프로젝트 결과 보고서

프로젝트 주제명	심전도 분석 그래프		
교과목명	파이썬과학프로그래밍	학번	20165115
개발 기간	2019.12.10~2019.12.16	이름	김승현
제출일	2019 12월 16일		

Ⅰ. 개발배경

기말고사 프로젝트 주제로 심전도관련 프로젝트가 주어졌다. 30분(1800초)간 측정된 심전도 데이터 파일에서 데이터를 읽어오고, 이를 그래프로 표시하고, 각 채널 신호의 최대, 최 소값과 그 시점을 알아내서 결과를 저장하는 프로그램를 작성하는 프로젝트이다.

* 데이터 파일의 채널당 샘플데이터의 갯수는 6269개 입니다. 기존 주려고 했던 파일은 30분데이터(channel 당 462600 samples)였으나 과제와 맞지 않아서 ecg_1.txt 파일로 하였고 이 경우 데이터의 갯수는 각 채널당 6259개이다
* 30분데이터(channel 당 462600 samples) => 초당 256개의 데이터를 읽으므로, 데이터 파일은 약 24초의 data로 구성되어 있다(6269/256)

Ⅱ. 개발 내용

이름,번호,sampling rate, gain을 입력하고 데이터 파일을 가져온후, 출력할 그래프의 channel을 선택,시작지점,끝지점을 입력합니다.

결과해석 버튼을 누르면 다음 그림과 같이 선택한 채널의 그래프가 순서대로 출력하고,
그래프 옆의 리스트 박스에는 각 채널마다 최소,최대값과 그에대한 시점이 출력됩니다.

결과 저장버튼을 누르면 "이름_번호_sampling rate_record.txt"파일 명으로 파일이 저장됩니다.

저장한 파일을 정보읽어오기 버튼으로 선택하여 로드한후 해당한 파일의 내용을 listbox에 출력할수있습니다.

코드 분석

```
# -*- coding: utf-8 -*-

import tkinter                  #tkinter 라이브러리 임포트
import numpy as np              #numpy 라이브러리 임포트 np이름으로 사용
from tkinter import filedialog  #tkinter의 filedialog 라이브러리 임포트, 선택한 파일 로드하기위함
from matplotlib.figure import Figure  #matplotlib.figure의 Figure 라이브러리 임포트, 그래프 출력하기 위함
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg, NavigationToolbar2Tk)
                                #matplotlib.backends.backend_tkagg의 FigureCanvasTkAgg 라이브러리 임포트, canvas에 그래프를 출력하기 위함

window = tkinter.Tk()           #tkinter라이브러리로 윈도우 생성



signal1 = 0                 #데이터 파일을 저장하기 위한 변수
start = 0                   #추출할 데이터범위의 시작 시점
end = 0                     #추출할 데이터범위의 끝 시점
start_2 = 0                 #추출할 데이터범위의 시작 데이터 인덱스
end_2 = 0                   #추출할 데이터범위의 끝 데이터 인덱스

def process():              #데이터 파일을 로드하는 함수
    file = filedialog.askopenfile(parent=window,mode='r',title='Choose a file')  #파일 선택
    if file != None:                    #파일이 존재한다면
        data = file.read()              #파일의 데이터 읽기
        file.close()                    #파일 닫기
        print("I got %d bytes from this file." % len(data))        #파일의 길이 출력

    global signal1                      #데이터파일을 저장할 변수를 global 변수로 선언
    signal1 = np.loadtxt(file.name,delimiter = ',')  #데이터 파일을 읽어 signal변수에 저장

def graph():                #그래프를 그리는 함수
    fig = Figure(figsize=(5,7),dpi=100)       #그래프를 그릴 범위를 500 pixel x 700 pixel로 잡아 fig객체에 저장

    arr = []                                  #생성한 그래프 객체를 저장할 배열, x축을 마지막 그래프만 출력하게 하기 위함

    global start            #start변수를 global변수로 선언
    global end              #end변수를  global 변수로 선언
```

```
    start = int(entry5.get())  #시점으로 입력한 값을 가져와 start에 저장
    end = int(entry6.get())    #끝으로 입력한 값을 가져와 end에 저장

    global start_2              #start_2를 global변수로 선언
    global end_2                #end_2를 global변수로 선언
    start_2 = start * 256       #초당 256개의 데이터를 읽으므로 start(초)*256를 계산해 시작 시점의 데이터 인덱스를 start_2에 저장
    end_2 = start*256 + (end-start+1)*256 #시작시점데이터인덱스 + (end(초)-start(초)+1)*256를 계산해 끝 시점의 데이터 인덱스를 end_2에 저장

    if start == 0 and end== 0: #start와 end의 시점이 0이면 end_2에 0을 저장,start*256 + (end-start+1)*256가 0이 안되기 때문
        end_2 = 0

    listbox.delete(0,tkinter.END) #listbox에 있는 모든 항목들을 삭제

    auto = 0            #listbox에 넣을 항목의 인덱스를 0을 초기화

    if CheckVariety_1.get() == 1:     #channel1의 체크박스에 체크되어있을 경우
        x = signal1[start_2:end_2,0] #start_2애서 end_2범위의 channel1데이터를 추출하여 x에 저장
        t = np.arange(len(x))*1.0/int(entry3.get()) #[0,1,2,3...] * 1/sampling rate(hz,주파수)의 공식을 적용해
                                                    #해당 채널의 데이터를 sampling한다, sampling된값은 t에 저장
                                                    #주파수는 입력한 주파수를 이용
        ax1 = fig.add_subplot(7,1,1)                #fig객처를 사용해 7행 1열의 첫번째 위치에 그래프를 위치설정, 객체 반환
        ax1.plot(t,x)                               #반환된 객체를 이용해 x축은 t값으로, y축은 x값으로 한다.
        arr.append(ax1)                             #그래프 객체를 arr배열에 추가한다.
        min_val, max_val,min_time,max_time = max_min(start_2,end_2,0)  #max_min함수에 start_2와 end_2,채널을 전달해 최대,최솟값과
                                                                       #그 시점을 반환받아 min_val, max_val,min_time,max_time에 각각
                                                                       #저장한다.

        auto = auto + 1                             #listbox의 항목 인덱스를 1증가 시킨다
        listbox.insert(auto,"channel"+str(1)+" 최소: "+ str(min_val)+"  시점: 약"+str(min_time)+"초")
                                                    #증가시킨 항목인덱스에 채널명,전달받은 최
```

```
소값, 최소값의 시점을 항목값으로 하여 listbox에 삽입
        auto = auto + 1                                  #listbox의 항목 인덱스를 1증가 시킨다
        listbox.insert(auto,"channel"+str(1)+" 최대: "+ str(max_val)+"   시점: 약"+str(max_time)+"초")
                                                         #증가시킨 항목인덱스에 채널명,전달받은 최
소값, 최소값의 시점을 항목값으로 하여 listbox에 삽입

        auto = auto + 1                                  #listbox의 항목 인덱스를 1증가 시킨다
        listbox.insert(auto,"₩n")                       #증가시킨 항목 인덱스에 "₩n"을 항목값으로 하
여 listbox에 추가한다.
                                                         #가독성을 위해
    #다음의 6개 if문은 위의 if문의 설명과 동일함
    if CheckVariety_2.get() == 1:
        x = signal1[start_2:end_2,1]
        t = np.arange(len(x))*1.0/int(entry3.get())
        ax2 = fig.add_subplot(7,1,2)
        ax2.plot(t,x)
        arr.append(ax2)

        min_val, max_val,min_time,max_time = max_min(start_2,end_2,1)

        auto = auto + 1
        listbox.insert(auto,"channel"+str(2)+" 최소: "+ str(min_val)+"   시점: 약"+str(min_time)+"초")
        auto = auto + 1
        listbox.insert(auto,"channel"+str(2)+" 최대: "+ str(max_val)+"   시점: 약"+str(max_time)+"초")

        auto = auto + 1
        listbox.insert(auto,"₩n")

    if CheckVariety_3.get() == 1:
        x = signal1[start_2:end_2,2]
        t = np.arange(len(x))*1.0/int(entry3.get())
        ax3 = fig.add_subplot(7,1,3)
        ax3.plot(t,x)
        arr.append(ax3)

        min_val, max_val,min_time,max_time = max_min(start_2,end_2,2)

        auto = auto + 1
        listbox.insert(auto,"channel"+str(3)+" 최소: "+ str(min_val)+"   시점: 약"+str(min_time)+"초")
```

```
        auto = auto + 1
        listbox.insert(auto,"channel"+str(3)+" 최대: "+ str(max_val)+"   시점: 약"+str(max_time)+"초")

        auto = auto + 1
        listbox.insert(auto,"₩n")

    if CheckVariety_4.get() == 1:
        x = signal1[start_2:end_2,3]
        t = np.arange(len(x))*1.0/int(entry3.get())
        ax4 = fig.add_subplot(7,1,4)
        ax4.plot(t,x)
        arr.append(ax4)

        min_val, max_val,min_time,max_time = max_min(start_2,end_2,3)

        auto = auto + 1
        listbox.insert(auto,"channel"+str(4)+" 최소: "+ str(min_val)+"   시점: 약"+str(min_time)+"초")
        auto = auto + 1
        listbox.insert(auto,"channel"+str(4)+" 최대: "+ str(max_val)+"   시점: 약"+str(max_time)+"초")

        auto = auto + 1
        listbox.insert(auto,"₩n")

    if CheckVariety_5.get() == 1:
        x = signal1[start_2:end_2,4]
        t = np.arange(len(x))*1.0/int(entry3.get())
        ax5 = fig.add_subplot(7,1,5)
        ax5.plot(t,x)
        arr.append(ax5)

        min_val, max_val,min_time,max_time = max_min(start_2,end_2,4)

        auto = auto + 1
        listbox.insert(auto,"channel"+str(5)+" 최소: "+ str(min_val)+"   시점: 약"+str(min_time)+"초")
        auto = auto + 1
        listbox.insert(auto,"channel"+str(5)+" 최대: "+ str(max_val)+"   시점: 약"+str(max_time)+"초")

        auto = auto + 1
        listbox.insert(auto,"₩n")
```

```
if CheckVariety_6.get() == 1:
    x = signal1[start_2:end_2,5]
    t = np.arange(len(x))*1.0/int(entry3.get())
    ax6 = fig.add_subplot(7,1,6)
    ax6.plot(t,x)
    arr.append(ax6)
    max_min(start_2,end_2,5)

    min_val, max_val,min_time,max_time = max_min(start_2,end_2,5)

    auto = auto + 1
    listbox.insert(auto,"channel"+str(6)+" 최소: "+ str(min_val)+"   시점: 약"+str(min_time)+"초")
    auto = auto + 1
    listbox.insert(auto,"channel"+str(6)+" 최대: "+ str(max_val)+"   시점: 약"+str(max_time)+"초")

    auto = auto + 1
    listbox.insert(auto,"₩n")

if CheckVariety_7.get() == 1:
    x = signal1[start_2:end_2,6]
    t = np.arange(len(x))*1.0/int(entry3.get())
    ax7 = fig.add_subplot(7,1,7)
    ax7.plot(t,x)
    arr.append(ax7)
    max_min(start_2,end_2,6)

    min_val, max_val,min_time,max_time = max_min(start_2,end_2,6)

    auto = auto + 1
    listbox.insert(auto,"channel"+str(7)+" 최소: "+ str(min_val)+"   시점: 약"+str(min_time)+"초")
    auto = auto + 1
    listbox.insert(auto,"channel"+str(7)+" 최대: "+ str(max_val)+"   시점: 약"+str(max_time)+"초")

    auto = auto + 1
    listbox.insert(auto,"₩n")

for i in range(len(arr)-1):          #모든 객체의 개수-1만큼 반복
    arr[i].set_xticklabels(())    #마지막에 추가된 객체(마지막 그래프)를 제외하고 x축 라벨을 없앰
```

```
    canvas = FigureCanvasTkAgg(fig,master=window)       #윈도우에 fig크기의 canvas를 생성
    canvas.draw()                                       #canvas에 그래프를 그리기
    canvas.get_tk_widget().place(x=130,y=60)            #canvas를 윈도우의 x = 130 , y = 60의 좌표에
위치시키기

def max_min(start,end,channel):                         #데이터의 최소,최대와 그 시점을 반환하는 함수
                                                        #데이터의 시작 인덱스,끝인덱스,채널을 전달받
음
    temp = signal1[start:end,channel]                   #전달받은 범위와 채널의 데이터를 temp에 저장
    index_min = np.argmin(temp)                         #최소값의 인덱스릴 index_min변수에 저장
    index_max = np.argmax(temp)                         #최대값의 인덱스릴 index_max변수에 저장
    min_time = round(index_min/256)                     #최소값의 인덱스를 256로 나눠서 반올림한후
min_time에 저장, 최소값의 시점
    max_time = round(index_max/256)                     #최대값의 인덱스를 256로 나눠서 반올림한후
max_time에 저장, 최대값의 시점

    return temp[index_min], temp[index_max], min_time, max_time #최소값과 최댓값, 그 시점을 리턴

def write_file():                                       #결과 해석(listbox)부분을 파일에 출력한후 저장하
는 함수
    all_items= listbox.get(0, tkinter.END)          #listbox의 모든 항목을 all_items에 저장
    name = entry1.get()+"_"+entry2.get()+"_"+entry3.get()+"_"+"record.txt"
                                                        #입력한  이름,번호,sampling  rate로  이름_번호
_sampling rate_record.txt을 문자열로
                                                        #생성한후 name변수에 저장
    f = open(name, 'w')                                 #name으로 파일을 생성, 쓰기모드로 열기
    for i in range(len(all_items)):                 #all_items의 길이만큼 반복
        if all_items[i] == "₩n":                    #줄바꿈문자를 만났을 경우
            continue                                    #해당 항목은 건너뛰기
        data = all_items[i] + "₩n"                  #해당 항목에 줄바꿈 문자를 추가해 data변수에 문
자열 형태로 저장
        f.write(data)                                   #data변수의 값을 파일에 쓰기
    f.close()                                           #파일 닫기

def read_file():                                        #파일을 읽는 함수

    f = filedialog.askopenfile(parent=window,mode='r',title='Choose a file') #선택한 파일을 f에 저장
```

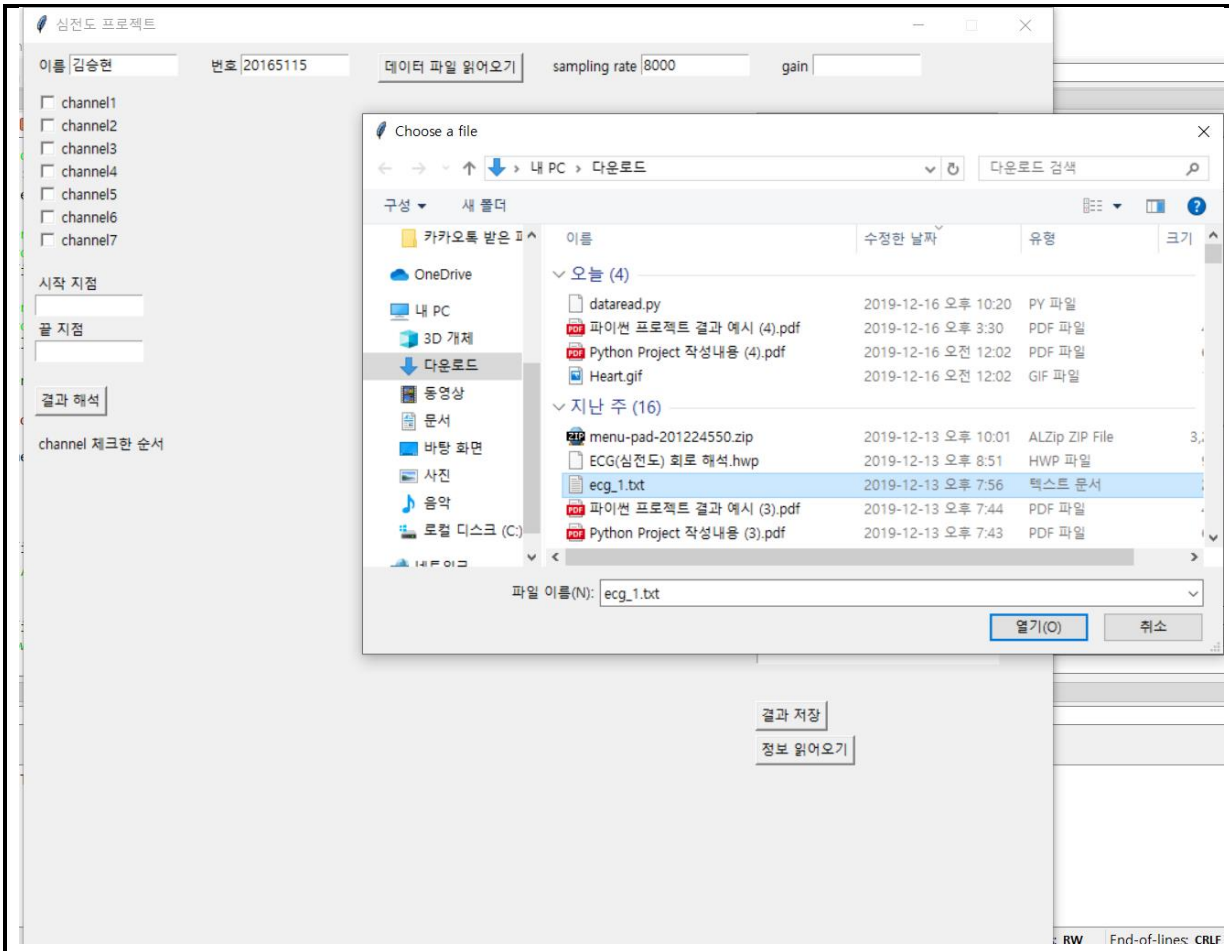
```
    if f != None:                                #f에 파일이 존재하면
        data = f.read()                          #f의 파일을 읽고 data변수에 저장
        f.close()                                #f파일 닫기
        print("I got %d bytes from this file." % len(data)) #data의 길이를 출력

    file = open(f.name, 'r')                     #해당 파일이름으로 'r'모드를 통해 파일을 읽어 file변수에 저
장
    arr_fname = f.name.split('/')                #파일이름에는 이름,번호,sampling rate정보가 있으므로 경로는
"/"단위로 나눠지기 때문에
                                                 #"/"을 기준으로 경로를 쪼갠다. 쪼개서 나온 문자 배열을
arr_fname변수에 저장한다.
    real_file = arr_fname[-1]                    #arr_fname의 마지막 배열의 원소는 파일명을 의미하므로
real_file변수에 저장한다.
    file_name_split = real_file.split("_") #real_file는 이름_번호_주파수_record.txt로 되어있으므로 "_"기준
으로 문자열을 쪼갠후
                                                 #쪼개서 나온 문자 배열을 file_name_split변수에 저장
한다.
    name = file_name_split[0]                    #file_name_split배열의 첫번째 원소는 이름을 나타낸다,
name변수에 저장
    number = file_name_split[1]                  #file_name_split배열의 두번째 원소는 학번을 나타낸다,
number변수에 저장
    hz = file_name_split[2]                      #file_name_split배열의 세번째 원소는 sampling rate을 나
타낸다, hz변수에 저장

    entry1.delete(0,tkinter.END)                 #원래 있던 entry값을 지우기
    entry1.insert(0, name)                       #새로운 entry값(이름) 넣기
    entry2.delete(0,tkinter.END)                 #원래 있던 entry값을 지우기
    entry2.insert(0, number)                     #새로운 entry값(학번) 넣기
    entry3.delete(0,tkinter.END)                 #원래 있던 entry값을 지우기
    entry3.insert(0, hz)                         #새로운 entry값(sampling rate) 넣기

    listbox.delete(0,tkinter.END)                #listbox에 있는 모든 항목 값 지우기
    auto = 0                                     #listbox의 항목 인덱스를 0으로 초기화

    while True:                                  #반복 무한
        line = file.readline()                   #파일을 한줄씩 읽어 line에 저장
        if not line:                             #line에 값이 없으면
            break                                #무한 반복 빠져나오기
```

```
            auto = auto + 1                         #listbox의 항목인덱스 1증가
            listbox.insert(auto,line)           #listbox의 auto인덱스에 line을 항목값을 추가

    file.close()        #파일 닫기


window.geometry("900x800+100+0")       #900x800크기의 윈도우를 (100,0)픽셀 위치에 띄우기
window.resizable(False, False)          #윈도우 크기 사이즈 조정 불가
window.title("심전도 프로젝트")         #윈도우 제목 지정

label1 = tkinter.Label(window,text="이름")    #"이름" label생성
label1.place(x=10, y=10)                          #해당 좌표에 label 위치 시키기

entry1 = tkinter.Entry(window,width=13)       #이름을 입력할 entry생성
entry1.place(x=40, y=10)                          #해당 좌표에 entry 위치 시키기

label2 = tkinter.Label(window,text="번호")    #"번호" label생성
label2.place(x=160, y=10)                         #해당 좌표에 label 위치 시키기

entry2 = tkinter.Entry(window,width=13)       #번호를 입력할 entry생성
entry2.place(x=190, y=10)                         #해당 좌표에 entry 위치 시키기

button1 = tkinter.Button(window, height=1,text="데이터 파일 읽어오기", command=process) #height가
1인 "데이터 파일 읽어오기"버튼 생성
                                                                                          #버튼
을 누르면 process함수와 연결
button1.place(x=310, y=9)                         #해당 좌표에 button을 위치 시키기

label3 = tkinter.Label(window,text="sampling rate")    #"sampling rate" label생성
label3.place(x=460, y=10)                                  #해당 좌표에 label 위치 시키기

entry3 = tkinter.Entry(window,width=13)       #sampling rate를 입력할 entry생성
entry3.place(x=540, y=10)                         #해당 좌표에 entry 위치 시키기

label4 = tkinter.Label(window,text="gain")    #"gain" label생성
label4.place(x=660, y=10)                         #해당 좌표에 label 위치 시키기

entry4 = tkinter.Entry(window,width=13)       #gain를 입력할 entry생성
entry4.place(x=690, y=10)                         #해당 좌표에 entry 위치 시키기
```

```
CheckVariety_1=tkinter.IntVar()                #각변수에 Int를 저장할 물리적 공간 할당
CheckVariety_2=tkinter.IntVar()
CheckVariety_3=tkinter.IntVar()
CheckVariety_4=tkinter.IntVar()
CheckVariety_5=tkinter.IntVar()
CheckVariety_6=tkinter.IntVar()
CheckVariety_7=tkinter.IntVar()

#channel1~7의 체크박스 생성해 변수에 저장, variable은 위에서 물리적 공간을 할당한 변수로 설정
checkbutton1=tkinter.Checkbutton(window, text="channel1", variable=CheckVariety_1)
checkbutton2=tkinter.Checkbutton(window, text="channel2", variable=CheckVariety_2)
checkbutton3=tkinter.Checkbutton(window, text="channel3", variable=CheckVariety_3)
checkbutton4=tkinter.Checkbutton(window, text="channel4", variable=CheckVariety_4)
checkbutton5=tkinter.Checkbutton(window, text="channel5", variable=CheckVariety_5)
checkbutton6=tkinter.Checkbutton(window, text="channel6", variable=CheckVariety_6)
checkbutton7=tkinter.Checkbutton(window, text="channel7", variable=CheckVariety_7)

#각 생성한 체크박스를 해당 좌표에 위치 시키기
checkbutton1.place(x=10, y=40)
checkbutton2.place(x=10, y=60)
checkbutton3.place(x=10, y=80)
checkbutton4.place(x=10, y=100)
checkbutton5.place(x=10, y=120)
checkbutton6.place(x=10, y=140)
checkbutton7.place(x=10, y=160)

label5 = tkinter.Label(window,text="시작 지점")    #"시작 지점" label생성
label5.place(x=10, y=200)                          #해당 좌표에 label 위치 시키기

entry5 = tkinter.Entry(window,width=13)       #시작지점를 입력할 entry생성
entry5.place(x=10, y=220)                      #해당 좌표에 entry를 위치시키기끝

label6 = tkinter.Label(window,text="끝 지점")    #"끝 지점" label생성
label6.place(x=10, y=240)                          #해당 좌표에 label 위치 시키기

entry6 = tkinter.Entry(window,width=13)       #지점를 입력할 entry생성
entry6.place(x=10, y=260)                      #해당 좌표에 entry를 위치시키기끝
```

```
button2 = tkinter.Button(window, height=1,text="결과 해석", command=graph) #height가 1인 "결과 해석"버튼 생성
                                                                                #버튼을 누르면 graph함수와 연결
button2.place(x=10, y=300)                                                      #해당 좌표에 button을 위치 시키기

label7 = tkinter.Label(window,text="channel 체크한 순서")                       #"channel 체크한 순서" label생성
label7.place(x=10, y=340)                                                       #해당 좌표에 label 위치 시키기

listbox = tkinter.Listbox(window, selectmode='extended', height=30,width=30) #listbox를 높이30, 너비30크기로 생성, 다중선택모드
listbox.place(x=640,y=60)                                                       #해당 좌표에 listbox 위치 시키기

button3 = tkinter.Button(window, height=1,text="결과 저장", command=write_file) #height가 1인 "결과 저장"버튼 생성
                                                                                #버튼을 누르면 write_file함수와 연결
button3.place(x=640, y=575)                                                     #해당 좌표에 button 위치 시키기

button4 = tkinter.Button(window, height=1,text="정보 읽어오기", command=read_file)  #height가 1인 "정보 읽어오기"버튼 생성
                                                                                #버튼을 누르면 read_file함수와 연결
button4.place(x=640, y=605)                                                     #해당 좌표에 button 위치 시키기


window.mainloop()                                                               #윈도우 띄우기
```

Ⅲ. 구현 결과



이름, 번호,sampling rate,gain을 입력하고 데이터파일을 읽어오기 버튼을 누릅니다.
그러면 위의 그림과 같이 파일을 선택하는 창이 뜹니다.
해당 창에서 파일을 선택하면 자동으로 파일의 데이터를 읽습니다.

```
def process():              #데이터 파일을 로드하는 함수
    file = filedialog.askopenfile(parent=window,mode='r',title='Choose a file')  #파일 선택
    if file != None:                    #파일이 존재한다면
        data = file.read()              #파일의 데이터 읽기
        file.close()                    #파일 닫기
        print("I got %d bytes from this file." % len(data))        #파일의 길이 출력

    global signal1                      #데이터파일을 저장할 변수를 global 변수로 선언
    signal1 = np.loadtxt(file.name,delimiter = ',')  #데이터 파일을 읽어 signal변수에 저장
```

체크박스에서 채널을 선택하고 시작 지점과 끝 지점(초 단위)을 입력한후 결과 해석 버튼을 누르면 위의 그림과 같이 체크박스에서 선택한 채널의 순서대로 그래프가 출력되게 됩니다.
listbox에는 각 채널의 최소,최대값과 그 시점의 정보들이 출력되게 됩니다.

```
def graph():                    #그래프를 그리는 함수
    fig = Figure(figsize=(5,7),dpi=100)       #그래프를 그릴 범위를 500 pixel x 700 pixel로 잡아 fig객체에 저장

    arr = []                                  #생성한 그래프 객체를 저장할 배열, x축을 마지막 그래프만 출력하게 하기 위함

    global start            #start변수를 global변수로 선언
    global end              #end변수를  global 변수로 선언
    start = int(entry5.get())  #시점으로 입력한 값을 가져와 start에 저장
    end = int(entry6.get())    #끝으로 입력한 값을 가져와 end에 저장
```

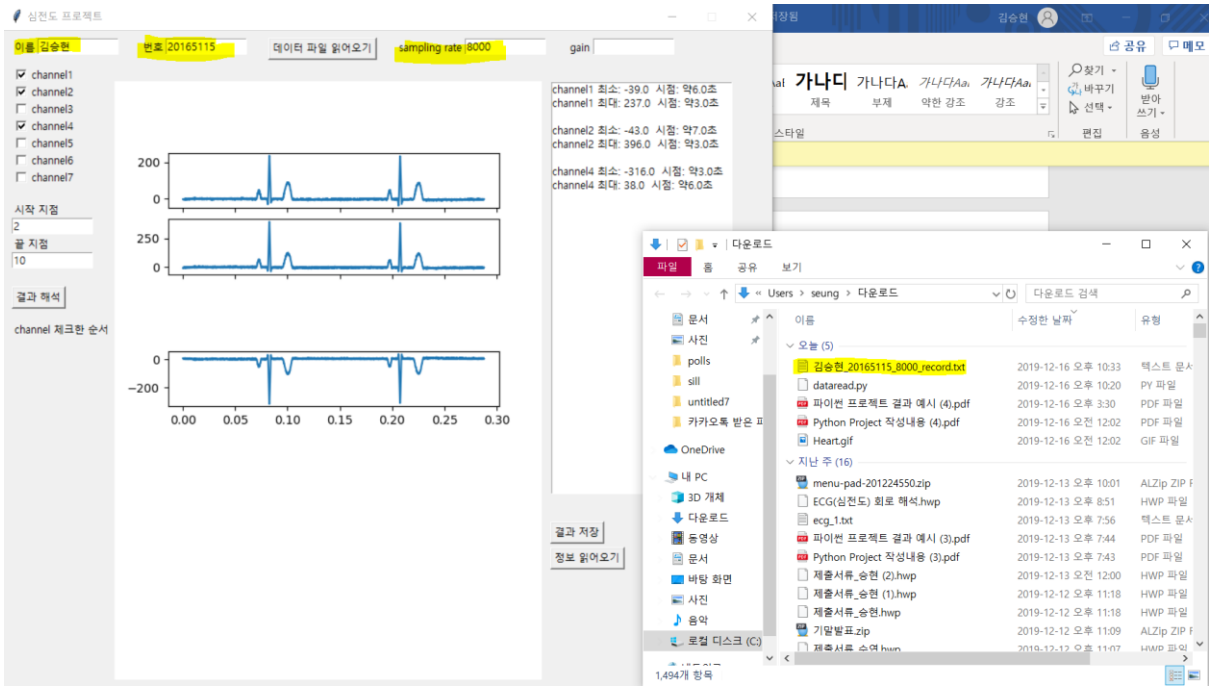
```
    global start_2                #start_2를 global변수로 선언
    global end_2                   #end_2를 global변수로 선언
    start_2 = start * 256         #초당 256개의 데이터를 읽으므로 start(초)*256를 계산해 시작 시점의 데이터 인덱스를 start_2에 저장
    end_2 = start*256 + (end-start+1)*256 #시작시점데이터인덱스 + (end(초)-start(초)+1)*256를 계산해 끝 시점의 데이터 인덱스를 end_2에 저장

    if start == 0 and end== 0: #start와 end의 시점이 0이면 end_2에 0을 저장,start*256 + (end-start+1)*256가 0이 안되기 때문
        end_2 = 0

    listbox.delete(0,tkinter.END) #listbox에 있는 모든 항목들을 삭제

    auto = 0            #listbox에 넣을 항목의 인덱스를 0을 초기화

    if CheckVariety_1.get() == 1:      #channel1의 체크박스에 체크되어있을 경우
        x = signal1[start_2:end_2,0] #start_2애서 end_2범위의 channel1데이터를 추출하여 x에 저장
        t = np.arange(len(x))*1.0/int(entry3.get()) #[0,1,2,3...] * 1/sampling rate(hz,주파수)의 공식을 적용해
                                                    #해당 채널의 데이터를 sampling한다, sampling된값은 t에 저장
                                                    #주파수는 입력한 주파수를 이용
        ax1 = fig.add_subplot(7,1,1)                #fig객처를 사용해 7행 1열의 첫번째 위치에 그래프를 위치설정, 객체 반환
        ax1.plot(t,x)                               #반환된 객체를 이용해 x축은 t값으로, y축은 x값으로 한다.
        arr.append(ax1)                             #그래프 객체를 arr배열에 추가한다.
        min_val, max_val,min_time,max_time = max_min(start_2,end_2,0)  #max_min함수에 start_2와 end_2,채널을 전달해 최대,최솟값과
                                                                       #그 시점을 반환받아 min_val, max_val,min_time,max_time에 각각
                                                                       #저장한다.

        auto = auto + 1                             #listbox의 항목 인덱스를 1증가 시킨다
        listbox.insert(auto,"channel"+str(1)+" 최소: "+ str(min_val)+"  시점: 약"+str(min_time)+"초")
                                                    #증가시킨 항목인덱스에 채널명,전달받은 최소값, 최소값의 시점을 항목값으로 하여 listbox에 삽입
        auto = auto + 1                             #listbox의 항목 인덱스를 1증가 시킨다
```

```
        listbox.insert(auto,"channel"+str(1)+" 최대: "+ str(max_val)+"  시점: 약"+str(max_time)+"초")
                                                    #증가시킨 항목인덱스에 채널명,전달받은 최소값, 최소값의 시점을 항목값으로 하여 listbox에 삽입

        auto = auto + 1                             #listbox의 항목 인덱스를 1증가 시킨다
        listbox.insert(auto,"₩n")                   #증가시킨 항목 인덱스에 "₩n"을 항목값으로 하여 listbox에 추가한다.
                                                    #가독성을 위해
    #다음의 6개 if문은 위의 if문의 설명과 동일함
    if CheckVariety_2.get() == 1:
        x = signal1[start_2:end_2,1]
        t = np.arange(len(x))*1.0/int(entry3.get())
        ax2 = fig.add_subplot(7,1,2)
        ax2.plot(t,x)
        arr.append(ax2)

        min_val, max_val,min_time,max_time = max_min(start_2,end_2,1)

        auto = auto + 1
        listbox.insert(auto,"channel"+str(2)+" 최소: "+ str(min_val)+"  시점: 약"+str(min_time)+"초")
        auto = auto + 1
        listbox.insert(auto,"channel"+str(2)+" 최대: "+ str(max_val)+"  시점: 약"+str(max_time)+"초")

        auto = auto + 1
        listbox.insert(auto,"₩n")

    if CheckVariety_3.get() == 1:
        x = signal1[start_2:end_2,2]
        t = np.arange(len(x))*1.0/int(entry3.get())
        ax3 = fig.add_subplot(7,1,3)
        ax3.plot(t,x)
        arr.append(ax3)

        min_val, max_val,min_time,max_time = max_min(start_2,end_2,2)

        auto = auto + 1
        listbox.insert(auto,"channel"+str(3)+" 최소: "+ str(min_val)+"  시점: 약"+str(min_time)+"초")
        auto = auto + 1
        listbox.insert(auto,"channel"+str(3)+" 최대: "+ str(max_val)+"  시점: 약"+str(max_time)+"초")
```

```
        auto = auto + 1
        listbox.insert(auto,"₩n")

    if CheckVariety_4.get() == 1:
        x = signal1[start_2:end_2,3]
        t = np.arange(len(x))*1.0/int(entry3.get())
        ax4 = fig.add_subplot(7,1,4)
        ax4.plot(t,x)
        arr.append(ax4)

        min_val, max_val,min_time,max_time = max_min(start_2,end_2,3)

        auto = auto + 1
        listbox.insert(auto,"channel"+str(4)+" 최소: "+ str(min_val)+"   시점: 약"+str(min_time)+"초")
        auto = auto + 1
        listbox.insert(auto,"channel"+str(4)+" 최대: "+ str(max_val)+"   시점: 약"+str(max_time)+"초")

        auto = auto + 1
        listbox.insert(auto,"₩n")

    if CheckVariety_5.get() == 1:
        x = signal1[start_2:end_2,4]
        t = np.arange(len(x))*1.0/int(entry3.get())
        ax5 = fig.add_subplot(7,1,5)
        ax5.plot(t,x)
        arr.append(ax5)

        min_val, max_val,min_time,max_time = max_min(start_2,end_2,4)

        auto = auto + 1
        listbox.insert(auto,"channel"+str(5)+" 최소: "+ str(min_val)+"   시점: 약"+str(min_time)+"초")
        auto = auto + 1
        listbox.insert(auto,"channel"+str(5)+" 최대: "+ str(max_val)+"   시점: 약"+str(max_time)+"초")

        auto = auto + 1
        listbox.insert(auto,"₩n")

    if CheckVariety_6.get() == 1:
```

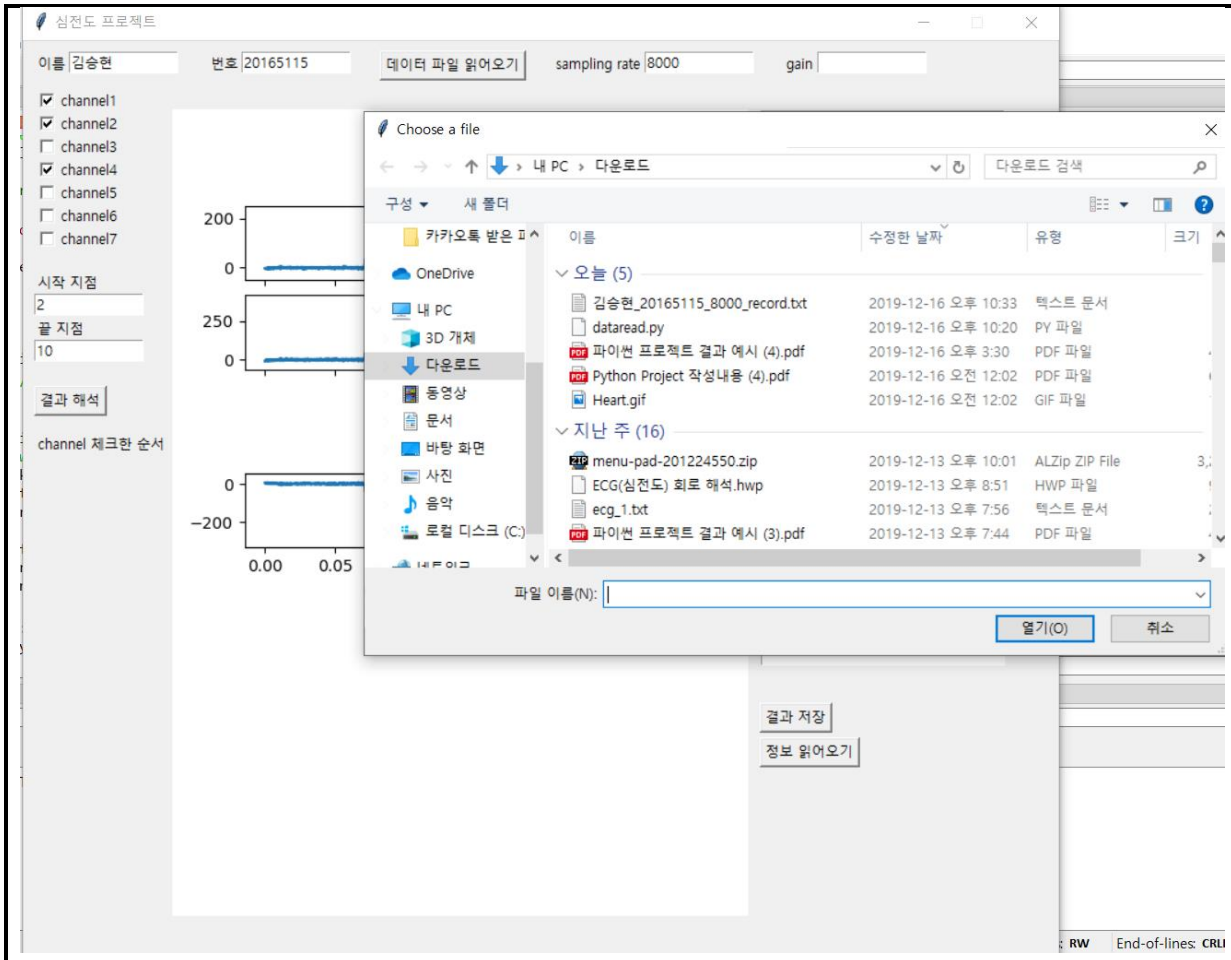
```
        x = signal1[start_2:end_2,5]
        t = np.arange(len(x))*1.0/int(entry3.get())
        ax6 = fig.add_subplot(7,1,6)
        ax6.plot(t,x)
        arr.append(ax6)
        max_min(start_2,end_2,5)

        min_val, max_val,min_time,max_time = max_min(start_2,end_2,5)

        auto = auto + 1
        listbox.insert(auto,"channel"+str(6)+" 최소: "+ str(min_val)+"   시점: 약"+str(min_time)+"초")
        auto = auto + 1
        listbox.insert(auto,"channel"+str(6)+" 최대: "+ str(max_val)+"   시점: 약"+str(max_time)+"초")

        auto = auto + 1
        listbox.insert(auto,"₩n")

    if CheckVariety_7.get() == 1:
        x = signal1[start_2:end_2,6]
        t = np.arange(len(x))*1.0/int(entry3.get())
        ax7 = fig.add_subplot(7,1,7)
        ax7.plot(t,x)
        arr.append(ax7)
        max_min(start_2,end_2,6)

        min_val, max_val,min_time,max_time = max_min(start_2,end_2,6)

        auto = auto + 1
        listbox.insert(auto,"channel"+str(7)+" 최소: "+ str(min_val)+"   시점: 약"+str(min_time)+"초")
        auto = auto + 1
        listbox.insert(auto,"channel"+str(7)+" 최대: "+ str(max_val)+"   시점: 약"+str(max_time)+"초")

        auto = auto + 1
        listbox.insert(auto,"₩n")

    for i in range(len(arr)-1):          #모든 객체의 개수-1만큼 반복
        arr[i].set_xticklabels(())    #마지막에 추가된 객체(마지막 그래프)를 제외하고 x축 라벨을 없앰

    canvas = FigureCanvasTkAgg(fig,master=window)       #윈도우에 fig크기의 canvas를 생성
```

```
    canvas.draw()                                   #canvas에 그래프를 그리기
    canvas.get_tk_widget().place(x=130,y=60)        #canvas를 윈도우의 x = 130 , y = 60의 좌표에
위치시키기

def max_min(start,end,channel):                     #데이터의 최소,최대와 그 시점을 반환하는 함수
                                                    #데이터의 시작 인덱스,끝인덱스,채널을 전달받
음
    temp = signal1[start:end,channel]               #전달받은 범위와 채널의 데이터를 temp에 저장
    index_min = np.argmin(temp)                     #최소값의 인덱스릴 index_min변수에 저장
    index_max = np.argmax(temp)                     #최대값의 인덱스릴 index_max변수에 저장
    min_time = round(index_min/256)                 #최소값의 인덱스를 256로 나눠서 반올림한후
min_time에 저장, 최소값의 시점
    max_time = round(index_max/256)                 #최대값의 인덱스를 256로 나눠서 반올림한후
max_time에 저장, 최대값의 시점

    return temp[index_min], temp[index_max], min_time, max_time #최소값과 최댓값, 그 시점을 리턴
```



결과 저장버튼을 누르면 입력한 이름, 번호, sampling rate를 통해 이름_번호_sampling rate_record.txt 명으로 listbox에 출력됐던 결과값을 저장할 수 있습니다. 위의 그림을 보면 해당 이름으로 저장된 파일을 확인할 수 있습니다.

```
def write_file():                                   #결과 해석(listbox)부분을 파일에 출력한후 저장하는 함수
    all_items= listbox.get(0, tkinter.END)          #listbox의 모든 항목을 all_items에 저장
    name = entry1.get()+"_"+entry2.get()+"_"+entry3.get()+"_"+"record.txt"
                                                    #입력한 이름,번호,sampling rate로 이름_번호_sampling rate_record.txt을 문자열로
                                                    #생성한후 name변수에 저장
    f = open(name, 'w')                             #name으로 파일을 생성, 쓰기모드로 열기
    for i in range(len(all_items)):                 #all_items의 길이만큼 반복
        if all_items[i] == "₩n":                    #줄바꿈문자를 만났을 경우
            continue                                #해당 항목은 건너뛰기
        data = all_items[i] + "₩n"                  #해당 항목에 줄바꿈 문자를 추가해 data변수에 문자열 형태로 저장
        f.write(data)                               #data변수의 값을 파일에 쓰기
    f.close()                                       #파일 닫기
```



정보 읽어오기 버튼을 누르면 읽어올 파일을 선택할 수 있습니다.

읽어온 파일은 다음과 같이 이름, 번호,samplig rate, listbox에 정보가 출력됩니다.

```
def read_file():                                    #파일을 읽는 함수

    f = filedialog.askopenfile(parent=window,mode='r',title='Choose a file') #선택한 파일을 f에 저장

    if f != None:                                   #f에 파일이 존재하면
        data = f.read()                             #f의 파일을 읽고 data변수에 저장
        f.close()                                   #f파일 닫기
        print("I got %d bytes from this file." % len(data)) #data의 길이를 출력

    file = open(f.name, 'r')                        #해당 파일이름으로 'r'모드를 통해 파일을 읽어 file변수에 저장
    arr_fname = f.name.split('/')                   #파일이름에는 이름,번호,sampling rate정보가 있으므로 경로는 "/"단위로 나눠지기 때문에
```

```
                                        #"/"을 기준으로 경로를 쪼갠다. 쪼개서 나온 문자 배열을 arr_fname변수에 저장한다.
    real_file = arr_fname[-1]           #arr_fname의 마지막 배열의 원소는 파일명을 의미하므로 real_file변수에 저장한다.
    file_name_split = real_file.split("_") #real_file는 이름_번호_주파수_record.txt로 되어있으므로 "_"기준으로 문자열을 쪼갠후
                                        #쪼개서 나온 문자 배열을 file_name_split변수에 저장한다.
    name = file_name_split[0]           #file_name_split배열의 첫번째 원소는 이름을 나타낸다, name변수에 저장
    number = file_name_split[1]         #file_name_split배열의 두번째 원소는 학번을 나타낸다, number변수에 저장
    hz = file_name_split[2]             #file_name_split배열의 세번째 원소는 sampling rate을 나타낸다, hz변수에 저장

    entry1.delete(0,tkinter.END)        #원래 있던 entry값을 지우기
    entry1.insert(0, name)              #새로운 entry값(이름) 넣기
    entry2.delete(0,tkinter.END)        #원래 있던 entry값을 지우기
    entry2.insert(0, number)            #새로운 entry값(학번) 넣기
    entry3.delete(0,tkinter.END)        #원래 있던 entry값을 지우기
    entry3.insert(0, hz)                #새로운 entry값(sampling rate) 넣기

    listbox.delete(0,tkinter.END)       #listbox에 있는 모든 항목 값 지우기
    auto = 0                            #listbox의 항목 인덱스를 0으로 초기화

    while True:                         #반복 무한
        line = file.readline()          #파일을 한줄씩 읽어 line에 저장
        if not line:                    #line에 값이 없으면
            break                       #무한 반복 빠져나오기

        auto = auto + 1                 #listbox의 항목인덱스 1증가
        listbox.insert(auto,line)       #listbox의 auto인덱스에 line을 항목값을 추가

    file.close()        #파일 닫기
```

Ⅳ. 기대효과

프로젝트를 진행하면서 4년동안 배웠던 파이썬을 한번에 정리할 수 있었습니다. 또한 그전에는 파이썬이라는 언어를 머신러닝에만 적용해서 사용해왔는데 tkinter를 처음으로 활용해 GUI를 구현할 수 있어

서 매우 유익한 시간이었습니다. 비록 gain을 데이터에 적용하는 식을 구하진 못하고 GUI만 구현해 프로젝트를 완벽하게 만들진 못했지만, tkinter말고도 다른 GUI프로그램을 통해 프로그램을 시각화 할 수 있는 능력이 생긴거 같아 뿌듯했습니다. 앞으로도 파이썬을 잘 활용하여 GUI프로그램 말고도 여러 프로그램을 개발하는 유능한 파이썬 프로그래머가 되고 싶습니다.