가상 프로토타이핑 개발도구 기반의 교차개발환경 설계 및 구현

Design and Implementation for Cross Development Environment based on Virtual Prototyping Development Tools

김대응, 이정배, 임기욱, 황영섭, 안성순
선문대학교 컴퓨터정보학과

Dae-Eung Kim(kde6767@dreamwiz.com), Jeong-Bae Lee(jblee@sunmoon.ac.kr),
Kee-Wook Rim(rim@sunmoon.ac.kr), Young-Sup Hwang(young@sunmoon.ac.kr),
Sung-Soon Ahn(ssAhn83@gmail.com)

요약

최근 임베디드 시스템 개발시간을 단축시키고 고객들의 요구사항을 충족시키기 위하여 많은 방법론과 개발환경 방법들이 제시되고 있다. 특히, 본 논문에서는 임베디드 시스템 개발 시 사용되는 교차개발환경을 개선하여 개발시간을 단축시키는 방법을 제안하였다. 이러한 방법은 가상 프로토타이핑 개발도구와 임베디드 시스템 개발 보드를 이용하여 개발환경을 하는 경우에 제한적으로 적용된다. 구현 및 실험결과 분석을 통하여 한 번 이상의 수정이 요구되는 교차개발환경에서는 기존 개발 방법보다 본 논문에서 제시한 방법이 우수함을 증명하였다.

■ 중심어 : | 실물 프로토타이핑 | 가상 프로토타이핑 | 통합 프로토타이핑 |

Abstract

Recently, many methodologies and development environment to shorten embedded system development time and to satisfy customer requirement were proposed. In this paper, a method to shorten development time by improving cross development environment is proposed for embedded system development, This method was designed and implemented just for the environment using virtual prototyping development tools and embedded development kit. According to implementation and result analysis, the proposed method show the better performance the than existing method in the cross development environment which is required to modify more than once.

■ keyword : | Physical Prototyping | Virtual Prototyping | Integrated Prototyping |

Ⅰ. 서론

최근 디지털 컨버전스의 영향으로 임베디드 시스템의 응용범위가 넓어지고 있다. 이는 기존의 IT기술 간의 융합에서 다른 산업 간의 융합으로 디지털 컨버전스 기술의 확장성이 매우 광범위하기 때문이다.

하지만 현실적으로 임베디드 시스템이 적용되는 각각의 산업마다 설계 기술과 구현방법이 [그림 1]과 같

* 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음. (IITA-2009-C1090-0902-0020)
접수번호 : #090506-007
접수일자 : 2009년 05월 06일
심사완료일 : 2009년 05월 20일
교신저자 : 김대응, e-mail : kde6767@dreamwiz.com

이 상이하기 때문에 해당 산업에 적합한 임베디드 시스템을 설계 및 개발하는데 어려움이 많다. 더욱이 기존과 달리 임베디드 시스템에 요구하는 기능들이 더욱 많아지고 복잡해짐에 따라 적합한 임베디드 시스템을 개발하기 위한 난이도가 높아졌을 뿐만 아니라 제품의 생명주기가 단축되고 있는 실정이다.

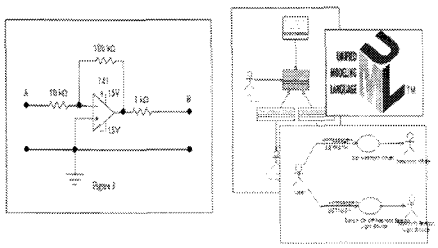


그림 1. 전자회로와 S/W의 설계 방법

임베디드 시스템 개발 작업은 일반적인 소프트웨어나 하드웨어 개발과는 달리 어렵다는 인식이 존재한다. 이런 인식들을 개선하기 위해서 다양한 임베디드 시스템 개발방법론과 개발도구들에 관한 연구가 진행되었다.

효과적으로 임베디드 시스템을 개발하기 위한 다양한 방법들이 연구되고 있지만 현실적인 사용자의 요구와 시스템 개발주기를 따라가기에는 많은 어려움이 존재하기 때문에 고객들의 요구사항에 대한 신속한 분석과 빠른 시스템 개발을 위한 방법으로 가상 프로토타이핑 개발도구를 사용해 교차개발환경(Cross Development Environment) 구성을 제안한다.

교차개발환경이란 실제 소프트웨어가 실행될 타겟 시스템과 개발을 위해 사용하는 호스트 시스템이 다른 환경을 의미한다. 임베디드 시스템의 특수성[1][2]으로 인해서 타겟 시스템에서 직접적인 소프트웨어의 개발이 힘들기 때문에 보통 교차개발환경을 구성하게 된다. 교차개발환경을 구성한 후 시스템을 호스트에서 개발한 후에 크로스 컴파일러(Cross Compiler)를 이용하여 타겟에 의존적인 실행코드를 생성하게 되고 그 실행코드를 타겟에 다운로드 및 이식을 시켜 실행을 하게 된다. 또한 크로스 디버거 (Cross Debugger)를 이용하여 타겟에서 수행 중인 프로그램을 호스트에서 관찰 할 수 있게 함으로써 디버깅이 가능하게 해준다. 하지만 실행 프로그램을 고치기 위해서는 다시 호스트에서 수정을 한 후에 다시 타겟에 이식을 시키는 과정을 반복해야 된다. 본 논문에서는 통합 프로토타이핑[3]에서 이용한 연동 인터페이스를 응용하여 교차개발환경에 적용시키므로써 이러한 반복적인 과정을 없애는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 프로토타이핑 기술과 개발도구들에 대해 기술한다. 3장은 본 논문이 제안하는 교차개발환경의 구성 및 시스템 설계에 대해서 기술한다. 4장에서는 제안한 교차개발환경을 위한 시스템 구현에 대해 기술한다. 5장에서는 기존 개발방법과 제안된 개발방법에 대한 비교 분석하였다. 마지막으로 6장에서 제안시스템에 대한 고찰 및 결론에 대해서 기술하였다.

II. 관련연구

본 장에서는 임베디드 시스템 개발에 사용되는 프로토타이핑 기술과 개발도구들을 소개한다.

1. 프로토타입과 프로토타이핑

실제 상품을 생산하기 전에 외형 및 기능을 시험하기 위하여 만든 실제 크기 혹은 축소형의 실물 모델을 프로토타입이라 한다. 그리고 프로토타입을 컴퓨터를 사용해 구현하는 작업을 프로토타이핑이라고 한다. 프로토타이핑은 크게 [표 1]과 같이 3가지로 구분해 볼 수 있다[3-7].

표 1. 프로토타이핑 방법

구분	설 명
가상 프로토타이핑	임베디드 시스템 개발 초기단계에서 사용자 혹은 의뢰자의 요구사항들을 효과적으로 추출할 수 있도록 고안된 시스템 요구 및 제약 조건 추출 방법론
실물 프로토타이핑	사용자 요구에 부합하는 실제크기, 실제성능을 가진 제품을 개발하는 방법론으로 실질적인 시장성 판단에 도움이 되나 개발비용 및 시간에 많은 노력이 투자되어야 함
통합 프로토타이핑	가상 프로토타이핑과 실물 프로토타이핑의 장점들만을 접목하고자 하는 개발방법론, 하지만 소형 임베디드 시스템이나 네트워크 연결이 이루어지지 않은 환경에서는 적용불가능

다양한 프로토타이핑을 통해서 타 분야의 전문가들과 긴밀한 협업과 시장진입시점이 단축되는 효과를 얻을 수 있기 때문에 여러 방면에서 적용하고자 하는 노력하고 있다.

2. 개발도구

본 절에서는 임베디드 시스템의 개발을 위해서 사용되고 있는 다양한 개발도구들 중에서 대표적인 3가지 개발도구에 대해서 검토한다. 이를 [표 2]에 나타내고 있다[9][11][14].

표 2. 임베디드시스템 개발도구

구분	설 명
Esto	Eclispe/CDT 기반의 통합 그래픽 개발환경 Qplus 플랫폼 기반의 타겟 시스템에서 펌웨어, 응용 소프트웨어 등을 쉽고 빠르게 개발 가능
Virtio	플랫폼 기반의 가상 프로토타이핑 개발도구 마이크로소프트사의 Visual C++과 유사한 인터페이스 제공해 개발자들에게 친숙함
RapidPLUS	가상 프로토타이핑 기반의 임베디드 시스템 HMI 디자인 및 개발도구 컴포넌트 기반 개발방식으로 직관적이고 쉬운 개발이 가능하여 개발자들에게 인기가 높음

살펴본바와 같이 기존의 임베디드 시스템 개발을 위해서 프로토타이핑 방법이 효과적이고 이를 개발하기 위한 도구로 가상 프로토타이핑 개발도구가 자주 사용됨을 알 수 있다. 하지만 프로토타이핑은 실제 제품을 개발하는 데에는 각각의 방법마다 약간의 부정적인 요소들을 지니고 있다. 또한 개발도구들도 플랫폼 혹은 인터페이스에 종속적이라는 특징 때문에 하나만을 사용하는데 어려움이 따른다. 이에 본 논문에서는 가상 프로토타이핑 개발도구를 사용해 호스트시스템에서 시스템을 개발하고 이를 크로스 컴파일러로 실행코드를 생성한 후 임베디드 시스템 개발 보드[15]에 이식하는 교차개발환경을 제시함으로써 임베디드 시스템에 보다 효과적인 개발환경을 제공할 수 있음을 보이고자 한다.

III. 교차개발환경 구성 및 시스템 설계

본 논문에서 사용된 가상 프로토타이핑 개발도구(RapidPLUS)는[14] UML의 State-Machine Diagram과 GUI기반으로 개발 경험이 많은 S/W 개발자들에게 매우 친숙한 구조를 가지고 있으며, 임베디드 시스템에 사용될 수 있는 각종 컴포넌트를 지원을 하여 드래그&드롭을 통하여 임베디드 시스템을 손쉽게 가상 프로토타이핑 할 수 있다. 본 장에서는 실험에서 사용될 임베디드 시스템 개발 보드에 대한 교차개발환경을 구성을 한다. 다음에는 임베디드 시스템 개발 보드에서 사용될 디바이스와 API에 대한 제작을 한다. 그런 후에 임베디드 시스템 개발 보드의 디바이스를 호출하여 가상 프로토타이핑 개발도구에서 이용 가능한 컴포넌트에 대한 설계를 한다.

1. 교차개발환경 구성

본 절에서는 본 논문에서 제안한 방법에 포함되는 엔진을 구현하기 위한 교차개발환경에 대해 설명하고자 한다.

임베디드 시스템의 특수성 때문에 임베디드 시스템 내에서 동작하는 프로그램은 보통 일반 컴퓨터에서 제작한 후에 크로스 컴파일러(Cross Compiler)를 통해서 임베디드 시스템에 맞는 의존적인 코드를 생성 후에 적재를 시켜 실행을 하게 된다. 이러한 개발환경을 교차개발환경이라고 한다.

교차개발환경을 구성하기 위해서는 실제 소프트웨어가 실행되는 시스템(타겟머신)과 개발하는 시스템(호스트머신)이 필요로 하고, 그 호스트머신과 타겟 머신을 연결할 수 있는 연결방법이 필요하다.

본 논문에서 구현할 교차개발환경에 대한 엔진에 대한 교차개발환경 [표 3]에 따라 구성하였다.

표 3. 교차개발환경 구성에 사용된 프로그램

HOST	Core2Duo PC, Linux 2.6 (우분투배포판)
TARGET	X-Hyper320TKU(PXA320), Linux 2.6
Connect	시리얼포트
Compiler	arm-linux-4.1.1

2. 임베디드 시스템 개발 Device 및 API 제작

디바이스 드라이버는 하드웨어 장치를 제어하는 루틴의 집합체이며 운영체제와 연동이 되어 하드웨어를 제어 할 수 있는 인터페이스를 제공한다. 그러한 디바이스 드라이버를 제작하기 위해서는 모듈 프로그래밍을 통한 디바이스 드라이버를 작성 몇 생성을 한 후 커널에 적재를 한다. 적재를 한 후 응용 프로그램에서 접근 테스트를 하여 동작하는지를 확인을 한다. 본 절에서는 실험에서 사용될 장비에 대한 디바이스를 작성하기 위한 설계를 한다.

본 논문에서 사용한 임베디드 시스템 개발 보드(X-Hyper320TKU)는 CPU보드 및 메인보드, IEB보드로 모듈별로 구성되어있다. 본 논문에서는 IEB 보드에 대한 제어를 실험한다. IEB보드의 사양은 [표 4][8]와 같다.

표 4. 임베디드 시스템 개발 IEB 확장보드

분 류	항 목	내 용
FPGA	EP1C6240PQFP	Cyclone(PQFP-240)
ROM	EPC2	EP2(PLCC/SOCKET)
RF디바이스	CC2420	CC2420(QLP48)
DC모터제어	L298	DIP
Step모터 제어	L297	DIP
ADC	ADC0804	
DAC	DAC0800	
7-Segment	7-Segment * 8	8 ea
DOT Matrix	DOT Matrix	5 * 7
문자LCD	문자 LCD	2 * 16
LED	LED Lamp	

본 절에서는 [표 4]에 있는 7-Segment와 DOT Matrix, 문자LCD, LED에 대한 Device와 API를 설계하도록 한다. [표 5]는 본 절에서 설계할 장치들에 대한 Static 메모리맵이다.

표 5. IEB 확장보드 메모리맵

이름	어드레스	범위	설명
FND1	0x11000000	0x100000	7-Segment 1
FND2	0x11100000	0x100000	7-Segment 2
FND3	0x11200000	0x100000	7-Segment 3
FND4	0x11300000	0x100000	7-Segment 4
FND5	0x11400000	0x100000	7-Segment 5
FND6	0x11500000	0x100000	7-Segment 6
FND7	0x11600000	0x100000	7-Segment 7
FND8	0x11700000	0x100000	7-Segment 8
DOT_1 Col	0x11800000	0x100000	Dot collumn1
DOT_2 Col	0x11900000	0x100000	Dot collumn2
DOT_3 Col	0x11A00000	0x100000	Dot collumn3
DOT_4 Col	0x11B00000	0x100000	Dot collumn4
DOT_5 Col	0x11C00000	0x100000	Dot collumn5
Character LCD	0x12300000	0x100000	LCD Control
LED	0x12400000	0x100000	LED

제작될 디바이스 중에서 Character LCD의 제어는 HD44780LCD 컨트롤러 모듈을 사용하고 나머지 FND, DOT, LED는 FPGA의 어드레스 디코딩에 의해서 제어하게 된다.

API는 운영체제가 제공하는 Device나 자원을 좀 더 쉽게 활용할 수 있도록 구성된 함수들의 모음이다. FND, DOT, LED, LCD등 네 가지 장치들에 대한 호출이 쉽도록 [표 6]과 같은 API를 제작한다.

표 6. 임베디드 시스템 개발 보드의 API

이름	설명
unsigned char asc_to_dot(int)	ASC를 DOT로 변환
unsigned char asc_to_fnd(int)	ASC를 FND로 변환
charLcd (char *)	LCD에 Text출력
lcdClear ()	LCD 초기화
segment (char *)	FND에 Text출력
segmentClear ()	FND 초기화
segmentAt (int, char *)	지정된 FND에 Text출력
dotTo (int, int)	DOT 위치 지정
dot (int)	DOT에 int 출력
dotClear()	DOT 초기화
setLED_on(int)	특정 위치 LED를 킨다.
setLED_off(int)	특정 위치에 LED를 끈다.
allClear ()	모든 장치를 끈다.

3. 가상 프로토타이핑 개발도구 Component 설계

본 실험에서 사용되는 가상 프로토타이핑 개발도구를 통해서 제작되는 프로그램은 [그림 2]와 같은 State Machine Diagram의 형식에 따라서 완성된다.

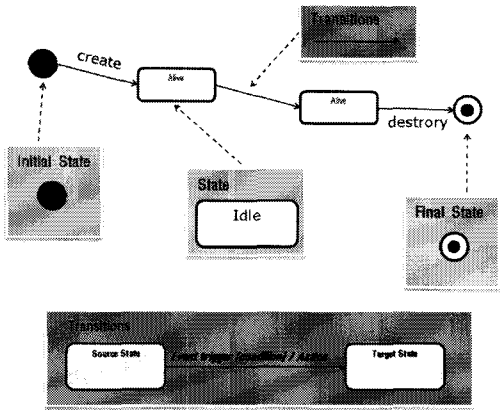


그림 2. State Machine Diagram 동작 흐름

가상 프로토타이핑 개발도구에서는 하나의 State를 Mode라고 이야기하고 Mode안에는 다수의 activity들곧 실행되는 동작들이 존재한다. 하나의 Mode에서 다른 Mode로 이동할 때는 Transition이 일어나는데 그 과정에서 Event와 Action이 동작을 하게 된다. 가상 프로토타이핑 개발도구에서는 그러한 상태변화에 맞춰서 [그림 3]에서 보이는 Logic Editor에서 동작을 정의하게 된다.

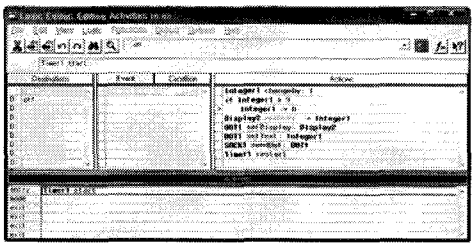


그림 3. 가상 프로토타이핑 개발도구에서의 Logic Editor

이러한 방법으로 3장 2절에서 설계한 임베디드 시스템 개발 보드의 Device에 맞춰 동일하게 가상 프로토타이핑 개발도구의 컴포넌트들도 설계를 한다. [표 7]은 앞으로 가상 프로토타이핑 개발도구에서 제작하게 될 임베디드 시스템 개발 보드의 Device에 대응되는 컴포넌트 목록이다.

표 7. 가상 프로토타이핑 개발도구에 추가될 컴포넌트 이름과 기능

컴포넌트명	기능
xhyper320TKU_LCD	임베디드 시스템 개발 보드의 LCD를 제어한다.
setDisplay(Display)	특정 Display를 설정한다.
setText(String)	Text값을 설정한다.
setText(String,Display)	Text와 Display를 설정한다.
setText(String,int,int,Display)	지정된 위치에 Text와 Display의 값을 설정한다.
getText()	설정된 Text값을 불러온다.
xhyper320TKU_LED	임베디드 시스템 개발 보드의 LED를 제어한다.
setLED(Lamp)	특정 Lamp를 설정한다.
setLocation(int)	Lamp의 위치를 설정한다.
getLocation()	설정된 Lamp의 위치를 가져온다.
xhyper320TKU_Dot	임베디드 시스템 개발 보드의 Dot를 제어한다.
setDisplay(Display)	특정 Display를 설정한다.
setText(String)	Text값을 설정한다.
getText()	설정된 Text값을 불러온다.
xhyper320TKU_FND	임베디드 시스템 개발 보드의 FND를 제어한다.
setDisplay(Display)	특정 Display를 설정한다.
setLocation(int)	Text가 표시될 Display 위치를 설정한다.
setText(String)	Text값을 설정한다.
getLocation()	설정된 Text의 위치를 불러온다.
getText()	설정된 Text값을 불러온다.
xhyper320TKU_Socket	임베디드 시스템 개발 보드와 가상 프로토타이핑을 연동시킨다.
sendDot(xhyper320TKU_Dot)	xhyper320TKU_Dot를 임베디드 시스템 개발 보드에 전송
sendLED(xhyper320TKU_LED)	xhyper320TKU_LED를 임베디드 시스템 개발 보드에 전송
sendLCD(xhyper320TKU_LCD)	xhyper320TKU_LCD를 임베디드 시스템 개발 보드에 전송
sendFND(xhyper320TKU_FND)	xhyper320TKU_FND를 임베디드 시스템 개발 보드에 전송
setServer(IP,PORT)	IP와 Port값을 설정한다.
connect()	임베디드 시스템 개발 보드와 연결한다.
send()	모든 Object를 전송한다.

Ⅳ. 시스템 구현

본 장에서는 3장에서 설계한 내용을 바탕으로 [그림 4]와 같은 구성을 갖는 임베디드 시스템 개발 보드 엔

진과 가상 프로토타이핑 개발도구의 Componet를 구현한다.

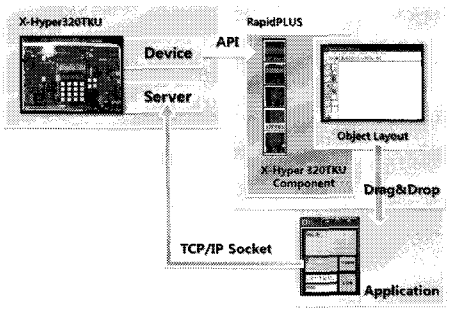


그림 4. 시스템 구성도

1. 임베디드 시스템 개발 보드 엔진 구현

3장 2절에서 제작한 디바이스 드라이브와 API를 이용하여서 [그림 4]에서 보듯이 응용프로그램과 임베디드 시스템 개발 보드가 TCP/IP 소켓으로 연결할 수 있는 엔진을 [그림 5]와 같이 구현하였다.

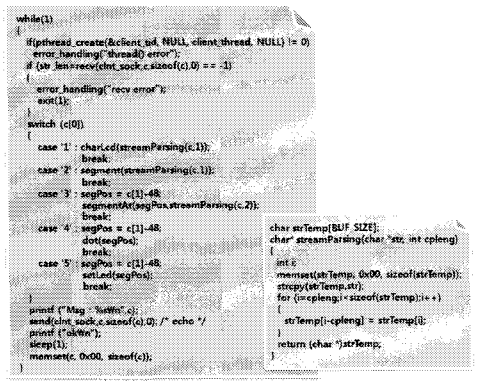


그림 5. 임베디드 시스템 개발 보드 Server

가상 프로토타이핑 개발도구의 컴포넌트에는 임베디드 시스템 개발 보드의 디바이스 드라이버에 맞는 변수값이 할당되어있고, [표 5]에 있는 xhyper320TKU_Socket 컴포넌트에 있는 connect() 메소드를 이용해서 임베디드 시스템 개발 보드에 Server로 접속한 후 send[디바이스]() 메소드를 호출하게 되면 Server에서는 streamParsing() 함수를 통해서 가상 프로토타이핑 개발도구에서 넘어온 디바이스 값을 확인 한 후 스래드를 생성시킨 후 생성된 쓰레드에서 해당 디바이스에 대한 명령을 실행하도록 구현하였다.

2. 가상 프로토타이핑 개발도구 Component 구현

가상 프로토타이핑 개발도구에서의 컴포넌트 구현은 3장 3절에서 소개한 방법을 따라서 [표 7]에 있는 컴포넌트와 메소드를 개발하도록 한다. Logic Editor에서 Edit Function을 눌러 [그림 6]처럼 Edit Function 모드로 들어간다.

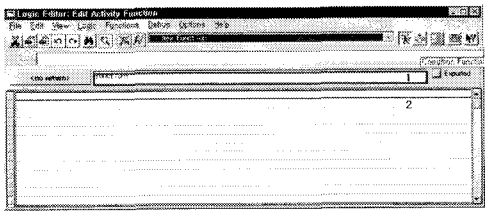


그림 6. Edit Function 창

[그림 6]에서 보이는 1에 메소드를 적고 2에 바디를 작성하면 된다. 작성이 완료된 후에는 Accept를 눌러 저장을 하고 나오면 된다. 이러한 방법으로 해당 메소드를 모두 작성한 후에 Object Layout창 메뉴에서 [EDIT]을 선택 한 수 [User Object Properties]메뉴를 [그림 7]처럼 실행시킨다.

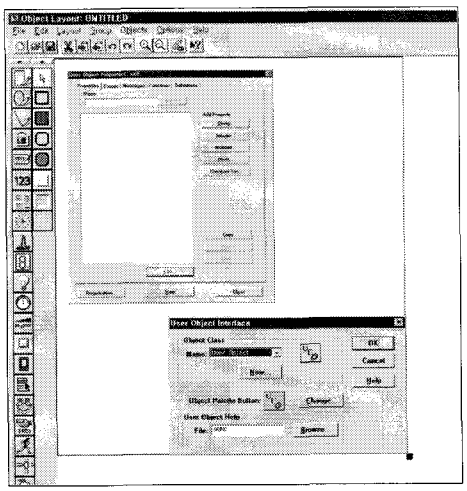


그림 7. Layout창과 그 하위 창들

User Object Properties에서는 컴포넌트에서 사용될 Property와 Function 그리고 Event의 허용여부를 선택할 수 있다. 그리고 하단에 있는 Registration을 선택하게 되면 제작한 컴포넌트를 꾸러미로 묶을 수 있게 해주는 메뉴가 뜨게 된다. 이와 같은 방법으로 컴포넌트를 제작한 후에 가상 프로토타이핑 개발도구에 이식 가능하도록 UDO(User Define Object)라는 파일 포멧을 사용하여 저장한다. 그런 다음 컴포넌트를 가상 프로토타이핑 개발도구에 등록하면 [그림 8]과 같이 추가된 컴포넌트를 이용할 수 있게 된다.

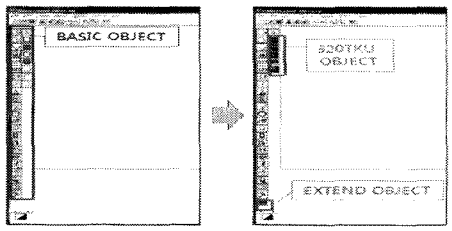


그림 8. 가상 프로토타이핑 개발도구에 추가된 컴포넌트

V. 구현결과 평가 및 분석

본 논문에서 구현한 가상 프로토타이핑 개발도구기반의 교차개발환경을 평가하기 위해서 3장에서 설계하고 4장에서 구현한 환경을 바탕으로 응용 프로그램을 제작을 하고, 응용 프로그램을 제작하는 과정을 분석하여 기존 임베디드 시스템 개발시간에서 단축되었는지를 평가하도록 한다.

1. 평가를 위한 응용프로그램 제작 및 실행

평가를 위해서 4장에서 구현한 임베디드 시스템 개발보드에 엔진을 포트 값을 주고 [그림 9]와 같이 실행시킨다.

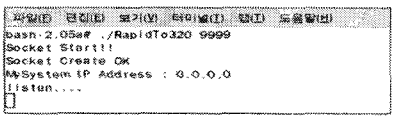


그림 9. 타겟 보드 엔진 실행 모습

가상 프로토타이핑 개발도구에 포함될 컴포넌트를 4장에서 구현하였다. 개발된 컴포넌트는 임베디드 시스템 개발 보드의 디바이스 호출을 포함하였다. 4장에서 구현한 추가된 컴포넌트를 이용하여 평가를 위한 응용프로그램을 작성하도록 하겠다. 작성하는 방법은 3장 3절에서 소개한 방법을 따르도록 한다. 추가된 각각의 컴포넌트에는 setDisplay(), setText()와 같은 함수가 있어서 기본 오브젝트를 배치를 하고 추가 컴포넌트의 함수를 이용하여 기본 오브젝트와 타겟 시스템과의 매치를 해준다.

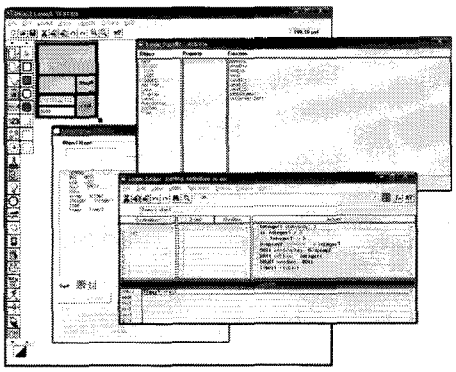


그림 10. 응용프로그램 제작 모습

[그림 11]은 오브젝트와 타겟 시스템과의 매치를 시켜주는 소스를 보여주고 있다.

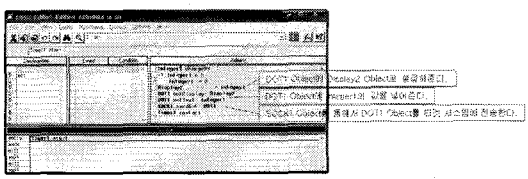


그림 11. 추가된 컴포넌트 사용 소스

프로그램을 제작하면서 프로토타이퍼(prototyper)를 실행시켜 임베디드 시스템 개발 보드와 연동되어 동작하는지 확인해본다.

[그림 12]는 프로그램은 CON버튼을 누르면 임베디드 시스템 개발 보드와 소켓으로 접속이 되고, Start버튼을 누르면 [그림 12]의 1번 디스플레이가 Ready에서

Run으로 바뀌고 2번 디스플레이는 0부터 9까지 카운터를 하는 모습을 보이고 있다.

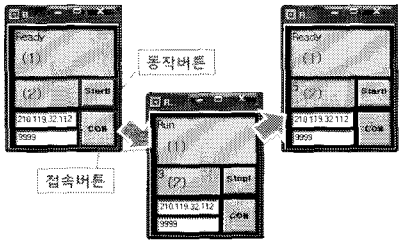


그림 12. 실행모습

가상 프로토타이핑 개발도구에서 [그림 12]와 같이 실행모습이 보일 때 임베디드 시스템 개발 보드에서 [그림 13]과 같이 실행하는 모습을 볼 수 있다.

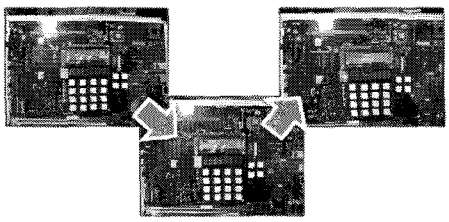


그림 13. 가상 프로토타입과 동기된 실행모습

[그림 12]와 [그림 13]을 통해서 가상 프로토타이핑 개발도구에서 프로그램을 개발할 때 임베디드 시스템 개발 보드에서 따로 크로스컴파일을 할 필요 없이 동작하는 모습을 확인 할 수 있다.

2. 응용프로그램 제작 과정을 통한 평가

본 절에서는 앞 절에서 제작된 응용프로그램의 제작 과정에 따라 기존 임베디드 시스템 제작방법에 비교해서 얼마나 단축할 수 있었는지를 평가하고자 한다.

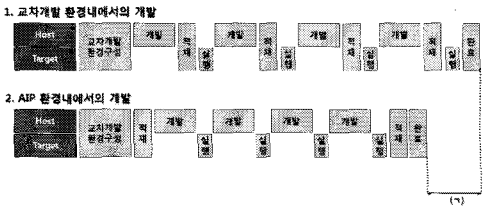


그림 14. 임베디드 시스템 개발 제작 과정

기존 교차개발환경 내에서의 임베디드 시스템 개발은 [그림 14]의 1에서 보듯이 교차개발환경을 구성을 한 후 호스트머신에서의 개발(C), 호스트머신에서 타겟머신으로의 적재(L), 타겟머신에서의 실행(R)의 과정을 거친 후에 프로그램이 완성이 된다. 하지만 실행 후 개발내용을 수정(E)하게 되면 C의 작업부터 다시 해야 된다. W1을 기존 교차개발환경 내에서의 작업횟수라고 정의하고 C와 L과 R의 각각의 과정은 수정 시 동일한 시간이 걸린다고 가정하고 수식으로 나타내면 다음과 같다.

$$W1 = (C + L + R)E \qquad (1)$$

본 논문에서 제안하고 구현한 내용의 제작과정(AIP-Advanced Integration Prototyping)은 [그림 14]의 2에서 보듯이 교차개발환경을 구성한 후 호스트 시스템에서의 개발(C), TCP/IP소켓통신을 이용한 실행(R), 그리고 수정(E)시 C와 R의 작업을 하고 E가 모두 끝났을 때 호스트 시스템에서 타겟 시스템으로의 적재과정(L)을 거치고 완료하게 된다. AIP에 대한 작업횟수를 W2라고 정의하고 C와 L과 R의 각각의 과정은 수정 시 동일한 시간이 걸린다고 가정했을 때 수식으로 표현하면 수식(2)과 같다.

$$W2 = (C + R)E + L \qquad (2)$$

W1과 W2를 비교하였을 때 시스템 개발에 한 번도 수정이 발생하지 않는다면 수식(1)에 따라 W1의 값이 우수하다. 하지만 현실적으로 시스템 개발 작업에는 여러 번의 수정이 발생하게 되기 때문에 수식(2)에서 나타내고 있는 W2를 사용하는 것이 개발시간을 단축할 수 있다. [그림 14]는 임베디드 시스템 개발 시 수정 작업이 3번 발생한다고 가정했을 때 기존 방법과 AIP 방법 간에 ㉠(W1-W2)만큼의 시간이 절약된 모습을 확인할 수 있다.

3. 기존 임베디드 시스템 설계 방법과의 비교 평가

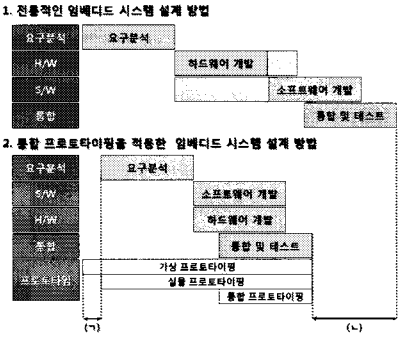


그림 15. 임베디드 시스템 개발방법

[그림 15]에서 1번은 프로토타이핑 방법이 나타나기 전의 전통적인 임베디드 시스템 설계 방법이다. 하지만 2번의 프로토타이핑 방법이 나타나게 됨에 따라서 소프트웨어와 하드웨어의 동시개발이 가능하게 되었다. 초기에 ㉠만큼의 프로토타이핑을 하는데 시간이 더 추가로 들지만 제품의 완성에서는 ㉡만큼의 시간이 절약되어 시장의 진입시점이 더 빨라지게 되었다.

VI. 결론

임베디드 시스템 제품의 시장진입시점을 빠르게 하여 제품에 대한 경쟁력을 높이기 위해서 많은 개발방법론과 개발도구들이 등장하기 시작하였다.

본 논문에서는 이러한 프로토타이핑 방법에 기반을 두고 가상 프로토타이핑 개발도구를 사용하여 교차개발환경 내에서 개발시간을 더 단축시키는 방법을 연구하였고, 연결 인터페이스를 이용한 실시간 테스트라는 방법을 찾아내었다. 그리고 5장에서 나타낸바와 같이 개발시간이 단축될 수 있음을 증명하였다.

향후 과제로는 본 논문에서 소개한 AIP방법에서 타겟머신에도 유저인터페이스를 제공하여 개발 도중에 타겟에서 호스트에 접근할 수도 있는 좀 더 유기적인 개발환경을 구성하고 테스트뿐 아니라 코드의 최적화를 통한 적재가 이루어질 수 있게 하는 것이라 할 수 있다.

참 고 문 헌

[1] 정기훈, 채화영, 김정길, 이재신, 강순주, "실물프로토타이핑 기법을 적용한 임베디드 실시간 시스템 소프트웨어 방법론", 전자공학회지, 제31권, 제11호, pp.30-41, 2004.

[2] 정기훈, 채화영, 김정길, 이재신, 강순주, "임베디드 시스템 프로토타이핑 기술", 정보처리학회지, 제11권, 제6호, pp.86-99, 2004.

[3] 김종일, 이정배, 양재수, 이영란, 정영진, 한강우, 강신관, 김대웅, "객체 기반의 임베디드 실물 및 가상 프로토타입 통합 기법", 정보처리학회논문지, 제14-A권, 제4호, pp227-234, 2007.

[4] 정영진, 이정배, 이영란, 권진백, 임기욱, 조상영, "프로토타이핑 기술을 적용한 임베디드 시스템 가상 시뮬레이션", 정보과학회지, 제24권, 제8호, pp.26-39, 2006.

[5] 이영란, "*임베디드시스템의 가상 및 실물 통합 프로토타이핑 시스템 설계 및 구현에 관한 연구*", 박사학위논문, 2006.

[6] 김종일, "*임베디드 시스템 개발을 위한 객체 기반의 가상 및 실물 프로토타이핑 통합 환경*", 박사학위논문, 2007.

[7] 이정배, 김영진, 안성순, 김대웅, "*자동차 임베디드 프로토타이핑 기술 소개*", 정보처리학회지, 제15권, 제5호, pp.49-58, 2008.

[8] 하이버스(주), *임베디드 리눅스 설계 및 응용 (PXA320 기반, Embedded Linux Kernel 2.6)*, 2008.

[9] (주)파이널이펙트, *RepidPlus 교재*, 2005.

[10] 강순주, 이정배, 박종진, *리눅스 기반 임베디드 시스템*, 대영사, 2004.

[11] 한국전자통신연구원 융합소프트웨어 연구본부, *Esto3.4 사용자 매뉴얼*, 임베디드 SW 기술 연구팀, 2009.

[12] 정영진, 이정배, 이영란, 권진백, 임기욱, 조상영, "프로토타이핑 기술을 적용한 임베디드 시스템 가상 시뮬레이션", 정보과학회지, 제24권, 제8호,

pp.26-39, 2006.
[13] ESPS, http://www.artsystem.co.kr
[14] RapidPLUS, http://www.e-sim.com
[15] X-Hyper320TKU, http://www.hybus.net

저 자 소 개

김 대 응(Dae-Eung Kim) 정회원

- 1992년 2월 : 한밭대학교 전자공학과 전자공학전공(공학학사)
- 1997년 2월 : 한밭대학교 대학원 전자공학과 전자공학전공(공학석사)
- 2007년 2월 : 선문대학교 대학원 컴퓨터정보학과(박사수료)
- 2004년 3월 ~ 현재 : 백석문화대학 컴퓨터정보학부 겸임전임교수

<관심분야> : 임베디드 시스템, 실시간 시스템, 3차원 기반 통합프로토타이핑, 실시간 시스템

이 정 배(Jeong-Bae Lee) 종신회원

- 1981년 2월 : 경북대학교 전자공학과 전산전공(공학학사)
- 1983년 2월 : 경북대학교 대학원 전산전공(공학석사)
- 1995년 2월 : 한양대학교 대학원 전자공학과(공학박사)
- 1982년 ~ 1991년 : 한국전자통신연구원 선임연구원
- 1991년 ~ 2002년 : 부산외국어대학교 컴퓨터공학과 교수
- 1996년 ~ 1997년 : U.C.Irvine 객원교수
- 2002년 ~ 현재 : 선문대학교 컴퓨터공학부 교수

<관심분야> : 임베디드 시스템, 실시간 시스템, 실시간통신 프로토콜

임 기 욱(Kee-Wook Rim) 정회원

- 1977년 : 인하대학교 전자공학과(공학학사)
- 1987년 : 한양대학교 컴퓨터공학과(공학석사)
- 1994년 : 인하대학교 컴퓨터공학부(공학박사)
- 2000년 ~ 현재 : 선문대학교 컴퓨터공학부 교수

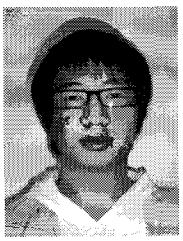
<관심분야> : 데이타베이스, 임베디드 시스템

황 영 섭(Young-Sup Hwang) 정회원

- 1989년 : 서울대학교 컴퓨터공학과(공학학사)
- 1991년 : 포항공과대학교 컴퓨터공학과(공학석사)
- 1997년 : 포항공대대학교 컴퓨터공학부(공학박사)
- 1997년 ~ 2002년 : 한국전자통신연구원 선임연구원
- 2002년 ~ 현재 : 선문대학교 컴퓨터공학부 교수

<관심분야> : 컴퓨터공학, 바이오공학

안 성 순(SungSoon Ahn) 준회원



- 2008년 9월 ~ 현재 : 선문대학교 전자계산학과 석사과정

<관심분야> : 임베디드 시스템, 3차원기반 프로토타이핑