ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

DIPARTIMENTO DI

INGEGNERIA DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE
"GUGLIELMO MARCONI"

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA BIOMEDICA

ANALISI DEL SEGNALE ELETTROENCEFALOGRAFICO ACQUISITO DURANTE MOVIMENTI LENTI E VELOCI DELL'ARTO SUPERIORE

Tesi in
Sistemi Neurali LM

Relatore:

Prof.ssa Elisa Magosso

Presentata da:

Monia Ricci

Correlatori:

Prof.ssa Silvia Fantozzi

Dott. Davide Borra

ALLEGATO

Anno Accademico 2018/2019

Codice per l’interfaccia grafica scritto in linguaggio Python:

```
from __future__ import division
from __future__ import print_function
from __future__ import absolute_import
from gpiozero import OutputDevice
from tkinter import *
import tkinter as tk
import pygame
import datetime
import time
import os
import random
import pickle
import configparser
import numpy as np
pygame.mixer.init()
pygame.mixer.music.load("beep_880_010.wav")

config = configparser.ConfigParser()
config.read('config.ini')
n_trials = int(config['TEMPORAL PARS']['n_trials'])
t0 = int(config['TEMPORAL PARS']['t0'])
t1_min = int(config['TEMPORAL PARS']['t1_min'])
t1_max = int(config['TEMPORAL PARS']['t1_max'])
t1 = np.random.randint(t1_min, t1_max, n_trials)
t2 = int(config['TEMPORAL PARS']['t2'])
conditions = config['TEMPORAL PARS']['conditions'].split(',')

trigger_pin = 4
trigger = OutputDevice(trigger_pin)

class MyClass:
    def __init__(self, parent):
        self.myParent = parent
        self.myContainer = tk.Frame(parent)
        self.myContainer.configure(background='black')
        self.myContainer.pack()
        self.button = tk.Button(self.myContainer,
             command=self.buttonClick)
        self.button.configure(text='START', fg = "white", width=10,
             background = "blue", font = ("Muli", 30, "bold"))
        self.button.pack(pady=100)
        self.button_esc = tk.Button(self.myParent,
             command=self.myParent.destroy)
        self.button_esc.configure(text='X', fg = "gray", width=5,
             background = "black", font = ("Muli", 10, "bold"))
        self.button_esc.pack(anchor=E, side=BOTTOM)
        root.bind('<Return>')
        my_list = []
        for i in range(len(conditions)):
            my_list.extend([i]*int(n_trials/(len(conditions))))
        random.shuffle(my_list)
        my_list.append(len(conditions))
        print(my_list)
        print(t1)
        self.sequence = my_list
        self.t0 = t0
        self.t1 = t1
        self.t2 = t2
        self.trigger = trigger
        self.conditions = conditions

    def buttonClick(self):
        self.button.destroy()
        self.myParent.config(cursor="none")
        self.label = tk.Label(text="+", font="Muli 30", fg=”white",
             background='black')
        self.label.pack(anchor=N)
        self.counter = 0
        now = datetime.datetime.now()
        self.current_datetime = str(now.year).zfill(4) +'_'+
                                str(now.month).zfill(2) +'_'+
                                str(now.day).zfill(2)+'_'+
                                str(now.hour).zfill(2)+'_'+
                                str(now.minute).zfill(2)
        self.label.after(1000, self.refresh_label)

    def clear_after(self):
        pygame.mixer.music.play()
        self.trigger.on()
        self.label.config(text="+")
        self.label.after(self.t2, self.refresh_label)

    def button_end(self):
        self.button_end = tk.Button(self.myContainer,
             command=self.myParent.destroy)
        self.button_end.configure(text='QUIT',fg="white",width=10,
             background = "blue", font = ("Muli", 30, "bold"))
        save_dir = "recordings_"+self.current_datetime
        if not os.path.isdir(save_dir):
             os.mkdir(save_dir)
        with open(os.path.join(save_dir,'config.ini'),'w') as
            configfile:
              config.write(configfile)
        np.savetxt(os.path.join(save_dir, 'sequence.csv'),
            np.array(self.sequence), delimiter=',')
        np.savetxt(os.path.join(save_dir, 'timings.csv'), self.t1,
            delimiter=',')
        self.button_end.pack(pady=100)

    def refresh_label(self):
        self.trigger.off()
        x = self.sequence[self.counter]
        if x!=len(self.conditions):
            self.label.config(text=self.conditions[x], fg="white",
                 background="black")
            self.label.after(int(self.t1[self.counter]),
                 self.clear_after)
        else:
            self.myParent.config(cursor="arrow")
            self.label.config(text="")
            self.label.after(1000, self.button_end)
        self.counter += 1

root = tk.Tk()
myClass = MyClass(root)
root.attributes('-fullscreen', True)
root.configure(background='black')
root.mainloop()
```