논문 2014-09-12

AUTOSAR 소프트웨어 설계를 위한 실습 환경

(AUTOSAR Starter Kit for AUTOSAR Software Design)

이 성 훈, 김 영 재, 금 대 현, 진 성 호*
(Seonghun Lee, Youngjae Kim, Daehyun Kum, Sungho Jin)

Abstract : An AUTomotive Open System ARchitecture (AUTOSAR) is a de-facto standardized software platform, which developed for an automotive Electronic Control Unit (ECU) in global automotive industry. AUTOSAR improves the reusability and the scalability, thus the software development can be easier, faster and more reliable. However, it requires a lot of time and efforts to develop an AUTOSAR software due to the difficulties of understanding of massive AUTOSAR documentations and complicated usage of AUTOSAR design tools. AUTOSAR training is offered by AUTOSAR design tool venders but it is limited to introduction of their simplified concept and usages based on PC. Therefore the training is not enough for industrial developers or graduate students. In this paper we present an AUTOSAR starter kit which allows industrial engineers and graduate students to practice the detailed process of AUTOSAR software development easily and more conveniently. The kit is composed of a practical environment similar to actual automotive system and a textbook that explains how to design AUTOSAR software. And we demonstrated the validity of our methodology based on a case study.

Keywords : AUTOSAR, Embedded software, Simulator, Kit

Ⅰ. 서 론

AUTOSAR는 차량 ECU에 사용되는 임베디드 소프트웨어 플랫폼의 국제 표준 규격 및 이를 제정하는 단체를 의미한다[1]. ECU 소프트웨어를 개발 시에 표준화된 인터페이스, 운영 체제, 소프트웨어 모듈을 사용함으로써 소프트웨어의 재사용성 및 확장성 등을 향상시켜, 복잡한 소프트웨어를 빠르고 신뢰성 있게 개발 할 수 있다[2-4].

2003년 AUTOSAR 단체가 설립된 이후로 3단계를 걸쳐 규격 R4.1까지 제정되었으며, 현재까지 지속적으로 규격을 보완 및 개정되고 있다. 유럽, 미국, 일본 등의 주요 AUTOSAR 회원사들은 이미 초기 버전의 AUTOSAR가 내장된 ECU를 일부 차종에 대해 출시하고 있으며, 수 년 이내에 최신 버전의 AUTOSAR를 ECU에 적용할 계획을 갖고 있다. 한편 한국은 현대·기아자동차그룹을 중심으로 AUTOSAR 관련 기술을 지속적으로 선행 개발중이며, 조만간 단계적으로 일부 차량에 적용할 계획을 갖고 있다. 또한 일부 국내 대학 및 연구기관에서는 AUTOSAR 관련 도구, 응용 방법 등에 관해 다양한 연구를 수행해오고 있다[1, 5-8].

하지만 자동차분야에 특화된 AUTOSAR를 활용한 ECU를 개발하기 위해서는 방대한 분량의 AUTOSAR 규격 문서를 이해하고, 다양한 개발 툴을 사용하여 제품을 개발해야 하므로 많은 시간과 노력이 필요하다. 특히 국내 완성차 및 부품 기업의 개발자들에 의해 수많은 차량 ECU들이 독자적으로 개발되고 있지만, 신기술에 대한 내부 교육 기회가 부족하고, 외부 교육의 경우 기본적인 공학 이론 교육에 국한되는 현실이다. AUTOSAR 교육의 경우, 개발 도구 업체의 사용법 위주의 교육으로 진행되는 경우가 대부분이며, 현장에 즉시 필요한 실무 기술에 대한 교육은 전무한 상황이다[9, 10]. 따라서

*Corresponding Author (sungho@dgist.ac.kr)
Received: 4 Dec. 2013, Revised: 6 Jan. 2014. Accepted: 21 Jan. 2014.
S.H. Lee, Y.J. Kim, D,H. Kum, S.H. Jin: DGIST
※ 본 논문은 미래창조과학부에서 지원하는 대구경북과학기술원 기관고유사업(13-RS-03)에 의해 수행되었음.

ⓒ J. IEMEK 2014 Apr.: 9(2) 87-99
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2014.9.2.87

국내 자동차관련 개발자들 대부분이 개발기간 단축을 위해 노력하는 점을 고려할 때, 보다 손쉽고, 빠르게 AUTOSAR 소프트웨어를 개발할 수 있는 환경이 필요한 실정이다[11, 12].

본 논문에서는 AUTOSAR 소프트웨어를 개발할 수 있는 Hardware In the Loop (HIL)기반 실습 환경인 AUTOSAR 스타터 키트를 제안한다. 현재 자동차의 바디 제어에 사용되고 하드웨어 환경과 AUTOSAR 설계 도구를 활용한 실습 예제까지 소개한다. 이 키트는 AUTOSAR의 기본 개념, 규격 정리, 그리고 차량 바디 분야 응용시스템 개발 과정까지 포함한다. 기존 AUTOSAR 관련교육의 문제점인 이론 설명이나 단순한 개발 도구의 사용법만을 다루지 않고, 실제 개발자가 차량 응용시스템을 개발하는 설계, 구현, 검증까지 전 과정을 포함하고, 실제 운용환경에 적합한 하드웨어 성능을 가지고, 대표적인 차량 바디분야 부하 입출력을 포함하는 임베디드 시스템 개발 실습 환경을 제안한다.

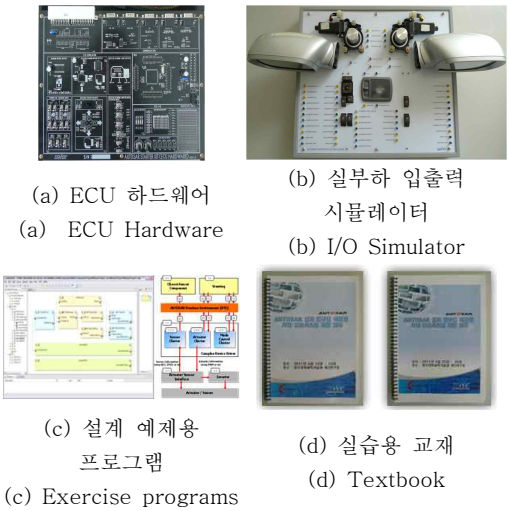
본 논문은 2장에서 AUTOSAR 스타터 키트의 구성 및 주요 설계 사양에 대해 상세히 설명하고, 3장에서 개발하려는 시스템에 따라 AUTOSAR 스타터 키트를 활용하는 방안에 대해 소개한다. 4장에서는 실습 환경을 이용한 AUTOSAR 소프트웨어 설계 사례를 통하여, 단계별 AUTOSAR 소프트웨어 개발 과정을 상세하게 제안하고, 5장에서 결론을 맺는다.

II. AUTOSAR 스타터 키트 구성 및 설계

통상적으로 임베디드 소프트웨어를 개발시 마이크로컨트롤러 제조사에서 제공하는 개발용 보드를 가장 먼저 사용하게 된다. 하지만 이 개발용 보드를 이용해서는 마이크로컨트롤러의 기본 환경 설정, TTL 로직 레벨의 입출력 신호 등 제한된 기능만 확인이 가능하다. 따라서 실제 차량 환경과 유사한 입출력 신호를 이용하여 다양한 제어 알고리즘을 개발하기 위해서는 실제 차량의 임베디드 시스템과 유사한 개발 환경이 필요하다.

제안하는 AUTOSAR 스타터 키트는 그림 1에서와 같이 차량 도어 모듈에 대한 제어로직을 담당하는 ECU 하드웨어와, 차량 도어 모듈의 신호 입출력을 담당하는 실부하 입출력 시뮬레이터, 실습용 예제 프로그램 및 교재로 구성된다.

ECU 하드웨어 간에는 대표적인 차량 직렬 통신인 Controller Area Network (CAN), Local Interconnect Network (LIN), FlexRay 통신으로 연결이 가능하고, 개발용 PC와 ECU 하드웨어는 Background Debug Module (BDM) 단자를 통해 프로그램을 다운로드 및 디버깅 할 수 있다. 그리고 ECU 하드웨어는 실부하 입출력 시뮬레이터와 연결되어 실제 차량 부하 신호와 동일한 상황에서 입출력 제어를 할 수 있다. 또한 몇 개의 응용 시스템에 대한 예제 프로그램과 교재를 이용하여 단계적 절차에 따라 이 환경을 용이하게 활용할 수 있다.



(a) ECU 하드웨어
(a) ECU Hardware

(b) 실부하 입출력 시뮬레이터
(b) I/O Simulator

(c) 설계 예제용 프로그램
(c) Exercise programs

(d) 실습용 교재
(d) Textbook

그림 1. AUTOSAR 스타터 키트의 구성
Fig. 1 Composition of AUTOSAR Starter Kit

1. ECU 하드웨어

차량용 ECU는 가혹한 환경에서도 정상적으로 동작되기 위해 전용 부품을 사용하여 설계되어야 한다. 또한 다양한 차량 입출력 부하 신호 전기적인 특성을 고려하여 설계되어야 하고, 차량 전용 네트워크 인터페이스도 설계되어야한다. 본 논문에서는 국내에서 가장 많이 독자 개발하고 있는 바디분야에 특화된 ECU 하드웨어를 제안한다.

일반적인 교육 장비의 하드웨어와 달리 차량의 윈도우 제어, 사이드미러 제어, 실내등 제어 용도에 적합하도록 설계되어야 한다. 따라서 마이크로컨트롤러, 모터 드라이버, 센서 등의 반도체 부품은 자동차 환경 규격을 만족하고 실제 사용되는 부품을 적용하여, 교육 개발 환경과 실제 개발 환경을 최대한 유사하도록 설계 하였다. 또한 교육용 특성에 맞도록 보관과 이동성을 고려하여 중량 및 부피를 최소화 하여 설계 하였다. 그림 2와 표 1에 ECU 하

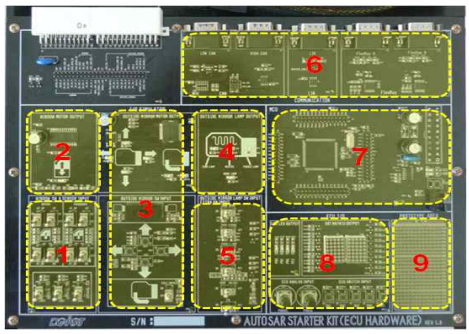


그림 2. ECU 하드웨어
Fig. 2 ECU Hardware

표 1. ECU 하드웨어 구성
Table 1. ECU hardware components

영역	이름	설명
1	윈도우 스위치 입력부	· 운전석 및 조수석의 윈도우 스위치 입력 · 윈도우 모터의 센서 입력 · 토글 스위치
2	윈도우 모터 출력부	· 윈도우 모터 출력 · 윈도우 모터 구동 전용 드라이버 IC · 모터 출력 방향 표시용 발광 다이오드
3	사이드미러 스위치 입력 및 모터 출력부	· 사이드미러 위치조절 스위치 입력 · 사이드미러 선택 스위치 입력 · 사이드미러 접힘 스위치 입력 · 버튼 및 토글 스위치 · 사이드미러 모터 출력 · 사이드미러 구동 전용 드라이버 IC · 모터 출력 방향 표시용 발광 다이오드
4	사이드미러 램프 출력부	· 방향등, 안내등 출력 · 실내등, 미러 히터 출력 · 발광 다이오드 출력
5	사이드미러 램프 스위치 입력부	· 방향등, 안내등 스위치 입력 · 도어 열림 상태 스위치 입력 · 토글 스위치로 구성
6	CAN, LIN, FlexRay 통신부	· High/Low speed CAN 통신부 · LIN 통신부 · FlexRay A, B 통신부
7	마이크로컨트롤러 회로부	· 프리스케일 MC9S12XF512 · 전용 레귤레이터
8	범용 입출력 회로부	· 발광 다이오드 출력 · 도트 매트릭스 디스플레이 출력 · 아날로그 입력, 디지털 입력
9	프로토타입부	· 프로토타입 영역 · 테스트 회로 추가시 사용

드웨어에 대해 회로 구성과 기능들을 상세하게 설명하였다.

마이크로컨트롤러는 대표적인 차량 반도체 사인 프리스케일사의 16bit 프로세서 MC9S12XF512를 사용하였고, 하드웨어 와치독이 내장된 전원 레귤레이터를 적용하였다[13]. 모터 및 램프를 구동하기 위해 Serial Peripheral Interface (SPI) 통신 등을 지원하는 전용 드라이버 IC를 각각 사용하였고, 차량의 대표적인 직렬통신인 CAN, LIN, FlexRay 통신 네트워크를 지원 가능하도록 설계에 반영하였다. CAN 통신은 최대 1Mbps 속도로 통신이 가능한 고속 통신 채널과 125Kbps 속도로 통신이 가능한 저속 통신 채널로 각각 구성하여 통신 속도별 설계 실습이 가능하도록 구성하였다. 또한 저렴한 비용으로 구현이 가능하여 CAN 통신의 하위 버스로 주로 사용되는 LIN 통신은 최대 19.2Kbps 속도로 통신이 가능한 채널로 구성하였다. 그리고 최대 10Mbps 속도까지 통신이 가능한 FlexRay는 이중화 설계를 고려하여 A, B 2개의 채널로 구성하였다.

실부하 입출력 시뮬레이터와 연결을 위해 차량 전용 커넥터를 사용하였고, 바디분야 전용 기능 목적이외에 기본적인 기능 테스트를 할 수 있도록 발광 다이오드, 도트 매트릭스 디스플레이를 활용할 수 있는 영역과 사용자가 간단한 회로를 구현할 수 있는 영역도 포함시켜 개발시 활용성을 높힐 수 있도록 했다.

2. 실부하 입출력 신호 시뮬레이터

차량용 임베디드 소프트웨어 개발시 실제와 유사한 환경을 제공하기 위해 실제 차량 부품을 이용하여 차량 신호의 입출력 제어를 해야 한다. 본 논문에서는 바디분야에 특화된 윈도우 제어, 사이드미러 제어, 실내등 제어 용도에 적합한 실부하 입출력 신호 시뮬레이터를 설계하였다. 다양한 입출력 신호를 획득하고 제어하기 위해 고급 편의 기능이 내장된 대형차량의 실제 부품의 전기적인 신호를 분석한 후, ECU 개발시 사용하기 편리하도록 배치 설계하였다. 그림 3과 표 2에 실부하 입출력 신호 시뮬레이터를 구성 하는 주요 부품과 그 기능들을 상세하게 설명하였다.

실부하 입출력 신호 시뮬레이터는 운전석 도어, 조수석 도어의 대표적인 부하인 윈도우와 아웃사이드 미러의 실부하를 탑재하였으며, 상단의 커넥터를 통해 운전석 및 조수석 ECU 하드웨어를 연결하여 윈도우와 아웃사이드 미러의 다양한 연동기능을 구현 및 검증할 수 있다. 우선 ECU 하드웨어에서 개발 시스템의 각 기능들을 개별적으로 구현 및 검증한 다음, 전용 케이블로 입출력 신호 시뮬레이터와 연결하여, 실제 부하의 입출력 신호의 제어를 확인하면서 검증 할 수 있게 된다. 그리고 ECU 하드웨어간 신호를 교환하는 윈도우 및 아웃사이드 미러의 연동 기능들은 CAN, LIN, FlexRay 등 차량 네트워크 중 하나를 선정하여 설계후 별도로 케이블로 연결하여 구현 할 수 있다.

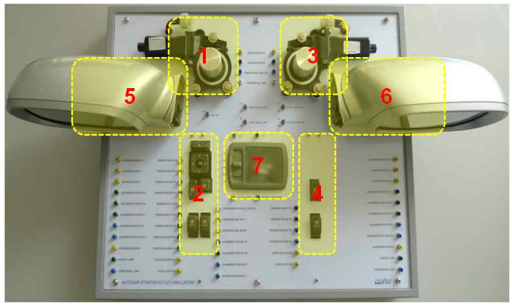


그림 3. 실부하 입출력 신호 시뮬레이터
Fig. 3 Real-load I/O Simulator

표 2. 실부하 입출력 신호 시뮬레이터 구성
Table 2. Real-load I/O Simulator components

영역	이름	설명
1	윈도우 모터 (운전석)	· 윈도우 레귤레이터용 DC 모터(12V, 8A) · 홀센서로 모터 회전 판별(2pulse/rev) · 신호 점검용 단자
2	윈도우 및 사이드미러 스위치 (운전석)	· 운전석 및 조수석의 윈도우 스위치 입력 · 사이드미러 선택 스위치 입력 · 사이드미러 위치조절 스위치 입력 · 사이드미러 접힘 스위치 입력 · 신호 점검용 단자
3	윈도우 모터 (조수석)	· 윈도우 레귤레이터용 DC 모터(12V, 8A) · 홀센서로 모터 회전 판별(2pulse/rev) · 신호 점검용 단자
4	윈도우 스위치 (조수석)	· 조수석의 윈도우 스위치 입력 · 신호 점검용 단자
5	사이드 미러 (운전석)	· 사이드미러 위치조절 DC 모터(12V, 1A) · 사이드미러 위치 potentiometer 센서(5V) · 사이드미러 접힘용 DC 모터(12V, 1A급) · 사이드미러 열선용 히터저항(12V) · 사이드미러 내장형 방향등(12V) · 사이드미러 내장형 안내등(12V) · 신호 점검용 단자
6	사이드 미러 (조수석)	· 사이드미러 위치조절 DC 모터(12V, 1A) · 사이드미러 위치 potentiometer 센서(5V) · 사이드미러 접힘용 DC 모터(12V, 1A) · 사이드미러 열선용 히터저항(12V) · 사이드미러 내장형 방향등(12V) · 사이드미러 내장형 안내등(12V) · 신호 점검용 단자
7	실내등	· 실내등 램프 · 신호 점검용 단자

그리고 각각 윈도우 모터, 사이드 미러 등의 부품별 신호들에 대한 점검용 단자를 이용하여, 구현 및 검증시에 전기적인 신호를 손쉽게 확인 가능하도록 설계에 반영하였다.

3. 실습용 예제 프로그램 및 교재

AUTOSAR 스타터 키트를 손쉽게 활용하기 위해서는 ECU 하드웨어 설정 방법, AUTOSAR 설계 도구 사용법뿐만 아니라 응용 시스템에 대한 소프트웨어 개발과정에 대한 상세한 정보가 필요하다. ECU 하드웨어 중 마이크로프로세서에 대한 내용은 프리스케일사 MC9S12XF512에 대한 관련 문서나 프로그램을 참고하여 설정할 수 있다[13]. 그리고 AUTOSAR 설계 도구 사용법 역시 판매 회사의 도구 교육과정이나, 관련 문서를 통해 정보를 습득할 수 있다[2, 3]. 하지만 이들을 이용하여 차량 응용 시스템을 개발하는 과정은 주어진 환경에 따라 다르므로 쉽게 정보를 찾을 수 없다.

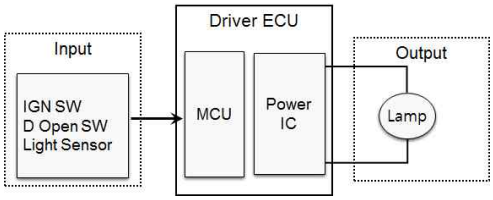


그림 4. 룸램프 제어 시스템 블록도
Fig. 4 Block diagram of Roomlamp Control System

실습용 예제 프로그램 및 교재는 본 ECU 하드웨어와 실부하 입출력 시뮬레이터 환경에서 벡터사의 AUTOSAR 설계 도구를 활용하여 차량 응용시스템을 개발하는 절차를 기술하였다. 차량 응용시스템에 대해 요구사항 분석, 시스템 설계, ECU 설계, 컴파일 및 디버깅, 기능 테스트 까지 AUTOSAR 설계 도구를 활용하여 단계적으로 개발하는 과정을 제시하였다. 대표적인 시스템으로 룸램프 제어시스템, 윈도우 제어 시스템, 사이드 미러 제어 시스템에 대해 예제 프로그램 소스 코드와 개발 과정이 교재에 포함 되어 있다.

3.1 룸램프 제어 시스템

룸램프 제어 시스템은 단일 ECU로 구성된 시스템으로 스위치 입력에 따른 룸램프의 단순 ON/OFF 제어부터 차량 상태에 따른 룸램프 감광 제어 및 외부밝기에 따른 룸램프 자동제어 기능을 수행하는 시스템이다. 그림 4는 룸램프 제어 시스템의 입출력 신호에 대한 블록도를 나타낸다. 이 과정에서는 기본적인 AUTOSAR 개념 및 구조에 대해 소개한다. 시스템 설계, ECU 설계, ECU 통합, 그리고 테스트에 걸친 전반 과정을 간략히 소개한다. 특히 가장 기본적인 AUTOSAR SoftWare Component (SWC), Run-Time Environment (RTE), Operating

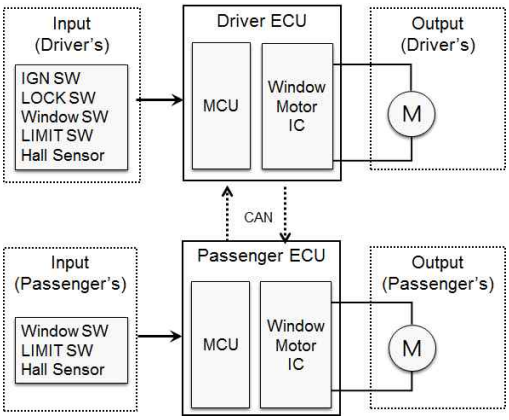


그림 5. 윈도우 제어 시스템 블록도
Fig.5 Block diagram of Window Control System

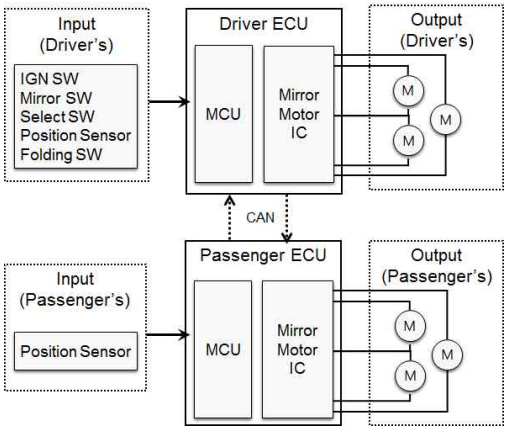


그림 6. 사이드 미러 제어 시스템 블록도
Fig. 6 Block diagram of Side mirror Control System

Systems (OS), Basic SoftWare (BSW) 등에 설정 방법에 대해 실습해 볼 수 있는 내용을 포함하고 있다.

3.2 윈도우 제어 시스템

윈도우 제어 시스템은 운전석 도어와 조수석 도어에 있는 윈도우의 개폐를 제어하는 시스템으로 2개의 ECU 하드웨어가 CAN 통신으로 연결되어 기능을 수행한다. 운전석 및 조수석에 있는 스위치의 신호 입력 상태에 따라 각각의 윈도우 모터가 동작하게 된다. 그림 5는 윈도우 제어 시스템의 입출력 신호에 대한 블록도를 나타낸다. 이 과정에서는 기본적인 AUTOSAR 개념이외에, CAN 통신에 대한 부분을 포함하므로 통신 서비스 모듈, 및 통신 하드웨어 추상화 모듈, 통신 드라이버 모듈에 대해 추가로 배우게 된다.

3.3 사이드 미러 제어 시스템

사이드 미러 제어 시스템은 운전석 도어와 조수석 도어에 있는 사이드 미러를 제어하는 시스템으로 2개의 ECU 하드웨어가 CAN 통신으로 연결되어 기능을 수행한다. 미러 조절 스위치에 의한 각 미러의 단순한 위치 제어 기능, 접힘/펼침 기능 이외에 현재의 위치를 저장하는 메모리 기능을 포함한다. 그림 6은 사이드 미러 제어 시스템의 입출력 신호에 대한 블록도를 나타낸다. 이 과정에서는 기본적인 AUTOSAR 개념 및 CAN 통신 부분 이외에 메모리 부분을 추가로 포함하므로, 메모리 서비스 모듈, 메모리 하드웨어 추상화 모듈, 메모리 드라이버 모듈에 대해 추가로 배우게 된다.

Ⅲ. AUTOSAR 스타터 키트 활용 방안

사용자는 개발 하려는 시스템의 기능 및 요구사양에 따라 AUTOSAR 스타터 키트의 구성과 소프트웨어 개발 절차를 다르게 적용해야 한다. 본 장에서는 AUTOSAR 스타터 키트의 구성 방법과 일반적인 소프트웨어 개발 절차에 대해 제안한다.

1. 실습 환경 구성

차량 내에서 임베디드 시스템은 목적 및 기능에 따라 단독으로 혹은 네트워크에 연결되어 기능을 수행한다. 본 AUTOSAR 스타터 키트는 CAN, LIN, FlexRay 네트워크를 이용하여, ECU 하드웨어간의 연결이 가능하고, 또한 실부하 입출력 시뮬레이터를 ECU 하드웨어에 연결하여 실제 부하를 이용한 입출력 제어가 가능하다.

1.1 독립형 시스템 개발의 경우

일반적으로 기능 사양이 간단한 임베디드 시스템은 차량 네트워크에 연결 없이 독립적으로 기능이 제어가 가능하다. 제안한 실습 환경을 이용하여 독립형 차량 임베디드 시스템을 개발할 경우, 크게 그림 7의 개념과 같은 순서로 개발할 수 있다.

첫째, PC 단독 환경에서 AUTOSAR 설계 도구를 활용하여 시스템 설계를 수행한다. 시스템 설계에는 런어블, 태스크, 로직 등 여러 사항에 대해 설계를 포함한다. 그리고 가상 ECU를 이

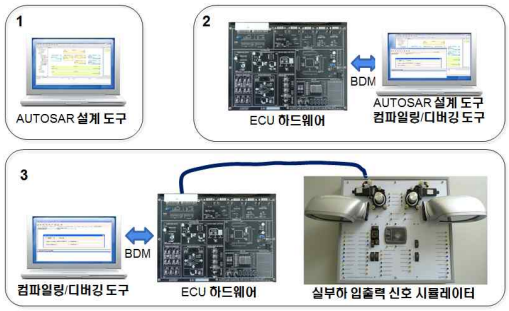


그림 7. 독립형 시스템 구성
Fig. 7 Configuration of single ECU system

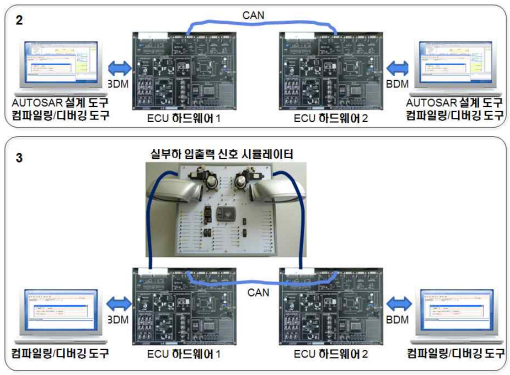


그림 8. 네트워크형 시스템 구성
Fig. 8 Configuration of network ECU systems

용한 ECU 설계까지 마쳐, 요구 사항에 대한 소프트웨어의 기능 오류를 검증한다. 기능 검증이 마친 응용 소프트웨어는 코드 생성기를 통해 컴파일이 가능한 c 코드를 출력 한다.

둘째, PC와 ECU 하드웨어를 BDM으로 연결 후, 컴파일 된 코드를 다운로딩, 디버깅, 테스팅을 실행한다. 이를 통해 ECU 하드웨어에 장착된 간이 입출력 신호 스위치 및 램프를 이용하여 시스템의 기능이 정상 동작하는 지를 검증한다. 상대적으로 낮은 전압, 전류에서 입출력 신호를 이용하여 시스템 기능을 테스트하므로 개발 초기에 쉽게 나타날 수 있는 과전류에 의한 ECU 회로 손상을 방지할 수 있다.

셋째, ECU 하드웨어와 실부하 입출력 시뮬레이터를 연결 후, 실제 차량 부하에서 생성되는 입출력 신호를 ECU 하드웨어에서 제어하여 기능의 정상 동작여부를 검증한다. 이 단계를 통해 실제 차량과 유사한 전기·기계적 환경에서 설계한 응용소프트웨어가 정상적으로 동작하는 지 여부를 판단할 수 있다. 실제 차량에서 발생되는 전압, 전류 신호를 이용하므로, 간이 입출력 신호를 이용하여 테스트 하는 경우보다 정확한 테스트를 수행할 수 있다.

1.2 네트워크형 시스템 개발의 경우

분산 배치된 센서나 액추에이터를 동시에 제어하거나 다른 ECU의 정보를 사용하기 위해서는 다양한 데이터를 네트워크를 통해 교환해야 한다. AUTOSAR 스타터 키트의 각 ECU 하드웨어는 2개의 CAN, 1개의 LIN, 2개의 FlexRay 네트워크를 포함하고 있다. 제안한 실습 환경을 이용하여 네트워크형 차량 임베디드 시스템을 개발할 경우, 크게 그림 8의 개념과 같은 순서로 개발 할 수 있다.

첫째, PC 단독 환경에서 AUTOSAR 설계 도구를 이용하여 시스템 설계를 수행한다. 독립형 시스템 개발과 달리 네트워크 설계과정을 추가로 필요하며, 이를 통해 네트워크 메시지, 신호 등에 대한 세부사항을 설계후 ECU 맵핑 과정을 각 ECU에 들어간 소프트웨어 컴포넌트 설정 등을 거쳐 시스템 설계를 마무리한다. 그리고 ECU 설계, 코드 생성을 거쳐 가상 ECU를 통해 요구 사항에 대한 기능 검증을 수행한다.

둘째, 두 ECU 하드웨어 간에 CAN/LIN/FlexRay 네트워크 케이블을 연결한다. 그리고, PC와 ECU 하드웨어를 BDM으로 각각 연결 후, 각 ECU에 해당하는 소프트웨어를 컴파일링, 다운로딩, 디버깅, 테스팅을 실시한다. ECU 하드웨어에 장착된 간이 저전력 입출력 신호 스위치 및 램프 등을 통해 두 ECU 하드웨어간의 네트워크가 정상적으로 연결되어 있고, 로직의 정상 동작하는 지 여부를 검증할 수 있다.

셋째, 2개의 ECU 하드웨어를 실부하 입출력 신호 시뮬레이터를 연결 후, 실제 차량 부하에서 생성되는 입출력 신호를 ECU 하드웨어에서 제어하여 기능의 정상 동작여부를 검증한다. 이 단계를 통해 실제 차량과 유사한 전기·기계적 환경에서 설계한 응용소프트웨어가 정상적으로 동작하는 지 여부를 판단할 수 있다. 실제 차량에서 발생되는 입출력 신호를 이용하므로, 스위치, 모터 등의 변화에 따른 센서 신호까지 이용하므로 보다 정확한 테스트를 수행할 수 있다.

2. 소프트웨어 개발 절차

AUTOSAR 소프트웨어 설계하기 위해서는 단계적인

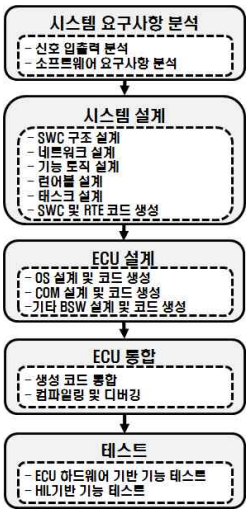


그림 9. AUTOSAR 소프트웨어 개발 절차
Fig. 9 Development process of AUTOSAR software

소프트웨어 개발 절차 과정이 필요하다. AUTOSAR 에서는 V-model 개발 절차와 유사한 개발 방법론을 제시하고 있으며, 이를 실습 환경을 이용한 개발 단계에 맞춰 그림 9와 같이 제안한다.

첫째, 가장 먼저 필요한 단계는 시스템 요구사항 분석이다. 개발하려는 시스템의 기능적, 비기능적 요구사항 분석을 통해 시스템의 하드웨어, 소프트웨어 개발 사항을 도출하고, 전체적인 시스템의 모습을 결정하는 단계이다.

둘째, 시스템 설계 단계에서는 응용 소프트웨어를 구성하는 SWC의 이름, 데이터 타입, 인터페이스 등의 상태를 설계하고, SWC를 ECU별로 맵핑한 후 필요한 정보를 교환하기 위한 네트워크를 설계한다. 또한 각 SWC에 필요한 소프트웨어 실행의 최소단위인 런어블을 설계하고, 시스템 실행 조건에 부합도록 태스크 설계를 거쳐 런어블을 적절히 할당한다. 그리고 시스템 기능에 로직을 모델링 도구를 사용하여 설계한다. 설계된 정보는 SWC Description, ECU Resource Description, System Constraint Description, System Configuration Description 이름의 eXtensible Markup Language (XML) 파일로 저장되어 ECU 단계에서 사용하게 된다. 마지막으로 후속 단계에서 필요한 SWC 및 RTE 코드를 생성한다.

셋째, ECU 설계 단계에서는 RTE 레이어 아래에 위치한 BSW에 대한 설계를 수행한다. OS를 포함한 시스템 서비스 계층, ECU 추상화 계층, 마이크로프로세서

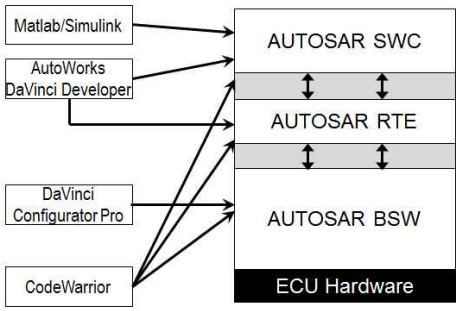


그림 10. 주요 설계 소프트웨어 도구
Fig. 10 AUTOSAR software design tools

추상화 계층 등에 포함된 수십여 개의 세부 모듈에 대해 개별적으로 속성을 설정하게 되고 모든 설정 정보는 ECU Configuration Description 파일에 저장된다. 그리고 마지막으로 각 모듈별로 c 코드를 생성한다.

넷째, ECU 통합 단계에서는 이전 과정에서 모듈별로 생성된 수많은 c 코드들을 실제 마이크로프로세서에 탑재할 수 있도록 하는 컴파일링, 디버깅하는 작업을 수행한다.

다섯째, 테스트 단계에서 마이크로프로세서의 기본 기능 정상동작 여부를 확인부터, HIL 환경을 이용하여 시스템의 기능 및 성능에 대한 평가를 수행한다.

IV. AUTOSAR 소프트웨어 설계 교육 사례 연구: 차량 윈도우 제어 시스템 개발 실습

본 장에서는 제안한 AUTOSAR 스타터 키트를 이용하여 AUTOSAR 기반 차량 윈도우 시스템 (Window Control System; WCS)을 단계적 절차에 따라 개발하는 설계 교육 과정을 소개한다. 개발에 필요한 AUTOSAR 설계 도구 및 설계 기준은 AUTOSAR R3.1규격과 호환된다.

1. 개발 환경 셋업

1.1 소프트웨어 설계 도구 셋업

윈도우 제어 시스템 소프트웨어 개발에 사용되는 주요 설계 소프트웨어 도구들은 그림 10과 같다. 시스템 설계에서는 한국 굿소프트웨어사의 오토웍스 (Autoworks) 도구를 사용하여 소프트웨어 구조, SWC, 네트워크 등을 설계하고, SWC 및 RTE 코드 생성은 독일 벡터(Vector) 사의 다빈치 디벨로퍼 (Davinci Developer) 도구를 사용한다. ECU 설계단계에서는 BSW 설

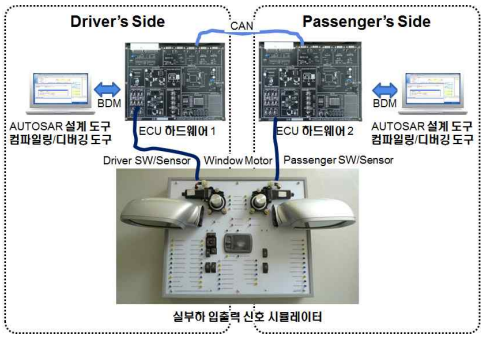


그림 11. AUTOSAR 스타터 키트 환경 셋업
Fig. 11 Set up for AUTOSAR Starter Kit

정 및 코드생성은 독일 벡터의 다빈치 컨피규레이터 (Davinici Configurator) 도구를 사용한다. 다빈치 컨피규레이터 도구는 OS, 통신, 기타 BSW 별로 별개의 설계 도구로 세분화 되어 있다. 기능 로직 모델링은 미국 매스웍스사의 매트랩/시뮬링크 (Matlab/Simulink)를 이용하였으며, 코드 통합, 컴파일링, 디버깅은 미국 프리스케일 (Freescale)사의 코드워리어 (CodeWarrior) 도구를 이용한다.

1.2 실습 환경 셋업

윈도우 제어 시스템은 운전석과 조수석 2개의 모듈이 CAN 네트워크로 연결되어 동작되는 시스템이다. 그러므로 그림 11과 같이 2개의 ECU 하드웨어를 CAN으로 서로 연결하고, 각 ECU 하드웨어를 실부하 입출력 시뮬레이터에 연결하여, 네트워크형 시스템 구조로 키트 환경을 셋업한다.

표 3. 기능 요구사항
Table 3. Functional requirements

번호	기능 요구사항
R01	차량의 키 상태가 IGN ON 상태가 아닐 때는 윈도우 모터는 동작 하지 않는다.
R02	Child Lock 상태에서는 운전석 측 조수석 윈도우 원격 스위치 조작으로만 조수석 윈도우 제어가 가능하다.
R03	윈도우 수동 상승/하강 스위치를 누르고 있는 동안 윈도우는 상승/하강 동작 한다.
R04	윈도우 스위치가 자동 상승/하강 스위치를 누르면 윈도우는 자동 상승/하강 동작 한다.
R05	윈도우의 수동 제어는 자동 제어보다 높은 우선 순위를 가진다
R06	LIMIT 스위치가 ON 상태가 되는 순간 상승/하강 동작 중인 윈도우 모터는 즉시 동작을 멈춘다.

2. AUTOSAR 소프트웨어 개발 절차

2.1 요구사항 분석

윈도우 제어 시스템은 그림 5와 같이 운전석 및 조수석의 윈도우 스위치 입력 상태에 따라 차량의 각 윈도우의 모터를 상승/하강 제어하는 시스템을 말한다. 이그니션(IGN) 스위치, 윈도우 동작 스위치, 윈도우 잠금(LOCK) 스위치와 윈도우 동작 제한(LIMIT) 스위치를 입력받아 제어 로직을 거친후 전용 모터 드라이버를 통해 윈도우 모터를 제어하게 된다. 다양한 스위치의 입력상태에 따라 운전석 및 조수석 윈도우를 제어하기 위한 기능 요구사항을 간략하게 정리하면 표 3과 같다.

2.2 시스템 설계

첫째, SWC 구조 설계에서는 개발 ECU 별로 SWC를 설계한다. 윈도우 제어 시스템은 운전

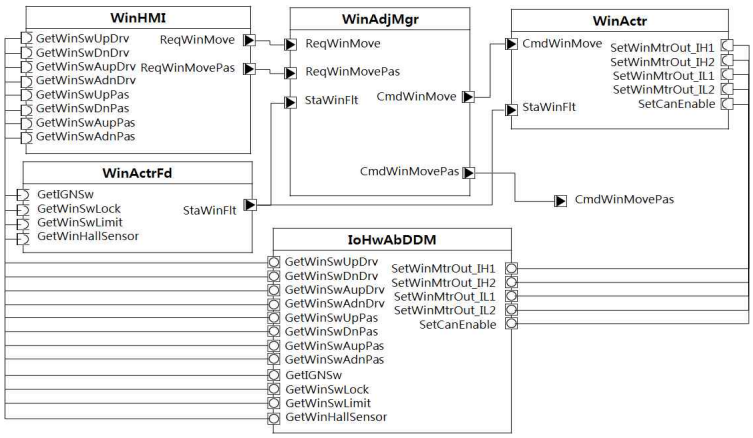


그림 12. AUTOSAR 소프트웨어 아키텍처
Fig. 12 AUTOSAR SWC Architecture

표 4. 소프트웨어 컴포넌트
Table 4. Software Components

SWC 이름	타입	설명
WinHMI	센서	차량 윈도우 스위치 입력 요구 감지
WinActrFd	센서	윈도우 모터 상태 감지
WinAdjMgr	응용	윈도우 제어 알고리즘
WinAtcr	액추에이터	윈도우 모터 출력 제어
IoHwAbDDM	추상화	추상화 컴포넌트

표 5. CAN 메시지
Table 5. CAN Message

메시지	ID	Format	Start bit	Length (bit)	signal
Pass RemReq	0x100	LSCAN	0	3	Pass Movement

석 ECU 및 조수석 ECU별로 구성되며 그림 12에는 운전석 ECU의 SWC 구조를 나타내었다. 주요 SWC는 윈도우 조절 스위치 입력을 담당하는 WinHMI SWC와 제어 알고리즘을 수행하는 WinAdjMgr SWC, 그리고 모터 출력을 담당하는 WinAtcr SWC가 있다. 각 소프트웨어 컴포넌트별 역할은 표 4와 같다.

둘째, 네트워크 설계에서는 먼저 네트워크 환경 구성 도구를 활용하여 운전석 ECU와 조수석 ECU간의 네트워크 환경을 설정한다. ECU간 통신은 Low speed CAN를 이용하여 125kbps 전송속도로 구성하였고, 표 5에 자세한 정보를 나타내었다. 운전석 모듈의 조수석 원격 제어 요청 메시지(PassRemReq) 및 조수석 윈도우 스위치 입력 시그널(PassMovement)를 정의하고 이 시그널들을 다빈치 디벨로퍼 도구를 활용해 각 ECU 상에 맵핑하여 ECU 통신 환경을 구성하였다.

셋째, 기능 로직 설계에서는 시스템 요구사항으로 바탕으로 실제 윈도우 제어 시스템을 위한 기능 제어 알고리즘을 설계한다. 알고리즘 설계는 매트랩/시뮬링크 도구를 사용하였으며 운전석 ECU의 최상위 알고리즘 최상위 모델은 그림 13과 같다. 각 블록들은 표 6과 같이 동일한 이름을 가지는 SWC의 런어블들의 기능을 수행한다. WinHMI블록은 윈도우 스위치 입력을 읽어들이는 기능을 수행하고 WinAdjMgr 블록은 윈도우 제어 관련 입력 신호를 분석해 윈도우

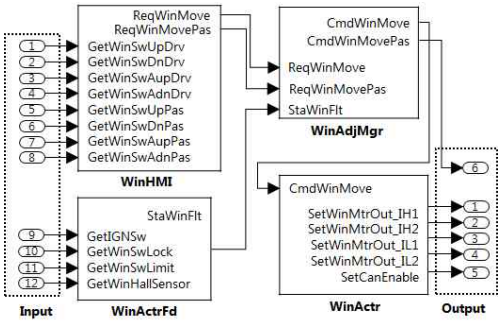


그림 13. 운전석 ECU의 알고리즘 모델
Fig 13. Algorithm model of Driver's ECU

표 6. 런어블 구성
Table 6. Runnables

SWC 이름	런어블 이름	설명
WinHMI	Runnable_WinHMI	윈도우 스위치 입력 신호 발생
	WinHMI_Init	WinHMI SWC 초기화
WinAdjMgr	Runnable_WinAdjMgr	윈도우 스위치 신호 입력 받아 윈도우 제어 모드 결정
	WinAdjMgr_Init	WinAdjMgr SWC 초기화
WinActrFd	Runnable_WinActrFd	윈도우 모터로부터 동작 상태 피드백
	WinActrFd_Init	WinActrFd SWC 초기화
WinAtcr	Runnalbe_WinActr	윈도우 제어 모드에 따른 윈도우 동작 명령
	WinActr_Init	WinAtcr SWC 초기화

제어 모드를 결정하는 기능을 한다. 그리고, Win- ActrFd 블록은 윈도우 모터의 동작 상태 피드백을 수행하고, WinActr블록은 윈도우 제어 모드에 따라 윈도우 모터를 동작시키는 기능을 수행한다.

넷째, 런어블 설계에서는 소프트웨어 컴포넌트의 기능을 수행하는 최소단위인 런어블을 설정한다. 런어블 설계 단계에서는 런어블 정의 및 실행 순서, RTE 이벤트 트리거 조건 및 주기 등의 속성들을 정의 하였다. 윈도우 제어 시스템의 소프트웨어 컴포넌트별 런어블 구성은 표 6과 같다.

다섯째, 태스크 설계에서는 런어블 설계 단계에서 구성한 런어블을 최종적으로 OS 스케줄러 실행 단위인 태스크에 맵핑한다. 윈도우 제어시스템은 간략 설계를 위해 WindowControlTask 단일 태스크로 구성하였으며 이 태스크는 10ms 간격으로 반복 수행된다.

표 7. SWC 및 RTE 관련 코드
Table 7. Codes for SWC & RTE

구분	주요 파일명	설명
SWC	WinHMI.c WinHMI.h	WinHMI SWC 설정 데이터 및 정보
	WinAdjMgr.c WinAdjMgr.h	WindAdjMgr SWC 설정 데이터 및 정보
	WinActrFd.c WinActrFd.h	WinActrFd SWC 설정 데이터 및 정보
	WinActr.c WinActr.h	WinActrSWC 설정 데이터 및 정보
	IoHwAb.c IoHwAb.h	IoHwAb SWC 설정 데이터 및 정보
RTE	Rte.WinActr.h,	WinActr SWC와 RTE 간의 인터페이스 정의
	Rte_WinActrFd.h	WinActrFd SWC와 RTE 간의 인터페이스 정의
	Rte_WinAdjMgr.h	WindAdjMgr SWC와 RTE 간의 인터페이스 정의
	Rte_WinHMI.h,	WinHMI SWC와 RTE 간의 인터페이스 정의
	Rte_IoHwAb.h	IoHwAb SWC와 RTE 간의 인터페이스 정의
	Rte.c	RTE API, COM callback, Task, LifeCycle API 설정 정보
	Rte.h	RTE 헤더 파일
	Rte_Type.h	RTE 데이터 타입 정의
	Rte_Cfg.h	RTE behavior 관련 사용자 정의
	Rte_Hook.h	RTE Hook function 정의
	Rte_Main.h	RTE lifecycle API 정의

마지막으로 여섯째, SWC 및 RTE 코드 생성에서는 시스템 디자인 단계에서 설정한 환경을 c 코드로 생성하는 단계이다. 다빈치 디벨로퍼 도구의 코드생성기능을 이용해 각 SWC 파일 및 관련 RTE 코드를 자동으로 생성 하였으며, 표 7에 관련 코드들을 간략히 정리하였다. 또한 AUTOSAR 시스템 설계 정보를 담고 있는 ARXML 파일, 그리고 태스크 정보를 가지는 OSEK Implementation Language (OIL) 파일들을 생성하였다.

2.3 ECU 설계

첫째, OS 설계 및 코드 생성에서는 윈도우제어 시스템의 실시간 OS 환경을 구성하기 위한 AUTOSAR OS 환경을 구성한다. AUTOSAR OS 코드 생성기를 활용하여 OIL 파일을 입력받아 추가적으로 필요한 태스크 및 스케줄링, 이벤트 정보들을 구성한 뒤 최종적으로 OS 환경 코드를 생성하였으며, 표 8에 관련 코드들을 간략히 정리하였다.

표 8. OS 관련 코드
Table 8. Codes for OS

구분	주요 파일명	설명
OS	osek.c, osek.h Os.h, Os_Cfg.h	OS 커널 파일
	tcb.c tcb.h	태스크 제어 블록과 OS 오브젝트 관련 설정 데이터 및 정보
	msg.c msg.h	메시지 관련 설정 데이터 및 정보
	trustfct.c trustfct.h	trusted function 관련 설정 데이터 및 정보
	intvect.c	인터럽트 벡터 테이블 설정 관련 데이터

표 9. 통신 모듈 관련 코드
Table 9. Codes for Communication modules

구분	주요 파일명	설명
Communication	Can.c, Can.h Can_Cfg.h, Can_Lcfg.c Can_PBcfg.c	CAN 모듈 관련 커널 파일 및 설정 정보 생성 파일
	Can_If.c, Can_If.h Can_If_cfg.h, Can_If_Lcfg.h CanIf_Pbcfg.h	CAN Interface 모듈 관련 커널 파일 및 설정 정보 생성 파일
	CanTp.c, CanTp.h CanTp_Lcfg.c, CanTp_PBcfg.c	CAN Transport Layer 모듈 관련 커널 파일 및 설정 정보 생성 파일
	COM.c, COM.h COM_Cfg.c, COM_Cfg.h Com_Lcfg.c, Com_PBcfg.c	AUTOSAR COM 모듈 관련 커널 파일 및 설정 정보 생성 파일
	PduR.c, PduR.h PduR_Cfg.h, PduR_Cfg.c PduR_LCfg.c, PduR_PBCfg.c	PDU Router 모듈 관련 커널 파일 및 설정 정보 생성 파일

둘째, 통신 모듈 설계 및 코드 생성에서는 ECU 간의 메시지 정보를 주고받기 위한 통신 관련된 하드웨어 환경을 설정하고 코드를 생성한다. CAN 코드 생성기를 활용해 AUTOSAR 시스템 설계 정보를 입력받은 후 시스템에 필요한 CAN 메시지와 관련 모듈을 생성하였으며, 표 9에 관련 코드들을 간략히 정리하였다.

셋째, 기타 BSW 설계 및 코드 생성에서는 앞서 설계한 OS와 통신 관련 모듈을 제외한 나머지 BSW 관련 하드웨어 모듈들을 설계하였다. 다빈치 컨피규레이터 도구를 사용하여 AUTOSAR 시스템 설계 정보를 입력받아 윈도우 제어 시스템에서 사용하는 BSW 단의 주요 모듈에 대한 환경을 구성하고 코드를 생성하였으며, 표 10에 관련 코드들을 간략히 정리하였다.

2.4 소프트웨어 통합

이전 단계들을 통해 생성된 윈도우 제어 시스템의 코드들의 주요 기능별 파일 크기는 표 11에

표 10. 기타 BSW 모듈 관련 코드
Table 10. Codes for Other BSW modules

구분	주요 파일명	설명
기타 BSW	EcuM.c, EcuM.h EcuM_PBcfg.c, EcuM_Cfg.h EcuM_Callout_Stubs.c	ECU State Manager 모듈 관련 커널 파일 및 설정 정보 생성 파일
	Mcu.c, Mcu.h Mcu_Irq.c Mcu_Irq.h Mcu_PBcfg.c, Mcu_Cfg.h	MCU 드라이버 모듈 관련 커널 파일 및 설정 정보 생성 파일
	Port.c, Port.h Port_PBcfg.c, Port_Cfg.h	PORT 드라이버 모듈 관련 커널 파일 및 설정 정보 생성 파일
	Dio.c, Dio.h Dio_Lcfg.c, Dio_Cfg.h	DIO 드라이버 모듈 관련 커널 파일 및 설정 정보 생성 파일

표 11. 윈도우 제어 시스템의 주요 코드
Table 11. Main Codes of WCS

구분	설명	파일 크기(Byte)	
		운전석ECU	조수석ECU
SWC	응용 소프트웨어 컴포넌트	1.1k	1.0k
RTE	AUTOSAR RTE	0.4k	0.3k
OS	AUTOSAR OS	7.3k	5.7k
COM	AUTOSAR COM	23.1k	20.1k
ECUM	ECU State Manager	3.3k	3.1k
MCU	MCU 드라이버	0.5k	0.4k
PORT	PORT 드라이버	0.3k	0.3k
DIO	DIO 드라이버	0.4k	0.3k
총 코드 크기		36.4k	31.2k

나타내었다. 각 ECU별로 수십 kByte 크기로 내장 플래시메모리 공간에 충분히 저장할 수 있는 크기이다. 이 코드들은 코드워리어 도구를 이용하여 프로젝트를 구성하고 코드 통합과 컴파일 및 디버깅을 진행하였다. 제어 시스템에 필요한 AUTOSAR BSW 라이브러리 파일 및 시스템 디자인 및 BSW 생성 파일들을 프로젝트에 추가하여 컴파일 및 디버깅 과정을 거쳐 최종적으로 운전석 ECU 하드웨어 및 조수석 ECU 하드웨어에 각각 다운로드하였다.

2.5 테스트

운전석 및 조수석 각 ECU 하드웨어에 윈도우 제어시스템 소프트웨어의 설치가 완료후 ECU 하드웨어의 윈도우 입력 스위치 와 LED 출력을 이용해 간단한 윈도우 제어 시스템에 대한 기능 테스트를 수행하였다. 이러한 ECU 하드웨어 기반 테스트를 완료 하면 최종적으로 실부하 입출력 시뮬레이터를 각각 ECU 하드웨어에 연결하여 실제 차량 환경과 동일한 HIL 기반에서 기능 테스트를 수행하였다. 표 3의 기능 요구사항을 기반으로 작성된 테스트 절차에 따라 윈도우 제어시스템의 각 기능에 대한 테스트를 수행하였으며 정상 동작함을 확인 할 수 있었다. 표 12에 테스트 절차와 테스트 결과를 정리하여 나타 내었다.

표 12. 기능 요구사항 테스트 결과
Table 12. Test result of functional requirements

번호	테스트 내용	테스트 결과
T01	IGN ON 상태에서, 윈도우 스위치 입력시 동작 여부 확인	OK
T02	Child Lock 상태에서, 운전석측의 조수석 윈도우 원격 스위치 조작으로만 조수석 윈도우 제어 여부 확인	OK
T03	윈도우 수동 상승/하강 스위치를 누르고 있는 동안, 윈도우는 상승/하강 동작 여부 확인	OK
T04	윈도우 자동 상승/하강 스위치를 누르면, 윈도우는 자동 상승/하강 동작 여부 확인	OK
T05	윈도우의 자동 제어 동작 중 수동 제어시, 수동 동작 여부 확인	OK
T06	LIMIT 스위치가 ON 상태시, 상승/하강 동작중인 윈도우 모터는 즉시 동작 여부 확인	OK

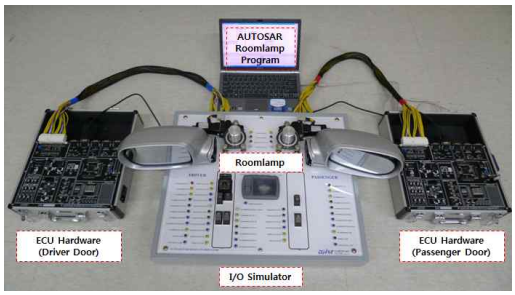


그림 14. AUTOSAR 스타터 키트 활용 사례 결과 (윈도우 제어 시스템)
Fig. 14 Result of case study using AUTOSAR starter kit(Window Control System)

3. 개발 결과

제안한 AUTOSAR 스타터 키트와 AUTOSAR 소프트웨어 개발 절차에 따라 개발된 윈도우 제어 시스템의 최종 결과물은 그림 14와 같다. 운전석 및 조수석 ECU 하드웨어는 각각 CAN 통신으로 연결되어 제어에 필요한 입출력 신호를 교환하고, 실부하 입출력 신호 시뮬레이터와 연결되어 윈도우 제어 스위치 입력 및 윈도우 모터 출력을 수행한다. 운전석 윈도우 제어 스위치로 운전석 윈도우를 수동 및 자동으로 상승, 하강 하는 기능들이 정상

적으로 동작 되었으며, CAN 네트워크를 통해 조수석 윈도우에 대한 제어 기능도 정상적으로 수행됨을 확인하였다. 또한 조수석 윈도우 제어 스위치로 조수석 윈도우를 수동 및 자동으로 제어하는 기능 역시 정상적으로 수행됨을 확인하였다.

결과적으로 AUTOSAR 스타터 키트를 이용하여 시스템 요구사항 분석부터, 테스트까지로 구성된 AUTOSAR 소프트웨어 개발 절차에 따라 용이하게 개발할 수 있었으며, 개발된 결과는 실제 차량과 유사한 운용환경에서 기능 검증을 할 수 있었다.

그리고 사례연구를 실습과정으로 교육시, 2~3일 정도기간이 소요 되었으며, 교육에 참가한 기업 엔지니어들은 실습과정을 통해 AUTOSAR 개념, AUTOSAR 설계 도구 사용법, 개발 절차 등 관련 기술을 전반적으로 습득할 수 있었다.

V. 결 론

본 논문은 자동차분야 소프트웨어 엔지니어들이 AUTOSAR 탑재한 차량 ECU를 편리하게 개발 할 수 있도록 HIL 기반의 AUTOSAR 스타터 키트를 제안하였다. 제안한 키트는 자동차관련 개발자들이 AUTOSAR를 쉽게 접근할 수 있도록 실제 차량환경과 유사한 ECU 하드웨어, 실부하 입출력 시뮬레이터로 구성되었으며, 개발과정을 단계적으로 실습해 볼 수 있는 예제 프로그램 및 교재도 포함 되어 있다.

또한 사례연구를 통해, 개발자들이 스스로 단계적 과정을 걸쳐 AUTOSAR 스타터 키트를 활용해 볼 수 있도록 방향을 제시함으로써 제안한 키트의 활용성을 확인할 수 있었다.

향후에는 제안한 개발 환경을 이용하여 대학, 연구소, 산업체의 엔지니어를 대상으로 수준별 맞춤형 AUTOSAR 소프트웨어 교육과정을 추진하여, 보다 편리하게 AUTOSAR 개념을 이해하고, 응용 시스템 개발에 접목할 수 있도록 보완해 나갈 예정이다.

References

[1] Autosar partnership. http://www.autosar.org

[2] A. Pretschner, M. Broy, I. Kruger, T. Stauner, “Software Engineering for Automotive Sustems: A Roadmap,” Proceedings of Future of Software Engineering, pp.55-71, 2007.

[3] K.V. Prasad, M. Broy, I. Krueger, “Scanning Advances in Aerospace & Automobile Software Technology,” Proceedings of the IEEE, Vol. 98, No. 4, pp.510-514, 2010.

[4] M.D. Natale, A. Sangiovanni-Vincentelli, “Moving From Federated to Integrated Architectures in Automotive: The Role of Standards, Methods and Tools,” Proceedings of the IEEE, Vol. 98, No. 4, pp.603-620, 2010.

[5] D.H Kum, J.K. Son, J.W. Son, M.G. Kim, “Automotive Embedded Systems Development and Validation with AUTOSAR and Model-based Approach,” Journal of Control, Automation, and System Engineering, Vol. 13, No. 12, pp. 1179-1185, 2007 (in Korean).

[6] D.H. Kum, S.H. Lee, G.M. Park, J.H. Cho, “Automated Testing System Using AUTOSAR XML,” Journal of IEMEK, Vol. 4. No. 4, pp.156-163, 2009 (in Korean).

[7] G.M. Park, D.H. Kum, S.H. Lee, “Model-Based Development and Test Method for The AUTOSAR Embedded Software,” Journal of IEMEK, Vol. 4. No. 4, pp.164-173, 2009 (in Korean).

[8] G.M. Park, D.H. Kum, B.J. Son, S.H. Lee, “Scheduling Design and Simulation of Software Component for EPS System based on AUTOSAR,” Journal of Institute of Control, Robotics and Systems, Vol. 16, No. 6, pp.539-545, 2010 (in Korean).

[9] Vector. http://www.vector.com

[10] Mathworks, http://www.mathworks.com

[11] S.H Lee, G.M, Park, D.H. Kum, Y.J. Kim, “Development of general embedded hardware environment for automotive embedded software design,” Proceedings of IEMEK 2011, pp. 170-182, 2011 (in Korean).

[12] Y.J. Kim, S.H. Jin, S.H Lee, “Development Methods AUTOSAR-based Software using General Embedded Hardware Environment,” Proceedings of IEMEK 2012, pp. 22-25, 2012 (in Korean).

[13] Freescale, http://www.freescale.com

저 자 소 개

이 성 훈

1996년 경북대학교 전자공학과 학사.
1998년 경북대학교 대학원 전자공학과 석사.
2007년 경북대학교 대학원 전자공학과 박사.

현재, 대구경북과학기술원 로봇시스템연구부 선임연구원.

관심분야: Safety critical 임베디드시스템, SIL/HIL Simulation.

Email: shunlee@dgist.ac.kr

김 영 재

2008년 경북대학교 전자전기컴퓨터학부 학사.
2010년 경북대학교 대학원 전자전기컴퓨터학부 석사.
현재, 대구경북과학기술원 로봇시스템연구부 연구원.

관심분야: 슈퍼컴퓨팅, 임베디드 시스템.

Email: youngjae@dgist.ac.kr

금 대 현

2001년 계명대학교 자동차공학과 학사.
2003년 계명대학교 대학원 자동차공학과 석사.
2011년 경북대학교 전자전기컴퓨터학과 박사수료.

현재, 대구경북과학기술원 로봇시스템연구부 선임연구원.

관심분야: 임베디드소프트웨어, 테스팅자동화, AUTOSAR, ISO26262.

Email: kumdh@dgist.ac.kr

진 성 호

1989년 경북대학교 전자공학과 학사.
1991년 경북대학교 대학원 전자공학과 석사.
2010년 경북대학교 전자전기컴퓨터학과 박사수료.

현재, 대구경북과학기술원 로봇시스템연구부 책임연구원.

관심분야: 차량용 임베디드 시스템, 모터 응용 시스템.

Email: sungho@dgist.ac.kr