논문 2017-12-34

수중로봇 소프트웨어 시스템의 요구사항을 반영한 ROS 기반의 계층화된 소프트웨어 아키텍처의 설계

(Design of Layered Software Architecture Based on ROS That Reflects the Requirements of Underwater Robot Software System)

이 정 우, 최 영 호, 이 종 득, 윤 성 조, 서 진 호*
(Jung-Woo Lee, Young-Ho Choi, Jong-Deuk Lee, Sung-Jo Yun, Jin-Ho Suh)

Abstract : Underwater robots operating in constrained underwater environment have requirements for software systems. Firstly, it is necessary to provide reusable common software components for hardware interface of sensors and actuators that are frequently used in underwater robots. Secondly, it is required to support distributed execution environment on multiple embedded controllers. Thirdly, it is need to implement a monitoring system capable of high-speed and large-data transmission for underwater robots operating in an environment where it is difficult to check the robot status. For these requirements, we have designed the layered architecture pattern and applied several design patterns to enhance the reusability and the maintainability of software components, In addition, we overlaid the broker architecture pattern to support distributed execution environments. Finally, we implemented the underwater robot software system using ROS framework based on the software architecture design. In order to evaluate the performance of the implemented software system, we performed an experiment to measure the response time between components and the transmission rate of the monitoring data, and obtained the results satisfying the required performance.

Keywords : Underwater robot software system, ROS, Layered architecture, Broker architecture, Software design pattern

Ⅰ. 서 론

수중로봇은 해양과 호수 등에서 수중 탐사와 작업을 위해 사용되며 추진기 (Thruster)를 사용하여 수중을 유영하거나 트랙 등으로 해저면에 착저하여 이동한다. 따라서 수중환경 내에서 운용되어지는 수중로봇은 그 운영 방법에 따라 ROV (Remotely Operated Vehicle)와 AUV (Autonomous Underwater Vehicle)로 구분할 수 있으며, ROV는 전원과 통신이 유선의 테더 케이블로 수상 제어부와 연결되어 수동 또는 반자동으로 원격 조정되어 운용되고, AUV는 무선으로 자체 배터리를 동력원으로 하여 미션 중심의 자동화된 프로그램에 의해 운용된다.

수중로봇은 수중이라는 운용 환경으로 인하여 이동을 위해 필요한 환경 측정 센서의 종류가 제약되어지고, 내압 구조의 기구부 구성으로 인하여 전장부를 내장할 수 있는 실제적 공간이 부족하여 고성능의 제어기와 고용량의 배터리를 적용하기 어려운 것이 실제적 환경이기에 실제적 수중로봇 개발을 진행시에는 개발목적에 따라 최적화된 임베디드

*Corresponding Author (suhgang@kiro.re.kr)
Received: July 31 2017, Revised: Aug. 30 2017, Accepted: Sep. 1 2017.
J.W. Lee, Y.H. Choi, J.D. Lee, S.J. Yun, J.H. Suh: Korea Institute of Robot and Convergence
※ 본 연구는 산업통상자원부 기술혁신사업 (로봇산업원천기술 개발 사업, No. 10043928)의 지원으로 수행되었음.

ⓒ IEMEK J. Embed. Sys. Appl. 2017 Oct. 12(5) 303-310
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2017.12.5.303

제어기와 배터리 팩을 설계하여 사용하게 된다.

기존에는 단순 목적의 수행을 위한 최소한의 센서와 제어 알고리즘의 실행이 요구되어 별도의 아키텍처 없이 단일 또는 복수의 프로그램의 조합으로 소프트웨어가 구성되었으나, 최근에는 다중 목적을 수행하기 위해 수중로봇에 다양한 센서가 장착되고 처리 알고리즘이 고도화되어 내부의 임베디드 처리기의 사양이 높아지는 추세이며 다중 복합된 미션의 수행과 이를 위한 수중로봇의 효율적인 운용이 요구되면서 기존의 단순 기능 구현 중심의 프로그램 개발이 아닌 수중로봇에서 요구되는 사항들을 대응할 수 있는 체계적인 소프트웨어의 설계와 구현이 필요하다.

수중로봇은 수중 환경에서 동작하기 때문에 육안으로는 상태를 확인할 수 없으므로 자체 진단과 상태 모니터링이 중요하며, 고장이나 저전력 상태 등 예외적인 상황 발생을 감시 및 감지하여 수상으로 복귀하기 위한 처리가 우선되어야 한다. 수중 환경은 여러 조건들의 변화 폭이 커 실제 현장에 투입하여 운용 시에는 소프트웨어에 대한 가변 조정이 필요하며, 소프트웨어를 구조화하여 유지보수가 쉽도록 하여야 한다.

수중로봇은 그림 1과 같이 RS232/485, CAN, Ethernet 등의 다양한 통신 방식을 사용하는 센서들을 장착하며 복수의 제어기 내에서 영상센서 또는 소나센서 데이터를 분산 처리하거나 자율주행 이동을 위한 알고리즘을 실행하고, 목적으로 하는 다양한 미션을 수행하여야 한다. 이를 위해서 통신과 제어, 연산처리 알고리즘 등의 요소들을 로봇 플랫폼의 종속적 또는 비종속적인 컴포넌트 단위로 분류 및 정의하여 다양한 미션들에 대해 재사용성을 확보하며, 분산된 하드웨어 실행 환경을 대응하고 제약된 운용 환경에서의 유지보수성과 안정성을 위한 소프트웨어 아키텍처 설계와, ROS 프레임워크를 사용하는 소프트웨어 시스템의 구현을 제안하고자 한다.

본 논문은 다음과 같이 구성된다. 2장에서는 수중로봇에서 소프트웨어 시스템에 요구되는 사항들에 대응하기 위한 아키텍처를 설계 방법을 제안하고, 3장에서는 소프트웨어 시스템의 구현을 위한 ROS 프레임워크의 적용 및 컴포넌트 단위의 디자인 패턴을 설명한다. 결론에서는 실제 소프트웨어 시스템의 구현에 대한 컴포넌트 간의 통신 속도 및 서비스 호출의 안정도 등의 성능 평가를 수행하여 요구 사항을 만족하는 설계가 이루어졌는지를 기술한다.

그림 1. 수중로봇에 사용되는 주요 센서 예시
Fig. 1 Examples of major sensors used in underwater robots

Ⅱ. 요구사항 분석 및 아키텍처 설계

소프트웨어 시스템의 아키텍처는 여러 소프트웨어 요소들을 구성하고 추상화된 관계로 표현한다. 소프트웨어 아키텍처는 적용 대상이 되는 소프트웨어 시스템에서 덜 유용한 정보나 세부 사항들을 제외하고 시스템의 특성에 맞는 요소들을 중심으로 관계를 명확하게 위한 목적으로 설계를 수행하게 된다. 따라서 수중로봇에서 요구되는 사항을 분석하고 이를 효과적으로 대응할 수 있는 소프트웨어 아키텍처의 설계가 필요하다.

수중로봇이 운용되는 수중 환경은 로봇의 자율 이동을 위한 환경 측정 센서의 종류에 제약이 많아 초음파를 사용하는 소나 센서와 관성에 의한 모션 측정 센서가 주를 이룬다. 또한 이동을 위한 수중 추진기 등 구동부의 경우에도 제어 방식은 유사한 경우가 많다. 로봇의 목적과 운용 방식에 따라 여러 형태의 수중로봇이 있으나 자율 이동과 관련한 센서류와 구동부는 한정되므로, 수중로봇을 위한 소프트웨어 시스템에서는 이를 공통 요소로서 활용할 수 있도록 설계하는 것이 필요하다. 각 센서와 구동부별로 통신을 하기 위한 물리적 채널의 하드웨어 자원을 점유 및 관리하는 소프트웨어 요소들을 배치하여 센서 데이터와 상태 정보를 소프트웨어 시스템 내부로 출력하며, 이는 상위의 제어 및 데이터 처리 알고리즘에 대한 소프트웨어 요소들의 입력으로서 사용되므로, 소프트웨어 아키텍처의 설계에서 주로 사용되는 여러 패턴들 중 [1, 2], 그림 2의 계층적 구조 패턴 (Layered Architecture Pattern)

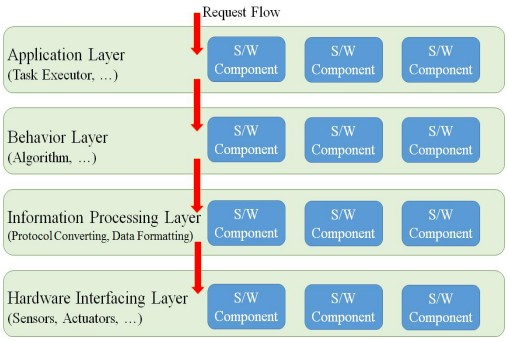

그림 2. 계층적 구조
Fig. 2 Layered architecture

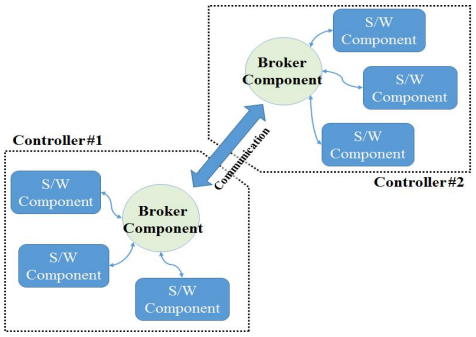

그림 3. 중개자 구조
Fig. 3 Broker architecture

[1, 3, 4, 5]을 적용하는 것이 적절할 것으로 판단된다. 계층적 구조 내에서 센서와 구동부 등의 장치와 알고리즘을 각 소프트웨어 요소로 구분하여 객체로서 입출력 관계를 정의할 수 있으며, 여러 소프트웨어 요소들은 기능적 범주의 계층별로 배치되며, 한 계층의 하단 또는 상단의 계층에 있는 소프트웨어 요소들과만 상호작용을 하게 된다. 따라서 소프트웨어 요소의 입출력 인터페이스에 대한 논리적인 계층적 기술이 되어 개발자 간의 이해도를 높이고 유지보수성을 향상시킨다.

수중 환경에서 로봇이 동작하기 위해서는 전장부에 대한 방수와 내압 성능을 가져야 하며, 내압에 유리한 원통형의 기구 설계가 주를 이룬다. 그러나 이러한 설계는 공간 사용의 효율성이 떨어져 제어기와 배터리의 실장을 어렵게 한다. 내압 방수 기구부의 형상에 맞춰 전장부를 설계 제작하여 사용하기도 하나, 고장이나 사양 변경 등에 대한 보수성이 떨어지고 비용이 증가하게 된다. 따라서 공간 활용성과 보수성, 비용을 종합하여 적정한 성능 수준의 소형 제어기와 배터리를 조합하여 실장하게 되며, 처리 성능을 확보하기 위해서 다수의 내압 방수 기구부 내에 분산된 제어기가 배치되기도 한다.

따라서 소프트웨어 시스템의 구조 설계 시 분산된 소프트웨어 실행 환경에 대응할 수 있도록 하여야 한다. 분산 환경에서의 소프트웨어 요소들은 모듈화 되어야하며 제어기 간의 데이터 전송이 가능해야 하는 요구사항을 충족시키기 위해, 여러 소프트웨어 아키텍처 설계 패턴 중 [1, 2], 그림 3의 중개자 구조 패턴 (Broker Architecture Pattern) [1, 3]을 적용하여 한 제어기 내의 모든 소프트웨어 요소들이 중개자 역할을 하는 모듈을 통해 다른 제어기의 소프트웨어 요소들과 통신할 수 있도록 설계하였다. 실제 소프트웨어 시스템의 실행에서는 중개자에 해당하는 모듈을 통해 제어기 간의 데이터 통신이 이루어지므로 기능 정지 등의 상황이 발생하지 않도록 구현 및 검증을 수행하여야 한다.

육상과 달리 수중 환경에서 동작하는 수중로봇은 동작 상태를 직접적으로 관찰하는 것이 어렵다. 해상에서는 탁도나 광량 부족으로 인해 영상 카메라를 통해서도 로봇의 상태를 확인하기가 어렵다. 그러므로 수중로봇의 각 부별 상태 확인과 제어 및 데이터 처리에 대한 알고리즘의 실행 상태를 소프트웨어적으로 모니터링하는 것이 육상 환경에서 동작하는 로봇에 비해 상대적으로 중요하다.

또한 기계적 고장이나 전장부의 누수 발생, 제어 실패로 인한 운용 범위를 벗어나는 등의 상황들이 발생하였을 때 운영자가 인지하고 대처하는데 시간이 소요되어, 이를 자동화하여 즉시 처리할 수 있는 체계의 구현이 필요하다. 따라서 소프트웨어 시스템의 여러 요소들에 대해 모니터링이 가능하여야 하며, 소프트웨어 요소들의 내부 통신을 방해하지 않도록 별도의 통신 채널을 확보할 수 있도록 설계하여야 한다. 또한 모니터링 된 상태에 따른 룰(Rule) 기반의 동작 실행 기능이 구현되어야 하며, 영상 등의 대용량 데이터 전송이 필요한 경우에도 대응할 수 있도록 하여야 한다. 모니터링을 위한 통신 채널로서 표준 프로토콜인 XMLRPC 등을 적용하여 간단한 제어 명령의 송신을 포함하여 대용량 데이터 수신이 가능하도록 설계하는 것이 필요하다. 표준 프로토콜을 사용하면 개발자 간의 이해도를 높여 설계 이후의 구현에서도 오류의 가능성을 줄이며 유지보수성 또한 높이게 된다.

이상과 같이 수중로봇의 소프트웨어 시스템에서 요구되는 사항들과 이를 대응하기 위해 제안하는 소프트웨어 아키텍처의 설계는 다음 표 1과 같다.

표 1. 수중로봇 소프트웨어 시스템의 요구사항과 대응 아키텍처 설계
Table 1. Requirements of underwater robot software system and corresponding architecture design

Software System Requirements	Corresponding Architecture Design
Underwater Robots are limited in the kinds of sensors and actuators used, so common elements need to be developed	Designed an architecture with the layered architecture pattern that can separate hardware -dependent elements and non-dependent elements for common use in various underwater robot platforms
Due to the shape of the waterproof pressure-resistant mechanism, it is difficult to apply a high-performance single controller. Using multiple decentralized mid-performance controllers requires a mutual communication scheme	In order to execute the software system in a distributed environment, the broker architecture pattern is applied to data transmission between controllers through brokers
High-speed monitoring (over 30Hz) is important because it is difficult to check underwater robot status with camera image or naked eye	Each software component implements functions for monitoring. the implementation of monitoring is designed to apply the observer design pattern and transmit information to the mothership control station via a separate communication channel using the XMLRPC protocol

수중로봇 소프트웨어 시스템의 기본 아키텍처는 계층적 구조 (Layed Architecture)를 적용하고, 분산된 제어기 상에서의 소프트웨어 실행을 위해 상위 계층의 소프트웨어 요소들에 대한 중개자 구조 (Broker Architecture)를 적용한다.

최하단의 하드웨어 통신 계층 (Hardware Interface Layer)은 센서류와 구동부 등 특정한 하드웨어 종속적인 통신을 담당하는 소프트웨어 요소들이 배치되며 저수준의 제어 명령을 송신하고 데이터를 수신하여 상위 계층으로 전달하는 역할을 한다. 이 계층의 소프트웨어 요소들은 하드웨어 자원을 점유하게 되므로 추가적으로 싱글톤 디자인 패턴을 적용하여 그 역할을 명확히 하도록 구현한다.

상위의 정보 처리 계층 (Information Processing Layer)에 배치된 소프트웨어 요소들은 각종 데이터를 가공 및 변환하여 출력하게 되고, 행동자 계층 (Behavior Layer)의 소프트웨어 요소들은 자율 주행 알고리즘이나 영상 처리 알고리즘 등 수중로봇의 목적별 기능이 구현된다.

최상위의 응용 계층 (Application Layer)은 수중 환경 탐사나 수중 구조물의 검사 및 작업 등 운용 목적에 대한 미션을 수행하도록 하는 소프트웨어 요소가 배치되며 하위의 행동자 계층과 연동하여 수중로봇을 운용하게 된다.

Ⅲ. 소프트웨어 시스템의 구현

수중로봇의 소프트웨어 시스템 아키텍처 설계를 바탕으로 실제 구현을 위해서는 프레임워크를 선택하여 적용하여야 한다. 로봇 시스템을 위한 소프트웨어 프레임워크로는 여러 가지가 있으며 [6, 7], 기존에는 마이크로소프트의 MSRDS (Microsoft Robotics Developer Studio)와 Evolution Robotics사의 ERSP (Evolution Robotics Software Platform), 유럽 주도의 오픈소스 프로젝트인 OROCOS (Open RObot Control Software) [8], 일본 AIST의 OpenRTM (Open RT Middleware) 등이 있고, 국내에서는 정부 주도하에 개발된 OPRoS (Open Platform for Robotic Service) [9, 10]가 있다. 최근에는 오픈 소스 프로젝트인 ROS (Robot Operating System) [11] 프레임워크가 사실 상의 표준 (De Facto Standard)로 사용되고 있다.

MSRDS는 Microsoft사에서 제작한 로보틱스 개발용 툴이며, 분산 실행 환경을 제공하고 닷넷 프레임워크의 C# 언어와 스크립트 형식의 언어를 자체 지원하여 GUI 상에서 쉽게 통합하고 실행할 수 있다. 주요 목적이 교육과 시뮬레이션 위주로 사용되어 로봇에 내장하여 단독으로 실행하기 위한 환경은 잘 갖추어져 있지 않다.

ERSP는 계층적 구조 설계가 반영된 프레임워크로 영상 기반 SLAM 알고리즘을 중심으로 계층이 형성되어 있고, 소니 AIBO나 ZMP 등 로봇 플랫폼의 하드웨어 인터페이스 계층을 내장하고 있다. 그러나 분산 실행 환경을 반영하지 못하고 시뮬레이터나 검사 (Inspection)를 위한 툴 등이 없어 개발이 어려운 측면이 있다.

OROCOS는 로봇의 주행이나 영상 인식 등 기능

에 대한 알고리즘을 라이브러리로 제공하는데 초점이 맞춰져 있으며, 전체 아키텍처를 제시하지는 않고 소프트웨어 컴포넌트 단위의 디자인 템플릿을 정의하고 있다.

OpenRTM은 CORBA (Common Object Request Broker Architecture) 기반으로 분산 실행 환경에서 컴포넌트를 표준화한 아키텍처를 제시하고 있으며, RTC (RT Component)를 국제 표준으로 제정하였다. 그러나 컴포넌트에 대한 실행 환경만 있고 개발 환경 등의 툴은 제공하지 않아 컴포넌트 개발과 구현, 연동 실행에 어려움이 있다.

ROS는 중개자 구조 (Broker Architecture)를 적용하여 'roscore'라는 중개자를 통해 분산 실행 환경을 지원하고 'node'로 표현되는 소프트웨어 요소들에 대해 객체 이름 서비스 (Naming Service)를 제공하여 다른 제어기에 위치한 소프트웨어 요소의 이름만으로도 메시지를 송수신하고 서비스를 호출할 수 있다. 'topic'으로 표현되는 메시지 전송은 발행자-구독자 디자인 패턴 (Publisher-Subscriber Design Pattern)이 적용되어 데이터의 송수신을 수행한다. 이러한 미들웨어 구현을 기반으로 방대한 커뮤니티에서 생성된 로봇 관련 알고리즘과 여러 센서 및 구동부 인터페이스 등을 라이브러리로 제공하며 실행 환경 외에도 개발 환경에 대한 도구 집합을 제공하고 있어 개발과 유지보수가 편리한 장점이 있다.

따라서 수중로봇 소프트웨어 시스템의 아키텍처를 구현하는데 있어서, 여러 소프트웨어 프레임워크 중에서 ROS 프레임워크를 선택하여 적용하도록 하며, 논리적인 계층적 구조에 소프트웨어 컴포넌트들을 배치하고 중개자 구조에 대해서는 ROS 프레임워크를 이용하도록 하였다. 그림 4에서 소프트웨어 시스템의 컴포넌트들의 배치를 보이고 있다.

대상으로 하는 수중로봇은 ROV 및 AUV 모드로 운용이 가능한 형태로, 이동을 위한 수중 스러스터 6기와 수중 환경 내 위치 인식을 위한 관성 센서 및 이미지 소나 센서, 광학 카메라 센서, 알티미터, 깊이 센서 등을 장착하고 있다 [12]. 수중로봇의 방수 내압 기구부 내에는 이동 제어 및 알고리즘 실행을 위한 제어기가 2조로 구성되어 있으며, 본 수중로봇의 소프트웨어 시스템은 분산된 환경 내에서 실행이 필요하다.

기 설계한 내용에 따라 계층적 구조에 소프트웨어 컴포넌트들을 배치하였고, 제어기 간의 상위 계층끼리는 ROS 프레임워크의 중개자 구조를 사용하여 데이터를 송수신하도록 하고 있다.

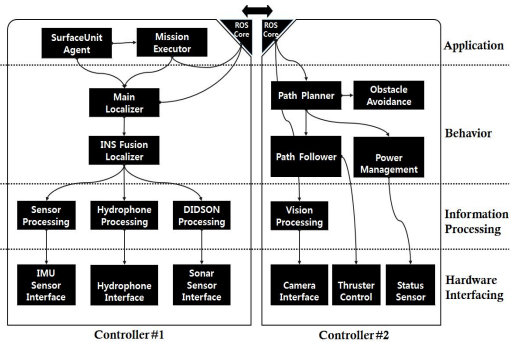

그림 4. 수중로봇 소프트웨어 시스템의 컴포넌트 배치도

Fig. 4 Component layout of underwater robot software system

하드웨어 통신 계층에는 센서 및 구동부가 물리 통신 채널을 통해 제어기별로 연결되며, 해당 물리 장치와의 통신을 담당하는 소프트웨어 컴포넌트들이 종속적으로 배치된다. 이 계층의 소프트웨어 컴포넌트들은 하드웨어 자원을 점유하여 사용하며 중복된 자원의 접근을 제한하여야 하므로, 중복된 인스턴스의 실행을 막고 단일 접속점을 제공하는 싱글톤 디자인 패턴 (Singleton Design Pattern)을 적용하여 구현한다.

또한 컴포넌트 내부의 구현을 위해 MVC 디자인 패턴 (Model-View-Controller Design Pattern) [13]을 사용하였으며, 이는 컴포넌트의 자료 구조와 데이터 및 상태를 유지 관리하는 역할의 모델과, 모델의 상태를 UI 등으로 표현하는 역할을 담당하는 뷰, 그리고 모델과 뷰를 컴포넌트 외부와 인터페이스하기 위한 역할의 컨트롤러로서 소프트웨어 컴포넌트의 내부 구조를 디자인한다. 이러한 역할의 구분을 두어 구현을 하게 되면 개발자 간의 이해도를 향상시키고 유지 관리도 쉬워진다. MVC 디자인 패턴은 모든 계층의 소프트웨어 컴포넌트 설계 및 구현에 적용되어 있다. 그림 5는 싱글톤 패턴에 대한 의사 코드이다.

상위의 정보 처리 계층은 관성 센서와 이미지 소나 센서 등의 저수준 데이터를 활용 가능한 정보 형태로 연산하여 오차를 필터링한 센서값이나 소나 이미지 형태 등으로 자료를 생성한다.

행동자 계층은 영상 데이터 기반의 위치 인식 알고리즘이나 관성 센서에 의한 항법 알고리즘, 장애물 회피 알고리즘, 경로 생성 알고리즘 등의 기능이 구현된다.

```
Class TemplateComponent
{
    public:
        static TemplateComponent& getInstance()
        {
            static TemplateComponent instance;
            return instance;
        }

        static std::unique_ptr<TemplateComponent_Model>& Model()
        {
            return TemplateComponent::getInstance().pModel;
        }

        static std::unique_ptr<TemplateComponent_View>& View()
        {
            return TemplateComponent::getInstance().pView;
        }

        static std::unique_ptr<TemplateComponent_Controller>&
                                                    Controller();
        {
            return TemplateComponent::getInstance().pController;
        }

    protected:
        TemplateComponent();
        ~TemplateComponent();

    private:
        std::unique_ptr<TemplateComponent_Model> pModel;
        std::unique_ptr<TemplateComponent_View> pView;
        std::unique_ptr<TemplateComponent_Controller> pController;
};
```

그림 5. 소프트웨어 컴포넌트의 싱글톤 디자인 패턴 적용을 위한 템플릿 코드

Fig. 5 Template code for applying a software component's singleton design pattern

최상위의 응용 계층에서는 ROV 모드 운용 시 수상 제어부에서 전달되는 세부 제어 명령이나 모니터링 정보를 송수신하도록 하고, AUV 모드 운용 시 계획된 미션 수행을 전달받아 룰 기반으로 행동자 계층의 소프트웨어 컴포넌트를 조합 실행하도록 구현된다.

수중로봇의 제어기에 구현된 모든 소프트웨어 컴포넌트는 MVC 디자인 패턴에 의한 컨트롤러 구현부에 옵저버 디자인 패턴 (Observer Design Pattern)을 적용하여 수상 관제부의 모니터링 서버와 연결되어 센서데이터나 알고리즘의 실행 상태 등의 정보를 송신하고 있으며, XMLRPC 프로토콜에 의한 별도의 통신 채널 인터페이스를 구현함으로써 ROS 기반의 로봇 시스템의 실행에 영향을 주지 않고 모니터링을 수행 할 수 있다.

ROS 프레임워크를 바탕으로 전체 소프트웨어 컴포넌트를 구현한 결과는 그림 6과 같다.

Ⅳ. 결 론

수중로봇 소프트웨어 시스템은 요구되는 사항들을 분석하여 공통된 구성 요소의 재활용성과 분산 제어기 간의 통신 체계의 필요성을 만족하기 위하여, 계층적 구조와 중개자 구조를 반영하여 소프트웨어 아키텍처를 설계하였고, ROS 프레임워크 기반으로 실제 소프트웨어 요소들을 구현하였다.

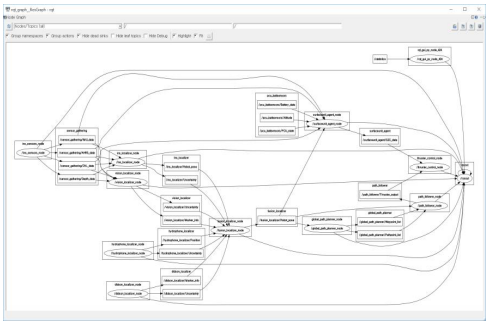

그림 6. ROS 프레임워크 기반 소프트웨어 시스템 구현

Fig. 6 Implementation of a software system based on ROS framework

또한 여러 소프트웨어 요소들의 모니터링을 구현하기 위한 XMLRPC 프로토콜 기반의 별도 통신 채널의 구현에 대한 성능 측정을 위해 정량적으로 측정할 수 있는 응답 속도와 모니터링 데이터의 대용량 전송 속도를 시험을 통해 구하였다.

응답 속도는 두 제어기 사이에 소프트웨어 요소 간의 서비스 호출 시점에서부터 서비스를 실행하여 응답을 받기까지의 시간차를 구하였으며, 그림 7과 같이 평균 850ms 소요되어 1KHz 이상의 제어 네트워크 주기를 가지는 것으로 측정되었고 대상으로 하는 수중로봇에서 필요로 하는 제어 주기를 만족하였다.

모니터링 데이터 전송 속도를 측정하기 위해서, 실해역 내에서 수상의 관제부와 수중로봇의 제어기

Evaluation Items	Unit	Target	Measure	Satisfaction
Networking Cycle between H/W Modules	Hz	1000	1176	O

Num	BeginTime(us)	EndTime(us)	ElapsedTime(us)	Fail(0)OrSuccess(1)
1	-236850811	-236849242	619	1
2	-236825484	-236823954	603	1
3	-236800074	-236798460	637	1
4	-236774553	-236772938	637	1
5	-236749404	-236747867	606	1
6	-236724079	-236722646	565	1
7	-236698725	-236697096	643	1
8	-236673375	-236671637	686	1
9	-236648042	-236646824	480	1
10	-236622731	-236621326	554	1
11	-236597383	-236596228	455	1
12	-236572044	-236570826	480	1
13	-236546731	-236545169	616	1
14	-236521331	-236520021	517	1
15	-236495964	-236494312	652	1

그림 7. 소프트웨어 요소 간 서비스 호출 시 응답속도 측정

Fig. 7 Measuring response time when calling service between components

Local Write (ms)		Number of Consecutive Transmissions								
		10	20	30	50	70	100	150	200	300
Transmit Data Size (KB)	1K	3.9	2.9	1.56	1.32	1.26	1.11	1.11	1.08	1.06
	2K	2.5	1.5	1.47	1.24	1.23	1.12	1.1	1.05	1.05
	4K	2.7	1.5	1.47	1.34	1.17	1.14	1.08	1.06	1.06
	16K	2.6	1.9	1.27	1.32	1.14	1.1	1.12	1.07	1.03
	512K	4.3	4.5	4.07	3.8	3.3	3.56	3.77	3.48	3.65
	1024K	8.1	7.1	6.77	6.58	6.26	6.08	5.96	6.19	6.2
	2048K	14.2	13.7	13.1	12.9	12.9	12.8	12.8	12.7	12.5
Remote Write (ms)		Number of Consecutive Transmissions								
		10	20	30	50	70	100	150	200	300
Transmit Data Size (KB)	1K	4.9	4.9	4.63	5.0	4.5	4.12	4.4	3.9	4.03
	2K	6.0	4.7	4.73	4.12	4.23	4.65	4.6	4.21	4.53
	4K	6.0	4.95	4.37	4.70	4.59	4.6	4.61	3.99	4.53
	16K	6.7	6.05	5.73	5.78	5.61	5.72	5.63	5.63	5.64
	512K	6.9	6.5	6.5	6.2	6.2	6.44	6.3	6.24	6.27
	1024K	12.9	12.05	12.07	11.8	12.08	11.89	11.8	11.8	13.28
	2048K	23.1	23.2	22.4	22.42	22.56	22.44	27.27	28.94	26.88

그림 8. 모니터링 데이터 전송 속도 측정 ((상)두 제어기 내, (하)한 제어기와 수상 제어부 간 전송시)

Fig. 8 Measuring transmission rate of monitoring data

간에 XMLRPC 프로토콜에 의한 전송 데이터 크기별로 연속 전송 시 소요된 시간을 평균하여 구하였다. 그림 8과 같이 한 제어기 내에서는 영상 데이터에 해당하는 초당 2메가바이트로 전송 시에 최대 14.2 밀리세컨드로 측정되었으며, 일반적인 상태 데이터에 해당하는 초당 2킬로바이트로 전송 시에 최대 2.5 밀리세컨드로 측정되었다. 한 제어기와 수상 제어부 간의 전송에서는, 초당 2메가바이트 전송 시 최대 29 밀리세컨드가 소요되었고, 초당 2킬로바이트 전송 시에는 최대 6 밀리세컨드가 소요되었다. 4킬로바이트 내외의 모니터링을 위한 데이터의 전송은 100Hz 이상으로 가능하고, 영상 등 대용량 데이터 전송은 약 30Hz 전후로 가능하여 수중로봇 소프트웨어 시스템에서 요구되는 모니터링과 관련한 기능 구현이 적정하였음을 확인하였다.

수중로봇의 소프트웨어 시스템에 요구되는 사항들을 분석하고 이에 대응하는 아키텍처를 적용하여 ROS 프레임워크 기반으로 구현한 결과, 하드웨어 종속적인 소프트웨어 컴포넌트와 비종속적인 컴포넌트를 분리하여 개발함으로서 재사용성을 높였고, 복수의 제어기로 구성된 분산 실행 환경에 대응하도록 하였으며, 향후 유지보수성을 높이기 위한 여러 디자인 패턴을 소프트웨어 개발에 적용하였다. 구현 후에는 수중로봇에서 요구되는 응답 속도와 고속 모니터링을 위한 전송 속도의 성능 향상을 위해 최적화를 수행하여 요구 성능을 만족하였다. 향후에는 실해역에서의 운용 결과를 바탕으로 개별 소프트웨어 컴포넌트의 안정화를 수행할 예정이다.

References

[1] L. Bass, R. Kazman, P. Clements, “Software Architecture in Practice,” 2012.

[2] List of Software Architecture Styles and Patterns, Available at: https://en.wikipedia.org/wiki/List_of_software_architecture_styles_and_patterns

[3] F. Buschmann, R. Meunier, H. Rohnert, “Pattern-Oriented Software Architecture,” Vol. 1, 1996.

[4] H. Durrant-Whyte, N. Roy, P. Abbeel, “Robotics: Science and Systems VII,” pp. 90-91, 2012.

[5] T.H. Kim, H.C. Kim, “A Software Architecture for Highly Reconfigurable Sensor Operaing Systems,” IEMEK J. Embed. Sys. Appl., Vol. 2, No. 4, pp. 242-250, 2007 (in Korean).

[6] B.W. Choi, “A Review and Outlook of Robotic Software Frameworks,” The Journal of Korea Robotics Society, Vol. 5, No. 2, pp. 169-176, 2010 (in Korean).

[7] N. Mohamed, J. Al-Jaroodi, I. Jawhar, “Middleware for Robotics: A Survey,” IEEE Conference on Robotics, Automation and Mechatronics, pp. 736-742, 2008.

[8] J. Yuh, “Design and Control of Autonomous Underwater Robots: A Survey,” Autonomous Robots, pp. 7-24, 2000.

[9] OPRoS Wiki, Available at: http://ropros.org

[10] Y.H. Choi, J.W. Lee, J.H. Suh, J.D. Lee, “The Integrated Autonomous Underwater Navigation System based on Open Platform for Robotic Services,” Proceedings of Modelling, Simulation and Applied Mathematics, pp. 53-56, 2015.

[11] ROS Official Website, http://ros.org

[12] J.H. Li, H.J. Kang, S.M. Hong, Y.H. Choi, J.H. Suh, “Demonstration of P-SURO II AUV’s Autonomous Navigation in the Water

Tank Environment," Proceedings of Ubiquitous Robots and Ambient Intelligence, 2016.

[13] J.W. Lee, J.E. Hong, "A Technique of ADD-based Architecture Design for Low Power Embedded Software," IEMEK J. Embed. Sys. Appl., Vol. 8, No. 4, pp.195-204, 2013 (in Korean).

Jung-Woo Lee (이 정 우)

He received the M.S. degree in software engineering from Korea Advanced Institute of Science and Technology in 2015. His research interests are software architecture for field robotics in various environments.

Email: ricow@kiro.re.kr

Young-Ho Choi (최 영 호)

He received the Ph.D. degree in Electronic and Electrical Engineering from Pohang University of Science and Technology (POSTECH), Korea in 2008. He is currently a director in Field Robotics R&D Division, Korea Institute of Robot and Convergence (KIRO). His research interests include mobile manipulation and machine intelligence.

Email: rockboy@kiro.re.kr

Jong-Deuk Lee (이 종 득)

He received B.E. degree in electronic engineering from Daegu University in 2009.

Email: artofgene@kiro.re.kr

Sung-Jo Yun (윤 성 조)

He received M.S. degree in electronic engineering from Kyung pook National University in 2008. His research interests are embedded system for robotics.

Email: yunsj@kiro.re.kr

Jin-Ho Suh (서 진 호)

He received Ph.D. degree in control engineering from Tokyo Institute of Technology in 2002. He is currently a director of disaster robotics R&D center and an Adjunct Professor of mechanical engineering at Pohang University of Science and Technology (POSTECH). His research interests are machine intelligence, human-robot interaction, and robot system integration.

Email: suhgang@kiro.re.kr