[UCI]I804:24011-200000233041

2006年 8月
碩士學位論文

영역센서를 가진 이동로봇의 동작시뮬레이션 및 응용

朝 鮮 大 學 校 大 學 院

制 御 計 測 工 學 科

金 成 俊

영역센서를 가진 이동로봇의 동작시뮬레이션 및 응용

Development and Application of Motion Simulator for a Mobile Robot with Range Sensors

*2006*년 *8*月 *25*日

朝 鮮 大 學 校 大 學 院

制 御 計 測 工 學 科

金 成 俊

영역센서를 가진 이동로봇의 동작시뮬레이션 및 응용

指導敎授 高 樂 溶

이 論文을 碩士學位申請 論文으로 提出함.

*2006*年 *4*月 日

朝鮮大學校 大學院

制御計測工學科

金 成 俊

金成俊의 碩士學位論文을 認准함

委員長 朝鮮大學校 敎授 印

委 員 朝鮮大學校 敎授 印

委 員 朝鮮大學校 敎授 印

*2006*年 *5*月 *29*日

朝鮮大學校 大學院

목 차

List of Tables

List of Figures

ABSTRACT

Obstacle Avoidance of Autonomous Mobile Robot using Relative Velocity

Kim, Sung Jun
Advisor : Prof. Ko, Nak Yong, Ph. D.
Dept. of Control and Instrumentation Eng.,
Graduate School of Chosun University

Development of autonomous mobile robot is one of the challenging works in the robotics field. Development of simulator is needed to reduce the danger, time, and expenses for autonomous mobile robot research.

In this paper, we describe a simulator developed for research on autonomous mobile robots. The simulator uses "PioneerⅡ" as a prototype robot. Also, the simulator uses the same command and protocol as these of the "PioneerⅡ". It uses IPC for communication between the modules of controller, sensors, and other equipments.

This facilitates development of softwares in some modules. Each software can be developed independently and can be interchangeable with other softwares modules of simulator functions. The simulator used OpenGL for graphic presentation. It is used to test an obstacle avoidance algorithm.

제 1 장 서론

자율 이동 로봇의 주행에 있어서 주변 환경인식이 매우 중요하며, 이동로봇의 작업 공간상에서 자율적으로 이동하기위해서는 현재 자신이 어디에 놓여 있는지를 알아내는 자기 위치 추정 기능과 어디로 어떻게 움직여야하는지를 결정하는 경로 계획 및 제어 기능이 필요하다.

가장 기본적인 주행 방법으로는 로봇의 양쪽 바퀴에 장착되어있는 엔코더 센서를 사용한 Dead-reckoning 방법이 사용된다. 그러나 이러한 Dead-reckoning 방법은 로봇바퀴의 슬립과 같은 물리적 현상이나 기구학적 계산의 오차와 같은 많은 문제점을 야기 할 수 있다. 이와 같은 단점을 보완하기 위해 엔코더 센서와 더불어 다양한 센서들과의 융합을 통한 연구가 활발히 이루어지고 있다.

본 논문에서는 여러 가지 로봇 제어 알고리즘을 적용할 수 있는 시뮬레이터를 개발 하였으며, 알고리즘 개발과 성능 향상의 과정 및 가상환경에서 실제 로봇 시스템과 근접한 특성을 갖는지를 비교하기 위해 실제 이동 로봇(PioneerⅡ)을 모델로 하였다. 로봇 하드웨어로는 할 수 없는 알고리즘을 시뮬레이션 하고 다양한 테스트를 한 후 실제 자율 이동로봇 시스템에 적용하여 로봇의 환경 인식, 이동 방향, 안정적 궤적형성 등에 대한 시뮬레이터의 결과와 실제 로봇의 결과를 비교하였다.

제 1 절 연구배경 및 목적

현대 산업사회의 생산현장에는 고정되어 있는 매니퓰레이터의 사용대신에 점차 움직임이 가능한 모바일 로봇 베이스의 적용시도가 진행되고 있으며, 이동 로봇에 매니퓰레이터를 부착할 경우 고정식보다 변화되는 작업에 쉽게 적용이 가능한 장점이 있다. 반면 주변 환경에 대한 정확한 인식과 대응이 작업의 성공을 위해 필요하게 된다. 이러한 이동 로봇을 만들기 위해서는 다음과 같은 필수적인 기술이 요

구된다. 첫째, 실시간으로 작업 환경에 존재하는 장애물에 대한 인식 기능이 있어야 한다. 둘째, 주어진 작업을 성공적으로 완료하기 위하여 작업 공간에서 가장 효율적인 이동 경로를 선택하여 주행하는 경로 계획 기술이다. 이 경로 계획의 문제를 대략적으로 분류하면, 작업의 시작 지점에서부터 작업을 최종적으로 끝마칠 때까지 주행하여야할 전역 경로 설정 문제와 이미 정해진 전역 경로를 따라 이동하는 과정에서 만날 수 있는 장애물에 대한 충돌 회피를 도모할 수 있는 지역 경로 계획 문제 등의 두 가지로 구별된다. 셋째, 작업 과정 전반에 걸쳐 이동 로봇의 상시 위치 파악이 가능하도록 하는 위치 평가 기술, 그리고 로봇에 주어지는 제반 작업 환경과 자율 주행 성능을 보이기에 적합한 이동 로봇 시스템의 하드웨어 구성 기술 등의 네 가지의 세부 기술이 요구된다.

이동 로봇이 주변 환경을 인식하고 환경의 변화에 대한 대처 능력을 갖도록 하기 위해서는 로봇에 사용되는 센서들의 감지 영역이 가능한 한 광범위한 영역에 대하여 정확한 정보 획득 및 신속성이 요구된다. 특히 이동 로봇이 작업하는 환경이 다양한 형태의 장애물이 산재한 공간일 경우, 이에 대한 효과적인 환경 검출 방법에 관한 연구가 있어야 할 뿐 만 아니라, 센서 별로 고유의 검출 특성을 가짐에 따라 인식할 수 없는 환경 범위나 물리량을 갖기 때문에 이에 대한 센서의 융합 대책도 필요하다. 그리고 자율 주행 알고리즘은 로봇이 계획한 전역 경로를 추종하는 동안 로봇의 정확한 위치 평가가 필요하고, 지역 장애물 회피 계획은 로봇을 구성하는 주행체의 동력학적 제한 조건과 환경 조건을 충분히 반영하도록 하며, 로봇에 사용하는 센서 시스템의 검출 특성과도 조화를 이루는 효율적인 알고리즘이 되어야 한다.

지역 장애물 회피 성능은 로봇이 수행하는 작업의 완성도를 높이는데 필수적인 성능으로서 센서 정보와 밀접한 관련을 갖는다. 센서 검출 정보에 기반을 두어 회피 동작을 계획하는 대부분의 방법들은 센서 정보의 불규칙한 정도와 실시간 검출 성능 등에 따라 회피 동작의 성능이 결정되기도 한다. 따라서 센서 데이터의 누적을 통해 데이터의 불확실성을 줄이는 방법이 타당하나 계산 단계가 늘어남에 따른 로봇의 실시간 회피 동작에 문제를 야기할 수도 있다. 그러므로 환경의 검출 단계

에서부터 정확도를 높일 수 있는 센서의 선택이 이루어져야 하고, 회피 동작을 계획하는 알고리즘은 간결하면서도 효율적 과정으로 구성되어야 한다. 또한 로봇은 실제 공간에서 운동하는 운동체이므로 로봇이 가지는 운동 특성이 회피 동작 계획 단계에서 반영되어야 함은 물론이며, 이로부터 안정적이고 유연하며 신속한 궤적을 보일 수 있어야 한다.

본 논문에서는 레인지센서 정보를 이용하여 로봇의 주변 환경을 인식하고, 이 인식에 근거하여 적합한 동작을 선택하여 주어진 목표 위치까지 장애물과 충돌하지 않으면서 도착하는 알고리즘을 시뮬레이션으로 구현하는데 목적을 둔다.

제 2 절 연구 방법

본 논문에서는 로봇의 시작위치에서 목표 위치까지 전체적인 동작을 계획하고 실시간 지역 장애물 회피를 위해 충돌회피가능도와 가상 거리 함수 개념을 도입하였다. 제안한 동작조정 알고리즘의 성능특성을 확인하기 위해 시뮬레이션 모의실험을 하였으며, 효과적인 주변 환경 인식 및 자율 주행 로봇의 위치 결정을 위해서 로봇의 자기 위치인식(self localization), 충돌회피(obstacle avoidance), 경로 계획(path planning)을 할 수 있도록 로봇의 전방에 정확한 거리 정보를 얻을 수 있는 레이저 레인지 파인더(LMS)를 로봇베이스(Pioneer2)에 장착하여 성능평가를 하였다.

제 2 장 기초 이론 및 문제 구성

제 1 절 레이저 센서의 검출 원리 및 특성

1. 레이저 센서의 원리와 특성

레이저는 유도 방출에 의한 빛의 증폭(light amplification by stimulated emission of radiation: LASER) 작용을 총칭하는 말이다. 주로 이동 로봇에서 사용하는 레이저 센서는 적외선 레이저 빔(LASER beam)으로 주변 환경을 탐색하는 광학 센서로서, 이 센서는 매우 짧은 빛을 발산하여 물체로부터 반사되어 돌아오는 시간으로부터 거리를 얻어내는 원리이다. 그림 2.1에 레이저 레인지 파인더의 거리 측정 원리를 나타내었다.

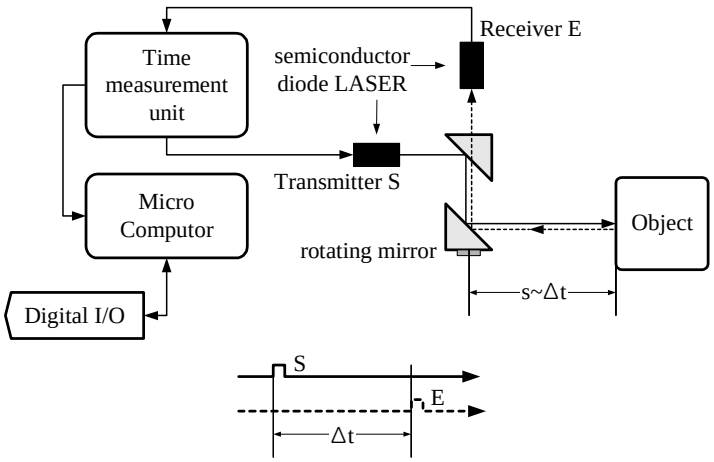

Fig. 2.1 Distance detection principle of SICK LMS-200[14]

발광부는 전기 신호를 레이저 신호로 변환하여 물체를 향해 주사하고, 물체로부터 반사된 신호의 일부는 수광부의 반도체 다이오드 레이저에서 전기적 신호로 변

환된다. 시간 측정 장치에서는 신호가 발생되고 반사되기까지 소요되는 시간을 수정 발진기의 주파수를 이용하여 계수하며, 내부 마이크로프로세서는 이 결과를 거리로 환산하여 검출 정보를 외부로 내보낸다.

Fig. 2.2 SICK LMS-200 LASER range finder[14]

그림 2.2에 SICK社의 LMS-200 레이저 레인지 파인더(LASER range finder)를 보였다. LMS-200은 RS-422/232 통신 방식으로 외부 기기에 인터페이스 되어 센서 제어 명령, 검출 데이터 등을 교환한다. 본 연구에서 사용한 레이저 센서는 LMS-200 레이저 레인지 파인더로서, 주요 사양은 다음 표 2.1와 같다.

Table 2.1 Main specification of SICK LMS-200[14]

Technical data of LMS 200	
Range	80, 10 [m]
Scanning angle	max. 180 ◦
Angualr resolution	0.25, 0.5, 1 [deg.] Adjustible
Respose time	53, 26, 13 [ms]
Resolution / systemic error	10 [mm]/typ. ± 15[mm]
Data interface	RS-232, RS-422
Switching outputs	3×PNP; typ. 24VDC
Laser protection class	1 (eye safe)
Operating ambient temperature	0 ... +50℃
Enclosure rating	IP65
Dimensions	155×210×156 mm3

2. 레이저 센서를 이용한 초음파 센서의 구조적 오류 교정

로봇이 주행하는 실내 환경에 대한 정확한 거리 정보에 기반을 둔 실시간 지역 장애물 회피 방법을 제안하기 위하여 본 연구에서는 레이저 레인지 파인더를 사용하였으며, 검출 결과를 그림 2.3 ~ 그림 2.17에 보였다.

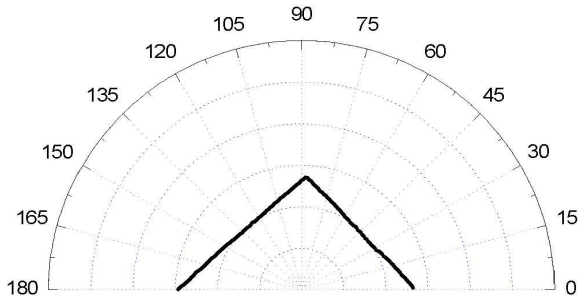

Fig. 2.3 Detect pattern of LASER sensor data about corner model.

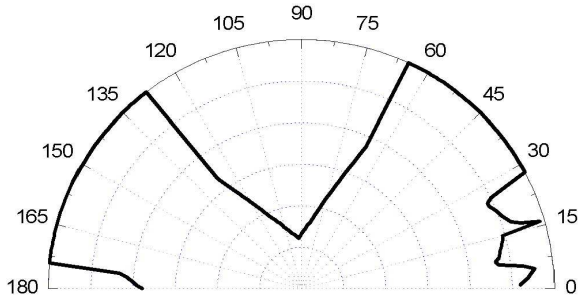

Fig. 2.4 Detect pattern of LASER sensor data about edge model.

그림 2.3는 corner 모델에 대한 검출 결과로서, 180개의 데이터는 실제의 환경을 정확하게 표현하고 있다. 그림 2.4는 edge 모델에 대한 실험 결과이다. 초음파 센서의 경우 15°단위로 배치되어 검출 거리가 길어질수록 데이터의 정확도가 떨어지는 반면 레이저 센서는 1°단위의 데이터 추출이 이루어지므로 검출 영역 전체에 걸쳐 상세한 환경 정보를 얻을 수 있다.

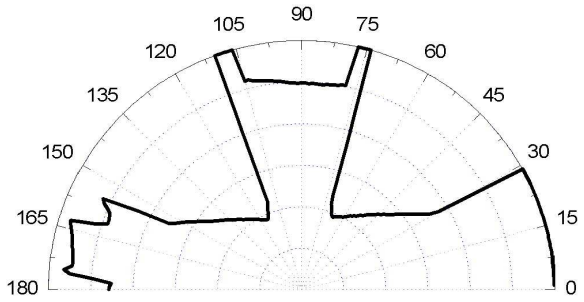

Fig. 2.5 Detect pattern of LASER sensor data about gate model

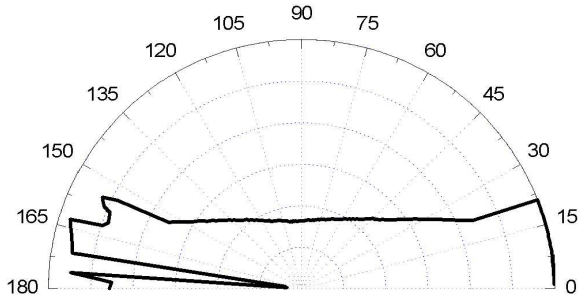

Fig. 2.6 Detect pattern of LASER sensor data about plane model

그림 2.5과 그림 2.6의 gate와 plane 모델에 대한 검출 결과에서도 실제 검출 모델에 대하여 정확성을 가진 상세한 환경 정보를 얻어내었다.

본 연구에서 개발한 이동 로봇용 다중 센서 시스템은 센서 데이터의 평균 갱신 주기가 500 [ms]를 보인다. 따라서 대략적인 로봇 주변의 전 방향에 대한 장애물 정보는 초음파 센서로부터 얻어내고, 로봇이 진행하는 전방에 대한 상세한 환경 정보는 레이저 센서로부터 얻어내어 데이터를 융합한다면, 실시간 지역 장애물 회피 알고리즘에 필요한 견실한 실내 환경 정보를 획득할 수 있을 것으로 판단된다.

3. 센서 모델링

레이저 스캐너로부터 얻어지는 값은 각 방위각에 대한 거리값으로, 극좌표계의 형태를 띄게 된다. LMS200의 경우, 거리 오차의 통계적 표준편차는 5mm이고, 이

는 해상도가 10mm인 것을 이용하여 대략[-5mm, +5mm]사이에서 균일하게 분포된다는 것을 알 수 있다. 방위각의 오차에 대한 통계적 표준편차는 해상도가 0.5˚이므로 대략 [-0.25˚, +0.25˚]사이에서 균일하게 분포된다고 가정할 수 있다. 대개 로봇의 X-Y 직교 좌표계에서 움직이므로, 이를 X-Y 직교 좌표계로 변호나해서 사용하는 것이 바람직하다.

극좌표계의 좌표 (r,θ)를 직교 좌표계의 좌표 (x,y)로 변환하면 다음과 같다.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} rcos\theta \\ rsin\theta \end{bmatrix} \tag{2-1}$$

r과 θ의 확률 분표 함수를 각각 $f_r(r) = \frac{1}{2\Delta r}, f_\theta(\theta) = \frac{1}{2\Delta\theta}$라 하고, r과 θ가 서로 연관되어 있지 않다고 가정하면, 다음과 같은 공분산 값을 얻을 수 있다.

$$\sigma_{xx} = \frac{1}{2}(\bar{r}^2 + \frac{1}{3}\Delta r^2)(1 + \cos 2\bar{\theta}\cos\Delta\theta\frac{\sin\Delta\theta}{\Delta\theta}) - \bar{r}^2\cos^2\bar{\theta}(\frac{\sin\Delta\theta}{\Delta\theta})^2 \tag{2-2}$$

$$\sigma_{yy} = \frac{1}{2}(\bar{r}^2 + \frac{1}{3}\Delta r^2)(1 - \cos 2\bar{\theta}\cos\Delta\theta\frac{\sin\Delta\theta}{\Delta\theta}) - \bar{r}^2\sin^2\bar{\theta}(\frac{\sin\Delta\theta}{\Delta\theta})^2 \tag{2-3}$$

$$\sigma_{xy} = \frac{1}{2}(\bar{r}^2 + \frac{1}{3}\Delta r^2)\cos 2\bar{\theta}\cos\Delta\theta\frac{\sin\Delta\theta}{\Delta\theta} - \bar{r}^2\sin\bar{\theta}cos\bar{\theta}(\frac{\sin\Delta\theta}{\Delta\theta})^2 \tag{2-4}$$

여기서 $\bar{r}$은 거리값의 평균이고, $\bar{\theta}$는 방위각의 평균이다. 위식을 살펴보면, x와 y 값은 서로 상관관계를 갖고 있으며, 거리와 벽면이 로봇을 향하고 있는 방향에 따라 오차가 틀려짐을 알 수 있다. 실제로 정확한 거리와 방위각을 모른다면 복잡한 모델을 사용하는 것이 더 안 좋은 결과를 보일 수 있다. 따라서 x와 y의 오차가 서로 연관관계를 갖고 있지 않고, 각각 5mm의 표준 편차를 갖는 가우시안 오차라는 간단한 수학적 모델을 사용하는 것이 바람직하다. 이와 같은 모델을 사용할 경우, 비교적 정확하게 위치추정을 수행할 수 있다.

제 2 절 이동 로봇의 동작조정

본 절에서는 다중 이동 로봇의 동작조정을 위해 본 논문에서 제안한 개념인 충돌 회피 가능도[11]와 가상 거리 함수에 대하여 알아본다.

첫 번째로 다중 이동 로봇의 동작조정을 수학적으로 표현하고 그 해결방법을 구하기 위한 표기법을 정의한다. 두 번째는 로봇간의 멀어지는 속도를 나타내는 외향속도와 로봇들 사이의 향후 충돌 가능성을 예측할 수 있는 충돌 회피 가능도[11]에 대하여 알아본다. 세번째, 외향속도 개념을 적용한 충돌 회피 가능도[11]의 하나인 가상 거리 함수를 도출해 낸다. 마지막으로 가상 거리 함수에 의해 도출된 거리 값에 기존에 충돌회피 알고리즘에 널리 사용되고 있는 인공 전위계(Virtual Force Field)를 적용하여 로봇에 작용되는 힘을 구하고 서술한다.

1. 다중 이동로봇의 동작조정 문제 구성

다중 이동 로봇의 동작조정 문제를 수학적으로 구성하고 그 해결방법을 구하기 위하여 다음과 같은 표기법을 사용한다.

$P = (x, y)$	2차원 평면에서의 한 점의 위치
$P_j(t) = (x_j(t), y_j(t))$	시각 t에서의 로봇 j 의 위치
$P_{gj} = (x_j, y_j)$	로봇 j 운동의 목표점
$P_{sj} = (x_j, y_j)$	로봇 j 운동의 시작점
m_j	로봇 j 의 질량
r_j	로봇의 j 의 반지름
t_i ($i = 0, ...$)	i 번째 샘플링 시각

위의 표기법을 사용하여 2차원 평면에서 각 시간 t 마다 각각의 이동 로봇의 위치가 주어질 때 로봇이 이를 피하여 시작점 P_{sj} 에서 목표점 P_{gj} 까지 이동하는데

t 시각에서의 로봇의 위치 $P_j(t)$를 결정한다. 구현의 편이성을 위하여 로봇은 모두 원형으로 하며, 로봇의 질량은 0으로 가정한다.

2. 외향 속도 와 충돌 회피 가능도

고정된 물체의 회피를 위해서는 로봇과 물체의 위치 그리고 그 사이의 거리만을 고려하면 된다. 그러나 빠른 속도로 이동하는 로봇의 회피를 위해서는 로봇간의 상대적인 이동성까지 고려하여야 충돌을 피할 수 있다. 이동 장애물의 경우 로봇간의 거리와 로봇간의 운동 방향 및 상대 속도에 의해 로봇들 사이의 향후 충돌 가능성을 예측할 수 있다. 본 장에서는 로봇간의 상대적인 이동성을 고려하기 위하여 기존에 제안되어 있는 내용인 충돌 회피 가능도[11]를 사용한다.

먼저 충돌 회피 가능도를 나타내기 위해 외향 속도[11]와 직선거리를 정의한다.

$d_{j,k}(t)$: t 시간에서의 로봇 j 와 로봇 k 의 직선거리

$v_{j,k}(t)$: t 시간에서의 로봇 j 가 로봇 k에 멀어지는 속도

외향속도 $v_{k,j}(t)$는 단위 샘플링 시간의 로봇 k 의 속도를 로봇 j 로부터 장애물 방향으로의 단위 벡터의 내적 값이다. 이 값이 양수이면 로봇 k 가 로봇 j 로부터 멀어지는 방향으로 이동하는 경우이고, 음수이면 로봇 k 가 로봇 j 를 향해 이동하는 경우이다. 그리고 로봇 k 의 속도에 비례하여 로봇 j 로부터 멀어지는 방향으로 운동할수록 값이 커지고 로봇 j 방향으로 운동할수록 그 값은 작아진다.

외향속도 $v_{j,k}(t)$는 단위 샘플링 시간의 로봇 j 의 속력을 로봇 k 로부터 로봇 j 방향으로의 단위 벡터의 내적 값이다. 이 값 또한 양수이면 로봇 j가 로봇 k 로부터 멀어지는 방향으로 이동하는 경우이고, 음수이면 로봇 j 가 로봇 k 를 향해 이동하는 경우이다. 즉 $v_{k,j}(t)$는 로봇 k 의 로봇 j 에 관한 운동성을 나타내며, $v_{j,k}(t)$는 로봇 j 의 로봇 k 에 대한 운동성을 나타낸다.

$$d_{j,k}(t)=\sqrt{(X_k(t)-X_j(t))^2+(Y_k(t)-Y_j(t))^2}-(r_k+r_j) \tag{2-5}$$

$$=\|P_k(t)-P_j(t)\|-(r_k+r_j)$$

$$v_{j,k}(t)=\dot{P}_j(t)\frac{(P_j(t)-P_k(t))}{\|P_j(t)-P_k(t)\|} \tag{2-6}$$

로 표현할 수 있으며 이것을 그림으로 나타내 보면 다음과 같다.

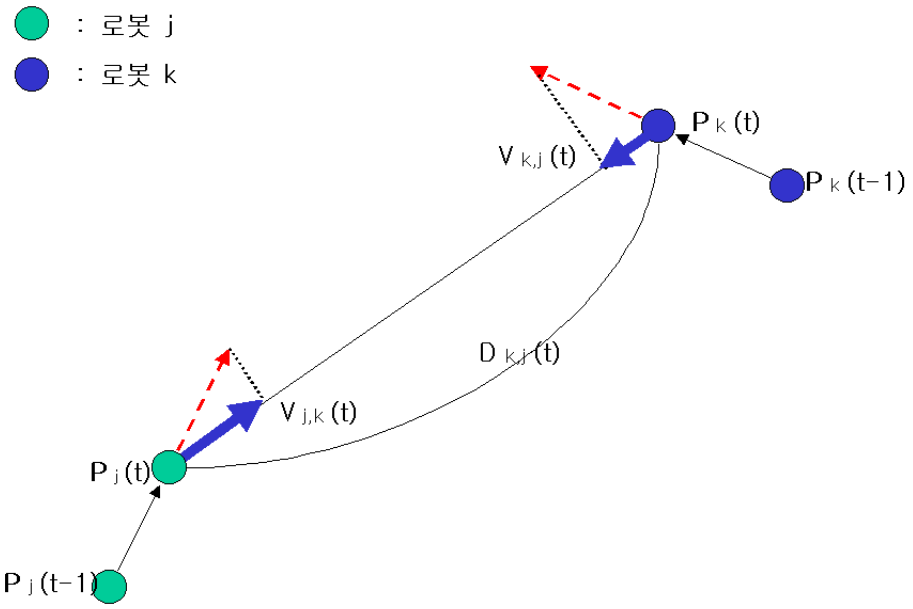

Fig 2.7 outword speed vj,k(t) and vk,j(t)

앞에서 정의한 외향속도 $v_{j,k}(t)$, $v_{k,j}(t)$와 로봇간의 직선거리 $d_{j,k}(t)$를 이용하여 로봇이 다른 이동 로봇을 피할 수 있는 가능성을 나타내는 지수인 충돌 회피 가능도 [11](AVM : Avoidability Measure)를 정의한다.

정의 *1*

시각 t 에서의 충돌 회피 가능도는 로봇간의 거리 $d_{j,k}(t)$, 로봇간의 외향속도 $v_{k,j}(t)$, $v_{j,k}(t)$에 관한 함수로 다음의 조건을 만족시킨다.

(조건1) 로봇 j 와 로봇 k 사이의 거리 $d_{j,k}(t)$가 클수록 충돌 회피 가능도가 크다.

(조건2) 로봇 k 의 로봇 j 에 대한 외향속도 $v_{k,j}(t)$가 클수록 충돌 회피 가능도가 크다.

(조건3) 로봇 j 의 로봇 k 에 대한 외향 속도 $v_{j,k}(t)$가 클수록 충돌 회피 가능도가 크다.

위의 충돌 회피 가능도에 관한 정의에 의해, 로봇의 장애물 회피 동작 계획 문제는 충돌 회피 가능도를 충돌 회피가 보장되는 최소값 이상 유지하도록 동작 계획하는 문제가 된다. 이러한 충돌 회피 가능도의 하나로써 가상 거리 함수를 제안한다.

3. 가상 거리 함수

정의 1의 충돌 회피 가능도로서의 세 가지 조건을 만족 시키는 함수는 여러 가지로 만들어 낼 수 있으며, 기존에 제안되어 있는 가상거리 함수는 다음과 같다.

$$vd_{j,k}(t) = \frac{\alpha + v_{j,k}(t)}{\beta - v_{k,j}(t)} d_{j,k}(t) \qquad (2\text{-}7)$$

위의 식은 기존 가상거리 함수이며, 이 함수는 로봇 j 의 k에 대한 외향속도 $v_{j,k}(t)$ 가 커지면 가상거리도 실제거리에 비해서 커지게 되고, 반대로 $v_{j,k}(t)$가 작아지면 가상거리도 실제거리에 비해 작아지게 된다. 마찬가지로 k의 j 에 대한 외향속도 $v_{j,k}(t)$ 가 커지면 가상거리는 실제거리에 비해 커지게 되며, 작아지면 그 반대의 현상이 나타나게 된다. 그러므로 기존의 가상거리 함수는 로봇 j 와 k 에 대한 운동성이 모두 고려된 거리함수 이다. 그러나 이 함수는 α 와 β 가 각각 분자와 분모에 위치하고 있기 때문에 α 와 β가 같지 않으면 가상거리가 의도와는 다른 값으로 나오게 된다. 예를 들자면 α = 100, β = 50인 경우 가상거리는 실제거리의 약

2배의 거리에서 커지고 작아지는 현상이 발생한다. 즉, 각 로봇들의 외향속도가 작아진다 할지라도 실제거리보다 가상거리가 크게 나오는 현상이 발생할 수 있기 때문에 로봇끼리의 충돌이 발생할 가능성이 있다. 즉 기존의 가상거리 함수는 α / β = 1 즉 α = β를 만족할 때 가상거리의 신뢰성을 믿을 수 있다. 그러나 α = β 인 경우는 로봇 j의 k에 대한 운동성과 로봇 k 의 j 에 대한 운동성 각각을 고려할 수 없다. 이러한 단점을 해결하기 위하여 개선된 가상 거리 함수를 만들어본다.

우선 가상 거리 함수는 $vd_{j,k}(t)$로 정의하고 이는 t 시간에서의 로봇 j 와 로봇 k 와의 직선거리인 $d_{j,k}(t)$ 비례하는 것이 정의 1에서의 조건 1과 부합하므로

$$vdj,k(t) = K \times d_{j,k}(t) \tag{2-8}$$

로 식을 구성해도 조건 1에 위배됨이 없다.

조건 2에서 $v_{k,j}(t)$가 클수록 충돌 회피 가능도가 크므로 $v_{k,j}(t)$가 커지면 실제 거리보다 가상거리가 크게 나올 수 있도록 표현되어야 하므로

$$vdj,k(t) = L \times (\alpha + v(t)) \tag{2-9}$$

처럼 표현할 수 있다.

조건 3에서 $v_{k,j}(t)$또한 커질수록 충돌 회피 가능도가 커지므로 $v_{k,j}(t)$가 커지면 실제 거리보다 가상거리가 크게 나올 수 있도록 표현되어야 하므로

$$vdj,k(t) = M \times (\beta + v_{k,j}(t)) \tag{2-10}$$

식으로 표현해도 조건에 위배됨이 없다.

위 식들에서 나타난 특성을 모두 종합해서 나타낼 수 있는 식은 무수히 많지만 본 논문에서 제안하는 식은 다음과 같다.

$$vd'_{j,k}(t) = (KLM)\frac{\sqrt{(\alpha + v_{j,k}(t))^2 + (\beta + v_{k,j}(t))^2}}{\sqrt{\alpha^2 + \beta^2}}\, d'_{j,k}(t) \tag{2-11}$$

위 식에서 KLM 부분은 상수 이므로 하나의 상수인 N으로 표현가능 하다. 또한 $v_{j,k}(t)$와 $\underline{v}(k,j)(t)$가 0에 가까울 때 $vd_{j,k}(t) = d_{j,k}(t)$인 것이 타당하므로 $KLM = N = 1$ 로 한다.

식을 다시 정리하면 다음과 같다.

$$vd'_{j,k}(t) = \frac{\sqrt{(\alpha + v_{j,k}(t))^2 + (\beta + v_{k,j}(t))^2}}{\sqrt{\alpha^2 + \beta^2}}\, d'_{j,k}(t) \tag{2-12}$$

단, 2-11과 2-12식에서 α 와 β 의 크기가 $v_{j,k}(t)$와 $v_{k,j}(t)$ 의 절대 값보다 작은 경우에는 가상거리 결과 값을 예측할 수 없는 경우가 발생할 수 있으므로 α 와 β의 값은 $v_{j,k}(t)$, $v_{k,j}(t)$의 절대 값보다 충분히 큰 값으로 결정해야 한다.

즉 α 와 β는 다음 식을 만족시키는 상수이다.

$$\begin{aligned} &\alpha > \max|v_{j,k}(t)| > 0 \\ &\beta > \max|v_{k,j}(t)| > 0, \quad (\alpha, \beta \in R) \end{aligned} \tag{2-13}$$

위의 식 2-13의 조건을 만족하는 식 2-12에서 $d_{j,k}(t)$ 가 증가하면 $vd_{j,k}(t)$가 커지고, $v_{k,j}(t)$가 증가하면 $vd_{j,k}(t)$가 커지며 $v_{j,k}(t)$가 증가하면 $vd_{j,k}(t)$도 커진다. 따라서 가상 거리 함수 $vd_{j,k}(t)$는 충돌 회피 가능도로서의 세 가지 조건을 모두 만족시킴을 알 수 있다. 그리고 α ≠ β 인 경우에도 충돌 회피 가능도로서의 세 가지 조건을 모두 만족한다. 즉 로봇들의 외향속도인 $v_{j,k}(t)$ 와 $v_{k,j}(t)$ 각각의 운동성을 따로 고려해 줄 수 있게 되어 기존의 가상거리함수의 단점을 보완하게 되었음을 알 수 있다.

만일 로봇 j 의 외향 속도가 0 이고 로봇 k 의 외향속도가 0 인 경우 가상거리는 실제 거리와 비례한 값을 가지게 된다. 만일 $\alpha = \beta$ 이고 α 와 β 가 각각의 외향속도를 무시할 수 있을 만큼 외향속도의 최대값 보다 충분히 큰 값일 경우 각각의 외향속도는 가상거리함수에 영향을 미치지 못하고 $vd_{j,k}(t) \cong d_{j,k}(t)$ 가 되어 로봇 사이의 운동성을 고려하지 못하게 된다.

가상거리 $vd_{j,k}(t)$가 일정한 값 이하일 때부터 로봇이 장애물 회피동작을 시작한다고 가정하면 α와 β가 외향속도를 민감하게 고려 할 만큼 크지 않은 수이면 로봇은 실제로 먼 거리에 떨어져 있을 때부터 회피 동작을 시작하고 그 반대의 경우이면 이면 로봇은 장애물에 근접하여야 회피 동작을 시작하게 된다. 따라서 α와 β는 각각의 사항들을 고려하여 각각의 문제에 대하여 적절한 값으로 선정되어야 한다.

4. 인공 전위계의 적용

인공 전위계는 로봇들 사이에는 척력을 발생시키고 로봇과 목표점 사이에는 인력을 발생시키는 가상역장[12](Virtual Force Field)이다. 로봇을 이 역장에서 발생하는 힘에 따라 이동하게 하여 로봇들과의 충돌을 피하게 한다. 이러한 역장을 이용하여 목적에 따라 여러 가지 인공 전위계 함수가 제안되었다. 그러나 이들의 경우 인공 전위계에 장애물 혹은 로봇의 이동성이 고려되어 있지 못하다. 그러므로 본 논문 에서는 로봇들의 이동성이 고려되어 있는 가상 거리 함수 $vd_{j,k}(t)$를 가상역장에 도입하여 Khatib가 제안한 인공 전위계를 이동 장애물 회피에 적용할 수 있도록 개선한다.

$$\begin{aligned} &Uart(P_j(t),\ P_k(t),\ P_g) \\ &= U_k(P_j(t),\ P_k(t) + U_g(P_j(t),\ P_g) \end{aligned} \tag{2-14}$$

$$U_k(P_j(t),\ P_k(t)) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{vd_{k,j}(t)} - \frac{1}{\varepsilon_{vd}}\right)^2 & \text{, if } vd_{k,j}(t) \le \varepsilon_{vd} \\ 0 & \text{, if } vd_{k,j}(t) > \varepsilon_{vd} \end{cases} \tag{2-15}$$

$$U_g(P_j(t), P_{jg}) = \frac{1}{2}\zeta \mid P_j(t) - P_{jg} \mid^2 \qquad (2\text{-}16)$$

위 식에서 $U_k(P_j(t), P_k(t))$는 $P_j(t)$에 위치한 로봇 j 를 로봇 k 로부터 밀어내는 척력을 발생시키는 인공 전위계이고, $U_g(P_j(t), P_g)$는 $P_j(t)$에 위치한 로봇 j를 목표점을 끌어당기는 인력을 발생시키는 인공 전위계이다. ε_{vd} 는 $U_k(P_j(t), P_k(t))$ 가 영향을 미치는 장애물로부터의 가상 거리 범위이다. 다시 말하면 ε_{vd} 이상의 범위에서는 전위계가 발생하지 않는다. 또한 ε_{vd}가 크면 클수록 $U_o(P_j(t), P_k(t))$가 더 넓은 영역에 영향을 미치게 되어 로봇 j 가 로봇 k 로부터 더 먼 거리에서부터 장애물 회피 동작을 시작하게 된다. η 와 ζ는 $U_o(P_j(t), P_k(t))$ 와 $U_o(P_j(t), P_g)$의 계수로서 η이 커지면 로봇 k에 대한 척력이 커지게 되고, ξ가 커지면 목표점에 대한 인력이 커지게 된다.

인공 전위계에 의해 $P_j(t)$에 위치한 로봇에 발생되는 힘은 인공 전위계의 그래디언트(Gradient)로부터 구해진다.

$$\begin{aligned}
&F_{art}(P_j(t)) \\
&= -grad(U_{art}(P, P_k(t), P_g)) \mid_{P=Pj(t)} \\
&= -grad(U_o(P, P_k(t)) \mid_{P=Pj(t)} \\
&\quad -grad(U_o(P, P_g(t)) \mid_{P=Pj(t)} \\
&= F_k(P_j(t)) + F_g(P_j(t)) \qquad (2\text{-}17)
\end{aligned}$$

장애물로부터의 척력 $F_k(P_j(t))$는 로봇이 다른 로봇에 접근 할수록 로봇을 밀어내는 방향으로 커지고, 목표점으로의 인력 $F_g(P_j(t))$는 로봇이 목표점에 접근 할수

록 작아진다. 따라서 인공 전위계에서 발생되는 힘을 바탕으로 로봇에 발생하는 힘을 계산하여 로봇이 자연스럽게 목표점으로 이동하도록 한다.

제 3 장 로봇 및 센서 시스템

제 1 절 *Pioneer*Ⅱ 시스템

PioneerⅡ는 산업용 컴퓨터와 초음파 그리고 접촉식 감지 모듈을 포함한 통합된 자율 이동 로봇 시스템이다. 이 제어 시스템은 센서와 모터제어 뿐만 아니라 통신까지도 수행한다. PioneerⅡ의 마이크로컨롤러는 32K FLASH-ROM과 32K DRAM이 내장된 20MHz Simens 88C166 마이크로프로세서에 의해 제어며 사용자의 작업환경이나 목적에 따라 내부에 설치된 마이크로컨트롤러로 제어하거나 라디오 모뎀이나 블루투스를 이용한 무선 중앙제어 시스템도 가능하다.

PioneerⅡ의 내부 블록 다이어그램은 그림 3.1에 나타내었다.

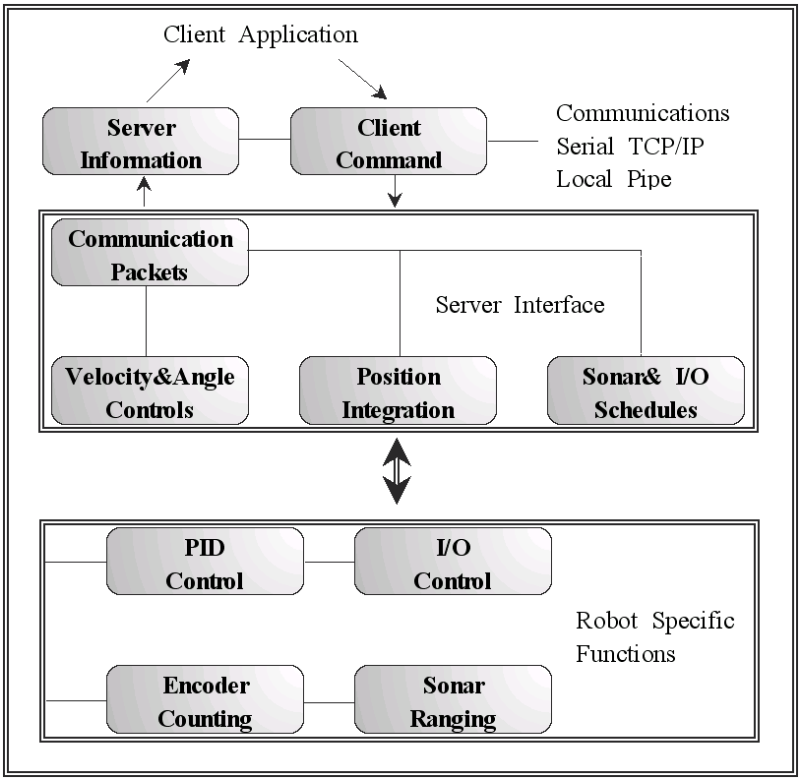

Fig. 3.1 Client-server control architecture

1. 하드웨어

실제 본 논문에서 사용된 자율 주행 로봇 시스템인 PioneerⅡ의 외형과 구조는 그림 3.2와 같다.

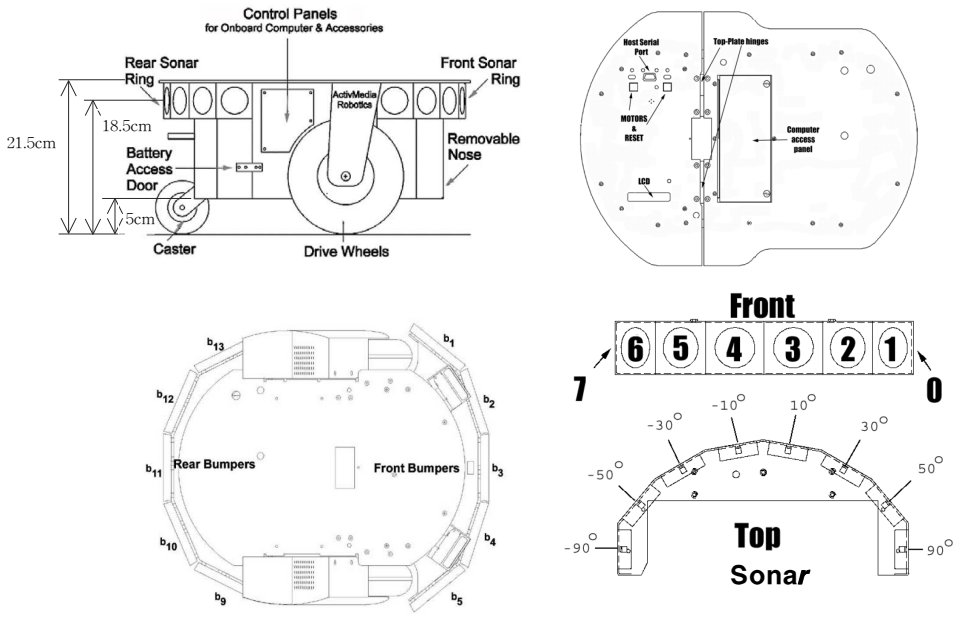

Fig. 3.2 PioneerⅡ's physical dimensions and sonar array

DC 12V 축전지 두 개를 병렬로 연결하여 12V의 전원으로 전체 시스템이 동작한다. 두 개의 DC모터를 사용하고 한 개의 보조 바퀴가 부착된 형태이다. 센서는 그림의 중심점(center point)에 부착되도록 설계되었다. 양쪽 바퀴의 직경은 195mm이며 외부 디바이스와의 통신을 위해 RS232 시리얼 통신 포트가 두 개 장착되어 있다. 한 개의 시리얼 포트는 내부적인 통신을 수행하고, 나머지 한 개는 외부에 장착된 SICK LMS 센서와 연결 되어있다.

2. 프로그램 흐름도

PioneerⅡ의 제어 프로그램 방법에는 직접 제어 모드와 클라이언트 제어 모드로 두 가지로 구분된다.

직접 제어 모드는 사용자 프로그램이 직접적으로 로봇 데몬(daemon)과 통신하여 로봇 제어용 프로그램으로 사용하는 방법이다. 이 프로그램은 로봇이 지속적으로 외부에서 데이터를 받아서 이것들을 실행하고, 데이터를 다시 전송하는 방법으로 동작한다.

또한, 클라이언트 제어 모드는 사용자 프로그램의 명령어를 서버가 받아서 서버가 로봇 데몬에 명령어를 전송하는 방법으로 사용자 프로그램을 가상 로봇의 시뮬레이터에서 테스트나 디버깅을 할 수 있고, 정확한 프로그램일 때 실제 로봇에 사용하는 방법이다.

PioneerⅡ의 프로그램 모드에 관한 내용을 그림 3.3에 나타내었다.

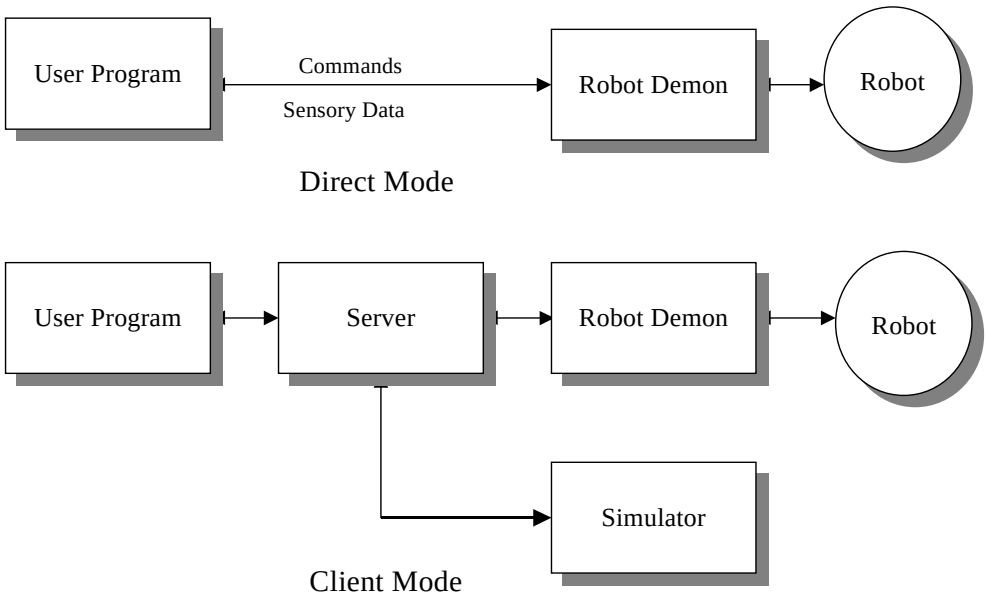

Fig. 3.3 Direct Mode and Client Mode

3. 환경 설정

가. 부트모드 설정

(1) FLASH 스위치를 로봇의 드라이버를 이용해 전방향으로 이동시킨다.

(2) MOTOR 버튼을 ON 상태로 유지시키면서 RESET버튼을 ON→OFF 한다.

(3) 마지막으로 MOTOR버튼을 3초이상 ON상태로 유지한 후에 OFF 한다.

나. *p2os*파일을 *C:*에 복사한다.

다. *MS-Dos*를 실행하고 *CD*명령을 사용해 *p2os*가 있는 디렉토리로 이동한다.

라. *p2oscf.exe*파일 실행*(*실제 실행화면은 그림 *3.4*과 같다*.)*

```
c:\>cd p2os
c:\p2os>p2oscf.exe

Opening serial port COM1
Sending 187 byte boot loader
Monitor is 1765 bytes
Monitor loaded!
% Serial port, boot loader and memory bus OK.

Retrieving parameters from flash ROM.....                    Pioneer
============== P2OS Constants ==================             P2PP
that you shouldn't touch...                                  BDBB1533
Type                                                         0
Subtype                                                      360
Serial Number                                                2200
FourMotors 0=no, 1=yes                                       360
RotVelTop deg/sec                                            4000
TransVelTop mm/sec                                           500
RotAccTop deg/sec/sec                                        128
TransAccTop mm/sec/sec
PwmMax 0-500 pwm counts
Encoder counts/mm
============= P2OS Variables ===================
KEYWORD                                                      CURRENT VALUES
Name                                                         Boo-Ki_1533
SInfoCycle 0=100ms, 1=50ms                                   0
HostBaud 0=9600, 1=19200, 2=38400                            2
```

AuxBaud 0=9600, 1=19200, 2=38400	0
HasGripper 0=none, 1=userio 2=genio	0
FrontSonars 0=none, 1=installed	1
RearSonars 0=none, 1=installed	1
AddedSonars 0=none, 1=8 sonars, 2=16 sonars	0
LowBattery 1/10 volt increments	110
WatchDog ms	2000
RevCount	33500
P2Mpacs 0=classic, 1=new	0
StallVal 0-500, never if > pwmmax	400
StallCount 10ms increments	100
Compass 0=none, 1=V2X, 2=TCM2	0
CompX, compass X cal. offset	0
CompY, compass Y cal. offset	0
RotVelMax deg/sec	100
TransVelMax mm/sec	500
RotAcc deg/sec/sec	50
RotDecel deg/sec/sec	100
RotKp	30
RotKv	60
RotKi	0
TransAcc mm/sec/sec	100
TransDecel mm/sec/sec	200
TransKp	40
TransKv	80
TransKi	0
JoyVelMax mm/sec	600
JoyRVelMax deg/sec	125
======== P2OS Config Commands =================	
Type: keyword alone to view current value	
keyword and new value to change it	

```
    'c' or 'constants' to view P20S constants
    'v' or 'variables' for current variables
    'a' or 'arm' to for current arm values
    'r' or 'restore' to restore original values
    'q' or 'quit' to exit *without* saving
    'save' to save changes and exit
    '?' or 'help' to see this menu again
command> HostBaud 0
HostBaud is 9600.
command> save
Comparing params with board FLASH. One moment, please...
Saving changes to flash...% Flash bank 3 cleared
% Flash bank 3 erased.
Writing parameters....
Saved to flash...
command> q
Comparing params with board FLASH. One moment, please...

Press <Enter> to quit...
C:\p2os>
```

Fig. 3.4 PioneerⅡ Reconfiguration

4. Update

가. 부트모드 설정

나. *p2os*파일을 *C:*에 복사한다.

다. *MS-Dos*를 실행하고 *CD*명령을 사용해 *p2os*가 있는 디렉토리로 이동한다.

라. *C:\p2os>p2osdl.exe p2os1_P.hex*(실제 실행화면은 그림 *3.5*와 같다.)

```
C:\p2os>p2osdl.exe p2os1_P.hex
% Loading Intel hex file p2os1_P.hex
Read p2os1_P.hex successfully
```

```
% 949 32-byte blocks
Opening serial port COM1
Sending 187 byte boot loader
Monitor is 1765 bytes
Monitor loaded!
% Serial port, boot loader and memory bus OK.
Syscon is 043f
FLASH bank 0: data
FLASH bank 1: data
FLASH bank 2: data
FLASH bank 3: data

Retrieving parameters from FLASH BANK3.....

Downloading flash data...
% Downloading 384 blocks to FLASH bank 0
% Flash bank 0 cleared
% Flash bank 0 erased.
.......................................................................
% Downloading 382 blocks to FLASH bank 1
% Flash bank 1 cleared
% Flash bank 1 erased.
.......................................................................
% Downloading 178 blocks to FLASH bank 2
% Flash bank 2 cleared
% Flash bank 2 erased.
.......................................................................
```

```
No parameters to update or add. Skipping Bank 3.


Press <Enter> to quit...


C:\p2os>
```

Fig. 3.5 Updating PioneerⅡ

5. ARIA Demo Program

가. ActivMedia Robotics의 Aria프로그램설치

나. MS-DOS 실행

다. Demo프로그램 실행(그림 3.6)

```
C:\Program Files\ActivMedia Robotics\Aria\bin>demo -h

Options for ArSimpleConnector (see docs for more details):

Robot options
-remoteHost <remoteHostNameOrIP>
-rh <remoteHostNameOrIP>
-robotPort <robotSerialPort>
-rp <robotSerialPort>
-robotBaud <baud>
-rb <baud>
-remoteRobotTcpPort <remoteRobotTcpPort>
-rrtp <remoteRobotTcpPort>
-remoteIsSim
-ris
```

Laser options
-connectLaser
-cl
-laserPort <laserSerialPort>
-lp <laserSerialPort>
-remoteLaserTcpPort <remoteLaserTcpPort>
-rltp <remoteLaserTcpPort>
-laserFlipped <true|false>
-lf <true|false>
-laserPowerControlled <true|false>
-lpc <true|false>
-laserDegrees <180|100>
-ld <180|100>
-laserIncrement <one|half>
-li <one|half>

EX) 보레이트 설정법

C:\Program Files\ActivMedia Robotics\Aria\bin>demo -robotBaud 38400

You may press escape to exit

Could not connect to simulator, connecting to robot through serial port COM1.

Syncing 0

Syncing 1

Syncing 2

Connected to robot.

Name: Boo-Ki_1533

Type: Pioneer

Subtype: p2pp

Loaded robot parameters from p2pp.p

ArACTS_1_2::openPort: Open failed: Connection refused.

You can do these actions with these keys:

quit: escape

You can switch to other modes with these keys:

help:	'h' or 'H' or '?' or '/'
tcm2 mode:	'm' or 'M'
command mode:	'd' or 'D'
acts mode:	'a' or 'A'
io mode:	'i' or 'I'
position mode:	'p' or 'P'
bumps mode:	'b' or 'B'
sonar mode:	's' or 'S'
camera mode:	'c' or 'C'
gripper mode:	'g' or 'G'
wander mode:	'w' or 'W'
unguarded teleop mode:	'u' or 'U'
teleop mode:	't' or 'T'
laser mode:	'l' or 'L'

You are in 'teleop' mode currently.

Teleop mode will drive under your joystick or keyboard control.

It will not allow you to drive into obstacles it can see,

though if you are presistent you may be able to run into something.

For joystick, hold in the trigger button and then move the joystick to drive.

```
For keyboard control these are the keys and their actions:
up arrow:  speed up if forward or no motion, slow down if going backwards
down arrow:  slow down if going forwards, speed up if backward or no
motion
left arrow:  turn left
right arrow:  turn right
space bar:  stop
transVel      rotVel      x      y      th      volts
Gripper:  querried, the robot has no gripper.
    0          0          0      0      0.0      13.5Warning: Task '
teleop' took 4946 ms to run (longer than the 250 warning time)
Warning: ArRobot sync tasks too long at 5047 ms, (100 ms normal 250 ms
warning)
    0          0          0      0      0.0      13.5Warning: Task '
You can do these actions with these keys:
quit: escape
help: 'h' or 'H' or '?' or '/'
You can switch to other modes with these keys:
                        tcm2 mode: 'm' or 'M'
                        command mode: 'd' or 'D'
                        acts mode: 'a' or 'A'
                        io mode: 'i' or 'I'
                        position mode: 'p' or 'P'
                        bumps mode: 'b' or 'B'
                        sonar mode: 's' or 'S'
                        camera mode: 'c' or 'C'
```

```
                    gripper mode: 'g' or 'G'
                    wander mode: 'w' or 'W'
                    unguarded teleop mode: 'u' or 'U'
                    teleop mode: 't' or 'T'
                    laser mode: 'l' or 'L'
You are in 'acts' mode currently.


ACTS mode will drive the robot in an attempt to follow a color blob.
          1 - 8      :  Pick a channel
          'x' or 'X' :  toggle acquire mode
          'z' or 'Z' :  start movement
          space bar  :  stop movement
```

Fig. 3.6 ARIA Demo Program

6. Communication Packet Protocol

Table 3.1 PioneerⅡ Packet Protocol

Component	*Bytes*	*Value*	*Description*
Header	2	0xFA, 0xFB	Packet header; same for client and server
Byte Count	1	N + 2	Number of subsequent data bytes, including the Checksum word, but not the Byte Count. Maximum 200 bytes.
Data	N	command or SIP	Client command or server information packet (SIP)
Checksum	2	computed	Packet integrity checksum

P2OS는 표 3.1와 같이 특별한 명령 패킷을 가지고 클라이언트가 서버에 명령을 내리거나 주어진 명령어에 대한 서버의 상태정보를 클라이언트에게 제공한다.

P2OS는 클라이언트로부터 명령을 받으면 자동으로 호스트 시리얼포트를 통해 패킷 정보를 매 100ms마다 접속되어 있는 클라이언트에게 보내게 된다. 서버패킷은 현재 로봇의 위치, Sonar, Bumper등의 정보를 표 3.2의 명령형식에 맞게 전송한

다.

Table 3.2 Server Information Packet

Name	*Data Type*	*Description*
Header	integer	Exactly 0xFA, 0xFB
Byte Count	byte	Number of data bytes + 2 (checksum); must be less than 201 (0xC9)
Status/Packet Type	byte = 0x3S; where S =	Motors status
	2	Motors stopped
	3	Robot moving
Xpos	unsigned integer (15 ls-bits)	Wheel-encoder integrated coordinates; platform-dependent units; multiply by DistConvFactor‡ to convert to millimeters.
Ypos	unsigned integer (15 ls-bits)	
Th pos	signed integer	Orientation in platform-dependent units－multiply by AngleConvFactor‡ for degrees.
L vel	signed integer	Wheel velocities (respectively Left and Right) in platform-dependent units; multiply by VelConvFactor‡ －currently 1.0 for all－to convert into millimeters per second.
R vel	signed integer	
Battery	byte	Battery charge in tenths of volts
Stall and Bumpers	integer	Motor stall and bumper accessory indicators. Bit 0 of the lsbyte is the left wheel stall indicator = 1 if stalled. Bits 1-5 of that same byte correspond to the bump switch states (1=on) for the rear bumpers accessory. Bit 0 of the msbyte is the right wheel stall; the bits 1-5 of that same msbyte correspond to the front bumpers switch states.
Control	signed integer	Setpoint of the server's angular position servo－multiply by AngleConvFactor‡ for degrees
FLAGS (was PTU)	unsigned integer	b0 - motors flag (1=motors enabled) b1 - sonar flag: enabled if 1.
Compass	byte	Compass heading in 2-degree units
Sonar	byte	Number of new sonar readings included in

readings		information packet; readings follow:
Sonar number	byte	Sonar number
Sonar range	unsigned integer	Sonar range; multiply by RangeConvFactor‡ −currently 0.268 for all−for millimeters
…rest of the sonar readings…		
Timer	unsigned int	Selected analog port number 1-5
Analog	byte	User Analog input (0-255=0-5 VDC) reading on selected port
Digin	byte	User I/O digital input
Digout	byte	User I/O digital output
Checksum	integer	Checksum (see previous section)

7. 클라이언트 명령코드

클라이언트는 반드시 명령패킷(표 3.3)을 최소 Watchdog(default : 2sec)시간내에 전송해야하며 그렇지 않으면 로봇은 자동으로 멈추게 된다.

로봇의 연속적인 동작이나 클라이언트의 원하는 동작 수행을 위해서는 최소 명령 사이클 시간 이내에 계속적인 명령이 전송되어야하고 클라이언트 명령패킷에 맞지 않는 명령어가 전송 될 경우 로봇은 잘못된 작업 수행을 하거나 동작을 멈추게 된다. 클라이언트 명령 전송 방법은 Header, 전송해야 할 명령수(Checksum을 포함), 클라이언트의 명령넘버(표 3.4), 전송되는 명령의 데이터형 그리고 이들 데이터에 대한 Checksum을 구한 후 하나의 명령패킷(표 3.2)으로 서버에 전송한다.

Table 3.3 Client command packet

Component	*Bytes*	*Value*	*Description*
Header	2	0xFA, 0xFB	Packet header; same for client and server
Byte Count	1	N + 2	Number of following command bytes plus Checksum's two bytes, but not including Byte Count. Maximum of 200.
Command Number	1	0 - 255	Client command number; see Table 4-4
Argument Type	1	0x3B or	Required data type of command argument:

(command dependent)		0x1B or 0x2B	positive integer (sfARGINT), negative integer or absolute value (sfARGNINT), or string (sfARGSTR)
Argument (command dependent)	n	data	Command argument; integer or string
Checksum	2	computed	Packet integrity checksum

Table 3.4 Client command set

Command	*Dec*	*Args*	*Description*	*PSOS*	*P2OS*
			Before Client Connection		
SYNC0	0	none	Start connection; P2OS echoes synchronization commands back to client.	3.x	1.0
SYNC1	1	none			
SYNC2	2	none			
After Established Connection					
PULSE	0	none	Client pulse resets server watchdog	3.x	1.0
OPEN	1	none	Starts the controller	3.x	1.0
CLOSE	2	none	Close server and client connection	3.x	1.0
POLLING	3	string	Set sonar polling sequence	3.9	1.0
ENABLE	4	int	Enable=1; disable=0 the motors	-	1.0
SETA	5	signed int	Resets translational acceleration parameter, if positive, or deceleration, if negative; in millimeters per second2	-	1.0
SETV	6	int	Reset maximum translational velocity, in millimeters per second	4.8	1.0
SETO	7	none	Resets server to 0,0,0 origin	3.x	1.0
MOVE	8	signed int	Translation distance to move in mm		
SETRV	10	int	Resets maximum rotational velocity in degrees per second	4.8	1.0
VEL	11	signed int	Move forward (+) or reverse (-) at millimeters per second	3.x	1.0
HEAD	12	signed int	Turn to absolute heading (+) = counterclockwise; ± degrees	4.2	1.0
DHEAD	13	signed int	Turn relative to current heading (+) = counterclockwise; ± degrees	3.x	1.0
SAY	15	string	As many as 20 pairs of duration (20 ms increments) /tone (half-cycle) pairs; int is string length	4.2	1.0
CONFIG	18	int	1=Request configuration SIP	-	1.4

ENCODER	19	int	Request 1 or continuous stream (>1), or tell to stop sending (0) Encoder SIPs	-	1.4
RVEL	21	signed int	Rotate at ± degrees per second	4.2	1.0
DCHEAD	22	signed int	Heading setpoint relative to last setpoint; ± degrees; (+) = counterclockwise		
SETRA	23	signed int	Sets rotational (+)acceleration or (-)deceleration, in mm/sec/sec	-	1.0
SONAR	28	int	1=Enable; 0=disable all the sonar	-	1.0
STOP	29	none	Stops robot (motors remain enabled)	-	1.0
DIGOUT	30	int	Msbits is a byte mask that selects output port(s) for changes; lsbits set (1) or reset (0) the selected port.	4.2	1.2
VEL2	32	signed int	Independent wheel velocities; lsb=right wheel; msb=left wheel; PSOS is in ±4mm/sec; P2OS in ±2cm/sec increments	4.1	1.0
GRIPPER	33	int	Pioneer 1 and Pioneer 2 Gripper server command. See the Pioneer Gripper manuals for details.	4.0	1.3
ADSEL	35	int	Select the A/D port number for analog value in SIP. Selected port reported in SIP Timer value.	-	1.2
GRIPPER VAL	36	int	Pioneer 2 gripper server value. See P2 Gripper Manual for details.	-	
GRIP REQUEST	37	none	Request 1 or continuous stream (>1), or tell to stop sending (0) Gripper SIPs	-	1.E
IO REQUEST	40	none	Request 1 or a continuous stream (>1) or tell to stop sending (0) IO SIPs	-	1.E
PTUPOS	41	int	msb is the port number (1-4) and lsb is the pulse width in 100μsec units PSOS, 10μsec units P2OS. Version 1.J uses RC-servo 40ms duty cycle.	4.5	1.2 - 1.J
TTY2	42	string	Send string argument to serial device connected to AUX port on microcontroller	4.2	1.0
GETAUX	43	int	Request to retrieve 1-200 bytes from the aux serial channel; 0 flushes the aux serial input buffer.	-	1.4
BUMP_STALL	44	int	Stop and register a stall if front (1), rear (2) or either (3) bump-ring contacted and robot motion is in direction of bump.	-	1.5
TCM2	45	int	TCM2 Module commands; see TCM2 Manual for details.	-	1.6
DOCK	46	int	Default is 0=OFF; 1=enable docking signals; 2=enable docking signals and stop the robot when docking power sensed.	-	1.C
JOYDRIVE	47	int	Default is O=OFF; 1=allow joystick drive from hardware port	-	1.G

E_STOP	55	none	Emergency stop, overrides deceleration	-	1.8
E_STALL	56	int	Emergency stop button causes stall	-	1.E
STEP	64	none	Single-step mode (simulator only)	3.x	1.0
ARM	70 - 80	-	Please consult the Pioneer 2 Arm Manual for details.	-	1.H
ROTKP	82	int	Change working rotation proportional PID drive factor (not FLASH default)	-	1.M
ROTKV	83	int	Change working rotation's derivative PID drive factor (not FLASH default)	-	1.M
ROTKI	84	int	Change rotation's integral PID drive factor (not FLASH default)	-	1.M
TRANSKP	85	int	Change working translation proportional PID drive factor (not FLASH default)	-	1.M
TRANSKV	86	int	Change working translation derivative PID drive factor (not FLASH default)	-	1.M
TRANSKI	87	int	Change working translation integral PID drive factor (not FLASH default)	-	1.M
REV COUNT	88	int	Change working revcount (not FLASH default)		1.M
PLAYLIST	90	int	Must be 0; request AmigoBot sound playlist	-	1.E

8. Pioneer Programming

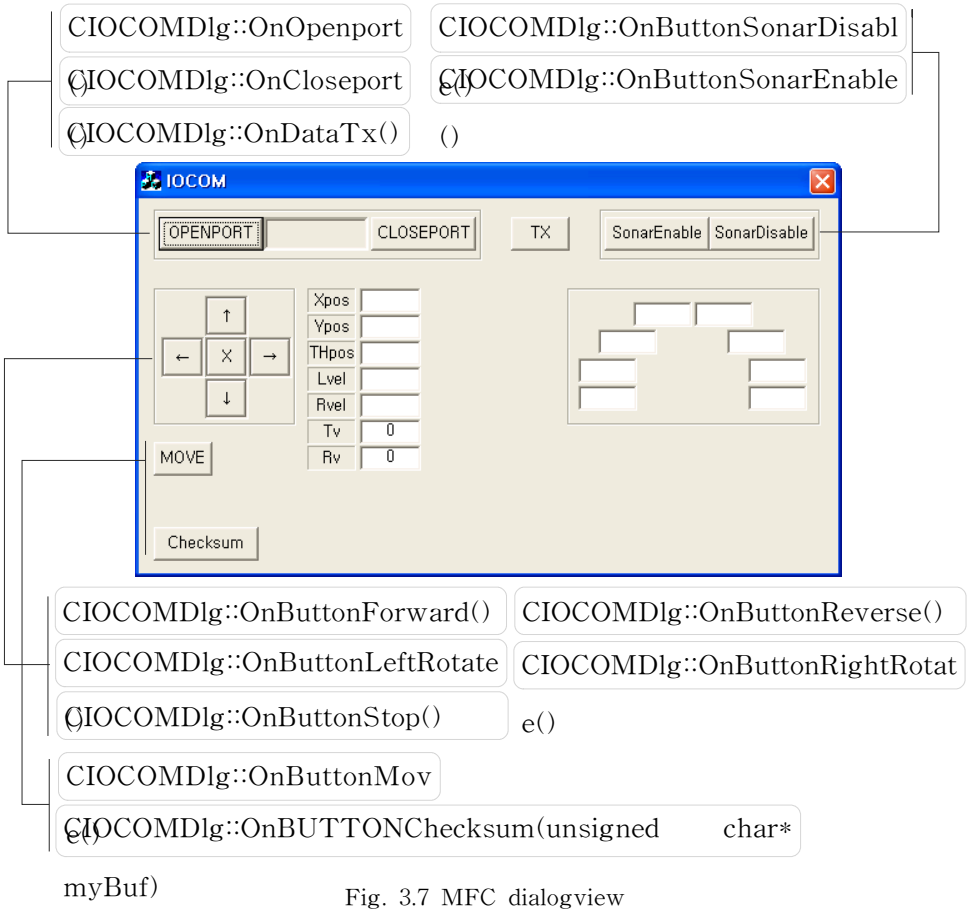

Fig. 3.7 MFC dialogview

가. *Pioneer*와 사용자 프로그램 *Baudrate* 설정

Pioneer와 클라이언트 사용자 프로그램(VC++)이 데이터 교환을 할 수 있도록 앞서 설명한 환경설정 방법을 이용하여 Baudrate을 맞춘다. VC++의 프로그램상에서 Baudrate을 환경설정부분에서 설정 했던 것과 같이 성정한다.

```
void CIOCOMDlg::OnOpenport()
{
        // TODO: Add your control notification handler code here

        BOOL bret;
```

```
m_port.setSendingMode(COMM_WITH_PC);
m_port.clearPort();
Sleep(10);
bret = m_port.openPort(1, this, CBR_38400, 8, NOPARITY,
ONESTOPBIT );
```

※Pioneer가 38400으로 설정되어 있으므로 프로그램상의 Baudrate도 같게 설정

나. *Pioneer*와 *Program*의 동기화 설정

Pioneer와 program간의 Baudrate가 끝나면 Pioneer를 제어하기 위한 기본 설정을 해야 한다. 즉, Pioneer의 각 센서부, 구동부, 데이터 수신방법 등을 설정하여 원하는 데이터를 쉽게 사용자가 사용할 수 있도록 하는 것이다.

```
void CIOCOMDlg::OnDataTx()
{
    // TODO: Add your control notification handler code here

    int i = 0;
    BYTE aByte[] = {0x57, 0x4D, 0x53, 0x32, 0x0D,
                    0xFA, 0xFB, 0x03, 0x00, 0x00, 0x00,
                    0xFA, 0xFB, 0x03, 0x01, 0x00, 0x01,
                    0xFA, 0xFB, 0x03, 0x02, 0x00, 0x02};
    i = sizeof(aByte);
    m_port.sendBytes(aByte, i);
```

※ 사용자프로그램과 pioneer 주기적인 데이터 교환을 위해 동기화를 시켜주는 부분이다. 로봇의 전/후, 좌/우, Sonar, Bumper 등 Pioneer의 모든 명령은 동기화를 시켜준 후에 사용할 수 있다.

다. *Pioneer Controller Open*과 상태 데이터 요구 설정

BYTE aByte3[] = {0xFA, 0xFB, 0x06, 0x01, 0x3B, 0x01, 0x00, 0x02, 0x3B

//Starts the controller(#1)

※ Pioneer의 컨트롤러를 오픈하여 사용자가 제어할 수 있도록 설정

, 0xFA, 0xFB, 0x06, 0x12, 0x3B, 0x01, 0x00, 0x13, 0x3B};

//1=Request configuration SIP(Server Information Packets)(#18)

※ 로봇의 상태정보를 주기적으로 송신하라는 명령. Pioneer는 이 명령을 받으면 사용자에게 현재 상태정보를 계속해서 보내게 된다.

i = sizeof(aByte3);

m_port.sendBytes(aByte3, i);

라. *Pioneer* 각 부분에 대한 사용 설정

사용자의 원하는 제어를 위해서 Motors, Sonars, Joystic 등의 사용여부를 결정해 주어야 한다. 그래야만 SIP를 요구 할 경우 불필요한 데이터가 수신되지 않는다.

BYTE aByte4[] = {//0xFA, 0xFB, 0x06, 0x11, 0x3B, 0x02, 0x00, 0x13, 0x3B

※ Joystic Request(#17), 조이스틱을 사용하도록 설정.

, 0xFA, 0xFB, 0x06, 0x25, 0x3B, 0x02, 0x00, 0x27, 0x3B

※ Gripper Quest(#37), 그리퍼를 사용하도록 설정

,0xFA, 0xFB, 0x06, 0x0B, 0x3B, 0x00, 0x00, 0x0B, 0x3B

※ VEL Move forward or reverse(#11), 로봇이 전/후진 정지

, 0xFA, 0xFB, 0x06, 0x15, 0x3B, 0x00, 0x00, 0x15, 0x3B

※ thRvel(#21), 로봇의 좌/우회전 정지

, 0xFA, 0xFB, 0x06, 0x04, 0x3B, 0x01, 0x00, 0x05, 0x3B

※ Enable=1, disable=0 the motors(#4) 모터제어를 위한 사용 설정.

, 0xFA, 0xFB, 0x06, 0x1C, 0x3B, 0x01, 0x00, 0x1D, 0x3B};

※ SONAR 1=enable, 0=disable, 이 명령을 야만 Sonar에 대한 데이터를 받을 수 있다. disable로 설정될 경우 SIP는 Sonar의 데이터를 송신패킷에 포함하지

않는다.

```
i = sizeof(aByte4);
m_port.sendBytes(aByte4, i);
```

※ 이것으로 Pioneer를 제어하기 위한 기본설정을 모두 끝났다. 다음은 Pioneer의 전/후, 좌/우 명령이나 Sonar의 ON/OFF, 에 대해 설명한다.

마. 전/후진

```
BYTE aByte[] = {0xFA, 0xFB, 0x06, 0x0B, 0x3B, 0x00, 0x01, 0x0B, 0x3C};
        //VEL Move forward or reverse(#11)
        i = sizeof(aByte);
        m_port.sendBytes(aByte, i);
```

0xFA, 0xFB : 로봇과 사용자의 데이터 송수신을 위한 정보패킷의 헤더

0x06 : 송신하고자 하는 데이터 수(Checksum포함)

0x0B :

0x3B : 데이터형(대부분0x3B를 사용함)

0x00 : 로봇속도설정(mm/s) — 하위바이트

0x01 : 로봇속도설정(mm/s) — 상위바이트

0x0B, 0x3C : Checksum

로봇 속도설정 부분에서 상위바이트의 MSB가 0이면 전진을 1이면 후진을 의미하며, 속도단위는 m/s로 되어 있다.

```
BYTE Move[] = {0xFA, 0xFB, 0x06, 0x08, 0x3B, 0x00, 0x02, 0x08, 0x3D};
        //Translation distance to move in mm
```

※ Move(0x08) 명령의 사용법은 위의 0x0B와 같지만 뒤의 로봇의 속도(m/s) 대신 거리(mm)를 쓴다.

바. *SONAR ON/OFF*

BYTE aByte1[] = {0xFA, 0xFB, 0x06, 0x1C, 0x3B, 0x01, 0x00, 0x1D, 0x3B};

//SONAR 0x01 LSB가 1=enable, 0=disable

0x1C는 LSB가 1이면 하위바이트의 LSB가 1(0x01)이면 Sonar가 On되어 거리 데이터를 출력하게 되고 LSB가 0(0x00)이면 Sonar가 Off되어 SIP에서는 쓰레기 데이터를 출력한다.

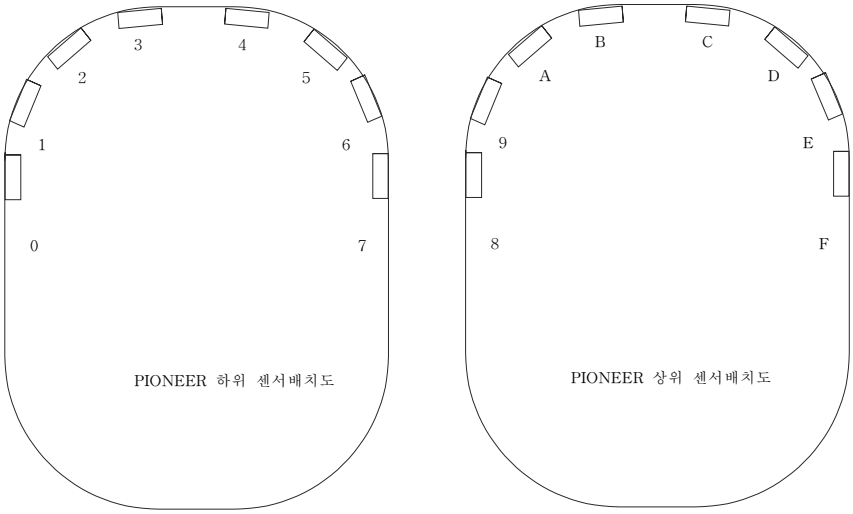

Fig. 3.8 Pioneer sonar array

논문에 사용된 Pioneer의 Sonar은 상위 8개, 하위 8개 총 16개로 되어있다.

사. 정지

BYTE aByte[] = {0xFA, 0xFB, 0x03, 0x1D, 0x00, 0x1D};

로봇 정지시 사용하는 명령어.

아. *Checksum*

PioneerⅡ의 설명서에 있는 Checksum은 프로그래밍시 실행되지 않아 http://robots.mobilerobots.com/의 최근 모델의 메뉴얼을 참고하여 만들었다.

```
void CIOCOMDlg::OnBUTTONChecksum()
{
```

```
        unsigned char myBuf[] = {0xFA, 0xFB, 0x06, 0x1C, 0x3B, 0x00, 0x00};
        int i;
        unsigned char n;
        int c = 0;
        i = 3;
        n = myBuf[2] - 2;
        while(n > 1)
        {
                c += ((unsigned char)myBuf[i]<<8) | (unsigned char)myBuf[i+1];
                c = c & 0xffff;
                n -= 2;
                i += 2;
        }
        if (n > 0)
                c = c ^ (int) ((unsigned char) myBuf[i]);
        CString str;
        str.Format("0x%x", c);
        AfxMessageBox(str);
}
```

자. *SIP* 데이터 분석

Pioneer는 정해진 순서에 따라 일정하고 연속적으로 데이터를 시리얼로 사용자 컴퓨터에 보내준다. 즉, 데이터의 각 부분에 대한 정보를 알지 못하면 수신된 데이터를 사용자의 의도나 목적에 맞게 사용하지 못하게 되는 것이다. 표 2.6, 표 2.7은 실제 Pioneer에서 송신하는 SIP패킷을 분석하여 나타내었다. SIP패킷의 크기가 다른 이유는 Sonar, Gripper, Motor, Laser, Bumper 등의 설정 유무에 따라서 패킷의

길이가 결정되기 때문이다. 표 3.5의 음영부분과 표 3.6을 비교해 보면 쉽게 이해할 수 있다.

Table 3.5 Pioneer TX data

추출 횟수	S I P
1	FA FB 1B 33 4D 05 FF FF FF 0F 90 00 90 00 87 00 00 FF 0F 01 00 00 00 05 00 54 DE F0 A3 2F
2	FA FB 1B 33 62 02 00 00 FF 0F 90 00 8F 00 87 00 00 FF 0F 01 00 00 00 05 00 54 DE F0 A0 04
3	FA FB 27 32 00 00 00 00 00 00 00 00 00 00 88 00 00 00 00 03 00 00 04 05 C1 1B 0D D0 7F 02 80 13 0A D0 7F 05 00 54 DE F0 66 30
4	FA FB 27 32 02 00 00 00 02 00 00 00 00 00 88 02 02 02 00 03 00 00 04 05 0A 0A 0D D0 7F 02 D0 7F 0A D0 7F 05 00 54 DE F0 C5 AF
5	FA FB 27 32 00 00 00 00 00 00 00 00 00 00 88 02 02 00 00 03 00 00 04 05 A1 08 0D D0 7F 02 D0 7F 0A D0 7F 05 00 54 DE F0 C1 02
6	FA FB 2D 32 00 00 00 00 00 00 00 00 00 00 88 00 02 00 00 03 00 00 06 03 50 03 0B D0 7F 06 4D 07 0E D0 7F 01 A9 0B 09 D0 7F 05 00 54 DE F0 21 A3
7	FA FB 2D 32 00 00 00 00 00 00 00 00 00 00 88 00 00 00 00 03 00 00 06 03 D0 7F 0B D0 7F 06 D0 7F 0E D0 7F 01 D0 7F 09 D0 7F 05 00 54 DE F0 8A 8B

Table 3.6 SIP(Server Information Packet) Analysis

Pioneer는 움직이고 Sonar가 작동하지 않은 경우										
Header	Byte Counter	Status/Packet Type	Xpos	Ypos	Th pos	L vel	R vel	Battery	Stall and Bumpers	Control
FA FB	1B	33	B3 05	FF FF	FF 0F	90 00	90 00	87	00 00	FF 0F
FLAGS	Compass	Sonar readings	Sonar number	Sonar range	Timer	Analog	Digin	Digout	CheckSum	
01	00 00	00	-	-	05	00 53	DE	F0	A3 B5	

Pioneer가 움직이지 않고 Sonar는 작동하고 있는 경우										
Header	Byte Counter	Status/Packet Type	Xpos	Ypos	Th pos	L vel	R vel	Battery	Stall and Bumpers	Control
FA FB	27	32	00 00	00 00	00 00	00 00	00 00	88	00 00	00 00
FLAGS	Compass	Sonar readings	Sonar number	Sonar range	Timer	Analog	Digin	Digout	CheckSum	
03	00 00	06	03 0B 06 0E 01 09	D0 7F D0 7F D0 7F D0 7F D0 7F D0 7F	05	00 54	DE	F0	8A 8B	

제 2 절 LMS 시스템

1. 주요데이터 전송

LMS200은 PC와 9600, 19200, 38400, 500000 baud rate으로 RS422또는 RS232 프로토콜 형태로 통신할 수 있다. 또한, 1개의 시작 비트와 1개의 정비비트, 8개의 데이터 비트로 이루어져 있으며, 패리티(parity) 비트는 없다.

STX	ADR	LENL	LENH	CMD	...	CRCL	CRCH

Fig. 3.8 LMS telegram structure

STX : 시작을 나타내는 HEX값-> 0X02

ADR : LMS의 주소 -> 0X00

LENL: 명령어 수 표시, 하위 8비트

LENH: 명령어 수 표시, 사위 8비트

CMD : 명령어(HEX),

CRCL: CRC프로그램에 CMD 명령어(HEX값)를 넣어 출력된 하위 8비트 HEX값

CRCH: CRC프로그램에 CMD 명령어(HEX값)를 넣어 출력된 상위 8비트 HEX값

2. Programming

가. LMS 초기설정

(1) 시리얼 포트 열기

sickPort.openPort(1, pWnd, CBR_9600, 8, NOPARITY, ONESTOPBIT) ;

(2) Baudrate를 원하를 속도로 변경한다.

9600 : 0x02, 0x00, 0x02, 0x00, 0x20, 0x42, 0x52, 0X08

19200 : 0x02, 0x00, 0x02, 0x00, 0x20, 0x41, 0x51, 0X08

38400 : 0x02, 0x00, 0x02, 0x00, 0x20, 0x40, 0x50, 0X08

(3) LMS 리셋

0x02, 0x00, 0x01, 0x00, 0x10, 0x34, 0X12

(4) 오퍼레이팅 모드

0x02, 0x00, 0x0A, 0x00, 0x20, 0x00, 0x53, 0X49, 0x43, 0x4B, 0x5F, 0x4C, 0x4D, 0x53, 0xBE, 0xC5 10개

*Default password = SICK_LMS

나. *LMS*의 센서 데이터를 요구

0x02, 0x00, 0x02, 0x00, 0x30, 0x01, 0x31, 0x18

위와 같은 데이터 명령을 LMS에 전송하게 되면 클라이언트는 계속해서 180° 측정된 데이터를 받을 수 있다.

다. 수신데이터 분석

06 02 81 03 00 A0 00 10 36 1A

06 02 81 : 수신시 항상 표시됨

03 : 수신 HEX값 하위 8비트

00 : 수신 HEX값 상위 8비트

00 A0 00 : 0x03에 해당되는 3개의 HEX값

Table 3.7 LMS telegram structure

Designation	Data [bits]	Description
STX	8	Start byte(02h)
ADR	8	Address of subscriber. 0:broadcast address(all subscribers addressed) 1-127 corresponding subscriber from n This address byte is intended for a bus connection, if present. The sensor adds the value 80h to the address when it responds to the AWE
Len	16	No. of following data bytes excluding CRC
CMD	8	Command byte from the AWE to the sensors. When the sensor responds to the AWE, it adds the value 80h to the original command
Data	n×8	Optional, depends on the previous command
Status	8	Optional, the sensor transmits its status message only when it transfers the data to the AWE. The AWE does not transmit a status message to the sensor
CRC	16	CRC checksum for the entire data package. Algorithm is described below.

LMS200의 데이터 형태는 그림 3.8과 표 3.7과 같이 이루어져 있다.

모든 전송되는 데이터는 STX(0x02)로 시작되고, 에러체크를 위한 CRC값으로 끝나게 되며 2바이트 이상의 모든 데이터는 INTEL방식으로 전송되게 된다. 즉, 상위 바이트와 하위 바이트가 바뀌어서 전송되게 된다. 호스트 컴퓨터가 보내는 명령이나 주소에 대한 응답으로 LMS는 0x80을 더한 값을 응답으로 보낸다. 즉, 0x00번지에 물려있는 LMS에 0x20의 명령을 보내게 되면, 0x80이라는 고유번지에 있는 LMS로부터 0xA0의 값이 전달되어 온다. Telegram에 있는 바이트 사이의 시간 간격은 55㎲-6㎳ 사이에 있어야 한다. 만약 모든 데이터가 정확하게 LMS에 전달되었다면, LMS는 정확하게 메시지를 받았다는 응답으로 ACK(0x60)의 메시지를 보내게 되고, 데이터에 에러가 있다면 NAK(0x92)값을 보내게 된다. 명령에 대한 응

답 메시지는 ACK값 뒤로 전달되게 된다. 즉, ACK + STX + Length + CMD + DATA + CRC와 같은 형태를 띄게 된다. CRC값은 ACK(0x06)의 신호를 제외한 STX부터 CRC값 전의 모든 바이트로 구해진다.

LMS의 제어 순서는 그림 3.9와 같다.

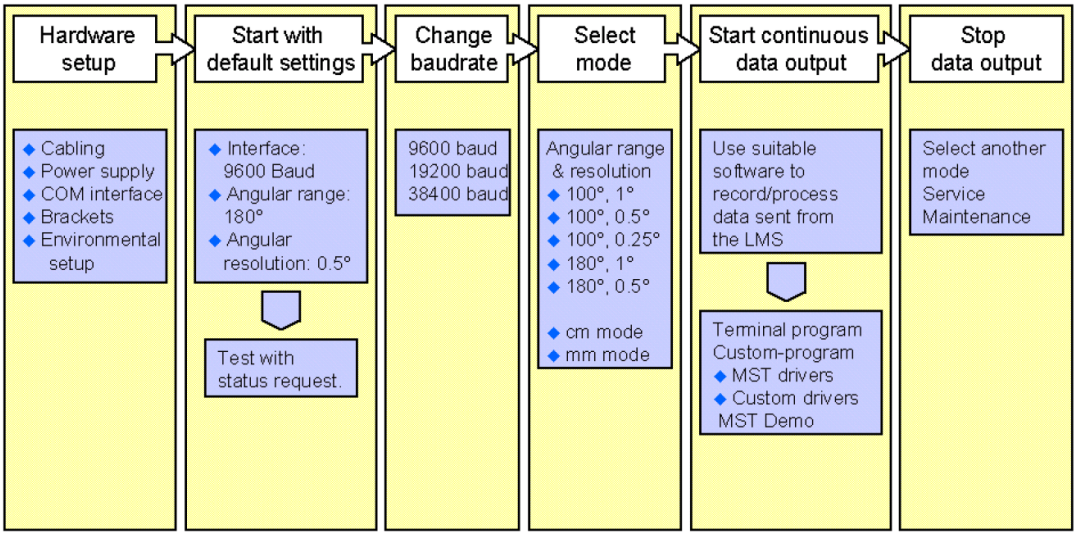

Fig. 3.9 Overview Schematic for LMS communication setup

제 3 절 *MOBILE ROBOT SIMULATOR*

1. 구성

로봇 시뮬레이터는 2차원 공간상에서 자율 이동로봇의 이동기능과 센서 시스템을 시뮬레이션 하였으며, 초기 로봇 응용프로그램 설계에서 알고리즘 개발과 성능 향상의 과정을 가상환경에서 로봇 하드웨어로는 불가능한 다양한 테스트 할 수 있는 장점이 있다.

본 논문에서는 VisualC++을 기반으로 개발된 프로그램을 사용하였으며, Controller, IPC, Simulator부로 구성 되어 있다.

가. *Controller*

컨트롤러는 시뮬레이터상의 로봇을 구동하기 위한 제어기이다. 프로그래머는 시뮬레이터로부터 현재 로봇위치, 장애물, 센서, Tv, Rv데이터를 받아서 원하는 제어를 시뮬레이션으로 구현할 수 있다.

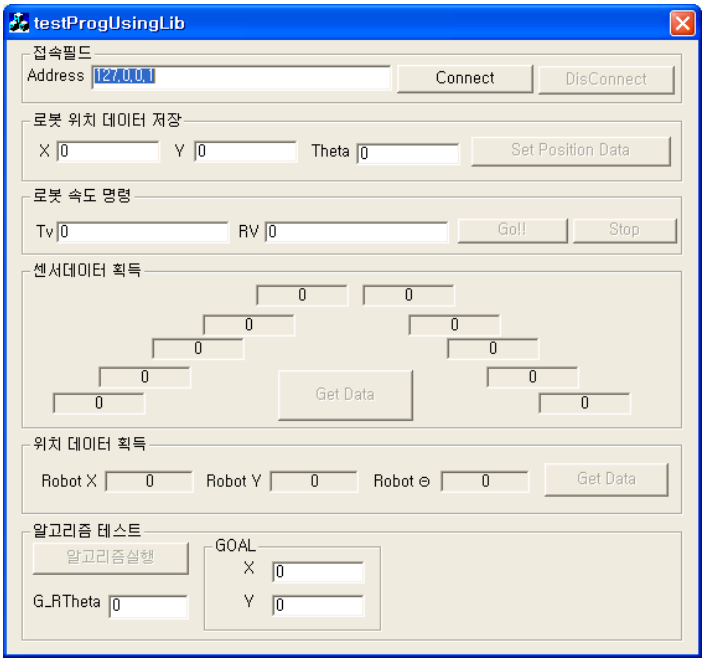

Fig. 3.10 Controller

(1) Connect : IPC와 커뮤니케이션을 할 수 있도록 한다. (시뮬레이터 역시 IPC에 Connect를 해야만 컨트롤러에서 송신하는 데이터를 받을 수 있다.)

(2) Set Position Data : 로봇의 현재 위치나 방향을 제어환경이나 제어 방법에 맞게 변환할 수 있도록 한다.

(3) Tv, Rv : 속도와 방향에 대한 값을 수동으로 주어 로봇의 움직임을 관찰하거나 진행 중인 로봇을 정지시킬 수 있다.

(4) 센서데이터 획득(Get Data) : 시뮬레이터에서 보내오는 센서데이터를 얻을 수 있다.(시뮬레이터에서 보내오는 센서신호는 180개이며, 컨트롤러에서 모든 데이터를 표하지 않고 12개의 데이터만 출력하였다.)

(5) 위치데이터 획득(Get Data) : 시뮬레이션상의 로봇의 위치와 방향값을 얻는다.

(6) 알고리즘실행 : 로봇의 제어를 위한 알고리즘 실행(여러가지 알고리즘 적용가능)

(7) GOAL : 로봇의 최종목적지를 절대 좌표값으로 기입한다.

나. *IPC(interprocess communication)*

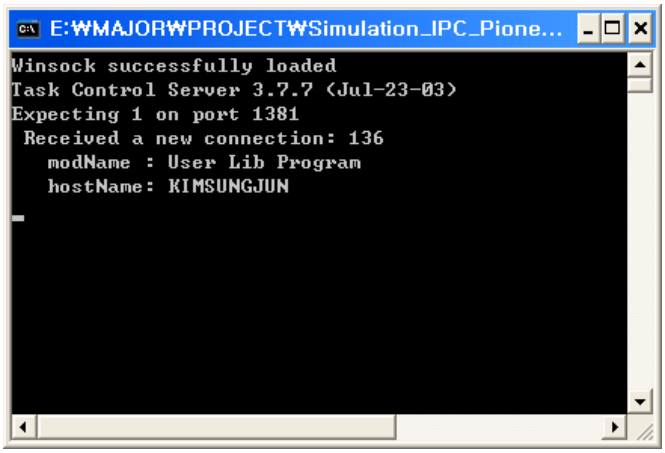

Fig. 3.11 IPC

프로그래머가 하나의 운영체계에서 동시에 수행될 개별 프로그램을 생성하고

다룰 수 있도록 해준다.

다. *Simulator*

시뮬레이터는 로봇의 센서와 제어기를 PC에서 시뮬레이션 하면서 로봇 하드웨어로는 불가능한 다양한 테스트를 할 수 있다.

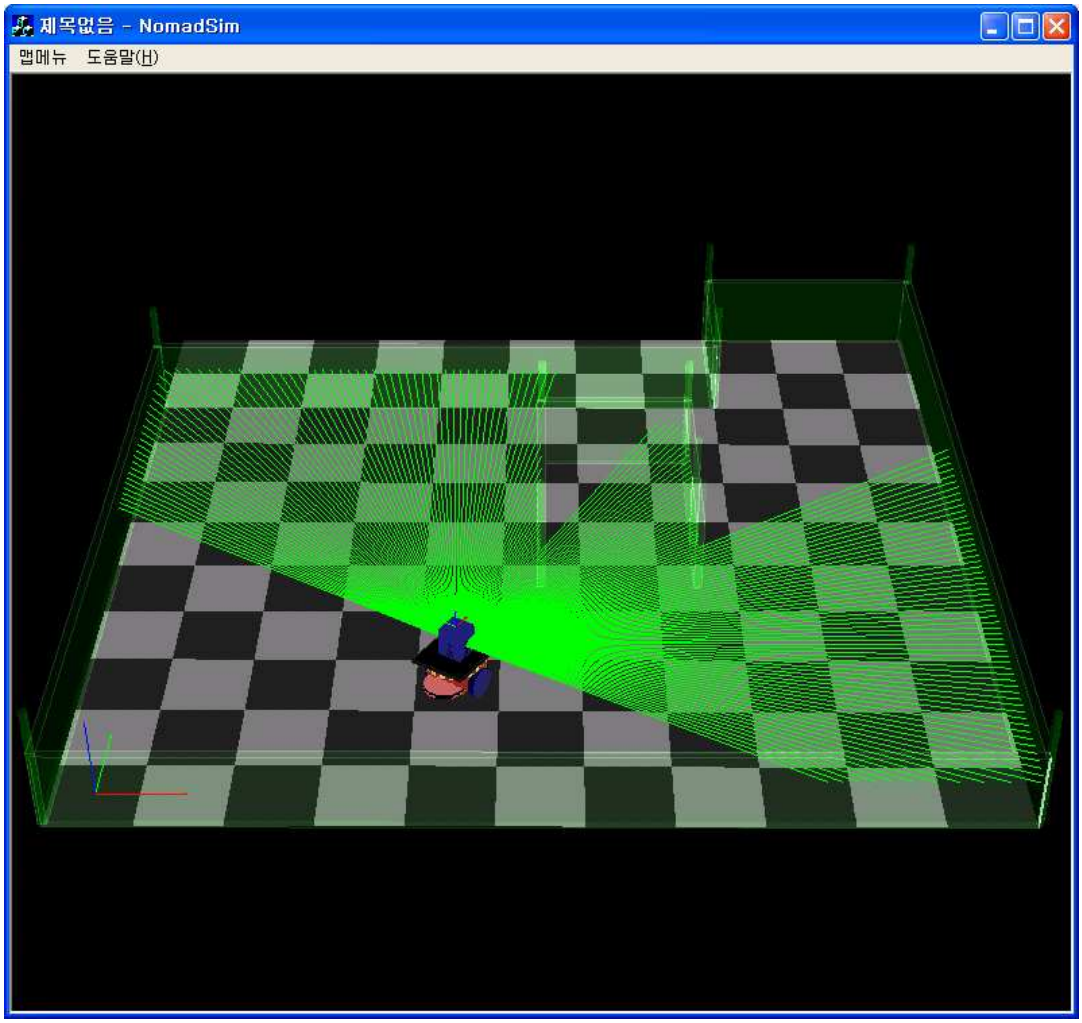

Fig. 3.12 Simulator

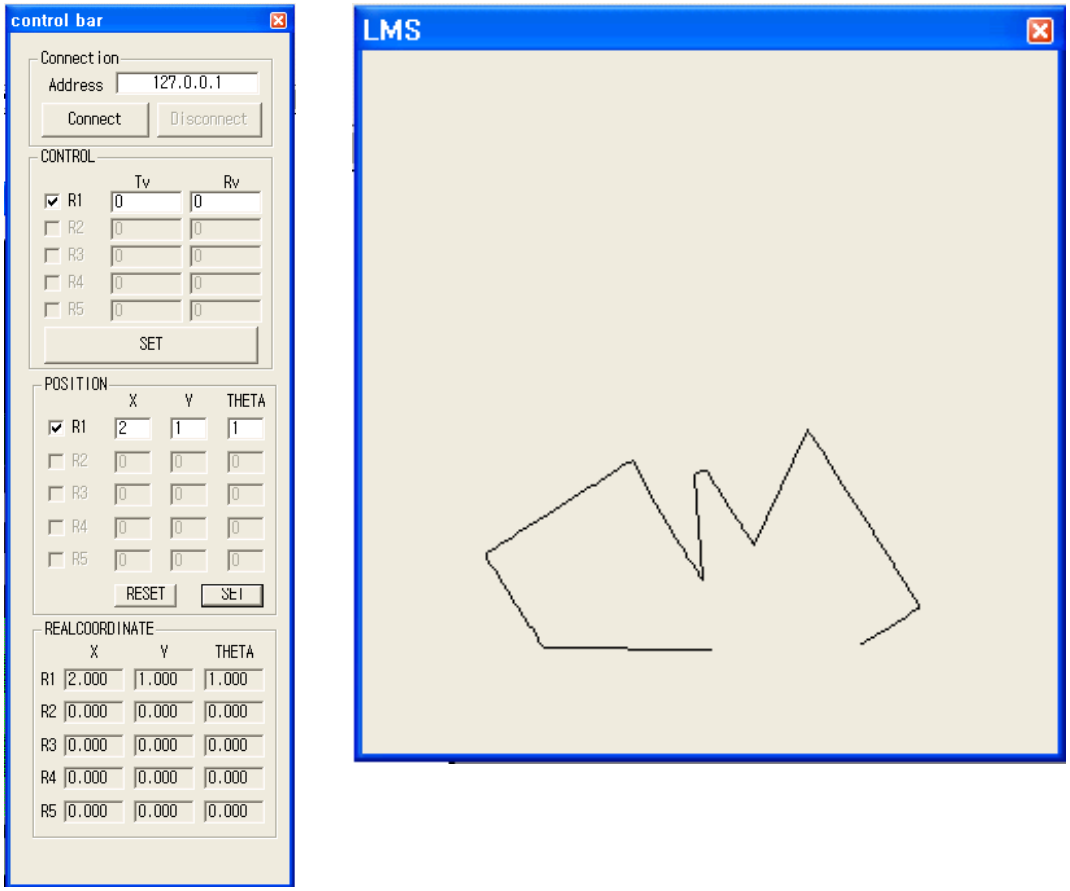

Fig. 3.13 Simulator controlbar and laser range finder

(1) 맵매뉴 : 프로그래머가 원하는 맵 설정

(2) Connect : IPC와 커뮤니케이션을 위한 접속

(3) CONTROL : Tv, Rv를 set하여 로봇제어 설정

(4) POSITION : 프로그래머가 로봇의 방향이나 위치설정

(5) REALCOORDINATE : 실시간으로 현재 로봇의 위치와 방향을 알려줌

(6) LMS dialog : laser range finder가 스캔한 값을 실시간 영상으로 나타냄

2. 사용법

Controller와 simulator는 독립적으로 사용하거나 IPC를 이용하여 통합적로 사용할 수 있다. 지금부터 전체적인은 사용방법에 대해 간략하게 설명한다.

가. *Controller*와 시뮬레이터를 상호 정보데이터를 주고받을 수 있도록 *IPC*에 접속

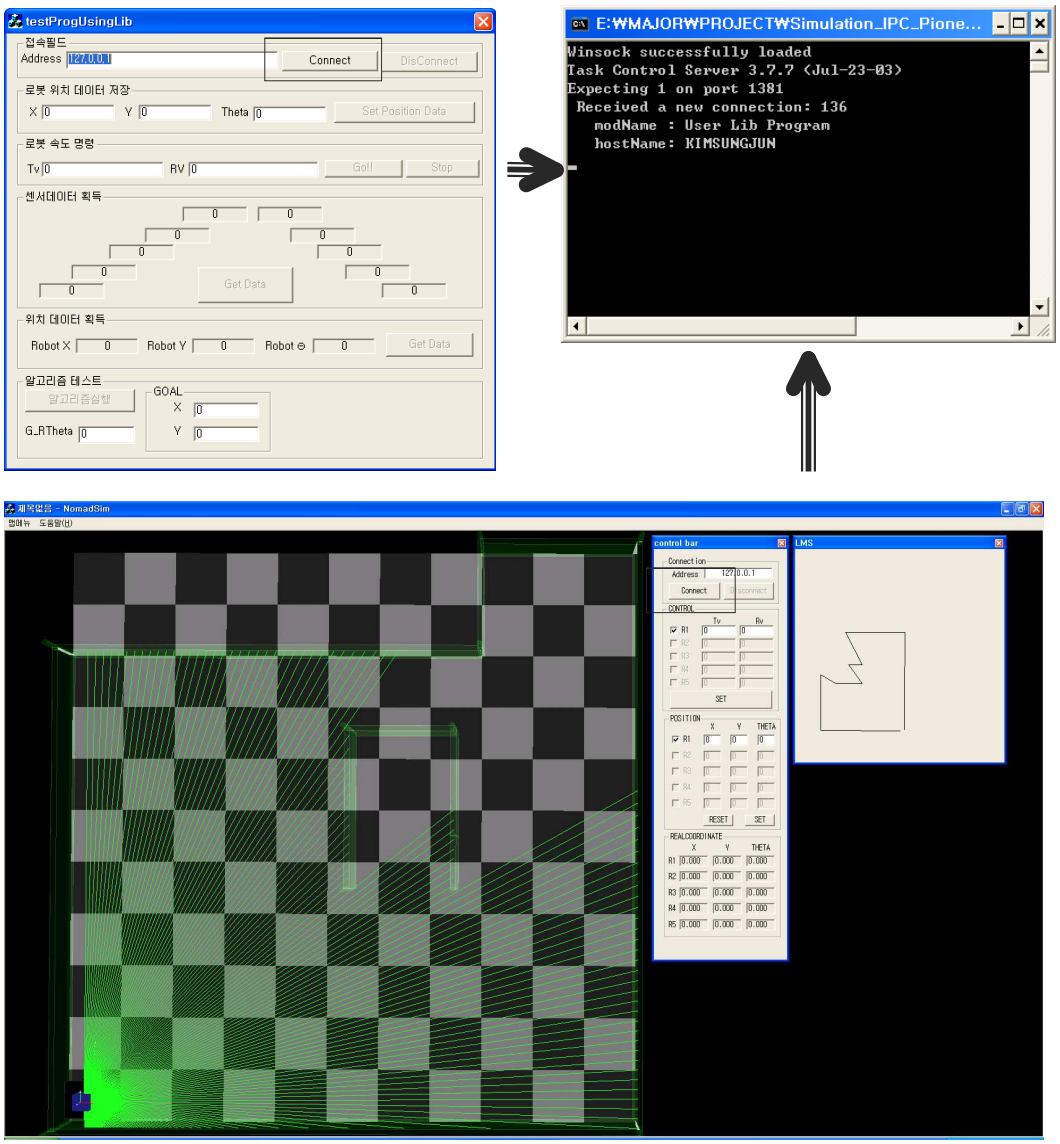

Fig. 3.14 Communication system between controller and simulator

나. 시뮬레이터의 *X*, *Y*, *Theta*에 프로그래머가 원하는 로봇의 위치와 방향 설정

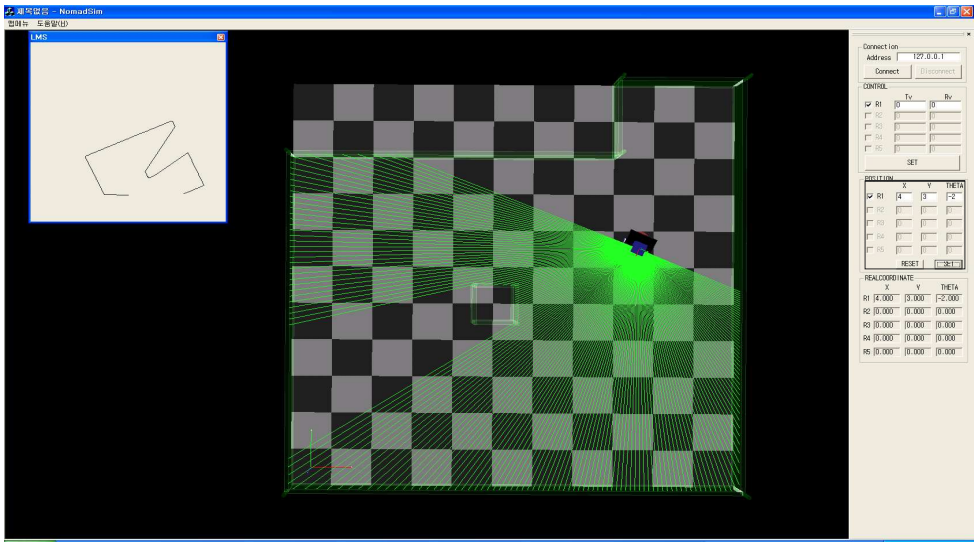

Fig. 3.15 Starting point coordination

다. 시뮬레이터에서 사용할 맵 설정

그림 3.16-1은 맵을 설정하지 않는 초기상태이며, 그림 3.16-2~4와 같이 제어조건에 맞게 맵 설정을 할 수 있다.

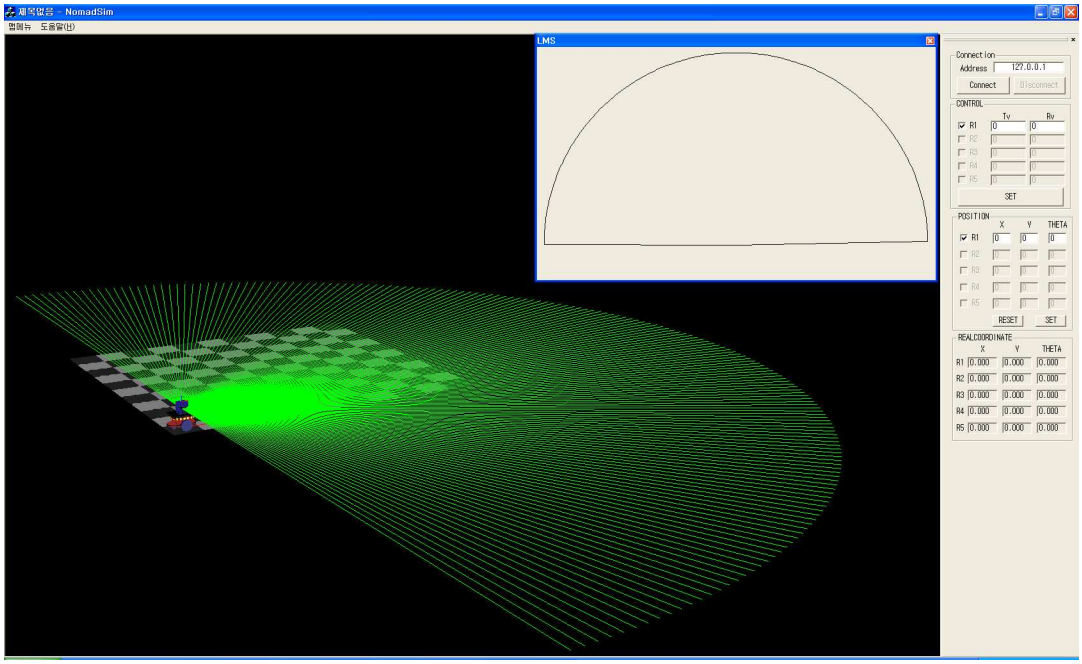

Fig. 3.16-1 Map coordination

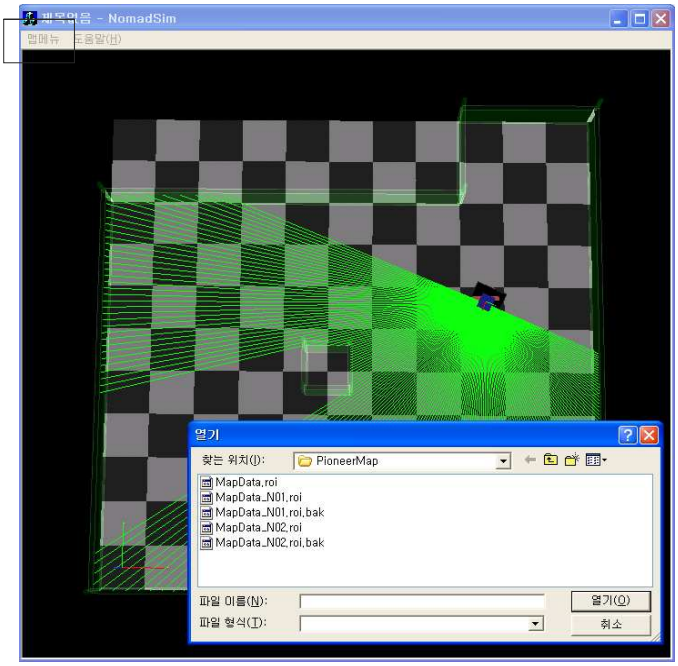

Fig. 3.16-2 Map coordination

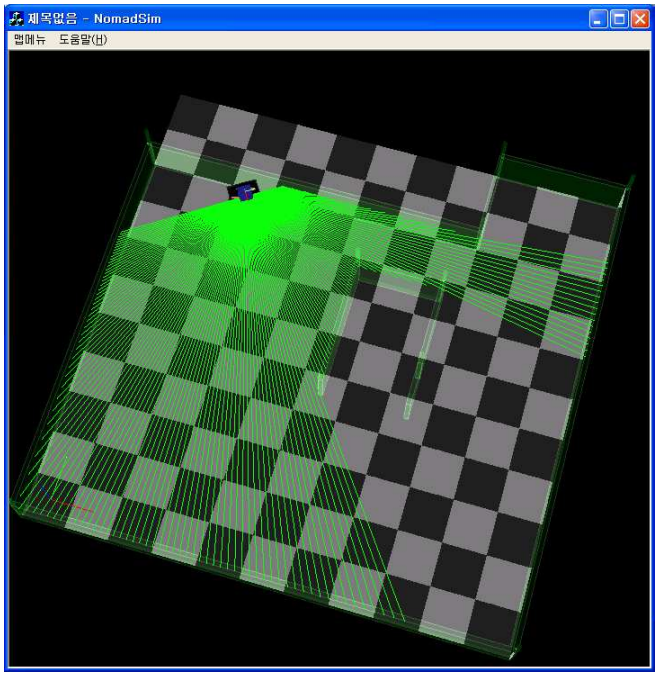

Fig. 3.16-3 Map coordination

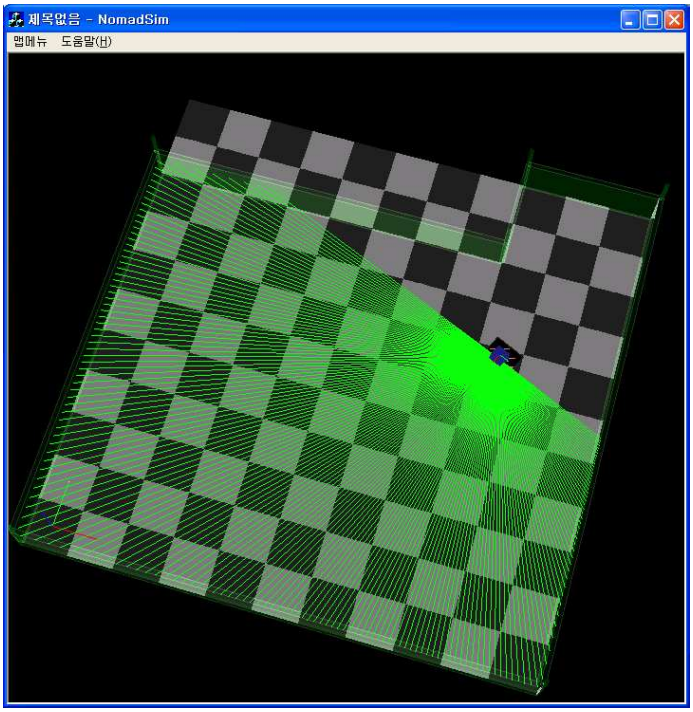

Fig. 3.16-4 Map coordination

라. Controller에 목표 위치 설정 및 알고리즘 실행

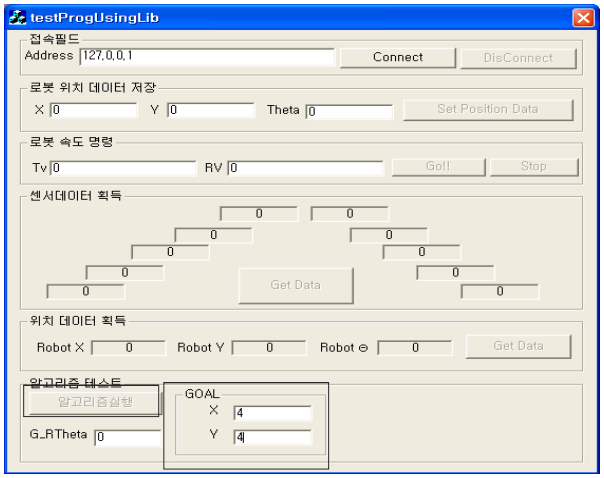

Fig. 3.17 Goal point coordination

제 4장 SIMULATION 실험

시뮬레이션상의 로봇은 PioneerⅡ를 모델로 하여 프로그램 되었으며 Virtual Force Field를 이용한 충돌 회피 시뮬레이션을 구현하였다.

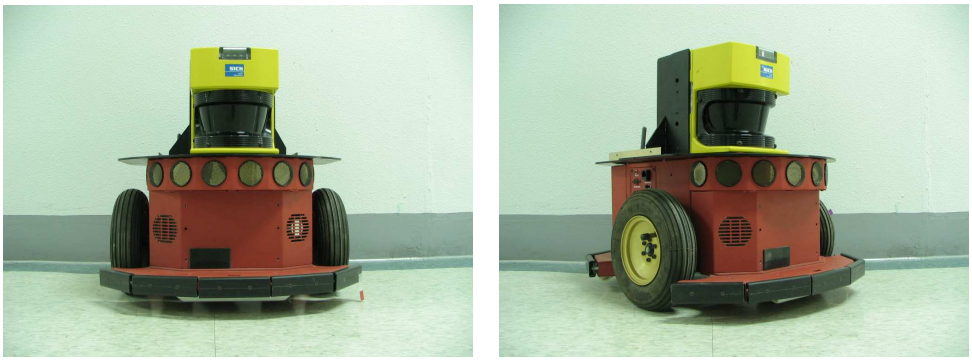

Fig. 3.18 PioneerⅡ

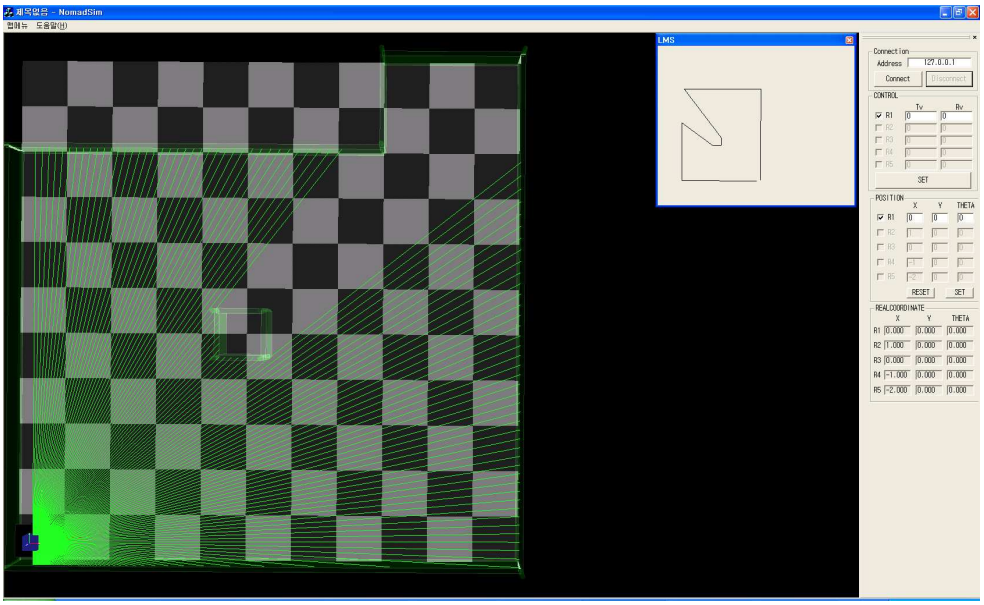

Fig. 3.19 Starting point

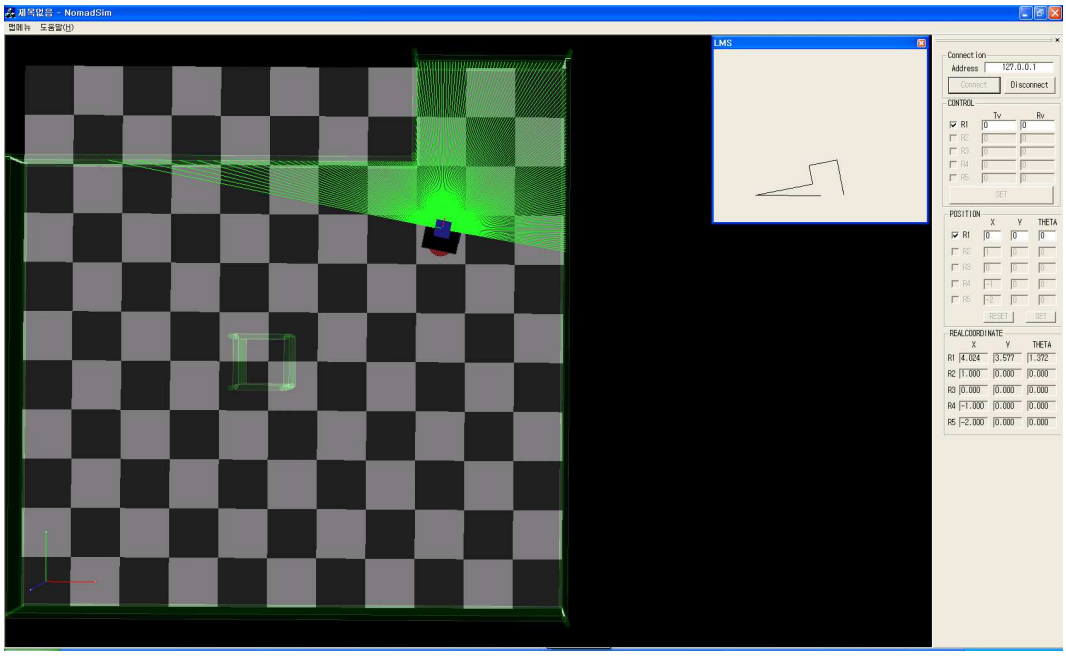

Fig. 3.20 Goal point

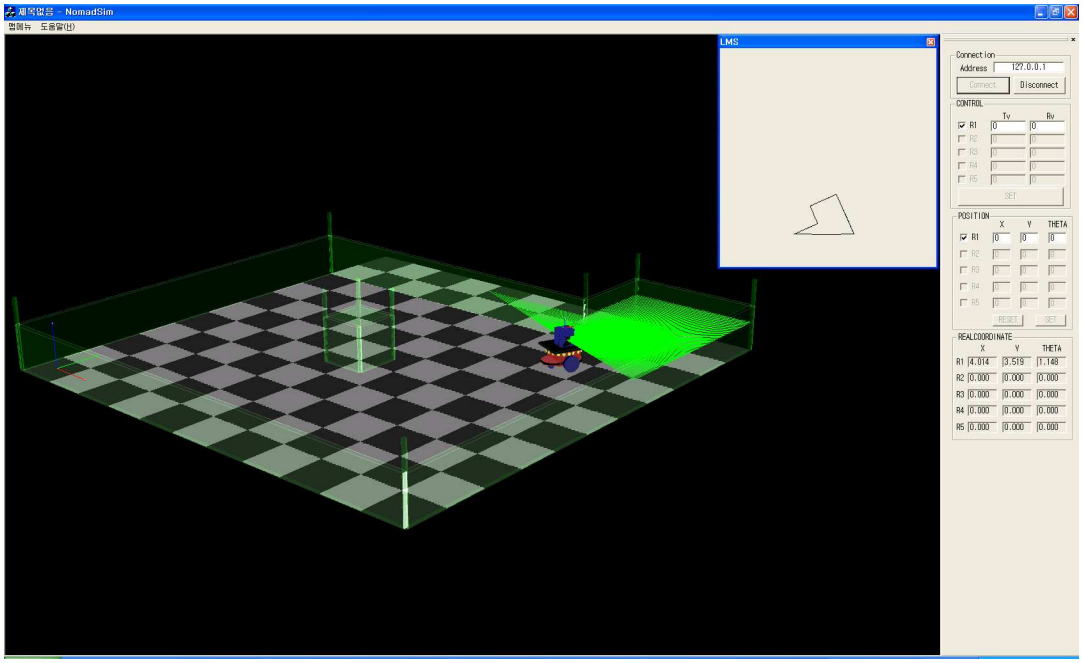

Fig. 3.21 Goal point

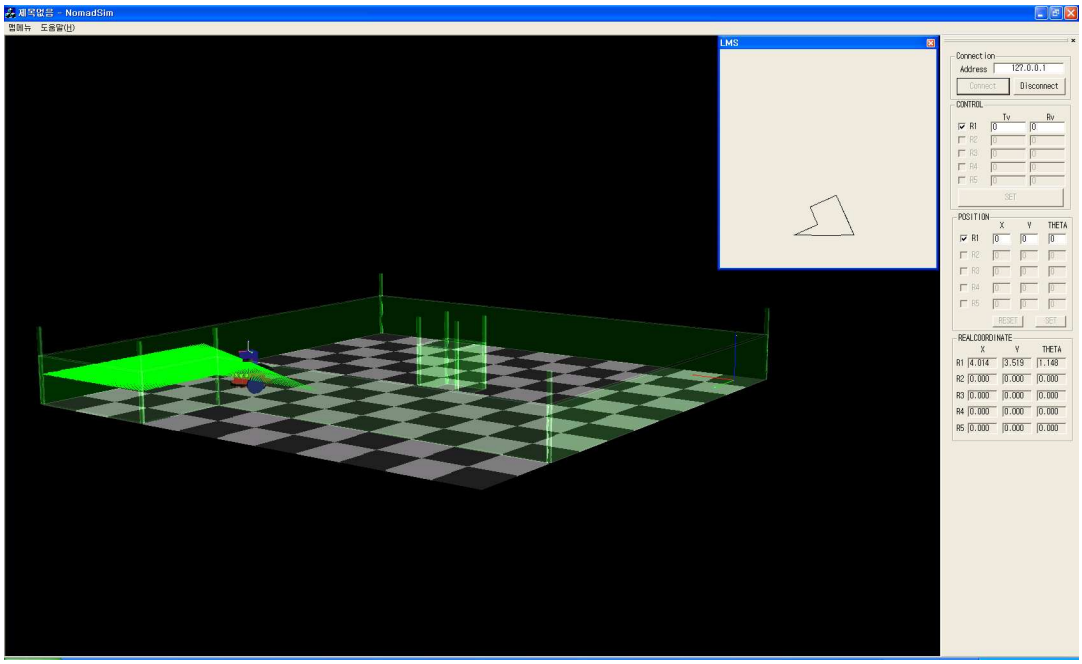

Fig. 3.22 Goal point

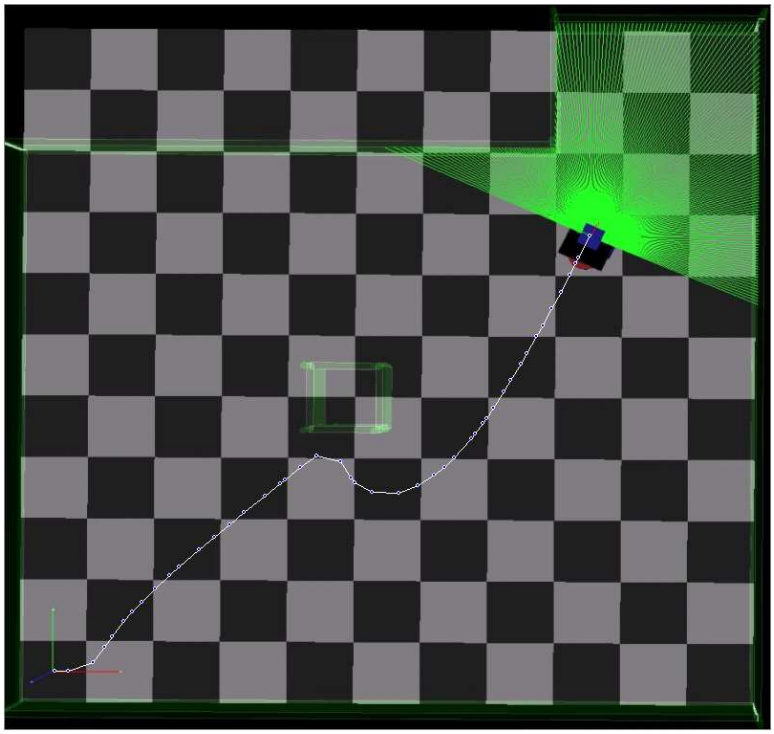

Fig. 3.23 Global path planning

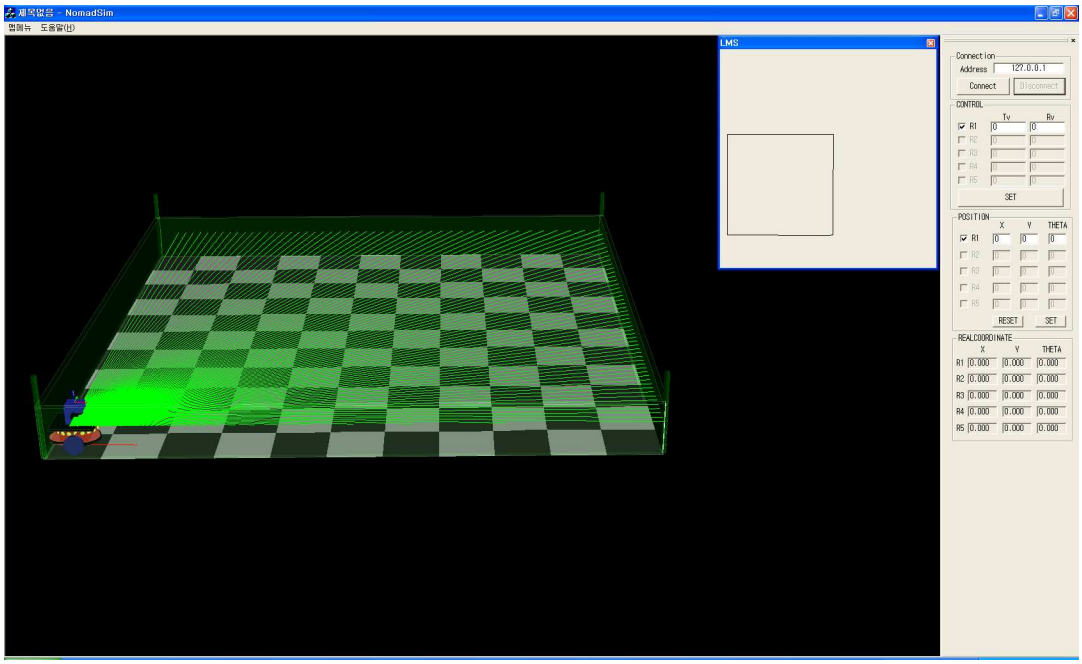

Fig. 3.24 Starting point

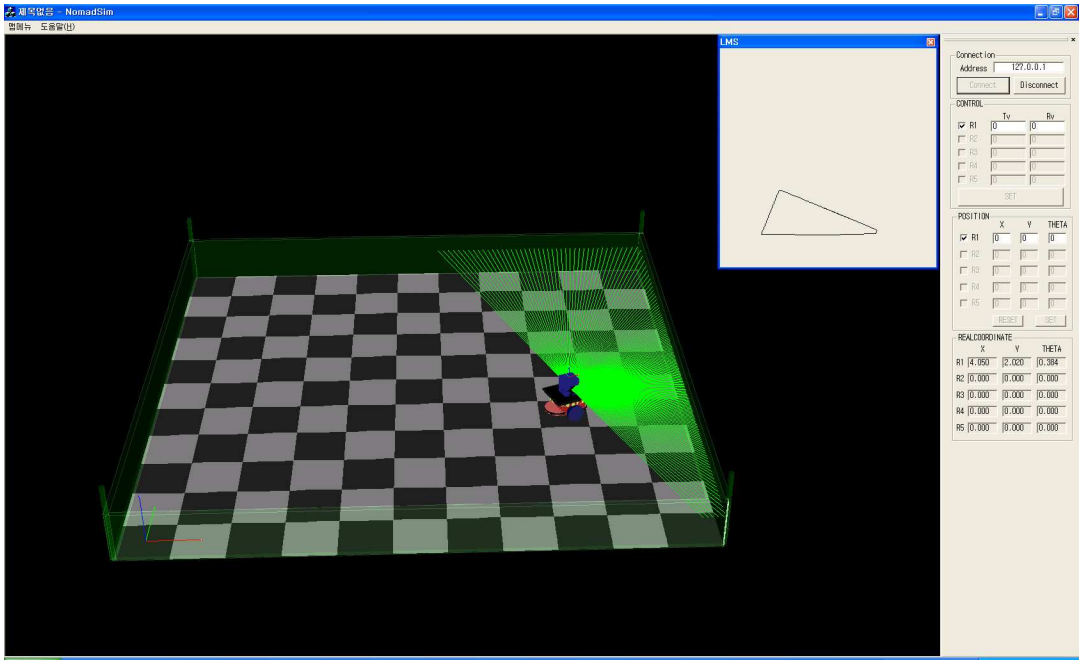

Fig. 3.25 Goal point

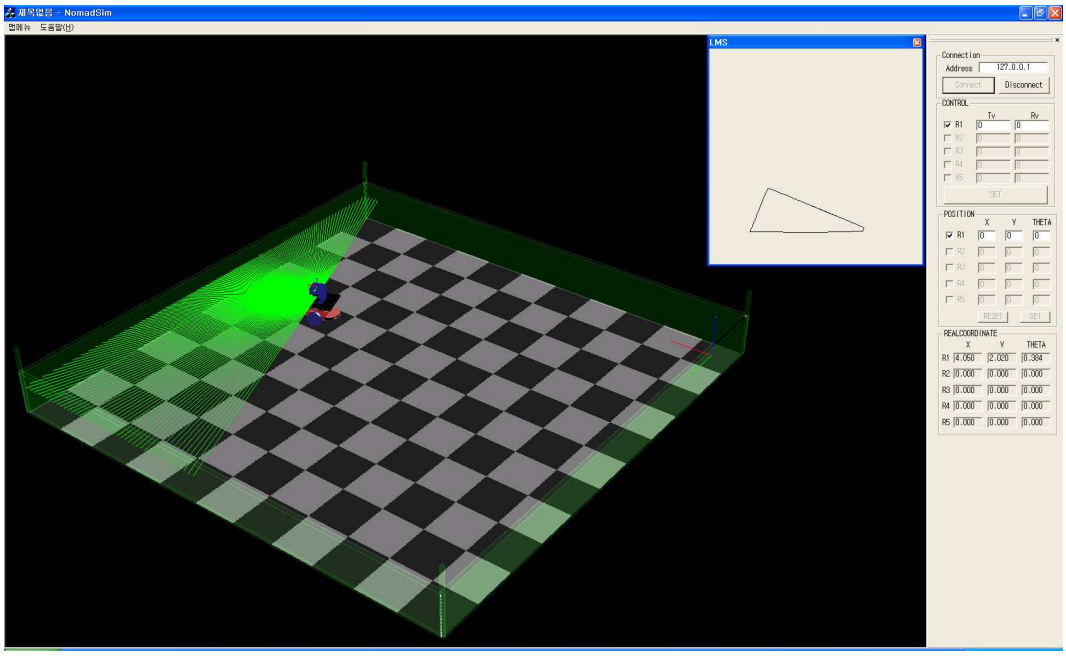

Fig. 3.26 Goal point

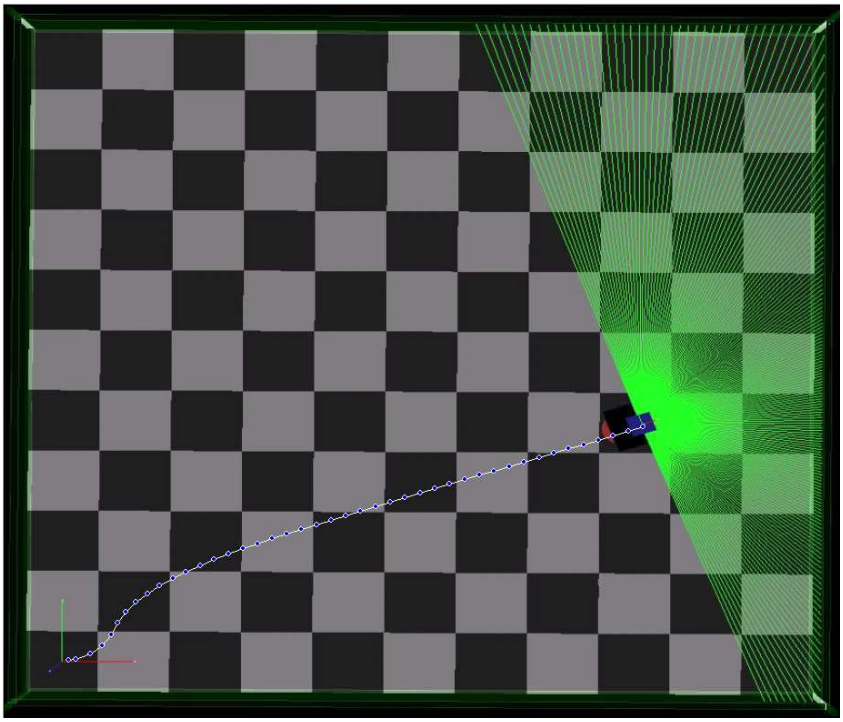

Fig. 3.27 Global path planning

그림 3.19~23은 로봇의 시작위치와 목표위치, 시뮬레이션 주행 실험 결과 목표 방향을 고려하여 장애물을 회피하고 목표위치에 도착하는 알고리즘에 대한 결과를 보여준다. 그리고 그림 3.24~27은 새로운 맵을 구성과 장애물이 없는 경우에서 가상거리를 이용한 인공전위계[12] 알고리즘을 적용하였다.

제 5 장 결론

본 논문에 자율 이동 로봇이 장애물 환경에서 주행하는 경우, 장애물에 대한 실시간 충돌 회피 동작의 문제에 대하여, 로봇이 장애물과 충돌하지 않는 안정적인 회피 동작과 신속한 주행, 유연한 궤적 생성, 그리고 효과적인 최적의 조향 동작을 계획할 수 있는 시뮬레이션을 개발하였다. 속도, 뱡향, 센싱, 주행경로 등을 비교하기 위해 실제 로봇을 모델로 하여 구현하였으며, 프로그래머가 하나의 운영체계에서 동시에 수행될 개별 프로그램을 생성하고 다룰 수 있도록 IPC를 사용하였다.

IPC를 사용하면 각각의 프로그램의 독립성이 보장되므로 개별제어가 가능하고 다른 프로그램과의 호환이 유리한 장점이 있다. 또한 독립적으로 구현된 프로그램들을 하나의 통합시스템으로 구현하기 쉽다.

본 논문에서 개발된 시뮬레이터는 제어부와 시뮬레이션부로 나누워 독립적으로 프로그래밍하고 IPC를 이용하여 통합하는 시스템으로 구현하였으며, 제어부는 로봇의 절대위치, 방향, 속도, 목표지점을 프로그래머가 할당할 수 있는 독립성을 보장하면서 IPC를 이용한 통합 시스템에서는 시뮬레이터에서 IPC를 통해 수신되는 센서 데이터, 위치데이터, Tv, Rv를 실시간으로 알 수 있다.

시뮬레이션 프로그램(Visual C++의 MFC)에서 사용된 OpenGL은 Mac OS, OS/2, Windows 95/NT, UNIX, Linux, BeOS와 같은 OS를 모두 지원하며, C, C++, Visual Basic, Java, Python과 같은 대부분의 언어에서 사용이 가능하고 3D 그래픽스용으로 설계되어 좀 더 비주얼하게 로봇의 움직임을 표현 할 수 있다.

시뮬레이션 통한 검증은 실제 시스템을 구축하지 않고 제안된 알고리즘을 적용함으로서 이동로봇의 속도, 방향, 경로, laser range finder 데이터 등에 관한 정보를 쉽게 얻을 수 있으므로 실제 시스템에서 발생할 수 있는 에러를 사전에 보정할 수 있어 로봇의 개발에 있어서 많은 시간과 비용을 단축 할 수 있으며,

OpenGL 라이브러리를 사용하여 프로그래밍 되었으므로 2차원 시뮬레이터보다 더 현실감 있는 시뮬레이션이 가능하다.

추후 과제로는 다양한 제어알고리즘에 더욱 적절하게 대응 할 수 있는 여러 가지 맵 구성물의 제작이 요구되며, 다양한 로봇형상, 센서, 매뉴퓰레이터의 추가가 필요하다. 또한 환경에 대해 더욱 적절하게 대응할 수 있는 제어 알고리즘의 개발과 실제 로봇에서의 환경인식이나 경로에서 생길 수 있는 오차나 에러를 발생 시킬 수 있는 모듈과, 로봇과 물체와의 충돌이나 관성력과 같은 물리현상을 시뮬레이션 할 수 있는 물리엔진모듈의 적용이 필요하다.

참고문헌

[1] T. Balch and R. Arkin., “Behavior-based formation control for multi-robot teams,” *IEEE Transactions on Robotics and Automation*, 14(6):1 - 15, 1998.

[2] R. Alur, A. Das, J. Esposito, R. Fierro, Y. Hur, G. Grudic, V. Kumar, I. Lee, J. P. Ostrowski, G. Pappas, J. Southall, J. Spletzer, and C. Taylor, "A framework and architecture for multirobot coordination," *Experimental Robotics: LNCS Series, Springer-Verlag*, 2001.

[3] Rachid Alami, Frederic Robert, Felix Ingrand, and Sho'ji Suzuki. "Multi-robot Cooperation through Incremental Plan-Merging," *IEEE R&A*, 1995.

[4] Larcommbe, M. H. E., "Tracking Stability of Wire Guided Vehicles," Proceeding of International Conference on Automatic Guided Vehicle System: 137-144, 1981.

[5] Mckey, E. S., Drake, K. C., and Inigo, R. M. "Range Measurements by a Mobile Robot Using a navigation Line," IEEE Transaction on PAMI 8: 105-109, 1986.

[6] Nilson, N. J., "A Mobile Automation: An Application of Artificial Intelligence Technique," Proceeding of 1st IJCAI, 1969.

[7] R. Fierro, A. Das, V. Kumar, and J. P. Ostrowski, "Hybrid control of formations of robots," *IEEE Int. Conf. on Robotics and Automation, Seoul, Korea*, May 2001, pp. 157-162.

[8] Tews A.and Wyeth G.F., “Multi-Robot Coordination in the Robot Soccer Environment,” *Proceedings of the Australian Conference on Robotics and Automation* March 30-April 1, 1999. Brisbane, pp. 90-95.

[9] A. Saffiotti, N.B. Zumel, and E.H. Ruspini., "Multi-Robot Team Coordination using Desirabilities," *Proc. of the 6th Intl. Conf. on Intelligent Autonomous Systems (IAS)*, pp. 107-114. Venice, Italy, 2000.

[10] R. Simmons, T. Smith, M. B. Dias, D. Goldberg, D. Hershberger, A. Stentz, and

R. Zlot, “A Layered Architecture for Coordination of Mobile Robots in Multi-Robot Systems From Swarms to Intelligent Automata,” A. Schultz and L. Parker (eds.), Kluwer, 2002.

[11] 고낙용, 이범희, “충돌회피 가능도를 이용한 로봇의 이동 장애물 회피”, *Journal of Control, Automation and System Engeering,* vol. 3. No. 2., pp. 169-178, April. 1997.

[12] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The Int. J. Robotics Research,* vol. 5. no. 1. Spring, 1986.

[13] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes., “Coordination for Multi-Robot Exploration and Mapping,” *Proceedings National Conference on Artificial Intelligence*, Austin TX, August 2000.

[14] Hancock, J. A., *Laser Intensity-Based Obstacle Detection and Tracking*, HTTP:// www.ri.cmu.edu/pub_files/pub1/hancock_jhon_1999_1/hancock_jhon_1999_1.pdf

[15] R.M. Zlot, A. Stentz, M.B. Dias, and S. Thayer, “Multi-Robot Exploration Controlled By A Market Economy,” *IEEE International Conference on Robotics and Automation*, May, 2002.

[16] F.C.A. Groen, J. Roodhart, M. Spaan, R. Donkervoort, and N. Vlassis. "A distributed world model for robot soccer that supports the development of team skills", *In Proceedings of the 13th Belgian-Dutch Conference on Articial Intelligence (BNAIC'01)*, 2001.

[17] Stentz, A., and Dias M. B., "A Free Market Architecture for Coordinating Multiple Robots", *Technical report, CMU-RI-TR-99-42, Robotics Institute, Carnegie Mellon University*, 1999.

[18] M.B. Dias, and A. Stentz, “A Market Approach to Multirobot Coordination” *Technical report CMU-RI-TR-01-26, Robotics Institute, Carnegie Mellon University*, August, 2001.

[19] Ko, N. Y. and Simmons, R. G., "The Lane-curvature Method for Local Obstacle Avoidance," *International Conference in Intelligent Robots and Systems(IROS 1998)*, Victoria, B.C, Canada, Oct. 13-17, 1998.

[20] Boernstein, J. and Koren, Y. "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transaction on Robotics Automation* 7(3) : 278-288, June, 1991.

[21] Boernstein, J. and Koren, Y. "A Mobile Platform for Nursing Robots," *IEEE Transactions on Industral Electronice* 32(2): 158-165, 1985

[22] Arkin, R. C., "Motor Schema-Based Mobile robot Navigation," *The International Journal of Robotics Research* : 92-112, August 1989.

[23] 고낙용,김재열, "이동 로봇의 실시간 장애물 회피를 위한 새로운 방법," [한국공작기계학회지] 7(4) : 1998.