논문 2004-41CI-2-4

블루투스를 이용한 가전기기 원격제어 시스템

(A Remote Control System Using Bluetooth)

이 우 중*, 황 우 식*, 김 정 선**

(Woo-Joong Lee, Woo-sik Hwang, and Jungsun Kim)

요 약

가전기기를 무선으로 제어하기 위한 리모트 컨트롤러의 상당수는 적외선을 제어 신호의 매개체로 사용한다. 그러나 적외선은 사용자가 가전기기에 보내는 제어 신호의 증폭을 할 수 없으며 물리적인 장벽을 통과하지 못한다. 적외선 대신 무선 주파수를 이용하여 사용자의 제어 신호를 전송하면 이러한 단점을 해결할 수 있으나 기기 간 호환성과 확장성이 부족하다.

본 논문에서는 이러한 제약을 극복하기 위하여 블루투스를 이용한 가전기기 제어 시스템을 제안한다. 블루투스는 근거리 무선 통신 표준으로서 저가, 저전력 등의 특성으로 모바일 장비에 적합하다. 본 논문에서는 임베디드 환경에서 블루투스 피코넷을 통한 가전기기 제어를 구현하였다. 이러한 기초 제어는 PDA나 휴대폰 등에 내장 될 경우 다양한 형태로 활용 가능할 것이다.

Abstract

Controlling household devices using an infra-ray (IR) has a limitation since the IR signal cannot only be extended, but also cannot be transmitted over the physical barrier. Using a radio frequency (RF) may overcome these shortcomings. Bluetooth is one of the emerging RF_based standard protocols for short-range wireless communications among various control devices. In this paper, we describe a control system for household appliances using Bluetooth under embedded Linux environment. The control system consists of a centralized main controller and geographically scattered wallplates. Some of the challenges and lessons learned will be outlined in the presentation. As a result of this research, we found that the Bluetooth is a promising technology for realizing wireless control systems for household appliances.

Keywords: Bluetooth/Home network/Piconet/Embedded Linux

I. 서 론

1. 가전 기기의 무선제어

1901년 마르코니가 모르스 부호를 무선으로 송수신하는 방법을 고안한 이후로 무선 통신 기법은 유선 통신에 비하여 사용자의 이동성과 편의성을 크게 증대시켰다. 이를 시작으로 많은 분야에서 무선 통신이 보편화되었으며 제어 장치에도 많이 사용된다. 현재 원거리의 음성 및 데이터 송수신에는 라디오 주파수를 주로 이용하고 있으나 유독 가전기기 등의 제어 컨트롤러에는 라디오 주파수가 아닌 적외선을 이용하여 사용자의 제어 신호를 전송한다.

이같이 적외선을 이용하여 제어 신호를 전송하면 적외선이 가지는 고유 특성인 직진성과 확산성으로 인하여 벽이나 문과 같은 물리적인 장애물을 지나서 제어 신호를 전송할 수 없으며, 신호를 중간에서 증폭시키는 것이 불가능하다.

* 학생회원, 한양대학교 컴퓨터공학과
(Dept. Computer Eng., Hanyang University)

** 정회원, 한양대학교 컴퓨터공학과
(Dept. Computer Eng., Hanyang University)

접수일자: 2003년10월22일, 수정완료일: 2004년2월27일

이에 반하여 주파수(Radio Frequency)를 이용하여 제어 신호를 전송할 경우 물리적인 장애물을 통과 할 수 있고, 신호의 증폭을 통해 비교적 먼 거리로도 사용자의 신호를 전송할 수 있다.

이런 점에도 불구하고 가전기기 무선 제어에 라디오 주파수를 쓰지 않은 이유는 무선 모듈의 가격이 적외선에 비하여 고가이며, 아파트와 같은 조밀지역에서는 타인의 기기까지 신호가 전송되어 동작에 간섭을 일으킬 수 있다.

가전기기의 무선 제어에도 라디오 주파수를 이용하려면 이러한 점의 해결이 선행되어야 하며, 이를 위해서 저가의 모듈을 사용하며 원하는 기기에만 선택적으로 제어 신호를 송신할 수 있는 무선 제어 프로토콜을 이용하여야 한다.

2. 가전기기 제어를 위한 홈 네트워킹 기술에 대한 무선 통신 기술의 비교

라디오 주파수를 이용하는 주요 무선 통신용 프로토콜은 무선랜(IEEE 802.11b), HomeRF 그리고 블루투스(Bluetooth)[1][2][3]가 있다.

이들은 모두 2.4 GHz의 ISM(Industrial Scientific Medical) 주파수 대역을 사용하며, 전송 거리도 10m 이상으로 가택내의 가전기기 제어 수준에 만족한다.

무선랜은 PC의 네트워크 선의 대체를 위해 개발된 기술로 빠른 전송속도를 자랑하며, 간섭 방지를 위하여 주파수 호핑(Frequency Hopping) 기술을 사용한다. 그러나 엑세스 포인트(Access Point) 및 랜 모듈이 고가이고 전력의 소비도 높다.

HomeRF는 용어에서 의미하는 바와 같이 가택 내에서의 무선 주파수를 이용한 통신망을 구축하는 기술로 연구되기 시작했다. 그러나 기술성에 있어서 보다 표준화 경쟁에서 열세하고 개발 지원 업체가 미비하다는 단점이 있어 가전기기 제어를 위한 범용 무선 기술로 채택하기에 어려움이 있다.

이에 비하여 블루투스는 현존하는 기기들의 유선 케이블의 대체가 목적이므로 PC 뿐만 아니라 가전기기와 컨트롤러에 부착하기가 쉽고, 단일 칩(One Chip)이나 모듈로 제공되므로 제작 단가도 저렴하다. 또한 최대 30mA 이하의 저전력을 소모하므로 휴대폰, PDA 등의 휴대용 기기에 가전제어 기능을 추가할 수 있게 한다.

또한 블루투스는 여러 가전기기 제어를 위한 통합 네트워크 구성을 가능하게 한다. Service Discovery Protocol을 통하여 블루투스 표준을 준수하는 다양한 기기에 대하여 통합적인 제어가 가능하다.

본 논문에서는 임베디드 환경(SA1110 아키텍처, Linux)에서의 블루투스 기기 간 통신과 블루투스의 피코넷(Piconet)구성을 다룬다. 통신모듈의 구현을 위한 블루투스 프로토콜 스택의 임베디드 시스템으로의 이식과 임베디드 시스템 개발 키트와 블루투스 개발 키트 간 통신, SA1110의 GPIO(General Purpose Input Output)를 이용한 가전기기 제어에 대한 구현을 소개한다.

Ⅱ. 블루투스 피코넷과 스캐터넷

블루투스 네트워크는 기본적으로 피코넷[1][2][3]을 구성하여 통신한다. 아래의 그림 1은 이러한 피코넷의 구성을 보여준다.

Master는 접속을 시도한 블루투스 모듈이며 Slave는 접속을 당한 블루투스 모듈이다. 전체적인 피코넷의 제어는 Master가 관리하며 각각의 Slave는 접속 초기화 시에 Master의 시스템 클록(Clock)과 동기화 된다. 블루투스 기기는 주파수 호핑(Frequency Hopping)을 통하여 간섭을 피하는데 이때 Hopping Sequence는 모듈의 시스템 클록을 통하여 구하게 된다. 블루투스 피코넷에서는 하나의 Master 당 최대 7개의 Slave가 접속 가능하다. 블루투스 피코넷은 Master기기가 접속 가능한 블루투스 장비를 조회(Inquiry)를 통하여 찾아내고 접속하여 네트워크를 구성한다.

블루투스 피코넷은 단지 7개의 Slave를 가질 수 있으므로 확장을 위하여 스캐터넷이 제안 되었다. 스캐터넷은 피코넷과 피코넷이 결합된 형태로서 본 논문에서는 단일 피코넷의 구성을 통한 가전기기 제어로 한정한다.

Ⅲ. 임베디드 시스템을 위한 리눅스 커널과 블루투스 스택의 이식

1. 개발 환경

본 논문에서 사용한 개발 환경은 임베디드 시스템 개발 키트(이후 Target board 지칭)로 Intrinsyc사의 Cerf Board를 사용하였다. 이는 StrongARM 1110 마이크로프로세서를 장착하고 있으며 세부적인 개발 키트의 특징은 표 1과 같다.

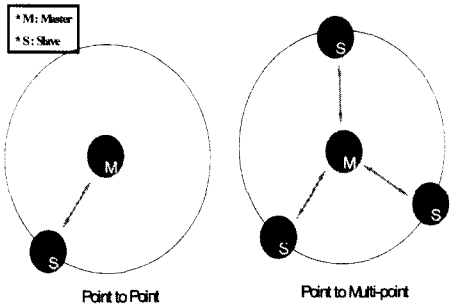


그림 1. 블루투스 피코넷.
Fig. 1. Bluetooth Piconet.

블루투스 개발 키트는 CSR BC01 모듈[7]이 탑재된 개발 키트를 사용하였다. 블루투스 개발 키트는 USB와 UART를 이용하여 호스트 머신과의 통신을 지원한다.

2. 리눅스 커널의 이식

본 논문에서는 임베디드 시스템을 위한 운영체제로 리눅스[15]를 사용하였다. 리눅스 커널은 GPL 라이선스로 배포되며 소스가 공개되므로 다양한 기기에 적합한 형태로 수정하여 사용할 수 있는 특징으로 임베디드 시스템을 위한 운영체제로 활용되기에 적합하다.

블루투스 스택으로는 리눅스 공식 블루투스 프로토콜 스택인 BlueZ 스택[5]을 사용하였다. BlueZ 스택은 리눅스 커널 2.4.6 버전 이상을 지원하므로 Target board에 기본적으로 적재되어있는 커널에는 BlueZ를 사용할 수가 없으므로 본 구현에서는 리눅스 커널[15] 2.4.18 버전을 이식하였다. 또한 GUI를 통한 응용 프로그램의 개발을 위하여 Trolltech의 QT Embedded[6]를 이식하였다.

표 2는 본 구현에서 구성한 Target board 내의 플래시롬의 구성을 보여주고 있다.

가. 리눅스 커널 패치 및 플래시롬 파티셔닝

SA1110 아키텍처에 리눅스 커널을 이식하기 위하여 Linux Arm Project[10]의 커널 패치를 적용하였다. 기본적인 커널 패치 이후 장치에 맞게 커널을 커스터마이징하는 과정이 필요하며 커널에서 이용하고자 하는 장치 및 리눅스를 운영하기 위한 기본 유틸리티를 램디스크 이미지에 생성해 주어야 한다.

플래시롬을 표 2와 같이 파티션으로 나누고 해당 파티션에 적용할 파일 시스템을 결정하고 해당하는 파일 시스템으로 포맷된 이미지를 만들어서 기록하게 된다.

표 1. 임베디드 시스템 개발 키트.
Table 1. Development kit for embedded Linux.

프로세서	Intel® StrongARM™ 1110 microprocessor @ 206 MHz
메모리	16MB FLASH 32MB SDRAM(100MHz)
LCD	Kyocera 5.7"
터치스크린	UCB1200 (1000x1000)
이더넷카드	Cirrus Logic CS8900A
UART	3개
OS	Linux kernel 2.4.0

표 2. 플래시롬의 구성.
Table 2. Memory map of the flash ROM.

파티션	주소 번지	저장 내용
1	0x00000000 ~ 0x00ffffff	부트로더 + 부트 스크립트
2	0x00100000 ~ 0x001fffff	리눅스 커널 이미지
3	0x00200000 ~ 0x002fffff	램 디스크
4	0x00300000 ~ 0x004fffff	블루투스 스택(BlueZ)
5	0x00500000 ~ 0x00ffffff	QT Embedded 환경, 응용프로그램

본 구현에서는 파일 시스템으로 램디스크 영역은 EXT2 파일 시스템[11]으로 블루투스 스택과 QT 임베디드 환경 및 응용 프로그램을 위한 영역은 CRAMFS 파일 시스템[12]으로 적용하였다.

CRAMFS는 압축을 통하여 좀더 많은 데이터를 플래시롬에 기록 가능하게 한다. 플래시롬의 파티셔닝 리눅스 커널에서 Memory Technology Device[13]를 통하여 가능하며 커널 컴파일 시에 이를 지원하도록 컴파일 하여야 한다. 이를 이용한 플래시롬의 파티셔닝은 KERNELDIR/drivers /mtd/maps/sa1100-flash.c 파일을 수정함으로써 가능하며 그림 2와 같이 적용하였다. 이후 각각의 파티션은 /dev/mtd0, 1, 2 ... 장치로서 접근 가능하며 이러한 장치를 램디스크에 생성하여 마운트를 하여야 한다. mtd 장치는 Major number 90 이며 Minornumber는 각각의 장치 이름에 붙여진 숫자와 동일하다.

```
#define CERF_FLASH_SIZE    0x01000000
static struct mtd_partition cerf_partitions[] = {
        {
                name:    "Boot Loader",
                size:    0x00100000,
                offset:  0,
        }, {
                name:    "Kernel",
                size:    0x00100000,
                offset:  0x00100000,
        }, {
                name:    "Ramdisk",
                size:    0x00100000,
                offset:  0x00200000,
        }, {
                name:            "/usr/local",
                size:            0x00200000,
                offset:          0x00300000,
        }, {
                name:            "/usr/qte",
                size:            0x00A00000,
                offset:          0x00500000,
        }
};
```

그림 2. sa1110-flash.c.
Fig. 2. sa1110-flash.c.

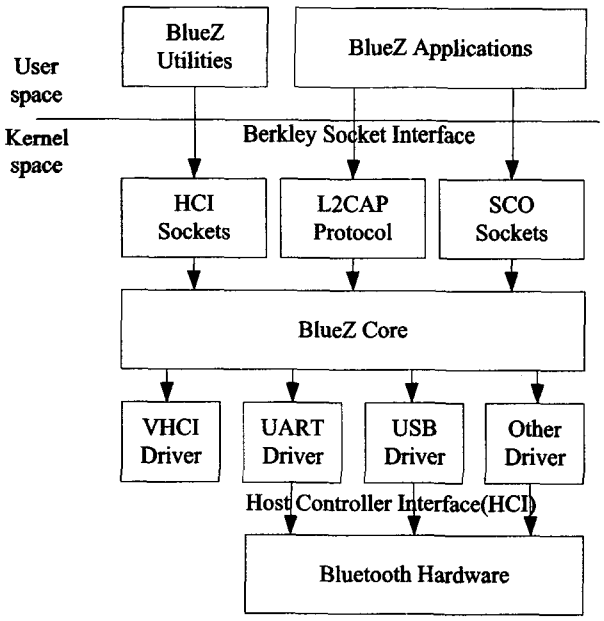


그림 3. BlueZ 프로토콜 스택의 구조.
Fig. 3. The Architecture of BlueZ protocol stack.

나. 블루투스 스택의 이식

블루투스 스택으로서 BlueZ[5]를 커널에 포함하기 보다는 커널 모듈로서 이를 포함하도록 하였다. 직접적으로 커널에 포함하는 것보다 모듈로 적용하면 BlueZ의 업그레이드를 위하여 커널 전체를 다시 컴파일 하는 과정을 불필요하게 한다. BlueZ는 커널 모듈, 라이브러리, 기본 유틸리티를 제공하고 있으며 본 구현에서는 BlueZ 커널 모듈과 라이브러리를 SA1110 아키텍처로 이식하였다. 구현에 사용된 버전은 각각 bluez-kernel-2.2, bluez-lib-2.0-pre9, bluez-utils-2.0-pre9 버전이다.

블루투스 모듈과 이를 제어하기 위한 호스트 간 통신은 블루투스 스택의 HCI(Host Controller Interface)계층을 통하여 통신한다. HCI는 UART 또는 USB 드라이버를 이용하여 블루투스 모듈과 호스트 간 데이터 전송을 지원하는데 본 논문에서는 UART를 이용한 통신을 하였다. BlueZ 스택의 구조는 그림 3과 같다.

SA1110 프로세서는 직접적으로 UART 통신을 지원하는데 UART의 기본적인 모드는 그림 4와 같다.

```
UART1 : GPCLK (default), UART
UART2 : Infrared Communication port
UART3 : UART
```

그림 4. SA1110 UART.
Fig. 4. SA1110 UART.

```
/dev/ttySA0, Major number 204, Minor number 5
/dev/ttySA1, Major number 204, Minor number 6
```

그림 5. SA1110 Serial 장치 파일.
Fig. 5. Serial device file for SA1110 architecture.

```
#!/bin/bash
#setting the name alias for BT modules
alias net-pf-31='bluez'
alias tty-ldisc-15='hci_uart'
alias bt-proto-0='12cap'

insmod bluez
insmod hci_uart
insmod l2cap

# HCID daemon start up
# BT device initialize
if [ -x /usr/local/Bluez/bin/hcid ]; then
        hcid
fi
if [ -x /usr/local/Bluez/bin/hciattach -a -c
/dev/ttySA1 ]; then
        hciattach ttySA1 csr 115200
        hciconfig hci0 up
fi
```

그림 6. 블루투스 모듈 및 장치 초기화.
Fig. 6. Initialization of Bluetooth modules and devices.

UART3는 커널 디버깅을 위한 시리얼 콘솔로 사용하였으며 UART1은 GPCLK 모드를 UART모드로 변환하여 사용하였다. 이를 변환하는 방법은 SA1110 레지스터 GPCLKR0 (0x8002 0060)의 SUS 비트의 값을 변경하는 것으로 가능하다. SUS비트가 0이면 GPCLK 모드로 동작하고 1이면 UART 모드로서 동작한다. 이를 변경한 이후 램디스크 이미지 내부에 다음과 같은 장치 파일을 생성함으로서 UART 장치를 접근할 수 있다.

그림 6은 Target board 부팅 시 자동으로 블루투스 모듈을 적재하고 장치를 초기화하기 위하여 작성한 쉘 스크립트이다.

다. GUI 구현을 위한 QT Embedded 툴킷의 이식과 UCB1200 장치를 통한 터치스크린의 제어

QT Embedded 툴킷[6]은 리눅스 프레임버퍼 장치를 이용하여 GUI 툴킷을 구현한 환경으로서 임베디드 환경에서 X 서버를 이용하지 않고 응용 프로그램을 제작하기 위한 라이브러리이다. QT Embedded를 이식하는 데에 있어서 리눅스 커널 프레임 버퍼 지원 옵션과 터치스크린 장치 드라이버의 커스터마이징이 필요하다. 터치스크린 장치로는 /dev/touchscreen/ucb1x00 장치를 Major number 10, Minor number 14 로 생성하였으며 장착된 터치스크린 장치가 1000x1000 해상도로 작동하므로 LCD의 해상도(640x480)와 매핑 될 수 있도록 조정하였고 QT에서 이 장치를 읽어서 작동하도록 구현하였다. 그림 7은 QT Embedded 툴킷에서의 QCustum TPanelHandlerPrivate의 장치를 읽어 들이는 부분에 대한 구현이다.

```
void
QCustomTPanelHandlerPrivate::readMouseData()
{
#ifdef QWS_CUSTOMTOUCHPANEL
    if(!qt_screen) return;
    CustomTPdata data;

    unsigned char data2[5];
    int ret;
    ret=read(mouseFD,data2,5);
    if(ret==5) {
        data.status=data2[0];
        data.xpos=(data2[1] << 8) | data2[2];
        data.ypos=(data2[3] << 8) | data2[4];
        QPoint q;
        q.setX(data.xpos);
        q.setY(data.ypos);
        mousePos=q;
        if(data.status & 0x40) {
          emit mouseChanged( mousePos,
                             Qt::LeftButton);
        } else {
          emit mouseChanged(mousePos,0);
        }
    }
    if(ret<0) {
        qDebug("Error %s",strerror(errno));
    }
#endif
}
```

그림 7. QTDIR/src/kernel/qwsmouse_qws.cpp.
Fig. 7. QTDIR/src/kernel/qwsmouse_qws.cpp.

라. GPIO 장치를 이용한 출력

가전기기 제어를 위한 입출력으로서 본 논문에서는 SA1110 아키텍처의 GPIO(General Purpose I/O)[14]를 이용하였다. 이를 위하여 /dev/GPIO 장치를 Major number 254, Minor number 0으로 하여 램디스크 이미지에 생성하였으며 GPIO 장치 드라이버를 구하여 Target board에 맞게 수정하였다. 전체 28개의 입출력 비트 중 터치스크린과 LCD 등의 장치에서 사용되는 출력을 제외한 8 비트의 입출력을 가전기기 제어를 위한 채널로 설정하였다.

표 3. GPIO 제어를 위한 레지스터.
Table 3. A register set for GPIO control.

주소	이름	역할
0x9004 0000	GPLR	GPIO pin level register
0x9004 0004	GPDR	GPIO pin direction register
0x9004 0008	GPSR	GPIO pin output set register
0x9004 000C	GPCR	GPIO pin output clear register
0x9004 0010	GRER	GPIO rising-edge detect register
0x9004 0014	GFER	GPIO falling-edge detect register
0x9004 0018	GEDR	GPIO edge detect status register
0x9004 001C	GAFR	GPIO alternate function register

Ⅳ. 가전기기 제어를 위한 블루투스 장비 간 피코넷 구성

본 논문에서는 가전기기에 내장되어 가전기기를 제어하면서 블루투스 접속을 유지하는 블루투스 장비를 메인 컨트롤러, 원격지에서 가전기기를 제어하기 위한 블루투스 장비를 클라이언트로 명명한다. 복수개의 클라이언트를 통한 제어와 블루투스 네트워크 상의 브로드캐스트를 통한 가전기기 상태 정보의 전송을 위하여 메인 컨트롤러는 Master, 클라이언트는 Slave가 되어야 한다.

가전기기 제어를 위한 블루투스 피코넷은 아래와 같은 두 가지의 방법으로 구성할 수 있다.

첫째, 메인 컨트롤러에 대한 제어를 원하는 클라이언트가 자신을 Inquiry Scan 상태로 만들고 하나의 메인 컨트롤러가 주기적으로 주위의 Inquiry Scan 상태에 있는 클라이언트를 조회(Inquiry)를 통하여 찾아내고 접속한다.

둘째, 블루투스 피코넷 범위 안에 있는 클라이언트가 조회(Inquiry)를 통하여 Inquiry Scan 상태의 메인컨트롤러에 Master로 접속하고 다른 클라이언트의 접속을 위하여 Role Switching 과정을 통하여 메인 컨트롤러를 Master로 동작하도록 역할을 변경한다.

가전기기를 위한 피코넷 구성에 있어서 첫 번째 방법은 클라이언트의 접속요구에 대한 시간이 Master기기의 Inquiry 주기에 의존하는 문제점을 가진다. 또한 주기적인 Master의 Inquiry과정은 네트워크의 부하를 초래한다.

본 논문에서는 두 번째 과정을 통하여 피코넷 구성을 하였다. 이러한 구성의 경우 메인 컨트롤러는 현재 구성하고 있는 클라이언트에게는 Master로써 동작하며 새로 접속을 요청하는 클라이언트에게는 Slave로서 동작하여야 한다. 이후 접속이 완료되면 Role Switch를 통하여 메인 컨트롤러가 Master로 동작하는 하나의 피코넷을 구성하고 가전기기의 상태 변화를 피코넷의 Slave인 복수개의 클라이언트에게 브로드 캐스팅 하여 동일한 정보를 유지할 수 있도록 하였다.

Ⅴ. SA1110 GPIO를 이용한 가전기기의 제어

가전기기를 제어하기 위한 GPIO 8bit를 통하여 많은 수의 제어정보를 전송하기 위하여 제어 시작 코드와 제어 종료 코드를 정의하고 시작코드와 종료 코드 사이에서 복수 Byte로 구성된 제어 코드를 가전기기에 전송하였다.

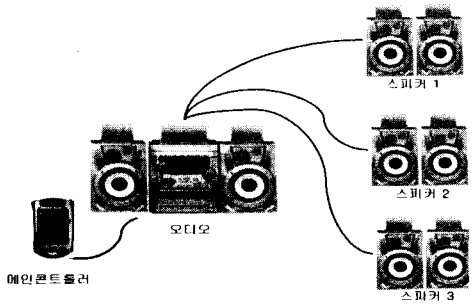


그림 8. 다(多)채널 오디오 시스템.
Fig. 8. A multi-channel audio system.

Ⅵ. 오디오 제어 시스템의 구현

1. 제어 대상 기기

본 논문에서 제어를 구현한 오디오 시스템은 각각의 스피커 별로 각각 다른 음향 자원의 출력이 가능한 오디오 시스템이다.

오디오 원격 제어 시스템은 원격지로부터의 제어 정보를 오디오 시스템에 전달하고 오디오의 상태를 브로드 캐스트 하는 메인 컨트롤러와 메인 컨트롤러를 통해 오디오 시스템을 제어하는 클라이언트인 월 플레이트로 구성된다. 월 플레이트는 가정이나 사무실의 각 방마다 하나씩 부착되어 있으며 각 방에는 유선으로 스피커가 연결 되어있다. 블루투스를 이용한 오디오 제어 시스템에서 각 방의 스피커를 유선으로 연결한 것은 블루투스의 전송속도 때문이다. 블루투스는 데이터 및 음성 전송을 지원하지만 음성 전송 속도가 낮아서 스피커의 음질이 저하될 수 있기 때문이다.

2. 응용 프로그램의 구현

가. BlueZ 라이브러리를 이용한 통신

BlueZ 라이브러리[5]는 일반적인 소켓 프로그래밍 방식과 동일 한 방식을 제공한다. 단지 소켓 생성시 bluetooth.h와 l2cap.h에서 제공하는 옵션을 사용하여 소켓을 생성하고 소켓의 옵션을 지정한다. HCI 계층의 명령을 위한 함수는 hci_lib.h에서 정의를 하고 있다. L2CP 소켓의 생성과 소켓 옵션의 설정의 예를 그림 9에서 나타내었다.

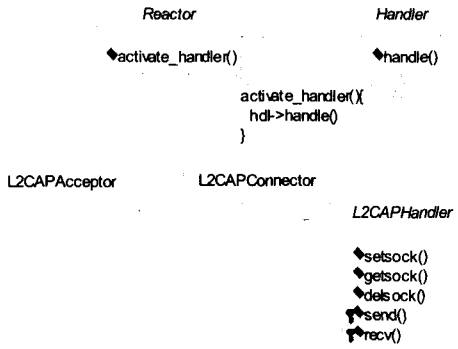


그림 10. Class diagram : L2CAPAcceptor and L2CAPConnector.
Fig. 10. Class diagram : L2CAPAcceptor and L2CAPConnector.

```
void handle() {
  L2CAPSock sock = getsock();
  this->recv(sock,buf,size);
  ...
  this->send(sock,buf);
  ...
  delsock(sock);
```

그림 11. handle 메소드 구현의 예
Fig. 11. Example codes : handle method.

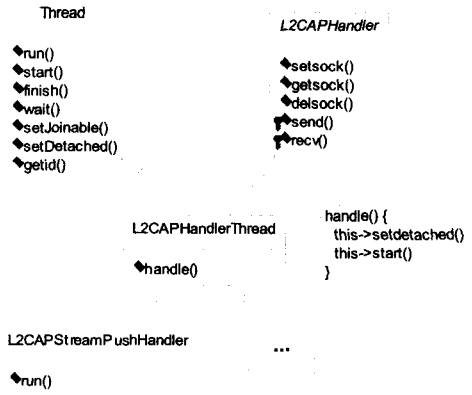


그림 12. Class diagram : L2CAPHandlerThread.
Fig. 12. Class diagram : L2CAPHandlerThread.

본 논문에서는 C++ wrapper 라이브러리를 구현하고이를 이용하여 응용 프로그램을 작성하였다. 그림 10은 L2CAPAcceptor와 L2CAPConnector의 구현에 대해서 다루고 있다. Reactor 클래스의 activate_handler를 호출하면 등록된 Handler의 handle 메소드가 호출 되며 이를 상속 받은 L2CAPHandler에서는 L2CAP 통신한 L2CAP 소켓을 위한 메소드를 구현하고 있다.

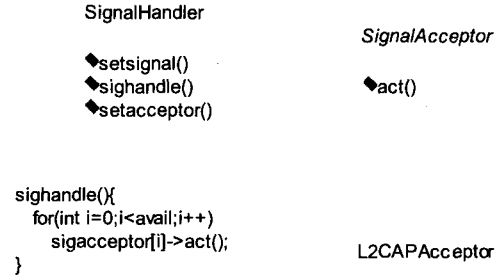


그림 13. 시그널 처리를 위한 L2CAP- Acceptor 구현.
Fig. 13. Class diagram : L2CAPAcceptor for signal handling.

```
L2CAPStreamPushHandler hdl;
SignalHandler sighdl;
try {
   L2CAPAcceptor acceptor(&hdl, 8000);
   sighdl.setsignal(SIGINT);
   sighdl.setacceptor(&acceptor);
   acceptor->tstart();
   ...
```

그림 14. L2CAPAcceptor 사용 예.
Fig. 14. Example codes : L2CAPAcceptor.

L2CAP 통신을 위한 핸들러를 구현하고자 할 때 L2CAPHandler를 상속받아서 handle 메소드를 구현해 주고 L2CAPAcceptor나 L2CAPConnector에 핸들러를 등록하여 사용하면 된다. L2CAPHandler는 유도 클래스를 위한 send와 recv 메소드를 지원한다. 그림 11은 handle 메소드 구현의 예이다.

쓰레드 처리가 필요한 핸들러를 위하여 따로 L2CAPHandlerThread 클래스를 정의하였다. 아래의 그림 12는 L2CAPHandlerThread를 보여주고 있다.

그림 11의 핸들러 구현 예제와 마찬가지로 쓰레드 처리가 필요한 핸들러는 L2CAPHandlerThread를 상속받아서 run 메소드를 구현해주고 start 메소드를 호출하면 새로운 쓰레드를 생성하여 처리 할 수 있다. Thread 클래스는 pthread를 이용하여 구현하였다.

L2CAPAcceptor는 Reactor와 SignalAcceptor를 다중 상속 받아서 구현하였으며 다른 프로세스로부터 시그널을 받았을 경우 시그널 핸들링을 act 메소드를 구현함으로서 처리할 수 있다. 그림 13는 이에 대한 클래스 다이어그램이다. 이러한 구조를 통하여 그림 14와 같은 형태로 L2CAPAcceptor를 사용할 수 있다.

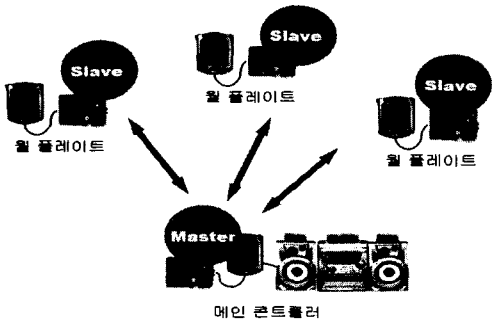


그림 15. 피코넷을 형성한 오디오 제어 시스템.
Fig. 15. Piconet for audio control system.

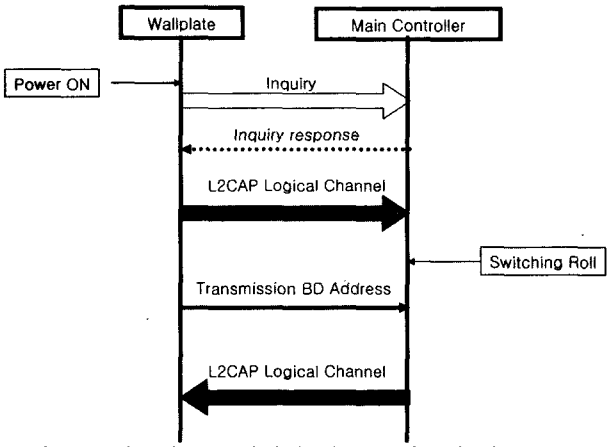


그림 16. 피코넷을 구성하기 월 플레이트와 메인 컨트롤러 간 통신.
Fig. 16. a communication flow between main con- troller and wallplate for Piconet formation.

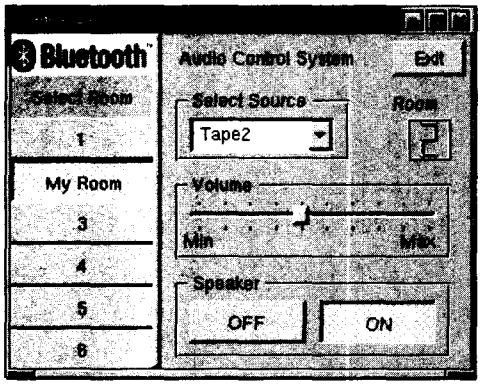


그림 17. 월 플레이트의 제어화면.
Fig. 17. the user interface of wallplate.

나. 메인 컨트롤러와 월 플레이트 간 통신 응용 프로그램

가전기기 제어에 있어서 접속하는 기기는 블루투스 모듈이 장착된 모바일 기기나 혹은 본 논문에서의 월 플레이트와 같은 클라이언트 기기이다. 블루투스 피코넷에서는 접속을 시도하는 기기가 Master이며 접속을 당하는 기기가 Slave이다. 하나의 가전기기를 여러 기기에서 접속해서 조작하려면 가전기기에 부착된 메인 컨트롤러는 Master로 동작해야 한다. 만일 Slave로서 동작한다면 다른 클라이언트 기기는 가전기기에 접속을 할 수 없다. 이는 두개의 Master의 시스템 클럭에 하나의 Slave기기가 동기화 될 수 없기 때문이다. 그림 15는 오디오 시스템에서의 블루투스 피코넷을 개략적으로 나타내었다.

응용 프로그램에서 피코넷을 구성하는 상세한 과정은 그림 16과 같다.

월 플레이트는 조회(Inquiry)를 통하여 메인 컨트롤러의 블루투스 디바이스 주소를 얻어오고 이를 통하여 L2CAP 논리 채널을 구성한다. 이후 Roll Switching 과정을 통하여 Master와 Slave의 역할을 바꾸고 월 플레이트는 자신의 블루투스 디바이스 주소를 메인 컨트롤러에게 전송한다. 마지막으로 메인 컨트롤러에서 월 플레이트로 L2CAP 논리 채널을 구성하여 상태 변화를 브로트 캐스팅하기 위한 채널을 확보하고 현재 오디오 상태를 전송 받아서 LCD화면에 표시하게 된다.

피코넷을 구성한 임의의 월 플레이트가 메인 컨트롤러에게 변경하고자 하는 정보를 전송하면 메인컨트롤러는 GPIO를 이용하여 오디오 기기의 상태를 변경하고 변경된 상태를 브로드캐스팅을 위한 L2CAP 논리 채널을 통하여 피코넷을 구성하고 있는 복수개의 월 플레이트에게 변경된 오디오 상태를 브로드 캐스팅 한다.

그림 17은 오디오를 제어하기 위해 구현한 간단한 월 플레이트 프로그램의 인터페이스를 나타내었다. 월 플레이트 프로그램은 메인 컨트롤러를 구성한 장비와 동일한 Cerf Board를 이용하여 구현하였으며, 터치스크린을 이용한 사용자 인터페이스를 통하여 원격지 오디오 시스템의 음원 소스나 자신의 방에 있는 스피커의 볼륨 및 전원을 제어할 수 있으며 다른 방의 상태를 모니터링하고 조작 할 수 있다.

Ⅶ. 결론 및 추후 연구

본 논문에서는 임베디드 환경에서 블루투스를 이용한 가전기기 제어를 구현하였으며 블루투스 피코넷의 응용을 제시하였다. 이러한 가정이나 사무실 등에서의 가전기기에 대한 기초 제어는 이후 휴대기기 PDA나 휴대폰 등에 내장 되어질 경우 다양한 형태로 활용 가능하다.

하지만 이후 피코넷의 제한사항을 극복하는 스캐터넷

의 적용과 다양한 가전기기들에 대한 제어를 위하여 블루투스 스펙에서 제안되고 있는 서비스 발견 프로토콜(Service Discovery Protocol)을 이용한 가전기기 제어에 대한 연구의 확장이 필요하다.

참 고 문 헌

[1] Jennifer Bray and Charles F Sturman, "Bluetooth", Prentice Hall, pp. 65-90 2001.

[2] Brent A. Miller and Chatschic Bisdikian, "Bluetooth Revealed", Prentice Hall, pp. 15-20 2001.

[3] Bluetooth SIG, "Specification of the Bluetoo th System", Bluetooth SIG

[4] Linux.Device.com, "Linux Device", URL: htt p://www.linuxdevices.com

[5] Maksim Krasnyanskiy, "BlueZ", URL: http:// bluez.sourceforge.net

[6] Trolltech, "QT", URL: http://www.trolltech .com

[7] Cambridge Silicon Radio, "Cambridge Silicon Radio", URL: http://www.csr.com

[8] Bluetooth SIG, "Bluetooth SIG", URL: http://qualweb.bluetooth.org

[9] Craig Hallabaugh, "Embedded Linux", Addison Wesley, pp. 21-63, 2002.

[10] Arm Linux Project, "Arm Linux Project" URL: http://www.arm.linux.org.uk

[11] Theodore Ts'o, "EXT2 File System", URL: http://e2fsprogs.sourceforge.net

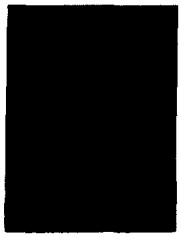
[12] CRAMFS Project, "CRAMFS", URL: http:// sourceforge.net/projects/cramfs

[13] David Woodhouse, "Memory Technology Device", URL: http://linux-mtdinfradead.org

[14] Intel, "Intel® StrongARM* SA-1110 Mic roprocessor Specification", Intel, 2001.

[15] Linux Kernel, "Linux Kernel", URL: http:// www.kernel.org

저 자 소 개



이 우 중(학생회원)

2002년 한양대학교
화학공학과 졸업.
2004년 한양대학교
컴퓨터공학과 석사 졸업.
현재 포항공과대학교
정보통신연구소 연구원
<주관심분야 : 분산 객체 컴퓨팅, 임베디드 시스템>

김 정 선(정회원)

1986년 서울대학교
컴퓨터공학과 학사 졸업.
1988년 Iowa State University
컴퓨터공학과 석사 졸업
1994년 Iowa State University
컴퓨터공학과 박사 졸업
현재 한양대학교 컴퓨터공학과 부교수
<주관심분야 : 분산 객체 컴퓨팅, 병렬/분산처리>

황 우 식(정회원)

2001년 대구대학교
제어계측공학과 졸업.
2003년 한양대학교
컴퓨터공학과 석사 졸업.
현재 포스데이타 POSPIA SM부
<주관심분야 : 분산 객체 컴퓨팅, 임베디드 시스템>