

0. 작성 시 주의사항

※아래의 작성 양식(제출분량, 폰트, 크기, 줄 간격 등)을 미준수 시 서류 평가의 감점요인됨

※ 제출 분량 : A4 용지 상세내용 포함 10 page 이내

※ 작성 양식 (폰트 : 맑은 고딕 / 폰트 크기 : 10pt / 자간 : 0% / 장평 : 100% / 줄 간격 : 130%)

※ 제출 포맷 : pdf

1. 팀 정보

팀명	eco_BARAM	팀장	김진원
팀원	이창훈	팀원	이동규
팀원	배재성	팀원	

2. 개발계획서 요약

개요 및 목표	스마트 팩토리 내에는 수많은 운반용 로봇들이 있다. 이 로봇들은 별도의 전원으로 가동되며 충전이 필요하다. 동시에 많은 수의 로봇이 충전을 한다면 생산성의 손실을 불러 일으킨다. 우리는 이러한 손실을 최소화하고자 한다.
개발 내용 (간단한 Spec기재)	라즈베리파이3B+, 아두이노 우노, Xbee 쉴드, Xbee 통신모듈, AX-12 서보모터(변경가능)

3. 개발계획서

0. 작품명 : 물류 운송 모바일 로봇의 가동률을 극대화하는 스케줄링 시스템

1. 개요

1.1. 작품 개요

이 시스템은 다수의 로봇을 가장 효율적으로 운영하는 방법을 제시한다. 그 중에서 배터리의 잔량 측면에 집중하여 시스템을 제어할 것이다.

아두이노로 제작된 운송로봇은 업무를 수행하며, 주기적으로 배터리의 잔량을 라즈베리파이에

전송한다. 라즈베리파이는 이를 고려하여 다수의 운송로봇에 업무를 분담하는 군집제어를 담당하게 된다.

1.2. 개발 목표

1. 아두이노를 이용하여 여러 대의 운송로봇을 제작한다.
2. 지그비 통신을 사용하여 운송로봇의 배터리 잔량을 라즈베리파이로 전송한다.
3. 최대한 많은 로봇이 지속적으로 가동되도록 하는 알고리즘을 만든다.
4. 로봇간에 동선이 충돌하지 않도록 하는 알고리즘을 고안한다.
5. 각 로봇의 배터리 잔량 값과 알고리즘을 통해 로봇들에게 충전 명령을 내린다. 로봇은 스스로 충전 도크로 간다.

2. 스마트 팩토리 관련 연구내용

기존 공장의 모바일 로봇의 스케줄링 시스템에 대한 상용화 사례는 찾아보기 힘들며, 아직은 연구 단계이다. 일례로 한국생산기술연구원에서는 A*알고리즘을 이용하는 등 로봇의 이동 시간을 줄이려고 노력하고 있다. 이외에도 다수의 연구가 작업 타겟과 로봇의 최단경로를 찾는 방향으로 진행되고 있다.

본 참가작은 충전 상태인 로봇의 수를 최소화하여 최대한 많은 로봇들이 가동되도록 하는 알고리즘을 제시할 것이다. 기존의 연구들이 집중하지 않았던 충전시간 관련 이슈를 해결한다. 이는 운송로봇의 배터리 잔량에 따라 어떤 로봇을 언제, 얼마나 충전시켜줄지에 관련한 스케줄링 시스템을 의미한다.

우선 기존 공장에서 사용하는 라인트레이서를 이용한다. 좀 더 정확한 라인 센싱을 위해 정규화, 가중치를 이용한다. 정규화를 통한 불규칙적인 데이터 값들의 형태를 맞춰주며, 이 값에 가중치를 부여하여 라인트레이서가 움직이는데 있어 정확성을 높인다.

n대의 라인트레이서들의 현재 위치, 예상 위치, 목표 위치, 배터리 잔량 등의 데이터를 저전력, 1대 다수의 통신을 가능하게 하는 Zigbee통신을 통해 Raspberry Pi로 전달한다. 전달받은 데이터는 Raspberry Pi의 충돌방지 알고리즘, 충전 스케줄링 알고리즘을 거쳐 다시 라인트레이서에 전달되어 공정을 효율적으로 운영한다.

이러한 솔루션을 기존 연구와 동시에 적용하게 되면 더욱 효율적인 스마트팩토리 환경을 구성할 수 있을 것이다.

참고자료 : [중대형 물류센터 적용이 가능한 물류운송로봇 스케줄링 기술 개발 , 2016.1.30]

3. 스마트 팩토리 관련 공부 내용

3.1. 기술적 요구사항

1. 라인 센싱

정규화 : 적외선 센서의 출력은 주변의 환경에 따라 값이 다르게 나오고, 각 센서마다의 값도 다르게 나타나므로 각 센서들의 최대, 최소값을 소프트웨어적으로 모두 같게 만드는 작업을 정규화 작업이라고 한다. 정규화 수식은 <그림 1>과 같다.

$$\text{정규화 값} = \frac{\text{센서값} - \text{센서의 최소값}}{\text{센서의 최대값} - \text{센서의 최소값}} * \text{분해능}$$

<그림 1>

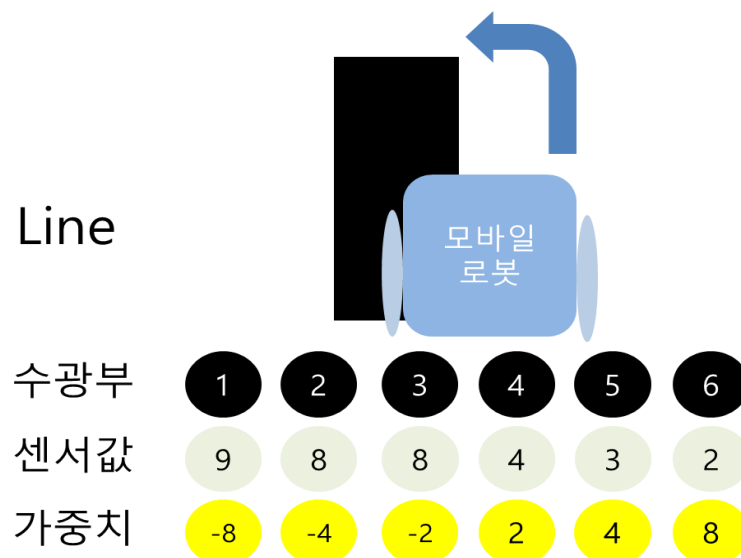
가중치: 모바일 로봇이 안정적으로 방향을 전환하기 위해서 각 센서 값에 가중치를 곱하여준다. 가중치를 적용하는 방법은 <그림 2>와 같다.

가중치가 적용된 데이터 = 정규화된 센서값 X 가중치 값



<그림 2>

가중치가 적용된 데이터로 모바일 로봇이 현재 어느 쪽으로 기울여져 있는지 알 수 있다.

<그림 5>를 예로 들자면 현재 모바일 로봇의 센서 값이 음의 값으로 치우쳐져 있으므로 모바일 로봇이 라인을 기준으로 오른쪽으로 치우쳐져 있음을 알 수 있다.



2. 통신 – Zigbee통신

구분	Bluetooth 	Wi-Fi 	Zigbee 
전송거리	~ 10m	~100m	~ 100m
전송속도	~ 24Mbps	11M / 54Mbps	~ 250Kbps
최대 채널 수	7	14	32000
소비전력	중간	높음	매우 낮음
복잡성	낮음	높음	낮음
비용	낮음	높음	낮음

- 용량이 적은 배터리를 사용하기에 저전력의, 그리고 1:N의 통신을 하기 위해 Zigbee 통신을 선택했다. Zigbee통신을 하기 위해 Xbee 모델을 사용할 것이며, X-CTU 프로그램을 이용하여 통신설정을 할 것이다.

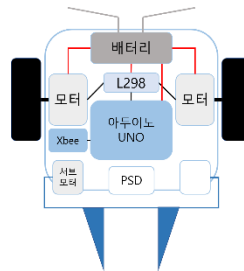
개발 툴: 아두이노 IDE, Raspbian-vim, nano text editor(개발 툴), GCC 컴파일러, X-CTU(XBee 통신설정 프로그램)

3.2. 개발 방법

- 결과물 도출을 위해 어떤 절차를 적용할 것인지 제시한다.
- 최적의 결과물 도출을 위해 중간 단계에서 어떤 평가방법을 활용할 것인지 제시한다.

1. 하드웨어 제작

1.1 운송로봇 제작

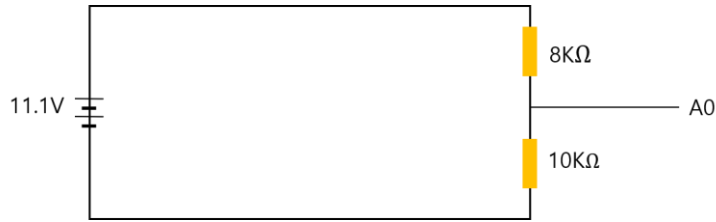


- 준비물(1대 제작기준) : 아두이노 우노, Xbee, Xbee 쉴드, DC 모터 2개, 모터드라이브, 서보모터 1개, 수광부*6, 발광부*6, 바퀴 2개, 볼케스터 1개, 배터리 1개
- 차체 앞에 수광, 발광부를 부착시켜 길을 찾을 수 있도록 제작한다.
- 차체 앞에는 지게차와 같은 역할을 하도록 서보모터를 부착하여 제어한다.
- 차체 뒤에는 배터리를 충전할 수 있도록 따로 금속을 떼낸다.

1.2. 배터리 잔량 확인방법

- 배터리는 11.1v를 사용한다 가정한다.
- 아두이노의 ADC는 10bit인 0~1023까지 읽을 수 있고, 1023은 5v에 매칭이 된다.

- 11.1를 5v까지 받을 수 있도록 회로를 설계해야 한다. 이는 즉 $\frac{5V}{11.1V} \approx 45\%$ 의 전압을 가져가면 되므로 약 8K Ω , 10K Ω 두개의 저항을 사용하여 ADC를 측정하면 된다.



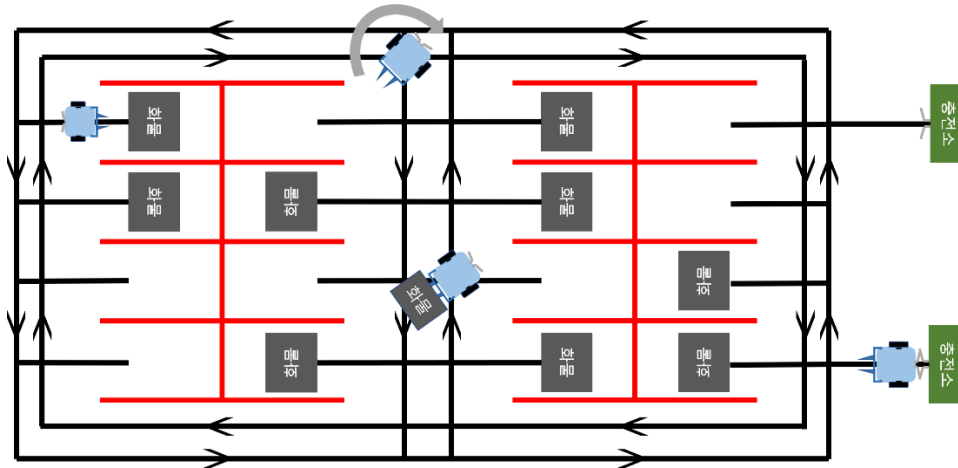
- 위 그림과 같이 회로를 구성하면 배터리의 잔량을 받을 수 있을 것이다.

1.3 배터리 충전



- 파워 서플라이를 라인트레이서 뒷부분과 잘 접촉할 수 있도록 금속판으로 만든다.

1.4 맵 구현



위와 같은 모양의 맵을 포맥스와 절연테이프를 이용하여 제작한다.

2. 통신 설정

2.1 N Xbee 통신 구축

Command	Description	가능한 값들	초기값
ID	XBee의 네트워크 ID	0~0xFFFF	3332
CH	XBee의 채널	0x0B~0x1A	0x0C
SH, SL	시리얼 넘버 (SH는 high 32비트, SL은 low 32비트)	0~0xFFFFFFFF	각각 다르다
MY	XBee 모듈의 내주소	0~0xFFFF	0
DH, DL	도착 주소 (DH는 high 32비트, DL은 low 32비트)	0~0xFFFFFFFF	DH = 0, DL = 0
BD	Baud rate	1200bps~1152bps	3(9600bps)

- 즉, CH, ID는 모두 같은 값을 가져야하고, MY는 자신의 지그비의 주소와 DL은 보낼 지그비의 주소를 설정해주면 된다

이후 아두이노의 시리얼 통신을 진행하면 된다.

3. 알고리즘 구현

3.1 충전 스케줄링 알고리즘

최대한 많은 로봇이 지속적으로 가동되도록 하는 알고리즘을 고안한다. 이는 곧 어떤 로봇을 언제, 얼마나 충전시킬지를 결정하는 알고리즘을 고안하는 것과 같다.

변수를 설정한 후, 필요한 조건을 구하고, 이에 따른 명령을 알아보도록 한다.

우선 필요한 변수들을 다음과 같이 칭한다.

전체 로봇 수	N
충전기 수	N_{chrg}
시간당 배터리 충전량	$Chrg$
시간당 최대 배터리 소모량	C_{max}
배터리의 잔여량	B (0~100의 범위)
n번째로 작은 배터리의 잔여량	B_n
모든 배터리 잔여량의 합	B_{sum} (0~100N의 범위)

다음의 세 가지 조건을 둔다.

$$\textcircled{1} (N - N_{chrg})C_{max} \leq N_{chrg} * Chrg$$

이 조건을 만족하지 않을 경우 모든 충전기를 사용하더라도 B_{sum} 은 감소한다.

즉 충전과 작업 중 어느 하나도 하지 않게 되는 로봇이 존재하게 된다.

예시)

$$N=4, N_{chrg}=1, C_{max}=12, Chrg = 30$$

로봇 4대의 초기 배터리량이 100, 100, 100, 100이었다고 생각해보자. ($B_{sum}=400$)

1대를 충전하고 3대를 작동시키면 130, 88, 88, 88이 된다.

다음 로봇을 충전시키면 118, 118, 76, 76이 된다.

다음 로봇을 충전시키면 106, 106, 106, 64가 된다.

다음 로봇을 충전시키면 94, 94, 94, 94가 된다. ($B_{sum}=376$)

모든 시간에 모든 충전기를 가동하였음에도 불구하고 B_{sum} 가 감소했다.

$$\textcircled{2} B_1 \geq B_{min}$$

B_1 : 배터리가 가장 작은 로봇의 배터리 잔여량

B_{min} : 하던 일을 마무리한 후 충전소로 복귀하기까지 필요한 배터리의 양

배터리가 B_{min} 보다 낮아지게 되면 충전소로 복귀하기 전에 배터리가 방전되는 일이 발생할 수 있다.

$$\textcircled{3} B_n \geq B_{min} + \left\lfloor \frac{n-1}{N_{chrg}} \right\rfloor C_{max}$$

가장 핵심적인 조건이다. 약간 복잡하므로, 우선 충전기의 수 N_{chrg} 을 1로 두고 도출한 수식에서 N_{chrg} 를 모든 자연수로 확장하도록 하겠다.

N_{chrg} 가 1이라고 가정하자. B_1, B_2, B_3 는 각각 $B_{min}, B_{min}+C_{max}, B_{min}+2C_{max}$ 의 값을 가져야 한다. 단위시간이 흘러 임의의 배터리 잔여량이 C_{max} 만큼 감소한 상황에서도 ①번 조건을 만족해야 한다. 즉 2번째로 충전할 배터리의 잔여량은 $B_{min}+C_{max}$, 3번째로 충전할 배터리의 잔여량은 $B_{min}+2C_{max}$, n 번째로 충전할 배터리의 잔여량은 $B_{min} + (n-1)C_{max}$ 보다 아래로 떨어지지 않아야 한다. 즉 $B_n \geq B_{min} + (n-1)C_{max}$ 의 식을 도출해낼 수 있다.

예시)

$$B_{min}=10, N_{chrg}=1, C_{max}=3$$

1번, 2번, 3번, 4번...

1st cycle 10, 13, 16, 19, ... 1번 로봇이 B_{min} 이다. 충전시킨다.

2nd cycle 20, 10, 13, 16, ... 2번 로봇이 정확히 B_{min} 이 되었다.

3rd cycle 17, 20, 10, 13, ... 3번 로봇이 정확히 B_{min} 이 되었다.

만약 2번 로봇의 초기값이 12였다면 2nd cycle에서 B_{min} 보다 낮게 내려가버렸을 것이다.

또한 3번 로봇의 초기값이 15였다면 3rd cycle에서 B_{min} 보다 낮게 내려가버렸을 것이다.

따라서 $B_1 \geq B_{min}, B_2 \geq B_{min}+C_{max}, B_3 \geq B_{min}+2C_{max}$ 를 만족해야 한다..

즉 $B_n \geq B_{min} + (n-1)C_{max}$ 를 만족한다.

이제 N_{chrg} 를 1 이상으로 확장하자. $B_1 \sim B_{N_{chrg}}$ 는 각각 B_{min} 까지 떨어져도 모두 충전 가능하므로 문제가 없다. $B_{N_{chrg}+1} \sim B_{2N_{chrg}}$ 은 $B_{min} + C_{max}$ 까지 떨어져도 무방하다. (1 단위시간 동안 $B_1 \sim B_{N_{chrg}}$ 는 B_{min} 인 상태에서 충전을 하고, 1단위시간이 끝난 직후 $B_{N_{chrg}+1} \sim B_{2N_{chrg}}$ 가 B_{min} 으로 떨어지고 충전을 시작한다.)

이에 따라 B_n 은 $B_{min} + [(n-1)/N_{chrg}] * C_{max}$ 까지 떨어져도 무방하며, 그 아래로 떨어질 경우 ①번 조건을 충족시키지 못하게 된다.

예시)

$B_{min} = 10, N_{chrg} = 2, Chrg = 10, C_{max} = 3$

1번, 2번, 3번, 4번...

1st cycle 10, 10, 13, 13, 16, 16, 19, ... 1, 2번 로봇이 B_{min} 이다. 충전시킨다.

2nd cycle 20, 20, 10, 10, 13, 13, 16, ... 3, 4번 로봇이 정확히 B_{min} 이 되었다.

3rd cycle 17, 17, 20, 20, 10, 10, 13, ... 5, 6번 로봇이 정확히 B_{min} 이 되었다.

만약 3번 로봇의 초기값이 12였다면 2nd cycle에서 B_{min} 보다 낮게 내려가버렸을 것이다.

또한 6번 로봇의 초기값이 15였다면 3rd cycle에서 B_{min} 보다 낮게 내려가버렸을 것이다.

따라서 $B_1, B_2 \geq B_{min}, B_3, B_4 \geq B_{min} + C_{max}, B_5, B_6 \geq B_{min} + 2C_{max}$ 를 만족해야 한다.

즉 $B_n \geq B_{min} + [(n-1)/N_{chrg}] * C_{max}$ 를 만족한다.

이상으로 세 가지 조건을 모두 설정하였다. 이제 각 조건의 충족 여부와 로봇의 충전 명령의 관계를 도출해보도록 한다.

우선 ①번 조건을 만족하도록 로봇과 충전기의 수를 결정해야 한다. 이 관계는 충전량 $Chrg$ 와 소모량 C_{max} 에 달려있다.

이제 ②, ③번 조건의 충족 여부에 따라 로봇을 어떻게 충전시킬지를 정한다.

②, ③번 조건을 모두 만족하는 상황에서는 모든 로봇을 작업에 투입한다.

②번 조건을 불만족하는 상황에서는 해당 로봇을 충전소로 보낸다.

③번 조건을 불만족하는 상황이 되면 N_{chrg} 만큼의 로봇을 충전소로 보낸다.

여기까지만 고려하게 되면 ②번 조건을 만족하는 상황에서 ③번 조건에 따라 0대 혹은

N_{chrg} 만큼의 로봇을 충전하게 된다. 그 사이의 값만큼의 로봇을 충전할 수 없다는 것은 상당히 비효율적이다.

그래서 ③번 조건의 N_{chrg} 변수를 1에서부터 1씩 늘리며 ③번 조건을 변형시킨다.

이렇게 만들어진 조건들을 ③-1번 조건~ ③- N_{chrg} 번 조건이라고 부르자. 한편 ③-1번 조건을 만족한다면 ③- N_{chrg} 번 조건을 만족하게 된다.

즉 ③-1번 조건이 더 만족하기 어려운 조건인데, 이 조건을 만족하면 로봇을 하나도 충전하지 않아도 된다. (②번 조건 만족시)

③-1번 조건을 불만족하며 ③-2번 조건을 만족하면 로봇을 1대 충전하면 된다. (②번 조건을 불만족하는 로봇이 1대 이하일 시)

③-2번 조건을 불만족하며 ③-3번 조건을 만족하면 로봇을 2대 충전하면 된다. (②번 조건을 불만족하는 로봇이 2대 이하일 시)

③- $(N_{chrg}-1)$ 번 조건을 불만족하며 ③- N_{chrg} 번 조건을 만족하면 로봇을 $N_{chrg}-1$ 대 충전하면 된다. (②번 조건을 불만족하는 로봇이 2대 이하일 시)

③- N_{chrg} 번 조건을 불만족하면 로봇을 N_{chrg} 대 충전하면 된다. (충전 가능한 모든 로봇을 충전)

물론 여기에서 n대를 충전하라는 명령은 배터리가 작은 순서대로 나열했을 때 앞에서부터 n대를 충전시키라는 명령이다.

위와 같은 방식으로 가장 효율적인 충전 대수 및 충전 대상을 찾을 수 있다.

3.2 최단 경로 알고리즘

맵은 위쪽, 아래쪽 통행로가 있으며 우측통행을 하도록 구성되어 있고, 위쪽으로 이동과 아래쪽으로 이동중 어떤 것이 더 빠를 것이냐 하는 이슈가 있다.

a = 시작점의 y 좌표, b = 도착점의 y 좌표를, N = y 축의 크기라 하면,

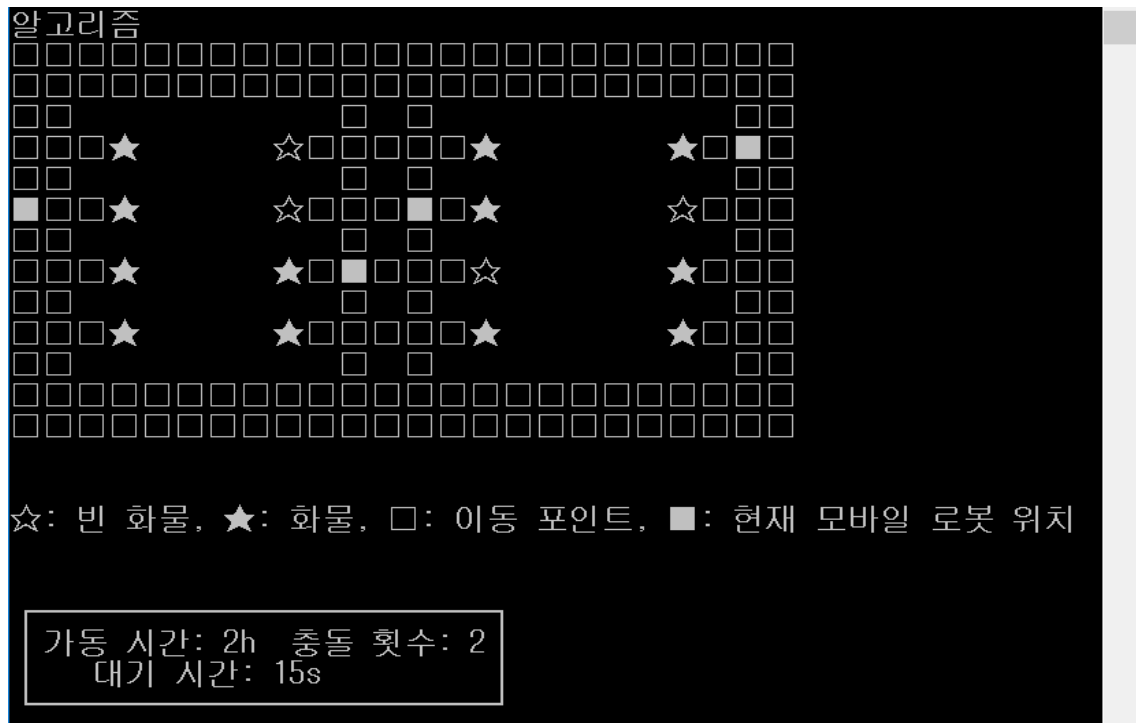
위로 이동할 때의 cost는 $a+b+1$, 아래로 이동할 때의 cost는 $N-a-1 + N-b = 2N-a-b-1$ 이 나오게 된다.

이 두개의 cost 중 낮은 값의 경로로 최단경로를 설정한다.

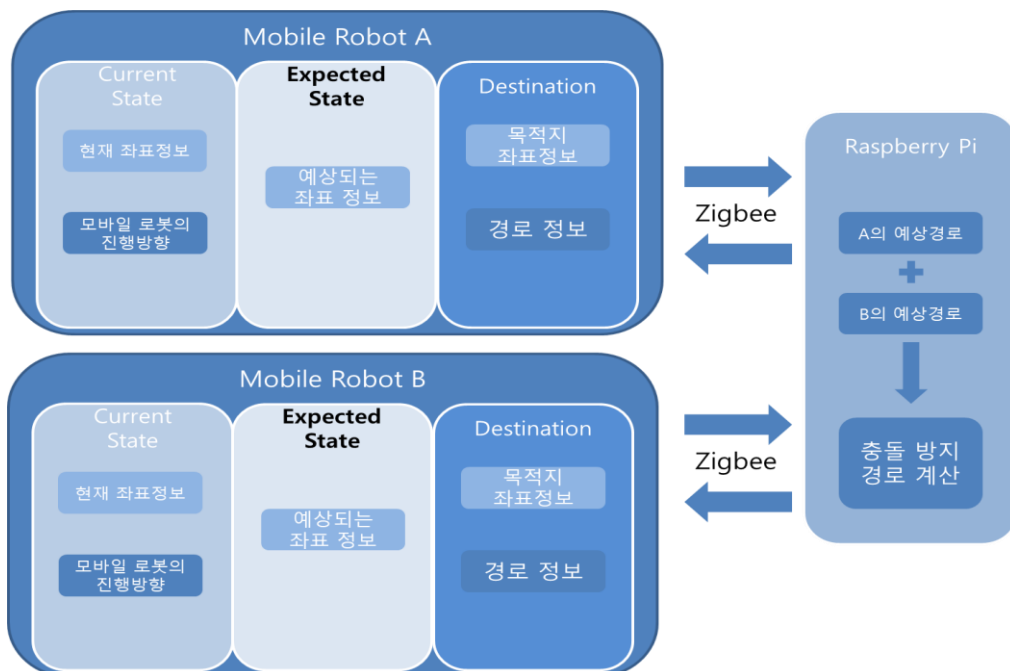
이 때 같은 cost가 나오는 경우 교차로를 지나지 않는 경로를 선택한다.

3.3 최단 경로 검증 알고리즘

Visual Studio를 이용하여 가상의 필드를 만든다. 가상의 필드 안에서 시뮬레이션을 통해 모바일 로봇간의 정면충돌 횟수, 대기시간 체크를 통해 충돌방지 알고리즘을 평가한다. .



모바일 로봇은 공정에서 이동하는 경로를 좌표 값과 자신의 위치를 나타내는 좌표정보, 예상되는 좌표정보를 나타내주며 실시간으로 갱신시킨다. 그리고 지그비 통신을 통해서 갱신된 좌표 값을 라즈베리파이로 전송한다. n대의 모바일 로봇은 이동하면서 자신이 가지고 있는 예상 좌표정보와 라즈베리파이가 가지고 있는 예상좌표를 비교하여 충돌을 피한다.



4. 예상되는 장애요인과 해결방안

예상문제1. 배터리 잔량 값이 선형적으로 증감하지 않을 수 있다.

- 라그랑주 보간법을 사용하여 배터리 잔량을 선형화한다.

5. 단계별 개발계획 및 참여인원과 업무 분장

김진원 : 충돌 방지 알고리즘 제작 담당

배재성 : 충전 스케줄링 알고리즘 담당

이창훈 : 운송로봇, 맵 제작,배터리 측정 및 충전기 제작 등 하드웨어 전반적인 부분 담당

이동규 : Zigbee 통신 설정, 알고리즘 시뮬레이션 담당

진행 목표	6M	7M	8M	9M
운송로봇 및 맵 제작				
배터리 측정 및 충전기 제작				
통신 설정				
충전 스케줄링 알고리즘				
충돌 방지 알고리즘				
알고리즘 시뮬레이션				
적용 및 디버깅				