https://doi.org/10.7840/kics.2022.47.12.2117

항공기용 인터컴 소프트웨어 실시간 성능 향상을 위한 연구

이 승 목◆, 김 기 형°

A Study to Improve Real-Time Performance of Intercom Software for Aircraft

Seungmok Lee◆, Kihyoung Kim°

요 약

항공기 인터컴 장비는 조종사, 부조종사, 동승자 및 무선통신원 등과 같은 관련 인력 간 음성 통화 기능을 담당하고 있다. 또한, 특정 경고, 주의 사항 조건에서 음성 경고음을 재생함으로써 탑승자들에게 즉각적인 상황인지 기능을 제공한다. 이러한 기능들을 성공적으로 수행하기 위해서 인터컴 장비는 임베디드 시스템 기반의 실시간성 보장이 필수이지만 여러 제약사항으로 인해서 실시간성이 저하 된다.

본 논문은 인터컴 장비의 실시간성 제약사항을 분석하고 연동 데이터 처리 관점과 이벤트 처리 관점이라는 두 가지 관점으로 각각의 실시간성 향상기법을 제안한다. 두 가지 제안 기법에 대해 설계 반영 전과 반영 후를 비교함으로써 정량적으로 검증한다.

Key Words : Avionics Intercom, Real Time, Mission Critical, Safety Critical, Avionics Software

ABSTRACT

The Intercom in aircraft takes the role to provide audio communication among relative personnel such as pilots, copilots, crew and the radio communicators. In addition the Intercom provides the aural warnings and cautions so that crews intuitively identify the situation awareness under particular warning and caution conditions. In order to successfully perform these functions, the Intercom should guarantee real time capabilities based on embedded system but the capabilities are degraded under several constraints.

In this paper, after analyzing factors of realtime constraints enhancement designs are suggested with two viewpoints which are the interface data handling and the event handling respectively. And then the two suggestions are verified quantitatively comparing between before design and after design.

Ⅰ. 서 론

항공기 인터컴 장비는 각종 LRU에서 수신되는 오디오를 조종사에게 또는 동승자에게 송신해 주고, 인터컴과 연동되는 타 장비 간 오디오 신호 및 각종 제어 신호를 송수신하는 장비이며[1] VHF-FM 무전기, U/VHF-AM 무전기와 함께 통신계통을 구성하는 장비[2]이다. 즉 항공기의 조종사, 부조종사 혹은 라디오 등의 통신계통과 음성통신을 제공함으로써 성공적인 임무 수행을 위한 상호 교신이 가능하고 청각을 활용

◆ First Author : Hanwhasystems Avionics System Team, seungmok23.lee@hanwha.com, 정회원
° Corresponding Author : Ajou University Department of Cyber Security, kkim86@ajou.ac.kr, 종신회원
논문번호 : 202209-204-C-RU, Received September 7, 2022; Revised October 19, 2022; Accepted October 23, 2022

한 다양한 경고 신호를 제공하여 위급상황에서 항공기 생존성을 높인다[3].

이러한 운영환경을 위해서 인터컴 소프트웨어는 세 가지 항목에 대해 고려가 되어야 한다. 첫 번째 고려사항은 인터컴 설치환경 특성이다. 이는 항공기 내 설치환경에 따른 장비 무게 절감 및 저전력을 위해 SWaP(Size, Weight and Power)을 고려한, 임베디드 환경 기반 인터컴 소프트웨어가 설계되어야 한다.

두 번째 고려사항은 항공기 운영환경 특성이다. 최근 발생한 보잉 737MAX의 2018년 10월 29일 인도네시아에서 발생한 라이온에어 JT610편 사고(Tjahjono, 2018)와 2019년 3월 10일 에티오피아 항공 ET302편 사고[4]와 같이 항공기 안전 측면과 고속의 비행 임무가 고려되어야 한다. 따라서 항공기 운용환경 특성을 위해서 안전 필수(Safety Critical)와 임무 필수(Mission Critical)를 보장하는 즉시성이 필요하다.

세 번째 고려항목은 항공기 내 인터컴 연동환경이다. 그림 1에서와 같이 다양한 장비와 연결이 되어 있으므로 사용자가 원하는 적기(適期)에 해당 장비와 연동이 되어야 한다. 즉, 세 가지 고려사항을 종합해 볼 때, 항공기의 안전 및 임무 수행과 밀접한 관련이 있어 고신뢰성과 강건성(Robustness)이 요구되므로[5] 인터컴 소프트웨어는 임베디드 시스템에 기반한 즉시성과 적시성의 실시간 특성이 인터컴 소프트웨어에 반영되어야 한다.

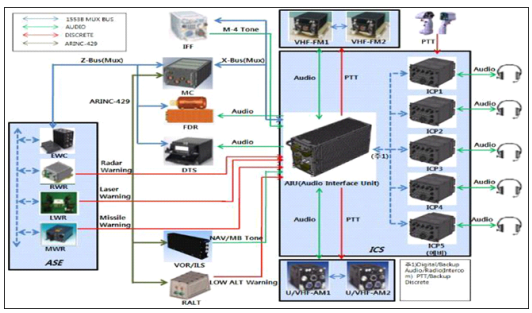


그림 1. KUH 인터컴 항공기 연결 Block Diagram[2]
Fig. 1. KUH Intercom interface Block diagram

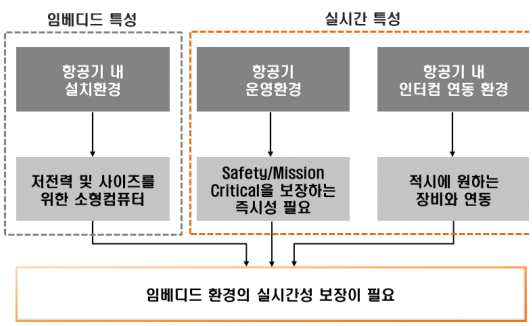


그림 2. 인터컴 소프트웨어 설계 고려사항
Fig 2. The consideration of Intercom SW design

본 논문은 2장에서 관련 연구를 통해 실시간성을 보장하기 위한 대표 기법을 언급하고 관련 연구에 대한 실시간성을 저해하는 요소를 식별한다. 3장에서 임베디드 기반의 인터컴 소프트웨어가 실시간 보장을 위해 제약사항 극복 전략과 극복을 위한 설계안을 제시한다. 4장에서 제시된 설계안에 대해 검증을 통해 효과성을 확인하고 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

임베디드 환경에서 실시간성을 보장하기 위해서 사용되는 대표적인 기법으로는 인터럽트 방식과 운영체제에서 지원하는 선점형 스케줄링이다. 2.1절에는 인터럽트 방식을 2.2절에서는 선점형 스케줄링에 대해서 관련 연구를 진행한다.

2.1인터럽트 방식

인터럽트는 CPU에게 비동기 이벤트를 전달하는 하드웨어 메커니즘이며 인터럽트가 인지될 경우, CPU는 현재 일부 혹은 전체의 CPU 값(레지스터 등)을 저장하고 ISR이라고 일컫는 특정 서브루틴으로 건너뛴다.[6] 운영체제에서 인터럽트는 CPU 내/외부의 긴급한 서비스 요청에 따라 현재 실행 중인 일을 중단하고, 그 요청에 합당한 서비스를 처리하는 기능으로 CPU가 입력 또는 출력을 수행하는 데 중요한 기능[7]이다.

2.1.1 인터럽트 처리 절차

인터럽트 처리 절차는 그림 3과 같은 순서대로 진행된다. 먼저 현재 실행 중인 프로그램이 인터럽트가

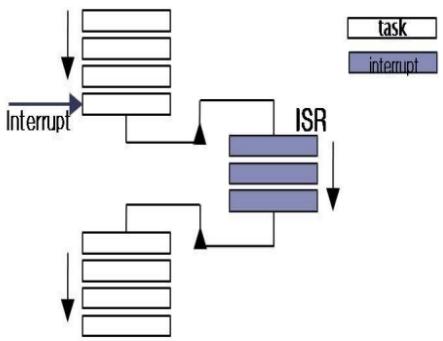


그림 3. 인터럽트 처리 흐름[7]
Fig. 3. Interrupt processing flow

발생하면 현 프로그램은 중단이 되고 보관이 된 다음 인터럽트 처리를 시작한다. 인터럽트 발생에 따른 관련 동작은 인터럽트 서비스 루틴으로 수행되며 서비스 루틴 실행 완료 후 중단된 이전 상태로 복구한 다음, 기존 프로그램을 계속 수행하게 된다.

2.1.2 다중 인터럽트의 처리

실질적으로 인터럽트는 여러 개의 입력처리를 해야 한다면 다중으로 인터럽트를 처리되어야 한다. 그림 4는 ISR인 Interrupt handler X가 수행되더라도 Interrupt handler Y에 대한 인터럽트가 발생하더라도 Interrupt handler X 완료 후 Y가 수행되는 순차적인 인터럽트 처리를 나타내었다. 이는 Interrupt Handler X를 처리할 동안 Interrupt Handler Y가 실행되지 않도록 하는 것인데 Interrupt handler X가 처리 중에 Interrupt handler Y를 고려하지 않으므로 자칫 Interrupt handler Y가 미 수행될 수 있다. 이러한 방식의 단점은 다중 인터럽트의 우선순위나 time critical 한 경우를 고려하지 않는 것이다[8].

그림 5는 Interrupt Handler X 실행 중에 이보다 더 높은 Interrupt Handler Y가 발생함에 따라 X 실행을 멈추고 Y가 실행되는 것을 나타내었다. 즉 연속적인 인터럽트 처리와 달리 인터럽트에도 상대적인 우선순위를 할당해 인터럽트 처리 중 높은 우선순위의 인터럽트를 처리할 수 있도록 하였다.

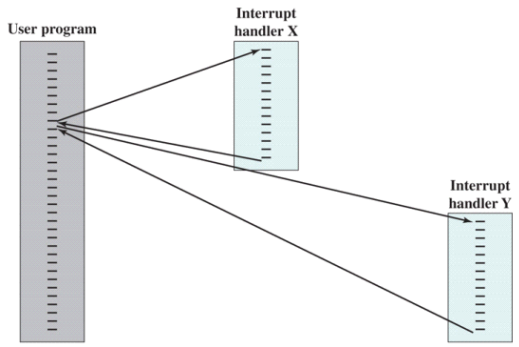


그림 4. 연속인 인터럽트의 처리[8]
Fig. 4. Sequential interrupt processing

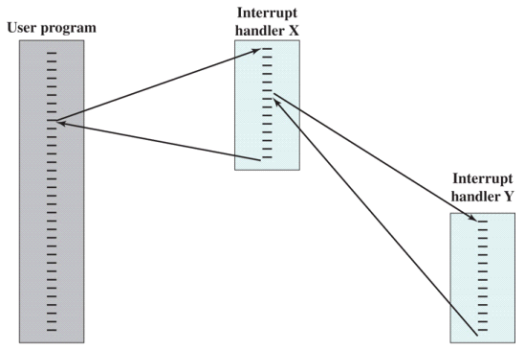


그림 5. 중첩 인터럽트 처리[8]
Fig. 5. Nested interrupt processing

2.2 선점형 스케줄링

스케줄링이란 여러 기능(태스크)을 수행하기 위해서 CPU 자원을 효율적으로 관리하기 위한 메커니즘을 의미한다. 즉, 태스크 수행을 위해 언제, 어떤 태스크에서 프로세서를 할당할 것인지 결정하는 작업[9]이다. 스케줄링은 비선점형 스케줄링과 선점형 스케줄링으로 나뉜다. 비선점형 스케줄링 방식은 어떠한 태스크가 프로세서를 할당받아 수행 중일 때 해당 태스크가 자발적으로 프로세서를 양보하지 않는 한 다른 태스크로의 전환이 발생하지 않는다.[9] 반면 선점형 스케줄링 방식은 태스크마다 우선순위가 동적 혹은 정적으로 할당이 되고, 그 우선순위에 따라서 실행의 순서가 조정이 된다. 즉, 선점형 스케줄링 방식은 태스크가 프로세서를 할당받아 수행 중일지라도 다른 태스크로의 문맥 전환이 발생할 수 있으며, 우선순위가 높은 태스크를 빠르게 처리할 수 있다[9].

그림 6는 비선점형 스케줄 방식을 도식화한 그림으로 인터럽트가 발생하면 해당 ISR은 High Priority 태스크를 준비시킨다. 하지만 현재 우선순위가 낮은 Low Priority 태스크가 동작 중이므로 비선점형 스케줄 방식은 High priority 태스크가 준비상태임에도 실행을 다 완료한 다음에 High Priority 태스크를 실행시킨다. 하지만 그림 7와 같이 선점형 스케줄 방식은 ISR이 High Priority 태스크를 준비시키면 우선순위

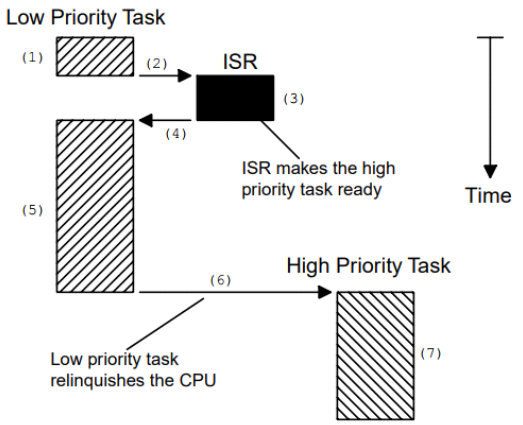


그림 6. 비선점형 스케줄링[6]
Fig. 6. Non-preemptive scheduling

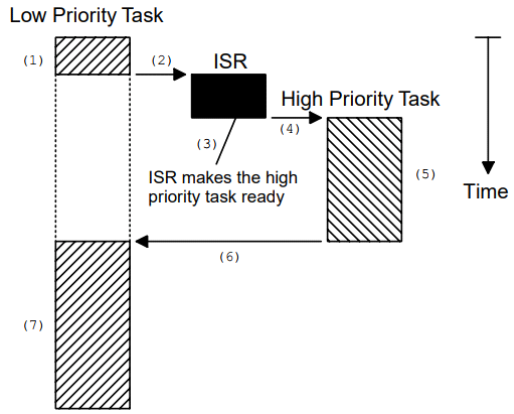


그림 7. 선점형 스케줄링[6]
Fig. 7. Preemptive scheduling

에 맞게 ISR 종료 후 즉시 High Priority 태스크를 실행 및 완료 후 Low Priority 태스크를 실행시킨다.

선점형 방식은 중요도 혹은 빨리 처리가 필요한 태스크에 높은 우선순위를 할당함에 따라 빠르게 처리 가능하므로 실시간 처리에 적절하다.

2.3. 인터럽트와 선점형 스케줄링의 실시간성 제약사항

인터럽트와 선점형 스케줄링은 임베디드 시스템 환경에서 실시간성을 제공하지만 2가지 관점에서 제약사항이 발생한다.

(제약사항 1) 인터럽트와 선점형 스케줄링 고유의 메커니즘 방식에 따른 제약

(제약사항 2) 하드웨어와 운영체제에 의존성이 높은 메커니즘

(제약사항 1)에 대해서 인터럽트와 선점형 스케줄링의 공통점은 중요도가 높거나 빠른 처리가 필요한 태스크에 먼저 프로세서 자원을 제공해서 신속하게 처리해 실시간성을 향상하는 것이다. 하지만 이러한 메커니즘이 오히려 실시간성을 저하시킬 수 있다.

표 1은 중첩 인터럽트로 인해서 기존 태스크(메인 프로그램)의 실행이 지연되는 상황을, 표 2는 높은 우선순위 태스크가 계속 수행됨에 따라 우선순위가 낮은 태스크는 CPU 자원 할당을 받지 못하는 경우를 순차적으로 설명하였다. 이는 각각의 고유 메커니즘의 특성으로 인해서 오히려 실시간성이 저하되는 경우가 발생한다.

(제약사항 2)에 대해서 인터럽트와 우선순위에 기반한 선점형 스케줄링 처리는 각각 하드웨어와 운영체제에 의존적이다. 이는 사용자가 최적화에 제약성을 제공한다.

표 3은 인터럽트와 선점형 스케줄링의 관할 영역을 식별 정리한 표이다. 인터럽트 영역은 외부 I/O와 프로세서(CPU, MCU)뿐만 아니라 하드웨어와 운영체

표 1. 중첩 인터럽트에 의한 실시간성 저하
Table 1. Realtime impeding for nested interrupt

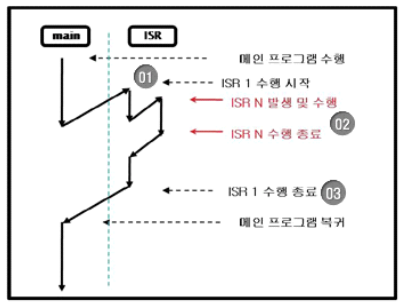
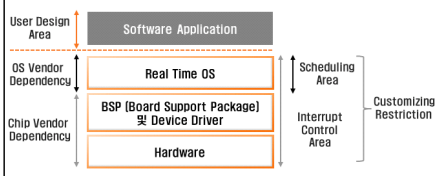
구 분	내 용	
개념도 [9]		
설명	① ISR1 수행	최초 인터럽트로부터 수행되는 ISR1 수행
	② ISR N 처리	인터럽트가 다중 발생 되면 ISR N이 처리되고 ISR1은 Wait
	③ ISR 1 처리지연	ISR N 처리 완료 후 ISR1 처리가 지연

표 2. 선점형 스케줄링에 의한 실시간성 저하
Table 2. Realtime impeding for preemptive scheduling

구 분	내 용	
개념도 [10]		
설명	① 높은 우선순위 처리	선점형 스케줄링의 정책에 따라 높은 우선순위 먼저 처리
	② 낮은 우선순위 Task 대기	반복적인 높은 우선순위 수행으로 낮은 우선순위 Task Wait
	③ 실행 불가 (Starvation)	우선순위가 낮은 Task는 CPU 할당을 못 받는 Starvation 발생

표 3. 하드웨어와 운영체제에 의존성 설명
Table 3. Dependency of HW and OS

구 분	내 용	
적용 부분		
설명	관련 영역	- 인터럽트: 하드웨어, 운영체제, BSP - 우선순위처리: 운영체제
	특징	운영체제, 칩 제조사 관할 영역: 개발자 및 r에 맞도록 커스트마이징(Customizing) 제한

제간 연계를 위해 BSP 혹은 디바이스 드라이버 영역도 포함되며 실시간 운영체제(RTOS)는 인터럽트 이벤트와 소프트웨어 어플리케이션과의 연계를 위해서 적용된다. 선점형 스케줄링은 Kernel 순수 메커니즘이므로 실시간 운영체제(RTOS)에 적용되는 고유의 영역이다. 이는 설계자 혹은 사용자가 소프트웨어 어플리케이션에서 성능 최적화를 위한 Customizing은 제약되므로 한계점을 보유하게 된다.

Ⅲ. 인터컴 소프트웨어 실시간 성능 향상을 위한 설계

3.1 실시간성을 저해하는 요소 분석 및 극복 전략

인터컴은 그림 1과 같이 다양한 장비와의 연동으로 인해서 연동데이터의 잦은 Transient가 발생할 경우 관련 처리를 위한 부하가 발생한다. 일반적으로 전용 입출력 보드와 해당 보드에서 동작하는 펌웨어 및 소프트웨어로써 버퍼링을 활용해 부하 대응이 가능하다. 하지만 인터컴은 SWaP 측면에서 전용 입출력 보드 및 펌웨어/소프트웨어가 제약적이므로 단일 소프트웨어로 잦은 Transient에 대해서 인터럽트 처리를 하거나 선점형 스케줄링을 사용한다면 2.3절의 (제약사항 1)로 인해 관련 기능의 응답성과 응답에 대한 적시성이 감소한다. 이러한 제약사항에 대한 완화를 위해서 인터럽트나 선점형 우선순위 스케줄링을 최적화하는 것은 하드웨어와 실시간 운영체제(RTOS)에 의존성(제약사항 2)에 따라 한계점이 존재한다. 2가지 제약사항을 극복하기 위해 연동 자체에 대한 데이터 처리 관점과 연동에 따른 이벤트 처리 관점, 이 두 가지 관점으로 접근해야 한다.

두 가지의 관점에 대해서 극복을 위한 전략은 인터

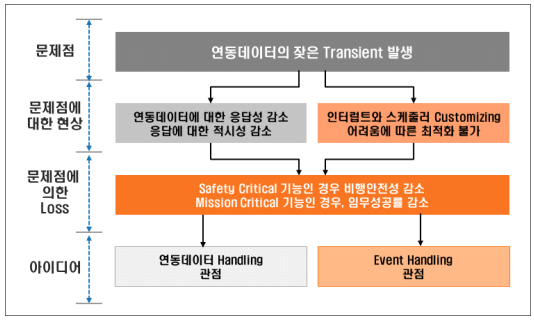


그림 8. 실시간성 저해 요소 분석과 극복을 위한 접근
Fig. 8. Factor analysis of impeding realtime and approach to overcome

컴 탑재 소프트웨어 내 사용자 설계영역 즉, 어플리케이션 영역에서

(접근 전략 #1) 연동데이터 처리 관점의 연동데이터의 정제 전략
(접근 전략 #2) 이벤트 처리 관점의 이벤트 부하 완화 전략

으로 접근할 수 있다. 이는 실시간 성능 향상을 위해 운영체제 스케줄링의 개선과 효율성에 집중된 현 논문들의 주요 연구 방향과 달리 어플리케이션 영역에서 성능 향상을 위한 접근이다.

연동데이터 처리 관점에서 연동데이터의 정제는 그림 9의 Interface Input Handling 영역에서 수행이 필요한데, 이는 Interface Input Handling 영역에서 입력을 직접적으로 처리하는 부분에서 과도한 입력이 수신되었다고 판단할 경우 입력을 정제해서 실효적인 데이터를 전달해야 하기 때문이다.

이벤트 처리 관점에서 이벤트 부하 완화는 그림 9의 Parsing & Processing 영역에서 수행이 필요한데, 이는 입력처리 영역에서 정제된 입력이 수신되더라도 입력데이터를 직접적으로 이용, 처리하면서 실질적으로 무의미한 데이터를 재정제함으로써 출력 및 반응

표 4. 실시간성 제약사항 극복을 위한 전략
Table 4. Strategy to overcome realtime constraints

관점	연동데이터 처리	이벤트 처리
전략	접근 전략 #1 연동데이터의 정제	접근 전략 #2 이벤트 부하 완화
설명	Interface Input Handling 영역의 정제를 통해 잦은 Transient의 경감	연동데이터의 정제에도 불구하고 데이터의 잦은 Transient에 대한 이벤트 부하 완화

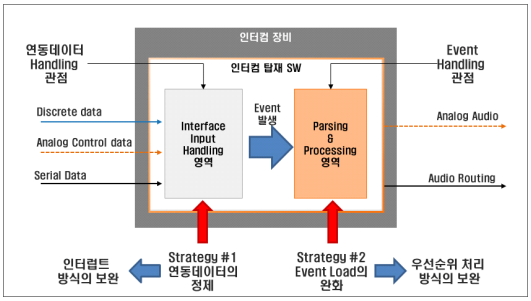


그림 9. 제약사항 극복을 위한 전략 개요
Fig. 9. Strategy overview to overcome constraints

처리 부분의 부하를 낮추도록 한다. 즉, 1차로 Interface input Handling 영역에서 정제하고 2차로 Parsing & Processing 영역에서 정제함으로써 실질적인 인터컴 출력의 반응시간을 최소화한다.

3.2 접근 전략 #1 연동데이터의 정제

인터컴 운용환경에서 입력데이터는 그림 1과 같이 음성, 오디오 등의 Analog 데이터 입력의 디지털 처리를 위해 양자화 수행이 필수적이다. 하지만 Analog 데이터를 온전히 디지털화할 수 없으므로 Analog 데이터가 샘플링 경곗값에 존재할 경우 샘플링 경계 전후 데이터로 다량의 Transient가 발생한다. 그림 10은 양자화 후 샘플링의 경계에 대해서 디지털 데이터가 다량으로 Transient 되는 예시로, Analog data가 양자화된 후 Serial Data로 전환될 때, Analog 영역에서 정상 구간 1과 2 사이의 경계 구간 데이터가 Digital 영역으로 변환 시 정상 구간 1과 2에 해당하는 각각의 디지털 데이터로 Transient가 매우 빈번하게 발생할 수 있으며 각각의 Transient에 대해서 인터럽트로 처리될 경우, 중첩 인터럽트가 다량 발생하여 실시간성이 떨어지게 되고 이는 연동데이터 정제의 대상이 되어야 한다.

이러한 무의미한 다량의 Transient를 정제하기 위해서 Analog 데이터의 변경 추이를 기반으로 샘플링

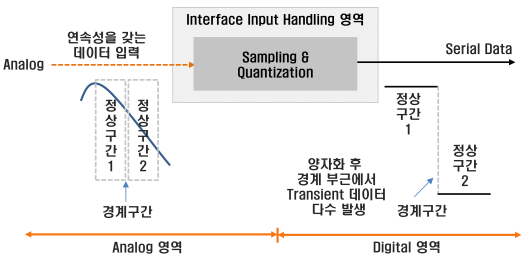


그림 10. 실시간 저하 Case - 다량의 Transient 발생
Fig. 10. Case of impeding realtime - multiple transient

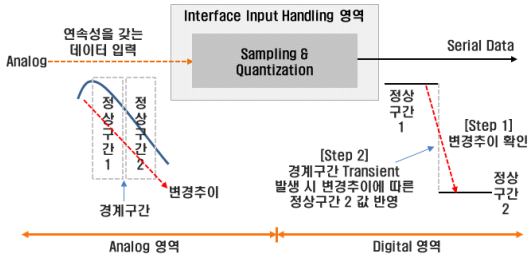


그림 11. 연동데이터 정제를 위한 제안
Fig. 11. Suggestion to refine interface data

경곗값의 Transient 발생 시 가중치를 적용한 후 디지털 데이터를 반영한다. 그림 11는 제안 기법을 도식화한 것으로 정상 구간 1과 정상 구간 2의 변경 추이를 확인한 다음(Step 1), 경계 구간 Transient 발생 시 추이에 따른 정상 구간 2 값을 반영(Step 2)한다.

또한, 데이터 Transient 발생마다 인터럽트 처리가 아닌 주기적인 Poll 방식을 활용함으로써 과도한 인터럽트 처리를 방지한다.

3.3 접근 전략 #2 이벤트 부하의 완화

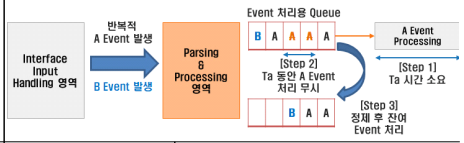
표 5와 같이 인터컴 운용환경에서 a Data와 이에 파생되는 A 이벤트가 b Data와 관련된 이벤트보다 우선순위가 높고 반복적으로 발생하면 선점형 스케줄링으로 B는 CPU 자원고갈(Starvation) 현상이 발생하게 된다.

이러한 이벤트 부하에 대해 다중 이벤트 발생 시,

표 5. 실시간 저하 Case - 낮은 우선순위의 자원고갈
Table 5. Case of Impeding realtime - Starvation for lower priority

구분		내용
문제 현상		
설명	발생 환경	- 외부 Digital 입력 데이터(A 데이터)의 반복적인 Transient 발생 - 내부 처리 우선순위: A 데이터에 대한 이벤트 처리 우선순위가 B 데이터에 대한 이벤트 처리 우선순위 대비 높음
	상세 설명	- 2개의 외부 Digital 입력데이터에 대해서 → A 데이터는 반복적으로 Transient → B 데이터는 단일 Transient - A 데이터 처리가 우선순위에 의해 반복적으로 수행됨에 따라 B 데이터 처리가 지연 발생

표 6. 이벤트 부하 처리를 위한 제안
Table 6. Suggestion to handling event load

구분	내용	
제안 개념		
상세 설명	Step 1 A 이벤트 처리 시간 확인	A 이벤트 처리에 대한 시간 소요 확인(Ta)
	Step 2 Ta 시간의 A 이벤트 확인	Ta 동안 반복 이벤트에 대해서 Queue에서 삭제
	Step 3 A 이벤트 정제	A 이벤트에 대해서 정제 후 잔여 이벤트 처리

A의 처리 시간을 고려해 처리 시간 동안 실행될 수 없는 무의미한 이벤트는 무시하도록 한다. 즉, 표 6과 같이 이벤트 처리를 위해 Queue에 해당 이벤트를 저장하고 A 이벤트 처리를 통해서 실질적으로 의미 없는 A 이벤트를 Queue에서 삭제해서 B 이벤트로 CPU 자원이 분배되도록 한다.

Ⅳ. 검증 및 검증 결과 분석

4.1 검증환경 구축

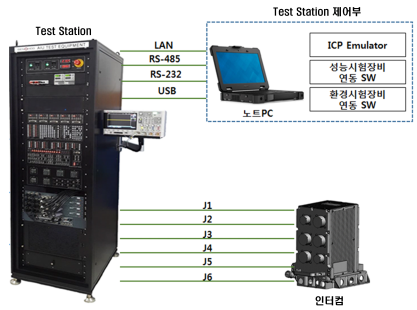
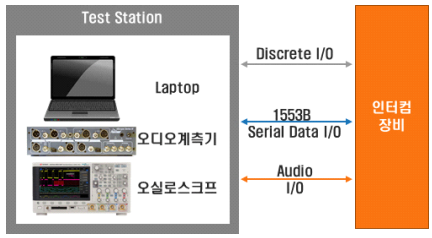
제안 아이디어에 대해 표 7와 같이 적용환경을 구축하고 검증 결과를 확인하였다.

해당 아이디어에 대한 정량적인 검증을 위해서 인터컴이라는 운용환경을 고려할 때 두 가지 항목을 확인할 필요가 있다. 첫째로 인터컴은 연동 장비가 많기 때문에 입출력에 대한 응답(1)이 빨라야 하며 두 번째로 인터컴은 음성 및 오디오 신호를 처리하기 때문에 실시간성이 저하될 경우 오디오 영향성을 확인(2)해야

표 7. 구현 환경
Table 7. The implementation environment

구분	내용	설명
대상 하드웨어	CPU	- Xilinx Zynq-7000 기반 SoC
	RAM	- 128MB
운영체제	사용 운영체제	- Highintegritysystems SafeRTOS v7.10
소프트웨어 개발환경	개발 도구	- Eclipse 기반 Xilinx SDK 2016.4 버전
	개발 언어	- C 언어

표 8. 검증환경 구축
Table 8. Set up the verification environment

구 분	내 용	
검증 환경 형상		
검증 Block 도		
구성별 설명	인터컴 장비	해당 아이디어 탑재 대상 장비
	Test Station	○ 노트 PC - 외부 이벤트 신호인가 - 예: Pull Up/Altitude Low ○ 오디오 계측기 - Audio I/O 계측 ○ 오실로스코프 - 신호 파형 및 시간 측정 ○ Test Station 본체 - 각종 I/O 인터페이스 제공

한다. (1) 에 대해서는 응답시간으로 확인할 수 있으며 인터컴 장비 입력에 대한 출력을 먼저 정의하고 정의된 입력 시간에서 출력 시간까지 시간 차이로 산출한다. (2) 에 대해서 인터컴이 정형파를 생성할 때 실시간성이 저하되면 신호에 대한 왜곡이 발생하므로 왜곡률을 활용해 확인한다. 왜곡률 계산은 실제 사용자 측면에서 잡음 성분까지 고려한 THD+N(Total Harmonic Distortion plus Noise)으로 계산하며 계산식은 아래와 같다.

$$THD + N = \frac{\sqrt{V_2^2 + V_3^2 + V_4^2 + \ldots + V_n^2 + V_{noise}{}^2}}{V_s} \quad (1)^{[11]}$$

Vs = 신호 진폭(RMS volts)
Vn = n번째 하모닉 신호 진폭
Vnoise = 측정 구역대의 잡음 RMS 값

두 가지의 검증항목을 위해 사용되는 연동 신호 및 측정 방법은 아래와 같다.

[검증항목 측정을 위한 연동 신호]

① Analog 인가 : 다량의 Transient인가

② Discrete #1 입력: 이벤트 인가, Active Low

③ Discrete #2 출력

- Discrete #1 입력에 대한 출력
- Analog 인가에 대한 처리 대비 낮은 우선순위
- Discrete #1이 Enable 되면 해당 신호 Enable (Active Low)

④ 헤드셋 오디오 출력

- Discrete #1에 대한 1kHz 사인파 출력
- Discrete #1이 Enable 되면 해당 신호 Enable (Active Low)

[검증항목 측정 방법]

① 응답시간

- 실시간성 저하에 대한 지연 측정
- 실시간 저하 시 응답시간 증가
- 입력기준 출력 응답시간 측정 : 그림 12의 iscrete #1 Enable 인가 시점 기준으로 Discrete #2 Enable이 출력되는 시간 측정

② 왜곡률

- 실시간성 저하에 대한 다른 태스크 영향성 측정
- 실시간성 저하 시 연속적인 Sinewave 모양이 clipping 되거나 비연속적인 모양으로 출력
- 출력데이터의 영향성 확인: Sinewave 출력에 대해서 오디오 계측기로 왜곡률 측정

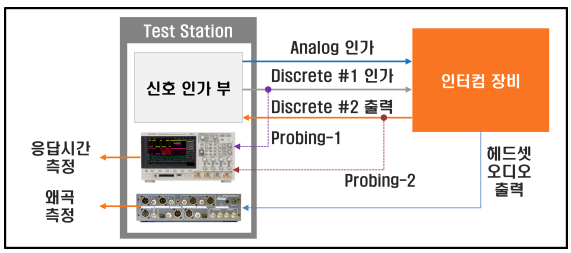


그림 12. 인터컴 기능을 활용한 아이디어 검증
Fig. 12. Verification of suggestion via Intercom function

4.2 검증 결과

검증을 위해 아이디어 적용 전과 적용 후 2가지의 소프트웨어 형상으로 시험하였으며 표 9 와 같은 결과를 획득하였다.

표 9. 시험 결과 요약
Table 9. Summary of test results

구분	로직 적용 전	로직 적용 후
응답시간	110.2 msec	2.15 msec
왜곡률	24.9 %	0.71 %

표 10. 로직 적용 전 응답시간 시험 상세 결과
Table 10. The detailed test result for response time before reflecting suggestion

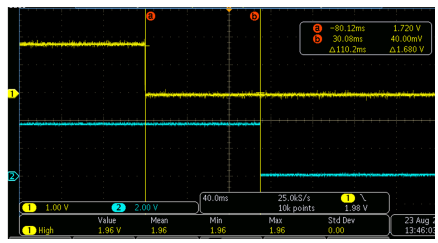
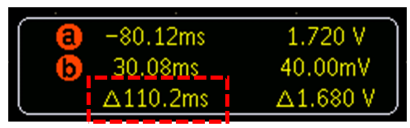
응답 차이	
응답 차이 측정	

표 11. 로직 적용 후 응답시간 시험 상세 결과
Table 11. The detailed test result for response time after reflecting suggestion

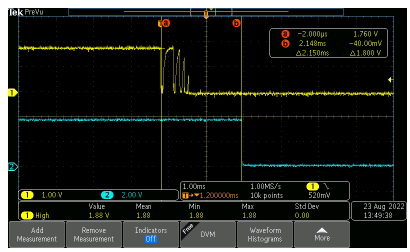
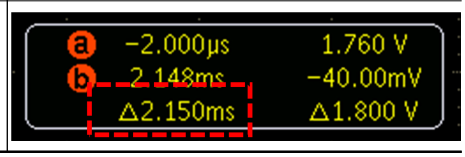
응답 차이	
응답 차이 측정	

표 12. 로직 적용 전 왜곡률 시험 상세 결과
Table 12. The detailed test result for distortion before reflecting suggestion

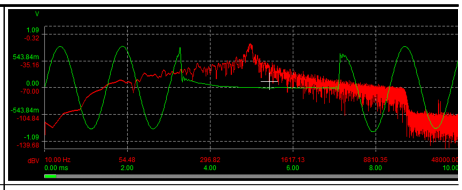
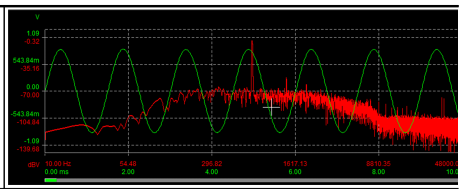
파형 및 주파수 분석	
왜곡률	

표 13. 로직 적용 후 Distortion 시험 상세 결과
Table 13. The detailed test result for distortion after reflecting suggestion

파형	
왜곡률	

V. 결 론

항공기용 인터컴은 다양한 장비와 연동을 통해 조종사의 임무 편의성을 제공해 준다. 따라서 인터컴 장비는 적시성과 즉시성이 굉장히 중요한 요소이다. 적시성과 즉시성을 보장하기 위해 관련 연구를 통해 인터럽트 방식과 선점형 스케줄링 방식을 확인하였으나 이는 하드웨어 및 운영체제에 상당히 의존적이므로 사용자 관점에서 제한성을 확인하였다. 3장에서 이러한 제한사항을 극복하기 위해서 연동데이터 처리 관점과 이벤트 처리 관점에 2가지 안을 제시하였고, 관련 설계하고 구현하였다. 4장에서는 2가지 안의 효과를 정량적인 수치로 검증을 하였고 2가지 안이 효과적임을 확인하였다. 2가지 제안은 항공기뿐만 아니라 실시간성이 요구되는 산업안전 분야 혹은 자동차 분야에도 활용이 가능할 것으로 판단된다.

References

[1] S. Lee, “A study on the audio routing processing for aircraft intercom considering reusability,” *J. Aerospace Syst. Eng.*, vol. 11, no. 6, pp. 1-9, 2017. (https://doi.org/10.20910/JASE.2017.11.6.1)

[2] Y. Kim, J. Chang, B. Jun, et al., “A study on voice communication quality improvement of intercom system for KUH,” *J. Korean Soc. Aeronautical and Space Sci.*, vol. 41, no. 12, pp. 1002-1010, 2013. (http://dx.doi.org/10.5139/JKSAS.2013.41.12.1002)

[3] J. Lee, “A study on voice warning message volume control technique and volume level preference analysis,” M.S. Thesis, Kyungpook National University, Korea, 2017.

[4] J. Moon, et al., “Accident analysis & lessons learned of B737MAX JT610 from a flight control system design perspective,” *J. Korean Soc. Aeronautical Sci. and Flight Oper.*, vol. 28, no. 1, pp. 106-114, 2020. (https://doi.org/10.12985/ksaa.2020.28.1.106)

[5] Y. Kim, et al., “Development of operational flight program for avionic system computer,” *The Korean Soc. Aeronautical and Space Sci.*, vol. 33, no. 9, pp. 104-112, 2005.

[6] J. J. Labrosse, *MicroC OS II: The Real Time Kernel 2/E, Chapter2: Real-Time Concepts*, CMP Books, 2002. (https://file.elecfans.com/web1/M00/7F/DD/pIYBAFwnEZGAcwiCAEGrbQfCqmc945.pdf)

[7] K. Choi, “An efficient technique for interrupt service routine for sensor operating systems,” M.S. Thesis, Soongsil University, Korea, 2010.

[8] W. Stallings, *Computer Organization and Architecture*, 11th Ed., Pearson, 2019.

[9] H. Noh, “Design and implementation of priority level scheduler for solving starvation,” M.S. Thesis, Hanyang University, Korea, 2016.

[10] J. Yoo, B. Kim, B. Choi, et al., “Design and implementation of preemptive EDF scheduling

algorithm in TinyOS," *The KIPS Trans. : Part A*, vol. 18, no. 6, p. 255, Dec. 2011. (http://dx.doi.org/10.3745/KIPSTA.2011.18A.6.255)

[11] W. Jung, *Op Amp Applications Handbook*, ELSEVIER, 2005. (ISBN 0-7506-7844-5)

이 승 목 (Seungmok Lee)

2009년 2월 : 경북대학교 전기전자컴퓨터학부 졸업
2009년~현재 : 한화시스템 근무
<관심분야> 항공전자, 인공지능, 임베디드, 소프트웨어 품질

김 기 형 (Kihyoung Kim)

1996년 8월 : 한국과학기술원 (KIST) 박사
1997년 3월~2005년 2월 : 영남대학교 컴퓨터공학과 부교수
2005년 3월~2015년 8월 : 아주대학교 정보컴퓨터공학과 교수
2015년 9월~현재 : 아주대학교 사이버보안학과 교수
<관심분야> 블록체인, 사물인터넷 보안, 임베디드 소프트웨어