4 족 보행로봇을 위한 소프트웨어 구조 설계
Design of Software Architecture for Quadruped Walking Robots

*이성훈 [1], #원대희 [1], 권오흥 [1], 백승호 [1]
*S. H. Lee[1], #Daehee Won(daehee@kitech.re.kr)[1] , O. H. Kwon[1], S. H. Baeg[1]
한국생산기술연구원 [1]

Key words : Quadruped Walking Robot, Software Architecture, Real-time Application, UML

1. 서론

로봇의 이용범위는 이미 산업용 로봇의 단순 반복작업에서 벗어나, 사람이 다루기 힘든 작업이나, 위험이 따르는 작업을 대신하는 극한작업용, 가사 노동의 대체 필요성과 사회의 고령화 및 복지 요구의 증대에 따른 개인 서비스용, 정밀한 수술이 요구되는 의료용, 특수 목적 및 군사용의 전문 서비스용 등 넓은 분야로 확대되고 있다. 이러한 역할이나 기능은 복잡한 로봇의 하드웨어 및 제어 소프트웨어의 구조를 수반하게 된다. 특히 다양한 센서 신호를 처리하고 각 관절의 액츄에이터를 실시간으로 통제해야 하는 제어 소프트웨어 구조의 중요성이 점점 부각되고 있다. 하지만, 기존 로봇 시스템에서는 한가지 태스크를 처리할 수 있는 구조로 개발되기 때문에 새로운 기능을 확장하기에 어려움이 있으며 코드의 작성과 이식에 많은 시간을 소비하고 있다. 따라서 로봇 소프트웨어 개발자들을 위한 로봇 모듈의 이식성과 호환성 그리고 재사용 성을 향상시키는 연구가 진행 되고 있다. [1]

통상 모바일 로봇은 그 운용되는 환경에 따라 여러 형태의 이동 메커니즘을 필요로 하게 된다. 이것은 마치 생태계에서 서식하는 동물들이 환경에 따라 진화된 다양한 형태의 이동방법이 취하고 있는 것과 같다. 지금까지 다양한 로봇 이동 메커니즘이 연구 및 제안되어 왔다.[2] 이 중에서 보행형 로봇은 지형이 일정하지 않은 곳에서 이동하기에 적합한 모델로 다리 수에 따라 2 족, 4 족형 보행로봇이 연구되고 있다. 하지만 족형 로봇은 다리 관절의 액츄에이터 제어에 필요한 연산, 로봇의 균형을 잡기 위한 자세 제어 연산 등 많은 양의 연산을 필요로 하여 로봇 운용에 많은 어려움을 보이고 있다. 이에 본 논문에서는 야외에서 이동이 유용한 4 족 보행 로봇을 개발함에 있어, 다연산 처리구조에 능동적이며, 이식성과 재사용성에 높은 효율성을 가지는 제어 소프트웨어 구조 설계 방법을 제시하고자 한다.

2 절에서는 제안하는 시스템의 소프트웨어 구조를 기술하며, 3 절에서는 UML 기반의 소프트웨어 엔지니어링에 관해 기술하였다. 그리고 4 절에서는 사용되는 하드웨어 구조에 대해 기술하며, 5 절에서 결론을 도출 하였다.

2. 다기능 통합 소프트웨어

이동형 로봇을 위한 다기능 통합 소프트웨어는 Fig. 1 과 같은 구조를 갖는다. 먼저 Task Controller Unit(TCU)는 Task Execute Layer(TEL), Behavior Execute Layer(BEL) 그리고 Hardware Abstract Layer(HAL)의 세 계층으로 나뉘게 된다. TEL 의 Task Manager 는 로봇 외부의 Operation Controller Unit(OCU)로부터 들어오는 Task command 를 Task Interpreter 를 통해 Task 를 해석하여 이를 실행할 수 있는 형태로 변환한다. Command Interpreter 는 주로 사용자의 상호작용을 통하여 Task 실행에 필요한 명령을 처리하는 모듈이고, Event Manager 는 로봇 상태의 변화를 감지하고 이를 Task Manager 에게 전달하여, Task 의 실행, 중지, 선택에 도움을 주며, 필요한 정보를 제공한다. BEL 의 High Level Behavior Manager 는 Patrolling, Localization, Path Planning 그리고 Navigation 등과 같은 고차원의 행위를 관리한다. 또한 HAL 은 Device Manager 를 통해 Sensor 의 정보를 얻게 되며, Behavior Controller Unit (BCU)로 센서의 상태를 전달 한다.

BCU 는 Behavior Execute Layer(BEL)와 Hardware Abstract Layer(HAL)로 구분된다. BCU 의 BEL 의 Behavior Manager 는 로봇의 보행 및 자세제어 등 실시간 제어가 필요한 낮은 차원의 행위를 관리한다. 이는 비정형화된 환경에서 로봇이 넘어지지 않고 보행하기 위해 균형을 2ms 이내로 자세제어, Local Controller Unit(LCU)와의 통신이 가능하도록 하였다. 또한 HAL 의 Device Manager 는 TCU 로부터 보행 패턴 및 제어 등에 필요한 알고리즘에서 연산된 결과를 받아 Behavior Manager 로 전달하고, 보행 및 자세제어 알고리즘 연산 후 LCU 로 전달하여 각 관절의 유압 서보시스템 제어 및 센서의 신호처리를 수행하도록 하며, 이 신호를 HAL 의 또 다른 Device Manager 로 전송하고 Sensor 와 Actuator, Power 의 현재 상태를 전달받는 역할을 한다.

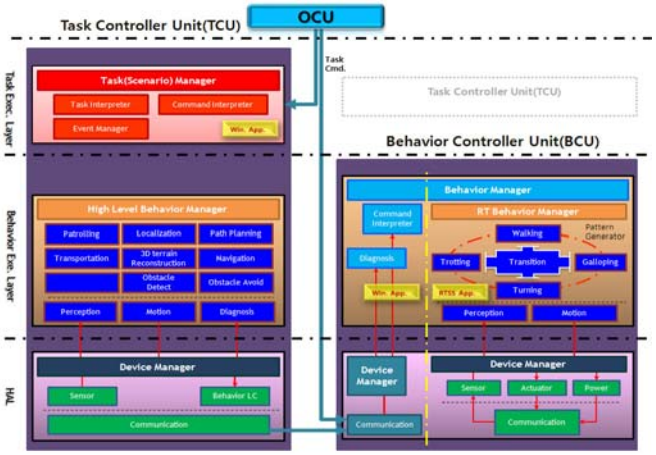

Fig. 1 Software Architecture of an Overall Control System

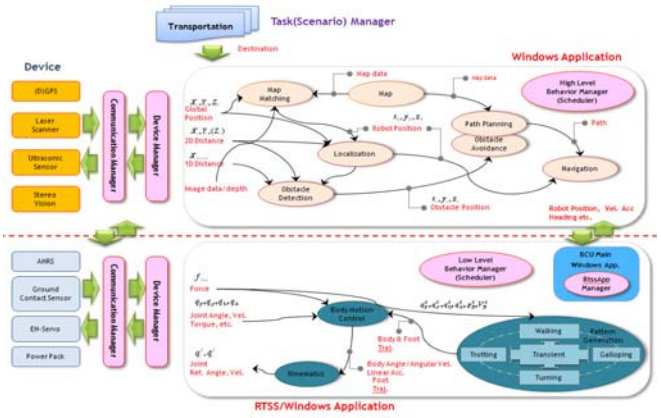

Fig. 2 Execution Flow Chart of the Transportation Task

URC-SA[1]에서는 TCU 는 High Level Manager 와 Device Manager 를 필요로 하지 않는다. 실제로 High Level Behavior Manager 는 BCU 의 Behavior Manager 에 포함 되어야 하며, TCU 의 Device Manager 또한 BCU 의 Device Manager 로 포함되어야 한다. 결과 적으로 Task Manager 만 포함된 TCU 가 Fig1 의 오른쪽 BCU 의 상위 TEL 로 귀속 되어야 한다.

그러나 TCU 에 High Level Behavior Manager 와 Device Manager 를 따로 생성한 이유는 TCU 의 최상위 알고리즘의 추가 및 확장이 용이하게 하고, BCU 는 로봇의 실시간성이 보장되어야 하는 자세제어, 보행 알고리즘 등을 수행하기 위해서이다. 이를 위해 BCU 에는 Real-time OS 를 탑재하였다. 이것은 Fig. 2 에 실행 예로 자세히 설명 할 수 있다. Windows Application 상에 고차원 행위 관리자는 (D)GPS, Laser Scanner, Ultrasonic Sensor, Stereo Vision 등의 장치와 BCU 의 Behavior Manager 로부터 온 로봇의 현재 상태 데이터를 이용하여 Obstacle Detection, Map Matching, Map Building, Path Planning, Navigation 을 통해 로봇의 진행 방향 및 속도 값을 BCU 의 Behavior Manager 에게 전달하게 된다.

BCU 의 RTSS/Windows Application 상에 Behavior Manager 는 Ground Contact Sensor, EH-Servo, Power Pack 등의 장치의 데이터 와 TCU 의 High Level Behavior Manager 로부터 받은 데이터를 이용하여 로봇의 자세 제어와 Kinematics 를 통해 각 Joint 제어를 명령을 하달하게 된다. 이렇게 높은 차원의 행위와 실시간성이 보장되어야 하는 행위를 이원화 함으로써, 로봇의 보행 안정성을 보장하고 앞서 서론에서 언급한 바와 같이 로봇의 이식성과 재사용성, 그리고 다연산 처리 구조에 능동적인 소프트웨어 구조 설계에 부합한다 할 수 있다.

3. UML 기반의 소프트웨어 엔지니어링

본 논문에서 제안한 소프트웨어 구조 기반의 로봇제어용 소프트웨어 구현 하였다. 그러나, 대부분의 로봇 소프트웨어가 개발 초기부터 UML 과 같은 소프트웨어 엔지니어링 기법을 적용하여 구현되지 않았기 때문에 이미 개발된 소프트웨어를 UML 기반의 소프트웨어 reverse 엔지니어링을 통해 UML 기반의 소프트웨어 개발 체계를 구축하였다. 본래는 Use Case View 에서부터 시작하여 Process View, Design View, Implementation View, Deployment View 등 5 단계의 Forward 엔지니어링[3] 절차가 필요하지만 앞에서 언급한 바와 같이 본 논문에서는 이미 개발된 소프트웨어에 5 단계의 절차를 역으로 적용한 S/W Reverse 엔지니어링을 수행하였다.

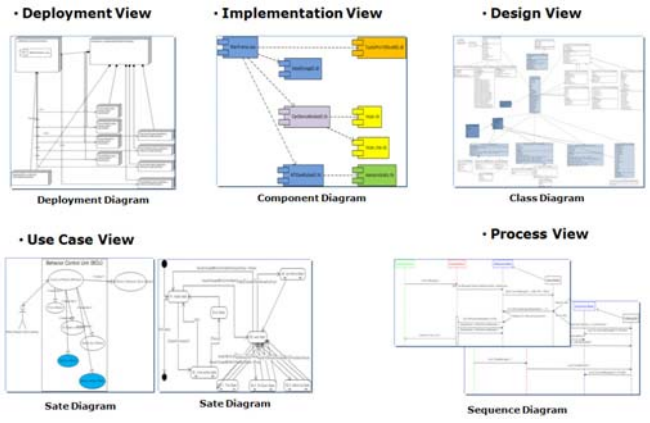

Fig. 3 Software architecture based on UML

또한 이렇게 개발된 소프트웨어의 효율적인 수정 및 보완과 유지보수 비용을 최소화하기 위해서는 주기적인 Forward/Reverse 엔지니어링을 혼합한 Round Trip 방식의 지속적인 소프트웨어 엔지니어링이 필요하다. [4]

4. 제어시스템 하드웨어 구조

본 논문에서 사용되는 로봇제어 시스템의 하드웨어 구조는 Fig 4 과 같다. Control System 은 Task Controller Unit(TCU)와 Behavior Controller Unit(BCU) 그리고 Local Controller Unit(LCU)로 구성되어 있다. TCU 와 BCU 는 각각 소형 PC 를 사용하였는데, 이것은 앞서 언급한 소프트웨어 구조 및 소프트웨어의 실행 성능 향상을 위해 높은 차원의 행위를 위한 TCU 와 자세제어에 실시간성이 보장되어야 하는 BCU 를 하드웨어 적으로 분리하여 설계한 것이다. TCU 와 BCU 간에는 TCP/IP 통신을 하며 BCU 와 LCU 사이에는 CAN 통신을 사용한 분산제어 방식을 사용 한다. LCU 소형 DSP 로 구성되어 있으며 관절 제어에 필요한 서보밸브 및 센서 앰프 등이 모두 탑재되어 있다.[5]

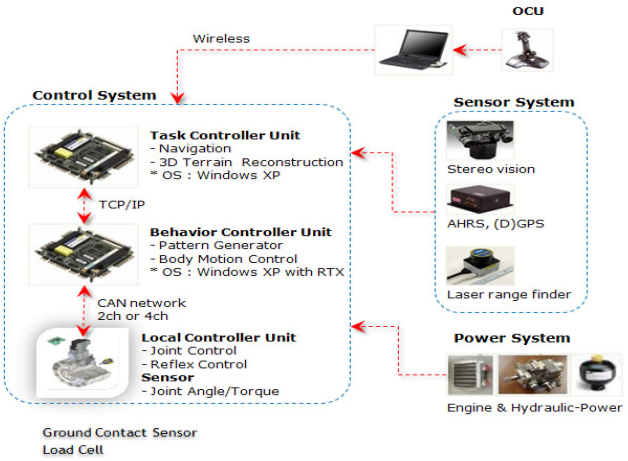

Fig. 4 Control System Hardware architecture

5. 결론

본 논문에서는 Task 단위의 최상위 알고리즘과 Behavior 및 Local 단위의 알고리즘이 하나의 H/W 플랫폼에서 실행될 때 보장하기 어려운 제어기의 실시간성 및 확장성, 재사용성 등을 보다 쉽게 확보할 수 있도록 TCU, BCU, LCU 등 3 단계로 구성된 분산 제어시스템 H/W 구조와 이에 적합한 S/W 구조를 제안하였다. 3 단계 각각의 제어유닛은 기능별로 그 역할이 명확히 구분되어 있기에 독립적으로 개발이 용이하다. 뿐만 아니라 TCU, BCU 에서의 S/W 개발시 체계화된 개발 프로세스를 구축하기 위해 UML 기반의 S/W 엔지니어링 기법을 도입하여 개발된 S/W 의 Reverse 엔지니어링 수행하였으며 이를 통해 얻은 다양한 문서 결과물 활용으로 S/W 개발기간을 단축하고 재사용성 및 확장성을 높일 수 있는 효율적인 S/W 개발 체계를 구축하였다.

향후 연구과제로서는 네트워크 기반 분산 제어 시스템의 실시간 통신방법 및 성능개선과 Round Trip 방식의 S/W 엔지니어링 기법을 보다 체계적으로 적용하는 연구가 필요하다.

참고문헌

1. 이승익, 장철수, 정승욱, 김중배, “로봇 소프트웨어 아키텍처의 연구동향과 현황,” [ETRI]전자통신동향분석, 제 20 권 제 2 호, pp. 1-13, 2005. 4.
2. 한창수, “보행형 로봇과 이동 메커니즘,” 대한기계학회 학술저널, 제 48 권 제 9 호, 2008. 9.
3. Kruchten, Philippe, “Architectural Blueprints — The “4+1” View Model of Software Architecture.” IEEE Software 12 (6), pp. 42-50., November 1995
4. E. J. Chikofsky and J. H. Cross, II, “Reverse Engineering and Design Recovery: A Taxonomy,” IEEE Software, vol. 7, no. 1, pp. 13-17, January 1990.
5. B.R. So, T.J. Kim, D.H. Won, O.H. Kwon, S.D. Park, and W.H. Son, “Development of Quadruped Walking Robot Using A Hydraulic Rotary Actuator”, 39th International Symposium on Robotics 2008, October 15~17, 2008.