2001년도 대한전기학회/대한전자공학회 시스템및제어분야 합동 추계학술대회 논문집 (2001.11.24)

내장형 제어 시스템 개발을 위한 래피드 프로토타이핑 구현에 관한 연구

송정현*, 이재인*, 선우명호**
*한양대학교 전기공학과, **한양대학교 자동차공학과

Development of a Rapid Controller Prototyping System for Embedded Control Systems

Junghyun Song*, Jaein Lee*, Myoungho Sunwoo**
*Dept. of Electrical Engineering and **Dept. of Automotive Engineering, Hanyang Univ.

Abstract - Pressures on the development of embedded control systems with higher performance and quality are increasing now, and the development of embedded control systems are still difficult. Hence, fast and reliable development method for implementing embedded control systems are necessary. Rapid prototyping is a powerful method to solve these problems. With the use of this method, the graphical representation of a model is converted to code that executes the model. In this paper, the development procedure of embedded control systems has been automated using this new technology. Also, an application example about lambda control in automotive engine has been carried out.

1. 서 론

내장형 제어 시스템은 지난 수십 년 간 산업계의 여러 분야에서 다양한 형태로 사용되어 왔으며, 디지털 제어기의 급속한 발전으로 인하여 보다 복잡하고 향상된 제어 알고리즘을 실시간으로 구현하여 검증할 수 있게 되는 등 그 중요성이 더욱 커지고 있다. 그러나 내장형 제어 시스템을 구현하기 위해서는 제어 이론은 물론 하드웨어 및 소프트웨어 디자인 이론 등이 필요하고 고성능, 고내구성의 내장형 제어 시스템에 대한 요구가 증가함에 따라 새로운 내장형 제어 시스템 개발 방법이 필요하게 되었다.[1][2]

이 논문에서는 내장형 제어 시스템 개발 방법으로서 래피드 프로토타이핑 기법을 적용하여 복잡하고 향상된 제어 알고리즘을 실시간으로 제어기에 탑재하여 제어 성능을 검증하는 것을 목적으로 한다.

이를 위하여 엔진 제어를 위한 장치관리자(device driver), 응용 프로그래밍 인터페이스(API), 장치관리자 도구상자(device driver toolbox) 및 프로그램 자동 생성을 위해 필요한 모듈을 구현한다.

마지막으로, 구현된 구성 요소 및 래피드 프로토타이핑 기법을 적용하여, 응용 예로서 자동차 엔진 제어를 위한 공연비(air/fuel ratio) 제어기를 구현하여 실험한다.

2. 본 론

2.1 래피드 프로토타이핑의 개념

래피드 프로토타이핑은 도식적으로 표현된 제어기 모델을 실시간 하드웨어 즉, 디지털 제어기에 정형화된 방법으로 구현하는 절차로서 정의할 수 있다.[2] 이 방법의 가장 큰 특징은 제어 프로그램의 자동 생성에 있다. 제어 알고리즘을 수동으로 프로그램화하는 대신 정형화된 절차를 이용하여 프로그램의 코딩, 컴파일, 링크 과정을 수행함으로써 제어 시스템의 개발 기간을 단축할 수 있고 또한 수동 작업 과정에서 발생할 수 있는 오류의 발생도 줄일 수 있게 된다.

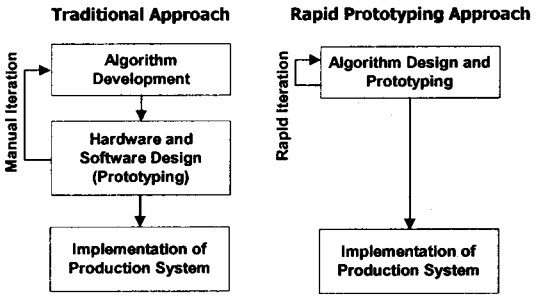

그림 1. 내장형 제어 시스템의 개발 방법 비교

그림 1은 전통적인 내장형 제어 시스템 개발 방법과 래피드 프로토타이핑 방법을 비교한 것이다. 그림에서 보는 바와 같이 전통적인 방법에서는 제어기 구현을 위해 수동으로 작업을 반복하는 반면 래피드 프로토타이핑 방법에서는 수동 작업이 최소화되어 사용자의 오류 또한 최소화되며 제어 시스템 설계자는 제어기 모델에 더욱 관심을 쏟을 수가 있게 된다.[5]

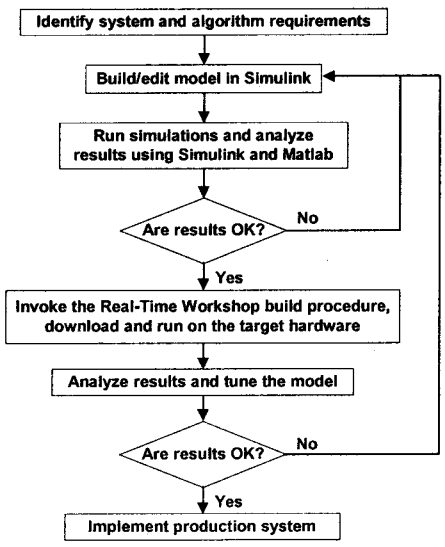

그림 2. 래피드 프로토타이핑 구현 절차

그림 2는 본 연구에서 채택한 래피드 프로토타이핑 개발 절차를 나타내고 있다.

현재 래피드 프로토타이핑을 적용한 내장형 제어 시스템 개발을 위해 다양한 도구가 사용되고 있으며, 그 예로서 MathWorks사의 Matlab 제품군, RTI사의 ControlShell, dSPACE사의 TargetLink, ETAS사의 ASCET-SD 등이 있다.

2.2 시스템 구조

본 논문에서 엔진 제어 시스템을 위해 구성한 하드웨어 및 소프트웨어 구조는 다음과 같다.

2.2.1 하드웨어 구조

내장형 제어 시스템을 개발하기 위해서는 다양한 하드웨어 장치가 필요하며, 그림 3은 본 연구에서 래피드 프로토타이핑을 구현하기 위해 구성한 하드웨어 시스템을 개략적으로 나타낸 것이다.

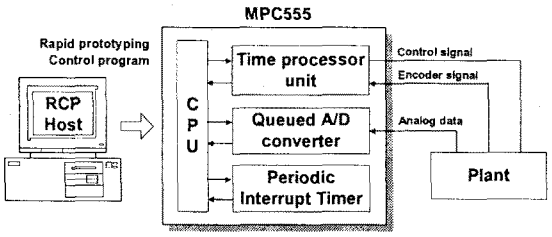

그림 3. 하드웨어 시스템의 개략도

MPC555는 32비트 마이크로컨트롤러로서 부동소수점 연산장치를 내장하고 있어 제어 프로그램의 수행에 유리하다.[7] TPU(Time Processor Unit)는 타이밍(timing) 입/출력을 위한 모듈이며, QADC(Queued A/D Converter)는 A/D 변환을 위한 모듈이다. RCP Host PC에서 래피드 프로토타이핑 기법을 적용하여 생성된 제어 프로그램은 컨트롤러에 다운로드 되어 플랜트를 제어하게 된다.

2.2.2 소프트웨어 구조

각각의 프로그램 모듈은 그림 4와 같은 층 단위 구조(layered architecture)를 이루게 된다. 아래층의 구조는 위층의 구조에서 필요로 하는 기능을 제공하며, 이를 통하여 응용 프로그램 작성의 유연성이 증가하고, 소프트웨어의 모듈별 유지/보수/성능 향상이 쉬워지며, 소프트웨어의 재사용성이 증가한다.

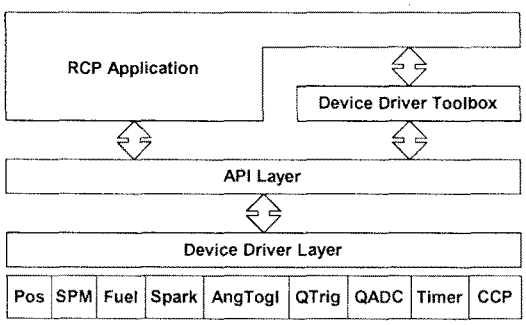

그림 4. 층 단위 소프트웨어 구조

2.3 소프트웨어 모듈의 구현

래피드 프로토타이핑 기법을 엔진 제어 시스템에 적용하기 위해 필요한 소프트웨어 모듈을 다음과 같이 구현하였다.

2.3.1 장치관리자

제어 프로그램을 실시간 하드웨어 장치에서 수행하기 위해서는 각종 입/출력, A/D 변환, 타이머(timer) 등의 기능이 필요하며 이를 위하여 하드웨어 관련 작업을 수행하는 장치관리자를 기능별로 구현하였다. 이 때, 장치관리자의 성능 향상 및 신뢰성을 높이기 위하여 Motorola사의 Low-Level Driver를 사용하여 장치관리자 계층을 구성하였다.[8] 특히, 엔진 제어 시스템에서는 각종 신호의 동기화 및 이벤트 처리가 중요한 요소이므로 이러한 요구조건을 만족하도록 적절하게 장치관리자를 구성하였다.

2.3.2 응용 프로그래밍 인터페이스

일반적으로 장치관리자가 하드웨어를 제어하기 위해 사용하는 매개변수(parameter)는 그 수가 많을 뿐만 아니라 그 의미도 하드웨어에 의존적인 경우가 많으므로 일반 사용자가 쉽게 사용하는데 어려움이 있다. 따라서 제어에 필요한 데이터의 연산 및 보정, 인코더 신호의 처리, 연료 분사 및 점화, A/D 변환 등의 작업을 보다 쉽게 하도록 하기 위한 도구로서 다양한 인터페이스 함수를 구현하여 응용 프로그램 작성의 편의를 도모하였다.

2.3.3 장치관리자 도구상자

래피드 프로토타이핑 방법을 적용하여 정형화된 방식으로 프로그램을 생성하기 위해서, 장치관리자는 제어기 모델과 도식적으로 연결되어야 한다. 따라서, 시뮬링크 모델을 구현하는 과정에서 사용자가 직접 하드웨어와 관련된 작업을 포함할 수 있도록 하기 위해서 장치관리자를 시뮬링크 도구상자의 형태로 구현하였다.[4] 이를 통하여 사용자는 보통의 시뮬링크 블록을 사용하는 것과 같이 장치관리자 도구상자를 제어기 모델에 직접 연결하여 하드웨어와 관련된 기능을 포함한 모델을 만들 수가 있게 된다. 제어기 모델과 연결된 시뮬링크 장치관리자 블록은 프로그램 생성 과정에서 실제 하드웨어 관련 프로그램 코드로 변환된다. 또한, 본 연구에서는 계층별 소프트웨어 구조를 채택하였으므로 장치관리자 도구상자의 추가가 용이하다.

2.3.4 자동화 과정 설정 모듈

제어 프로그램을 시뮬링크 모델로부터 자동으로 생성하기 위해서는 시스템 타겟 파일(system target file), 템플릿 메이크파일(template makefile), 메이크 명령(make command)과 같은 프로그램 생성 과정을 제어하는 소프트웨어 모듈이 필요하다.[5][6]

본 연구에서는 타겟 하드웨어로 사용된 Motorola사의 MPC555 컨트롤러 및 Diab C 컴파일러에 적합하도록 위의 모듈을 구성하여 제어 프로그램 생성 과정을 자동화하였다.

2.4 응용 예

앞서 구현한 장치관리자, 응용 프로그래밍 인터페이스, 장치관리자 도구상자, 시뮬링크 모델 등을 이용하여 래피드 프로토타이핑 기법의 적용 예로서 자동차 엔진의 공연비 제어를 위한 내장형 제어 시스템을 구현하였다.

2.4.1 제어기 모델

날로 강화되고 있는 자동차 연비 및 배기가스 규제에 대처하기 위해 엔진 제어 시스템에서 정상상태 또는 과도상태에서의 공연비 제어가 중요한 문제로 대두되고 있다. 따라서 본 논문에서는 그림 5와 같은 이산(discrete-time) PI 제어기를 사용하여 제어 시스템을 구성하였다.

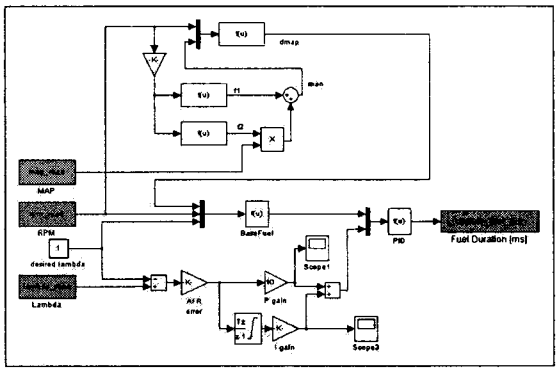

그림 5. 공연비 제어기 모델

위의 그림에서 어둡게 그려진 블록이 실제 하드웨어 입/출력을 위해 앞서 구현한 장치관리자 도구상자를 나타내고 있다.

2.4.2 래피드 프로토타이핑

Matlab Real-Time Workshop의 프로그램 생성 기능을 이용하여 제어기 모델의 실행 프로그램을 생성하였다.[5] 이 때 앞서 구현한 소프트웨어 모듈 및 프로그램 생성 과정을 제어하기 위해 변경된 시스템 타겟 파일 및 템플릿 메이크파일, 메이크 명령 등을 사용하였다. 프로그램 자동생성 과정을 통하여 제어기 모델에 대한 실행 파일을 얻을 수 있게 된다.

2.4.3 프로그램 검증

생성된 제어 프로그램의 성능을 검증하기 위해, 먼저 제어기 모델을 시뮬링크 상에서 오프라인(off-line)으로 시뮬레이션 하였으며 그 결과는 각각 그림 6, 7, 8과 같다. 그림 5에서 제어 모델은 엔진 플랜트가 주기적으로 팁-인(tip-in), 팁-아웃(tip-out)을 하는 동안 수행되며 다음의 결과 그림은 팁-아웃 시의 제어 수행 결과를 나타내고 있다.

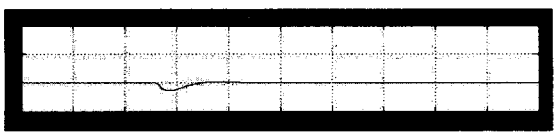

그림 6. 공연비 (시뮬레이션 결과)

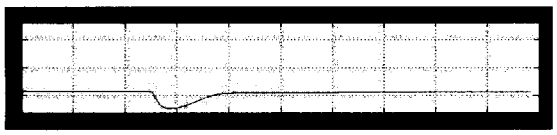

그림 7. 연료 분사량 (시뮬레이션 결과)

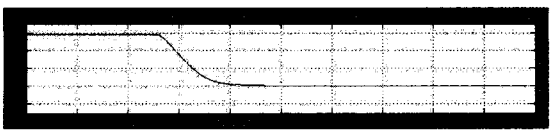

그림 8. RPM (시뮬레이션 결과)

다음의 그림 9, 10, 11은 래피드 프로토타이핑 기법을 적용하여 시뮬링크 모델로부터 자동으로 생성한 제어 프로그램에 의해 공연비 제어가 수행된 결과이며, 엔진 시뮬레이터의 스코프(scope) 창에 나타난 파형을 보여주고 있다.

그림 9. 공연비 (RCP 결과)

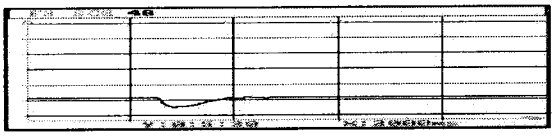

그림 10. 연료 분사량 (RCP 결과)

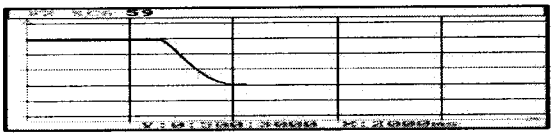

그림 11. RPM (RCP 결과)

위에서 보는 바와 같이 래피드 프로토타이핑을 적용하여 생성한 제어 프로그램의 수행 결과가 제어기 모델의 오프라인 시뮬레이션 결과와 거의 일치함을 알 수 있다.

3. 결 론

이 논문에서는 새로운 내장형 제어 시스템 개발 방법의 하나인 래피드 프로토타이핑 기법을 적용하여 제어 시스템을 구현하였다. 이를 통하여 제어 시스템 구현의 편의를 증대하였으며, 나아가 제어 시스템의 개발 기간과 구현 과정에서의 오류 발생을 줄일 수 있었다.

이러한 내장형 제어 시스템 구현 방법은 엔진 제어뿐만 아니라 전기, 전자, 통신 등의 많은 분야에서 활발히 응용되고 있으며, 앞으로 그 적용 범위가 더욱 확대될 것이며 이에 관한 연구 또한 활발히 이루어질 것으로 예상된다.

[참 고 문 헌]

[1] Woo-Seng Gan, Young-Kim Chong, Wilson Gong, Wei-Tong Tan, "Rapid Prototyping System for Teaching Real-Time Digital Signal Processing", IEEE transactions on education, vol.43, pp.19-24, 2000

[2] AW Stylo, G Diana, "An Advanced Real-Time Research and Teaching Tool for Design and Analysis of Control". Africon, vol.1, pp.511-515, 1999

[3] H.Hanselmann, U.Kiffmeier, L.Koster, M.Meyer, A.Rukgauer, "Production Quality Code Generation from Simulink Block Diagrams", Proc. of the IEEE international symposium on computer-aided control system design, pp.213-218, 1999

[4] Sven RebeschieB, "MIRCOS-Microcontroller Based Real Time Control System Toolbox for use with Matlab/Simulink", Proc. of the IEEE international symposium on computer-aided control system design, pp.256-272, 1999

[5] MathWorks Inc., "Real-Time Workshop User's Guide ver.4"

[6] MathWorks Inc., "Target Language Compiler Reference Guide ver.4"

[7] Motorola Inc., "MPC555 User's Manual"

[8] Motorola Inc., "Motorola Low-Level Drivers User's Manual"

[9] Diab Data Inc., "D-CC & D-C++ Compiler Suites User's Guide ver.4.3"