KIPS Tr. Software and Data Eng.
Vol.3, No.6 pp.201~214 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2014.3.6.201

Destructive Test of a BLDC Motor Controller Utilizing a Modified Classification Tree Method

Shin Jae Hyuk† · Chung Ki Hyun†† · Choi Kyung Hee†††

ABSTRACT

In this paper, we propose a test case generation method adequate to destructive test of the BLDC(Brush Less Direct Current) motor controller used for the MDPS(Motor Driven Power Steering) system embedded in an automobile. The proposed method is a modified CTM(Classification Tree Method). CTM generates test cases assuming that all inputs are equally important. Therefore, it is very hard to generate test cases for extreme situations. To overcome the drawback and generate test cases specialized for destructive test. a modified CTM that compensates the limitation of traditional CTM is proposed. The proposed method has an advantage that it can intensively generate the test scenarios adequate to extreme situations by combining the test cases generated by the transitional CTM the while keeping the merit of the traditional CTM. The test scenarios for destructive test for the MDPS system embedded in a commercial automobile are generated utilizing the proposed method. The effectiveness of the proposed algorithm is verified through the test.

Keywords : Embedded Testing, Classification Tree Method, BLDC Motor

변형된 Classification Tree Method를 이용한 BLDC 모터제어기 파괴 시험

신 재 혁† · 정 기 현†† · 최 경 희†††

요 약

본 논문에서는 차량에 사용되는 MDPS(Motor Driven Power Steering) 시스템에 사용되는 BLDC(Brush Less Direct Current) 모터 제어기의 파괴 시험에 적합한 테스트 케이스 생성 방법을 제시한다. 제안하는 방법은 입력 기반 테스트 방법의 하나인 CTM(Classification Tree Method)을 수정한 방법이다. CTM은 모든 입력의 중요도를 동등하게 가정하고 테스트 케이스를 생성하는 방법이다. 따라서 극한 상황과 같은 테스트 케이스를 생성하기 힘들다. 이와 같은 단점을 극복하고, 파괴 시험에 적합한 테스트 케이스를 생성하기 위해 CTM의 한계점을 보완한 변형된 CTM 재구성 방법을 제안한다. 제안하는 방법은 CTM의 장점을 유지하면서 CTM으로 만들어진 테스트 케이스를 조합하여 특정 상황의 테스트에 적합한 시나리오를 집중적으로 생성할 수 있는 장점을 가진다. 제안한 방식을 이용해 파괴 시험을 위한 테스트 시나리오를 생성하고, 이를 이용하여 승용차에 사용되고 있는 MDPS 시스템에 대한 테스트를 수행하여 제안한 방법의 유용성을 검증한다.

키워드 : 임베디드 테스트, 분류 트리 방법(CTM), BLDC 모터

1. 서 론

현재 모바일 기기부터 의료 기기, 국방 사업에 이르기까지 많은 분야에서 임베디드 시스템이 사용된다. 이러한 환경 속에서 임베디드 시스템의 하드웨어 성능이 향상되고 있으며, 임베디드 시스템의 요구 사항 또한 복잡해지고 있다. 덕분에 시스템의 동작을 결정하는 소프트웨어의 중요성이 점차 대두되고 있다. 실제 임베디드 시스템에서 소프트웨어가 차지하는 비율은 10~20%에 불과하지만 발생하는 오류의 80% 이상이 하드웨어가 아닌 소프트웨어에 의한 오류이다[1]. 이러한 상황에 맞추어 시스템의 품질과 신뢰성을 확보하기 위한 테스트의 중요성이 높아지고 있다.

테스트는 소스 코드 수준의 시스템 분석을 토대로 테스트 케이스를 생성하는 화이트 박스 테스트와 입출력을 바탕으로 테스트 케이스를 생성하는 블랙박스 테스트로 나눌 수 있다. 화이트 박스 테스트는 테스트 대상 시스템의 소스코

† 준 회 원 : LG전자 VC사업부 연구원
†† 정 회 원 : 아주대학교 전자공학과 교수
††† 정 회 원 : 아주대학교 컴퓨터공학과 교수
Manuscript Received : November 4, 2013
First Revision : March 19, 2014
Accepted : March 21, 2014
* Corresponding Author : Chung Ki Hyun(khchung@ajou.ac.kr)

드 및 설계 사양을 확보할 수 있을 때 사용할 수 있다. 이와 같은 정보 획득이 어렵거나, 임베디드 시스템과 같이 하드웨어와 결합한 완성된 시스템에서 테스트를 시도할 경우에는 시스템의 입력과 출력 간의 관계를 이용한 블랙박스 테스트가 적합하다[2].

블랙박스 테스트를 위한 테스트 케이스 생성 기법으로는 모델 기반 생성 기법, 입력 조합을 이용한 생성 기법 등이 있다. 이 중 입력 조합을 기반으로 한 생성 기법은 테스트 대상 시스템의 자세한 사양이나 정보 없이 테스트 케이스 생성이 가능하다는 점 때문에 많이 이용되고 있다.

일반적인 임베디드 시스템의 입력 조합은 무수히 많다. 입력 수가 많을 뿐만 아니라, 각 입력이 가지는 값 또한 많아서 입력을 적절히 분류하여 테스트를 수행하는 것이 일반적이다. 입력 도메인을 특정한 규칙을 이용해 분할하는 방법을 파티셔닝 방법(partition method)이라 하고 입력 도메인에서 무작위로 입력 값을 선택하는 방법을 임의 방법(random method)이라고 한다.

일반적으로 파티셔닝 방법이 임의 방법보다 오류 발견을 더 잘할 수 있다고 알려져 있다[3]. 파티셔닝 방법을 사용할 때, 어떻게 파티셔닝을 설계하는가에 따라 좋은 테스트 케이스가 생성되기도 하고 의미 없는 테스트 케이스가 생성되기도 한다[4]. 그래서 파티셔닝을 잘 설계하고 테스트를 성공적으로 수행하기 위한 연구가 많이 진행되었다.

파티셔닝 설계 문제를 해결하기 위한 한 가지 방법은 CTM[5](Classification Tree Method)이다. CTM은 설계 명세에 근거하여 체계적이고 효율적으로 파티셔닝을 설계하기 위한 CPM[6](Category-Partition Method)을 향상시킨 방법으로, 파티셔닝 방법에 이론적 기반을 둔 효율적인 테스트 케이스 생성 기법이다. 이 CTM은 Daimler-Benz Group의 내부 조직들에서도 성공적으로 사용[7]되는 등 오랜 시간 동안 다양한 분야에서 사용해 온 검증된 방법이다.

최근 들어 BLDC(Blush Less Direct Current)모터가 다양한 분야에서 사용되는데, 특히 자동차에서의 활용이 두드러지게 나타난다[8]. 그런데 실제 자동차가 운전 중일 때 BLDC 모터를 구동하는 인버터 내의 MOSFET(Metal Oxide Semiconductor Field Effect Diode)소자가 소손되거나, 인버터 자체가 고장 나서 모터가 멈추는 상황이 간혹 발생한다. 모터 정지는 BLDC모터가 어떤 위치에서 사용되는가에 따라 탑승자의 생명을 위협할 수 있는 중요한 문제가 될 수 있다.

MOSFET 소손의 경우 다양한 상황들이 원인으로 제기되었고, 원인을 찾지 못하는 알 수 없는 상황에서 MOSFET 소자가 소손하는 경우도 존재한다[9]. 이렇게 갑자기 발생하는 MOSFET의 소손처럼 모터 구동 중 발생하는 인버터 고장에 대처하기 위한 연구들이 많이 존재한다. 이 연구들은 시스템이 어떤 원인으로 고장이 발생했는지 파악[10]하고 고장을 시스템에서 분리하거나 복구[11]하도록 구성하여 시스템의 안정성과 신뢰성을 확보하기를 제안한다. 하지만 이러한 연구는 시뮬레이션이나 드라이버 단만 테스트 할 뿐 모터까지 연결된 실제 시스템의 동작을 테스트 하지 않는다.

따라서 자동차와 같은 가혹한 환경에서 구동되는 임베디드 시스템의 안전성과 신뢰성 검증을 검증하기 위해서는 임베디드 시스템을 실제 그 시스템이 구동하는 환경과 흡사한 환경을 구성하여 시뮬레이션(HILS: Hardware In the Loop)하는 방법과 실제 환경에서 그 시스템을 장착하여 수행하는 데스트 방법들이 있다.

본 논문은 실제 자동차에 사용되는 BLDC 모터와 인버터에 대해 수행한 기능 테스트 및 파괴 테스트에 관한 내용을 다룬다. CTM을 이용해 BLDC 모터를 내장한 MDPS(Motor Driven Power Steering) 시스템을 대상으로 테스트를 수행한다. 또한 특정한 경우를 테스트하기 위한 테스트 목적을 달성하기 위해 특정 값을 묶고 테스트 케이스를 생성하는 고정 CTM과 트리 자체를 재구성하여 테스트 케이스를 생성하는 방법을 제안한다. 이러한 내용을 통한 본 논문의 주요한 기여는 다음과 같다.

- 특정 입력을 고정하여 특정 상태를 집중적으로 테스트할 수 있는 테스트 케이스를 생성하는 방법과 트리 자체를 재구성하여 CTM이 가지는 한계를 보완하는 방법을 제안한다.
- CTM으로 만들어진 테스트 케이스를 조합하여 파괴 시험을 위한 데스트 시나리오를 만드는 방법을 제안한다.
- 실제 자동차의 MDPS에 장착되는 BLDC 모터 제어기의 파괴 테스트를 위하여 제안된 방법으로 만들어진 테스트 시나리오를 적용해 테스트 하고 결과를 분석한다. 이를 통하여 제안한 방법의 유용성을 밝힌다.

본 논문의 구성은 다음과 같다. 2장에서 CTM에 대하여 설명하고 CTM의 장단점을 살펴본다. 또한 CTM과 관련된 몇 가지 연구들을 분석한다. 3장에서는 테스트의 대상이 되는 BLDC 제어기에 관해 설명한다. 4장에서는 새로운 CTM 기법을 제안하고 이를 활용하여 테스트 케이스를 생성하는 방법을 소개한다. 5장에서 실제 차량에 사용되는, BLDC 모터를 이용하는 MDPS 시스템을 대상으로 하여 테스트 환경을 꾸민다. 그리고 테스트 케이스를 생성해 테스트를 수행하고 결과를 분석한다. 마지막으로 6장에서 테스트 결과를 정리하고 추가적으로 진행되어야 할 연구에 대해 설명한다.

2. 관련 연구

CTM은 시스템을 분류 트리(Classification tree)로 구현하여 효과적인 파티셔닝을 설계하는 데 유용하게 사용할 수 있는 방법으로, 블랙박스 테스트의 한 종류인 파티셔닝 방법에 이론적 기반을 두고 있고 있는 카테고리 파티셔닝 방법(category-partition method)의 개념을 향상시킨 방법이다.

2.1 CTM 개념

CTM의 제안자인 Grimm과 Grochtmann은 테스트 대상

선정(Selecting test objects), 분류 트리 설계(Designing a classification tree) 그리고 테스트 케이스 생성을 위한 분류 클래스 조합(Combining classes to form test cases)이라는 세 단계의 과정을 거쳐 복잡한 시스템을 CTM으로 테스트 하는 방법을 설명한다[7].

1) 테스트 대상 선정

실제 큰 규모의 시스템을 하나의 트리로 표현하는 것은 힘들고 이를 이용해 테스트 케이스를 생성하기도 어렵다. 따라서 하나의 복잡한 시스템을 여러 개의 트리로 나누어 테스트 케이스를 생성해야 한다.

이를 위한 CTM의 첫 단계는 테스트 대상 시스템(SUT: System Under Test)을 여러 개의 "테스트 대상"(test object)으로 나누는 것이다. "테스트 대상"이란 SUT를 기능별로 분류하거나 하위 모듈로 분류한 것을 의미한다. 이를 선택할 때, 각각의 "테스트 대상"들은 서로 독립적이어야 한다. 그리고 "테스트 대상"을 모두 합치면 전체 시스템이 되어야 한다. 이러한 조건을 만족하도록 전체 SUT를 "테스트 대상"들로 분류한다.

2) 분류 트리 설계

SUT의 입력을 트리를 이용해 간단한 형태로 표현한 것이 "분류 트리"이다. 이때 시스템의 상태 천이(state transition)와 같이 어떠한 동작이 일어나기 위한 명세서상의 조건을 "분류 조건"(classification)이라 하고 트리의 노드로 표현한다. 그리고 각 분류조건의 입력이 될 수 있는 값을 "클래스"(class)라고 한다.

이 "분류 조건"과 "클래스"를 설계하는 데 가장 필요한 정보는 시스템에 대한 명세이다. CTM의 두 번째 단계에서 테스터는 경험과 창의성과 명세서를 바탕으로 각 "테스트 대상"을 위한 "분류 트리"를 생성한다. 이 단계가 CTM의 성능에 많은 영향을 미치는 단계이다.

3) 분류 클래스 조합으로 테스트 케이스 생성

"테스트 대상"에 대해 구성한 "분류 트리"를 이용해 테스트 케이스를 생성한다. Grimm과 Grochtmann은 테스트의 효율성을 위해 테스트 케이스를 적게 생성하면서 "테스트 대상"의 중요부분을 확인할 수 있는 방법을 제안했다.

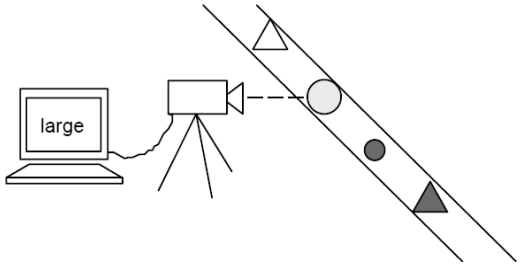


Fig. 1. A sample computer Vision System

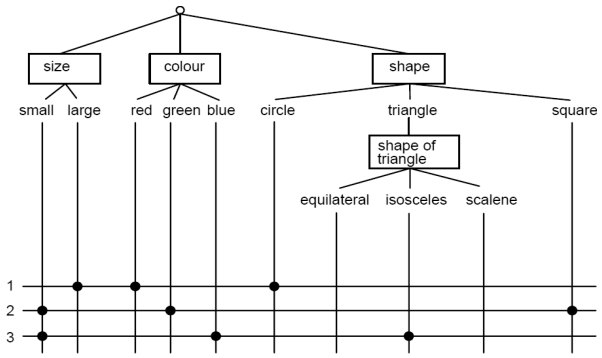


Fig. 2. Classification Tree and Test case generation

Fig. 1, 2[7]는 작업 통로를 통해 들어오는 물건을 카메라를 이용해 파악하고 분류하는 가상 시스템에 대해 CTM을 적용하는 예시이다. (1) 테스트 대상 선정 단계에서 SUT의 여러 가지 동작 중 카메라의 물건 인식기능을 "테스트 대상"으로 선정하였다. (2) 분류 트리 설계 단계에서 "분류 조건"을 "size", "colour" 그리고 "shape"로 분류한다. 분류조건 "size" 및 "colour"는 각각 2, 3개의 "클래스"를 가진다. 한편, 분류조건 "shape"는 두 개의 "클래스"와 하나의 하위 "분류조건"인 "shape of triangle"를 가진다. (3) 분류 클래스 조합으로 테스트 케이스 생성 단계에서는 각 "클래스"들의 값을 조합하여 테스트 케이스를 생성하는 것을 볼 수 있다.

2.2 CTM의 장단점

파티셔닝 방법의 단점은 파티셔닝 설계가 테스터의 경험과 생각에 따라 다양하게 나타난다는 것이다. 이런 특성 때문에 테스트 케이스를 생성하면 좋은 테스트 케이스가 나오기도 하고, 의미 없는 테스트 케이스가 나오기도 한다. CTM은 이러한 파티셔닝의 문제를 다소 해결할 수 있는 장점이 있다. 파티션 설계를 테스터의 경험에 의존해야 한다는 문제를 CTM이 완전히 해결한 것은 아니지만 분류 트리를 이용해 체계적인 방법으로 설계하도록 해주며 테스트 케이스의 의미를 파악하기 쉽게 도와준다. 또한 테스트 케이스를 생성할 때 CTM은 효율성에 초점을 두고 테스트 케이스를 최소한으로 생성하거나, 테스트 결과를 위해 최대한의 개수로 생성할 수 있다. 이렇듯 CTM은 상황에 따라 필요한 전략을 선택할 수 있는 유연함이 있다.

하지만 클래스의 개수가 많아지면 모든 클래스 조합을 이용한 테스트 케이스 생성이 현실적으로 불가능해진다. 이 경우 CTM을 이용하여 생성된 테스트 케이스는 좋은 테스트 케이스 집합이 될 수 없다[12]. 복잡한 시스템의 경우, 사람의 생각이나 시스템의 사용을 테스트에 적용하는 시나리오 기반 테스트가 효과적이다[13]. 하지만 CTM으로는 시나리오 기반이나, 특정 순서를 가지는 테스트 케이스는 생성할 수 없다. 따라서 CTM을 사용하더라도 더 성공적인 테스트를 위해서는 시나리오 테스트 기법 같은 다양한 방법을 적용해야 한다.

2.3 관련 연구

1) Classification Tree Editor(CTE)

분류 트리로부터 테스트 케이스를 생성하기 위해 클래스들을 조합하는 규칙은 간단하다. 하지만 직접 분류 트리를 그리거나 클래스를 조합하는 것은 번거로운 일이다. CTE는 이런 작업을 편하게 하고, 자동으로 테스트 케이스를 생성도록 도와주는 도구이다[14]. 현재 Hitex, berner & mattner와 같은 곳에서 CTE를 개발, 판매하고 있다.

CTE는 트리를 그리는 기본적인 기능뿐만 아니라 트리로부터 테스트 케이스를 생성할 때 클래스 간의 규칙을 정하거나, 생성한 테스트를 그룹으로 관리할 수 있는 기능을 제공하기도 한다.

2) Integrated Classification Tree Methodology

CTM의 테스트 케이스 생성 방식은 테스터의 경험에 따라 좌우되는 경우가 있다. Chen과 Poon은 이러한 CTM의 불확실성을 해결하기 위해 "계층적 분류표"(classification hierarchy table)을 이용하여 체계적으로 분류 트리를 만드는 방법을 제안했다[15]. 이후 분류표를 바탕으로 트리 설계 및 테스트 케이스를 생성에 관한 과정을 종합하여 ICTM (Integrated Classification Tree Methodology)을 제안하였다[16].

ICTM은 "계층적 분류표"을 토대로 분류 트리를 만든다. "분류표"를 만들기 위해서 각 분류조건들의 상관관계를 조사한다. 이후 분류조건들이 서로 종속 관계일 때 '⇒'기호, 서로 동시에 존재 할 수 없을 때 '~'기호, 서로 아무런 관계가 없을 경우 '⊗'기호를 사용하여 표시한다.

예를 들어 A, B, C, D라는 4개의 분류조건이 있으며 각각 a1, a2, b1, b2, c1, c2, d1, d2를 클래스로 가진다고 할 때, 시스템 특성상 다음의 조합들은 입력이 될 수 없다고 가정한다.

(A is a2) and (C is c1 or c2)
(A is a2) and (D is d1 or d2)
(C is c1 or c2) and (D is d1 or d2)

이러한 입력 값들의 관계를 바탕으로 분류조건 간의 관계를 확인한다. Table 1은 위 관계를 이용하여 만든 "계층적 분류표"이다.

Table 1. Example of hierarchy table

	A	B	C	D
A	⊗	⊗	⇒	⇒
B	⊗	⊗	⊗	⊗
C	⊗	⊗	⊗	~
D	⊗	⊗	~	⊗

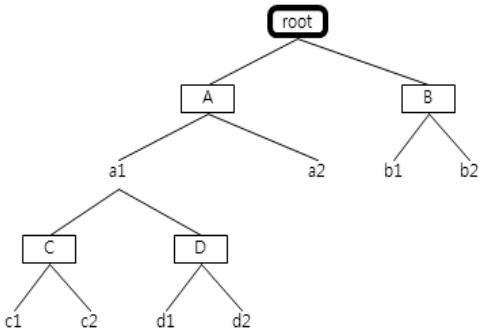


Fig. 3. Classification tree using hierarchy table

"계층적 분류표"를 기반으로 분류 트리를 만든 것이 Fig. 3이다.

ICTM은 "계층적 분류표"를 이용해 완성한 트리로부터 테스트 케이스를 생성한다. 이때 "계층적 분류 표"의 입력간의 관계를 테스트 케이스 생성에 반영한다. 이를 통해 ICTM은 체계적인 트리 설계 기법과 효율적인 테스트 케이스 생성 방법을 제공한다.

하지만 모든 분류조건들의 관계가 독립적일 때 ICTM방식은 기존의 CTM과 같은 결과가 나온다. 또한 입력의 관계가 분명하게 드러나야 "계층적 분류표"를 생성할 수 있기 때문에 사용에 제한이 있다.

3) 모델/시나리오 기반 방법

문서 기반의 비정형화된 명세서는 상황이나 사람에 따라 다르게 해석될 수 있다. 때문에 시스템을 잘못 이해해서 모호한 테스트 케이스를 생성하거나 구현 불가능한 상황을 만들기도 한다. 많은 경우 이러한 비정형화된 명세서가 주로 사용되고 이로 인해 발생하는 문제가 많다.

명세서 비정형화의 문제를 해결하기 위하여 시스템의 개발 과정이 모델을 기반으로 진행되기도 한다. 임베디드 시스템 개발 및 테스팅에 관련된 연구[17,18]는 비정형화된 명세서가 야기하는 문제들과 시스템 개발 방식의 추세를 볼 때 정형화 명세서를 이용한 모델기반 방법과 특정 상황을 표현하는 시나리오 기반 방법이 합리적이라고 말한다.

이때 CTM을 이용한 테스트는 명세서를 이용하거나 모델을 이용하는 경우 모두에 적용하여 사용할 수 있다. 이런 이유 때문에 모델기반 방법에서 CTM을 사용하는 방법을 모색하기도 한다[18,19].

3. 차량용 BLDC 모터의 인버터

자동차에서 사용하거나 생성되는 전원은 12.5V~15V 사이의 DC 전원이다. 3상 AC전원을 필요로 하는 차량용 BLDC 모터를 위해서는 DC전원을 3상 AC전원으로 변화시키는 인버터가 필요하다.

일반적으로 인버터는 power TR, IGBT, power MOSFET 등의 소자를 이용해 구현한다. Fig. 4는 3상 AC전원 생성을

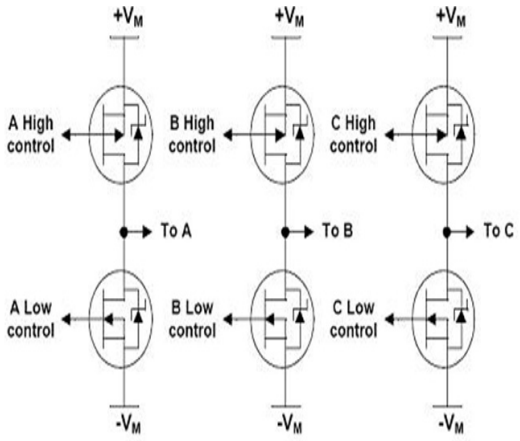


Fig. 4. Conceptual configuration of 3 phase inverter

위한 인버터 회로의 개념도이다. 제어기가 “A high control”, “A low control”처럼 각 A, B C상을 위한 여섯 개의 제어 신호로 3상 전원을 생성하여 모터에 공급하게 된다.

그런데 실제 시스템에서 인버터 회로가 파손되는 경우가 종종 생긴다. 시스템이 구동 중에 인버터가 파손되는 원인은 다양한 것이 존재한다. 예를 들면, 컨트롤러의 문제로 과전압이나 과전류가 인가되어 발생하는 “애벌런시 파괴”(avalanche failure), 물이나 먼지 같은 이유로 발생하는 “외부 요인 파괴”(foreign object failure), 모터의 갑작스러운 정지나 강한 부하에 의해 발생하는 “역기전력 파괴”(Back EMF failure) 등의 원인에 의해 인버터가 파손될 수 있다. 심지어는 이러한 원인들 때문에 모터 자체가 파손되는 경우도 있다.

이렇게 시스템의 일부가 파손되는 경우 모터의 구동은 불가능하게 된다. 모터가 시스템에서 사용된 위치에 따라 모터의 정지는 인명피해와 같은 큰 사고로 이어질 수 있다. 따라서 자동차 내에 BLDC 모터의 사용이 증가됨에 따라 시스템의 안전을 위해 BLDC 모터와 인버터에 대한 안전성 및 신뢰성을 확실히 검증해야 한다.

4. 파괴 시험을 위한 테스트 케이스 생성

테스트는 실제 차량에 사용되는 조향 도움 장치 MDPS 시스템에 내장되는 BLDC 모터를 대상으로 수행한다. 대상 MDPS는 BLDC 모터 시스템을 이용하여 그 기능을 구현하고 있다. MDPS는 시스템은 차량 주행 시 오류가 발생하면 큰 사고로 연결될 수 있는 중요한 부분이기 때문에 매우 높은 신뢰도가 요구된다. 이번 장에서 MDPS 시스템의 입력을 기반으로 분류 트리를 구성하고, 극한 상황 시 시스템이 견딜 수 있는지를 테스트하기 위한 파괴 테스트 케이스를 CTM기법을 이용하여 생성한다.

4.1 분류 트리 구성

테스트 대상 시스템(SUT: System Under Test)인 MDPS의 동작을 결정하는 다양한 입력들이 존재한다. 이 입력은 내장된 BLDC모터 제어 시스템에 전달된다. 입력의 종류에는 차량 내 다른 전자제어장치(ECU: Electrical Control Unit)들로부터 오는 다양한 종류의 CAN 메시지, 배터리 전원, 사용자의 핸들 조작을 통한 센서 신호, 홀 센서 신호, 전류 센서 신호, 온도 센서 신호, 모터 속도 등이 있다. 또한 바퀴를 통해 전해지는 부하나 습도, 먼지, 외부 온도와 같은 환경적인 요인들이 시스템의 외부 입력으로 인가될 수 있다.

시스템에 인가될 수 있는 다양한 입력을 분류조건으로 선정하여 분류 트리를 구성한다. 이후 명세서 분석을 통해 각 분류조건에 적합한 클래스를 선택한다. 또한 외부 입력들 중 자동차 바퀴를 통해 인가되는 부하는 MDPS 시스템의 직접적인 입력은 아니지만 시스템의 동작에 큰 영향을 미치는 중요한 외부환경이다. 때문에 부하를 분류 트리에 포함시킨다.

전체 시스템(MDPS)의 분류조건은 전원 “BAT”, 핸들 조작 “Handle”, 모터에 걸리는 부하 “Load”, Hall 센서의 신호 “Hall data”, 점화 장치 “Ignition Key”, 모터 온도 “Temp data”, 모터 전류 “Current”, 그리고 다른 ECU나 사용자로부터 전송된 CAN 데이터 “CAN DATA”라는 여덟 개의 분류조건으로 나누어진다.

분류조건의 클래스는 극한 상황을 만들기 위한 조합으로 구성한다. 배터리를 통한 전원 공급을 의미하는 “BAT”는 전원 공급이 전혀 되지 않거나(off), 정상적인 전원(13V), 저전압(9V) 및 고전압(16V)인 경우에 각기 다른 동작을 한다. 따라서 이들을 “BAT”의 클래스로 선정한다. 핸들의 움직임을 나타내는 “Handle”은 TAS(Torque and Angle Sensor)로부터 핸들의 위치 및 토크를 받기 때문에 두 개의 하위 분류조건을 가진다. 첫 번째는 핸들이 위치하고 있는 곳에 따라 왼쪽 끝(Left lock), 오른쪽 끝(Right lock) 그리고 양쪽 끝이 아닌 임의의 위치 중간(Center) 클래스를 가지는 “위치”(Position)이다. 두 번째는 핸들과 바퀴의 비틀림 정도를 나타내는 “TAS input”이다. “TAS input”은 핸들이 어떤 방향으로 비틀려 있는가에 따라 왼쪽(Left), 오른쪽(Right), 비틀리지 않음(Stop)을 클래스로 가진다. 내・외부에서 인가되는 모터의 부하 “Load”또한 두 개의 하위 분류조건을 가진다. 부하가 모터에 인가되는 방향(Direction)에 따라 정방향(CW: Clock Wise)과 역방향(CCW: Counter Clock Wise)으로 나타낼 수 있다. 그리고 인가되는 토크(Torque)는 일반적인 상황의 약한 부하(Low), 돌부리에 걸리는 것같이 가끔 발행하는 강한 부하(High), 바퀴가 구멍에 빠지거나 걸려서 움직이지도 못하는 특별한 상황인 정지 부하(Stop)를 클래스로 가진다. 자동차의 시동에 사용되며 MDPS의 구동 여부를 결정하는 “Ignition key”는 가동(on) 및 미가동(off)의 클래스를 가진다. 모터 상태를 검사하기 위해 존재하는 Hall 센서, 온도 센서, 전류 센서에 해당하는 “Hall data”, “Temp data”, “Current”들은 모두 정상 상태(normal)와 고장 상태(malfunction)를 클래스로 가진다. 마지막으로 CAN 통신을 통해 들어가는 신호 “CAN DATA”는 MDPS 동작을 위해 필요한 여러 가지 데이터뿐만 아니라 개발 과정이나 디버깅 용도로 사용된 다양한 명령 및 제품 생산 후 검사 및 관리

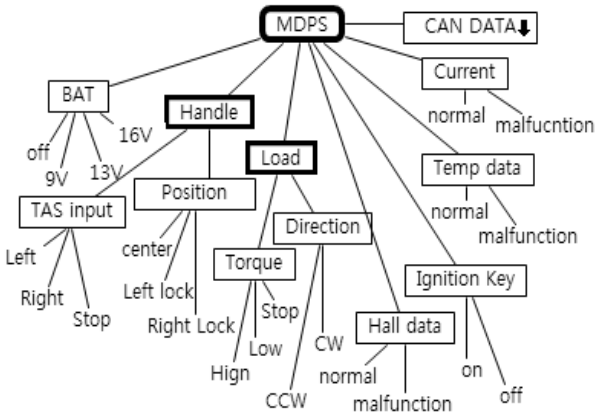


Fig. 5. Classification Tree for MDPS

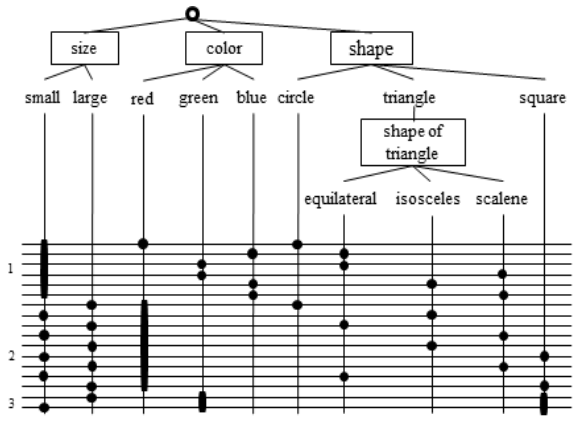


Fig. 6. Test case generation using fixed class CTM

를 위한 명령 등 다양한 클래스가 존재한다.

Fig. 5는 위의 분류조건과 클래스를 사용해 구성한 분류 트리이다. CAN DATA는 앞서 설명한 것처럼 외부 모듈들로부터의 메시지, 모니터링을 위한 각종 변수의 데이터를 요구하는 메시지 등 매우 많은 클래스를 가지기 때문에 그림에 표시하지 않는다.

기존 CTM은 테스트의 효율성이나 추구하는 커버리지에 알맞게 테스트 케이스를 생성한다. 하지만 본 연구에서는 MDPS의 기능 테스트 같은 일반적인 상황뿐만 아니라 극한 동작 시 BLDC 모터 및 인버터의 손상에 대한 시스템의 신뢰성 검증이라는 테스트 목적에 적합하게 테스트 케이스를 생성하는 것을 추구한다. 즉, 특정 조건에서 소프트웨어 기능 및 하드웨어 기능을 중점적으로 테스트할 수 있는 케이스를 생성하고자 한다. 특수한 테스트 목적에 부합하는 테스트 케이스를 생성하기 위해 다음과 같은 방법들을 제안한다.

1) 클래스 고정 CTM

기존 CTM을 이용한 생성 규칙에 따라 테스트 케이스를 생성하는 경우, 특정 클래스가 다른 요소보다 중요하거나, 특정 상황에서 집중적으로 테스트를 진행하기 위한 테스트 케이스 생성 필요 등은 테스트 케이스에 생성에 특별히 반영되지 않는다. 일반적으로 SUT이 입력이 많아 모든 클래스의 조합을 다 생성할 수 없기 때문에 특정 클래스에서 테스트 케이스를 집중적으로 생성하는 것은 기존의 방법으로는 어렵다. Fig. 2의 분류 트리를 이용하여 기존 방법으로 테스트 케이스를 생성하는 경우를 예로 들면, 주요 대상이 “color” 분류조건의 “blue” 클래스일 때 다른 분류조건의 클래스를 위주로 테스트 할 수 있는 테스트 케이스를 생성하지 못한다. 이 경우는 “color” 분류조건의 “blue” 클래스를 고정시키고 다른 분류조건의 클래스만을 바꾸어 테스트 케이스를 생성 하는 것이 좋다.

Fig. 6과 같이 특정 클래스를 고정해 놓고 테스트 케이스를 집중적으로 생성할 경우, 특정 조건에 맞는 테스트 케이스를 집중적으로 생성할 수 있다. 테스트 케이스 집합 1은 분류조건 “size”의 클래스 “small”을 고정하고 있고, 테스트 케이스 집합 3은 분류조건 “color”의 클래스 “green”과 분류조건 “shape”의 클래스 “square”를 고정하고 다른 클래스 들을 조합하여 테스트 케이스를 생성하는 것을 보여주고 있다.

클래스 고정 CTM을 MDPS 시스템 테스트에 사용할 경우 몇 가지 클래스를 고정함으로써 차량의 정상주행 상태에 조건을 유지한 채, 중요한 요소라고 판단되는 중요한 클래스를 변화시켜 가면서 테스트 케이스를 만들 수 있다. 예를 들어 모든 테스트 케이스에 분류조건 “BAT”을 클래스 “13V”로 고정하고, 분류조건 “Ignition Key”를 클래스 “on”으로 고정하는 경우, 자동차열쇠, 차량발전기(Alternator), 전압조정기(Regulator)가 정상적인 상태에서 다른 조건에 따라 MDPS가 어떻게 동작하는지를 집중적으로 테스트할 수 있는 테스트 케이스를 생성함으로써 정상 상태에서 제어기 파괴에 대한 안전성 검증 같은 극한 테스트를 수행할 수 있다. 이 방법은 CTM을 지원하는 도구인 CTE 등에서 고장하고자 하는 클래스를 고정하고 테스트 케이스를 생성하면 가능하다.

클래스를 고정하여 놓고 테스트 케이스를 생성할 경우 원하는 테스트 케이스를 집중적으로 생성할 수 있는 장점을 가진다. N개의 클래스를 가진 분류트리에서 테스트 케이스를 생성할 경우를 예를 들어 보자. 클래스들의 모든 조합을 생성할 경우, 2N개의 케이스가 생성된다. N이 큰 값일 경우 생성된 2N개의 케이스는 현실적으로 모두 테스트 할 수 없다. 따라서 pairwise[20]와 같은 방법을 사용하여 적절한 수의 테스트 케이스를 생성하여 사용한다. 하지만 만약 N개의 클래스 중, M개의 클래스를 고정 시킬 경우, (N-M)개의 고정되지 않는 클래스만을 가지고 테스트 케이스를 만들기 때문에 생성되는 테스트 케이스 수가 훨씬 줄어들 것이다. (N-M)이 작은 수일 경우 모든 조합에 대한 테스트케이스를 만든다 하더라도 2(N-M)개의 테스트케이스는 현실적으로 테스트 가능한 수가 될 것이다. 따라서 파괴 시험 같은 극한 환경에서 다양한 조합의 테스트 케이스 생성에서는 특수한 환경을 유지하는 일부 클래스를 고정하고 테스트 하는 것이 매우 유리할 수 있다.

2) 분류 트리 재구성

SUT를 모델링하여 생성한 분류 트리 Fig. 5를 그대로 사용하여 테스트 케이스를 생성할 경우 매우 많은 테스트 케이스가 생성된다. 따라서 특정 조건과 목적에 부합하는 테스트 케이스를 효율적으로 생성하기 위해 분류 트리 자체를 축소하여 재구성한다. 분류 트리를 재구성 할 때, 중요하다고 판단되는 클래스에 대한 테스트 케이스를 집중적으로 생성하기 위해 중요하지 않다고 판단되는 클래스를 고정한다. 또한, 중요하다고 판단된 분류조건을 더욱 세분화하여 다양한 클래스로 나누는 방법을 사용한다.

먼저 Fig. 5의 분류 트리에서 고정 CTM의 아이디어를 적용해 분류 트리를 재구성한다. 정상주행 중 발생하는 BLDC 모터 및 컨트롤러의 고장에 대해 SUT의 신뢰성을 검증하기 위한 테스트 목적에 맞게, 중점적으로 테스트 하고자하는 클래스를 제외한 몇 가지 분류조건의 클래스를 고정하고 분류 트리를 재구성한다. 분류조건 “BAT”은 정상전압 “13V”으로 고정한다. “Ignition Key”는 동작 상태 “on”으로 고정한다. “Hall data”, “Temp data”, “Current”는 모두 정상 상태를 의미하는 “normal”로 고정한다. “CAN DATA”는 MDPS가 정상적으로 운전하기 필요한 “엔진 신호”를 고정한다. 이들 여섯 개의 “분류 조건”들의 클래스는 중요하다고 판단되는 분류조건에서 제외된다. 고정된 여섯 개의 클래스 들은 테스트를 수행 동안 SUT에 고정된 값이 입력된다.

대상 SUT인 MDPS의 극한 시험에서 중요하다고 판단되는 “Handle” 및 “Load” 분류조건도 테스트 목적에 맞게 분석을 통해 재구성한다. Fig. 5에서의 “TAS input”은 세 가지의 클래스를 가진다. 이는 명세서 상의 MDPS 동작에 관한 설명일 뿐 이를 이용해 극한의 상황을 만들기는 어렵다. 따라서 “TAS input”의 클래스를 세분화 하여 “Rmax”, “Rmed”, “Lmax”, “Lmed”, “STOP”이라는 다섯 개의 클래스로 나눈다. 또한 토크(Torque)는 방향성을 가진 힘의 단위이기 때문에 “Load” “분류 조건”을 “Torque”만을 이용해 표현하도록 재구성한다. 차량에 걸리는 부하를 분석하고 구현하기 위해 “H+”, “L+”, “STOP”, “L-”, “H-”라는 “클래스”로 나눈다.

Fig. 7은 이러한 과정을 통해 재구성 된 분류 트리를 보여준다. 또한 각 클래스의 의미는 다음과 같다.

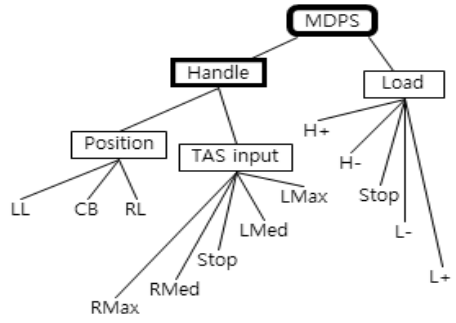


Fig. 7. Reconstructed classification tree

Position:

- LL(Left LOCK): 핸들이 가장 왼쪽까지 돌아서 더 이상 왼쪽으로 진행할 수 없는 상태
- RL(Right Lock): 핸들이 가장 오른쪽까지 돌아서 더 이상 오른쪽으로 진행할 수 없는 상태
- CN(Center): LL과 RL 사이에 핸들이 위치한 것을 의미하는 대표값

TAS input:

- LMax(Left Maximum): 왼쪽으로 핸들을 빠른 속도로 돌릴 때 발생하는 TAS 신호
- RMed(Left Medium): 왼쪽으로 핸들을 중간 속도로 돌릴 때 발생하는 TAS 신호
- Stop: 핸들을 움직이지 않을 때의 TAS 신호
- RMed(Right Medium): 오른쪽으로 핸들을 중간 속도로 돌릴 때 발생하는 TAS 신호
- RMax(Right Maximum): 오른쪽으로 핸들을 빠른 속도로 돌릴 때 발생하는 TAS 신호

Load:

- H+(High plus): MDPS의 힘과 반대 방향으로 MDPS 힘보다 작게 가하는 상황
- L+(Low plus): MDPS의 힘과 반대 방향으로 MDPS 힘보다 아주 작게 가하는 상황
- Stop: 정지 토크를 가하여 움직이지 않게 하는 상황
- L-(Low minus): MDPS의 힘과 반대 방향으로 MDPS 힘보다 강하게 가하는 상황
- H-(High minus): MDPS의 힘과 반대 방향으로 MDPS 힘보다 아주 강하게 가하는 상황

4.2 테스트 케이스 생성

본 논문에서는 재구성된 CTM을 이용하여 테스트 케이스를 생성하고 테스트 케이스를 다양한 방법으로 조합하여 CTM의 한계점을 보완한다.

1) 기본 테스트 케이스 조합

재구성된 분류 트리의 모든 클래스를 조합하여 테스트 케이스를 생성한다. 이를 통해 테스트에 대한 신뢰성을 최대한 올릴 수 있다. Fig. 7의 “Position”, “TAS input”, “Load”는 각각 3개, 5개, 5개의 클래스를 가진다. 이들을 전부 조합하면 모두 75개의 테스트 케이스(3×5×5)를 만들 수 있다.

이 중에서 실행불가 테스트 케이스를 제거하면 55개의 기본 테스트 케이스가 남는다. 실행불가 테스트 케이스의 한 가지 예시는 분류조건 “position”, “TAS input”, “Load”가 각각 클래스 “LL”, “LMax”, “L+”인 경우이다. 이는 핸들을 왼쪽 한계까지 돌린 상태에서 핸들을 왼쪽으로 강하게 회전하는 입력이다. 이는 자동차 핸들이 부서져서 회전하지 않는 이상 불가능한 동작이다. 이와 같이 구현 불가능한 입력들을 제외한 기본적인 55개의 테스트 케이스는 Table. 2와 같다.

Table 2. Base test cases

Number	Preposition			TAS input					Load				
	LL	CN	RL	LMax	LMed	Stop	RMed	RMax	Stop	H+	L+	H-	L-
1	o			o					o				
2	o			o								o	
3	o			o									o
4	o				o				o				
5	o				o							o	
6	o				o								o
7	o					o			o				
8	o					o				o			
9	o					o					o		
10	o						o		o				
11	o						o			o			
12	o						o				o		
13	o							o	o				
14	o							o		o			
15	o							o			o		
16		o		o					o				
17		o		o						o			
18		o		o							o		
19		o		o								o	
20		o		o									o
21		o			o				o				
22		o			o					o			
23		o			o						o		
24		o			o							o	
25		o			o								o
26		o				o			o				
27		o				o				o			
28		o				o					o		
29		o				o						o	
30		o				o							o
31		o					o		o				
32		o					o			o			
33		o					o				o		
34		o					o					o	
35		o					o						o
36		o						o	o				
37		o						o		o			
38		o						o			o		
39		o						o				o	
40		o						o					o
41			o	o					o				
42			o	o						o			
43			o	o							o		
44			o		o				o				
45			o		o					o			
46			o		o						o		
47			o			o			o				
48			o			o						o	
49			o			o							o
50			o				o		o				
51			o				o					o	
52			o				o						o
53			o					o	o				
54			o					o				o	
55			o					o					o

각각의 55개의 테스트 케이스는 핸들을 한쪽 방향으로 돌리거나, 바퀴에 힘이 작용해 부하가 생기는 단편적인 의미만 가지고 있다. 이 단편적인 55개의 동작을 조합하여 다양한 상황을 생성한다.

2) 이중 조합

하나의 테스트 케이스는 전, 후에 실행되는 테스트 케이스와는 무관하게 특정한 사건에 대한 단순한 사건을 테스트하는 목적으로 사용된다. 하지만 두 개 이상의 테스트 케이스가 일정한 순서로 실행될 경우 특별한 의미를 가질 수 있다. 예를 들면, 우회전하는 테스트 케이스와 외부 부하에 의해 정지하는 테스트 케이스가 순서대로 실행되면, 이는 핸들 조작 중 돌부리에 걸리는 것 같은 극한 테스트를 위한 상황이 된다.

이와 같이 두 개의 테스트 케이스를 연속하여 수행할 시 하나의 테스트 케이스를 수행할 때와는 다른 의미를 가질 수 있다. 이때 테스트 케이스 선택 순서에 따라 의미가 달라지며 생성되는 테스트 케이스 숫자는 다음과 같다.

$$ {}_{55}P_2 = \frac{55!}{(55-2)!} = 55 \times 54 = 2970 $$

2,970개의 테스트 케이스 중 실행불가 테스트 케이스를 제거한다. 실행불가 테스트 케이스의 한 종류는 Table 2의 12, 9번의 조합이다. 12번은 “Position”이 “LL”인 상태에서 핸들을 오른쪽으로 돌리는 동작이다. 이 동작 이후에 이어지는 동작은 9번과 같이 “Position”이 “LL”인 상황이 올 수가 없다.

이러한 실행불가 테스트 케이스를 모두 제외하면 2,970개의 조합된 테스트 케이스 중 1240개의 조합된 실행가능(feasible) 테스트 케이스를 구할 수 있다. 실행가능 테스트 케이스의 한 가지 예는 Table 2의 12, 31번의 조합이다. 12번은 앞서 설명한 것처럼 핸들을 우측으로 돌리고 모터가 회전하는 상황이다. 이때 31번은 핸들 조작은 12번과 마찬가지지만, “Load”가 “STOP”인 입력으로 부하에 의해 모터가 멈춰있는 상황이다. 다시 말해 12, 31번 조합은 핸들이 왼쪽 한계까지 돌아간 상태에서 운전자가 핸들을 우측으로 돌리다가 돌부리나 구멍 등에 의해 외부 부하에 의해 모터가 정지하는 상황이다. 이와 같이 조합을 통해 연속적인 상황이 만들어진다.

3) 삼중 조합

세 개 테스트 케이스의 순서에 따라 극한 상황이 발생하는 경우, 이에 대한 테스트 케이스 생성을 위해 앞에서 추출한 55개의 테스트 케이스에서 세 개의 테스트 케이스를 순열로 선택하면 다음과 같은 조합을 얻을 수 있다.

$$ {}_{55}P_3 = \frac{55!}{(55-3)!} = 55 \times 54 \times 53 = 157410 $$

이 중에서 실행불가능 테스트 케이스를 골라내는 작업은 어렵기 때문에 1240개의 “double permutation”에 한 가지 테스트 케이스를 조합하는 방법으로 “triple pair”를 구성한다. 이 때 조합의 세 번째에 위치 할 수 있는 테스트 케이스는 두 번째 위치의 테스트 케이스를 제외한 54개이다. 따라서 이론적으로 가능한 조합의 수는 1,240 × 54 = 66,960이다.

66,960개의 테스트 케이스 중 실행 불가능한 한 가지 예는 12, 17, 12번과 같은 조합이다. 12번은 “Position”이 “LL”인 상태에서 핸들을 오른쪽으로 돌린다. 이어지는 17번은 “TAS input”이 “Lmed”, “Load”가 “H+”로써, 핸들을 왼쪽으로 살짝 돌리지만 돌부리와 같은 외부 부하에 의해 바퀴가 오른쪽으로 돌아 핸들이 오른쪽으로 돌아가는 상황이다. 따라서 마지막 세 번째 입력에서는 “Position”이 “LL”이 될 수 없기 때문에 12번이 이어지는 것은 실행불가 테스트 케이스이다.

실행 불가능한 경우를 모두 제외하면 총 31,470개의 실행가능 조합 테스트 케이스를 구할 수 있다. 실행가능 테스트 케이스는 12, 17, 31번 조합이다. 앞서 설명한 12, 17번의 동작이 이루어지다가 “Load”가 “STOP”인 31번에 의해 모터가 정지하는 상황이다. 이는 운전자가 우회전 후 좌회전을

시도하는데, 외부 부하에 의해 핸들이 오른쪽으로 튄 후 바퀴가 걸려버리는 상황이다. 이처럼 세 개의 테스트 케이스를 조합하면 좀 더 복잡한 상황이 만들어진다.

세 개 이상의 테스트 케이스 순서에 영향을 받는 순서쌍은 극한 시험에 영향을 많이 주지 않는다.

4) 시나리오기반 테스트 케이스

테스트 목적에 대한 명확한 이해와 SUT에 대한 정확한 정보는 시나리오 테스트를 수행함에 있어서 필수적이다. 본 논문에서 수행하고자 하는 테스트의 목적은 파괴 시험을 통한 시스템의 안정성 및 신뢰도 검증이다.

이를 토대로 SUT에 충격을 주기 위한 몇 가지 시나리오를 생성할 수 있다. 예를 들어 운전자의 운전 미숙에 의해 핸들이 빠른 속도로 좌우로 흔들리는 시나리오나, 핸들은 고정된 상태에서 바퀴가 바위에 부딪혀 모터가 돌아가는 시나리오 등을 만들 수 있다. 이렇게 일상에서 발생할 수 있는 상황 이외에 SUT에 극한의 상황을 주어 신뢰성을 검증해 보기 위한 시나리오도 만들 수 있다. 예를 들어 핸들을 한쪽 끝으로 돌린 이후, 핸들을 살짝 풀었다가 다시 끝으로 세게 내려치는 동작을 빠르게 반복하는 시나리오가 있다. 극한 시나리오에 대한 내용을 5장에서 상세히 설명한다. 이러한 시나리오는 CTM을 이용해 생성한 55개의 기본 테스트 케이스를 이용해 구현한다.

5) 무작위 테스트 케이스

SUT에 대한 파괴 시험을 수행하기 위해 이중 조합, 삼중 조합 및 시나리오 기반 테스트 케이스를 생성한다. 하지만 다양한 테스트 이론에 입각하여 테스트 케이스를 생성하더라도, 완벽한 테스트를 수행한다고 보장할 수 없다.

테스트의 성능 향상을 위해 다양한 노력을 한 상태에서 추가적으로 할 수 있는 일은 가능한 입력들을 무작위로 생성하는 것이다. 따라서 마지막 테스트 케이스 생성 방식으로 무작위 테스트 방식을 사용한다.

5. 실 험

5.1 실험 환경 구축

Fig. 8은 테스트를 수행하기 위해 실제로 구성한 환경이다. 실제 테스트를 수행할 때는 대상이 되는 SUT 이외에 다양한 장비들이 존재한다. 먼저 부하 입력을 위한 모터가 SUT와 한 축으로 고정된다. 또한 부하 모터를 제어하기 위한 드라이버와 SUT의 입력을 위해 존재하는 시뮬레이터 및 테스트 수행기가 모여 테스트 환경을 구성한다.

Fig. 9는 테스트 환경을 구성하는 각 요소들의 관계 및 입출력 신호를 블록 다이어그램을 이용해 알기 쉽게 보여준다. 각 요소들과 사용되는 신호에 대한 상세한 설명은 다음과 같다.

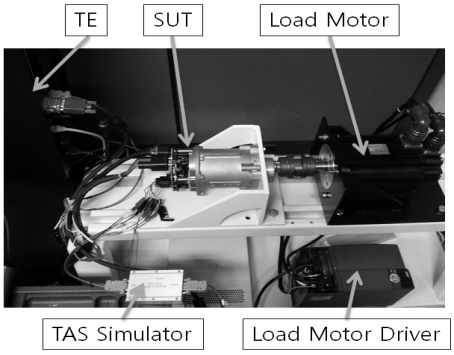


Fig. 8. Test environment

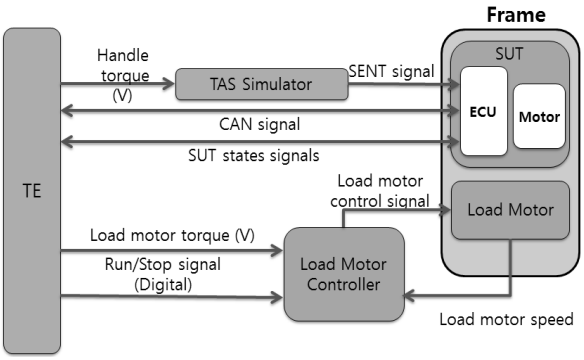


Fig. 9. Test environment block diagram

TE(Test Executer): NI(National Instrument)에서 제공하는 시간 결정성을 위한 실시간 운영체제(RTOS: Real Time Operating system)을 탑재한 NI-DAQ 카드를 장착한 장비이다. RTOS를 사용하기 때문에 정확한 시간에 신호를 수집하거나 생성할 수 있다. 사용하는 DAQ 카드의 종류에 따라 아날로그, 디지털 신호를 처리하거나, CAN, LIN 같은 통신도 할 수 있다. TE는 테스트 스크립트를 토대로 필요한 신호를 생성하여 테스트를 수행하고, 입출력을 로그파일로 남긴다.

- TAS Simulator: TE로부터 아날로그 전압을 입력받아 SENT 신호로 변환한다. 이때 SENT 신호란 차량의 핸들에 사용되는 TAS 센서의 출력 신호이다.
- Frame: SUT와 부하 생성을 위한 모터를 한 축으로 연결한다. 이를 통해 부하 모터를 제어하면 SUT에 원하는 부하를 인가 할 수 있다.
- ECU & Motor: 테스트 대상이 되는 MDPS 시스템으로 부하모터와 TE, TAS Simulator들로부터 입력을 받는다.
- Load Motor: SUT에 부하를 인가하기 위한 모터로, 빠른 반응속도와 큰 토크를 가지는 AC 서보모터를 사용한다.
- Load Motor Controller: 부하 모터 구동을 위한 드라이버이다. TE에서 부하 모터의 동작 방향과 토크 지령치를 입력 받아 부하 모터를 제어한다.

Table 3. Physical values of classes

Classification	Class	Value (V)	
Preposition	LL	Load torque	0
		Load Limit RPM	0.5
	CN	Load torque	0
		Load Limit RPM	10
	RL	Load torque	0
		Load Limit RPM	-0.5
TAS sensor	LMax	TAS	0
	LMed	TAS	1.5
	Stop	TAS	2.5
	RMed	TAS	3.5
	RMax	TAS	5
Load torque	Stop	Load torque	0(Left) / 0(Right)
		Load Limit RPM	0.5(Left) / -0.5(Right)
	H+	Load torque	(EPS torque * 0.9)/2.98*10
		Load Limit RPM	10
	L+	Load torque	(EPS torque * 0.2)/2.98*10
		Load Limit RPM	10
	H-	Load torque	(EPS torque * 1.9)/2.98*10
		Load Limit RPM	10
	L-	Load torque	(EPS torque * 1.2)/2.98*10
		Load Limit RPM	10

5.2 Test script 생성

생성된 테스트 케이스는 실제 SUT에 입력 가능한 적절한 값으로 변환시킨 후 사용해야 한다. 각 테스트 케이스는 SUT 사양 분석을 통하거나 테스트를 수행하면서 필요에 따라 수정하여 알맞은 값을 선택한다. Table 3은 사양 분석 이후에 각 클래스에 해당하는 심벌을 실제 값으로 표시한 것이다. 이 값을 테스트 케이스에 적용해 테스트 스크립트를 생성한다.

5.3 테스트 수행 및 결과 분석

테스트 스크립트를 실행하고 그 결과를 분석한다. 각 테스트 케이스의 조합을 연속적으로 입력하면, 이전 조합의 구동 결과가 시스템에 영향을 준다. 따라서 매번 SUT 초기화를 수행하고 테스트 하였다.

1) 조합 테스트 케이스 실행

Table 4는 테스트의 결과 분석과정을 설명하기 위한 몇 가지 입력 신호가 담긴 간단한 테스트 스크립트 예시이다.

Table 4. Double permutation test script sample

IG_key	TAS	Load	L_Limit	TIME	Comment
0	2.5	0	0	1000	Initialization Start
1	2.5	0	0	1000	IG & TAS on
1	2.5	0	0	1000	Remote on
1	2.5	0	0	1000	Load Run on
1	2.5	0	0	3000	Set Preposition
1	2.5	0	0	500	CAN on & Test Start
1	1.5	0.3	10	1000	Case 23 Lmed + L+
1	3.5	0	0.5	1000	Case 36 Rmed + STOP
0	2.5	0	0	1000	Test End

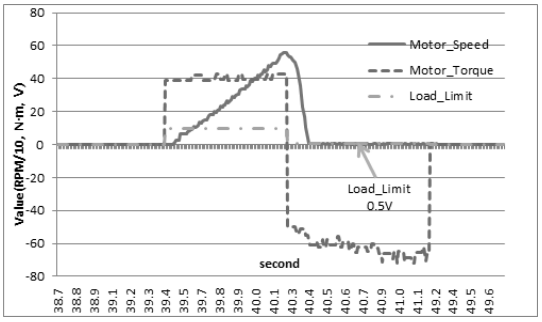


Fig. 10. Double permutation test analysis

55개의 기본 테스트 케이스 중 23, 36번 조합을 테스트한다. SUT 초기화 이후 수행되는 23번 테스트 케이스는 “Position”이 “CN”, “TAS input”이 “Lmed”, “Load”가 “L+”이다. 즉, 핸들을 왼쪽으로 약간 꺾고, 모터 반대 방향의 약한 부하가 걸린다. 따라서 23번 케이스를 실행하면 모터는 좌회전 할 것이다. 36번 테스트 케이스는 “Position”이 “CN”, “TAS input”이 “Rmed”, “Load”가 “STOP”이다. 즉, 핸들을 오른쪽으로 약간 꺾지만 정지부하가 걸린다. 따라서 36번 케이스를 실행하면 모터는 정지할 것이다. 이러한 추측을 통해 23, 36번 조합은 핸들을 왼쪽으로 꺾었다가, 원위치로 돌리는 도중 바퀴가 구멍에 빠지게 되어 움직이지 못하는 상황으로 볼 수 있다. 따라서 이 조합을 테스트하면 모터가 좌회전 하다가 멈추게 된다는 것을 짐작할 수 있다.

Fig. 10은 전체 테스트 로그 중에서 23, 36번 조합을 테스트하는 구간에서 SUT의 모터 스피드와 토크, 값을 나타낸다. 이때 모터의 속도단위 RPM을 10배 감쇠하여 두 신호를 보기 편하게 나타내었다. 로그 파일을 분석해 보면, 23번 케이스가 실행됨에 따라 SUT의 토크가 좌측(Motor_Torque 값이 양수)으로 걸리고, SUT가 좌측(Motor_Speed 값이 양수)으로 돌았다는 것을 확인할 수 있다. 이후 36번 케이스가 실행되면 SUT의 토크가 우측(Motor_Torque 값이 음수)으로 걸리지만, 부하모터에 의해 SUT가 회전하지 않는 모습(Motor_Speed 값이 0)을 확인할 수 있다. 이런 동작은 스크립트를 보고 예상했던 동작과 일치함을 알 수 있다. 이런 방식으로 테스트 중 실제 입력된 신호와 모터의 동작들을 모두 비교한 결과 SUT는 주어진 테스트 케이스 입력에 따라 정상적으로 동작하는 것을 확인하였다. 즉, SUT는 조합 테스트 케이스를 이용한 테스트에서 오류 발생 없이 정상적으로 동작하였다.

2) 시나리오 기반 테스트 케이스 실행

SUT에 대해 수행한 시나리오 테스트는 핸들 치기, 핸들 고속 조작, 외부의 힘 적용 등 다양한 시나리오가 존재한다. 이러한 시나리오들은 모두 기본 테스트 케이스를 이용하여 만들어졌다. 본 논문에서는 수행한 시나리오 테스트 중에서 극한의 입력을 주어 SUT의 신뢰성을 테스트한 “핸들 치기 시나리오”에 대해 설명한다.

Table 5는 운전자가 핸들을 왼쪽으로 최대한 꺾은 상태

에서 핸들을 살짝 풀었다가 다시 왼쪽으로 세게 꺾는 상황을 나타낸다. Table 5에 명시된 상황 1-5는 다음과 같은 동작을 테스트 케이스로 나타낸 것이다.

Table 5. Left extreme position test scenario

Situation	Position			TAS Input					Load torque				
	LL	CN	RL	LMax	LMed	Stop	RMed	RMax	Stop	H+	L+	H-	L-
1		o			o						o		
		o			o				o				
		o				o			o				
2	o						o				o		
		o				o			o				
3		o		o							o		
		o		o					o				
4	o						o				o		
		o				o			o				
5		o		o							o		
		o		o					o				

- Situation 1: 핸들을 좌측으로 끝까지 회전하는 동작
- Situation 2: 핸들을 우측으로 살짝 회전하는 동작
- Situation 3: 핸들을 좌측으로 강하게 회전하는 동작
- Situation 4: 핸들을 우측으로 살짝 회전하는 동작
- Situation 5: 핸들을 좌측으로 강하게 회전하는 동작

시나리오 테스트는 시나리오 1을 초기 값으로 수행한 이후 시나리오 2, 3을 반복 수행한다.

Fig. 11은 극한 시나리오 Table 5를 스크립트로 만들어 테스트를 수행한 로그 파일의 일부를 도식화 한 것이다. 이때 RPM값을 100배 감쇠하여 신호들을 알아보기 쉽게 하였다. 또한 로그 파일 안에 존재하는 입력 신호를 표시하여 직관적으로 SUT의 상태를 살펴보면서 시나리오 진행을 살펴볼 수 있도록 하였다. Fig. 11의 앞부분의 “Motor Speed”를 살펴보면 모터가 우측(음수)으로 회전하다가 정지(0 RPM)하는 것을 관찰할 수 있다. 이때의 입력 “TAS signal”이 3.5V로 “Rmed”인 것을 볼 때, 이 부분은 Table 5의 2번 상황에 해당하는 것을 알 수 있다. 이후 Fig. 11의 뒷부분의 “Motor Speed”를 살펴보면 모터가 좌측(양수)으로 회전하다가 정지하는 것을 관찰할 수 있다. 이때의 입력 “TAS signal”이 0V로 “Lmax”인 것을 볼 때, 이 부분은 Table 5의 3번 상황에 해당하는 것을 알 수 있다. 이는 핸들을 우측으로 살짝 풀었다가 왼쪽으로 세게 치는 동작이다. 이렇게 스크립트의 내용을 실행하면 SUT는 그에 맞는 동작을 정상적으로 하는 것을 로그 파일을 분석함으로써 알 수 있다. 또한 테스트 수행 중 육안 관측을 통해 시스템이 정상적으로 동작하는지 확인하고, 테스트 종료 후 간단한 입력을 인가해서 정상 상태 유무를 확인하였다. 이러한 행동을 통해 SUT는 시나리오 테스트에 대해 정상적으로 동작하는 것을 확인하였다.

3) 무작위 테스트 케이스 실행

시스템의 입력을 무작위로 선택하는 방법으로 테스트를 수행한다. 이때, 무작위 테스트에는 기본 테스트 케이스를 사용하지 않고 분류 트리의 클래스를 무작위로 선택하여 테스트한다. 또한 테스트를 시작할 때의 SUT의 초기 상태가 무엇인가도 중요한 요소이기 때문에 다양한 초기조건으로 무작위 테스트를 반복 수행하였다. 각 초기조건에 대해 테스트를 수행 할 때 SUT를 25분간 가동하여 충분히 예열하여 실제 차량과 비슷한 환경을 만든 이후 테스트를 진행하였다.

Fig. 12는 무작위 테스트의 로그 파일 내용 중 일부를 도식화 한 것이다. RPM의 경우 100배 감쇠하여 표시하였다. 그림을 총 7개의 구간으로 나눈 후, 각 구간을 분석하여 인가된 입력에 대한 예상 동작과 실제 모터의 속도를 비교함으로써 테스트의 결과를 확인한다. 이때 “Motor Speed”가 음수이면 우회전, 양이면 좌회전이다.

- 구간 ①
  - 입력: TAS 신호 1.5V는 “Lmed”, 부하 신호 - 6V는 “H-”이다. 이 입력은 핸들 조작과 반대 방향으로 돌부리에 걸린 것 같은 매우 강한 부하가 걸리는 것이다. 따라서 SUT는 부하에 의해 우회전 할 것이다.
  - 출력: “Motor Speed”가 음수(우회전)이며, 음의 방향으로 증가한다. 즉, 모터는 우회전하며 속도가 점차 빨라진다.

- 구간 ②
  - 입력: 부하 한계신호 0.5V는 부하 “STOP”을 의미한다. 따라서 SUT는 정지할 것이다.
  - 출력: “Motor Speed”가 0 RPM으로 유지된다. 즉, 모터는 ② 구간 동안 정지한 상태이다.

- 구간 ③
  - 입력: TAS 신호 0V는 “Lmax”, 부하 신호 - 1.5V는 “H+”이다. 이 입력은 핸들과 반대의 약한 부하가 걸리

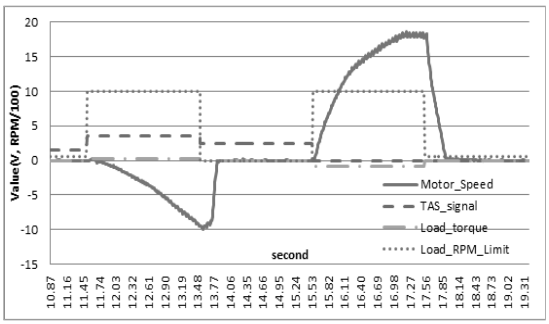


Fig. 11. Scenario test analysis (V ,RPM/100)

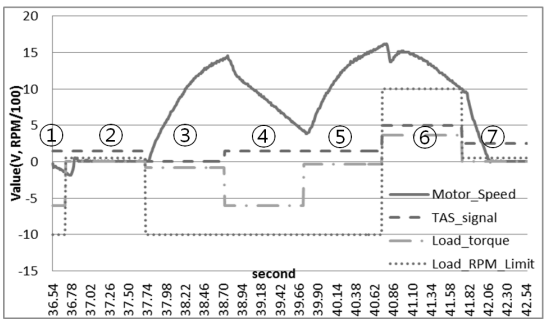


Fig. 12. Random test output

는 것이다. 따라서 SUT는 핸들 조작에 의해 좌회전 할 것이다.

- 출력: “Motor Speed”가 양수(좌회전)이며, 양의 방향으로 증가한다. 즉, 모터는 좌회전하며 속도가 점차 빨라진다.

• 구간 ④

- 입력: TAS 신호 1.5는 “Lmed”, 부하 신호 - 6V는 “H-”이다. 이 입력은 구간 ①과 같은 입력이기 때문에 SUT는 우회전 할 것이다.
- 출력: “Motor Speed”가 양수(좌회전)이고 크기가 줄어들고 있다. 즉, 모터는 좌회전하며 속도가 점차 느려진다. 이는 구간 ③의 영향이 남아 있는 것으로, 모터는 좌회전 하지만 입력 값에 알맞게 우측으로 토크가 걸려 속도가 줄어드는 것을 확인할 수 있다.

• 구간 ⑤

- 입력: TAS 신호 1.5V는 “Lmed”, 부하 신호 - 0.5V는 “L+”이다. 이 입력은 핸들과 반대의 매우 약한 부하가 걸리는 것이다. 따라서 핸들 조작에 의해 SUT는 좌회전 할 것이다.
- 출력: “Motor Speed”가 양수(좌회전)이며, 양의 방향으로 증가한다. 즉, 모터는 좌회전하며 속도가 점차 빨라진다.

• 구간 ⑥

- 입력: TAS 신호 5V는 “Rmax”, 부하 신호 3.6V는 “H+”이다. 이 입력은 핸들과 반대의 약한 부하가 걸리는 것이다. 따라서 SUT는 핸들 조작에 의해 우회전 할 것이다.
- 출력: “Motor Speed”가 양수(좌회전)이고 크기가 줄어들고 있다. 즉, 모터는 좌회전하며 속도가 점차 느려진다. 이는 구간 ⑤의 영향이 남아 있는 것으로, 모터는 좌회전 하지만, 입력 값에 알맞게 우측으로 토크가 걸리는 것을 확인할 수 있다.

• 구간 ⑦

- 입력: 부하 한계신호 0.5V는 부하 “STOP”을 의미한다. 따라서 SUT는 정지 할 것이다.
- 출력: “Motor Speed”가 0 RPM으로 유지된다. 즉, 모터는 ⑦ 구간 동안 정지한 상태이다.

①~⑦ 구간을 분석한 결과, 모든 구간에서 입력된 신호의 예상 동작과 같은 출력이 나오는 것을 확인하였다. 이처럼 각 구간의 입력신호와 출력신호를 분석하는 방법으로 모든 무작위 테스트의 로그 파일을 조사하여 무작위 테스트에 대한 SUT의 동작을 살펴보았다. 조사 결과 무작위로 생성된 다양한 입력에 맞추어 SUT가 정상적으로 동작하는 것을 확인하였다. 또한 테스트 종료 후에도 간단한 입력을 인가하여 SUT의 동작을 관찰하여 정상적으로 동작함을 확인하였다.

4) 정리

네 가지의 테스트(이중 조합, 삼중 조합, 시나리오, 무작위)를 진행하였다. 테스트 도중에는 육안관측을 통해 SUT가 파손이나 오동작 없이 정상적으로 구동하는 것을 확인하였다. 또한 로그 파일 분석을 통해 입력에 따른 SUT의 동작 또한 이상이 없음을 확인하였다.

본 연구의 목적이 파괴 시험을 통한 SUT의 안전성 검증에 목적이 있는 만큼 BLDC 모터가 고속 회전 중 부하 모터에 의해 정지하거나, 역방향으로 회전하는 등 강도 높은 테스트가 테스트 케이스 곳곳에 존재한다. 이러한 테스트 케이스가 시스템에 어떠한 영향을 주는지 측정하기 위해 모터의 출력 전압과 모터에 흐르는 전류를 측정하였다.

Fig. 13은 구동 중인 BLDC 모터의 U, V상 전압을 나타낸다. 파형의 전반부에서 정상 구동 중에도 역기전력(back-EMF)에 의한 리플(ripple)을 확인 할 수 있다. 이때 갑작스러운 부하나 핸들 조작에 의한 역회전 등 모터에 충격이 생기는 경우, 그림에 표시된 원 안쪽처럼 전압이 파형의 전반부의 리플과 비교해서 약간 상승하는 것을 확인할 수 있었다.(타원 부분 안) 이는 인버터 소자에 치명적인 크기는 아니지만, SUT의 일반적인 동작에서 지속적으로 나타는 것을 확인 할 수 있었다.

Fig. 14는 BLDC 모터에 흐르는 전류를 측정하기 위해 사용한 센서 회로 및 사양이다. 이 센서를 통해 1A의 전류는 10mV의 전압으로 변경되어 출력된다. 정상 구동의 전류 흐름을 살펴보고, 부하 모터에 의한 급속 정지, 역방향 회전 등 강도 높은 테스트 케이스를 실행 할 때의 전류 흐름을 살펴보았다.

SUT가 정상 동작할 때 BLDC 모터의 한 상에 흐르는 전류는 Fig. 15에서 확인할 수 있다. 전압이 ±1V의 사인파로 나타나기 때문에 모터에 흐르는 전류는 ±100A이다. 이러한 모습은 장시간의 운전에도 일정한 모습을 보인다. 이후 강도 높은 테스트 케이스에 대해 모터의 전류는 어떤 모습을 보이는지 확인한다.

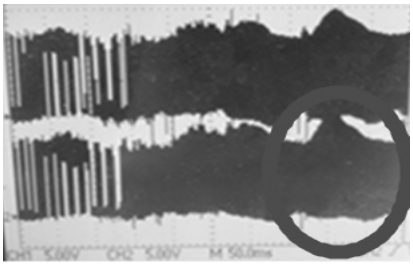
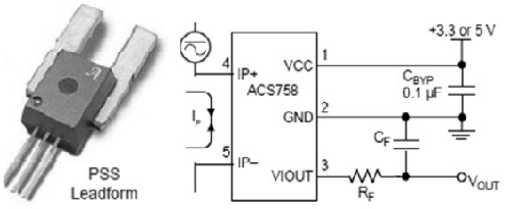


Fig. 13. U, V phase of BLDC motor



Part Number[1]	Package		Primary Sampled Current, I_P (A)	Sensitivity Sens (Typ.) (mV/A)	Current Directionality
	Terminals	Signal Pins			
ACS758ECB-200B-PSS-T	Straight	Straight	±200	10	Bidirectional

Fig. 14. Circuit and characteristics of current sensor

관측 결과 강도 높은 테스트 케이스를 실행 할 때, 시스템에 흐르는 전류에 변화가 있음을 확인할 수 있다. 핸들을 좌우로 빠르게 움직일 경우 모터는 한쪽 방향으로 회전하다가 역회전을 하게 된다. Fig. 16은 이와 같은 상황에서 모터에 흐르는 전류를 나타낸다. 약 0.3V의 전압 증가를 통해 역회전 시 실제 모터에 약 30A의 전류가 더 흐른다는 것을 알 수 있다. 또한 모터를 회전 시키다가 부하 모터를 이용해 강제로 정지시키는 경우 Fig. 17과 같은 전류 흐름을 관측 할 수 있다. 이 경우에도 전압이 약 0.4V 상승하는 모습을 통해 부하를 통한 강제 정지 시 모터에 약 40A의 전류가 더 흐르는 것을 알 수 있다. 이와 같이 강도 높은 테스트 케이스의 경우 시스템에 이상 고전압, 이상 고전류 등 의도하지 않은 충격을 줄 수 있다. 따라서 실제 차량에 적용되는 시스템의 경우 이러한 충격을 잘 견딜 수 있어야 한다.

이러한 충격에 대한 SUT의 안전성, 신뢰성 검증을 위해 CTM기반으로 고정 CTM 아이디어를 적용한 재구성된 트리를 이용해 생성한 테스트 케이스를 이용해 테스트를 수행하였다. 그 결과 다양한 입력에 대해 시스템은 정상적으로 동작하였고, 강도 높은 테스트 케이스에 대한 충격도 잘 견디는 것을 확인하였다.

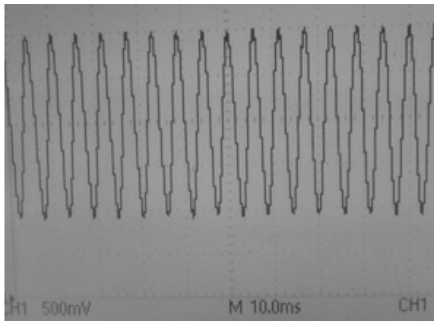


Fig. 15. Current flow - normal condition

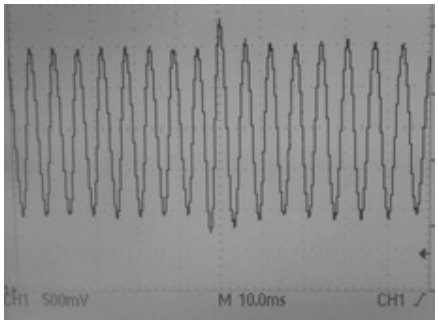


Fig. 16. Current - reversed rotation

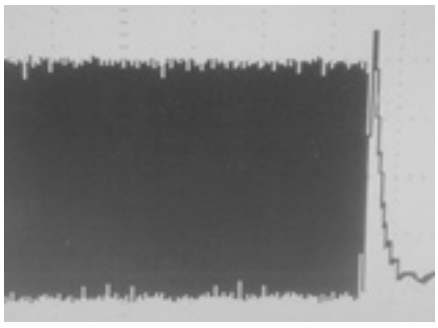


Fig. 17. Current - forced stop

6. 결 론

BLDC 모터의 사용 비중이 증대되는 만큼 BLDC모터가 사용되는 시스템의 안전성과 신뢰성에 대한 중요성이 커지게 되었다. 이러한 동향에 맞추어 실제 차량에 사용되는 MDPS 시스템을 대상으로 신뢰성 확보를 위한 테스트 방법에 대한 연구를 진행하였다. 이를 통해 수행한 테스트가 시스템의 안전성, 신뢰성을 검증할 수 있다는 이론적 근거를 확립하고자 하였다.

이를 위해 검증된 테스트 케이스 생성 방법인 CTM을 기초로 테스트 케이스 생성 전략을 연구하여 고정 CTM, CTM 재구성이라는 방법을 제안 하였다. 이들을 이용하면 복잡한 트리를 간략화 하거나, 특정 목적에 집중된 테스트 케이스를 생성할 수 있기 때문에 효율적인 테스트를 수행할 수 있다는 장점이 있다. 본 논문에서는 SUT를 분석하여 분류 트리를 생성한 후, 파괴 시험에 초점을 맞추어 CTM 재구성 방법을 사용해 실제 자동차에 사용되는 MDPS 시스템을 테스트 하였다. 이때 본 연구의 목적인 정상 구동중인 시스템에 대해 극한 테스트를 통한 신뢰성 검증을 위해 정상 입력을 위주로 분류 트리를 재구성 하였다.

이후 각 클래스를 조합하여 55개의 기본적인 테스트 케이스를 선별하였다. 기본 테스트 케이스를 “double permutation”, “triple pair”, “scenario test”, “random test”라는 네 가지 방법의 테스트 전략을 적용하여 새로운 테스트 케이스를 만들었다. 이후 생성한 테스트 케이스를 스크립트로 변환한 후 테스트를 수행하였다. 이때, 테스트를 수행하는 동안 생성하거나 수집한 데이터들은 모두 로그파일로 남겨 분석하였다.

동작하는 시스템에 대한 육안 관측, 테스트 종료 이후의 간단한 기능 테스트를 통한 시스템의 상태 파악, 테스트 로그파일을 통한 테스트 입력에 대한 시스템의 예상 동작과 실제 동작의 비교, 모터의 속도, 전류의 측정값 분석 등을 수행하였다. 이렇게 시스템의 내·외적인 요소를 분석한 결과 MDPS 시스템은 네 가지의 테스트에 대해 모두 이상 없이 수행된 것을 확인하였다.

또한 전압 및 전류 측정을 통해 수행한 테스트의 몇 가지 테스트 케이스가 시스템에 충격을 주는 것을 확인하였다. 따라서 수행된 네 가지 테스트의 무사통과라는 결과는 테스트 대상인 MDPS 시스템이 기능적으로 정상적이고 갑작스러운 충격에 대해 안전성과 신뢰성을 가지고 있음을 말해준다.

향후 연구로 분류 트리 구성 시, 시나리오에 필요한 클래스에 가중치를 주거나, 클래스에 입력 값 이외의 시나리오 정보를 담아 테스트 케이스를 생성할 때 참고하는 등 다양한 방법을 이용해 시나리오 테스트 케이스 생성 방법을 제안하기 위해 노력하고 있다. 자동으로 분류 트리로부터 테스트 시나리오를 생성할 수 있는 체계적이고 간단한 방법이나 도구에 대한 이론을 정립하고 구현하는 것도 향후 연구과제로 남아 있다.

Reference

[1] J. Y. Seo, A. Y. Sung, B. J. Choi and S. B. Kang, "Automating Embedded software Testing on an Emulated Target Board," Proc. of the Second International Workshop on Automation of Software Test, p.9, August, 2007.

[2] S. Y. Jeong, Y. W. Chang and C. J. Yoo, "Test Case Genration Technique Based on State Transition Model for Embedded System," Journal of Korean Institute of Information Technology, Vol.9, No.4, pp.11-21, 2011.

[3] Walter J. Gutjahr, "Partition Testing vs. Random Testing: The Influence of Uncertainty", IEEE Trans. software Eng., Vol.25, No.5, pp.661-674, September, 1999.

[4] E. J. Weyuker and Bingchiang Jeng, "Analyzing Partition Testing Strateges," IEEE Trans. software Eng., Vol.SE-17, pp.703-711, July, 1991.

[5] Grochtmann, M., Grimm, K., "Classification Trees for Partition Testing," Software Testing, Verification & Reliability, Vol.3, No.2, pp.63-82, June, 1993.

[6] Ostrand, T., Balcer, M., "The Category-Partition Method for Specifying and Generating Functional Tests," Communications of the ACM, Vol.31, No.6, pp.676-686, June, 1988.

[7] Grochtmann, M., "Test Case Design Using Classification Trees," Proceedings of STAR '94, Washington, D.C, pp.93-117, May, 1994.

[8] Rowe, Alexander, G. Sen Gupta, and Serge Demidenko. "Instrumentation and control of a high power BLDC motor for small vehicle applications," Instrumentation and Measurement Technology Conference, 2012 IEEE International, pp.559-564, May, 2012.

[9] VISHAY SILICONIX, "Power MOSFET Failures in Automotive Applications," Application Note 910, Apr., 2009.

[10] R. L. A. Ribeiro, C. B. Jacobina, E. R. C. da Silva and A. M. N. Lima, "Fault Detection of Open-Switch Damage in Voltage-Fed PWM Motor Drive Systems," IEEE Trans. Power Electron., Vol.18, No.2, pp.587-593, Mar., 2003.

[11] S. Bolognani, M. Zigliotto and M. Zordan, "Innovative Remedial Strategies for Inverter Faults in IPM Synchronous Motor Drives," IEEE Trans. Energy Conversion, Vol.18, No.2, pp.306-312, June, 2003.

[12] P.L Poon, T.Y Chen, T.H. Tse, "Choices, Choices: Comparing between CHOC'LATE and the Classification-Tree Methodology," 17th Ada-Europe International Conference on Reliable Software Technologies, Stockholm, Sweden, pp.11-15, June, 2012.

[13] Cem Kaner, J. D, "An Introduction to Scenario Testing," Florida Tech, June, 2003.

[14] M. Grochtmann, K. Grimm, J. Wegener, "Tool-Supported Test Case Design for Black-Box Testing by Means of the Classification-Tree Editor," Proc. of EuroSTAR '93, pp.169-176, 1993.

[15] T.Y. Chen, P.L. Poon, "Construction of classification trees via the classification hierarchy table," Information and Software Technology, Vol.39, No.13, pp.889-896, 1997.

[16] T. Y. Chen, P. L. Poon, T. H. Tse, "An integrated classification-tree methodology for test case generation," International Journal of Software Engineering and Knowledge Engineering, Vol.10, No.6, pp.647-679, 2000.

[17] A. Pretschner, O. Slotosch, E. Aiglstorfer, S. Kriebel, "Model-based testing for real," International Journal on Software Tools for Technology Transfer, Vol.5, No.2-3, pp.140-157, March, 2004.

[18] M. Conrad, I. Frey, S. Sadeghipour "Systematic Model-Based Testing of Embedded Automotive Software," Electronic Notes in Theoretical computer Science, Vol.111, pp.13-16, 2005.

[19] Conrad, M., Dörr, H., Fey, I., Yap, A., "Model-based Generation and Structured Representation of Test Scenarios," Workshop on Software-Embedded Systems Testing (WSEST), Gaithersburg, USA, November, 1999.

[20] http://www.pairwise.org

신 재 혁

e-mail : shinmre@naver.com

2012년 아주대학교 전자공학과(학사)

2014년 아주대학교 전자공학과(석사)

2014년~현 재 LG전자 VC사업부 연구원

관심분야 : 시스템 소프트웨어, 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