

## 창의설계입문 프로젝트 설계서

프로젝트 제목	인공생명체 로봇 만들기				작성년월일	2020/12/25
프로젝트 팀 이름	학번	201921120	-	-	-	
일등조	성명	우다현	000	000	000	
	역할	팀장 및 전략 및 프로그래밍 및 문서작성	구조물 설계 및 도안 정리	전략 및 작동 시나리오 구성	전략	

### 1. 요약

본 프로젝트의 목적은 인공지능 로봇을 설계하는 것이다. 인공지능 기술은 인간의 뇌를 복제하는 것을 목표로 한다. 인공지능 기술은 뉴런 모델링을 통해 인공 신경망을 설계하고 학습시킨다. 인공 신경망을 여러 개가 상호작용하여 심층 신경망을 설계할 수 있고, 이러한 심층 신경망을 학습 시키는 기술이 딥러닝(Deep Learning)이다. 반도체 기술의 발전으로 인한 고성능 하드웨어의 등장으로 기술발전이 가속화되고 있고, 인간의 두뇌를 모사한 심층신경망과 딥러닝 기술을 통해 오늘날의 인공지능 기술의 발전을 이끌고 있다. 그 예로는 예쁜 꼬마선충 로봇 등이 있다. 또 다른 예시로, Bio-inspired Technology 중 달팽이관 달팽기관 구조를 모사하여 Radio-frequency 칩 구조를 구현해 기존 구조보다 100 배 이상 적은 전력을 소모 하는 기술이 있으며 변의, 교차, 선택 등, 자연계의 진화이론을 차용하여 최적화된 값을 찾아내는 방법론인 유전 알고리즘이 있다.

우리는 본 프로젝트에서 인공생명체를 설계할 것이다. 생명체의 특징에는 구성(구조적 기능적 기본 단위인 세포로 구성), 물질 대사(생명을 유지하기 위한 에너지의 흡수와 방출), 자극에 대한 반응(외부 환경요인에 의한 행동,운동, 혹은 대사과정의 변화), 항상성 유지 (외부 환경 변화에 내부 환경을 일정하게 유지), 생식과 발달 (생식에 의한 탄생, 발달 단계를 거치는 성장과 죽음), 적응과 진화(외부 환경요인에 유리한 특성을 획득)가 있다. 본 프로젝트에서는 로봇에 이러한 생명체의 특징을 구현해내는 것에 목표를 둘 것이다.

프로젝트의 필수 수행 내용은 다음과 같다. 로봇은 경기장 정보에 대해서 모르며 무작위로 움직인다. 경기장 가장자리를 인식하여 밖으로 나가면 안되며(반응), 이탈시마다 감점한다. 로봇을 먹이(빨간색)를 찾으려고 하며 (에너지 획득) 먹이를 먹은 뒤 집(파란색)으로 돌아가야 한다(대사 활동). 먹이를 만나거나 집으로 돌아오면 소리를 낸다. 적(사각박스)을 만나면 피해야한다. (반응). 제한시간 (3분) 이내에 먹이를 최대한 많이 먹는(먹이와 집을 왕복하는) 로봇을 구현하는 것(적응)이 목표이다.

제작실습의 평가 기준은 다음과 같다. 필수 수행내용 구현 여부, 아이디어의 참신성, 효율적으로 필수 기능이 구현되었는지, 스스로 학습하는 기능이 구현되었는지, 필수 수행내용 이외의 생명체의 특징을 구현하였는지, 그리고 작업시 효율성과 협동성, 규정 준수 및 공정 경쟁을 하였는지가 본 프로젝트의 평가 기준이 될 것이다.

우리 조는 평가기준에 맞추어 로봇을 설계하는 것을 목적으로 하였다. 어떻게 로봇을 설계할지 토의를 하며 전략을 세웠고 이를 바탕으로 로봇과 프로그램을 설계하였다.

## 2. 제품 구조도

가. 제품명 : 살아있는 로봇

나. 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.

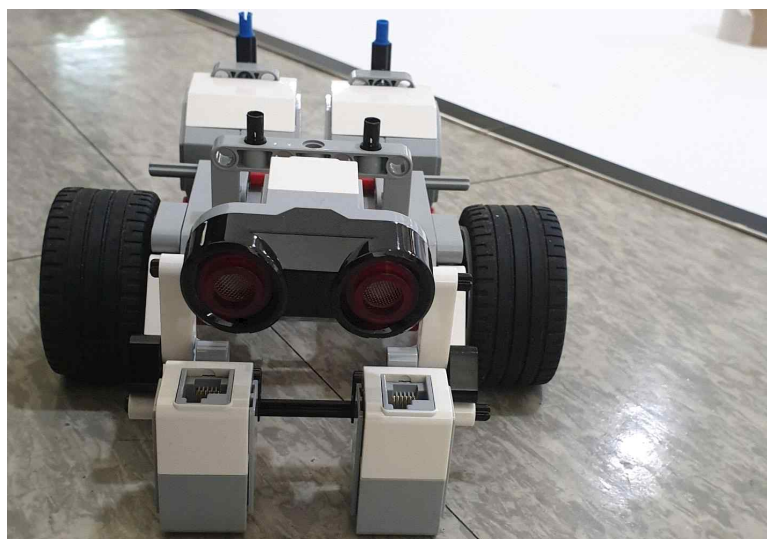
### (3) 차체

- EV3 브릭의 위치가 차체의 위에 오도록 한다.

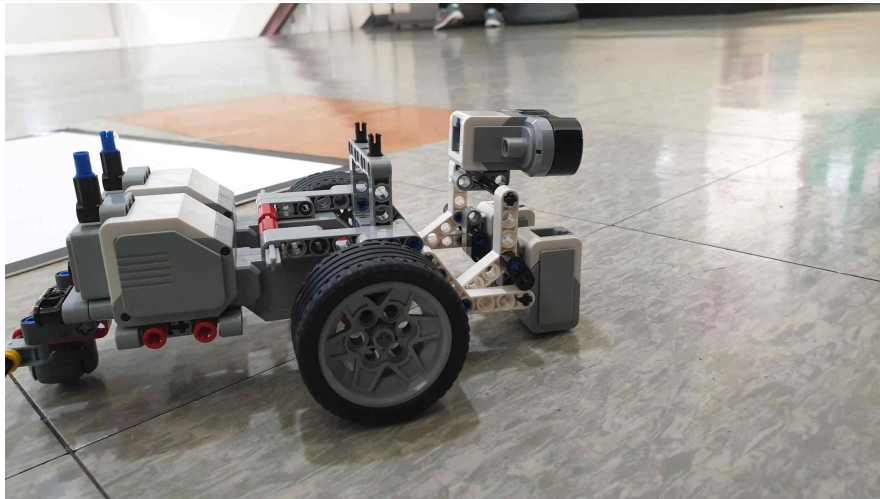
II) 전체 제품 구조도 (주요 부위의 중간 통합 조립형태)



( 그림 1. 제품 구조 설계 도안: 평면 )



(그림 2. 제품 구조 설계 도안 : 정면)



(그림 3. 제품 구조 설계 도안 : 측면)

### 3. 프로그램 설계

#### 가. 전략

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 소화하지 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, 반응, 항상성 유지, 진화에 대한 함수를 따로 구현해서 생명체처럼 동작하도록 하였다. 이는 생명체의 특징 중 구성, 물질대사, 자극에 대한 반응을 구현한 것이다.

#### 나. 최종 작동 시나리오 - 생명체 구현 전략

##### 1) 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.

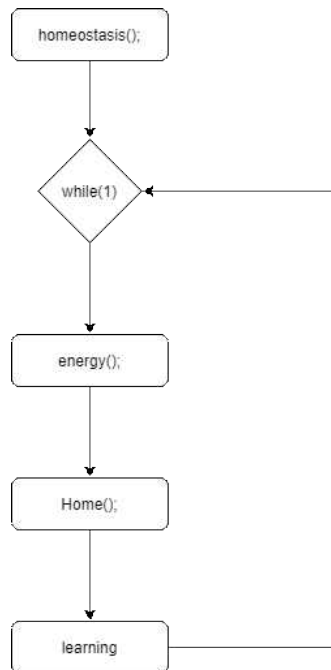
(2) 센싱부의 센서 S4에는 초음파 센서를 달았고 물체를 감지하고 거리를 측정하게 했다. 출력 함수는 SensorValue[S4]를 이용하였다.

(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) 순서도



- (1) 좌표 표현을 위해 초기화를 한다.
- (2) 직진, 후진, 좌회전, 우회전, 유턴, 무작위 동작에 관한 함수를 구현한다. 동작할 때 좌표 또한 함께 변화한다.
- (3) 소화, 반응, 에너지 획득, 집으로 돌아오는 각각에 대한 함수를 선언하고 구현한다.
- (4) learning과 관련된 함수를 선언하고 구현한다.
- (5) 메인 함수에서 에너지를 얻고 집으로 돌아오는 행동을 반복한다. 자체적으로 learning한다.

## 나. 설계된 프로그램 사진

```
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 row_dest; //현재의 행위치
9  int col_dest; //현재의 열위치
10
11 int dest_num = 0;
12 int row_destination[10]; //최종 행위치, 먹이는 크게 잡아 10개라고 가정
13 int col_destination[10]; //최종 열위치
14
15 int r;
16
17 int total_reward;
18 int honor_destination[2];
19
```

```

20 //declare
21 void digestion(); //대사활동, 소화 //소리를 내는 함수
22 void homeostasis(); // 항상성 유지
23 void react(); //반응
24 void energy(); //에너지 획득
25 void Home(); //집으로 돌아옴.
26
27 int getState();
28 int getReward(int state);
29
30 void Forward();
31 void Backward();
32 void LeftTurn();
33 void RightTurn();
34 void U_Turn();
35 void random_move();
36
37 //function
38 void digestion()
39 {
40     playSound(soundBeepBeep);
41 }
42
43 void homeostasis(){
44     row_dest = 0;
45     col_dest = 0;
46 }
47
48 void react()
49 {
50     if(SensorValue[S2]==1){ //왼쪽 센서가 장애선을 인식했을 때
51         motor[mb] = 0; motor[mc] = 0; sleep(500);
52         Backward();
53         RightTurn(); //오른쪽 trun
54     }
55     else if(SensorValue[S3]==1){ //오른쪽 센서가 장애선을 인식했을 때
56         motor[mb] = 0; motor[mc] = 0; sleep(500);
57         Backward();
58         LeftTurn(); //왼쪽 Turn
59     }
60     else{
61         random_move(); //Forward();
62     }
63
64     while(SensorValue[S4]<15) //적외선 센서로 장애물을 인식`했을 때
65     {
66         motor[mb] = 0; motor[mc] = 0; sleep(500);
67         Backward();
68         LeftTurn();
69     }
70 }

```



```

72 void energy() //먹이를 찾음
73 {
74     while(true){
75         react();
76         if(SensorValue[S2]==5 || SensorValue[S3]==5) //빨간색
77         {
78             Forward(); //원 안으로 들어감
79             if(dest_num != 0){
80                 for(int i=0; i<dest_num; i++){
81                     if(row_destination[i]-3<= row_dest && row_dest <= row_destination[i]+3
82                        && col_destination[i]-3 <= col_dest && col_dest <= col_destination[i]+3
83                        break; //현재 위치가 저장된 위치들과 비슷 (먹이의 크기 고려)한지 판단 , 비슷하면 이미 먹은 먹이임
84                 }
85             }
86         }
87         row_destination[dest_num] = row_dest; //현재 위치 저장
88         col_destination[dest_num] = col_dest;
89         dest_num++;
90         homeostasis();
91
92         total_reward += getReward(getState());
93
94         motor[mb]=0; motor[mc]=0; sleep(500);
95         digestion();
96         U_Turn();
97         break;
98     }
99 }
100 }

102 void Home()
103 {
104     while(true){
105         react();
106         homeostasis(); //집으로 돌아올 때는 위치 저장할 필요 없음
107         if(SensorValue[S2]==2 || SensorValue[S3]==2) //파란색
108         {
109             Forward(); //원 안으로 들어가기
110
111             total_reward += getReward(getState());
112
113             motor[mb]=0; motor[mc]=0; sleep(500);
114             digestion();
115             U_Turn();
116             break;
117         }
118     }
119 }

121 int getState()
122 {
123     if(SensorValue[S2]==5 && SensorValue[S3]==5)
124     {
125         return 4;
126     }
127     else if(SensorValue[S2]==3 && SensorValue[S3]==3)
128     {
129         return 3;
130     }
131     else if(SensorValue[S2]==5 || SensorValue[S3]==5)
132     {
133         return 2;
134     }
135     else if(SensorValue[S2]==3 || SensorValue[S3]==3)
136     {
137         return 1;
138     }
139     else
140     {
141         return 0;
142     }
143 }
144 }

```

```

147     int getReward(int state)
148     {
149         int reward;
150
151         switch(state)
152         {
153             case 0:
154                 reward = 100;
155                 break;
156             case 1:
157                 reward = -25;
158                 break;
159             case 2:
160                 reward = -25;
161                 break;
162             case 3:
163                 reward = -50;
164                 break;
165             case 4:
166                 reward = -50;
167                 break;
168             default:
169                 return 0;
170                 break;
171         }
172
173         return reward;
174     }
175
176
177     task main()
178     {
179         homeostasis();
180         while(true)
181         {
182             total_reward = 0;
183             total_reward += getReward(getState()); //+100
184             energy();
185             Home();
186             if(total_reward == 0){ //가장 잘 배움, perfect //파턴, 목적지 기억해두기
187                 playSound(soundBeepBeep);
188                 honor_destination[0] = row_destination[dest_num-1];
189                 honor_destination[1] = row_destination[dest_num-1];
190             }
191         }
192     }
193 }

```

```

195 void LeftTurn() //mode1
196 {
197     motor[mb]=-40; motor[mc]=40; sleep(600);
198     row_dest += 1;
199 }
200
201 void RightTurn() //mode2
202 {
203     motor[mb]=40; motor[mc]=-40; sleep(600);
204     row_dest -= 1;
205 }
206
207 void Forward() //mode3
208 {
209     motor[mb]=30; motor[mc]=30; sleep(150);
210     col_dest += 1;
211 }
212
213 void Backward() //mode4
214 {
215     motor[mb]=-30; motor[mc]=-30; sleep(600);
216     col_dest -= 1;
217 }
218
219 void U_Turn() //LeftUTurn;
220 {
221     motor[mb]=-40; motor[mc]=40; sleep(1300); //180도 회전
222 }
223
224 void random_move()
225 {
226     r = random(9); //0~8 (n-1)의 무작위 수
227
228     if(r==0) LeftTurn();
229     if(r==1) RightTurn();
230     if(r==2 || r==3 || r==4 || r==5 || r==6 || r==7 || r==8) Forward();
231 }

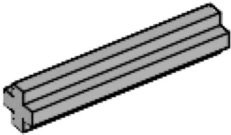
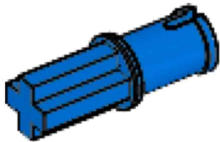
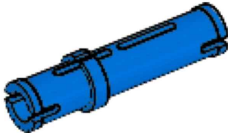
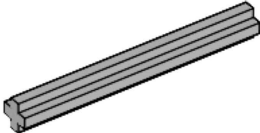
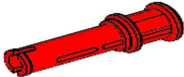


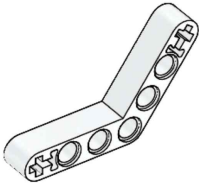
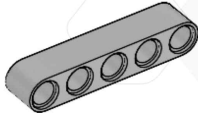
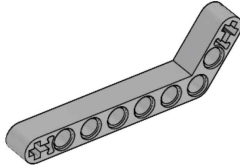
```

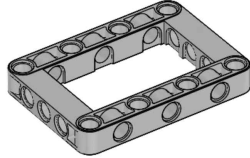
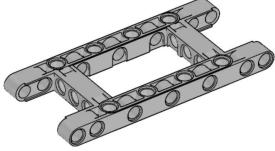
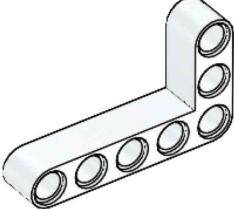
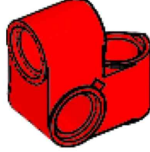

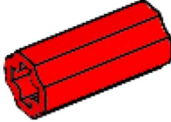
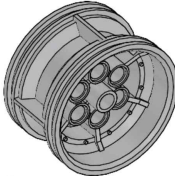
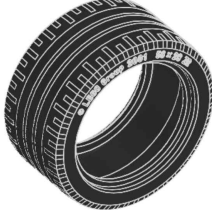
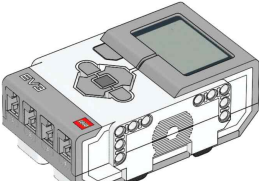
#### 4. 부품표

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 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-m odule, 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 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	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	 <div>1x USB Cable 6036901</div>

## 6. 제작 과정 기록

### 가. 제작 방법

본 설계에는 bric 본체, 컬러 센서 2개, 적외선 센서 1개, Large Motor 2개, 연결하는 봉 여러 개를 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 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 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.

#### 나. 제작 과정 사진 및 기록

- (1) 제작 초기에는 bric과 로봇, 로봇과 적외선 센서가 고정이 잘 되지 않았다.
- (2) 연결 부품을 사용하여 bric과 로봇을 밀착시켰고 적외선 센서를 잘 고정하였다.



( 그림 4. 제작 과정 사진 )

## 6. 결과 토의

#### 가. 설계서와 실제 제작품과의 차이점

설계서와 실제 제작품과의 차이점은 거의 없었다. 다만, 하드웨어 제작 과정에서 자외선 센서와 bric을 더욱 밀착해서 조립하였다.

#### 나. 제작 과정의 애로 사항

우선, 프로그램을 작성해보고 검증하는 과정에서, 제작 비용의 한계로 모든 경우의 수를 고해볼 수 없었다는 점이 힘들었다. 우리 조의 경우 맵을 한 개만 만들었고, 한정된 맵 안에서 여러 경우의 수를 고려해서 프로그램 수정을 계속해서 반복했다. 또한 전략을 세우는 것도 어려웠다. 마지막으로, 시험 기간에 프로젝트를 진행했기 때문에 팀원들의 참여도가 저조해 조장이 많은

것을 도맡아 해야 했다는 점이 힘들었다.

#### 다. 평가 기준 충족

##### (1) 필수 수행 내용 구현 여부

random\_move()함수를 이용해 로봇이 무작위로 움직이는 것을 구현했고(경연 시에는 무작위 움직임이 먹이와 집을 찾는데 시간이 너무 오래 걸리는 점을 고려해서 기본적으로 forward()함수를 이용했다), react()함수에서 양쪽 컬러센서 중 하나라도 검정색 선, 즉 경기장 가장자리를 인식하여 밖으로 나가지 않도록 하도록 구현했다(반응). 또한 양쪽 컬러센서가 먹이(빨간색)을 인지해서 먹이를 찾을 때(에너지 획득)까지 움직이도록 하였고, 집으로 돌아오는 함수 Home()을 통해 유턴을 해서 집으로 돌아오도록 구현하였다. 또한 digestion()함수를 이용해 먹이를 만나거나 집으로 돌아오면 소리를 내도록 했다(대사활동). 마지막으로 react()함수에서 적외선 센서를 이용해 적(사각박스)를 만나면 피하도록 하였다. 그리고 로봇이 맵을 돌아다니면서 적응하도록 하였다.

##### (2) 아이디어의 참신성

우선 각각의 생명체의 특징을 객체 지향적인 함수로 구현했다는 점이 참신하다. 실제로 생명체는 각 기관들이 기본 단위로 나뉘어져 분업하는 특징을 구현한 것이다. 또한, 현재의 위치를 좌표로 저장해서 먹이를 먹었을 때의 좌표를 저장하는 기능을 구현한 것이 참신하다. 마지막으로, 얼마나 정확하게 원안에 들어가는지 학습하는 기능을 구현하는 방법이 참신하다.

##### (3) 효율적으로 필수 기능 구현

각 생명체의 특징에 해당하는 함수를 작성해서 깔끔하고 효율적으로 필수 기능을 수행하도록 하였다.

##### (4) 스스로 학습하는 기능

스스로 학습하는 기능을 구현하기 위해서 보상을 주는 방식을 생각해보았다. 본 프로젝트에서는 얼마나 정확하게 먹이, 그리고 집에 도착했는지에 따라 보상을 다르게 줌으로써 정확하게 길을 찾는 방법을 학습시키고자 하였다. 또한, 이 외에도 스스로 현재 위치를 좌표로 저장하면서 가는 길을 찾는 방법을 학습했다고도 할 수 있다.

##### (5) 필수 수행 내용 이외의 생명체의 특징 구현

생명체의 특징에는 구성(구조적 기능적 기본 단위인 세포로 구성), 물질 대사(생명을 유지하기 위한 에너지의 흡수와 방출), 자극에 대한 반응(외부 환경요인에 의한 행동, 운동, 혹은 대사과정의 변화), 항상성 유지 (외부 환경 변화에 내부 환경을 일정하게 유지), 생식과 발달 (생식에 의한 탄생, 발달 단계를 거치는 성장과 죽음), 적응과 진화(외부 환경요인에 유리한 특성을 획득)가 있다.

위 프로그램에서는 여러 함수를 각각 구현함으로써 구성을 구현했고, digestion()함수를 통해 물질대사를 구현했고, react()함수를 통해 경기장의 경계를 벗어나지 않고 장애물을 피하는 자극에 대한 반응을 구현했고, homeostasis()함수를 통해 현재 좌표를 초기화 시켜 내부 환경을 일정하게 유지하는 항상성 유지를 구현했고, energy()함수에서 좌표를 생성하고 갱신하고 초기화 시키면서 생식과 발달을 구현했고, getState(),getReward(),main()함수에서 스스로 학습하는 적응과 진화를 구현했다.

라. 경연에서 미흡했던 점과 그 개선 방안

우리 조는 경연에서 준비한 만큼 좋은 성과를 내지 못했다. 그 이유는 다음과 같다.

#### (1) 같은 먹이 중복 인식

우리 조의 로봇은 먹이에 대한 중복 인식을 할 줄 알았지만, 가끔씩 중복 인식을 하지 못할 때도 있었다. 그래서 경연이 끝난 후 코드를 보았고, 그 결과 문제점을 찾아냈다.

아래는 기존의 코드의 일부를 가져온 것이다.

```
for(int i=0; i<dest_num; i++){  
    if (row_dest==row_destination[i] && col_dest==col_destination[i]) break;  
}
```

위 코드에서는 현재의 위치와 지금까지의 먹이의 위치가 동일하면 함수를 나가는 코드이다. 하지만 위 코드는 원의 크기를 고려하지 못했고, 어느 정도의 오차도 포함하지 못한다는 문제가 있었다.

그래서 수정한 코드는 아래와 같았다.

```
for(int i=0; i<dest_num; i++){  
    if(row_dest-3<=row_destination[i] && row_destination[i]<=row_dest+5 &&  
col_dest-5<=col_destination[i] && col_destination[i]<=col_dest+5) break;  
}
```

이렇게 코드를 수정하면, 어느 정도의 오차를 수용할 수 있다. 하지만 현재 위치인 row\_dest를 위주로 코드를 작성해야 하는데 이전 위치를 기준으로 코드를 작성했다는 것을 점을 깨달았다.

수정한 최종 코드는 아래와 같다.

```
for(int i=0; i<dest_num; i++){  
    if(row_destination[i]-3 <= row_dest && row_dest <= row_destination[i]+3 &&  
col_destination[i]-3 <= col_dest && col_dest <= col_destination[i]+3) break;  
}
```

이렇게 수정한 코드로 경연을 다시 했을 때, 중복 먹이를 인식할 수 있으리라고 확신한다.

#### (2) 장애물 인식 실패

이 경우는 하드웨어의 문제가 컸다. 대각선으로 움직일 때 적외선 센서의 바로 앞에 장애물이 위치하지 않아 인식하지 못하는 경우가 많았고, 그 결과 부딪히는 경우가 많았다. 이 문제를 해결하기 위해서는 적외선 센서를 본체에 완전히 밀착해서 제작하고, 본체의 크기를 작게 해서 의도치 않은 부딪힘을 방지하고, 작은 바퀴를 사용하는 방법이 있을 것이다.

#### 마. 경연에서의 중요한 사항에 대한 해결방안 (다른 조에서의 문제점 해결방안)

경연을 할 때, 다른 조에서의 문제점은 선을 벗어나고 장애물에 부딪히는 등 제어가 잘되지 않는다는 점이였다. 이에 대한 해결방안은 주행 시간을 줄이고 제어 시간을 촘촘히 하는 데에 있다.

우리 조는 경연 할 때 제어가 잘 된다는 평가를 받은 바가 있다. 우리 조는 여러 시행착오를 통해 제어를 잘 하는 방법을 알아냈는데, 짧은 시간마다 선을 벗어났는지 장애물이 앞에 있는지 확인하는 것이였다. 그래서 기본적으로 주행하는 Foward() 함수의 직진 시간을 매우 줄였다. 다른 조도 이런 식으로 프로그램을 수정한다면 더 좋은 결과를 얻을 수 있으리라고 생각한다.



#### 사. 소감

본 프로젝트를 진행하면서 로봇을 제작하고 robotc 프로그램을 작성하면서 로봇에 대한 이해도를 높일 수 있었다. 앞으로 완성된 로봇을 볼 때 어떻게 로봇이 조립되어 있는지, 프로그램은 어떻게 작성되어 있는지, 부품과 프로그램이 어떻게 유기적으로 연결되어 있는지 생각해볼 수 있을 것이다. 또한, 앞으로의 진로를 생각할 때 로봇 엔지니어를 생각해 볼 수 있는 계기를 얻었다.