논문 2014-09-38

가상 프로토타입 기반 임베디드 소프트웨어의 테스트 기법

(A Testing Technique based on Virtual Prototype for Embedded Software)

류 호 동, 정 수 용, 이 성 희, 김 지 훈, 박 흥 준, 이 승 민, 이 우 진*
(Hodong Ryu, Sooyong Jeong, Sunghee Lee, Jihun Kim, Heungjun Park, Seungmin Lee, Woo Jin Lee)

Abstract : Recently, software reliability and safety issues are seriously considered since failures of embedded systems may cause the damages of human lifes. For verifying and testing embedded software, execution environment including sensors and actuators should be prepared in the actual plants or virtual forms on PC. In this paper, we provide the virtual prototype based code simulation techniques and testing framework on PC. Virtual prototypes are generated by combining the Adobe's Flash SWF images corresponding to the state machine of HW or environment components. Code simulation on PC is possible by replacing the device drivers into virtual drivers which connect to virtual prototypes. Also, testing is performed by controlling the states of virtual prototype and simulators. By using these tools, embedded software can be executed in the earlier development phase and the efficiency and SW quality can be enhanced.

Keywords : Automated testing, Virtual prototype, Embedded software, Code simulation

Ⅰ. 서 론

대부분의 임베디드 시스템들은 실생활과 밀접하게 연계되어 사용되므로 조그만 오류가 발생하더라도 인명 및 재산상의 심각한 손상을 끼칠 수 있기 때문에 임베디드 시스템 또는 소프트웨어의 신뢰성 문제는 지금까지 꾸준히 연구되는 분야이다. 특히 최근에는 하드웨어적인 오류보다는 소프트웨어적인 오류로 인한 문제가 더욱 조명을 받고 있다.

기존의 임베디드 소프트웨어의 신뢰성에 대한 연구로는 크게 소스 코드 중심의 연구와 시스템 모델 중심의 연구로 나뉘어 볼 수 있다. 소스 코드 중심의 연구로는 소스 코드를 분석하여 오류를 찾는 방법으로 정적 분석(static analysis) 기법[1-4]과 특정 시나리오를 직접 수행하는 임베디드 소프트웨어 테스팅 기법[5-9]이 주로 사용된다. 모델 중심의 연구는 요구사항 및 설계 단계에서 코드 분석보다는 좀더 높은 추상화 단계에서 시스템 모델을 작성하고 이를 바탕으로 다양한 검증을 수행하며, 주로 사용되는 검증 기법으로는 모델 검사(model checking)[10, 11] 등이 있다.

이 중에서 소스 코드에 적용 가능한 검증 방법으로 시나리오 기반의 테스팅 기법이 가장 널리 사용되고 있다. 일반적으로 임베디드 소프트웨어의 테스팅을 진행하기 위해서는 플랜트, 센서, 구동기 등의 임베디드 시스템의 운영 환경을 구축하여야 한다. 이러한 운영 환경이 제공되지 않으면 임베디드 소프트웨어 수행이 불가능하며 또한 테스팅 수행도 불가능하다. 이러한 임베디드 소프트웨어를 수행하기 위해서는 실제 수행환경 또는 PC 상의 가상 수행환경이 필요하다. 실제 수행환경은 개발 완료 시점에 적용 가능함으로 소프트웨어 개발 단계에서 적용하기 어려운 점이 있으며 또한 실제 환경에서 검증하기 어려운 운행 시나리오도 있기 때문에 PC

*Corresponding Author(woojin@knu.ac.kr)
Received: 5 Nov. 2014, Revised: 24 Nov. 2014, Accepted: 26 Nov. 2014.
H. Ryu, S. Jeong, S. Lee, J. Kim, H. Park, S. Lee, W.J. Lee: Kyungpook National University
※ 본 논문은 2014년도 정부(교육부)의 재원으로 한국과학재단의 지원을 받아 수행된 기초연구사업(No. NRF-2014R1A1A2058733)과 2013년도 경북대학교 학술연구비에 의하여 연구되었음.

 2014 Dec.: 9(6) 307-314
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2014.9.6.307

상의 가상 수행 환경을 만들어 사용하는 것이 효율적이다. 가상 수행환경의 예로는 Android Phone 에뮬레이터인 AVD(Android Virtual Device)[12], 자동차 분야의 AUTOSAR 소프트웨어 개발지원 도구인 Vector 사의 CANoe 도구[13], 항공 분야의 SiL(Software-in-the-Loop) 시뮬레이터 등이 있으며 이들은 특정 분야의 시스템에 맞춤형으로 구현되어 범용성이 없어 타 분야의 일반적인 임베디드 시스템 개발에 활용하기 어렵다. 또한 기존의 가상 환경들은 모두 프로그래밍 언어 기반으로 제작되었으며 특정 분야에만 적용 가능하여 범용성이 떨어지는 단점이 있다. 다양한 임베디드 분야의 적용할 수 있는 PC 기반 가상 수행환경을 구축하기 위해, 프로그래밍 언어로 구축하는 것이 아니라 상태도 기반으로 생성하는 방법을 고려할 수 있다.

이 연구에서는 임베디드 시스템의 주변 환경 및 하드웨어에 대한 가상 프로토타입을 손쉽게 생성할 수 있는 방법을 제시하고 이를 기반으로 임베디드 소프트웨어의 시뮬레이션 기법 및 자동 테스팅 기법을 제안한다. 가상 프로토타입 생성 방법은 상태 다이어그램으로 환경 구성요소나 하드웨어의 행위를 모델링하고 상태별로 Adobe Flash 이미지를 제작하여 매핑하고 상태도 정보와 매핑된 이미지를 합성하여 하나의 Adobe Flash 이미지를 만든다. 이러한 매핑-합성 과정을 통한 가상 프로토타입 생성은 GUI 프로그래밍보다 훨씬 수월할 뿐만 아니라 기능 변경 및 보완 등의 유지보수가 아주 쉽다. 가상 프로토타입이 완성되면 모델 및 코드 시뮬레이터와 메시지 브로커로 연동되어 모델 단계의 시뮬레이션뿐만 아니라 소스 코드 단계의 시뮬레이션도 가능하다. 가상 프로토타입의 궁극적인 목적은 테스팅과 연계하여 테스트 환경을 구축하는 것이며 이 연구에서는 가상 프로토타입 기반 테스트 환경을 제공한다. 일반적으로 임베디드 소프트웨어를 테스팅하기 위해서는 특정 테스트 상황을 설정하여야 하는데, 실제 환경에서는 이러한 환경을 만들어 주는 것이 어려운 상황이 존재하는데 반해, 가상 환경에서는 가상 프로토타입의 상태 변수 등의 값을 설정하면 된다.

본 논문은 다음과 같이 구성된다. 2장에서는 가상 프로토타입 관련 연구를 다루며, 3장에서는 Adobe Flash 기술을 이용하여 가상 프로토타입을 개발하는 방법을 다룬다. 4장에서는 VP와 소스코드를 연동하여 시뮬레이션하는 방법을 다루며 5장에서는 가상 프로토타이핑(Virtual Prototyping; VP) 임베디드 소프트웨어 테스트 기법을 다룬다. 6장에서는 Lego MindStorms으로 엘리베이터 프로그램을 작성하여 이를 VP 기반 테스팅 도구로 검증하는 사례를 기술한다. 마지막으로 7장에서는 결론을 맺는다.

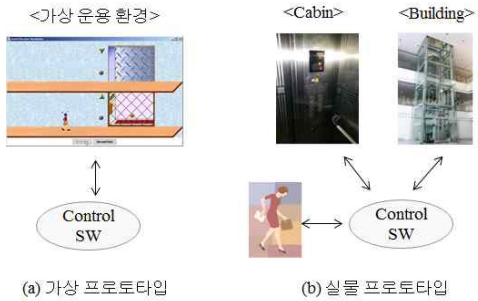

그림 1. 가상 및 실물 프로토타이핑 환경에서의 임베디드 소프트웨어 수행
Fig. 1 Embedded Software Execution in Virtual and Physical Prototype Environment

Ⅱ. 가상 프로토타입 관련 연구

일반적으로 임베디드 소프트웨어의 테스팅을 진행하기 위해서는 플랜트, 센서, 구동기 등의 임베디드 시스템의 운영 환경을 구축하여야 한다. 이러한 운영 환경이 제공되지 않으면 임베디드 소프트웨어 수행이 불가능하며 또한 테스팅 수행도 불가능하다. 그림 1은 가상 및 실물 프로타이핑 기법으로 엘리베이터 시스템의 운영 환경을 구축한 예를 보여준다.

그림 1(a)는 가상 프로토타이핑 기법으로 PC 상에서 건물, Cabin, 사람 등을 가상화하여 생성한 것이며 그림 1(b)는 실물 프로토타입으로 운영 환경을 구축한 것이다. 가상 프로토타입은 실물 프로토타이핑에 비해 상대적으로 구축 비용이 적게 들고 프로젝트 초기부터 만들어 사용할 수 있는 장점이 있다. 하지만 프로그래밍 언어만으로 가상 프로토타입을 구축하는데는 많은 노력과 비용이 소요되며 경우에 따라서는 제어 소프트웨어 개발보다 더 많이 드는 경우도 발생한다. 가상 프로토타이핑 도구의 예로는 Android Phone 에뮬레이터인 AVD(Android Virtual Device)[12], 자동차 분야의 AUTOSAR 소프트웨어 개발지원 도구인 Vector 사의 CANoe 도구[13], 항공 분야의 SiL(Software-in-the-Loop) 시뮬레이터 등과 같이 특정 도메인의 보편적인 환경에 맞춰져 있으며 대부분 상용으로

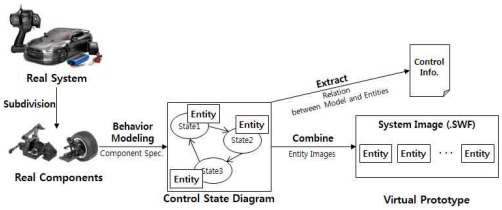

그림 2. 가상 프로토타입 생성 개념
Fig. 2 The concept of virtual prototype generation

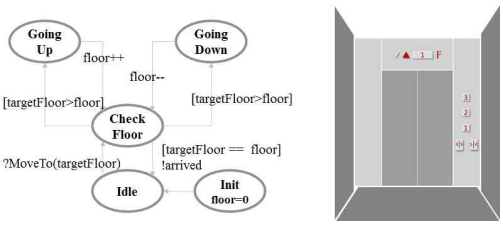

그림 3. 캐빈의 상태도 및 VP 이미지
Fig. 3 Cabin's state diagram and VP image

고가에 판매되고 있다. 하지만 전혀 새로운 임베디드 시스템을 개발하거나 특화된 임베디드 시스템을 개발할 경우, 가상 프로토타입 환경이 존재하지 않아 새롭게 구축하여야 하며 이는 또 다른 프로젝트로 간주될 만큼 어려운 작업이 될 수도 있다.

Ⅲ. SWF 기반 가상 프로토타입 생성

본 논문에서는 Lego MindStorms NXT[14]을 이용하여 지상 1~3층 사이를 운행하는 엘리베이터 시스템을 간단하게 제작하고 이를 PC 상에서 가상 운영 환경을 구축하여 테스트를 진행하는 방법을 제시한다.

SWF(Shockwave Flash) 기반 가상 프로토타입 생성 방법[15]은 그림 2와 같다. 먼저 만들고자 하는 대상 하드웨어 또는 환경요소의 상태도를 작성하고 상태도의 상태별로 Adobe Flash 이미지를 매핑하여 합성하는 과정을 거친다. 가상 프로토타입의 상태도는 시뮬레이터와의 통신 메시지를 정의할 뿐만 아니라 SWF 이미지를 합성하는 기준이 되므로 각 가상 프로토타입의 행위를 파악하여 초기 상태도를 정확하게 기술하는 것이 중요하다.

MindStorm 엘리베이터의 주요 가상 프로토타입 대상으로 엘리베이터 캐빈, 건물, 캐빈내 버튼, 층별 버튼 등이 있다. 이중 캐빈의 상태도는 그림 3의 왼

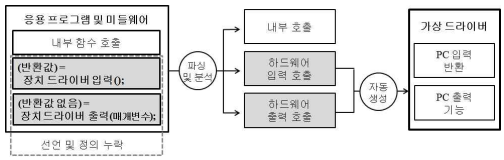

그림 4. 가상 드라이버 설정
Fig. 4 Virtual driver setting

쪽과 같으며 전체 생성된 가상 프로토타입 이미지는 그림 3의 오른쪽에 나타나 있다. 가상 프로토타입은 객체와 유사하게 내부적으로 상태변수를 가지고 있다. 그림 3의 상태도에서 floor라는 상태변수는 캐빈이 어느 층에 있는지를 나타낸다. 상태도는 소스 코드와의 상호작용을 나타내는 역할도 겸하고 있는데, 받는 API는 '?' 로 시작하고 보내는 API는 '!'로 시작한다. 상태도에서 "?MoveTo(targetFloor)" API는 코드 부분에서 호출하는 API로 targetFloor를 설정하며 "!arrived"는 소스 쪽으로 도착했음을 알리는 이벤트를 전달하기 위함이다. 또한 상태변수는 상태가 바뀔 때마다 변경되며 이 상태변수 값을 조회 및 변경하는 API가 제공된다. 이러한 API들은 테스팅 도구에서 주로 활용된다.

Ⅳ. 가상 드라이버 기반 코드 시뮬레이션

1. 가상 드라이버 설정

임베디드 소프트웨어는 센서 및 구동기와 연동하여 동작하므로 이러한 하드웨어들은 디바이스 드라이버에 의해 연동된다. 그리고 대부분의 임베디드 시스템은 하위에 디바이스 드라이버가 있으며 그 위에 제어 프로그램이 존재한다. PC 상에서 시뮬레이션을 위해서는 하드웨어 부분은 VP로 바뀌며 디바이스 드라이버는 VP와 연동되도록 가상 드라이버로 교체된다. 그림 4는 가상 드라이버를 설정하는 과정을 보여준다. 먼저 소스 코드에서 디바이스 드라이버를 제외한 어플리케이션 소스를 분석한다. 소스에서 누락된 디바이스 드라이버 함수의 헤더 정보를 분석하여 VP와 연동되는 코드로 가상 드라이버 함수의 몸체 부분을 작성한다.

2. VP 기반 코드 시뮬레이션 기법

제어 소프트웨어는 주기에 따라 수행되는 반복 루틴이므로 VP 기반 시뮬레이션을 위해 추가되는 코드는 탐침효과(probe effect)의 부작용을 최소화하여야 한다. 그림 5의 왼쪽 부분은 기본적인 제어

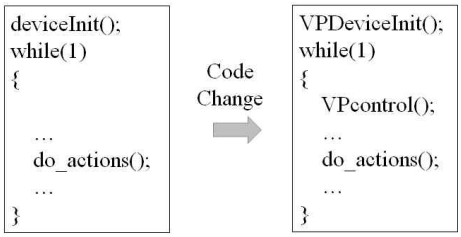

그림 5. 시뮬레이션을 위한 코드 변경
Fig. 5 Code change for simulation

프로그램의 구조를 보여준다. 시스템을 시작하면 먼저 센서 및 구동기 장치들을 초기화하고 while()문 내에서 주기에 따라 반복적인 작업을 수행한다.

그림 5의 오른쪽은 시뮬레이션을 위해 추가되는 코드를 보여준다. 변형된 VPDeviceInit() 함수는 deviceInit() 함수를 대신하여 가상 드라이버를 초기화하는 것으로 대체된다. 그리고 VPcontrol() 함수는 주기에 따라 가상 프로토타입의 값을 변경하고 가상 드라이버에 값을 입출력하는 기능을 갖는다. 또한 시뮬레이션의 시작, 정지 등의 제어 기능을 갖는다. 시뮬레이션을 위해 추가 또는 변경되는 코드는 위 두 개의 함수이며 실제 메인 주기에 영향을 미치는 부분은 VPControl() 함수 하나밖에 없으므로 기존 코드의 변경을 최소화하였다.

VPcontrol() 함수는 소스 코드와 가상 프로토타입과 상호작용 기능을 담당하지만 가상 프로토타입과 직접적으로 연결되어 동작하지 않는다. 그림 6은 코드 시뮬레이터와 가상 프로토타입이 메시지 브로커를 통해 연동되는 매카니즘을 보여준다. 메시지 브로커는 코드 시뮬레이터와 가상 프로토타입간의 메시지 전송을 담당하며 ZeroMQ의 Pub-Sub 패턴[16]을 활용하여 구현한다. 코드 시뮬레이터는 하나지만 여러 가상 프로토타입들이 존재할 수 있으므로 가상 프로토타입의 등록 및 해제를 할 수 있어야 한다. 코드 시뮬레이터의 메시지는 모든 가상 프로토타입에 동시에 뿌려지며 또한 가상 프로토타입의 메시지는 코드 시뮬레이터 및 다른 가상 프로토타입에게도 전송된다.

V. VP 기반 테스팅 기법

소프트웨어 테스팅은 소스 코드 수행을 전제로 하고 있으므로 앞서 제시한 VP 기반 코드 시뮬레이션은 궁극적으로 PC상에서 테스팅을 진행하기 위해서이다. VP 기반 테스팅은 각 테스트 케이스의 수

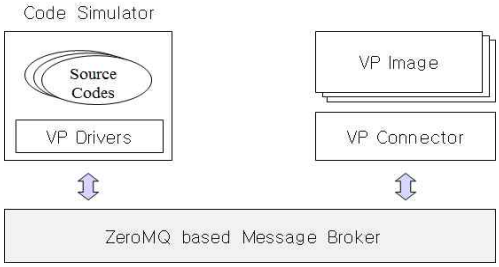

그림 6. VP 기반 코드 시뮬레이터의 구조
Fig. 6 Structure of VP-based code simulator

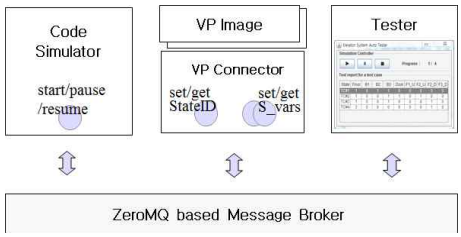

그림 7. VP 기반 테스트 수행기 구조
Fig. 7 Structure of VP-based test executor

행조건을 VP 환경에 설정하고 시뮬레이터를 실행한 후에 다시 VP의 상태변수 값을 검색하여 수행이 제대로 되었는지를 확인한다.

테스팅을 진행하기 위해서는 테스터가 가상 프로토타입과 시뮬레이터를 제어할 수 있어야 한다. 그림 7은 VP 기반 테스터의 구조를 보여준다. 일반적으로 테스팅은 사전조건 설정, 테스트 행위 수행, 결과값 확인 과정을 거친다. 이 중에서 사전조건 설정과 결과값 확인은 가상 프로토타입을 통해서 이루어지며, 테스트 행위 수행은 시뮬레이터를 통한다. 테스팅을 위해 코드 시뮬레이터가 제공하는 API로는 시뮬레이션을 제어하는 start, stop, pause, resume 등이 있다. 그리고 각각의 VP 연결자가 제공하는 API에는 VP의 상태를 확인 및 설정하는 set/getStateID()와 VP가 내부적으로 가지는 상태변수들의 값을 확인 및 설정하는 set/getS_vars() 등이 있다. 각각의 테스트 케이스는 설정함수인 set API 함수를 이용하여 사전 조건 설정을 진행하고 get API 함수를 이용하여 결과값을 검사한다.

예를들어, 1층에서 엘리베이터를 타고 3층으로 가는 상황을 테스트하려면, 테스트 케이스에서 캐빈 VP의 floor 상태변수를 1로 설정하고 캐빈내의 3층

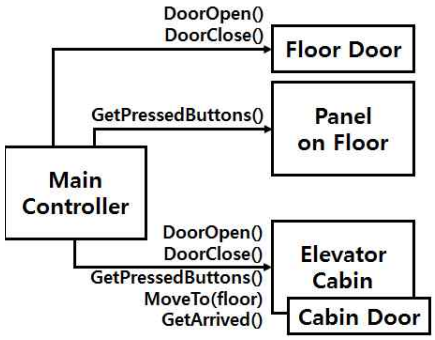

그림 8. 엘리베이터의 상황도
Fig. 8 Context diagram of elevator

표 1. 테스트 케이스 작성 예
Table 1. Example of test case description

	Elevator Cabin					Floor Buttons			
State	Floor	B1	B2	B3	Door	F1U	F2U	F2D	F3D
TC#1	1	0	1	1	0	0	0	0	0
TC#2	1	0	0	1	1	0	1	0	0
TC#3	1	0	0	1	0	0	0	1	0
TC#4	3	0	0	0	0	0	0	1	0

버튼을 나타내는 상태변수를 켜진 것을 나타내는 '1'로 설정한 상황에서 시뮬레이터를 가동하면, 엘리베이터의 VP가 1층에서 3층으로 이동하여 3층에서 문이 제대로 열리는지를 확인할 수 있다. 자세한 테스트 케이스 기술 예는 6장의 표 1에 나타나 있다.

Ⅵ. 테스팅 사례 적용

이 장에서는 VP 기반 코드 시뮬레이션 기술과 테스팅 기술의 효용성을 보이기 위해 Lego MindStorm NXT로 단순히 제작된 엘리베이터 시스템에 적용하는 사례 연구를 보인다.

1. 엘리베이터 시스템의 상황도

엘리베이터 시스템은 그림 8의 엘리베이터의 상황도(context diagram)과 같이 메인 제어기, 엘리베이터 케빈, 층별 문, 층별 버튼 등으로 구성되어 있으며 각 구성요소간의 상호작용을 API로 나타내고 있다. 메인 제어기에 임베디드 소프트웨어가 배치되며 나머지 Floor Door, Panel on Floor,

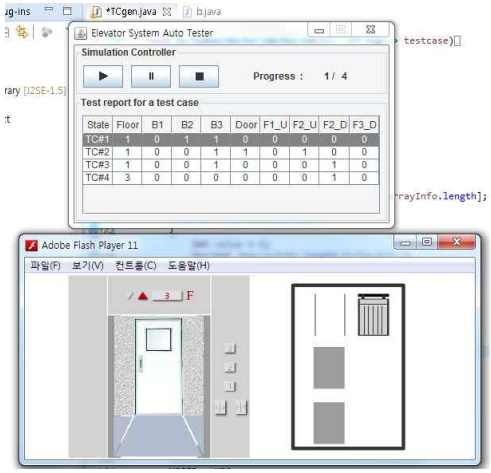

그림 9. 표 1의 테스트 케이스 1번 적용화면
Fig. 9 Snapshot of test case #1 execution in Table 1

Elevator 등은 플랜트 요소를 나타내므로 가상 프로토타입으로 제작되어 메인 제어기와 연동되어 동작된다.

2. 정상 및 비정상 테스트 시나리오

테스트 케이스 작성은 가상 프로토타입에 set API를 통해 설정되는 값을 지정하는 것이다. 표 1은 4개의 테스트 케이스의 작성 예를 보여준다. 먼저 TC#1은 1층에서 엘리베이터를 타고 2층과 3층을 누른 상황을 나타내며 TC#2는 1층에서 엘리베이터를 타고 3층을 누른 상황이며 아직 Door가 열려있는 상황에 2층에서 올라가는 버튼을 누른 상황을 나타낸다. TC#3은 1층에서 3층으로 올라갔다가 2층에서 아래로 내려가는 버튼이 누른 시나리오를 나타내고 있다. TC#4는 비정상 시나리오로 TC#3을 수행하면 2층에 엘리베이터가 있으므로 제어기는 2층에 캐빈이 있는 것으로 알고 있지만 VP에서는 3층으로 캐빈의 위치를 변경하여 실제위치와 인식위치가 다른 비정상 상황을 만든 다음, 2층에서 내려가는 버튼이 누른 상황을 검사하는 테스트 시나리오이다. VP를 활용하면 이와 같은 비정상 상황도 쉽게 설정할 수 있는 장점이 있다.

그림 9는 표 1의 테스트 케이스 1번을 적용한 시나리오의 수행화면을 캡쳐한 것을 보여준다. 왼쪽 VP 이미지는 캐빈을 나타내며 오른쪽 VP 이미지는 층별 문 상태와 엘리베이터가 해당 층에 있는지를 나타낸다. 테스트 케이스 1의 경우 작성된 제어 프

로그램이 정상적으로 수행되어 캐빈이 2층, 3층에 도달하여 문이 정상적으로 열렸다 닫혔으며 캐빈 실내 층버튼 램프도 제대로 꺼지는 것을 확인할 수 있었다.

그러나 표 1의 테스트 케이스 4번을 수행한 경우에는 실제 캐빈은 3층에 있지만 제어기는 2층에 있는 것으로 인지하여 2층의 Floor Door를 열고 캐빈은 3층에서 문이 열리는 비정상 현상이 발생하였다. 이는 제어 소프트웨어가 정상 상황만을 가정하여 작성된 것이어서 제어 소프트웨어에서 실제 엘리베이터의 위치를 확인하여 동기화하는 부분을 추가하여 비정상 상황이 발생하지 않게 수정하였다.

Ⅶ. 결론

본 논문에서는 PC 상에서 임베디드 소프트웨어를 시뮬레이션하고 테스팅하기 위한 가상 수행 환경 구축 방법을 제안하였다. 그리고 가상 드라이버를 활용한 VP 기반 코드 시뮬레이션 기법과 가상 프로토타입과 시뮬레이터를 제어하여 테스트를 진행하는 방법을 기술하였다. 그리고 이러한 방법의 활용성을 보이기 위해 Lego MindStorms 엘리베이터 소프트웨어의 시뮬레이션 및 테스트 진행 사례를 보였다. 이러한 VP 기반 임베디드 소프트웨어 개발 환경을 통해 개발 초기 단계에 소프트웨어를 수행할 수 있으므로 소프트웨어 개발의 효율성 및 소프트웨어 자체의 신뢰성을 높이는데 많은 도움을 줄 수 있다.

현재 사용된 테스트 도구는 입력된 값을 자동으로 설정하지만 VP를 보고 테스터가 수행결과를 확인하는 형태로 되어 있으므로 이러한 부분을 자동화하는 연구가 필요하다. 또한 현재 상태도는 이산(discrete) 이벤트 중심으로 환경 요소들을 추상화하여 사용하고 있어, 연속(continuous) 속성을 가지는 환경 요소를 좀더 정확하게 나타내는 데는 어려움이 있다. 이를 위해서는 timed 오토마타와 같은 시간 속성 또는 하이브리드 모델 방법 등으로 확장하는 연구가 필요하다.

References

[1] N. Ayewah, D. Hovemeyer, J.D. Morgenthaler, J. Penix, W. Pugh, “Using Static Analysis to Find Bugs,” IEEE Software, Vol. 25, No. 5, pp. 22-29, 2008.

[2] W. Schilling, M. Alam, “A methodology for quantitative evaluation of software reliability using static analysis,” Proceedings of Annual Reliability and Maintainability Symposium, pp. 399-404, 2008.

[3] T. Reinbacher, J. Brauer, M. Horauer, B. Schlich, “Refining assembly code static analysis for the Intel MCS-51 microcontroller,” Proceedings of IEEE International Symposium on Industrial Embedded System, pp. 161-170, 2009.

[4] D. Melski, T. Teitelbaum, T. Reps, “Static Analysis of Software Executables,” Proceedings of Conference For Homeland on Cybersecurity Applications & Technology, pp. 97-102, 2009.

[5] Y. Yin, B. Liu, “A Method of Test Case Automatic Generation for Embedded Software,” Proceedings of International Conference on Information Engineering and Computer Science, pp. 1-5, 2009.

[6] H. Gross, P.M. Kruse, J. Wegener, T. Vos, “Evolutionary White-Box Software Test with the EvoTest Framework: A Progress Report,” Proceedings of International Conference on Verification and Validation Workshops, pp. 111-120, 2009.

[7] T. Kanstren, “A Study on Design for Testability in Component-Based Embedded Software,” Proceedings of International Conference on Software Engineering Research, Management and Applications, pp. 31-38, 2008.

[8] H. Qian, C. Zheng, “A Embedded Software Testing Process Model,” Proceedings of International Conference on Computational Intelligence and Software Engineering, pp. 1-5, 2009.

[9] X. He, “Embedded systems based modular test automation,” Proceedings of International Colloquium on Computing, Communication, Control and Management, pp. 83-86, 2009.

[10] T. Reinbacher, M. Kramer, M. Horauer, B. Schlich, "Motivating Model Checking of Embedded Systems Software," Proceedings of IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, pp. 546-551, 2008.

[11] Z. Gu, K.G. Shin, "Model-checking of component-based event-driven real-time embedded software," Proceedings of IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 410-417, 2005.
[12] Android, "Managing Virtual Devices," http://developer.android.com/tools/devices/index.html
[13] Vector, "ECU Developement & Test - Vector," http://vector.com/vi_canoe_en.html
[14] MindStorms NXT, Lego MindStorms NXT robotics kit, http://www.nxtprograms.com/
[15] S.Y. Jang, J. Kim, W.J. Lee, "Development of SWF Based Virtual Prototyping Framework for Simulating Ubiquitous Systems," Lecture Notes in Electrical Engineering, Vol. 280, 2014.
[16] P. Hintjens, *ZeroMQ: Messaging for Many Applications*, O'Reilly, 2013

저 자 소 개

류 호 동

2007년 경북대학교 전자전기컴퓨터학부 학사.
2009년 경북대학교 컴퓨터학부 석사.
2009년-현재, 경북대학교 컴퓨터학부 박사과정.
관심분야: 모델기반 테스팅, 분산환경기반 테스팅
Email: hodong@knu.ac.kr

정 수 용

2012년 경북대학교 컴퓨터학부 학사.
2014년 경북대학교 컴퓨터학부 석사.
2014년~현재, 경북대학교 컴퓨터학부 박사과정.
관심분야: 임베디드 소프트웨어 테스팅, 가상 프로토타이핑, 시뮬레이션
Email: kyo1363@naver.com

이 성 희

2012년 경북대학교 컴퓨터학부 학사.
2013년-현재, 경북대학교 컴퓨터학부 석사과정.
관심분야: 임베디드 소프트웨어 테스팅, 분산 컴퓨팅 환경 테스팅 등
Email: lee3229910@gmail.com

김 지 훈

2012년 영남대학교 컴퓨터공학과 학사.
2013년-현재, 경북대학교 융합소프트웨어학과 석사과정.
관심분야: 임베디드 소프트웨어 테스팅, 소프트웨어 시뮬레이션
Email: hjkwlgns@gmail.com

박 흥 준

2013년 금오공과대학교 컴퓨터학과 학사.
2014년-현재, 경북대학교 컴퓨터학부 석사과정.
관심분야: 임베디드 소프트웨어 테스팅 등
Email: bbakeung@naver.com

이 승 민

2012년 BNSoft 주임연구원.
2013년 학점은행제 학사.
2014년-현재, 경북대학교 컴퓨터학부 석사과정.
관심분야: 임베디드 소프트웨어 테스팅
Email: seung.min.lee.k@gmail.com

이 우 진

1992년 경북대학교 전자계산학과 학사.
1994년 KAIST 전산학과 석사.
1999년 KAIST 전산학과 박사.
1999년-2002년 ETRI 선임연구원.
2002년-현재, 경북대학교 IT대학 컴퓨터학부 교수.
관심분야: 임베디드 소프트웨어 테스팅, 임베디드 소프트웨어 개발환경, 실시간 시스템 분석 등
Email: woojin@knu.ac.kr