JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 18, No. 3, pp. 607-612, June. 2017

라즈베리파이 시스템을 이용한 회로 에뮬레이터 솔루션 개발

나 방 현[1] · 이 영 운[2] · 김 병 규[3*]
[1]선문대학교 컴퓨터공학부
[2]선문대학교 컴퓨터융합전자공학과
[3]숙명여자대학교 IT공학과

Development of Circuit Emulator Solution using Raspberry Pi System

Bang-hyun Nah[1], Young-woon Lee[1] · Byung-gyu Kim[2*]

[1]Department of Computer Science and Engineering, Sunmoon University, Asan, 31460, Korea

[2*]Department of Information Technology Engineering, Sookmyung Women's University, Seoul, 04310, Korea

[요 약]

최근 많이 활용되고 있는 라즈베리파이 기반 임베디드 시스템 구축에 있어 사용자는 회로에 대한 이해, 하드웨어 비용 측면에서 어려움을 갖는다. 본 논문은 이러한 시스템을 가상으로 테스트하는 솔루션을 제안한다. 솔루션은 사용자가 실제 회로를 구성하듯이 가상의 공간에 모듈을 배치하고 선을 연결하는 등, 회로를 구성하고 동작을 테스트할 수 있으며 회로편집기, 인터프리터, 시뮬레이터의 세 가지 요소로 구성되어 있고 전체 9개의 모듈을 제공한다. 각 모듈은 제조사에서 제공하는 데이터시트와 제원을 바탕으로 실제 회로 테스트를 거쳐 추상화하였다. 솔루션은 프로토 타입이지만 품질수준을 높인다면 비용절감과 학습, 교육 측면에서 유용할 것이며 이를 위해서, 전기 물리엔진의 구현, 실제 보드로 포팅이 가능한 수준의 인터프리터, 시뮬레이션 로직의 일반화가 필요하다.

[Abstract]

The use of RaspberryPi in building an embedded system may be difficult for users in understanding the circuit and the hardware cost. This paper proposes a solution that can test the systems virtually. The solution consists of three elements; (i) editor, (ii) interpreter and (iii) simulator and provides nine full modules and also allows the users to configure/run/test their own circuits like real environment. The task of abstraction for modules through the actual circuit test was carried out on the basis of the data sheet and the specification provided by the manufacturer. If we can improve the level of quality of our solution, it can be useful in terms of cost reduction and easy learning. To achieve this end, the electrical physics engine, the level of interpreter that can be ported to the actual board, and a generalization of the simulation logic are required.

색인어 : 가상 환경, 라즈베리파이 시스템, 소프트웨어 솔루션, 임베디드 시스템, 회로 에뮬레이터
Key word : Circuit Emulator, Embedded System, Raspberrypi System, Software Solution, Virtual Environment

http://dx.doi.org/10.9728/dcs.2017.18.3.607

Received 08 June 2017; **Revised** 21 June 2017
Accepted 20 June 2017

***Corresponding Author; Byung-Gyu Kim**

Tel: +82-02-2077-7293
E-mail: bg.kim@sookmyung.ac.kr

Ⅰ. 서 론

최근에 라즈베리파이, 아두이노 등의 저가의 임베디드 보드들이 임베디드 시스템에 대한 관심을 상승시키는 견인차 역할을 하고 있다[1]. 이들은 저렴한 가격과 다양한 오픈 소스를 통해 모든 사람들에게 손쉬운 접근성을 제공한다. 그러나 전기회로에 대한 충분한 이해가 갖추어져야 하고 회로를 구성하기 위한 별도의 비용이 요구된다.

에뮬레이션은 특정 하드웨어와 프로그램을 컴퓨터 위에서 동작하는 소프트웨어로 모델링 해주는 기술을 말한다. 에뮬레이터를 이용하면 실제 하드웨어를 가지고 작업하는 것과 같이 빠른 프로토타입 개발 및 실험을 위한 최적의 환경 설정 등을 제공해 준다. 또한 하드웨어를 사용할 때 발생할 수 있는 여러 가지 복잡한 문제없이 프로그램 개발에만 집중할 수 있도록 해준다. 하드웨어를 가지고 작업하기 시작한 후에도 에뮬레이터는 실제 동작과 원하는 동작을 비교해 볼 수 있는 유용한 디버깅 도구가 될 수도 있다[2].

또 다른 용어인 시뮬레이션은 실제로 실행하기 어려운 실험을 간단히 행하는 모의실험을 뜻한다. 실제 하드웨어 시스템을 구축하기에 앞서 가상으로 회로를 테스트해 볼 수 있는 환경이 제공된다면 다음과 같은 장점을 얻을 수 있다[3].

첫째 실제 시스템에 대한 실질적인 구축 없이도 시뮬레이션을 이용하여 평가 할 수 있다. 둘째 비용이 많이 들고 실질적인 실험을 하기에 위험한 운영시스템인 경우 시뮬레이션을 이용해 쉽게 해결 할 수 있다. 셋째 실험의 목적이 강도의 한계를 결정해야 하는 경우, 시뮬레이션을 이용해 시스템 파괴의 위험을 배제할 수 있다.

이와 같이 라즈베리파이를 기반으로 하는 임베디드 시스템을 가상으로 테스트 할 수 있는 솔루션을 개발하게 된다면 시스템 개발과 교육, 학습 분양에서 널리 활용될 수 있을 것으로 기대된다. 라즈베리파이 시스템은 현재 온도제어, 병렬화 프로그래밍 지원, 통신 기술 시뮬레이션 등에서 다양하게 활용되고 있다[4],[5],[6].

이러한 기술 개발의 추세를 반영하여 본 논문에서는 라즈베리파이 기반의 임베디드 시스템 에뮬레이터를 제안하고, 이를 .NetFramework 4.0 기반의 소프트웨어 솔루션으로 개발한 사례를 소개하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 서비스 흐름도를 소개하고 3장에서는 제안된 시스템 구조에 대해서 자세히 설명한다. 실험결과를 4장에서 살펴보고, 마지막으로 5장에서는 본 논문의 결론을 언급하고자 한다.

Ⅱ. 서비스 흐름도

Fig. 1은 개발된 라즈베리파이 기반 에뮬레이터의 서비스 흐름도를 나타낸 것이다. 사용자는실제 전기회로를 구성하는 것과 마찬가지로 시스템에서 제공되는 부품 모듈을 편집공간에 배치하고 모듈과 모듈 사이에 선을 연결함으로써 회로를 구성하게 된다. 회로가 구성되면 사용자는 작동 테스트를 할 수 있으며 테스트는 두 가지 부분으로 구분 할 수 있다.

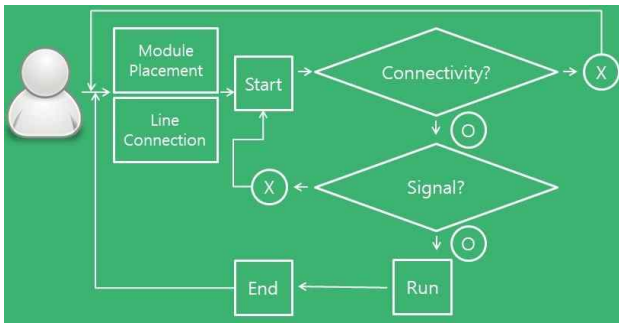


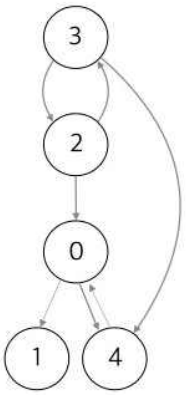
그림 1. 서비스 흐름도
Fig. 1. Service Flow

먼저, 회로연결성 검사이다. 설정된 전원 지점으로부터 접지 지점까지 연결이 되어 있는지 여부와 모듈과 모듈 사이에 연결 여부를 판단하는 것이다. 회로의 연결성 검사는 식 (1)과 같이 그래프 알고리즘의 구현으로 이루어진다.

$$G=(V,E) \tag{1}$$

즉, Fig. 2와 같이 그래프를 인접행렬로 표현하여 회로의 연결유무를 표현하는 것이다. 상호연결이 가능한 모든 정점들을 인접행렬에 저장해두고 선이 연결되면 값을 True, 선이 연결되어 있지 않으면 값을 False로 설정한다. 검사 시점에서는 그래프 순회 알고리즘을 통해 값이 True인 지점들을 탐색하여 간단히 연결여부를 판단 할 수 있게 된다.

회로연결성 검사를 마치고 나면 테스트하고자 하는 부품모듈에 가해진 신호를 분석한다. 이 신호에 따라 모듈이 적절하게 동작하게 된다.



	0	1	2	3	4
0	0	1	0	0	1
1	0	0	0	0	0
2	1	0	0	1	0
3	0	0	1	0	1
4	1	0	0	0	0

그림 2. 인접행렬을 이용한 그래프의 표현
Fig. 2. Represented graph by adjacency matrix

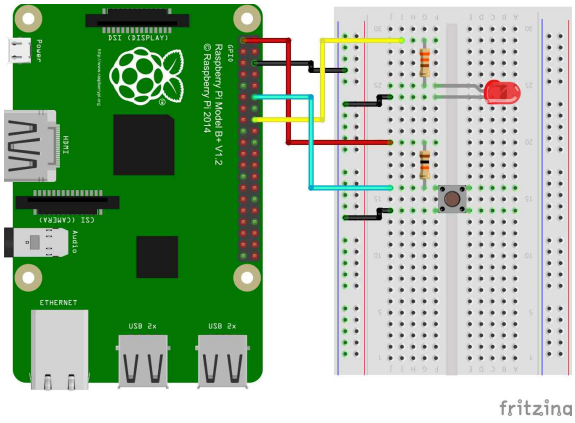


그림 3. LED 회로의 구성
Fig. 3. Configuration of the LED Circuit

예를 들어, Fig. 3와 같이 간단한 LED 회로를 구성했다고 가정한다면 회로연결성 검사 단계에서 RaspberryPi의 어떤 GPIO가 전원과 접지로 선택되었는지를 판단하고 전원과 접지 사이에 LED가 적절하게 배치되었는지 여부와 전원과 LED 사이에 저항이 존재하는지 등을 판단한다.

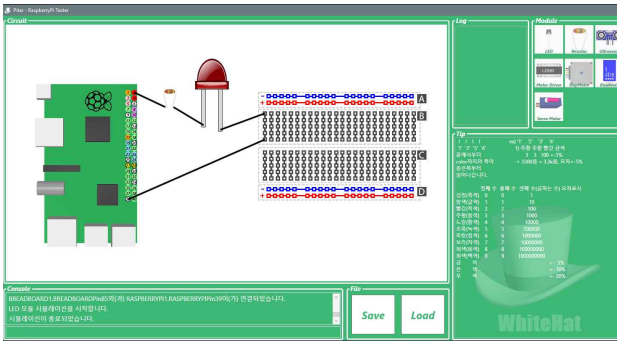


그림 4. LED가 꺼진 상태
Fig. 4. LED Off

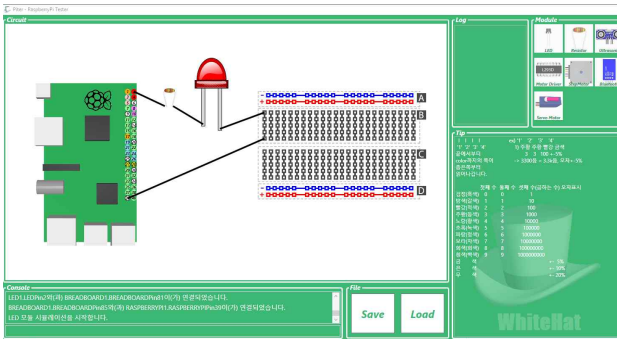


그림 5. LED가 켜진 상태
Fig. 5. LED On

설정을 마치고 시뮬레이션을 시작하게 되면 시작점 GPIO로부터 전달된 신호가 HIGH인지 LOW인지를 판단하여 HIGH 신호에서 LED가 점등되고 LOW 신호에서 LED가 소등되는 동작이 수행되는 것이다.

Ⅲ. 시스템 아키텍처

Fig. 6은 개발된 라즈베리파이 기반 에뮬레이터의 시스템 아키텍처를 표현한 것으로 회로편집기(Editor), 인터프리터(Interpreter), 시뮬레이터(Simulator)의 3가지 부분으로 구성되어 있다.

회로편집기는 라즈베리파이, 브레드보드, LED 등의 모듈을 배치하고 회로를 구성할 수 있는 부분으로서 이는 다시 시각적인 표현을 담당하는 Figure 모듈과 논리적인 표현을 담당하는 Circuit 모듈의 조합으로 구성된다.

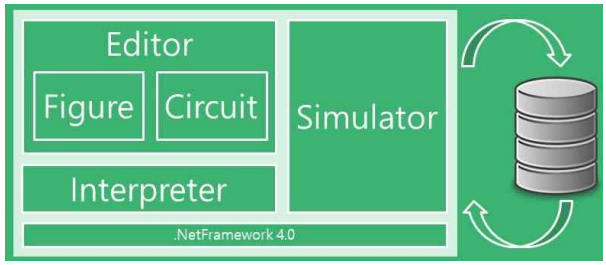


그림 6. 시스템 구조
Fig. 6. System Architecture

즉, WPF상에서 2차원 그래픽 요소를 표현하기 위해 제공되는 Canvas 클래스, 움직이는 모듈 객체를 표현하기 위하여 사용된 Thumb 클래스, 모듈과 모듈 사이에 선 연결을 표현하기 위한 Path 클래스 등의 모든 그래픽적인 표현은 Figure 모듈에서 처리하도록 구현되었다.

표 1. 명령어 종류와 의미
Table. 1. Commands and Meaning

Command	Meaning
create(modul_name)	Module Creation
delete(module_name)	Module Deletion
connect(pin1, pin2)	Making Connection
disconnect(pin1, pin2)	Removing Connection
work(module_name)	Running Simulation
help	Displaying Help

또한, 라즈베리파이, 브레드보드, LED등 프로그램에서 제공하는 부품모듈을 추상화한 Mudule 클래스, 라즈베리파이의 GPIO나 LED의 두 단자와 같이 선으로 서로 연결할 수 있는 정점을 표현한 Pin 모듈, 이러한 정점과 정점 사이에 논리적인 연결과 해제를 표현하기 위한 그래프와 같이 모든 논리적인 부분은 Circuit 모듈에서 처리하도록 구현되어 있다.

좀 더 구체적으로 설명하자면, 사용자가 편집기 내에 LED 모듈을 생성했을 경우, Figure 모듈에 의해서 LED 모양을 나타내는 객체를 생성하여 편집 공간에 배치를 하게 되고 이와 동시에 Circuit 모듈에서 논리적인 LED를 생성한 후에 차후에 모듈 간의 연결과 해제를 처리할 수 있도록 그래프에 LED의 정점을 추가하고 연결 상태를 False로 초기화하는 작업을 수행하는 것이다. 이와 같이 Figure 모듈과 Circuit 모듈의 작업은 1:1의 대응관계를 갖게 되어 있다.

인터프리터는 GUI 즉, 마우스조작에 의해 수행되는 모든 동작을 CUI 즉, 텍스트 명령어 입력을 통해서도 동일하게 수행되도록 하기 위한 부분이다. 예를 들어, 편집공간에 LED를 배치하기 위하여 사용자는 모듈 선택부분에서 LED 모듈에 해당하는 버튼을 마우스로 클릭한 후 편집공간으로 마우스를 이동하여 원하는 위치에서 클릭하게 된다. 그러나 이와 같은 동작은 프로그램의 콘솔 입력창에 “create(led)”라는 명령어를 입력하는 것으로도 가능하다.

표 2. 모듈 종류 및 명령어
Table. 2. Modules and Command

Module	Command
LED	led
Resistor	res
Ultrasonicwave Sensor	us
Bluetooth Module	ble
Servo Motor	serv
Stepping Motor	step
Motor Driver	md
GPIO	gpio
Breadboard	board

Table 1과 Table 2를 통해 프로그램에서 사용가능한 명령어 목록이 정리되어 있다.

시뮬레이터는 모듈에 대한 구체적인 동작을 테스트하는 부분에 해당한다. 개발된 프로그램에서는 모듈마다 별도의 시뮬레이션 로직을 갖도록 구현되어 있다.

Fig. 7은 LED 모듈의 테스트를 위한 테스트 윈도우의 모습이다. 10개의 시퀀스로 구성되어 있으며 각 시퀀스의 체크박스가 HIGH와 LOW 신호를 나타낸다. Start 버튼을 누르게 되면 회로연결성 검사가 우선적으로 수행되고 회로가 올바르게 구성되지 않았을 경우 에러 메시지를 출력하게 된다. 회로연결성 검사를 통과하면 연결된 모듈과 핀 정보가 표시되고 체크박스의 상태를 순차적으로 읽어 들인다. 체크박스가 체크되어 있을 경우 HIGH 신호가 되어 LED는 점등되고 체크박스가 해제되어 있을 경우 LOW 신호가 되어 LED는 소등된다. Stop 버튼을 누를 때까지 이 동작은 반복된다.

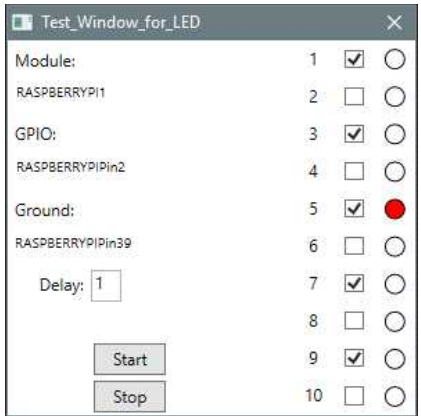


그림 7. LED 테스트 시퀀스를 입력하는 윈도우
Fig. 7. LED Test Window

프로그램에서 제공하는 9개의 모듈 중에서 실질적으로 테스트 윈도우를 가지고 시뮬레이션을 수행할 수 있는 모듈은 5가지이며 저마다 별도로 디자인된 테스트 윈도우를 갖는다. 즉, 앞서 설명한 바와 같이 LED 회로를 구성하게 되면 단순히 신호에 따라 점등과 점멸의 동작을 테스트하게 되지만 블루투스 회로를 구성하게 되면 별도의 컴퓨터에서 Fig. 8과 같은 ‘가상의 집’이라는 프로그램을 실행시킨 후 시리얼통신에 의해 집안의 TV, 에어컨, 조명 등을 조작하는 식으로 시뮬레이션이 가능하다.

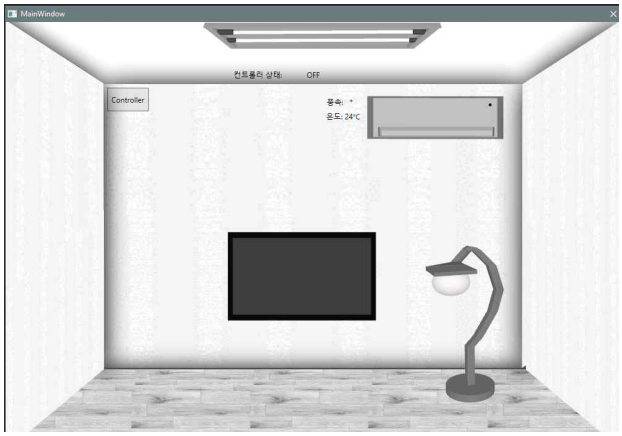


그림 8. 통신 테스트용 가상의 집 프로그램
Fig. 8. Virtual House Program

끝으로 이와 같이 구성된 프로그램 내에서 사용자가 생성한 회로는 사용자가 언제든지 저장매체에 저장해두었다가 필요한 시점에 불러와서 다시 사용할 수 있도록 저장/불러오기 기능이 구현되어 있다.

Ⅳ. 구현 결과

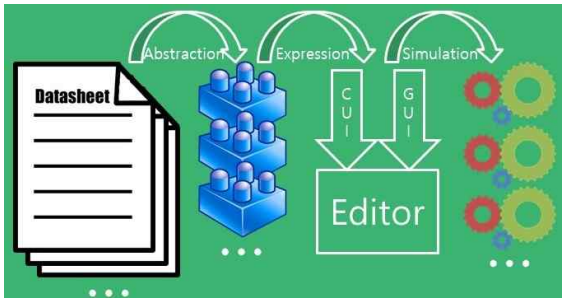


그림 9. 핵심 기술
Fig. 9. Core Technology

Fig. 9은 개발된 프로그램의 주요기술을 표현한 것이다. 도시된 바와 같이 프로그램은 전체 개발공정을 추상화(Abstraction) 단계, 표현(Expression) 단계, 시뮬레이션(Simulation) 단계에 걸쳐 3가지 핵심적인 기술을 가지고 개발되었다.

표 3. 전체 모듈 목록
Table. 3. Module List

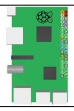
Module	Model	Shape
LED	-	
Resistor	-	
Bluetooth Module	HC-06	
Motor Driver	L293D	
Ultrasonicwave Sensor	HC-SR04	
Stepping Motor	KH42HM2-901	
Servo Motor	SG90	
Raspberry Pi	Raspberry Pi B+	
Breadboard	-	

Table 3에 나타난 바와 같이 프로그램에서는 전체 9가지의 모듈을 제공하고 있다. 이 모듈들을 제공하기 위해서 각각의 모듈 제조사에서 제공하는 데이터시트와 제원표를 참조하였고 이를 토대로 모듈을 추상화하는 작업을 수행하였다.

추상화한 모듈을 회로편집기에 표현하기 위하여 .NetFramework 4.0에서 제공하는 Canvas 클래스, Thumb 클래스, Path 클래스를 활용하여 GUI를 구현하였고 뿐만 아니라 명령어에 의한 회로조작이 가능하도록 인터프리터 구현에 의한 CUI 기능을 제공하였다.

시뮬레이션은 실제로 해당 회로를 구성한 후에 동작 테스트를 수행한 결과를 바탕으로 구현되었다. 또한, 모듈마다 적절한 동작 결과를 보여주기 위하여 각각 별도로 테스트 윈도우를 고안하였다.

Fig. 10은 개발된 프로그램의 전체 모습이다. 왼쪽의 흰색 바탕으로 된 부분이 회로편집기 부분으로 프로그램에서 제공하는 9가지 모듈이 배치되어 있는 모습이다.

회로편집기 하단부는 시스템 메시지가 출력되는 부분, 명령어를 입력할 수 있는 부분, 회로를 저장하거나 불러오기를 할 수 있는 두 개의 버튼으로 구성되어 있다.

회로편집기 우측 상단부는 사용자가 입력한 명령어 로그를 전시하는 부분, 프로그램에서 제공하는 모듈을 선택할 수 있는 부분으로 구성되어 있고 하단부는 도움말 부분이다. 도움말 부분에는 모듈 버튼을 선택할 경우 해당 모듈에 대한 상세 정보가 표시되며 콘솔 창에 "help" 명령어를 입력하여 프로그램에서 제공하는 콘솔 명령어를 확인할 수 있다.

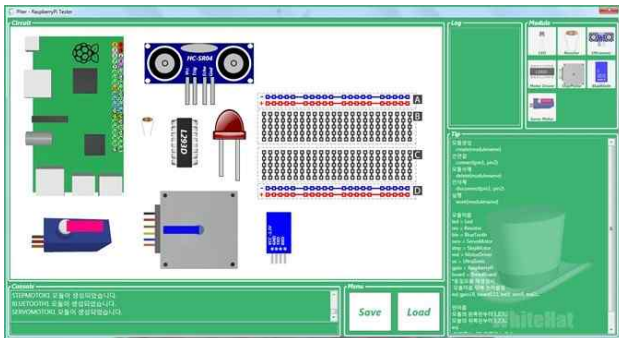


그림 10. 개발된 프로그램
Fig. 10. Developed Program

모듈의 배치는 모듈선택부분에서 원하는 모듈을 클릭한 후에 회로편집기 내의 원하는 위치를 클릭하면 가능하고 배치된 모듈은 마우스로 드래그하여 위치를 이동시킬 수 있다. 모듈의 삭제는 모듈을 오른쪽 마우스로 클릭하여 나타나는 드롭다운 메뉴에서 "Delete"를 클릭한다.

정점의 연결은 시작점을 더블클릭한 후에 연결하려는 정점을 다시 더블클릭하면 두 정점 사이에 연결선이 생성된다. 선을 오른쪽 마우스로 클릭하여 드롭다운 메뉴에서 색상을 변경하거나 선을 삭제할 수 있다.

모듈을 오른쪽 마우스로 클릭하면 드롭다운 메뉴에서 시뮬

레이션을 선택하여 테스트 윈도우를 생성할 수 있다. 테스트 윈도우 내에서 적절한 입력 값을 설정하여 모듈의 동작을 테스트 할 수 있다.

Ⅴ. 결 론

본 논문에서는 .NetFramework 4.0 기반 WPF 응용프로그램으로 라즈베리파이 에뮬레이터를 개발 및 소개하였다. 개발된 프로그램은 품질수준으로 보았을 때 프로토타이핑 수준으로 볼 수 있으나 그 수준을 끌어올린다면 임베디드 시스템을 개발하는 프로젝트에서 비용을 절감시킬 수 있는 수단이 될 수 있으며 교육과 학습 분야에서 널리 활용할 수 있는 가능성이 충분하다.

이와 같은 수준으로 거듭나기 위해서는 다음의 몇 가지 사항이 반드시 충족되어야 할 것이다. 첫째, 전기 물리엔진의 구현이다. 개발된 프로그램에서 단순히 HIGH, LOW 신호에 의한 동작을 표현한 것에서 벗어나 일반적인 전기 물리법칙에 의한 회로의 동작을 표현할 수 있어야 할 것이다. 둘째, 하이레벨의 언어를 해석하여 실제 라즈베리파이 보드에 포팅할 수 있는 수준까지 인터프리터가 향상되어야 할 것이다. 개발된 프로그램을 사용하기 위하여 별도의 매크로나 명령어를 숙지해야 한다는 것은 실용성이 없다고 볼 수 있다. 뿐만 아니라 사용자는 실제 라즈베리파이를 사용하는 것과 동일하게 가상 테스트를 수행하고자 할 것이므로 이와 같은 요구를 만족시키기 위해서는 높은 수준의 인터프리터 기능이 필수적으로 요구된다.

셋째, 시뮬레이션 로직의 일반화가 필요하다. 개발된 프로그램에서는 모듈마다 동작을 테스트하기 위한 별도의 로직이 구성되 있어 번거롭다. 전기 물리엔진이 올바르게 구현된다면 모든 모듈의 동작이 전기 원리에 의해서 수행될 것이므로 좀 더 실제와 같은 시뮬레이션이 가능하게 될 것이다.

이상과 같은 요구사항이 충족된다면 프로그램의 상용화가 가능한 수준까지 품질수준을 높일 수 있을 것이라고 판단된다.

감사의 글

본 연구는 2015학년도 선문대학교 교내학술연구비 지원에 의해 이루어졌습니다.

참고문헌

[1] RaspberryPi official site, https://www.raspberrypi.org/

[2] So-Im Kim, *Prototyping and designing of an embedded emulator creating tool without coding*, Graduate School of the Univ. of Seoul, 2013.

[3] Jong-Sung Park, *A Study on Electric Circuit Education Utilizing Simulation in Technical High School*, Graduate School of Industry, Chonnam National University, 1998.

[4] W.-H. Hur, E.-T. Kim, H.-D. Yang, H.-S. Park, J.-Y. Jung, *Development environment of Deep Learning Training Parallelizing Software,* Korea Institute of Information Science and Engineering Autumn Conference, pp. 37-39, 2015.

[5] Y.-E. Kim, S.-M. Park, K.-M. Oh, D.-K. Park, *Isothermal-Isohumidity Control System Using Raspberry Pi*, Korean Institute Of Information Technology Summer Conference, pp. 406-409, 2015.

[6] Seoung-Kyo Oh, Jae-Deok Choi, *6LoWPAN Based IP-USN System Implementation for Improving Scalability,* THE JOURNAL OF KOREA INFORMATION AND COMMUNICATIONS SOCIETY, Vol. 38B, No. 09, pp. 687-699, 2013.

나방현(Bang-hyun Nah)

1993년 : 고려대학교 토목환경공학과 (석사)
2010년 : 한세대학교 유시티IT산업정책학과 (박사)
2012년 ~ 선문대학교 컴퓨터공학부 부교수

※관심분야 : 영상처리, 지리정보, 컴퓨터교육

이영운(Young-woon Lee)

2016년 : 선문대학교 행정학과 (행정학사)
2016년 : 선문대학교 컴퓨터공학과 (이학사)
2016년 ~ 선문대학교 컴퓨터융합전자공학과 석사과정

※관심분야 : 영상처리

김병규(Byung-kyu Kim)

1995년 : 부산대학교 전기공학과 (공학사)
1998년 : 한국과학기술원 전기 및 전자공학 석사
2004년 : 한국과학기술원 전기 및 전자공학 박사

※관심분야 : 영상/비디오 신호처리, 비디오 부호화, 딥 러닝