목차

1. 수행한 과제

Buzzer

- 부저 1 개를 아두이노와 연결하시오.
- Tone generator 를 사용하여 30 초 이상의 멜로디를 연주하시오.

2. 부품 리스트

핵심 사용 부품

- 아두이노 우노 R3(Arduino Uno R3) x1
- 수동 부저(Passive Buzzer) x1
- 택트 스위치(Tact Switch) x1
- 10K 가변 저항(10K Variable Resistance)
- 초록색 LED (Green LED) x1

기타 사용 부품

- 830 포인트 브레드보드(830 Point Breadboard) x1
- 숫숫 점퍼 케이블(Male to Male Jumper Cable) x9
- 220Ω 저항(220Ω Resistance) x1

3. 구현하고자 하는 기능 및 설명

Core - 1. 아두이노와 수동 부저를 이용한 30 초 이상의 멜로디 연주 기능

아두이노와 수동 부저를 연결하고, PWM 을 이용한 Tone Generation 을 통해 30 초 이상의 멜로디 연주를 소프트웨어적으로 처리하여 구현한다. 이 과정에서 tone 과 noTone 함수를 사용하였다.

Idea - 1. 택트 스위치를 이용한 연주 트리거 기능

택트 스위치를 눌렀을 때 연주가 시작되도록 트리거 버튼을 만든다. 별도 추가 저항 없이 pinMode 함수의 설정을 변경하여 아두이노에 내장된 풀업 저항을 사용하여 구현하였다.

Idea - 2. LED 를 이용한 멜로디 연주중 상태 시각적 피드백 기능

아두이노에 초록색 LED(220 옴 저항과 함께)를 연결하여 연주가 진행되고 있을 때 점등하여 멀리서도 확인할 수 있도록 시각적 피드백을 구현하였다.

Idea - 3. 가변 저항을 이용한 부저 소리 크기 조절 기능

수동 부저에 가변 저항을 연결하고 사용자가 부저에 흐르는 전류의 양을 조절 가능하게 하여, 부저의 소리 크기를 조절할 수 있도록 구현하였다.

Idea - 4. 직렬 통신을 이용한 멜로디 교체 기능

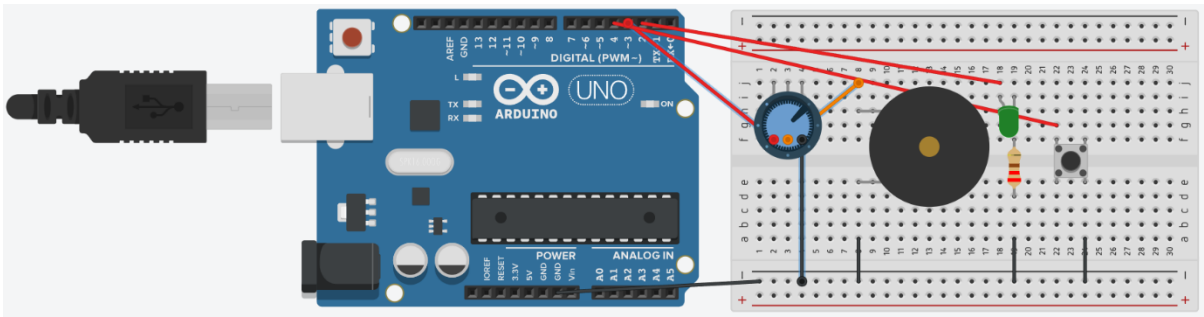
컴퓨터와 USB 로 아두이노 우노 보드를 연결한 뒤, 직렬 통신을 이용하여 간이 음악 문자열 정보(직접 만든 포맷)를 기기가 내려받을 수 있도록 구현하였다.

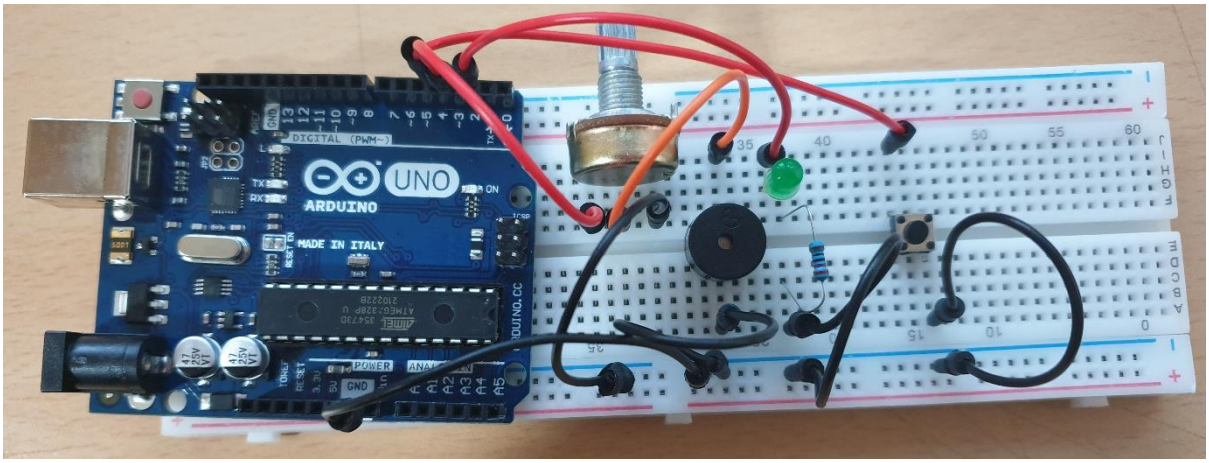
Etc - 1. Node.js 를 이용한 간단한 RTTTL(Ring Tone Text Transfer Language) 파서

RTTTL 이란? -> 노키아가 개발한 휴대폰 벨소리 교환간에 사용할 수 있는 벨소리 기술 언어

RTTTL 포맷을 간이 음악 문자열 정보로 변환하는 파서가 있다면 기기에 손쉽게 멜로디 정보를 전송 가능하다. 해당 파서를 만들어, 멜로디 재생 코어 로직에서 사용하였고, 인터넷에서 RTTTL 포맷으로 기술된 여러 벨소리를 찾아, 간이 음악 문자열 포맷으로 변환하였다. (5. 프로그램 소스 – MUSIC.txt 참조)

4. 부품과 아두이노 우노 보드와의 연결도





5. 프로그램 소스

GitHub: https://github.com/refracta/koreatech-assignment/tree/master/MicroprocessorNPractice/Assignment1

<Assignment1.ino>

```
#define DEBUG_MODE true
// 디버그 플래그

#if DEBUG_MODE
  #define debug(log) Serial.println(log);
#else
  #define debug(log)
#endif

#define LED_PIN 2
#define BUZZER_PIN 3
#define SWITCH_PIN 4
// 핀 상수 설정

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(SWITCH_PIN, INPUT_PULLUP);
  Serial.begin(9600);
  // LED, BUZZER, SWITCH 의 pinMode 를 설정
  // Serial 통신을 위해 9600 Baud Rate 설정
}

const int frequencies[] =
{
        31,
  // octave 0
```

```
27.       33, 35, 37, 39, 41, 44, 46, 49, 52, 55, 58, 62,
   // octave 1
28.       65, 69, 73, 78, 82, 87, 93, 98, 104, 110, 117, 123,
   // octave 2
29.       131, 139, 147, 156, 165, 175, 185, 196, 208, 220, 233, 247,
   // octave 3
30.       262, 277, 294, 311, 330, 349, 370, 392, 415, 440, 466, 494,
   // octave 4
31.       523, 554, 587, 622, 659, 698, 740, 784, 831, 880, 932, 988,
   // octave 5
32.       1047, 1109, 1175, 1245, 1319, 1397, 1480, 1568, 1661, 1760,
   1865, 1976, // octave 6
33.       2093, 2217, 2349, 2489, 2637, 2794, 2960, 3136, 3322, 3520,
   3729, 3951, // octave 7
34.       4186, 4435, 4699, 4978
   // octave 8
35. };
36. /*
37.  * data from
   https://www.arduino.cc/en/Tutorial/BuiltInExamples/toneMelody
38.  * 아두이노 튜토리얼 문서의 헤더 파일을 배열 형태로 정리한 것, 각 음계의 진동수의
   상수 정의이다.
39.  */
40. const char notes[] = {'c', 'C', 'd', 'D', 'e', 'f', 'F', 'g', 'G',
   'a', 'A', 'b'}; // length = 12
41. // "도 도# 레 레# 미 파 파# 솔 솔# 라 라# 시" 순의 출력 처리용 character 배열
42.
43. // note 에 해당하는 notes 의 index 를 반환한다.
44. int getNoteIndex(char note) {
45.   for(int i = 0; i < 12; i++){
46.     if(notes[i] == note) {
47.       return i;
48.     }
49.   }
50.   return -1;
51. }
52.
53. // note, octave 의 출력으로 duration 동안 부저에 소리를 출력하는 함수
54. void playTone(char note, int octave, int duration){
55.   int frequency = frequencies[getNoteIndex(note) - 11 + octave *
   12];
56.   tone(BUZZER_PIN, frequency, duration);
57.   delay(duration);
58.   noTone(BUZZER_PIN);
59. }
60.
61. /* 간이 음악 문자열 포맷을 출력해주는 함수이다.
```

```
62.  *  "c=도, C=도#, d=레, D=레#, e=미, f=파, F=파#, g=솔, G=솔#, a=라, A=라#, b=시, P=쉼표"를 출력
63.  *  S%숫자%: Tempo(Bit per minute)를 숫자으로 설정
64.  *  숫자: 현재 옥타브를 숫자로 설정
65.  *  example) 6S100abc7abc => 6 옥타브 설정, Tempo=100, 라, 시, 도, 7 옥타브 설정, 라, 시, 도
66.  */
67. void playMusic(String music) {
68.   int musicLength = music.length();
69.   int currentOctave = 4;
70.   int currentSpeed = 250;
71.   for(int i = 0; i < musicLength; i++) {
72.     char m = music.charAt(i);
73.     if(m == 'S' || m == 's') {
74.       String sb = "";
75.       for(i = i + 1; i < musicLength; i++){
76.         char c = music.charAt(i);
77.         if('0' <= c && c <= '9') {
78.           sb += c;
79.         } else {
80.           i--;
81.           break;
82.         }
83.       }
84.       currentSpeed = (60.0 / sb.toInt()) * 1000;
85.     } else if (m == 'P' || m == 'p'){
86.       delay(currentSpeed);
87.     } else if ('0' <= m && m <= '9') {
88.       currentOctave = m - '0';
89.     } else {
90.       playTone(m, currentOctave, currentSpeed);
91.     }
92.   }
93. }
94.
95. // LED 의 상태를 변경한다.
96. void setLED(bool status) {
97.    digitalWrite(LED_PIN, status ? HIGH : LOW);
98. }
99.
100.       // 현재 스위치가 눌림 여부를 반환한다.
101.       bool isSwitchActive() {
102.         return digitalRead(SWITCH_PIN) == LOW;
103.       }
104.
105.       // 버튼 트리거시 재생되는 간이 음악 문자열 (Indiana Jones, Take On Me, Star Wars)
```

```
106.      String currentMusic =
   "5S250eS500pfgp6S63c5S333pS250dS500peS63fS167pS250gS500pabp6S63f5S25
   0paS500pb6S125cde5S250eS500pfgp6S63c5S250p6d5S500p6eS42f5S250gS500pg
   6S167e5S500p6S250d5S500pg6S167e5S500p6S250d5S500pg6S167f5S500p6S250e
   5S500p6dS125c5S320FFFd4pbp5e4p5e4p5eGGabaaae4p5d4p5F4p5F4p5FeeFeFFFd
   4pbp5e4p5e4p5eGGabaaae4p5d4p5F4p5F4p5Fee5S360pFFFS60b6FS360eDCS60bS1
   20FS360eDCS60bS120FS360eDeS60C5S360FFFS60b6FS360eDCS60bS120FS360eDCS
   60bS120FS360eDeS90C";
107.
108.      // 직렬 통신으로 간이 음악 문자열 수신 처리
109.      void handleSerialRequest(){
110.        if(Serial.available()){
111.          currentMusic = Serial.readString();
112.          Serial.println("Received successfully.");
113.        }
114.      }
115.
116.      // 코어 로직 루프
117.      void loop() {
118.        if(isSwitchActive()){
119.          setLED(true);
120.          playMusic(currentMusic);
121.          setLED(false);
122.        }
123.        handleSerialRequest();
124.      }
```

<ConvertRTTTL.js>

```
1. let target = process.argv.slice(2).join(" ");
2. let[name, setting, data] = target.split(":").map(e => e.trim());
3. data = data.toLowerCase();
4. setting = Object.assign(...setting.split(",").map(e =>
   Object.fromEntries([e.trim().split('=')])));
5. data = data.split(',').map(e => e.trim().replace('c#',
   'C').replace('d#', 'D').replace('f#', 'F').replace('g#',
   'G').replace('a#', 'A').split(''));
6. data = data.map(e => {
7.     let d = e.filter(e=>e==='.').length;
8.     e = e.filter(e=>e!=='.');
9.     if (!isNaN(e[0]) && !isNaN(e[1])) {
10.        e = [parseInt(e[0] + e[1]), ...e.slice(2)];
11.    }
12.    let b = parseInt(setting.b);
13.    let o = parseInt(setting.o);
14.    let v;
15.    if (e.length == 1) {
```

```
        v = e[0];
    } else if (e.length == 2) {
        if (isNaN(e[0])) {
            v = e[0];
            o = parseInt(e[1]);
        } else {
            b = b * parseInt(e[0]) / 4;
            v = e[1];
        }
    } else if (e.length == 3) {
        b = b * parseInt(e[0]) / 4;
        v = e[1];
        o = parseInt(e[2]);
    }
    let base = 1;
    let ratio = 1;
    for (let i = 0; i < d; i++) {
        base /= 2;
        ratio += base;
    }
    b /= ratio;
    b = Math.round(b);
    return {
        b,
        v,
        o
    };
});
console.log(data);
data = data.map((e, i, a) => {
    if (i == 0) {
        return `${e.o}S${e.b}${e.v}`;
    } else {
        let pe = a[i - 1];
        let sb = e.v;
        if (pe.b != e.b) {
            sb = 'S' + e.b + sb;
        }
        if (pe.o != e.o) {
            sb = e.o + sb;
        }
                return sb;
    }
}).join('');
console.log('Name: ' + name);
console.log('Setting: ' + JSON.stringify(setting));
console.log('Data: ' + data);
```

<MUSIC.txt>

Super Mario - Main Theme:
5S125aS167fS500cdfpS125fS500dcpfpfp6S250c5S167aS125gS500cS125aS167fS500cdfpS125fS500dcpfpAagS63fS500pS167afS250cS167aS125fS500GfcpS167GS63gS167afS250cS167aS125fS500GfS250c6S63c

Super Mario - Title Music:
7S250ddd6d7ddd6d7S63DS250d5S125pS1000p6S250dbbbdbbbdbbbS500b7c6S125bS250adaaadaaadaaaS500abS125aS250gdbbbdbbbdbbbS500ab7S125ceS250ddd6d7ccc6FS63g

SMBtheme:
6S400ee5S800p6S200eS400cS200eg5pgp6c5S400pS200gS400pS200eS400pS200abS400AS200aS267g6S400egS200aS400fS200geS400cd5S200bS400p6S200c5S400pS200gS400pS200eS400pS200abS400AS200aS267g6S400egS200aS400fS200geS400cd5S200bp6S400gFfD5p6e5pGa6c5pa6cd5S200p6S400gFfD5p6e5p7c5p7cc5S100p6S400gFfD5p6e5pGa6c5pa6cd5S200p6S400D5S200p6S400d5S200p6S400c

SMBwater:
5S225deFgaAbbb6p5b6p5S113b6S225p5g6S75eDeS225p5gab6cdS75eS113DS225fS75eS113pS225p5g6S75dCdS225p5gab6cCS75d5S113g6S225fS75eS113pS225p5g6S75gggS225gapgS75fffS225fgpfS75e5S225ab6feeS150e5S225b6S75c

Picaxe:
5S101g6cS202cS101cedS202cS101dS202edS101cS202cS101egS51aS101agS202eS101ecdS202cS101dS202edS101c5S202aS101ag6S51c

The Simpsons: 6S107cS160eFS320aS107gS160ec5S320aFFFS80gS320ppFFFgS107A6S320cccS160c

Indiana:
5S250eS500pfgp6S63c5S333pS250dS500peS63fS167pS250gS500pabp6S63f5S250paS500pb6S125cde5S250eS500pfgp6S63c5S250p6d5S500p6eS42f5S250gS500pg6S167e5S500p6S250d5S500pg6S167e5S500p6S250d5S500pg6S167f5S500p6S250e5S500p6dS125c

TakeOnMe:
5S320FFFd4pbp5e4p5e4p5eGGabaaae4p5d4p5F4p5F4p5FeeFeFFFd4pbp5e4p5e4p5eGGabaaae4p5d4p5F4p5F4p5Fee

Entertainer:
5S280dDe6S140c5S280e6S140c5S280e6S47cS280cdDecdS140e5S280b6S140dS70c5S140pS280dDe6S140c5S280e6S140c5S280e6S47c5S280pagFa6cS140eS280dc5a6S70d

Muppets:
6S250cc5abS500aS250bgp6cc5aS500bapS167gS250peegfS500eS250f6S500c5cdS250eS500eepeS250gS125p6S250cc5abS500aS250bgp6cc5aS500bS250aS167gS250peegfS500eS250f6S500c5cdS250eS500eS250dS500dS250c

Xfiles:
5S125ebab6d5S42bS31pS125ebab6e5S42bS31p6S125gFede5S42bS31p6S125gFedF5S42bS31pS125ebab6d5S42bS31pS125ebab6e5S42bS31p6S125e5S42b

Looney: 5S1120p6S140cS280fedc5S93a6S280cfedDS93eS280eecdcecd5a6c5gAaf

Bond:
5S640p6S320CS640DDS320DS160DS320CCCCS640eeS320eS160eS320DDDCS640DDS320DS160DS320CCCCS640eeS320eS160eS320DdC7CS53c6S320GFS53G

MASH:
5S140agFgpFpgpFpS47eS140pFeFeFpepS93dS140pFedepdpepdpS47CS140pdCdCdpepFpapbabpapbpS47aS140pabababpS47aS140aFabp6d5p6S93eS140d5bpapS70b

StarWars:
5S360pFFFS60b6FS360eDCS60bS120FS360eDCS60bS120FS360eDeS60C5S360FFFS60b6FS360eDCS60bS120FS360eDCS60bS120FS360eDeS90C

GoodBad:
5S448pA6D5A6D5S75AS149FGS56DS448A6D5A6D5S75AS149FG6S56C5S448A6D5A6D5S75AS149FS299fDS56CS448A6D5A6D5S75AS149GS56D

TopGun:
4S248pS124CGGS248FfFfS124DDS248CDS124fS248DfS124FS248fCS124fS31DS124CGGS248FfFfS124DDS248CDS124fS248DfS124FS248fCS31G

A-Team:
6S125D5A6S63D5S500pS125GAS83DS125pS500gA6S125D5A6fS63D5S500p6S83CS500c5AS83GS63A

Flinstones:
5S320p6S160f5A6AS320gS160f5S107A6S160fS320DddDf5S160A6cS40dS160f5S107A6S160AS320gS160f5S107A6S320ffDddDf5S160A6c5S40A6S160aS107dS160AS320aagFaS80gS160gS107cS320aaggfegS80fS160f5S107A6S160AS320gS160f5S107A6S160fS320DddDf5S160A6S107cS320dDf5S160A6S107cS320dDfS160A7c6S53A

Jeopardy:
6S125cfc5f6cfS63cS125cfcfS83aS250gfedCS125cfc5f6cfS63cS83fS250dS125c5Aagf6pDGD5G6DGS63DS125DGDG7S83c6S250AGgfeS125DGD5G6DGS63DS83GS250fS125DCcp5A6p5S83G6S125DG

Gadget: 5S400DfFGS50AFafGFS400DfFGS50A6Dd5S400DfFGS50AFafGFS100D

MahnaMahna:
6S125CS83c5S125bS167A6fS125GAS83gS125DS250pS125CS83c5S125bS167A6fS83GS167AS125gS250pS125CS83c5S125bS167A6fS125GfS83gS167DS125fS83gS167DS125fS250gS167DS125fS250gS125DS250c5S125A6S167DDS125DS167D