Transactions of KSAE, Vol. 21, No. 1, pp.68-77 (2013)

pISSN 1225-6382 / eISSN 2234-0149
DOI http://dx.doi.org/10.7467/KSAE.2013.21.1.068

차량용 전자제어시스템을 위한 AUTOSAR 대응 경량화 소프트웨어 아키텍처 연구

이 강 석[1)] · 박 인 석[2)] · 선 우 명 호[3)] · 이 우 택[*4)]

창원대학교 대학원 제어계측공학과[1)] · 한양대학교 대학원 자동차공학과[2)] ·
한양대학교 미래자동차공학과[3)] · 창원대학교 제어계측공학과[4)]

AUTOSAR-ready Light Software Architecture for Automotive Embedded Control Systems

Kangseok Lee[1)] • Inseok Park[2)] • Myoungho Sunwoo[3)] • Wootaik Lee[*4)]

[1)]Department of Control & Instrumentation Engineering, Graduate School, Changwon National University, Gyeongnam 641-773, Korea
[2)]Department of Automotive Engineering, Graduate School, Hanyang University, Seoul 133-791, Korea
[3)]Department of Automotive Engineering, Hanyang University, Seoul 133-791, Korea
[4)]Department of Control & Instrumentation Engineering, Changwon National University, Gyeongnam 641-773, Korea
(Received 13 February 2012 / Revised 21 May 2012 / Accepted 18 July 2012)

Abstract : This paper presents AUTOSAR-ready light software architecture (AUTOSAR-Lite), which is a light weighted version of the AUTOSAR, for automotive embedded control systems. The proposed AUTOSAR-Lite reduces overhead problems caused by the excessive standard specifications of AUTOSAR. Concurrently, AUTOSAR-Lite keeps advantages of AUTOSAR such as a scalability, re-usability, reliability, and transferability. The fundamental design of AUTOSAR-Lite is originated from the AUTOSAR standard. AUTOSAR-Lite is composed of three layers such as an application software, runtime environment, and basic software layer. The application software layer adopts component-based design methodology as AUTOSAR. The runtime environment layer integrates interfaces between application and basic software layers. In case of the basic software layer, restrictions of the module configurations and interfaces of basic software are minimized. In order to validate the feasibility of AUTOSAR-Lite, a software design result based on AUTOSAR-Lite software architecture for electronic throttle control (ETC) system is suggested.

Key words : Software architecture(소프트웨어 아키텍처), AUTOSAR(오토사), AUTOSAR-lite(AUTOSAR 대응 경량화 소프트웨어 아키텍처), ETC(전자식 스로틀 제어), Layered architecture(계층 구조)

1. 서 론

소비자들의 편의를 위한 요구사항 증가 및 각종 규제 강화로 인하여 차량 내 내장형 제어시스템들의 규모는 1990년도부터 비약적으로 증가하고 있다.[1)] 급격하게 증가한 내장형 제어시스템의 규모만큼 기능 복잡도 역시 기하급수적으로 증가하고 있어, 내장형 제어시스템의 소프트웨어 개발 비용 및 시간을 절감하고, 유지보수성, 재사용성을 향상시키기 위한 노력들이 다양한 분야에서 시도되어 왔다.[2)] 이러한 시도 중 가장 대표적인 것은 새로운 Software architecture 개발에 관련 된 연구들이며,[3)] 최근 자동차 산업에서는 AUTOSAR 컨소시엄을 중심으로 활발하게 연구를 진행되고 있다.

AUTOSAR 컨소시엄은 완성차, 부품회사, 설계도

*Corresponding author, E-mail: wootaik@changwon.ac.kr

구 개발 회사들이 개방형 표준 차량용 소프트웨어 플랫폼을 제시하기 위하여 결성 되었다. 2005년 초에 최초로 Release 1.0의 사양서가 공개된 이후로, 2011년 현재까지 Release 4.0 까지 공개 되었다. 2008년부터는 AUTOSAR 컨소시엄에 속한 완성차 업체 및 부품업체들의 AUTOSAR 표준사양의 양산적용 사례가 늘어가고 있는 추세이다.

AUTOSAR 표준 도입이 현실화됨에 따라 관련 연구들도 분야별로 활발하게 진행되어 왔다. 시스템 레벨에서 ECU 맵핑이나, 응답시간을 최적화 하는 연구,[4,5] 어플리케이션 적용 연구,[6-9] 모델기반 소프트웨어 설계 기법 통합 연구,[10,11] 설계 도구 개발[12] 등 다양한 분야에서 AUTOSAR 표준 도입을 돕기 위한 기법들이 연구되고 있다.

AUTOSAR에서 제안하는 구조는 자동차의 전기 전자 시스템에 국한되는 것이 아니라 제어기의 소프트웨어 플랫폼을 개발하는데 필요한 범용의 가이드라인을 제시하고 있다.[9] 이러한 범용의 가이드라인을 준수함으로 인해서 소프트웨어의 신뢰성, 재사용성, 유지보수성 등의 다양한 측면에서 장점을 갖지만 많은 과도한 범용성으로 인하여 다음과 같은 다양한 오버헤드가 존재한다. AUTOSAR표준의 다양한 사양 버전의 혼재로 인한 호환성 문제들이 발생하기 쉬우며, AUTOSAR의 복잡한 스펙은 요구되는 하드웨어 리소스를 높인다. 이와 같은 오버헤드들은 추가적인 비용증가를 일으키며 AUTOSAR 도입의 장애 요소로써 작용하고 있다.[14-17]

반면 기존의 타겟에 특화된, 표준화 되지 않은 소프트웨어는 유지보수성, 재사용성, 신뢰성이 떨어진다. 하지만 요구되는 하드웨어 리소스가 상대적으로 낮고, 초기 개발비용이 적게 든다는 장점을 갖는다.

이와 같이 AUTOSAR와 같은 범용(General) 소프트웨어 아키텍처와 기존의 표준화 되지 않은 타겟 특화(Target Specific) 소프트웨어 아키텍처 사이에 Fig. 1과 같은 상호 보완적인 장단점이 존재한다.

이 연구에서는 AUTOSAR 표준에 부합하는 범용 소프트웨어 아키텍처와 기존의 타겟 특화 소프트웨어 아키텍처의 중요한 특징들을 선별, 채택하여 중소 규모의 차량용 전자제어시스템에 적합한 경량화

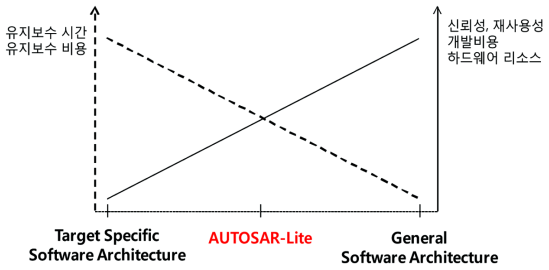

Fig. 1 Characteristics of AUTOSAR-Lite

된 소프트웨어 아키텍처(AUTOSAR-Lite)를 제시하고자 한다.

제시된 AUTOSAR 대응 경량화 소프트웨어 아키텍처를 이용하여 적은 비용으로 소프트웨어의 개발용이성, 유지보수성, 재사용성, 신뢰성을 확보할 수 있다. 또한 개발된 소프트웨어를 효율적으로 AUTOSAR 표준으로 전이할 수 있다.

이 논문의 구성은 다음과 같다. 2장에서는 AUTOSAR에 대해 살펴보고, 경량화 된 소프트웨어 아키텍처를 제시하기 위해 필수적인 요소, 오버헤드로 작용되는 요소들에 대해 분석한다. 이 분석 결과를 기초로 3장에서는 경량화 된 소프트웨어 아키텍처인 AUTOSAR-Lite를 소개하고, 이를 이용한 설계 예시를 제시한다. 마지막으로 4장에서는 결론과 함께 끝을 맺는다.

2. AUTOSAR 분석

AUTOSAR 에서 제시된 개발 방법론은 하드웨어와 네트워크 아키텍처에 대한 소프트웨어의 의존성을 최소화함으로써 개발 비용과 시간을 절약하는 것은 목표로 하고 있다. 이를 달성하기 위하여 AUTOSAR의 개발 프로세스는 크게 2 단계로 구성되어 있다.

첫 번째 단계에서는 가상의 통신 계층인 Virtual functional bus(VFB)를 통해 연결되는 어플리케이션 소프트웨어 컴포넌트(Software Component, SwC) 들을 개발한다. 이 과정에서 하드웨어와 네트워크에 의존적인 정보들은 배제되기 때문에 하드웨어나 네트워크의 설계 변경으로 인한 의존성을 최소화하고 어플리케이션의 기능적인 요소에 집중하여 개발이 이루어진다. 두 번째 단계에서는 ECU 하드웨어 설

계 정보를 기초로, SwC 들의 ECU 배치가 결정되고 가상 계층이었던 VFB가 각각의 ECU 별로 Runtime environment(RTE)라는 이름의 통합 계층으로 구현된다.

2.1 AUTOSAR 구조

2.1.1 계층 구조

AUTOSAR 표준에서 정의하는 소프트웨어 구조는 계층 구조를 기초로 하고 있으며, 크게 Application software(ASW), RTE, Basic software(BSW) 들로 구성되어 있다. 각각의 계층들은 표준화 된 API를 이용하여 연결되며 Fig. 2와 같은 구조를 이룬다.

2.1.2 통신 구조

AUTOSAR 표준에서 정의하고 있는 SwC의 통신은 포트와 인터페이스로 나누어 정의할 수 있다. 각각의 SwC 들은 통신을 위한 여러 개의 포트들을 가질 수 있으며, 인터페이스를 통해 통신 패턴이 결정된다. 포트는 자료 또는 서비스를 제공하는 경우에 PPort(Provide Port) 라고 불리며, 요청하는 경우에는 RPort(Require port)라고 정의된다.

통신 인터페이스는 패턴에 따라 Server-client 방식과 Sender-receiver 방식으로 구분된다. Server-client

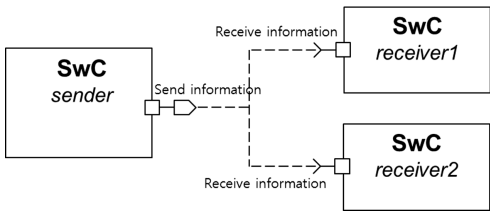

Fig. 3 Example of sender-receiver communication pattern

인터페이스는 널리 이용되는 동기 통신 방식으로 Server SwC는 Client SwC들의 요구에 따라 정보나 서비스를 제공한다. Fig. 3은 Sender-receiver 인터페이스 방식인 비동기 통신 방식 예제를 보여주고 있으며, Sender SwC에서 제공한 정보를 Receiver SwC들에게 배포하는 형태로 구현된다.

2.2 AUTOSAR 주요 특징

2.2.1 장점 및 기대효과

아래 Fig. 4와 같이 기존의 타켓 특화 소프트웨어 구조를 계층화 시키고, 표준화 된 인터페이스를 사용함으로써 AUTOSAR 표준을 충족시키는 차량용 전자제어시스템 소프트웨어를 개발 할 수 있다.

AUTOSAR 표준을 준수함으로써, 소프트웨어의 모듈화(Modularity), 확장성(Scalbility), 전이성(Trans-

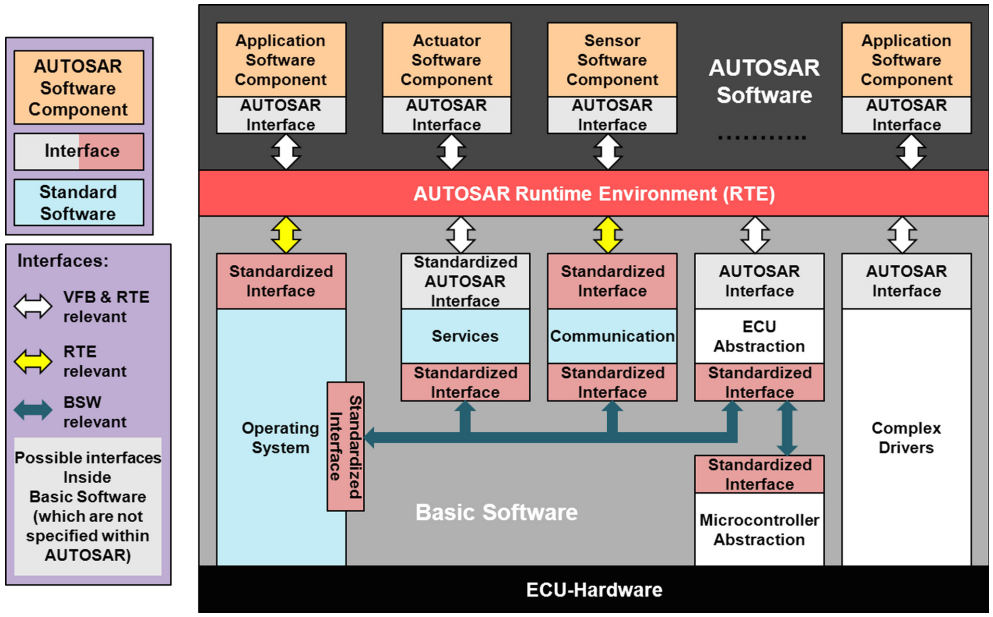

Fig. 2 AUTOSAR software architecture[13)]

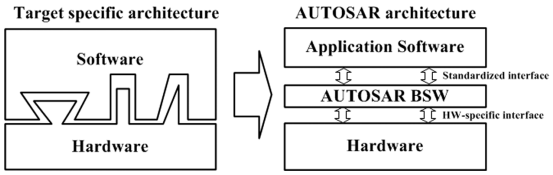

Fig. 4 Comparison between Target specific and AUTOSAR architecture[13)]

ferbility), 재사용성(Reusability) 들이 향상되는 긍정적인 효과들을 얻을 수가 있으며, 이를 바탕으로 장기적으로 개발 비용과 시간을 절감할 수 있을 것으로 평가 받는다.[18,19)]

2.2.2 단점

앞서 설명한 AUTOSAR 소프트웨어의 장점이나 기대효과들은 계층구조와 표준 인터페이스에 대한 방대한 규모의 사양을 충족함으로써 달성이 가능하다. 하지만, 구현과정에서 불가피하게 발생 가능한 오버헤드 및 한계 등의 단점이 아래와 같이 존재한다.[17)]

1) AUTOSAR에서는 방대한 규모의 표준 사양들을 요구[13)]하고 있기 때문에 이를 만족하는 소프트웨어 모듈을 만들기가 쉽지 않다. 또한 기존에 사용되던 소프트웨어 모듈, 가령 어플리케이션 모듈, OS, BSW의 재사용하기 위해서는 적지 않은 시간과 노력을 요구한다. 따라서 방대한 AUTOSAR 표준 사양에 능동적으로 대응하기 위해서는 전문 기술 인력을 필요로 하며, 기존 인력의 재교육이나 전문가 영입과 같은 추가적인 비용을 필요로 하며 개발의 용이성이 떨어지게 된다.
2) BSW의 경우 표준사양을 만족하기 위해서 상용 BSW 모듈을 구매하는 경우가 종종 발생하게 되며 이는 추가적인 비용의 증가로 이루어지게 된다. 80 여 개가 넘는 상용 BSW 모듈을 사용하기 위해서는 설정해야 할 파라미터의 수가 많을 뿐더러, 파라미터 간 의존성 때문에 설계 복잡성을 높이게 된다.[14-16)] 상용 BSW 모듈들은 상대적으로 높은 성능의 프로세서와 큰 메모리 용량을 필요로 하기 때문에 생산단가에 민감한 차량용 전자제어시스템 개발에 있어 무시하기 어려운 장애요소가 된다.[17)]
3) 계층 간, 모듈 간의 정보 교환을 위한 포트와 통신 인터페이스의 구성을 위하여 방대한 규모의 표준 사양들을 제시한다.[13)] 이러한 사양을 모두 만족하는 포트와 통신 인터페이스의 구성을 위하여 상용 프로그램을 사용하는 경우 코드의 복잡성이 증가하게 되며 이는 BSW의 경우와 마찬가지로 상대적으로 높은 성능의 프로세서와 큰 메모리 용량을 필요로 하여 개발 비용이 증가하게 된다.

이와 같은 AUTOSAR의 단점들은 개발참여 기관이 많고, 규모가 큰 차량용 전자제어시스템 개발에 있어서는 AUTOSAR의 장점 및 기대효과 같은 긍정적인 요소를 얻기 위한 투자로 받아들일 수 있겠으나, 개발기관이 적고, 규모도 작은 중소 규모의 차량용 전자제어시스템 개발의 경우 무시하기 어려운 장애요소로 여겨진다.

3. AUTOSAR-Lite

3장에서는 AUTOSAR의 단점을 최소화시키고, 기존의 타켓 특화 소프트웨어와 AUTOSAR의 장점들을 잘 반영할 수 있는 경량화된 소프트웨어 아키텍처인 AUTOSAR-Lite 소개한다.

AUTOSAR-Lite는 AUTOSAR 표준에 기반을 둔 계층적 구조로 구성된다. 또한 컴포넌트 기반 설계로 기본적인 개념은 AUTOSAR 표준과 동일하며 다른 SwC의 존재 및 위치 등에 대한 정보 없이 독립적으로 설계가 가능하다. 반면 Basic Software Layer의 경우, Specific software architecture 의 경우와 마찬가지로 하드웨어에 종속적인 구현을 허용함으로써 요구 하드웨어 리소스를 낮추고 소프트웨어를 경량화, 단순화 하였다.[3,18)]

3.1 AUTOSAR-Lite의 구조

AUTOSAR-Lite는 AUTOSAR 표준에 기반을 둔 계층적 구조로 구성되며 Fig. 5와 같다. 계층적 구조의 최하단에는 하드웨어에 종속적인 BSW 계층이 존재하며 최상단에는 어플리케이션의 기능을 구현하는 ASW가 존재한다. 또한 BSW와 ASW 사이에는 통신계층인 RTE가 존재한다. 이러한 계층적 구조로 인해서 각 계층은 다른 계층에 독립적으로 구현될 수 있으며 확장성, 재사용성이 확보되게 된다.

3.1.1 Application Software Layer

Application Software (ASW) 계층에서는 하드웨어와 네트워크에 의존적인 정보들은 배제하고 어플리케이션의 기능적인 요소를 포함한다. 컴포넌트 기반 설계로 기본적인 개념은 AUTOSAR 표준과 동일하다. ASW 계층에 위치하는 소프트웨어 컴포넌트는 다른 소프트웨어 컴포넌트의 존재 및 위치, 하드웨어 등에 대한 정보 없이 독립적으로 설계가 가능하도록 구성되며, 하나의 SwC는 하나의 .c, .h 파일로 구성된다.

ASW에 포함되는 소프트웨어 컴포넌트는 Sensor/Actuator SwC와 Application SwC로 분류된다. Sensor/Actuator SwC는 시스템에 실제 존재하는 Sensor와 Actuator를 추상화한 SwC로서, BSW를 통해 Sensor/Actuator와 상호작용한다. Application SwC는 Sensor/Actuator SwC를 제외한 일반적인 SwC로서 어플리케이션의 기능의 구현이 이루어진다.

Sensor/Actuator SwC와 BSW와의 통신 인터페이스는 AUTOSAR 표준의 Server-client 방식을 채택하였으며 SwC간의 통신 인터페이스는 Sender-receiver 방식을 채택하였다.

각각의 SwC는 Runnable을 포함한다. Runnable은 Scheduling 의 최소 단위이며 각 SwC 에서 구현되어야 할 기능을 함수 형태로 구현하게 된다. 같은 SwC 내의 Runnable 간 인터페이스는 Inter-runnable variable (IRV)을 통해 이루어진다. IRV는 SwC를 구성하는 파일의 내부의 정적변수로 구현한다.

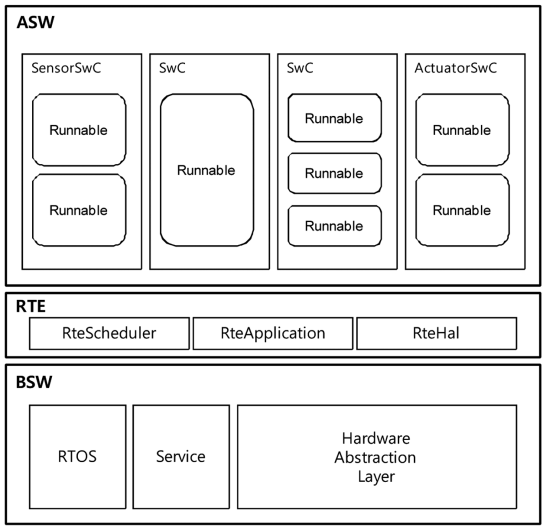

Fig. 5 AUTOSAR-Lite Software architecture

3.1.2 Runtime Environment

AUTOSAR에서 Runtime Environment(RTE)는 AUTOSAR SwC의 포트들을 연결 시켜주는 역할을 하는 가상의 통신 계층으로서 하드웨어나 네트워크의 의존성을 최소화하고 어플리케이션의 기능적인 요소에 집중하여 개발이 이루어질 수 있도록 한다. AUTOSAR-Lite에서는 RTE를 채택하여 이러한 장점을 계승하였다.[15)]

하지만 AUTOSAR-Lite에서는 RTE 구현에 있어 AUTOSAR의 표준 사양 중 필수 기능인 포트 및 인터페이스 기능을 위주로 구현하며 오류 처리, Trace 및 디버깅 등의 부가적인 기능을 배제하였다. 이로 인해 RTE 구성을 위한 상용 프로그램을 이용할 필요 없이 사용자가 직접 RTE를 구성할 수 있으며 코드의 복잡성을 낮출 수 있고, 요구 비용, 프로세서 성능, 메모리 용량 등의 오버헤드를 줄일 수 있다.

AUTOSAR-Lite의 RTE는 RteHal, RteApplication, RteScheduler의 세 가지 모듈들로 구성된다. Sensor/Actuaotr SwC와 HAL모듈간의 인터페이스와 SwC 간의 인터페이스를 구분하였고, 각각의 커넥터 구현은 RteHal 모듈과 RteApplication 모듈에서 담당한다. 또한 SwC내의 Runnable의 태스크로의 맵핑을 RteScheduler 모듈에서 담당한다.

3.1.2.1 RteHal

Sensor/Actuator SwC와 HAL간의 인터페이스 함수와의 맵핑을 통해 구현된다. 이러한 RteHal로 인해서 Application Software의 하드웨어, 네트워크의 의존성을 최소화할 수 있다.[15)] RteHal의 구현은 Fig. 6과 같다.

RteHal의 API는 BSW내부에 구현된 각 하드웨어 모듈의 API를 호출함으로써 Sensor/Actuator SwC와 HAL간의 인터페이스가 이루어진다. 이러한 구조를 채택함으로 인해서 BSW 구성의 자유도를 높일 수 있으며 기존에 사용하던 BSW 모듈의 사용을 가능하게 한다. 또한, ASW에서는 하드웨어와 네트워크에 의존적인 정보들을 배제하고 어플리케이션의 기능적인 요소만으로 구성할 수 있으며 하드웨어 및 네트워크 변경에 유연하게 대처할 수 있다.[8)]

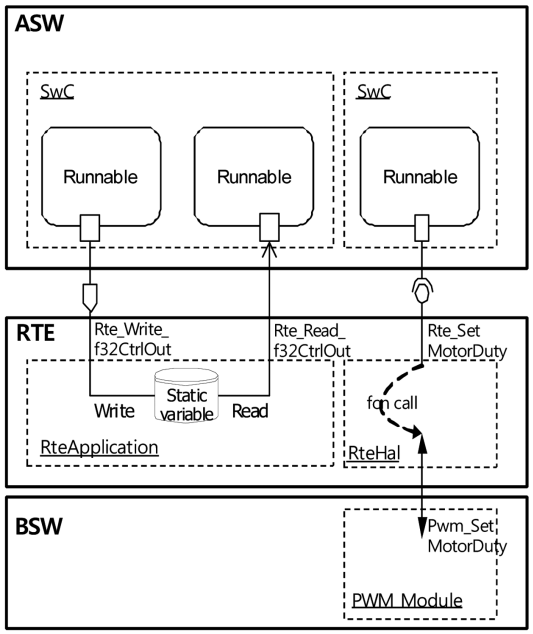

Fig. 6 Implementation of RteApplication and RteHal

3.1.2.2 RteApplication

각 SwC간의 인터페이스는 RteApplication 모듈에서 구현된다. RteApplication은 비동기 통신 방식인 Sender-receiver 인터페이스 형태로 구현되며 Fig. 6과 같다. RteApplication 은 내부에 ASW의 인터페이스를 위한 정적변수를 가지며 RteApplication의 API를 이용하여 정적변수에 값을 저장, 호출하는 방식으로 인터페이스가 이루어진다.

3.1.2.3 RteScheduler

RteScheduler 모듈은 Fig. 7과 같이 SwC내의 Runnable의 태스크로의 맵핑을 담당한다. RteScheduler에 구현된 태스크는 Real-time operating system (RTOS)에 의해서 수행되게 된다.

3.1.3 Basic software layer

Basic software(BSW) 계층은 Hardware abstraction layer(HAL), Real-time operating system(RTOS) 그리고 Service로 구성된다. BSW 계층의 경우 AUTOSAR와 비교해서 BSW 내부 계층 구조(MCAL & EAL, Comm.) 들을 단순화시킴으로써 효율성을 높이고, 표준 인터페이스에 대해 제약하지 않음으로써 기존 소프트웨어의 활용성을 높이도록 유도하였다.

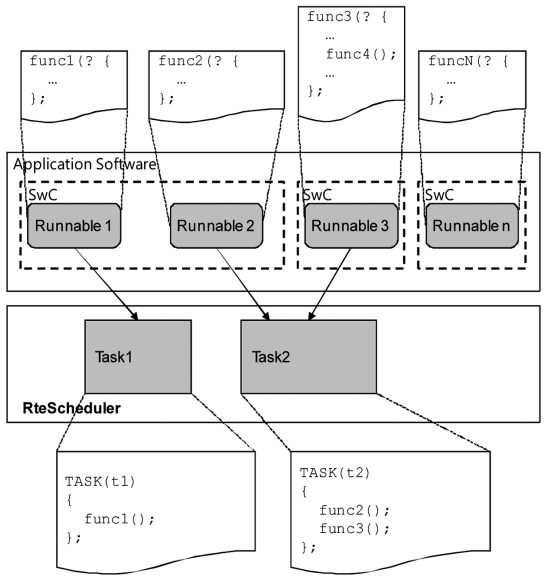

Fig. 7 Implementation of runnable and task

3.1.3.1 HAL

AUTOSAR 표준 에서는 MCU와 ECU Abstraction을 구분하지만, AUTOSAR-Lite에서는 별도로 구분하지 않으며 하드웨어를 추상화한 HAL을 갖는다.

3.1.1장에서 소개한 바와 같이 Sensor/Actuator SwC가 RTE를 통해 HAL과의 인터페이스가 이루어진다. 즉, 하드웨어 I/O를 통하여 입출력 되는 데이터들은 HAL을 통하여 상위 계층으로 전달되며, Sensor/Actuator SwC에서의 추상화를 통하여 데이터에 물리적인 의미가 부여되게 된다. 이러한 방식은 제어 알고리즘을 구현할 때 데이터를 쉽게 사용할 수 있도록 하며 개발된 소프트웨어의 가독성 높인다.[8)]

3.1.3.2 Real-time Operating System(RTOS)

AUTOSAR-Lite는 ASW에서 독립된 모듈로 구현된 태스크들의 동작을 보장하기 위하여 RTOS를 포함한다. 태스크들은 수행하는 동안 주기가 바뀌지 않으며 선언된 수행주기와 시간마진을 이용하여 시간 축 상에 배열된다. 여기서 태스크는 하나의 기능을 수행하는 단위가 아니라 동일한 수행 주기를 가지는 Runnable의 집단을 의미한다.

3.1.3.3 Service

AUTOSAR-Lite는 통신 및 메모리 관리기능 등의

부가 동작을 위한 Service모듈을 갖는다. Service 모듈의 통신기능은 네트워크에서 입출력되는 통신 메시지를 해당 프로토콜에 맞춰 인코딩 및 디코딩 하는 동작을 수행하게 된다. 이러한 Service 모듈을 구현함으로 인해서 통신 프로토콜과 하드웨어 변경에 대하여 SW의 변경을 줄이고 유연하게 대처할 수 있다.[9)]

3.2 사례 연구 : 전자식 스로틀 제어시스템

이 장에서는 AUTOSAR-Lite의 이해를 돕기 위하여 제시된 소프트웨어 아키텍처를 기반으로 한 차량의 전자식 스로틀 제어시스템(Electronic throttle control system, ETC)의 소프트웨어 구조를 제시한다.

Fig. 8은 ETC 제어 시스템을 묘사한 그림이다.[20)] ETC 제어 시스템은 제어 ECU와 기구부로 이루어진다. ETC 기구부는 내부에 DC모터와 ETC의 위치를 검출하여 아날로그 전압으로 출력해주는 위치센서를 포함한다.

또한 제어 ECU에는 Microcontroller Controller Unit(MCU)을 포함하며 ETC 의 DC모터 구동을 위한 H-Bridge Driver, ETC 위치 신호를 위한 필터회로, 제어 지령치 입력을 위한 가변저항 회로를 포함한다.

이러한 ETC 제어 시스템 구동을 위한 소프트웨어를 AUTOSAR-Lite를 기반으로 하여 Fig. 9와 같이 구성하였다. ASW는 Sensor, Control, Actuator SwC를 포함하며 이를 이용하여 ETC제어 알고리즘을 수행하게 된다. Sensor SwC의 두 Runnable은 각각 ETC의 위치센서와 제어 지령 위치 센서를 추상화한 것으로서, BSW에서 입력받은 ETC의 위치정보와 제어 지령치 정보를 제어 알고리즘에 사용 가능한 물리량으로 변환한다. 변환된 물리량으로부터

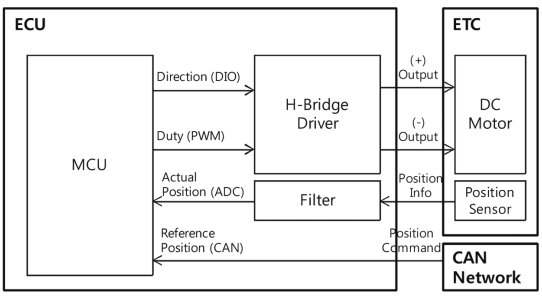

Fig. 8 Electronic throttle control system

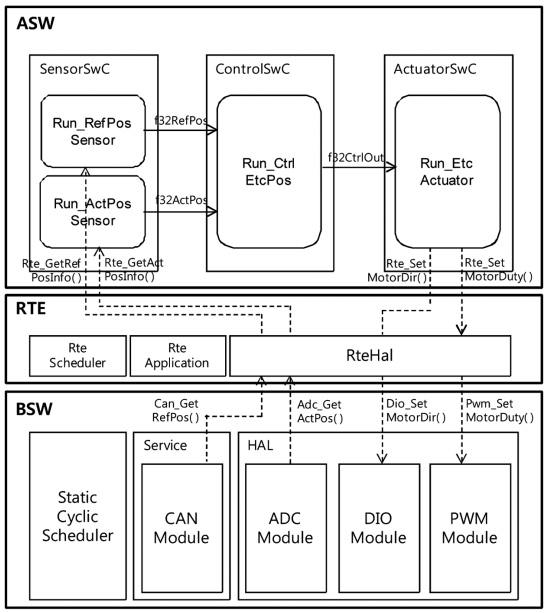

Fig. 9 Software architecture of ETC

Control SwC의 Runnable에서 제어기를 통하여 위치제어 알고리즘을 수행시켜 제어 출력 값을 결정하게 된다. 최종적으로 Actuator SwC에서 DC 모터 구동 신호를 출력하여 ETC를 구동시키게 된다. ASW의 구현은 Fig. 10과 같이 구현된다.

RTE는 AUTOSAR-Lite에서 제시하는 RteHal, Rte-Application, RteScheduler의 세 모듈을 포함한다. 앞서 언급한 바와 같이 RteHal을 ASW와 BSW간의 인터페이스 함수와의 맵핑을 통해 구현하게 됨으로써 ASW에서는 하드웨어와 네트워크에 의존적인 정보들을 배제하고 어플리케이션의 기능적인 요소만으로 구성된다. RTE의 구현은 Fig. 11과 같이 구현된다.

BSW는 Static Cyclic Scheduler와 HAL로 구성된다. RTOS로 다양한 기능을 갖는 OS를 배제하고 Non-preemptive 방식의 Static Cyclic Scheduler를 사용하여 태스크의 수행을 보장하며 소프트웨어를 경량화 시켰다. HAL은 ETC 제어 시스템에서 사용되는 Sensor, Actuator와 인터페이스하는 페리페럴을 선정 및 분석하여, 각 페리페럴의 모듈형태로 구성하였다. Serivce는 CAN 네트워크로 부터 입력되는 통신 메시지를 CAN 프로토콜에 맞춰 디코딩 하여 상위 계층으로 전달하는 동작을 수행하도록 구성하였다.

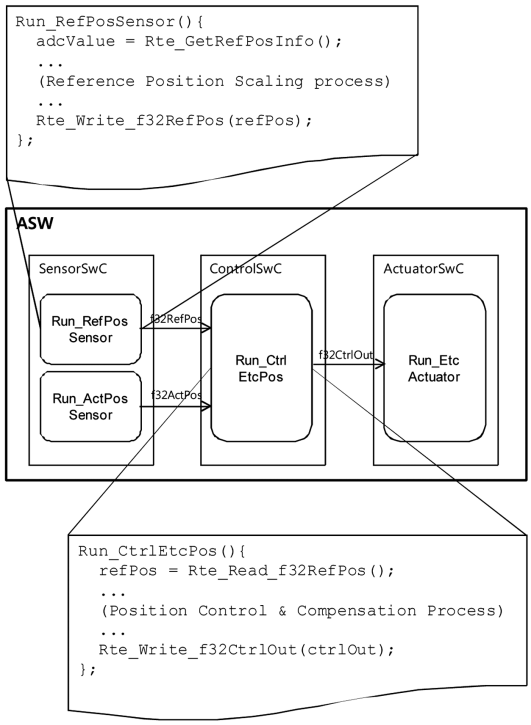

Fig. 10 Implementation of ASW

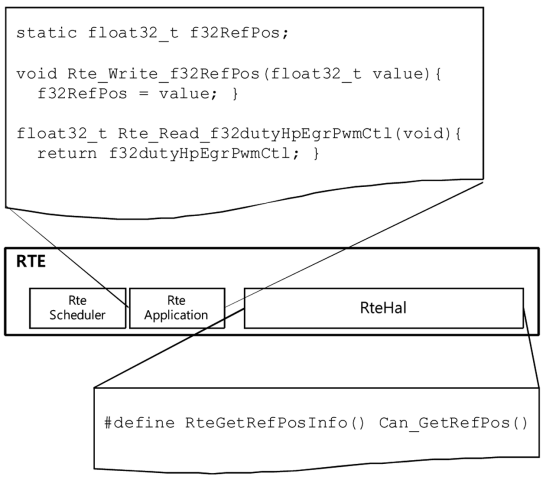

Fig. 11 Implementation of RTE

Table 1은 각각 AUTOSAR 와 AUTOSAR-Lite 기반의 ETC제어 소프트웨어를 분석, 비교한 결과이다. 이 논문에서는 ETC제어 소프트웨어의 RTE와 BSW만을 대상으로 분석한 결과를 나타내었으며 AUTOSAR 기반의 ETC 제어 소프트웨어는 상용 프로그램을 이용하여 생성하였다.

Table 1 Comparision of ETC Software

	AUTOSAR	AUTOSAR-lite
Number of functions	1,001	215
Cyclomatic complexity	3,293	384
Line of code	26,529	2,074
Number of files	272	84

Table 1에서는 각 소프트웨어의 파일, 코드, 함수의 개수 및 크기를 나타내었으며 소프트웨어 코드의 복잡성을 나타낼 수 있는 지표인 순환 복잡도(Cyclomatic complexity)[21]를 분석하여 나타내었다. Table 1의 결과에서 확인할 수 있듯이 AUTOSAR-Lite 기반의 소프트웨어는 AUTOSAR 기반의 소프트웨어에 대비하여 21% 수준의 함수 개수를 갖는다. 이는 AUTOSAR-Lite기반의 소프트웨어는 AUTOSAR 표준의 필수적인 기능만을 구현하였으며 AUTOSAR 표준의 다양한 기능을 지원하지 않는다는 단점을 나타낸다.

반면 AUTOSAR-Lite 기반의 소프트웨어는 AUTOSAR 기반의 소프트웨어에 대비하여 12% 수준의 순환 복잡도를 갖는다. 또한 31% 수준의 코드 라인 수와 8% 수준의 파일 개수로 AUTOSAR-Lite 기반의 소프트웨어가 AUTOSAR 기반의 소프트웨어에 대비하여 소프트웨어 복잡성이 낮다는 것을 확인할 수 있다. 이로 인하여 소프트웨어의 개발 요구 비용, 프로세서 성능, 메모리 용량 등의 오버헤드를 줄일 수 있다.

Fig. 12는 구성된 전자식 스로틀 제어시스템의 구동 결과를 나타낸다. 그림에서 ETC의 제어 지령치가 각각 50%, 100%로 변경되고, 변경된 제어 지령에 따라 ETC의 위치가 제어되는 것을 확인할 수 있다. 위치 제어 성능은 정상상태 오차 2%, 정착시간 100ms 이내, 백분율 오버슈트 0%로서 요구되는 ETC의 제어 성능을 만족한다. 또한 이 결과를 통해 제시된 AUTOSAR-Lite에 따라 구성된 ASW의 SwC와 Runnable 그리고 RTE 및 BSW의 모든 모듈이 정상적으로 동작함을 확인할 수 있다.

AUTOSAR-Lite에 따라 구성된 ETC 제어 소프트웨어는 AUTOSAR 기반의 소프트웨어 대비 파일 및 함수의 개수, 코드 라인 수가 작다. 또한 AUTOSAR

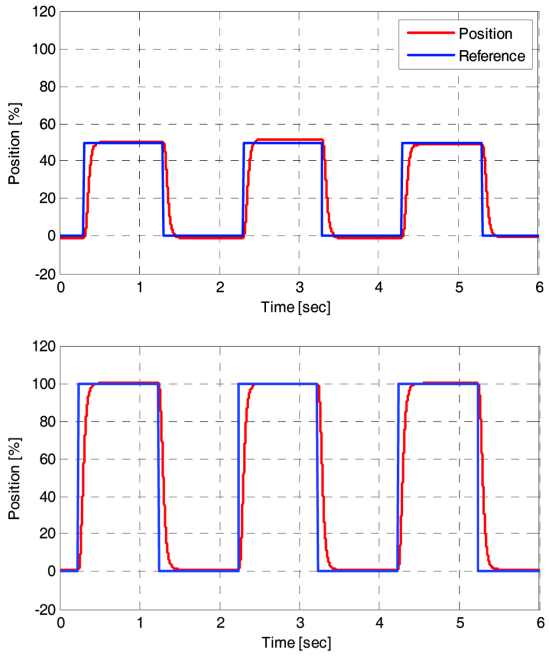

Fig. 12 Test Result of ETC

기반의 소프트웨어 대비 낮은 순환 복잡도를 갖고, 기존에 사용하던 Legacy 코드 및 BSW를 재사용 할 수 있다. 따라서 코드가 단순하고 소프트웨어 개발 비용, 개발 효율, 프로세스 요구 성능 면에서 오버헤드를 줄일 수 있다.

4. 결 론

이 논문에서 제시한 AUTOSAR-Lite는 AUTOSAR 표준과 기존의 타켓 특화 소프트웨어의 중요 특징을 선별, 채택하여 각 소프트웨어 아키텍처가 갖는 장점을 계승하였다.

AUTOSAR-Lite는 다음과 같은 특징을 갖는다.

1) ASW, RTE, BSW의 계층구조를 갖는다.
2) ASW는 컴포넌트 기반 설계로 기본적인 개념은 AUTOSAR 표준과 동일하다.
3) RTE는 역할에 따라 세 가지 모듈로 구분하여 구성하고 계층간, SwC간의 인터페이스 및 Runnable의 태스크로의 맵핑을 담당한다.
4) BSW는 내부 계층 구조를 단순화시키고 표준 인터페이스에 대해 제약하지 않음으로써 개발 효율성을 높이고, 기존 소프트웨어의 활용성을 높였다.

이러한 특징들로 인하여 AUTOSAR-Lite는 AUTOSAR 표준에 비하여 상대적으로 적은 비용으로 소프트웨어의 개발용이성, 유지보수성, 재사용성 신뢰성들을 확보할 수 있으며 AUTOSAR 표준으로의 전이에도 효율적으로 대응할 수 있다.

후 기

이 연구는 지식경제부와 한국산업기술재단의 전략기술인력양성사업과 지식경제부 산업원천기술개발 사업의 일환으로 수행된 연구결과(No. 10039673)이며, 2011년도 교육과학기술부의 재원으로 한국연구재단의 지원을 받아 수행된 연구이다(No. 2011-0017495). 또한 지식 경제부 에너지자원기술개발사업의 일환(2006ETR11P091C)로 수행되었다.

References

1) N. Navet, Y. Song, F. Simonot-Lion and C. Wilwert, “Trends in Automotive Communication Systems,” Proceedings of the IEEE, Vol.93, pp.1204-1223, 2005.
2) M. Broy, I. Kruger, A. Pretschner and C. Salzmann, “Engineering Automotive Software,” Proceedings of the IEEE, Vol.95, pp.356-373, 2007.
3) S. Voget, “Future Trends in Software Architectures for Automotive Systems,” Advanced Microsystems for Automotive Applications, Germany, pp.457-469, 2003.
4) J. Kim, G. Bhatia, R. Rajkumar and M. Jochim, “An AUTOSAR-Compliant Automotive Platform for Meeting Reliability and Timing Constraints,” SAE 2010-01-0448, 2011.
5) K. Lakshmanan, G. Bhatia and R. Rajkumar, “Integrated end-to-end Timing Analysis of Networked AUTOSAR-compliant Systems,” in Design, Automation & Test in Europe Conference & Exhibition (DATE), pp.331-334. 2010.
6) A. Graf, M. Jehle and G. Winkler, “Open Engine Management System for Future Engine Concepts Using the Example of the PowerSAR - Powertrain Manager,” SAE 2011-01-0209, 2011.

7) G. Park, S. Lee, D. Kum, C. Choi, W. Lee and W. Won, “Development of Software Component for EPS System based on AUTOSAR,” Annual Conference Proceedings, KSAE, pp.1867-1873, 2009.
8) J. Kim, W. Lee, S. Hong, S. Oh and K. Lee, “Development Technique of the AUTOSAR Standard-Based Body System Software,” Spring Conference of KSAE, Vol.3, pp.1423-1428, 2006.
9) C. Lim, J. Kim and W. Lee, “Development of a SW Platform for a MPC5554 Based Control System,” Symposium of Electric, Electronics and ITS part, KSAE, pp.72-77, 2006.
10) O. Niggemann, U. Eisemann, M. Beine and U. Kiffmeier, “Behavior Modeling Tools in an Architecture-driven Development Process - From Function Models to AUTOSAR,” SAE 2007-01-0507, 2007.
11) R. Yerushalmi and R. A. Felice, “Implementing AUTOSAR Atomic Software Components Using UML/SYSML in C,” SAE 2010-01-0265, 2010.
12) N.-Y. Cho, M.-G. Kyung and D. Min, “Development of AUTOSAR Conformance Test Architecture Modeling Tool Using Eclipse GMF,” Proceedings of Korea Computer Congress 2010, Vol.37, No.1(B), pp.337-340, KIISE, pp.337-340, 2010.
13) S. Bunzel, “Overview on AUTOSAR Cooperation,” in 2nd AUTOSAR Open Conference, Tokyo, Japan, 2010.
14) AUTOSAR Consortium, List of Basic Software Modules Ver.1.5.0, Release 4.0, Rev2, 2010.
15) AUTOSAR Consortium, Layered Software Architecture Ver.3.1.0, Release.4.0, Rev2, 2010.
16) AUTOSAR Consortium, Specification of Communication, Ver.4.1.0, Release.4.0, Rev2, 2010.
17) T. M. Galla and R. Pallierer, “AUTOSAR - Challenges and Solutions from a Software Vendor's Perspective,” Elektrotechnikund Informationstechnik, Vol.128, pp.234-239, 2011.
18) M. Di Natale and A. L. Sangiovanni-Vincentelli, “Moving from Federated to Integrated Architectures in Automotive: The Role of Standards, Methods and Tools,” Proceedings of the IEEE, Vol.98, pp.603-620, 2010.
19) D. Choi, D. Hong and S. Hong, “Embedded Real-time Software Architectures for Automotive Systems,” Symposium of Electric, Electronics and ITS Part, KSAE, pp.43-50, 2005.
20) S. Jin, J. Kang and W. Lee, “Electronic Throttle Body Model Allowing for Non-linearity of DC Motor Driver,” Transactions of KSAE, Vol.16, No.1, pp.71-77, 2008.
21) T. J. McCABE, “A Complexity Measure,” IEEE Transactions on Software Engineering, Vol.SE-2, pp.308-320, 1976.