특집

실물 프로토타이핑 기법을 적용한 임베디드 실시간 시스템 소프트웨어 개발 방법론

정기훈, 채화영, 김정길, 이재신, 강순주 (경북대학교 전자공학과)

I. 서 론

임베디드 시스템 소프트웨어 개발은 작은 컴퓨터에 탑재되는 소규모 소프트웨어를 작성하는 단순한 작업으로 여겨질 수 있으나 이는 잘못된 생각이다. 임베디드 소프트웨어 개발은 범용 소프트웨어 개발과 달리 저 전력, 저 메모리를 사용하여 제품 단가, 실시간성 등의 요구사항을 맞춰야 하는 매우 엄격한 제약 조건을 가진다. 뿐만 아니라 필수적으로 하드웨어와의 동시 설계 및 역할 분담 과정을 거쳐야 하며, 이로 인해 일반 소프트웨어 개발 방법론과는 차별화된 개발 절차를 요구한다. 크게는 요구 분석, 하드웨어/소프트웨어 역할 분담 및 동시 설계, 그리고 재 통합의 과정을 필수적으로 포함해야 하며, 이 과정 속에서 하드웨어/소프트웨어에 대한 설계 및 구현, 그리고 그에 대한 검증이나 디버깅 작업 또한 거쳐야 한다. 이러한 개발과정은 개발 프로젝트에 대한 생명 주기 모델 선택, 응용 분야, 혹은 개발 환경에 따라 다양하게 달라질 수 있으며, 개발자들에게는 문제에 대한 정확한 이해와 기술 변화에 대한 직시, 그리고 엄정한 기술 분석에 의한 도구 및 개발 환경 선정 등 현명한 전문적 판단이 요구된다.

따라서 본 논문에서는 임베디드 실시간 시스템 소프트웨어의 차별성과 특수성을 반영한 효과적인 개발 방법론을 제안하고, 이를 기반으로 한 임베디드 시스템 개발과정 전체를 체험할 수 있는 실험 실습 교육 과정에 대해서도 논하고자 한다.

II. 임베디드 실시간 시스템의 차별성

임베디드 시스템은 일반 데스크톱 컴퓨터 시스템과 달리 응용분야에 따라서 매우 다양한 설계 및 구현 상의 제약 조건 들을 내포하고 있다. 따라서 임베디드 소프트웨어 개발 과정은 이러한 제약 조건을 충실히 반영하여야 성공적인 개발을 보장할 수 있다. 우선 임베디드 시스템 소프트웨어가 응용분야에 따라 얼마나 다양한 제약 조건들을 내포하고 있는 가를 서술한 후에 적절한 개발 방법론을 제시하겠다.

임베디드 소프트웨어는 스프레드시트, 워드프로세서, 통계 프로그램 등의 일반 데스크톱 컴퓨터에서 운용되는 소프트웨어들과는 달라서, 센서나 엑츄에이터를 이용해 외부 환경과 입출력

〈표 1〉 임베디드 시스템 개발 요건에 대한 응용 분야별 분류

An example of	Signal Processing	Mission Critical	Distributed	Small
Computing speed	1 GFLOPS	10-100 MIPS	1-10 MIPS	100,000 IPS
I/O Transfer Rates	1 Gb/sec	10 Mb/sec	100Kb/sec	1 Kb/sec
Memory Size	32-128 MB	16-32 MB	1-16 MB	1 KB
Units Sold	10-500	100-1000	100-10,000	1,000,000+
Development Cost	$20M-$100M	$10M-$50M	$1M-$10M	$100K-$1M
Lifetime	15-30 years	20-30 years	25-50 years	10-15 years
Environment	Vibration, Heat	Heat, Vibration Lightning	Dirt, Fire	Over-voltage, Heat, Vibration
Cost Sensitivity	$1000	$100	$10	$0.05
Other Constraints	Size, weight, power	Size, weight	Size	Size, weight, power
Safety	-	Redundancy	Mechanical Safety	-
Maintenance	Frequent repairs	Aggressive fault detection/maintenance	Scheduled maintenance	"Never" breaks
Digital content	Digital except for signal I/O	~1/2 Digital	~1/2 Digital	Single digital chip, rest is analog/power
Certification authorities	Customer	Federal Government	Development team	Customer; Federal Government
Repair time goal	1-12 hours	30 minutes	4 min-12 hours	1-4 hours
Initial cycle time	3-5 years	4-10 years	2-4 years	0.1-4 years
Product variants	1-5	5-20	10-10,000	3-10
Engineering allocation method	Per-product budget	Per-product budget	Allocation from large pool	Demand-driven daily from small pool
Other possible examples in this category :	Radar/Sonar Video Medical imaging	Jet engines Manned spacecraft Nuclear power	High-rise elevators Trains/trams/subways Air conditioning	Automotive auxilliaries Consumer electronics "Smart" I/O

을 해야 한다. 이로 인해 센서 신호의 입력이나 엑츄에이터 제어를 위한 신호들은 당연히 유효한 허용 시간을 가지게 된다. 또한 임베디드 소프트웨어들은 제어 이론 혹은 신호처리 알고리즘, 유한 상태 변화 개념 등을 이용하여, 외부환경의 상태 변화를 정확히 인식해야 하며, 상태 변화에 따른 적절한 조치를 자율적으로 수행할 수 있어야 한다.

다음 <표 1>은 임베디드 시스템의 대표적인 응용분야 4가지를 선정하고 각각 응용분야에서 요구하는 제약 조건의 차별성을 분류한 것이다[1]. 응용분야는 크게 레이다/소나 또는 비디오/오디오 신호 등을 처리하는 신호처리 분야, 유인/무인 항공기, 수화력 및 원자력 발전소 또는 인공위성 등 안전성(Safety)이 매우 중시되는 특수 목적용(Mission) 제어 응용분야, 엘리베이터, 지하철 제어, 공장 제어 혹은 백색 가전 등에 적용되는 분산 제어 응용분야, 그리고 최근에 활발한 개발 경쟁이 이뤄지고 있는 PDA, 셋탑박스, 게임기 등 소규모 정보가전 제품 응용분야로 나눌 수 있다.

<표 1>에서 보듯이 응용분야별로 매우 다양한 제약조건들을 가지고 있음을 알 수 있다. 사례별로 살펴보면, 신호처리 응용분야는 매우 빠른 컴퓨팅 스피드를 중요시 하는 반면, 특수 목적용 제어 시스템은 안정성을 확보하기 위한 중복 제어 구조와 이상신호에 대한 매우 예민한 반응을 중요시 한다. 또한 분산 제어 시스템에서도 안전성 확보가 중요한 요건이면서 분산 네트워크의 통신 품질(QoS) 확보를 중요한 요건으로 하며,

소규모 정보가전 응용 분야의 경우 데스크톱 컴퓨터 소프트웨어와 유사한 구조를 가지고 있으나, 제품 단가에 매우 민감하고 저전력 능력 등이 중요한 요건으로 작용한다.

이렇게 개발을 위한 제약 조건의 다양성 때문에, 임베디드 시스템을 위한 소프트웨어를 개발하면서 일반화된 소프트웨어 공학 이론에 근거한 개발 방법론을 적용하는 것은 의미가 없으며 응용분야에 최적화된 개발 방법론을 제시하는 것이 보다 실용적이고 현실적인 방안이라고 판단된다.

III. 임베디드 실시간 시스템 개발 방법론의 요건 및 관련 연구

위에서 언급한 바와 같이 임베디드 시스템 개발에서는 이 분야의 차별성과 다양성으로 인해 일반론적인 소프트웨어 공학을 기반으로 한 개발 방법론은 실용적이지 못하다. 따라서 보다 실용적이면서 생산적인 개발 방법론이 필요하며, 이를 위해 임베디드 실시간 시스템 개발 방법론은 다음과 같은 요건들을 필수적으로 만족시켜야 한다.

가) 사용자 기호 및 개발 내용에 대한 조기 성능 예측 및 이해 과정 요구

모든 소프트웨어 개발은 사용자의 요구에서 시작되며, 사용자의 요구를 면밀히 분석하여 개발 제품의 외형, 주요 기능 및 성능을 결정하게 된다. 특히, 임베디드 시스템은 개발의 성패가 전적으로 이 단계에 의존한다고 해도 과언이 아니다. 이 단계에서 가장 큰 문제는 사용자들이 원하는 내용을 어떻게 효과적으로 도출하고, 그 내용을 구체화 하느냐 이다. 하지만 사용자들 입장에서는 개발자들이 사용하는 개발 문서는 거부감이 들뿐만 아니라, 개발자 기준의 문서 혹은 요구 수집 과정은 너무 경직될 수가 있어서, 일반 사용자들의 자유로운 요구 표현을 받아들이기 어려운 문제가 상존한다. 게다가 일반적으로 사용자들은 자신이 원하는 내용이 기술적으로 어떤 것인지 확실하게 알고 있지 못하며, 구체적으로 표현하지도 못하는 경우가 많다. 이러한 환경임에도 불구하고 사용자의 요구를 정확히 추출하기 위한 많은 연구들이 수행되고 있으며 대표적인 사례 중의 하나가 건축 등에서 일반화되어 활용되고 있는 개념인 가상 프로토타이핑 기법이다.

나) 철저한 응용분야 적응형 개발 방법론 요구

앞에서 언급한 바와 같이 임베디드 시스템은 그 응용분야에 따라 매우 다양한 제약조건을 가지고 있어서, 일반적인 소프트웨어 공학적 접근법으로는 제품 개발의 성공을 결코 보장할 수 없다. 따라서 응용분야의 특수성을 잘 반영할 수 있고, 응용분야의 소프트웨어 특성에 맞게 최적화가 가능한 개발 방법을 적용해야 한다. 이러한 접근 방법을 응용분야 적응형(Domain-Specific) 개발 방법론이라고 하며, 임베디드 시스템 소프트웨어 개발 분야에서 최근에 활발히 도입되고 있다.

다) 재사용 가능하고 Time-to-Market을 최소화 할 수 있는 환경 요구

임베디드 시스템은 주로 상품화에 시급을 요하는 첨단 신제품들이다. 따라서 안정성이 매우 중요하고 개발 난도가 높음에도 불구하고, 상품 개발 기간을 최소화 해야만 시장을 선도하는 첨단 신제품의 자격을 얻을 수 있게 된다. 따라서

임베디드 시스템을 위한 소프트웨어는 가능한 한 공통 소프트웨어 구조를 이용한 재사용 가능한 컴포넌트 형태로 개발해야 하며, 이들 컴포넌트 또한 응용분야에 최적화할 수 있도록 재구성 기능을 지원해야 한다. 최근 연구 결과에 의하면 많은 임베디드 소프트웨어가 적어도 60% 이상의 공통적인 소프트웨어 플랫폼을 가지고 있으며, 응용분야 최적화를 적용할 경우 신제품을 위한 소프트웨어 개발에서도 최대 90% 정도는 기존 개발 소프트웨어를 재사용 할 수 있다고 한다.

라) 개발 초기 혹은 개발 중간에서 개발 결과의 성능에 대한 예측 가능 모델 요구

초기 명세 단계에서의 개발 제품에 대한 성능 예측뿐만 아니라 개발 과정 중의 수시적 성능 예측 및 개선 작업도 임베디드 실시간 시스템 개발에는 매우 중요한 요소이다.

마) 전공이 서로 다른 개발자들 간에 하드웨어/소프트웨어 동시 설계를 위한 이해 과정 요구

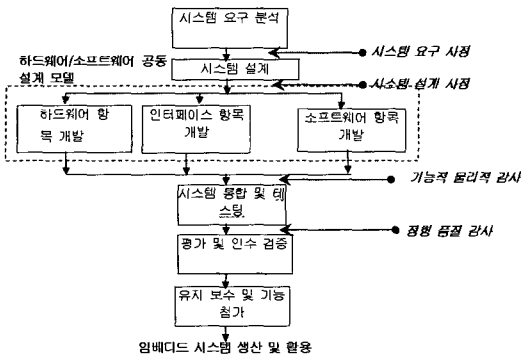
임베디드 시스템 소프트웨어는 공통적으로 하드웨어 개발자(제어 전문가, 기계공학 전문가, 회로 설계 전문가 등)와 소프트웨어 개발자가 공동 작업을 한다. 실제 소프트웨어는 소프트웨어 전문가가 작성하지만, 그 핵심 기능은 하드웨어 및 시스템 설계자가 제시한 이론적 알고리즘(예: 제어 알고리즘, 신호처리 알고리즘 등)에 대한 구현일 뿐이다. 따라서 소프트웨어 개발자는 기본 이론에 대한 이해가 부족하고, 비 소프트웨어 엔지니어는 소프트웨어 플랫폼(운영체제, 디바이스 드라이버, 통신 프로토콜 특성 등)에 대한 이해가 부족해 서로 다른 생각을 할 수가 있다. 이로 인해 매우 쉽게 해결 할 수 있는 문제도 개발자 간의 이해 부족으로 어렵고 복잡한 방법을 택하게 되어 프로젝트를 실패하는 사례도 빈번하게 발생한다.

바) 하드웨어/소프트웨어 동시 설계 및 구현 시 병행 개발 환경 요구

많은 임베디드 시스템은 하드웨어 및 소프트웨어를 동시에 개발해야 한다. 하지만, 기술 특성상 하드웨어가 개발되어야만 그 하드웨어를 기반으로 한 소프트웨어 개발이 진행 될 수 있기 때문에 인력 운영 상의 많은 낭비가 초래되고 있다. 이러한 문제 해결을 위해서 최근에 선진국을 중심으로 개발 하드웨어의 주요 기능을 충실히 수행할 수 있으면서도 쉽게 제작이 가능한 실물 프로토타이핑[2] 기법에 대한 연구가 활발히 진행되고 있으며 국내외적으로 개발 도구에 대한 연구도 진행되고 있다.

IV. 임베디드 실시간 시스템 개발 방법론 제안

본 논문에서 제안하는 임베디드 시스템 소프트웨어 개발 방법론은 처음부터 본 연구진이 고안한 방법론은 아니며, 다양한 환경에서 활용되고 있는 소규모 개발 방법론들을 임베디드 시스템 특성에 적합하게 재 조합한 것이다. 그러나, 다양한 임베디드 시스템 개발에 공통적으로 요구되는 내용들을 고려해 가급적 불필요한 공정을 최소화 할 수 있도록, 소프트웨어 공학의 일반적 이론을 바탕으로 최적화한 방법론을 제시하는 것을 목표로 하였다. 여기서는 하드웨어와 소프트웨어를 통합한 개발 방법론과 소프트웨어 부분만을 고려한 임베디드 소프트웨어 개발 방법론으로 분리하여 서술한다.



〈그림 1〉 임베디드 시스템 개발 프로젝트 생명 주기도

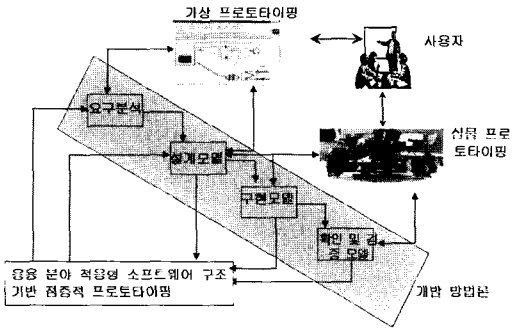
1. 하드웨어/소프트웨어 통합 개발 방법론

<그림 1>은 하드웨어/ 소프트웨어를 통합한 임베디드 시스템 개발 프로젝트의 생명 주기를 도식화 하여 설명한 것이다. 임베디드 시스템 개발 프로젝트의 생명주기는 <그림 1> 에서 보는 바와 같이 크게 6단계로 나눌 수 있으며, 각 단계마다 대응하는 검증 절차들을 가진다. 특히 임베디드 시스템은 설계 단계에서 하드웨어 설계, 소프트웨어 설계, 그리고 필요에 따라 인터페이스(예, 디바이스 드라이버 혹은 펌웨어 등) 설계 단계가 하드웨어/소프트웨어 동시 설계 개념으로 병행 수행되어야 한다. 임베디드 시스템 설계는 기본적으로 하드웨어 요소와 소프트웨어 요소를 동시에 고려해야 하므로, 개발자들은 어떤 기능은 하드웨어로 해결해야 하고 어떤 부분은 소프트웨어로 해결해야 하는 지를 최우선으로 결정해야 한다. 특히 최근에 하드웨어의 소프트웨어화와 소프트웨어의 하드웨어화를 위한 기술들이 발전하면서 이러한 하드웨어 기능과 소프트웨어 기능을 명쾌하게 분리하기가 쉽지 않은 경우가 종종 발생한다. 따라서 이 하드웨어 및 소프트웨어 기능 분리 문제는 정확한 해법이 있는 것은 아니고 단지 최적화의 문제이며, 많은 임베디드 시스템들이 앞에서 정의한 응용분야별 제약 조건에 민감하기 때문에 비 표준화, 철저한 Time-to-market 생존 전략, 그리고 특허 등의 독점적 기술 회피 문제 등 여러 가지 기술 외부적 요구 또한 고려해야 한다.

이러한 상충되는 요구들은 임베디드 시스템 개발을 최적화하는데 악영향을 미치기도 한다. 특히 개발 과정에서 대상 시스템을 위한 CPU 선택은 이러한 하드웨어/소프트웨어 기능 분리 문제 또는 개발 도구 선정 문제 등에 직접적으로 영향을 준다. 역으로 보면 어떤 CPU를 선택하느냐에 따라서 하드웨어/소프트웨어의 기능 분리나 개발 도구의 선택폭에 제한을 받을 수도 있다는 것이다. 하드웨어 및 소프트웨어 기능분리에 따른 설계가 완료되면, 다음 과정으로 분리 개발된 요소 들을 통합해야 하는데, 이 또한 매우 고난도의 개발 방법론과 개발도구를 필요로 한다. <그림 1>의 개발 공정은 설명의 편의상 폭포수 모델의 형태를 취하고 있으나, 일반적으로 모든 소프트웨어 및 하드웨어의 시제품 개발 과정에는 잦은 피드백 과정이 포함된다. 하드웨어의 경우 최근에는 VHDL과 FPGA 등을 이용한 소프트웨어화된 개발 기술 들이 주류가 되면서 이러한 피드백(feed back) 공정이 더욱 활발해지고 있다. 소프트웨어의 경우에 대해서는 다음 절에서 내용을 서술하겠다.

2. 임베디드 소프트웨어 개발 방법론

앞에서 언급한 바와 같이 임베디드 시스템 개발 프로젝트는 단순 범용 소프트웨어 개발 프로젝트에 비해 고난도의 기술을 연구할뿐만 아니



〈그림 2〉 임베디드 소프트웨어 개발 방법론

라, 개발 팀원들 또한 하드웨어와 소프트웨어에 대한 동시 이해 능력을 보유해야만 성공적인 프로젝트 수행이 가능하다. 특히, 범용 소프트웨어 개발 프로젝트에서는 필요에 따라 쉽게 단계적 개선 및 전 과정으로의 피드백이 가능하나, 임베디드 시스템에서는 하드웨어와 소프트웨어의 동시 설계 및 동시 진행으로 인해, 이전 과정으로의 피드백이 불가능하거나 많은 위험요소를 내포할 수 있다. 이러한 위험 요소를 최소화 하기 위해서 본 연구팀에서는 다양한 임베디드 시스템 소프트웨어 개발 경험을 바탕으로 하여 <그림 2>와 같은 개발 방법론을 제안한다.

이 방법론 모델의 큰 특징은 하드웨어 및 소프트웨어의 동시 설계 과정에서의 위험성을 최소화 하기 위하여, 모형 주택과 같은 의미의 폐기형 가상 프로토타이핑(Throw-away style Virtual Prototyping) 기법을 적용한 점이다. 이 기법으로 하드웨어와 소프트웨어 간의 역할 분담 등을 개발 초기 단계에서 명확히 하고, 응용분야 적응형 소프트웨어 구조를 중심으로 한 점증적 프로토타이핑 기법을 개발의 전체 과정에 적용하였다. 이로 인해 소프트웨어는 반복적으로 그 기능을 점진적으로 확장시켜 가면서 완성 할 수 있다. 또한, 개발 중간 단계에서부터 개념 실물 프로토타이핑(Physical Prototyping) 방법론을 도입하여 하드웨어/소프트웨어 동시 개발 환경을 빠르게 구축하고, 개발 소프트웨어의 성능을 수시로 확인하여 다양한 전공의 개발자들 간에 이해 증진을 높일 수 있도록 하였다. 이는 최종 개발 제품의 성능을 개발 중간 단계에서도 예측 할 수 있는 환경을 마련한 셈이다. 각각 단계별 개발 절차를 자세히 설명하면 다음과 같다:

가) 가상 프로토타이핑

가상 프로토타이핑은 임베디드 소프트웨어 개발 초기 단계에서 사용자 혹은 개발 의뢰자의 정확한 요구, 선호 및 취향 등 기능적, 비 기능적 요구들을 효과적으로 추출할 수 있도록 고안된 시스템 요구 및 제약조건 추출 방법론이다. 이 방법론은 마치 아파트 건설회사에서 모형 아파트(모델 하우스)를 통하여 고객들의 선호도를 조사하는 방식과 유사하다. 컴퓨터 화면에 3차원으로 실물과 동일한 모형을 표시하고, 마우스 또는 키보드를 통한 사용자의 입력에 실제 시스템과 유사하게 반응 할 수 있는 동적 처리 기능도 포함하고 있다. 컴퓨터 프로그램으로 프로토타입 모델을 작성하고 시연하기 때문에 사용자의 선호에 따라 가상 모형의 모양을 변경하거나 기능을 새로이 추가하는 등의 변화를 손쉽게 반영 할 수 있다. 따라서, 개발 초기에 사용자의 요구를 효과적으로 수용할 수 있는 개발 방법론으로 평가 받고 있다. 요구 추출 과정이 완료되면 개발된 프로토타입 모형은 모형 주택처럼 폐기되기 때문에 폐기형 프로토타입 모델이라고도 하며, 소프트웨어 분야뿐만 아니라 실물 생산이 필요한 많은 산업분야에서 급속히 채택되고 있는 개발 방법론이다. 가상 프로토타이핑을 지원하는

대표적인 임베디드 시스템 개발 도구로는 Rapid PLUS, MSC-sim, RT-LAB 등이 있으며, 최근에는 UML 등 통합 개발 방법론과 결합된 Computer-Aided Software Engineering (CASE) 형태로 발전하고 있다. i-Logix 사의 Rapsody같은 것이 대표적인 통합 개발 환경이다.

나) 응용분야 적응형 점증적 프로토타이핑과 결합된 개발 방법론

점증적 프로토타이핑 방법론은 나선형 소프트웨어 생명주기 모델에 근거한 대표적인 개발 방법론으로 소프트웨어 개발에서의 위험요소(risk)를 최소화 하기 위해서, 개발하려는 시스템의 주요 기능을 한 번에 동시 개발하지 않고, 주요 기능에 우선순위를 설정하여 우선순위대로 소규모로 핵심 기능들을 구현하고 기능 및 성능에 대한 정확한 평가 후에 추가 기능을 점차적으로 구현하는 개발 방법론이다. 이런 개념적 토대 하에 응용분야 적응형(domain-specific) 소프트웨어 구조(architecture)라는 보다 실용적인 이론을 첨가하면, 다양한 요구 조건과 제약 요건을 가진 임베디드 시스템 개발에 적합한 소프트웨어 개발 구조로 활용할 수 있다. 이 소프트웨어 개발 기법은 많은 이들에 의해 연구 개발 및 실용화가 진전되고 있다. 정보가전 응용분야를 위한 개발 환경인 Philips의 Koala, Arcticus사에서 개발한 Time-Triggered 프로토콜을 기반으로 한 경성 실시간 시스템 개발을 위한 Rubus, 유럽의 GN&C에서 개발중인 DSSA 프로젝트 등이 잘 알려진 사례들이다[3].

다) 실물 프로토타이핑

임베디드 시스템에 공통적으로 활용되는 표준 센서와 엑츄에이터, 그리고 쉽게 조립할 수 있고 재사용 가능한 부품 및 블록들을 활용하여 실제 개발하고자 하는 실물과 유사한 모양과 기능을 가진 실물 프로토타입을 개발하여 소프트웨어 개발과정에 접목하면 다음과 같은 이득을 얻을 수 있다. 우선, 임베디드 시스템 특성상 다양한 전공분야의 개발자들이 공동 작업을 하게 되는데, 서로 간에 이해 증진을 위한 도구로 활용할 수 있으며, 보다 효과적인 문제 해결 방법을 새로이 고안해 내기 위한 도구로써도 활용이 가능하다. 다음으로, 임베디드 시스템은 하드웨어 개발과 소프트웨어 개발이 함께 되어야 하나 종래의 개발 방법론에서는 하드웨어 개발이 완료된 후에야 소프트웨어 개발이 진행되므로 개발인력에 대해 효과적인 활용이 불가능하고, 개발 기간이 길어지며, 소프트웨어 개발 시 발견되는 하드웨어적 결함에 대한 대처 능력이 떨어지는 문제점이 상존하고 있다. 실물 프로토타입을 이용하면 개발 중간에서부터 하드웨어 개발자와 소프트웨어 개발자가 동시작업을 진행하면서, 소프트웨어 적용 시 발생하는 문제점들을 바로 하드웨어 개발자에게 전달하여 최종 제품에 대한 품질을 개발 단계에서부터 보장할 수 있게 된다. 최근에 이러한 실물 프로토타이핑 개념의 중요성이 강조되면서 많은 개발 도구들이 연구 개발되고 있다. 대표적으로 UC Berkeley 의 MICA&TinyOS, EU "Disapearing Computer" 연구팀의 smart-Its, Physical Prototyping Research Consortium, 그리고 본 연구팀에서 개발한 Embedded System Prototyping Suite (ESPS)[4] 등이 있다.

V. 제안된 방법론의 효과 분석

제안된 임베디드 소프트웨어 개발 방법론은

다소 복잡하고 소모적인 공정들이 포함될 가능성도 내재하고 있으며, 또한 이러한 추가공정들로 인해 개발 기간이 길어질 수 있다고 지적할 수 있을 것이다. 하지만 이 방법론을 따라 철저하게 응용분야에 최적화 시키는 작업을 꾸준히 할 경우 그 효과는 매우 클 것으로 판단된다. 이 방법론을 산업체의 임베디드 시스템 제품 생산에 적용 했을 때의 효과와 이를 임베디드 시스템 교육 과정에 적용했을 때의 효과로 나누어 분석해보면 다음과 같다.

가) 산업체 적용 시 효과

실제 임베디드 시스템 개발에 본 개발 방법론을 적용할 경우 초기에는 약간의 지연 시간이 있을 수 있으나, 일정 시간이 경과하고 유사 제품 개발에 대한 경험이 축적될 경우 매우 중요한 time-to-market의 경쟁력을 확보 할 것으로 판단된다. 본 방법론을 따르면 개발 초기 및 중기에 제품에 대한 기능 및 성능을 보장 할 수 있도록 작업이 진행되므로, 최종 제품의 질이 여타 방법론에 비해 확실히 보장이 되며 이로 인해 제품 개발 기간의 감축 효과뿐만 아니라 프로젝트의 위험요소를 최소화하는 효과까지 얻을 수 있다. 또한 프로토타이핑 모델을 통하여 다양한 분야의 전문가들이 동시 개발을 진행하면서 문제 해결 방법에 대한 이해의 폭을 넓힐 수 있기 때문에 매우 효과적인 개발 환경을 구축할 수 있다.

특히 프로토타입을 통해서 설계 과정들 간에 강한 연결 관계를 설정하기 때문에 설계 내용과 구현 내용이 일치하지 않는 등의 문제를 야기할 가능성이 적다는 큰 장점도 있다.

나) 교육과정 적용 시 효과

고급 임베디드 시스템 개발 인력 확보를 위해서는 개발 인력들이 하드웨어 개발에서부터 소프트웨어 개발까지 전 과정을 경험해 보는 것이 매우 중요하다. 하지만 임베디드 시스템 특성상 실제 개발되는 임베디드 시스템을 직접 강의실 수준에서 체험해보기는 곤란하다. 따라서 각종 프로토타이핑 도구를 연결한 모형 임베디드 시스템을 개발해보면, 실제 시스템 제작과 동일한 경험을 할 수 있는 장점이 있다. 또한 다양한 전공 혹은 이론을 직접 프로토타입에 적용해 봄으로써 학제적 연구를 위한 모델이 될 수도 있다. 이러한 제안 방법론의 우수함을 바탕으로 하여 국제적으로 우수한 연구기관들을 중심으로 실물 프로토타이핑을 이용한 임베디드 시스템 개발 도구가 많이 개발되었으며 실제 교육에 활용되고 있다[5].

VI. 실물 프로토타이핑 기반의 임베디드 시스템 개발 교육 환경

앞 장에서 살펴보았듯이 임베디드 시스템은 응용분야에 따른 다양한 제약조건들을 만족해야 함과 동시에 다양한 서비스를 제공할 수 있어야 하기 때문에 컴퓨터 구조 및 운영체제 탑재 등의 측면에서 범용 컴퓨터와 유사한 내부 구조를 가지고 있으면서도 차별화된 시스템 개발 방법론과 개발 기법을 적용해야 한다. 이러한 임베디드 시스템 개발의 복잡성은 현장의 엔지니어들에게 하드웨어 소프트웨어 동시 설계 능력, 시스템 소프트웨어에 대한 폭 넓은 이해, 그리고 소프트웨어 공학론을 기반으로 한 완고한 시스템 설계 및 구현 능력 등 전문적이고 체계적인 기술을 요구한다.

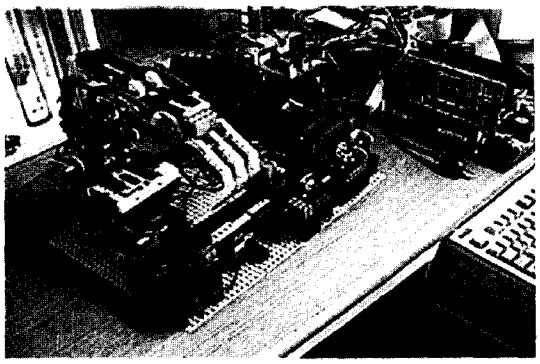
그러나 이러한 전문 인력 양성을 위한 체계적인 교육 과정을 구축하고 실용적인 교육을 하기

에는 기술적 난관이 산재해 있다. 즉, 임베디드 시스템은 기본적으로 타겟 시스템(예: 항공기, 발전소, 자동차 등)에 내장되어 동작하는 시스템임에도 불구하고 그 타겟 시스템을 학교나 일반 교육장의 실험실에서 접근하기는 용이하지 않다. 또한, 모형이 있다 하여도 실용되는 타겟 시스템의 내부 동작 원리를 파악하는 것은 쉽지 않고, 다양한 종류를 구비하거나 재사용하기가 곤란하며, 하드웨어 소프트웨어 동시 설계, 실시간 운영체제 기반의 디바이스 드라이버 제작, 그리고 소프트웨어 공학적 설계 및 구현 방법론 등 관련 이론 및 기술들을 체계적으로 집약하여 교육하기가 쉽지 않다.

하지만, 본 연구진은 최적화된 임베디드 시스템 교육과정을 개발하기 위해 많은 노력을 기울였다. 이러한 노력의 결과로 재구성 가능한 완구인 Lego Dacta[6] 모형 혹은 Fischertechnik 사의 fischer 모형[7]을 실물 프로토타이핑 전용 모델로 활용하고 관련 하드웨어와 소프트웨어를 추가 개발하여 대학 또는 대학원에서 임베디드 시스템을 교육할 수 있는 실험 도구를 개발하였다[8]. 다음 절에서는 이 도구를 이용하여 앞 장에서 제시한 임베디드 소프트웨어 개발 방법론을 기반으로 하여 개발 전 과정을 체험 해볼 수 있는 실습 과정을 제시하였다. 이를 통해 임베디드 시스템 교과 과정에 대한 보다 실용적인 교육을 위한 실험 도구와 그것을 이용한 실습 과정을 제안하고자 한다.

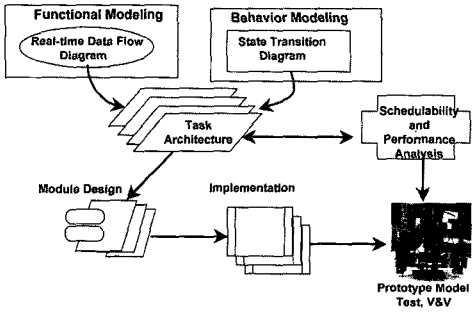
1. 실물 프로토타입 모델 제작

이 교육과정에서는 ESPS 도구에서 제공되는 API와 레고 혹은 피셔 블록 및 관련 센서/엑츄에이터를 이용하여 임베디드 실시간 시스템의 실물 프로토타이핑 타겟 모델을 제작하게 된다. 어떤 시스템을 제작할 것인지는 학생들이 직접 창의적인 아이디어를 내어 결정하게 되며, 교육 여건을 고려해 폐기형 가상 프로토타이핑 모델 제작은 생략하고, 바로 레고 혹은 피셔를 이용해 실물 프로토타이핑 모델을 제작한다. <그림 3>은 실례로 제작된 실물 프로토타이핑 모델로 공장 자동화 시스템의 로봇 팔과 컨베이어 벨트 시스템을 레고로 재현한 것이다.

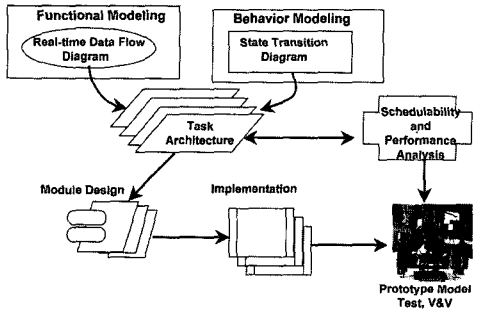


〈그림 3〉 로봇 팔과 컨베이어 벨트를 이용한 공장 자동화 시스템 프로토타입

이 시스템은 컨베이어 벨트로 운반하는 블록이 합격인지 불합격인지를 파악하여, 불합격인 경우 로봇 팔로 집어서 버린다. 3개의 광 센서와 4개의 회전 센서, 4개의 모터를 사용하며, 3개의 광 센서는 작업의 시작과 끝을 확인하고, 운반하는 블록의 색을 감별해 합격, 불합격의 판정을 내리는데 이용되며, 4개의 모터는 컨베이어 벨트와 로봇 팔의 움직임을 위해 사용된다. 회전 센서는 모터의 회전 수를 기준으로 벨트 및 팔의 움직임을 파악하고 제어하기 위해 쓰인다. 로봇 팔은 상하 좌우로 움직일 수 있으며, 손을 오므리고 펴는 것이 가능하다.



〈그림 4〉 Software Engineering Process



〈그림 5〉 정보은닉모듈을 포함한 태스크구조도

2. 점증적 소프트웨어 프로토타이핑 절차

실물 모델을 제작한 후에는 본격적으로 실시간 시스템 응용 프로그래밍을 위한 소프트웨어 공학론에 따르는 작업이 진행된다. 여기서는 CODARTS[9]방식을 따라서 진행하였다. 이 방식에 의한 프로그램 개발 절차는 <그림 4>처럼 도시할 수 있다. 즉, 타켓 시스템의 요구 사항을 분석하고 그것을 해결하기 위한 프로그램 구조를 디자인하며, 설계된 프로그램의 구조에서 스케쥴링 가능성과 기타 문제점을 분석 및 보완하여, 최종적으로 <그림 5>과 같은 소프트웨어 아키텍쳐를 도출하고, 그에 따라 프로그램을 구현하는 과정을 거친다. CODARTS방식에 따른 개발과정에 대한 자세한 사례 소개가 필요하다면 이전에 발표한 논문[10]을 참조하기 바란다.

이와 같은 요구사항 분석 및 설계 과정 자체는 이전에 흔히 시도되던 방식들과 크게 다르지가 않다. 그러나 여기서는 미리 실물 프로토타이핑 모델을 제작한 상태에서 작업을 진행한다는 점에서 차이를 가진다. 제작한 모델 덕분에 요구사항을 구체화할 수 있으며, 제어 및 감시해야 될 대상이 무엇인지 알 수 있고, 시스템의 물리적인 한계 사항을 구체적으로 이해한 상태로 프로그램 구조를 설계할 수 있는 것이다. 그리고 설계 과정에서 지속적으로 모델을 운영하는데 타당한 소프트웨어 구조를 모색하면서 점증적인 구체화가 가능하여 좀 더 최적화된 구조를 얻어낼 수 있다.

3. 응용 프로그램 구현 및 검증

분석, 설계와 검증이 완료되면 남은 일은 최종적으로 응용 프로그램을 작성하여 타켓 시스템에 적용하는 과정이다. 소프트웨어 작성은 디자인된 소프트웨어 구조를 바탕으로 진행하게 되지만, 구현 중에도 지속적인 보완 및 개선이 이뤄져 최종적인 프로그램 구조는 작성하는 환경에 따라 일부 변경될 수도 있다.

이렇듯 앞에서 이론적으로 디자인한 구조를 실제로 적용하는 과정에서 실습자는 미리 제작해 둔 실물 모델과 프로그램을 연동시켜서 작업을 진행할 수 있다. 응용 시스템의 목표에 따라, 프로그램을 일부 변경할 수도 있고, 역으로 실물 모델을 개선된 구조로 만들 수도 있다. 또한 설계 과정에서 도출된 스케줄 가능성 분석 자료를

바탕으로 하여 실제 구현 결과가 요구사항을 만족하는 지도 검증하게 된다.

VII. 결 론

본 논문에서는 임베디드 실시간 시스템 소프트웨어 개발 분야의 고충과 문제점을 살펴보고, 이를 해결하기 위한 여러 연구들을 소개하였다. 그리고 최근에 임베디드 실시간 시스템 소프트웨어 개발 과정에서 각광받고 있는 실물 프로토타이핑 모델 방법을 이용한 임베디드 소프트웨어 개발 방법론을 소개하고, 본 연구진이 개발한 내장형 실시간 시스템 교육 및 실습을 위한 도구를 이용해 실제 구현하는 과정을 소개하였다. 이에 따르면, 가상 프로토타이핑 모델을 컴퓨터 상에서 먼저 실현하여 사용자들의 생생한 요구를 수렴하고 이를 이용하여 기본적인 요구 분석 및 설계 작업이 시행된다. 이 과정이 어느 정도 구체화 되면 실물 프로토타이핑 모델을 제작하여 개발 중간 단계에서 최종 제품에 대한 기능 및 성능을 보장할 수 있는지 검증하는데 이용할 수 있으며, 이러한 검증 단계를 지속적으로 거치면서 프로그램 구조를 설계하고 구현하는 과정이 점증적으로 반복하며 진행하게 된다. 따라서 제시된 도구와 개발 방법론을 이용하면 응용 모델에 알맞은 점증적인 개발 과정을 거칠 수 있다. 이러한 내용을 임베디드 실시간 시스템 개발 현장뿐만 아니라 실험 교육에도 효과적으로 적용 가능하며, 시스템 개발에 필요한 모든 단계를 망라해 볼 수 있어서 교육 효과가 매우 높을 것으로 예상된다.

본 도구와 개발 방법론을 교육에 이용하면 학생들이 하드웨어와 소프트웨어를 동시에 설계 및 구현할 수 있는 기회를 가지게 되고, 재사용이 가능한 레고 블록을 이용하여 다양한 형태의 실물 임베디드 실시간 시스템 모형을 스스로 제작해 볼 수 있으며, 추상적으로 배우던 최신의 임베디드 실시간 시스템 개발 방법론의 실체를 체득할 수 있다. 또한, 경우에 따라서는 산업체에서 실제 개발해야 하는 실시간 시스템의 프로토타입 모델 개발에도 이용할 수 있다.

이 도구는 현재 경북대학교 전자 전기 컴퓨터학부에서 내장형 시스템 실험 과목과 대학원 전자공학과 실시간 시스템 설계 과목에서 실험 도구로 활용하고 있다.

참고문헌

[1] Philip Koopman, "Embedded System Design Issues (the Rest of the Story)," Proceedings of the 1996 International Conference on Computer Design (ICCD96), October, 1996.

[2] Anneke Vandevelde, Roland Van Dierdonck and Bert Clarysse, "The Role of Physical Prototyping in the Product Development Process," Vlerick Leuven Gent Management School Working Paper Series 2002-7, 2002.

[3] Anders Möller et al., "An Industrial Evaluation of Component Technologies for Embedded Systems," MRTC Report ISSN 1404-3041 ISRN MDH-MRTC-155/2004-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, February 2004.

[4] ART System, http://www.art-system.co.kr/

[5] Rapid Design Through Virtual and Physical Prototyping Project, http://www-2.cs.cmu.edu/~radproto/

[6] Lego, http://www.lego.com

[7] fischertechnik, http://www.fischertechnik.com/

[8] 박성호, 이상문, 박동환, 강순주, "실시간 시

스템 교육 및 실험 Tool Kit 개발," KISS`2000 Spring Conf., 2000년 4월.

[9] Hassan Gomaa, Software Design Methods for Concurrent and Real-Time Systems, Addison-Wesley, 1996.

[10] 정기훈, 김도훈, 박성호, 강순주, "임베디드 실시간 시스템 개발 교육 과정," 정보처리학회지, 제 9권, 제 1호, pp.103-111, 2002년 1월.

저자소개

정 기 훈

2001년 경북대학교 전자전기공학부 학사
2003년 경북대학교 전자공학과 석사
2003년 - 현재 경북대학교 디지털기술연구소 연구원
2004년 - 현재 경북대학교 전자공학과 박사과정
주관심분야 Real-Time System, System Software, Home Network, IEEE1394

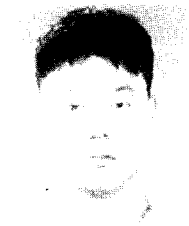
채 화 영

2004년 경북대학교 전자전기공학부 학사
2004년 - 현재 경북대학교 전자공학과 석사과정
주관심분야 Real-Time System, Home Network, LonWork Network

저자소개

김 정 길

2004년 경북대학교 전자전기공학부 학사
2004년 - 현재 경북대학교 전자공학과 석사과정
주관심분야 Real-Time System, Home Network, USB, PCI Device Driver



이 재 신

2004년 경북대학교 전자전기공학부 학사
2004년 - 현재 경북대학교 전자공학과 석사과정
주관심분야 Real-Time System, Control Area Network, Autonomous Flying System

강 순 주

1983년 경북대학교 전자공학과 학사
1985년 한국과학기술원 전자계산학과 석사
1995년 한국과학기술원 전자계산학과 박사
1985년 - 1996년 한국원자력연구소 연구원, 핵인공지능연구실 선임 연구원, 전산정보실 실장
2000년 - 2002년 University of Pennsylvania, Dept. of Computer and Information Science, 객원연구 교수
1996년 - 현 재 경북대학교 전자전기컴퓨터학부 정보통신전공 부교수
주관심분야 Real-Time System, Software Engineering, Knowledge-Based System