바퀴/4 족 동작 전환으로 계단 및 문턱 오르기가 가능한 서비스 하이브리드 이동 로봇 개발

Development of a Service Hybrid Mobile Robot for Climbing Stairs and Thresholds by Switching Wheel and Leg Gait

김 진 백*, 김 병 국
(Jin-Baek Kim and Byung-Kook Kim)

Abstract : In this paper, we developed a new hybrid mobile robot which can climb stairs and go over thresholds by crawl gait with embedded real-time control software. This robot is also categorized into hybrid robot that has advantages of wheeled mobile robot and legged mobile robot, but adopts gait feature of crocodile named belly crawl. We imitated the belly crawl using four legs of 2 DOF, four omni-directional wheels, and embedded control software which controls legs and wheels. This software is developed using RTAI/Linux, real-time drivers. As a result, the new hybrid mobile robot has crawl gait. Using this feature, the new hybrid mobile robot can climb stairs and go over thresholds just by path planning of each leg with size of stairs and thresholds, and computing the movement distance of robot body center without considering stability. The performance of our new hybrid mobile robot is verified via experiments.

Keywords : hybrid mobile robot, real-time embedded system, real-time software architecture

I. 서론

한 곳에만 머물지 않는 사람들의 요구를 충족시키기 위해 서비스 로봇은 이동할 수 있어야만 한다. 그러나, 사람들이 있는 곳의 지형은 항상 평지가 아니라 계단 및 둔턱 같은 불규칙한 지형을 포함한다. 따라서, 서비스 로봇을 바퀴주행만 가능하게 해선 로봇의 사용이 제한될 수밖에 없다. 이러한 한계를 벗어나기 위해 평지뿐만 아니라, 계단 및 둔턱 같은 지형에서도 충분한 이동성을 가지고 있는 서비스 로봇의 개발이 필요하다.

일반적인 이동 서비스 로봇은 바퀴주행 로봇의 형태를 띠고 있다. 이 바퀴 주행 로봇의 바퀴를 구동시키는데 모터를 사용한다. 이 모터만 자유롭게 제어한다면, 바퀴 주행 로봇은 평지라면 어디든 이동해 갈 수 있다. 모터에 대한 연구와 이를 제어하기 위한 연구는 오랜 세월 동안 진행 되어져 다른 어느 분야만큼 충분한 지식과 경험이 축적되어 있다. 따라서, 바퀴 주행 로봇이 이동 로봇의 가장 효율적이라 말 할 수 있다. 그러나, 오직 한가지 문제가 이를 가로막고 있는데, 이는 바퀴의 본질적인 특성으로 인한 것으로 바퀴 주행 로봇은 계단 이나 둔턱, 또는 불규칙한 지형에서는 이동 능력이 현저히 떨어지거나 아예 이동 불가능이 될 수 있다는 점이다. 바퀴 주행 로봇은 그 목적과 형태에 따라 이미 많은 연구가 진행 되어 왔는데, 바퀴 주행 로봇은 자동차 같은 형태를 취하지 않고 좌우로 움직일 때 공간소모가 적은 전방향 바퀴를 사용하는 것이 일반적이다. 이러한 형태의 로봇은 이미 kit형태로 판매가 될 정도로 많은 연구가 진행 되어져 왔다[1]. 이에 멈추지 않고 Robomoto 연구소에서는 전방향 바퀴를 사용하지 않고 일반적인 바퀴로도 전방향 바퀴와 같은 효과를 내는 방법을 연구하여 개발해 내었다[2].

많은 장점을 가진 바퀴 주행 로봇의 한계점을 극복하기 위해 연구원들은 다리 이동 로봇을 개발하였다. 이는 다리를 제어할 수 있는 방법이 많아 어떠한 지형에서도 로봇에게 이동능력을 확보해 준다는 큰 장점을 가지고 있다. 그러나, 각 다리가 하나의 manipulator이므로 로봇을 이동 시키기 위해선 너무나 많은 제어 계산이 필요하고 안정성의 확보가 어렵고 전력소비도 심하다는 단점을 갖고 있다.

로봇의 안정성 확보에 많은 이득을 지니고 있는 다족 로봇에 대한 연구가 많이 진행 되었다. 6족 이상의 로봇은 하나의 다리가 어떻게 움직여도 로봇의 안정성에는 어떠한 영향을 주지 못 한다는 장점을 가지고 있어서 가장 먼저 연구된 분야이다. 물론, 많은 다리를 가지고 있으므로 로봇이 이동하기 위해 제어 연산의 양과 복잡함이 높다는 단점을 가지고 있다. 좋은 예로, 6족과 각 다리의 압력 센서를 통해 자세 제어 및 이동이 가능해 사람 대신 험준한 지형을 뚫고 오지를 탐사하기 위해 개발된 LAURON3가 있다[3].

다족 이동로봇 중 가장 일반적인 것이 4족 로봇이다. 이는 우리 주변 동물들의 가장 흔한 형태가 4족인 덕분에 사람들이 가장 오랫동안 관찰해 왔고 4족 동물들에 대한 연구도 많이 되어서 그 동물들의 걸음새를 흉내내기 쉽다는 데 이유가 있다. 이런 4족 형태에 특정 목적에 맞도록 설계된 TITAN series는 오랫동안 연구가 진행 되어져 온 대표적인 4족 로봇이다[4]. 4족 로봇은 움직이기 위해 다리 하나를 이동 시키면 지면에 나머지 3개의 다리가 지지하고 있다는 점에서 2족 보다는 높은 안전성을 확보할 수 있지만, 로봇의 무게 중심이 다리 세 개로 이루어진 면에 포함되어져야 한다는 요구사항은 제어 구조에 또 다른 제약 조건이 된다.

사람들의 가장 큰 이목을 받는 다족 이동로봇은 2족(biped) 로봇이다. 이는 우리 인간 외에 항상 직립보행을 하는 유일한 창조물인 터이다. 바로 이러한 특징으로 사람들은 2족로

* 책임저자(Corresponding Author)
논문접수 : 2007. 9. 28., 채택확정 : 2007. 10. 2.
김진백, 김병국 : 한국과학기술원 전자전산학과 전기전자전공
(jbkim@rtcl.kaist.ac.kr/bkkim@ee.kaist.ac.kr)

봇에 대해 다른 형태의 로봇보다 더 친근함을 가지고 있고 각종 SF영화에서 2족 로봇의 등장으로 보다 익숙해져 있었기 때문이다. 그러나, 이 2족 로봇은 직립보행이라는 가장 어려운 인간의 자세를 흉내내야 한다는 부담이 있고 지면으로부터 어느 로봇들 보다 높은 곳에 무게중심이 있어 제어하기가 어렵다[5,6].

크게 두 가지 형태로 나누어 지는 이동 로봇 각각의 장점이 서로의 단점을 보완해 주게 된다면, 서로의 장점을 동시에 가지게 되는 로봇이 될 것이다. 바로 바퀴와 다리의 혼합형태인 하이브리드(hybrid) 로봇을 연구하기 시작하였다. 이 하이브리드 로봇은 일반적으로 바퀴와 다리를 동시에 가지고 있어서 평지에서는 바퀴를 이용하여 빠른 주행 속도로 이동하고 바퀴로 더 이상 움직일 수 없을 때 다리를 이용하여 그 상황을 극복한다. 이러한 바퀴와 다리가 어떠한 방식으로 결합되어 동작하는가에 따라 로봇 전체 구조에서부터 제어 프로그램까지의 형태가 결정될 것이다. 하이브리드 로봇에 대한 연구 중 가장 대중의 이목을 받았던 것은 NASA의 화성 탐사에 사용된 Sojourner라는 화성 탐사로봇이다. 얼핏 보면 이는 단순한 바퀴 주행 로봇이지만, 바퀴와 로봇 몸체의 연결이 일반적이지 않다. 즉, 바퀴가 몸체에 고정이 되어 있는 경우 지형과 몸체가 항상 수평이 되는 방법 밖에 존재하지 않지만, 움직일 수 있는 완충장치를 이용하여 바퀴와 몸체를 연결하게 되면 불규칙한 지형에서도 안정성을 확보할 수 있고 완충장치를 어떻게 제어하는가에 따라 장애물을 넘어 갈 수도 있게 된다[7]. 파리 6 대학의 로봇 연구소에선 이에 더 나아가 추가적인 움직임이 가능하도록 더 많은 자율성을 추가하여 주행 중에 높은 안정성을 확보하도록 개발한 Hylos robot을 통해 이를 확인하였다[8,9]. 일본 산업기술종합연구소의 기계 공학 연구실에선 3 DOF(Degree of Freedom)를 가진 앞다리와 1 DOF를 가진 뒷다리, 그리고 이를 위해 새로운 메커니즘을 통해 둔 턱을 올라갈 수 있는 하이브리드 로봇을 개발하였다[10]. 그러나, 이러한 하이브리드 로봇들은 바퀴와 추가적인 움직임을 위한 DOF나 다리가 결합되어진 형태로 항상 지면에 바퀴가 닿아 있어 바퀴와 추가적인 DOF를 동시에 제어해야 하고 전방향 바퀴를 사용하지 않아 좌우로 움직일 때 공간의 제약이 따른다는 단점을 가지고 있다.

본 논문에서는 새로운 형태의 하이브리드 로봇을 제안하고 이에 대한 프로토타입인 CrocoHybrid를 구현하여 그 성능을 살펴보기로 한다. 기존의 연구되어진 하이브리드 로봇의 형태와는 다른 방식으로 문턱과 계단을 오르기에 적합하도록 설계하였다. 기존의 하이브리드 로봇은 다리의 끝에 바퀴가 설치되어 있어서 지면의 굴곡에 따라 수직방향으로 움직일 수 있는 여유가 있어서 바퀴만 있는 바퀴주행 로봇에 비해 높은 자유도를 가질 수 있다. 그러나, 평지 주행 시에도 바퀴를 조향하기 위해서 다리를 제어해야 하고 발이 아닌 바퀴가 있으므로 다리로써의 능력을 활용하기 어렵다라는 단점을 가지고 있다. 실제 실내·외 환경에서 지면의 굴곡이 급격하게 변하는 곳은 거의 드물고 사람들의 활동 영역에서는 심한 굴곡 보다는 둔 턱이나 계단 같은 수직 장애물들이 많기 때문에 일반적인 바퀴 주행 이동 로봇에 둔 턱이나 계단을 오를 수 있는 기능을 추가하게 된다면, 활동 영역이 보다 더 넓은 서비스 이동 로봇으로써의 역할을 할 수 있을 것이다. 따라서, 본 논문에선 바퀴 주행 로봇을 기반으로 다리를 추가하여 문 턱과 계단을 오르도록 하는 하이브리드 로봇을 개발하는데 초점을 맞추었다. 이에 따른 시뮬레이션과 실험을 통해서 그 성능을 입증하도록 한다. 또한, 이동 로봇이므로 가능한 저전력으로 동작하도록 하고 개방형 실시간 운영체제인 Linux와 실시간 확장형인 RTAI를 사용하여 제어 시스템의 실시간성을 추가해 보다 정확한 제어가 가능하도록 목표로 하였다.

논문은 총 6장으로 구성이 된다. 2장에서는 새롭게 제안된 하이브리드 로봇의 구조적인 형태와 그에 따른 로봇 모델과 기구학에 대해서 기술 하였고, 3장에서는 이 로봇을 제어하기 위한 소프트웨어의 구조 및 설계에 대한 내용을 기술하였다. 4장에서는 제안된 하이브리드 로봇의 형태로 인한 걸음새와 다리의 움직임을 위해 경로 설정 알고리즘에 대해서 다룬다. 5장에서는 4장에서 제시된 알고리즘을 시뮬레이션을 통해서 확인하고 구현된 하드웨어와 소프트웨어를 통해 실험 및 검증을 한다. 마지막으로 6장에서는 결론을 맺도록 한다.

II. CrocoHybrid 설계

1. CrocoHybrid의 기구 구조

CrocoHybrid는 일반적인 4족 로봇의 형태를 따르지만, 로봇의 각 다리가 갖는 자유도(DOF)는 최소한으로 설정하여 전체 로봇의 하중을 줄이고 다리 제어 시 계산량을 줄이도록 하기 위해 2 자유도를 갖도록 하였다. 여러 서보 모듈들을 연결 프레임 이용하여 링크(link)를 구성하고 마지막 링크인 다리는 로봇의 무게는 지탱하되 최대한 적은 무게가 되도록 제조하였다. 게다가 CrocoHybrid는 각 다리의 동작범위에 바퀴가 있어서 서로의 움직임에 방해가 될 수 있고 앞다리와 뒷다리가 서로의 동작 범위가 겹치게 될 수 있다. 이를 방지하기 위해, 일반적인 4족 다리의 움직임과 다르게, 앞다리와 뒷다리의 방향성을 반대로 하였다.

로봇에 사용된 바퀴는 전방향 이동이 가능하도록 하기 위하여 전방향 바퀴(omni-directional wheel)를 사용하였다. 이 전방향 바퀴를 로봇에 설치하는 방법은 여러 가지가 있겠지만, 크게 삼각구도와 사각구도로 나누어진다. 그러나, 일반적으로 삼각구도가 많이 사용되는데, 이는 사각구도보다 바퀴를 1개 덜 사용하게 되어 시스템 자원도 덜 사용할 수 있기 때문이다. 그러나, 본 논문에서는 바퀴를 일종의 로봇 지지대로 사용하고자 하므로 바퀴가 지지대로 사용하였을 때 로봇에게 충분한 안정도를 제공할 수 있어야 하기 때문에 사각구도를 사용하였다. 결과적으로 바퀴 그 자체만으로는 잉여 자유도를 추가하는 것이 되지만, 삼각구도에 비해 특정 방향으로의 직진 주행에 대한 제어가 간략하게 되고 로봇의 수평이 기울여 졌을 때도 지면과의 접촉면이 많아지므로 로봇에게 보다 높은 안정도를 제공하여 줄 수 있다.

이렇게 구성된 구동 파트 위에 제어시스템과 인터페이스 보드, 전지와 구동회로 등을 stack형태로 쌓아 전체적인 로봇을 구성시켰다. 개발된 하이브리드 로봇은 그림 1에서 보는 바와 같다.

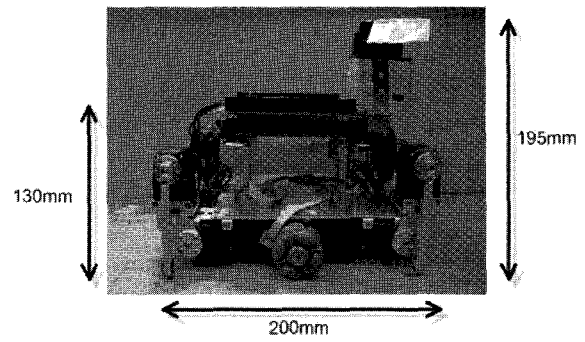


그림 1. 하이브리드 로봇.
Fig. 1. Hybrid robot.

2. CrocoHybrid의 기구학

본 논문에서 구성한 하이브리드 로봇의 기구학적 특성은 다리와 바퀴가 제각기 독립적으로 동작이 가능하다는 것에 있다. 따라서, 바퀴의 동작에 다리가 방해가 되어서는 안되도록 설계하여 그림 2와 같이 구성하였다. 이를 바탕으로 각 다리의 축을 정하여 역기구학(inverse kinematics)을 구하여야 한다[11].

이 축들은 역기구학이 최대한 간단하게 나올 수 있도록 모든 다리의 관절의 축은 로봇 몸체의 z축과 방향이 같도록 하였다. 이러므로 로봇 몸체에서 각 다리로의 변환 매트릭스(transformation matrix)가 간략하게 되는데, 로봇 몸체에서 각 다리의 첫 번째 관절로의 변환 매트릭스는 단순한 평행이동이 되고 나머지 변환 매트릭스는 모든 다리가 동일하게 된다. 이렇게 변환 매트릭스가 비슷하게 구성되면 역기구학을 구할 때 각 다리마다 따로 계산을 해 줄 필요가 없이 간단한

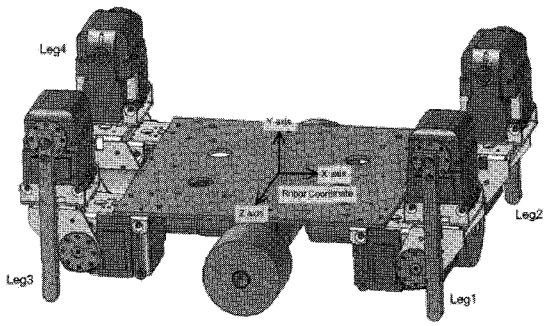


그림 2. 하이브리드 로봇 모델.
Fig. 2. Hybrid robot model.

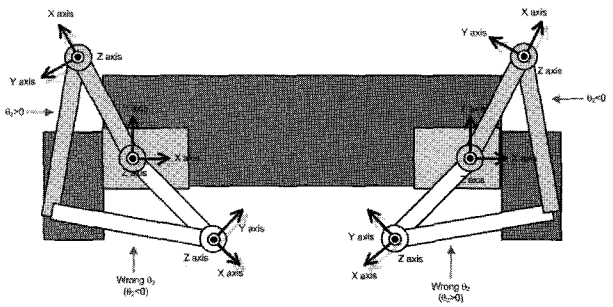


그림 3. 다수의 역기구학 해를 제거하기 위한 조건.
Fig. 3. Condition for only one solution.

수식을 첨가하므로 각 다리의 역기구학을 구할 수 있다는 장점이 있다. 각 다리의 역기구학은 2 자유도여서 간단한 문제이다. 다만, 다수의 역기구학 해가 존재하여 이를 없애기 위해 모든 다리는 항상 중력 방향을 향한다는 조건을 추가하였다. 그림 3은 이를 나타내주고 있다.

역기구학을 위한 과정은 다음과 같다.

$$
{}^{1}A_4 = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & \pm Bx + l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & +By + l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2) \\ 0 & 0 & 1 & \pm Bz + d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}
$$

모든 다리의 transformation matrix는 (1)로 통일된다. 다만, 각 다리의 부호만 아래와 같이 달라진다.

다리 1(Leg 1)	다리 2(Leg 2)
$+Bx, +By, +Bz$	$+Bx, +By, -Bz$
다리 3(Leg 3)	다리 4(Leg 4)
$-Bx, +By, +Bz$	$-Bx, +By, -Bz$

따라서, 모든 다리의 역기구학의 방정식은 하나로 통일될 수 있고 위의 부호만 다리에 맞춰 바꿔주기만 하면 된다. 이로 인해 실제 소프트웨어에선 계산량을 약 1/4로 줄일 수 있다. 다리의 끝 위치를 (P_x, P_y)라 하면 (2)와 (3)을 얻게 된다. CrocoHybrid 형태상 P_z는 항상 일정하므로 계산에 고려하지 않는다. 여기서 변하지 않는 고정 값을 가지는 Bx와 By를 빼주어 각 다리의 joint1에서부터의 상대적 위치인 (P_x', P_y')를 얻게 된다.

$$P_x = Bx + l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) \tag{2}$$

$$P_y = By + l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2) \tag{3}$$

$$P_x' = P_x - Bx \tag{4}$$

$$P_y' = P_y - By \tag{5}$$

(4)와 (5)를 조합하면

$$
\begin{aligned} P_x'^{\,2} + P_y'^{\,2} &= (l_1 c_1 + l_2 c_{12})^2 + (l_1 s_1 + l_2 s_{12})^2 \\ &= l_1^2 + l_2^2 + 2l_1 l_2 (c_1 c_{12} + s_1 s_{12}) \end{aligned} \tag{6}
$$

cos함수는 다음 특성을 가지고 있으므로,

$$\cos(\theta_1 - (\theta_1+\theta_2)) = c_1 c_{12} + s_1 s_{12} = \cos\theta_2 \tag{7}$$

(6)은 다음과 같이 정리가 된다.

$$P_x'^{\,2} + P_y'^{\,2} = l_1^2 + l_2^2 + 2l_1 l_2 c_2 \tag{8}$$

$$\cos\theta_2 = \frac{P_x'^{\,2} + P_y'^{\,2} - l_1^2 - l_2^2}{2l_1 l_2} \tag{9}$$

$$\sin\theta_2 = \pm\sqrt{1-\cos^2\theta_2} \tag{10}$$

(10)의 부호에 의해 2개의 해가 나오는데, 그림 3의 조건을

통해 하나는 배제하고 이로 인해 얻어지는 θ_2 는 (11)과 같다.

$$\theta_2 = atan2(\sin\theta_2, \cos\theta_2) \tag{11}$$

θ_1 을 구하는 과정은 아래와 같다. (4)와 (5)를 정리하면,

$$P_x' = (l_1 + l_2 c_2)s_1 + l_2 s_2 c_1 \tag{12}$$

$$P_y' = (l_1 + l_2 c_2)c_1 - l_2 s_1 s_2 \tag{13}$$

(12)와 (13)를 이용하면 아래의 식이 얻어진다.

$$\frac{P_y'}{P_x'} = \frac{(l_1 + l_2 c_2)s_1 + l_2 s_2 c_1}{(l_1 + l_2 c_2)c_1 - l_2 s_1 s_2} = \frac{\frac{s_1}{c_1} + \frac{l_2 s_2}{(l_1 + l_2 c_2)}}{1 - \frac{l_2 s_1 s_2}{(l_1 + l_2 c_2)c_1}} = \frac{\tan\theta_1 + \frac{l_2 s_2}{(l_1 + l_2 c_2)}}{1 - \tan\theta_1 \frac{l_2 s_2}{l_1 + l_2 c_2}} \tag{14}$$

$$= \frac{\frac{s_1}{c_1} + \frac{l_2 s_2}{(l_1 + l_2 c_2)}}{1 - \frac{l_2 s_1 s_2}{(l_1 + l_2 c_2)c_1}} = \frac{\tan\theta_1 + \frac{l_2 s_2}{(l_1 + l_2 c_2)}}{1 - \tan\theta_1 \frac{l_2 s_2}{l_1 + l_2 c_2}}$$

(14)는 $\tan(\phi_1 + \phi_2) = \frac{\tan\phi_1 + \tan\phi_2}{1 - \tan\phi_1 \tan\phi_2}$ 의 형태를 띠고 있으므로, (15)를 만족한다.

$$\tan^{-1}\left(\frac{P_y'}{P_x'}\right) = \tan^{-1}(\tan\theta_1) + \tan^{-1}\left(\frac{l_2 s_2}{(l_1 + l_2 c_2)}\right) \tag{15}$$

결국, θ_1 은 다음과 같이 얻어지게 된다.

$$\theta_1 = \tan^{-1}\left(\frac{P_y'}{P_x'}\right) - \tan^{-1}\left(\frac{l_2 s_2}{l_1 + l_2 c_2}\right) = atan2(P_y', P_x') - atan2(l_2 s_2, l_1 + l_2 c_2) \tag{16}$$

3. CrocoHybrid의 hardware

자율 주행 및 4개의 다리를 제어하기 위한 시스템은 임베디드 보드를 사용하였는데, Falinux사의 EZ-X5(Xscale board)를 채택하였다. 이 EZ-X5는 400MHz ARM RISC MCU를 가지며 다양한 통신 포트와 64Mbytes용량의 RAM/ROM, 그리고 160개의 확장 pin을 통한 높은 확장성을 통해 쉬운 개발 환경을 제공한다.

로봇의 바퀴를 주행시키고 다리를 조절하기 위해 DC 모터를 사용하였다. 다리에 사용된 모터는 로보티즈(robotis)의 CX-28이라 불리는 모듈로써 DC모터와 마이크로 컨트롤러가 내장되어 데이시 체인(daisy chain) 연결방식으로 모듈들을 연결할 수 있어 모듈간의 연결이 쉽고 제공된 프로토콜에 맞게 명령을 전송시키면 구동시킬 수 있어 사용이 용이하다[12]. 단 이 모듈은 CAN 통신을 기반이다. 따라서, 로봇 전체 통신을 여러 가지 방식으로 구성하면 구조상의 복잡함과 통신상의 병목 현상이 발생할 수 있어서 바퀴 모터 제어기도 CAN 통신으로 통일하기로 했다. 각 바퀴를 구동시키는 모터는 정확한 엔코더가 내장되어 있고 작은 사이즈의 Faulhaber 사의 2224-012SR를 사용하였다.

EZ-X5가 다양한 직렬 통신 포트를 제공함에도 불구하고

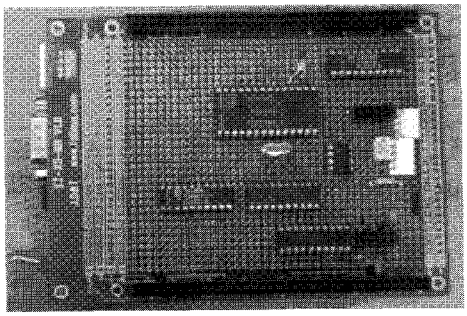


그림 4. CAN통신을 위한 인터페이스 보드.
Fig. 4. CAN interface board.

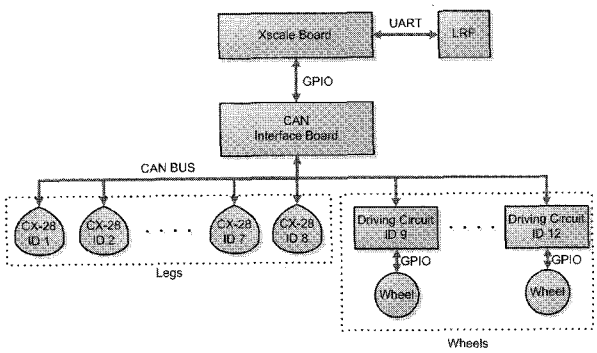


그림 5. 하이브리드 로봇의 하드웨어 구조.
Fig. 5. Hardware architecture of hybrid robot.

CAN 통신을 위한 포트는 제공하고 있지 않다. 따라서, CAN 통신을 위해 추가적인 인터페이스가 필요하다. CAN의 최대 통신 속도인 1 Mbps를 사용하기 위해 EZ-X5의 GPIO를 사용하여 CAN 통신 에뮬레이션을 하기 위하여 CAN interface board를 설계 구현하였다.

결국 전체 하드웨어 구조는 그림 5와 같게 된다.

III. CrocoHybrid 의 소프트웨어 플랫폼

1. 로봇을 위한 실시간 시스템 소프트웨어

다리 이동 로봇의 각 다리가 정확히 원하는 경로를 이동하는 것은 물론 다양한 환경에 대해서도 안정적인 보행이 가능하기 위해선 기계적인 구조 향상도 중요하지만, 실시간 제어 시스템을 통해 다양한 프로그램간의 데이터 흐름을 명확히 정의하고 제한된 시스템 자원을 효율적으로 분배하는 실시간 소프트웨어 구조의 설계 및 구현이 또한 필요하다. 경로 설정, 역기구학 등을 계산하는데, 시간 제약조건(timing constraint)을 만족시키지 못하면, 로봇의 다리 움직임이 부자연스럽게 되고 각 관절의 위치제어 오차가 계속 누적되어 결국에는 목적지에 도달하지 못하게 되고 로봇 전체 움직임의 불안정은 물론, 로봇의 목적지에서 이탈하는 현상을 보이게 될 것이다. 따라서, 본 논문의 하이브리드 로봇에게는 각 다리의 모터 제어, 센싱, 위치 추정, 경로 계획 등과 같은 실시간 태스크가 필요하다. 태스크들은 각자의 주기와 우선순위, 수행시간, 계산량을 가지고 있다. 이런 다양한 태스크들 중 로봇의 안정도에 큰 영향을 미치는 센싱과 모터에 대한 것들은 경성 실시간성(hard real time)을 만족시키도록 해야 하기 때문에 효율적으로 관리하고 실시간 제어를 위한 소프트웨

어 구조 또한 필요하게 된다는 것이다.

이러한 실시간성을 제공하기 위한 다양한 OS(Operating System)들이 존재한다. 그 중에서도 경성 실시간성을 제공하며 Linux를 베이스로 구동되는 것은 RT-Linux와 RTAI(Real Time Application Interface)가 있다. RT-Linux와 RTAI는 기본적으로 서로 비슷하다. 하지만, RTAI는 새로운 커널 버전에 대한 패치가 빠르고 많은 사용자들이 각자가 만든 실시간 응용 프로그램들을 테스트하고 있어서 충분한 안정성을 보장하기에 본 논문의 OS로 선정하였다. 이 RTAI의 장점은 Linux와 같이 open source이고 고속 실시간 테스크 동작이 가능하고, 매우 낮은 jitter를 가지는데 있다. 또한, Linux Real-Time(LXRT)라는 구조를 선택하였기 때문에, 실시간 API를 커널 영역이 아닌 사용자 영역에서도 그대로 사용이 가능하도록 하여 작업 환경이 유연하고 디버깅이 용이하다는 특징을 가진다. 또한, 다양한 프로세서를 호환하고 POSIX를 지원하며, IPC(InterProcess Communication), 동적 메모리 할당, 실시간 태스크의 one shot mode 및 periodic mode를 제공한다[13,14]

본 논문에서는 리눅스 커널 2.4.1과 RTAI 3.1를 사용하였는데, RTAI는 그 특징상 커널 모듈로 램디스크에 추가하여 시스템 전원을 켰을 때 자동으로 커널에 적재하도록 하여 실시간성을 활성화 시켰다. 또한, 센서로부터 받는 데이터의 실시간성을 보장하기 위해 RS-232의 실시간 직렬 통신 장치 드라이버인 SPDRV(Serial Port Driver)를 적용시켰다. 또한, 소속된 연구실에서 개발한 RT-CAN(Real-Time CAN) 드라이버를 적용하여 EZ-X5의 CAN 통신을 위한 구동 동작이 실시간성을 가지도록 하였다.

2. 실시간 프로세스 구조

그림 6은 본 논문에서 하이브리드 로봇을 위한 계층적인 소프트웨어 제어 구조를 나타낸다.

본 논문의 하이브리드 로봇의 특징 상 바퀴와 다리 제어를 각각 할 수 있도록 해야 된다. 따라서, 모드(mode)의 선택으로 이를 조절 하도록 하였다. 바퀴와 다리를 제어하기 위해선 진행 궤적 생성, 로봇 모니터링, 피드백 정보 처리, 명령 처리, 제어명령 처리를 위한 요소들이 필요하다. 바퀴를 이용한 평지에서의 동작은 간단히 목표점이 선정이 되면 필요한

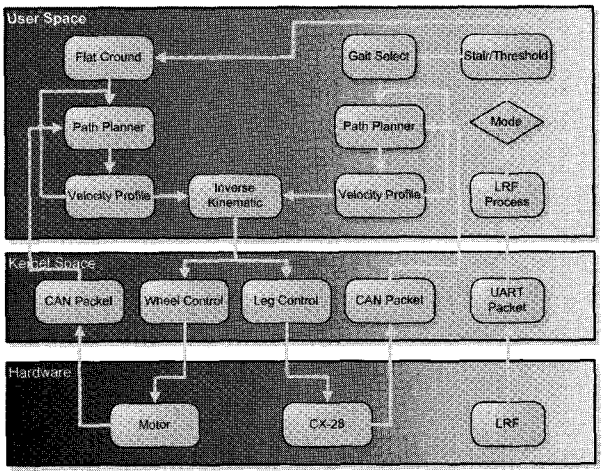


그림 6. 하이브리드 로봇을 위한 실시간 소프트웨어 구조.
Fig. 6. Real-time software architecture for hybrid robot.

궤적을 행성하고 이를 로봇 바퀴의 역기구학을 통해 바퀴를 속도 제어하게 된다. 이를 통해 평지에서 로봇을 원하는 위치로 이동시킬 수 있다. 다리를 이용하는 경우는 다음과 같다. 먼저, 계단이나 문턱을 넘어가기 위해선 계단과 문 턱의 폭과 높이를 알아야 한다. 따라서, LRF(Laser Range Finder)로 계단과 문턱의 폭과 높이 측정하여 이 정보를 이용하여 필요한 궤적을 생성해 내도록 하였다. 결국, 로봇이 하나의 동작을 수행하기 위해선 다음과 같은 절차를 거치게 된다. LRF를 이용하여 계단과 문턱의 높이와 폭을 측정하고 이를 이용하여 다리의 궤적을 생성한 다음, 사용자 영역에서 역기구학을 수행하여 각 다리의 관절 각도 값을 얻어낸다. 이 값들을 RT-FIFO에 저장하면 커널 영역에서 이 값을 가져다가 RT-CAN driver를 통해 각 다리에 전달한다. 이때 사용자 영역에서의 태스크들은 시간 동기를 위해서 thread로 작성하였다. 일단, RT-FIFO에 계산된 값들을 저장되기 시작하면 RT-FIFO 핸들러에 의하여 제어 태스크들이 비활성인 상태에서 벗어나 주기적으로 10ms마다 수행되기 시작한다. 물론, 이미 각 다리의 초기 위치를 설정하는 명령이 로봇 시스템이 시작되었을 때 one shot mode로 제일 먼저 수행되게끔 하였다. 이 주기적인 태스크 외에 LRF로부터 주기적으로 매 200ms마다 데이터를 받아들이고 이 데이터를 사용자 영역으로 RT-FIFO를 통해 넘겨주게 된다.

제어시스템과 각 다리 관절의 CX-28은 데이터 패킷을 주고 받으면서 CAN 통신을 하게 된다. 제어시스템에서 각 서보모듈(CX-28)로 전송하는 패킷은 명령 패킷(command packet)이고 이 패킷을 받은 서보모듈은 명령을 수행하고 그 결과를 다시 제어시스템으로 전송하는 패킷은 반환 패킷(status report packet)이다. 이 반환 패킷을 이용하여 에러 확인은 물론, 서보 모듈의 각종 정보(온도, 토크, 속도, 등) 피드백을 받을 수 있다. 각각의 서보모듈은 자신만의 고유한 ID를 가질 수 있어서 명령 패킷에 포함된 ID가 자신의 ID와 일치할 때만 그 명령을 수행하게 되며 단, 브로드캐스트(broadcast) ID 명령일 경우에는 ID에 상관없이 모든 모듈들이 수신하게 된다.

다리 제어(leg control) 태스크는 사용자 영역에서 생성한 궤적에 따른 각 관절의 각도 값을 계산하여 넘겨준 데이터를 명령 패킷의 프로토콜에 맞게 변환하여 주기적으로 매 10ms마다 전송하는 역할을 한다. 총 8개의 서보모듈에 명령을 전송한다면, 명령 패킷이 108비트에 CAN 통신의 최대 속도인 1Mbps로 전송하여도 첫번째 서보모듈이 명령을 받는 시간과 마지막 서보모듈이 명령을 받은 시간의 차이는 약 0.85ms가 된다. 이 차이는 다리가 움직이기 시작하는 시간의 차이로 이 차이가 시스템이 구동되기 시작하였을 때는 작지만, 시간이 지날수록 이 차이가 누적이 되어 전체 다리 동작이 엉망이 된다. 즉, 다리들이 제각각 움직이게 된다. 따라서, 각 서보모듈로 전송된 명령이 동시에 수행되도록 하는 조치가 필요한데 이미 서보모듈의 프로토콜에서 명령 패킷 저장 명령과 action 명령이 정의가 되어 있어서 쉽게 해결 할 수가 있었다. 각 서보모듈에게 명령을 내리되 당장 수행하는 것이 아니라 저장시키도록 하고 모든 서보모듈에 명령 패킷이 전달, 저장된 후 저장된 명령을 수행하라는 action 명령 패킷을

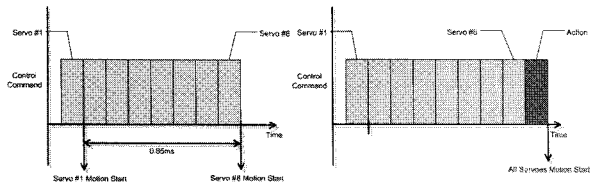


그림 7. 다리 제어의 타이밍 다이어그램.
Fig. 7. Timing diagram for control task.

브로드캐스트로 전달하면 되는 것이다. 다시 말해, 그림 7처럼 관절의 각도 값들을 각 서보모듈에 전달하는 8개의 명령과 전달된 후에 브로드캐스트 명령을 하나로 묶어서 제어 주기마다 전송하는 것이다. 이러므로, 8개의 서보모듈은 각 제어주기마다 동시에 동작할 수 있게 된다.

IV. CrocoHybrid 의 걸음을 위한 경로 설정

1. CrocoHybrid의 악어 걸음새 개념

악어는 그 생태학적으로 걸음걸이가 특이한 파충류이다. 몸 전체길이에 비해 엄청 짧은 다리는 걸음걸이에 별 도움이 되어 보이지 않는다. 하지만, 이런 걸음 동작으로 악어는 한 번도 넘어진 적이 없다. 넘어질 수 없는 걸음새를 하고 있는 것이다. 악어의 배부분이 거의 땅에 닿을 듯이 하는 걸음걸이여서 동물학자들은 이를 배기어가기(belly crawl)이라 명명하였다[15].

악어의 이 같은 걸음새를 흉내 내어 하이브리드 로봇의 다리를 움직이고자 한다. 즉, 다리를 이용하여 걸음을 걸을 때, 로봇의 몸체가 땅에서 최대한 멀어지지 않게 하는 것이다. 그리고, 다리를 움직일 때 최대한 시간을 단축하는 것이 시스템 자원을 적게 사용하게 될 것이고 또한 목적지에 도달하는데 소비하는 시간도 줄어들게 되는 장점을 얻을 수 있을 것이다. 따라서, 본 논문의 하이브리드 로봇은 앞다리와 뒷다리가 동시에 움직이도록 하였으므로 전체 다리가 움직이는데 소비하는 시간은 하나의 다리가 소비하는 시간과 같다. 물론, 이렇게 되면 움직임의 다양성은 확보할 수는 없으나, 본 논문의 목적인 계단을 올라가고 문턱을 넘어가는 데에는 앞다리와 뒷다리가 동시에 움직여도 가능하다. 이에 따라, 동시에 움직이도록 하면 전체적인 움직임은 그림 9처럼 된다.

그림 9처럼 본 논문의 하이브리드 로봇은 다리와 바퀴가 완전히 독립적으로 동작하는데, 다리를 움직일 때 로봇의 바퀴를 디딤대로 사용하는 특성' 을 가지도록 하였다. 이는 앞에서 언급한 악어의 걸음새를 흉내 내고자 한 개념에 평지가 아닌 계단을 올라가고 문턱을 넘어가기 위한 특징을 첨가한 결과이다. 즉, 바퀴에 대한 제어는 멈추거나, 필요 시 다시 제어가 가능한 상태로 하고 다리만을 이용하는데 계속해서 로봇의 몸체를 지면에서 띄운 채로 이동하는 것이 아니라 다리의 한번 스윙 동작으로 한번씩 몸체를 이동해서 지면에 고정시키고 다시 다리를 움직여 몸체를 이동시키는 동작을 목적지에 도달할 때까지 연속적으로 수행한다는 것이다.

이렇게 앞다리와 뒷다리를 동시에 움직이도록 하고 로봇의 바퀴를 디딤대로 사용하게 되면, 여러 가지 장점을 가지게 된다. 첫째, 일반적인 4족 로봇에 비해 로봇의 몸체가 지면에서 떨어져 있는 시간이 적고 지면에서 떨어져서 몸체가

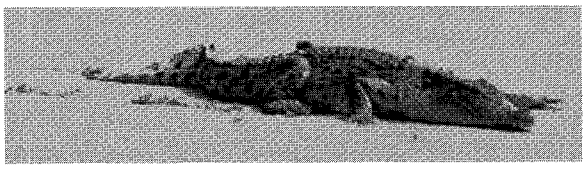


그림 8. 악어의 배기어가기.
Fig. 8. Crocodile belly crawl.

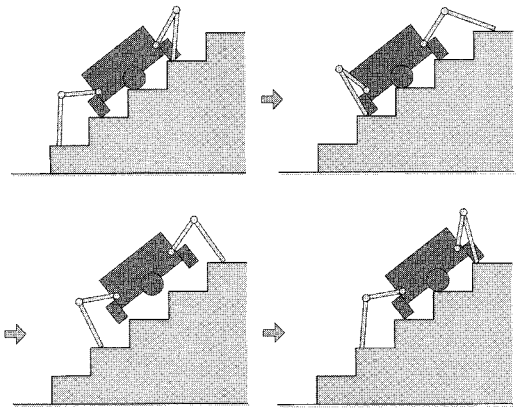


그림 9. 하이브리드 로봇의 단계별 다리 동작.
Fig. 9. Gait steps of hybrid robot.

이동할 때 조차도 지면에서 멀리 떨어지지 않기 때문에 안정도를 확보하게 된다. 둘째, 4족 로봇들의 다리는 적어도 3자유도를 가져야만 전방향으로 다리로 움직일 수 있는 반면, 본 하이브리드 로봇은 다리를 이용할 때는 계단을 오르고 문턱을 넘어갈 때로 한정하였으므로 최소의 DOF를 가질 수 있다. 셋째, 4개의 다리를 각각 움직이면서 로봇의 안정도까지 고려하는 것보다 적은 계산량을 가진다. 물론, 단점으로 최소한의 자유도를 가지게 하였으므로 움직임에 대한 다양한 선택을 할 수 없어서 계단을 오르거나, 문턱을 넘어 갈 때 로봇의 방향이 틀어지게 된다면 원래 방향으로 몸체를 돌리기 어렵다는 점이 있다. 이 논점은 본 논문의 목적을 달성하는데 큰 어려움이 아니고 추후 자유도를 추가하므로써 개선 할 수 있다[16].

게다가, 하이브리드 로봇의 다리가 계단을 올라가고 문턱을 넘어가는 목적으로만 사용될 수 있는 것은 아니다. 평지에서 바퀴 주행을 진행하다가 바퀴가 모래구덩이나, 진흙 같은 바퀴만으로 더 이상 이동이 불가능할 시에도 바퀴와 다리를 동시에 사용하여 이동 불가 상태를 벗어 날 수도 있다.

2. 계단과 문턱을 오르기 위한 궤적 생성 알고리즘

4족 로봇이 계단을 오르고 문턱을 넘어가기 위해서는 4다리에게 움직일 경로를 생성해 주어야 한다[17,18]. 본 논문에서는 앞다리 2개와 뒷다리 2개가 동시에 움직이므로 전체적으로 한번의 움직임을 위해서 앞다리와 뒷다리 각 2개의 경로 생성만 필요하게 된다. 궤적을 생성하기 전에 각 다리가 동작범위(work space)에서 벗어난 곳을 진행하지 않도록 해야 한다는 점을 명심해야 한다. 또한, 하이브리드 로봇의 동작 특성상 앞다리와 뒷다리가 동시에 지면에서 떨어지고 동시에 지면에 도달해야 하므로 앞다리와 뒷다리가 목적지에 도착하는 시간이 같도록 하는 속도 프로파일을 생성해야 한다

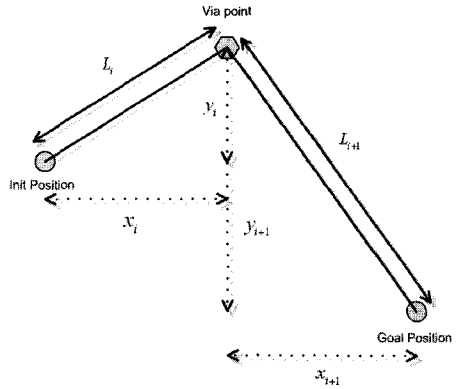


그림 10. 요구 궤도.
Fig. 10. Desired trajectory.

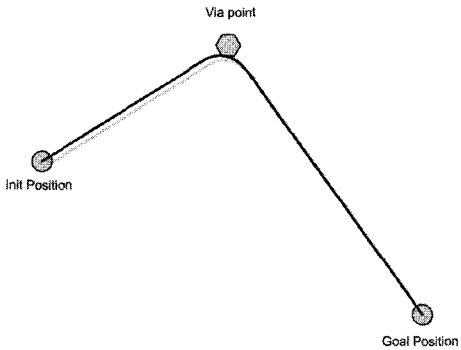


그림 11. 수정된 부드러운 궤도.
Fig. 11. Generated smooth trajectory.

는 점 또한 고려해야 한다. 하지만, 본 논문의 하이브리드 로봇은 굳이 안정도를 고려할 필요가 없는 특징으로 제어가 간결해 고려해야 할 사항이 줄었다. 따라서, 본 논문에서는 각 다리의 끝(end-effecter)이 움직이는 경로 궤적을 속도 프로파일을 바탕으로 생성하게끔 하여 제어 주기마다 다리의 끝의 위치를 계산하면 되었다. 전체적인 과정은 다음과 같다. 움직여야 할 궤적을 생성할 때 반드시 지나야 할 경유점(via point)를 생성하고 그 포인트들을 직선으로 연결하여 다리가 움직여야 할 총 이동거리를 산출해 낸다. 이 총 이동거리가 제어 주기에 따른 속도프로 파일의 적분 값과 같으므로 이를 이용하여 속도프로파일을 생성하고 각 제어 주기에 따라 속도프로파일을 적분하여 다리의 이동 좌표를 계산하여 제어 주기마다 각 관절에 각도 정보를 전달하여 다리를 구동하였다. 다만, 각 경유점 사이를 직선으로 움직이게 되어도 다리에 큰 무리는 없지만, 동작이 자연스럽지 못하고 만약, 움직이던 속도를 유지하면서 경유점에서 급격히 방향을 바꾸게 되면 전체 가속도가 커져서 자칫 CX-28의 한계 가속도를 넘는 경우도 생길 수 있어 결국 시스템을 다운시킬 수도 있다. 따라서, 각 경유점 부근에서는 속도를 줄여서 가속도의 절대값을 일정하게 유지시켜 한계 가속도를 넘지 않는 속도 프로파일을 생성하였다. 단, 이 속도 프로파일은 전체 속도에 대한 값이므로 각 x,y,z 베이시스 축으로 반영하여 각 축의 속도를 구해내야만 제어 주기에 따른 좌표공간에서의 발 끝의 위치를 계산해 낼 수 있다. 이렇게 생성된 궤적으로부터 실제 관절에서의 각은 역기구학을 수행하여 얻어내게 된다.

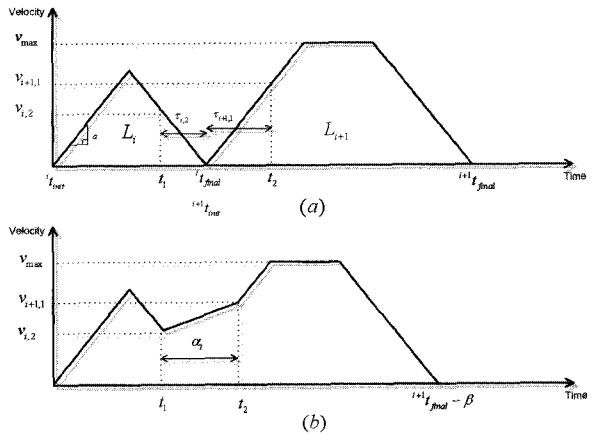


그림 12. (a) 요구 속도프로파일 (b) 생성된 연속 속도프로파일.
Fig. 12. (a) Desired velocity profile (b) Generated continuous velocity profile.

또한, 다리를 앞으로 내디딜 때와 뒤로 당길 때의 총 이동거리는 다르게 된다. 이는 다리를 뒤로 당겨서 움직이는 것은 다리가 움직이는 것이 아니라 로봇이 움직이는 것이 되므로 뒤로 당기는 총 거리는 로봇이 얼마나 이동하여야 하는지에 따라 결정된다. 이 뒤로 당기는 거리는 로봇의 중심좌표를 어느 위치까지 이동시키느냐, 다리를 얼마나 앞으로 내디뎠느냐, 현재 로봇의 기울기에 따라 결정이 되므로 현재 로봇의 위치 정보를 이용하여 계산하게 하였다.

정해진 경유점이 그림 10과 같을 때(z축 생략), 총 이동거리에 따라 속도프로파일을 생성할 시 가속도(a)와 최대 속도($v_{\max}$)를 정해 놓으면 경유점 사이의 직선 거리 양(L_i, L_{i+1})에 따라 그림 12(a)처럼 삼각형태와 사다리꼴 형태로 만들어지게 된다.

이렇게 만들어진 속도프로파일은 경유점마다 속도가 0이 된다. 이는 부자연스러운 동작을 유발하므로 속도프로파일에서 0이 되는 부분이 없어야 한다. 따라서, i번째 속도프로파일과 $i+1$번째 속도프로파일의 사이를 일정 가속도로 연결해 주기 위해 각각 $\tau_{i,2}$(i번째 감속구간)과 $\tau_{i+1,1}$($i+1$번째 가속구간)를 정해 놓는다. 이 τ 값은 설계자가 선정하는 값으로 i번째 속도프로파일의 가속 또는 감속시간이 ${}^{a}t_i$라면 τ의 범위는 $0 \le \tau \le {}^{a}t_i$에서 값을 선정해야 한다. 다만, τ가 0에 가까울수록 경유점 부근까지 움직이고 ${}^{a}t_i$에 가까울수록 경유점에서 멀어진 채 경유점을 지나게 될 것이다. 이 경유점들은 path planning시 CrocoHybrid의 workspace 내부에 존재해야 하고 또한, 외부 장애물과의 충돌을 하지 않아야 한다는 고려사항을 통해서 결정된 것이기에 제시된 범위 값 내의 어느 τ 값을 취해도 된다. 다만, 연속된 속도프로파일 생성의 근본 목적에 더 부합되는 τ 값은 선정한 최대 가속도의 값보다 되도록 작아야 한다는 것이기에 τ를 ${}^{a}t_i/2$로 선정한다면 가장 작은 가속도 값을 가지게 될 것이다. 따라서, 본 논문에서는 τ를 ${}^{a}t_i/2$로 선정하였다. 추가되는 τ 시간에 의해 0이 되는 속도가 없어지고 늘어난 속도프로파일의 면적은 곧 이동거리의 증가로 이어지므로 이를 보상하는 작업이 필

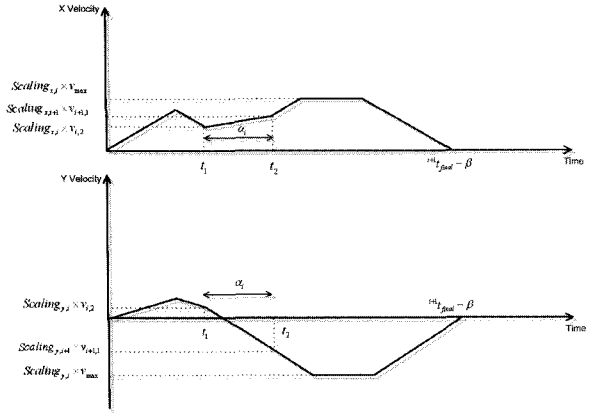


그림 13. 각 축으로 변환된 속도프로파일.
Fig. 13. Converted velocity profile in each axis.

요하다. 결과적으로 전체 n개의 속도프로파일은 그림 12(b)처럼 1개로 변환되면서 총 소요시간은 β 만큼 줄어들게 된다.

$$\beta = \sum_{1}^{n-1} \tau_{i+1,1} + \sum_{1}^{n-1} \tau_{i,2} - \sum_{1}^{n-1} \alpha_i \tag{15}$$

$$\alpha_i = \frac{v_{i,2}\tau_{i,2} + v_{i+1,1}\tau_{i+1,1}}{v_{i,2} + v_{i+1,1}} \tag{16}$$

이렇게 완성된 전체 속도프로파일을 다시 각 축의 이동거리에 따라 스케일링하여 적용하면 최종적인 각 축의 속도프로파일을 구하게 된다. 이 스케일링은 각 축의 방향과 크기를 반영하므로 손쉽게 각 축의 방향까지도 얻게 된다.

$$Scaling_{x,i} = \frac{x_i}{L_i},\ Scaling_{y,i} = \frac{y_i}{L_i},\ Scaling_{z,i} = \frac{z_i}{L_i} \tag{17}$$

여기서 $L_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$ 로 i 번째 총 이동거리 x_i, y_i, z_i 는 각축의 이동거리를 뜻한다. 이렇게 해서 구해진 각 축의 속도프로파일은 그림 13과 같아 진다.

이렇게 구해진 각 축의 속도프로파일을 각각 제어주기에 따른 적분을 통해 다리가 움직여야 할 좌표를 구해내고 이 좌표들을 역기구학을 통해 각 관절 값을 구하게 된다.

정리하면, 계단의 높이와 폭이 주어지면 이를 바탕으로 한 단계씩의 경로와 속도 프로파일을 생성하여 제어주기에 따른 위치를 구하고 이를 역기구학으로 각 관절의 각도 값을 CX-28로 전송하는 일련의 방법을 반복하여 수행한다. 이렇게 함으로써 계단을 오르고 문턱을 넘어갈 수 있게 된다. 위에서 보다시피 로봇의 안정도가 확보되어 있어서 로봇의 무게 중심이 어디에 있는지에 대한 계산이 없어도 되어 단지 다리의 이동 경로와 이에 대한 역기구학만 계산하면 로봇을 원하는 위치로 이동시킬 수 있다.

V. 시스템 구현 및 실험

1. CrocoHybrid의 구현 및 시뮬레이션

실험에 앞서 계단을 오르는데 또, 문턱을 넘어가는데 CrocoHybrid의 다리가 제대로 경로를 따라가는지, 각 관절의 값들이 제대로 나오는지를 시뮬레이션을 통해 확인해 보았다. MATLAB에서 프로그램을 작성하여 이에 대한 성능을 확

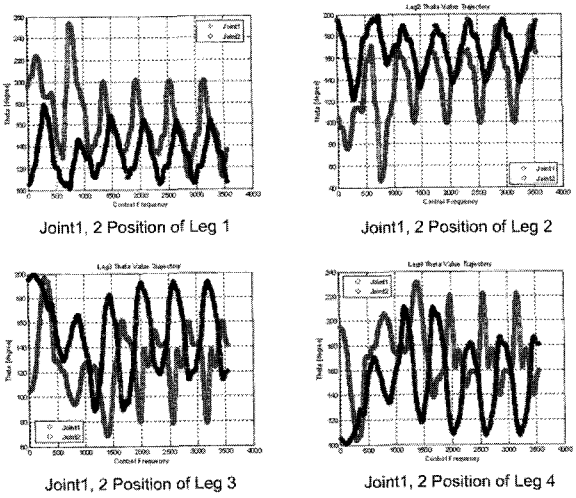


그림 14. 계단 오를 때 다리 별 각 관절의 각도 값.
Fig. 14. Joint values os each leg at climbing stair.

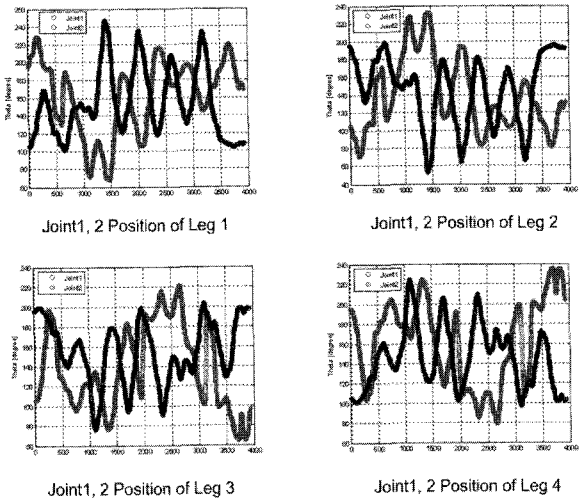


그림 15. 문턱을 넘어갈 때 다리 별 각 관절의 각도 값.
Fig. 15. Joint values of each leg at going over threshold.

인하였다. 그림 14와 15는 계단을 오를 때와 문턱을 넘어갈 때 각 다리의 관절 각도 값을 표시한 것이다. 그림 14와 15를 보다시피 leg1과 leg2, leg3과 leg4의 각도 값은 정확히 상하 대칭을 이루고 있다는 것을 알 수 있는데, 이는 로봇을 해석할 때, 각 다리의 관절 축을 통일시킨 결과이다.

본 논문의 하이브리드 동작 특성상 로봇의 몸체가 연속적으로 움직이는 것이 아니고 한번에 한번씩 몸을 움직이는 불연속적인 동작을 계속하므로 이 불연속적인 동작에 의해 각 다리의 관절 값까지 불연속이 되면 동작과 동작 사이에서의 각도가 급격히 움직이게 되어 부자연스럽고 불안정한 동작이 될 수 있다. 이를 방지하기 위해 각 동작의 결과로 변화하는 로봇의 수평 기울기와 로봇의 중심 좌표를 정확히 정의하면 그림 14와 15처럼 관절의 각도 값이 연속성을 띄게 할 수 있음을 확인하였다. 단, 간혹 각도 값이 연속이 안되었던 부분은 다리의 미끄러짐 현상으로 예상한 로봇 중심의 위치가 벗어났을 때 발생하였다. 이는 예상치 못한 외란(disturbance)인데 각 CX-28은 입력 각도만큼 움직였으므로 피

드백 되는 각도는 정상으로 나오게 된다. 따라서, 이 불연속을 방지하기 위해선 미끄러짐을 완전히 방지해야 하지만, 현실적으로 불가능하고 단지 이를 최소화하는 방법만이 있을 뿐이다.

물론, 미끄러짐은 마찰력의 급감을 뜻하므로 힘 제어를 사용한다면 이러한 문제는 보상할 수 있을 것이다. 따라서, 추후 힘 제어를 통해 적응 제어가 되어야 할 것이다.

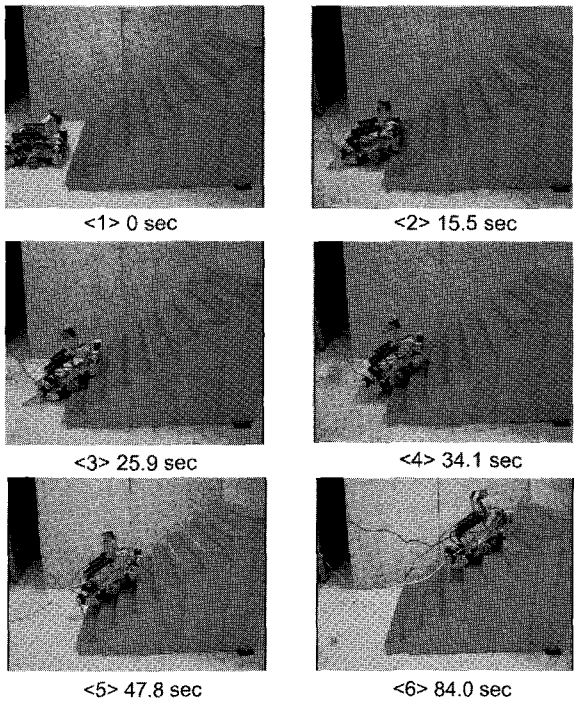


그림 16. 계단 오르기 실험.
Fig. 16. Experiment for climbing stair.

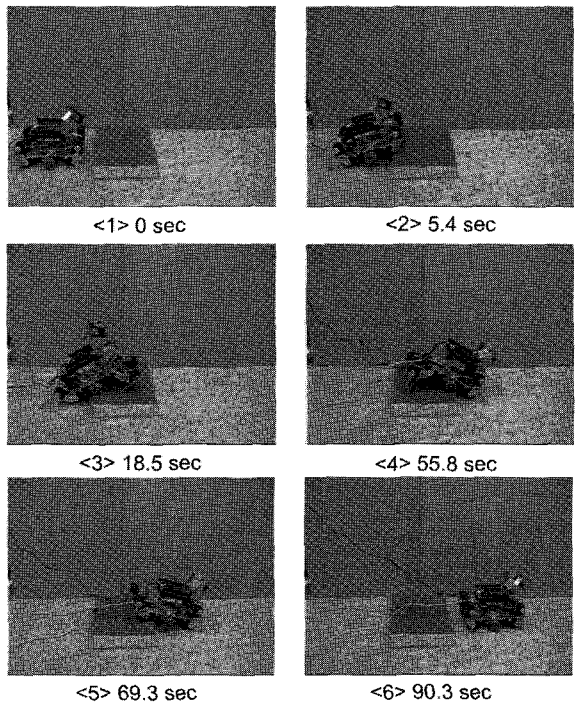


그림 17. 문턱 넘어가기 실험.
Fig. 17. Experiment for going over threshold.

2. 실험

시뮬레이션을 통해 각 관절의 각도 값이 연속적으로 제어가 가능하다는 것을 확인하였으므로 실험에 임하였다. 실험은 계단을 구성시켜 이를 제대로 올라가는 지와 간이로 문턱을 만들어 이를 넘어 갈 수 있는지를 확인하였다. 앞의 그림 16과 17의 각 그림의 아래의 숫자는 동작을 시작한 후의 시간을 나타낸다.

VI. 결론

평지에서 고속 주행이 가능한 바퀴 주행 로봇의 장점을 그대로 살리고 계단 오르기와 문턱 넘어가기란 단점을 극복하기 위해 바퀴와 몸체를 지지대로 사용하는 기능을 가진 CrocoHybrid를 설계, 구현하였다. 이와 같은 구조적인 특성으로 계단이나 문턱 뿐만 아니라, 모래와 같은 지형에서도 바퀴의 한계를 극복할 수 있는 잠재력 또한 가지게 되었다. 즉, 모래와 같이 바퀴의 미끄러짐이 심한 곳에서 장착된 다리들을 사용한다면, 바퀴가 모래에 묻히더라도 쉽게 빠져나올 수 있는 가능성을 가지게 된 것이다. 이 CrocoHybrid를 통해 평지에 특화되어 있는 전방향성 바퀴의 구조와 특성을 그대로 가지면서도 바퀴 주행 로봇의 가장 큰 어려움인 계단 오르기와 문턱을 넘어가기를 4개의 다리에 최소한의 자유도만 가져도 극복할 수 있다는 것을 확인하였다. 바퀴를 적극적으로 활용하기 위해 선택된 악어의 걸음걸이 흉내내기가 안정적이고 간편한 제어를 제공할 수 있다는 것 또한 확인하였다.

이러한 각각의 과정을 시뮬레이션을 통해서 먼저 그 가능성을 확인한 후 구현하였고 실험을 통해 원하는 목적을 달성할 수 있음을 확인하였다. 게다가 로봇의 동작을 실시간 제어 구조를 통해 구현하여 시간제약 조건을 만족시킬 수 있었고 실시간 성능을 갖춘 모듈화된 소프트웨어를 이용하여 추후 추가되는 센서나 사용자가 요구하는 다양한 기능을 제공하는 것이 가능하게 되었다.

향후 연구과제로서 다리와 바퀴의 미끄러짐을 통해 나타난 오류를 보상하기 위해 센서를 통한 주변위치 인식과 힘 제어가 요구 되고, 이를 통해 계단 오르기와 문턱 넘어가기에 소요되는 시간을 줄이는 연구도 필요하다.

참고문헌

[1] Acroname Robotics, http://www.acroname.com
[2] Robomotio, http://www.robomotio.com
[3] K.-U. Scholl, B.Gassmann, and K Berns, "Locomotion of LAURON III in rough terrain," Advanced Intelligent Mechatronics, *Proc. of IEEE/ASME*, Como, Italy, vol. 2, pp. 959-964, 2001.
[4] Yoneda K. Hirose, S. and H. Tsukagoshi, "TITAN VII: Quadruped walking and manipulating robot on a steep slope," Robotics and Automation, *Proc. of IEEE*, Albuquerque, NM, USA, vol. 1, pp. 494-500, 1997.
[5] Joel Chestnutt, Manfred Lau, German Cheung, James Kuffner, Jessica Hodgins, and Takeo Kanade "Footstep planning for the honda ASIMO humanoid," *Proc. of IEEE International Conference on Robotics and Automation*, Barcelona, Spain, pp. 629-634, 2005.

[6] I.-W. Park, J.-Y. Kim, J. H. Lee, and J.-H. Oh "Mechanical design of humanoid robot platform KHR-3(KAIST Humanoid Robot-3: HUBO)," *Proc. of IEEE-RAS Internatinal Conference on Humanoid Robots*, Tsukuba, Japan, pp. 321-326, 2005.

[7] NASA Sojourner Rover http://mpfwww.jpl.nasa.gov/rover/sojourner.html

[8] F. Plumet Ch. Grand, F. Ben Amar and Ph. Bidaud, "Stability control of a wheel-leg mini-rover," in *5th International Conference on Climbing and Walking Robots*, Paris, France, pp. 323-331, 2002.

[9] F. Plumet Ch. Grand, F. Ben Amar and Ph. Bidaud, "Decoupled control of posture and trajectory of the hybrid wheel-legged robot Hylos," *Robotics and Automation, Proc. of IEEE-ICRA International Conference*, New Orleans, USA, vol. 5, pp. 5111-5116, 2004.

[10] T. Arai A Shimiza H. Adachi, N. Koyachi and Y Nogami, "Mechanism and control of a leg-wheel hybrid mobile robot," *IROS Proc. of IEEE/RSJ International Conference*, Kyongju, South Korea, vol. 3, pp. 1792-1797, 1999.

[11] P. J. McKerrow, *Introduction to Robotics*, AddisonWesley, 1991.

[12] ROBOTIS, http://www.robotis.com

[13] DIAPM RTAI Programming Guide 1.0, 2000 http://www.rtai.org

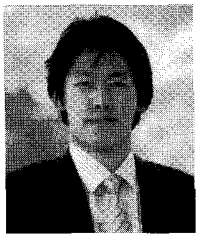
[14] 김성진, 김병국, "전동 휠제어의 자율 주행을 위한 실시간 소프트웨어 구조의 개발" 제어 · 자동화 · 시스템 공학 논문지, vol. 10, no. 10, pp. 940-946, 2004.

[15] Crocodilian Biology Database, http://www.flmnh.ufl.edu/cnhc/cbd-gb4.htm

[16] J. Pan and J. Cheng, "Study of quadruped walking robot climbing and walking down slope," *Proc. IROS IEEE/RSJ International Workshop*, Osaka, Japan, vol. 3, pp. 1531–1534, 1991.

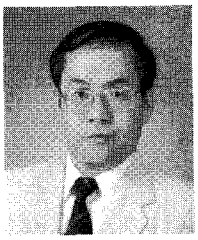
[17] B. Lantos I. Harmati and S. Payandeh, "Stratified motion planning concept in stair-like environment," RoMoCo *Proc. of the Fourth International Workshop*, Puszczykowo, Poland, pp. 259-264, 2004.

[18] Lining Sun Bo Huang and Yufeng Luo, "Statically balanced stair climbing gait research for a hybrid quadruped robot," *Mechatronics and Automation of IEEE International Conference*, vol. 4, pp. 2067-2071, 2005.



김 진 백

1979년 2월 6일생. 2005년 한양대학교 전기 · 전자공학과 졸업. 2005년 한국과학기술원 전기전자전공 석사. 2007년~현재 동 대학원 전기전자전공 박사과정 재학중. 관심분야는 이동 로봇, 제어 소프트웨어 구조, 임베디드 제어.



김 병 국

1952년 10월 5일생. 1975년 서울대학교 전자공학과 졸업. 1975년 한국과학기술원 전기 및 전자 공학과 석사. 1981년 동 대학원 박사. 1981년~1986년 우진계기(주) 연구실장. 1982년~1984년 University of Michigan 방문연구. 1986년~현재 한국과학기술원 전기 및 전자공학과 교수. 관심분야는 실시간 시스템, 로보틱스, 임베디드 제어.