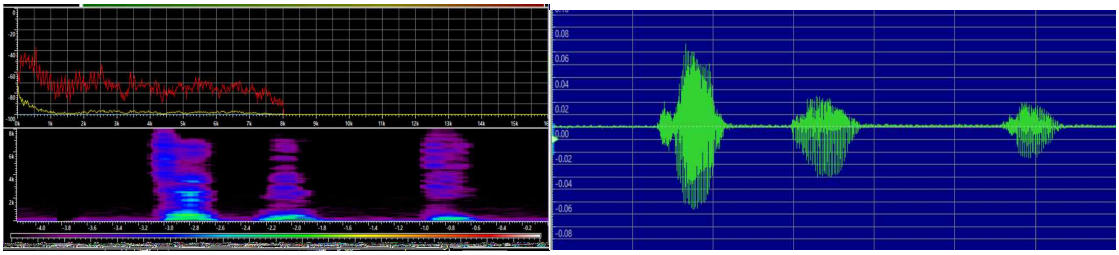
전기전자 공학부 202114160 채민기

original ‘채민기’ 이름을 녹음 하였습니다.



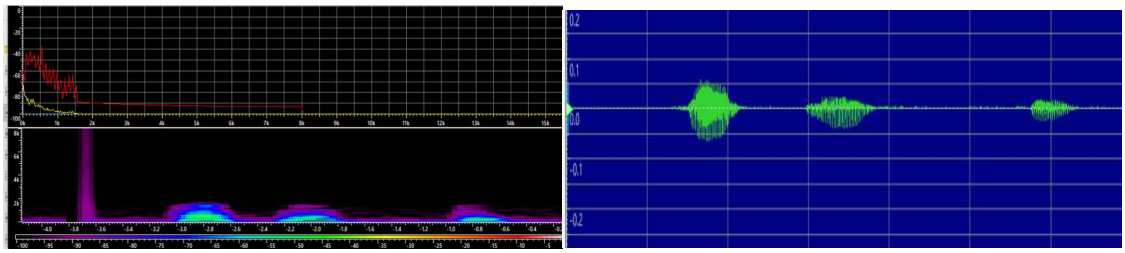
LPF

실험전 예상

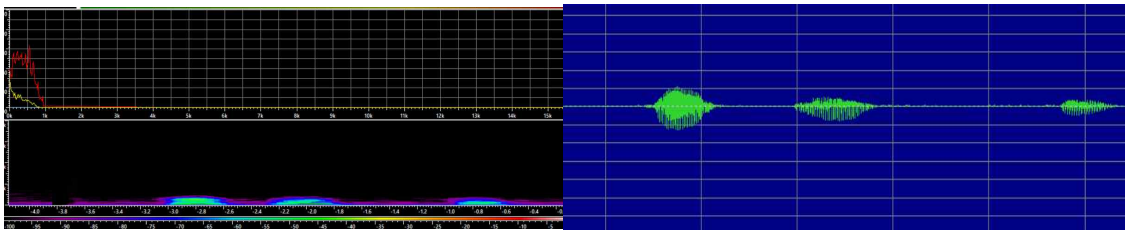
지난 주차에서 moving average filter를 적용해 보았듯이 lowpass 필터를 적용하면 남자목소리의 대부분이 저주파 대역에 속하여 cut off 주파수 설정에 따라 소리의 크기는 비교적 유지되지만 고주파 성분이 빠지게 되어 소리가 먹먹하게 들릴 것이라고 예상할 수 있다.

실험결과 및 분석

0.2 lowpass



0.1 low pass



LPF 에서는 예상과 같게 소리의 크기는 크게 줄어들지 않았으나 어딘가 막힌 듯 먹먹한 소리를 들을 수 있었다. 또한 가장 크게 변한소리는 무성음인 ‘ㅊ’이 들어간 ‘채’ 였다.
음성 신호의 파형 역시 첫 신호인 ‘채’ 만이 눈에띄는 변화를 보여주었다. fcut이 0.2에서 ‘ㅊ’과 ‘ㅈ’사이의 소리가 들였으나 차단주파수를 더 낮게 설정하자 ‘채’가 아닌 ‘재’로 들렸다.

또한 0.2에서 0.1로 즉1.6k에서 800hz로 cut off 주파수를 변경하자 바람소리와 같은 잡음이 좀더 동굴같이 울리는 잡음으로 섞여 들렸다. 이는 진동과 같이 울리는 특성의 잡음이 저주파 대역의 신호기 때문에 차단 주파수를 낮출수록 잡음도 더 울리는 소리만이 남게 된다고 판단 할 수 있다.

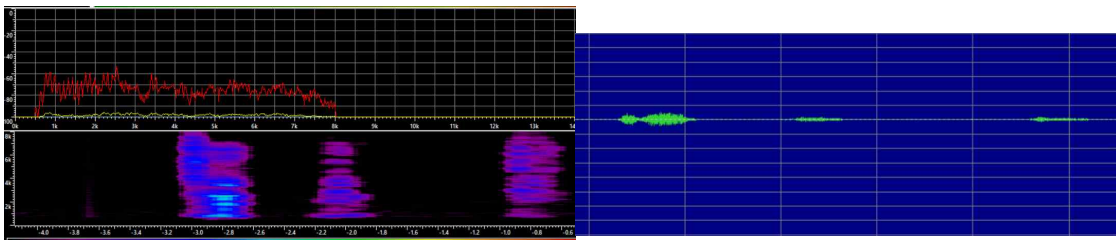
HPF

실험전 예상

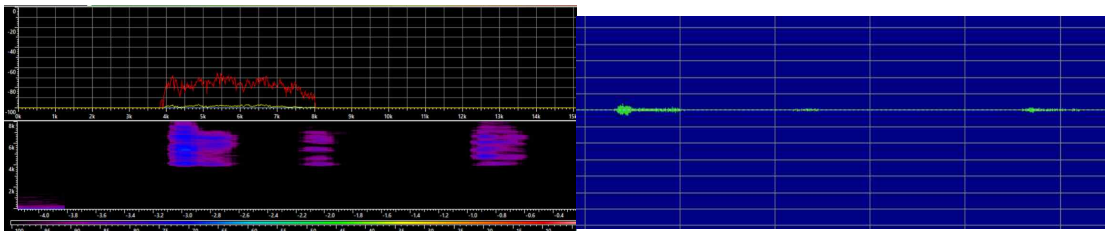
앞선 lowpass filter를 통해 저주파 대역의 신호가 본인의 음성의 대부분을 차지 하고 있음을 확인 할 수 있었다. 따라서 음성의 극히 일부를 포함할것이라 예상되는 만큼 소리의 크기가 작을 것이라고 생각한다.

실험결과 및 분석

0.1highpass



0.5highpass



HPF에서 소리는 잘들리지만 힘이 부족한 느낌을 받았다. 차단 주파수를 키웠을 때 소리의 크기는 더욱 약해 지는 것을 확인 할 수 있었다. 이는 우측에 위치한 음성신호 사진을 통해 확인 할 수 있듯 신호의 크기가 약해진 것을 확인할 수 있었다.

LPF와 비교하였을 때 가장 큰 차이는 두가지 였다. 하나는 기존의 가장 크게 변화한 'ㅊ'소리였다. 세 글자에서 비교적 소리가 크게 들렸는데 이는 '채민기' 세 글자에서 'ㅊ'이 고주파 성분이 가장 많았기 때문임을 알 수 있다. 실제 음성신호의 진폭역시 '채'에서 가장 큰 것을 확인 할 수 있다.

두 번째는 잡음의 차이였다. 잡음은 LPF와 달리 울리는 듯한 잡음은 없었지만 날카로운 잡음이 존재하였다. 이를 통해 날카로운 소리가 고주파 대역에 위치한 다는 사실을 한번 더 확인 할 수 있었다.

BPF & BSF

code<전체 코드 별도 첨부>

```
def filter_ft(mag,fcut,fcut2,ftype):
    #ft_bin =
    Nf,ft_bin = mag.shape
    fmag = np.zeros([Nf,ft_bin])
    # 필터의 상승 하강 엣지 정의
    fcut_pos =int(ft_bin*fcut)
    fcut_neg =int(ft_bin*fcut2)
    # filter type
    if ftype =='lowpass':
        fmag[:,0:fcut_pos]=mag[:,0:fcut_pos]
    elif ftype =='highpass':
        fmag[:,fcut_pos:ft_bin]=mag[:,fcut_pos:ft_bin]
    elif ftype =='bandpass':
        fmag[:, fcut_pos:fcut_neg]= mag[:, fcut_pos:fcut_neg]
    elif ftype =='bandstop':
        fmag[:,0:fcut_pos]= mag[:,0:fcut_pos]
        fmag[:,fcut_neg:ft_bin]= mag[:,fcut_neg:ft_bin]
    else:
        print("error can't find filter")
        return-1
    return fmag
```

2개의 차단 주파수가 필요한 bandpass 와 bandstop 필터의 특성으로 fcut2를 추가하여 두 개의 차단 주파수를 설정할 수 있도록 하였다.

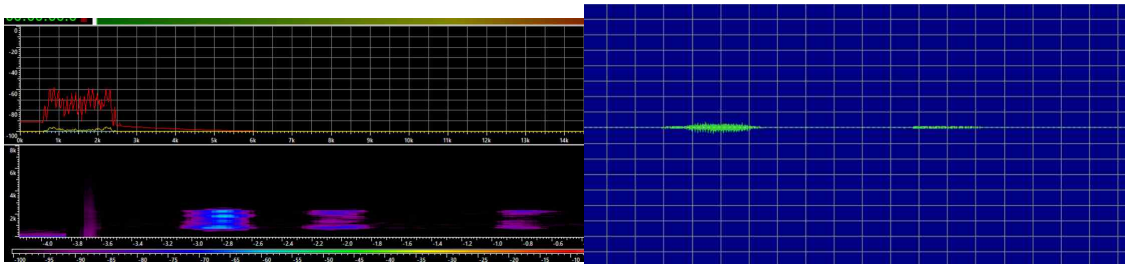
BPF

실험전 예상

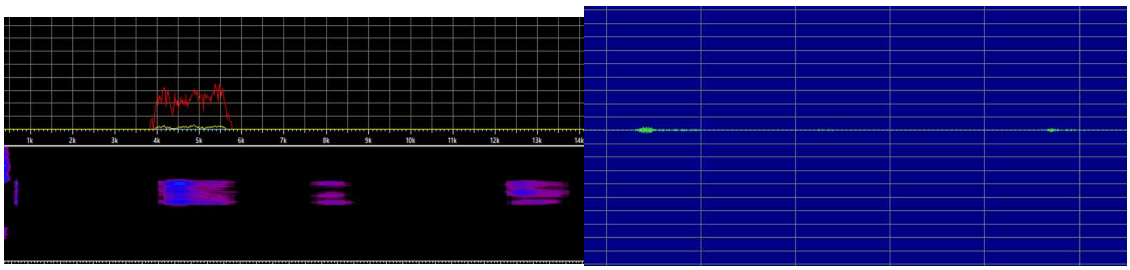
lpf와 hpf실험을 통해 각 필터를 적용하였을 때 음성신호의 변화를 보았다.
이를 통해 울리는 소리가 줄고 음성신호의 크기가 저주파의 차단으로 인해 발생하고 고주파의 차단으로 인해 소리가 답답함을 줄것이라 예상 할 수 있다.

실험결과 및 분석

bandpass 0.1~0.3



bandpass 0.5~0.7



차단주파수를 0.1~0.3 즉 800hz~2.4khz와 0.5~0.7인 4khz~5.6Khz로 설정하여 보았다.
음성 자체가 잡음이 많이 들어간 무전기의 소리와 유사하게 들렸다. 또한0.1~0.3구간에서 '채', '민'에 비해 '기' 소리가 작게들렸고. 0.5~0.7구간에서는 '민'의 소리만 작게 들렸다.
이는 음성신호의 진폭을 통해 다시 확인 할 수 있다. 상대적으로 울리는 소리인 '민'이 0.5~0.7구간에서 저주파 성분이 줄어 다른 두 글자에 비해 작게 들린 것이라고 생각 할 수 있다.

두신호 모두 lpf와 hpf의 특성을 통해 알아보았듯 저주파가 적어 소리의 크기와 울리는 소리가 줄었으며 고주파 성분의 부족함으로 인해 답답한 느낌도 같이 받을 수 있었다.

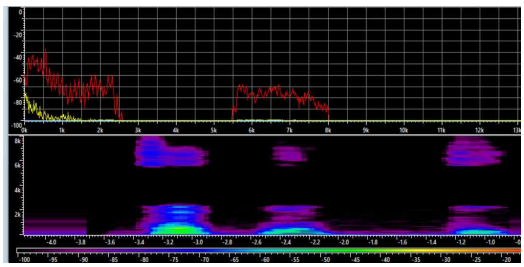
BSF

실험전 예상

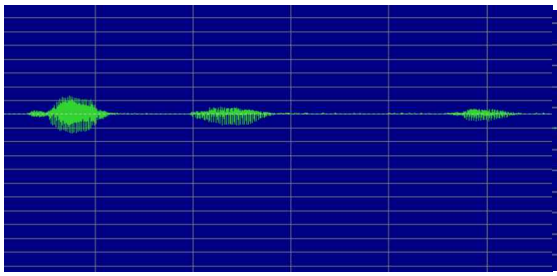
저주파를 대부분 살려 소리의 크기는 비교적 줄지 않고 일정부분의 고주파로 인해
저주파 필터보다 답답한 느낌이 줄어 들것이라고 예상해 보았다.
저주파 대역에서 들린 소리에 일부 고주파성분을 살리면 줄어들었던 '채'가 잘 들릴지 중점적으로 실험해 보고자 하였다.

실험결과 및 분석

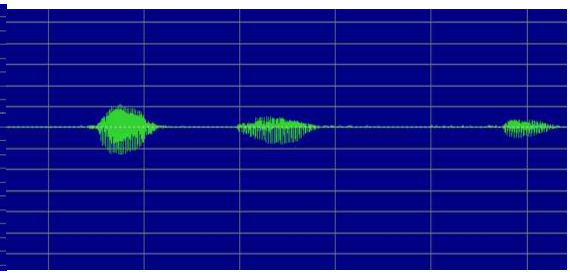
bandstop 0.3~0.7



bandstop 0.3~0.7



lowpass 0.1



해당 실험에서 다른 필터들에 비해 소리가 크게 줄어들지 않았으며, 고주파 성분이 일부 남아있어 lpf를 적용한 소리에서 먹먹함이 줄어든 소리를 들을 수 있었다.
하지만 해당 실험에서도 'ㅈ'과 'ㅊ'을 명확하게 구분하기엔 어려움이 있었다.
일부 고주파 대역의 소리가 남아있음에도 lpf와 유사하게 'ㅈ', 'ㅊ'이 구분이 되지 않았던 것이다. 이를통해 ㅊ이 되는 고주파 성분은 bsf를 통해 차단된 주파수 대역인 2.4khz~5.6khz사이에 분포할것이라고 추측 할 수 있다.

code

```
import numpy as np
from scipy import signal
from scipy.signal import get_window
from scipy.io import wavfile
import sys
WL =256
WR =128
# x= real array wlen= window length, step = window shift step, w= window
def stft(x, w, step):
    wlen =len(w)
    nsample =len(x)
    # 1e-12~~ = 0값을 피하기위한 극소값
    Xtf = np.array([np.fft.rfft(w*x[i:i+wlen])
                    for i inrange(0,nsample - wlen+1, step)])+ 1e-12*(1+1j)
    return Xtf
# inverse short time F-T
def istft(Xtf, w,step, nsample):
    nframe =len(Xtf)
    wlen =len(w)
    y = np.zeros(nsample)
    ws = np.zeros(nsample)
    ind =0
    for i inrange(0, nsample-wlen +1, step):
        y[i:i+wlen] += w*np.fft.irfft(Xtf[ind])
        ws[i:i+wlen] += w*w
        ind +=1
    ws[ws==0] =1
    y = y/ws
    return y
def filter_ft(mag,fcut,fcut2,ftype):
    #ft_bin =
    Nf,ft_bin = mag.shape
    fmag = np.zeros([Nf,ft_bin])
    # 필터의 상승 하강 엣지 정의
    fcut_pos =int(ft_bin*fcut)
    fcut_neg =int(ft_bin*fcut2)
    # filter type
    if ftype =='lowpass':
        fmag[:,0:fcut_pos]=mag[:,0:fcut_pos]
    elif ftype =='highpass':
        fmag[:,fcut_pos:ft_bin]=mag[:,fcut_pos:ft_bin]
```

```
    elif ftype =='bandpass':
        fmag[:, fcut_pos:fcut_neg]= mag[:, fcut_pos:fcut_neg]
    elif ftype =='bandstop':
        fmag[:,0:fcut_pos]= mag[:,0:fcut_pos]
        fmag[:,fcut_neg:ft_bin]= mag[:,fcut_neg:ft_bin]
    else:
        print("error can't find filter")
        return-1
    return fmag
# 명령어 인자 순서  1. input 2. output  3.filter 4.fcut 5.fcut2
fs, data = wavfile.read(sys.argv[1])
fcut = float(sys.argv[4])
w = get_window('hann', WL)
Xf =stft(data,w, WR)
Mag = np.abs(Xf)
Phs = np.angle(Xf)
# 인자가 여러개인 경우 = cutoff 주파수 값이 2개 이상인경우
iflen(sys.argv)>5:
    fcut2 = float(sys.argv[5])
else:
    fcut2 =1
fMag = filter_ft(Mag, fcut,fcut2,sys.argv[3])
Xfr = fMag*np.exp(1j * Phs)
y = istft(Xfr,w,WR,len(data))
wavfile.write(sys.argv[2],fs,y.astype(np.int16))
```