МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум № 4**

з дисципліни «**Методи та системи штучного інтелекту**»

на тему: «МОДЕЛЮВАННЯ ШТУЧНОЇ ЕКГ З АЛЬТЕРНАЦІЄЮ ЗУБЦЯ Т»

**Виконала:**

студентка гр. БС-12

Сипко В.І.

**Перевірив:**

доц. каф. БМК

Піднебесна Г.А.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2023

**Мета роботи**: реалізувати програмний модуль побудови штучної електрокардіограми з альтернацією зубця Т.

**Основні задачі роботи**:

1. Освоїти технологію моделювання штучних сигналів складної форми.

2. Навчитись використовувати параметри еталону моделі циклу штучної ЕКГ.

3. Згенерувати спотворені послідовні цикли штучної ЕКГ.

4. Програмно реалізувати моделювання електричної альтернації зубця Т.

**Лістинг програми**

import tkinter as tk  
import numpy as np  
import matplotlib.pyplot as plt  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
  
win = tk.Tk()  
win.title("Cardiocycle model")  
win.geometry("920x600+300+50")  
win.resizable(False, False)  
  
param\_static = {"P": [0.14, 0.025, 0.026, 0.273],  
 "Q": [-0.15, 0.003, 0.008, 0.392],  
 "R": [1.453, 0.0091, 0.01, 0.4446],  
 "S": [-0.34, 0.0115, 0.0417, 0.498],  
 "ST": [0.03, 0.1, 0.024, 0.555],  
 "T": [0.45, 0.026, 0.044, 0.734]}  
  
param\_changeble = {"P": [0.14, 0.025, 0.026, 0.273],  
 "Q": [-0.15, 0.003, 0.008, 0.392],  
 "R": [1.453, 0.0091, 0.01, 0.4446],  
 "S": [-0.34, 0.0115, 0.0417, 0.498],  
 "ST": [0.03, 0.1, 0.024, 0.555],  
 "T": [0.45, 0.026, 0.044, 0.734]}  
  
def del\_frames():  
 try:  
 P\_frame.destroy()  
 except:  
 print('no such frame')  
 try:  
 Q\_frame.destroy()  
 except:  
 print('no such frame')  
 try:  
 R\_frame.destroy()  
 except:  
 print('no such frame')  
 try:  
 S\_frame.destroy()  
 except:  
 print('no such frame')  
 try:  
 ST\_frame.destroy()  
 except:  
 print('no such frame')  
 try:  
 T\_frame.destroy()  
 except:  
 print('no such frame')  
  
  
fig, ax = plt.subplots()  
  
  
def main(params, hr):  
 Fs = hr \* 1000  
 t0 = 60 \* 1000 / Fs  
  
 def find\_t(a):  
 t1 = params[a][3] - 3 \* params[a][1]  
 t2 = params[a][3] + 3 \* params[a][2]  
 return t1, t2  
  
 def find\_b(a, t):  
 if t <= params[a][3]:  
 return params[a][1]  
 else:  
 return params[a][2]  
  
 def Gauss(a, time):  
 return params[a][0] \* np.exp(-((time - params[a][3]) \*\* 2) / ((2 \* find\_b(a, time) \*\* 2)))  
  
 global t  
 t = np.arange(0, t0, 0.001)  
 global x  
 x = []  
  
 for i in t:  
 if (find\_t('P')[0] <= i < find\_t('P')[1]):  
 x.append(Gauss("P", i))  
 elif (i >= find\_t("Q")[0] and i < find\_t("Q")[1]):  
 x.append(Gauss("Q", i))  
 elif (i >= find\_t("R")[0] and i < find\_t("R")[1]):  
 x.append(Gauss("R", i))  
 elif (i >= find\_t("S")[0] and i < find\_t("S")[1]):  
 x.append(Gauss("S", i))  
 elif (i >= find\_t("ST")[0] and i < find\_t("ST")[1]):  
 x.append(Gauss("ST", i))  
 elif (i >= find\_t("T")[0] and i < find\_t("T")[1]):  
 x.append(Gauss("T", i))  
 else:  
 x.append(0)  
 def plot():  
 ax.clear()  
 ax.plot(t, x, color = "#444444", linewidth =1)  
 ax.grid(True, linewidth=0.9, color="#c8e6e9")  
 ax.set\_xlabel("c", loc='right')  
 ax.set\_ylabel("мВ", loc='top', rotation=0, labelpad=0.1)  
 fig.patch.set\_facecolor('#F0F0E2')  
 canvas.draw()  
  
 canvas = FigureCanvasTkAgg(fig, master=main\_frame)  
 canvas.get\_tk\_widget().place(x=240, y=1, width=640, height=500)  
 plot()  
  
def get\_amp\_P(val):  
 param\_changeble["P"][0] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_b1\_P(val):  
 param\_changeble["P"][1] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_b2\_P(val):  
 param\_changeble["P"][2] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_t\_P(val):  
 param\_changeble["P"][3] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_amp\_Q(val):  
 param\_changeble["Q"][0] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_b1\_Q(val):  
 param\_changeble["Q"][1] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_b2\_Q(val):  
 param\_changeble["Q"][2] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_t\_Q(val):  
 param\_changeble["Q"][3] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_amp\_R(val):  
 param\_changeble["R"][0] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_b1\_R(val):  
 param\_changeble["R"][1] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_b2\_R(val):  
 param\_changeble["R"][2] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_t\_R(val):  
 param\_changeble["R"][3] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_amp\_S(val):  
 param\_changeble["S"][0] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_b1\_S(val):  
 param\_changeble["S"][1] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_b2\_S(val):  
 param\_changeble["S"][2] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_t\_S(val):  
 param\_changeble["S"][3] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_amp\_ST(val):  
 param\_changeble["ST"][0] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_b1\_ST(val):  
 param\_changeble["ST"][1] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_b2\_ST(val):  
 param\_changeble["ST"][2] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_t\_ST(val):  
 param\_changeble["ST"][3] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_amp\_T(val):  
 param\_changeble["T"][0] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_b1\_T(val):  
 param\_changeble["T"][1] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_b2\_T(val):  
 param\_changeble["T"][2] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def get\_t\_T(val):  
 param\_changeble["T"][3] = float(val)  
 main(param\_changeble, int(hr.get()))  
  
def labels(text, x, y):  
 tk.Label(main\_frame, text=text,  
 bg="white",  
 font=("Arial", 12),  
 width=11,  
 height=1,  
 # cursor="sizing",  
 relief=tk.RAISED).place(x=x, y=y)  
  
def scale\_amp(frame, x, y, func):  
 tk.Scale(frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=-1,  
 to=1.7,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=func,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.01).place(x=x, y=y)  
  
def scale\_t(frame, x, y, func):  
 tk.Scale(frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=0,  
 to=0.9,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=func,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.001).place(x=x, y=y)  
  
def scales(frame, x, y, func):  
 tk.Scale(frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=-0.1,  
 to=0.1,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=func,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.001).place(x=x, y=y)  
  
hr = tk.StringVar()  
def entry(frame):  
 tk.Entry(frame,  
 bg='#d4eeee',  
 fg='#444444',  
 justify='center',  
 width=5,  
 takefocus=0,  
 textvariable=hr).place(x=18, y=329)  
  
def P\_page():  
 del\_frames()  
 global P\_frame  
 P\_frame = tk.Frame(main\_frame, bg="#F0F0E2")  
 P\_frame.pack(side='left')  
  
 labels("Amplitude", 15, 15)  
 labels("Width (b1)", 15, 87)  
 labels("Width (b2)", 15, 159)  
 labels("Time", 15, 231)  
 labels("Heart rate", 15, 303)  
  
 scale\_amp(P\_frame, 15, 42, get\_amp\_P)  
 scales(P\_frame, 15, 114, get\_b1\_P)  
 scales(P\_frame, 15, 186, get\_b2\_P)  
 scale\_t(P\_frame, 15, 258, get\_t\_P)  
  
 entry(P\_frame)  
  
 P\_frame.pack\_propagate(False)  
 P\_frame.configure(width=230, height=800)  
  
def Q\_page():  
 del\_frames()  
 global Q\_frame  
 Q\_frame = tk.Frame(main\_frame, bg="#F0F0E2")  
 Q\_frame.pack(side='left')  
  
 labels("Amplitude", 70, 15)  
 labels("Width (b1)", 70, 87)  
 labels("Width (b2)", 70, 159)  
 labels("Time", 70, 231)  
 labels("Heart rate", 70, 303)  
  
 scale\_amp(Q\_frame, 15, 42, get\_amp\_Q)  
 tk.Scale(Q\_frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=-0.01,  
 to=0.05,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=get\_b1\_Q,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.001).place(x=15, y=114)  
 tk.Scale(Q\_frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=-0.005,  
 to=0.01,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=get\_b2\_Q,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.001).place(x=15, y=186)  
 tk.Scale(Q\_frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=0.3,  
 to=0.5,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=get\_t\_Q,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.001).place(x=15, y=258)  
  
 entry(Q\_frame)  
  
 Q\_frame.pack\_propagate(False)  
 Q\_frame.configure(width=230, height=800)  
  
def R\_page():  
 del\_frames()  
 global R\_frame  
 R\_frame = tk.Frame(main\_frame, bg="#F0F0E2")  
 R\_frame.pack(side='left')  
  
 labels("Amplitude", 70, 15)  
 labels("Width (b1)", 70, 87)  
 labels("Width (b2)", 70, 159)  
 labels("Time", 70, 231)  
 labels("Heart rate", 70, 303)  
  
 scale\_amp(R\_frame, 15, 42, get\_amp\_R)  
 tk.Scale(R\_frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=-0.004,  
 to=0.01,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=get\_b1\_R,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.0001).place(x=15, y=114)  
 tk.Scale(R\_frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=0,  
 to=0.01,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=get\_b2\_R,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.0001).place(x=15, y=186)  
 tk.Scale(R\_frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=0.46,  
 to=0.4,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=get\_t\_R,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.0001).place(x=15, y=258)  
  
 entry(R\_frame)  
  
 R\_frame.pack\_propagate(False)  
 R\_frame.configure(width=230, height=800)  
  
def S\_page():  
 del\_frames()  
 global S\_frame  
 S\_frame = tk.Frame(main\_frame, bg="#F0F0E2")  
 S\_frame.pack(side='left')  
  
 labels("Amplitude", 15, 15)  
 labels("Width (b1)", 15, 87)  
 labels("Width (b2)", 15, 159)  
 labels("Time", 15, 231)  
 labels("Heart rate", 15, 303)  
  
 scale\_amp(S\_frame, 15, 42, get\_amp\_S)  
 tk.Scale(S\_frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=-0.007,  
 to=0.038,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=get\_b1\_S,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.0001).place(x=15, y=114)  
 tk.Scale(S\_frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=0.003,  
 to=0.1,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=get\_b2\_S,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.0001).place(x=15, y=186)  
 scale\_t(S\_frame, 15, 258, get\_t\_S)  
  
 entry(S\_frame)  
  
 S\_frame.pack\_propagate(False)  
 S\_frame.configure(width=230, height=800)  
  
def ST\_page():  
 del\_frames()  
 global ST\_frame  
 ST\_frame = tk.Frame(main\_frame, bg="#F0F0E2")  
 ST\_frame.pack(side='left')  
  
 labels("Amplitude", 15, 15)  
 labels("Width (b1)", 15, 87)  
 labels("Width (b2)", 15, 159)  
 labels("Time", 15, 231)  
 labels("Heart rate", 15, 303)  
  
 scale\_amp(ST\_frame, 15, 42, get\_amp\_ST)  
 scales(ST\_frame, 15, 114, get\_b1\_ST)  
 scales(ST\_frame, 15, 186, get\_b2\_ST)  
 tk.Scale(ST\_frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=0.4,  
 to=0.65,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=get\_t\_ST,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.001).place(x=15, y=258)  
  
 entry(ST\_frame)  
  
 ST\_frame.pack\_propagate(False)  
 ST\_frame.configure(width=230, height=800)  
  
def T\_page():  
 del\_frames()  
 global T\_frame  
 T\_frame = tk.Frame(main\_frame, bg="#F0F0E2")  
 T\_frame.pack(side='left')  
  
 labels("Amplitude", 15, 15)  
 labels("Width (b1)", 15, 87)  
 labels("Width (b2)", 15, 159)  
 labels("Time", 15, 231)  
 labels("Heart rate", 15, 303)  
  
 scale\_amp(T\_frame, 15, 42, get\_amp\_T)  
 scales(T\_frame, 15, 114, get\_b1\_T)  
 scales(T\_frame, 15, 186, get\_b2\_T)  
 tk.Scale(T\_frame,  
 orient="horizontal",  
 bg="#d4eeee",  
 troughcolor='grey',  
 length=200,  
 relief="groove",  
 from\_=0.6,  
 to=0.9,  
 width=5,  
 takefocus=0,  
 # variable=a,  
 command=get\_t\_T,  
 activebackground="white",  
 bd=1.3,  
 resolution=0.001).place(x=15, y=258)  
  
 entry(T\_frame)  
  
 T\_frame.pack\_propagate(False)  
 T\_frame.configure(width=230, height=800)  
  
def hide\_indicators():  
 P\_indicate.config(bg="#c8e6e9")  
 Q\_indicate.config(bg="#c8e6e9")  
 R\_indicate.config(bg="#c8e6e9")  
 S\_indicate.config(bg="#c8e6e9")  
 ST\_indicate.config(bg="#c8e6e9")  
 T\_indicate.config(bg="#c8e6e9")  
  
def indicate(lb, page):  
 hide\_indicators()  
 lb.config(bg = "#518281")  
 page()  
zub\_choise\_frame = tk.Frame(win, bg = "#c8e6e9")  
  
  
P\_button = tk.Button(zub\_choise\_frame, text="P", font=('bold', 14), bd=1, fg="#518281", command=lambda: indicate(P\_indicate, P\_page))  
P\_button.place(x=10, y=7)  
P\_indicate = tk.Label(zub\_choise\_frame, text="", bg="#c8e6e9")  
P\_indicate.place(x=2, y=5, height=30)  
  
Q\_button = tk.Button(zub\_choise\_frame, text="Q", font=('bold', 14), bd=1, fg="#518281", width=1, command=lambda: indicate(Q\_indicate, Q\_page))  
Q\_button.place(x=10, y=47)  
Q\_indicate = tk.Label(zub\_choise\_frame, text="", bg="#c8e6e9")  
Q\_indicate.place(x=2, y=45, height=30)  
  
R\_button = tk.Button(zub\_choise\_frame, text="R", font=('bold', 14), bd=1, fg="#518281", command=lambda: indicate(R\_indicate, R\_page))  
R\_button.place(x=10, y=87)  
R\_indicate = tk.Label(zub\_choise\_frame, text="", bg="#c8e6e9")  
R\_indicate.place(x=2, y=85, height=30)  
  
S\_button = tk.Button(zub\_choise\_frame, text="S", font=('bold', 14), bd=1, fg="#518281", command=lambda: indicate(S\_indicate, S\_page))  
S\_button.place(x=10, y=127)  
S\_indicate = tk.Label(zub\_choise\_frame, text="", bg="#c8e6e9")  
S\_indicate.place(x=2, y=125, height=30)  
  
ST\_button = tk.Button(zub\_choise\_frame, text="ST", font=('bold', 14), bd=1, fg="#518281", width=1, command=lambda: indicate(ST\_indicate, ST\_page))  
ST\_button.place(x=10, y=167)  
ST\_indicate = tk.Label(zub\_choise\_frame, text="", bg="#c8e6e9")  
ST\_indicate.place(x=2, y=165, height=30)  
  
T\_button = tk.Button(zub\_choise\_frame, text="T", font=('bold', 14), bd=1, fg="#518281", command=lambda: indicate(T\_indicate, T\_page))  
T\_button.place(x=10, y=207)  
T\_indicate = tk.Label(zub\_choise\_frame, text="", bg="#c8e6e9")  
T\_indicate.place(x=2, y=205, height=30)  
  
  
  
zub\_choise\_frame.pack(side=tk.LEFT)  
zub\_choise\_frame.pack\_propagate(False)  
zub\_choise\_frame.configure(width = 60, height = 800)  
  
  
main\_frame = tk.Frame(win, highlightbackground='grey', highlightthickness=1)  
  
def reset():  
 main(param\_static, int(hr.get()))  
  
T\_button = tk.Button(main\_frame, text="Reset", font=('bold', 16), bd=1, fg="#518281", command=reset)  
T\_button.place(x=743, y=540)  
  
  
def generate():  
 win\_2 = tk.Tk()  
 win\_2.title("Noise generation")  
 win\_2.geometry("1000x450+300+100")  
 win\_2.resizable(False, False)  
  
 figu, axe = plt.subplots()  
  
 menu\_frame = tk.Frame(win\_2, bg="#c8e6e9")  
 menu\_frame.pack(side=tk.BOTTOM)  
 menu\_frame.pack\_propagate(False)  
 menu\_frame.configure(width=1000, height=75)  
  
 # def get\_cycle\_num(val):  
 # cycle\_num = val  
 # return cycle\_num  
  
 cycle\_num = tk.StringVar(win\_2)  
 cycle\_num.set('8')  
  
 def get\_cycle():  
 global Cycle  
 Cycle = int(cycle\_number.get())  
 try:  
 main\_2(param\_changeble, int(hr.get()), Cycle, alter)  
 except:  
 main\_2(param\_changeble, int(hr.get()), Cycle, 0)  
  
  
 cycle\_number = tk.Spinbox(menu\_frame,  
 bg='#F1F2EA',  
 fg='#444444',  
 justify='center',  
 width=5,  
 from\_=2,  
 to = 30,  
 increment=1,  
 textvariable=cycle\_num,  
 command=get\_cycle)  
 cycle\_number.place(x=65, y=29)  
  
  
 def main\_2(params, hr, cycle, alt):  
 Fs = hr \* 1000  
 t0 = 60 \* 1000 / Fs  
  
 def find\_t(a):  
 t1 = params[a][3] - 3 \* params[a][1]  
 t2 = params[a][3] + 3 \* params[a][2]  
 return t1, t2  
  
 def find\_b(a, t):  
 if t <= params[a][3]:  
 return params[a][1]  
 else:  
 return params[a][2]  
  
 def Gauss(a, time):  
 return params[a][0] \* np.exp(-((time - params[a][3]) \*\* 2) / ((2 \* find\_b(a, time) \*\* 2)))  
  
 def Gauss\_T(a, time):  
 return params[a][0] \* (1 + (alt / params[a][0])) \* np.exp(  
 -((time - params[a][3]) \*\* 2) / ((2 \* find\_b(a, time) \*\* 2)))  
  
 def x\_arr(change):  
 x = []  
 for i in t:  
 if (find\_t('P')[0] <= i < find\_t('P')[1]):  
 x.append(Gauss("P", i))  
 elif (i >= find\_t("Q")[0] and i < find\_t("Q")[1]):  
 x.append(Gauss("Q", i))  
 elif (i >= find\_t("R")[0] and i < find\_t("R")[1]):  
 x.append(Gauss("R", i))  
 elif (i >= find\_t("S")[0] and i < find\_t("S")[1]):  
 x.append(Gauss("S", i))  
 elif (i >= find\_t("ST")[0] and i < find\_t("ST")[1]):  
 x.append(Gauss("ST", i))  
 elif (i >= find\_t("T")[0] and i < find\_t("T")[1]):  
 if change == 0:  
 x.append(Gauss("T", i))  
 if change == 1:  
 x.append(Gauss\_T("T", i))  
 else:  
 x.append(0)  
 return x  
  
 t = np.arange(0, t0, 0.001)  
  
 global all\_arrs  
 all\_arrs = []  
  
 for i in range(0, cycle):  
 if i % 2 == 0:  
 all\_arrs += x\_arr(0)  
 else:  
 all\_arrs += x\_arr(1)  
  
 t\_generate = np.arange(0, t0\*(cycle), 0.001)  
  
 def generate\_plot():  
 axe.clear()  
 axe.plot(t\_generate, all\_arrs, color="#444444", linewidth=1)  
 axe.set\_ylim([-0.6, 1.9])  
 axe.grid(True, linewidth=0.9, color="#c8e6e9")  
 # axe.set\_xlabel("c", loc='right')  
 # axe.set\_ylabel("мВ", loc='top', rotation=0, labelpad=0.1)  
 figu.patch.set\_facecolor('#F0F0E2')  
 canvas.draw()  
  
 canvas = FigureCanvasTkAgg(figu, master=win\_2)  
 canvas.get\_tk\_widget().place(x=-50, y=-5, width=1100, height=380)  
 generate\_plot()  
  
  
 def get\_alt(val):  
 global alter  
 alter = float(val)  
 main\_2(param\_changeble, int(hr.get()), Cycle, alter)  
  
 tk.Label(menu\_frame, text="Alternation level, mV",  
 bg="#c8e6e9",  
 font=("Arial", 13),  
 width=20,  
 height=1,  
 bd=0,  
 # cursor="sizing",  
 relief=tk.RAISED).place(x=220, y=8)  
 tk.Label(menu\_frame, text="Noise level",  
 bg="#c8e6e9",  
 font=("Arial", 13),  
 width=20,  
 height=1,  
 bd=0,  
 # cursor="sizing",  
 relief=tk.RAISED).place(x=750, y=8)  
 tk.Label(menu\_frame,  
 text="Number of cycles",  
 bg="#c8e6e9",  
 font=("Arial", 13),  
 width=20,  
 height=1,  
 bd=0,  
 # cursor="sizing",  
 relief=tk.RAISED).place(x=20, y=8)  
  
 tk.Scale(menu\_frame,  
 orient="horizontal",  
 bg="#c8e6e9",  
 troughcolor='#645D5D',  
 length=200,  
 relief="groove",  
 from\_=-3,  
 to=3,  
 width=6,  
 takefocus=0,  
 activebackground="#c8e6e9",  
 bd=0,  
 command=get\_alt,  
 resolution=0.01).place(x=200, y=25)  
  
 tk.Scale(menu\_frame,  
 orient="horizontal",  
 bg="#c8e6e9",  
 troughcolor='#645D5D',  
 length=200,  
 relief="groove",  
 from\_=-5,  
 to=5,  
 width=6,  
 takefocus=0,  
 activebackground="#c8e6e9",  
 bd=0,  
 resolution=0.1).place(x=730, y=25)  
  
  
  
 main\_2(param\_changeble, 60, 8, 0)  
  
 win\_2.mainloop()  
  
  
Generate\_button = tk.Button(main\_frame, text="Generate", font=('bold', 16), bd=1, fg="#518281", command=generate)  
Generate\_button.place(x=625, y=540)  
  
main(param\_static, 60)  
  
main\_frame.pack(side=tk.LEFT)  
main\_frame.pack\_propagate(False)  
main\_frame.configure(height=600, width=970, bg="#F0F0E2")  
  
win.mainloop()

**Результат**

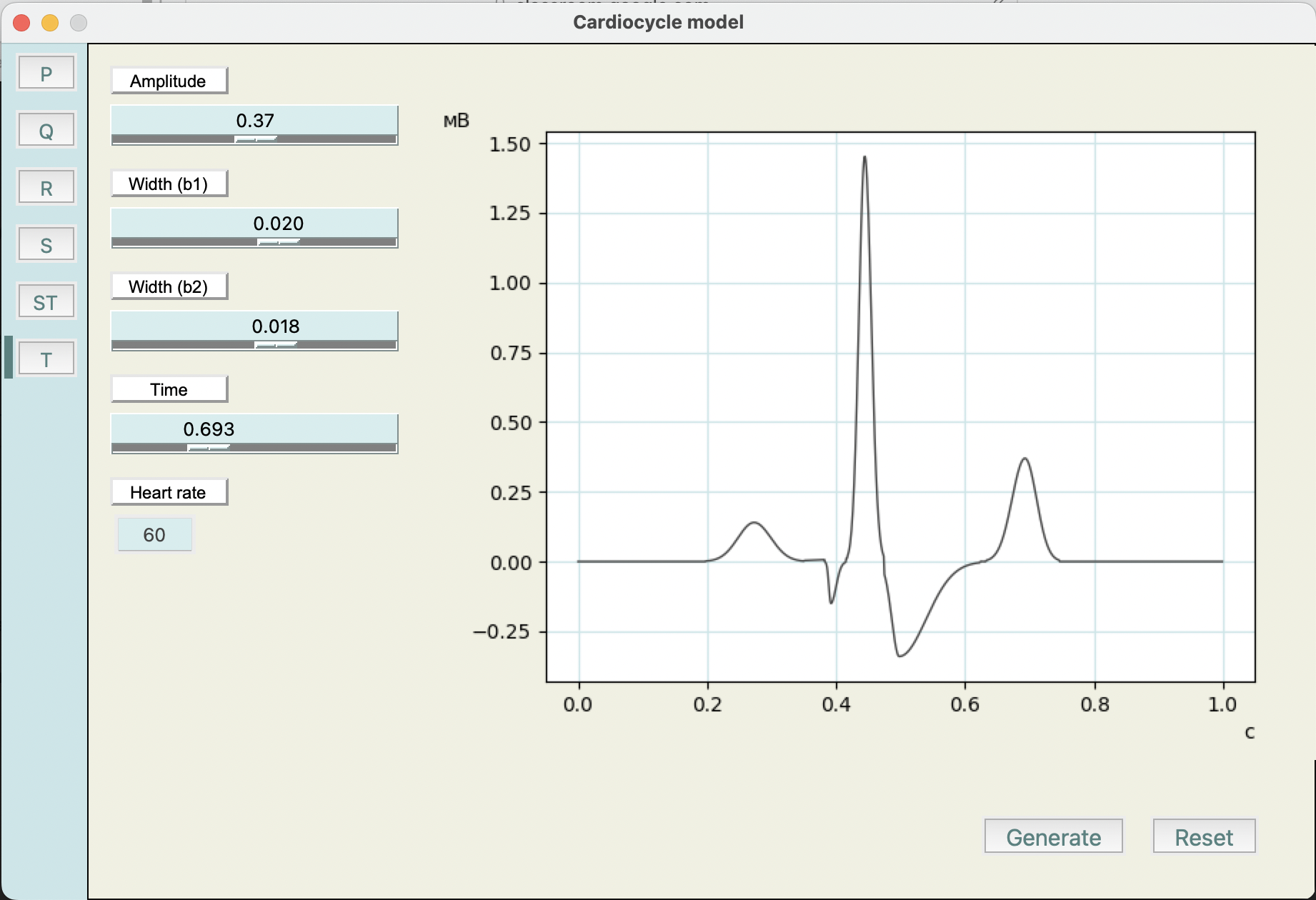
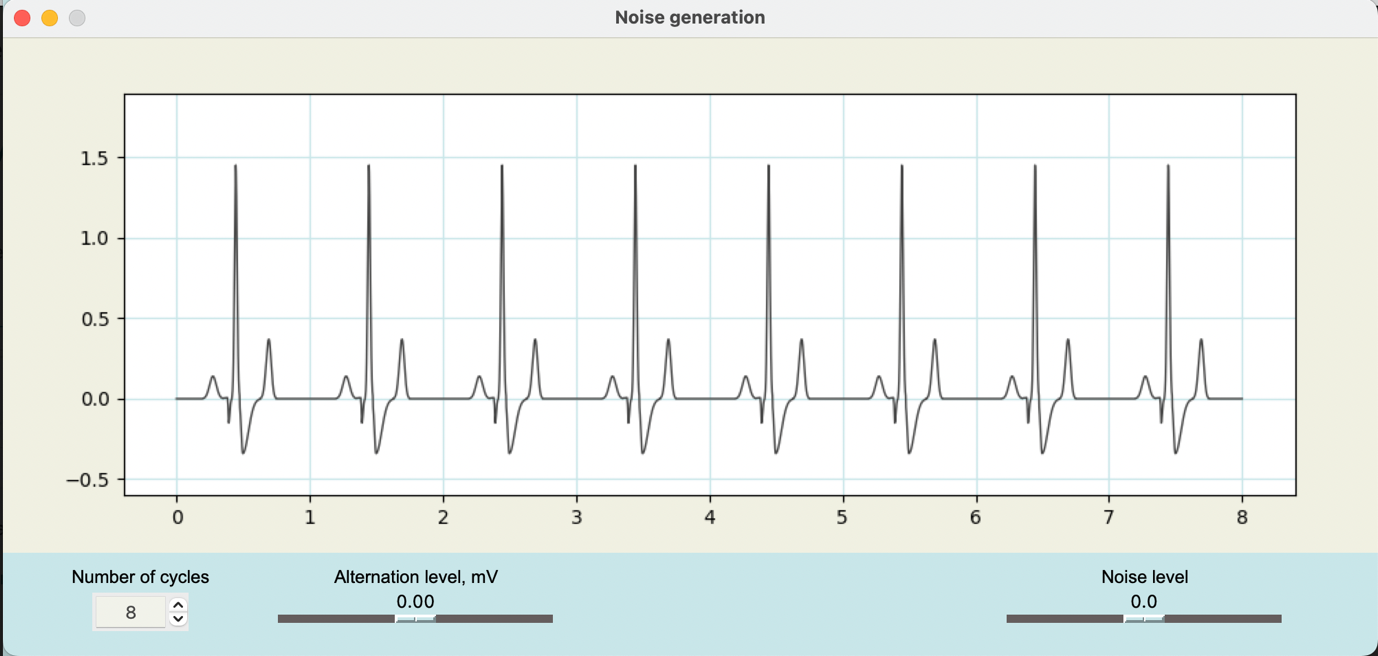
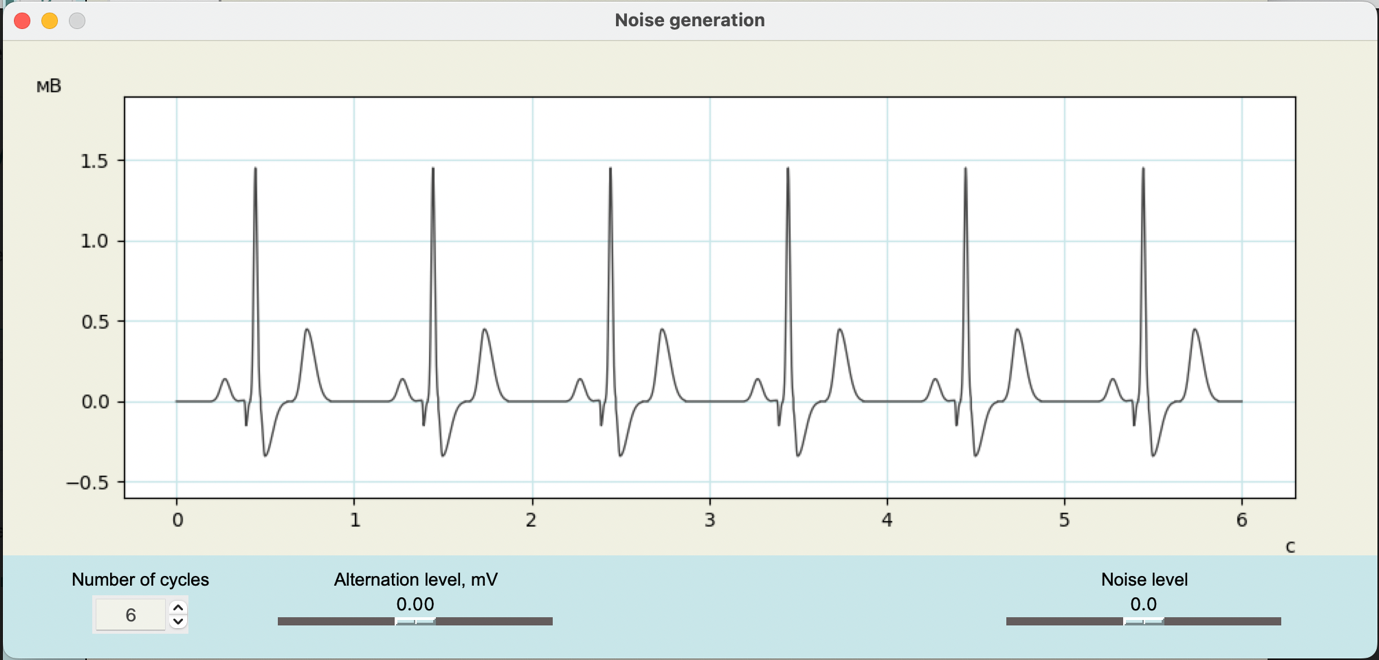
Рис. 1. Кнопка GENERATE

Рис. 2. Вікно, що відкривається після натискання на кнопку



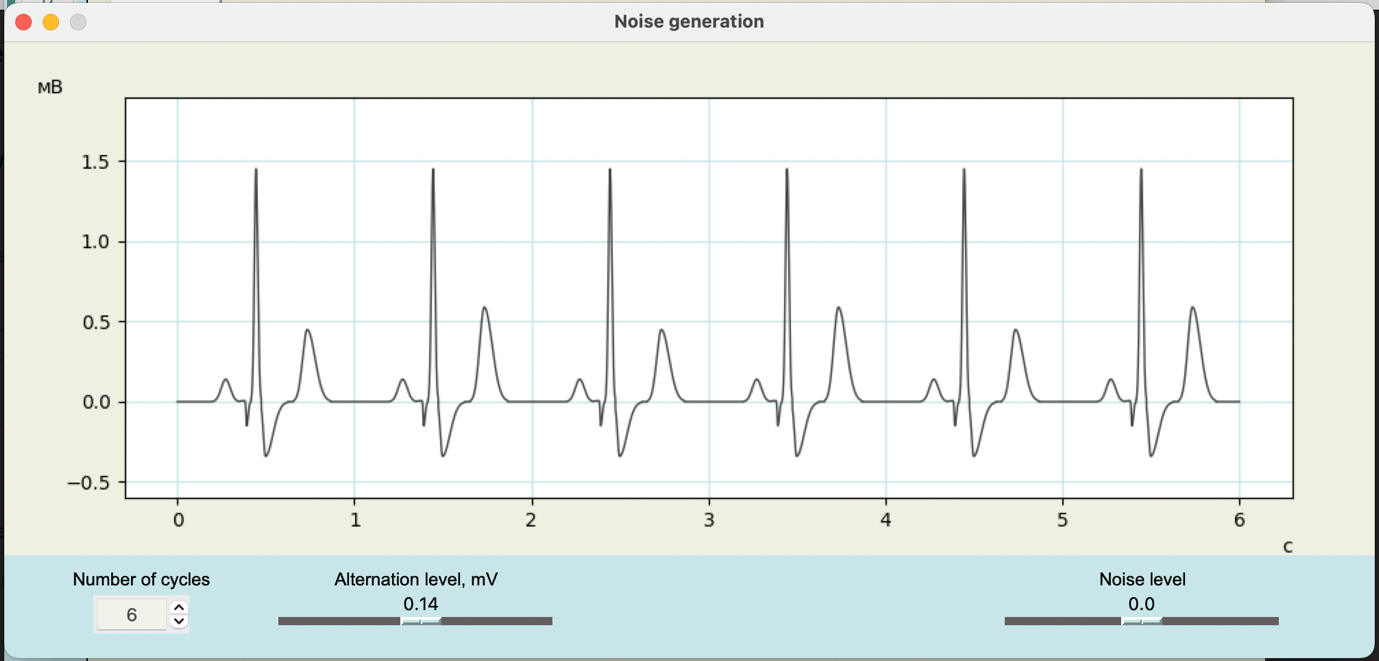
Рис. 3. Можливість змінити кількість циклів

Рис. 4. Альтерація зубця Т

Висновок: у даній лабораторній роботі було виконано можливість генерації різної кількості циклів ЕКГ, та зміни амплітуди зубця Т. Завдання виконано мовою програмування Python, з використанням бібліотеки Tkinter для віконних додатків.