신디사이저와 IoT  
중간보고서

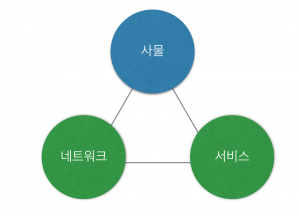
컴퓨터 시스템 I  
지도 교수 : 김호원 교수님  
하프시코드  
201224540 조용래  
200824499 이용석

목차

1. 요구 조건 및 제약 사항
2. 설계 상세  
   2.1 MIDI(Musical Instrument Digital Interface)  
   2.2 신디사이저 구성  
   2.3 통신 구조  
   2.4 SD 카드 & 파일 시스템 제어  
   2.5 이더넷  
   2.6 Node js를 사용한 서버 구축, 웹앱 개발
3. 과제 추진 계획
4. 구성원 별 진척도

**1. 요구 조건 및 제약 사항**

먼저 착수보고서에서의 홈 엔터테인먼트 로봇에서 신디사이저와 IoT로의 주제를 변경한 이유에 대해 간략히 설명하겠다. 로봇 엔터테인먼트의 구성 요소는 다양하다. 감정 표현하기, 놀아주기, 간단한 게임하기와 같은 것들이 있다. 당연히 이 모든 것들을 소프트웨어적으로 손수 구현하는 것은 매우 복잡하고 어렵다. 하드웨어 쪽도 마찬가지다. 그래서 단일한 기능을 택하여 그것을 중점적으로 개발시켜 나가야 한다. 음악 연주를 그 주제로 정하였고 구체적으로는 신디사이저를 만들어 보기로 하였다.  
  
**그렇다면 왜 이 주제를 정하였는가?**  
아두이노는 본래 예술가들에게 미디어 아트를 가르칠 명목으로 탄생되었다. 마시모 밴지, 톰 아이고와 같은 아두이노 초창기 멤버이자 피지컬 컴퓨팅 선구자들은 컴퓨터에 문외한인 미대 학생들에게 최대한 쉽고 편리하게 프로그래밍을 가르치길 원했고 그들이 아이디어 구현할 때, 컴퓨터 공학의 기술적 어려움을 최소화 시키려고 노력하였다. 그래서 아두이노 관련된 프로그래밍 환경이나 하드웨어적 요소들은 간단하게 사용할 수 있도록 만들어진 것들이 많다. 예를 들어, 이더넷 통신은 통신 구조에 대해 전문적인 지식이 없어도 기본으로 제공되는 라이브러리와 간단한 원리만 알면 초보자들도 쉽게 따라할 수 있도록 되어 있다. 모터나 센서 제어는 말할 것도 없다. 무엇보다 커뮤니티가 어마어마하고 하드웨어 부품도 값 싸고 다양한 것들이 많은게 큰 장점이다.  
그 중에서 특히 아두이노를 악기나 악기를 제어하는 용도로 사용하는 경우도 흔하다. 스피커, MP3 플레이어 쉴드, MIDI 쉴드, 시퀀서와 같은 음악 연주를 위한 제품들과 오픈 소스로 쓰여진 MIDI 라이브러리도 있다. 따라서 음악 전용 하드웨어들을 조립하여 악기를 구성하는 것은 비록 컴퓨터 공학을 전공하고 있는 학생이지만 이러한 지식들을 이용하여 음악을 이해하고 전자 악기를 만들 수 있다.   
물론 악기 하나만 뚝딱 만들고 끝내자는 것은 아니다. 악기 하나만 만들고 끝내버리면 기존에 제작된 신디사이저와의 차이점을 부각할 수 없고(당연히 상업용 신디사이저와의 순수 음악적 기능과 품질은 비교하기 어렵지만) 단순 지적 유희만으로 졸업 과제를 즐길 수는 없기 때문이다.  
현재 컴퓨터 업계에 불고 있는 유망한 키워드는 IoT이다. 클라우드나 로봇, 인공 지능, 스마트 카 등이 있지만 악기에 가장 색깔이 잘 들어맞는 단어는 단연 IoT일 것이다. IoT는 정의가 다양하지만 사물, 네트워크, 서비스로 요악된다. 악기라는 사물에 통신 기능을 부여하여 악기와 서버, 스마트 폰 간에 연결을 시도할 것이다. 물론 하드웨어적으로 센서 달고 네트워크 기능을 추가한다고 하여 IoT라고는 할 수는 없다. 이것들로 유용한 서비스를 만들어 내야 한다. 그래서 스마트 폰을 통해 악기를 제어할 수 있게 한다. 스마트 폰과 악기 간의 세부적인 인터페이스는 아직 정하지 않았지만 웹 앱을 기반으로 하고 사용자가 재생할 수 있는 곡 리스트를 보여주거나 녹음, 원격 악기 연주 등을 기본 기능으로 정할 것이다.

정리하자면, 요구 사항은 신디사이저 제작과 더불어 이에 IoT를 넣는 것이다. 제약 사항은 신디사이저의 퀄리티는 상업용에 비해 제한될 수 밖에 없겠지만 사용자와의 다양한 인터페이스 구현과 통신 기능을 만들어 차별화를 시킬 예정이다. IoT를 보는 또 다른 관점은 디바이스, 서버, 앱이다. 단순 통신을 넘어 스마트 폰이라는 디바이스로 앱을 실행하여 흔히 악기로 수행할 수 있는 기능 또한 차별화의 요소이다.

<그림 1> IoT의 3대 요소

**2. 설계 상세**

**2.1 MIDI(Musical Instrument Digital Interface, 이하 미디)**

**MIDI**(미디)는 악기 디지털 인터페이스(Musical Instrument Digital Interface)를 줄인 말로 [전자 악기](https://ko.wikipedia.org/wiki/%EC%A0%84%EC%9E%90_%EC%95%85%EA%B8%B0)끼리 디지털 신호를 주고 받기 위해 각 신호를 규칙화한 일종의 규약이다. 다시 말해 악기와 컴퓨터, 악기와 악기끼리 주고받을 수 있는 언어와 통로의 신호 체계 표준이라 할 수 있다. 어떤 전자 악기(건반, [신서사이저](https://ko.wikipedia.org/wiki/%EC%8B%A0%EC%84%9C%EC%82%AC%EC%9D%B4%EC%A0%80), 모듈 등)가 이 표준에 따라 만들어졌다면, 그 전자 악기가 미디를 지원한다고 할 수 있다.(위키)  
채널 보이스 메시지(Channel Voice Messages)는 미디 컨트롤러가 명령을 보내는 방식이다. 뒤에서 다시 설명하겠지만, 이번 과제에서는 아두이노 메가가 미디 컨트롤러 역할을 담당하고 미디음을 합성시켜 주는 사운드 모듈(또는 신디사이저)에 채널 보이스 메시지를 전송한다. 메시지 구성은 다음과 같다.

<그림 2> 채널 보이스 메시지의 구성

명령 바이트

데이터 바이트

데이터 바이트

8 bits

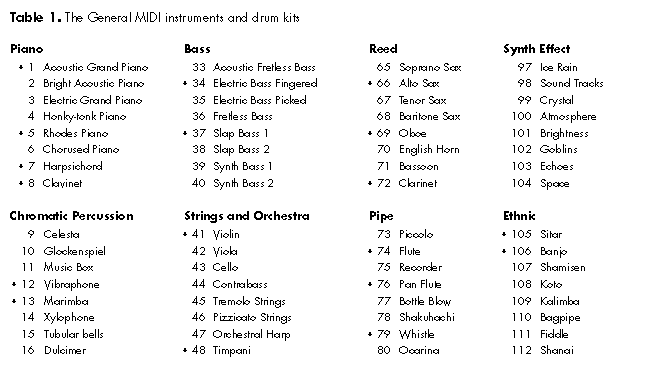
7 bits

7 bits

명령 바이트로 올 수 있는 것은 음을 키거나 끄고, 특정 채널의 악기를 변경하거나, 전체 소리를 늘리거나 줄이거나 하는 등의 연주에 대한 제어 신호를 담당하고 뒤의 두 개의 7비트 데이터 바이트는 명령 바이트에 따른 추가적인 정보를 포함한다. 다음은 몇가지 예시이다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Status Byte** | **Data Byte 1** | **Data Byte 2** | **Message** | **Legend** |
| 1000nnnn | 0kkkkkkk | 0vvvvvvv | Note Off | n=channel\* k=key # 0-127(60=middle C) v=velocity (0-127) |
| 1001nnnn | 0kkkkkkk | 0vvvvvvv | Note On | n=channel k=key # 0-127(60=middle C) v=velocity (0-127) |
| 1010nnnn | 0kkkkkkk | 0ppppppp | Poly Key Pressure | n=channel k=key # 0-127(60=middle C) p=pressure (0-127) |
| 1011nnnn | 0ccccccc | 0vvvvvvv | Controller Change | n=channel c=controller v=controller value(0-127) |
| 1100nnnn | 0ppppppp | [none] | Program Change | n=channel p=preset number (0-127) |
| 1101nnnn | 0ppppppp | [none] | Channel Pressure | n=channel p=pressure (0-127) |
| 1110nnnn | 0ccccccc | 0fffffff | Pitch Bend | n=channel c=coarse f=fine (c+f = 14-bit resolution) |

<테이블 1> 채널 보이스 메시지

위의 명령이 신디사이저에 전달되면 해당하는 명령을 수행한다. 보통 미디는 16개의 채널로 구성되고 127가지의 악기를 지원한다. 그림 3은 신디사이저가 명령을 받고 출력하기까지의 과정을 간략하게 나타낸다.  
  


<그림 3> 메시지 전달 과정

악기 뱅크

채널(1~16)

피아노

바이올린

드럼

…

기타

1

3

2

…

16

앰프

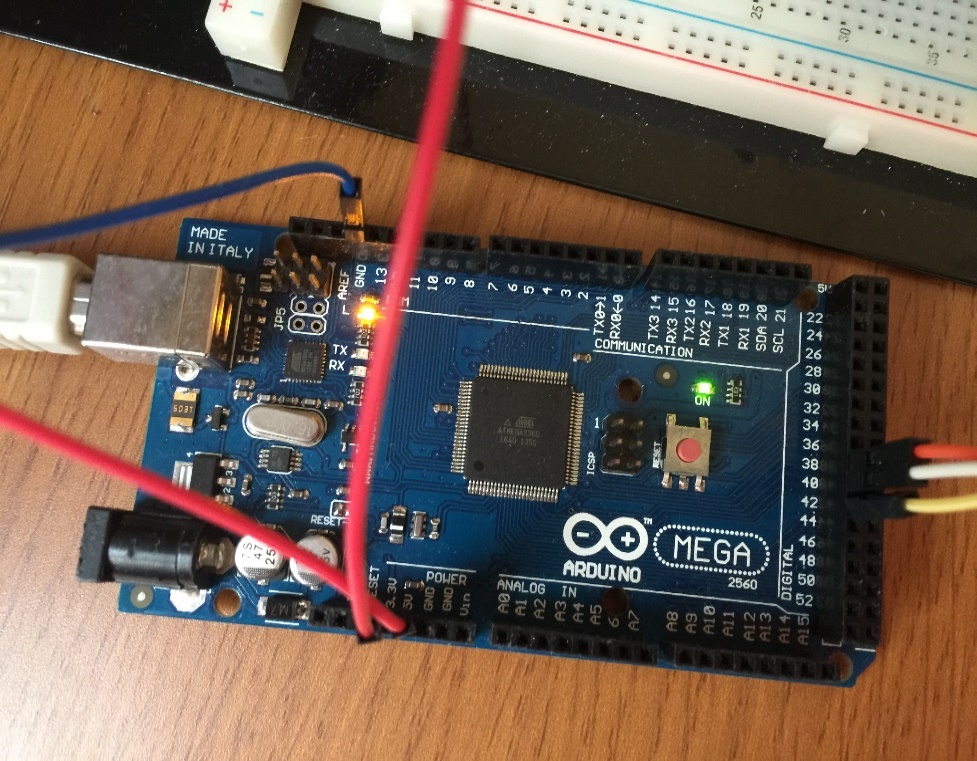
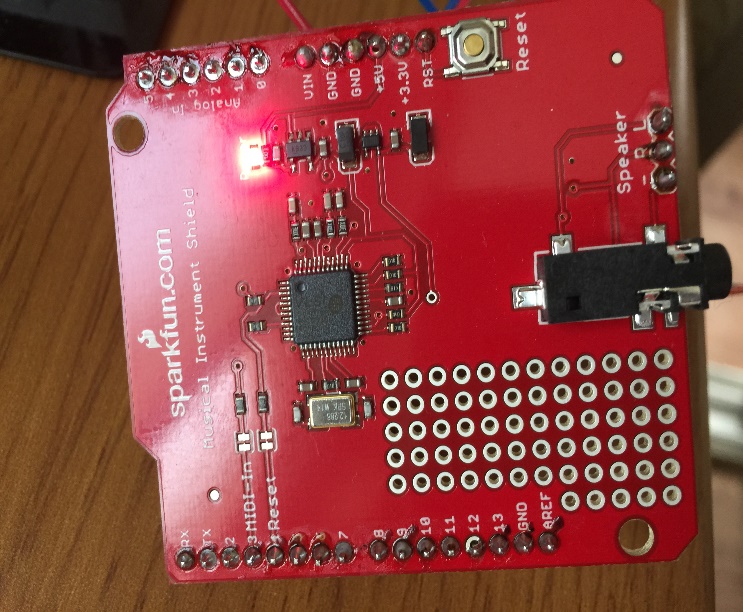
스피커

<그림 4> 미디 악기 목록(1~112)

미디는 비동기식 직렬 접속 방식이고 통신 속도는 31250 bps이다.   
이렇게 많은 악기를 다룰 때 필요한 소프트웨어는 다행스럽게도 오픈 소스로 이미 작성되어 있다. 아두이노 미디 라이브러리(<http://playground.arduino.cc/Main/MIDILibrary>)는 C++로 작성되었으며 모든 악기와 제어 신호를 다루기 위한 함수들이 작성되어 있다.

**신디사이저 구성**

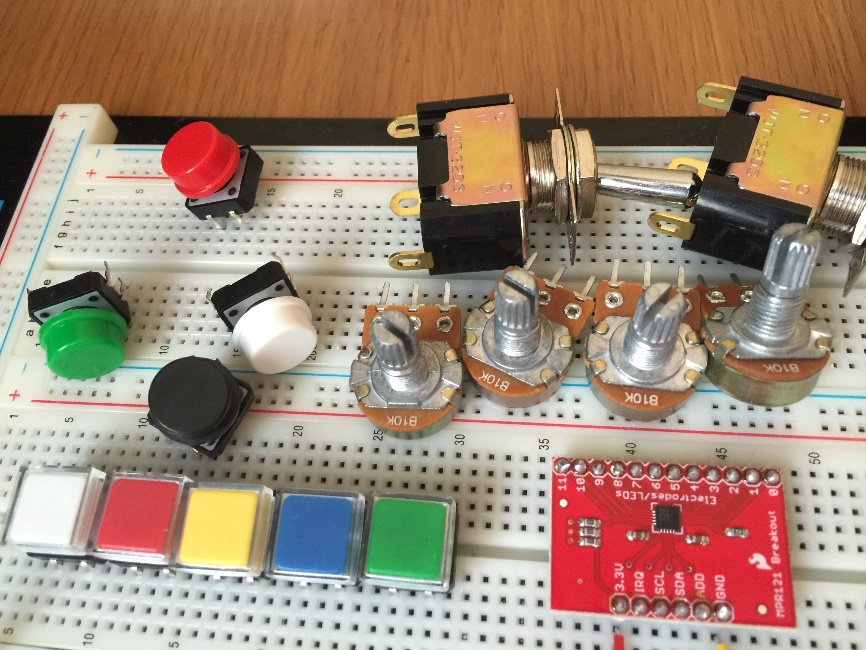
미디 악기는 크게 3파트로 구성된다. 미디 명령을 내리는 미디 컨트롤러와 이 메시지를 받아 음을 합성해주는 신디사이저, 그리고 음을 출력해주는 스피커이다. 앞서 신디사이저는 사용자와 입력을 받는 버튼들이 매우 많다. 추가적으로 센서도 있으니 입력 핀이 많이 필요하다. 그래서 아두이노 메가(그림 7)를 미디 컨트롤러로 사용하였다. 신디사이저는 SparkFun에서 제작된 미디 쉴드를 사용하였다. 메가에서 미디 핀으로 연결된 이 쉴드에 명령을 보내면 쉴드에 연결된 스피커에서 소리가 난다. <그림 7>의 미디 쉴드는 3번 핀을 미디 입력으로 사용하고 있고 오디오 잭이 있어서 헤드폰이나 스피커를 간단하게 연결할 수 있다. 잭이 없는 경우는 그 오른쪽에 스피커를 연결할 수 있는 핀이 마련되어 있다. 상세 스펙은 지면상 생략한다. <그림 9>는 신디사이저에 붙일 각종 인터페이스 장치들이다. 택트 스위치, 토글 스위치, 가변 저항 노브, 정전식 터치 센서 등이 있다. 사진에는 나와 있지 않지만 슬라이더나, 다른 버튼들의 구성과 센서도 얼마든지 가능하다. <그림 10>은 신디사이저의 시각적 출력에 해당한다. 현재 연주하고 있는 음악이나 볼륨, 사용 중인 채널, 통신 관련 메시지들과 음에 따른 비주얼적 효과를 넣을 것이다.   
아직 몸체 제작으로는 들어가지 않았다. 디자인 박스나 아크릴 판 또는 원목 같은 소재들을 활용하여 제작할 것이다.



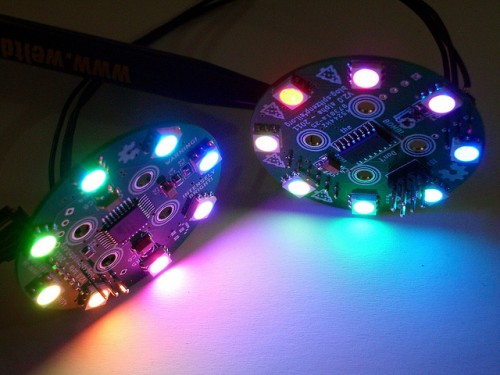
<그림 8> 스피커

<그림 7> 스파크 펀 미디 쉴드

<그림 5> 미디 컨트롤러로 사용될 아두이노 메가

****

<그림 10> 신디사이저의 시각적 출력 인터페이스



<그림 9> 신디사이저의 인터페이스

**통신 구조**

아두이노 메가에는 통신 기능이 없다. 그래서 우노에다 이더넷 쉴드 달아서 우노와 메가 간의 통신을 구축한다. 이 둘 사이에는 직렬 주변기기 인터페이스(SPI : Serial Peripheral Interface Bus)로 연결한다. 우노는 TCP/IP를 통해 node js로 구축된 서버와 연결된다. SPI를 통해 우노는 서버 쪽에서의 요청이 들어오면 이에 맞는 적절한 명령을 신디사이저에 보내기도 하고 반대로 신디사이저에서 사용자가 통신 관련 기능을 요구하면 명령을 받고 서버 쪽으로 데이터를 보내기도 한다. 이들 간의 상세한 인터페이스적 사항들은 구체적으로 나오지는 않았지만 흔히 예상할 수 있을 만한 기능들을 위주로 한다. (사진 속의 신디사이저는 이미 DIY로 제작된 사례이다.)  
서버 개발 플랫폼으로 node js를 사용하는 이유는 node js에 이미 아두이노를 제어하기 위한 모듈이 있다. duino와 firmata와 같은 것들이다. 이미 node js는 아두이노와 연동해 IoT 개발 플랫폼으로 상당히 주목을 받고 있기도 하고 모듈 형태의 확장 가능한 플랫폼을 지향하여 개발하기에 용이하다. 자바스크립트 언어를 사용하기 때문에 처음부터 다시 배울 필요가 없는 것도 장점이다.

<그림 11> 통신 구조



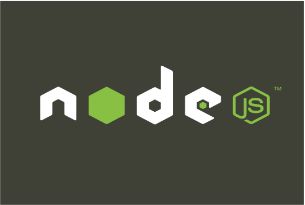
아두이노 우노 + 이더넷 쉴드 with SD카드

SPI

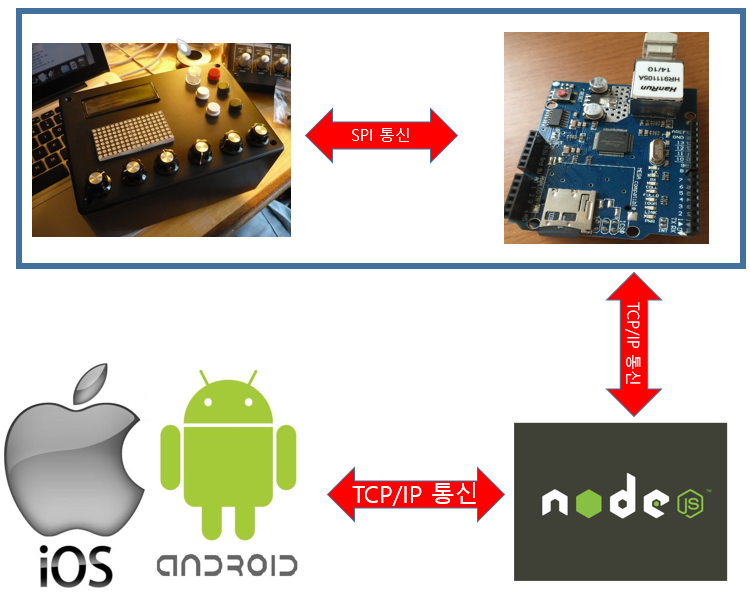
TCP/IP



서버



<그림 12>는 이번 졸업 과제의 전체적인 통신 구조를 나타낸다. <그림 11>에서 스마트 폰이 추가 되었다. 스마트 폰에서 원격으로 신디사이저를 제어할 수 있도록 웹 앱을 작성할 것이다. 웹 앱은 노트북이나 라즈베리파이에 구현된 서버를 통해 신디사이저에 접근할 수 있다. 웹 앱 자체에 대한 자세한 구성은 아직까지 미정이다.

<그림 12> 전체 통신 구조

**SD카드, 이더넷**

아두이노에서 SD카드 제어를 위한 라이브러리가 제공된다. FAT16, FAT32 파일 시스템을 SD카드에서 사용할 수 있다. SPI통신를 이용하며 이더넷 쉴드를 사용하는 우노의 디지털 핀 11, 12, 13번을 사용한다. 이더넷 역시 관련 라이브러리가 있고 SPI통신을 이용하며 핀 11, 12, 13을 사용한다.  
SD카드에는 주로 사용자가 재생할 곡이나 서버에서 받아온 음악 파일, 녹음할 파일을 저장할 것이다. 우노는 이더넷 쉴드를 통해 서버쪽과 TCP/IP 통신을 하게 된다.

**Node js로 개발한 서버, 웹앱**

Node js로 아직까지 제대로된 서버를 만들진 못 했지만 간단하게 나마 환경을 만들었고 집에서 서버 테스트를 해보았다. 모듈 형식으로 되어 있고 아두이노 제어 모듈(duino, firmata)가 있다. 이를 통해 서버쪽에서 아두이노 제어를 좀 더 쉽게 할 수 있도록 해준다. 서버 컴퓨터는 노트북으로 사용할 가능성이 높고 나중에 라즈베리파이에 설치해 볼 수도 있을 것 같다.  
스마트 폰에서의 악기 제어를 위한 앱은 웹으로 작성한다. 안드로이드나 iOS전용 앱을 만들기 보다 웹 앱을 만드는 편이 호환성에서 유리하기 때문이다. 그리고 iOS개발은 비용이 꽤나 크다.

**3. 과제 추진 계획**

8월 – 신디사이저 제작을 위한 갖가지 지식들을 습득하고 몸체를 만들어 낸다. 라이브러리 사용과, 미디 쉴드 사용에 익숙해지고, 시연을 위해 연주해보기도 하고, 간단한 드럼 비트나 음악 패턴을 만들기도 해보고, 시퀀서, MP3 쉴드와 같은 사운드 모듈들과 스피커, 앰프 등을 추가로 테스트 해 보면서 사운드를 조율해 본다. 그리고 이더넷, SD카드도 제어해 본다.  
8월 중 후반부터는 완성된 신디사이저를 테스트 해보고 몸체 수정이 필요한 부분을 고치면서 동시에 웹 쪽 개발에도 진행한다.  
9월부터는 서버, 웹앱, 안드로이드 앱 개발을 본격적으로 시작하여 9월 중순에 끝내고 그 이후부터는 테스팅을 위주로 한다.

**4. 구성원 별 진척도**

|  |  |
| --- | --- |
| 이름(학번) |  |
| 조용래(201224540) | 신디사이저 제작 중, 관련 부품들 모아서 조립해보며 음들 테스트 해보고 라이브러리 사용 중임. 주로 음악 쪽에 신경 쓰는 중. |
| 이용석(200824499) | 안드로이드 앱 개발 착수. SD카드와 이더넷 제어에 대한 자료 조사와 더불어 개발 환경을 만들고 시작. |