07주차

아두이노 출력 모듈

: 능동부저 수동부저

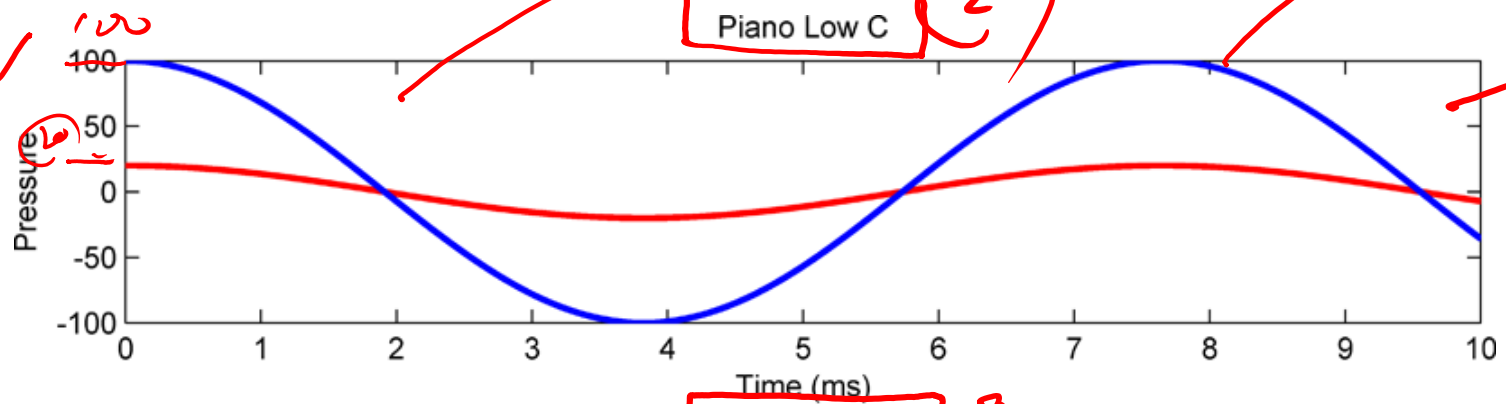
목차

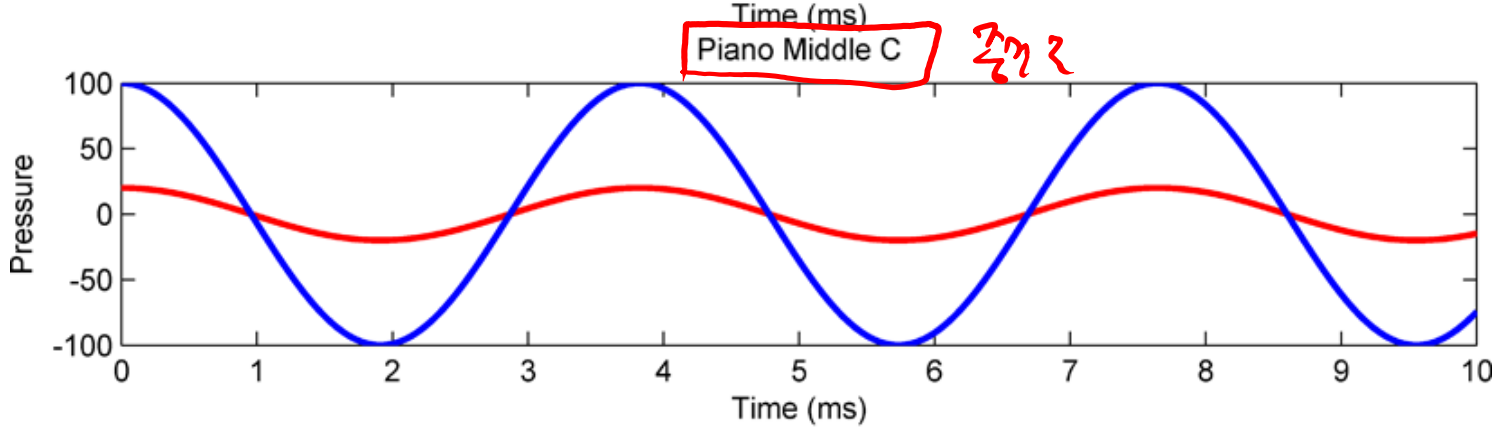
01 스피커 작동 원리

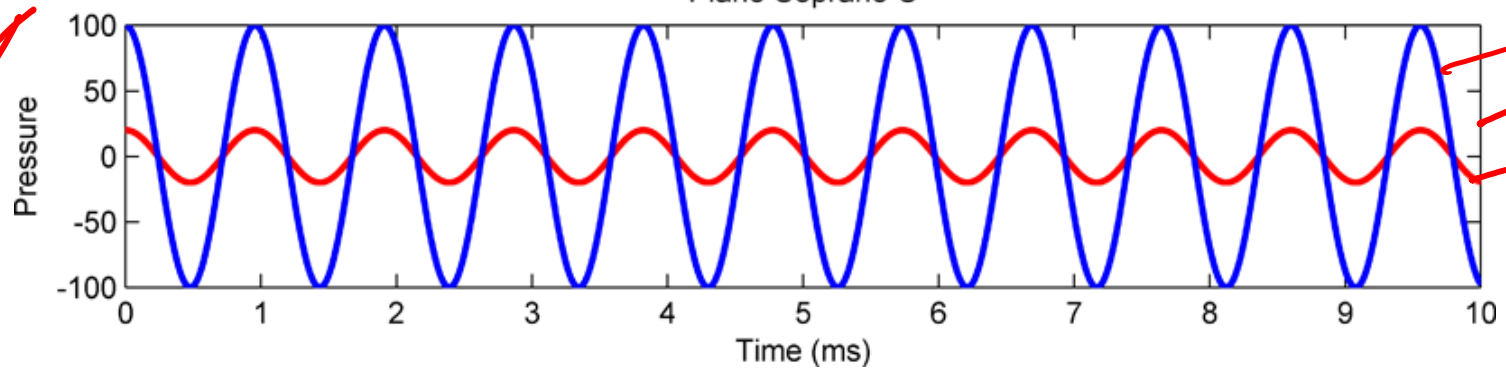
음파의 속성

- 압력파(pressure wave) : 압력 크기의 변화가 만들어 내어 매개물을 통해 전달되는 파동
  - 소리는 공기로 전달되는 압력파
  - 스피커, 드럼, 종 등 물체가 진동하면 주변의 공기도 진동
  - 공기 분자는 진동하면서 주변 분자에 에너지를 전달하고 주변의 공기 분자도 함께 진동
  - 진동 입자의 연쇄반응을 통해 인간의 고막으로 소리를 전달
- 주파수(frequency): 공기 입자가 얼마나 빨리 앞뒤로 진동하는지 나타내는 것
  - 주파수가 높아질 수록 음은 높아지고, 낮아질 수록 음도 낮아짐
  - 중간 도 음의 주파수=261.63Hz
    - 1초 동안 정확히 261.63번 진동하면서 도 음을 만든다는 뜻
- 진폭(amplitude): 진동의 크기
  - 진폭이 클 수록 소리가 크고, 진폭이 작을 수록 소리가 작음
  - 진폭은 스피커로 흐르는 전류의 양으로 조절 가능

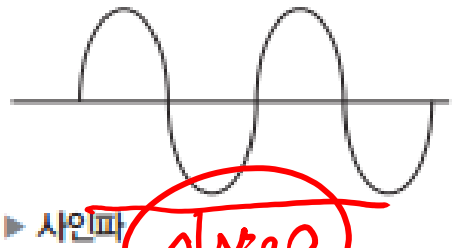
다양한 주파수와 진폭의 음파



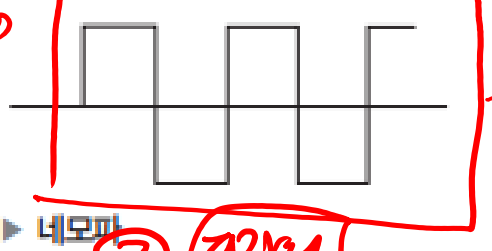




사인파와 네모파



▶ 사인파



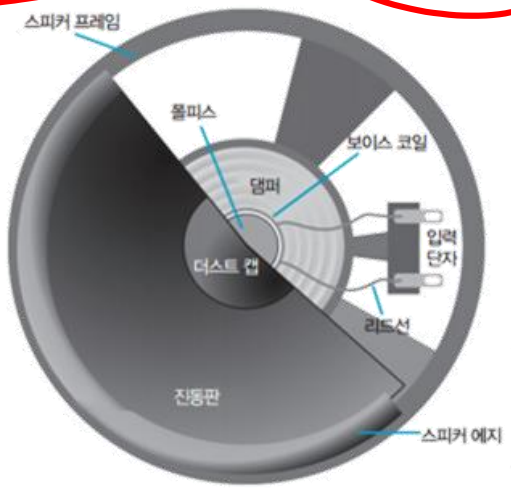
▶ 네모파

- 실제 소리는 사인 곡선 형태의 파형으로 나타냄 ➔ 아날로그 파형
- 아두이노 우노는 실제 소리와 비슷한 사인파를 만들지 못함
- 네모파로도 소리를 만드는 압력파를 생성할 수는 있음 ➔ 디지털 파형(네모파)
  - 디지털 주기 파형으로 HIGH값과 LOW값 사이 를 즉시 바꾸는 형태
- 아날로그 형태의 사인파처럼 소리가 예쁘지는 않다.
- MP3 등 음악이나 영화도 고해상도의 네모파로 만든 사인파
- DAC(Digital to Analog Converter)를 사용하면 고해상도의 사인파를 만들 수 있고, 컴퓨터에 저장된 모든 음악 파일은 이런 고해상도의 디지털 사인파로 구성된 것

스피커의 구조

스피커의 내부 구조

- 모터와 마찬가지로 스피커도 전기를 운동으로 바꾸기 위해 전자기력을 사용
- 전압 신호(사인파 또는 네모파)가 보이스 코일로 보내지면 전류의 변화가 시작되어 자기장을 유도
- 이 자기장의 영향으로 영구자석이 보이스 코일을 끌어당겼다 밀어냈다 하면서 폴피스와 보이스코일에 연결된 진동판을 앞뒤로 떨리게 함
- 앞뒤로 움직이는 이 떨림이 스피커 앞의 공기를 진동시켜 음파가 생성되어 우리 귀까지 소리 전달

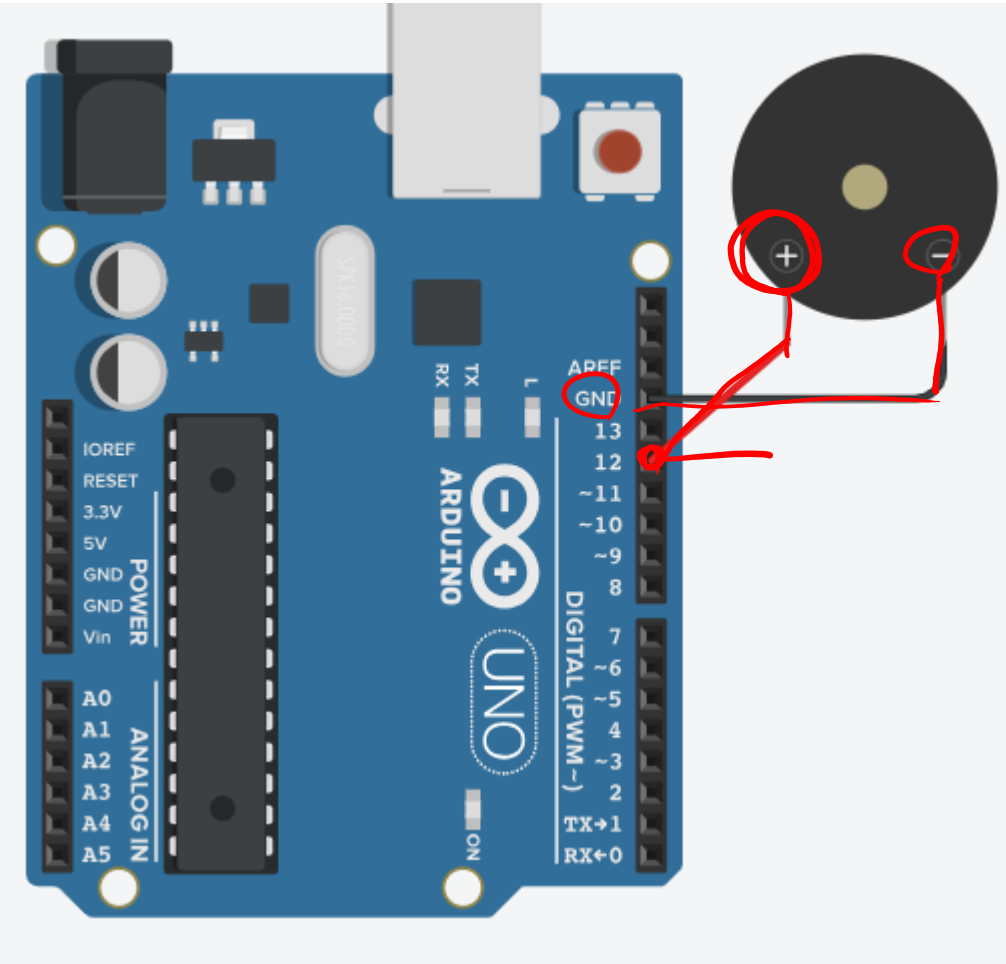


부저

- 부저 (Buzzer, beeper)
  - 단순한 소리를 내는 스피커 같은 장치
  - 간단하게 소리를 생성 가능
  - 아두이노 디지털 핀에 연결해서 디지털 출력으로 제어 가능
- 능동부저 : 전류만 흐르면 자동으로 소리 출력
- 수동부저 (피에조 부저) : 소리 + 헤르츠 지정
  - 아두이노 uno의 경우 31~65,535Hz의 음 출력 가능

02 능동부저

능동부저 회로도


IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
TX
RX
L
ARDUINO
UNO
ON
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)

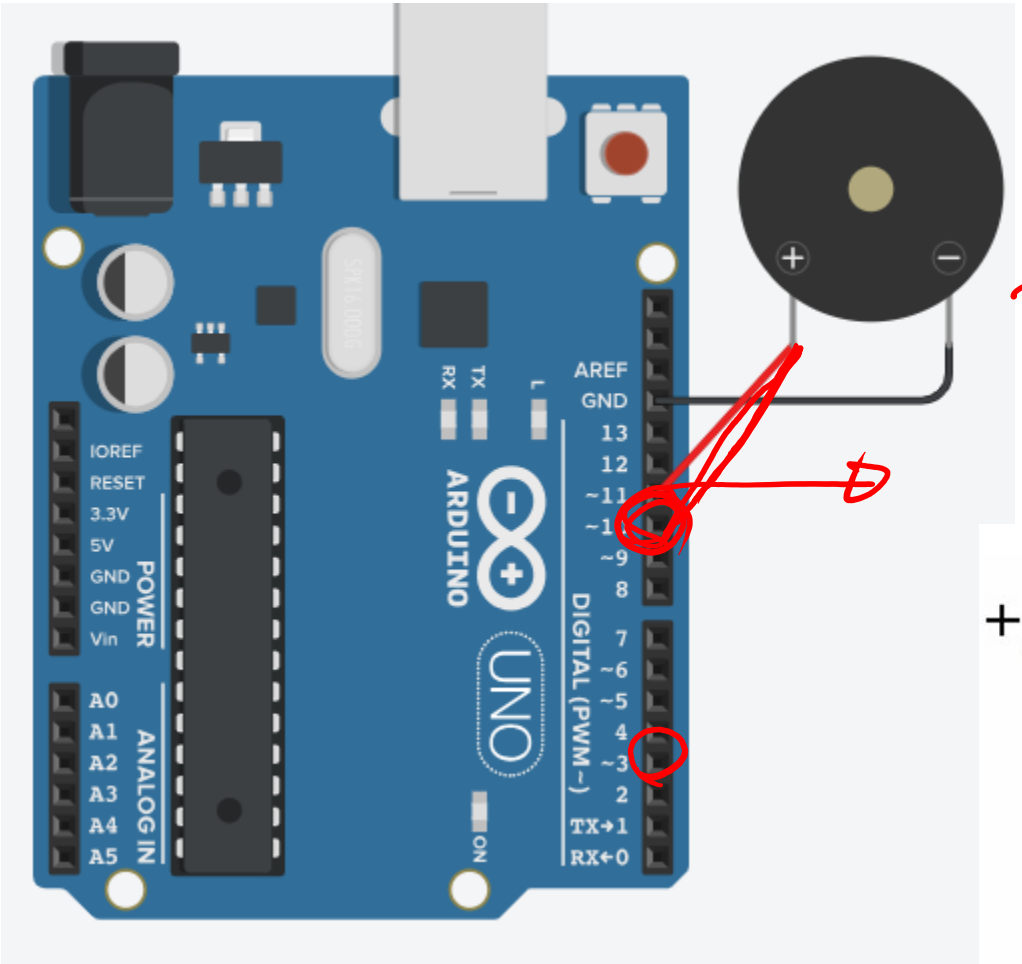

REMOVE SEAL AFTER WASHING
긴 쪽 +
짧은 쪽 -

uno_ex07_01: 능동부저 예제

```
1   int piezo = 12;
2
3   void setup() {
4     pinMode(piezo, OUTPUT);
5   }
6
7   void loop() {
8
9     digitalWrite(piezo, HIGH);
10    delay(1000);
11    digitalWrite(piezo, LOW);
12    delay(1000);
13
14  }
```

03 수동부저

수동부저 회로도


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
ARDUINO
UNO
ON
TX
RX
L

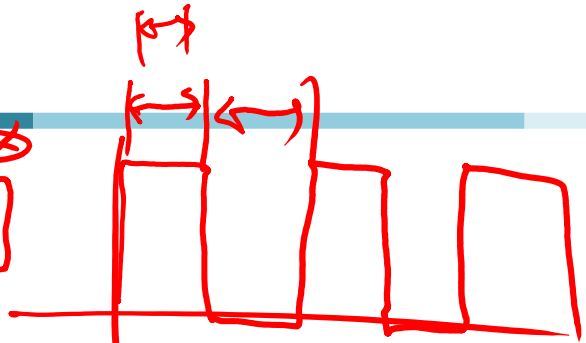

+/- 극성 표시되어 있음

tone( ) 함수

- tone 함수
  - 지정한 핀으로 지정한 주파수의 구형파(네모파) 출력
  - 매개변수
    - 첫 번째 : 구형파를 출력할 핀
    - 두 번째 : 재생할 음의 주파수
    - 세 번째 : 음 재생 시간, 생략하면 noTone() 함수 호출 때까지 재생 계속
  - 하드웨어 타이머를 사용하는 논블로킹(non-blocking) 함수 ➔ "논블로킹"은 코드가 멈추지 않는다는 뜻
    - 재생을 시작하면 바로 반환하며, 다른 작업을 수행할 수 있음

tone() 함수는 이 하드웨어 타이머를 이용해
예를 들어 440Hz라면,
→ 1초에 440번 핀을 HIGH/LOW로 바꿔서 스피커를 진동함
이게 바로 소리의 진동(=주파수) 이 됨

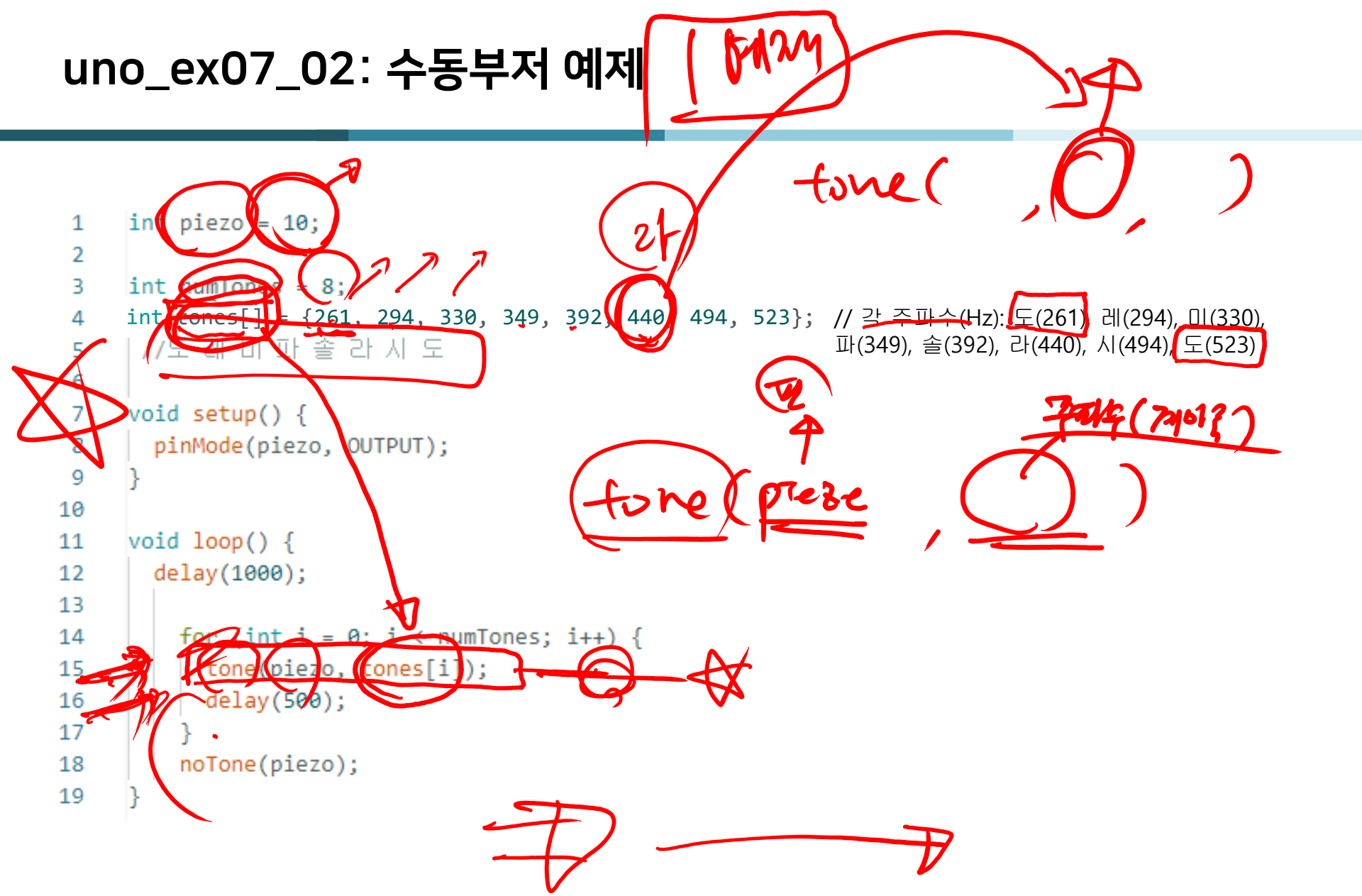
4. tone( ) 함수의 한계



- tone( ) 함수는 PWM 신호를 사용
  - 하드웨어 타이머 사용
  - 3번과 11번 핀의 PWM 신호 출력은 사용할 수 없음(* 테스트상 되긴 함, 가급적 사용 금지)
- tone( ) 함수는 구형파를 사용
  - 구형파로 스피커 구동은 가능하지만, 정현파만큼 자연스럽지는 않음
    - 아두이노 듀에와 같이 DAC이 포함된 아두이노 보드 사용
    - 음악 파일 재생을 위한 확장 실드 사용
- tone( ) 함수는 한 번에 하나의 핀으로만 음 재생 가능
  - 여러 개 스피커를 구동하기 위해서는 타이머 관련 인터럽트를 직접 제어해야 할 수 있음

uno_ex07_02: 수동부저 예제

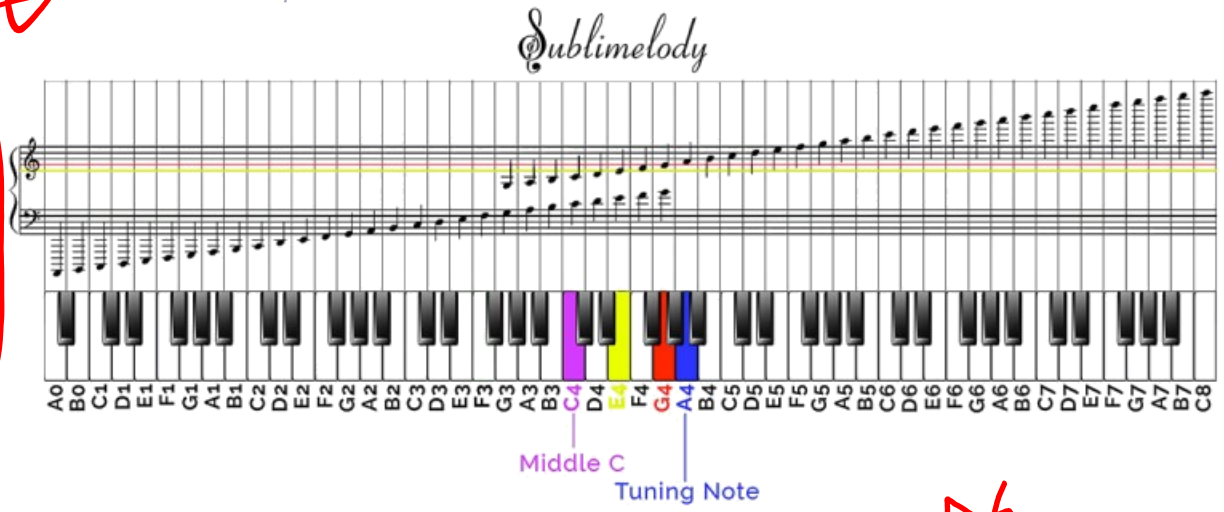
```
1   int piezo = 10;
2
3   int numTones = 8;
4   int tones[] = {261, 294, 330, 349, 392, 440, 494, 523};  // 각 주파수(Hz): 도(261), 레(294), 미(330), 파(349), 솔(392), 라(440), 시(494), 도(523)
5     //도 레 미 파 솔 라 시 도
6
7   void setup() {
8     pinMode(piezo, OUTPUT);
9   }
10
11  void loop() {
12    delay(1000);
13
14      for (int i = 0; i < numTones; i++) {
15        tone(piezo, tones[i]);
16        delay(500);
17      }
18      noTone(piezo);
19  }
```



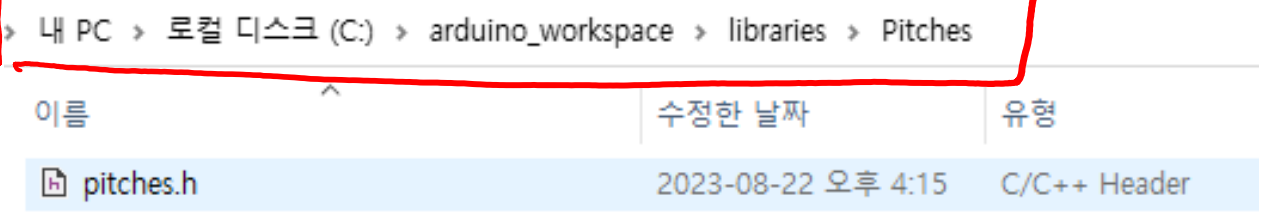
“pitches” 라이브러리 추가

- pitches.h : 음계를 저장한 헤더파일

```
/*************************************************
 * Public Constants
 * pitches.h
 *************************************************/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
   ⋮
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```



- “Pitches” 폴더 생성 후 하위에 “pitches.h” 파일 추가

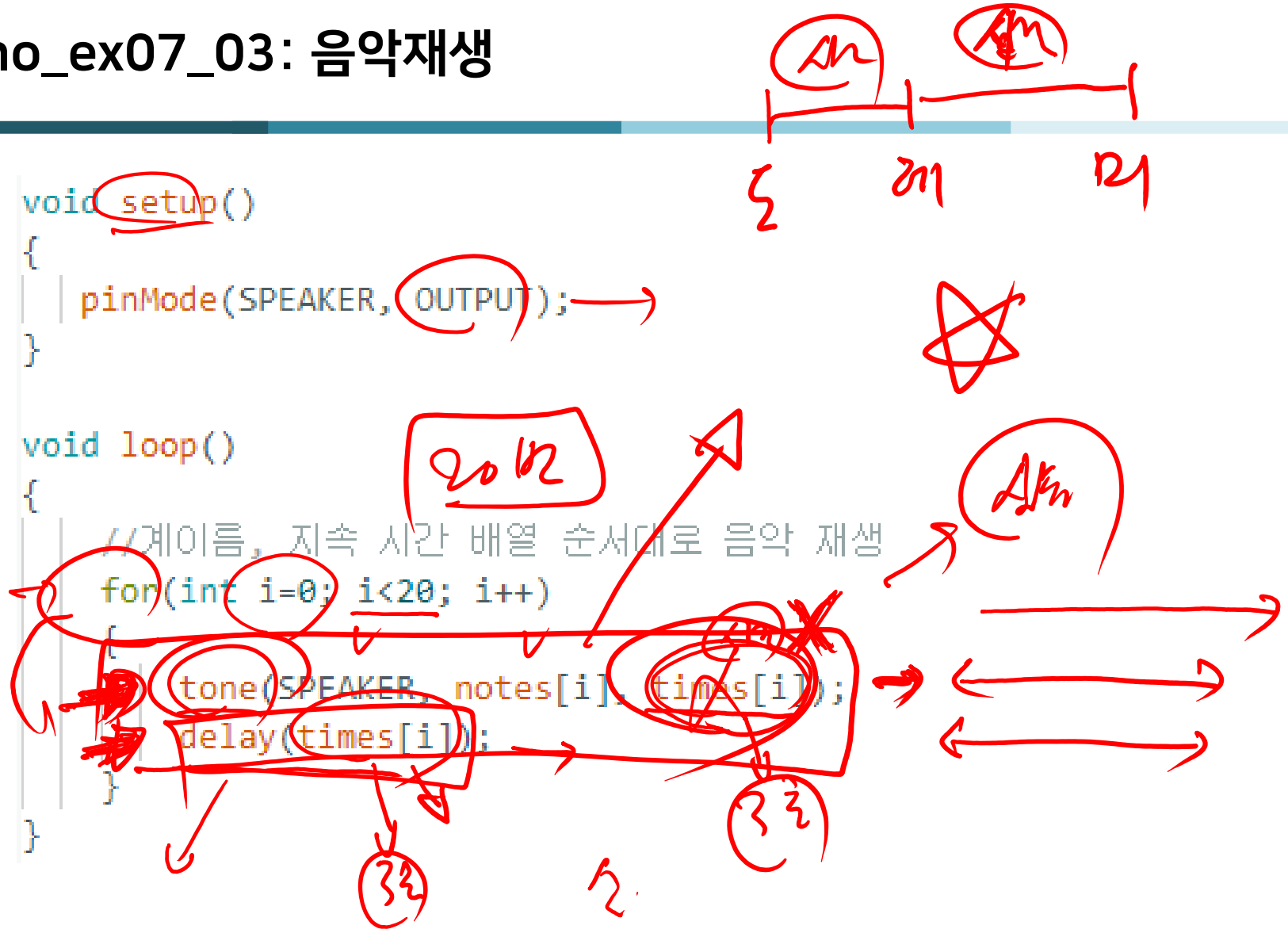


uno_ex07_03: 음악재생

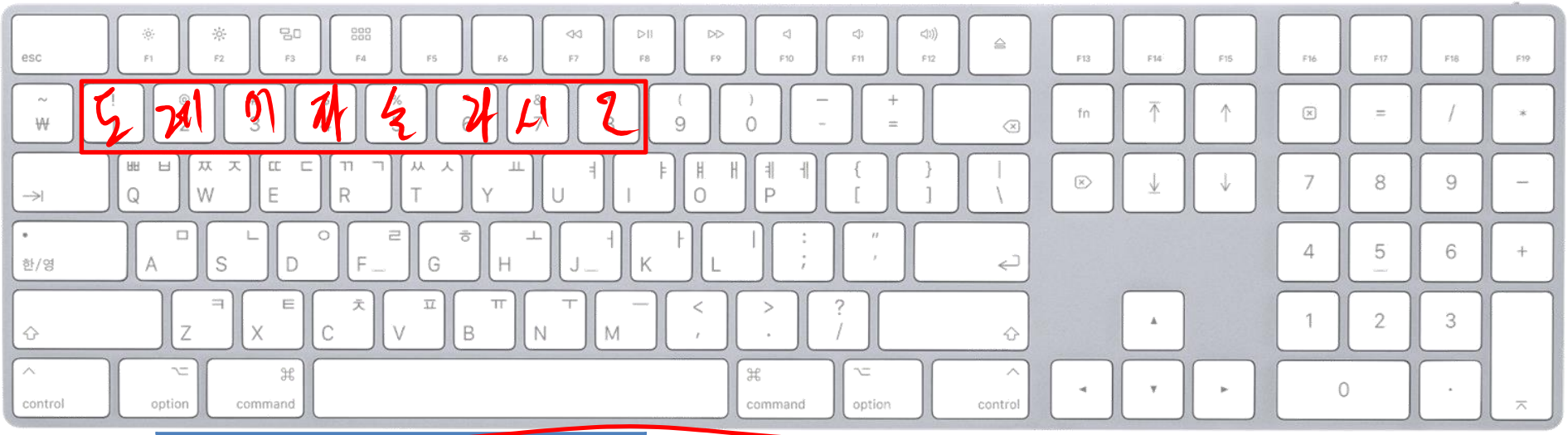
```
1  #include <pitches.h> //계이름이 정의된 헤더파일 삽입
2
3  const int SPEAKER=10; //10번 핀을 사용하는 SPEAKER 상수 정의
4
5  //연주할 음이 저장된 배열 notes 선언
6  int notes[]= {
7      NOTE_A4, NOTE_E3, NOTE_A4, 0,
8      NOTE_A4, NOTE_E3, NOTE_A4, 0,
9      NOTE_E4, NOTE_D4, NOTE_C4, NOTE_B4, NOTE_A4, NOTE_B4, NOTE_C4, NOTE_D4,
10     NOTE_E4, NOTE_E3, NOTE_A4, 0
11 };
12
13 //음의 시작 시간이 저장된 배열 times 선언. 단위는 ms
14 int times[]= {
15     250, 250, 250, 250,
16     250, 250, 250, 250,
17     125, 125, 125, 125, 125, 125, 125, 125,
18     250, 250, 250, 250
19 };
```

uno_ex07_03: 음악재생

```
21  void setup()
22  {
23      pinMode(SPEAKER, OUTPUT);
24  }
25
26  void loop()
27  {
28      //계이름, 지속 시간 배열 순서대로 음악 재생
29      for(int i=0; i<20; i++)
30      {
31          tone(SPEAKER, notes[i], times[i]);
32          delay(times[i]);
33      }
34  }
```



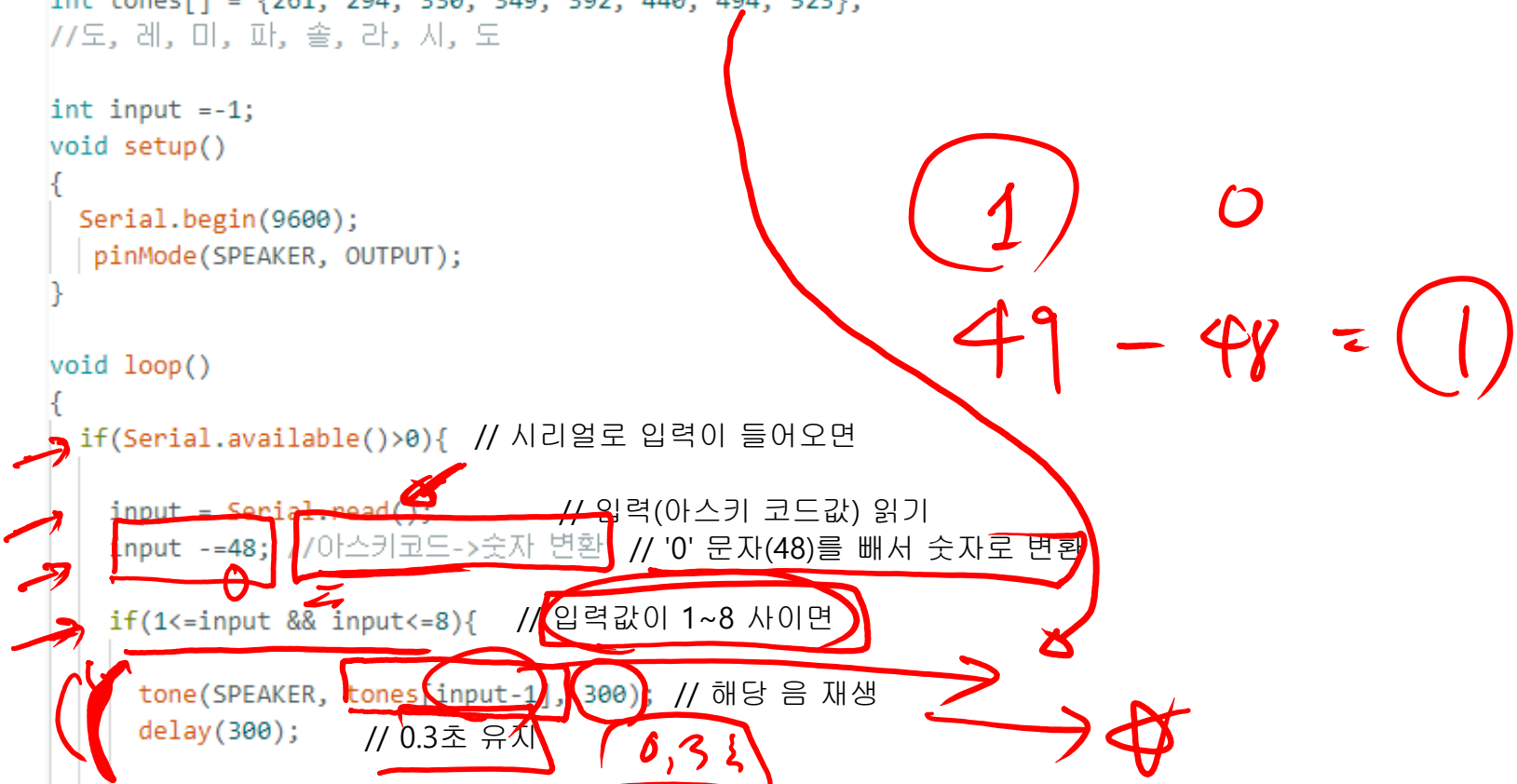
uno_ex07_04: 키보드 피아노



키보드	음계	아스키코드
1	도	49
2	레	50
3	미	51
4	파	52
5	솔	53
6	라	54
7	시	55
8	도	56

uno_ex07_04: 키보드 피아노

```
#include <pitches.h> //계이름이 정의된 헤더 파일 삽입

const int SPEAKER=10; //10번 핀을 사용하는 SPEAKER 상수 정의

int tones[] = {261, 294, 330, 349, 392, 440, 494, 523};
//도, 레, 미, 파, 솔, 라, 시, 도

int input =-1;
void setup()
{
  Serial.begin(9600);
   pinMode(SPEAKER, OUTPUT);
}

void loop()
{
  if(Serial.available()>0){  // 시리얼로 입력이 들어오면

    input = Serial.read();          // 입력(아스키 코드값) 읽기
    input -=48; //아스키코드->숫자 변환  // '0' 문자(48)를 빼서 숫자로 변환

    if(1<=input && input<=8){   // 입력값이 1~8 사이면

      tone(SPEAKER, tones[input-1], 300); // 해당 음 재생
      delay(300);       // 0.3초 유지

    }else if(input != -38) Serial.println("wrong input");

  }

}
```



Thank You