

창의설계입문 프로젝트 설계서						
프로젝트 제목	라인트레이서 제작				작성년월일	2020/11/07
프로젝트 팀 이름	학번	201921120	-	-	-	
1등조	성명	우다현	000	000	000	
	역할	팀장 및 전략 및 프로그래밍 및 문서작성	구조물 설계 및 도안 정리	작동 시나리오 구성 및 작성	발표 및 문서작성	

1. 요약

가. 요약

본 프로젝트는 컬러센서를 이용하여 주어진 길을 따라서 출발지에서 목적지까지 운행하는 라인트레이서를 제작하는 것이다. 이 때, 주어진 길이 꺾이는 부분이 많고, 짧은 선으로 된 장애물이 있으며, 길과 길 혹은 길과 장애물 사이의 거리가 좁아 길을 잘못 인식할 수도 있으므로 이러한 사소하지만 심각한 문제들을 개선한 라인트레이서를 만드는 것이 이번 프로젝트의 관건이 될 것이다. 선을 이탈할 경우 감점이 있을 것이고, 주행시간이 빠를수록 유리할 것이다.

나. 전략

본 설계에는 bric 본체, 컬러 센서 1개, Large Motor 2개, 뒷바퀴, 연결하는 봉 여러 개를 사용하였다. 컬러 센서는 검은 선과 바닥을 구분한다. 모터는 바퀴 구동을 위해 2개 모두 사용하였다.

설계 시, 유의해야 할 부분은 다음과 같다. 첫째, 센서 인식이 잘되어야 한다. 둘째, 차체가 선을 완전히 벗어나면 안 된다. 셋째, 선과 가까이에 존재하는 짧은 선의 장애물을 인식했을 때 멈추어야 한다.

이를 해결하는 방법은 다음과 같다. 첫째, 제품을 설계할 때 센서 앞을 다른 부품으로 가리지 않도록 하고 센서 각도를 잘 조절해야 한다. 둘째, 선이 꺾이는 부분에서의 회전 각도를 잘 조절해야 하며 선과 선 사이의 거리가 짧을 때 다른 선을 인식하지 않도록 해야 한다. 셋째, 선과 가까이에 존재하는 짧은 선의 장애물을 인식했을 때 멈추는 기능을 추가한다.

차체 조립 방법은 다음과 같다. 먼저, Large Motor를 bric 본체에 흔들리지 않게끔 조립한다. 모터에 바퀴를 설치할 수 있도록 봉을 끼워준다. 다른 모터에도 봉을 연결한다. 사각형 블록을 이용하여, 이어붙여 두 모터를 이어붙여 고정하여 준다. 그리고 모터가 잘 고정될 수 있도록 아래쪽에도 H모양의 블록으로 고정하여 준다. 다음으로, 브릭의 뒤쪽과 모터를 고정해줄 연결부를 조립한다. 그 후 두 모터와 브

릭을 연결해 보도록 한다. 옆면도 고정해준다. 모터와 브릭 본체의 연결을 완료하고 마지막으로 컬러 센서를 브릭 앞단에 연결해준다. 컬러 센서는 하나만 사용한다.

컬러센서의 위치를 결정하기 위해 컬러센서의 위치를 가운데, 차체에서 왼쪽, 차체에서 오른쪽으로 바꾸어보며 여러 맵에서 시도를 해보았다. 그 결과 차체에서 왼쪽에 컬러센서를 위치하게 하는 것이 가장 인식을 잘했고, 컬러센서의 위치를 차체에서 왼쪽으로 결정하게 되었다. 하지만 경연맵의 모양에 따라 컬러센서의 위치는 달라질 수도 있을 것이다.

이후 뒷바퀴가 부실하여 속도가 느려지는 것을 고려해 뒷바퀴를 새 것으로 교체하였다.

프로그램 설계를 위해서는 컬러센서가 인식한 값과 그에 따른 차체 회전각도의 조절이 중요하다. 그래서 간단한 라인트레이싱 프로그램을 작성하고 라인을 따라가면서 컬러센서에서 인식한 값을 모두 출력하는 코드를 넣어 실험해 보았다. 우리 조에서는 이 과정을 동영상으로 녹화했고 인식값을 분석하며 문턱값과 회전각도를 어떻게 조절할지 분석해보았다. 경연맵 2개로 15번의 시도를 했고, 그 중 완주하지 못한 경연맵의 5개 동영상을 분석해서 문턱값과 회전각도를 결정하였다.

이후 회전각도와 기준값 그리고 기본동력 등의 관계를 선형적인 식으로 나타내서 회전각도를 조절해 보았지만 예행 연습을 할 시간이 부족해서 선을 벗어나는 일이 발생해 코드를 사용하지 못하였다. 또한, 정지선에서 멈추는 코드도 구현하였으나 연습 시간이 부족했고 라인을 따라가는 중간에 멈출 가능성이 있었기에 사용하지 않았다.

경연 날에는 기존의 코드에서 회전각도를 조금씩 키워가며 선을 벗어나지 않는 최고 각도로 수정하여 경연을 진행했다. 그 결과 58초라는 비교적 빠른 속도로 경연맵을 완주할 수 있었다.

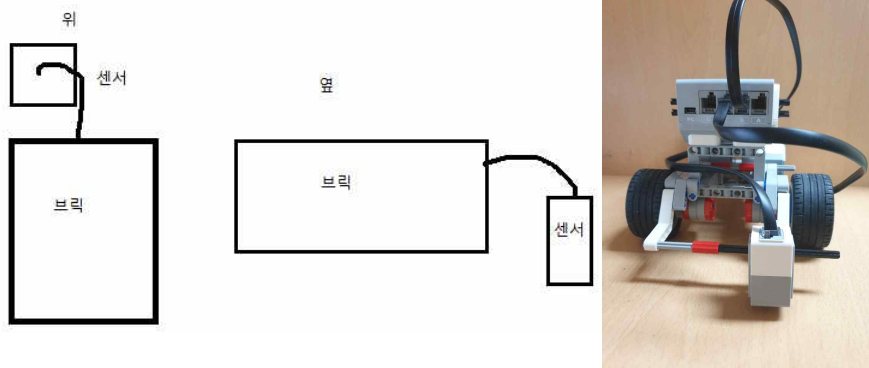
2. 제품 구조도

가. 제품명 : 라인만 보인단 말이야

나. 주요 부위별 설계 내용 (부위별 도안 포함)

(1) 센싱부

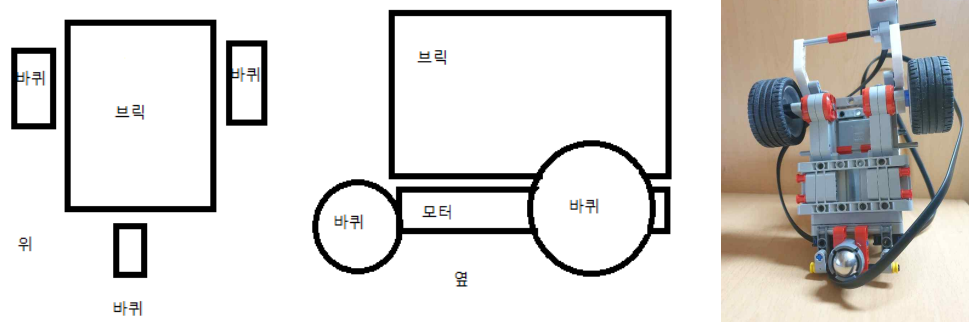
1개의 라이트 센서를 왼쪽 바퀴 앞에 설치하여 센서가 라인을 감지하도록 함. 라이트 센서가 바닥을 향하게 설치하여 바닥에 있는 라인의 색을 인지하여 검정색 라인을 따라갈 수 있게 함.



(그림 1. 사진1. 센싱부 도안 및 사진)

(2) 구동부

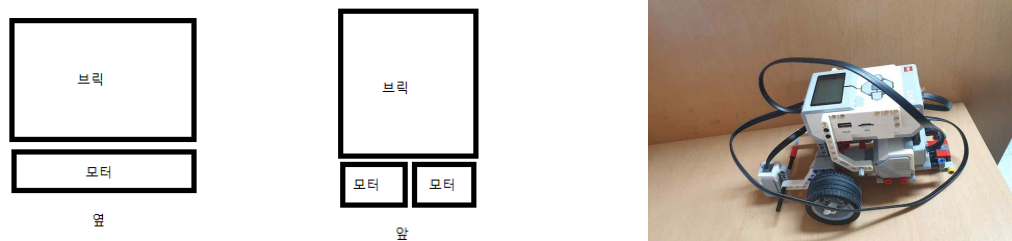
2개의 서보 모터를 사용하여 왼쪽, 오른쪽에 바퀴를 연결하여 전진, 좌회전, 우회전이 가능하도록 함.



(그림 2. 사진2. 구동부 도안 및 사진)

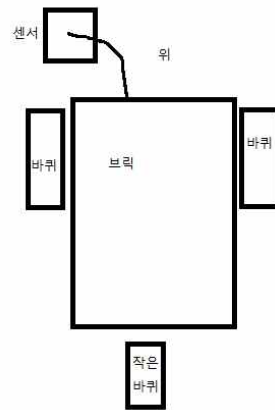
(3) 차체

ev3 브릭의 위치가 차체의 중심에 오도록 하고, 위쪽으로 이동하도록 장착하여 브릭조작을 편리하게 한다. 구동부는 브릭 바로 아래에 연결하고, 센싱부는 오른쪽 바퀴 앞에 연결한다.

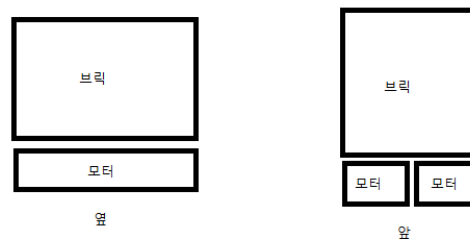


(그림 3. 사진3. 차체 도안 및 사진)

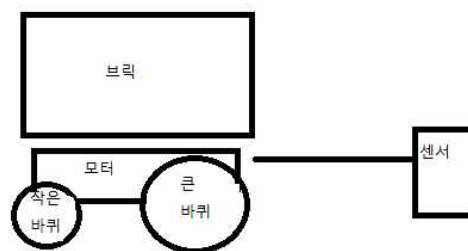
(4) 전체 제품 구조도



(그림 4. 제품 구조 설계 도안: 평면)



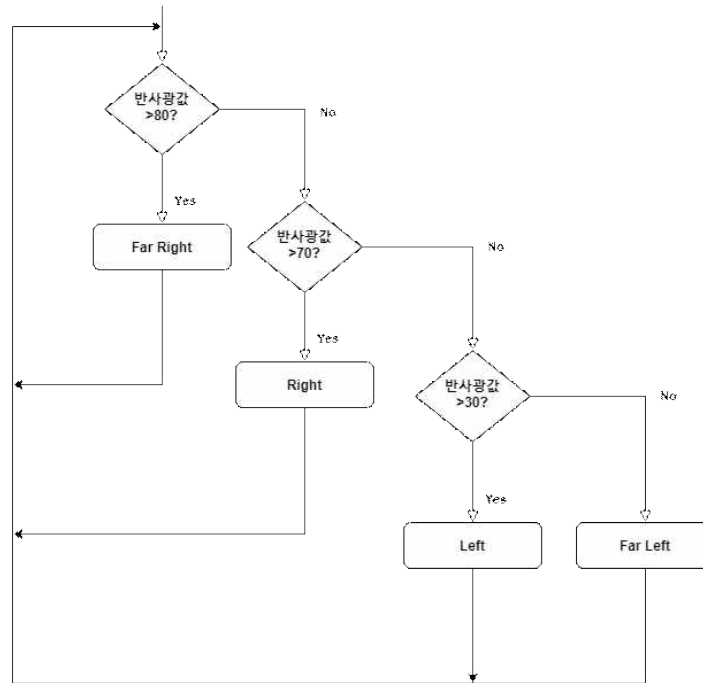
(그림 5. 제품 구조 설계 도안: 정면)



(그림 6. 제품 구조 설계 도안: 측면)

3. 프로그램 설계

가. 최종 작동 시나리오



(그림 7. 프로그램 순서도)

나. 최종 설계된 프로그램 사진

```

1  task main()
2  {
3      while(1)
4      {
5          writeDebugStreamLine("%d", SensorValue[S2]);
6
7          if(SensorValue[S2]>80) //100~80, A;
8          {
9              motor[motorB]=60;
10             motor[motorC]=-10;
11         }
12         else if(SensorValue[S2]>70) //80~70, c0E;
13         {
14             motor[motorB]=55;
15             motor[motorC]=-5;
16         }
17         else if(SensorValue[S2]>30) //70~30, c0E;
18         {
19             motor[motorB]=-5;
20             motor[motorC]=55;
21         }
22         else if (SensorValue[S2]>0) //30~0, ;
23         {
24             motor[motorB]=-10;
25             motor[motorC]=60 ;
26         }
27     }
28 }
  
```

(그림 8. 설계된 프로그램 모습)

다. 프로그램 설계 내용

컬러센서의 반사광 값이 문턱 값(밝은지 어두운지)보다 작다면 오른쪽으로 회전하고, 컬러센서의 반사광 값이 문턱 값보다 크다면 라인이 없는 곳으로 인식해서 왼쪽으로 회전한다. 회전하기 위해서는 두 개의 바퀴 속도를 다르게 하여 회전할 것이다. 왼쪽으로 회전하기 위해서는 오른쪽에 있는 바퀴의 속도를 더 크게 할 것이고, 오른쪽은 그 반대로 설정했다.

I. 제품 구조의 주요 부위와 함수와의 1:1 연결성

센싱부는 컬러 센서에서 센싱을 하므로 컬러 센서 함수를 사용하여 반사광 값을 인식한다. 구동부는 2개의 라지 모터 구동을 한다. 포트 B, C에 모터 2개를 각각 하나씩 지정하였다. 그리고 motor 함수를 사용하여 모터속도를 다르게 지정하여 앞으로 나아가되 라인을 벗어나지 않게 한다.

II. 작동 시나리오(순서도)와의 함수와의 1:1 연결성

(1) 컬러 센서에서 센싱을 하므로 if 구조문을 이용하여 컬러 센서의 반사광 모드 의 함수 값과 문턱 값을 비교한다. 이때 조건문을 하나만 쓰기보다는 여러개의 else if 문을 사용함으로써 좀 더 세세한 조건을 추가하였다. 그 결과 하나만 썼을 때 보다 정확하게 라인을 따라가도록 하였다.

(2) 이후 좌회전, 우회전을 반복하게 한다. 일반적으로 문턱값보다 크다면 우회전을 하고 주어진 값보다 작다면 좌회전을 하게 하였다. 이때, 좌회전시 우측 모터는 전진, 좌측 모터는 후진을 한다. 동일하게 우회전시 좌측 모터는 전진, 우측 모터는 후진을 한다. 즉 양쪽 바퀴를 서로 반대 방향으로 회전시키는 Point Turn 회전 방법을 이용하여 빠르게 방향 변경이 가능하도록 하였다.

(3) 좀 더 개선하기 위하여 기준 문턱값을 여러개로 지정하였다. display 출력함수를 이용하여 라인을 따라갈 때 출력값도 따로 Debug Stream에 출력하여 분석을 해보았다. 그 결과 출력값이 80~100 일때는 완전 종이 위에 있게되었다. 이때는 오른쪽으로 크게 회전하도록 하였다. 출력값이 70~80일때는 종이 위에 있었고 이때는 비교적 작게 오른쪽으로 회전하도록 하였다. 출력값이 30~70일때는 선과 종이의 중간에 있었고 이때는 왼쪽으로 회전하게 하였다. 마지막으로 0~30일때는 완전히 선 안으로 들어왔고 이때는 크게 왼쪽으로 회전하도록 하였다. 중간 값을 30~70으로 설정하게 된 이유는 데이터를 토대로 분석하였을 때, 밝은 값은 70~90이 나왔고, 어두운 값은 10~30의 값이 나왔기 때문이다.

라. 프로그램 개선 과정

경연에서의 목표는 우리 조의 라인트레이서가 정확하게 선을 따라가고, 빠른 속도로 완주하는 것이었다.

4번의 기회 중에 첫 번째 기회에는 (그림9. 설계된 프로그램 후보1)로 진행했다. 그리고 빠르지 않은 속도이지만 완주에 성공하였다.

두 번째와 세 번째 기회에서는 새롭게 구현한 코드로 (그림 10. 설계된 프로그램 후보2)를 값을 바꿔가며 경연을 진행했다. (그림10)은 회전 각도와 기준값 그리고 기본동력 등의 관계를 선형적인 식으로 표현한 코드이다. 이전에 엑셀로 문턱값과 회전 각도를 선형적으로 분석해본 경험과 자료조사를 코드를 작성해 본 것이다. C 언어로 visual studio 프로그램에서 printf 함수로 미리 값들을 출력해보며 코딩해보고 정상적으로 작동하는 것을 확인한 코드이지만, 실제로 본체를 가지고 코드를 돌렸을 때 선을 잘 벗어나는 문제점을 보였다. 또한 (그림 11. 정지코드 후보 1),(그림 12. 정지코드 후보 2)로 정지선에서 정지하는 코드도 구현을 해보았다. (그림 11) 흰색 바탕이 계속 노출되면 정지하는 코드이고, (그림11)은 정확히 정지선 사이를 지나가는 시간을 측정하여 그 시간만큼 기다렸다가 다시 한 번 선이 인식되면 정지하는 코드이다. 이러한 코드들은 구현은 해보았지만 경연 도중 정지할 가능성이 있었기 때문에 경연에서 사용하지 않기로 결정하였다.

여러 시행착오를 겪은 후 네 번째 기회에서는 (그림 9)의 코드를 다시 사용하였다. 첫 번째 경주에서 다른 조와 비교하였을 때, 시간이 오래 걸린다는 점을 인지했던 우리 조는 값을 조정하였다. 그 결과 최선의 속도인 (그림 8)의 결과를 만들 수 있었다. 설명을 덧붙이자면, 큰 회전에서는 기존 50~-20의 값에서 60~-10으로 수정하였더니 속도는 개선되고 회전각은 많이 틀어지지 않는 좋은 결과를 가져왔다. 작은 회전도 위와 동일한 방법으로 기존 45_-15의 값에서 55_-5으로 변경하였다. 추가적으로 반사광 값도 수정을 해보았으나 기존의 값이 정확함을 인지하여 기존의 값을 그대로 사용하였다.

```

1 task main()
2 {
3     while(1)
4     {
5         writeDebugStreamLine("%d", SensorValue[S2]);
6
7         if(SensorValue[S2]>80) //80~100
8         {
9             motor[motorB]=50;
10            motor[motorC]=-20;
11        }
12        else if(SensorValue[S2]>70) //80~70
13        {
14            motor[motorB]=45;
15            motor[motorC]=-15;
16        }
17        else if(SensorValue[S2]>30) //70~30
18        {
19            motor[motorB]=-15;
20            motor[motorC]=45;
21        }
22        else if (SensorValue[S2]>0) //30~0
23        {
24            motor[motorB]=-20;
25            motor[motorC]=50;
26        }
27    }
28 }

```

(그림 9. 설계된 프로그램 후보1)

```

1 int LR; //left rotate
2 int RR; //right rotate
3 int threshold = 45; // 기준값
4 int base = 20; //기준 출력
5 float rotation;
6 float r = 0.6;
7
8 task main()
9 {
10     while (1) {
11         rotation = r * (threshold - SensorValue[S2]);
12
13         LR = base + rotation;
14         RR = base - rotation;
15
16         motor[motorB] = LR;
17         motor[motorC] = RR - 10;
18     }
19 }
20

```

(그림 10. 설계된 프로그램 후보2)


```

1
2 task main()
3 {
4     if (SensorValue[S2]>95)
5     {
6         wait1msec(1500);
7         if (SensorValue[S2]>95)
8         {
9             motor[motorB] = 0;
10            motor[motorC] = 0;
11        }
12    }

```

(그림 11. 정지코드 후보 1)



```

1
2 task main()
3 {
4     if (SensorValue[S2]<10)
5     {
6         wait1msec(1000);
7         if (SensorValue[S2]<10)
8         {
9             motor[motorB] = 0;
10            motor[motorC] = 0;
11        }
12    }

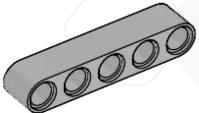
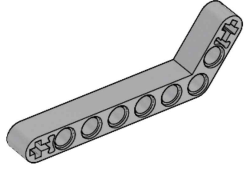
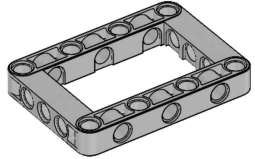
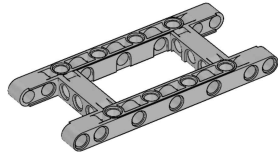
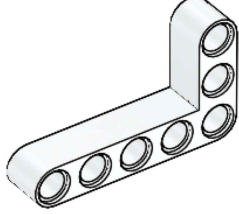
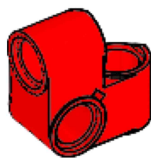

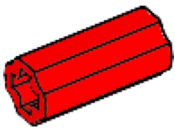
```

(그림 12. 정지코드 후보 2)

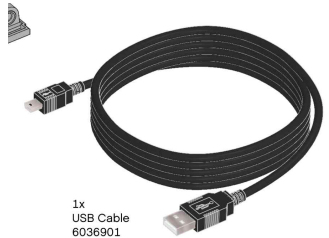
5. 부품표

부품번호	부품명	수량	이미지
4142822	Beam, 3-module, black	1	
4121715	Connector peg with friction, 2-module, bl ack	19	

4535768	Axle, 9-module ,grey	2	
4211815	Axle, 3-module ,grey	1	
4206482	Connector peg with friction/axle, 2-mo dule, blue	16	
4514553	Connector peg with friction,3/module,blu e	2	
4140806	Connector peg with bushing, 3-module, red	4	
4211639	Axle, 5-module,grey	4	
370626	Axle, 6-module, black	1	
6006140	Beam with crosshol e, 2-module, black	2	
4509912	Angular beam, 4*4-m odule, white	2	

4211651	Beam, 5-module ,grey	1	
4211624	Angular beam, 3*7- module, grey	2	
4539880	Frame, 5*7-module ,grey	1	
4540797	Frame, 5*11-modul e, gery	1	
4585040	Angular beam, 3*5 -module, white	2	
6008527	Cross beam, 2*1- module, red	2	
4563044	Connector peg with handle, black	1	
4513174	Bushing/axle exten der, 2-module, red	1	

4634091	Hub, 43.2*26 mm, grey	2	
6035364	Low profile tire, 56*28 mm, black	2	
6009996	EV3 Brick	1	
6008919	Colour Sensor	1	
6009430	Large Motor	2	
6024583	Cable, 35cm/14in	2	<div>35 cm / 14 in.</div> 
6024585	Cable, 50cm/20in	1	<div>50 cm / 20 in.</div> 

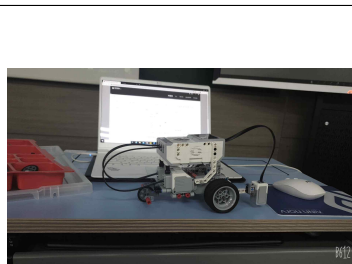
6036901	USB Cable	1	 <p>1x USB Cable 6036901</p>
---------	-----------	---	---

6. 제작 과정 기록

차체 조립 방법을 다음과 같다. 먼저, Large Motor를 bric 본체에 흔들리지 않게끔 조립한다. 모터에 바퀴를 설치할 수 있도록 봉을 끼워준다. 다른 모터에도 봉을 연결한다. 사각형 블록을 이용하여, 이어붙여 두 모터를 이어붙여 고정하여 준다. 그리고 모터가 잘 고정될 수 있도록 아래쪽에도 H모양의 블록으로 고정하여 준다. 다음으로, 브릭의 뒤쪽과 모터를 고정해줄 연결부를 조립한다. 그 후 두 모터와 브릭을 연결해 보도록 한다. 옆면도 고정해준다. 모터와 브릭 본체의 연결을 완료했고 마지막으로 컬러 센서를 브릭 앞단에 연결해준다. 컬러 센서는 하나만 사용한다.

컬러센서의 위치를 결정하기 위해 컬러센서의 위치를 가운데, 차체에서 왼쪽, 차체에서 오른쪽으로 바꾸어보며 여러 맵에서 시도를 해보았다. 처음에는 컬러센서의 위치가 차체에서 오른쪽이었으나, 여러 시도 결과 차체에서 왼쪽에 컬러센서를 위치하게 하는 것이 가장 인식을 잘했다. 그래서 컬러센서의 위치를 차체에서 왼쪽으로 결정하게 되었다. 즉, 센서를 라인트레이서의 왼쪽에 부착해야 라인트레이서가 선의 중앙에 위치할 것으로 생각하여 센서를 왼쪽 바퀴 앞으로 옮긴 것이다.

이후 뒷바퀴가 부실하여 속도가 느려지는 것을 고려해 작은 뒷바퀴를 구슬 바퀴로 교체하여 360도 회전을 가능하도록 했다.



(사진 4. 변경 전)



(사진 5.변경 후1)



(사진 6.변경 후2)

7. 결과 토의

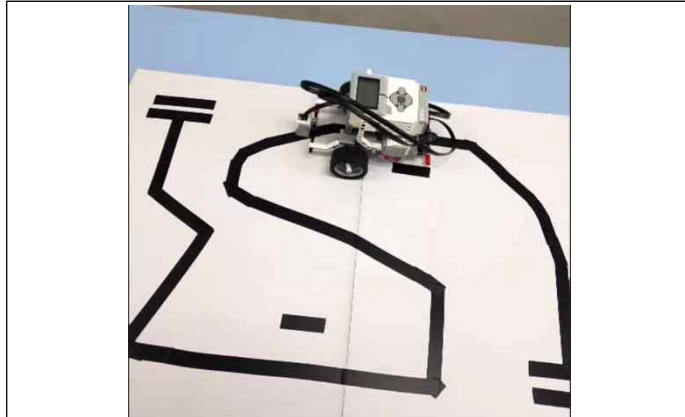
우선 경연 전까지의 진행상황에 관해 이야기해보고자 한다. 10/31에 첫 번째 대면수업에서는 기본적인 부품과 센서에 대해 이해했고 차체조립을 시작했다. 11/6 일에는 추가만남을 가져 의사소통을 하고 차체 조립방법에 대해 생각해보았다. 11/7일 두 번째 대면 수업에서는 차체조립을 완전히 끝냈고 다음 만남까지 해야할 일을 정리하고 이에 대해 역할분담하였다. 11/13일에는 추가만남을 가지고 조원들이 각자 해온 일을 토대로 자체제작한 경연맵에서 경연을 해보며 문제점을 찾고 추가적인 프로그래밍을 했다. 11/14 대면수업에서는 제공된 경연맵에서 연습을 했고, 문턱값과 회전각도를 조절하면서 경연맵을 완주하도록 노력하였다. 또한 이 때 과정을 녹화하여 어느 부분에서 문제가 생기는지 분석하고 프로그래밍적인 개선방안을 생각해보았다. 11/20일에 추가만남을 가지고 마지막으로 경연맵을 만들어서 경주를 해보았다. 그리고 경연 전날까지 라인을 정확하게 따라갈 수 있도록 개선점을 찾아보았다.

경연맵을 정확하고 빠르게 완주하기 위해서는 안정적인 차체와 체계적인 프로그래밍이 필요하다. 그래서 현재 진행상황에 대해 조립과 프로그래밍으로 나누어 설명해 볼 것이다.

블랙보드 강의를 참고해서 조립한 후 차체에 대해 개선점을 생각해보았을 때, 가장 중요했던 부분은 컬러센서의 위치였다. 컬러 센서의 정확한 인식을 위해서 최대한 바닥에 가깝게 수직각도로 설치하였다. 컬러센서를 하나만 사용하여 하기 때문에 차체의 중앙, 왼쪽, 오른쪽에 전부 설치를 하여 테스트를 해보았고, 왼쪽 바퀴에 최대한 가깝게 설치하는 것이 가장 정확했기에 왼쪽에 컬러센서를 설치했다. 또한 뒷바퀴를 구슬바퀴로 교체하여 360도 회전이 가능하도록 하였다.

프로그램을 설계하면서 발생했던 가장 큰 문제점은 차체가 각도가 큰 회전을 할 때 경로를 벗어나 라인 인식을 하지 못하고 회전하는 문제였다. 라인을 따라 정확하게 가는 것이 우선순위였기 때문에 우선은 회전 각도를 줄여 차체의 속도를 줄이고 라인을 최대한 벗어나지 않도록 하였다. 더 정확하게 이해하기 위해 반사광모드로 실시간으로 출력되는 값을 분석해보았다.

프로그래밍적인 해결방안을 찾기 위해 라인을 따라가면서 모든 반사광값을 출력하였다. 그리고 반사광값의 실시간 출력값을 녹화한 후 엑셀로 분석해보았다. 여러 경연 맵 중에 완주하지 못하는 경연맵을 골랐다. 그리고 15번의 실험을 했고 한 경연맵의 동영상 5개 표본을 골라 분석해보았다. 그 경연맵은 (사진 7. 연습 경연맵)과 같다.



(사진 7. 연습 경연맵)

[그림1]의 경연맵에서는 둔각으로 꺾이는 부분에서 차체가 180도 회전을 하는 문제가 지속적으로 발생하였다. 반사광값 출력값을 분석하고 회전각도를 조절해본 결과 이 문제의 원인은 다음과 같았다.

간단하게 생각하면 첫 째, 차체가 오른쪽 즉 바깥쪽으로 먼저 회전하는 것이었다. 왼쪽 즉 안쪽부터 회전을 시작했을 때는 선을 정확하게 인식하고 좌회전에 성공했다. 둘 째, 문턱값을 넘는 값, 즉 하얀색이 많이 인식되었을 때 회전각도가 너무 컸다. 분석해보았을 때 문턱값을 넘는 값이 대략 5번정도 연속했을 때 180도 회전을 하는 것을 알 수 있다. 그에 반해 문턱값을 넘지 않는 값, 즉 검정색이 많이 인식되었을 때는 거의 직선 운동을 하는 것으로 보아 문턱값을 넘을 때와 넘지 않을 때 회전각도의 괴리감이 있었음을 확인할 수 있었다. 회전각도가 문턱값을 넘을 때와 넘지 않을 때 괴리감이 있다면 회전각도가 큰 상황이 연속되었을 때 차체가 완전히 회전을 할 가능성이 있기에 조심해야한다.

이를 해결하는 방법은 무엇일까? 첫 번째와 같은 문제가 발생했을 때 선부터 인식했다면 종이부터, 종이부터 인식했다면 선부터 인식해서 회전방향을 바꾸는 것이다. 두 번째와 같은 문제가 발생하면 회전각도를 조절하면 된다. 여기서 회전각도는 두 라지 모터의 속도 차이값이다. 값 자체를 키우기만 하면 회전각도는 크게 변하지 않고 속도만 빨라진다. 그래서 위에서 분석한 결과에서 두 라지모터의 속도 차이값은 동일하게 유지하고 라지모터 속도의 값을 높여 속도를 빠르게 할 것이다. 또한, 문턱값의 기준을 세부적으로 나누면 좋다. 엑셀로 분석했을 때 종이를 인식했을 때 70~90의 값, 선을 인식했을 때 보통 30이하의 값을 인식했다. 그리고, 선과 종이 사이를 인식했을 때 대략 50 부근의 값을 인식했고, 이러한 값은 선을 인식했을 때와 같은 방향으로 회전하는 것이 유리했기에 선과 같은 방향으로 회전하게 했다. 그래서 우리조는 문턱값을 30,70,80으로 나누어서 회전각도를 조절했다. (0~30,30~70,70~80,80~100) 또한, 위에서 언급한 대로 문턱값을 넘을 때와 넘지 않을 때의 회전각도가 너무 크게 차이나지 않도록 조절하는 것이 좋을 것이다. 추가로, 보험이 있어도 좋을 것이다. 차체가 완전히 회전할 가능성이 있으므로 연속적으로 문턱값을 넘는 값이나 넘지 않는 값이 발생할 때 이에 따라 우회전이나 좌회

전을 하는 알고리즘을 만드는 것이다. 이러한 알고리즘은 c언어 list에 저장하는 방법을 통해 구현하는 것을 생각해볼 수 있다. 하지만 경연 맵에 따라 차체가 완전히 회전하는 데 필요한 문턱값을 초과하는 횟수가 달라질 수 있으므로 실현 가능성은 적어 보인다.

11/21 경연 당일, 우리 조는 가장 오랜 시간을 투자한 (그림9)의 첫 번째 코드로 완주를 목표로 시도하고, 두 번째 시도부터 새롭게 한 프로그래밍 (그림10)으로 속도와 정확성을 높여보는 것으로 전략을 세웠다.

첫 번째 시도에서는 2분에 가까운 도착 시간이었지만 완주에 성공했다. 두 번째 시도와 세 번째 시도에서는 (그림 10)의 코드로 경연을 진행했지만, 차체가 생각처럼 동작하지 않아 실격을 당했다. 결국 우리는 처음의 코드로 돌아와서 라지모터의 속도와 회전각도 그리고 문턱값을 조정하면서 최선의 속도를 찾았다. 그동안의 연습했던 경험을 총동원해서 효율적으로 값을 조정한 결과 마지막 시도에서는 58초를 기록하며 처음 시도했던 기록보다 1분 정도를 줄여 경연을 좋은 결과로 마무리할 수 있었다.