RTEMS 운영체제를 위한 개발 자동화 프레임워크 설계 및 구현

(Design and Implementation of a Development Automation Framework for RTEMS)

강민우*, 박준용**, 김형호***, 신지우*, 장준혁****, 정진만*****

(Minwoo Kang, Junyong Park, Hyeongho Kim,
Jiwoo Shin, Joonhyouk Jang, Jinman Jung)

요 약

항공우주 시스템에서 실시간 운영체제(RTOS) 기반 소프트웨어의 신뢰성 확보의 중요성이 커지고 있다. 대표적인 항공우주용 운영체제인 RTEMS는 신뢰성 확보를 위해 편리한 사전 인증을 위한 다양한 개발 및 패키지 도구를 제공하고 있지만, 개발 환경 구축과 성능 인증 과정에서 많은 복잡성이 존재한다. 본 논문은 이러한 개발 및 인증 과정의 복잡성을 해결하기 위해 RTEMS 운영체제를 위한 개발 자동화 프레임워크를 설계 및 구현한다. 제안하는 프레임워크는 Docker 컨테이너 기술을 기반으로 표준화된 개발 환경을 제공하고, IDE와 통합된 보조 도구를 통해 사용자의 입력에 따라 복잡한 시스템 구성 파일을 자동으로 생성한다. 또한 GUI 기반의 자동화 도구를 제공하여 인증 과정의 실행, 검증, 결과 분석 과정을 자동화하는 기법을 제안한다.

■ 중심어 : RTOS ; RTEMS ; 개발 자동화 프레임워크 ; 컨테이너화

Abstract

The reliability of real-time operating system (RTOS)-based software is becoming increasingly critical in aerospace systems. RTEMS, a representative aerospace operating system, offers various development and packaging tools to support pre-qualification efforts. However, its ecosystem presents significant challenges regarding the complexity of development environment setup and qualification. To address these complexity in the development and qualification lifecycle, this paper designs and implements a development automation framework specifically for the RTEMS operating system. The proposed framework provides a convenient development environment based on Docker container technology. It features an assistant tool integrated into the IDE that automatically generates complex system configuration files based on user input. Furthermore, this paper proposes a GUI-based automation tool to streamline the execution, verification, and result analysis stages of the qualification process.

■ keywords : RTOS ; RTEMS ; Development Automation Framework ; Software Containerization

Ⅰ. 서 론

현대 항공우주 시스템은 기존의 기계 중심 설계에서, 정교하고 복잡한 소프트웨어에 의해 핵심 기능이 제어되는 소프트웨어 정의 시스템(Software-Defined System)으로 진화하고 있다. 자율 비행, 인공위성 운용 등의 작업에서 소프트

* 정회원, 인하대학교 전기컴퓨터공학과
** 준회원, 인하대학교 전기컴퓨터공학과
*** 준회원, 인하대학교 컴퓨터공학과
**** 종신회원, 한남대학교 컴퓨터공학과
***** 종신회원, 인하대학교 컴퓨터공학과
이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.RS-2023-00252501).
본 연구는 2025년 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음 (2022-0-01057)
접수일자 : 2025년 07월 31일 게재확정일 : 2025년 08월 25일
교신저자 : 정진만 e-mail : jmjung@inha.ac.kr

웨어의 역할이 확장됨에 따라 시스템의 복잡성 또한 증가하였으며, 이는 시스템 전체의 복잡성을 크게 증가시켰다. 이는 단일 소프트웨어의 결함이 전체 시스템의 실패로 이어질 수 있는 새로운 위험 요소가 발생할 수 있으며, 이로 인해 시스템의 신뢰성 확보가 최우선 과제로 떠오르게 되었다.

최근 RTEMS(Real-Time Executive for Multiprocessor Systems) 운영체제는 항공우주와 같이 높은 신뢰성과 결정론적 실시간 성능이 요구되는 임베디드 시스템 분야에서 많이 사용되고 있다. 특히 미 항공우주국(NASA)과 유럽우주국(ESA)의 주요 우주 임무에 성공적으로 채택되어 우수성을 전 세계적으로 입증받았다.

RTEMS가 항공우주 분야에서 높은 성능과 신뢰성을 제공하기 위해 개발 과정에서 고려해야 할 사항은 아래와 같다.

첫 번째로, RTEMS의 개발 및 테스트 환경 구축에는 복잡한 설정 절차와 다양한 수작업 요소를 고려해야 한다. 개발자는 특정 타겟 보드(예: GR712RC, GR740)에 맞는 크로스 컴파일 툴체인을 직접 설정하고, 보드 지원 패키지(BSP)를 구성하며, 시뮬레이터(SIS)를 연동하는 등 모든 과정을 수작업으로 진행해야 한다. 디버깅 및 테스트 또한 CLI 기반의 수동적인 방식으로 이루어져 테스트 시나리오 작성부터 결과 확인까지 개발자가 직접 진행해야 한다. 이는 개발 생산성을 저해하고 잠재적인 오류 발생 가능성을 높이는 주요 원인이 된다[1].

두 번째로, RTEMS가 주로 사용되는 우주, 항공 시스템은 DO-178C(항공 시스템 및 장비 인증에 대한 소프트웨어 고려사항)나, ECSS(European Cooperation for Space Standardization)의 표준 규약 등 엄격한 국제 안전 표준을 반드시 준수해야 한다. 이러한 표준은 요구사항 명세서, 설계 문서, 테스트 계획서, 검증 결과서 등 방대한 공식 산출물을 요구하며, 애플리케이션만이 아닌 운영체제 커널 수준까지의 검증을 수행해야 한다. 개별 개발팀이 RTOS 전체를 대상으로 코드 커버리지 측정, 정적 분석 등을 포함한 모든 인증 요구사항을 충족하는 것은 많은 시간과 비용이 소모된다.

이러한 개발 복잡성과 인증의 복잡성을 해결하기 위해 본 연구는 RTEMS 운영체제를 위한 개발 자동화 프레임워크를 설계하고 구현하고자 한다. 제안하는 프레임워크는 Docker 컨테이너 기술을 이용해 RTEMS 커널과 BSP, 개발 도구를 포함한 실행 환경을 사전에 구축하고, 개발자는 컨테이너 환경에 원격으로 접속하여 로컬 PC의 설정과 무관하게 일관된 개발과 디버깅 작업을 수행할 수 있다. 또한 인증 과정의 실행, 검증, 결과 분석 과정을 자동화할 수 있도록 지원하여 인증 과정의 복잡성 해결에 기여한다.

논문의 구성은 다음과 같다. 2장은 SIS, RTEMS의 QDP와 최신 개발 프레임워크의 연구 동향에 대해 살펴본다. 3장에서 이를 기반으로 RTEMS를 위한 개발 자동화 프레임워크를 설계하고 제안한다. 4장에서는 제안한 프레임워크를 기반으로 실제 구현 결과를 제시하고 토의한다. 마지막 5장에서 전체 연구를 요약하고 향후 연구 방향을 제시하며 결론을 내린다.

Ⅱ. 관련 연구

본 장에서는 제안하는 개발 자동화 프레임워크의 이론적, 기술적 기반을 마련하기 위해 RTEMS의 핵심 구성 요소와, 개발 프레임워크의 최신 연구 동향에 대한 분석을 수행한다. 이를 통해 기존 기술의 현황과 한계를 분석하고, 본 논문에서 제안하는 기법을 구체화한다.

1. SIS (Simple Instruction Simulator)

RTEMS 개발 과정에서 실제 타겟 하드웨어 없이 애플리케이션의 기능적 동작을 확인하고 디버깅할 수 있도록 지원하는 핵심 에뮬레이터 환경으로 SIS가 제공된다. SIS는 SPARC 아키텍

처를 중심으로 하는 오픈소스 명령어 집합 에뮬레이터로서 크로스 컴파일된 RTEMS 바이너리를 호스트 PC 환경에서 직접 실행할 수 있도록 지원한다[2].

GDB 원격 프로토콜을 지원하여 소스 코드 레벨의 디버깅이 가능하며, RTEMS의 테스트 프레임워크와 연동되어 많은 수의 내장 테스트 스위트를 하드웨어 없이 자동으로 실행할 수 있다. 이를 통해 개발자는 코드 변경에 따른 기능적 오류를 신속하게 검출하고 수정할 수 있다.

그러나 SIS는 안전 필수 시스템에 요구되는 엄격한 검증 및 확인(V&V) 표준을 충족시키기에는 실제 하드웨어와의 충실도 격차(Fidelity Gap)가 존재하는 한계가 존재한다. SIS는 시뮬레이터이기 때문에, 복잡한 메모리 컨트롤러의 동작이나 캐시의 영향, 주변장치의 인터럽트 타이밍과 같은 실제 하드웨어의 동적 특성을 정확하게 재현하기 어렵다. 또한 소프트웨어 기반의 에뮬레이터 특성상 실제 하드웨어보다 실행 속도가 느리기 때문에 실시간 성능 분석에 부적합하다는 단점이 존재한다.

결론적으로 SIS는 개발 초기 단계에서 기능적 정확성과 코드의 논리적 오류를 신속하게 검출하는 데 효과적이나, 하드웨어 상호작용의 정확성과 타이밍 결정성이 중요한 실시간 시스템에서의 최종 검증 수단이 되기는 어렵다.

2. RTEMS QDP (Qualification Data Package)

유럽우주국(ESA) 주도로 개발된 사전 인증 툴킷인 RTEMS QDP는 RTEMS 기반 시스템이 ECSS와 같은 엄격한 항공우주 표준을 준수하는 데 따르는 부담을 경감시키기 위해 설계되었다. 사전 인증이란 RTEMS 커널이 엄격한 개발 프로세스에 따라 개발되었음을 입증하는 포괄적 증거 자료 묶음을 제공하는 것으로, 이를 통해 사용자는 RTEMS 커널 위에 추가되는 자신들의 어플리케이션 코드에만 집중할 수 있다.

QDP는 ECSS 표준 문서 세트, 정형화된 요구사항 기준선, 정형화된 요구사항, 자동화 테스트 스위트, 추적성 관리 도구, 인증된 툴체인 및 BSP로 구성되어 인증과정을 효과적으로 지원하나[3], 실제 환경에서 QDP 적용의 여러 가지 어려움이 존재한다.

첫 번째로. 사용자 프로젝트에 QDP를 적용하기에 많은 복잡성이 존재한다. 사용자가 필요로 하는 드라이버를 추가하거나 특정 하드웨어에 맞게 수정하는 과정이 복잡하여 높은 전문성이 필요하다. 두 번째로, QDP가 제공한 모든 테스트를 자신의 하드웨어에서 재실행하고 그 결과가 QDP의 검증 보고서와 일치함을 입증하여야 사전인증의 효력이 발생하는 불편함이 존재한다. 마지막으로 QDP는 특정 시점의 RTEMS 버전을 기반으로 하므로, 장기적인 유지보수가 어려울 수 있다.

따라서 RTEMS의 QDP는 사용자 프로젝트의 항공우주 표준 준수에 도움을 주나 실제 환경에서의 적용 어려움이 존재하기 때문에, 본 논문은 QDP의 복잡한 실행 워크플로우를 자동화하는 프레임워크 구축 방법을 제시하여 사용자의 RTEMS 어플리케이션 구성에 편의성을 제공하는 목표를 가진다.

3. 개발 자동화 프레임워크 연구 동향

RTOS 분야에서도 정적 분석, 테스트, 빌드 과정을 자동화하려는 시도가 증가하고 있다[4]. 특히 정적 코드 분석 도구를 소프트웨어 자동 빌드 파이프라인에 통합하거나 자동화된 파이프라인 생성 기법이 실제 프로젝트에 적용된 사례들이 존재한다[5]. 또한 임베디드 소프트웨어의 효율적인 개발을 위한 프레임워크에 대한 연구가 수행되었으며[6], 가상화 기술을 통해 개발 플랫폼을 구현하는 연구도 진행되었다[7].

그러나 RTEMS와 같은 항공우주용 운영체제에서는 인증과정을 포함한 전체 개발 흐름을 자동화하는 프레임워크의 연구는 미비하다. 기존 자동화 연구는 주로 코드 검증이나 배포 자동화

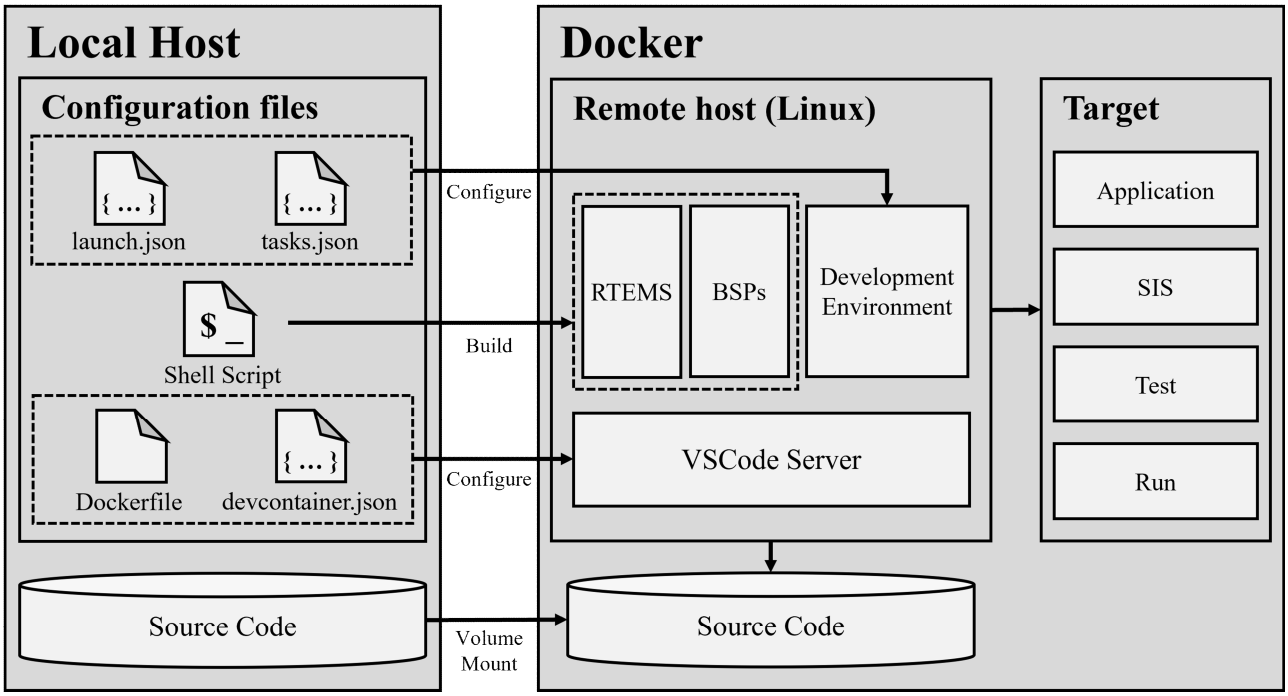

그림 1. RTEMS 개발 자동화 프레임워크 아키텍처

에 집중되어 있으며, 인증에 요구되는 산출물 자동 생성이나 검증 로그 분석은 대부분 수작업에 의존한다.

이러한 한계를 극복하기 위해 모델 기반 시스템 설계(Model-Based Systems Engineering)를 도입하여 설계-검증 연계 및 추적성 확보를 시도하는 연구가 존재하며[8], RTEMS의 커널 수준에서 최악 실행 시간 분석을 통해 인증 신뢰도를 높이려는 연구도 진행되었다[9].

RTEMS의 멀티코어 활용 사례 분석에서는 개발 및 인증의 복잡성과 비효율성이 지적되었으며[10], 인증 대상 산출물의 일관성과 반복 가능성을 확보하기 위해 엣지 실시간 시스템을 위한 자동 코드 생성 연구도 존재한다[11]. 그러나 RTEMS 운영체제에서 인증 과정까지 포괄하는 자동화는 미비하며, GUI 기반 도구나 통합된 개발, 인증 워크플로우에 대한 실질적 구현 사례는 찾기 어렵다.

본 논문은 이를 해결하기 위해 RTEMS QDP 기반의 인증 과정을 포함하는 개발 자동화 프레임워크를 제안한다.

Ⅲ. RTEMS를 위한 개발 자동화 프레임워크 설계

본 장에서는 앞서 2장에서 설명된 RTEMS 개발 환경 구축의 복잡성과 QDP(Qualification Data Package) 적용의 비효율성 해결을 위해 본 논문에서 제안하는 개발 자동화 프레임워크의 전체 아키텍처와 핵심 구성 요소 설계를 상세하게 설명한다. 제안하는 프레임워크는 개발환경 표준화와 워크플로우 자동화라는 목표를 달성하도록 설계되었다.

1. 프레임워크 개요

전통적인 RTEMS 개발 및 인증은 복잡한 과정으로 인해 개발자 역량에 의존하여 비일관성과 비효율성, 오류 가능성이 존재할 수 있다.

제안하는 프레임워크는 논리적으로 두 개의 주요 계층으로 구분된다. 첫 번째는 통합 개발 환경 계층으로, 개발에 필요한 모든 도구와 의존성을 격리하고 표준화한다. 두 번째는 자동화 지원 계층으로, 개발 환경 위에서 복잡한 개발 및 인증 워크플로우를 추상화하고 자동화한다. 이러한 방식은 개발 환경 구축에서부터 최종 인증 산

출물 생성까지의 모든 과정을 연결하고 자동화하여 개발 생산성과 결과물의 신뢰도 향상을 목표로 한다.

2. 도커 환경 기반 통합 개발환경 지원

3.1절에서 제시한 통합 개발 환경을 구현하기 위해 본 프레임워크는 VSCode의 Dev Container 기능을 중심으로 아키텍처를 설계하였다. 이는 개발에 필요한 모든 요소를 코드로 정의하고 관리하여 모든 호스트 시스템에서 동일한 개발 환경을 재현할 수 있도록 보장한다.

그림 1은 RTEMS 개발 자동화 프레임워크의 구체적인 아키텍처를 나타낸다. 아키텍처는 사용자의 Local Host와 원격 실행 환경인 Docker 환경으로 구성된다. Local Host는 VSCode의 Dev Container 기능을 활용하여 Configuration files('devcontainer.json' 등)를 통한 전체 자동화 워크플로우를 제어하는 핵심적인 역할을 수행한다. Docker 컨테이너는 RTEMS, BSPs, VSCode Server 등을 포함하는 통합 개발 환경 계층의 역할을 담당한다.

이러한 아키텍처를 구현하기 위한 핵심 설계 요소는 다음과 같다. 환경의 기반이 되는 컨테이너 이미지는 Dockerfile을 통해 설계되며, 기반 OS인 Ubuntu 22.04에서 RTEMS 툴체인 및 QDP 도구 빌드에 필수적인 패키지(C/C++ 빌드 도구, GDB, Python, LLVM 등)를 선언적으로 설치하여 버전 충돌이나 라이브러리 누락과 같은 환경 설정 문제를 해결한다.

Dockerfile을 기반으로 실제 개발 환경을 구성하고 자동화하는 devcontainer.json은 VSCode가 컨테이너를 개발 환경으로 인식하고 통합하는 역할을 수행한다. 지정된 Dockerfile을 사용하여 컨테이너를 빌드 및 실행하고, Local Host의 Source Code 디렉토리를 컨테이너 내부로 마운트하여 실시간 동기화가 가능하다.

또한 CLI 기반의 복잡한 디버깅 절차를 자동화하기 위해 tasks.json과 launch.json을 연동한다. 디버깅 세션에서, tasks.json이 SIS를 GDB 원격 연결 대기 모드로 실행하고, launch.json은 사용자가 실행할 바이너리만 선택하면 시뮬레이션 준비 상태를 감지하여 GDB를 자동으로 연결한다. 따라서 개발자는 복잡한 개발 환경 구축 없이 GUI 환경에서 디버깅 활동을 수행할 수 있다.

3. 인증(Qualification) 자동화 지원

QDP의 복잡한 설정 및 실행 과정을 추상화하고 자동화하기 위해 VSCode의 확장 프로그램인 'RTEMS QDP Builder'를 설계하였다. 이 도구는 QDP 툴체인 사이의 인터페이스 역할을 수행하며, 인적 오류 최소화 및 작업 효율 증가를 목표로 한다.

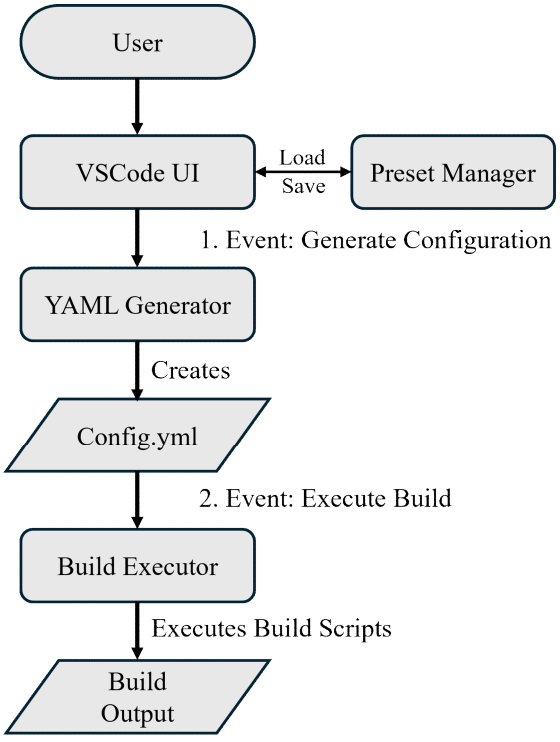

그림 2. RTEMS QDP Builder 워크플로우

그림 2는 'RTEMS QDP Builder'의 내부 작동 원리를 설명한다. 사용자가 VSCode 웹뷰(Webview)로 구현된 UI를 통해 입력값을 작성하면, 이는 Preset Manager와 YAML Generator를 거쳐 QDP가 요구하는 형식(.yml)의 설정 파일로 변환된다. 최종적으로 사용자가 '빌드 실행' 버튼을 클릭하면 빌드 실행 엔진이 이 설정 파일

(a) (b) (c) (d)

그림 3. RTEMS QDP Builder의 주요 기능별 UI

을 기반으로 실제 빌드 명령어를 생성하여 컨테이너에서 실행되는 워크플로우로 설계되었다.

이러한 워크플로우의 상세 동작 원리는 다음과 같다. 먼저 QDP의 핵심 설정 파일인 YAML의 복잡성을 해결하기 위해 VSCode의 웹뷰 기술을 활용한 UI를 설계하였다. 사용자는 다양한 UI 컴포넌트(입력창, 드롭다운, 체크박스)를 통해 타겟 아키텍처, BSP와 같은 환경 정보를 선택할 수 있고, 실제 운영체제 바이너리를 생성하는 Core Build나 인증에 필요한 각종 검증 문서를 생성하는 Documentation 항목 등 40여개의 빌드 구성을 선택할 수 있다.

또한 반복적인 빌드 설정을 간소화하고 재사용성을 높이기 위해 프리셋 관리 시스템을 설계하였다. 이를 통해 사용자가 현재의 설정을 자신만의 프리셋으로 저장, 로드, 삭제할 수 있는 기능을 제공하여, 사용자의 맞춤형 빌드 구성을 쉽게 재사용이 가능하다.

마지막으로 UI에서 빌드 실행 버튼을 클릭하면 설정 파일 생성부터 빌드 명령어 실행, 결과 로깅까지의 모든 과정이 자동으로 처리되는 빌드 실행 엔진을 설계하였다. 이는 Node.js의 child process 모듈을 활용하여 컨테이너 내부에서 빌드에 필요한 qdp_config.py와 qdp_build.py 스크립트를 순차적으로 호출한다. 빌드 과정의 모든 표준 출력과 에러는 VSCode의 출력 패널에 실시간으로 출력되어, 사용자가 진행 상황을 확인하고 에러 상황에 즉각 대응할 수 있다.

Ⅳ. 구현 결과

본 장에서는 3장에서 설계한 개발 자동화 프레임워크를 실제로 구현한 결과물을 제시하며 그 효과를 분석한다. 설계에 따라 Dockerfile, devcontainer.json 등 관련 설정 파일을 작성하여 통합 개발 환경을 구축하였고, 'RTEMS QDP Builder' VSCode 확장 프로그램을 Typescript 및 웹뷰 기술을 이용하여 개발하였다.

그림 3은 'RTEMS QDP Builder' 확장 프로그램의 주요 사용자 인터페이스를 보여준다. 이는 VSCode 내부에 통합되어 세로 스크롤 형태의 긴 UI를 가지므로, 가독성을 위해 핵심 기능을 네 부분으로 나누어 제시하였다. (a)는 사용자의 타겟 아키텍처, BSP, SMP 모드를 설정하는 환경 체크 영역이며, (b)는 빌드 디렉토리 지정과 같은 기본 설정 및 빌드 구성을 저장하고 불러오는 프리셋 관리 기능을 보여준다. (c)와 (d)는 40여 개에 달하는 개별 빌드 스텝을 선택하는 체크

표 1. 전통적 워크플로우와 제안 프레임워크의 비교 분석

평가 항목	전통적 수동 워크플로우	제안 프레임워크	개선 효과
환경 구축	수십 개의 명령어 수동 입력	단일 명령 실행	환경 구축의 복잡성 제거
빌드 재현성	낮음 (호스트 환경에 따라 결과 상이)	높음 (Docker 이미지 기반)	빌드의 신뢰성 및 일관성 확보
QDP 설정	YAML 파일의 문법 및 구성과, QDP 구조에 대한 높은 이해 필요	GUI를 통한 직관적 설정	개발자의 진입 장벽 완화
빌드 실행	다단계 명령어 수동 실행	빌드 자동 실행	오류 가능성 감소

박스 목록의 일부를 보여주며, (d)의 하단에는 구성된 설정을 바탕으로 YAML 파일 생성 및 최종 빌드 실행 버튼이 존재한다. 사용자는 해당 확장 프로그램을 통해 모든 QDP 관련 작업을 직관적으로 수행할 수 있다.

표 1은 RTEMS QDP 빌드의 전통적 워크플로우와 제안하는 프레임워크를 비교한다. 제안하는 프레임워크는 RTEMS 개발 및 인증 과정의 생산성과 신뢰성을 크게 향상시키는 것으로 나타났다. Docker를 통해 환경 구축의 복잡성을 제거하고 빌드의 재현성을 보장하였으며, VSCode 확장 프로그램을 통해 복잡한 QDP의 설정 및 실행 과정을 GUI 기반으로 추상화하였다. 이로 인해 개발자는 운영체제의 내부 구조나 인증 절차의 세부 사항을 모두 알지 못하더라도 핵심적인 개발 업무에 집중할 수 있게 되었다. 따라서 제안하는 프레임워크는 개발자의 인지적 부담을 줄이고 인적 오류를 최소화하여 소프트웨어 개발 안정성을 높이는 데 기여한다.

V. 결론

전통적인 RTEMS 개발 환경은 복잡한 구축 과정과 인증 절차로 인해 개발 생산성을 저해하는 문제점이 존재하였으며, 이를 해결하기 위해 본 논문에서는 RTEMS 개발 자동화 프레임워크를 설계하고 구현하였다. 제안하는 프레임워크는 Docker를 통해 개발 환경을 표준화하고, VSCode 확장 프로그램으로 QDP의 복잡한 설정과 실행 과정을 GUI 기반으로 자동화하였다.

구현된 프레임워크는 Docker 기반의 일관된 인터페이스를 통해 환경 구축과 빌드의 재현성을 보장하였으며, QDP의 내부 구조에 대한 깊은 이해 없이도 인증 관련 작업을 직관적으로 수행할 수 있게 하여 인적 오류 가능성의 최소화 및 개발 생산성 및 안정성 향상의 의의를 가진다.

REFERENCES

[1] 정문재, “소프트웨어 개발 시 프로젝트의 크기가 구현에 미치는 영향,” *스마트미디어저널,* 제1권, 제4호, 79-83쪽, 2012년

[2] RTEMS Documentation Project, *RTEMS SPARC Applications Supplement,* OAR Corporation, Tech. Rep., 2001. https://docs.rtems.org/releases/4.5.1-pre3/rtemsdoc/pdf/sparc.pdf (accessed Aug., 19, 2025).

[3] RTEMS Project, *RTEMS Software Engineering,* Tech. Rep., 2025. https://docs.rtems.org/docs/main/eng.pdf (accessed Aug., 19, 2025).

[4] P. Katapara and A. Sharma, “Embedded DevOps: A Survey on the Application of DevOps Practices in Embedded Software and Firmware Development,” *arXiv preprint arXiv:2507.00421,* 2025.

[5] Z. Wadhams, A.M. Reinhold, and C. Izurieta, “Automating Static Code Analysis Through CI/CD Pipeline Integration,” *Proc. of 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering-Companion (SANER-C),* pp. 119 - 125, 2024.

[6] 강기욱, 이정환, 홍지만, “오픈소스 하드웨어에서 효율적인 임베디드 소프트웨어 개발을 위한 프레임워크,” *스마트미디어저널,* 제5권, 제4호, 49-56쪽, 2016년

[7] 김정녀, “안전한 스마트 단말을 위한 가상화 기반 도메인 분리 보안 플랫폼 구현,” *스마트미디어저널,* 제5권, 제4호, 116-123쪽, 2016년

[8] M.W. Anwar, S. Qamar, F. Azam, W.H. Butt, and M. Rashid, “Bridging the Gap between Design and Verification of Embedded Systems in Model Based System Engineering: A Meta-model for Modeling Universal Verification

Methodology (UVM) Test Benches," *Proc. of the 12th International Conference on Computer Modeling and Simulation,* pp. 82 - 87, 2020.
[9] A. Colin and I. Puaut, "Worst-Case Execution Time Analysis of the RTEMS Real-Time Operating System," *Proc. of the 13th Euromicro Conference on Real-Time Systems,* pp. 191 - 198, 2001.
[10] G. Bloom, J. Sherrill, T. Hu, and I.C. Bertolotti, *Real-Time Systems Development with RTEMS and Multicore Processors,* Taylor & Francis, pp. 1 - 534, 2020.
[11] M. Becker and D. Casini, "The MATERIAL Framework: Modeling and Automatic Code Generation of Edge Real-Time Applications under the QNX RTOS," *Journal of Systems Architecture,* vol. 154, pp. 1 - 13, 2024.

저 자 소 개

강민우(정회원)
2024년 인하대학교 컴퓨터공학과 학사 졸업.
2024년~현재 인하대학교 전기컴퓨터공학과 석사과정 재학.

<주관심분야 : 지능형 임베디드 소프트웨어, 고신뢰성 컴퓨팅>

박준용(준회원)
2025년 인하대학교 컴퓨터공학과 학사 졸업.
2025년~현재 인하대학교 전기컴퓨터공학과 석사과정 재학.

<주관심분야 : 운영체제, 임베디드 시스템, 시스템 소프트웨어>

김형호(준회원)
2020년~현재 인하대학교 컴퓨터공학과 학사 재학.

<주관심분야 : 지능형 임베디드 소프트웨어, 컴퓨터 비전, 엣지 컴퓨팅>

신지우(정회원)
2023년 인하대학교 컴퓨터공학과 학사 졸업.
2025년 인하대학교 전기컴퓨터공학과 석사 졸업.
2025년~현재 인하대학교 전기컴퓨터공학과 박사과정 재학.

<주관심분야 : 지능형 임베디드 소프트웨어, AI 모빌리티>

장준혁(종신회원)
2009년 서울대학교 전기·컴퓨터공학부 학사 졸업.
2015년 서울대학교 전기·컴퓨터공학부 박사 졸업.
2021년~현재 한남대학교 컴퓨터공학과 교수.

<주관심분야 : 시스템소프트웨어, 운영체제, 임베디드시스템>

정진만(종신회원)
2008년 서울대학교 컴퓨터공학과 학사 졸업.
2014년 서울대학교 전기컴퓨터공학과 박사 졸업.
2014년~2021년 한남대학교 정보통신공학과 부교수.
2021년~현재 인하대학교 컴퓨터공학과 부교수.

<주관심분야 : 운영체제, 임베디드 시스템, 시스템 소프트웨어>