MSRDS를 이용한 모바일 로봇의 시뮬레이션 동기화 및 온라인 피드백 제어

*이윤섭, **최상호, ***이영삼, ***김진걸
*인하대학교 전기공학부 석사과정, **인하대학교 자동화공학과 박사과정, ***인하대학교 전기공학부 교수

Synchronization in Simulation and On-line Feedback Control for Mobile Robots Using The MSRDS

*Yoon-Sub Lee, **Sang-Ho Choi, *** YoungSam Lee, ***Jin-Geol Kim
*School of Electrical Engineering, Inha University, **Dept. of Industrial Automation Engineering, Inha University
***School of Electrical Engineering, Inha University

Abstract - 로봇을 개발함에 있어 많이 겪는 어려움 중 하나는 열악한 시뮬레이션 환경 및 소프트웨어 개발이다. Microsoft社의 MSRDS(Microsoft Robotics Developer Studio)는 3D시뮬레이션 환경과 로봇 구동을 위한 각종 표준화된 서비스들을 지원함으로써 이러한 문제점들을 해결할 수 있다.

본 논문에서는 원자력 발전소의 중 저준위 폐기물 저장창고 검사 로봇을 대상으로 MSRDS를 이용하여 3D 시뮬레이션 및 온라인 피드백 제어를 수행하였다. 로봇은 사륜으로 이루어져 있으며, 주행 환경 확인을 위한 카메라, 장애물 감시용 LRF(Laser Range Finer)센서, 주변 환경 감지를 위한 온도센서, 습도센서 및 MSRDS가 장착된 산업용 PC가 탑재되어 있다.

MSRDS의 제어신호는 이벤트로 생성되어 3D 시뮬레이션 렌더링 시간과 동기화되며, Brick 서비스를 통해 전송된다. 피드백은 10ms 주기로 LRF 센서, Motor 엔코더 값을 받아 3D 시뮬레이션에 반영된다. 본 논문에서는 MSRDS의 시뮬레이션 및 실제 로봇과의 동기화 방식을 제시하며 구동 실험으로 검증하였다.

1. 서 론

로봇을 개발함에 있어 직면하는 현실적 문제점은 특정 하드웨어에 의존적인 S/W 개발 작업이다. 로봇의 S/W 모듈들이 표준화되어 있지 않기 때문에 많은 서비스들을 직접 개발을 해야 한다. 이 때 많은 부분들이 코드의 재사용성 및 모듈화를 염두해 두고 코드를 구성하지 않기 때문에 로봇마다 특화된 S/W 모듈들로 이루어진다. 이로 인해 코드는 로봇 개발자 간에 공유 될 수 없으며, 로봇 기술 교류를 위한 커뮤니티 공간의 활성화를 방해하는 요인이 된다. 이로 인해 많은 시간과 인력을 투자해 서비스를 재개발하고 있다.

MSRDS는 다양한 로봇 하드웨어 플랫폼 상에서 실행되는 추상화된 S/W 컴포넌트들을 제공하여 로봇 애플리케이션을 쉽게 개발할 수 있도록 지원하는 툴이다. 객체지향적이며, 컴포넌트들 간의 느슨한 연결로 다양한 서비스를 손쉽게 사용할 수 있으며, 이로 인해 재사용성이 향상된다. Navigation service, Web Cam service, GPS service, TTS(Text To Speech) service 등 로봇 개발에 있어 필요한 다양한 서비스들을 제공한다. 성능 측면에서는 CCR(Concurrency and Coordination Runtime)을 통해 동시성 처리 및 병렬 처리를 지원하며, DSS(Decentralized Software Service)로 네트워크상에서의 서비스를 지원한다. 이러한 서비스들의 지원으로 로봇 개발자들은 로봇에 맞는 드라이브 및 일부 서비스 개발만으로도 로봇에 맞는 시스템을 구성할 수 있게 된다.

이러한 장점으로 인해 본 논문에서는 MSRDS를 이용하여 로봇 구동 및 제어 시스템을 구성하였으며, 원자력 발전소 중저준위 폐기물 저장창고에서 폐기물 드럼통의 이상 유무 및 방사선 검출을 위해 개발 중에 있는 로봇을 사용하였다. MSRDS의 시뮬레이션과 실제 로봇의 동기화 방식을 제시하며 온라인 피드백 제어 실험을 통해 검증하였다.

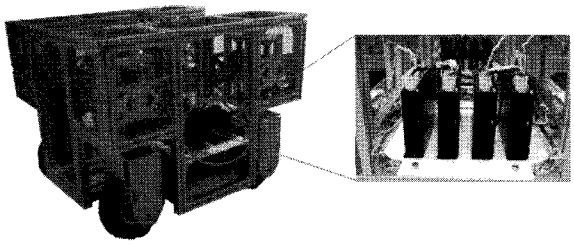
2. 본 론

2.1 로봇 소개

본 실험에 사용한 로봇은 원자력 발전소의 중저준위 폐기물을 담은 드럼통의 부식 및 방사능 누수의 검사를 수행할 목적으로 개발되었다. 현재 제작중인 프로토타입 로봇은 몸체를 알루미늄 프로파일로 구성하였으며, 바퀴 모듈은 10mm의 두께의 알루미늄으로 제작하였다. 바퀴모듈은 총 4개이며 같은 구조로 제작되어 있다. 모터는 Maxon-motor社의 150W 급의 RE40 모터와 기어비가 43:1인 감속기가 결합된 모터를 사용하였다. 이 모터는 8000RPM(Revolution Per Minute)의 속도를 낼 수 있으며 정격전압은 48V 이다. 엔코더는 512CPT(Counts Per Turn)의 분해능을 가지고 있으며 3-Channel의 출력을 가진다. 모터 드라이버로는 Maxon-motor社의 EPOS를 사용하였다.

각각의 바퀴 모듈에는 조향과 스피드를 위해 2개의 모터를 사용하였다. 모터 드라이버간의 데이터 전송은 CAN(Controller Area Network)으로 이루어지며, 외부 PC와는 RS-232 방식을 사용하였다. 조향범위는 180°로 제한하였으며, 축에 리밋스위치를 장착하여 동작범위에 제한을 두었다. 추진부에도 타이밍풀리와 타이밍벨트를 사용하였다.

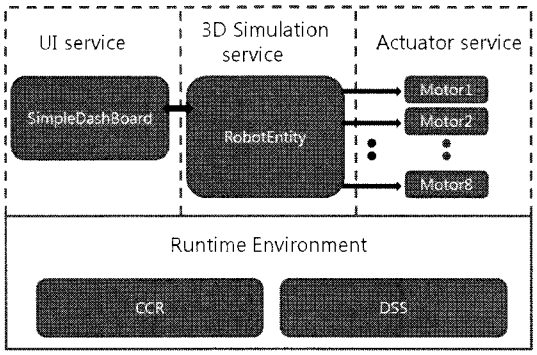
8개의 모터를 움직이기 위해 8개의 모터드라이버를 각각 장착하였다. 발열에 충분히 대비할 수 있도록, 그림 1과 같이 적절한 간격을 두어 배치하였다. 그림 1은 제어용 PC를 제거한 상태의 로봇 프레임이며, 우측의 확대 그림이 모터 드라이버이다.



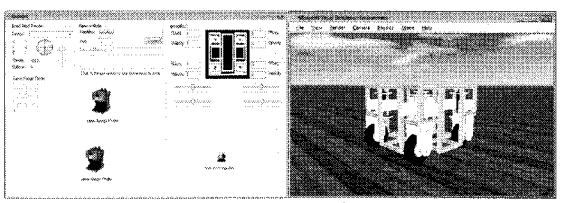
<그림 1> 로봇 프레임 및 드라이버

2.2 MSRDS 구성

시스템은 그림 2와 같이 크게 4가지 서비스로 구분할 수 있다. Runtime Environemt 서비스는 CCR, DSS로 구성되어 동시 처리 및 서비스 연결을 처리한다. UI service의 SimpleDashBoard는 사용자에게 로봇에 센서값 및 로봇의 현재 상태를 모니터링 할 수 있게 하며, 컨트롤 박스를 통해 로봇을 제어한다. 미리 정의된 서비스를 로봇 시스템에 맞게 수정하였으며, 그림 3의 좌측 그림이 수정된 서비스 UI이다. 3D Simulation service는 실제 동작 환경을 3D 모의 환경으로 구성해 놓았으며, 실제 로봇을 모델링하여 3D 로봇을 구현하였다. 정확한 시뮬레이션을 위한 물리 엔진이 지원되며, 로봇과 연동하지 3D 모의 환경에서도 제어 및 컨트롤 할 수 있다. Actuator service는 제어신호를 하드웨어에게 전송 및 수신 하는 서비스로, 데이터 흐름 제어를 한다. 연산시간을 단축하기 위해 Brick 서비스를 만들지 않고 Maxon-motor社의 EPOSAPI를 이용하여 직접 서비스를 구성하였다.



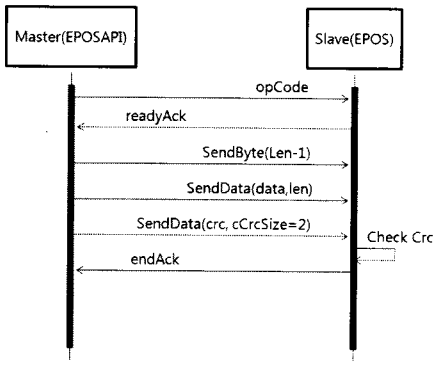
<그림 2> MSRDS 시스템 구조



<그림 3> SimpleDashBoard 및 3D simulation

3D 시뮬레이션 서비스 내의 RobotEntity는 그림 3의 우측에 위치한 3D

로봇을 나타낸다. RobotEntity의 핵심 함수중 하나는 Update() 함수이다. Update() 함수는 SystemCall에 의해 호출되며, 16ms 주기로 호출된다. 함수의 역할은 RobotEntity를 다시금 그려 주는 역할을 한다. 이에 Update가 호출될 때마다. 실제 로봇에게 제어신호를 보내면 쉽게 동기화시킬 수 있다. 하지만 EPOSAPI에서는 그림 4와 같이 연결지향성 전송을 하기 때문에 최대 500ms의 지연이 생길 수 있다. 실제 구현해 본 결과 EPOSAPI에서의 Data전송에 따른 지연시간으로 인해 시뮬레이션 화면에서 계단현상이 발생하는 문제점이 있다. 이러한 문제점을 해결하기 위해서는 시간 지연이 발생하는 EPOSAPI를 모듈화해서 동작시키되, 필요에 따라 Update()함수와 동기화 시켜주어야 한다.



<그림 4> EPOS 패킷 전송 다이어그램

시뮬레이션과 로봇의 움직임을 동기화하기 위해서는 로봇 모터의 엔코더 값을 피드백 받아야 한다. 피드백 받은 엔코더값이 전에 받은 엔코더값 보다 증가한 크기 값을 τ_e, 로봇 바퀴 1회전시의 엔코더 값을 τ_o, 로봇 바퀴의 반지름을 r, 엔코더 값을 받아 오기 전까지 걸린 시간은 T 라 할 때, 엔코더 값에 의한 실제 로봇 바퀴의 속도 ν는 다음과 같다.

$$\nu = \frac{\tau_e 2\pi r}{\tau_o T} \quad (1)$$

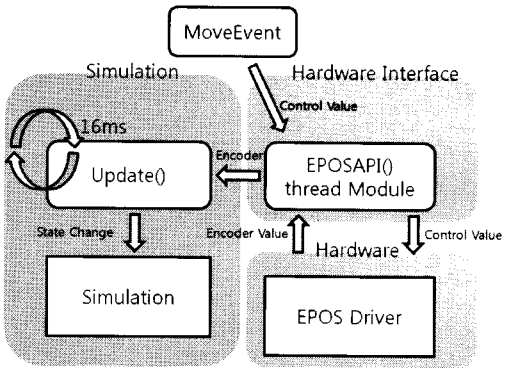
로봇 구동과 동기화된 시뮬레이션을 위해 Update 함수에서는 식(1)을 통해 속력으로 변환한 후 RobotEntity의 바퀴 속력 값으로 지정하면, 로봇이 실제 이동한 거리만큼 3D 로봇도 이동하게 된다.

2.3 실험

본 실험에서는 산업용 PC대신 노트북을 사용하여 시스템을 구성하였다. 실험용 노트북의 모델은 Intel CPU기반의 Windows OS를 탑재하여 실험용 노트북을 구성하였고, 로봇 프레임 위에 노트북을 얹어 실험을 진행하였다.

EPOSAPI를 Update()함수 내에서 사용하면 시뮬레이션 데이터 전송 중의 블록킹 현상으로 시뮬레이션이 원활히 진행되지 않는다. 따라서 본 논문에서는 그림 5와 같은 시스템을 구성하여 실험하였다.

MoveEvent가 발생하게 되면, EPOSAPI()에 데이터가 전송된다. EPOSAPI()는 별도의 스레드로 관리되며, MoveEvent에서 요청된 데이터를 EPOS Driver에게 전송한다. 또한 주기마다 엔코더값을 요청 수신 받아 Update() 와 공유된 엔코더 변수에 값을 할당해 놓는다. Update()는 엔코더 값을 피드백 받아 증가된 값 만 큼 속도로 변환하여 RobotEntity에 반영하여 실제 로봇과 동일한 움직임을 구현해 낸다. 스레드와 Update간의 동기화에는 ManualResetEvent을 사용하여 공유 자원에 대한 충돌을 방지 하였다.



<그림 5> 동기화 메커니즘 시스템

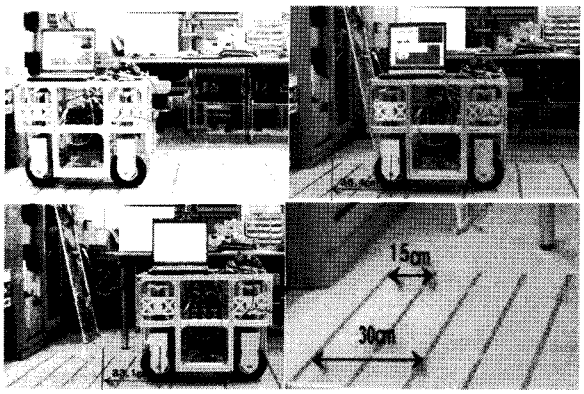
실험의 검증은 타이머를 이용하여 일정시간 동안 이동시킨 후 시뮬레이션에서의 로봇의 움직인 거리와 실제 로봇의 이동 거리를 측정 비교하였다. 그림 6에서와 같이 시작지점은 붉은색 라인에서 위치를 초기화 한 후 1.9초간 동일한 속도 값을 주어 일직선으로 이동시켜 주었다. 붉은색 라인과 파란색 라인간의 거리는 15cm이며, 청색 라인간의 거리는 30cm이다.

2.4 결과

Update를 통해 EPOSAPI를 호출하는 것에 비해 시뮬레이션 화면의 갱신 지연 현상의 발생 빈도수가 줄었다. 결과는 표 1, 그림 6과 같으며, 시뮬레이션과 실제 로봇의 오차율은 0.54%를 지연시간에 의한 오차율은 4.7% 이다. 시간의 오차는 EPOSAPI의 전송 지연의 여부 및 외부 인터럽트에 의한 시간 지연으로 발생한 것으로 보인다. 시뮬레이션과 실제이동거리의 오차는 OS 인터럽트에 의한 영향 및 로봇의 슬립 및 기타 외부 외란에 의한 것으로 소프트웨어 알고리즘 및 외부 보조 센서를 통해 해결될 수 있다.

<표 1> 지연시간

실험	실제이동거리	시뮬레이션 이동거리	시간(sec)
실험1	36.3cm	36.1cm	1.87
실험2	33.1cm	32.8cm	1.90
실험3	40.7cm	40.5cm	1.86
실험4	36.0cm	35.9cm	1.97



<그림 6> 로봇을 이용한 실험결과

3. 결 론

MSRDS를 이용하여 단시간에 서비스들을 조합하여 시뮬레이션 수행과 실제 구동실험을 행하였으며, 그 결과 무리 없이 제어가 가능 하다는 것을 확인할 수 있었다. 또한 하드웨어와 시뮬레이션의 동기화를 위한 시스템으로 그림 5와 같이 제시하여 그 성능을 실험 검증하였다.

추후 연구과제로서 실시간 운영체제인 WinCE에서의 동작 테스트 및 Brick서비스와 API를 이용한 직접 호출과의 차이점을 분석 및 검증을 수행할 예정이다.

후 기

이 논문은 2007년도 산업자원부의 지원에 의하여 기초전력공학공동연구소 (R-2007-2-059) 주관으로 수행된 과제임.

[참 고 문 헌]

[1] Padraig Osborne, Ryan McLellan, "A Force-Feedback Joystick for Control and Robotics Education", IEEE Control Systems Magazine, October 2004, pp.74-77

[2] Oscar Almeida, Johannes Helander, Henrik Nielsen, Nishith Khantal, "Connecting Sensors and Robots Through The Internet by Integrating MICROSOFT Robotics Studio and Embedded WEB Services", 2007

[3] Johannes helander, "Deeply Embedded XML Communication -Towards and Interoperable and Seamless World", 2005

[4] 김영준, "CCR 기초 강좌 통합본", 2007

[5] 김영준, "Microsoft Visual Simulation 사용자 가이드", 2007

[6] Young Joon Kim, "Beginning MSRS Simulation Programming Step by Step", 2007.

[7] Http://msdn.microsoft.com/ko-kr/robotics/default(en-us).aspx

[8] 차유성, "Robotics studio를 利用한 移動로봇의 制御 소프트웨어 모듈", 석사학위 논문, 성균관대학교, 2007