

2008년 항우기 종합설계

밸런싱 모바일 로봇 설계/제작



항공우주기계공학부 로봇시스템제어연구실

(빈 페이지)

1. 모바일 로봇 제작의 목적

무인항공기를 통해 원거리 영상과 정보를 획득하고 무인항공기에서 얻지 못하는 근거리, 실내영상과 정보는 자율주행 로봇을 통해 얻는다.

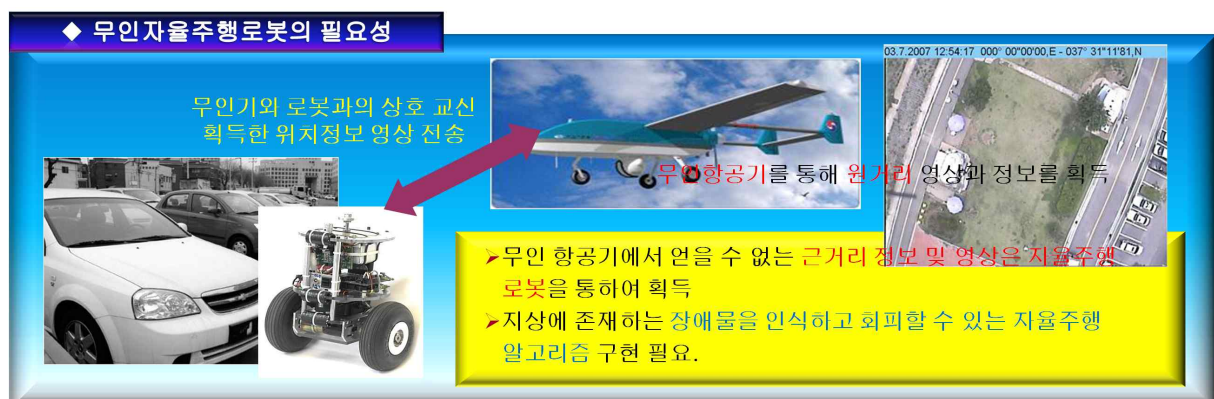
이러한 근거리, 실내영상에서는 무인항공기와 달리 지상에서의 많은 구조물로 인해 장애물 인식 및 자율주행 알고리즘 등이 필요하다. 또한 장시간 정보획득과 무인기 연동을 위한 빠른 움직임을 가져야한다.

위의 2조건을 만족하기 위한 구성으로 최근 들어 두 바퀴를 가지는 밸런싱 로봇에 대한 연구가 활발히 이루어지고 있다. 이는 서비스 및 주행 로봇의 알고리즘이 휴머노이드에서 밸런싱 로봇으로 변화되었기 때문이다. 밸런싱 로봇은 휴머노이드에 비하여 사용되는 모터의 수가 적고 균형을 잡으려면 관절마다 값비싼 고성능 모터가 필요하며 이를 가동하려면 전력도 많이 소모되며 대용량 배터리를 장착할 수밖에 없게 된다. 반면 바퀴로 움직이는 로봇은 전력이 적게 들고 이동도 쉽다. 또한 이동의 속도가 빠르며, 두 바퀴로 움직이기 때문에 험로 주행 및 계단에서의 주행이 가능하다.

개발 동기에 따른 로봇을 제작하기 위해 크게 3가지로 개발 단계를 구성하였으며 첫 번째 단계로는 실내에서 Host PC와 연동할 수 있는 무인정찰로봇 제작, 두 번째는 실외에서 험로 주행 및 원거리 통신 구현, 마지막 세 번째는 무인기와와의 다양한 mission 수행 로봇을 제작이다.

현 단계는 1단계인 실내에서 Host PC와 연동할 수 있는 무인정찰로봇이다. 1단계에서의 목적으로는 Navigation 및 Localization 알고리즘을 구현, 두바퀴 로봇의 밸런싱, 영상처리를 통한 각각의 담당부분의 보완, 마지막으로 영상전송이 있다.

◆ 무인자율주행로봇의 필요성



The diagram illustrates the necessity of an unmanned autonomous driving robot. It shows a ground robot (a two-wheeled balancing robot) and an unmanned aircraft (a small plane) working together. The ground robot is responsible for collecting data in areas where the aircraft cannot reach, such as close-range information and indoor environments. The aircraft is used for long-range video and information collection. The ground robot also handles obstacle recognition and avoidance on the ground. The diagram includes a yellow box with text explaining the roles of the aircraft and the ground robot, and a red arrow pointing from the ground robot to the aircraft.

무인기와 로봇과의 상호 교신
획득한 위치정보 영상 전송

무인항공기를 통해 원거리 영상과 정보를 획득

무인 항공기에서 얻을 수 없는 근거리 정보 및 영상은 자율주행 로봇을 통하여 획득

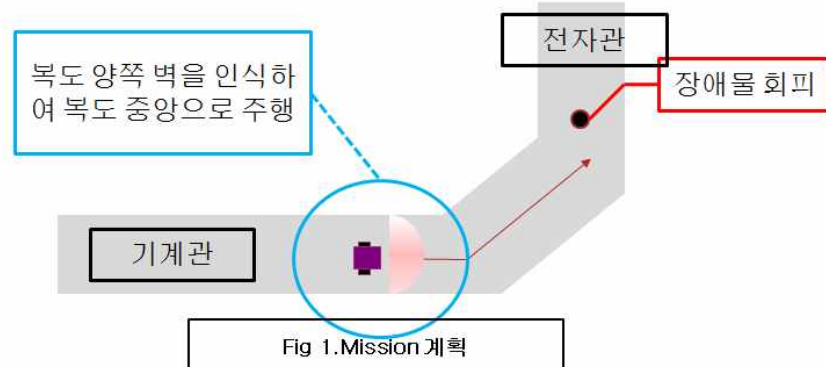
지상에 존재하는 장애물을 인식하고 회피할 수 있는 자율주행 알고리즘 구현 필요.

2. 모바일 로봇의 임무 및 기능

본 과제의 목적은 실내에서 Host PC와 연동할 수 있는 무인정찰 로봇 제작이다. 수행할 mission은 아래 그림과 같이 공학관 복도에서 장애물을 회피하는 왕복 주행이다.

▶ 무인기 연동운용을 위한 자율주행 로봇설계제작

▶ Step1. 실내에서 Host PC와 연동할 수 있는 무인정찰로봇 제작





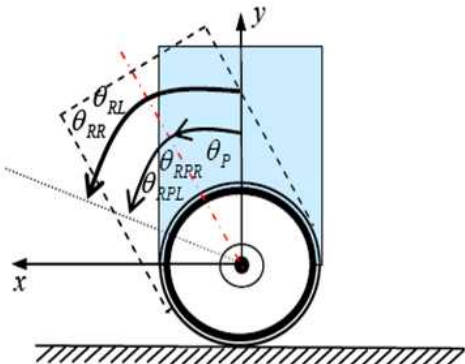

▶ Step2. 실외에서 험로 주행 및 원거리 통신 구현

▶ Step3. 무인기와의 연동을 통한 다양한 Mission 수행 로봇

다음 그림은 본 과제를 통해 제작된 로봇의 모듈별 구성을 나타낸다.



본 과제에서는 다음과 같이 크게 4가지 수행능력을 갖는 로봇을 제작하였다.

수행능력	내용
Navigation	
Localization	
Balancing	
Communication	

3. 로봇 설계 및 제작 과정

3.1 밸런싱 로봇 메커니즘 설계·제작

3.1.1 서론

밸런싱 로봇은 두 바퀴로 구동하는 로봇이기에 자세 제어가 필수적이다. 밸런싱 로봇의 모터 제어 및 정확한 모델링, 그리고 적절한 설계는 자세 제어를 용이하기 위하여 거쳐야 할 과정들이다.

3.1.2 밸런싱 로봇의 설계

밸런싱 로봇의 설계는 밸런싱을 위하여 가장 고려되어야 할 완전적인 과제이다. 기본적으로 고려해야 할 사항은 밸런싱 로봇의 프레임(Frame)과 바퀴, 그리고 밸런싱 로봇을 구동하게 될 서보모터와 모터를 구동하기 위한 배터리가 필요하다. 또한 밸런싱을 위한 자세 검출용 센서와 이를 제어해야 할 제어회로가 필요하다.

(1) 밸런싱 로봇 설계의 착안점

밸런싱 로봇의 운동은 직선 주행과 곡선 주행으로 나눌 수 있다. 첫 번째로 생각해야 할 요소인 프레임은 바퀴를 직선상 횡으로 부찰 할 수 있는 밑판을 구성해야 하며, 모터의 출력과 배터리의 용량을 고려한 무게를 적용해야 한다. 또한 회전 운동 시 원심력에 의하여 넘어지지 않을 정도의 충분한 넓이의 프레임이 필요하다.

두 번째로 액추에이터 부분으로 바퀴, 모터 및 배터리를 선정하여야 한다. 바퀴는 타이어의 광폭이 커질수록 동력 전달의 효율성과 코너링이 올라가지만 이에 따른 무게의 증가를 충분히 고려하여 알맞은 크기의 바퀴를 선정해야 한다. 또한 밸런싱 로봇을 구동시킬 서보모터는 자세제어에 필요한 최소한의 모터 출력 이상을 가져야 하며, 배터리는 선택한 모터의 전압과 고려해야 하며, 너무 무거운 배터리를 선택할 시 모터의 출력이 높아져야 한다. 또한 주행거리와 자세제어에 필요한 전압 이상의 용량이 필요하다.

세 번째로 센서 모듈로 자세 검출용 센서와 제어회로를 고려해야 한다. 밸런싱 로봇은 두 바퀴로서 있기 때문에 기울기 센서 및 자이로 센서의 샘플링 타임을 충분히 고려해야 한다. 또한 제어 회로로서 주행 시 차체가 쓰러지지 않게 하기 위하여 자세 제어가 필요하며, 회전 시 한쪽 바퀴는 그대로

주행하며 회전하고자 하는 방향의 바퀴는 더 빠르게 움직이므로 세 회전이 가능한 회로를 구성해야 한다.

(2) 밸런싱 로봇의 설계

위의 착안점을 기준으로 CATIA를 이용하여 밸런싱 로봇을 제작하였다. 로봇의 밸런싱을 위하여 무게가 무거운 파트들은 밑으로 배치하였으며, 좌우의 무게 중심을 최대한 맞추기 위하여 대칭적인 배치를 하였다. Fig 1은 CATIA를 이용하여 밸런싱 로봇을 설계한 것이다.

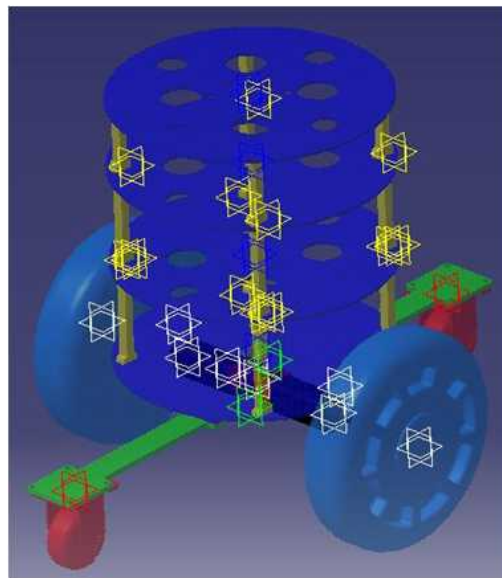


Fig. 1 Balancing Robot Design

3.1.3 밸런싱 로봇(Balancing)의 모델링

밸런싱 로봇은 두 개의 바퀴로만 구동되기에 자세제어를 위한 정확한 모델링이 필수적이다. 전체적인 밸런싱 로봇의 모델링은 바퀴부분(Wheel Part)과 몸체부분(Chasis Part)로 구분하여 모델링을 하였다. 이 모델링은 기본적으로 역진자(Inverted Pendulum)의 자세제어 알고리즘을 이용하여 접근하였다. 또한 위의 모델링 값을 이용하여 RK4를 통한 수치 해석적 접근을 통하여 밸런싱 로봇의 동역학적 움직임을 유추할 수 있다.

(1) 바퀴부분 모델링

첫 번째로 바퀴부분에 대한 모델링을 고려하였다. 바퀴는 오른쪽과 왼쪽을 동일 시 하여 모델링 하였으며 아래와 그림과 같이 직선운동(Liner Motion)과 회전운동(Rotational Motion)으로 나누어 생각하였다. 또한 수직

(Vertical) 방향의 힘은 생각하지 않았다. 이는 바퀴에서의 운동이 위의 두 개의 운동으로만 구성되기 때문이다. 첫 번째로 직선운동은 뉴턴의 운동방정식(Newton's Law of Motion)을 이용하였다.

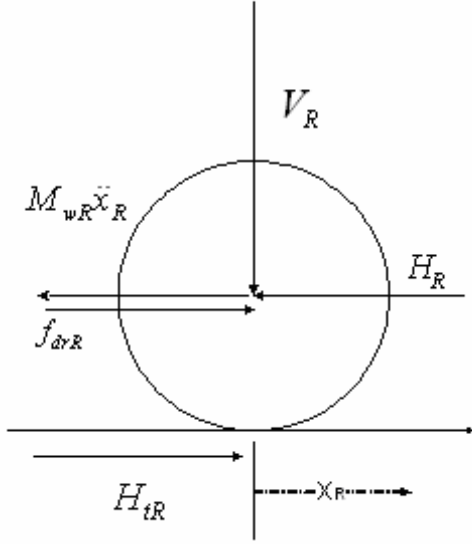


Fig. 2 Translation force-free body diagram of right wheel

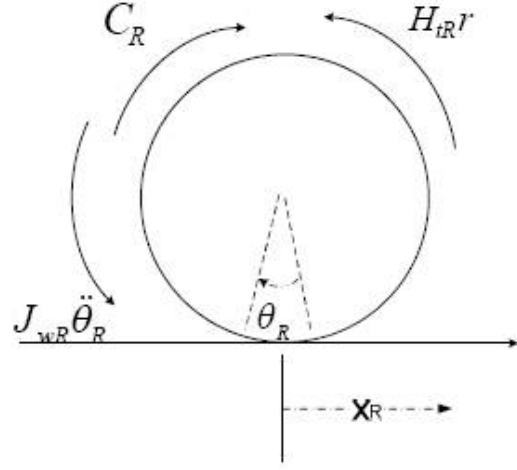


Fig. 3 Rotational force-free body diagram of right wheel

$$\sum F_x = M\ddot{x}$$

$$H_{tR} - H_R + f_{drR} = M_{wR}\ddot{x}_R \quad (1)$$

두 번째로 회전운동은 위와 같이 뉴턴의 운동방정식을 이용하여 바퀴 중심에서의 모멘트의 합으로 나타내었다.

$$\sum M = J\ddot{\theta}$$

$$C_R - H_{tR}r = J_{wR}\ddot{\theta}_R \quad (2)$$

위의 두 식(1)과 (2)를 H_{tR} 에 대하여 연립하여 다음과 같은 식을 얻었다.

$$M_{wR}\ddot{x}_R = \frac{C_R}{r} - \frac{J_{wR}}{r}\ddot{\theta}_R - H_R + f_{drR} \quad (3)$$

또한 직선운동을 회전운동으로 표현하기 위하여 다음과 같은 관계식을 이용하였다.

$$r\theta = x$$

$$\therefore \theta = \frac{x}{r} \Rightarrow \dot{\theta} = \frac{\dot{x}}{r} \Rightarrow \ddot{\theta} = \frac{\ddot{x}}{r} \quad (4)$$

(4)식을 이용하여 (3)식을 다음과 같이 변형하였다.

$$\left[M_{wR} + \frac{J_{wR}}{r} \right] \ddot{x}_R = \frac{C_R}{r} - H_R + f_{drR} \quad (5)$$

왼쪽 바퀴의 움직임 또한 오른쪽 바퀴와 동일하기에 식 (5)을 왼쪽 바퀴에 대하여 아래와 같이 나타낼 수 있다.

$$\left[M_{wL} + \frac{J_{wL}}{r} \right] \ddot{x}_L = \frac{C_L}{r} - H_L + f_{drL} \quad (6)$$

또한 $M_w = M_{wL} = M_{wR}$, $J_w = J_{wR} = J_{wL}$ 이기에 위의 식 (5)과 (6)는 다음과 같이 정의 된다.

$$H_R = \frac{C_R}{r} - \left[M_w + \frac{J_w}{r} \right] \ddot{x} + f_{drR} \quad (7)$$

$$H_L = \frac{C_L}{r} - \left[M_w + \frac{J_w}{r} \right] \ddot{x} + f_{drL} \quad (8)$$

하지만 밸런싱 로봇은 C_L 과 C_R 의 값에 따라 Yaw방향의 운동을 하고, 또한 회전방향을 정확히 제어하기 위하여 Ω 이라는 값을 정의하였다. Ω 값에 따른 운동은 아래 표와 같다.

Table 1 Physical Meaning when Ω Change

Value of Ω	Physical Meaning
0	Stop
1	Forward/Backward
$0 < \Omega < 1$	Turn Left
$1 < \Omega < 2$	Turn Right

따라서 식(8)은 아래와 같이 변하게 된다.

$$H_L = \frac{C_L}{r} - \left[M_w + \frac{J_w}{r} \right] \Omega \ddot{x} + f_{drL} \quad (9)$$

(2) 몸체부분 모델링

두 번째로 고려해야 할 사항은 몸체부분(Chasis)이다. 몸체부분의 모델링은 각각 수평방향힘(Horizontal force)과 수직방향힘(Vertical force) 그리고 모멘텀(Momentum)으로 나누어 생각하였다.

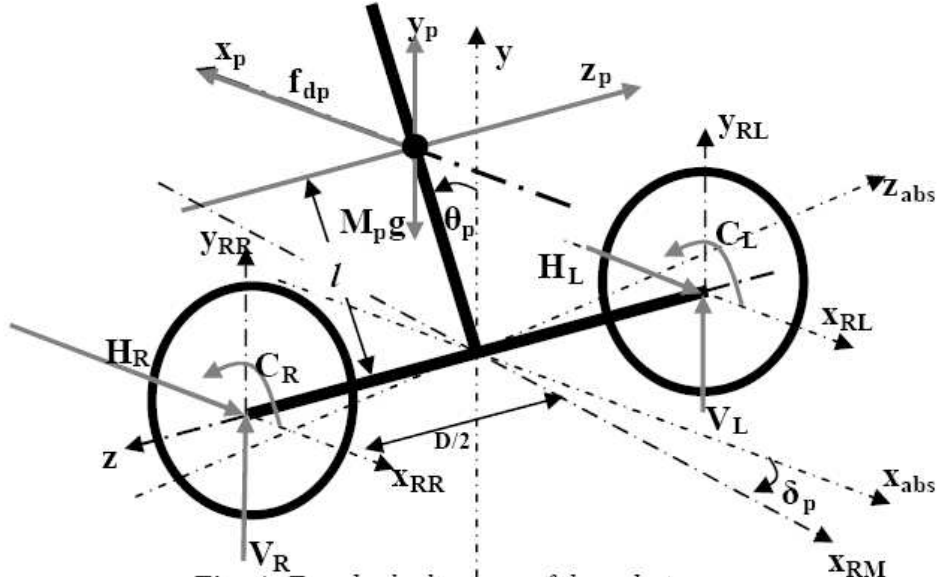


Fig. 4 Free body diagram of the robot

일단 수평방향의 힘은 아래와 같이 나타내어진다.

$$\sum F_x = M\ddot{x}$$

$$(H_L + H_R) = M_P\ddot{x} + M_P l\ddot{\theta}\cos\theta - M_P l\dot{\theta}^2\sin\theta - f_{dp} \quad (10)$$

또한 수직방향의 힘 아래와 같이 나타낼 수 있다.

$$\sum F_P = M\ddot{x}\cos\theta$$

$$(H_L + H_R)\cos\theta + (V_L + V_R)\sin\theta - M_P g\sin\theta - M_P l\ddot{\theta} + f_{dp}\cos\theta = M_P\ddot{x}\cos\theta \quad (11)$$

마지막으로 몸체의 무게중심에서의 모멘텀의 합은 아래와 같이 나타낼 수 있다.

$$\sum M_o = J_P\ddot{\theta}$$

$$-(H_L + H_R)l\cos\theta - (V_L + V_R)l\sin\theta = J_P\ddot{\theta} + (C_L + C_R) \quad (12)$$

위의 식(10), (11), (12)을 이용하여 아래와 같은 식을 얻어낸다.

$$\gamma\ddot{\theta} + (C_L + C_R) + M_P g l\sin\theta - f_{dp}l\cos\theta = -M_P l\ddot{x}\cos\theta \quad (13)$$

여기서 $\gamma = J_P + M_P l^2$ 이다. 그리고 식 (13)을 $\ddot{\theta}$ 에 관한 식으로 나타내어 식 (14)을 얻어내었다.

$$\ddot{\theta} = -\frac{M_P l\cos\theta}{\gamma}\ddot{x} - \frac{C_R}{\gamma} - \frac{C_L}{\gamma} + \frac{l\cos\theta}{\gamma}f_{dp} - \frac{M_P g l\sin\theta}{\gamma} \quad (14)$$

또한 위의 식(8), (9)을 식(10)에 대입하여 아래와 같은 식(15)을 얻어 낸다.

$$\ddot{x} = -\frac{M_P l \cos \theta}{\beta} \ddot{\theta} + \frac{C_R}{\beta r} + \frac{C_L}{\beta r} + \frac{f_{drR}}{\beta} + \frac{f_{drL}}{\beta} + \frac{f_{dp}}{\beta} + \frac{M_P l \dot{\theta}^2 \sin \theta}{\beta} \quad (15)$$

여기서 $\beta = M_P + (1 + \Omega) \left(M_w + \frac{J_w}{r} \right)$ 이다. 결론적으로 위의 전체 밸런싱 로봇 시스템에 대한 식을 종합하여 보면 아래와 같이 정리할 수 있다.

$$x_1 = x = \text{Displacement}$$

$$x_2 = \dot{x} = \text{Displacement velocity}$$

$$x_3 = \theta = \text{Pendulum's angle}$$

$$x_4 = \dot{\theta} = \text{Pendulum's angular velocity}$$

위의 식 (16)을 각각 시간에 따라 미분하여 다음과 같은 식을 얻어낼 수 있다.

$$\frac{d}{dt} x_1 = \dot{x} = \text{Displacement velocity}$$

$$\frac{d}{dt} x_2 = \ddot{x} = \text{Displacement acceleration 식(15)}$$

$$\frac{d}{dt} x_3 = \dot{\theta} = \text{Pendulum's angular velocity}$$

$$\frac{d}{dt} x_4 = \ddot{\theta} = \text{Pendulum's angular acceleration 식(16)}$$

3.1.4 PID 제어를 통한 밸런싱 로봇의 자세제어 시스템

(1) PID 제어

PID제어란 자동제어 방식 가운데서 가장 흔히 이용되는 제어 방식으로

P : Proportional (비례), I : Integral (적분), D: Derivative (미분) 3가지 조합으로 제어하는 것으로 유연한 제어가 가능해진다. PID 제어의 기본적인 개념은 Fig 5와 같이 간단히 나타낼 수 있다.

각각의 항에 대하여 적절한 게인(Gain)값을 조절하여 제어하는 시스템이다. PID 제어의 최적의 계수는 상태와 시스템에 따라 달라지며, 이는 사용자의 시스템에 맞는 개별 제어 특성을 고려하여 구한다. 특히 가장 고려해야 할 사항은 안정된 성능, 빠른 응답과 아주 작은 정상상태 편차이다. 각각의 항에 대한 특성을 살펴보면 비례 동작(Proportional action)은 제어기는 기준 입력과 현재 값의 편차를 줄여가는 방향으로 제어하며 아래와 같은 식으로 나타낼 수 있다.

$$y(t) = KZ(t)$$

K는 비례정수로 비례동작을 강하게 할 것인가, 약하게 할 것인가를 결정하며 이를 크게 하면 빠르게 수렴하나 출력이 진동하게 되어 제어의 안정성에 악영향을 미치고, 이 수치가 낮아지면 천천히 접근하여 잔류편차가 생길 우려가 있다.

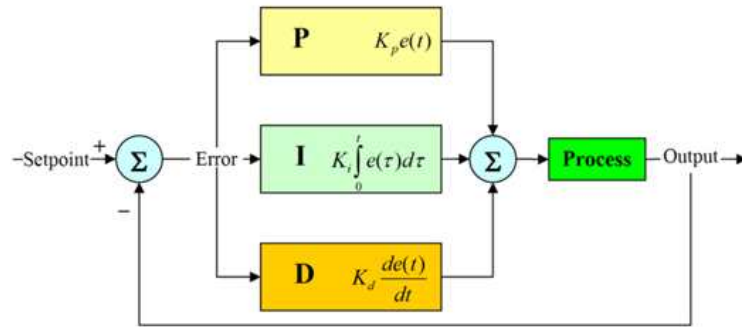


Fig. 5 PID Controller

적분동작(Integral control action)은 제어대상에 주어지는 조작량의 변화속도가 동작신호에 비례하는 동작으로 아래와 같은 식으로 나타낼 수 있다.

$$y(t) = KZ(t)dt$$

적분시간을 길게 하면 조작량이 . 하지만 적분시간이 짧으면 조작량이 적어지고 따라서 기준치에 접근하는 시간이 길어진다. 하지만 적분시간이 짧으면 조작량이 많아지게 되어 기준치에 접근하는 시간이 짧아진다.

미분동작(Derivative action)은 조작량 $y(t)$ 가 동작신호 $z(t)$ 에 미분동작을 하고 아래와 같은 식으로 나타낼 수 있다.

$$y(t) = K \frac{dZ(t)}{dt}$$

미분동작은 편차의 변화율에 상응하는 조작량을 연산하여 편차의 변화를 억제한다.

(2) 자세제어 시스템

앞서 모델링한 결과를 RK4를 이용하여 수치 해석적으로 풀이한다. 하지만 이 과정은 사람의 손으로 계산하기는 오랜 시간이 걸리게 되므로 컴퓨터를 이용한 연산을 하였다. Visual C++를 이용한 RK4의 연산과 또한 PID 제어를 이용한 자세 제어 시스템을 구성하였다. 자세 제어 시스템은 아래 그림과 같이 구성하였다.

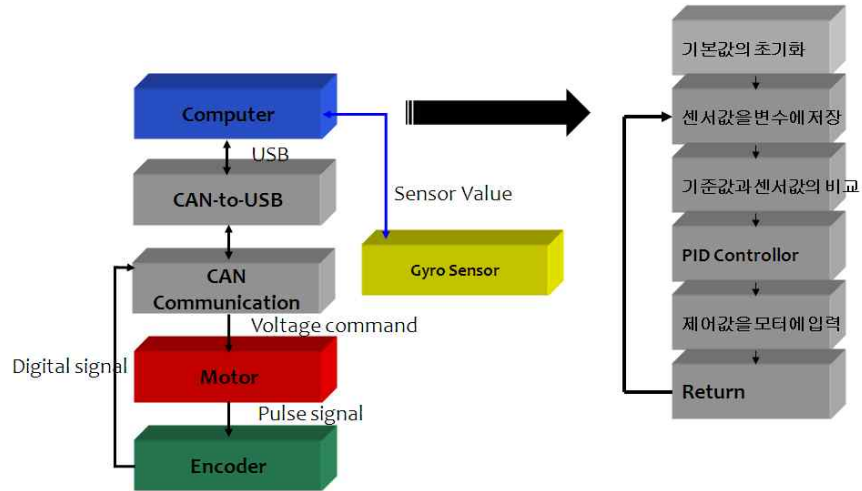


Fig. 6 PID Controller algorism

(3) 시뮬레이션 결과

위의 자세제어 시스템을 이용하여 Balancing Robot의 동작을 시뮬레이션 하였다. 자세제어에 들어가는 직접적인 힘은 모터의 토크 값으로서 모터 최대 토크의 한계를 주어 시뮬레이션 하였고, 또한 모터 최대 토크의 20%에 해당하는 외란을 발생시켰다. 시뮬레이션 결과는 아래 그림과 같다.

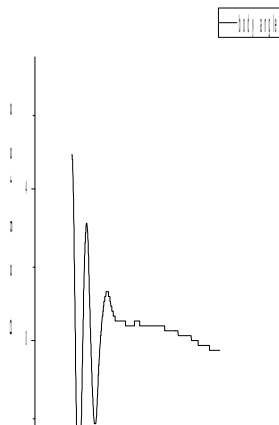


Fig. 7 Body Angle

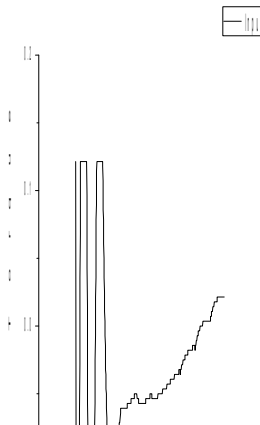


Fig. 8 Input Torque

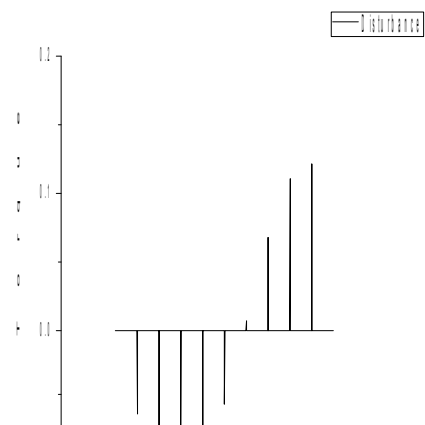


Fig. 9 Disturbance

Fig 7에서 보면 결과 값이 3초 이후로 수렴해 나가는 것을 볼 수 있다. 또한 위의 시뮬레이션을 통하여 최대 제어할 수 있는 밸런싱 로봇의 각도는 7° 라는 것을 유추해 내었다.

(4) 레고 마인드스톤 NXT를 이용한 선행 학습

하드웨어의 구성 전까지 레고 마인드스톤 NXT를 이용하여 밸런싱 로봇을

구현하였다. 레고 마인드 스톰의 구성은 밸런싱 로봇과 비슷한 구성을 지니고 있다. 프로그램을 연산하는 메인 프로세서(Main Processor)와 또한 각각의 센서 및 컴퓨터와 통신할 수 있는 블루투스(Bluetooth)로 구성되어있다. 프로그램은 레고 마인드 스톰내의 전용프로그램을 사용할 수 있고, NXC 및 Robot C와 같은 프로그램을 사용하여 코딩할 수 있다.

밸런싱 로봇을 구현하기 위하여 기본적으로 사용한 센서는 자이로 센서와 빛센서를 이용하였다. 자이로 센서의 값이 항상 양의 값을 검출하여 이를 보완해 줄 수 있는 빛 센서를 바닥과 밀접하게 장착하여 하드웨어를 구성하였다. 또한 NXC를 이용하여 프로그램을 코딩하였다. NXC는 Not Exactly C의 약자로 기본적인 언어는 C와 비슷하지만 컴파일러가 다르다.

레고 마인드 스톰NXT를 이용하여 밸런싱 로봇을 구현하였지만, 자이로 센서의 레졸루션과 빛 센서의 불안정 등의 원인으로 인하여 최대 밸런싱 구현 시간은 20초를 넘기지 못하였다. 하지만 이를 통하여 기본적인 밸런싱 알고리즘을 완성하였고, 하드웨어에 대한 감각을 익혔다.

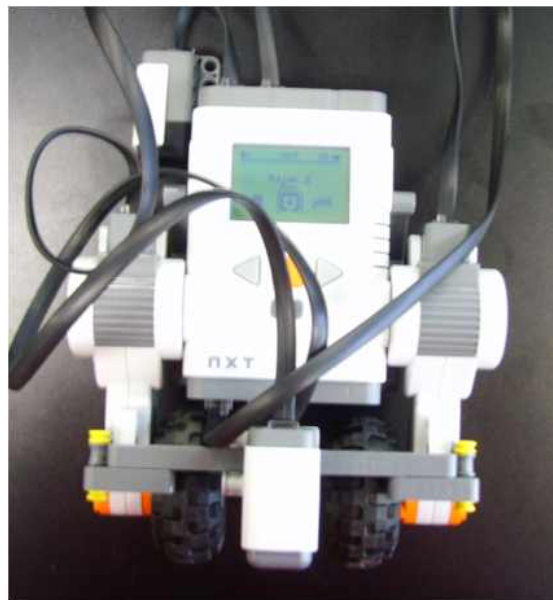


Fig. 10 Balancing Robot(LEGO Mindstorm)

3.1.5 하드웨어의 구성

밸런싱 로봇의 이동을 담당하는 모터파트의 구성은 크게 두 가지로 나누어 볼 수 있다. 액추에이터(Actuator)와 이를 브레인 모듈(Brain Module)과 연결해 주는 통신(Communication)이다. 또한 액추에이터는 모터(Motor)와 엔코더(Encoder), 또한 이를 제어할 수 있는 모션 컨트롤러(Motion Controller)로 이루어져 있으며, 통신은 캔통신(CAN Communication)을 사

용하였다.

(1) 액추에이터(Actuator)

기본적으로 모터는 DC서보모터(Servo Motor)를 사용하였다. 서보(Servo)라 함은 ‘추종한다, 따른다. 는 의미이며, 사용자의 명령을 따르는 모터를 서보모터라 한다. DC서보모터는 무조건 전류에 비례하는 토크를 내기 때문에 제어가 아주 쉬워서 서보응용에 매우 많이 쓰인다. DC서보모터의 치명적인 단점은, 모터 안에 있는 브러시와 정류자의 기계적인 접촉전환을 이용하여 회전력을 유지하기 때문에, 브러시나 정류자가 쉽게 손상된다는 점입니다. 이 단점을 극복한 것이 영구자석동기전동기(PMSM)인데, 반도체 소자를 이용한 인버터를 사용하여 회전자계를 만들어주기 때문에 모터 자체가 닳거나 부서지지 않는 한 반영구적으로 사용할 수 있다. 다만, 직류모터에는 없는 인버터를 사용하기 때문에 구동시스템이 복잡해진다는 단점이 있다.

AC서보모터는 서보시스템에 BLDC 모터나 동기전동기를 사용한 경우 이들 모터를 지칭하는 말이다. DC Servo Motor에 비해 구동 시스템이 복잡하다는 단점이 있지만, 같은 크기의 DC 모터에 비해 더 큰 힘을 낼 수 있고 방열특성이 좋으며 브러시와 정류자가 없어서 수명이 길다는 장점이 있다.

(2) 캔통신(CAN Communication)

통신은 캔통신(CAN Communication)을 이용하였다. CAN은 Controller Area Network의 약자로써 계측제어통신망이라는 뜻이다. 캔통신은 1988년 Bosch와 Intel에서 개발된 차량용 네트워크시스템으로 마이크로 컨트롤러들 간의 통신을 위해 설계된 시리얼 네트워크 통신방식이다. 이는 1993년도에 ISO에서 국제표준규격으로 제정되었다. 캔통신의 아래의 그림과 같이 구성되어 있다.

그림을 통하여 볼 수 있듯이 두 개의 선으로 되어 있으며 두 선의 전압 차이에 의하여 데이터를 읽는 방식이다. 간단한 구성으로 되어 있으며, 하나의 캔 버스(CAN BUS)를 이용하여 여러 장치를 연결할 수 있는 장점이 있다. 또한 데이터를 우선순위에 따라 처리할 수 있으며, 방대한 양을 처리할 수 있는 장점이 있다.

캔 통신의 동작 원리를 살펴보면 일단 캔 버스의 사용 여부 및 메시지 총

돌을 검사 하고 이상이 없을 경우 식별자(ID)를 통하여 메시지를 선별하게 된다. 이렇게 식별된 메시지는 우선순위를 결정하게 되며, 낮은 순위의 메시지가 우선적으로 전송되는 시스템이다.

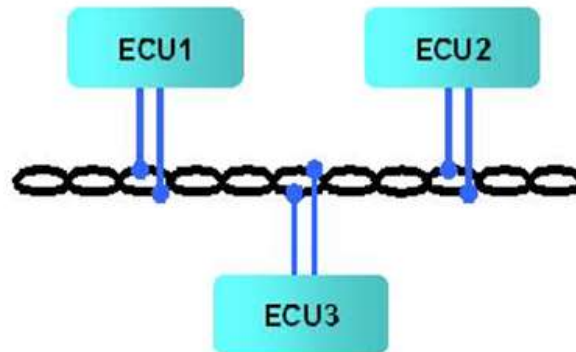


Fig. 11 CAN Communication block diagram

(3) 하드웨어의 연결

컴퓨터와 모터와의 연결은 캔통신을 통하여 바로 연결할 수 없기에 USB to CAN 컨버터(Converter)를 사용하여 Fig.11 과 같이 연결하였다. 캔통신에 필요한 프로그램은 Visual C++를 이용하여 코딩하였다. 또한 각각의 하드웨어에 따라 필요로 하는 전압이 다르기 때문에 이를 해결하기 위하여 DC-DC Converter를 이용하여 전압을 조절하였다.

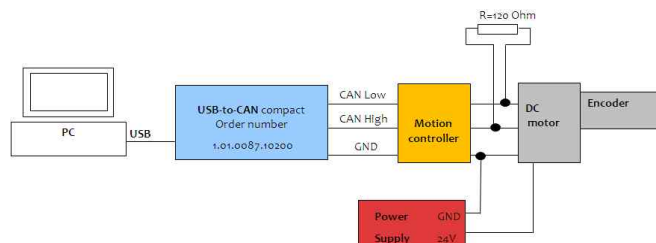


Fig. 12 Free body diagram of the robot

3.2. 밸런싱 로봇의 센서 인터페이스

3.2.1 서론

두 바퀴를 이용하여 균형을 잡으면서 구동하는 밸런싱 로봇을 제작하기 위해서는 필수적으로 로봇의 기울기를 감지할 수 있는 센서가 필요하다. 또 장애물을 피하면서 주행하기 위해서는 물체를 식별할 수 있는 센서가 필요하다. 적절한 센서의 선정과 보상 알고리즘의 적용을 통해 밸런싱 로봇의

자세제어와 장애물 회피를 구현해 본다.

3.2.2 밸런싱 로봇의 자세제어를 위한 센서 선정의 착안점

두 바퀴를 이용하여 균형을 잡는 밸런싱 로봇의 기본 개념은 로봇이 넘어지려는 방향으로 바퀴를 움직여 로봇 몸체의 무게 중심과 바퀴의 회전 중심을 바닥에 수직인 일직선상에 오도록 제어하여 로봇이 넘어지지 않게 하는 것이다. 이를 위해서는 로봇이 기우는 정도와 방향을 알 수 있는 센서가 필요하다. 각도 정보를 바로 알 수 있는 센서를 사용하는 것이 가장 좋은 방법이지만 센서 특성에 따른 문제점 때문에 각속도 정보를 알 수 있는 자이로 센서와 가속도 정보를 알 수 있는 가속도센서를 적절히 활용하여 자세 제어를 구현해 보았다.

3.2.3 센서의 선택 및 구매

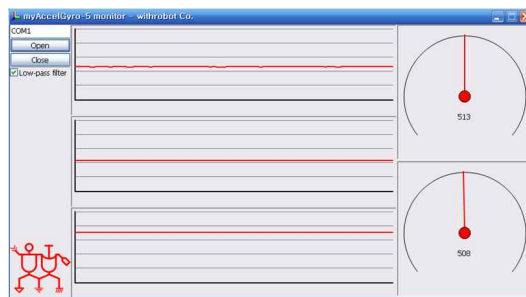


그림 23 센서 모듈의 테스트 프로그램

센서를 구매함에 있어서 먼저 몇 가지 모델을 먼저 선정하여 보았다. 센서 제품은 크게 3가지로 나누어졌는데 자이로 센서 단품과, 센서 All-In-One, 자이로 센서 모듈을 놓고 선택하였다. 센서 모듈은 저렴한 가격이 장점이었지만 실제 모바일 로봇에 적용하기 위해서는 가속도 센서의 추가 구매와 안정적인 데이터 사용을 위한 필터를 추가로 구입하여 제작해야하는 단점이 있었다. 다음으로 자이로 센서모듈은 성능이 보장되어 있고 모듈화 되어 있어서 높은 신뢰도를 가지는 데이터 값을 전송하고, 즉시 사용이 가능한 장점이 있는 반면에 가속도 센서를 따로 다시 구매하여 추가로 장착하여야 하고, 가격이 높다는 단점이 있었다. 마지막으로 센서 All-In-One은 3축 가속도 센서와 2축 자이로센서가 결합된 제품으로 필터가 내장되어 있어 안정된 데이터를 수신할 수 있었다. 소프트웨어적으로 수신된 데이터를 사용 환경에 맞게 변환시켜야하는 단점이 있었지만, 세 가지 품목 중 가장 개발

이 간편하고 로봇이 점점 개선됨에 따라 발생 할 수 있는 추가적인 필요 기능도 만족하여 최종적으로 센서 All-In-One제품을 선정하였다.

그림 1의 왼쪽 위에서부터 반시계 방향으로 X축 방향 가속도, Y축 방향 가속도, Z축 방향가속도, Pitch 각속도, Yaw 각속도를 나타낸다.

3.2.4 시리얼통신

구입한 센서 All-In-One모듈은 컴퓨터와 인터페이스를 하기 위해서는 시리얼 통신을 필요로 한다. 시리얼 통신에는 동기통신과 비동기통신의 2종류가 있다. 동기통신의 경우, 2개의 디바이스 사이에서 동기를 취하고 그 타이밍에 따라 데이터를 송수신한다. 데이터의 교환이 없는 사이도 제어용의 신호가 흐르고 있으므로 상대와의 동기를 유지하는 것이 가능하다. 실제 데이터를 송수신 할 때는 그것을 송수신하고 데이터가 없는 때에는 대기 상태를 나타내는 신호를 교환한다. 이처럼 통신이 확립되면 실 데이터를 송수신한 것에 데이터의 시작과 종료를 나타내는 신호가 존재하지 않기 때문에 전송속도는 빨라지게 된다. PC의 시리얼 포트는 비동기 장치이다. 그러므로 비동기 시리얼 통신만 지원한다. 비동기란 동기의 반대로, 송신과 수신 아이들(idle)문자가 필요 없다. 그러나 데이터의 처음과 끝에는 반드시 스타트비트와 스톱비트가 붙는다. 스타트비트는 데이터의 시작을 나타내고 스톱비트는 데이터의 종료를 나타낸다. 따라서 이들 두 개의 비트가 추가되기 때문에 비동기 통신의 속도는 동기 통신에 비해 약간 늦어지게 된다. 하지만, 프로세서는 대기상태 때 여분의 아이들(idle) 문자를 처리할 필요가 없게 된다.

밸런싱 로봇을 제어하기 위한 응용 프로그램은 VC++에서 동작되기 때문에 VC++기반의 시리얼 통신 프로그램을 오픈소스를 제공하는 인터넷 사이트 데브피아에서 다운받아 개발 환경에 맞게 변환하여 시리얼 통신을 구현하였다.

3.2.5 자이로 센서와 기울기 센서

(1) 자이로 센서와 가속도 센서의 특징

가속도 센서는 내부 구조가 액체 또는 스프링 형태로 되어 있어 관성의 영향을 많이 받고, 외부의 작은 충격에도 가속도 성분이 더해져 외란에 민감하게 반응하기 때문에 실질적인 기울기 값을 획득하기에는 어려운 점이

많다. 하지만 기울기 센서는 지구의 중력에 대해 절대적인 기울임 값을 측정할 수 있는 장점이 있다. 외란에 강한 회전 측정 센서에는 자이로 센서가 있으나 자이로 센서는 각도를 바로 측정할 수 있는 센서가 아니라 각속도를 정보를 측정하는 센서이다. 따라서 우리가 필요로 하는 각도 정보는 각속도를 컴퓨터로 실시간 적분 후 원하는 각도 정보를 얻을 수 있다. 자이로 센서의 이와 같은 특성은 시간이 지남에 따라 적분 시 발생하는 누적오차를 발생시킨다. 또 양자화, 온도, A/D변환 등의 요인에 대해서도 오차가 발생하는 단점이 있다.

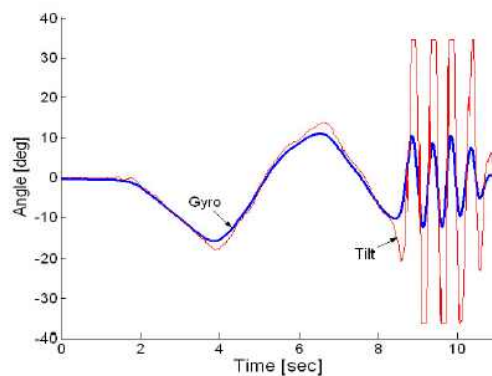


그림 24 외란에 따른 가속도 센서의 특징

그림 2에서와 같이 가속도 센서에서 추출한 기울기 값은 느린 움직임에서는 자이로 센서에서 추출한 기울기 값과 거의 차이를 보이지 않지만 빠른 움직임에서는 자이로 센서와 많은 차이를 보인다.

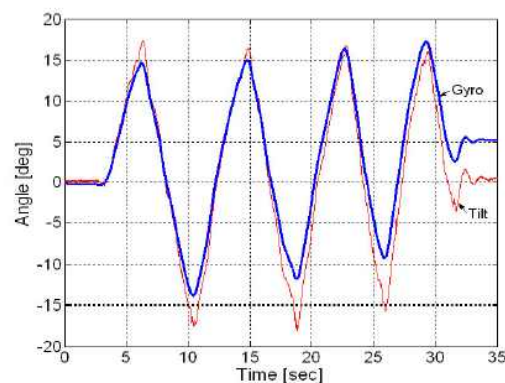


그림 25 시간에 따른 자이로 센서의 특징

그림 3에서와 같이 가속도 센서에서 추출한 기울기 값은 시간이 경과 하더라도 평행 상태에서 0값을 출력한다. 그러나 자이로 센서에서 적분을 통해 추출한 기울기 값은 시간이 지남에 따라 오차가 생김을 알 수 있다.

(2) 자이로 센서와 가속도센서의 적용

밸런싱 로봇의 수평유지를 위해 자이로 센서를 단독으로 사용하여 제어하는 경우 초기에는 균형이 잘 잡힌 채 서 있지만 시간이 지남에 따라 자이로 센서의 누적오차로 인해 점차적으로 한쪽 방향으로 기울어지게 된다. 그리고 가속도 센서를 단독으로 사용하여 제어하는 경우 중력에 대하여 기울임 각도를 절대적으로 알 수 있어 제어가 용이하나 밸런싱 로봇에 외란이 가해지면 심하게 요동하게 된다. 이것은 지금까지 개발된 가속도 센서의 내부 구조는 액체 또는 스프링 방식을 사용하므로 기울임 각도 뿐 아니라 외란에 의한 관성력에도 민감하게 반응하기 때문이다. 반면 각속도 정보를 출력하는 자이로 센서는 외란이나 기타 작용되는 가속도성분에 민감하게 반응하지 않기 때문에 각속도 정보를 적분하여 절대적인 기울기 정보로 활용되고 있다. 하지만 시간이 지남에 따라 발생하는 누적오차를 실시간 보정해 주어야 한다.

이러한 문제점을 해결하기 위하여 자이로 센서와 가속도 센서를 상호 융합하여 실시간 자이로 센서의 누적오차를 보상하여 안정적으로 밸런싱 로봇을 제어하려고 한다.

3.2.6 각도 값 추출

(1) 자이로 센서에서 각도 값 추출

자이로 센서는 각속도 정보를 출력한다. 이것을 각도 정보로 변환하기 위해서는 각속도 정보를 적분하여야한다. 자이로 센서의 각속도 정보를 적분하여 각도 정보로 변환하는 과정을 나타내면 다음과 같다.

$$\theta_{GYRO}[k] = K_{GYRO} \sum_{j=1}^k (w_{GYRO}[j] - K_{GYROREF})$$

여기서 자이로 센서의 출력 정보인 각속도 w_{GYRO} 는 센서가 특정 방향으로 회전하지 않고 고정되어 있으면 중간 값(513)을 출력하고 자이로 센서가 특정 방향으로 회전하면 중간 값에서 영의 값 또는 음의 값만큼 합산되어 출력된다. 각속도 정보를 각도 정보로 변환하기 위하여 자이로 센서의 각속도 w_{GYRO} 에서 고정 상태에서의 센서 출력 값, $K_{GYROREF}$ 값을 빼면 0레벨로 변환된다. 이후 적분 과정을 행한 후 이득상수, K_{GYRO} 를 곱해주면 원하는

각도 정보로 변환된다. 자이로 센서의 각 θ_{GYRO} 정보를 얻기 위한 이득 상수 K_{GYRO} 는 $1/\text{sampling time}$ 으로 둔다.

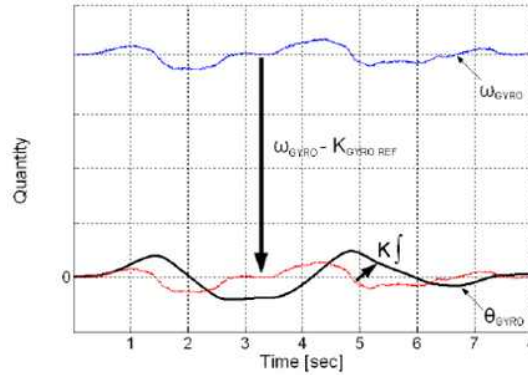


그림 26 자이로센서 에서의 각도 값 추출

(2) 가속도 센서에서 각도 값 추출

가속도 센서에서 중력가속도(1g)는 항상 수직방향으로 작용한다. 따라서 Z축에서의 가속도의 변화를 통해 정지 상태에서의 기울기 변화를 알 수 있다.

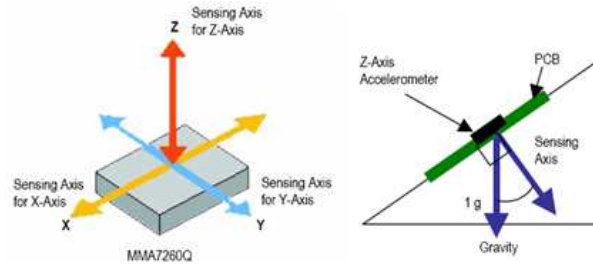


그림 27 기울기 변화에 따른 가속도 값의 변화

역진자의 기울임 각도를 수식으로 표현하면 다음과 같다.

$$A_{OUT} = A_{OFFSET} + \left(\frac{\Delta A}{\Delta g} \times 1g \times \sin\theta \right)$$

$$\theta_{TILT} = \arcsin\left(\frac{A_{OUT} - A_{OFFSET}}{\frac{\Delta A}{\Delta g}} \right)$$

이 식에서 A_{OUT} 는 가속 시 센서에서 출력되는 가속도 값을 나타내고 A_{OFFSET} 은 0g일 때 센서에서 출력되는 가속도 값을 나타낸다. $\frac{\Delta A}{\Delta g}$ 는 1g당 가속도 값을 나타내고 여기에서 1g는 지구에서의 중력가속도이다.

밸런싱 로봇의 PID제어를 위하여 목표 각 θ_{REF} 에 대한 오차 e_θ 는 다음과 같이 정의하여 구한다.

$$e_{\theta}[k] = \theta_{REF}[k] - \theta_{GYRO}[k]$$

이 식에서 목표 각 θ_{REF} 에서 자이로 센서의 각 θ_{GYRO} 을 빼면 오차 e_{θ} 를 구할 수 있다.

초기에 밸런싱 로봇이 수평을 유지할 때 자이로 센서의 각 θ_{GYRO} 은 0을 나타내지만 시간이 지남에 따라 변화되는 적분에 따른 자이로 센서의 누적 오차로 인하여 센서의 출력 값이 증가하거나 감소하게 된다. 따라서 누적되는 자이로 센서의 오차를 보정하기 위해 보정 각 θ_{COMP} 를 정의한다.

$$\theta_{COMP}[k] = K_{COMP} \sum_{j=1}^k (\theta_{TILT}[j])$$

기울기 센서의 각 θ_{TILT} 을 실시간 적분하여 이득상수 K_{COMP} 를 곱해주면 자이로 센서의 보상 각 θ_{COMP} 를 얻을 수 있다. 이 식에서 기울기 센서의 각 θ_{TILT} 은 모바일 로봇이 수평을 유지하면 0의 값을 가진다. 또한 K_{COMP} 는 오차 보정 상수로 실험적으로 결정한다.

θ_{COMP} 를 사용하여 오차 e_{θ} 를 자이로 센서의 누적오차에 따른 영향을 최소화하면 다음과 같은 식으로 정리 된다.

$$e_{\theta}[k] = \theta_{REF}[k] - \theta_{GYRO}[k] + \theta_{COMP}[k]$$

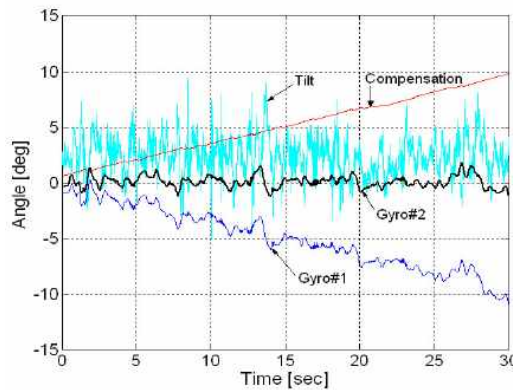


그림 28 자이로 센서의 누적 오차 보상

3.2.7 레이저 스캔 센서

(1) 레이저 스캔 센서의 특징

로봇 주위의 장애물을 파악하여 장애물과의 충돌을 방지하고 원하는 목적지까지 안전하게 주행하기 위해서는 물체를 감지 할 수 있는 센서가 필요하다. 이때 선택하는 가장 대중적인 선택은 적외선 센서나 초음파 센서이다. 적외선 센서나 초음파 센서는 저렴한 가격으로 구입이 가능하고 제품을

생산하는 곳이 많아 다양한 스펙의 제품을 구하기가 쉽다는 장점이 있다. 하지만 대부분의 경우 빛이나 소리를 한정된 범위 안에서 주고받기 때문에 넓은 범위를 인식하기 위해서는 로봇의 몸체에 다수의 센서를 설치하여야 하고 빛이나 소리의 분산 때문에 먼 거리의 물체를 파악하는 것이 힘들다는 단점이 있다. 이에 비해 레이저 스캔 센서는 상대적으로 판매하는 회사가 적고 가격이 비싸다는 단점이 있지만, 레이저의 직진성이 좋아 상대적으로 훨씬 더 멀리 있는 물체를 감지 할 수 있고, 분산이 잘 되지 않기 때문에 측정영역에 대해 상대적으로 정밀하게 측정할 수 있다. 그리고 측정할 수 있는 범위가 넓기 때문에 하나의 센서로 넓은 범위를 측정할 수 있다. 레이저 스캔 센서의 원리는 기준신호와 물체에 반사되어 돌아오는 레이저 광 신호간의 시간차이를 이용하여 거리를 측정한다.

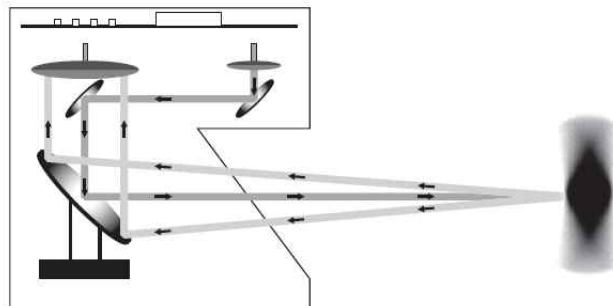


그림 29 레이저 스캔 센서의 원리

(2) 레이저 스캔 센서의 선정

레이저 스캔 센서의 구매에 앞서 다음과 같은 몇 가지 모델을 선정하여 비교하였다.

	rotoScan (로이체)	LMS200 (SICK)	URG-04LX (HOKUYO)	UTM-08LX (HOKUYO)
측정각도(°)	190	180	240	270
측정거리(m)	0~65	0~80	0.6~4	0.1~8
주사시간 (ms/scan)	40	40	100	67
인터페이스 형식	RS232/422	RS232/422	USB RS232	USB RS232
크기(W*D*H) /무게	155*156*210 /2000g	155*156*210 /2000g	50*50*70 /160g	86*83*83 /370g



그림 30 선정된 레이저 스캔
센서 UTM-08LX

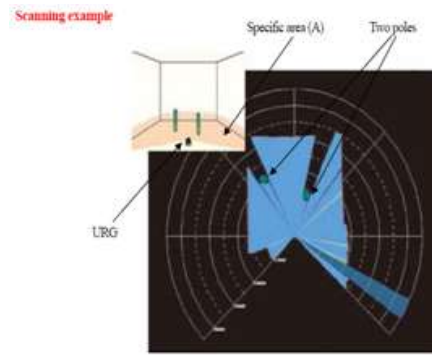


그림 31 레이저 스캔 센서의 작동
모습

레이저 스캔센서는 대부분이 일본이나 독일에서 만든 레이저 스캔센서를 사용하고 있었다. 과거 한국생산기술 연구원에 자문을 구하고 로봇 주행에 관한 논문이나 참고자료를 조사하여 현재 판매되고 있는 제품 중 HOKUYO의 UTM - 08LX 모델을 선정하였다. 선정 기준은 측정 각도와 측정거리 응답시간, 인터페이스 형식과 등을 고려하였다. 또 너무 무겁거나 크기가 크면 밸런싱을 잡기가 어려운 로봇의 특성을 고려하여 크기가 작고 가벼운 제품을 선정하였다. HOKUYO의 UTM-08LX 레이저 스캔 센서는 8m라는 적당한 측정거리와 USB2.0 인터페이스를 사용하고 다른 레이저 스캔센서에 비해 가볍고 작은 장점을 가지고 있다.

3.2.8 결론

이번 보고서는 밸런싱 로봇의 구현과 장애물 회피를 위한 센서 인터페이스에 관한 내용이다. 역진자 방식의 자세제어를 위하여 로봇의 정확한 자세의 기울어짐을 검출하는 것이 필요하기 때문에 자이로 센서와 가속도 센서를 사용하였고, 로봇이 주행하면서 장애물을 만났을 때 회피할 수 있도록 레이저 스캔 센서를 사용하였다. 모바일 역진자의 수평유지를 위해 가속도 센서를 단독으로 사용할 경우 처음에는 쉽게 균형을 유지하지만 점차 몸체가 심하게 진동을 하면서 시스템이 불안정하게 되며 자이로 센서를 단독으로 사용할 경우에는 시간이 지남에 따라 누적 오차가 더해서 실질적인 밸런싱 로봇의 제어가 어렵게 된다. 본 보고서에서는 상용화된 2축 자이로 센서와 3축 가속도 센서 모듈을 이용하여 시스템을 구성하였으며 센서퓨전 기술과 동적 기울기 보상 알고리즘을 통해 밸런싱 로봇의 안정적인 수평유

지가 가능할 것이라고 기대하였다. 하지만, 실제 모바일 로봇에 적용시켜 본 결과 노이즈 등 예상치 못한 문제점으로 인하여 정확한 자세제어가 불가능 하였다. 따라서 앞으로 로봇의 무게를 경량화 시키고 로봇의 자세 변화에 따른 정확한 센서 값의 검출을 구현한다면 밸런싱 로봇의 자세제어가 가능할 것으로 예상된다. 이번 보고서에서 제시한 수평유지 기법은 밸런싱 로봇 뿐 만 아니라 항공기, 선박, 자동차 등의 분야에서도 유용하게 활용될 것으로 기대된다.

3.3 자율주행로봇 시뮬레이션

3.3.1 시뮬레이션의 목적

실제 모델의 메커니즘을 구현하기 이전에 여러 가지 형태로 가상 시뮬레이션을 해 봄으로써 실제 로봇 구동시 발생 할 수 있는 문제점을 사전에 파악하고 해결하여 실제 메커니즘에 적용을 시키는 것을 목적으로 한다. 이렇게 하여 실제 모델을 작동 시켰을때 발생하는 문제점으로 인해 값비싼 장비들이 손상이 가는 것을 막고 메커니즘을 구성할 때 에도 시간을 절약시킬 수가 있으며 최적의 메커니즘을 설계하는데에 도움을 줄 수가 있다. 또한 실제 실험이 아닌 소프트웨어적인 실험이라는 것 자체에도 의미가 있다.

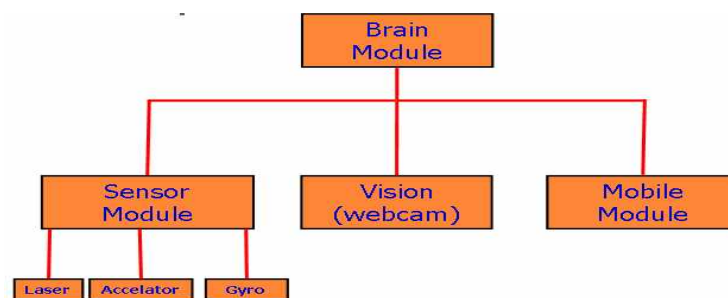


Fig.1 로봇 전체시스템 구성도

3.3.2 전체 구성

(1) 시스템 구성

전체 시스템은 아래의 Fig.1처럼 Brain Module , Sensor Module ,

Mobile Module 이렇게 크게 3가지로 분류가 되고 Brain Module이 로봇의 머리에 해당하는 부분으로 모든 각종 장비간의 통신을 이루어 질 수 있도록 하는 매개체 역할을 함과 동시에 Sensor Module의 여러 센서로부터 받아들이는 센서 값을 바탕으로 Mobile Module에 명령을 주어 모터를 작동시키는 역할을 하고 영상장치인 웹캠으로부터 영상정보를 받아들이어 전송을 하는 역할을 하기도 한다.

(2) 각 Module의 구성요소 및 역할

Brain Module은 실제로 로봇에 있어서 두뇌와 같은 역할을 하는 것이며 실제 메커니즘 구성을 할때 SBC(Single Board Computer)를 사용을 하고 이 SBC의 역할은 각종 장비의 간에 통신을 하기위한 매개체로서의 역할을 하고 각각의 센서로부터 읽어들이는 값을 읽어들이어 구성된 알고리즘을 실행시키고 알고리즘 결과에 해당되는 모터의 각속도 값을 모터에 명령을 내려 주는 역할을 한다.

Sensor Module은 2가지로 분류를 할 수가 있는데 일단 로봇과 장애물과의 거리를 판단하기 위하여 사용되는 Laser Sensor 그리고 지금 로봇이 얼마나 기울어져 있는지를 알수 있게 해주는 Gyroaccelerator Sensor가 있다. Laser Sensor는 일종의 거리센서인데 다른 거리센서들에 비해서 오차가 크지 않고 정확한 값을 읽어 들인다는 장점이 있고 이 센서로 통해 장애물과의 거리를 판단하게 되고 장애물회피 알고리즘을 구현하는데 핵심이 되는 센서이다. 다음으로 Gyroaccelerator Sensor는 Gyro와 Accelerator의 기능을 동시에 하는 센서로서 로봇이 지면으로 하여금 얼마만큼의 각속도로 변화하고 있는지를 측정을 해주기 때문에 이 값을 알고리즘 구현 시에 적분을 하고 오차를 보상해 줌으로써 지면으로부터 얼마나 기울어져 있는지를 각도로 알 수가 있고 현재 로봇의 가속도가 얼마인지를 3축으로 알 수 있게 해줌으로써 이 값을 바탕으로 알고리즘을 구현하여 로봇의 balancing에서 핵심이 되는 센서이다. 따라서 추후에 로봇에서 캐스터를 제거하고 양쪽 두바퀴로 구동을 했을때 이 센서의 역할을 아주 커지게 된다.

마지막으로 Mobile Module은 양쪽의 두 바퀴를 말하는 것인데 이 두 바퀴는 DC모터로 구동이 되며 모터의 각속도를 알수 있게 하기 위하여 엔코더가 부착되어있고 SBC와 통신을 하기 위하여 드라이버도 또한 설치가 되어있다. 엔코더와 드라이버를 통하여 모터의 각속도를 측정하여 이에 대한 정보를 SBC에 보낼 수가 있고 Sensor Module에 따라 명령을 받는 것이

아니라 Mobile Module 자체적으로도 SBC를 통해 제어가 가능하다. 그리고 모터는 없지만 로봇의 안정성을 높여주고 로봇이 이동시에 앞뒤로 쓰러지는 것을 방지하기 위해 2개의 캐스터가 앞뒤로 부착되어 있다.

실제로봇의 메커니즘에서 이렇게 각각의 Module이 다른 Module과 Communication을 하기 위해서는 통신방식의 설정이나 그에 대한 알고리즘 구현이 필요한데 시뮬레이션 과정에서는 시뮬레이션 소프트웨어에서 제공하는 함수를 가지고 알고리즘을 구현을 하고 한 가지 인터페이스 안에서 모든 알고리즘을 구현하기 때문에 실제 메커니즘을 구성할 때 보다 훨씬 간단하고 시간도 많이 절약을 할 수 있다는 장점이 있다.

3.3.3 모델링

모델링은 크게 2가지로 분류가 되는데 로봇의 모델링과 그리고 로봇이 이동하게될 경로에 대한 모델링이 바로 그것이다. 조금 세부적으로는 로봇의 모델링에서 센서부분에 대한 모델링이 포함이 되고 경로모델링에서는 경로에 사용되는 Lighting이나 Texture가 포함되어 있다..

(1) 로봇 모델링

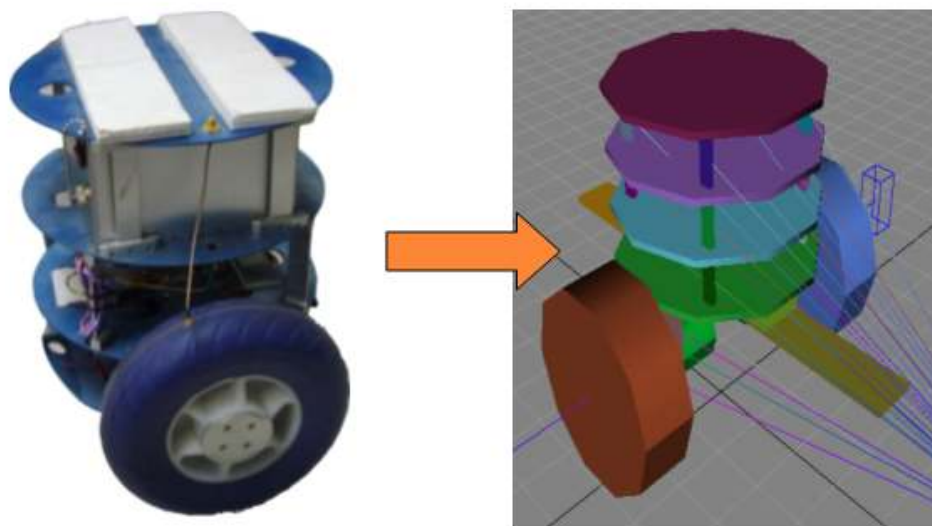


Fig.2 로봇의 모델링

위의 Fig.2에서 보는바와 같이 실제 로봇의 메커니즘의 모습을 MRS(Mariliou Robotics Studio)을 통하여 그대로 모델링을 하게 되면 MRS

시뮬레이터가 스스로 각 부분에 해당되는 관성이나 재료의 특성을 고려하여 입력에 따라 작동을 하게 되므로 각종 운동방정식이나 복잡한 계산들을 사용하지 않고도 모델이 어떻게 작동되는지를 알 수가 있다.

(2) 센서 모델링

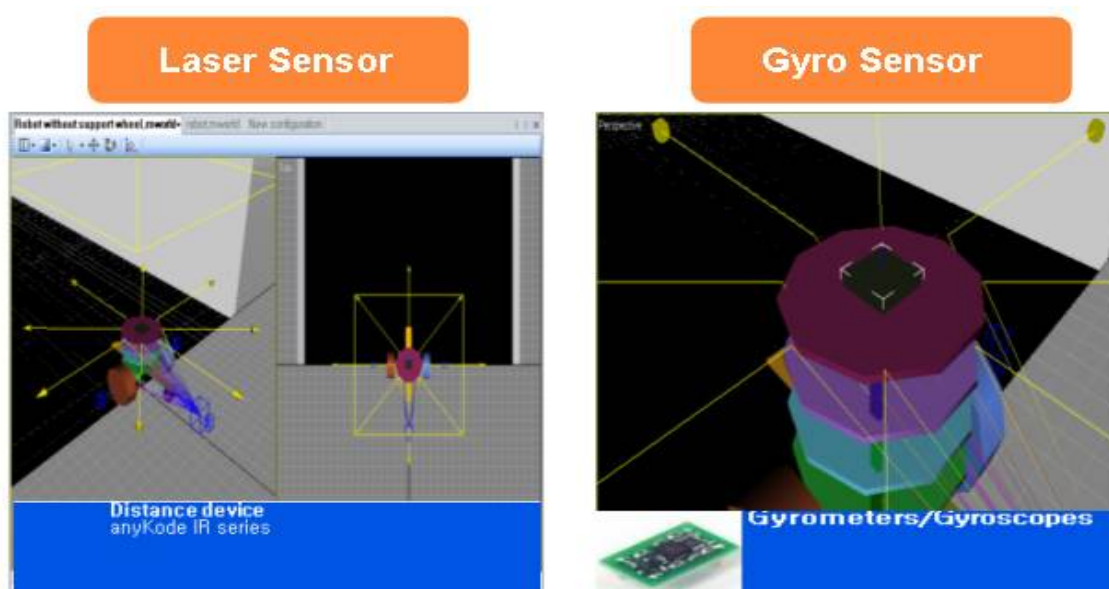


Fig.3 센서의 모델링

위의 Fig.3에서 보면 로봇에 부착되게 될 Laser Sensor와 Gyroaccelerator Sensor가 모델링이 되어 있는 것을 볼 수가 있는데 각각의 센서는 MRS에서 제공되는 Gyroaccelerator Sensor와 Distance Sensor로 모델링을 하였는데 실제 메커니즘에서 사용되게 될 센서 모델과 다를 수도 있으나 가상공간에서 똑같은 역할을 한다는 것을 기준으로 모델링을 하였다. 그리고 실제 사용되는 Laser Sensor가 270도까지 감지할 수 있는 것을 감안하여 45도 간격으로 8개의 Distance Sensor를 부착하였다. 또한 Gyroaccelerator Sensor는 로봇의 가장 높은 곳에 부착하여 앞뒤로 로봇이 흔들릴 때에 가장 민감하게 값을 측정할 수 있도록 설계를 하여 이후 실제 메커니즘 설계에도 적용을 하였다. 그리고 시뮬레이션을 할 때에 모든 작업이 가상으로 이루어지기 때문에 실제 사용되는 센서의 제원과 약간의 차이가 있더라도 시뮬레이션을 하는 목적에 영향을 미치지 않는다. 또한 이 모델을 제어하기 위해서는 Visual C++을 사용하였는데 이는 MRS와 상호

호환이 가능하기 때문이다. 따라서 모델링을 제외한 센서의 작동과 출력 그리고 Sensor Module, Brain Module과 Mobile Module 간의 통신 그리고 양쪽 바퀴의 모터를 구동시키는 명령을 모두 Visual C++상에서 구성이 되었다.

(3) 경로 모델링

로봇이 이동하게 될 경로를 모델링 하였는데 경로는 기계관4층과 전자관4층인데 로봇의 모델링 할때와 마찬가지로 우선 경로에 대해서 실측을 하여 그 실측값을 그대로 모델링을 하였는데 경로에 관한 정보와 그에 대한 모델링을 한 모습을 Fig.4에서 나타내었다.

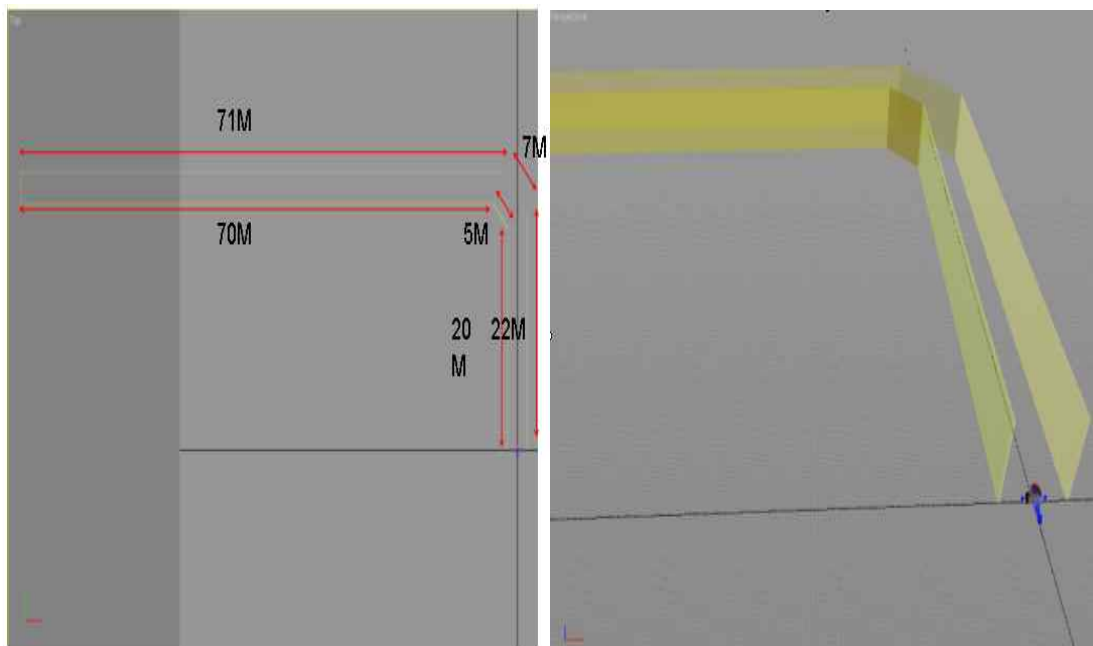


Fig.4 경로의 실측 및 모델링

위의 Fig.5에서는 경로를 모델링하는데 있어서 더욱더 실질적으로 보이기 위하여 실제 경로에 해당되는 사진을 3m X 3m 간격으로 찍어서 경로모델링 위에 부착하는 형태로 Texture작업을 하여 바닥, 천장, 벽을 실제 모습과 동일하게 구현을 하였고 또한 천장에 있는 형광등은 실제와 동일하게 2.5m간격으로 Lighting Source를 모델링하여 역시 C++상에서 제어를 하였고 이는 시뮬레이션을 할때 로봇이 이동되는 경로에 불빛이 비추어 질수 있도록 알고리즘을 구현 하였다.

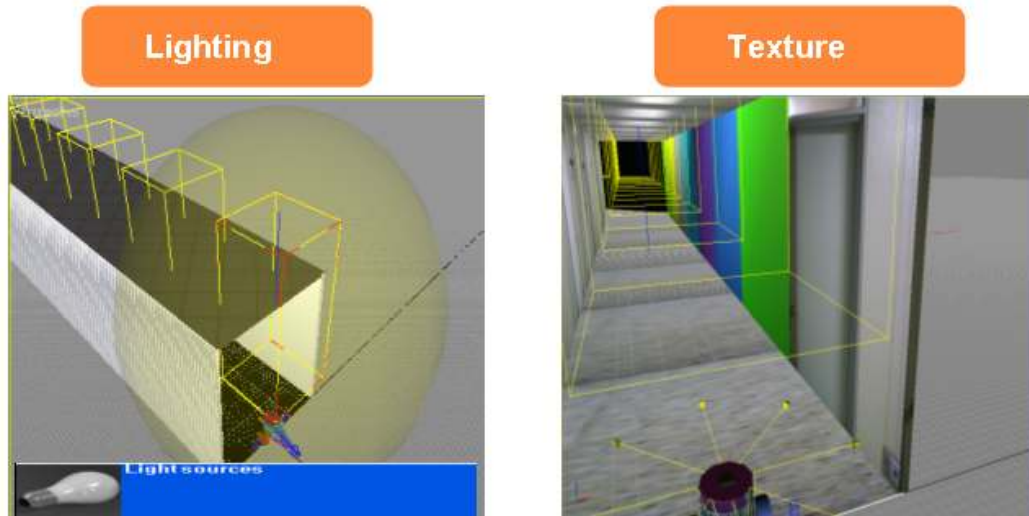


Fig.5 Lighting & Texture

위의 Fig.6는 로봇의 모델링부터 시작하여 센서, 그리고 경로, Lighting, Texture 까지 앞에서 설명한 모든 모델링을 하나로 합쳐서 최종적으로 완성된 모델링의 모습을 보여 주고 있다.



Fig.6 최종 모델링된 시뮬레이션 환경

3.3.4 장애물 회피 시뮬레이션

모든 모델링을 마친뒤에 실제적으로 이동경로를 로봇이 받아들인 센서값

에 따라 적절히 이동을 하도록 하는 알고리즘을 구성하였다. 이는 실제 DEMO를 보여주게 될 실제 상황에 적합하게 알고리즘을 구성하였으며 이 알고리즘에 대한 아이디어는 실제 로봇구동시에도 적용을 하게 된다. 알고리즘에 대한 아이디어는 크게 2가지로 구성되는데 로봇이 직선주행시, 로봇이 전방 장애물을 만나서 커브주행시가 바로 그것이다.

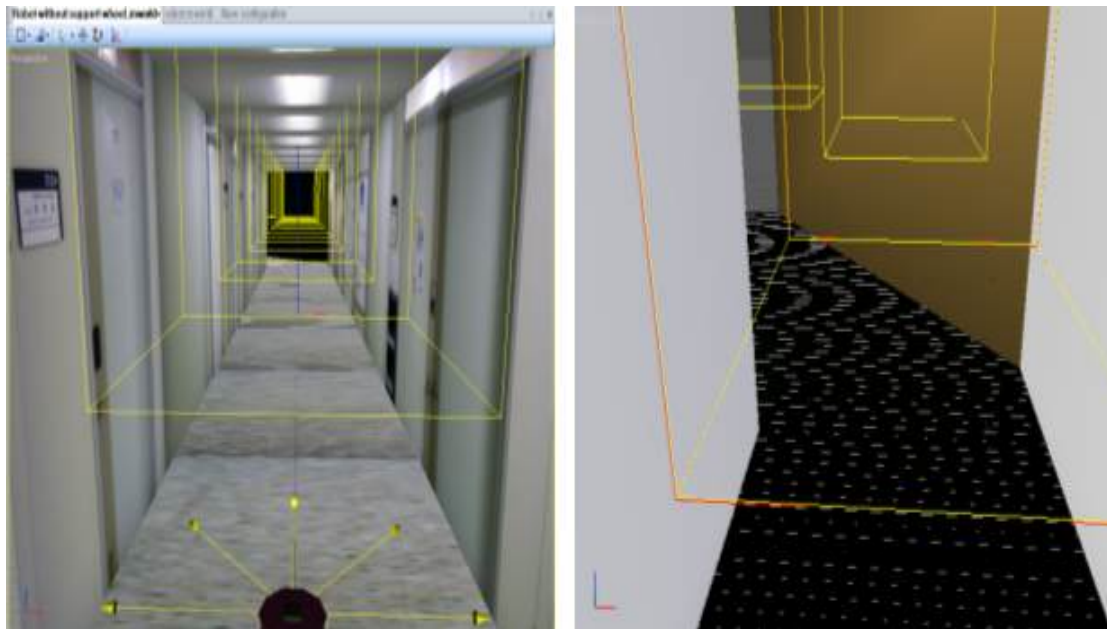


Fig. 7 직선주행시(좌)커브주행시(우)

(1) 직선 주행시

기본적으로 직선주행시에는 로봇이 통로의 중앙을 유지하면서 이동을 하는 것이 양쪽 측방의 장애물과의 접촉가능성을 가장 낮출 수 있다고 판단하여 로봇이 가운데로 이동할 수 있도록 알고리즘을 구성하였다. 우선 이 알고리즘을 구성하기 위해 90도와 180도에 위치한 Distance Sensor를 사용하였고 실제 모터의 토크를 각속도로 환산을 하여 로봇이 통로 정중앙을 유지할 때에는 양쪽 바퀴에 동일한 값 150degree/s 를 주어서 한쪽으로 치우치지 않고 통로중앙을 유지할 수 있도록 알고리즘을 구성하였다. 그리고 만약 로봇이 양쪽벽과 1.1m 이내로 근접 시에는 한쪽 바퀴의 각속도를 1degree/s 만큼 줄여서 다시 통로 가운데로 주행을 할 수 있게 구성을 하였다. 여기서 1.1m 인 이유는 센서가 부착된 곳으로부터 벽까지의 거리가 약 1.3m 인 것을 감안하여 그보다는 약간 작은 값을 선택한 것이다.

(2) 커브 주행시

커브 주행시에는 최전방에 위치한 Distance Sensor를 이용하였는데 이 센서가 2.3m이내에 어떠한 장애물을 인식하였을 때에는 모터에 각속도를 조절하여 왼쪽방향으로 회전을 할 수 있도록 알고리즘을 구성하였다. 이때에 모터의 각속도는 좌측모터가 -50degree/s 이고 우측모터가 50degree/s 로 작동을 하여 회전을 하게 된다. 여기에서 거리를 2.3m로 정한 이유는 현재 구성된 로봇의 메커니즘에서 캐스터를 부착하기 때문에 로봇은 항상 뒤로 약간 기울어진 상태로 주행을 하게 되고 이렇게 되면 레이저 센서도 로봇이 기울어진 각도를 따라가게 되어 약간 위쪽을 향하게 된다. 이렇게 되면 레이저 센서가 천장을 감지하여 원하지 않는 작동을 할수도 있는데 이를 방지하기 위한 최대의 거리가 2.3m인 것이다. 이 값이 만약 2.3m보다 크게 되면 Distance Sensor가 천장을 감지하게 된다.



Fig. 8 로봇 주행 시뮬레이션

3.3.5. 결 론

Distance sensor의 값을 받아들여서 장애물회피 방법을 연구하였는데 우선 액추에이터 역할을 하는 모터의 각속도를 제어하여 항상 로봇이 통로 중앙을 이동할 수 있도록 알고리즘을 구현하여 시뮬레이션을 하여 실제 로

봇이 통로 중앙을 유지하면서 이동하는 것을 확인할 수가 있었으면 또한 전방에 장애물이 발견 되었을 때 전방에 설치되어있는 Distance sensor가 거리를 감지하여 2.3m이내에 장애로 값을 읽어 들이게 되면 장애물을 피하여 이동하는 것을 확인 할 수가 있었다. 이때에 2.3m라는 값이 로봇의 Distance sensor가 천장을 감지하지 않는 최대의 거리이고 로봇자체의 관성력을 고려해 보았을 때 가장 적합한 거리로 산출되었다. 그리고 장애물을 회피한 이후에도 한쪽 벽에 치우치지 않고 계속적으로 통로의 중앙을 유지하면서 이동하는 것을 볼 때 모든 알고리즘이 동시에 작동되고 있다는 것 또한 확인 할 수가 있었다.

3.3.6. 후 기

앞에서 언급한 모델링과 시뮬레이션 알고리즘을 모두 한꺼번에 통합하여 시뮬레이션을 한 결과 설정한 이동경로에 따라 이동을 하는 만족스러운 결과를 얻을 수가 있었으며 이를 통해 실제 메커니즘 구성시 에도 적용을 할 수가 있었다. 또한 실제 물리적인 실험이 아니라 소프트웨어적인 실험이라는 면에서도 의의가 있고 앞으로도 실제 비싼 장비들을 구입하지 않고서도 실험을 할 수 있다는 가능성을 본 것으로 큰 보람을 느낀다. 또한 이것이 실제 목적을 다하여 실제 발생 될 수 있는 문제점을 사전에 파악하여 해결한 뒤에 실제 메커니즘에 적용을 할 수 있다면 경제적으로 시간적으로 아주 많은 효과를 볼 수 있을 것으로 기대된다. 하지만 무엇보다도 한 시스템에 대해서 전체적으로 공부를 할 수 있었고 세부적으로는 각 각의 부분에 대해서 도 알 수가 있었던 아주 좋은 기회였다. 이번 종합설계를 바탕으로 앞으로 사회에 나가서 로봇은 아니지만 어떠한 시스템에 대해서 구성을 할 때에 조금더 남들보다 자신감 있게 임할 수 있을 거라 생각을 한다. 그리고 1년 가까이 여러명의 동기들과 종합설계를 진행하면서 팀웍이 무엇인지에 대해서도 많이 알게 되었고 이 또한 앞으로 사회생활을 하는데 있어서 큰 도움이 되리라 믿는다. 또한 대학생활의 마지막을 정리하면서 아주 의미 있는 종합설계를 하게 해주신 교수님과 대학원생들 그리고 함께 참여한 동기들에게 감사하다는 말을 전하고 싶다.

3.4 영상처리 기법을 이용한 모바일 로봇의 이동 간의 명패인식 알고리즘 개발

3.4.1 서 론

영상처리 기법이란 다양한 방법으로 영상을 목적에 맞게 변경하여 사용하는 것이다. 예를 들면 무인 개폐 시스템, 주정차 단속 차량 시스템, 자동 불량 검출 시스템 등이 있다. 이뿐만 아니라 영화산업, 군사산업, 공장시설, 의료장비 등 많은 분야에서 사용 중이며 앞으로 적용 가능한 분야도 무수히 많다고 볼 수 있다.



그림 1. 현재 사용 중인 영상처리 기법을 이용한 다양한 장비들.

영상처리 기법의 과정을 간단히 설명하자면 마치 사람이 사물을 눈으로 보고 그것이 무엇인지 머리로 인지하는 것과 같다고 할 수 있다. 이를 영상처리 기법으로 표현 하면 카메라를 통해 영상을 입력 받고 입력된 영상을 저장할 장비와 영상을 판단할 소프트웨어가 있으면 되는 것이다. 이처럼 영상 입력 장비, 영상 저장 장비, 영상 판단 장비 만 갖추어 진다면 원하는 소프트웨어를 탑재하여 다양한 용도로 쓸 수 있기 때문에 영상처리 기법을 다양한 곳에 활용 할 수 있는 것이다.



(그림 2. 영상처리 기법의 과정)

이번 연구에서는 이러한 영상처리 기법을 사용해서 이동 중 주위환경의 영상을 이용한 간접적 위치추정을 해보고자 한다. 이를 활용하면 거리의 표지판이나 도로 표지판 또는 특정 건물 등을 이용해 위치를 알 수 있으며 이를 GPS 시스템과 더불어 사용한다면 GPS가 되지 않는 지역에서도 위치 파악이 가능하며 고가의 추가 장비 없이 실현 가능한 부분이라 생각된다. 실제 요즘 차량에는 차량용 블랙박스라 하여 사고 시 사고 장면을 기록하기 위한 카메라와 하드웨어를 탑재하는 차량이 늘고 있는데 이런 차량의 경우 별다른 추가 장비 없이 소프트웨어의 추가만으로 그 기능을 더할 수 있다는 장점이 있다.



차량용 블랙박스 장착 차량(카메라, 센서)

차량 내부 하드웨어

(그림 3. 차량용 블랙박스에 장착된 카메라와 하드웨어의 모습)

3.4.2 숫자인식 알고리즘 개발

(1) 시스템 구성

이번 연구는 학교의 특정 건물 내에서 이루어진다. 목적은 모바일 로봇에 캠을 장착하여 모바일 로봇의 이동 중에 캠을 통해 영상을 입력받고 그 영상을 영상처리기법을 이용하여 처리한 후 현재 로봇이 이동 중인 주위환경을 파악, 로봇의 위치를 간접적으로 알게 하는 알고리즘을 구현 하려고 한다. 이는 복도에 있는 수많은 방의 명패를 로봇이 지나가면서 영상처리 기법을 이용하여 분석하고 이를 통해 몇 호실을 지나고 있는지 파악하여 간접적 위치파악을 해보고자 하는 것이다. 이번 연구를 실행하기 위해서 먼저 몇 가지 제약적 조건을 정해야 했는데 첫째, 로봇에 장착된 캠의 시야각에 의해 로봇은 복도 중앙을 이동해야 하는 것이다. 이는 로봇의 높이와 캠의 시야각 때문에 복도 중앙을 이동하지 않으면 명패의 영상을 입력 받을 수 없기 때문이다. 둘째, 로봇의 주행 시 로봇은 평행을 유지하고 로봇의 속도는 1m/s 이하로 하여야 한다. 이는 로봇의 평행이 유지되지 않을 시나 속도가 1m/s이상이 될 시에는 캠의 성능 상 입력 영상이 흔들려 영상에 잔상이 남아 영상처리를 하더라도 그 영상을 사용하여 주위환경의 특징을 파악하기는 힘들기 때문이다. 따라서 이번 연구의 시스템 구성은 위의 조건을 갖춘 로봇의 복도 이동 중 1초에 한번 씩 영상처리 기법을 수행해주게 되는 것이다. 이때 영상 내에 명패가 없으면 인식이 불가능 한 것으로 하고 명패가 들어오면 그 명패의 숫자를 파악하여 몇 호실 앞인지 인식하게 한다.



그림 4. 영상처리 기법을 이용한 명패인식 과정 시스템 구성도

(2) 영상처리 기법을 이용한 숫자인식

(가) 영상 입력

모바일 로봇에 장착 된 캠을 통하여 복도 환경의 영상을 입력 받는다.



그림 5. 모바일 로봇의 주행 중 복도 영상 입력

(나) 흑백처리와 이진화

입력 된 영상에 영상처리를 좀 더 용이하게 해주는 흑백처리와 이진화 작업을 해 준다. 영상을 흑백으로 처리하는 이유는 영상처리 기법을 사용하여 숫자인식을 하기위해 굳이 컬러 영상을 쓸 필요가 없고 컬러 영상을 쓰게 되면 연산처리의 시간이 늘어나기 때문에 빠른 연산이 필요하므로 흑백처리를 해준다. 이진화는 영상의 모든 화소를 어떤 특정 기준값에 의하여 0 (검은색) 아니면 255(흰색)의 값으로 바꾸어 주는 작업을 말하는데 이는 흑백의 영상을 좀 더 영상처리하기 용이하게 바꾸어 주는 작업이다. 여기서 경계값을 정하는데 쓰인 알고리즘은 흑백 영상의 히스토그램 분포를 이용하여 특정 경계값으로 나누었을 때 그 분산이 최대가 되는 경계값을 자동

으로 설정 해 주는 알고리즘을 사용하였다. 이 알고리즘의 장점은 영상에 따라 달라질 수 있는 기준값을 자동으로 설정 해준다는 것 이지만 단점은 때로 히스토그램 분포가 적절치 못할 경우 기준값도 적절하지 않은 기준값 이 잡힌다는 것이다. 하지만 이번 연구의 조건에서는 사용하기에 무리가 없는 것으로 판단되어 사용하였다.

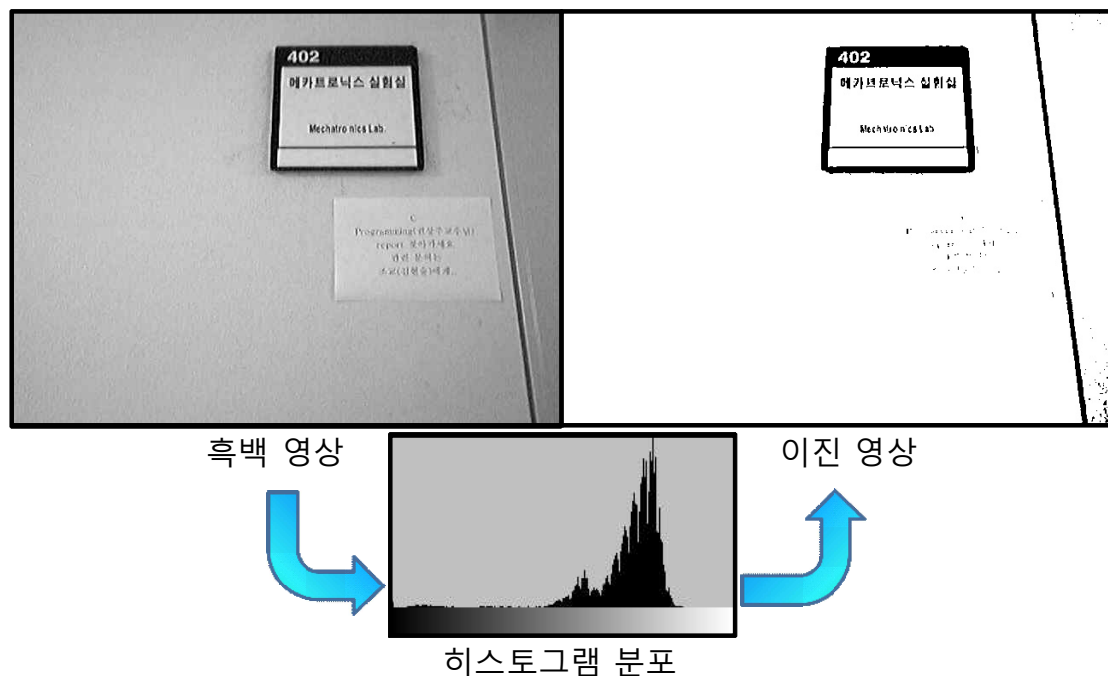


그림 6. 영상처리 기법을 이용한 입력 영상의 흑백, 이진화

(다) 라벨링 기법을 통한 노이즈 제거

위의 이진화한 영상을 보면 여기저기에 노이즈가 끼여 있는 것을 볼 수 있다. 영상처리 기법을 통하여 숫자 인식을 하기 위해서는 이를 제거해 주는 작업이 필요한데 여러 가지 방법이 있지만 이번 연구에서는 라벨링 기법을 통한 노이즈 제거법을 사용하도록 한다. 라벨링 기법은 영상의 화소들을 검사하여 원하는 화소값을 가지는 화소를 만나게 되면 그 화소부터 라벨을 붙이며 그 화소와 인접한 화소들로 이어나가며 계속 라벨을 붙여주는 작업이다. 쉽게 말하면 인접한 화소에는 동일 라벨이 붙게 되는 것으로 그 동일한 라벨이 붙은 화소들의 크기도 알 수 있게 되는데 이를 이용하여 노이즈를 제거 할 수 있는 것이다. 일정 크기 이상의 동일 라벨을 가지는 화소들을 제거하거나 특정 라벨값을 가지는 화소들을 제거하는 방법을 사용하는데 이번 연구에서는 20000픽셀 이하의 동일 라벨을 가지는 화소들 중

가장 큰 값을 가지는 화소를 찾고 나머지는 제거하는 방법으로 하였다. 이는 노이즈의 경우가 크게는 20000픽셀이 넘어가는 경우가 있었기 때문에 20000픽셀 이하의 값을 가지는 경우에서 관심영역을 선택한 것이다.

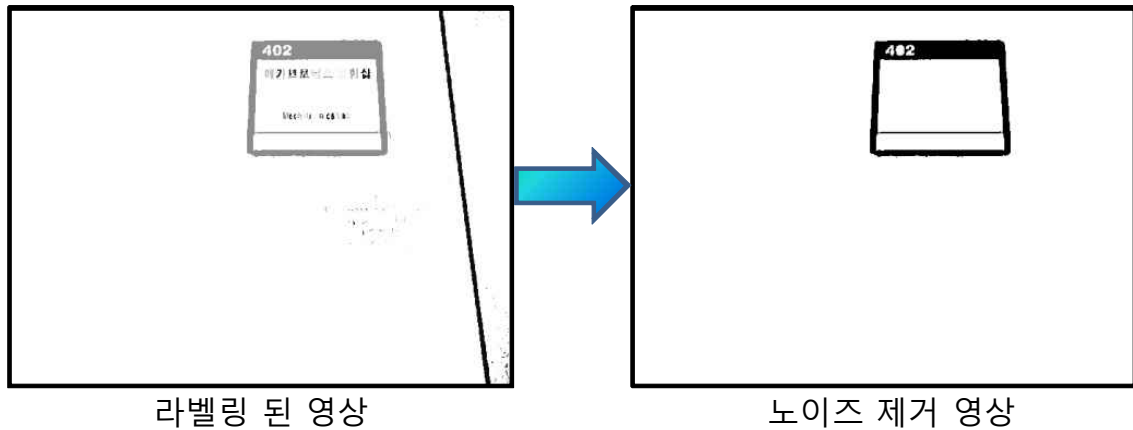


그림 7. 이진 영상의 라벨링과 노이즈 제거

(라) 관심 영역 추출

입력 된 영상에서 연구에 필요한 부분은 명패 부분이다. 따라서 명패 부분을 따로 떼어내기 위한 작업을 한다. 이는 나중에 명패에 있는 숫자 부분을 인식하기 위해 미리 선행하는 작업이기도 한데, 복도에 걸쳐 있는 명패의 숫자 부분의 위치가 일정 하다는 점에 착안 하여 이 방법을 사용하고자 한다. 이 방법은 관심 영역의 추출 시 관심 영역이 제대로 추출 되지 않는다면 숫자 영역 추출 까지 벗어나버리는 단점이 있지만 연산이 간단하기 때문에 수행 시간이 단축 된다는 장점이 있다.

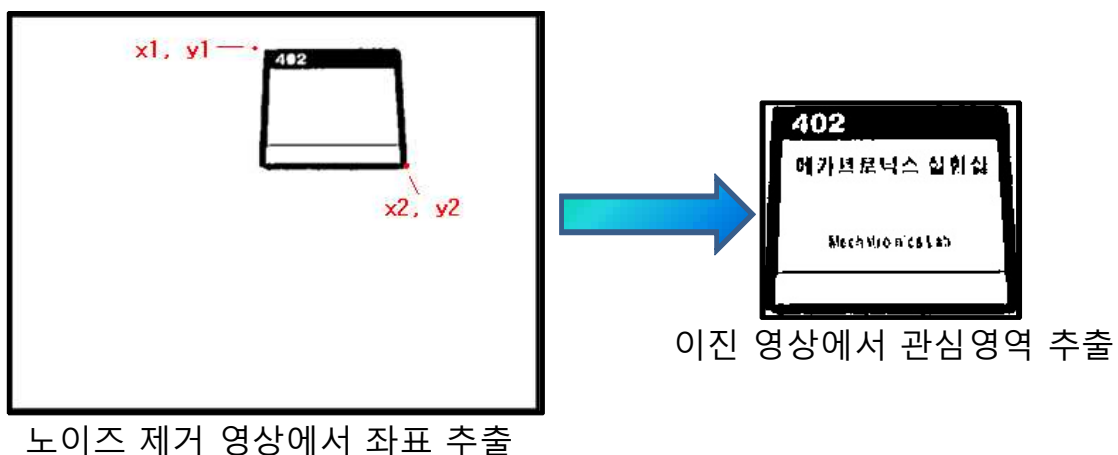


그림 8. 노이즈 제거 영상에서 관심 영역을 추출하기 위한 좌표 추출

관심 영역 추출은 노이즈 제거 된 영상에서 화소값을 검색하여 0의 값을 가지는 화소들의 좌표를 추출한 후 각각의 좌표에서 가장 작은 x_1 , y_1 값과 가장 큰 x_2 , y_2 값을 선택하여 두 점의 좌표를 추출한다. 그리고 이 추출 된 좌표값을 원래 이진 영상에 적용하여 관심 영역을 추출해 준다.

(마) 관심 영역 정규화

위에서 추출 된 관심 영역을 일정한 크기로 변환 해 주는 작업을 한다.

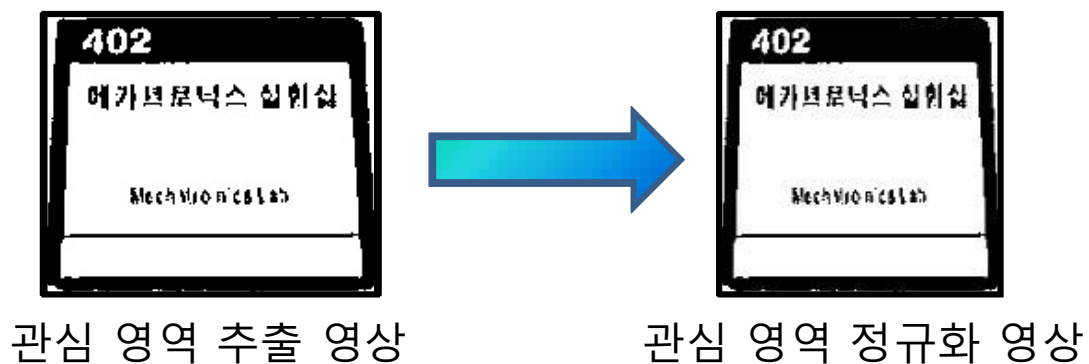


그림 9. 관심 영역 추출 영상의 정규화

이는 명패가 입력 된 영상의 조건이 매번 다를 수 있기 때문에 이후 숫자 영역 추출 작업을 용이하게 하기 위하여 해준다. 여기서는 가로x세로 150x150의 크기로 정규화 해주었다. 정규화 하는 작업에 있어서 기존에 추출된 관심 영역이 정규화 하는 사이즈보다 작을 경우 정규화 하게 되면서 영상이 확대 되어 영상에 흠 이 생기게 되는데 이는 주변 화소값의 평균값을 구해서 채워 주는 보정 작업을 거치게 하여 채워 준다.

(바) 왜곡 보정

위와 같이 정규화 된 관심영역의 영상을 가지고 숫자 영역을 추출해야 하는데 보는 것과 마찬가지로 영상이 기울어져 있다. 이는 로봇의 크기가 명패의 위치에 비해 작다 보니 캠에서 명패를 찍으면 아래에서 위를 올려다 보는 영상이 되기 때문이다. 따라서 영상은 아래가 길고 위가 짧은 사다리꼴 영상이 되며 숫자 영역을 제대로 추출하기 위해서는 이를 보정 해 주는 작업이 필요하다.

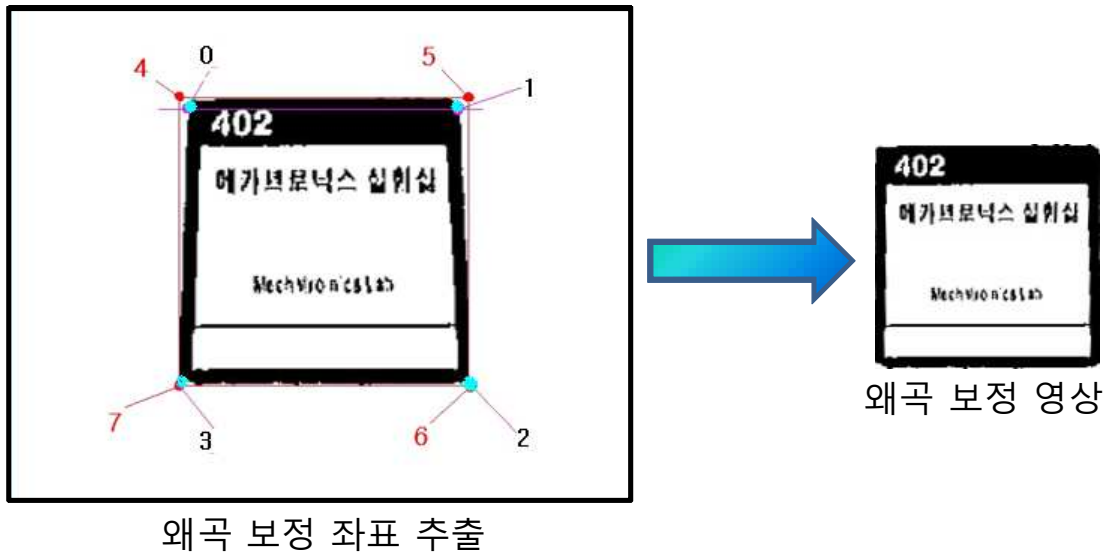
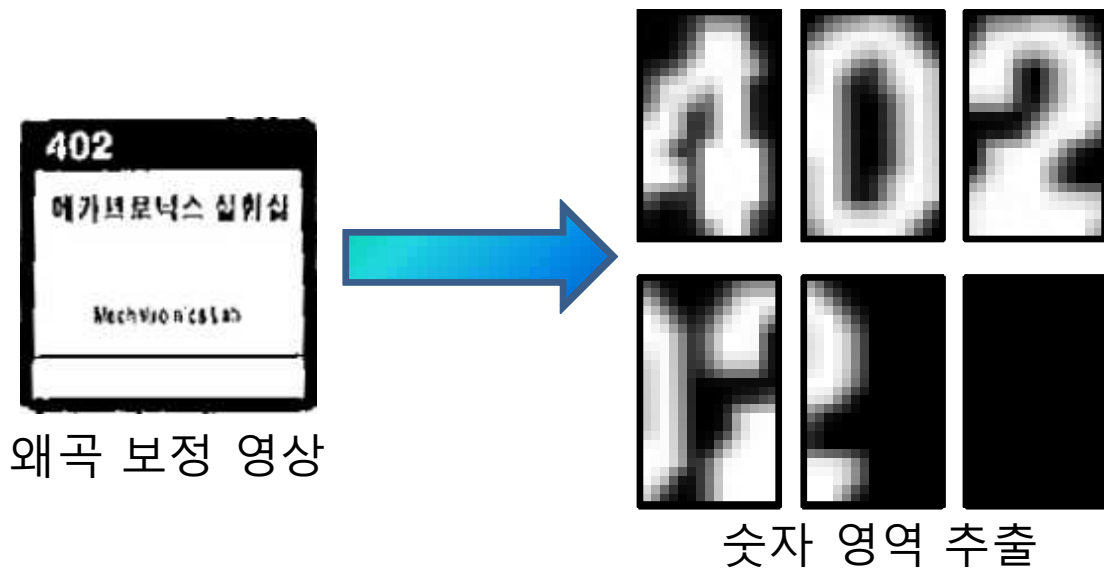


그림 10. 왜곡 보정을 위한 좌표 추출과 왜곡 보정

왜곡을 보정하는 작업으로 우선 정규화 된 영상에서 네 꼭짓점의 좌표를 추출한다. 여기서 좌표를 추출하는 방법은 위에서 관심 영역을 추출 한 방법과 비슷한데 2, 3 두 좌표는 어차피 정규화 된 영상이기 때문에 그냥 꼭짓점을 추출 하였고 0, 1 두 좌표는 명패에서 숫자영역이 시작되는 위치의 비율이 7%이기 때문에 정규화 영상의 7%까지 높이를 검사해서 추출하게 하였다. 이때 이 7%위치에서 좌표를 추출하는 이유는 관심 영역 추출 시 사용한 방법을 쓰면 관심 영역에 노이즈가 낄 경우 그 노이즈를 포함한 좌표값이 추출 되기 때문이다. 그리고 4, 5, 6, 7의 좌표를 구하여 왜곡 된 영상을 올바르게 펴주는 작업을 해준다. 이때도 영상의 확대가 있을 시 보정해주는 작업을 겸한다.

(사) 숫자 영역 추출

왜곡 보정 영상에서 숫자 영역을 추출한다. 명패마다 숫자 영역이 위치하는 자리가 같음을 이용하여 숫자가 있는 좌표를 설정하고 가로, 세로의 길이를 지정해 주어 숫자 영역을 추출한다. 여섯 영역을 추출하는 것은 명패의 종류가 두 종류 이고 두 종류의 숫자영역의 위치가 각각 다르기 때문에 다음과 같이 추출하였다. 이는 앞에서 말한 것처럼 노이즈에 의해 관심 영역의 추출이 올바르게 되지 않는다면 벗어나게 되는 경우가 발생 하지만 연산 수행의 속도가 빠르다는 장점이 있기 때문에 사용 할만하다.



(그림 11. 왜곡 보정 영상에서의 숫자 영역 추출)

(아) Template Matching 기법을 이용한 숫자 인식

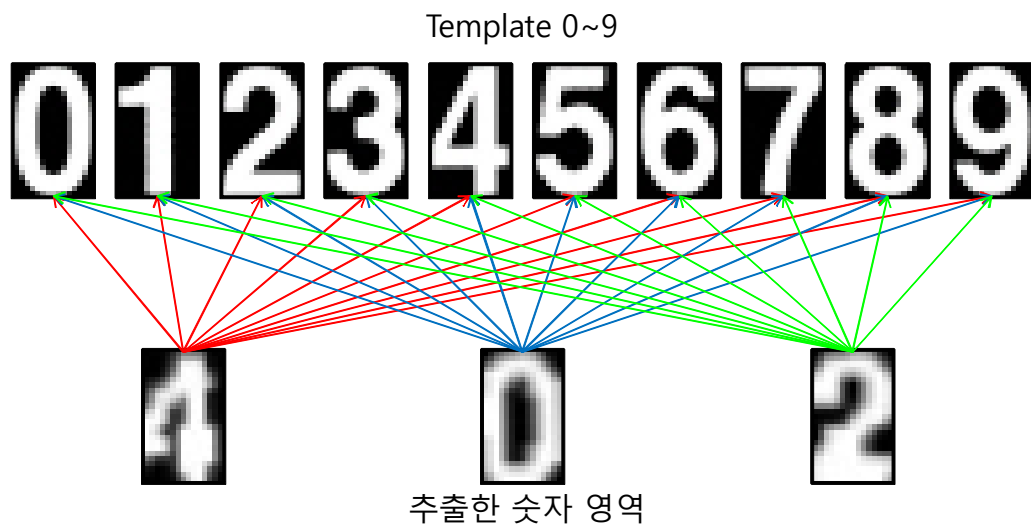


그림 12(a). 숫자 영역 1,2,3과 템플릿 사이의 Template Matching

Template Matching 이란 템플릿을 미리 준비하여 두고 이를 영상과 비교하여 매칭이 어느 정도 이루어 졌는지 파악하는 것이다. 이번 연구에서 숫자를 인식하게 하는데 있어서 이 기법을 사용하였다. 우선 0~9까지의 템플릿을 준비하고 위에서 추출한 숫자 영역을 각각의 템플릿과 매칭을 통해서 그 정도를 파악하여 인식 여부를 결정하게 한다.

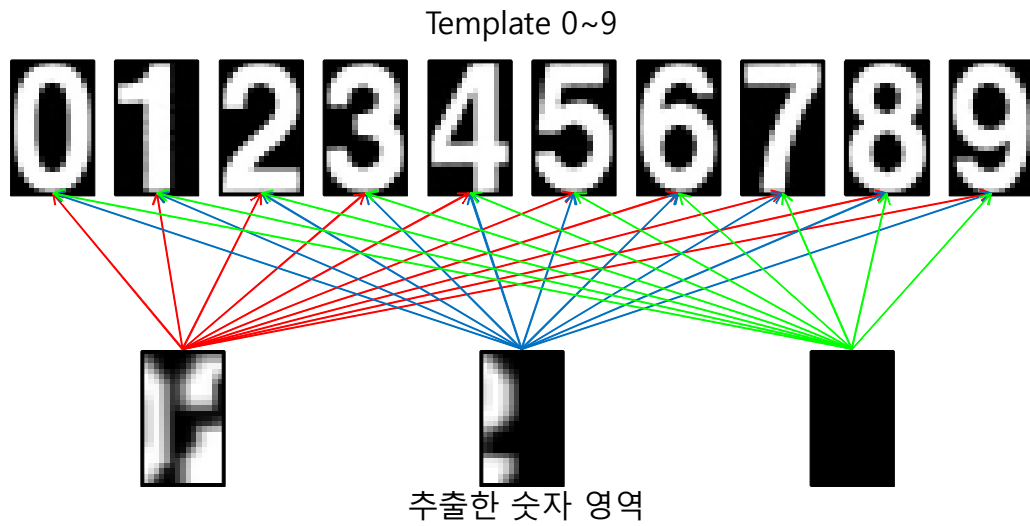


그림 12(b) 숫자 영역 4,5,6과 템플릿 사이의 Template Matching

위와 같이 각각의 템플릿과 추출 된 숫자 영역을 매칭 시킨 결과는 다음과 같다.

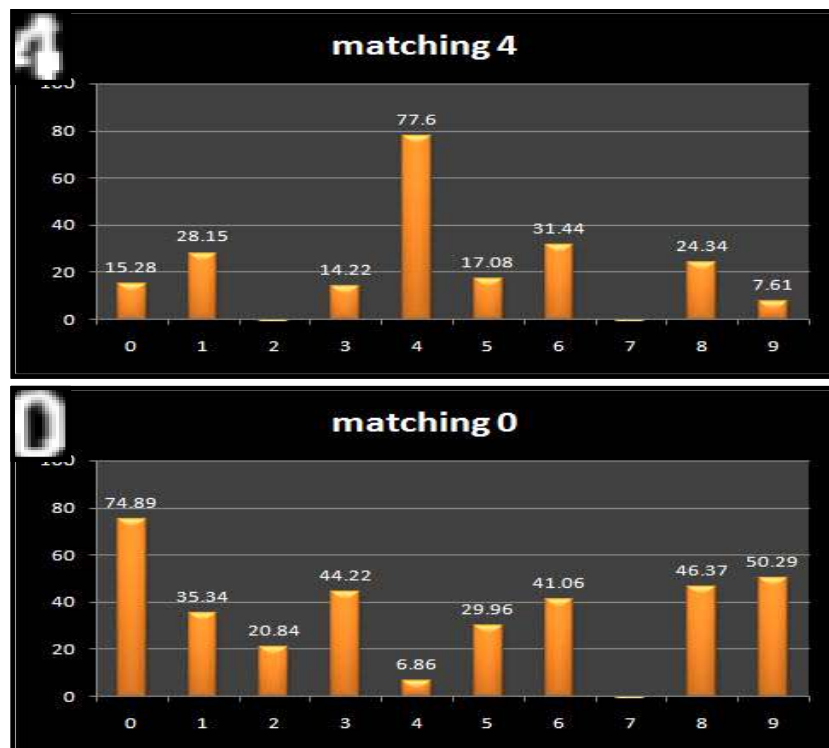


그림 13(a) 숫자 영역 1, 2의 매칭 결과



그림 13(b) 숫자 영역 3, 4의 매칭 결과

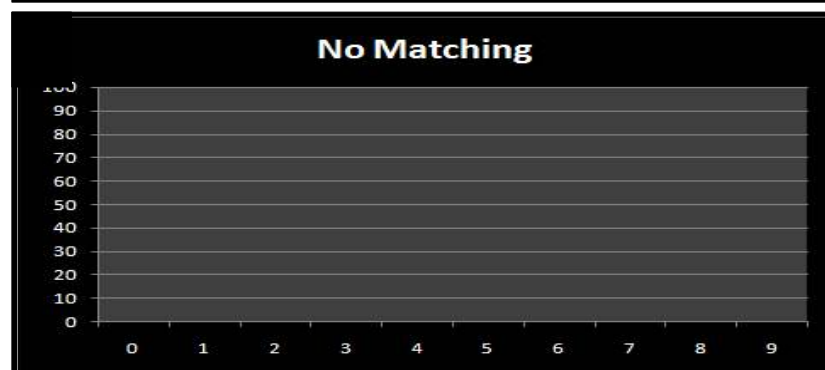
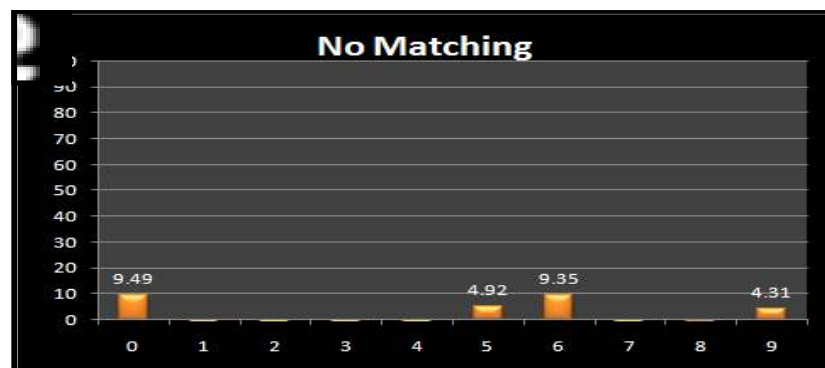


그림 13(c) 숫자 영역 5, 6의 매칭 결과

(자) 신뢰 구간의 설정

위와 같이 템플릿 매칭을 해본 결과 숫자 영역이 어떻게 추출이 되든 매칭값은 나왔다. 매칭값의 크기에 차이가 있을 뿐이기 때문에 어느 정도의 값 이하는 매칭이 안 된 것으로 판단 하고자 하는 신뢰 구간의 설정이 필요하다. 따라서 이번 연구에서의 신뢰 구간은 0.55, 즉 55%의 매칭률 이상을 가질 때에만 신뢰 할 수 있으므로 그 이상의 값을 가질 경우에만 매칭이 되었다고 판단하게 한다. 이는 어떤 공식에 의해서 정해지는 것이 아니라 여러 번의 실험에 의해 결정 되어진 것으로 본 보고서에 포함하지 못한 경우에는 매칭이 되었으나 그 값이 작은 경우도 있었고 다수의 수행 결과 55%정도면 매칭이 된 경우와 안 된 경우를 분간 할 수 있을 것이라 판단하여 설정 하였다.

(차) 명패 입력 영상이 다른 경우

다음은 명패의 영상이 앞에 영상 보다 좀 더 크게 들어온 경우 인데 알고리즘 수행 결과 다음과 같이 나오게 되었다. 여기서는 숫자 영역이 앞에서처럼 명확히 구분되어 추출되었기 때문에 매칭 결과도 비슷한 결과를 보였다.

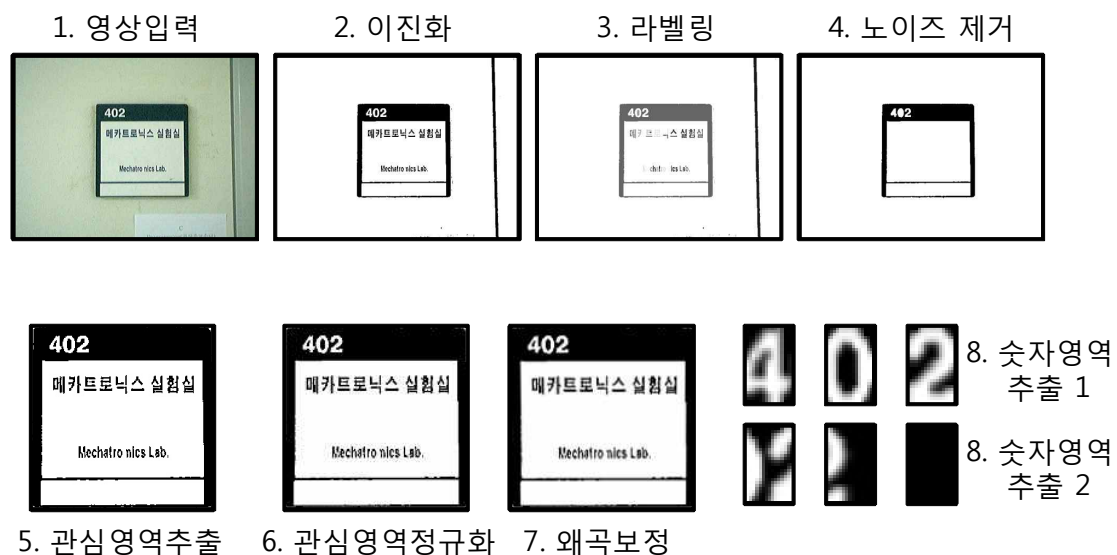


그림 14. 명패 크기가 다른 영상에서의 숫자 인식 과정 흐름도

3.4.3 결 론

숫자 영역 추출에 있어서 그 영역에서 크게 벗어나지 않는다면 신뢰구간에 부합하는 결과를 얻을 수 있었다. 숫자 영역이 잘 추출 된 경우는 99%

이상 신뢰 구간에 합당한 매칭 결과를 나타내었다. 하지만 그 영역이 벗어나는 경우가 있었는데 명패에 노이즈가 같이 붙어서 생기는 경우였다. 이번 연구에서 사용한 노이즈 제거 기법은 라벨링을 통한 제거 기법인데 작은 노이즈를 없애거나 특정 노이즈를 지정해서 없애기에는 강력하지만 이 라벨링의 단점이 인접한 화소들은 모두 하나로 간주 한다는 것이다. 따라서 명패의 주위에 노이즈가 같이 붙어서 발생하는 경우 이를 제거할 방법으로 는 부족한 부분이 많았다.

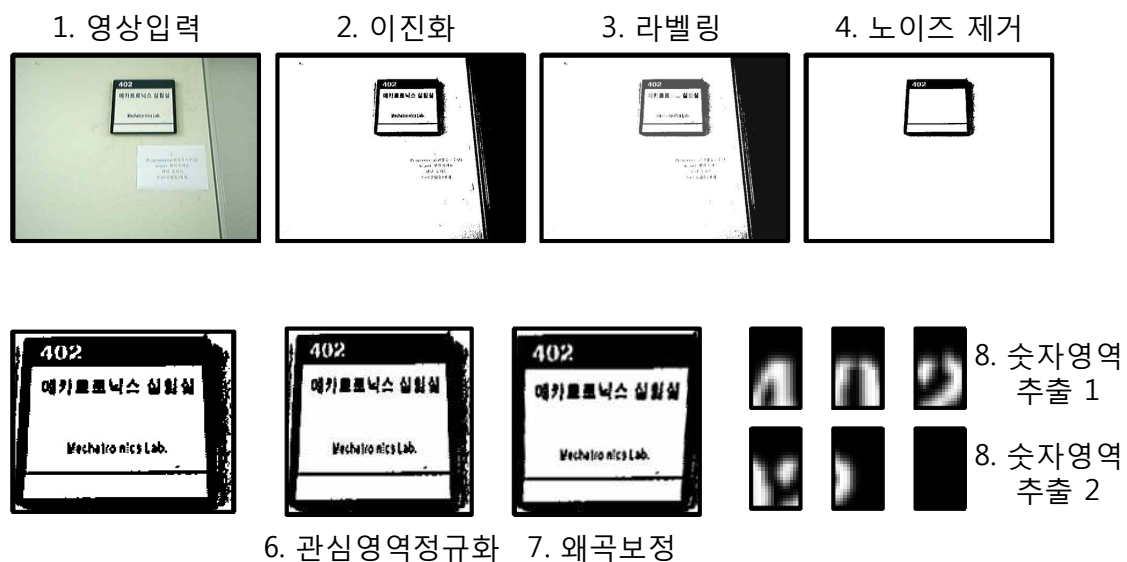


그림 15. 명패에 노이즈가 붙어서 생기는 경우

하지만 이와 같은 경우가 아니면 노이즈 제거는 쉽게 된 편이었고 원하는 데로 영역도 잘 지정되고 매칭도 좋은 결과를 볼 수 있었다. 노이즈 제거 부분에 있어서는 라벨링 기법에 추가적인 영상처리 기법을 동원하여 노이즈가 이와 같이 발생하는 경우에도 이 알고리즘을 사용할 수 있도록 하는 방안을 강구 하여야 할 듯하다.

3.4.4 후 기

이번 연구를 통해서 영상처리 기법에 대해 많은 것을 배웠다. 사실 기계공학을 전공 한 터라 C언어도 하나도 모르는 상태였고 개인 적인 사정으로 인해 2학기에 복학하게 되었기 때문에 시간적으로도 많이 부족 했다. 그래서 처음에는 너무 막막하였는데 C언어에 대해서 공부하고 조금씩 하다 보니 크진 않지만 어느 정도 노력의 결과가 나오기 시작했다. 이것만 가지고

는 연구의 결과라 하기에는 너무 부족한 점이 없지 않지만 사실 원가를 새로 만들거나 남들이 모르는 무엇인가를 발견 해내는 것이 목적이 아니라 지금 구현되어 있는 것에 대해 흥미를 가지고 배워 나간다는 것이 목표였던지라 그 목표에는 어느 정도 많이 도달 한 것 같다. 개인적으로 영상처리 기법을 공부 하면서 나중에 들어서는 흥미가 많이 생긴 것이 사실이다. 이번 알고리즘을 만들면서 느낀 것인데 사람의 머리는 정말 대단한 것 같다. 눈으로 보고 머리로 판단하는 것이 영상처리라 그냥 보면 굉장히 단순 할 것 같은데 숫자 하나를 판단하게 하는 것도 그렇게 많은 작업이 소요되니... 하지만 이런 점이 더 흥미를 끄는 것 같다. 언젠가 진짜 영상처리 기법이 발달하게 된다면 거리에 장착된 카메라로 사람의 얼굴을 찍어 누군지 판단하여 범죄자를 잡는 데에 용이 할 것이고 도로의 과속방지 카메라에 프로그램을 추가하여 차량 번호, 차종, 운전자가 누구인지 까지 판단 할 수 있을 것이다. 또 무인 정찰기 등에 탑재 한다면 목적을 이루기 위해 알아서 정찰을 하고 판단하는 것이 가능할 것이다. 이처럼 카메라와 컴퓨터 하나로 많은 것을 할 수 있게 하는 영상처리에 대해 조금이나마 배울 수 있어서 좋았고 앞으로 기회가 된다면 좀 더 배워보고 싶다.

3.5. 무선 영상 전송 프로그램 개발

3.5.1 서 론

최근에는 로봇의 발달이 가속화 되어가고 있다. 그리고 로봇은 원격제어를 통해서 원거리 영상 및 조작을 할 수 있는 단계까지 왔다. 로봇의 눈과 같은 기능을 담당하는 영상전송 시스템을 통해서 영상을 전송하고 제어장치를 통하여 로봇을 조작할 수 있다.

여기서는 로봇의 영상전송을 통해서 제어를 가능토록 하는 로봇을 만들어 보기로 했다. 현재 로봇은 거리의 제약과 전송속도의 단점이 있기 때문에 많은 보완을 했다. 원격 영상 로봇은 사람이 직접 작업하고자 하는 장소에 가지 않고 원격으로 로봇을 조정하고 로봇에 달린 카메라를 통해서 여러 가지 작업을 수행 할 수 있게 한 것이다.

학교의 무선 인터넷망을 통해서 학교안의 어디서나 무선으로 영상통신을 할 수 있다는 점을 착안하여 TCP/IP를 이용하여 시스템을 구축하였다. 그렇기 때문에 인터넷을 이용할 수 있는 곳에서는 영상을 전송하고 수신할

수 있게 되었다. 우리가 개발한 로봇을 모바일 로봇으로 로봇에 웹캠을 설치하였다. 모바일 로봇에 장착된 서버 컴퓨터와 웹캠을 서로 연결시키고 동영상 일단을 서버 컴퓨터로 받아들이게 하였다. 이후 받아들이는 영상을 클라이언트 컴퓨터로 전송하여 사용자가 볼 수 있도록 된다. 영상전송을 위해서는 TCP/IP에 대한 이해가 필요하고, 프로그램을 개발하기 위하여 MFC함수를 이용했다. 통신을 위해서 소켓프로그래밍과 영상 인식 및 전송을 위해서 VFW(Video For Windows)프로그래밍이 필요하다.

3.5.2 영상 전송 시스템 구성

(1) 기본 시스템

실험에서는 1개의 웹캠과 2개의 PC를 통해서 모바일 로봇에 웹캠과 PC 1개를 장착하고 남은 PC는 우리가 볼 수 있도록 설치한다. 웹캠이 설치된 모바일 로봇에서는 영상을 얻어서 PC에 저장된다. 그리고 무선영상 전송을 통해서 멀리 떨어진 PC에 영상을 전송하는 기반을 마련한다. 무선영상 전송으로 받아진 영상은 클라이언트에서 화면에 볼 수 있도록 설정되어 직접 사용자가 영상을 확인 할 수 있도록 시스템을 구성한다.

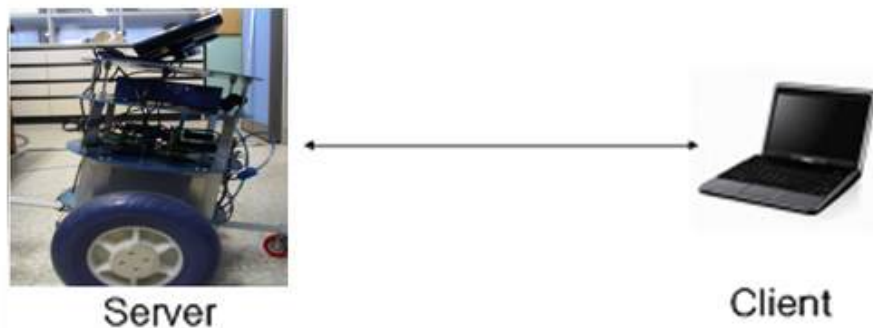


Fig.1 기본 시스템 구조

Fig.1과 같이 영상전송을 통해서 로봇이 지나가는 경로와 위치를 시각적으로 파악할 수 있고 장애물 인식 및 회피도 할 수 있을 것이다.

(2) TCP, UDP/IP를 통한 영상전송

영상전송 시스템을 무선으로 송수신 하기위한 방법에는 여러 가지가 있으나 여기서는 TCP 또는 UDP를 통한 전송방법을 채택했다. 통신 프로토콜을 가리키는 가장 일반적인 말로 TCP(Transmission Control Protocol)와

IP(Internet Protocol)가 있다. 요즘 인터넷에서 많이 사용하는 프로토콜로서 서로 다른 PC 사이의 통신을 위한 전송규약을 말한다.

통신을 하기 위해서 서버부분과 클라이언트부분을 설정하고 서로 데이터를 주고받을 수 있도록 구성한다.

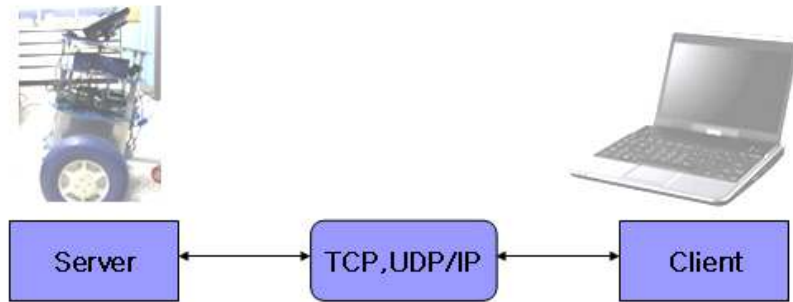


Fig.2 TCP, UDP/IP 서버와 클라이언트

TCP와 UDP는 서로 다른 개념을 지니고 있다.

먼저 TCP 통신은 전화와 같은 방식으로 동작하는 것과 비유 할 수 있다. 전화의 경우 상대방의 전화번호를 알고 있어야 전화를 거는 것과 같이 TCP 통신을 하기 위해서는 상대방의 IP 주소와 Port를 알고 있어야 연결을 요청할 수 있다. 그리고 전화에서 상대방이 받을 때까지 기다리는 것처럼 대기상태가 있다. 그래서 서버에서 응답을 하지 않으면 계속해서 요청을 보내는 것이다. 그리고 결국 요청을 받아들이지 않는 경우는 멈춰 버린다. 그리고 전화가 연결이 되었다면 양방향으로 대화하는 것처럼 TCP도 계속해서 서버와 클라이언트 간에 데이터를 주고받는다.

만약 전화가 잘 들리지 않을 경우 우리는 다시 말해 달라고 하는 것처럼 TCP 프로토콜도 데이터를 받지 못했거나 오류가 생겼다면 다시 데이터를 보내달라고 요청하는 메커니즘을 가지고 있다.

이와 달리 UDP 통신은 편지에 비유할 수 있다. 편지를 쓸 때 편지지에 데이터를 기록하고, 편지봉투에 상대방의 주소와 자신의 주소를 표시한 후 그냥 보내면 편지는 배달이 된다. 그리고 편지를 보낸 사람은 편지가 도착했는지 여부는 알 수 없게 된다. 이러한 원리는 UDP와 비슷함을 알 수 있다.

그래서 데이터를 생성한 후 상대방의 주소와 자신의 주소를 기록하고 보내면 받은 쪽에서는 데이터를 받았는지 안 받았는지를 확인할 방법은 없다. 그냥 데이터를 보냈다는 사실만 존재할 뿐이다. UDP 통신 자체는 상대방과 연결되어 있는 TCP와 같은 개념이 아니기 때문에 편지를 우체통에 넣

어버리는 개념과 비슷한 것이다.

TCP와 UDP는 이런 개념의 차이가 있기 때문에 각각의 장단점을 지니고 있다. 첫째로 TCP는 신뢰성이 있고 UDP는 신뢰성이 없다. TCP는 지속적인 연결 상태에 있기 때문에 데이터의 오류가 발생했을 경우 다시 재전송해준다. 그러나 UDP의 경우는 다르게 한쪽에서 일방적으로 데이터를 보내기만 하기 때문에 오류가 나거나 문제가 발생하여도 재전송을 하지 않는다. 그래서 받은 데이터의 신뢰성이 매우 떨어진다고 할 수 있다.

둘째로 UDP가 TCP보다 고속의 데이터를 전송 할 수 있다. UDP는 데이터의 신뢰성은 떨어진 대신에 계속해서 일방적으로 데이터를 전송하기 때문에 속도는 매우 빠르다. 그래서 실제는 동영상을 실시간으로 보내는 경우는 아주 적합하다고 할 수 있다. 이와 달리 TCP는 속도가 떨어지지만 데이터의 신뢰성이 보장되기 때문에 정확한 데이터를 전송 할 수 있다.

여러 가지 내용을 비교, 종합하고 실험을 통해서 우리는 TCP를 사용하기로 결정했다. TCP의 서버(Server)와 클라이언트(Client)의 프로그램을 구축하기 위해서 Visual C++를 이용했다. MFC함수를 사용하였다.

3.5.3 영상획득 및 압축

(1) 영상획득 방법

영상을 획득하는 방법에는 여러 가지가 있다. 먼저 웹캠으로부터 들어오는 영상을 어떤 식으로 받아들일지를 생각해야 한다. 그리고 받아들인 영상을 어떻게 보일 것이지도 생각해 봐야 한다. Visual C++을 통한 영상획득을 위해서는 VFW(Video For Window)나 등의 라이브러리를 사용하는 경우가 대부분이다. 그래서 본 논문에서는 VFW라이브러리 함수를 이용한 획득에 대해서 알아본다.

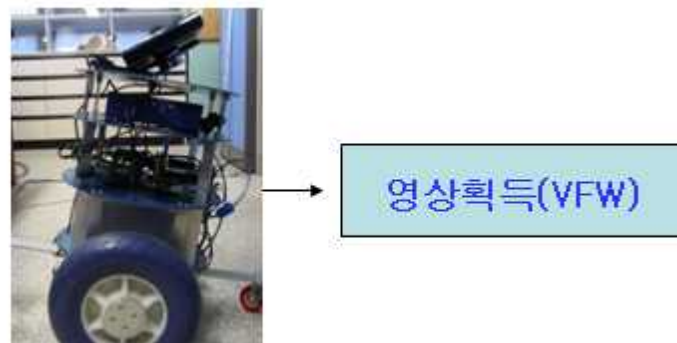


Fig3. 영상획득을 위한 구성

VFW에서는 웹캠을 통해서 영상을 지속적으로 캡처한다. 웹캠을 통해 캡처된 영상을 화면에 지속적으로 보여준다. 사람은 보통 1초당 15장면을 연속적으로 보여주면 동영상처럼 느낄 수 있기 때문에 캡처된 영상을 화면에 보여줌으로써 쉽게 동영상을 획득할 수 있다.

그러면 MFC함수를 통한 전송의 순서를 살펴보면 먼저 작성 소스 상단에 Vfw.h를 include 해주어야 한다. include를 해줬기 때문에 Vfw.h라이브러리 함수를 사용해서 영상을 받아들일 수 있는 준비가 된다.

간단한 작업순서를 보면 먼저 캡처된 화면을 보여줄 윈도우 영역 지정해야 한다. 윈도우 영역을 지정했다면 다음으로 비디오캡처 윈도우를 지정된 영역에 생성한다. 그래서 비디오를 캡처하기 위한 준비 및 캡처 후 영상을 저장할 변수를 지정한다.

영상을 저장할 준비가 되었기 때문에 이제는 이전의 캡처윈도우와 웹캠을 연결한다. 장치 드라이브와 연결함으로써 이제 장치에서 영상을 받아들일 준비를 한다.

캡처윈도우에 프리뷰 모드를 통해서 영상을 받아들이고 자신의 PC화면에 영상을 보여준다. 여기까지 설정함으로써 호스트 화면에 카메라를 통해서 영상을 디스플레이 한다.

이렇게 받아들인 영상을 다른 PC로 전송하기 위해서는 받아들인 각 이미지를 캡처 한 후 셋팅을 해야 한다. 그러기 위해서는 콜백(callback)함수라는 것을 볼 수 있는데 콜백함수를 간략히 설명하면 데이터를 세팅하기 위해서 윈도우에서 콜백함수를 부르면 작업 후 다시 되돌아가는 함수다. 그래서 영상의 경우 지속적인 콜백함수를 사용하게 된다.

이제 콜백함수를 알았으면 콜백함수의 호환을 위해서 비디오 포맷을 한다. 그리고 캡처되는 이미지의 정보를 얻어내고 콜백함수를 지정한다. 그리고 모든 작업이 끝났다면 캡처에 필요한 정보를 세팅 후 저장한 후 캡처링을 실시하면 된다. 여기까지 설정함으로써 콜백(callback) 함수를 통한 실시간 이미지 전송기반이 마련되게 된다.

Table.1을 보면 알 수 있듯이 각 함수가 하는 부분이 정해져 있고 그 부분에 맞게 작업을 처리하면 손쉽게 영상을 획득할 수 있다. 영상을 호스트 PC에 보여주고 동시에 무선으로 영상을 전송하기 위한 기반이 마련되어 있다.

캡처함수	설명
capCreateCaptureWindow	비디오캡처 윈도우를 지정된 영역에 생성
capDriverConnect	캡처 윈도우와 드라이버 연결
capPreview capPreviewRate capPreviewScale	캡처윈도우에 프리뷰 모드로 보여줌
capSetVideoFormat	콜백 함수의 호환을 위해서 비디오 포맷 설정
capGetVideoFormat	캡처되는 이미지의 정보를 얻어냄
capSetCallbackOnVideoStream	콜백 함수를 지정
capCaptureSetSetup	캡처에 필요한 정보를세팅 후 저장
capCaptureSequenceNoFile	캡처링을 실시

Table.1 캡처를 위한 각 함수

(2) 영상압축

영상데이터는 많은 양의 정보를 필요로 하므로 전송할 때는 높은 지연이 생기고 과부하가 생기게 마련이다. 이런 상태에서는 영상데이터를 전송할 때에는 압축을 해줄 필요가 있다.

해상도	크기		
	BMP	TIF	JPEG
640×480	901kB	143kB	53kB
320×240	226kB	50kB	19kB

Table.2 사이즈별 압축

모바일 로봇에 경우 24비트의 컬러를 사용하였고 320×240의 영상을 사용하였다. 그런데 이 경우 226kB가 나와서 꽤 크다는 것을 알 수 있다. 이런 경우는 실시간을 통해서 들어오는 데이터가 크기 때문에 영상데이터 전송의 속도에 영향을 끼친다. 그래서 우리가 실제 원하는 로봇의 상황 및 환경을 뒤늦게 파악하게 될 것이다. TIF의 압축방식도 있지만 JPEG 경우는 월등히 높은 압축률을 가지고 있다. 전체적으로 압축시간 및 압축정도가 우수하기 때문에 이번 모바일 로봇 연구에서는 JPEG압축을 이용하여 영상을 전송할 수 있도록 구성했다.

3.5.4 TCP/IP를 통한 영상전송

(1) 소켓프로그래밍 원리

소켓 인터페이스는 TCP/IP 네트워크에 대한 API(Application Programming Interface)로, 네트워크 애플리케이션이 쉽게 작성될 수 있도록 해주는 운영체제 차원에서 제공되는 인터페이스이다. 여기서 윈도우즈는 코드가 직접 하드웨어를 제어하는 것을 허용하지 않는다. 항상 운영체제에 해당 작업을 요청하도록 되어 있다. 그래서 윈도우즈가 네트워킹을 하고자 하는 프로그래머에게는 소켓 API를 제공하는 것이다.

소켓이란 운영체제에서 제공되는 API들 중에서 특별히 네트워킹을 위한 API들에서 사용하는 것으로, 그 많고 많은 핸들 중의 하나이다. 때문에 누군가가 네트워크 통신을 하고자 한다면 맨 처음 소켓 객체를 만들고 그 핸들을 얻어야 한다. 통신이 끝나면 그 핸들을 닫아 주어야 하는 것이다. 그러니까 소켓은 파일 디스크립터(descriptor)와 유사하다고 볼 수 있다.

우리가 일상생활에서 편지를 주고받는 것에 비유를 해보자. 편지를 보내기 위해서는 편지를 쓰고, 우표를 붙인 다음 우체통에 넣는다. 그러면 그 편지가 도착하기까지의 과정은 우체국에서 다 해주어야만 한다. 편지를 보내는 사람은 단지 봉투에 정확한 주소와 이름을 적어서 우체통에 넣어주면 되는 것이다. 이 과정에서 소켓이란 우체통에 해당하며, 편지는 소켓을 통해 전송하고자 하는 데이터에 해당한다고 볼 수 있다. 우체국은 운영체제와 인터넷서비스망에 비유할 수 있다.

소켓은 두 가지의 대표적인 형태를 갖는데, 바로 데이터 그램과 스트림이다. 이것은 소켓객체가 처음 생성 될 때 지정되는 것으로, 한번 지정되면 그 객체가 소멸할 때까지 바꿀 수 없다. 앞에서 설명한 바가 있지만 데이터 그램 소켓은 UDP 데이터 그램을 사용해 데이터를 전송한다. 이 데이터 전송방식은 데이터의 정확성을 모든 경우에 대해서 보장하지 못한다. 데이터 그램은 전송하는 데이터가 분실될 수도 있으며, 데이터의 순서가 뒤바뀔 수도 있다.

반면 스트림 소켓은 데이터 전송을 위해 TCP 연결을 사용한다. TCP는 기록된 데이터에 에러가 없도록 하고, 전송 순서에 맞추어 정확히 전송되는 것을 보장한다. 만약 에러발생시 TCP는 문제가 발생한 것을 인식하고 데이터의 재전송이나 재 정렬을 통해 적절히 보상해 준다. 우리가 사용할 MFC 소켓클래스는 두 형식의 소켓 중 스트림 소켓을 즉 TCP연결을 사용했다.

(2) TCP/IP의 구조

TCP/IP의 기본적인 개념은 앞에서 설명을 하였다. 이제는 실제로 어떻게 통신이 가능하고 데이터를 주고받을 수 있는지 설명한다.

서버와 클라이언트의 소켓을 생성하고 각각 어떻게 데이터를 주고받는 구조를 나타냈다.

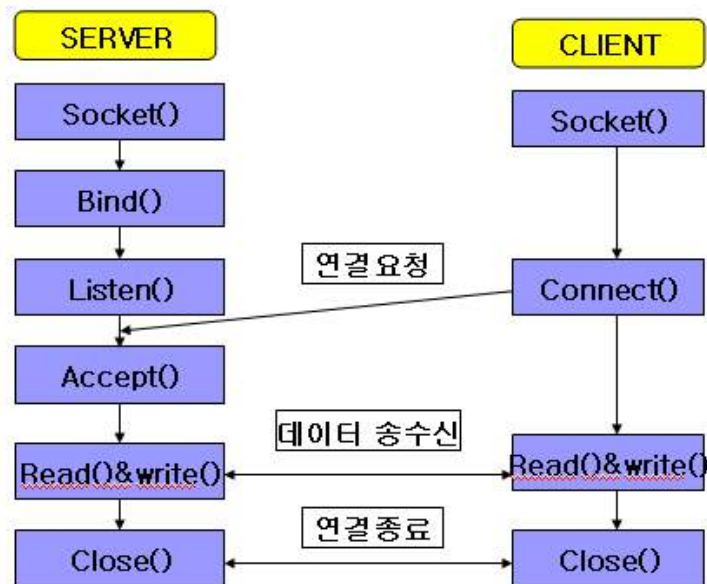


Fig.4 TCP/IP의 함수 구조

Fig.4에서 보듯이 TCP/IP는 서버부분과 클라이언트부분으로 나눌 수 있다. 서버와 클라이언트를 나누고 쉽게 설명하면 클라이언트에서 서버로 접속하는 것이다. 또 여러 개의 클라이언트가 하나의 서버에 접속하는 경우도 있지만 우리는 1대1 통신을 하도록 구성했다. 먼저 서버와 클라이언트는 처음에 각각 소켓을 생성해야 한다. 소켓을 생성해야 소켓통신이 가능하기 때문에 각각의 소켓을 생성한다. 소켓을 생성 시 우리는 TCP통신을 사용하고 데이터형태는 스트림방식을 이용했기 때문에 이 부분을 정확하게 지정한다. 그러면 이제 서버부분을 살펴보면 서버에서는 소켓을 생성한 뒤 `Bind`함수를 통해서 주소를 생성한다. 우리가 전화를 보낼 때 상대방의 전화번호가 필요하듯이 주소를 할당하여 클라이언트가 접속할 수 있도록 한다. 그리고 `Listen`함수는 연결대기 상태이다. 이제 주소를 할당했으니 이곳으로 접속할 수 있도록 대기를 하는 것이다. 그리고 클라이언트에서 접속이 왔을 경우 `Accept`함수를 통해서 연결을 받아들인다. 연결을 받아들인 후에는 연결이 지속되어 있기 때문에 `Read`함수, `write`함수 또는 `Recv`함수,

send함수를 사용하여 데이터를 주고받을 수 있다. 연결이 지속된 상태로 데이터를 주고받다가 모든 데이터를 주고받은 후에는 Close함수를 통해서 연결을 종료 시킬 수 있다.

(3) CAsyncSocket클래스를 통한 서버와 클라이언트 구축

우리는 MFC를 사용했기 때문에 MFC에 대한 소켓 클래스가 있다. 이 소켓 클래스를 살펴보면 CAsyncSocket 과 CSocket 클래스 인데, CSocket 은 CAsyncSocket으로 부터 상속된 클래스로, 개발자가 좀 더 편리하게 사용할 수 있도록 해 놓은 것이다.

우리가 사용한 것은 CAsyncSocket 객체는 비동기 소켓객체라고 하는데 즉 소켓에서 발생하는 모든 이벤트를 알아내기 위해 프로그래머가 전부 코딩으로 처리하는 것이 아니라 윈속이 제공하는 인터페이스를 사용하는 소켓 객체이다. 그리고 소켓의 이벤트는 객체에서 제공하는 이벤트 핸들러를 오버라이드해서 적절히 사용할 수 있다.

이제 CAsyncSocket클래스를 통해서 생성한 Create함수, Listen함수, Connect함수, Accept함수를 알아 볼 수 있다.

먼저 Create함수는 소켓을 사용하기 위해서는 맨 처음 소켓을 초기화해야 한다. 이 과정은 단순히 함수하나를 호출하면 끝난다.

```
AfxSocketInit(NULL);
```

이 함수를 애플리케이션 객체(CWinApp)의 InitInstance함수에서 호출하면 된다. 어디서 호출하든, 소켓을 사용하기 전에 소켓이 초기화 되어야 한다. 애플리케이션에서 소켓을 초기화 했으면, 이제 소켓을 사용할 수 있다. 소켓을 사용하기 위해서는 소켓 객체의 인스턴스를 생성하고 이 인스턴스의 멤버인 Create함수를 호출한다.

```
CAsyncSocket mySoc;  
mySoc.Create(UINT nPort);
```

지금 위에 있는 것은 서버측에서 클라이언트의 접속요청을 기다릴 소켓을 생성하는 것이다.

Create함수는 윈도우즈 소켓을 생성하고, 이를 호출한 인스턴스에 생성된 소켓을 붙여준다.

이 함수는 원래 4개의 인자를 받는데, 모두 다 디폴트 값이 지정되어 있다. 원형은 다음과 같은 형태이다.

```
CAsyncSocket::Create(UINT nSocketPort, int nSocketType, long lEvent,
```

LPCTSTR lpszSocketAddress)

첫 번째 인자 nSocketPort는 소켓이 사용할 포트이다. 디폴트는 0인데, 이 값에 0을 전달하면, 포트의 번호를 소켓이 자동으로 할당해서 사용하게 한다. 두 번째 인자 nSocketType은 소켓의 형태를 지정하는 상수로, 디폴트는 SOCK_STREAM인데, 이것은 생성되는 소켓을 스트림 소켓으로 생성하도록 한다. 만일 이 값으로 SOCK_DGRAM을 전달하면, UDP 프로토콜을 사용하는 데이터그램(Datagram) 소켓을 생성한다.

세 번째 인자 lEvent는 비트마스크로, 소켓에서 처리하고자 하는 메시지를 소켓에게 알려준다. 디폴트값은 소켓에서 발생하는 모든 메시지를 처리하는 것인데, 보통 이대로 사용한다.

네번째 인자 nSocketAddress는 접속되는 소켓의 IP어드레스를 가진 문자열상수의 포인터인데 디폴트는 널(NULL)이며, 보통 이대로 사용한다. 즉, 임의의 클라이언트로부터 접속요청을 받는다.

Listen() 함수는 Create를 호출한 후 호출해야 할 함수다. 이 함수에는 정수형(int) 인자를 하나 전달하는데 접속요청을 받아들일 수 있는 최대 접속요청 수이다. 최대값은 5개까지의 접속요청을 받는데 디폴트값 역시 5로 되어 있다.

```
mySoc.Listen();
```

위와 같이 호출을 하면 서버의 소켓인 mySoc은 접속 대기 상태가 되고, 클라이언트의 접속을 받아들일 준비가 되었다.

Connect 함수 서버에서 Listen()을 호출하고 소켓이 대기 상태가 되면, 클라이언트에서는 Connect()함수를 호출하여 접속을 요청할 수 있다. 이 함수는 두개의 인자를 받는다. 함수에게 전달하는 첫 번째 인자는 서버의 애플리케이션이 구동되고 있는 컴퓨터의 IP주소를 가진 문자열의 포인터이다. 두 번째 인자는 유닛(UINT)형의 포트번호이다. 그리고 이것은 당연히 서버 측 소켓에 전달한 것과 같은 포트번호이어야 한다.

Accept함수는 클라이언트 측에서 접속을 요청하면 서버에서는 접속을 받아들이기 위해서 Accept함수를 호출한다.

```
CSocket acceptSoc;
```

```
mySoc.Accept(acceptSoc);
```

위의 함수를 통해서 접속을 받아들이게 된다.

3.5.5 결과

영상획득과 영상전송을 통해서 서버와 클라이언트에서 각각 영상을 확인할 수 있었다. 각각의 영상을 통해서 TCP/IP를 통해서도 영상을 전송하는데 성공을 하였다. 그리고 각각의 영상에 대해서 몇 가지 실험을 하였다.

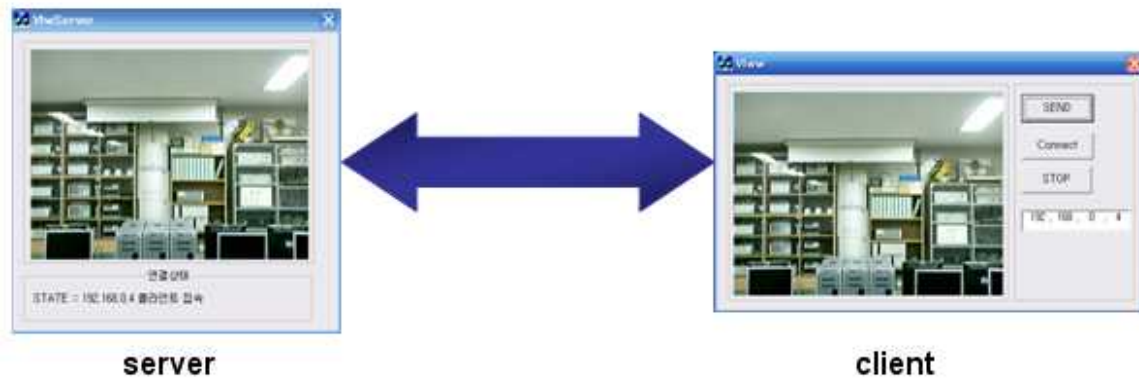


Fig 5. 영상전송 화면

각각의 실험은 영상전송을 할 때 프레임의 차이에 따라서 어떤 차이가 생길 것인지 알아보는 실험이었다. 그래서 실험을 통해서 최적의 영상전송속도를 찾아보려고 했다.. 실험에서는 일단 15프레임을 통해서 영상을 전송해보았다. 그런데 15프레임의 수는 매우 크기 때문에 생각보다 전송률이 낮았다. 그래서 서버와 클라이언트간의 차이 매우 심하게 나서 서버의 한참 지나간 영상이 클라이언트에서 나타나는 현상이 생겼다. 그래서 프레임을 낮춰서 영상을 전송할 수 있도록 해보았다. 일단 프레임을 10프레임 7프레임 4프레임으로 3가지 경우에 대해서 실험해 보았다. 10프레임의 경우 역시 프레임수가 많기 때문인지 15프레임처럼 클라이언트에서 늦게 화면에 나타나는 현상이 생겼다. 그런데 7프레임 그리고 4프레임으로 낮추자 조금은 더 빨리 화면에 나타났다. 낮은 프레임을 가질 경우 속도가 빨라짐을 알 수 있었다. 프레임을 떨어뜨려서 전송률은 좋아졌지만 영상의 질은 조금 떨어짐을 알 수 있었다. 하지만 일반적으로 화상채팅같은경우도 5프레임 정도를 사용한다고 하니 걱정할 정도는 아님을 알 수 있다.

3.6 RFID를 이용한 위치인식

3.6.1 서 론

로봇의 현재 위치를 파악하여 로봇이 시작위치로부터 목적지 까지 얼마나 왔는지 또한 어디로 가야하는지 에대한 방향성을 제시하기 위해 RFID를 이용한 위치인식 기술 개발을 선택하였다.

3.6.2 RFID 시스템 설계

처음 로봇의 위치 인식을 구상할 때 먼저 로봇의 하단 부 공간이 비어있다는 것을 착안 바닥에 RFID를 깔고 이것을 인식하는 시스템을 제안하게 되었다. 로봇의 하단부의 협소한 공간을 최대한 이용하고 정확한 위치인식에 RFID 시스템은 상당히 알맞은 선택이었다.

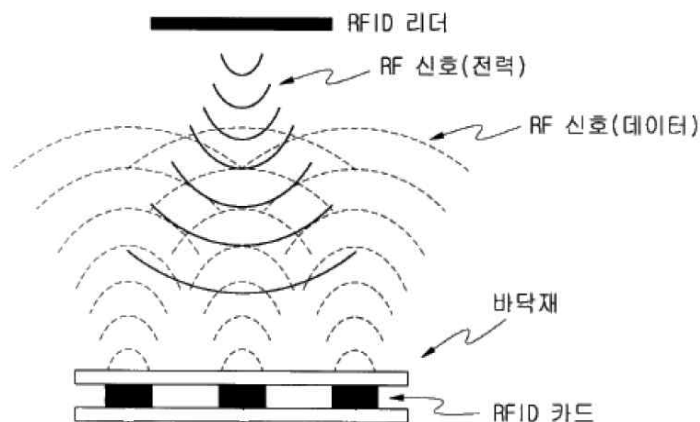


그림 1. RFID의 원리

(1) RFID 선택 및 착안점

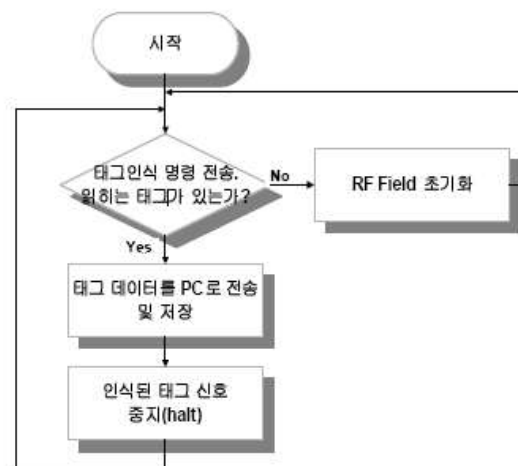
이동 로봇의 위치 및 방향을 인식시키기 위해서는 현재 위치에 대한 절대 좌표와 이동 변위에 대한 상대 좌표를 획득하고 이 절대 좌표에 상대 좌표를 부가하여 현재의 위치 및 방향을 인식한다. 이와 같은 이동 로봇의 위치 및 방향 인식 장치 및 방법은 RFID 좌표계를 통합 운영함으로써 RFID에 의해 오차 범위가 일정 크기 이내로 제한되면서 높은 속도를 얻을 수 있다. 또한 고유의 위치 정보가 부여된 다수개의 RFID 카드를 이동 로봇이 작업할 영역의 바닥에 매설해 놓게 되면, 이동 로봇이 이 작업 영역의

바닥면을 이동하면서 RFID 리더를 통해 RFID 카드를 검출하여 고유의 위치정보를 판독함으로써 이동 로봇 자신의 현재 위치를 인식할 수 있다. RFID 카드는 패시브 방식이기 때문에 별도의 전력 공급이 필요치 않다.

3.6.3 RFID를 이용한 위치인식 구현

(1) 모바일 로봇의 위치인식

대부분의 이동로봇은 엔코더, 관성항법 장치, 영상시스템과 같은 상대적인 항법 시스템을 이용해 위치를 인식한다. 그러나 이러한 상대적 계측 시스템은 이동거리 증가, 이동경로의 불규칙한 노면상태, 장애물과의 충돌 등 여러 가지 요소에 의해 발생하는 오차를 누적하게 된다. 엔코더의 경우 바퀴의 미끄럼 현상, 로봇의 기구학적 불균형 등의 이유로 오차가 증가된다. 이러한 문제를 해결하기 위해 절대위치 인식시스템의 개발이 요구되며 현재 까지 널리 연구되어 왔다. 몇 연구기관에서 레이저센서를 사용한 절대위치 인식 시스템에 관한 연구가 소개되었다. 레이저 센서를 사용할 경우 매우 정확한 위치계측이 가능하지만 그 활용범위가 제한적이며, 센서가 고가라는 단점으로 널리 활용되지는 못하고 있다. 반면 RFID는 비교적 저렴한 구축 비용으로 절대위치인식을 가능하게 한다.

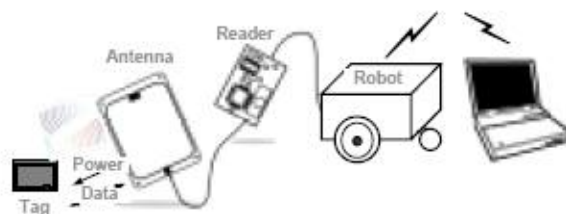


RFID기반의 위치인식실험을 하기 위해 로봇이 이동할 경로의 바닥 면에 구역을 설정하고 일렬로 간격마다 태그를 배치하였다. 배열된 태그 각각에는 부착된 위치의 절대좌표가 칩의 메모리에 저장되어있다. 로봇이 임의의 태그 위에 있으면 로봇의 하부에 장착된 안테나에 의해 형성된 RF신호를 받은 태그는 전력을 얻어 활성화된다. 안테나는 로봇하단에 부착된 상태로

지면에서 10cm높이에 있다. 태그들은 모두 활성화되며 이 태그들의 데이터를 순차적으로 모두 읽기 위해 그림과 같은 절차가 요구된다.

위치인식이 시작되면 리더기는 인식되는 태그가 있는지 조사한다. 리더기는 가장 우선적으로 들어오는 태그의 신호만을 인식하므로 RF신호 영역 내에 태그가 하나이상 있을 경우에도 우선적으로 신호가 전달된 한 태그의 정보만 반복해서 읽게 된다. 인식영역 내에 있는 다른 태그들의 데이터도 모두 받기 위해 우선 인식된 태그의 데이터를 PC에 저장한 후 이 태그의 신호를 일시적으로 중지시킨다. 그러면 리더기는 다음순서로 인식되는 태그의 정보를 읽을 수 있으며, 이절차를 한번 형성된 RF field 영역 내에 더 이상 인식되는 태그가 없을 때까지 반복한다. 태그의 정보를 모두 저장하고 더 이상 새로운 태그가 인식되지 않으면 RF field를 초기화함으로써 태그들의 신호중지상태를 해제한다. 로봇의 위치는 RF field를 초기화하기 전 수집된 태그들의 위치데이터를 기반으로 계산된다. 로봇이 이동하는 동안 이 과정을 반복함으로써 로봇의 위치를 실시간으로 예측할 수 있게 된다. RFID의 리더기는 태그의 존재여부만을 확인할 수 있으므로 리더기의 인식범위 내에 놓인 태그의 좌표 값을 통해 로봇의 위치를 결정하는 것은 리더기 인식범위만큼의 오차를 발생시킬 수 있다.

아래 그림은 RFID 시스템 동작 구성도를 나타내고 있다. 모바일 로봇이 움직임에 따라 로봇의 위치는 리더기를 통해 바닥의 태그정보를 입력받고 이는 컨트롤부에 수신하도록 되어 있다.

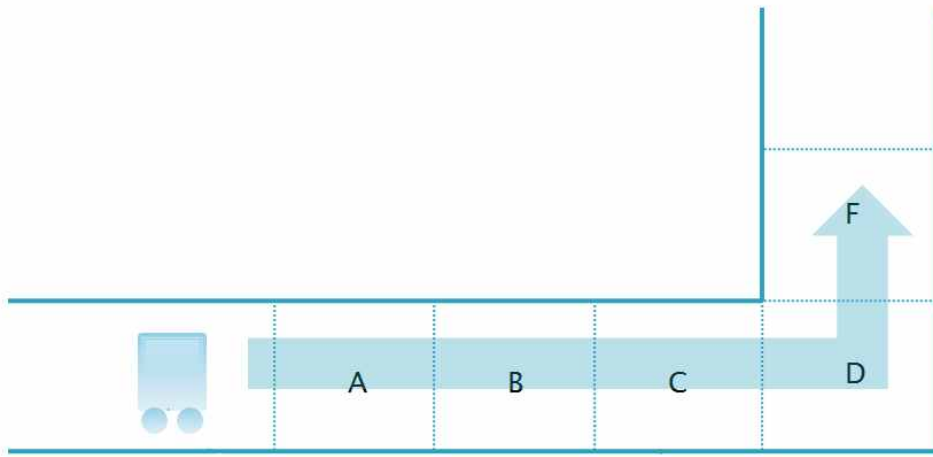


(2) RFID의 구역설정

모바일 로봇에서 RFID를 이용한 위치인식 구현하려면 태그의 배치에 따라 그 역할이 달라진다. 장애물을 회피하며 진행하려는 세밀한 위치를 인식하려면 태그를 바둑판형으로 배치를 하여 실험주행을 하는 것이 그의 따른 예가 되겠다.

이번 모바일로봇의 시험 주행과 그에 따른 위치인식을 위하여 우리가 설정한 태그배치는 구역을 설정한 후 구역간의 경계를 태그로 인식하는 방법이다. 시연하게 될 이동경로가 상당히 길며 또한 세밀한 위치보다 현재 이동하는 지역에 대한 정보가 더 중요시 되기에 구역설정에 따른 태그배치를 설정하였다.

우선 시연을 할 경로는 90도로 꺾여있는 복도이다. 이동로봇은 센서를 통해 중앙으로 이동을 할 것이고 설정되어있는 경로를 이탈하지 않는다는 전제하에 위치를 측정할 것이다.

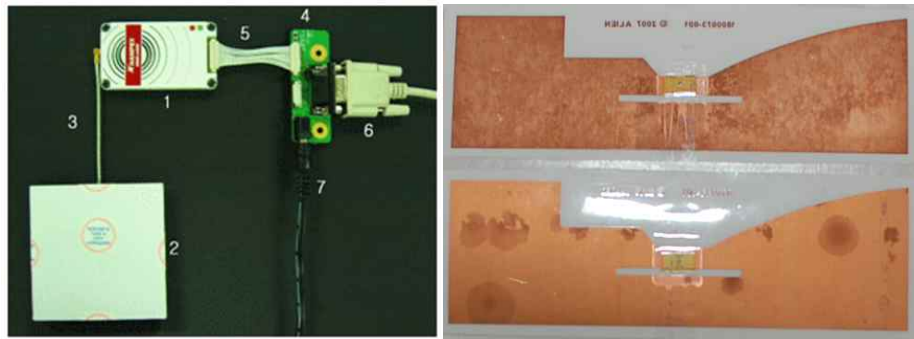


그림과 같이 구역을 설정을 하고 고유코드를 지정한다. 그 고유코드는 경계에 부착될 태그에 동일한 정보가 입력된다. 구역과 구역의 경계에는 태그가 배치될 것이고 모바일 로봇은 그 태그를 지나갈 때 로봇하단에 부착되어있는 리더기가 그 태그를 인식하게 되면 현재 어떤 구역으로 진입했는지를 판단할 수 있다. 복도간의 양 쪽의 거리는 2m로 태그는 간격 10cm 정도로 총 10개가 직선으로 부착된다. 실험시 사용한 태그는 스티커형으로 부착이 용이하며 리더기가 인식했을 시 동일한 위치정보를 전송하도록 같은 구역에는 같은 정보를 저장한다.

(3) Reader와 Tag

그림은 실제 실험에 사용된 RFID 리더와 태그를 보여준다. 본 실험에 사용된 RFID 시스템은 passive type으로 RFID 시스템의 주파수대역은 908.5~914 MHz이며, 크기는 3cm×6cm 이다. 태그와 리더사이의 인식거리는 직선거리로 약 3m 이다. 하지만 로봇하단에 사용되므로 인식거리는 10cm로 실험하게 된다. 그림의 2번에 보이는 리더 안테나의 크기는

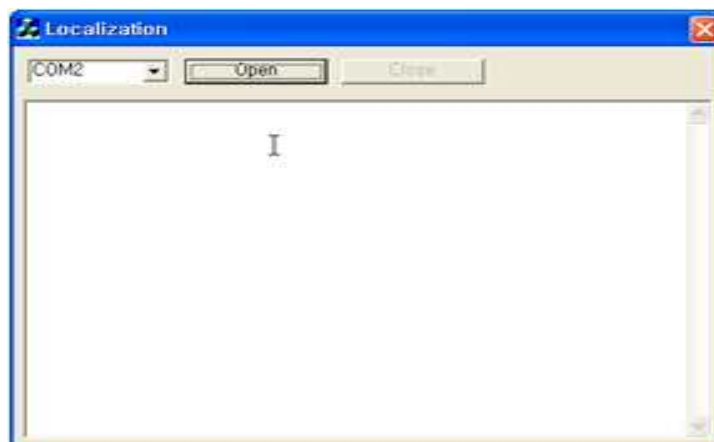
10cm×10cm 이며, tag 는 epoxy type으로 크기는 2cm×8cm 이다. 스티커 형으로 탈부착이 가능한 형태로 한 구역에 10개씩 사용된다.



앞 절에서 언급했던 것처럼 리더기는 로봇하단에 부착될 것이다. 태그에는 경로에 대한 정보를 입력한후 구역에 설치될 것이다. 태그의 정보는 26개의 숫자와 문자로 정의 되어있으며 이는 변경가능한 정보로 구역에 따른 정보를 저장하게 된다. 이번 시연에서는 A구역에서 B구역으로 넘어가는 경계는 “1111111111111111”의 정보를 입력하였고 B구역에서 C구역으로 넘어가는 경계에는 ”222222222222”의 정보를 입력하는 식으로 정보를 입력하여 구역을 설정하였다.

(4) 위치인식에 관한 콘솔 제작

RFID가 하드웨어라면 소프트웨어의 제작이 요구되었다. 즉 위치인식을 확인할 수 있는 콘솔 학습및 제작을 하였다. 로봇이 지나가면서 태그를 읽는 것을 제어부에서 확인할 수 있게 제작 되었고 이는 지시하는 사람이 로봇의 위치를 제자리에서 확인 할 수 있게 하는 중요한 작업으로 VC++을 이용한 시리얼 통신 학습을 기본으로 작업하였다.



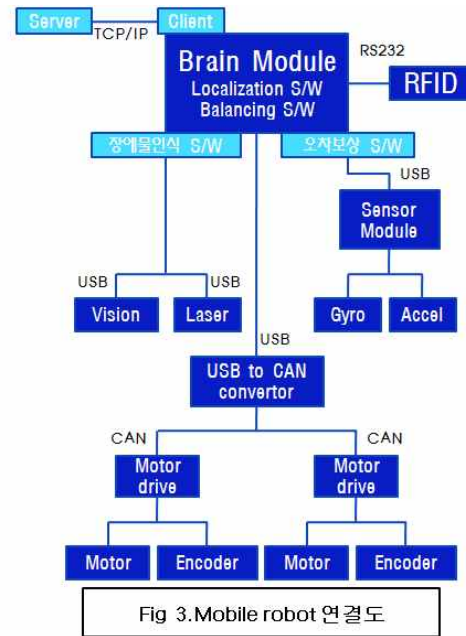
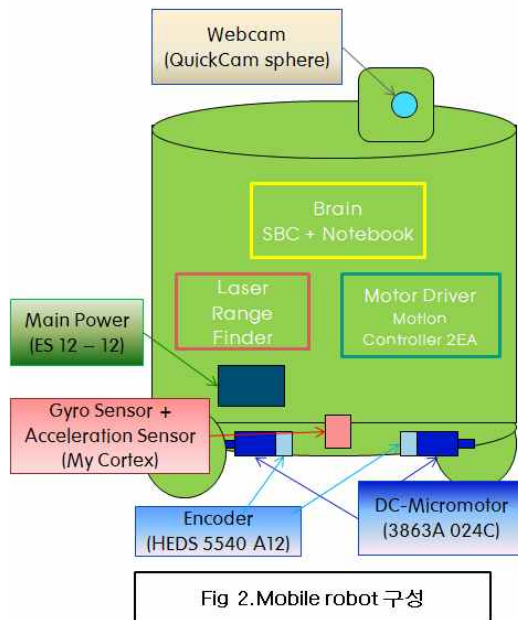
그림은 완성된 콘솔 화면으로 포트제어와 시작을 설정하면 로봇이 움직이며 태그를 지날 때 마다 인식하도록 프로그래밍 되었다.

3.6.4 결론

이번 RFID를 이용한 연구를 통해 기존의 위치 인식 시스템에서의 문제점을 보완하고 효과적인 위치 인식을 위해 RFID시스템을 이용한 위치 인식 기술을 제안하였다. 공간 내에 이동하는 물체의 위치를 RFID tag와 reader를 통해 효과적으로 파악할 수 있음을 알 수 있었다. RFID기반의 절대위치 인식시스템은 많은 수의 태그를 요구하고 또한 모든 환경에 적용되기에는 제약이 따르므로 특정 위치에서 상대적 항법시스템의 계측오차를 보정하는 방향으로도 활용될 수 있다.

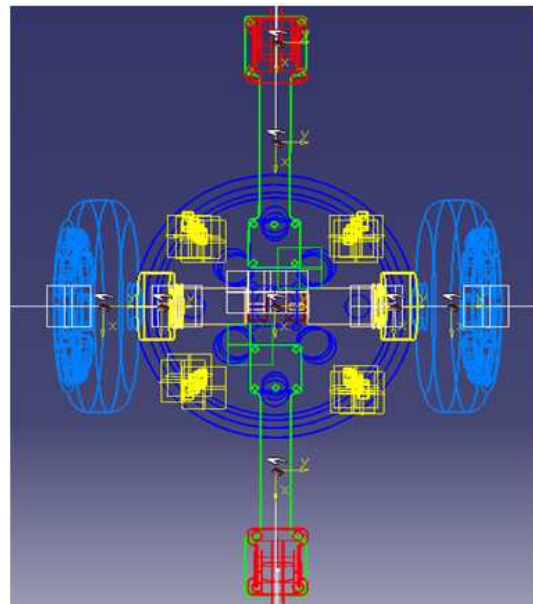
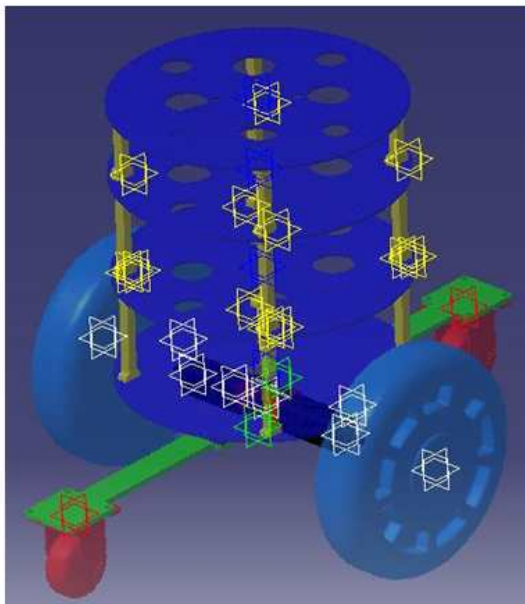
4. 상세설계 결과

4.1 로봇의 구성도 및 각 Module 간 통신 연결도

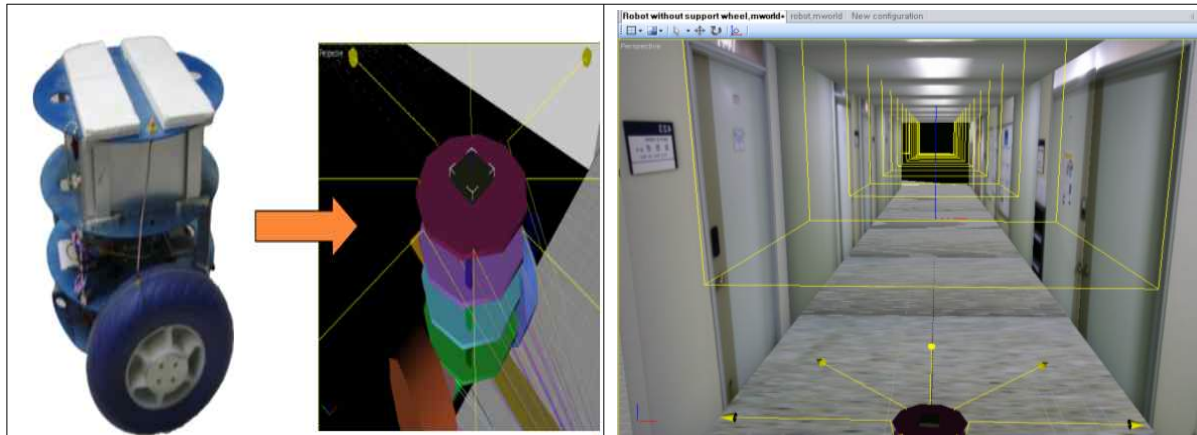


4.2 제작 도면

▶ CATIA를 이용한 도면의 제작

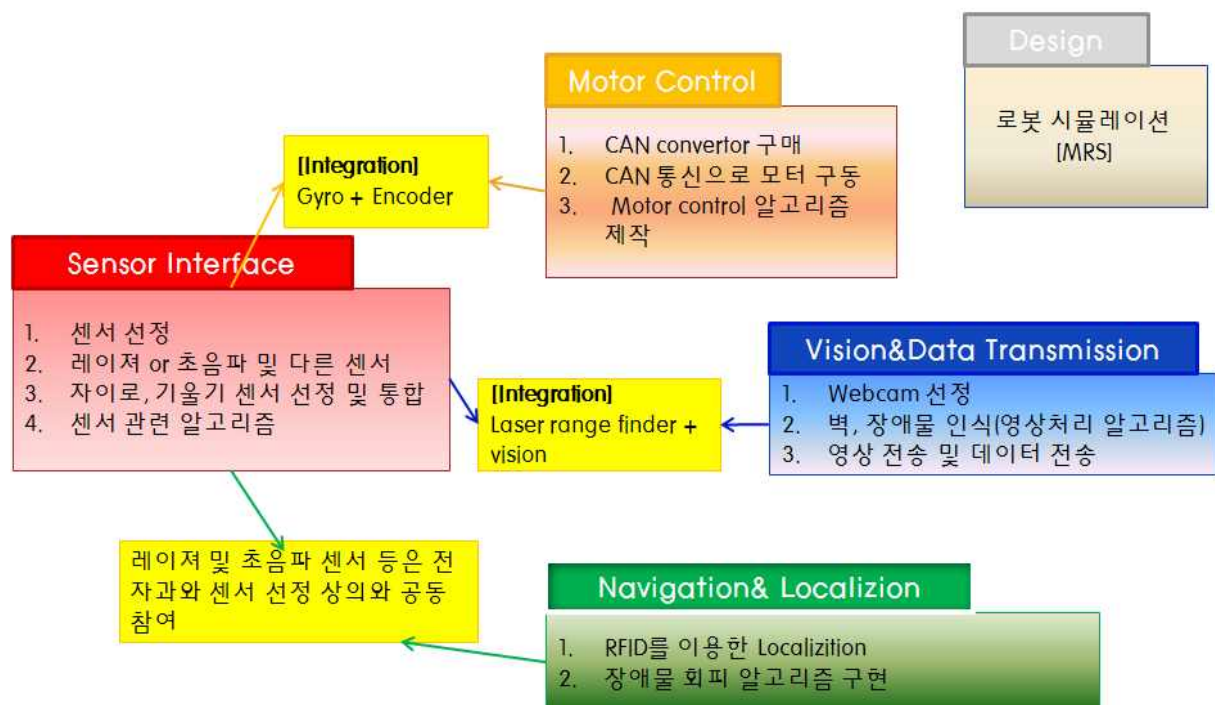


- 시뮬레이션을 통한 설계 타당성 확인



4.3 각 팀의 목표 및 상호 연계도

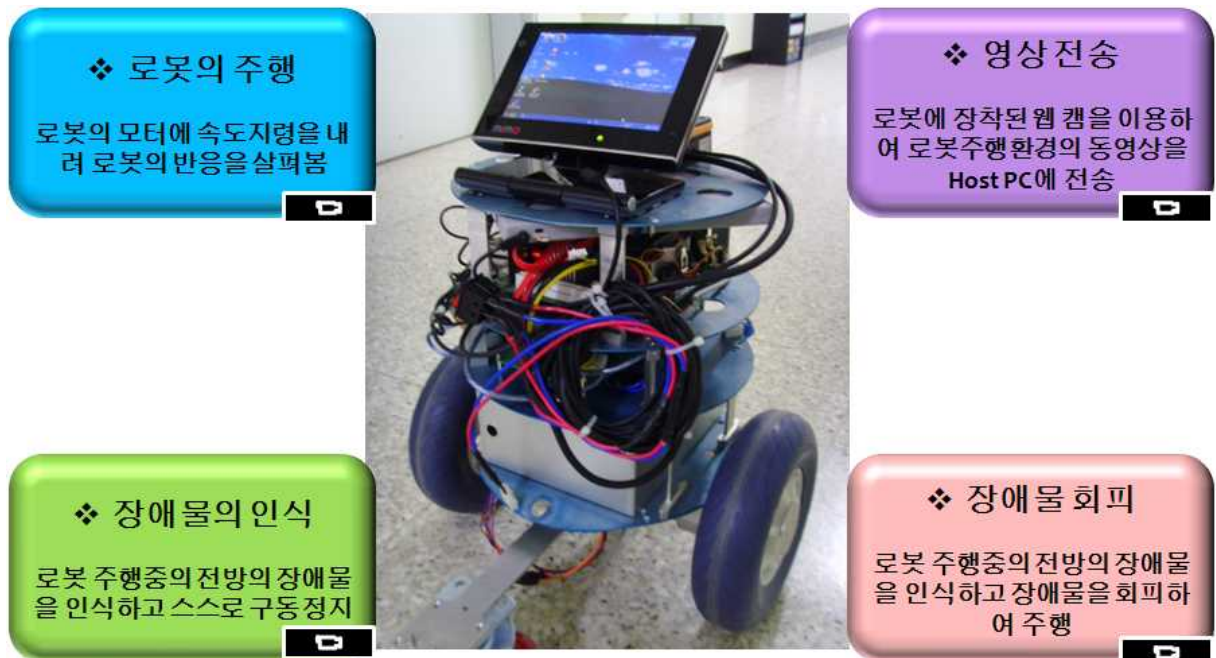
❖각 팀의 목표와 연관성



4.4 실제 완성 로봇 및 모듈 구성



4.5 결론 몇 보완



- ① 로봇의 밸런싱을 위하여 조금 경량의 재료를 이용하여 로봇을 제작
 -로봇에 장착된 배터리(메인 전원부, SBC 전원부)의 용량을 낮출 필요가 있음.
- ② 각종 소프트웨어의 연계성을 고려하여 버퍼가 적은 프로그램으로의 개선이 필요.
 -영상전송 및 비전 영상처리 프로그램의 과도한 메모리 부하로 인하여 연산 과정 및 전송과정에서의 버퍼의 개선
- ③ 로봇 디자인의 개선이 필요.
 -각각의 모듈의 케이스화가 필요.

5. 팀원간 역할 분담

이 름 (Name)	역 할	참여도(%)
김 지웅 (Kim, Ji woong)	Motor control (CAN을 통한 통합 모터드라이브 및 모터 구동)	15
이 형대 (Lee, Hyung dai)	Motor control (Robot balancing 알고리즘 구현)	15
권 원주 (Kwon, Won ju)	Vision & Data transmission (Webcam을 통한 장애물 인식 알고리즘 구현)	15
김 인후 (Kim, In hoo)	Vision & Data transmission (Mobile robot의 영상 및 Data 전송알고리즘 구현)	15
권 진만 (kwon, Jin man)	Sensor interface (Gyro&Lazer Sensor를 이용 Sensor Module 제작)	15
김 태우 (Kim, Tae woo)	Navigation & Localization (RFID를 이용한 Localization 알고리즘 구현)	15
이 창욱 (Lee, Chang wook)	Navigation & Localization (Mobile robot의 Navigation 알고리즘 구현)	15

참고 문헌

1. David P. Anderson. (2007), "nBot Balancing Robot"
<http://geology.heroy.smu.edu/~dpa-www/robo/nbot/>
2. S. W. Nawawi, M. N. Ahmad, and J. H. S. Osman. (2008), "Real-Time Control of a Two-Wheeled Inverted Pendulum Mobile Robot". pp 216~217.
3. Sophan Wahyudi Nawawi, Mohammad Noh Ahmad, Johari Halim shah Osman. (2005), "Modeling of two-wheels inverted pendulum mobile robot" pp 3~7.
4. Seonghee Jeong, Takayuki Takahashi. (2008), "Wheeled inverted pendulum type assistant robot : design concept and moblie control". Intel serv Robotics (2008). pp 313~320.
5. S. W. Nawawi, M. N. Ahmad, and J. H. S. Osman. (2002),"Real-Time Control of a Two-Wheeled Inverted Pendulum Mobile Robot"
6. Jung S. W and Lee M. H (2008),"OpenCV를 이용한 컴퓨터 비전 실무 프로그래밍"
7. Kim K, Lee C. W, Xu S, Cui Y, H. (2008),"로봇의 자율 항해를 위한 비전 기반의 객체 인식"
8. Lee I, Choi N. Y, Kim I, J (2007),"선 추적과 템플릿매칭을 이용한 악보 인식 시스템"
9. Lee K. M, 2007"이질적 템플릿매칭의 융합을 이용한 얼굴 영역 검출"
10. Hong J. P, Lee G. D, (1991)"숫자의 실시간 인식"
11. Kim J. S, Byun Y. C, Kim G. W, Choi Y. W, Lee Y. B, (1999), "지로 서식 문서의 인쇄체 숫자 인식"
12. Park J. S, Dong J. Y, (2003), "얼굴 특징 정보를 이용한 얼굴 방향성 검출"
13. Kang D. J, Ha J. E, "Visual C++을 이용한 디지털 영상처리"
14. Lee S. W, Kim H. Y, (1996), "투영법을 이용한 회전불변 템플릿 매칭"
15. Hong, K. D., Brydon, A., Leweke, T. and Thompson, M. C., (2004), "Interactions of the Wakes of Two Spheres Placed Side by Side," *Trans. of the KSME(B)*, Vol. 23, No. 5, pp. 137~145.
16. Youn S. W (2003). "TCP/IP 소켓 프로그래밍"

본 교재는 2008년도 수도권 대학 특성화 지원사업에 의한 결과임