제16회 임베디드SW경진대회

개발일지 [주니어 임베디드SW 챌린저]

산업통상자원부

작품명 : 전통시장을 위한 스마트 셔틀로봇

팀 명 : S-ROBO1

팀번호	17038
대표소속	송도중학교
팀원이름	윤태현, 박재영, 강민수
지도교사(멘토) 이름	노영택
시연 동영상(URL)	https://youtu.be/QYY6t-ExLpY

임베디드 소프트웨어 경진대회
The World Embedded Software Contest

임베디드 소프트웨어 경진대회
The World Embedded Software Contest

개발 일정

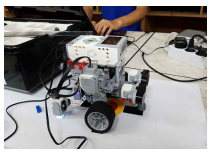
번호	연구기간	연구내용
1	06월 04일 ~ 06월 19일	로봇 구조 변경, 1단계 주행 구현
2	06월 20일 ~ 07월 11일	로봇 구조 재변경, 1단계 주행 재구현
3	07월 12일 ~ 08월 02일	2단계 최단거리 알고리즘 설계 및 구현
4	08월 03일 ~ 08월 25일	3단계 주행 및 승객 하차 구현
5	08월 27일 ~ 09월 19일	경기용 매트에 맞게 주행 재구현, 알고리즘 변경
6	09월 20일 ~ 10월 01일	개발일지 작성

전통 시장을 위한 스마트 셔틀로봇 제작	
프로젝트 기간 및 목표	
기간	06월 04일 ~ 10월 01일
목표	전통 시장을 위한 스마트 셔틀로봇 제작

프로젝트 활동내용

1. 로봇 제작

1) Version 0.1



i. 좌우 컬러센서로 상점 양 옆 라인을 잡아 보정하는 방식이다.
ii. 승객 하차 장치는 아직 개발이 되지 않은 상태
iii. 컬러센서 배치 구조상 보정속도가 느리며 정확성도 보장되지 않는다.
iv. 무게중심이 너무 균등해서 멈추거나 갑자기 움직였을 때 흔들림이 많아서 센싱을 하는데 문제가 생김

2) Version 1.0

i. 앞 뒤 컬러센서로 검은 색 라인의 좌우를 잡아 주행 중 보정을 하고 가운데 컬러 센서로 상점 정보를 읽는 방식이다.
ii. 미디엄 모터와 크랭크 구조를 이용해서 승객을 밀어냄(승객이 규정에 정해지지 않았을 때)
ii. 센서의 위치상 로봇의 모든 자세를 보정 할 수는 없었다.

3) Version 2.0

i. 앞쪽에 컬러센서를 모아놓은 구조이다. 가운데 컬러센서로 상점 정보를 읽고 뒤쪽 좌우 컬러센서로 라인을 잡아 주행 중 보정을 한다.

ii. 2단계 정보마당 진입을 위해 초음파 센서를 장착해 상점 주첨기를 찾는 용도로 사용
iii. 미디엄 모터를 사용해 승객을 한 명씩 밀어낸다. 주행 중 승객이 로봇에서 떨어질 수 있으므로 레고 고무줄을 활용해서 평소에는 닫혀있다 임의로 힘을 주었을 때 열릴 수 있는 가이드를 만들어 승객이 떨어지는 것을 방지했다.
iv. 가운데 컬러센서와 좌우 컬러센서 간의 앞뒤 간격이 넓어 보정시간이 너무 짧아 안정적인 보정이 어려웠다.

4) **Version 2.1**

i. 가운데 컬러센서와 좌우 컬러센서 간의 앞뒤 간격을 좁혀서 보정시간을 늘려 즉각적인 보정이 가능하도록 했다.

2. 주행 구현

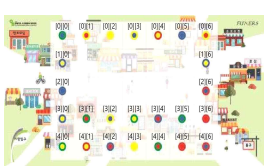
1) 1단계 상점 스캔을 위한 주행 & 3단계 승객 하차를 위한 주행

1,3단계에서 사용하는 주행 함수은 기본적인 라인트레이싱을 원리를 사용한다. 주행 함수은 아래와 같다. 주행경로와 모드를 인수로 받게 되는데 모드로 1단계와 3단계를 구분하게 된다.

주행 함수

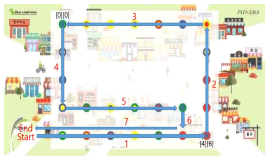
i. 회전을 해야 하는지 확인 후 회전을 진행
ii. 가운데 컬러센서에 상점 색(파노빨)이 인식될 때까지 좌우 컬러센서를 이용해 왼쪽 컬러 센서가 라인을 잡으면 왼쪽으로 오른쪽 컬러센서가 라인은 잡으면 오른쪽으로 약간씩 보정해주면서 전진하여 상점에 진입한다.
iii. 상점에 진입하게 되면 로봇을 임의로 약간 앞으로 빼내어 상점 추천 정보를 저장한다 (3단계에서는 상점 추천 정보를 저장하지 않고 승객 하차 위치를 검사하여 승객을 하차 시킨다.)
iv. 저장 후 상점을 나갈 때도 임의로 약간 앞으로 빼주어 상점에서 나간다.

위 과정을 여러번 실행하여 경로를 따라 주행을 하게된다.



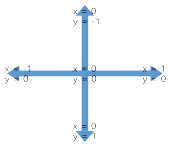
여기서 경로란 시장 맵의 상점들에 5*7 인덱스를 부여해서 이동해야하는 순서대로 그 상점들의 인덱스를 저장해놓은 배열을 말한다.

1단계 상점 추천 정보 스캔



위 그림은 상점 추천 정보를 스캔하는 경로이다.
주행 함수를 보면 다음 상점으로 직진하기 전에 회전을 해야 하는지 확인을 하고 회전을 진행하게 되는데 회전을 해야 하는지 판단하는 기준은 현제 상점의 인덱스와 다음 상점의 인덱스를 비교해서 다음 상점의 방향을 알아내고 그것을 로봇의 방향과 비교하여 회전을 해야 하는지, 회전 방향을 결정하게 된다.

여기서 방향이란 현제 상점의 x, y인덱스와 다음 상점의 x, y인덱스의 크고 작음을 비교해서 아래 그림과 같이 변수에 나타낸 것을 말한다.



3단계 승객 하차

3단계에서는 2단계에서 만들어진 최단 경로를 주행하면서 2단계에서 만들어진 경유지 배열을 통해 승객을 하차시키게 된다.

주행 함수의 iii을 보면 3단계에서는 추천 정보를 저장하지 않고 현제 위치에 승객을 하차하게 되어있다. 이때 1단계와 3단계를 인수로 받은 모드를 통해 구별한다.

2) 2단계 정보마당 진입

2단계에서는 추천정보 추첨기에 정확하게 찍고 돌아와야하기 때문에 좀 더 정밀한 회전이 필요하다. 그래서 항상 같은 위치에서 추첨기 방향으로 회전할 수 있게 위치를 고정하기 위해서 아래와 같은 주행 방법을 사용했다.

i. 임의로 라인을 벗어나게 회전
ii. 반대로 컬러센서가 라인을 잡을 때까지 반대로 회전

이 방법을 사용하면 항상 라인의 같은 쪽을 잡을 수 있게 되어 로봇의 자세가 일정해진다. 그래서 상점 입구에서 추첨기와 가장 가까운 상점까지 이동하는 동안 자세가 일정하게 유지되어 추첨기 방향으로 회전하는 위치가 일정해졌다.

추첨기 방향으로 회전을 할때는 초음파센서로 거리를 측정하여 값이 일정 범위 안으로 들어올 때까지 회전하도록 했다. 여기서 약간 덜 돌기 때문에 임의로 돌려주는 데 초음파

센서로 회전한 후 잠깐 멈추지 않고 바로 돌려주게 되면 모터의 회전 각도가 일정하지 않아서 잠깐 모터를 멈추게 한 뒤 돌려주었다.

3. 2단계 최단거리 알고리즘 구현

1) 경유지점 검색

1단계에서 구한 시장 상점 추천 정보에서 2단계 추첨기에서 읽은 추천 정보를 검색하여 경유지 배열에 저장하게 되는데 무조건 처음과 끝은 경유해야하므로 경유지 배열의 처음과 끝에는 무조건 [0][0]과 [4][6]을 저장하고 그 사이에 검색하여 얻은 4개의 상점의 인덱스를 저장한다.

2) 최단 경유 순서 조합 검색

경유지를 경유하는 순서에 따라 이동해야 하는 거리가 달라진다. 그래서 배열에 저장된 경유지들을 조합 가능한 경우의 수들로 배치해서 각각의 거리를 계산한 뒤 그중 가장 짧은 조합을 찾게 된다. 각각의 조합들의 거리를 구하는 공식은 다음과 같다.

i. **가로세로 또는 세로가로 이동 중 빈공간이 없는 경우**
| (현제 경유지의 x) - (다음 경유지의 x) | + | (현제 경유지의 y) - (다음 경유지의 y) |
ii. **가로세로 또는 세로가로 이동 중 빈공간이 없는 경우(경유지가 좌우로 있는 경우)**
| 1 - (현제 경유지의 y) | + 6 + | 1 - (다음 경유지의 y) |
or
| 4 - (현제 경유지의 y) | + 6 + | 4 - (다음 경유지의 y) |
iii. **가로세로 또는 세로가로 이동 중 빈공간이 없는 경우(경유지가 위아래로 있는 경우)**
| 0 - (현제 경유지의 x) | + | (현제 경유지의 y) - (다음 경유지의 y) | +
| 0 - (다음 경유지의 x) |
or
| 6 - (현제 경유지의 x) | + | (현제 경유지의 y) - (다음 경유지의 y) | +
| 6 - (다음 경유지의 x) |

가장 짧은 조합을 찾게 되면 그 조합으로 경유지 배열을 재배치하여 저장한다.

3) 경로 생성

가장 짧은 조합으로 재배열된 경유지 배열을 반복문에 넣어 모든 경유지를 지나는 경로를 만들게 된다.

현제 경유지와 다음 경유지의 인덱스를 비교해 방향을 파악하고 세로가로 또는 가로세로 이동 중 빈공간이 있는지 판단해서 "세로가로 이동만 가능", "가로세로 이동만 가능", "세로가로, 가로세로 둘다 가능", "세로가로, 가로세로 둘다 불가"로 구분하여 각각에 케이스에 따라 로봇 주행 시뮬레이션을 하여 그 과정에서 지나가게 되는 상점들을 저장한다.

세로가로 이동 가로세로 이동

4. 작업 사진

5. 역할 분담

1) 윤태현 최단경로 구하는 프로그램 개발

1) 강민수 1단계 주행 및 상점정보 수집, 정보센터로 이동 및 추천 상점정보 읽기, 3단계 주행 및 승객 하차 프로그램 개발

1) 박재영 주행 및 승객 하차 하드웨어 개발, 연습 경기장 만들기

6. 소스 코드

```
#pragma config(Sensor, S1, LC, sensorEV3_Color, modeEV3Color_Color)
#pragma config(Sensor, S2, CC, sensorEV3_Color, modeEV3Color_Color)
#pragma config(Sensor, S3, RC, sensorEV3_Color, modeEV3Color_Color)
#pragma config(Sensor, S4, SONA, sensorEV3_Ultrasonic)

int STD_SPEED =45;
int CORRECTION_SPEED =20;

typedefstruct arr_type {
    int min_path[40][2];
    int scan_path[32][2];
} Array;

int rb_dir_x =1, rb_dir_y =0, storeInfo =0;
int map_scan_path[32][2] = {
```



```
    { 4,-1 },{ 4,0 },{ 4,1 },{ 4,2 },{ 4,3 },{ 4,4 },{ 4,5 },{ 4,6 },
    { 3,6 },{ 2,6 },{ 1,6 },{ 0,6 },
    { 0,5 },{ 0,4 },{ 0,3 },{ 0,2 },{ 0,1 },{ 0,0 },
    { 1,0 },{ 2,0 },{ 3,0 },
    { 3,1 },{ 3,2 },{ 3,3 },{ 3,4 },{ 3,5 },
    { 4,5 },{ 4,4 },{ 4,3 },{ 4,2 },{ 4,1 },{ 4,0 }
};
int goInfoCenter_path[5][2] = {
    { 4,0 },{ 3,0 },{ 2,0 },{ 1,0 },{ 0,0 }
};
//int map[5][7];
int map[5][7] = {
    {3,3,3,3,4,3,3},
    {3,0,0,0,0,0,3},
    {4,0,0,0,0,0,3},
    {3,3,3,3,3,3,3},
    {3,3,4,3,4,3,3}
};
int point_combi[24][6] = {
    {0,1,2,3,4,5},
    {0,1,2,4,3,5},
    {0,1,3,2,4,5},
    {0,1,3,4,2,5},
    {0,1,4,2,3,5},
    {0,1,4,3,2,5},

    {0,2,1,3,4,5},
    {0,2,1,4,3,5},
    {0,2,3,1,4,5},
    {0,2,3,4,1,5},
    {0,2,4,1,3,5},
    {0,2,4,3,1,5},

    {0,3,1,2,4,5},
    {0,3,1,4,2,5},
    {0,3,2,1,4,5},
    {0,3,2,4,1,5},
    {0,3,4,1,2,5},
    {0,3,4,2,1,5},

    {0,4,1,2,3,5},
    {0,4,1,3,2,5},
    {0,4,2,1,3,5},
    {0,4,2,3,1,5},
```

```
    {0,4,3,1,2,5},
    {0,4,3,2,1,5}
};
int waypoints[6][2];
int real_waypoints[4][2];
int move_path[40][2];

int CheckPath(int ax, int ay, int bx, int by);
int CheckLC();
int CheckCC();
int CheckRC();
int isBack(int rbx, int rby, int plx, int ply);
int isLeft(int rbx, int rby, int plx, int ply);
int isRight(int rbx, int rby, int plx, int ply);
void turnLeft();
void turnRight();
void turnBack();
void goInfoCenter();
void MovePath(Array way, int move_type);

//=start                                                                    main
task=============================================================================
=

task main()
{
    int i, j, k, rank =100, shortest_idx;
    Array way;

    // level 1
    setLEDColor(ledOff);

    moveMotorTarget(motorB, 360*1.4, STD_SPEED);
    moveMotorTarget(motorC, 360*1.4, STD_SPEED);
    waitUntilMotorStop(motorC);
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);

    memset(map, 0, sizeof(map));
    for(i =0; i <32; i++) {
        for(j =0; j <2; j++) {
            way.scan_path[i][j] = map_scan_path[i][j];
        }
    }
```



```
    MovePath(way, 0);

    moveMotorTarget(motorB, 100, STD_SPEED);
    moveMotorTarget(motorC, 100, STD_SPEED);
    waitUntilMotorStop(motorC);
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);

    setLEDColor(ledRed);
    eraseDisplay();
    displayBigTextLine(3, "Map scan");
    displayBigTextLine(5, "complete");

    playTone(700, 100);
    sleep(3000);

    // level 2
    setLEDColor(ledOff);
    moveMotorTarget(motorB, 100, -STD_SPEED);
    moveMotorTarget(motorC, 100, -STD_SPEED);
    waitUntilMotorStop(motorC);
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);

    goInfoCenter();

    sleep(250);
    setMotorSpeed(motorB, -5);
    setMotorSpeed(motorC, 5);
    sleep(800);
    while(!(getUSDistance(SONA) >45&&getUSDistance(SONA) <46)){;}
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);
    sleep(250);

    moveMotorTarget(motorB, 13, -STD_SPEED);
    moveMotorTarget(motorC, 13, STD_SPEED);
    waitUntilMotorStop(motorC);
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);

    moveMotorTarget(motorB, 360*2.34, STD_SPEED);
    moveMotorTarget(motorC, 360*2.33, STD_SPEED);
```

```
    waitUntilMotorStop(motorB);
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);

    storeInfo =getColorName(CC);
    sleep(250);

    moveMotorTarget(motorB, 360*1.7, -STD_SPEED);
    moveMotorTarget(motorC, 360*1.7, -STD_SPEED);
    waitUntilMotorStop(motorB);
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);
    turnRight();
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);

    //storeInfo = 4;

    // find way points
    memset(waypoints, 0, sizeof(waypoints));
    waypoints[0][0] =0; waypoints[0][1] =0;
    waypoints[5][0] =4; waypoints[5][1] =6;
    k =1;
    for(i =0; i <5; i++) {
        for(j =0; j <7; j++) {
            if(map[i][j] == storeInfo) {
                waypoints[k][0] = i;
                waypoints[k][1] = j;
                k +=1;
            }
        }
    }

    // find shortest point combination
    for(i =0; i <24; i++) {
        k =1;
        for(j =0; j <5; j++) {
            int ax = waypoints[point_combi[i][j]][1], ay = waypoints[point_combi[i][j]][0];
            int bx = waypoints[point_combi[i][j+1]][1], by = waypoints[point_combi[i][j+1]][0];
            int px =0, py =0;

            if(ax > bx) px =-1; elseif(ax < bx) px =1; else px =0;
            if(ay > by) py =-1; elseif(ay < by) py =1; else py =0;
```



```
            if(CheckPath(ax, ay, bx, by) !=0) k +=abs(ax-bx)+abs(ay-by); // path 1,2,3
            else { // path 0
                if((ax==0||ax==6)&&(bx==0||bx==6)) {
                    if(abs(1-ay)+abs(1-by) <abs(4-ay)+abs(4-by)) k +=abs(1-ay)+abs(1-by)+6;
                    else k +=abs(4-ay)+abs(4-by)+6;
                }
                else {
                    if(abs(0-ax)+abs(0-bx)                    <abs(6-ax)+abs(6-bx))                    k
+=abs(0-ax)+abs(0-bx)+abs(ay-by);
                    else k +=abs(6-ax)+abs(6-bx)+abs(ay-by);
                }
            }
        }
        k +=1;
        if(rank > k) {
            rank = k;
            shortest_idx = i;
        }
    }

    // making move path
    memset(move_path, -1, sizeof(move_path));
    move_path[0][0] =0; move_path[0][1] =-1;
    k =1;
    for(i =0; i <5; i++) {
        int          ax          =          waypoints[point_combi[shortest_idx][i]][1],          ay          =
waypoints[point_combi[shortest_idx][i]][0];
        int          bx          =          waypoints[point_combi[shortest_idx][i+1]][1],          by          =
waypoints[point_combi[shortest_idx][i+1]][0];
        int px =0, py =0;

        if(ax > bx) px =-1; elseif(ax < bx) px =1; else px =0;
        if(ay > by) py =-1; elseif(ay < by) py =1; else py =0;

        switch(CheckPath(ax, ay, bx, by)) {
            case1:
                XMOVE:
                j = ax;
                while(j != bx) {
                    move_path[k][0] = ay;
                    move_path[k][1] = j;
                    k +=1; j += px;
                }
                j = ay;
```

```
            while(j != by) {
                move_path[k][0] = j;
                move_path[k][1] = bx;
                k +=1; j += py;
            }
        break;
        case2:
            YMOVE:
            j = ay;
            while(j != by) {
                move_path[k][0] = j;
                move_path[k][1] = ax;
                k +=1; j += py;
            }
            j = ax;
            while(j != bx) {
                move_path[k][0] = by;
                move_path[k][1] = j;
                k +=1; j += px;
            }
        break;
        case3:
            if(py ==-1) goto XMOVE;
            elsegoto YMOVE;
        break;
        case0:
            if((ax==0||ax==6)&&(bx==0||bx==6)) {
                if(abs(1-ay)+abs(1-by) <abs(4-ay)+abs(4-by)) {
                    j = ay;
                    while(j !=1) {
                        move_path[k][0] = j;
                        move_path[k][1] = ax;
                        k +=1; j +=-1;
                    }
                    j = ax;
                    while(j != bx) {
                        move_path[k][0] =1;
                        move_path[k][1] = j;
                        k +=1; j += px;
                    }
                    j =1;
                    while(j != by) {
                        move_path[k][0] = j;
                        move_path[k][1] = bx;
```



```
                        k +=1; j +=1;
                    }
                }
                else {
                    j = ay;
                    while(j !=4) {
                        move_path[k][0] = j;
                        move_path[k][1] = ax;
                        k +=1; j +=1;
                    }
                    j = ax;
                    while(j != bx) {
                        move_path[k][0] =4;
                        move_path[k][1] = j;
                        k +=1; j += px;
                    }
                    j =4;
                    while(j != by) {
                        move_path[k][0] = j;
                        move_path[k][1] = bx;
                        k +=1; j +=-1;
                    }
                }
            }
            else {
                if(abs(0-ay)+abs(0-by) <abs(6-ax)+abs(6-bx)) {
                    j = ax;
                    while(j !=0) {
                        move_path[k][0] = ay;
                        move_path[k][1] = j;
                        k +=1; j +=-1;
                    }
                    j = ay;
                    while(j != by) {
                        move_path[k][0] = j;
                        move_path[k][1] =0;
                        k +=1; j += py;
                    }
                    j =0;
                    while(j != bx) {
                        move_path[k][0] = by;
                        move_path[k][1] = j;
                        k +=1; j +=1;
                    }
```

```
                    }
                    else {
                        j = ax;
                        while(j !=6) {
                            move_path[k][0] = ay;
                            move_path[k][1] = j;
                            k +=1; j +=1;
                        }
                        j = ay;
                        while(j != by) {
                            move_path[k][0] = j;
                            move_path[k][1] =6;
                            k +=1; j += py;
                        }
                        j =6;
                        while(j != bx) {
                            move_path[k][0] = by;
                            move_path[k][1] = j;
                            k +=1; j +=-1;
                        }
                    }
                }
                break;
            }
        }
        move_path[k][0] =4; move_path[k][1] =6;
        for(i =0; i <40; i++) {
            way.min_path[i][0] = move_path[i][0];
            way.min_path[i][1] = move_path[i][1];
        }

        memset(real_waypoints, 0, sizeof(real_waypoints));
        for(i =0; i <4; i++) {
            real_waypoints[i][0] = waypoints[point_combi[shortest_idx][i+1]][0];
            real_waypoints[i][1] = waypoints[point_combi[shortest_idx][i+1]][1];
        }

        eraseDisplay();
        if(real_waypoints[0][0] + real_waypoints[0][1] !=0) displayTextLine(1, "(%d%d)", 0, 0);
        displayTextLine(3, "(%d%d)", real_waypoints[0][0], real_waypoints[0][1]);
        displayTextLine(5, "(%d%d)", real_waypoints[1][0], real_waypoints[1][1]);
        displayTextLine(7, "(%d%d)", real_waypoints[2][0], real_waypoints[2][1]);
        if(real_waypoints[3][0] ==4&& real_waypoints[3][1] ==6){
            displayTextLine(9, "(%d%d)", 4, 6);
```



```
        }
        else {
            displayTextLine(9, "(%d%d)", real_waypoints[3][0], real_waypoints[3][1]);
            displayTextLine(12, "(%d%d)", 4, 6);
        }
        sleep(5000);

        // level 3
        setLEDColor(ledOff);
        rb_dir_x =1; rb_dir_y =0;
        MovePath(way, 1);
        if(rb_dir_x ==0&& rb_dir_y ==-1) turnRight();
        elseif(rb_dir_x ==0&& rb_dir_y ==1) turnLeft();
        setMotorSpeed(motorB, 0);
        setMotorSpeed(motorC, 0);

        moveMotorTarget(motorB, 100, STD_SPEED);
        moveMotorTarget(motorC, 100, STD_SPEED);
        waitUntilMotorStop(motorC);
        setMotorSpeed(motorB, 0);
        setMotorSpeed(motorC, 0);
}

//=end                                                                    main
task=================================================================================
===

//=start                                                                  define
functions============================================================
int CheckPath(int ax, int ay, int bx, int by) {
        int i, px, py, path_a =0, path_b =0;

        if(ax > bx) px =-1; elseif(ax < bx) px =1; else px =0;
        if(ay > by) py =-1; elseif(ay < by) py =1; else py =0;

        for(i =0; i <abs(ax-bx)+1; i++) if(map[ay][ax+i*px] ==0) path_a =1;
        for(i =0; i <abs(ay-by)+1; i++) if(map[ay+i*py][bx] ==0) path_a =1;
        for(i =0; i <abs(ay-by)+1; i++) if(map[ay+i*py][ax] ==0) path_b =1;
        for(i =0; i <abs(ax-bx)+1; i++) if(map[by][ax+i*px] ==0) path_b =1;

        if(path_a ==0&& path_b ==0) return3;
        elseif(path_a ==1&& path_b ==0) return2;
        elseif(path_a ==0&& path_b ==1) return1;
        elsereturn0;
```

```
}

int CheckLC() {
    int color =getColorName(LC);
    if(color !=2&& color !=3&& color !=4&& color !=5) return1;
    elsereturn0;
}

int CheckCC() {
    int color =getColorName(CC);
    if(color !=2&& color !=3&& color !=4&& color !=5) return1;
    else {
        sleep(25);
        if(color ==getColorName(CC)) return0;
        elsereturn1;
    }
}

int CheckRC() {
    int color =getColorName(RC);
    if(color !=2&& color !=3&& color !=4&& color !=5) return1;
    elsereturn0;
}

intis Back(int rbx, int rby, int plx, int ply) {
    if((rbx == plx && rby != ply) || (rbx != plx && rby == ply)) return1;
    elsereturn0;
}
// 1 0
int isLeft(int rbx, int rby, int plx, int ply) {
    if((rbx ==0&& rby ==1) && (plx ==1&& ply ==0))      return1;
    elseif((rbx ==0&& rby ==-1) && (plx ==-1&& ply ==0)) return1;
    elseif((rbx ==1&& rby ==0) && (plx ==0&& ply ==-1)) return1;
    elseif((rbx ==-1&& rby ==0) && (plx ==0&& ply ==1)) return1;
    elsereturn0;
}

int isRight(int rbx, int rby, int plx, int ply) {
    if((rbx ==0&& rby ==1) && (plx ==-1&& ply ==0))    return1;
    elseif((rbx ==0&& rby ==-1) && (plx ==1&& ply ==0)) return1;
    elseif((rbx ==1&& rby ==0) && (plx ==0&& ply ==1)) return1;
    elseif((rbx ==-1&& rby ==0) && (plx ==0&& ply ==-1)) return1;
    elsereturn0;
}
```



```
void turnLeft() {
    setMotorSpeed(motorB, -20);
    setMotorSpeed(motorC, 20);
    sleep(1000);
    while(getColorName(RC) != colorBlack &&getColorName(CC) != colorBlack){;}
    if(getColorName(CC) == colorBlack) {
        while(getColorName(CC) == colorBlack) {
            setMotorSpeed(motorB, 10);
            setMotorSpeed(motorC, 10);
        }
        while(getColorName(RC) != colorBlack) {
            setMotorSpeed(motorB, -20);
            setMotorSpeed(motorC, 20);
        }
    }
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);

    while(getColorName(LC) == colorBlack) {
        setMotorSpeed(motorB, -20);
        setMotorSpeed(motorC, 20);
    }
    while(getColorName(RC) == colorBlack) {
        setMotorSpeed(motorB, 20);
        setMotorSpeed(motorC, -20);
    }
}

voidturnRight() {
    setMotorSpeed(motorB, 20);
    setMotorSpeed(motorC, -20);
    sleep(1000);
    while(getColorName(LC) != colorBlack &&getColorName(CC) != colorBlack){;}
    if(getColorName(CC) == colorBlack) {
        while(getColorName(CC) == colorBlack) {
            setMotorSpeed(motorB, 10);
            setMotorSpeed(motorC, 10);
        }
        while(getColorName(LC) != colorBlack) {
            setMotorSpeed(motorB, 20);
            setMotorSpeed(motorC, -20);
        }
    }
```

```
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);

    while(getColorName(RC) == colorBlack) {
        setMotorSpeed(motorB, 20);
        setMotorSpeed(motorC, -20);
    }
    while(getColorName(LC) == colorBlack) {
        setMotorSpeed(motorB, -20);
        setMotorSpeed(motorC, 20);
    }
}

voidturnBack() {
    turnLeft();
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);
    sleep(100);
    turnLeft();
}

voidgoInfoCenter() {
    turnRight();
    for(int i =0; i <8; i++) {
        moveMotorTarget(motorB, 200, STD_SPEED);
        moveMotorTarget(motorC, 200, STD_SPEED);
        waitUntilMotorStop(motorC);
        setMotorSpeed(motorB, 0);
        setMotorSpeed(motorC, 0);
        moveMotorTarget(motorB, 60, -STD_SPEED);
        moveMotorTarget(motorC, 60, STD_SPEED);
        waitUntilMotorStop(motorC);
        setMotorSpeed(motorB, 0);
        setMotorSpeed(motorC, 0);

        setMotorSpeed(motorB, CORRECTION_SPEED);
        setMotorSpeed(motorC, -CORRECTION_SPEED);
        while(getColorName(CC) != colorBlack){;}
        setMotorSpeed(motorB, 0);
        setMotorSpeed(motorC, 0);

        moveMotorTarget(motorB, 5, STD_SPEED);
        moveMotorTarget(motorC, 5, -STD_SPEED);
        waitUntilMotorStop(motorC);
```



```
        setMotorSpeed(motorB, 0);
        setMotorSpeed(motorC, 0);

        setMotorSpeed(motorB, STD_SPEED);
        setMotorSpeed(motorC, STD_SPEED);
        while(getColorName(CC) == colorBlack) {;}
        setMotorSpeed(motorB, 0);
        setMotorSpeed(motorC, 0);
    }
}

voidMovePath(Array way, int move_type) {
    int i, j =0, count =0, ax, ay, bx, by, px, py, isDrop =0;
    if(move_type ==0) count =31;
    elseif(move_type ==1) count =39;

    for(i =0; i < count; i++) {
        if(move_type ==0) {
            ax = way.scan_path[i][1];
            ay = way.scan_path[i][0];
            bx = way.scan_path[i+1][1];
            by = way.scan_path[i+1][0];
        }
        elseif(move_type ==1) {
            ax = way.min_path[i][1];
            ay = way.min_path[i][0];
            bx = way.min_path[i+1][1];
            by = way.min_path[i+1][0];
        }

        if(move_type ==1&& bx ==-1&& by ==-1) break;

        if(ax > bx) px =-1;
        elseif(ax < bx) px =1;
        else px =0;
        if(ay > by) py =-1;
        elseif(ay < by) py =1;
        else py =0;

        if(bx ==-1&& by ==-1) break;

        eraseDisplay();
        if(isBack(rb_dir_x, rb_dir_y, px, py)) {
            displayBigTextLine(3, "Turn Back");
```

```
            turnBack();
        }
        elseif(isLeft(rb_dir_x, rb_dir_y, px, py)) {
            displayBigTextLine(3, "Turn Left");
            turnLeft();
        }
        elseif(isRight(rb_dir_x, rb_dir_y, px, py)) {
            displayBigTextLine(3, "Turn Right");
            turnRight();
        }
        else {
            displayBigTextLine(3, "Go Straight");
        }
        rb_dir_x = px; rb_dir_y = py;

        resetMotorEncoder(motorB);
        resetMotorEncoder(motorC);

        while(CheckCC()) {
            if(CheckLC() ==0||getColorName(LC) == colorBlack) {
                setMotorSpeed(motorB, 0);
                setMotorSpeed(motorC, CORRECTION_SPEED);
            }
            elseif(CheckRC() ==0||getColorName(RC) == colorBlack){
                setMotorSpeed(motorB, CORRECTION_SPEED);
                setMotorSpeed(motorC, 0);
            }
            else {
                setMotorSpeed(motorB, STD_SPEED);
                setMotorSpeed(motorC, STD_SPEED);
            }
        }
        moveMotorTarget(motorB, 90, CORRECTION_SPEED);
        moveMotorTarget(motorC, 90, CORRECTION_SPEED);
        waitUntilMotorStop(motorC);
        setMotorSpeed(motorB, 0);
        setMotorSpeed(motorC, 0);

        playTone(700,10);
        if(move_type ==0&& map[by][bx] ==0) map[by][bx] =getColorName(CC);
        if(move_type ==1&& ax == real_waypoints[j][1] && ay == real_waypoints[j][0]) {
            moveMotorTarget(motorA, 200, -45);
            waitUntilMotorStop(motorA);
            setMotorSpeed(motorA, 0);
```



```
            if(j !=3) j +=1;
        }

        moveMotorTarget(motorB, 120, CORRECTION_SPEED);
        moveMotorTarget(motorC, 120, CORRECTION_SPEED);
        waitUntilMotorStop(motorC);
        setMotorSpeed(motorB, 0);
        setMotorSpeed(motorC, 0);

        if(getMotorEncoder(motorB) <200) i -=1;
    }
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);
}
//=end                                                                    define
functions================================================================
```

이번 프로젝트에 대한 반성 / 다음 프로젝트 계획
- 2단계에서 승객 하차 위치를 만들 때 승객을 하차시킨 위치를 다시 방문하지 않게 순서를 바꾸어 주는 기능은 아직 넣지 못했다. 추후 그 기능을 넣을 것이다. - 처음 개발을 시작할 때는 아직 경기 맵이 판매가 되지 않은 상태라 일단 플로터로 뽑은 임시 맵을 사용하여 작업을 했다. 그 후에 경기 맵을 구매한 뒤에도 경기 맵과 플로터로 뽑은 맵이 비슷할 거라고 생각해서 작업을 계속 진행했다. 하지만 경기 맵은 반사율도 다르고 각종 이미지들이 출력되어있다 보니 색을 인식하는데 문제가 있었다. 그래서 그때까지 만든 코드를 버리고 새로 만들어야 되었다. 그로인해 시간을 많이 낭비하게 되어 많이 아쉬움이 남는 부분이다.

지도교사 확인 및 의견	
1단계부터 3단계까지 완주를 확인하였음. 모든 것을 처음배우는 1학년 팀(S-ROBO2)을 위주로 개발과정 일정을 진행하고 도와주다보니 구입한 경기장의 추천 상점정보 구성도 늦어지게 되었고 개발과정을 도와주지 못하여 미안한 마음을 가지고 있음. 각 단계별 주행 규정을 살펴 규정에 맞게 주행하도록 다시 점검하고 1단계에서 개발되는 기본 주행을 다시 살펴, 속도를 높이면서 안정적 주행을 할 수 있도록 할 필요가 있음. 3단계 출발 전 주행경로를 LCD에 표시할 때 문자의 위치와 크기를 수정해서 가독성을 높여야함. 개발과정에서의 실패와 시행착오는 소프트웨어가 실제 세계와 만나는 피지컬 컴퓨팅 또는 임베디드 시스템에서는 어쩔 수 없는 부분일 수 있으며 현실에서의 다양한 환경에 대하여 제한된 정보와 예기치 않은 오류를 접하여 이를 극복하는 과정은 배우는 과정에서 반드시 겪어야 하는 학습의 한 과정이며 아무런 문제없이 완료하는 것 보다 이러한 과정을 거치는 것을 통하여 더 많은 것을 배울 수 있음을 이해해야함.	(서명)