 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №3**

з дисципліни «Методи штучного інтелекту»

на тему: «ПОБУДОВА ЦИКЛУ ШТУЧНОЇ ЕЛЕКТРОКАРДІОГРАМИ»

**Виконала:**

студентка гр. БС-24

Сапон О. С.

**Перевірив:**

проф. Файнзільберг Л. С.

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

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

(підпис викладача)

Київ-2024

**Мета роботи**: програмно реалізувати процедуру побудови циклу штучної електрокардіограми (ЕКГ).

**Основні задачі роботи:**

1. Освоїти технологію моделювання штучних сигналів складної форми.

2. Визначити значення параметрів моделі циклу штучної ЕКГ.

3. Програмно реалізувати модель побудови циклу штучної ЕКГ

4. Оцінити можливі обмеження форм реалістичного циклу штучної ЕКГ.

**Завдання до комп’ютерного практикуму**

1. Для спрощення виконання поставленої задачі припустимо, що всі зубці ЕКГ симетричні за виключенням зубця Т, тобто:

2. Беручи до уваги властивості гаусівської функції будемо вважати, що моменти початку та завершення кожного і-го фрагменту, i ∈ {P,Q,R,S,ST,T} ,пов’язані з параметрами , та наступним чином

3. Для забезпечення реалістичності форми циклу штучної ЕКГ мають бути дотримані наступні обмеження:

Де t0 – загальна тривалість циклу (мс), яка пов’язана з частотою FH серцевих скорочень (уд/хв) наступним співвідношенням:

4. Вихідними даними для реалізації програми мають бути параметри моделі (1).

5. Користувачем в діалоговому вікні задаються такі параметри:

- частота серцевих скорочень FH;

- амплітуда AT зубця T;

- момент часу, коли зубець зубець T приймає ектремальне значенния;

- параметри та , що визначають симетрію зубця T , а інші параметри моделі задаються константами в тексті програми

6. Результатом роботи програми мають бути графіки циклів штучної ЕКГ, що реалізують такі варіанти форми: нормальний цикл, негативний Т, асиметричний Т.

Хід роботи:

Програму було реалізовано мовою програмування Python з використанням бібліотек PyQT та PySide6 для реалізації інтерфейсу. Лістинг наведено в додатку.

Результати:

Перш за все користувача проситься ввести частоту серцевих скорочень, оскільки від цього залежатиме тривалість кардіоцикла. В подальшому це значення можна змінювати і спостерігати за змінами графіка. При збільшенні ЧСС цикл буде коротшим, а при зменшенні – більш тривалим. Після введення будується цикл ЕКГ з параметрами, заданими в коді за замовчуванням (рис 1).

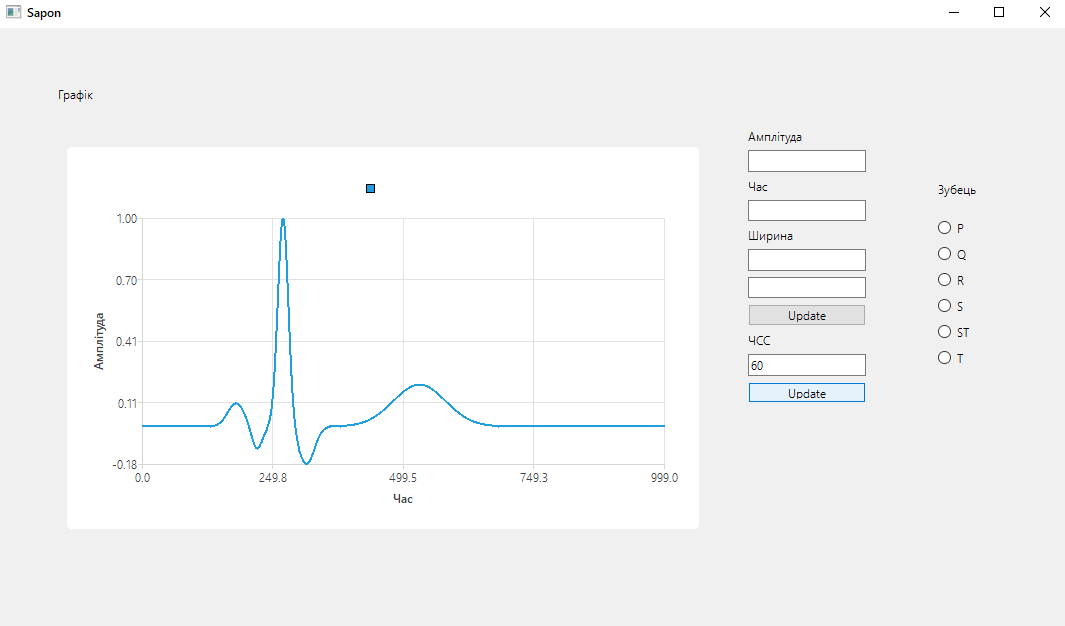


Рисунок 1. Вікно програми (нормальний цикл)

Тепер можна, наприклад, зробити зубець Т негативним і симетричним. Для цього у вікні обираємо зубець, який хочемо змінити і вводимо бажані параметри (рис 2).

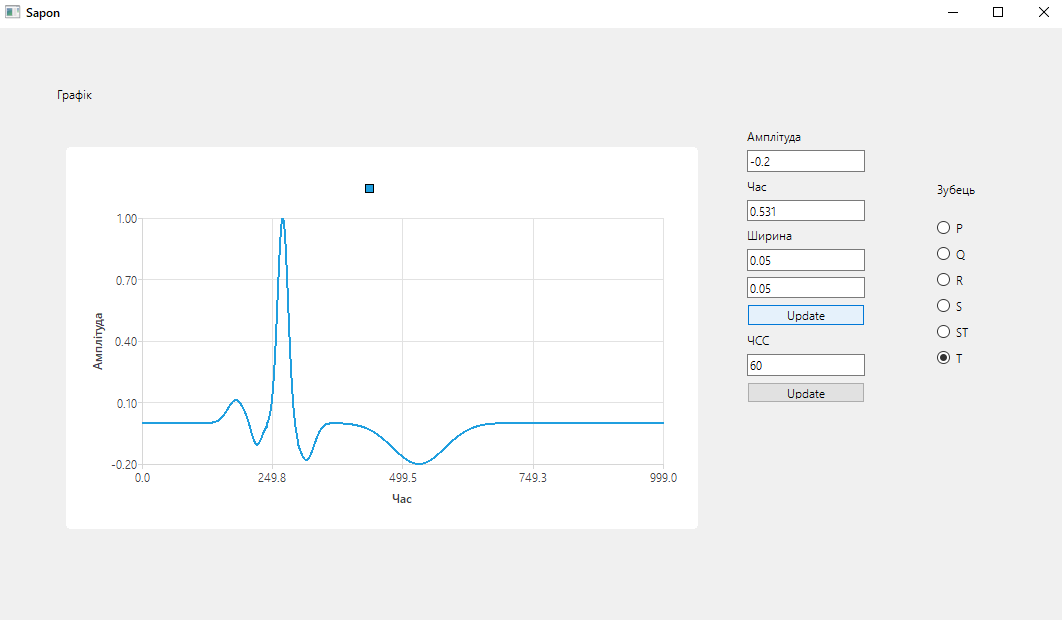


Рисунок 2. Негативний зубець Т

Або можемо зробити зубець Т позитивним і асиметричним (рис 3).

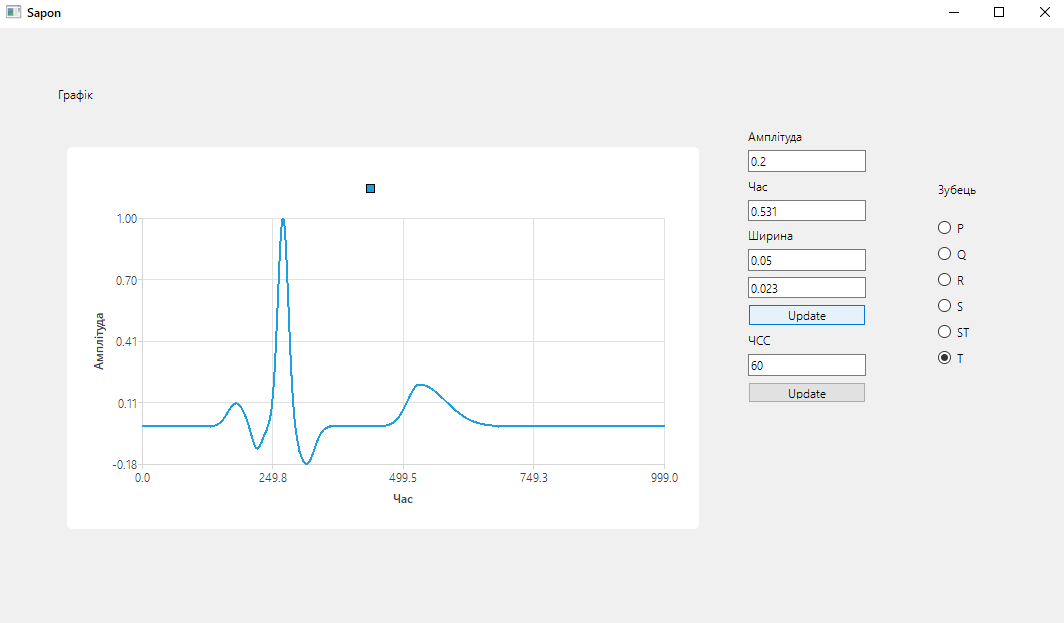


Рисунок 3. Асиметричний зубець Т

Висновки: в ході виконання комп’ютерного практикуму було розроблено програму, що моделює сигнал ЕКГ у вигляді суми гаусових функцій. Програма коректно реалізує поставлені задачі, але має потенціал до вдосконалення, наприклад шляхом підвищення інтерактивності під час роботи.

**Додаток**

Лістинг програми (інтерфейс)

# -\*- coding: utf-8 -\*-  
  
################################################################################  
## Form generated from reading UI file 'cardio\_cycle\_updated.ui'  
##  
## Created by: Qt User Interface Compiler version 6.6.2  
##  
## WARNING! All changes made in this file will be lost when recompiling UI file!  
################################################################################  
  
from PySide6.QtCharts import QChartView  
from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,  
 QMetaObject, QObject, QPoint, QRect,  
 QSize, QTime, QUrl, Qt)  
from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,  
 QFont, QFontDatabase, QGradient, QIcon,  
 QImage, QKeySequence, QLinearGradient, QPainter,  
 QPalette, QPixmap, QRadialGradient, QTransform)  
from PySide6.QtWidgets import (QApplication, QFormLayout, QLabel, QLineEdit,  
 QMainWindow, QPushButton, QRadioButton, QSizePolicy,  
 QVBoxLayout, QWidget)  
  
class Ui\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 if not MainWindow.objectName():  
 MainWindow.setObjectName(u"MainWindow")  
 MainWindow.resize(1070, 600)  
 self.centralwidget = QWidget(MainWindow)  
 self.centralwidget.setObjectName(u"centralwidget")  
 self.label\_5 = QLabel(self.centralwidget)  
 self.label\_5.setObjectName(u"label\_5")  
 self.label\_5.setGeometry(QRect(940, 140, 61, 41))  
 self.graphicsView = QChartView(self.centralwidget)  
 self.graphicsView.setObjectName(u"graphicsView")  
 self.graphicsView.setGeometry(QRect(60, 110, 650, 400))  
 self.label = QLabel(self.centralwidget)  
 self.label.setObjectName(u"label")  
 self.label.setGeometry(QRect(60, 50, 51, 31))  
 self.widget = QWidget(self.centralwidget)  
 self.widget.setObjectName(u"widget")  
 self.widget.setGeometry(QRect(940, 190, 43, 152))  
 self.formLayout = QFormLayout(self.widget)  
 self.formLayout.setObjectName(u"formLayout")  
 self.formLayout.setContentsMargins(0, 0, 0, 0)  
 self.radioButton = QRadioButton(self.widget)  
 self.radioButton.setObjectName(u"radioButton")  
  
 self.formLayout.setWidget(0, QFormLayout.LabelRole, self.radioButton)  
  
 self.radioButton\_3 = QRadioButton(self.widget)  
 self.radioButton\_3.setObjectName(u"radioButton\_3")  
  
 self.formLayout.setWidget(1, QFormLayout.LabelRole, self.radioButton\_3)  
  
 self.radioButton\_5 = QRadioButton(self.widget)  
 self.radioButton\_5.setObjectName(u"radioButton\_5")  
  
 self.formLayout.setWidget(2, QFormLayout.LabelRole, self.radioButton\_5)  
  
 self.radioButton\_4 = QRadioButton(self.widget)  
 self.radioButton\_4.setObjectName(u"radioButton\_4")  
  
 self.formLayout.setWidget(3, QFormLayout.LabelRole, self.radioButton\_4)  
  
 self.radioButton\_2 = QRadioButton(self.widget)  
 self.radioButton\_2.setObjectName(u"radioButton\_2")  
  
 self.formLayout.setWidget(4, QFormLayout.LabelRole, self.radioButton\_2)  
  
 self.radioButton\_6 = QRadioButton(self.widget)  
 self.radioButton\_6.setObjectName(u"radioButton\_6")  
  
 self.formLayout.setWidget(5, QFormLayout.LabelRole, self.radioButton\_6)  
  
 self.widget1 = QWidget(self.centralwidget)  
 self.widget1.setObjectName(u"widget1")  
 self.widget1.setGeometry(QRect(750, 100, 118, 276))  
 self.verticalLayout = QVBoxLayout(self.widget1)  
 self.verticalLayout.setObjectName(u"verticalLayout")  
 self.verticalLayout.setContentsMargins(0, 0, 0, 0)  
 self.label\_2 = QLabel(self.widget1)  
 self.label\_2.setObjectName(u"label\_2")  
  
 self.verticalLayout.addWidget(self.label\_2)  
  
 self.lineEdit = QLineEdit(self.widget1)  
 self.lineEdit.setObjectName(u"lineEdit")  
  
 self.verticalLayout.addWidget(self.lineEdit)  
  
 self.label\_4 = QLabel(self.widget1)  
 self.label\_4.setObjectName(u"label\_4")  
  
 self.verticalLayout.addWidget(self.label\_4)  
  
 self.lineEdit\_2 = QLineEdit(self.widget1)  
 self.lineEdit\_2.setObjectName(u"lineEdit\_2")  
  
 self.verticalLayout.addWidget(self.lineEdit\_2)  
  
 self.label\_3 = QLabel(self.widget1)  
 self.label\_3.setObjectName(u"label\_3")  
  
 self.verticalLayout.addWidget(self.label\_3)  
  
 self.lineEdit\_3 = QLineEdit(self.widget1)  
 self.lineEdit\_3.setObjectName(u"lineEdit\_3")  
  
 self.verticalLayout.addWidget(self.lineEdit\_3)  
  
 self.lineEdit\_4 = QLineEdit(self.widget1)  
 self.lineEdit\_4.setObjectName(u"lineEdit\_4")  
  
 self.verticalLayout.addWidget(self.lineEdit\_4)  
  
 self.pushButton = QPushButton(self.widget1)  
 self.pushButton.setObjectName(u"pushButton")  
  
 self.verticalLayout.addWidget(self.pushButton)  
  
 self.verticalLayout.addLayout(self.verticalLayout)  
  
 self.label\_6 = QLabel(self.widget1)  
 self.label\_6.setObjectName(u"label\_6")  
 self.verticalLayout.addWidget(self.label\_6)  
  
 self.lineEdit\_5 = QLineEdit(self.widget1)  
 self.lineEdit\_5.setObjectName(u"lineEdit\_5")  
 self.verticalLayout.addWidget(self.lineEdit\_5)  
 self.lineEdit\_5.setGeometry(QRect(100, 350, 50, 27))  
  
 self.pushButton\_1 = QPushButton(self.widget1)  
 self.pushButton\_1.setObjectName(u"pushButton\_1")  
  
 self.verticalLayout.addWidget(self.pushButton\_1)  
  
 MainWindow.setCentralWidget(self.centralwidget)  
  
 self.retranslateUi(MainWindow)  
  
 QMetaObject.connectSlotsByName(MainWindow)  
 # setupUi  
  
 def retranslateUi(self, MainWindow):  
 MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow", u"Sapon", None))  
 self.label\_5.setText(QCoreApplication.translate("MainWindow", u"Зубець", None))  
 self.label.setText(QCoreApplication.translate("MainWindow", u"Графік", None))  
 self.radioButton.setText(QCoreApplication.translate("MainWindow", u"P", None))  
 self.radioButton\_3.setText(QCoreApplication.translate("MainWindow", u"Q", None))  
 self.radioButton\_5.setText(QCoreApplication.translate("MainWindow", u"R", None))  
 self.radioButton\_4.setText(QCoreApplication.translate("MainWindow", u"S", None))  
 self.radioButton\_2.setText(QCoreApplication.translate("MainWindow", u"ST", None))  
 self.radioButton\_6.setText(QCoreApplication.translate("MainWindow", u"T", None))  
 self.label\_2.setText(QCoreApplication.translate("MainWindow", u"Амплітуда", None))  
 self.label\_4.setText(QCoreApplication.translate("MainWindow", u"Час", None))  
 self.label\_3.setText(QCoreApplication.translate("MainWindow", u"Ширина", None))  
 self.pushButton.setText(QCoreApplication.translate("MainWindow", u"Update", None))  
 self.pushButton\_1.setText(QCoreApplication.translate("MainWindow", u"Update", None))  
 self.label\_6.setText(QCoreApplication.translate("MainWindow", u"ЧСС", None))  
 # retranslateUi

Лістинг програми (основна програма)

import sys  
from PySide6.QtWidgets import QApplication, QMainWindow, QGraphicsView  
from cardio\_cycle\_updated import Ui\_MainWindow  
from PySide6 import QtCharts  
from PySide6 import QtCore  
import numpy as np  
import matplotlib.pyplot  
  
def gauss\_function(t, A, mu, sigma):  
 return A \* np.exp(-((t - mu) \*\* 2) / (2 \* sigma \*\* 2))  
  
def generate\_ECG\_component(t, A, mu, sigma\_function, duration):  
 signal = np.zeros\_like(t)  
 idx = (t >= mu - duration \* sigma\_function(mu)) & (t <= mu + duration \* sigma\_function(mu))  
 signal[idx] = gauss\_function(t[idx], A, mu, sigma\_function(t[idx]))  
 # print("Generated component:", signal)  
 return signal  
  
def sigma\_function(t, mu, sigma\_max, sigma\_min):  
 return np.where(t < mu, sigma\_min, sigma\_max)  
  
class CardioCucle(QMainWindow):  
 def \_\_init\_\_(self):  
 super(CardioCucle, self).\_\_init\_\_()  
 self.ui = Ui\_MainWindow()  
 self.ui.setupUi(self)  
  
 # Параметры гауссовых функций для генерации компонентов ЭКГ  
 self.parameters = {  
 'P': {'A': 0.11, 'mu': 0.18, 'sigma\_max': 0.016, 'sigma\_min': 0.016, 'duration': 3},  
 'Q': {'A': -0.11, 'mu': 0.22, 'sigma\_max': 0.01, 'sigma\_min': 0.01, 'duration': 3},  
 'R': {'A': 1.0, 'mu': 0.27, 'sigma\_max': 0.01, 'sigma\_min': 0.01, 'duration': 3},  
 'S': {'A': -0.18, 'mu': 0.315, 'sigma\_max': 0.015, 'sigma\_min': 0.015, 'duration': 3},  
 'ST': {'A': 0, 'mu': 0.38, 'sigma\_max': 0.04, 'sigma\_min': 0.04, 'duration': 3},  
 'T': {'A': 0.2, 'mu': 0.531, 'sigma\_max': 0.05, 'sigma\_min': 0.05, 'duration': 3}  
 }  
  
 self.ui.pushButton.clicked.connect(self.update\_parameters)  
 self.ui.pushButton\_1.clicked.connect(self.graph)  
  
 self.graph()  
  
 def update\_parameters(self):  
 amplitude\_str = self.ui.lineEdit.text()  
 time\_str = self.ui.lineEdit\_2.text()  
 width1\_str = self.ui.lineEdit\_3.text()  
 width2\_str = self.ui.lineEdit\_4.text()  
  
 selected\_key = self.get\_selected\_radio\_button()  
  
 if amplitude\_str and time\_str and width1\_str and width2\_str and selected\_key:  
 amplitude = float(amplitude\_str)  
 time = float(time\_str)  
 width1 = float(width1\_str)  
 width2 = float(width2\_str)  
  
 self.parameters[selected\_key]['A'] = amplitude  
 self.parameters[selected\_key]['mu'] = time  
 self.parameters[selected\_key]['sigma\_max'] = width1  
 self.parameters[selected\_key]['sigma\_min'] = width2  
  
 self.graph()  
  
 def get\_selected\_radio\_button(self):  
 if self.ui.radioButton.isChecked():  
 return 'P'  
 elif self.ui.radioButton\_3.isChecked():  
 return 'Q'  
 elif self.ui.radioButton\_5.isChecked():  
 return 'R'  
 elif self.ui.radioButton\_4.isChecked():  
 return 'S'  
 elif self.ui.radioButton\_2.isChecked():  
 return 'ST'  
 elif self.ui.radioButton\_6.isChecked():  
 return 'T'  
 else:  
 return None  
  
 def graph(self):  
 # Обновляем график ЭКГ  
 heart\_rate\_str = self.ui.lineEdit\_5.text()  
 if heart\_rate\_str:  
 heart\_rate = float(heart\_rate\_str)  
 rate = 60 / heart\_rate  
 t = np.linspace(0, 1, 1000)  
 ECG\_components = {}  
 for component, params in self.parameters.items():  
 ECG\_components[component] = generate\_ECG\_component(t/rate, params['A'], params['mu'],  
 lambda t: sigma\_function(t, params['mu'],  
 params['sigma\_max'],  
 params['sigma\_min']),  
 params['duration'])  
  
 ECG\_signal = sum(ECG\_components.values())  
  
 series = QtCharts.QLineSeries()  
 for i in range(len(t)):  
 ecg\_value = sum(ECG\_components[component][i] for component in self.parameters.keys())  
 series.append(i, ecg\_value) # Добавляем значения времени t[i] и сигнала ECG\_signal[i]  
  
 chart = QtCharts.QChart()  
 chart.addSeries(series)  
  
 chart.createDefaultAxes()  
 chart.axisX().setGridLineVisible(True)  
 chart.axisY().setGridLineVisible(True)  
  
 # Подписываем оси  
 chart.axisX().setTitleText("Час")  
 chart.axisY().setTitleText("Амплітуда")  
  
 self.ui.graphicsView.setChart(chart)  
  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 window = CardioCucle()  
 window.show()  
  
 sys.exit(app.exec())