특집

URC를 위한 로봇 S/W 아키텍처 기술

김성훈, 김중배(한국전자통신연구원 지능형로봇연구단)

요 약

URC(Ubiquitous Robotic Companion) 환경에서 소프트웨어 로봇은 인공지능 기술과 네트워크 기술을 활용하여 상황에 따라 사용자에게 맞는 맞춤형 서비스를 제공하는 지능형 에이전트 기술로 정의할 수 있다. 소프트웨어 로봇은 인터넷 상에 존재하는 가상의 로봇이며, 실물 로봇을 제어하기 위해서는 로봇 내부에 로봇의 태스크와 행위, 하드웨어 디바이스를 제어하기 위한 아키텍처가 필요하다. 본 고에서는 소프트웨어 로봇이 실물 로봇을 제어하기 위해서 필요한 로봇 내부의 소프트웨어 아키텍처의 요구사항과 시스템을 소개한다.

I. 서 론

정해진 일을 반복 수행하는 산업용 로봇과 달리 지능형 서비스 로봇은 일반 가정 또는 사무실에서 사람들과 더불어 생활하면서 인간과 상호작용하고 주어진 상황에 맞게 적절한 태스크를 수행함으로써 인간의 안녕과 복지를 도모하는 로봇이다. 서비스 로봇에 대한 사용자들의 요구와 기대감은 스타워즈, A.I와 같은 SF영화나 소설 등에 나오는 로봇들과 정보 사회의 발달로 인해 상당히 높아져 있는 것이 사실이다. 이러한 사회 구성원의 의식 변화와 더불어 포화 상태에 있는 산업용 로봇의 시장에 대한 새로운 활로를 위해, 최근 로봇 산업은 산업용 로봇 분야에서 가정과 사무실에서 사람들에게 서비스하는 지능형 서비스로봇 분야로 확장되고 있다.

정부에서는 2004년도부터 국민 2만 달러 시대를 여는 핵심 기술로 로봇을 선정하고 정부, 학계, 연구소, 산업계가 공동으로 URC(Ubiquitous Robotic Companion)라는 개념으로 네트워크 기반의 지능형 서비스 로봇 사업을 추진하고 있다. URC는 "언제 어디서나, 나와 함께 하며, 나에게 필요한 서비스를 제공하는 로봇"을 의미하며, 세계 최고 수준의 유무선 인프라를 활용하여 로봇 기술을 고도화하는 것을 목표로 하고 있다. URC 사업에서는 네트워크 기능을 활용하여 로봇의 인식기술과 HRI(Human Robot Interaction) 요소 기술의 성능을 개선하고 이를 통해 지능형 로봇의 서비스 질과 지능을 높일 수 있을 것으로 기대하고 있다.

URC 기반의 소프트웨어 로봇은 주변 환경으

로부터 필요한 정보를 습득할 수 있고, 사용자가 어디에 있든지 네트워크를 통해 사용자에게 다가갈 수 있는 속성을 가지고 있다. 그러나 소프트웨어 로봇이 실물 로봇을 통해 집안을 청소하거나 음료 서비스와 같은 물리적인 상호 작용을 하고자 할 때는 반드시 실물 로봇을 제어할 수 있는 기능이 있어야 한다. 또한 로봇 내부에는 표준화된 형태의 아키텍처가 존재하여야 하며 이를 통해 소프트웨어 로봇이 다양한 형태의 실물 로봇을 제어할 수 있을 것이다.

본고에서는 소프트웨어 로봇이 물리적인 로봇을 통해 환경을 인지하거나 로봇을 제어하고자 할 때 필요로 하는 로봇 내부의 소프트웨어 아키텍처에 관하여 소개하고자 한다. 본고의 로봇 소프트웨어 아키텍처는 기존 로봇 소프트웨어 아키텍처와 달리 시나리오 기반으로 로봇 태스크를 구동하고, 로봇이 가지는 기본 행위들을 자동화하고, 하드웨어 디바이스에 대한 표준을 제공함으로써 소프트웨어 로봇이 다양한 실물 로봇을 제어할 수 있는 기반을 제공할 수 있다.

본 고의 Ⅱ장에서는 로봇 내부 소프트웨어 아키텍처에 관한 개요를 기술한다. Ⅲ장에서는 로봇 소프트웨어 아키텍처와 구성에 대해 설명하고 Ⅳ장에서 결론을 기술한다.

Ⅱ. URC 로봇 소프트웨어 아키텍처 개요

일반적인 네트워크 기반 지능형 서비스 로봇의 소프트웨어 구성은 다음과 같이 5개의 계층으로 구분할 수 있다.

— 하드웨어 디바이스 연결 계층
— 로봇 행위 실행 계층
— 로봇 태스크 실행 계층
— 로봇 태스크 실행 조정 계층
— 로봇 핵심 기술 서비스 연동 계층

상기와 같은 구성은 로봇 제조회사나 개발자와 무관하게 공통적으로 로봇에 필요한 사항들이다. 로봇 소프트웨어 아키텍처는 상기와 같은 소프트웨어를 보다 손쉽게 개발하고 유지 보수할 수 있도록 표준화된 형태의 구조와 플랫폼을 제공하는 것이다. 표준화된 로봇 소프트웨어 아키텍처상에서 개발된 소프트웨어들은 향후 다른 로봇을 개발할 때 재사용이 가능할 뿐만 아니라 개발 시간을 단축시킬 수 있는 장점이 있다. 로봇 소프트웨어 시스템은 상기 구성들 중에서 로봇이 가져야 할 기본 요소들을 시스템에서 기본으로 제공하고 개발자들은 단지 서비스 시나리오만을 작성함으로써 빠르게 새로운 영역의 서비스를 개발할 수 있도록 하는 것이다.

본고의 로봇 응용은 사용자의 얼굴인식, 단순한 위치 이동등과 같이 단편적인 로봇 기술이 아닌 "아침 6시가 되면 안방에 가서 아빠를 깨우고, 아빠가 일어나면 신문을 읽어줘라"와 같이 사용자가 일상적으로 생각하고 행동하는 방식의 복잡한 형태를 가지고 있다. 본고에서는 상기와 같은 복합적인 형태의 로봇 응용을 보다 손쉽게 개발할 수 있도록 로봇 소프트웨어 개발자를 두 가지 부류로 분류한다. 첫 번째는 시나리오를 기반으로 로봇응용을 개발하는 개발자와 두 번째는 로봇의 핵심 기술 또는 이를 이용한 로봇 행위를 만드는 로봇 개발자이다. 로봇 응용 개발자는 상기와 같은 태스크의 시나리오를 작성하고 로봇 개발자는 이러한 태스크를 수행하기 위한 기본 행위들인 위치 이동, 얼굴 인식, 사람 추종, 인터넷 컨텐츠 연동 등의 행위 컴포넌트를 작성한다. 로봇 소프트웨어 아키텍처는 로봇 응

용 개발자가 작성한 시나리오에 대해서 로봇 개발자가 작성한 행위 컴포넌트를 연결하여 실행하고, 오류 및 예외 상황 처리, 외부 서비스 실행 등의 일을 수행함으로써 로봇 개발자와 로봇 응용 개발자 사이를 연결시켜주는 역할을 수행한다.

로봇 소프트웨어 아키텍처는 로봇 응용 개발자들과 로봇 개발자들을 위한 방법론을 동시에 제공하여야 한다. 로봇 개발자들에게는 로봇 행위 개발을 위한 실시간의 속성과 더불어 프로그램을 위한 프레임워크를 제공하는 것이 바람직하며, 로봇 응용 개발자들에게는 개발자에게 친숙한 형태로 시나리오를 작성할 수 있도록 해야 한다. 본고의 Ⅲ장에서는 로봇 응용 개발자와 로봇 개발자들을 위한 로봇 소프트웨어 아키텍처를 소개하고 구성 내역에 대하여 설명한다.

Ⅲ. URC 로봇 소프트웨어 시스템

URC 환경을 지원하는 지능형 서비스 로봇은 비구조화되고(unstructured), 동적으로 변화하며(dynamic), 부분적으로만 인지할 수 있는(partially perceptual) 환경에서 동작한다. 이러한 URC 환경을 바탕으로 로봇 소프트웨어 아키텍처가 가져야 할 요구사항을 정리하면 다음과 같다[1].

첫째, 분산성을 지원해야 한다.

기존 로봇 소프트웨어 아키텍처들은 대부분 하나의 하드웨어 보드를 가진 모바일 로봇을 대상으로 하였다. 그러나 URC 로봇은 내부적으로는 기능이 분산된 하드웨어 모듈로 구성되며, 네트워크 상에서 URC 서버를 통해서 기능을 확장할 수 있어야 한다.

둘째, 환경 변화에 즉각 대응할 수 있는 반응성을 지원해야 한다.

로봇은 동적이고 부분적으로 밖에 인지할 수 없는 환경에서 동작한다. 따라서 미리 계획된 행동과 더불어 상황에 따라서 즉각적으로 반응할 수 있는 행동특성을 지원해야 한다.

셋째, 하드웨어 및 운영체제 독립성을 지원해야 한다.

한번 작성된 로봇 소프트웨어가 다양한 형태의 하드웨어에 장착되어 동작하기 위해서는 하드웨어 및 운영체제에 대해 독립적인 구조를 지원해야 한다.

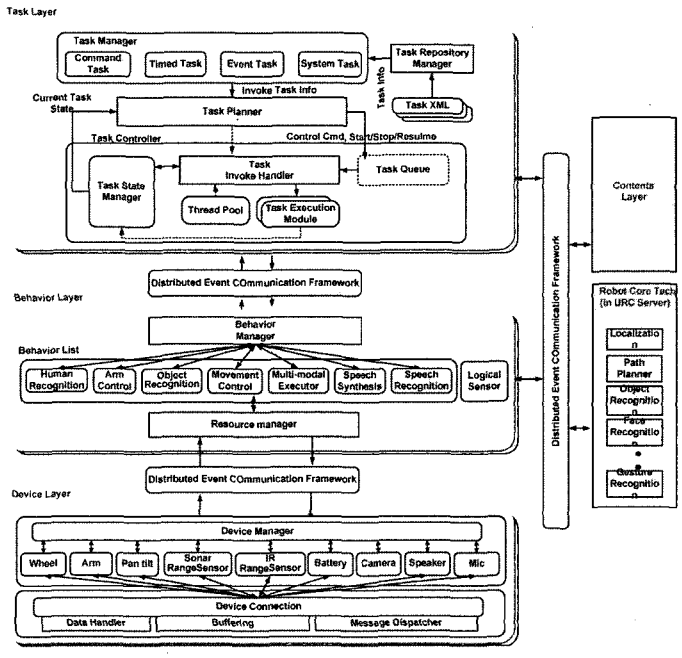
넷째, 예외 상황과 오류에 대해 강건성을 지원해야 한다.

URC 로봇은 실제 가정에서 서비스를 제공하기 때문에 실행시간에 발생할 수 있는 각종 오류에 대해 적절히 대처해야 한다. 특히, 일부 센서의 고장과 같은 하드웨어적인 오류뿐만 아니라 로봇 내부의 판단 오류 등을 정확히 진단하고 적절히 대응할 수 있는 구조를 지원해야 한다.

다섯째, 이벤트 및 시간 기반의 태스크를 지원해야 한다.

URC 환경에서 소프트웨어 로봇과 더불어 로봇이 동작하기 위해서는 기본적으로 외부 환경 변화나 내부 상태에 따라 적절히 행동을 결정할 수 있어야 한다. 또한 시간을 기반으로 예약된 태스크를 수행하기 위해서는 시간 기반의 태스크 지원을 필수라고 할 수 있다.

여섯째, 태스크 수행의 순차성/병렬성/반복성을 지원해야 한다.



〈그림 1〉 URC 로봇 소프트웨어 아키텍처 구조도

URC 로봇의 태스크는 기본적으로 순차적 실행, 반복적 실행, 병렬 실행의 특성을 가지고 있다. 이러한 특성은 사람의 행동 패턴과도 일치하는 것으로 로봇이 사용자에게 보다 가깝게 다가서기 위해서 반드시 필요한 기능이라 할 수 있다.

일곱째, 외부 시스템과 연동을 위한 분산 통신 프레임워크를 제공해야 한다.

로봇 소프트웨어는 외부 소프트웨어 로봇뿐만 아니라 URC서버와 같은 외부 서버와 연동할 수 있다. 따라서 다양한 형태의 로봇 내 외부 시스템과 연동하기 위하여 운영체제와 개발 언어에 무관하게 동작할 수 있는 통신 프레임워크를 제공해야 한다.

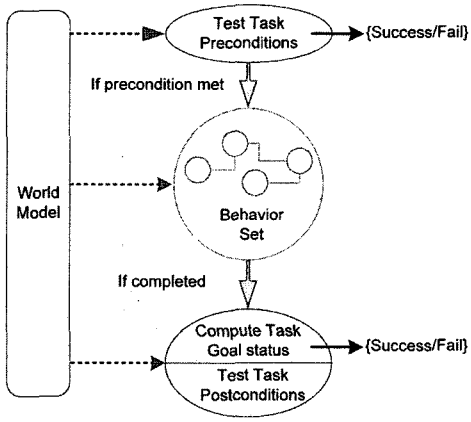
상기 요구 사항을 기반으로 하는 로봇 소프트웨어 아키텍처를 설명하기 전에 먼저 기존 로봇 소프트웨어 아키텍처들을 살펴보면, 우선 먼저, 1980년대 중반부터 시작된 행위 기반 로봇 소프트웨어 아키텍처는 기존의 SPA에 대한 구조적인 단점을 극복하고자 행동지향의 센싱과 행위의 병렬 수행 등을 특징으로 하고 있다[2][3]. 이러한 방식은 인지과정이 빠르고 간결하여 즉각적인 행위 수행을 할 수 있다는 장점이 있다. 그러나 이 방식은 곤충과 같은 저 수준의 지능은 표현하기에 적합하나 그 이상의 지능을 표현하기에는 부족하다는 단점이 있다. 또 다른 형태의 아키텍처인 하이브리드 아키텍처는 행위 기반 아키텍처와 SPA의 구조적인 장점을 살린 아키텍처로써 상위 수준의 지능과 빠른 응답성을 보장하고 있다. 본고에서 소개하는 아키텍처도 이러한 하이브리드형 아키텍처로써 계층별로 구분된 형태를 취하고 있다. <그림 1>은 상기 요구사항들을 기반 작성된 URC 로봇 소프트웨어 아

키텍처에 대한 구조도이다. <그림 1>의 아키텍처는 계층적 구조를 지향하고 있으며 각 계층에 대한 상세한 설명은 다음과 같다.

1. 태스크 실행 계층

태스크 실행 계층은 로봇 응용 개발자가 작성한 서비스 시나리오를 구동하는 계층이며, 소프트웨어 로봇에서 요청된 태스크에 대해 행위 실행 계층의 행위들을 제어하여 주어진 목적을 달성하는 계층이다. 태스크 실행 계층에서 수행하는 로봇 태스크에 대한 예를 들면 "아침에 일어나면, 아빠를 깨우고 신문을 읽어준다"와 같은 형태를 가지고 있다. 이러한 태스크들은 반복성, 순차성, 병렬성의 속성을 갖고 있으며, 로봇의 비전문가에 의해 작성되는 특징을 가지고 있다. 따라서 제공된 로봇 하드웨어를 기반으로 손쉽게 서비스를 개발하고 이를 로봇에 탑재할 수 있어야 한다. 이를 위해서는 로봇 태스크 개발도구의 지원이 필수적이며, 로봇으로 태스크를 배포할 수 있는 방법도 간편하고 쉬어야 한다.

본고의 로봇 태스크는 태스크가 실행되는 조건에 따라서 명령형, 시간형, 이벤트형, 시스템형의 네 가지 형태로 구분한다. <그림 1>의 태스크 관리자는 태스크 저장소에 배포된 각 유형의 로봇 태스크를 실행 대기 상태로 만들고 각 태스크가 실행될 수 있는 조건을 지속적으로 검사하는 부분이다. 실행 조건이 충족된 태스크는 태스크 계획 모듈로 전달되고 태스크 실행 계획 모듈은 수행되고 있는 태스크와 전달된 태스크의 상태에 따라서 태스크의 실행 여부와 순서를 결정하는 부분이다. 본고의 로봇 태스크들은 XML형태로 개발되고 태스크 실행 계층에서는 XML을 태스크 객체로 변환하여 실행한다. 본고에서 소



〈그림 2〉 로봇 태스크 모델

개하는 로봇 태스크의 모델은 <그림 2>와 같다. <그림 2>의 로봇 태스크 모델은 3가지의 요소로 구성된다. 첫번째는 외부 또는 내부 환경에 대해서 수행할 태스크가 활성화될 수 있는 조건을 검사하는 부분이다. 두번째는 태스크를 구성하는 행위들의 집합이다. 행위 집합은 태스크의 목적을 직접적으로 수행하는 부분이다. 마지막으로 로봇이 수행한 태스크의 성공과 실패를 판단하고 자식 태스크를 실행할 조건을 검사하는 부분이다. 로봇 소프트웨어 시스템에서는 <그림 2>의 태스크 모델에 대한 객체를 관리하며 첫번째 구성 요소인 태스크 활성화 조건 검사를 통해 다른 태스크를 동시에 수행하는 기능을 구현할 수 있으며, 마지막 구성 요소인 사후 검사 기능을 통해 다른 태스크를 실행하거나 자신을 반복 실행시킬 수 있는 기능을 구현할 수 있다. 이를 통해 태스크 수행의 병렬성/순차성/반복성을 지원할 수 있다.

2. 행위 실행 계층

행위 실행 계층은 로봇 개발자들을 위한 계층

이며, 로봇 태스크를 구성하는 행위들을 실행하는 계층이다. 상기의 로봇 태스크들은 시나리오 기반으로 작성되기 때문에 개발도구와도 통합이 용이한 XML로 개발되는 것이 좋으나, 로봇의 행위는 다양하고 복잡한 형태를 취하고 있기 때문에 프로그램 코드 형태의 컴포넌트 기반으로 개발되는 것이 바람직하다[4]. 컴포넌트는 재사용성과 이식성을 높일 수 있으며, 로봇 개발자들의 개발 생산성을 높일 수 있는 장점이 있다. 컴포넌트 기반으로 작성된 로봇 행위들이 다양한 형태의 하드웨어에서도 동작하기 위해서는 행위 계층을 지원하는 하드웨어 디바이스 계층과 로봇 핵심 기술 연동 계층에 대한 인터페이스 표준화가 우선되어야 한다.

로봇 개발자들에게 다양한 형태의 행위 컴포넌트를 개발할 수 있도록 하기 위해서 행위 컴포넌트에 대한 요구사항들을 정리하면 다음과 같다.

▷ Thread/Non Thread 실행을 위한 템플릿 제공
개발자가 작성한 Behavior는 Thread 또는 Non Thread로 실행할 수 있어야 한다.

▷ 행위들의 동시 실행 및 동시 중지를 위한 수단 제공
행위개발자가 작성한 행위들은 실행 동기화의 기능을 사용할 수 있으며, 이를 위한 인터페이스를 제공해야 한다.

▷ 행위들의 순차실행과 행위 컴포넌트 내부에서 다른 행위를 호출할 수 있는 수단제공
행위 컴포넌트 안에서 다른 행위를 호출할 수 있는 수단 제공해야 된다.

▷ 행위간에 이벤트를 사용할 수 있는 메커니즘 제공
행위간에 이벤트를 정의하고 상호 이벤트를 주고 받을 수 있어야 한다.

▷ 행위가 호출될 때 입력 정보 값을 실행시간에 전달
일반적인 로직을 포함하는 행위에 대해서 실행을 위해 필요한 정보 값을 실행 시간에 전달할 수 있는 수단의 제공해야 된다.

▷ 로봇의 자율 주행을 위한 서비스 제공
지도 정보, 경로 계획, 자기 위치 추정 등의 기능을 제공 받아야 한다.

▷ 하드웨어 디바이스를 접근하기 위한 수단 제공
단일한 인터페이스를 통해서 모터 또는 모든 센서에 대한 접근성 보장해야 한다.

▷ 분산 행위 컴포넌트 제공
로봇의 행위가 서로 다른 프로세스에 분산되어 있을 경우, 위치 투명성을 보장하고 행위를 실행할 수 있어야 한다.

▷ 행위 로직의 실행 시간 보장
행위 컴포넌트의 로직은 모터를 구동하는 로직이 대부분을 차지하고 있으므로 행위를 작성하는 사용자의 요구에 따라 실행 시간을 보장할 수 있어야 한다.

<그림 1>에서 행위 관리자는 태스크 실행 계층으로부터 전달받은 행위 실행 요청을 수행하는 부분이며, 요청된 행위 객체에 대한 생명주기 관리와 개발자가 작성한 로직을 수행하는 역할을 한다. 리소스 관리자는 여러 행위 컴포넌트가

동시에 동일한 액츄에이터를 접근하는 것을 방지하는 기능을 수행한다.

3. 디바이스 추상화 계층

디바이스 추상화 계층은 URC 로봇에 탑재될 각종 센서 및 액추에이터들에 대한 표준화된 인터페이스를 제공하고 추상화된 디바이스를 접근할 수 있는 방법을 제공한다. 로봇 행위 개발자는 디바이스 추상화 계층의 인터페이스를 통해 코드를 작성함으로써 표준을 준수하기가 용이하며, 작성된 코드는 재사용하기가 쉬워진다. 디바이스 추상화 계층에서는 하드웨어에 대한 공통의 인터페이스 외에 원격지에서 디바이스를 접근할 수 있는 수단을 제공해야 된다. 이를 통해서 소프트웨어 로봇등과 같은 외부 시스템이 표준화된 형태로 로봇을 제어할 수 있다. <그림 1>의 디바이스 연결 모듈은 하드웨어 장치와 대화하기 위한 계층으로서 상위 계층에서 받은 추상화된 사용자 명령을 실제 물리적인 장치에게 운영체제의 장치 드라이버를 이용하여 원시 데이터 형태로 명령을 내려 수행토록 한다. 디바이스 연결 모듈은 다수의 클라이언트들이 동시에 사용 가능하도록 지원해야 하며, 또한 장치와 비동기적인 통신이 가능하도록 지원해야 한다.

4. 분산 통신 미들웨어 계층

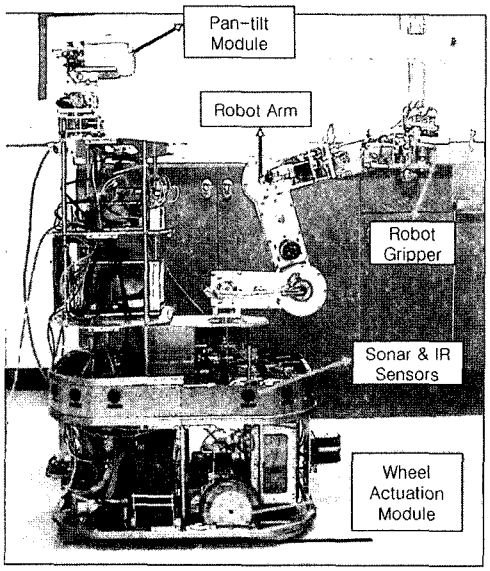
URC 로봇은 저가격의 하드웨어로 구성된다. 따라서 대부분 적은 리소스(메모리, 프로세서 속도 등)를 기반으로 임베디드 개발 환경을 지향하고 있다. 이러한 환경에서 사용자에게 보다 다양한 형태의 서비스를 제공하기 위해서는 필요한 요소 기술들을 네트워크를 통해 적절히 분산할 필요가 있다. 분산 통신 미들웨어는 분산된 로봇의 요소 기술들을 하나로 묶어서 서비스할 수 있는 기반을 제공한다. 분산 통신 미들웨어가 가져야 할 요구 사항들은 다음과 같다.

- ▷ 서버 클라이언트 구조를 가져야 한다.
- ▷ 이벤트(주기/비주기) 기반의 통신을 지원해야 한다.
- ▷ 요청/응답 방식을 제공해야 한다. (동기요청, 타임아웃 기반 동기 요청, 비동기 요청)
- ▷ 통신 프로토콜을 플러그인 할 수 있어야 한다.
- ▷ 다양한 형태의 운영체제에서 동작할 수 있어야 한다.
- ▷ 경량이어야 하고 사용하기 간편해야 한다.
- ▷ 다양한 형태의 개발 언어(C/C++/Java)를 지원해야 한다.

본고에서 소개하는 아키텍처는 <그림 1>에서 보는 바와 같이 각 계층과 외부 로봇 핵심기술을 연동하기 위하여 DECoF(Distributed Event Communication Framework)라는 이름의 분산 통신 미들웨어를 사용하고 있다.

V. 결 론

본고의 로봇 소프트웨어 아키텍처는 uROSE(urc Robot Software development Environment)라는 시스템으로 한국전자통신연구원이 개발한 연구용 로봇 플랫폼인 Wever R2에 장착되어 구동되고 있다. <그림 3>의 Wever R2는 6축 시리얼 타입의 로봇 암을 장착한 모바일 서비스 로봇으로 소프트웨어 로봇과 연동하여 로봇을 구동하거나 URC 서버를 통해 음성 합성, 음성 인식을 수행하고 뉴스/날씨와 같은



〈그림 3〉 웨버 R2

URC 컨텐츠 서비스를 제공한다.

소프트웨어 로봇이 유비쿼터스 환경을 기반으로 사용자에게 보다 다양한 형태의 서비스를 제공하기 위해서는 본고에서 소개한 형태의 로봇 소프트웨어가 필수적이라고 할 수 있다. 또한 다양한 형태의 하드웨어를 가진 로봇과 상호작용하기 위해서는 로봇 내부에 표준화된 형태의 로봇 소프트웨어와 통신 메커니즘이 필요하다. 로봇 소프트웨어 아키텍처는 소프트웨어 로봇에 물리적인 능력을 부여함으로써 URC를 위한 보다 나은 서비스를 제공할 수 있을 것으로 기대한다.

참고문헌

[1] 이승익, 장철수, 정승욱, 김중배, "로봇소프트웨어 아키텍처 연구동향과 현황", 정보통신동향분석 제20권 제2호 2005년 4월.

[2] R.A. Brooks, "A Robust Layered Control System for a Mobile Robot", IEEE Journal on Robotics and Autonomous, Vol.2. No.1. 1986.

[3] Paolo Pirjanian, "Behavior Coordination Mechanisms", Tech report IRIS 99 375, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, October, 1999.

[4] Antonio Carlos Dominguez Brito, "CoolBOT: a Component Oriented Programming Framework for Robotics", Ph.D Paper, 2005.

저자소개

김 성 훈

1995년 광운대학교 전자공학과 졸업
1997년 광운대학교 전자공학과 석사
1996년-현 재 한국전자통신연구원 지능형로봇 연구단
주관심분야 분산시스템, 시스템S/W, 로봇S/W아키텍처

김 중 배

1986년 고려대학교 산업공학과
1988년 KAIST 산업공학과 석사
1988년-1991년 (주)대한항공 시스템부
1991년-현 재 한국전자통신연구원 지능형로봇 연구단
주관심분야 분산시스템, 시스템S/W, 로봇S/W아키텍처