Journal of The Korea Society of Computer and Information
Vol. 22 No. 10, pp. 47-54, October 2017

www.ksci.re.kr
https://doi.org/10.9708/jksci.2017.22.10.047

Design and Implementation of a Connected Car Platform Architecture for New ICT Convergence Services

Joongjin Kook*

Abstract

In this paper, we propose a connected car platform architecture called Mobile Second for developing of verity convergence services. A Mobile Second platform architecture is designed to provide more powerful and diverse convergence services for vehicles and drivers by applying technologies of Connected Car and ICT Convergence in various ways. The Mobile Second platform is implemented by applying Tizen IVI and Android to hardware platforms for IVI, Nexcom's VTC1010 and Freescale's i.MX6q/dl respectively. The Mobile Second platform provides the driver with the vehicle's information via IVI devices, mobile devices and PCs, etc., and provides Vehicle Selective Gateway(VSG) and Vehicle Control Framework for the driver to control his/her vehicle, and also provides a web framework to enable the use of VSG's APIs for the monitoring and controlling the vehicle information in various mobile environments as well as IVI devices. Since the Mobile Second platform aims to create new variety of services for Connected Car, it includes service frameworks for Smart Care / Self diagonostics, Mood & Entertainment services, and Runtime, libraries and APIs needed for the development of related applications. The libraries given by the Mobile Second Platform provides both a native library for native application support and a Java Script-based library for web application support, minimizing the dependency on the platform and contributing the convenience of developers at the same time.

▸Keyword: Connected Car, Mobile Second, ICT Car, IVI, In-Vehicle-Infotainment

I. Introduction

2009년 Luke Wroblewski가 제창한 모바일 퍼스트(Mobile First) 개념은 모바일 디바이스의 확산으로 인해 기존에 PC를 우선적인 수단으로 고려했던 것과 달리 점차 모든 기능들이 모바일 디바이스에 초점을 맞추어 개발되고 있음을 시사한다[1]. 이와 달리 모바일 세컨드(Mobile Second) 전략은 모바일 기기에서 이루어지던 작업을 다른 디바이스로 확대함으로써 새로운 산업분야와 결합하여 새로운 융합서비스를 창출하는 것을 목표로 한다.

특히 우리는 최근 가장 활발히 연구와 개발이 이루어지고 있는 커넥티드 카(Connected Car)에 초점을 맞추어 차량에 탑재되는 In-Vehicle-Infotainment(IVI) 디바이스를 주 디바이스(First Device)로 하고 스마트폰을 부 디바이스(Second Device)로 구성하여 IVI 디바이스 중심의 커넥티드 카 환경을 위한 서비스를 제공할 수 있도록 모바일 세컨드 플랫폼의 구조를 설계하였다. 물론 스마트폰은 부 디바이스로서 IVI 디바이스와 연계된 서비스를 제공할 수 있다.

최근 ICT(Information and Communication Technology) 컨버전스 기술들은 점차 자동차에 더해져 자동차를 더 이상 단순한 운송 수단이 아닌 IT 기술이 집약된 새로운 플랫폼으로 탈바꿈시키고 있다. ICT 컨버전스 기술이 결합된 ICT 카와 연계되는 모바일 세컨드 플랫폼은 차량과 모바일 기기를 연결하

• First Author: Joongjin Kook, Corresponding Author: Joongjin Kook
*Joongjin Kook (kook@smu.ac.kr), Dept. of Information Security Engineering, Sangmyung University
• Received: 2017. 06. 19, Revised: 2017. 07. 19, Accepted: 2017. 10. 11.

여 차량, 운행, 운전자와 관련한 다양한 서비스를 제공하기 위한 통합 플랫폼이며, 차량과 관련한 주요 정보를 운전자에게 제공하고, 반대로 운전자는 IVI 디바이스와 다른 모바일 기기를 기반으로 차량의 제어를 가능하게 한다. 또한 모바일 세컨드 플랫폼은 스마트 드라이빙 서비스, 스마트케어/자가정비 서비스, 무드/엔터테인먼트 등 다양한 자동차 관련 서비스를 창출할 수 있도록 다양한 형태의 서비스 프레임워크를 제공한다.

기존의 차량 헤드 유닛(Head-Unit)은 단순히 공조시스템(HVAC: Heating Ventilating and Air Conditioning)의 제어, 라디오 수신과 같은 단편적인 기능을 제공하였으나 커넥티드 카[2] 개념의 등장과 함께 헤드 유닛은 IVI 디바이스로 진화되어 AV 기능, 네비게이션, 차량 부속과 관련한 정보(e.g. 공기압, 오일게이지 등), 지불 기능 등 다양한 기능을 제공한다.

커넥티드 카 개념을 한 단계 더 확장한 모바일 세컨드 플랫폼은 IVI 디바이스나 모바일 기기를 통해 차량 정보를 운전자에게 제공하고 차량을 제어할 수 있도록 차량 제어 프레임워크를 기본적으로 제공하며, 네이티브 애플리케이션과 웹 애플리케이션 개발에 필요한 API를 지원한다. 또한 모바일 디바이스의 연동, 클라우드의 연동, 그리고 다양한 형태의 새로운 서비스 모델 개발이 가능하도록 서비스 프레임워크를 제공한다.

본 논문에서는 모바일 세컨드 플랫폼의 기본 구조를 제시하고, 가장 기반이 되는 차량 제어용 프레임워크와 게이트웨이 디바이스의 개발 방법, 그리고 차량 제어 프레임워크를 통해 응용 소프트웨어 개발이 가능하도록 하는 API와 라이브러리의 개발 방법을 설명할 것이다. 본 논문을 통해 모바일 세컨드 플랫폼 아키텍처의 개념과 구성, 그리고 차량 제어 프레임워크와 게이트웨이를 통해 제공되는 API를 기반으로 한 차량의 제어 방법을 보여줄 것이다. 모바일 세컨드 플랫폼은 차세대 ICT 카를 위한 새로운 형태의 융합서비스 개발을 위한 개방형 플랫폼으로 커넥티드 카를 위한 완전한 통합 플랫폼의 역할을 수행할 것이다.

II. Related Works

커넥티드 카(Connected Car)는 단순한 네트워크 연결을 뛰어 넘어 내 자동차와 모바일 디바이스를 연결하고, 다른 차량들과 연결되며, 주행, 사고, 정비, 엔터테인먼트 등 다양한 서비스들과 연결된다[3]. 또한 집, 사무실, 그리고 다양한 인프라까지 연결되도록 함으로써 자동차를 단순한 운송 수단이 아닌 IT 기술이 집대성된 새로운 형태의 디바이스로 탈바꿈시킨다. 커넥티드 카 플랫폼은 기본적으로 인포테인먼트 기능, 운전자의 안전과 관련한 기능, 차량의 진단 기능, 네비게이션, 결제 기능에 대한 컴포넌트를 포함해야 한다[6].

커넥티드 카는 자동차의 상태, 운전자의 상태, 그리고 외부 환경을 모두 고려한 통합적 상황 인지와 이를 기반으로 한 서비스로 연결이 될 수 있어야 한다. 따라서 자동차를 구성하는 주요 컴포넌트(센서, 액추에이터 포함)로부터 데이터를 수집할 수 있어야 하며, 운전자의 상태 정보(피로도, 감성 등) 획득이 가능해야 한다. 또한 자율주행을 고려한다면 차량 외부의 인프라, 주변 차량 정보, 보행자 정보의 획득도 가능해야 할 것이다. 이렇게 수집된 다양한 정보를 토대로 '주행 문맥(Driving Context)'에 대한 추론이 가능해야 하며, 추론 결과는 차량을 제어하거나 운전자를 제어할 수 있는 서비스와 연결되어야 한다.

커넥티드 카에서 자동차를 네트워킹이 가능한 IT 디바이스로 만들어 줌과 동시에 운전자와 차량의 상호작용을 위한 인터페이스 역할을 수행하는 것은 IVI(In-Vehicle-Infotainment) 디바이스이다. IVI 디바이스는 주로 AV 기기의 역할과 네비게이션 기능이 주가 되었지만 현재에 이르러서는 차량의 공조 시스템과 결합되고, ECU로부터 다양한 센서 정보들을 받아들여 자동차의 상태를 운전자에게 제공하기도 한다[4]. 또한 음성인식을 기반으로 운전자의 요청에 응답하기도 하며, 스마트폰과 연동되어 스마트폰이 제공하는 고유의 서비스를 IVI 디바이스에서 사용하는 것도 가능해졌다[5].

IVI 디바이스는 자동차와 운전자를 이어주는 인터페이스의 역할을 하고 있으며, IVI 디바이스의 역할이 확대됨에 따라 다양한 태스크를 효율적으로 수행하고, 다양한 네트워크 인터페이스를 지원하며, 다양한 애플리케이션을 실행시킬 수 있는 IVI용 SW 플랫폼이 요구되기 시작했다. 대표적으로 QNX의 Car2[7], MS의 Embedded Windows Automotive 7[8], MontaVista의 Automotive Technology Platform (ATP)[9], Samsung의 Tizen IVI[10], Wind River IVI[11] 등이 IVI 디바이스를 위한 SW 플랫폼 선점을 위해 경쟁하고 있는 상태이며, 완성차 제조사와의 협력을 통해 IVI용 SW 플랫폼이 탑재된 자동차들이 등장하고 있다.

최근 삼성전자가 인수한 하만은 커넥티드 카 플랫폼인 이그나이트(Ignite)를 출시했다. 하만의 이그나이트 플랫폼은 차량의 전장 부분으로부터 데이터를 입력받아 차량의 상태를 진단하고 각종 통계 정보를 제공하기 위한 Vehicle Analytics, Vehicle Health & Diagnostics 컴포넌트를 포함하고 있으며, V2H, V2V, V2I 기반의 차량간 네트워크를 지원한다. 또한 서비스 컴포넌트로서 Maas(Map as a Service), CaaS(Car as a Service) 컴포넌트를 포함하여 주행과 관련한 기본적인 서비스를 제공한다[20].

커넥티트 카의 서비스는 자율주행과 연관되는 주행 중심의 서비스와 사고 예방이나 엔터테인먼트를 포함하는 운전자 중심의 서비스로 구분될 수 있다. 주행 관련 서비스는 LIDAR를 기반으로 한 주변 차량과의 거리 인식 및 차량의 속도 제어, 카메라 기반의 차선 이탈 방지, 사물/사람 인식 등이 자율주행 분야에서 연구되고 있다.

운전자 중심의 서비스는 크게 사고 예방 관련 서비스와 엔터테인먼트 서비스로 구분할 수 있으며, 두 가지 서비스 모두 운전자의 상태를 기반으로 제공될 수 있어야 한다. 운전자의 상태를 파악하기 위해서는 생체 신호를 측정할 수 있는 센서(EEG,

GSR 등)나 카메라를 기반으로 한 얼굴, 표정, 안구의 인식과 추론 기술이 요구된다[4][19].

기존의 운전자 중심 서비스를 위한 디바이스는 차량의 헤드 유닛과 연계되지 않고 개별적으로 구성되는 형태였기 때문에 하드웨어와 소프트웨어 모두 차량에 대한 의존성이 높아 범용성이 떨어지는 한계가 있으며, 따라서 본 논문에서 제안하는 커넥티드 카 플랫폼은 다양한 유형의 디바이스를 쉽게 구성하고 헤드 유닛과 ECU를 연계하여 제어할 수 있으며, 이를 통해 다양한 서비스의 구현이 가능하도록 하고 있다.

III. Mobile Second Platform Architecture

자동차는 이제 더 이상 과거와 같이 단순한 운송을 위한 기계적인 장치가 아니다. 자동차는 각 부속의 기능별로 센서와 액추에이터를 가지고 있으며, 센서의 데이터 처리와 액추에이터의 제어를 위해 Micro Controller Unit(MCU)으로 이루어진 ECU가 구성된다. ECU는 차량을 구성하는 각각의 파트별로 Powertrain, Braking, Chassis Module 등으로 구성되며, 이들 간 데이터 교환을 위한 CAN, LIN, MOST 등의 다양한 네트워크로 구성된다[12][13]. 결국 차량을 구성하는 주요 제어시스템은 ECU들 간의 거대한 네트워크로 볼 수 있으며, 이러한 ECU 기반의 제어시스템을 표준화된 방식으로 관리할 수 있도록 하기 위해 OSEK[14]과 AUTOSAR[15]가 도입되었다.

ECU를 기본 단위로 하여 차량을 구성하는 주요 컴포넌트들이 모듈화되고 이들은 CAN, LIN, Flexlay, MOST 등의 네트워크로 연결된다. 실제로 차량의 정보를 IVI 디바이스를 통해 운전자에게 제공하고 운전자로 하여금 차량의 제어가 가능하게 하기 위해서는 AUTOSAR를 통해 추상화된 ECU의 접근과 제어가 가능해야 하지만, 차량의 제어 정보와 같은 중요 정보는 차량 제조사에서 공개적으로 관리하지 않기 때문에 역공학에 의한 분석 기법이나 제조사의 협력이 필요한 실정이다.

자동차를 구성하는 주요 모듈과 센서들을 토대로 필요한 데이터를 수집하고, 이를 기반으로 운전자에게 서비스를 제공하기 위해서는 기본적으로 차량 내부의 데이터 접근과 차량 주요 모듈의 제어가 가능해야 한다. 차량의 주요 모듈과 센서는 제조사와 차량 모델에 따라 네트워크의 구조와 특성이 상이하며, 수십~수백 개의 ECU로부터 필요한 데이터를 필터링하여 필요한 데이터를 적절하게 처리하기 위한 클러스터링 기술이 요구된다[16][17].

모바일 세컨드 플랫폼은 기존의 커넥티드 카 개념을 더욱 확장함으로써 스마트 드라이빙 서비스, 스마트케어/자가정비 서비스, 무드/엔터테인먼트 서비스 등 다양하고 새로운 서비스를 창출할 수 있도록 각 서비스의 구현을 위해 필요한 서비스 프레임워크를 제공한다. 그림 1은 모바일 세컨드 플랫폼의 구조와 차량, IVI 디바이스의 관계를 보여주고 있으며, 또한 소프트웨어적인 관점에서 표준과의 대응 관계를 보여준다.

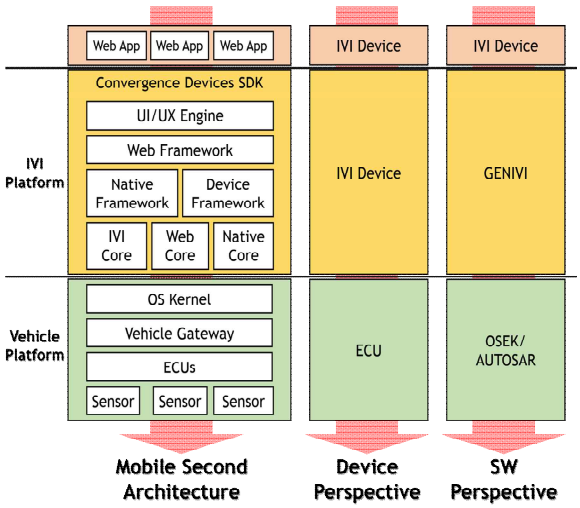


Fig. 1. Mobile Second Architecture & Device/SW Perspective

모바일 세컨드 플랫폼은 차량의 제어를 위한 'Vehicle platform'과 사용자 인터페이스 및 응용 소프트웨어, 서비스의 제공을 위한 'IVI Platform' 계층으로 크게 구분될 수 있다.

1. Vehicle Selective Gateway

자동차를 구성하는 주요 부품과 센서에서 발생되는 데이터는 ECU를 통해 관리되며, IVI를 통해 차량의 상태를 모니터링하고 터치스크린이나 음성을 통해 명령을 전달하기 위해서는 ECU와 IVI 디바이스 간 인터페이스가 필요하다.

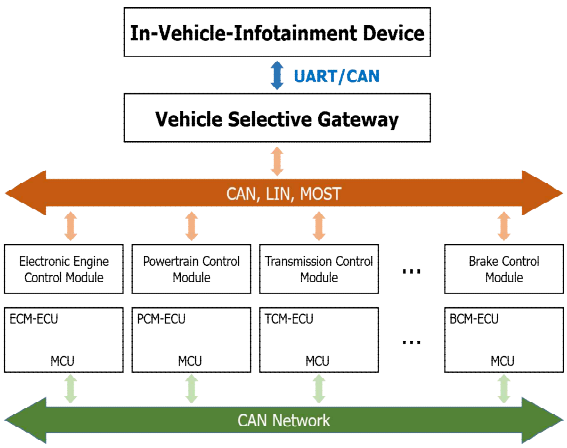


Fig. 2. ECU, VSG, and IVI Inter-connection

차량은 수십 개의 ECU(Electronic Control Unit)가 차량별 특성에 따라 단일 또는 복수의 네트워크에 연결되어 차량의 상태정보를 상호교환 하도록 설계된다. 예를 들어, 차량의 속도와 같은 기본적인 정보들은 바퀴에 설치되어 있는 엔코더 값을 이용하여 계측되고 이를 엔진제어 ECU에 전송하여 차량의 제어에 활용하는 형태이다. 그림 2는 ECU의 네트워크 구조를 표현한 것이며, 우리는 IVI 디바이스와 ECU의 통신을 위한 인터페

이스로 Vehicle Selective Gateway(VSG)를 개발하여 IVI 디바이스를 기반으로 한 차량의 제어가 가능하도록 하였다. 차량의 제어를 위해서는 ECU의 구성과 ECU별 센서 및 액추에이터의 정보를 알 수 있어야 하며, 우리는 모바일 세컨드 플랫폼의 검증을 위해 현대자동차의 소나타 모델과 기아자동차의 티볼리 모델에 대한 정보를 해당 업체로부터 제공받아 실험해 보았다.

VSG는 차량과 모바일 세컨드 디바이스를 연결시켜 줄 수 있는 게이트웨이의 역할을 수행하며, ECU에 탑재되는 소프트웨어로 구현되거나 별도의 하드웨어로 구현될 수 있다. 완성차를 제작하는 제조업체에서는 ECU 기반의 소프트웨어를 구현하는 것이 일반적인 형태이지만 우리는 실험을 위해 FreeScale의 MC9S08DZ60을 기반으로 VSG를 개발하였다. MC9S08DZ60 기반의 게이트웨이 구조는 그림 3과 같으며, CAN을 통해 차량의 ECU에 연결되어 필요한 시그널을 얻고 제어할 수 있다.

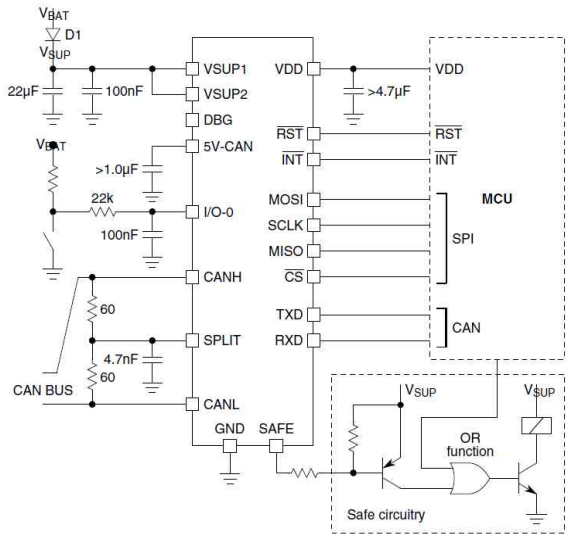


Fig. 3. VSG Architecture

차량 제어를 위한 VSG의 기능 구현을 위해서는 완성차 제조사의 모델별로 접근 및 제어가 가능한 ECU의 구조를 파악할 수 있어야 한다. 우리는 현대자동차의 소나타 모델과 기아자동차의 티볼리 모델에 대한 정보를 해당 제조사로부터 제공받아 실차에 적용한 테스트를 수행해 볼 수 있었다. 그림 4는 티볼리의 E/E 컴포넌트 구조를 분석한 것이며, Body Control Module(BCM)의 회로도를 제공받아 차량의 전장에 연결된 스위치 등 대부분의 기능과 작동구조에 대해서 파악할 수 있었다.

기본적으로 BCM은 차량의 바디와 관련된 모든 기능(도어열림/닫힘, 창문 열림/닫힘, 선루프, 공조기 On/Off, 헤드램프 On/Off, 스마트 키 등)을 직접 제어하는 데 활용되며, 따라서 VSG의 API는 BCM과 관련한 기능들을 기본적으로 포함한다.

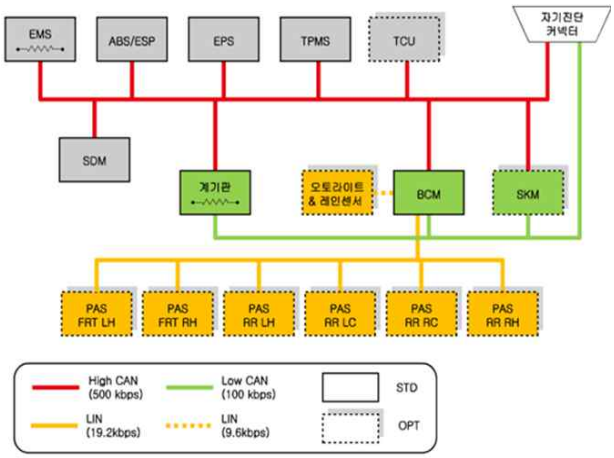


Fig. 4. E/E Components for TIVOLI

그림 5는 차량의 클러스터링 메시지정보와 차량정보 처리속도에 대해서 자체적으로 평가한 내용을 나타낸 것으로, 평가를 위한 장비로는 차량용 네트워크 설계를 위해서 산업적으로 가장 널리 사용되고 있는 VECTOR사의 CANoE를 이용하였다. 클러스터링된 메시지에서 실시간으로 수집되어야 하는 차량의 동적정보를 별도로 분류하였으며, 모두 8개의 정보(타이어상태, 기어상태, 에어백상태, 연료량상태, 엔진 RPM, 연료량상태, 종방향 가속도, 차량속도, 차량의 ID)가 이에 해당된다.

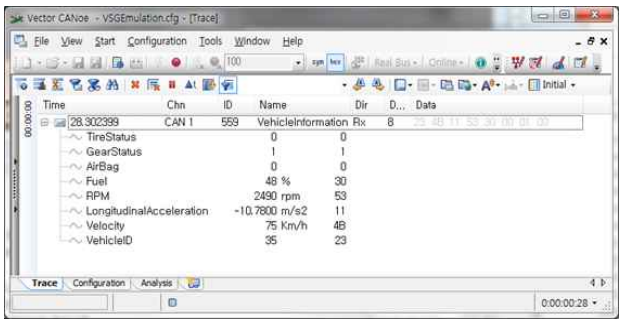


Fig. 5. Clustering Messages for VSG

그림 6은 클러스터링된 8개의 차량정보를 VSG에서 처리하는 데 소요되는 시간을 측정한 것으로 수집된 데이터의 평균 처리 시간은 100.003ms 였다.

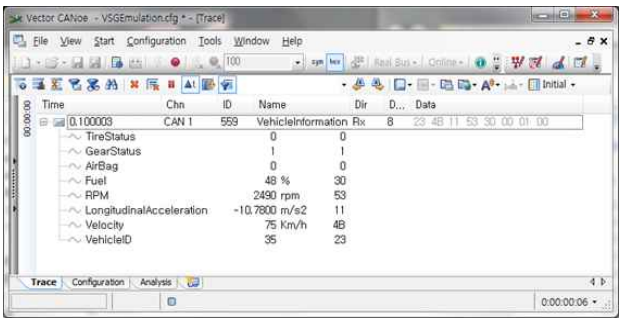


Fig. 6. Message Processing Latency of VSG

VSG는 차량 내부의 정보를 IVI 디바이스에 제공하고 IVI 디바이스를 통해 제어 명령을 수신할 수 있으며, 실제 제어 가능한 차량 구성 요소에 대해 모바일 세컨드 플랫폼의 차량 제어 프레임워크에서 제공하는 API는 표 1과 같다.

Table 1. VSG API Functions

API Function	Description
getStartVehicleStatus( )	Get initial status of vehicle
getDriverDoorOpen( )	Get status of driver door
getAssistDoorOpen( )	Get status of assistant door
getFuelLevel( )	Get fuel gauge
getEngineRPM( )	Get RPM of engine
getTotalDistance( )	Get accumulated distance
getDriveSpeed( )	Get current speed of a vehicle
getAccCount( )	The number of acceleration
getRedCount( )	The number of deceleration
getPBreak( )	Get status of parking break
getAirbagStatus( )	Get airbag status
getCheckEngine( )	Get abnormal code for engine
getVehicleID( )	Get vehicle ID
getSideBrakeStatus( )	Get status of hand break

VSG와 IVI 디바이스는 표 1에 정의된 API를 통해 양방향 정보 전달이 이루어지며, 클라우드와 연동되어 차량 외부에서도 모바일 디바이스나 PC 등에서 차량 정보의 모니터링과 제어가 가능하다.

2. Mobile Second Platform SW Architecture

모바일 세컨드는 커넥티드 카의 개념을 더욱 확장하고 진보시킴으로써 실제로 차량의 운행 과정에서 운전자에게 제공할 수 있는 서비스의 유형을 정의하고 이를 구체적으로 실현하기 위해 필요한 기술 요소를 세부적으로 표현한다. 그림 1의 모바일 세컨트 구조에서, 차량 내부의 각종 기능과 관련된 센서와 액추에이터를 관장하는 ECU는 전장용 게이트웨이 역할을 담당하는 VSG와 CAN으로 연결되며, VSG는 IVI 디바이스와 연결되어 차량과 운전자간 인터페이스의 역할을 수행한다. IVI 디바이스를 통해 본래 헤드 유닛이 제공하던 본연의 기능을 수행할 수 있도록 IVI core, Web core, Native core 컴포넌트가 구성되며, 컨버전스 디바이스(Convergence Devices) SDK를 제공하여 네이티브 애플리케이션뿐만 아니라 웹 애플리케이션 개발을 가능하게 한다. 이를 통해 차량의 상태 정보를 운전자에게 제공하고, 운전자의 요청에 반응하며, 클라우드 연동을 통해 외부의 다른 서비스와 연동이 가능하다.

모바일 세컨드 플랫폼과 연계 가능한 기본적인 서비스로는 스마트 드라이빙 서비스, 차량의 자가 진단 서비스, 사고의 예방과 구조 요청을 위한 스마트 케어 서비스, 그리고 쾌적한 운전 환경을 위한 무드/엔터테인먼트 서비스 등이 있으며, 다양한 서비스 프레임워크의 지원으로 인해 새로운 형태의 서비스를 쉽게 추가하는 것이 가능하도록 설계되었다.

2.1 Tizen IVI-based Mobile Second Implementation

본 논문에서는 VSG와 연결하기 위한 IVI 기기로 Nexcom의 VTC 1010을 사용하였으며, VTC 1010의 주요 특징은 다음과 같다:

- Intel® Atom™ processor E3827, 1.75GHz
- Dual SIM cards + dual WWAN modules support
- Wide operating temperature -30°C ~ 70°C
- Built-in CAN 2.0B. Optional CAN/OBDII module (CAN Bus 2.0B or OBDII SAE J1939)
- 4 x mini-PCIe socket rich expansion capability
- Wake on RTC/SMS via WWAN module

Voice communication via WWAN module

- Compliant with MIL-STD-810G
- Built-in U-blox M8N GPS, optional dead reckoning support

VTC 1010은 IVI SW 플랫폼 중 하나인 Tizen IVI 2.0을 지원하며, Tizen IVI는 리눅스 커널을 기반으로 하며, GENIVI 2.0 표준을 수용한다. 또한 차량 데이터의 접근을 위한 Vehicle Services API를 제공하며, 이를 통해 웹 애플리케이션을 구현할 수 있도록 Web API를 제공한다[18].

모바일 세컨드 플랫폼에서는 IVI 디바이스와 VSG의 연동을 위해 네이티브 프레임워크와 웹 프레임워크를 모두 이용하였다. 네이티브 프레임워크 기반의 UART 라이브러리를 구현하여 VSG의 API를 호출하도록 하였으며, UI와 서비스 구현을 위해서는 크로스 프레임워크인 크로스워크(Crosswalk)를 기반으로 웹 애플리케이션의 형태로 구현하였다. VSG의 차량 제어 API 호출을

Table 2. VSG Command Packet Format

ECU Command	Packet Format & Value			
	Type	Content	Position	Value[0-4]
get_driver_seatbelt_status	0xAA	0x41	0x10	0x01 0x00 0x00 0x00 0x55
get_start_vehicle_status	0xAA	0x41	0x13	0xFF 0x00 0x00 0x00 0x55
set_start_vehicle	0xAA	0x51	0x13	0xFF 0xFF 0x00 0x00 0x55
set_stop_vehicle	0xAA	0x51	0x13	0xFF 0x00 0x00 0x00 0x55
get_driver_temperature_c	0xAA	0x41	0x21	0x01 0x00 0x00 0x00 0x55
set_driver_temperature_c	0xAA	0x51	0x21	0x01 0x80 0x80 0x00 0x55
get_passenger_temperature_c	0xAA	0x41	0x21	0x10 0x00 0x00 0x00 0x55

위한 UART 라이브러리는 VSG의 커맨드 패킷 형식(command packet format)에 해당하는 바이트 열을 표 2와 같이 정의한다.

VSG로 전달할 제어 커맨드는 표 2의 API 함수들로 래핑(wrapping) 되어 라이브러리 형태로 빌드함으로서 IVI 디바이스의 애플리케이션에서 사용이 가능하도록 하였다. UART 라이브러리를 기반으로 하는 Node.js 애플리케이션에서 VSG API의 호출 방식은 표 3과 같은 형태가 된다.

Table 3. VSG API Usage Example

```
VSG UART Module Functions

1. Load serial module
var my = require('./build/Release/serial');
2. getDriverSeatBeltStatus()
--> ret: 1 (ON)
--> ret: 0 (OFF)
console.log(my.getDriverSeatBeltStatus());
3. getStartVehicleStatus()
--> ret: 1 (ON)
--> ret: 0 (OFF)
console.log(my.getStartVehicleStatus());
4. setStartVehicle(unsigned char onoff)
--> onoff: 1 (ON)
--> onoff: 0 (OFF)
console.log(my.setStartVehicle(onoff));
```

IVI 디바이스의 VSG 제어 애플리케이션에서 Node.js를 사용한 이유는 이후 모바일 디바이스와 클라우드 환경에서 VSG의 API 호출이 용이하도록 하기 위해서이며, WebSocket 또는 Socket.IO를 추가로 사용하여 VSG 애플리케이션에 서버 기능을 추가하면 차량의 원격 제어가 가능해 진다.

2.2 Android-based Mobile Second Implementation

본 논문에서는 VSG와 연결하기 위한 IVI 디바이스를 안드로이드 기반으로 개발하기 위해 Freescale사의 i.MX6dl 과 i.MX6q를 사용하였으며, 그림 7은 i.MX6의 블록 다이어그램을 보여준다. 안드로이드 기반의 i.MX6 플랫폼은 차량 제어를 위한 ECU와의 FlexCAN 인터페이스를 포함하며, IVI 디바이스의 역할을 동시에 수행할 수 있도록 다양한 하드웨어 인터페이스와 멀티미디어 처리 기능을 포함하고 있다.

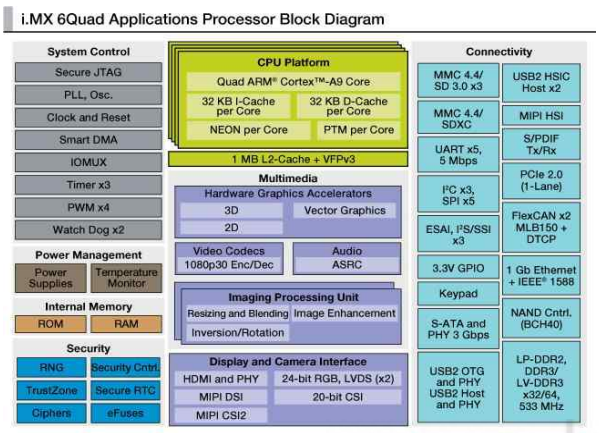


Fig. 7. Freescale iMX 6dl/6q Block Diagram

i.MX6 기반의 IVI 디바이스에는 안드로이드 4.4를 이식하였으며, 따라서 VSG와 통신하여 차량을 제어할 수 있도록 하기 위해서는 안드로이드 기반의 네이티브 애플리케이션 또는 웹 애플리케이션에서 사용 가능한 API의 개발이 요구된다. 우리는 네이티브 애플리케이션 개발을 위해 필요한 VSG 라이브러리를 안드로이드의 프레임워크 계층에서 서비스(Service) 컴포넌트 형태로 개발하였으며, VSG API를 사용하는 안드로이드 애플리케이션 개발을 용이하게 하기 위해 관련 클래스들을 묶어 jar 형식의 클래스 라이브러리로 구현하였다. VSG API를 제공하는 안드로이드 서비스 컴포넌트의 바인딩 방식은 다음과 같다.

```
IFlexcanService flexcanService    =
    IFlexcanService.Stub.asInterface(
        ServiceManager.getService("flexcan"));
```

VSG API의 구성은 표 1과 같으며, 호출 방식은 다음과 같이 FlexcanService 컴포넌트의 인터페이스로 지정된다.

```
flexcanService.setStartVehicle();
```

모바일 세컨드 플랫폼은 안드로이드 네이티브 애플리케이션 뿐만 아니라 웹 애플리케이션을 지원하기 위해 그림 8과 같이 웹 프레임워크를 제공한다.

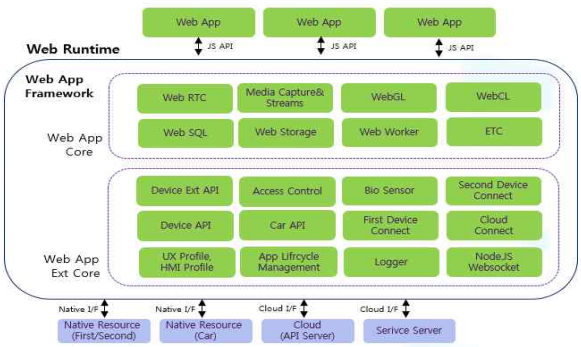


Fig. 8. Web Runtime Architecture of Mobile Second Platform

또한, 웹 런타임을 기반으로 웹 애플리케이션의 개발을 위해 필요한 VSG 웹 인터페이스를 개발하였다. 이를 위해 크로스워크(Crosswalk)를 사용하였으며, 자바 스크립트를 통해 VSG API의 호출이 가능하도록 하였다. 우리가 제공하는 VSG의 웹 애플리케이션 인터페이스는 다음과 같은 형태로 구현되었다.

```
XWalkView.addJavascriptInterface(
    new WebAppInterface(), "AndroidVSG");

public class    WebAppInterface {
    @org.xwalk.core.JavascriptInterface
    public void setStartVehicle() {
        flexcanService.setStartVehicle();
    }
}
```

웹 애플리케이션의 구현을 위해 사용되는 자바 스크립트에서는 다음과 같은 형태로 VSG API를 호출할 수 있다.

```
AndroidVSG.setStartVehicle();
```

2.3 Experiments

모바일 세컨드 플랫폼은 커넥티드 카 환경에서 차량의 데이터 수집과 차량의 제어를 기반으로 하는 차량 제어 프레임워크와 다양한 주변 장치를 통해 운전자의 안전과 편의성을 도모할 수 있는 서비스의 구현이 쉽게 가능하도록 웹 애플리케이션 프레임워크를 포함한다. 또한 모바일 세컨드 플랫폼은 Tizen IVI와 안드로이드를 기반으로 설계 및 구현되어 개방성을 지향하며, 쉽게 새로운 디바이스를 추가하고 이를 통해 새로운 서비스의 구현을 가능하게 한다. 우리는 모바일 세컨드 플랫폼을 실제 차량에 적용하고 애플리케이션 개발을 위해 설계한 IVI-VSG API를 통해 IVI 디바이스와 모바일 기기를 이용한 차량의 제어 가능성을 검증해 보았다.

Fig. 9. Mobile Second Adapted on Real Car(TIVOLI)

그림 9는 쌍용자동차의 TIVOLI 모델에 안드로이드 기반의 모바일 세컨드 플랫폼을 실제 적용하여 IVI 디바이스와 스마트폰을 통해 차량 데이터의 수집과 제어를 실험해 본 것이며, 표 3은 커넥티드 카 플랫폼을 기반으로 한 제어 대상 컴포넌트와 제어를 위해 사용한 디바이스의 유형을 나열한 것이다. 기 분석된 해당 모델의 E/E 아키텍처를 기반으로 차량 시동의 On/Off, 비상등 점멸, 도어락의 잠금과 해제, 창문의 개폐, 경적 울림, 헤드라이트 제어, 썬루프 개폐 등의 제어가 가능했으며, 클라우드를 연동하여 IVI 디바이스뿐만 아니라 스마트폰과 스마트워치를 이용한 제어도 동일하게 수행 가능함을 확인할 수 있다.

Table 3. VSG API Functions

Evaluation Item	Device Type
Vehicle startup On/Off	IVI, SmartPhone, SmartWatch
Emergency light On/Off	IVI, SmartPhone, SmartWatch
Door lock/unlock	IVI, SmartPhone, SmartWatch
Window Open/Close	IVI, SmartPhone, SmartWatch
Horn	IVI, SmartPhone, SmartWatch
Low beam On/Off	IVI, SmartPhone, SmartWatch
Sunroof On/Off	IVI, SmartPhone, SmartWatch

하지만, 특정 차량 모델의 ECU에 내장되는 프로세스의 형태가 아닌 추가적인 하드웨어 인터페이스와 다층의 소프트웨어 계층을 거쳐 처리되는 방식이기 때문에 헤드 유닛을 통한 차량 정보의 요청이나 차량 컴포넌트를 제어하기 위한 명령이 실제 수행될 때 까지 소요되는 지연 시간이 존재하기 때문에 실시간성이 요구되는 서비스의 지원을 위해서는 하드웨어의 구성과 소프트웨어의 최적화에 대한 연구가 더 진행되어야 한다.

IV. Conclusions

우리는 Tizen IVI 기반의 모바일 세컨드 플랫폼과 안드로이드 기반의 IVI 플랫폼에 대해 각각 VSG를 기반으로 한 차량 제어 API를 개발하였으며, 응용 소프트웨어 계층에서 이 기능을 사용할 수 있도록 네이티브 애플리케이션과 웹 애플리케이션에 대한 라이브러리를 구현하였다. 현대자동차의 소나타 모델과 기아자동차의 티볼리 모델에 각각 Tizen IVI와 안드로이드 기반의 모바일 세컨드 플랫폼을 적용시켜 VSG와 IVI 디바이스의 인터페이스를 구성하였으며, IVI의 웹 애플리케이션을 구현하여 차량 데이터의 수집과 제어를 실험해 보았다.

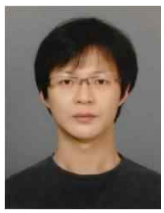
모바일 세컨드 플랫폼은 커넥티드 카를 더욱 확장하기 위한 아키텍처로써 차량의 각종 정보를 운전자에게 제공해줌과 동시에 스마트 드라이빙 서비스, 차량의 자가 진단 서비스, 사고의 예방과 구조 요청을 위한 스마트 케어 서비스, 그리고 쾌적한 운전 환경을 위한 무드/엔터테인먼트 서비스 등 주행과 운전 중 발생될 수 있는 다양한 상황에 대한 서비스의 제공을 가능케 한다. 본 논문에서는 특히 스마트 드라이빙 서비스와 관련하여 모바일 세컨드 플랫폼이 가장 기본적으로 제공해야 하는 차량의 제어와 관련한 프레임워크의 연구를 통해 차량에서 운전자와의 인터페이스를 담당하는 IVI 디바이스가 이러한 기능을 포함할 수 있도록 하기 위한 방법을 연구하고 구현해 보았다. 물론 완벽한 차량 제어를 위해서는 차량의 제어를 위해 필요한 정보를 제조사로부터 온전히 얻을 수 있어야하기 때문에 모든 완성차에 대해 우리가 개발한 플랫폼을 적용시키는 것은 현재로서는 불가능하지만, 완성차 업체에서 더욱 진화된 커넥티드 카 환경을 만들고 다양한 서비스를 제공하기 위한 레퍼런스의 역할을 수행할 수 있을 것으로 기대한다.

REFERENCES

[1] Mobile First, https://www.lukew.com/ff/entry.asp?933
[2] McKinsey & Company, “Connected car, automotive value chain unbound,” Apr. 2017.
[3] Riccardo Coppola and Maurizio Morisio, “Connected Car: Technologies, Issues, Future Trends,” ACM Computing

Surveys, Vol. 49, No. 3, Article 46, Oct. 2016.
[4] Melanie Swan, "Connected Car: Quantified Self becomes Quantified Car," Journal of Sensor and Actuator Networks, Vol. 4, Issue 1, pp. 2-29, Apr. 2015.
[5] O'Donnell, Jayne, "Disconnect in the distracted-driving blame game," USA Today, July, 2012.
[6] Connected Car, http://www.autoconnectedcar.com /definition-of-connected-car-what-is-the-connected -car-defined/
[7] QNX Car 2, http://www.qnx.com/content/qnx/en/ products/qnxcar/index.html
[8] MS Windows Automotive 7, https://www.microsoft.com/ windowsembedded/en-us/windows-embedded-automo tive-7.aspx
[9] MontaVista ATP, http://www.mvista.com/press_ release_detail1.php?fid=news/2011/MontaVista_ATP_ Declared_GENIVI_Compliant.html&d=530
[10] Tizen IVI, https://wiki.tizen.org/IVI
[11] Wind River IVI, https://www.windriver.com/news /press/pr.html?ID=12843
[12] Ren Yu, Wan Jian, "Software Simulator of Embedded Application System," Computer Application, Vol. 11, No. 7, Jul. 2004, pp. 144-146
[13] Ren Yu, "Threads Communication Performance of Embedded Simulator," Computer Application, Vol. 25, No. 25, pp. 12-14, Dec. 2005.
[14] A. Zahir and P. Palmieri, "OSEK/VDX - operating systems for automotive applications," IEE Seminar OSEK/VDX Open Systems in Automotive Networks, Nov. 1998.
[15] AUTOSAR, http://www.autosar.org/standards/
[16] Jihoon Park, Sibok Yu, Dongwhan Jung, Chulbeom Jo, Wonjae Lee, "Design and Evaluation of a VSG based on ISO 13185 standard for Connected CAR Applications," The Korean Society of Automotive Engineers 2016 Spring Conference, pp. 1130, May, 2016.
[17] Jihoon Park,, Youngil Kim, Jongsuk Lim, Youngmin Joo, "Quantitative Evaluation Model for Automotive E/E Architecture,", The Korean Society of Automotive Engineers 2016 Spring Conference, pp. 1185, May, 2016.
[18] Tizen IVI Architecture, https://events.linuxfoundation. org/images/stories/pdf/lcjp2012_mitsue.pdf
[19] Jong-Suk Choi, Jae Won Bang, Hwan Heo, and Kang Ryoung Park, "Evaluation of Fear Using Nonintrusive Measurement of Multimodal Sensors," MDPI, Sensors, Vol. 15, No. 7, pp. 17507-17533, Jul. 2015.
[20] Harman Ignite Platform, https://services.harman.com/ industries/automotive-connected-car/harman-ignite- platform

Authors



Joongjin Kook received the B.S., M.S. degrees in Computer Science and Engineering from Kwangwoon University, Korea, in 2005 and 2007, respectively, and received Ph.D. degrees in the School of Computer from Soongsil University, Korea, in 2012. Dr. Kook joined the VR/AR Research Center at KETI, Seoul, Korea, in 2005. He is currently a Professor in the Department of Information Security Engineering, Sangmyung University. He is interested in Embedded Systems, Operating Systems, IoT Platforms and IoT security.