홈 네트워킹 제어 미들웨어인 UPnP를 이용한 Control Point 및 내장형 시스템 상에서의 DTV와 전등 제어기 에뮬레이터 구현

Implementation of Control Point, Digital TV, and Light Controller Emulator on Embedded System Using UPnP Home Networking Control Middleware

전호인 • 경원대학교 전기정보전자공학부
Ho-In Jeon • Division of Electrical, Information, and Electronic Engineering, Kyung-Won University

요 약

본 논문에서는 인텔社의 UPnP SDK v1.0을 임베디드 리눅스 시스템 개발 보드인 아사벳(assabet)보드에 포팅하고, UPnP SDK 패키지에서 제공하는 API를 이용하여, 리눅스 PC에서 동작하는 UPnP Control Point와 임베디드 리눅스 시스템에서 동작하는 디지털 TV 에뮬레이터, 그리고 전등 제어기를 C언어로 구현하였다. 디지털 TV의 기능을 분석하여 UPnP 서비스로 설계하고, 이를 UPnP 디바이스 프로그램에 적용하였다.

본 논문에서 사용한 UPnP SDK v1.04는 UPnP 홈 네트워킹 제어 미들웨어의 핵심 프로토콜인 HTTP와 SSDP(Simple Service Discovery Protocol), SOAP(Simple Object Access Protocol), GENA(General Event Notification Architecture), 그리고, XML DOM Level-1을 리눅스에서 지원하기 위한 API들로 들로 구성되어 있다.

본 논문에서 작성한 Control Point 프로그램은 리눅스 PC에서 실행시키고, 디지털 TV 에뮬레이터 프로그램과 전등 제어기 프로그램은 임베디드 리눅스 보드에서 실행하였다. 실행된 Control Point는 네트워크에 연결된 디바이스들을 찾아 그 리스트를 콘솔에 출력하고, 디바이스가 제공하는 서비스를 콘솔입력으로 선택하여 실행시킨다. 본 논문에서 작성한 디바이스와 Control Point 프로그램이 UPnP의 핵심 기능들을 완벽하게 지원하는 것을 실험을 통해 확인하였다.

ABSTRACT

In this paper, we have implemented UPnP Devices which emulate a Control Point, a Light Controller, and a Digital TV. The Control Point has been developed on Linux host system by using C language. The UPnP Devices emulating the Digital TV and Light Controller are running on embedded linux developer board. For the development of UPnP Devices, UPnP SDK API V1.04 made by Intel Co. Ltd. has been ported on Assabet Linux Reference board

to implement the UPnP protocol. After we analyze and design some services of Digital TV device, we have applied UPnP Device program to those devices. UPnP SDK v1.04 consists of APIs which support HTTP, SSDP, SOAP, GENA and XML DOM Level-1 that are cores of UPnP protocol.

The C program written for the UPnP Control Point has been compiled and executed on Linux-based PC. The embedded system running on Embedded Linux OS has been connected all together through Ethernet which allows IP-based communications. Under this environment, the UPnP programs are being executed on each device. Control Point, when in operational mode, discovers UPnP Devices on the network and displays the device list on the consol. By selecting one of the functionalities of the device services that are displayed on the Control Point, the controllability has been accomplished. The experiment that we performed in this thesis have revealed that the Control Point and UPnP Devices have supported the protocols including SSDP, SOAP, GENA, and DHCP.

Ⅰ. 서 론

21세기는 디지털 TV 방송과 IMT-2000서비스의 개시로 유무선 홈 네트워크 [1, 2]와 인터넷을 연동하여 활용할 수 있는 인터넷 정보가전기기에 많은 관심이 집중되고 있다. 이러한 인터넷 정보가전 분야는 가정 내 기기들을 연결하기 위한 전화선, 전력선, 무선 등 홈 네트워크 통신방법에 대한 연구와 홈 네트워크에 연결되는 기기들 간에 상호 운용성을 보장하는 미들웨어 기술 및 홈 네트워크에 연결되는 단말기술이 주류를 이루고 있다.

가정 내에 홈 네트워크를 구축하여 다양한 가전기기들을 홈 네트워크에 연결하고, 외부에서 인터넷으로 이를 통제 할 수 있는 시스템을 많은 사용자는 바라고 있다. 예를 들면 외출했다가 집에 들어올 때 핸드폰을 통하여 인터넷으로 집안의 보일러나, 조명기기 등을 미리 켜고 끌 수 있거나, 자신의 방에서 욕실의 온수를 미리 준비할 수도 있다. 이를 위해 보다 다양한 기기들이 홈 네트워크에 접속될 것이고 이를 감지하고 제어하기 위한 미들웨어 기술의 중요성은 나날이 더해지고 있다.

홈 네트워킹 미들웨어 기술은 HomePNA, PLC, 이더넷, IEEE1394와 같은 유선 기술 뿐만 아니라, 블루투스나 IEEE802.11a, IEEE802.11b [2]와 같은 무선 기술을 하나로 통합하여 서로 간의 정보교환을 보장하여 주는 기술이다. 미들웨어는 가정 내의 가전기기, 정보기기 등을 모두 지원하는 기반 플랫폼이므로 집안으로 들어오는 ADSL이나 케이블모뎀 등을 통하여 인터넷에 연결되며, 미들웨어 하부의 홈 네트워크 물리 매체나 운영체계에 무관한 응용서비스를 제공해야 하고, 홈 서버를 중심으로 하여 홈 네트워크에 연결된 다양한 업체에서 개발된 기기들 간의 상호호환성을 보장하며, 홈 네트워크 자원관리 및 각 기기들을 제어할 수 있는 기능을 제공하여야 한다.

미들웨어 기술은 인터넷 정보가전기기 시장의 핵심기술로 인식되어, 미들웨어 분야에서의 승패에 따라 21세기 정보가전기기 시장에서의 우위가

결정될 수 있다고 판단한 많은 기업들이 자사의 기술을 표준으로 만들기 위해 치열한 경쟁을 벌이고 있다.

이러한 미들웨어기술 중 대표적인 기술로 마이크로소프트사가 제안한 UPnP(Universal Plug and Play)기술[3, 4]과 삼성이 제안한 VHN(Versatile Home Network), 소니사와 A/V 관련업계가 제안한 HAVi(Home Audio Video interoperability), 그리고 썬마이크로시스템사가 자바를 기반으로 만든 Jini[2]가 있다. 특히, 마이크로소프트가 제안한 UPnP는 기존에 널리 쓰이는 프로토콜인 TCP/IP와 HTTP[5], XML[6] 등을 이용하여 보다 광범위하고 쉽게 네트워크 구성에 참여할 수 있도록 하고 있다.

본 논문에서는 홈 네트워크의 주요 장비인 디지털 TV의 기능을 가진 UPnP 디바이스와 전등 제어기를 임베디드 리눅스 상에서 구현하고, PC 상에서 구현된 UPnP Control Point가 네트워크를 통해 디바이스를 제어하도록 시스템을 구성하였다. 제 2장에서는 UPnP의 기본적인 동작에 대해 설명하였으며, 제 3장에서는 Embedded Linux 시스템에 대해 간략하게 소개하였다. 본 논문의 실험 내용 및 결과는 제 4장에서 설명하였고 제 5장에서 결론을 맺었다.

Ⅱ. UPnP(Universal Plug and Play)의 동작

Microsoft의 운영체제인 Windows에 Plug and Play기능이 추가된 이후로 프린터나 각종 주변기기를 Windows PC에 추가설치하고 설정하는 일이 훨씬 간편해졌다. UPnP(Universal Plug and Play)[3, 4]는 이러한 편리함을 가정 내 네트워크 전체로 확장시켜 Gateway, Printer 및 각종 가전제품들을 쉽게 설치, 인식하고 제어할 수 있도록 설계되었다.

UPnP규격에 맞는 디바이스들은 운영체제나 제조업체가 달라도 특별한 설정이 필요하거나 특정 유틸리티를 사용할 필요가 없이 디바이스들끼리 자유로운 통신이 가능하다는 큰 장점을 가지고 있다.

UPnP를 지원하는 디바이스가 네트워크에 접속되면 자동으로 유효한 IP주소를 할당하고, UPnP 프로토콜 전용의 멀티캐스트 채널을 통해 자신의 존재와 기능을 다른 디바이스 및 Control Point에 알려주며 Control Point로부터 UPnP 메시지를 받아 외부로부터 자신의 기능을 수행하고 상태를 알려준다. 네트워크에서 빠져 나올 때에도 다른 기기에 영향을 주지 않고 쉽게 연결을 해지할 수 있다. 이러한 모든 과정이 사용자의 작업 없이 자동으로 수행되기 때문에 홈 네트워크 환경에 적합하다.

UPnP 네트워크에 연결될 수 있는 디바이스의 종류는 매우 다양하고 지금도 UPnP 포럼에서 새로운 디바이스들에 대한 논의가 계속해서 진행되고 있다. PC 뿐만 아니라 디지털 TV, 오디오 등의 AV 시스템 기기는 물론 전자레인지, 보일러, 냉장고 등의 생활 가전기기와 PDA, 스마트폰 등의 무선 인터넷 기기까지 다양한 기기들을 UPnP 네트워크에 접속할 수 있다.

이 많은 기기들은 TCP/IP라는 인터넷 표준 프로토콜을 기반으로 UPnP에 접속되기 때문에 기존의 네트워크와도 쉽게 통합될 수 있고, 예전부터 오랫동안 쓰여온 프로토콜을 기반으로 하기 때문에 기존의 문제점과 자료를 쉽게 활용할 수 있는 장점이 있다. UPnP는 사용하는 프로토콜에 의해 정의되는 표준 네트워크 아키텍처를 따르는 분산형태이기 때문에, 운영체제, 프로그래밍언어, 물리적 매체에 상관없이 독립적이다.

이러한 UPnP 디바이스 및 서비스에 대한 표준을 제정하는 모임으로 1999년 10월 18일에 UPnP Forum이 설립되었고, 이 포럼에는 가전기기제조, 컴퓨터관련, 홈 오토메이션, 모바일 디바이스 제조업체 등 200여 제조사들이 모여 있다. UPnP 포럼은 http://www.upnp.org/를 통하여 지금까지 표준화된 스키마와 디바이스, 그리고 서비스에 대한 템플릿, 정의, 도움말, 포럼의 일정과 작업 진척 상황 등을 알려주고 있다.

1. UPnP Network의 구성요소

UPnP 네트워크의 기초구성 단위는 디바이스와 서비스, 그리고 Control Point이다. UPnP 디바이스는 임베디드 디바이스와 서비스를 내장하고 있고 Control Point에게 각각의 디바이스와 서비스에 대한 디스크립션을 전달하며 네트워크를 통하여 Control Point에게 정의된 서비스를 실행할 수 있는 기능을 제공한다.

UPnP 디바이스는 서비스 및 중첩된 디바이스들의 컨테이너이다. 예를 들어 VCR 디바이스는 테이프 이동 서비스, 튜너 서비스, 그리고 시간 서비스 등으로 구성되어 있다. TV/VCR Combo 디바이스라면 서비스들 이외에 중첩된 디바이스로 구성되어 있다고 할 수 있다. UPnP 디바이스의 종류에 따라 서비스 및 내장된 디바이스의 셋트는 서로 다르다. 예를 들어 VCR내의 서비스들은 프린터의 서비스와는 다를 것이다. 따라서 각 종류별 디바이스가 제공할 서비스들은 각기 다른 워킹 그룹들이 표준화하게 된다.

이런 정보들은 모두 그 디바이스가 갖고 있어야 하는 XML 디바이스 디스크립션 문서 안에 캡쳐 된다. 디바이스 정의에는 서비스 셋 이외에도 해당 디바이스에 연결된 프로퍼티(예를 들면 디바이스 이름과 아이콘 등)들도 들어있다.

UPnP Network에서 가장 작은 컨트롤단위가 서비스이다. 서비스는 액션을 노출시키며 상태변수를 통해 현재 상태를 모델링한다. 시계 서비스를 예로 들면, 시계 서비스는 시계의 상태를 정의해주는 current_time을 상태 변수로 가지며, 사용자가 서비스를 제어할 수 있게 해주는 set_time과 get_time라는 두 개의 액션을 가질 수 있다. 이런 정보 역시 디바이스 디스크립션과 마찬가지로 UPnP 포럼이 표준화하는 XML 서비스 디스크립션의 일부분이다. 이런 서비스 정의문을 가리키는 포인터인 URL은 디바이스 디스크립션 문서 안에 포함된다. 그리고, 디바이스에는 여러 개의 서비스가 포함될 수 있다.

UPnP 디바이스 내의 서비스는 상태 테이블, 컨트롤서버, 그리고 이벤트 서버로 구성된다. 상태 테이블은 상태변수를 통해 서비스의 상태를 모델링하며 상태가 변경되면 그것을 업데이트시킨다.

컨트롤 서버는 액션요청을 받아서 (set_time 등과 같은) 해당 서비스를 실행하고, 상태 테이블을 업데이트한 후 응답을 리턴 한다. 이벤트 서버는 서비스의 상태가 바뀔 때마다 관련 구독자(Subscriber)에게 이벤트의 발생을 알리는 역할을 한다. 예를 들어 화재 경보 서비스라면 그것의 상태가 "벨 울림"으로 바뀌면 관련된 구독자들에게 그 이벤트의 정보를 전송하게 되는 것이다.

UPnP Network에서의 Control Point는 다른 디바이스들을 감지하고 제어하는 기능을 갖춘 컨트롤러이다. 다른 디바이스를 감지하면 Control Point는 다음과 같은 일을 한다.

- 디바이스 정의를 검색하여 관련 서비스들의 목록을 취한다.
- 관련 서비스들에 대한 서비스 정의를 검색한다.

- 서비스를 제어하기 위한 액션을 발생시킨다.
- 서비스의 이벤트소스를 구독한다. 이벤트 서버는 서비스의 상태가 바뀔 때마다 Control Point로 이벤트를 내보낸다.

2. UPnP 네트워킹의 단계별 구현 과정

UPnP 네트워크는 그림 1과 같이 Addressing과 Discovery, Description, Control, Eventing, Presentation의 5가지의 과정을 거쳐 구현된다. 이를 과정별로 설명하면 다음과 같다.

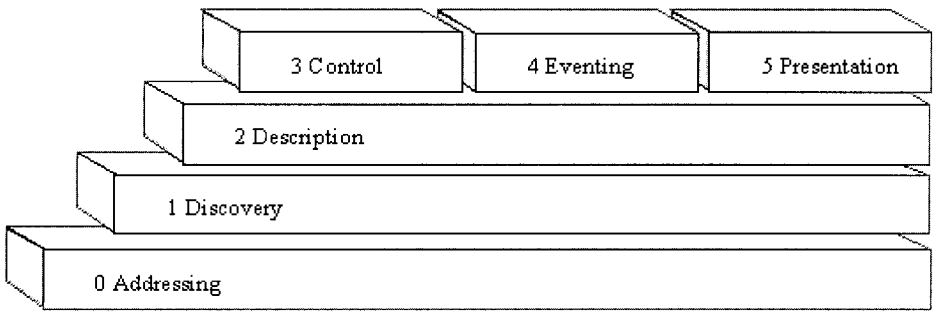
- Step 0: Control Point와 디바이스가 IP 주소를 할당받음
- Step 1: Control Point가 관심있는 디바이스를 찾는 과정
- Step 2: Control Point가 디바이스의 기능을 탐지하는 과정
- Step 3: Control Point가 디바이스에 액션을 일으키는 과정
- Step 4: Control Point가 디바이스의 상태 변화를 듣는 과정
- Step 5: Control Point가 디바이스와 디바이스의 상태를 HTML UI로 컨트롤하는 과정

UPnP 프로토콜의 각 단계는 HTTP를 기반으로 하는 기존의 여러 프로토콜들로 구성되어 있다. 첫 번째 단계인 Addressing은 IP 자동 할당 프로토콜인 DHCP로 이루어지고, 두 번째 Discovery는 SSDP[7]를, Description은 XML로 정의된 문서를 HTTP를 통해 전달한다. Control은 SOAP[8]을, Eventing은 GENA[9] 프로토콜을 사용한다. 마지막 Presentation은 HTML[10] 문서로 작성된다.

1) 주소지정(Addressing)

UPnP 네트워크의 토대는 TCP/IP 프로토콜을 이용하고 있고, 이를 위해 주소지정이 핵심이다. 모든 디바이스는 DHCP(Dynamic Host Configu- ration Protocol) 클라이언트를 갖고 있어야 하며 디바이스가 네트워크에 연결되면 제일 먼저 DHCP 서버를 찾아야 한다. DHCP 서버는 자신이 가지고 있는 IP Pool 중에서 하나의 IP를 디바이스에게 할당한다. DHCP 서버가 존재하지 않는 경우, 디바이스는 Auto IP를 이용하여 UPnP가 정한 169.254/16 Range내에서 임의의 주소를 자신의 IP로 정하고 ARP 프로토콜로 IP 충돌을 방지한다.

디바이스는 디바이스에 대해 친숙한 이름을 사



〈그림 1〉 UPnP 네트워킹 구현 단계

용하는 UPnP의 외부에 있는 더 높은 상위 레벨의 프로토콜을 구현할 수도 있다. 이 경우에는 호스트(디바이스) 이름을 IP주소로 해석해야 한다. 이 용도로 사용되는 것이 DNS(Domain Name Services)이다. 이 기능을 필요로 하거나 사용하는 디바이스에는 DNS 클라이언트가 포함될 수 있고, 자체의 이름과 주소 매핑에 동적 DNS 등록을 지원할 수도 있다.

2) 기기 발견(Discovery)

디바이스가 네트워크에 연결되고 적당한 주소가 할당되고 나면, 새로운 기기를 발견하게 된다. 이 단계는 앞서 말하였듯이 SSDP에 의해 처리된다. 디바이스가 네트워크에 연결되면 SSDP는 그 디바이스가 자신의 서비스들을 네트워크상의 Control Point에게 알릴 수 있게 해준다. Control Point가 네트워크에 연결된 경우라면 SSDP는 그 Control Point가 네트워크 상에서 관심있는 디바이스들을 검색할 수 있게 해준다.

이 두 가지 경우에 있어 핵심적으로 교환되는 것은 디바이스에 관한 핵심적인 스펙이나 그것의 서비스들 중 하나(예를 들어 디바이스 종류, 식별자, 그것의 XML 디바이스 정의 문서를 가리키는 포인터)가 들어 있는 메시지를 감지하는 것이다.

Control Point는 UPnP 네트워크에 접속하여 IP주소를 할당받은 후 자신이 관심있는 디바이스나 서비스들에게 M-SEARCH 메시지를 UPnP 멀티캐스트 채널인 239.255.255.250:1900 주소로 전송한다. Control Point로부터 ST 메시지를 받은 디바이스나 서비스는 자신의 정보와 유효기간을 담은 응답 메시지를 보낸다.

UPnP 네트워크에 접속된 디바이스나 서비스가 자신의 존재를 통보할 때는 ssdp:alive 메시지를 보내고, 자신이 사용될 수 없음을 통보할 때는 ssdp:byebye 메시지를 전송한다. 이러한 메시지는 신뢰할 수 없는 UDP상에서 보내지기 때문에 한번 이상 메시지를 발송한다.

3) 표현(Description)

UPnP 디바이스가 SSDP를 통해 Control Point에게 전달하는 내용은 자신이 어떤 종류의 디바이스라는 간단한 정보이다. Control Point는 디바이스를 제어하기 위해 Discovery 메시지를 통해 전달된 Description URL 주소로부터 XML 문서를 다운로드한다. XML 문서에는 모델이름과 번호, 시리얼 번호, 제조업체 이름, 제조사 지정 웹사이트의 URL 등과 같은 제조 업체 고유의 정보가 들어 있다. 또한, 이 문서에는 디바이스가 제공하는 서비스의 리스트를 포함한다. 서비스 목록의 URL을 통해 서비스의 명령, 액션, 파라미터나 인자 등의 상세한 정보를 담은 XML 문서를 다시 다운로드하고 컨트롤, 이벤트 정보를 분석하여 디바이스를 제어하고 관리한다.

4) 컨트롤(Control)

Control Point는 디바이스를 제어하기 위해 디바이스의 서비스에게 액션 요청을 전송한다. 이렇게 하기 위해 Control Point는 서비스에 대한 컨트롤 URL로 적절한 컨트롤 메시지를 전송한다. 이 컨트롤 메시지 역시 SOAP를 이용한 XML로 표현된다. 액션을 발생시키는 것은 일종의 원격 프로시저 호출(Remote Procedure Call)이라고 할 수 있다. 즉 Control Point는 디바이스의 서비스로 액션을 전송하고 액션이 완료(혹은 실패하면)되면 서비스는 그 값이나 에러를 리턴해 준다.

또한 Control Point는 상태 변수들의 값을 폴

링할 수도 있다. 액션의 효과는 서비스의 상태 변수에 있는 변화를 기준으로 모니터링될 수 있다. 이러한 상태 변수의 변화는 이벤트에 정의되어 있는 것에 따라 관련된 모든 Control Point로 통보되지만 이 상태 변수들의 값은 컨트롤 요청의 일종인 쿼리가 될 수 있다. 컨트롤 메시지에 들어 있는 제조사 고유의 정보들 중에는 인자 값들도 있다.

UPnP 포럼 위원회는 액션이름, 인자이름, 그리고 메시지에 들어 있는 변수들을 정의하고 있다. 이런 정보들은 UPnP 고유 포맷으로 캡슐화되며 SOAP로 포맷된 다음 TCP/IP상의 HTTP를 이용하여 전송된다. 디바이스는 이 컨트롤 요청에 30초 이내에 응답해야 한다. 디바이스는 현재 액션이 보류중이거나 이벤트가 완료되었다는 등의 결과를 SOAP 메시지로 작성하여 Control Point에게 응답한다. 또한 컨트롤 애플리케이션은 특정한 서비스 변수의 상태를 쿼리할 수도 있고 쿼리를 하나씩 전송하여 하나의 상태변수만 받을 수도 있다.

5) 이벤트(Eventing)

서비스에 대한 UPnP 표현에는 그 서비스가 반응을 하는 액션들의 목록과 서비스의 상태를 나타내는 변수들의 목록이 포함된다. 서비스는 이 변수들이 변화하면 업데이트를 통보하며 Control Point는 이를 구독하여 이 정보를 수신할 수 있다.

서비스는 이벤트 메시지를 전송함으로써 업데이트를 통보한다. 이벤트 메시지에는 변수들의 이름과 이 변수들의 현재 값이 들어 있다. 이 메시지들 역시 XML로 표현되며 GENA를 이용하여 포맷된다. Control Point가 처음 구독을 하면 특별한 초기 이벤트 메시지가 전송되는데, 이 이벤트 메시지에는 이벤트된 변수들 모두에 대한 이름과 값이 들어 있고 구독자가 서비스 상태의 모델을 초기화시킬 수 있게 해준다. 여러 개의 Control Point를 구독하기 위해 모든 구독자들은 모든 이벤트 메시지를 전송하며 구독자는 모든 이벤트된 변수들에 대한 이벤트 메시지를 수신한다. 그리고, 이벤트 메시지는 상태 변수가 왜 바뀌었는지에 상관없이 액션 요청이나 상태 변화에 반응하여 전송된다. 우선 구독자(Control Point)가 발행자(디바이스)에게 구독을 요청하고 발행자가 이를 승인하면 구독(Subscription)이 이루어진다. 구독자는 구독예약 기한이 지나기 전에 갱신(Renew)해야 하고 만약 구독을 취소하려면 취소 메시지를 보내야 한다.

구독자가 발행자에게 구독 요청(Subscription Request)메시지를 보낸다. 구독 요청이 수락되면 발행자(Publisher)는 30초안에 응답메시지를 보내야한다. 이 응답 메시지에는 이벤트 구독에 대한 고유한 UUID인 SID가 포함된다. 만약 발행자가 갱신(renewal)을 수행할 수 없으면 30초 이내에 에러 메시지를 보낸다. 구독(Subscription)이 정상적으로 취소(Cancel)되면 발행자(Publisher)는 30초안에 응답메시지를 보내야한다.

디바이스의 상태변수 값에 변화가 있을 경우, 디바이스는 Control Point에게 이벤트 메시지(Event Message)를 구독자(Subscriber)에게 보낸다. 메시지의 헤더에 구독 시 발행된 SID와 몇 번째로 발생된 이벤트인지를 나타내는 SEQ를 포함하고, Body 부분에 변경된 상태변수의 리스트를 첨부한다.

Publisher로부터 Event 메시지를 받으면 Subscriber는 30초 이내에 응답하여야 한다. 만약 구독자로부터 응답이 오지 않으면 발행자는 구독 기한(Subscription Expiration)동안 계속

이벤트 메시지를 보내야 한다. 구독자가 이벤트 메시지를 놓쳤다든지, 구독이 수리되어야할 경우가 생기면 구독자는 구독을 취소하고 다시 구독해야 한다.

6) 제시(Presentation)

Control Point가 UPnP 네트워크에 접속한 후 디바이스를 찾고 디바이스의 표현(Description)을 다운로드 하여 분석한 후 제시(Presentation)를 실행한다. 만약 디바이스가 제시를 위한 URL을 갖고 있다면 Control Point는 이 URL 페이지를 검색하여 그것을 브라우저로 로드할 수 있으며, 그 페이지의 기능에 따라서는 사용자가 디바이스를 제어하거나 디바이스 상태를 볼 수 있게 해준다. 이런 것들의 수행 정도는 표현 페이지 및 디바이스의 특정 기능에 따라 달라진다. 컨트롤 포인트의 웹 브라우저를 통해 디바이스에 HTTP GET요청을 보내고, 제시 페이지의 HTML 내용을 받아서 컨트롤하고 상태를 검색한다.

Ⅲ. 임베디드 리눅스 시스템

컴퓨터 기술과 인터넷을 포함한 정보 통신의 눈부신 발전은 PC 보급 확산을 재촉했다. PC는 사무 자동화, 교육 및 훈련, 정보 사냥 등 다양한 서비스를 제공해 주고 있으며 고속 성장을 거듭하고 있다. 또한 PC 대중화는 노트북 PC 시장 동반 성장에 한 걸음 더 나아가 핸드 헬드 PC(HPC), 개인 휴대 단말기(PDA) 등의 발전을 낳고 있다. PDA는 개인 데이터베이스 관리, MP3 플레이어, 음성 및 영상 통신, 정보통신, 인터네트워킹, 지리 정보 시스템(GPS), 교육 훈련 등의 서비스를 편하고 손쉽게 제공해 줄 것이다. 또한 컴퓨터 기술은 디지털 카메라, 의료 및 산업 원격 조종기 등에 널리 응용되고 있는데, PDA를 포함한 이런 시스템들이 바로 임베디드 시스템(Embedded System)에 속한다.

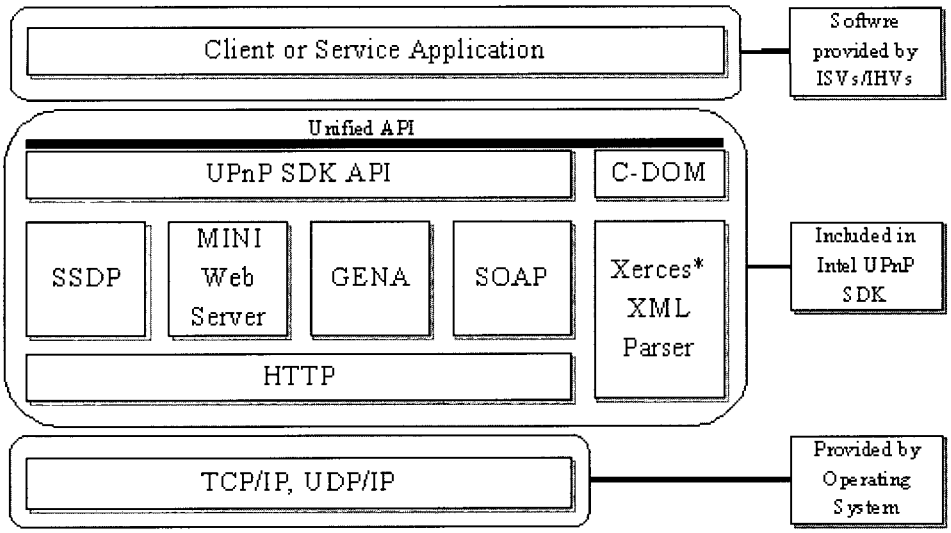
자유정신의 공개 운영체제인 리눅스는 인터넷의 성장과 더불어 비약적인 발전을 기록하고 있다. 리눅스는 소스가 공개되어 있고, 안정적으로 다양한 서비스를 제공함은 물론 확장 가능하다. 또한 무료 배포로 가격 경쟁력을 지니며, 안정적이고 다양한 서비스를 제공해 품질 경쟁력 면에서도 매력적이다. 소스가 공개되어 있다는 사실은 개발자들의 새 소프트웨어에 대한 연구 및 개발을 촉진시키며 운영체제가 확장 가능해 임베디드 시스템에도 적용 가능하다. 따라서 최근에 많은 임베디드 시스템에 리눅스가 채택되고 있다.

1. 개발환경

개발자가 임베디드 시스템에 적용될 디바이스 드라이버나 어플리케이션을 개발하는 작업은 타겟보드가 아닌 호스트 컴퓨터에서 이루어지게 된다. 타겟보드의 구성은 호스트 컴퓨터와 환경이 다르기 때문에 임베디드 시스템을 개발하기 위해서는 크로스 개발환경을 구성할 필요가 있다.

호스트컴퓨터와 타겟보드는 Ethernet, Serial Cable, JTAG 등으로 연결되어 있다. 타겟보드는 이더넷을 통하여 FTP로 제작된 어플리케이션을 다운로드할 수 있다. 호스트컴퓨터는 시리얼 케이블을 통해 타겟 보드의 상태를 모니터링하고 콘솔입력을 조정할 수 있다. JTAG 인터페이스로는 타겟 보드의 메모리를 직접 Read, Write할 수 있다.

타겟 보드에는 Ethernet, Serial 등의 인터페이스가 필요하고, 부트 로더와 리눅스 커널이 포팅되어 있어야 한다. 호스트 PC는 리눅스로 부팅



〈그림 2〉 UPnP SDK v1.0 소프트웨어 구조

되고, Minicom과 FTP 서버 데몬이 설치되어 있어야 하며, ARM용 Cross Compilier가 설치되어야 한다[11].

타겟보드에 OS가 포팅되고 기본적인 디바이스 드라이버가 작성이 되면 해당 임베디드 시스템에 요구되는 어플리케이션 개발 작업이 가능하다. 어플리케이션을 작성할 때는 타겟 보드에 포팅하기 전에 PC상에서 충분히 테스트한 뒤에 다시 컴파일 환경을 바꾸는 것이 좋다. 타겟 보드에서의 디버깅 작업이 어렵기 때문이다.

호스트 PC에서 완성된 어플리케이션을 임베디드용으로 포팅하는 방법은 Configuration 설정을 바꿔주고, Makefile을 수정하는 것이다. System과 관련된 설정 옵션을 조정하여 Makefile을 생성한 다음, Makefile에 있는 컴파일러, 링커 등을 ARM용으로 바꿔준다.

완성된 임베디드용 어플리케이션을 타겟 보드에 올리기 위한 방법은 두 가지가 있다. 하나는 타겟 보드의 루트 파일시스템에 직접 포함시키는 방법이고 두 번째는 타겟 보드를 부팅 시킨 뒤, FTP를 통해 다운로드하는 방법이다. 루트 파일 시스템에 포함시키는 방법은 타겟 보드의 플래시 메모리에 write 되므로 시스템이 재부팅 된 후에도 파일이 남아있게 되는 반면, FTP를 통해 다운로드 하는 방법은 부팅되어 있는 상태에서만 일시적으로 사용할 수 있다. 어플리케이션을 작성한 뒤 보드에서 테스트하고 디버깅할 때에는 FTP를 통해 다운로드하는 방법을 사용하는 것이 좋고, 디버깅이 끝난 후 지속적으로 사용하기 위해서는 루트 파일 시스템에 포함시키는 방법을 택하는 것이 좋을 것이다.

2. UPnP SDK v1.0 for Linux

Universal Plug and Play (UPnP)는 네트워크 안에 있는 디바이스들이 사용자의 간섭 없이 자동으로 다른 디바이스를 발견하고 서비스를 컨트롤할 수 있게 해준다. 디바이스는 클라이언트에게 자신의 서비스를 게시할 수 있는 서버로 동작한다. Control Point로 알려진 클라이언트 시

스템은 네트웍을 통해 정해진 서비스를 찾을 수 있다. 디바이스로부터 관심있는 서비스를 찾았을 때, Control Point는 디바이스와 서비스에 대한 좀 더 자세한 Description을 검색하고 서로 상호 작용 한다.

UPnP SDK API는 서로 떨어져 있는 Control Point와 Service Application이 통일된 인터페이스를 통해 접근하기 위한 UPnP Protocol의 핵심 기능들로 구성되어 있다. UPnP SDK(Universal Plug and Play Software Development Kit Version 1.0 for Linux)는 인텔社에서 만든 Tool 이며, http:// upnp.sourceforge.net/에서 소스와 PDF문서를 다운 받을 수 있다.

그림 2에 UPnP SDK v1.0의 프로토콜 구조를 나타내었다. UPnP는 IP 기반에서 동작하며 TCP를 기반으로 하는 HTTP 유니캐스트와 UDP를 기반으로 하는 HTTP 멀티캐스트를 통해 메시지를 전달한다. UPnP SDK는 HTTP 메시지를 처리하기 위한 Mini Web Server를 내장하고 있으며, UPnP의 핵심 프로토콜인 SSDP, SOAP, GENA 프로토콜을 구현한 API를 포함한다. Xerces XML Parser와 C-DOM을 사용하여, Control Point와 디바이스가 주고받는 XML 메시지를 파싱한다.

SSDP는 UPnP의 Discovery 단계를 처리한다. SSDP는 멀티캐스트로 동작하며 Control Point가 네트워크 내에 있는 디바이스들을 찾아내기 위한 프로토콜이다.

UPnP 디바이스는 UPnP의 Multicast 채널인 239.255.255.250:1900 포트를 Listen하며 메시지가 도착할 때까지 대기한다. Control Point로부터 Discovery 메시지가 도착하면 XML로 작성되어있는 자신의 간단한 디바이스 정보를 반환한다. Control Point는 반환된 XML 메시지를 파싱하여 디바이스의 타입을 알아내고, 발견된 디바이스를 목록에 추가시킨다.

UPnP SDK는 Device/Service Description을 전달하기 위한 Mini Web Server를 구현하고 있다. Mini Web Server는 UPnP Device에 내장되어, Control Point로부터 요청 받은 모든 HTTP 메시지를 처리한다. Control Point는 SSDP를 통해 발견된 디바이스 중에서 자신이 필요로 하는 디바이스가 판별되면, 해당 디바이스로부터 반환된 SSDP 응답 메시지의 Service Description URL로 디스크립션 페이지를 요청한다.

Control Point가 HTTP-GET 메시지로 디스크립션 페이지를 요청하면, UPnP Device의 Mini Web Server는 자신의 웹 디렉토리에서 해당 파일을 찾아 그 문자열을 HTTP로 응답한다. 또한, Mini Web Server는 SOAP과 GENA 메시지를 처리하기 위한 콜백함수를 포함한다. Control Point로부터 SOAP이나 GENA 메시지가 도착하면 해당 콜백함수를 호출하여 메시지를 처리한다.

GENA는 UPnP의 Eventing 단계를 처리한다. Control Point는 디바이스의 상태를 모니터링 하기 위해 필요한 디바이스에게 Eventing 구독을 요청한다. 디바이스는 구독 요청된 서비스에 대해 ID를 부여하여 요청이 받아들여졌음을 Control Point에게 알리고, 해당 서비스와 관련된 상태변수에 변화가 발생하면 Control Point에게 변경된 내용을 XML로 기술하여 알려준다.

SOAP은 UPnP의 Control 단계에 해당하는 프로토콜이다. Control Point는 서비스 디스크립션을 파싱하여 디바이스가 제공하는 기능을 찾아낸다. 서비스 디스크립션의 내용에는 디바이스를 Control 하기 위한 Action 정보가 포함되어있다. Control Point가 디바이스를 제어하고자 할 때, 해당 Action을 SOAP 메시지를 통해 디바이스에게 요청한다. 요청을 받은 디바이

스는 XML String으로 표현되어있는 SOAP 메시지를 파싱하여 해당 서비스를 수행한 후, 그 결과를 반환한다.

UPnP SDK는 그림 2와 같이 TCP/IP 위에서 동작하며, HTTP 프로토콜과 SSDP, GENA, SOAP 프로토콜, 그리고 XML DOM Parser를 포함하며, 이 프로토콜들을 사용하기 위한 API로 구성되어있다. UPnP Control Point와 디바이스 어플리케이션은 UPnP SDK의 API를 호출하여 UPnP의 모든 기능들을 구현할 수 있다.

UPnP 어플리케이션 제작을 위하여 UPnP SDK를 설치하려면 pthread 라이브러리와 uuid 라이브러리가 필요하다. pthread 라이브러리는 glibc 패키지에 포함되어 있으며 본 실험에서 사용할 타겟보드에도 포팅 되어있다. uuid 라이브러리는 e2fsprogs 패키지 안에 포함되어 있으며 타겟보드에는 포팅 되어있지 않기 때문에 UPnP SDK를 설치하기 전에 미리 설치해주어야 한다.

UPnP SDK를 컴파일 하면 최종적으로 공유 라이브러리인 libupnp.so 파일을 생성한다. 본 실험을 위해 UPnP SDK를 ARM용으로 컴파일하고 예제 어플리케이션과 함께 타겟보드인 아사벳 보드에 다운로드 하여 실행시킨 결과 보드와 크로스컴파일러의 링커 타입이 일치하지 않아 에러를 발생 시켰다. 실험의 결과를 얻기 위해 UPnP SDK 라이브러리를 정적 라이브러리로 변환하여 타겟보드에서도 UPnP 어플리케이션이 동작하도록 하였다.

4. 실험 및 결과 고찰

본 논문에서는 인텔의 SA-1110 Evaluation 보드인 Assabet 보드에 디지털 TV 기능을 가진 UPnP 에뮬레이터를 올리기 위하여 한 대의 PC에 리눅스를 셋업하고 ARM용 크로스 개발 툴 킷을 설치하였다. 설치한 크로스 컴파일러의 버전은 다음과 같다.

- arm-linux-binutils-2.10
- arm-linux-gcc-2.95.2
- arm-linux-glibc-2.13.2

타겟보드와 호스트 컴퓨터는 RS232와 이더넷으로 연결되어 있다. Minicom을 실행하면 RS232를 통해 호스트 컴퓨터의 모니터로 타겟보드의 콘솔 입출력을 관찰할 수 있다. 호스트 컴퓨터에서 ARM용으로 크로스 컴파일된 어플리케이션은 이더넷을 통해 타겟보드로 다운로드 될 수 있다.

본 실험에서는 디지털 TV의 기능과 서비스를 분석하여 UPnP Description을 XML 문서로 작성하고 Intel UPnP SDK에서 제공하는 API를 이용하여 시스템이 UPnP 디바이스로 동작하도록 어플리케이션을 작성하였다. 호스트 컴퓨터에서 UPnP Control Point를 실행시키고, 타겟보드에서 UPnP TV 디바이스를 실행시켜 네트워크로 연결된 두 시스템이 UPnP 프로토콜을 통해 동작하는 과정을 확인하였다.

1. UPnP DTV 디바이스의 서비스 분석

DTV(Digital TV)는 DSTB(Digital Set-Top Box)로부터 수신된 디지털 방송을 출력하기 위한 장치이다. 본 논문에서는 DTV의 기능을 UPnP로 구현하여 리눅스 시스템에서 동작하도록 어플리케이션을 구성하였다.

UPnP 디바이스를 프로그래밍하기 위해서는 먼저 디바이스가 가진 기능을 XML 디스크립션

으로 작성해야 한다. 본 논문에서는 가장 단순한 기능을 가진 DTV 설계를 목적으로 하였으며, DTV 디바이스의 서비스를 두 가지로 정의하였다. 두 가지 서비스 중 하나는 채널, 볼륨 등을 조절할 수 있는 Control 서비스이고, 다른 하나는 컬러, 밝기 등을 조절할 수 있는 Picture 서비스이다.

본 논문에서 설계한 DTV Device의 디스크립션은 XML v1.0으로 작성되었다. root 태그의 xmlns(XML Name Space)는 이 문서가 UPnP Device v1.0에 적합한 문서임을 나타낸다. specVersion은 UPnP Device Architecture version 1.0 임을 나타낸다. URLBase는 모든 디스크립션 문서가 http://192.168.1.109:2000 주소 아래에 위치함을 나타낸다.

Device 정보는 DeviceType, 사용자가 쉽게 구분할 수 있게 하기 위한 FriendlyName, 제품정보, 모델정보, 디바이스 uuid와 씨리얼 넘버로 구성된 Unique Device Name, 패키지에 대한 고유번호인 Universal Product Code와 디바이스가 제공하는 서비스의 리스트, 그리고 프리젠테이션 페이지의 주소를 포함한다.

Service List에는 UPnP TV Device의 두 가지 서비스인 tvcontrol-1과 tvpicture-1 서비스에 대한 정보가 있다. 각각의 서비스는 Service ID, Control URL, Event URL과 Service Description 주소인 SCPD(Service Control Protocol Definition) URL을 가지고 있다. 각각의 주소는 URLBase를 제외한 하위 주소만으로 표현되어있다.

UPnP TV Device의 Control Service는 Power, Channel, Volume의 상태를 모니터하고 컨트롤 하기 위한 서비스이다. 표 1과 같이 UPnP TV Control Service에서는 PowerOn, PowerOff, SetChannel, IncreaseChannel, DecreaseChannel, SetVolume, IncreaseVolume, DecreaseVolume 등 8개의 Action이 제공됨을 알 수 있다. Control Service에는 3개의 상태변수가 존재한다. Power 변수는 0 또는 1값을 가질 수 있고, Power의 On, Off 상태를 나타낸다.

Channel 변수는 채널번호를 나타내며, 1에서 100까지의 정수 값을 가질 수 있다. 마지막으로 Volume 변수는 볼륨의 크기를 나타내며, 1에서 10까지의 정수 값을 갖는다.

UPnP TV Device의 Picture Service는 Color(색), Tint(색채), Contrast(명암), Brightness(밝기)를 조절하기 위한 서비스를 제공한다. Action List에는 SetColor, IncreaseColor, DecreaseColor, SetTint, IncreaseTint, DecreaseTint, SetContrast,

〈표 1〉 tvcontrol 서비스의 액션 목록

Action Name	Argument
PowerOn	-
PowerOff	-
SetVolume	Volume
IncreaseVolume	-
DecreaseVolume	-
SetChannel	Channel
IncreaseChannel	-
DecreaseChannel	-

〈표 2〉 tvcontrol 서비스의 상태변수들

Argument Name	Allowed Value	Default Value
Power	0 또는 1	0
Channel	1에서 100까지의 정수	1
Volume	1에서 10까지의 정수	5

IncreaseContrast, DecreaseContrast, SetBrightness, IncreaseBrightness, DecreaseBrightness 등 12개의 Action이 기술되어 있다. Picture Service의 상태변수는 Color, Tint, Contrast, Brightness 네 가지가 제공되며, 각각의 상태변수는 1에서 10까지의 정수값으로 표현된다.

Control Point가 DTV 디바이스를 발견하고 디바이스에게 디스크립션을 요청하면 디바이스에 내장된 웹서버는 디바이스 디스크립션을 HTTP 프로토콜을 통해 전송한다. Control Point는 디바이스 디스크립션을 받아서 파싱한 뒤, SCPDURL을 찾아 다시 서비스 디스크립션을 요청한다. Control Point는 서비스 디스크립션의 action name과 argument name으로 디바이스를 컨트롤 할 수 있고, stateVariable의 값이 변경되었을 때 디바이스는 Control Point에게 이벤트가 발생했음을 알려준다.

〈표 3〉 tvpicture 서비스의 액션 목록

Action Name	Argument
SetColor	Color
IncreaseColor	-
DecreaseColor	-
SetTint	Tint
IncreaseTint	-
DecreaseTint	-
SetContrast	Contrast
IncreaseContrast	-
DecreaseContrast	-
SetBrightness	Brightness
IncreaseBrightness	-
DecreaseContrast	-

〈표 4〉 tvpicture 서비스의 상태변수들

Argument Name	Allowed Value	Default Value
Color	1에서 10까지의 정수	5
Tint	1에서 10까지의 정수	5
Contrast	1에서 10까지의 정수	5
Brightness	1에서 10까지의 정수	5

2. UPnP 디바이스와 Control Point 프로그램

본 논문의 실험을 위해 임베디드 리눅스 시스템에서 동작하는 UPnP 디바이스를 C 프로그램으로 작성하였다. 다음은 UPnP TV 디바이스 프로그램의 실제 소스 중 main 함수를 발췌한 것이다.

```
int main(int argc, char** argv)
{
  int ret=1;
  int port;
  char *ip_address=NULL,
       *desc_doc_name=NULL,
       *web_dir_path=NULL;
  char desc_doc_url[200];
  pthread_t cmdloop_thread;
  int code;
  int sig;
  sigset_t sigs_to_catch;
  ip_address=argv[1];
  sscanf(argv[2],"%d",&port);
  desc_doc_name=argv[3];
  web_dir_path=argv[4];
  sprintf(desc_doc_url,
  "http://%s:%d/%s", ip_address, port,
  desc_doc_name);
```

```
printf("Intializing UPnP \n\t with
desc_doc_url=%s\n\t", desc_doc_url);
printf("\t ipaddress=%s port=%d\n",
ip_address, port);
printf("\t web_dir_path=%s\n",
web_dir_path);
if ((ret = UpnpInit(ip_address, port)) !=
UPNP_E_SUCCESS) {
    printf("Error with UpnpInit —
    %d\n", ret);
    UpnpFinish();
    exit(1);
}
printf("UPnP Initialized\n");
printf("Specifying the webserver root
directory — %s\n", web_dir_path);
if ((ret = UpnpSetWebServerRootDir
(web_dir_path)) !=UPNP_E_SUCCESS) {
    printf("Error specifying webserver
    root    directory — %s: %d\n",
    web_dir_path, ret);
    UpnpFinish();
    exit(1);
}
printf("Registering the RootDevice\n");
if ((ret = UpnpRegisterRootDevice
(desc_doc_url, TvDeviceCallbackEventHandler,
&device_handle, &device_handle)) !=
UPNP_E_SUCCESS) {
    printf("Error registering the
    rootdevice : %d\n", ret);
    UpnpFinish();
    exit(1);
} else {
    printf("RootDevice Registered\n");
    printf("Initializing State Table\n");
    TvDeviceStateTableInit(desc_doc_url);
    printf("State Table Initialized\n");
    if ((ret = UpnpSendAdvertisement
      (device_handle, default_advr_expire))
      != UPNP_E_SUCCESS) {
    printf("Error sending advertisements
    : %d\n", ret);
    UpnpFinish();
    exit(1);
    }
    printf("Advertisements Sent\n");
}
/* start a command loop thread */
code =
pthread_create(&cmdloop_thread,
NULL, TvDeviceCommandLoop,
NULL);
/* Catch Ctrl-C and properly shutdown */
sigemptyset(&sigs_to_catch);
sigaddset(&sigs_to_catch, SIGINT);
sigwait(&sigs_to_catch, &sig);
printf("Shutting down on signal
%d...\n", sig);
UpnpUnRegisterRootDevice(device_handle);
UpnpFinish();
exit(0);
}
```

어플리케이션이 실행되면 UPnP의 각 구성 요소들을 초기화하고 시스템을 UPnP 디바이스로 등록한다. UpnpInit() 함수는 디바이스를 초기화하는 함수로서, IP를 설정하고, 인터럽트를 초기화하며, MiniServer에 SOAP과 GENA Callback 함수를 등록한 후, SSDP를 위하여 멀티캐스트 채널을 설정하고, WebServer를 동작시

킨다. UpnpSetWebServerRootDir() 함수는 웹 서비스를 위한 기본 디렉토리를 WebServer에 저장하고, HTTP-GET 메시지를 처리하기 위한 Callback 함수를 등록한다.

UpnpRegisterRootDevice() 함수는 프로그램을 UPnP Root Device로 등록하는 함수로, 이벤트 핸들러를 설정하고, 자기자신의 디스크립션을 다운로드 하여 웹서버의 동작상태를 확인하고, 디스크립션을 파싱하여 서비스 리스트 테이블을 초기화한다.

초기화가 끝나고 나면 UPnP Multicast Channel을 통해 Advertise Message를 전송함으로써 UPnP Control Point들에게 자신이 네트워크 내에 들어왔음을 알린다. 그 후에는 커맨드 라인으로부터 프로그램 종료 명령이 입력될 때까지 네트워크나 시스템에서 발생하는 이벤트 메세지를 처리한다. 네트웍을 통해 발생하는 이벤트는 MiniServer가 처리하며 SOAP 메시지는 ProcessSoapEventPacket() 함수로, GENA 메시지는 genaCallback() 함수로, SSDP 메시지는 SsdpCallback- EventHandler() 함수로 처리한다.

다음으로, UPnP 디바이스를 제어하기 위해 리눅스 호스트 시스템에서 동작하는 UPnP Control Point를 설계하였다. 다음은 UPnP Control Point의 프로그램 중 main 함수의 소스코드이다.

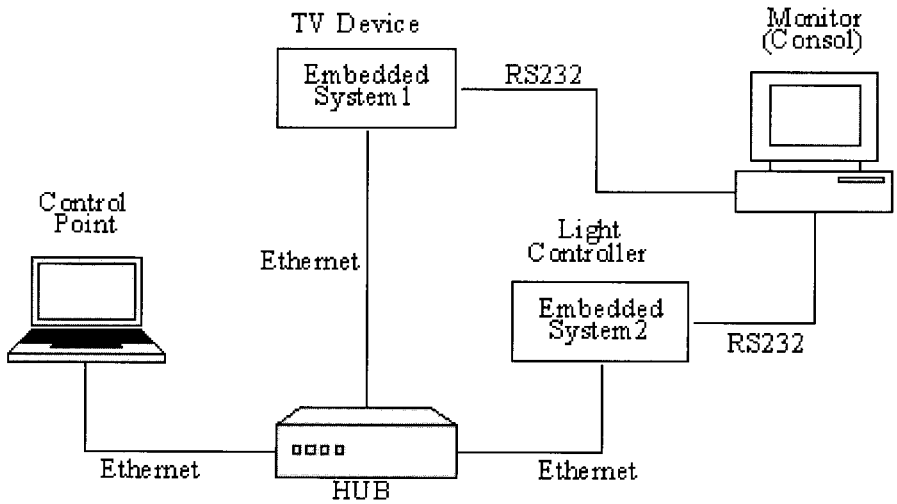
```
int main(int argc, char** argv)
{
    int ret=1;
    pthread_t              timer_thread,
    cmdloop_thread;
    int code;
    int port;
    char *ip_address;
    int sig;
    sigset_t sigs_to_catch;
    CMD_STATE = VIEW_DEVICE_LIST;
    DEVICE_STATE = 0;
    SERVICE_STATE = 0;
    QUERY_ACTION_STATE = 0;
    COMMAND_NUM = 0;
    sscanf(argv[2],"%d",&port);
    ip_address="192.168.1.109";
    port=2000;

    printf("Intializing UPnP \n\twith
    ipaddress=%s  port=%d  \n" ,
    ip_address, port);
    if ((ret = UpnpInit(ip_address, port)))
    {
        printf("Error with UpnpInit --
        %d\n", ret);
        UpnpFinish();
        exit(1);
    }
    printf("UPnP Initialized\n");
    printf(" Registering the Control
    Point\n");
    if ((ret = UpnpRegisterClient(UCP_
    CallbackEventHandler, &ctrlpt_
    handle, &ctrlpt_handle)))
    {
    printf("Error registering control point :
    %d\n", ret);
        UpnpFinish();
        exit(1);
    }
    printf("Control Point Registered\n");
```

```
    UCP_Refresh();
    // start a command loop thread
    code = pthread_create( &cmdloop_
    thread, NULL, UCP_CommandLoop,
    NULL );
    // start a timer thread
    code = pthread_create( &timer_
    thread, NULL, UCP_TimerLoop, NULL
    );
/* Catch Ctrl-C and properly shutdown */
    sigemptyset(&sigs_to_catch);
    sigaddset(&sigs_to_catch, SIGINT);
    sigwait(&sigs_to_catch, &sig);
    printf("Shutting down on signal
    %d...\n", sig);
    UpnpUnRegisterClient(ctrlpt_handle);
    UpnpFinish();
    exit(0);
}
```

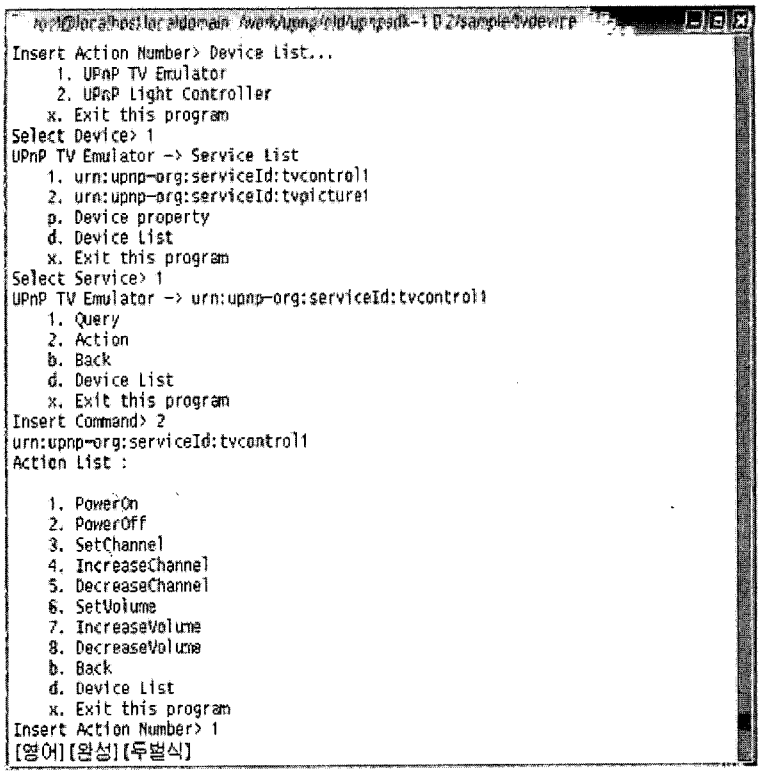
위의 프로그램에서 컨트롤 포인트 프로그램이 실행되면 UpnpInit() 함수로 UPnP를 초기화하고 UpnpRegisterClient() 함수로 이벤트 핸들러를 설정한다. ControlPointRefresh() 함수는 디바이스 리스트를 초기화하고, Device Search 메시지를 전송한다.

SSDP Search 프로토콜에 의해 발견된 디바이스들의 정보를 디바이스 리스트 구조체에 저장하고 커맨드 루프와 타이머 쓰레드를 생성한다. 타이머 쓰레드는 30초마다 Advertisement Timeout을 체크하여 디바이스 타임아웃 정보를 수정하고 제한 시간이 60초 남았을 경우 해당 노드에 Search 메시지를 전송하여 응답이 있을 경우 타임 아웃 정보를 수정한다.

Renewal 메시지 응답이 없고, 제한시간이 0보다 작으면 해당 디바이스가 네트워크 내에서 빠져나간 것으로 보고 디바이스 리스트에서 해당노드를 삭제한다.



〈그림 3〉 UPnP 네트워크 구성도



〈그림 4〉 TV 디바이스의 PowerOn 액션을 실행

3. 어플리케이션의 실행

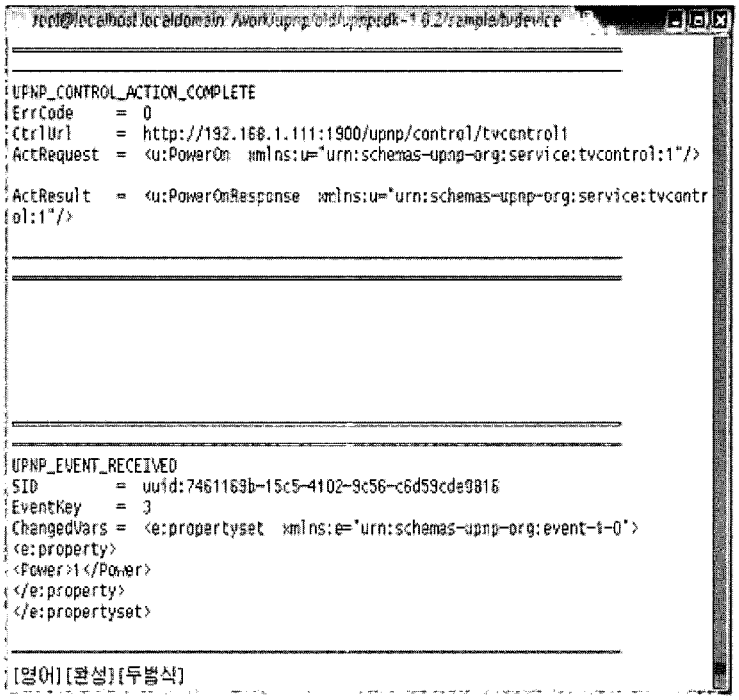
본 논문의 실험을 위하여 그림 3과 같이 디바이스들의 네트워을 구성하였다. 먼저 한 대의 노트북에 리눅스를 셋업하고 이더넷을 연결하였다. 두 대의 임베디드 리눅스 개발보드를 UPnP TV 디바이스와 UPnP Light Controller 디바이스로 정하고 마찬가지로 이더넷을 허브에 연결한다.

UPnP 멀티캐스트 채널 사용을 위해 노트북과 타겟 보드에

```
route add -net 239.0.0.0 netmask
255.0.0.0 eth0
```

명령을 실행하면 라우팅 정보를 추가하거나, 시스템에서 지원하도록 미리 설정된다. 모니터링용으로 사용할 PC의 직렬포트에 타겟보드를 연결하고, Minicom을 실행시켜 타겟보드의 콘솔에 로그인한다. 부팅이 시작되면 노트북과 두 대의 보드는 허브에 연결된 공유기로부터 IP를 자동으로 할당받는다.

이제, 노트북에서 Control Point를 실행시키고 Minicom을 통해 타겟보드에서 TV 디바이스를 실행하면, 프로그램이 초기화되고, Advertisement 메시지를 전송하는 것을 볼 수 있다. 그림 4는 컨트롤포인트에서 DTV 디바이스의 PowerOn 명령을 실행시키는 모습이다. UPnP Protocol 상에서 네트워크로 연결된 원격지의 디바이스를 모니터링하고 컨트롤할 수 있다. Control Point 프로그램은 디바이스의 서비



〈그림 5〉 PowerOn 액션을 실행을 완료

스를 실행시키기 위하여 디바이스와 서비스를 선택하는 과정을 단계별로 진행할 수 있도록 설계되었다. Control Point는 네트워크에 존재하는 UPnP 디바이스를 찾아내어 목록을 콘솔에 출력한다. Device 목록은 Device Description의 FriendlyName으로 표현되고, Service 목록은 ServiceId로 표현된다.

화면에서 Control Point는 UPnP TV Emulator와 UPnP Light Controller 두 개의 디바이스를 찾아낸 것을 볼 수 있다. 첫 번째 단계로 디바이스 목록 중에서 '1'을 입력하여 UPnP TV Emulator를 선택하면, Control Point는 UPnP TV Emulator의 서비스 목록을 출력한다. tvcontrol-1과 tvpicture-1 두 개의 서비스와 디바이스 정보를 출력하는 메뉴, 디바이스 리스트를 출력하는 메뉴 그리고 프로그램을 종료하는 메뉴를 제공한다.

두 번째 단계로 디바이스의 서비스 중 tvcontrol-1 서비스를 선택한다. Control Point는 디바이스의 상태변수를 볼 수 있는 Query와 디바이스 기능을 실행시킬 수 있는 Action 메뉴를 제공한다. '2'를 입력하여 Action을 선택하면, Control Point는 tvcontrol-1 서비스의 8가지 액션을 출력한다. '1'을 입력하여 PowerOn 명령을 선택하면 Control Point는 PowerOn 명령에 대한 SOAP 메시지를 작성하여 디바이스에게 전달한다.

디바이스는 SOAP 메시지를 수신하여 요청된 명령을 실행한 후 실행된 결과를 Control Point에게 반환한다. Control Point는 반환된 메시지를 분석하여 그림 5와 같이 출력한다. UPNP_CONTROL_ACTION_COMPLETE 메

시지는 명령이 정상적으로 수행되었음을 나타낸다. CtrlUrl에 서비스가 실행된 위치를 알 수 있고, ActRequest에 tvcontrol-1의 PowerOn 명령이 실행되었음이 나타나 있다. ActResults는 PowerOn의 결과임을 표시한다.

디바이스가 외부로부터 명령을 받아 서비스가 실행되면 디바이스의 상태변수에 변화를 일으키게 된다. 그러면, 디바이스는 Control Point에게 변경된 상태변수 값을 이벤트로 알려야 한다. 그림 5의 아랫부분은 Control Point가 디바이스로부터 GENA 메시지를 통해 전달받은 이벤트를 분석하여 표시한 결과이다. 메시지의 SID는 이벤트 구독 시 발행된 고유번호이며, 어떤 서비스로부터 온 메시지인지를 구분한다. EventKey는 세 번째로 발생된 이벤트임을 나타내고, ChangedVars의 XML 문장을 통해 Power 변수가 1값으로 변경되었음을 알린다.

Ⅴ. 결론

본 논문에서는 UPnP Control Point와 디지털 TV, 그리고 전등 제어기 에뮬레이터를 리눅스 PC와 임베디드 리눅스 시스템에서 사용하기 위해, 인텔의 UPnP SDK v1.04를 임베디드 시스템으로 포팅하고, C언어로 어플리케이션을 프로그래밍 하였다.

본 논문에서 사용한 UPnP SDK v1.04는 인텔사에서 배포한 것으로, 리눅스 시스템 상에서 UPnP Control Point와 UPnP 디바이스를 개발하기 위한 API를 제공하는 패키지이다. 임베디드 리눅스에서 동작하는 UPnP 디바이스를 만들기 위해, 이 패키지를 StrongARM CPU에서 동작하도록 포팅 하였다.

UPnP를 지원하는 DTV 디바이스 에뮬레이터를 제작하기 위해 디지털 TV의 기능을 분석하여 각각의 서비스를 정의하고 이 서비스의 UPnP XML Description을 작성하였다. UPnP SDK가 제공하는 API를 사용하여 UPnP Control Point와 UPnP 디바이스를 작성하고, 작성된 UPnP 디바이스를 StrongARM 시스템으로 포팅 하였다.

한 대의 PC를 Control Point로 설정하고, 두 대의 임베디드 시스템을 디지털 TV와 전등 제어기로 동작하도록 설정한 후, 각각의 타겟보드와 PC를 네트워크로 연결하고 Control Point와 디바이스를 실행시켜 실험하였다.

실험 결과 Control Point와 디바이스 프로그램이 UPnP 프로토콜의 Discovery, Description, Control, Eventing 단계를 지원하는 것을 콘솔을 통해 확인하였다. UPnP 프로토콜 중 Presentation 단계는 본 논문의 실험에서는 고려되지 않았고, Addressing 단계는 리눅스 시스템에서 제공되는 DHCP 클라이언트를 통해 지원된다.

UPnP는 어떤 기기가 이 네트워크에 참여하면 스스로 이 기기를 찾아내고 기기의 속성을 파악해내며 파악한 속성을 이용하여 모든 기기를 제어할 수 있을 뿐만 아니라 각각의 기기에 대한 상태의 변화를 직접 알려 주고 이에 따라 제어할 수 있는 Device-to-Device Control 기능을 가지고 있다.

즉 UPnP의 가장 큰 철학은 Plug-and-Play 기능을 제공하자는 것이다. 그러나 이러한 기능이 그 네트워크의 보안 문제를 그대로 외부에 노출하는 결과를 초래하므로 Plug-and-Play 기능을 어느 수준으로 포기하며 데이터의 보안 문제를 유지할 수 있는 방안이 필요하다.

이를 위해 UPnP v2.0에서는 UPnP 네트워크의 보안 문제를 고려하고 있으며 이와 같은 두 개

의 서로 상충되는 문제를 해결하는 방안이 앞으로 연구해야 할 과제이다. 또한 UPnP 네트워크에 연결되어 있는 기기의 수가 많아지면 기기를 발견하는 시간이 길어지므로 이에 대한 속도 향상 방법을 찾는 것이 본 연구에서 앞으로 수행할 예정이다.

저자 소개

전 호 인

1981년 연세대 전자공학과 (학사)
1984년 연세대 전자공학과 (석사)
1986년 University of Southern California (석사)
1990년 University of Alabama (Ph.D.)

현재 Binary CDMA Forum 표준분과위원회 위원장
Home Station 표준화 Forum 전문위원회 위원장
초고속 무선랜 포럼 표준분과 위원회 위원장
ISO/IEC JTC1 SC25 전문위원회 위원장
ISO/IEC JTC1 SC6 전문위원회 위원
1394 Forum 의장
3D TV 추진협의회 의장
경원대 전자공학과 교수

■ 참고문헌

[1] 전호인, 차세대 IT주자 홈 네트워크, 프로그램세계, pp. 198-225, 2001년 8월
[2] 전호인, 신 용섭, "홈 네트워킹 기술 및 표준화 동향," 전자공학회지 제 29권 6호, pp. 18 - 39, 2002년 6월.
[3] Universal Plug and Play Device Architecture v1.0, Microsoft/UPnP Forum, 2000
[4] Universal Plug and Play White Paper, Microsoft, 2000
[5] "Hypertext Transfer Protocol — HTTP/1.1", R. Fielding, J. Gettys, J. Mogul, H. Nielsen, T. Berners-Lee, January 3, 1997. Available at http:// www.w3.org/TR/REC-CSS1-961217
[6] "Extensible Markup Language (XML)", Version 1.0, W3C Recommendation (http:// www.w3.org/TR/REC_xml)
[7] Simple Service Discovery Protocol/1.0 INTERNET DRAFT Document, Available at http://upnp.org/download/draft_cai_ssdp_ v1_03.txt
[8] Simple Object Access Protocol (SOAP) 1.1, W3C, Available at http://www.w3.org/ TR/SOAP/
[9] General Event Notification Architecture INTERNET DRAFT Document, Available at http://upnp.org/download/draft-cohen- gena-client-01.txt
[10] "HTML 4.0 Specification", D. Raggett, A. Le Hors, I. Jacobs, 24 April 1998. Available at http://www.w3.org/TR/REC- html40/
[11] Linux Networking - HOWTO, TCP/IP 네트워크 관리 2판, 한빛출판사, 1999