Journal of the The Korean Institute of Defense Technology Vol. 07, No. 2, pp. 0015-020, 2025

플러터(Flutter)를 활용한 유도탄 시험세트 소프트웨어 설계

Design of a Missile System Test Set Software Using Flutter

임현재[1)] . 박주현[*,1)] . 홍재연[1)] . 김주원[1)]

Hyunjae Im[1)] . Juhyeon Park[*,1)] . Jaeyeon Hong[1)] . Juwon Kim[1)]

[초 록]

본 논문은 유도탄 시험세트 소프트웨어 개발에 플러터를 도입하는 방안을 제안한다. 기존 MFC, WPF와 같은 Windows 기반 GUI 애플리케이션 개발의 한계점을 분석하고, 플러터의 장점인 높은 생산성, 크로스플랫폼 지원, 모던한 디자인 등을 소개한다. 또한 성능 실험을 통해 플러터의 성능 및 한계점을 확인했다. UI 로드가 있는 상태에서 플러터 애플리케이션은 50Hz까지 안정적인 통신 성능을 발휘했다. 본 연구는 유도탄 시험세트 소프트웨어의 개발 방식 및 품질 발전에 기여할 것으로 예상된다.

[ABSTRACT]

This paper proposes a plan to adopt Flutter for the development of guided missile test set software. We analyze th e limitations of developing Windows-based GUI applications such as MFC and WPF, and introduce the advantages of Flutter, such as high productivity, cross-platform support, and modern design. Also we check the performance and l imitations of Flutter through performance experiments. The Flutter application showed stable communication perform ance up to 50Hz under the load of the UI. This research is expected to contribute to the development paradigm and quality of Missile System Test Set software.

Key Words : MSTS(유도탄 시험세트), Cross-Platform(크로스플랫폼), Flutter(플러터)

1. 서 론

유도탄 시험세트는 유도조종컴퓨터, 탐색기, 구동장치 등 다양한 구성품을 통합하는 과정에서 내/외부 ICD(Interface Co ntrol Document)를 검증하고 유도탄의 기능과 성능을 확인하는 장비다. 장비의 특성과 사업의 요구조건에 따라 유도탄 시험세트(Missile System Test Set) 또는 유도탄 종합점검장비(Missile Assembly Test System)로 불린다. 이 장비는 요구조건에 따라 연구개발 및 양산단계에 활용되며 정비 부대에 배치되어 이용된다.[1] 기존 유도탄 시험세트의 GUI 애플리케이션은 주로 Windows 운영체제가 탑재된 컴퓨터에서 실행되며[2], Microsoft에서 제공하는 MFC(Microsoft Foundation Classes) 또는 WPF(Windows Presentation Foundation) 프레임워크를 활용하여 구현되었다. MFC는 C++ 언어를, WPF는 C#과 XAML 언어를 사용하여 개발하도록 구현되어 있다. 이러한 프레임워크는 게임, PC 메신저, ERP 시스템 등 다양한 Windows GUI 애플리케이션 개발에 활용되었으나, 개발 생산성이 상대적으로 낮고 전문 개발자가 많지 않다는 한계를 가지고 있다.

특히, 그림 1 의 Google Trends 데이터에서 볼 수 있듯이 MFC와 Visual C++에 대한 개발자들의 관심도가 지속적으로 감소하고 있다. 이는 스마트폰 보급 확산으로 인한 모바일 애플리케이션 시장의 성장과 관련이 깊다.[3] 개발자의 관심 감소는 해당 프레임워크를 학습하고 사용하는 개발자의 수가 줄어들고 있을을 의미하며, 취업 시장에서도 Python, Java, Javascript와 같은 언어에 대한 수요가 높아지고 있다. 심지어 대학에서도 C/C++, C# 언어 교육이 감소하고 있는 추세이다. 이러한 현상은 C++와 C# 언어를 사용하는 프로젝트의 경우 개발자 인력 수급에 어려움을 겪을 수 있으며, 이는 개발되는 소프트웨어 품질에도 영향을 미칠 수 있다.

1) ㈜LIG 넥스원 미사일시스템 체계종합연구소(Missile Systems, Naval Surface-to-air Missile Development, LIG Nex1)
* Corresponding author, E-mail: juhyeon.park@lignex1.com

Received : April 30, 2025 Revised : May 21, 2025
Accepted : June 30, 2025

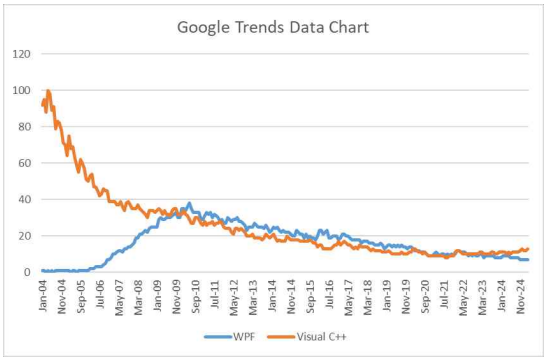

그림 1. WPF 및 Visual C++ 사용자 관심도 추이
Fig. 1. User interest trends for WPF, Visual C++

MFC와 WPF 프레임워크는 Windows GUI 소프트웨어 개발에 특화되어 있으며 성능 면에서는 우수하지만, 산업 환경 변화로 인해 모던한 GUI 디자인, 개발자 채용, 개발 도구 발전 등 다양한 측면에서 한계에 직면해 있다. 이러한 한계점을 극복하고 유도탄 시험세트 소프트웨어의 개발 방식 및 품질을 발전시키기 위해 본 논문에서는 크로스플랫폼 프레임워크인 플러터(Flutter)의 도입을 제안한다.

2. 크로스플랫폼 프레임워크 플러터 소개

2.1 플러터 개요

플러터는 하나의 코드를 기반으로 여러 플랫폼에서 실행 가능한 애플리케이션을 개발할 수 있도록 설계된 크로스플랫폼 프레임워크이다. 구글이 주도하여 개발한 플러터는 모바일(Android, iOS) 디바이스뿐 아니라 데스크톱(Windows, macOS, Linux), 웹, 임베디드 기기까지 지원한다.[4] 플러터의 범용성은 독자적인 구조에서 기인한다. 그림 2는 플러터의 아키텍처를 보여준다. 플러터는 플랫폼 고유의 UI 컴포넌트를 사용하는 대신, 구글에서 개발한 오픈소스 2D 그래픽 엔진 Skia를 활용하여 모든 위젯을 직접 그리는 방식을 택하고 있다. 이를 통해 각 플랫폼의 UI 일관성 문제를 해소하며, 개발자는 동일한 코드로 다양한 플랫폼에서 동일한 디자인과 동작을 구현할 수 있다.

2.2 플러터 사용 사례 및 장점

플러터는 출시 이후 빠르게 성장하며 다양한 산업군에서 실제 서비스에 적용되고 있다. Alibaba는 애플리케이션의 일부 기능을 플러터로 구현하여 배포 주기를 단축하고 유지보수 효율을 높였다. BMW는 전 세계 사용자에게 일관된 차량 애플리케이션 UI/UX를 제공하기 위해 플러터를 선택하였다. 또한 구글 Ads, eBay Motors 등 주요 애플리케이션들도 플러터 기반으로 제작되었으며, 다양한 산업 분야에서 활용 범위가 계속 넓어지고 있다. 실제 서비스에 플러터를 적용한 공통적인 이유는 빠른 개발 속도, UI 일관성, 네이티브 수준의 성능, 커뮤니티와 생태계의 성장성 등을 원인으로 볼 수 있다.[5] 이는 단순한 기술 트렌드를 넘어, 생산성과 사용자 경험을 동시에 고려한 선택임을 보여준다.

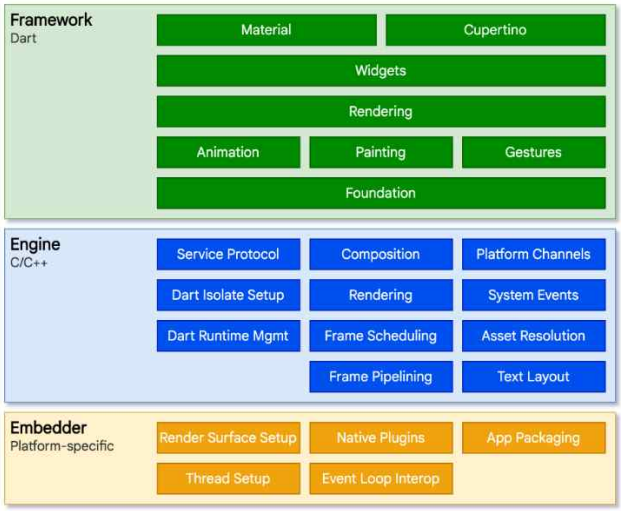

그림 2. 플러터 아키텍처 계층
Fig. 2. Flutter Architectural layers

플러터는 BSD 3-Clause 라이선스를 따르는 오픈소스 프로젝트로, 상업적 활용에 제약이 거의 없다. 이는 개발자들이 자유롭게 소스를 수정하고, 원하는 방식으로 재배포할 수 있음을 의미하며, 기업 환경에서도 라이선스 리스크 없이 채택할 수 있는 중요한 근거이다. 구글은 플러터의 주요 기여자이지만, GitHub 기반의 공개 저장소 운영을 통해 전 세계 개발자들의 참여를 적극 장려하고 있다. 실제로 플러터 GitHub 저장소에는 수만 명의 기여자와 이슈 보고자가 활동하고 있으며, 이를 통해 기능 향상과 버그 수정이 빠르게 이루어지고 있다. 아울러 공식 패키지 저장소인 pub.dev(https://pub.dev)는 수많은 오픈소스 라이브러리와 플러그인을 제공하고 있으며, 이는 개발자의 생산성을 크게 향상시키는 핵심 생태계이다. 오픈소스 프레임워크의 특성상 특정 기업에 종속되지 않고 커뮤니티 중심의 기술 발전이 가능하다는 점도 플러터의 경쟁력이다. 활발한 커뮤니티와 생태계 덕분에 최신 기술 트렌드에 빠르게 대응할 수 있으며, 다양한 개발 환경에서 유연하게 적용 가능하다는 점이 큰 장점으로 작용한다.

이러한 장점으로 인해 대학에서는 모바일 및 크로스 플랫폼 수업에 플러터를 적극적으로 도입하고 있다. 플러터는 직관적인 UI 설계 구조와 접근성 높은 Dart 언어를 기반으로 하여 학생들이 비교적 짧은 시간 내에 애플리케이션을 직접 구현해 볼 수 있어 교육 효과가 뛰어나다. 반면 C/C++은 복잡한 문법과 수동 메모리 관리, 플랫폼 의존적인 빌드 환경 등으로 인해 높은 학습 곡선을 요구한다. 이로 인해 실제로 C/C++ 개발자를 주니어 레벨에서 채용하기는 쉽지 않다. 이에 반해, 대학 교육과 산업 수요의 교차점에서 플러터의 입지는 점차 확고해지고 있으며, 기술 초기 단계에서 신속한 제작이 필요한 프로젝트에서는 C/C++보다 플러터가 더 실용적이고 전략적인 선택이 될 수 있다.

2.3 플러터의 주요 장점

2.3.1 모던한 사용자 인터페이스 제공

플러터는 모바일 중심 프레임워크로 시작된 만큼, 사용자 중심의 모던한 UI 설계에 최적화되어 있다.[6] 개발 초기부터 구글의 디자인 시스템인 Material Design을 깊이 반영하고 있으며, 동시에 iOS 사용자 경험을 위한 Cupertino 위젯도 기본 제공한다. 이에 따라 Android와 iOS 양쪽 모두에 자연스럽고 세련된 UI를 구현할 수 있다. 고급 애니메이션과 복잡한 상태 전이도 상대적으로 간단하게 구현이 가능하다. 플러터는 모든 UI를 자체 렌더링하기 때문에 프레임 단위의 정밀 제어가 가능하며, 이에 따라 부드럽고 반응성 높은 애니메이션을 손쉽게 만들 수 있다. 이러한 점은 특히 사용자 경험이 중요한 분야에서 큰 장점으로 작용한다. 결과적으로 플러터는 시각적 완성도, 디자인 유연성, 성능이라는 세 요소를 고르게 충족시키며, 기존 네이티브 애플리케이션 개발 방식에 비해 훨씬 빠르고 효율적인 UI 구현을 가능하게 한다.[7] 그림 3은 플러터 프레임워크를 기반으로 유도탄 시험세트의 사용자 인터페이스(UI)를 구현한 예시이다.

그림 3. 유도탄 시험세트 UI 예시
Fig. 3. Example of Missile System Test Set UI

2.3.2. 높은 생산성

플러터는 Hot Reload 기능을 통해 개발자가 코드를 변경한 사항을 즉시 반영할 수 있다. Hot Reload는 업데이트된 소스 코드 파일을 실행 중인 Dart 가상머신(VM)에 주입하여 동작하는 방식으로, VM이 새 버전의 필드와 함수로 클래스를 업데이트한 후 플러터 프레임워크는 자동으로 위젯 트리를 다시 빌드하여 변경 사항을 빠르게 확인할 수 있게 된다. 이 기능을 통해 UI를 실시간으로 수정하고 테스트할 수 있으며, 반복적인 작업 시간을 줄인다. 또한, 플러터는 객체지향 프로그래밍과 함수형 프로그래밍을 모두 지원하여, 개발자는 자신의 선호에 따라 코드 구조를 유연하게 설계하고 생산성을 향상시킬 수 있다.

2.3.3 뛰어난 확장성

플러터는 확장성 측면에서도 매우 유리한 구조를 갖추고 있다. 위젯 기반의 모듈화 아키텍처를 중심으로, 각 기능을 독립적인 컴포넌트 단위로 구현할 수 있어 프로젝트 규모가 커지더라도 기능 추가 및 변경이 쉽다. 이러한 구조는 코드의 재사용성과 테스트 가능성을 동시에 확보하며, 유지보수성과 팀 협업 측면에서도 큰 이점을 제공한다. 또한, pubspec.yaml 파일을 통해 의존성 패키지를 체계적으로 관리할 수 있다. 이는 필요한 외부 라이브러리를 손쉽게 프로젝트에 통합할 수 있게 하며, 버전 충돌이나 업데이트 관리에서도 안정적인 환경을 제공한다. 공식 패키지 저장소인 pub.dev에는 수천 개 이상의 고품질 플러그인이 등록되어 있으며, 이를 통해 네트워크, 데이터베이스, 상태 관리 등 다양한 기능을 빠르게 확장할 수 있다.

2.3.4 우수한 성능

플러터 프레임워크는 기존 크로스플랫폼 프레임워크와 달리 브릿지를 사용하지 않고 캔버스에 직접 렌더링을 수행한다. 이를 통해 브릿지를 통해 UI 컴포넌트를 호출하는 React Native 등의 프레임워크에서 발생하는 오버헤드를 줄일 수 있게 된다. 플러터의 렌더링 아키텍처는 크게 UI 스레드와 GPU 스레드로 구분된다. UI 스레드는 widgetTree의 업데이트 이후 RenderTree를 업데이트하며 위젯들의 픽셀화를 수행하고, 그 결과를 LayerTree에 담아 GPU 스레드에 전달한다. GPU 스레드는 LayerTree에 전달된 위젯의 픽셀 정보들에 대해 Composition/Rasterization을 수행하고 최종적으로 디스플레이에 그려낼 화면을 만들어낸다. 이러한 과정에서 하드웨어 가속을 통해 애니메이션과 그래픽 처리 속도가 향상되고, 네이티브 애플리케이션과 유사한 성능을 낼 수 있게 된다.

플러터는 다양한 비동기 프로그래밍 패턴을 지원하며, 이를 통해 blocking/non-blocking 작업을 효율적으로 수행한다. Async와 await을 사용하여 비동기 함수를 작성하거나, future와 stream을 이용하여 비동기 작업을 처리한다. 이를 통해 플러터에서는 코드의 블로킹 문제를 회피하고, 실행 흐름을 유지하며 성능을 보장할 수 있게 된다. 또한, FFI(Foreign Function Interface)를 통해 C/C++로 작성된 네이티브 라이브러리를 직접 호출할 수 있는 기능을 제공하며, 이는 성능이 중요한 연산이나 하드웨어 제어와 같은 저수준 작업에 효과적으로 활용될 수 있다. 이러한 특징은 네이티브 수준의 실행 속도를 요구하는 상황에서 플러터 애플리케이션의 성능을 크게 향상시킬 수 있으며, 기존에 보유한 C/C++ 코드 자산을 그대로 재사용할 수 있다는 점에서 실무 개발자들에게 실질적인 이점을 제공한다.

2.3.5. 기타 장점

플러터는 효율적이고 체계적인 프레임워크 구조를 기반으로 협업 개발에 적합한 환경을 제공하며, 상대적으로 낮은 학습 곡선을 갖춘다는 점에서 주목받고 있다. Dart 언어는 C/C++에 비해 문법이 단순하고 직관적이기 때문에 개발 초기 진입 장벽이 낮고, 이를 통해 팀 내 개발자 간의 코드 이해도와 유지보수 효율성 또한 높아진다. 이러한 환경은 단일 개발자뿐 아니라 팀 단위의 협업에서도 높은 생산성을 보장하며, 코드 일관성과 재사용성 측면에서도 유리하게 작용한다.

2.4 유사 크로스플랫폼과 종합 비교

표 1은 플러터, React Native을 비교한 결과이다.[8] 이를 살펴보면 플러터는 자체 렌더링 엔진을 기반으로 플랫폼과 관계없이 동일한 UI를 구성할 수 있어 사용자 경험 측면에서 뛰

어난 일관성을 제공한다. 또한 풍부한 커스텀 위젯과 매끄러운 애니메이션 구현 능력은 다른 프레임워크 대비 우수한 점으로 평가된다.[9] 개발자는 비교적 짧은 시간 내에 고품질의 애플리케이션을 제작할 수 있다. 다만, 플러터는 비교적 신생 프레임워크인 만큼 일부 플랫폼(특히 데스크톱, 웹, 임베디드)에서는 기능적 제약이나 플러그인 미지원 문제가 존재할 수 있다.

표 1. 플러터, React Native 비교
Table 1. Flutter vs React Native

항목	플러터	React Native
개발 언어	Dart	JavaScript/ TypeScript
UI 렌더링 방식	자체 렌더링 엔진 (Skia)	네이티브 브릿지
성능	매우 우수	중간
위젯 제공	매우 풍부	제한적(플러그인 의존)
코드 재사용성	매우 높음	높음
커뮤니티 및 생태계	활발 (pub.dev)	매우 활발 (npm)
네이티브 접근성	용이함	브릿지 필요

3. 성능 실험 및 결과

일반적으로 유도탄 시험세트 소프트웨어는 고속의 데이터를 처리해야 하므로 성능 실험은 필수적이다.[10] 본 논문에서는 빠른 주기로 출력되는 유도탄의 특성을 고려하여, 플러터 애플리케이션의 성능 실험을 수행한다. 유도탄의 고속 통신을 모의하기 C++ 애플리케이션(이하 'CommTester')을 구현하고, 플러터 애플리케이션(이하 'UI_APP')과의 통신 실험을 수행하였다.

3.1 실험 방법

실험은 CommTester가 송신한 메시지를 UI_APP에서 간단한 처리 후 다시 CommTester에게 송신하는 형태로 수행한다. 실험 구성은 그림 4와 같다.

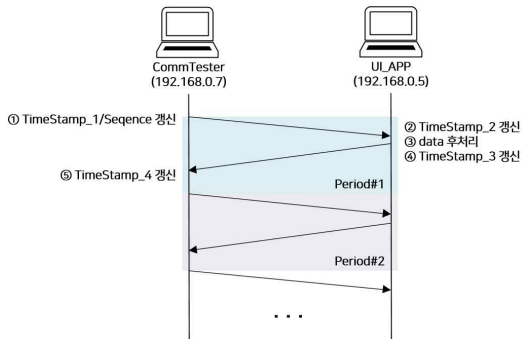

그림 4. 실험 구성
Fig. 4. Experimental Setup

실험 과정에서 각 애플리케이션은 유도탄에서 출력되는 메시지를 모사한 메시지를 송수신 한다. 그림 5는 통신 실험에 사용하는 메시지의 구조를 나타낸다. 메시지는 유도탄에서 출력되는 데이터를 고려하여 1064bytes로 구성하고, 데이터 분석을 위해 세부 필드를 정의 했다. 우선 메시지의 누락 여부를 확인하기 위해 Sequence 필드를 8bytes 할당하고, Data필드에는 1024bytes 데이터를 모사하였다. 또 성능 확인을 위해 TimeStamp필드에 각 프로그램의 송/수신 시각을 할당하여 저장하였다.

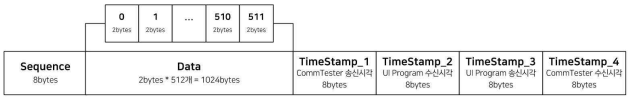

그림 5. 메시지 구조
Fig. 5. Message Structure

표 2. 실험 환경
Table 2. Experimental Environment

항목	UI_APP	CommTester
CPU (프로세서)	Intel® i5-6200U @ 2.3GHz	Intel® i5-7200U @ 2.5GHz
메모리(RAM)	8.00GB	8.00GB
GPU	Intel® HD Graphics 520	Intel® HD Graphics 620
운영체제(OS)	Windows 10 Home	Windows 10 Pro
통신 프로토콜	UDP	

실험은 서로 두 대의 노트북을 이용하여 수행되었으며, 실험 환경은 표 2와 같다. 실험은 먼저 UI 프로그램에 인가되는 부하 정도에 따른 성능 차이를 확인하기 위해, UI 갱신 여부에 따라 실험 세트를 구성하였다. 또한 각 세트는 UI 프로그램이 수용 가능한 통신주기를 확인하기 위해, 통신주기를 20Hz/50Hz/100Hz/200Hz/500Hz로 변경하며 실험을 5회씩 10분 동안 진행했다.

3.2 실험에 활용한 애플리케이션

3.2.1 CommTester

CommTester는 UI의 통신 성능을 검증하기 위해 C++로 작성된 프로그램으로, 통신에는 Boost Asio 라이브러리를 사용한다. 이 프로그램은 송신, 수신 및 로깅 기능을 포함하며, 송신 스레드는 공유 변수 g_msg를 실험 주기에 맞춰 UI 프로그램에 송신한다. 수신 스레드는 UI로부터 비동기 방식으로 메시지를 수신하여 공유 변수 g_msg를 갱신하고, 로깅 스레드는 수신한 메시지 개수가 100이 될 때마다 외부 CSV 파일로 출력한다. 성능 실험이 끝난 후, 출력된 데이터의 누락, 지연, 송수신 주기를 분석하기 위해 CSV 파일을 분석하는 순서로 실험이 진행된다. 이를 통해 CommTester는 UI_APP의 통신 성능을 평가할 수 있는 환경을 제공한다.

3.2.2 UI_APP

UI_APP에서는 패킷의 송수신 및 후처리를 다음과 같은 절차로 수행하였다. 우선, 패킷의 송신 및 수신 시 Packet 클래스의 팩토리 생성자인 toBytes와 fromBytes를 이용하여 Dart의 고수준 자료형과 이진 데이터 간의 변환을 수행한다. toBytes는 객체 형태의 데이터를 직렬화하여 전송 가능한 이진 데이터로 변환하며, fromBytes는 수신된 이진 데이터를 역 직렬화하여 Packet 객체로 복원한다.

수신된 데이터에 대한 후처리 과정을 통해 수신된 data 배열의 각 원소에 정수 1을 더한 새로운 배열을 생성하고, 이를 다시 패킷에 저장하였다. 이와 같은 후처리는 수신 데이터의 검증 및 단순 가공 처리 시나리오를 시뮬레이션하기 위한 것이다. 또한, 각 패킷에는 수신 후 송신까지의 시간을 포함시켰다. stopwatch.elapsedMicroseconds() 메서드를 사용하여 송수신 소요 시간을 정밀하게 기록했다.

3.3 실험 결과

CommTester을 기준으로 (수신시각-송신시각)을 계산하여, RTT(Round Trip Time)를 구하고 이를 통해 실험결과를 분석한다. RTT의 평균을 통해 10분간의 통신에서 평균적으로 발생하는 RTT의 값을 확인하였고, 실험마다 최대 RTT 값을 통해 발생할 수 있는 최대 수준의 지연을 확인하였다. 또한 RTT값 간의 표준편차를 통해 변동성을 확인하여, 얼마나 안정적으로 통신환경이 운용되는지 확인하였다. 마지막으로 RTT가 통신주기보다 커져 주기 내에 다시 CommTester로 도달하지 못해 지연이 발생한 경우인 지터(Jitter)의 발생 횟수를 측정하였다.

표 3은 UI_APP에서 UI 갱신을 하지 않고, 통신주기를 변화시키며 실험을 진행한 결과이다. 실험 결과, 통신주기 약 50Hz 수준까지 지연 없이 안정적으로 통신할 수 있다는 것을 확인할 수 있었다. 하지만 100Hz 이상의 통신에서는 지터가 발생하기 시작하며, 500Hz에서는 20,140개의 무시할 수 없는 수준의 지터가 발생하였다.

표 3. UI 로드가 없는 상황에서 실험 결과
Table 3. Experiments results without UI updates

주기(Hz)	RTT 평균 (us)	RTT 최댓값(us)	RTT 표준편차(us)	Jitter (회)
20	3004.39	172984	3148.70	0
50	3154.69	8701	560.33	0
100	2836.99	33121	588.78	2
200	2251.50	26803	350.90	7
500	1731.37	12550	363.65	20140

표 4. UI 로드가 있는 상황에서 실험 결과
Table 4. Experiments results with UI updates

주기(Hz)	RTT 평균 (us)	RTT 최댓값(us)	RTT 표준편차(us)	Jitter (회)
20	4238.23	27142	3458.39	0
50	3001.91	27036	1872.38	0
100	2139.80	23251	950.07	4
200	2252.80	22052	752.88	97
500	2144.94	16442	819.775	43991

표 4는 UI_APP에서 UI 갱신을 수행하면서, 통신주기에 따라 실험을 진행한 결과이다. 실험 결과, UI_APP에 UI 갱신에 의한 부하가 걸리더라도 통신주기 약 50Hz 수준까지 지연 없이 안정적으로 통신할 수 있다는 것을 확인할 수 있었다. 하지만 높은 수준의 주기 통신에서는 이전 실험보다 더 많은 수준으로 지터가 발생하는 것을 확인하였다.

결과적으로 표 3과 표 4의 실험 결과를 비교하면, UI 갱신 여부와 관계없이 모두 50Hz 주기 통신을 수행하는 데에는 문제가 없었지만, 그 이상의 주기로 통신을 수행할 때 뚜렷한 차이를 보였다. UI 갱신이 있는 경우, 지터 발생 횟수가 크게 증가하였으며 표준편차도 크게 증가하여 비교적 불안정한 주기 통신을 수행한다는 것을 확인할 수 있다. 그림 6은 50Hz 주기 통신에서 발생하는 데이터를 100개씩 샘플링하여 평균 RTT를 계산하고, 변화 추세를 확인한 결과이다. 이를 통해 UI 갱신 여부에 따라 통신의 안정성이 크게 영향을 받는다는 것을 확인할 수 있다.

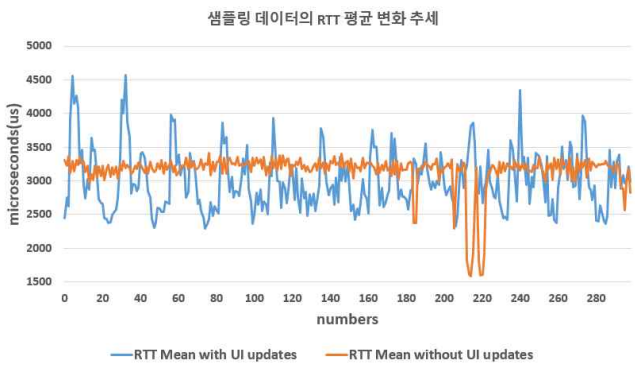

그림 6. RTT 평균 변화 추세.
Fig. 6. RTT average changing trend

본 실험을 통해서, 플러터 프레임워크를 사용한 애플리케이션의 UI 부하 여부에 따른 영향성을 확인할 수 있었다,

4. 결 론

본 논문에서는 유도탄 시험세트 개발 시에 기존 방식과 비교하여 다양한 장점을 갖고 있는 플러터 프레임워크를 이용한 새로운 개발 방식을 제안한다. 플러터를 사용하면 크게 4가지

장점을 확인 할 수 있었다. 첫 번째로 기존 프레임워크보다 생산성이 높다. 두 번째로 모던한 디자인의 애플리케이션 개발이 가능하다. 세 번째로 경쟁 프레임워크에 비해 성능이 우수하다. 네 번째로 주니어 개발자 채용이 C++와 C#언어에 대비 쉽다.

플러터의 성능 실험에서는 유도탄 출력 모사 데이터를 UDP Protocol로 전송할 때 100Hz 이상의 부하가 가해지면 성능 저하가 발생하는 것을 확인하였다. 그러나 이러한 성능 저하는 UI 부하, 통신 데이터의 크기, 네트워크 환경, PC 성능 등 다양한 요인에 따라 달라질 수 있으므로, 로드 분석 및 실험이 필요하다. 만약 플러터의 성능으로 처리할 수 없는 양의 고속/대용량 통신이 요구되는 경우 실시간 통신을 중계하는 애플리케이션을 별도로 구현하는 아키텍처를 고려할 수 있다.

향후 연구는 플러터 유도탄 시험세트 소프트웨어를 실제로 개발하여 그 성능과 한계점을 확인할 것이다. 그리고 플러터가 크로스플랫폼 프레임워크인 점을 활용하여 모바일 기기에서의 동작 성능을 확인할 예정이다. 과거 하드웨어 장치 전면 패널의 램프를 활용한 점검 방식에서 Windows GUI 애플리케이션으로 점검 방식이 발전한 것처럼 모바일 기기 화면과 터치 기능을 활용한 점검 소프트웨어 개발로 패러다임의 변화를 불러올 수 있다.

References

[1] S. Ko, S. Han, G. Lee, Y. Lee, Y. Kim and D. Park, "Development Method for Improving the Reliability of Missile Inspection Equipment," J. Korea Academia-Industrial Cooperation Society, Vol. 19, No. 8, pp. 37-43, 2018.

[2] J. H. Choi, H. M. Jo, S. J. Yun and D. W. Ryu, "A Design of the Integrated Software Architecture for Missile System Test Set," Proc. Symp. Korean Institute of Communications and Information Sciences, Daejeon, pp. 214-215, 2014.

[3] Keith Ward, “Study: 85 percent of Windows Developers Are Asked to Go Mobile”, 2013, Retrieved from https://visualstudiomagazine.com/articles/2013/09/12/85-percent-of-win-devs-asked-to-go-mobile.aspx

[4] A. Tashildar, et al., "Application Development Using Flutter," Int. Res. J. Modernization in Engineering Technology and Science, Vol. 2, No. 8, pp. 1262-1266, 2020.

[5] K. Nagaraj, B. Prabakaran and M. O. Ramkumar, "Application Development for a Project using Flutter," Proc. 2022 3rd Int. Conf. Smart Electronics and Communication (ICOSEC), pp. 947-951, 2022.

[6] Y. Song and H. Lee, "A Case Study on Usability Enhancement through UI/UX Continuity in the Flutter Framework," Proc. Korean Institute of Communications and Information Sciences Conference, pp. 1144-1145, 2024.

[7] Damian Białkowski and Jakub Smołka, "Evaluation of Flutter Framework Time Efficiency in Context of User Interface Tasks," J. Comput. Sci. Inst., Vol. 25, pp. 309-314, 2022.

[8] E. Gulcuoglu, A. B. Ustun and N. Seyhan, "Comparison of Flutter and React Native Platforms," J. Internet Applications and Management, Vol. 12, No. 2, pp. 130-143, 2021.

[9] B. Suri, S. Taneja, I. Bhanot, H. Sharma and A. Raj, "Cross-Platform Empirical Analysis of Mobile Application Development Frameworks: Kotlin, React Native and Flutter," Proc. 4th Int. Conf. Information Management & Machine Intelligence (ICIMMI), pp. 60:1-60:6, 2022.

[10] H. Kim, Y. Huh and B. Kwon, "Design and Performance Verification of Real-Time Inspection Equipment Software Based on Windows Operating System," J. Korea Contents Association, Vol. 17, No. 10, pp. 1-8, 2017.