# 블루투스 무선통신을 이용한 USB제품 테스트 시스템

권오상* · 박춘명**

## 1. 서 론

오늘날 윈텔(Windows+Intel)로 대표되고 있는 개인용컴퓨터(PC)는 1970년대에 등장한 애플사(Apple)에 의해 발표된 "애플"을 무너뜨리기 위해 IBM에서 개방형 구조를 갖는 개인용 컴퓨터를 1980년대 초에 발표하면서 시작되었다. IBM은 시장에 나와 있는 마이크로형 부품들을 조합할 수 있는 개방형 표준구조를 제정하였고 인텔의 CPU(8086)와 마이크로소프트의 운영체제(MS-DOS)를 채택하여 본격적인 PC시대를 맞이하게 되었다. IBM이 채택한 개방형 구조는 표준화된 부품들로 조립만하면 누구든지 PC를 만들 수 있었으므로 수많은 IBM 호환 PC업체들을 생겨나게 했고 이때부터 PC는 놀라운 속도로 급성장을 하였다.

1990년대초 IBM의 독자적 구조시도가 호환PC 제조업체 및 주변기기 제조업체의 개방형 구조 사수 결의에 부딪히고 응용소프트웨어와 주변기기의 부족으로 실패로 끝났다.

마이크로소프트사의 "Windows 3.1"과 인텔의 "Pentium"프로세서의 발표로 윈텔시대를 맞이한 PC는 인터넷환경이 확산되면서 사무용으로만 머물러 있지 않고 가정을 위시한 여러 영역으로 확산되기 시작하였다.

초기의 PC사용자들은 숙련된 지식인이나 전문인들이었다. 이후 워드프로세나 스프레드시트 등의 응용프로그램의 등장으로 PC는 회계업무나 사무용으로 활용되기 시작했으며 아이콘(Icon)에 의한 GUI(Graphic user interface)라는 사용자에게 쉽고 친숙한 사용자인터페이스를 갖춘 매킨토시와 윈도우의 등장은 근거리 통신망(LAN)과 더불어 사무자동화(OA)시대를 열었다.

주로 사무용으로 인식되어 오던 PC는 90년대에 이르러 멀티미디어와 인터넷이라는 화두를 접하면서 급격히 발전하여 오락과 교육의 용도로 확장되면서 가정용으로 확산되게 되었다.

따라서 최근에는 PC의 보급률이 높아졌고 이러한 기술의 발전은 컴퓨터의 주변기기를 발전시키는 견인차 역할을 해왔으나 주변기기 인터페이스는 여전히 복잡하고 다양한 양상을 띠고 있다. 또한 디지털오디오나 컴퓨터 전화통합과 같이 몇년전만 해도 꿈도 꾸지 못할 애플리케이션으로 인해 PC주변기기 포트의 용량이 한계에 이르렀고 사용의 편리성에 있어서 여러 가지 문제가 발생하고 있다. 예로서, PC에 접속되는 주변기기는 그 종류에 따라 커넥터나 케이블이 다르고 주변기기의 증가에 따라 PC가 갖추고 있는 커넥터의 수가 부족하다는 것을 들 수 있다. 또한 새로운 장치의 장착 및 기존장치의 탈착에 있어서 컴퓨터 케

* (주)한울로보틱스
** 충주대학교 전기 · 전자 및 정보공학부 컴퓨터공학 전공

이스를 열어야 하고 전원을 차단해야 되는 여러 가지 불편한 상황이 전개되고 있다. 이러한 단점들을 개선하고자 7개의 회사들(컴팩, DEC, IBM, 인텔, 마이크로소프트, NEC, Nortel)에 의해 개발된 것이 USB(Universal Serial Bus)이고 현재 차세대 표준으로 자리를 잡고 있다.

USB는 지속적으로 발전하고 있으며 Spec. 2.0부터는 480Mbps를 지원하므로 PC주변의 거의 모든 장치들은 USB 인터페이스로 대체될 것으로 보인다. 현재 500개 이상의 제품이 세계전역에서 개발되었거나 개발 중에 있으며 이는 Host PC 시스템을 위시하여 프린터, 스캐너, 키보드, 마우스, 조이스틱, 게임패드, 비디오카메라, 스틸화상 카메라, 전화기, 모뎀, 적외선 장치, ISDN어댑터, 마더보드, 측정장치, 진단시스템, 무선장치, 파워 시스템 등으로 확산되고 있다.

USB의 주요장점은 다음과 같다.

① 확장성 : PC주변장치 확장이 용이함(127개 device 지원)
② 고속화 : 12Mbps의 고속전송 및 저가격 실현 (USB 2.0 : 480Mbps 가능)
③ 대용량 : 음성과 압축된 비디오에 대한 실시간 데이터에 대한 완벽한 지원
④ 핫 플러깅(Hot PnP) 지원
⑤ 리얼타임성이 확보된 전송방식
⑥ 저속기기의 기존 표준 인터페이스 (RS232C, PS/2, Centronics I/F)를 대체

일반적으로 생산현장에서 제품을 생산 시에는 최종적으로 제품의 기능을 테스트하는 과정이 요구된다. 전 세계적으로 500여종 이상의 제품이 출시되고 있는 USB제품을 생산할 때도 이와 같은 과정이 요구된다. 현재 USB 제품의 생산시 테스트를 위한 방법은 주로 USB 호스트 역할을 하는 PC를 사용하며 이로 인하여 PC 설치비용 및 유선 환경 구축비용 등 생산제품 테스트를 위한 부가비용이 커지게 된다. 이러한 방식의 문제점은 비용적인 측면이외에도 각 생산라인에 근무하는 사람은 전자나 PC환경에 익숙하지 않으며 모니터 등을 포함하여 테스트하는 장비의 장소차지도 무시할 수 없다. 또한 PC의 하드웨어나 운영체제의 불안정성에 기인하여 테스트를 위한 PC가 자주 문제를 야기시킨다. 따라서 소형 사이즈의 전용테스트장비가 요구되며 컴퓨터에 익숙하지 않아도 사용하기가 용이한 장비가 요구된다. 또한 시험결과는 단순하게 표현되어 작업자가 이에 대한 조치를 쉽고 빠르게 할 수 있어야 한다.

개발된 USB 제품 호스트 테스트 장비는 Embedded 형태의 USB호스트를 개발하여 다양한 USB 제품생산시 테스트 비용의 절감과 전용장비로서의 역할을 할 수 있도록 하였다. 또한 USB 제품의 종류가 매우 다양하고, 한 개의 생산라인에서 취급하는 모델도 다수이기 때문에 각각의 테스트 모듈을 블루투스로 연결하여 PnP과정에서 인식되는 제품 ID, Product ID에 따라 각 제품별로 적합한 테스트 프로그램을 다운로드 받아 테스트를 수행하게 하면 다양한 USB 제품을 동시에 테스트할 수 있고, 이때 인식되는 ID들을 통하여 제품의 생산관리가 가능하다. 또한 OS에 의해 걸리는 시간을 단축하여 빠른 시간에 원하는 테스트가 수행되어야 생산성의 향상을 가져올 수 있다.

## 2. USB 테스트 시스템

### 1. USB 테스트 시스템 구성

USB 테스트 시스템의 구성은 아래의 그림 1과 같다.

시스템의 블록도를 보면 전체 시스템을 제어하

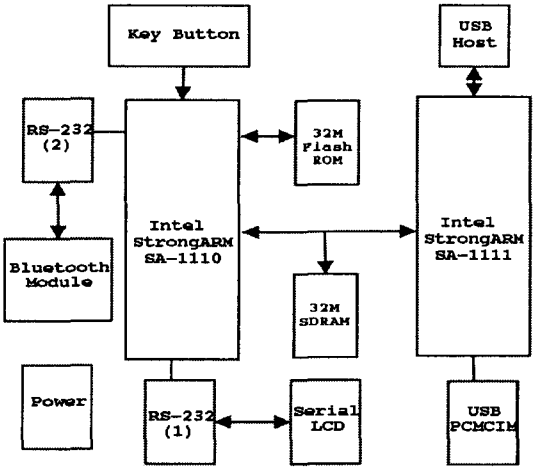


그림 1. USB 테스트 시스템의 블록도

는 마이크로 프로세서(SA-1110)와 USB 장치(USB 마우스, USB 키보드, USB 카메라 등)의 데이터를 읽을 수 있는 SA-1111 그리고 SA-1111를 통하여 읽은 데이터를 무선으로 서버의 컴퓨터에 전송하는 블루투스 모듈로 구성되었다. 또한 사용자 인터페이스로는 테스트 결과를 확인할 수 있는 디스플레이(LCD)와 사용자의 입력을 받는 키 버튼으로 구성되었다. USB 테스트 시스템의 동작은 순서는 먼저 USB 장치가 접속되면 SA-1111를 통하여 USB 장치의 데이터를 읽고 이 데이터가 올바른 것이지 테스트하고, 그 결과를 사용자가 확인 할 수 있도록 LCD에 표시하다. 또한 근 거리에 있는 서버용 PC에 테스트 결과를 무선으로 전송하기 위해 블루투스 모듈를 사용한다.

아래는 USB 테스트 시스템의 사양과 실제 구성된 장비를 나타내었다.

- H/W
- StrongARM(SA-1110, SA-1111)
- Clock 속도 : 220MHz
- Flash ROM(32 MB)
- SDRAM(32MB)
- Serial Port 2개(UART1, UART1)
- S/W
- OS : Linux Kernel 2.4.2
- USB 테스트 프로그램
- Bluetooth 사양
- 주파수 대역 : 2.4GHz
- Interface 방식 : UART(115200bps)
- 송수신 범위 : 10m
- 전송속도 : 최대 712Kbps
- Bluetooth Stack : Axis 사용(Linux 용)
- LCD
- 한글 : 16 * 8
- 영문 : 32 * 8

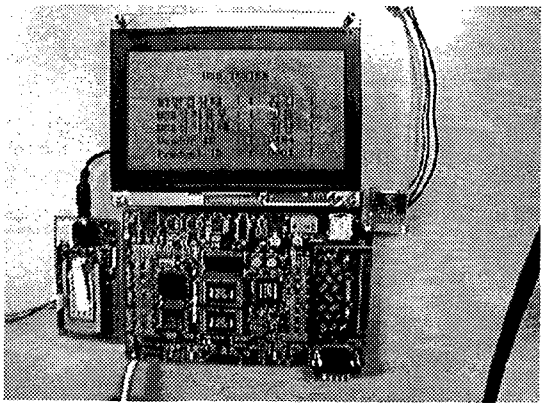

그림 2. 제작된 USB 테스트 시스템

가. 메인 컨트롤러

1) StrongARM

스트롱암(StrongARM)은 웹패드, 개인정보단말기(PDA) 등의 소형 정보기기에 탑재되는 인텔의 임베디드 리스크(RISC) 프로세서이다. 스트롱암의 전신은 마이크로프로세서 개발업체인 영국 ARM사와 미국의 DEC가 공동으로 개발한 ARM 프로세서로, 인텔은 자사가 갖추지 못한 모바일 프로세서 제품군을 보완하기 위해 ARM사와 라이선스 계약을 체결하고 스트롱암 1.1을 공급하고

있다. 한편 코어 라이선스 영업을 하고있는 ARM 사는 인텔 외에도 히타치, 록웰, TI, 디지털, 소니, 필립스, LSI 등 유명 반도체업체들과 라이선스 계약을 체결되었다. 스트롱암 프로세서는 기존 ARM 코어를 적용한 제품보다도 뛰어난 성능을 제공하는 것이 특징이며, 소비전력을 기존 칩의 2분의 1 수준인 40~450mW로 낮추고 구동 주파수를 150~600MHz까지 높일 수 있다. 이에 따라 185~750MIPS(초당 100만개의 명령을 수행하는 연산속도 단위)의 데이터를 처리 성능을 제공한다.

SA-1110은 스트롱암 중에서 가장 빠른 마이크로 프로세서로 SA-1 코아를 제공한다. 또한 외부에 인터페이스 할 수 있는 시리얼 포트 2개와 LCD 컨트롤로, IrDA, CODEC, 메모리와 PCMCIA 컨트롤로 모듈이 내장되었다. 그리고 전류 소비를 최소화하기 위하여 전원 관리 부분이 따로 존재한다.

2) 블루투스 인터페이스

현재 주류를 이루고 있는 블루투스 칩은 3칩 세트로 가격이 바싸다는 점과 배터리 소모율이 높다는 단점을 지니고 있다. 이러한 단점을 해결

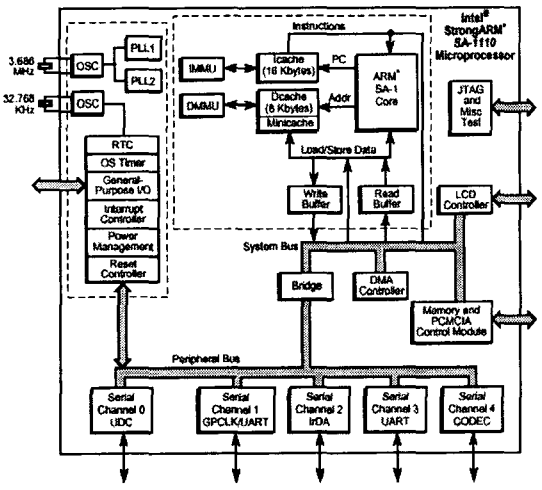


그림 3. SA-1110의 블록도

표 1. SA-1110의 특징

| | SA-1110 |
|---|---|
| Clock | 206Mhz |
| Core Power Supply | 1.75V |
| 소비 전력 | 400mW 이하 |
| I/O Interface | 3.3V |
| Mode | Sleep current = <50uA |
| | Idle power =<40mW |
| Package type | 256-pin mini BGA |
| cashes | Instruction cashe : 16Kbyte<br>Data cash : 8Kbyte |
| Integrated clock generation | 3.686Mhz : CPU core<br>32.768Mhz : RTC or Power Manager |

하기 위해 세계적으로 단일 칩 생산을 연구개발, 양산화 직접단계에 와있다. 블루투스 칩은 소형화, 저렴화, 저전력화를 목표로 디자인을 추구해야한다. 또한 빠른 주파수 호핑(Hopping)과 단일칩 통합기술을 확보해야 한다.

저 가격의 전파기술을 구현하기 위해서는 단일칩, 오프칩 컴포넌트, 메인스트림 기술, 시분할 듀플렉스 등 다양한 기술이 요구된다.

블루투스 기능 구현에는 대기전류 0.3mA, 음성모드전류 10mA, 데이터전송모드전류 6mA가 필요하다. 이러한 블루투스를 실현하기 위해서는 이동전호단말기, PDA 등의 전원을 활용하거나 저전력을 구현하는 배터리 및 전원회로 설계 기술이 연구돼야 한다.

현재 알카텔과 CSR이 1칩 CMOS 솔루션을 선보이고 있으며 커넥서트, 에릭슨, 루슨트테크놀로지스, 필립스, 실리콘웨이브 등이 다양한 방식의 블루투스 칩을 개발 발표하고 있다.

사용된 CSR 모듈의 인터페이스는 UART와 USB 모두 지원한다. 개발된 USB 테스트 장비에는 UART를 사용하였다.

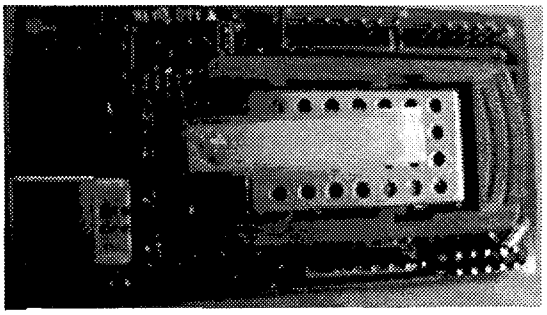

그림 4. 개발된 블루투스 보드

그림 4는 CSR 모듈을 사용하여 개발한 보드이다.

(1) 블루투스 스택

AXIS사에서 개발한 OpenBT는 개방형 블루투스 스택이다. 이들은 이 스택을 그들의 임베디드 시스템과 액세스 포인트를 위해 개발하였다. 스택 자체만 놓고 본다면 그리 완전하다고는 볼 수 없지만 오픈소스이기 때문에 용도에 맞게 고칠 수 있다는 장점을 지니고 있고 현재도 메일링 리스트를 통해서 논의되고 발전되고 있는 스택이다.

Axis 스택에서 PPP (or User program)을 사용하는 경우에는 다음의 형태를 갖는다.

포트 드라이버 인터페이스는 블루투스 문자 디바이스 드라이버 인터페이스로써 ttyBT0~ttyBT6로 표현된다. 그리고 여기에 제어 통로인 ttyBTC의 라인을 에뮬레이션 시킨다. 각각은 초기화 루틴과 처리 루틴으로 이루어져 있으며 데이타 형식은 Core Specification을 따른다. 스택의 구현은 HCI, L2CAP, RFCOMM, SDP, TCS가 구현되어 있으며 본 시스템에 사용되지 않는 TCS는 설명에서 제외했다. 호스트와 스택의 제어는 ttyBTC를 이용하고 데이터는 ttyBT0~ttyBT6중의 하나를 선택해서 한다.

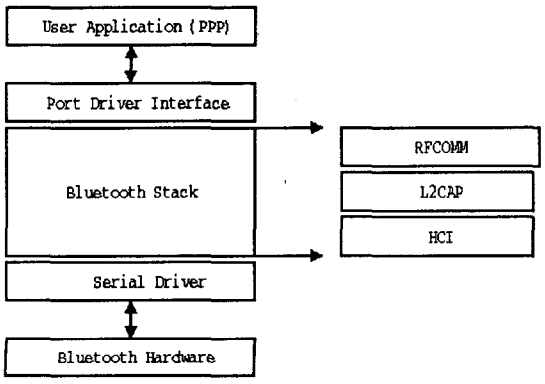


그림 5. Axis 블루투스 스택

(가) RFCOMM

스택의 맨 상위에 위치하는 프로토콜이다. 먼저 이 프로토콜을 사용하기 위해서는 RFCOMM을 초기화하는 단계가 필요하다. 이는 L2CAP위에 RFCOMM을 위치시키는 과정이다. 현재 RFCOMM의 역할에서 해야 하는 것은 상위 레이어에서 받은 명령을 하위 레이어인 L2CAP에 전달하는 것이므로 L2CAP의 중요한 프로시져들을 함수로 묶어 놓았다. 예를 들면 connect의 명령을 받으면 rfcomm_connect_req의 함수를 부르게 하여 L2CAP의 상태와 유사하게 rfcomm의 상태에 따라서 프로시져를 정의한다. 그리고 이러한 프로시져에 대응하는 프로시져도 함수로 묶여서 정의된다. 전체적인 RFCOMM 블록 다이어그램을 살펴보면 스테이트는 DISCONNECTING, DISCONNECTED, CONNECTING, CONNECTED, NEGOTIATING으로 나뉘고 이러한 상태는 이벤트에 의해 변하게 된다.

그림 6은 RFCOMM의 전체 스테이트 머신이다. Connection Request같은 이벤트와 Connection Confirm이나 Send 와 Receive 데이터 같은 것은 모두 함수로 짜여있다. 초기상태는 Disconnect상태이다. 이 경우 상위 레이어에서 Connection 요구나 하위 레이어에서 Connection indicate의 이벤트가 일어나면 Connecting의 상태로 전이된다. 이 경우 RFCOMM은 Connection Confirm 연결 확인을 수행하여 Connection 프로시져의 다음 단계를 준비한다. 이 바로 다음 단계

로 협상(Negotiating)의 상태로 전이되는데 이것은 Control 채널이 열리면 바로 진행된다.(이는 L2CAP이 RFCOMM으로 알려준다.) 이 상태에서 설정을 서로 협의하게 되고 이것이 완료되었을 때 설정 확인을 수행 한다. 그리고 연결된 상태로 진행한다. 연결된 상태는 데이타 채널이 열렸음을 의미하며 이 상태에서 데이타를 주고받을 수 있다. Disconnect request나 Disconnect indicate신호를 받았을 경우(request는 상위 레이어로부터의 요청이고 indicate는 L2CAP으로부터의 신호이다.) Disconnecting의 상태로 전이되고 조건을 테스트한 후 Disconnect Confirm을 수행한다. 그리고 마지막으로 DISC 신호를 발생시키거나 받음으로써 Disconnected의 상태로 전이한다. 이상에서 보듯이 RFCOMM은 상위 레이어의 요청에 반응이나 하위 레이어인 L2CAP으로부터의 신호를 받아서 상태를 변화시키면서(이는 L2CAP의 상태와 밀접한 관련이 있다.) 원하는 작업을 수행한다.

(나) L2CAP

L2CAP은 RFCOMM과 HCI의 중간에 위치하는 프로토콜이다. 이 때 HCI가 단순히 인터페이스라고 생각하면 실제로 하드웨어나 펌웨어와 통신하는 부분이라고 할 수 있을 것이다. L2CAP은 블루투스 스택상에서 많은 변화가 있을 수 있는 부분으로 이는 여러 가지 프로토콜을 통합하고 분리하는 일을 주요한 일로 한다. RFCOMM과의 통신으로 어플리케이션으로부터의 요청에 대응하게 된다. 그러므로 L2CAP과 RFCOMM과의 연동으로 생각했을 때 L2CAP에서 여러 개의 프로토콜에 대해 관리해야 하므로 이들을 등록시키는 절차가 필요하다. 이제 L2CAP의 스테이트 머신을 Initiator의 관점과 Acceptor의 관점에서 살펴보자.

그림 7은 L2CAP의 연결시의 Connect를 시작하는 쪽의 스테이트 머신을 나타낸 것이다. Open 상태가 서로 통신이 가능한 상태이다. 먼저 초기 위치는 Close에 위치한다. 이제 이 상태에서 L2CAP connection Request의 요청을 상위 레이어로부터 받으면 L2CAP은 L2CAP connection Request의 과정을 수행한다. 이 때 상태는 wait 상태로 전이된다. 이 wait 상태에서 L2CAP connect response를 기다리게 되는데 이 이벤트가 발생하게 되면 Config 상태로 전이가 되고 L2CAP config request를 하위 레이어로 전달한다.

L2CAP config response neg신호를 하위 레이어로부터 전달받게 되면 계속 이 상태에 있으며 L2CAP config request 신호를 보내게 되고 만약 L2CAp config response 신호를 받게 되면 Open 상태로 전이가 되면서 L2CAP config confirm의 과정을 수행하게 된다. 이제 Open 상태이므로 패킷을 주고받을 수 있는 채널이 형성된 것이다. 이

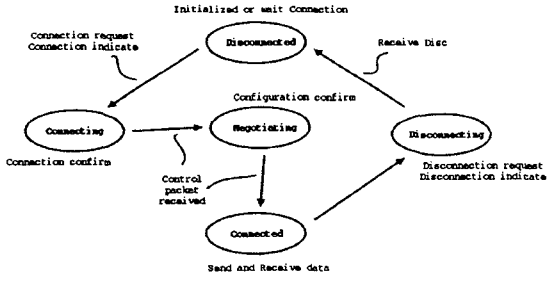


그림 6. Axis 스택에서의 RFCOMM 스테이트 머신

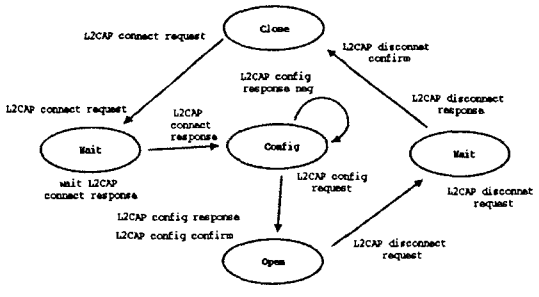


그림 7. L2CAP Initiator 스테이트 머신

제 주고받는 일이 끝나면 L2CAP disconnect request를 상위 레이어로부터 받게 되고 L2CAP은 L2CAp disconnect request를 수행하면서 wait 상태로 들어간다. 이제 L2CAP disconnect response를 받게 되면 L2CAP disconnect confirm을 하면서 Close상태로 돌아간다.

그림 8은 L2CAP이 Close상태에 있는데 L2CAP connect Request를 하위 레이어로부터 받았을 경우 스테이트 머신을 나타내고 있다. 이 Connect Request를 받으면, L2CAP connect Indicate를 수행하고 Wait의 상태로 들어가 L2CAP connect response를 기다린다. 이제 L2CAP connect response를 받으면 config의 상태로 들어가는데 이때 L2CAP config request를 받았을 경우 조건이 만족하지 않으면 L2CAP config response neg를 수행하고 만족하면 L2CAp config response를 수행하며 Open의 상태가 된다. 그리고 l2CAP disconnect request를 하위 레이어로부터 받게 되면 L2CAP disconnect Indicate를 수행하며 Wait 상태로 전이되고 이제 L2CAP disconnect response를 수행하면서 Close상태가 된다.

(다) HCI

스택의 맨 아래에 위치한 레이어이다. 레이어라기보다는 인터페이스의 성격이 강한 HCI는 일종의 구문 분석을 하는 컴파일러라고 해도 무방할 것 같다. 실제 link Manager나 Link Controller에 접근할 수 있는 명령어의 집합인 HCI는 L2CAP으로부터 명령을 받아서 그것에 해당하는 것을 일을 수행한다. 실제 스택의 구현에서는 읽어들이는 부분과 보내는 부분으로 나누어서 구현했다. 이는 상위 레이어에서 받은 L2CAP 패킷을 HCI 커맨드 패킷과 HCI 이벤트 패킷 그리고 HCI 데이타 패킷으로 구문 분석하는 부분과 하위 레이어로부터 받은 HCI 패킷을 L2CAP 패킷의 형태로 바꿔주는 역할을 한다. 기능적인 부분으로 크게 나누면 다음과 같은 3개의 기능 블록으로 나누어 줄 수 있다.

HCI Receive Block은 하위 레이어로부터 HCI 패킷형태를 가져와서 먼저 데이타 패킷인지 커맨드 패킷인지 이벤트 패킷인지 확인하고 이벤트 패킷이면 HCI 이벤트 Process Block으로 넘겨준다. 만약 데이타 패킷이나 커맨드 패킷이면 L2CAP 패킷형태로 L2CAP으로 넘겨주는 일을 한다. HCI Send Block은 상위 레이어로부터 L2CAP 패킷형태를 넘겨받아서 이의 type을 결정해서 이벤트 패킷이면 HCI 이벤트 Process Block으로 넘겨주고 만약 커맨드 패킷이나 데이타 패킷이면 하위 레이어로 전달한다.

HCI 이벤트 Process Block은 이벤트에 대한 처

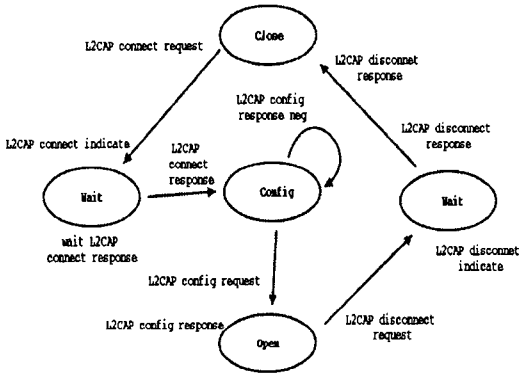


그림 8. L2CAP Accept 스테이트 머신

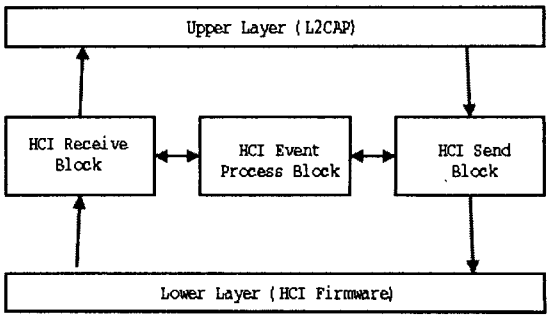


그림 9. HCI 기능 블록

리를 수행한다.

(라) 응용 프로그램

응용프로그램은 시리얼 프로파일에 근거하여 작성되었다. 프로파일은 한마디로 블루투스를 어플리케이션에서 어떻게 사용하는 가를 결정한 규격이다. 이것의 종류는 10종류 이상으로 여러 어플리케이션에 블루투스의 각 레이어의 기능과 레이어와 어플리케이션과의 연결 방법에 대해 기술하고 있다. 대표적인 프로파일로는 다음과 같은 것들이 있다.

① HeadSet
② InterCom
③ Serial Port
④ Dial-up Network
⑤ FAX
⑥ LAN
⑦ File Transfer
⑧ Synchronization

시리얼 프로파일은 PC to PC의 연결 등에 해당하는 명세이다. 이러한 시리얼 프로파일은 응용프로그램에 대해 3단계의 과정을 명시해 놓았다. 이는 응용 프로그램이 연결을 위한 설정을 해주고 연결을 시도하는 단계와 이러한 연결을 받아들이는 쪽의 응용 프로그램이 이를 받아들이는 단계와 각각의 SDP 서버에 응용 프로그램에 대한 서비스를 등록하는 단계로 이루어진다. 응용프로그램은 리눅스 기반 PC에서 커널 2.4 환경에서 작성되었다. 그리고 프로그램은 KDE를 기반으로 한 Qt 툴킷을 사용하였고, 이에 대한 라이센스는 Troll Tech Company에 있다. 스택은 AXIS사의 LAN Access Point 테스트용으로 개발된 OpenBT를 기반으로 하였다. OpenBT는 오픈소스이므로 임의로 소스를 고쳐 테스트해볼 수 있다는 장점이 있다. 많은 버그를 수정해 놓은 버전이지만 응용프로그램의 목적이 틀리므로 수정할 필요가 있다. 타깃으로 하는 하드웨어는 CSR사의 BlueCore01b 칩이다. 그리고 이 칩은 이미 CSR사의 테스트 프로그램인 Bluechat에서 테스트되었다.

그림 10에서 보이듯이 응용 프로그램 계층은 스택의 사이에 인터페이스가 존재하는 형태로 구현된다. 이는 본 프로젝트에서는 포트 디바이스 드라이버가 될 것이다. 실제 응용프로그램에서 스택으로 어떤 명령이나 과정을 실행시키기 위해서는 ttyBTC를 통해서 ioctl (Input Output ConTroL) 명령을 보내야 한다. 그리고 데이터는 블르투스 스택에 의해 만들어진 가상의 가상 포트장치를 통하여 전달된다. 응용프로그램이 스택으로 보내주어야 하는 제어는 대표적으로 Inquiry, Connect, Disconnect, Send 등이다. 이런 제어는 ioctl을 통해서 전달된다. 실제 프로그램을 짤 때 생각해 주어야 하는 것은 HCI Transport 레이어로 사용되는 시리얼 포트와의 연계성이다. 시리얼 포트 위에 스택이 존재하기 때문에 시리얼 포트와 HCI 사이에 있는 버퍼가 그 용량을 넘기지 않아야 한다. 그렇기 때문에 시리얼 포트의 옵션을 주의깊게 설정해야하며 이는 프로그램의 초기에 행해져야 한다. 먼저 프로그램을 실행하면 스택과 시리얼 포트를 초기화해야 한다.

초기화과정을 거쳐서 응용 프로그램이 스택,

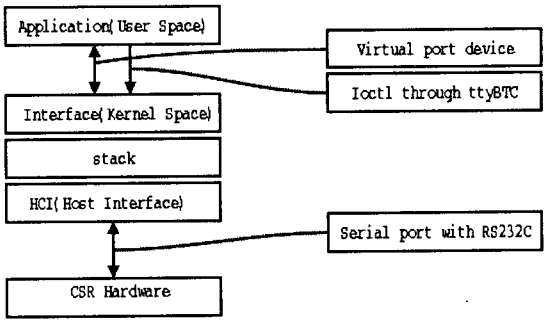


그림 10. 응용 계층과 Stack

하드웨어와 연동하여 동작할 수 있는 상태가 된다. 통신하기 위해선 밑의 계층부터 초기화시킨다. 가능하다면 하드웨어를 리셋하고 그 다음엔 시리얼 포트의 초기화 그리고 스택을 초기화한다. 그 다음에 인터페이스를 초기화하는데 이는 가상 라인중 하나일 것이다. 그림 11이 초기화 과정을 설명하고 있다.

초기화가 끝나면 Inquiry와 Connect 명령을 통하여 다른 장치와 접속을 시도하고 이것이 성공하면 데이터를 주고받을 수 있다. 데이터를 주고받기 위해선 스택이랑 통신할 필요가 있는데 이는 가상 포트장치에 쓰고 읽는 것으로 해결이 된다. 하지만, 비동기 통신이 되어야 하므로 타이머를 써서 일정한 시간에 쓰고 읽기를 한다. 이를 위해서 수신 버퍼와 송신 버퍼를 체크하여 이 버퍼가 쓰고 읽기를 하도록 구현하였다. 그림 12는 데이터를 주고받는 알고리즘을 설명한다.

여기서는 연결하는 입장에서 주로 설명하였으며 연결되는 입장에서는 시리얼 프로파일에 근거하여 자신이 연결을 받을 것이라는 것만 명시하면 된다. 같은 스택을 쓰고 옵션이 같으므로 장치 드

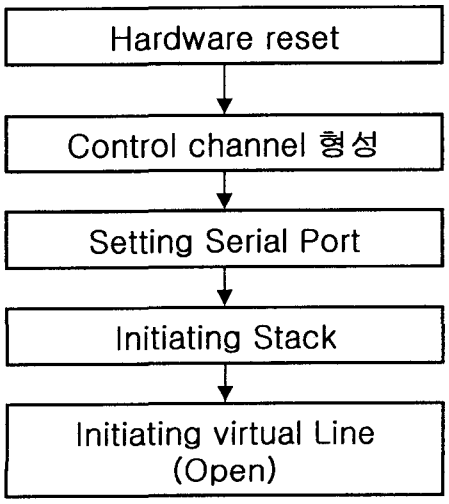


그림 11. 초기화 과정

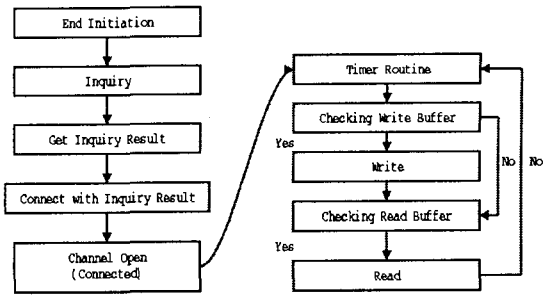


그림 12. 초기화 과정 이후

라이버에서 연결 요청이 왔을 때, 장치 드라이버 쪽에서 라인을 검사하고 가능하면 응답하면 된다.

3) USB 인터페이스

본 시스템에서는 USB 인터페이스를 위하여 SA-1111를 사용하였다. SA-1111은 USB 호스트 인터페이스 컨트롤러가 내장된 프로세서이며, USB Spec. 1.1에 대한 12Mhz의 Full Speed와 1.5 Mhz의 Low Speed 모두 사용이 가능하다.

SA1111에 USB 장치가 연결되면 SA1110에 대하여 인터럽트가 발생하고, SA1110에서는 발생된 인터럽트를 확인하여 USB 장치가 연결되었는지 확인한다.

그리고 SA1111의 공유 메모리 컨트롤로(Shared Memory Controller)를 사용하여 USB 장치의 데이터를 SDRAM에 쓰게 된다. 이러한 과정은 PC에 USB 장치를 연결했을 경우와 마찬가지로 Enumeration 과정을 거친다.

아래의 과정은 USB 테스트 시스템의 Enumeration 과정을 나타낸다.

① 디바이스가 처음 연결되면 Hub나 Root Hub가 호스트에게 어떤 이벤트가 발생되었음을 알린다. USB Hub에는 Pull down resistor가 있는데 디바이스가 붙으면 D+단의 Pull Up resistor의 Level을 높여 연결을 알린다.

② 호스트가 디바이스가 있는 포트로 "Port

enable & reset" 명령을 내보낸다.

③ 호스트는 10ms 동안 리셋 신호를 내보낸 후 100mA의 Power를 공급한다. (Powered state) 디바이스의 관점에서는 이 Signal을 처음 받으면서 호스트의 Default Address(Address 0, endpoint0)로 통신하게된다(Default State).

④ 호스트가 GET DESCRIPTOR setup packet을 내보내면서 이에 해당하는 data를 요구한다. 이 data가 만족되면 호스트가 SET ADDRESS 명령을 내보낸다.

⑤ SET ADDRESS 명령에 따라 호스트가 디바이스에 특별한 주소를 할당한다.

⑥ 이 주소를 통해 다시 GET DESCRIPTOR (device, configuration) 명령을 내보내고 SET CONFIGURATION 명령을 내보냄으로 디바이스를 쓸 준비를 마친다.

4) 사용자 인터페이스

USB 테스트 시스템의 사용자 인터페이스는 사용자가 USB 장치의 테스트 결과를 시각적으로 확인 할 수 있도록 시리얼 LCD를 사용하여 구성하였다. 시리얼 LCD에는 블루투스 연결상태, USB 장치의 연결상의 유무, USB 장치의 ProductID와 Vender ID를 표시하였다. 사용된 시리얼 LCD는 한글과 영문을 혼합하여 사용할 수 있다. 그리고 사용자가 원하는 경우에 언제든지 사용할 수 있도록 키 버튼 기능을 추가하였다.

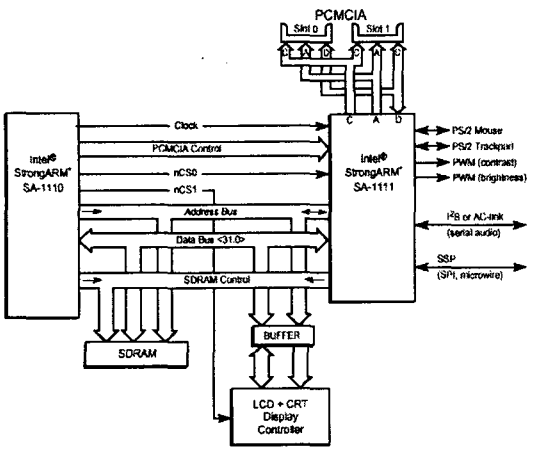


그림 13. 시스템 블록도

5) 운영 소프트웨어(Operating System)

임베디드 시스템은 일반적인 시스템과는 달리 특정한 작업만을 하도록 설계되는데 초기의 임베디드 시스템은 비교적 단순해서 운영체제가 필요없이 사람이 순차적인 프로그램을 작성해서 실행되도록 하였고, 중간에 인터럽트가 발생되는 경우에만 그 순차적인 프로그램에서 잠시 벗어나는 정도였다. 이전의 임베디드 시스템들은 주로 간단하고 단순한 순차적인 작업에 관련되었기 때문에 굳이 OS를 사용한다는 것은 낭비가 되었었고 그럴 필요조차 없었다. 하지만 최근의 임베디드 시스템 분야에서는 그 시스템 자체가 커지게 되고, 네트워크나 멀티미디어가 시스템에 기본으로 자리 잡으면서 임베디드 시스템이 해야 할 일들도 많아지고 복잡해졌기 때문에 순차적인 프로그램 작성이 매우 어렵게 되었다. 따라서 임베디드 시스템에서도 운영체제의 개념이 필요하게 되었다. 임베디드 시스템에 사용될 수 있는 OS는 상용 OS부터 공개 OS까지 그 수가 많지만 각각 장단점이 있다. USB 테스터도 역시 임베디드 시스템이며 여러 가지 사항을 고려하여 임베디드 리눅스를 OS로 사용하였다.

## 3. USB Tester 실험

### 1. 테스트 환경

USB Tester를 테스트하기 위하여 그림 14와 같은 환경을 구성했다. PC쪽 구성요소로는 PC와 PC에서 실행되는 USB test프로그램, PC와 시리얼 포트를 통해 연결되는 블루투스 모듈이 있다.

USB Tester쪽 구성요소로는 임베디드 리눅스를 탑재한 USB Tester, 블루투스 모듈, 정보출력을 위한 LCD가 있다. 그리고 마지막으로 테스트 대상이 되는 USB장치로는 Cypress USB mouse를 준비하였다.

PC는 중앙 시스템으로서 데이터 베이스 역할을 한다. USB Tester는 테스트한 장치에 대한 결과 데이터를 블루투스 무선 통신을 통해 PC로 전송하는데 이 PC는 USB장치를 테스트한 결과 데이터를 받아 처리하는 역할을 한다. PC에서 실행되는 프로그램은 전송받은 데이터를 화면에 표시해주고 불량인지 정상인지 나타내며 불량 비율을 백분율로 나타낸다.

USB Tester는 테스트할 USB장치에 대한 프로파일을 가지고 있으며 테스트되는 USB장치에 대한 프로파일과 비교되어 장치의 불량유무를 판단한 후 그 결과를 블루투스 무선 통신을 통해 PC로 전송하고 테스트하는 사람에게는 LCD를 통해서 결과를 보여준다.

다음 절에서는 이 테스트 환경에서 Cypress USB mouse를 테스트하는 절차와 테스트 결과에 대해서 살펴본다. 불량제품으로 사용한 USB장치로는 프로파일을 전혀 읽을 수 없는 고장난 USB 마우스와 다른 프로파일을 가지는 타회사의 USB 마우스를 사용하였다.

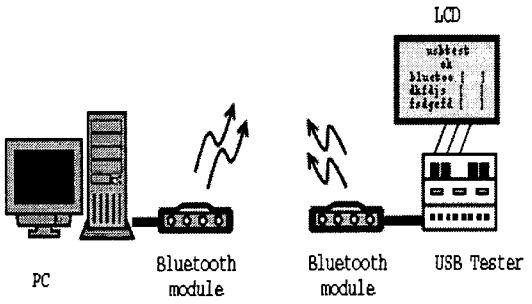


그림 14. USB Tester실험 환경 구성

## 2. 테스트 과정 및 결과

### 가. PC에서 USB test프로그램 실행

PC에서 실행되는 이 프로그램은 PC에 시리얼 포트를 통해 연결된 블루투스 모듈을 초기화하고 상대 블루투스 모듈과 연결을 시도하며 USB Tester로부터 장치 프로파일을 전송받아 표시하고 불량유무에 대해서 나타낸다. 다음 그림 15는 PC에서 실행되는 USB test프로그램을 캡쳐한 것이다.

### 나. USB Tester의 power on(또는 리셋)

USB Tester는 PC와 마찬가지로 블루투스 모듈과 시리얼 포트를 통해 연결되어 있다. 또한 LCD 역시 시리얼 라인을 통해 연결되어 있다. USB Tester의 전원을 켜거나 또는 리셋을 시키면 먼저 부트로더가 실행이 된다. 부트로더는 플리시 메모리에 있는 커널이미지와 램디스크 이미지를 램으로 다운로드하고 램에 복사된 커널시작 번지로 점프해서 커널 부팅을 시작한다. 커널이 부팅된 후에는 디바이스 드라이버를 커널에 삽입하고 USB 테스트 프로그램과 블루투스 응용프로그램을 실행한다.

### 다. PC에서 USB Tester와 블루투스 통신 연결

USB Tester가 power on후에 초기화되어 준비

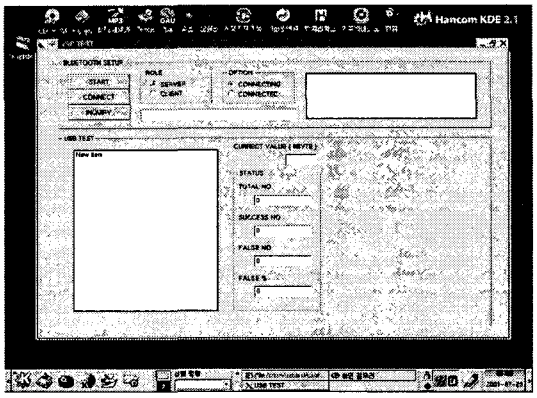


그림 15. PC에서 실행되는 USB test프로그램

가 되면 PC에서 실행중인 USB test프로그램에서 USB Tester와 블루투스 통신 연결을 시도한다. 정상적으로 연결이 되면 USB Tester의 LCD에는 그림 16에 보이는 'BT연결상태'에 '끊김'이라고 되어 있는 부분이 '연결'로 바뀐다. 그리고 USB Tester는 블루투스 무선통신을 통해서 USB 장치의 프로파일을 전송할 수 있는 상태가 된다.

여기까지 하면 USB장치를 테스트할 모든 준비가 완료된 것이다. 이제 다음으로 USB장치를 삽입하고 테스트하게 된다.

라. USB장치의 삽입과 버튼 누름

USB장치를 삽입한다. 특히 이 테스트에서는 Cypress USB mouse를 대상으로 하므로 이 마우스를 꽂는다. 그리고 버튼을 누른다. 버튼을 누르게 되면 인터럽트가 발생하게 되는데 이 인터럽트에 대한 처리는 스위치 디바이스 드라이버에서 하게 된다. 인터럽트 핸들러는 프로세스 ID를 가져와서 그 pid로 SIGUSR1 시그널을 발생시킨다. pidbox.o가 가지는 pid는 USB테스트 프로그램인 usbtest의 pid이므로 SIGUSR1 시그널에 대한 처리는 usbtest가 맡아서 하게 된다.

SIGUSR1 시그널 핸들러는 커널 모듈과 연결된 장치 파일을 열어서 USB장치 프로파일을 읽어온다. 읽어온 프로파일과 정상적인 프로파일을 비교하여 불량인지 테스트하게 된다. 테스트 결과에 따라 그림 16과 같이 LCD에 '정상', '비정상'을 표시하게 되며 그 프로파일 역시 나타낸다. 또한 블루투스가 연결되어 있는 상태인 경우 블루투스를 통해 그 프로파일을 PC로 전송하게 된다.

USB mouse가 불량인 경우에는 장치 프로파일을 읽을 수 없거나 읽더라도 제대로 된 프로파일이 아닐 것이다. 테스트를 위해서 완전히 고장난 USB 장치를 삽입했을 경우와 Cypress 제품이 아닌 장치를 삽입했을 경우로 나누어 테스트해 보았다. 완전히 고장난 USB 장치의 경우에는 프로파일 자체를 읽을 수 없어서 LCD에는 프로파일이 모두 0으로 나타났으며 '비정상'으로 표시되었다. 또한 다른 제품인 경우에는 프로파일을 읽었지만 Cypress mouse의 프로파일과 다르다. 그래서 LCD에는 읽혀진 프로파일이 나타났지만 '비정상'으로 표시되었다.

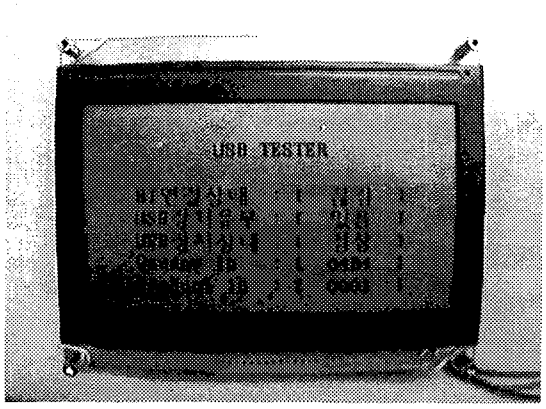


그림 16. USB 장치가 정상일 때

## 4. 결 론

본 연구에서는 USB 디바이스 생산라인의 최종 테스트 과정에 사용되는 저가 및 휴대형 USB 테스트 시스템을 개발하였다. 개발된 USB 테스트 시스템은 기존의 USB 제품 생산 공장에서는 PC를 사용함으로서 발생되는 공간적, 비용적, 시간적 문제점을 해결하였고, 특히 휴대가 가능하고, 사용이 간편하며, 저가형으로 개발을 추진하였다.

개발된 테스트 시스템은 저전력 고성능의 Strong ARM SA1110 프로세서를 사용하였고, 임베디드 리눅스 운영시스템을 채용하였으며, CSR 블루투스 칩을 사용하여 테스트 시스템과 PC간의 무선 통신이 가능하도록 하였다. 특히 USB 호스트 컨트롤러를 구현하기 위하여 SA1111을 사용하였고, SA1111 용 디바이스 드라이버를 개발하였다.

실험실 환경에서의 개발된 USB 테스트 시스템의 실험 결과 USB 마우스, 키보드 등의 생산량이 많고 본 개발된 테스트 시스템을 대량으로 필요로 하는 USB 디바이스 경우 우수한 테스트 성능을 보여주었다.

## 참 고 문 헌


[1] Kurt Wall, Mark Watson, and Mark Whitis, Linux Programming, SAMS, 2000.

[2] 이귀영, Add-on Linux Kernel Programming, 글로벌, 2001.

[3] Richard Stones and Neil Matthew, Beginning Linux Programming, 정보문화사, 2000.

[4] Steve Furber, ARM System-on-chip Architecture, Adison-Wesley, 2000.

[5] Universal Serial Bus Specification Revision 2.0, Intel Corp., 2000.

[6] Wooi Ming Tan, Developing USB PC Peripherals-Using the Intel 8x930Ax USB Microcontroller, Annabook, 1997.

[7] Jean J. Labrosse, Embedded System Programming, RD Books, 2000.

[8] Walter Oney, Programming the Microsoft Windows Driver Model, 정보문화사, 2001.

[9] Jan Axelson, USB Complete, Lakeview Research, 1999.

[10] John Hyde, USB Design by Example, Intel University Press, 1999.

[11] *Specification of the Bluetooth System Core*; available online at http://www.bluetooth.com/developer/specification/Bluetooth_11_Specifications_Book.pdf.

[12] *Specification of the Bluetooth System Profiles*; available online at http:// www.bluetooth.com/developer/specification/Bluetooth_11_Profiles_Book.pdf.

[13] G. Miklos et al., *Performance Aspects of Bluetooth Scatternet Formation*, poster presentation at Mobile Ad Hoc Networks and Computing (MobiHOC 2000), IEEE/ACM workshop, Aug. 2000.

[14] T. Salonidis et al., *Distributed Topology Construction of Bluetooth Personal Area Networks, Proc. IEEE Infocom 2001*, IEEE Communication Society, New York, 2001.

[15] J.C. Haartsen, et al., *Bluetooth - A New Low-Power Radio Interface Providing Short-Range Connectivity*, IEEE Proceedings of The IEEE Vol. 88 No. 10, OCTOBER 2000

[16] P. Bhagwat, *Industrial Report - Bluetooth : Technology for Short- Range Wireless Apps*, IEEE Internet Computing MAY, JUNE 2001

[17] Jim Connolly, "Frequency management and regulatory challenges in the 2.4GHz ISM band," A Int. Conf. on Bluetooth '99, London, June 1999.

[18] Mahesh. Bhave, "Bluetooth in the mobile telephony environment," A Int. Conf. on Bluetooth '99, London, June 1999.

권 오 상

- 1990년 2월 인하대학교 전자공학과(BS)
- 1992년 2월 인하대학교 전자공학과(MS)
- 1999년 2월 인하대학교 전자공학과(Ph. D)
- 1992년 1월~1996년 2월 대우중공업 중앙연구소 주임연구원
- 1999년 3월~2000년 1월 건양대학교 강의교수
- 2000년 6월~현재 한울로보틱스 지능로봇연구소 소장
- 관심분야 : 가정용로봇, 재활보조로봇, 홈네트워킹

박 춘 명

- 1983년 2월 인하대학교 공과대학 전자공학과(BS)
- 1986년 2월 인하대학교 대학원 전자공학과(정보공학전공)(MS)
- 1994년 2월 인하대학교 대학원 전자공학과(정보공학전공)(Ph. D)
- 1995년 9월~현재 충주대학교 전기.전자 및 정보공학부 컴퓨터공학전공/교수
- 1984년~현재 IEEE Computer Society Member
- 1998년~2002년 6월 현재 한국멀티미디어학회 종신회원/이사/ 학회지편집부위원장
- 2000년~2002년 6월 현재 디지털컨테츠학회 정회원/이사
- 1984년~2002년 6월 현재 대한전자공학회 정회원/충북지부 총무이사/멀티미디어분과 전문위원/대한전자공학회 정보화위원
- 1999년~2002년 6월현재 충북중소기업정보화위원
- 2002년 6월~현재 충주대학교 글로벌미디어연구소 개설책임자
- 저서 : 정보공학입문, 컴퓨터구조, 멀티미디어 등 다수
- 관심분야 : 임베이드시스템, 멀티미디어시스템, 차세대 디지털논리시스템 및 컴퓨터구조, 마이크로프로세서 응용 등