Transactions of the Korean Society of Automotive Engineers, Vol.11, No.1, 2003

엔진 제어시스템을 위한 래피드 콘트롤 프로토타이핑 플랫폼에 관한 연구

송 정 현[1] · 이 우 택[1] · 선 우 명 호[*2]

한양대학교 대학원[1] · 한양대학교 자동차공학과[*2]

Development of a Rapid Control Prototyping Platform for Engine Control System

Junghyun Song[1] · Wootaik Lee[1] · Myoungho Sunwoo[*2]

[1]*Graduate School, Hanyang University, Seoul 133-791, Korea*

[*2]*Department of Automotive Engineering, Hanyang University, Seoul 133-791, Korea*

(Received 6 August 2002 / Accepted 29 October 2002)

Abstract : The design and implementation of an engine control system has become an important area in developing a new car, but the implementation of an engine control system is becoming a tedious and time-consuming work as the level of complexity increases. In order to shorten the development cycle of the control system, rapid control prototyping (RCP) technique deserves developers' attention.

A new RCP platform has been developed for an automotive engine control application. This prototyping system strictly adheres to the layered architecture of the final production ECU, and separates the automatically generated part of software, or the application area, from the hand coded area, which generally carefully designed and tested because of the hardware dependency and the efficiency of microcontroller. The Matlab® tool-chain of Mathworks Inc. has been selected as a base environment in this study. A newly developed Engine Control Toolbox of Real-Time Workshop® converts a graphically represented control algorithm into optimized application codes and links them with other parts of the software to generate executable code for the target processor.

Key words : Embedded control system(내장형 제어시스템), Rapid control prototyping(RCP, 래피드 콘트롤 프로토타이핑), Automatic code generation (자동 코드 생성)

1. 서 론

제어기술 및 전자기술의 급속한 발전으로 인하여 보다 복잡하고 향상된 제어 알고리즘을 이용할 수 있는 엔진 제어시스템이 등장하게 되었다. 이러한 제어 알고리즘을 구현하기 위해서는 제어이론은 물론 하드웨어 및 소프트웨어 디자인 이론과 개발 도구들에 관한 지식 등 다양한 분야의 관련 지식이 필요하다. 제어 알고리즘이 복잡해질수록 제어기 구현에 긴 시간이 필요하게 되며 오류가 발생할 가능성이 점점 더 커지게 된다. 또한 자동차 시장에서의 극심한 경쟁으로 인하여 엔진 제어시스템의 개발 기간 단축에 대한 요구도 커지고 있다.

따라서, 고성능의 제어시스템을 보다 짧은 기간 내에 효율적으로 개발하기 위한 새로운 개발 방법으로서 래피드 콘트롤 프로토타이핑(RCP, rapid controller prototyping) 기법이 등장하게 되었

*To whom correspondence should be addressed.
msunwoo@hanyang.ac.kr

다.[1-4]

이 연구를 통하여 복잡한 제어 알고리즘을 빠르고 쉽게 실제의 마이크로콘트롤러에 구현하여 검증하기 위한 래피드 콘트롤 프로토타이핑 플랫폼을 개발하였다. 이 플랫폼의 소프트웨어는 계층 구조에 따라서 구성하였다. 자동 코드 생성 기법에 의하여 생성되는 응용프로그램 영역과 하드웨어에 의존적인 영역으로 분리하고 도구상자를 개발하여 두 영역을 손쉽게 연결할 수 있도록 하였다. 응용 예로서 자동차 엔진의 공연비 제어기를 구성하고 실험을 수행하였다.

2. 래피드 콘트롤 프로토타이핑의 개념

래피드 컨트롤 프로토타이핑은 도식적으로 표현된 제어기 모델을 제어기에 자동화된 방법으로 구현하는 절차로 정의된다.[1,2] Fig. 1은 래피드 콘트롤 프로토타이핑을 사용한 제어시스템 개발과정의 개념도이다. 이 방법의 가장 큰 특징은 제어 프로그램의 자동 생성에 있으며 제어 알고리즘을 수동으로 프로그래밍하는 대신 자동화된 절차를 이용하여 코딩, 컴파일, 링크 과정을 수행함으로써 제어시스템의 개발 기간 단축 및 수동 프로그래밍 과정에서의 오류 발생을 줄일 수 있게 된다.

Fig. 2는 전통적인 내장형 제어시스템의 구현 방법과 래피드 콘트롤 프로토타이핑 기법을 비교한 것이다. 그림에서 보는 바와 같이 전통적인 방법에서는 제어기 구현을 위한 여러 단계를 수동으로 반복하여야 하는 반면, 래피드 콘트롤 프로토타이핑 기법에서는 하드웨어와 소프트웨어 설계에 관한 부분이 자동으로 수행되어 수동 작업이 최소화되고 프로그램 작성시 발생할 수 있는 오류가 최소화된다. 따라서 설계자는 제어 알고리즘 설계에 더욱 관심을 쏟을 수가 있게 된다.

따라서 현재 래피드 콘트롤 프로토타이핑 기법에 대한 학계 및 산업계의 관심이 고조되고 있으며, 이 기법을 채용한 다양한 도구들이 개발되고 있다. 그 예로서, MathWorks사의 Real-Time Workshop, RTI사의 Controlshell, dSPACE사의 TargetLink, ETAS사의 ASCET-SD 등이 있다.

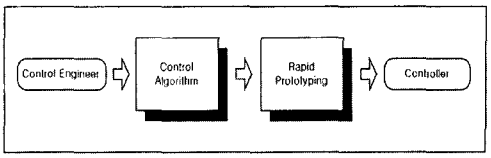

Fig. 1 Diagram of rapid control prototyping

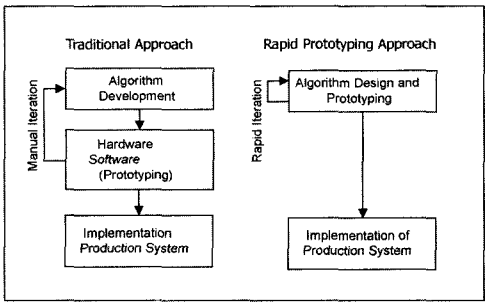

Fig. 2 Comparison of developing procedures for embedded control system

3. 시스템 구조

이 연구에서는 모토로라(Motorola)사의 MPC 5555)를 CPU로 사용하였으며 프로그램 자동 생성 기능을 위하여 MathWorks사의 Matlab, Simulink,[6] Real-Time Workshop[7]을 사용하였고, 컴파일러로 Diab사의 컴파일러[8]를 이용하였다.

3.1 하드웨어 구조

Fig. 3은 이 연구에서 엔진 제어시스템 개발을 위

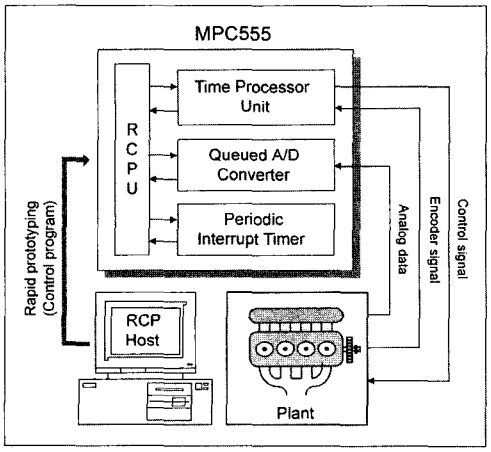

Fig. 3 Hardware block diagram

하여 구성한 하드웨어 시스템을 개략적으로 나타낸 것이다.

PC555는 32비트 마이크로컨트롤러로서 부동소수점 연산장치를 내장하고 있어 제어 프로그램의 수행에 유리하다. 또한 내장 되어 있는TPU(Time Processor Unit)는 타이밍(timing) I/O를 위한 모듈이며, QADC(Queued A/D Converter)는 A/D 변환을 위한 모듈이다. RCP Host PC에서 래피드 프로토타이핑 기법을 이용하여 제어 프로그램을 생성하고, 실행화일을 마이크로콘트롤러에 다운로드하여 수행하게 한다.

3.2 소프트웨어 구조

이 연구에서는 각각의 프로그램 모듈이 Fig. 4와 같은 계층 구조를 이루게 된다. 아래층의 구조는 위층의 구조에서 필요로 하는 기능을 제공하게 되며, 이러한 층 단위 구조를 채용함으로써 구성 요소의 모듈화가 이루어져 응용프로그램 작성의 유연성이 증가하고 소프트웨어의 모듈별 유지, 보수, 성능 향상이 쉬워지며 재사용성을 증가시켰다.

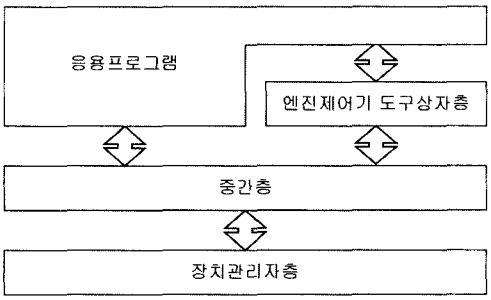

Fig. 4 Layered software architecture

3.2.1 장치관리자층

제어 프로그램을 실시간 하드웨어 상에서 수행하기 위해서는 각종 입/출력, A/D변환, 타이머 인터럽트 발생 등의 기능이 필요하며, 이를 위하여 하드웨어와 관련된 작업을 수행하는 장치관리자를 기능별로 구현하였다.

3.2.2 중간층

일반적으로 장치관리자가 하드웨어를 제어하기 위하여 사용하는 매개변수(parameter)는 그 수가 많을 뿐 아니라 의미도 하드웨어에 의존적인 경우가 많으므로 일반 사용자가 쉽게 사용하기가 어렵다. 제어에 필요한 데이터의 연산 및 보정, 인코더 신호의 처리, 연료 분사 및 점화, A/D변환 등의 작업을 보다 쉽게 하기 위하여 다양한 함수들을 설계하고 이를 모듈화하여 중간층을 구성하였다. 이러한 중간층의 함수는 응용프로그램에서 직접 호출하여 사용하거나 엔진제어기 도구상자를 통하여 간접적으로 호출하여 사용한다. 이 중간층을 이용하여 하드웨어에 의존적인 장치관리자층을 응용프로그램과 구분하여 계층별 프로그램 모듈의 관리 및 성능 향상을 이룰 수가 있다.

3.2.3 엔진제어기 도구상자층

래피드 콘트롤 프로토타이핑 기법을 적용하여 자동화된 방식으로 제어 프로그램을 생성하기 위해서, 장치관리자는 제어기 모델과 도식적으로 연결되어야 한다. 따라서 Simulink를 이용하여 제어기 모델을 설계하는 과정에서 사용자가 하드웨어와 관련된 기능을 모델에 포함할 수 있도록 장치관리자와 중간층의 함수를 사용하여 엔진 제어시스템을 위한 Simulink 도구상자를 구현하였다. 도구

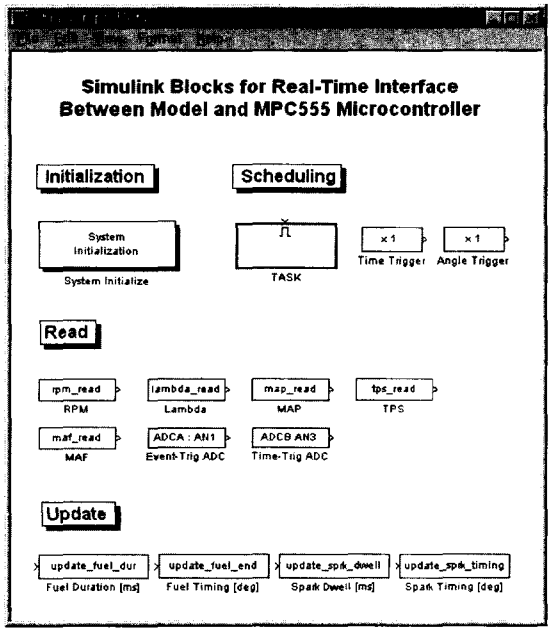

Fig. 5 Engine control toolbox

상자를 통하여 사용자는 보통의 Simulink 블록을 사용하는 방법으로 제어기 모델에 직접 하드웨어와 관련된 기능을 포함할 수 있게 된다. 이렇게 제어기 모델에 연결된 도구상자 블록은 제어 프로그램을 자동으로 생성하는 과정에서 실제 하드웨어 작업을 처리하는 프로그램 코드로 바뀌게 된다. 또한, 이 연구에서는 제어기 모델을 태스크 단위로 구성하여 각각의 태스크들을 정해진 시간이나 이벤트에 의하여 수행할 수 있도록 기본적인 태스크 스케쥴링을 위한 도구상자 블록을 추가적으로 구현하였다.

엔진제어기 도구상자 블록은 s-function을 사용하여 구성하였으며 마스크(mask) 기능을 이용하여 하드웨어 제어를 위한 각종 매개변수를 편리하게 설정할 수 있도록 하였다.[9]

Fig. 5는 엔진 제어를 위하여 구현한 도구상자를 보여주고 있다.

3.3 응용프로그램과 자동화 과정

자동 코드 생성 기법은 래피드 프로토타이핑의 가장 중요한 특징이다. 이를 위해서는 제어 알고리즘을 프로그래밍 하는 과정을 자동화된 절차로 구현하여 프로그램의 코딩, 컴파일, 링크 및 타겟 하드웨어에 내장하는 일련의 과정을 자동화하는 것이 필요하다.

Fig. 6은 Real-Time Workshop을 이용하여 응용프로그램의 실행 파일을 생성하는 과정을 나타내고 있으며, 또한 각 단계에서 필요한 파일과 생성되는 파일들을 보여주고 있다.

Real-Time Workshop의 코드 생성기를 이용하여 제어기 모델의 실행 파일을 생성하기 위해서는 프로그램 생성 과정을 사용자의 개발 환경에 맞도록 변경하는 과정이 필요하다.[9,10] 필요한 변경 작업 및 추가해야 할 모듈은 크게 다음과 같이 나눌 수 있다.

1) 컴파일러/링커 관련 설정 : 프로그램의 자동 생성 과정은 마이크로컨트롤러를 비롯한 제어시스템 개발 환경에 의존적이므로 제어 프로그램을 탑재할 하드웨어에 적합하도록 프로그램 생성 과정을 제어하여야 한다. 이를 위하여 template makefile, target language compiler file, make command 등을 사용하고자 하는 개발 환경에 적합하도록 구성하여 코드 생성 과정을 제어할 수 있다.

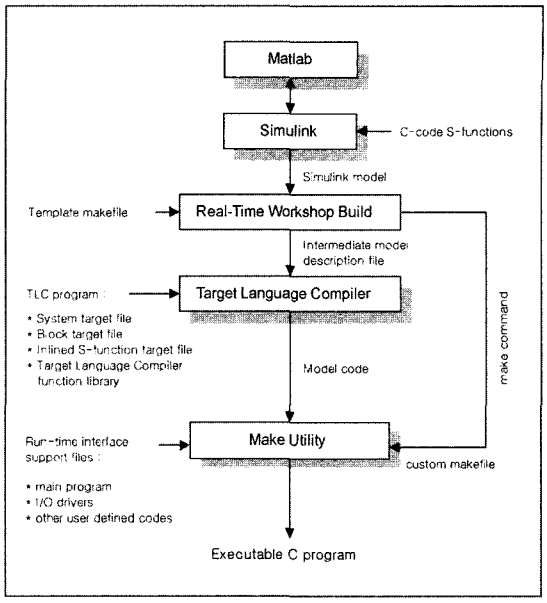

Fig. 6 Code generation procedure

2) 주 프로그램 모듈(main program) : 제어기 모델을 이용하여 만들어진 프로그램 코드는 사용자가 정의한 주 프로그램 모듈에 의하여 이벤트 기반 또는 시간 기반의 제어를 수행할 수 있으며 기타 부가적인 기능의 추가 또한 가능하다.

4. 엔진 공연비 제어 실험

이 연구에서 구현한 래피드 콘트롤 프로토타이핑 플랫폼 이용 예로서 엔진의 공연비 제어기를 구성하여 시험을 수행하였다.

4.1 제어기 모델

이 연구에서는 공연비 제어를 위하여 이산 비례적분(PI) 제어기를 구성하였으며 Fig. 7은 하나의 태스크로 구현한 제어기와 시스템 초기화 블록 및 하드웨어 제어를 위한 장치관리자 도구상자 블록들이 연결된 상태를 보여주고 있다.

Fig. 8은 Fig. 7에서 부시스템으로 표현된 PI 제어기의 내부를 보여주고 있다.

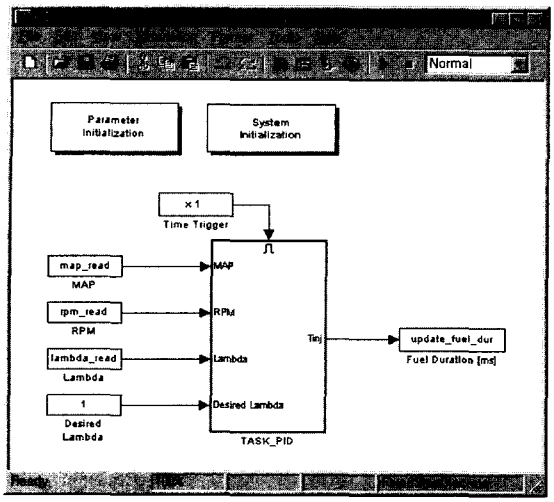

Fig. 7 PI controller

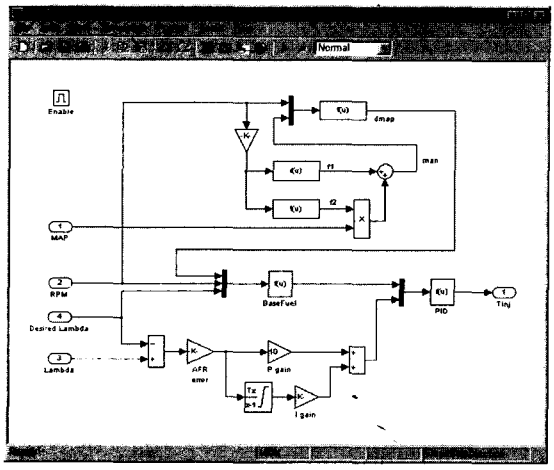

Fig. 8 PI controller (subsystem model)

4.2 폐루프 실험

제어 시스템의 개발 및 성능 평가를 위하여 엔진 모델[11]을 사용하였으며, 오프라인 시뮬레이션을 수행한 후 제안된 래피드 프로토 타이핑 플랫폼을 이용하여 폐루프 실험을 수행하였다.

오프라인 시뮬레이션의 경우에는 정밀도를 향상하기 위하여 제어기 모델은 5 *m*sec 과 엔진 모델은 500 μsec의 수행 주기를 갖는 복수개의 시간 간격(multi-rate)으로 해석하는 기법을 사용하였다.

래피트 콘트롤 프로토타이핑 플랫폼을 이용한 실험의 경우에는 Real-Time Workshop의 프로그램 생성 기능과 앞서 구현한 각종 하드웨어/소프트웨어 모듈을 이용하여 제어기 모델의 실행 프로그램을 생성하고 실행 파일을 MPC555 마이크로컨트롤러에 다운로드하고 수행시켰다. 또한 최근 실제의 제어 대상을 대신하여 가상적인 실험 환경을 제공하는 HILS(Hardware-in-the-loop simulation) 환경과 오프라인 시뮬레이션에서 사용한 엔진 모델을 이용하여 폐루프 실험을 수행하였다.[12]

Fig. 9는 오프라인(off-line) 시뮬레이션과 래피트 콘트롤 프로토타이핑 플랫폼을 이용한 실험 결과이다. Fig. 9의 (a)에서 나타낸 바와 같이 주기적으로 쓰로틀을 변화 시켰으며, 그에 따른 공연비의 변화량은 (c)와 (d)에 각각 나타내었다.

오프라인 시뮬레이션에 비하여 래피트 콘트롤 프로토타이핑의 경우에 공연비 제어 성능이 조금 떨어진 것을 확인할 수 있다. 이는 A/D 변환기의 성능,

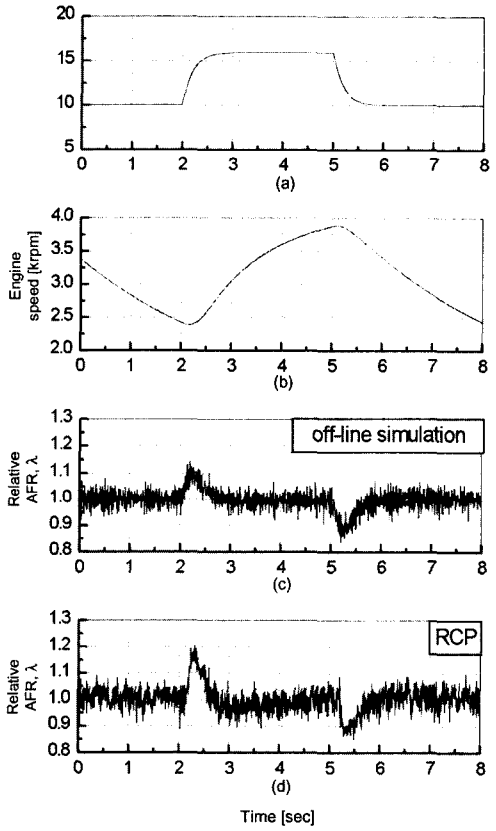

Fig. 9 Comparison of off-line simulation and RCP experiment

제어기의 수행시간, 외부 노이즈와 같은 실제 구현에 있어서 발생할 수 있는 여러 요인들 때문이다. 이와 같이 래피드 콘트롤 프로토타이핑 플랫폼을 사용하여 실제 제어 실험에서 발생할 수 있는 문제들을 효율적으로 확인하고 대처할 수 있다.

5. 결 론

복잡한 엔진 제어 알고리즘을 빠르게 엔진 제어기에 구현하여 검증하기 위한 래피드 콘트롤 프로토타이핑 플랫폼을 개발하였다.

이를 통하여 제어 프로그램 작성의 편의 증대 및 시스템 개발 기간을 단축할 수 있으며 특히, 계층 소프트웨어 구조를 채택하여 프로그램의 유지, 보수, 성능 향상을 용이하게 하였다. 또한 기본적인 태스크 스케쥴링 기능을 구현함으로써 제어기 모델을 태스크 단위로 분리하여 시간 기반 또는 이벤트 기반으로 수행할 수 있도록 하였다.

References

1) W.-S. Gan, Y.-K. Chong, W. Gong, W.-T. Tan, "Rapid Prototyping System for Teaching Real-Time Digital Signal Processing," IEEE Transactions on Education, Vol.43, pp.19-24, 2000.
2) A. W. Stylo, G. Diana, "An Advanced Real-Time Research and Teaching Tool for the Design and Analysis of Control," Afrion, 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) S. Rebeschiess, "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.267-272, 1999.
5) MPC555 User's Manual, Motorola Inc.
6) Using Simulink Ver.4, MathWorks Inc.
7) Real-Time Workshop User's Guide Ver.4, MathWorks Inc.
8) D-CC & D-C++ Compiler Suites User's Guide ver.4.3 : Diab Data Inc.
9) Writing S-Functions Ver.4, MathWorks Inc.
10) Target Language Compiler Reference Guide Ver.4, MathWorks Inc.
11) P. Yoon, M. Sunwoo, "A Nonlinear Dynamic Modeling of SI Engines for Controller Design," International Journal of Vehicle Design, Vol.26, No.2/3, pp.277-297, 2001.
12) J. Song, "Development of a Rapid Controller Prototyping System for Embedded Control Systems," Hanyang University, Master Degree Thesis, 2001.