

창의설계입문 프로젝트 설계서

프로젝트 제목	인공생명체 로봇 만들기				작성년월일	2020/12/13
프로젝트 팀 이름	학번	201921120	-	-	-	
일등조	성명	우다현	000	000	000	
	역할	팀장 및 전략 및 프로그래밍 및 문서작성	구조물 설계 및 도안 정리	작동 시나리오 구성 및 작성	전략 및 문서작성	

1. 요약

본 프로젝트는 로봇에 코딩 프로그램을 주입시켜 생명체의 특징을 가지는 인공생명체 로봇을 구축하는 것이다. 이때 로봇이 가지는 특징은 주어진 필드에서 장애물을 인지하고 먹이를 먹은 후 집으로 다시 돌아오는 동작을 반복적으로 수행하게 만들어 주어진 먹이를 주어진 시간내에 최대한 많은 먹이를 먹게하는 것이 이번 프로젝트의 목표이다.

로봇이 필요로 수행해야 할 내용으로는 경기장 가장자리를 인식하여 밖으로 나가지 않게 해야 한다. 또한 먹이를 만나거나 집으로 돌아올때는 소리를 내야한다. 즉 생명체의 특성인 반응, 에너지 획득, 대사활동, 반응, 적응을 로봇이 보이게 한다.

2. 설계 내용

가. 동작 설명서

1) 전략 (Strategy)

1) 차체

본 설계에는 bric 본체, 컬러 센서 2개, 적외선 센서 1개, Large Motor 2개, 연결하는 봉 여러개를 사용하였다. 컬러 센서는 컬러 모드로 집인 파란색과 먹이인 빨간색 그리고 경연맵의 테두리인 검은색을 인식하기 위해 사용하였다. 적외선 센서는 장애물을 인식하고 피하기위해 사용하였다. 모터는 바퀴 구동을 위해 2개 모두 사용하였다.

설계 시, 유의해야 할 부분은 다음과 같다. 첫째, 센서 인식이 잘되어야 한다. 둘째, 장애물을 피해 먹이를 찾고 집으로 돌아오는 과정을 반복한다. 셋째, 차체가 경연맵을 벗어나지 않도록 한다.

이를 해결하는 방법은 다음과 같다. 첫째, 제품을 설계할 때 센서 앞을 다른 부품으로 가리지 않도록 하고 센서 각도를 잘 조절해야 한다. 둘째, 차체가 먹이(빨간점), 집(파란점) 그리고 장애물을 만나면 후진을 하는데 정확한 인식을 위해 컬러센서를 차체에 가깝게 설계한다. 셋째, 장애물의 높이를 고려하여 적외선센서의 위치를 잘 조절한다. 넷째, 차체가 경연맵을 벗어나지 않도록 센서들을 차체의 앞쪽에 설계한다.

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 센서 2개와 적외선 센서를 브릭 앞단에 연결해준다.

적외선 센서를 고정하여 설치하는게 어려워 다양한 방법을 시도해보았다. 처음에는 적외선 센서가 수평이 아닌 위쪽을 향하고 있었고, 그 다음은 아래쪽을 향하고 있었다. 그 뒤로 컬러센서부터 다시 설계를 하여 적외선센서가 장애물을 잘 인식할 수 있게 차체와 수평이 되도록 했다.

2) 프로그램

동작 프로그램을 작성할 때 전략을 여러 가지를 세워보았다. 많은 시행착오를 겪어 다음과 같은 결과를 낼 수 있었다.

첫 번째로, 직진해서 먹이를 찾은 후 후진을 해서 집으로 돌아오는 전략을 세웠다. 이렇게 프로그램을 작성해서 로봇을 동작시켜보았을 때 프로그램이 간단하고 집으로 쉽게 돌아올 수 있다는 장점이 있었지만, 센서가 앞쪽에 있었기 때문에 후진할 때 장애물을 인식하지 못하거나 경기장 밖으로 벗어나는 일이 발생하였다.

두 번째로, 경기장을 좌표로 생각해서 집과 먹이의 위치를 저장하고 동작하는 전략을 세워보았다. 행열로 경기장을 나누고 동작 시마다 좌표를 찍도록 프로그래밍을 해보았지만 PID control을 사용하지 않았으므로 차체가 완벽하게 원하는 대로 움직이지 않아 정확한 좌표를 찍지 못했다. 또한, 로봇은 경기장 정보에 대해서 모르며 무작위로 움직인다는 조건에 위배되는 부분이 있었기에 이 전략은 보류했다.

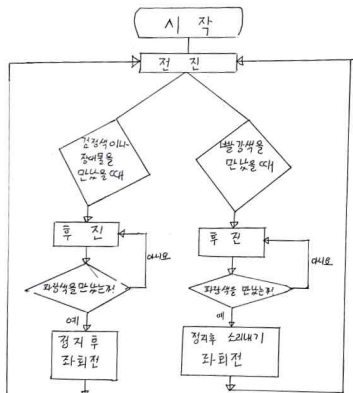
세 번째로, 로봇이 먹이를 먹을 때까지의 동작을 배열에 저장하고 먹이를 먹은 후 유턴하고 거꾸로 동작하는 전략을 세워보았다. 하지만, 이 전략 또한 PID control을 사용하지 않아 차체가 완벽하게 원하는 대로 움직이지 않아 정확히 집으로 돌아가지 못했다.

마지막으로, 로봇이 무작위로 먹이를 먹고 먹이를 먹은 후에도 무작위로 집을 찾아서 돌아오는 프로그램을 구현했다. 이 때, 빨간색 먹이의 좌표를 저장해서 같은 먹이를 먹었을 때 먹이를 소환하지 않도록 설계했다. 이러한 전략은 여러 시행착오 결과 가장 정확하게 동작할 수 있는 방법이었다. 마지막 전략을 세울 때는, ‘생명체’를 설계하는 것에 중심을 두었다. 반응, 에너지 획득, 대사활동, 반응에 대한 함수를 따로 구현해서 생명체처럼 동작하도록 하였다. 이는 생명체의 특징 중 구성, 물질대사, 자극에 대한 반응을 구현한 것이다. 시험 전까지는 항상성 유지, 생식과 발달, 적응과 진화 특징에 관해 더 구현해볼 예정이다.

이번 설계 보고서에서는 첫 번째 전략과 마지막 전략의 프로그램에 대한 설명을 할 예정이다.

II) 작동 시나리오 (순서도를 사용해도 됨)

1) 후진 전략



2) 생명체 구현 전략

- (1) 좌표 표현을 위해 초기화를 한다.
- (2) 직진, 후진, 좌회전, 우회전, 유턴에 관한 함수를 구현한다. 동작할 때 좌표 또한 함께 변화한다.
- (3) 소화, 반응, 에너지 획득, 집으로 돌아오는 각각의 행동에 대한 함수를 선언하고 구현한다.
- (4) 메인 함수에서 에너지를 얻고 집으로 돌아오는 행동을 반복한다.

나. 제품 구조도

I) 제품명 : 살아있는 로봇

II) 주요 부위별 설계 내용 (부위별 도안 포함)

(1) 센싱부

- 2개의 컬러 센서를 사용하여 바퀴 앞쪽에 장착하여 먹이(빨강), 집(파랑)을 감지하도록 함.
- 적외선 센서는 정면을 보도록 설치해 장애물을 피해갈 수 있도록 함.

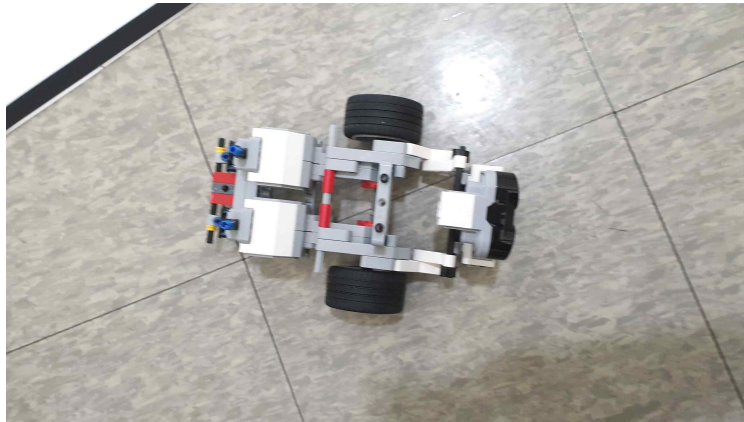
(2) 구동부

- 2개의 모터를 장착하여 전, 후진, 방향 전환이 가능하도록 함.
- 바퀴의 위치는 차체의 좌우에 위치하여 구동력이 제대로 전달될 수 있도록 한다.

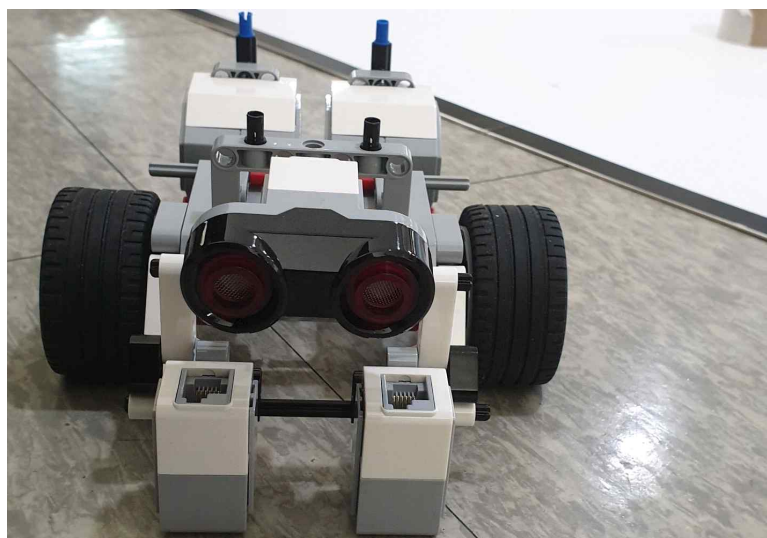
(3) 차체

- EV3 브릭의 위치가 차체의 위에 오도록 한다.

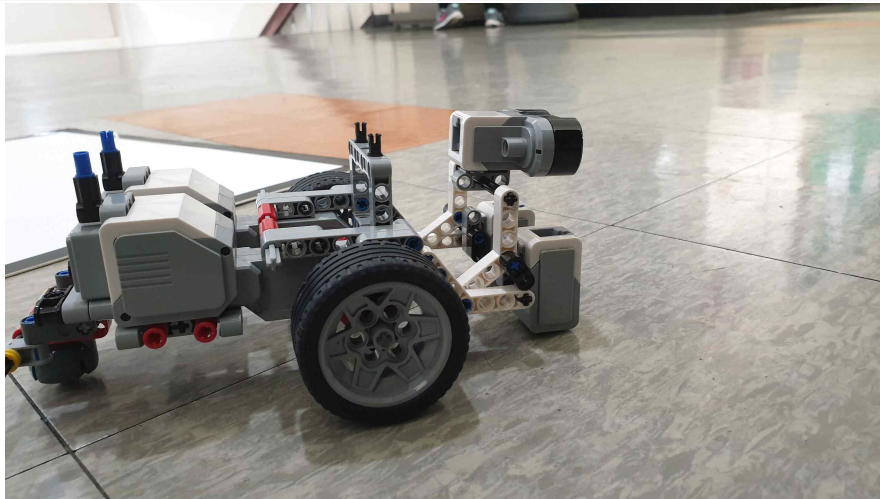
III) 전체 제품 구조도 (주요 부위의 중간 통합 조립형태)



(그림 4. 제품 구조 설계 도안: 평면)



(그림 5. 제품 구조 설계 도안 : 정면)



(그림 6. 제품 구조 설계 도안 : 측면)

다. 프로그램 설계

1) 후진 전략

I) 제품 구조의 주요 부의와 함수와의 1:1 연결성

- (1) 센싱부의 센서 S2와 S3에는 컬러 센서를 달았고 이때 컬러 센서는 컬러 모드로 설정하여 색깔을 인식하도록 하였다. 출력 함수는 SensorValue[S2]와 SensorValue[S3]를 이용하였다.
- (2) 센싱부의 센서 S4에는 초음파 센서를 달았고 물체를 감지하고 거리를 측정하게 했다. 출력 함수는 SensorValue[S4]를 이용하였다.
- (3) 구동부의 포트 B, C에는 라지 모터를 달았다. 함수는 motor 함수를 이용하였다.

II) 작동 시나리오(순서도)와의 함수와의 1:1 연결성

- (1) while문을 이용하여 특별한 조건이 없는 한, 로봇은 계속 전진하도록 설정한다.
- (2) 만약 컬러센서에서 검정색을 인지하거나 초음파센서가 주어진 거리보다 짧다면 첫 번째 if 조건문이 실행이 된다. 위의 조건이 하나라도 충족되었을 경우 (or 조건문을 이용) 파란색(집)이 나올 때까지 후진을 한다. 이후 파란색에 도달했을 경우 잠시 정지하고 좌측으로 회전하고 다시 (1)로 돌아가게 된다.
- (3) 만약 컬러센서에서 먹이를 인지하면 위와 동일하게 집에 도착할 때까지 후진을 한다. 이후 집에 도착 했을 경우 잠시 정지하고 소리를 낸다. 이때 소리는 소리함수 playSound(soundBeepBeep)를 사용하였다. 이후 좌측으로 회전하고 (1)로 돌아가게 된다.

III) 설계된 프로그램 도안

```

1  #pragma config(Sensor, S2,      color,      sensorEV3_Color, modeEV3Color_Color)
2  #pragma config(Sensor, S3,      color,      sensorEV3_Color, modeEV3Color_Color)
3  #pragma config(Sensor, S4,      Ultrasonic, sensorEV3_Ultrasonic)
4  #pragma config(Motor,  motorB,   left,        tmotorEV3_Large)
5  #pragma config(Motor,  motorC,   right,       tmotorEV3_Large)
6  /** Code automatically generated by 'ROBOTC' configuration wizard
7  //1=black 2=blue 3=green 4=yellow 5=red 6=white 7=brown
8
9  task main()
10 {
11     while(true)
12     {
13         motor[motorB] = 30;
14         motor[motorC] = 30;
15
16         if(SensorValue[S2]==1||SensorValue[S3]==1||SensorValue[S4]<50)
17         {
18             while(SensorValue[S2]!= 2)
19             {
20                 motor[motorB] = -30;
21                 motor[motorC] = -30;
22             }
23
24             if(SensorValue[S2]== 2)
25             {
26                 motor[motorB] = 0;
27                 motor[motorC] = 0;
28                 wait1Msec(500);
29                 motor[motorB] = 15;
30                 motor[motorC] = 30;
31
32                 wait1Msec(500);
33             }
34
35             if(SensorValue[S2]== 5)
36             {
37                 while(SensorValue[S2]!=2)
38                 {
39                     motor[motorB] = -30;
40                     motor[motorC] = -30;
41                 }
42                 if(SensorValue[S2]==2)
43                 {
44                     motor[motorB] = 0;
45                     motor[motorC] = 0;
46                     wait1Msec(500);
47                     playSound(soundBeepBeep);
48                     wait1Msec(1000);
49                     motor[motorB] = 15;
50                     motor[motorC] = 30;
51                     wait1Msec(500);
52                 }
53             }
54         }
55     }

```

2) 생명체 구현 전략

I) 제품 구조의 주요 부의와 함수와의 1:1 연결성

- (1) 센싱부의 센서 S2와 S3에는 컬러 센서를 달았고 이때 컬러 센서는 컬러 모드로 설정하여 색깔을 인식하도록 하였다. 출력 함수는 SensorValue[S2]와 SensorValue[S3]를 이용하였다.
- (2) 센싱부의 센서 S4에는 초음파 센서를 달았고 물체를 감지하고 거리를 측정하게 했다. 출력 함수는 SensorValue[S4]를 이용하였다.
- (3) 구동부의 포트 B, C에는 라지 모터를 달았다. 함수는 motor 함수를 이용하였다.

II) 작동 시나리오(순서도)와의 함수와의 1:1 연결성

- (1) while문을 이용하여 특별한 조건이 없는 한, 로봇은 계속 전진하도록 설정한다.
- (2) 만약 컬러센서에서 검정 선을 인지하거나 초음파센서가 주어진 거리보다 짧다면 회전을 한다.
- (3) 먹이를 인지하면 유턴을 한 후 위의 과정을 반복한다. 먹이를 인지했을 때와 집으로 돌아왔을 때 소리함수 playSound(soundBeepBeep)를 사용해 소리를 낸다.

III) 설계된 프로그램 도안

```
1  #pragma config(Sensor, S2, color, sensorEV3_Color, modeEV3Color_Color)
2  #pragma config(Sensor, S3, color, sensorEV3_Color, modeEV3Color_Color)
3  #pragma config(Sensor, S4, Ultrasonic, sensorEV3_Ultrasonic)
4  #pragma config(Motor, motorB, mb, tmotorEV3_Large)
5  #pragma config(Motor, motorC, mc, tmotorEV3_Large)
6
7  //initialize
8  int dest_num = 1;
9  int destination[2]; // (Row, Col)
10 int all_destination[10][2];
11
12 //declare
13 void digestion(); //대사활동, 소화 //소리를 내는 함수
14 void react(); //반응
15 void energy(); //에너지 획득
16 void Home(); //집으로 돌아옴.
17 //void initialize(); // 할당성 유지
18
19 void Forward();
20 void Backward();
21 void LeftTurn();
22 void RightTurn();
23 void U_Turn();
24
25 void digestion()
26 {
27     playSound(soundBeepBeep);
28 }
29
30 void react()
31 {
32     if(SensorValue[S2]==1){ //왼쪽 센서가 검정선을 인식했을 때
33         motor[mb] = 0; motor[mc] = 0; sleep(500);
34         RightTurn(); //오른쪽 turn
35     }
36     else if(SensorValue[S3]==1){ //오른쪽 센서가 검정선을 인식했을 때
37         motor[mb] = 0; motor[mc] = 0; sleep(500);
38         LeftTurn(); //왼쪽 Turn
39     }
40     else{
41         Forward();
42     }
43
44     while(SensorValue[S4]<10) //적외선 센서로 장애물을 인식했을 때
45     {
46         motor[mb] = 0; motor[mc] = 0; sleep(500);
47         RightTurn();
48     }
49 }
50
```



```

51 void energy() //목이름 맞추기
52 {
53     while(true){
54         react();
55         if(SensorValue[S2]==5 || SensorValue[S3]==5)
56         {
57             motor[mb]=0; motor[mc]=0; sleep(500);
58             digestion();
59             U_Turn();
60             break;
61         }
62         /*
63         목이(팔관측)의 위치 저장
64         if(dest_num == 0){
65             dest_num++;
66
67             motor[mb]=0; motor[mc]=0; sleep(500);
68             digestion();
69             U_Turn();
70             break;
71         }
72         else{
73             for(int i=0; i<dest_num; i++){
74                 else if(all_destination[i][0] != all_destination[dest_num][0] && all_destination[i][1] != all_destination[dest_num][1])
75                 {
76                     dest_num++;
77                     all_destination[dest_num][0] = destination[0];
78                     all_destination[dest_num][1] = destination[1];
79
80                     motor[mb]=0; motor[mc]=0; sleep(500);
81                     digestion();
82                     U_Turn();
83                     break;
84                 }
85             }
86         }
87     }
88 }
89 */

```



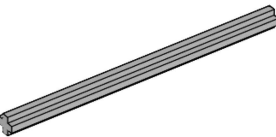
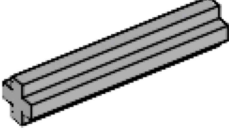

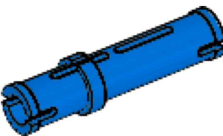
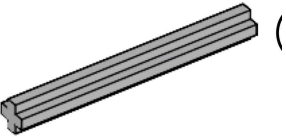
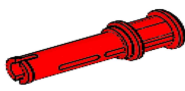

```



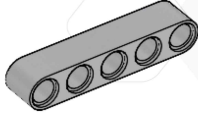
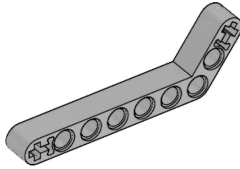
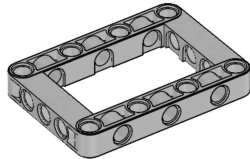
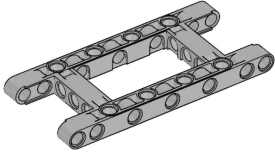
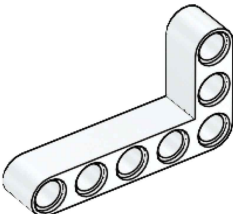
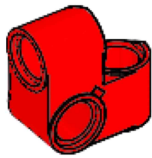

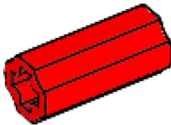
90
91 void Home ()
92 {
93     while(true)
94     {
95         react();
96         if(SensorValue[S2]==2 || SensorValue[S3]==2)
97         {
98             motor[mb]=0; motor[mc]=0; sleep(500);
99             digestion();
100             U_Turn();
101             break;
102         }
103     }
104 }
105
106 task main()
107 {
108     playSound(soundBeepBeep);
109     while(true)
110     {
111         energy();
112         Home();
113     }
114 }
115
116 void LeftTurn() //mode1
117 {
118     motor[mb]=-40; motor[mc]=40; sleep(600);
119     destination[0] += 1;
120 }
121
122 void RightTurn() //mode2
123 {
124     motor[mb]=40; motor[mc]=-40; sleep(600);
125     destination[0] -= 1;
126 }
127
128 void Forward() //mode3
129 {
130     motor[mb]=30; motor[mc]=30; sleep(500);
131     destination[1] += 1;
132 }
133
134 void Backward() //mode4
135 {
136     motor[mb]=-30; motor[mc]=-30; sleep(500);
137     destination[1] -= 1;
138 }
139
140 void U_Turn()
141 {
142     //LeftUTurn;
143     motor[mb]=-40; motor[mc]=40; sleep(1200); //180도 회전

```

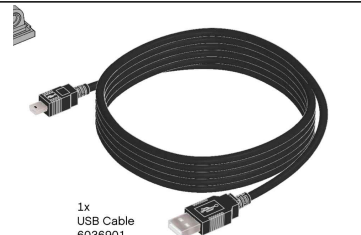
라. 부품표

설명 : 개발 제품이 어떤 부품들로 구성되는지에 대한 데이터(부품번호, 부품명, 수량 등)를 나타낸다. 부품번호와 부품명은 Lego-Mindstorms-NXT-Education-Kit.pdf 파일을 참고.

부품번호	부품명	수량	이미지
4142822	Beam, 3-module, black	1	
4121715	Connector peg with friction,2-module,bl ack	20	
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 crosshole, 2-module, black	2	
4509912	Angular beam, 4*4-module, 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-module, grey	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 extender, 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	2	
6008924	Ultrasonic Sensor	1	
6009430	Large Motor	2	
6024583	Cable, 35cm/14in	4	<div>35 cm / 14 in.</div> 
6024585	Cable, 50cm/20in	1	<div>50 cm / 20 in.</div> 

6036901	USB Cable	1	
---------	-----------	---	--

3. 토의

본 프로젝트에서 로봇에 코딩 프로그램을 주입시켜 생명체의 특징을 가지는 인공생명체 로봇을 만들어 보았다.

로봇 제작을 할 때, 우리 조는 센싱부의 컬러센서와 적외선 센서의 위치에 대한 토의를 해왔다. 그 결과 컬러센서 두 개는 아래로 두 개 있고, 적외선 센서는 정면을 보는 지금의 로봇을 만들었다. 하지만 프로그램을 작성하면서 수정이 될 가능성도 있다. 예를 들면, 적외선 센서의 고정 방식을 수정하거나 컬러센서의 위치나 개수를 수정할 가능성이 있다.

프로그램을 작성할 때, 최적의 프로그램을 작성하기 위해 노력했으나 부족한 점이 몇몇 있었다. 이는 다음과 같다.

첫 번째로, 로봇이 검정선을 벗어나 결국 경기장 바깥으로 나가는 문제가 있었다. 여러 시행착오 끝에 회전 각도를 키워서 경기장 경계에 걸렸을 때 경기장 안으로 완전히 들어오도록 프로그램을 작성했으나, 아예 선을 인식을 못하거나 두 센서가 동시에 인식했을 때 차체가 도는 문제를 해결해야 한다. 또한, 경기장을 완전히 벗어났을 때 후진하는 코드를 삽입하는 것도 좋을 것이다. 또한, 컬러센서가 서로 너무 가까이 붙어있어서 인식을 못할 수도 있으므로 컬러센서 사이의 거리를 띄우는 방법도 생각해봐야 한다.

두 번째로, 장애물을 만났을 때의 반응이다. 우선은 장애물을 만났을 때 무조건 회전을 하도록 구현했으나 장애물을 완벽하게 지나가는 코드를 구현하는 것도 좋을 것이다. 하지만 이는 장애물의 크기를 미리 알 수 없으므로 구현하기 힘들 가능성이 있다.

세 번째로, 먹었던 먹이는 지나치는 기능을 완벽하게 추가해야 한다. 현재의 코드는 동작할 때 마다 회전 각도가 조금씩 달라지는 문제로 좌표가 완벽하지 않아서 완벽한 좌표를 찍지 못했다. 다른 방법을 생각해볼 필요가 있다.

네 번째로, 현재는 기본 주행모드가 직진인데, 기본적으로 무작위로 동작하도록 코드를 수정해야 할 것이다. 무작위로 움직이는 것이 생명체의 특성일뿐더러, 무작위로 동작하도록 해야 더욱 많은 경로를 찾아내고 먹이를 잘 찾아낼 수 있을 것이다.

마지막으로, 생명체의 특성을 더 구현해야 한다. 지금까지는 생명체의 특징 중 구성, 물질대사, 자극에 대한 반응을 구현했다. 앞으로 항상성 유지, 생식과 발달, 적응과 진화에 대해 더 구현해볼 것이다. 항상성 유지 기능은 모든 함수를 초기화하는 함수를 추가해 볼 것이다. 생식과 발달 적응과 진화 특징은 사전 자료조사를 했지만 시간 부족 문제로 구현하지 못했던 인공지능 Qlearning을 적용해볼 것이다.