Embedded Linux 시스템 설계 및 구현에 관한 연구

유임종* · 고성찬*
*안동대학교

A Development and Design of Embedded Linux System

Im-jong yoo* · Sung-chan Ko
*Andong National University

요 약

본 논문에서는 실시간 운영체제하에 가전제품 및 간단한 통신모듈로 최근 많이 응용되고 있는 StrongArm SA1110을 메인 CPU로 하여 정보통신 분야에 적용될 수 있는 VoIP분야 중 RTP를 통한 음성데이터의 통신 Embedded Linux 시스템의 설계 및 구현에 관한 내용을 언급한다.

본 논문에서는 하드웨어 측면으로 임베디드 리눅스 CPU 개발 Toolkit인 타이눅스 박스II를 이용하여 VoIP 시스템을 구성하였으며, VoIP 소프트웨어 처리부분으로는 실시간 전송 프로토콜인 RTP를 이용해 설계 구현하였다. 본 논문의 개발환경은 타겟보드와 개발용 리눅스 PC간 연결 인터페이스를 위한 RS-232C의 직렬 접속, USB 접속, Ethernet LAN 접속 연결로 구성하였다. RS-232C는 직렬 접속으로 개발용 리눅스 PC의 터미널 에뮬레이션을 위한 콘솔로 사용하였다.

VoIP 통신을 위한 환경은 ADC/DAC 변환부를 통해 아날로그 신호를 디지털로 변화시키고 PCM 압축하도록 구성되어야 하나 wave 파일로 대체하여 사용하였고, 양측간의 통신을 위한 연결 설정부는 VoIP를 위한 H.323이나 SIP에서 필요한 Gatekeeper나 Network Server를 단말 양측간 소켓통신으로 본 과정을 대체 하였다. 본 논문에서는 VoIP 시스템의 운용 중 일반적 기술에 관한 것을 언급하였고 임베디드 리눅스 개발보드를 이용하여 RTP 프로토콜의 동작하는 메커니즘을 중점적으로 기술하였다.

ABSTRACT

In this paper, which sees the Strong-ARM SA1110 it used the main CPU and RTP in VoIP system. It will be able to apply the information communication field it embodied. It used the Tynux_box2 with the hardware side and it composed a VOIP system. And it used the RTP which is a real-time protocol in software control portion. The development environment of the paper that used the Target board and a Linux PC for connection used the RS-232C, USB connection, Ethernet LAN. The VoIP the environment for a communication used the wave file in the substitution which changes analog signal with the digital signal. And For the communication of the both sides it used the socket. This paper explained the fact that against a general technique from the operation of VoIP system. Using the Embedded linux development board which explained an operational process of the RTP protocol.

키워드
Embedded Linux, VoIP, RTP, Strong Arm

I. 서 론

임베디드 시스템은 마이크로프로세서의 소형화와 및 집적화에 따라 최근에는 가전기기, CD 플레이어, PDA 등에 널리 활용되고 있다. 또한 디지털 TV등을 포함한 향후 IT 제품들은 거의 필수적으로 멀티미디어(영상, 음성, Data, 통신 등) 기능을 가져야하기 때문에 임베디드 시스템은 계속해서 발전해 나갈 전망이다.

이러한 추세에 따라 본 논문에서는 Embedded

Linux 시스템을 이용하여 Voip 시스템 구현하였고, 패킷을 실시간으로 전송할 수 있는 RTP 프로토콜을 통해 데이터의 송/수신을 구현하였다.

RTP는 IETF RFC(Requests For Comments) 1889에서 표준화한 프로토콜로 실시간 전송을 위하여 UDP(User Datagram Protocol)위에서 동작하는 프로토콜이다. 데이터 전송은 RTP를 통하여 이루어지고 단말 양측간의 연결설정을 위해서는 H.323이나 SIP를 대신하여 UDP 프로토콜 위에 RTP를 적재하여 소켓을 통한 데이터 전송을 하였다. 이러한 시스템의 개발환경을 위해 임베디드 리눅스 CPU 개발 Toolkit인 타이눅스 박스II를 이용하였다.

2장에서는 VoIP 시스템에 대한 전반적인 사항들을 언급하였고 3장에서는 RTP 프로토콜에 대해 간단히 설명하였며 4장에서는 Embedded Linux로 구현한 RTP 기능에 대하여 언급하였다.

II. 본 론

2.1 VoIP 시스템 개요

기존 데이터 전달 방법에서, 음성(Voice)은 음성망(PSTN-Public Switched Telephony Network)을 통해서 데이터(Data)는 데이터 통신망(IP-Internet)을 통해서 전달하였지만, 음성을 음성망 뿐만 아니라 데이터 통신망을 통하여 전달하고자 하는 목적에서 새롭게 등장한 기술이 VoIP이다. 즉 기존 음성망(PSTN)과 데이터통신망(IP)를 통합하는 것이다.

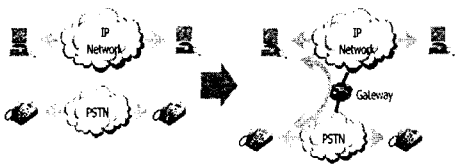

그림 2.1. VoIP의 개념

그림에서 볼 수 있듯이 PC에서 PC로 데이터 통신망을 통하여 음성통화가 가능하며 또한 PC에서 또는 전화에서 기간통신사업 교환센타에 있는 데이터통신망과 음성통신망을 게이트웨이 (VoIP Gateway)장비를 설치함으로써 음성망에 있는 음성신호가 VoIP로 변환되어 일반 전화사용자가 PC에 있는 사용자와 직접 통화가 가능하며 반대로 PC에서 VoIP 방식으로 음성신호를 보내면 게이트웨이를 통하여 음성망에 음성신호로 변환, 전달되어 일반 PC 사용자가 전화 사용자와 통화가 가능하게 된다.

2.2 음성신호의 패킷화

VoIP 통신 구성을 위해서 음성신호의 패킷화 작업이 필요하다. 네트워크(PSTN, IP)에서 데이터가 전달되기 위해서는 전달 가능한 데이터로 우선 변경한 후에 여러 가지 압축방법(G.711 PCM, G.726 등)으로 전환한 후에 IP망에서 멀티미디어 데이터 전송을 RTP(Real-time Transport Protocol)/RTCP (RTP Control Protocol)와 같은 프로토콜을 통해서 전송한다.

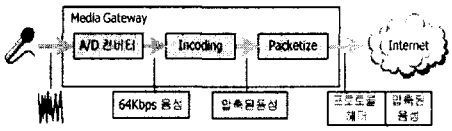

그림 2.2. 음성신호의 패킷화 과정.

2.3 연결 설정

VoIP에서는 양측 단말간 연결설정을 위해 SIP와 H.323을 이용하고 있다.

SIP는 단말간 또는 사용자들간에 기존의 VoIP 서비스뿐만 아니라 다양한 서비스의 호 설정 프로토콜이다. SIP는 매우 간단한 텍스트 기반의 응용계층 제어 프로토콜로서, 하나 이상의 참가자들이 함께 세션을 만들고, 수정하고 종료할 수 있게 한다.

SIP는 텍스트 기반의 SMTP와 HTTP 이후에 설계되었다.

H.323은 실시간 멀티미디어 데이터를 IP 기반 패킷 네트워크 망에서 교환하기 위한 프로토콜이다. 초기에는 다자간 화상회의를 목적으로 개발되었으나 VoIP 방향으로 다양하게 활용되고 있다.

2.4. Addressing/Naming & Routing

PSTN에서는 E.164(현재 ITU 주관하에 사용하는 국제전화번호체계)주소체계를 사용하고, 일반 IP 네트워크에서는 IPv4, IPv6등의 주소 체계를 사용하고 네트워크 요소를 고유하게 구별할 수 방법이 요구된다.

3. RTP 프로토콜 분석

RTP 프로토콜은 실시간 전송에 적합하다. TCP의 경우에는 안정적인 전송을 위해 여러 가지 복잡한 전송 방식을 채택하고 있는데 RTP에서는 이러한 메커니즘을 정의하고 있지 않다. 즉 이러한 메커니즘이 필요하다면 하부의 프로토콜 계층에서 이러한 작업들을 수행해 주어야 한다. 하지만, 보통 멀티미디어 응용에서는 이러한 안정성의 확보보다 낮은 부하가 더 요구되기 때문에 TCP보다 UDP를 RTP의 하부 네트워크 계층으로 사용하는 것이 보통이다.[그림. 3.1]과 같이 부하가 적은 UDP를 하부 네트워크 계층으로 채택하고, 여기에 작은 크기의 RTP헤더를 덧붙임으로써, 부하가 적

은 전송이 구현될 수 있다.

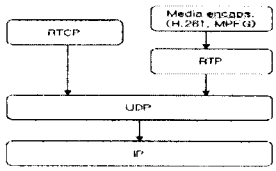

그림 3.1. UDP상의 RTP

실제 오디오와 비디오 데이터를 송수신하는 프로토콜은 RTP(Real Time Protocol)이다. RTP는 오디오와 비디오간의 동기를 맞추고 실시간성을 유지하는 데 사용되는 정보들을 포함하는 패킷 헤더라 할 수 있다. RTP 헤더의 형식은 [그림3.2]와 같으며, 각 필드의 의미는 다음과 같다.

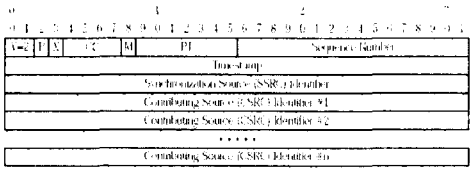

그림. 3.2 RTP 고정 헤더 형식

처음 12바이트는 모든RTP 패킷에 존재하는 반면, CSRC는 RTP믹서에 의해 추가된 경우에만 있다.

① version(V) 2 비트 : 이 필드는 RTP의 버전을 지정한다. 여기서 설명하는 RTP의 버전은 2이다.

② padding(P) 1 비트 : 이 필드가 세팅되어 있는 경우에는 그 패킷의 끝에 전송하려는 데이터 외에 추가적인 데이터들이 포함되어 있다.

③ Extension(X) 1비트: 이 필드가 세팅되어 있는 경우에는 이 RTP 헤더 뒤에 확장 헤더가 있음을 의미한다.

④ CSRC count(CC) 4비트: 이 필드는 RTP 헤더의 마지막에 CSRC가 몇 개 있는지를 지정하는 역할을 담당한다. 이 필드에 4비트가 할당되어 있으므로 하나의 RTP 패킷에 CSRC는 총 15개까지 추가될 수 있다.

⑤ Marker(M) 1비트: 이 필드의 의미는 전송되어 지는 데이터의 종류에 따라 다르다. 이 필드는 프레임들간의 경계와 같은 특별한 경우를 표시한다.

⑥ Payload Type(PT) 7비트: 이 필드는 RTP패킷에 의해 전송되고 있는 데이터의 형식을 지정한다. 응용 프로그램에서는 이 필드에 저장되어 있는 값을 통해서 그패킷에 저장되어 있는 데이터를 해석하는 방식을 결정한다.

⑦ Sequence Number 16비트: Sequence number는 하나의 RTP 패킷을 전송할 때마다 1씩 증가되는데, 수신자가 패킷 손실 유무를 검사하기 위한 용도나, 패킷들의 순서를 결정할 때에 사용되어진다. 이 값의 초기 값은 임의로 결정되는데, 이는 보안상의 이유이다.

⑧ Timestamp 32비트: Timestamp는 RTP 데이터 패킷에서 첫 번째 바이트의 샘플링 순간을 반영한다. Timestamp의 초기 값도 sequence number의 경우와 마찬가지로 무작위로 결정되어진다. 이것 역시 보안상의 이유에서이다.

⑨ SSRC 32비트: 이 필드는 RTP패킷을 만든 소스를 지정하는 역할을 담당한다. 하나의 RTP 세션 내에 동일한 SSRC 값을 가지는 서로 다른 송신 측이 있을 확률을 작게하기 위해서 이 값도 무작위로 결정한다.

⑩ CSRC 32비트: 믹서에 의해 추가되는 필드로서, 패킷에 포함되어 있는 데이터의 생산을 위해 공헌한 소스들을 지정하는 역할을 담당하고 있다.

III. Embedded Linux로 구현한 RTP

4.1 개발환경

개발환경은 타겟보드와 개발용 리눅스 PC간 연결 인터페이스를 위한 RS-232C의 직렬 접속, USB 접속, Ethernet LAN 접속 연결로 구성하였다. RS-232C는 직렬 접속으로 개발용 리눅스 PC의 터미널 에뮬레이션을 위한 콘솔로 사용하였다.

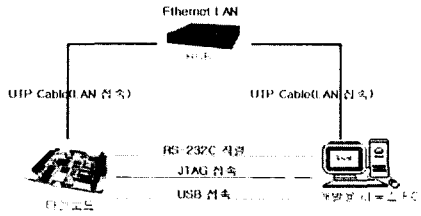

그림 4.1. Embedded Linux 개발환경

4.2 시스템 구성

VoIP 통신을 위한 환경은 ADC/DAC 변환부를 통해 아날로그 신호를 디지털로 변화시키고 PCM 압축하도록 구성되어야 하나 wave 파일로 대체하여 사용하였고, 양측간의 통신을 위한 연결 설정부는 VoIP를 위한 H.323이나 SIP에서 필요한 Gatekeeper나 Network Server를 단말 양측간 소켓통신으로 본 과정을 대체 하였다.

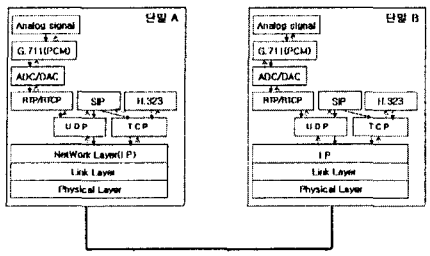

그림 4.2. 시스템 구성도

4.3 결과 분석

아래 그림은 개발용 리눅스 PC에서 타겟보드로 RTP패킷을 전송할 때 네트워크 분석기로 패킷을 검출해 분석했다.

```
00000000: 00 90 08 a6 0b cd 00 0b be 76 2c ff 08 00 45 00
00000010: 00 c8 9c 98 00 00 71 11 e7 b5 d7 c2 a5 e5 d3 f2
00000020: 07 77 5e aa 05 1e 00 b4 95 c8 80 00 96 04 6b eb
00000030: a1 76 dc e4 77 7e 7b f6 7c fa f3 ba 66 f1 7e 5c
00000040: fd 7c 69 78 77 67 68 77 7d 76 6a f9 fb 6c 6b 7b
00000050: 6d 71 6e 7a 77 6b 7a 79 69 75 ff 7a 77 6b ea 74
00000060: 68 6b 74 6b ff f6 7b 7f fb 7d f6 e3 f9 ef e8 e7
00000070: f9 ed ed ee 6f ed ef f4 f3 e7 ed ed df e7 71 ef
00000080: e7 f1 f2 e9 f4 ef e6 e7 f6 eb e5 ed fd e6 e0 f9
00000090: 7f ed e8 ef ef f6 ed fd fe ee 73 6f 7e 78 73 74
000000a0: 6d 68 77 74 6c 6d 77 fc 6d 6d 7d 77 77 fc 73 69
000000b0: 76 75 69 69 6b 67 69 6b 6a 6b 64 63 66 61 63 66 그
000000c0: 64 63 f5 59 5e 6c 6b 63 70 ff 7b 66 79 73 f5 f0
000000d0: 6d f2 f5 ff fa 76
```

그림 4.3 RTP 패킷 검출

빨간색 부분은 IP헤더를 나타내는 부분이고 검정색 부분은 UDP헤더를 나타낸다. 파란색 부분이 RTP의 헤더 부분인데 첫 번째 80의 의미는 버전넘버 ' 2' ,Padding '0' , Extension '0' ,CSRC Count '0' 임을 나타낸다. 그 뒤 00 00은 Marker '0' , Payload Type '0' 임을 표시하고, 96 04는 이 패킷의 시퀀스 넘버를 나타낸다, 6b eb a1 76은 Timestamp, dc e4 77 7e는 SSRC를 나타낸다.

IV. 결 론

본 논문에서는 VoIP용 RTP를 실시간 운영체제 지원하는 StrongArm chip 임베디드 시스템에 리눅스로 구현한 결과를 제시하였다. 실시간 특성을 가진 데이터의 전송에 적합한 프로토콜인 RTP를 UDP/IP 상에 구현하여 사용하였고. 이를 통한 음성데이터의 통신 Embedded Linux 시스템의 설계 및 구현되었다.

현재 구현된 시스템은 만들어진 데이터를 전송한 것인데 이를 마이크를 통해 실제 음성신호로 데이터화하고 압축하는 과정이 필요하다. 그리고 송수신을 처리하는 어플리케이션의 개발로 음성데이터 신호를 사용자가 들을 수 있도록 확장 가능하다.

참고 문헌

[1] H.Schulzrinne et al., "RTP: A Transport Protocol for Real-Time Applications," Internet Standard, RFC1889
[2] Embedded Linux 2002, 8, 25 이연우 · 우명찬
[3] 실시간 운영체제와 임베디드 시스템. 이두원
[4] Hyper102 SA1110 Evaluation Board Development Manual
[5] http://tynuxbox.palmpalm.com
[6] Linux Programming Bible 2002,6 권수호