제16회 임베디드SW경진대회

개발일지 [주니어 임베디드SW 챌린저]

작품명 : 전통시장 스마트 셔틀 로봇 운행

팀 명 : S-ROBO2

팀번호	17039
대표소속	송도중학교
팀원이름	임현태, 나신영, 김규민
지도교사(멘토) 이름	노영택
시연 동영상(URL)	https://youtu.be/WQL7awImGEQ

임베디드 소프트웨어 경진대회
The World Embedded Software Contest

개발일지 목차

프로젝트1 (1단계 주행)	
프로젝트 기간 및 목표	
기간	7월 30일 ~ 8월 8일
목표	1단계 주행 소프트웨어와 하드웨어 개발

프로젝트 활동내용

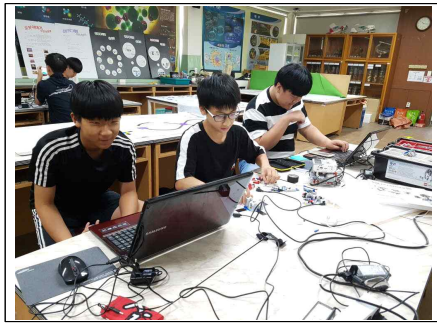
<1> 팀원 역할

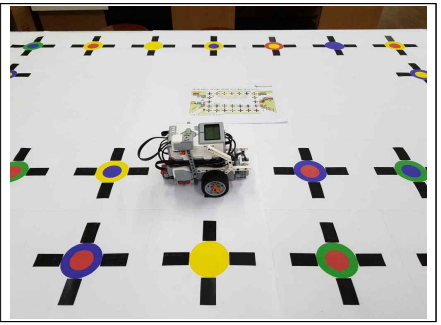
현태 : 로봇 에듀케이터 기반에 센서 2개 추가와 주행 안전화를 위해 센서부 전면 수정 그리고 승객 승하차 장치 연구와 저장된 상점에서 방문할 상점 위치 찾아 저장하기 알고리즘 연구를 하였습니다.

신영 : 1단계 주행을 위한 상점 이동 함수 개발(직진, 우회전, 좌회전, 뒤돌기), 1단계 주행 시 상점 점보 저장 배열 선언과 초기화, 1단계 주행 시험과 오류 수정, 저장된 상점에서 방문할 상점 위치 찾아 저장하기 알고리즘 연구- 시험 프로그램 작성을 하였습니다.

규민: 1단계 주행 개발과정에서 코드와 주행을 살펴 주행 오류를 수정하고 안정된 주행을 하도록 도와주었고 3단계 주행 알고리즘을 미리 검토하고 2단계와 3단계 개발을 준비 하였습니다.

<2> 사진





<3> 작성한 프로그램 소스----------

```
int step_1();
int step_2();
int untilblackline();
int between_blackline();
int test_S1();
int test_S3();
int right_turn();
int left_turn();
int go_center();
int go_straight();
int go_right();
int go_left();
int store_information[5][7];
int go_back();
int visit_index;
int visit_X[10];
int visit_Y[10];
int read_color;
int distance(int my, int mx, int ny, int nx);
task main(){
step_1();
}
// step function ++++++++++++++++++++++
int step_1()
{
untilblackline();
go_center();
store_information[4][0]=getColorName(S4);
go_left();
store_information[3][0]=getColorName(S4);
go_straight();
store_information[2][0]=getColorName(S4);
go_straight();
store_information[1][0]=getColorName(S4);
go_straight();
store_information[0][0]=getColorName(S4);
go_right();
store_information[0][1]=getColorName(S4);
go_straight();
store_information[0][2]=getColorName(S4);
go_straight();
store_information[0][3]=getColorName(S4);
go_straight();
store_information[0][4]=getColorName(S4);
go_straight();
```


임베디드 소프트웨어 경진대회
The World Embedded Software Contest

```
store_information[0][5]=getColorName(S4);
go_straight();
store_information[0][6]=getColorName(S4);
go_right();
store_information[1][6]=getColorName(S4);
go_straight();
store_information[2][6]=getColorName(S4);
go_straight();
store_information[3][6]=getColorName(S4);
go_right();
store_information[3][5]=getColorName(S4);
go_straight();
store_information[3][4]=getColorName(S4);
go_straight();
store_information[3][3]=getColorName(S4);
go_straight();
store_information[3][2]=getColorName(S4);
go_straight();
store_information[3][1]=getColorName(S4);
go_left();
store_information[4][1]=getColorName(S4);
go_left();
store_information[4][2]=getColorName(S4);
go_straight();
store_information[4][3]=getColorName(S4);
go_straight();
store_information[4][4]=getColorName(S4);
go_straight();
store_information[4][5]=getColorName(S4);
go_straight();
store_information[4][6]=getColorName(S4);
go_back();
go_straight();
go_straight();
go_straight();
go_straight();
go_straight();
```



```
return 0;
}
// function defenition =================
int between_blackline()
{
                moveMotorTarget(motorB, 130,20);
                moveMotorTarget(motorC, 130,20);
        /*      waitUntilMotorStop(motorB);
                playTone(1000,15);
        sleep(5000);*/
 return 0;
}

//1111111111111111111111
int test_S1()
{
        if(getColorName(S1)!= colorYellow && getColorName(S1)!= colorBlue &&
    getColorName(S1)!= colorGreen
&& getColorName(S1)!= colorRed)
        {
                return 1;
        }
        else
        {
                return 0;
        }
}
//22222222222222222222222
int test_S3()
{
        if(getColorName(S3)!= colorGreen && getColorName(S3)!= colorYellow &&
    getColorName(S3)!= colorBlue && getColorName(S3)!= colorRed)
        {
                return 1;
        }
        else
```

```
        {
            return 0;
        }
}
//33333333333333333333333333
int right_turn()
        {
        /*      while(getColorName(S4)!= colorBlack)
        {
        setMotorSpeed(motorB,-10);
        setMotorSpeed(motorC,-10);
        }
            /*while(test_S1()&& test_S3())
        {
            setMotorSpeed(motorB, -10);
            setMotorSpeed(motorC, -10);
        }*/

        while(getColorName(S4)== colorBlack)
        {
            setMotorSpeed(motorB,20);
            setMotorSpeed(motorC,-20);
        }
        while(getColorName(S4)!= colorBlack)
        {
            setMotorSpeed(motorB,20);
            setMotorSpeed(motorC,-20);
        }
  while(getColorName(S4)== colorBlack)
        {
            setMotorSpeed(motorB,20);
            setMotorSpeed(motorC,20);
        }
        return 0;
        }
 //4444444444444444444444444444
 int left_turn()
```



```
        {
/*      while(getColorName(S4)!= colorBlack)
        {
            setMotorSpeed(motorB,-10);
            setMotorSpeed(motorC,-10);
        }
        /*while(test_S1()&& test_S3())
        {
            setMotorSpeed(motorB, -10);
            setMotorSpeed(motorC, -10);
        }*/
        while(getColorName(S4)== colorBlack)
        {
            setMotorSpeed(motorB,-20);
            setMotorSpeed(motorC,20);
        }
        while(getColorName(S4)!= colorBlack)
        {
            setMotorSpeed(motorB,-20);
            setMotorSpeed(motorC,20);
        }
        while(getColorName(S4)== colorBlack)
        {
        setMotorSpeed(motorB,20);
            setMotorSpeed(motorB,20);
            setMotorSpeed(motorC,20);
}
         return 0;
        }
        //55555555555555555555555555555555
        int go_center()
        {
            while(test_S1() || test_S3())
            {
                if(getColorName(S1)== colorBlack || test_S1()==0)
                {
                    setMotorSpeed(motorB,0);
```

```
                setMotorSpeed(motorC,5);
            }
            else  if(test_S3()==0||getColorName(S3)==colorBlack)
            {
                setMotorSpeed(motorB,5);
                setMotorSpeed(motorC,0);
            }
            else
            {
                setMotorSpeed(motorB,20);
                setMotorSpeed(motorC,20);
            }
            }
        setMotorSpeed(motorB,0);
        setMotorSpeed(motorC,0);
        playTone(784,  15);
return  0;
}
//6666666666666666666666666666666
int  go_straight()
{
        untilblackline();
while(getColorName(S4)==  colorBlack)
{
        setMotorSpeed(motorB,20);
        setMotorSpeed(motorC,20);
}
while(getColorName(S1)==  colorBlack)
{
        setMotorSpeed(motorB,0);
        setMotorSpeed(motorC,5);
}
while(getColorName(S3)==  colorBlack)
{
        setMotorSpeed(motorB,5);
        setMotorSpeed(motorC,0);
}
```



```
between_blackline();
 go_center();
 return  0;
}
//7777777777777777777777777777777777777
int  go_right()
{
        untilblackline();
        while(getColorName(S4)==  colorBlack)
{
        setMotorSpeed(motorB,20);
        setMotorSpeed(motorC,20);
}
        while(getColorName(S4)!=  colorBlack)
{
        setMotorSpeed(motorB,-20);
        setMotorSpeed(motorC,-20);
}
while(test_S1()&&test_S3())
{
        setMotorSpeed(motorB,-20);
        setMotorSpeed(motorC,-20);
}
 right_turn();
 while(getColorName(S4)==  colorBlack)
{
        setMotorSpeed(motorB,20);
        setMotorSpeed(motorC,20);
}
while(getColorName(S1)==  colorBlack)
{
        setMotorSpeed(motorB,0);
        setMotorSpeed(motorC,5);
}
while(getColorName(S3)==  colorBlack)
{
        setMotorSpeed(motorB,5);
```

```
                setMotorSpeed(motorC,0);
        }
between_blackline();
        go_center();
         return 0;
        }
        //88888888888888888888888888888888
        int go_left()
        {
                untilblackline();
                while(getColorName(S4)== colorBlack)
        {
                setMotorSpeed(motorB,20);
                setMotorSpeed(motorC,20);
        }
        while(getColorName(S4)!= colorBlack)
        {
                setMotorSpeed(motorB,-20);
                setMotorSpeed(motorC,-20);
        }
        while(test_S1()&&test_S3())
        {
                setMotorSpeed(motorB,-20);
                setMotorSpeed(motorC,-20);
        }
                left_turn();
        while(getColorName(S1)== colorBlack)
        {
                setMotorSpeed(motorB,0);
                setMotorSpeed(motorC,5);
        }
        while(getColorName(S3)== colorBlack)
        {
                setMotorSpeed(motorB,5);
                setMotorSpeed(motorC,0);
        }
between_blackline();
```



```
        go_center();
         return 0;
        }
        //99999999999999999999999
        int go_back()
        {
                        untilblackline();
        while(getColorName(S4)== colorBlack)
        {
                setMotorSpeed(motorB,20);
                setMotorSpeed(motorC,20);
        }
        while(getColorName(S4)!=colorBlack)
        {
                setMotorSpeed(motorB,-20);
                setMotorSpeed(motorC,-20);
        }
        while(test_S1()&&test_S3())
        {
                setMotorSpeed(motorB,-20);
                setMotorSpeed(motorC,-20);
        }
        while(getColorName(S4)==colorBlack)
        {
                setMotorSpeed(motorB,-20);
                setMotorSpeed(motorC,20);
        }
        while(getColorName(S4)!=colorBlack)
        {
                setMotorSpeed(motorB,-20);
                setMotorSpeed(motorC,20);
        }
                while(getColorName(S4)==colorBlack)
        {
                setMotorSpeed(motorB,-20);
                setMotorSpeed(motorC,20);
        }
```

```
        while(getColorName(S4)!=colorBlack)
        {
            setMotorSpeed(motorB,-20);
            setMotorSpeed(motorC,20);
        }
         setMotorSpeed(motorB,0);
         setMotorSpeed(motorC,0);
            sleep(1000);
        while(getColorName(S4)== colorBlack)
        {
            setMotorSpeed(motorB,20);
            setMotorSpeed(motorC,20);
        }
        while(getColorName(S1)== colorBlack)
        {
            setMotorSpeed(motorB,0);
            setMotorSpeed(motorC,5);
        }
        while(getColorName(S3)== colorBlack)
        {
            setMotorSpeed(motorB,5);
            setMotorSpeed(motorC,0);
        }
        between_blackline();
        go_center();
         return 0;
}
```

이번 프로젝트에 대한 반성 / 다음 프로젝트 계획

/ 다음 목표

2 단계 최단 거리 주행 경로 구하기

3 단계 주행 방법 개발하고 프로그램하기

지도교사 확인 및 의견

1단계 주행 완료 확인함. 상점을 건너갈 때, 건너와서 센터에 들어갈 때 오류 가능성에 대한 처리가 필요함. 주행 중 상점 정보를 정확히 읽어 들이는 지는 시험되지 않았으므로 2단계와 3단계 개발 중에 확인 되어야함. 하드웨어는 구조가 부실하여 수정할 필요가 있어 보이고 승객탑승하차 방법이 공지되면 전체적인 수정이 필요할 것임. 2단계와 3단계는 분리해서 진행하게 되므로 2단계에서 사용하는 변수 중 3단계에서 참조할 변수를 맞추어 개발해야함.	(서명)

임베디드 소프트웨어 경진대회
The World Embedded Software Contest

프로젝트2 (최단거리 주행)	
프로젝트 기간 및 목표	
기간	8월9일~8월25일
목표	2단계 최단거리 알고리즘 구현, 3단계 주행 개발
프로젝트 활동내용	

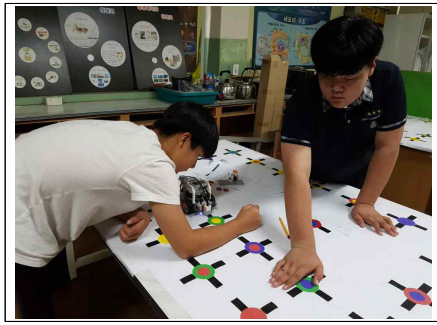
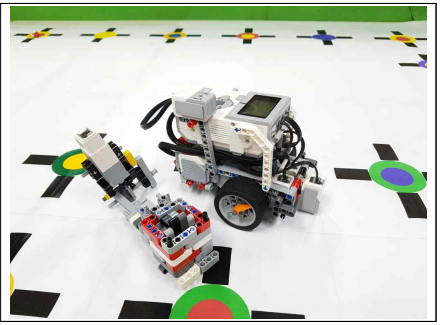
<1> 팀원 역할

현태: 승객승하차 장치를 연구하고 3단계-가로방향 우선 주행 방법에 맞추어 2단계 두 점 거리 구하기와 출발부터 도착까지 거리 구하기 그리고 최단거리 방문순서 구하기와 최단거리 경로로부터 실제 경로 구하기 마지막으로 손님 승하차 기능 추가하기 등 최단거리 주행 경로를 구하고 승객 하차 위치 설정하도록 프로그램 하였습니다.

신영: 2단계에서 구할 실제 주행경로를 따라 3단계 주행을 하도록 프로그램하고 손님을 하차할 위치에 하차 시킬 수 있도록 프로그램 코드를 추가 했습니다. 3단계 주행 시 상점에서 상점으로 이동시 생기는 이탈 현상과 센터를 잘못 인식하는 문제 해결을 위한 작업을 하였습니다.

규민: 3단계 주행 알고리즘에 따라 신영이와 함께 코딩 과정에 참여하고 프로그램 검토하고 주행 분석을 하였습니다. between_blackline함수가 짧아지는 현상이 센터에서 출발하여 출발상점의 블랙라인에서의 인식 잘못으로 다음 상점의 블랙라인에 도달하지 못하는 오류 등을 수정 하였습니다.

<2> 사진

<3-1> 작성한 프로그램 소스 /2단계----------

```
int step_2();
int store_information[5][7]=
{
        {9, 9, 5, 9, 5, 9, 9},
        {9,-1,-1,-1,-1,-1, 9},
        {5,-1,-1,-1,-1,-1, 9},
        {9, 9, 5, 9, 9, 9, 9},
        {9, 9, 9, 9, 9, 9, 9}
};
int visit_index;
int visit_X[4];
int visit_Y[4];
int read_color;
int distance(int my, int mx, int ny, int nx);
int all_path[24][4]=
{
        {0,1,2,3},{0,1,3,2},
        {0,2,1,3},{0,2,3,1},
        {0,3,1,2},{0,3,2,1},
        {1,0,2,3},{1,0,3,2},
        {1,2,0,3},{1,2,3,0},
        {1,3,0,2},{1,3,2,0},
        {2,0,1,3},{2,0,3,1},
        {2,1,0,3},{2,1,3,0},
        {2,3,0,1},{2,3,1,0},
        {3,0,1,2},{3,0,2,1},
        {3,1,0,2},{3,1,2,0},
        {3,2,0,1},{3,2,1,0}
};
int min_distance=100000;
int min_path[4];
int insert_path(int my, int mx, int ny, int nx);
int real_path_Y[15];
int real_path_X[15];
int real_path_drop[15];
int real_path_index;
task main()
{
        step_2();
}
//====================
int step_2()
```

```
{
		waitForButtonPress();
		read_color=getColorName(S4);
		visit_index=0;
		for(int i=0; i<5;i=i+1)
		{
			for(int j=0; j<7;j=j+1)
			{
				if(read_color==store_information[i][j])
				{
					visit_X[visit_index]=j;
					visit_Y[visit_index]=i;
					visit_index=visit_index+1;
				}
			}
		}
		int distance_sun=0;
		for(int i=0; i<24; i=i+1)
		{
			distance_sun=
			distance(0,0,visit_Y[all_path[i][0]],visit_X[all_path[i][0]])

	+distance(visit_Y[all_path[i][0]],visit_X[all_path[i][0]],visit_Y[all_path[i][1]],visit_X[all_path[i][1]])

	+distance(visit_Y[all_path[i][1]],visit_X[all_path[i][1]],visit_Y[all_path[i][2]],visit_X[all_path[i][2]])

	+distance(visit_Y[all_path[i][2]],visit_X[all_path[i][2]],visit_Y[all_path[i][3]],visit_X[all_path[i][3]])
				+distance(visit_Y[all_path[i][3]],visit_X[all_path[i][3]],4,6);
			if(min_distance>distance_sun)
			{
				min_distance=distance_sun;
				min_path[0]=all_path[i][0];
				min_path[1]=all_path[i][1];
				min_path[2]=all_path[i][2];
				min_path[3]=all_path[i][3];
			}
		}
		real_path_index=0;
```



```
		insert_path(0,0,visit_Y[min_path[0]],visit_X[min_path[0]]);
		real_path_Y[real_path_index]=visit_Y[min_path[0]];
		real_path_X[real_path_index]=visit_X[min_path[0]];
		real_path_drop[real_path_index]=1;
		real_path_index=real_path_index+1;

	insert_path(visit_Y[min_path[0]],visit_X[min_path[0]],visit_Y[min_path[1]],visit_X[min_path[1]]);
		real_path_Y[real_path_index]=visit_Y[min_path[1]];
		real_path_X[real_path_index]=visit_X[min_path[1]];
		real_path_drop[real_path_index]=1;
		real_path_index=real_path_index+1;

	insert_path(visit_Y[min_path[1]],visit_X[min_path[1]],visit_Y[min_path[2]],visit_X[min_path[2]]);
		real_path_Y[real_path_index]=visit_Y[min_path[2]];
		real_path_X[real_path_index]=visit_X[min_path[2]];
		real_path_drop[real_path_index]=1;
		real_path_index=real_path_index+1;
insert_path(visit_Y[min_path[2]],visit_X[min_path[2]],visit_Y[min_path[3]],visit_X[min_path[3]]);
		real_path_Y[real_path_index]=visit_Y[min_path[3]];
		real_path_X[real_path_index]=visit_X[min_path[3]];
		real_path_drop[real_path_index]=1;
		real_path_index=real_path_index+1;
		insert_path(visit_Y[min_path[3]],visit_X[min_path[3]],4,6);
		real_path_Y[real_path_index]=4;
		real_path_X[real_path_index]=6;
		real_path_drop[real_path_index]=1;
		real_path_index=real_path_index+1;
		for(int i=0; i<real_path_index; i=i+1)
		{
			eraseDisplay();
			displayBigStringAt(70, 50, "%d", real_path_Y[i]);
			displayBigStringAt(130, 50, "%d", real_path_X[i]);
			displayBigStringAt(70, 70, "%d", real_path_index);
			sleep(3000);
		}
		/*
		displayBigStringAt(70, 50, "%d", min_path[1]);
```

```
		displayBigStringAt(100, 50, "%d", min_path[2]);
		displayBigStringAt(130, 50, "%d", min_path[3]);
		sleep(5000);*/
		return 0;
}
//============================
/*int insert_path(int my, int mx, int ny, int nx)
{
		int result=0;
		int result2=0;
		if(my<=2 && ny<=2)
		{
			if(my!=0 && ny!=0 && mx!= nx)
			{
				result=abs(0-my)+abs(0-ny)+7;//4
				result2=abs(3-my)+abs(3-ny)+7;
				if(result2<result)
				{
					//insert path 00,06
					real_path_Y[real_path_index]=0;
					real_path_X[real_path_index]=0;
					real_path_drop[real_path_index]=0;
					real_path_index=real_path_index+1;
					real_path_Y[real_path_index]=0;
					real_path_X[real_path_index]=6;
					real_path_drop[real_path_index]=0;
					real_path_index=real_path_index+1;
				}
				else
				{
					//insert path 30,36
					real_path_Y[real_path_index]=3;
					real_path_X[real_path_index]=0;
					real_path_drop[real_path_index]=0;
					real_path_index=real_path_index+1;
					real_path_Y[real_path_index]=3;
					real_path_X[real_path_index]=6;
```



```
					real_path_drop[real_path_index]=0;
					real_path_index=real_path_index+1;
				}
			}
			else if((my==1||my==2)&&(nx>0&&nx<6))
			{
				if(mx==0)
				{
					//insert path 00
					real_path_Y[real_path_index]=0;
					real_path_X[real_path_index]=0;
					real_path_drop[real_path_index]=0;
					real_path_index=real_path_index+1;
				}
				else if(mx==6)
				{
					//insert path 06
					real_path_Y[real_path_index]=0;
					real_path_X[real_path_index]=6;
					real_path_drop[real_path_index]=0;
					real_path_index=real_path_index+1;
				}
			}
			else
			{
				//nothing ?
			}
		}
		else if(my>2 && ny>2)
		{
			//nothing
		}
		else
		{//////////////
			result=abs(my-3)+abs(mx-0)+abs(ny-3)+abs(nx-0);//3
			result2=abs(my-3)+abs(mx-6)+abs(ny-3)+abs(nx-6);
			if(result2<result)
```

```
		{
			//insert path 36
			real_path_Y[real_path_index]=3;
			real_path_X[real_path_index]=6;
			real_path_drop[real_path_index]=0;
			real_path_index=real_path_index+1;
		}
		else
		{
			//insert path 30
			real_path_Y[real_path_index]=3;
			real_path_X[real_path_index]=0;
			real_path_drop[real_path_index]=0;
			real_path_index=real_path_index+1;
		}
	}
	return 0;
}
*/
int insert_path(int my, int mx, int ny, int nx)
{
	int result=0;
	int result2=0;
	if(my==0)//uper
	{
		//(ny<=2) nothing
		if(ny>2)
		{
			result=abs(my-3)+abs(mx-0)+abs(ny-3)+abs(nx-0);
			result2=abs(my-3)+abs(mx-6)+abs(ny-3)+abs(nx-6);
			if(result2<result)
			{
				//insert path 36
				real_path_Y[real_path_index]=3;
				real_path_X[real_path_index]=6;
				real_path_drop[real_path_index]=0;
				real_path_index=real_path_index+1;
```



```
			}
			else
			{
				//insert path 30
				real_path_Y[real_path_index]=3;
				real_path_X[real_path_index]=0;
				real_path_drop[real_path_index]=0;
				real_path_index=real_path_index+1;
			}
		}
	}
	else if(my==1||my==2)//middle
	{
		if(ny==0)
		{
			if(mx==0)
			{
				//0mx
				real_path_Y[real_path_index]=0;
				real_path_X[real_path_index]=0;
				real_path_drop[real_path_index]=0;
				real_path_index=real_path_index+1;
			}
			else if(mx==6)
			{
				real_path_Y[real_path_index]=0;
				real_path_X[real_path_index]=6;
				real_path_drop[real_path_index]=0;
				real_path_index=real_path_index+1;
			}
		}
		else if(ny>2)
		{
			if(mx==0)
			{
				//insert path 30
				real_path_Y[real_path_index]=3;
```

```
                        real_path_X[real_path_index]=0;
                        real_path_drop[real_path_index]=0;
                        real_path_index=real_path_index+1;
                }
                else if(mx==6)
                {
                        //insert path 36
                        real_path_Y[real_path_index]=3;
                        real_path_X[real_path_index]=6;
                        real_path_drop[real_path_index]=0;
                        real_path_index=real_path_index+1;
                }
        }
}
else
{
        if(ny==0)
        {
                result=abs(my-0)+abs(mx-0)+abs(ny-0)+abs(nx-0);
                result2=abs(my-0)+abs(mx-6)+abs(ny-0)+abs(nx-6);
                if(result2<result)
                {
                        real_path_Y[real_path_index]=0;
                        real_path_X[real_path_index]=6;
                        real_path_drop[real_path_index]=0;
                        real_path_index=real_path_index+1;
                }
                else
                {
                        real_path_Y[real_path_index]=0;
                        real_path_X[real_path_index]=0;
                        real_path_drop[real_path_index]=0;
                        real_path_index=real_path_index+1;
                }
        }
                        /*//0mx, 0nx || 0my, 0ny
                        real_path_Y[real_path_index]=0;
```



```
                        real_path_X[real_path_index]=0;
                        real_path_drop[real_path_index]=0;
                        real_path_index=real_path_index+1;
                        real_path_Y[real_path_index]=0;
                        real_path_X[real_path_index]=6;
                        real_path_drop[real_path_index]=0;
                        real_path_index=real_path_index+1;
                }
                else
                {
                        real_path_Y[real_path_index]=3;
                        real_path_X[real_path_index]=0;
                        real_path_drop[real_path_index]=0;
                        real_path_index=real_path_index+1;
                        real_path_Y[real_path_index]=3;
                        real_path_X[real_path_index]=6;
                        real_path_drop[real_path_index]=0;
                        real_path_index=real_path_index+1;*/
                //(my==nx) nothing
                /*if(ny==0)
                {
                //insert path 00
                real_path_Y[real_path_index]=0;
                real_path_X[real_path_index]=0;
                real_path_drop[real_path_index]=0;
                real_path_index=real_path_index+1;
        }
        else
        {
                //insert path 06
                real_path_Y[real_path_index]=0;
                real_path_X[real_path_index]=6;
                real_path_drop[real_path_index]=0;
                real_path_index=real_path_index+1;
                }*/
                        //(ny==1 || ny==2); nothing
                        //(my>2);   nothing
```

```
		}
		return 0;
}
//---------------------------------------
int distance(int my, int mx, int ny, int nx)
{
		int result=0;
		int result2=0;
		if(my<=2 && ny<=2)
		{
			if(my!=0 && ny!=0 && mx!= nx)
			{
				result=abs(0-my)+abs(0-ny)+7;//4
				result2=abs(3-my)+abs(3-ny)+7;
				if(result2<result)
				{
					result=result2;
				}
			}
			else
			{
				result=abs(my-ny)+abs(mx-nx);//1
			}
		}
		else if(my>2 && ny>2)
		{
			result=abs(my-ny)+abs(mx-nx);//2
		}
		else
		{
			result=abs(my-3)+abs(mx-0)+abs(ny-3)+abs(nx-0);//3
			result2=abs(my-3)+abs(mx-6)+abs(ny-3)+abs(nx-6);
			if(result2<result)
			{
				result=result2;
			}
		}
```



```
		return result;
}
<3-2> 작성한 프로그램 소스 /1단계,3단계--------------------------------------
int step_1();
int step_3();
int untilblackline();
int untilblackline_2();
int between_blackline();
int test_S1();
int test_S3();
int test_S4();
int right_turn();
int left_turn();
int go_center();
int go_straight();
int go_right();
int go_left();
int store_information[5][7];
int go_back();
int real_path_Y[5]={0,3,3,4,4};//----
int real_path_X[5]={3,6,3,2,6};//----
int real_path_drop[5]={0,0,0,0,0};//----
int real_path_index=0;//----
int hacha_set=0;
int now_Y=0;//----
int now_X=0;//----
int now_direction_Y=0;//----
int now_direction_X=1;//----
int cha_Y=0;//----
int cha_X=0;//----
int left_turn_direction();
int right_turn_direction();
int speed_backward=-20;
int speed_forward=20;
int speed_turn=20;
int speed_jase=5;
task main(){
```

```
step_1();
//step_3();
}
// step function ++++++++++++++++++++++
int step_1()
{
untilblackline();
while(getColorName(S4)==colorBlack)
{
        setMotorSpeed(motorB,speed_forward);
        setMotorSpeed(motorC,speed_forward);
}
go_center();
store_information[4][0]=getColorName(S4);
go_left();
store_information[3][0]=getColorName(S4);
go_straight();
store_information[2][0]=getColorName(S4);
go_straight();
store_information[1][0]=getColorName(S4);
go_straight();
store_information[0][0]=getColorName(S4);
go_right();
store_information[0][1]=getColorName(S4);
go_straight();
store_information[0][2]=getColorName(S4);
go_straight();
store_information[0][3]=getColorName(S4);
go_straight();
store_information[0][4]=getColorName(S4);
go_straight();
store_information[0][5]=getColorName(S4);
go_straight();
store_information[0][6]=getColorName(S4);
go_right();
store_information[1][6]=getColorName(S4);
go_straight();
```



```
store_information[2][6]=getColorName(S4);
go_straight();
store_information[3][6]=getColorName(S4);
go_right();
store_information[3][5]=getColorName(S4);
go_straight();
store_information[3][4]=getColorName(S4);
go_straight();
store_information[3][3]=getColorName(S4);
go_straight();
store_information[3][2]=getColorName(S4);
go_straight();
store_information[3][1]=getColorName(S4);
go_left();
store_information[4][1]=getColorName(S4);
go_left();
store_information[4][2]=getColorName(S4);
go_straight();
store_information[4][3]=getColorName(S4);
go_straight();
store_information[4][4]=getColorName(S4);
go_straight();
store_information[4][5]=getColorName(S4);
go_straight();
store_information[4][6]=getColorName(S4);
go_back();
go_straight();
go_straight();
go_straight();
go_straight();
go_straight();
moveMotorTarget(motorB,800,speed_forward);
moveMotorTarget(motorC,800,speed_forward);
waitUntilMotorStop(motorC);
return  0;
}
//----------------------------------------
```

```
int  step_3()
{
   untilblackline();
   go_center();
   now_Y=0;
              now_X=0;
              now_direction_Y=0;
              now_direction_X=1;
              real_path_index=0;
              //start          ------------------------------------------------------------------------
              while(now_X!=6||now_Y!=4)
              {
              cha_X=  real_path_X[real_path_index]-now_X;
              cha_Y=  real_path_Y[real_path_index]-now_Y;
              while(cha_X!=0)
                    {
                          while(now_direction_X!=cha_X/abs(cha_X))
                          {
                                untilblackline_2();
                                left_turn();
                                left_turn_direction();
                          }
                          go_straight();
                          now_X=now_X+now_direction_X;
                          now_Y=now_Y+now_direction_Y;
                          cha_X=real_path_X[real_path_index]-now_X;
                    cha_Y=real_path_Y[real_path_index]-now_Y;
                          //hacha_set==1  ?
               if(hacha_set==1)
               {
                                          //medium  motor
          hacha_set=0;
               }
              }
                    while(cha_Y!=0)
                    {
                          while(now_direction_Y!=cha_Y/abs(cha_Y))
```



```
                          {
                                untilblackline_2();
                                right_turn();
                                right_turn_direction();
                          }
                          go_straight();
                          now_X=now_X+now_direction_X;
                          now_Y=now_Y+now_direction_Y;
                          cha_X=real_path_X[real_path_index]-now_X;
                          cha_Y=real_path_Y[real_path_index]-now_Y;
                          //hacha_set==1  ?
                    if(hacha_set==1)
                   {
                                          //medium  motor
          hacha_set=0;
                    }
                   }
                   //real_path_drop[real_path_index]==1  ?
                   if(real_path_drop[real_path_index]==1)
                   {
          hacha_set=1;
                   }
                   real_path_index=real_path_index+1;
  }
             //end----------------------------------------------------------------------------
moveMotorTarget(motorB,1000,speed_forward);
moveMotorTarget(motorC,1000,speed_forward);
waitUntilMotorStop(motorC);
             return  0;
}
int  left_turn_direction()
{
             if(now_direction_X==0&&now_direction_Y==+1)
             {
                   now_direction_X=+1;
                   now_direction_Y=0;
             }
```

```
        else if(now_direction_X==+1&&        now_direction_Y==0)
        {
            now_direction_X=0;
            now_direction_Y=-1;
        }
        else if(        now_direction_X==0&&now_direction_Y==-1)
        {
            now_direction_X=-1;
    now_direction_Y=0;
        }
        else if(now_direction_X==-1&& now_direction_Y==0)
        {
            now_direction_X=0;
            now_direction_Y=+1;
        }
        return 0;
}
int right_turn_direction()
{
        if(now_direction_X==0&&now_direction_Y==+1)
        {
            now_direction_X=-1;
            now_direction_Y=0;
        }
        else if(now_direction_X==-1&&now_direction_Y==0)
        {
            now_direction_X=0;
            now_direction_Y=-1;
        }
        else if(now_direction_X==0&&now_direction_Y==-1)
        {
            now_direction_X=+1;
    now_direction_Y=0;
        }
        else if(now_direction_X==+1&& now_direction_Y==0)
        {
            now_direction_X=0;
```



```
            now_direction_Y=+1;
        }
        return 0;
}
// function defenition =================
int untilblackline()
{
        while(getColorName(S4)!= colorBlack)
        {
            setMotorSpeed(motorB,speed_forward);
            setMotorSpeed(motorC,speed_forward);
        }
        return(0);
}
//0000000000000000000000000000000000000000000000000
int untilblackline_2()
{
        moveMotorTarget(motorB,120,speed_forward);
        moveMotorTarget(motorC,120,speed_forward);
        waitUntilMotorStop(motorC);
        if(test_S4()&&test_S1())
            {
                moveMotorTarget(motorB,40,-speed_forward);
                moveMotorTarget(motorC,40,-speed_forward);
                waitUntilMotorStop(motorC);
                while(getColorName(S4)!=colorBlack)
                {
                    setMotorSpeed(motorB,speed_turn);
                    setMotorSpeed(motorC,-speed_turn);
                }
                setMotorSpeed(motorB,0);
                setMotorSpeed(motorC,0);
            }
        else if(test_S4()&&test_S3())
        {
                moveMotorTarget(motorB,40,-speed_forward);
                moveMotorTarget(motorC,40,-speed_forward);
```

```
                waitUntilMotorStop(motorC);
                while(getColorName(S4)!=colorBlack)
                {
                        setMotorSpeed(motorB,-speed_turn);
                        setMotorSpeed(motorC,speed_turn);
                }
                setMotorSpeed(motorB,0);
                setMotorSpeed(motorC,0);
        }
                moveMotorTarget(motorB,50,-speed_turn);
                moveMotorTarget(motorC,50,speed_turn);
                waitUntilMotorStop(motorC);
                while(getColorName(S4)!=colorBlack)
                {
                        setMotorSpeed(motorB,speed_turn);
                        setMotorSpeed(motorC,-speed_turn);
                }
                setMotorSpeed(motorB,0);
                setMotorSpeed(motorC,0);
    return(0);
}
int between_blackline()
{
    moveMotorTarget(motorB, 140,speed_forward);
        moveMotorTarget(motorC, 140,speed_forward);
        waitUntilMotorStop(motorC);
        if(getColorName(S4)!=colorBlack)
        {
                moveMotorTarget(motorB,110,-speed_turn);
                moveMotorTarget(motorC,110,speed_turn);
                waitUntilMotorStop(motorC);
                while(getColorName(S4)!=colorBlack)
                {
                        setMotorSpeed(motorB,speed_turn);
                        setMotorSpeed(motorC,-speed_turn);
                }
                setMotorSpeed(motorB,0);
```



```
                        setMotorSpeed(motorC,0);
                }
    return 0;
}
//1111111111111111111111111
int test_S1()
{
        if(getColorName(S1)!= colorYellow && getColorName(S1)!= colorBlue &&
    getColorName(S1)!= colorGreen
&& getColorName(S1)!= colorRed)
        {
                return 1;
        }
        else
        {
                return 0;
        }
}
//22222222222222222222222
int test_S3()
{
        if(getColorName(S3)!= colorGreen && getColorName(S3)!= colorYellow &&
    getColorName(S3)!= colorBlue && getColorName(S3)!= colorRed)
        {
                return 1;
        }
        else
        {
                return 0;
        }
}
int test_S4()
{
        if(getColorName(S4)!= colorGreen && getColorName(S4)!= colorYellow &&
    getColorName(S4)!= colorBlue &&
                        getColorName(S4)!= colorRed && getColorName(S4)!=colorBlack &&
    getColorName(S4)!=colorBrown)
         {
```

```
                return 1;
        }
        else
        {
                return 0;
        }
}
//33333333333333333333333333
int right_turn()
        {
        /*      while(getColorName(S4)== colorBlack)
        {
                setMotorSpeed(motorB,speed_forward);
                setMotorSpeed(motorC,speed_forward);
        }*/
                while(test_S4()==0)
        {
                setMotorSpeed(motorB,speed_forward);
                setMotorSpeed(motorC,speed_forward);
        }
                while(getColorName(S4)!= colorBlack)
        {
                setMotorSpeed(motorB,speed_backward);
                setMotorSpeed(motorC,speed_backward);
        }
        while(test_S1()&&test_S3())
        {
                setMotorSpeed(motorB,speed_backward);
                setMotorSpeed(motorC,speed_backward);
        }
        while(getColorName(S4)== colorBlack)
        {
                setMotorSpeed(motorB,speed_turn);
                setMotorSpeed(motorC,-speed_turn);
        }
        while(getColorName(S4)!= colorBlack)
        {
```



```
                setMotorSpeed(motorB,speed_turn);
                setMotorSpeed(motorC,-speed_turn);
        }
        return 0;
        }
//44444444444444444444444444444
int left_turn()
        {
                /*while(getColorName(S4)== colorBlack)
        {
                setMotorSpeed(motorB,speed_forward);
                setMotorSpeed(motorC,speed_forward);
        }*/
                while(test_S4()==0)
        {
                setMotorSpeed(motorB,speed_forward);
                setMotorSpeed(motorC,speed_forward);
        }
                while(getColorName(S4)!= colorBlack)
        {
                setMotorSpeed(motorB,speed_backward);
                setMotorSpeed(motorC,speed_backward);
        }
        while(test_S1()&&test_S3())
        {
                setMotorSpeed(motorB,speed_backward);
                setMotorSpeed(motorC,speed_backward);
        }
        while(getColorName(S4)== colorBlack)
        {
                setMotorSpeed(motorB,-speed_turn);
                setMotorSpeed(motorC,speed_turn);
        }
        while(getColorName(S4)!= colorBlack)
        {
                setMotorSpeed(motorB,-speed_turn);
                setMotorSpeed(motorC,speed_turn);
```

```
}
 return 0;
}
//55555555555555555555555555555555
int go_center()
{
	while(1)
	{
		while(test_S1() || test_S3())
		{
			if(getColorName(S1)== colorBlack || test_S1()==0)
			{
				setMotorSpeed(motorB,0);
				setMotorSpeed(motorC,speed_jase);
			}
			else if(test_S3()==0 || getColorName(S3)== colorBlack)
			{
				setMotorSpeed(motorB,speed_jase);
				setMotorSpeed(motorC,0);
			}
			else
			{
				setMotorSpeed(motorB,speed_forward);
				setMotorSpeed(motorC,speed_forward);
			}
		}
		moveMotorTarget(motorB,15,speed_forward);
		moveMotorTarget(motorC,15,speed_forward);
		waitUntilMotorStop(motorC);
		if(test_S1()==0&& test_S3()==0)
		{
			moveMotorTarget(motorB,15,-speed_forward);
			moveMotorTarget(motorC,15,-speed_forward);
			waitUntilMotorStop(motorC);
			break;
		}
		else
```



```
		{
			while(getColorName(S1)== colorBlack)
			{
				setMotorSpeed(motorB,0);
				setMotorSpeed(motorC,speed_jase);
			}
			while(getColorName(S3)== colorBlack)
			{
				setMotorSpeed(motorB,speed_jase);
				setMotorSpeed(motorC,0);
			}
		}
	}
	setMotorSpeed(motorB,0);
	setMotorSpeed(motorC,0);
	playTone(784, 15);
	sleep(500);
return 0;
}
//66666666666666666666666666666666
int go_straight()
{
	untilblackline_2();
/*while(getColorName(S4)== colorBlack)
{
	setMotorSpeed(motorB,speed_forward);
	setMotorSpeed(motorC,speed_forward);
}*/
while(test_S4()==0)
{
	setMotorSpeed(motorB,speed_forward);
	setMotorSpeed(motorC,speed_forward);
}
while(getColorName(S1)== colorBlack)
{
	setMotorSpeed(motorB,0);
	setMotorSpeed(motorC,speed_jase);
```

```
        }
        while(getColorName(S3)== colorBlack)
        {
                setMotorSpeed(motorB,speed_jase);
                setMotorSpeed(motorC,0);
        }
        between_blackline();
         go_center();
         return 0;
        }
        //77777777777777777777777777777777777777
        int go_right()
        {
                untilblackline_2();
         right_turn();
         while(getColorName(S4)== colorBlack)
        {
                setMotorSpeed(motorB,speed_forward);
                setMotorSpeed(motorC,speed_forward);
        }
        while(getColorName(S1)== colorBlack)
        {
                setMotorSpeed(motorB,0);
                setMotorSpeed(motorC,speed_jase);
        }
        while(getColorName(S3)== colorBlack)
        {
                setMotorSpeed(motorB,speed_jase);
                setMotorSpeed(motorC,0);
        }
between_blackline();
        go_center();
         return 0;
        }
        //88888888888888888888888888888888
        int go_left()
        {
```



```
                untilblackline_2();
                left_turn();
                while(getColorName(S4)== colorBlack)
                {
                setMotorSpeed(motorB,speed_forward);
                setMotorSpeed(motorC,speed_forward);
                }
        while(getColorName(S1)== colorBlack)
        {
                setMotorSpeed(motorB,0);
                setMotorSpeed(motorC,speed_jase);
        }
        while(getColorName(S3)== colorBlack)
        {
                setMotorSpeed(motorB,speed_jase);
                setMotorSpeed(motorC,0);
        }
         between_blackline();
        go_center();
         return 0;
        }
        //99999999999999999999999
        int go_back()
        {
                        untilblackline_2();
/*      while(getColorName(S4)== colorBlack)
        {
                setMotorSpeed(motorB,speed_forward);
                setMotorSpeed(motorC,speed_forward);
        }*/
                while(test_S4()==0)
        {
                setMotorSpeed(motorB,speed_forward);
                setMotorSpeed(motorC,speed_forward);
        }
        while(getColorName(S4)!=colorBlack)
        {
```

```
        setMotorSpeed(motorB,speed_backward);
        setMotorSpeed(motorC,speed_backward);
    }
    while(test_S1()&&test_S3())
    {
        setMotorSpeed(motorB,speed_backward);
        setMotorSpeed(motorC,speed_backward);
    }
    while(getColorName(S4)==colorBlack)
    {
        setMotorSpeed(motorB,-speed_turn);
        setMotorSpeed(motorC,speed_turn);
    }
    while(getColorName(S4)!=colorBlack)
    {
        setMotorSpeed(motorB,-speed_turn);
        setMotorSpeed(motorC,speed_turn);
    }
        while(getColorName(S4)==colorBlack)
    {
        setMotorSpeed(motorB,-speed_turn);
        setMotorSpeed(motorC,speed_turn);
    }
    while(getColorName(S4)!=colorBlack)
    {
        setMotorSpeed(motorB,-speed_turn);
        setMotorSpeed(motorC,speed_turn);
    }
     setMotorSpeed(motorB,0);
     setMotorSpeed(motorC,0);
        sleep(1000);
    while(getColorName(S4)==  colorBlack)
    {
        setMotorSpeed(motorB,speed_forward);
        setMotorSpeed(motorC,speed_forward);
    }
    while(getColorName(S1)==  colorBlack)
```



```
        {
            setMotorSpeed(motorB,0);
            setMotorSpeed(motorC,speed_jase);
        }
        while(getColorName(S3)==  colorBlack)
        {
            setMotorSpeed(motorB,speed_jase);
            setMotorSpeed(motorC,0);
        }
        between_blackline();
        go_center();
         return  0;
}
```

이번 프로젝트에 대한 반성 / 다음 프로젝트 계획

/ 다음 목표
승객 탐승하차 장치 개발하기
1단계 귀한 후 정보마당으로 이동하여 색상 선택 장치를 읽고 3단계로 연결하는 프로그램 개발하기
1, 2, 3 단계 개발 프로그램 연결하여 과제를 완주하기

지도교사 확인 및 의견

2단계 최단경로 구하기, 실제 주행 경로 구하기 시험 완료 확인 3단계 1,2 단계 없이 변수 설정으로 주행 시험 완료 확인 주행의 불안정성은 어느 정도 해결되었으나 다음 상점이로 이동시 이탈 문제 발생 가능성이 여전히 존재함.	(서명)

프로젝트3 (개발 완료)	
프로젝트 기간 및 목표	
기간	8월27일~9월 5일
목표	승객 탑승 및 하차 장치 설계와 제작, 2단계 주행 개발, 1,2,3단계 연결 및 개발 완료

프로젝트 활동내용

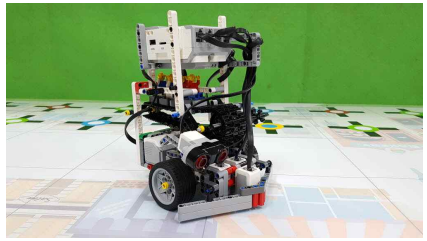
<1> 팀원의 역할

현태: 승객 탑승 하차 장치를 설계하여 제작하면서 하드웨어를 크게 수정했고 하차 승객이 걸리지 않도록 가드를 제작 했습니다. 색상 선택 장치에 접근할 수 있게 초음파센서를 달았고, 1,2,3단계 연결 주행을 완료하기 위해 코드와 주행을 살폈습니다.

신영: 1단계에서 2단계를 거쳐 3단계로 연결하기 위한 step_1_2 프로그램을 개발하고 1,2,3 단계를 연결하여 주행하도록 시험하고 주행과정에 따른 과제를 수행하도록 LED, 소리 등 추가하여 전체 과제를 완주하도록 했습니다.

규민: step_1_2 프로그램을 개발과정을 함께 하고 2단계와 1,3단계 주행코들 합하고 필요 없는 부분 삭제하고 정리한 후 주행코드 검토하고 전체 주행 분석을 하였고 전체적으로 잘 주행하도록 테스트 하였습니다.

<2> 사진



<3> 작성한 프로그램 소스----------

```
int step_2();
int store_information[5][7]=
{
        {9, 9, 9, 9, 9, 9, 9},
        {9,-1,-1,-1,-1,-1, 9},
        {9,-1,-1,-1,-1,-1, 9},
        {9, 9, 9, 9, 9, 9, 9},
```



```
        {9, 9, 9, 9, 9, 9, 9}
};
int visit_index;
int visit_X[4];
int visit_Y[4];
int read_color;
int distance(int my, int mx, int ny, int nx);
int all_path[24][4]=
{
        {0,1,2,3},{0,1,3,2},
        {0,2,1,3},{0,2,3,1},
        {0,3,1,2},{0,3,2,1},
        {1,0,2,3},{1,0,3,2},
        {1,2,0,3},{1,2,3,0},
        {1,3,0,2},{1,3,2,0},
        {2,0,1,3},{2,0,3,1},
        {2,1,0,3},{2,1,3,0},
        {2,3,0,1},{2,3,1,0},
        {3,0,1,2},{3,0,2,1},
        {3,1,0,2},{3,1,2,0},
        {3,2,0,1},{3,2,1,0}
};
int min_distance=100000;
int min_path[4];
int insert_path(int my, int mx, int ny, int nx);
int real_path_Y[15];
int real_path_X[15];
int real_path_drop[15];
int real_path_index;
//-------------------------------------
int step_1();
int step_1_2();
int step_3();
int untilblackline();
int untilblackline_2();
int between_blackline();
int test_S1();
```

```
int  test_S3();
int  test_S4();
int  right_turn();
int  left_turn();
int  go_center();
int  go_straight();
int  go_right();
int  go_left();
//int  store_information[5][7];
int  go_back();
//int  real_path_Y[5]={0,3,3,4,4};//----
//int  real_path_X[5]={3,6,3,2,6};//----
//int  real_path_drop[5]={0,0,0,0,0};//----
//int  real_path_index=0;//----
//int  read_color;
int  hacha_set;
int  now_Y;
int  now_X;
int  now_direction_Y;
int  now_direction_X;
int  cha_Y;
int  cha_X;
int  left_turn_direction();
int  right_turn_direction();
int  speed_backward=-20;
int  speed_forward=20;
int  speed_turn=20;
int  speed_jase=5;


task  main()
{
		step_1();
		step_1_2();
		//read_color=5;
		step_2();
		displayBigStringAt(50,110,"%d",0);
```



```
		displayBigStringAt(70,110,"%d",0);
		displayBigStringAt(50,90,"%d",visit_X[min_path[0]]);
		displayBigStringAt(70,90,"%d",visit_Y[min_path[0]]);
		displayBigStringAt(50,70,"%d",visit_X[min_path[1]]);
		displayBigStringAt(70,70,"%d",visit_Y[min_path[1]]);
		displayBigStringAt(50,50,"%d",visit_X[min_path[2]]);
		displayBigStringAt(70,50,"%d",visit_Y[min_path[2]]);
		displayBigStringAt(50,30,"%d",visit_X[min_path[3]]);
		displayBigStringAt(70,30,"%d",visit_Y[min_path[3]]);
		displayBigStringAt(50,10,"%d",6);
		displayBigStringAt(70,10,"%d",4);
		sleep(5000);
		step_3();
}
// step  function  ++++++++++++++++++++++
int  step_1()
{
		setLEDColor(ledOff);
		moveMotorTarget(motorB,200,speed_forward);
		moveMotorTarget(motorC,200,speed_forward);
		waitUntilMotorStop(motorC);

		go_center();
		store_information[4][0]=getColorName(S4);
		go_left();
		store_information[3][0]=getColorName(S4);
		go_straight();
		store_information[2][0]=getColorName(S4);
		go_straight();
		store_information[1][0]=getColorName(S4);
		go_straight();
		store_information[0][0]=getColorName(S4);
		go_right();
		store_information[0][1]=getColorName(S4);
		go_straight();
		store_information[0][2]=getColorName(S4);
		go_straight();
```

```
	store_information[0][3]=getColorName(S4);
	go_straight();
	store_information[0][4]=getColorName(S4);
	go_straight();
	store_information[0][5]=getColorName(S4);
	go_straight();
	store_information[0][6]=getColorName(S4);
	go_right();
	store_information[1][6]=getColorName(S4);
	go_straight();
	store_information[2][6]=getColorName(S4);
	go_straight();
	store_information[3][6]=getColorName(S4);
	go_right();
	store_information[3][5]=getColorName(S4);
	go_straight();
	store_information[3][4]=getColorName(S4);
	go_straight();
	store_information[3][3]=getColorName(S4);
	go_straight();
	store_information[3][2]=getColorName(S4);
	go_straight();
	store_information[3][1]=getColorName(S4);
	go_left();
	store_information[4][1]=getColorName(S4);
	go_left();
	store_information[4][2]=getColorName(S4);
	go_straight();
	store_information[4][3]=getColorName(S4);
	go_straight();
	store_information[4][4]=getColorName(S4);
	go_straight();
	store_information[4][5]=getColorName(S4);
	go_straight();
	store_information[4][6]=getColorName(S4);
	go_back();
	go_straight();
```



```
	go_straight();
	go_straight();
	go_straight();
	go_straight();

	untilblackline_2();
	while(getColorName(S4)==colorBlack)
	{
		setMotorSpeed(motorB,speed_forward);
		setMotorSpeed(motorC,speed_forward);
	}
	moveMotorTarget(motorB,240,speed_forward);
	moveMotorTarget(motorC,240,speed_forward);
	waitUntilMotorStop(motorC);
	setMotorSpeed(motorB,0);
	setMotorSpeed(motorC,0);
	setLEDColor(ledRed);
	playTone(784, 300);
	sleep(3000);

	return 0;
}
//sstteepp__11__22
int step_1_2()
{
	while(getColorName(S4)==colorWhite)
	{
		setMotorSpeed(motorB,-speed_turn);
		setMotorSpeed(motorC,speed_turn);
	}
	untilblackline();
	go_center();
	go_left();
	go_straight();
	go_straight();
	go_straight();
	untilblackline_2();
```

```
	//left_turn();

	while(test_S4()==0)
	{
		setMotorSpeed(motorB,speed_forward);
		setMotorSpeed(motorC,speed_forward);
	}
	while(getColorName(S4)!= colorBlack)
	{
		setMotorSpeed(motorB,speed_backward);
		setMotorSpeed(motorC,speed_backward);
	}
	while(test_S1()&&test_S3())
	{
		setMotorSpeed(motorB,speed_backward);
		setMotorSpeed(motorC,speed_backward);
	}
	//teacher-----for safe start
	setMotorSpeed(motorB,0);
	setMotorSpeed(motorC,0);
	sleep(500);
	//teacher----for safe start
	while(getUSDistance(S2) > 50)
	{
		setMotorSpeed(motorB, -10);
		setMotorSpeed(motorC, 10);
	}
	//teacher-----for safe start
	setMotorSpeed(motorB,0);
	setMotorSpeed(motorC,0);
	sleep(500);
	//teacher----for safe start
	moveMotorTarget(motorB,35, -10);
	moveMotorTarget(motorC,35, 10);
	waitUntilMotorStop(motorC);
	//teacher-----for safe start
	setMotorSpeed(motorB,0);
```



```
	setMotorSpeed(motorC,0);
	sleep(500);
	//teacher----for safe start

	moveMotorTarget(motorB,425/(3.14*56)*360,speed_forward);
	moveMotorTarget(motorC,425/(3.14*56)*360,speed_forward);
	waitUntilMotorStop(motorC);

	// color read test----------
	read_color=5;

	if(getColorName(S4)==colorBlue || getColorName(S1)==colorBlue ||
getColorName(S3)==colorBlue)
	{
		read_color=2;
	}

	else if(getColorName(S4)==colorYellow || getColorName(S1)==colorYellow ||
getColorName(S3)==colorYellow)
	{
		read_color=4;
	}
	// color read test----------

	setMotorSpeed(motorB,0);
	setMotorSpeed(motorC,0);
	sleep(2000);

	moveMotorTarget(motorB,300/(3.14*56)*360,speed_backward);
	moveMotorTarget(motorC,300/(3.14*56)*360,speed_backward);
	waitUntilMotorStop(motorC);

	setMotorSpeed(motorB,0);
	setMotorSpeed(motorC,0);
	sleep(500);

	while(getColorName(S4)!=colorBlack)
```

```
	{
		setMotorSpeed(motorB,speed_turn);
		setMotorSpeed(motorC,-speed_turn);
	}
	//teacher-----for turn stop
	setMotorSpeed(motorB,0);
	setMotorSpeed(motorC,0);
	//sleep(500);
	//teacher----for turn stop
	return 0;
}

//-------------------------------------------
int step_3()
{
	untilblackline();
	go_center();
	hacha_set=0;
	now_Y=0;
	now_X=0;
	now_direction_Y=0;
	now_direction_X=1;
	cha_Y=0;
	cha_X=0;

	real_path_index=0;

	//start		-------------------------------------------------------------------
	while(now_X!=6||now_Y!=4)
	{
		cha_X=  real_path_X[real_path_index]-now_X;
		cha_Y=  real_path_Y[real_path_index]-now_Y;

		while(cha_X!=0)
		{
			while(now_direction_X!=cha_X/abs(cha_X))
			{
```



```
				untilblackline_2();
				left_turn();
				left_turn_direction();
			}
			go_straight();
			now_X=now_X+now_direction_X;
			now_Y=now_Y+now_direction_Y;
			cha_X=real_path_X[real_path_index]-now_X;
			cha_Y=real_path_Y[real_path_index]-now_Y;
			if(hacha_set==1)
			{
				moveMotorTarget(motorA,83,20);
				waitUntilMotorStop(motorA);
				setMotorSpeed(motorA,0);
				hacha_set=0;
			}
		}
		while(cha_Y!=0)
		{
			while(now_direction_Y!=cha_Y/abs(cha_Y))
			{
				untilblackline_2();
				right_turn();
				right_turn_direction();
			}
			go_straight();
			now_X=now_X+now_direction_X;
			now_Y=now_Y+now_direction_Y;
			cha_X=real_path_X[real_path_index]-now_X;
			cha_Y=real_path_Y[real_path_index]-now_Y;
			if(hacha_set==1)
			{
				moveMotorTarget(motorA,83,20);
				waitUntilMotorStop(motorA);
				setMotorSpeed(motorA,0);
				hacha_set=0;
			}
```

```
                }

                if(real_path_drop[real_path_index]==1)
                {

                        hacha_set=1;

                }

                real_path_index=real_path_index+1;

        }
        //end---------------------------------------------------------------------------

        if(now_direction_X==0&&  now_direction_Y==1)
        {
                untilblackline_2();
                left_turn();
        }
        moveMotorTarget(motorB,800,speed_forward);
        moveMotorTarget(motorC,800,speed_forward);
        waitUntilMotorStop(motorC);

        return  0;
}

int  left_turn_direction()
{
        if(now_direction_X==0&&now_direction_Y==+1)
        {
                now_direction_X=+1;
                now_direction_Y=0;
        }
        else  if(now_direction_X==+1&&          now_direction_Y==0)
        {
                now_direction_X=0;
                now_direction_Y=-1;
```



```
        }
        else  if(now_direction_X==0&&now_direction_Y==-1)
        {
                now_direction_X=-1;
                now_direction_Y=0;
        }
        else  if(now_direction_X==-1&&  now_direction_Y==0)
        {
                now_direction_X=0;
                now_direction_Y=+1;
        }

        return  0;
}


int  right_turn_direction()
{
        if(now_direction_X==0&&now_direction_Y==+1)
        {
                now_direction_X=-1;
                now_direction_Y=0;
        }
        else  if(now_direction_X==-1&&now_direction_Y==0)
        {
                now_direction_X=0;
                now_direction_Y=-1;
        }
        else  if(now_direction_X==0&&now_direction_Y==-1)
        {
                now_direction_X=+1;
                now_direction_Y=0;
        }
        else  if(now_direction_X==+1&&  now_direction_Y==0)
        {
                now_direction_X=0;
                now_direction_Y=+1;
```

```
        }
        return 0;
}

// function defenition ================

int untilblackline()
{

        while(getColorName(S4)!= colorBlack)
        {
                setMotorSpeed(motorB,speed_forward);
                setMotorSpeed(motorC,speed_forward);
        }

        return(0);
}

//0000000000000000000000000000000000000000000000000
int untilblackline_2()
{
        if(getColorName(S4)!=colorBlack)
        {
                moveMotorTarget(motorB,120,speed_forward);
                moveMotorTarget(motorC,120,speed_forward);
                waitUntilMotorStop(motorC);

                if(test_S4()&&test_S1())
                {
                        moveMotorTarget(motorB,40,-speed_forward);
                        moveMotorTarget(motorC,40,-speed_forward);
                        waitUntilMotorStop(motorC);

                        while(getColorName(S4)!=colorBlack)
                        {
                                setMotorSpeed(motorB,speed_turn);
                                setMotorSpeed(motorC,-speed_turn);
```



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

                else if(test_S4()&&test_S3())
                {
                        moveMotorTarget(motorB,40,-speed_forward);
                        moveMotorTarget(motorC,40,-speed_forward);
                        waitUntilMotorStop(motorC);

                        while(getColorName(S4)!=colorBlack)
                        {
                                setMotorSpeed(motorB,-speed_turn);
                                setMotorSpeed(motorC,speed_turn);
                        }

                        setMotorSpeed(motorB,0);
                        setMotorSpeed(motorC,0);
                }
                moveMotorTarget(motorB,50,-speed_turn);
                moveMotorTarget(motorC,50,speed_turn);
                waitUntilMotorStop(motorC);

                while(getColorName(S4)!=colorBlack)
                {
                        setMotorSpeed(motorB,speed_turn);
                        setMotorSpeed(motorC,-speed_turn);
                }
                setMotorSpeed(motorB,0);
                setMotorSpeed(motorC,0);
        }
        return(0);
}

int between_blackline()
{
```

```
		moveMotorTarget(motorB, 145,speed_forward);
		moveMotorTarget(motorC, 145,speed_forward);
		waitUntilMotorStop(motorC);

		if(getColorName(S4)!=colorBlack)
		{
			moveMotorTarget(motorB,110,-speed_turn);
			moveMotorTarget(motorC,110,speed_turn);
			waitUntilMotorStop(motorC);

			while(getColorName(S4)!=colorBlack)
			{
				setMotorSpeed(motorB,speed_turn);
				setMotorSpeed(motorC,-speed_turn);
			}
			setMotorSpeed(motorB,0);
			setMotorSpeed(motorC,0);
		}

		return 0;
}

//11111111111111111111111
int test_S1()
{
		if(getColorName(S1)!= colorYellow && getColorName(S1)!= colorBlue &&
	getColorName(S1)!= colorGreen
			&& getColorName(S1)!= colorRed)
		{
			return 1;
		}
		else
		{
			return 0;
		}
}
```



```
//222222222222222222222222
int test_S3()
{
		if(getColorName(S3)!= colorGreen && getColorName(S3)!= colorYellow &&
	getColorName(S3)!= colorBlue && getColorName(S3)!= colorRed)
		{
			return 1;
		}
		else
		{
			return 0;
		}
}

int test_S4()
{
		if(getColorName(S4)!= colorGreen && getColorName(S4)!= colorYellow &&
	getColorName(S4)!= colorBlue &&
			getColorName(S4)!= colorRed && getColorName(S4)!=colorBlack &&
	getColorName(S4)!=colorBrown)
		{

			return 1;
		}
		else
		{
			return 0;
		}
}

//33333333333333333333333333
int right_turn()
{
		while(test_S4()==0)
		{
			setMotorSpeed(motorB,speed_forward);
			setMotorSpeed(motorC,speed_forward);
		}
```

```
		while(getColorName(S4)!= colorBlack)
		{
			setMotorSpeed(motorB,speed_backward);
			setMotorSpeed(motorC,speed_backward);
		}

		while(test_S1()&&test_S3())
		{
			setMotorSpeed(motorB,speed_backward);
			setMotorSpeed(motorC,speed_backward);
		}
		setMotorSpeed(motorB,0);
		setMotorSpeed(motorC,0);
		sleep(500);
		while(getColorName(S4)== colorBlack)
		{
			setMotorSpeed(motorB,speed_turn);
			setMotorSpeed(motorC,-speed_turn);
		}
		while(getColorName(S4)!= colorBlack)
		{
			setMotorSpeed(motorB,speed_turn);
			setMotorSpeed(motorC,-speed_turn);
		}


		return 0;
}

//4444444444444444444444444444444
int left_turn()
{
		while(test_S4()==0)
		{
			setMotorSpeed(motorB,speed_forward);
			setMotorSpeed(motorC,speed_forward);
		}
```



```
		while(getColorName(S4)!= colorBlack)
		{
			setMotorSpeed(motorB,speed_backward);
			setMotorSpeed(motorC,speed_backward);
		}

		while(test_S1()&&test_S3())
		{
			setMotorSpeed(motorB,speed_backward);
			setMotorSpeed(motorC,speed_backward);
		}
  setMotorSpeed(motorB,0);
		setMotorSpeed(motorC,0);
		sleep(500);
		while(getColorName(S4)== colorBlack)
		{
			setMotorSpeed(motorB,-speed_turn);
			setMotorSpeed(motorC,speed_turn);
		}
		while(getColorName(S4)!= colorBlack)
		{
			setMotorSpeed(motorB,-speed_turn);
			setMotorSpeed(motorC,speed_turn);
		}

		return 0;
}

//5555555555555555555555555555555
int go_center()
{
		while(1)
		{
			while(test_S1() || test_S3())
			{
				if(getColorName(S1)== colorBlack || test_S1()==0 || getColorName(S1)==
   colorBrown)
```

```
                {
                        setMotorSpeed(motorB,0);
                        setMotorSpeed(motorC,speed_jase);
                }

                else    if(test_S3()==0     ||     getColorName(S3)==     colorBlack     ||
getColorName(S3)== colorBrown)
                {
                        setMotorSpeed(motorB,speed_jase);
                        setMotorSpeed(motorC,0);
                }

                else
                {
                        setMotorSpeed(motorB,speed_forward);
                        setMotorSpeed(motorC,speed_forward);
                }

        }

        moveMotorTarget(motorB,15,speed_forward);
        moveMotorTarget(motorC,15,speed_forward);
        waitUntilMotorStop(motorC);

        if(test_S1()==0&& test_S3()==0)
        {
                moveMotorTarget(motorB,15,-speed_forward);
                moveMotorTarget(motorC,15,-speed_forward);
                waitUntilMotorStop(motorC);
                break;
        }
        else
        {
                while(getColorName(S4)== colorBlack)
                {
                        setMotorSpeed(motorB,0);
                        setMotorSpeed(motorC,speed_jase);//--------
```



```
                }

                while(getColorName(S3)== colorBlack)
                {
                        setMotorSpeed(motorB,speed_jase);
                        setMotorSpeed(motorC,0);
                }
            }
        }

        setMotorSpeed(motorB,0);
        setMotorSpeed(motorC,0);
        playTone(784, 15);
        sleep(500);


        return 0;
}

//6666666666666666666666666666666
int go_straight()
{
        untilblackline_2();

        while(test_S4()==0)
        {
            setMotorSpeed(motorB,speed_forward);
            setMotorSpeed(motorC,speed_forward);
        }

        while(getColorName(S1)== colorBlack)
        {
            setMotorSpeed(motorB,0);
            setMotorSpeed(motorC,speed_jase);
        }

        while(getColorName(S3)== colorBlack)
```

```
        {
            setMotorSpeed(motorB,speed_jase);
            setMotorSpeed(motorC,0);
        }

        between_blackline();

        go_center();

        return 0;
}

//7777777777777777777777777777777777777777
int go_right()
{
        untilblackline_2();

        right_turn();

        while(getColorName(S4)== colorBlack)
        {
            setMotorSpeed(motorB,speed_forward);
            setMotorSpeed(motorC,speed_forward);
        }
        while(getColorName(S1)== colorBlack)
        {
            setMotorSpeed(motorB,0);
            setMotorSpeed(motorC,speed_jase);
        }
        while(getColorName(S3)== colorBlack)
        {
            setMotorSpeed(motorB,speed_jase);
            setMotorSpeed(motorC,0);
        }
        between_blackline();

        go_center();
```



```
        return 0;
}


//88888888888888888888888888888888
int go_left()
{
        untilblackline_2();

        left_turn();

        while(getColorName(S4)== colorBlack)
        {
            setMotorSpeed(motorB,speed_forward);
            setMotorSpeed(motorC,speed_forward);
        }

        while(getColorName(S1)== colorBlack)
        {
            setMotorSpeed(motorB,0);
            setMotorSpeed(motorC,speed_jase);
        }
        while(getColorName(S3)== colorBlack)
        {
            setMotorSpeed(motorB,speed_jase);
            setMotorSpeed(motorC,0);
        }

        between_blackline();

        go_center();

        return 0;
}

//99999999999999999999999
```

```
int  go_back()
{
        untilblackline_2();

        while(test_S4()==0)
        {
                setMotorSpeed(motorB,speed_forward);
                setMotorSpeed(motorC,speed_forward);
        }
        while(getColorName(S4)!=colorBlack)
        {
                setMotorSpeed(motorB,speed_backward);
                setMotorSpeed(motorC,speed_backward);
        }

        while(test_S1()&&test_S3())
        {
                setMotorSpeed(motorB,speed_backward);
                setMotorSpeed(motorC,speed_backward);
        }
        setMotorSpeed(motorB,0);
        setMotorSpeed(motorC,0);
        sleep(500);
        while(getColorName(S4)==colorBlack)
        {
                setMotorSpeed(motorB,-speed_turn);
                setMotorSpeed(motorC,speed_turn);
        }
        while(getColorName(S4)!=colorBlack)
        {
                setMotorSpeed(motorB,-speed_turn);
                setMotorSpeed(motorC,speed_turn);
        }
        while(getColorName(S4)==colorBlack)
        {
                setMotorSpeed(motorB,-speed_turn);
                setMotorSpeed(motorC,speed_turn);
```



```
        }
        while(getColorName(S4)!=colorBlack)
        {
                setMotorSpeed(motorB,-speed_turn);
                setMotorSpeed(motorC,speed_turn);
        }

        while(getColorName(S4)==  colorBlack)
        {
                setMotorSpeed(motorB,speed_forward);
                setMotorSpeed(motorC,speed_forward);
        }

        while(getColorName(S1)==  colorBlack)
        {
                setMotorSpeed(motorB,0);
                setMotorSpeed(motorC,speed_jase);
        }
        while(getColorName(S3)==  colorBlack)
        {
                setMotorSpeed(motorB,speed_jase);
                setMotorSpeed(motorC,0);
        }

        between_blackline();

        go_center();

        return  0;
}
//-----------------------------------------------
int  step_2()
{

        visit_index=0;

        for(int  i=0;  i<5;i=i+1)
```

```
        {
            for(int j=0; j<7;j=j+1)
            {
                if(read_color==store_information[i][j])
                {
                    visit_X[visit_index]=j;
                    visit_Y[visit_index]=i;
                    visit_index=visit_index+1;
                }
            }
        }

        int distance_sun=0;
        for(int i=0; i<24; i=i+1)
        {
            distance_sun=
            distance(0,0,visit_Y[all_path[i][0]],visit_X[all_path[i][0]])

+distance(visit_Y[all_path[i][0]],visit_X[all_path[i][0]],visit_Y[all_path[i][1]],visit_X[all_path[i][1]])

+distance(visit_Y[all_path[i][1]],visit_X[all_path[i][1]],visit_Y[all_path[i][2]],visit_X[all_path[i][2]])

+distance(visit_Y[all_path[i][2]],visit_X[all_path[i][2]],visit_Y[all_path[i][3]],visit_X[all_path[i][3]])
            +distance(visit_Y[all_path[i][3]],visit_X[all_path[i][3]],4,6);
            if(min_distance>distance_sun)
            {
                min_distance=distance_sun;
                min_path[0]=all_path[i][0];
                min_path[1]=all_path[i][1];
                min_path[2]=all_path[i][2];
                min_path[3]=all_path[i][3];
            }
        }
        real_path_index=0;

        insert_path(0,0,visit_Y[min_path[0]],visit_X[min_path[0]]);

        real_path_Y[real_path_index]=visit_Y[min_path[0]];
```



```
        real_path_X[real_path_index]=visit_X[min_path[0]];
        real_path_drop[real_path_index]=1;
        real_path_index=real_path_index+1;


    insert_path(visit_Y[min_path[0]],visit_X[min_path[0]],visit_Y[min_path[1]],visit_X[min_path[1]]);

        real_path_Y[real_path_index]=visit_Y[min_path[1]];
        real_path_X[real_path_index]=visit_X[min_path[1]];
        real_path_drop[real_path_index]=1;
        real_path_index=real_path_index+1;


    insert_path(visit_Y[min_path[1]],visit_X[min_path[1]],visit_Y[min_path[2]],visit_X[min_path[2]]);

        real_path_Y[real_path_index]=visit_Y[min_path[2]];
        real_path_X[real_path_index]=visit_X[min_path[2]];
        real_path_drop[real_path_index]=1;
        real_path_index=real_path_index+1;


    insert_path(visit_Y[min_path[2]],visit_X[min_path[2]],visit_Y[min_path[3]],visit_X[min_path[3]]);

        real_path_Y[real_path_index]=visit_Y[min_path[3]];
        real_path_X[real_path_index]=visit_X[min_path[3]];
        real_path_drop[real_path_index]=1;
        real_path_index=real_path_index+1;

        insert_path(visit_Y[min_path[3]],visit_X[min_path[3]],4,6);

        real_path_Y[real_path_index]=4;
        real_path_X[real_path_index]=6;
        real_path_drop[real_path_index]=1;
        real_path_index=real_path_index+1;


        return 0;
}
```

```
//============================
int insert_path(int my, int mx, int ny, int nx)
{
	int result=0;
	int result2=0;

	if(my==0)//uper
	{
		//(ny<=2) nothing
		if(ny>2)
		{
			result=abs(my-3)+abs(mx-0)+abs(ny-3)+abs(nx-0);
			result2=abs(my-3)+abs(mx-6)+abs(ny-3)+abs(nx-6);
			if(result2<result)
			{
				//insert path 36
				real_path_Y[real_path_index]=3;
				real_path_X[real_path_index]=6;
				real_path_drop[real_path_index]=0;
				real_path_index=real_path_index+1;
			}
			else
			{
				//insert path 30
				real_path_Y[real_path_index]=3;
				real_path_X[real_path_index]=0;
				real_path_drop[real_path_index]=0;
				real_path_index=real_path_index+1;
			}
		}
	}

	else if(my==1||my==2)//middle
	{
		if(ny==0)
		{
			if(mx==0)
```



```
			{
				//0mx
				real_path_Y[real_path_index]=0;
				real_path_X[real_path_index]=0;
				real_path_drop[real_path_index]=0;
				real_path_index=real_path_index+1;
			}
			else if(mx==6)
			{
				real_path_Y[real_path_index]=0;
				real_path_X[real_path_index]=6;
				real_path_drop[real_path_index]=0;
				real_path_index=real_path_index+1;
			}


		}
		else if(ny>2)
		{
			if(mx==0)
			{
				//insert path 30
				real_path_Y[real_path_index]=3;
				real_path_X[real_path_index]=0;
				real_path_drop[real_path_index]=0;
				real_path_index=real_path_index+1;
			}
			else if(mx==6)
			{
				//insert path 36
				real_path_Y[real_path_index]=3;
				real_path_X[real_path_index]=6;
				real_path_drop[real_path_index]=0;
				real_path_index=real_path_index+1;
			}
		}
	}
```

```
		else
		{
			if(ny==0)
			{
				result=abs(my-0)+abs(mx-0)+abs(ny-0)+abs(nx-0);
				result2=abs(my-0)+abs(mx-6)+abs(ny-0)+abs(nx-6);
				if(result2<result)
				{
					real_path_Y[real_path_index]=0;
					real_path_X[real_path_index]=6;
					real_path_drop[real_path_index]=0;
					real_path_index=real_path_index+1;
				}
				else
				{
					real_path_Y[real_path_index]=0;
					real_path_X[real_path_index]=0;
					real_path_drop[real_path_index]=0;
					real_path_index=real_path_index+1;
				}
			}
		}
		return 0;
}
//--------------------------------------
int distance(int my, int mx, int ny, int nx)
{
		int result=0;
		int result2=0;

		if(my<=2 && ny<=2)
		{
			if(my!=0 && ny!=0 && mx!= nx)
			{
				result=abs(0-my)+abs(0-ny)+7;//4
				result2=abs(3-my)+abs(3-ny)+7;
				if(result2<result)
```



```
				{
					result=result2;
				}
			}
			else
			{
				result=abs(my-ny)+abs(mx-nx);//1
			}
		}
		else if(my>2 && ny>2)
		{
			result=abs(my-ny)+abs(mx-nx);//2
		}
		else
		{
			result=abs(my-3)+abs(mx-0)+abs(ny-3)+abs(nx-0);//3
			result2=abs(my-3)+abs(mx-6)+abs(ny-3)+abs(nx-6);
			if(result2<result)
			{
				result=result2;
			}
		}
		return result;
}
```

이번 프로젝트에 대한 반성 / 다음 프로젝트 계획

/ 다음 목표

오류 가능성 처리, 속도 높이기

지도교사 확인 및 의견

1단계부터 3단계까지 완주 확인 다음 상점으로 건너갈 때, 색상선택기를 읽으러갈 때 잘못될 경우가 있을 수 있으므로 오류 가능성에 대비하고 느린 속도를 개선할 필요가 있음. 손님을 내려줄 경로의 중복이 생길 경우 먼저 내린 손님을 밀어내는 경우가 있으므로 이러한 가능성을 근본적으로 제거할 필요가 있음. 하차점이 시작점 또는 종료점에 있는 경우 경로 디스플레이와 승객하차에 이상이 없는 지 시험해보고 수정해야함.	(서명)