KIPS Tr. Software and Data Eng.
Vol.3, No.7 pp.243~254 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2014.3.7.243

Reconfigurable Test Execution Machine for Embedded System

Kim Kyoung Jin† · Chung Ki Hyun†† · Choi Kyung Hee†††

ABSTRACT

When building a testing environment with a testing platform, the configuration of test executor and its interface should be built to be appropriate for the system under test (SUT). That is, it is necessary to build the test executor and interface environment that can properly handle the input and output signals of SUT. If the testing platform is not extendable, it should be modified significantly whenever new SUTs and models are tested. It is a serious drawback that the test executor and interface configuration need to be modified depending on testing targets. To overcome the drawback, this paper proposes TEM(test Execution Machine), which allows for test executor to reconfigure its environment suitable to new SUTs by modifying the configuration file. The proposed TEM is verified through testing two real systems.

Keywords : Testing, Embedded System, Test Executor, Realtime System

재구성이 가능한 임베디드 시스템 테스트 실행기

김 경 진† · 정 기 현†† · 최 경 희†††

요 약

테스트 플랫폼을 이용하여 테스트 환경을 꾸밀 때는 테스트 대상에 맞게 테스트 실행기와 인터페이스 환경을 구성하여야 한다. 즉, 테스트 대상의 입출력에 해당하는 신호를 처리할 수 있는 테스트 실행기와 인터페이스 환경을 구축하여야 한다. 따라서 테스트 플랫폼이 확장성이 없다면, 새로운 모델이나 다른 테스트 대상에 테스트마다 큰 수정이 불가피하고, 그에 맞는 새로운 테스트 실행기 및 인터페이스 환경을 구성해야 하는 단점이 있다. 본 논문에서는 이러한 문제점을 해결하기 위해서 간단한 실행조건 파일(configuration file)의 수정을 통해서 여러 테스트 대상에 적용 가능한 재구성이 가능한 실시간 테스트 실행기 TEM(test Execution Machine)를 제안한다. 제안한 TEM은 두 개의 실제 시스템 테스트를 통하여 그 효용성을 검증한다.

키워드 : 테스트, 임베디드 시스템, 테스트 실행기, 실시간 시스템

1. 서 론

현대사회에서 임베디드 시스템의 중요성은 날로 높아져가고 있다. 실제로 사회 활동의 대부분이 스마트폰을 비롯한 각종 스마트 제품들과 같은 임베디드 시스템을 기반으로 행해지고 있다. 이러한 상황에서 임베디드 시스템의 신뢰성 및 안정성의 확보는 반드시 필요한 요소가 되었다.[1][2][3] 이러한 신뢰성 및 안정성이 확보되지 않는다면 전반적인 사회 활동에 큰 문제를 발생시키고 더 나아가 심각한 인적, 물적 손실을 유발할 수 있다.

실예로 1985년 캐나다 AECL에서 개발한 방사선 치료기인 'Therac 25'라는 제품은 제품 출시 후 약 20개월 동안 6번의 소프트웨어 오류에 의한 사고로 3명이 죽고 3명은 심각한 방사능 후유장애에 시달려야 했다. 또한 1996년에는 아리안 5 로켓은 정상적으로 날고 있던 로켓이 공중에서 폭발하는 사고가 있었다. 2001년에도 역시 파나마에서 미 육군의 치누크 헬리콥터가 추락하여 28명이 사망하는 사건이 있었다. 간단한 소프트웨어 결함에도 치명적인 인명사고와 많은 경제적 손실을 야기할 수 있다는 것을 보여주는 대표적인 사례들이다.[4][5]

이러한 사고들을 예방하기 위해서는 충분한 테스트가 수행되어야 한다. 이를 위해서는 테스트 환경이 구성되어야 한다. 테스트 환경이란 말 그대로 테스트를 진행할 수 있는 환경을 의미한다. 테스트 환경은 세 가지 구성요소로 나눌

† 준 회 원 : LG전자 MC사업부 연구원
†† 정 회 원 : 아주대학교 전자공학과 교수
††† 정 회 원 : 아주대학교 컴퓨터공학과 교수
Manuscript Received : November 12, 2013
First Revision : March 21, 2014
Accepted : March 21, 2014
* Corresponding Author : Chung Ki Hyun(khchung@ajou.ac.kr)

수 있다. 첫 번째는 테스트 대상 시스템(SUT: System Under Test)이다. 이는 신뢰성 및 안정성을 확인하고자 하는 제품이다. 둘째는 테스트 실행기이다. 테스트 실행기란 테스트 대상에 원하는 입력 신호를 입력하고 이에 따른 출력을 읽어서 기록하는 장비이다. 셋째는 인터페이스(interface) 환경이다. 인터페이스 환경은 테스트 실행기와 테스트 대상을 연결해주는 환경을 의미한다.[6][7]

일반적으로 테스트 환경을 꾸밀 때는 단일 테스트 대상에 맞게 테스트 실행기와 인터페이스 환경을 구성하여 테스트를 진행한다. 즉, 테스트 대상의 입출력에 해당하는 신호만 처리할 수 있는 테스트 실행기와 인터페이스 환경을 구축한다. 이 경우에는 확장성이 많이 떨어져 새로운 모델이나 다른 테스트 대상에 대해서 적용할 수 없는 문제점이 있다. 따라서 매번 새로운 모델의 제품이 출시될 때 그에 맞는 새로운 테스트 실행기 및 인터페이스 환경을 구성해야 하는 단점이 있다.

입출력 장치가 추가되거나 삭제되는 다양한 여러 테스트 대상을 데스트하기 위해 테스팅 엔진 자체의 수정 없이 추가되는 입출력 장치를 위한 하드웨어 변경과 이를 위한 드라이버만을 등록하므로 새로운 테스트 환경을 꾸밀 수 있는 재구성성이 있다면 위 단점을 해결하는 데 크게 도움이 될 것이다.

본 논문에서는 이러한 문제점을 해결하기 위해서 간단한 실행조건 파일(configuration file)의 수정을 통해서 입출력 장치가 추가되거나 삭제되는 다양한 여러 테스트 대상에 적용 가능한 재구성이 가능한 테스트 실행기를 제안한다.

테스트 실행기의 성능 중, 가장 기본적인 것 중 하나는 얼마나 정확한 입력 신호 값을 원하는 시점에 입력하는 점과 테스트 대상의 출력 값을 얼마나 정확히 원하는 시점에 읽어올 수 있는 점이다. 정확한 입력 신호 값과 출력 값을 처리하는 것은 충분한 분해능과 샘플링 주파수(sampling frequency)를 가지는 D/A 변환기, A/D 변환기 장비를 구비한다면 쉽게 해결할 수 있는 문제이다. 또한 원하는 시점에 입력 값을 입력하거나 출력 값을 읽어오는 것은 테스트 실행기가 실시간성을 가진다면 만족할 수 있는 문제이다.

간단한 예를 들면 입력 신호로 A를 입력하였을 때 출력 신호의 값이 B로 바뀐 후 50ms 후에 C로 바뀌는 기능을 가진 시스템이 있을 때 입력 신호 A를 입력하고 나서 50ms 이내로 출력 신호의 값을 읽고 50ms 이후에 출력 신호의 값을 읽어서 B에서 C로 값이 바뀌는 것을 확인해야만 해당 기능이 정상적으로 동작했음을 테스트를 통해서 확인했다 할 수 있다. 하지만 실시간성을 만족 못 한다면 테스트를 통해 얻은 값이 실제 테스트 대상이 출력하는 출력 신호의 값과 다를 수가 있다.

이러한 실시간성을 확보하는 방법으로는 실시간 운영체제(Real-time OS) 기반으로 테스트 환경을 구성하는 방법과 실제로 실시간 시스템을 직접 설계하는 방법이 있다. 본 논문에서 제안하는 테스트 실행기는 National Instruments사(NI사)가 제공하는 LabVIEW Real-time module이라는 실시간 운영체제를 기반으로 구성하였다. LabVIEW Real-time module을 사용할 경우 NI사에서 판매하는 여러 장비들을 다른 추가 작업 없이 제공해주는 라이브러리만을 사용하여 구동시킬 수 있다는 장점이 있다. 또한 다른 하드웨어가 필요 없이 일반적인 PC부품을 사용하여 구성할 수 있어 확장성이 좋다는 장점이 있다.

이를 이용하여 테스트 실행기를 구성하면 비교적 간단한 작업을 통해서 기본 기능을 충분히 만족할 뿐만 아니라 여러 테스트 대상에 적용이 가능할 수 있게 구성할 수 있다.

제안하는 테스트 실행기가 가지는 가장 큰 장점은 재구성성(reconfigurable)한 특성을 가지고 있다는 것이다. 이는 FPGAs에서 말하는 재구성 컴퓨팅(reconfigurable computing)과 유사하다.[8] 재구성 컴퓨팅이란 FPGAs나 프로그래머블 하드웨어(programmable hardware)에서 사용되는 알고리즘으로 간단히 설명하자면 설정에 따라서 로직이나 프로그램 실행이 변경되는 것을 의미한다.

제안하는 테스트 실행기의 재구성성은 기본적인 기능인 입력 신호의 생성과 출력 신호의 수집이 설정에 따라서 변경되는 것을 의미한다. 예를 들어서 실행조건 파일에 입력 신호로 아날로그 신호를 2개, 출력 신호로 디지털 신호를 1개로 설정하면 테스트 실행기는 아날로그 신호 2개를 생성하고 디지털 신호 1개를 수집한다. 이처럼 기본적인 틀을 가지고 있는 상태에서 입출력에 관련된 재구성 가능 오브젝트(configurable object)를 실행조건 파일에 해당 내용을 기술하여 선택적으로 사용함으로써 재구성성을 만족할 수 있도록 한다.

제안하는 실행기는 디지털, 아날로그, CAN(Control Area Network), PWM(Pulse Width Modulation) 등의 일반적인 입출력을 가진 임베디드 시스템 테스팅 프레임워크 구성에 적용할 수 있다.

본 논문에서는 2장에서 다른 테스트 환경 등에 대한 관련 연구를 소개하고, 3장에서 제안하는 실시간 테스트 실행기인 Test Execution Machine(TEM)에 대해서 설명을 한다. 4장에서는 제안한 TEM이 실제로 테스트에 적합한지 증명한 실험 결과에 대해서 설명을 하고 마지막으로 5장에서 결론과 앞으로 진행되어야 하는 연구 방향에 대해서 설명한 후 논문을 마치도록 한다.

2. 관련 연구

자동차 엔진 컨트롤 유닛(engine control unit)을 새로 개발하거나 새로운 알고리즘을 개발하여 소프트웨어가 변경되었을 때 이를 테스트하고자 Hardware in the loop system (HILs)의 형태로 테스트하는 벤츠사의 트럭 엔진 시스템

(Mercedes-Benz truck engine system)이 있다.[9] HILs테스트를 위해서는 우선 실제 테스트에 사용할 부품과 시뮬레이션을 할 부품으로 나뉜다. FSM 컨트롤 유닛(FMS-control unit), PLD 컨트롤 유닛(PLD-control unit), 인젝션 펌프(injection pumps)는 실제 부품을 사용하였고 엔진, EPS, ABS, 기타 센서 부품 등은 실시간 컴퓨터 시스템에서 시뮬레이션 한다.

시뮬레이션의 결과 값과 컨트롤 패널(control panel)에서의 입력 값은 액추에이터 인터페이스와 센서 인터페이스에 I/O 모듈을 이용해서 전송된다. 액추에이터 인터페이스와 센서 인터페이스는 실시간 컴퓨터 시스템에서 받아온 결과 값을 신호로 생성하여 각각 알맞은 실제 부품에 전달한다. 이 시스템에서 테스트 실행기 부분은 실시간 컴퓨터 시스템이다.

논문 [9]에서는 실시간 컴퓨터 시스템을 구현하는 데 있어서 2가지 방법을 제안한다. 첫 번째 방법은 강력한 PC 파워를 제공하여 모든 모델을 병렬로 연산하는 것이다. 두 번째 방법은 DEC Alpha 프로세서와 디지털 신호 연산 처리가 가능한 dSPACE-시스템을 사용해 구현하였다. 이 경우 각자 자신의 연산만을 처리하므로 병렬 연산이 필요 없게 된다. 또한 MATLAB/SIMULINK의 모델들을 사용할 수 있는 장점이 있다.

이 두 가지 방법 모두 실시간 시스템을 직접 설계하는 방법이다. 이 경우 특정 테스트 대상에 알맞게 구성이 가능하다는 장점을 가지지만 다른 테스트 대상에 대해서 테스트 환경을 구성할 경우 최악의 경우 처음부터 다시 구현해야 한다는 단점을 가지고 있다.

논문 [10]에서 저자들은 전기 유압식 작동기 테스트 실행기(electro-hydraulic test machine(EHTM))의 설계 및 구현 방안을 제안하였다. EHTM의 목적은 전기 유압식 작동기(electro-hydraulic actuator(EHA))의 성능확인, 안정성, 강인성을 테스트하는 것이다. 이를 위해서 EHA 이외에 인터페이스 환경으로 힘 생성기(force generator)와 외란 생성기(disturbance generator)가 필요하다. PC 부분은 D/A 변환기(PCI 1720)를 통해서 힘 생성기와 외란 생성기에 입력 값을 주어서 각각의 힘을 생성하고 A/D 변환기(PCI 1711)를 통해서 힘 생성기에 걸리는 힘과 EHA에 걸리는 힘을 측정한다.

이 시스템에서 테스트 실행기에 해당하는 부분은 PC이다. PC에서 PCI(Peripheral Component Interconnect) 타입의 D/A 변환기를 이용해 힘 생성기와 외란 생성기를 제어하고 그에 따른 결과 값을 마찬가지로 PCI 타입의 A/D 변환기를 이용해 측정한다. 이때 힘 생성기와 외란 생성기를 제어 하는 신호는 MATLAB/SIMULINK에 포함된 실시간 윈도우 툴박스(Real-time Windows Target Toolbox of MATLAB)를 사용하여 생성한다. 실시간 윈도우 툴박스는 윈도우Windows 운영체제상에서 Simulink 모델을 실행시키는 실시간 엔진을 제공해준다.

이 방법을 사용해서 테스트 실행기를 구현할 경우 MATLAB/SIMULINK를 통해서 모델을 작성하면 여러 가지 테스트 대상에 적용가능하다는 장점을 가지고 있다. 하지만 MATLAB/SIMULINK를 통해서 모델을 작성하는 일에 많은 시간과 인적자원이 소모된다. 또한 실시간성을 확보하는 관점에서는 완벽하게 실시간성을 확보할 수 없는 상황이 발생한다. 실시간 윈도우 툴박스는 실시간성을 확보할 수 있지만 이를 구동시키는 윈도우 운영체제상에서 문제가 발생할 경우 실시간성의 확보가 어렵다는 단점이 있다.

연구[11]은 BLDC(Brush Less DC) 모터 제어기의 기본적인 기능과 고장모드를 평가하기 위한 HILS의 구현은 목표로 한다. BLDC 모터 제어기에 해당하는 MCU를 제외한 다른 요소들은 모두 HIL 에뮬레이션 장비에서 시뮬레이션을 통해서 각종 신호를 제공한다. HIL 에뮬레이션 장비에서 시뮬레이션되는 요소로는 MCU 구동에 필요한 위치 센서 신호 및 전류 센서 신호 이외에도 MCU의 PWM(Pulse Width Modulation) 및 오류(Fault) 신호를 받아들여 HIL 장비 내부 모델의 시뮬레이션을 가능하게 한다. MATLAB/SIMULINK로 작성된 시스템 및 전동 부품 모델들을 RTI(Real-Time Interface)를 통해 HIL 내부에서 동작할 수 있는 형태로 변경되며 HIL 내부에서 생성되는 신호는 컨트롤 데스트(ControlDesk)를 통해서 모니터링 및 제어가 가능하게 한다.

이 시스템에서 테스트 실행기에 해당하는 부분은 HIL이다. 컨트롤 데스트로부터 받은 입력을 입력 신호로 생성하여 MCU에 입력하고 MCU로부터 피드백받은 값을 이용해서 시뮬레이션을 실행하고 이 결과를 다시 MCU로 전송을 한다.

이 구성은 앞서 EHTM과 마찬가지로 MATLAB/SIMULINK를 통해서 모델을 작성하는 일에 많은 시간과 인적자원이 소모된다. 또한 BLDC 모터 제어기가 아닌 다른 테스트 대상에 적용하기 어렵다는 단점이 있다.

앞선 연구들에서 테스팅 프레임워크의 필수적인 요소인 재구성성에 관한 보고를 찾기는 매우 어렵다. 본 연구에서는 테스트 대상의 특성에 따라 재구성이 가능한 테스팅 프레임워크를 제안한다.

3. TEM(Test Execution Machine)

TEM은 테스트 실행기로서 기본 기능은 다른 실시간 테스트 실행기와 마찬가지로, 테스트 대상에게 입력 신호를 생성하여 주거나 대상으로부터 출력 신호를 수집하는 장치이다. 기본 기능인 신호의 생성과 수집뿐만 아니라 생성 및 수집한 모든 신호를 로그 파일로 저장, 테스트 스크립트 파일 실행, 컨트롤 유저 인터페이스(CUI: control user interface)로 데이터 전송 등의 추가적인 기능을 포함하고 있다. TEM이 다른 테스트 실행기들에 비해서 가지는 장점은 재구성성과 실시간성이다.

3.1 재구성 특성

TEM은 테스트 대상 및 목적에 필요한 설정에 따라서 입출력 신호를 선택적으로 사용할 수 있고 해당 신호 특성에 맞게 설정이 변경이 가능하다. 이는 재구성 컴퓨팅(reconfiguration computing)과 유사한 개념이다.[8]

테스트 대상 시스템에 따라 입출력 장치가 다를 것이다. 입출력 장치가 다를 경우, 해당하는 입출력을 담당하는 하드웨어와 소프트웨어(드라이버)만을 등록함으로써 테스트 대상 시스템에 맞게 TEM을 재구성할 수 있다. 이러한 재구성성은 실제 임베디드 테스트 환경 구성에서 필요한 노력 및 시간을 크게 줄일 수 있을 것 이다.

1) 하드웨어 구성

TEM을 구성하는 하드웨어는 크게 중심부인 PC와 신호처리를 담당하는 National Instrument(NI)사의 데이터 수집(Data AcQuisition) 카드(NI DAQ 카드), 두 부분으로 분류가 가능하다.

PC 부분은 CPU, RAM과 같은 일반적인 컴퓨터 자원으로 전체적인 성능에 영향을 주는 부분이다. 일반적인 PC 부품을 사용하지만 몇 가지 제약사항이 있다. LabVIEW의 실시간 운영체제인 “Real-time Module”을 사용하고 있는데 PC 부품 중 해당 운영체제가 지원하지 못하는 구성요소들이 있다. 따라서 LabVIEW “Real-time Module”이 지원 가능한 부품들을 사용해야 한다. Table 1은 구현한 TEM이 사용하고 있는 주요 구성품들이다.

Table 1. TEM Main PC parts

Part	Model
CPU	Intel Core i7 (Sandy Bridge architecture)
RAM	Samsung 2GB PC3-10600U-09-11-A1
Hard Driver	WD 1.0TB WD-10BALX
Ethernet Board	Intel PRO/100+ Adapter
Mother Board	GIGABYTE GA-PA65-UD3-B3

NI 카드는 아날로그 신호, 디지털 신호 등 여러 신호 생성 및 수집과 CAN(Control Area Network)과 같은 각종 통신에 필요한 PCI 타입의 카드들이다. 마더 보드에서 지원하는 PCI 슬롯의 개수만큼 카드를 사용할 수 있고 필요에 따라서 PCI 확장 카드를 사용해서 여러 개의 카드를 사용가능하며 테스트하는 테스트 대상에 따라서 그 구성을 달리할 수 있다. 확장 시에 해당 드라이버를 등록함으로써 TEM을 재구성할 수 있다.

이를 통해서 다양한 테스트 대상에 대해 대응할 수 있다. 간단한 예시로 기존의 테스트 대상인 A는 아날로그 신호 8개를 사용하여 NI-DAQ 카드를 아날로그 포트 8개짜리를 사용하고 있었다. 하지만 새로운 모델인 B를 테스트하려고 할 때 입출력이 추가되어 아날로그 신호 10개를 사용한다면 소프트웨어나 특별한 추가 작업 없이 NI-DAQ 카드 중 아날로그 포트가 10개 이상인 모델로 교환하는 방법으로 손쉽게 다시 구성할 수 있다. 또는 같은 NI-DAQ 카드를 하나 더 추가함으로써 I/O 변경 및 개수 확대 등, 하드웨어 변경에 관한 문제를 해결할 수 있다.

2) TEM 소프트웨어 구성

TEM은 주 제어부을 제외하고는 모든 동작에 대해서 오브젝트 단위로 수행을 한다. TEM의 소프트웨어 구성은 기본 틀이 되는 레지던트 오브젝트(resident object)와 선택적으로 사용되는 재구성 가능 오브젝트로 구분된다.[12]

레지던트 오브젝트는 TEM의 기능 중 환경 설정에 필요한 사용자 인터페이스를 제공하는 CUI(Control User Interface)와 통신을 담당하는 기능, 로그 파일을 저장하는 기능과 같은 실제 테스트 실행에 직접적인 영향을 주지 않는 편의 기능들을 담당하는 오브젝트를 의미한다. 리포터(Reporter) / 커맨더(Commander) / 로거(Logger)의 총 3가지 레지던트 오브젝트가 존재한다. 이 레지던트 오브젝트들은 TEM을 실행 시킬 때 반드시 하나씩 존재하도록 실행 준비파일에 기술해주어야 한다. 리포터, 커맨더의 경우에는 CUI와 통신을 통해 받은 명령어를 처리하거나 메시지를 생성하는 역할을 하므로 이 둘을 기술하지 않을 경우 CUI와의 통신이 제대로 이루어지지 않는다. 또한 로거는 테스트가 실행되는 동안 생성하거나 수집한 신호들의 정보를 로그 파일로 저장하는 역할을 한다. 이 로그 파일이 없는 경우 테스트가 진행되더라도 결과를 확인하기 어렵다.

재구성 가능 오브젝트는 실제 테스트 수행에 대한 역할을 담당한다. 대표적인 기능으로는 신호의 생성과 수집이 있다. 이 재구성 가능 오브젝트들은 실행 준비파일에 기술을 하지 않더라도 TEM이 실행되는 데 오류가 발생하지 않는다. 하지만 테스트 대상에 알맞은 재구성 가능 오브젝트들을 선택하여 구성해야 한다.

Fig. 1은 TEM의 전체적인 소프트웨어 구조를 보여주고 있다. 구조는 크게 3가지로 구분된다. 첫 번째는 네트워크 인터페이스(Network Interface)이다. 이 부분은 실질적으로 CUI와 메시지를 주고받는 역할을 하는 부분이다. 두 번째는 드라이버 인터페이스(Driver Interface)이다. 이 부분은 재구성 가능 오브젝트들로 이루어져 있고 NI 카드들을 제어하여 신호들의 생성 및 수집을 담당하는 부분이다. 마지막으로 나머지 부분은 레지던트 오브젝트와 재구성 가능 오브젝트가 같이 구성되어 있다. 로그 파일을 생성하거나 CUI와 주고받은 메시지를 생성하거나 테스트 스크립트 파일의 정보를 스

케줄링(Scheduling)을 하는 일과 같은 역할을 수행한다.

a) 상주 오브젝트 (Resident Object)

상주 오브젝트는 TEM이 실행함에 있어서 기본적인 역할을 담당한다. 이는 CUI와 통신을 담당하는 기능, 로그 파일을 저장하는 기능과 같은 실제 테스트 실행에 직접적인 영향을 주지 않는 편의 기능과 같이 테스트를 보좌하는 TEM의 기본 동작을 담당한다.

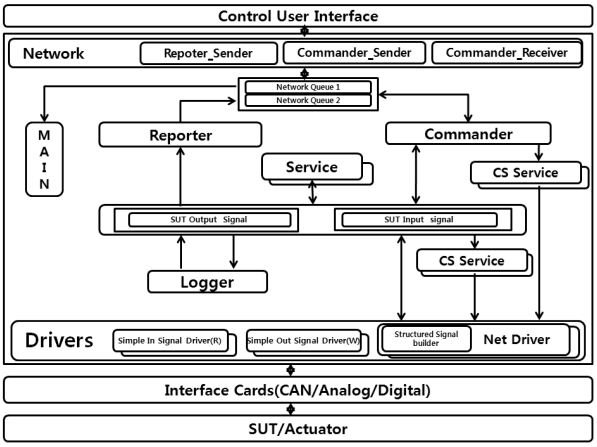

Fig. 1. Architecture of TEM

리포터와 커맨더의 경우에는 특별히 실행 준비파일에 설정 값을 변경해 줄 필요가 없지만 로거는 실행하는 테스트 대상의 입출력 신호의 특성에 맞게 기록 주기, 변수명 등을 설정 해주어야 한다. 다음 Table. 2는 상주 오브젝트들의 기본 역할을 서술해 놓은 것이다.

Table 2. Resident Object information

Object name	Function
Reporter	Transmit the variable to CUI
Commander	Receive the command from CUI
Logger	Create log file for variable

리포터의 경우에는 CUI로 전송하는 변수들을 선택하여 메시지로 변환하여 네트워크 큐에 저장하는 역할을 한다. 테스트 대상의 입출력으로 사용되는 모든 변수의 값 변화는 모두 로그 파일에 저장을 하지만 테스트가 완료되어야 확인할 수 있는 단점이 있다. 이를 보완하기 위해 테스트를 진행하는 엔지니어가 관심 있게 보고자 하는 정보만을 선택하여 CUI에서 간단하게 확인할 수 있게 하였다. 즉, 리포터에서 선택하는 변수들이 엔지니어가 보고자하는 정보들이 된다.

커맨더는 CUI로부터 입력받은 명령을 처리하는 역할을 한다. 테스트 시작, 정지와 같은 기본적인 명령은 주 제어부에서 처리하지만 테스트 중에 임의로 입력하고자 할 때 CUI에서 입력하는 기능과 같은 명령은 커맨더에서 수행한다.

로거는 TEM이 테스트 대상과 주고받는 모든 변수의 값을 로그 파일로 기록하는 역할을 한다. 이 로그 파일을 분석해서 테스트 결과를 도출한다.

b) 재구성 가능한 오브젝트(Configurable Object)

재구성 가능 오브젝트는 TEM이 테스트를 수행 및 신호의 생성과 수집을 담당한다. 예를 들어 테스트 대상에 입력을 전송하거나 마찬가지로 테스트 대상으로부터 출력을 읽어 들이거나 테스트 스크립트 파일을 읽어서 처리하는 것과 같은 테스트에 해당하는 기능들을 담당하는 오브젝트이다.

재구성 가능 오브젝트는 여러 가지 테스트 대상에 맞게 테스트 실행기를 구성함에 있어서 핵심적인 요소이다. 만약 재구성 가능 오브젝트를 하나도 생성하지 않는다면 TEM의 기본 동작에는 영향을 주지 않지만 원하는 테스트는 실행이 되지 않는다. 그러므로 테스트 대상에 알맞은 재구성 가능 오브젝트들을 선택하여 사용해야 한다. 또한 각 재구성 가능 오브젝트는 표준화 하여 모두 같은 구조를 가진다. 레지던트 및 재구성 가능 오브젝트는 주제어부에서 관리한다. 이때 사용할 수 있는 모든 오브젝트를 전부 등록시키지 않는다. 실행조건 파일에 기록되어 있는 오브젝트만을 선택적으로 등록시킨다. 실행조건 파일에 대한 내용은 4. TEM Configuration file 부분에서 다루도록 하겠다.

예를 들어, 테스트 대상 시스템이 입력으로 아날로그 신호 5개, 디지털 신호 2개, CAN 통신을 사용한다면, 실행조건 파일에 Voltage_OUT 5개, Digital_OUT 2개, CAN_Writer 1개, CAN_Reader 1개의 재구성 가능 오브젝트를 등록한다.

또한 이 시스템이 출력으로 아날로그 신호 2개, 디지털 신호 1개를 사용한다면 TEM이 이들 정보를 수집하기 위해서 추가적으로 Voltage_OUT 2, Digital_OUT 1개의 재구성 가능 오브젝트를 등록한다.

이처럼 테스트 대상의 입출력 신호의 개수에 맞게 오브젝트를 생성하면 원하는 개수만큼의 신호를 생성하는 테스트 실행기를 구성할 수 있다.

또한 사용 중인 NI 카드가 생성 및 수집 가능한 신호의 개수보다 많은 수의 입출력을 사용하는 경우에는 해당 NI 카드를 더 많은 신호가 처리 가능한 카드로 교체를 하거나 또 다른 NI 카드를 추가적으로 사용하면 손쉽게 해결할 수 있다. 다른 카드로 변경하는 경우에는 변경한 카드가 현재 재구성 가능 오브젝트에서 사용하는 라이브러리를 지원하는지 확인 후 필요에 따라서 몇 가지 설정 값만 수정하면 사용할 수 있다. 그리고 같은 카드를 추가로 사용하는 경우에는 이미 만들어 놓은 재구성 가능 오브젝트를 수정할 필요가 없이 바로 사용이 가능하다.

현재 TEM에서 사용 중인 재구성 가능 오브젝트는 다음 Table 3과 같다.

Table 3. Configurable Object information

Object name	Function
Service	Set the value for each variable
CAN_Writer	Write the CAN message
CAN_Reader	Read the CAN message
Voltage_IN	Read the analog signal
Voltage_OUT	Write the analog signal
Digital_IN	Read the digital signal
Digital_OUT	Write the sigital signal
SSB (Structured Signal Builder)	Create CAN message frame. Or analysis CAN message frame

Service는 TEM 내부의 변수 값을 변경하는 역할을 한다. 예를 들어 테스트 스크립트 파일을 입력받아 테스트하는 경우 테스트 스크립트 파일 안의 정보를 알맞은 변수에 저장하는 역할을 한다. 변수의 값을 어느 시점에 변경하느냐에 따라서 전체적인 테스트의 수행을 조절할 수 있다. 따라서 Service는 스케줄링 오브젝트라 할 수 있다.

CAN Writer / Reader는 CAN 통신을 사용해서 테스트 대상과 통신 주고받는 역할을 한다.

Voltage IN / OUT은 아날로그 신호를 수집 / 생성의 역할을 한다.

Digital IN / OUT은 Voltage IN / OUT과 유사하게 디지털 신호를 수집 / 생성의 역할을 한다.

마지막으로 SSB는 CAN 통신에 사용하는 CAN 프레임을 해석 및 생성하는 역할을 수행한다. CAN 통신의 경우 데이터 프레임 8bytes를 이용하여 각기 다른 방식으로 정보를 표현한다. 그렇기 때문에 해당 테스트 대상에 맞게 SSB를 생성해주어야 한다. SSB의 경우에는 그 구조 및 내용이 간단하기 때문에 필요에 따라 간단하게 구현하여 사용할 수 있다.

3) 재구성 가능 오브젝트 생성

재구성 가능 오브젝트의 구현은 DLL 파일로 되어 있다. DLL 파일의 내부는 Table 4와 같이 3가지로 나뉜다. 첫째가 Gen 함수이고 둘째가 Func 함수 마지막으로 TERM 함수가 있다.

Table 4. Configuration Object Function structure

Function type	Function
Gen	Create of structure or memory allocation
Func	Generate of signal
TERM	Terminate of structure or memory allocation

Gen 함수는 해당 오브젝트를 생성할 때 필요한 정보를 실행조건 파일에서 읽어와 저장하는 역할과 해당 오브젝트가 사용하는 하드웨어를 초기화하는 역할을 한다. 따라서 실행조건 파일에는 해당 오브젝트의 동작특성과 관련된 변수와 사용하는 하드웨어 설정 값과 사용할 변수에 대해서 기술되어 있다. 이 실행조건 파일의 각 변수의 값을 어떻게 설정하느냐에 따라서 수행되는 기능이 달라진다. 또한 필요한 변수들을 동적 할당을 통해서 생성한다. Gen 함수의 흐름은 다음과 같다.

① CFG (Configuration)초기화
② CFG내의 실행 조건 정보 획득
③ NI 카드 드라이버 설정

Gen 함수에서 제일 먼저 수행하는 기능은 실행조건 파일의 정보를 저장할 구조체를 초기화하는 작업이다. 구조체의 초기화를 마친 뒤 실행조건 파일의 정보를 읽어와 구조체에 저장을 한다.

마지막으로 NI 카드의 드라이버를 설정한다. 이 단계는 필요에 따라서 NI카드 사용 여부에 따라 하지 않을 수 있다. 신호의 생성 및 수집의 기능을 담당하는 재구성 가능 오브젝트는 NI 카드를 이용한다. 따라서 사용할 NI 카드에 대한 설정을 이 단계에서 진행한다. NI 카드를 사용하지 않는 재구성 가능 오브젝트의 경우에는 이 단계를 생략할 수 있다.

두 번째로 메인 함수이다. 이 부분에서는 실제적으로 입출력 신호를 생성하거나 수집하는 기능에 대해서 구현되어 있다. 메인 함수는 주기적으로 반복해서 실행이 된다. 이는 CUI로부터 테스트 시작 신호가 입력되면 실행을 시작해서 테스트가 종료되거나 테스트 정지 신호가 입력될 때까지 실행된다. 예를 들어 Voltage_OUT의 경우에는 메인 함수가 주기에 한 번씩 실행될 때마다 아날로그 전압 신호를 생성하여 출력하는 함수를 실행한다.

마지막 부분은 TERM 함수 부분이다. 이 부분은 앞서 Gen 함수에서 동적 할당으로 생성한 메모리를 해제하고 하드웨어 설정을 초기화 하는 역할을 한다. TERM 함수의 흐름은 다음과 같다.

① 메모리 해제
② CFG 제거

TERM 함수는 두 단계에 거쳐서 수행이 된다. 먼저 앞서 Gen 함수에서 실행준비 파일 및 기타 정보를 저장하기 위해서 동적할당을 통해서 설정한 메모리를 해제한다. 그 뒤 마찬가지로 Gen 함수에서 실행준비 파일의 정보를 저장했던 구조체를 제거한다.

기존의 소프트웨어를 수정하지 않고 새로운 재구성 가능 오브젝트를 사용하기 위해서, 해당 오브젝트(xxx)를 위한

설정파일을 읽어서 dll을 생성하는데 사용하는 xxx_Gen, 해당 쓰레드(thread)를 등록하기 위한 xxx_Pool, 타이머를 등록하기 위한 xxx_Timer, 단순 함수를 호출할 때 사용하는 xxx_Func, dll 제거에 사용하는 xxx_Term 함수 등을 사용된다.

4) TEM 구성 파일(TEM Configuration File)

TEM Configuration File은 실행조건 파일로서 TEM이 동작함에 있어서 전체적으로 필요한 정보들을 기술해 놓은 파일이다. 이를 수정함으로써 테스트 대상에 맞게 원하는 TEM을 구성할 수 있다. 파일에 기술된 정보들은 Object, Variable, Processor allocation 항목으로 구분된다. 이 정보들은 SYSTEM.cfg 파일에 기술되어 있다.

Object는 사용할 오브젝트들의 기본정보이고 Variable은 TEM에서 사용하는 변수들의 정보이다. 마지막으로 Processor allocation은 TEM이 가지고 있는 프로세서들이 시스템 그룹과 비시스템 그룹 중 어느 그룹에 속하는지 기술되어 있다. 여기서 시스템 그룹이란 운영체제가 실행 중인 프로세서가 포함된 프로세서들의 집합이다. 비시스템 그룹은 반대로 시스템 그룹에 속하지 않는 프로세서들의 집합이다. 모든 프로세서는 기본적으로 시스템 그룹에 속한다. 하지만 설정을 통해서 0번 프로세서를 제외한 다른 프로세서들을 비시스템 그룹으로 만들 수 있다. 시스템 그룹의 프로세서들은 운영체제에 의해 자동으로 일을 할당받지만 비시스템 그룹은 원하는 오브젝트를 할당할 수 있다. 이를 통해서 부하가 많이 걸리는 오브젝트나 중요한 기능을 담당하는 오브젝트를 하나의 프로세서에 전담으로 처리하도록 할 수 있다.

첫 번째 항목인 Object는 각 오브젝트들의 기본 정보가 기술되어 있다. TEM이 테스트를 실행해서 사용할 오브젝트들에 대한 기본 정보를 의미한다. 각 오브젝트를 선택함으로서 TEM의 기본 기능을 제외한 입출력에 관한 모든 사항을 제어할 수 있다. 세부적인 정보는 다음 Table 5와 같다.

Table 5. Object information

Item	Meaning
Name	Unique object name
CFG_Path	File path of configuration file
DLL_Path	File path of dll file
GEN_Func	Name of the generation function
TERM_Func	Name of the terminate function
Processor_group	Means the number of the processor group to be that the object is included

Name은 각 오브젝트의 고유한 이름을 의미한다. 함수로 동작하는 오브젝트의 경우 이름을 통해서 어떠한 함수를 선택할지를 정하기 때문에 반드시 각 오브젝트마다 고유의 이름을 정의해주어야 한다.

CFG Path와 DLL Path는 각 오브젝트가 사용하는 실행조건 파일과 DLL 파일이 저장되어 있는 위치를 의미한다. 기본적으로 TEM 내부에 각 파일이 저장되어 있는 폴더가 정해져 있으므로 이를 맞게 기입해야 한다.

GEN Func과 TERM Func은 DLL 내부에 정의되어 있는 함수의 이름을 의미한다. 함수의 이름은 각각 DLLName_gen / DLLName_TERM의 규칙으로 생성된다. 그러므로 이를 맞게 기입해주어야 한다.

Processor group은 해당 오브젝트가 어떠한 프로세서 그룹에서 동작할지 정해주는 것을 의미한다. 비동기 타이머나 함수로 동작하는 오브젝트의 경우에는 의미 없지만 쓰레드 풀로 동작하는 오브젝트의 경우에는 반드시 비시스템 그룹에 속하는 프로세서의 숫자중 하나를 선택해서 입력해야 한다. 시스템 그룹과 비스템 그룹은 NI에서 제공하는 라이브러리에서 정의하는 프로세서 그룹이다. 시스템 그룹은 모든 인터럽트를 처리하는 0번 코어가 포함된 프로세서 그룹으로 주 제어부와 함수와 비동기 타이머로 동작하는 오브젝트가 자동으로 할당되는 프로세서 그룹이다. 그 이외의 프로세서들을 비시스템 그룹이라 한다. 쓰레드 풀로 동작하는 오브젝트들은 비시스템 그룹에 포함된 코어 중 원하는 코어에 수동으로 할당할 수 있다.

두 번째 항목인 Variable은 전체 시스템에서 사용할 변수들에 대한 정보가 기술되어 있다. TEM에서 사용하는 모든 변수들은 이곳에서 먼저 정의를 해야 테스트를 진행 시 사용할 수 있다. 세부적인 정보는 다음 Table 6과 같다.

Table 6. Variable information

Item	Meaning
Variable_ID	Unique identification number for variable
Direction	Flag for Input / Output

Variable ID는 TEM 내부에서 사용되는 변수가 가지는 고유한 ID이다. 이를 통해서 각 오브젝트는 자신이 사용하는 변수를 찾는다. 이는 SymbolDB.csv 파일에 있는 정보와 일치 시켜주어야 한다. 여기서 SymbolDB.csv 파일은 각 변수에 대한 정보가 정의되어 있는 파일이다.

Direction은 해당 변수의 입력 / 출력 여부를 나타내준다. 테스트 대상의 입력으로 사용될 경우 1, 테스트 대상의 출력으로 사용될 경우 2의 값을 기입한다.

세 번째 항목인 Processor allocation은 TEM에서 존재하는 코어들의 정보가 기술되어 있다. 각 코어들을 시스템 그룹과 비시스템 그룹으로 구분하여 정의 한다. 세부적인 정보는 다음 Table 7과 같다.

Table 7. Processor allocation information

Item	Meaning
Core	TEM's processor number
Processor_group	Define a group of processors that the processor is assigned. However, processor 0, you must belong to the system group always

Core는 TEM의 프로세서 넘버를 의미한다. 이는 정해주는 것이 아니라 현재 TEM에서 사용하는 CPU에 따라서 정해진다. 현재 사용하고 있는 CPU의 프로세서 개수만큼 정의하면 된다.

프로세서 그룹은 해당 프로세서가 할당되는 프로세서 그룹을 의미한다. 시스템 그룹으로 사용할 경우 0, 비시스템 그룹으로 사용할 경우 1을 기입하면 된다. 단 0번 프로세서는 항상 시스템 그룹에 속해야 한다. 여기서 시스템 그룹이란 메인 함수가 실행되는 프로세서들의 그룹이다. 시스템 그룹 안의 프로세서를 이용해서 메인 함수를 비롯해 비동기 타이머로 동작하는 쓰레드들을 실행한다. 그리고 쓰레드 풀로 동작하는 오브젝트의 경우에는 비시스템 그룹에 해당하는 프로세서들을 사용한다.

3.2 실시간 특성

테스트 실행기가 가져야할 성능 중, 가장 기본적인 것 중 하나는 얼마나 정확한 입력 신호 값을 원하는 시점에 입력하는 점과 테스트 대상의 출력 값을 얼마나 정확히 원하는 시점에 읽어올 수 있는 점이다.

정확한 입력 신호 값과 출력 값을 처리하는 것은 충분한 분해능과 샘플링 주파수를 가지는 D/A 변환기, A/D 변환기 장비를 구비한다면 쉽게 해결할 수 있는 문제이다. 또한 원하는 시점에 입력 값을 입력하거나 출력 값을 읽어오는 것은 테스트 실행기가 실시간성을 가진다면 만족할 수 있는 문제이다.

간단한 예를 들면 입력 신호로 A를 입력하였을 때 출력 신호의 값이 B로 바뀐 후 50ms 후에 C로 바뀌는 기능을 가진 시스템이 있을 때 입력 신호 A를 입력하고 나서 50ms 이내로 출력 신호의 값을 읽고 50ms 이후에 출력 신호의 값을 읽어서 B에서 C로 값이 바뀌는 것을 확인해야만 해당 기능이 정상적으로 동작했음을 테스트를 통해서 확인했다 할 수 있다. 하지만 실시간성을 만족 못 한다면 테스트를 통해 얻은 값이 실제 테스트 대상이 출력하는 출력 신호의 값과 다를 수가 있다.

이러한 실시간성을 확보하는 방법으로는 실시간 운영체제 기반으로 테스트 환경을 구성하는 방법과 실제로 실시간 시스템을 직접 설계하는 방법이 있다. 이 중 실제로 실시간 시스템을 직접 설계하는 방법은 원하는 테스트에 알맞게 구성할 수 있다는 장점을 가지고 있지만 실제 개발 시간이 너무 오래 걸릴 수 있다는 단점이 있다. 반면에 이미 상용화된 실시간 운영체제를 기반으로 테스트 환경을 구성한다면 직접 설계하는 방법보다 개발 시간을 단축시킬 수 있다는 장점이 있다. 하지만, 직접 실시간 시스템을 설계하는 방법에 비해서 시스템이 무거워질 수 있는 단점이 있다.

TEM은 두 번째 방법인 상용화 된 실시간 운영체제를 기반으로 테스트 환경을 구성하는 방법을 사용한다. NI사가 제공하는 LabVIEW Real-time module을 실시간 운영체제로 사용하였다. 이는 NI사에서 Pharlap사의 ETS라는 실시간 운영체제를 NI사에서 판매하는 여러 장비들과 호환시키기 위해서 재포팅하여 배포한 것이다. 따라서 LabVIEW Real-time module를 사용할 경우 NI사에서 판매하는 여러 장비들을 다른 추가 작업 없이 제공해주는 라이브러리만을 사용하여 구동시킬 수 있다는 장점이 있다. 또한 다른 하드웨어가 필요 없이 일반적인 PC부품을 사용하여 구성할 수 있어 확장성이 좋다는 장점이 있다.

4. 실 험

TEM의 실시간성과 재구성성을 검증하기 위해서 두 개의 테스트 대상에 대해서 각각 테스트 환경을 구축하여 간단한 테스트를 수행해 보았다. 첫 번째 실험대상은 간단한 BLDC 모터를 구동하는 BLDC 모터 제어기 이다. 그리고 두 번째 실험대상은 실제 차량에 사용되는 운전석 백미러(Driver Door Module : DDM) 제어기이다. 실험의 목적이 TEM에 대한 검증이므로 테스트 케이스와 같은 항목에 대한 내용은 생략하고 테스트 환경을 중점으로 설명한다.

4.1 실험1 - BLDC 모터 제어기

실험에 사용한 BLDC 모터 제어기는 최대 200W BLDC 모터의 속도 제어에 사용되는 모델이다. 사용하는 입력은 2개로 전원에 해당하는 아날로그 신호와 제어 명령에 해당하는 CAN 통신이다. 출력도 마찬가지로 2개이다. 제어 명령에 대한 응답인 CAN 통신과 BLDC 모터의 홀 센서(Hall sensor) 신호이다. 입출력 정보가 Table 8에 정리되어 있다.

Table 8. BLDC Motor Controller Input / Output

Part	Signal	Number
Input	Analog	1
	CAN	1
Output	Digital	1
	CAN	1

이러한 입출력에 맞게 TEM에서 사용한 NI 카드는 2개이다. 하나는 아날로그 신호의 생성 및 수집과 디지털 신호의 생성 및 수집이 가능한 NI-PCI-6733이다. 다른 하나는 CAN 통신에 필요한 NI-PCI-CAN/XS2이다.

TEM이 이러한 카드들을 구동하기 위해서 선택한 재구성 가능 오브젝트는 총 6개이다. 우선 테스트 스크립트 파일을 이용해 테스트를 진행하기 위해서 이를 처리할 Service 오브젝트와 입출력에 관여하는 Digital_IN, CAN_Writer, CAN_Reader, SSB 2개가 사용된다. SSB의 경우에는 CAN_Writer, CAN_Reader가 각각 하나씩 사용하기 때문에 2개가 필요하다. Table 9에 오브젝트 정보가 정리되어 있다.

Table 9. BLDC Motor Controller TEM Objects

TEM Object	Number
Service	1
Digital_IN	1
CAN_Writer	1
CAN_Reader	1
SSB	2

인터페이스 환경으로는 BLDC 모터의 홀 센서의 신호에 잡음이 많이 검출되어 이를 제거하기 위한 필터를 연결해 주었다. 그리고 BLDC 모터 제어기의 전원을 제공할 SMPS 구성하였다. 다음 Fig. 2와 Fig. 3은 구성도와 실제 구성 모습이다.

TEM에서 사용하는 NI-PCI-CAN/XS2의 CAN 포트는 테스트 대상인 BLDC 모터 제어기에 직접 연결하였다. 또한 홀 센서의 신호의 수집을 담당할 NI-PCI-6733은 잡음을 제거할 필터에 연결되어 있다.

인터페이스 환경에는 BLDC 모터 제어기의 전원으로 사용될 SMPS는 BLDC 모터 제어기에 직접 연결하였다. 잡음을 제거할 필터 회로는 TEM과 BLDC 모터의 홀 센서에 연결되어 있다. 테스트 대상인 BLDC 모터와 제어기는 서로 전원과 제어 신호를 전달할 수 있게 연결되어 있다.

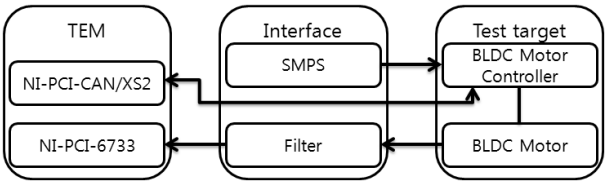

Fig. 2. BLDC motor controller test configuration

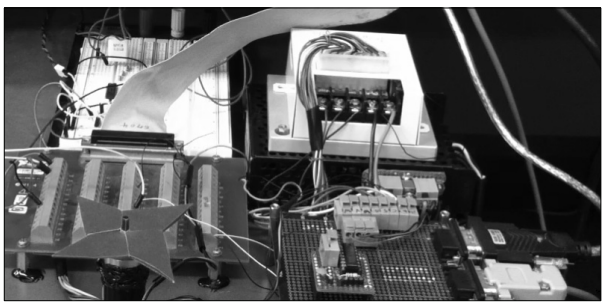

Fig. 3. Actual configuration of the BLDC motor controller test

이러한 테스트를 통해서 BLDC 모터 제어기의 설정 값인 가감속 딜레이와 데드 벤드의 값을 변경할 때 BLDC 모터 제어기에 이상이 발생해 BLDC 모터가 순간 정지했다가 다시 회전하는 것을 확인할 수 있었다.

4.2 실험2 - Driver Door Module

실험에 사용한 Driver Door Module(DDM)은 차량의 운전수 측 차문에 있는 모듈로 각종 스위치 입력을 받아 차문의 창문 제어, 사이드 미러 제어 등의 여러 기능을 담당하는 모듈이다. 이 중에서 사이드 미러 제어에 관련된 기능을 테스트 한다. 사용하는 입력은 스위치 입력 10개와 전원에 해당하는 아날로그 신호 1개이다. 출력은 CAN 통신으로 출력되는 상태정보와 사이드 미러를 조절하는 모터로 입력되는 아날로그 신호 3개이다. Table 10은 그 입출력 정보이다.

Table 10. Driver Door Module Input / Output

Part	Signal	Number
Input	Digital	10
	Analog	1
Output	Analog	3
	CAN	1

이러한 입출력에 맞게 TEM에서 사용한 NI 카드는 3개이다. 기존 실험 1에서 사용한 NI 카드들로는 디지털 신호를 처리하기에 NI-PCI-6733의 디지털 port의 수가 8개로 수가 부족하다. 그래서 NI-PCI-MIO-16E-4를 추가로 사용하였다.

TEM이 이러한 카드들을 구동하기 위해서 선택한 재구성 가능 오브젝트는 총 15개이다. 앞서 실험 1과 마찬가지로 테스트 스크립트 파일을 이용해 테스트를 진행하기 위해서 이를 처리할 Service 오브젝트가 우선 필요하다. 또한 입출력에 관여하는 Voltage_IN 3개, Digital_OUT 8개, CAN_Reader, SSB가 사용된다. 각 Voltage_IN과 Digital_OUT은 신호의 개수만큼 필요하다. 실험 1과는 다른 점은 우선 신호의 개수가 증가해 아날로그 신호와 디지털 신호를 처리하는 오브젝트의 수를 증가시킨 점과 CAN 통신을 통해서 입력이 없기 때문에 CAN_Writer와 SSB 한 개를 사용하지 않았다. Table 11에 오브젝트 정보를 정리하였다.

인터페이스 환경으로는 사이드 미러를 움직이는 DC모터의 입력 전압이 12V로 NI 카드로 읽을 수 있는 최대 전압인 10V를 넘기 때문에 이를 1/2로 전압 강하를 해주는 회로와 스위치 입력을 나타내는 디지털 신호를 DDM의 회로에 연결하기 위해서 필요한 AND, OR 게이트로 이루어진 논리회로를 연결해 주었다. 그리고 DDM의 전원을 제공할 차량용 배터리를 구성하였다. Fig. 4와 Fig. 5는 구성도와 실제 구성 모습이다.

Table 11. Driver Door Module TEM Objects

TEM Object	Number
Service	1
Voltage_IN	3
Digital_OUT	8
CAN_Reader	1
SSB	1

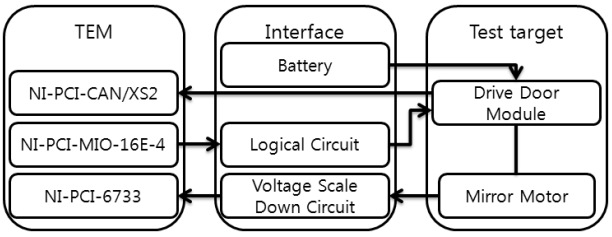

Fig. 4. Configuration of the driver's window control test

TEM에서 사용하는 NI-PCI-CAN/XS2의 CAN 포트는 테스트 대상인 DDM에 직접 연결하였다. 또한 스위치 입력을 담당할 NI-PCI-6733은 스위치 입력을 DDM의 회로에 연결하기 위한 논리 회로에 연결되어 있다. 사이드 미러의 DC 모터의 입력 전압을 측정할 NI-PCI-6733은 전압 강하 회로에 연결되어 있다.

인터페이스 환경에는 DDM의 전원으로 사용될 차량용 배터리는 DDM에 직접 연결하였다. 논리 회로와 전압 강하 회로는 각각 TEM과 테스트 대상에 연결되어 있다. 테스트 대상인 DDM과 사이드 미러의 DC 모터는 서로 전원과 제어 신호를 전달할 수 있게 연결되어 있다.

Fig. 5. Actual configuration of the driver's window control test

이 테스트를 통해서 DDM의 사이드 미러를 제어하는 기능을 확인할 수 있었다. 정상적인 입력뿐만 아니라 비정상적인 입력에도 별다른 오류 없이 정상적으로 동작하였다.

4.3 실험 결과

BLDC 모터 제어기와 Driver Door Module(DDM)를 테스트 대상으로 하여 각각 테스트 환경을 꾸며 간단한 테스트 한 실험 두 가지를 통해 TEM의 재구성성과 실시간성을 확인할 수 있다.

본 논문에서 제안하는 재구성이 가능한 환경에서는 실험 1를 수행하고 실험 2를 수행할 경우, 실험 2에서 필요한 추가로 필요한 입출력 장치 드라이버를 위한 오브젝트만을 등록하고 TEM의 다른 부분은 수정없이 테스트 환경 구성이 가능하다. 하지만, 제구성이 가능하지 않는 환경에서는 테스트 엔진의 일부를 수정하여야 할 것이다. 예를 들면, 추가되는 입출력을 위한 엔진 수정 및 인터페이스 수정 및 추가, 데이터 복사 및 전달을 위한 루틴 수정 및 추가, 이들을 위한 태스크 관리 등을 위해 엔진 자체의 수정이 필요할 것이다. 이는 엔진의 관리 측면 및 새로운 테스트 환경 구성에 큰 부담을 줄 것이다.

Fig. 6과 Fig. 7은 BLDC모터 제어기와 운전석 창문 제어기에 대한 테스트 결과 실측 데이터를 그린 그림이다.

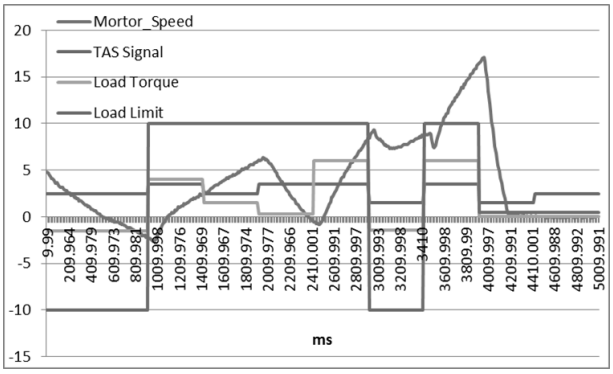

Fig. 6. For example, BLDC motor test results data

입력 신호 중 아날로그 모터 속도(Motor_Speed)와 입력 신호 중 CAN 신호인 핸들 속도(TAS Signal)에 따라, 디지털 출력인 핸들 부하 토크(Load Torque)와 CAN출력인 부하 극한치(Load Limit)를 생성하는 것을 보여준다. 가로축은 테스트 시간을(단위 msec) 나타낸다. 이 결과는 테스트 대상인 BLDC모터의 스펙과 일치하여 해당 테스트 결과는 정확하다고 판단되고 제안한 TEM으로 테스트 가능함을 보여주었다. (BLDC모터의 자세한 스펙은 본 논문의 범위가 아닌 관계로 기술하지 않는다.)

여러 기능 중 백미러를 열고 닫는 기능에 대한 시험 결과이다. 백미러를 열고 닫는 디지털 신호(Mirror_Unfolding 입력)대해 적절히 동작하는 아날로그 출력(Mirror_Unfolding 아날로그 입력)의 동작으로 통해 기능이 제대로 동작하고 있음을 알 수 있다.

위 두 실험을 통해 제안하는 시스템의 재구성성과, 실시

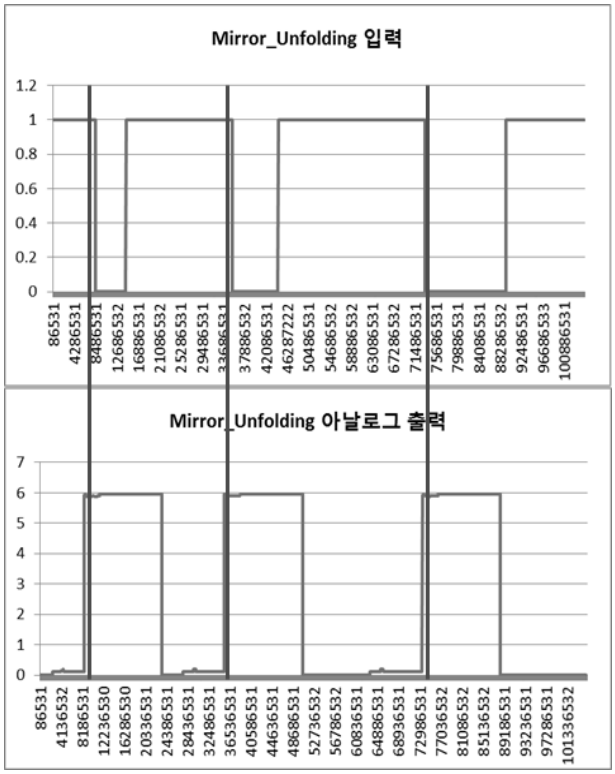

Fig. 7. For example, the driver's window control test result data

간성을 확인할 수 있었다. 첫 번째 재구성성은 서로 다른 두 테스트 대상에 대해서 TEM은 NI 카드를 추가 및 교체와 오브젝트 선택을 통해서 원하는 사양의 테스트 실행기를 구성할 수 있었다. 이 두 실험을 하기 위해서 각각 수행한 일은 각기 CAN 프로토콜이 달라 사용하는 SSB의 작성과 인터페이스 환경을 구축한 것이다. SSB의 작성은 정해진 형식 안에 간단한 코드 작성으로 통해서 이루어졌고 인터페이스 환경은 각 테스트 대상에 맞는 간단한 회로 구성을 통해서 이루어졌다. 이처럼 여러 테스트 대상에 대해서 매번 테스트 실행기를 제작할 필요 없이 TEM을 이용해서 대부분의 테스트 대상을 테스트할 수 있다.

두 번째 실시간성은 각 테스트의 로그 파일을 통해서 확인할 수 있다. 로그 파일에는 TEM이 생성하거나 수집한 모든 신호에 대한 정보가 기록되어 있다. 모든 정보는 해당 정보가 생성되거나 수집된 시간과 같이 기록이 된다. 이를 확인해본 결과 생성된 신호의 경우 테스트 스크립트 파일 안에 기술되어 있는 타임 스텝과 비교했을 때 최대 오차 2us였다. 또한 수집된 신호의 경우 설정한 신호 수집 주기와 비교했을 때 최대 오차 3us였다. 이를 통해서 실시간성을 만족하는 것을 확인하였다.

5. 결 론

본 논문에서는 재구성이 쉬운 실시간 임베디드 시스템 테스트 실행기를 제안하고 이를 구현하였다. BLDC 모터 제어기와 DDM이라는 서로 다른 기능 및 입출력을 가지는 테스트 대상에 대해서 각각 테스트 환경을 구성하고 실험을 통해서 구축한 테스트 실행기의 재구성성 및 실시간성을 검증하였다.

두 테스트 대상에 대해서 테스팅 프레임워크를 구성해본 결과 서로 다른 입출력에 대해서 실행준비 파일의 변경과 간단한 SSB 작성만으로 적용이 가능하다는 것을 보였다. 또한 테스트 결과인 로그 파일을 확인해본결과 두 테스팅 프레임워크 모두 실시간 특성이 확인되었다.

향후 연구로는 시스템 전체에 대한 테스트는 전체 시스템의 신뢰성 및 안정성은 확보할 수 있으나 에러를 발견했을 경우 어느 모듈에서 에러가 발생했는지 찾기가 매우 어렵고 각 모듈 간의 상호작용에 의해서 발생하는 에러 같은 경우는 발견하는 것이 어렵다. 따라서 추후 복수개의 테스트 대상을 유기적으로 테스트할 환경구축에 대한 연구가 필요하다.

Reference

[1] Ebert, Christof, and Capers Jones, "Embedded software: Facts, figures, and future," Computer 42.4, pp.42-52, 2009.

[2] J. Y. Seo, A. Y. Sung, B. J. Choi, S. B. Kang, "Automating Embedded software testing on an Emulated Target Board," Proc. of the Second International Workshop on Automation of Software test, pp.9-9, August, 2007.

[3] Youngsuk Jang, Gidae Yeo, Hyundong Lee, "Empirical Study for Manual vs. Automated Test of Embedded Software," Journal of Korean Institute of Information Technology, Vol.30, No.2, pp.343-345, 2003.

[4] Daeyoung Lee, "Software test, the "march" toward at higher," Wisewires, 2007.

[5] Sanggyun Hong, "Embedded SW test, the start of safety and reliability," National IT Industry Promotion Agency, 2008.

[6] Edwards, Stephen H., "A framework for practical, automated black? box testing of component?based software," Software Testing, Verification and Reliability, 11.2, pp.97-111, 2001.

[7] Whittaker, James A., "What is software testing? And why is it so hard?," Software, IEEE 17.1, pp.70-79, 2000.

[8] Compton, Katherine, and Scott Hauck, "Reconfigurable computing: a survey of systems and software," ACM Computing Surveys (csuR), 34.2, pp.171-210, 2002.

[9] Sermann, R., Schaffnit, J., Sinsel, S., "Hardware-in- the-loop simulation for the design and testing of engine- control systems," Control Engineering Practice 7.5, pp.643-653, December, 1999.

[10] AHN, Kyoung Kwan, DINH, Quang Truong, “Self-tuning of quantitative feedback theory for force control of an electro-hydraulic test machine,” Control Engineering Practice, 17, pp.1291-1306, June, 2009.
[11] Kiyun Jeong, Sehyun Kim, Raecheong Kang, Inbeom Yang, “BLDC MCU Failure Mode Evaluation by HILS,” The Korean Society of Automotive Engineers, pp.1438-1443 2011.
[12] Aksit, Mehmet, and Louis Marie Johannes Bergmans. “Obstacles in object-oriented software development,” ACM Sigplan Notices, 27.10, pp.341-358, 1992.

김 경 진
e-mail : klm0012@ajou.ac.kr
2012년 아주대학교 전자공학과(학사)
2014년 아주대학교 전자공학과(석사)
2014년～현 재 LG전자 MC사업부 연구원
관심분야 : 임베디드 시스템, 테스팅

정 기 현
e-mail : khchung@ajou.ac.kr
1984년 서강대학교 전자공학과(학사)
1988년 미국 Ilinois주립대 EECS(석사)
1990년 미국 Purdue대학 전기전자공학부 (박사)
1991년～1992년 현대반도체 연구소
1993년～현 재 아주대학교 전자공학과 교수
관심분야 : 컴퓨터구조, VLSI 설계, 멀티미디어/실시간 시스템 등

최 경 희
e-mail : khchoi@ajou.ac.kr
1976년 서울대학교 수학교육과(학사)
1979년 프랑스 그랑데꼴 Enseeiht 대학 (석사)
1982년 프랑스 Paul Sabatier 대학 정보공학부(박사)
1982년～현 재 아주대학교 컴퓨터공학과 교수
관심분야 : 운영체제, 분산시스템, 실시간/멀티미디어 시스템 등