

Algorithm & Code Review

Sound Parameter 측정 알고리즘 & 챗봇 API

알고리즘 및 코드 리뷰





목 차

- 전체 코드 구성/알고리즘
- floornoise-sound-tool
 - 코드리뷰
- floornoise-chatbot-API
 - 코드리뷰



코드 구성

- laMax, laEq, threshold 초과 횟수는 음원을 시간 단위로 분할하여 계산
- Noise Level은 측정된 laMax를 기반으로 측정
- L-Index는 음원의 특정 주파수를 추출하여 계산
- floornoise-sound-tool에서는 음원의 parameter를 계산하고 데스크탑에서 음원의 그래프를 뷰잉
- floornoise-chatbot-API에서는 수집된 음원의 parameter를 바탕으로 noise level, l-index 등을 산정하고 사용자에게 action 제공

추가 개발 방향

- 사용자에게 제공하는 문구나 noise level/l-index/sensitivity 관련된 지표 등을 코드에 직접 삽입하는 것이 아닌 외부 엑셀 파일을 통해 관리할 수 있도록 개발
- 모바일 앱에서 분할되어 넘어오는 음원 조각 각각의 parameter를 계산하고 서버에 적재한 후 전체 음원의 parameter를 계산하도록 설계

laMax, laEq 측정

음원 Frame 분할

Frame별 laEQ, laMAX 측정

최종 laEQ, laMAX 측정값 업데이트

laMax, laEq 측정

- Laeq, LaMax 측정

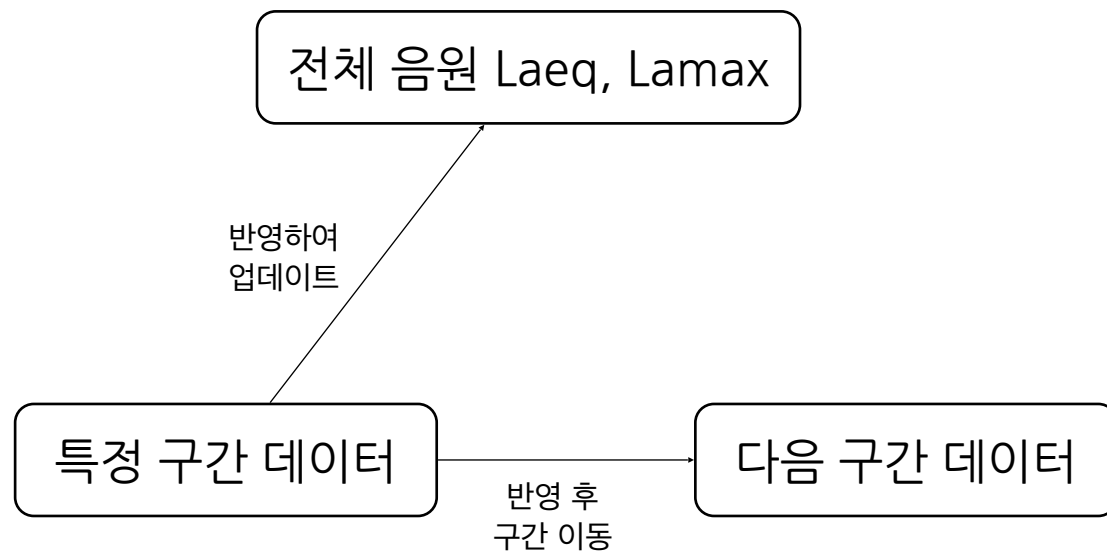
전역 변수로 laeq, lmax 초기값(default) 설정

주어진 음원을 여러 개의 frame으로 분할

첫 frame부터 해당 구간의 laeq, lmax를 측정

이전 구간까지의 측정 결과를 반영하여
Laeq, lmax 값을 업데이트하고 다음 구간으로 이동

마지막 frame의 데이터까지 반영된 결과를
음원의 laeq, lmax 값으로 결정



threshold 초과 횟수 측정

음원 Frame 분할

frame을 이동하며 Threshold(임계치) 측정

최종 threshold 초과 횟수 업데이트

threshold 초과 횟수 측정

- 총간소음 기준치 초과 횟수 측정

전역 변수로 threshold 초과 횟수(default=0) 설정

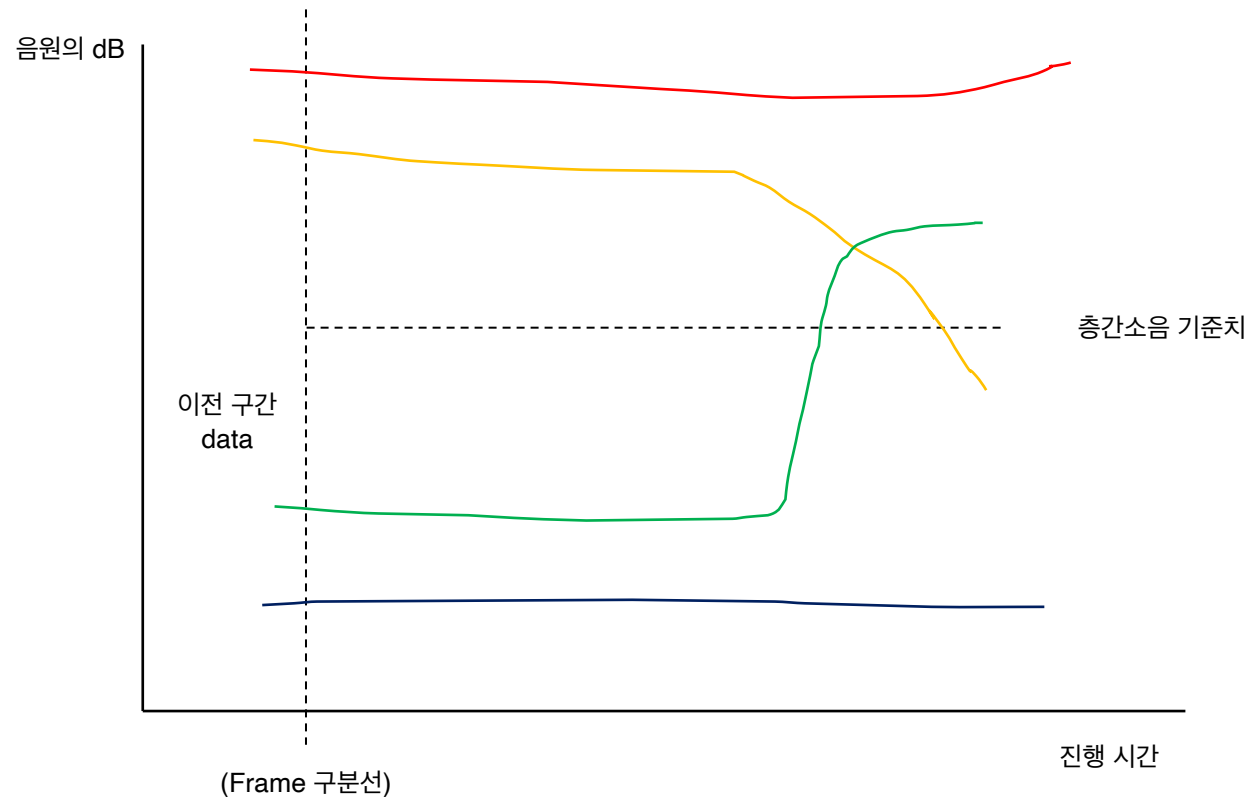
주어진 음원을 여러 개의 frame으로 분할

구간 시작점에서 음원의 dB이
총간소음 기준치를 초과하는가?

구간 내에서 음원의 dB이
기준치 미만으로 내려갔는지 판
별

구간 내에서 음원의 dB이
기준치를 초과하는지 판별

마지막 frame의 데이터까지 반영된 결과를
음원 내 총간소음 기준치 초과 횟수로 결정



Case1: 구간 시작점에서 기준치 초과 O, 구간 내 기준치 미만으로 내려가지 않음

Case2: 구간 시작점에서 기준치 초과 O, 구간 내 기준치 미만으로 내려감 (초과 횟수 +1)

Case3: 구간 시작점에서 기준치 초과 X, 구간 내에서 기준치를 초과함

Case4: 구간 시작점에서 기준치 초과 X, 구간 내에서 기준치를 초과하지 않음

“총간소음 기준치를 최초로 통과한 지점부터
다시 기준치 미만으로 내려간 지점까지를
1번의 초과 횟수로 count”

L-Index 측정

$$dB = 10 \times \log_{10}(amplitude)$$

특정 주파수 대역의 음원 분리

- 음원 내 63/125/250/500Hz의 음성만을 추출

각 주파수 대역에서의 amplitude/magnitude 측정

- 각 음성의 amplitude/magnitude 측정

측정결과 변환 및 I-index 판별기준과 비교

- Magnitude-> db 변환 후 I-index 판별 기준과 비교

최종 I-index 판별

- 주파수별 I-index 수치의 최대치를 음원의 I-index로 설정

L-index 관련 레퍼런스

표 4 일본건축학회 기준

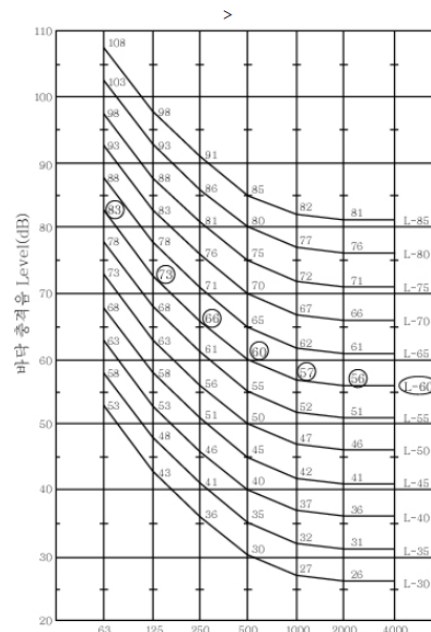
	바닥충격음		
	뛰는 행위, 발걸음 소리 등	의자, 물건의 떨어지는 소리	기타의 예
L-30	거의 들리지 않는다	전혀 들리지 않는다	어린이가 크게 소리쳐도 괜찮다
L-35	조용할 때 들린다	들리지 않는다	다소 뛰며 다녀도 된다
L-40	멀리서 들리는 느낌	거의 들리지 않는다	신경 쓰지 않고 생활할 수 있다
L-45	들릴지언정 거슬리지 않는다	샌들소리는 들린다	약간 알 수 있다
L-50	거의 거슬리지 않는다	칼소리는 들린다	약간 주의하면서 생활 한다
L-55	약간 마음이 쓰인다	슬리퍼 소리도 들린다	주의하면 문제없다
L-60	약간 거슬린다	수저를 떨어뜨려도 들린다	상호간에 결널 수 있는 정도
L-65	잘 들려 거슬린다	동전이 떨어져도 들린다	어린이가 있으면 꾸짖게 된다
L-70	매우 잘 들려 거슬린다	1원짜리 동전일지라도 들린다	어른들도 마음을 쓰게 된다
L-75	매우 귀찮다	좌동	주의하여도 시비가 온다
L-80	시끄러워 견딜 수 없다	좌동	참는 생활이 필요
비고	저음역의 음, 타이어 값	고음역의 음, 태핑머신의 값	타이어, 태핑머신, 모두 합격시

Table 2
Seven selected statements based on the EAI scale for floor impact sound.

성능등급	Class	Scale value	Statement
성능 수준	A	0.6	The indoor environment is quiet.
	B	1.4	Noise does not bother me.
	C	2.2	If you have patience, noise problems can be handled.
	D	3.0	We need to be considerate about each other.
차음성	E	3.9	Noise disturbs my relaxation time.
	F	4.5	Indoor conversation cannot be heard due to noise from upstairs neighbors.
	G	5.3	It is impossible for people to live here.

표 2 중량충격음 기준

등급	친환경 건축물 인증제도 가중치	역A특성 가중 바닥충격음 레벨
1급	1.0	$L'_{iFmax,AW} \leq 40$
2급	0.75	$40 < L'_{iFmax,AW} \leq 43$
3급	0.5	$43 < L'_{iFmax,AW} \leq 47$
4급	0.25	$47 < L'_{iFmax,AW} \leq 50$



옥타브 밴드 중심주파수(Hz)
<그림. 일본 L지수 곡선과 각 주파수별 음압 수치>

L - index	63Hz	125Hz	250Hz	500Hz
L-30	53	43	36	30
L-35	58	48	41	35
L-40	63	53	46	40
L-45	68	58	51	45
L-50	73	63	56	50
L-55	78	68	61	55
L-60	83	73	66	60
L-65	88	78	71	65
L-70	93	83	76	70
L-75	98	88	81	75
L-80	103	93	86	80
L-85	108	98	91	85

63, 125, 250, 500Hz의 옥타브밴드 측정치를 L지수곡선에 플로팅하고 그 값이 모든 주파수 대역에서 기준곡선 이하가 되면 최소 기준곡선에 부여된 수치에 의해 차음등급을 표시한다. 이때 각 주파수 대역별 측정결과가 등급곡선보다 최대 2dB까지 상회하는 것을 허용(JIS A 1419)

Noise level 판별

Normal Sensitivity 기준

LAFmax(dBA)	성가심도	문구
40이하	10% 이하	1. 실내 환경은 조용합니다. 외부 소음에 노출되지 않았습니니다.
40~42.5	15% 이하	2. 소음은 나를 괴롭히지 않는다. 소음이 일상생활에 지장을 줄만큼 크지 않습니다.
42.5~48	35% 이하	3. 당신이 인내심을 가지면 소음 문제에 대처할 수 있는 수준입니다. - 소음이 지속되어 괴롭다면, 이웃과 소통하시길 권장합니다.
48~54	65% 이하	4. 우리는 서로를 배려할 필요가 있습니다. - 이웃과 한번 소통할 필요가 있으며, 주의를 부탁드려 보세요. - 이웃에게 측정 결과를 제시해주시고, 이야기해보시면 좋겠습니다.
54~60	85% 이하	5. 소음은 당신의 휴식 시간을 방해합니다. - 층간소음 법적 기준을 초과합니다. 이웃과 적극적으로 소통할 필요가 있습니다. - 만약 소음이 계속 발생한다면 이웃사이 센터에 상담 문의해주세요. - 측정 결과를 토대로, 상담 챗봇에 연결하겠습니다.
60~73	99%이하	6. 위층 이웃의 소음으로 인해 거주 공간에서 대화가 들리지 않습니다. - 소음으로부터 건디기 어려운 상황입니다. 층간소음 법적 기준을 상당히 초과합니다. - 이웃과 소통이 어렵다면, 즉시 이웃사이 센터에 상담 문의해주세요. - 상담 문의를 원한다면 측정 결과를 이웃사이센터에 전달 후, 상담 연결해 드리겠습니다.
73초과	100%	7. 사람들이 여기에 사는 것은 불가능하다.

층간소음의 기준 (제3조 관련)

층간소음의 구분		층간소음의 기준 [단위: dB(A)]	
		주간 (06:00 ~ 22:00)	야간 (22:00 ~ 06:00)
1. 제2조제1호에 따른 직접충격 소음	1분간 등가소음도 (Leq)	43	38
	최고소음도 (Lmax)	57	52
2. 제2조제2호에 따른 공기전달 소음	5분간 등가소음도 (Leq)	45	40

- 비고
- 직접충격 소음은 1분간 등가소음도(Leq) 및 최고소음도(Lmax)로 평가하고, 공기전달 소음은 5분간 등가소음도(Leq)로 평가한다.
 - 위 표의 기준에도 불구하고 「주택법」 제2조제2호에 따른 공동주택으로서 「건축법」 제11조에 따라 건축허가를 받은 공동주택과 2005년 6월 30일 이전에 「주택법」 제16조에 따라 사업승인을 받은 공동주택의 직접충격 소음 기준에 대해서는 위 표 제1호에 따른 기준에 5dB(A)을 더한 값을 적용한다.
 - 층간소음의 측정방법은 「환경분야 시험·검사 등에 관한 법률」 제6조제1항제2호에 따라 환경부장관이 정하여 고시하는 소음·진동 관련 공정시험기준 중 동일 건물 내에서 사업장 소음을 측정하는 방법을 따르되, 1개 지점 이상에서 1시간 이상 측정하여야 한다.
 - 1분간 등가소음도(Leq) 및 5분간 등가소음도(Leq)는 비고 제3호에 따라 측정한 값 중 가장 높은
 - 최고소음도(L

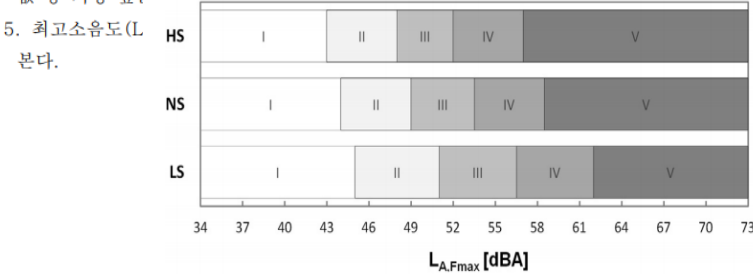


Fig. 7. Classification of floor impact sounds for different noise sensitivity groups (HS: high sensitivity, NS: normal sensitivity, LS: low sensitivity).

Noise level 판별 – sensitivity 관련 자료

질문 제시(어플 초기 실시)

1. 기존 소음 민감도 조사(11점 척도 0~10점, 223명 대상) – 어플 적용시 간소화

S01 나는 주택이 괜찮으면 시끄러운 도로변에 살 아도 괜찮다. (-)

S02 나는 예전에 비해 소음이 더 신경 쓰인다.

S03 누군가가 가끔 음악을 크게 트는 것에 대해 아 무도 언짢아해서는 안 된다. (-)

S04 영화관에서 소곤대는 소리나 사탕봉지 소리 가 신경 쓰인다.

S05 나는 소음으로 인해 쉽게 잠에서 깬다.

S06 내가 공부하고 있는 곳이 시끄럽다면, 나는 문/ 창문을 닫거나 다른 곳으로 옮기려 할 것이다.

S07 나는 내 이웃이 시끄러우면 짜증이 난다.

S08 나는 별 어려움이 없이 대부분의 소음에 익숙 해진다.(-)

S09 나는 소방서 맞은편에 살고 싶지 않다.

S10 이따금씩 소음은 내 신경에 거슬리고 나를 짜 증나게 한다.

S11 내가 집중해야 할 때는 평소애 좋아하는 음악 도 방해가 된다.

S12 나는 내 이웃이 매일같이 내는 소리(발소리, 물소리 등)가 신경 쓰이지 않는다. (-)

S13 내가 혼자 있고 싶을 때 바깥에서 들려오는 소 음이 방해가 된다.

S14 내 주변에 무슨 일이 일어나건 상관없이 나는 집중을 잘 한다.(-)

S15 도서관에서 사람들이 대화를 조용히만 이어 나간다면 나는 상관하지 않는다.(-)

S16 가끔 나는 완벽한 적막속에 있고 싶을 때가 있다.

S17 오토바이 소리가 시끄럽다.

S18 나는 시끄러운 곳에서 휴식을 취하는 것이 힘 들다.

S19 나는 내가 잠이 들거나 일을 하는데 방해가 되 는 소음을 내는 사람들에게 화가 난다.

S20 나는 벽이나 천장, 바닥이 얇은 주택에서 사는 것을 개의치 않는다.(-)

S21 나는 소음에 민감하다.

Table 1

Questionnaire for sensitivity investigation.

Questions
I cannot concentrate well in noisy surroundings.
I complain once I have run out of patience with a noise.
I often desire a quiet environment.
I am annoyed even by low noise levels.
There are often times when I want complete silence.
I get mad when I hear loud music.
When I hear a noise, I picture the source (acts) in my head.
Even music I normally like will bother me if I am trying to concentrate.
Noise disturbs my concentration when reading a newspaper.
I cannot fall asleep easily because of the noise.
I find it hard to relax in a noisy place.
I would not want to live on a noisy street, even if the house/apartment was nice.
I would not like to live across the street from a fire station.
I would not want to live in a house with poor noise insulation.
I am easily awakened by noise.
I am confused by noise.
Noise during a meal makes me uncomfortable.
I worry that my neighbor can hear noise coming from my apartment.
I am uneasy when I hear noise.
I often experience headaches and/or digestive disorders due to noise.

Table 1. Examples of the collected statements

No.	Statement
1	The indoor environment is quiet.
2	It is as if there is no one living upstairs.
3	Noise can't be heard.
4	Live a pleasant life.
5	It is possible to concentrate on things.
6	I can take a nap.
7	It does not bother me.
8	I do not worry about floor-to-floor noise.
9	It is possible to live without being aware of the upstairs neighbors
10	If you have patience, problems can be handled.
11	In any case, complaints do not occur.
12	I go upstairs to complain to the neighbors.
13	Seriously contemplate sound proofing your home.
14	I get angry because of the noise.

512명 피험자 대상 설문(109명 실험실 환경), 7점 척도

설문조사를 통한 소음 민감도에 따른 층간소음에 의한 성가심과 생활방해 연구(2018, 정정호, 이성찬)

Influence of noise sensitivity on annoyance of indoor and outdoor noises in residential buildings (Jeon et. al., 2011, Applied Acoustics)

Noise level 판별

- Annoyance 와 Allowance 간의 상관 관계 분석 및 방정식 도출

$$\%A_{NONE} = 5.21L_{A,Fmax} - 231.37 (R^2 = 0.99, p < 0.01) \quad (2)$$

$$\%A_{HRTF} = 5.31L_{A,Fmax} - 200.84 (R^2 = 0.99, p < 0.01) \quad (3)$$

$$\%A_{HMD} = 5.91L_{A,Fmax} - 265.85 (R^2 = 0.99, p < 0.01) \quad (4)$$

$$\%A_{HRTF+HMD} = 6.36L_{A,Fmax} - 253.21 (R^2 = 0.99, p < 0.01) \quad (5)$$

$$\%AL_{NONE} = -3.81L_{A,Fmax} - 276.39 (R^2 = 0.99, p < 0.01) \quad (6)$$

$$\%AL_{HRTF} = -4.61L_{A,Fmax} - 291.79 (R^2 = 0.99, p < 0.01) \quad (7)$$

$$\%AL_{HMD} = -5.01L_{A,Fmax} - 336.88 (R^2 = 0.99, p < 0.01) \quad (8)$$

$$\%AL_{HRTF+HMD} = -5.61L_{A,Fmax} - 339.16 (R^2 = 0.99, p < 0.01) \quad (9)$$

Table 6
Classification of heavy-weight floor impact sounds in terms of experimental setup.

Class	%A	$L_{A,Fmax}$ [dBA]				Existing Criteria
		NONE	HRTF	HMD	HRTF + HMD	
I	0-20%	< 48.0	< 41.5	< 48.0	< 42.5	Daytime
II	20-40%	< 52.0	< 45.5	< 52.0	< 46.0	
III	40-60%	< 56.0	< 49.0	< 55.0	< 49.0	Night-time
IV	60-80%	< 59.5	< 52.5	< 58.0	< 52.0	
V	80-100%	≥ 59.5	≥ 52.5	≥ 58.0	≥ 52.0	

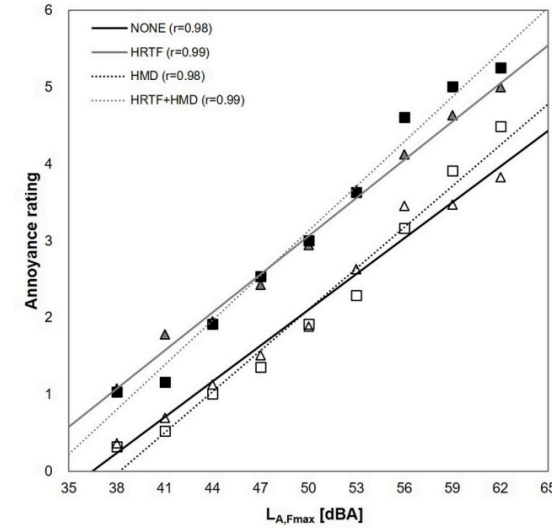


Fig. 6. Mean annoyance ratings as function of SPL (□: NONE, △: HRTF, ■: HMD, ▲: HRTF + HMD).

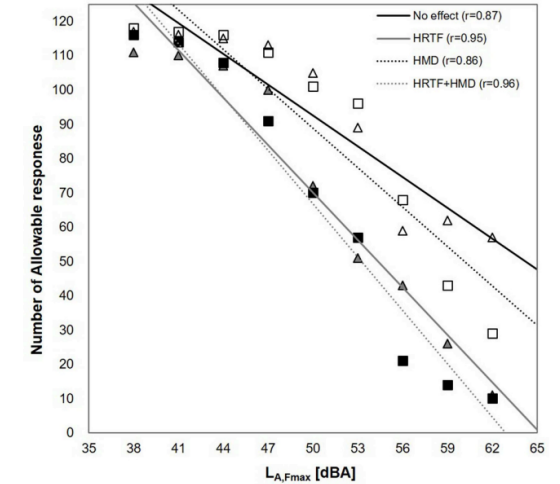


Fig. 7. Number of participant responses indicating allowable noise as function of SPL (□: NONE, △: HRTF, ■: HMD, ▲: HRTF + HMD).

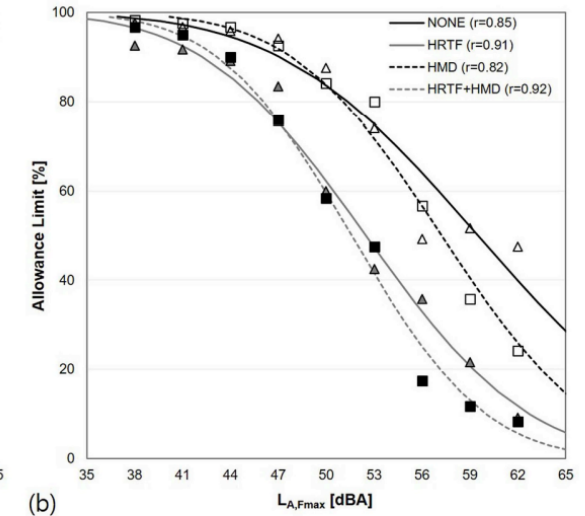
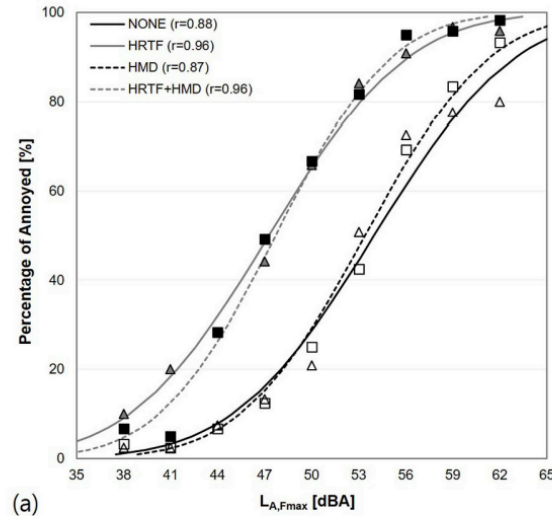


Fig. 9. Percentage of subjects reporting (a) annoyance and (b) allowable noise as function of $L_{A,Fmax}$ for different setups (□: NONE, △: HRTF, ■: HMD, ▲: HRTF + HMD).

챗봇 알고리즘

음원 측정/업로드

- 인앱 녹음기능을 통한 층간소음 음원 녹음
- 휴대폰에서 사용자가 직접 녹음 후 업로드

Sound Quality 측정

- laEQ, laMAX 등 대상 음원의 SQ지표 측정
- SQ 측정/분석 알고리즘

층간소음 정도 분석

- 주파수 분석을 통한 L-Index와 층간소음 기준 기반
- 사용자의 sensitivity 고려

사용자 action에 따른 문구 출력

- 측정/분석된 결과를 토대로 사용자 action에 따른 층간소음 정보 및 문구 제공

코드 구성

수집한 음원의 sound parameter 측정을 위한 테스트 툴

- Desktop GUI 애플리케이션 형태로 제작
- app.py에서 메인 애플리케이션을 실행하고 음원 분석 및 parameter 측정을 위한 각 동작단계에서는 /sound 내의 모듈에 정의된 함수를 호출

app.py

- GUI 애플리케이션 실행, 분석/측정 프로세스 진행에 따른 연쇄적 함수 호출

/sound/level.py

- lmax, laeq 등 sound parameter 측정 및 수치 보정과 관련된 함수들로 구성

/sound/A-weighting.py

- A-weighting 보정에 사용하는 함수

/sound/__init__.py

- app.py에서 함수를 호출하기 이전 오디오 파일의 filename과 프레임 정보 등을 저장

app.py [개요]

```
app.py
floornoise-sound-tool-master > floornoise-sound-tool-master > app.py > ...

1  import librosa
2  import math
3  import numpy as np
4  import csv
5  import tkinter as tk
6  from tkinter import ttk, filedialog as fd, StringVar
7  from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
8  import matplotlib.pyplot as plt
9  from numpy.typing import *
10 from sound import load_audio
11 from sound.level import calculate_rms, rms2db, calculate_lamax, calculate_laeq, find_spike_indices,
12 from sound.weighting import A_weighting
13
14 audio_info = {
15     'generator': iter([]),
16     'wav': np.array([]),
17     'sr': 0,
18     'spl': np.array([]),
19     'len': 1.0,
20     'frame_start_t': 0.0,
21     'processed_t': 0.0,
22 }
23
24 analysis_info = {
25     'beg': 0.0,
26     'end': 0.0,
27     'threshold': 57.0,
28     'spike_indices': np.array([], dtype=np.int32),
29 }
30
31 root = tk.Tk()
32 root.title("Sound Analysis")
33 root.resizable(False, False)
34
35 frame = tk.Frame(root)
36 frame.pack()
37
38 # setup sound pressure level plot
39 figure = plt.Figure(figsize=(13, 5), dpi=120)
40 spl_ax = figure.add_subplot(111)
41 spl_ax.set_title('Sound Pressure Level (dBA)')
42 spl_ax.set_ylabel('dBA')
43 spl_ax.set_xlabel('time (s)')
44 spl_ax.yaxis.grid()
45
46 plot_canvas = FigureCanvasTkAgg(figure, frame)
47 plot_canvas.get_tk_widget().grid(column=0, row=0, columnspan=7, padx=20, pady=10)
```

- GUI 애플리케이션 형태로 바닥충격음 음원을 분석하여 sound parameter를 측정하기 위해 제작된 툴
- app.py에서는 다른 모듈 내에 정의된 함수들을 연쇄적으로 호출하여 **sound parameter를 측정**
- app.py 자체에도 여러 함수들이 내장되어 있으며 알고리즘 파악을 위해서는 연쇄적으로 호출되는 함수들의 flow를 파악하는 과정이 필요
- Python의 tkinter 라이브러리를 통해 GUI 애플리케이션의 형태로 구성되었기 때문에 변수를 선언/초기화하는 방법 등 일부 구간에 한해 스크립트 환경과 차이가 존재(tkinter 애플리케이션의 구성 요소로 활용하기 위함)
- Line14~29: app.py 내에서 활용할 오디오 파일의 정보(파형, 길이 정보 등)을 기본값으로 초기화. Audio_info는 오디오 파일의 정보이며 analysis_info는 분석 중에 활용하는 변수들

app.py [1]

```

app.py
floornoise-sound-tool-master > floornoise-sound-tool-master > app.py > ...
31 root = tk.Tk()
32 root.title("Sound Analysis")
33 root.resizable(False, False)
34
35 frame = tk.Frame(root)
36 frame.pack()
37
38 # setup sound pressure level plot
39 figure = plt.figure(figsize=(13, 5), dpi=120)
40 spl_ax = figure.add_subplot(111)
41 spl_ax.set_title('Sound Pressure Level (dBA)')
42 spl_ax.set_ylabel('dBA')
43 spl_ax.set_xlabel('time (s)')
44 spl_ax.yaxis.grid()
45
46 plot_canvas = FigureCanvasTkAgg(figure, frame)
47 plot_canvas.get_tk_widget().grid(column=0, row=0, columnspan=7, padx=20, pady=10)
48
49 # setup input fields
50 # range input fields
51 tk.Label(frame, text='calculation range (beg)', font=('Arial', 15), fg='black').grid(column=1, row=1)
52 tk.Label(frame, text='calculation range (end)', font=('Arial', 15), fg='black').grid(column=1, row=2)
53 laeq_range_beg_str = StringVar()
54 laeq_range_end_str = StringVar()
55 laeq_range_beg_entry = tk.Entry(frame, textvariable=laeq_range_beg_str)
56 laeq_range_beg_entry.grid(column=2, row=1)
57 laeq_range_end_entry = tk.Entry(frame, textvariable=laeq_range_end_str)
58 laeq_range_end_entry.grid(column=2, row=2)
59
60 laeq_str = StringVar()
61 laeq_str.set('LAeq (dBA): 0')
62 tk.Label(frame, textvariable=laeq_str, font=('Arial', 15), fg='black').grid(column=3, row=2, pady=10)
63
64 # LAmx
65 lamax_str = StringVar()
66 lamax_str.set('LAmx (dBA): 0')
67 tk.Label(frame, textvariable=lamax_str, font=('Arial', 15), fg='black').grid(column=4, row=2, pady=10)
68
69 # threshold input field
70 tk.Label(frame, text='level threshold (dBA)', font=('Arial', 15), fg='black').grid(column=5, row=1)
71 threshold_str = StringVar()
72 threshold_str.set(f"[analysis_info['threshold']]")
73 threshold_entry = tk.Entry(frame, textvariable=threshold_str)
74 threshold_entry.grid(column=5, row=2, padx=10)
75
76 exceed_count_str = StringVar()
77 exceed_count_str.set('Exceed count: 0')
78 tk.Label(frame, textvariable=exceed_count_str, font=('Arial', 15), fg='black').grid(column=6, row=2)

```

- Line31~78
- Tkinter 라이브러리를 통해 **GUI 애플리케이션을 빌드하는 구간**
- 애플리케이션 내 그래프의 라벨과 속성, 버튼, 측정값을 뷰잉할 위치 등을 설정
- StringVar 형태로 일부 변수를 선언하고(ex: laeq_range_beg_str 등) 필요한 함수를 호출하며 **올바른 값으로 업데이트하는 방식** (tkinter 애플리케이션의 요소로 활용하기 위해 StringVar 형태로 설정)

app.py [2]

```
def open_audio_file():  
    filetypes = (  
        ('raw audio file', '.wav'),  
    )  
    filename = fd.askopenfilename(filetypes=filetypes)  
    if filename != '':  
        stream, sr = load_audio(filename, block_len=220)  
        audio_info['generator'] = stream  
        audio_info['sr'] = sr  
        audio_info['processed_t'] = 0.0  
        audio_info['frame_start_t'] = 0.0  
  
        next_audio_frame()
```

- Function open_audio_file()
- GUI 애플리케이션 내에서 'open an audio file' 버튼을 누를 시 호출되는 함수
- Raw audio file 형식 또는 .wav 확장자에 해당하는 파일을 선택할 수 있도록 구성됨 (현재 sound parameter 측정 알고리즘과 향후 서술할 챗봇 API에서 활용하는 라이브러리들이 .wav 확장자에 맞게 설계되어 있기 때문)
- Filename이 공백이 아닐 경우, 즉 파일이 정상적으로 선택되었을 경우 **stream, sr, audio_info**에 해당하는 정보들을 선택된 음원의 정보로 업데이트 (app.py [개요] 참고)
- 분석 진행 시간과 분석 중인 프레임의 시작점 등을 **audio_info**에 업데이트한 후 next_audio_frame 함수 호출

app.py [3]

```
134 def next_audio_frame():
135     wav = next(audio_info['generator'], None)
136     if wav is None:
137         return
138     wav, sr = A_weighting(wav, audio_info['sr'])
139     rms = calculate_rms(wav, sr)
140     spl = rms2db(rms)
141
142     audio_info['wav'] = wav
143     audio_info['spl'] = spl[0]
144     audio_info['len'] = librosa.get_duration(y=wav, sr=sr)
145     audio_info['frame_start_t'] = audio_info['processed_t']
146     audio_info['processed_t'] += audio_info['len']
147
148     laeq_range_beg_str.set(f"{audio_info['frame_start_t']:.7f}")
149     laeq_range_end_str.set(f"{audio_info['processed_t']:.7f}")
150
151     update_laeq_lamax_label(redraw=False)
152     update_threshold(redraw=False)
153     draw_spl()
```

- Function next_audio_frame()
- 현재 sound parameter 측정 알고리즘은 음원의 길이가 길 경우 전체 음원의 s.p를 한번에 측정하는 대신 음원을 여러 개의 frame으로 분할하여 각각의 구간에서 laeq, lamax, threshold(소음의 임계치) 초과 횟수를 측정함
- next_audio_frame()은 애플리케이션에서 분석할 음원을 선택하거나 애플리케이션 내의 next 버튼을 누를 시 호출됨
- A_weighting은 가청 음성과 관련된 보정, calculate_rms는 raw data를 rms 값으로 변환, rms2db는 변환된 rms 수치를 db로 변환하는 함수이며 /sound 디렉토리 내에 정의되어 있음
- 이 함수 내에서 laeq&lamax, threshold 초과 횟수를 업데이트하고 그래프를 다시 그려야 할 경우(다음 구간의 음성이 현재 구간과 다를 경우) 그래프를 업데이트함 (draw_spl)

app.py [4]

```

169 def update_laeq_lamax_label(redraw: bool = True):
170     spl_len = audio_info['spl'].shape[0]
171     if spl_len == 0:
172         return
173
174     start_t = audio_info['frame_start_t']
175
176     try:
177         beg_sec = float(laeq_range_beg_str.get())
178     except ValueError:
179         beg_sec = start_t
180     try:
181         end_sec = float(laeq_range_end_str.get())
182     except ValueError:
183         end_sec = audio_info['len']
184         laeq_range_end_str.set(f"{audio_info['len']}")
185
186     beg_sec = np.clip(beg_sec, start_t, audio_info['processed_t'])
187     end_sec = np.clip(end_sec, start_t, audio_info['processed_t'])
188
189     beg = round(spl_len * ((beg_sec - start_t) / audio_info['len']))
190     end = round(spl_len * ((end_sec - start_t) / audio_info['len']))
191     if beg > end:
192         beg, end = end, beg
193         beg_sec, end_sec = end_sec, beg_sec
194     elif beg == end:
195         end += 1
196
197     beg_sec = (beg / spl_len) * audio_info['len'] + start_t
198     end_sec = (end / spl_len) * audio_info['len'] + start_t
199
200     laeq_range_beg_str.set(f'{beg_sec:.7f}')
201     laeq_range_end_str.set(f'{end_sec:.7f}')
202
203     laeq = calculate_laeq(audio_info['spl'], frames_beg=beg, frames_end=end)
204     laeq_str.set(f'LAeq (dBA): {laeq:.5f}')
205
206     lamax = calculate_lamax(audio_info['spl'], frames_beg=beg, frames_end=end)
207     lamax_str.set(f'LAmax (dBA): {lamax:.5f}')
208
209     analysis_info['beg'] = beg_sec
210     analysis_info['end'] = end_sec
211
212     if redraw:
213         draw_spl()

```

- Function `update_laeq_lamax_label()`
- GUI 애플리케이션 내에서 `laeq`와 `lamax`를 표시하는 라벨의 값을 업데이트하는 함수로 실제 `laeq`와 `lamax`를 계산하는 과정은 line 203, 206에서 호출하는 `/sound/level.py` 모듈의 함수들에 나타남
- 음원을 균일한 사이즈의 **frame**으로 분할하기 위해 프레임의 길이와 시작점, 끝점에 해당하는 시간을 계산하고 해당 구간을 인수로 넘겨 `calculate_laeq`와 `calculate_lamax`를 호출
- 분석한 프레임의 시작점과 끝점 정보를 `analysis_info`에 업데이트하고, 그래프의 위상이나 정보가 달라지므로 `draw_spl` 함수를 호출하여 업데이트함 (`update` 함수가 호출되었다는 것 자체로 이미 그래프 및 s.p에 변동이 있음을 의미)

app.py [5]

```
215 def update_threshold(redraw: bool = True):
216     spl_len = audio_info['spl'].shape[0]
217     if spl_len == 0:
218         return
219
220     try:
221         analysis_info['threshold'] = float(threshold_str.get())
222     except ValueError:
223         analysis_info['threshold'] = 57.0
224         threshold_str.set(f"{analysis_info['threshold']}")
225
226     analysis_info['spike_indices'] = find_spike_indices(audio_info['spl'], analysis_info['threshold'])
227     exceed_count_str.set(f"Exceed count: {analysis_info['spike_indices'].shape[0]}")
228
229     if redraw:
230         draw_spl()
```

- Function update_threshold()
- GUI 애플리케이션 내에서 threshold를 표시하는 라벨의 값을 업데이트하는 함수로 실제 threshold 초과 횟수를 측정하는 과정은 line 226에서 호출하는 /sound/level.py 모듈의 함수(find_spike_indices)에 나타남
- 이전 update_laeq_lamax_label에서 현재 구간의 시작점과 끝점을 설정하였으므로 중복 작업은 없으며 **사용자의 설문조사 결과 및 민감도를 반영하는 threshold의 기준(=소음 임계치 기준)을 사용하여 threshold 초과 횟수를 측정**
- 해당 frame의 threshold 초과 횟수를 analysis_info에 저장하고 그 래프를 업데이트하기 위해 draw_spl()를 호출

app.py [6]

```

96 def get_ticks(times: NDArray, interval: float = 1.0):
97     start_t = audio_info['frame_start_t']
98     end_t = audio_info['processed_t']
99     start_t = math.ceil(start_t) + interval if math.ceil(start_t) - start_t < interval
100     end_t = math.floor(end_t) + interval if end_t - math.floor(end_t) > interval else
101     ticks = np.arange(start_t, end_t, interval)
102
103     start = [] if round(times[0], 1) == round(ticks[0], 1) else [round(times[0], 1)]
104     end = [] if round(times[-1], 1) == round(ticks[-1], 1) else [round(times[-1], 1)]
105
106     ticks = np.concatenate((start, ticks, end))
107     return ticks
108
109 def draw_spl_data(spl: NDArray, sr:int):
110     start_t = audio_info['frame_start_t']
111     times = get_times(spl, sr, start_t)
112
113     spl_ax.cla()
114     spl_ax.plot(times, lamax_adjustment(spl), markevery=analysis_info['spike_indices'])
115
116     spl_ax.axvline(x=analysis_info['beg'], color='orange')
117     spl_ax.axvline(x=analysis_info['end'], color='orange')
118     spl_ax.axhline(y=analysis_info['threshold'], color='red')
119
120     spl_ax.set_xlim(xmin=times[0])
121     spl_ax.set_xlim(xmax=times[-1])
122     spl_ax.set_xticks(get_ticks(times, interval=get_interval(times[-1])))
123
124     spl_ax.set_title('Sound Pressure Level (dBA)')
125     spl_ax.set_ylabel('dBA')
126     spl_ax.set_xlabel('time (s)')
127     spl_ax.yaxis.grid()
128
129     plot_canvas.draw()
130
131 def draw_spl():
132     draw_spl_data(audio_info['spl'], audio_info['sr'])
133

```

- Function get_ticks(), draw_spl_data(), draw_spl(), get_times(), get_interval()
- GUI 애플리케이션에 삽입되는 그래프(time-dB plot과 threshold 초과 지점 뷰잉) 구성을 위한 함수들

```

232 ttk.Button(
233     frame,
234     text='calculate LAeq, LAmax',
235     command=update_laeq_lamax_label,
236 ).grid(column=3, row=1, columnspan=2)
237
238 ttk.Button(
239     frame,
240     text='update threshold',
241     command=update_threshold,
242 ).grid(column=6, row=1, padx=10)
243
244 ttk.Button(
245     frame,
246     text='open an audio file',
247     command=open_audio_file,
248 ).grid(column=0, row=1, padx=10)
249
250 ttk.Button(
251     frame,
252     text='next',
253     command=next_audio_frame,
254 ).grid(column=0, row=2, padx=10)
255
256 root.mainloop()

```

- GUI 애플리케이션 내의 버튼들을 구성하는 부분이며 각 버튼의 기능에 맞게 함수를 호출하는 command를 설정함

/sound/level.py [1]

```

7  # raw data -> rms values 변환하는 함수
8  def calculate_rms(data, sr) :
9      s, phase = librosa.magphase(librosa.stft(data, center=False))
10     rms = librosa.feature.rms(S=s)
11     return rms
12
13 # rms values -> dB values 변환하는 함수
14 def rms2db(rms) :
15     ref = 2 * 10 ** (-5)
16     spl = 10 * np.log10((rms ** 2) / (ref ** 2))
17     return spl
18
19 # LAmax 값 보정을 함수
20 def lamax_adjustment(db) :
21     return db + 10.5006
22
23 # LAeq 값 보정을 함수
24 def laeq_adjustment(db) :
25     return db + 17.3279
26
27 # 주어진 dB values 에서 [frame_beg, frame_end) 범위의 LAmax 계산
28 def calculate_lamax(dba, frames_beg, frames_end) :
29     dba = lamax_adjustment(dba)
30     return np.max(dba[frames_beg:frames_end]).item()
31
32 # 주어진 dB values 에서 [frame_beg, frame_end) 범위의 LAeq 계산
33 def calculate_laeq(dba, frames_beg, frames_end) :
34     dba = laeq_adjustment(dba)
35     return np.mean(dba[frames_beg:frames_end]).item()
36
37 # dB values 를 받아서 피크 인덱스들을 반환 threshold 이상의 피크들만 반환
38 def find_spike_indices(dba, threshold) :
39     peaks, _ = find_peaks(dba, height=threshold, prominence=10)
40     return peaks
41

```

- 실제 laeq, lamax 및 대상 구간 내 threshold 초과 횟수를 측정하기 위한 함수들과 전처리(음원 데이터의 형태, 수치보정 등)을 정의한 모듈
- Function calculate_rms()
 - 수치화된 음성 데이터를 rms(root mean square) 배열 형태로 변환
 - Librosa 라이브러리에 정의된 stft(short time fourier transform) 메소드를 사용
 - Librosa를 사용하는 **Raw audio data -> rms value -> dB 변환**을 위한 일반적인 과정
- Function rms2db()
 - Rms value를 dB로 변환
 - Ref과 spl(sound pressure level)은 rms와 **dB-spl** 값 사이의 변환 공식에 따른 계산과정
- Function lamax_adjustment(), laeq_adjustment()
 - Librosa를 통해 계산된 **lamax, laeq** 값을 보정

/sound/level.py [2]

```

7  # raw data -> rms values 변환하는 함수
8  def calculate_rms(data, sr) :
9      s, phase = librosa.magphase(librosa.stft(data, center=False))
10     rms = librosa.feature.rms(S=s)
11     return rms
12
13  # rms values -> db values 변환하는 함수
14  def rms2db(rms) :
15      ref = 2 * 10 ** (-5)
16      spl = 10 * np.log10((rms ** 2) / (ref ** 2))
17      return spl
18
19  # LAmax 값 보정을 함수
20  def lamax_adjustment(db) :
21      return db + 10.5006
22
23  # LAeq 값 보정을 함수
24  def laeq_adjustment(db) :
25      return db + 17.3279
26
27  # 주어진 db values 에서 [frame_beg, frame_end) 범위의 LAmax 계산
28  def calculate_lamax(dba, frames_beg, frames_end) :
29      dba = lamax_adjustment(dba)
30      return np.max(dba[frames_beg:frames_end]).item()
31
32  # 주어진 db values 에서 [frame_beg, frame_end) 범위의 LAeq 계산
33  def calculate_laeq(dba, frames_beg, frames_end) :
34      dba = laeq_adjustment(dba)
35      return np.mean(dba[frames_beg:frames_end]).item()
36
37  # db values 를 받아서 피크 인덱스들을 반환 threshold 이상의 피크들만 반환
38  def find_spike_indices(dba, threshold) :
39      peaks, _ = find_peaks(dba, height=threshold, prominence=10)
40      return peaks
41

```

- Function calculate_lamax()
 - Frames_beg, frames_end(프레임의 시작점 및 종료점에 해당하는 시간 정보)와 dba 데이터를 바탕으로 대상 프레임의 **lamax**를 계산하는 함수
 - 앞서 정의한 lamax_adjustment 함수를 호출하여 인수로 받아온 dba 데이터를 보정하고, ndarray 형태로 저장되어 있는 dba배열의 최댓값에 해당하는 요소를 반환
- Function calculate_laeq()
 - 대상 프레임의 **laeq**를 계산하는 함수로 calculate_lamax 함수와 동일한 방식으로 구현
 - Nddarray 형태로 저장되어 있는 dba배열의 평균값을 반환

/sound/level.py [2]

```

7  # raw data -> rms values 변환하는 함수
8  def calculate_rms(data, sr) :
9      s, phase = librosa.magphase(librosa.stft(data, center=False))
10     rms = librosa.feature.rms(S=s)
11     return rms
12
13 # rms values -> dB values 변환하는 함수
14 def rms2db(rms) :
15     ref = 2 * 10 ** (-5)
16     spl = 10 * np.log10((rms ** 2) / (ref ** 2))
17     return spl
18
19 # LAmax 값 보정용 함수
20 def lamax_adjustment(db) :
21     return db + 10.5006
22
23 # LAeq 값 보정용 함수
24 def laeq_adjustment(db) :
25     return db + 17.3279
26
27 # 주어진 dB values 에서 [frame_beg, frame_end) 범위의 LAmax 계산
28 def calculate_lamax(dba, frames_beg, frames_end) :
29     dba = lamax_adjustment(dba)
30     return np.max(dba[frames_beg:frames_end]).item()
31
32 # 주어진 dB values 에서 [frame_beg, frame_end) 범위의 LAeq 계산
33 def calculate_laeq(dba, frames_beg, frames_end) :
34     dba = laeq_adjustment(dba)
35     return np.mean(dba[frames_beg:frames_end]).item()
36
37 # dB values 를 받아서 피크 인덱스들을 반환 threshold 이상의 피크들만 반환
38 def find_spike_indices(dba, threshold) :
39     peaks, _ = find_peaks(dba, height=threshold, prominence=10)
40     return peaks
41

```

- Function find_spike_indices()
 - 함수가 호출될 시 dba데이터 배열과 threshold(소음 기준치)를 인수로 받음
 - Scipy.signal 라이브러리의 find_peaks 함수를 호출하며 Peak의 height값을 threshold값으로 설정함으로써 파형의 극대값을 탐색
 - Prominence 값을 설정한 이유는 threshold 기준보다 높은 db의 구간에 서도 여러 차례의 ‘큰 소음(극댓값)’이 발생할 수 있으므로 **유의미한 소음의 발생 횟수를 정확히 측정하기** 위함

/sound/A_weighting.py


```

1  from numpy import pi, convolve
2  from scipy.signal.filter_design import bilinear
3  from scipy.signal import lfilter
4  from typing import *
5  from numpy.typing import *
6
7  def A_weighting(wav, sr) :
8      """Design of an A-weighting filter.
9
10     B, A = A_weighting(Fs) designs a digital A-weighting filter for
11     sampling frequency Fs. Usage: y = lfilter(B, A, x).
12     Warning: Fs should normally be higher than 20 kHz. For example,
13     Fs = 48000 yields a class 1-compliant filter.
14
15     Originally a MATLAB script. Also included ASPEC, CDSGN, CSPEC.
16
17     Author: Christophe Couvreur, Faculte Polytechnique de Mons (Belgium)
18     couvreur@thor.fpms.ac.be
19     Last modification: Aug. 20, 1997, 10:00am.
20
21     http://www.mathworks.com/matlabcentral/fileexchange/69
22     http://replaygain.hydrogenaudio.org/mfiles/adsgn.m
23     Translated from adsgn.m to PyLab 2009-07-14 endolith@gmail.com
24
25     References:
26     [1] IEC/CD 1672: Electroacoustics-Sound Level Meters, Nov. 1996.
27
28     """
29     # Definition of analog A-weighting filter according to IEC/CD 1672.
30     f1 = 20.598997
31     f2 = 107.65265
32     f3 = 737.86223
33     f4 = 12194.217
34     A1000 = 1.9997
35     NUMs = [(2 * pi * f4) ** 2 * (10 ** (A1000 / 20)), 0, 0, 0, 0]
36     DENs = convolve([1, +4 * pi * f4, (2 * pi * f4) ** 2],
37                     [1, +4 * pi * f1, (2 * pi * f1) ** 2], mode='full')
38     DENs = convolve(convolve(DENs, [1, 2 * pi * f3], mode='full'),
39                     [1, 2 * pi * f2], mode='full')
40     # Use the bilinear transformation to get the digital filter.
41     # (Octave, MATLAB, and PyLab disagree about Fs vs 1/Fs)
42     B, A = bilinear(NUMs, DENs, sr)
43     x = lfilter(B, A, wav)
44     return x, sr

```

- **A-weighting(A-가중 음향 레벨) 보정** 과정을 파이썬 함수로 표현한 부분
- 같은 db의 소리라도 특정 주파수 영역에서 더욱 민감하게 반응하는 귀의 특성을 반영하는 보정 과정으로 40phon의 등청감곡선을 바탕으로 주어진 db데이터를 보정함
- 층간소음은 인간이 느끼는 영역이므로 녹음 및 수집된 음원을 보정하는 과정이 필요

/sound/__init__.py

```
◎ floornoise-sound-tool-master > floornoise-sound-tool-master > sound >  __init__.py >
1  import librosa
2  from typing import *
3
4  def load_audio(filename: str, block_len=256, frame_len=2048):
5      sr = librosa.get_samplerate(filename)
6      stream = librosa.stream(
7          filename,
8          block_length=block_len,
9          frame_length=frame_len,
10         hop_length=frame_len,
11     )
12     return stream, sr
13
```

- app.py에서 오디오 파일을 선택하고 대상 음원의 raw data와 sample rate를 활용하기 위해 정의한 함수
- Sample rate와 별도로 stream에 block length, frame length, hop length를 설정하기 위해 sr과 stream을 각각 다른 방식으로 초기화

app.py [개요]

```

25
26 app = Flask(__name__)
27
28 # 설문조사 결과에 따른 sensitivity를 반영하는 함수 필요
29 # 구현된 바 없으므로 default값으로 normal 설정
30 sensitivity = "normal"
31
32 # 각 지표의 default값 선언
33 user_laeq = ""
34 user_lamax = ""
35 user_threshold = ""
36 user_noise_level = ""
37 user_sensitivity = sensitivity
38 user_l_index = np.nan
39
40 # 모바일 애플리케이션 내 파일 선택/녹음 기능으로 filename 설정
41 filename = 'trial.wav'
42
43 laeq_range_beg_str = ""
44 laeq_range_end_str = ""
45 laeq_str = ""
46 lamax_str = ""
47 threshold_str = ""
48 exceed_count_str = ""
49
50 audio_info = {
51     'generator': iter([]),
52     'wav': np.array([]),
53     'sr': 0,
54     'spl': np.array([]),
55     'len': 1.0,
56     'frame_start_t': 0.0,
57     'processed_t': 0.0,
58 }
59
60 analysis_info = {
61     'beg': 0.0,
62     'end': 0.0,
63     'threshold': 57.0,
64     'spike_indices': np.array([], dtype=np.int32),
65 }
66
67 def open_audio_file():
68     filetypes = (
69         ('raw audio file', '.wav'),
70     )
71     filename = fd.askopenfilename(filetypes=filetypes)
72     if filename != '':
73         stream, sr = load_audio(filename, block len=220)

```

- Flask를 활용하여 디자인한 API로 파일을 선택하고 사용자 sensitivity(민감도)를 설정한 후 정해진 url을 호출할 시 대상 음원의 laeq, lamax, threshold 초과 횟수, l-index 등을 반환하도록 설계
- Sensitivity 설정과 파일 선택 과정은 모바일 애플리케이션 내에서 동작에 의해 이루어지므로 이 코드에서 sensitivity는 default 값인 normal로, filename은 테스트 음원인 trial.wav로 설정
- Laeq, lamax, threshold 초과 횟수 측정은 floornoise-sound-tool에서 사용한 알고리즘과 유사 혹은 동일하며 l-index를 측정하는 알고리즘이 추가됨
- 본 자료에서는 floornoise-sound-tool과 중복되는 부분은 제외하고 API의 구조 및 두 프로그램의 차이점을 중점적으로 설명

app.py [1]

```
# 설문조사 결과에 따른 sensitivity를 반영하는 함수 필요
# 구현된 바 없으므로 default값으로 normal 설정
sensitivity = "normal"

# 각 지표의 default값 선언
user_laeq = ""
user_lamax = ""
user_threshold = ""
user_noise_level = ""
user_sensitivity = sensitivity
user_l_index = np.nan

# 모바일 애플리케이션 내 파일 선택/녹음 기능으로 filename 설정
filename = 'trial.wav'

laeq_range_beg_str = ""
laeq_range_end_str = ""
laeq_str = ""
lamax_str = ""
threshold_str = ""
exceed_count_str = ""
```

- Sensitivity, user_laeq, user_lamax, user_threshold, user_noise_level, user_sensitivity, user_l_index를 선언
- 파일이 정상적으로 선택되지 않거나 일부 지표만이 정상적으로 측정되지 않았을 때 request 자체가 동작하지 않거나 실행 시 발생하는 오류를 방지하기 위해 공백 또는 default 값으로 초기화하였음
- *_str 변수들의 경우 floornoise-sound-tool에서는 GUI 애플리케이션의 요소로 활용하기 위해 stringvar 형태로 선언 및 사용되었지만 API 내에서는 **결과 뷰잉 및 챗봇 응답 내 구성요소로 활용되어야 하기** 때문에 string 형태로 선언 및 사용함

app.py [2]

```
281 @app.route('/')
282 def setparameter():
283     global user_laeq, user_lamax, user_threshold, user
284     fileset()
285     user_laeq, user_lamax = update_laeq_lamax_label()
286     user_threshold = update_threshold()
287     user_noise_level = update_noise_level(user_sensiti
288     user_l_index = update_lindex(filename)
289     return "Parameters are set"
```

```
81 def fileset():
82     stream, sr = load_audio(filename, block_len=220)
83     audio_info['generator'] = stream
84     audio_info['sr'] = sr
85     audio_info['processed_t'] = 0.0
86     audio_info['frame_start_t'] = 0.0
87
88     next_audio_frame()
```

- App.route('/')
 - 로컬 서버 접근과 동시에 실행
 - 전역 변수인 user_*에 접근하여 저장된 값을 변경하도록 설계
 - Fileset()함수를 호출하여 대상 음원의 정보를 저장하고 음원 분석을 시작
- Fileset()
 - 대상 음원의 raw data와 sample rate 등 오디오 정보를 저장
 - Floornoise-sound-tool의 open_audio_file과 동일한 역할

app.py [3]

```
def update_noise_level(user_sensitivity, user_lamax):  
    noise_level = 6  
    t_low = (40, 43, 50, 58, 65, 76)  
    t_normal = (40, 42.5, 48, 54, 60, 73)  
    t_high = (39, 41, 47, 53.5, 58.5, 68)  
  
    if user_sensitivity == 'low':  
        for i in range(len(t_low)):  
            if user_lamax <= t_low[i]:  
                noise_level = i  
    elif user_sensitivity == 'high':  
        for i in range(len(t_high)):  
            if user_lamax <= t_high[i]:  
                noise_level = i  
    else: # user_sensitivity == 'normal'  
        for i in range(len(t_normal)):  
            if user_lamax <= t_normal[i]:  
                noise_level = i  
  
    return noise_level
```

- Function update_noise_level()
- User sensitivit에 따른 소음 기준을 참고하여 **대상 음원의 lamax를 기준으로 소음의 level을 판별**
- 초기 개발 당시 조건문을 활용하여 하드코딩한 부분이나 튜플을 사용하여 각 sensitivity에 해당하는 noise level 측정 기준을 저장하고 조건반복문을 통해 판별하도록 구성
- 주/야간 시간대에 따른 보정값 반영 예정

app.py [4-1]

```
217 def update_lindex(filename):
218     wav, sr = librosa.load(filename)
219     wav, sr = A_weighting(wav, sr)
220     ref = 2 * 10 ** (-5)
221
222     f63 = signal.firwin(101, cutoff=[62, 64], fs=sr, pass_zero='bandpass')
223     f125 = signal.firwin(101, cutoff=[122, 124], fs=sr, pass_zero='bandpass')
224     f250 = signal.firwin(101, cutoff=[249, 251], fs=sr, pass_zero='bandpass')
225     f500 = signal.firwin(101, cutoff=[499, 501], fs=sr, pass_zero='bandpass')
226
227     band63 = signal.lfilter(f63, [1,0], wav)
228     band125 = signal.lfilter(f125, [1,0], wav)
229     band250 = signal.lfilter(f250, [1,0], wav)
230     band500 = signal.lfilter(f500, [1,0], wav)
231
232     s63, phase63 = librosa.magphase(librosa.stft(band63, center=False))
233     s125, phase125 = librosa.magphase(librosa.stft(band125, center=False))
234     s250, phase250 = librosa.magphase(librosa.stft(band250, center=False))
235     s500, phase500 = librosa.magphase(librosa.stft(band500, center=False))
236
237     rms63 = librosa.feature.rms(S=s63)
238     rms125 = librosa.feature.rms(S=s125)
239     rms250 = librosa.feature.rms(S=s250)
240     rms500 = librosa.feature.rms(S=s500)
241
242     spl63 = 10 * np.log10((rms63 ** 2) / (ref ** 2)) + 10.5006
243     spl125 = 10 * np.log10((rms125 ** 2) / (ref ** 2)) + 10.5006
244     spl250 = 10 * np.log10((rms250 ** 2) / (ref ** 2)) + 10.5006
245     spl500 = 10 * np.log10((rms500 ** 2) / (ref ** 2)) + 10.5006
246
247     band_lamax = [
248         np.max(spl63).item(),
249         np.max(spl125).item(),
250         np.max(spl250).item(),
251         np.max(spl500).item()
252     ]
```

- Function update_lindex()
- L-index 판별 기준의 대상 주파수 대역인 63/125/250/500hz에 해당하는 음원을 추출하여 db를 측정하고 가장 높은 레벨을 나타내는 주파수 대역이 포함되는 구간을 음원의 I-index로 산출하는 함수
- 대상 음원의 raw data와 sample rate를 load하여 a-weighting 과정을 거쳤음
- Fir필터를 통해 63/125/250/500hz에 해당하는 데이터를 추출
- Lfilter를 통해 1차원 데이터로 필터링 (필터 적용 과정의 일부)

app.py [4-2]

```

217 def update_lindex(filename):
218     wav, sr = librosa.load(filename)
219     wav, sr = A_weighting(wav, sr)
220     ref = 2 * 10 ** (-5)
221
222     f63 = signal.firwin(101, cutoff=[62, 64], fs=sr, pass_zero='bandpass')
223     f125 = signal.firwin(101, cutoff=[122, 124], fs=sr, pass_zero='bandpass')
224     f250 = signal.firwin(101, cutoff=[249, 251], fs=sr, pass_zero='bandpass')
225     f500 = signal.firwin(101, cutoff=[499, 501], fs=sr, pass_zero='bandpass')
226
227     band63 = signal.lfilter(f63, [1,0], wav)
228     band125 = signal.lfilter(f125, [1,0], wav)
229     band250 = signal.lfilter(f250, [1,0], wav)
230     band500 = signal.lfilter(f500, [1,0], wav)
231
232     s63, phase63 = librosa.magphase(librosa.stft(band63, center=False))
233     s125, phase125 = librosa.magphase(librosa.stft(band125, center=False))
234     s250, phase250 = librosa.magphase(librosa.stft(band250, center=False))
235     s500, phase500 = librosa.magphase(librosa.stft(band500, center=False))
236
237     rms63 = librosa.feature.rms(S=s63)
238     rms125 = librosa.feature.rms(S=s125)
239     rms250 = librosa.feature.rms(S=s250)
240     rms500 = librosa.feature.rms(S=s500)
241
242     spl63 = 10 * np.log10((rms63 ** 2) / (ref ** 2)) + 10.5006
243     spl125 = 10 * np.log10((rms125 ** 2) / (ref ** 2)) + 10.5006
244     spl250 = 10 * np.log10((rms250 ** 2) / (ref ** 2)) + 10.5006
245     spl500 = 10 * np.log10((rms500 ** 2) / (ref ** 2)) + 10.5006
246
247     band_lamax = [
248         np.max(spl63).item(),
249         np.max(spl125).item(),
250         np.max(spl250).item(),
251         np.max(spl500).item()
252     ]

```

- 필터링을 거친 각 주파수의 데이터를 푸리에 변환하고(librosa.stft) phase 정보를 제외한 세기 정보만 추출하기 위해 librosa.magphase 메소드를 활용 (s*: magnitude, phase*: phase)
- 각 주파수 대역별 세기 데이터의 rms value를 구함 (floornoise-sound-tool의 calculate_rms와 유사한 과정)
- Rms를 spl로 변환하기 위해 magnitude-db 변환 공식을 사용 (floornoise-sound-tool의 rms2db와 유사한 과정)
- 각 주파수 대역의 **lamax** 값을 저장 (floornoise-sound-tool의 calculate_lamax와 유사한 과정)

app.py [4-3]

```
l_index = 85
t63 = tuple(range(110, 54, -5))
t125 = tuple(range(100, 44, -5))
t250 = tuple(range(93, 37, -5))
t500 = tuple(range(87, 31, -5))
t_index = tuple(range(85, 29, -5))

for i in range(len(t_index)):
    if band_lamax[0] > t63[i] or band_lamax[1] > t125[i] or band_lamax[2] > t250[i] or band_lamax[3] > t500[i]:
        l_index = t_index[i]
        break

return l_index
```

- 각 주파수의 **lamax** 데이터가 속한 구간 중 가장 높은 레벨을 음원의 **I-index**로 산출하여 반환
- 초기 개발 당시 단순 조건문으로 구현된 부분이나, 추후 데이터 및 보정값 변화에 따른 수정이 용이하도록 tuple 형태로 각 I-index의 유 기준값들을 저장하고 조건반복문을 통해 판별하도록 구성
- ‘바닥충격음 논문 스터디’ 자료 내의 I-index 기준표에서 각 주파수별 측정결과가 등급곡선을 2db까지 상회하는 것을 허용한 수치

app.py [5]

```
def analysis_sentence(user_noise_level, user_l_index):
    t_noise_level_sentences = [
        '실내 환경은 조용합니다. 외부 소음에 노출되지 않았습니다.',
        '소음이 당신을 괴롭히지 않습니다. 소음이 일상생활에 지장을 줄 만큼 크지 않습니다.',
        '인내심을 가지고 소음 문제에 대처할 수 있는 수준입니다. 소음이 지속되어 괴롭다면, 이',
        '소음 문제에 대처하기 위해 이웃과 소통할 필요가 있습니다. 주의를 부탁드립니다. 보세요. 이',
        '소음이 당신의 휴식 시간을 방해합니다. 층간소음 법적 기준을 초과합니다. 이웃과 적극적',
        '소음으로 인해 거주 공간에서 대화가 들리지 않습니다. 소음으로부터 건디가 어려운 상황',
        '사람이 이곳에 사는 것은 불가능합니다.'
    ]
    t_l_index_sentences = [
        ['위는 행위나 발걸음 소리가 거의 들리지 않습니다.\n', '의자나 물건이 떨어지는 소리가'],
        ['위는 행위나 발걸음 소리가 조용할 때 들립니다.\n', '의자나 물건이 떨어지는 소리가'],
        ['위는 행위나 발걸음 소리가 멀리서 들리는 느낌입니다.\n', '의자나 물건이 떨어지는 소'],
        ['위는 행위나 발걸음 소리가 들리지만 거슬리지 않습니다.\n', '생들을 신고 걷는 소리가'],
        ['위는 행위나 발걸음 소리가 들리지만 거의 거슬리지 않습니다.\n', '칼을 사용하는 소리'],
        ['위는 행위나 발걸음 소리에 약간의 마음이 쓰입니다.\n', '슬리퍼를 신고 걷는 소리가'],
        ['위는 행위나 발걸음 소리가 약간 거슬립니다.\n', '수저를 떨어뜨리는 소리가 들립니다.'],
        ['위는 행위나 발걸음 소리가 잘 들려 거슬립니다.\n', '동전을 떨어뜨리는 소리가 들립니'],
        ['위는 행위나 발걸음 소리가 매우 잘 들려 거슬립니다.\n', '1원짜리 동전을 떨어뜨리는'],
        ['위는 행위나 발걸음 소리가 매우 귀찮습니다.\n', '주의하여도 시비가 붙는 수준입니다.'],
        ['위는 행위나 발걸음 소리가 시끄러워 견딜 수 없습니다.\n', '소음을 참는 생활이 필요']
    ]

    l_index = list(range(30, 86, 5))

    sentence = ''
    sentence += '녹음된 전체 음원을 통해 주거 환경의 소음 발생 정도를 분석합니다. \n'
    for i in range(len(l_index)):
        if l_index[i] == user_l_index:
            for j in t_l_index_sentences[i]:
                sentence += j
    sentence += '발생한 최고 소음을 기반으로 층간소음의 정도를 분석합니다. \n'
    sentence += t_noise_level_sentences[user_noise_level]

    return sentence
```

- Function analysis_sentsensis()
- 각 Lamax 구간에 따른 성가심도를 성문화하고 분석된 음원을 통해 사용자 환경의 noise level과 l-index를 바탕으로 적절한 문구를 반환하는 부분으로 API의 주요 함수
- 각 Noise level과 l-index에 해당하는 문구들을 1/2차원 배열에 저장하고 사용자 환경의 수치에 해당하는 문구들을 간단한 설명과 함께 반환할 수 있도록 구성
- Noise level과 성가심도, l-index와 중량충격음 기준, 사용자 민감도 등 API 내에서 처리하는 변수나 보정과정을 다양화하며 시나리오에 반영하고 문구를 추가할 수 있도록 개발 및 고도화 중

app.py [6]

```
291 @app.route('/showresult')
292 def showparameter():
293     return jsonify({
294         "laeq": str(user_laeq),
295         "lamax": str(user_lamax),
296         "sensitivity": str(user_sensitivity),
297         "threshold": str(user_threshold),
298         "noise_level": str(user_noise_level),
299         "l-Index" : str(user_l_index)
300     })
301
302 if __name__ == "__main__":
303     app.run()
```

- Address/showresult url로 호출 시 대상 음원의 laeq, lamax, user sensitivity, threshold 초과 횟수, noise level, l-index를 json 형태로 반환하도록 설계
- API 고도화 과정에서는 showparameter() 함수 자체가 아니라 서버에 전역변수 형태로 저장되어 있는 수치 각각을 활용하여 반환하는 문구에 반영할 수 있도록 함수를 추가하는 중