SMouse & Faketooth

건국대학교 4학년 이영준 & 숭실대학교 4학년 김정현

개발 동기

- **스마트폰의 재발견**
  - 어떤 불편함이 있더라도 누구나 스마트폰은 항상 소지함
  - 다양한 하드웨어 장치를 탑재했음에도, 스마트폰을 하드웨어적으로 활용하는 경향은 적음

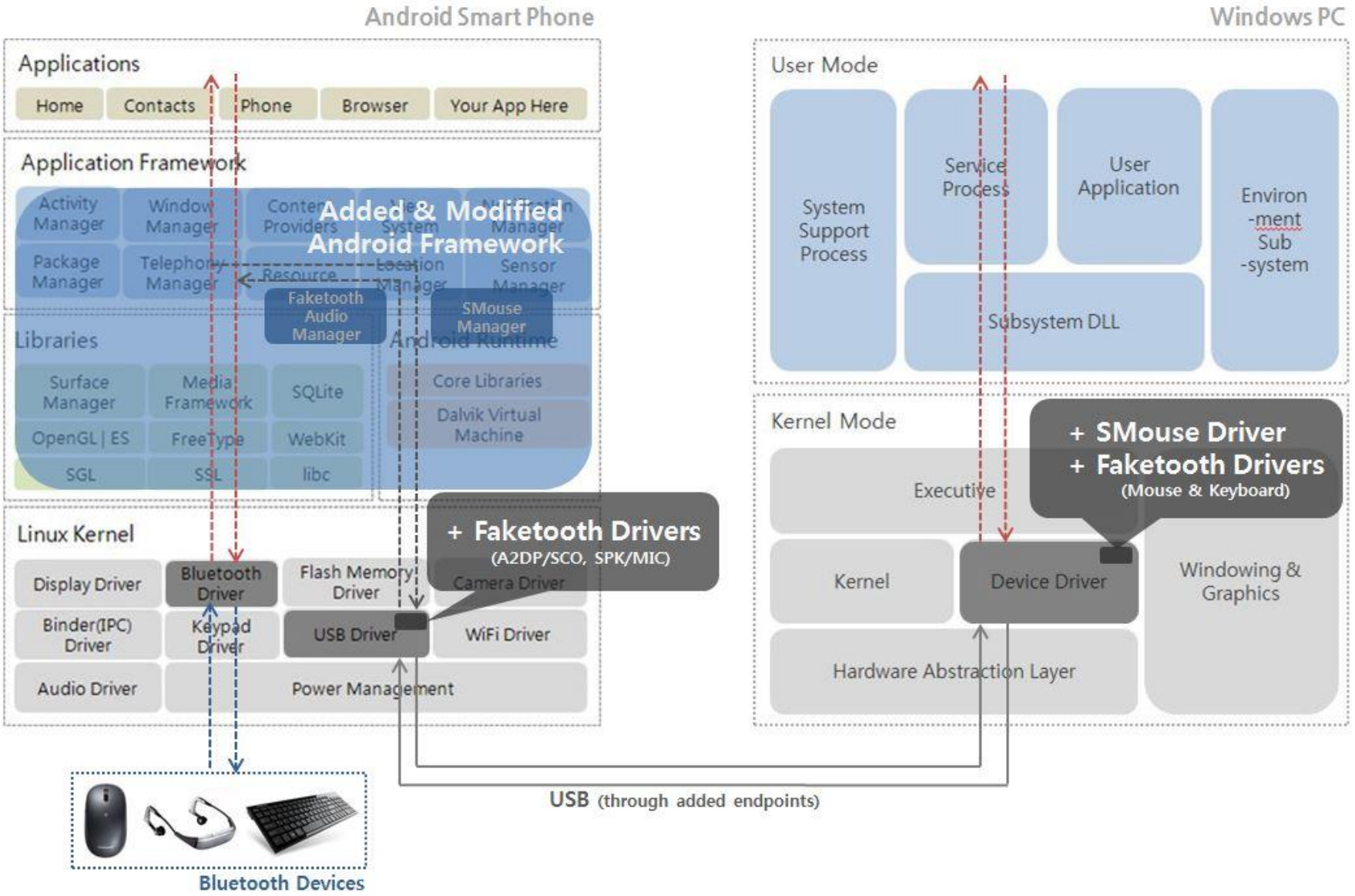
개발 동기

- **마우스를 별도로 들고 다니는 불편함**
  - 노트북 터치패드 사용이 불편한 사용자들은 마우스를 따로 구매하여 소지하고 다녀야 함

개발 동기

- 블루투스 주변기기의 제한적인 활용
  - 데스크톱 환경은 별도의 블루투스 어댑터 필요

시스템 구성도


Android Smart Phone
Windows PC
Applications
Home
Contacts
Phone
Browser
Your App Here
Application Framework
Activity Manager
Window Manager
Content Providers
Package Manager
Telephony Manager
Resource
Location Manager
Sensor Manager
Added & Modified Android Framework
Faketooth Audio Manager
SMouse Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Bluetooth Driver
Flash Memory Driver
Camera Driver
Binder(IPC) Driver
Keypad Driver
USB Driver
WiFi Driver
Audio Driver
Power Management
+ Faketooth Drivers
(A2DP/SCO, SPK/MIC)
User Mode
System Support Process
Service Process
User Application
Environ -ment Sub -system
Subsystem DLL
Kernel Mode
Executive
Kernel
Device Driver
Windowing & Graphics
Hardware Abstraction Layer
+ SMouse Driver
+ Faketooth Drivers
(Mouse & Keyboard)
USB (through added endpoints)
Bluetooth Devices

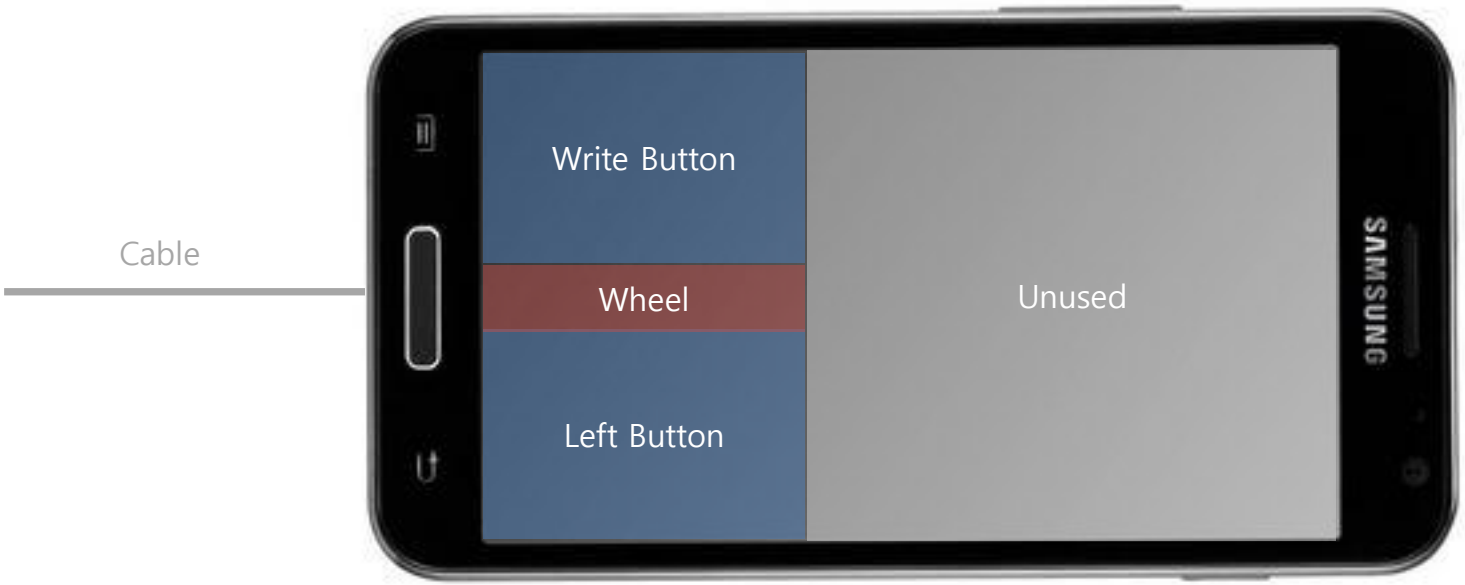
개발 내용

- Kernel - USB 통신

USB Gadget 드라이버 수정
각각 기능에 대한 파이프 개방
JNI 통한 직접 접근

개발 내용

- Button 및 Wheel



개발 내용

- ## 포인터 움직임

(1) Reactive Lock Filter
- 들고 놓는 과정에서 생기는 잡음 필터링

(2) Calibration
- 센서나 주변 환경의 기울기에 다른 중력 작용을 교정

(3) Mechanical Filtering Window
- 정지 상태에서 발생하는 미세 잡음 필터링

(4) Window Movement Check
- 이론상 정지 시 가속도의 총 적분, 속도는 0에 도달
- 현실에선 결코 0에 도달하지 않아 발생하는 움직임 조정

(5) Gyroscope Movement Check
- 자이로스코프를 활용한 정지 상태 인식

(6) Gravity Movement Check
- 중력센서를 활용한 들고 놓는 상태 인식

---

(7) Double Integral Calculus

$$\text{속도} = \text{직전속도} + (\text{가속도} * \text{샘플링 시간})$$

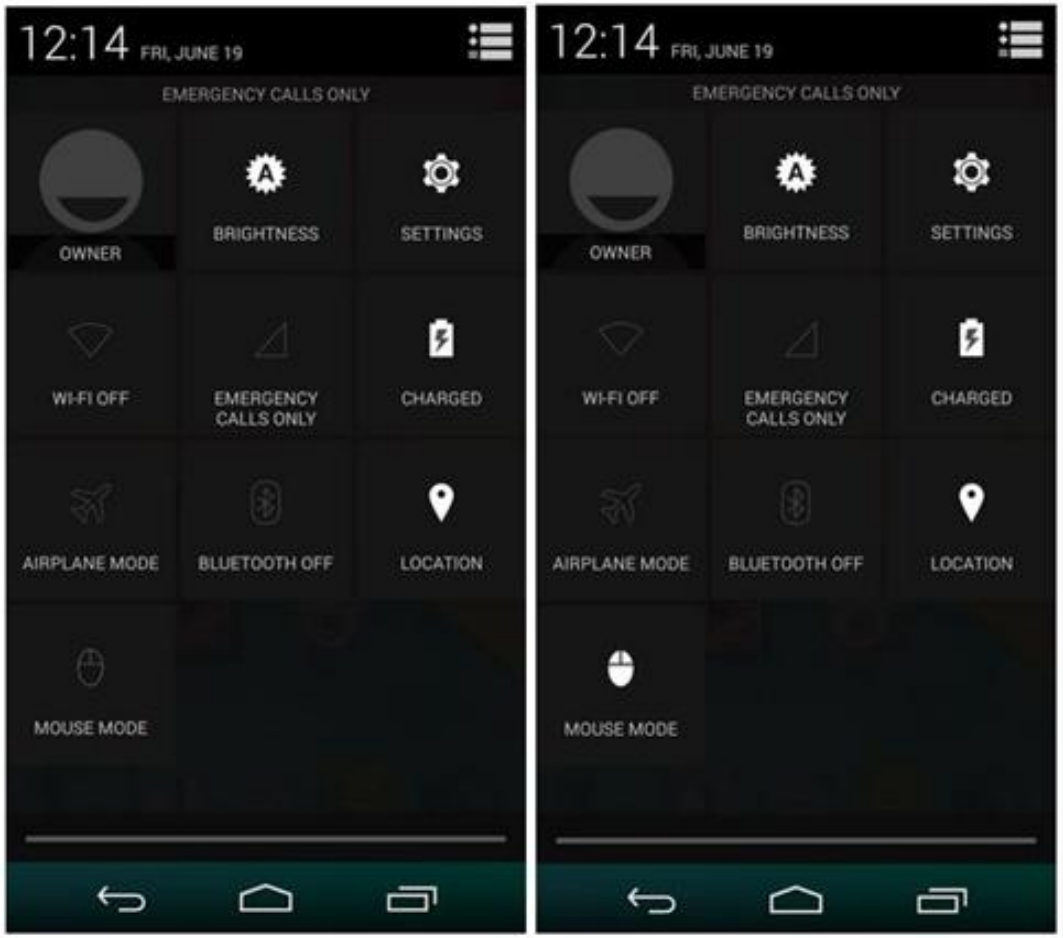
$$\text{이동거리} = (\text{직전속도} * \text{샘플링 시간}) + ((\text{가속도 값} * (\text{샘플링 시간})^2) / 2)$$

---

(8) Scale up & Positioning

개발 내용

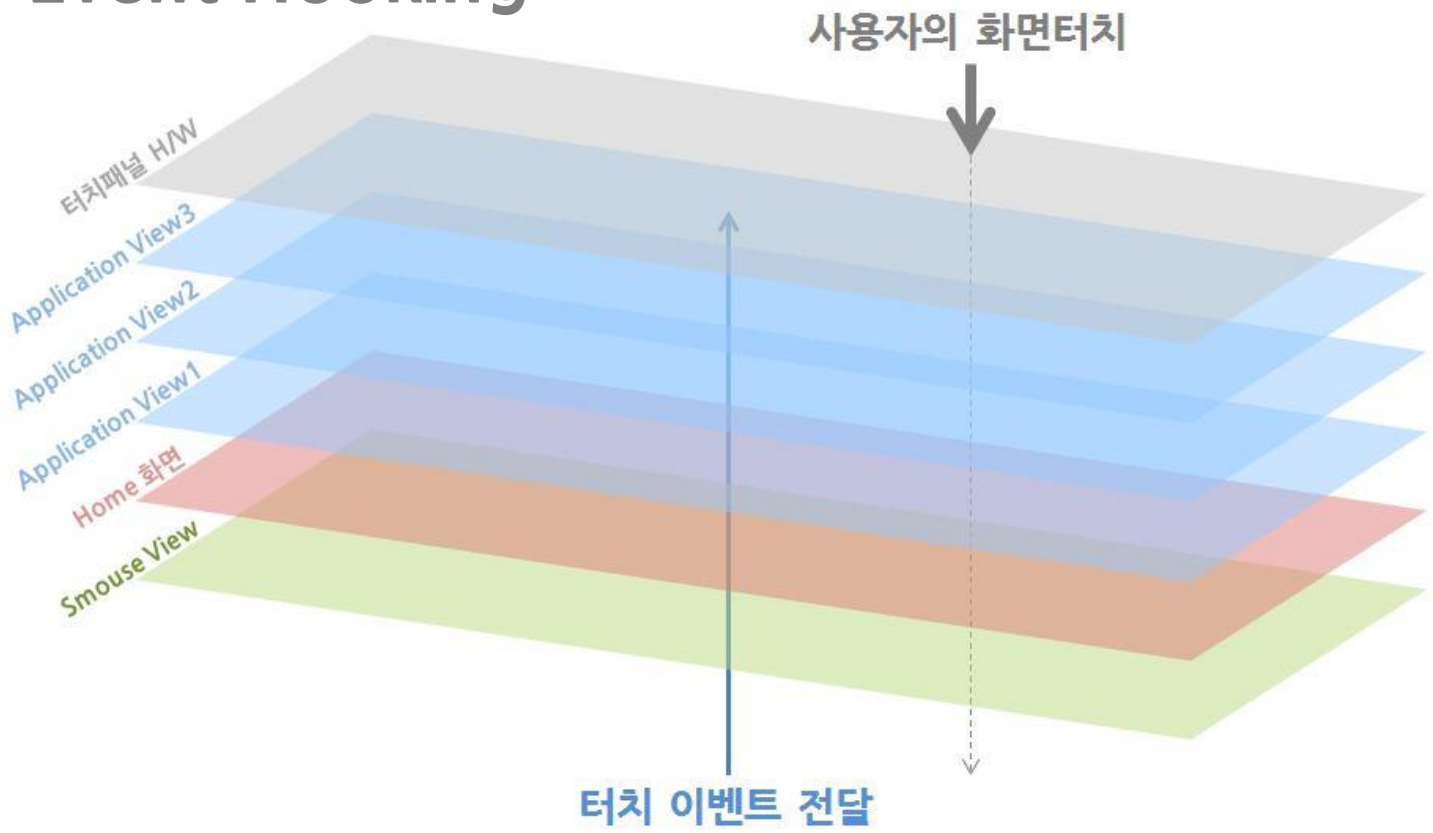
- SMouse ON / OFF



개발 내용

(SMouse 사용중에도 스마트폰 사용 가능)

- Event Hooking



일반적인 방법대로라면 최상위 하나의 View만 터치 이벤트 획득 가능

→ 최하단에 투명한 View를 삽입하여 사용자에게 영향없이 모든 이벤트 획득

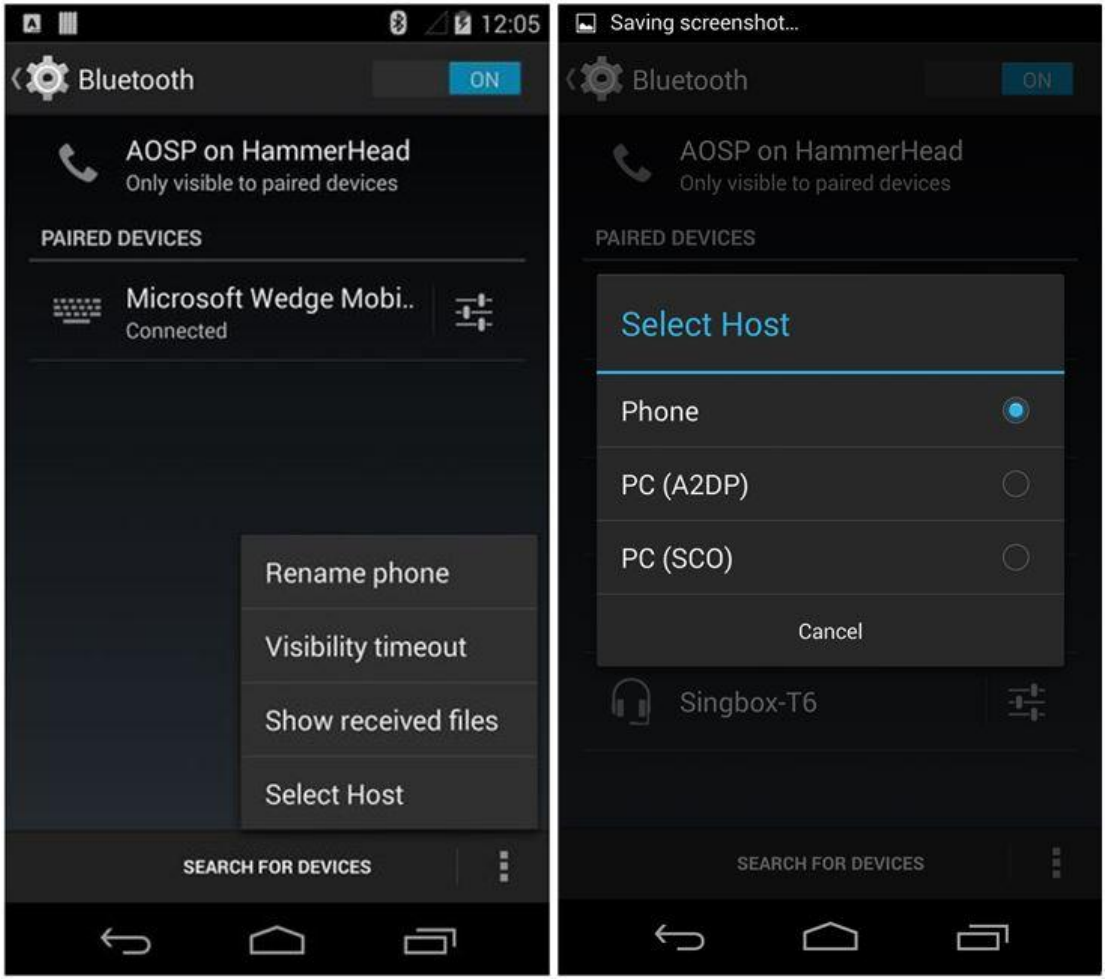
개발 내용

(스마트폰 슬립모드에서도 SMouse 사용가능)

- Keep Awake
  - Input Dispatcher Enable / Disable
    - 발생된 모든 입력 이벤트를 각각의 적절한 위치로 분배하는 부분
  - Display Touch Event Enable / Disable
    - 터치 패널을 소프트웨어적으로 켜고 끄는 부분
  - Display Power On/Off
    - 터치 패널을 하드웨어적으로 켜고 끄는 부분

개발 내용

- 선택적 사용 메뉴



개발 내용

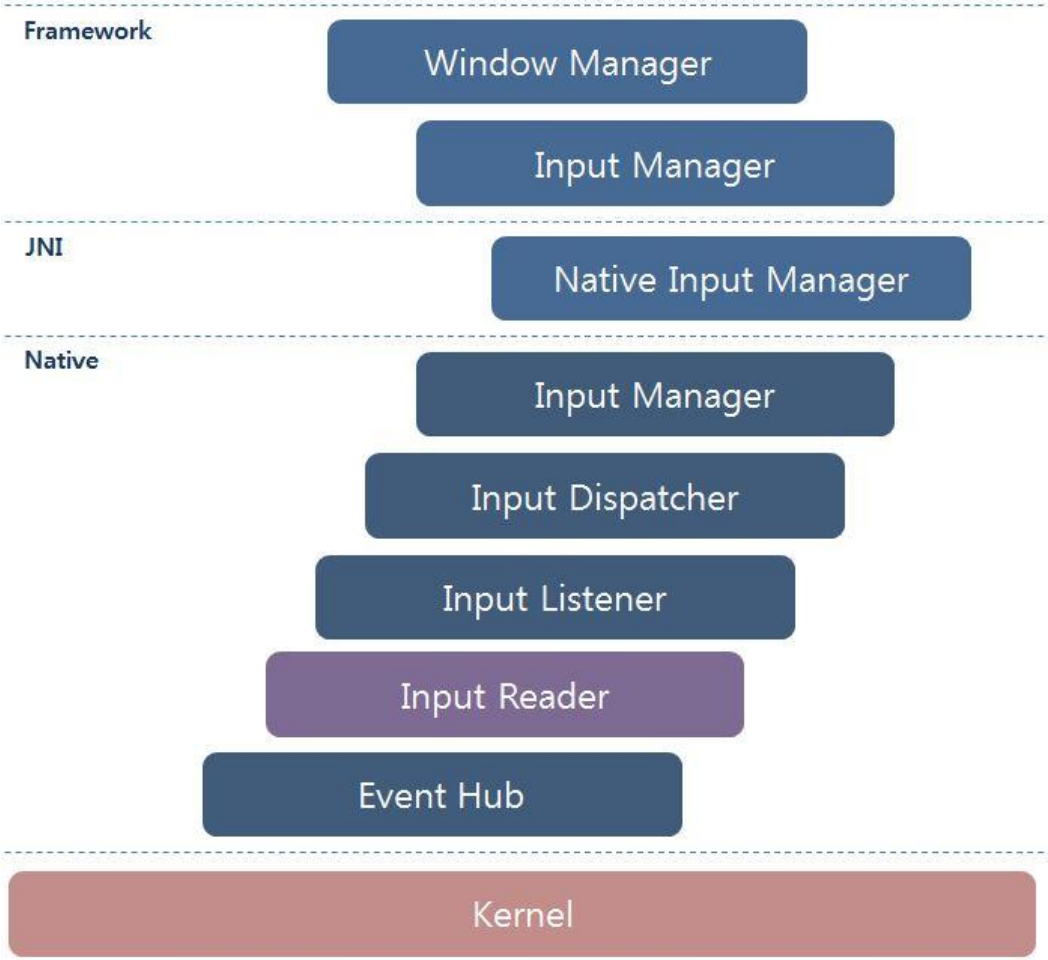
- 선택적 사용 메뉴

InputManagerService
→ (Native) InputReader

AudioService
→ (HAL) AudioPolicyManagerBase

개발 내용

- HID part



- Forward
- Hook
- Write
- Cut off

개발 내용

- HID part

```
/* Describes a motion event. */
struct NotifyMotionArgs : public NotifyArgs {
    nsecs_t eventTime;
    int32_t deviceId;
    uint32_t source;
    uint32_t policyFlags;
    int32_t action;
    int32_t flags;
    int32_t metaState;
    int32_t buttonState;
    int32_t edgeFlags;
    int32_t displayId;
    uint32_t pointerCount;
    PointerProperties pointerProperties[MAX_POINTERS];
    PointerCoords pointerCoords[MAX_POINTERS];
    float xPrecision;
    float yPrecision;
    nsecs_t downTime;

    inline NotifyMotionArgs() { }

    NotifyMotionArgs(nsecs_t eventTime, int32_t deviceId, uint32_t source, uint32
            int32_t action, int32_t flags, int32_t metaState, int32_t buttonState
            int32_t edgeFlags, int32_t displayId, uint32_t pointerCount,
            const PointerProperties* pointerProperties, const PointerCoords* poi
            float xPrecision, float yPrecision, nsecs_t downTime);

    NotifyMotionArgs(const NotifyMotionArgs& other);

    virtual ~NotifyMotionArgs() { }

    virtual void notify(const sp<InputListenerInterface>& listener) const;
};
```

```
/* Describes a key event. */
struct NotifyKeyArgs : public NotifyArgs {
    nsecs_t eventTime;
    int32_t deviceId;
    uint32_t source;
    uint32_t policyFlags;
    int32_t action;
    int32_t flags;
    int32_t keyCode;
    int32_t
    int32_t
    nsecs_t

    inline

    NotifyK

    NotifyK

    virtual

    virtual
};
```

```
enum {
    AINPUT_SOURCE_CLASS_MASK = 0x000000ff,

    AINPUT_SOURCE_CLASS_NONE = 0x00000000,
    AINPUT_SOURCE_CLASS_BUTTON = 0x00000001,
    AINPUT_SOURCE_CLASS_POINTER = 0x00000002,
    AINPUT_SOURCE_CLASS_NAVIGATION = 0x00000004,
    AINPUT_SOURCE_CLASS_POSITION = 0x00000008,
    AINPUT_SOURCE_CLASS_JOYSTICK = 0x00000010,
};

enum {
    AINPUT_SOURCE_UNKNOWN = 0x00000000,

    AINPUT_SOURCE_KEYBOARD = 0x00000100 | AINPUT_SOURCE_CLASS_BUTTON,
    AINPUT_SOURCE_DPAD = 0x00000200 | AINPUT_SOURCE_CLASS_BUTTON,
    AINPUT_SOURCE_GAMEPAD = 0x00000400 | AINPUT_SOURCE_CLASS_BUTTON,
    AINPUT_SOURCE_TOUCHSCREEN = 0x00001000 | AINPUT_SOURCE_CLASS_POINTER,
    AINPUT_SOURCE_MOUSE = 0x00002000 | AINPUT_SOURCE_CLASS_POINTER,
    AINPUT_SOURCE_STYLUS = 0x00004000 | AINPUT_SOURCE_CLASS_POINTER,
    AINPUT_SOURCE_TRACKBALL = 0x00010000 | AINPUT_SOURCE_CLASS_NAVIGATION,
    AINPUT_SOURCE_TOUCHPAD = 0x00100000 | AINPUT_SOURCE_CLASS_POSITION,
    AINPUT_SOURCE_TOUCH_NAVIGATION = 0x00200000 | AINPUT_SOURCE_CLASS_NONE,
    AINPUT_SOURCE_JOYSTICK = 0x01000000 | AINPUT_SOURCE_CLASS_JOYSTICK,

    AINPUT_SOURCE_ANY = 0xffffff00,
};
```

– 기기 분류

개발 내용

※ A2DP vs SCO

개발 내용

- Audio part

- Performance
- AudioService

개발 내용

- Audio part

Issue.

PC에서 헤드셋 사용중 발생하는 스마트폰 사운드

개발 내용

- Audio part

```
39  /* Audio stream types */
40  typedef enum {
41      AUDIO_STREAM_DEFAULT          = -1,
42      AUDIO_STREAM_VOICE_CALL       = 0,
43      AUDIO_STREAM_SYSTEM           = 1,
44      AUDIO_STREAM_RING             = 2,
45      AUDIO_STREAM_MUSIC            = 3,
46      AUDIO_STREAM_ALARM            = 4,
47      AUDIO_STREAM_NOTIFICATION     = 5,
48      AUDIO_STREAM_BLUETOOTH_SCO    = 6,
49      AUDIO_STREAM_ENFORCED_AUDIBLE = 7, /* Sounds that cann
50      AUDIO_STREAM_DTMF             = 8,
51      AUDIO_STREAM_TTS              = 9,
52      AUDIO_STREAM_FAKETOOTH        = 10, // Faketooth
53
54      AUDIO_STREAM_CNT,
55      AUDIO_STREAM_MAX              = AUDIO_STREAM_CNT - 1,
56  } audio_stream_type_t;
```

```
157  enum routing_strategy {
158      STRATEGY_MEDIA,
159      STRATEGY_PHONE,
160      STRATEGY_SONIFICATION,
161      STRATEGY_SONIFICATION_RESPECTFUL,
162      STRATEGY_DTMF,
163      STRATEGY_ENFORCED_AUDIBLE,
164      STRATEGY_FAKETOOTH, // Faketooth
165      NUM_STRATEGIES
166  };
```

- 새로운 Faketooth **전용 스트림 타입 추가**
- 새로운 Faketooth 스트림 타입 **전용 출력 정책 추가**

개발 내용

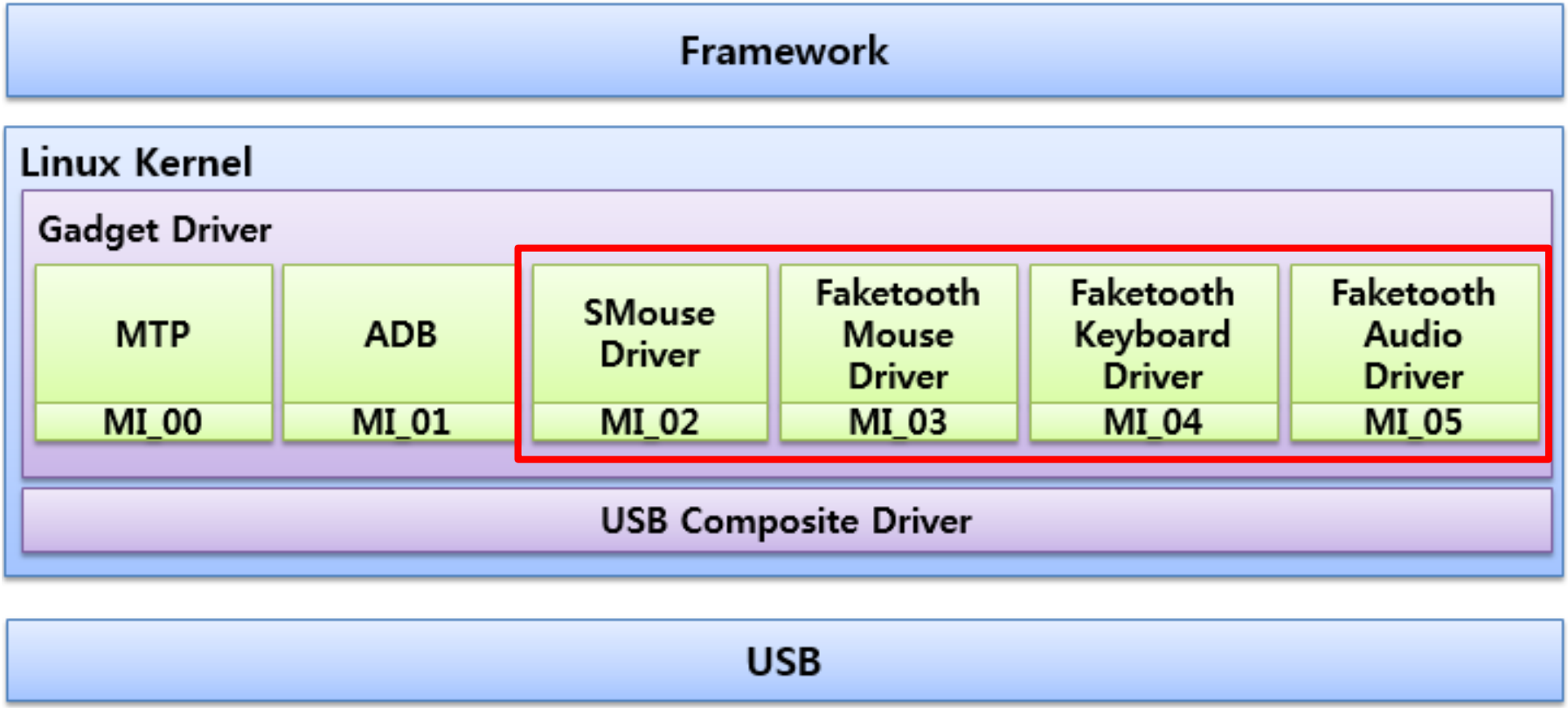
- Audio part

스마트폰 사운드 ↔ PC 사운드 **완전 분리**
스마트폰은 헤드셋이 **완전히 없는 것으로 인식**

Hardware **A**bstraction **L**ayer

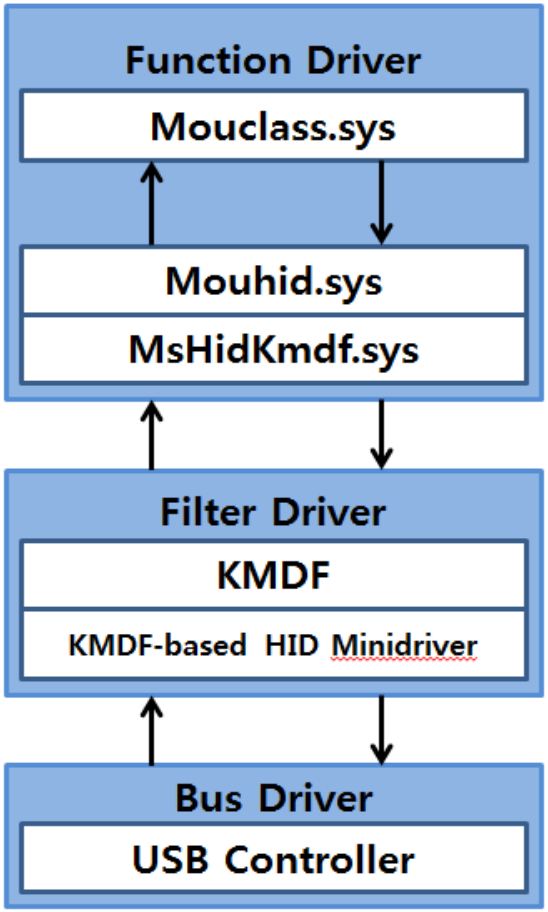
개발 내용

- Linux Kernel - Driver 구분



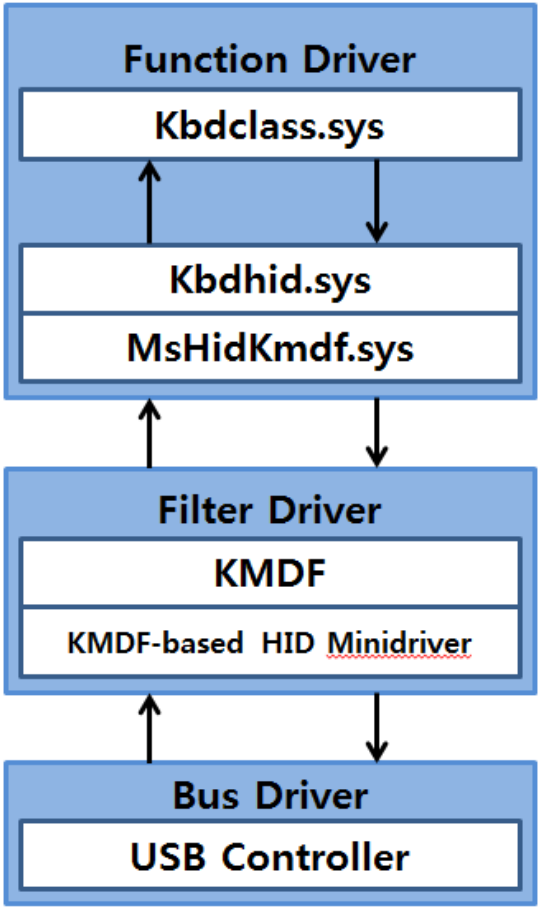
개발 내용

- Windows - Mouse Driver
  - KMDF HID MiniDriver
  - GUID : HID
  - HID Mouse Descriptor
  - Interrupt Pipe



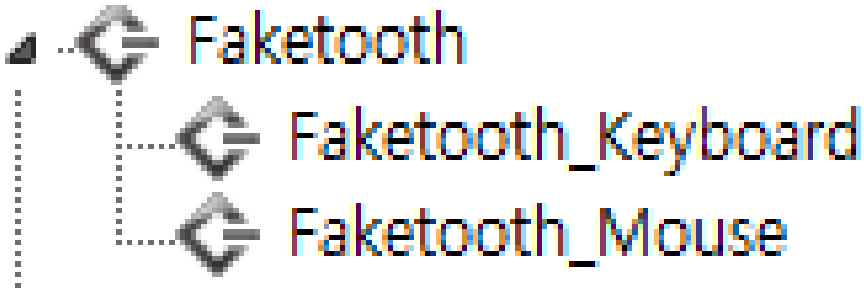
개발 내용

- Windows - Keyboard Driver
  - KMDF HID MiniDriver
  - GUID : HID
  - HID Keyboard Descriptor
  - Interrupt Pipe



개발 내용

- Windows - INF 파일 수정
  - Loading MsHidKmdf.sys
  - 새로운 Class 생성



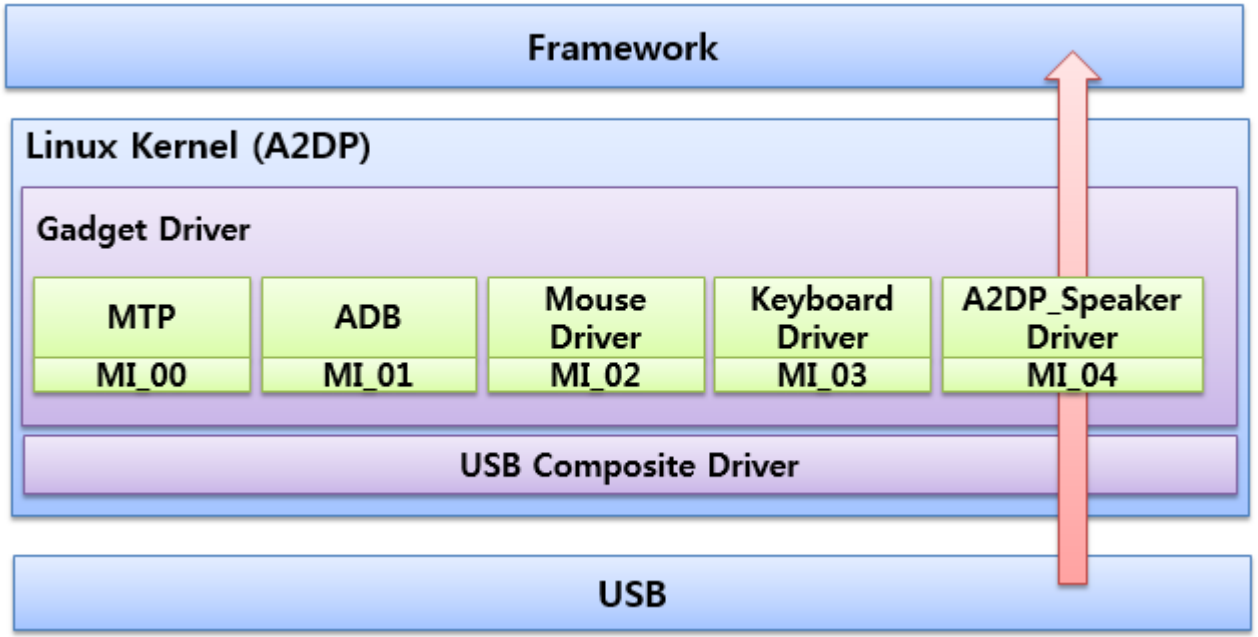
개발 내용

- Linux Kernel - Audio Descriptor

Device Descriptor	Contains general information about the device that applies to all device configurations. A USB device can have only one Device Descriptor
Configuration Descriptor	Describes a collection of interfaces (paths for data traffic). Call to retrieve this descriptor returns all subordinate descriptors (as seen below). This configuration contains 4 interfaces.
Audio Control Interface Descriptor	Describes audio interface from perspective of data paths and controls. Subordinate descriptors follow.
Interface Header Audio Class Descriptor	Lists the different data paths (terminals) and their capabilities (units).
Input Terminal Audio Class Descriptor	Describes audio control data traveling from host to device.
Feature Unit Audio Class Descriptor	Lists controls (volume, muting, etc.) associated with the above terminal.
Output Terminal Audio Class Descriptor	Describes the audio data traveling out of the device and into the host.
Audio Stream Interface Descriptor	Describes the audio stream that uses an isochronous endpoint to transfer the audio.
Alternate Audio Interface Descriptor	Required alternate interface that does not use the isochronous endpoint. This interface is chosen by the host when lack of bandwidth prevents the use of the isochronous endpoint.
Audio Stream Audio Class Descriptor	Describes the audio stream type in greater detail.
Format Type Audio Descriptor	Gives specific specifications of audio stream.
Isochronous Endpoint Descriptor	Assigns an endpoint to be the audio stream's isochronous end-point.
Isochronous Endpoint Audio Class Descriptor	Gives Audio Class-specific characteristics of the isochronous endpoint.

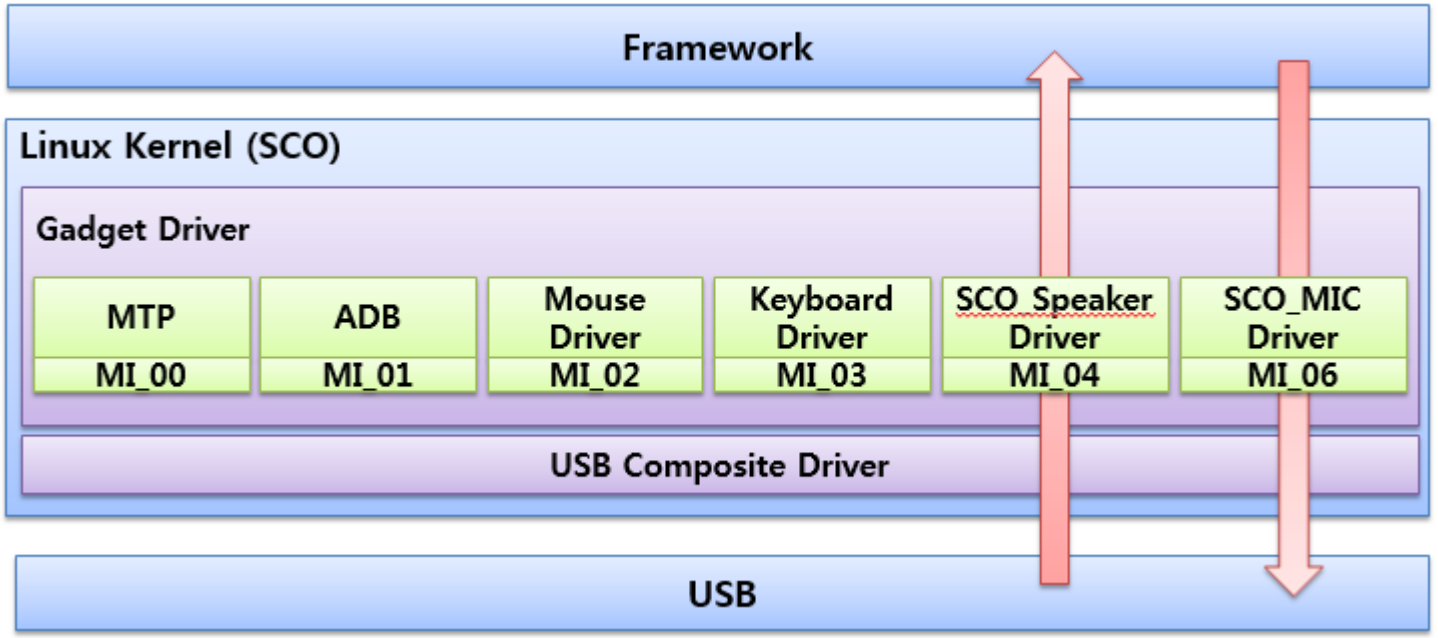
개발 내용

- Linux Kernel - A2DP Speaker
  - Sampling rate : 44100Hz, Stereo(2ch)
  - Playback_queue, Isochronous Pipe



개발 내용

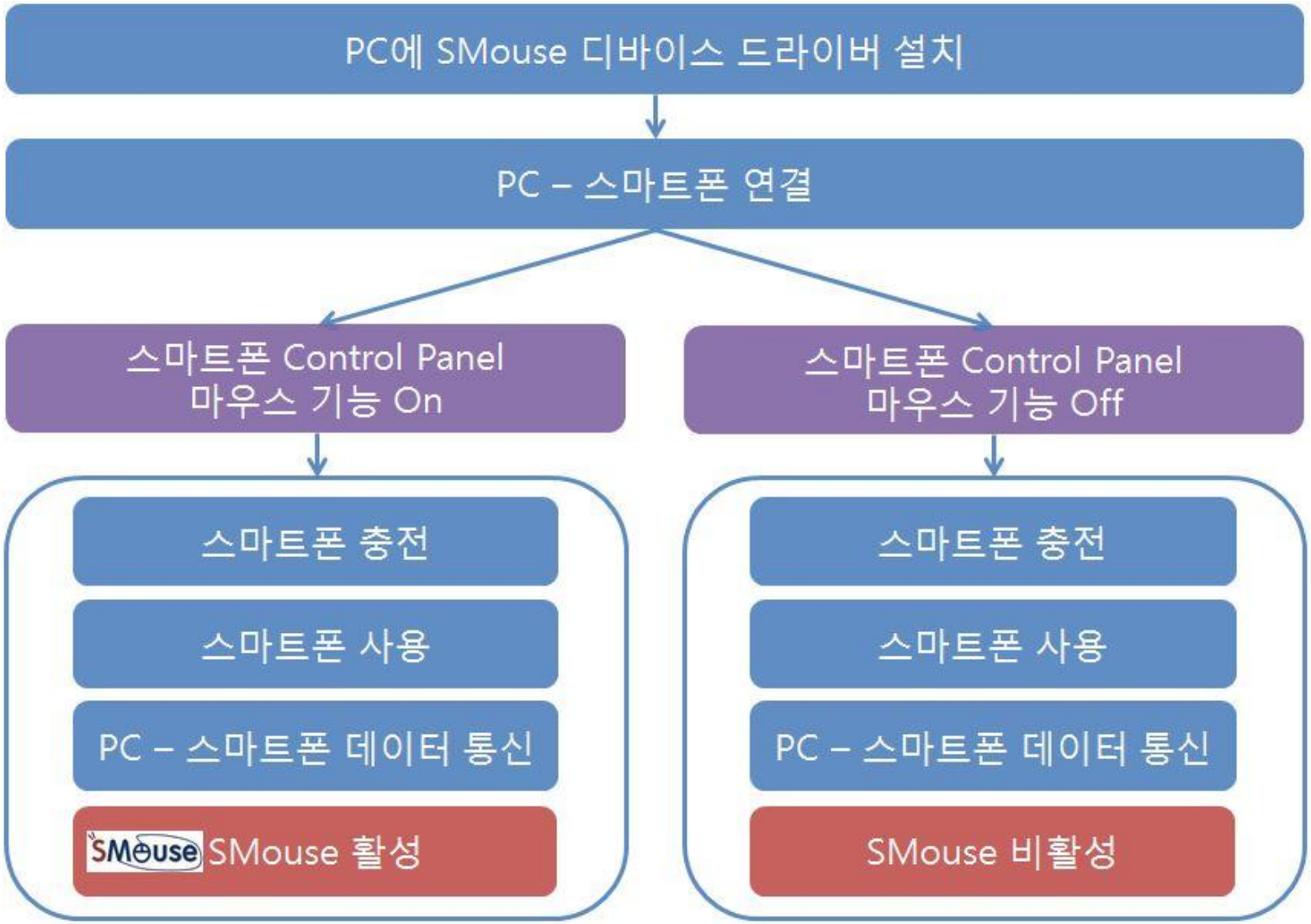
- Linux Kernel - SCO Speaker & MIC
  - Sampling rate : 8000Hz, Mono(1ch)
  - Playback & Capture_queue, Isochronous Pipe



개발 내용

- Linux Kernel - RAM disk
  - Chmod 0666 /dev/faketooth*
  - Set sys.usb.config
    - » Add function to Gadget

사용자 시나리오


PC에 SMouse 디바이스 드라이버 설치
PC – 스마트폰 연결
스마트폰 Control Panel
마우스 기능 On
스마트폰 Control Panel
마우스 기능 Off
스마트폰 충전
스마트폰 사용
PC – 스마트폰 데이터 통신
SMouse 활성
스마트폰 충전
스마트폰 사용
PC – 스마트폰 데이터 통신
SMouse 비활성

사용자 시나리오


스마트폰 - 주변기기(마우스, 키보드, 헤드셋) 블루투스 연결
스마트폰 Setting – Bluetooth – Popup menu
Select Host → √ Phone
Bluetooth
스마트폰에서 주변기기 동작
스마트폰 Setting – Bluetooth – Popup menu
Select Host → √ PC(A2DP or SCO)
Faketooth
PC에서 주변기기 동작

개발 환경

- ✓ Windows 7 Professional x86
- ✓ Ubuntu 12.04.4 LTS x64
- ✓ Android 4.4.2 KitKat
- ✓ Android Kernel 3.4.0
- ✓ Nexus 5 (KOT49H)

시연