MOST 네트워크 기반의 오디오 플레이어 기능의 설계 및 구현

김태환*, 전영준*, 장시웅*
*동의대학교

Design and Implementation of Audio Player Functions based on MOST Network

Tae-hwan Kim*, Young-joon Jeon*, Si-woong Jang*
*Dongeui University
E-mail : cc2208@lycos.co.kr, biggood@deu.ac.kr, swjang@deu.ac.kr

요 약

최근 차량의 멀티미디어 장치들이 증가하면서 이 장치들을 네트워크로 연결하여 멀티미디어 데이터를 송 수신해서 사용할 수 있는 MOST 네트워크를 적용한 차량들이 늘어나고 있다. 본 논문에서는 MOST 네트워크를 적용한 장치들을 제어하기 위한 HMI를 구현하는 방법에 대하여 연구하였다. MOST는 네트워크 전체를 관리하는 하나의 마스터와 하나 이상의 슬레이브 장치들로 구성된다. 또한 MOST 네트워크에 연결된 장치를 제어하는 컨트롤 인터페이스가 있는데 이를 HMI라고 한다. 일반적으로 HMI는 마스터 역할을 하면서 네트워크의 각 각 장치들을 제어한다. 멀티미디어 장치 중 차량에서 일반적으로 많이 사용되는 장치가 오디오 장치이며 MOST 네트워크에서 사용되는 대표적인 장치이기도 하다. 본 논문에서는 MOST 네트워크의 대표적인 장치인 오디오 장치를 제어하는 HMI 인터페이스 프로그램을 구현하였으며, MOST 오디오 장치를 네트워크로 연결하여 구현한 프로그램을 테스트한 결과 정상적으로 동작함을 확인하였다.

키워드
MOST, 오디오 인터페이스, MOST NetServices, HMI

Ⅰ. 서 론

자동차 기술의 추세는 IT 기술의 발전으로 자동차의 전장제품의 비중이 급격히 증가하고 있으며, 모든 제어는 자동차 제어 네트워크를 통하여 이루어지므로 기존에 다 대 다 형태의 케이블이 제거되면서 간편한 형태로 디바이스 간의 연결이 가능해졌으며, 차량네트워크를 이용한 제어 및 데이터 전송이 이루어지는 형태로 발전하고 있다. 최근 차량의 멀티미디어 시스템을 연결하는 것에 MOST 네트워크 시스템을 적용하여, EMI 문제를 해결하고 와이어 하니스 무게와 부피를 줄이며 실시간으로 영상 및 신호의 송수신이 가능하도록 해주고 있다.

MOST 네트워크에서는 다중의 멀티미디어 장치들을 컨트롤할 수 있도록 해주는 HMI (Human Machine Interface)가 필요하다. 이 HMI가 한 노드의 마스터가 되어 다중의 슬레이브 장치들을 컨트롤할 수 있게 된다. 멀티미디어 장치 중 차량에서 일반적으로 많이 사용되는 장치가 오디오 장치이며 MOST 네트워크에서 사용되는 장치 중에 가장 대표적인 장치이기도 하다 [1]. 본 논문에서는 MOST 네트워크를 기반으로 동작하는 오디오 장치를 제어하기 위한 HMI 인터페이스 응용 프로그램을 구현하였다.

Ⅱ. 관련 연구

2.1 MOST NetSevices API

MOST NetServices는 MOST가 접목된 기기장치의 개발 시간을 단축시킬 수 있도록 네트워크 서비스를 표준으로 구현한 API이다[2]. INIC과

EHC간의 통신 및 네트워크 관리를 위한 라이브러리를 포함하고 있으며 이에 관련된 접근을 메시지 기반으로 처리할 수 있도록 도와주는 역할을 수행한다[2]. 넷서비스는 Layer1, Layer2, MOST High API, Packetizer API로 구성된다. Layer1에서는 INIC과 Application 사이의 Interface를 제공해주며, Layer2에서는 Layer1과 MOST 디바이스의 연결을 제공한다. 그리고 부가적인 보안 계층을 포함해 데이터 패킷 송수신을 위한 대안의 전송 경로를 제공해주는 MOST High API, 제어 채널의 부하를 줄여 주어서 어플리케이션 메시지를 위한 전송 루트를 제공하는 Packetizer 서비스로 구성된다.

2.2 MOST 구조

MOST는 링 구조로 되어 있어 하나의 노드로 다중의 디바이스를 컨트롤하고 제어할 수 있다[3]. 그림 1은 MOST의 링 구조를 보여준다.

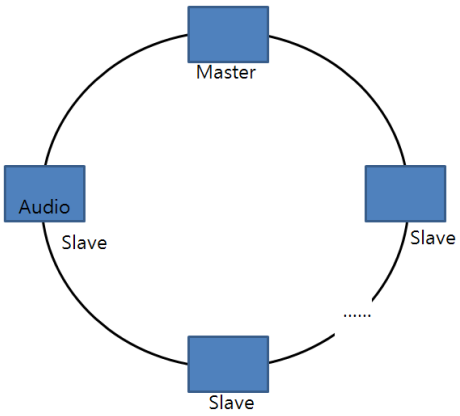

그림 1. MOST 구조

MOST의 링 구조는 하나의 Master에 여러 개의 Slave로 구성되어 있다. 이러한 구조를 사용하고 있기 때문에 하나의 노드로 다중의 기기장치를 동작시킬 수 있는 이점을 가지고 있는 것이다.

Ⅲ. 오디오 장치 제어 인터페이스 설계

오디오 장치는 각각의 동작에 해당되는 메시지를 MOST에 연결된 장치를 제어하는 기능을 가진 HMI에서 보내 오디오를 제어한다. 메시지가 HMI로부터 오디오 플레이어로 전달이 되면 메시지에 해당되는 내용에 따라 오디오 플레이어가 동작하게 된다.

3.1 오디오 장치 제어 프로그램 주요 기능

오디오 플레이어를 컨트롤 하는 해당 메시지의 종류로 Tgt_Adr, FBlock_ID, InstID, FktID, OPType, Length, Data가 있다. 오디오 플레이어로 보내는 함수의 종류는 여러 가지가 있는데 본 오디오플레이어에 사용된 함수의 종류는 Allocate, DeAllocate, Connect, DisConnect, Volume, Mute, Temperature, SinkInfo 등이 있다.

Allocate는 오디오 장치의 입력받은 데이터를 MOST 링으로 흘려보내 주기 위하여 사용한다. 이는 재생버튼을 눌렀을 때의 동작으로 사용하였다. Connect는 오디오 장치의 입력받은 데이터를 Allocate를 통해 MOST링으로 흘려보내주면 그 데이터를 연결시켜주기 위해서 사용한다. 이 함수도 역시 Allocate와 함께 재생버튼을 눌렀을 때의 동작으로 사용함으로써 입력받은 데이터를 출력할 수 있게 된다. DeAllocate는 MOST 링으로 흐르고 있는 일련의 데이터를 흐르지 못하게 하는 함수로 정지버튼을 눌렀을 때의 동작으로 사용하였다. DisConnect는 오디오 장치와 Connect된 데이터의 연결을 끊는 역할을 하는 함수로 DisConnect와 마찬가지로 정지 버튼을 눌렀을 때의 동작으로 사용함으로써 출력되고 있는 데이터를 끊어주게 된다. Volume은 Increment, Decrement라는 OPType을 사용하여 +버튼과 -버튼에 대해서 출력되고 있는 데이터의 소리 크기를 올리거나 내리기 위해 사용하였다. Mute는 음원을 음소거 상태로 만들어주는 함수로 해당 Mute 버튼을 한번 누르면 음원이 음소거 상태가 되고 한번 더 누르면 원래의 상태가 되도록 구성하였다. Temperature는 오디오 기기장치의 온도를 체크하기 위하여 사용한다. 오디오의 상태 창에 출력함으로써 항상 온도 확인이 가능하다. SinkInfo는 Allocate시킨 음원이 오디오장치에 Connect되었을 때 할당된 채널을 확인하기 위해 사용한다.

3.2 메시지 전송

그림 2는 HMI에서 오디오 플레이어로 보내는 메시지와 다시 반환받는 메시지를 나타내었다. 오디오 플레이어를 컨트롤하기 위한 HMI의 각 버튼 마다 수행되는 내용은 다르다. 버튼을 한번 누르게 되면 해당 메시지를 오디오 플레이어에 전달이 되면서 동작하게 되고 그 결과를 다시 반환받아서 상태 창에 출력하게 된다.

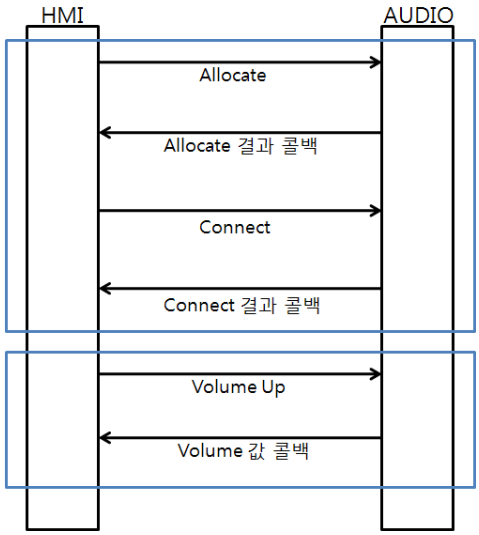

그림 2. 메시지 동작 구조

오디오 플레이어의 재생 버튼을 누르게 되면 Allocate와 Connect 함수를 동시에 수행하게 되는데 Allocate의 과정을 통해서 음원을 MOST 링으로 흘려보내주게 되며 채널을 할당하게 된다. Connect의 과정을 통해서 MOST 링으로 흐르고 있는 음원을 오디오가 출력할 수 있게 된다. Allocate와 Connect까지의 과정을 수행하게 되면 HMI와 오디오간 연결이 완료되었음을 의미하며 두 기기장치 사이의 매체인 광케이블을 통하여 제어가능 메시지들로 오디오를 컨트롤할 수 있게 된다. 즉, 소리를 높이고 줄이거나 음소거를 하는 등의 다른 기능들을 수행할 수 있게 된다.

표 1은 MOST 네트워크에 오디오 데이터를 전송하기 위한 채널을 할당하고, 그 채널에 접속하기 위한 메시지를 나타낸다. 각 메시지의 종류마다 파라미터로 들어가는 값이 다른 것을 알 수 있다. Connect의 Data[2]~Data[5]까지의 값들은 장치 간 연결을 위한 채널을 할당하는 부분으로 동적으로 할당 받는다.

표 1. 장치 간 연결 메시지 표

	Allo	DeAllo	Con	DisCon
FBlock_ID	0x24		0x22	
Inst_ID	0x00			
FktID	0x101	0x102	0x111	0x112
OPType	0x02			
Length	0x01		0x06	0x01
Data[0]	0x01			
Data[1]	–	–	0x01	–
Data[2]			동적 할당	
Data[3]				
Data[4]				
Data[5]				

표 2는 오디오 플레이어를 직접적으로 컨트롤하기 위한 메시지를 나타낸다. 각 메시지의 종류마다 파라미터로 들어가는 값이 다른 것을 알 수 있다. SinkInfo의 Data[5]~Data[8]까지의 값들은 Connect를 통해 동적으로 할당을 받은 채널의 값들이 들어가는 부분이며 채널의 상태를 출력해 줄 때 사용하는 부분이다.

표 2. 컨트롤 메시지 표

	Sink	Vol	Mute	Tem
FBlock_ID	0x22			
Inst_ID	0x00			
FktID	0x110	0x400	0x113	0xC04
OPType	0x01	0x03	0x02	0x01
		0x04		
Length	0x09	0x01	0x02	0x00
Data[0]	0x01			NULL
Data[1]	0x00	–	0x00	–
			0x01	
Data[2]	0x02		–	
Data[3]	0x02			
Data[4]	0x01			
Data[5]	동적 할당			
Data[6]				
Data[7]				
Data[8]				

Ⅳ. 오디오 플레이어 구현

본 논문에서 제안한 오디오 장치 제어를 위한 HMI의 운용 인터페이스 프로그램은 다음과 같다.

그림 3. 오디오 플레이어 컨트롤 프로그램

재생 버튼으로 Allocate, Connect 함수를 수행한다. 정지 버튼으로 DeAllocate, Connect 함수를 수행한다. 음량조절 버튼으로 Increment, Decrement함수를 수행하여 볼륨을 조절할 수 있도록 한다. 음소거 버튼으로 Mute, UnMute 함수를 수행 할 수 있도록 한다. 정보창에서는 각각의 버튼을 수행하였을 때의 동작을 출력할수 있도록 하고 SinkInfo 함수를 수행하여 해당 채널 정보를 받아서 출력 할 수 있도록 하였다. 본 논문에서 구현한 프로그램의 정상적인 동작여부를 확인하기 위하여 다음과 같이 테스트를 수행하였다.

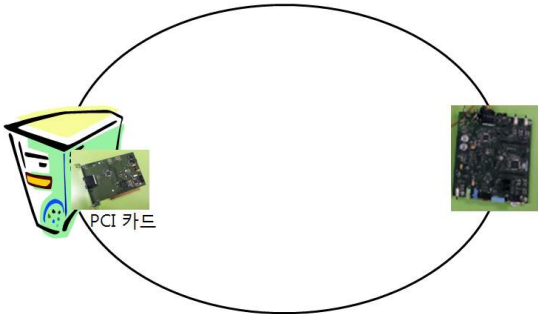

그림 4. PCI 카드와 오디오의 연결

그림 4와 같이 PC의 PCI 카드를 통해서 오디오 장치와 광케이블로 연결하였다. 이렇게 해서 HMI인 PC가 마스터가 되어 슬레이브인 오디오 장치를 대상으로 하여 구현한 프로그램을 이용하여 명령을 내리게 되고 오디오는 그 명령을 수행하고 결과를 PC에게 전달하게 된다. 테스트를 수행한 결과 오디오 장치가 정상적으로 명령에 따라 동작함을 확인하였다.

Ⅴ. 결론

본 논문에서는 MOST를 이용한 오디오 플레이어의 HMI 응용 프로그램을 구현하였다. MOST NetServices API를 이용하여 각각의 메시지 전송 방법을 이해하고 오디오 장치의 제어를 위한 인터페이스 프로그램을 구현하여 응용성을 확인하였다. 그 결과 오디오를 구동하는데 있어 다중의 케이블 사용 없이 하나의 케이블로 문제없이 동작함을 확인하였다. 향후에는 MOST를 이용한 차량에 직접 적용할 수 있도록 임베디드 기반의 HMI를 구현할 것이며, MOST150 네트워크를 기반으로 한 제어 인터페이스도 구현할 예정이다.

Acknowledgement

본 논문은 중소기업청에서 지원하는 2010년도 산학연공동기술개발사업(No.00042243)의 연구수행으로 인한 결과물임을 밝힙니다.

참고문헌

[1] SMSC OS81050 Evaluation Board, 2008
[2] SMSC MOST NetServices Layer I Wrapper for INIC V2.1.x, 2009. 01. 30
[3] Andread Grzemba, "MOST", FRANZIS, 2008. 01
[4] 정동근, “Visual C++ 윈도우 스킨 & 테마 프로그래밍 :프로젝트 따라하기”, 2009. 7. 25