The Design, Implementation, Demonstration of the Architecture, Service Framework, and Applications for a Connected Car

Joongjin Kook*
Deptartment of Information Security Engineering, Sangmyung University
Cheonan, South Korea
[e-mail: kook@smu.ac.kr]
*Corresponding author: Joongjin Kook

Received December 7, 2020; revised January 18, 2021; accepted January 27, 2021; published February 28, 2021

Abstract

While the conventional vehicle's Head-Units played relatively simple roles (e.g., control of heating ventilation and air conditioning, the radio reception), they have been evolving into vehicle-driver interface with the advent of the concept of Connected Car on top of a rapid development of ICT technology. The Head-Unit is now successfully extended as an IVI (In Vehicle Infotainment) that can operate various functions on multimedia, navigation, information with regards to vehicle's parts (e.g. air pressure, oil gauge, etc.). In this paper, we propose a platform architecture for IVI devices required to achieve the goal as a connected car. Connected car platform (CoCaP) consists of vehicle selective gateway (VSG) for receiving and controlling data from major components of a vehicle, application framework including native and web APIs required to request VSG functionality from outside, and service framework for driver assistance. CoCaP is implemented using Tizen IVI and Android on hardware platforms manufactured for IVI such as Nexcom's VTC1010 and Freescale's i.MX6q/dl, respectively. For more practical verification, CoCaP platform was applied to an real-world finished vehicle. And it was confirmed the vehicle's main components could be controlled using various devices. In addition, by deriving several services for driver assistance and developing them based on CoCaP, this platform is expected to be available in various ways in connected car and ITS environments.

Keywords: Connected Car, In-Vehicle Communication, In-Vehicle-Infotainment

1. Introduction

Vehicles have rapidly evolved; in the past, they were usually seen as mechanical systems. However, they are now becoming electronic systems on the basis of emerging ICT researches including Internet, Communications, Big data and Convergence technologies. Referred as IT based Automobile technologies, it is broadly divided into two major domains: the one is an autonomous driving as advances in a basic role of the vehicles [1], and the other is a Connected Car as advances in user experience (UX) in the vehicles [2, 3]. The latter have been paid much attention as the life style requires not only more frequent use of vehicles but also more time to spend in the vehicles.

The concept of Connected Car is a convergence of state-of-the-art ICT technologies in order to improve both the level of comfort and the quality of driving experience inside the vehicles. To this end, it includes multimedia services including Audio/Video, navigation, the vehicle control based on the voice recognition, Diagnosis Efficiency and payment service, etc. The following shows the basic functionalities of Connected Car fundamentally required [4].

- capable of accessing the Internet at anytime, using either a built-in device or brought in user devices
- equipped with a set of modern applications and dynamic contextual functionalities, offering advanced infotainment features to the driver and passengers
- capable of interacting with other smart devices on the road or in mechanical shops, leveraging vehicle-to-road infrastructure communication technologies
- capable of interacting with other vehicles, leveraging vehicle-to-vehicle communication technologies

Connected Car is embodied in the form of In-Vehicle-Infotainment (IVI) which acts as a control interface between diverse functions of vehicle and the driver. Software (SW) venders such as QNX Car 2, MS Windows Embedded Automotive 7, Wind River IVI, and Samsung Tizen IVI have led the SW platforms as a means to implement IVI devices, and GENIVI Alliance has led the standardization of such SWs. In addition to this, the display link in automotive IVI devices has appeared to favor the smartphones' UX/UI similar to DLNA, WiDi, Miracast, AirPlay, and Chromecast [5] with the development of network and mobile technologies. The Connected Car Consortium (CCC) adopted MirrorLink as the global standard for this being integrated onto Connected Car as a mandatory. Google and Apple, two leading IT giants use Android Auto and CarPlay to wirelessly transfer smartphone's UIs to IVI devices, respectively.

In line with such progress, we here propose *CoCaP*, a software platform architecture particularly designed to implement these essential functionalities of *Connected Car*. First, we implemented them on Tizen IVI and Android in order to examine its efficacy, and deployed it in the real vehicles afterwards. As a platform, *CoCaP* provides broadly three categories of open-frameworks in the pursuit of (i) Smart Driving, (ii) Smart Care and Self Diagnostics of vehicles, and (iii) Mood and Entertainment services.

For the smart driving, VSG is developed for the vehicle control based on *CoCaP*. Coupled with this, a vehicle control framework is developed both to monitor vehicles' status and to control the vehicle towards the desirable states by connecting VSG and IVI devices. IVI's vehicle control APIs are available for both Native Application and Web Application on top of Tizen IVI-based Native App type and Android-based Web App type. These VSG APIs can be accessed from external environments including mobile devices by linking to clouds. Like

Google's Android Auto and Apple's CarPlay, *CoCaP*, provides a display link framework using MirrorLink that allows the driver's smartphone applications to use the IVI device.

In addition, *CoCaP* includes service frameworks for Smart Care/Self diagnostics, Mood & Entertainment services, and Runtime libraries and APIs needed for amalgamating fundamental functionalities such as (i) an emotional reasoning framework for preventing drivers from drowsiness so as to provide pleasant driving condition, (ii) a camera-based face/head tracking framework, (iii) a cloud framework to extend the functions of IVI devices to various mobile devices, and (iv) a display link framework for screen linking of mobile and IVI devices, etc. These are implemented on a basis of both a native library for supporting native applications and javascript-based library for supporting web applications. This allows to minimize the dependency on the platform and maximize the convenience of developers at the same time.
Given the frameworks in *CoCaP*, we present three representative examples of Connected Car services. Smart Care service based on the physiological signal sensing device and the emotional reasoning framework is firstly presented. Next, the drowsy driving prevention service based on the camera and the image processing is described. Afterwards, MirrorLink-based vehicle control framework and service are presented.

This paper is organized as follows. After the introduction in Section 1, brief summary of related works is presented in Section 2. It is followed by the details on *CoCaP* platform following the order of smart driving, smart care and self diagnostics, mood and entertainment services, and examples utilizing the platform features in section 3, 4 and 5, respectively. Finally, we conclude the paper with discussions and future works in section 6.

2. Backgrounds

2.1 Connected Car

The car is no longer a mechanical means for the general transportation, but has evolved into an electronic product controlled by an Electronic Control Unit (ECU). Original electronic equipment for vehicles consisted of a small number of ECUs, each of which is totally uncoupled from the others and is dedicated to individual functions. Due to the rapid progress on ICT technologies, exchange of data between interconnected ECUs have become possible so that enables the development of highly sophisticated communication architectures between them [6].

Within this context, the number of embedded systems in automotive domain is steadily growing for body electronics, infotainment, and telematics applications [7, 8]. A modern vehicle is equipped with more than 200 sensors. They are used to measure several readings including distance, position, acceleration, vibration, angular velocity, pressure, flow, light and temperature. For example, a high-end Lexus manufactured by Toyota contains 67 microprocessors, and even the world's cheapest car, Tata Nano, has a dozen of them as well. Voice-driven satellite navigation is typically used by millions of people in worldwide. Radar-equipped cruise control allows vehicles to adjust their speed automatically in traffic. Some vehicles can even perform the car park by themselves.

Connected Car is representative form of this transformation. For example, it links to satellites navigation and communications networks, and even further directly to other vehicles. Thanks to this, it could transform driving, preventing motorists from getting lost, stuck in traffic or involved in accidents. And connectivity can improve entertainment and productivity for both driver and passengers [9, 10]. Connected car in effect means that vehicles are now part of the connected world, continuously connected on the internet, generating and

transmitting data, which on the one hand can be helpfully integrated into applications, like real-time traffic alerts broadcast to smartwatches, but also raises security and privacy concerns [11, 12].

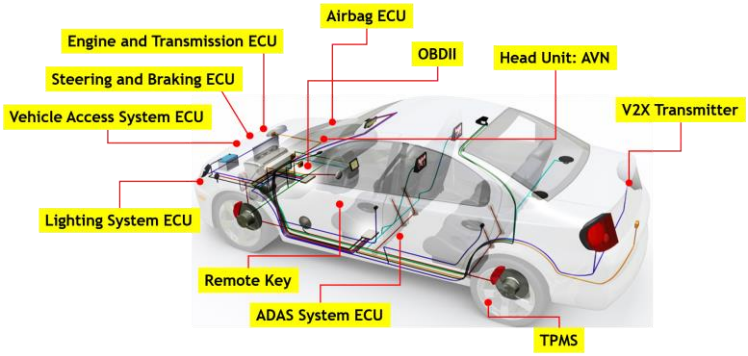

Fig. 1. Some examples of ECUs installed on today's cars

As cars turn into computers on wheels, it becomes vital to do experiment with novel application scenarios and to envision suitable abstractions for integrating cars into existing enterprise information systems. This insightful project report tells the story of a prototyping platform for connected car software and shares the project's experience with a cloud computing pattern language that helped drive the architectural platform design [13].

2.2 In-Vehicle-Infotainment Device

The vehicle has sensors and actuators for each function of each part, and ECU made up of Micro Controller Units (MCU) is composed for the data processing of sensors and the control of actuators. ECUs usually consist of Powertrain, Braking, Chassis, and Steering module and so on constituting the vehicle. ECUs are also composed of various networks such as CAN, LIN, MOST and FlexRay for the data exchange among them. In this context, the main control system can be seen as a huge network between ECUs.

OSEK [14, 15] and AUTOSAR [16, 17] have been introduced to control these ECU-based control systems in a standardized manner. AUTOSAR is an automotive software specification and an execution environment designed for automobile companies to share, perform abstracting MCU and ECU, and provide RTE (Application Runtime Environment) based on Communication, Network, I/O, OS Service frameworks. AUTOSAR is built by SW Component (SW-C), Run-Time Environment (RTE) and Basic Software (BSW). It allows application services independent from HW to develop through adopting the concept of RTE and separating application-level SW-C from BSW, HW-related SW. SW-C implements parts of the functions of application SW and exchanges mutual data via port and interface as the basic unit mapped to ECU.

It is necessary to receive the support of ECU Datasheet from the manufacturers in order to collect data from each sensor constituting the vehicle and to perform the control via the ECU. This development method requires the cooperation with the automobile manufacturers, and there is a limitation to applying directly SW to ECU of the real vehicle. To use these functions of Connected Car so far, it is necessary to purchase a finished car already equipped with these functions or to replace it with an IVI device supporting Android Auto and CarPlay.

MS's Embedded Windows Automotive (EWA) 7, one of the leading platforms for IVI devices. For MS EWA, parts subordinated to a CPU architecture are composed of third-party component of Hardware Layer, and parts needed manufacturer-level support are represented

as Tier 1 component.

Connected Car can be defined as a new form of vehicles that combines ICT technology with automobiles. The most pivotal device serving as an interface between the car and the driver in Connected Car is the IVI device. In the past, vehicles provided the HAVC control and the radio reception functions by placing Head-Units at the centerfascia and operated them by placing physical control units such as buttons. The Head-Units have gradually evolved into providing multimedia based functions, which become a high-end computer system supporting various networks such as Bluetooth, WiFi and LTE by now. Therefore, they have provide wider services, and many SW platforms for IVI have appeared to provide the vehicle control and the driver's convenience functions in the form of applications. Typically, there are QNX Car 2, MS Windows Embedded Automotive 7, Wind River IVI, and Samsung Tizen IVI, etc. These OSs are customized by each model of automobile manufactures and are applied to finished vehicles.

2.3 Display Link

With the spread of mobile devices starting with smartphones and tablets, various technologies have appeared to project contents of them into other devices by copying the screens of mobile devices. DLNA, WiDi, Miracast, Chomrecast and AirPlay are good examples. They transfer contents on displays of laptop PCs, tablets, and smartphones to the displays of TV or other PCs.

The representative display-link technologies for Connected Car are MirrorLink, Google's Android Auto, and Apple's CarPlay. As IVI devices begin to support the wireless connectivity (e.g., WiFi, BT) and the ports like USB for the connection to peripherals, Google and Apple have launched Android Auto and CarPlay, respectively. They allow the smartphones' UX to use in cars equally, and recently released vehicles have supported either or both products.

The Car Connectivity Consortium, made up of various automobile and electronic manufacturers, has joined together to establish an industry standard for certifying applications, services and devices which are both safe and useful for drivers, called MirrorLink. The joint effort by car manufacturers and mobile phone makers is aimed at developing public standards to define operations of smartphones linked to cars. MirrorLink is an interoperability standard on mobile devices that offers integration between a smartphone and car's infotainment system. MirrorLink amalgamates smartphones with automotive application platforms where apps are hosted and run on the smartphone while drivers and passengers interact with them through the steering wheel controls, dashboard buttons and touch screens of their car's In-Vehicle Infotainment (IVI) system [18]. MirrorLink utilizes a set of well-established, non-proprietary technologies such as IP, USB, Wi-Fi, Bluetooth, Real-Time Protocol (RTP, for audio) and Universal Plug and Play (UPnP). In addition, MirrorLink uses Virtual Network Computing (VNC) as the baseline protocol to display the user interface of the smartphone applications on the infotainment system screens and to communicate user input back to the mobile device. *CoCaP* enables mobile apps to use as they are in IVI via MirrorLink instead of directly supporting Android Auto or CarPlay. In addition, a vehicle control framework based on MirrorLink-VSG is designed in order to control the vehicle via the MirrorLink-based applications.

2.4 ICT Convergence Services for a Car

Intelligent vehicles, and all their services pertaining to security, efficiency, economic and environmental impact, and transportation comfort, are part of what is called an Intelligent Transport System (ITS) [19]. An ITS comprehends not only the vehicles but also pieces of the

road infrastructure (like traffic signs and toll collection machines), pedestrians, and so on. A set of different communication means can be used to make these elements interact with each other. **Fig. 2** gives a glimpse of the very diverse set of elements with which interact [4].

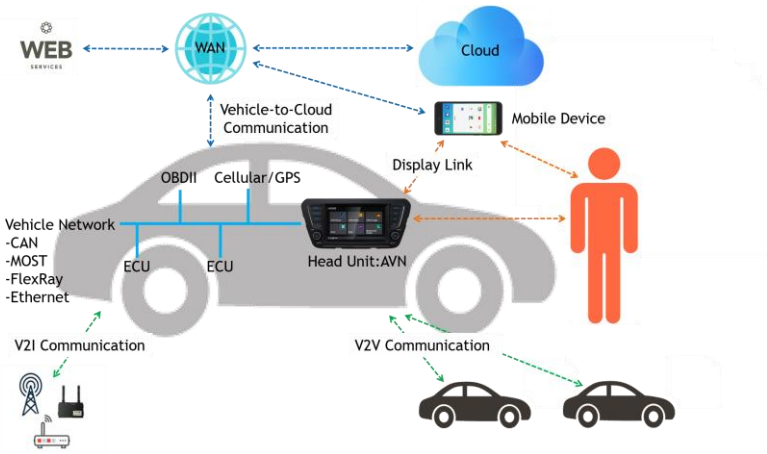

Fig. 2. Overview of the connected car system

The types of ICT convergence services in the domain of Connected Car are various. Among them, the service for the driver's safety is considered with the most care. In this paper, several sensors and their associated functions are constituted for the various services with regards to the driver's safety including accident prevention, driver's fatigue sensing, real-time assistance for parking and accident, and anger management and stress reduction [20, 21].

The British Royal Society for the Prevention of Accidents estimates that driver fatigue may be a contributing factor in up to 20% of road accidents and up to a quarter of fatal and serious accidents [22]. To inspect the driver's fatigue and to alarm the time of the rest in due course, Electrocardiography (ECG) sensors are inserted under the seat. In line with this, products are launched to diagnose the driver's condition and provide services by measuring the driver's heart rates [23].

Emotion recognition has been researched in many fields, such as robotic systems and advanced driver assistance systems (ADASs) [24]. MIT's Media Lab goes even farther in this, with "empathic vehicles" project called *AutoEmotive*. Here, sensors attempt to detect a range of driver's emotions, and predictively manage the onset of his/her anger and stress [25]. This research implies that emotion-sensing might be helpful in reducing road rage incidents [26]. Methods to measure the emotion are visible-light cameras, thermal cameras, voice data or physiological signals, such as from electrocardiography (ECG), electroencephalograms (EEGs) or skin temperature (SKT) data [27]. Proposed emotional reasoning engine based on mobile user situation is on the basis of a context-aware and multimodal emotional reasoning model taking into account both the internal physiological variables (SKT, GSR, PPG, Movement) and the external environment variables (Temperature, Humidity, illumination) [28, 29].

Another major cause of traffic accident is driving in drowsiness and the distracted driving [30]. The driver's physiological signals such as PPG, GSR and SKT are good indicators to evaluate the driver's emotions. In addition, a camera-based framework to judge the drowsy driving can also contribute to this, which has been studied in various ways such as face recognition, eye tracking, head position tracking and blinking. Denso's Driver Status Monitor has a function to extract 17 features of the face by taking a picture of the driver with the camera installed at the center of the steering wheel. The driver is warned when drowsiness or looking

elsewhere while driving occurs by sensing the direction of the face or the open state of the eyes. The signs of drowsiness such as the degree of heavily lidded-eyes, the number of blinking, the irregular movements of the eyes can be figured out in various ways. The levels of drowsiness are divided into six levels, and the warning levels change according to the levels of drowsiness. If the level of drowsiness is determined as strong, the collision avoidance device operates faster and strongly. In the case of a light drowsy stage, the air conditioner turns on strongly [31].

The device of AISIN SEIKI detects the driver's breath, pulse and so on by installing a camera module on the steering wheel and by building a pressure sensor and a vibration device into the seat. It is impossible to distinguish whether the driver is sleeping or fainting with only the camera. Thus, the comprehensive judgment with a camera and a pressure sensor makes the distinction between drowsiness and fainting possible. If a fainting condition is detected, an emergency notification can be automatically sent to the fire department. In the case of light drowsiness, the voice guidance only operates. However, when an emergency such as deep drowsy driving or fainting is detected, the driver is awaken by the voice guidance or the vibrating seat before an accident [32].

Trywin's dormancy prevention device, *Dramoni* is sold at 44,000 yen and solves many problems of the existing camera systems and various sensors such as complicated installation, high price and limited driver's behaviors. *Dramoni* categorizes the drowsy driving into seven levels by detecting the driving motion from the motion of the driver's back and backbone. When driving, the driver always performs basic operations such as adjusting the steering wheel, stepping on the accelerator and checking the mirror. *Dramoni* pays attention to the fact that the basic operations are not properly performed when concentration is reduced [33]. In Germany, Bosch has commercialized a device that calculates the driver's fatigue via the manipulation of the steering wheel. Based on that the sudden manipulation of the steering wheel tends to increase on the lack of concentration, the icon shaped like a coffee cup is blinking when it exceeds a certain level, informing that rest is necessary. This product was adopted for Volkswagen's luxury sedan, Passat in 2010 [34].

3. CoCaP Architecture

3.1 Vehicle Selective Gateway

As For the data collection of the vehicle, IVI applications must be developed based on SDK provided by SW platforms such as QNX Car, Windows Embedded Automotive, Tizen IVI, Android, Automotive Grade Linux and GENIVI to control the vehicle through IVI devices. In addition, applications and devices may be required to comply the standard data collection/transfer interface like the OBD2. In this case, dependent applications must be developed depending on the IVI platforms, vehicle manufacturers and models, otherwise compatibility cannot be guaranteed. In *CoCaP* architecture, VSG is developed to ensure the independence of the IVI platform for the vehicle data collection and the vehicle control, and VSG APIs are designed to abstract the functions of the vehicle data collection and the vehicle control. Therefore, VSG APIs allows you to call and to control the functions independent of the vehicle. The IVI devices can be used independently on the OS by providing VSG APIs as a web application library based on the Web Framework.

While the automobile manufactures can develop the AUTOSAR-based application SW mapped to the ECU and apply it directly to the vehicle, it is almost impossible for the third-parties to develop this type of vehicle SW. Therefore, VSG is developed for the indirect control

of the IVI-based ECU and the collection of the vehicle data. Then the control via IVI can be possible by connecting VSG and IVI. **Fig. 3** shows the interface among the vehicle's ECU network, the VSG, and the IVI device.

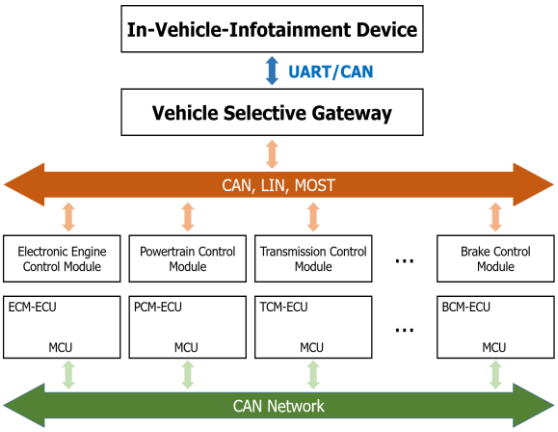

Fig. 3. ECU, VSG, and IVI Inter-connection

Body Control Module (BCM) built on the vehicle, through which almost functions and operating structures such as switches connected to the vehicle's electric field can be understood. Basically, the BCM is used to directly control all functions related to the body of the vehicle (door open/close, window open/close, sunroof, HAVC On/Off, head-beam On/Off, smart key, etc.). VSG's APIs basically include functions related to BCM. ISO 13185 standard led by ICT industry provides standard interfaces and APIs for securely collecting vehicle information and provides VSG's design guidelines as shown in **Fig. 4**.

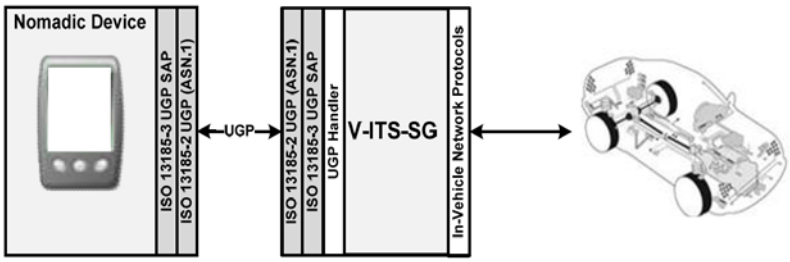

Fig. 4. V-ITS-SG and Vehicle Component Interface

Since there is no case that the ICT industry-led standardization is attempted in the domestic automobile industry, there is a limitation that the practical impact of the service is relatively low and the vehicle-oriented services such as safety service and remote control cannot be provided. Inside the vehicle, 37.5KBYTE (@ 300Kbps) of information is generated. In case of simply collecting this information, 162TB (100,000 vehicles / 12 hours) of huge data will be produced. If it is used in the connected car service, processing using data filtering is required. Therefore, as for the standard API, the basic interface is defined based on ISO 13185 standard, and APIs that should be considered from the viewpoint of the car industry are additionally developed such as clustering function for combining vehicle information properly and screening function for blocking external intrusion. **Fig. 5** shows the VSG's block diagram for the interface between the vehicle and the VSG, and the connection among the VSG, the IVI device, and the service.

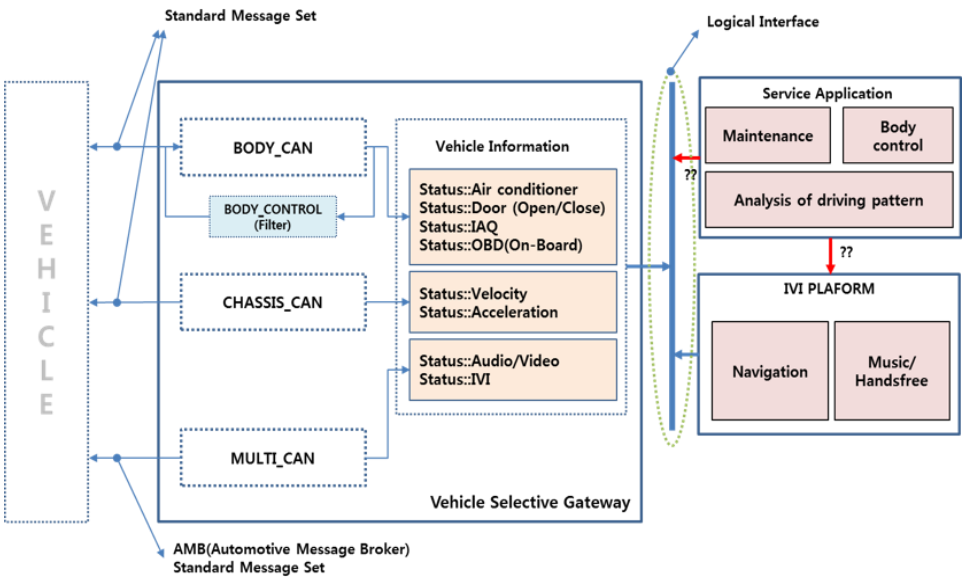

Fig. 5. Vehicle Selective Gateway Block Diagram

In order to implement the function of VSG for the vehicle control, it is necessary to understand the structure of ECU which can be accessed and controlled for each model of the car manufacturers. In the tests, the information on Hyundai Motors' Sonata and SsangYong Motors' Tivoli models are offered from the manufacturers thus applied our proposal to the commercial manufactured vehicles.

VECTOR's CANoE, which is widely used in the industry for the vehicle network design, is used as the evaluation device. The dynamic information of the vehicle that should be collected in real time in the clustered message is classified separately and 8 information (tire status, gear status, airbag status, amount of fuel, engine RPM, longitudinal acceleration, velocity, vehicle ID) is included. Time required to process 8 clustered vehicle information in the VSG. The average processing time of collected data is 100.003 ms.

VSG verifies the vehicle controllability by getting E/E information about some models of *Hyundai* and *Ssangyong* included in our consortium and by applying it to commercial vehicles. Based on this, APIs provided in the level of VSG are defined. **Table 1** lists the vehicle control APIs provided by *CoCaP*.

Table 1. VSG APIs

API Function	Description
setStartVehicle()	Vehicle Startup On
setStopVehicle()	Vehicle Startup Off
setHighbeam()	Headbeam On/Off
setHorn()	Horn On/Off
setSunroof()	Sunroof Open/Close
setOpenWindow()	Window Open/Close
setDriverDoorOpen( )	Driver’s door open/close
setAssistDoorOpen( )	Assist’s door open/close
settEmergencyLight()	Emergency light on/off
setIncarTemperature()	Vehicle temperature setting
setVSGmsgStart()	VSG message transmit start
setVSGmsgStop()	VSG message transmit stop

getStartVehicleStatus( )	Start Status of vehicle
getDriverDoorOpen( )	Driverdoor open/close
getAssistDoorOpen( )	Assistdoor open/close
getFuelLevel()	Fuel level
getEngineRPM( )	Engine RPM
getTotalDistance( )	Total distance
getDriveSpeed( )	Drive Speed
getAccCount( )	The number of rapid acceleration
getRedCount( )	The number of rapid deceleration
getPBreak( )	Parking Gear Status
getAirbagStatus( )	Airbag deployment Status
getIncarTemp()	Vehicle temperature
getCheckEngine( )	Engine abnormal status
getVehicleID( )	Vehicle ID
getSideBrakeStatus( )	Parking Brake status

In order to develop the VSG that serves as the interface with the IVI device and the vehicle through the CAN network, NXP's MPC5566 of 32-bit MCU and FreeScale's MC9S08DZ60 of 8-bit microcontrollers are used.

The VSG (Vehicle Selective Gateway), which contains the vehicle information, should communicate with the IVI device based on a common format, and therefore the UGP (Unified Gateway Protocol) message format defined in ISO 13185-2 is applied. As for the UGP message format, data structures such as '*dataParamList*' and '*dataParamMapping*' are defined as data parameters to respond to the request. According to ISO 13185-2 standard, these data parameters should be defined in the VSG depending on the manufacturers.

Two types of IVI platforms are adopted to apply *CoCaP* in IVI devices. In the first stage, Tizen IVI is ported onto Nexcom's VTC 1010 model to build the system. In the second stage, Android 4.4 (KitKat) is ported to on Freescale's i.MX6DL to implement the system. Tizen IVI in the first stage provides native framework and web framework to support both native app and web app, respectively. However, since Android in the second stage does not support framework for web app development, web framework is implemented by applying Crosswalk. Web framework in *CoCaP* enables VSG-based vehicle control SW to be implemented in both native app and web app. The methods of VSG API implementation depend on the types of SW platforms of IVI.

First, Tizen IVI-based *CoCaP* connects VSG and IVI to UART. It implements Native library in IVI for UART communication and transmits commands to ECU. Web app interface wraps it with javascript functions in order to be equally used in web-app. In the Android-based *CoCaP*, VSG library for native app development is implemented as the format of service component included in application framework layer of Android architecture. Therefore, Android native application can be developed as an application including the vehicle control by including VSL APIs-related class library in the project. In addition, web app can easily call them by using Crosswalk's javascript interface in order to provide the web app interface. How to build a platform-specific library and how to develop such a library-based application are described in the next section.

3.2 CoCaP Architecture

CoCaP is the platform SW for connected car integrating (i) the IVI core for monitoring and controlling the vehicle condition, (ii) the native core and web framework for developing and running applications for IVI devices, and (iii) the network framework and device framework for supporting various convergence services. And it supports clouds to ensure interoperability.

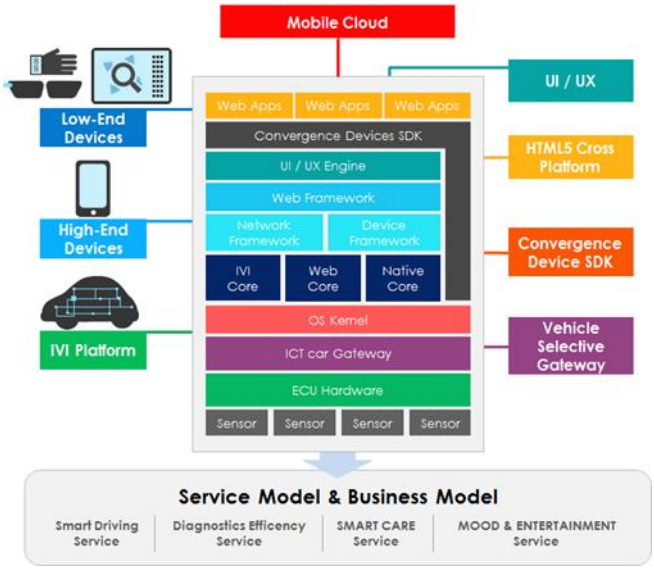

Fig. 6. *CoCaP*: Overall Architecture and Service Models

Fig. 6 shows the SW Architecture of the *CoCaP* platform. In the lowest layer, ECUs managing sensors and actuators associated with various functions inside the vehicle are connected to the VSG and the CAN, which serve as a gateway between the vehicle and the IVI device. The VSG is connected to the IVI again to serve as an interface between the vehicle and the driver.

IVI core, *Web core*, and *Native core* components in the middle layer are essential components for developing applications for IVI devices. IVI core includes the vehicle control framework including VSG APIs. Native core includes a framework supported by SW platform for IVI, and SDK. *Web core* includes a web browser engine.

Network Framework and *Device Framework* on top of component layer are included for the connectivity between devices. *Convergence Devices SDK* is provided to enable web application and native application for IVI devices to be developed. It provides the driver with the vehicle status information through IVI devices or mobile devices such as smart phones, tablets, and smart watches, and responds to the driver's requests, and interoperates with other external services by linking to clouds.

The *CoCaP* platform provides web runtime to support web application as well as Tizen IVI and Android native application. Web runtime in *CoCaP* includes web app core framework and web app extension core framework to provide various services based on web application. It also provides native interface for the connectivity to IVI devices and mobile devices, and VSG-based native interface for the vehicle control, and provides cloud interface for the connectivity to cloud and web services.

Web runtime is a collection of applications for IVI devices, VSG related to the vehicle information, external devices such as cameras and bio-signal sensors, and components for various web services. The web app framework consists of a WebApp core consisting of a library provided by the Web RunTime Engine (WRT) and a WebApp extension core, which is a library for processing the necessary extensions (e.g., bio-signals, cameras, application of user profiles, etc.). The Web app framework provides the access to the resource for mobile devices such as VSGs, IVIs, and smartphones through the native interface, enabling the execution of service applications on IVI devices or mobile devices via the cloud interface. At

this time, each component of WRT provides APIs by binding data accessed from WRT through Native interface and cloud interface into javascript in order to be used in web application. For the application and the experiment of *CoCaP*, IVI devices supporting Tizen IVI and Android are sorted out. Since both Tizen IVI and Android are open sources based on Linux kernel, they are considered to easily add or change the necessary functions. The possibility of supporting web framework or porting web framework is additionally considered.

3.2.1 Tizen IVI-based CoCaP Platform

Nexcom's VTC 1010, which supports Tizen IVI [35] is used as an IVI device to connect with VSG for the implementation of *CoCaP*. Nexcom's VTC 1010 supports Tizen IVI 2.0, one of the IVI SW platforms, and Tizen IVI includes both a native framework for Native Application and a web framework for Web Application.

UART channels between IVI and VSG are composed to test the vehicle control based on VSG APIs through IVI application. The UART library in Tizen IVI is implemented for UART-based communication between IVI device and VSG, through which VSG' APIs can be called in the application of IVI device. The UI and service of IVI device application are implemented as Web Application based on Crosswalk. The UART library to call VSG's vehicle control API defines the byte array corresponding to VSG's command packet format as **Table 2**.

Table 2. ECU Command Packets

ECU Command	Packet Format & Value							
	Type	Content	Position	Val[0]	Val[1]	Val[2]	Val[3]	Val[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

Table 2 enumerates data packet per ECU command defined by the analysis of the BCM and E / E component structure of the vehicle model, *Hyundai Motor's* SONATA, that we used for the experiment. In the Tizen IVI-based native application, VSG APIs can be called using the UART library for the ECU command in **Table 2**. In order to support the web application, they can be used in IVI web application by wrapping the ECU command with the API functions in **Table 1** and by building into UART library. **Table 3** shows how to use VSG APIs in the Node.js application via the Web-based UART library.

Table 3. VSG APIs & Node.js Example

Function	Usage for Node.js
Load Serial Package	var vsg = require('./build/Release/serial');
getDriverSeatBeltStatus	vsg.getDriverSeatBeltStatus();
getStartVehicleStatus	vsg.getStartVehicleStatus();
setStartVehicle	vsg.setStartVehicle();

We note that the reason to use Node.js in the VSG control application is to make the API call of the VSG easier in the mobile device and the cloud environment. In addition, to add a server component to IVI application of VSG APIs by using WebSocket or Socket.IO makes it possible to remotely control the vehicle in other mobile devices such as smartphones as well as IVI.

3.2.2 Android-based CoCaP Platform

Freescale's i.MX6 platform [36] is specialized for IVI devices, equipped with all the essential elements to develop IVI application SW. It supports FlexCAN as an interface for the connection with the vehicle. It also provides UART and USB interface for connecting to peripherals, and provides basic network interfaces such as WiFi, Bluetooth, and Ethernet.

As seen, Android 4.4 is ported to the i.MX6-based IVI device, and the development of APIs that can be used in Android-based Native Application or Web-App is required for the vehicle control by communicating with the VSG. The VSG here is customized for developing Android Native Application in the form of Service of Android Framework layer. To do so using these VSG APIs, it is implemented as a class library of jar format by binding related classes. The binding method of Android Service providing VSG APIs is as follows. We note that VSG's Service Component is named as '*flexcan*', and Android-based native application can call VSG's APIs through the interface of flexcan service component in this paper.

The configuration of the VSG APIs is seen in **Table 1**, and the calling method is specified by each function provided by the interface of the *FlexcanService* component.

The *CoCaP* provides Web Runtime to support web app. We use Crosswalk to develop VSG interface on web runtime, which makes the call of VSG APIs possible through javascript.

4. Convergence Services for Connected Cars

4.1 Emotion Sensing

When driving a vehicle, it is common that an accident occurs due to the drowsy condition caused by several factors such as long time driving, accumulated fatigue, or environmental situations [37]. The driver's drowsiness is recognized by tracking of driver's face, eyes and head via the images processing techniques from the images captured by camera sensors and by extracting them. Or it could alternatively done by the evaluation of driver's condition using their physiological signals. In *CoCaP* platform, a device for the evaluation of emotional state in conjunction with an emotional reasoning framework are applied to this end through face tracking and head tracking based on camera and image processing.

Fig. 7 shows the sensing device for collecting driver's physiological signals using PPG, GSR and STK. We installed these sensors on the steering wheel and armrests as a means to collect the data of fingers, wrists and forearms. These physiological data fuel the emotion reasoning framework shown in **Fig. 8** that performs the inference on the emotional states of the driver in due course. External factors such as temperature, humidity and illumination can also affect to the emotional states of the driver. The Temperature Humidity Index (THI) is therefore considered to evaluate the human's emotional state more accurately coupled with physiological data.

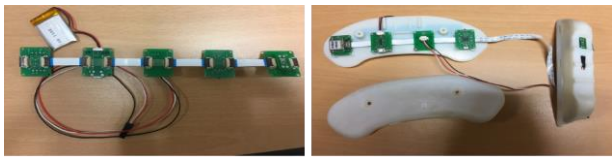

Fig. 7. Emotion Sensing Device

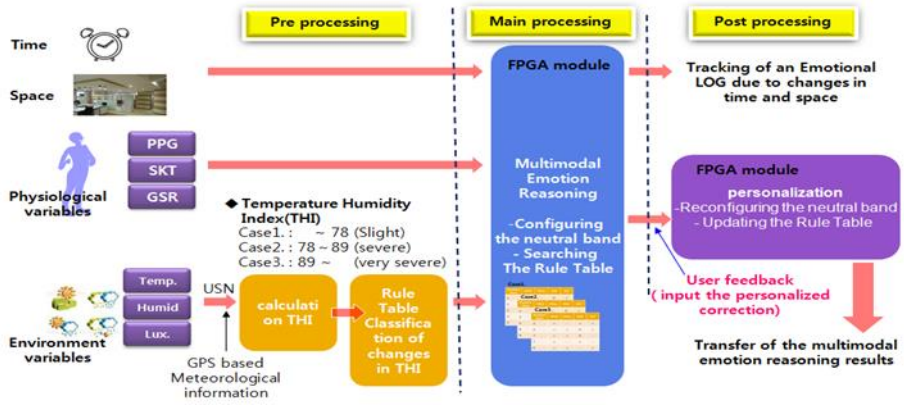

Fig. 8. Emotion Recognition & Reasoning Framework

The results of the emotional reasoning can be expressed in a degree of driver's arousal status, relaxation, neutrality, comfort, or discomfort. A service to induce a state of awakening from a relaxed state or that to induce a state of comfort or neutrality from an uncomfortable state are provided depending on the reasoning results. These could be used to prevent drivers from the drowsy driving, one of the most typical causes of traffic accidents. Moreover, they can be used to figure out an emergency situation such as the driver's anger control or cardiac arrest.

Based on the physiological signals collected through the Emotion Sensing device, the emotional state of the driver can be inferred as one out of three states: Normal, Stress and Sleepy. The IVI can run the application to relieve the driver's Stress or feeling of Sleepy depending on the inference outcomes.

In **Fig. 8**, the FPGA is used to improve the computation speed for emotional evaluation with such two types of data. Since the state of human's emotion changes in a very short cycle, however, this does not take place in real-time.

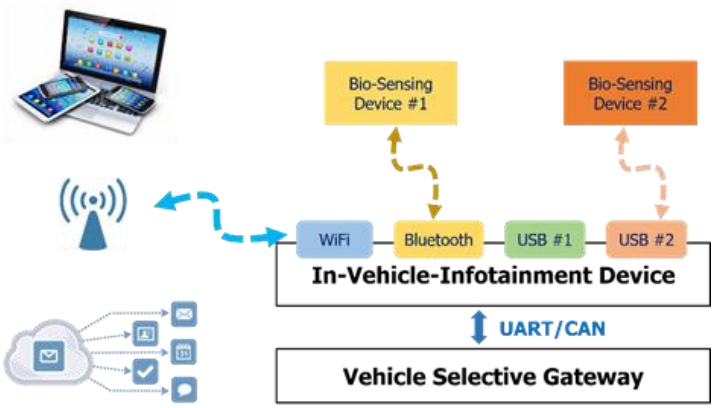

Fig. 9. Emotion Sensing Device and IVI Inter-connection

The emotional reasoning results in the control of the application of IVI to lead the driver's status to being pleasant or awaken by transmitting a stimulus (in the form of tapping) to the seat via the VSG. Sometimes, it alerts the driver through the wearable device as shown in **Fig. 9**. In the emergency situation where the driver is unconscious, a loud emergency call is sent via the driver's smartphone.

4.2 Emotional Service

The *CoCaP* is also equipped with a vision based system composed of a camera device and image processing unit based on OpenCV in order to monitor driver's status. It is particularly designed for the appropriate response to accidental situations caused by drowsy condition. **Fig. 10** shows the Driver Vision Sensing Framework (DVSF) of the *CoCaP* platform to determine the situations so that responds to those appropriately, such as the driver's drowsiness or fainting. DVSF acquires images by connecting a USB-type webcam to the IVI device, and recognizes and traces the driver's face through the OpenCV-based *Haarcascade* algorithm in order to infer the driver's status.

OpenCV is ported to i.MX6 applying to Android, and Driver Vision Sensing Native (DVSN) API is developed to predict normal/abnormal driving posture such as drowsy driving and distracted driving based on face tracking. In this process, the modification of the driver and the framework is required for IVI device to recognize a USB-type webcam.
DVSN API is used to detect and track the face region from the driver's image captured by the USB camera. We assume that a threshold region for Awake status is pre-defined. When departure event (e.g., up, down, left, right) from the threshold region is detected, the result code (NORMAL = 0, ABNORMAL = 1) is sent to the server in the format of WebSocket message; both result and image used for the prediction are sent to the WebRTC server outside the IVI device.

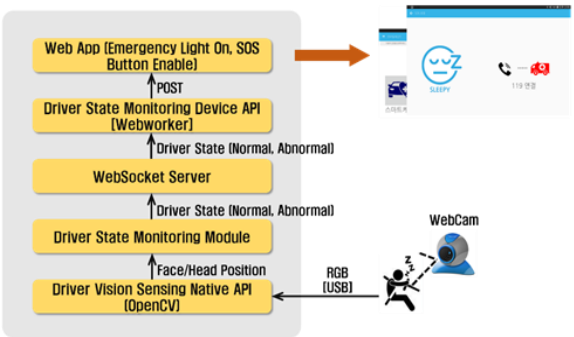

Fig. 10. Emotion Sensing Device

4.3 Display Link

Display Link technology allows the exactly same user experience in all connected devices as that in the smartphone. Google's Android Auto, Apple's CarPlay, and Connected Car Consortium (CCC)'s MirrorLink play the leading role in Connected Car research. The CoCaP supports MirrorLink universally since MirrorLink is the global standard for Connected Car: Android, Windows, and Tizen that is based on mobile Linux.

We believe that this can deliver great benefit to drivers in the context of Connected Car since existing venders only supports the limited platforms: Android Auto is only supported on Android version 5.0 Lollipop and above, and CarPlay is only supported on iOS 7.1 and above. In addition, the vehicle control based on Android Auto and CarPlay requires manufacturer-level supports. This limited support on restricted platform could significantly diminish the user experience in Connected Car domain.

The major IVI platform introduced in Chapter 2 supports MirrorLink as the connectivity for IVI devices and mobile devices. MirrorLink uses USB as an essential interface for the connection between IVI device and mobile device, and additionally supports WiFi and Bluetooth.

MirrorLink transmits UI of the smartphone to the IVI display based on Remote Frame Buffer (RFB) using USB or WiFi P2P in a wired or wireless way. It next enables functions of VC, Call, Messaging, Productivity, Search, Navigation and Music to use. *CoCaP* develops MirrorLink-VSG Device using Raspberry Pi (RPi), an open hardware platform for the direct vehicle control through IVI and MirrorLink applications.
MirrorLink-VSG Device supporting the VSG APIs operated on RPi. First, Serial Command Receiver is executed as background service in IVI device. Next, connection is executed when the MirrorLink application on the smartphone is executed by building a Web Server based on Node.js in MirrorLink-VSG Device. When a connection is done and IVI MirrorLink application calls VSG APIs, a message about corresponding API is transmitted to the RPi server. The MirrorLink-VSG Device is connected to the IVI device via UART. When a Server application receives a VSG API message, it is sent to the IVI Serial Command Receiver after checking the validity of the message.

MirrorLink-VSG-based message definition and data flow required for communication between MirrorLink application and RPi server are shown as **Fig. 11**. We note that Raspberry Pi is used as a device for message transfer between MirrorLink-VSG devices, but it is quite possible to use other devices supporting UART and WiFi.

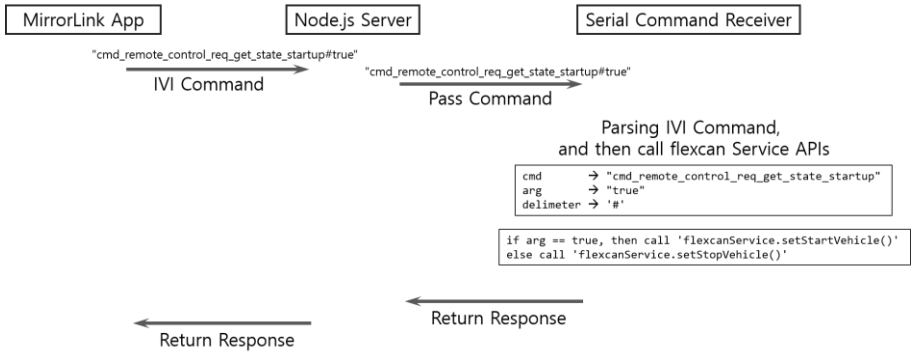

Fig. 11. MirrorLink-VSG Message Format and Data Flow

5. Experiments

To integrate those components in one system, we deploy *CoCaP* in the real environment, a commercially manufactured *'Ssangyong Tivoli'* consisting of: (i) vehicle control through applications based on IVI device and mobile device, (ii) the offer of services based on the result of the driver's emotional reasoning, (iii) the evaluation of the driving posture based on the driver's face tracking, and (iv) the normal operation of the vehicle control based on the MirrorLink application. We performed the test on these by using the *CoCaP* platform implemented based on Tizen IVI and Android platform.

5.1 VSG-based Vehicle Control

The *CoCaP* platform aims to provide a variety of new services for connected car. The most important one is the vehicle control by using IVI devices and mobile devices. Therefore, whether the application of IVI device and application of smartphone can control the vehicle through VSG APIs developed for the vehicle control is tested. The evaluation items and the types of devices for the evaluation are shown in **Table 4**.

By analyzing the BCM and E/E component for several vehicle models, the controllable

main components and network types of the vehicle are come out. We examine whether IVI device is well operated or not by calling VSG APIs through both the native and the web application of the IVI device**.** IVI device, smartphone, and smartwatch were all connected via WiFi. For the performance evaluation, the proper operation and the response time for each function were checked 20 times using IVI app, MirrorLink app, Smartphone App, and SmartWatch app, and then the average response count and time was calculated. Because it was a stable network environment based on WiFi, there was no case of failure of the response, but it can be confirmed that a slight difference in response time depending on the object of the request.

Table 4. Average response time for VSG-based vehicle unit control requests **(**ms**)**

Evaluation items	IVI App	MirrorLink App	SmartPhone	SmartWatch
Vehicle startup On / Off	1.53	2.04	1.83	1.94
Emergency light On / Off	1.23	1.59	1.57	1.76
Door Lock / Unlock	1.73	2.41	2.37	2.65
Window Open / Close	1.15	1.51	1.49	1.92
Horn	1.04	1.98	1.95	2.05
Low beam On / Off	1.27	1.46	1.48	1.97
Sunroof On / Off	1.66	1.92	1.88	2.32
Average Response Time	1.37	1.84	1.80	2.09

5.2 Emotion Sensing

Emergency situations such as the driver's fatigue, anger, and cardiac arrest are one of the major causes of accidents. To respond adequately to emergencies, the driver's bio-signals (PPG, GSR, SKT) are acquired and the driver's emotion is evaluated. Then, one of the Smart Care service models, which send a warning message to the driver via smartphone, wearable devices and controllable windows, emergency lights, a sunroof, a seat, etc., depending on the evaluation results, is implemented and tested. The device for collecting the bio-signals is shown in **Fig. 7**. Here one device in the set of the bio-signal sensing devices is connected to the IVI device via UART, and the other device is composed to be used on the network (including both wired/wireless). APIs for requesting bio-signal measurement and sending and receiving results are also implemented based on UART and based on Bluetooth. **Table 5** shows the evaluation items and the evaluation methods for bio-signal sensing devices.

Table 5. The evaluation methods of the emotional reasoning based on Emotional Reasoning Framework and Sensing Device

Evaluation items	Evaluation methods
Sensing Device Paring(BT)	Check Bluetooth Paring/Connection of IVI device and Sensing Device
Acquisition of physiologic signal	Acquisition of physiological signal via Sensing Device
Data Communication	Request for the emotional reasoning between Bluetooth/USB-type sensing device and IVI/Smart Phone/Smart Watch, MirrorLink App and acquisition its results
Sensing Device Enable/Disable	Enable and Disable, Start and Stop Operation for a Sensing Device
Emotion Reasoning	Results of the emotional reasoning based on the emotional reasoning framework
Service Link	Appropriate application link according to the results of the emotional reasoning

The fact that whether or not USB-type and Bluetooth-type emotional reasoning device are operated properly is tested with the emotional reasoning device and framework. With IVI application, smart phone, smart watch, and MirrorLink application, we verified whether the driver's emotion is classified by comfort/discomfort/ awaken/relaxation/neutrality by using PPG, GSR and SKT data.

5.3 Emotional Services

One of the major causes of accidents while driving is drowsy driving and distracted driving. In the *CoCaP* platform, a camera and image processing are used to predict the driver's conditions, and give the driver a warning message to resolve the cause of the accident. The type of warning messages can be applied in the same way as described in the previous section. The biggest strength of Android-based IVI is that it provides various libraries, and OpenCV, widely used for image processing, is also easily portable. The OpenCV-based *Haarcascade* algorithm is used to detect the driver's face, and the area for the normal driving posture is set, and a warning message is sent to the driver when the posture is out of the normal state for more than a certain time or more than a certain number of times. **Table 6** shows the evaluation items and evaluation methods for face recognition and the driver's condition using camera.

Table 6. The evaluation item of driver’s posture based on Vision Sensing Framework

Evaluation items	Evaluation methods
Camera Enable/Disable	Check Camera Enable/Disable
Background Face Tracking	Face recognition and awake area setting based on Background processing
Driver’s posture determination (Normal/Abnormal)	Determination (Normal/Abnormal) by tracking the driver’s posture
Alert	Vehicle control such as emergency light, window, sunroof, etc., or alert using a smartphone and a smart watch in case of an abnormal posture
Service Link	Link application to induce relaxation or to make an emergency call in case of an abnormal posture

The application for evaluating the emergency situations such as driver's drowsy driving and fainting is executed in the background mode on the IVI device, so other applications such as navigation can be executed on the IVI device at the same time. If the driver is predicted to be in an abnormal posture, a message is shown that ‘awakens the driver’, or propose to relax to the users by using the vehicle's window, sunroof, seat shock, or warning sound. To give an alert is also possible by using a smartphone or a watch through cloud service. In addition, if the driver's posture continues to be in an abnormal state, it is also possible to make an emergency call to remotely check the driver's condition and to take an appropriate rescue.

5.4 Mirror Link Services

CoCaP supports MirrorLink in order to use smartphone's US on IVI devices. Unlike the original MirrorLink application, the *CoCaP*’s MirrorLink application makes it possible to obtain and to control the information of the vehicle status by calling VSG APIs.

Table 7. The evaluation items of the vehicle control based on MirrorLink application

Evaluation items	Evaluation methods
MirrorLink Connection	Check MirrorLink connection between a smartphone and IVI device
MirrorLink Application	Check the execution of MirrorLink application on IVI device
Vehicle control based on MirrorLink application	Check the VSG APIs call and the operation of MirrorLink application

The service applications based on the *CoCaP* is implemented as Android application compatible with universal MirrorLink. In this paper, for the direct control of the vehicle, VSG APIs of the vehicle control framework are called, and whether the vehicle is controlled is tested in the MirrorLink as well.

6. Conclusions

The *CoCaP* platform is an architecture designed to satisfy all requirements of Connected Car, enabling the collection of key data and the control of key components of the vehicle through VSG-based API and message clustering. It also enables the remote control based on universal mobile devices such as a smartphone, smart watch and IVI devices. Tizen IVI and Android are ported to Nexcom's VTC1010 and Freescale's i.MX6, respectively, in order to apply the *CoCaP* to the real-world environments, and consequently, to perform functional verification based on real cars. Vehicle Selective Gateway device is developed for the vehicle control on two types of the *CoCaP* and the controllability of each vehicle module are evaluated, and APIs for controllable objects are implemented. APIs are developed in two types to support both Native and Web Application, and each function is verified by connecting IVI device and VSG. In order to apply *CoCaP* to the actual vehicle, the control status of Hyundai Motors' Sonata and *Ssangyong Motors' Tivoli* depending on the API calls is successfully controlled.

The *CoCaP* enables the development of various services for vehicles and drivers based on the controllability of vehicles by using IVI and smartphones. It includes a vehicle control framework, an emotional reasoning framework, a driver vision sensing framework, a web framework and so on for the major services in the Connected Car domain. It also contains a cloud interface to connect various additional devices and web services. In this paper, we present empirical examples of Smart Care service based on (i) driver's bio-signals, (ii) the accident prevention service based on camera and face/head tracking among the major service models of the *CoCaP*. The method for the remote control of the vehicle through MirrorLink-based smartphone application is also presented. The *CoCaP* based on Tizen IVI and Android, which is designed and developed in this paper, includes vehicles, IVI devices, mobile devices, clouds, and major frameworks for interoperability of various types of additional sensors and devices to help the driver and the vehicle. Some examples of the service models that Connected Car is aiming for are implemented and tested. The *CoCaP* is expected to serve as a reference platform for developing services in the Connected Car sector, and to be a cornerstone for the development of the next generation ICT convergence Car industry.

Acknowledgement

This research was funded by a 2019 research Grant from Sangmyung University.

References

[1] J. Gwak, J. Jung, R. Oh, M. Park, M. A. K. Rakhimov, and J. Ahn, "A Review of Intelligent Self-Driving Vehicle Software Research," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 11, pp. 5299-5320, 2019. Article (CrossRef Link)

[2] Q. Hong, R. Wallace, and G. Krueger, "Connected vs. Automated Vehicles as Generators of Useful Data," Center for Automotive Research (CAR), Michigan Department of Transportation, Michigan, MI, USA, 2014.

[3] Y. Cha, J. Kim, B. Park, Y. Park, and S. Kim, "Development of an ICT Car Service Applying a Human-Centered Design," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 6, pp. 3071-3085, 2017. Article (CrossRef Link)

[4] R. Coppola and M. Morisio, "Connected Car: Technologies, Issues, Future Trends," *ACM Computing Surveys*, vol. 49, no. 3, pp. 1-28, Oct. 2016. Article (CrossRef Link)

[5] Wireless Display Standard Explained. Article (CrossRef Link)

[6] M. Broy, I. H. Kruger, A. Pretschner, and C. Salzmann, "Engineering Automotive Software," in *Proc. of the IEEE*, vol. 95, no. 2, pp. 356-373, Feb. 2007. Article (CrossRef Link)

[7] N. Navet and F. S.Lion, "Automotive Embedded Systems Handbook," *CRC Press*, 2009. Article (CrossRef Link)

[8] A. S. Vincentelli and M. D. Natale, "Embedded System Design for Automotive Applications," *Computer*, vol. 40, no. 10, pp. 42-51, Oct. 2007. Article (CrossRef Link)

[9] Setting the Framework for Car Connectivity and User Experience. Article (CrossRef Link)

[10] M. Swan, "Connected Car: Quantified Self becomes Quantified Car," *Journal of Sensor and Actuator Networks*, vol. 4, no. 1, pp. 2-29. Feb. 2015. Article (CrossRef Link)

[11] D. H. Kim, S. J. Baek, and J. Lim, "Measures for Automaker's Legal Risks from Security Threats in Connected Car Development Lifecycle," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 2, pp. 865-882, 2017. Article (CrossRef Link)

[12] Automotive ECU Updates: Keeping the Hackers Out. Article (CrossRef Link)

[13] T. Häberle, L. Charissis, C. Fehling, J. Nahm, and F. Leymann, "The Connected Car in the Cloud: A Platform for Prototyping Telematics Services," *IEEE Software*, vol. 32, no. 6, pp. 11-17, 2015. Article (CrossRef Link)

[14] OSEK/VDK Operating System (V.2.2.3). [Online]. Available: Specification OSEK OS 2.2.3 (irisa.fr)

[15] OSEK/VDK System Generation (V.2.5). [Online]. Available: oil25.pdf (irisa.fr)

[16] AUTOSAR Specification of Operating System. [Online]. Available: Specification of Operating System (autosar.org)

[17] AUTOSAR Specification of ECU Configuration. [Online]. Available: Specification of ECU Configuration (autosar.org)

[18] J. O'Donnell, "Disconnect in the distracted-driving blame game," 2016. Article (CrossRef Link)

[19] J. Machan and C. Laugier, "Intelligent vehicles as an integral part of intelligent transportation systems," *ERCIM News*, vol. 94, pp. 6-7, 2013. Article (CrossRef Link)

[20] S. Hahm and H. Park, "Drowsiness Driving Prevention System using Bone Conduction Device," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 9, pp. 4518-4540, 2019. Article (CrossRef Link)

[21] S. F. Ali and M. T. Hassan, "Feature Based Techniques for a Driver's Distraction Detection using Supervised Learning Algorithms based on Fixed Monocular Video Camera," *KSII Transactions on Internet and Information Systems*, vol. 12, no. 8, pp. 3820-3841, 2018. Article (CrossRef Link)

[22] Royal Society for the Prevention of Accidents, "Driver Fatigue and Road Accidents: A Literature Review and Position Paper," 2001. Article (CrossRef Link)

[23] D. S. Kwon, Y. K. Kwak, J. C. Park, M. J. Chung, E. S. Jee, K. S. Park, H. R. Kim, Y. M. Kim, J. C. Park, E. H. Kim, K. H. Hyun, H. J. Min, H. S. Lee, J. W. Park, S. H. Jo, S. Y. Park, and K. W. Lee, "Emotion Interaction System for a Service Robot," in *Proc. of the 16th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 351-356, 2017. Article (CrossRef Link)

[24] F. A. Machot, A. H. Mosa, K. Dabbour, A. Fasih, C. Schwarzlmüller, M. Ali, and K. Kyamakya, "A Novel Real-time Emotion Detection System from Audio Streams based on Bayesian Quadratic Discriminate Classifier for ADAS," in *Proc. of the 3rd International Workshop on Nonlinear Dynamics and Synchronization and 16th International Symposium on Theoretical Electrical Engineering*, pp. 1-5, 2011. Article (CrossRef Link)
[25] J. Hernandez, X. Benavides, P. Maes, D. McDuff, J. Amores, and R. W. Picard, "AutoEmotive: Bringing Empathy to the Driving Experience to Manage Stress," in *Proc. of the 2014 Companion Publication on Designing Interactive Systems*, pp. 53-56, 2014. Article (CrossRef Link)
[26] W. Dron, "Empathetic Vehicles Cloud Predict Road Rage and Calm Drivers," May 2014. Article (CrossRef Link)
[27] J. S. Choi, J. W. Bang, H. Heo, and K. R. Park, "Evaluation of Fear Using Nonintrusive Measurement of Multimodal Sensors," *MDPI Sensors*, vol. 15, no. 7, pp. 17507-17533, 2015. Article (CrossRef Link)
[28] M. Whang and J. Lim, "A Physiological Approach to Affective Computing," *Intechopen*, pp. 309-318, 2018. Article (CrossRef Link)
[29] H. Prendinger and M. Ishizuka, "Human Physiology as a Basis for Designing and Evaluation Affective Communication with Life-Like Characters," *IEICE Transactions on Information and Systems*, vol. E88-D, no. 11, pp. 2453-2460, Nov. 2005. Article (CrossRef Link)
[30] "Drowsy Driving 2015, Traffic Safety Facts," *NHTSA's National Center for Statistics and Analysis*, Oct. 2017. Article (CrossRef Link)
[31] Denso, Driver Status Monitor. [Online] Available. https://www.denso.com
[32] Connected Car. [Online] Available. https://www.aisin.com
[33] Trywin Dramoni. [Online] Available: http://unibox.co.kr
[34] Volkswagen Driver Alert System. [Online] Available: Driver alert system | Volkswagen UK
[35] Tizen IVI. [Online] Available: IVI/IVI 2.0 VMware - Tizen Wiki
[36] Freescale i.MX 6. [Online] Available: i.MX 6 Series Applications Processors | Multicore Arm Cortex-A7/A9/M4 | NXP
[37] A Study on Types and Causes of Distracted Driving in a Vehicle Accident, Korea Transportation Safety Authority, Dec. 2012.

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. degree 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.