모듈형 퍼스널 로봇의 소프트웨어 아키텍처 개발

Development of Software Architecture for Modular Personal Robot

이 호 길*, 김 홍 석, 양 광 웅, 최 무 성, 원 대 희
(Ho-Gil Lee, Hong-Seok Kim, Kwang-Woong Yang, Moo-Sung Choi, and Dae-Heui Won)

Abstrac : In this paper, a standard robot design methodology is suggested and a software architecture for modular robot is introduced. The robot is modularized by several functions, and the module is produced according to a standard proposal. Each module requires standard interface for communicate in distributed environments. Software architecture was developed to support distributed component environment, and application development support tools are developed for user convenience. Many robot softwares are developed in a library form so that, they are being used widely robot application software development. Also a device driver was developed for the mostly used sensor and actuator. It is verified that the modular robot can be applied in various fields through guide, errand and guard scenario.

Keywords : module, software component, standard interface, personal robot, software architecture.

I. 서론

휴머노이드(humanoid) 로봇, 의료용 로봇, 복지용 로봇, 홈 서비스(home service) 로봇, 청소 로봇과 같은 다양한 종류의 로봇이 개발됨에 따라, 로봇에 사용되는 같은 기능의 부품이 호환되기 위해서는 모듈화와 표준화가 이루어져야 한다. 또한 개발자들은 비전(vision), 자율 이동(navigation), 사용자 인터페이스(user interface)와 같은 로봇 개발에 일반적으로 사용되는 소프트웨어 라이브러리를 필요로 하고 있으며, 이를 체계적으로 개발하고 공유할 수 있는 소프트웨어 아키텍처(architecture)를 필요로 하고 있다.

이와 같은 필요성에 의해 CMR (Component based Modularized Robot)과 FMR (Function based Modularized Robot)등이 개발되어왔다[1,2,3]. 이 연구의 목적은 모듈화· 표준화 된 방법으로 로봇을 개발하여 다양한 종류의 로봇으로 발전 가능한 로봇 기반 소프트웨어/하드웨어 플랫폼의 아키텍처를 연구하는 것이었다. CMR은 하드웨어 부품 단위로 모듈화 되었으며 상품화를 고려한 염가형 로봇 플랫폼이고, FMR은 작업 기능 단위로 모듈화 되었으며 퍼스널 로봇의 전 기능을 탑재하여 미래 지향적 기술 개발을 위한 로봇 플랫폼이다. 현재는 CMR의 기구적 모듈화 방식을 개량하고 소프트웨어 아키텍처와 라이브러리를 보강한 CMR-P2가 개발되고 있다. CMR-P2는 자율 이동을 위하여 전방향(omni-directional) 이동 모듈과 레이저 스캐너(laser scanner)를 사용하며, 객체를 인식하여 잡고 운반하기 위하여 팔 모듈과 비전을 사용한다. 로봇의 소프트웨어는 센서(sensor)나 액츄에이터(actuator)의 제어· 모니터링 알고리즘, 비전, 자율 이동, 사용자 인터페이스와 같은 라이브러리와, 이들이 분산 환경에서 상호 결합되어 동작하도록 하는 기반을 제공하는 아키텍처로 이루어진다. 본 논문에서는 로봇의 제어 알고리즘과 같은 세부 기술은 제외하고, CMR-P2에 사용된 모듈화· 표준화 방안과 분산 환경에서 각 모듈이 연결되기 위한 아키텍처에 대하여 소개한다. 일본 로봇 공업회(Japan Robot Association; JARA)에서는 일본 내의 로봇 메이커들의 참여로 ORiN(Open Robot interface for the Network) 규격을 제정하였다[4]. ORiN은 분산 객체간 통신을 위한 미들웨어로 DCOM을 채택하였다. ORiN을 기반으로 로봇 컨트롤러 간의 데이터 이용을 위한 오픈화와 인터페이스의 표준화가 진행 중이며 적용 사례도 찾아볼 수 있다 [5,6]. 미들웨어 분야는 OMG(Object Management Group), Microsoft를 중심으로 국제 표준을 제정하여 CORBA(Common Object Request Broker Architecture) [7], DCOM(Distributed Component Object Model) [8] 등의 소프트웨어 개발 전반에 걸쳐 사용 가능한 기술 표준이 이미 나와 있다. CORBA와 DCOM은 범용적인 목적으로 개발되었기 때문에 로봇에 사용하기에는 과다한 자원을 요구하고, 예측 가능한 동작을 보장 받기 힘들다. 또한 제어 자동화 분야에서 사용되는 CAN, RS-232같은 통신 미디어에 대한 하부 프로토콜을 개발하는 것에 많은 노력이 요구된다. 애완 로봇 AIBO [9]와 휴머노이드 로봇 SDR(Sony Dream Robot)-X4 [10]를 개발한 소니에서는 자사의 로봇 플랫폼을 개발하기 위하여 Open-R[11] 아키텍처를 사용하고 있다. Open-R은 다양한 하드웨어의 조합이 가능하도록 하드웨어 모듈간 호환성을 가지게 하며, Plug-and-Play로 각 모듈이 연결된다. 또한 새로운 응용 소프트웨어를 위한 호환성 있는 소프트웨어의 작성이 가능하며 새로운 기능의 확장이 용이한 구조이다. Evolution Robotics는 ERSP (Evolution Robotics Software Platform) [12]라 불리는 소프트웨어 컨트롤 아키텍처를 포함한 소프트웨어 개발 키트를 개발하였다. Evolution Robotics 사는 Windows와 Linux OS 기반의 TCP/IP 만 지원하는 소프트웨어 개발 플랫폼을 상용화하였지만, 다양한 네트워크는 지원하지 못하고 있으며 하드웨어도 자회사의 특정 기기만 이용할 수 있는 폐쇄형 소프트웨어 플랫폼 이라고 볼 수 있다.

본 논문에서는 모듈화· 표준화된 방법으로 로봇을 설계하

* 책임저자(Corresponding Author)
논문접수 : 2004. 9. 25., 채택확정 : 2004. 10. 13.
이호길 : 한국생산기술연구원 로봇기술개발본부
(leehg@kitech.re.kr)
김홍석, 양광웅 원대희 : 한국생산기술연구원 제어·지능연구팀
(hskim@kitech.re.kr/ygkgwg@kitech.re.kr/daehee@kitech.re.kr)
최무성 : 한국생산기술연구원 운동메커니즘연구팀
(moosung@kitech.re.kr)

고 개발하기 위한 소프트웨어 아키텍처를 제시하고 있다. 이 아키텍처는 다음과 같은 특징을 가진다: 1) 하드웨어 부품의 모듈화와 소프트웨어 인터페이스의 표준화, 2) 비전, 자율 이동, 상호 작용과 같은 로봇에서 범용적으로 사용되는 라이브러리 제공, 3) 분산 환경에서 다양한 OS와 하드웨어를 지원하는 소프트웨어 아키텍처, 4) 개발자의 선택에 따라 다양한 개발 언어 사용 가능, 5) 모듈간 연결에 다양한 통신 매체를 지원하는 통신 미들웨어(middleware) 제공 등이 있다.

II. 아키텍처

지금까지 퍼스널 로봇에 적용될 수 있는 모듈화 기술은 다양한 방법과 세부 기술들로 개발되어 왔다. 이러한 모듈화로 인한 분산 로봇 시스템의 장점에도 불구하고 여러 제조업체에서 제공되는 기능독립적인 모듈이 소프트웨어적으로 통합이 이루어지지 않아 로봇 개발에 큰 장애가 되고 있다. 또한, 로봇의 활용 분야가 다양해짐에 따라 이를 구현하는 소프트웨어 또한 복잡해지고 방대한 분량의 코드가 작성되어야 한다. 예로서 Evolution Robotics 사의 ERSP 개발 플랫폼은 삼십만 라인 이상의 C++ 코드로 구현되었다. 결국 로봇의 개발은 로봇의 다양한 기능을 구현하기 위하여 다양한 부류의 개발자가 개발한 소스 코드가 통합되는 대형 프로젝트가 되어간다.

위에서 제시한 문제들을 해결하기 위하여 다음과 같은 사항들이 요구된다: 1) 센서와 액츄에이터의 모듈화와 모듈의 표준화된 인터페이스, 2) 운영체제와 하드웨어에 영향을 적게 받는 소프트웨어 구조, 3) 다양한 개발 언어와 개발 환경 사용, 4) 이기종 운영체제와 하드웨어의 쉬운 연계와 다양한 통신 미디어 사용, 5) 로봇 응용 프로그램 개발에 범용적으로 사용 가능한 라이브러리 개발 등.

본 연구에서는 위에서 제시한 사항을 해결하기 위하여 다음과 같은 해결 방안을 제시한다.

- 로봇의 소프트웨어적/하드웨어적 모듈화· 표준화
- 분산 소프트웨어 아키텍처 도입
- 디바이스 드라이버(device driver)와 컴포넌트 제작

1. 모듈화 · 표준화

모듈화· 표준화 기술은 로봇 부품 개발자와 로봇 응용 산업 그리고 로봇 수요층을 효과적으로 연결시킬 중요한 요소이다. 모듈의 기계적/전기적/소프트웨어적 인터페이스 표준과 모듈의 사양 표준에 의하여 같은 종류의 모듈에 대한 호환성과 재사용성을 보장하고, 소프트웨어 컴포넌트의 표준화와 개방화를 가능하도록 한다.

다방면의 로봇 전문가들로 구성된 집단에 의해 모듈화와 표준안이 수립되더라도 모든 종류의 로봇에 적용하기는 어려운 면이 있다. 따라서 본 논문에서는 연구한 표준화 안은 범용적으로 사용 가능한 퍼스널 로봇의 각 모듈에 대한 것이다.

1.1 모듈화

로봇을 구성하는 디바이스는 종류도 많고 기능과 성능도 다양하다. 이들 모두를 고려한 모듈화는 비현실적이며 기술이 진보할수록 모듈화 기준도 달라질 것이다. 본 연구에서는 퍼스널 로봇을 직접 구성하면서 기능적으로 확연히 구분되는 부품 단위로 모듈화를 시도하였으며, 주제어기 모듈, 센서 모듈, 이동 모듈, 팔 모듈 등 네 개의 모듈로 구분되었다.

표 1. 이동 모듈의 기능과 사양.
Table 1. Function and specification of mobile module.

기능	- 이동 제어(X,Y,V) - 오드메트리에 의한 자기 위치 설정 - Steering 위치제어, 속도제어, 리스크 제어	
사양	자유도	2
	가반 중량	40Kg 이상
	주행 속도	1~2Km/h
	주행 오차	주행 거리 비례 ± 2%이하
	주행 능력	높이 20mm 이상 문턱 넘기
	본체 질량	15Kg 이하
	본체 크기	WDH: 400mm× 400mm× 300mm

1.2 기능과 사양의 표준화

센서나 액츄에이터와 같은 하드웨어와 소프트웨어의 발전은 끊임없이 진행되고 있다. 하지만 이들이 서로 호환되기 위해서는 적절한 선에서 기능과 사양이 정의되어야 한다. 개발자는 적절한 사양과 필요한 기능을 가지는 모듈을 조합하여 로봇을 개발한다. 표 1은 이동 모듈의 기능과 사양에 대한 표준화 예시이다.

1.3 하드웨어 인터페이스의 표준화

본 연구에서 제안된 모듈과 프레임의 기계적인 커넥터는 볼트 체결 없이 착 탈이 가능한 구조다. 전원, 통신 상의 연결을 위한 전기적 커넥터로는 HARTING 사의 Type M 모델을 사용하였고, 이는 4개의 전원용 핀과 60개의 통신용 핀으로 구성되어 있어서 다양한 통신 인터페이스의 지원이 가능하다. 통신용 핀은 Ethernet용 2개, USB용 3개, IEEE1394용 2개, RS-232용 4개, CAN용 2개로 나누어 사용한다. 전기적 통신 인터페이스로는 TCP/IP, USB, IEEE1394, RS-232, CAN 등이 있으며 신호의 전압, 주파수, 타이밍과 같은 전기적 특성은 각 통신 인터페이스의 표준 규약을 따른다. 모듈에 공급되는 전원은 직류 24V이며 추가로 다른 전압의 전원 공급이 가능하다.

1.4 소프트웨어 인터페이스의 표준화

로봇 소프트웨어를 구성하는 디바이스 드라이버와 소프트웨어 컴포넌트들은 여러 부분으로 나누어 서로 다른 개발자가 구현하게 된다. 소프트웨어 조각들이 결합되기 위해서는 이들 간의 인터페이스가 명확하게 정의되어야 한다. 표2는 이동 모듈의 소프트웨어 인터페이스의 입출력에 대한 예시이다. 소프트웨어 인터페이스는 분산된 네트워크상에 따로 위치한 소프트웨어 컴포넌트 간의 투명한 접근을 제공 하는 서로간의 약속이다. 이런 약속을 정의할 때 C, C++, Java와 같은 특정 언어에 의존하지 않는 인터페이스 정의 언어가 필요한데, 본 연구에서는 이를 위해 CORBA와 DCOM의 IDL과 유 사하며 로봇 모듈에 적합하도록 수정한 MIDL(Module Interf ace Definition Language)을 개발하였다. MIDL로 기술되는 인터페이스는 속성(attribute)과 동작(method)의 묶음으로 이루어지고 인터페이스의 상속 기능을 사용할 수 있다.

MIDL로 기술된 인터페이스는 MIDL 컴파일러를 통해 원하는 프로그래밍 언어에 적합한 스텁/스캐러튼(stub/skeleton)

표 2. 소프트웨어 인터페이스 입출력 사양.
Table 2. Input output specification of software interface.

	항목	함수
입력	위치 명령	void Set_pos(int x, int y)
	속도 명령	void Set_vel(int drv)
	제어 모드	void Set_ctl_mod(int mod)
	시스템 상수	void Set_const(int id, int v)
출력	현재 위치	int Get_pos_x() int Get_pos_y()
	현재 속도	int Get_vel()
	제어 모드	int Get_ctl_mod()
	상수와 변수	int Get_const(int id)

코드로 변환된다. MIDL은 구현을 위한 언어가 아니며 소프트웨어 컴포넌트의 구현은 C, C++, RPL, Java 등의 프로그래밍 언어를 통하여 이루어진다. 현재 MIDL은 C, C++, RPL 언어로의 매핑을 지원하고 있으며 앞으로 Java, Python과 같은 언어로의 매핑이 추가될 예정이다.

2. 분산 소프트웨어 아키텍처

로봇의 기능이 복잡해지면서 로봇은 다양한 CPU와 운영체제를 사용하는 한 개 이상의 모듈로 구성되는 경우 모듈간 통신을 고려해야 하고, 프로그램이 하드웨어와 OS상에서 바로 개발되는 경우 하부의 하드웨어와 OS의 구조에 영향을 많이 받게 된다.

본 논문에서 제안하는 MRSF(Modular Robot Software Framework) 아키텍처는 응용 프로그램을 하부의 하드웨어와 OS로부터 분리시키고 분산 환경에 대응하도록 제안되어 있다. MRSF 아키텍처는 로봇 소프트웨어를 4계층으로 분리하고, 4개의 소프트웨어 개발 그룹으로 나뉘어 개발된다.

- 응용 프로그램 개발
- 라이브러리, 컴포넌트 개발
- 아키텍처 개발
- 디바이스 드라이버 개발

응용 프로그램은 최상위 소프트웨어로서 HRI(Human Robot Interface)와 같은 것이 해당된다. 응용 프로그램 개발에는 하부 계층의 디바이스 드라이버와 라이브러리가 이용된다. 모듈 소프트웨어 개발자들은 하드웨어 모듈에 운영체제를 설치하고 MRSF 아키텍처와 개발에 필요한 라이브러리를 설치하여 응용프로그램이나 라이브러리 개발이 가능한 상태의 모듈을 제공하여야 한다. 라이브러리와 컴포넌트는 특정 기능을 구현한 소프트웨어의 모음이다. 예를 들자면, 위치 추정, 개체 인식과 같은 소프트웨어가 해당된다. 라이브러리는 언어에 종속적인 반면 컴포넌트는 MIDL로 정의되는 인터페이스를 가지기 때문에 언어에 독립적이다. 한 개 이상의 모듈이 로봇을 구성하는 경우 컴포넌트는 각 모듈에 분산되어 수행될 수 있다.

MRSF 아키텍처는 개념으로서만 개발자에게 제공되는 것이 아니라, 소스 코드로 구현된 형태가 제공되어 개발에 바로 사용될 수 있다. 결국 응용 프로그램 개발자들은 아키텍처에 대한 전반적인 이해 없이도 자신의 영역에 해당하는 개발을 완수할 수 있다. 또한 아키텍처는 일관된 구조 아래서 아키텍처 내에 새로운 기능을 추가할 수 있도록 설계되었다. 예를 들어, 새로운 인터프리터의 추가, 새로운 통신 프로토콜의 추가 혹은 새로운 디바이스의 추가 등이 이에 해당한다.

디바이스 드라이버(혹은 Internal API)는 하드웨어에 종속된 소스 코드의 모음이다. 모듈이 하드웨어 입출력 장치를 가진다면 이들을 제어 모니터링 하는 기능이 구현되고, 표준 인터페이스로 정의되며, 이는 상위 계층의 라이브러리나 응용 프로그램에서 이용될 수 있다. 또한 디바이스 드라이버는 아키텍처의 일부로 포함되고 아키텍처와 함께 C 언어로 구현되어 있다.

우리는 선행 연구 [13,14]를 통해 로봇을 제어하기 위한 언어(Robot Programming Language; RPL)와 로봇 가상머신(Robot Virtual Machine; RVM), 그리고 통신 미들웨어에 대하여 연구하였다. RPL은 플랫폼 독립적으로 로봇의 라이브러리나 응용 프로그램을 작성하기 위해 사용한다. RPL의 플랫폼 독립적인 특성은 여러 가지 장점을 가지고 있는데, 통상적으로 알려진 바와 같이 컴파일 된 결과물인 바이트 코드에 이동성을 제공하여 줄뿐만 아니라 모듈이나 로봇의 최종 사용자 지향적인 프로그램 작성을 가능하게 해준다. RPL은 대부분의 프로그래머에게 익숙한 C 언어를 기반으로 개발되었으며, 포인터와 같이 플랫폼에 의존적이며 오류 유발성이 높은 문법이 포함되어 있지 않은 특징을 갖는다.

바이트 코드를 실행하기 위하여 RVM이 필요하다. 일반적인 가상머신의 정의는 어떤 기계 구조나 하드웨어 플랫폼을 모사한 환경 또는 이것에 의해 실현된 가상적인 기계 환경을 의미한다. 가상머신은 실제의 하드웨어와 달리 소프트웨어적으로 구현되기 때문에 하드웨어의 변경과 같은 일이 일어나도 가상머신 상에서 운영되는 로봇제어 프로그램의 변경이 요구되는 가능성을 줄여주게 된다.

통신 미들웨어는 로봇을 구성하는 모듈 간에 수행되는 통신 서비스를 제공하기 위한 메시지 포맷 및 전송 방법을 정의 한다. 모듈 사이의 메시지라 함은 모듈에서 수행되는 모듈제어 프로세서가 원격지의 모듈제어 프로세서 데이터를 요구하거나 원격지의 모듈제어 프로세서의 함수를 호출하기 위한 목적으로 전송되는 메시지를 말한다. 이러한 기능을 위해서는 모듈 간에 표준화된 메시지 포맷을 사용하여야 한다.

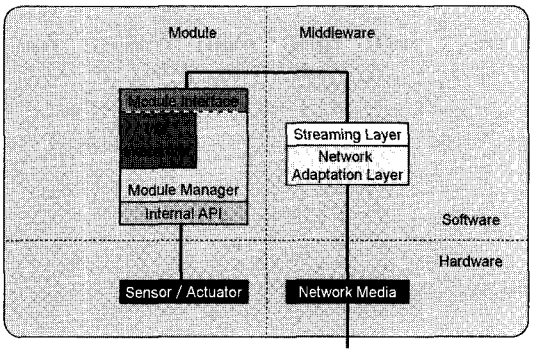


그림 1. MRSF 아키텍처.
Fig. 1. MRSF architecture.

그림 1에서 보듯이, MRSF 아키텍처는 인터프리터(interpreter)와 미들웨어(middleware), 디바이스 드라이버 (혹은 Internal API)로 구성된다. 디바이스 드라이버와 컴포넌트간의 인터페이스는 MIDL로 기술되며, 각 모듈이 따라야 할 표준 기능에 따라 기술되어야 한다.

2.1 인터프리터

MRSF 아키텍처 상에서 로봇의 라이브러리와 응용 프로그램을 개발할 때 사용되는 언어는 C, C++, RPL 등이 있다. RPL은 본 연구에서 C언어를 참고하여 개발 한 로봇 프로그래밍 언어이다. RPL로 쓰여진 소스 코드는 컴파일 되어 바이트 코드를 생성하는데, 바이트 코드를 수행하기 위해 RVM 인터프리터가 필요하다.

앞으로 로봇의 라이브러리와 응용 프로그램 개발에 Java, Python 등이 추가로 사용될 예정이며, 새로운 언어를 지원하기 위하여 아키텍처에는 새로운 인터프리터가 추가되어야 한다. 자바 가상머신(Java Virtual Machine; JVM), Python 인터프리터 등이 추가될 예정이다.

2.2 통신 미들웨어

통신 미들웨어는 두 계층으로 구성되는데, 1계층인 네트워크 적응 계층(Network Adaptation Layer; NAL) 과 2계층인 스트리밍 계층(Streaming Layer; SL)으로 구분된다. SL에서는 응용 프로그램이 전송하는 메시지 데이터 구조를 스트림으로 변환하거나 그 역의 기능을 수행한다. NAL은 다양한 이종 네트워크를 수용하며 이들을 추상화 시킨다[15].

NAL은 SL로부터 넘겨받은 데이터 스트림(data stream)을 플랫폼과 연결된 네트워크 미디어를 통하여 원격 노드의 NAL로 전송하고, 원격 노드의 NAL은 수신된 스트림을 주어진 패킷의 형식과 조건에 따라 필요하면 재조합 과정을 거쳐 최종적으로 완성된 데이터를 SL로 전달한다. 이때 사용되는 네트워크 미디어는 무선 랜, IEEE1394, Bluetooth, Ethernet 그리고 USB와 같이 다양하게 사용될 수 있다. NAL은 이러한 다양한 이종 네트워크 미디어를 동시에 수용할 수 있고, 이종 네트워크간 연결을 위해 라우터 기능을 가진다.

SL은 프로세서가 원격지의 로봇제어 프로세서에 명령을 전송할 경우, 명령을 표현하는 객체는 스트림으로 엔코딩(encoding)되어 NAL을 통해 원격지로 전송된다. 원격지에서는 스트림을 명령을 표현하는 객체로 디코딩(decoding)하는 과정을 거치게 된다.

3. 디바이스 드라이버와 라이브러리

디바이스 드라이버는 실질적으로는 MRSF 아키텍처에 포함되는 코드로서 하드웨어 및 운영체제에 의존적인 부분이거나 모듈 제조업체의 기존 소프트웨어 기술력을 보존하기 위해 구현을 공개하지 않는 부분이 될 수 있다. 소프트웨어 라이브러리는 로봇 응용프로그램 개발자가 소프트웨어를 작성하면서 필요로 하는 기능을 체계적으로 분류하여 라이브러리화 한 것이다. 라이브러리는 언어에 종속적이지만 컴포넌트화 되었을 때는 언어에 독립적으로 사용 가능하다. 객체 인식, 문자 인식, 음성 인식· 합성과 같은 소프트웨어는 이미 컴퓨터 응용프로그램들로 사용되고 있으면서 직접 개발하기에는 많은 노력과 시간이 필요한 부분이므로, 이러한 것들은 기존 소프트웨어를 활용할 계획이다. 네비게이션과 개체인식 라이브러리는 논문이나 기술서적을 통하여 공개된 방법을 라이브러리로 구현한다.

4. 개발 도구

통합 개발 환경, 컴파일러, 링커, 하드웨어 설계 툴과 같이 로봇 개발 과정에 필수적으로 혹은 개발자 편의를 위해 사용되어야 할 도구들이 있다. 본 절에서는 연구 결과로 개발된 로봇 디자인 센터와 컴파일러에 대하여 소개한다. 로봇 디자인 센터를 이용한 소프트웨어 개발 방법론은 [16]에 소개되어 있다.

4.1 로봇 디자인 센터

로봇 디자인 센터 (Robot Design Center; RDC)는 모듈화 된 로봇 부품을 조합하여 하나의 로봇을 완성하기 위해 개발자가 사용하는 통합 개발 환경이다 [17]. RDC는 개발자가 로봇을 개발하기 위하여 사용하는 툴이므로, 개발자가 개발을 편리하게 진행할 수 있도록 택스트 편집기(text editor), 블록 편집기(block editor), 라이브러리 관리, 디버깅, 컴파일, 프로그램 설치, 모듈 상수 설정과 같은 기능을 가진다. 로봇 디자인 센터는 모듈 개발자의 로봇제어 프로그램 개발, 소프트웨어 통합 환경 및 유지 보수 그리고 로봇 응용 프로그램 개발에 사용된다.

4.2 로봇 프로그래밍 언어

로봇을 제어하기 위해 C, C++, RPL, Java와 같이 다양한 언어로 프로그램을 작성할 수 있다. 그 중 RPL은 플랫폼 독립적으로 로봇의 라이브러리나 응용 프로그램을 작성하기 위해 사용한다.

RPL 컴파일러(RPL Compiler; RC)는 RPL로 작성된 소스 코드를 가상머신에서 수행 가능한 형태인 바이트코드로 컴파일 한다. 바이트코드는 하드웨어 플랫폼에 무관하게 모듈의 가상머신에 의해 실행될 수 있도록 정의된 중간코드이다. 따라서, 소스 코드가 로봇 바이트 코드(byte code)로 컴파일 되었을 경우, 모듈에 가상머신이 구현되어 있는 경우 어디에서든 실행시켜 줄 수 있다. 이는 소프트웨어 개발자로 하여 하드웨어 플랫폼이나 운영체제에 무관하게 프로그램을 작성할 수 있도록 한다.

4.3 MIDL 컴파일러

현재 MIDL 컴파일러에서 생성 가능한 스텁(stub) 및 스캐러튼(skeleton) 코드는 C, C++, RPL 이며, 앞으로 Java, Python에 대한 스텁과 스캐러튼을 생성할 수 있도록 기능이 추가될 예정이다.

III. CMR-P2 개발

모듈을 통합하고 성능을 평가하기 위해 제작되는 로봇 CMR-P2는 주제어기 모듈, 이동 모듈, 센서 모듈, 팔 모듈 등 4개의 모듈로 구성된다. 모듈을 착탈하는 로봇의 프레임에는 전원 공급부와 네트워크 연결부를 추가하였다. 각 모듈에는 X-scale, Intel Pentium CPU등 다양한 하드웨어 상에서 Embedded Linux나 Windows 운영체제가 설치되어있다.

1. 프레임

본 연구에 앞서 개발된 CMR[18]의 모듈은 아래에서 위로 쌓아가는 형식으로 모듈화 되었다. 하지만 앞으로 개발 될 로봇의 크기와 모양의 다양성을 고려할 때, 이러한 방식으로

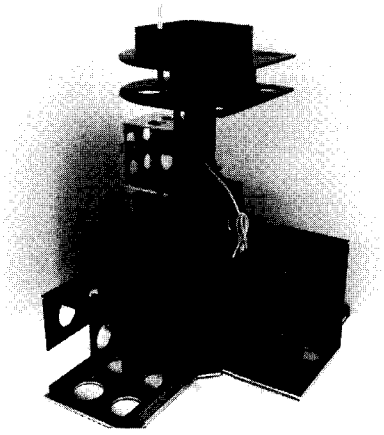


그림 2. CMR-P2 프레임.
Fig. 2. Frame of CMR-P2.

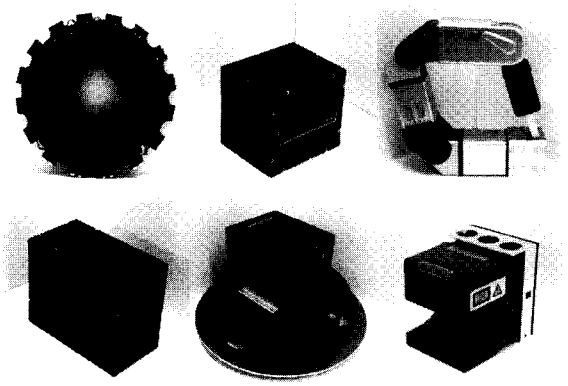


그림 3. 하드웨어 모듈.
Fig. 3. Hardware module.

는 모듈의 크기가 일정하게 정해져 있으므로, 모듈을 구성하는 기계적 인터페이스 부분을 새로 개발해야 하는 문제점이 있다. 그래서 CMR-P2에서 사용한 방식은 로봇의 개략적인 모양을 결정하는 로봇 프레임과 각 모듈을 기능적으로 분리하여 제작하는 형식으로 하였다.

CMR-P2에서 사용하는 프레임을 그림 2에서 보여주고 있다. 프레임은 모듈을 착탈 가능하게 하는 로봇의 골격으로서, 각 모듈에 전원을 공급하고 모듈을 통신 선으로 연결한다.

2. 하드웨어 모듈

하드웨어 모듈은 표준화된 사양에 따라 제작되어야 한다. 모듈의 기계적 접속부는 표준으로 정해져 있지 않고 자율적으로 제작이 가능하게 하였으나, 전원 공급과 통신에 사용되는 연결 커넥터는 표준안에 따라 제작되었다. CMR-P2에서 사용된 하드웨어 모듈을 아래의 그림 3에서 보여주고 있다. 좌측 상단부터 초음파 센서, 센서 모듈, 팔 모듈, 주제어기 모듈, 이동 모듈, 레이저 센서이다. 초음파 센서와 레이저 센서는 RS-232 프로토콜로 연결되지만 각 사에서 정의한 프로토콜만 사용 가능하므로 표준 소프트웨어 인터페이스로 변환하기 위해 센서 모듈이 사용된다. 모듈의 종류에 따라 소프트웨어 인터페이스와 기능은 표준으로 정의되어 있다. 각 모듈의 기능에 따라 하드웨어를 제어하기 위한 디바이스 드라이버가 표준안에 따라 개발되어야 하기 때문에, 디바이스 드라이버는 표준안에 따라 C언어로 작성되어 아키텍처 소프트웨어와 함께 컴파일·링크 되어 모듈에 설치되었다.

계층화된 소프트웨어 플랫폼 구조에서 디바이스 드라이버는 하드웨어 추상 계층(Hardware Abstraction Layer; HAL)에 속하며, 하드웨어 추상 계층은 라이브러리나 응용 프로그램으로부터 디바이스(sensor, actuator 등) 종속성을 제거하는 역할을 한다. 이러한 특성은 아키텍처와 응용 프로그램의 이식성을 보장하고 모듈의 호환이 가능하도록 한다.

2.1 주제어기 모듈

주제어기 모듈은 센서 모듈에서 읽어 들인 정보를 처리하여 그 결과를 이동 모듈이나 팔 모듈로 내려 보내는 기능을 가진다. 각종 알고리즘과 사용자 인터페이스를 위한 프로그램들이 동작하여야 하기 때문에 GUI (Graphic User Interface) 개발이 쉬운 Windows 운영체제를 선택하였으며, 높은 성능을 가지는 Pentium 호환 800MHz CPU를 사용하는 CPU 보드를 선택하였다. 주제어기 모듈에서는 경로 계획(path planning), 위치 추정(localization), 이동(navigation) 알고리즘이 동작하고, 비전을 이용한 물체 인식, 얼굴 인식, 표정 인식이 알고리즘이 동작한다.

2.2 센서 모듈

센서 모듈은 CMR에 사용하던 초음파 센서 어레이에 적외선 스캐너, 자이로 센서, 레이저 스캐너를 추가하여 센서의 종류를 늘이고 디바이스 드라이버의 기능을 추가하였다.

2.3 이동 모듈

이동 모듈은 ㈜한울로보틱스 모바일 베이스 3차 모델을 사용하였다. 모바일 베이스에는 ARM CPU를 사용하는 프로그래밍 가능한 보드가 내장되어 있어 아키텍처 소프트웨어를 이식하기 위한 추가적인 장치가 필요하지 않다.

2.4 팔 모듈

팔 모듈은 ㈜로봇앤드디자인에서 제작한 모듈형 팔(modular arm)을 사용하였다. 모듈형 팔의 각 모듈은 CAN으로 연결되어있다. 모듈형 팔의 경우, 팔 자체에 프로그래밍 가능한 제어기가 없어서 ARM CPU를 사용하는 임베디드 보드를 따로 장착하고 여기에 아키텍처 소프트웨어를 이식하고 동작을 제어하는 디바이스 드라이버를 개발하였다.

3. 소프트웨어 라이브러리

라이브러리는 크게 비전, 자율 이동, 상호 작용으로 나누어 개발하고 있다. 각 라이브러리는 논문이나 기술서적을 통하여 공개된 알고리즘을 참고하여 구현하였으며, 문자 인식, 음성 인식·합성과 같이 구현이 까다로운 소프트웨어는 기존 소프트웨어를 활용할 계획이다.

3.1 비전

비전 라이브러리는 카메라 영상으로부터 물체 인식, 동작 탐지, 색 탐지, 위치 추정을 위한 특징 점 추출과 같은 다양한 작업을 수행한다. 본 연구에서의 비전 기술은 범용으로 사용되기에는 부족한 면이 있으며 앞으로 추가 연구가 필요한 분야이다. 비전의 영상 처리 속도를 높이기 위하여 빠른 속도의 CPU를 필요로 한다. 비전을 이용한 객체인식 알고리즘은 Pentium 호환 800MHz CPU에서 320x240 크기의 영상을 초당 10 프레임 속도로 처리하였다.

3.2 자율 이동

자율 이동 라이브러리는 로봇이 목적지까지 최단시간과 경로로 이동 가능한 경로를 생성하고 지도에 존재하지 않는 장애물이 감지되는 경우 이를 회피하는 동작까지 포함하였다. 지도는 로봇의 자율 이동에 중요한 정보를 제공하며, 사용하는 지도의 종류는 위상 지도(topological map), 격자 지도(grid map), 벡터 지도(vector map) 3 가지가 있다.

3.3 상호 작용

상호 작용 라이브러리는 사람-로봇 인터페이스 개발에 사용되는 다양한 API를 제공하고 있다. 사람의 감정을 인식하고 로봇의 눈과 입 모양으로 감정 표현이 가능한 라이브러리들이 개발되어 있다. 사람과 대화를 위한 음성 인식 및 텍스트-음성 변환 라이브러리는 기존 소프트웨어를 도입할 예정이다.

e-Mail과 휴대폰 문자 전송 라이브러리는 사용자가 없는 환경에서 로봇이 단독으로 동작하는 중에 사용자에게 알릴 메시지가 있을 경우 사용한다. 일반적인 통보 메시지나 영상, 음성 정보는 e-Mail을 통해 전송하고, 긴급한 메시지의 전송에는 SMS를 이용하였다.

원격 조정 라이브러리는 PDA등을 이용하여 로봇과 멀리 떨어진 상태에서 로봇의 제어와 모니터링이 가능하도록 구현되어 있다.

4. 통합 테스트

본 장에서 제안한 구조대로 모듈화· 표준화 된 로봇 플랫폼의 개발 용이성과 성능을 검증하기 위하여 응용 가능한 시나리오를 만들고 이에 따라 로봇을 구현하였다. 안내, 심부름, 경비 작업 등 세 가지 시나리오를 가지는 로봇을 목표로, 모듈을 조립하고 응용 프로그램을 작성하여 로봇 플랫폼을 완성하였다. 그리고, 모듈의 추가 및 제거를 통해 개발된 로봇 플랫폼이 각 시나리오에서 요구되는 다양한 기능의 하드웨어 플랫폼으로 단시간에 변경이 가능하다는 것과, 소프트웨어의 별다른 수정 없이도 변경된 플랫폼에서 각 시나리오의 목표 수행이 가능하다는 것을 보여주었다.

VI. 결론

본 연구에서는 모듈화· 표준화된 개발 방법과 이를 지원하기 위한 소프트웨어/하드웨어 아키텍처에 대하여 연구하고, 이를 바탕으로 로봇 플랫폼을 제작하여, 몇 가지 시나리오를 통하여 연구의 유용성을 검증하였다. 연구 결과는 다음과 같이 요약할 수 있다.

1) 모듈화· 표준화 안 제시
2) 분산 소프트웨어 아키텍처 개발
3) 디바이스 드라이버 및 컴포넌트 개발
4) 로봇 플랫폼 개발

연구 결과는 다양한 종류의 로봇을 구현하는 기반 소프트웨어/하드웨어 플랫폼으로 사용될 것이다. 앞으로 Java, Python 등이 로봇의 응용 프로그램 개발에 사용될 언어로 추가될 계획이며, CAN 프로토콜이 미들웨어의 NAL에 추가될 계획이다. 무엇보다 응용 프로그램 개발에 필수적으로 사용되는 비전, 이동, 상호 작용과 같은 소프트웨어 라이브러리의 기능이 향상되어야 하고, 기 개발된 로봇 플랫폼을 기반으로 하여 새로운 소프트웨어 컨텐츠가 지속적으로 개발되고 공유되어야 할 것이다.

참고문헌

[1] S. G. Roh, S. M. Baek, D. H. Lee, K. H. Park, T. K. Moon, S.W. Ryew, J. Y. Kim, T. Y. Kuc, H. S. Kim, H. G. Lee, H. R. Choi, "Development of personal robot platform : approach for modular desing," *ICCAS*, pp. 2313-2318, October 2002.

[2] S. G. Roh, K. H. Park, K. W. Yang, H. S. Kim, H. G. Lee, and H. R. Choi, "Development of dynamically reconfigurable personal robot," *ICRA*, pp. 4023-4028, 2004.

[3] S. G. Roh, K.H. Park, K.W. Yang, J.H. Park, H.S. Kim, H.G. Lee and H.R. Choi, "Dynamic infrastructure for personal robot : dynI," *ICCAS*, pp. 2039 - 2044, 2003.

[4] M. Mizukawa, H. Matsuka, T. Koyama, A. Matsumoto, "ORiN: open robot interface for the network, a proposed standard," *Industrial Robot*, vol. 27 no. 5, pp. 344-350, September 2000.

[5] M. Mizukawa, H. Matsuka, T. Koyama, T. Inukai, A. Noda, H. Tezuka, Y. Noguchi, and N. Otera, "ORiN: open robot interface for the network, the standard network interface for industrial robots and its applications," *ISR*, 2002.

[6] M. Mizukawa , T. Koyama , T. Ihukai , A. Noda, N. Kanamaru, Y. Noguchi and N. Otera, "Proposal of open-network-interface for industrial robots (ORiN) and its experimental evaluation," *Proceedings of the 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, vol. 2, pp. 689-694 , July 2001.

[7] "Common object request broker architecture: core Specifi cation," *OMG*, March 2004.

[8] Markus Horstmann and Mary Kirtland, "DCOM architecture," *MSDN*, July 1997.

[9] http://www.us.aibo.com

[10] http://www.sony.co.jp/en/SonyInfo/News/Press/200203/020319E/.

[11] M. Fujita and K. Kageyama, "An open architecture for robot entertainment," *Proc. International Conference on Autonomous Agents*, pp. 435-440, 1997.

[12] http://www.evolution.com/

[13] G. Yoon, H. Y. Kim, J. S. Lee, H. S. Kim, H. S. Park, "Middleware structure for personal robot," *ICCAS*, pp. 153-157, 2003. 6.

[14] K. W. Yang, H.-S. Kim, Jaehyun Park, "A virtual machine for modularized personal robot controller," *Proceedings of ICCAS*, pp. 2170-2173, Oct. 16-19, 2002, Muju, Korea.

[15] 윤건, 김형육, 박홍성, "모듈 기반 퍼스널 로봇을 위한 미들웨어 구조," 제어·자동화·시스템공학 논문지, 제10권, 제5호, pp. 464-474, 2004. 5.

[16] H. G. Kim, D. W. Kim, HongSeok Kim, and Hogil Lee, "Toward the personal robot software framework," *ICCAS Conf.*, pp. 2307-2312, 2002.

[17] "표준형 로봇디자인센터(Robot Design Center)의 프레임워크 개발," 제2회 퍼스널로봇 기반기술개발 Workshop, pp. 175-193, 2003.

[18] S.-W. Ryu, K. W. Yang, H.-S. Kim, H.-G. Lee, "Functionally distributed modular robot system using virtual machine," *Proceedings of ICCAS*, pp. 2330-2335, Oct. 16-19, 2002, Muju, Korea.

이 호 길

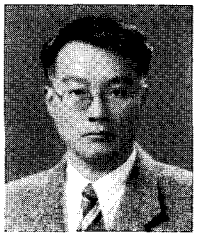
1989년 오사카대학교 로봇공학 공학박사 졸업, 1980년~1982년 현대정공 사원, 1989년~1991년(일본) ASTEM 연구소 주임연구원, 1991년~현재 한국생산기술연구원 수석연구원. 관심분야는 로봇제어, 신호처리, 로봇환경기술.

양 광 웅

1998년 인하대학교 자동화공학과 공학석사 졸업, 1998년-2002년 (주)두산메카택 근무, 2002년-현재 한국생산기술연구원 근무. 관심분야는 Robot SoftwareArchitecture, Compiler, Virtual Machine.

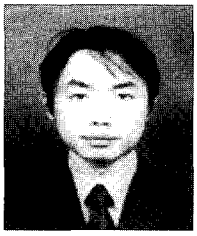
원 대 희

2002년 한양대학교 대학원 정밀기계공학과 공학석사 졸업, 2002년-현재 한국생산기술연구원 근무. 관심분야는 Hovering Robot, Ubiquitous Computing & Sensor Network Embedded System (DSP &Micr ocontroller).



김 홍 석

1990년 서울대학교 대학원 제어계측공학과 공학박사 졸업, 1990년~1991년 한국과학기술연구원 응용전자연구실 연수연구원, 1991년~현재 한국생산기술연구원 로봇기술개발본부 수석연구원/팀장. 관심분야는 제어이론, 제어기 설계 및 평가, 시뮬레이션(simulation), 전자회로.



최 무 성

2003년 한양대학교 정밀기계공학과 공학석사, 2003년~현재 한국생산기술연구원 근무. 관심분야는 Mobile robots, Biped robot.