**S/W멤버십 완료 보고서**

**과제명 : Android Framework Smart Voice, JellyBoys.**

* 소 속 : 대구 소프트웨어 멤버십
* 작성자 : 노진석, 김우현, 문용호
* 작성일 : 2015. 3. 01

**S/W멤버십 완료 보고서**

|  |  |  |  |
| --- | --- | --- | --- |
| **과 제 명** | Android Framework Smart Voice | | |
| **과제구분** | 창의과제 | | |
| **과제기간** | 2014. 11. 01 ~ 02. 28 (4개월) | | |
| **지 역** | 대구 멤버십 | **참여인원** | 3명 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **회 원 명** | **학교** | **학과** | **학년** | **연락처** | **E-mail** |
| 노진석 | 금오공과대학교 | 컴퓨터소프트웨어공학과 | 4 | 010-9183-8016 | sinhaw8016@nate.com |
| 김우현 | 금오공과대학교 | 컴퓨터소프트웨어공학과 | 4 | 010-9999-7291 | battlexnwl@naver.com |
| 문용호 | 금오공과대학교 | 컴퓨터소프트웨어공학과 | 4 | 010-3742-7543 | moonyh7543@gmail.com |

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **내용** | | |
| **개발 목적**  **및 동기** | 사용자 인터페이스의 궁극적인 목표는 사용하기 편리함이다. 일상 생활에서 사람들이 쉽게 어떤 대상과 소통하는 도구로 음성이 있다. 따라서 우리는 개발자들에게 이러한 편리한 음성을 이용하여 쉽게 사물을 조작할 수 있는 인터페이스를 제공하는 프레임워크를 개발한다. 그리고 이렇게 음성을 이용한 서비스 컨텐츠가 지속적으로 확장될 수 있도록 플랫폼 환경을 구축한다. | | |
| **개발 환경**  **및 일정** | - 개발환경  OS : Linux, Windows 7, Android  TOOL : Git, Eclipse, make  Language : Java  - 개발일정  2014년 11월 01일 ~ 2015년 02월 28일 | | |
| **창의성/**  **우수성** | 사용자에게 음성 프레임워크를 탑재한 기기를 제어하거나 기기가 스스로 서비스 할 수 있도록 하여 편의성을 제공한다. 그리고 3rd Party들에게 음성 개발 플랫폼 환경을 제공함으로써 많은 다양한 서비스들이 지속 양산될 수 있도록 한다. | | |
| **활용성/**  **사업성** | 스마트 홈 시대의 도래에 따라, 미래의 새로운 기기가 개발되더라도 우리가 개발한 프레임워크가 플랫폼이 탑재된다면 음성만으로 그 기기를 제어할 수 있게 된다. 또한 그 기기에 맞는 새로운 음성 제어 및 서비스를 개발자가 쉽게 제작할 수 있다. | | |
| **지원부서** | S/W멤버십 | **기술 지원 연구원** | 임충렬 |
| **전화번호** | 010-5140-6765 | **E-Mail** | cdark.lim@samsung.com |

**목 차**

**- 내용 목차 -**

[1. 개발 목적 5](#_Toc413872609)

[2. 개발 목표 5](#_Toc413872610)

[3. 개발 내용 6](#_Toc413872611)

[1) Framework System Service 6](#_Toc413872612)

[2) 음성 서비스 프레임워크 7](#_Toc413872613)

[3) 형태소 분석 NIO Server 8](#_Toc413872614)

[4) 데이터 베이스 설계 및 구축 10](#_Toc413872615)

[5) 3rd Party & Framework Test Android Voice Service 12](#_Toc413872616)

[6) User Interface 12](#_Toc413872617)

[7) Android PDK 14](#_Toc413872618)

[8) Android Custom Platform Porting 15](#_Toc413872619)

[4.개발 일정 16](#_Toc413872620)

[5. 용어 정리 16](#_Toc413872621)

[1)aidl 16](#_Toc413872622)

[2)RPC 16](#_Toc413872623)

[3)Binder IPC 16](#_Toc413872624)

[4)Java System Service 16](#_Toc413872625)

[5)Java NIO 16](#_Toc413872626)

[6)PDK 16](#_Toc413872627)

[6. 참고 문헌 17](#_Toc413872628)

**- 그림 목차-**

[Figure 1 시스템 아키텍쳐 5](#_Toc413920307)

[Figure 2 소프트웨어 구조 6](#_Toc413920308)

[Figure 3 Java System Service 6](#_Toc413920309)

[Figure 4 Client - Server 8](#_Toc413920310)

[Figure 5 Nio Server Structure 9](#_Toc413920311)

[Figure 6 Client – Nio Server Process 10](#_Toc413920312)

[Figure 7 SmartVoice Service Database Structure 10](#_Toc413920313)

[Figure 8 Database Management Class Structure 11](#_Toc413920314)

[Figure 9 3rd Party 12](#_Toc413920315)

[Figure 10 차례로 Profile UI, Main UI, Settings UI 14](#_Toc413920316)

[Figure 11 Java System Service 14](#_Toc413920317)

[Figure 12 프로그레시브 다운로드와 스트리밍 차이 15](#_Toc413920318)

[Figure 13 개발 일정 16](#_Toc413920319)

# 1. 개발 목적

이번 프로젝트는 실시간 음성 처리를 통해 사용자의 음성을 통해 사물을 제어할 수 있도록 Android OS 플랫폼을 수정 및 음성 제어 프레임워크를 추가하여 개발하는 프로젝트이다. 우리는 Android가 탑재된 Smart Phone 기기를 Target Device로 한다.

Android 플랫폼에서 음성 기반 사용자 인터페이스를 구축하여 사용자의 음성만으로 스마트 폰의 어플리케이션 실행, 전화, 검색, 알람 설정 등 기본적인 조작이 가능하도록 한다. 또, 이러한 제한된 기능 외에도 3rd Party 개발자들이 쉽게 특정 음성에 대한 서비스를 개발할 수 있도록 오픈 플랫폼 환경을 제공한다.

# 2. 개발 목표

- Android OS의 아키텍쳐를 분석하여 개발에 필요한 새로운 시스템을 추가한다.

- 음성 처리, 자연어 처리를 통해 보다 자연스럽고, 정확성 높은 인터페이스를 구축한다.

- 음성 인터페이스를 통해 사용자가 다양한 기능을 활용할 수 있도록 한다.

- 3rd Party 및 Framework 개발자들이 쉽게 음성 서비스를 추가 할 수 있는 플랫폼을 구축한다.

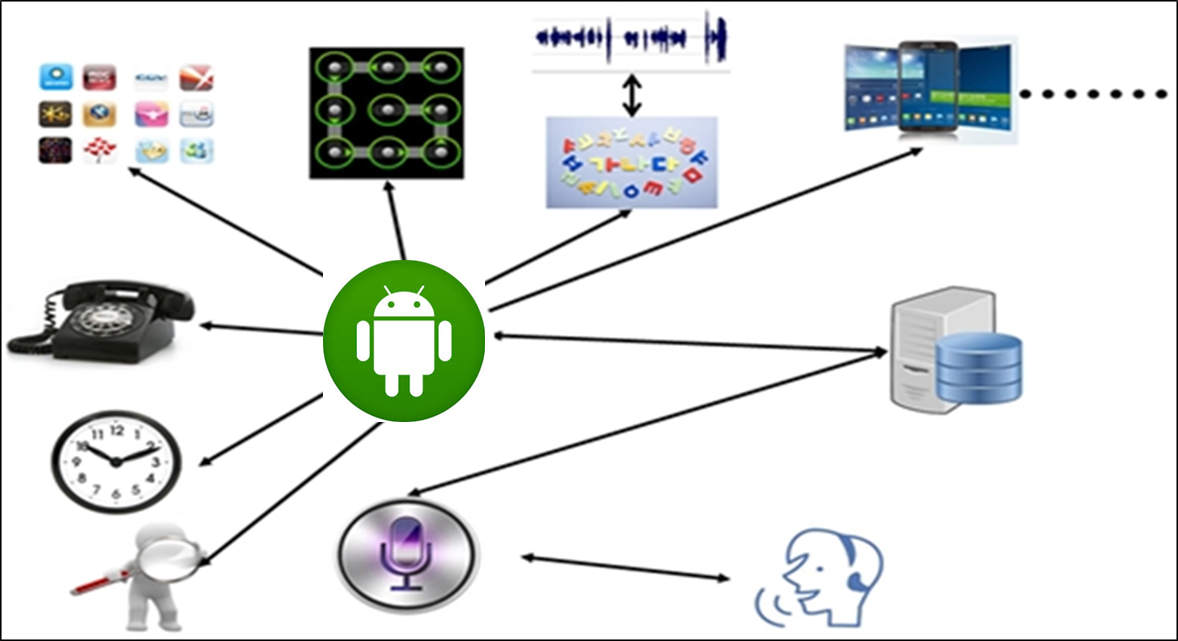


Figure 시스템 아키텍쳐

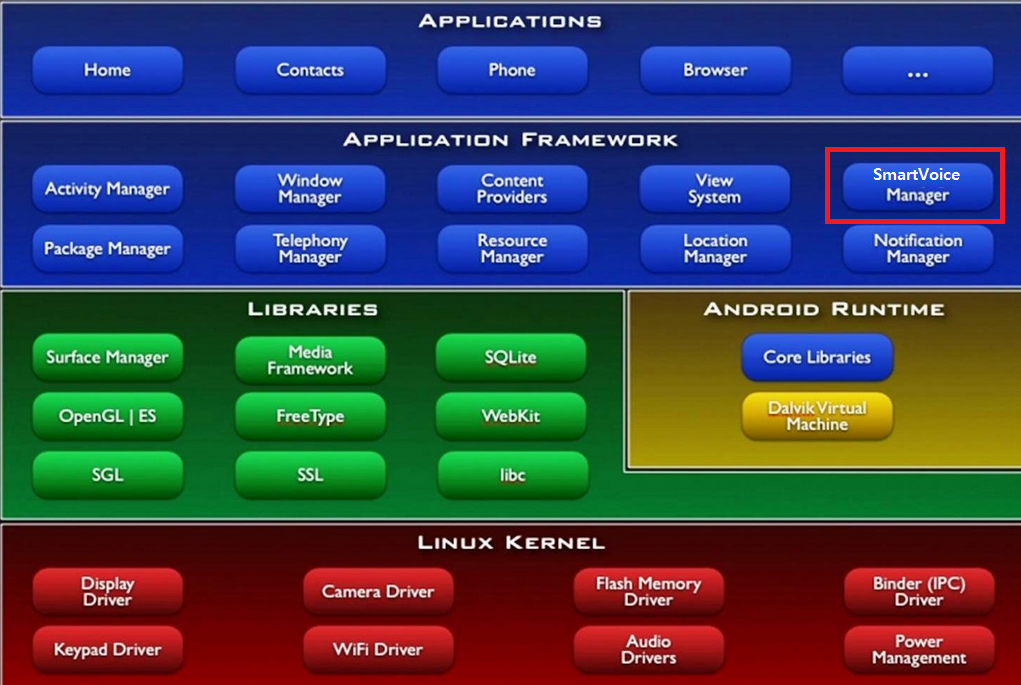


Figure 소프트웨어 구조

# 3. 개발 내용

## 1) Framework System Service

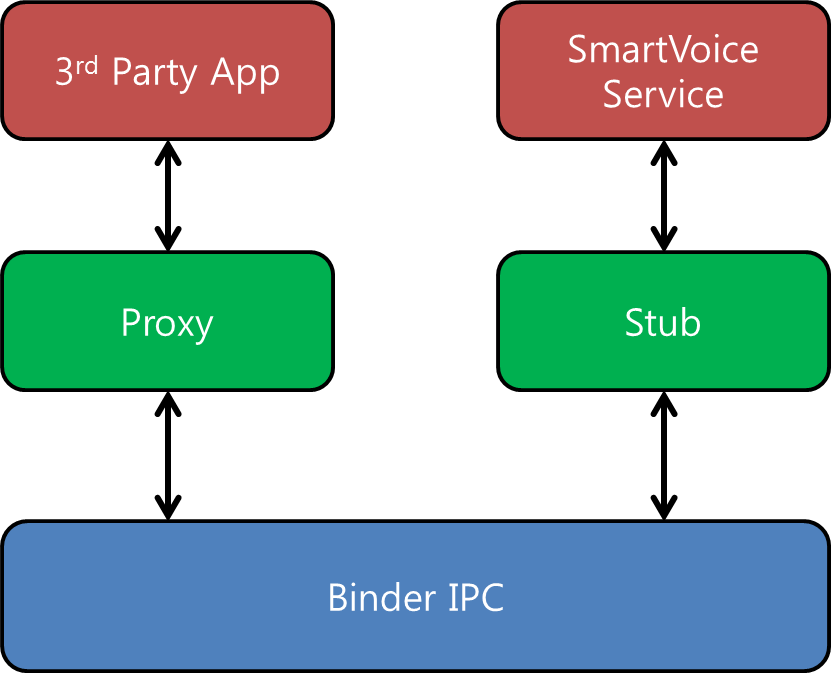


Figure Java System Service

안드로이드 aidl을 이용해 3rd Party가 새로운 서비스를 개발할 때 사용하는 인터페이스를 제공한다. 실제로 3rd Party 개발자는 프레임워크가 제공하는 getSystemService를 이용하여 음성 개발에 필요한 매니저 객체를 얻을 수 있다. 그리고 이렇게 얻어진 매니저를 통해 3rd Party가 새롭게 만든 서비스를 등록할 수 있다.

## 2) 음성 서비스 프레임워크

2.1) Smart Voice Framework Sequence

SmartVoice System의 개념도는 다음과 같다.



가장 먼저 System에서 음성탐지를 통해 결과 데이터를 전달 받으면, 해당 데이터를 기반으로 자연어 처리를 한다. 이 후 해당 자연어에 대한 처리 결과에 대응하는 서비스를 수행하는 것을 목적으로 하고 있다.

2.2) 자연어 처리 정책

먼저 자연어 처리를 애플리케이션 개발자나 프레임워크 개발자가 자연어 처리 정책을 그들 스스로 선택할 수 있도록 하기 위해서 배려하였다.

개발자의 자연어 처리 정책을 구현하는 방법은 크게 세 가지로 나뉜다. 가장 먼저 개발자가 직접 자연어를 처리를 원하는 경우(LOCAL\_PROCESS), 그리고 개발자가 형태소 분석기 서버를 거쳐 가공된 데이터를 활용하는 경우(SERVER\_PROCESS), 마지막으로 개발자가 프레임워크 개발사의 서버와 연동하여, 서버에서 미리 준비된 Proxy 서비스를 이용하는 경우(SERVER\_PROCESS | GET\_PROXY\_DATA)이다. 따라서 개발자는 자신이 원하는 옵션에 따라 자연어를 자유롭게 처리할 수 있다. 만약 프레임워크가 자사에서 개발되었으며 자연어 처리를 담당하는 자사의 서버를 연동한 경우, 실제 사용자들이 어떤 기능을 사물로부터 요구하는지 서버를 통해 Data Mining 할 수 있으므로, 유지보수와 확장에 매우 유리하게 될 것이다.

2.3) 개발자의 서비스 구현

자연어 처리 결과 자신이 수행하고자 하는 서비스를 개발자 스스로 자유롭게 구현할 수 있다. 3rdParty/Framework 개발자는, 프레임워크에서 제공하는“android.app.BaseVoiceService”를 import하고 상속함으로써 서비스 확장이 가능하다. 개발자는 자신이 음성 프레임워크를 통해 서비스 하고자 하는 알고리즘을 추상함수인 serviceProcess()에 구현하는 것으로 간단하게 서비스를 제공할 수 있다. 단, serviceProcess() Callback 함수가 수행되기 위해서는 자신의 서비스를 호출하기 위한 matchString을 세팅하여야 하며 사용자는 가장 적합한 문자열을 추가하면 된다. 모든 것을 작성하였으면, 개발자는 getSystemService()를 통해 음성 프레임워크 매니저에게 자신이 개발한 서비스를 설치하도록 하면 모든 과정이 종료된다. 그러면 음성 프레임워크는 내부적으로 자연어를 수신하고 개발자가 신규 추가한 서비스를 호출하기 위해서 최소한의 자연어 처리를 수행 후, 등록된 서비스 중 적합한 서비스를 호출하게 된다.

2.4) 사용자 편의성

음성 처리 프레임워크에 추가된 서비스들은 음성 프레임워크의 환경 설정을 담당하고 있는 System application에 자동으로 등록된다. 이 곳에서 사용자는 자신의 프레임워크에 내장되어있거나 혹은 직접 설치한 음성 서비스를 확인할 수 있다. 그리고 해당 서비스에 대한 사용 여부를 체크할 수 있다.(아래 6)번 설명 참조)

## 3) 형태소 분석 NIO Server

3.1) Non-Blocking Input-Output Server

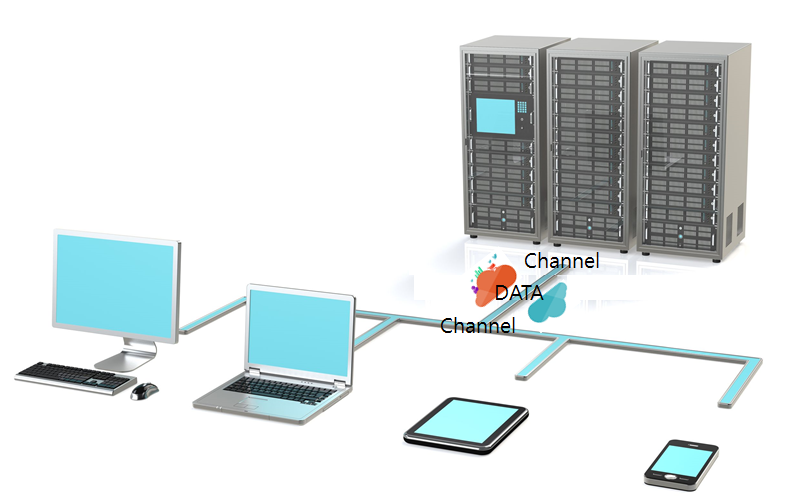


Figure Client - Server

형태소 분석 NIO Server는 위의 그림에서 보는것과 같이 다중의 Client 와 Server 사이의 Channel 을 통해 Data가 비동기 방식으로 처리하는 방식으로 작동된다.  
 서버의 구조는 아래 그림 5 에 해당되는 Nio Server Structure 와 같이 구성되는데, Server는 크게 DatasetManager 라는 Client로부터 받은 Data를 처리해주는 Manager와 NounExtractor 라는 자연어를 처리 해주는 클래스를 가진다. NounExtractor는 기본적으로 Android SmartPhone 에서 Voice Recognize Engine 을 통해 입력된 사용자의 가공되지 않은 음성 데이터를 받아와 띄어쓰기와 보다 정밀한 목적어, 동사를 찾아내는 작업을 하는데 서버 내 자체적인 알고리즘과 User Dictionary를 활용하여 처리한다.

DatasetManager는 Server에서 처리 가능한 Proxy Funtion Class를 가지고 있는데, 이는 Client의 Phone Local 상에서 처리되어야 하는 작업들의 부하를 줄여주고, Server에서 선처리해 둔 자료 혹은 특정 작업을 통해 Client의 요구에 맞게 행동해주는 클래스이다.



Figure Nio Server Structure

3.2) Client – Server 처리 과정

그림 6와 같이 Client와 Server 사이에는 Channel 이라는 것을 통해 Data가 Send/Receive 된다. Data는 Client와 Server 사이에 맺어진 규약이 있는데, 이는 Data의 Header에 특정 Option을 덧붙여 주고받는 방식으로 이루어진다. Client으로부터 받은 해당 Data의 Option을 Server에서 확인하고, 해당 Option에 맞는 Proxy Function을 호출하여, Data를 Client의 요구에 맞게 가공하여 가공된 데이터를 다시 Client로 보내주는 방식을 취한다.

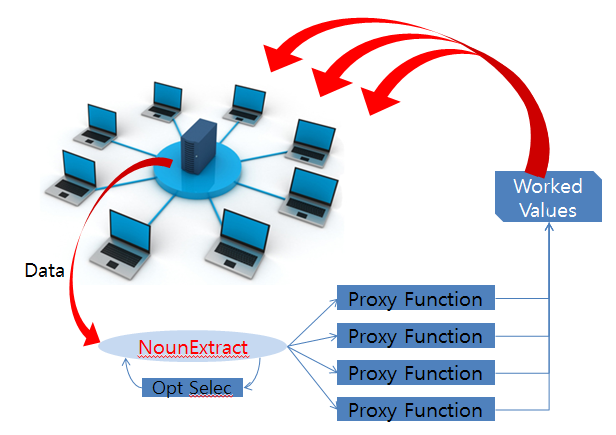


Figure Client – Nio Server Process

## 4) 데이터 베이스 설계 및 구축

4.1) SmartVoice Service Database Structure

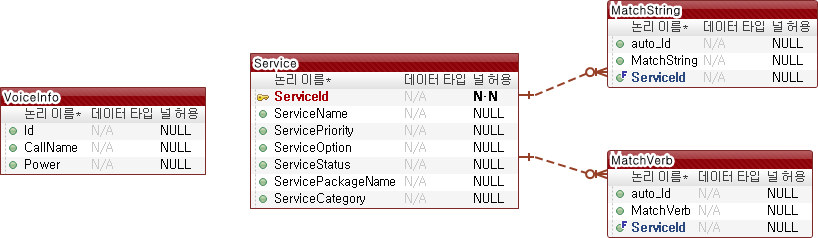


Figure SmartVoice Service Database Structure

Smartvoice Service Database 는 크게 4개의 Table   
 1) VoiceInfo : SmartVoice Service를 대표하는 CallName(명칭), Power(전원상태)로 구성  
 2) Service : SmartVoice Service에 포함되는 데이터 값을 나타내고, 그림과 같이 서비스

고유의 Id, 이름, 우선순위, 옵션, 각 Service에 해당하는 전원상태, 포함되는

패키지명, 종류로 구성  
 3) MatchString/MatchVerb : MatchString와 MatchVerb는 Service Table로부터 고유한

ServiceId에 해당되는 서비스 매치키워드와, 매치동사로 구성

로 구성된다.

Database는 Android Platform에 적합한 SQLite를 사용하였으며, SQLite에서는 Foreign Key가 제공되지 않는 점을 감안하여, Trigger로 MatchString / MatchVerb Table을 Service Table과 연관시켜 설계되어있다.

4.2) Data Management

SmartVoice Service 내에 Database를 관리해주는 Package에는 그림8 과 같은 Database Management Class Structure를 볼 수 있다. 해당 구조는 크게 4개의 Class로 구성되는데  
 1) VoiceServiceDAO : Data Access Object로 DB와, Service 객체를 가지고있는 Bean에

접근하는 Class

2) VoiceServiceBean : Service에 해당되는 Data를 가지고 있고, Get / Set 해주는 Class

3) MySQLiteHandler : Database 접근(Load, Save, Update, Insert, Delete)을 Handle 하는

Class

4) MySQLiteOpenHelper : Database를 Create 혹은 Open 을 담당하는 Class

로 구성되고, 전체적인 데이터는 DAO에서 모두 관리하는 구조로 구성되어있다.



Figure Database Management Class Structure

## 5) 3rd Party & Framework Test Android Voice Service



Figure 3rd Party

5.1) Application Execute

패키지 매니저를 통해서 현재 단말에 설치되어 있는 어플리케이션 목록과 사용자가 실행하기 원하는 어플리케이션 이름을 비교한 후 일치하는 어플리케이션을 실행해준다.

5.2) Phone Call

안드로이드 연락처를 불러와 사용자가 연락을 원하는 이름과 가장 유사한 사람에게 전화를 걸어준다.

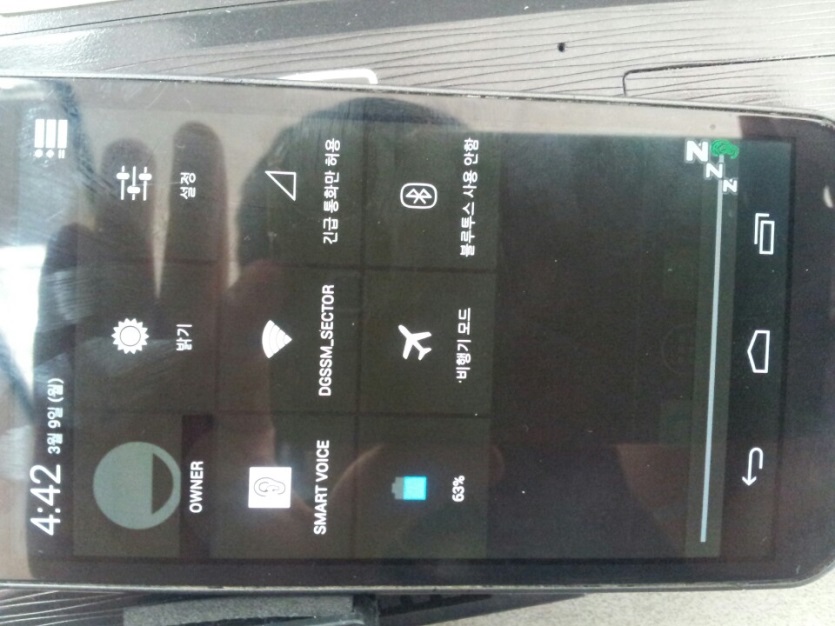
5.3) 심심이

실제 상용화 되어 있는 심심이 서비스를 이용하여 이러한 서비스도 3rd Party가 충분히 개발할 수 있다는 것을 보여준다.

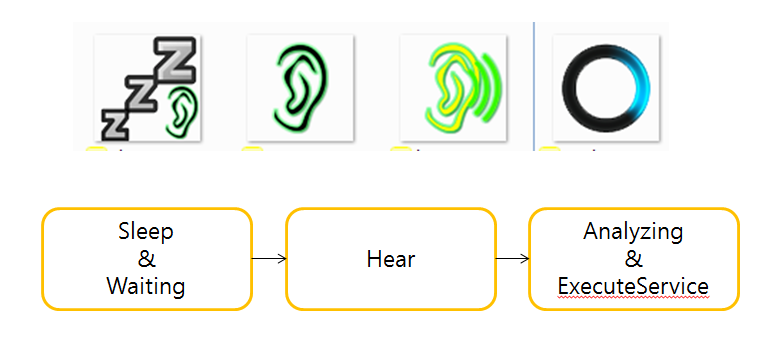
## 6) User Interface

6.1) SystemUI

Android의 SystemUI는 사용자에게 기기를 통해 정보를 항상 확인할 수 있도록 구현되어 있다. System UI에 종속된 대표적인 예로 Notification, QuickPanel 등이 있으며, 사용자의 기기 제어에 대한 상호작용을 담당하고 있다.



이에 음성 제어 시스템과 상호작용을 위해 QuickPanel에 Smart Voice를 새롭게 추가하고, 사용자에게 음성에 대한 상호작용을 실시간으로 알릴 수 있도록 SmartVoice를 위한 System UI에 종속되는 OverlapUI 개념을 도입하였다.



이제 사용자는 음성 제어 시스템을 SystemUI 중의 하나인 QuickPanel에서 종료하거나 시작시킬 수 있다.(그림의 QuickPanel 좌측 중앙 Tile) 또한, 음성 제어 시스템이 가동된 경우 SystemUI에 추가된 OverlapUI가 작동하여(우측 하단UI), 음성 제어 상황을 직관적인 Symbol로 표현하여 준다.

6.2) Smart Voice Settings

음성 제어 프레임워크에서 시스템 애플리케이션으로 SmartVoice Settings을 추가하였다.

초기 환경설정 애플리케이션을 실행하면 실시간으로 상단에 현재 사용자의 음성 수신 데이터, 내부 분석 결과 버퍼의 데이터, 호출하는 서비스가 표시된다. 그리고 하단의 아이콘을 통해서 Smart Voice Profile과 설치된 Service List를 확인할 수 있다.



Figure 차례로 Profile UI, Main UI, Settings UI

Profile UI는 현재 SmartVoice의 호칭이 표시되며 사용자가 호칭을 변경할 수 있다.

Settings UI에서 현재 Android OS에 설치된 Service List를 확인할 수 있다. 이 중에서 사용자는 원하지 않는 기능을 종료할 수 있다.

## 7) Android PDK



Figure Java System Service

Android PDK 환경을 구축하기 위해서는 먼저 Linux를 host OS로 설치해야 한다. AOSP에서는 Linux 버전 중 Ubuntu를 권장하고 있다. 이 Ubuntu도 안드로이드 버전에 따라 권장되는 버전이 있는데 이 프로젝트에서 사용한 Jelly Bean 4.3 버전에서는 Ubuntu 12.04 버전의 설치를 권장하고 있다.

그리고 안드로이드의 오픈소스 프로젝트로서 AOSP의 커뮤니티가 운영되고 있고, Git을 통하여 관리되고 있다. 이렇게 Git을 통해 관리되는 소스들이 안드로이드 버전 별로 존재하는데 리눅스의 repo 도구를 이용해 안드로이드의 특정 버전 소스를 받아올 수 있다.

## 8) Android Custom Platform Porting



Figure 프로그레시브 다운로드와 스트리밍 차이

수정 된 안드로이드를 실제 단말기에 올리려면 먼저 특정 단말기에 대한 H/W에 접근할 수 있는 Binary 파일이 필요하다. 특정 레퍼런스 폰에 대한 H/W Binary 파일은 구글에서 제공하고 있기 때문에 레퍼런스 폰을 사용하여 포팅을 진행하였다. 사용한 레퍼런스 폰은 갤럭시 넥서스인데 포팅하고자 하는 안드로이드 버전과 제공해주는 Binary 파일을 잘 보고 맞는 단말기를 선택해야한다.

이렇게 Binary 파일을 스크립트 명령을 통해 설치하고 난 후 수정된 안드로이드 소스에 대한 이미지 파일을 만들어야 한다. 먼저 이러한 이미지 파일을 만들기 위해 어떠한 단말기를 타겟으로 이미지 파일을 만들지 선택해준다. 그리고 만약 3rd Party에게 제공하는 API가 수정되었다면 make update-api를 이용해 current.txt 파일도 갱신해 주어야 한다. 이 과정이 완료된 후 make 명령어를 통해 안드로이드 소스를 빌드하게 되면 포팅할 이미지 파일이 만들어지게 된다.

그리고 이렇게 만들어진 이미지 파일을 fastboot 도구를 이용하여 이미지를 올려주면 포팅이 완료된다.

# 4.개발 일정

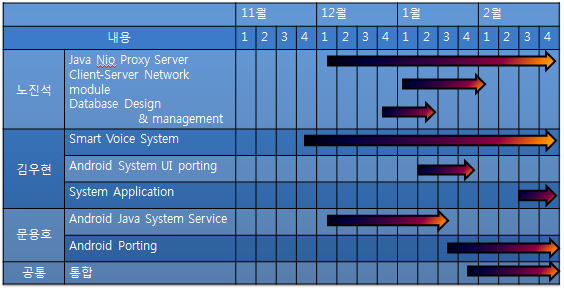


Figure 개발 일정

# 5. 용어 정리

## 1)aidl

Android Interface Definition Language. 안드로이드에서 RPC 통신을 하기 위해 정의한 IDL 언어이다. 이 aidl 파일을 이용하여 RPC 통신에서 필요한 Proxy와 Stub을 자동으로 생성할 수 있다.

## 2)RPC

Remote Procedure Call. 서로 다른 프로세스에 존재하는 원격 함수를 마치 자신의 함수인 것 처럼 호출 할 수 있게 해주는 형식

## 3)Binder IPC

리눅스에서 프로세스간 통신을 안드로이드에 맞게 수정된 IPC

## 4)Java System Service

안드로이드 프레임워크 단에서 다른 프로세스에게 서비스를 제공해주는 서버. 대표적인 예로 Activity Manager, Package Manager, Alarm Manager 등이 있다.

## 5)Java NIO

Java New I/O. 자바에서 비동기 입출력을 지원하기 위해 개발된 새로운 IO 방식

## 6)PDK

Platform Development Kit. 안드로이드 프레임워크를 개발하는 환경

# 6. 참고 문헌

1)송형주, 김태연, 박지훈, 이백, 임기영, "인사이드 안드로이드"

위키북스

2)고현철, 유형목, “안드로이드의 모든 것 분석과 포팅”

한빛미디어

3)김태연, 박지훈, 김상엽, 이왕재, “안드로이드 아나토미”

정권