SAMSUNG

# S/W멤버십 과제 보고서

과제명 : { Android Virtualization을 이용한 IoT 구현 }


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

- 작성일 : 2014. 06. 04

삼성소프트웨어멤버십
secmem.org

# S/W멤버십 과제 보고서

| 과 제 명 | Android Virtualization을 이용한 IoT 구현 | | |
|---|---|---|---|
| 과제구분 | 창의과제 | | |
| 과제기간 | 2014. 04. 01 ~ 06. 04 (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 DarkCloud 폴더를 마운트시켜 사용하는 방식을 사용했다. 그래서 '진정한 Cloud'라는 주제에 다가섰지만, 그 내용이 '파일 스토리지' 기능에만 집중되어 있다는 것에 아쉬움을 느꼈다. 이에 우리팀은 Android Virtualization을 통해 Internet of Things(IoT)를 구현하여 CloudPhone을 개발하기로 하였다. |
| 개발 환경 및 일정 | OS : Windows 7, Ubuntu 13.04<br>Tool : Android Development Kit(Eclipse), Android Debug Bridge(ADB), Android Virtual Device(AVD), MySQL, Visual Studio 2013(C, C++, C#), Windows Driver Kit(WDK) 8.1, WinDBG |
| 창의성/ 우수성 | Dark Cloud는 엄밀히 말해 파일 스토리지 서비스에 국한되어 있다.<br>하지만 Cloud Phone은 Android Virtualization을 통하여 Device에 종속적이지 않은, 진정한 Cloud Service를 이용할 수 있다. |
| 활용성/ 사업성 | 이 서비스를 통해 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 -



## - 그림 목차 -

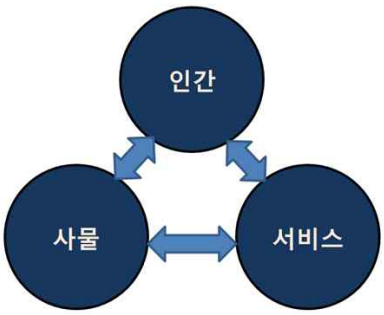



삼성소프트웨어멤버십
secmem.org

# 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)

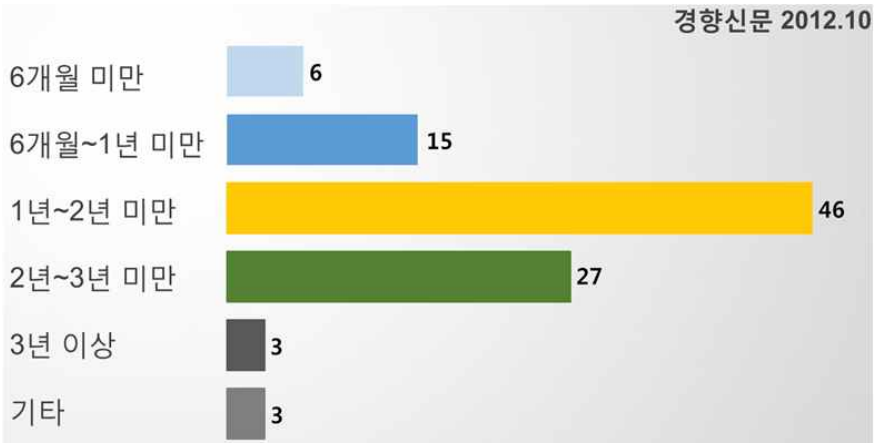
IoT(사물인터넷)가 되기 위한 가장 중요한 시스템 중 하나가 클라우드 시스템이라고 할 수 있다. 모든 사물들이 인터넷을 통해 클라우드에 연결됨으로써 사용자는 실시간으로 모든 상황을 인지할 수 있는 것이다.

이 것을 '우리가 평소 사용하는 스마트폰에 적용시켜보자' 하는 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 약정 기간이 기본 2년, 길어야 3년 밖에 하지 않는 현실에서, 실제 직장인들의 휴대폰 교체 주기는 1년 ~ 2년 미만의 짧은 기간이 대부분이었다.




[그림 2] 휴대폰 교체 주기(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, 동영상뿐만 아니라 스마트 폰 이용 환경까지도 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. 클라이언트가 서버에 접속 후 서버 컴퓨터의 데이터베이스에 저장되어 있는 클라이언트 정보를 가져오면, 서버는 그 정보를 판단하여 클라이언트 AVD를 실행, 해당하는 디바이스에 화면을 전송하게 된다.

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

### 2) Android Virtual Device Framebuffer 압축 및 전달

Android Frmawork 내에는 Framebuffer Driver가 존재한다. Framebuffer Driver는 Android 기기의 화면에 출력되는 모든 정보를 가지고 있다. 그러므로 Framebuffer를 통해서 서버에서 동작하는 AVD의 화면을 실시간으로 추출할 수 있다. 추출한 Framebuffer는 디바이스의 해상도 크기만큼의 픽셀 데이터를 가지고 있다. 그러므로 얻어진 Framebuffer의 용량이 매우 크다. 이를 해결하기 위해 손실 압축방법의 표준방식인 JPEG 알고리즘으로 압축을 하여 용량을 줄이고 각각의 Cloud Phone Device들로 전송할 것이다.

### 3) Android framework for Cloud Phone

CloudPhone을 위한 안드로이드 프레임워크란, 서버에서 동작하는 AVD에서 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 WI-FI 연결 및 계정 설정 등을 통해 CloudPhone 서버에 연결할 수 있는 환경을 제공한다. 서버와 연결되면 서버로부터 받은 Framebuffer 화면만 실시간으로 출력하는 기능을 수행한다.

### 4) CloudPhone Service

CloudPhone Service는 스마트폰의 센서 장치의 데이터를 수집 및 전송하는 기능을 수행한다. 서버에서 동작하는 AVD는 가상으로 존재하므로 클라이언트 디바이스에 장착



삼성소프트웨어멤버십
secmem.org

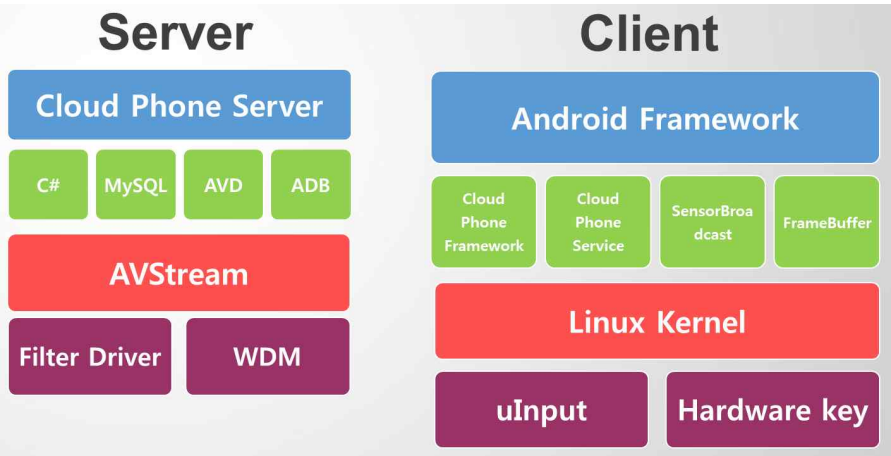
된 센서를 활용할 수 없고, Touch 정보, Hardware Key Input 정보를 AVD에 매핑시키는 작업을 해야 한다. 그러므로 스마트폰에 존재하는 센서중 GPS, 자이로센서, Battery 데이터, Touch 정보, Hardware Key Input 정보를 수집하여 Server로 전송한다. 이로써 사용자는 서버에 존재하는 가상 디바이스가 아니라 실제 자신의 휴대폰을 사용하는 듯한 착각을 하게 될 것이다.

### 5) Virtual Web Cam for Android Virtual Device

최근 Android SDK에 포함되어있는 Android Virtual Device는 Windows에 연결되어 있는 Web Cam을 Android Virtual Device의 카메라로 인식시킬 수 있는 기능을 가지고 있다. 다시 말하면 Web Cam을 통해 Android Virtual Device도 카메라를 활용한 어플리케이션을 사용할 수 있는 환경이 제공된다는 것이다. 이 점을 착안하여 실제 핸드폰의 Camera를 서버 컴퓨터의 Virtual Web Cam으로 제공해줄 수 있다면, Android Virtual Device가 Camera를 가지지 못하는 한계점을 실제 디바이스의 Camera로 해결할 수 있을 것이라고 생각하였다. 즉, CloudPhone의 카메라 기능을 구현할 수 있을 것이라고 생각하여 이 부분을 개발 목표로 선정하였다. 먼저 CloudPhone을 통해 새로운 Android Virtual Device를 할당 받게 되면 서버에 Web Cam을 연동시키도록 구현할 것이며, 후에 CloudPhone에서 카메라 기능을 실행하면 해당 폰의 카메라에서는 카메라 데이터를 생성하며, 생성한 데이터를 서버로 전달하도록 구현할 것이다. 서버에서는 전달된 데이터를 Virtual Web Cam에 출력하도록 하여 결론적으로 Android Virtual Device에 스마트 폰의 카메라를 통한 데이터가 표현될 수 있도록 구현할 것이다.

## 3. 개발 내용

### 1) System Architecture




[그림 3] System Architecture

### 2) Cloud Phone Server

#### (1) 서버 설계 및 구축, DB 연동

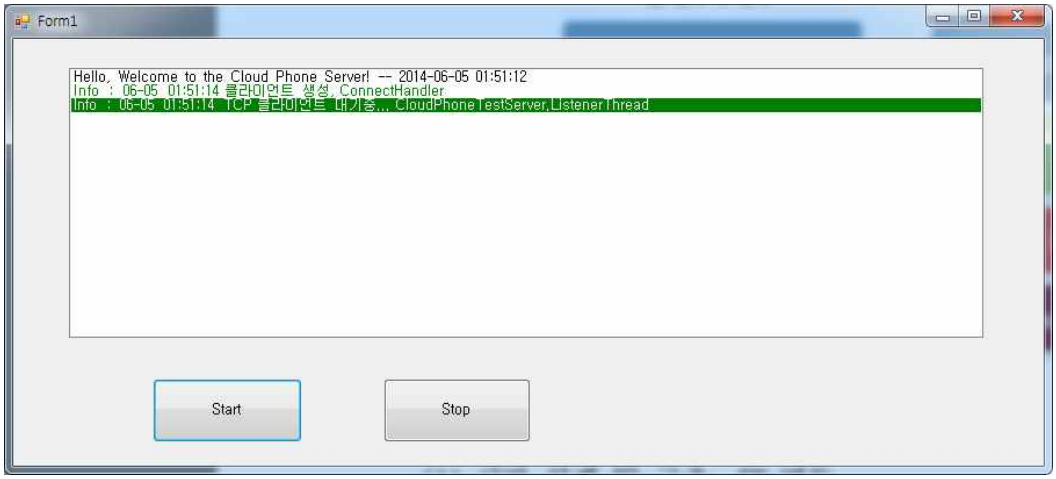
서버는 처음에 Node.JS를 사용하여 유연성있는 프로그램을 작성하려고 하였으나, AVD 및 ADB의 제어와 Android Client와의 원활한 통신을 위해서 개발 툴을



삼성소프트웨어멤버십
secmem.org

Visual Studio 2013으로 재조정하였고, 사용 언어도 C#으로 정하였다.

WinForm으로 작성하였으며, 디버깅 및 클라이언트와 주고받는 메시지를 확인하기 편하기 위해 Log 메시지 창을 띄워 출력하도록 하였다.




[그림 4] CloudPhone Server WinForm

전작인 DarkCloud에서 Document 형식으로 저장되는 MongoDB를 썼다면, 이번에는 클라이언트의 정보를 SQL로 관리할 수 있는 관계형 DB인 MySQL을 사용하였다. 이 DB에서는 클라이언트의 ID를 관리하게 된다.

CloudPhone 서버에서는 먼저 여러 대의 클라이언트를 관리하기 위한 서버 쓰레드, 그리고 각각의 클라이언트마다 연결할 수 있는 TCP 쓰레드, 배경화면을 전송하기 위한 UDP 쓰레드, 클라이언트에서 정해진 시간동안 ACK를 받지 못하면 종료시키게 하는 Timer 쓰레드 등 여러 개의 쓰레드를 종합 관리하고 있다.

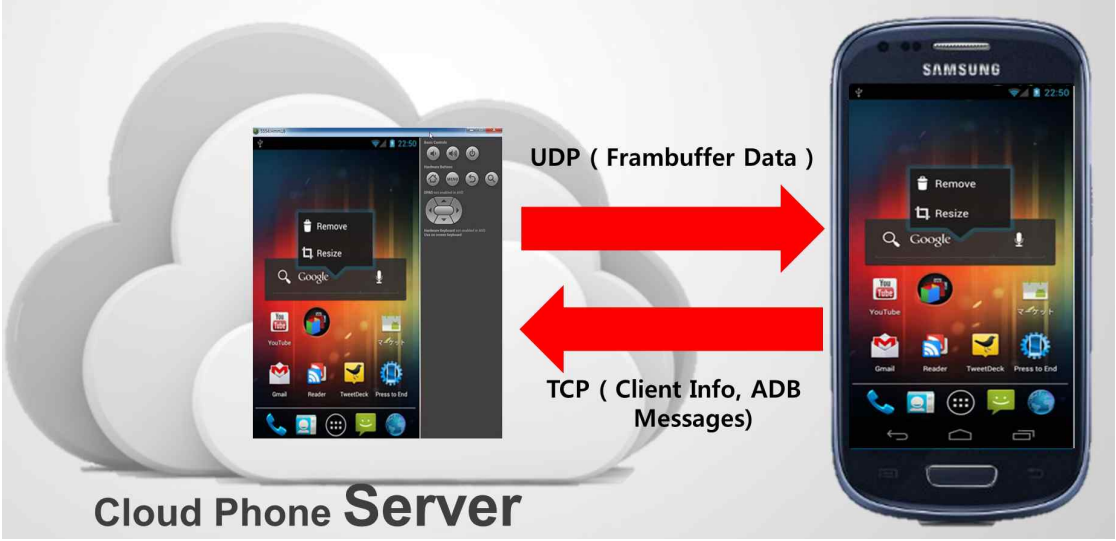
(2) 프로토콜 설계

클라이언트 디바이스와 통신하기 위해서 처음에는 TCP만을 생각했으나, 클라이언트 디바이스가 메시지를 보내서 AVD에 전달하는 통로와, AVD에서 Framebuffer를 받아와서 클라이언트 디바이스에 보내주는 통로를 분리하여 생각하였다. 따라서 하나의 TCP 연결과 하나의 UDP 소켓을 선언하게 되었다.

클라이언트와 처음부터 세션을 유지하면서 정확한 ADB 메시지를 받기 위해서 TCP를 사용한다. 또한, 클라이언트 디바이스에 최대한 빠른 속도로 화면을 전송해주기 위해서 UDP를 사용한다.



삼성소프트웨어멤버십
secmem.org




[그림 5] Server <-> Client 통신 프로토콜

통신 프로토콜 외에도, 클라이언트와 통신을 하기 위해서는 일종의 메시지 규약이 필요하다. 클라이언트 개발자와 협의를 거쳐서 TCP 통신을 할 때 주고받는 메시지 프로토콜을 아래 그림과 같이 정할 수 있었다.

1. 각종 센서값 검사 주기

   A. GPS : 10초 주기로 클라이언트에서 서버로 전송
   B. 배터리 : 배터리 상태가 바뀜과 동시에 클라이언트에서 서버로 전송
   C. Touch Event : Event 발생 시 클라이언트에서 서버로 전송
   D. Hardware Key : Event 발생 시 클라이언트에서 서버로 전송

2. 전송 방식 : String, 각각의 Value는 '/' 로 구분 , Data Field는 ','(쉼표)로 구분

| Header | ClientID | Data Field |
|---|---|---|

   A. Header : MSG Type

      i. Login MSG ( Login과 관련된 MSG )
      ii. AVD ACK MSG ( Client의 ACK를 Check하는 MSG )
      iii. ACTION MSG ( ACTION_UP, ACTION_DOWN, ACTION_MOVE )
      iv. KEYCODE MSG ( KeyCode )
      v. GPS MSG ( GPS 값 )
      vi. BATTERY MSG ( 배터리 값 )

   B. ClientID : AVD에 메시지를 전할 Client의 ID

   C. Data : Data Value , 각각의 Value는 쉼표 : ',' 로 구분

      i. LoginMSG : [ 0 / Size / ClientIP , Client_ID ]
      ii. AVD ACK MSG : [ 1 / ClientID ]
      iii. ACTION MSG : [ 2 / ClientID / ACTION_TYPE , Data, Data2, ... ]
      iv. KEYCODE MSG : [ 3 / ClientID / KEY_CODE ]
      v. GPS MSG : [ 4 / ClientID / 위도 , 경도 ]
      vi. BATTERY MSG : [ 6 / ClientID / 잔량 , 충전상태 ]

[그림 6] Server <-> Client 메시지 프로토콜



삼성소프트웨어멤버십
secmem.org

(3) ADB 메시징 처리 설계 및 구현

클라이언트 디바이스에서는 서버에 자신의 ID, Touch 정보, 하드웨어 키 입력, 배터리 상태, GPS 정보 등 여러 가지 정보를 보내야 한다. 그것을 서버에서 프로토콜을 통해 받아서 처리하게 되는데, 서버에서 띄우는 Android Virtual Device(AVD)에 적용하게 하기 위해서는 Android Debug Bridge(ADB)라는 프로그램의 도움을 받아야 한다.


[그림 7] Android Debug Bridge(ADB)

ADB를 사용함으로써, 명령 프롬프트(Command Line)을 통해 AVD와 통신하면서 간접적으로 센서 값을 적용시킬 수 있다. 예를 들어, 5554 포트를 가진 AVD에 Hardware Key 입력을 한다고 하면, 다음과 같은 명령어를 줌으로써 AVD에 직접적으로 Hardware Key를 누르게 한 것과 같은 효과를 줄 수 있다.


[그림 8] adb를 이용하여 Emulator에 명령어를 전달하는 화면

CloudPhone Server를 사용하면서 Command Prompt라는 외부 프로그램을 직접적으로 이용하는 방법을 찾아본 결과, System.Dignostics.Process 클래스를 사용하여 외부 프로그램을 직접적으로 불러오고, 또한 거기에 명령어를 입력시킬 수 있다는 사실을 알게 되었다.

따라서 서버를 실행한 후 클라이언트가 접속하게 되면 TCP 프로토콜을 통해 받은 데이터를 ADB 명령어로 변환해서 AVD에 각종 센서값(배터리, GPS, Touch, Hardware Key) 들을 적용시킬 수 있다.

## 3) Cloud Phone Framework 및 센서

(1) Android Framework for CloudPhone

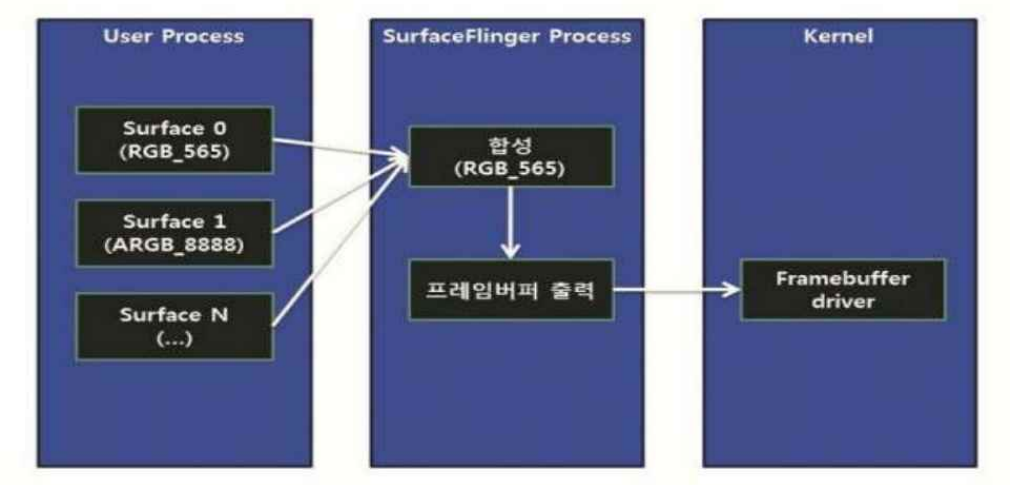
CloudPhone을 위한 안드로이드 프레임워크란 서버에서 동작하는 AVD(안드로이드 가상 디바이스)에서 동작하는 Android Framework와 실제 디바이스에서 동작할 Framework를 통칭한다. 서버용 프레임워크는 기존 안드로이드 시스템에 FrameBuffer 추출 및 전송을 담당하는 Service를 구현하고, 클라이언트용은 기존 안드로이드 시스템을 수정하여 스마트폰이 부팅이 완료된 후 동작할 때 CloudPhone 런쳐형태로 실행된다. CloudPhone서버에 연결할 수 있는 환경을 제공한다. 서버와 연결되면 서버로부터 받은 FrameBuffer 화면만 실시간으로 출력하는 기능과 Sensor 전달기능을 수행 한다.



삼성소프트웨어멤버십
secmem.org

(2) Android Virtual Device Framebuffer 압축 및 전달

Framebuffer Driver를 읽어서 AVD의 화면을 실시간으로 추출한뒤 host PC로 바이너리를 전송한다. FrameBuffer장치는 파일의 권한이 660으로 READ 권한이 없으므로 root권한이 없이 추출을 하기위해서 adb shell 명령어를 통해서 AVD가 실행되는 시점에 '/dev/graphics/fb0' Framebuffer 파일을 777로 변경해 주도록 하였다. 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 크므로 전송속도 개선을 위해 손실 압축방법인 JPEG 알고리즘으로 압축하고 TCP 통신보다 속도가 빠른 UDP 통신을 사용하여 개발을 하였다. 개발과정에서 가장 시행착오가 많았던 부분인 AVD의 퍼포먼스 속도를 개선하기 위해서 HAXM과 GPU 하드웨어 자원을 사용하려 시도 하였으나 현재 안드로이드에서 버그가 존재하여 GPU자원을 활용할 수가 없었다.




[그림 9] Android 화면이 Framebuffer에 저장되는 과정

(3) Sensor Mapping

CloudPhone 서비스는 Android 시스템이 부팅완료 시점부터 런쳐형태로 동작하고, 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, 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. 다만 AVD에서 자이로스코프 센서는 장치자체가 존재하지 않아 매핑을 시켜도 반응이 일어나지 않았다. 서비스에서 수집하는 정보는 아래 표로 정리되어 있다.

| | | |
|---|---|---|
| GPS | 위도 | 매핑완료 |
| | 경도 | |



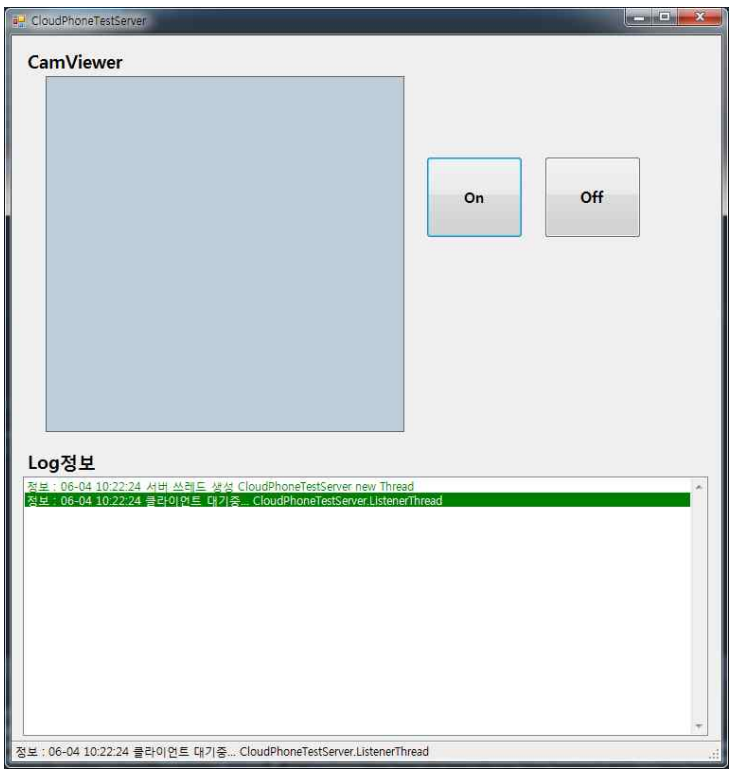
삼성소프트웨어멤버십
secmem.org

| | | |
|---|---|---|
| Battery | 잔량 | 매핑완료 |
| | 충전 상태 | |
| Touch Event | ACTION_DOWN, X, Y | 매핑완료 |
| | ACTION_MOVE, X, Y | |
| | ACTION_UP | |
| | ACTION_POINTER_DOWN | 멀티터치<br>Emulator 미지원 |
| | ACTION_POINTER_UP | |
| Hardware Key Input | KEYCODE_HOME | 매핑완료 |
| | KEYCODE_VOLUME_DOWN | |
| | KEYCODE_VOLUME_UP | |
| | KEYCODE_BACK | |
| | KEYCODE_MENU | 매핑완료 |
| Gyroscope | 각각도 | Emulator 미지원 |
| | 가속도 | |

### 4) Virtual Web Cam for Android Virtual Device

#### (1) 테스트 서버 & 클라이언트 구현

가장 먼저 Virtual Web Cam을 구현하기 위하여 테스트를 위한 서버를 C#으로 구현하였다. 테스트 서버는 C#, 윈폼으로 작성되었으며 UDP통신, 통신을 통한 이미지 Viewing, 서버 Log정보 출력의 기능을 가지고 있다.




[그림 10] C# Test Server



삼성소프트웨어멤버십
secmem.org

좌측의 네모 영역은 CamViewer영역으로 클라이언트로부터 받아온 이미지를 예시로 뿌려 주는 역할을 한다. 개발 초기에는 RGB형태가 아닌 Jpeg형태로 이미지 스트림을 받아서 처리하였기 때문에, CamViewer를 활용하였으나 RGB형태로 바꾼 뒤에는 특별히 활용하지 않고 이후 작업을 진행하였다. 하단은 서버 Log 정보를 출력해주는 영역으로 서버의 상태 혹은 클라이언트의 접근 관련 Log를 확인할 수 있도록 구현하였다.

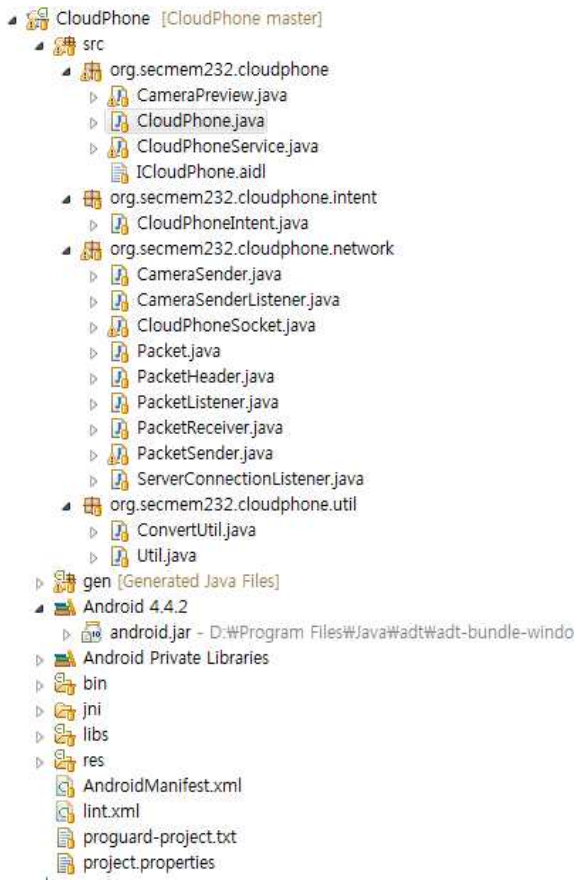
테스트 클라이언트는 C#, 안드로이드 두가지 형태로 구현하였다. C# 형태의 테스트 클라이언트는 로컬에 있는 이미지를 이미지 스트림 형태로 변환하여, 테스트 서버에 전달하도록 구현하였다. 안드로이드 테스트 클라이언트는 서비스 형태의 어플리케이션으로 GPS Sensor가 활성화되면 카메라의 기능을 활용하여 Preview화면을 서버로 이미지 스트림 형태로 전달할 수 있도록 구현하였다. 이 부분에 대해서는 아래에 더 자세히 언급하도록 하겠다.

(2) Android Camera Preview Service

안드로이드로 만든 테스트 클라이언트를 확장하여, 안드로이드 서비스 어플리케이션을 개발하였다. 안드로이드 서비스 어플리케이션은 특별한 UI를 가지지 않으며, 백그라운드에서 대기하다가 Camera 데이터를 서버로 전송해야할 때(CloudPhone 서버에서 카메라 관련 앱을 실행하였을 때) UDP를 통해 서버로 이미지를 전송하도록 구현하였다. 카메라에서 촬영하는 Preview 이미지 형태를 받기 위해서 Camera.Preview.Callback 인터페이스의 onPreview 콜백 함수를 오버라이딩하여 구현하였다. onPreview 함수가 리턴하는 Preview 이미지의 포맷은 기본적으로 YUV 형태를 갖는데, RGB에 비해 상대적으로 사이즈가 크므로, YUV 포맷을 RGB 형태로 변환해주는 함수를 구현하였다. 단순 사칙연산 단위의 수치계산이 많으므로 JNI를 통한 Native로 구현을 하였다. 또한 이미지를 보낼 때에 일정 크기의 버퍼(이 프로젝트의 버퍼 크기는 40960 byte이다.) 크기 단위로 쪼개서 보낼 수 있도록 구현하였다. Android Camera Preview Service는 단순 UDP 방식의 서버가 아닌 구현은 복잡하지만 활용가능성이 높을 수 있도록 기능 단위로 클래스를 분할 설계하여 구현하였다.



삼성소프트웨어멤버십
secmem.org




[그림 11] Android Camera Preview Service 소스트리

(3) 서버 & Web Cam Device Driver 연동을 위한 IOCTL 구현

기본적으로 운영체제는 두 개의 계층 사용자 공간과 커널 공간으로 나뉜다. 문서 편집기, 곰플레이어와 같은 응용 프로그램 코드는 사용자 공간에 상주하는 반면 네트워크 스택, 디스크 드라이버와 같은 운영체제의 기반이 되는 코드는 커널에 상주한다. 커널 코드는 중요하고 민감한 리소스를 관리하며 응용 프로그램 사이의 보안과 신뢰 장벽을 제공한다. 이러한 까닭에 사용자 모드 응용 프로그램은 CPU에 의해 커널 리소스를 직접 접근하는 것을 막는다.

사용자 공간 응용프로그램에서 커널 공간에 있는 디바이스에 접근할 수 있도록 하기위해서 사용하는 것이 IOCTL 인터페이스 이며, IOCTL 인터페이스를 통해 다양한 디바이스의 특정한 요청의 방향을 바꿀 수 있다. 그러므로 커널은 관리가 불가능한 시스템 콜 테이블을 만들지 않고도 유동적으로 디바이스에 대한 콜을 처리할 수 있다.

CloudPhone 서버는 C#으로 구현되어 있는데 C#에서는 IOCTL을 사용할 수 없다는 한계점을 가지고 있었다. 따라서 IOCTL 기능을 하는 C/C++ 모듈을 Dynamic Linking Library 형태로 구현하여, C#에서 참조할 수 있도록 구현하였다.



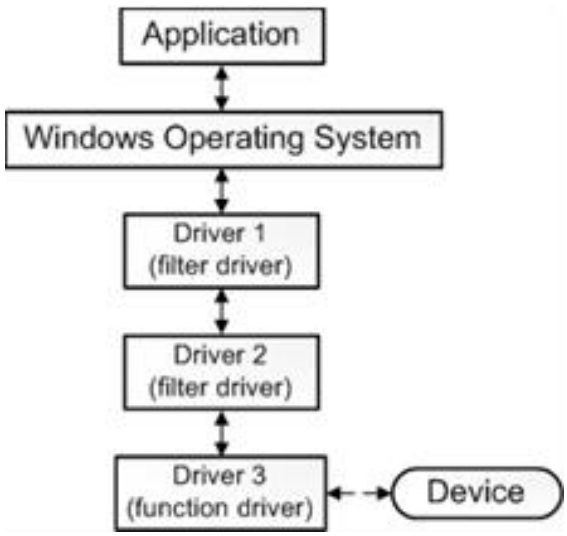
삼성소프트웨어멤버십
secmem.org

```
extern "C" __declspec(dllexport) int DisplayWebCam(BYTE *buf, int length)
{
    WCHAR DeviceLink[] = L"\\\\.\\cloudphone";
    HANDLE hdevice = CreateFileW(
        DeviceLink,
        GENERIC_READ | GENERIC_WRITE,
        0,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_OFFLINE,
        NULL
        );
    if (hdevice == INVALID_HANDLE_VALUE)
    {
        printf("Unable to open UsbcameraFilter device - error %d\n",
            GetLastError());
        return 1;
    }

    DWORD dwRet;
    if (!DeviceIoControl(hdevice, IOCTL_IMAGE, buf, sizeof(buf), 0, 0, &dwRet, NULL))
    {
        printf("DeviceIOControl Fail!! \n");
        _getch();
        CloseHandle(hdevice);
        return 2;
    }

    CloseHandle(hdevice);

    return 0;
}
```

[그림 12] IOCTL 기능을 위한 DLL 작성

(4) Windows Driver Kit 활용

Virtual Web Cam을 구현하기 위해서 Microsoft에서 제공하는 Windows Driver Kit 8.1을 활용하여 구현하였다. Windows Driver Kit(이하 WDK)란 Windows를 위한 드라이버를 빌드, 테스트, 디버그 및 배포할 수 있는 도구가 포함된 프레임워크를 말한다. WDK를 활용하여 드라이버를 제작하기 위해서는 Driver Model을 선택해야 한다. 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 활용하여 구현하였다.



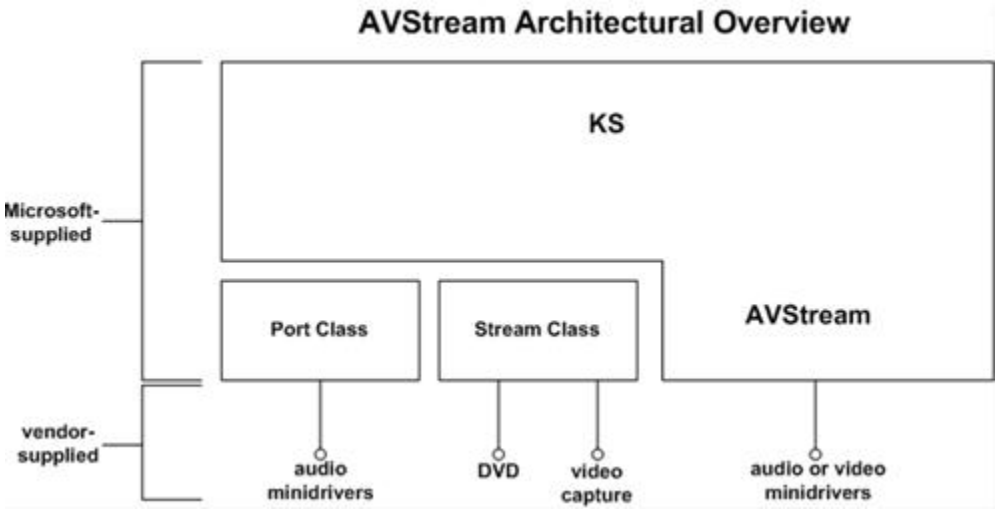

[그림 13] 장치 필터 드라이버의 동작



삼성소프트웨어멤버십
secmem.org

장치 필터 드라이버는 선택적인 드라이버로서 장치의 동작에 값을 더하거나 수정하는 역할을 하며 비장치 드라이버의 역할을 할 수 있다. 필터 드라이버는 하나 이상의 장치를 도울 수 있으며, 앞에서 말한 장치 기능 드라이버를 보조하는 역할을 한다. 이번 프로젝트에서 장치 기능 드라이버에 IOCTL을 통해 받은 이미지 스트림 데이터를 가상 웹캠에 전송하도록 구현하였다.

앞에서 설명한 WDK를 활용하면 Web Cam Device를 생성할 수는 있지만, Web Cam에 올바른 데이터를 표현하도록 구현하기 위해서는 AVStream을 활용해야한다. AVStream은 Microsoft에서 제공하는 Multimedia Class Driver이다. AVStream은 비디오 혹은 비디오/오디오 스트리밍을 제공한다. Microsoft는 AVStream을 Ks.sys를 사용하여 운영체제의 일부로 제공한다. 하드웨어 제공자는 미니 드라이버를 Ks.sys 밑에서 작동하도록 제공한다. AVStream Driver Model에서 제공자는 Microsoft Class Driver와 상호작용하는 미니 드라이버를 제공해야 한다. AVStream에 대한 Overview를 그림으로 보면 다음과 같다.




[그림 14] AVStream Overview

(5) Virtual Web Cam Device Driver 세부 구현

Device 클래스는 디바이스와 관련된 함수들의 집합으로 대표적으로 DriverEntry, MyCamDeviceControl, CaptureDeviceDispatch, PnpStart 함수가 있다.

DriverEntry 함수는 드라이버의 진입점이며 시스템에 설치될 때 레지스트리에 등록되는 정보들을 초기화하며 시스템에 설치된 이후 IOCTL 혹은 CREATE 등의 인터럽트를 핸들링 하기 위한 핸들러를 등록할 수 있다. 또한 AVStream 초기화 함수인 KsInitializerDriver를 호출하여, WDM 드라이버 객체를 생성하고 가상 Hardware를 준비하는 역할을 하는 함수이다.

MyCamDeviceControl는 DeviceIoControl 함수를 통해 사용자 공간에서 디바이스를 접근할 때 호출되는 핸들러로 CloudPhone 프로젝트에서는 IOCTL을 통해 받은 이미지 스트림 데이터를 Web Cam 디바이스에 뿌리는 역할을 한다. Web Cam 디바이스에 이미지 스트림을 뿌리기 위해 전역변수인 imgBuf를 활용하였으며, 커널 레



삼성소프트웨어멤버십
secmem.org

벨에서 값을 전달하기 위해서는 RtlCopyBytes 매크로를 활용하여 메모리에 값을 전달하도록 구현하였다. 받은 데이터를 최종적으로 가상 하드웨어 장치가 인터럽트를 발생시켜서 imgBuf 변수에 들어있는 값으로 화면에 출력하도록 구현하였으나 정확한 이유는 모르겠으나 정상적으로 작동하지 않는다.

CaptureDeviceDispatch 테이블은 PnpStart, PnpEnd 와 같이 플러그앤 플레이를 통해 호출될 수 있는 핸들러 함수들을 등록해놓는 테이블이다.

# 4. 개발 일정

## 1) 윤재석


[그림 15] 윤재석 일정

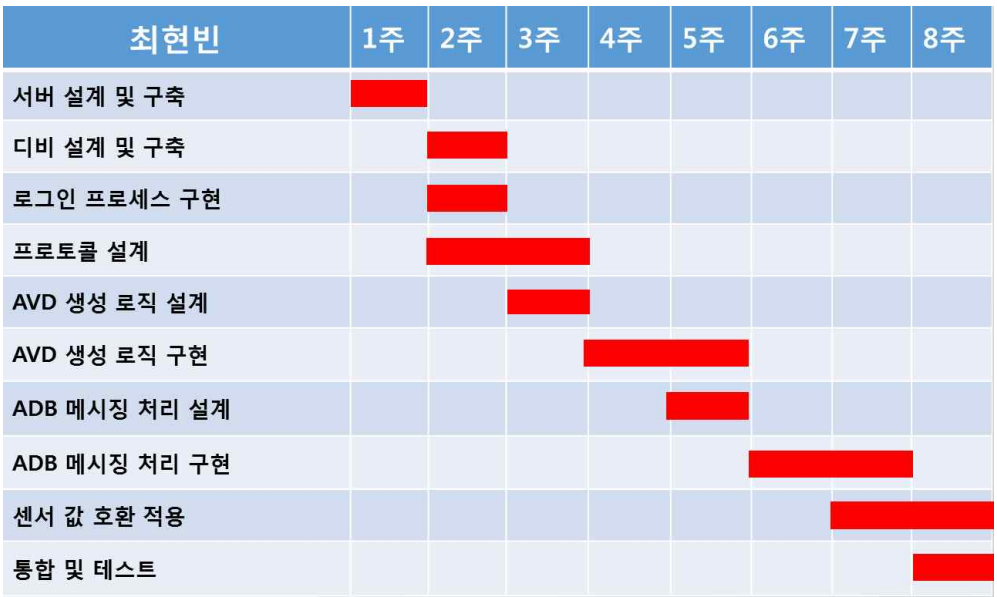
## 2) 장정규



삼성소프트웨어멤버십
secmem.org


[그림 16] 장정규 일정

### 3) 최현빈




[그림 17] 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



삼성소프트웨어멤버십
secmem.org

### 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) Framebuffer

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)





삼성소프트웨어멤버십
secmem.org