TRACE32를 활용한 OSEK OS 디버깅

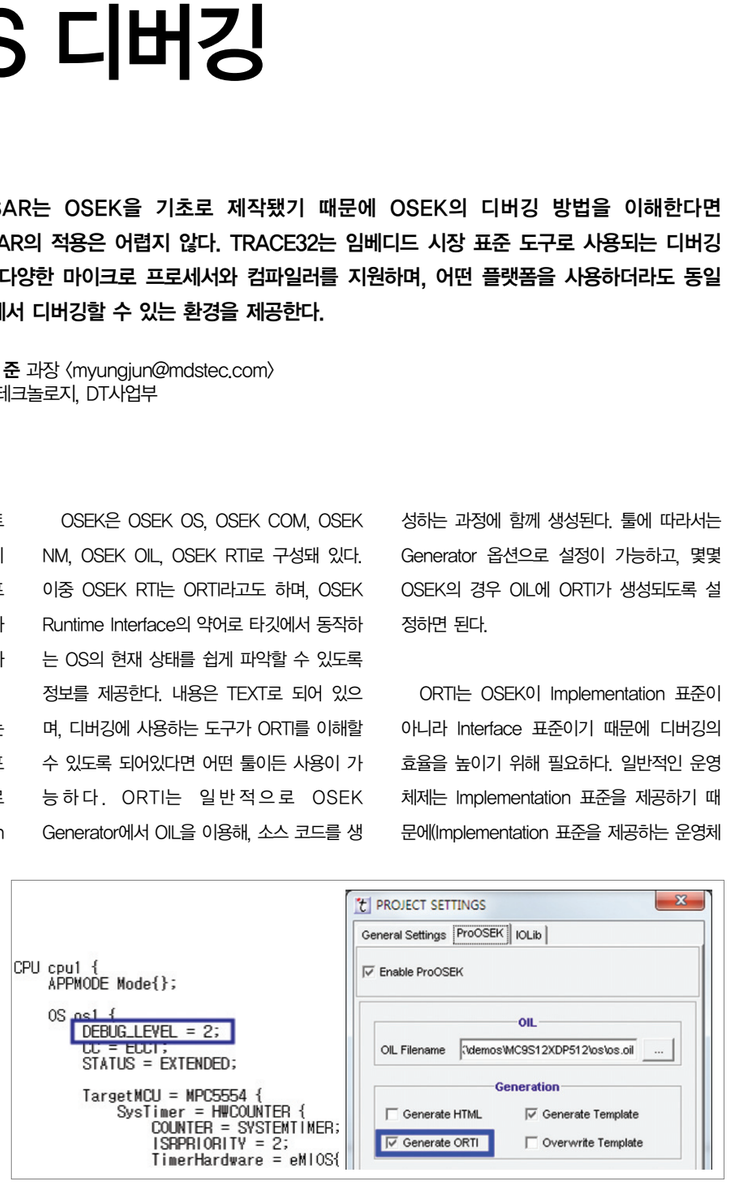
AUTOSAR는 OSEK을 기초로 제작됐기 때문에 OSEK의 디버깅 방법을 이해한다면 AUTOSAR의 적용은 어렵지 않다. TRACE32는 임베디드 시장 표준 도구로 사용되는 디버깅 툴이다. 다양한 마이크로 프로세서와 컴파일러를 지원하며, 어떤 플랫폼을 사용하더라도 동일한 GUI에서 디버깅할 수 있는 환경을 제공한다.

글 | **성 명 준** 과장 〈myungjun@mdstec.com〉
MDS테크놀로지, DT사업부

복잡한 소프트웨어를 잘 관리하고, 소프트웨어의 재활용성을 높이기 위해 제안된 것이 바로 AUTOSAR다. AUTOSAR는 전장 소프트웨어의 아키텍처와 개발 방법론을 표준화하는 것을 목표로 제안된 공개 표준으로 자동차 전장의 많은 분야에 적용될 것이다.

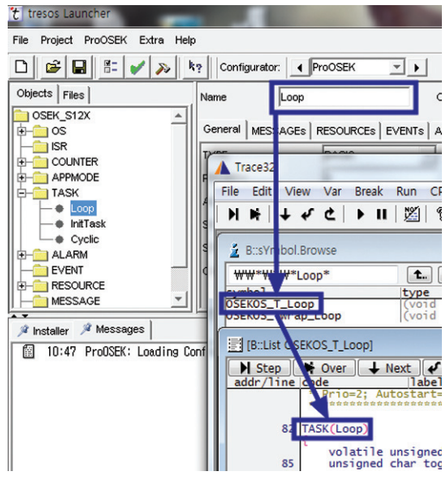
OSEK은 차량용 전자 제어장치에 사용하는 소프트웨어의 공개 아키텍처다. 독일의 대표 자동차 개발사들이 중심돼 시작된 프로젝트로 'Offene Systeme und deren Schnittstellen fur die Elektronik im Kraftfahrzeug (영어로는 Open Systems and their Interfaces for the Electronics in Motor Vehicles)'를 의미한다. 최근에는 OSEK을 기초로 작성된 AUTOSAR가 사용되고 있으며, OSEK은 더 이상 업데이트가 이뤄지지 않고 있다. 최신 트렌드가 AUTOSAR이지만 OSEK을 기초로 제작됐기 때문에 OSEK의 디버깅 방법을 이해한다면 AUTOSAR의 적용은 어렵지 않다.

OSEK은 OSEK OS, OSEK COM, OSEK NM, OSEK OIL, OSEK RTI로 구성돼 있다. 이중 OSEK RTI는 ORTI라고도 하며, OSEK Runtime Interface의 약어로 타깃에서 동작하는 OS의 현재 상태를 쉽게 파악할 수 있도록 정보를 제공한다. 내용은 TEXT로 되어 있으며, 디버깅에 사용하는 도구가 ORTI를 이해할 수 있도록 되어있다면 어떤 툴이든 사용이 가능하다. ORTI는 일반적으로 OSEK Generator에서 OIL을 이용해, 소스 코드를 생성하는 과정에 함께 생성된다. 툴에 따라서는 Generator 옵션으로 설정이 가능하고, 몇몇 OSEK의 경우 OIL에 ORTI가 생성되도록 설정하면 된다.



【그림 1】 ORTI 파일을 생성하기 위해 OIL 작성 또는 툴 설정 방법, 프리스케일의 OSEKturbo OIL 파일 내용, EB의 Tresos의 툴 설정

ORTI는 OSEK이 Implementation 표준이 아니라 Interface 표준이기 때문에 디버깅의 효율을 높이기 위해 필요하다. 일반적인 운영체제는 Implementation 표준을 제공하기 때문에(Implementation 표준을 제공하는 운영체제라고 하더라도 CPU 아키텍처에 종속적인 부분은 다른 코드를 제공한다), 동일한 소스 코드를 갖고 있는 반면, Interface 표준만 제공하는 OSEK은 제조사마다 제공하는 소스 코드가 다르다. 예를 들어 Task의 이름을 MySample이라는 이름으로 정의하면, MySample Task는 Generator에 따라서 tskMySample, funcMySample, OSEKOS_T_MySample과 같이 다양한 형태로 생성된다. 이로 인해 디버깅 과정에 Task를 찾거나 현재 상태를 확인하는 것이 어렵다. **그림 2**처럼 작성한 것과 다른 이름으로 나타나면 디버깅이 쉽지 않게 된다.

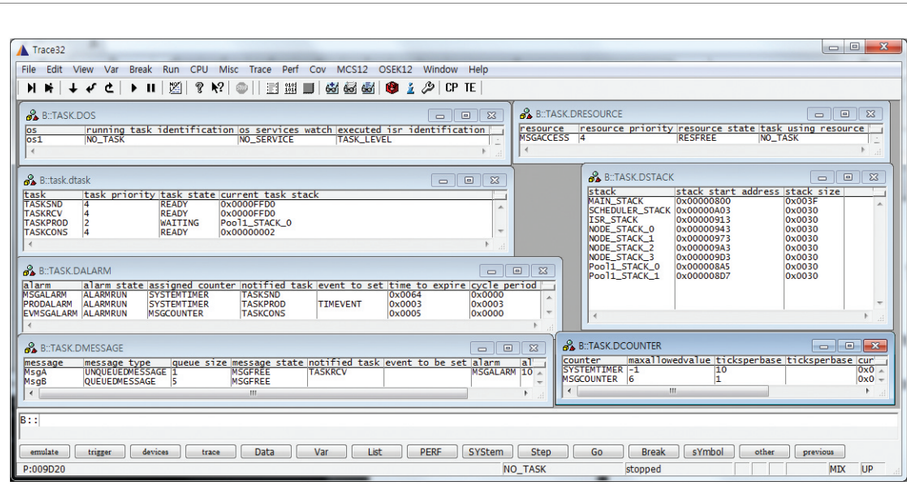


【그림 2】 작성된 OIL의 내용과 실제 코드 내용. tresos를 이용해 OIL을 작성할 때, Task 이름이 Loop 였지만, 실제 디버깅을 하는 환경에서는 OSEKOS_T_ Loop로 검색해야 해당 함수를 찾을 수 있다.

TRACE32는 임베디드 시장 표준 도구로 사용되는 디버깅 툴이다. 다양한 마이크로 프로세서와 컴파일러를 지원하며, 어떤 플랫폼을 사용하더라도 동일한 GUI에서 디버깅할 수 있는 환경을 제공한다. 이러한 장점은 자동차 시장처럼 다양한 플랫폼을 사용하는 엔지니어에게 툴을 다시 학습할 필요가 없는, 다시 말해 어떤 플랫폼을 사용하더라도 바로 업무에 활용할 수 있도록 하는 장점을 제공한다. 그리고 TRACE32에서 제공하는 다양한 기능은 소프트웨어의 문제 원인을 쉽게 찾고, Board Bring-up부터 최종 양산에 이르기까지 다양한 개발 단계에서 모두 적용할 수 있다.

추가로, TRACE32에서는 OS Awareness 기능을 제공한다. OS Awareness라는 타깃에서 동작하는 운영체제의 정보를 쉽게 관찰할 수 있는 기능을 제공한다. OSEK OS 또한 TRACE32의 OS Awareness 기능을 쉽게 디버깅을 할 수 있다. OSEK OS의 제조사와 사용하는 마이크로 프로세서가 다르더라도, 동일한 OSEK의 정보를 관찰할 수 있다.

그림 2에서와 같이 Task의 이름이 바뀌더라도, 다양한 상태 정보를 빠르게 관찰할 수 있다. 뿐만 아니라, 운영체제가 가지고 있는



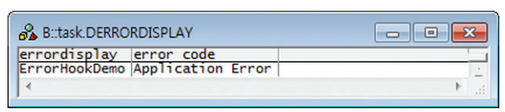
【그림 3】 OSEK OS Awareness를 이용한 디버깅 환경. OSEK OS가 가지고 있는 모든 리소스의 상태를 한 눈에 관찰할 수 있다.

```
VERSION
{
      KOIL = "2.1";
      OSSEMANTICS = "ORTI", "2.1";
};
IMPLEMENTATION ProOSEK_4_0 {
...<생략>...

  TASK {
      OSEKOS_ve_ID = "ID";
      ENUM  [ "SUSPENDED"=0,"READY"=1,"RUNNING"=2,"WAITING"=3 ] STATE, "Taskstate";
...<생략>...
      };
...<생략>...
};

...<생략>...
TASK Cyclic {
      OS_ID = "2";
      STATE = "OSEKOStaskStatus[2]&0x3";
...<생략>...
};
TASK InitTask {
      OS_ID = "1";
      STATE = "OSEKOStaskStatus[1]&0x3";
...<생략>...
};
...<생략>...
```

【그림 4】 ORTI 예제. TASK에 대한 Declaration Section과 두 TASK에 대한 Implementation Section로 작성된다.

```
IMPLEMENTATION ProOSEK_4_0 {
...
   ErrorDisplay {
      ENUM UINT8 ["System Error"=1, "Application Error"=2] ErrorCode, "Error Code";
   };
...
};
...
ErrorDisplay ErrorHookDemo{
    ErrorCode = "gErrorCode";
};
...
```

【그림 5】 ORTI에 사용자 코드 추가. 추가된 리소스를 TRACE32를 통해 출력할 수 있다.



【그림 6】 Error Code Display – 사용자 정의 Error Code를 출력할 수 있다.

다양한 리소스도 동일한 방식으로 제공한다. **그림 3**의 내용처럼, 운영체제가 가지고 있는 모든 리소스를 한 눈에 관찰이 가능하다. 특히 이 내용은 디버깅 심볼 정보에 의존할 경우 Task나 모든 리소스의 정보가 OIL을 작성할 때와 다르게 나타나는데, ORTI를 참고해 출력하면 OIL에 작성할 때와 동일한 형식으로 나타나 현재 운영체제의 상태를 더 빠르게 관찰할 수 있다.

이러한 ORTI는 KOIL(Kernel Object Interface Language)로 작성됐다. KOIL 문법은 C의 구조체를 작성하는 것과 유사하다. KOIL은 Declaration Section과 Information Section으로 구분된다. Declaration Section은 Kernel Object가 가지는 속성(Attribute)이 열거되며, Information Section에서는 해당 정보를 실제 Kernel에서 어떤 Symbol 정보를 참조해야 하는지 알려준다.

그림 4에서 Implementation Section에서 TASK의 두 번째 속성은 TASK의 상태정보이다. 만일 현재 상태를 확인하려면, InitTask는 OSEKOStaskStatus[1]&03 값을 참고하면 된다. Cyclic의 경우 OSEKOStaskStatus[2]&03 값을 참고하면 된다. 만일 OSEKOStaskStatus[1]&03가 0이면 InitTask는 'SUSPENDED' 상태이며, 2이면 'RUNNING' 상태가 된다.

TRACE32의 OS Awareness 기능은 OSEK OS에서 제공하는 정보를 더욱 정밀하게 관찰 할 수 있다. 예를 들면, OSEK에서는 ErrorHook을 이용해, 문제가 발생하면 특정 변수에 문제의 정보를 기록해 둘 수가 있다. 만일 시스템에서 문제가 발생하면 gErrorCode라는 전역 변수에 Error Code를 기록하도록 코드를 작성하고, gErrorCode의 값이 1이면 "System Error"로 정의하고, 2이면 "Application Error"로 정의하면 **그림 5**와 같이 ORTI를 작성할 수 있다.

만일 시스템에 문제가 발생하면, 시스템을 멈추고 Error Code 내용을 **그림 6**과 같이 출력할 수 있다.

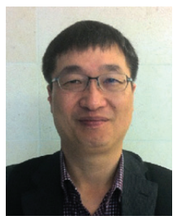
전장 소프트웨어는 시간이 갈수록 그 복잡도가 높아질 것이다. 이러한 소프트웨어 디버깅은 점차 개발자에게 많은 양의 데이터를 이용해 디버깅하는 환경을 요구하게 된다. 특히, 본문에서 활용한 OSEK보다 AUTOSAR는 더 많은 양의 데이터가 개발자에게 제공된다. 이러한 정보는 체계적으로 관찰하지 않으면, 소프트웨어 개발이 매우 어려워진다. 하지만, TRACE32의 OS Awareness는 소프트웨어의 많은 데이터를 체계적인 정보로 제공한다. 복잡한 소프트웨어를 개발하는 과정에 매우 편리한 디버깅 기능을 제공함으로써 필요한 정보를 보기 위한 소비적인 프로세스를 줄일 수 있다.

미러링크, 인포테인먼트의 대세될 것

MDS테크놀로지 **노 재 민** 상무

MDS테크놀로지가 지난 3월 양재동 엘타워에서 자동차 인포테인먼트 환경 구축을 위한 솔루션 적용사례와 업계 기술동향을 소개하는 '자동차 인포테인먼트 솔루션 세미나'를 개최했다. 노재민 상무가 인포테인먼트 시스템과 스마트폰의 결합 동향을 설명했다. 노 상무는 스마트폰 기반의 인포테인먼트 시스템이 대세가 될 것이라고 했다.

정리 | **한 상 민** 기자 〈han@autoelectronics.co.kr〉


노 재 민 상무

스마트한 기기란 언제 어디서나 돼는 연결성을 지니면서, 다양한 앱을 활용할 수 있고, 멀티미디어들을 플레이 할 수 있는 기기다. 스마트한 기기에는 자동차도 포함된다. 다만 스마트한 기기가 커뮤니케이션, 정보, 미디어 엔터테인먼트, 금융 등의 기능을 지녔다면, 스마트한 자동차는 안전과 보안, 차량 성능 모니터링, 커넥티드 내비게이션, 전기차용 애플리케이션 등의 추가적 기능을 포함한다.

스마트 카를 구현하는 방법

스마트 카 개발의 필요성은 인터넷의 이용, 용이한 업그레이드 및 커스터마이징, 스마트 라이프란 고객의 니즈를 충족시키는 것이다. 때문에 카 메이커와 서플라이어는 가능한 비용을 줄이면서 스마트한 자동차를 구현해 브랜드 이미지를 높이려 하고 있다.

스마트 카, 인포테인먼트 시스템을 통해 고객의 니즈를 충족시키는 방법은 크게 두 가지로 나뉜다. 하나는 자동차 스스로 모든 것을 해 스마트해지는 것이고 다른 하나는 어느 정도의 시스템을 갖춘 스마트폰과 같은 기기의 도움을 받는 '하이브리드 방식'이다. 즉 스마트 카 구현을 위한 접근법은 애플리케이션이 차량, 클라우드, 스마트폰 중 어디에 있는지, 커넥티비티는 자동차, 스마트폰 중 누가 담당하는지에 따라 구분된다 할 수 있다. 스마트 카 구현을 위한 주요 기능은 크게 4가지 부문으로 구성된다. 자동차에는 주로 커넥티비티, HMI 컨트롤, 애플리케이션이 들어간다. 커넥티비티 부분에는 USB, 블루투스, WiFi, MHL 등이 있다. 스마트폰에는 커넥티비티, HMI 컨트롤, 애플리케이션 솔루션이 있다. HMI 컨트롤 요소만 보면 화면, 터치스크린, 버튼과 스위치, 음성인식과 출력 등이 있다.

임베디드 앱 플랫폼(embedded app platform)인 전자의 경우 자동차가 하드웨어, 커넥티비티, 미들웨어 OS, 미들웨어, 애플리케이션 모두를 구현한다. 커넥티비티 부문에서 모듈을 임베디드 시키고 유저 SIM을 넣기도 한다. 이같은 시스템은 스마트폰의 대중화, 애플리케이션 마켓 활성화, 다양한 앱의 탄생한 소비가전 시장의 변화에 대응 중이다. 예를 들면 뭔가 부족함은 있지만 자체적인 앱 스토어를 구축하기 시작한 스마트TV와 비교할 수 있다. OS는 GENIVI, 안드로이드, MS AUTO 등이 있지만 오픈, 표준화 경향이 강하게 나타나고 있다. 전반적으로 볼 때 인포테인먼트 시스템의 핵심이 애플리케이션이기 때문에 자체적인 마켓, 에코시스템을 구현하는 방향으로 진행되고 있다.

비용 큰 임베디드 앱

하이브리드는 인포테인먼트 등 다양한 기능을 스마트폰의 힘을 빌려 차량 시스템에 통합하는 방식이다. 대표적 시스템으로는 토요타의 엔튠(Entune), BMW의 커넥티드드라이브(Connected Drive) 등이 있다. 스마트폰은 다양한 장점을 지니고 있다. 프로세싱 파워가 좋고, 외부와의 연결성이 이미 구현돼 있으며 사용자 경험도 풍부하다. 그러나 폰을 차에 끌어들이는 것은 장점보다 많은 단점을 지니기도 한다. 예를 들자면 운전부주의 유발 등의 문제다. 때문에 차내에서 스마트폰의 기능을 안전하게 사용할 수 있도록 하는 다양한 솔루션이 개발되고 있다.

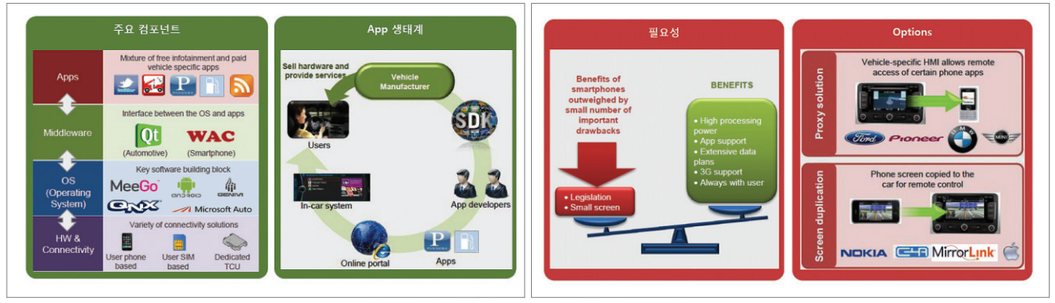
애플리케이션을 헤드유닛에 임베디드한 시스템은 기본적으로 애플리케이션, HMI 컨트롤 모두를 헤드유닛과 클러스터를 통해 한다. 커넥티비티에 따라 이미 보편화된 2가지 모델이 있는데 하나는 차량 내 자체 통신 모듈을 이용하는 카 커넥티드 방식이고, 다른 하나는 스마트폰 테더링(tethering)을 통한 드라이버 커넥티드 방식이다. 이같은 시스템의 장점은 외부 단말에 의존하지 않기 때문에 안전과 보안성이 높다는 점이다. 제조사는 HMI 차별화 등을 통해 브랜드를 강화하는데 이용할 수 있고 기존의 텔레매틱스 서비스나 최근 유럽에서 의무화된 eCall 등의 결합도 용이하다.

단점은 자체 플랫폼과 애플리케이션을 만들어야하기 때문에 비용이 많이 들고, 소비가전 시장에 익숙해진 사용자의 니즈, 애플리케이션의 라이프사이클에 신속히 대응해야한다는 부담이 있다. 통신 모듈이 둘이기 때문에 사용자의 비용 부담 문제도 고려해야한다.

아이팟 아웃

현재 오픈 플랫폼으로 인해 특정 애플리케이션이 전체 시스템을 망가뜨리는 상황 대응을 위해 가상화 기술 제공이 모색되고 있고, 하이브리드 방식의 일부 기능들을 끌어들이려 하고 있다.

스마트폰 애플리케이션을 기반(application

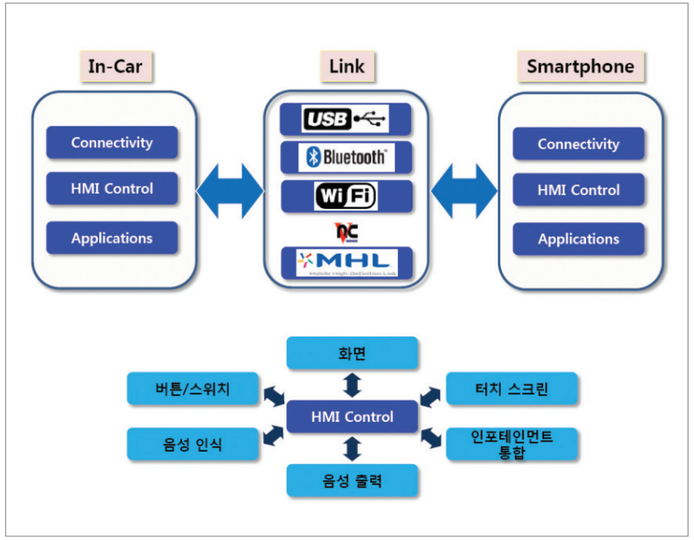


임베디드 앱 플랫폼(左)과 스마트폰 기반 시스템

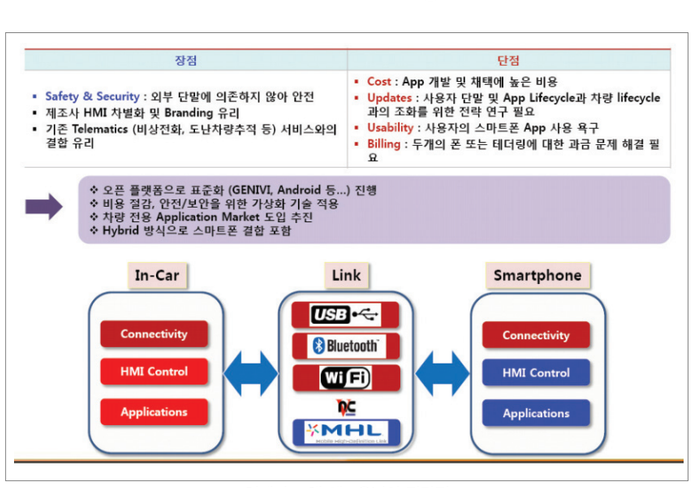
based in smartphone)으로 하는 시스템은 애플리케이션과 커넥티비티가 스마트폰에 있고 HMI만 차량에 구현된 타입이다.

애플의 아이폰, 아이패드, 아이팟을 연결하는 아이팟 아웃(iPod Out) 솔루션, 터미널모드(Terminal Mode)로 불렸던 카커넥티비티 컨소시엄(Car Connectivity Consortium, CCC)의 미러링크(MirrorLink) 등이 대표적이다. 또 대부분 미러링크의 업데이트에 포함되고 있지만, 스마트폰의 HMI 복제 형태가 아닌 헤드유닛의 HMI를 사용하면서 매우 간단한 데이터만 이용하는 판도라의 리모트스킨(Remote Skin)이나 QNX와 RIM의 Simple UI 방식도 있다. 스마트폰 기반 방식의 커넥티비티에는 HTML5, HDMI에 대응하는 마이크로 USB 인터페이스의 MHL(Mobile High-Definition Link), 인텔의 WiDi, USB 3.0 등이 도입되고 있다.

아이팟 아웃은 USB, 블루투스를 이용해 아이폰, 아이팟 터치 등을 연결하는 솔루션으로 BMW, 토요타 등이 진행하고 있다. 앱은 아이폰에서 실행되며 콘텐츠는 차량 헤드유닛이나 클러스터에서 출력된다. HMI 컨트롤은 아이팟 액세서리를 사용해 아이폰에 전달한다. 장점은 다수의 아이폰 사용자에게 어필할 수 있다는 점, 애플의 지속적 업데이트 지원 등이다. 단점은 애플 전용이고, 인터페이스를 위한 개당 4달러 정도의 라이센스 칩이 필요하다는 점이다.



스마트 카 구현을 위한 주요 기능 구분



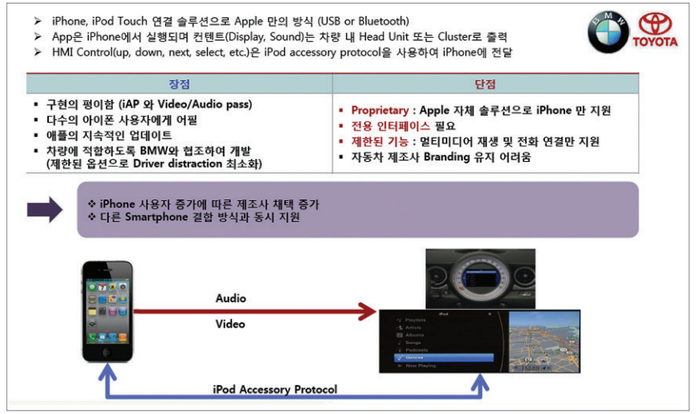
임베디드 앱 헤드유닛의 장단점

미러링크

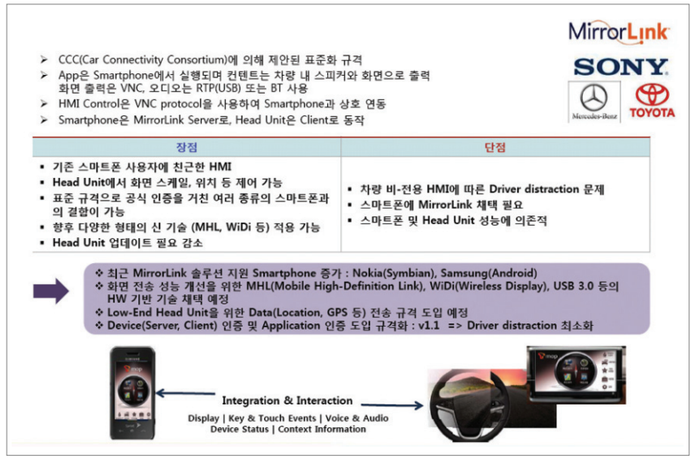
미러링크는 본래 노키아와 CE4A의 터미널 모드로 출발했다가 CCC에 의해 새롭게 제안된 표준화 규격이다. 현재 1.1 규격이 릴리스됐다. 앱이 스마트폰에서 실행되면서 화면과 소리가 차량의 헤드유닛에 전달된다. HMI 컨트롤은 차량에서 담당해 스마트폰으로 입력 사항이 전달돼 폰의 애플리케이션을 구동하는 방식이다. 스마트폰이 미러링크의 서버, 헤드유닛이 클라이언트 역할을 하는 셈이다.

미러링크가 확산되기 위해서는 필수적으로 스마트폰 제조사가 미러링크 기능을 탑재해야만 한다. 다행히 CCC에는 삼성, LG, HTC, 소니 등의 폰 메이커들이 가입해 있고, 최근에 삼성전자와 토요타가 미러링크 솔루션을 발표하기도 하는 등 저변이 확대되고 있다. 토요타 터치 라이프(Touch Life), 알파인 ICS-X8, 소니 카 엔터테인먼트 시스템 등이 대표적 시스템이고 메르세데스 벤츠 등 다양한 메이커들이 이에 대응하고 있다.

장점은 기존 스마트폰 사용자에게 HMI가 친근하고 헤드유닛에서 화면 스케일, 위치 등의 제어가 가능하다는 점이다. 또 표준 규격으로 공식 인증된 다양한 폰의 결합이 가능하다.



▲ 아이팟 아웃 ▼ 미러링크



화면 전송은 밴드위스를 필요로 하는데 현재 VNC 프로토콜을 이용되고 있지만 향후 MHL, WiDi, USB 3.0 등의 적용을 통해 HD급 화면 전송이 가능해질 전망이다. 헤드유닛의 업데이트 필요성이 낮고, 저사양 헤드유닛에서도 화면 없이 단순한 텍스트 형태의 데이터 전송을 통해 턴바이턴 내비게이션과 같은 간단한 기능 구현도 가능하다.

스마트폰 기반이 대세

임베디드 앱 플랫폼은 향후 플랫폼의 오픈 표준화가 지속될 것이고, 커넥티드 기능이 발전되고 있다. 또 BMW 미니와 같이 클러스터의 디지털화를 동반할 것이다. 비용 절감, 보안을 위해 가상화 기술도 적용될 전망이다.

스마트폰 애플리케이션 기반 시스템은 미러링크 등의 규격화와 애플의 아이폰 아웃 등이 동시 진행될 것이며, 운전부주의 저감 능력이 더욱 발전할 것이다. 성능은 MHL, WiDi, USB 3.0 도입으로 더욱 높아질 것이다. 개인적으로 스마트폰 애플리케이션 기반 시스템이 인포테인먼트 시스템의 대세가 될 것으로 본다.

MDS, 인포테인먼트 솔루션 세미나 개최

MDS테크놀로지가 지난 3월 양재동 엘타워에서 자동차 인포테인먼트 환경 구축을 위한 솔루션 적용사례와 업계 기술동향을 소개하는 '자동차 인포테인먼트 솔루션 세미나'를 개최했다. 세미나에는 국내 카 메이커, 인포테인먼트 시스템 개발사 관계자 100여명이 참석해 높은 관심을 보였다.

스마트폰-인포테인먼트 연결 솔루션을 비롯해 가상화 기술 기반의 ECU 통합 솔루션, 모델기반 HMI 개발 솔루션 등 인포테인먼트 시스템 개발에 필요한 핵심 솔루션들과 최신 업계 기술동향과 국내외 완성차 및 IT업계 간의 다양한 협업사례가 소개됐다. MDS테크놀로지는 스마트폰-자동차 인포테인먼트 시스템을 연결하는 글로벌 표준인 미러링크(MirrorLink)를 기반으로 자체 개발한 '네오링크(NeoLink)'를 선보였다. 네오링크는 별도의 애플리케이션 개발 없이 스마트폰의 다양한 자원을 자동차에서 활용할 수 있게 한다.

MDS테크놀로지는 차량용 인포테인먼트에서 ECU 통합에 이르는 자동차 개발 핵심 솔루션과 개발 컨설팅 외에도 자동차 SW 전문 교육을 제공하고 있다. 지난해 매출액 611억 원 중 자동차 분야 매출이 전년 대비 62% 성장한 140억 원이었다. ❖