BACHELOR'S THESIS

Developing Arduino Based Synthesizer with interactive user interface and communication

Yongrae Jo(조 용 래)

Department of Computer Science and Engineering of
Pusan National University

TEAM: Harpsichord

STUDENT #: 201224540

SUBDIVISION: Computer System Ⅰ

ADVISOR: Professor Howon Kim

October 2015

목차

Abstract

In this paper, we aim to develop a synthesizer capable of Ethernet communication and interactive user interface using Arduino platform. During the process of development, we use and reference many source codes from various open source community. And we represent how the computer engineering's technology incorporates a variety of hardware components, producing a synthesizer with some application of musical knowledge.

List of Figures

List of Tables

1. 서론

컴퓨터 공학과 전자 공작 기술의 지식을 활용하여 아두이노 기반의 통신 기능이 있는 전자 악기(신디사이저)를 만든다.

전자 악기의 제작을 주제로 선정한 이유는 대학교에서 배웠던 컴퓨터 공학적 지식을 종합적으로 활용할 수 있다는 판단이 들었기 때문이다. 컴퓨터 공학은 다른 분야에 곧잘 적용되기 쉽다. 여기서는 전자 공작과 음악 분야에 컴퓨터 기술을 적용하여 악기를 제작한다.

컴퓨터는 뿌리부터 수학이고 음악 역시 마찬가지이다. 피아노 건반만 봐도 하나하나의 건반이 모두 특정 주파수의 음을 내는 것이고, 음들의 지속 시간, 악보의 표기법 역시 수학의 형식적 요소를 빼다 닮은 것이다. 컴퓨터는 수학적 지식을 표현하는데 발군의 능력을 자랑한다. 수학적으로 표현될 수 있는 음악 역시 컴퓨터로 완전하게 표현될 수 있다. 그러한 표현 방법 중 하나인 미디 프로토콜을 본 프로젝트에서 핵심 기술로 사용한다.

그렇다고 비중이 음악 쪽에만 치중될 수는 없다. 사용자 인터페이스를 구현하기 위해서는 각종 전자 공작 테크닉들이 많이 필요하기 때문이다. 이런 지식들은 회로 이론의 어려운 이론들이 아니라 간단한 옴의 법칙이나, 공구 도구들의 활용, 배선, 납땜, 그리고 기타의 각종 재료들의 활용 능력 등이 요구된다.

4년 간 컴퓨터 공학을 전공하면서 대개 스크린 속에서 비춰진 텍스트와 정보 조작에 대부분의 시간을 쏟았다. 하지만 임베디드 보드와 피지컬 컴퓨팅을 접하면서 모니터 밖에서 이뤄지는 사람과 컴퓨터와의 감각적이고 적극적인 상호작용에 흥미를 느꼈고, 이를 구현하는 방향으로 졸업 과제의 초점을 맞추었다.

이 보고서는 총 11장으로 구성되어 있다. 1장은 서론으로 과제를 선정한 이유에 대해 간략하게 소개한다. 2장에서는 과제 진행 도중 설계 상 변경 사항 및 제약 사항에 대해 언급한다. 이후의 3~6장은 이 보고서의 뼈대에 해당한다. 3장은 악기를 구성하는 하드웨어 부품들에 대해 설명하고 이를 어떻게 활용할 것인가 논의한다. 4장에서는 음악의 기본 개념을 컴퓨터로 표현하는 방법을 다룬다. 미디 프로토콜과 더불어 프로그램 상에서 음악을 다루는 방식을 다룬다. 5장은 사용자가 악기로 할 수 있는 기능들을 나열한다. 6장은 악기와 웹 서버, 그리고 사용자 간의 형성되는 통신 구조를 설명한다. 7장은 이전 장들을 전체적으로 훑으며 본 프로젝트를 다시 간단하게 언급한다. 8장은 과제 진행 시 도움이 되었던 자료들을 나열하고 9장은 사용한 부품들의 리스트와 부품들의 주된 구입처를 보여준다. 10장과 11장은 필자가 졸업 과제를 수행 후 느꼈던 점과 과제를 무사히 끝낼 수 있게 도움을 준 분들에게 감사의 말을 전하는 장으로 구성된다.

2. 설계 상 변경 및 제약 사항[1]

수정 사항

General MIDI의 일부만 지원

먼저 소리를 생성하는 미디 보드의 제약에 대해 먼저 설명을 하겠다. 미디 보드는 일반 미디 프로토콜(General MIDI)를 지원하지만 그 중에 일부만 지원한다. 예를 들어, 일반 미디 프로토콜은 는 포르타멘토나 트레몰로와 같은 음 변조 테크닉 명령을 명시하였지만**[8]** SparkFun에서 제작된 미디 보드(SparkFun Music Instrument Shield)는 이러한 명령들을 지원하지 않는다. **[9]** 이유는 미디 보드에 사용되는 칩(vs1053)이 미디 명령의 일부만 지원하기 때문인데 **[11]** 자세한 내용은 이후에 다시 다루겠다.

음악 전송 방식 수정

중간보고서를 재검토 후 '음악을 전송'한다는 개념이 마치 바이너리 음악 파일을 주고 받는 것처럼 들릴 수 있을 것 같다. 여기서 정정한다. 음악은 노트들의 나열이고 노트는 수치로 표현된다. 음악의 모든 것은 숫자와 형식으로 표현될 수 있고 전자 악기의 공통 프로토콜인 미디 언어로 재표현될 수 있다. 여기서는 미디가 음악을 어떻게 다루는지 이해하고 이를 바탕으로 음악을 노트들의 나열로 다루었다. (노트는 1~127사이의 정수 값이다.) 자료 구조도 이에 맞게 재설계 되었다.

SPI에서 I2C 통신으로 변경

중간보고서에는 입출력 보드(아두이노 메가)와 우노 보드와의 통신을 SPI라고 했는데 I2C 통신으로 수정한다. SPI는 SD카드와의 통신을 하기 위해 주로 쓰인다. SPI가 그래서 속도가 빠른 것이다. 메가에서는 SD카드를 음악 파일을 저장하고 읽거나 길다란 문자열들(악기 목록)을 읽기 위한 수단으로 쓰인다.

일반 스피커 사용

중간 보고서에는 작은 스피커 사진을 올려 놨었는데 진행해본 결과 일반 스피커나 헤드폰만 있어도 충분하다. 따라서 필요 없어서 제거하였다.

TCP Socket에서 Web Socket으로 변경

마지막으로 서버쪽에서의 통신은 우노나 클라이언트나 둘 다 TCP 통신으로 되어 있는데 좀 더 정확하게 말하면 우노 쪽과는 TCP 소켓 방식으로 통신을 하고 클라이언트 쪽으로는 웹 소켓 방식으로 통신한다. 클라이언트는 웹 페이지를 보여주는 브라우저를 대상으로 한다. (안드로이드/iOS 앱이 아니

[1] 필자가 다니는 학교의 졸업 논문은 3단계로 구성된다. 착수 보고서, 중간 보고서, 최종 보고서 순서로 제출한다. 2장은 중간 보고서 때 제안했던 기능들 중 일부를 변경한 내용을 서술한다.

다.)

참고 사항

이 작품 특성상 사용하는 부품 수가 매우 많고, 각 기능별 상세 구현의 수도 많다. 이들 전부를 상세하게 서술한다면 보고서 분량이 매우 길어지고 지루하기 때문에 세부 내용으로는 들어가지 않겠다.

3. 부품 별 설명과 활용 방안

졸업 과제의 대부분의 시간이 사실 하드웨어들을 짜맞추고 테스트 하고 갈아 끼우고 하는 곳에 많이 투자 되었다. 아두이노와 같은 임베디드 보드를 다루는 피지컬 컴퓨팅은 키보드만 두드리며 소프트웨어를 만들었던 필자에게 익숙하지 않은 분야였다. 전선을 피복하는 것도 처음에는 니퍼로 무식하게 힘만 쓰다 날려 버린 전선 조각이 한 두개가 아니다. 하지만 얼마 후에 와이어 스트리퍼와 전선 규격에 대한 이해가 선행된 이후에는 짧고 빠른 시간에 원하는 길이의 전선을 피복할 수 있게 되었고, 배선이나 납땜과 같이 익숙지 못 했던 기술들 또한 얼마간의 노하우가 생긴 후에는 꽤나 안정적으로 원하는 결과를 얻어낼 수 있었다.

하드웨어 구성은 구축 대상의 물리적 토대를 결정한다. 이후의 소프트웨어는 이런 토대 위에 구축되기 때문에 하드웨어 조각들의 모음과 구성 방식은 매우 중요하다. 제작하면서 이것저것 많이 쓰게 되었다. 먼저 전체적인 구성을 소개한 뒤 하나씩 차례대로 각 부품의 기능과 활용 방안을 간략하게 사진과 함께 설명한다.

전체 구성

신디사이저를 이루는 구성 요소는 Figure 1에 나타난다. 먼저 신디사이저는 전체적으로 계산을 위한 컴퓨터, 음악을 위한 하드웨어, 그리고 사용자 인터페이스로 구성된다. 기본적으로 입출력 처리와 음악적 정보를 처리할 수 있는 컴퓨터가 필요하다. 여기에는 악기와 통신을 위한 컴퓨터가 있다. 악기 컴퓨터는 사용자 인터페이스를 이루는 입출력 수단들과 통신 컴퓨터에서 넘어온 명령을 수행한다. 통신 컴퓨터는 이더넷 기반으로 서버와 TCP/IP 통신을 한다. 서버와 접속하고 웹에서 클라이언트가 넘겨 준 문자열을 파싱하여 악기 컴퓨터가 처리하기 좋게 가공한다.

소리 쪽은 미디 프로토콜을 지원하는 미디 보드와 스피커로 구성된다. 악기 컴퓨터가 미디 명령어 형식에 맞게 값을 미디 보드에 넘겨주면, 미디 보드의 출력이 스피커를 통해 나오게 된다. 이 부분이 소리가 합성되고 재생되는 부분이다.

사용자 인터페이스는 3부분으로 구성된다. 건반, 스위치, 그리고 디스플레이로 이루어져 있다. 건반은 2옥타브의 피아노 건반을 제공한다. 스위치는 사용자가 음을 합성하기 위해 각종 음색 파라미터를 조절할 수 있도록 되어 있다. 디스플레이는 음색의 여러 속성들의 수치 값들과 메뉴 등 사용자에게 도움이 되는 정보를 출력한다.

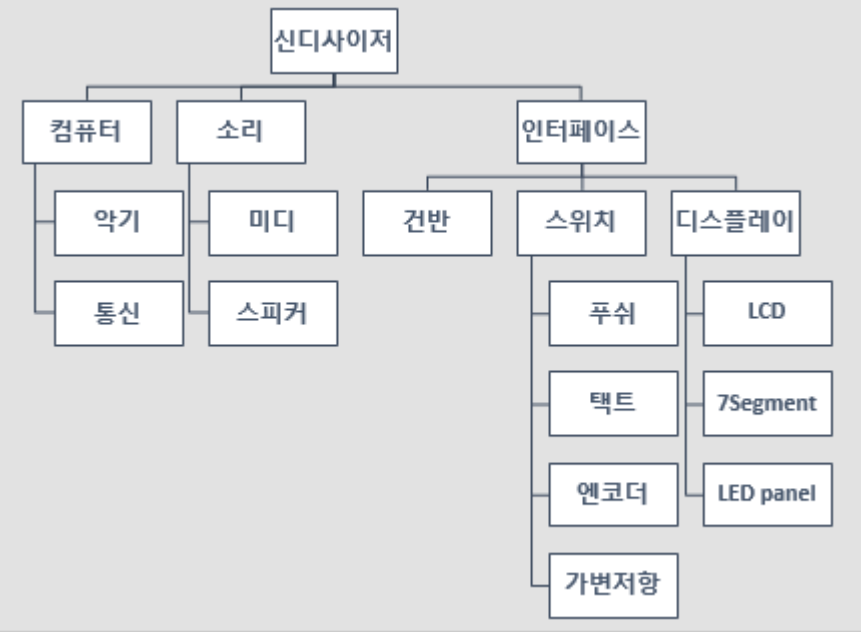


Figure 1 신디사이저의 구성

아두이노 우노 R3

아두이노 우노 보드는 악기와 서버와의 통신을 위해 사용된다. 구체적으로는, 서버 쪽에서 받은 JSON 형식의 문자열을 오픈 소스 ArduinoJSON을 사용하여 파싱하고 해당 값들을 I2C 방식으로 연결된 메가에게 넘겨 준다. 통신 구조에 대한 상세한 설명은 7장을 참고하라. Figure 1은 이더넷 쉴드가 부착된 아두이노 우노 보드이다.

한 가지 짚고 넘어갈 문제가 있다. 아두이노 메가 역시 이더넷 쉴드를 사용해 통신할 수 있는데, 왜 굳이 우노를 여분으로 사용하는가? 과제 초기 단계에서는 통신할 내용 자체가 적었기 때문에 이런 질문이 필요가 없었다. 하지만 TCP 통신을 하면서, 이더넷 라이브러리와 JSON형식의 문자열을 파싱하는 과정, 그리고 이후 추가적으로 통신 파트에 들어갈 기능과 같은 확장성을 고려하게 되었다. 악기는

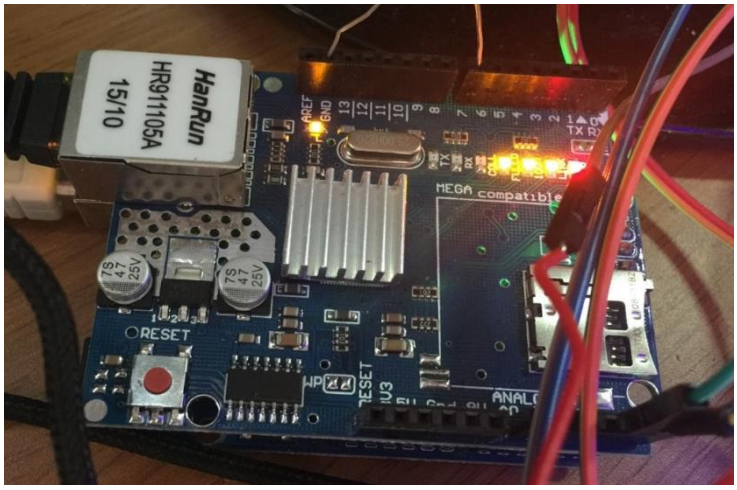


Figure 2 이더넷 쉴드가 부착된 아두이노 우노

악기만을 위한 서비스를 제공하고, 통신은 통신만 전용으로 다룸으로써 서로 간의 독립성을 유지하는 것이 좋다는 설계 상의 판단이 들었다. 이와 같이 레이어를 한 층 더 쌓는 것은 복잡성을 증대시키지만 그 만큼 구조는 견고해지기 마련이다.

아두이노 메가 2560

신디사이저를 한 번이라도 본 사람이라면 키보드 건반 위로 수 많은 노브들과 슬라이더, 버튼들과 LCD 등과 같이 사용자와 소통할 수 있는 수단이 다양하다는 것을 알 것이다. 이 뜻은 아두이노로 신디사이저를 제작하기 위해서는 그 만큼 많은 입출력 핀이 필요하다는 것이다. 우노의 핀 수는 적지만 메가 보드는 충분히 많다. [2]

우노를 사용하면, 부족한 핀 수를 충당해야 한다. 예를 들어, 멀티플렉서 모듈(MUX, 4 input to 16 output)을 사용 수도 있다. 하지만, MUX를 사용할 경우 달갑지 않은 배선을 추가로 해야하고 코드가 쓸데없이 복잡해진다. 다행스럽게도 메가 보드의 핀 수 만으로도 충분하다. 그리고 메가 보드는 입출력 핀 뿐만 아니라 메모리 용량 또한 우노에 비해 많기 때문에 좀 더 자유로운 프로그래밍이 가능하다.[3] 다음은 메가 보드 사진이다. 중간 보고서에 있는 사진과는 다르게 선들이 많이 보인다.

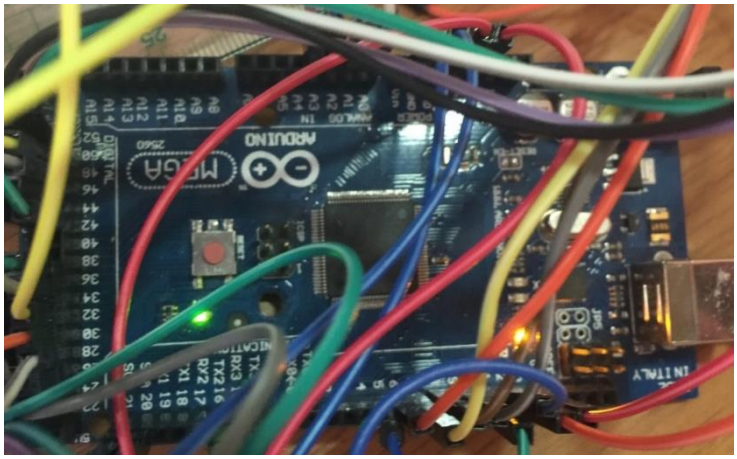


Figure 3 아두이노 메가

미디 보드

메가 보드가 악기에서의 두뇌 역할을 한다면 미디 보드는 손발 역할일 것이다. 악기에서 가장 중요한 소리를 내는 알고리즘을 내장한 칩을 가지고 있다. 서두에 일반 미디 명령 중 일부만 지원한다고

---

[2] https://www.arduino.cc/en/Main/ArduinoBoardUno#techspecs에 따르면, 아날로그 핀과 디지털 핀을 합쳐서 20개 밖에 안 된다. 반면, https://www.arduino.cc/en/Main/arduinoBoardMega에 따르면 메가 2560 보드는 아날로그 핀과 디지털 핀을 합쳐서 70개나 사용할 수 있다.

[3] 하지만 과도한 동적 할당이나 연속 메모리 할당 메커니즘을 사용하지 않는 것이 좋다. 이 부분에서 잠시 할 말이 있는데, 데스크톱에서와 임베디드 보드와의 결정적 차이점은 OS존재의 유무이다. 아두이노는 OS가 없다. 따라서 메모리 관리를 못 하기 때문에 빈번한 동적 할당은 프래그먼테이션 현상을 일으킨다. 그리고 이는 보드를 일순간에 마비시킬 수 있다. (참고로 필자는 100가지 이상이나 되는 악기 목록 이름 전부를 메모리에 동적으로 할당하려고 했지만, 에러가 생겨서 대체 왜 이러나 싶었지만 이 문제가 메모리 동적 할당이 문제라고 생각했고 이를 고정된 크기로 선언함으로써 해결하였다.)

했다. Figure 3은 그 목록을 보여준다. 신디사이저에서 자주 쓰이는 애프터터치나 포르타멘토 트레몰로와 같은 고급 명령어들은 지원하지 않는다. 기본적인 명령어들을 주로 지원한다. 노트 온/오프, 채널 볼륨, 음색 뱅크 선택이 있고 컨트롤 체인지 명령어로는 리버브나 팬 포트가 있다. 서스테누토나 홀드 명령어도 있다.

앞서 언급했듯이, 미디 보드에 내장된 칩은 VS1053b 이지만 스파크 펀에서 제조한 Musical Instrument Shield는 명세된 이 칩의 기능 중 미디 부분 일부만 사용 가능하도록 만들어 놓았다. 여기에서 제약 사항이 나온다.[4] 처음에는 미디 명령어 대다수를 지원하며 음 자체를 자유자재로 변조해보면서 주파수 변화의 매력을 제공하려고 했지만 안타깝게도 하드웨어 지원의 한계 때문에 제한된 기능들만 제공하게 된다는 점이다. [5]

Supported MIDI messages:

- meta: 0x51 : set tempo
- other meta: MidiMeta() called
- device control: 0x01 : master volume
- channel message: 0x80 note off, 0x90 note on, 0xc0 program, 0xe0 pitch wheel
- channel message 0xb0: parameter
  - 0x00: bank select (0 is default, 0x78 and 0x7f is drums, 0x79 melodic)
  - 0x06: RPN MSB: 0 = bend range, 2 = coarse tune
  - 0x07: channel volume
  - 0x0a: pan control
  - 0x0b: expression (changes volume)
  - 0x0c: effect control 1 (sets global reverb decay)
  - 0x26: RPN LSB: 0 = bend range
  - 0x40: hold1
  - 0x42: sustenuto
  - 0x5b effects level (channel reverb level)
  - 0x62,0x63,0x64,0x65: NRPN and RPN selects
  - 0x78: all sound off
  - 0x79: reset all controllers
  - 0x7b, 0x7c, 0x7d: all notes off

Figure 4 VS1053b 지원하는 미디 메시지

[4] 학과에서 요구하는 졸업 과제 보고서의 형식을 준수한다.

[5] 이 문제를 8월 초에 깨닫고 해외 사이트에서 미디만을 전용으로 다루는 CPU를 구매했었지만, 사용 방법이 생소하였고 시간 상의 제약 때문에 스킵하였다. 참고로 필자가 구입한 사이트는 http://www.highlyliquid.com/이다.

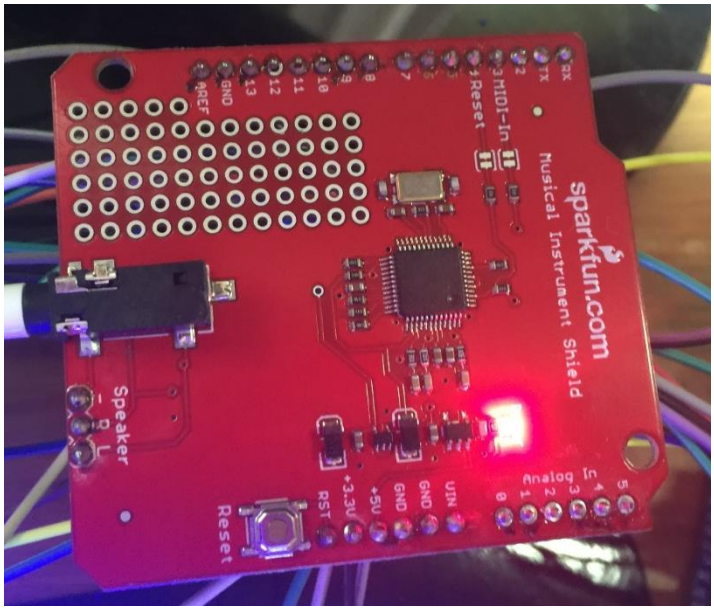


Figure 5 미디 보드

정전식 터치 센서(MPR121)

미디 보드도 이 센서가 없으면 키보드를 만들어 내지 못할 것이다. I2C 프로토콜로 연결되는 이 정전식 터치 센서는 총 12개의 정전식 접점을 가진다. 12라는 숫자는 피아노의 한 옥타브와 일치한다. 그래서 이 센서를 처음 사용했을 때 피아노 건반으로 연결하면 좋겠다는 생각이 들었다. MPR121을 제조한 여러 회사들이 여럿 있지만, 여기서는 Spark Fun에서 나온 보드를 사용했다.

MPR121은 I2C에서 0x5A, 0x5B, 0x5C, 0x5D를 주소 값으로 가질 수 있다. 초기에는 0x5A로 주소가 할당 되어 있다. MPR121을 여러 개 사용하기 위해서는 이를 바꿔야 한다. 이는 ADDR핀을 재설정하면 된다. ADDR핀은 처음에 GND에 연결되어 있다. 보드 뒷면을 보면 가느다란 선이 있는데 이를 칼로 그어 GND 쪽으로의 선을 잘라준 다음에 다른 핀으로 연결해야 I2C에서의 주소가 제대로 인식된다. 정전기 현상을 입력으로 받아 들이기 때문에 건반은 금속판이 적당하다. 피아노 건반 구축은 이후의 4.9에서 다룬다.

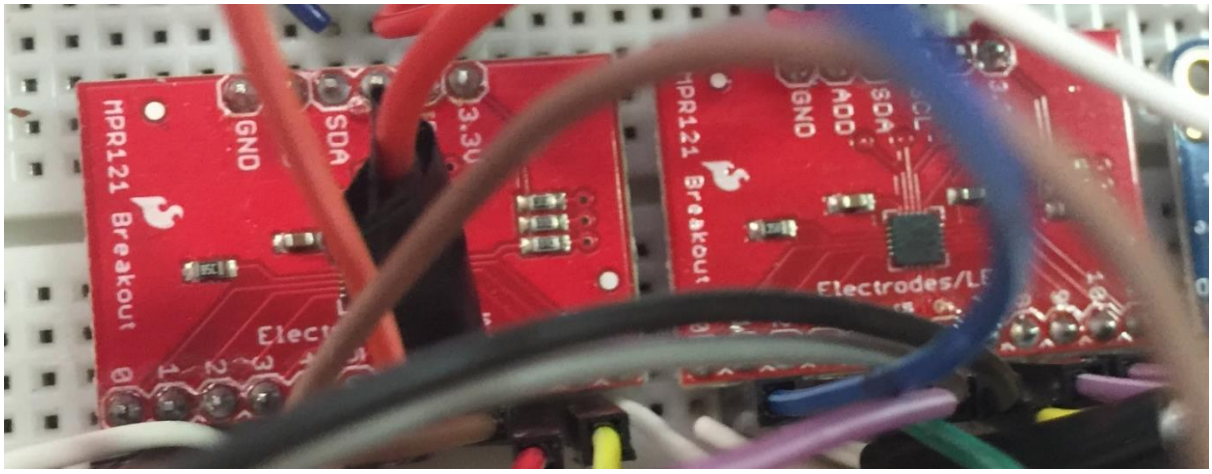


Figure 6 MPR121 정전식 터치 센서

Pin No.	Pin Name	Description
1	$\overline{IRQ}$	Active Low Open-drain Interrupt Output
2	SCL	I^2C Serial Clock
3	SDA	I^2C Serial Data
4	ADDR	I²C Slave Address Pin Selects. Connect to VSS, VDD, SDA, SCL to choose address 0x5A, 0x5B, 0x5C, 0x5D respectively.

Figure 7 MPR121 핀 할당

LCD, 7-segment

LCD나 7segment는 악기에서 디스플레이 역할을 한다. LCD에는 시스템 메시지나 건반 위치, 현재 선택된 마디나 방금 누른 건반 키 값, 현재 옥타브 따위를 표현한다. 세그먼트는 LCD 를 보조해주는 역할을 한다. 주로 방금 변화된 수치 값을 찍어낸다.

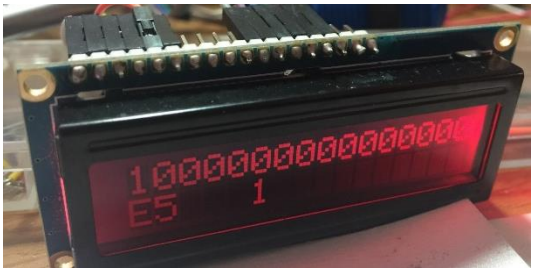


Figure 9 LCD

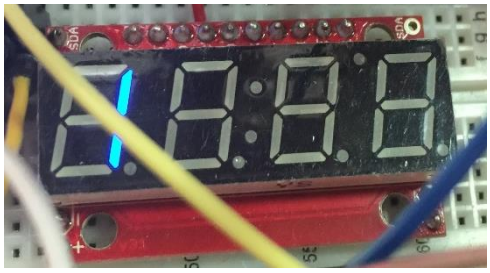


Figure 8 7-segment display

쉬프트 레지스터

메가 보드가 아무리 핀이 많다지만 32개나 되는 LED를 인심 좋게 내주기에는 부담스럽다. 다수의 입력 핀이 필요한 경우 앞서 언급한 먹스를 사용할 수 있지만, 출력 핀을 위해서는 디코더나 쉬프트 레지스터가 있다. 디코더는 배타적 출력을 내기 때문에 동시에 여러 LED를 점등해야할 상황일 때는 부적합하다. 쉬프트 레지스터는 시리얼 입력을 1바이트로 받는데 이때 1의 위치에 따라 해당 핀의 출력에 전류를 흘려 LED를 점등시킨다.

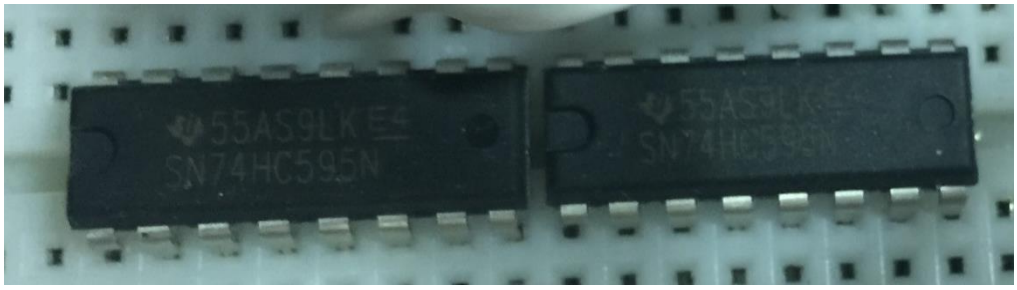
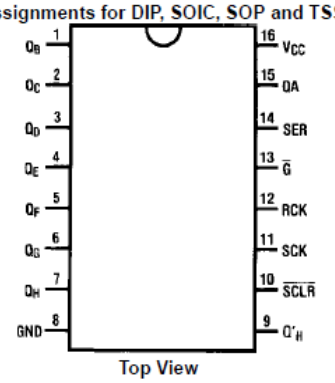


Figure 10 Shift Register(74HC595)

Connection Diagram

Pin Assignments for DIP, SOIC, SOP and TSSOP



Top View

Truth Table

RCK	SCK	$\overline{SCLR}$	$\overline{G}$	Function
X	X	X	H	Q_A thru Q_H = 3-STATE
X	X	L	L	Shift Register cleared $Q_H = 0$
X	↑	H	L	Shift Register clocked $Q_N = Q_{n-1}$, Q_0 = SER
↑	X	H	L	Contents of Shift Register transferred to output latches

Figure 11 Shift Register 핀 할당

LED 패널

LED 패널은 32개의 LED로 이루어져 있다. 총 2개의 줄에 각 16개의 LED가 있다. LED는 PCB보드 위에 납땜이 되어 있다. 한 줄에 16개의 LED를 정한 이유는 다음과 같다. 5장에서 다시 설명하겠지만, 음악에서 한 마디를 16분음표 16개로 표현한다. 마디 안의 음이 하나씩 울리면, 이에 해당하는 LED가 점등된다. Figure 11은 LED 패널을 나타낸다.

Figure 12 LED 패널

여러 스위치들

가변 저항

볼륨, 템포, 리버브, 팬 포트와 같이 값 변화에 민감하게 반응하지 않는 요소들로 선택한다. 왜냐하면 가변 저항은 전류가 흐르는 정도를 저항을 조절함으로써 입력으로 넣는데 종종 전류가 불안정할

때가 있다.[6] 때문에 변화를 주지 않아도 값이 조금씩 변할 때가 있다.

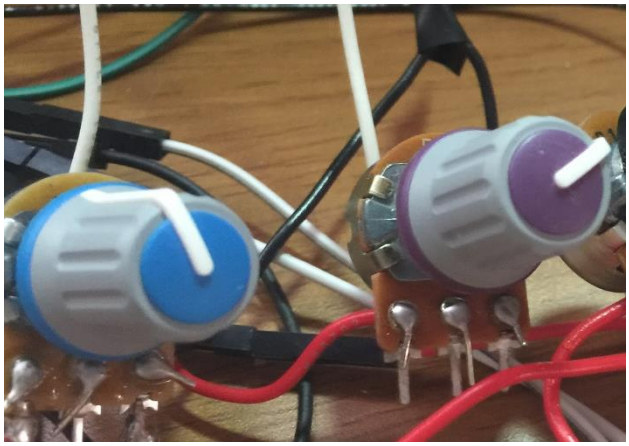


Figure 13 가변 저항

로타리 엔코더 스위치

가변 저항의 문제를 해결하게 위해 로타리 엔코더 스위치가 필요하다. 2개의 핀을 사용하는 이 스위치는 값 변화가 일어 났느냐의 여부(토글), 방향(왼쪽 또는 오른쪽)의 정보를 0, 1로 받아 소프트웨어적으로 변수의 값을 변화시키는 것이다. 따라서 이산적인 값을 조절하는 데 적합하다. 노트 위치 조절, 음색 변경, 그리고 채널 변경에 사용된다. 아두이노 포럼에 에서 엔코더 스위치를 제어하는 오픈 소스가 있어서 약간 변형한 후 사용하였다.[7] **[7]**

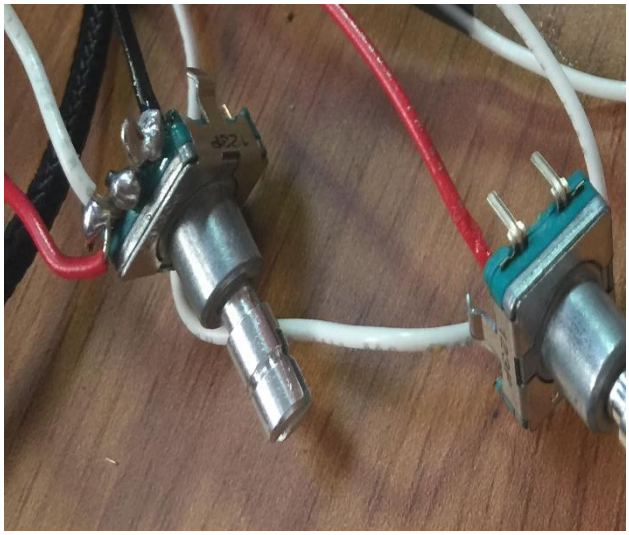


Figure 14 로타리 엔코더 스위치

---

[6] 예를 들어, 본체를 손으로 움직이면 브레드 보드에 연결된 전선들도 같이 출렁거린다. 또는 노트북 USB로 전압을 공급 받고 있는 상황에서 노트북을 끄면 전류도 낮아지게 된다.

[7] 원본은 인터럽트 호출 시에 값이 변하지만, 여기서는 토글이 일어났을 때 변경

게임 스위치

오락실 게임기에 쓰이는 스위치인데 재밌어 보여서 사용했다. 한 마디를 재생하거나 정지 또는 현재 위치의 노트를 제거하는데 사용된다.

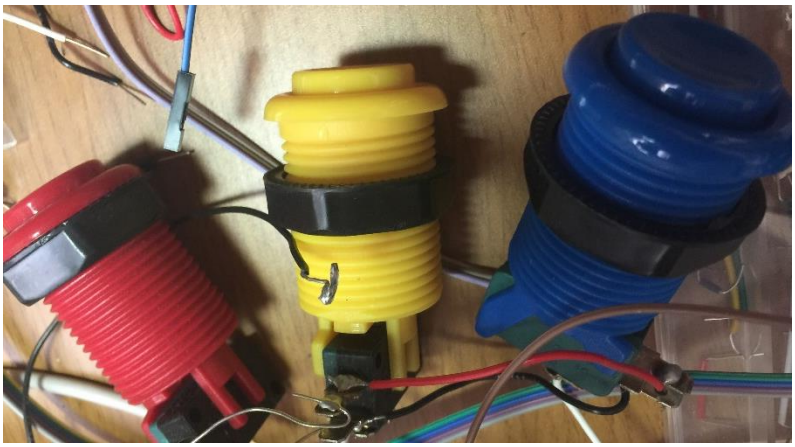


Figure 15 게임 스위치

택트 스위치

범용적인 스위치로 많이 쓰이는 스위치다. 아직 음악 소프트웨어 설계에 대해 자세히 설명하지 않았기 때문에 이전 설명에 이해가 안가는 부분이 있을 것 같다. 이 스위치는 현재 선택된 마디에서의 다음 마디, 이전 마디, 마디 추가, 마디 삭제 등의 마디 연산이 있다. 그리고 현재 음에서의 반음 쉬프트 기능을 제공한다. 반음 조절은 악전(musical notation)으로는 샾이나 플랫으로 표현한다.

Figure 16 택트 스위치

피아노 건반

하드웨어 구성 중에 가장 많이 갈아 치웠던 부분이 피아노 건반이었다. 처음에는 금속판이면 전선

을 강력 테이프로 접착해서 이어 붙이면 되지 않을까라고 생각해서 그렇게 하다가 건반이 이리저리 이동하는 과정에서 접촉 불량이 계속 생겨서 전도성 테이프로 바꾸었다. 전도성 테이프도 처음에는 잘 되더니 건반을 들었다 놨다 몇 번 반복하니 한 두 군데에서 접촉 불량이 계속 생기게 되었다. 결국에는 납땜으로 강하게 결합시키는 것이 필요하다고 생각해서 기존 접촉들을 다 떼어내고 다시 붙이게 되었다. 이때 사용된 판은 아연판이다. 알루미늄 판, 철 판, 구리 판을 써보면서 아연 판이 제일 낫다고 판단했다. 알루미늄은 납땜 페이스트를 발라도 납 접착이 잘 안되는 반편, 철판은 강도가 너무 높아서 자르기가 힘들고, 구리 판은 색상이 너무 틔어서 피아노 건반 색상과는 잘 안 맞는다고 생각하였다. 그래서 결국엔 아연판으로 선택하였다. Figure 16과 Figure 17은 피아노 건반의 위, 아래에서 찍은 사진이다.[8]

정전식 터치 센서를 언급할 때 나왔던 12개의 접점이 피아노 한 옥타브에 정확히 맞아 떨어지는 것을 볼 수 있다. 색을 보면 하얀색과 검은색이 있는데 이는 스프레이로 색을 입힌 것이다.

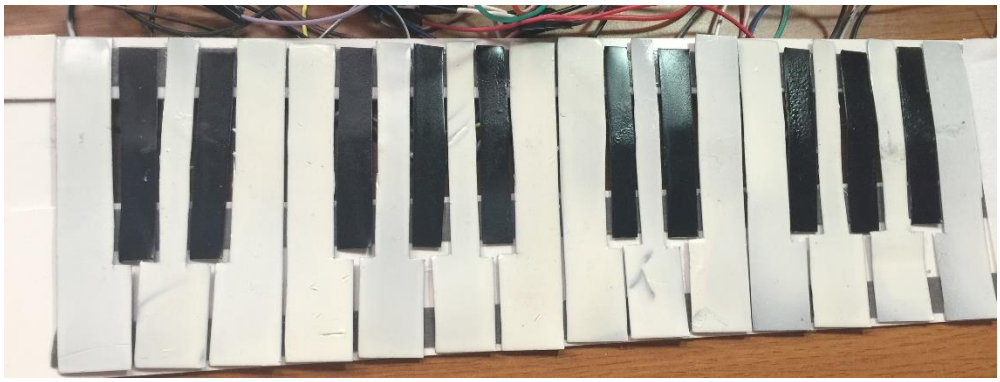


Figure 17 피아노 건반(위)

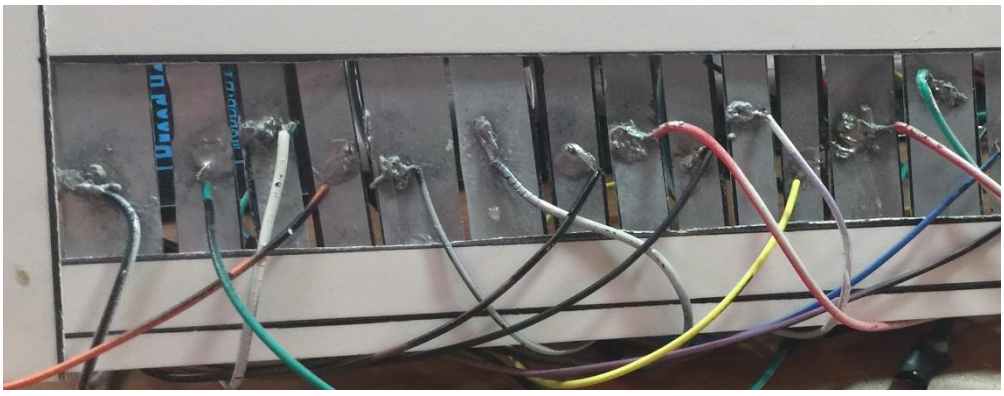


Figure 18 피아노 건반(아래)

[8] 구매한 판은 보통 초등생 과학 실험으로 제작된 금속 판들이어서 얇다.

Figure 19 아연 판

Figure 20 몬태나 스프레이(검/흰)

Figure 21 스프레이 뿌리고 난 직후의 아연 건반들

이상으로 하드웨어 구성에 대한 설명을 모두 마쳤다. 처음부터 이렇게 조립해보라고 정해준 메뉴얼이 없었다. 부품 하나하나가 시행착오의 대상이다. Figure 21은 브레드 보드 위에 조립된 하드웨어들의 총집합이다.

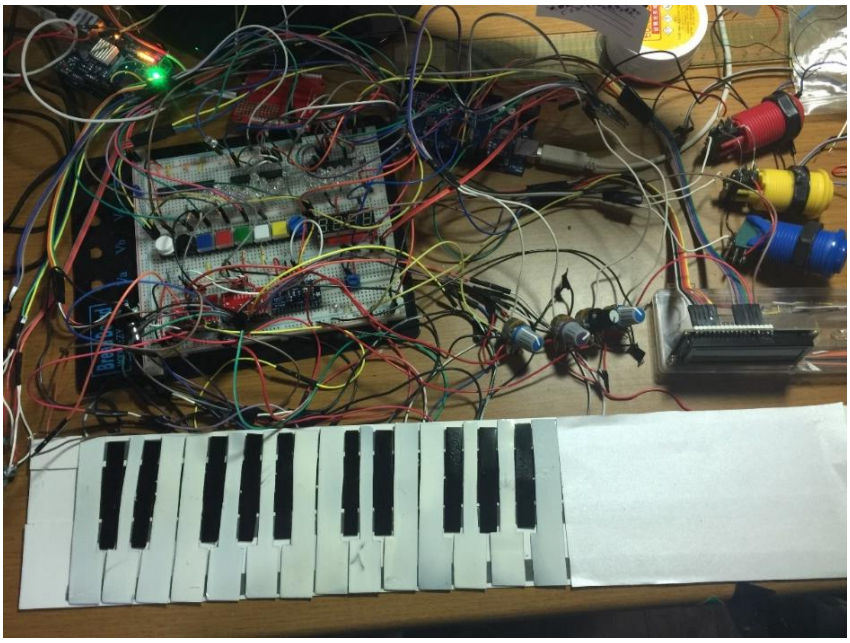


Figure 22 하드웨어 총 집합(조립 전)

Figure 23 신디사이저 전면

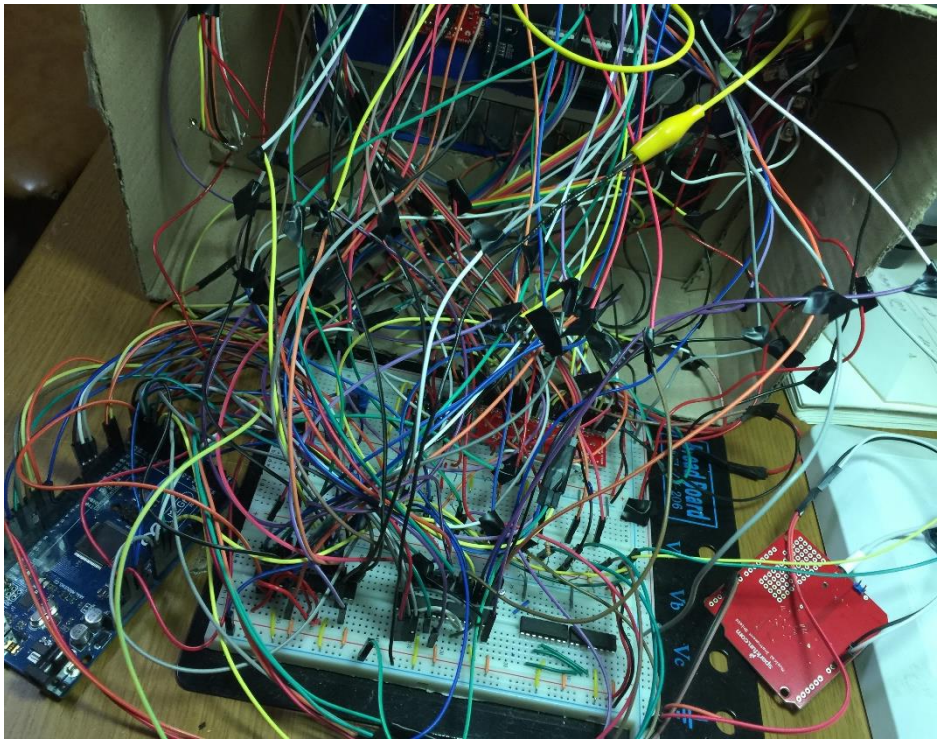


Figure 24 신디사이저 후면

4. 음악 프로그래밍

소프트웨어 구성은 아두이노 메가에 들어가는 음악 소프트웨어, 우노에 들어가는 통신 소프트웨어, 서버와 웹 클라이언트 쪽 소프트웨어가 있다. 먼저 음악을 숫자로 표현하는 방법을 살펴 보는 것이 좋을 것이다.

음악의 표현

음악은 음(note)들의 나열이다. 음들은 또한 마디(measure)라는 단위로 묶어진다. 마디는 마디끼리 묶여 선율(melody)을 형성한다. 운이 좋게도 음악의 기본인 음표는 컴퓨터가 다루기 수월하게 2의 배수로 되어 있다. 아래 Figure 24는 음표와 시간 간격이다.

가장 긴 온음표는 16음표가 16개로 이루어져있다. 이 단위가 한 마디라고 정하겠다. 따라서 32분음표는 표현하지 않는다. (실제로 잘 안 씀) 점이 붙은 경우는 역시 해당 음표의 절반만큼만 증가하므로 점16분음표를 제외하고는 표현가능하다.

구현 초기 단계에서는 이러한 시간 간격이라는 개념을 노트 on/off 메시지를 통해 구현하려고 했었다. 하지만 마디에 멜로디가 얹어 지면서 메모리 공간을 걱정하게 되었다. 시간 간격을 표현하기 위해서는 한 마디의 한 노트가 기본 단위이기 보다는 한 단위에 여러 노트 이벤트들이 존재한다는 것이다. 예를 들어 4분음표가 켜지고 꺼지기 바로 직전에 16분음표가 켜지게 되면 그 다음 시간에는 4분음표를 끄는 명령과 16분음표를 켜는 명령 두 개가 동시에 실행되어야 한다. 이러면 하나의 시간 단위에는 여러 개의 노트 값을 허용하게 되고 그렇게 되면 마디 하나의 메모리 사이즈는 수배 단위로 증가하게 되어 메모리 관리에 위협적이 된다. 따라서 한 타임 유닛 당 동일 채널에 있는 다른 음들을 전부 끄고 새로운 음을 내는 식으로 한다.

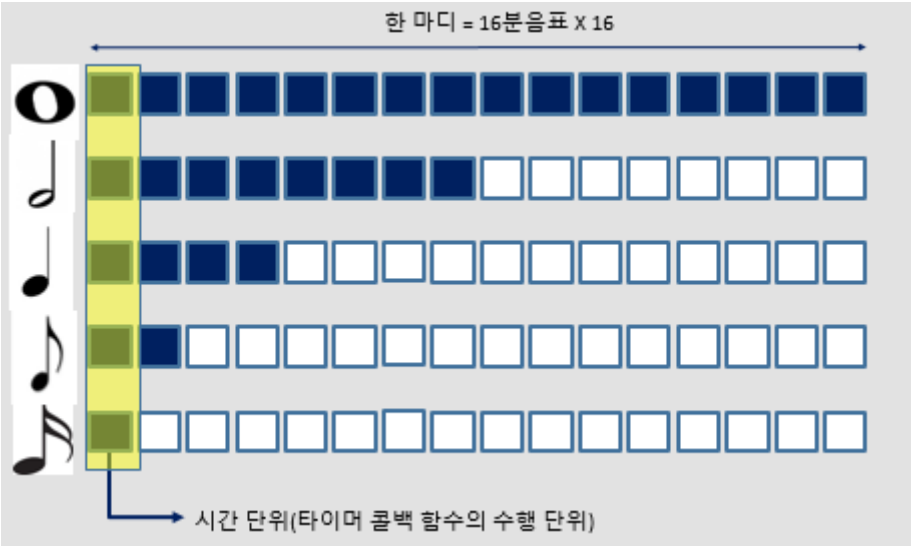


Figure 25 음표 별 시간 할당

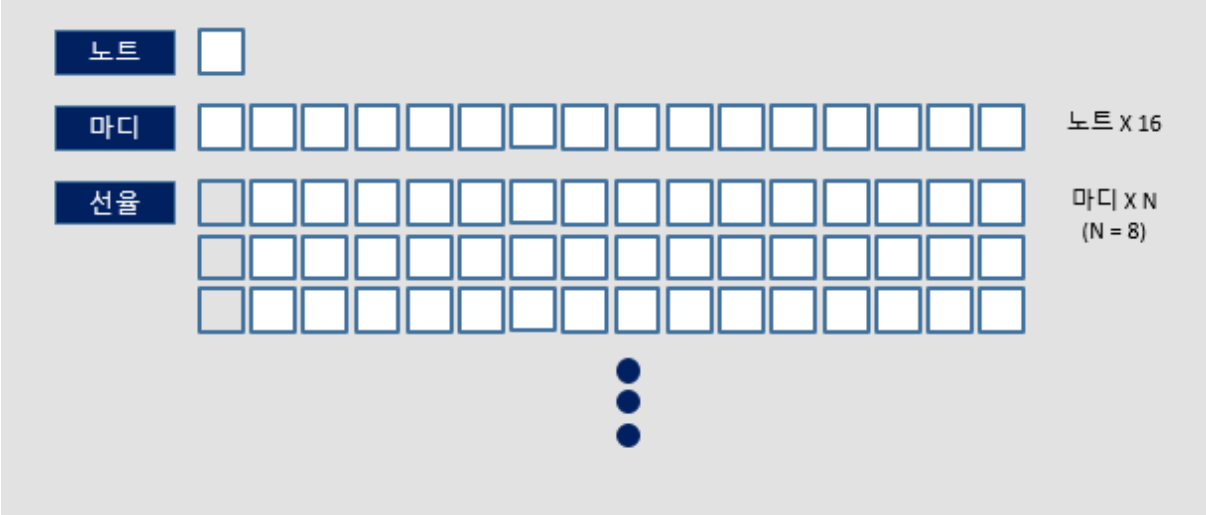


Figure 26 노트, 마디, 선율의 표현

미디 프로토콜

MIDI(미디)는 악기 디지털 인터페이스(Musical Instrument Digital Interface)를 줄인 말로 전자 악기끼리 디지털 신호를 주고 받기 위해 각 신호를 규칙화한 일종의 규약이다. 다시 말해 악기와 컴퓨터, 악기와 악기끼리 서로 주고받을 수 있는 언어와 통로의 신호 체계 표준이라 할 수 있다. 어떤 전자 악기(건반, 신디사이저, 모듈 등)가 이 표준에 따라 만들어졌다면, 그 전자 악기가 미디를 지원한다고 할 수 있다.[9]

채널 보이스 메시지(Channel Voice Messages)는 미디 컨트롤러가 명령을 보내는 방식이다. 뒤에서 다시 설명하겠지만, 이번 과제에서는 아두이노 메가가 미디 컨트롤러 역할을 담당하고 미디 음을 합성시켜 주는 사운드 모듈(또는 신디사이저)에 채널 보이스 메시지를 전송한다. 메시지 구성과 그 목록은 Table1과 Table2에 나와있다.

Table 1 채널 보이스 메시지의 구성

명령 바이트	데이터 바이트	데이터 바이트
8 bits	7 bits	7 bits

Table 2 채널 보이스 메시지 리스트

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)

[9] [8] "일반 MIDI", https://en.wikipedia.org/wiki/General_MIDI

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)

Table 3 미디 명령어 예시(간소화)

MIDI commands	
0x**8**0	Note Off
0x**9**0	Note On
0x**A**0	Aftertouch
0x**B**0	Continuous controller
0x**C**0	Patch change
0x**D**0	Channel Pressure
0x**E**0	Pitch bend
0x**F**0	(non-musical commands)

Table3의 명령들이 미디 보드에 전달되면 이에 해당하는 명령을 수행한다. 보통 미디는 16개의 채널로 구성되고 127가지의 악기를 지원한다. Table4는 미디 프로토콜에서 지원하는 악기 목록의 일부이다. 1번에서 48번까지만 표현했다. 표에서 볼 수 있듯이, 8개의 악기 묶음을 뱅크라고 부른다. 음색 뱅크 하나에 채널 하나가 할당된다.

Table 4 미디 악기 목록(1~48)

Piano Timbres:	Chromatic Percussion:	Organ Timbres:
1 Acoustic Grand Piano	9 Celesta	17 Hammond Organ
2 Bright Acoustic Piano	10 Glockenspiel	18 Percussive Organ
3 Electric Grand Piano	11 Music Box	19 Rock Organ
4 Honky-tonk Piano	12 Vibraphone	20 Church Organ
5 Rhodes Piano	13 Marimba	21 Reed Organ
6 Chorused Piano	14 Xylophone	22 Accordion
7 Harpsichord	15 Tubular Bells	23 Harmonica
8 Clavinet	16 Dulcimer	24 Tango Accordion

Guitar Timbres:	Bass Timbres:	String Timbres:
25 Acoustic Nylon Guitar	33 Acoustic Bass	41 Violin
26 Acoustic Steel Guitar	34 Fingered Electric Bass	42 Viola
27 Electric Jazz Guitar	35 Plucked Electric Bass	43 Cello
28 Electric Clean Guitar	36 Fretless Bass	44 Contrabass
29 Electric Muted Guitar	37 Slap Bass 1	45 Tremolo Strings
30 Overdriven Guitar	38 Slap Bass 2	46 Pizzicato Strings
31 Distortion Guitar	39 Synth Bass 1	47 Orchestral Harp
32 Guitar Harmonics	40 Synth Bass 2	48 Timpani

미디 라이브러리

미디 명령어를 소프트웨어로 다루기 위해서서는 별도의 라이브러리가 필요하다. 명령어 구조 자체는 단순하기 때문에 직접 구현할 수 있겠지만, 내용이 많고 이미 누군가가 오픈 소스 형태로 만들어 배포하였다. 아두이노 자체 플랫폼에서 미디 라이브러리를 제공하기는 하지만 내용이 빈약하고 범용성이 떨어진다. 여기서는 github에 올라온 FortySevenEffects / Arduino_midi_library를 사용했다.[10] 미디 라이브러리는 아두이노가 미디 컨트롤러 역할을 할 때 매우 간편한 인터페이스를 제공한다.

주로 쓰는 함수 목록은 Figure 26과 같다. 위에서부터 순서대로 노트 온, 노트 오프, 프로그램 체인지, 컨트롤 체인지 명령어를 지원하는 함수 인터페이스를 보여준다. 이전 장에서 미디 명령어 형식을 소개할 때, 명령어 바이트는 8비트, 2개의 데이터 바이트는 각 7비트임을 기억할 것이다. 아래 함수에서는 명령어 바이트에 따라 함수를 분리시켜 놓았다. 따라서 함수의 인자로 넘겨주는 값은 데이터 바이트와 채널 번호만 필요하다. 채널 번호는 명령어 바이트의 하위 4비트로 채워진다.

[10] [2] https://github.com/FortysevenEffects/arduino_midi_library

```
53  public:
54      inline void sendNoteOn(DataByte inNoteNumber,
55                             DataByte inVelocity,
56                             Channel inChannel);
57
58      inline void sendNoteOff(DataByte inNoteNumber,
59                              DataByte inVelocity,
60                              Channel inChannel);
61
62      inline void sendProgramChange(DataByte inProgramNumber,
63                                    Channel inChannel);
64
65      inline void sendControlChange(DataByte inControlNumber,
66                                    DataByte inControlValue,
67                                    Channel inChannel);
```

Figure 27 기본적인 미디 명령 함수

타이머 콜백 함수

시간은 음악에서 가장 중요한 요소이다. 특정 시간이 지나고 다음 음을 울리기 위해서는 시간을 다루는 데 익숙해야 한다. 앞서 음악을 노트들의 나열이라고 정의했다. 사실 노트들 사이의 침묵의 시간이 있어야 음악이 성립한다. 음악 교과서에 나오는 얘기다. 컴퓨터 프로그래밍에서, 시간 개념을 다루기 안성맞춤인 도구가 있다. 바로 타이머 콜백 함수이다

타이머의 구성은 호출 주기와 각 호출 마다 하는 일(콜백 함수)로 정의된다. 아두이노는 하드웨어에 내장된 타이머가 있지만, 여기서는 소프트웨어로 구현된 타이머를 사용한다. 왜냐하면 하드웨어 타이머는 주기를 한 번 설정해주면 동작 중에 마음대로 바꿀 수 없기 때문이다. 사용한 소프트웨어 타이머 코드는 오픈 소스로 작성된 코드이다.[11]

Figure 27은 타이머 콜백 함수 measure_performer()를 보여준다. 타이머는 정해진 주기마다 이 함수를 실행시킨다. 밑의 set_measure_tempo() 함수에서 타이머 주기를 바꾸는 코드를 볼 수 있는데, 이것이 소프트웨어 타이머를 사용하는 이유다. 신디사이저 기능 중에 음악의 템포를 조절하는 기능이 있다. 이를 위해서는 타이머 호출 주기를 실시간으로 변경할 수 있어야 한다. 하드웨어 타이머는 고정되어 있기 때문에 구현하기 힘들다. 원래 소프트웨어 타이머도 이를 금지했으나, 코드 내부를 파헤쳐 타이머 자료구조를 임의로 조작하여 템포 조절 기능을 추가하였다. 주기 설정은 처음 타이머를 생성할 때만 정하도록 인터페이스가 만들어져 있는데 사실 타이머 밑의 이벤트 함수를 찾고 들어가면 템포를 실시간으로 변경할 수 있다.

[11] [4] https://github.com/JChristensen/Timer

```
297 void measure_performer() {
298   static vector<nNote>::iterator curr;
299   // Serial.println("mea per");
300   curr = temp_measure.get_current_tracker();
301   MIDI_BOARD.sendControlChange(AllNotesOff, 127, curr_channel);
302   MIDI_BOARD.sendNoteOn(curr->pitch, 127, curr_channel);
303   temp_measure.inc_note_tracker();
304 }
305
306 void set_measure_tempo(long const tempo) {
307   measure_timer._events[mea_tmr_id].period = tempo;
308 }
```

```
mea_tmr_id = measure_timer.every(curr_tempo, measure_performer);
```

Figure 28 타이머 콜백 함수와 템포 조절

채널 자료 구조

앞서 미디를 설명할 때 채널에 관해 언급했다. 채널은 음악 재생의 핵심 개념이다. 16개의 분리된 채널을 통해 미디 메시지를 전달하기 때문이다. 채널의 개념을 여기서 한 번 더 상세히 설명하겠다. 음악은 서로 독립적인 채널에 전달되는 서로 다른 악기들의 연주이다. 채널 하나에 하나의 악기가 배정되고 악기는 자신의 채널에서 악보에 따라 음을 재생한다. 다른 악기는 신경 쓰지 않지만, 전체적으로 보면 악기들은 서로 조화롭게 어울려 음악이 탄생한다.

이처럼 채널은 독립적인 구조를 가지고 있고, 채널에서 재생될 음색이 필요하다. 이를 소프트웨어로 구현하기 위해서는 적당한 자료 구조가 필요하다. 자료 구조에 포함되어야 할 목록은 Figure 28과 같다. 먼저 각 채널은 자기만의 번호를 갖고 있어야 한다. 이는 채널 번호로 구분 된다. 상태는 활성 상태와 비활성 상태로 나뉜다. 활성 상태는 현재 채널이 사용할 수 있다는 의미이다. 비활성 상태는 채널 사용을 막아 놓는다. 이 둘을 구분하는 이유는 여러 채널을 동시에 재생하다가 중간에 특정 악기를 추가하거나 삭제하기 위한 정보가 필요하기 때문이다. 음색 목록은 채널에 할당된 음색 뱅크를 나타내고, 현재 음색은 음색 목록 뱅크 중에 하나의 음색을 나타낸다. 상태가 활성 상태일 때 재생이나 녹음이 가능하다. 이는 재생 정보 필드에서 나타낸다. 채널은 선택 상태, 재생 상태, 녹음 상태로 나뉜다. 선택 상태는 현재 선택된 채널을 끄거나, 악기를 바꾸거나, 마디를 수정하는 등의 작업을 할 수 있고, 재생 상태는 마디에 저장된 정보와 음색을 바탕으로 음악을 연주한다. 녹음 상태는 사용자가 건반을 두드리며 음을 녹음할 때 사용된다. 마지막으로 나오는 마디는 음들의 나열이 배열 구조로 들어 있다. 채널을 재생할 때 마디 배열을 읽어서 순차적으로 각 음들을 재생한다.

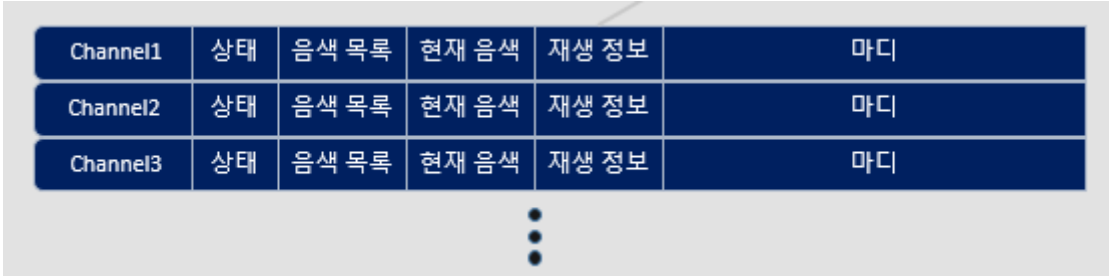


Figure 29 채널 자료 구조

5. 신디사이저의 기능

지금까지 악기의 서비스에 대한 설명은 없었다. 다음은 사용자가 악기를 가지고 할 수 있는 기능들이다.

자유 연주

자유 연주는 글자 그대로 건반을 두드리며 마음대로 연주할 수 있다. 음색이나 옥타브를 바꿔보면서 음을 만들어낼 수도 있다.

녹음

엔코더 스위치를 돌리면서 저장할 노트의 위치를 정하고 해당 음을 지정하는 방식으로 마디를 만들어 나간다.

재생/중지

재생은 타이머 콜백 함수가 선택된 마디 또는 멜로디를 호출 될 때마다 한 음씩 연주하게 한다. 중지는 재생을 멈추게 한다.

시퀀싱

드럼 머신과 비슷한 개념으로 이것 때문에 led 패널이 있다. 이 기능은 흔히 시퀀서 프로그램에서 보는 것과 비슷하게 기능한다.

상세 기능

사용자는 스위치를 조절하면서 다음 기능들을 수행할 수 있다.

1. 현재 노트 위치 조절
2. 현재 노트 위치 값 삭제
3. 채널 조절
4. 음색 조절
5. 템포 조절
6. 볼륨 조절
7. 리버브 효과 조절
8. 팬포트 효과 조절
9. 마디/멜로디 재생
10. 옥타브 쉬프트(샾)
11. 옥타브 쉬프트(플랫)
12. 다음 마디
13. 이전 마디
14. 마디 추가
15. 마디 삭제

16. 다음 멜로디
17. 이전 멜로디
18. 마디/멜로디 중지

Figure 28은 신디사이저의 기능을 한 눈에 알기 쉽게 설명해준다.

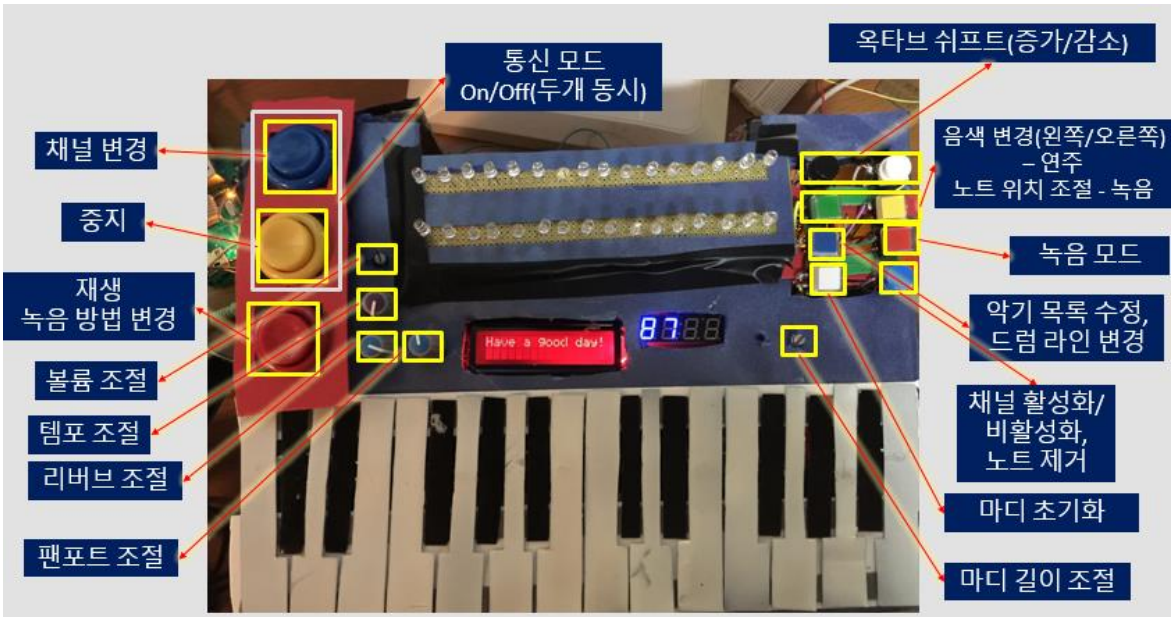


Figure 30 신디사이저의 기능

6. 통신

전체적인 통신 구조는 Figure 28과 같다. 신디사이저는 자체적으로 이더넷 보드와 I2C방식의 통신을 한다. 사용자와는 각종 입출력 장치를 통해 신호를 주고 받는다. 이더넷 보드는 Node js로 구축된 서버와 TCP 소켓을 주고 받는 방식으로 통신한다. 이 때 주고 받는 데이터는 JSON형식의 스트링이다. 따라서 JSON을 별도로 처리하기 위한 선처리 작업이 필요하다. 이는 오픈 소스 ArduinoJSON으로 구현되어 있다.[12]

웹 브라우저 상에서 동작하는 클라이언트 웹 페이지는 서버와 웹 소켓 방식으로 통신한다. 악기 제어는 실시간 통신이 적합하기 때문이다. HTML5 canvas와 jQuery UI를 중심으로 인터페이스가 구현되었다. 사용자는 이를 통해 신디사이저에 원격으로 음악과 관련된 구체적인 값들을 전달할 수 있다. Figure 29를 보면, 드럼 시퀀싱 모드를 수행할 때, 울리고 싶은 음은 각 원을 클릭하면 된다.

통신하는 포트 A와 웹 소켓으로 통신하는 포트 B가 있다. 포트 A는 우노와 통신하고 B는 웹 클라이언트와 통신하게 된다. 전체 그림은 웹 클라이언트 쪽에서 미디와 관련된 몇 가지 값 변화를 보내면 서버가 받아서 우노 쪽으로 보내는 것이다. 이때는 json형식으로 보내기 때문에 받는 우노 쪽에서는 이를 ArduinoJson이라는 오픈 소스를 이용해 파싱한다. 그리고 그 결과를 I2C를 통해 메가 쪽으로 보낸다. 전체 통신 구조 그림은 Figure 30에 나와 있다.

[12] https://github.com/bblanchon/ArduinoJson

신디사이저의 통신 구조

I2C 프로토콜(Inter-integrated Circuit Protocol)

Figure 28을 보면 신디사이저와 이더넷 보드 사이에 I2C 통신 계층을 볼 수 있다. I2C는 간단히 말해서 여러 개의 슬레이브 컴퓨터가 하나 이상의 마스터 컴퓨터와 통신하기 위해 사용하는 프로토콜이다. 속도가 낮고, 거리가 짧지만 구현하기 간단하다. 상세한 내용은 참고 자료 9번 "I2C Tutorial"을 참고하라.

I2C 통신을 통해 이더넷 보드는 웹 브라우저에서 받은 JSON을 파싱하여 얻어낸 값을 신디사이저에게 보낸다. 전달 받은 값은 신디사이저가 음악적인 용도로 활용한다.

이더넷 기반의 TCP/IP 통신

앞서 언급했듯이 사용자는 신디사이저를 하드웨어 장치를 통해 직접적으로 접근할 수도 있지만 웹 브라우저를 통해 간접적으로도 접근할 수 있다. 이 때 웹 브라우저를 통해 접근한다. 웹 브라우저는 다시 서버와 통신하고 서버는 사용자로부터 값을 이더넷 보드에 넘겨준다. 즉, 실제로 인터넷에 연결되어 자료를 주고 받는 주 무대는 TCP/IP 기반의 통신이다.

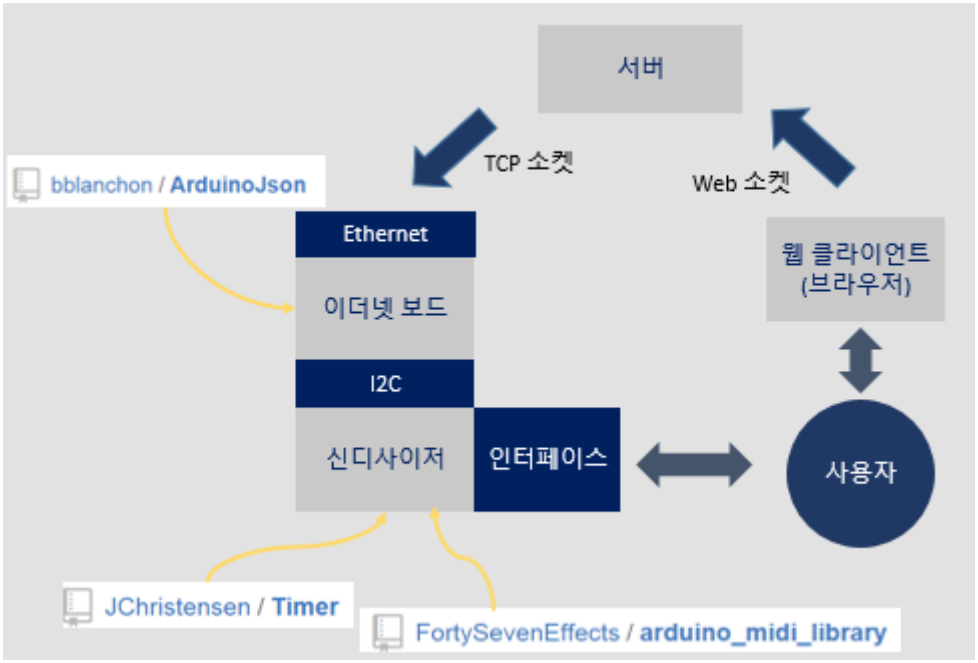


Figure 31 전체 통신 구조

웹 클라이언트

Figure29는 웹 기반의 인터페이스를 나타낸다. Figure 31에서 볼 수 있듯이, 위 쪽은 드럼 시퀀서를 조절할 수 있고, 아래 쪽은 구체적인 수치를 슬라이더바를 이용해 조절할 수 있다.

드럼 시퀀서 부분을 살펴보자. 총 5개의 행에 각각 16개의 원이 그려져 있다. 이는 드럼 비트를 만들기 위해 5개의 채널을 사용하고 마디 하나를 사용하겠다는 의미이다. 16 step은 일반적으로 자주 쓰이

는 비트 수이다. 아래 슬라이더의 Drum line을 움직여서 채널을 선택하고, Drum timbre를 조절하여 드럼의 소리의 종류를 선택할 수 있다. 사실 말이 드럼이지, 드럼 뱅크 안에는 드럼 이외의 소리도 많이 포함되어 있다. 이렇게 선택하고 나서 위의 원에 마우스를 갖다 대고 클릭을 하면 해당 드럼 비트가 기록된다.

위의 작업을 끝낸 후, 아래의 Play버튼을 클릭하면 드럼 시퀀싱의 정보가 신디사이저로 넘어가고 그 쪽에서 드럼이 연주된다. 정지하기 위해서 Stop버튼을 누르고, 현지 기록된 비트 정보를 지우기 위해서는 Remove버튼을 클릭한다. Sync버튼은 가끔씩 웹 클라이언트에서 드럼 라인을 따라 이동하는 선과 신디사이저에서 LED가 점등하는 열이 서로 다를 때 사용한다. 드럼 시퀀스가 시작되면 16개의 열이 하나씩 순차적으로 번갈아 가며 색깔이 변하기 때문이다. LED쪽은 물론 빛이 켜진다.

그럼 아래 쪽의 슬라이더들을 구체적으로 살펴보자. 먼저 템포 슬라이더를 제외한 모든 슬라이더는 마우스를 갖다 대고 조절하는 즉시 변경된 값이 서버로 전송된다. 이는 jQuery UI에서 슬라이더의 값이 변하는 이벤트에 호출되는 함수를 호출함으로써 구현 된다.

볼륨 슬라이더는 이름 그대로 악기의 볼륨을 조절한다. 템포 슬라이더는 조금 까다로운데, 타이머와 관련되어 있기 때문이다. 템포에 변화를 주는 연산은 신디사이저와 웹 브라우저에게 모두 부담스러운 연산이다. 때문에 템포 값은 최종적으로 변경된 값만 서버로 전송된다. 이렇게 하지 않고 실시간으로 템포를 변경하면 신디사이저와 웹 브라우저 간의 싱크 문제가 심해진다. 다음으로는 음색(Timbre)를 선택하는 슬라이더가 있다. Table4에서 일부 목록을 확인할 수 있다. 다음 2개의 슬라이더는 앞서 설명한 드럼 시퀀싱과 관련이 있다. Note 슬라이더는 음의 높이를 조절하는 역할을 한다. 드럼 시퀀싱이라고 해서 모든 음을 드럼 뱅크에서만 선택해야 하는 것은 아니다. 피아노 음도 따올 수 있기 때문에 음 조절하는 기능도 필요하다. 마지막으로 Reverb와 Pan Port가 있다. Reverb는 울림 효과의 정도를 의미하고, Pan Port는 좌우 스피커에 나오는 소리의 균형을 조절한다.

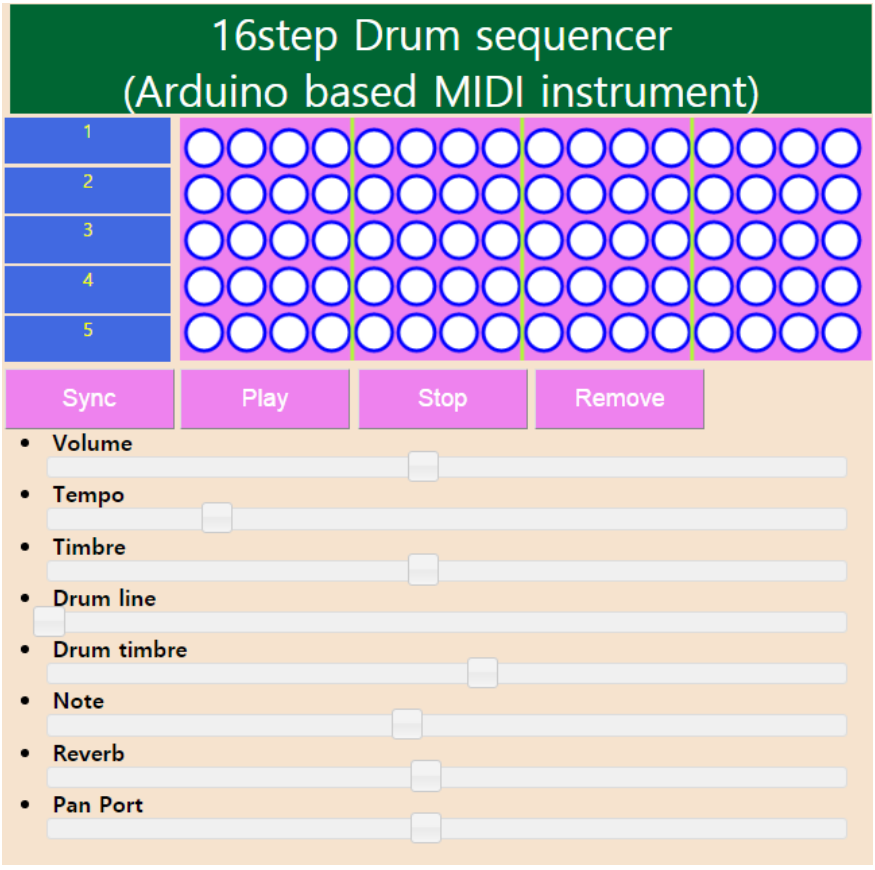


Figure 32 웹 브라우저에서의 아두이노 제어 화면

서버

서버 구조는 간단하다. 사용자로부터 오는 정보를 처리하기 위해 웹 소켓을 만들고 이를 그대로 이더넷 보드 쪽으로 던져주기만 하면 되기 때문이다. 그리고 Node JS는 서버에 최적화된 플랫폼이기 때문에 코딩 역시 단순하다. Figure 31과 32는 서버 쪽의 코드를 보여준다.

```
client_server = http.createServer(onRequest).listen(cli_port);
arduino_server = net.createServer().listen(ardu_port);


client_web_socket = io.listen(client_server);
// arduino_tcp_socket = io.listen(arduino_server);

client_web_socket.sockets.on('connection', function(socket){
  //send data to client
  setInterval(function(){
      socket.emit('date', {'date': new Date()});
  }, 1000);

  socket.write("connected to the tcp server\r\n");

  socket.on('client_data', function(data){
    process.stdout.write(data.letter);
  });

  socket.on("music_value_change", function(data) {
    var jstring = JSON.stringify({name : data.name, value : data.valu
    if(ardu_conn == true) {
      sendMsgToUno(jstring);
    } else {
      console.log("it doesn't have receiver.");
    }
    process.stdout.write(data.name + " " + data.value + " " +
                         jstring + "\n\r");
  });
});
```

Figure 33 서버 측에서의 웹 소켓

```
arduino_server.on('connection', function(socket) {
  asocket = socket;
  ardu_conn = true;
  socket.write("this is new arduino socket communication");
  console.log("connection is made on port " + ardu_port + "#" + arduino_server.con

  // recevie data from arduino and print it
  socket.on('data', function(data) {
    console.log('received on tcp socket : ' + data);
  });

  socket.on('error', function(e) {
    console.log('error event ocurred\n\rarduino might reconnect to this host.');
    socket.write("Reconnecting...");
  });
});

//recieve client data

console.log("Server has started.");
}
function sendMsgToUno(str) {
  asocket.write(str);
```

Figure 34 아두이노를 위한 TCP 소켓 생성

7. 결론

이상으로 신디사이저에 대한 많은 내용을 설명했다. 다시 한번 언급하지만, 이 과제는 단순히 하드웨어 부품들을 주워담아 신디사이저의 외형만 만드는 것도 아니고, 음악 소프트웨어만 구현하는 것도 아니고 통신 기능만 달랑 있는 허무한 악기가 아님을 강조하고 싶다. 이 프로젝트를 진행한 이유는 하드웨어 부품들과 음악적 지식을 악기 제작이라는 깃발 아래 끌어 모아 신디사이저라는 하나의 대상으로 통합해 보기 위함이다. 이 과정에서 컴퓨터 공학은 이들 사이에 다리를 놓아 서로를 연결하는 가교 역할을 한다. 전자 공학은 개별 하드웨어 부품에 집중하고, 음악은 음악 이론에 집중하지만 컴퓨터 소프트웨어는 이들을 결합하는데 주력한다.

신디사이저를 구성하는 기술들을 다시 한 번 훑어보자. 먼저 악기의 본체를 구성하기 위한 토대가 필요하다. 이는 2개의 컴퓨터, 즉 악기 컴퓨터, 통신 컴퓨터로 구성된다. 악기 컴퓨터는 아두이노 메가 2560 보드를 사용하고 통신 컴퓨터는 아두이노 우노 R3를 사용한다. 메가 보드는 다양한 입출력 수단으로 사용자로부터 피드백을 주고 받는다. 입출력 수단으로는 건반, 푸시 버튼, 택트 스위치, 엔코더 스위치, 가변 저항 노브, LED 패널, LCD, 7segment가 있다. 이들을 통해 사용자와 소통하며 음악을 연주한다. 우노 보드는 서버와의 통신을 위해 사용된다. TCP/IP 기반의 자료를 JSON 형식으로 주고 받는다. 사용자는 또한 웹 브라우저를 통해 신디사이저를 간접적으로 제어할 수 있다. 웹 클라이언트는 서버와 웹 소켓 방식을 사용하여 값을 빠르게 신디사이저로 전달한다.

위의 기술들과 더불어 오픈 소스 커뮤니티로부터 많은 소스들을 사용하였다. 대표적으로 미디 라이브러리, ArduinoJson, Timer, 그리고 하드웨어 부품 활용 예제 코드 등이 있다.

오픈 소스 시대의 교훈은 "*당신은 바닥부터 만들 필요가 없다. 당신과 비슷한 고민을 한 사람이 지구상에 반드시 한 명 이상이 있을 것이고 그가 당신을 위한 코드를 어딘가에 이미 만들어 놓았다. 당신은 그저 사용하기만 하면 된다*" 는 말로 대변할 수 있을 것이다. 이 과제는 오픈 소스를 적극적으로 사용하지 않았다면 결코 완성할 수 없었다고 생각한다. 사용된 오픈 소스 코드를 직접 구현하는 것은 엄청난 노력과 인내력과 시간이 필요하다. 오픈 소스 문화 공동체의 힘은 그러한 수고를 덜어준다. 덕분에 아이디어에 좀 더 치중할 시간이 많아졌고, 악기 제작, 하드웨어/소프트웨어 디버깅과 수정 보완 작업에 시간을 더 쓸 수 있었다.

8. 참고 자료

오픈 소스

[1] https://github.com/adafruit/Adafruit_MPR121

[2] https://github.com/FortySevenEffects/arduino_midi_library

[3] https://github.com/bblanchon/ArduinoJson

[4] https://github.com/JChristensen/Timer

[5] https://forum.arduino.cc/

[6] https://learn.sparkfun.com/

[7] https://playground.arduino.cc/Main/RotaryEncoders

참고 자료

[1] 유하영, 전우영, *아두이노 로봇 보난자*, 비제이퍼블릭, 2014

[2] 황주선, 김현규, *재잘재잘 피지컬 컴퓨팅 DIY: 아두이노, 센서, 네트워크를 통해 세상을 보고 듣고 느끼는 방법*, 인사이트, 2014

[3] 타카하시 노부유키, *컴플리트 MIDI북*, SRM(SRmusic), 2007

[4] 채진욱, *아두이노 for 인터랙티브 뮤직*, 인사이트, 2011

[5] 김유희, *음악기초이론의 이해*, 예솔, 2010

[6] "일반 MIDI", https://en.wikipedia.org/wiki/General_MIDI

[7] "I2C Tutorial", https://learn.sparkfun.com/tutorials/i2c

[8] SparkFun Music Instrument Shield, https://www.sparkfun.com/products/10587

[9] "VS1053b Datasheet – Ogg Vorbis/MP3/AAC/WMA/MIDI AUDIO CODEC", https://www.sparkfun.com/datasheets/Components/SMD/vs1053.pdf, pp. 32,

[10] 이경전, 사물 인터넷에 대한 올바른 정의와 LPWAN, KISA Report, 한국인터넷진흥원, 2015

[11] 유재필, 오픈소스 하드웨어 플랫폼(OPHW) 동향 및 전망, 한국인터넷진흥원, 2013

9. 부록

부품 리스트

Table 5 부품 일람표

No	부품 명	용도
1	Arduino Mega 2560 x 1	신디사이저 본체 컴퓨터
2	Arduino UNO R3 x 1	이더넷 통신 처리
3	Arduino Ethernet Shield W5100 x 1	TCP/IP 스택, 이더넷 포트
4	SparkFun Music Instrument Shield x 1	MIDI 프로토콜과 음 생성
5	RGB Backlight negative LCD 16 x 2	LCD 디스플레이
6	7 Segment Serial – Blue x 1	숫자 디스플레이
7	MPR121 Capacitive Touch Sensor Board x 1	건반 입력
8	Shift Register, 74HC595	LED 이어 붙이기
9	10K Potentiometer x 4	음악 수치 값 조절
10	Tact switches x 8	사용자 입력
11	Game Push switches x 3	사용자 입력
12	Rotary Encoder switches x 1	노트 길이 조절
13	LED – 3파이, White Color x 32	시퀀싱 디스플레이
14	Zinc plates	건반 제작
15	Bread Board	부품 결합
16	Jumper cable, wire	보드와 부품들 연결
17	Shift Register, 74HC595	LED 이어 붙이기
18	몬타나 스프레이 – 흰색, 검은색	건반에 색 입히기
19	절연 테이프	전선 이어붙이기, 부품 고정
20	일반 상자	본체 제작
21	아크릴 판 두께 3T	본체 제작

부품 구입 사이트

1. http://www.mechasolution.com/
2. http://www.eleparts.co.kr/
3. http://www.devicemart.co.kr/

10. 돌아보며

졸업 과제를 수행하면서 많은 우여곡절이 있었다. 주제 선택과 문제 정의를 할 때부터 갈팡질팡했었다. 지속적으로 자료 조사를 하면서 나에게 가치 있는 문제란 무엇인가를 고민하면서 도중에 주제를 바꿔 보기도 했다. 그러던 와중에 오픈 소스 문화를 다시금 접했고, 하드웨어 영역에도 오픈소스가 강세를 보이고 있다는 사실을 알게 되었다. 이에 아두이노와 관련된 자료들을 훑어보았고, 유튜브에서 기기묘묘한 아두이노 기반의 작품들을 수차례 보면서 나 역시 멋있는 작품을 하나 만들어 보고 싶었다. 평상시 음악에 관심이 있었기 때문에, 졸업 과제를 기회로 전자 악기를 직접 제작해 보는 과정은 뜻깊은 경험이었다.

교훈이라고 하면 '혼자서 할 수 있는 일은 한계가 있다'는 것을 뼈저리게 느꼈다. 나는 여기서 사용된 오픈 소스와 인터넷의 이름 모를 수 많은 프로그래머로부터 많은 도움을 받았다. 누군가가 고민한 문제를 커뮤니티에 올리면 그와 비슷한 고민을 했던 사람이 나타나 조언을 해준다. 이러한 문화는 기술 개발에 있어 매우 중요하다. 결국 제품 제작은 많은 기술들이 상충되는 기능과 알맞은 기능들이 서로 찢어지고 합쳐지는 과정을 통해 적절한 모습으로 나타나기 때문이다. 이러한 기술이라는 퍼즐 조각들을 원하는 위치에 맞춰 넣기 위해서는 다양한 사람들이 다양한 기술에 관해 알고 있어야 하고 이들을 공유하려는 문화가 활발해야 한다.

마지막으로 필자가 오픈 소스 문화와 관련된 글 중 공유 문화의 본질을 잘 묘사하는 글을 인용하면서 후기를 마치려고 한다.

> 인간은 생존을 위한 지적 호기심과 함께 도구를 제작하려는 본능을 갖고 있다. 이는 다른 생물들과 달리 인류가 진화적인 측면에 우위를 갖게 했고 지금과 같은 문명과 사회를 구성하게 된 주된 본능이다. 이러한 욕구는 단순 생존 욕구와는 차원이 다르며, 실질적인 생존과 관계가 없이도 발현되는 본능이다. 이러한 본능에 더해서 인류가 번영하게 된 이유 중 하나는 사회성이다. 이 사회성은 다른 생물들의 유전학적인 결과로 만들어지는 사회성과 매우 다른 특징을 보인다. 그 특징을 한마디로 표현한다면 정보의 공유를 통한 사회성이다. 인류는 본능적으로 자신이 갖고 있는 정보를 타인에게 전달하고 이를 통해 사회 구성원 전체의 발전에 기여하 고자 한다. 이와 같은 정보 공유를 위해 언어를 발달시켰으며 문자를 발명하고 지 금과 같은 IT 기기들을 만들어 감으로써 생물학적인 한계를 극복한 시간과 공간을 가로지르는 정보 공유를 통해서 현재와 같은 인류 문명을 구축하게 되었다. [13]

[13] [11], 유재필, 오픈소스 하드웨어 플랫폼(OPHW) 동향 및 전망, 한국인터넷진흥원, 2013

11. 감사의 글

먼저 이 과제를 허락해 준 김호원 교수님께 감사의 말씀을 드리고 싶다. 혼자서 수행하게 되었음에도 불구하고 저를 믿어 주셨고 과제를 계속 진행하도록 허락해주셨기 때문에 이렇게 끝까지 밀고 나갈 수 있었다고 생각한다.

그리고 나를 지원해준 부모님께도 역시 감사의 말을 보내고 싶다. 특히나 부품과 관련해서 비용이 꽤 들었다. 용돈만으로는 필요한 부품을 모두 구입하기에는 부족했는데, 부모님의 안정적인 지원 덕분에 원하는 부품들을 구입할 수 있게 되었다.