The Transactions of the Korea Information Processing Society
TKIPS Vol.13, No.3, pp.111~121
eISSN: 3022-7011
https://doi.org/10.3745/TKIPS.2024.13.3.111

Model-Based Intelligent Framework Interface for UAV Autonomous Mission

Son Gun Joon† · Lee Jaeho††

ABSTRACT

Recently, thanks to the development of artificial intelligence technologies such as image recognition, research on unmanned aerial vehicles is being actively conducted. In particular, related research is increasing in the field of military drones, which costs a lot to foster professional pilot personnel, and one of them is the study of an intelligent framework for autonomous mission performance of reconnaissance drones. In this study, we tried to design an intelligent framework for unmanned aerial vehicles using the methodology of designing an intelligent framework for service robots. For the autonomous mission performance of unmanned aerial vehicles, the intelligent framework and unmanned aerial vehicle module must be smoothly linked. However, it was difficult to provide interworking for drones using periodic message protocols with model-based interfaces of intelligent frameworks for existing service robots. First, the message model lacked expressive power for periodic message protocols, followed by the problem that interoperability of asynchronous data exchange methods of periodic message protocols and intelligent frameworks was not provided. To solve this problem, this paper proposes a message model extension method for message periodic description to secure the model's expressive power for the periodic message model, and proposes periodic and asynchronous data exchange methods using the extended model to provide interoperability of different data exchange methods.

Keywords : Intelligent Framework, Drone Autonomy, Interface, Interoperability

무인기 자율임무를 위한 모델 기반 지능형 프레임워크 인터페이스

손 건 준† · 이 재 호††

요 약

최근 영상 인식 등의 인공지능 기술 발전에 힘입어 무인기 자율화에 관한 연구가 활발히 이루어지고 있다. 특히 전문 조종 인력 육성에 큰 비용이 들어가는 군용 무인기 분야에서 관련 연구가 늘어나고 있으며, 그중 하나가 정찰용 무인기의 자율적인 임무 수행을 위한 지능형 프레임워크 연구이다. 해당 연구에선 서비스 로봇을 위한 지능형 프레임워크 설계의 방법론을 활용해 무인기용 지능형 프레임워크를 설계하고자 하였다. 무인기의 자율적인 임무 수행 능력을 위해선 지능형 프레임워크와 무인기 모듈의 연동이 원활하게 이루어져야 한다. 하지만 기존 서비스 로봇을 위한 지능형 프레임워크의 모델 기반 인터페이스로는 주기성 메시지 프로토콜을 사용하는 무인기에 대한 연동 제공이 어려웠다. 먼저 주기성 메시지 프로토콜에 대한 메시지 모델의 표현력이 부족했고, 다음으로 주기성 메시지 프로토콜과 지능형 프레임워크의 비동기적 데이터 교환 방식의 상호운용성이 제공되지 않는다는 문제가 있었다. 본 논문에서는 이러한 문제를 해결하기 위해 메시지 주기성 서술을 위한 메시지 모델 확장 방법을 제안하여 주기성 메시지 모델에 대한 모델의 표현력을 확보하고, 확장된 모델을 이용한 주기적 및 비동기적 데이터 교환 방법을 제안하여 서로 다른 데이터 교환 방식의 상호운용성을 제공하고자 한다.

키워드 : 지능형 프레임워크, 무인기 자율화, 인터페이스, 상호운용성

1. 서 론

최근 영상 인식 등의 인공지능 기술 발전에 힘입어 무인기 자율화에 관한 연구가 활발하게 진행되고 있다. 무인기 자율화 연구에선 무인기를 사람의 조종 없이 자율적으로 운용하여 배송 업무 등에 활용하는 것을 목표로 한다. 특히 군용 무인기 분야에선 조종사의 수 부족 및 높은 훈련 비용 문제로 무인기의 자율적인 임무 수행을 위한 연구가 늘어나고 있다.

이러한 연구로 군사 목적 정찰용 무인기의 자율적인 임무 수행을 위한 지능형 프레임워크 연구[1]가 진행되었다. 무인기가 운용자의 통제 없이 자율적으로 임무를 수행할 수 있으려면 자체적으로 환경을 이해하고 행동을 결정할 수 있어야 한다. 해당 연구에선 이러한 요구사항을 만족시키기 위해 기존

※ 본 연구는 2020년 국방과학연구소 미래도전국방기술 연구개발사업(912904601)의 지원을 받았음.
† 비 회 원 : 서울시립대학교 전자전기컴퓨터공학과 석사
†† 종신회원 : 서울시립대학교 전자전기컴퓨터공학부 교수
Manuscript Received : November 10, 2023
Accepted : January 17, 2024
* Corresponding Author : Lee Jaeho(jaeho@uos.ac.kr)

에 연구된 서비스 로봇을 위한 지능형 프레임워크 설계의 방법론을 적용하여 무인기 자율임무를 위한 지능형 프레임워크를 설계하고자 하였다. 서비스 로봇을 위한 지능형 프레임워크는 로봇이 사용자 요구에 대응하여 서비스를 제공하기 위해 필요한 도메인 서비스 환경에 대한 지식 관리 기능, 센서 정보 등을 활용하여 주변 상황을 인식하는 상황 관리 기능, 현재 상황에 적합한 작업을 선택하고 수행하는 작업 관리 기능 등을 통합한 프레임워크이며, 각 기능을 담당하는 에이전트 및 외부 서비스 활용과 통신을 위한 인터페이스로 구성[2]된다.

자율임무를 수행하는 무인기가 무인기의 상태/센서 정보를 통해 환경 변화를 감지하고 현재 상황에 따른 적절한 행동을 수행할 수 있으려면, 지능형 프레임워크와 무인기 모듈이 원활하게 연동될 필요가 있다. 서비스 로봇을 위한 지능형 프레임워크 연구에서도 서비스 로봇과 지능형 프레임워크의 원활한 연동을 위한 인터페이스 구현 방안에 관한 연구[3]를 진행되었다. 해당 연구에서는 로봇 미들웨어와 지능형 프레임워크의 연동을 위해 로봇 메시지 서술을 위한 메시지 모델 서술 방식과 서술된 메시지 모델 기반의 인터페이스를 통한 연동 방식이 제시되었다. 그러나 무인기 자율임무를 위한 프레임워크에서도 해당 방식을 적용하려 하였으나, 지능형 프레임워크와 무인기 모듈의 원활한 연동이 이루어지지 않았다.

이러한 문제는 Fig. 1과 같이 지능형 프레임워크와 무인기 모듈의 통신 방식의 차이로 인해 발생하였다. 지능형 프레임워크 내 통신은 관련성이 높은 정보끼리 묶은 지식정보를 메시지 단위로 사용하며, 메시지 교환이 비동기적으로 이루어진다. 반면에 무인기 모듈은 통신의 효율을 위해 여러 종류의 정보를 포함하는 명령 혹은 상태 메시지를 단위로 이루어진다. 또한 정찰과 같은 무인기의 임무 특성상 실시간성 정보의 확인이 필요하므로, 무인기 모듈은 요청에 따라 메시지를 주고받는 것이 아니라, 일정한 주기를 가지고 지속해서 통신을 수행한다.

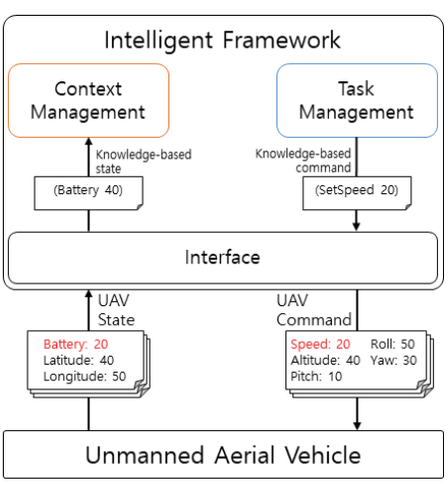

Fig. 1. Message Exchange Process Between Intelligent Framework and UAV Module

이러한 통신 방식의 차이로 인해, 기존 연동 방식엔 고려되지 않았던 지능형 프레임워크와 무인기 모듈의 메시지 교환에 관한 새로운 요구사항이 도출된다. 첫째, 지능형 프레임워크에서 무인기 모듈로 명령을 전달할 때 명령 메시지를 주기적으로 생성하고 송신할 수 있어야 한다. 지식정보 메시지를 단위로 통신하는 지능형 프레임워크에서 무인기가 요구하는 큰 단위의 주기성 메시지를 주기적으로 생성하고 송신할 수 있어야 한다. 둘째로, 무인기 모듈에서 지능형 프레임워크로 무인기 상태를 전달할 때 주기적으로 수신되는 상태 메시지로 인한 데이터 중복 문제에 대처할 수 있어야 한다. 주기적으로 수신되는 무인기 상태 메시지 내의 모든 데이터를 지식정보로 변환하면 중복되는 데이터로 인해 지능형 프레임워크 내 메시지의 시간당 처리량이 불필요하게 높아질 수 있다. 이는 소형화 및 경량화가 중요한 정찰용 무인기에 지능형 프레임워크를 탑재할 때 걸림돌이 될 수 있다. 따라서 메시지 내에서 지능형 에이전트가 필요로 하는 변화된 데이터만을 지식정보로 생성하여 이에 대처할 필요가 있다.

하지만 기존의 모델 기반의 서비스 로봇-지능형 프레임워크 연동 방법은 위의 요구사항에 대해 다음과 같은 한계가 있었다. 먼저, 메시지 모델의 표현력이 부족하다. 기존 연구에서 제시한 메시지 모델 서술 방식은 메시지 주기성에 관한 서술 방법을 제공하고 있지 않아, 주기적인 통신을 요구하는 모듈과의 연동에 적용하기 어렵다. 다음으로, 데이터 교환 방식에 대한 상호운용성을 제공하지 않는다. 기존 연구에서 제시된 메시지 모델은 지능형 프레임워크와 같이 비동기적 통신을 수행하는 모듈과의 연동을 전제로 하고 있어, 주기성 메시지 프로토콜 기반의 통신을 수행하는 무인기 모듈과 같이 다른 데이터 교환 방식을 가지고 있는 모듈과의 연동에 적용하기 어렵다.

본 논문에서는 기존에 연구된 서비스 로봇-지능형 프레임워크 연동 방식으론 무인기와 지능형 프레임워크의 원활한 연동이 어렵다는 문제를 해결하기 위해 다음과 같은 해결 방안을 제안한다.

먼저, 메시지 모델의 확장을 통해 주기성 메시지 프로토콜에 대한 메시지 서술 방법을 제공한다. 기존 메시지 모델에 메시지 주기성 관련 프로퍼티를 추가하여, 확장된 메시지 모델을 통해 주기성 메시지 프로토콜에 대한 메시지 서술 방법을 제공하고자 한다.

다음으로, 메시지 버퍼를 이용해 무인기와 지능형 프레임워크의 상이한 데이터 교환 방식에 대한 상호운용성을 제공한다. 확장된 메시지 모델을 기반으로 하는 메시지 버퍼 관리 기능을 통해 주기성 메시지의 생산 및 송수신을 처리하는 방법으로 주기성 메시지 프로토콜 기반 데이터 교환 방식과 지식정보 기반 비동기 데이터 교환 방식의 상호운용성을 제공하고자 한다.

본 논문에서 제안하는 메시지 모델의 확장을 통한 모델 개선 및 메시지 버퍼 기반의 주기적 및 비동기적 데이터 교환 방법 제공을 통한 인터페이스의 개선으로 무인기와 지능형 프레임워크의 원활한 연동을 제공할 수 있을 것으로 기대된다.

2. 관련 연구

2.1 개요

본 논문의 목표는 기존에 제시되었던 지능형 서비스 로봇의 행위 모듈을 위한 인터페이스 및 모델의 개선을 통해 무인기 자율임무를 위한 무인기 및 지능형 프레임워크 간 원활한 연동을 제공하는 것이다.

이를 위해 본 논문에서는 지능형 프레임워크와 외부 서비스 간 연동을 위한 인터페이스를 관련 연구로 분석하였다.

2.2 지능형 서비스 로봇을 위한 모델 기반 클라우드 서비스 인터페이스

서비스 로봇은 사용자의 요구에 따라 클라우드 서비스와 같은 외부 서비스를 이용할 수 있어야 한다. 하지만 클라우드 서비스는 계속해서 변화하고, 새로운 서비스가 개발되기도 하여 서비스 기능 및 데이터 변경에 적응가능한 일반적이고 유연한 인터페이스가 필요[4]하다. 이러한 요구사항에 따라 모델의 추가 및 교체만으로 클라우드 서비스를 확장할 수 있는 모델 기반의 클라우드 서비스 인터페이스가 연구되었다.

클라우드 서비스 인터페이스는 클라우드 서비스 활용에 필요한 데이터를 서술하기 위해 클라우드 서비스 프로파일을 정의하고 있다. 클라우드 서비스 프로파일은 클라우드 서비스 모델과 인증 모델로 구성된다. 먼저 클라우드 서비스 모델은 Fig. 2와 같이 서비스에 접근하기 위한 주소, HTTP 메소드, 서비스 활용에 필요한 각종 프로퍼티를 포함한다. 인증 모델은 Fig. 3과 같이 클라우드 서비스 활용에 대한 인증 절차를 위한 파라미터 등을 포함한다.

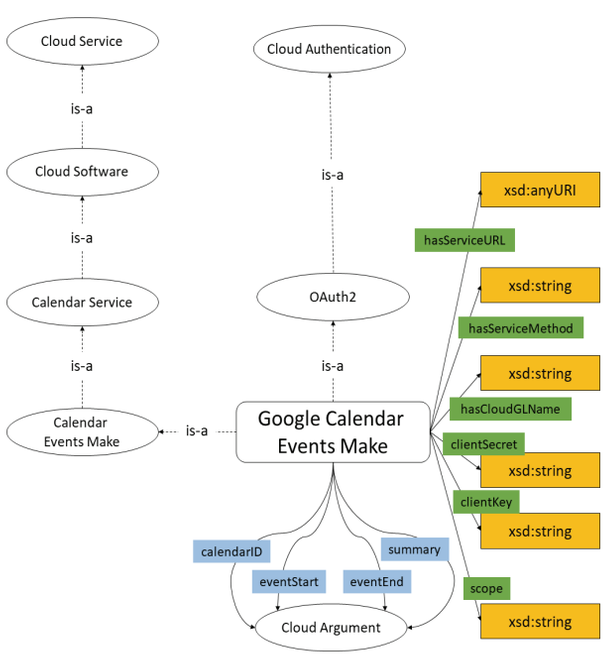

Fig. 2. Cloud Service Profile

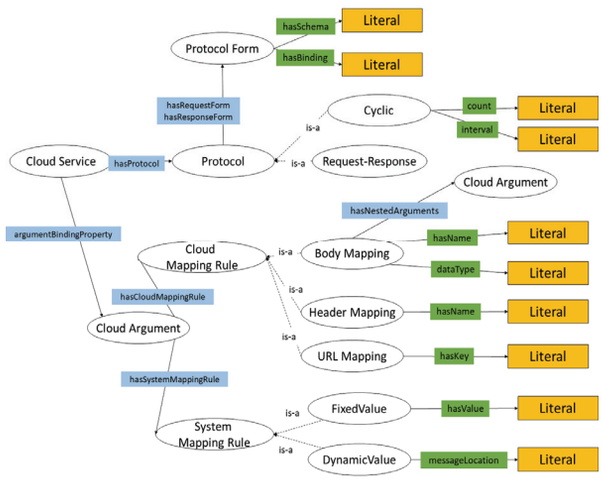

Fig. 3 Cloud Property Model

클라우드 서비스 사용에는 클라우드 프로퍼티도 필요하며, 클라우드 서비스 데이터와 맵핑되기 위한 맵핑 룰, 연계 대상 시스템과 연계하기 위한 맵핑 룰, 프로토콜 형식에 따른 Protocol Type 등으로 구성된다. 클라우드 서비스 인터페이스 연구에선 클라우드 프로퍼티 서술을 위한 모델도 설계하였다.

클라우드 서비스 인터페이스 연구는 이러한 모델들과 이를 기반으로 하는 인터페이스를 통한 클라우드 서비스 연계 방법을 제공한다. 모델 기반의 인터페이스를 통해 외부 서비스의 변화 및 확장에 유연하다는 강점이 있지만, 모델과 모델 기반의 연동 방법에 주기성 메시지 프로토콜에 대한 고려가 되어 있지 않아 무인기와의 연동에는 적합하지 않다.

2.3 CRAM Process Module

CRAM(Cognitive Robot Abstract Machine)은 자율 로봇의 설계 및 구현을 위한 소프트웨어 toolbox[5]이다. 행동 결정을 위한 추론 도구를 포함하는 CRAM은 유연하고 안정적인 자율 로봇을 개발할 수 있게 한다. 다음 Fig. 4는 CRAM의 아키텍처를 나타낸 것이다.

CRAM은 로봇 플랫폼 혹은 시뮬레이션 환경 등의 외부 연동을 위한 Process Module이라는 인터페이스를 정의한다. Process Module은 로봇 기능의 추상화를 통해 로봇이 실제로 사용하는 저수준 모듈과 고수준 부분을 분리하고, 이로써 다양한 로봇들을 같은 고수준 작업 계획으로 제어할 수 있도록 한다.

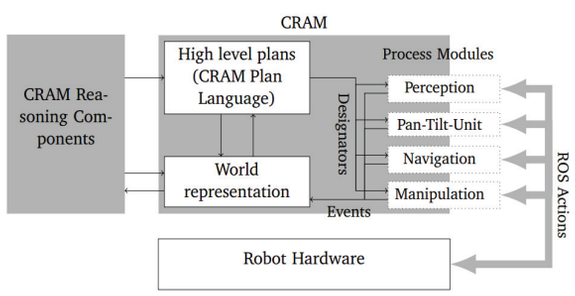

Fig. 4. Architecture of CRAM

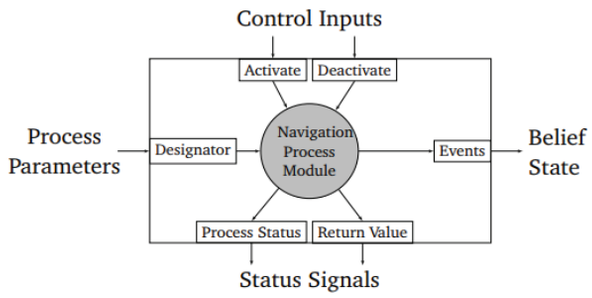

Fig. 5. Example of Process Module for Navigation

CRAM Process Module은 Fig. 5에 나타난 것처럼 저수준 모듈의 연동 방법을 구체적으로 알지 않아도 제어할 수 있도록 해주지만, 지식정보 기반 비동기 데이터 교환 방식과 주기성 메시지 프로토콜의 직접적인 연동 방법에 대한 고려는 없어 무인기와 지능형 프레임워크의 연동에는 적합하지 않다.

3. 배경 연구

3.1 서비스 로봇을 위한 지능형 프레임워크

서비스 로봇을 위한 지능형 프레임워크는 로봇이 사용자 요구에 대응하여 서비스를 제공하는데 필요한 서비스 환경에 대한 정보, 현재 상황에 대한 인지 기능, 작업 계획에 대한 추론 기능 등을 통합한 프레임워크이며, 서비스 수행을 위해 필요한 에이전트 및 외부 서비스 활용과 통신을 담당하는 인터페이스로 구성된다.

위의 서비스 로봇을 위한 지능형 프레임워크의 아키텍처(Fig. 6)[6]를 구성하는 주요 요소는 다음과 같다.

- 작업 관리자: 현재 상황에 따른 의사결정, 작업 추론, 작업의 실행 및 모니터링을 담당하는 에이전트이다.
- 지식 관리자: 서비스 수행에 필요한 지식을 관리하는 에이전트이다.
- 상황 관리자: 추론 규칙을 기반으로 현재 상황을 추론하고 관리하는 에이전트이다.

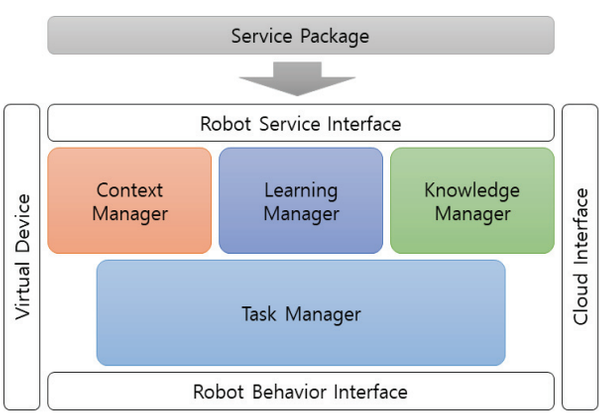

Fig. 6. Architecture of ARBI

- 로봇 행위 인터페이스: 로봇 센서 모듈, 로봇 조작 모듈 등의 프레임워크 외부 요소와의 연계 및 외부 서비스의 활용을 위한 에이전트이다.
- 서비스 패키지: 작업 계획, 상황 정보, 지식 등 서비스 수행을 위해 필요한 정보를 모아 로봇이 활용할 수 있도록 만든 패키지이다.

3.2 무인기 자율임무를 위한 지능형 SW 프레임워크

군용 무인기 전문 인력이 있는 큰 부대급에선, 지상의 운용자가 무인기를 직접 조종하여 무인기를 활용하는 임무를 정상적으로 수행할 수 있다. 그러나, 비교적 작은 부대급에선 운용 인력의 한계로 무인기를 활용하는 임무 수행이 제한적일 수 있다. 따라서, 운용 인력이 제한된 규모의 부대에서 무인기를 활용하는 임무를 온전하게 수행하기 위해선 지상 운용자의 최소한의 제어만으로도 무인기가 자율적으로 임무를 수행할 수 있는 지능형 소프트웨어가 필요하다.

무인기가 운용자의 통제 없이 자율적으로 임무를 수행하기 위해선 무인기가 센서 정보 등을 통해 자체적으로 환경을 이해하고 비행, 센서 제어 등의 행동을 결정할 수 있어야 한다. 무인기 자율임무를 위한 지능형 SW 프레임워크 연구에선 이러한 요구사항을 만족시키기 위해 서비스 로봇을 위한 지능형 프레임워크 설계의 방법론을 적용한 무인기용 지능형 프레임워크를 설계하였다.

무인기용 지능형 프레임워크는 무인기의 자율적인 임무 수행을 위해 실시간 상황 추론을 위한 상황 관리자, 작업의 계획과 실행을 위한 작업 관리자, 지식 관리자, 외부 모듈 활용을 위한 인터페이스 등으로 구성된다. Fig. 7은 무인기용 지능형 프레임워크의 아키텍처를 나타낸 것이다.

3.3 로봇 미들웨어-지능형 프레임워크 연동을 위한 모델 기반 인터페이스

서비스 로봇을 위한 지능형 프레임워크 기반의 로봇 시스템에선 로봇 제어를 위해 로봇 통합 미들웨어를 사용할 수 있

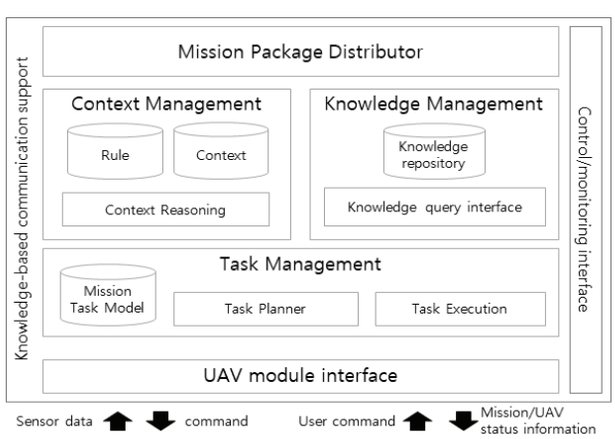

Fig. 7. Intelligent Framework Architecture for Unmanned Aerial Vehicles

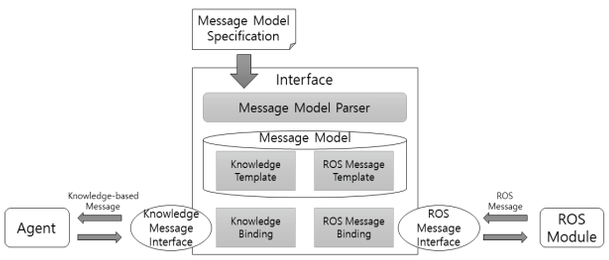

Fig. 8. Robot Middleware-Intelligent Framework Interface

다. 대표적인 로봇 통합 미들웨어로는 ROS(Robot Operation System)[7]이 있다. ROS는 다양한 로봇의 모듈 및 기능을 통합하기 위한 메타-운영체제이다. ROS는 하드웨어 추상화나 시각화 등의 장점이 있지만, 지식정보 기반 커맨드 및 프로토콜에 대한 고려는 되어 있지 않다.

지능형 프레임워크 기반의 로봇 시스템이 ROS와 같은 로봇 미들웨어를 유연하게 사용하기 위해서 로봇 미들웨어-지능형 프레임워크 연동을 위한 모델 기반 인터페이스가 개발되었다. 개발된 인터페이스는 Fig. 8과 같은 구조를 가지며 다음과 같은 주요 기능으로 구성된다.

- 통신 프로토콜 어댑터: ROS 플랫폼에서 제공하는 Topics, Services 등의 통신 프로토콜들을 위한 어댑터로, 지능형 프레임워크의 작업 계획 내에서 호출될 수 있다.
- 메시지 모델 추상화: 다양한 방법으로 정의된 ROS 메시지들을 지능형 에이전트들이 유연하게 사용할 수 있도록 모델 기반의 메시지 서술 방법을 제공한다. 인터페이스는 메시지 모델을 통해 지식 메시지와 ROS 메시지 교환 기능을 제공한다.

로봇 미들웨어-지능형 프레임워크 연동을 위한 모델 기반 인터페이스는 무인기 통신 방식에 대한 연동 제공에는 한계가 있다. 메시지 모델이 메시지 주기성에 대한 서술 방법을 제공하지 않아 메시지 모델의 표현력이 부족하며, 무인기와 같은 주기성 메시지 프로토콜에 대한 상호운용성을 제공하지 않는다. 이에 관해 본 연구는 주기성 관련 정보를 서술할 수 있는 메시지 모델을 개발하고 주기적 및 비동기적 데이터 간의 교환 방법을 제공하기 위해 진행되었다.

4. 연구 내용

본 논문의 목표는 무인기 자율임무를 위해 무인기와 지능형 프레임워크의 원활한 연동을 제공하는 것이다.

이를 위해 먼저 주기성 서술 방법을 제공하기 위한 확장된 메시지 모델을 제시한다. 이는 기존 메시지 모델에 메시지 주기성을 서술하기 위한 프로퍼티를 추가함으로써, 주기성 및 비주기성 메시지를 포괄하는 메시지 서술할 수 있도록 메시지 모델의 표현력을 개선한다.

다음으로, 주기적 및 비동기적 데이터 간의 교환 방법을 제공하기 위해 메시지 버퍼를 이용한 데이터 교환 방법을 제안한다. 이 방법은 확장된 메시지 모델을 기반으로 메시지 버퍼를 관리하고, 메시지 버퍼를 통해 주기성 메시지를 생성하고 송수신하여 상이한 데이터 교환 방식 간의 상호운용성을 제공한다.

4.1 메시지 모델 확장

기존의 서비스 로봇-지능형 프레임워크 연동을 위한 인터페이스 구현 방식에선 로봇 메시지를 모델 기반으로 추상화하여 새롭게 정의되는 메시지들을 지능형 프레임워크에서 유연하게 서술할 수 있도록 하는 메시지 서술 방법을 제공한다. 하지만 무인기는 운용 특성상 연결 상태나 실시간성 정보의 확인이 필요하여 주기성 메시지 프로토콜 기반의 통신 방식을 사용한다. 기존의 메시지 서술 방법은 이러한 프로토콜에 대한 고려가 없어, 무인기 메시지에 대한 온전한 서술이 어렵다.

이러한 문제를 해결하기 위해 기존 메시지 모델을 확장하여 주기성 및 비주기성 메시지를 포괄하는 메시지 서술 방법을 제안한다.

먼저 기존 메시지 모델의 명세의 프로퍼티는 Table 1과 같다.

Name과 ID는 메시지를 구분하기 위한 이름과 식별자이다.

GlMsgStructure는 지식정보 메시지의 구조로, 지식정보의 이름 및 형식의 목록이 포함되며 Table 2는 그 예시를 나타낸 것이다.

Table 1. Existing Message Model Specification

	Description	Type	Arity
Name	Name of message	String	1
ID	Message ID	String	1
GlMsgStructure	Structure of Knowledge message	GlMessage	1
ModuleMsgStructure	Structure of external module's message	ModuleMessage	1
Type	producer/consumer type	String	1
Variable	Variable list in message	Variable[]	1 or more

Table 2. GlMsgStructure Example

```
"glMessageStructure":
[
  {
    "name": "RobotLocation",
    "expression": "(RobotLocation $msgID $timestamp
$x $y)"
  },
]
```

Table 3. Variable Example

```
{
  "name": GroundAltitude
  "type": "short",
}
```

Table 4. Additional Properties for Periodic Message Description

	Description	type	Arity
ProtocolType	Cyclic or Asynchronous	String	1
Frequency	Periodicity Transmission cycle of messages	Integer	0 or 1

ModuleMsgStructure는 외부 모듈(로봇 모듈, 무인기 모듈 등) 메시지의 구조로, 외부 모듈 메시지를 구성하는데 필요한 정보를 포함한다.

Type은 메시지 활용 객체 시점에서의 메시지 producer/consumer 타입이다.

Variable은 메시지 내 포함된 데이터 변수들의 정보 목록이다. 변수명과 타입 등의 정보를 포함되며 Table 3은 그 예시를 나타낸 것이다.

Table 4는 기존 모델에 기존 메시지 모델에 주기성 메시지 서술을 위해 추가하는 프로퍼티인 ProtocolType과 Frequency이다.

ProtocolType은 주기성 메시지와 비주기성 메시지를 구분하기 위한 프로퍼티이다. Cyclic이나 Asynchronous 중 하나의 값이 사용된다.

Frequency는 주기성 메시지의 송신 주기가 서술될 프로퍼티이다. 위의 ProtocolType 프로퍼티의 값이 Cyclic인 주기성 메시지인 경우에만 사용된다.

4.2 메시지 버퍼 기반 주기적 및 비동기적 데이터 교환 방법

기존 서비스 로봇-지능형 프레임워크 연동을 위한 인터페이스 구현 방식은 지능형 프레임워크와 무인기 내 통신 방식의 차이에 대한 고려가 없어 상이한 통신 방식에 대한 상호운용성이 제공되지 않았다. 지능형 프레임워크 내에선 관련성 높은 정보끼리 묶은 지식정보 단위의 메시지를 비동기적으로 교환하는 통신 방식을 사용하고 있으며, Table 5는 지능형 프레임워크 내에서 사용되는 메시지의 예시이다.

무인기 내에선 무선 통신의 효율을 이유로 여러 종류의 정보를 포함하는 비교적 큰 단위의 메시지를 주기적으로 교환하는 통신 방식을 사용하고 있다. Table 6은 무인기 통신에 필요한 데이터 목록을 나타낸 것이다.

Table 5. Examples of Knowledge-based Command Message

Name	Expression
SetMode	(SetMode $mode)
SetRotation	(SetRotation $pitch $roll $yaw)
SetVerticalThrottle	(SetVerticalThrottle $verticalThrottle)

Table 6. Example of Drone Command Message Format

No	Variable Name	Type	Byte Size
1	id	Int	4
2	mode	Int	4
3	pitch	Float	4
4	roll	Float	4
5	yaw	Float	4
6	verticalThrottle	Foat	4

무인기의 모드를 변환하기 위해서는 지능형 프레임워크 내에서는 위 표와 같이 SetMode 메시지를 통해 모드 값 하나만 전달되지만, 실제로 무인기에게 전달되는 메시지에는 모드값 뿐만 아니라 pitch, roll 등과 같이 다른 데이터가 함께 전달되어야 한다.

이러한 지능형 프레임워크 내 프로토콜과 무인기 통신 프로토콜 간의 차이점으로 인해 프레임워크에서 무인기 모듈로 명령 메시지를 전달할 때 큰 단위의 메시지를 주기적으로 생성하고 송신할 방법이 필요하고, 주기적으로 수신되는 메시지로 인해 발생하는 데이터 중복을 대처하는 방법이 필요하다.

본 논문에선 이를 위해 메시지 버퍼를 기반으로 하는 주기적 및 비동기적 데이터 간의 교환 방법을 제안한다.

먼저 무인기가 요구하는 큰 단위의 메시지를 주기적으로 생성하기 위해선 주기성 메시지의 형식과 메시지에 포함될 실제 데이터를 알아야 한다. 따라서 주기성 메시지에 포함될 실제 데이터들이 저장될 공간이 필요하고, 데이터들이 작업 관리자 등의 지능형 에이전트의 지식정보 명령으로 갱신될 수 있어야 한다.

이를 위해 메시지 생성에 필요한 데이터를 저장하는 메시지 버퍼를 활용하는 모듈형 어댑터를 인터페이스에 구성하는 방법을 제안한다. Fig. 9와 같이 모듈형 어댑터는 메시지 모델을 기반으로 메시지 버퍼를 생성하고 관리하여 주기성 메시지 프로토콜과의 연동을 제공한다.

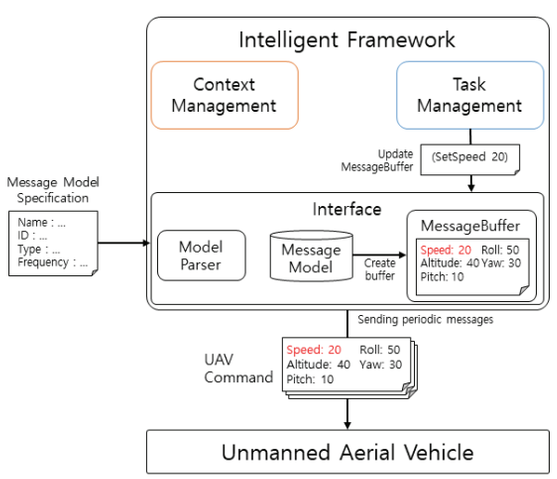

Fig. 9. Message Buffer Operation During Command Transmission

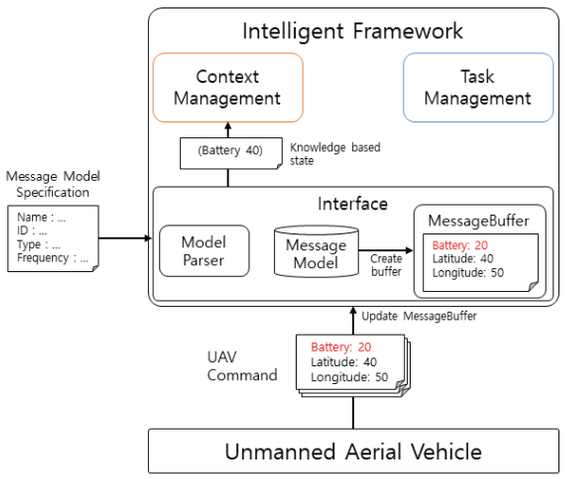

Fig. 10. Message Buffer Operation During UAV State Perception

메시지 버퍼는 메시지 모델 명세로부터 해석된 메시지 모델을 기반으로 생성되며, 메시지가 포함하는 모든 데이터가 메시지별로 저장된다. 주기성 메시지의 생성은 생성하려는 메시지의 메시지 버퍼 내의 데이터를 통해 이루어진다. 메시지 버퍼 데이터의 갱신은 지식정보 명령 메시지를 통해 이루어지며, 지식정보 명령 메시지가 포함하는 데이터 부분만이 갱신된다.

위와 같은 방식으로 관리되는 메시지 버퍼를 통해 완성된 주기성 메시지를 생성할 수 있으므로, 인터페이스에서 주기성 메시지 별로 메시지 모델의 Frequency 프로퍼티를 통해 메시지 별 주기에 맞게 주기성 메시지를 생성하고 송신하는 것이 가능해진다.

다음으로, 주기성 상태 메시지 수신 과정에서 발생하는 데이터 중복에 대처하기 위해선 수신된 메시지 내에서 지능형 에이전트가 필요로 하는 변화된 데이터만을 지식정보로 생산하는 기능이 필요하다. 소형화 및 경량화가 중요한 무인기의 특성상, 데이터 중복을 줄여 지능형 프레임워크의 메시지 처리량을 최적화할 필요가 있다.

이를 위해 위에서 소개한 메시지 버퍼를 기반으로 Fig. 10과 같이 무인기 상태 메시지 내에서 변화된 데이터를 지식정보로 생산하는 모듈형 어댑터의 구성 방안을 제안한다.

5. 실 험

5.1 실험 개요

본 논문에선 무인기-지능형 프레임워크의 연동을 위해 주기성 메시지를 서술하기 위해 확장된 메시지 모델과 이를 기반으로 작동하는 메시지 버퍼 기반 주기적 및 비동기적 데이터 교환 방법을 제안하였다.

이를 검증하기 위해 실제 무인기를 활용한 실험을 설계하였다. 무인기 정찰 임무 시나리오를 설계하고, 해당 임무를 무인기가 자율적으로 수행하는 과정을 분석하였다.

시험용 무인기로는 DJI Air 2S 기종을 사용하였으며, 제조사인 DJI에서 제공하는 무인기용 SDK[8]를 활용해 명령/상태 메시지를 구성하고 무인기와 통신하였다.

먼저 주기성 메시지 프로토콜에 대한 모델의 표현력을 검증하기 위해 정찰 임무 시나리오의 실제 무인기 명령 및 상태 메시지를 본 논문에서 제안하는 모델로 서술하였다.

다음으로 주기성 메시지 프로토콜에 대한 상호운용성을 검증하기 위해 시험용 무인기가 임무를 수행하는 과정에서의 메시지 송수신 상태를 분석하였다. 우선 명령 송신 과정에서 주기성 메시지의 생성 및 송신, 주기성 메시지 데이터의 갱신이 제대로 이루어지는지 확인하였다. 그리고 상태 수신 과정에서 메시지 버퍼를 이용해 변화된 데이터만을 지식정보로 생산하는 경우와 모든 데이터를 지식정보로 생산하는 경우의 지식정보 메시지 처리량을 비교하여 데이터 중복에 대한 대처 성능을 검증하였다.

5.2 실험 환경

지능형 프레임워크와 무인기의 연동을 검증하기 위해 실제 무인기를 활용한 실험을 구성하였다. 실험을 위해 선정한 시험용 무인기는 DJI에서 제조한 DJI Air 2S 기종이다. 제조사 DJI에선 무인기와의 통신을 위한 인터페이스를 스마트기기용 SDK를 통해 제공한다. 지능형 프레임워크에서 무인기에 제어 명령을 보내고 상태정보를 받아 보기 위해 스마트기기용 SDK를 기반으로 통신 중계용 Mobile Application을 개발하여 다음 그림과 같은 방식으로 무인기와의 통신을 구현하였으며, 그 통신 과정은 Fig. 11과 같다.

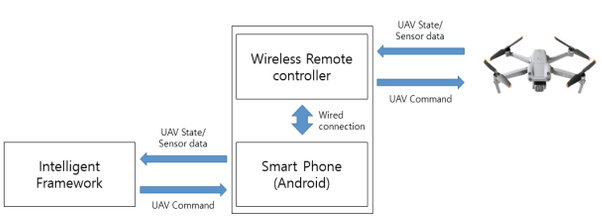

Fig. 11. Configuration of Communication Relay Between Intelligent Framework and Test UAV

Table 7. Example of Drone Status Message Format

No	Variable Name	Type	Byte Size
1	id	Int	4
2	latitude	String	4
3	longitude	String	4
4	altitude	Float	4
5	pitch	Double	4
6	roll	Double	4
7	yaw	Double	4
8	takeoffed	Int	4
9	gpsSignalLevel	String	4
10	flightTimeInSeconds	Int	4

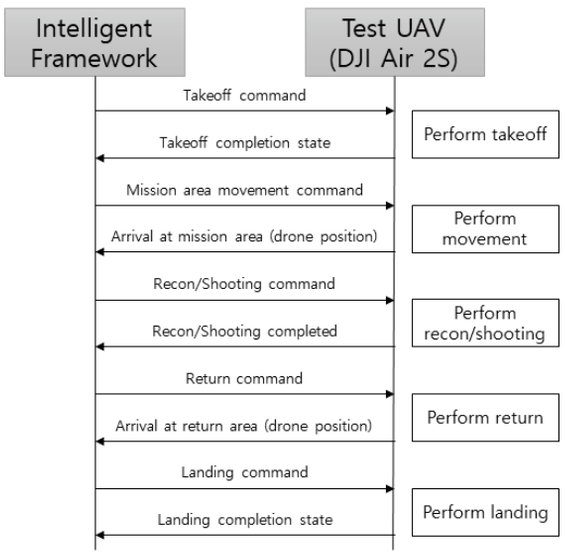

Fig. 12. Sequence Diagram of Reconnaissance Mission Scenario

무인기 명령 및 상태 메시지는 스마트기기용 SDK의 인터페이스를 기반으로 Table 7과 같이 설계하고, 이를 본문에서 제안한 확장된 메시지 모델로 서술하였다.

무인기가 수행할 정찰 임무 시나리오는 Fig. 12와 같이 설계하였다.

정찰 임무 시나리오에서 무인기는 이륙 후 임무 지역으로 이동하여 임무 지역을 정찰한다. 마주치는 대상에 따라 회피 기동 등의 적절한 행동을 자율적으로 수행하고, 기지에 도착했다고 판단하면 착륙한다.

5.3 실험 결과

먼저 본 논문에서 제시된 확장된 메시지 모델 명세를 통해 서술된 메시지 모델을 확인한다. Table 8은 DJI 스마트기기용 SDK에서 제시된 드론 메시지 프로토콜을 기반으로 작성된 메시지 모델이다.

주기성 메시지의 지식정보 메시지 구조와 무인기 메시지 구조에 더해 Frequency 및 주기/비동기 프로퍼티를 서술하여 메시지 활용에 필요한 정보를 온전하게 서술하였다. 이를 통해 주기성 메시지 프로토콜에 대한 메시지 모델의 표현력을 검증하였다.

다음으로 정찰 임무 수행 과정에서의 지능형 프레임워크와 무인기 간의 명령 및 상태 메시지 교환 결과를 확인한다. 이를 통해 지능형 프레임워크 인터페이스의 주기성 메시지 프로토콜과 지식정보 기반 데이터 교환 방식의 상호운용성 제공 여부를 검증한다.

먼저 지능형 프레임워크 인터페이스를 통한 무인기 명령 메시지 송신 결과는 다음 Table 9와 같다.

주기적으로 송신되는 하나의 무인기 상태 메시지를 통해 세 가지의 지식정보(DroneStatus, DroneLocation, DroneAttitude)를 생성한 것을 확인할 수 있다. 또한, 메시지 버퍼를 활용해 주기적으로 수신되는 메시지 내에서 변화된 데이터만을 지식정보로 생산하였다. 메시지 처리량 비교를 위해 데이터 중복성 대처 방법을 적용한 것과 하지 않은 구현체를 활용하여 같은 임무를 같은 시간 동안 수행하여 발생한 지능형 프레임워크 내 메시지 양을 비교하였다.

Table 10은 데이터 중복성 대처 방법을 적용한 경우의 로그이고, Table 11은 적용하지 않은 경우의 로그이다. 표 10에서 나타난 로그를 보면 ASSERT된 메시지의 값이 계속해서 변화하는 것을 확인할 수 있다. 하지만 Table 11에서 나타난 로그를 보면 두 번째와 세 번째로 ASSERT 된 DroneAttitude 메시지의 데이터가 동일한 것을 확인할 수 있다.

다음의 Table 12는 데이터 중복성 대처 방법이 적용된 경우와 적용되지 않은 경우의 초당 메시지 처리량을 비교한 것이다.

정찰 임무 수행 과정에서 메시지 버퍼 기반의 데이터 교환 방식 적용을 통해 지식정보 메시지 처리량이 약 39% 감소한 것을 확인할 수 있다.

위의 지능형 프레임워크와 무인기의 메시지 교환 결과를 통해 메시지 버퍼 기반 주기적 및 비동기적 데이터 교환 방법의 주기성 메시지 프로토콜과 지식정보 기반 데이터 교환 방식의 상호운용성 제공 여부를 검증할 수 있었다.

6. 결 론

기존에 개발되었던 로봇 미들웨어-지능형 프레임워크의 연동을 위한 모델 기반 인터페이스는 로봇과의 통신을 목적으로 하고 있다. 이 때문에 무인기와 같이 주기성 메시지를 요구하는 대상과의 통신에 적용하기에 메시지 모델의 표현력 부족, 서로 다른 데이터 교환 방식에 대한 상호운용성을 제공하지 못한다는 한계를 가지고 있었다.

본 논문에서는 이러한 한계를 극복하기 위하여 지능형 프레임워크와 무인기 간의 통신 지원을 위한 주기성 및 비동기적 메시지를 서술할 수 있는 메시지 모델과 이를 활용한 메시지 버퍼 기반의 주기적 및 비동기적 데이터 교환 방법을 제안하였다.

확장된 메시지 모델은 추가된 프로퍼티를 통해 메시지의 주기성 여부 및 메시지 주기 서술이 가능하게 되었으며, 메시지 버퍼 기반의 데이터 교환 방법을 통해 비동기적으로 통신하는 지능형 프레임워크와 주기적으로 통신하는 무인기 간의 메시지 교환을 지원할 수 있었다.

또한 실제 무인기를 활용한 실험을 통해 지식 기반의 메시지를 활용한 비동기적 통신을 수행하는 지능형 프레임워크와 주기성 메시지 프로토콜을 사용하는 무인기 간의 상호운용성을 확인할 수 있었다.

본 논문에서는 무인기와 같은 주기성 메시지 프로토콜을 활용하는 대상에 대한 연동을 제공하는 방법을 중점으로 연구하였다. 하지만, 주기성 메시지의 송수신이 갑작스럽게 단절되는 예외 상황 등에 대해서 인터페이스가 대처할 수 있어야 한다. 이를 위해 주기성 메시지 프로토콜을 활용하는 외부 모듈과의 연동 상황에서 발생하는 예외 상황의 처리에 관한 추가적인 연구가 필요하다.

Table 8. Example of Drone Status Message Model

```
{
  "Name": "DJI_Demo_DroneStatus",
  "ID": "3",
  "Type": "Consumer",
  "ProtocolType": "Cyclic",
  "Frequency": 4,
  "ModuleMessageStructure":
  [
    { "name": "id", "size": 4, "type": "string",
"classification": "Const", "value": "3" },
    { "name": "latitude", "size": 8, "type": "string", "classification": "Val" },
    { "name": "longitude", "size": 8, "type": "string", "classification": "Val" },
    { "name": "altitude", "size": 4, "type": "float", "classification": "Val" },
    { "name": "pitch", "size": 4, "type": "double", "classification": "Val" },
    { "name": "roll", "size": 4, "type": "double", "classification": "Val" },
    { "name": "yaw", "size": 4, "type": "double", "classification": "Val" },
    { "name": "takeoffed", "size": 4, "type": "int", "classification": "Val" },
    { "name": "gpsSignalLevel", "size": 4, "type": "string", "classification": "Val" },
    { "name": "flightTimeInSeconds", "size": 4, "type": "int", "classification": "Val" }
  ],
  "Variable":
  [
    { "name": "latitude", "type": "string", "unit": "None", "resolution": "None" },
    { "name": "longitude", "type": "string", "unit": "None", "resolution": "None" },
    { "name": "altitude", "type": "float", "unit": "None", "resolution": "None" },
    { "name": "pitch", "type": "double", "unit": "None", "resolution": "None" },
    { "name": "roll", "type": "double", "unit": "None", "resolution": "None" },
    { "name": "yaw", "type": "double", "unit": "None", "resolution": "None" },
    { "name": "takeoffed", "type": "int", "unit": "None", "resolution": "None" },
    { "name": "gpsSignalLevel", "type": "string", "unit": "None", "resolution": "None" },
    { "name": "flightTimeInSeconds", "type": "int", "unit": "None", "resolution": "None" }
  ],
  "GlMessageStructure":
  [
    { "name": "DroneLocation",
      "expression": "(DroneLocation $msgID $timestamp $latitude $longitude $altitude)" },
    { "name": "DroneStatus",
      "expression": "(DroneStatus $msgID $timestamp $takeoffed $gpsSignalLevel $flightTimeInSeconds)" },
    { "name": "DroneAttitude",
      "expression": "(DroneAttitude $msgID $timestamp $pitch $roll $yaw)" }
  ],
  "MessageInfo":
  { "sender": "agent://www.ai.com/demo/agent/drone/TestDrone",
    "receiver": "agent://www.ai.com/demo/KnowledgeManager" }
}
```

Table 9. Result of Receiving Drone Command Message During Reconnaissance Mission

```
write result: 32, content: 1 1 0 0 0 0
write result: 32, content: 1 1 0 0 0 0
write result: 32, content: 1 1 0 0 0 0
on request to agent://www.ai.com/demo/BehaviorInterface
            sender                  : agent://www.ai.com/demo/TaskManager
            request                 : (ControlDrone "1.000000" "789" "4" "0" "0" "0" "0.250000")
write result: 32, content: 1 1 0 0 0 0
[TCP] message from client: 3 0.0 0.0 0.5 0.0 -0.7 -25.1 1 4 3
[InternalCommunicator] sending (assert (DroneLocation 26 0 "0.0" "0.0" 0.500000))
[InternalCommunicator] sending (assert (DroneStatus 27 0 1 "4" 3))
[InternalCommunicator] sending (assert (DroneAttitude 28 0 0.000000 -0.700000 -25.100000))
write result: 32, content: 1 1 0 0 0 0
write result: 32, content: 1 1 0 0 0 0
write result: 32, content: 1 1 0 0 0 0
write result: 32, content: 1 1 0 0 0 0
write result: 32, content: 1 1 0 0 0 0
write result: 32, content: 1 4 0 0 0 0.250000
write result: 32, content: 1 4 0 0 0 0.250000
write result: 32, content: 1 4 0 0 0 0.250000
```

Table 10. Case 1 Results of Receiving Drone Status Messages During a Reconnaissance Mission

[ASSERT] (DroneStatus 126.000000 0.000000 1.000000 "4" 28.000000)
[NOTIFY] (DroneStatus 126.000000 0.000000 1.000000 "4" 28.000000)
[ASSERT] (DroneAttitude 127.000000 0.000000 -0.300000 -0.400000 -25.200000)
[NOTIFY] (DroneAttitude 127.000000 0.000000 -0.300000 -0.400000 -25.200000)
[ASSERT] (DroneLocation 128.000000 0.000000 "0.0" "0.0" 1.700000)
[NOTIFY] (DroneLocation 128.000000 0.000000 "0.0" "0.0" 1.700000)
[ASSERT] (DroneStatus 129.000000 0.000000 1.000000 "4" 29.000000)
[NOTIFY] (DroneStatus 129.000000 0.000000 1.000000 "4" 29.000000)
[ASSERT] (DroneAttitude 130.000000 0.000000 0.200000 -0.500000 -25.300000)
[NOTIFY] (DroneAttitude 130.000000 0.000000 0.200000 -0.500000 -25.300000)
[ASSERT] (DroneLocation 131.000000 0.000000 "0.0" "0.0" 1.700000)
[NOTIFY] (DroneLocation 131.000000 0.000000 "0.0" "0.0" 1.700000)
[ASSERT] (DroneStatus 132.000000 0.000000 1.000000 "4" 30.000000)
[NOTIFY] (DroneStatus 132.000000 0.000000 1.000000 "4" 30.000000)

Table 11. Case 2 Results of Receiving Drone Status Messages During a Reconnaissance Mission (Data Redundancy Response Method Not Applied)

[ASSERT] (DroneStatus 210.000000 0.000000 1.000000 "4" 29.000000)
[NOTIFY] (DroneStatus 210.000000 0.000000 1.000000 "4" 29.000000)
[ASSERT] (DroneAttitude 211.000000 0.000000 0.000000 0.700000 -16.300000)
[NOTIFY] (DroneAttitude 211.000000 0.000000 0.000000 0.700000 -16.300000)
[ASSERT] (DroneLocation 212.000000 0.000000 "0.0" "0.0" 1.800000)
[NOTIFY] (DroneLocation 212.000000 0.000000 "0.0" "0.0" 1.800000)
[ASSERT] (DroneStatus 213.000000 0.000000 1.000000 "4" 30.000000)
[NOTIFY] (DroneStatus 213.000000 0.000000 1.000000 "4" 30.000000)
[ASSERT] (DroneAttitude 214.000000 0.000000 0.100000 0.900000 -16.500000)
[NOTIFY] (DroneAttitude 214.000000 0.000000 0.100000 0.900000 -16.500000)
[ASSERT] (DroneLocation 215.000000 0.000000 "0.0" "0.0" 1.800000)
[NOTIFY] (DroneLocation 215.000000 0.000000 "0.0" "0.0" 1.800000)
[ASSERT] (DroneStatus 216.000000 0.000000 1.000000 "2" 30.000000)
[NOTIFY] (DroneStatus 216.000000 0.000000 1.000000 "2" 30.000000)
[ASSERT] (DroneAttitude 214.000000 0.000000 0.100000 0.900000 -16.500000)
[NOTIFY] (DroneAttitude 214.000000 0.000000 0.100000 0.900000 -16.500000)

Table 12. Message Throughput Comparison

	message / sec
Case 1	4.4
Case 2	7.2
Case 1 / Case 2	61.1%

References

[1] J. C. Shin, S. W. Kim, G. H. Baek, and M. G. Seo, "A proposal for software framework of intelligent drones performing autonomous missions," *Journal of Advanced Navigation Technology*, Vol.26, No.4, pp.205-210, 2022.

[2] B. G. Choi, "A specification-based service development for intelligent service robots," Ph.D. Dissertation, University of Seoul, Seoul, Korea, 2020.

[3] C. S. Park, B. G. Choi, and J. H. Lee, "Behavior interface for robot intelligence integration framework," in *Korea Computer Congress 2018*, pp.935-937, 2018.

[4] B. G. Choi, J. U. Lee, S. K. Park, and J. H. Lee, "A model-based interface to cloud services for intelligent service robots," in *KIPS Transactions on Software and Data Engineering*, Vol.9, No.1, pp.1-10. 2020.

[5] M. Beetz, L. Mösenlechner, and M, Tenorth, "CRAM - A cognitive robot abstract machine for everyday manipulation in human environments," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, pp.1012-1017, 2010.

[6] Intelligence Architecture [Internet], http://www.robot-intelligence.kr/index.php/Intelligence_Architecture

[7] Robot Operating System [Internet]. http://www.ros.org.

[8] DJI Developer Tehnologies [Internet]. https://developer.dji.com/

손 건 준

https://orcid.org/0009-0008-0323-3332
e-mail : gunjoons@naver.com
2021년 서울시립대학교
전자전기컴퓨터공학부(학사)
2023년 서울시립대학교
전자전기컴퓨터공학과(석사)
관심분야 : 인공지능, 인터페이스

이 재 호

https://orcid.org/0000-0002-3332-3207
e-mail : jaeho@uos.ac.kr
1985년 서울대학교 계산통계학과(학사)
1987년 서울대학교 계산통계학과(석사)
1997년 University of Michigan(박사)
1998년 ~ 현 재 서울시립대학교
전자전기컴퓨터공학부 교수
관심분야 : 인공지능, 지능 로봇