**설계 프로젝트**

**제안서**

날짜 : 2012.03.15(목)

지도교수 : 이민석 교수님

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lemonade | 학번 | 이름 | E-Mail | 전화번호 |
| 0592073 | 이상현 ○ | avenger86@naver.com | 010-8839-0477 |
| 0592051 | 김종욱 | vhfmxpznq01@naver.com | 010-3115-9196 |
| 0792047 | 강인구 | 2002gig@naver.com | 010-9686-7175 |
| 0792078 | 이윤재 | sparada@naver.com | 010-2657-8421 |

**문서 정보 수정 내역**

|  |  |
| --- | --- |
| 파일명 | USB/IP 제안서 |
| 원본 작성자 | 이상현, 이윤재 |
| 수정자 | 이상현, 김종욱, 강인구, 이윤재 |

**버전 정보**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 날짜 | 수정자 | 버전 | 수정/추가 | 내용 |
| 2012.01.06(금) | 이상현 | 1.0 | 추가 | 초안, 양식 작성 |
| 2012.02.24(금) | 강인구 | 1.1 | 추가 |  |
| 2012.02.28(화) | 이윤재 | 1.2 | 추가 |  |
| 2012.03.09(금) | 이상현 | 1.3 | 수정 | 배경, 목적, 활용 수정 |
| 2012.03.15(목) | 강인구 | 1.4 | 추가 | 관련기술소개 추가 |
|  |  |  |  |  |
|  |  |  |  |  |

**목차**

1. **프로젝트 수행 목적**
   1. **프로젝트의 배경**
   2. **프로젝트의 목적**
   3. **프로젝트의 활용**
      1. **PC에서 스마트폰 애플리케이션 이용**
      2. **PC에서 스마트폰 센서, 장치를 주변기기로 이용**
      3. **스마트폰의 단일 사용자성 이용**
2. **프로젝트 결과물의 개요**
   1. **프로젝트 결과물의 설명**
   2. **관련기술 소개**
      1. **USB**
      2. **USB/IP**
      3. **LibUSB**
   3. **프로젝트 결과물의 구조**
   4. **현실적 제약조건**
   5. **개발 도구**
   6. **기존 소스나, 다른 과목 결과물 등 자료이용**
3. **결과로서 제출할 실적 물 목록**
   1. **소스**
   2. **매뉴얼**
   3. **실행 파일**
4. **프로젝트 수행 추진 체계 및 일정**
   1. **조직도**
   2. **마일스톤 및 일정**
   3. **주 단위의 프로젝트 수행**
5. **참고자료**
6. **프로젝트 수행 목적**
   1. **프로젝트의 배경**



인터넷 트래픽 조사 기관인 CAIDA에 의하면 2000년에는 전체 인터넷 트래픽의 절반을 웹이 차지했지만, 2010년에는 23%로 줄었고, 지금도 계속 하락하고 있다고 한다. 스마트폰과 태블릿PC 등 모바일 단말기 사용자의 증가가 주요 원인 중 하나다. 최근 2011년 10월에 2천만명을 넘어설 정도로 국내 스마트폰 가입자수는 급격히 증가하고 있다. 스마트폰은 이제 우리 실생활에서도 빼놓을 수 없는 필수품이 되었다.

스마트폰 시장이 커지면서 제조사들이 앞다퉈 대용량 메모리, 고화질 카메라, 빠른 처리 속도, 다양한 센서 등 많은 기능과 뛰어난 성능을 갖춘 신제품을 출시하고 있고, 스마트폰에서만 사용할 수 있는 서비스들이 많아졌다. 사용자는 언제 어디서든 접속 가능한 인터넷, 카카오톡 같은 메신저, GPS이용한 위치정보 등 수 많은 서비스에 손쉽게 접근할 수 있다. 하지만 스마트폰 단독으로 사용 가능한 기능은 많은데 비해, PC와 연결하여 사용할 수 있는 기능은 단순한 메모리 이외에는 많이 활용되지 않는다.

PC는 스마트폰에 비해서 처리 속도와 인터넷 속도가 빠르고, 키보드처럼 쉬운 입력 인터페이스, 더 큰 화면을 가지고 있는 장점이 있다. 스마트폰과 PC를 연결해 서로의 강점만 사용할 수 있다면 기존 서비스는 더 편리해질 수 있고, 기존에 없던 새로운 가치를 창출해낼 수 있을 것이다.

* 1. **프로젝트의 목적**

USB(Universal Serial Bus / 범용 직렬 버스)는 마우스, 키보드 같은 주변기기를 컴퓨터와 연결하는 데 쓰이는 입출력 표준 가운데 하나이다. USB는 각각의 장치마다 다양했던 직렬, 병렬 연결방식을 하나의 방식으로 통합 및 대체하기 위해 만들어 졌다. 현재는 USB는 윈도우나 리눅스 같은 현재 대부분의 주요 운영체제가 탑재하고 있는 인터페이스다. 표준화가 잘 돼있어서 인터페이스를 제어하는 복잡한 기능은 USB가 직접 연결되는 호스트컴퓨터가 지원한다. 그래서 USB장치 쪽은 다른 인터페이스에 비해 개발비용이 저렴한 편이고 신뢰성, 빠른 속도, 저 전력소비의 장점도 있기 때문에, USB 메모리 같은 우리가 자주 사용하는 장치부터 값비싼 산업장비, 의료장비들까지 다양한 장치들이 USB로 만들어져 손쉽게 PC와 연결되어 사용되고 있다.

우리는 안드로이드 스마트폰의 카메라, 마이크 등 특정 하드웨어나 애플리케이션 같이 USB장치가 아닌 것을, PC에서 USB장치처럼 인식하도록 ‘USB장치화’하고, 스마트폰의 서비스들을 PC에서도 쉽게 사용 할 수 있도록 하고자 한다. 이것을 가능하도록 하는 것이 우리가 스마트폰에 적용할 USB/IP라는 기술이다. USB/IP는 두 PC간 USB장치를 공유하는 시스템이다. 서버 쪽 PC에서는 직접 연결된 USB장치를 IP네트워크를 통해 공유하고, 클라이언트 쪽 PC에서 마치 자신의 PC에 직접 연결된 것처럼 사용한다.

USB/IP는 오픈 소스로, 리눅스는 서버와 클라이언트, 윈도우는 클라이언트로 사용 가능하다. 이 중 리눅스 서버용 소스를 안드로이드 스마트폰용으로 가능하도록 수정한다. 클라이언트는 윈도우 PC로 한다. 서버 부분인 안드로이드 스마트폰에서는 기존에 사용중인 USB장치를 포함해 USB장치가 아닌 하드웨어나 애플리케이션까지 USB장치화하여 공유한다. 공유된 후, 클라이언트 PC에서는 USB장치처럼 연결해 사용할 수 있다.

* 1. **프로젝트의 활용**
     1. **PC에서 스마트폰 애플리케이션 이용**

PC는 정해진 신호에 대해 기계적으로 반응한다. PC에서 우리가 원하는 동작을 할 수 있도록 하려면 정해진 신호에 맞춰서 만들어주면 된다. USB/IP는 실제 USB표준에 맞춰서 송수신 하기 때문에 PC는 연결된 장치가 USB장치인지 아닌지에 관계 없이 USB장치가 연결된 것처럼 이용할 수 있다.

그래서 실제 USB장치가 아닌 안드로이드 애플리케이션 같은 소프트웨어도 USB신호처럼 송수신 하기만 하면 PC에서는 USB장치로 이용이 가능하다.

* + 1. **PC에서 스마트폰 센서, 장치를 주변기기로 이용**

근접센서, 조도센서, 가속센서와 같은 스마트폰의 각종 센서들은 PC에서 새로운 USB장치로서 사용될 수 있다. 카메라, 스피커, 터치스크린 같이 스마트폰에 있는 많은 하드웨어 장치들도 PC에서 웹 캠이나 스피커, 마우스, 키보드 같은 무선 주변기기로 사용 될 수 있다.

예를 들어 기존에는 PC작업중에 카메라를 이용하고자 한다면 스마트폰 카메라로 사진 촬영 후 PC로 이동해야 했었다면, 스마트폰 카메라가 USB장치로 연결 되어 있을 경우 PC에서 이용하는 주변기기로 동작하기 때문에 촬영하는 것이 바로 PC서비스에 적용된다. 그러므로 USB장치로 사용하게 될 경우 작업절차를 줄일 수 있다.

* + 1. **스마트폰의 단일 사용자성 이용**

PC는 주로 공동으로 사용하기 때문에, 결제나 로그인 등의 개인작업 시 본인인증 절차가 필요하다. 반면에 스마트폰은 사용자가 한 명이라 PC에 비해 본인인증 절차가 단순한 편이다. 본인인증을 하기 위해 본인명의 휴대폰으로 인증번호를 보내고 PC에서 이 번호를 입력해 인증하는 방법이 있을 정도다.

스마트폰이 개인정보를 제공하는 USB장치로 동작한다면 PC에서 간단히 본인인증이 가능할 것이다. PC는 본인인증이 필요할 때, 연결된 장치로부터 개인정보를 확인하고 본인인증을 할 수 있게 된다.

1. **프로젝트 결과물의 개요**
   1. **프로젝트 결과물의 설명**

PC에 USB 포트를 이용하여 다양한 주변기기를 연결하여 사용하듯 스마트폰을 인터넷을 통하여 PC에 연결하여 USB장치를 연결하듯 연결하여 PC에서 스마트폰의 다양한 기능을 사용할 수 있도록 만든 것

* 1. **관련기술 소개**
     1. **USB**

USB(Universal Serial Bus / 범용 직렬 버스)는 컴퓨터와 주변 기기를 연결하는 데 쓰이는 입출력 표준 가운데 하나이다. 대표적인 버전으로는 USB 1.0, 1.1, 2.0, 3.0 등이 있다. USB는 다양한 기존의 직렬, 병렬 방식의 연결을 대체하기 위하여 만들어졌다.



기존의 다양한 포트

키보드, 마우스, 게임패드, 조이스틱, 스캐너, 디지털 카메라, 프린터, PDA, 저장장치와 같은 다양한 기기를 연결하는 데 사용되고 있다. 이러한 기기 연결의 대부분은 표준 연결 방식을 이용하여 이루어지고 있다. USB는 PC를 위하여 개발되었지만 지금은 PDA나 게임콘솔 등에서도 채택되어 사용되고 있고, USB의 전원 공급 기능을 이용하여 충전 용도로도 많이 사용되고 있다. 전 세계적으로 약 20억개 이상의 USB 장치가 있다.

**USB의 속도**

USB 표준의 이론상 최고 전송 속도는 다음과 같다.

* Low speed(1.5 Mbps)
* Full speed(12 Mbps)
* High speed(480 Mbps)
* Super speed(5 Gbps)

USB 초기에 Low speed로 연결되는 키보드, 마우스 같은 제품들이 있었지만 이제는 거의 쓰이지 않는다. Full speed의 USB 1.0도 1.1 규격으로 업데이트된 다음 USB 2.0으로 전환되었다. 2010년 초 USB 3.0을 적용한 제품들이 나오기 전에는 대부분 USB 2.0을 사용했다. USB 2.0 제품의 이론상 최대 전송속도는 480Mbps이지만 실제 속도는 절반 정도인 240Mbps가 나온다.

USB 3.0 방식을 사용하는 외장 저장장치 시제품의 속도를 측정해보면 연속 읽기 속도가 960Mbps에 달하고 쓰기 속도도 824Mbps가 나온다. 이는 현재 사용되는 하드디스크의 최대 전송 속도를 초과한다.

그러나 USB의 속도에는 치명적인 제약이 있는데 주 컨트롤러(host controller)에 연결된 기기들 간에는 대역폭을 나누어 쓰게 되므로 장치가 늘어날수록 속도는 현저히 떨어진다.

**USB의 전송방식**

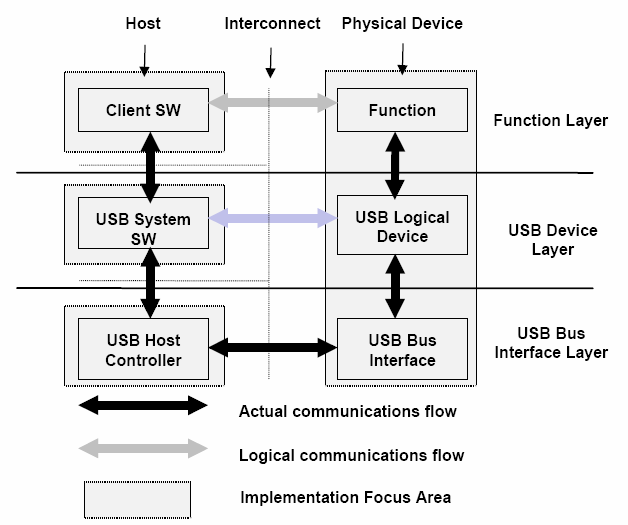
- 제어(Control)전송 : 비 동기통신. 장치 기본 제어에 사용. Device설정, Address할당 등에 사용.

- 벌크(Bulk)전송 : 비 동기통신. 인터럽트전송이나 등시성전송이 쉬고 있을 때 일반적인 데이터통신에 사용. 정확성이 보장되는 프린터나 디스크 등에 사용.

- 인터럽트(Interrupt)전송 : 비 동기통신. 소량의 데이터 통신을 하는 이벤트 통지 같은 처리에 적합. 마우스, 키보드 등에 사용.

- 등시성(Isochronous)전송 : 동기통신. Real time성이 요구되는 전송방식. 미리 정해진 대역을 확보하여 설계자가 의도한 시간에 전송이 가능. USB 스피커나 헤드폰에 사용.

**USB의 데이터 전송구조**



USB 구조

- USB의 물리적 장치 : USB 케이블 종단에 물려있는 사용자에 있어서 특별한 하드웨어 구조에 따른 Function을 나타낸다.

- 클라이언트 SW : USB 장치에 대응하는 호스트에 대한 소프트웨어를 실행하며 실제의 장치에 의해서 결정되는 소프트웨어이며 USB와 함께 제공되는 특정 운영체제와 타겟장치에 따라서 제공된다.

- USB 시스템 SW : 특정 운영체제 에서 USB를 지원하는 소프트웨어이며 운영체제에는 의존적이지만 특정 클라이언트 소프트웨어 또는 장치에 대해서는 독립적이다.

- USB호스트 컨트롤러 : 호스트 측에서 버스 인터페이스를 나타내며, USB 장치를 호스트에 결합할 수 있도록 해주는 모든 소프트웨어 또는 하드웨어를 의미 한다.

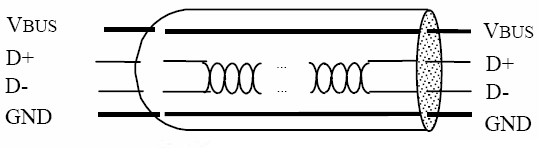
USB 호스트란 USB 호스트 컨트롤러와 USB 시스템 소프트웨어 그리고 마지막으로 클라이언트를 가진다. USB시스템 소프트웨어에서는 USB드라이버(USBD), 호스트 컨트롤러 드라이버(HCD) 호스트 소프트웨어로 구성되어 있다. 호스트는 결합된 장치에 대하여 모든 책임과 권한을 가지고 있으며 USB 장치는 버스로 접근할 수 있는 사용권을 호스트에 의해 부여 받는다. 또한 호스트는 USB의 토폴로지를 항상 모니터하고 있어야 하고 모든 접근을 통제해야 한다.

USB 장치란 USB 인터페이스, USB 논리적 장치(Endpoint), Function으로 구성된다. USB 장치는 다양한 성능을 가지고 있으나 호스트와의 송수신을 위해 동일한 기본 인터페이스를 가지고 있다. 즉, 호스트가 각종 장치를 동일한 방법으로 관리할 수 있다. 확인과 설정을 위해 장치는 자신의 정보를 호스트로 보고하며 포맷은 모든 장치에 대해 공통적인 부분과 각 장치에 특별한 부분으로 구분될 수 있다.

장치는 마치 호스트와 직접적으로 연결된 논리적 장치로 데이터를 송/수신한다. 클라이언트 S/W와 Function과의 관계를 살펴보면 클라이언트 소프트웨어는 USB에 같이 연결된 다른 장치와는 독립적으로 관계된 Function을 직접적으로 다뤄야 한다. Function 들을 처리하기 위해서는 USB 소프트웨어 프로그래밍 인터페이스들을 사용해야 한다. 클라이언트 소프트웨어는 USB에 연결되어 있는 다른 장치들과는 독립적이 되며, 이는 설계자들이 장치의 설계와 클라이언트 소프트웨어를 하드웨어와 소프트웨어간의 상호 동작에 대해서만 생각 할 수 있도록 해주는 것이다.

**USB 신호처리**

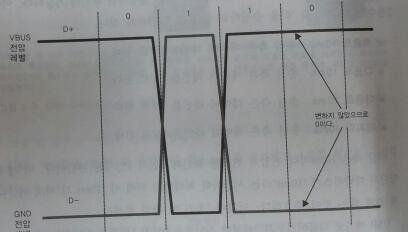
USB 케이블은 아래 그림과 같이 차동형인 데이터 선(D+,D-)과 전원(Vbus), 그라운드(GND)의 4개의 선으로 구성된다.



데이터 신호는 차동(differential / 전압, 전류, 그 밖의 양의 두 차이에 의해서 동작하도록 한 것) 신호이므로 D+와 D- 신호 선은 트위스트 페어 선으로 하고 외부의 잡음을 줄이기 위하여 케이블에 실드 하였다. 로우 스피드용 케이블은 트위스트 페어나 실드가 필요하지 않다(케이블 길이 3m 제한됨). 전원 핀은 +5v에서 최대 500mA의 전류를 흘린다. 그리고 각 선은 색을 다르게 지정되어있다. USB커넥터의 치수도를 나타낼 수 있는데 시리즈 A 플러그는 상류 측에서 접속하고 시리즈B 플러그는 하류 측에 접속한다.

다음으로 케이블의 핀 배치를 보면 전원용인 1핀과 4핀의 길이는 신호용에 비해 길고 2핀과 3핀은 짧다. 이는 삽입 시 전원이 먼저 공급된 다음 신호 선이 접속되도록 하고 반대로 뺄 경우 신호 선이 먼저 절단된 다음 전원이 절단되게 하여 디바이스가 파괴되지 않도록 하였다.

2개의 데이터라인으로 1비트를 표현한다. 이 2개의 데이터라인(D+와 D-)의 전압변화로 0인지 1인지를 판단한다. 연속으로 0을 보낼 때는 전압을 변화하지 않는다. 하지만 동기 시키기 위해 6비트 연속 0이면 무조건 1로 만들고, 이 1은 동기 시키는 데만 사용하고 데이터로 다루지 않는다(비트 스태핑). 1비트별 시간간격은 로우/풀/하이 스피드 별로 다르다.



**USB 프로토콜 (통신규약)**

USB는 Frame 단위로 전송되는데 하나의 Frame은 여러 개의 Transaction으로 구성이 되고 이 Transaction은 USB 상에서 돌아다니는 여러 개의 Packet으로 구성이 됨.

(Frame > Transaction > Packet)

* **Frame :** 1ms마다 발생하는 전송단위 (SOF와 Transaction 들로 이루어짐)

**<SOF>** (Start Of Frame) Packet – 24bit

|  |  |  |
| --- | --- | --- |
| PID  (8 bit) | Frame Number  (11 bit) | CRC5  (5bit) |

PID : Packet ID Frame Number : Frame 번호 CRC5 : Error 유무 확인

* **Transaction :** Host와 Device간의 전송단위 (3개의 Packet으로 이루어짐)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Token | | | | Data | | Handshake | | |
| IN | OUT | SOF | SETUP | Data0 | Data1 | ACK | NAK | STALL |

* **Packet :** 전송의 최소단위

<Token Packet>

|  |  |  |  |
| --- | --- | --- | --- |
| PID  (8 bit) | ADDR  (7 bit) | ENDP  (4 bit) | CRC5  (5 bit) |

<Data Packet>

|  |  |  |
| --- | --- | --- |
| PID  (8 bit) | Data  (1~1023 byte) | CRC16  (16 bit) |

<Handshake Packet>

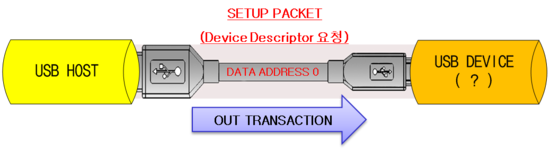
|  |
| --- |
| PID  (8 bit) |

**USB Enumeration**

어플리케이션이 디바이스와 통신하기 위해 디바이스에 관한 정보를 얻고 적합한 디바이스 드라이버를 결정하는 정보 교환 과정.

디바이스 주소할당, 디바이스로부터 descriptor 읽기, 적합한 디바이스 드라이버 선정과 로딩, 디바이스 전원 요구사항과 인터페이스 지정하는 컨피규레이션 선택 등이 있다.

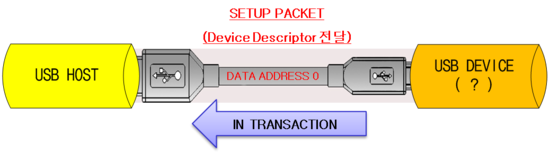
* + - 1. 사용자가 디바이스를 USB 포트에 연결하고, 허브는 디바이스를 감지한다.
      2. 호스트 PC 가 Data Address0를 통해 Device Descriptor 요청 (SETUP Packet)



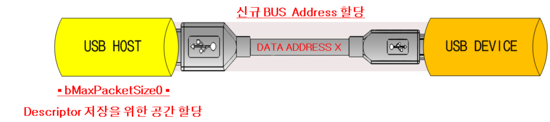
* + - 1. 호스트 명령을 디코드하고 프로그램 메모리에서 Descriptor 검색.



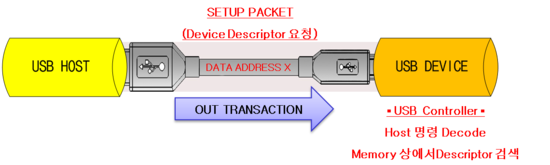
* + - 1. Device Descriptor 전송.

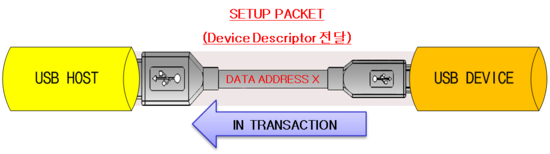


* + - 1. HOST가 신규 Bus Address 할당.

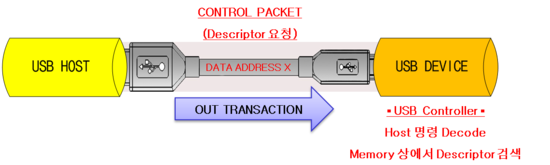


* + - 1. 새 Data Address를 통해 Device Descriptor 요청(SETUP Packet) 및 수신





* + - 1. Device Descriptor 제외한 나머지 Descriptor (컨피규레이션 Descriptor와 리포트 Descriptor) 요청 및 수신



* + 1. **USB/IP**

USB/IP는 2005년 FREENIX 학회에서 논문을 통하여 처음 소개 되었고 리눅스에서 구현 되었다. USB/IP는 모든 유형의 USB 장치를 공유하는 것이 목적이다. 이를 위해 실제 장치가 없어도 장치가 있는 것처럼 가상화 해주는 VHCL(Virtual Host Controller Interface) 드라이버를 구현한다. 또한 서버에 연결된 장치에 발생한 USB 입출력 데이터를 클라이언트로 전달 할 수 있는 스텁 드라이버(stub driver)를 구현하였다. USB 입출력은 URB(USB Request Bloc)라고 하는 공통된 구조체로 표현이 되며 이것은 USB 프로토콜 스펙에 정의 되어 있는 것이다. URB는 네트워크를 통하여 클라이언트와 서버 사이를 IP 패킷에 담겨 송수신되게 된다. 리눅스는 커널에서도 손쉽게 패킷을 전송할 수 있는 라이브러리를 제공 한다. USB/IP에서도 이것을 사용하여 VHCL 드라이버와 스텁 드라이버 간에 IP 네트워크 통신을 구현하였다. SUB/IP에서도 이것을 사용하여 VHCI 드라이버와 스텁 드라이버 간에 IP 네트워크 통신을 구현하였다.

클라이언트

서버

응용 프로그램

USB Per-Device Drivers

USB Core Driver

Virtual Host Controller Interface Driver

USER

KERNEL

Stub Driver

USB

Host Controller Driver

USB Core Driver

USB Device

**URB 요청**

**URB 응답**

VHCI driver는 Client host에서 USB HCD역할을 하고, Stub driver는 Server host에서 USB PDD의 역할을 한다. VHCI driver는 원격 USB장치가 IP네트워크를통해 Client host에 연결된 경우 USB Core Driver의 포트 상태 변경을 통지하고, USB Root Hub의 동작을 에뮬레이트한다. USB/IP는 USB device 번호가 client device 번호와 server device 번호 사이에서 번역되는 것을 보장한다. 추가적으로, SETADDRESS와 CLEAR\_HALT와 USB요청은 USB Core driver가 관리 데이터를 올바르게 업데이트 할 수 있도록 인터셉트하고 있다.

IP 네트워크를 통한 모든 URB들의 전송은 TCP 프로토콜을 통해서 간다. 그러나 버퍼링 지연을 방지하고, 가능한 빨리 TCP/IP 패킷을 전송하기 위해 Nagle algorithm은 비활성화 돼있다. 현재 USB/IP는 통신하기 위해 UDP를 사용하지 않게 구현돼있다. 이것은 USB와 UDP/IP의 전송 오류의 특성이 매우 다르기 때문이다. host controller는 isochronous 트랜잭션 실패 시 재전송 하지 않지만, USB PDD들과 devices는 거의 트랜잭션 실패가 발생하지 않는다. 또한 devices에 어떤 물리적 문제가 발생하지 않는 한 트랜잭션 실패는 거의 USB에서 발생하지 않는다. 그러므로 일반적으로 transport layer는 URB들이 올바르게 도착하거나 손실된 패킷 재전송을 보장해야 한다.

device 공유 시스템의 핵심디자인은 운영체제에서 멀티 가상 버스 지원이다. 가상버스는 각각의 복잡한 주변 장치 버스로 구현된다.

USB/IP를 사용하며 아래와 같은 장점을 얻을 수 있다.

-모든 기능 사용(Full Functionality)

운영체제는 실제 장치가 연결되어 있는 것으로 알기 때문에 기능을 제한 없이 공유하여 사용할 수 있다.

-네트워크 투명성 제공(Network Transparency)

장치를 접근하여 사용하고 있는 어플리케이션은 장치가 실제 연결된 것인지 가상 연결된 것인지 구분하지 못한다.

-범용성 제공(Interoperability)

VHCL 드라이버가 다른 운영체제에서도 구현 가능하다면 USB 프로토콜은 스펙이 표준으로 정해져 있으므로 상호 다른 운영체제 간에도 USB 장치를 공유하여 사용할 수 있다.

* + 1. **LibUSB**

커널 레벨이 아닌 유저 영역에서 USB드라이버를 개발 시 사용되는 오픈 소스 라이브러리이다.

* 1. **프로젝트 결과물의 구조**
* **USB 구조**

Client SW

USB System

SW

USB Host

Controller

Function

USB Logical

Device

USB Bus

Interface

Host

Physical Device

Interconnect

User

Kernel

H/W

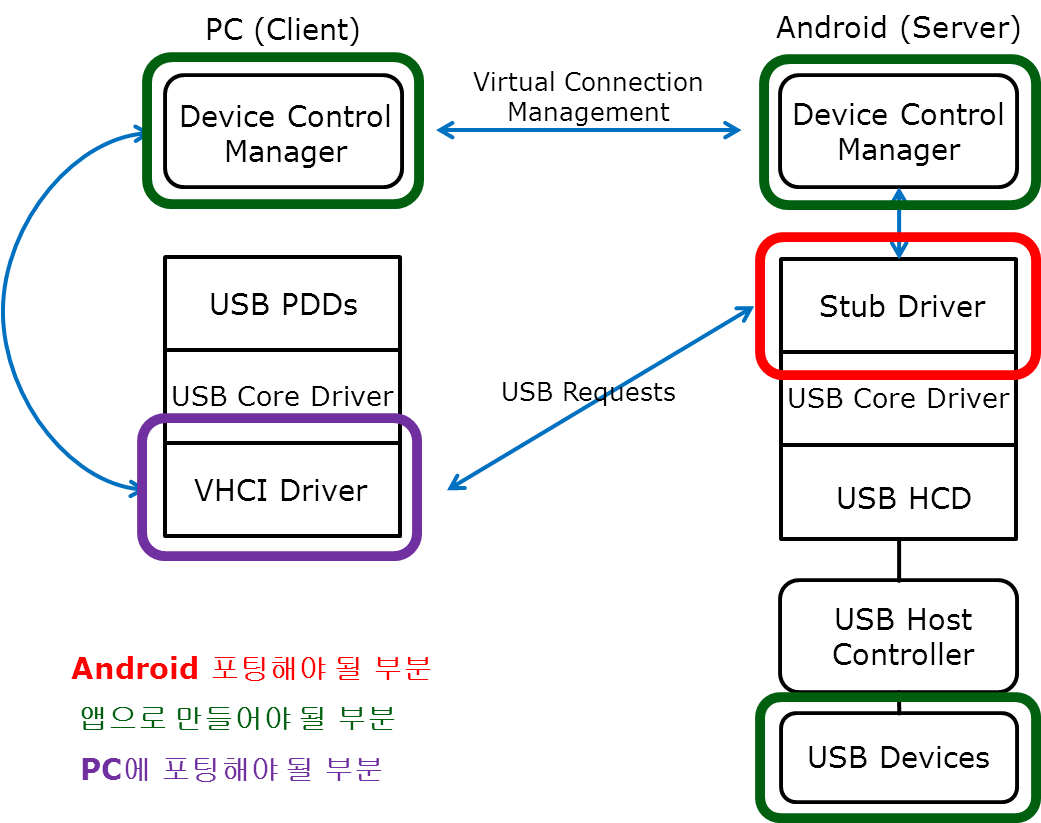
Function Layer

USB Device Layer

USB Bus

Interface Layer

* **USB/IP 구조**



* 1. **현실적 제약조건**

산업표준에 따른 제약: 프로젝트의 기반 기술인 USB/IP가 GPL라이선스를 따르고 LibUSB는 LGPL를 따르기 때문에 Android-USB 프로젝트는 GPL라이선스를 따른다.

신뢰성: USB는 이론상 최대 480Mbps(USB 2.0기준)의 전송속도를 가진다. Android-USB는 PC와 스마트폰의 연결을 인터넷을 통해 연결되기 때문에 인터넷속도에 따라 전송속도가 제한된다. USB의 전송속도를 최대한 활용하고, 안정적인 데이터 전송을 위해 속도가 1Gbps이상의 네트워크가 지원 되야 한다.

* 1. **개발 도구**
     1. **Windows XP / Ubuntu 11.10**
     2. **eclipse – indigo**
     3. **Cygwin – 1.0.3**
     4. **Android – 2.3.3**
     5. **LibUSB**
  2. **기존 소스나, 다른 과목 결과물 등 자료이용**

USB/IP

LibUSB

1. **결과로서 제출할 실적 물 목록**
   1. **소스**
   2. **매뉴얼**
   3. **실행 파일**
2. **프로젝트 수행 추진 체계 및 일정**
   1. **조직도**

지도 교수

이상현

팀장

김종욱

팀원

이윤재

팀원

강인구

팀원

* 1. **마일스톤 및 일정**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 담당자 | 3월 | | | | 4월 | | | | 5월 | | | | 6월 | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 |
| 제안서 | 초안 작성 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 수정 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* 1. **주 단위의 프로젝트 수행**

1. **참고자료**