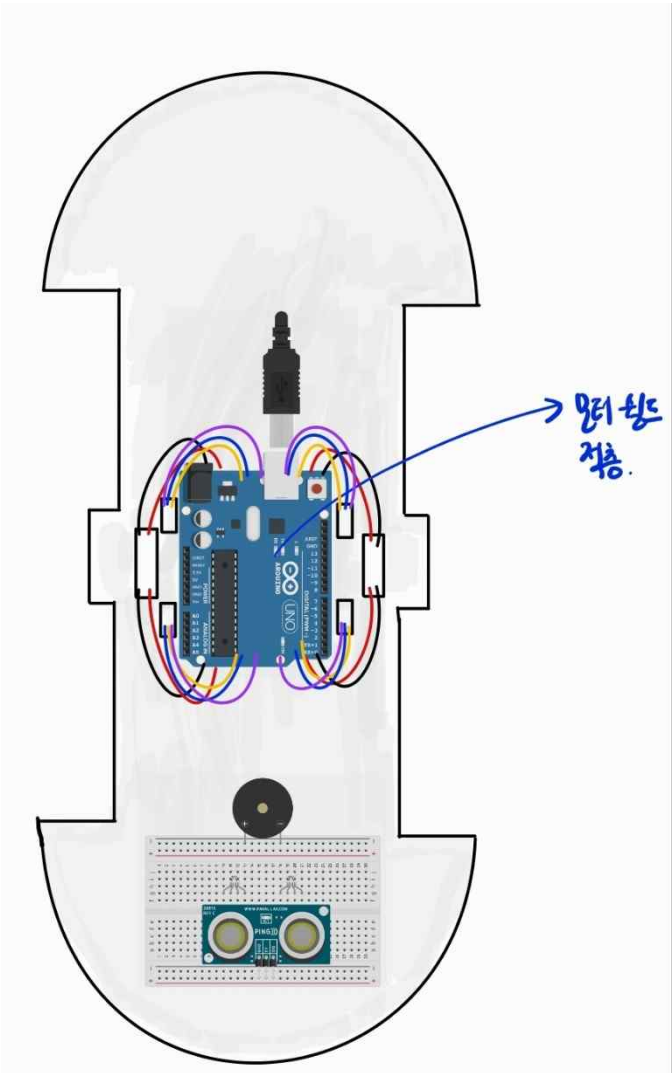
< 결선 도면 >



< 구조물 배치 도면 >



<아랫면>



<윗면>

2. 아두이노 Uno에 의해 작동하는 H/W 제작 계획

좌측 적외선 센서에 라인이 감지된다면 차가 라인보다 오른쪽으로 향한다는 것을 뜻하므로 그 때 왼쪽으로 꺾도록 알고리즘을 구성하였다. 반대의 경우는 이와 반대로 하도록 알고리즘을 구성하였다. 여기서 좌측으로 꺾을 때는 좌측 바퀴들은 후진하고 우측 바퀴는 전진하는 방법으로 알고리즘을 구성하였다. 또한 장애물을 만났을 때는 우선 정지 후 후진 모드로 변경하는 알고리즘을 사용하였다. 여기서 중요한 부분은 전방의 적외선 센서와 후방의 적외선 센서의 간섭을 막는 것이었다. 후진을 할 때 전방 센서 역시 동작한다면 오류가 발생할 확률이 크기 때문이다. 이를 위해 장애물을 감지했을 때와 아닐 때 각각 다른 루프를 돌도록 구성하였다.

3. 아두이노 Uno와 센서, 액츄에이터를 구조물에 설치하는 방법

구조물 구성에 필요한 재료는 아두이노 Uno, 아두이노 Uno serial cable, 브레드보드, 듀폰케이블, 기어박스 세트(모터, 바퀴), 모터 드라이브 쉴드, 18650 배터리, 18650 배터리 홀더, 18650 배터리 충전기, 볼트(M4, M6 등), 3mm 너트, 드라이버이고, 이들을 활용하여 아두이노 Uno와 센서, 액츄에이터를 구조물에 설치하는 방법은 다음과 같다.

우선 모터 4개를 M4 볼트와 3mm 너트를 이용하여 본체 판 2개중 아랫판에 고정시키고 바퀴를 모터에 꽂는다. 그 후 본체 아랫판의 바닥 부분에 적외선 감지 센서를 M4 볼트와 3mm 너트를 이용하여 전방에 2개, 후방에 2개를 장착한다. 이후 본체 판 2개를 M6 볼트를 이용하여 간격을 두어 윗판과 아랫판이 적당히 떨어져 있을 수 있도록 고정시키고, 아두이노 Uno와 적외선 센서가 연결된 브레드보드를 연결하여 윗판 위쪽에 고정시킨다. 그 다음으로 모터에 달린 선을 아두이노 Uno에 연결시키고, 듀폰 케이블을 이용하여 아두이노 Uno와 적외선 센서들을 연결시킨다.

필요한 제작도구에 대한 활용계획 ;

18650 배터리 충전기: 라인 트레이서가 움직일 수 있는 동력을 제공할 배터리를 충전한다.

드라이버: 나사와 너트로 물체를 고정시킬 때 사용한다.

4. H/W를 작동시키기 위한 Sketch 작성 계획

라인 트레이서의 하드웨어(H/W) 구성을 설계하고 제작하기 위한 스케치 작성 과정은 체계적이고 효율적인 계획을 필요로 한다.

1. 하드웨어 요구사항 분석

먼저, 라인 트레이서의 목적과 기능을 고려하여 필요한 하드웨어 요소를 분석한다. 이 단계에서는 모터, 적외선 센서, 초음파 센서, LED, 피에조 스피커 등을 고려한다.

2. 서보 모터 및 바퀴 배치 계획

몸체의 하단에 서보 모터와 바퀴를 최대한 지면과 접촉하도록 배치한다. 서보 모터의 출력 문제를 해결하기 위해 고출력이면서도 정교한 제어가 가능한 서보 모터를 선택하고, 모터와 바퀴를 각각 어떻게 배치할지 스케치에 표현한다.

3. 적외선 센서 및 위치 선정

적외선 센서는 주행 경로의 색상 변화를 감지하므로, 라인 트레이서의 전반적인 동작에 큰 영향을 미친다. 전방, 후방, 좌우에 각각 두 개씩의 적외선 센서를 배치하고, 스케치 상에서 센서의 위치와 방향을 표시한다.

4. 초음파 센서 위치 및 각도 조정

초음파 센서는 전방의 장애물을 감지하여 정지 및 방향 조절을 담당한다. 초음파 센서를 윗판의 전방에 설치하고, 정확한 감지 각도를 위해 스케치 상에서 센서의 위치 및 각도를 조정한다.

5. LED 및 피에조 스피커 위치 계획

LED는 시각적인 신호를 제공하며, 피에조 스피커는 음성 신호를 발생시킨다. 각각의 위치를 정하고, 특히 구급차의 초록색 빛을 표현하기 위한 LED의 위치를 결정한다. 또한, 피에조 스피커는 전방에 장애물이 있으면 초음파 센서에서 받아들인 정보에 따라 소리를 내도록 스케치에 표현한다.

6. 아두이노 Uno 설치 위치 및 전원 관리

몸체의 윗판 상단에 아두이노 Uno를 설치하고, 코드 업로드 및 전원을 쉽게 관리할 수 있도록 스케치에 해당 부분을 표시한다. 전원 소비를 최소화하기 위해 배터리의 위치 및 전원 관리 회로도 함께 계획한다.

7. 부가적인 메카니즘 및 조정

모터, 센서, LED 등의 부품들을 부착하고 조립할 때, 부가적인 메커니즘을 도입하여 조정이 가능하도록 계획한다. 센서의 위치 및 각도를 조절할 수 있는 메커니즘을 추가하여 실험 및 테스트 과정에서 센서의 정확한 위치를 조정할 수 있도록 한다.

8. 스케치와 도면 작성

최종적으로 하드웨어 스케치를 토대로 도면을 작성한다. 각 부품의 상세한 크기 및 간격을 명시하고, 전체 시스템의 외관 및 구조를 정확히 기록하여 제작 단계에서 필요한 정보를 명확히 전달한다.

이러한 계획을 바탕으로 하드웨어 스케치를 작성하면, 라인 트레이서의 제작에서 발생할 수 있는 다양한 문제를 예방하고 효과적으로 해결할 수 있을 것이다.

[CREATE]

1. 기능성과 심미성의 관점에서 제작 방향에 대한 설명

기능성: 우리 조는 적외선 센서를 전방에 2개, 후방에 2개 설치하였다. 후방에 2개를 장착함으로써 전진 주행 시에는 전방의 2개의 적외선 센서가 작동하여 라인을 감지하고, 초음파 센서가 장애물을 감지하면 전방의 적외선 센서는 감지를 중지하고 후방의 2개의 적외선 센서가 작동하여 후진 주행을 하게 된다. 또한 사용하고자 계획했던 스텝 모터는 출력이 예상보다 부족하여 정교성은 부족하지만 출력이 좋은 서보모터를 사용하여 바퀴가 원할하게 굴러갈 수 있도록 하였다. 부족한 정교성은 코드를 수정하여 보완하기로 하였다. S/W 부분의 역할에 대한 설명은 다음과 같다.

```
#include <NewPing.h>
#include <AFMotor.h>
```

이 부분은 헤더파일을 불러오는 부분이다. 우리는 DC모터를 사용했기에 AFMotor.h라는 헤더를 불러왔다.

```
#define TRIGGER_PINA0
#define ECHO_PINA1
#define max_distance 50
// ir sensor
#define irLeftA5
#define irRightA4
#define irbLeftA3
#define irbRightA2
// motor
#define MAX_SPEED 200
#define MAX_SPEED_OFFSET 20
```

이 부분은 define 즉 어떠한 값을 보기 편하도록 치환해준 부분이다. 가령 irbLeft는 후방 ir센서를 뜻한다.

```
NewPing sonar(TRIGGER_PIN, ECHO_PIN, max_distance);
AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(4, MOTOR12_1KHZ);
AF_DCMotor motor3(2, MOTOR34_1KHZ);
AF_DCMotor motor4(3, MOTOR34_1KHZ);
```

이 부분은 초음파 센서와 DC모터의 객체를 설정해준 부분이다.

```
Int distance = 0;
Int leftDistance;
Int rightDistance;
boolean object;
Int bLeftValue;
Int bRightValue;
```

이 부분은 변수를 미리 만들어 둔 부분이다.

```
void setup() {
  Serial.begin(9600);
  pinMode(irLeft, INPUT);
  pinMode(irRight, INPUT);
  pinMode(irbLeft, INPUT);
  pinMode(irbRight, INPUT);
  motor1.setSpeed(200);
  motor2.setSpeed(200);
  motor3.setSpeed(120);
  motor4.setSpeed(120);
}
```

setup함수는 아두이노의 가장 기초적인 세팅을 다루는 부분이다. pinMode는 아두이노 우노에 연결한 선 번호를 코드에 입력함으로써 S/W와 H/W의 연동을 담당하는 함수이다. 그 다음은 모터 각각의 객체에 대해 파워를 설정한 코드이다. 아까 정의해준 motor1,2,3,4객체에 AFMotor헤더파일의 함수인 .setSpeed() 함수를 사용하여 설정해주었다.

```
void loop() {
  if (isPathClear()) {
    if (digitalRead(irLeft)== 0&& digitalRead(irRight)== 0) {
      moveForward();
    } else if (digitalRead(irLeft)== 0&& digitalRead(irRight)== 1) {
      Serial.println("TL");
```

```
    moveLeft();
  } else if (digitalRead(irLeft)== 1&& digitalRead(irRight)== 0 ){
    Serial.println("TR");
    moveRight();
  } else if (digitalRead(irLeft)== 1&& digitalRead(irRight)== 1) {
    Stop();
    delay(100);
  }
```

먼저 첫 번째 if문을 보면 isPathClear함수가 true을 리턴했을 때만 실행되는 것을 알 수 있다. 이는 장애물을 발견하지 못했을 때라는 뜻이다. 그 if문에 속해 있는 두 번째 if-else문은 ir센서가 좌측에서 감지되었을때는 좌측으로 이동, 우측은 반대로 이동 등과 같이 알고리즘을 구성한 부분이다.

```
} else {

  intcheck=0;
  while (1) {
    bLeftValue = digitalRead(irbLeft);
    bRightValue = digitalRead(irbRight);
    if (bLeftValue == 0&& bRightValue == 0) {
      moveBackward();
    } else if (bLeftValue == 0&& bRightValue == 1) {
      motor1.run(BACKWARD);
      motor2.run(BACKWARD);
      motor3.run(FORWARD);
      motor4.run(FORWARD);
    } else if (bLeftValue == 1&& bRightValue == 0) {
      motor1.run(FORWARD);
      motor2.run(FORWARD);
      motor3.run(BACKWARD);
      motor4.run(BACKWARD);
    } else {
      Stop();
      //여기에 종료음 넣을 예정
      break;
    }
```

위에 첫 번째 if문의 else일 경우를 살펴보면 장애물이 감지됐을 경우인 걸 알 수 있다. 장애물이 감지되면 후진을 해야 하므로 여기서는 while문을 사용하여 else문 안에서 반복하도록 하였다. 그러면 전방의 IR센서와 간섭이 없을뿐더러, 정상적으로 후진 알고리즘을 작동시키게 된다. 여기서 문제는 잘못된 장애물을 인식했을 때이다. 왜냐하면 이 while문은 도착지점을 도달할 때까지 무한 반복되기에 자칫 오류가 생길 수 있다. 이를

해결하기위해 isPathClear함수에서 이 문제점을 해결했다. 여기서 하드웨어와 연계되는 부분은 digitalRead()함수이다. 여기에 들어있는 irLeft, irRight등은 아까 정의해준 시리얼 넘버에 해당하고 digitalRead() 함수의 변수에 시리얼 넘버 값을 넣으면 그 리턴 값은 적외선에 감지되었는가 아닌가의 결과가 출력되게 된다. 이러한 방식으로 H/W와 S/W를 연계시켰다.

```
    }
  }
}
void dest() {
  intcnt=0;

  while(cnt<30){
  digitalWrite(1,HIGH);//LED 조정 필요
  delay(500);
  digitalWrite(1,LOW);
  delay(500);
  cnt+=1;
  }
}
```

이 부분은 도착했을 때 발생하는 알고리즘으로 도착한다면 LED가 특정한 패턴으로 켜지도록 구성했다. 즉 알고리즘을 해석해보면 30번 점멸하는 것을 알 수 있다. 여기서 H/W의 연계부분은 digitalWrite()함수이다. 여기에 1은 시리얼 넘버이며 HIGH은 불을 킨다는 명령이다.

```
boolean isPathClear() {
  intdistance = getDistance();
  if (distance >= 15) {
    return true;
  }
  else {
    digitalWrite(1,HIGH);
    Stop();
    delay(100);
    distance=getDistance();
    if(distance<15){ //1초동안 계속 장애물이 유지된다면 후진 아니면 전진알고리즘 적용
      return false;
    }
    else {
      return true;
```

```
    }
  }
}
```

이 부분은 isPathClear함수로 초음파센서로 거리를 측정해 만약 그 거리가 15cm미만이라면 false를 이상이라면 true를 리턴하도록 하였다. 여기서 주의점은 장애물을 잘 못 인식한 경우다. 가령 실험자의 손 등이 변수였다. 이를 해결하기위해 1초의 텀을 두고 또 다시 장애물이 15cm 미만일시에 false를 리턴하도록 하였다.

```
int getDistance() {
  delay(50);
  intcm = sonar.ping_cm();
  if (cm == 0) {
    cm = 100;
  }
  returncm;
}
```

이 부분은 초음파 센서로부터 거리를 구해오는 함수를 만든 것이다. 여기서 중요했던 부분은 Sonar.ping_cm()이라는 함수인데 이는 NewPing이라는 헤더파일의 함수로, 초음파로 감지되는 거리를 읽어오는 역할을 한다.

```
void Stop() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}
void moveForward() {
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
void moveBackward() {
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
}
void moveRight() {
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
```

```
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
void moveLeft() {
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
}
```

이 부분은 움직임에 대한 코드를 줄이기위해 함수로 설정한 부분으로 오른쪽과 왼쪽 바퀴의 전진, 후진에 대한 함수이다. 여기서 하드웨어와의 연동 부분은 AFMotor 헤더파일에서 나온 .run()함수이다. 이 함수는 먼저 설정해둔 motor1,2,3,4각각의 객체의 구동방식을 설정하는 함수이다.

심미성: 우리 조는 5파이 그린 LED를 이용하여 초록색 빛을 내어 구급차라는 컨셉에 맞게 제작하였다. 또한 피에조 스피커를 이용하여 라인 트레이서가 주행하다 장애물과 만나게 되면 초음파 센서에서 받아들인 정보를 사용하여 이 스피커를 통해 경적 소리나 다양한 음성이 나올 수 있도록 구현하였다. 그리고 텍스트 LED를 이용하여 라인 트레이서가 장애물과 마주하면 피에조 스피커와 마찬가지로 초음파 센서에서 받아들인 정보를 사용하여 "전방에 장애물 조심"이라는 문구를 띄울 수 있게 만들었다.

2. Plan 단계에서 수립한 제작 계획을 바탕으로 성과물 제작 과정에 대한 설명

10월 24일, 25일 : 라인 트레이서 교재 수령 및 교재 정리 **(재료의 조달)**

; 라인 트레이서를 제작할 수 있는 교재와 라인 트레이서가 작동할 수 있도록 돕는 센서 및 액츄에이터를 수령하였고, 이후 개수 및 종류를 정리하여 표로 작성하였다. 모든 활동이 끝난 후 라인 트레이서를 분해하여 작성한 표를 기준으로 반납할 예정이다. 또한 역할을 분담하여 김종민 학생의 리드 하에 코딩 및 H/W 제작은 김종민 학생이, 센서 및 액츄에이터 연결, H/W 제작은 김재윤 학생이, 계획 구성과 시험 주행은 김동윤 학생이, 도면 및 Sketch 등의 Drawing은 김남준 학생이 진행하기로 결정하였다.

10월 31일 : 라인 트레이서 제작 시작, 스티커 제거 및 아랫판과 모터, 바퀴 결합 **(H/W 제작 1)**

; 교재 중 라인 트레이서의 몸체를 구성하는 부품들을 조립하였다. 각 판의 스티커를 제거하였고, 아랫판에 DC 모터 및 바퀴를 결합하여 라인 트레이서의 기본적인 몸체를 제작하였다.

11월 1일 : 간단한 Sketch 작성(센서, 배터리 등의 위치 설정, 프로그래밍 Flowchart 작성), **(Sketch 작성)**

; 10월 31일에 제작한 몸체에 센서와 액츄에이터, 배터리 등을 어느 위치에 결합해야 할지 의견을 나누어보았고, 라인 트레이서가 설치한 센서와 액츄에이터 등을 활용하여 어떻게 라인을 따라갈지 대략적인 Flowchart를 작성하여 프로그래밍의 기초 틀을 확립하였다.

11월 7일 : Sketch에 따라 센서 및 액츄에이터 결합 **(H/W 제작 2) (보완 계획 1)**

; 11월 1일에 작성한 Sketch에 따라 교재에 있는 아두이노 Uno, 적외선 센서, 배터리 등을 몸체에 결합하였고, 아두이노 Uno에 코드를 업로드하여 주행을 시도해보았지만 올바른 주행에 실패하여 이유에 대해 의견을 나누어보았다. 이유로 배터리의 전력 부족, DC 모터의 민감도 및 출력 등의 의견이 나와 배터리를 집에서 충전해오고, 바퀴의 민감도 및 출력은 코드에서 DC 모터의 출력을 90에서 130으로 늘리기로 결정하였다.

11월 8일 : 코드 수정 및 초음파 센서 결합 **(H/W 제작 3) (S/W 제작 1)**

; 충전해온 배터리를 결합하고 코딩을 통해 DC 모터의 출력을 높였을 때 바퀴가 돌아가는 힘에는 문제가 없었지만, 바퀴 각각의 돌아가는 방향이 달라 아두이노 Uno에서 출력되는 핀의 위치와 코드를 수정하여 각각의 방향을 일치시켰고, 절연 테이프를 이용하여 전진 및 후진이 잘되는 것을 확인할 수 있었다. 이후 실제 라인을 따라갈 때 있을 장애물을 대비하여 장애물을 감지할 수 있도록 초음파 센서를 결합하였다.

11월 14일 : 꺾여있는 라인을 따라갈 수 있도록 코딩 작업 **(S/W 제작 2)**

; 적외선 센서를 활용하여 적외선이 반사되면 0, 검은색의 라인이 적외선을 흡수하여 반사되지 않으면 1로 인식하고, 좌측의 센서에서 1로 감지하여 라인이 감지되면 좌측 바퀴들은 후진, 우측 바퀴들은 전진하여 왼쪽으로 꺾고, 이와 반대의 경우에는 반대의 과정을 통해 라인이 꺾이는 코너에서 라인을 올바르게 따라갈 수 있도록 알고리즘을 구성하였

다.

11월 15일 : 장애물을 감지하여 후진할 수 있도록 코딩 작업 **(S/W 제작 3)**

; 초음파 센서를 활용하여 장애물을 인식하였을 때, 전진 모드로 수행하던 라인 트레이서가 우선 정지 후 후진 모드로 운전 방법을 바꿀 수 있도록 코딩하였다. 후진으로 수행할 때는 설치한 4개의 적외선 센서 중 전진 시 사용한 초음파 센서 쪽 전방의 두 개의 적외선 센서는 더 이상 감지하지 않고, 반대 쪽 후방의 두 개의 적외선 센서가 라인을 감지하여 후진으로 라인을 따라갈 수 있도록 알고리즘을 구성하였다.

11월 21일 : 제작 후 모의주행 실시 **(제작 후 시험 평가 1) (S/W 제작 4)**

; 지난주까지 구성한 코드 및 알고리즘 수정을 마치고, 아래층에 만들어진 실제 라인을 주행해보았다. 이 때 확인한 문제점으로는 DC 모터의 출력이 조금씩 달라졌고, 바퀴가 지면과 닿지 않아 헛돈다는 것이었다. DC 모터의 출력과 관련한 코드를 일부 수정하였고, 배터리는 다시 충전하여 오기로 하였으며, 바퀴가 지면과 잘 맞닿을 수 있도록 DC 모터 및 바퀴의 위치를 수정하기로 하였다.

11월 22일 : DC 모터 및 바퀴 위치 수정 및 모의주행 실시 **(H/W 제작 4) (제작 후 시험 평가 2) (보완 계획 2)**

; 어제 모의주행을 해본 결과 출력 및 바퀴의 위치에 문제가 있음을 알게 되었다. 일부 코드를 수정하고 배터리를 충전해온 결과 출력에는 문제가 없어졌지만 아직 바퀴가 헛돈다는 문제가 있어 바퀴가 지면과 충분히 닿을 수 있도록 라인 트레이서의 윗판과 아랫판 사이에 위치해있던 DC 모터의 위치를 아랫판의 밑부분에 설치하였다. 이후 모의주행을 실시했을 때 모터의 출력에 이상이 있거나 바퀴가 헛도는 문제점은 해결됐음을 알 수 있었다.

- **참고한 자료를 분석하여 적용한 것과 팀의 독창적 아이디어를 적용한 사항 :**

  void 함수를 이용해서 직진, 멈춤, 우회전, 좌회전의 함수를 따로 적어두고 loop 칸에 비교적 간단하게 적었다. if, else 구문에는 장애물을 발견하면 1초간 텀을 가진 뒤 움직이도록 코딩하였다. 모터의 스피드는 모터1과 모터2는 200, 모터3과 모터4는 120으로 설정하였다. 듀폰 케이블을 사용하여 아두이

노 우노와 적외선 센서와 연결하였다.

- **제작 과정에서 발생하는 문제점과 극복하는 과정 :**

우선, 스텝모터를 사용하였다가 출력이 약해 라인 트레이서가 제대로 주행하지 못하였다. 그래서 스텝모터에 비해 정교성은 떨어지지만 출력이 좋은 서브모터를 사용하였고, 정교성이 떨어지는 부분은 코딩을 통해 민감도를 조절하여 해결하였다.

다음으로, 바퀴가 땅에 닿지 않고 헛도는 문제가 있었다. 이를 해결하기 위해 본체를 다시 분해하고 모터와 바퀴의 위치를 조절하였다.

또한 테스트를 자주 하다 보니 배터리의 전원 공급이 원활하지 않았다. 그래서 최대한 전원을 끈 상태로 유지하고 배터리도 꾸준히 충전하였다.

마지막으로 센서의 각도가 적절하지 않아 라인 트레이서가 원하는 방향으로 움직이지 않는 문제가 있었다. 이를 해결하기 위해 적외선 센서를 본체 밑판에 설치하여 최적의 감지 각도로 맞췄다.

[IMPROVE]

1. 설계물 시험 평가 결과 ;

- 결함/오류 유발 요인에 대한 분석, 수정 보완 과정 :
  아두이노 라인 트레이서의 제작 과정에서 발생할 수 있는 결함과 오류, 그리고 이를 수정하는 보완 과정에 대한 상세한 분석을 다음과 같이 설명할 수 있다.

  1. 센서 캘리브레이션 오류
  <원인>
  - 라인 센서의 값이 정확하지 않거나 캘리브레이션이 부적절하면 라인을 정확하게 감지하지 못할 수 있다.
  <수정과 보안>
  - 센서 값의 정확한 캘리브레이션을 위해 테스트를 수행하고, 검은색과 흰색 경계에서 올바른 값이 읽히는지 확인한다.
  - 임계값을 동적으로 조절하는 방식을 도입하여 주행 중에도 적응할 수 있도록 설계한다.

2. 모터 제어 오류

<원인>

- 모터 핀이 부적절하게 설정되거나 모터 드라이버가 정확하게 동작하지 않아 로봇의 움직임이 부정확했다.
- 모터마다 출력의 세기가 달라서 주행을 할 때 원하는 방향대로 차가 움직이지 않았다.
- 모터에 연결된 선이 너무 잘 끊어져서 연결 과정에 난항이 있었다.

<수정과 보안>

- 모터 핀 및 드라이버 설정을 확인하고, 모터의 회전 방향 및 속도를 정확하게 조절할 수 있는 함수를 구현한다.
- 주행 연습을 해주면서 모터의 출력이 다르다면 최대한 다른 모터들과 출력이 비슷한 것으로 모터를 교체해준다.
- 모터에 연결된 선이 끊어지지 않도록 모터를 연결한 선을 다른 선들과 엮이지 않게 잘 분리해 놓는다.
- 속도 제어 및 부드러운 모터 가속/감속을 위해 PWM(펄스 폭 변조)을 사용하여 모터를 제어한다.

3. 전원 공급 및 전압 문제

<원인>

- 충분한 전원이 공급되지 않거나 전압이 일정하지 않으면 모터나 센서의 동작이 불안정해
지거나 잘 작동되지 않을 수 있다.

<수정과 보안>

- 안정적인 전원 공급을 위해 충분한 용량의 배터리를 사용하거나 안정화된 전원 공급 장치를 도입한다.
- 전압 및 전류를 측정하고 필요한 경우 보정하여 시스템의 안정성을 높인다.

4. 물리적 결함 문제

<원인>

- 적외선 센서 중 제대로 작동하지 않는 것이 많아서 라인을 제대로 감지하지 못하고 멋대로 움직이는 경우가 많았다.
- 바퀴가 생각했던 대로 부드럽게 움직이지 않아서 주행 과정이 원활하지 않았다.

<수정과 보안>

- 제대로 작동이 되지 않는 적외선 센서를 찾아서 정상적인 적외선 센서로 교체해주었다.
- 바퀴가 차체의 윗부분에 걸려 잘 움직이지 않았기에 차체의 높이를 높여 바퀴가 차체에 걸리지 않도록 수정하였다.
- 필요한 경우 메커니즘을 보완하고 물리적 안정성을 높이기 위한 개선 작업을 수행했다.

5. 기타 오류

<원인>

- 피에조 스피커를 연결해보려 하였지만 배터리가 공급하는 전력이 충분하지 않아 피에조 스피커를 연결할 수 없었다.
- 모터 쉴드의 출력 핀이 충분하지 않았다.

<수정과 보안>

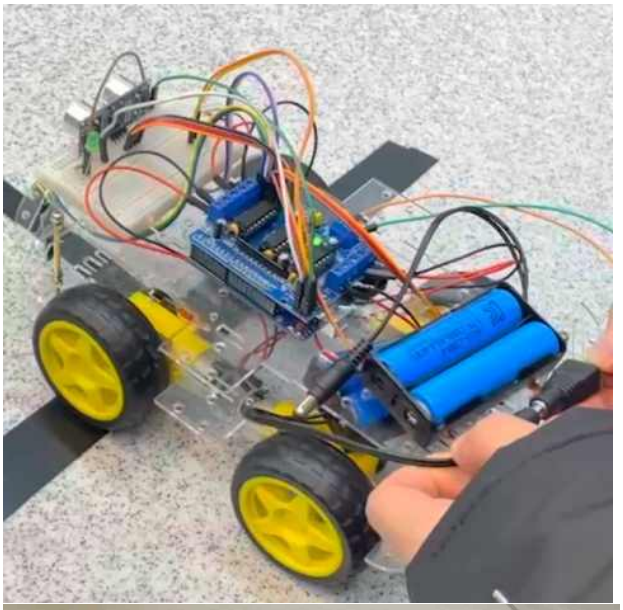
- 더 많은 전력을 공급할 수 있는 배터리를 사용하여야 피에조 스피커와 LCD등을 연결할 수 있을 것 같다.
- 모터 쉴드의 핀이 충분하지 않아서 브레드보드를 활용하여 이를 해결하였다.

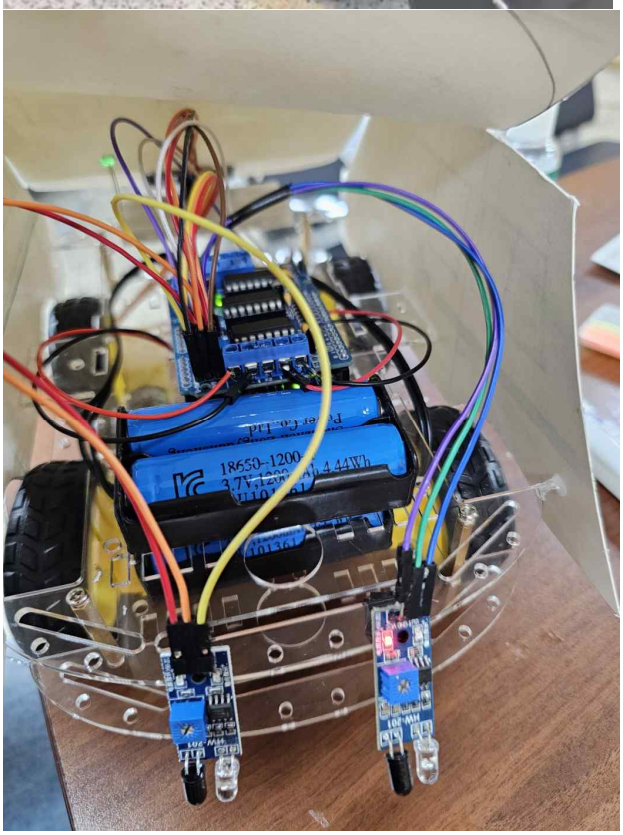
마무리:

라인 트레이서의 제작에서 발생하는 결함과 오류는 다양하며, 이에 대한 해결과 보안은 체계적인 테스트, 디버깅, 그리고 수정 작업을 통해 이루어집니다. 실제로 로봇이 주행하면서 얻은 데이터를 분석하고 개선 작업을 지속적으로 수행하여 최적의 성능을 달성할 수 있습니다.

2. 최종 성과물에 반영된 수정 보완 사항의 설명 ;

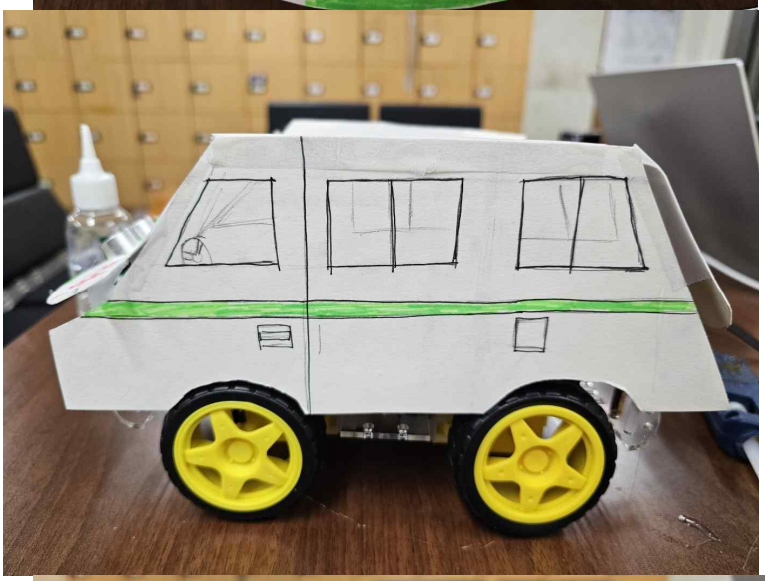
- H/W 설명 :
  - ■ 설계물 사진, 사진에 대한 설명 :





몸체의 아랫판에 4개의 바퀴와 모터를 설치하였고, 적외선 센서를 라인 트레이서의 앞쪽에 2개, 뒤쪽에 2개 설치하였다. 몸체의 윗쪽판에는 아두이노 Uno와 브레드보드, 배터리 홀더 등을 설치하였고, 브레드보드에 초음파 센서와 LED를 설치하였다. 적외선 센서와 초음파 센서, 브레드보드 등은 듀폰케이블(F-M, F-F 등)을 사용하여 아두이노 Uno에 연결하였다.


환자이송



우리 조의 Line Tracer의 테마는 응급실의 구급차로, 외관을 구급차의 초록색 위주으로 꾸미고, 5파이 그린 LED를 사용하여 구급차의 녹색 경광등을 표현하고자 하였다. 또한 초음파 센서를 사용하여 장애물을 감지하였을 때, 회전이 어려운 좁은 골목길에 들어간 구급차를 염두에 두어 후진으로 주행 모드를 변경하여 빠져나올 수 있도록 하였다.

■ 설계물 작동 동영상(스트리밍 서버 주소)과 설명 :

스트리밍 서버 주소 : https://youtube.com/shorts/AVQpFn3z9r4

Line Tracer의 Arduino Uno에 18650 배터리를 이용하여 전원을 공급하면 전방에 달려있는 2개의 적외선 센서로 검은색 선과 바닥을 구분하여 선을 따라가게 된다. 선이 휘어지는 코스가 나오면 바퀴의 회전 방향과 세기를 조절하여 선을 올바르게 따라간다. 이후 장애물을 마주하게 되면 초음파 센서가 이를 감지하여 라인 트레이서가 잠시 멈추게 되고, 후진으로 주행 모드를 변경하여 후방의 2개의 적외선 센서를 활성화하여 다시 선을 따라 주행한다. 마찬가지로 적외선 센서가 선을 감지하여 후진하고 바퀴의 회전 방향과 세기를 조절하여 방향을 전환한다. 왕복으로 돌아오고 나면 라인 트레이서가 정지하고 LED에서 불빛이 나게 된다.

- S/W 설명 :

■ 스케치에 대한 수정 과정의 상세한 설명 :

우선 크게 "1. 주행 관련 함수, 2. 장애물 탐지 함수, 3. 메인 부분인 루프 함수 4. 기능적인 함수"로 나누어 설명하겠다.

1. 주행 관련 함수

앞으로 설명하게 될 함수들은 코드의 간결성을 위해 정의한 코드들이다. 물론 코드의 메인 부분에서 모터 4개에 대한 코드를 씀으로써 아래 함수들을 사용하지 않을 수 있지만, 간결성은 현 프로젝트에서 굉장히 중요한 부분이므로 함수를 따로 정의하게 되었다.

```
void Stop() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}
```

위 함수는 멈추는 기능을 가지는 함수로 해당 함수에서는 문제가 발생하지 않았으므로 수정하지 않았다.

```
void moveForward() {
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
```

위 함수는 전진 기능을 가진 함수로 전진할 때는 큰 문제가 없었기에 수정하지 않았다.

```
void moveBackward() {
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
}
```

위 함수는 후진 기능을 가진 함수로 후진할 때는 큰 문제가 없었기에 수정하지 않았다.

```
void moveRight() {
  motor1.setSpeed(115);
  motor2.setSpeed(115);
  motor3.setSpeed(115);
  motor4.setSpeed(115);
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
```

```
}
```

위 함수는 우회전 기능을 가진 함수로 우회전 시 바퀴가 마찰력으로 인해 멈추는 현상이 테스트를 통해 관찰되었으므로 해당 함수에서 모터 스피드를 기존 셋업보다 올려주어 문제를 해결하였다.

```
void moveLeft() {
    motor1.setSpeed(115);
  motor2.setSpeed(115);
  motor3.setSpeed(115);
  motor4.setSpeed(115);
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
}
```

위 함수는 좌회전 기능을 가진 함수로 좌회전시 바퀴가 마찰력으로 인해 멈추는 현상이 테스트를 통해 관찰되었으므로 해당 함수에서 모터 스피드를 기존 셋업보다 올려주어 문제를 해결하였다.

2. 장애물 탐지 함수

다음에 설명하게 될 함수는 장애물을 탐지하는 함수로써 초음파 센서를 SW로 적용하는 함수이다.

```
#include <NewPing.h>
```

다음은 초음파 센서를 사용하려면 필요한 헤더파일로 따로 불러오게 되었다.

```
#define TRIGGER_PIN A0
#define ECHO_PIN A1
#define max_distance 50
```

다음은 미리 일정 값을 정의한 상수 명들이다.

```
NewPing sonar(TRIGGER_PIN, ECHO_PIN, max_distance);
```

다음은 초음파 센서의 객체를 만들어준 부분이다. 즉 여기서 HW와 SW의 연동이 이루어졌다.

```
boolean isPathClear() {
  intdistance = getDistance();
  if (distance >= 15) {
```

```
    return true;
  }
  else {
    digitalWrite(LED_PIN, HIGH);
    Stop();
    delay(100);
    distance=getDistance();
    if(distance<15){ //1초동안 계속 장애물이 유지된다면 후진 아니면 전진알고리즘 적용
      return false;
    }
    else {
      return true;
    }
  }
}
```

위 함수는 전방에 장애물이 있는지 확인하는 함수로써 초음파 센서로부터 장애물의 거리를 받아오고 그 거리가 15cm미만인 경우 false를 반환하는 기능을 가진 함수이다. 여기서 중요한 점은 테스트를 거쳐 실험자의 손이나 여러 오류 상황을 고려했다는 것이다. 초반과 달라진 점은 우선 장애물이 임계 거리안에 들어오면 정지를 하고 1초후 다시 거리를 측정하게 해서 만약 그때도 장애물이 임계 거리 안에 있다면 그제서야 false를 반환하게 하고, 그렇지 아니면 True를 반환하여 오류에 대한 대처를 하였다.

```
int getDistance() {
  delay(50);
  Int cm = sonar.ping_cm();
  if (cm == 0) {
    cm = 100;
  }
  returncm;
}
```

위 코드는 함수명 그대로 거리를 반환하는 함수로 sonar라는 객체에 대해 거리를 불러오고 거리를 반환하게 하였다. 여기서 발생할 수 있는 오류를 제거하기 위해 거리가 0으로 측정될 경우 100츠로 수정하여 반환하도록 조치하였다.

3. 메인부분인 루프 함수

다음 설명할 부분은 셋업 함수와 메인인 루프 함수이다. 셋업 함수란 핀번호의 역할을 정해주고 모터의 스피드 등 여러 가지 기본 세팅을 하는 부분이며, 루프 함수란 계속 함수 자신을 반복하는 기능을 수행하는 함수이다.

```
void setup() {
  Serial.begin(9600);
  pinMode(irLeft, INPUT);
  pinMode(irRight, INPUT);
  pinMode(irbLeft, INPUT);
  pinMode(irbRight, INPUT);
  pinMode(LED_PIN,OUTPUT);

  motor1.setSpeed(93);
    motor2.setSpeed(93);
    motor3.setSpeed(93);
    motor4.setSpeed(93);
}
```

위 셋업함수를 보면 serial 통신속도를 정하는 serial.begin과 핀모드를 설정하는 부분으로 구성되어 있다. IrLeft~irbRight를 모드 pinMode함수를 이용해 input으로 설정하였는데 이는 적외선 센서 부분에 해당한다. 그 다음 Led_pin은 LED를 조정하는 pin으로 OUTPUT으로 설정하여 + 전기 신호를 방출하도록 하였다. 여기서 중요한점은 모터 쉴드의 핀 개수가 부족하여 서보모터 핀까지 사용했다는 점이다. 서보모터 핀도

역시나 전기적 핀이기에 여타 다른 핀과 동일하게 사용해도 문제없다는 것을 이용하였다.

```
void loop() {
  if (isPathClear()) {
    if (digitalRead(irLeft)== 0&& digitalRead(irRight)== 0) {
      moveForward();
    } else if (digitalRead(irLeft)== 0&& digitalRead(irRight)== 1) {
      Serial.println("TL");
      moveLeft();
    } else if (digitalRead(irLeft)== 1&& digitalRead(irRight)== 0 ){
      Serial.println("TR");
      moveRight();
    } else if (digitalRead(irLeft)== 1&& digitalRead(irRight)== 1) {
      Stop();
      delay(100);
    }
  } else {
    motor1.setSpeed(93);
    motor2.setSpeed(93);
```

```
    motor3.setSpeed(93);
    motor4.setSpeed(93);
    intcheck=0;
    while (1) {
       digitalWrite(LED_PIN, HIGH);

      bLeftValue = digitalRead(irbLeft);
      bRightValue = digitalRead(irbRight);
      if (bLeftValue == 0&& bRightValue == 0) {
        moveBackward();
      } else if (bLeftValue == 0&& bRightValue == 1) {
        motor1.run(BACKWARD);
        motor2.run(BACKWARD);
        motor3.run(FORWARD);
        motor4.run(FORWARD);
      } else if (bLeftValue == 1&& bRightValue == 0) {
        motor1.run(FORWARD);
        motor2.run(FORWARD);
        motor3.run(BACKWARD);
        motor4.run(BACKWARD);
      } else {
        Stop();
        dest();
        delay(1000);
        //여기에 종료음 넣을 예정
        break;
      }
    }
   digitalWrite(LED_PIN, LOW);
  }
}
```

위의 루프함수를 설명하면 우선 isPathClear함수가 True일 때와 아닐 때로 구분되어있는 것을 볼 수 있다. 우리는 후진이라는 기능을 택하였기에 위 작업은 필연적이었다. 바로 전진시에 후방 적외선 센서의 간섭 문제와 후진시의 전방 적외선 센서의 간섭 문제이다. 이를 해결하기위해 isPathCLear함수를 도입하여 만약 장애물이 감지되면 else를 따라 후진하게 하였다. 그리고 후진 시 루프 한번 돌때마다 LED가 점멸하게 하여 심미적으로 보이게 하였다. 그리고 전진 후진 시 적외선 센서를 이용하여 왼쪽 적외선이 1을 출력하면 왼쪽으로 반대의 경우에는 반대로 가게 하였다. 그리고 마지막 도착시에 좌우 적외선 센서가 1을 리턴 시 멈추고 dest()함수를 호출하고 10초 후 while문을 탈출하여 처음 세팅

으로 돌아가게 하였다 이렇게 함으로써 테스트를 쉽게 하였다.

다음은 전체 코드이다.

```
#include <NewPing.h>
#include <AFMotor.h>
// hc-sr04 sensor
#define TRIGGER_PINA0
#define ECHO_PINA1
#define max_distance 50
// ir sensor
#define irLeftA5
#define irRightA4
#define irbLeftA2
#define irbRightA3
#define LED_PIN 10
#define SPEAKER_PIN 11


NewPing sonar(TRIGGER_PIN, ECHO_PIN, max_distance);
AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(4, MOTOR12_1KHZ);
AF_DCMotor motor3(2, MOTOR34_1KHZ);
AF_DCMotor motor4(3, MOTOR34_1KHZ);
intdistance = 0;
intleftDistance;
intrightDistance;
boolean object;
intbLeftValue;
intbRightValue;
void setup() {
  Serial.begin(9600);
  pinMode(irLeft, INPUT);
  pinMode(irRight, INPUT);
  pinMode(irbLeft, INPUT);
  pinMode(irbRight, INPUT);
  pinMode(LED_PIN,OUTPUT);

  motor1.setSpeed(93);
    motor2.setSpeed(93);
    motor3.setSpeed(93);
    motor4.setSpeed(93);
}
void loop() {
  if (isPathClear()) {
    if (digitalRead(irLeft)== 0&& digitalRead(irRight)== 0) {
      moveForward();
    } else if (digitalRead(irLeft)== 0&& digitalRead(irRight)== 1) {
      Serial.println("TL");
      moveLeft();
    } else if (digitalRead(irLeft)== 1&& digitalRead(irRight)== 0 ){
      Serial.println("TR");
      moveRight();
```

```
    } else if (digitalRead(irLeft)== 1&& digitalRead(irRight)== 1) {
      Stop();
      delay(100);
    }
  } else {
    motor1.setSpeed(93);
    motor2.setSpeed(93);
    motor3.setSpeed(93);
    motor4.setSpeed(93);
    intcheck=0;
    while (1) {
       digitalWrite(LED_PIN, HIGH);

      bLeftValue = digitalRead(irbLeft);
      bRightValue = digitalRead(irbRight);
      if (bLeftValue == 0&& bRightValue == 0) {
        moveBackward();
      } else if (bLeftValue == 0&& bRightValue == 1) {
        motor1.run(BACKWARD);
        motor2.run(BACKWARD);
        motor3.run(FORWARD);
        motor4.run(FORWARD);
      } else if (bLeftValue == 1&& bRightValue == 0) {
        motor1.run(FORWARD);
        motor2.run(FORWARD);
        motor3.run(BACKWARD);
        motor4.run(BACKWARD);
      } else {
        Stop();
        dest();
        delay(1000);
        //여기에 종료음 넣을 예정
        break;
      }
    }
   digitalWrite(LED_PIN, LOW);
  }
}
void dest() {
  intcnt=0;

  while(cnt<30){
  digitalWrite(LED_PIN, HIGH);
  delay(500);
  digitalWrite(LED_PIN, LOW);
  delay(500);
  cnt+=1;
  }
}

boolean isPathClear() {
  intdistance = getDistance();
  if (distance >= 15) {
    return true;
```

```
  }
  else {
    digitalWrite(LED_PIN, HIGH);
    Stop();
    delay(100);
    distance=getDistance();
    if(distance<15){ //1초동안 계속 장애물이 유지된다면 후진 아니면 전진알고리즘 적용
      return false;
    }
    else {
      return true;
    }
  }
}
int getDistance() {
  delay(50);
  intcm = sonar.ping_cm();
  if (cm == 0) {
    cm = 100;
  }
  returncm;
}
void Stop() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}
void moveForward() {
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
void moveBackward() {
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
}
void moveRight() {
  motor1.setSpeed(115);
  motor2.setSpeed(115);
  motor3.setSpeed(115);
  motor4.setSpeed(115);
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
void moveLeft() {
    motor1.setSpeed(115);
  motor2.setSpeed(115);
```

```
  motor3.setSpeed(115);
  motor4.setSpeed(115);
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
```

4. 기능적인 함수

다음에 설명할 함수는 dest()함수로 도착시 LED를 점멸하게 하는 함수이다.

```
void dest() {
  intcnt=0;

  while(cnt<30){
  digitalWrite(LED_PIN, HIGH);
  delay(500);
  digitalWrite(LED_PIN, LOW);
  delay(500);
  cnt+=1;
  }
```

위 함수는 도착 시 LED를 500ms 간격으로 30번 점멸하는 기능을 가진 함수이다.

다음은 전체 코드이다.

```
#include <NewPing.h>
#include <AFMotor.h>
// hc-sr04 sensor
#define TRIGGER_PINA0
#define ECHO_PINA1
#define max_distance 50
// ir sensor
#define irLeftA5
#define irRightA4
#define irbLeftA2
#define irbRightA3
#define LED_PIN 10
#define SPEAKER_PIN 11

NewPing sonar(TRIGGER_PIN, ECHO_PIN, max_distance);
AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(4, MOTOR12_1KHZ);
AF_DCMotor motor3(2, MOTOR34_1KHZ);
```

```
AF_DCMotor motor4(3, MOTOR34_1KHZ);
intdistance = 0;
intleftDistance;
intrightDistance;
boolean object;
intbLeftValue;
intbRightValue;
void setup() {
  Serial.begin(9600);
  pinMode(irLeft, INPUT);
  pinMode(irRight, INPUT);
  pinMode(irbLeft, INPUT);
  pinMode(irbRight, INPUT);
  pinMode(LED_PIN,OUTPUT);

  motor1.setSpeed(93);
    motor2.setSpeed(93);
    motor3.setSpeed(93);
    motor4.setSpeed(93);
}
void loop() {
  if (isPathClear()) {
    if (digitalRead(irLeft)== 0&& digitalRead(irRight)== 0) {
      moveForward();
    } else if (digitalRead(irLeft)== 0&& digitalRead(irRight)== 1) {
      Serial.println("TL");
      moveLeft();
    } else if (digitalRead(irLeft)== 1&& digitalRead(irRight)== 0 ){
      Serial.println("TR");
      moveRight();
    } else if (digitalRead(irLeft)== 1&& digitalRead(irRight)== 1) {
      Stop();
      delay(100);
    }
  } else {
    motor1.setSpeed(93);
    motor2.setSpeed(93);
    motor3.setSpeed(93);
    motor4.setSpeed(93);
    intcheck=0;
    while (1) {
       digitalWrite(LED_PIN, HIGH);

      bLeftValue = digitalRead(irbLeft);
      bRightValue = digitalRead(irbRight);
      if (bLeftValue == 0&& bRightValue == 0) {
        moveBackward();
```

```
      } else if (bLeftValue == 0&& bRightValue == 1) {
        motor1.run(BACKWARD);
        motor2.run(BACKWARD);
        motor3.run(FORWARD);
        motor4.run(FORWARD);
      } else if (bLeftValue == 1&& bRightValue == 0) {
        motor1.run(FORWARD);
        motor2.run(FORWARD);
        motor3.run(BACKWARD);
        motor4.run(BACKWARD);
      } else {
        Stop();
        dest();
        delay(1000);
        //여기에 종료음 넣을 예정
        break;
      }
    }
   digitalWrite(LED_PIN, LOW);
  }
}
void dest() {
  intcnt=0;

  while(cnt<30){
  digitalWrite(LED_PIN, HIGH);
  delay(500);
  digitalWrite(LED_PIN, LOW);
  delay(500);
  cnt+=1;
  }
}

boolean isPathClear() {
  intdistance = getDistance();
  if (distance >= 15) {
    return true;
  }
  else {
    digitalWrite(LED_PIN, HIGH);
    Stop();
    delay(100);
    distance=getDistance();
    if(distance<15){ //1초동안 계속 장애물이 유지된다면 후진 아니면 전진알고리즘 적용
      return false;
    }
    else {
```

```
      return true;
    }
  }
}
int getDistance() {
  delay(50);
  intcm = sonar.ping_cm();
  if (cm == 0) {
    cm = 100;
  }
  returncm;
}
void Stop() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}
void moveForward() {
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
void moveBackward() {
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
}
void moveRight() {
  motor1.setSpeed(115);
  motor2.setSpeed(115);
  motor3.setSpeed(115);
  motor4.setSpeed(115);
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
void moveLeft() {
    motor1.setSpeed(115);
  motor2.setSpeed(115);
  motor3.setSpeed(115);
  motor4.setSpeed(115);
  motor1.run(FORWARD);
  motor2.run(FORWARD);
```

```
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
}
```

참고문헌 :

1. 김철민, 신호준, 조희영, 조희영, 윤태성. (2022). RFID 기반 최단시간 알고리즘 라인트레이서. 한국전자통신학회 논문지, 17(6), 1221-1228. https://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE11262728

2. 러봇랩. [아두이노기초] 거리감지 센서의 종류와 원리 | 초음파 센서, 적외선 센서, 레이저 센서 https://youtu.be/io7RqNrnGpI?si=yVB3531seymtVgBJ

3. 미래인재연구소. 아두이노고급 2강 아두이노 이론 액츄에이터 사용하기 https://www.youtube.com/watch?v=AV2QZ6zT9sQsssssssssss

4. DIY Builder. How To Make A DIY Arduino Line Follower Car At Home https://www.youtube.com/watch?v=t7k9D1jDEtk

5. Mr ProjectsoPedia. Arduino obstacle avoiding line follower robot projects code 2021 https://www.youtube.com/watch?v=ZiqAyuLpS3o