모바일 장치를 이용한 기타 연주

요 약

최근 세계적으로 Touch Screen 기반의 Mobile Device 의 발달로 Mobile Device 의 쓰임이 점점 다양해지고 있다. 따라서 본 문서는 두 대의 Android 기반 Touch Screen Device(Smart Phone, Tablet)에서 한대의 Device 는 기타의 몸체(기타 줄을 치는 부분)가 되고 나머지 한 대의 Device 는 기타의 목(기타의 코드를 잡는 부분)이 되어 기타를 연주할 수 있는 실제 악기와 유사한 GUI 를 갖는 어플리케이션을 제안하고 구현한다.

1. 서론

1.1. 연구배경

최신 스마트 장비들의 다양성과 스마트 장비의 수요가 기하급수적으로 늘어나고 있음에 따라 어플리케이션 개발에 많은 관심이 모이고 있다.

스마트 장비를 사용하는 사용자들의 자신을 만족시켜 줄 수 있는 다양한 컨텐츠들의 요구가 증가함에 따라 스마트폰 어플리케이션의 개발이 급격히 증가하고 있다.

많이 개발되고 있는 어플리케이션 부분은 대부분의 사람들이 취미로 갖고 있는 음악 연주나 음악 재생에 관련된 어플리케이션 등이 많이 나오고 있다.

많은 사람들이 기타 연주와 소리 재생에 대한 어플리케이션을 이미 어플리케이션 시장에 제공하였으나 기존에 나온 기타 어플리케이션들은 자신이 음악을 연주한다는 느낌은 배제된 채 코드를 잡아주어 기타 소리에 중점을 두어서 만들어 졌다.

기존의 어플리케이션과는 다르게 좀 더 실제와 비슷한 느낌의 연주법과 기존에 나온 어플리케이션의 장점을 파악하여 두 스마트 장비를 사용하여 실제 기타를 치는 듯한 느낌을 주도록 구현하였다.

실제 기타와 같이 코드를 잡는 부분과 기타를 치는 부분을 두 개의 스마트 장비로 분류하여 두 장치간 블루투스(Bluetooth) 통신을 통하여 잡은 코드부분의 소리를 기타를 치는 부분의 스마트 장비에서 사용자가 기타를 치는 세기에 따라 소리의 강도를 조절하여 음악을 재생한다. 기타를 치는 세기는 속도로 측정하여 화면에 포인터를 잡고 손이 이동한 거리를 이동했을 때의 시간으로 나누어서 측정한다. 코드를 잡아 소리를 전송하는 부분은 여러 화음이 섞여 코드소리가 잘 나도록 멀티 터치를 사용하여 실제 기타가 잡히는 방법과 최대한 같이 될 수 있도록 UI 를 제공한다.

두 장치를 이용하여 기타를 연주해 보는 어플리케이션 이므로 두 손 모두 자유롭게 사용할 수 있으며 기타 소리를 듣는 어플리케이션이 아닌 실제 연주하는 느낌을 사용자에게 줄 수 있다.

1.2. 연구목표

두 스마트 장비를 이용한 음악 악기 연주를 하는 어플리케이션으로 실제와 같은 느낌을 주는 것을 목적으로 실제 악기 보다 쉬운 연주 기법과 실제 악기의 불편을 디지털이라는 장점을 이용하여 새로운 사용자 사용환경을 제공하여 쉽게 사용가능하고 접근 가능하게 만들 것이며 어디서든 사용해 볼 수 있다는 시공간적 장점을 활용할 것이다.

첫 번째 목표로 음을 중시하는 어플리케이션 이므로 기존의 어플리케이션의 한계점인 코드소리만 제공 하거나 더 추가된 것은 기타 한 줄 한 줄 소리를 재생해 주지만 생동감 넘치는 소리재생을 제공하기 위해 두 장치간 코드(Chord) 잡는 장치와 스트로크(Stroke)를 치는 두 스마트 장치로 나누어 자신이 기타를 실제로 치면서 소리가 나오는 느낌이 들게 모델을 구성하였다.

두 번째 손기술이 많이 필요로 하는 악기인 만큼 손에 전달되는 감각을 최대한으로 하기 위해 코드를 치게 되면 미세진동으로 자신이 코드를 잡고 실제 기타처럼 치고 있다는 감각을 전달해준다.

세 번째로 음악 악기를 다루는 목적이 음악적 지식과 음악적 활동을 하는 것이므로 사용자가 연주한 음악을 악보로 제공하여 주거나 갖고 있는 악보를 활용하는 기능을 추가하여 활용성을 높일 수 있는 방향으로 구현되었다.

2. 관련연구

2.1. 스마트 장치간 근거리 통신

코드를 잡는 부분과 스트로크를 치는 장치간 음 인자를 전송시켜 동일한 음을 갖고 있어야 한다.

2.1.1. 블루투스

블루투스(Bluetooth)는 10M 정도의 짧은 거리에서 사용이 가능한 근거리 무선 통신망으로, 다른 기술에 비해 상대적으로 저렴하고 전력 소모가 적어 대부분의 스마트폰 기기에 기본 장착되어 있다. 헤드폰, 키보드 등 무선 기기를 연결하거나 데이터를 전송하는 기능을 수행한다. 2012 년 현재 스마트폰에 사용되는 블루투스는 3.0~4.0 버전으로, 최대 24Mbps 전송 속도를 지원하고 있지만 최대 300Mbps 가 넘는 와이파이 다이렉트에 비해서는 상대적으로 전송 속도가 느린 편이다.

2.1.2. 와이파이

스마트폰 무선 인터넷 접속 시 자주 사용하는 와이파이(Wi-Fi)는 무선랜을 이용해 무선 데이터를 송수신하는 데이터 기술이다. 특히 가정에서 무선 인터넷 공유기 설치만으로 AP(Access Point)를 설치해 반경 10~20M 거리까지 무선 통신을 쓸 수 있다. 반면 와이파이 다이렉트(Wi-Fi Direct)는 AP 없이 무선으로 직접 다른 기기에 접속해 데이터를 송수신하는 기술로, 와이파이 다이렉트를 지원하는 스마트 기기끼리 데이터를 주고 받을 때 주로 사용된다. 와이파이 다이렉트는 블루투스보다 전송 반경도 넓고 속도도 빠르기 때문에, 무선 프린터, 스마트 TV, 대전 게임 등 다양한 형태로 사용되고 있다.

2.1.3. 안드로이드 빔/S 빔

안드로이드 빔(Android Beam)은 10cm 이내의 가까운 거리에서 데이터를 송수신하는 통신 기술인 NFC(Near Field Communication)를 이용, 안드로이드폰에서 데이터 통신을 할 수 있도록 지원하는 기술이다. S 빔은 갤럭시 3 에 새롭게 등장한 기술로 NFC 와 와이파이 다이렉트 기술을 결합해 전송 속도를 향상시킨 기술이다.

2.2. 멀티터치

코드를 잡을 때 화음이 섞여야 되므로 여러 가지 음을 잡고 동시에 소리를 치게 된다.

멀티터치(Multi-touch)는 터치스크린, 터치패드가 동시에 여러 개의 터치 포인트를 인식하는 기술로, 일반적인 하나의 터치 포인트만 인식을 하는 것보다 더 다양한 조작을 할 수 있다. 현재 정전식 터치 기술이 사용된 터치패드, 터치스크린에서만 적용되는 기술이다.

터치를 통해서 위치 변화만 입력할 수 있었기 때문에 다양한 조작을 위하여 보조 단추 같은

별도의 조작이 필요했던 기존의 터치방식과는 달리, 감지되는 터치 포인트의 갯수에 따라 터치에 대한 장치의 반응을 지정할 수도 있고 터치포인트 간격 변화를 통한 조작도 가능하기 때문에 더 직관적이고 쉽고 편하게 조작할 수 있다.

2.3. 음악소리

여러 소리를 동시에 재생하여 음악이 재생이 되고 음과의 연결이 하나의 노래가 되는 것이므로 소리의 중첩이 중요한 기술로 요구된다.

사운드풀(SoundPool)이라는 안드로이드 내부 라이브러리를 사용하여 여러 소리를 중첩하여 사용하는 미디어 플레이어 클래스이다. Raw 에 저장된 소리를 클래스 생성시 얼마나 중첩하여 사용하는지 기본변수를 입력 받아 생성할 수 있으므로 소리 중첩에 관련된 기술을 해결하기 위해 효과적으로 사용 될 수 있다.

2.4. 기존 기타에 관련된 어플리케이션 연구

코드를 이용하여 소리를 제공하나 자신이 직접 치는 듯한 느낌은 조금은 부족하고 코드소리의 빠른 전환이 힘들다. 뿐만 아니라 멀티터치 제공에서 모든 코드를 잡을 수 있을 정도의 기술력을 제공하고 있지 않고 실제기타와 같은 크기의 범위를 제공하는 어플리케이션은 없다.

2.5. 기존 연구의 문제점 및 해결 방안

2.5.1. 연구의 문제점

위에서 언급한 세가지 기술력을 기반으로 어플리케이션 개발에 중점을 잡고 있다.

두 스마트 장치간 사람이 소리로 인지하지 못할 정도의 전송속도로 소리 인자를 전달해 주어야 하며 실제와 같은 코드를 잡을 수 있게 하려면 많은 코드의 모양을 검색해 보아 멀티 터치를 제공해 주어야 한다. 또한 제일 중요한 소리부분에서 여러 화음이 귀에 거슬리지 않게 소리를 중첩시켜 하모니를 제공해야 한다.

위 세가지 기술적 문제를 사용자가 사용하기 편한 사용자 환경을 제공하여 실제 기타와 같은 느낌이 제공되어야 한다.

2.5.2. 해결 방안

2.5.2.1. BlueTooth

두 스마트 장치간 전송 방법 중 블루투스를 이용하여 두 장치간 전송 속도를 계산하여 본 결과 200ms 를 평균으로 제공해 준다. 이정도 속도로 측정해본 결과 사람이 움직이는 속도보다 빠른 전송을 제공해 주기 때문에 소리 전달 방법으로 무리 없이 사용할 수 있다.

2.5.2.2. 사용자 환경을 이용한 간편한 연주기법

많고 다양한 주법이 필요로 하는 악기인 기타이므로 초보자 혹은 전자장치를 이용한 쉬운 연주를 위하는 사람들을 위해 새로운 방식의 사용환경을 제공하여 실제 기타를 칠 때 힘들었던 부분을 간추릴 수 있을 것이다.

진동이나 손가락에 의해 눌린 기타 줄 마다 빛을 발하는 기능을 제공하여 사용자가 자신이 잡고 있는 코드를 인지시켜주고 실제 기타의 떨림을 느낄 수 있게 해준다.

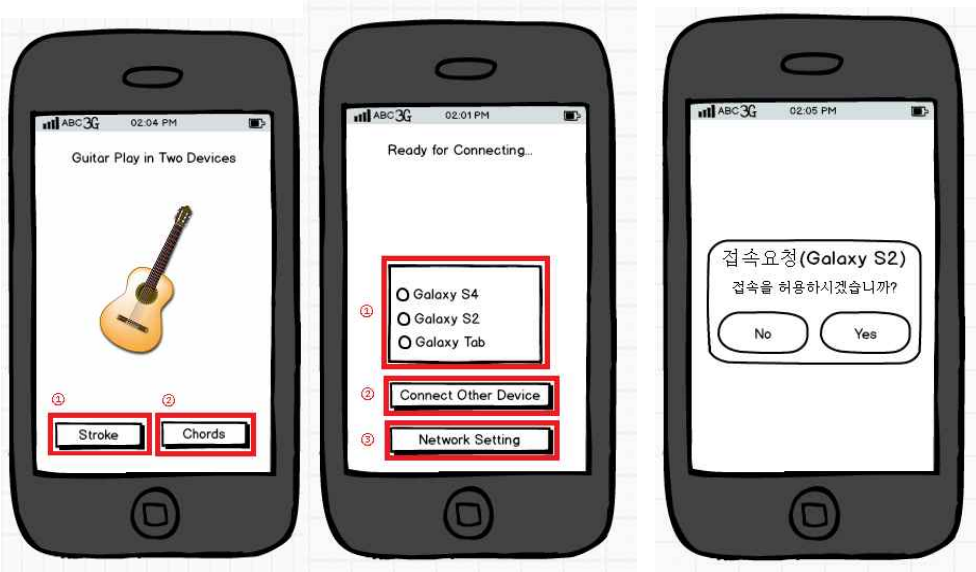
2.5.2.3. SoundPool

미디어 파일들을 안드로이드에서 제공해 주는 라이브러리이다. 여러 소리의 동시 재생이 가능하고 자신이 원하는 소리를 선택하여 사용하므로 여러 기타소리를 제공해 줄 수 있다.

3. 프로젝트 내용

3.1. 시나리오

3.1.1. Default GUI



[그림 1] 메인 화면 [그림 2]연결 단말 선택 화면 [그림 3]접속요청 화면

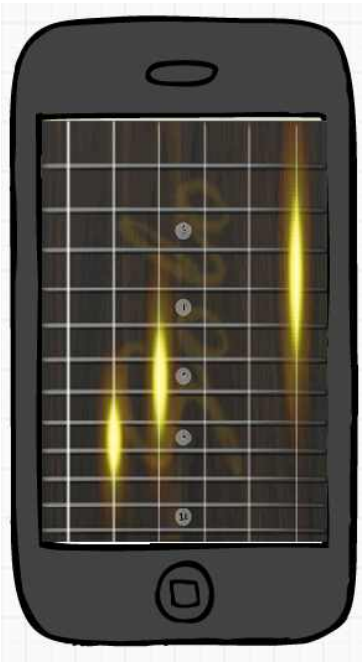
[그림 1]는 어플리케이션의 메인 화면이다. 메인 에서는 Stroke(치는 부분)과 Chords(코드 잡는 부분)을 선택할 수 있는 화면이 출력된다. 1 번 버튼 Stroke 를 클릭하면 해당 Device 는 Server 역학을 하게 되며 다른 Device 의 접속을 대기한다. 2 번 버튼 Chords 를 클릭하면 **[그림 2]**화면이 출력되고 현재 블루투스로 연결되어있는 디바이스들 중에 Server 역할(Stroke 부분)을 하는 Device 들을 1 번에 출력한다. 2 번 버튼은 Stroke 로 접속대기중인 디바이스에 접속 요청을 한다. 3 번 버튼은 블루투스 환경설정으로 연결된다. 블루투스로 연결된 디바이스가 없다면 먼저 디바이스를 연결한 후 어플리케이션을 재 시작한다. Stroke 의 대기상태에서 다른 디바이스가 접속요청을 한다면 **[그림 3]**화면이 출력되고 확인을 누르면 어플리케이션이 시작된다.

3.1.2. Main Device GUI - Stroke

Stroke 부분은 별다른 인터페이스가 없이 6 개의 줄로만 구성되어 있다. Stroke 를 하면 진동이 울리게 된다.

3.1.3. Serve Device GUI – Chords

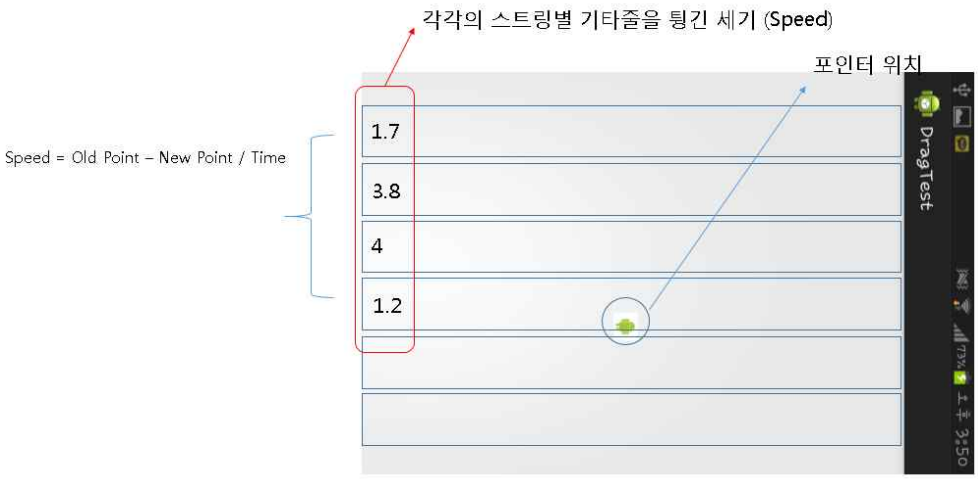
Chords 부분은 코드를 잡는 부분이다.
Chords 를 잡으면 잡은 부분이 활성화 되고 Stroke 부분에서 Stroke 을 하면 진동이 울리게 된다. 옆의 마디로 이동하는 것은 별도의 버튼이나 볼륨 업다운 키로 조정한다.(Device 별로 다름)



3.2. 요구사항

3.2.1. Stroke Device 에 대한 요구사항

- 기타 줄(6 string)에 대해 소리가 융화되어 화음소리가 나도록 구현한다.
- 속도 알고리즘 (속도 = 거리/시간) 사용 하여 소리의 세기 계산한다. 기타 줄을 Stroke 할 때, 살살 연주하면 작은 소리가 나도록 하고 작은 소리가 나는 만큼 음의 길이도 짧게 구현한다. 반대로 세게 연주하면 큰 소리가 나도록 구현하고 음의 길이를 길게 구현한다.
- 화면을 누르는 압력을 사용하여 소리의 세기 계산한다. 한 줄 한 줄 튕기는 연주를 할 때 살살 연주하면 작은 소리가 나도록 하고 반대로 세게 화면을 누르면 큰 소리가 나도록 구현한다.
- 서버의 역할을 한다. Stroke 부분에서는 현재 Chords 부분에서 전송되는 Chords 를 받아서 소리를 재생한다. 한번 받은 Chords 는 Chords 부분에서 다시 받기 전까지 유지된다. 소리의 크기 및 길이를 판단하기 위하여 사용자의 Touch Speed 를 계산하여 구현한다.



[그림 4] Stroke Proto Type

3.2.2. Code Device 에 대한 요구사항

- 플랫 이동을 화면 아래쪽에 버튼을 사용한다. 디바이스 하나의 장치에 많은 플랫이 표현되지 않기 때문에, 화면 아래쪽 3~4 개의 버튼을 사용하여 화면이동 구현한다.
- Barre Code 를 위해 각 플랫 마다 화면 위쪽에 버튼 사용한다. 기타 여섯 줄을 다 눌러야 하는 Barre Code 를 구현하기 위해, 화면 상단 버튼을 눌렀을 때 해당 플랫에 모든 줄이 눌리도록 구현한다.
- 기타 줄이 튕기는 느낌을 위해 Haptic 진동 사용한다. 기타 연주 시 실제로 기타가 진동하는 모습을 구현하기 위해 Code Device 에 진동효과를 주도록 구현한다.

3.2.3. BlueTooth 통신에 대한 요구사항

- 이 프로그램만의 Data 전송 약속을 만들어 사용한다. 전송속도를 줄이기 위해 Data 의 원래 모습 그대로 전송한다면, Data 양이 많아져 전송속도가 느려진다. 따라서 이진수로 변환하여 전송하고 데이터를 받은 후 해석한다. 다음은 이에 해당하는 프로토콜이다. 기타는 총 6 개 줄이고 마디를 최대 20 개라 가정한다. 아래 그림은 데이터 포맷이다.

Mute (xxxxxx)	잡고있는줄 (xxxxxx)	1 번째 줄 (xxxxx)	2 번째 줄 (xxxxx)	3 번째 줄 (xxxxx)	4 번째 줄 (xxxxx)	5 번째 줄 (xxxxx)	6 번째 줄 (xxxxx)

X 는 1bit 이다. Mute 는 Mute 인 줄을 1 로 표시한다. 1, 2 번째 줄이 Mute 라면 Data 는 110000 이 된다. 잡고 있는 줄도 Mute 와 같다. 3, 4 번째 줄을 잡고 있다면 001100 으로 표시되고, 뒤에 나오는 데이터는 몇 번째 줄은 어떤 마디를 잡고 있는지 표시한다. 잡고 있지 않을 줄의 데이터는 생략한다. 따라서 1, 2 번째 줄이 Mute 고 3 번째 줄의 2 번째 마디를 잡고 있고 4 번째 줄의 3 번째 마디를 잡고 있다면 프로토콜은 아래와 같다.

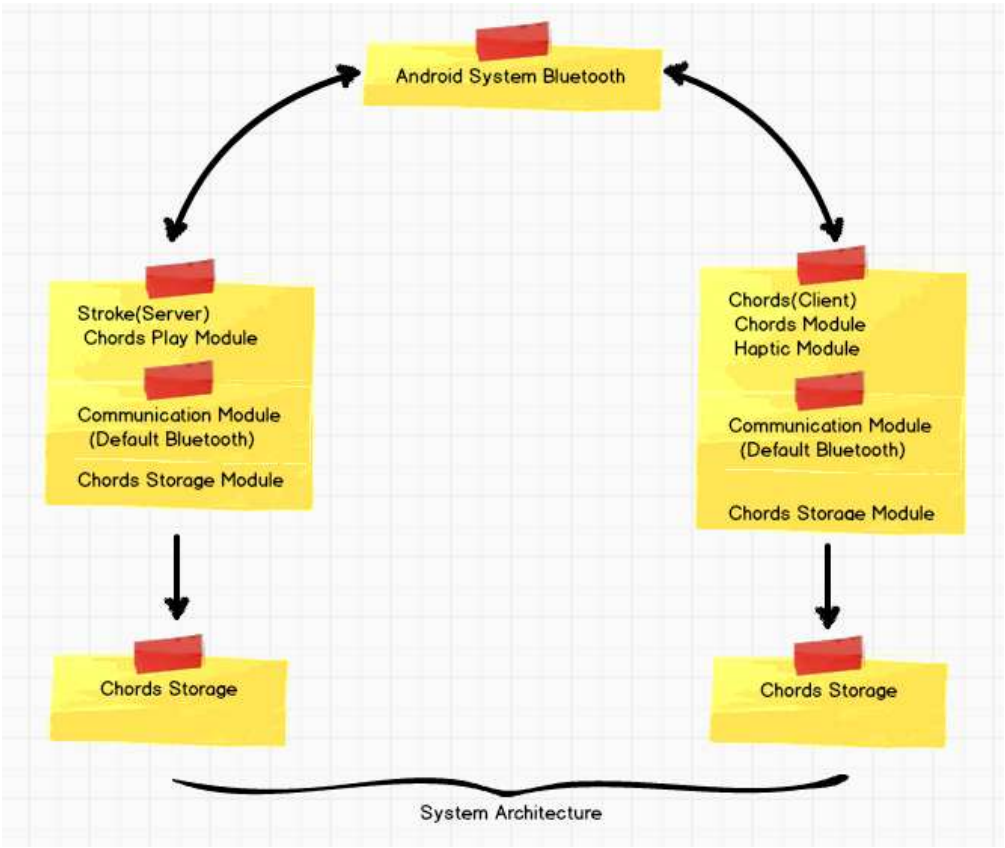
110000, 001100, 00010, 00011 이다.(',' 는 편의상의 문자)

따라서 가장 클 때의 데이터가 42bit 이므로 6byte 를 넘지 않는다.

- Code Device 에서 Data 를 전송한다. Code Device 에서 사용자가 누르고 있는 부분을 Stroke Device 로 전송한다. (음 전달)
- Stroke Device 에서 Data 를 전송한다. Stroke Device 에서 줄을 튕길 때 Code Device 장치로 상태 변화를 전달한다.(진동 상태 전달)

3.3. 시스템 설계

3.3.1. 시스템 구성도

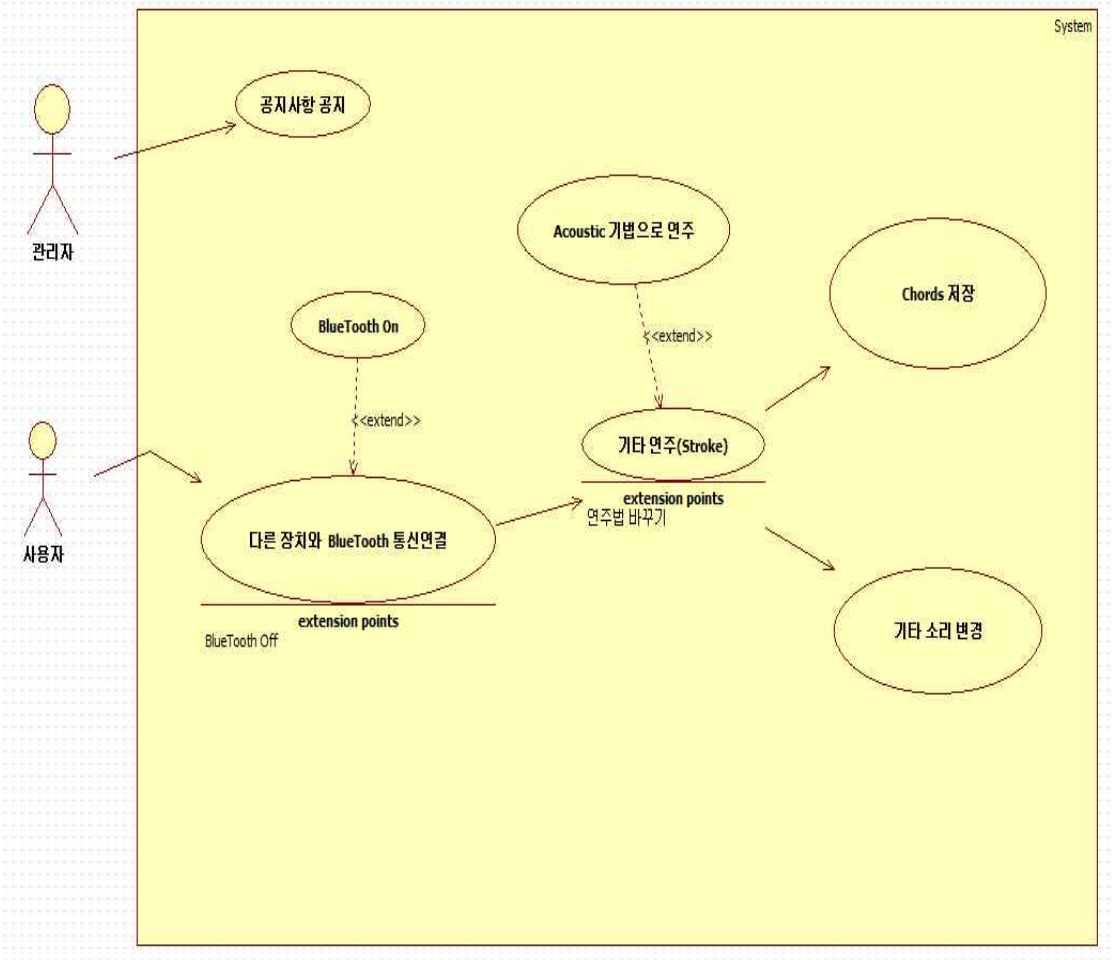


[그림 5] 시스템 구성도

시스템 구성도는 [그림 5]과 같다.

3.3.2. UML Diagram 을 통한 시스템 모델링

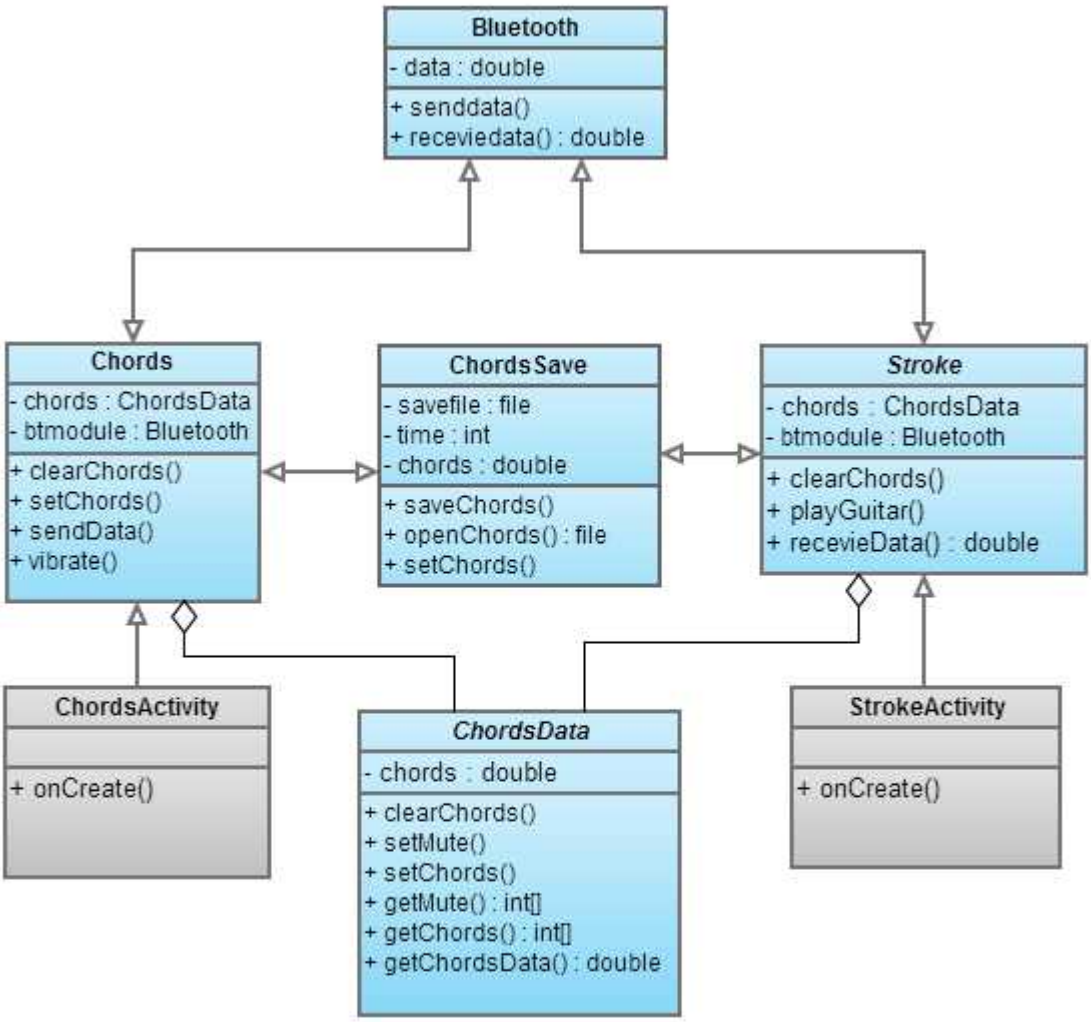
3.3.2.1. Use Case Diagram



[그림 6] Use Case Diagram

Usecase 는 간단하다. 관리자는 프로그램에서 공지사항을 올리는 것 이외에는 관여하지 않는다. 사용자는 두 대의 장치를 BlueTooth 를 통하여 통신을 연결시킨다. 각 장치마다 Server 와 Chord 부분을 설정한 후 기타 연주를 한다. 기타 연주 할 때 Stroke 방법과 Acoustic 방법 중 하나를 선택할 수 있으며, Default 로 Stroke 방법이 설정된다. 연주 후에는 기타 소리를 변경하거나 연주한 것을 Chord 로 저장한다.

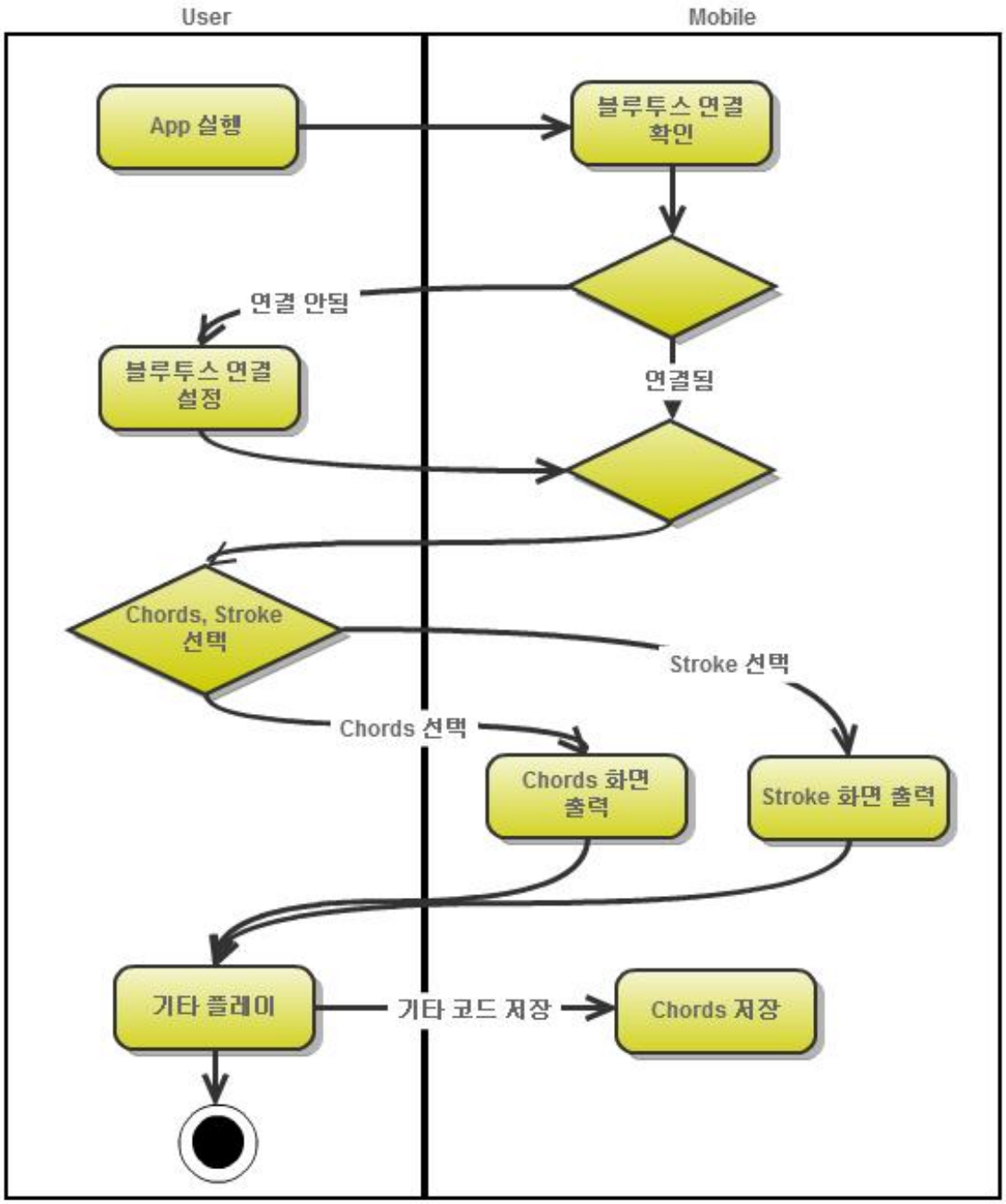
3.3.2.2. Class Diagram



[그림 7] Class Diagram

클래스는 크게 Bluetooth, Chords, Chords 저장부분, Stroke, Chords Protocol 부분으로 이루어져 있다. Chords 를 잡는 부분은 사용자가 잡은 코드를 블루투스를 통해서 Stroke 에 보내고 Stroke 는 Chords 부분으로부터 받은 코드로 Sound 를 재생한다. ChordsSave 부분은 연주되는 Chords 부를 저장한다.

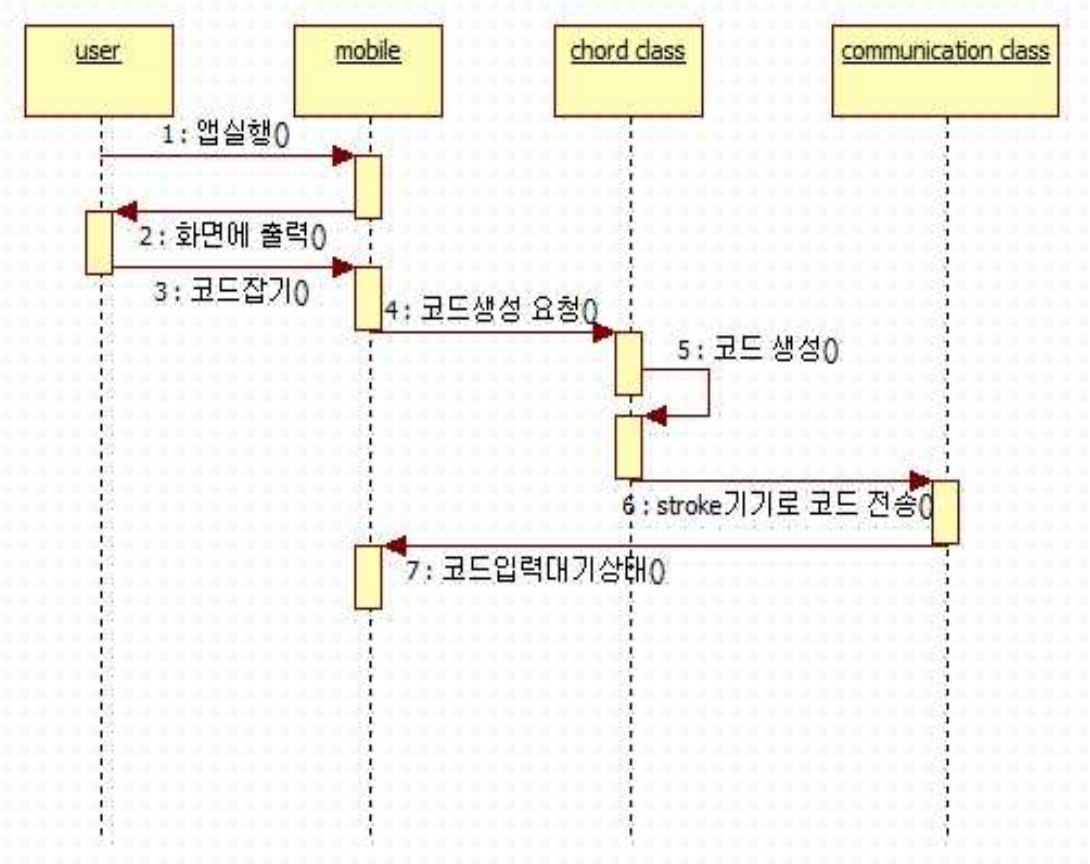
3.3.2.3. Activity Diagram



[그림 8] Activity Diagram

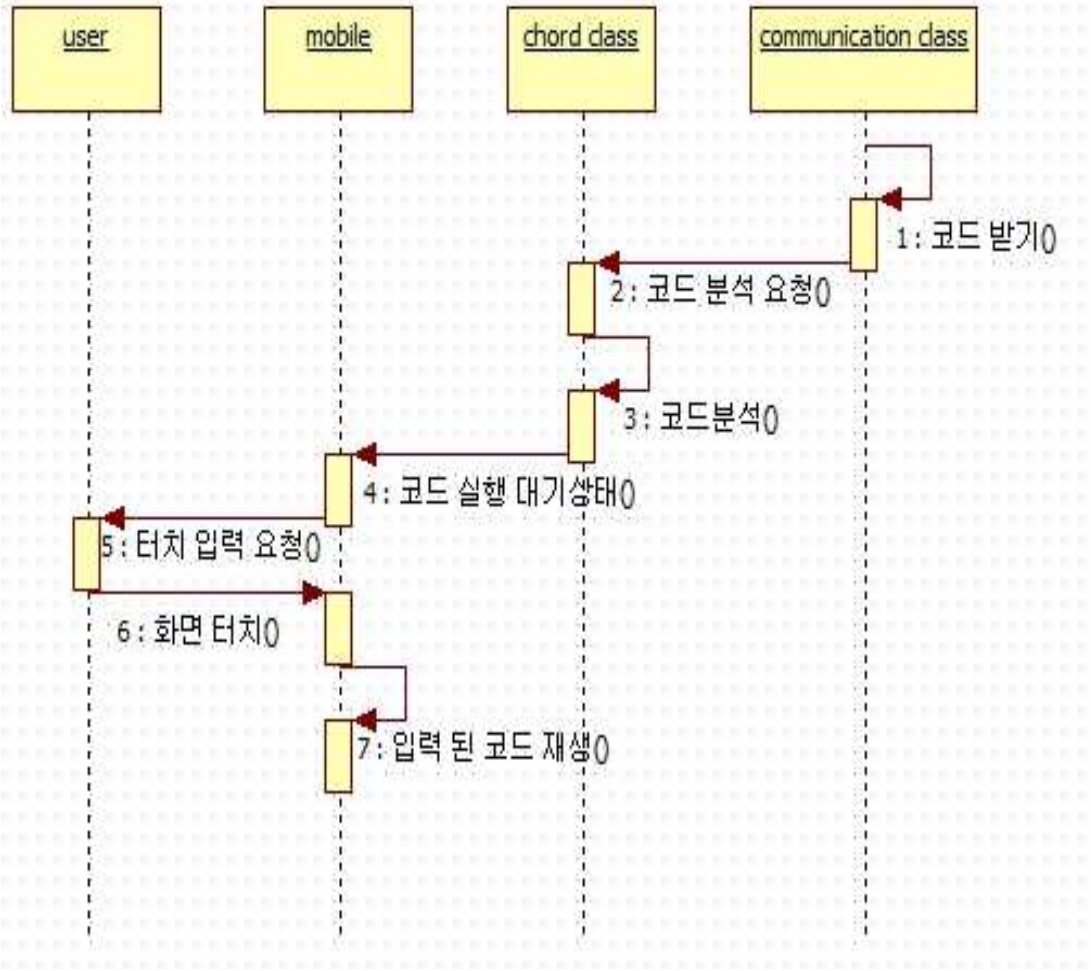
Activity는 매우 간단하게 이루어져있다. 어플리케이션을 실행하면 블루투스 연결을 확인하고 코드를 잡는 부분과 스트로크를 치는 부분으로 나뉘어서 어플리케이션이 실행되게 된다. 추후 기능 추가를 통하여 블루투스가 연결되지 않았을 경우 하나의 디바이스로도 기타를 칠 수 있는 부분을 구현할 것이다.

3.3.2.4. Sequence Diagram



[그림 9] Chord 기기 기능의 Sequence Diagram

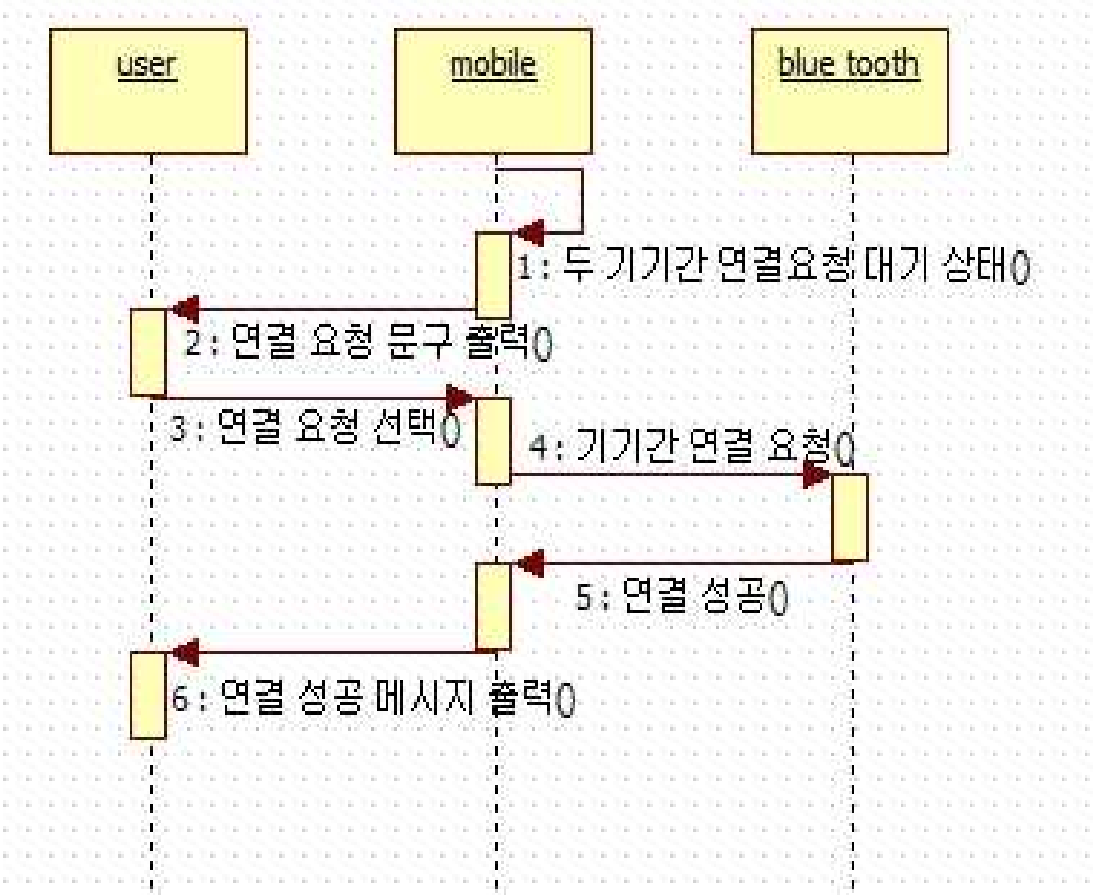
Chord 기기 기능의 Sequence Diagram을 보여주고 있다. Mobile기기는 사용자의 입력을 대기하고 있으며 사용자의 입력이 들어오는 대로, 즉 사용자가 화면에 출력 된 기타모양의 인터페이스에 터치를 하는 데로 입력을 받아 받은 입력의 위치를 Chord Class에서 코드화 시킨다. 코드화 시킨 전송 가능한 데이터를 Stroke 기기로 전송을 한 후 다시 코드 대기 상태로 돌아간다.



[그림 10] Stroke 기기 기능의 Sequence Diagram

Stroke 기기 기능의 Sequence Diagram을 보여주고 있다. 연결이 된 Chord 기기로부터 코드가 생성이 되어 전송된 데이터를 받는다. 받은 데이터는 Chord Class를 통하여 코드를 분석하고 분석한 코드의 소리를 각각의 해당 기타 줄에 적용한다.

모바일 기기는 기타 줄 화면을 출력하고 있어 사용자로부터 터치 입력 대기 상태를 하고 있고 터치가 들어오면 입력되어 있는 해당 기타 줄 소리를 재생한다.



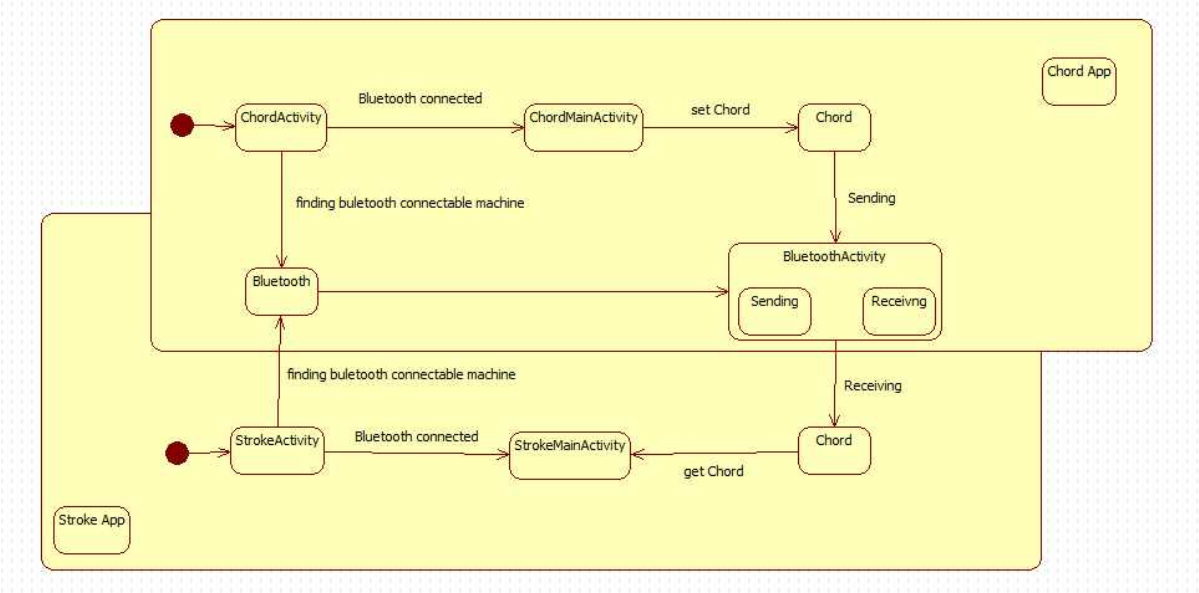
[그림 11] 연결 기능의 Sequence Diagram

앱을 처음 사용할 때 두 기기간 장치를 연결 하여야 한다. 모바일 창에서는 처음 기기를 연결하는지를 물어보고 입력 받은 요청에 따라 처리한다.

기기 연결을 입력 받으면 그 뒤 블루투스로 연결 가능한 기기를 찾는다. 연결 가능한 기기를 찾은 후 연결을 시도한다.

연결이 성공되면 모바일 창에 연결 성공 메시지를 띄운다.

3.3.2.5. State Diagram

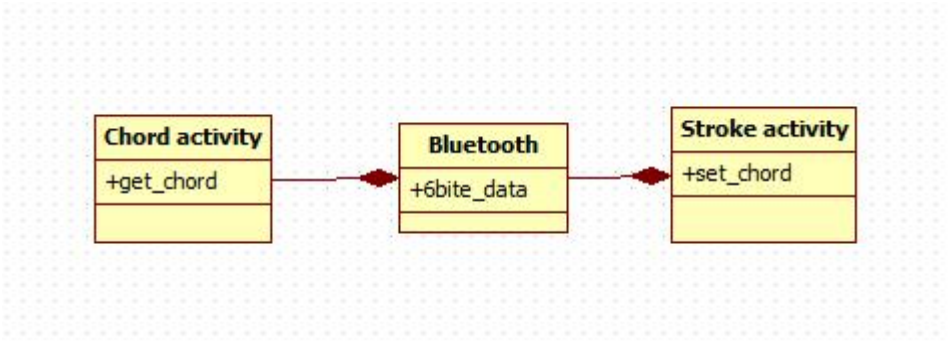


[그림 12] System 의 State Diagram

각각의 기반 App 의 주요 기능에 기반하여 전체 시스템을 구현하였다. 사용자가 앱 사용관점에서 설계적인 면으로 설명했으며 간단한 주요기능만을 작성하였다. 메인 메뉴 밑 하위 기능의 경우 Sequence Diagram 에서 좀 더 세분화 하여 다루었으며, 현재의 State (Machine) Diagram 은 Event Table 의 Trigger 를 기준으로 작성하였다.

3.3.2.6. Entity-Relation Diagram

Database 에서의 상태를 좀 더 명확하게 명시하기 위한 ER Diagram 이다.



[그림 13] ER Diagram

[Table 관계도]

- **Chord activity 와 Bluetooth (1:1)**

  Chord activity 는 하나의 **Bluetooth** 연결을 갖는다.

- **Stroke activity 와 Bluetooth (1:1)**

  Stroke activity 는 하나의 **Bluetooth** 연결을 갖는다.

[Table 정보]

- Chord activity 테이블 정보

Chord activity			
열명	**타입**	**크기**	**옵션**
get_chord	int	6	기본키

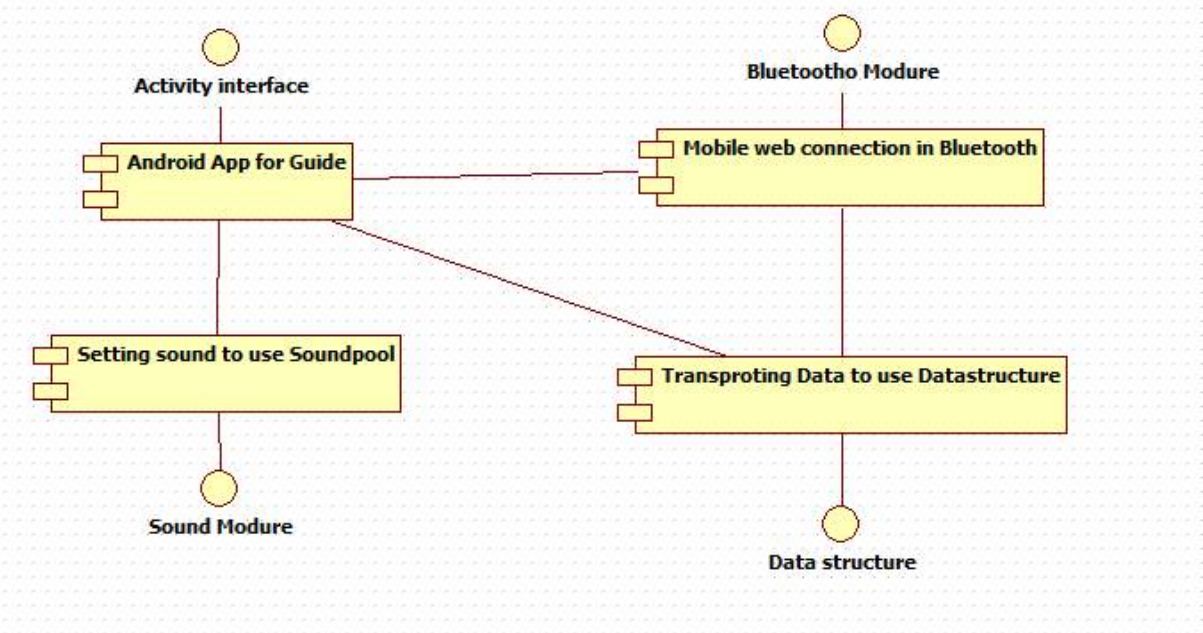
- Bluetooth 테이블 정보

Bluetooth			
열명	**타입**	**크기**	**옵션**
6bit_data	Data	6bit	기본키

- Stroke activity 테이블 정보

Stroke activity			
열명	**타입**	**크기**	**옵션**
set_chord	int	6	기본키

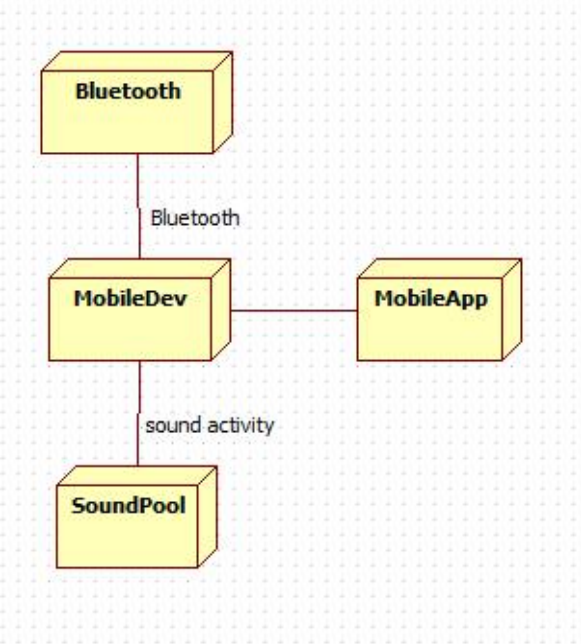
3.3.2.7. Component Diagram



[그림 14] 시스템 환경에 관한 Component Diagram

전반적으로 다른 Diagram 들과 동일하며, 각각의 Component 에서 사용되는 대략적인 Interface 및 Artifact, 즉, 활용 정보에 관한 부분을 요약한 Diagram 이다. 물리적 시스템 구축 단계를 표현한 Deployment Diagram 과 겹쳐보면 전반적인 구현 사항을 확인 할 수 있다.

3.3.2.8. Deployment Diagram



[그림 15] 기능적 시스템 환경에 대한 Deployment Diagram

상기 설정된 Component Diagram 을 기반으로 시스템의 기능적인 구성 요소를 좀 더 넓게 모델링 한 다이어그램이다. 내용적으로는 Component 와 많이 다르지 않으며, 구현단계에서 물리적인 장비들을 상세화 함으로써, 구현단계의 테스트 단계에서는 수정될 예정이다. 대체로 Mobile App 과 그에 관련된 기능을 중심으로 나타나 있다.

3.3.2.9. Event Table

각 입력에 따라 발생할 수 있는 모든 이벤트를 표기하였으며, 각 기기(Chord 기기, Stroke 기기) 따라 발생할 수 있는 상황을 모두 정의하였다.

Guitar Playing System Event Table					
Event	**Trigger**	**Source**	**Use Case**	**Response**	**Dest**
사용자가 코드화면 터치	코드 데이터화 성공	사용자	입력 코드 데이터화	코드 입력 창 출력	Stroke 기기
사용자가 코드 변경	코드 데이터화 성공	사용자	입력 코드 데이터화	코드 입력 창 출력	Stroke 기기
사용자에 의한 터치가 없을 때	코드 데이터화 성공	사용자	무입력코드 데이터화	코드 입력 창 출력	Stroke 기기
사용자가 기타화면 터치	코드 재생	사용자	코드 재생 대기	기타화면 출력	Stroke 기기
사용자에 의한 터치가 없을 때	아무 재생 없음	사용자	코드 재생 대기	기타화면 출력	Stroke 기기
사용자가 기타화면 강하게 터치	강도에 의한소리 재생	사용자	강도 계산	기타화면 출력	Stroke 기기

사용자가 기타화면 강하게 터치	강도에 의한 진동 재생	사용자	강도 계산	기타화면 출력	Chord 기기
Chord 기기 처음 연결	기기간 연결 성공	사용자	연결 입력 확인	연결 성공 출력	Stroke 기기
Stroke 기기 처음 연결	기기간 연결 성공	사용자	연결 입력 확인	연결 성공 출력	Chord 기기
코드 생성	코드 데이터화 성공	사용자	코드 데이터화	다음 입력 대기	Chord 기기
코드해독	코드 해독 성공	사용자	코드 해독화	다음 입력 대기	Stroke 기기
기기간 연결 해제	해제 성공	사용자	연결 해제 시도	연결 해제 출력	모든 기기

3.4. 구현

소스코드 경로 및 폴더 구성은 아래와 같다. (보충 설명 필요)

경로 : Source_code

4. 프로젝트 결과

4.1. 연구 결과

이번 프로젝트에서 설계 구조상 가장 중요한 점은 두 스마트 장치간 Bluetooth 통신을 통하여 어플리케이션에 최적화된 프로토콜을 통하여 데이터를 최소화하여 최대한 빠른 통신을 하는 것이며, 음악 재생할 때 실제 악기와 비슷한 구조로 인터페이스를 제공하여 실감나게 만드는 것이다.

4.1.1. 연주 속도에 따른 소리 크기 알고리즘의 활용 : Stroke Activity 의 기능적 요소

기존에 상용되는 기타 연주 프로그램은 일반적으로 스트로크 세기와 상관없이 연주되는 소리의 음량이 일정하다. 하지만 실제 기타는 이와 달리 사용자의 스트로크 세기의 비례하여 소리의 음량이 달라진다. 따라서 스트로크 디바이스에서 사용자의 터치가 클수록 소리의 음량이 커진다는 사실을 사용하여 터치 속도를 계산한다. 터치 속도는 (속도 = 거리/시간)을 사용하여 스트로크 디바이스에 적용한다.

그림 1 에서와 같이 스트로크 디바이스는 처음 사용자가 눌렀던 포인터 위치를 기억하고, 마지막으로 손을 떼었던 위치를 기억하여 거리를 계산한다. 그리고 스트로크한 시간을 기억하여 시간 값에 대입한다. 그 후 터치 속도를 계산한다.

[그림 16] 스트로크 디바이스에서의 터치 속도 계산

4.1.2. 새로이 제시한 통신 기술의 활용 : Bluetooth 통신시 최적화된 프로토콜

실제 기타는 코드를 잡고 스트로크하면 시간이 지연되지 않고 소리가 연주된다. 이처럼 스트로크 디바이스에서도 코드 디바이스에게 코드를 전송 받아 소리가 재생될 때 소리가 지연되지 않아야 한다. 따라서 이 연구에서 코드 데이터 사이즈를 줄이기 위해 별도의 프로토콜을 설계하였다.

무음 (xxxxxx)	잡고있는줄 (xxxxxx)	1번째 줄 (xxxxx)	2번째 줄 (xxxxx)
3번째 줄 (xxxxx)	4번째 줄 (xxxxx)	5번째 줄 (xxxxx)	6번째 줄 (xxxxx)

[그림 17] 코드 송수신 프로토콜

제안하는 프로토콜은 그림 17 과 같다. 기타는 총 6 줄이고 마디를 최대 20 개라 가정한다. X 는 1bit 이고 X 가 6 개일 때 6bit 가 된다. 먼저 무음은 무음인 줄을 1 로 표시한다. 1, 2 번째 줄이 무음이라면 데이터는 110000 이 된다.

잡고 있는 줄도 무음과 같이 데이터가 적용된다. 만약 3, 4 번째 줄을 잡고 있다면 데이터는 001100 으로 표시된다. 각 n 번째 줄 데이터는 각 줄마다 어떤 마디를 잡고 있는지 표시한다. 최대 마디 수를 20 개로 가정하고 5bit 로 정의한다. 이 때 잡고 있지 않을 줄의 데이터는 생략한다. 예를 들어 1, 2 번째 줄이 무음이고 3 번째 줄의 2 번째 마디를 잡고 있고, 4 번째 줄의 3 번째 마디를 잡고 있다면 전송되는 데이터는 아래와 같다.

<110000 , 001100, 00010, 00011>

다시 설명하면, 110000(무음), 001100(잡고있는 줄), 00010(3 번 째 줄 2 번 마디), 00011(4 번 째 줄 3 번 마디) 이다. 이처럼 데이터를 전송 할 때 가장 큰 데이터가 42bit 를 넘지 않으므로 6byte 를 넘지 않는다. 6byte 를 전송하였을 때 평균적으로 200ms 의 시간이 소요되며 코드를 잡으며 연주하는 상황에서는 충분한 시간이다.

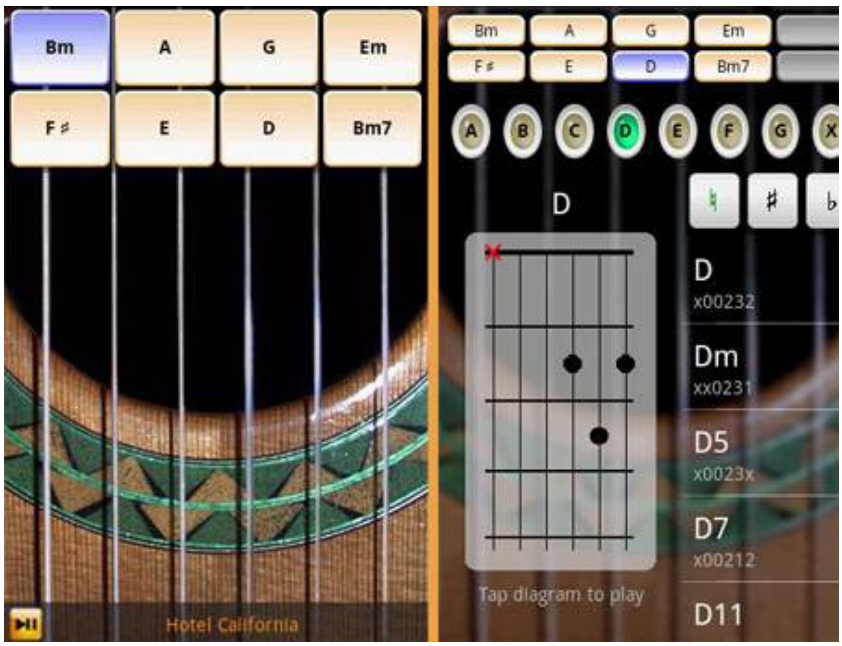
4.1.3. 실제 비슷한 인터페이스 제공 : App 의 시각적 요소

실제 기타와 비슷한 인터페이스를 제공하여 사용성에 있어 좀더 효과를 높인다.

[그림 18] 어플리케이션 구상도

4.2. 성능평가

프로젝트 설계 기간 중 자료를 구하는 기간 중 Google Play 에서 제공된 Guitar : Solo App. 이 있어 이를 대상으로 비교해 본다.



[그림 19] SOLO 어플리케이션

구글플레이에서 가장 인기가 많은 기타 어플리케이션이다. 이 solo 기타 어플리케이션은 직접 기타처럼 사용하기 보다는 정해진 코드를 선택한 후 기타화면에서 기타 줄을 터치하는 화면으로 기타소리의 재생을 들려주고 있다. 좋은 소리와 정확한 코드의 소리를 가지고 있어 소리를 재생해 보는 연습으로는 알맞지만 실제 기타를 치는 느낌은 조금 미약하다.

그래서, 본 연구결과에서 제시하는 작품에서는 기타를 직접 연주하는 느낌을 강하게 하기 위하여 블루투스 통신을 활용한 두 대의 장치를 이용하여 코드와 연주하는 부분을 달리 만들어 주었고, 세기에 따른 소리의 크기를 조절하기 위하여 소리세기에 관련된 알고리즘을 제시하여 좀더 감성적인 느낌을 전달해 주었다.

5. 결론

5.1. 기대효과

스마트폰을 통해 실제 기타와 같은 연주를 할 수 있다. 1 차 목표는 사용자가 실제 기타가 없을 때에도 자신의 장치와 타인의 장치를 사용하여 언제 어디서든 실제와 같은 연주를 할 수 있다. 혹은 장치가 두 개를 가지고 있는 사용자는 혼자만으로도 즉시 연주가 가능하다. 이 프로그램의 통신 방법으로 Blue Tooth 를 사용하기 때문에 사용자가 3G 혹은 LTE 가 제공되지 않은 지역에서도 장치만 있다면 서로 데이터를 주고 받으며 연주한다. 또한 사용자에게 하나의 장치에서 하나의 사운드만 제공하는 것이 아니라 여러 종류의 사운드 모드를 제공하기 때문에 사용자는 조금 더 다양한 연주(여러 종류의 sound 연주)를 할 수 있다.

또한 전문적인 사용자(Musician)들은 작곡을 할 때 하나의 장치를 사용하지 않고 여러 장치의 소리들을 사용한다. 그렇기에 이 연구는 하나의 장치에서 여러 종류의 음들을 연주하며 체험할 수 있기 때문에 작곡의 많은 도움이 된다. 그래서 여러 장치를 사야 하는 비용적인 측면을 많이 감소할 수 있다. 그리고 사용자가 여러 장치를 연결하며 작곡할 때 생기는 시간 적인 비용 문제도 많이 감소할 수 있다.

5.2. 추후 연구 방향

추 후에 현재는 안드로이드 (Andoroid) 장치들만 연결 할 수 있는 문제를 극복하여, 여러 Flatform 에서도 작동 가능 하게 한다. 예를 들어 IOS, Blackberry 등 다양한 사용자들에게 제공 한다. 모든 Flatform 에서 작동이 가능 된다면 안드로이드-안드로이드 혹은 아이폰-아이폰과 같이 같은 플랫폼끼리만 통신이 되어 연주를 하는 것이 아니라, 서로 다른 Flatform 끼리도 통신이 되어 연주가 가능하게 한다. 그리고 현재 프로그램에 사용자가 연주했던 음들을 TAB 악보로 바로 볼 수 있게 하여 작곡을 조금도 손쉽게 할 수 있는 측면을 고려한다. 혹은 그 반대로 Tab 악보를 인식하여 화면에 출력하여 교육용 게임 등으로도 발전시킬 수 있다.

6. 참고문헌

[1] 정재곤 저, 안드로이드 앱 프로그래밍

[2] 윤성우 저, 난 정말 JAVA 를 공부한 적이 없다구요

[3] 남궁성 저, Java 의 정석

[4] Goding Caveman, Guitar : Solo(Google Play)

[5] NETTuno, Virtual Guitar(Google Play)

[6] Android Reference: http://developer.android.com/