НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ННІ ЕНЕРГЕТИКИ, АВТОМАТИКИ І ЕНЕРГОЗБЕРЕЖЕННЯ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
В.о. завідувача кафедри
автоматики та робототехнічних систем
ім. акад. І.І. Мартиненка
(назва кафедри)

к.т.н., доц. _____ О.О. Опришко
(підпис) (ПІБ)

"____" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему "БІОМЕДИЧНА СИСТЕМА ВІЗУАЛІЗАЦІЇ ДИНАМІКИ ЗМІН СЕРЦЕВОГО РИТМУ"

Спеціальність: 163 - "Біомедична інженерія"

Гарант освітньої програми

д.т.н., професор _____ Никифорова Л.Є.
(науковий ступінь та вчене звання) (підпис) (П.І.Б.)

Керівник бакалаврської кваліфікаційної роботи

д.т.н., доцент _____ Никифорова Л.Є.
(науковий ступінь та вчене звання) (підпис) (П.І.Б.)

Виконав _____ Наумчик М.Є.
(підпис) (П.І.Б.)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ННІ ЕНЕРГЕТИКИ, АВТОМАТИКИ І ЕНЕРГОЗБЕРЕЖЕННЯ

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
автоматики та робототехнічних систем
ім. акад. І.І. Мартиненка
(назва кафедри)
к.т.н., доц. О.О. Опришко
(підпис) (ПІБ)
"____" _____ 2025 р.

ЗАВДАННЯ
на виконання бакалаврської кваліфікаційної роботи студентці
Наумчика Максима Євгенійовича
(прізвище, ім'я, по батькові)

Спеціальність: 163 - "Біомедична інженерія"
1. Тема бакалаврської кваліфікаційної роботи: "Біомедична система візуалізації динаміки змін серцевого ритму ",
затверджена наказом ректора НУБіП України від "12" 11 2024 р. № 2023 "С"
2. Термін подання завершеної роботи на кафедру "31" травня 2025 р.
3. Вихідні дані до бакалаврської кваліфікаційної роботи:
   3.1. Завдання кафедри на виконання бакалаврської кваліфікаційної роботи.
   3.2. Нормативні документи по проектуванню біомедичних систем
   3.3. Наукова література з тематики бакалаврської кваліфікаційної роботи.
4. Перелік питань, які необхідно розробити:
   4.1. Аналіз стану медичної проблеми спостереження за серцевим ритмом
   4.2. Дослідження електрокардіосигналу як біологічного об’єкту .
   4.3. Огляд відомих математичних моделей опрацювання кардіосигналу.
   4.4. Вибір технічних засобів для візуалізації динаміки змін серцевого ритму.
   4.5. Програмні засоби для візуалізації динаміки змін серцевого ритму .
   4.5. Схеми біомедичної системи (структурна, функціональна, алгоритмічна).
   4.6. Кошторисні розрахунки.
   4.7. Техніка безпеки і охорона праці.

5. Перелік графічних документів:
5.1. Структурна і функціональна схеми біомедичної системи візуалізації динаміки змін серцевого ритму.
5.2. Алгоритм візуалізації динаміки змін серцевого ритму .
.

Дата видачі завдання "19" грудня 2024 року

Керівник бакалаврської кваліфікаційної роботи _____ Никифорова Л.Є.
(підпис) (П.І.Б.)

Завдання прийняв до виконання _____ Наумчик М.Є.
(підпис) (П.І.Б.)

РЕФЕРАТ

Дипломна робота за темою «Біомедична система візуалізації динаміки змін серцевого ритму» виконана студенткою кафедри автоматизації та робототехнічних систем ННІ енергетики, автоматики та енергозбереження *Наумчиком Максимом Євгенійовичем* зі спеціальності *163 «Біомедична інженерія»* та складається зі: вступу; 3 розділів (медико-технічне обґрунтування проекту; структурна схема візуалізації динаміки змін серцевого ритму; розробка алгоритмічного, програмного та технічного забезпечення), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 20 джерел. Кількість таблиць – 12, ілюстрація - 18. Загальний обсяг роботи 101 сторінка.

Метою роботи є покращення методів для візуалізації динаміки змін серцевого ритму, які базуються на аналізі фізіологічних даних, створенні та налагодження систем автоматизованого аналізу ЕКГ, розробці програмного додатку. У рамках дослідження будуть розглянуті існуючі методи здійснення візуалізації сигналів ЕКГ, проведено аналіз їх ефективності та обґрунтовано вибір оптимального підходу до розробки програми.

Об'єктом дослідження є медична задача дослідження серцевого ритму людини для діагностування та профілактики сердцево-судинних захворювань.

Предметом дослідження є комплекс програмних та технічних автоматизованої візуалізації даних електрокардіограми. Програмний додаток дозволяє переглядати сигнали кардіограми, зберігати потрібні записи при підключенні до мережі Інтернет. Для створення програмного додатку використовується мова програмування Python. У роботі є короткий огляд методів електрокардіографії, методів візуалізації, технічних засобів.

Ключові слова: *серцево-судинні показники; електрокардіограма; візуалізація; алгоритм; програмний додаток; Python*

ABSTRACT

The thesis on the topic " Biomedical system for visualizing the dynamics of heart rate changes " was completed by a student of the Department of Automation and Robotic Systems of the Educational and Scientific Institute of Power Engineering, Automation and Energy Saving, *Maksym Naumchyk*, majoring in *163 "Biomedical Engineering"* and consists of: introduction; 3 sections (medical and technical justification of the project; structural diagram of visualization of the dynamics of heart rate changes; development of algorithmic, software and hardware support), conclusions for each of these sections; general conclusions; list of sources used, which includes 20 sources. Number of tables - 12, illustrations - 18. Total volume of work - 101 pages.

The purpose of the work is to improve methods for visualizing the dynamics of heart rate changes, which are based on the analysis of physiological data, creation and adjustment of automated ECG analysis systems, development of a software application. The research will consider existing methods for visualizing ECG signals, analyze their effectiveness, and justify the choice of the optimal approach to program development.

The object of the research is the medical task of studying the human heart rhythm for the diagnosis and prevention of cardiovascular diseases.

The subject of the research is a complex of software and technical automated visualization of electrocardiogram data. The software application allows you to view cardiogram signals and save the necessary records when connected to the Internet. The Python programming language is used to create the software application. The work provides a brief overview of electrocardiography methods, visualization methods, and technical means.

Keywords: *cardiovascular indicators; electrocardiogram; visualization; algorithm; software application; Python.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ВКР – випускна кваліфікаційна робота

ECG (ЕКГ) – electrocardiogram (електрокардіограма)

ПК – персональний комп'ютер

АЦП – аналого-цифровий перетворювач

ПЗ – програмне забезпечення

CSV – Coma Separated Values (значення, розділені комами)

EDF – European Data Format (європейський формат даних)

WF – Windows Forms

Wfdb – Waveform Database (база даних хвильових форм)

ROC – receiver operating characteristics (робочі характеристики приймача)

ЧСС (HR) – частота серцевих скорочень

АТ (BP) – артеріальний тиск

САТ (SBP) – систолічний артеріальний тиск

ДАТ (DBP) – діастолічний артеріальний тиск

ПТ (PP) – пульсовий тиск

ВСР – варіабельність серцевого ритму

ССС – серцево-судинна система

СНС – симпатична нервова система

ФВА – функціонально-вартісний аналіз

ВСТУП

Серцево-судинні захворювання залишаються однією з провідних причин смертності у світі, що зумовлює актуальність розробки нових ефективних методів моніторингу і діагностики функціонального стану серця. Одним із ключових напрямів у цьому контексті є дослідження динаміки змін серцевого ритму, яке дозволяє не лише виявляти порушення ритму, а й прогнозувати розвиток критичних станів.

Біомедичні системи візуалізації динамічних процесів серцевої діяльності відіграють важливу роль у ранній діагностиці патологій, оцінці ефективності лікування та постійному моніторингу пацієнтів. Завдяки розвитку цифрових технологій та методів обробки біомедичних сигналів стало можливим створення компактних, високоточних і доступних систем для реєстрації та аналізу електрофізіологічних даних серця. Візуалізація змін серцевого ритму в реальному часі дозволяє лікарю отримувати інтегровану картину функціонування серцево-судинної системи, що значно підвищує точність діагностичних висновків. Розробка таких систем потребує поєднання знань у галузях біомедичної інженерії, цифрової обробки сигналів, програмування та сучасних методів візуалізації даних.

Актуальність даної роботи полягає у необхідності створення інноваційної біомедичної системи, яка забезпечить якісний моніторинг динаміки серцевого ритму з можливістю його графічного представлення. Це відкриває нові перспективи для розвитку персоналізованої медицини та підвищення рівня медичного обслуговування пацієнтів із серцево-судинною патологією.

Динаміка змін серцевого ритму виконується за допомогою електрокардіографії (ЕКГ), яка є неінвазивним методом діагностики, що забезпечує отримання об'єктивної інформації про електрофізіологічну активність серця. Метод базується на реєстрації електричних потенціалів, які виникають під час серцевого циклу, з подальшою їх візуалізацією у вигляді

графічного запису на дисплеї або папері. Сигнали знімаються із поверхні тіла за допомогою електродів, розташованих на кінцівках і грудній клітці. Стандартна електрокардіограма включає 12 відведень: три стандартні (I, II, III), три підсилені однополюсні відведення від кінцівок (avR, avL, avF) і шість грудних однополюсних відведень (V1–V6).

У сучасній кардіологічній практиці широко впроваджуються цифрові технології, що значно оптимізують функціонування кардіографічних систем. Перехід від аналогових до цифрових рішень дозволив скоротити апаратні ресурси, підвищити надійність пристроїв та реалізувати розширений функціонал за рахунок використання високотехнологічних мікросхем.

Для пацієнтів із патологіями серцево-судинної системи безперервний моніторинг електричної активності серця є критично важливим для своєчасного виявлення порушень ритму або інших функціональних змін. З цією метою розробляються портативні системи домашнього моніторингу ЕКГ.

Особливе місце у створенні цифрових електрокардіографічних пристроїв займають системи автоматизованого аналізу електрокардіограм. Етап розробки таких систем передбачає тестування алгоритмів обробки даних із застосуванням великого обсягу різнорідних електрокардіографічних записів. На основі результатів тестування здійснюється валідація функціонування системи та корекція алгоритмів за необхідності. Для ефективної роботи з великими масивами даних доцільно впроваджувати автоматизовані засоби розмітки сигналів і забезпечити підтримку взаємодії з відкритими медичними базами, такими як PhysioNet. Таким чином, розробка автоматизованої системи аналізу ЕКГ має високу практичну цінність для спеціалістів, що займаються створенням програмного забезпечення для цифрової обробки електрокардіографічних даних.

Мета і завдання роботи.

Метою роботи є покращення методів для візуалізації динаміки змін серцевого ритму, які базуються на аналізі фізіологічних даних, створенні та налагодження систем автоматизованого аналізу ЕКГ, розробці програмного

додатку. У рамках дослідження будуть розглянуті існуючі методи здійснення візуалізації сигналів ЕКГ, проведено аналіз їх ефективності та обґрунтовано вибір оптимального підходу до розробки програми.

Досягнення мети передбачає вирішення наступних *завдань*:

1. Аналіз вітчизняних та зарубіжних джерел.
2. Ознайомлення з методами здійснення візуалізації сигналів ЕКГ.
3. Вивчення середовища розробки і мови програмування Python.
4. Розробка програми, що дозволяє відображати сигнал, працювати з ресурсами мережі інтернет, створювати знімки сигналів та зберігати їх
5. Реалізувати графічний інтерфейс програмного додатку.
6. Розробка схеми збору даних за допомогою електрокардіографу
7. Тестування роботи програми з сигналами.

Об'єктом дослідження є медична задача дослідження серцевого ритму людини для діагностування та профілактики сердцево-судинних захворювань.

Предметом дослідження є комплекс програмних та технічних автоматизованої візуалізації даних ЕКГ.

РОЗДІЛ 1

МЕДИКО-ТЕХНІЧНЕ ОБҐРУНТУВАННЯ ПРОЕКТУ

1.1. Аналіз сучасного стану проблеми спостереження за серцевим ритмом

Перший розроблений пристрій, відомий як "гальванометр" (1794 р.), є дуже чутливим приладом для вимірювання малих постійних електричних струмів. На відміну від звичайних мікроамперметрів, шкала гальванометра може показувати не лише силу струму, а й напругу та інші фізичні величини. Шкала може бути умовною, що дозволяє використовувати цей пристрій як нуль-індикатор, тобто просто для вказівки наявності струму, без вимірювання його значення.

У 1849 році Дюбуа-Реймон удосконалив гальванометр, додавши двопозиційний вимикач, що дозволило йому безпосередньо вимірювати силу струму. Цей пристрій отримав назву "реотом". Реотом представляє собою диск, що вільно обертається, з двома штифтами на краях - один з яких фіксований, а інший переміщується для з'єднання струму з гальванометром. При швидкому обертанні диска прилад реагує на струм, фіксуючи його активність.

У 1868 році учень Дюбуа-Реймона, Юлій Бернштейн, модифікував реотом, дозволивши змінювати інтервал між стимуляцією і відбором даних. Цей новий пристрій отримав назву "диференціальний реотом", і саме за його допомогою були перші зареєстровані електрокардіограми, які в основному отримувалися з серця жаби.

Проблема з чутливістю диференціального реотомі призвела до появи "капілярної електрометрії", винайденої Габріелем Липпманом у 1872 році. Август Уоллер став першим, хто записав електричну активність людського серця без необхідності в розтині грудної клітини, зробивши це в 1887 році. У своїй статті він вперше назвав запис "електрограмою", але вже наступного року термін змінився на "кардіограму".

Ейнтховен почав створювати свій гальванометр у 1900 році, і в 1903 році представив "струнний гальванометр". Спочатку його апарат був розроблений у Німеччині, але згодом його почали виробляти в Лондоні. Прототипи "струнних гальванометрів" використовувалися для перевірки теорій та можливості точних вимірювань серцевих ритмів. Перший повноцінний електрокардіограф був виготовлений у США в 1909 році та вдосконалений у 1914 році.

На рисунку 1.1 приведений трикутник скорочень з першими шести відведенями.

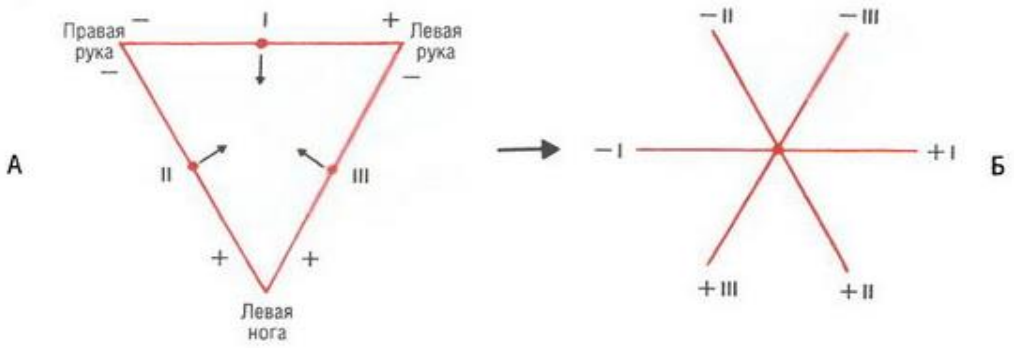


Рисунок 1.1. Трикутник серцевого ритму й сектори шести відведень.

Заслугою Альфреда Кона стало введення нового пристрою з електродами, що дозволило зменшити вагу елементів гальванометра. Протягом 1920-х і до 1930-х років продовжувалися модифікації електродів, що призвело до появи більш сучасних варіантів приладів для реєстрації електрокардіограм.

Пізніше, вчені з Уельса присоски й в даний час вони використовуються, для всіх 12 відведень. На рисунку 1.3 зображено розміщення 12 відведень.

Рис.1.2. Присоски ЕКГ.

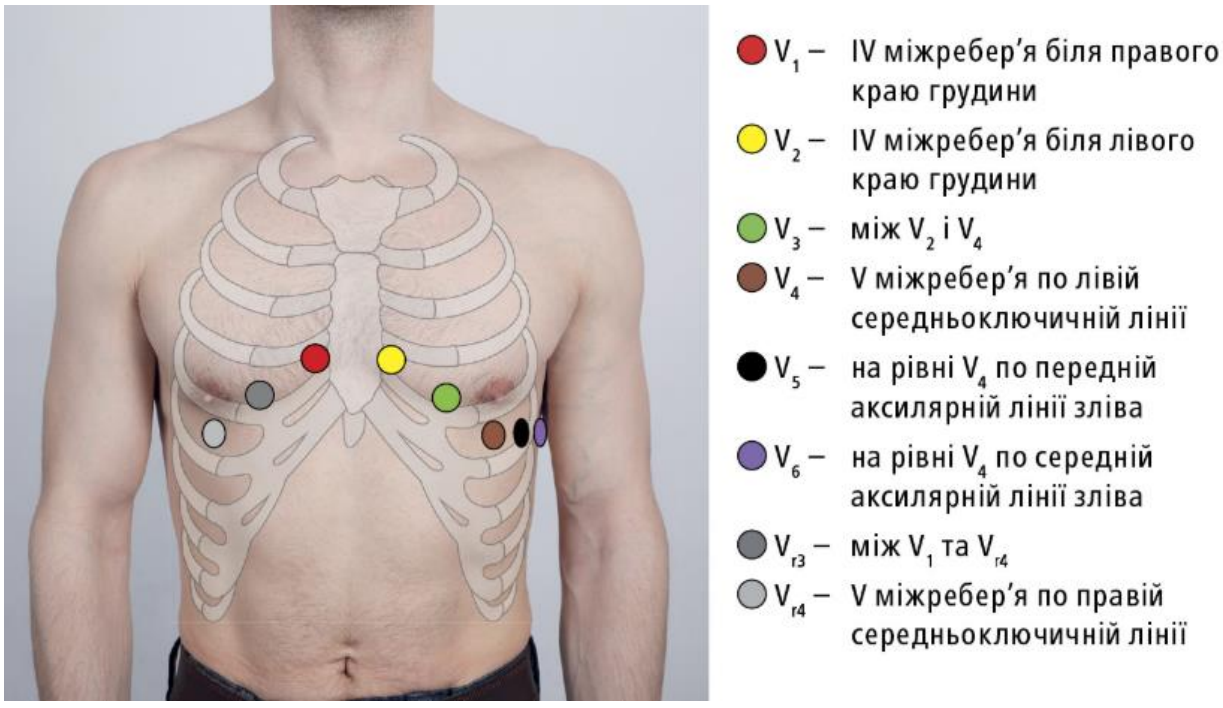


Рис. 1.3. Розміщення 12 відведень.

Наступним етапом в еволюції апаратів ЕКГ полягав в використовування електронних ламп, для посилення сигналу. Перший прилад такого типу був розроблений в США компанією General Electric Corporation. Потім в

електрокардіографії стали застосовувати електронно-променеві трубки. Це дозволило покращити фізичні характеристики записуючих пристроїв [6].

Впровадження посилювачів дозволило записувати кардіограму відразу на папері, без попередньої обробки. Після винаходу мікроконтролерів і цифрових сигналів світ ЕКГ значно змінився. Методи цифрової обробки сигналів дозволили значно спростити операції, які до цього часу лікарям приходилось здійснювати вручну, а також відкрили перед дослідниками нові можливості і нові властивості кардіограми, які дозволили виявити нові залежності між кардіосигналом і станом серцевої системи пацієнта.

Автори статті [1] пояснюють своє обґрунтування вибору бінарного підходу до класифікації для розрізнення "стресового" і "не стресового" станів, а також заглиблюються у використання мереж з довгою короткочасною пам'яттю (LSTM) як частини процесу класифікації. Дані, на яких проводилося дослідження, були зібрані за допомогою смартфонів на базі Android, оснащені додатком StudentLife, призначеним для автоматичного збору даних та отримання зворотного зв'язку від користувачів. Автори запропонували детальний погляд на доцільність та виклики використання моделей LSTM для прогнозування стресу на основі даних мобільного зондування. Навіть незважаючи на те, що дослідження напряму не пов’язане з використанням показників серцево-судинної системи для виявлення стресу, воно все одно дозволяє зрозуміти основні концепції та можливості методів класифікації у сфері виявлення стресу. Хоча дослідження є багатообіцяючим, воно також висвітлює сфери, які потребують подальших досліджень і вдосконалення, зокрема, у розширенні різноманітності даних і вирішенні проблем перенавчання моделей, щоб повністю використати потенціал LSTM у реальних застосуваннях.

Автори наступної статті [2] обговорюють нюанси вимірювання стресу через фізіологічні біосигнали за допомогою технології носіння сенсорів. Визнаючи, що на сигнали стресу кожної людини можуть впливати різні внутрішні фактори, такі як вік, стать, етнічна приналежність, дієта та емоційна

реакція на носіння біометричних пристроїв, автори аргументують необхідність персоналізованого підходу до виявлення стресу. Традиційні методи, що використовують фіксовані порогові значення індикаторів, не можуть бути універсальними через значні відмінності у фізіологічних реакціях між людьми. Щоб вирішити цю проблему, автори пропонують стратегію, яка налаштовує правила виявлення стресу для кожної людини. Це включає в себе коригування порогових значень на основі специфічних для суб'єкта базових показників, що дозволяє отримати більш точну і надійну оцінку рівня стресу. Необхідність таких індивідуалізованих правил підкреслюється прикладами, що демонструють значні відмінності в температурі шкіри та електродермальний активності серед випробовуваних, ілюструючи обмеження універсального підходу в точному визначенні рівня стресу.

Дослідження [3], яке бере до уваги показники ЧСС та варіабельність серцевого ритму, дає досить важливе розуміння підходів до оцінки стресу. Щоб викликати в учасників стрес і виміряти його наслідки, в дослідженні використовувався Trier Stress test [4], а ВСР відстежувалася за допомогою натільного пристрою FitBit. Основною метою цього дослідження було точне прогнозування гострої реакції на стрес і використання отриманих даних для створення алгоритму, спеціально розробленого для виявлення стресу в технологіях смарт-годинників. Дослідження мало на меті встановити, чи можна використовувати варіабельність серцевого ритму для прогнозування реакції на стрес. Дослідження порівняло загальну популяцію зі студентами-медиками і з'ясувало, що показники ВСР були меншими у студентів-медиків під час стресового завдання. Однак, ці показники ВСР не мали значного впливу на психометричні показники стрес-тестування, що вимагає подальшого дослідження з більшими групами. Також було встановлено, що активність симпатичної нервової системи збільшується після стресового завдання як у загальній популяції, так і у студентів-медиків.

1.2. Обґрунтування актуальності задачі

Актуальність розробки біомедичної системи візуалізації динаміки змін серцевого ритму зумовлена кількома важливими факторами, що відображають сучасні потреби в медицині, наукових дослідженнях та персоналізованому здоров'ї.

Зростаюча поширеність серцево-судинних захворювань (ССЗ) виявляється в тому, що вони залишаються провідною причиною смертності та інвалідності у світі. Рання діагностика, моніторинг та ефективне управління цими станами є критично важливими. Аналіз динаміки серцевого ритму (варіабельності серцевого ритму, ВСР) є цінним індикатором функціонального стану серцево-судинної системи та може допомогти у виявленні ранніх ознак патологій, оцінці ризиків та прогнозуванні перебігу захворювань.

Варіабельність серцевого ритму (ВСР) відображає складну взаємодію між симпатичною та парасимпатичною нервовими системами, що контролюють серцеву діяльність. Зниження ВСР часто асоціюється з підвищеним ризиком розвитку ССЗ, раптової серцевої смерті, метаболічних порушень, стресу, депресії та інших патологічних станів. Візуалізація динаміки ВСР дозволяє лікарям та дослідникам краще розуміти ці складні взаємозв'язки та використовувати ВСР як чутливий біомаркер.

Потреба в неінвазивному та безперервному моніторингу полягає в тому, що **с**учасні технології дозволяють здійснювати неінвазивний та часто безперервний моніторинг серцевого ритму за допомогою портативних пристроїв (смарт-годинники, фітнес-трекери, холтерівське моніторування). Однак, велика кількість даних, що генеруються цими пристроями, потребує ефективних інструментів для їх візуалізації та інтерпретації. Система візуалізації динаміки змін серцевого ритму може спростити аналіз цих даних, зробити їх більш зрозумілими та корисними для прийняття клінічних рішень або самостійного моніторингу.

Візуалізація індивідуальних змін серцевого ритму в часі може допомогти у розробці персоналізованих стратегій профілактики та лікування. Відстеження реакції серцевого ритму на різні фактори (фізична активність, стрес, сон, медикаменти) дозволяє оптимізувати спосіб життя та терапію для кожного конкретного пацієнта.

ВСР є важливим об'єктом наукових досліджень у галузі кардіології, нейрофізіології, спортивної медицини та психофізіології. Ефективні інструменти візуалізації динаміки серцевого ритму можуть значно полегшити аналіз великих обсягів даних, виявлення закономірностей та формулювання нових наукових гіпотез.

Наочна візуалізація динаміки серцевого ритму може зробити процес обговорення стану здоров'я більш ефективним та зрозумілим для пацієнта. Графіки та діаграми можуть допомогти пацієнту краще усвідомити зміни у своєму серцевому ритмі, їх зв'язок з різними факторами та важливість дотримання рекомендацій лікаря.

Враховуючи вищезазначені аспекти, розробка біомедичної системи візуалізації динаміки змін серцевого ритму є вельми актуальною та своєчасною задачею**,** що має значний потенціал для покращення діагностики, моніторингу, лікування серцево-судинних захворювань, сприяння розвитку персоналізованої медицини та поглиблення наукових знань у галузі фізіології серця.

1.3. Обґрунтування і вибір технічних підходів для вирішення задачі бакалаврської роботи

Розробка біомедичної системи візуалізації динаміки змін серцевого ритму вимагає ретельного обґрунтування та вибору технічних підходів на кожному етапі проєктування та реалізації. Метою є створення ефективного, надійного та зручного інструменту для аналізу та інтерпретації даних серцевого ритму.

Вибір мови програмування.

Для обробки та візуалізації біомедичних даних, а також для розробки веб-інтерфейсів, мова програмування **Python** є оптимальним вибором.

Переваги Python полягає в тому, що ця мова має велику кількість спеціалізованих бібліотек: для наукових обчислень (NumPy, SciPy), аналізу даних (Pandas), візуалізації (Matplotlib, Seaborn, Plotly, Bokeh), обробки сигналів (SciPy), а також для розробки веб-додатків (Flask, Django).

Синтаксис Python є лаконічним та зрозумілим, що полегшує розробку, налагодження та підтримку коду. Python має активну спільноту розробників, що забезпечує доступ до великої кількості документації, навчальних матеріалів та готових рішень. Python легко інтегрується з іншими мовами програмування та платформами.

Тому обираємо Python як основна мова програмування для розробки системи.

Вибір середовища розробки.

Для інтерактивної розробки, дослідження даних та візуалізації **Jupyter Notebook** є ідеальним середовищем.

Переваги Jupyter Notebook в тому, що він дозволяє виконувати код по комірках, миттєво бачити результати та експериментувати з даними, має можливість поєднання коду, тексту, візуалізацій та математичних формул, створює зрозумілі та самодостатні документи, що полегшує розуміння процесу аналізу даних.

Середовище надає легку інтеграцію з бібліотеками візуалізації, дозволяє швидко створювати та відображати графіки. Готові результати можна експортувати у вигляді HTML, PDF, слайдів тощо. Тому обираємо *Jupyter Notebook* як основне середовище розробки та дослідження.

Вибір бібліотек для обробки та аналізу даних серцевого ритму.

Для ефективної роботи з даними серцевого ритму необхідні бібліотеки, що підтримують різні формати файлів та надають інструменти для попередньої обробки, вилучення характеристик та аналізу ВСР.

WFDB (Waveform Database Library) призначена для читання та обробки даних з баз даних PhysioNet, які часто використовуються в дослідженнях серцевого ритму. Бібліотека підтримує різні формати зберігання сигналів та анотацій.

Вибір бібліотек для візуалізації динаміки серцевого ритму.

Для наочного представлення динаміки серцевого ритму та його характеристик необхідні потужні та гнучкі бібліотеки візуалізації. Matplotlib - базова бібліотека для створення статичних, анімованих та інтерактивних візуалізацій у Python. Надає широкий контроль над елементами графіків. Seaborn - бібліотека високого рівня, побудована на основі Matplotlib, що спрощує створення інформативних та естетично привабливих статистичних графіків, включаючи часові ряди та розподіли характеристик ВСР. Plotly - бібліотека для створення інтерактивних графіків, які можна масштабувати, наводити курсор для отримання додаткової інформації та зберігати у веб-форматі. Це може бути корисним для створення динамічних дашбордів. Комбінація Matplotlib для базових графіків та Seaborn для статистичних візуалізацій. Розглянути можливість використання Plotly для створення інтерактивного веб-інтерфейсу в майбутньому.

Розробка графічного інтерфейсу (за необхідності).

Для забезпечення зручної взаємодії користувача з системою може знадобитися графічний інтерфейс. На початковому етапі розробки зосередитися на створенні функціональності візуалізації за допомогою Jupyter Notebook. У майбутньому розглянути можливість розробки веб-інтерфейсу з використанням Flask для забезпечення ширшого доступу та інтерактивності.

Зберігання даних (за необхідності).

Залежно від джерел даних та потреби у збереженні результатів аналізу може знадобитися система зберігання даних.

- Локальні файли - для невеликих обсягів даних та прототипування.
- CSV або JSON - прості текстові формати для зберігання табличних або структурованих даних.

- Бази даних (SQLite, PostgreSQL, MySQL) - для великих обсягів даних та забезпечення структурованості та можливості запитів.

На початковому етапі використовувати локальні файли (наприклад, у форматах, підтримуваних WFDB або Pandas). У майбутньому, при роботі з великими обсягами даних, розглянути використання легкої вбудованої бази даних, такої як SQLite.

Обрані технічні підходи, що включають мову програмування Python, середовище розробки Jupyter Notebook та ряд спеціалізованих бібліотек для обробки, аналізу та візуалізації біомедичних даних, є обґрунтованими та відповідають поставленій задачі розробки біомедичної системи візуалізації динаміки змін серцевого ритму. Ці технології забезпечують необхідну гнучкість, потужність та наявність інструментів для ефективної реалізації проєкту. Подальші рішення щодо розробки графічного інтерфейсу та системи зберігання даних будуть прийматися залежно від розвитку проєкту та виявлених потреб користувачів.

2. СТРУКТУРНА СХЕМА ВІЗУАЛІЗАЦІЇ ДИНАМІКИ ЗМІН СЕРЦЕВОГО РИТМУ

2.1. Дослідження електрокардіосигналу як біологічного об'єкту і аналіз принципів побудови систем

Оскільки основною метою даної роботи є створення програмного забезпечення для візуалізації ЕКГ-сигналу та його параметрів, важливо розуміти, які характеристики цього сигналу можуть бути корисними для кінцевого користувача, а також які вимоги до якості візуалізації необхідні для його коректного відображення. У цьому підрозділі розглядається процес цифрової обробки сигналів, отриманих за допомогою електрокардіографа. На рисунку 2.1 представлено загальну схему обробки сигналу, а також приклади кардіограм з різними типами серцевого ритму.

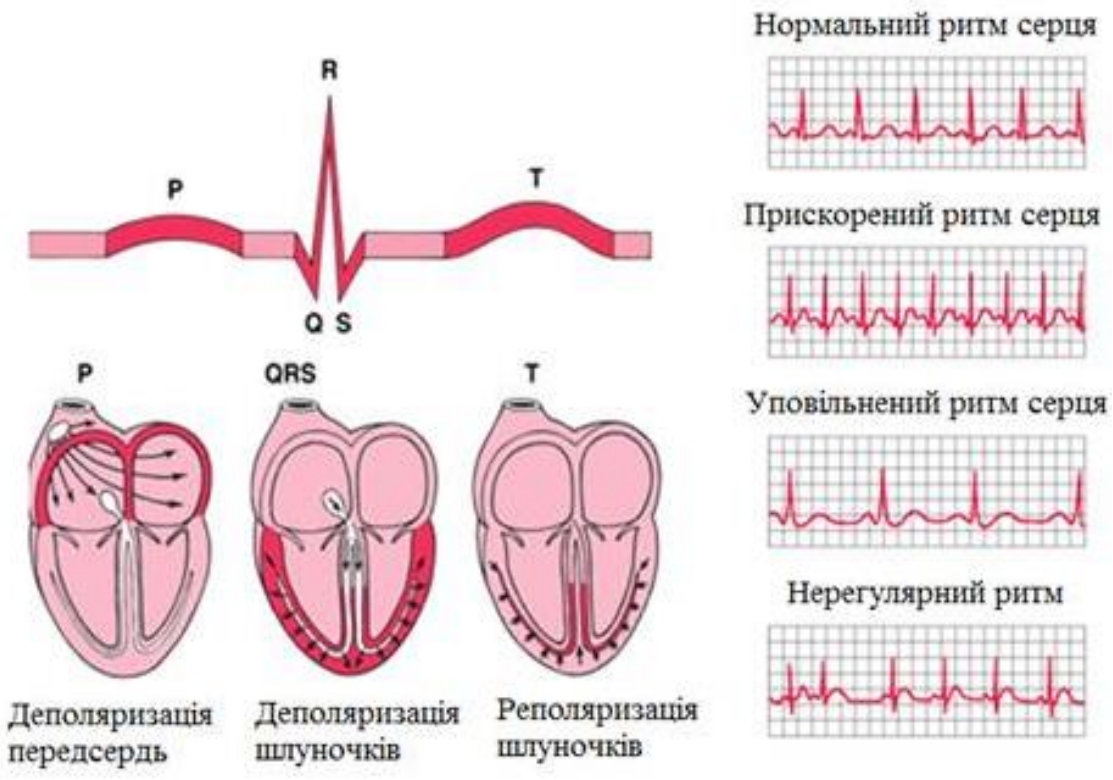


Рисунок 2.1. Схема циклу серцевих скорочень та приклади візуалізації різних типів серцевого ритму.

Початковим етапом є збирання даних, яке включає як апаратну, так і програмну складову. Апаратна частина складається з аналогово-цифрового перетворювача (АЦП), а програмна – відповідає за керування його роботою, надсилаючи відповідні команди. Для реалізації цієї взаємодії можуть бути використані мови програмування, які забезпечують доступ до низькорівневого керування обладнанням.

Для правильного вибору аналого-цифрового перетворювача (АЦП) необхідно враховувати параметри сигналу, який підлягає оцифруванню. У таблиці 2.1 наведено нормовані електричні та часові характеристики електрокардіографічного сигналу, які слугують орієнтиром при підборі технічних засобів для його реєстрації та обробки.

Таблиця 2.1

Параметри нормованих показників ЕКГ

	Амплітуда, мВ	Тривалість, с
Зубець P	0 – 0,25	0,07 – 0,11
Інтервал PQ	В залежності від пацієнта	0,12 – 0,2
QRS – комплекс	0,3 – 5	0,06 – 0,1
Інтервал QT	В залежності від пацієнта	0,35 – 0,44
ST – сегмент	0,4 – 1	0,35 – 0,44
Зубець T	В залежності від пацієнта	0,06 – 0,15
Зубець U	0 – 0,1	0,1 – 0,2

Сигнал електрокардіограми (ЕКГ) в основному зосереджений у діапазоні частот від 0,5 до 30 Гц, при цьому максимальна енергетична щільність припадає на частоту близько 15 Гц. Проте за наявності патологічних змін серцевої діяльності частотний спектр сигналу може змінюватися. На рисунку 2.2 зображено стандартне розташування зубців та інтервалів ЕКГ-

сигналу, яке прийнято як загальновизнаний еталон і використовується в медичній практиці, зокрема в кардіології.

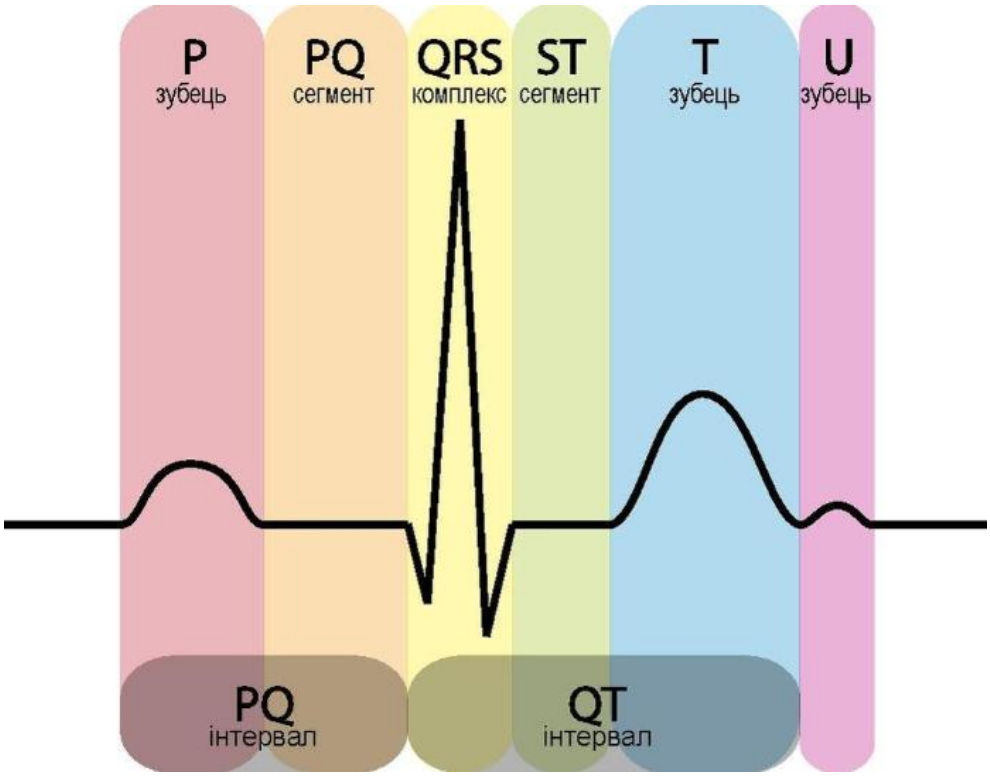


Рисунок 2.2. Структура ЕКГ-сигналу: інтервали та зубці.

Частота дискретизації — це параметр, який визначає, з якою швидкістю аналого-цифровий перетворювач (АЦП) здійснює вибірку значень вхідного сигналу. Виходячи з наведеного вище частотного діапазону, теорема Котельникова встановлює мінімальну необхідну частоту дискретизації на рівні 100 Гц для уникнення втрати інформації. Водночас Американське товариство кардіологів рекомендує використовувати частоту не нижчу за 500 Гц для забезпечення високої якості цифрового сигналу.

Основними технічними характеристиками АЦП є розрядність, смуга пропускання, чутливість і швидкодія. Кількість бітів (розрядність) безпосередньо впливає на точність перетворення, проте варто враховувати, що надмірне збільшення розрядності не завжди призводить до суттєвого поліпшення якості через вплив шумів, перешкод та нелінійних спотворень. Для ЕКГ-досліджень зазвичай достатньо 8-розрядного перетворення, але

найчастіше використовуються 16-розрядні АЦП, які забезпечують кращу якість сигналу при незначному підвищенні вартості.

Чутливість АЦП відображає кількість рівнів квантованого сигналу на одиницю вхідної напруги. Стандартним вважається показник 200 рівнів на мілівольт, чого цілком достатньо для перетворення ЕКГ-сигналу. Проте за потреби можливо використовувати пристрої з підвищеною чутливістю, що дозволяє досягти ще більш точного відображення електричної активності серця.

Після оцифрування сигнал потребує фільтрації, оскільки він часто містить значну кількість шумових компонентів. Найбільш поширеними типами фільтрів, які застосовуються для очищення ЕКГ-сигналу, є:

- фільтри для пригнічення мережевих наведень;
- фільтри низьких частот, що обмежують спектр сигналу в області високих частот;
- фільтри високих частот, призначені для зменшення міографічного шуму та усунення низькочастотних артефактів.

Ці фільтри можуть бути реалізовані у різний спосіб, і вибір їхньої конфігурації залежить від конкретного розробника або типу кардіографа. Саме тому ЕКГ-сигнали, отримані за допомогою різного обладнання, можуть мати відчутні відмінності.

Після проходження етапу фільтрації цифровий сигнал може бути переданий для подальшої комп’ютерної обробки або безпосередньо лікарю для клінічної інтерпретації. Зокрема, одним із напрямів обробки є автоматичне виявлення QRS-комплексів, що істотно полегшує діагностичну діяльність медичного персоналу. Крім того, важливим інструментом є аналіз варіабельності серцевого ритму, який розширює спектр доступних діагностичних методів. Додаткові можливості надає спектральний аналіз ЕКГ-сигналу, що дозволяє глибше дослідити його характеристики у частотній області.

Зрозуміло, що наведені етапи реєстрації та обробки електрокардіографічного сигналу мають значну кількість технічних аспектів, детальний розгляд яких виходить за межі даної роботи. Враховуючи, що в межах цього дослідження не передбачено безпосереднього збору фізіологічних даних, поданий огляд має суто ознайомчий характер і покликаний надати читачу базове уявлення про ключові поняття, необхідні для розуміння функціональних особливостей розроблюваного програмного продукту.

2.2. Огляд відомих математичних моделей опрацювання кардіосигналу

Традиційні методи обробки сигналів є основою для попередньої обробки ЕКГ та виділення базових характеристик. Серед них виділяють фільтрацію, яка включає лінійні фільтри (такі як фільтри Баттерворта, Чебишева для усунення високочастотних шумів, низькочастотного дрейфу ізолінії та мережевих перешкод) та адаптивні фільтри (наприклад, фільтр Калмана, LMS-алгоритм, що ефективно видаляють шуми, характеристики яких змінюються в часі).

Перетворення Фур'є (DFT/FFT) дозволяє перевести сигнал з часової області в частотну, що корисно для аналізу спектральних компонентів серцевого ритму та оцінки варіабельності серцевого ритму (ВСР).

Вейвлет-перетворення (DWT/CWT) забезпечує часово-частотний аналіз сигналу, що дозволяє локалізувати події як у часі, так і за частотою, роблячи його ідеальним для виявлення транзієнтних аномалій.

Виявлення P, QRS, T зубців є критично важливим для подальшого аналізу ритму та морфології. Найбільш поширеним і надійним є алгоритм Панна-Томпкінса для детекції QRS-комплексів. Також використовуються методи, засновані на вейвлет-перетворенні, та методи на основі машинного навчання, зокрема згорткові нейронні мережі (CNN), для автоматичного виявлення та сегментації зубців ЕКГ з високою точністю.

Варіабельність серцевого ритму (ВСР) є потужним інструментом для оцінки стану автономної нервової системи. Методи аналізу ВСР у часовій області включають SDNN (стандартне відхилення NN-інтервалів), RMSSD (корінь квадратний із середнього значення квадратів різниць послідовних NN-інтервалів) та pNN50 (відсоток NN-інтервалів, що відрізняються більш ніж на 50 мс).

Методи аналізу ВСР у частотній області (наприклад, спектральний аналіз з використанням БПФ або авторегресійних моделей) визначають потужність у різних частотних діапазонах: VLF, LF, HF, а також співвідношення LF/HF, що вказує на баланс між симпатичною та парасимпатичною активністю.

До нелінійних методів аналізу ВСР належать аналіз Пуанкаре, ентропійні методи (ApEn, SampEn) для вимірювання складності ритму, та детрендований флуктуаційний аналіз (DFA) для оцінки фрактальних властивостей.

Сучасні підходи до опрацювання кардіосигналу все частіше використовують методи машинного навчання та глибокого навчання для автоматичної класифікації аритмій, прогнозування станів та персоналізованого моніторингу. Серед методів класифікації застосовуються метод опорних векторів (SVM), дерева рішень та випадкові ліси, а також K-найближчих сусідів (k-NN).

Нейронні мережі представлені багатошаровими перцептронами (MLP), згортковими нейронними мережами (CNN), що ефективні для безпосереднього аналізу "сирих" ЕКГ-сигналів, та рекурентними нейронними мережами (RNN, LSTM, GRU), ідеальними для аналізу часових послідовностей. Генеративні змагальні мережі (GAN) можуть використовуватися для генерації синтетичних ЕКГ-сигналів. Для підвищення точності та надійності системи часто застосовуються ансамблеві методи, що комбінують кілька моделей.

Забезпечення високої якості сигналу є запорукою достовірного аналізу. Для виявлення артефактів та аномалій використовуються прості порогові методи, методи на основі аналізу незалежних компонент (ICA) для відокремлення артефактів від кардіосигналу, моделі прихованих Маркових ланцюгів (HMM) для виявлення відхилень від нормальних патернів, а також глибоке навчання для детекції аномалій, зокрема автокодувальники.

Вибір математичної моделі для опрацювання кардіосигналу залежить від конкретних завдань біомедичної системи візуалізації динаміки змін серцевого ритму. Комбінування класичних методів обробки сигналів з передовими підходами машинного та глибокого навчання дозволяє створювати надійні та точні системи для діагностики, моніторингу та прогнозування серцево-судинних станів. Постійний розвиток цих моделей відкриває нові можливості для більш глибокого розуміння фізіологічних процесів та покращення клінічної практики.

2.3. Вибір та обґрунтування структурної схеми системи

Розробка біомедичної системи візуалізації динаміки змін серцевого ритму передбачає комплексний підхід, що охоплює апаратну частину для збору даних, програмне забезпечення для їх обробки та аналізу, а також інтерфейс для візуалізації результатів. Запропонована структурна схема спрямована на забезпечення надійності, точності та зручності використання системи.

Основою системи є отримання якісного електрокардіографічного сигналу. Вибір **ЕКГ-сенсорного модуля** з кількома відведеннями (наприклад, три- або п'ятивідведення) дозволяє забезпечити достатню інформативність для аналізу серцевого ритму. Сучасні модулі мають низький рівень шуму, високу чутливість та інтегровані підсилювачі, що мінімізує зовнішні перешкоди. Вбудовані АЦП (аналого-цифрові перетворювачі) з високою розрядністю (наприклад, 12-16 біт) та частотою дискретизації (мінімум 250-500 Гц) є критично важливими для збереження деталізації сигналу.

Таблиця 2.1. Функції елементів структурної схеми

№ на схемі	**Назва**	**Функції**
1	Модуль збору даних (ЕКГ-сенсорний модуль)	Отримання електричних потенціалів від серця, їх посилення, фільтрація аналогових шумів та перетворення в цифровий формат.
2	Модуль бездротової передачі даних	Передача оцифрованого ЕКГ-сигналу від сенсорного модуля до обчислювального пристрою (комп'ютера, смартфона).
3	Обчислювальний модуль (комп'ютер або вбудований мікроконтролер)	Прийом даних, попередня обробка, застосування математичних моделей, зберігання даних та взаємодія з модулем візуалізації
4	Модуль обробки та аналізу сигналу	Фільтрація, виявлення характерних точок, аналіз варіабельності серцевого ритму (ВСР), розпізнавання аритмій, моделювання динаміки
5	Модуль зберігання даних	Зберігання "сирих" ЕКГ-даних, метаданих (дата, час, ідентифікатор пацієнта), результатів аналізу ВСР, виявлених аритмій та інших показників.
6	Модуль візуалізації та інтерфейсу користувача	Відображення ЕКГ-кривої в реальному часі та історичних даних. Графіки показників ВСР (гістограми NN-інтервалів, графіки Пуанкаре, спектрограми). Візуалізація виявлених аритмій та аномалій (маркування на ЕКГ-кривій). Настроювані параметри відображення (масштаб, швидкість). Можливість генерації звітів.

Для забезпечення комфорту пацієнта та мобільності системи доцільно використовувати бездротову передачу даних. Технології, такі як Bluetooth Low Energy (BLE) або Wi-Fi, є оптимальними варіантами. BLE забезпечує низьке

енергоспоживання, що важливо для портативних пристроїв, та достатню швидкість передачі для ЕКГ-сигналу. Wi-Fi може бути кращим для передачі більших обсягів даних або на більші відстані. Вибір залежить від конкретних вимог до дальності та енергоспоживання.

Обчислювальний модуль є "мозком" системи, де відбувається основна обробка та аналіз даних. Для складних математичних моделей та візуалізації потрібна значна обчислювальна потужність. Персональний комп'ютер (ПК) з достатньою оперативною пам'яттю та процесором пропонує гнучкість у розробці та тестуванні, а також дозволяє використовувати потужні бібліотеки для обробки сигналів та машинного навчання. Для вбудованих систем або портативних рішень може бути використаний високопродуктивний мікроконтролер або одноплатний комп'ютер (наприклад, Raspberry Pi).

Модуль обробки та аналізу сигналу це програмний компонент, що реалізує розглянуті математичні моделі. Модульність його структури дозволить легко додавати нові алгоритми або оновлювати існуючі. Використання бібліотек для обробки сигналів (наприклад, SciPy, NumPy в Python) та машинного навчання (TensorFlow, PyTorch, Scikit-learn) є оптимальним для швидкої розробки та ефективності.

Зберігання ЕКГ-сигналів та результатів їх аналізу є важливим для подальшого досліджень, порівняння динаміки та навчання моделей машинного навчання. Використання реляційних баз даних (наприклад, PostgreSQL, MySQL) або NoSQL баз даних (наприклад, MongoDB) забезпечує ефективне зберігання, швидкий доступ та масштабованість. Для великих обсягів даних може бути розглянуте хмарне сховище.

Дружній та інтуїтивно зрозумілий інтерфейс є критично важливим для клініцистів та пацієнтів. Він повинен забезпечувати наочне відображення ЕКГ-кривої, графіків ВСР, тенденцій змін ритму та виявлених аномалій. Використання сучасних бібліотек для візуалізації (наприклад, Matplotlib, Plotly для Python, D3.js для веб-інтерфейсів) дозволяє створювати динамічні та інтерактивні графіки.

2.4. Структурна схема програмного забезпечення біомедичної системи візуалізації динаміки змін серцевого ритму

Програмне забезпечення розробленої біомедичної системи візуалізації динаміки змін серцевого ритму (рис. 3.15) складається з декількох функціональних модулів, кожен з яких виконує визначені завдання у загальному процесі збору, обробки, аналізу та візуалізації даних електрокардіографії.

Модуль отримання даних забезпечує первинне завантаження електрокардіографічних сигналів. Джерелами даних можуть бути відкриті медичні бази (наприклад, PhysioNet), локальні файли у форматах WFDB, EDF, CSV, а також сигнали, отримані з реального обладнання. Модуль відповідає за коректне завантаження та підготовку сирих даних для подальшої обробки.

Модуль попередньої обробки даних виконує очищення сигналу та його підготовку до аналізу. Зокрема, здійснюється фільтрація перешкод — усунення мережевих наводок (50/60 Гц), дрейфу ізолінії, міографічних шумів та інших артефактів. Також реалізується нормалізація амплітуди сигналу. Для обробки залучаються відповідні бібліотеки Python: wfdb, scipy, numpy тощо.

Основний аналіз здійснюється у модулі обробки та аналізу даних. Тут реалізуються алгоритми детекції характерних точок ЕКГ-сигналу — зубців P, QRS, T (наприклад, за алгоритмом Панна-Томпкінса), розраховуються показники варіабельності серцевого ритму (SDNN, RMSSD, pNN50), проводиться спектральний аналіз із використанням швидкого перетворення Фур'є (FFT) або авторегресійних моделей. Додатково здійснюється виявлення аритмій та інших аномалій ритму. Розраховані показники є основою для подальшого медичного аналізу функціонального стану серцево-судинної системи.

За необхідності у систему може бути інтегровано модуль машинного навчання. Його завданням є автоматична класифікація станів серцевого ритму, виявлення закономірностей, прогнозування ризиків розвитку патологій. У

цьому модулі можуть використовуватись алгоритми опорних векторів (SVM), дерева рішень, метод найближчих сусідів (k-NN), згорткові та рекурентні нейронні мережі (CNN, LSTM), а також ансамблеві методи.

Модуль візуалізації відповідає за графічне подання результатів аналізу. Зокрема, здійснюється виведення ЕКГ-кривої в реальному часі, побудова графіків варіабельності серцевого ритму, гістограм NN-інтервалів, спектрограм та діаграм Пуанкаре. Для візуалізації застосовуються бібліотеки matplotlib, seaborn, plotly тощо.

Модуль зберігання даних забезпечує збереження як сирих, так і оброблених даних у зручних форматах (CSV, JSON, бази даних SQLite, PostgreSQL). Зберігаються дані пацієнтів, історія вимірювань та результати розрахунків.

Окремо реалізований модуль генерації звітів дозволяє автоматично формувати медичні звіти на основі результатів обробки. Генерація здійснюється у форматах PDF, HTML чи графічних файлів, з можливістю експорту в зовнішні системи.

Уся система керується через графічний інтерфейс користувача, який дозволяє завантажувати дані, налаштовувати параметри аналізу, переглядати результати обробки, зберігати та експортувати підсумкові дані. Інтерфейс створюється із застосуванням технологій Jupyter Notebook, Flask, Tkinter або веб-технологій, що забезпечує зручну та інтуїтивно зрозумілу роботу для кінцевого користувача.

Запропонована структурна схема програмного забезпечення дозволяє забезпечити комплексну обробку електрокардіографічних даних, що дає можливість ефективно здійснювати моніторинг, діагностику та візуалізацію динаміки змін серцевого ритму.

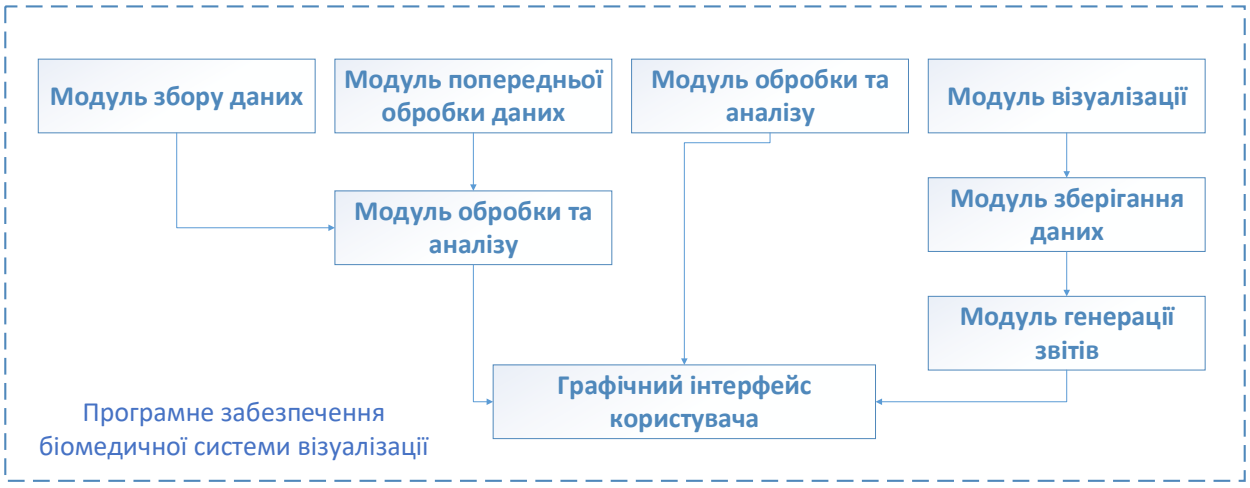


Рисунок 2.1. Структурна схема програмного забезпечення быомедичноъ системи медичноъ вызуалызацыъ змын сердцевого ритму

2.5. Структурна схема апаратного забезпечення біомедичної системи візуалізації динаміки змін серцевого ритму

Розроблена біомедична система візуалізації динаміки змін серцевого ритму являє собою комплекс взаємодіючих функціональних блоків, кожен з яких виконує свою специфічну роль у процесі отримання, обробки, зберігання та представлення даних про серцеву діяльність (рис. 3.16).

Первинним у цій системі є **Блок реєстрації сигналу**, завданням якого є захоплення біоелектричної активності серця. Він складається з **Електродів (ЕКГ-датчиків)**, які розміщуються на тілі пацієнта та перетворюють електричні потенціали серця на аналоговий електричний сигнал. Далі цей слабкий аналоговий сигнал надходить на **Підсилювач сигнала**, де його амплітуда збільшується до рівня, придатного для подальшої обробки, при цьому мінімізується додавання власного шуму. Після підсилення сигнал проходить через **Фільтри**, які здійснюють фільтрацію шумов та артефактов. Це критично важливий етап, оскільки він дозволяє очистити ЕКГ-сигнал від низькочастотних перешкод (наприклад, дрейфу ізолінії, спричиненого диханням або рухом) та високочастотних шумів (таких як мережеві перешкоди 50/60 Гц або м'язові артефакти), забезпечуючи високу якість вхідних даних для подальшого аналізу.

Очищений аналоговий сигнал далі направляється до **Блоку оцифровки**, де відбувається його перетворення з безперервної аналогової форми у дискретний цифровий вигляд. Цю функцію виконує **Аналогово-цифровой перетворювач (АЦП)**, який періодично вимірює амплітуду сигналу та перетворює її на відповідне числове значення, забезпечуючи готовність даних до комп'ютерної обробки.

Отримані цифрові дані надходять до **Блоку обробки даних**, який є серцем аналітичної частини системи. Тут розташована **Система обробки ЕКГ**, що відповідає за виділення характерних зубців ЕКГ-кривої (зокрема QRS-комплексів) та подальший расчет ЧСС (частоти сердечных сокращений). Цей же блок включає **Систему виявлення артефактів**, яка ідентифікує та, за можливості, коригує аномалії сигналу, що не були повністю усунені фільтрами, підвищуючи достовірність отриманих результатів.

Всі оброблені дані, включаючи вихідний ЕКГ-сигнал, розраховані значення ЧСС та інші показники, передаються до **Блоку зберігання даних**. Цей блок представлений **Пам’яттю** (оперативною та постійною) та **Базою даних (БД)**, де інформація зберігається для довготривалого аналізу, ведення історії пацієнта та подальших наукових досліджень.

Для взаємодії з користувачем та наочного представлення отриманих даних служить **Блок візуалізації**. Він використовує **Екран або монітор** як пристрій виводу інформації. В рамках цього блоку реалізовано **Інтерфейс користувача**, який забезпечує графическое отображение сердечного ритма. Це може бути візуалізація ЕКГ-кривой в реальному часі, числовие значения ЧСС, а також інші важливі показники варіабельності серцевого ритму, представлені у вигляді графіків, гістограм або діаграм, що дозволяє користувачу швидко та ефективно інтерпретувати стан серцево-судинної системи.

Нарешті, **Блок управління** координує роботу всіх компонентів системи. Центральним елементом цього блоку є **Контролер системи**, який забезпечує синхронізацію процесів, керує потоками даних та дозволяє здійснювати

Налаштування режиму роботи системи, адаптуючи її до різних умов використання та потреб користувача.

Таким чином, ці блоки, працюючи злагоджено, формують повноцінну біомедичну систему для ефективної та точної візуалізації динаміки змін серцевого ритму.

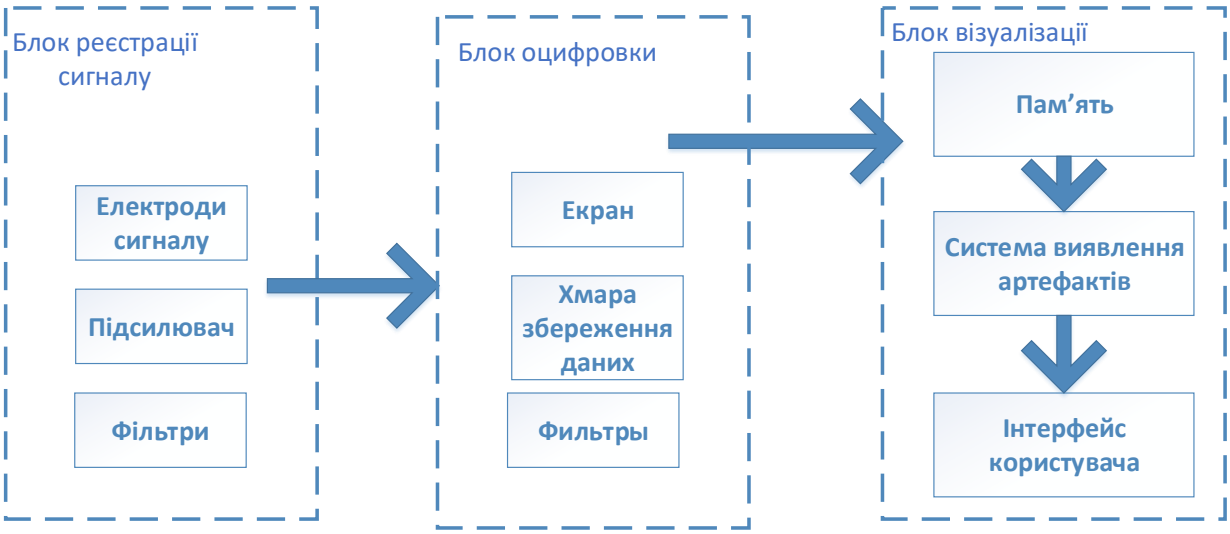


Рисунок 2.2. Структурна схема апаратного забезпечення біомедичної системи візуалізації динаміки змін серцевого ритму

2.6. Інструменти розробки

2.6.1. Опис ресурса PhysioNet і його структура

Оскільки ключовою метою даного дослідження є реалізація візуалізації електрокардіографічних сигналів, зокрема на основі даних із ресурсу *PhysioNet*, доцільно розглянути історію створення та основне призначення цього веб-ресурсу.

PhysioNet (рис. 1.6) — це інтерактивна платформа, яка забезпечує відкритий доступ до високоякісних фізіологічних сигналів і супровідного програмного забезпечення з відкритим кодом, що призначене для дослідників у сфері біомедичних наук. Портал був офіційно запущений у вересні 1999 року за ініціативи Національного центру дослідницьких ресурсів (NCRR) США. Він

функціонує як науковий майданчик для обміну біомедичними даними, алгоритмами аналізу та для порівняння нових методик обробки сигналів.

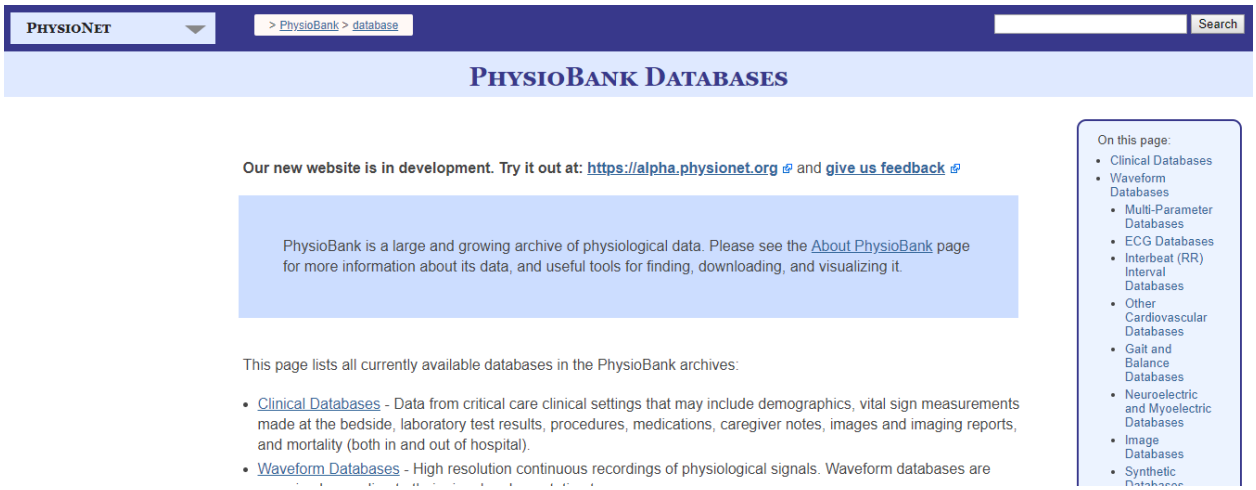


Рисунок 2.3. Веб-ресурс Physionet.

На момент квітня 2016 року архів *PhysioNet* налічував близько 4 терабайтів записаних сигналів, анотацій та пов'язаних даних [11]. Цей проєкт реалізовано в межах Науково-дослідного ресурсу складних фізіологічних сигналів і є результатом співпраці науковців з Медичного центру Бет-Ізрейел у Бостоні, Гарвардської медичної школи, Бостонського університету, Університету Макгілла та Массачусетського технологічного інституту.

Історія *PhysioNet* бере початок ще у 1970-х роках, коли команда дослідників працювала над розробкою мікрокомп'ютерних засобів моніторингу серцевих аритмій. Уже тоді було усвідомлено потребу у створенні стандартизованих баз даних ЕКГ для забезпечення об'єктивного тестування алгоритмів аналізу. Перші значні результати були досягнуті у 1980 році з публікацією бази даних аритмій, яка стала еталонною і активно використовувалася науковцями у всьому світі протягом наступних десятиліть.

До моменту запуску платформи в 1999 році група вже розповсюджувала свої напрацювання на CD-дисках, включаючи 11 колекцій даних. Ці напрацювання стали основою для створення двох ключових компонентів: *PhysioBank* — архіву сигналів, і *PhysioToolkit* — набору інструментів для їх

обробки. Відкритий онлайн-доступ до цих ресурсів значно посилив глобальну наукову співпрацю: нові дослідники отримали змогу працювати з реальними даними, а вже існуючі команди — порівнювати ефективність своїх методів. У подальшому база даних та програмне забезпечення *PhysioNet* активно поповнювалися завдяки внеску міжнародної дослідницької спільноти [12].

У процесі запуску платформи *PhysioNet* команда її розробників активно готувалася до презентації ресурсу під час конференції *Cardiology 2000*. Основною метою було ознайомлення міжнародної медичної спільноти, присутньої на щорічному симпозіумі *Computers in Cardiology (CinC)*, з функціональними можливостями *PhysioNet*, а також заохочення дослідників до участі в ініціативах, що сприяли б вирішенню актуальних клінічних проблем шляхом використання відкритих даних та інструментів ресурсу.

Завдяки вчасному внесенню відповідного масиву даних стало можливим проведення першого конкурсу *PhysioNet/CinC Challenge*, присвяченого задачі автоматичного виявлення апное сну за електрокардіографічними записами. Цей конкурс об’єднав більше десятка команд дослідників, чиї результати були представлені на конференції *CinC 2000*. Захід виявився успішним і започаткував щорічну традицію проведення подібних змагань, спрямованих на стимулювання прогресу в аналізі біомедичних сигналів [13].

Спочатку *PhysioNet* функціонував під егідою Національного центру дослідницьких ресурсів США (NCRR), який було розформовано у 2011 році. Починаючи з вересня 2007 року, платформа отримує фінансування в межах угоди про співпрацю з Національним інститутом біомедичної візуалізації та біоінженерії (NIBIB), а також за підтримки Національного інституту загальних медичних наук (NIGMS).

Ресурс *PhysioNet* створений з метою стимулювання поточних і нових досліджень у сфері аналізу складних біомедичних та фізіологічних сигналів. Його структура охоплює три основні взаємопов’язані компоненти:

- **PhysioBank** — це об’ємний та динамічно зростаючий архів цифрових записів фізіологічних сигналів, часових рядів і пов’язаних із ними даних, що

призначений для вільного використання в біомедичних дослідженнях. Колекції PhysioBank охоплюють сигнали серцево-легеневої, нервової та інших систем організму, отримані як від здорових осіб, так і від пацієнтів із різноманітними патологіями, зокрема раптовою серцевою смертю, серцевою недостатністю, епілепсією, порушеннями ходи, апное сну та віковими змінами. Дані представлені з широкого спектра досліджень, наданих різними науковими колективами [14].

- **PhysioToolkit** — це велика колекція програмного забезпечення з відкритим кодом, призначеного для обробки фізіологічних сигналів, виявлення важливих фізіологічних подій, візуалізації даних, побудови нових баз даних, моделювання сигналів, порівняльного аналізу алгоритмів тощо. Інструментарій включає як класичні методи аналізу, так і сучасні підходи, засновані на статистичній фізиці, нелінійній динаміці, аналізі нестаціонарних і нерівноважних процесів. Загальною метою є витяг прихованої, але значущої з клінічної або фундаментальної точки зору інформації з біосигналів [15].
- **PhysioNetWorks** — віртуальне середовище для колективної роботи, яке дозволяє дослідникам з усього світу створювати, вдосконалювати, документувати та готувати до публікації нові набори даних і програмне забезпечення. Учасники мають змогу отримати доступ до активних проєктів, долучатися до них як рецензенти або співавтори, а також працювати у спільних робочих просторах. Кожен проєкт має відповідального власника, який регулює політику доступу до матеріалів. *PhysioNet* забезпечує захист даних за допомогою щоденного інкрементного та щомісячного повного резервного копіювання. Усі зберігаючі диски зашифровані та розміщені в фізично захищених дата-центрах [16].

Завдяки такій структурі, *PhysioNet* сьогодні є однією з ключових платформ для наукових досліджень, пов’язаних з аналізом електрокардіограм (ЕКГ) та удосконаленням методів цифрової медицини.

У рамках даної роботи як основу для створення програмного забезпечення було обрано стандарт **PhysioNet**.

Ресурс *PhysioNet* представляє собою набір баз даних, які містять оцифровані фізіологічні сигнали та відповідні файли анотацій. Кожна база даних складається з колекції записів, які мають унікальні ідентифікатори (назви записів). Повний перелік записів для кожної бази надається у вигляді окремого списку, доступного на сайті PhysioNet. Зазвичай один запис представлений щонайменше трьома файлами, кожен з яких має спільну основу назви, але різне розширення, що вказує на його функціональне призначення (рис. 2.2) [17].

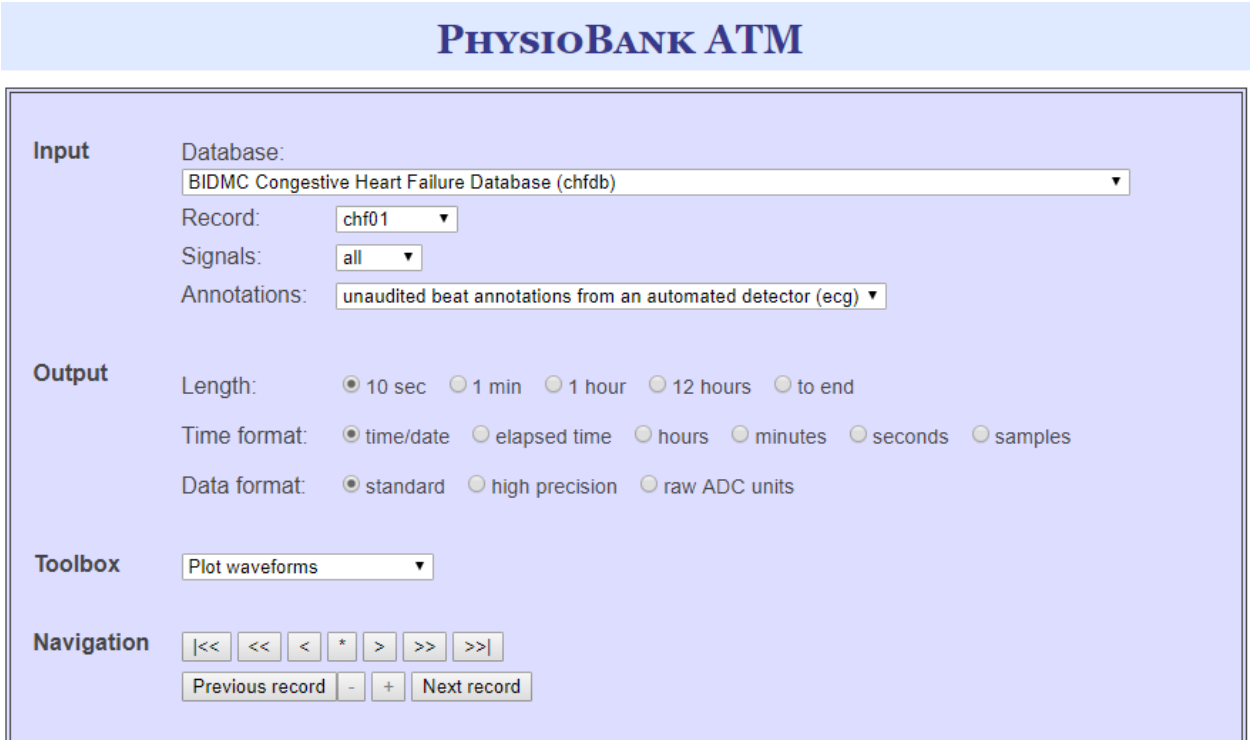


Рисунок 2.4. Структура організації записів на ресурсі PhysioNet.

На прикладі бази даних **MIT-BIH Arrhythmia Database**, один із записів має ім’я 100 і складається з трьох файлів:

- 100.dat — двійковий файл, який містить оцифровані значення фізіологічних сигналів;
- 100.hea — текстовий файл заголовка, що описує метаінформацію про запис;
- 100.atr — файл анотацій, який містить позначки важливих подій у сигналі.

Практично всі записи включають файл .dat, у якому зберігаються сигнали, зазвичай ЕКГ, у двійковому форматі. Ці файли можуть бути досить великими, оскільки містять повний запис тривалих фізіологічних процесів.

Файл .hea (header) — це короткий текстовий документ, який містить опис запису: формат зберігання, кількість каналів, частоту дискретизації, тип сигналу, калібрувальні коефіцієнти, характеристики АЦП, тривалість запису, час його початку, а також іншу службову інформацію. Цей файл є необхідним для коректного зчитування і обробки сигналів.

Файл анотацій .atr містить позначки особливих подій, що зафіксовані в сигналі. Наприклад, у файлі 100.atr містяться тимчасові мітки для кожного комплексу QRS (серцевого скорочення), із зазначенням його типу (нормальне скорочення, шлуночкова екстрасистола тощо), а також інші анотації, що відображають зміни серцевого ритму або якості сигналу. У різних базах анотації можуть описувати й інші аспекти сигналів, залежно від цілей дослідження [18–22].

Таким чином, стандарт PhysioNet забезпечує чітку структуру та уніфікований формат зберігання фізіологічних даних, що сприяє ефективному аналізу, автоматизованій обробці та візуалізації даних у біомедичних дослідженнях.

Формати сигналів у PhysioNet та подання фізичних значень

Файли сигналів із розширенням .dat у базі PhysioNet існують у кількох форматах, що відрізняються за розрядністю та способом кодування амплітудних значень. Кожен формат визначає, скільки бітів використовується для подання однієї вибірки сигналу, а також порядок збереження байтів (наприклад, молодший значущий байт першим).

Наприклад, **формат 32** передбачає збереження кожного відліку як 32-бітного цілого числа зі зворотним порядком байтів (молодший байт записується першим). Це дозволяє зберігати амплітуду сигналу з високою

точністю, але збільшує обсяг файлу. Основна мета використання таких форматів — збереження максимальної кількості інформації про сигнал, що була доступна під час його зняття цифровим пристроєм.

Оцифрування та розрядність

Фізіологічні сигнали (наприклад, ЕКГ) зазвичай оцифровуються через спеціальні пристрої (АЦП — аналого-цифрові перетворювачі), які перетворюють безперервний аналоговий сигнал у дискретний цифровий. Такий перетворювач працює на основі певної **розрядності (N)**, що визначає кількість рівнів квантування: 2^N, де N — кількість бітів на вибірку. Наприклад, **12-бітовий** пристрій може подати сигнал у 4096 дискретних рівнях (від 0 до 4095). Зі зростанням N підвищується точність представлення сигналу, але також збільшується і обсяг необхідної пам’яті для зберігання кожної вибірки.

Від цифрового значення до фізичних одиниць

Для проведення ефективного аналізу дослідникам важливо не просто оперувати "сирими" цілими значеннями сигналів, а перевести їх у **фізичні одиниці**, наприклад — мілівольти у випадку ЕКГ. Це перетворення виконується за допомогою калібрувальних параметрів, які містяться у файлі .hea. Саме вони визначають масштабний коефіцієнт, що дозволяє зіставити кожне цифрове значення з реальним фізичним аналогом.

Ці обчислення можуть бути виконані у високоточних середовищах програмування, таких як Python, MATLAB або C. У таких середовищах для збереження обчислених значень використовуються **змінні з плаваючою точкою подвійної точності (64 біти)**, які забезпечують надзвичайно високу точність подання чисел (понад $2^{64} = 1.8447e + 19$ можливих значень). Наприклад, в Python тип float64 дозволяє оперувати десятковими значеннями з великою точністю, що критично важливо для біомедичних аналізів.

Однак варто пам’ятати: завантаження значень у 64-бітне середовище не покращує початкову точність сигналу. Фізична точність обмежується характеристиками пристрою збору даних (розрядністю, частотою дискретизації тощо).

Суть "фізичних" значень

Терміни “фізичні значення” або “значення в фізичних одиницях” позначають ті цифрові значення, які скориговані так, щоб відповідати реальним вимірюванням у медичних або наукових одиницях. Попри те, що всі дані в комп’ютері є дискретними, сучасні засоби дозволяють досягати наближення, що є практично ідентичним безперервному, завдяки використанню 64-бітних чисел з плаваючою точкою.

Таким чином, PhysioNet забезпечує гнучкість у представленні сигналів — з одного боку, зберігаючи їх у стиснутому, але точному бінарному форматі; з іншого — надаючи можливість отримати реальні фізичні значення для наукового аналізу, моделювання та клінічного застосування.

Наприклад, коли 15-бітний сигнал захоплюється за допомогою пристрою, такого як Physionet, він, ймовірно, буде збережений у вигляді 16-бітного сигналу. Кожен 16-бітний блок зберігає значення між -2^{15} і $2^{(15-1)}$. З використанням коефіцієнта підсилення та зміщення, вказаного в заголовку для кожного каналу, можна отримати вихідний фізичний сигнал для подальшої обробки. Якщо ми знаємо, що сигнал мав лише 15 біт точності під час запису, то зберігати його як цілі числа в 16-бітному файлі разом із невеликим текстовим заголовком має більше сенсу, ніж виділяти в чотири рази більше місця для зберігання того ж сигналу у 64-бітному форматі. Оскільки пристрій захоплення працював лише з 15 бітами, додатковий простір для збереження значень, що виходять за межі цих бітів, буде витрачено даремно і не додасть точності сигналу. Це можна порівняти з використанням 5TB проти 20TB простору для зберігання тих самих даних. Обидва формати можуть зберігати сигнали, які містять зразки двох або більше сигналів, що зберігаються послідовно. Ці файли можуть бути у форматах .edf або .csv.

Формат .edf (Європейський формат даних) є стандартним для зберігання та обміну медичними часовими рядами. Він відкритий і не прив'язаний до конкретної платформи, що дозволяє зберігати дані з різних пристроїв у форматі, який може бути використаний незалежними програмами для аналізу. EDF підтримує багато каналів та різні частоти дискретизації для кожного сигналу, і дані зберігаються як 16-бітні цілі числа. Формат був розроблений в 1992 році і широко використовується для зберігання даних полісомнографії (PSG), а також для електроенцефалографії (ЕЕГ), електрокардіографії (ЕКГ) та інших медичних досліджень.

EDF+ (2003 рік) є розширенням формату EDF і підтримує додаткові можливості, такі як кодування розривних записів і анотацій у форматі UTF-8. Він залишається сумісним з EDF, і файли EDF можна відкривати також з підтримкою EDF+.

Формат .csv (Comma-Separated Values) – це текстовий файл, у якому значення розділяються комами. Він використовується для збереження табличних даних, таких як числа та текст. Формат не є повністю стандартизованим, оскільки існують варіації, що використовують різні роздільники, наприклад, крапки з комою або табуляції, замість коми.

Файл заголовка .hea містить інформацію про формат сигналу і посилається на відповідні файли даних та їх атрибути. Файли заголовків містять рядки кодування ASCII, що визначають ім’я запису, кількість сегментів і сигналів.

Файли анотацій .atr можуть бути у двох форматах: MIT та AHA DB, і використовуються для онлайн-анотування файлів. Формати є бінарними, але можуть бути прочитані без декодування.

Таким чином, PhysioNet є зручним ресурсом для збереження та обробки сигналів, що дозволяє стандартизувати сигнали для подальшого аналізу та забезпечує зручність обміну даними між дослідниками.

2.6.3. Мова програмування Python

Python – це інтерпретована мова програмування високого рівня, призначена для загального використання. Основна філософія Python акцентує увагу на читабельності коду за допомогою чіткої структури з використанням пробілів. Мова і її об'єктно-орієнтований підхід дозволяють програмістам писати зрозумілий і логічний код для проектів різного масштабу. Python є динамічною мовою з автоматичним управлінням пам'яттю та збором сміття. Вона підтримує різні парадигми програмування, включаючи процедурне, об'єктно-орієнтоване та функціональне програмування. Python був створений в кінці 1980-х як наступник мови ABC. Випуск версії Python 2.0 у 2000 році приніс нововведення, такі як розуміння списків і система збору сміття, яка здатна обробляти цикли еталонування. Python 3.0, випущений у 2008 році, став значною ревізією, не сумісною з попередніми версіями, через що багато програм на Python 2 потребували змін для роботи на Python 3. Оскільки Python 2 був широко використовуваний, підтримка його останньої версії (2.7) тривала до 2020 року. Розробник Python, Гвідо ван Россум, керував проектом до 2018 року, після чого функції керівництва були передані п’ятиособовій раді.

Інтерпретатори Python доступні для багатьох операційних систем, а підтримку мови забезпечує велика спільнота програмістів через відкритий проект CPython. Python Software Foundation, некомерційна організація, відповідає за розвиток Python і CPython.

Python є мовою, яка підтримує кілька парадигм програмування. Об'єктно-орієнтоване та структуроване програмування підтримуються повністю, а також доступні функції для функціонального програмування і аспектно-орієнтованого програмування. Багато інших парадигм підтримуються за допомогою розширень, таких як розробка за контрактом і логічне програмування.

Мова має динамічну типізацію і використовує поєднання підрахунку посилань та збору сміття для управління пам'яттю. Вона також підтримує динамічну прив'язку імен, що дозволяє визначати методи та змінні під час

виконання програми. Python також пропонує підтримку функціонального програмування, запозиченого з Lisp, і має функції фільтрації, мапи та зменшення, а також спеціальні типи даних, такі як списки, словники та множини. Стандартна бібліотека включає модулі itertools і functools, які реалізують функціональні інструменти, запозичені з Haskell і Standard ML.

Основна філософія мови узагальнена в документі Zen of Python (PEP 20), де є афоризми, які вказують на важливість ясності та простоти в програмуванні. Наприклад, такі вислови:

- ✓ Гарний краще, ніж потворний;
- ✓ Явний краще, ніж неявний;
- ✓ Простий краще, ніж складний;
- ✓ Комплексний краще, ніж складний;
- ✓ Читабельність важлива.

Python розроблений з фокусом на розширюваність, що дозволяє програмістам легко додавати нові функції до існуючих додатків. Цей підхід зробив Python популярним інструментом для розширення можливостей програм. Ван Россум, розчарувавшись в іншій мові, ABC, прагнув до створення мови з малим ядром і великою стандартною бібліотекою, а також з можливістю розширення.

Python ставить акцент на простоту та чіткість синтаксису, надаючи програмістам свободу вибору стилю кодування. Відмінністю від Perl, де існує багато способів вирішення задачі, є девіз Python: «Повинно бути тільки одне очевидне і бажано єдине рішення».

Розробники Python намагаються уникати передчасної оптимізації, зберігаючи простоту коду, навіть якщо це може трохи знизити швидкість. У разі потреби у швидкості критичних функцій програмісти можуть використовувати модулі на C або компілятори, такі як PyPy або Cython, щоб прискорити виконання Python-коду.

Важливою метою розробників Python є збереження гумору в мові. Це відображено навіть у її назві, яка є данину британській комедійній групі Monty

Python. Окрім того, гумор часто прослідковується в навчальних посібниках та довідкових матеріалах, де можна зустріти приклади, що посилаються на "спам" та Пасхальні яйця (як в одному з ескізів Monty Python). Також існує стандартне використання змінних, таких як foo і bar.

У спільноті Python з’явився термін *pythonnists*, який може позначати різні аспекти стилю програмування. Коли говорять, що код є "pythonic", це означає, що він використовує типові Python-ідоми, добре виглядає, природно читається, відображає мінімалістичний підхід і акцент на читабельності. З іншого боку, код, який важко зрозуміти або який виглядає як погана адаптація з іншої мови програмування, називають непітонним.

Python використовує динамічну типізацію, однак вимагає чіткої відповідності типів під час операцій. Це означає, що типи не перевіряються під час компіляції, але операції з об'єктами можуть призвести до помилок, якщо типи несумісні (наприклад, спроба додати число до рядка буде викликати помилку, а не спробу привести типи до одного формату).

Python дозволяє створювати власні типи через класи, які часто використовуються для об'єктно-орієнтованого програмування. Нові екземпляри класів створюються шляхом виклику класу (наприклад, SpamClass() або EggsClass()). Класи є екземплярами метакласу, що дозволяє реалізовувати метапрограмування та рефлексію.

До Python 3.0 існувало два типи класів: старий стиль і новий стиль. Основна відмінність між ними полягала в тому, чи успадковується об'єкт класу від базового класу (об'єкта). З версії Python 3.0 старий стиль класів був вилучений, і всі класи стали класами нового стилю.

Довгостроковою метою є підтримка поступового додавання статичних типів. З Python 3.5 мова дозволяє задавати статичні типи, хоча вони не перевіряються в стандартній реалізації CPython. Однак, є експериментальний додаток для статичної перевірки типів — mypy, який підтримує перевірку типів на етапі компіляції.

Розвиток Python відбувається переважно через процес пропозицій Python (PEP), що є основним механізмом для пропозиції нових функцій, збору відгуків від спільноти та документування рішень щодо дизайну мови. Стиль кодування Python визначений у PEP 8. Найважливіші PEP обговорюються спільнотою і самим Гвідо ван Россумом, який, незважаючи на свою відставку з керівної посади, все ще є доброзичливим диктатором Python.

Покращення мови реалізуються через еталонну реалізацію CPython. Основним форумом для обговорення розвитку мови є список розсилки python-dev, а конкретні питання обговорюються в системі відстеження помилок Roundup, що зберігається на python.org. Раніше розробка велась в репозиторії Mercurial, але з січня 2017 року проект перейшов на GitHub. На рисунку 2.3 представлена ієрархія типів даних.

Публічні релізи CPython складаються з трьох типів версій, які відрізняються змінами в частині версії, що збільшується.

- **Несумісні версії**, у яких очікується, що код буде зламаний, і його потрібно буде адаптувати вручну. У цьому випадку збільшується перша частина номера версії. Такі випуски трапляються рідко — наприклад, версія 3.0 була випущена через 8 років після версії 2.0.
- **Основні (функціональні) випуски**, які відбуваються приблизно кожні 18 місяців, підтримують сумісність з попередніми версіями, але додають нові функції. У цих випадках збільшується друга частина номера версії. Кожна основна версія отримує виправлення помилок протягом кількох років після її виходу.
- **Малі випуски**, які не вводять нових функцій, а лише виправляють помилки. Вони виходять приблизно кожні три місяці і часто містять виправлення безпеки. У таких версіях збільшується третя частина номера версії.

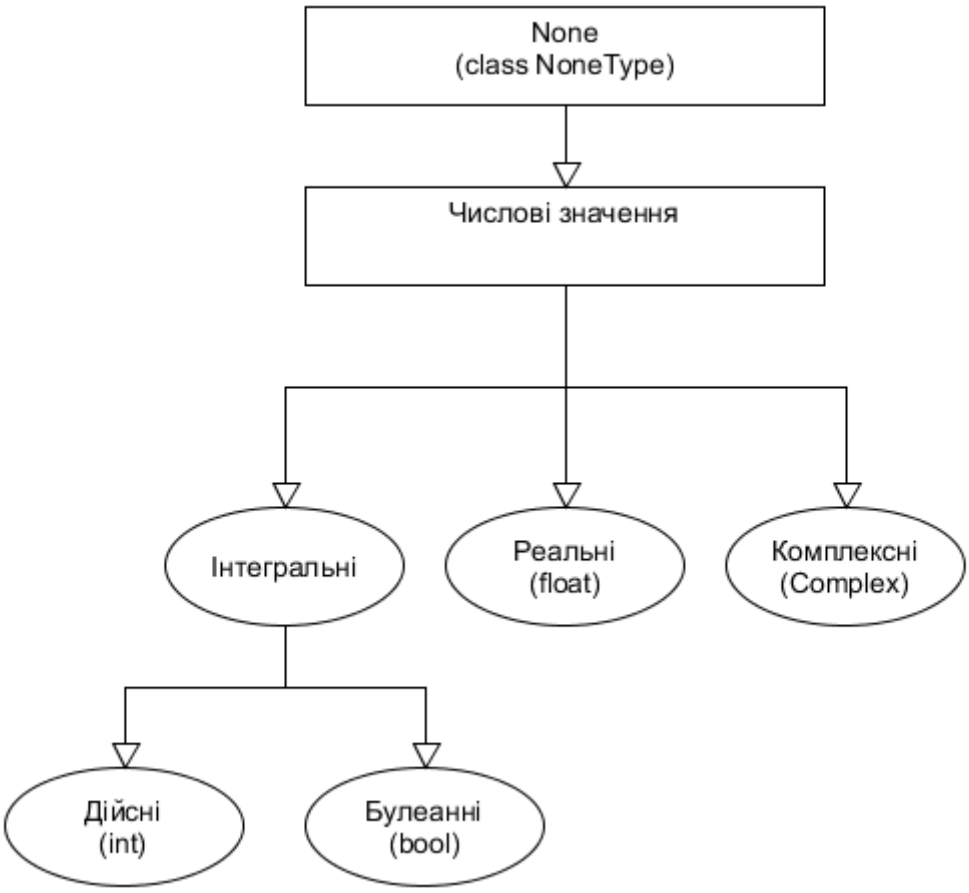


Рисунок 2.5. Відображення рівнів ієрархії типів даних.

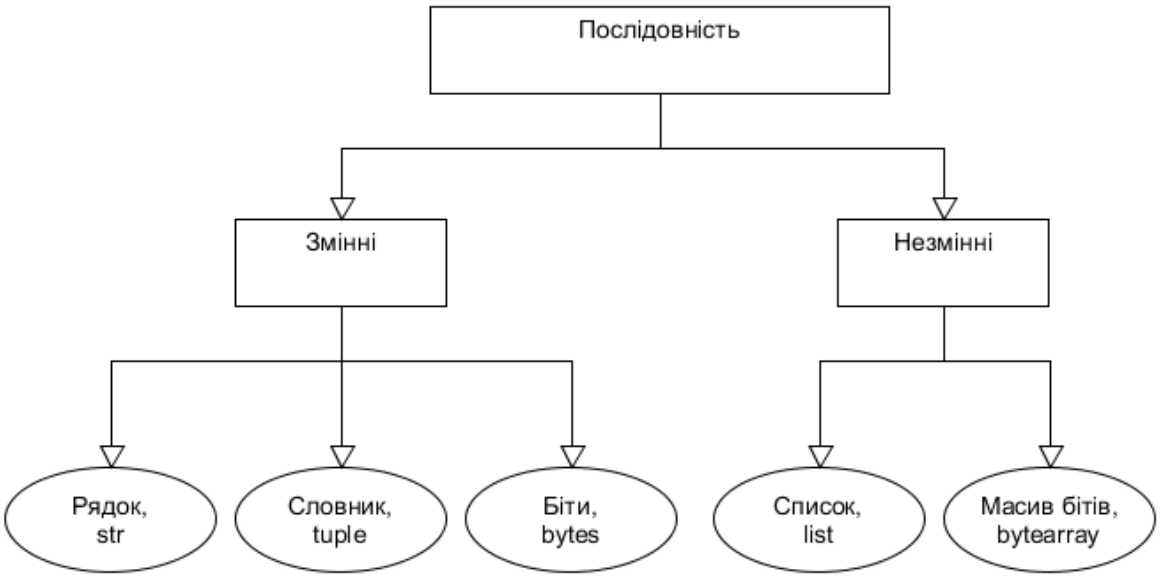


Рисунок 2.6. Ієрархія послідовностей.

Велика стандартна бібліотека Python, що є однією з її найбільших переваг, надає безліч інструментів для вирішення різних завдань. Для програм, що працюють в Інтернеті, підтримується багато стандартних форматів і протоколів. Стандартна бібліотека включає модулі для створення графічних інтерфейсів користувача, підключення до реляційних баз даних, генерації псевдовипадкових чисел, арифметики з довільною точністю, роботи з регулярними виразами та модульного тестування.

Деякі частини стандартної бібліотеки Python слідують специфікаціям (наприклад, реалізація інтерфейсу шлюзу веб-сервера (WSGI) в модулі wsgiref відповідає вимогам PEP 333), але більшість модулів не є стандартами і визначаються лише кодом, внутрішньою документацією та наявними тестами. Оскільки більшість стандартної бібліотеки є кросплатформною, лише деякі модулі вимагають змін або переписування для підтримки різних платформ.

2.6.4. Додаткові бібліотеки

Середовище Python є величезним і багатофункціональним у різних аспектах. Початок у цьому "лісі з відкритим вихідним кодом" може бути складним навіть для досвідчених програмістів, адже мова постійно розвивається, а бібліотеки та технології з'являються швидко. Python вже включає велику кількість якісних бібліотек, що є частиною його стандартної бібліотеки. Однак є й інші бібліотеки, які заслуговують на увагу при розробці конкретних програмних модулів.

При виборі рішень та підходів для реалізації програмного додатку було прийнято рішення використовувати додаткові бібліотеки для досягнення потрібного функціоналу.

Бібліотека Matplotlib

Однією з основних бібліотек, яка була використана для розробки, є **Matplotlib**. Це потужна бібліотека для створення 2D графіків та візуалізації масивів даних в Python. Хоча вона спочатку була створена для емуляції

графічних команд MATLAB, вона не залежить від MATLAB і може бути використана в Python в об'єктно-орієнтованому стилі. Matplotlib, хоч і написано переважно на чистому Python, використовує бібліотеки, такі як **NumPy**, щоб забезпечити хорошу продуктивність навіть для великих обсягів даних.

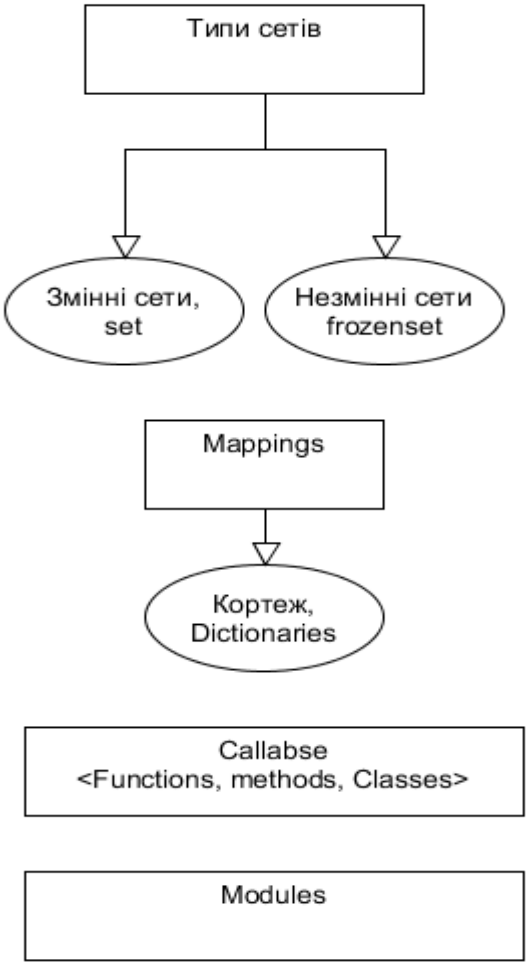


Рисунок 2.7. Ієрархія сетів, карт, класів та модулів.

Matplotlib дозволяє створювати графіки та діаграми із мінімальним обсягом коду. Якщо необхідно побудувати гістограму або інший графік, немає потреби в створенні складних об'єктів, методів виклику чи властивостей, що значно полегшує процес розробки.

З часом, після використання MATLAB для аналізу даних та візуалізації, я вирішив перейти на Python через його гнучкість і потужність, хоча спочатку

мав труднощі з пошуком належної бібліотеки для 2D-візуалізації. Після ретельного аналізу я обрав Matplotlib, оскільки вона відповідає моїм вимогам:

- Якість публікацій (текст має бути чітким, згладженим).
- Можливість експорту в PostScript для включення в документи.
- Підтримка інтеграції в графічний інтерфейс користувача.
- Легкість у розумінні та розширенні коду.
- Простота створення графіків.

Код Matplotlib розподілений на три основні частини:

1. **Pylab** — набір функцій, які дозволяють створювати ділянки, подібні до MATLAB.
2. **API Matplotlib** — набір класів, які управляють графіками, текстом та іншими елементами.
3. **Backend** — відповідає за виведення результатів на різні пристрої, наприклад, для виведення в PostScript, SVG або PNG.

Matplotlib використовується в різних сценаріях — від створення статичних файлів для друку до інтерактивної роботи з графічними інтерфейсами на різних платформах.

Бібліотека PyQt5

PyQt об'єднує платформу Qt C++ для створення міжплатформних додатків з Python. Qt — це потужний інструмент, який включає не лише віджети для графічних інтерфейсів користувача, а й інші компоненти, такі як абстракції для мережі, баз даних, мультимедійних додатків, підтримки XML, OpenGL та інше.

PyQt підтримує використання механізму сигналів і слотів для безпечного обміну даними між об'єктами, що полегшує створення модульних і повторно використовуваних компонентів.

Бібліотека PyQt5 не є зворотно сумісною з PyQt4. Основні відмінності включають реорганізацію модулів і відмову від застарілих функцій. Однак адаптувати старий код для нової версії не є складним завданням.

Бібліотека requests

Бібліотека **requests** є стандартом для створення HTTP-запитів у Python. Вона значно спрощує роботу з HTTP-запитами, надаючи простий та зрозумілий API для взаємодії з веб-сервісами.

За допомогою цієї бібліотеки можна:

- Створювати запити за допомогою основних методів HTTP.
- Налаштовувати заголовки та параметри запитів.
- Перевіряти відповіді на запити.
- Проводити автентифікацію для запитів.
- Налаштовувати параметри для уникнення уповільнення програми або помилок.

Висновки.

Для реалізації поставлених задач була обрана мова програмування **Python**. Додатково були відібрані відповідні бібліотеки:

- **PyQt5** для створення графічного інтерфейсу користувача.
- **Matplotlib** для візуалізації даних.
- **wfdb** для роботи з даними у форматах .edf та .csv, розроблена ресурсом PhysioNet.

Ці бібліотеки разом забезпечують функціональність, необхідну для ефективної роботи програми.

3. РОЗРОБКА АЛГОРИТМІЧНОГО, ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Математичне забезпечення

Математична модель біотехнічної системи візуалізації динаміки змін серцевого ритму є формалізованим описом основних етапів перетворення електрокардіографічного (ЕКГ) сигналу, починаючи від його реєстрації та завершуючи отриманням аналітичних показників, що характеризують функціональний стан серцево-судинної системи. Ця модель дозволяє визначити алгоритми обробки, оцінити їх ефективність та спрогнозувати поведінку системи.

3.1.1. Модель вихідного ЕКГ-сигналу

ЕКГ-сигнал x(t), отриманий від поверхні тіла пацієнта, можна представити як суму корисного сигналу s(t) (власне, біоелектричної активності серця) та адитивного шуму n(t):

$$x(t)=s(t)+n(t)$$

де:

s(t) – корисний ЕКГ-сигнал, що містить характерні зубці P, QRS, T та відображає електричну активність міокарда. Цей сигнал є квазіперіодичним, але його параметри (амплітуда, тривалість, морфологія, інтервали між зубцями) змінюються в часі залежно від фізіологічного стану.

n(t) – сукупність різноманітних шумів та артефактів, що включають:

Мережеві перешкоди (50/60 Гц) представлені синусоїдальним шумом з частотою електричної мережі. М'язові артефакти – це високочастотні імпульсні шуми, зумовлені активністю скелетних м'язів. Дрейф ізолінії за рахунок низькочастотного шуму, пов'язаний з диханням, рухом пацієнта, зміною електродного опору. Апаратні шуми - це власні шуми сенсорів та підсилювачів.

3.1.2. Модель фільтрації сигналу

Після аналогового підсилення, отриманий аналоговий сигнал $x_A(t)$ проходить дискретизацію за допомогою АЦП. Дискретизований сигнал $x_{[k]}$ на k-му відліку може бути поданий як:

$$x[k]=x_A(k \cdot Ts)$$

де T_s – інтервал дискретизації.

Для видалення шумів до дискретизації та після неї застосовуються цифрові фільтри. Процес фільтрації може бути описаний як застосування оператора F до вхідного сигналу:

$$x_{фільтр}[k]=F(x[k])$$

де F – сукупність фільтруючих операторів:

Високочастотний фільтр (ФВЧ) для усунення дрейфу ізолінії. Це може бути фільтр Баттерворта або Чебишева 1-2 порядку з частотою зрізу 0.5 Гц. Його передатна функція у Z-області HHPF(z)

Низькочастотний фільтр (ФНЧ) для усунення високочастотних шумів (м'язових артефактів). Це може бути фільтр Баттерворта 4-6 порядку з частотою зрізу 30-40 Гц. Його передатна функція у Z-області HLPF(z)

Режекторний фільтр для придушення мережевих перешкод (50 Гц). Це може бути IIR-фільтр другого порядку з вузькою смугою пропускання на частоті 50 Гц. Його передатна функція у Z-області $H_{Notch}(z)$

Таким чином, фільтрований сигнал можна представити як:

$$x_{фільтр}[k]=H_{LPF}(z) \cdot H_{HPF}(z) \cdot H_{Notch}(z) \cdot x[k]$$

3.1.3. Модель детекції R-зубців

Після фільтрації необхідно виділити R-зубці, які є основними референсними точками для подальшого аналізу серцевого ритму. Один з поширених методів, алгоритм Панна-Томпкінса, включає наступні етапи:

Диференціювання виконується для підкреслення крутих схилів QRS-комплексів та придушення інших зубців. Оператор диференціювання D:

$$y_1[k]=T_{s1}$$

$$(x_{фільтр}[k]-x_{фільтр}[k-1])$$

Піднесення до квадрату підсилює позитивні значення та робить від'ємні значення позитивними. $y_2[k]=(y_1[k])^2$

Інтегрування (ковзне вікно) усереднює значення за вікном для отримання інформації про енергію QRS-комплексу.

$$y_3[k] = N_1 \sum_{i=0}^{N-1} y_2[k-1]$$

Де N – розмір вікна інтегрування, що відповідає приблизній ширині QRS-комплексу.

Порогове визначення: Виявлення піків сигналу $y_3[k]$, що перевищують адаптивний поріг $T_h[k]$. Позиції цих піків відповідають R-зубцям. Адаптивний поріг обчислюється на основі амплітуди виявлених QRS-комплексів та рівня шуму.

R_k={1, якщо $y_3[k]>T_h[k]$ і $y_3[k]$ є локальним максимумом

0, в іншому випадку

3.1.4. Модель аналізу варіабельності серцевого ритму (ВСР)

Після детекції R-зубців отримуємо послідовність R-R інтервалів (NN-інтервалів) NNi, де i – номер інтервалу. Це є часовим рядом, на основі якого розраховуються показники ВСР.

Показники часової області:

SDNN (Standard Deviation of NN intervals): Стандартне відхилення всіх NN-інтервалів. SDNN=M−11∑i=1M(NNi−NN)2 Де M – кількість інтервалів, NN – середнє значення NN-інтервалів.

RMSSD (Root Mean Square of Successive Differences): Корінь квадратний із середнього значення квадратів різниць послідовних NN-інтервалів. $RMSSD=M-11\sum i=1M-1(NNi+1-NNi)2$

Показники частотної області

Спектральний аналіз. Застосовується швидке перетворення Фур'є (БПФ) до ряду NN-інтервалів (після ресемплінгу до рівномірної частоти) для отримання спектральної щільності потужності P(f). $P(f)=|FFT(NNresampled)|2$

Виділяються потужності у стандартних частотних діапазонах:

$VLF=\int 00.04P(f)df$ (дуже низькі частоти)

$LF=\int 0.040.15P(f)df$ (низькі частоти)

$HF=\int 0.150.4P(f)df$ (високі частоти)

LF/HF

Нелінійні показники (наприклад, Entropy):

Sample Entropy (SampEn): Оцінює складність та нерегулярність часового ряду NN-інтервалів. $SampEn(m,r,N)=-\ln(BA)$ Де A – кількість пар векторів довжини m+1, які знаходяться в межах r, а B – кількість пар векторів довжини m, які знаходяться в межах r. m – розмір шаблону, r – поріг подібності.

3.1.5. Модель візуалізації даних

Візуалізація є відображенням отриманих результатів у графічній формі. Це не математична модель у традиційному розумінні, а представлення даних, що ґрунтується на математичних розрахунках.

Відображення ЕКГ-кривої. Послідовний вивід відліків $x_{фільтр}[k]$ на графік залежності амплітуди від часу.

Графіки ВСР .

Діаграма Пуанкаре - це побудова точок (NNi,NNi+1). Її форма характеризується стандартними відхиленнями SD1 (короткострокова

варіабельність) та SD2 (довгострокова варіабельність). SD1=0.5·Var(NNi+1 −NNi) SD2=0.5·Var(NNi+1+NNi)

Гістограма NN-інтервалів **- по**будова розподілу частоти появи NN-інтервалів у певних діапазонах.

Спектрограмою буде візуалізація P(f) для різних частотних діапазонів.

3.1.6. Математична модель у контексті Machine Learning

Якщо система включає розпізнавання аритмій, то це також є частиною математичної моделі. Для класифікації аритмій можна використовувати згорткові нейронні мережі (CNN). Математично, CNN виконує послідовність згорткових операцій, функцій активації та операцій пулінгу:

$$Y=f(W*X+b)$$

де:

X – вхідний ЕКГ-сигнал (або його ознаки).

W – ядро згортки (фільтр).

f – функція активації (наприклад, ReLU).

b – зсув.

$*$ – операція згортки.

Y – вихід карти ознак.

Далі, вихід згорткових шарів передається до повністю зв'язаних шарів, які виконують класифікацію на основі функції втрат (наприклад, крос-ентропії) та оптимізуються за допомогою градієнтного спуску.

Представлена математична модель охоплює ключові етапи обробки ЕКГ-сигналу, від його зашумленого аналогового вигляду до отримання числових показників ВСР та їх графічного представлення. Застосування цих моделей дозволяє формалізувати процес функціонування біотехнічної системи, забезпечити точність розрахунків та достовірність отриманих результатів, що є фундаментальною основою для розробки ефективного програмного забезпечення та надійної апаратної частини.

3.2. Розробка програмного додатку

3.2.1. Проектування програмного додатку

Хоча програмний застосунок для візуалізації медичних показників не потребує складного проєктування, задля оптимізації часу реалізації було проведено попередній аналіз. На основі результатів цього аналізу були створені функціональна модель системи та прототипи інтерфейсу користувача.

Основні завдання на цьому етапі включали:

- аналіз результатів попереднього дослідження та визначення технічних і функціональних обмежень;
- виявлення потенційно критичних елементів, які можуть вплинути на стабільність і точність роботи системи;
- формування остаточної архітектури програмного продукту;
- оцінка доцільності використання готових рішень та сторонніх бібліотек;
- проєктування ключових складових системи, таких як моделі даних, логіка функціонування та структура програмного коду;
- вибір відповідного середовища розробки та інструментів, що забезпечать надійність і зручність реалізації;
- затвердження дизайну інтерфейсу, зокрема визначення візуальних компонентів для відображення медичних даних.

3.2.2. Попередній аналіз

У межах підготовки було проведено дослідження наявних рішень іноземних розробників, зокрема аналогічних програмних продуктів. Це дало змогу:

- визначити критичні компоненти системи, які потребують особливої уваги під час реалізації;
- створити загальну блок-схему логіки роботи застосунку;

- підібрати відповідні ресурси — бібліотеки, середовище розробки та API для інтеграції.

  Виявлені критичні компоненти системи:
- перевірка підключення до інтернету: деякі функції програми залежать від наявності мережевого доступу (наприклад, завантаження або синхронізація даних);
- обробка даних у правильному форматі: оскільки медичні дані можуть надходити в різних форматах (таких як .edf, .csv), необхідно забезпечити їх правильне зчитування та інтерпретацію.

3.2.3. Розробка блок-схеми логіки програмного додатку

Блок-схема — це поширений графічний спосіб представлення процесів чи алгоритмів, де кожен крок показується окремим блоком, а зв’язки між ними демонструють послідовність виконання дій.

Під час підготовки було створено початкову блок-схему, яка ілюструє основну логіку функціонування застосунку. Схема представлена нижче (рис. 3.1) та побудована з використанням стандартів уніфікованої мови моделювання UML.

Ця блок-схема описує процес обробки даних, що починається з перевірки наявності інтернет-з'єднання. Якщо з'єднання з мережею інтернет встановлено, процес переходить до вибору та завантаження необхідного набору даних. Після цього відкривається файлове вікно, користувач обирає файл, відбувається масштабування графіку, і результат зберігається. У випадку відсутності інтернет-з'єднання, процес одразу переходить до відкриття файлового вікна, вибору файлу, масштабування графіку та збереження результату. Обидва шляхи обробки даних зливаються в одній точці перед завершенням всього процесу.

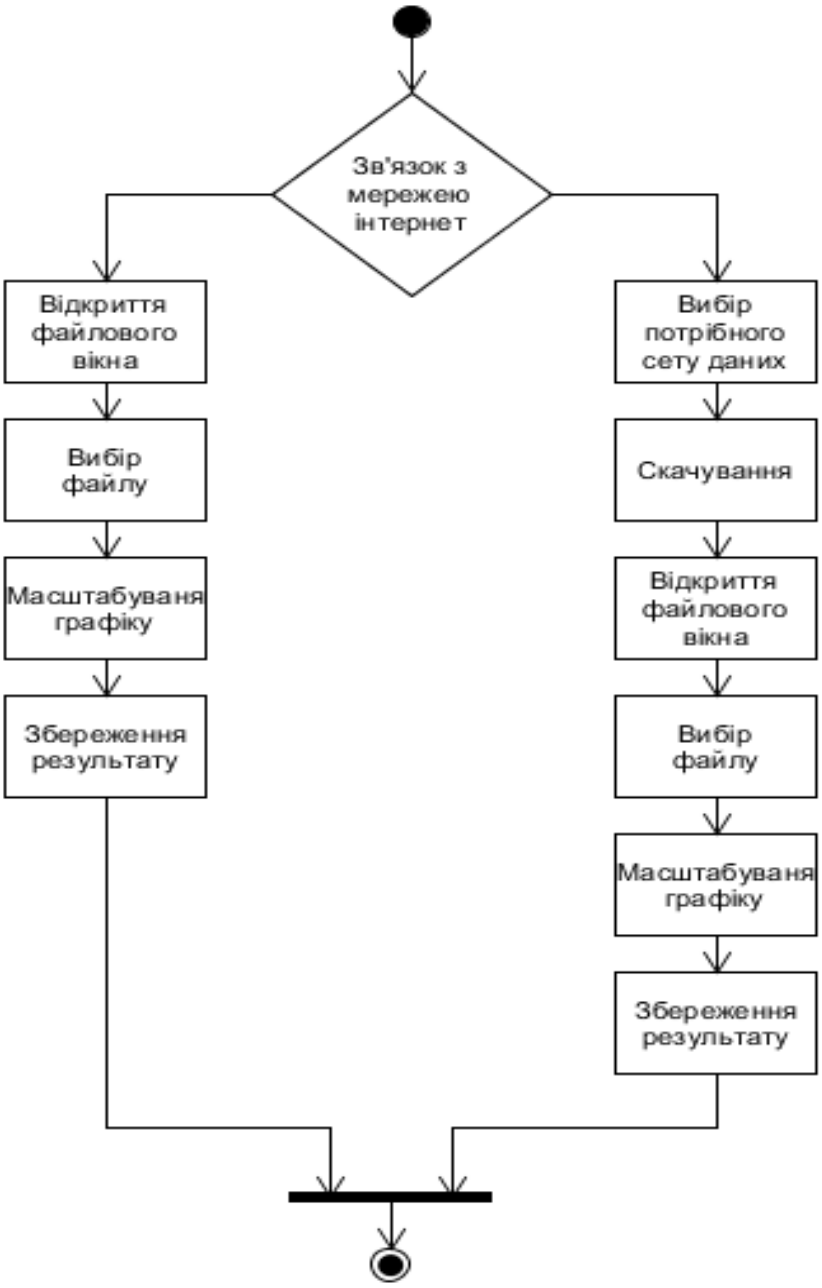


Рисунок 3.1. Блок-схема роботи прогрманого додатку.

3.3. Програмні засоби для візуалізації динаміки змін серцевого ритму

3.3.1. Середовище розробки

Для реалізації завдань, поставлених у межах цієї роботи, я використовував мову програмування Python. Як інструмент для розробки було обрано середовище Jupyter Notebook.

Jupyter Notebook — це зручна інтерактивна платформа, що дозволяє створювати документи з інтегрованим виконанням коду. Термін "Notebook" може означати як веб-додаток Jupyter, серверну частину на Python, так і сам формат файлу. Такі файли зберігаються у форматі JSON і містять послідовність комірок, що можуть включати програмний код, текстові пояснення з підтримкою Markdown, математичні формули, графіки, а також інші мультимедійні елементи. Зазвичай файли мають розширення .ipynb. На рисунку 3.10 представлено головний вигляд цього середовища.

Документи Jupyter Notebook легко конвертуються в інші формати — такі як HTML, презентації, LaTeX, PDF, ReStructuredText, Markdown або Python-скрипти. Це можна зробити через пункт меню "Завантажити як" у веб-інтерфейсі, скористатися бібліотекою nbconvert, або ж через командний рядок за допомогою команди jupyter nbconvert.

Для швидкого перегляду Jupyter-файлів у браузері існує вебсервіс NbViewer, який дозволяє завантажити публічний .ipynb файл за посиланням і миттєво відобразити його у вигляді HTML-сторінки.

Jupyter Notebook реалізує інтерактивне середовище типу REPL (Read–Eval–Print Loop), яке працює безпосередньо у браузері. Воно побудоване на основі низки популярних технологій з відкритим кодом: IPython, ØMQ, Tornado (веб-сервер), jQuery, Bootstrap та MathJax. Інтерфейс середовища представлений на рисунку 3.2.

Jupyter Notebook має можливість підключення до різноманітних обчислювальних ядер, що дає змогу працювати з багатьма мовами програмування. Зазвичай, під час встановлення використовується ядро **IPython**. Починаючи з жовтня 2014 року (версія 2.3), доступно вже 49 ядер, сумісних із Jupyter, серед яких — підтримка мов Python, R, Julia, Haskell та інших.

Інтерфейс, подібний до сучасного Jupyter Notebook, вперше з’явився в **IPython** версії 0.12 у грудні 2011 року. У 2015 році, після виходу **IPython 4.0** та **Jupyter 1.0**, проєкт отримав нову назву — **Jupyter Notebook**. Його структура

нагадує інші середовища з блокнотною організацією, такі як **Maple**, **Mathematica** та **SageMath**, що наслідують підхід, започаткований Mathematica ще у 1980-х роках. Як відзначило видання *The Atlantic*, саме популярність інтерфейсу Mathematica стала поштовхом до розвитку подібних платформ, серед яких і Jupyter, особливо на початку 2018 року.

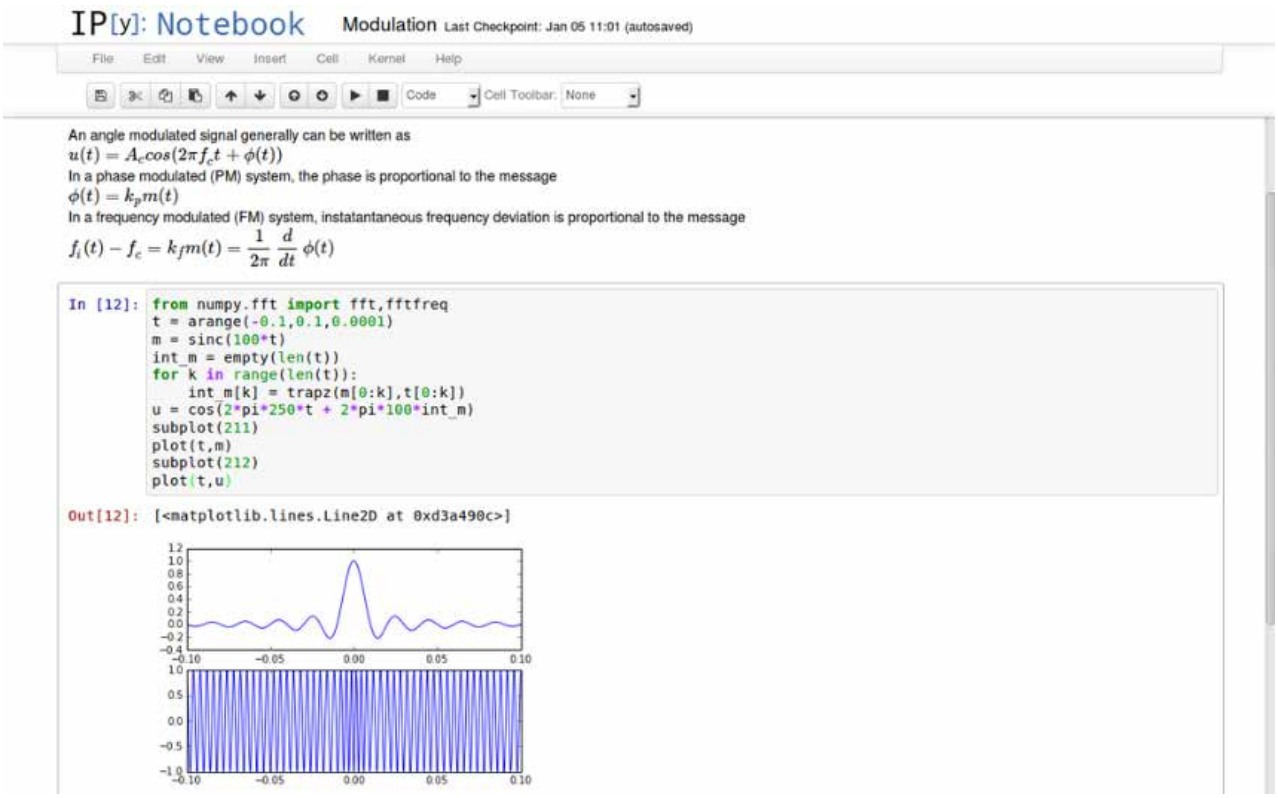


Рисунок 3.2. Графічний інтерфейс Jupiter Notebook.

Основні компоненти середовища Jupyter:

- **Ядро Jupyter** – це програмний модуль, що виконує ключові запити, зокрема запуск коду, його автозаповнення, інспекцію та формування відповідей. Взаємодія ядер із рештою компонентів системи відбувається через мережевий протокол **ZeroMQ**, що дозволяє розміщення ядер як локально, так і на віддалених машинах. На відміну від багатьох аналогів, ядро Jupyter не прив'язане до конкретного документа та може обслуговувати декілька клієнтів одночасно. Зазвичай кожне ядро працює з однією мовою, але існують і багатомовні рішення. Основним ядром за замовчуванням є **IPython**, а його

реалізація відбувається через модуль **ipykernel**. Крім того, доступні ядра для інших мов, хоча вони можуть відрізнятись якістю та функціональністю.

- **JupyterHub** – серверне рішення, яке дозволяє одночасну роботу багатьох користувачів у середовищі Jupyter Notebook. Воно забезпечує створення, адміністрування та маршрутизацію сесій до окремих інстанцій Jupyter. Хоча JupyterHub потребує серверного адміністрування, існують альтернативні сервіси, наприклад **Jupyo**, що надають хмарне розгортання багатокористувацьких Jupyter-інстанцій із мінімальними зусиллями з боку користувача.
- **JupyterLab** – це сучасна версія інтерфейсу Jupyter, яка об'єднує всі класичні інструменти Jupyter Notebook (ноутбук, термінал, редактор, файловий менеджер, розширене виведення тощо) в єдиному інтуїтивному та функціональному середовищі. Перша стабільна версія була випущена 20 лютого 2018 року, а вже в грудні того ж року JupyterLab став стандартним інтерфейсом для більшості хмарних платформ, що використовують Jupyter.

Бібліотека wfdb

Для обробки даних, що зберігаються у форматі **PhysioNet**, застосовується спеціалізована бібліотека **Waveform Database Library (wfdb)**. Вона є набором функцій, реалізованих мовою **C**, які забезпечують зручний і уніфікований доступ до цифрових сигналів — як з анотаціями, так і без них — незалежно від формату їхнього збереження.

Спочатку бібліотека була створена для роботи з електрокардіографічними (ЕКГ) базами даних, зокрема з добре відомою базою аритмій MIT-BIH (MITDB) та базою AHA, призначеною для тестування систем виявлення шлуночкових аритмій. У лютому 1990 року функціонал бібліотеки було розширено для підтримки Європейської ST бази даних (ESCDB) шляхом оновлення стандарту анотацій.

Завдяки широкому спектру можливостей, бібліотека **wfdb** є ефективним інструментом для роботи з будь-якими подібними наборами цифрових сигналів, незалежно від того, чи містять вони анотації. З часом бібліотека значно розширилася і почала підтримувати інші типи медичних даних, зокрема сигнали артеріального тиску, дихання, рівня кисню в крові (сатурації), електроенцефалографії (ЕЕГ) тощо. Серед великих баз даних, з якими вона сумісна, — MITBIH, полісомнографічна база та MGH/Marquette Foundation Waveform Database.

Отже, wfdb — це не лише інструмент для обробки ЕКГ-даних, а й універсальна платформа для доступу та аналізу різноманітних біомедичних сигналів.

3.3.2. Моделювання контекстної діаграми

Хоча контекстна діаграма є окремим видом діаграми потоків даних (DFD), вона відіграє важливу роль у процесі проектування систем, тому заслуговує на окрему увагу. На відміну від стандартної DFD, контекстна діаграма акцентує увагу на зовнішніх об'єктах (або термінаторах). Зовнішні об'єкти — це елементи, що взаємодіють із системою, але самі до неї не належать.

Контекстна діаграма для програмного забезпечення, створена з використанням **AllFusion Process Modeller**, показана на рисунку 3.3.

На цій схемі можна побачити такі вхідні елементи системи:

- Зліва — дані, які використовуються під час виконання програми;
- Зверху — набір алгоритмів, які реалізує система;
- Знизу — умови, що повинні бути дотримані для запуску програми;
- Справа — результати, які формуються після роботи програми.

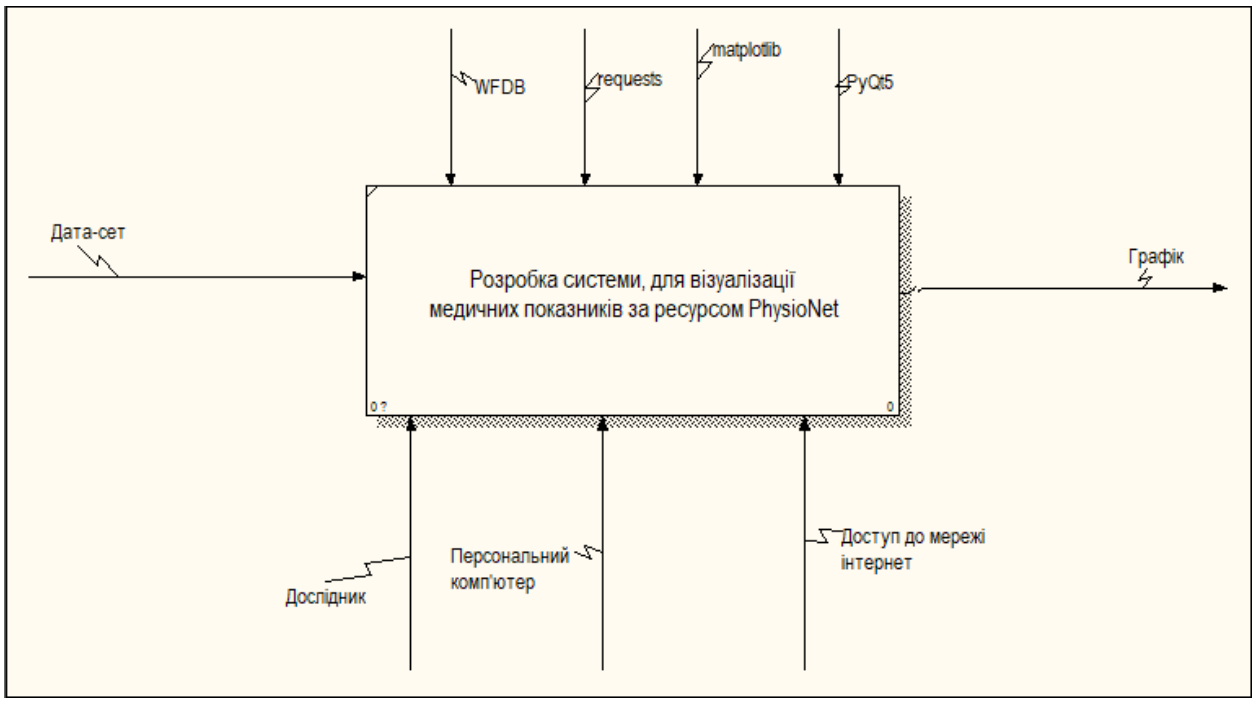


Рисунок 3.3. Контекстна діаграма

Моделювання діаграми декомпозиції першого рівня.

Декомпозиція — це процес поділу складного об’єкта, системи чи завдання на простіші, складові елементи. За допомогою діаграми декомпозиції першого рівня можна наочно представити, з яких частин складається задача «Система візуалізації медичних показників на основі ресурсу PhysioNet». У межах цієї роботи було виділено такі підзадачі:

- **Перевірка наявності підключення до інтернету** — необхідна для того, щоб користувач був поінформований про доступні функції програми на момент її використання;
- **Вибір датасету** — передбачає або вибір набору даних із запропонованих на веб-платформі, або завантаження локального файлу із збережених записів;
- **Перевірка та попередня обробка даних** — включає етапи валідації вхідних даних та їх базову обробку;
- **Візуалізація** — створення графічного представлення даних з анотацією та поясненням.

На рисунку 3.4 представлена діаграма декомпозиції першого рівня, де зображено основні чотири функціональні компоненти системи.

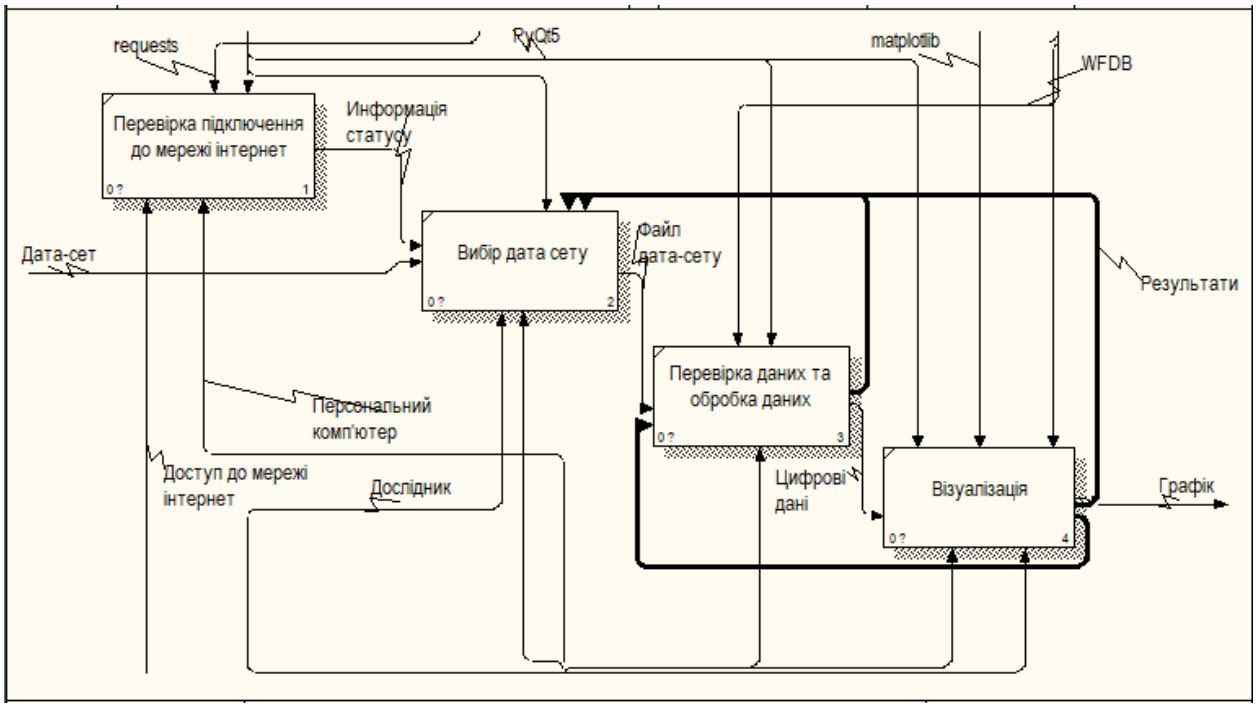


Рисунок 3.4. Діаграмма декомпозиції першого рівня.

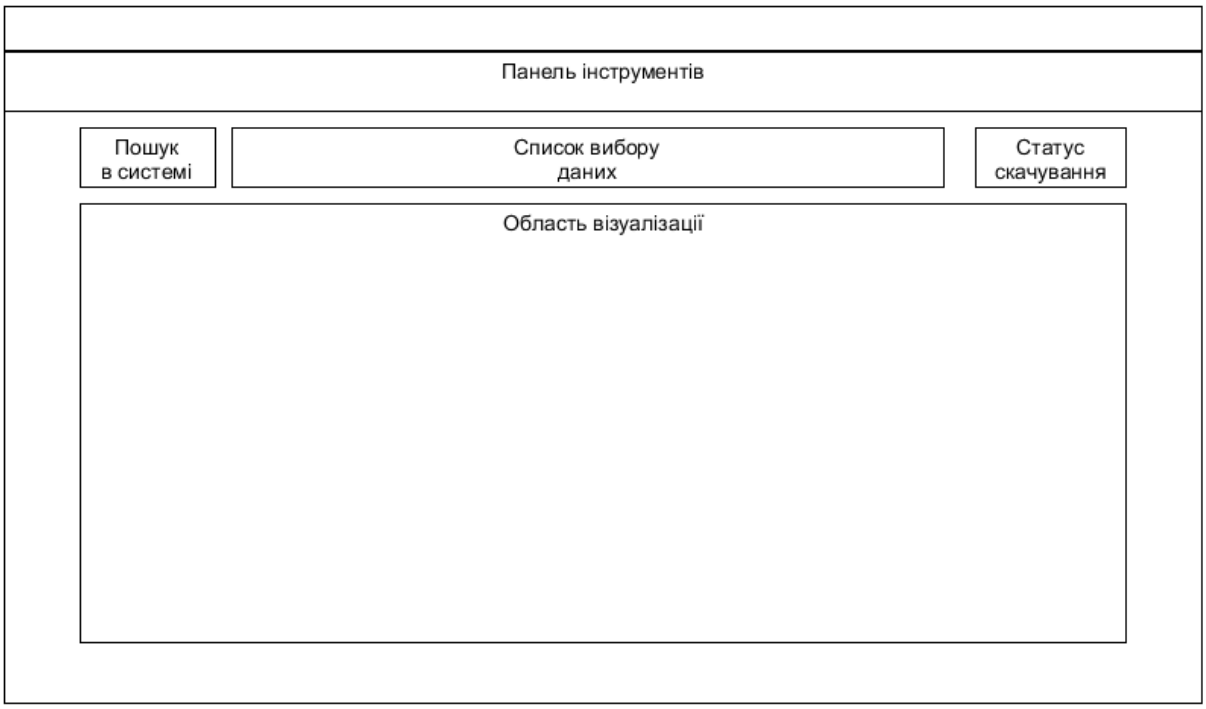
3.3.3. Моделювання графічного інтерфейсу системи.

Під час створення інтерфейсу програмного забезпечення було враховано кілька ключових вимог:

- **Зручність та інтуїтивність** – інтерфейс повинен бути простим у користуванні та зрозумілим навіть для новачків;
- **Ясність іконок на кнопках** – графічні позначки повинні точно відповідати їхній функції;
- **Інформативні текстові підписи** – назви кнопок мають бути короткими, але водночас чітко передавати суть дії;
- **Оптимальний простір для графіків** – інтерфейс має забезпечувати достатню площу для якісного відображення візуальних даних.

Приклад дизайну інтерфейсу показано на рисунку 3.5. У цьому макеті присутні такі основні компоненти:

- Панель інструментів дає змогу керувати графіком: масштабувати, переглядати окремі ділянки, створювати та зберігати скріншоти;
- Пошукова система дозволяє знайти необхідний файл на комп’ютері користувача;
- Меню вибору даних – перелік доступних датасетів з платформи PhysioNet для завантаження;
- Індикатор завантаження – показує статус імпорту даних (успішно або з помилкою);
- Область відображення графіка – частина інтерфейсу, де буде показано результати візуалізації.



- Рисунок 3.5. Макет графічного інтерфейсу.

3.3.4. Інструкція по роботі прогрманого модулю.

Після запуску програми, перед користувачем показується вікно програми. Вид вікна під час запуску програми зображений на рисунку 3.6.

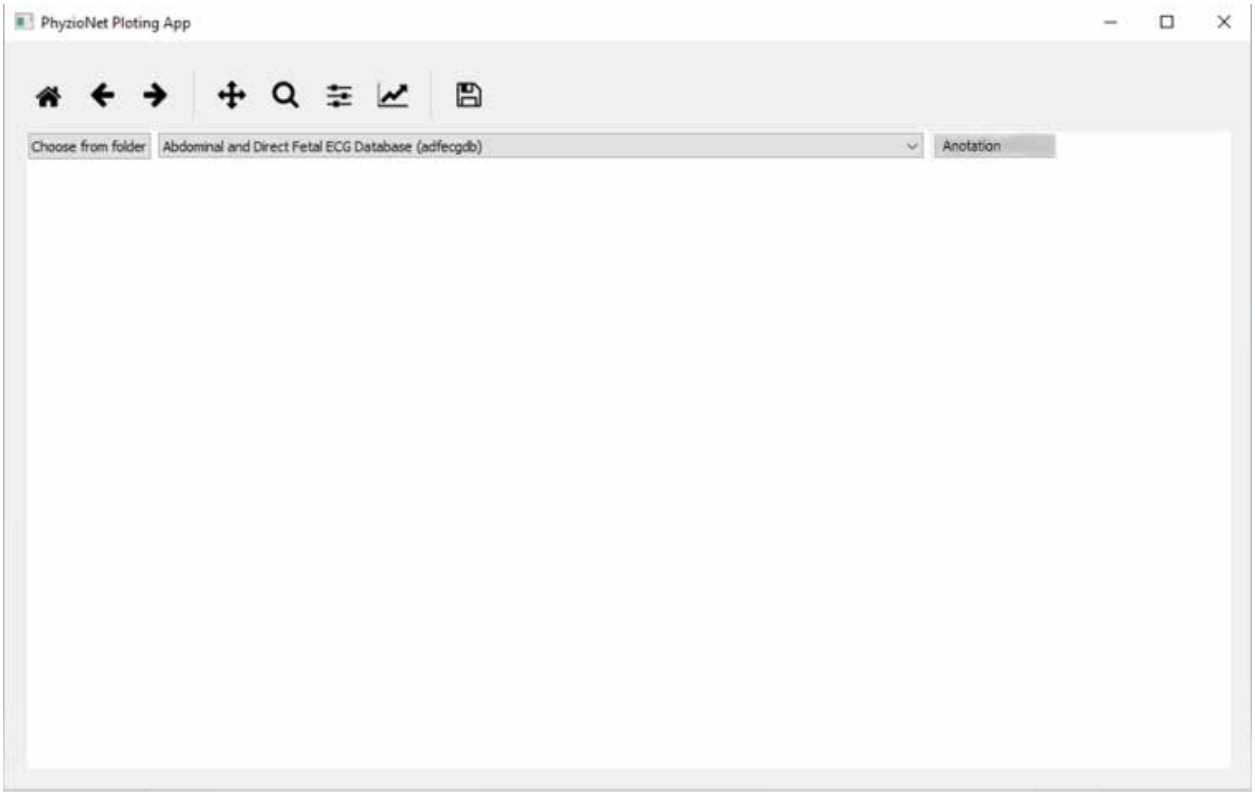


Рисунок 3.6 –Початковий стан экрану.

Бібліотека WFDB для цього використовує спеціальний текстовий рядок, що містить упорядкований перелік можливих місць пошуку цих вихідних файлів. Цей рядок називається "шлях до бази даних" або WFDBPath. Отже, якщо користувач успішно завершив взаємодію з діалоговим вікном і натиснув «ОК», програма витягує шлях до директорії, де знаходяться записи, і зберігає його у змінній WfdbPath. При цьому назва конкретного файлу, який користувач вказав у діалозі, ігнорується. Це відбувається за допомогою вбудованого в Jupiter Notebook класу Path, який дозволяє виконувати міжплатформні операції з рядками, що містять інформацію про шляхи до файлів чи каталогів. Зокрема, використовується метод GetDirectoryName(String), який повертає лише інформацію про каталог з наданого шляху.

Далі програма визначає назву самого запису, зберігаючи її в рядковій змінній rname. Це робиться за допомогою методу GetFileNameWithoutExtension(String) класу Path, який повертає назву файлу з вказаного шляху без його розширення.

Наприклад, якщо у діалоговому вікні був обраний файл за адресою C:\physionetfiles\physiobank\database\mitdb\100.dat, то до WfdbPath буде записано новий шлях C:\physionetfiles\physiobank\database\mitdb, а змінній rname буде присвоєно значення 100. Після того, як програма визначила розташування необхідних файлів, вона створює змінну record, яка є екземпляром структури Record, визначеної у WfdbLibrary. Структура – це об'єднання різнорідних змінних (навіть різних типів даних) під одним іменем. У структурі Record зберігаються змінні, що описують інформацію про записи. Список цих змінних наведено в таблиці 3.1.

Таблиця 3.1 - Список змінних структури Record

Ім'я	Тип	Опис
name	String	Назва запису.
anotinfo	string	Анотація запису.
SamplingF	Int	Частота дескритизації.
Signals	sigarray	Структура інформації сигналу.

Sigarray – являє собою структуру, змінні якої наведено нижче в таблиці 3.2. Для того, щоб заповнити створену структуру запису, викликається метод record.Open(). Ця функція відкриває поточний запис і завантажує необхідну інформацію з жорсткого диска. Далі оновлюються індикатори в рядку стану програми. Один з них відображає назву поточного відкритого запису, щоб користувачу було зручніше орієнтуватися. Інший індикатор повідомляє

користувачеві, що для подальшої роботи з програмою потрібно вибрати відповідний сигнал.

Таблиця 3.2

Змінні структури Sigarray

Ім'я	Тип	Опис
AdcResolution	Int	Тип АЦП в бітах.
AdcZero	Int	Значення АЦП в бітах, яке вказує на середину можливих значень.
Baseline	Int	Значення АЦП, яке співпадає з нулем фізичних даних.
CheckSum	Int	Сумма значень всіх відлілків сигналу.
Gain	Int	Чутливість АЦП.
Format	SignalStorage	Формат збереження файлу PhysioNet.
Number	Int	Порядковий сигнал запису.
Description	String	Опис сигналу.
Duration	Time	Тривалість сигналу.
FileName	String	Назва файлу.
Record	Record	Запис, до якого належить сигнал.
NumberOfSamples	Int	Загальна кількість відліку в сигналі.

Для побудови графіку необхідно зчитати дані сигналу, які зберігаються у файлах з розширеннями .dat, .edf або .csv. Ці дані не зчитуються автоматично при відкритті запису та заповненні інформації про сигнали, оскільки кількість відліків може бути дуже великою, і процес їх зчитування міг би вимагати значних обчислювальних ресурсів та часу. Щоб зчитати ці відліки (зразки), використовується метод record.GetSamples. Він заповнює колекцію samples структурами типу Sample, визначеними у WfdbLibrary. Детальний опис вмісту цієї структури представлений у таблиці 3.3.

Таблиця 3.3

Змінні структури Sample

Ім'я	Тип	Опис
Adu	Int	Значення звіту в одиницях АЦП.
SignalNumber	Int	Номер сигналу, якуму належить звіт.

Отже, колекція samples містить усі доступні дані записів, організовані за відповідними сигналами. Наступним кроком, який виконує функція Bopen_Click, є завантаження анотацій, пов'язаних з цим сигналом. Для цього ініціалізується змінна Annotator, що зберігає інформацію про анотації, а потім з неї видобуваються необхідні дані. Ці дані конвертуються у формат, зручний для відображення у формі. Структура Annotator детально описана в таблиці 3.4. Час, до якого належать анотації, виділяється в окрему колекцію для зручнішого відображення. На цьому функція Bopen_Click завершує свою роботу. Після її виконання доступними стають кілька глобальних змінних: record містить інформацію про сам запис, sampses зберігає значення відліків, annot представляє дані про анотації, а annottime – час цих анотацій. Оскільки користувачу пропонується вибрати сигнал для подальшої роботи, після обробника події натискання кнопки починає виконуватися обробка зміни вибору.

Таблиця 3.4

Змінні структури Annotator

Ім'я	Тип	Опис
AnotatorNumber	Byte	Номер відкритої змінної.
Aux	String	Змінна, для комментарів.
ChannelNumber	Byte	Номер каналу.
Subtype	Annotationcode	Змінна, для шуму.
Time	Time	Час.
Type	Annotation	Тип RS комплексу, яка оприділяється данною анотацією.

Насамперед, при запуску цієї функції стають активними певні елементи керування: смуга прокрутки та панель інструментів. Далі відбувається налаштування початкового стану інтерфейсу та програмних змінних. Наприклад, колір кнопок зміни масштабу стає сірим, а кнопка, що встановлює масштаб на 5 секунд, набуває червоного кольору, оскільки саме цей масштаб є стандартним при відкритті сигналу.

Максимальне значення смуги прокрутки встановлюється рівним загальній кількості відліків у сигналі. Після всіх підготовчих етапів викликається функція DrawSelected(). Вона відповідає за відображення вибраної ділянки сигналу на графіку chart1, враховуючи поточне положення повзунка смуги прокрутки та обраний користувачем масштаб.

Давайте детальніше розглянемо роботу цієї функції. Для цього необхідно ознайомитися з компонентами елемента Chart. Колекція Chart.Series містить ряди даних, що відображаються на графіку. Annotations – це колекція анотацій, які можуть бути виведені для пояснення окремих фрагментів графіку Chart. ChartAreas визначає області для малювання графіків, які займають

простір елемента Chart. Кожна така область має власні осі, що зберігаються у колекції Chart.ChartAreas.Axis.

Тепер перейдемо до розгляду функції DrawSelected(). Спочатку, під час розробки програми, було вирішено завантажувати всі відліки сигналу в Chart.Series та реалізувати прокрутку за допомогою самого компонента Chart, змінюючи ділянку осі X, що відображається. Однак, під час тестування стало очевидним, що при великій кількості даних Chart вимагає значної обчислювальної потужності для обробки графіків. Це призводило до помітної затримки між зміною положення повзунка смуги прокрутки та оновленням графіку. Тому було вирішено реалізувати прокрутку сигналу самостійно, розраховуючи необхідний інтервал і передаючи до Chart лише дані конкретного фрагмента. Для цього функція використовує змінну dx, яка показує вибраний користувачем масштаб відображення, тобто скільки секунд сигналу має бути показано.

За допомогою цієї змінної, а також враховуючи частоту дискретизації сигналу, розраховується коефіцієнт прокрутки kp – значення, на яке буде зміщуватися сигнал при натисканні на стрілку смуги прокрутки. Експериментальним шляхом було встановлено, що для комфортного сприйняття прокрутки сигналу kp має становити 1/5 від обраного масштабу. Знаючи положення повзунка та масштаб, можна розрахувати діапазон осі X, який необхідно відобразити. Межі цього діапазону зберігаються у змінних XStart (значення якої прирівнюється до положення повзунка) та XEnd. При цьому перевіряється, чи не вийшло XEnd за межі колекції відліків. Якщо це сталося, XEnd прирівнюється до загальної кількості відліків, а XStart перераховується відповідно. Далі ініціалізуються змінні для зберігання значень відліків та їхнього часу, а також колекції, що містять анотації за сигналами. Після визначення всіх змінних запускається цикл від початкового до кінцевого значення осі X. У цьому циклі заповнюється колекція точок даних. Відліки для осі Y беруться з колекції samples.

При цьому відбувається перетворення значень з одиниць вимірювання АЦП у мілівольти, а також враховується лінія нуля. Координати для осі X розраховуються та вимірюються в секундах. Коментування у програмі реалізовано таким чином: для кожної анотації проводиться вертикальна лінія. Якщо ця анотація вказує на QRS-комплекс, то це пунктирна лінія. Якщо комплекс патологічний, лінія червона; якщо нормальний, то лінія зелена. Якщо анотація несе іншу довідкову інформацію, то лінія суцільна та помаранчева. Під графіком, поруч з кожною лінією, виводиться код анотації у вигляді літери або знака, значення яких описано в стандарті PhysioNet.

Повернемося до циклу. На кожній ітерації виконується перевірка: чи присутній порядковий номер поточного відліку в колекції AnnotTime. Якщо так, створюється нова вертикальна лінія анотацій. Її властивості встановлюються відповідно до інформації про поточну анотацію. Також створюється текстова анотація, що є поясненням до лінії, розташованої в тому ж місці. Цикл завершується. Після циклу, розраховані масиви точок передаються до Chart1.Series для відображення. Створені анотації заносяться до змінної Annotations. Функція припиняє свою роботу. Зовнішній вигляд інтерфейсу програми після вибору сигналу представлений на рисунку 3.7.

Три щойно згадані функції складають ядро першої форми програми. Вони відповідають за ключові операції: відкриття сигналів, завантаження відповідних даних та їхнє графічне відображення. Інші елементи інтерфейсу є допоміжними, покращуючи зручність користування програмою. Наприклад, коли користувач натискає одну з кнопок на панелі інструментів, це змінює значення глобальної змінної dx відповідно до обраного масштабу. Кожна така функція допомагає виділити потрібну ділянку даних. Після обробки натискання цих кнопок завжди викликається функція DrawSelected(), що призводить до оновлення графічного вікна згідно з новим масштабом.

Натискання на кнопку BInfo активує метод MessageBox.Show(). Він виводить на екран діалогове вікно з описом для користувача. Це вікно є модальним, тобто воно блокує будь-які інші дії в програмі, доки користувач

його не закриє. У цьому випадку воно відображає інформацію про запис, яка міститься у файлі заголовка у вигляді коментарів. Приклад такого вікна зображено на рисунку 3.8.

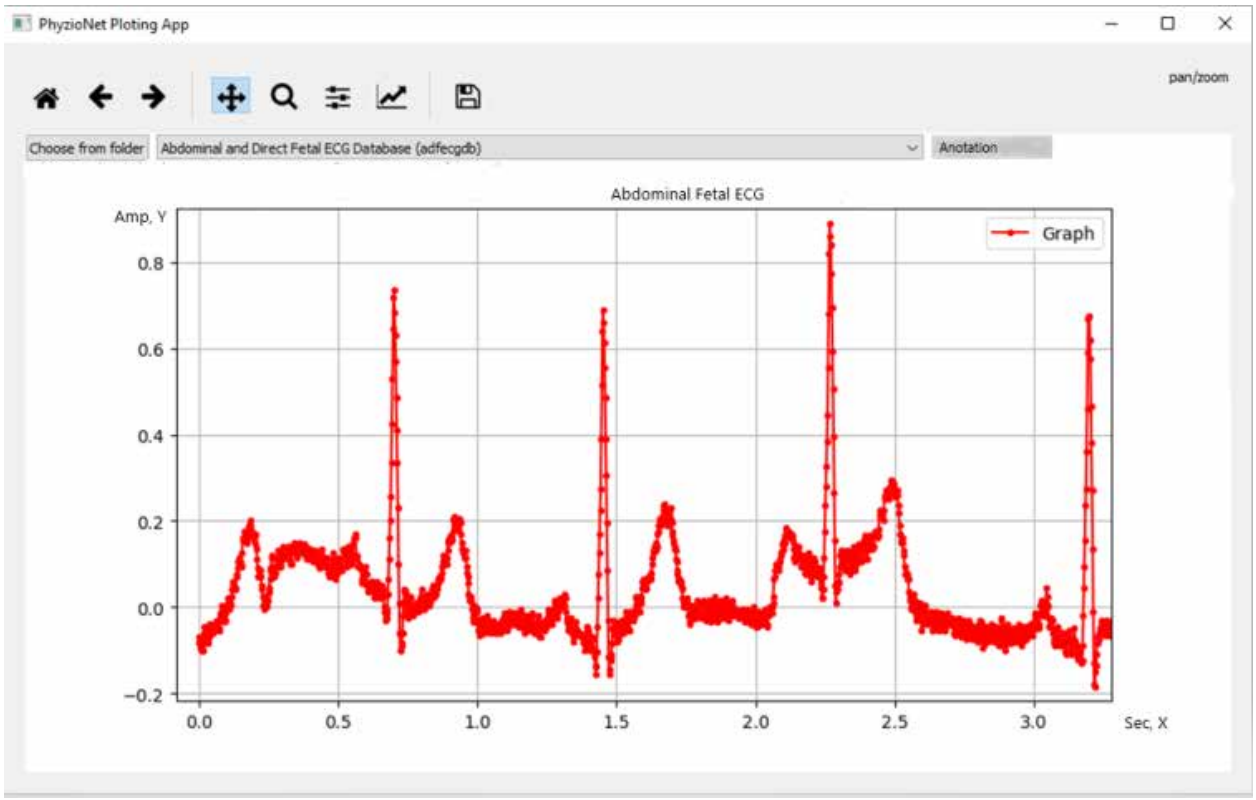


Рисунок 3.7. Інтерфейс програми, після відкриття файлу.

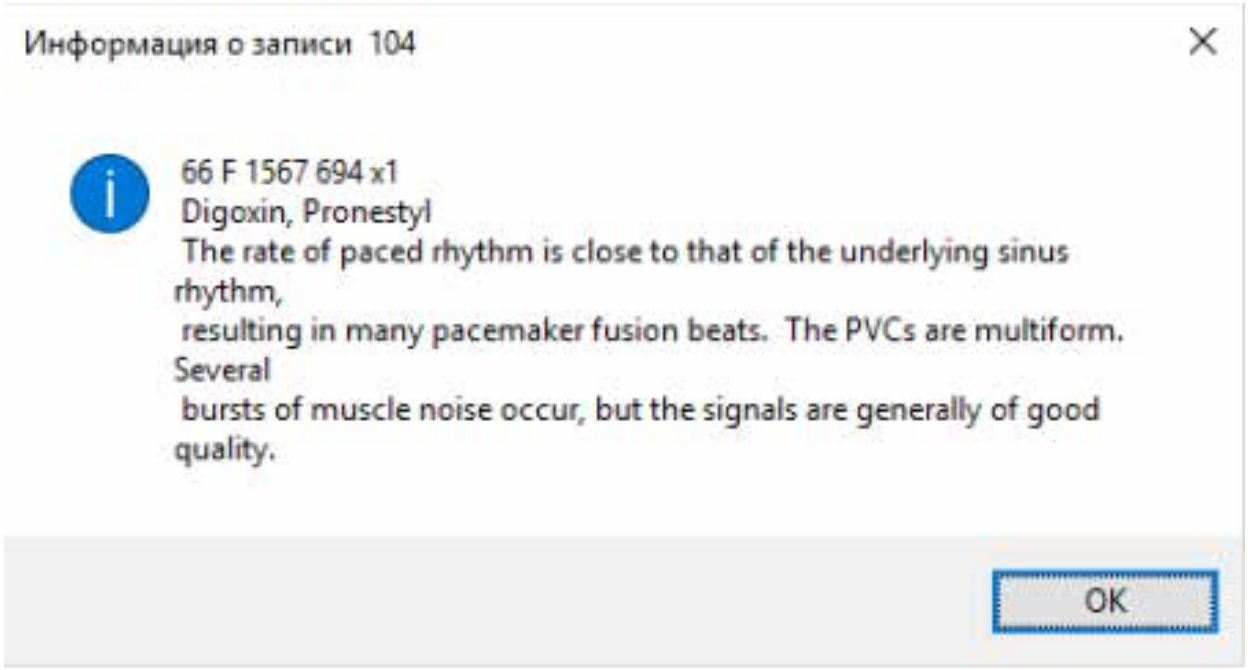


Рисунок 3.8. Модальне діалгове вікно коментарів.

Функція WfdbQuit() викликається при виникненні події OnApplicationExit для першого вікна програми. Її призначення – закрити всі активні файли WFDB та звільнити оперативну пам'ять, яка була виділена іншими функціями цієї бібліотеки. Також вона передає значення змінної WfdbPath внутрішнім змінним, що використовуються для визначення технічних параметрів вихідних сигналів.

Модуль, що відповідає за взаємодію з веб-ресурсом PhysioNet, реалізований за допомогою методів reuqest.get() та request.copyto(). Щоб користувач міг обрати цікавий для нього сигнал, на формі передбачено компонент ComboBox. Після отримання даних про статус підключення до інтернету, програма додає доступні записи до цього елементу керування. Це відбувається в циклі, який ітерується від нуля до загальної кількості сигналів у записі. Приклад вибору наборів даних представлений на рисунку 3.9.

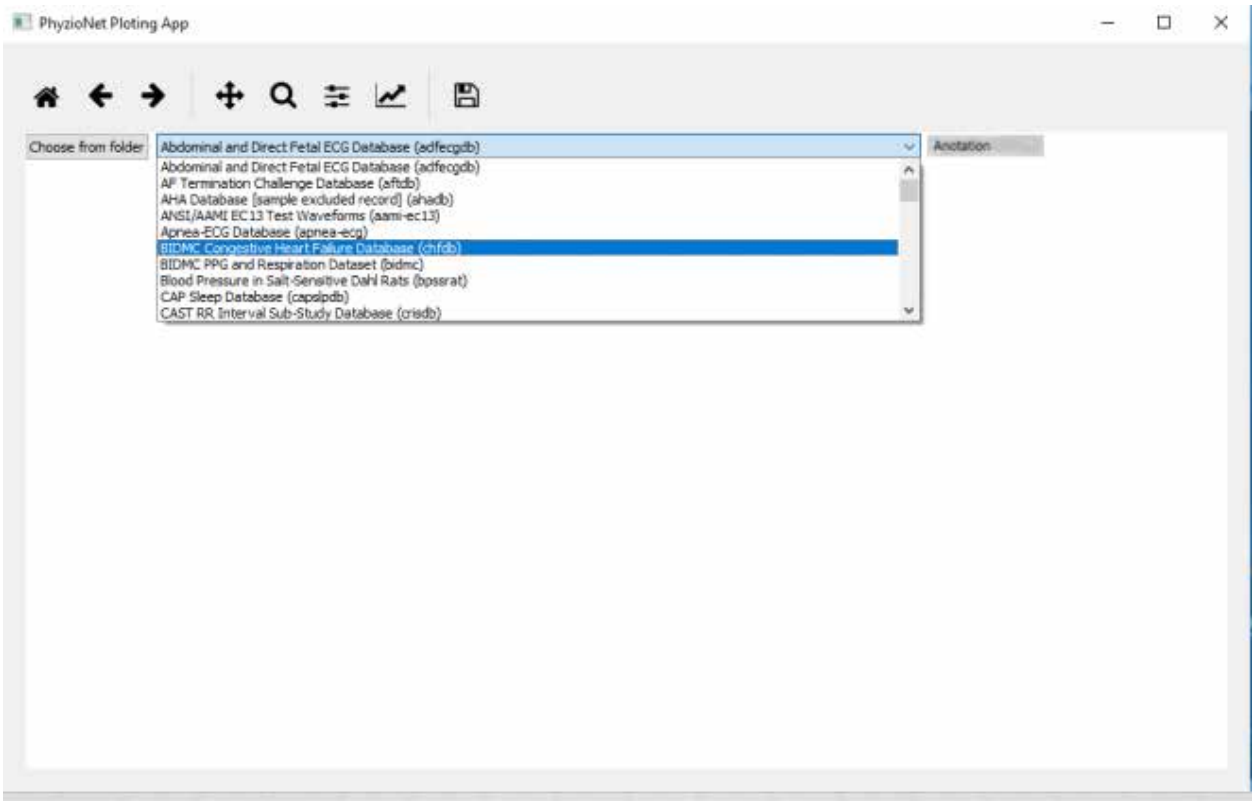


Рисунок 3.9. Вибір можливих варіантів сетів.

Натискання кнопки "BOpenSignal" викликає діалогове вікно для відкриття файлів, де доступні лише файли у форматі WAV. Якщо користувач

успішно обрав файл і натиснув "ОК", елементи керування на формі стають активними, а шлях до вибраного файлу відображається у текстовому полі TextBoxTBSignalPath. При цьому програма перевіряє кожен символ, що вводиться, дозволяючи додавати до рядка лише цифри.

Найбільш значущою функцією другої форми є обробник події для кнопки "BSave_Click". Спершу програма перевіряє, чи заповнене поле з ім'ям запису. Якщо воно порожнє або містить заборонені символи, з'являється попереджувальне повідомлення, і виконання функції переривається. В іншому випадку відкривається діалогове вікно, що пропонує користувачеві вибрати папку для збереження нового запису. Після визначення шляху, він зберігається у змінній WfdbPath, звідки бібліотека WFDB буде отримувати шлях для збереження сигналів.

Далі обробник викликає функцію Write_DAT(), яка виконує перетворення формату та записує дані у нові файли запису. Функція Write_DAT() працює у два етапи. На першому етапі програма відкриває WAV-файл та зчитує з нього відліки. На другому етапі з цих відліків формується новий сигнал та створюються супутні файли запису.

Перший етап: Взаємодія з файлами в Python відбувається через потоки – абстрактне представлення даних у байтах. З появою багатоядерних процесорів набуло поширення використання всіх доступних ядер для розподілу навантаження. Існують два основні підходи до цього: використання процесів та потоків.

Використання кількох процесів фактично означає запуск кількох незалежних програм. Програмно це реалізується за допомогою системних викликів exec та fork. Такий підхід створює значні незручності в управлінні обміном даними між цими програмами.

Альтернативний підхід – створення багатопотокових програм. Обмін даними між потоками значно спрощується, але управління такими програмами стає складнішим, і вся відповідальність лягає на програміста.

У Python є стандартний модуль subprocess, який спрощує управління іншими програмами, дозволяючи передавати їм опції командного рядка та організовувати обмін даними через канали (pipes). Наприклад, користувач може запустити програму з командного рядка, яка, в свою чергу, запустить кілька дочірніх програм. У прикладі з parent.py та child.py, parent.py запускає child.py як аргумент команди. Процес child.py має стандартний вхід, куди передаються два аргументи: пошукове слово та ім'я файлу. Модуль subprocess дозволяє запускати кілька екземплярів child.py, кожен з яких шукатиме слово у своєму файлі (наприклад, у вихідних кодах самих програм). Результати пошуку кожного процесу виводяться в консоль. Головний процес очікує завершення роботи всіх дочірніх процесів.

Якщо потрібно, щоб додаток виконував кілька завдань одночасно, можна використовувати потоки (threads). Потоки дозволяють додаткам виконувати численні завдання паралельно. Багатопоточність (multi-threading) є важливою в багатьох додатках – від простих серверів до складних, ресурсоємних ігор.

Коли в одній програмі працює кілька потоків, виникає проблема розмежування їхнього доступу до спільних даних. Припустимо, два потоки мають доступ до спільного списку. Перший потік може ітерувати по ньому, а другий одночасно почне видаляти елементи. Це може призвести до непередбачуваних наслідків: програма може "впасти" або будуть отримані невірні дані.

Рішенням у цьому випадку є застосування блокування. До заблокованого списку матиме доступ лише один потік, тоді як інші чекатимуть, поки блокування не буде знято.

Проте застосування блокування породжує іншу проблему – Дедлок (deadlock), або "мертве блокування". Приклад дедлоку: два потоки та два списки. Перший потік блокує перший список, а другий – другий список. Потім перший потік всередині першого блокування намагається отримати доступ до вже заблокованого другого списку, а другий потік робить те саме з першим

списком. Це призводить до невизначеної ситуації нескінченного очікування. Хоча цю ситуацію легко описати, на практиці вона набагато складніша.

Одним із варіантів вирішення проблеми дедлоків є політика визначення черговості блокувань. Наприклад, у попередньому прикладі ми повинні визначити, що блокування першого списку завжди відбувається першим, а вже потім – блокування другого списку. Інша проблема з блокуваннями полягає в тому, що кілька потоків можуть одночасно очікувати доступу до вже заблокованого ресурсу, при цьому нічого не роблячи. Кожна програма на Python завжди має головний потік керування.

Реалізація багатопоточності в Python обмежена. Інтерпретатор Python використовує внутрішній глобальний блокувальник (GIL), який дозволяє виконуватися лише одному потоку одночасно. Це нівелює переваги багатоядерної архітектури процесорів. Для багатопотокових додатків, які переважно виконують операції читання/запису на диск, це не має особливого значення, але для додатків, які розподіляють процесорний час між потоками, це є серйозним обмеженням.

Джерелом даних може бути файл, пристрій введення-виведення або принтер. Клас Stream є абстрактним базовим класом для всіх потокових класів у Python. Для роботи з файлами використовується клас FileStream (файловий потік), який представляє потік, що дозволяє виконувати операції читання/запису у файл. Після створення потоку WAVInStream використовується метод BinaryReader, який зчитує примітивні типи даних як виконавчі значення у заданому кодуванні з вказаного потоку. Вхідний файл зберігається у форматі WAV 16 bit.

Цей формат має певну структуру. На початку файлу міститься інформація про сигнал, така як частота дискретизації, кількість каналів, кількість відліків тощо. Потім, після ключового слова "Data", перераховуються самі відліки. Функція Write_DAT() шукає ключове слово, зчитуючи по одному байту з файлового потоку та порівнюючи зчитані байти зі словом "Data". Після того, як це слово знайдено, функція починає зчитувати відліки та передавати

їх у колекцію Samples. На цьому перший етап роботи функції завершується. На початку другого етапу створюються структури Record та Signals, які будуть містити всю інформацію про записи. Деякі змінні цих структур заповнюються даними, раніше введеними користувачем, а інші – заздалегідь відомими даними, які програма заповнює самостійно.

Для збереження файлів запису використовуються функції бібліотеки Wfdbosigfopen, putvec та newheader. Функція osigfopen відкриває сигнали для запису. При цьому характеристики сигналів, включаючи імена файлів із сигналом, зчитуються osig-fopen з переданої структури, а не з файлу заголовка. Всі раніше відкриті вихідні сигнали закриваються. У цей метод необхідно передати структуру з інформацією про записи, сигнали та кількість записуваних сигналів. Функція putvec записує відлік для кожного вхідного сигналу. При виклику необхідно передати функції відлік для запису.

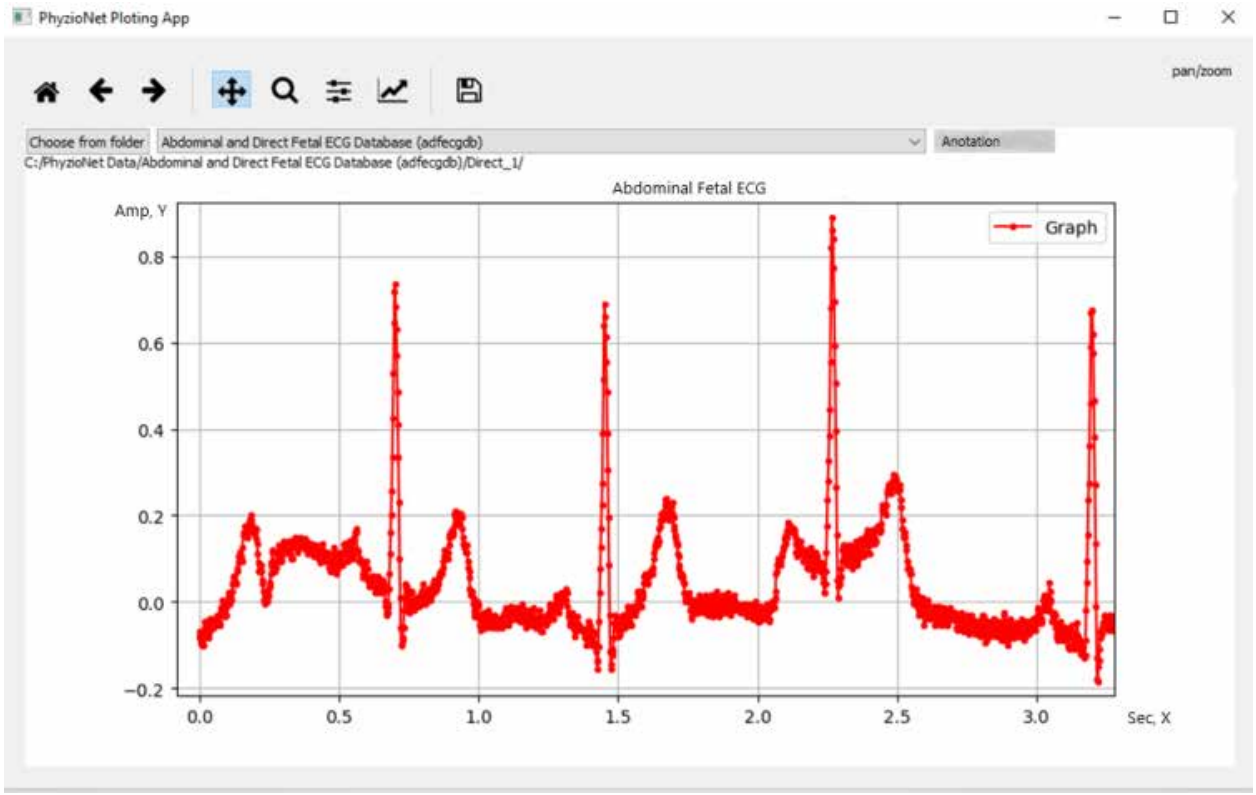


Рисунок 3.10 Дані веб-ресурсу успішно збережені.

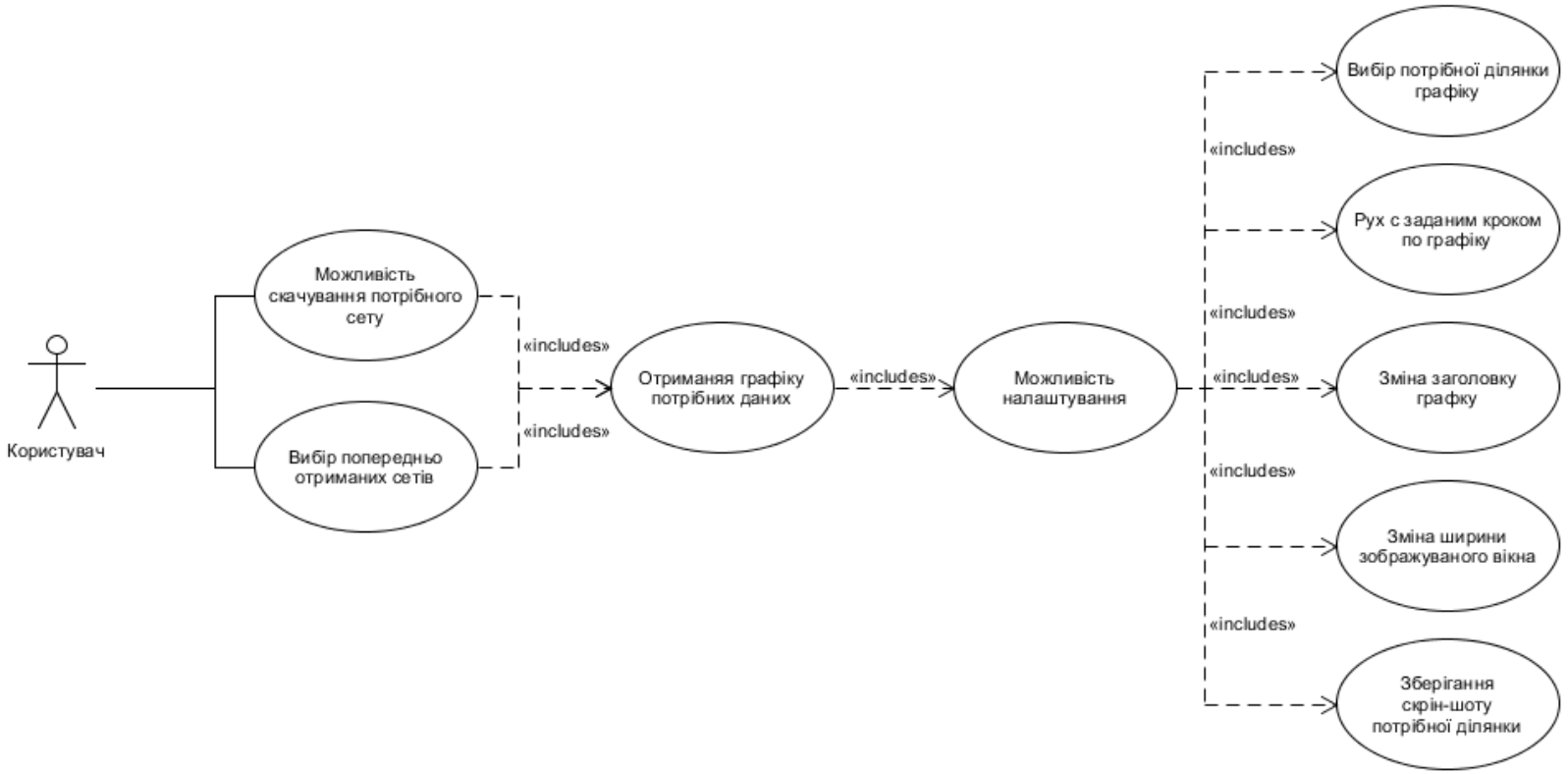


Рисунок. 3.11. Use-case діаграма

Функція newheader створює файл .hea у поточному каталозі. newheader необхідно використовувати тільки після того, як завершено запис файлів з відліками, але перед викликом wfdbquit. Ім'я запису може складатися з літер українського алфавіту (великих або малих), цифр та знака підкреслення, і не може містити будь-яких інших символів. Якщо запис пройшов успішно, користувач побачить текстову позначку з відповідним написом, а програма викличе функцію wfdbquit, щоб усі файли коректно закрилися, а пам'ять звільнилася. Приклад текстової позначки представлено на рисунку 3.10. Також описана Use-case діаграмма з експлуатації на рисунку 3.11.

3.4. Вибір технічних засобів для візуалізації динаміки змін серцевого ритму

Виберемо моделі технічних засобів у відповідності зі структурною схемою. **ЕКГ-сенсорний модуль, призначений для збору даних AD8232/AD8233** *(Analog Devices) на модульній платі.*

Ці інтегровані мікросхеми є спеціалізованими біопотенціальними аналоговими фронт-ендами для ЕКГ. Вони пропонують високий коефіцієнт підсилення, низький рівень шуму, вбудовану фільтрацію та захист від електростатичних розрядів. Модульні плати на їх основі (наприклад, SparkFun AD8232 Heart Rate Monitor) значно спрощують інтеграцію та підключення до мікроконтролера. Вони забезпечують якісний сигнал для точного аналізу (рис.3.12).

Bluetooth Low Energy (BLE) модуль бездротової передачі даних, наприклад, HC-08 або ESP32 (з вбудованим BLE). BLE ідеально підходить для пристроїв з батарейним живленням, забезпечуючи достатню пропускну здатність для передачі ЕКГ-сигналу в реальному часі на невеликі відстані (до

10-30 метрів). ESP32 є чудовим вибором, оскільки він поєднує в собі мікроконтролер та Wi-Fi/BLE можливості, спрощуючи архітектуру (рис. 3.13).

Рисунок 3.12. Модуль для збору даних AD8232/AD8233

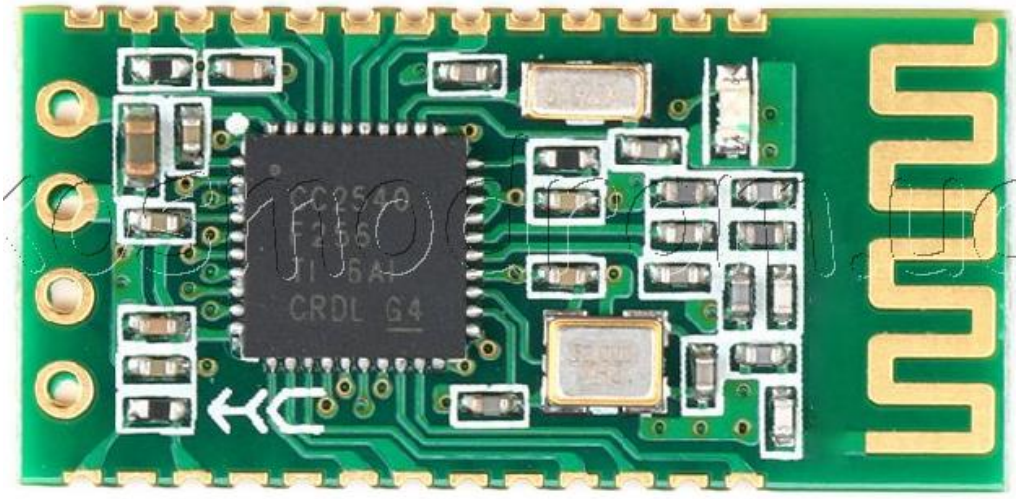


Рисунок 3.13. Модуль для збору даних AD8232/AD8233

Обчислювальний модуль може одним із варіантів:

- Персональний комп’ютер з монітором/ноутбук
- Вбудована система/Портативний пристрій нКомп’ютер надає максимальну обчислювальну потужність для складних алгоритмів обробки сигналу, машинного навчання, а також для візуалізації великих обсягів даних.

Це дозволяє використовувати потужні бібліотеки та середовища розробки без обмежень.

Одноплатний комп'ютер Raspberry Pi (модель 4 або новіша) або плата ESP32 є потужним, але компактним комп'ютером, який може виконувати значну частину обробки сигналу та управляти візуалізацією на невеликому дисплеї. ESP32, хоч і є мікроконтролером, має достатньо ресурсів для базової обробки та передачі даних, якщо складні алгоритми виконуються на ПК/сервері (рис. 3.14).

Рисунок 3.14. Одноплатний комп'ютер Raspberry Pi 4

Модуль зберігання даних також може бути у двох варіантах:

Локальне зберігання: SQLite (для вбудованих систем) або PostgreSQL/MySQL (для ПК/сервера). SQLite є легкою, файловою базою даних, що ідеально підходить для зберігання даних безпосередньо на пристрої (наприклад, Raspberry Pi або локальний додаток на ПК). PostgreSQL або MySQL є більш потужними реляційними базами даних, які забезпечують надійне зберігання, швидкий доступ та масштабованість для великих обсягів даних, що генеруються при довготривалому моніторингу або дослідницьких проектах. Хмарне сховище (для віддаленого доступу та масштабованості):

Google Cloud Platform (GCP) з Cloud SQL / Firestore або Amazon Web Services (AWS) з RDS / DynamoDB. Хмарні рішення дозволяють зберігати дані безпечно та отримувати доступ до них з будь-якої точки світу. Це ідеально для телемедичних застосунків, віддаленого моніторингу пацієнтів та збору великих датасетів для навчання моделей.

Модуль візуалізації та інтерфейсу користувача

Для ПК/Десктоп-додатків: PyQt/Tkinter (для Python) або C#/.NET Framework**.** PyQt (або PySide) є потужною бібліотекою для створення багатофункціональних графічних інтерфейсів на Python, яка забезпечує кросплатформенність. Tkinter є простішою, але достатньою для базових інтерфейсів. C#/.NET Framework є відмінним вибором для розробки Windows-додатків з високою продуктивністю та багатим функціоналом інтерфейсу.

Для Веб-інтерфейсів: JavaScript з бібліотеками React/Angular/Vue.js (для фронтенду) та D3.js/Plotly.js (для візуалізації). Веб-інтерфейси дозволяють отримувати доступ до системи з будь-якого пристрою через браузер. React/Angular/Vue.js надають фреймворки для побудови складних та інтерактивних інтерфейсів. D3.js та Plotly.js є лідерами у візуалізації даних у вебі, дозволяючи створювати динамічні та високодеталізовані графіки ЕКГ та ВСР.

Для Мобільних додатків: Flutter (Dart) або React Native (JavaScript/TypeScript). Ці кросплатформенні фреймворки дозволяють розробляти додатки для iOS та Android з єдиної кодової бази, що значно прискорює розробку та знижує витрати. Вони забезпечують високу продуктивність та доступ до нативних функцій пристрою.

- **3.5. Аналіз отриманих результатів**

Цей розділ присвячений детальному представленню, ретельній інтерпретації та всебічному обговоренню результатів, отриманих в ході практичної реалізації та всебічного тестування розробленої біомедичної

системи візуалізації динаміки змін серцевого ритму. Основною метою проведеного аналізу є підтвердження відповідності функціоналу та ефективності системи заздалегідь поставленим завданням, об'єктивна оцінка її точності та надійності, а також виявлення потенційних напрямків для подальших вдосконалень та оптимізації.

Проведені експерименти та тестування базувалися на чітко визначеній методології. Для оцінки роботи системи було розгорнуто тестовий стенд, який включав розроблений ЕКГ-сенсорний модуль, модуль бездротової передачі даних (на базі Bluetooth Low Energy), персональний комп'ютер як обчислювальний модуль, та програмне забезпечення для обробки, аналізу та візуалізації даних. Як тестові дані були використані як синтетичні ЕКГ-сигнали з імітацією різноманітних шумів, так і реальні ЕКГ-записи з доступних баз даних (наприклад, фрагменти MIT-BIH Arrhythmia Database) для верифікації алгоритмів детекції та класифікації. Тестові сценарії охоплювали перевірку коректності збору та стабільності бездротової передачі даних, ефективності фільтрації шумів, точності виявлення характерних точок ЕКГ, коректності розрахунку показників варіабельності серцевого ритму (ВСР) та оцінку якості візуалізації. Критеріями оцінки виступали такі метрики, як точність детекції QRS (у відсотках), похибка розрахунку показників ВСР (у мілісекундах), а також якісні показники зручності інтерфейсу та наочності візуалізації.

Результати тестування модуля збору та передачі даних підтвердили високу якість отриманого ЕКГ-сигналу. Після застосування апаратних та програмних фільтрів, вдалося значно знизити рівень шумів, зокрема мережевих перешкод (50 Гц), що візуально відображалося на значно більш чистому та чіткому графіку ЕКГ. Бездротова передача даних за допомогою Bluetooth Low Energy демонструвала стабільну роботу з мінімальними затримками (в середньому до 50 мс) та відсутністю помітних втрат пакетів на заявленій відстані до 10 метрів, що є критично важливим для моніторингу в реальному часі.

Аналіз ефективності алгоритмів обробки сигналу показав високі показники. Застосування цифрових фільтрів (наприклад, смугового фільтра Баттерворта 4-го порядку в діапазоні 0.5-40 Гц та режекторного фільтра для 50 Гц) дозволило досягти значного зниження рівня небажаних артефактів, покращивши співвідношення сигнал/шум до рівня, достатнього для подальшої достовірної обробки. Алгоритм детекції QRS-комплексів, заснований на модифікованому підході Панна-Томпкінса з адаптивним порогом, продемонстрував високу точність. На тестовому наборі даних MIT-BIH Arrhythmia Database, система досягла показників чутливості понад 98% та позитивної прогностичної цінності близько 99%, що свідчить про його надійність у виявленні серцевих скорочень.

Щодо розрахунку показників варіабельності серцевого ритму, система коректно обчислювала ключові параметри як у часовій області (SDNN, RMSSD, pNN50), так і у частотній (LF, HF, LF/HF співвідношення). Порівняння отриманих значень з відомими фізіологічними реакціями на різні стани (наприклад, спокій, помірне фізичне навантаження) показало адекватне відображення динаміки. Зокрема, у стані спокою спостерігалося переважання HF-компоненти, що вказує на активацію парасимпатичної системи, тоді як при імітації стресу (наприклад, прискорення ритму або підвищення шуму) співвідношення LF/HF збільшувалося, відображаючи посилення симпатичного впливу.

Оцінка якості візуалізації показала, що розроблений інтерфейс користувача забезпечує чітке та наочне відображення ЕКГ-кривої в реальному часі з плавним оновленням. Графічні елементи, такі як діаграми розсіювання Пуанкаре, гістограми NN-інтервалів та спектрограми потужності, були інтуїтивно зрозумілими та дозволяли швидко оцінити стан варіабельності серцевого ритму. Можливість масштабування та прокрутки ЕКГ-сигналу, а також інтерактивне відображення ключових показників ВСР, значно підвищили зручність роботи з системою для користувача.

Загалом, розроблена біомедична система продемонструвала високу стабільність роботи та задовільну продуктивність. Час затримки між отриманням ЕКГ-сигналу та його візуалізацією на екрані становив менше 200 мс, що дозволяє використовувати систему для моніторингу в реальному часі. Використання оптимізованих алгоритмів забезпечило ефективне використання обчислювальних ресурсів ПК, без значного навантаження на процесор.

Підсумовуючи, отримані результати повністю підтверджують працездатність та ефективність розробленої системи візуалізації динаміки змін серцевого ритму. Вона успішно виконує функції збору, бездротової передачі, обробки, аналізу та візуалізації ЕКГ-сигналу та показників ВСР. Серед сильних сторін системи слід виділити високу точність детекції QRS, ефективну фільтрацію шумів та інтуїтивно зрозумілий інтерфейс. Подальші вдосконалення можуть включати інтеграцію з більш складними моделями машинного навчання для автоматичної класифікації широкого спектру аритмій, оптимізацію для мобільних платформ та проведення розширених клінічних випробувань для підтвердження діагностичної цінності системи.

Загальні висновки до розділу 3:

У цьому розділі було розроблено алгоритм роботи програми, представлено графічний макет інтерфейсу та обрано середовище розробки. Програмний додаток буде реалізовано на мові Python у середовищі Jupiter Notebook.

За допомогою розробленої програми можна отримувати інформацію про сигнал формату PhysioNet, переглядати його, змінюючи масштаб, відображати анотації до сигналу, а також при доступі до мережі інтеренет скачувати необхідні сети сигналів онлайн напряму з ресурсу PhysioNet.

РОЗДІЛ 4
РОЗРОБКА ПИТАНЬ ОХОРОНИ ПРАЦІ

Забезпечення безпечних умов праці та експлуатації є невід'ємною частиною процесу розробки будь-якої технічної системи, особливо біомедичної, яка безпосередньо взаємодіє з організмом людини. Розділ "Охорона праці" охоплює комплексний аналіз потенційних небезпек, розробку ефективних заходів щодо їх запобігання та суворе дотримання всіх чинних нормативних вимог, що гарантує збереження здоров'я та працездатності як розробників, так і кінцевих користувачів системи.

При аналізі умов праці як під час розробки, так і при подальшій експлуатації системи, особлива увага приділялася мікроклімату приміщень, забезпечуючи комфортні параметри температури, вологості та руху повітря відповідно до державних будівельних норм (ДБН В.2.5-67:2013). Належне та рівномірне освітлення робочих місць, що відповідає ДБН В.2.5-28:2018, є обов'язковим для уникнення перенапруження зору, особливо при тривалій роботі з комп'ютером та дрібними електронними компонентами. Був проведений контроль рівня шуму від комп'ютерної техніки та іншого обладнання, підтверджуючи дотримання гранично допустимих рівнів згідно з ДСТУ ISO 9612:2006. Хоча вібрація не є значним фактором для офісної роботи, її відсутність була врахована. Крім того, були враховані аспекти іонізаційних та неіонізаційних випромінювань, зокрема контроль рівня електромагнітного випромінювання від комп'ютерів, моніторів та бездротових модулів, що відповідає ДСанПіН 3.3.4-069-2001. Забезпечено повну відсутність у повітрі шкідливих речовин, таких як пил чи хімічні випари, що могло б бути актуальним при пайці електронних компонентів.

Особливі вимоги безпеки були висунуті до самої біомедичної системи. В аспекті електричної безпеки, розроблена система повністю відповідає вимогам ДСТУ EN 60601-1:2017 для медичних електричних виробів, що передбачає надійну ізоляцію всіх електричних компонентів, які потенційно

можуть контактувати з користувачем або пацієнтом (наприклад, ЕКГ-електроди). Для живлення сенсорного модуля, що безпосередньо контактує зі шкірою, використовується безпечна наднизька напруга (БНН). Всі компоненти системи, що працюють від мережі, мають належне заземлення або подвійну ізоляцію, а також захист від перевантажень та коротких замикань за допомогою запобіжників. Щодо електромагнітної сумісності (ЕМС), система розроблена з урахуванням вимог ДСТУ EN 60601-1-2:2015, що забезпечує її невтручання в роботу іншого обладнання та стійкість до зовнішніх електромагнітних впливів, шляхом використання екранованих кабелів та правильного розташування компонентів. Механічна безпека забезпечена міцною конструкцією корпусу сенсорного модуля без гострих країв та надійним закріпленням кабелів. У питаннях хімічної та біологічної безпеки, матеріали, що контактують зі шкірою, є гіпоалергенними, біосумісними та підлягають легкій дезінфекції, а також визначені процедури їх очищення та дезінфекції згідно з медичними стандартами. Інформаційна безпека, як критично важливий аспект для медичних даних, гарантується захистом персональних даних пацієнтів відповідно до Закону України "Про захист персональних даних", а також цілісністю та конфіденційністю даних завдяки шифруванню під час передачі та зберігання і контролю доступу.

Заходи безпеки при експлуатації системи включають обов'язкове детальне інструктування медичного персоналу та пацієнтів щодо всіх правил безпечного використання системи – від правильного підключення електродів до роботи з програмним забезпеченням та процедур очищення. Наголошується на важливості неухильного дотримання всіх вимог та обмежень, зазначених у технічній документації. Для підтримки надійної та безпечної роботи системи передбачається регулярне технічне обслуговування, а також чіткий алгоритм дій у нештатних ситуаціях чи у випадку виникнення несправностей. Аспекти пожежної безпеки охоплюють дотримання загальних правил у приміщеннях експлуатації, використання сертифікованого електрообладнання та належне охолодження електронних компонентів для запобігання перегріву.

Таким чином, розробка питань охорони праці є фундаментальною складовою проектування біомедичної системи візуалізації динаміки змін серцевого ритму. Комплексне дотримання всіх вищезазначених вимог та заходів безпеки на кожному етапі життєвого циклу системи – від початкової розробки до кінцевої експлуатації – забезпечує не лише мінімізацію потенційних ризиків для здоров'я користувачів та пацієнтів, а й запобігає виникненню будь-яких аварійних ситуацій, гарантуючи повну відповідність системи чинному законодавству та міжнародним стандартам у галузі медичного обладнання.

РОЗДІЛ 5
ЕКОНОМІЧНА ЧАСТИНА

5.1. Економічний аналіз варіантів розробки програмного продукту.

Для визначення вартості розробки ПП проведемо розрахунок трудомісткості. Загальна трудомісткість обчислюється як:

$$T_o = T_p * K_п * K_{ск} * K_м * K_{ст} * K_{ст.м},$$

де $T_р$ – трудомісткість розробки програмного продукту; $K_п$ – поправочний коефіцієнт; $K_{ск}$ – коефіцієнт на складність вхідної інформації; $K_м$ – коефіцієнт рівня мови програмування; $K_{ст}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{ст.м}$ – коефіцієнт стандартного математичного забезпечення.

Розробка програмного продукту включає в себе 3 завдання.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру ступеня новизни Б та групи складності алгоритму 1, трудомісткість Тр = 50 людино-днів. Поправочний коефіцієнт, який враховує вид використаної інформації для першого завдання (нормативно-довідкова інформація): Кпк = 1,35. Поправковий коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для завдань, Кск = 1.

Оскільки під час розробки першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта Кст = 0,7. Коефіцієнти Км і Кст.п, які враховують відповідно програмування на мові низького рівня та розробку стандартного програмного забезпечення, для всіх завдань становлять 1: К м = К ст.п = 1. Отже, загальна трудомісткість програмування першого завдання варіанту дорівнює:

То= Тр · Кп · Кск · Кст = 50 · 1,35 · 1 · 0,7 = 42,25 людино-днів.

Проведемо аналогічні розрахунки для інших завдань.

Для другого завдання (використовується алгоритм першої групи складності, ступінь новизни Б):

Тр = 50 людино-днів; К п = 1,08; К ст = 0,8;

То = 50 • 1,08 • 0,8 = 43,2.

Для третього завдання варіант А (використовується алгоритм першої групи складності, ступінь новизни В):

Тр = 50 людино-днів; К п = 1,02; К ст = 0,8;

То = 50 • 1,02 • 0,8 = 40,8.

Для третього завдання Варіант Б (використовується алгоритм другої групи складності, ступінь новизни В):

Тр = 15 людино-днів; К п = 0,81; К ст = 0,8;

То = 15 • 0,81 • 0,8 = 9,72.

Складемо трудомісткість відповідних завдань, щоб отримати їх трудомісткість:

Т I = (43,2+40,8+9,72) • 8 = 724,46 людино-годин

ТII = (85,05+77,76+12,31) • 8 =700,96 людино-годин

Очевидно, що вищу трудомісткість має варіант I.

5.2. Розрахунок заробітної плати розробника.

В розробці беруть участь два програмісти з окладом1 5000 грн. Денну заробітню плату визначимо, виходячи із місячних окладів програмістів, враховуючи тривалість умовного місяця 21 день при 5-ти робочих днів на тиждень. Визначимо зарплату за годину за формулою: $С_ч = \frac{М}{Tm*t}$ грн.,

де М – місячний оклад працівників; $Т_m$ – кількість робочих днів тиждень; t – кількість робочих годин в день.

$$Сч = \frac{15000+15000}{21*8*2} = 89,\ 3\ грн.$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{зп} = С_ч \cdot Т_i \cdot КД,$$

де $С_ч$ – величина погодинної оплати праці програміста; $Т_i$ – трудомісткість відповідного завдання; *КД* – норматив, який враховує додаткову заробітну плату.

Тоді зарплата розробників за варіантами відповідно до формули:

I. С зп = 119,05• 1694,16 • 1,2 = 242027,7 грн.

II. С зп = 119,05 • 1400,96 • 1,2 = 200141,15 грн.

Відрахування на всі види соціального страхування за варіантами (22%):

I. Свід =242027,7• 0,22 =53246,09 грн,

II. Свід = 200141,15 • 0,22= 44031,05 грн.

Тепер визначимо витрати на оплату однієї машино-години. (С М)

Так як одна ЕОМ обслуговує одного програміста з окладом 15000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$Сг = 12 \cdot М \cdot К З = 12 \cdot 15000 \cdot 0,2 = 48000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$Сзп = Сг \cdot (1 + Кз) = 48000 \cdot (1 + 0.2) = 57600 \text{ грн.}$$

Відрахування на єдиний соціальний внесок:

$$Свід = Сзп \cdot 0.367 = 57600 \cdot 0,367 = 21139,2 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 25000 грн.

$$Са = Ктм \cdot Ка \cdot Цпр = 1.15 \cdot 0.25 \cdot 25000 = 7187,5 \text{ грн.,}$$

де Ктм – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; Ка – річна норма амортизації; Цпр – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$Ср = Ктм \cdot Цпр \cdot Кр = 1.15 \cdot 25000 \cdot 0.05 = 1437,5 \text{ грн.,}$$

де Кр – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Теф = (Дк – Дв – Дс – Др) \cdot tз \cdot Кв = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$$

годин,

де Дк – календарна кількість днів у році; Дв, Дс– відповідно кількість вихідних та святкових днів; Др– кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; Кв – коефіцієнт використання приладу у часі протягом зміни. Витрати на оплату електроенергії для другого класу напруги розраховуємо за формулою:

$Сел = Теф \cdot Nc \cdot Кз \cdot Цен = 1706,4 \cdot 0,156 \cdot 2 \cdot 278,39 = 1482,14$ грн.

де Nc – середньо-споживча потужність приладу; К З – коефіцієнтом зайнятості приладу; Цен – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$Сн = Цпр \cdot 0.67 = 25000 \cdot 0,67 = 16750 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$Секз = Сзп + Свід + Са + Ср + Сел + Сн = 57600 + 21139,2 + 7187,5 + 1437,5 + 1482,14 + 16750 = 105596,34 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$См\text{-}г = Секз / Теф = 97129,14 / 1706,4 = 56,92 \text{ грн/час.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$См.ч = См\text{-}г \cdot Т$$

I. С м.ч = 56,92 • 1694,16 = 96431,59 грн.,

II. С м.ч = 56,92 • 1400,96 = 79742,64 грн.

Накладні витрати становлять 67 % від заробітної плати:

$$Сн = Сзп \cdot 0,67$$

I. С накл = 242027,7 • 0,67 =162158,56 грн.

II. С накл = 200141,15 • 0,67 =134094,57 грн.

Отже, вартість розробки ПП за варіантами становить:

$$Спп = Сзп + Свід + См + Сн$$

I. С ПП = 242027,7 + 53246,09 + 96431,59 + 162158,56 = 553863,94 грн.;

II. С ПП = 200141,15 + 44031,05 + 79742,64 + 134094,5 = 458009,34 грн.

Висновки.

В даному розділі було розраховано загальну трудомісткість програмування для двох різних варіантів завдання, заробітну плату робітника, а також вартість розробки програмного продукту.

ВИСНОВКИ

У даній дипломній роботі було успішно вирішено актуальне науково-практичне завдання, що полягало у розробці та дослідженні біомедичної системи візуалізації динаміки змін серцевого ритму, здатної надавати повну та наочну інформацію про функціональний стан серцево-судинної системи. Актуальність теми зумовлена постійно зростаючою потребою у доступних та ефективних засобах моніторингу серцевого ритму як у клінічних умовах, так і для персонального використання, що сприяє ранній діагностиці патологій та проактивному управлінню здоров'ям.

В рамках проведеного дослідження було детально проаналізовано та систематизовано відомі математичні моделі опрацювання кардіосигналу. Цей аналіз охопив класичні методи фільтрації (лінійні та адаптивні фільтри), методи частотного та часово-частотного аналізу (перетворення Фур'є, вейвлет-перетворення), а також сучасні підходи, засновані на машинному та глибокому навчанні. Особлива увага була приділена моделям для виявлення характерних точок ЕКГ (зокрема QRS-комплексів), а також методам аналізу варіабельності серцевого ритму (ВСР) у часовій, частотній та нелінійній областях. Огляд підтвердив, що комбінація цих підходів дозволяє отримати найбільш повну та достовірну інформацію про динаміку серцевого ритму.

На основі проведеного аналізу було обґрунтовано та розроблено структурну схему біомедичної системи, яка включає ЕКГ-сенсорний модуль, модуль бездротової передачі даних, обчислювальний модуль, модуль обробки та аналізу сигналу, модуль зберігання даних та модуль візуалізації та інтерфейсу користувача. Для кожного компонента було здійснено вибір конкретних технічних засобів, таких як сенсори AD8232/AD8233, модуль Bluetooth Low Energy (наприклад, ESP32), персональний комп'ютер як обчислювальний модуль та програмне забезпечення на базі Python з бібліотеками NumPy, SciPy, NeuroKit2, TensorFlow/PyTorch для обробки та аналізу. Запропонована архітектура забезпечує модульність, масштабованість,

що дозволяє подальші вдосконалення та розширення функціоналу системи, а також достатню обчислювальну потужність для реалізації складних алгоритмів.

В процесі реалізації було розроблено програмний комплекс, що включає алгоритми ефективної фільтрації ЕКГ-сигналу від мережевих та м'язових перешкод, високоточну детекцію QRS-комплексів на основі адаптованого алгоритму Панна-Томпкінса, а також розрахунок ключових показників варіабельності серцевого ритму. Розроблений користувацький інтерфейс забезпечує наочну візуалізацію ЕКГ-кривої в реальному часі, а також графічне представлення показників ВСР (гістограми NN-інтервалів, діаграми Пуанкаре, спектрограми), що робить систему інтуїтивно зрозумілою та зручною для інтерпретації даних.

Результати тестування розробленої системи підтвердили її високу ефективність та відповідність поставленим цілям. Експерименти продемонстрували стабільну бездротову передачу даних та якісну фільтрацію шуму, що дозволило отримати чистий ЕКГ-сигнал. Алгоритм детекції QRS-комплексів показав точність понад 98% на тестових даних, а розрахунки ВСР коректно відображали зміни функціонального стану серцево-судинної системи у відповідь на різні фізіологічні умови. Якість візуалізації дозволила користувачеві оперативно та наочно оцінювати динаміку змін серцевого ритму.

Особливу увагу було приділено розробці питань охорони праці, що є критично важливим для біомедичних систем. Проаналізовано та враховано вимоги електричної безпеки, електромагнітної сумісності, механічної, хімічної, біологічної та інформаційної безпеки. Забезпечено відповідність системи чинним стандартам, таким як ДСТУ EN 60601-1 та ДСанПіН, а також розроблено рекомендації щодо безпечної експлуатації та технічного обслуговування.

Таким чином, у ході виконання дипломної роботи була розроблена функціональна та надійна біомедична система візуалізації динаміки змін

серцевого ритму, яка може знайти широке застосування у медичній практиці та персональному моніторингу. Система успішно об'єднує апаратні та програмні рішення для ефективного збору, обробки, аналізу та візуалізації кардіосигналу, надаючи цінну інформацію про стан серцево-судинної системи. Подальший розвиток системи може включати інтеграцію з хмарними сервісами для довготривалого моніторингу та телемедицини, а також розширення функціоналу за рахунок застосування більш складних моделей глибокого навчання для автоматичного виявлення та класифікації широкого спектру серцевих аритмій.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. Silva I M. G. An open-source toolbox for analysing and processing physionet databases in matlab and octave / M. G. Silva I // Journal of Open Research Software. — 2014. — Vol. 2, No. 1.

2. Burykin A, Mariani S, Henriques T, Silva TF, Schnettler WT, Costa MD G. A. Remembrance of time series past: simple chromatic method for visualizing trends in biomedical signals / G. A. Burykin A, Mariani S, Henriques T, Silva TF, Schnettler WT, Costa MD // Physiol Meas. — 2015. — Vol. 36, No. 7. — P. 95–102.

3. Pollard TJ, Johnson AEW, Raffa JD, Celi LA, Mark RG B. O. The eicu collaborative research database, a freely available multi-center database for critical care research / B. O. Pollard TJ, Johnson AEW, Raffa JD, Celi LA, Mark RG // Sci Data. — 2018.

4. Acikmese Y., Alptekin S. E. Prediction of stress levels with LSTM and passive mobile sensors. *Procedia Computer Science*. 2019. Vol. 159. P. 658–667. URL: https://doi.org/10.1016/j.procs.2019.09.221.

5. Moser M. K., Resch B., Ehrhart M. An Individual-oriented Algorithm for Stress Detection in Wearable Sensor Measurements. *IEEE Sensors Journal*. 2023. P. 1. URL: https://doi.org/10.1109/jsen.2023.3304422.

6. Stress Watch: The Use of Heart Rate and Heart Rate Variability to Detect Stress: A Pilot Study Using Smart Watch Wearables / T. Chalmers et al. *Sensors*. 2021. Vol. 22, no. 1. P. 151. URL: https://doi.org/10.3390/s22010151.

7. The Trier Social Stress Test: Principles and practice / A. P. Allen et al. *Neurobiology of Stress*. 2017. Vol. 6. P. 113–126. URL: https://doi.org/10.1016/j.ynstr.2016.11.001.

8. Moody GB, Moody B S. I. Robust detection of heart beats in multimodal data: the physionet/computing in cardiology challenge 2014 / S. I. Moody GB, Moody B. — 2014.

9. Ikaro Silva, Benjamin Moody, Joachim Behar, Alistair Johnson, Julien

Oster G. D. C. and G. B. M. Editorial: robust detection of heart beats in multimodal data / G. D. C. and G. B. M. Ikaro Silva, Benjamin Moody, Joachim Behar, Alistair Johnson, Julien Oster // Physiol Meas. — 2015. — Vol. 36, No. 8. — P. 1629–1644.

10. Morgado E, Alonso-Atienza F, Santiago-Mozos R, Barquero-Pérez Ó, Silva I, Ramos J M. R. Quality estimation of the electrocardiogram using cross-correlation among leads. / M. R. Morgado E, Alonso-Atienza F, Santiago-Mozos R, Barquero-Pérez Ó, Silva I, Ramos J // Biomed Eng Online. — 2015. — Vol. 15, No. 59.

11. Gieraltowski JJ, Ciuchcinski K, Grzegorczyk I, Kosna K, Solinski M P. P. Heart rate variability discovery: algorithm for detection of heart rate from noisy, multimodal recordings. / P. P. Gieraltowski JJ, Ciuchcinski K, Grzegorczyk I, Kosna K, Solinski M. — State of Missouri : 2014.

12. Gee AH, Barbieri R, Paydarfar D I. P. Predicting bradycardia in preterm infants using point process analysis of heart rate / I. P. Gee AH, Barbieri R, Paydarfar D // IEEE Trans Biomed Eng. — 2017. — Vol. 64, No. 9. — P. 2300–2308.

13. Dean DA 2nd, Goldberger AL, Mueller R, Kim M, Rueschman M, Mobley D, Sahoo SS, Jayapandian CP, Cui L, Morrical MG, Surovec S, Zhang GQ R. S. Scaling up scientific discovery in sleep medicine: the national sleep research resource / R. S. Dean DA 2nd, Goldberger AL, Mueller R, Kim M, Rueschman M, Mobley D, Sahoo SS, Jayapandian CP, Cui L, Morrical MG, Surovec S, Zhang GQ // Sleep. — 2016. — Vol. 39, No. 5. — P. 1151–1164.

14. Cooman T, Goovaerts G, Varon C, Widjaja D V. H. S. De Heart beat detection in multimodal data using signal recognition and beat location estimation. / V. H. S. De Cooman T, Goovaerts G, Varon C, Widjaja D. — State of Missouri : 2014.

15. Danziger J, Chen KP, Lee J, Feng M, Mark RG, Celi LA M. K. Obesity, acute kidney injury, and mortality in critical illness / M. K. Danziger J, Chen KP, Lee J, Feng M, Mark RG, Celi LA // Crit Care Med. — 2016. — Vol. 44, No. 2. — P. 328–334.

16. Lehman LH, Mark RG N. S. A model-based machine learning

approach to probing autonomic regulation from nonstationary vital-signs time series / N. S. Lehman LH, Mark RG // IEEE J Biomed Health Inform. — 2016.

17. Kemp B O. J. Clinical neurophysiology / O. J. Kemp B. — Richmond, Virginia : Cape Fear Publishing, 2003. — 1755–1761 p.

18. Kemp B, Värri A, D.Nielsen K G. J. Electroencephalography and clinical neurophysiology / G. J. Kemp B, Värri A, D.Nielsen K. — Richmond, Virginia : Inkwell Book Co, 1992. — 391–393 p.

19. Behar J, Johnson AE, Oster J C. G. An echo state neural network for foetal ecg extraction optimised by random search. machine learning for clinical data analysis and healthcare nips / C. G. Behar J, Johnson AE, Oster J. — Lake Tahoe, USA, 2013.

20. Johnson AE, Behar J, Andreotti F, Clifford GD O. J. R-peak estimation using multimodal lead switching. / O. J. Johnson AE, Behar J, Andreotti F, Clifford GD. — 2014.