임베디드 미들웨어 및 플랫폼 시험

이 상 수 | TTA 시험인증연구소 SW시험인증팀 전임연구원
신 석 규 | TTA 시험인증연구소 SW시험인증팀 팀장

1. 서론

임베디드 시스템은 다양한 하드웨어와 운영체제를 이용하여 구성이 가능하다. 그러므로 임베디드 미들웨어는 하드웨어와 시스템 소프트웨어에 구애받지 않으며, 미들웨어 표준을 준수하는 공통 프레임워크를 응용프로그램에 제공할 필요가 있다. 이를 위해서는 미들웨어가 표준에 적합하도록 설계되었는지를 확인하는 표준적합성 시험과 다양한 임베디드 시스템 및 운영체제에서 작동하는지를 확인하는 상호운용성에 대한 시험을 통해 소프트웨어의 품질을 확보할 필요가 있다[1, 2, 3].

임베디드 플랫폼은 데스크톱용 시스템과 다르게 하드웨어를 구성할 때, 특정 목적에 따라 다양한 장치를 사용하며, 시스템의 구조가 표준화되어 있지 않는 것이 특징이다. 그러므로 임베디드 플랫폼은 통합된 시스템에서 기능의 정상동작 여부 확인이 필요하다[4, 5].

여기서는 임베디드 미들웨어 및 플랫폼의 효과적인 시험법으로 임베디드 미들웨어의 표준적합성 및 상호호환성을 시험하는 방법과 임베디드 시스템 테스트 방법에 대해 알아보고자 한다.

2. 임베디드 미들웨어 테스트 방법

2.1 임베디드 미들웨어

임베디드 미들웨어는 위성 및 지상파 DMB 등의 서비스를 휴대폰 및 이동단말기 등의 임베디드 시스템에서 제공하기 위해 사용된다. 임베디드 미들웨어는 해당기기의 주 기능 동작에 영향을 미치지 않고 안정적인 상태로 동작해야 한다. 이를 위해 미들웨어는 일정수준의 품질 확보를 위해 표준에 적합한지, 다양한 운영체제에서 동작 가능한지가 테스트되어야 한다.

2.2 임베디드 미들웨어 시험방법

임베디드 미들웨어를 테스트하는 방법은 미들웨어가 제공하는 API 자체를 화이트박스 기법으로 테스트하는 정적분석 방법과 API를 사용하여 구현한 기능을 실시간으로 실행하여 테스트하는 동적분석 방법이 있다.

정적분석 방법은 API의 입력 파라미터를 변경하여 출력의 정상 유무를 확인하는 방법으로 해당 API의 무결성을 검증하기 위한 방법이다. 이 방법을 통해 API 각각에 대한 정상동작 유무를 확인할 수 있으나, API 간의 상호작용에 따른 기능의 정상유무는 확인할 수 없다. 동적분석 방법은 표준 또는 기술문서에서 제안하는 기능을 여러 API를 사용하여 구현한 후, 실시간으로 실행하여 기능의 정상 동작여부를 확인하는 시험방법이다. 이를 사용하면 API 간의 상호작용의 정상동작 여부 및

미들웨어 표준 준수여부를 확인할 수 있다. 미들웨어의 정상동작 여부는 정적분석 방법과 동적분석 방법을 병행해서 표준적합성 및 상호운용성 테스트를 통해 확인할 수 있다.

2.3 임베디드 미들웨어 결함모델

임베디드 미들웨어는 시스템 소프트웨어와 응용 소프트웨어의 중간에서 특정 응용분야에 최적화된 공통 프레임워크이다. 일반적으로, 미들웨어는 그림 1에서와 같이 응용프로그램과 운영체제 중간에 위치하고 있다. 응용프로그램, 미들웨어, 운영체제 및 하드웨어 계층은 각각 인터페이스 및 API 함수를 통해 상호 호출될 수 있다. 본 논고에서는 각 계층 간의 상호호출 시 발생하는 결함을 미들웨어 결함, 운영체제 결함, 하드웨어 결함으로 정의한다. 응용프로그램으로부터 ① API 호출을 받은 미들웨어는 ④ API 리턴을 통해 결과를 되돌리는 상호호출 시 발생하는 결함을 미들웨어 결함으로 정의하고 미들웨어가 운영체제에 ② 시스템 콜을 보내면 ⑤ 시스템 리턴을 되돌리는 상호호출 시 발생할 수 있는 결함을 운영체제 결함이라고 하며, 운영체제가 상호호출(③, ⑥)을 하는 경우 발생하는 결함을 하드웨어 결함으로 정의한다.

일반적으로, 데스크톱용 미들웨어는 단일 운영체제를 사용하여 개발하므로 ② 시스템 콜 및 ⑤ 시스템 리턴의 상호 호출에 관한 처리가 표준화되어 있기 때문에 미들웨어 결함만을 다루면 된다. 그러나 임베디드용 미들웨어의 경우, 설계자는 여러가지 운영체제에서도 동작할 수 있도록 개발하여야 하므로 다양한 운영체제가 제공하는 시스템 콜 및 리턴 등의 상호호출에 대해서도 상호운용이 가능한지를 고려해야 한다. 이러한 설계단계의 복잡성으로 인해, 동일 미들웨어를 여러 단말기에 설치할 경우 일부 이동단말기는 정상적으로 동작하지 않는 경우가 발생하게 된다. 물론 이러한 문제를 해결하기 위해 하드웨어 종류에 구애받지 않도록 하기위해 이동용 단말기 미들웨어를 자바로 개발할 수도 있으나, 이동용 단말기 하드웨어 자원의 한계 등으로 인해 현실적으로는 어려운 실정이다.

임베디드 미들웨어 테스트는 데스크톱용이 미들웨어 결함만 고려하는 것과 다르게 미들웨어 결함 뿐만 아니라 운영체제 결함도 함께 고려한다. 임베디드 미들웨어 결함은 운영체제 결함과 연관성이 있으며, 미들웨어의 상호운용성을 보장하기 위해서는 운영체제 결함시험이 필요하다. 운영체제 결함시험을 위한 상호 운용성 테스트는 하나의 테스트 케이스에 대해 운영체제 개수만큼 반복적이고 기계적으로 이루어진다.

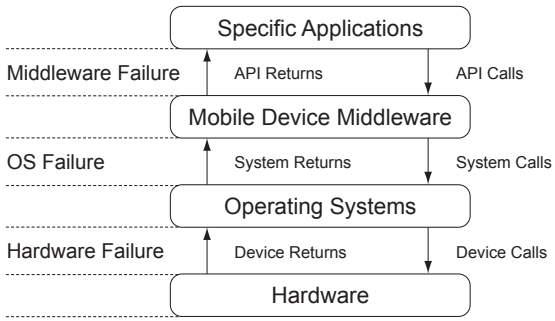

그림 1. 임베디드 미들웨어 결함 모델

2.4 표준적합성 테스트

일반적으로, 미들웨어는 미들웨어 표준 및 기술문서를 기반으로 개발된다. 그림 2에서와 같이, 미들웨어 표준이 ① 필수적인 기능과 ② 부가적인 기능으로 구성되어 있으면 개발자는 여러 개의 API 함수로 해당 기능을 구현하게 된다. 즉, 표준에서 요구하는 기능은 여러 개의 API가 조합으로 구현될 수 있기 때문에 미들웨어 표준적합성 테스트는 단일 API의 정상 동작을 확인하는 유닛 테스트와 API의 유기적 조합으로 구현된 기능의 정상 동작여부 및 표준 적합성 여부를 살피는 기능 테스트가 병행되어야 한다. 단일 API 함수의 정상동작 여부 확인만으로는 미들웨어 표준이 제안하는 기능의 정상동작 여부는 알 수 없으므로 기능 테스트는 표준 준수여부 확인을 위해 필수적으로 수행되어야 한다.

유닛 테스트 및 기능 테스트를 위해 활용할 수 있는 문서로 API 함수 사용설명서 및 기능구현 설명서가 있다. 유닛 테스트는 API 사용설명서를, 기능 테스트는 기능구현 설명서를 참조하여 테스트할 수 있다. 그림 4에서는 유닛테스트와 기능 테스트 모델을 보여주고 있

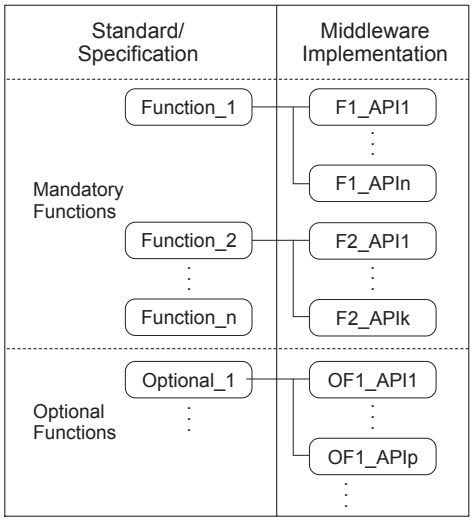

그림 2. 임베디드 미들웨어 표준 및 구현

다. 유닛테스트는 API의 요구사항인 입/출력 파라미터에 대해 함수의 리턴값을 검증하는 것이며, 기능 테스트는 미들웨어를 사용하여 구현한 기능의 입/출력 및 리턴값을 실시간으로 실행하여 검증하는 것이다. 기능 테스트는 미들웨어 표준에 명시되어 있는 기능에 대한 검증이므로 표준적합성 테스트에 해당한다.

미들웨어의 표준적합성 테스트는 앞에서 밝힌바와 같이 유닛 테스트 및 기능모듈 테스트가 같이 수행되어야 의미가 있다. 미들웨어 표준에서 제안하는 기능은 API 함수의 유기적인 결합으로 구현되기 때문이다.

앞서 제안한 것과 같이 유닛 테스트 및 기능 테스트는 정적분석 및 동적 분석방법을 사용한다. 주로 정적분석 방법은 유닛테스트 시 화이트 박스 테스트를 사용하며 동적분석 방법은 기능 테스트에서 블랙박스 테스트를

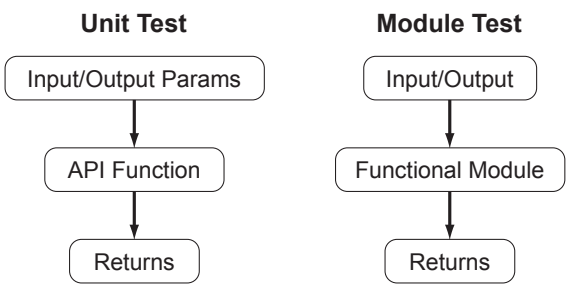

그림 3. 유닛 테스트 및 기능모듈 테스트 모델

사용하게 된다. 두 가지 시험을 수행하는데 있어서 주의할 점은 유닛 테스트와 기능 테스트 두 가지 테스트 중 하나를 빠뜨리고서는 표준적합성 만족 여부를 확인할 수 없다는 점이다. 왜냐하면 각 API 함수가 정적분석을 통해 무결성이 증명되더라도 이를 조합하여 구현한 기능이 실시간으로 실행될 때 결함이 없다고 할 수 없으며, 표준에 명시된 기능이 정상동작하더라도 그 기능을 구현한 각 API에 결함이 없다고 할 수 없기 때문이다. 즉, 표준적합성 테스트는 유닛 테스트와 기능 테스트를 병행하여 이루어져야 한다.

2.5 시스템 소프트웨어 상호운용성 테스트

그림 1에서와 같이 임베디드 미들웨어는 운영체제에 ② 시스템 콜을 하고 운영체제는 미들웨어에 ⑤ 시스템 리턴을 되돌린다. 일반 PC용 미들웨어의 경우 하드웨어 및 운영체제가 표준화되어 있어 시스템 콜 및 시스템 리턴이 단일화되어 있다. 그러나 임베디드 운영체제는 다양한 하드웨어와 운영체제로 인해 시스템 콜 및 리턴이 달라 각각의 하드웨어 및 운영체제를 지원해야 한다.

임베디드 미들웨어 상호운용성 테스트는 미들웨어가 다양한 하드웨어 및 운영체제로 구성된 멀티 플랫폼에서 결함없이 동작하는지를 확인하는 시험이다. 일반적으로 임베디드 미들웨어는 단말기에 특정 하드웨어(예: DMB 수신기)를 설치한 후 이를 이용하여 서비스를 제공하는 특징이 있다. 이때 대부분 운영체제는 특정 하드웨어의 디바이스 드라이버를 제공하지 않아 미들웨어가 해당 하드웨어의 디바이스 드라이버를 제공해야 한다. 예를 들어, PDA 및 휴대폰에 WinCE 및 임베디드 리눅스 운영체제를 각각 설치한 후 DMB 서비스를 제공하려고 하면 WinCE 및 임베디드 리눅스용 DMB 하드웨어 디바이스 드라이버를 모두 제공해야 한다.

미들웨어가 멀티 플랫폼을 지원하기 위해서는 드라이버 포팅과 시스템 콜 포팅이 필요하다. 표 1은 멀티 플랫폼 지원 체크리스트로써 미들웨어가 각 플랫폼의 시스템 콜 및 디바이스 드라이버의 포팅과 시스템 콜 및 디바이스 드라이버 인터페이스를 지원하는지를 시험하기 위한 체크리스트의 예이다. 테스트 기술자는 표 1과 같은 체크리스트를 활용하여 각각의 운영체제에 대해

표 1. 멀티 플랫폼 지원 체크 리스트

대항목	멀티 플랫폼 지원여부		
	플랫폼1	플랫폼2	플랫폼3
System Call Porting	Y/N	Y/N	Y/N
Device Driver Porting	Y/N	Y/N	Y/N
System Call Interface	Y/N	Y/N	Y/N
Device Driver Interface	Y/N	Y/N	Y/N

반복적인 테스트를 수행하여 미들웨어의 상호운용 가능 여부를 확인할 수 있다.

3. 임베디드 플랫폼 테스트 방법

일반적으로, 임베디드 시스템은 PC 기반 시스템과 하드웨어 및 소프트웨어가 달라 테스트 방식 또한 다르다.

PC 기반 시스템은 주요 입/출력 장치가 표준화되어 있어 공통된 테스트 방식을 사용할 수 있다. 그러나 임베디드 시스템은 사용목적에 따라 다양한 형태의 하드웨어를 사용하고 입/출력 장치가 다양하므로, 해당 시스템에 맞는 테스트 방식을 사용해야 한다[6]. 이에 따라 임베디드 시스템의 개발단계별 테스트 항목을 살펴보고 각 단계별 테스트 시 고려할 사항을 살펴본다.

3.1 개발단계별 테스트

임베디드 시스템의 개발단계에 따른 테스트 방법은 소프트웨어 공학에서 제시하는 V&V(Verificaiton & Validation) 모델에 따라 적합한 테스트 방법을 적용한다.

그림 4에서 임베디드 시스템의 개발 단계별 테스트 방법을 보이고 있다. 임베디드 시스템의 개발단계는 하드웨어 단계, O/S 단계, Transport 단계, Control Logic 단계 그리고 System 통합 단계로 구분될 수 있다.

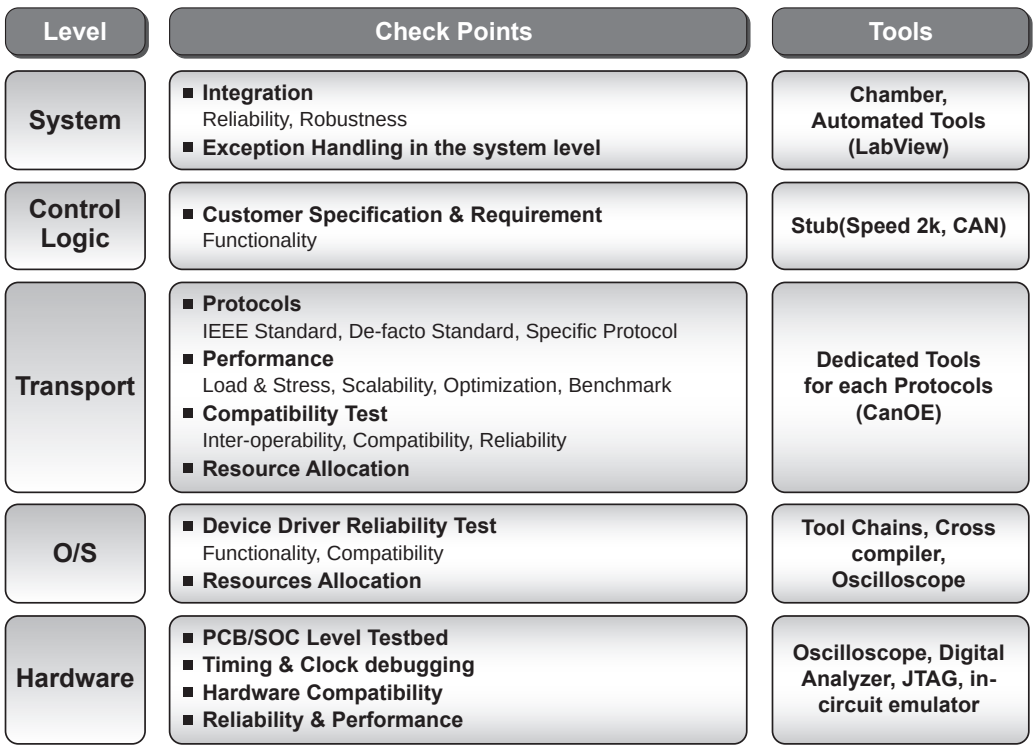

그림 4. 임베디드 시스템 개발단계별 테스트 방법

3.2 하드웨어, O/S 개발, Transport 단계 테스트

하드웨어와 O/S 개발 단계에서는 오실로스코프, JTAG 및 컴파일러를 이용하여 하드웨어 정상동작 여부를 테스트한다[7, 8]. 이 단계에서는 정적 및 동적분석을 활용한 화이트박스 및 블랙박스 테스트 기법의 소프트웨어 테스트 방식과 로직 게이트레벨의 칩 타이밍 분석의 하드웨어 테스트 방법을 병행한다. Transport 단계는 임베디드 미들웨어 등에 대한 테스트이며, 이는 앞에서 설명한 바와 같이 표준적합성 및 상호운용성 등을 테스트한다.

3.3 Control Logic 레벨 테스트

Control Logic 레벨에서는 PC 기반의 일반 응용 소프트웨어에서 사용하는 블랙박스 테스트를 적용할 수 있다. 단, 입/출력이 PC 기반과 같이 표준화되어 있지 않고 PC 기반과 다르므로 해당 시스템에 따라 테스트 시스템을 구성하여야 한다.

3.4 시스템 통합 레벨 테스트

임베디드 시스템은 단일 시스템에서 정상동작하더라도 시스템 통합 후, 정상동작 여부는 보장할 수 없으므로 통합시스템 기반의 테스트가 필수적으로 요구된다. 시스템 통합 테스트(System Validation Test)를 수행하기 위해서는 다른 시스템과 실제 통합(System Integration)이 필요하다. 그러나 일반적으로, 테스트를 위한 통합시스템 구성에는 많은 제약이 따른다. 따라서 시스템의 실제 센서나 모터 등의 장비 대신 아날로그 및 디지털 신호 발생기를 사용하여 테스트 환경을 구성한다. 예를 들어 디지털 및 아날로그 입력을 만들기 위해 그림 5와 같이 Multi Funtion I/O 및 Wavefrom Generator를 이용하거나 출력을 생성 및 모니터링하기 위해 Multi function I/O를 사용하여 시스템 통합 테스트 환경을 구성한다.

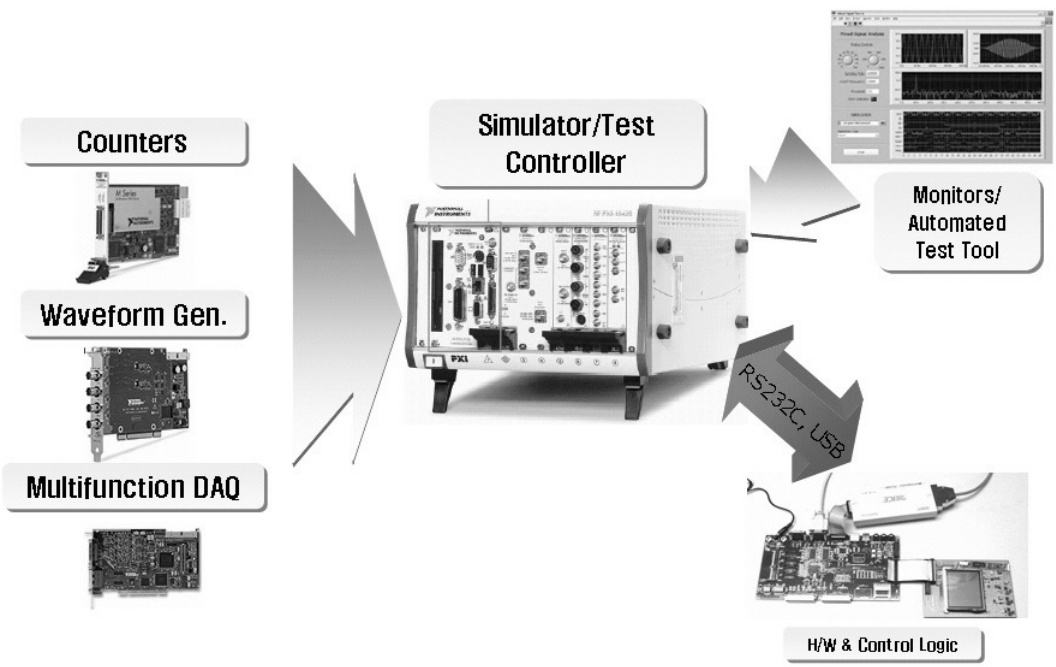

그림 5. 임베디드 테스트 하드웨어 구성 예

4. 결론

임베디드 미들웨어 테스트의 표준적합성 및 상호운용성 시험을 위해서는 정적분석과 동적분석이 병행되어야 함을 강조하였다. 이 방식을 이용하여 임베디드 미들웨어를 테스트할 경우, 정적분석과 동적분석을 병행한 상호호환적 테스트 방법을 이용하여 미들웨어의 표준적합성 및 상호운용성을 시험할 수 있다.

임베디드 플랫폼 테스트가 하드웨어 및 소프트웨어 개발 단계별로 다양함을 보였고 임베디드 시스템의 특성 상 시스템 통합 테스트가 중요함을 보였다.

임베디드 미들웨어 및 플랫폼은 임베디드 시스템의 다양성을 만족하도록 설계되어야 하며 안정적인 동작을 위해 일정한 품질이 보장되어야 한다. 제안한 미들웨어 및 플랫폼 테스트 방법을 적용하여 해당 시스템의 품질을 보장하는데 활용할 수 있다.

[참고문헌]

[1] E.M. Clarke, E.A. Emerson, and A.P. Sistla. "Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specification", ACM Transactions on Programming Languages and Systems, 8(2), April 1986

[2] M.J.C. Gordon, "HOL: aproof generating system for higher-order logic", VLSI Specification, Verification and Synthsis, 73-128, 1988

[3] R.P. Kurshan, "Reducibility in analysis of coordination", LNCS, volume 103, 19-39, 1987

[4] J.Burch, E. Clarke, D.Dill, L.Hwang, and K. McMillan, "Symbolic model checking: 10^{20} states and beyond", Proceedings of the 5th IEEE Symposium on Logic in Computer Science, 1990

[5] H. Ben-Abdallah, D.Clarke, I. Lee, and O.Sokolsky, "PARAGON: A Paradigm for the Specification, Verification, and Testing of Real-Time Systems", IEEE Aerospace Conference, Vol. 2, 469-488, Feb 1-8, 1997

[6] Gray Pollice, Testing Embedded Software

[7] Madisetti, Vijay K., Akgul, Tankut Debugging Embedded Systems, 2006.

[8] Broekman, Bart & Notenboom, Edwin, Testing Embedded Software TTA