SAMSUNG

S/W멤버십 과제 제안서

과제명 : { Android Virtualization을 이용한 IoT 구현 }

- 소 속 : 강남 소프트웨어 멤버십
- 작성자 : 윤재석, 장정규, 최현빈

- 작성일 : 2014. 03. 17

삼성소프트웨어멤버십
secmem.org

S/W멤버십 과제 기획서

과 제 명	Android Virtualization을 이용한 IoT 구현		
과제구분	창의과제		
과제기간	2014. 04. 01 ~ 05. 31 (2개월)		
지 역	강남 멤버십	참여인원	3 명

회 원 명	학교	학과	학년	연락처	E-mail
윤재석	숭실대	컴퓨터학부	4	010-2634-0424	yjaeseok@gmail.com
장정규	단국대	소프트웨어학과	3	010-2662-0890	jkinject@gmail.com
최현빈	건국대	컴퓨터공학부	4	010-9799-1407	binensky@gmail.com

구 분	내 용
개발 목적 및 동기	이전에 창의과제로 개발했던 'Dark Cloud'에서는 동기화 방식이 아니라 Server에서 Android Phone으로 바로 Dark Cloud 폴더를 Mount시켜 사용하는 방식을 사용했다. 그래서 '진정한 Cloud'라는 주제에 다가섰지만, 그 내용이 '파일 스토리지'기능에만 집중되어 있다는것에 아쉬움을 느꼈다. 이에 우리팀은 Android Virtualization을 통해 Internet Of Thing(IoT)를 구현하여 Cloud Phone을 개발하기로 하였다.
개발 환경 및 일정	OS : Windows 7 Tool : Node.JS, Android Development Kit(Eclipse), Adnroid Debug Bridge, Android Virtual Device, MySQL
창의성/ 우수성	Dark Cloud는 엄밀히 말해 파일 스토리지 서비스에 국한되어 있다. 하지만 Cloud Phone은 Android Virtualization을 통하여 Device에 종속적이지 않은, 진정한 Cloud Service를 이용할 수 있다.
활용성/ 사업성	이 서비스가 제대로 개발이 된다면, 사용자는 Android 기기가 아무리 자주 바뀌어도, 잃어버려도 언제 어디서든지 자신이 사용하던 OS, Application을 사용할 수 있으므로 매우 활용성이 높다.

지원부서	S/W멤버십	기술 지원 연구원	정동욱 연구원
전화번호	010-9155-8977	E-Mail	donguk.jung@samsung.com

목 차

- 내용 목차 -

1. 개발 목적 5
    1) Internet of Things(IoT) 5
    2) 휴대폰 교체 주기에 따른 문제점 5
    3) Cloud Phone의 활용 6
2. 개발 목표 6
    1) Cloud Phone Server 구축 6
    2) Android Virtual Device FrameBuffer 압축 및 전달 6
    3) Android framework for CloudPhone 6
    4) CloudPhone Service 7
    5) Virtual Web Cam for Android Virtual Device 7
3. 개발 내용 7
    1) System Architecture 7
    2) Cloud Phone Server 8
    3) Cloud Phone Framework 및 센서 10
    4) Virtual Web Cam for Android Virtual Device 12
    5) Evaluation Factors 14
4. 개발 일정 14
    1) 윤재석 14
    2) 장정규 15
    3) 최현빈 15
5. 용어 정리 16
    1) Internet of Things 16
    2) Android Virtual Device 16
    3) Android Debug Bridge 16
    4) Frame Buffer 16
    5) Windows Driver Kit 16
6. 참고 문헌 16

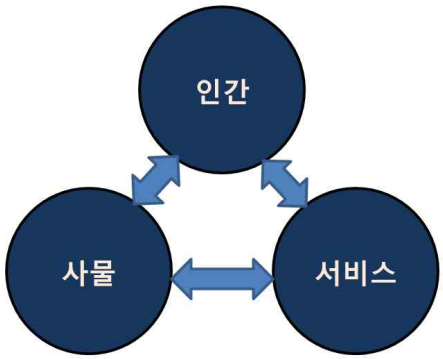
- 그림 목차 -

[그림 1] Internet of Things(IoT)..... 5
[그림 2] 휴대폰 교체주기(경향신문)..... 5
[그림 3] Cloud Phone System Architecture..... 7
[그림 4] MySQL & Node.JS..... 8
[그림 5] Android VIrtual Device 작동 화면..... 9
[그림 6] Android Debug Bridge 작동 화면..... 9
[그림 7] Android 화면이 Framebuffer에 저장되는 과정..... 10
[그림 8] 안드로이드 시스템 더블버퍼링 구조..... 11
[그림 9] 장치 필터 드라이버의 동작..... 13
[그림 10] AVStream Overview..... 13
[그림 11] Evaluation Factors..... 14
[그림 12] 윤재석 일정..... 14
[그림 13] 장정규 일정..... 15
[그림 14] 최현빈 일정..... 15

1. 개발 목적

1) Internet of Things(IoT)

Internet of Things(이하 IoT)는 인간과 사물, 서비스 세 가지 분산된 환경 요소에 대해 인간의 명시적 개입 없이 상호 협력적으로 센싱, 네트워킹, 정보 처리 등 지능적 관계를 형성하는 사물 공간 연결망이다.



[그림 1] Internet of Things(IoT)

IoT(사물인터넷)이 되기 위한 가장 중요한 시스템 중 하나가 클라우드 시스템이라고 할 수 있다. 모든 사물들(Things)이 인터넷을 통해 클라우드에 연결됨으로써 사용자는 실시간으로, 모든 상황을 인지할 수 있는 것이다.

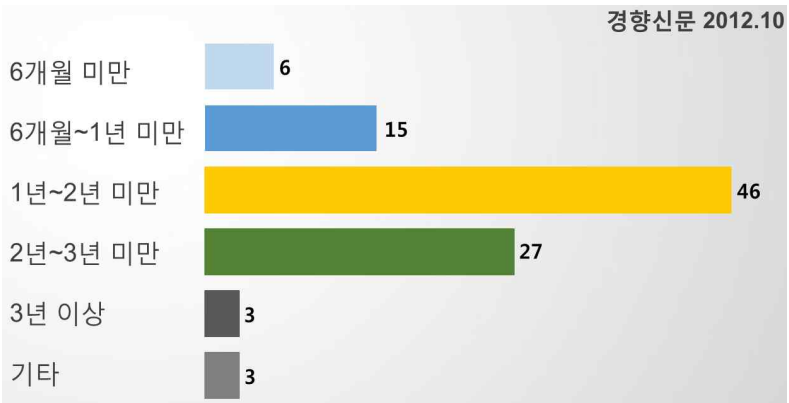
이 것을 우리가 평소 사용하는 스마트폰에 적용시켜보자는 아이디어에서 Cloud Phone 프로젝트를 시작하게 되었다.

2) 휴대폰 교체 주기에 따른 문제점

요즘 스마트폰 디바이스를 사용하는 대부분의 사람들은 자신이 산 핸드폰의 약정이 끝나기만을 기다린다.

이것은 하드웨어 및 소프트웨어의 급격한 발전에 따라 신형 스마트폰의 출시도 그만큼 잦아진 이유가 없지 않아 있다.

그래서 약정기간이 기본 2년, 길어야 3년밖에 안하는 현실에서, 실제 직장인들의 휴대폰 교체 주기는 1년~2년 미만의 짧은 기간이 대부분이었다.



[그림 2] 휴대폰 교체주기(경향신문)

하지만 휴대폰 성능이 좋아진다는 장점 뒤에는, 자신이 쓰던 어플리케이션을 다시 설치하며 그에 맞는 설정을 세팅해야 한다는 점, 연락처 옮기기와 사진 및 백업 데이터의 이동 등 여러 가지 문제점이 있었다.

이에 우리 Cloud Phone 팀은 가상화 기술을 이용하여 하드웨어에 구애받지 않는 'Cloud Phone'을 개발해보고자 한다.

3) Cloud Phone의 활용

Cloud Phone은 가상화 기술을 사용하여 실제로 상용화된다면 언제 어디서나 자신이 스마트폰을 바꾸어도 하드웨어에 구애받지 않고 Cloud Phone 서버를 통해 사용하던 어플리케이션, 사진, 동영상 등을 그대로 사용할 수 있다.

2. 개발 목표

1) Cloud Phone Server 구축

Android Virtual Device를 사용하여 PC에 가상화 시킨 화면을 실제 클라이언트 디바이스로 전송을 시키는 Cloud Phone Server를 구축한다. 서버 컴퓨터의 데이터베이스에 저장되어 있는 클라이언트 정보를 가져오면, 서버는 그 정보를 클라이언트와 접속하여 화면을 그대로 뿌려주게 된다.

또한, 클라이언트 디바이스에서 발생하는 센서 값들을 매핑시켜서, 실제 안드로이드 기기를 PC에서 사용하는 것처럼 보이게 인식할 것이다.

2) Android Virtual Device FrameBuffer 압축 및 전달

Android Framework내에는 Framebuffer Driver가 존재한다. Framebuffer Driver는 Android 기기의 화면에 출력되는 모든 정보를 가지고 있다. 그러므로 Framebuffer를 통해서 서버에서 동작하는 AVD(안드로이드 가상 디바이스)의 화면을 실시간으로 추출할 수 있다. 추출한 Framebuffer는 디바이스의 해상도 크기 만큼의 픽셀 데이터를 가지고 있다. 그러므로 얻어진 Framebuffer의 용량이 매우 크다. 속도 개선을 위해 손실 압축방법의 표준방식인 JPEG 알고리즘으로 압축을 하여 용량을 축소하고 각각의 CloudPhone(실제 디바이스)으로 전송할 것이다.

3) Android framework for CloudPhone

CloudPhone을 위한 안드로이드 프레임워크란 서버에서 동작하는 AVD(안드로이드 가상 디바이스)에서 동작하는 Android Framework와 실제 디바이스에서 동작할 Framework를 통칭한다. 서버용 프레임워크는 기존 안드로이드 시스템에 FrameBuffer 추출 및 전송부분이 추가되고, 클라이언트용은 기존 안드로이드 시스템을 대폭 수정하여 스마트폰이 부팅이 완료된 후 동작할 때 WIFI 연결 및 계정 설정 등을 통해 CloudPhone서버에 연결할 수 있는 환경을 제공한다. 서버와 연결되면 서버로부터 받은 FrameBuffer 화면만 실시간으로 출력하는 기능을 수행 한다.

4) CloudPhone Service

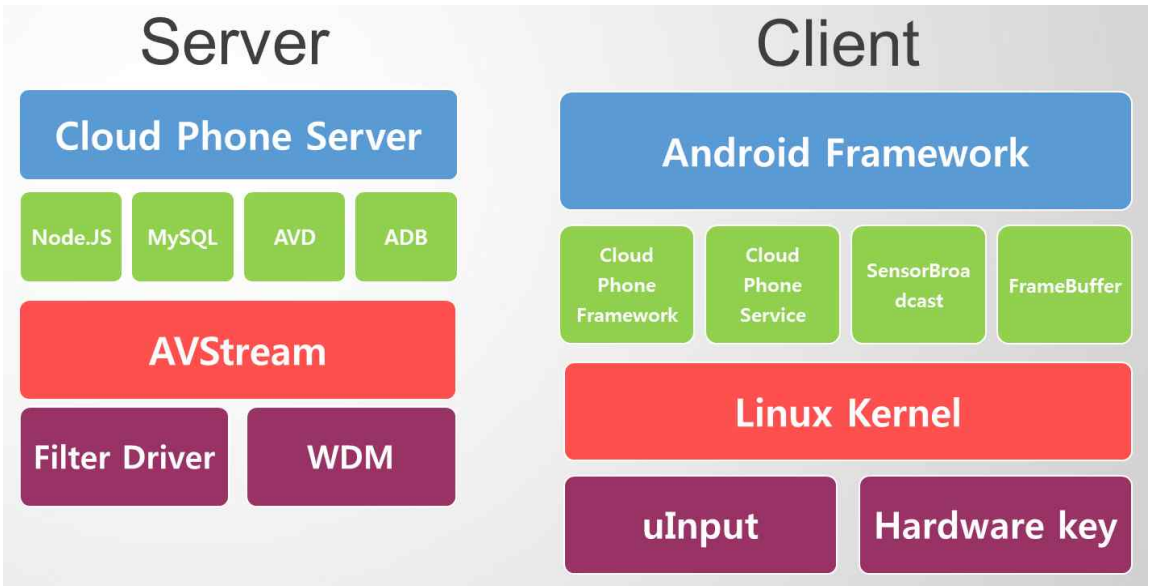
CloudPhone Service는 스마트폰의 센서 장치의 데이터를 수집 및 전송하는 기능을 수행한다. 서버에서 동작하는 AVD는 가상으로 존재하므로 클라이언트 디바이스에 장착된 센서를 활용할 수없고, Touch 정보, Hardware Key Input 정보를 AVD에 매핑시키는 작업을 해야 한다. 그러므로 스마트폰에 존재하는 센서중 GPS, 자이로센서, Battery 데이터, Touch 정보, Hardware Key Input 정보를 수집하여 Server로 전송한다. 이로써 사용자는 서버에 존재하는 가상 디바이스가 아니라 실제 자신의 휴대폰을 사용하는 착각하게 될 것이다.

5) Virtual Web Cam for Android Virtual Device

최근 Android SDK에 포함되어있는 Android Virtual Device는 Windows에 연결되어 있는 Web Cam을 카메라로 인식시킬 수 있는 기능을 가지고 있다. 다시 말하면, Android Virtual Device도 카메라를 활용한 어플리케이션 테스트를 할 수 있는 환경이 제공된다는 것이다. 이 점을 착안하여 실제 핸드폰의 Camera를 서버 컴퓨터의 Virtual Web Cam으로 제공해줄 수 있다면, CloudPhone의 카메라 기능도 구현할 수 있을 것이라고 생각하여 이 부분을 개발 목표로 선정하였다. 먼저 CloudPhone을 통해 새로운 Android Virtual Device를 할당 받게 되면 서버에 새로운 Web Cam 하나가 생성되도록 구현할 것이다. 후에 CloudPhone에서 카메라 기능을 실행하면 해당 폰의 카메라에서는 카메라 데이터를 생성하며, 생성한 데이터를 서버로 전달하도록 구현할 것이다. 서버에서는 전달된 데이터를 Virtual Web Cam에 출력되도록 하여 결론적으로는 Android Virtual Device에 스마트 폰의 카메라를 통한 데이터가 표현될 수 있도록 구현할 것이다.

3. 개발 내용

1) System Architecture



[그림 3] Cloud Phone System Architecture

2) Cloud Phone Server

(1) Node.JS와 MySQL을 통한 웹 서버 구축

Cloud Phone Server를 구축하기 위한 제일 첫 단계로, Node.JS Framework를 사용하여 웹 서버를 구축한다. 전작 Dark Cloud에서도 사용하였던 Node.JS는 비교적 문법이 간단한 javascript 언어로 작성되어 있고, 확장성 있는 이벤트 기반의 네트워크 서버를 작성할 때 매우 유용하기 때문에 선택하였다.

또한 MySQL을 사용하여 DB를 구축할 것이다. 뛰어난 성능과 저렴한 비용으로 강력한 DB를 구축할 수 있어서 선택하였으며, DarkCloud에서 No SQL에 도전하였다면, 이번에는 SQL을 사용하여 강력한 검색기능 및 조합 기능을 활용할 예정이다.

[그림 4] MySQL & Node.JS

(2) 클라이언트와 통신할 프로토콜 설계 및 구현

Cloud Phone에서는 서버에서 화면을 클라이언트로 뿌려줘야 하는데, 그 작업을 하기 위해선 통신하여 데이터를 주고받을 수 있는 프로토콜이 있다는 것을 전제로 해야 한다. 따라서 웹 서버에서 클라이언트로 데이터를 주고받을 수 있는 하나의 프로토콜을 설계 및 구현한다.

Node.JS를 사용(Java Script)하며, 기본적으로 웹서버와 클라이언트가 연결되면 계속 데이터를 주고받을 수 있도록 하나의 세션을 설정할 것이다.

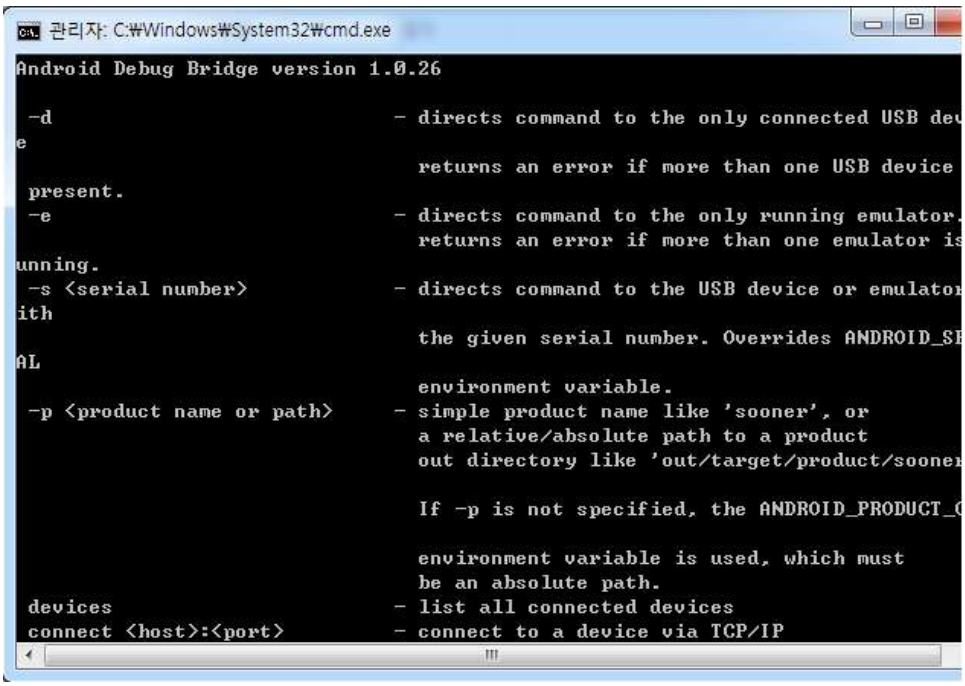
(3) AVD와 ADB를 활용한 Android 화면 가상화

Android Virtual Device는 실제 Android 기기가 아닌, PC에서 똑같이 Android를 이용할 수 있도록 해주는 일종의 Emulator이다. 이번 Cloud Phone에서는 AVD를 이용하여 이 가상 Android 장치에서 일어나는 일들을 모두 실제 클라이언트 기기에 전송 할 것이다.

[그림 5] Android VIrtual Device 작동 화면

이 Android Virtual Device를 실제 프로그램에서 작동시키기 위해서는 Android Debug Bridge(ADB)라고 하는 프레임워크가 필요하다. 이 프레임워크를 사용하여 AVD 장치를 이용할 수 있으며, 심지어는 파일 복사 및 어플리케이션 설치/삭제까지 시킬 수 있다.

ADB shell을 통해 안드로이드에 직접 접속하거나 혹은 간접적으로 명령을 내릴 수도 있다. 이를 통해 Android 화면을 제어할 수 있도록 Cloud Phone 서버를 개발할 것이다.



[그림 6] Android Debug Bridge 작동 화면

(4) 각종 센서 값 매핑

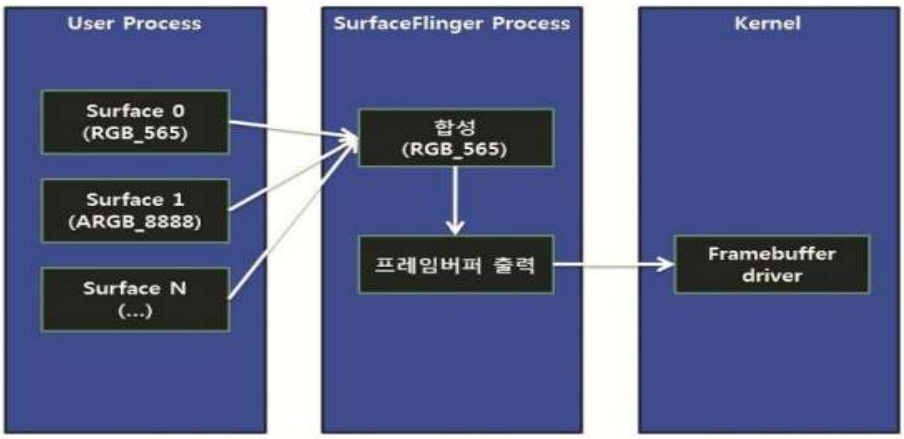
서버를 통하여 주고받는 데이터는 서버에서 클라이언트기기로 보내는 화면과 각종 값들만 있는것이 아니라, 클라이언트에서 서버로 보내는 센서의 정보들도 많다ㅇ.

예를 들어 GPS, 카메라, Hardware Key, 자이로스코프 센서값 등이 있다. 이 값들이 전송되어 실제 Android에 적용되는 것처럼 보여야 하기 때문에, 이 센서 값들을 전송받아 실제 기기에 매핑 시키는 작업을 한다.

3) Cloud Phone Framework 및 센서

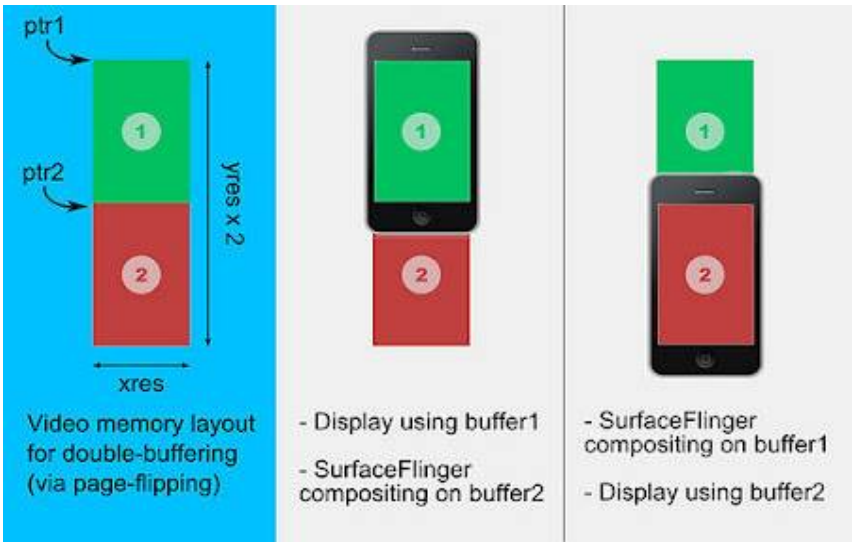
(1) Android Virtual Device FrameBuffer 압축 및 전달

CloudPhone을 구현하기 위해서는 가상 디바이스의 화면을 실시간으로 실제 디바이스에 출력하는 기술을 구현해야 한다. 먼저 기기의 화면을 출력하기 위해서 Android Framework에 존재하는 Framebuffer Driver를 이용하여, Android 기기의 화면 Buffer를 추출할 수 있다.



[그림 7] Android 화면이 Framebuffer에 저장되는 과정

Framebuffer는 안드로이드 시스템내 '/dev/graphics/fb0' 라는 파일로 위치한다. 하지만 파일의 권한이 660으로 READ 권한이 없으므로 권한을 664로 변경 후 사용가능하다. 추출하여 떨어진 Byte Array 정보에는 디바이스의 해상도 크기 만큼의 픽셀 데이터의 두배를 가지고 있다. 왜냐하면 안드로이드 시스템은 더블버퍼링을 사용하므로 버퍼에 Background Buffer와 Front Buffer를 모두 저장하고 있으므로 얻어진 데이터의 용량이 매우 크다. 속도 개선을 위해 한개의 버퍼만 사용하고, 손실 압축방법의 표준방식인 JPEG 알고리즘으로 압축을 하여 용량을 축소하고 각각의 CloudPhone(실제 디바이스)으로 전송할 것이다.



[그림 8] 안드로이드 시스템 더블버퍼링 구조

(2) Android framework for CloudPhone

CloudPhone을 구현하기위해서는 안드로이드 프레임워크를 수정해야 한다. 기존 안드로이드 소스코드를 받아서 SystemUI와 Default Launcher변경을 통해서 커스텀 안드로이드 프레임워크를 구현을 목표로 한다. CloudPhone이 부팅이 되면 스마트폰을 설정하는 화면이 나타난다. 우선 인터넷이 사용 가능해야 되므로 WIFI 설정 화면이 나타나도록 구현할 것이다. 인터넷이 연결된 후 로그인을 통해 인증과정을 거치게 되면 클라이언트는 모든 설정이 완료 된다. 서버와 연결되면 서버로부터 받은 FrameBuffer 화면만 실시간으로 출력하는 기능을 수행 한다.

(3) CloudPhone Service

CloudPhone Service는 Android 시스템이 부팅완료 시점부터 서비스는 동작하고, 스마트폰의 센서 장치의 데이터를 수집 및 전송하는 기능을 수행한다. 스마트폰에 존재하는 센서들의 정보를 수집하여 Server로 전송한다. 이로써 사용자는 서버에 존재하는 가상 디바이스가 아니라 실제 자신의 휴대폰을 사용하는 착각하게 될 것이다. 서비스에서 수집하는 정보는 아래 표로 정리되어 있다.

GPS	위도
	경도
	고도
자이로센서	각각도
	가속도
베터리	잔량
	충전 상태
Touch Event	ACTION_DOWN
	ACTION_MOVE

	ACTION_UP
	ACTION_POINTER_DOWN
	ACTION_POINTER_UP
Hardware Key Input	KEYCODE_HOME
	KEYCODE_VOLUME_DOWN
	KEYCODE_VOLUME_UP
	KEYCODE_POWER

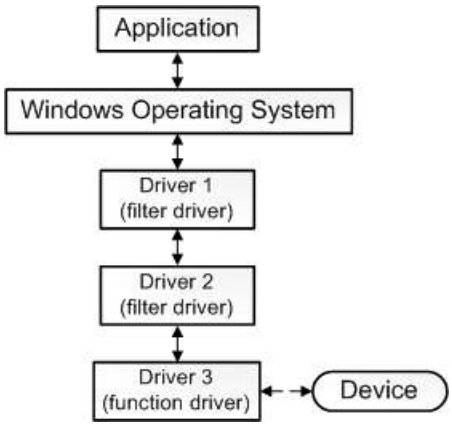
4) Virtual Web Cam for Android Virtual Device

(1) Android Camera Mechanism

CloudPhone에서 카메라에 접근하게 되면, 해당 폰은 Service로 카메라가 작동된다. Service에서는 Camera.PreviewCallback 인터페이스를 활용하여, 카메라에서 촬영된 데이터를 Byte Array로 받을 수 있도록 구현할 것이다. 떨어진 Byte Array 정보는 기본적으로 YUV(YCbCr) 형태로 리턴하도록 구현되어있다. YUV 형태란 RGB와 마찬가지로 색을 표현하는 하나의 표현 방법을 말한다. YUV는 휘도 신호(Y), 휘도 신호와 청색 성분의 차이(Cb) 휘도 신호와 적색 성분의 차이(Cr)로 구현되어 있다. 그러나 YUV 형태의 이미지는 JPEG 형태의 이미지보다 용량대비 품질이 떨어지는 면을 보인다. 따라서 YUV 형태의 이미지를 JPEG 형태로 변환하여 서버로 전달하도록 구현할 것이다.

(2) Windows Driver Kit(WDK)

CloudPhone에서 새로운 핸드폰을 생성하게 되면 Server에서는 Android Virtual Device를 생성하고 Virtual Web Cam을 생성하도록 구현할 것이다. Windows에서 Virtual Web Cam 을 구현하기 위해서는 Windows Driver Kit(이하 WDK) 이라는 프레임워크를 활용해야 한다. WDK란 Windows를 위한 드라이버를 빌드, 테스트, 디버그 및 배포할 수 있는 도구가 포함된 프레임워크를 말한다. WDK 를 활용하여 드라이버를 제작하기 위해서는 Driver Model을 선택해야한다. Driver Model의 종류에는 장치 기능 드라이버, 장치 필터 드라이버, 소프트웨어 드라이버, 파일 시스템 필터 드라이버, 파일 시스템 드라이버가 있으며 우리는 이번 프로젝트에서 장치 필터 드라이버를 활용하고자 한다.

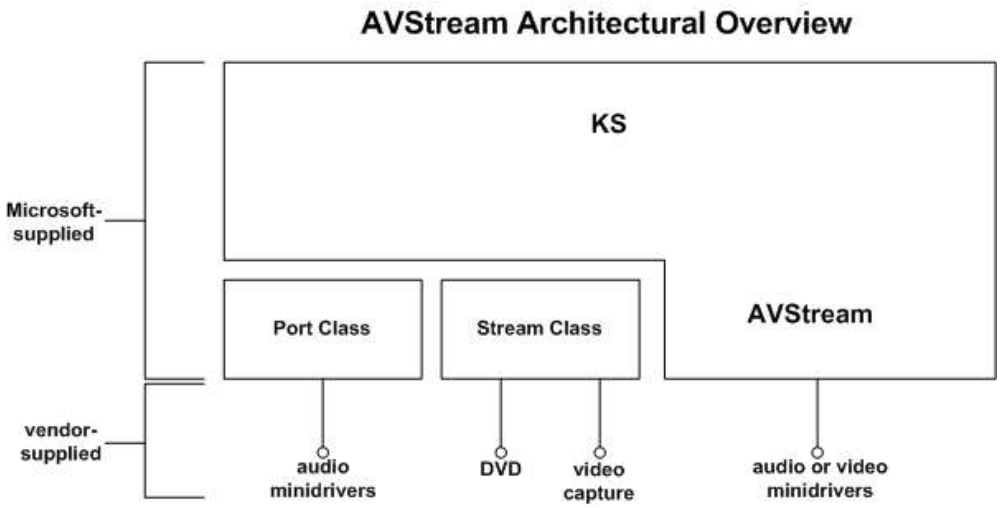


[그림 9] 장치 필터 드라이버의 동작

장치 필터 드라이버는 선택적인 드라이버로서 장치의 동작에 값을 더하거나 수정하는 역할을 하며 비장치 드라이버의 역할을 할 수도 있다. 필터 드라이버는 하나 이상의 장치를 도와줄 수 있으며, 앞에서 말한 장치 기능 드라이버를 보조하는 역할을 한다.

(3) Virtual Web Cam

앞에서 설명한 WDK를 활용하면 Web Cam Device를 생성할 수는 있지만, Web Cam에 올바른 데이터를 표현하도록 구현하기 위해서는 AVStream을 활용해야한다. AVStream은 마이크로소프트에서 제공하는 멀티미디어 클래스 드라이버이다. AVStream은 비디오 혹은 비디오/오디오 스트리밍을 제공한다. Microsoft는 AVStream을 Ks.sys를 사용하여 운영체제의 일부로 제공한다. 하드웨어 제공자는 미니드라이버를 Ks.sys 밑에서 작동하도록 제공한다. AVStream Driver Model에서 제공자는 마이크로소프트 클래스 드라이버와 상호작용하는 미니드라이버를 제공해야 한다. AVStream에 대한 Overview를 그림으로 보면 다음과 같다.



[그림 10] AVStream Overview

5) Evaluation Factors

평가요소	평가비율
Cloud Phone 가상 디바이스 생성 및 실행	30%
Cloud Phone 프레임워크 구현	20%
Cloud Phone 센서 구현	10%
Cloud Phone 카메라 메커니즘 구현	20%
통합 완성도	10%

[그림 11] Evaluation Factors

4. 개발 일정

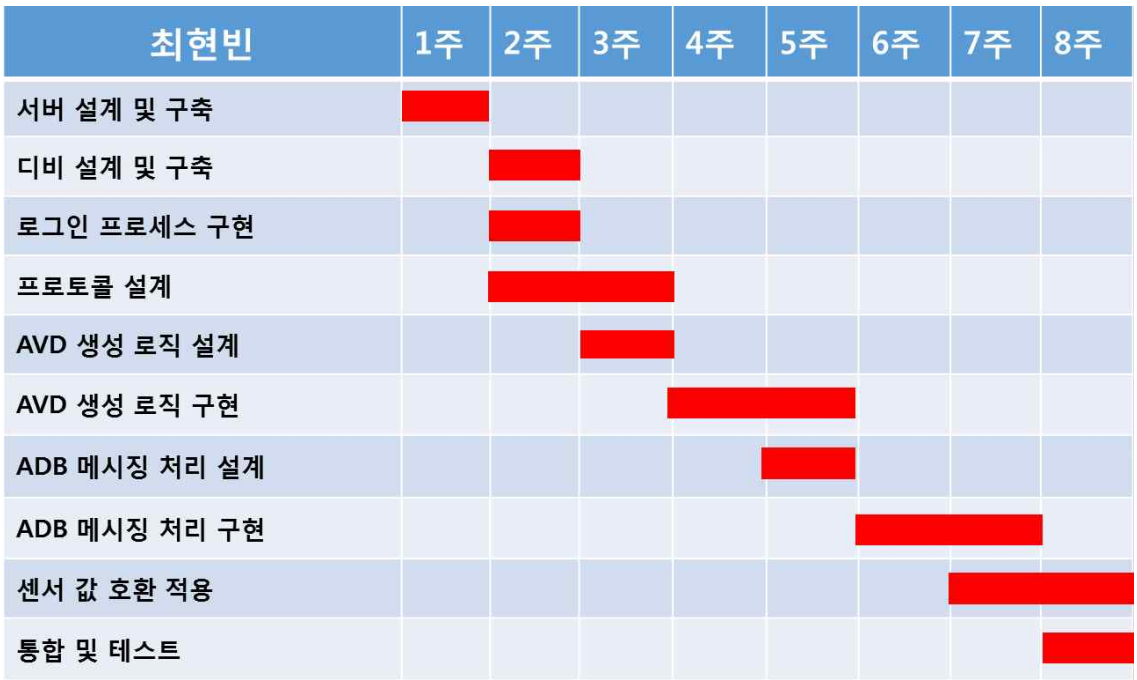
1) 윤재석

[그림 12] 윤재석 일정

2) 장정규

[그림 13] 장정규 일정

3) 최현빈



[그림 14] 최현빈 일정

5. 용어 정리

1) Internet of Things

인간과 사물, 서비스 세 가지 분산된 환경요소에 대해 인간의 명시적 개입 없이 상호 협력적으로 센싱, 네트워킹, 정보처리 등 지능적 관계를 형성하는 사물 공간 연결망

여기서의 '사물'은 유무선 네트워크에서의 end-device뿐만 아니라, 인간, 차량, 교 량, 각종 전자장비, 문화재, 자연 환경을 구성하는 물리적 사물 등이 포함됨

2) Android Virtual Device

실제 환경이 아닌 PC 환경(가상)에서 Android 폰에 있는 기능들을 쓸 수 있도록 가상화해놓은 Android SDK Tool이며, 여러 가지 OS 버전을 사용할 수 있으며 안정적으로 모든 기능을 지원한다.

3) Android Debug Bridge

안드로이드 SDK 에 포함되어 있으며, 안드로이드 개발자가 사용하거나 혹은 안드로이드를 사용한 스마트폰에 컴퓨터로 접속하여 자료를 백업하는 툴로 많이 사용된다.

명령어를 통해 직접적으로 명령을 내릴 수 있다.

(ex : adb -s <serialNumber> <command>

4) Frame Buffer

Frame Buffer는 Linux 상에서 화면으로 데이터를 출력하기 위한 일종의 추상화 드라이버이다. 안드로이드 또한 Frame Buffer 드라이버를 통해서 화면을 출력하고 시스템내 '/dev/graphics/fb0' 라는 파일로 존재 한다.

5) Windows Driver Kit

Windows를 위한 드라이버를 빌드, 테스트, 디버그 및 배포할 수 있는 도구가 포함된 프레임워크

6. 참고 문헌

안드로이드의 모든것 분석과 포팅(고현철, 유형목)

안드로이드의 모든것 NDK : C/C++을 이용한 안드로이드 앱 개발 방법(고현철, 전호철)

안드로이드 미디어 프레임워크(김태연, 이왕재, 이선진, 장우현, 박지훈)

윈도우즈 드라이버 모델 WDM : Writing Windows WDM Device Drivers(Chirs Cant/박햇님)

Filter Driver (Lambert M.Surhone)

윈도우 USB 디바이스 드라이버 : WDM부터 최신 KDMF와 UMDF까지 윈도우 드라이버 모델 입문과 실전(하마다 켄이치로)

Real MySQL : 개발자와 DBA를 위한(이성욱)

KISA Library '인터넷 & 시큐리티 이슈' Net Term 6월호 사물인터넷(Internet of Things)