Safety critical 시스템의 안정성 확보를 위한 정형 모델링 및 검증

(Formal Modeling and Verification of Safety Critical System)

Department of Computer Science & Engineering, Konkuk University

요 약

최근 인공지능, 로봇공학, IoT 기술들이 발전함에 따라 분야를 막론하고 실제 물리세계와 컴퓨팅의 영역이 겹쳐지고 있다. 이는 우리 생활에서 큰 편리함을 가져다 주었지만 한편으로는 잘못된 제어로 인해서 실제 물리세계의 요소들 즉, 사람이나 특정 물체들에 심각한 손상을 야기할 수 있는 문제의 요소를 내포하고 있다는 것을 의미한다. 즉, 이러한 페러다임에서 우리가 알고 있는 대부분의 로봇이나 IoT 제품들은 잠재적 위험요소가 있는 Safety Critical System이라고 할 수 있다. 일반적으로 가장 접하기 쉬운 로봇은 드론이라고 할 수 있는데, 드론은 항공촬영, 정찰, 감시 그리고 레저에 이르기까지 많은 분야에서 유용하게 사용되어지고 있다. 최근 드론에 사용되는 MCU는 많은 센서와 통신 장치를 하나의 MCU로 만들 수 있을만큼 기술이 발전했고, 내부에 작성될 소프트웨어에 따라서 작동을 하게 된다. 즉, 하나의 MCU로 주위의 많은 모듈을 제어하고 또 데이터를 획득하기 위해서는 알고리즘이 매우 주의 깊게 작성되어야 한다는 것을 의미한다.

즉, 알고리즘이 올바른지를 검증을 해야 한다는 것을 의미한다. 이는 Formal Modeling을 통한 Verification으로 보장할 수 있다. 특히 모델링 및 검증 도구 중에서 UPPAAL은 timed automata 이기 때문에 실시간성을 반영할 수 있어서 보다 정확한 검증이 가능하다.

1. 서 론

로봇은 실제 물리세계와 컴퓨팅의 조합으로 만들어져 있다. 현재 대부분의 로봇은 내부에서 모든 것을 알아서 처리하는 Closed System이 아니라 어떠한 방식이든 외부의 신호를 받고 그 신호에 따라 특정 제어를 하도록 설계가 된 Connected System 이다. 이는 단순한 제어뿐만이 아니라 실제 물리세계에 손상을 야기할 수 있는 문제가 발생됐을 때 이 행동을 중지를 시킬 수 있도록 한다. 즉, 우리가 알고 있는 로봇은 Safety Critical System 이라고 할 수 있는데 이러한 특성을 보장해주는 것이 통신과 제어라고 할 수 있다.

최근 드론[1]은 많은 분야에서 유용하게 사용되어지고 있으며 주위에서 쉽게 찾아볼 수도 있다. 이러한 드론의 눈부신 발전은 최근 MCU의 성능이 바약적으로 좋아진것에 기인한다. 즉, 하나의 칩에서 많은 연산을 수행할 수 있고 많은 센서와 장치도 제어할 수 있게 되었다. 이러한 발전에서 값싸고 작은 드론이 개발될 수 있었던 것이다. 하지만 하나의 MCU에 많은 센서와 통신 장치를 부착했기 때문에 다수의 모듈과 시리얼 통신을 동시에 수행해야 한다는 점이 문제가 될 수 있다.

예를 들어 앞서 설명한 대로 로봇의 한 종류인 드론은 자율 비행이 가능할지라도 사용자의 통제를 받을 수 있어야만 한다. 이러한 명령은 일반적으로 통신 모듈을 거쳐 시리얼 통신으로 MCU에 전달되며 이 제어가 실제로 드론의 행동에 반영이 되어야만 한다.

통신 모듈과 센서는 일반적으로 MCU와 시리얼 통신을 한다. 즉, 각각의 시리얼 포트에 물리적으로 할당이 되고 데이터를 송수신 하는 구조이다. 시리얼 통신의 데이터 형태는 센서 제조사나 제어 소프트웨어에서 정의한 패킷의 양

1) 조종사 없이 무선전파의 유도에 의해서 비행 및 조종이 가능한 비행기나 헬리콥터 모양 등의 무인항공기(UAV : Unmanned Aerial Vehicle)의 총칭 은 정찰, 감시, 전투와 같은 임무에 유용하게 활용되어지고 있다.

식대로 바이트 단위의 데이터를 전달하는데 MCU는 언제 들어올지 모르는 바이트 단위의 무작위 데이터를 온전한 패킷의 형태로 만들어 내고 이것이 정말 온전한지를 확인해야만 한다. 이러한 상황은 복합적이기 때문에 막연한 아이디어로 작성한 코드는 Safety Critical System의 가장 핵심적인 통신과 제어 알고리즘이 확실한지 확인을 해야 한다.

이 논문에서는 Formal Modeling을 통한 Verification으로 알고리즘을 검증 할 것이며 실시간성을 반영하기 위해서 UPPAAL이라는 도구로 Timed Automata를 작성할 것이다.

2. 관련 기술 및 비행체 모델링

이 절에서는 드론 시스템의 모델링 및 검증에 사용될 UPPAAL에 대한 배경 설명과 검증 대상이 될 매커니즘에 대해서 설명한다.

2.1 UPPAAL

UPPAAL은 1995년에 Aalborg 대학교와 Uppsala대학에서 개발 및 공개된 Timed Automata 기반의 실시간 시스템으로써 모델링, 시뮬레이션 및 검증을 위해 개발된 모델 체킹 도구이다. UPPAAL의 가장 큰 특징은 Timed Automata를 GUI 의 형태로 작성 가능하다는 점인데, UPPAAL은 실시간 시스템 검증을 위해 필요한 기술적 요소가 모두 구현되어 있다.

UPPAAL의 모델 체커는 Automata와 declaration에서 작성한 시스템이 만족해야 할 속성에 대해서 검증한다. UPPAAL의 시뮬레이터는 GUI 기반에서 작성된 Automata와 Declaration을 시뮬레이션 해주며 시스템에 대한 동작 및 상태를 실시간으로 확인 할 수 있도록 한다. Timed automata를 기반으로 한 시스템 모델링은 시간개념을 반영할 수 있으며 이는 실시간 시스템인 로봇(드론)을 모델링하여 동적인 실행 과정을 확인할 수 있다.

UPPAAL은 독립적인 프로세스(모듈)단위로 설계할 수 있는데, 각각의 프로세스(모듈)는 채널로 동기화 될 수 있고 전역 변수등으로 데이터를 공유할 수도 있다. 채널 통신은 동기화 방식이기 때문에 송수신 이벤트로 구성이 된다. 즉, 어디선가는 송신 이벤트를 발생을 해야 수신 이벤트가 발생된다는 것이다. 이는 MCU와 센서, 통신 모듈 등에서 발생되는 메커니즘과 상당히 흡사하다고 할 수 있다.

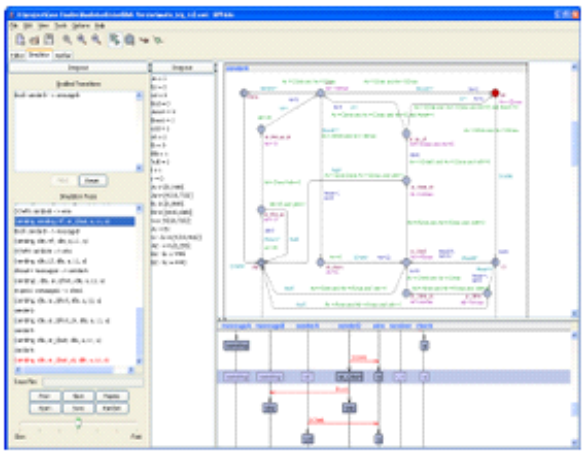

그림 1 UPPAAL

본 논문에서는 검증하고자 하는 드론 제어 시스템은 센서와 외부 제어 신호가 Serial Communication 기반으로 작동을 하고 있다. 즉, 단순히 기능적 입출력 뿐만 아니라 실제 센서나 모듈이 발생시키는 데이터 전송률에 따른 통신 특성을 반영해야 한다. 이는 실제 알고리즘이 시간 제약 사항을 충족하고 있음을 검증할 필요가 있다는 점을 의미한다.

드론 제어 시스템은 일반적인 SW 시스템과 달리 알고리즘이 올바르게 작성 되었는지 직접 관찰하기가 어렵다. 그리고 물리적인 환경과 매우 복잡하게 상호 작용을 하기 때문에 오류가 발견되었다 할지라도 그러한 상황에 대해 재현하는 것이 쉽지 않다.

드론 제어 시스템에서 기능적 입출력, 즉, 단순히 입력과 출력이 되어서 제어가 된다는 차원의 문제가 아니라 특정 상황에서 발생될 수 있는 시스템 한계나 버그를 찾아내는 것이 더 중요한 의미를 가지고 있다. 이러한 문제가 완벽하게 해결된 시스템이 진정한 Safety Critical System 이라고 할 수 있다. 이러한 드론 제어 시스템의 기본이 되는 Serial Communication이 가진 특수성(시간 제약, 바이트 단위의 전송)에서 제어 시스템 행위 확인하고 예상치 못한 오류로 인해 사고가 발생되지 않도록 UPPAAL을 사용하여 제어 시스템의 안정성을 향상시킬 수 있는 방법을 연구하고자 한다.

2.2 Serial Communication on UAV System

본 논문에서는 검증할 드론 제어 시스템은 하나의 MCU를 중심으로 드론의 자세를 전달하는 센서, 그리고 제어 패킷을 전달하는 RF Module의 범주로 한다. 센서와 RF Module은 바이트 단위의 신호를 MCU에 전달하는데 MCU에서 Controller에 대항하는 알고리즘은 센서

와 RF Moduel에서 전달 받은 데이터를 이용하여 UAV의 Motor를 제어하도록 설계 되어 있다.

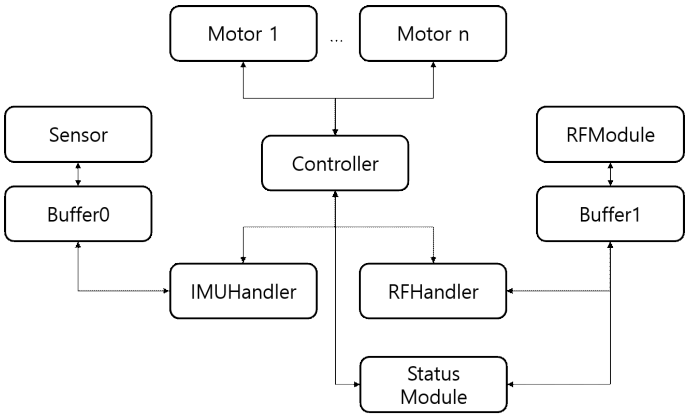

그림 2 UAV Control Program

드론 제어 시스템은 위 그림과 같이 구성 되었다. 크게 Sensor, RF Module, Controller로 구성하였으며 Controller는 UAV 제어를 위한 신호를 보내는 Module, 수신된 신호의 유실 확인 및 수신된 신호를 Motor에 전달하는 Module이 포함되어 있다. motor는 수신된 신호를 전달받아 실제 동작을 하는 역할을 한다.

Sensor 혹은 RF Module에서 수신된 신호는 각각 Buffer0, Buffer1에 Byte 단위로 수신하게 되며, Byte단위로 수신된 데이터는 각각의 구분 된 공간에 저장 된다. 각각의 공간에 축적된 신호는 Handler를 통해 결합되며 결합 시 신호의 유실유무를 확인한다. 신호가 유실되었음을 확인하면, 수신된 데이터를 폐기하고 다시 새로운 신호를 수신하게 된다. 수신이 완료된 신호는 Controller를 통해 Motor 제어에 반영되거나 IMU Data로 저장되어 RF 모듈을 통해 그라운드로 송신된다.

저장 된 IMU Data는 Status Module을 통해 Buffer1에 전달되는데, 이는 RF Module을 통해 그라운드로 송신하기 위함이다. 한편, RF Module에서 Motor까지의 Data전달이 되는 수신 동작과 Status Module에서 RF Module까지 전달되는 송신 작업은 지속적으로 동시에 발생하게 된다. 이 때, Data 전달 과정 중 RF Data와 IMU Data의 훼손을 방지하기 위해 송신과 수신중 하나의 동작이 Buffer1을 점유 시 다른 동작은 Buffer1 사용이 중지된다.

3. 시스템 모델링

3.1 모델 개요

이 제안한 시스템을 정형검증하기 위해 우리는 앞 절에서 소개한 UPPAAL을 이용하여 UAV 시스템을 모델링 했다. 이 시스템을 위한 모듈로 IMU 데이터를 발생시키는 Sensor와 Buffer0, RF Data를 발생시키는 RF Module과 Buffer1, Buffer0으로부터 데이터를 읽어 패킷의 형태를 검증하는 IMU Handler, Buffer1로부터 데이터를 읽어 패킷의 형태를 검증하고 Emergency 상태를 확인하는 RF Handler를 정의하였다. Sensor의 경우 MCU와 시리얼 포트에 물리적으로 할당 된 상태로 통신하기 때문에 버퍼 데이터를 잃을 확률이 없지만 RF 통신의 경우 무선으로 통신하기 때문에 버퍼 데이터를 잃을 수 있다. 이를 반영하여 RF Module을 1%의 확률로 Buffer로부터 값을 가져오지 못하도록 모델링하였다. 또한 RF Handler로부터 발생한 데이터를 Motor에 전달하고 IMU Handler로부터 발생한 데이터를 Status Module로 전달하는 Controller, Controller로부터 받은 데이터를 Buffer1을 거쳐 RFModule을 통해 그라운드로 전달하도록 하는 Status Module를 정의하였다. 이 모듈 간 통신은 동기화 여부에 따라서 채널, 혹은 전역 변수를 통해 이루어지며, RF Module 및 IMU Module 각각에 포함된 클럭이 본 모델의 시간요소에 해당하며 한 클럭 주기를 1 마이크로세컨드(ms)로 간주하여 구성하였다.

3.2 Sensor

Sensor는 주기적으로 Sensor 데이터를 발생시킨다. 본 모듈에서 발생되는 데이터는 UAV에 장착된 다양한 센서들의 데이터를 포함한다. 한 주기를 3클락(3ms)로 모델링 하였으며, 한 주기에 한 바이트에 해당하는 데이터를 발생시킨다. 발생한 바이트는 시작 플래그를 포함해 IMU 패킷을 구성하는 요소가 된다. IMU 패킷은 8개의 바이트로 구성되어 있어 여덟 번의 주기가 지날 때 마다 한 패킷의 데이터가 생성된다. 매 주기 마다 들어오는 데이터는 버퍼에 일시 저장되며, bufferIMU 변수를 Enable 시킴으로써 버퍼에 저장된 데이터 가용성을 표현한다. 한편, 8개의 바이트가 정상적으로 IMU 패킷으로 구성되기 위해서는 각 바이트에 대한 순서 정보가 필요한 데, imuIndex 변수가 그 역할을 하게 된다.

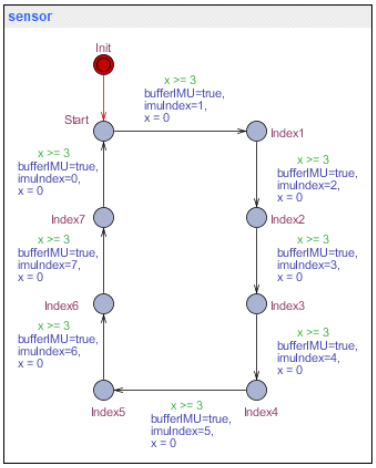

그림 3 Sensor Automata

3.3 buffer0

buffer0은 Sensor로부터 데이터를 IMU 패킷으로 변환하는 과정에서 Sensor와의 공유 변수를 통해 Sensor로 부터의 데이터 생성 여부를 확인하고, Controller로부터 IMU 인터럽트 신호를 확인하였을 때 Sensor로부터 한 바이트의 데이터를 읽는다. 데이터를 읽을 때에 인덱스 정보를 저장하여 저장된 인덱스 정보에 대해 IMU 핸들러가 참조할 수 있도록 채널 통신을 통해 동기화가 이루어진다.

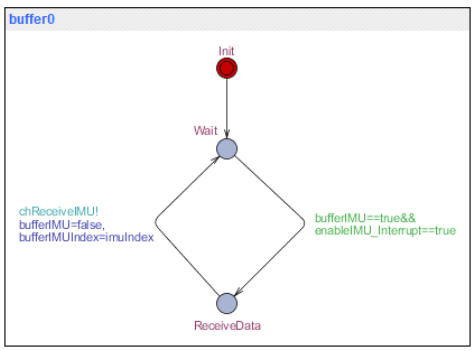

그림 4 Buffer0 Automata

3.4 IMU Handler

IMU Handler는 buffer0과 채널로 동기화되어 있으며 Buffer0이 한 바이트를 읽을 때마다 동작한다. 한 바이트씩 읽을 때마다 본 모듈은 버퍼가 읽어 들인 데이터의 인덱스를 검증하고 패킷이 완성되었을 때 컨트롤러와 채널을 통해 통신하는 역할을 한다. 본 모듈은 시작 플래그가 나타나는 시점을 체크하는데 시작 플래그가 올바른 위치에 들어오지 않으면 처음부터 다시 패킷을 구성한다.

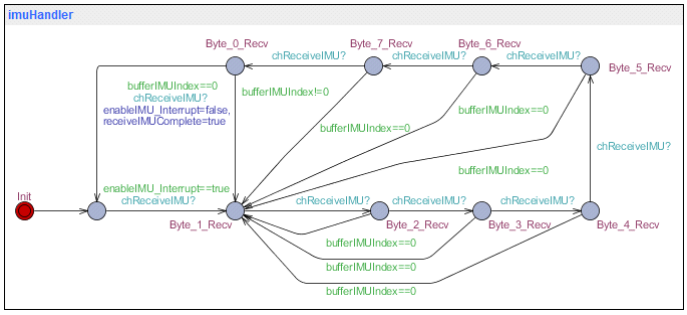

그림 5 IMU Handler Automata

3.5 RF Module

RF Module은 주기적으로 RF 데이터를 발생시킨다. 실제 RF 통신 주기를 반영하여 한 주기를 50클락(50ms)로 모델링하였다. 한 주기에 한 바이트에 해당하는 데이터를 발생시킨다. 발생한 바이트는 시작 플래그를 포함해 RF 패킷을 구성하는 요소가 된다. RF 패킷은 8개의 바이트로 구성되어 있어 여덟 번의 주기가 지날 때 마다 한 패킷의 데이터가 생성된다. 매 주기 마다 들어오는 데이터는 버퍼에 일시 저장되며, bufferRF 변수를 Enable 시킴으로써 버퍼에 저장된 데이터 가용성을 표현한다.

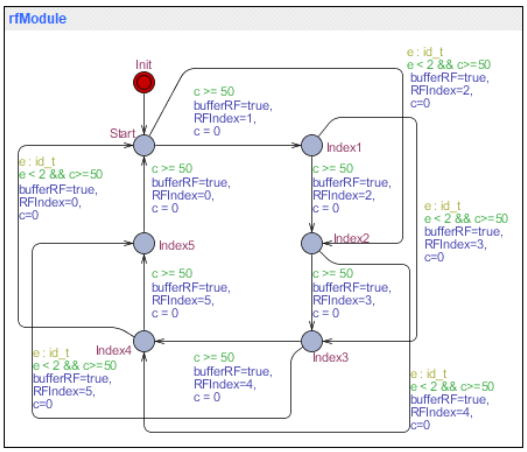

그림 6 RF Module Automata

3.2절의 Sensor 모듈에서 지적된 바와 같이 패킷이 정상적으로 구성되기 위하여 각 바이트에 대한 순서정보가 필요한 데, wifiIndex 변수가 그 역할을 하게 된다. 한편, Sensor 모듈은 MCU에 물리적으로 연결되어있어 버퍼의 손실이 없다고 할 수 있으므로 이러한 부분이 고려되지 않았지만, 이 RF Module은 무선통신에 의해 버퍼가 손실 될 수 있음을 고려하여 1%의 확률로 버퍼를 손실하는 경우를 모델링하였다. 이 경우도 마찬가지로 패킷이 잘못 완성 되었을 때 다시 패킷을 생성하도록 하였다.

3.6 Buffer1

buffer1은 RF Module로부터 데이터를 RF 패킷으로 변

환하는 과정에서 RF Module과의 공유 변수를 통해 RF Module로부터의 데이터 생성 여부를 확인하고, Controller로부터 RF 인터럽트 신호를 확인하였을 때 RF Module로부터 한 바이트의 데이터를 읽는다. 데이터를 읽을 때에 인덱스 정보를 저장하여 저장된 인덱스 정보에 대해 RF 핸들러가 참조할 수 있도록 채널 통신을 통해 동기화가 이루어진다.

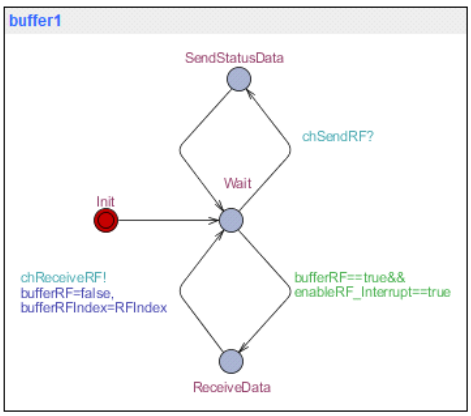

그림 7 Buffer1 Automata

단, Buffer0과는 다르게 데이터를 수신하는 역할만 수행하지 않고, 데이터를 송신하는 역할도 추가된다. Controller로부터 IMU Data를 송신한다는 메시지를 채널을 통해 받을 수 있으며, 이 경우 RF Data 수신은 중단되고, Status Module로부터 전달받은 IMU Data를 송신하게 된다.

3.7 RF Handler

RF Handler는 buffer1과 채널로 동기화되어 있으며 Buffer1이 한 바이트를 읽을 때마다 동작한다. IMU Handler와 마찬가지로 한 바이트씩 읽을 때마다 본 모듈은 버퍼가 읽어 들인 데이터의 인덱스를 검증하고 패킷이 완성되었을 때 컨트롤러와 채널을 통해 통신하는 역할을 한다. 또한 시작 플래그가 나타나는 시점을 체크하여 시작 플래그가 올바른 위치에 들어오지 않으면 처음부터 다시 패킷을 구성한다. 한편, RF Handler를 통해 만들어지는 패킷은 일반 데이터 패킷과 Emergency 패킷으로 나누어진다. 일반 데이터 패킷과는 다르게 Emergency 패킷은 언제 발생할지 모르기 때문에 한 바이트씩 읽을 때마다 Emergency 패킷의 시작 플래그가 나타나는지 매번 확인하고, Emergency 패킷의 시작플래그가 발생한 경우 이후 바이트들이 Emergency 패킷의 바이트 열인지 확인하는 과정을 거친다. 만약 Emergency 패킷이 발생한 경우 Controller에 채널을 통해 Emergency 상태를 알린다.

3.8 Controller

Controller는 UAV Control System의 메인과 같은 역할을 한다. IMU 패킷의 수신, 변환 및 송신과정, RF 패킷의 수신, 변환 과정, Emergency 상태 등을 제어한다. 또한 각 프로세스의 수행 시간을 고려하여 각 State 마다 다른 크기의 클럭을 설정하여 Time Automata를 모델링하였다. 초기 State부터 제어가 시작되었을 때, 본 모듈은 IMU 패킷의 여부를 체크하며 IMU 패킷이 완성 되었을 경우 IMU Convert를 실시하고 그렇지 않으면 다음 프로세스로 넘어간다. 다음 프로세스는 RF Module 대한 핸들링인데 RF Module의 경우 RF 데이터를 수신하거나 Status Module을 통해 상태 데이터를 송신해야 하므로 제어 주기 3번 당 2번은 수신하는 작업을 수행하여 완성된 RF 패킷에 대한 RF Convert 작업을 수행하고 나머지

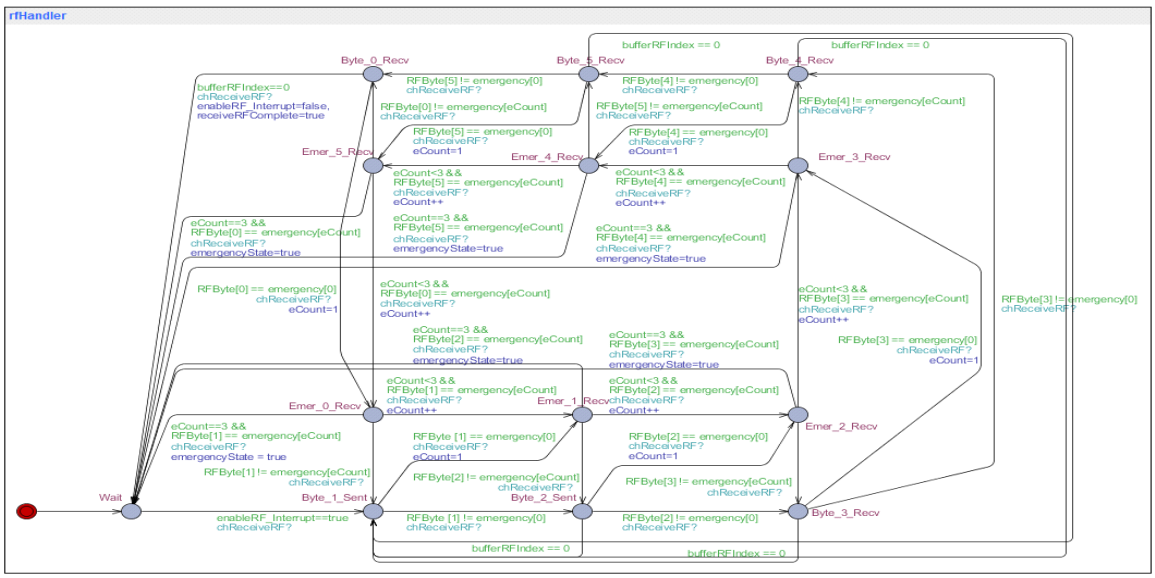

그림 8 RF Handler Automata

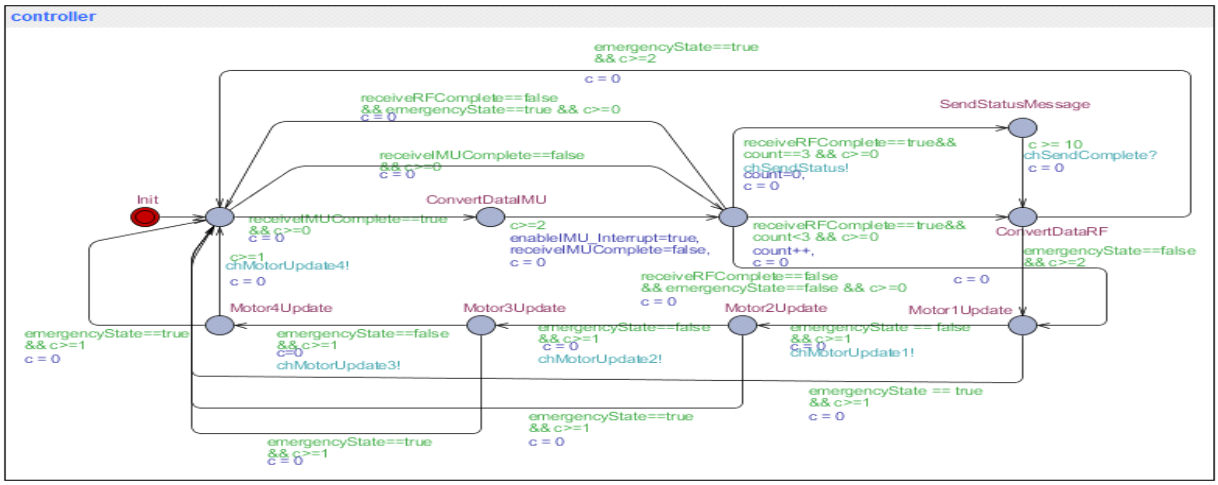

그림 9 Controller Automata

1번은 현 비행체의 상태 정보를 RF Module로 송신한다. 이러한 과정이 완료되면 그 이후에는 RF Module로부터 수신한 데이터를 이용해 4개 모터에 대하여 일괄적인 제어작업이 시작된다. 그 이후에는 다시 IMU 패킷을 확인하고 Convert하는 과정으로 진행한다. 또한 매번 Motor 제어를 하기 전에 각 Motor State마다 Emergency State를 확인하여 Emergency State인 경우 Motor 제어를 중단 하도록 설계하였다.

3.9 Status Module

Status Module은 Controller에 의해 Controller의 3주기당 한 번씩 호출되는데, 이 때 Status Module을 이용해 전송할 패킷을 만들고 RF 모듈을 제어한다. 4개의 바이트로 Status 패킷을 바이트 단위로 Buffer1로 전송하여 RF Module을 통해 UAV의 상태를 전송하도록 모델링 하였다. 또한 Status 패킷 전송이 완료된 후 Status 패킷 전송 완료 여부를 Controller에 알려 다음 작업을 수행할 수 있도록 한다.

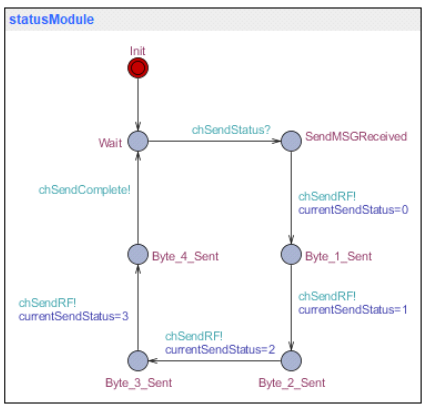

그림 10 Status Module Automata

3.10 모터 모듈

각각의 Motor는 Controller와의 채널 통신을 통해 Motor 상태 값을 업데이트하며, Emergency 상태에서는 Motor 제어가 중단 된다.

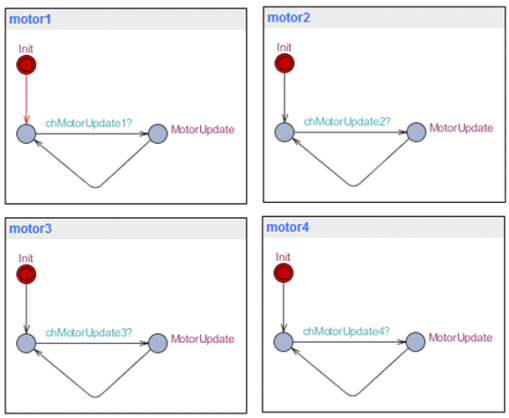

그림 11 Motor Automata

4. 시스템 검증 및 결과

모델링한 UAV Control 시스템의 검증을 위해 UPPAAL에서 제공하는 Verifier를 사용하였다. Critical Safety 시스템을 검증하기 위해 7가지의 Property를 수립하였으며, 이 Property는 Critical Safety 시스템의 안정성을 위해 보장하기 원하는 최소한의 요구사항들이다. 검증을 위해 UPPAAL Verifier가 요구하는 CTL 표기법에 따라 프로퍼티를 작성하였다.

- A[] not deadlock : 데드락이 발생하지 않는다.
- E<> !((receiveRFComplete==true) == (RFIndex!=5)) : 5Byte의 RF 패킷을 받지 않으면 RF 데이터 패킷은 만들어지지 않는다.
- EE<> !(controller.ConvertDataRF

== (receiveRFComplete==false)) : RF 데이터 패킷 전송이 완료되기 전까지 RF 데이터의 Convert가 일어나지 않는다.

- A[] ((receiveRFComplete==true) imply controller.ConvertDataRF) : RF 패킷을 모두 수신한 뒤에 RF 패킷을 Convert 한다.
- E<> !(((emergencyState == true) == controller.Motor2Update) imply controller.Motor3Update) : emergencyState가 발생한 이후에는 Motor를 제어하지 않는다.
- E<> !(controller.SendStatusMessage == (enableRF_Interrupt==true)) : RFInterrupt가 Enable인 상태에서는 Status 메시지를 전송하지 않는다.
- E<> ! (rfModule.c<50 && (rfModule.Index1 imply rfModule.Index2)) : 50 Clock 이 지나기 전에 RFModule의 수신 Index가 변하지 않는다.

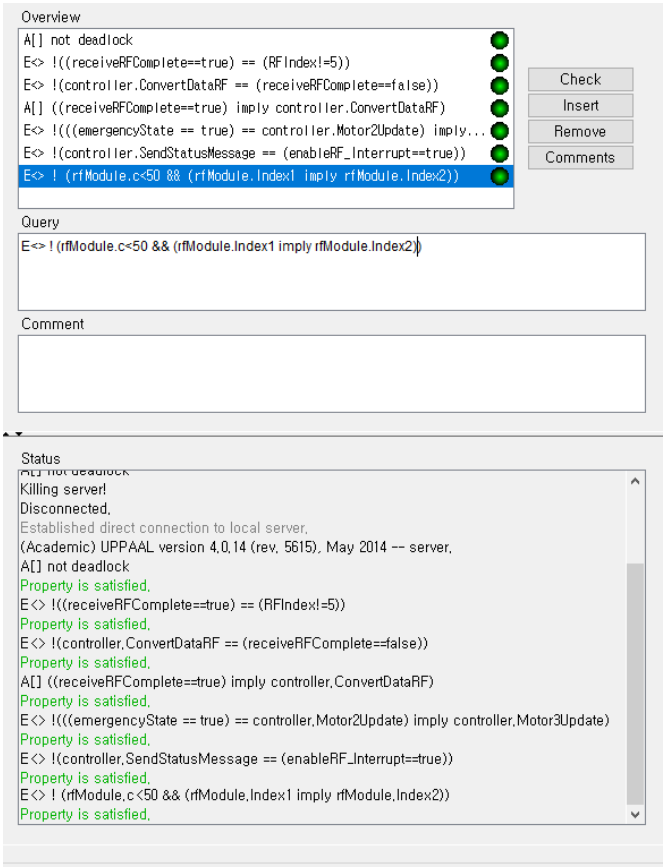

그림 12 UPPAAL의 Verifier를 이용한 검증 결과

본 논문은 위 7가지 Property에 대하여 검증을 실시하였고 우리가 모델링한 시스템 상에서 제시된 모든 Property가 만족되어짐을 확인하였다.

5. 결론

본 연구에서는 UAV System의 Control의 알고리즘 중에서 가장 핵심적인 부분을 모델링하고 있다. 또한 UPPAAL을 이용해 그 알고리즘이 모든 경우에서 올바른 패킷으로 수신함을 보장하는지 확인하였고, RF Module이 수신, 송신의 기능이 겹치지 않고 동작함을 확인하였다.

우선, 이 연구를 진행할 때 가장 중요하게 생각했던 부분은 모델링이 실제의 시스템과 얼마나 일치하느냐는 점이었다. 이를 위해 무선 통신인 RF 모듈에서 일정 확률로 버퍼를 손실한다는 점을 함께 모델링하였고, 실제 MCU로 사용된 FPGA기반 STM32 M3 모듈의 데이터시트를 참고하여 시리얼 통신의 인터럽트 방식을 그대로 구현하였다. 또한 하드웨어적으로 모듈을 구분하였고 channal을 형성하여 동기화 시켰다. 이 부분이 인터럽트의 가장 핵심적인 하드웨어적인 개념을 충분히 반영했기 때문에 이 연구의 결과가 신뢰성이 있다고 말할 수 있다. 또한 메인 모듈과 같은 Controller에서는 바이트 단위로 데이터를 쏟아내는 주위 모듈의 데이터들을 각각의 인터럽트 채널로 나눠 수집하여 패킷을 구성하는 과정 또한 실제의 시스템의 알고리즘을 반영하여 구성하였다.

철저하게 모델링 한 후 UPPAAL이라는 검증 도구를 이용하여 Timed Automata를 작성하여 UPPAAL에서 제공하는 Verifier를 이용하여 CTL 표기법에 따라 Property를 작성하고, 이에 대한 Verification을 진행하였다.

Verification 결과로 모든 Property를 만족하였고, 이를 통해 다양한 변수 속에서 가능한 모든 경우에 대해서 정확하고 완전한 패킷을 생성함과 Emergency 상태를 확인함을 보장할 수 있었다. 즉, 알고리즘이 잘 작성 되었다는 것을 신뢰성 있게 보장할 수 있게 되었다.

참 고 문 헌

[1] Behrmann, Gerd, Alexandre David, and Kim Larsen. "A tutorial on uppaal." Formal methods for the design of real-time systems (2004): 33-35.

[2] Bengtsson, Johan, et al. "UPPAAL—a tool suite for automatic verification of real-time systems." Hybrid Systems III (1996): 232-243.

[3] Bengtsson, Johan, and Wang Yi. "Timed automata: Semantics, algorithms and tools." Lecture Notes in Computer Science 3098 (2004): 87-124.

[4] Hessel, Anders, et al. "Testing real-time systems using UPPAAL." Formal methods and testing (2008): 77-117.