

텔레매틱스 단말기용 소프트웨어 플랫폼 설계

A Design of Service Platform for Telematics Terminal

김기영

(주)센트로닉스, 주임연구원

김동균

(순천향대학교, 박사과정)

이상정

(순천향대학교, 교수)

Key Words : 텔레매틱스, 텔레매틱스 서비스 플랫폼, 임베디드 텔레매틱스 시스템, 블루투스, CAN, GPS, PDA

목 차

- | | |
|-----------------|---------------|
| I. 서론 | 3. 제어 메시지 파서 |
| II. 소프트웨어 플랫폼 | III. 서비스 시나리오 |
| 1. 텔레매틱스 API 설계 | IV. 결론 |
| 2. CAN 제어 메시지 | 참고문헌 |

I. 서론

최근 정보통신의 발달로 초고속 인터넷 인프라를 기반으로 다양한 콘텐츠 서비스 산업이 크게 활성화 되어 있으며, 무선통신의 발달은 사람들에게 공간의 제약에 대한 인식을 불식시켜주는 계기가 되었다.

텔레매틱스를 통해 자동차를 사무실과 가정에 이어 제 3의 인터넷 공간(Connected Car)으로 재구성함으로써 가정과 사무실에서 이용하는 서비스를 차량에서도 단절 없이 제공할 수 있다. 최근 우리나라에서 텔레매틱스(Telematics)가 크게 주목받고 있는 이유는 자동차 안에서 보내는 시간을 보다 가치 있는 시간으로 활용하고자 하는 욕구를 기반으로 자동차와 정보통신 관련 산업을 중심으로 새로운 개념의 부가가치 서비스를 창출할 것으로 기대되기 때문이다[1]. 텔레매틱스는 이제까지 서로 다른 산업이었던 자동차 산업과 정보통신 산업을 융합함으로써, 자동차 산업에는 경쟁력 강화를 통한 새로운 마케팅 기회와 수익을 제공하며, 이동통신 사업자에게는 새로운 사업 기회와 발전 가능성을 제공할 수 있을 것이다[2].

국내에서 IT839전략의 하나로 크게 주목받고 있는 텔레매틱스 서비스는 네비게이션, 위치추적, 인터넷 접속, 원격 차량 진단, 사고감지, 교통정보 등을 제공하는 것을 목표로 하고 있다. 하지만 지금까지 국내의 텔레매틱스 서비스는 주로 교통정보 및 네비게이션 서비스 중심의 한정된 서비스만이 제공되고 있다[3]. 그러나 운전자에게 보다 향상된 텔레매틱스 서비스 제공을 위해서는 자동차 관리 및 진단, 차량 편의시설 제어 등의 기능이 추가되어야 한다. 이를 위해서는 자동차 내부의 네트워크 정보와 PDA나 핸드폰과 같은 운전자 개인의

모바일 단말기 사이에 원활한 데이터 교환을 위한 서비스 플랫폼이 요구된다.

본 논문에서는 자동차 내의 다양한 네트워크 자원과 사용자의 정보단말기 및 외부 서비스 망에 접속하기 위한 임베디드 텔레매틱스 시스템 상에서 일관된 응용 서비스 구현 및 개발을 위한 소프트웨어 플랫폼을 설계한다. 설계된 소프트웨어 플랫폼은 텔레매틱스용 소프트웨어를 개발하는 개발자에게 하부 자동차 네트워크와 각종 하드웨어 장치, 무선 네트워크 장치에 독립적인 개발방법을 제공한다. 이를 위해 각 장치별로 제공되는 서비스를 분류하고 정형화한 텔레매틱스 API(Application Programming Interface)를 설계하였다.

II. 소프트웨어 플랫폼

임베디드 텔레매틱스 시스템에는 여러 가지 역할을 하는 다양한 각종 자동차 네트워크 및 무선 네트워크 장치들과 장비 등으로 인해 일관된 방법의 응용 소프트웨어 개발이 어렵다. 따라서 하부 자동차 네트워크와 하드웨어 장치, 무선 네트워크에 독립적인 소프트웨어 플랫폼을 제공하여 일관된 방법으로 응용 프로그램을 개발하고 유지보수가 용이한 소프트웨어 플랫폼을 제공해야 한다[4]. 본 논문에서는 이를 위해 각 무선 접속기기 별로 제공되는 서비스를 분류하고 정형화한 텔레매틱스 API(Application Programming Interface)를 설계하였다.

자동차 내부 장치를 4가지형태로 분류하고 고유한 식별자 값의 범위를 설정하여 표준 CAN 메시지 포맷에 따라 제어메시지를 설계하였다.

자동차 내부에는 다양한 네트워크 장치들이 존재 한다. 텔레매틱스 서비스를 원활히 수행하기 위해서는 메시지 포맷이 다른 이기종의 네트워크간 통신이 빈번하게 발생한다. 따라서 서로 다른 메시지의 호환성을 위해 메시지 파서를

본 연구보고서는 정보통신부 정보통신연구진흥원에서 지원하고 있는 정보통신기초연구지원사업 No. B1220-0501-0145의 연구결과입니다.

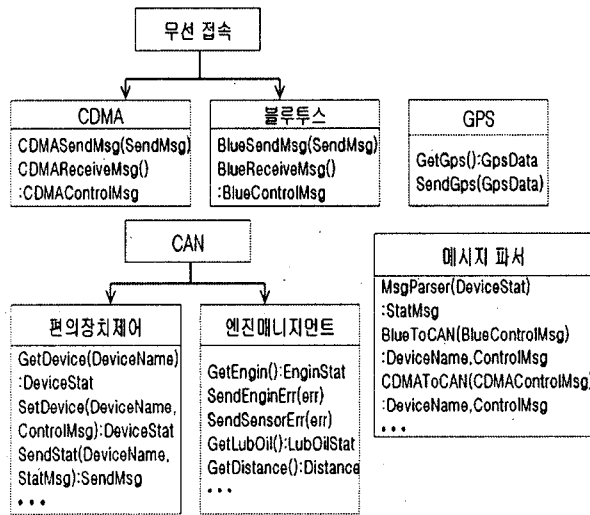
설계하였다.

1. 텔레매틱스 API 설계

<그림 1>은 소프트웨어 플랫폼을 위하여 정형화한 일부 API 함수를 보여주는 그림이다.

본 논문의 소프트웨어 플랫폼은 크게 네 가지 부분으로 구성된다. 무선 네트워크 장치들과 임베디드 텔레매틱스 시스템의 데이터 송수신을 위한 무선 접속부분과 자동차의 각종 네트워크 장치들의 서비스를 위한 CAN 네트워크부분, 위치 기반 서비스를 위해 필요한 GPS 데이터를 수신하는 부분, 그리고 이 기종의 네트워크 간 통신 시 발생하는 상이한 데이터 포맷의 상호 호환성을 위한 메시지 파서 부분으로 구성하였다.

이러한 구분은 텔레매틱스용 소프트웨어를 개발하는 개발자에게 하부 자동차 내부 네트워크와 각종 하드웨어 장치, 무선 네트워크에 독립적인 개발방법을 제공해 준다.



<그림 1> 텔레매틱스 API 함수

블루투스용 API 함수들은 사용자의 모바일 단말을 통한 도어, 윈도우등과 같은 자동차의 각종 편의장치 제어와 엔진 매니지먼트의 모니터링 서비스, 근거리 핫스팟 지역에서 차량 진단 및 유지관리 서비스 등을 위해 사용된다. CDMA용 API 함수들은 근거리 핫스팟 지역을 벗어난 곳에서의 에어백 시스템의 사용 발생과 같은 중대한 사고 발생시 사고처리, 도난 방지와 같은 서비스를 위해 GPS 데이터를 연동하여 서비스 제공자에게 자동으로 정보를 전송하는데 사용된다.

소프트웨어 플랫폼은 급변하는 기술 및 사용자의 요구 사항들로 인해 추가 및 수정이 용이해야 한다. 따라서 최소한의 가장 핵심적인 기능을 할 수 있도록 설계하고 구현한다.

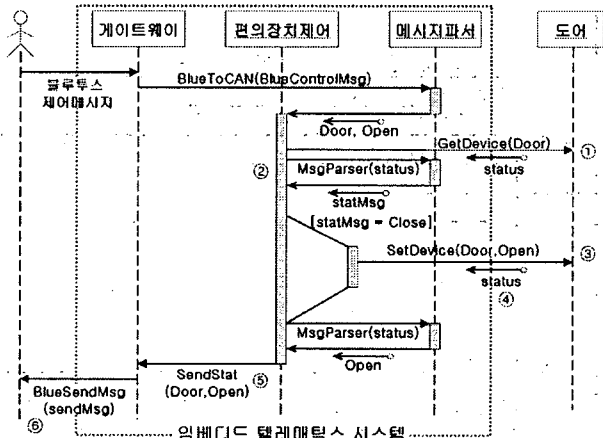
다음 <표 1>은 위에서 분류한 서비스별 API 함수의 리스트와 각 함수의 기능에 대해 설명하고 있다.

<표 1> API 함수 리스트 및 기능

분류	함수 기능
CDMA	void CDMA SendMsg (struct CDMAMsg, CDMA SendMsg); 원거리의 TSP 및 사용자에게 메시지를 송신한다.
	void CDMAMsg CDMA ReceiveMsg (void); 원거리 사용자의 제어 및 TSP로부터 메시지를 수신한다.
블루투스	void Blue SendMsg (char * SendMsg); 블루투스 통신을 이용하여 상태정보를 송신한다.
	char * Blue ReceiveMsg (void); 사용자 및 핫스팟지역에서 블루투스 통신을 이용하여 제어 및 모니터링 데이터를 수신한다.
GPS	void Blue Set (void); 블루투스 통신을 위한 초기 설정과정을 수행한다.
	char * GetGps (void); GPS 수신기로부터 위치정보를 수신한다.
GPS	void SendGps (char GpsData, struct CDMAMsg sendMsg); 원격지의 TSP에게 현재 자동차의 위치를 송신한다.
	struct CANMsg GetDevice (UINT32 DeviceId); CAN통신을 이용하여 자동차 내부 장치의 상태를 모니터링 하기 위해 상태정보를 얻어온다.
CAN	struct CANMsg SetDevice (struct CANMsg SetCANMsg); CAN 통신을 이용하여 자동차 내부의 각종 장치를 제어하는데 사용한다.
	char * SendStat (struct CANMsg StatCanMsg); 현재 상태정보를 송신하는데 사용한다.
CAN	struct CANMsg GetEngin (void); 현재 엔진의 상태 정보를 CAN통신을 이용하여 얻어온다.
	void CANSet (void); CAN통신 모듈을 사용하기 위하여 통신 초기 설정을 수행한다.
메시지 파서	UINT8 * MsgParser (struct CANMsg StatCanMsg); CAN 메시지 형태의 자동차 내부 장치의 상태를 파악하기 위해 상태정보만 추출한다.
	struct CANMsg BlueToCAN (char * BlueControlMsg); 사용자로부터 수신된 제어메시지를 CAN제어 메시지 형태로 변환한다.
메시지 파서	char * CANToBlue (struct CANMsg statCanMsg); 자동차 장치로부터 받은 CAN메시지 형태의 정보를 사용자가 이해할 수 있는 형태인 텍스트 메시지 형태로 변환한다.
	struct CANMsg CDMAToCAN (struct CDMAMsg ReceiveMsg); CDMA 장치로부터 받은 제어 메시지를 CAN 제어메시지 형태로 변환한다.
메시지 파서	char * CANToCDMA (struct CANMsg StatCanMsg); CAN 메시지 형태의 상태정보를 CDMA메시지 형태로 변환한다.
	char * CANIdToStr (UINT32 id); CAN으로부터 받은 장치 식별자 값을 사용자가 이해할 수 있는 형태로 변환한다.
기타	char * CANInfoToStr (UINT8 data); CAN으로부터 받은 상태정보를 사용자가 이해할 수 있는 형태로 변환한다.
	UINT32 StrToCANId (char * id); 사용자로부터 받은 제어메시지 중 장치명을 CAN 메시지 식별자 값으로 변환한다.
기타	UINT8 StrToCANInfo (char * data); 사용자로부터 받은 제어메시지 중 제어신호를 CAN 메시지 제어 데이터 값으로 변환한다.
	struct CDMAMsg StrToCDMAMsg (char * receiveCDMA); 원격지의 TSP 또는 사용자에게 정보를 보내기 위해 생성한 텍스트 형태의 메시지를 CDMA 메시지 자료형으로 변환한다.

다음 <그림 2>는 API에 대한 시퀀스 다이어그램이다. 여

러 가지 편의장치제어 서비스 중에서 자동차 도어 Open 서비스에 대한 시퀀스 다이어그램의 예를 보여주고 있다. 사용자는 휴대하고 있는 모바일 단말인 PDA를 이용하여 텔레매틱스 서비스 게이트웨이에 블루투스 접속을 한다. 이후 도어 Open 제어 메시지를 송신하게 된다. 임베디드 텔레매틱스 시스템은 이 메시지를 메시지 파서에 넘겨주게 되고 메시지 파서에 의해 메시지 내용을 해석하게 된다. 편의장치 제어 서비스에서 도어 Open이라는 내용을 전달받게 된다.



<그림 2> API 시퀀스 다이어그램

편의장치제어 서비스는 CAN 버스를 통해 이 메시지를 도어 장치에 다음과 같은 순서로 송신하게 된다.

- ① GetDevice()함수를 이용하여 도어의 현재 상태를 얻어온다.
- ② 메시지 파서를 통해 도어의 현재 상태를 해석한다.
- ③ 현재 상태가 Close인 경우 SetDevice()함수를 이용하여 도어를 Open하게 된다. 이때 현재 상태가 Open인 경우 현재 상태를 바로 반환한다.
- ④ Open된 도어는 자신의 상태를 편의장치제어 서비스에 반환
- ⑤ SendStat()함수를 이용하여 블루투스용 메시지 생성 후 게이트웨이에 전달
- ⑥ BlueSendMsg()함수를 이용하여 사용자의 모바일 단말인 PDA에 블루투스 통신을 통하여 상태를 반환

2. CAN 제어 메시지

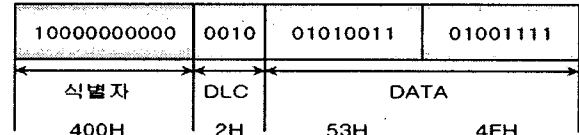
<표 2>는 CAN 제어 메시지의 11비트로 구성되는 식별자 값을 설정하기 위한 구분과 범위를 나타낸다.

<표 2> 11비트 식별자 구분과 범위

구분	범위
실시간 제어 애플리케이션 (안정성 제어, 엔진관리, 변속 등)	100H - 2FFH
일반적인 기능 (파워 윈도우, 계기 등)	300H - 3FFH
편의기능 및 고급기능 (전자부트 작동기, 섀시 등)	400H - 5FFH
멀티미디어 서비스 (인터넷, DTV 등)	600H - 7FFH

<표 2>에 보이는 바와 같이 식별자의 값이 100H~2FFH까지의 범위를 갖는 장치는 파워 트레인, 안정성 제어(ABS, 견인제어, 액티브 서스펜션), 엔진관리, 변속 같은 실시간 제어 애플리케이션에 사용되는 장치들이 사용한다. 식별자 값이 300H~3FFH까지의 범위를 갖는 장치는 파워 윈도우, 좌석 조절장치, 계기 같은 일반적인 장치들이 사용한다. 식별자 값이 400H~5FFH까지의 범위를 갖는 장치는 전자 부트 작동기, 전동 거울 조정장치, 비 탐지, 섀시, 기상관리 등과 같은 편의 기능 또는 고급 기능들을 하는 장치들이 사용한다. 식별자 값이 600H~7FFH까지의 범위를 갖는 장치는 인터넷, 음성 및 동영상 등과 같은 멀티미디어 서비스를 제공하는 장치들이 사용한다.

다음 <그림 3>은 CAN 메시지 형식에 따른 도어 Open 제어 메시지의 예를 보여준다.



<그림 3> CAN 제어 메시지 예

도어에 대한 유일한 식별자 값은 16진수 400H로 설정한 값이고, 제어 데이터 길이는 2바이트의 데이터를 나타내는 16진수 2H를 나타낸다. 실제 제어 데이터는 2바이트로 나타낼 수 있는데 첫 번째 바이트는 Set을 의미하는 'S'의 아스키 코드값 53H, 두 번째 바이트는 Open을 의미하는 'O'의 아스키 코드값 4FH로 나타낸다.

임베디드 텔레매틱스 시스템의 서비스 게이트웨이는 사용자의 PDA로부터 블루투스 통신을 통해 도어 Open에 대한 메시지를 수신한 후 메시지 파서에 의해 도어 Open에 대한 메시지임을 해석한 후 그림 3과 같은 메시지를 생성한다. 이후 CAN 버스에 연결된 모든 장치들은 이 메시지를 수신한 후 식별자 값을 확인하여 자신의 데이터가 아니면 메시지를 무시하게 된다. 도어는 이 값을 수신한 후 메시지를 해석해서 자신의 식별자 값인 400H와 비교를 한 후 자신의 데이터임을 확인한 후 도어를 Open상태로 변경하게 된다.

3. 제어 메시지 파서

텔레매틱스 시스템에는 여러 가지 역할을 하는 다양한 각종 자동차 장비 및 자동차 하부 네트워크와 사용자 및 외부 네트워크와의 연결을 위한 무선 네트워크 장치들이 존재한다. 따라서 이러한 하부 자동차 네트워크와 하드웨어 장치, 각종 무선 네트워크간 통신이 빈번하게 발생하게 된다.

본 논문에서는 이 기종의 네트워크간 통신 시 발생하는 가장 큰 문제점인 메시지 호환성을 위해 메시지 변환을 담당하는 메시지 파서를 설계하였다. <그림 4>는 메시지 변환을 위한 주요 자료구조 정의부분에 대한 그림이다.

```

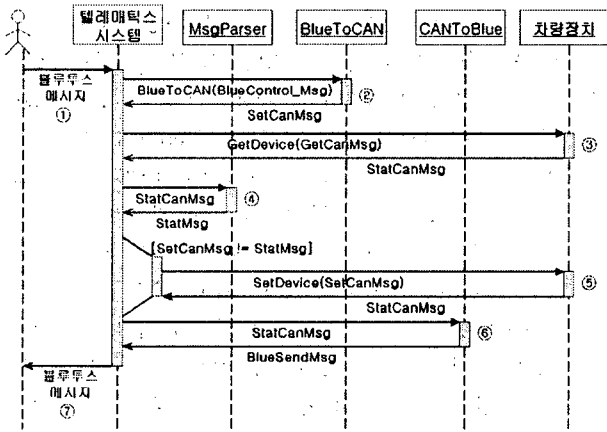
// #ifndef UINIT8
// #define UINIT8 unsigned char
// #endif
// #ifndef UINIT16
// #define UINIT16 unsigned short
// #endif
// #ifndef UINIT32
// #define UINIT32 unsigned long
// #endif
.....
// =====
// // Vehicle Device ID
// =====
// #define PCU 0x100
// #define ECU 0x101
// #define ES 0x102
// #define ARV 0x103
// #define ARFS 0x104
// #define ARFP 0x105
// #define ECM 0x106
// #define TPS 0x107
// #define CEPF 0x108
// #define PCSV 0x109
// #define TCU 0x140
// #define DOOR 0x400
// #define WINDOW 0x300
// #define ROOF 0x410
// #define ECM 0x420
.....
// =====
// // Message information
// =====
// #define SET 0x43
// #define GET 0x47
// #define OPEN 0x44
// #define CLOSE 0x43
// #define ERR 0x45
.....
typedef struct
{
    UINIT32 id; /* message identifier */
    UINIT8 dlc; /* Data Length Code (0-8) */
    UINIT8 mdf; /* message frame format */
    UINIT8 a_data[8]; /* CAN data */
    UINIT32 time_stamp; /* receive time stamp */
} CANMsg;

typedef struct
{
    char PhoneNumber[11];
    char Msg[80];
} CDMA_Msg;
.....

```

<그림 4> 메시지 자료구조 정의

사용자로부터 텔레매틱스 시스템으로 송신되는 제어메시지 정보는 일반 텍스트형태의 메시지와 CDMA SMS 메시지 형태의 두 가지로 구분되어 전송되게 된다. 사용자 및 근거리 핫스팟 지역에서 블루투스 통신을 통해 전송되는 제어 메시지는 일반 텍스트 형태로 전송되고 원거리의 사용자 및 TSP로부터 전송되는 메시지는 CDMA SMS 메시지 형태로 전송된다. 전송된 제어메시지는 자동차 하부 네트워크인 CAN을 이용하여 각종 장치들을 제어하고 모니터링 해야 하기 때문에 CAN 메시지 형태의 변환과정을 거쳐 CAN 버스를 통해 메시지를 실제 장치에 전송하게 된다. 실제 장치에서는 전송 받은 CAN 메시지를 해석하여 메시지에 해당하는 동작을 하게 되고 동작 후에 상황을 다시 재전송하게 된다.



<그림 5> 메시지 흐름도

<그림 5>는 사용자로부터 전달 받은 메시지를 CAN 메시지 형태로 변환 후 차량장치를 제어하고 제어 후의 상태인 CAN 메시지를 사용자에게 다시 반환 해주는 과정을 나타내는 메시지 흐름도이다.

제어 메시지의 송신에서부터 제어 후의 상태를 모니터링 하기까지의 과정은 크게 7가지 부분으로 나누어 볼 수 있다.

① 텔레매틱스 시스템에서는 사용자로부터 일반 텍스트 형태의 제어메시지인 BlueControl_Msg를 수신한다.

② 수신된 BlueControl_Msg는 BlueToCAN()함수에 전달 되어 토큰분류를 통해 제어하고자 하는 장치명과 제어 메시지를 분류하여 해당되는 장치를 제어하기 위한 CAN 메시지 형태인 SetCanMsg를 생성한다. 생성된 SetCanMsg는 텔레매틱스 시스템에 반환되게 된다.

③ 제어하고자 하는 장치의 현재 상태를 알아보기 위해 SetCanMsg의 식별자 값을 사용하여 CAN 메시지 형태인 GetCanMsg를 생성, 차량장치에 현재의 상태를 요청한다. 차량장치는 현재 상태를 CAN 메시지 형태인 StatCanMsg를 텔레매틱스 시스템에 반환하게 된다.

④ 위 과정에서 반환된 StatCanMsg를 MsgParser()함수에 전달하여 현재 상태 정보인 StatMsg를 추출한다. 추출된 StatMsg는 텔레매틱스 시스템에 반환한다.

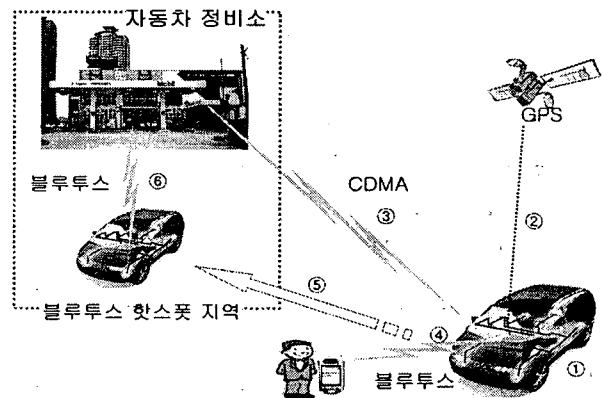
⑤ 반환된 StatMsg와 ②번과정에서 생성된 SetCanMsg의 제어메시지를 비교하여 현재 상태와 제어하기위한 상태를 판단한다. 이때 제어하기 위한 상태와 현재상태가 같지 않다면 현재상태를 변경시켜줘야 한다는 것을 의미하기 때문에 SetCanMsg를 차량장치에 송신하여 장치를 제어하게 된다. 제어 후의 상태인 StatCanMsg를 다시 텔레매틱스 시스템에 반환하게 된다.

⑥ 반환된 StatCanMsg를 다시 사용자에게 알려주기 위해 CANToBlue() 함수에 전달되게 되고 사용자에게 변환 후의 정보를 알려주기 위해 일반 텍스트 형태의 메시지인 BlueSendMsg를 생성하여 텔레매틱스 시스템에 반환하게 된다.

⑦ 텔레매틱스 시스템에서는 반환된 BlueSendMsg를 사용자에게 전달함으로써 제어과정을 완료하게 된다.

III. 서비스 시나리오

<그림 6>은 텔레매틱스 응용서비스 시나리오에 대한 그림이다. 사용자는 블루투스 통신이 가능한 개인 정보단말기인 PDA를 가지고 있다. 사용자는 자동차의 자가 진단서비스인 소모성 부품 상태 및 차량의 이상부품 발생여부를 개인 정보 단말기로 모니터링 한 후 이상부품 발생과 소모성 부품의 교체시기를 전달 받게 된다.



<그림 6> 응용서비스 시나리오

서비스 시나리오는 크게 6단계로 구분해서 설명하면 다음과 같다.

① 사용자의 자동차 운행도중 자동차 부품의 이상을 감지한 후 자동차의 디바이스 장치에서는 CAN 버스를 통해 CAN 데이터 형식의 오류 메시지를 임베디드 텔레매틱스 시스템에 전달하게 된다.

② 임베디드 텔레매틱스 시스템에서는 이상이 발생한 부품을 교체, 서비스를 받아야 한다는 것을 사용자에게 알려주어야 한다. 현재 자동차의 위치를 파악하기 위해 GPS 수신기로부터 GPS 데이터를 얻어온다. 현재 자동차의 위치를 중심으로 가장 가까운 자동차 정비소를 탐색한다.

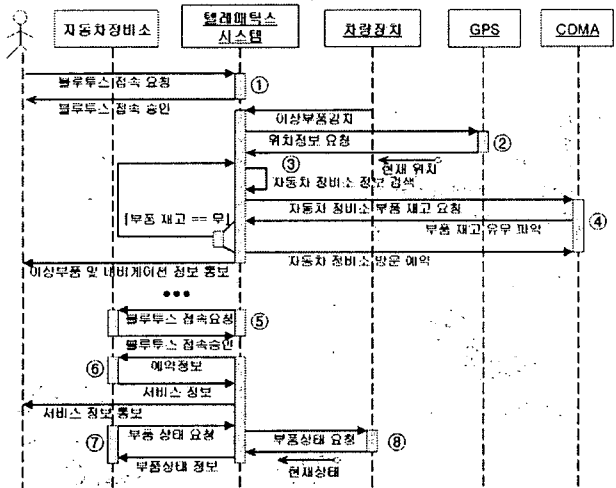
③ 탐색된 자동차 정비소에 CDMA SMS를 이용하여 현재 이상이 발생한 부품 정보를 보내 재고 유무를 요청한다. 자동차 정비소에서는 재고 유무를 파악한 후 수신된 번호를 통해 재고 유무를 임베디드 텔레매틱스 시스템에 재전송하게 된다. 재고 여부에 따라 자동차 정비소 방문 예약 메시지를 송신한다.

④ 임베디드 텔레매틱스 시스템에서는 블루투스로 연결되어 있는 사용자의 정보단말기인 PDA에 이상이 발생한 부품에 대한 정보와 현재 위치에서 가장 가깝고, 이상부품 재고를 보유하고 있는 방문 예약한 자동차 정비소의 정보를 송신하게 된다.

⑤ 사용자는 가장 가까운 자동차 정비소에 대한 정보를 네비게이션 서비스를 이용하여 최적의 경로로 이동하게 된다.

⑥ 자동차 정비소에는 블루투스 통신을 이용하여 근거리 핫스팟을 형성하고 있다. 따라서 사용자의 자동차가 자동차 정비소의 근거리 핫스팟 지역에 진입하는 순간 자동차내의 임베디드 텔레매틱스 시스템은 자동차 정비소의 블루투스 AP와 블루투스 접속을 요청한 후 블루투스 연결설정을 완료하게 된다. 블루투스 연결설정이 이루어진 후에는 임베디드 텔레매틱스 시스템에서 방문 예약 정보 메시지를 자동차 정비소에 송신하고, 자동차 정비소에서는 접수번호와 담당자, 정비 라인 번호를 전송한다.

<그림 7>은 응용 서비스 시나리오에 대한 서비스 흐름도이다.



<그림 7> 서비스 흐름도

① 과정에서 사용자와 텔레매틱스용 임베디드 시스템간의 블루투스 접속을 수행한다. 다음 <그림 8>은 위 서비스 흐름도에서 ②부터 ④까지의 과정을 함수 호출과정을 통해 설명하고 있다.

```

int telematic_service (int blue_id, int can_id, int cdma_id) {
.....
FD_SET (blue_id, &read_fds); //bluetooth 통신
FD_SET (can_id, &read_fds); //CAN 통신
FD_SET (cdma_id, &read_fds); //CDMA 통신
char phonenum [11]; char servicecenter [80]; char position [70];
CDMAmsg cdmaSendMsg; cdmaReceiveMsg;
char errStatMsg [80]; char reservation [80] = "1390차량 예약입니다.";
if (select (maxfd+1, &read_fds, NULL, NULL, NULL) < 0) { ... }
if (FD_ISSET (can_id, &read_fds)) {
    BCL_ReceiveCanMsg (CANBoard, Controller, &receiveMsg, BCL_NO_WAIT);
    UINT8 status = MsgParser (receiveMsg);
    if (status == ERR) {
        while (1) {
            position = GetGps (); // ②
            servicecenter = SearchService (position);
            phonenum = SearchPhone (servicecenter);
            errStatMsg = CANToCDMA (receiveMsg); // ③
            cdmaSendMsg->PhoneNumber = &phonenum;
            cdmaSendMsg->Msg = &errStatMsg;
            CDMA_SendMsg (cdmaSendMsg);
            while (cdmaReceiveMsg->Msg != NULL) {
                cdmaReceiveMsg = CDMA_ReceiveMsg ();
                if (!strcmp (cdmaReceiveMsg->Msg, "OK")) break;
            }
            cdmaSendMsg->Msg = &reservation;
            CDMA_SendMsg (cdmaSendMsg); // ④
        }
    }
}
}

```

<그림 8> 정비소 예약과정까지의 호출과정

②의 과정에서 자동차의 부품인 에어플로우 센서의 이상을 감지한 후 CAN 네트워크를 통해 임베디드 시스템에 error 상태 메시지를 전송하게 된다. 임베디드 시스템에서는 MsgParser 함수를 이용하여 현재 상태를 판단하게 되고, 현재 상태가 ERR 일 경우 GPS수신기로부터 현재 자동차의 위치 정보를 GetGps 함수를 이용하여 수신하게 된다. ③의 과정에서 수신된 자동차의 위치정보를 네비게이션 서비스 데이터베이스에서 가까운 자동차 정비소의 상호를 SearchService 함수를 이용하여 얻어온다. 또 이동중인 자동차이기 때문에 CDMA 통신을 이용해야하므로 정비소의 상호를 바탕으로 지역 전화번호 데이터베이스에서 CDMA번호를 SearchPhone 함수를 통해 얻어온다. ④의 과정에서 현재 error의 상태정보는 CANMsg 형식으로 되어 있기 때문에 CDMA 메시지 형태로 변환하기 위해 CANToCDMA 함수를 통해 일반 텍스트 형태로 변환을 한다. 변환된 정보를 ③의 과정에서 얻은 자동차 정비소의 CDMA번호와 함께 CDMAmsg 형태의 메시지를 생성하고 CDMA_SendMsg 함수를 통해 자동차 정비소에 에어플로우 센서의 재고 여부를 문의하게 된다. 자동차 정비소에서는 이를 확인 후 바로 재고 여부를 재전송 해주게 되고 임베디드 시스템에서는 이를 수신하여 재고가 있으면 현재 CDMA번호를 재사용해서 사용자 자동차의 번호를 이용하여 예약 메시지를 자동차 정비소에 CDMA_SendMsg 함수를 이용하여 송신하게 된다. 이와 동시에 사용자에게 이상부품 정보를 전송하고 네비게이션 시스템에 자동차 정비소의 위치를 보여주게 된다. 사용자는 네비게이션 서비스를 이용하여 빠르고 안전하게 자동차 정비소까지 이동을 하게 되고, 블루투스 근거리 핫스팟을 구성하고 있는 자동차 정비소에 진입하자마자 ⑤부터 ⑧까지의 과정이 이루어지게 된다. 다음 <그림 9>는 자동차 정비소에 도착했을 때의 과정이다.

```

int telematic_service (int blue_fd, int can_fd, int cdma_fd) {
.....
FD_SET(blue_fd, &read_fds); //bluetooth 통신
FD_SET(can_fd, &read_fds); //CAN 통신
FD_SET(cdma_fd, &read_fds); //CDMA 통신
CANMsg BlueControlMsg, statMsg;
char RetReservation[80] = "1980 차량입니다";
char emStatMsg[80]; char reservation[80] = "1980차량 예약합니다.";
char receiveBlueMsg[256]; char sendBlueMsg[256];
if(select(maxfdp, &read_fds, NULL, NULL, < 0) { ... }
if (FD_ISSET(blue_fd, &read_fds))
{
receiveBlueMsg = BlueReceiveMsg();
if (strcmp(receiveBlueMsg, "welcom"))
{
BlueControlMsg = BlueToCAN(&receiveBlueMsg);
if (BlueControlMsg->id != 0)
{
statMsg = GetDevice(BlueControlMsg->id); ⑦
sendBlueMsg = CANToBlue(statMsg);
BlueSendMsg(sendBlueMsg) ⑧
}
}
BlueSendMsg(&receiveBlueMsg);
}
else BlueSendMsg(&RetReservation); ⑥
}
}

```

<그림 9> 정비소 도착 후 호출과정

⑤의 과정은 ①의 과정과 동일하게 동작을 한다. 즉 블루투스 근거리 핫스팟을 형성하고 있는 자동차 정비소에 도착하면 블루투스 장치를 인식하게 되고 연결설정을 자동적으로 실행하여 통신이 가능한 상태가 된다. 그 후 ⑥의 과정에서 1980차량으로 예약했다는 메시지를 자동차 정비소에 BlueSendMsg함수를 이용하여 송신하고, 자동차 정비소에서는 서비스를 담당할 담당자와 서비스 받을 장소인 0번 게이트의 정보를 보내주게 된다. 임베디드 시스템에서는 담당자와 서비스 장소를 사용자에게 알려주게 되고 사용자는 서비스 장소로 이동을 한다. 서비스 장소 이동 후 ⑦의 과정에서 담당자는 블루투스를 이용하여 이상이 발생했던 에어플로우 센서의 상태를 다시 한번 요청하게 된다. ⑧의 과정에서 임베디드 시스템은 CAN통신을 이용하여 에어플로우 센서에 현재 상태를 요청하고 상태를 받는다. 받은 상태를 담당자에게 알려줌으로써 서비스를 받게 된다.

IV. 결론

본 논문에서는 하부 자동차 네트워크와 하드웨어 장치, 무선 네트워크에 독립적인 일관된 방법으로 응용 프로그램을 개발하고 유지보수가 용이하도록 하기 위해 각 무선 접속기기 별로 제공되는 서비스를 분류하고 정형화한 텔레매틱스 API를 설계하였다. 무선 네트워크 장치들과 임베디드 텔레매틱스 시스템과의 데이터 송수신을 위한 무선 접속부분과 자동차의 각종 네트워크 장치들을 위한 CAN 네트워크 부분, 위치기반 서비스를 위해 GPS 부분, 이 기종의 네트워크간 통신 시 발생하는 상이한 데이터 포맷의 상호 호환성을 위한 메시지 파서를 설계하였다.

참고문헌

[1] 안선일, 장병준, 이윤덕 “텔레매틱스 동향 및 기술개발 방향”, 한국 정보과학회 학회지, 제 23권 제2호 p.77-82, 2005년 2월
[2] 김주완 “텔레매틱스 기술 및 서비스 동향”, IITA 주간기술동향, 1144호, 2004년 5월

[3] 최지훈, 장병태 “텔레매틱스 기술 및 서비스동향”, IITA 주간기술동향, 1157호, 2004년 8월
[4] Magnus Gustafson, Per Magnusson, “Wireless Application Development in a Telematics environment”, Master’s Degree in Computing Science Chalmers University of Technology, Sweden, 2003.
[5] Magnus Persson, Fredrik Gustafsson, “A Bluetooth-CAN Gateway”, Electrical Engineering Students at Chalmers University of Technology for Sigma System AB, 2001
[6] P.Bhagwat, “Bluetooth: Technology for Short-Range Wireless Apps”, IEEE INTERNET COMPUTING, May, 2001.
[7] D.Reilly, A. Taleb-Bendiab, “A Service-Based Architecture for In-Vehicle Telematics Systems”, 22nd International Conference on Distributed computing Systems Workshops, July 2002, P.741
[8] 이진희, 윤용익, “차세대 텔레매틱스 서비스를 위한 통합 제어 플랫폼”, 한국정보처리학회 학회지, Vol:11 No.4 P.45-55, 2004년 7월
[9] PHILIPS, <http://www.semiconductors.philips.com>
[10] CAN in Automation, <http://www.can-cia.org>
[11] Telematics Research Group Incorporated, <http://www.telematicsresearch.com>
[12] Gerrit de Boer, Peter Engel, Werner Praefcke, “Generic remote Software Update for Vehicle ECUs using a Telematics Device as a Gateway”, 9th International Forum on Advanced Microsystems for Automotive Applications Berlin, March 2005
[13] 김기영, 김동균, 이상정, “텔레매틱스 서비스 게이트웨이 설계 및 구현”, 한국정보과학회 한국컴퓨터종합학술대회 2005, 2005년 7월