



RTSP를 이용한 임베디드 하드웨어 기반 동시 방송 시스템 개발

Development of a Simultaneous Audio Broadcasting System Based on Embedded Hardware using RTSP

Raimarius Delgado* · 이승형** · 안병덕** · 최병욱*†

Raimarius Delgado, Seung-Hyung Lee, Byung-Dug Ahn and Byoung Wook Choi†

*서울과학기술대학교 전기정보공학과, ** (주)에이앤디시스템

†Dept. of Electrical and Information Engineering, Seoul National University of Science and Technology

**A&D System Co., Ltd

요약

자연 재해의 증가는 다양한 방식으로 우리의 생활 방식을 변화시키고 있다. 본 논문에서는 이러한 재난 상황에서 경보에 사용될 수 있는 임베디드 플랫폼 기반의 오디오 동시 방송 시스템을 구현하였다. 이 시스템은 네트워크에 연결된 여러 클라이언트에 오디오 신호를 전송하는 서버로 구성된다. 실제 미디어 전달은 실시간 전송 프로토콜인 RTSP(real-time streaming protocol)에 캡슐화 된 실시간 프로토콜 RTP(real-time protocol) 스트림 개념을 사용하여 구현되어 미디어 전송을 제어하고 각 구성 요소 간의 데이터 전송을 보장한다. 서버 및 클라이언트는 하드웨어 오디오 코덱을 포함한 임베디드 플랫폼으로 개발되었다. ALSA(Advanced Linux Sound Architecture)를 사용하는 장치 드라이버는 하드웨어 코덱을 구동하고 아날로그 오디오 신호와 디지털 신호의 변환을 위하여 개발되었다. 오픈 소스 라이브러리는 서버와 클라이언트에서 구현되어 RTSP와 RTP를 이용한 데이터 교환을 지원한다. 임베디드 환경에서 무 버퍼 알고리즘을 사용하여 전송 손실과 딜레이를 해결하였고, 유니캐스트 및 멀티캐스트 스트리밍 방식으로 개발된 시스템의 실용성을 검증하는 실험을 수행하였다.

키워드 : 동시방송시스템, RTSP, ALSA, 임베디드 하드웨어

Abstract

Increasing occurrence of natural disasters lead to changes in our lifestyle in various ways. This paper presents a simultaneous audio broadcasting system based on an embedded platform that can be used in disaster warning applications. The system consist of a server that streams audio signals to different clients connected within a network. The actual media delivery is implemented using real-time protocol (RTP) stream concept encapsulated into real-time streaming protocol (RTSP) to control media transmission and ensure transmission of data between each component. We develop embedded platforms, with attached hardware audio codec, as streaming server and clients. Device driver using Advanced Linux Sound Architecture (ALSA) is developed to operate the hardware codecs and convert analog audio signals to digital signals and vice versa. Open source libraries are implemented in the server and clients to support data exchange using RTSP and RTP. Transmission loss and delay were solved in the embedded environment using μ -law. Experiments are performed in unicast and multicast streaming to validate the practicality of the developed system.

Key Words : Simultaneous Broadcasting System, RTSP, ALSA, Embedded Hardware

Received: Feb. 12, 2018

Revised: Apr. 18, 2018

Accepted: Apr. 24, 2018

†Corresponding authors

bwchoi@seoultech.ac.kr

본 연구는 중소기업 산학협력 기술개발 사업(도약과제)의 지원으로 수행한 '오디오 동시방송을 위한 경보방송장치용 임베디드 MCU 플랫폼 개발' [C0443644] 과제의 성과입니다.

This is an Open-Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. 서론

지진, 홍수, 태풍과 같은 자연재해는 지구 온난화로 인해 더욱 심해지는 추세이다. 이러한 재난의 피해를 최소화하기 위해 비상경보 시스템의 중요성이 대두되고 있다. [1]. 현재 시스템에는 셀룰러 네트워크 기반의 경고 알림 서비스와 특정 셀 영역의 모든 사용자에게 경고 메시지를 전송하는 DMB(Digital Multimedia Broadcasting) 수신기가 포함되어있다 [2-3].

그러나 사용자가 셀룰러 신호가 약한 환경에서 긴급 메시지를 받지 못할 가능성이 있다. 이 문제를 해결하기 위해 인터넷 프로토콜(IP) 기반의 통신 방송 솔루션은 전 세계 사용자의 꾸준한 증가로 그 어느 때보다도 인기가 높아지고 있다[4]. 인터넷을 통한 미디어 스트리밍의 가장 일반적인

방법은 RTSP(Real-time Streaming Protocol)를 사용하는 것이다[8-10]. 스트리밍 서비스에 HTTP 를 사용할 수도 있지만, RTSP는 동시 오디오 방송에서 중요한 시작, 정지, 재생과 같은 멀티미디어 스트림 전송을 제어할 수 있다.

본 논문에서는 하드웨어 오디오 코덱인 ADAU1442 가 추가된 i.MX6S 기반 임베디드 보드를 개발하였다. 리눅스 컨텍스트에서 오디오 코덱을 제어하기 위해 ALSA (Advanced Linux Sound Architecture)를 사용하여 디바이스 드라이버를 개발하였다[11]. 스트리밍 서버는 오픈 소스인 LIVE555 스트리밍 서버를 이용하여 구현하였으며, 클라이언트는 FFmpeg 미디어 플레이어를 사용하였다. 임베디드 환경에서 발생할 수 있는 전송손실, 전송지연, 전송제한 등을 감안하여 류법칙 (μ -law) [12]으로 압축된 오디오 데이터를 유니캐스트와 멀티캐스트 전송방법으로 실험하여 본 시스템의 상용화를 검증하였다.

2 임베디드 하드웨어 플랫폼

2.1 i.MX6S 기반 하드웨어 플랫폼

i.MX6 시리즈는 NXP사에서 제작하는 ARM 아키텍처 기반의

표 1. i.MX6S 기반 하드웨어 플랫폼에 대한 사양

Table 1. Hardware Specifications of the i.MX6S-based Hardware Platform

Parts		Specifications
CPU		800MHz (up to 1GHz/ Commercial)
Ext Memory	External DDR3 x2	(256MBx16=4Gb) 512MB * 2EA=1GB
	NAND Flash x1	(512Mx8=4Gb)=512MB
	EEPROM x1	64K(8192x8)
Ethernet	LAN8720 x1ch	10/100 Base-T
	KSZ9031 x1ch	1000 Base-T
Serial	RS-232 x12ch	External Device Interface (Extend)
	RS-232 x1ch	External Device Interface
	UART x1ch	Monitoring (Debug port)
	SPI x4ch	Expander SPI to I/O Control
	I2C x4ch	Internal Device Serial Interface
Audio Sigma DSP (ADAU1442)	I2S x1ch/ I2C x1ch	Digital Audio & Control Interface
	Input x 8 Ports	I2S Input (Sigma DSP B,D)
	Output x 6 Ports	I2S Output (Sigma DSP B,D)

싱글, 듀얼, 그리고 쿼드-코어 패밀리를 포함하는 멀티코어 플랫폼이며 저전력 기반의 응용을 위한 전용 제어 오픈 하드웨어 플랫폼이다 [13-14].

본 논문에서는 i.MX6Solo Single core CPU를 사용하였으며, 클럭 속도는 800MHz(Industrial)로 1GHz(Commercial)까지 지원 가능하다. 또한 CPU를 모듈화 하였기에 제품 개발 및 적용에 있어 쉽고 기간을 단축 할 수 있는 장점이 있다. 그 외에도 4GB NAND Flash, 4Gbit DDR3 SDRAM, CPU상태를 감시하는 Watchdog Timer Reset (IMP1832), Temperature Sensor (Timp100), Power Regulator, 외부 Reset을 위해 사용하는 Reset S/W, Real Time Clock으로 I2C 제어하는 RTC 등을 제공한다.

현재 i.MX 시리즈의 제품은 자동차, 산업 및 제어 장치로 시장 전반에 적용되어 사용되고 있다. 본 연구에서 개발한 i.MX6S기반 하드웨어 플랫폼의 자세한 사양은 표 1과 같다.

2.2 ADAU1442 오디오 코덱

ADAU1442는 SigmaDSP기반 하드웨어 오디오 코덱이며 28-bit 신호처리, 필터의 안정성 보장을 위한 동기화 방법, 그리고 SigmaStudio와의 호환성과 같은 특징을 가지고 있다. ADAU1442의 작동 방법은 외부 컨트롤러를 이용하여 SPI/I2C 포트를 통해 동적으로 제어하는 방법과 모든 설정을 EEPROM에 저장하여 스스로의 부팅하는 방법, 두 가지로 나뉘어 있다.

본 논문에서는 ADAU1442를 중심으로 아날로그 스위치, 디지털 스위치, Level Detect IC, ADC, DAC로 구성된다. 오디오 코덱 (ADAU1442)의 입출력은 입력 7개, 출력 7개, GPO 12개가 있다. I2C 제어 방식으로 이루어져 있다. 그림 1은 SigmaStudio를 통해 ADAU1442 설정된 포트와 레지스터를 나타낸다.

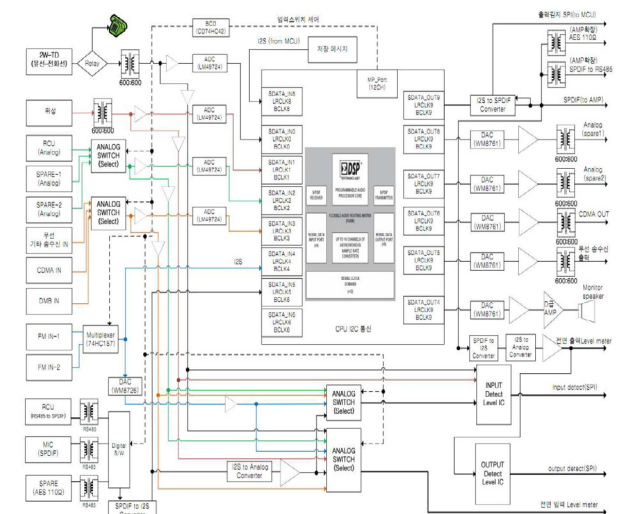


그림 1. ADAU1442 데이터 경로 설정

Fig. 1. Data Routing for ADAU1442

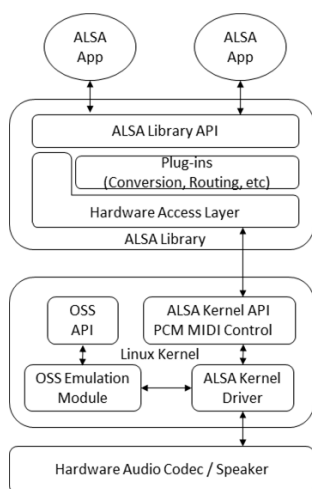


그림 2. ALSA 소프트웨어 구조

Fig. 2. Software architecture of ALSA

2.3 ALSA 기반 ADAU1442 디바이스 드라이버

ALSA는 사운드 카드용 장치 드라이버를 제공하기 위한 리눅스 커널의 구성 요소이다. ALSA 프로젝트의 목표들 가운데 하나는 사운드 카드 하드웨어를 자동으로 구성하고 시스템에 여러 개의 사운드 장치를 멋지게 관리하는 것이다. Jack Audio Connection Kit [11]과 같은 두 개의 다른 프레임워크들은 ALSA를 사용하여 낮은 레이턴시로 고급 수준의 오디오 편집과 소리 조절을 수행할 수 있게 한다. ALSA의 소프트웨어 구조는 그림 2와 같다.

ALSA 드라이버에는 두 가지 트리가 있다. 하나는 ALSA의 ftp 사이트에서 tarball 또는 cvs로 제공되는 트리이고 다른 하나는 2.6 버전 이후의 리눅스 커널 트리이다. 두 가지를 모두 동기화하기 위해 ALSA 드라이버 트리가 alsa-kernel과 alsa-driver라는 두 개의 다른 트리로 분할된다. alsa-driver에는 리눅스 커널 트리 외부에서 ALSA 드라이버를 컴파일하기 위한 많은 파일, 구형 2.2 및 2.4 커널용 래퍼 기능, 최신 커널 API 및 아직 개발 중이거나 테스트 중인 추가 드라이버가 포함되어 있다. alsa-driver 트리의 드라이버가 완료되면 커널 2.6 트리로 이동하여 정상 작동이 완료되었음을 확인하였다.

ADAU1442를 리눅스 환경에서 사용하기 위하여 ALSA 기반 디바이스 드라이버가 필요로 한다. 일반적으로 사용자 지정 보드에 내장된 사용자 지정 보드의 경우, 리눅스 기반의 하드웨어를 사용하는 장치는 플랫폼을 사용하여 장치를 설명하고 SoC에 연결된 보드 고유 구조를 가리킨다. 여기에는 사용 가능한 포트, 칩 종류, 신호 모드, 기본 초기화, 추가 핀 역할 등이 포함될 수 있다.

ADAU1442 같은 경우, I2C 통신으로 i.MX6S 기반 보드와 연결되어 있다. PCI 또는 USB 장치와 달리 I2C 장치는 하드웨어 수준에서 열거되지 않는다. 대신 소프트웨어는 각 I2C 버스 세그먼트에 어떤 장치가 연결되어 있는지, 이들 장치가 사용하고 있는 주소를

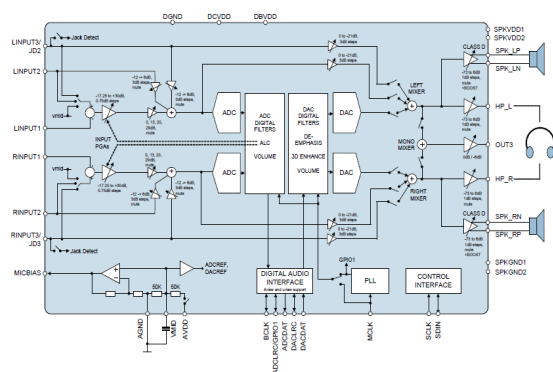


그림 3. 입출력 신호 도표

Fig. 3. Input and Output Signal Diagram

알아야 한다.

이러한 이유로 커널 코드는 I2C 장치를 명시적으로 구제화해야 하며 상황과 요구 사항에 따라 이를 달성하는 여러 가지 방법이 있다. 그러나 가장 일반적인 방법은 버스 번호로 I2C 장치를 선언하는 것이다. 이 방법은 많은 임베디드 시스템에서와 같이 I2C 버스가 시스템 버스일 때 적합하며, 각 I2C 버스는 사전에 알려진 번호를 가지고 있다. 따라서 이 버스에 있는 I2C 장치를 미리 선언 할 수 있다. 그림 3과 같이 ADAU1442에서의 I2C 통신 타이밍 다이어그램에 따라서 디바이스 드라이버 작성 시 리눅스 I2C API를 사용하였다. SigmaStudio로 설정되어 있는 각 구성요소의 레지스터 값에 따라 ALSA Kernel API를 이용하여 그림 4와 같이 입출력 신호를 처리하였다.

3. RTSP를 이용한 동시 방송시스템

3.1 RTP(Real-time Protocol)

RTP는 실시간 스트리밍 미디어, 리얼타임, 엔드 투 엔드 전송을 위해 설계되었다. 이 프로토콜은 IP 네트워크에서 전송하는 동안 일반적인 지터 보상 및 데이터의 탐지를 위한 기능을 제공한다. RTP는 IP 멀티 캐스트를 통해 여러 대상으로 데이터를 전송할 수 있다. RTP는 IP 네트워크에서 오디오/비디오 전송의 주요 표준으로 간주되며 관련 프로파일 및 페이로드 형식과 함께 사용된다.

실시간 멀티미디어 스트리밍 응용 프로그램은 적시에 정보를 전달해야 하며 이러한 목표를 달성하기 위해 일부 패킷 손실을 허용 할 수 있다. 예를 들면, 오디오 어플리케이션에서 패킷 손실은 적절한 오류 은폐 알고리즘을 사용하여 데이터의 손실이 거의 발생하지 않는다. TCP는 RTP 용도로 표준화되어 있지만, TCP는 적시성에 비해 안정성을 선호하기 때문에 RTP 애플리케이션에서는 일반적으로 사용되지 않는다. 대부분의 RTP 구현은 사용자

데이터그램 프로토콜 (UDP)을 기반으로 한다. 멀티미디어 세션을 위해 특별히 설계된 다른 전송 프로토콜인 SCTP 및 DCCP은 현재는 주로 사용되지는 않는다.

RTP는 IETF 표준 조직의 오디오/비디오 전송 작업 그룹에서 개발했다. RTP는 H.323 및 RTSP와 같은 다른 프로토콜과 함께 사용된다. RTP 표준은 한 쌍의 프로토콜인 RTP와 RTCP를 정의한다. RTP는 멀티미디어 데이터 전송에 사용되며, RTCP는 주기적으로 제어 정보 및 QoS 매개 변수를 전송하는 데 사용 된다

3.2 RTSP(Real-time Streaming Protocol)

실시간 스트리밍 프로토콜 (RTSP)은 스트리밍 미디어 서버를 제어하기 위해 엔터테인먼트 및 통신 시스템에서 사용하도록 설계된 네트워크 제어 프로토콜이다. 이 프로토콜은 엔드 포인트 간의 미디어 세션을 설정하고 제어하는 데 사용된다. 미디어 서버의 클라이언트는 서버에서 클라이언트 (주문형 비디오) 또는 클라이언트에서 서버로 스트리밍 되는 미디어의 실시간 제어를 쉽게 하기 위해 재생, 기록 및 일시중지와 같은 VCR 스타일의 명령을 실행 한다 (보이스 녹음).

스트리밍 데이터 자체의 전송은 RTSP의 작업이 아니다. 대부분의 RTSP 서버는 미디어 스트림 전송을 위해 RTCP(Real-time Control Protocol)와 함께 실시간 전송 프로토콜(RTP)을 사용한다. 그러나 일부 공급 업체는 독점적인 전송 프로토콜을 구현한다.

RTSP는 RealNetworks, Netscape 및 Columbia University에서 1996 년에 IETF에 제출된 첫 번째 초안으로 개발되었으며, 이것은 1998년 IETF (Internet Engineering Task Force)의 Multiparty Multimedia Session Control Working Group (MMUSIC WG)에 의해 표준화되었으며 RFC 2326으로 출판되었다.

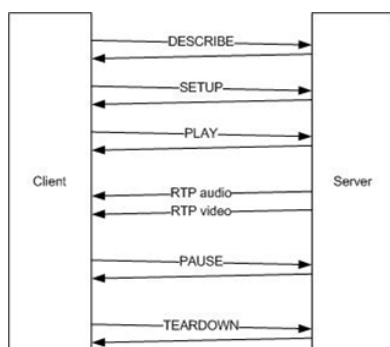


그림 4. RTSP Handshake 단계

Fig. 4. RTSP Handshake

RTSP 2.0은 2016 년에 RTSP 1.0을 대체하여 RFC 7826에 게시되었다. RTSP 2.0은 RTSP 1.0을 기반으로 하지만 기본 버전 협상 메커니즘 이외의 하위 호환성은 없다. 스트리밍 서버와 클라이언트

연결 시, 그림 4와 같은 Handshake가 이루어진다. 그림에서 나타난 바와 같이 RTSP Handshake는 DESCRIBE, SETUP, PLAY, PAUSE, 그리고 TEARDOWN으로 구분 되어 있다.

DESCRIBE 단계에서는 처리할 수 있는 응답 데이터의 유형이 포함되어 있으며 이의 응답에는 스트리밍 정보, 즉 메타데이터, 설명이 포함된다. 무엇보다 프레젠테이션 설명에는 집계된 URL 로 제어되는 미디어 스트림이 나열된다. 일반적인 경우 오디오 및 비디오 각각에 대해 하나의 미디어 스트림이 나뉘져 있다. SETUP 요청은 단일 미디어 스트림을 전송해야하는 방법을 지정하며 RTP 와 RTCP 데이터를 수신할 수 있는 로컬 포트를 포함하고 있다. PLAY 요청은 모든 미디어 스트림이 재생되며 재생 요청은 여러 개의 PLAY 요청을 보내 스택을 쌓을 수 있다. PAUSE는 일시적으로 하나 또는 모든 미디어 스트림을 중지 시키므로 나중에 PLAY 요청으로 다시 재생 할 수 있다. 마지막으로 TEARDOWN 요청은 세션을 종료 하는 데 사용 된다.

3.3 개발 환경 구축

본 논문에서의 개발환경은 그림 5와 같다. RTSP로 동시방송을 하기 위하여 서버에서는 오픈소스 라이브러리인 Live555(LGPL)를 구현하였으며 클라이언트에서는 오픈소스 라이브러리인 FFmpeg 플레이어 사용하였다.

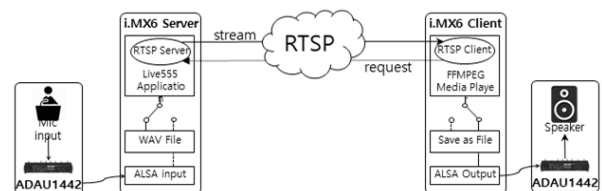


그림 5. ALSA 소프트웨어 구조

Fig. 5. Software architecture of ALSA

LIVE555 스트리밍 미디어는 멀티미디어 스트리밍을 위해 Live Networks, Inc.에서 개발 한 오픈 소스 (LGPL) C ++ 라이브러리 집합이다. 이 라이브러리는 스트리밍을 위해 RTP / RTCP 및 RTSP 와 같은 공개 표준을 지원하며 H.264, H.265, MPEG, VP8 및 DV 와 같은 비디오 RTP 페이로드 형식과 MPEG, AAC와 같은 오디오 RTP 페이로드 형식도 관리 할 수 있다. AMR, AC-3 및 Vorbis. VLC 나 mplayer와 같은 잘 알려진 소프트웨어에 의해 내부적으로 사용된다. 소프트웨어 배포에는 완전한 RTSP 서버 응용 프로그램, RTSP 클라이언트 및 RTSP 프록시 서버가 포함되어 있다. RTSP 서버, 오픈소스 코덱, 8kHz와 그 이상의 소프트웨어 코덱인 G.711, G.722, 그리고 G.726이 들어있다. 데이터 전송 방법은 멀티캐스트 또는 유니캐스트를 지원하며 WAV과 같은 오디오 파일도 스트리밍이 가능하다.

클라이언트에서는 오픈소스 라이브러리인 FFmpeg Player(LGPL)를 사용하였다. FFmpeg은 디지털 음성 스트림과 영상 스트림에 대해서 다양한 종류의 형태로 기록하고 변환하는 컴퓨터 프로그램이다. FFmpeg은 명령어를 직접 입력하는 방식으로 동작하며 여러가지 자유 소프트웨어와 오픈 소스 라이브러리로 구성되어 있다. 라이브러리 중에는 libavcodec도 들어있는데, 이 라이브러리는 음성/영상 코덱 라이브러리로 여러 프로젝트에서 쓰이고 있다. 또, libavformat라는 음성/영상 다중화, 역다중화 라이브러리도 있다.

4. 실험 및 결과

본 논문에서는 동시방송시스템의 실현 가능성을 확인하기 위하여 그림 7과 같은 환경에서 실험을 진행하였다. 스트리밍 서버는 임베디드 환경에서 구현하기 전에 일반 PC에서의 작동을 확인하였고 클라이언트로써 개발한 i.MX6S 기반 보드가 사용되었다.

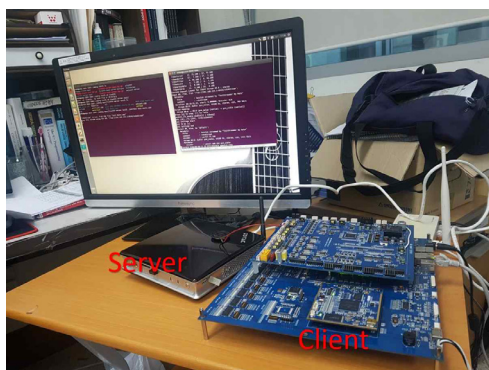


그림 6. 실험환경

Fig. 6. Experimental environment

실험은 유니캐스트와 멀티캐스트 전송방법으로 수행하였으며 그림8과 같이 Wireshark를 통해 확인하였다. 서버에서 48 kHz 샘플링 된 WAV 파일을 읽고 RTSP 스트림을 수행하였다. 그림 9에서 보는 바와 같이 클라이언트에서 받을 때 ALSA buffer xrun(underrun)과 RTP Missed Packet에 대한 문제가 발생하였다. Buffer underrun 같은 경우 두 장치 또는 프로세스 간에 통신하는 데 사용되는 버퍼에 데이터가 읽히는 속도보다 느린 속도로 데이터가 공급 될 때 발생하는 상태이다. RTP Missed Packet는 클라이언트에서 스트리밍 데이터 손실이 발생된 것을 의미한다.

이러한 문제들을 해결하기 위해 서버에서 뮤 법칙(μ -law) 압축 알고리즘을 적용하였다. 뮤 법칙은 companding 알고리즘으로 오디오 신호의 동적 범위를 줄인다. 아날로그 시스템에서 전송 중에 달성되는 신호 대 잡음비(SNR)를 높일 수 있다. 디지털 영역에서는

No.	Time	Source	Destination	Protocol	Length	Info
6	25.417995000	192.168.153.6	192.168.153.2	RTSP	159	OPTIONAL rtsp://192.168.153.2
7	25.418019000	192.168.153.6	192.168.153.2	TCP	66	9554-9551 [ACK] Seq=1 A
8	25.418240000	192.168.153.2	192.168.153.6	RTSP	218	Reply: RTSP/1.0 200 OK
9	25.418319000	192.168.153.6	192.168.153.2	TCP	66	95521-4554 [ACK] Seq=94 A
10	25.419229000	192.168.153.6	192.168.153.2	RTSP	185	DESCRIBE rtsp://192.168.153.2
11	25.419330000	192.168.153.2	192.168.153.6	RTSP/SDP	633	Reply: RTSP/1.0 200 OK
12	25.421469000	192.168.153.6	192.168.153.2	RTSP	212	SETUP rtsp://192.168.153.2
13	25.421734000	192.168.153.2	192.168.153.6	RTSP	256	Reply: RTSP/1.0 200 OK
14	25.423044000	192.168.153.6	192.168.153.2	RTSP	195	PLAY rtsp://192.168.153.2
15	25.423244000	192.168.153.2	192.168.153.6	TCP	70	TCP segment of a retrans
16	25.423341000	192.168.153.2	192.168.153.6	RTCP	1514	Sender Report Source d
17	25.423672000	192.168.153.6	192.168.153.2	TCP	66	95521-4554 [ACK] Seq=488
18	25.423721000	192.168.153.2	192.168.153.6	RTSP	1514	PTDynamicRTP-Type=96, S
19	25.447505000	192.168.153.2	192.168.153.6	TCP	1514	PTDynamicRTP-Type=96, S
20	25.452208000	192.168.153.6	192.168.153.2	TCP	66	95521-4554 [ACK] Seq=488
21	25.452255000	192.168.153.2	192.168.153.6	RTCP	1450	PTDynamicRTP-Type=96, S
22	25.479020000	192.168.153.2	192.168.153.6	TCP	1514	PTDynamicRTP-Type=96, S
23	25.479309000	192.168.153.6	192.168.153.2	RTCP	66	95521-4554 [ACK] Seq=488

(a)

No.	Time	Source	Destination	Protocol	Length	Info
16	6.160642000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
17	6.170570000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
18	6.192360000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
19	6.208233000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
20	6.224111000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
21	6.255510000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
22	6.271731000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
23	6.287585000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
24	6.303473000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
25	6.319330000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
26	6.335204000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
27	6.351068000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
28	6.366932000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
29	6.382796000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
30	6.398660000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des
31	6.414524000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Des

(b)

그림 7. 오디오 스트리밍 방법: (a) 유니캐스트 (b) 멀티캐스트
Fig. 7. Audio streaming method: (a)Unicast (b)Multicast

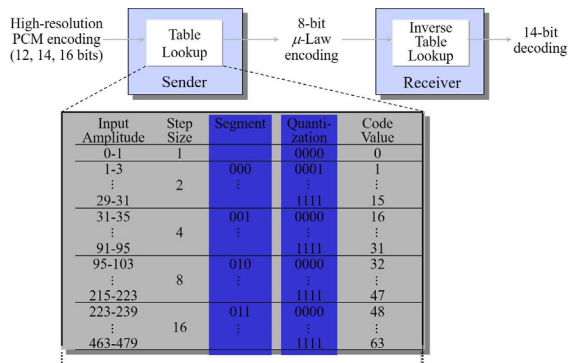
1	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
2	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
3	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
4	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
5	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
6	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
7	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
8	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
9	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
10	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
11	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
12	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
13	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
14	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
15	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
16	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
17	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
18	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
19	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
20	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
21	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
22	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
23	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
24	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
25	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
26	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
27	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
28	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
29	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
30	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
31	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
32	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
33	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
34	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
35	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
36	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
37	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
38	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
39	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
40	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
41	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
42	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
43	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
44	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
45	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
46	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
47	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
48	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
49	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
50	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
51	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
52	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
53	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
54	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
55	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
56	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
57	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
58	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
59	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
60	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
61	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
62	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
63	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
64	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
65	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
66	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
67	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
68	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
69	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
70	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
71	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
72	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
73	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
74	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
75	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
76	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
77	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
78	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
79	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
80	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
81	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
82	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
83	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
84	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
85	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222
86	0.000000	192.168.153.2	232.50.249.30	ENIP	1454	Source port: 2222 Destination port: 2222

ALSA Buffer

xrun

RTP Missed

Packet

그림 9. μ -law 압축 알고리즘Fig. 9. μ -law compression

```

embd@embd-ZBOX-ID84:~
$ ld/lib --extra-libs='ld -lphresSI TX DC (1)
ad -ln -lasound'
libavutil 55.100.104 / 55.100.104
libavcodec 57.75.100 / 57.75.100
libavformat 57.7.100 / 57.7.100
libavdevice 57.7.100 / 57.7.100
libswresample 2.8.100 / 2.8.100
libpostproc 54.6.100 / 54.6.100
GUESSED CHANNEL LAYOUT for Input Stream #0:0: stereo
Input #0, rtsp, from 'rtsp://192.168.153.2:8554/rainstream':
Metadata:
  title           : Session streamed by "TestStream by Rain"
  comment         : rainStream
  duration        : 00:10:31.12, start: 0.000000, bitrate: N/A
  Stream #0:0: Audio: pcm_mulaw, 44100 Hz, stereo, s16, 705 kb/s
Stream mapping:
  Stream #0:0 -> #0:0 (pcm_mulaw (native) -> pcm_s16le (native))
Press [q] to stop, [?] for help
Output #0, also, to 'default':
Metadata:
  title           : Session streamed by "TestStream by Rain"
  comment         : rainStream
  encoder         : Lavf57.75.100
  Stream #0:0: Audio: pcm_s16le, 44100 Hz, stereo, s16, 1411 kb/s
Metadata:
  encoder         : Lavc57.100.104 pcm_s16le
size=0K time=00:00:09.34 bitrate=N/A speed=0.00x

```

그림 10. μ -law 법칙을 이용한 오디오 스트리밍Fig. 10. Audio streaming using μ -law compression

5. 결론

본 논문에서는 RTSP를 이용한 동시방송 소프트웨어 솔루션과 이를 위한 임베디드 하드웨어를 개발하였다. 하드웨어 오디오 코덱인 ADAU1442를 리눅스 환경에서 사용하기 위하여 ALSA 구조를 채택하여 오디오 스트리밍을 통한 동시방송 모듈을 개발하였다.

본 연구 결과는 향후 상품화 과정을 통하여 재난 동시 정보 시스템으로 활용될 예정이다.

References

- [1] Soyoung Ahn, Inchan Jeon, Jihee Kim, Yong Tae Lee and Seong Jong Choi, "A Study on Internet Emergency Alert Distribution System Adaptive to the Receiver Characteristics," *Journal of Broadcast Engineering*, Vol. 20, No. 4, pp. 598-605, Jul, 2015.
- [2] Mihwa Song, Kyungho Jun, and Sekchin Chang, "An Efficient Multiplexing Method of T-DMB and Cell Broadcast Service," *IEEE Transactions on Consumer Electronics*, Vol. 60, No. 4, pp. 549-557, Nov, 2014.
- [3] Hyun-Ho Choi, Hyung Lee and Kwan-Jong Yoo, "Implementation of System for Emergency Alert Broadcasting Service in Local Area," *Journal of the Korea Contents Association*, Vol. 13, No. 8, pp. 20-27, Aug. 2013.
- [4] Andrea Tanner, Daniela B. Friedman, Alexis Koskan and Daphney Barr, "Disaster Communication on the Internet: A Focus on Mobilizing Information," *Journal of Health Communication*, Vol. 14, No. 8, pp. 741-755, Dec. 2009.
- [5] Jae-Uk Ryu and Kwang-Baek Kim, "Recognition System of Passports by Using Enhanced Fuzzy Neural Networks," *Journal of Korean Institute of Intelligent Systems*, Vol. 13, No. 2, pp. 155-161, Sep, 2009.
- [6] Sumi Helal, Raja Bose, Shinyoung Lim and Hyun Kim, "Networked Robots using ATLAS Service-Oriented Architecture in the Smart Spaces," *International Journal of Fuzzy Logic and Intelligent Systems*, Vol. 8, No. 4, pp. 288-298, Dec. 2008.
- [7] Hyunsoo Lee, "Embedded System Framework and Its Implementation for Device-to-Device Intelligent Communication of Manufacturing IoT Device considering Smart Factory," *Journal of Korean Institute of Intelligent Systems*, Vol. 27, No. 5, pp. 459-465, Oct. 2017.
- [8] Yeongyun Kim and Dong Hun Kim, "Implementation of a Smartphone Interface for a Personal Mobility System Using a Magnetic Compass Sensor and Wireless Communication," *Journal of Korean Institute of Intelligent Systems*, Vol. 25, No. 1, pp. 48-56, Feb. 2015.
- [9] Jaegol Yim and Gyeyoung Lee, "Implementation of a Prototype Personal Live Broadcasting System," *International Journal of Software Engineering and Its Applications*, Vol. 7, No. 4, pp. 295-304, Jul. 2013.
- [10] Sung-Won Lee, Je-Hun Yu and Kwee-Bo Sim, "Real-time Streaming and Remote Control for the Smart Door-Lock System based on Internet of Things," *Journal of Korean Institute of Intelligent Systems*, Vol. 25, No. 6, pp. 565-570, Dec. 2015.
- [11] J. Newmarch, *Linux Sound Programming*, Apress, 2017.
- [12] A. Dodd, *The Essential Guide to Telecommunications*, Prentice Hall, 2012.

- [13] Raimarius Delgado and Byoung Wook Choi, "Comparative Analysis of a User Space EtherCATMaster Application for Hard real-time Controllers," *International Journal of Applied Engineering Research*, Vol. 12, No. 8, pp. 1675~1682, Nov, 2017.
- [14] Young-Baek Kim, Hong-Chang Lee and Sang-Yong Rhee, "The Visual Inspection of Key Pad Parts Using a Fuzzy Binarization Algorithm," *International Journal of Fuzzy Logic and Intelligent Systems*, Vol. 11, No. 3, pp. 211-216, Sep, 2011.
- [15] Sang Min Yun, Yeon Seok Eom and Seok Jong Lee, "Properties of Intuitionistic Fuzzy Rough Sets," *Journal of Korean Institute of Intelligent Systems*, Vol. 26, No. 1, pp. 45-46, Apr, 2016.
- [16] Keun-Chang Kwak and Jeong-Woong Ryu, "Neuro-Fuzzy System and Its Application by Input Space Partition Methods," *Journal of Korean Institute of Intelligent Systems*, Vol. 8, No. 2, 433-439, Nov, 1998.

저자 소개



Raimarius Delgado

2014, 2016년 : 서울과학기술대학교 전기정보
공학과 공학사, 공학석사
2016년~현재 : 서울과학기술대학교
전기정보공학과 박사과정

관심분야 : Real-time Systems, Embedded Linux, Real-time Streaming Protocol

Phone : ※개인정보 표시제한

E-mail : raim223@seoultech.ac.kr



이승형(Seung-Hyung Lee)

1990년 : 광운대학교 전자계산기 공학석사
2000년~2007년 : 컴테크코리아 대표
2007년~2016년 : 에이앤디엔지니어링(주)
연구임원
2016년~현재 : (주)에이앤디시스템 연구임원

관심분야 : 재난 예·경보시스템, System Integration,

Phone : ※개인정보 표시제한

E-mail : gospellee@adeng.com



안병덕(Byung-Dug Ahn)

1993년 : 아주대학교 전자공학과 공학사
1996년 : 아주대학교 전자공학과 공학석사
1996~1999 : LG산전 중앙연구소
1999~2004 : (주)필테라
2004~2007 : (주)퍼펙트아이앤씨

2007~2016 : 에이앤디엔지니어링(주)

2016~ 현재 : (주)에이앤디시스템

관심분야 : Embedded System, Real-Time OS, Digital Audio Transmission

Phone : +82-2-2163-5361

E-mail : andy@adeng.co.kr



최병욱(Byoung Wook Choi)

1986년 : 한국항공대학교 항공전자공학공학사
1988년, 1992년 : 한국과학기술원 전기·
전자공학 공학석사, 공학박사
1988년~2000년 : LG산전 중앙연구소,
책임연구원

2000년~2005년 : 선문대학교 제어계측공학과 부교수

2003년~2005년 : 임베디드웹 대표이사

2007년~2008년 : Nanyang Technological University, Senior Fellow

2005년~현재 : 서울과학기술대학교 전기정보공학과 교수

관심분야 : Real-time embedded systems design, Embedded Linux, Software Platform

Phone : ※개인정보 표시제한

E-mail : bwchoi@seoultech.ac.kr