실시간 객체 모델을 이용한 내장형 무인항공기 제어시스템의 설계 및 구현

박한솔°, 신찬희, 김문회
건국대학교 컴퓨터정보통신공학과
{parkhs°, newchany, mhkim}@konkuk.ac.kr

Design and Implementation of an Embedded Real-Time UAV Control System based on the TMO Model

Hansol Park°, Chanhee Shin, Moon Hae Kim
Dept. of Computer Science and Engineering, Konkuk University

요 약

최근 무인항공기는 정찰임무를 포함하여, 기상관측, 해상관측 등의 여러 임무로 사용되고 있으며, 점점 활용도가 높아지고 있다. 이러한 무인항공기 제어시스템은 과거에 비해 복잡도가 증가했으며, 보다 정밀한 실시간 제어가 요구되고 있다. 이러한 제어시스템을 위한 객체지향 설계방법론은 복잡한 모델에 대한 분석을 보다 빠르게 해주고, 재사용이 가능한 시스템 구축을 가능하게 해 줄 것이다. 이러한 목적을 위해 본 논문에서는 실시간 객체 모델인 TMO모델을 사용하여 시스템을 설계하고, TMO모델을 지원해 주는 리눅스 기반의 미들웨어를 사용하여 무인항공기 제어 시스템을 구현함으로써, 정밀한 실시간 제어시스템에 대한 TMO모델의 적합성을 판단하고, TMO모델 지원 미들웨어의 성능을 평가해 본다.

1. 서 론

오늘날 무인항공기는 군사적 목적의 정찰을 비롯해 기상, 해양 등의 탐사, 위험한 실험의 관측 등 다양한 목적으로 활용되고 있다. 이러한 다양한 활용은 무인항공기가 수행해야 하는 임무를 보다 복잡하게 하고 있으며, 이로 인해 무인항공기 제어를 위한 소프트웨어 역시 기존의 소프트웨어에 비해 복잡해지고 있다.

이상적인 무인항공기의 제어 소프트웨어는 안전한 비행을 위해서 정밀한 실시간 제어를 효과적으로 수행하고, 유도/항법 알고리즘을 제한된 시간 내에 계산해야 하며, 추후에 다른 무인항공기의 제어 소프트웨어로 재사용 가능하도록 작성되어야 한다. 하지만 현재 무인항공기에 탑재되는 제어컴퓨터에는 일반적으로 저 사양의 컨트롤러가 장착되고 있으며, 단순한 임무를 수행하는데 필요한 정보를 처리하기에는 충분하지만 다양한 임무를 수행하는데 필요한 정보를 처리하기에는 부족한 메모리와 컨트롤러 성능을 가지고 있어 복잡해지는 소프트웨어를 동작시키기에는 제한적인 환경을 제공하였다. 하지만 최근 집적 회로 기술의 발달에 힘입어 무인항공기에 탑재 가능한 소형의 고성능 컨트롤러가 등장함에 따라 복잡도가 높은 소프트웨어의 동작을 위한 충분한 환경을 갖게 되었다.

더욱더 복잡해지는 무인항공기 제어소프트웨어의 효율적인 개발과 다양한 분야에서의 재사용성을 위해 좀 더 체계적인 개발 방법론이 필요하며, 이를 위해 본 논문에서는 실시간 성능 보장이 가능한 TMO모델을 이용하여 무인항공기 제어시스템을 설계하고, TMO모델 지원 미들웨어인 TMOSM을 사용하여 공개 운영체제인 리눅스 기반환경에서 설계된 제어시스템을 구현할 것이다. 또한 성능 평가를 위해 가상의 시뮬레이션 환경에서 모의테스트를 통해 TMO모델이 무인항공기의 제어와 같은 정밀한 실시간 제어에 적합한지 판단해 보고, 지원 미들웨어의 성능을 평가해 보겠다.

2. 배경연구

2.1 TMO 모델의 소개

TMO는 Time-triggered Message-triggered Object의 약자로써, Kane Kim 등에 의해서 개발된 Object Structuring Scheme이다. TMO는 기존의 객체 모델을 경성 실시간 시스템에서 높은 효율성을 보일 수 있는 객체 모델로 확장하기 위한 연구에서 나온 결과이다. 따라서 TMO는 실시간 시스템이 가지는 시간적인 특성과 행동을 쉽게 추상화 할 수 있는 구조를 가지고 있을 뿐 아니라, 적시 서비스 능력 (timely service capability)을 시스템 설계 단계에서부터 보장할 수 있다[1].

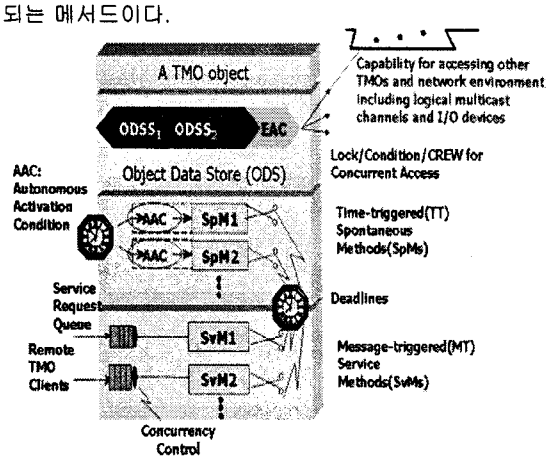
TMO 객체모델의 구조는 다음 4개의 부분으로 구성된다.

- Object Data Store (ODS) : 실시간 데이터를 저장하기 위한 부분으로 Object Data Store Segment (ODSS) 단위로 관리된다. ODSS는 TMO 메서드인 SpM과 SvM에 의해서 상호배타적으로 접근 가능하다.
- Environment Access Capability (EAC) : 외부와의 논리적인 통신 채널, 그리고 I/O 디바이스 인터페이스 등에 대한 연결 창구다.
- Spontaneous method (SpM) : 주기성을 띠거나 시간

본 논문은 정보통신부 대학 IT연구센터 육성·지원사업의 연구결과임.

성을 갖는 메서드이다.

• Service method (SvM) : 기존의 객체 모델이 가지는 메서드 그룹과 같은 형태로써 클라이언트로부터 온 메시지에 의해 분산 TMO 객체의 서비스 호출을 위해서 제공되는 메서드이다.



< 그림 1 > TMO 모델의 기본구조

<그림 1>은 TMO 객체 모델의 기본적인 구조를 보여주고 있으며, 기존의 객체 모형과는 다른 다음과 같은 특징들을 가지고 있다.

(1) 분산 컴퓨팅 컴포넌트

TMO들은 서버에 있는 서비스 메서드에 대한 클라이언트 호출을 통해서 서로 상호작용을 한다. 이때, 멀티노드의 TMO 객체들은 non-blocking 형태의 RMI (Remote Method Invocation)을 통하여 분산 처리를 수행한다.

(2) Spontaneous method (SpM)

Time-triggered method인 SpM은 클라이언트의 서비스 요청에 의해서 실행되는 SvM과는 달리 TMO 설계 시에 명세한 시간이나 주기가 되면 실시간 클럭(clock)에 의해 자동으로 실행되는 메서드이다. SpM의 시간 조건은 디자인 시에 Autonomous Activation Condition (AAC)에 상수로 명세 된다.

아래는 AAC 명세 방법에 대한 예시를 보여주고 있다.

```
"for t =    from 9:00am to 10:30am
every 20min
start-during (t, t+1min)
finish-by t+5min"
```

위 AAC 명세의 의미는, " 9:00am에서 10:30am 까지 20분을 주기로 활성화되어야 하고, 매 주기의 시작은 t+1분 사이에 활성화되며 t+5분 안에 끝마쳐야 한다." 로 해석할 수 있다.

(3) Basic concurrency constraint (BCC)

TMO들의 시간적인 서비스 능력을 보장하기 위한 제약조건으로써, SpM과 SvM이 공유데이터 ODS를 동시에 접근하려고 할 때 발생할 수 있는 충돌을 방지하기 위한 수행 규칙이다. 정확히 말하자면 SpM과 SvM 사이에는 데이터를 공유하기 위한 ODSS (Object Data Store Segment)가 존재하는데, 이를 동시에 액세스 하려는 경우에 SpM이 SvM 보다 더 높은 우선순위를 가지는 것이다. 그러므로 객체의 수행되는 시간을 디자인 단계에서 고정시킬 수 있고, SpM의 수행은 SvM에 의해서 방해 받지 않으며, SpM의 수행 시 그 시간을 보장할 수 있는 것이다.

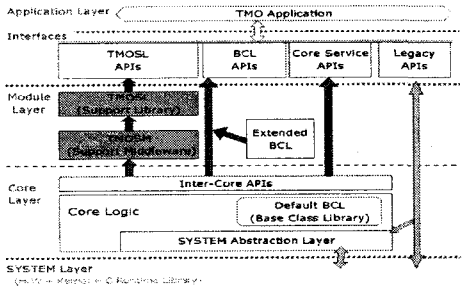
(4) 종료시간과 데드라인 보장

디자이너가 메서드의 시작시간, 종료시간 그리고 데드라인을 명세함으로써 시스템의 적시 서비스 능력을 디자인 단계에서 보장할 수 있도록 지원 한다.

현재 TMO 객체모델은 실시간 처리를 필요로 하는 군사, 공장제어, 교통, 그리고 멀티미디어 등의 응용과 실시간 시뮬레이션 분야에 적용되어 활발한 연구가 진행 중이다.

2.2 TMO 지원 미들웨어

TMOSM/Linux 는 건국대학교 소프트웨어 연구센터에서 실시간 객체 모델인 TMO의 주요 기능을 C++ 객체로 구현한 처리엔진으로써 리눅스 및 임베디드 리눅스 플랫폼에서 TMO 모델을 지원할 수 있는 미들웨어 및 최적화 된 실시간 지원 런타임 라이브러리를 총칭한다. TMOSM/Linux는 TMO 모델을 지원하는데 발생하는 여러 가지 문제점들을 고려해서 실시간 지원, 분산 처리 부분에 내부적으로 개선 가능성을 내포한 구조를 하고 있다. < 그림 2 >는 TMOSM/Linux의 구조를 나타낸다[7].



< 그림 2 > TMOSM/Linux 구조

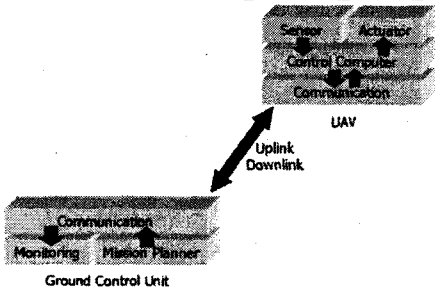
2.3 무인항공기 운용시스템의 소개

무인항공기의 운용을 위해서는 항공기의 제어시스템 외에도 별도의 지상스테이션이 필요하다. 지상 통제 시스템에서는 무인항공기가 수행해야 하는 임무를 계획하기 위한 계획 및 전략부가 필요하며, 만들어진 임무 및 계획 데이터를 지상 송수신 장비를 통해 무인항공기로 전달해 주어야 한다. 무인항공기는 이 정보를 이용하여 임무를 수행하게 되고 임무를 수행하는 동안에 계속해서 상태 정보를 지상 송수신 장비를 통해 지상 통제 시스템으로 보낸다. 지상 통제 시스템에서는 이 데이터를 이용하여 비행

정보 및 임무에 대한 분석이 이루어지게 된다. 이와 같은 일련의 작업들은 매우 정교하게 이루어 져야 하며, 군사용 목적으로 사용되는 경우 통신상의 보안도 고려하여야 한다.

지상 통제 시스템의 경우 두 가지 부분으로 나눌 수 있다. 임무를 계획하고 무인항공기에 전달해 주기 위한 부분이 첫 번째 이고, 무인항공기의 상황을 모니터링 하기 위한 부분이 두 번째 이다[5,6].

무인항공기 제어 컴포넌트 또한 마찬가지로 두 가지 부분으로 나눌 수 있다. 항공기의 자세를 안정적으로 유지하기 위한 부분과, 항공기의 경로를 제어하는 부분이 바로 그것이다. 항공기의 자세를 제어하기 위해서는 Gyro-Scope 센서, 관성 센서 등이 필요하나 본 논문에서는 Gyro-Scope만을 사용한다. 그리고 경로 처리를 위해서는 GPS 센서가 사용된다. 이와 같은 센서 부분과 이 센서 데이터를 처리하기 위한 제어 컴퓨터가 필요하며, 무인항공기의 통신 컴포넌트를 통해 관련 정보를 지상으로 보내주게 된다. < 그림 3 >은 이와 같은 구조를 나타낸다.



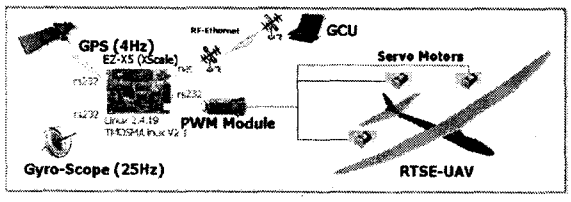
< 그림 3 > 무인항공기 운용시스템의 구조

3. 무인항공기 제어를 위한 시스템의 설계

3.1 운용시스템의 구조 및 설계

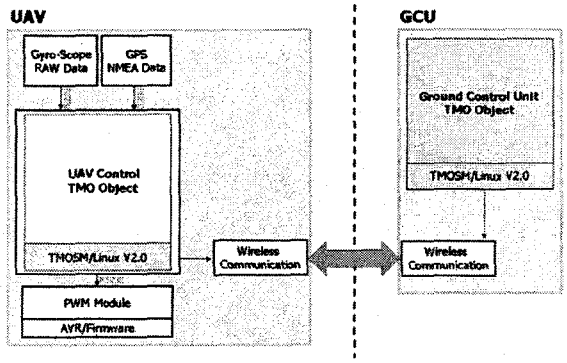
다음 < 그림 4 >는 무인항공기 운용시스템의 하드웨어 구성도 이다. 무인항공기가 유도/항법 제어를 수행하기 위해서는 다음과 같은 하드웨어가 요구된다.

- 제어 보드(EZ-X5) : 처리를 위한 메인 컴퓨터
- GPS : 위치 정보를 얻기 위한 Global Position Sensor
- Gyro-Scope : 자세 정보를 얻기 위한 3축 자이로 센서
- PWM Module : 디지털 신호를 서보 컨트롤을 위한 아날로그 신호로 바꾸는 컨버터
- GCU (Ground Control Unit) : 지상에서 임무를 작성하여 무인항공기로 전송해 주는 스테이션



< 그림 4 > 무인항공기 제어시스템 하드웨어 구성도

< 그림 4 >에 주어진 하드웨어를 이용하여 설계된 운용시스템의 전반적인 구조는 다음 < 그림 5 >과 같다.



< 그림 5 > 무인항공기 제어시스템의 설계

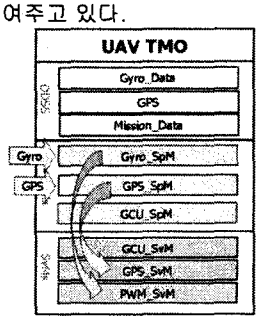
운용시스템은 무인항공기 제어시스템과 지상스테이션으로 구분되며, 각각 UAV Control Object와, Ground Control Unit TMO Object로 구분되며, 해당 객체가 동작하기 위한 TMO 미들웨어인 TMOSM V2.0을 사용한다.

3.2 무인항공기 제어시스템의 설계

무인항공기 제어시스템은 다음과 같은 요구사항을 갖는다[2,3,4].

- Gyro-Scope로부터 25Hz로 데이터를 받아서 40ms 이내에 처리해야 한다.
- GPS로부터 4Hz로 데이터를 받아서 250ms 이내에 처리해야 한다.
- Gyro-Scope와 같은 동작속도인 25Hz로 PWM Module을 컨트롤해서 자세를 제어해야 한다.
- 지상에서 임무를 받을 수 있음
- 현 비행 상황을 지상으로 보낼 수 있음

Gyro_SpM과 GPS_SpM에서는 빠른 처리를 위해 각각의 센서에서 데이터를 읽어 와서 바로 해당 처리를 하는 SvM으로 처리를 위임하게 되며, 데이터는 공유데이터 영역인 ODSS에 저장하게 된다. SpM의 특성상 우리는 SpM에 시간 조건을 정해 줄 수 있으며, 각각 데드라인을 40ms와 250ms로 둔다. 다음 < 그림 6 >은 설계된 UAV TMO 객체를 보여주고 있다.

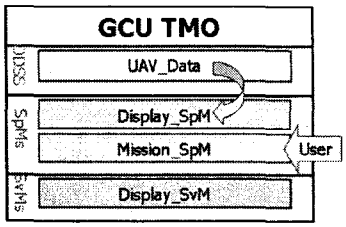


< 그림 6 > 무인항공기 제어시스템 TMO객체의 설계

3.3 지상 통제 시스템의 설계
지상 스테이션의 경우, 요구 조건은 다음의 3가지이다.

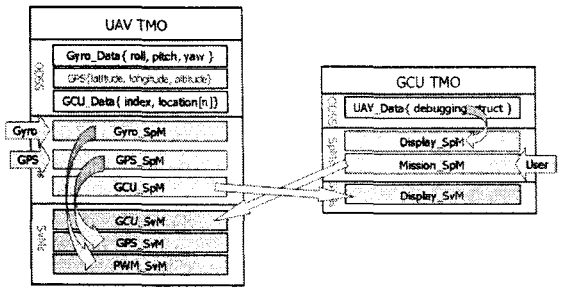
- 임무의 수정 및 작성 가능
- 무인항공기로 임무 전송 가능
- 무인항공기의 현 비행 상황 모니터링

임무의 수정 및 작성을 위해 별도의 GUI 파트가 추가되며, 데이터 전송을 위해서 TMO모델의 Gate가 사용된다. Gate를 이용하기 때문에, Display_SvM을 통해서 데이터를 받게 되며, 이 데이터는 ODSS에 저장되었다가, Display_SpM에 의해서 주기적으로 GUI에 출력하게 된다. 다음의 <그림 n>은 요구 조건이 반영된 GCU TMO 객체를 나타낸다.



< 그림 7 > 지상 통제 시스템 TMO객체의 설계

각각의 설계된 UAV TMO객체와 GCU TMO객체는 다음 < 그림 8 >과 같이 상호 데이터를 교환하게 된다. UAV TMO에서는 GCU SpM에서 비행 데이터를 GCU TMO로 Gate를 이용해서 전송해 주며, GCU TMO에서는 Display_SvM에서 전송된 데이터를 받아 ODSS에 저장하게 된다.
GCU TMO에서는 Mission_SpM에서 미션 데이터를 UAV TMO의 GCU_SvM으로 전송해 주며, UAV TMO에서는 이를 받아서 자신의 ODSS에 저장하게 된다.

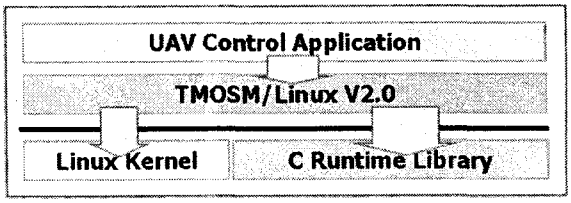


< 그림 8 > 설계된 UAV TMO객체와 GCU TMO객체

4. 무인항공기 제어시스템의 구현 및 테스트

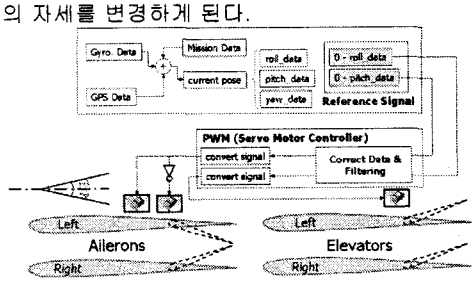
4.1 무인항공기 제어시스템의 구현
< 그림 9 >에서와 같이, UAV 제어시스템은 리눅스 커널과 사용자 어플리케이션 사이에 있는 미들웨어인 TMOSM/Linux V2.0을 사용하여 개발되었으며,

TMOSM/Linux는 TMO모델에서 정의된 실시간 속성들을 실제 실행시간에 보장해 주기 위한 미들웨어 이다.



< 그림 9 > TMO 제어소프트웨어의 동작 환경

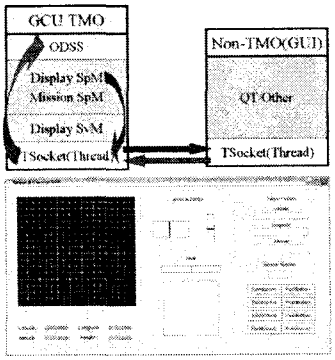
무인항공기 제어시스템은 < 그림 10 >과 같이 Gyro-Scope 데이터와 Mission Data, 그리고 GPS 데이터를 이용하여 정해진 임무를 수행할 수 있도록, 현 상태에 대한 정보를 기준으로 자세를 변경하는 데이터를 생성하여 PWM 모듈로 전송하여 서보 모터의 위상을 변경하여 비행기의 자세를 변경하게 된다.



< 그림 10 > 구현된 무인항공기 제어시스템의 제어 흐름

4.2 지상 제어시스템의 구현

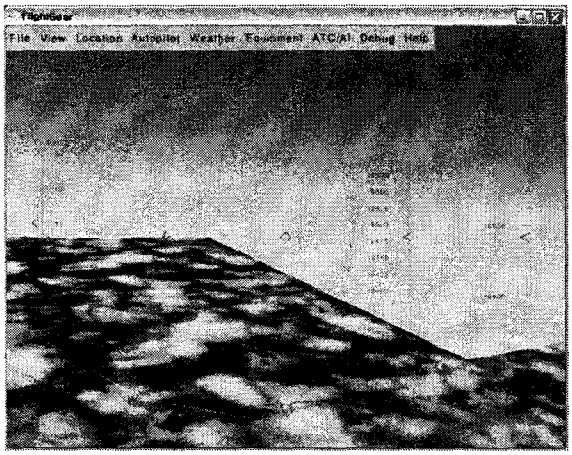
지상제어시스템에는 사용자로부터 임무를 입력 받는 부분이 있어, 이를 위한 GUI 파트가 있으며, 실시간과는 맞지 않는 GUI의 특성 때문에 TMO 미들웨어와는 IPC를 이용한 통신방법을 사용한다. 이를 위해 TMO 지원 소켓인 TSocket 인터페이스를 구현하여 사용하며, GUI는 QT를 사용하여 구현하였다. 다음 그림은 TSocket을 이용한 GCU TMO객체의 구조와 GUI의 관계를 보여준다.



< 그림 11 > TSocket을 사용한 GCU TMO와 GUI

4.3 모의 비행 시뮬레이터 FlightGear

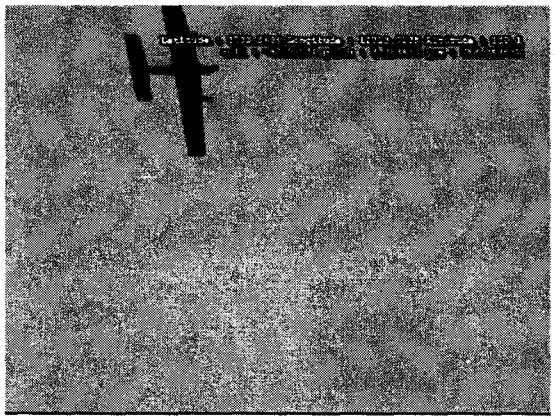
작성된 무인항공기 운용시스템을 테스트하기 위해서 가상의 비행환경을 제공해 주는 오픈 소스 기반의 FlightGear를 사용하였다. FlightGear에서는 가상의 비행환경을 만들어서 구현된 제어시스템으로 보내주게 되고, 제어시스템에서는 이 환경을 인식하여 필요한 데이터를 취득한 뒤, 임무에 맞는 반응을 다시 FlightGear로 보내 화면상에서 실제 비행과 흡사한 장면을 볼 수 있도록 하였다.



< 그림 12 > FlightGear를 사용한 무인항공기 제어시스템의 테스트

4.4 시험 비행 테스트

가상의 비행 실험 이외에도 실제 비행환경에서 제어시스템이 문제없이 작동하는지 확인하기 위해, On-Flight Simulation을 수행하였으며, 비행 중에도 데이터의 취득이나 동작에 문제가 없는 것을 확인하였다.



< 그림 13 > 실험 비행 장면

5. 결론 및 향후 과제

기존의 무인항공기 제어시스템들은 대부분 하드웨어 의존적으로 개발되어 특정 시스템에서는 잘 동작하지만, 다른 하드웨어에서는 동작하지 않는 문제점이 있었다. 따라서 기존에 작성된 코드를 다른 무인항공기 제어시스템에 재사용하는 것은 매우 힘든 작업이었다. 또한 처리장치의 사용전력이나 크기의 문제로 인해 저 사양의 하드웨어를 사용하여 다양한 임무 수행을 위한 복잡한 소프트웨어의 작성이 어려웠다.

본 논문에서는 이에 따른 해결방안으로 고성능의 소형 처리장치를 사용하였으며, 실시간 객체 모델인 TMO모델을 이용하여 다양함 임무를 수행할 수 있는 무인항공기 제어시스템을 설계 및 구현하고, 이를 이용해 재사용 가능한 무인항공기 전체 운용 시스템을 위한 프레임워크를 구축하였다. 그리고 모의 환경에서의 테스트와 실제 비행 실험을 통해 구현한 제어 시스템의 성능을 평가하여, 무인항공기와 같은 고 정밀 시스템의 설계에 TMO모델의 적용 가능성과 지원 미들웨어인 TMOSM/Linux의 성능을 확인하였다.

향후 고장에 대한 허용감내 기능을 탑재하여 보다 안정적인 무인항공기의 제어시스템을 기대할 수 있겠다.

6. 참고문헌

[1] Kim, K.H., " Real-Time Object-Oriented Distributed Software Engineering and the TMO Scheme" , Int' l Jour. of Software Engineering & Knowledge Engineering, Vol. No.2, April 1999, pp.251-276

[2] T. J. Koo. J. Liebman. C. Ma. and S. Sastry.: Hierarchical approach for design of multi-vehicle multi-modal embedded software. Proc. EMSOFT, LNCS 2211, pages 344-360. Springer-Verlag, October 2001.

[3] T. John Koo, Judith Liebman, Cedric Ma, Benjamin Horowitz, Alberto Sangiovanni-Vincentelli, Shankar Sastry: Platform-based embedded software design and system integration for autonomous vehicles. Proceedings of the IEEE 91(1): 198-211 (2003)

[4] J.S. Jang and C.J. Tomlin.: Longitudinal Stability Augmentation System Design of the DragonFly UAV using a Single GPS Receiver. 2003 GNC Conference

[5] 박춘배, 최기영, " 무인항공기 탑재 전자 장치" , 제어 자동화 시스템 공학회지 제7권, 제 5호 2001.

[6] 이영재, 황희철, " 제 2회 한국 로봇항공기 경영대회 자동비행구현 기술보고서" , May 50, 2003.

[7] 이은파, " 리눅스 기반 고성능 TMO 미들웨어 아키텍처의 설계 및 구현" , Feb, 2006.