Technical Paper
Journal of Aerospace System Engineering
Vol.18, No.4, pp.102-108 (2024)
EISSN 2508-7150
http://dx.doi.org/10.20910/JASE.2024.18.4.102

항공기용 터보팬 엔진 FADEC의 소프트웨어 미들웨어 아키텍처에 관한 연구

이창열[1,†] · 조영호[1] · 임익찬[1] · 권기혁[1] · 김중회[2] · 나규진[2] · 장호연[3]
[1]LIG넥스원
[2]국방과학연구소
[3]한화에어로스페이스

A Study on the Software Middleware Architecture of Turbo Fan Engine FADEC for Aircraft

Changyeol Lee[1,†], Youngho Cho[1], Ikchan Lim[1], Kihyuk Kwon[1],
Junghoe Kim[2], Gyujin Na[2] and Hoyeon Jang[3]
[1]LIGNEX1
[2]ADD
[3]Hanhwa Aerospace

Abstract

With the recent increase in the development of domestic independent turbofan engines for aircraft, there is a need to develop software for FADEC(Full Authority Digital Engine Control) with real-time fault diagnosis functions to enhance fuel efficiency, engine performance, and reliability. As engine control algorithms become more sophisticated, software is being developed using Model-Based Design(model-based development) methods. This paper introduces the Middleware architecture of FADEC(Full Authority Digital Engine Control), which connects hardware with Model-Based Design(model-based development) software. Given the high reliability and safety required for turbofan engines in aircraft, the design complies with DO-178C[1] International Airborne Systems and Equipment Certification Guidelines.

초 록

최근 항공기용 터보팬엔진의 국내 독자 개발 증가에 따라 연료 효율, 엔진 성능과 신뢰성 향상을 위한 실시간 고장진단 기능을 갖춘 FADEC(전자식통합엔진제어장치)의 소프트웨어 개발 필요성이 대두되고 있다. 엔진 제어에 관한 알고리즘은 고도화되고 복잡해짐에 따라 모델기반 개발 방식을 사용해 소프트웨어를 제작하는 추세이다. 본 논문에서는 하드웨어와 모델기반 개발 소프트웨어를 연결시킬 FADEC(전자식통합엔진제어장치)의 미들웨어 아키텍처를 소개한다. 항공기용 터보팬엔진의 높은 신뢰성과 안전성을 고려하여 DO-178C[1] 국제감항인증 가이드라인에 따라 설계했다.

Key Words : Aircraft Engine(항공기 엔진), FADEC(전자식통합엔진제어장치), Redundancy(이중화), Model-Based Design(모델기반 개발)

1. 서 론

FADEC(전자식통합엔진제어장치)은 엔진전자제어장치(Electronic Engine Controller, EEC)로 불리는 디지털 컴퓨터로 구성된 시스템으로 엔진 성능을 제어하는 모든 기능들과 연관되어 있다[2]. EEC는 엔진을 디지털 방식으로 제어하거나, 수동 조작이 가능한 장치를 말한다. EEC를 포함하고 있는 FADEC은 트로틀 레버위치, 엔진 온도 그리고 엔진 압력 등 다양한 엔진센서 신호를 처리하고 제어한다[2, 3]. FADEC은 엔진센서 신호를 입력받고 연료, FIGV(Fan Inlet Guide Vane), 노즐목을 제어한다. 이를 통하여 엔진의

Received: May 10, 2024 Revised: Jul. 16, 2024 Accepted: Jul. 23, 2024
† Corresponding Author
Tel:+82-31-5178-4609,E-mail: changyeol.lee@lignex1.com

시동과 재시동에 관여하고 입력 받은 데이터를 통해 엔진의 상태를 모니터링한다. 이처럼 FADEC은 항공기 엔진 구동에서 큰 역할을 하기 때문에 FADEC의 안전성은 항공기의 안전성과 직결된다. 많은 경우 안전성을 위해 FADEC은 이중화 채널로 구성한다[4]. 각 채널은 자가 고장 진단을 통하여 채널의 안전성을 평가하여 안전한 채널을 통하여 엔진을 제어한다. 이러한 이중화 채널의 FADEC을 제어하기 위해서는 복잡하고 고도화된 알고리즘이 필요하게 되는데, 모델기반 개발 방식의 소프트웨어 제작을 통하여 복잡하고 고도화된 알고리즘을 간단히 모델화하는 추세이다. 이를 통하여 이해관계자들 간의 의사소통을 돕고 시스템의 신뢰성과 안전성을 높인다. 많은 경우에 감항인증을 획득한 모델기반 개발 Tool을 이용하여 설계하고 설계한 코드를 자동 생성하여 안전성과 생산성을 높인다. 하지만, 모델기반 개발 Tool은 시스템 요구에 따라 제작된 하드웨어의 제어 기능을 제공하기 어렵기 때문에 복잡성이 높은 알고리즘만을 모델기반 개발 방식으로 구현하고 하드웨어 종속적인 소프트웨어는 직접 구현한다[5, 6, 7].

본 논문에서는 모델기반 개발 방식의 소프트웨어와 하드웨어 종속적인 소프트웨어를 연결하는 이중화 채널의 미들웨어 아키텍처를 설계했다. 미들웨어 아키텍처는 신뢰성과 안전성을 고려하여 국제감항인증 표준인 DO-178C 가이드라인을 따라 계획, 요구사항 분석, 설계했으며 안전성 요구사항에 따라 이중화 설계를 적용하고 DO-178C 가이드라인의 권장사항에 따라 시스템 모니터링 기능을 수행하는 미들웨어 아키텍처를 설계했다[1].

본 논문 2장에서는 DO-178C 가이드라인에 따른 개발 절차와 DO-178C 가이드라인에 따른 권장 설계에 대해 설명하고 3장에서는 설계한 FADEC의 미들웨어 아키텍처를 상세히 소개한다.

2. DO-178C Level-A 인증을 고려한 설계

항공기용 소프트웨어 개발에 있어 신뢰성과 안전성을 위해 감항인증 기준이 필요하다. 이를 위해 미국연방항공국(FAA)에서 항공용 제품에 대한 소프트웨어 안전성 평가 기준으로 권장하는 DO-178C 국제감항인증 표준에 따라 미들웨어 아키텍처를 설계했다[1]. DO-178C 가이드라인의 라이프 사이클 프로세스는 Fig 1.과 같이 소프트웨어 개발의 계획, 개발, 통합 세 가지 라이프 사이클 프로세스로 구분한다. 각 단계별 신뢰성, 안전성 평가 및 라이프 사이클 전반의 추적성 관리를 통해 소프트웨어의 신뢰성, 안전성을 확보한다. 본 논문에서는 Fig 1.의 파란색으로 표기된 프로세스인 설계까지 연구했다. DO-178C 가이드라인의 계획 단계에서는 설계 보증 레벨(Design Assurance level, DAL)을 정하여 소프트웨어의 실패가 항공기에 어떠한 영향을 미치는 지 평가한다. Table 1에서 보이는 바와 같이 DAL-A는 항공기에 치명적인 실패를 유발하는 소프트웨어 레벨을 말하며 FADEC 소프트웨어의 실패는 항공기에 직결된 문제를 야기할 수 있으므로 DAL-A에 속한다. DAL A, B는 C, D와 다르게 요구사항에 따른 아키텍처 설계의 안전성 평가를 포함하며 컨트롤 플로우와 데이터 플로우을 고려한 안전성 설계

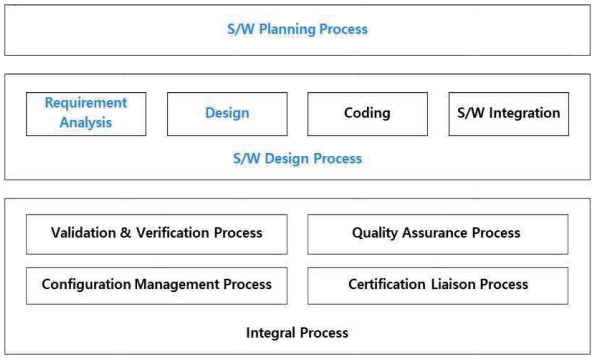

Fig. 1 DO-178C Life Cycle

Table 1 Design Assurance Level Table

Level	Probability Requirement	Effect on the aircraft
A	< 1.0-E09 PFH*	Prevents continued safe flight and landing
B	< 1.0 -E7 PFH*	Large reduction in functional capabilities or safety margins
C	< 1.0 -E5 PFH*	Significant reduction in functional capabilities or safety margin
D	< 1.0 -E3 PFH*	Slight reduction in functional capabilities or safety margin

*Probability of failure per hour

방안을 제시해야한다. DO-178C 가이드라인의 컨트롤 플로우는 각 기능마다 모듈을 분리하여 모듈간의 응집도를 높이고 결합도를 최소화하게 설계하는 것을 의미하며, 데이터 플로우는 각 모듈마다 데이터 인터페이스를 분리하여 데이터의 응집도를 높이고 결합도를 최소화하게 설계하는 것을 의미한다[1]. 또한 DO-178C 가이드라인에 따르면 시스템의 안전성을 모니터링하고 예기치 못한 고장에 대응할 수 있도록 권장한다. 따라서 본 논문의 아키텍처는 큰 관점에서 모델기반 개발 소프트웨어와 하드웨어 제어 소프트웨어의 구조를 명확히 나누고 그것을 연결시킬 미들웨어로 구성된 계층 구조의 FADEC 소프트웨어를 설계했으며, 미들웨어의 세부 설계는 컨트롤 플로우와 데이터 플로우의 응집도와 결합도를 고려하여 설계했다. 컨트롤 플로우 권고사항을 고려하여 엔진 운용주기에 따라 2가지 운용 모듈과 통신 인터럽트 모듈로 구분하여 설계했으며 이중화 채널의 데이터는 별도의 임계영역 버퍼를 설계하여 모듈 간 데이터 오염을 방지했다. 데이터 플로우 권고사항을 고려하여 각 채널과 모듈별 입출력 데이터를 분리하여 각 모듈의 데이터가 다른 모듈에 영향을 주지 못하게 설계했다. 또한 일정 시간의 입출력 데이터를 모니터링하여 입출력 신호의 오염이나 하드웨어적 고장에 대해서 적절한 대응을 할 수 있도록 설계했다.

3. 미들웨어 아키텍처 설계

3.1 FADEC 소프트웨어 계층 구조

본 논문에서는 FADEC 소프트웨어의 구조를 Fig 2.와 같이 각 기능별로 계층을 분리하여 계층별 독립적인 설계 및 개발을 할 수 있도록 구성했다. FADEC 소프트웨어는 Fig 2.와 같이 RTOS(Real Time Operating System), BSP(Board Support Package), 미들웨어, Application 소프트웨어의 계층 구조를 가진다. FADEC 소프트웨어의 실시간 응답을 보장하기 위해 TI사의 RTOS인 SYS/BIOS를 선택했다. BSP는 하드웨어를 초기화하고 제어하는 기능을 하며, 하드웨어에 종속적이다. 미들웨어는 BSP와 Application 소프트웨어를 연결하는 소프트웨어이다. 기능별 모듈의 입출력 신호처리 및 고장진단 모니터링, 이중화 채널 통신 및 인터럽트처리로 구성된다. 또한, 하드웨어에 종속적

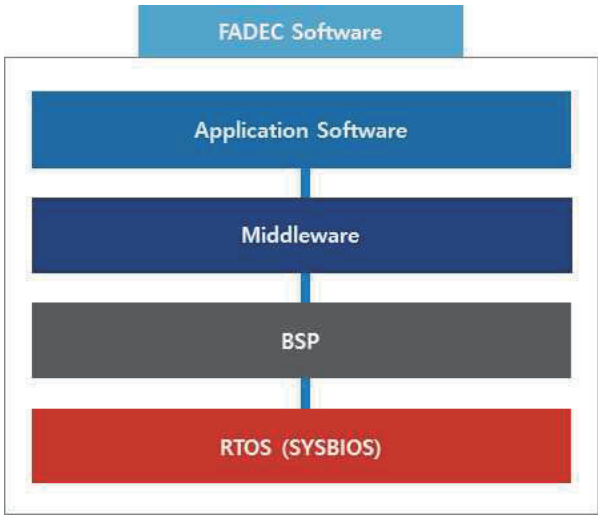

Fig. 2 FADEC Software Hierarchy

Table 2 Table of Functions and Feature by Layer

Layer	Functions	Feature
RTOS	Task Scheduling	Ensuring real-time performance
BSP	HW Initialization & Control	Dependent on hardware
Middleware	Signal processing, Monitoring, Redundant channel management	Not dependent on hardware Safety and reliability shall be guaranteed
Application Software	Comprehensive Engine Control Algorithm	Model-Based Design Complex Advanced

이지 않도록 설계했다. Application 소프트웨어는 모델기반 개발 방식으로 제작된 소프트웨어로 미들웨어에서 전달받은 데이터를 바탕으로 엔진을 운용하며 고장진단 결과에 따라 채널을 변경한다. Table 2는 각 계층의 기능과 특징을 요약한다.

미들웨어 소프트웨어는 FADEC에서 생성되는 하드웨어 데이터를 입력받고 관리하여 고장을 진단하고 이를 Application 소프트웨어 전달한다. 또한 Application 소프트웨어로부터 하드웨어 제어 명령을 받으며, BS

P 소프트웨어를 통해 엔진을 제어한다. 이를 통하여 Application 소프트웨어는 하드웨어 특성을 고려하지 않고 미들웨어에서 전달되는 신뢰성 높은 데이터를 사용하여 엔진을 제어할 수 있다. 따라서 Application 소프트웨어는 엔진 제어 알고리즘에 집중할 수 있으며 실시간으로 엔진 데이터의 건전성을 모니터링 할 수 있다.

3.2 미들웨어 구조

미들웨어는 크게 엔진 구성품 제어 주기에 따라 구동기제어와 엔진제어 주기로 나뉘며 Application 소프트웨어 또한 두가지 주기의 소프트웨어로 구성된다. 내부 구조는 Fig 3.에서 보이는 바와 같이 구동기제어와 엔진제어로 분리하여 설계했다. 미들웨어의 구동기제어 모듈과 엔진제어 모듈은 각각 FPGA(Field Programmable Gate Array)로부터 별도의 엔진센서 데이터를 받으며, 각각 해당 제어 모듈의 Application 소프트웨어로 데이터를 송수신한다. 각 모듈은 시스템의 주기 요구사항에 따라 설계되었으며, 각 모듈에 할당된 Task들은 SYS/BIOS Task 선점 OS(Operating System) 정책을 통해 원자적으로 동작이 가능하도록 설계했다. 이러한 분리 설계를 통해 구동기제어와 엔진제어에 사용되는 입출력 데이터를 분리하여 원자성을 보장함으로써 소프트웨어의 안전성을 높였다. 이는 DO-178C 가이드라인의 컨트롤 플로우와 데이터 플로우의 권고사항으로 기능에 따라 제어하는 모듈과 그 모듈에 입출력되는 데이터를 완전히 분리하여 안전성을 높여야 한다는 권고사항을 따른다[1]. 또한 Fig 3.에서 보이는 바와 같이 미들웨어는 이중화 채널 통신과 항공기체계 통신 모듈을 가진다. 별도의 통신 모듈을 가짐으로써 비동기 모듈이 주제어 모듈들에게 영향성이 없도록 설계했다. 통신 데이터는 이중 버퍼 구조의 임계영역을 할당하여 제어 모듈의 데이터와 통신의 데이터의 동기화시 안전성을 높였다. 위와 같이 FADEC 운용 특성과 DO-178C 가이드라인에 따라 각 모듈의 분리 설계와 통신 모듈의 이중 버퍼 구조 설계를 통해 미들웨어의 신뢰성과 안전성을 높였다.

3.3 이중화 설계

Fig 4.는 본 논문에서 제안하는 FADEC 이중화 구조를 나타내며, 해당 구조는 하드웨어 구성품, FPGA, DSP(Digital Signal Processor)의 미들웨어 그리고 Application 소프트웨어로 이루어진다. Fig 4.의 Application S/W부터 I/O블록까지 하나의 채널로 구성된다. 각 채널은 Fig 4.의 가운데 화살표와 같이 CCDL(Cross Channel Data Link)을 통해 데이터를 주고 받는다. 각 채널의 입출력 데이터는 FPGA에서 입력받으며, FPGA는 CCDL 통신을 통해 상대 채널의 입출력 데이터를 입력받는다. 따라서 각 채널의 FPGA는 자신의 채널의 입출력 데이터와 이중화 채널의 입출력 데이터를 가진다. FPGA는 이중화 채널의 입출력

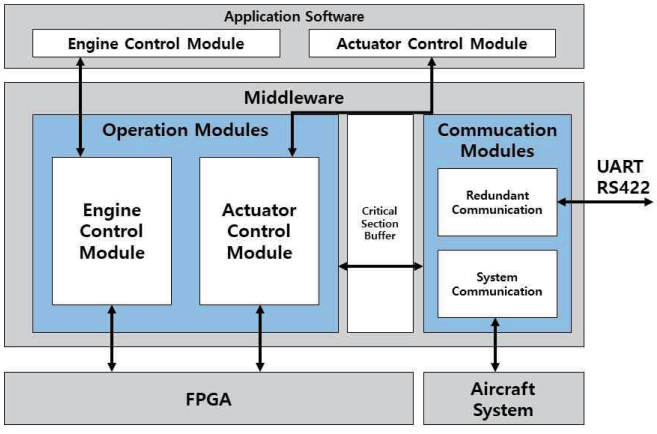

Fig. 3 Middleware Structure

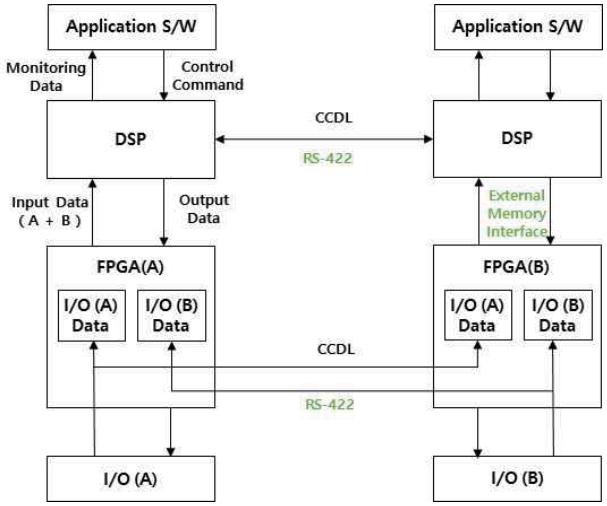

Fig. 4 FADEC Redundant Structure

데이터를 DSP의 외부메모리 인터페이스를 통해 DSP에 전달한다. 이는 Fig 3.의 하단부분의 FPGA와 미들웨어의 화살표와 같다. 미들웨어는 CCDL 통신을 통해 엔진제어 데이터와 채널의 건전성 신호를 송수신하여 채널제어에 필요한 데이터를 주고받는다. 이는 Fig 3.의 우측 통신 모듈과 같으며, 이중화 채널 통신 데이터는 비동기적으로 동작하므로 인터럽트를 통해 수신받으며, 제어 모듈에 영향성을 줄이기 위해 이중 버퍼 구조를 설계하여 데이터 오염을 방지했다. 미들웨어는 Application 소프트웨어로 이중화 채널의 엔진센서 데이터, 고장진단 결과 그리고 통신 데이터를 전달한다. FADEC은 DSP와 FPGA의 각 이중화 채널 통신으로 구성되어 있으며, 이는 2개의 Tx 라인과 2개의 Rx 라인으로 구성되는 것이다. 결과적으로 이중화 채널의 라인 중복으로 볼 수 있으나, FPGA는 이중화 채널의 하드웨어 입출력을 모두 통신하고 DSP는 이중화 채널의 엔진 제어 알고리즘에서 생성되는 데이터를 통신하게 설계하여 각 구성품 간의 종속성을 줄였다.

3.4 데이터 모니터링 및 고장진단

미들웨어는 FPGA로부터 하드웨어에서 제공하는 고장 코드와 센서신호를 전달 받는다. 고장 코드는 센서 혹은 센서측정 칩에서 제공하는 데이터다. 미들웨어는 고장코드와 센서신호의 특성을 판단하여 Open, Short 등 다양한 고장을 분리하여 진단한다. Fig 5.에서 보이는 바와 같이 미들웨어는 구동기제어와 엔진제어 주기마다 FPGA로부터 고장코드와 센서신호를 읽어 원형 큐에 저장한다. Fig 5.의 원형 큐 크기는 엔진 특성에 따라 모니터링 시간을 고려하여 설정한다. 예를 들어 원형 큐의 크기가 200이고 엔진제어 주기 5ms인 경우, 매 엔진제어 주기마다 엔진제어 모듈에서는 원형 큐에 고장코드와 센서신호를 저장한다. 미들웨어는 원형 큐에 저장된 데이터를 통해 하드웨어를 모니터링하게 되는데, 이는 총 1000(200x5)ms의 데이터를 모니터링하는 것이다. 미들웨어는 모니터링 데이터를 바탕으로 엔진센서의 확정적 고장, 순간적 고장을 구별하여 Application 소프트웨어에 전달한다. 확정적 고장과 순간적 고장은 모니터링 데이터를 통해 구분한다. 미들웨어는 모니터링 시간 동안 엔진센서의 고장 값을 누적시키며 각 엔진센서의 특성을 고려하여 설계된 고장 임계값을 고장 누적 값이 넘길 경우 확정적 고장으로 판단한다. 이를 통해 Application 소프트웨어는 순간적인 노이즈에 의한 고장과 확정적 고장을 구분하여 사용할 수 있다. 또한 엔진 제작과 시험 과정에서 엔진센서의 특성을 파악하여 확정적 고장의 기준 값만 변경하면 확정적 고장에 대한 기준을 쉽게 적용할 수 있다는 장점이 있다.

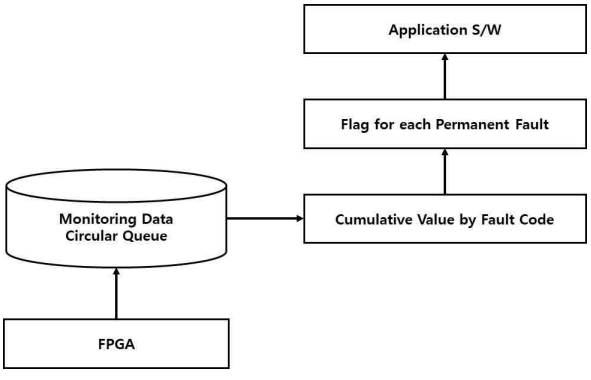

Fig. 5 Fault Diagnosis Design

미들웨어의 고장진단 설계를 이중화 채널 구조로 확장하면 Fig 6.과 같다. 각 채널의 FPGA는 이중화 채널 통신을 통하여 각 채널의 엔진센서 데이터를 송수신 받는다. 미들웨어는 FPGA로부터 양 채널의 엔진센서 데이터를 받으며 미들웨어의 이중화 채널 통신을 통하여 이중화 채널의 건전성을 확인하며 상대 채널의 Application 소프트웨어로부터 엔진제어 명령을 전달받는다. 구해진 종합 데이터 중 고장코드를 통하여 구성된 하드웨어의 건전성을 확인하고 만약 고장일 경우 해당 하드웨어에 종속적인 구성품들은 믿을 수 없기 때문에 모두 결함으로 판단한다. 고장코드를 판단한 후 하드웨어 구성품의 설계 특성에 따라 측정한 데이터를 기반으로 고장진단을 수행한다. 그 후 종합적인 데이터를 Fig 5.의 원형 큐에 기록하여 모니터링한다. 모니터링된 데이터를 바탕으로 순간적인 고장 혹은 확정적 고장을 판단하여 Application 소프트웨어에 전달한다.

4. 결론 및 추후연구

본 논문에서는 신뢰성과 안전성을 고려하여

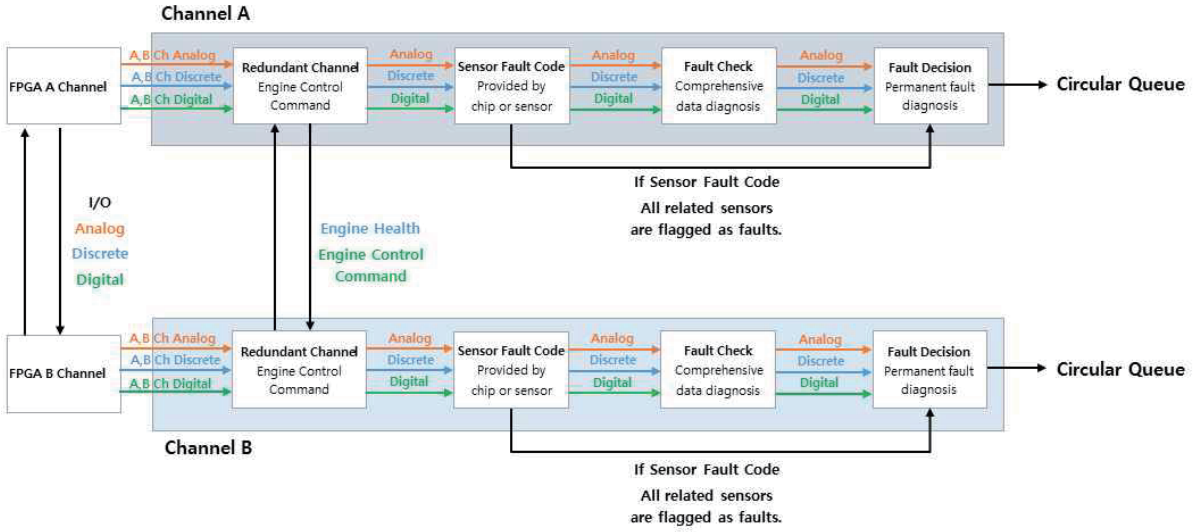

Fig. 6 Redundant Channel Fault Diagnosis

DO-178C 가이드라인에 따른 이중화 구조의 FADEC 미들웨어 아키텍처 설계 방안을 제안한다. DO-178C DAL-A 가이드라인과 안전성 요구사항에 따라 이중화 채널의 구조와 계층구조를 지닌 미들웨어 아키텍처를 설계했다. DO-178C 가이드라인의 컨트롤 플로우 관점에서 SYS/BIOS RTOS를 사용하여 구동기제어와 엔진제어 모듈을 기능적으로 구분했으며, 비동기 통신 모듈은 이중 임계영역 버퍼를 두어 제어모듈에 영향성을 줄였다. DO-178C 가이드라인의 데이터 플로우 관점에서 모듈마다 구분된 데이터 인터페이스를 가지게 설계했다. 이를 통해 모듈별로 데이터를 이원화하여 안전성을 높였다. 또한 하드웨어의 건전성을 모니터링하고 모니터링 데이터를 기반으로 고장을 진단하여 고장 발생 시 엔진제어기가 고장을 판단할 수 있도록 설계하여 안정성을 높였다. 이를 통해 다양한 고장상황에서 FADEC이 자체적으로 대처할 수 있도록 설계했다. 향후에는 DO-178C 가이드라인에 따라 설계된 아키텍처를 소프트웨어로 구현할 필요가 있으며, 구현된 소프트웨어를 DAL-A에 맞추어 신뢰성 시험을 진행할 필요가 있다. 또한 제작된 소프트웨어를 실제 엔진시험에 적용할 예정이며, 시험을 기반으로 설계된 아키텍처의 제한사항을 파악하여 개선 방안을 연구할 예정이다.

후 기

이 논문은 2024년 정부(방위사업청)의 재원으로 국방과학연구소의 지원을 받아 수행된 연구이다. (UC19007GD)

References

[1] RTCA. Inc, “Software Considerations in Airborne System and Equipment Certification,” *RTCA DO-178C*, Dec 2011

[2] C. N. Kumar and S. Kumar, “Full Authority Digital Engine Control (FADEC),” *International Journal of Emerging Trends in Science and Technology*, Vol. 02, Issue. 10, pp. 3298-3302, Mar 2015.

[3] S. Garg, "Aircraft Turbine Engine Control Research at NASA Glenn Research Center," *Glenn Research Center*, NASA/TM 2013-217821, 2013.

[4] A. V. Grekov, S. F. Tyurin, “Fault Tolerant Electronic Engine Controller,” *Proceedings of the 2018 IEEE 9th International Conference on Dependable Systems, Services, and Technologies*, pp. 222-224, May 2018.

[5] F. Mhenni, J. Y. Choley, A. Riviere, N. Nguyen and H. Kadima, “SysML and safety analysis for mechatronic systems,” *Mechatronics-REM*, MECATRONICS.2012.6451042, Nov 2012

[6] X. Fei, C. Bin, L. Rui and H. Shunhua, “A Model-B

ased System Engineering Approach for aviation syste m design by applying SysML modeling," *Chinese Co ntrol and Decision Conference,* CCDC49329.2020.916 4443, Aug 2020

[7] H. Parthasarathi, S. Ramachandra and P. N. Srinivasa murthy, "Model-Based Systems Engineering for Aero Gas Turbine Engine Subsystems," *Asia-Pacific Interna tional Symposium on Aerospace Technology*, pp. 1730 -1744, Dec 2019