

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
КАФЕДРА БІОМЕДИЧНОЇ ІНЖЕНЕРІ**

До захисту допущено:

Завідувач кафедри

_____ Владислав ШЛИКОВ

«__» _____ 2026 р.

**Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Медична інженерія»
спеціальності 163 «Біомедична інженерія»
на тему: «Пристрій реєстрації та аналізу рівня кислотності слини»**

Виконав (-ла):

студент (-ка) IV

курсу, групи БМ-21

Панченко Анастасія Євгенівна

Керівник:

Д.т.н., доц., завідувач кафедри БМІ

Шликов Владислав Валентинович

Консультант з охорони праці:

Доцент кафедри ОПЦБ, к.т.н.

Демчук Гліб Вікторович

Рецензент:

доц. каф. ББЗЛ, к.б.н.

Косякова Галина Василівна

Засвідчую, що у цьому дипломному
проекті немає запозичень з праць інших
авторів без відповідних посилань.

Студент (-ка) _____

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет біомедичної інженерії
Кафедра біомедичної інженерії

Рівень вищої освіти	<u>Перший (бакалаврський)</u>
Спеціальність	<u>163 Біомедична інженерія</u>
Освітньо-професійна програма	<u>Медична інженерія</u>

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Владислав ШЛИКОВ

«___» _____ 2026 р.

ЗАВДАННЯ

на дипломну роботу студенту

Панченко Анастасії Євгенівні

1. Тема роботи «Пристрій реєстрації та аналізу рівня кислотності слини», керівник роботи Шликов В.В. д.т.н., доц., завідувач каф. БМІ, затверджені наказом по університету від «27» травня 2026 р. № 1924-с
2. Термін подання студентом роботи «07» червня 2026 р.
3. Вихідні дані до роботи: науково-технічна література, технічні характеристики компонентів пристрою, програмне середовище *Arduino IDE*, середовище *Fritzing* для побудови схеми, *SolidWorks* для моделювання корпусу, *UltiMaker Cura* для підготовки до 3D-друку, *Android Studio* для розроблення мобільного додатку.
4. Зміст роботи: здійснити огляд літератури щодо фізіологічного значення рН слини, методів його вимірювання та існуючих пристроїв-аналогів; сформулювати медико-технічні вимоги до пристрою для реєстрації та аналізу рН слини; розробити апаратну частину пристрою на основі рН-сенсора, мікроконтролера *Arduino Uno*, *LCD*-дисплея та *Bluetooth*-модуля; створити програмне забезпечення для мікроконтролера; спроектувати корпус пристрою та підготувати його до 3D-друку; розробити мобільний додаток для фіксації, збереження, візуалізації та статистичного аналізу результатів вимірювання рН.
5. Перелік ілюстративного матеріалу: презентація у форматі *MS PowerPoint*.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
5	Демчук Г. Б., доц. каф. ОПЦБ		

7. Дата видачі завдання «17» квітня 2026 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за тематикою роботи	20.04.2026-26.04.2026	
2	Вибір компонентів і розробка апаратної частини пристрою та схеми підключення компонентів	27.04.2026-03.05.2026	
3	Розробка мобільного додатку для фіксації та аналізу результатів вимірювання	04.05.2026-10.05.2026	
4	Проектування корпусу, підготовка до 3D-друку та збірка прототипу	11.05.2026-17.05.2026	
5	Оформлення розділу «Охорона праці»	27.05.2026-30.05.2026	
6	Надання ДР рецензенту		
7	Подання ДР та анотації		
8	Захист ДР		

Студент

Анастасія ПАНЧЕНКО

Керівник

Владислав ШЛИКОВ

АНОТАЦІЯ

Тема дипломної роботи: «Пристрій реєстрації та аналізу рівня кислотності слини».

Обсяг дипломної роботи становить 127 сторінок, містить 27 ілюстрацій, 29 таблиць. Загалом опрацьовано 39 джерел.

Актуальність: Контроль рН слини є важливим для оцінювання кислотно-лужного стану ротової порожнини та умов, що впливають на демінералізацію й ремінералізацію емалі. Більшість портативних рН-метрів забезпечують лише разове вимірювання без збереження результатів, побудови графіків і статистичного аналізу. Тому актуальною є розробка пристрою, який поєднує вимірювання рН слини, бездротове передавання даних і мобільний додаток для їх фіксації та аналізу.

Мета дипломної роботи: вдосконалення способу оцінювання стану ротової порожнини та кислотно-лужного балансу для оцінки ризику розвитку карієсу.

Завдання для виконання мети:

1. Провести огляд значення рН слини, методів його вимірювання та існуючих пристроїв-аналогів.
2. Розробити апаратну частину пристрою на основі рН-сенсора, *Arduino*, *LCD*-дисплея та *Bluetooth*-модуля.
3. Створити програмне забезпечення для мікроконтролера.
4. Спроектувати корпус і виконати збірку прототипу.
5. Розробити мобільний додаток для збереження, візуалізації та статистичного аналізу результатів.

Ключові слова: рН слини, кислотність, рН-сенсор, *Arduino Uno*, *Bluetooth Low Energy*, мобільний додаток, біомедичний пристрій.

ANNOTATION

The topic of the diploma work: “Device for recording and analyzing the acidity level of saliva”.

Thesis contains 127 pages, 27 illustrations, 29 tables, 39 sources according to the list of references.

Actuality: Monitoring salivary pH is important for assessing the acid-base state of the oral cavity and the conditions affecting enamel demineralization and remineralization. Most portable pH meters provide only single measurements without saving results, graph construction, or statistical analysis. Therefore, the development of a device that combines salivary pH measurement, wireless data transmission, and a mobile application for recording and analyzing results is relevant.

Objective: Improvement of the method for assessing the condition of the oral cavity and acid-base balance to evaluate the risk of dental caries development.

Tasks to achieve the goal:

1. To review the significance of salivary pH, measurement methods, and existing analogue devices.
2. To develop the hardware part based on a pH sensor, Arduino, LCD display, and Bluetooth module.
3. To create microcontroller software.
4. To design the enclosure and assemble the prototype.
5. To develop a mobile application for saving, visualizing, and statistically analyzing the results.

Keywords: salivary pH, acidity, pH sensor, Arduino Uno, Bluetooth Low Energy, mobile application, biomedical device.

ЗМІСТ

АНОТАЦІЯ	4
ANNOTATION.....	5
Зміст	6
Вступ.....	8
Розділ 1 Теоретична частина.....	9
1.1 Слина як біологічне середовище та показник кислотно-лужного стану ротової порожнини.....	9
1.2 Методи вимірювання рН слини та біологічних рідин.....	12
1.3 Сенсори для вимірювання рН та особливості їх застосування у біомедичних системах	13
1.4 Огляд існуючих пристроїв-аналогів для вимірювання рН	16
1.5 Формування медико-технічних вимог до пристрою для реєстрації та аналізу рН слини	21
Висновок до розділу 1	23
Розділ 2 Розробка апаратної частини пристрою	25
2.1 Функціональна схема та загальний принцип роботи пристрою	25
2.2 Вибір компонентів та їх технічні характеристики	26
2.3 Збірка пристрою, підключення компонентів та ілюстрація схеми.....	31
2.4 Розроблення програмного забезпечення для мікроконтролера <i>Arduino</i>	34
2.5 Калібрування рН-сенсора, принцип і процес вимірювання	37
Висновок до розділу 2	40
Розділ 3 Моделювання корпусу	41
3.1 Формування вимог до корпусу пристрою	41
3.2 Моделювання корпусу пристрою.....	42
3.3 Підготовка моделі до 3D-друку та виготовлення корпусу	43
3.4 Збірка корпусу та розміщення компонентів пристрою.....	45
Висновок до розділу 3	46
Розділ 4 Написання мобільного додатку	48
4.1 Функціональні вимоги до мобільного додатку.....	48
4.2 Архітектура мобільного додатку.....	50
4.3 Реалізація <i>Bluetooth</i> -з'єднання з пристроєм.....	53

4.4 Збереження результатів вимірювання	57
4.5 Візуалізація та статистичне оброблення даних	61
4.6 Оцінювання динаміки рН слини за результатами моніторингу.....	64
Висновок до розділу 4	69
РОЗДІЛ 5 Охорона праці	71
5.1 Характеристики пристрою для аналізу рН слини.....	71
5.1.1 Характеристика компонентів пристрою	71
5.1.2 Складові частини системи.....	73
5.1.3 Характер взаємодії об'єкту в системі «людина – об'єкт»	74
5.2 Оцінювання потенційних небезпек та заходи їх усунення.....	74
5.2.1 Біологічна небезпека.....	75
5.2.2 Хімічна небезпека	77
5.2.3 Небезпека ураження електричним струмом.....	78
5.2.4 Пожежна небезпека.....	80
5.3 Розробка «Інструкції по техніці безпеки при експлуатації пристрою для аналізу рН слини»	82
Висновок до розділу 5	85
Висновки	87
Перелік використаних джерел	89
Додаток А.....	92
Додаток Б.....	97
Додаток В	100

ВСТУП

Сучасний розвиток біомедичної інженерії створює можливості для розроблення портативних систем контролю фізіологічних показників. Одним із них є рН слини, що характеризує кислотно-лужний стан ротової порожнини та змінюється під впливом харчування, гігієнічних процедур, мікрофлори, слиновиділення й індивідуальних особливостей організму.

Контроль рН слини важливий для спостереження за умовами, які впливають на демінералізацію та ремінералізацію емалі. Зниження рН у кислий бік може бути несприятливим для твердих тканин зубів. Тому актуальною є розробка пристрою, що поєднує реєстрацію рН слини, бездротове передавання даних і мобільний додаток для фіксації та аналізу результатів.

Тема дипломної роботи: пристрій реєстрації та аналізу рівня кислотності слини.

Мета дипломної роботи: вдосконалення способу оцінювання стану ротової порожнини та кислотно-лужного балансу для оцінки ризику розвитку карієсу.

Завдання для виконання мети:

1. Провести огляд фізіологічного значення рН слини, методів його вимірювання та існуючих пристроїв-аналогів.
2. Розробити апаратну частину пристрою на основі рН-сенсора, *Arduino*, *LCD*-дисплея та *Bluetooth*-модуля.
3. Створити програмне забезпечення для мікроконтролера для зчитування, оброблення, відображення та передавання значення рН.
4. Спроекувати корпус пристрою, підготувати його до 3D-друку та виконати збірку прототипу.
5. Розробити мобільний додаток для підключення до пристрою, фіксації результатів, перегляду історії, побудови графіка та розрахунку статистичних показників.

РОЗДІЛ 1

ТЕОРЕТИЧНА ЧАСТИНА

1.1 Слина як біологічне середовище та показник кислотно-лужного стану ротової порожнини

Слина є складною біологічною рідиною, яка безпосередньо впливає на стан ротової порожнини, тверді тканини зубів і мікробіологічну рівновагу. Вона секретується великими слинними залозами, а також малими залозами слизової оболонки ротової порожнини. За своїм складом слина переважно складається з води, проте також містить неорганічні іони, білки, ферменти, муцини, гормони, продукти метаболізму та мікроорганізми [1]. Саме багатокomпонентний склад слини визначає її участь у підтриманні гомеостазу ротової порожнини.

Основними функціями слини є зволоження слизової оболонки, участь у початкових етапах травлення, очищення поверхні зубів, антимікробний захист, нейтралізація кислот і підтримання мінерального балансу емалі. Важливу роль у цих процесах відіграють буферні системи слини, а також іони кальцію, фосфату та гідрокарбонату. Завдяки цим компонентам слина частково компенсує дію кислот, які утворюються внаслідок метаболізму вуглеводів мікроорганізмами зубного нальоту [1].

Одним із показників, що характеризує кислотно-лужний стан ротової порожнини, є рН слини. У нормі рН слини переважно перебуває у слабокислому або нейтральному діапазоні, приблизно 6,0–7,5 [1]. Значення цього показника не є сталим: воно змінюється під впливом харчування, вживання напоїв, рівня гігієни ротової порожнини, активності мікрофлори, швидкості слиновиділення, часу доби та індивідуальних особливостей організму [2]. Тому рН слини доцільно розглядати не як ізольований одноразовий параметр, а як динамічний показник, що відображає поточний стан ротового середовища.

Зниження рН слини свідчить про зміщення середовища в кислий бік. Такий стан є важливим з погляду розвитку карієсу, оскільки органічні кислоти, що утворюються в зубному нальоті після вживання вуглеводів, порушують рівновагу між демінералізацією та ремінералізацією емалі. За тривалого або повторюваного кислотного навантаження посилюється втрата мінеральних компонентів емалі, що може сприяти формуванню початкових каріозних уражень [2].

Початковим клінічним проявом демінералізації емалі є ураження типу білої плями (рис. 1.1). Такі зміни характеризуються підповерхневою пористістю, зниженням прозорості емалі та появою матово-білої ділянки. На ранніх етапах цей процес може бути зворотним, оскільки емаль здатна до ремінералізації за наявності сприятливих умов у ротовій порожнині [3]. Тому контроль рН слини має практичне значення для спостереження за кислотно-лужним станом ротової порожнини та оцінювання умов, у яких відбуваються процеси демінералізації й ремінералізації.

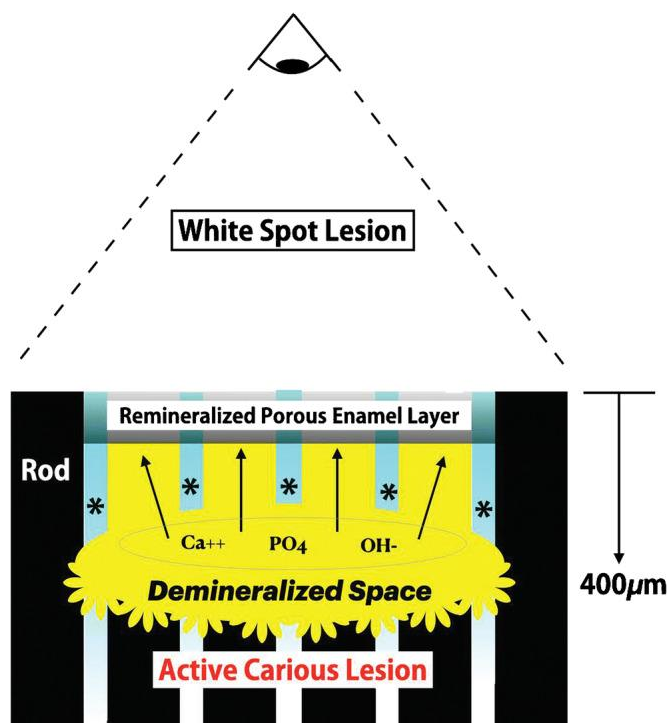


Рисунок 1.1 – Схематичне зображення утворення білої плями емалі внаслідок підповерхневої демінералізації [4]

Динаміку зміни кислотності ротового середовища після вживання вуглеводів описує крива Стефана (рис. 1.2). Вона відображає типову зміну рН зубного нальоту в часі: після надходження ферментованих вуглеводів мікроорганізми біоплівки швидко утворюють органічні кислоти, внаслідок чого рН різко знижується. Надалі, завдяки буферним властивостям слини, розведенню та поступовому видаленню кислот, рН повільно повертається до початкового рівня [6]. Особливе значення має так зване критичне значення рН, яке для емалі часто наводять близько 5,5. Якщо рН зубного нальоту знижується нижче цього рівня, створюються умови для розчинення мінеральних компонентів емалі та посилення демінералізації. За нормальних умов після одноразового кислотного навантаження слина поступово нейтралізує кислоти, і середовище повертається до зони, сприятливої для ремінералізації. Проте при частому вживанні цукровмісних продуктів або напоїв рН багаторазово знижується, а час перебування емалі в зоні демінералізації збільшується [6].

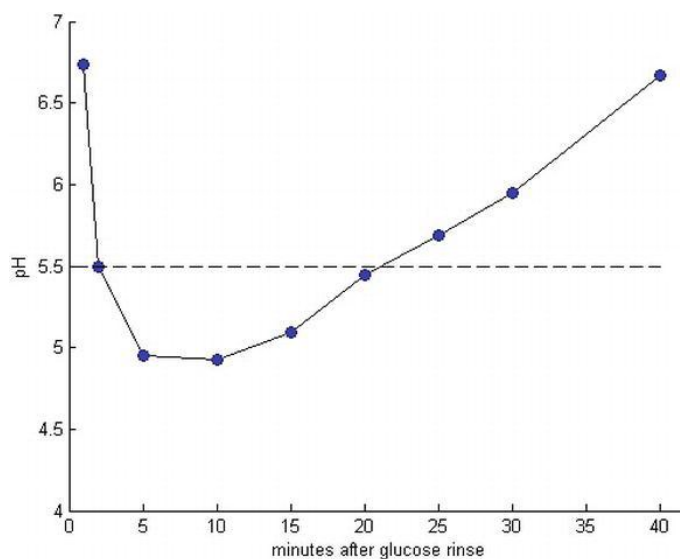


Рисунок 1.2 – Крива Стефана [7]

Особливістю слини як діагностичного середовища є її доступність і неінвазивність отримання. На відміну від крові, зразки слини можна збирати багаторазово без травматизації пацієнта, що є важливим для довготривалого моніторингу. Водночас слина є варіабельним середовищем, тому результати вимірювання можуть залежати від умов відбору проби. Для підвищення

відтворюваності результатів необхідно враховувати попереднє вживання їжі та напоїв, чищення зубів, використання ополіскувачів, фізичне навантаження, час доби та інші чинники [8].

1.2 Методи вимірювання рН слини та біологічних рідин

Вимірювання рН біологічних рідин є поширеним методом оцінювання кислотно-лужного стану середовища. Для слини цей показник має особливе значення, оскільки він відображає баланс між утворенням кислот у ротовій порожнині та здатністю слини нейтралізувати їх. На практиці для визначення рН слини застосовують індикаторні, електрохімічні та оптичні методи вимірювання [1, 2].

Найпростішим способом оцінювання кислотності є використання індикаторних рН-смужок. Їхній принцип роботи ґрунтується на зміні кольору індикатора залежно від концентрації іонів водню у зразку. До переваг такого методу належать доступність, простота використання та відсутність потреби у складному обладнанні. Однак індикаторні смужки мають обмежену точність, залежать від якості освітлення, часу контакту зі зразком і суб'єктивного сприйняття кольору користувачем. Тому цей метод доцільний переважно для орієнтовної оцінки рН, але є недостатнім для точного динамічного моніторингу [2].

Більш точним методом є електрохімічне вимірювання рН за допомогою рН-метрів. У таких приладах використовується вимірювальний електрод і електрод порівняння, між якими виникає різниця потенціалів, залежна від активності іонів водню у досліджуваному середовищі. Отриманий електричний сигнал перетворюється у числове значення рН. Саме електрохімічні рН-метри забезпечують вищу точність і відтворюваність результатів порівняно з індикаторними смужками, що робить їх придатними для клінічних і дослідницьких вимірювань [9]. Лабораторні рН-метри зазвичай мають високу

точність, можливість калібрування за буферними розчинами та стабільні метрологічні характеристики. Водночас вони не завжди зручні для регулярного персонального використання, оскільки мають більші габарити, потребують підготовки електрода, контролю умов вимірювання та обслуговування. Портативні цифрові рН-метри є компактнішими та зручнішими, однак більшість таких пристроїв переважно призначена лише для відображення поточного значення рН без автоматичного збереження результатів, побудови графіків і статистичного аналізу.

Окрему групу становлять оптичні методи вимірювання рН. Вони базуються на зміні оптичних властивостей індикаторної речовини або сенсорного шару залежно від кислотності середовища. До таких змін можуть належати зміна кольору, інтенсивності флуоресценції або інших оптичних параметрів. Оптичні підходи є перспективними для мініатюрних сенсорних систем, проте їх практичне застосування для слини ускладнюється впливом забарвлення зразка, домішок, залишків їжі, мікроорганізмів та інших компонентів слинної матриці [1].

1.3 Сенсори для вимірювання рН та особливості їх застосування у біомедичних системах

Сенсор для вимірювання рН є первинним перетворювачем, який реагує на зміну активності іонів водню у досліджуваному середовищі та формує електричний або оптичний сигнал, пов'язаний із кислотністю зразка. У біомедичних системах такі сенсори застосовують для аналізу біологічних рідин, зокрема слини, сечі, крові, поту та інших середовищ, у яких зміна рН може мати діагностичне або моніторингове значення [1].

Найпоширенішим технічним підходом до вимірювання рН є потенціометричний метод. Його принцип полягає у вимірюванні різниці потенціалів між вимірювальним електродом, чутливим до іонів водню, та

електродом порівняння зі стабільним потенціалом (рис. 1.3). Отримана різниця потенціалів залежить від кислотності середовища і після перетворення може бути подана у вигляді числового значення pH. Саме цей принцип покладено в основу більшості лабораторних і портативних цифрових pH-метрів [9].

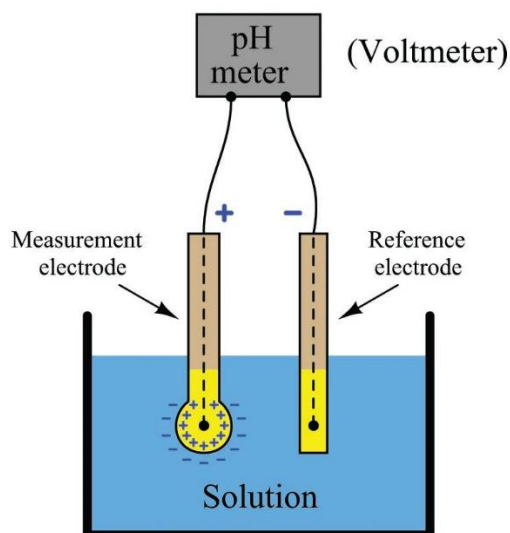


Рисунок 1.3 – Схема потенціометричного вимірювання pH [9]

Класичним варіантом pH-сенсора є скляний електрод, мембрана якого вибірково реагує на активність іонів водню (рис. 1.4). Такий електрод характеризується достатньою точністю та широким діапазоном вимірювання, однак потребує обережного поводження, періодичного калібрування та правильного зберігання. Для роботи з біологічними рідинами важливо враховувати можливе забруднення поверхні електрода білками, залишками їжі, мікроорганізмами та іншими компонентами слинної матриці, що може впливати на стабільність показів [1].

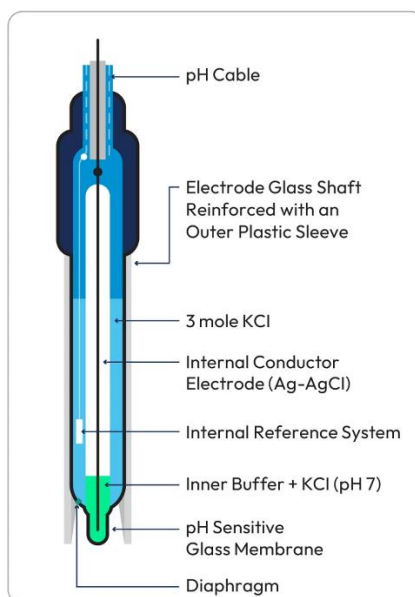


Рисунок 1.4 – Будова комбінованого рН-електрода [10]

У портативних електронних системах рН-електрод зазвичай використовується разом із вимірювальним модулем, який виконує узгодження сигналу з мікроконтролером. Оскільки вихідний сигнал рН-електрода є слабким і потребує високого входного опору вимірювального кола, безпосереднє підключення електрода до плати Arduino є недоцільним. Тому застосовується рН-модуль, який перетворює сигнал електрода у напругу, придатну для зчитування аналоговим входом мікроконтролера. Після цього Arduino виконує аналого-цифрове перетворення, оброблення сигналу та розрахунок значення рН.

Важливою умовою коректної роботи рН-сенсора є калібрування. Воно виконується за допомогою буферних розчинів із відомим значенням рН, наприклад рН 4,00; 6,86 або 7,00; 9,18. Калібрування дозволяє встановити відповідність між напругою на виході сенсорного модуля та реальним значенням рН. Без калібрування отримані результати можуть мати значну похибку і використовуватися лише як орієнтовні. Для біомедичного застосування, особливо під час порівняння результатів у динаміці, процедура калібрування є обов'язковою.

Специфічною проблемою сенсорів, що працюють у ротовій порожнині або зі зразками слини, є вплив біоплівки та органічних компонентів слини. Слина містить білки, муцини, ферменти, іони та мікроорганізми, які можуть осідати на

поверхні сенсора. Це явище погіршує контакт чутливої поверхні з досліджуваним середовищем, уповільнює відгук і може спричиняти дрейф показів [1]. Тому під час розроблення пристроїв для аналізу слини необхідно передбачати очищення електрода, контроль умов вимірювання та періодичну перевірку точності.

1.4 Огляд існуючих пристроїв-аналогів для вимірювання рН

Для обґрунтування доцільності розроблення пристрою для реєстрації рН слини було проаналізовано сучасні портативні та лабораторно-навчальні рішення, які можуть використовуватися для вимірювання рН рідких зразків. Основними критеріями порівняння були: принцип вимірювання, діапазон рН, точність, наявність калібрування, можливість передавання або збереження даних, придатність до роботи з малими об'ємами зразка та зручність для динамічного моніторингу.

Одним із найбільш близьких аналогів є *HORIBA LAQUAtwin pH-22* (рис. 1.5). Це кишеньковий рН-метр, призначений для роботи з малими об'ємами рідини. Пристрій може виконувати вимірювання з мінімальним об'ємом зразка 0,1 мл, підтримує автоматичне калібрування, автоматичну температурну компенсацію, функцію утримання стабільного значення та має пиловологозахист *IP67* [11]. Така конструкція є зручною для аналізу біологічних рідин, зокрема слини, оскільки не потребує великої кількості проби. Водночас пристрій переважно виконує функцію автономного вимірювання і не орієнтований на повноцінне накопичення історії вимірювань у мобільному додатку.



Рисунок 1.5 – Кишеньковий рН-метр *HORIBA LAQUAtwin pH-22* [11]

Іншим сучасним рішенням є *Hanna Instruments HALO2* — бездротовий рН-тестер із підтримкою *Bluetooth* (рис. 1.6). Особливістю цього пристрою є можливість підключення до смартфона або планшета через застосунок *Hanna Lab App*, що дозволяє переглядати рН і температуру в реальному часі, виконувати калібрування, зберігати та передавати результати [12]. За функціональністю цей пристрій є найближчим до концепції розроблюваної системи, оскільки поєднує рН-електрод, бездротовий зв'язок і мобільний інтерфейс. Основним обмеженням для навчально-інженерної розробки є те, що це готове комерційне рішення із закритою апаратною архітектурою, яке не передбачає вільної модифікації алгоритмів обробки або конструкції пристрою.



Рисунок 1.6 – Бездротовий рН-тестер *Hanna Instruments HALO2* [12]

До поширених портативних рН-метрів належить *Milwaukee PH55 PRO* (рис. 1.7). Пристрій має діапазон вимірювання від -2,0 до 16,0 рН, роздільну здатність 0,1 рН, точність $\pm 0,1$ рН, автоматичну температурну компенсацію та змінний електрод [13]. Його перевагами є простота використання, водозахищене виконання та можливість калібрування. Недоліком у контексті цієї дипломної роботи є відсутність вбудованого бездротового передавання даних і функцій статистичного аналізу результатів.



Рисунок 1.7 – Портативний рН-метр Milwaukee PH55 PRO [13]

Подібним за призначенням є *Apera PH60* — портативний цифровий рН-тестер, який підтримує калібрування за стандартними буферними розчинами та використовується для точного вимірювання рН рідких середовищ (рис. 1.8) [14]. Такі пристрої забезпечують достатню точність для побутових, навчальних і лабораторних вимірювань, однак здебільшого обмежуються індикацією поточного значення рН. Тому їх використання для довготривалого моніторингу потребує ручного запису результатів або додаткового програмного забезпечення.



Рисунок 1.8 – Портативний цифровий рН-тестер Apera PH60 [14]

Окрему групу становлять навчально-лабораторні цифрові сенсори, наприклад *Vernier Go Direct pH Sensor* (рис. 1.9). Цей сенсор призначений для вимірювання рН водних розчинів і може підключатися до пристрою через *Bluetooth* або *USB* [15]. Його перевагою є наявність цифрового інтерфейсу та сумісність із програмним забезпеченням для побудови графіків. Водночас такий пристрій орієнтований переважно на освітні лабораторні роботи, а не на створення спеціалізованої системи персонального моніторингу рН слини.



Рисунок 1.9 – Цифровий навчально-лабораторний рН-сенсор Vernier Go Direct pH Sensor [15]

У таблиці 1.1 наведена порівняльна характеристика вище описаних пристроїв-аналогів.

Таблиця 1.1 – Порівняльна характеристика пристроїв-аналогів для вимірювання рН

Пристрій	Тип пристрою	Основні характеристики	Переваги	Обмеження
<i>HORIBA LAQUAtwin pH-22</i>	Кишеньковий рН-метр	Малий об'єм зразка від 0,1 мл, автоматичне калібрування, температурна компенсація, IP67 [11]	Зручний для малих проб, компактний, автономний	Немає повноцінної мобільної аналітики та гнучкої модифікації
<i>Hanna Instruments HALO2</i>	Бездротовий рН-тестер	Bluetooth, мобільний застосунок, вимірювання рН і температури, збереження та передавання даних [12]	Найближчий функціональний аналог за наявністю мобільної інтеграції	Комерційна замкнена система, обмежена можливість модифікації
<i>Milwaukee PH55 PRO</i>	Портативний рН-метр	Діапазон - 2,0...16,0 рН, точність $\pm 0,1$ рН, АТС, змінний електрод [13]	Простий, водозахищений, придатний для польових вимірювань	Немає Bluetooth, графіків і статистики
<i>Apera PH60</i>	Портативний рН-тестер	Цифрове вимірювання рН, калібрування за буферними розчинами [14]	Доступний, компактний, точніший за індикаторні методи	Орієнтований переважно на разове вимірювання
<i>Vernier Go Direct pH Sensor</i>	Цифровий навчально-лабораторний рН-сенсор	Підключення через Bluetooth або USB, діапазон 0–14 рН [15]	Цифровий інтерфейс, можливість побудови графіків	Більше орієнтований на лабораторне навчальне використання
<i>DFRobot Gravity Analog pH Sensor/Meter Kit V2</i>	Мікроконтролерний рН-модуль	Аналоговий вихід, живлення 3,3–5,5 В, BNC-роз'єм, сумісність з Arduino, двоточкове калібрування [16]	Відкрита архітектура, зручний для прототипування, сумісний з Arduino	Потребує самостійної калібровки, програмної обробки та перевірки точності

Проведене порівняння показує, що більшість портативних рН-метрів забезпечують достатню точність вимірювання, але переважно орієнтовані на автономне відображення поточного значення рН. Пристрої з *Bluetooth*-інтеграцією, зокрема *Hanna HALO2* і *Vernier Go Direct pH Sensor*, мають

розширені можливості передавання даних, однак є готовими комерційними системами із закритою архітектурою.

1.5 Формування медико-технічних вимог до пристрою для реєстрації та аналізу рН слини

На основі аналізу фізіологічного значення слини, методів вимірювання рН, особливостей рН-сенсорів та існуючих пристроїв-аналогів можна сформувати медико-технічні вимоги до розроблюваної системи. Основним призначенням пристрою є реєстрація рН слини як показника кислотно-лужного стану ротової порожнини з подальшим збереженням, візуалізацією та статистичним аналізом результатів у мобільному додатку.

Оскільки рН слини є динамічним показником і може змінюватися під впливом харчування, гігієнічних процедур, активності мікрофлори та швидкості слиновиділення, пристрій має бути орієнтований не лише на одноразове вимірювання, а й на регулярне спостереження впродовж певного періоду [1, 8]. Такий підхід дозволяє оцінювати не окреме ізольоване значення, а загальну тенденцію зміни кислотно-лужного стану ротової порожнини.

З медичної точки зору пристрій не повинен розглядатися як самостійний засіб діагностики карієсу або інших стоматологічних захворювань. Його доцільно визначати як допоміжну систему моніторингу рН слини, яка може використовуватися для спостереження за змінами ротового середовища, зокрема в контексті профілактики демінералізації емалі та контролю умов, що впливають на ремінералізацію [2, 3].

Враховуючи результати огляду аналогів, розроблюваний пристрій має поєднувати функції портативного рН-метра та цифрової системи накопичення даних. На відміну від більшості звичайних портативних рН-метрів, які переважно відображають лише поточне значення, система повинна забезпечувати передавання результатів до мобільного додатку, збереження

історії вимірювань, побудову графіків і розрахунок статистичних показників [11–16].

Основні медико-технічні вимоги до пристрою наведено в таблиці 1.2.

Таблиця 1.2 – Медико-технічні вимоги до пристрою для реєстрації та аналізу рН слини

Група вимог	Параметр	Вимога
Медичне призначення	Контрольований показник	рН слини як показник кислотно-лужного стану ротової порожнини
Медичне призначення	Тип використання	Допоміжний моніторинг, не самостійна клінічна діагностика
Біологічний зразок	Досліджуване середовище	Слина або модельні розчини під час калібрування й тестування
Біологічний зразок	Умови вимірювання	Бажано проводити вимірювання за стандартизованих умов: до/після їжі, у визначений час, із фіксацією дати та часу
Вимірювальний канал	Метод вимірювання	Електрохімічний потенціометричний метод
Вимірювальний канал	Сенсор	рН-електрод із модулем узгодження сигналу
Вимірювальний канал	Орієнтовний діапазон вимірювання	Не менше 4,0–9,0 рН для покриття фізіологічно значущих змін слини; бажано 0–14 рН відповідно до можливостей сенсора
Вимірювальний канал	Роздільна здатність	Не гірше 0,1 рН
Вимірювальний канал	Калібрування	Можливість калібрування за буферними розчинами з відомим рН
Апаратна частина	Мікроконтролер	<i>Arduino</i> або сумісна мікроконтролерна платформа з аналоговим входом
Апаратна частина	Індикація	Відображення поточного значення рН на <i>LCD</i> -дисплеї
Апаратна частина	Бездротовий зв'язок	Передавання даних до мобільного додатку через <i>Bluetooth</i> -модуль
Програмна частина	Обробка даних	Перетворення аналогового сигналу сенсора у значення рН
Програмна частина	Збереження результатів	Запис результату вимірювання з датою та часом
Програмна частина	Аналіз результатів	Розрахунок середнього, мінімального та максимального значення рН
Програмна частина	Візуалізація	Побудова графіка зміни рН у часі
Мобільний додаток	Основні функції	Підключення до пристрою, приймання даних, перегляд історії, статистика, графіки
Конструкція	Корпус	Розміщення <i>Arduino</i> , рН-модуля, <i>Bluetooth</i> -модуля, дисплея та елементів живлення
Конструкція	Зручність експлуатації	Доступ до роз'ємів, можливість підключення рН-електрода, зручність перенесення

Продовження таблиці 1.2

Безпека	Електробезпека	Живлення від низьковольтного джерела, електричне відокремлення користувача від небезпечних напруг
Експлуатація	Обслуговування	Очищення електрода після вимірювання, періодична перевірка точності та повторне калібрування

Сформовані вимоги визначають загальну архітектуру майбутньої системи. Апаратна частина повинна забезпечувати зчитування сигналу з рН-сенсора, його первинну обробку та передавання результату. Програмна частина мікроконтролера має виконувати перетворення аналогового сигналу у значення рН, а мобільний додаток — забезпечувати збереження й аналіз даних.

Конструктивні вимоги пов'язані з необхідністю створення портативного прототипу. Корпус має забезпечувати розміщення основних електронних компонентів, механічний захист плати та зручний доступ до дисплея, роз'ємів і сенсорного входу. Оскільки пристрій працює з біологічною рідиною, електронні компоненти мають бути відокремлені від зони контакту зі зразком.

Таким чином, розроблювана система повинна поєднувати функції вимірювального пристрою, засобу передавання даних і програмного інструменту для аналізу результатів. Саме інтеграція рН-сенсора, мікроконтролера, бездротового зв'язку та мобільного додатку є основною відмінністю запропонованого рішення від більшості простих портативних рН-метрів.

Висновок до розділу 1

У першому розділі було розглянуто теоретичні основи використання рН слини як показника кислотно-лужного стану ротової порожнини. Встановлено, що слина є інформативним біологічним середовищем, яке бере участь у підтриманні мінерального балансу емалі, нейтралізації кислот і забезпеченні умов для ремінералізації. Значення рН слини є динамічним показником, який може змінюватися під впливом харчування, гігієнічних процедур, мікробіологічної активності, швидкості слиновиділення та часу доби.

Проаналізовано основні методи вимірювання рН біологічних рідин, зокрема індикаторні, електрохімічні та оптичні методи. Визначено, що для розроблення портативного пристрою найбільш доцільним є електрохімічний потенціометричний метод із використанням рН-електрода, оскільки він забезпечує можливість отримання числового результату та подальшого програмного оброблення даних.

Також було розглянуто особливості рН-сенсорів і проаналізовано існуючі пристрої-аналоги для вимірювання рН. Порівняння показало, що більшість портативних рН-метрів орієнтовані переважно на разове вимірювання та відображення поточного значення, тоді як можливості збереження, візуалізації та статистичного аналізу результатів часто є обмеженими. На основі проведеного аналізу сформовано медико-технічні вимоги до розроблюваної системи, яка повинна поєднувати вимірювання рН, бездротове передавання даних і подальший аналіз результатів у мобільному додатку.

РОЗДІЛ 2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ ПРИСТРОЮ

2.1 Функціональна схема та загальний принцип роботи пристрою

Розроблюваний пристрій призначений для реєстрації рН слини, локального відображення результату та передавання виміряного значення до мобільного додатку для подальшого збереження й аналізу. Система побудована за модульним принципом, що спрощує підключення окремих компонентів, налагодження схеми та подальшу модернізацію пристрою.

Основними функціональними блоками пристрою є рН-електрод, рН-модуль узгодження сигналу, мікроконтролер *Arduino Uno*, *LCD*-дисплей з *I2C*-інтерфейсом, *Bluetooth*-модуль *HM-10*, блок живлення та мобільний додаток. Загальна структура системи наведена на рис. 2.1.

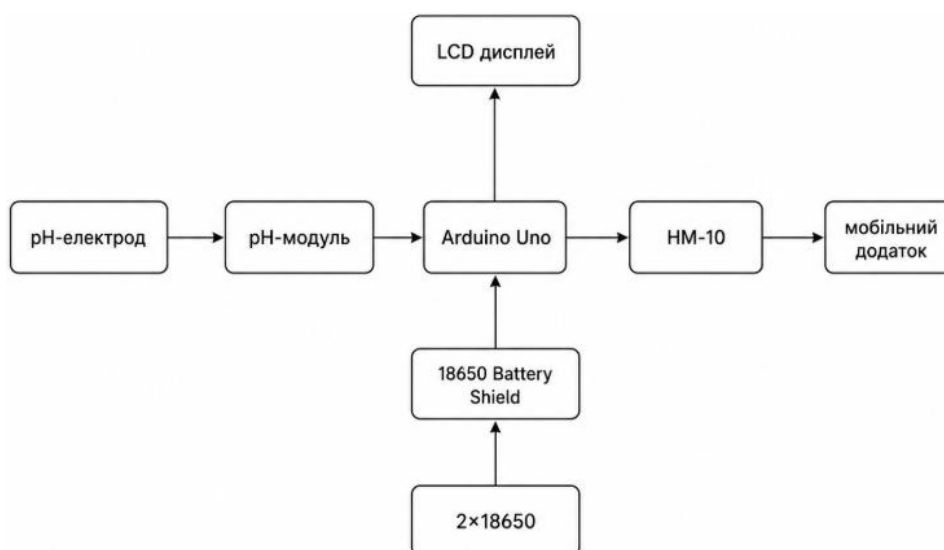


Рисунок 2.1 – Функціональна схема пристрою

Сенсорна частина пристрою представлена комбінованим рН-електродом, який контактує зі зразком слини та формує сигнал, пов'язаний з активністю іонів водню. Через слабкий рівень сигналу електрод підключається до рН-модуля через *BNC*-роз'єм. Модуль перетворює сигнал електрода в аналогову напругу, придатну для зчитування *Arduino* [17]. Центральним блоком пристрою є *Arduino Uno*, що зчитує аналоговий сигнал із рН-модуля, виконує аналого-цифрове

перетворення, первинне оброблення даних і розрахунок значення рН. Використання *Arduino Uno* є доцільним завдяки наявності аналогових входів, цифрових інтерфейсів, робочій напрузі 5 В і підтримці бібліотек для дисплеїв та модулів зв'язку [18]. Результат вимірювання виводиться на *LCD*-дисплей 16×2 з *I2C*-адаптером, що дозволяє користувачу бачити поточне значення рН без підключення до смартфона. *I2C*-інтерфейс зменшує кількість з'єднань, оскільки використовує лише лінії *SDA* та *SCL* [19]. Для передавання даних у мобільний додаток використовується *Bluetooth*-модуль *HM-10*, який забезпечує бездротовий зв'язок між *Arduino* та смартфоном через *UART*-інтерфейс [20]. Мікроконтролер формує текстове повідомлення з результатом вимірювання та передає його до додатку. Мобільний додаток відображає отримане значення рН, зберігає його з датою і часом, формує історію вимірювань, будує графік зміни рН та розраховує статистичні показники. Живлення пристрою здійснюється від низьковольтного джерела: *USB*-живлення *Arduino* або автономного акумуляторного блока. Такий варіант є безпечним і зручним для експлуатації прототипу.

2.2 Вибір компонентів та їх технічні характеристики

Вибір компонентів для розроблюваного пристрою виконувався з урахуванням функціонального призначення системи: вимірювання рН слини, первинне оброблення сигналу, локальне відображення результату, бездротове передавання даних до мобільного додатку та автономне живлення. Оскільки пристрій розробляється як макетний прототип, перевага надавалася доступним модулям, сумісним з *Arduino*, що мають просту схему підключення та достатні характеристики для реалізації поставленої задачі.

Основним вимірювальним елементом системи є рН-сенсорний модуль із комбінованим рН-електродом (рис. 2.2). Його призначення полягає у перетворенні зміни активності іонів водню в досліджуваному розчині на

електричний сигнал. Оскільки електродний сигнал має малу амплітуду та потребує високого вхідного опору вимірювального кола, рН-електрод підключається не безпосередньо до *Arduino*, а через спеціальний модуль узгодження сигналу.



Рисунок 2.2 – Комбінований рН-електрод з BNC-роз'ємом [16]

У роботі використовується рН-модуль з *BNC*-роз'ємом для підключення електрода та аналоговим виходом *Po*, який подається на аналоговий вхід мікроконтролера (рис. 2.3). Такий модуль є зручним для макетування, оскільки забезпечує сумісність з *Arduino* та дозволяє виконувати програмне перетворення вихідної напруги у значення рН [17].



Рисунок 2.3 – рН-сенсорний модуль з BNC-роз'ємом та аналоговим виходом
[16]

Центральним елементом апаратної частини є плата *Arduino Uno*, яка виконує функції зчитування, оброблення та передавання даних (рис. 2.4). Вибір цієї плати обумовлений її доступністю, простотою програмування, наявністю аналогових входів і достатньою кількістю цифрових виводів для підключення додаткових модулів. *Arduino Uno* побудована на мікроконтролері *ATmega328P*, має 6 аналогових входів, 14 цифрових входів/виходів, працює з тактовою частотою 16 МГц [18].



Рисунок 2.4 – Плата мікроконтролера *Arduino Uno* [18]

У межах цієї роботи аналоговий вхід *A0* використовується для зчитування сигналу з рН-модуля, лінії *A4* та *A5* — для підключення *LCD*-дисплея через *I2C*, а цифрові виводи *D10* і *D11* — для програмного *UART*-зв’язку з *Bluetooth*-модулем *HM-10*.

Основні технічні характеристики *Arduino Uno* наведено в таблиці 2.2.

Таблиця 2.2 – Технічні характеристики *Arduino Uno*

Параметр	Значення
Мікроконтролер	<i>ATmega328P</i>
Робоча напруга	5 В
Рекомендована вхідна напруга	7–12 В
Цифрові входи/виходи	14
<i>PWM</i> -виходи	6
Аналогові входи	6
Роздільна здатність АЦП	10 біт

Для локального відображення результату вимірювання використовується *LCD*-дисплей 16×2 з *I2C*-адаптером (рис. 2.5). Дисплей дає змогу виводити поточне значення рН без необхідності постійного підключення до смартфона. Це підвищує зручність експлуатації пристрою під час тестування та калібрування.

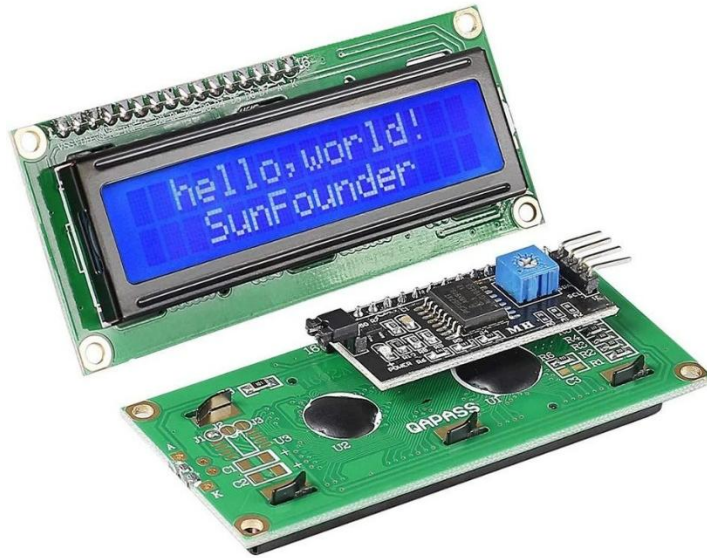


Рисунок 2.5 – *LCD*-дисплей 16×2 з *I2C*-адаптером [19]

Використання *I2C*-адаптера зменшує кількість з'єднань між дисплеєм і мікроконтролером. Замість паралельного підключення багатьох цифрових ліній застосовуються лише два сигнальні контакти: *SDA* та *SCL* [19]. У платі *Arduino Uno* цим лініям відповідають входи *A4* і *A5*.

Основні технічні характеристики *LCD*-дисплея 16×2 *I2C* наведено в таблиці 2.3.

Таблиця 2.3 – Основні характеристики *LCD*-дисплея 16×2 *I2C*

Параметр	Значення
Тип дисплея	Символьний <i>LCD</i>
Формат відображення	16 символів × 2 рядки
Інтерфейс	<i>I2C</i>
Лінії підключення	<i>VCC</i> , <i>GND</i> , <i>SDA</i> , <i>SCL</i>
Робоча напруга	5 В
Призначення в системі	Виведення поточного значення рН

Для бездротового передавання даних до мобільного додатку використовується *Bluetooth*-модуль *HM-10* (рис. 2.6). Цей модуль підтримує

технологію *Bluetooth Low Energy* та призначений для організації обміну даними між мікроконтролером і смартфоном [20].



Рисунок 2.6 – *Bluetooth Low Energy* модуль HM-10 [20]

HM-10 працює через *UART*-інтерфейс, тому для його підключення використовуються лінії *TXD* і *RXD*. У макетній схемі обмін з *Arduino* реалізується за допомогою програмного послідовного порту через цифрові виводи *D10* та *D11*. Такий підхід дозволяє залишити основний апаратний послідовний порт *Arduino* доступним для налагодження через *USB*.

Основні технічні характеристики *Bluetooth Low Energy HM-10* наведено в таблиці 2.4.

Таблиця 2.4 – Основні характеристики *Bluetooth*-модуля HM-10

Параметр	Значення
Тип модуля	<i>Bluetooth Low Energy</i>
Версія <i>Bluetooth</i>	4.0 BLE
Інтерфейс підключення	<i>UART</i>
Типова швидкість обміну	9600 біт/с
Лінії підключення	<i>VCC, GND, TXD, RXD</i>
Призначення в системі	Передавання значення pH до мобільного додатку

Для живлення пристрою може використовуватися *USB*-живлення від комп'ютера або автономний блок живлення. У макетній реалізації передбачено використання *18650 Battery Shield* з двома літій-іонними акумуляторами формату 18650 (рис. 2.7). Такий модуль забезпечує подавання напруги на *Arduino* через *USB*-вихід, що робить пристрій більш мобільним і не прив'язаним до стаціонарного комп'ютера [21].



Рисунок 2.7 – 18650 *Battery Shield* для автономного живлення пристрою [21]

До додаткових елементів системи належать макетна плата, з'єднувальні дроти, роз'єми та корпус. Макетна плата використовується для швидкого складання та перевірки схеми без пайки. Корпус призначений для розміщення компонентів, механічного захисту електроніки та забезпечення зручності експлуатації пристрою.

2.3 Збірка пристрою, підключення компонентів та ілюстрація схеми

Після вибору елементної бази було виконано макетну збірку пристрою для реєстрації рН слини. Основною метою цього етапу була перевірка сумісності компонентів, правильності електричних з'єднань, стабільності живлення, роботи дисплея та передавання даних через *Bluetooth*-модуль. На етапі прототипування компоненти були розміщені на безпайковій макетній платі, що дало змогу швидко змінювати з'єднання та перевіряти працездатність окремих вузлів системи.

Перед фізичним складанням було створено схему підключення у середовищі *Fritzing* (рис. 2.8). Це дозволило візуально перевірити розташування модулів, сигнальні лінії, шини живлення та відповідність підключення окремих виводів до *Arduino Uno* [22].

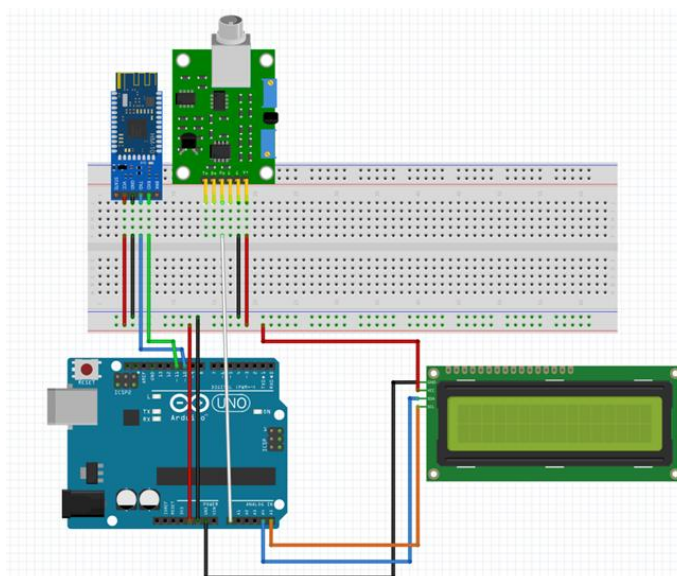


Рисунок 2.8 – Схема підключення компонентів пристрою у середовищі *Fritzing*.

Основним вимірювальним вузлом є рН-модуль, до якого через *BNC*-роз'єм підключається комбінований рН-електрод. Живлення рН-модуля здійснюється від виводу 5 V Arduino, загальний провід підключається до *GND*, а аналоговий вихід *Po* з'єднується з аналоговим входом *A0* мікроконтролера. Саме через цей вхід Arduino зчитує напругу, яка після програмного перетворення використовується для розрахунку значення рН [17].

LCD-дисплей 16×2 з *I2C*-адаптером підключено до *Arduino* через чотири виводи: *VCC*, *GND*, *SDA* та *SCL*. Лінії *SDA* і *SCL* відповідають аналоговим виводам *A4* і *A5* плати *Arduino Uno*. Використання *I2C*-інтерфейсу зменшує кількість з'єднувальних провідів і спрощує монтаж схеми [19].

Bluetooth-модуль *HM-10* підключено через *UART*-інтерфейс. У макетній реалізації для обміну даними використовувалися цифрові виводи *D10* і *D11*, що дозволило реалізувати програмний послідовний порт і залишити основний *USB*-порт *Arduino* доступним для завантаження скетчу та налагодження. Вивід *TXD* модуля *HM-10* підключено до *D10* *Arduino*, а *RXD* — до *D11*. Живлення модуля здійснювалося від шини 5 В та *GND* відповідно до використаного модуля [20].

Основні електричні з'єднання компонентів наведено в таблиці 2.6.

Таблиця 2.6 – Схема підключення компонентів пристрою

Компонент	Вивід/інтерфейс компонента	Підключення до <i>Arduino</i>
pH-модуль	V+	5 V
pH-модуль	G / GND	GND
pH-модуль	Po	A0
LCD 16×2 I2C	VCC	5 V
LCD 16×2 I2C	GND	GND
LCD 16×2 I2C	SDA	A4
LCD 16×2 I2C	SCL	A5
HM-10	VCC	5 V
HM-10	GND	GND
HM-10	TXD	D10
HM-10	RXD	D11
18650 Battery Shield	USB-вихід живлення	USB-роз'єм <i>Arduino Uno</i>
Акумулятори 18650	Контакти тримача <i>Battery Shield</i>	Встановлені у відсіки <i>Battery Shield</i>

Під час складання схеми всі модулі були об'єднані спільною лінією заземлення. Це є необхідною умовою коректної роботи системи, оскільки pH-модуль, дисплей, *Bluetooth*-модуль і *Arduino* повинні мати спільний електричний нульовий потенціал. Живлення основних модулів здійснювалося від стабілізованого виходу *Arduino*. Автономний блок живлення на основі 18650 *Battery Shield* не підключався безпосередньо до шин 5 V і GND макетної плати, а подавав живлення на *Arduino* через USB-роз'єм, після чого від *Arduino* живилися інші вузли пристрою.

Фізична збірка виконувалася у такій послідовності: спочатку було встановлено *Arduino Uno* та макетну плату, після цього підключено pH-модуль і перевірено наявність сигналу на аналоговому вході A0. Далі було під'єднано LCD-дисплей та перевірено його ініціалізацію через I2C-інтерфейс. Після цього до схеми було додано *Bluetooth*-модуль HM-10 і перевірено передавання даних до мобільного додатку. Завершальним етапом стало підключення автономного блока живлення та перевірка роботи пристрою без підключення до комп'ютера. Зібрана схема пристрою зображена на рисунку 2.9.

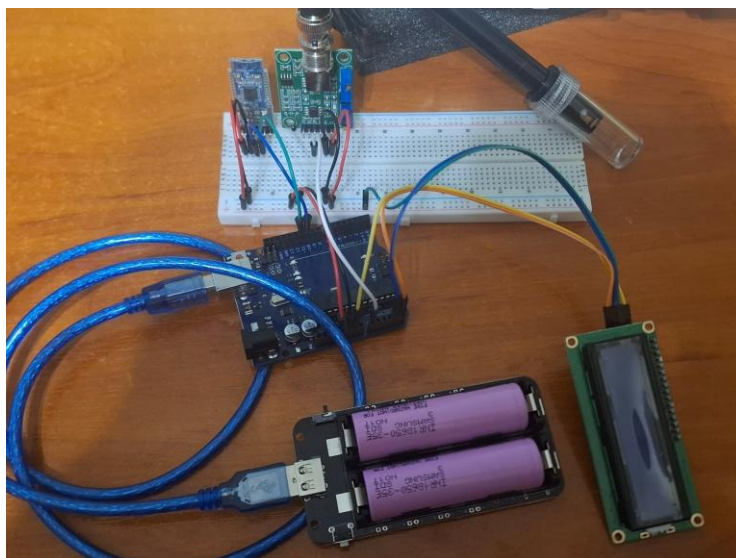


Рисунок 2.9 – Зовнішній вигляд зібраної макетної схеми пристрою

Після складання макета було виконано первинну перевірку працездатності основних вузлів. Перевірялося виведення значення рН на *LCD*-дисплей, зчитування аналогового сигналу з рН-модуля, передавання даних через *HM-10* та стабільність роботи від акумуляторного живлення. Зібрана схема підтвердила можливість реалізації основної функціональної логіки пристрою: отримання сигналу від рН-сенсора, оброблення на *Arduino*, локальна індикація результату та передавання значення до мобільного додатку.

2.4 Розроблення програмного забезпечення для мікроконтролера *Arduino*

Для забезпечення роботи апаратної частини пристрою було розроблено програмне забезпечення для мікроконтролера *Arduino Uno*. Основним завданням програми є зчитування аналогового сигналу з рН-модуля, його первинне оброблення, перетворення у значення рН, відображення результату на *LCD*-дисплеї та передавання даних до мобільного додатку через *Bluetooth*-модуль *HM-10*.

Програму було створено у середовищі *Arduino IDE*, яке використовується для написання, компіляції та завантаження скетчів у мікроконтролерні плати

Arduino. У програмі застосовано бібліотеки *Wire.h*, *LiquidCrystal_I2C.h* та *SoftwareSerial.h*. Бібліотека *Wire.h* використовується для роботи з I2C-інтерфейсом, *LiquidCrystal_I2C.h* — для керування LCD-дисплеєм 16×2, а *SoftwareSerial.h* — для організації програмного послідовного порту, через який здійснюється обмін даними з *Bluetooth*-модулем *HM-10*.

На початку програми задаються основні апаратні підключення. Аналоговий вихід рН-модуля *P0* підключений до входу *A0 Arduino*, тому в коді цей вхід визначено як основний канал зчитування сигналу. Для роботи з *Bluetooth*-модулем використано цифрові виводи *D10* і *D11*, які виконують роль програмних ліній *RX* і *TX*. Такий підхід дозволяє залишити апаратний послідовний порт *Arduino* доступним для налагодження через *USB*.

Основна логіка програми передбачає такі етапи:

1. Ініціалізація послідовного порту, *Bluetooth*-модуля та LCD-дисплея.
2. Виведення службового повідомлення про готовність пристрою.
3. Очікування команди від мобільного додатку.
4. Зчитування аналогового сигналу з рН-модуля.
5. Усереднення отриманих значень для зменшення шуму.
6. Перетворення значення АЦП у напругу.
7. Розрахунок орієнтовного значення рН.
8. Виведення результату на дисплей.
9. Передавання результату до мобільного додатку у заданому текстовому форматі.

Для взаємодії з мобільним додатком у програмі передбачено систему текстових команд. Команда *PING* використовується для перевірки зв'язку між додатком і пристроєм. Команда *READ* запускає одиничне вимірювання. Команда *START* вмикає періодичне передавання значень рН, а команда *STOP* зупиняє потік даних. Така структура команд дає змогу керувати пристроєм безпосередньо з мобільного додатку та контролювати режим його роботи.

Зчитування сигналу від рН-модуля виконується за допомогою функції *analogRead()*. Оскільки аналоговий сигнал може містити випадкові коливання, у програмі застосовано усереднення кількох вимірювань. Для підвищення стабільності результату з вибірки відкидаються найменше та найбільше значення, після чого обчислюється середнє. Це дозволяє частково зменшити вплив короточасних шумів і нестабільності сигналу.

Перетворення цифрового значення АЦП у напругу виконується за формулою 2.1:

$$U = \frac{ADC \cdot U_{ref}}{1023}, \quad (2.1)$$

де U — напруга на аналоговому вході, В; ADC — усереднене значення аналого-цифрового перетворювача; U_{ref} — опорна напруга *Arduino*, що дорівнює 5 В; 1023 — максимальне значення 10-бітного АЦП *Arduino Uno*.

Після цього значення напруги перетворюється у рН за лінійною залежністю 2.2:

$$pH = C - k \cdot U, \quad (2.2)$$

де C — калібрувальний коефіцієнт, заданий у програмі; k — коефіцієнт нахилу характеристики рН-сенсора; U — виміряна напруга на аналоговому вході.

У тестовій версії програми використовується калібрувальний коефіцієнт, який надалі коригується під час калібрування рН-сенсора за буферними розчинами. Це необхідно, оскільки фактична залежність між напругою та рН може відрізнятися залежно від конкретного електрода, рН-модуля, температури та умов вимірювання.

Результат вимірювання виводиться на *LCD*-дисплей у вигляді поточного значення рН. Одночасно *Arduino* формує текстове повідомлення для мобільного додатку у форматі: РН:7.12

Такий формат було обрано для спрощення обробки даних у мобільному додатку. Префікс *РН*: дозволяє програмі відрізнити результат вимірювання від

службових повідомлень, а числове значення після двокрапки використовується для відображення, збереження та статистичного аналізу.

Також у програмі передбачено передавання службових статусів, зокрема *READY*, *STARTED*, *STOPPED* та *UNKNOWN_COMMAND*. Вони використовуються для інформування мобільного додатку про стан пристрою: готовність до роботи, початок вимірювання, зупинку передавання або отримання невідомої команди.

Повний лістинг коду для роботи схеми наведено у додатку А.

2.5 Калібрування рН-сенсора, принцип і процес вимірювання

Калібрування та перевірка рН-сенсора є необхідним етапом підготовки пристрою до роботи, оскільки вихідна напруга рН-модуля залежить від характеристик конкретного електрода, стану його чутливої мембрани, стабільності живлення, температури середовища та параметрів вимірювального модуля. У межах розроблення макетного прототипу основною задачею було не метрологічне калібрування лабораторного рівня, а перевірка того, чи коректно сенсор реагує на зміну кислотності середовища та чи правильно *Arduino* перетворює аналоговий сигнал у числове значення рН.

Для практичної перевірки роботи сенсора було використано модельні водні розчини зі штучно зміненим рівнем рН (табл. 2.9). Такий підхід дозволяє відтворити зміну кислотності в контрольованих умовах без використання спеціальних лабораторних буферних розчинів. Як базове середовище використовували склянку води, після чого рН змінювали у кислий або лужний бік за допомогою доступних речовин.

Підвищення рН моделювали шляхом додавання невеликої кількості харчової соди, тобто гідрокарбонату натрію. Для цього до склянки води додавали приблизно 1/8–1/4 чайної ложки соди, після чого розчин перемішували та занурювали в нього рН-електрод. Іншим способом отримання слабколужного

середовища є кип'ятіння води протягом 10–15 хвилин, оскільки під час кип'ятіння з води видаляється частина розчиненого вуглекислого газу, що може сприяти підвищенню рН до слабколужних значень.

Пониження рН моделювали шляхом додавання до води кількох крапель лимонного соку або невеликої кількості лимонної кислоти. Також для створення слабокислого середовища можна використовувати 1–2 краплі натурального яблучного оцту. Після додавання кислого компонента розчин перемішували, очікували стабілізації показів і фіксували значення, яке виводилося пристроєм.

Таблиця 2.9 – Способи моделювання зміни рН водного середовища

Напрямок зміни рН	Речовина або метод	Очікуваний ефект
Підвищення рН	Додавання невеликої кількості харчової соди	Зміщення середовища в лужний бік
Підвищення рН	Кип'ятіння води протягом 10–15 хвилин	Часткове видалення CO_2 і формування слабколужного середовища
Пониження рН	Додавання кількох крапель лимонного соку	Зміщення середовища в кислий бік
Пониження рН	Додавання кількох крупинок лимонної кислоти	Виражене зниження рН залежно від кількості речовини
Пониження рН	Додавання 1–2 крапель яблучного оцту	Формування слабокислого середовища

Перед початком перевірки рН-електрод промивали дистильованою водою та обережно видаляли залишки рідини з його поверхні. Спочатку виконували вимірювання у звичайній воді, яка використовувалася як умовний початковий зразок. Після цього послідовно проводили вимірювання в розчинах із підвищеним і зниженим рН. Така послідовність дозволяла оцінити, чи змінюються покази пристрою відповідно до очікуваного напрямку: при додаванні соди значення рН повинно зростати, а при додаванні лимонного соку, лимонної кислоти або оцту — зменшуватися.

Під час кожного вимірювання електрод занурювали у зразок так, щоб чутлива частина повністю контактувала з рідиною. Після занурення очікували стабілізації сигналу, оскільки покази рН-сенсора можуть змінюватися протягом перших секунд контакту з новим середовищем. Після стабілізації *Arduino*

зчитувала аналоговий сигнал із виходу рН-модуля, виконувала усереднення кількох значень і перетворювала результат у напругу.

У процесі налаштування програмні коефіцієнти коригували таким чином, щоб пристрій адекватно відображав зміну кислотності у модельних розчинах. Послідовність перевірки та налаштування рН-сенсора описана у таблиці 2.10. При цьому важливо враховувати, що такий спосіб не замінює повноцінного лабораторного калібрування за еталонними буферними розчинами або перевірки за сертифікованим рН-метром. Однак для макетного прототипу він є придатним для перевірки працездатності сенсора, напряму зміни показів і стабільності роботи програмно-апаратної системи.

Таблиця 2.10 – Послідовність перевірки та налаштування рН-сенсора

Етап	Опис дії
Підготовка електрода	Промивання електрода дистильованою водою перед вимірюванням
Початкове вимірювання	Занурення електрода у воду та фіксація базового значення
Моделювання лужного середовища	Додавання соди або використання кип'яченої води
Моделювання кислого середовища	Додавання лимонного соку, лимонної кислоти або яблучного оцту
Зчитування сигналу	Отримання аналогового значення з рН-модуля через вхід А0
Обробка результату	Усереднення сигналу та перетворення напруги у рН
Перевірка реакції	Оцінка відповідності зміни показів очікуваному напряму
Очищення електрода	Промивання після кожного вимірювання для уникнення змішування зразків

Після перевірки на модельних розчинах пристрій може бути використаний для вимірювання рН слини. Для цього зразок слини збирають у чисту ємність, після чого електрод занурюють у рідину та очікують стабілізації показів. Отримане значення виводиться на *LCD*-дисплей і передається через *Bluetooth*-модуль до мобільного додатку. У додатку результат зберігається разом із датою та часом, що дозволяє надалі аналізувати зміну рН у динаміці.

Висновок до розділу 2

У другому розділі було розроблено апаратну частину пристрою для реєстрації рН слини. Сформовано функціональну схему системи, яка включає рН-електрод, рН-модуль узгодження сигналу, мікроконтролер *Arduino Uno*, *LCD*-дисплей з *I2C*-інтерфейсом, *Bluetooth*-модуль *HM-10*, блок живлення та мобільний додаток. Така структура забезпечує повний цикл роботи пристрою: отримання сигналу від сенсора, його оброблення, локальне відображення результату та передавання даних на смартфон.

Було обґрунтовано вибір основних компонентів пристрою. *Arduino Uno* використано як центральний блок керування завдяки наявності аналогових входів, цифрових інтерфейсів і сумісності з необхідними модулями. рН-модуль забезпечує перетворення сигналу рН-електрода в аналогову напругу, *LCD*-дисплей дозволяє відображати поточне значення рН, а *Bluetooth*-модуль *HM-10* забезпечує бездротове передавання результатів до мобільного додатку.

У процесі роботи було виконано макетну збірку пристрою, підключення компонентів відповідно до електричної схеми та первинну перевірку працездатності основних вузлів. Також було розроблено програмне забезпечення для *Arduino*, яке виконує зчитування аналогового сигналу, усереднення значень, розрахунок рН, відображення результату на дисплеї та передавання даних у форматі, придатному для оброблення у мобільному додатку. Проведена перевірка на модельних розчинах підтвердила здатність системи реагувати на зміну кислотності середовища та виконувати основні функції прототипу.

РОЗДІЛ 3

МОДЕЛЮВАННЯ КОРПУСУ

3.1 Формування вимог до корпусу пристрою

Корпус пристрою для реєстрації рН слини призначений для розміщення електронних компонентів, їх механічного захисту та забезпечення зручності користування прототипом. Оскільки початкова схема була зібрана на безпайковій макетній платі, така конструкція не є достатньо надійною для перенесення та демонстрації.

Під час формування вимог до корпусу враховувалися габарити основних компонентів: *Arduino Uno*, рН-модуля, *Bluetooth*-модуля *HM-10*, *LCD*-дисплея, елементів живлення та з'єднувальних проводів. Корпус має забезпечувати достатній внутрішній простір для розміщення цих елементів, а також доступ до дисплея, роз'ємів живлення та підключення рН-електрода [17–21].

Важливою вимогою є наявність вікна для *LCD*-дисплея, оскільки користувач повинен мати можливість бачити поточне значення рН без відкривання корпусу. Також необхідно передбачити отвір для виведення рН-електрода, який контактує зі зразком слини, тоді як електронні компоненти залишаються всередині корпусу. Це зменшує ризик механічного пошкодження елементів і випадкового потрапляння рідини на електроніку.

Корпус повинен складатися з основи та знімної кришки, що дає змогу виконувати складання, перевірку з'єднань і обслуговування пристрою. Оскільки корпус виготовляється методом *3D*-друку, його конструкція має бути придатною до друку, мати достатню жорсткість, прийнятну товщину стінок і не містити надмірно складних геометричних елементів.

Підсумовані основні вимоги до корпусу пристрою описаний у таблиці 3.1.

Таблиця 3.1 – Основні вимоги до корпусу пристрою

Група вимог	Вимога
Функціональні	Розміщення <i>Arduino Uno</i> , рН-модуля, <i>HM-10</i> , <i>LCD</i> -дисплея, живлення та проводів
Конструктивні	Наявність основи, кришки, вікна для дисплея та отворів для роз'ємів
Експлуатаційні	Зручність перенесення, доступ до живлення та можливість обслуговування
Захисні	Механічний захист компонентів і відокремлення електроніки від зони контакту з рідиною
Технологічні	Придатність до <i>FDM 3D</i> -друку, достатня жорсткість і товщина стінок

Отже, корпус має забезпечити компактне розміщення апаратної частини, захист електронних компонентів і зручність використання пристрою. Сформовані вимоги стали основою для подальшого моделювання корпусу та підготовки його деталей до *3D*-друку.

3.2 Моделювання корпусу пристрою

3D-модель корпусу було розроблено у середовищі *SolidWorks*. Перед початком моделювання було визначено орієнтовне розташування компонентів усередині корпусу, щоб уникнути надмірного натягу проводів і забезпечити доступ до рН-електрода, дисплея та живлення. Такий підхід дозволив перейти від відкритої макетної схеми до конструктивно завершеного прототипу пристрою.

Конструкція корпусу була поділена на кілька основних елементів: нижню частину корпусу, кришку та внутрішню перегородку. Нижня частина призначена для розміщення основних електронних компонентів (рис. 3.1). Кришка забезпечує закриття корпусу та захист внутрішніх елементів. Перегородка використовується для більш раціонального розділення внутрішнього простору та зменшення зміщення компонентів під час перенесення пристрою.



Рисунок 3.1 – 3D-моделі деталей корпусу в програмі *SolidWorks*

У моделі корпусу було передбачено вікно для *LCD*-дисплея, щоб поточне значення рН можна було зчитувати без відкривання пристрою. Також було враховано необхідність виведення рН-електрода назовні, доступу до роз'ємів живлення та можливості технічного обслуговування електронної частини. Розміщення отворів і вирізів виконувалося відповідно до функціонального призначення окремих модулів. Під час моделювання враховувалися технологічні особливості подальшого 3D-друку. Корпус мав бути достатньо жорстким, але не надмірно масивним. Геометрія деталей була спроектована так, щоб їх можна було виготовити методом *FDM*-друку без значного ускладнення процесу. Для цього уникалися надто тонкі стінки, складні нависання та елементи, які могли б легко деформуватися під час друку. Після завершення моделювання було отримано набір деталей, придатних для подальшої підготовки до друку. Креслення змодельованих деталей представлені у додатку Б.

3.3 Підготовка моделі до 3D-друку та виготовлення корпусу

Після завершення моделювання корпусу окремі деталі були підготовлені до виготовлення методом 3D-друку. Для цього створені у середовищі *SolidWorks*

моделі було експортовано у формат *STL*, який є одним із найпоширеніших форматів для передавання геометрії тривимірних об'єктів у програми-слайсери. Такий формат містить інформацію про форму поверхні моделі та дозволяє підготувати її до пошарового виготовлення на 3D-принтері.

Підготовка моделей до друку виконувалася у середовищі *UltiMaker Cura* (рис. 3.2). На цьому етапі було виконано імпорт *STL*-файлів, розміщення деталей на робочій платформі, вибір орієнтації моделі та налаштування основних параметрів друку. Правильна орієнтація деталей має важливе значення, оскільки вона впливає на якість поверхні, міцність готового виробу, кількість підтримок і тривалість друку. Для корпусних деталей обиралися така орієнтація, яка забезпечувала стійке розташування на платформі та мінімізувала кількість нависаючих елементів.

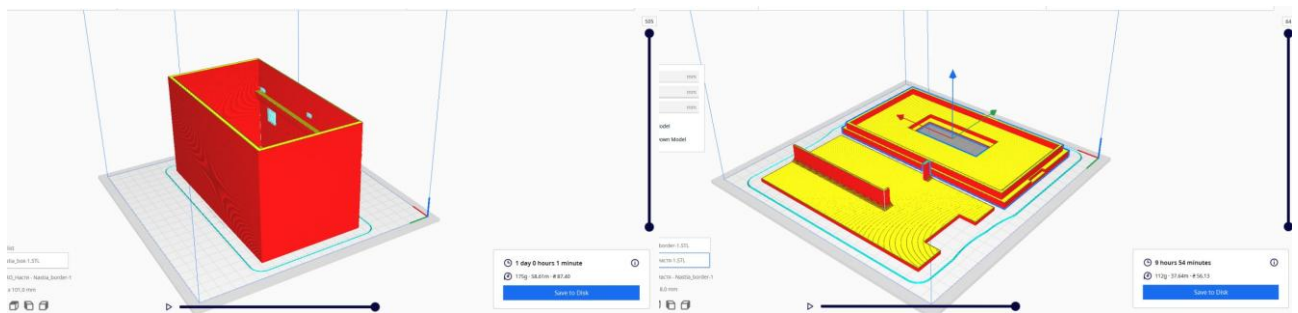


Рисунок 3.2 – Підготовка моделей деталей корпусу до друку в програмі *UltiMaker Cura*

Оскільки корпус складається з кількох деталей, кожен з них було підготовлено до друку окремо. Основна частина корпусу розміщувалася так, щоб забезпечити рівномірне формування стінок і внутрішнього простору. Кришка орієнтувалася пласкою стороною до робочої платформи, що дозволяло отримати рівну поверхню та зменшити потребу в додаткових підтримках. Перегородка також розміщувалася з урахуванням її геометрії та необхідності збереження точності розмірів після друку.

Під час підготовки до друку враховувалися такі параметри, як товщина шару, відсоток заповнення, товщина стінок, швидкість друку та необхідність використання підтримок. Для корпусу прототипу важливо було забезпечити

достатню механічну міцність, оскільки деталі мають утримувати електронні компоненти та витримувати навантаження під час складання й перенесення пристрою. Водночас надмірне заповнення не було доцільним, оскільки воно збільшує тривалість друку та витрату матеріалу без суттєвої потреби для макетного корпусу.

Після налаштування параметрів у слайсері було згенеровано файл *G-code*, який містить послідовність команд для 3D-принтера. Виготовлення корпусу виконувалося методом *FDM-друку*, який базується на пошаровому нанесенні розплавленого термопластичного матеріалу.

Після завершення друку було виконано первинний контроль якості виготовлених деталей. Перевірялася цілісність корпусу, відповідність отворів і вирізів, якість поверхні, можливість встановлення кришки та сумісність деталей між собою. Особливу увагу було приділено вікну для *LCD*-дисплея, отвору для підключення рН-електрода та місцям, через які має здійснюватися доступ до живлення або *USB*-роз'єму. Фото виготовлених деталей корпусу зображено на рисунку 3.3.

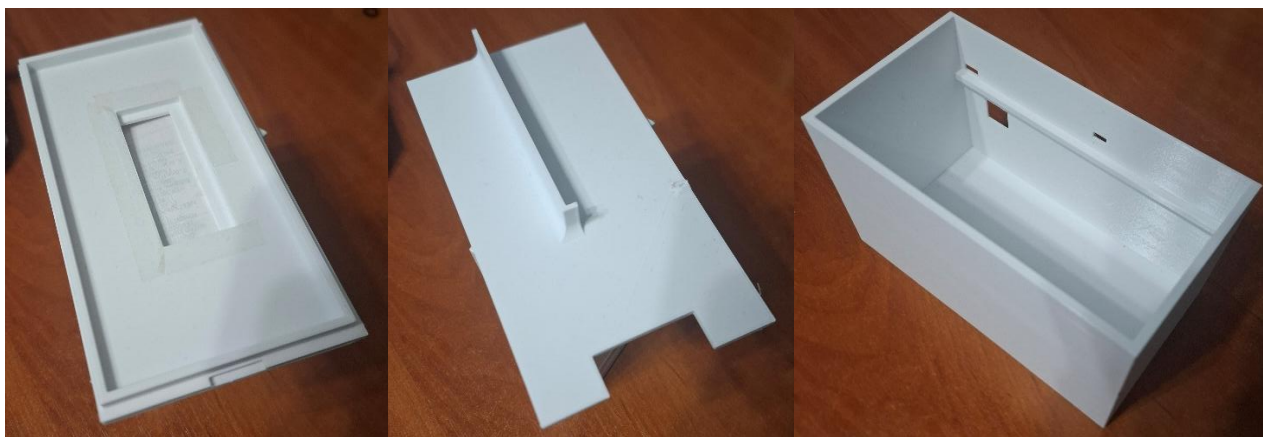


Рисунок 3.3 – Виготовлені деталі корпусу пристрою.

3.4 Збірка корпусу та розміщення компонентів пристрою

Після виготовлення деталей корпусу було виконано збірку пристрою та розміщення основних електронних компонентів усередині конструкції (рис. 3.4).

Метою цього етапу було перевірити відповідність надрукованого корпусу реальним габаритам апаратної частини, забезпечити зручне компонування елементів і переконатися, що корпус не перешкоджає роботі пристрою, підключенню рН-електрода, живленню та візуальному зчитуванню результатів.



Рисунок 3.4 – а – перший, б – другий, в – третій крок виконання збірки пристрою і корпусу

Висновок до розділу 3

У третьому розділі було виконано моделювання корпусу пристрою для реєстрації рН слини. На початковому етапі сформовано вимоги до корпусу, які враховують необхідність розміщення *Arduino Uno*, рН-модуля, *Bluetooth*-модуля *HM-10*, *LCD*-дисплея, елементів живлення та з'єднувальних проводів. Також

було визначено, що корпус має забезпечувати механічний захист компонентів, зручний доступ до дисплея, роз'ємів живлення та підключення рН-електрода.

3D-модель корпусу було розроблено у середовищі *SolidWorks*. Конструкція передбачає основну частину корпусу, кришку та внутрішню перегородку, що дозволяє раціонально розмістити електронні компоненти та зменшити ризик їх зміщення під час експлуатації. У моделі було передбачено вікно для *LCD*-дисплея, отвори для виведення рН-електрода та доступу до необхідних роз'ємів.

Після завершення моделювання деталі корпусу були підготовлені до виготовлення методом *FDM 3D*-друку в середовищі *UltiMaker Cura*. Було виконано експорт моделей у формат *STL*, налаштування параметрів друку, виготовлення деталей і первинну перевірку їхньої якості. У результаті було отримано корпус, придатний для розміщення апаратної частини пристрою, захисту електронних компонентів і демонстрації прототипу в більш завершеному конструктивному вигляді.

РОЗДІЛ 4

НАПИСАННЯ МОБІЛЬНОГО ДОДАТКУ

4.1 Функціональні вимоги до мобільного додатку

Мобільний додаток є програмною частиною розроблюваної системи та призначений для приймання, фіксації, збереження й аналізу результатів вимірювання рН слини. Якщо апаратна частина пристрою забезпечує реєстрацію сигналу з рН-сенсора та передавання значення через Bluetooth-модуль, то мобільний додаток виконує функцію користувацького інтерфейсу і засобу подальшої роботи з отриманими даними.

Основна ідея додатку полягає в тому, щоб користувач міг не лише побачити поточне значення рН, а й зберегти його з датою та часом вимірювання, переглянути історію результатів, побудувати графік зміни рН і отримати базові статистичні показники. Такий підхід дозволяє перейти від разового вимірювання до динамічного моніторингу кислотно-лужного стану ротової порожнини.

Додаток було передбачено для роботи зі значеннями рН, які надходять від *Arduino* через *Bluetooth*-модуль *HM-10* у текстовому форматі. Оскільки *HM-10* працює за технологією *Bluetooth Low Energy*, обмін даними в *Android*-додатку реалізується з урахуванням особливостей *BLE*-з'єднання та *GATT*-профілю [25]. У межах роботи пристрою мікроконтролер формує повідомлення у вигляді *РН:значення*, наприклад *РН:7.12*. Мобільний додаток повинен приймати цей рядок, виділяти з нього числову частину, перевіряти коректність даних і відображати результат користувачу.

Функціональні вимоги до мобільного додатку сформовано з урахуванням загальної логіки роботи системи. Додаток має забезпечувати пошук і підключення до *Bluetooth*-пристрою, приймання даних, запуск або фіксацію вимірювання, збереження результатів, перегляд історії, побудову графіка та розрахунок статистичних параметрів. Для пристроїв на *Android* важливо

враховувати дозвільну модель *Bluetooth*-з'єднань: для *Android* 12 і новіших версій застосовуються окремі дозволи на пошук і підключення до *Bluetooth*-пристроїв, зокрема *BLUETOOTH_SCAN* і *BLUETOOTH_CONNECT* [26]. Основні функціональні вимоги до мобільного додатку наведено в таблиці 4.1.

Таблиця 4.1 – Функціональні вимоги до мобільного додатку

Група вимог	Вимога	Призначення
Підключення до пристрою	Пошук або вибір <i>Bluetooth</i> -модуля <i>HM-10</i>	Забезпечення зв'язку між смартфоном і апаратною частиною
Підключення до пристрою	Підтримка <i>Bluetooth Low Energy</i>	Сумісність із модулем <i>HM-10</i>
Приймання даних	Отримання рядків формату <i>pH</i> :значення	Приймання результату вимірювання від <i>Arduino</i>
Оброблення даних	Виділення числового значення <i>pH</i> з отриманого рядка	Перетворення повідомлення у формат, придатний для аналізу
Оброблення даних	Перевірка коректності значення	Запобігання збереженню помилкових або неповних даних
Відображення	Показ поточного значення <i>pH</i> на головному екрані	Швидке інформування користувача про результат вимірювання
Збереження	Запис значення <i>pH</i> разом із датою та часом	Формування історії вимірювань
Історія	Перегляд попередніх вимірювань	Аналіз результатів за певний період
Візуалізація	Побудова графіка зміни <i>pH</i> у часі	Оцінювання динаміки кислотно-лужного стану
Статистика	Розрахунок середнього, мінімального та максимального <i>pH</i>	Кількісна оцінка результатів моніторингу
Експорт	Можливість збереження або передавання результатів	Подальше використання даних у звіті або аналізі
Інтерфейс	Зрозуміла структура екранів і кнопок керування	Спрощення роботи користувача з додатком

Окрім функціональних вимог, для додатку важливими є вимоги до зручності використання. Інтерфейс має бути простим і логічним: користувач повинен мати можливість швидко підключитися до пристрою, отримати результат вимірювання, зберегти його та переглянути попередні дані. Надмірна кількість екранів або складна навігація недоцільні, оскільки додаток має виконувати прикладну функцію — фіксацію та аналіз *pH*.

З технічної точки зору додаток має підтримувати роботу з локальними даними. Для збереження невеликих структурованих наборів даних в *Android* можуть використовуватися *DataStore*, *SharedPreferences* або база даних *Room*. Офіційна документація *Android* рекомендує *DataStore* як сучасніше рішення для зберігання невеликих наборів даних, тоді як *Room* доцільний для структурованого локального збереження даних [27–29].

4.2 Архітектура мобільного додатку

Мобільний додаток для фіксації та аналізу результатів вимірювання рН слини було розроблено для операційної системи *Android*. Його основним призначенням є приймання даних від апаратного пристрою через *Bluetooth*, відображення поточного значення рН, збереження результатів вимірювання та подальше їх опрацювання у вигляді історії, графіка і статистичних показників.

Архітектура додатку побудована за модульним принципом. Це дозволяє логічно розділити основні функції програми: роботу з інтерфейсом, приймання *Bluetooth*-даних, оброблення отриманого значення, збереження результатів і візуалізацію вимірювань. Такий підхід відповідає загальним принципам побудови *Android*-додатків, оскільки розділення функціональних частин спрощує підтримку, налагодження та подальше розширення програми [30].

Загальна логіка роботи мобільного додатку полягає у послідовному виконанні кількох етапів. Спочатку користувач запускає додаток і виконує підключення до *Bluetooth*-модуля *HM-10*. Після встановлення з'єднання додаток отримує від *Arduino* текстове повідомлення у форматі РН:значення. Далі програма виділяє з цього рядка числову частину, перетворює її у формат числа з плаваючою комою та відображає поточне значення рН на екрані. Після фіксації результату значення зберігається разом із датою та часом вимірювання.

Основним екраном додатку є головний екран, на якому користувач бачить статус підключення, поточне значення рН та кнопки керування. Саме з цього

екрана виконується підключення до пристрою, запуск вимірювання, збереження результату і перехід до перегляду історії або графіка. Інтерфейс має бути максимально простим, оскільки додаток виконує прикладну функцію і не повинен ускладнювати процес вимірювання.

Для реалізації інтерфейсу було використано *XML*-розмітку, а основна логіка роботи додатку реалізована мовою *Kotlin* у файлі *MainActivity.kt*. *XML*-розмітка в *Android* використовується для опису структури екрана та розміщення елементів інтерфейсу, зокрема текстових полів, кнопок і контейнерів [31]. У *Kotlin*-кодi реалізується обробка натискань кнопок, приймання *Bluetooth*-даних, збереження результатів і оновлення інтерфейсу.

Основні програмні компоненти додатку наведено в таблиці 4.2.

Таблиця 4.2 – Основні компоненти архітектури мобільного додатку

Компонент додатку	Призначення
Головний екран	Відображення поточного рН, стану підключення та кнопок керування
<i>Bluetooth</i> -модуль програми	Пошук, підключення та приймання даних від <i>HM-10</i>
Модуль оброблення даних	Виділення числового значення рН з рядка РН:значення
Модуль збереження	Запис результату з датою та часом
Модуль історії	Відображення попередніх вимірювань
Модуль статистики	Розрахунок середнього, мінімального та максимального рН
Модуль візуалізації	Побудова графіка зміни рН у часі

Для збереження одного результату вимірювання у програмі використовується структура даних, яка містить значення рН, дату та час фіксації.

У спрощеному вигляді таку модель можна подати так:

```
data class PhMeasurement(
    val phValue: Float,
    val dateTime: String
)
```

Така структура дозволяє зберігати не лише саме значення рН, а й часову прив'язку вимірювання. Це є важливим для подальшого аналізу, оскільки оцінювання динаміки рН слини можливе лише за умови накопичення послідовності результатів у часі.

Отримані від *Arduino* дані надходять у вигляді текстового рядка. Для коректної роботи додатку необхідно перевірити, чи містить повідомлення префікс *PH:*, після чого виділити числове значення. Приклад оброблення такого рядка можна подати так:

```
fun parsePhValue(data: String): Float? {  
    return if (data.startsWith("PH:")) {  
        data.removePrefix("PH:")  
            .trim()  
            .toFloatOrNull()  
    } else {  
        null  
    }  
}
```

У цьому фрагменті коду функція перевіряє формат повідомлення, видаляє службовий префікс і намагається перетворити отриманий текст у числове значення. Якщо дані мають неправильний формат або не можуть бути перетворені у число, функція повертає *null*. Це запобігає збереженню некоректних результатів.

Після отримання коректного значення рН додаток оновлює інтерфейс користувача. Поточний результат виводиться на головному екрані, а після натискання відповідної кнопки може бути збережений у списку вимірювань. Надалі цей список використовується для формування історії, побудови графіка та розрахунку статистики.

Архітектура додатку передбачає розділення поточного вимірювання і збережених результатів. Поточне значення рН може змінюватися після кожного нового повідомлення від *Arduino*, однак у базу або локальне сховище потрапляють лише ті результати, які користувач фіксує як окремі вимірювання. Це зменшує ризик випадкового накопичення нестабільних або проміжних значень під час роботи сенсора.

Для подальшого аналізу збережені результати використовуються у двох основних формах: табличній та графічній. Табличне подання дає змогу переглянути послідовність вимірювань із датою та часом, а графік дозволяє оцінити загальну тенденцію зміни рН. Статистичний модуль розраховує середнє, мінімальне та максимальне значення рН за вибраний набір вимірювань.

4.3 Реалізація *Bluetooth*-з'єднання з пристроєм

Для передавання результатів вимірювання від апаратної частини до мобільного додатку використано *Bluetooth*-з'єднання. У складі пристрою застосовано модуль *HM-10*, який працює за технологією *Bluetooth Low Energy* та приймає дані від *Arduino* через *UART*-інтерфейс. *Arduino* формує повідомлення з результатом вимірювання у текстовому форматі, після чого передає його на *HM-10*, а модуль надсилає ці дані до мобільного додатку.

У мобільному додатку *Bluetooth*-з'єднання реалізовано засобами *Android Studio* мовою *Kotlin*. Оскільки *HM-10* є *BLE*-модулем, підключення до нього виконується не як до класичного *Bluetooth*-пристрою типу *SPP*, а через механізм *BLE*-сканування, підключення до *GATT*-сервера та роботу з відповідною характеристикою. *Android* надає *API* для пошуку *BLE*-пристроїв, підключення до них, виявлення сервісів і обміну даними [25]. Для сучасних версій *Android* також потрібно враховувати дозволи на сканування та підключення до *Bluetooth*-пристроїв, зокрема *BLUETOOTH_SCAN* і *BLUETOOTH_CONNECT* [26].

Загальна логіка *Bluetooth*-з'єднання в додатку включає такі етапи:

1. Перевірка наявності *Bluetooth*-модуля у смартфоні.
2. Запит необхідних дозволів.
3. Запуск *BLE*-сканування.
4. Пошук модуля *HM-10*.
5. Підключення до знайденого пристрою.
6. Виявлення *GATT*-сервісів.
7. Підписка на отримання даних.
8. Приймання рядка формату *RH:значення*.
9. Оброблення та відображення отриманого значення *pH*.

Для роботи з *Bluetooth* у файлі *AndroidManifest.xml* необхідно вказати відповідні дозволи. Їхній набір залежить від версії *Android*, але для новіших

версій системи важливими є дозволи на сканування та підключення до *Bluetooth*-пристроїв [26].

Фрагмент опису дозволів у *AndroidManifest.xml*:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Дозвіл *ACCESS_FINE_LOCATION* може бути потрібний для *BLE*-сканування на старіших версіях *Android*, тоді як для *Android* 12 і новіших версій застосовуються окремі *Bluetooth*-дозволи. Через це в додатку передбачено перевірку дозволів перед початком пошуку пристрою [26].

Після надання дозволів виконується пошук *BLE*-пристроїв. У процесі сканування додаток аналізує знайдені пристрої та відбирає той, що відповідає *Bluetooth*-модулю апаратної частини. У спрощеному вигляді запуск сканування можна подати так:

```
private val bluetoothAdapter: BluetoothAdapter? =
    BluetoothAdapter.getDefaultAdapter()

private val bluetoothLeScanner = bluetoothAdapter?.bluetoothLeScanner

private fun startBleScan() {
    bluetoothLeScanner?.startScan(scanCallback)
}
```

Оброблення знайденого пристрою виконується у *callback*-функції. Якщо знайдений пристрій відповідає *HM-10*, додаток зупиняє сканування та виконує підключення.

```
private val scanCallback = object : ScanCallback() {
    override fun onScanResult(callbackType: Int, result: ScanResult) {
        val device = result.device
        val deviceName = device.name ?: ""

        if (deviceName.contains("HM", ignoreCase = true)) {
            bluetoothLeScanner?.stopScan(this)
            connectToDevice(device)
        }
    }
}
```

Підключення до *BLE*-пристрою виконується через метод *connectGatt()*, який використовується для встановлення *GATT*-з'єднання з *BLE*-пристроєм [25].

Після успішного підключення додаток запускає пошук сервісів, доступних на модулі.

```
private fun connectToDevice(device: BluetoothDevice) {
    bluetoothGatt = device.connectGatt(
        this,
        false,
        gattCallback
    )
}
```

Основна обробка стану з'єднання реалізується через *BluetoothGattCallback*, що дозволяє відстежувати зміну стану підключення, виявлення сервісів і надходження даних від *BLE*-пристрою [25]. У ньому додаток визначає, чи було встановлено з'єднання, чи пристрій від'єднався, а також коли завершено пошук сервісів.

```
private val gattCallback = object : BluetoothGattCallback() {

    override fun onConnectionStateChange(
        gatt: BluetoothGatt,
        status: Int,
        newState: Int
    ) {
        if (newState == BluetoothProfile.STATE_CONNECTED) {
            runOnUiThread {
                statusTextView.text = "Статус: пристрій підключено"
            }
            gatt.discoverServices()
        } else if (newState == BluetoothProfile.STATE_DISCONNECTED) {
            runOnUiThread {
                statusTextView.text = "Статус: пристрій не підключено"
            }
        }
    }

    override fun onServicesDiscovered(
        gatt: BluetoothGatt,
        status: Int
    ) {
        if (status == BluetoothGatt.GATT_SUCCESS) {
            enablePhDataReceiving(gatt)
        }
    }
}
```

Після виявлення сервісів додаток підписується на отримання даних із характеристики, через яку *HM-10* передає повідомлення від *Arduino*. У межах розробленої системи очікується, що *Arduino* надсилає рядок у форматі *RH:значення*. Наприклад: *RH:7.12*

Після отримання повідомлення додаток декодує байтовий масив у текстовий рядок і передає його на подальше оброблення.

```

override fun onCharacteristicChanged(
    gatt: BluetoothGatt,
    characteristic: BluetoothGattCharacteristic
) {
    val receivedData = characteristic.value
        .toString(Charsets.UTF_8)
        .trim()

    val pHValue = parsePhValue(receivedData)

    if (pHValue != null) {
        runOnUiThread {
            currentPhTextView.text = "pH: %.2f".format(pHValue)
        }
    }
}

```

Для виділення числового значення з отриманого рядка використовується окрема функція. Вона перевіряє, чи починається повідомлення з префікса *PH:*, після чого перетворює числову частину у тип *Float*.

```

private fun parsePhValue(data: String): Float? {
    return if (data.startsWith("PH:")) {
        data.removePrefix("PH:")
            .trim()
            .toFloatOrNull()
    } else {
        null
    }
}

```

Такий підхід дозволяє уникнути помилок під час оброблення службових повідомлень або неповних рядків. Якщо отримані дані не відповідають очікуваному формату, вони не використовуються для оновлення інтерфейсу та не зберігаються як результат вимірювання.

Для керування вимірюванням додаток може надсилати команди до *Arduino*. Наприклад, команда *READ* використовується для одиничного вимірювання, *START* — для запуску передавання даних, а *STOP* — для його зупинки. Передавання команди виконується через *BLE*-характеристику, доступну для запису.

```

private fun sendCommand(command: String) {
    val data = command.toByteArray(Charsets.UTF_8)
    writableCharacteristic?.value = data
    bluetoothGatt?.writeCharacteristic(writableCharacteristic)
}

```

Основні повідомлення, що використовуються для обміну між додатком і апаратною частиною, наведено в таблиці 4.3.

Таблиця 4.3 – Формати повідомлень *Bluetooth*-обміну

Напрямок обміну	Повідомлення	Призначення
Додаток → <i>Arduino</i>	<i>PING</i>	Перевірка зв'язку
Додаток → <i>Arduino</i>	<i>READ</i>	Запуск одиничного вимірювання
Додаток → <i>Arduino</i>	<i>START</i>	Запуск періодичного передавання
Додаток → <i>Arduino</i>	<i>STOP</i>	Зупинка передавання
Додаток → <i>Arduino</i>	<i>READY</i>	Пристрій готовий до роботи
Додаток → <i>Arduino</i>	<i>PH:7.12</i>	Передавання результату вимірювання
Додаток → <i>Arduino</i>	<i>UNKNOWN_COMMAND</i>	Отримано невідому команду

Після встановлення *Bluetooth*-з'єднання користувач бачить зміну статусу на головному екрані додатку. Якщо пристрій не підключено, додаток відображає відповідне повідомлення. Після успішного підключення з'являється можливість отримувати значення рН, запускати вимірювання та зберігати результати. Така структура взаємодії робить роботу з пристроєм зрозумілою та контрольованою.

Під час тестування було важливо врахувати, що *HM-10* працює як *BLE*-пристрій, а не як класичний *Bluetooth*-модуль з послідовним профілем *SPP*. Тому для коректної роботи мобільного додатку було реалізовано саме *BLE*-сканування та підключення через *GATT*. Це дозволило забезпечити приймання даних від *Arduino* та подальше використання отриманих значень у додатку.

4.4 Збереження результатів вимірювання

Збереження результатів вимірювання є однією з основних функцій мобільного додатку, оскільки саме воно забезпечує перехід від разового визначення рН до моніторингу показника в динаміці. Поточне значення рН, отримане через *Bluetooth*-з'єднання від апаратної частини пристрою, саме по собі є інформативним лише в момент вимірювання. Для подальшого аналізу необхідно фіксувати результат разом із датою та часом, щоб сформувати послідовність вимірювань за певний період.

У розробленому додатку збереження результату виконується після отримання коректного значення рН від *Arduino*. Дані надходять у форматі *PH:значення*, після чого додаток виділяє числову частину повідомлення,

перевіряє її коректність і відображає на головному екрані. Такий формат було обрано для спрощення передавання та подальшого оброблення даних, що надходять від *Arduino* через *Bluetooth*-модуль [20]. Після натискання кнопки збереження результат записується до локального сховища мобільного пристрою.

Для представлення одного результату вимірювання у програмі використовується структура даних, яка містить значення рН та часову мітку. У спрощеному вигляді модель запису можна подати так:

```
data class PhMeasurement(
    val phValue: Float,
    val dateTime: String
)
```

Поле *phValue* містить числове значення рН, а поле *dateTime* — дату та час фіксації результату. Саме наявність часової мітки дозволяє надалі будувати графік зміни рН та аналізувати результати у хронологічному порядку.

Перед збереженням результату додаток перевіряє, чи було отримано коректне значення рН. Це необхідно для того, щоб у список вимірювань не потрапляли службові повідомлення, порожні рядки або нестабільні дані. Наприклад, якщо *Bluetooth*-модуль передає повідомлення *READY*, *STARTED* або *STOPPED*, такі рядки не повинні зберігатися як результати вимірювання.

Фрагмент функції збереження результату може мати такий вигляд:

```
private val measurements = mutableListOf<PhMeasurement>()

private fun saveCurrentMeasurement(currentPh: Float?) {
    if (currentPh == null) {
        Toast.makeText(this, "Немає значення для збереження",
            Toast.LENGTH_SHORT).show()
        return
    }

    val currentDateTime = SimpleDateFormat(
        "dd.MM.yyyy HH:mm:ss",
        Locale.getDefault()
    ).format(Date())

    val measurement = PhMeasurement(
        phValue = currentPh,
        dateTime = currentDateTime
    )

    measurements.add(measurement)
    saveMeasurementsToStorage()
    updateHistoryView()
}
```

У цьому фрагменті спочатку перевіряється наявність поточного значення рН. Якщо значення відсутнє, користувач отримує повідомлення про неможливість збереження. Якщо значення наявне, додаток формує поточну дату і час, створює об'єкт *PhMeasurement*, додає його до списку результатів і оновлює історію вимірювань.

Для локального збереження результатів у межах прототипу може використовуватися механізм *SharedPreferences*, який в *Android* застосовується для збереження невеликих наборів даних у вигляді пар “ключ-значення” [28]. Він є достатнім для збереження невеликого набору даних, оскільки результати вимірювання мають просту структуру: значення рН та дата/час. Дані можуть зберігатися у вигляді текстового рядка, наприклад у форматі *JSON*.

Приклад збереження списку результатів:

```
private fun saveMeasurementsToStorage() {
    val json = Gson().toJson(measurements)

    getSharedPreferences("ph_storage", MODE_PRIVATE)
        .edit()
        .putString("measurements", json)
        .apply()
}
```

Під час повторного запуску додатку збережені результати завантажуються з локального сховища. Це дозволяє користувачу переглядати історію вимірювань навіть після закриття програми або перезавантаження смартфона.

```
private fun loadMeasurementsFromStorage() {
    val json = getSharedPreferences("ph_storage", MODE_PRIVATE)
        .getString("measurements", null)

    if (json != null) {
        val type = object : TypeToken<MutableList<PhMeasurement>>() {}.type
        val savedMeasurements: MutableList<PhMeasurement> =
            Gson().fromJson(json, type)

        measurements.clear()
        measurements.addAll(savedMeasurements)
    }
}
```

Після завантаження даних додаток оновлює елементи інтерфейсу: список історії вимірювань, статистичні показники та графік. Такий підхід забезпечує цілісність роботи додатку, оскільки всі модулі використовують один і той самий список збережених результатів.

У додатку важливо відокремити поточне значення рН від збережених результатів. Поточне значення може змінюватися при кожному новому повідомленні від пристрою, однак до історії потрапляють лише ті вимірювання, які користувач спеціально зафіксував. Це зменшує ризик накопичення випадкових або проміжних значень, які можуть виникати під час стабілізації рН-сенсора. Збережені результати можуть відображатися у вигляді списку. Кожен запис містить дату, час та значення рН. Такий формат є зручним для перегляду послідовності вимірювань і подальшого перенесення результатів у таблиці або звіт. Для підвищення практичної цінності додатку передбачено можливість подальшого використання збережених даних для побудови графіка, розрахунку середнього значення, визначення мінімального та максимального рН.

Зображення інтерфейсу і роботи додатку зображено на рисунку 4.1.

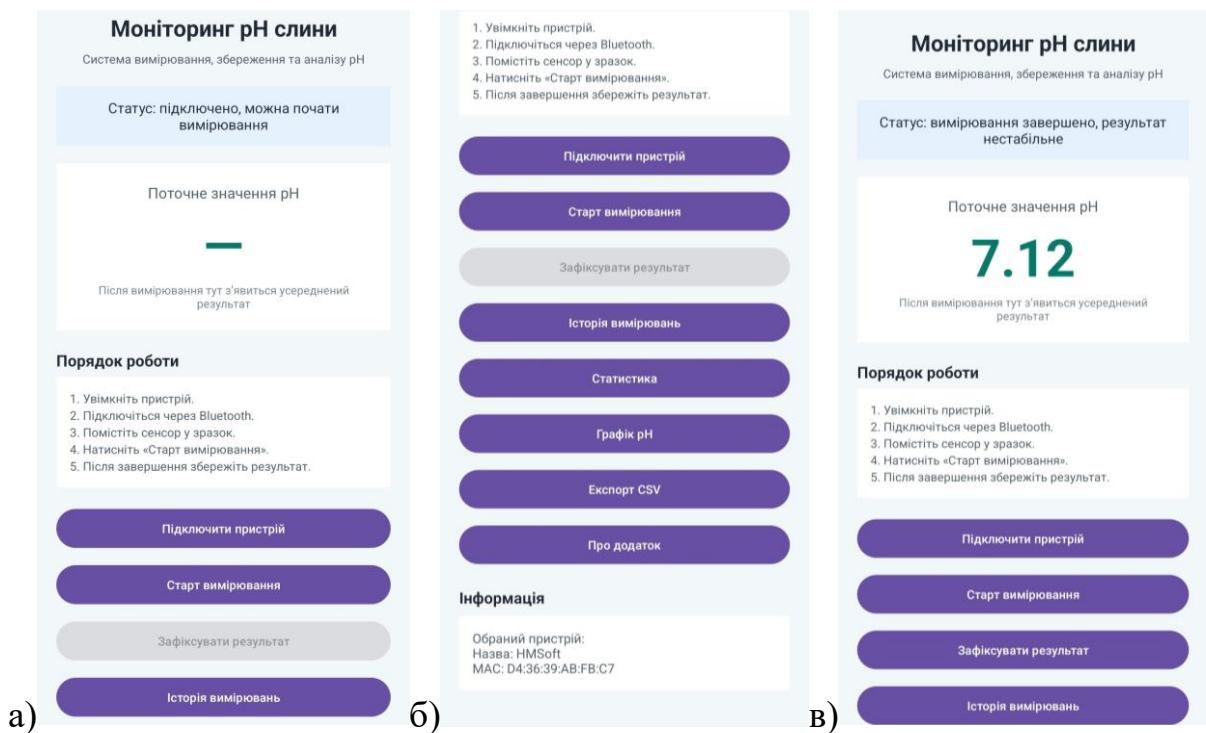


Рисунок 4.1 – а) головний екран мобільного додатку; б) відображення інформації про обраний *Bluetooth LE*-модуль; в) відображення результату поточного вимірювання рН слини

4.5 Візуалізація та статистичне оброблення даних

Після збереження результатів вимірювання мобільний додаток має забезпечити їх подальше подання у зручній для користувача формі. Окреме значення рН дає інформацію лише про стан зразка в момент вимірювання, однак для оцінювання змін кислотно-лужного стану слини більш важливим є аналіз послідовності результатів. Саме тому в додатку передбачено відображення історії вимірювань, побудову графіка зміни рН та розрахунок базових статистичних показників.

Візуалізація даних виконує дві основні функції. По-перше, вона дозволяє користувачу швидко переглянути попередні результати вимірювань у хронологічному порядку. По-друге, графічне подання дає змогу візуально оцінити характер зміни рН: стабільність показника, наявність підвищення або зниження, а також можливі різкі відхилення окремих значень.

У межах розробленого додатку результати вимірювання можуть подаватися у трьох формах:

1. У вигляді поточного значення рН на головному екрані.
2. У вигляді списку збережених вимірювань із датою та часом.
3. У вигляді графіка та статистичних показників.

Поточне значення рН оновлюється після отримання нового повідомлення від *Arduino* через *Bluetooth*. Це значення відображається на головному екрані додатку та використовується для подальшого збереження. Після натискання кнопки збереження результат додається до історії вимірювань і надалі враховується під час побудови графіка та розрахунку статистики.

Історія вимірювань формується на основі списку збережених записів типу *PhMeasurement*, кожен із яких містить числове значення рН і дату з часом фіксації. Для відображення історії кожен запис перетворюється у текстовий рядок, зручний для перегляду користувачем.

Приклад формування текстового списку результатів:

```
private fun updateHistoryView() {
    val historyText = measurements.joinToString(separator = "\n") { measurement
->
        "${measurement.dateTime} – pH %.2f".format(measurement.phValue)
    }

    historyTextView.text = if (historyText.isNotEmpty()) {
        historyText
    } else {
        "Історія вимірювань порожня"
    }
}
```

Такий спосіб відображення дозволяє користувачу бачити послідовність вимірювань у зрозумілому форматі. Наявність дати та часу є важливою, оскільки рН слини може змінюватися залежно від прийому їжі, напоїв, гігієнічних процедур, часу доби та індивідуальних особливостей слиновиділення [1–3, 5, 6, 8].

Для кількісного аналізу результатів у додатку розраховуються базові статистичні показники: середнє, мінімальне та максимальне значення рН. Середнє значення дозволяє оцінити загальний рівень рН за період спостереження. Мінімальне значення вказує на найбільш виражене зміщення у кислий бік, а максимальне — на найбільше значення рН серед збережених результатів.

Основні статистичні показники наведено в таблиці 4.5.

Таблиця 4.5 – Статистичні показники, що розраховуються у мобільному додатку

Показник	Призначення
Середнє значення рН	Оцінювання загального рівня рН за період спостереження
Мінімальне значення рН	Визначення найнижчого зафіксованого значення
Максимальне значення рН	Визначення найвищого зафіксованого значення
Кількість вимірювань	Оцінювання обсягу накопичених даних
Діапазон зміни рН	Визначення різниці між максимальним і мінімальним значенням

Фрагмент програмного коду для розрахунку статистичних показників може мати такий вигляд:

```
private fun updateStatistics() {
    if (measurements.isEmpty()) {
        avgTextView.text = "Середнє рН: –"
        minTextView.text = "Мінімальне рН: –"
    }
}
```

```

        maxTextView.text = "Максимальне рН: —"
        countTextView.text = "Кількість вимірювань: 0"
        return
    }

    val values = measurements.map { it.phValue }

    val average = values.average()
    val minimum = values.minOrNull()
    val maximum = values.maxOrNull()
    val count = values.size

    avgTextView.text = "Середнє рН: %.2f".format(average)
    minTextView.text = "Мінімальне рН: %.2f".format(minimum)
    maxTextView.text = "Максимальне рН: %.2f".format(maximum)
    countTextView.text = "Кількість вимірювань: $count"
}

```

У цьому фрагменті коду спочатку перевіряється, чи є у списку збережені вимірювання. Якщо список порожній, додаток виводить порожні значення статистики. Якщо результати наявні, програма формує список числових значень рН і розраховує середнє, мінімальне, максимальне значення та кількість вимірювань.

Для додаткового аналізу може бути розрахований діапазон зміни рН, тобто різниця між максимальним і мінімальним значенням. Цей показник дозволяє оцінити, наскільки сильно змінювався рН у межах обраного періоду.

```

val phRange = if (minimum != null && maximum != null) {
    maximum - minimum
} else {
    0f
}

rangeTextView.text = "Діапазон зміни рН: %.2f".format(phRange)

```

Графічна візуалізація результатів реалізується на основі збережених вимірювань. На горизонтальній осі графіка відкладається послідовність вимірювань або час їх проведення, а на вертикальній осі — значення рН. Такий графік дає змогу швидко оцінити зміну показника в динаміці та виявити загальну тенденцію.

У спрощеному вигляді підготовку даних для побудови графіка можна подати так:

```

private fun prepareGraphData(): List<Pair<Float, Float>> {
    return measurements.mapIndexed { index, measurement ->
        Pair(index.toFloat(), measurement.phValue)
    }
}

```

У цьому прикладі кожному вимірюванню присвоюється порядковий номер, який використовується як координата по осі X , а значення рН використовується як координата по осі Y . Такий підхід є зручним для прототипу, оскільки дозволяє побудувати графік навіть без складного форматування часової осі.

Для оновлення графіка після додавання нового результату використовується загальний принцип: після збереження вимірювання список даних оновлюється, повторно розраховуються статистичні показники та перебудовується графічне подання.

```
private fun refreshDataViews() {  
    updateHistoryView()  
    updateStatistics()  
    updateGraph()  
}
```

Тобто після кожного нового збереження додаток автоматично оновлює всі елементи, пов'язані з аналізом даних. Це дозволяє користувачу одразу бачити, як нове вимірювання вплинуло на середнє значення, мінімум, максимум і загальну форму графіка.

Важливою перевагою такого підходу є те, що користувач отримує не лише окремі числові значення, а й узагальнену картину зміни рН. Наприклад, якщо більшість вимірювань знаходиться приблизно на одному рівні, це свідчить про відносну стабільність показника. Якщо ж графік демонструє поступове зниження або часті різкі коливання, це може бути підставою для більш уважного спостереження та подальшого аналізу умов, за яких проводилися вимірювання.

4.6 Оцінювання динаміки рН слини за результатами моніторингу

Оцінювання динаміки рН слини є завершальним етапом роботи мобільного додатку з результатами вимірювань. Якщо окреме значення рН дозволяє оцінити кислотно-лужний стан слини лише в момент проведення вимірювання, то накопичення результатів протягом певного періоду дає змогу простежити зміну

показника в часі. Саме тому у розробленій системі важливим є не тільки вимірювання рН, а й подальше збереження та аналіз отриманих даних.

У межах роботи пристрою моніторинг рН слини передбачає регулярне виконання вимірювань із фіксацією дати та часу. Такий підхід дозволяє оцінити, чи залишається рН слини відносно стабільним, чи спостерігається тенденція до зниження або підвищення. Особливу увагу під час аналізу доцільно приділяти зниженню рН у кислий бік, оскільки тривале або часте зменшення кислотно-лужного показника може створювати умови, несприятливі для процесів ремінералізації емалі [1–3, 5, 6, 8].

Для коректного оцінювання динаміки рН важливо, щоб вимірювання проводилися за подібних умов. Значення рН слини може змінюватися після прийому їжі, вживання напоїв, чищення зубів, використання ополіскувачів, фізичного навантаження або залежно від часу доби. Тому для отримання більш порівнюваних результатів бажано виконувати вимірювання в однаковий час або фіксувати умови проведення вимірювання.

У спрощеному вигляді алгоритм оцінювання динаміки рН слини за результатами моніторингу включає такі етапи:

1. Накопичення результатів вимірювань рН із датою та часом.
2. Перегляд історії вимірювань.
3. Побудову графіка зміни рН.
4. Розрахунок середнього, мінімального та максимального значення.
5. Визначення діапазону коливань рН.
6. Оцінювання загальної тенденції зміни показника.
7. Виявлення періодів зниження рН у кислий бік.

Основні критерії, які можуть використовуватися для аналізу динаміки рН слини, наведено в таблиці 4.7.

Таблиця 4.7 – Критерії оцінювання динаміки рН слини за результатами моніторингу

Критерій	Що оцінює	Практичне значення
Середнє значення рН	Узагальнений рівень рН за період спостереження	Дозволяє оцінити загальний кисло-лужний стан
Мінімальне значення рН	Найнижчий зафіксований показник	Вказує на найбільш виражене зміщення у кислий бік
Максимальне значення рН	Найвище зафіксоване значення	Дозволяє визначити верхню межу коливань
Діапазон зміни рН	Різниця між максимальним і мінімальним значенням	Характеризує стабільність або мінливість показника
Кількість вимірювань	Обсяг накопичених даних	Впливає на достовірність аналізу
Напрямок зміни рН	Підвищення, зниження або стабільність	Дозволяє оцінити загальну тенденцію
Частота низьких значень	Кількість вимірювань у кислому діапазоні	Може вказувати на потребу в уважнішому спостереженні

У мобільному додатку оцінювання динаміки рН може виконуватися на основі вже збереженого списку вимірювань. Найпростішим способом визначення загальної тенденції є порівняння першого та останнього значення за обраний період. Якщо останнє значення помітно нижче за початкове, можна говорити про тенденцію до зниження рН. Якщо значення зростає — про тенденцію до підвищення. Якщо різниця незначна, показник можна вважати відносно стабільним.

Фрагмент програмної логіки для оцінювання напрямку зміни рН може мати такий вигляд:

```
private fun evaluatePhTrend(measurements: List<PhMeasurement>): String {
    if (measurements.size < 2) {
        return "Недостатньо даних для оцінювання динаміки"
    }

    val firstValue = measurements.first().phValue
    val lastValue = measurements.last().phValue
    val difference = lastValue - firstValue

    return when {
        difference > 0.2f -> "Спостерігається тенденція до підвищення рН"
        difference < -0.2f -> "Спостерігається тенденція до зниження рН"
        else -> "рН залишається відносно стабільним"
    }
}
```

У цьому прикладі порогове значення 0,2 рН використовується як умовна межа для відокремлення незначних коливань від помітнішої зміни показника.

Для прототипу такого підходу достатньо, однак у подальшому програму можна вдосконалити, додавши аналіз за днями, тижнями або обраними періодами.

Окрім порівняння першого й останнього значення, доцільним є аналіз кількості вимірювань, які потрапляють у нижчий діапазон рН. Наприклад, якщо протягом періоду спостереження значна частина результатів має знижені значення, це може свідчити про частіше зміщення середовища у кислий бік. Водночас такі результати не є самостійним діагностичним критерієм, а мають розглядатися лише як допоміжна інформація для оцінювання стану ротової порожнини.

Приклад підрахунку кількості знижених значень рН:

```
private fun countLowPhValues(measurements: List<PhMeasurement>): Int {
    return measurements.count { it.phValue < 6.0f }
}
```

У цьому фрагменті програма підраховує кількість вимірювань, у яких рН був нижчим за умовно заданий рівень 6,0. Таке значення може використовуватися для орієнтовного аналізу в межах додатку, але його не слід розглядати як універсальну клінічну межу. Остаточне трактування результатів залежить від умов вимірювання, індивідуальних особливостей користувача та методики дослідження.

Для кращого розуміння динаміки рН результати доцільно аналізувати не лише чисельно, а й графічно. Графік дозволяє побачити коливання показника протягом періоду моніторингу, виявити поодинокі різкі зміни або поступове зміщення значень. Наприклад, стабільний графік без значних коливань свідчить про відносно сталі умови кислотно-лужного балансу, тоді як часті зниження можуть бути підставою для більш детального спостереження.

У межах мобільного додатку результати аналізу можуть подаватися користувачу у вигляді короткого текстового висновку. Такий висновок формується автоматично на основі статистичних показників і напряму зміни рН.

Приклад формування короткого висновку:

```
private fun generateSummary(
    average: Double,
    min: Float,
    max: Float,
```

```

trend: String
): String {
    return """
        Середнє значення pH: %.2f
        Мінімальне значення pH: %.2f
        Максимальне значення pH: %.2f
        Висновок: $trend
    """
}

```

Основні варіанти інтерпретації результатів моніторингу наведено в таблиці 4.8.

Таблиця 4.8 – Орієнтовна інтерпретація результатів моніторингу pH слини

Характер результатів	Можлива інтерпретація
Значення pH залишаються близькими між собою	Показник є відносно стабільним
Спостерігається поступове зниження pH	Можливе зміщення кислотно-лужного стану у кислий бік
Спостерігається поступове підвищення pH	Можливе зміщення показника у нейтральний або слабколужний бік
Наявні поодинокі різкі зниження pH	Можливий вплив їжі, напоїв або умов вимірювання
Часто фіксуються знижені значення pH	Потребує уважнішого спостереження та стандартизації вимірювань
Даних мало або вимірювання нерегулярні	Динаміку оцінити складно

Важливо підкреслити, що розроблений додаток не встановлює діагноз і не замінює стоматологічне обстеження. Його призначення полягає у фіксації результатів вимірювання, візуалізації змін і наданні користувачу зручного інструменту для спостереження за pH слини. Отримані дані можуть бути використані як допоміжна інформація для оцінювання індивідуальної динаміки показника або для демонстрації роботи програмно-апаратного прототипу.

Зображення інтерфейсу, процесу фіксації результатів вимірювань, статистичного і графічного аналізу зображено на рисунку 4.2.

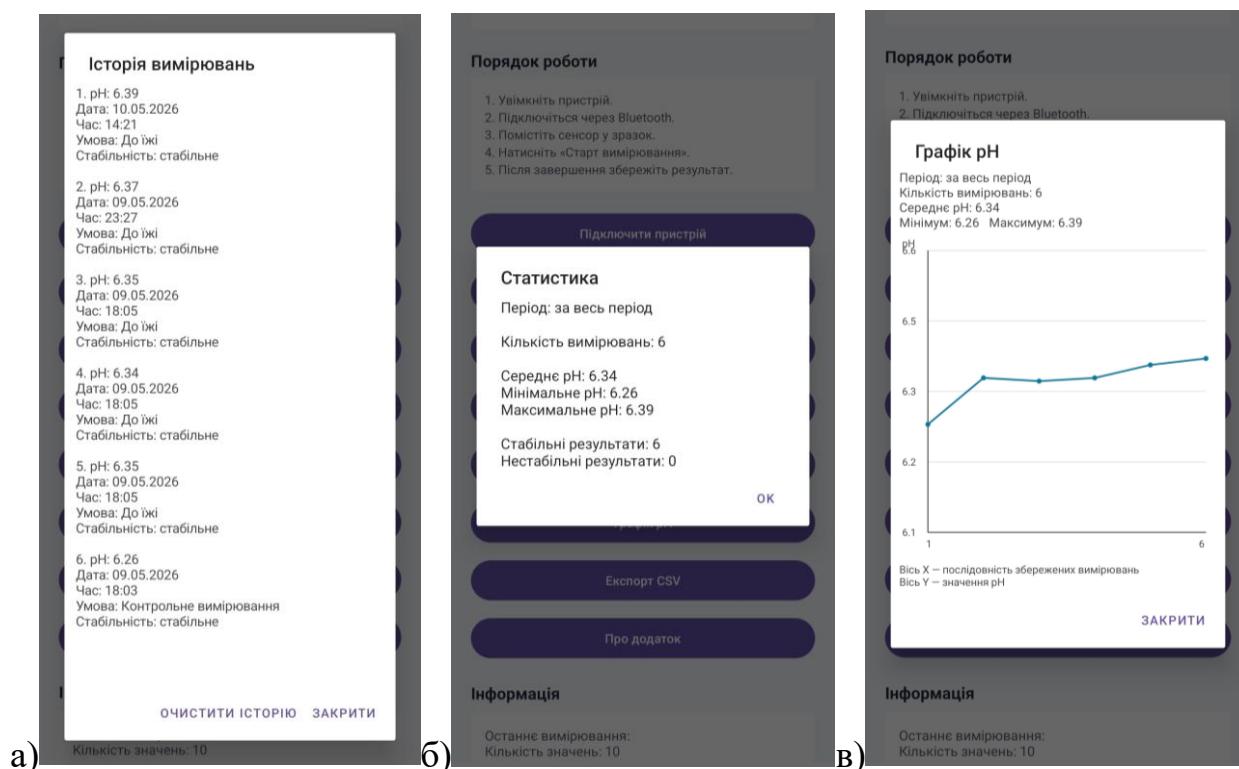


Рисунок 4.2 – а) Історія збережених вимірювань рН слини; б) вікно статистичного аналізу збережених результатів; в) графік зміни рН за послідовністю збережених вимірювань

Висновок до розділу 4

У четвертому розділі було розроблено мобільний додаток для фіксації та аналізу результатів вимірювання рН слини. Повний лістинг коду мобільного додатку наведено у додатку В. На основі функціональних вимог визначено, що додаток має забезпечувати підключення до пристрою через *Bluetooth Low Energy*, приймання даних від *Arduino*, відображення поточного значення рН, збереження результатів із датою та часом, перегляд історії вимірювань, побудову графіка та розрахунок статистичних показників.

Було описано архітектуру мобільного додатку, яка включає модулі інтерфейсу, *Bluetooth*-з'єднання, оброблення отриманих даних, локального збереження, історії вимірювань, статистики та графічної візуалізації. Основна логіка додатку реалізована мовою *Kotlin*, а структура інтерфейсу — за допомогою *XML*-розмітки. Для обміну з апаратною частиною використано

повідомлення у форматі *pH*:значення, що спрощує виділення числового результату та його подальше опрацювання.

У додатку реалізовано *Bluetooth*-з'єднання з модулем *HM-10*, збереження результатів у локальному сховищі, формування історії вимірювань, розрахунок середнього, мінімального та максимального значення *pH*, а також побудову графіка зміни показника. Передбачено оцінювання динаміки *pH* слини за результатами моніторингу, що дозволяє користувачу аналізувати не лише окремі вимірювання, а й загальну тенденцію зміни кислотно-лужного стану в часі. Розроблений додаток не виконує діагностичну функцію, однак є важливою складовою програмно-апаратної системи, оскільки забезпечує накопичення, візуалізацію та базове статистичне оброблення результатів вимірювання.

РОЗДІЛ 5

ОХОРОНА ПРАЦІ

Оскільки розробка пристрою системи спрямована на створення портативного пристрою для аналізу рН слини з подальшим передаванням результатів у мобільний додаток, у даному розділі розглянуто потенційні небезпеки, які можуть виникнути під час експлуатації цієї системи, та заходи щодо їх усунення. Особливістю об'єкта є поєднання електронних компонентів, вимірювального рН-електрода, низьковольтного живлення, бездротового передавання даних і контакту вимірювальної частини зі зразком біологічної рідини [32, 33].

5.1 Характеристики пристрою для аналізу рН слини

5.1.1 Характеристика компонентів пристрою

Пристрій спроектовано на основі *Arduino Uno R3*, рН-сенсорного модуля зі скляним електродом, *LCD*-дисплея 16×2 з *I2C*-інтерфейсом, *Bluetooth Low Energy* модуля *HM-10*, макетної плати, з'єднувальних провідників і джерела живлення 5 В постійного струму. *Arduino Uno* працює від 5 В, а у прототипі живлення переважно здійснюється через *USB* або *18650 Battery Shield*, що зменшує ризики, пов'язані з підключенням до мережі 220 В [18].

рН-сенсорний модуль перетворює сигнал скляного електрода в аналогову напругу, яка подається на вхід *A0* мікроконтролера. *Arduino* обробляє отриманий сигнал, розраховує значення рН та виводить результат на дисплей [17]. При цьому *BNC*-з'єднувач і плата рН-модуля повинні залишатися сухими, оскільки зволоження може погіршити точність вимірювання [16].

Передавання даних до мобільного додатку реалізовано за допомогою *Bluetooth 4.0 BLE* модуля *HM-10*, який працює через *UART*-інтерфейс [20].

Таблиця 5.1 – Характеристики компонентів системи

№	Компонент	Основні характеристики компонентів комплексу	Кількість
1	Мікроконтролер <i>Arduino Uno R3</i>	Клас виробу за ступенем захисту: умовно <i>IP20*</i> при відкритому макетному виконанні. Робоча напруга: 5 В через <i>USB</i> ; рекомендована напруга через зовнішній роз'єм: 7–12 В. Аналогові входи: <i>A0–A5</i> . Інтерфейси: <i>UART, I2C, SPI</i> . Функція в системі: обробка сигналу рН-сенсора, керування дисплеєм і передавання даних [18].	1
2	рН-сенсорний модуль зі скляним електродом	Клас виробу за ступенем захисту: <i>IP20*</i> для електронної плати. Живлення: 5 В постійного струму. Вихід: аналоговий сигнал <i>Po</i> до входу <i>A0</i> . Функція: вимірювання рН зразка слини або калібрувального розчину. Електронна частина модуля повинна залишатися сухою [16, 17].	1
3	<i>LCD</i> -дисплей 16×2 з <i>I2C</i> -модулем	Клас виробу за ступенем захисту: <i>IP20*</i> . Живлення: 5 В постійного струму. Інтерфейс: <i>I2C</i> . Функція: локальне відображення поточного значення рН та службових повідомлень.	1
4	<i>Bluetooth Low Energy</i> модуль <i>HM-10</i>	Клас виробу за ступенем захисту: <i>IP20*</i> . Тип зв'язку: <i>Bluetooth 4.0 BLE</i> . Інтерфейс з <i>Arduino</i> : <i>UART</i> . Типова швидкість обміну: 9600 бод. Функція: передавання результатів вимірювання до мобільного додатку [20].	1
5	З'єднувачі-перемички та макетна плата	Клас виробу за ступенем захисту: <i>IP20*</i> . Функція: електричне з'єднання компонентів прототипу. При пошкодженні ізоляції або неправильному підключенні можливе коротке замикання.	1 комплект
6	<i>USB</i> -кабель / <i>18650 Battery Shield</i> / ноутбук	Живлення пристрою напругою 5 В постійного струму. Одночасно забезпечує передавання даних або автономне живлення. Пошкодження кабелю чи роз'єму може спричинити нестабільну роботу або коротке замикання.	1
7	Корпус пристрою	Нестандартний компонент, який передбачається конструкцією. Функція: механічний захист плат, розділення сухої електронної зони та вологої вимірювальної зони, зменшення ризику випадкового дотику до контактів.	1

* Примітка щодо класу захисту *IP20*: більшість використаних електронних модулів у макетному виконанні не мають власного захисту від пилу та вологи. Згідно з класифікацією ступенів захисту оболонками електрообладнання *IP20* забезпечує захист від твердих об'єктів розміром понад 12,5 мм і не забезпечує захисту від води [34]. Тому для безпечної експлуатації проєктом передбачається розміщення електронних компонентів у корпусі з вищим ступенем захисту, наприклад *IP54* або *IP65* залежно від умов використання.

Класифікація компонентів за критеріями безпеки здійснюється з урахуванням вимог до електрообладнання для вимірювання, контролю та лабораторного використання [35].

5.1.2 Складові частини системи

У пристрої можна виділити три основні функціональні частини: систему збору сигналу, систему обробки та систему відображення й передавання даних. До системи збору сигналу належить рН-електрод із вимірювальним модулем. До системи обробки належить мікроконтролер *Arduino Uno R3*. До системи відображення та передавання належать *LCD*-дисплей, *BLE*-модуль *HM-10* і мобільний додаток. рН-електрод контактує зі зразком слини або калібрувальним розчином, після чого аналоговий сигнал із рН-модуля подається на вхід *A0* *Arduino Uno*. Мікроконтролер обробляє сигнал і передає результат на *LCD*-дисплей та *BLE*-модуль *HM-10*. Через *Bluetooth Low Energy* результат вимірювання надходить до мобільного додатку, де зберігається і може бути використаний для побудови графіків та статистики. Живлення системи здійснюється від джерела 5 В постійного струму.

5.1.3 Характер взаємодії об'єкту в системі «людина – об'єкт»

Робота системи частково автоматизована програмним забезпеченням, тому втручання користувача обмежується підготовкою зразка, зануренням рН-електрода, увімкненням пристрою, контролем показників на дисплеї та роботою з мобільним додатком. Найбільша увага приділяється безпечній роботі з біологічною рідиною, недопущенню потрапляння вологи на електронні компоненти та правильному від'єднанню пристрою після завершення вимірювань.

Таблиця 5.2 – Характер взаємодії об'єкту в системі «людина – об'єкт»

№	Найменування	Вид відображення інформації або керування	Кількість
1	Мікроконтролер <i>Arduino Uno R3</i>	Індикатор підключення на платі; виконання програми обробки сигналу; зв'язок із рН-модулем, дисплеєм і <i>BLE</i> -модулем.	1
2	рН-сенсорний модуль	Світлодіод живлення на платі; аналоговий сигнал, який не відображається користувачу безпосередньо, а обробляється <i>Arduino</i> .	1
3	рН-електрод	Безпосереднього індикатора не має; користувач контролює правильність занурення електрода у зразок або буфер.	1
4	<i>LCD</i> -дисплей 16×2	Відображення поточного значення рН, службових повідомлень, стану вимірювання або калібрування.	1
5	<i>BLE</i> -модуль <i>HM-10</i>	Індикатор підключення; передавання даних у мобільний додаток через <i>Bluetooth Low Energy</i> .	1
6	Мобільний додаток	Відображення статусу підключення, значення рН, журналу вимірювань, графіків і статистики.	1
7	<i>18650 Battery Shield</i> / <i>USB</i> -кабель	Фізичне підключення живлення; контроль справності кабелю та відсутності перегріву.	1

5.2 Оцінювання потенційних небезпек та заходи їх усунення

Беручи до уваги особливості використання системи, необхідно оцінити такі основні групи небезпек: біологічну небезпеку через контакт із слиною,

хімічну небезпеку при використанні буферних і дезінфекційних розчинів, небезпеку ураження електричним струмом та пожежну небезпеку. Фізичні фактори у вигляді шуму, вібрації, надлишкового тиску, інфразвуку, ультразвуку, лазерного чи іонізуючого випромінювання у конструкції пристрою не створюються. Електромагнітне випромінювання *BLE*-модуля є малопотужним і характерним для стандартних засобів бездротового зв'язку.

5.2.1 Біологічна небезпека

Оскільки вимірювальний електрод контактує зі слиною, у системі виникає біологічний фактор небезпеки. Слина є неінвазивним біозразком, але її склад залежить від індивідуальних і поведінкових факторів; під час досліджень слід враховувати можливе забруднення зразка, особливості збору, рН, швидкість слиновиділення та інші чинники [8, 36]. Ротова порожнина містить складну мікробну екосистему з бактеріями, грибами, археями та вірусами, тому використаний електрод і допоміжні ємності розглядаються як потенційно біологічно забруднені.

Таблиця 5.3 – Оцінка причин, джерел та наслідків біологічної небезпеки

№	Компонент / процес	Джерело небезпеки	Причини небезпеки	Можливі наслідки
1	Зразок слини	Біологічна рідина	Контакт зі зразком під час підготовки, вимірювання або утилізації ємності	Забруднення рук, поверхонь, електрода; потенційне перенесення мікроорганізмів
2	рН-електрод	Забруднена вимірювальна частина	Повторне використання без належного промивання та дезінфекції	Перехресне забруднення зразків, біологічне забруднення робочої зони
3	Одноразові ємності та серветки	Матеріали після контакту зі слиною	Неправильна утилізація або залишення на робочому місці	Поширення біологічного матеріалу на робочі поверхні
4	Робоча поверхня	Бризки або проливання слини	Необережне занурення електрода, нестійке розташування ємності	Забруднення робочого місця та електронної частини пристрою

Таблиця 5.4 – Реальні та нормативні значення факторів біологічної небезпеки

№	Фактор небезпеки	Реальні значення	Нормативні значення / безпечний стан
1	Контакт зі слиною	Наявний під час вимірювання	Виключення прямого контакту зі зразком без засобів індивідуального захисту
2	Біологічне забруднення рН-електрода	Наявне після кожного вимірювання	Електрод очищений, промитий і продезінфікований після контакту зі зразком
3	Забруднення робочої поверхні	Можливе при проливанні зразка	Відсутнє; поверхня очищена та оброблена після роботи
4	Повторне використання ємностей	Можливе при порушенні методики	Не допускається; застосовуються чисті або одноразові ємності

Таблиця 5.5 – Заходи із забезпечення охорони праці щодо біологічної небезпеки

№	Група номенклатурних заходів з ОП	Вид заходу	Критерій вибору
1	Технічні заходи	Конструктивне розділення вологої вимірювальної зони та сухої електронної частини пристрою.	Запобігання потраплянню слини на електронні компоненти та зменшення ризику короткого замикання.
2	Технічні заходи	Використання підставки або стійкої ємності для зразка.	Зменшення ймовірності проливання слини під час вимірювання.
3	Організаційні заходи	Використання одноразових рукавичок і чистих ємностей для кожного вимірювання.	Запобігання прямому контакту зі зразком та перехресному забрудненню.
4	Режимні заходи	Проведення вимірювання лише на чистій робочій поверхні з подальшою обробкою.	Підтримання санітарного стану робочого місця.
5	Експлуатаційні заходи	Промивання та дезінфекція рН-електрода після кожного контакту зі зразком.	Унеможливлення перенесення біологічного матеріалу між вимірюваннями.

5.2.2 Хімічна небезпека

Хімічна небезпека пов'язана не з електронною частиною пристрою, а з допоміжними речовинами, які використовуються під час калібрування, промивання та дезінфекції електрода. До таких речовин належать тестові розчини рН 4,01, рН 6,86 або рН 7,00, рН 9,18, дистильована вода, розчин для зберігання рН-електрода та спиртовмісні дезінфекційні засоби.

Таблиця 5.6 – Оцінка причин, джерел та наслідків хімічної небезпеки

№	Речовина / процес	Джерело небезпеки	Причини небезпеки	Можливі наслідки
1	Буферні розчини для калібрування	Рідини з різним рН	Потрапляння на шкіру, слизові оболонки або в очі при необережній роботі	Подразнення шкіри або очей, забруднення поверхні
2	Дезінфекційні засоби	Спиртовмісні або інші хімічні речовини	Надмірне використання, випаровування, робота поблизу джерел нагрівання	Подразнення шкіри, подразнення дихальних шляхів, пожежна небезпека
3	Розчин для зберігання електрода	Електроліт або спеціальний розчин	Проливання під час обслуговування електрода	Подразнення шкіри, пошкодження поверхонь або електроніки
4	Пошкодження скляного рН-електрода	Витік внутрішнього електроліту	Механічне пошкодження електрода	Контакт з електролітом, порізи уламками скла

Таблиця 5.7 – Реальні та нормативні значення факторів хімічної небезпеки

№	Фактор небезпеки	Реальні значення	Нормативні значення / безпечний стан
1	Контакт із буферними розчинами	Можливий при калібруванні	Відсутній прямий контакт зі шкірою та очима; робота виконується у рукавичках
2	Випаровування дезінфекційних засобів	Можливе при відкритих флаконах	Мінімальне; робота у провітрюваному приміщенні, флакони закриті
3	Проливання рідин	Можливе при роботі зі зразком і буферами	Відсутнє або негайно ліквідоване; робоча поверхня очищена

4	Пошкодження рН-електрода	Можливе при ударі або неправильному зберіганні	Відсутнє; електрод цілий, без тріщин і витікання електроліту
---	--------------------------	--	--

Таблиця 5.8 – Заходи із забезпечення охорони праці щодо хімічної небезпеки

№	Група номенклатурних заходів з ОП	Вид заходу	Критерій вибору
1	Технічні заходи	Використання малих об'ємів буферних і дезінфекційних розчинів.	Зменшення можливих наслідків проливання та контакту з речовинами.
2	Організаційні заходи	Зберігання буферів і дезінфектантів у підписаних закритих ємностях.	Запобігання помилковому використанню речовин.
3	Режимні заходи	Проведення роботи у провітрюваному приміщенні, подалі від відкритого полум'я.	Зменшення впливу парів та пожежного ризику спиртовмісних засобів.
4	Експлуатаційні заходи	Заборона використання тріснутого або пошкодженого рН-електрода.	Запобігання контакту з електролітом і травмуванню склом.

5.2.3 Небезпека ураження електричним струмом

Пристрій працює від низьковольтного джерела живлення 5 В постійного струму, тому рівень електричної небезпеки є значно нижчим, ніж у пристроїв, що безпосередньо живляться від мережі 220 В. Однак ризик ураження струмом або пошкодження обладнання може виникнути через відкриті провідники, неправильне підключення модулів, пошкодження *USB*-кабелю, використання несправного адаптера або потрапляння рідини на електронні компоненти. Для електрообладнання має бути забезпечений захист від прямого та непрямого дотику до струмопровідних частин [37].

Таблиця 5.9 – Оцінка причин, джерел та наслідків небезпек ураження електричним струмом

№	Назва компонентів комплексу	Джерело небезпеки	Причини небезпеки	Можливі наслідки
1	Мікроконтролер <i>Arduino Uno R3</i>	Постійний струм 5 В	Неправильне підключення живлення, пошкодження контактів, відсутність корпусу	Пошкодження плати, коротке замикання, нестабільна робота системи
2	pH-сенсорний модуль	Постійний струм 5 В та аналоговий вихід	Потрапляння вологи на плату, неправильне підключення <i>V+</i> , <i>GND</i> або <i>Р0</i>	Коротке замикання, спотворення результатів, відмова модуля
3	<i>LCD</i> -дисплей і <i>BLE</i> -модуль <i>HM-10</i>	Постійний струм, сигнальні лінії <i>I2C</i> та <i>UART</i>	Пошкодження провідників, помилкове підключення ліній живлення і сигналу	Перегрівання, вихід з ладу модулів
4	<i>USB</i> -кабель / <i>18650 Battery Shield</i> / ноутбук	Джерело живлення 5 В	Пошкодження кабелю, роз'єму або використання неякісного адаптера	Коротке замикання, перегрів роз'єму, відключення системи
5	З'єднувачі-перемички	Відкриті провідники та контакти	Механічне пошкодження ізоляції, слабе з'єднання, випадкове замикання	Нестабільна робота, коротке замикання, пошкодження компонентів

Таблиця 5.10 – Реальні та нормативні значення небезпек електричного характеру

№	Фактор небезпеки	Реальні значення	Нормативні значення / безпечний стан
1	Напруга живлення електронної частини	5 В постійного струму через <i>USB</i> або <i>18650 Battery Shield</i>	Живлення не повинно перевищувати допустимі значення для <i>Arduino</i> та модулів; використання стабілізованого джерела 5 В [18]
2	Відкриті плати та контакти	Наявні у макетному прототипі	Відсутність доступу користувача до струмопровідних частин; розміщення в корпусі [37]
3	Потрапляння рідини на електроніку	Можливе при порушенні процедури вимірювання	Відсутнє; електронна частина розміщена у сухій зоні
4	Пошкодження <i>USB</i> -кабелю або адаптера	Можливе при тривалій експлуатації	Відсутнє; використовується справний кабель і сертифіковане джерело живлення

Таблиця 5.11 – Заходи із забезпечення охорони праці щодо електробезпеки

№	Група номенклатурних заходів з ОП	Вид заходу	Критерій вибору
1	Технічні заходи	Розміщення <i>Arduino</i> , рН-модуля, <i>BLE</i> -модуля та дисплея у корпусі.	Унеможливлення прямого контакту користувача з відкритими контактами.
2	Технічні заходи	Розділення електронної частини та зони контакту рН-електрода зі зразком.	Запобігання потраплянню слини або буферів на електронні плати.
3	Організаційні заходи	Проведення інструктажу щодо правильного підключення та вимкнення пристрою.	Уникнення помилок користувача під час експлуатації.
4	Режимні заходи	Заборона роботи з пристроєм у вологому середовищі або мокрими руками.	Зниження ризику короткого замикання і пошкодження пристрою.
5	Експлуатаційні заходи	Візуальна перевірка ізоляції, кабелю, корпусу та роз'ємів перед кожним використанням.	Своєчасне виявлення несправностей.

5.2.4 Пожежна небезпека

Пожежна небезпека пристрою є невисокою через малу споживану потужність і низьковольтне живлення. Проте займання може виникнути у разі короткого замикання, перегріву елементів, використання несправного джерела живлення, пошкодження акумуляторного джерела або неправильного застосування спиртовмісних дезінфекційних засобів. Організаційні заходи пожежної безпеки повинні відповідати Правилам пожежної безпеки в Україні [38]. Технічний регламент низьковольтного електричного обладнання застосовується до обладнання у визначених діапазонах напруг, однак його загальні підходи до безпечності джерел живлення враховуються при виборі адаптерів та блоків живлення [39].

Таблиця 5.12 – Основні небезпеки, пов’язані з ризиком займання

№	Найменування компонентів комплексу	Джерело небезпеки	Причини небезпеки	Наслідки небезпеки
1	<i>Arduino Uno R3</i> та макетна плата	Коротке замикання, електричні іскри	Помилкове підключення, потрапляння вологи, пошкодження плати або провідників	Вихід з ладу компонентів, локальне нагрівання, можливе займання ізоляції
2	pH-модуль, <i>LCD, HM-10</i>	Перегрів електронних компонентів	Несправне живлення, замикання сигнальних ліній, неправильне підключення	Пошкодження модуля, перегрів, відмова системи
3	<i>USB</i> -кабель або мережевий адаптер	Перегрів роз’єму, коротке замикання	Використання пошкодженого кабелю або несертифікованого адаптера	Нагрівання, запах гару, займання кабелю
4	<i>18650 Battery Shield</i> або акумулятор ноутбука	Порушення цілісності акумулятора	Механічне пошкодження, перегрів, несправність контролера заряду	Займання, теплові опіки, пошкодження пристрою
5	Дезінфекційні засоби	Легкозаймисті пари	Використання спиртовмісних речовин поблизу нагрітих елементів або джерел іскріння	Пожежа або локальне займання

Таблиця 5.13 – Реальні та нормативні значення факторів пожежної та теплової небезпеки

№	Фактор небезпеки	Реальні значення	Нормативні значення / безпечний стан
1	Коротке замикання	Можливе при помилковому підключенні або потрапленні рідини	Відсутнє; контакти ізольовані, електроніка суха
2	Електричні іскри	Можливі при пошкодженні кабелю або роз’єму	Відсутні; використовується справний кабель і джерело живлення
3	Перегрів електронних компонентів	Можливий при несправності живлення або замиканні	Відсутній; температура корпусу не створює ризику опіку або займання
4	Пошкодження акумуляторного джерела	Можливе при механічному впливі або перегріванні	Відсутнє; акумулятор або powerbank не має здуття, тріщин, слідів перегріву

Таблиця 5.14 – Заходи із забезпечення охорони праці щодо пожежної небезпеки

№	Група номенклатурних заходів з ОП	Вид заходу	Критерій вибору
1	Технічні заходи	Використання корпусу з важкозаймистого або термостійкого полімерного матеріалу.	Зменшення ризику поширення займання при аварійному перегріві.
2	Технічні заходи	Застосування ізольованих провідників, надійних роз'ємів і стабілізованого джерела живлення.	Запобігання короткому замиканню та перегріванню компонентів.
3	Організаційні заходи	Наявність інструкції користувача та первинних засобів пожежогасіння у приміщенні.	Забезпечення швидкого реагування при ознаках займання.
4	Режимні заходи	Відключення пристрою від джерела живлення при запаху гару, перегріванні або нестабільній роботі.	Запобігання розвитку аварійної ситуації.
6	Експлуатаційні заходи	Повний огляд пристрою перед використанням, заборона роботи з пошкодженим кабелем або здутим акумулятором.	Своєчасне виявлення причин, що можуть призвести до займання.

5.3 Розробка «Інструкції по техніці безпеки при експлуатації пристрою для аналізу рН слини»

1. Загальні положення, стосовно прав і обов'язків обслуговуючого персоналу по виконанню вимог безпеки:

Обслуговуючий персонал або користувач, який працює з пристроєм для аналізу рН слини, повинен дотримуватися вимог охорони праці, електробезпеки, пожежної безпеки та санітарно-гігієнічних правил роботи з біологічними зразками. До роботи з системою допускаються особи, які ознайомлені з принципом роботи пристрою, схемою підключення компонентів, правилами

експлуатації рН-електрода та порядком очищення робочого місця після вимірювання.

Обслуговуючий персонал зобов'язаний:

- перевіряти справність корпусу, кабелю, роз'ємів, рН-електрода і джерела живлення перед початком роботи;
- не допускати роботи пристрою у вологому середовищі або при наявності рідини на електронних компонентах;
- використовувати рукавички під час роботи зі слиною, буферними розчинами і дезінфекційними засобами;
- припиняти експлуатацію у разі перегріву, запаху гару, нестабільної роботи, пошкодження електрода або кабелю;
- проводити очищення та дезінфекцію рН-електрода після кожного вимірювання.

Обслуговуючий персонал має право відмовити у використанні або обслуговуванні пристрою, якщо було порушено правила експлуатації: пристрій контактував з рідиною, мав механічні пошкодження, підключався до невідповідного джерела живлення або використовувався з пошкодженим рН-елекродом.

2. Положення, що стосуються прав і обов'язків користувача щодо дотримання вимог охорони праці та техніки безпеки:

- первинне складання та налаштування системи виконується особою, яка має навички роботи з електронними компонентами *Arduino*;
- експлуатація пристрою користувачем дозволяється після ознайомлення з інструкцією;
- забороняється самостійно змінювати схему підключення під час поданого живлення;
- забороняється використовувати пристрій для клінічної діагностики без відповідної сертифікації;

- отримані значення рН мають розглядатися як дослідні дані, а не як підстава для самостійного медичного висновку.

3. Технологічні вимоги щодо дотримання заходів безпеки перед початком роботи, під час роботи, після закінчення роботи та в умовах аварійної ситуації:

Перед початком роботи необхідно:

- перевірити цілісність корпусу, *USB*-кабелю, провідників, роз'ємів і рН-електрода;
- переконатися, що електронна частина пристрою суха;
- підготувати чисту робочу поверхню, одноразову ємність для зразка, дистильовану воду, буферні розчини та серветки;
- за потреби виконати калібрування рН-електрода за буферними розчинами;
- увімкнути пристрій лише після перевірки правильності підключення модулів.

Під час роботи необхідно:

- занурювати у зразок лише вимірювальну частину рН-електрода;
- не допускати потрапляння слини, буферних розчинів або води на *Arduino*, рН-модуль, дисплей, *BLE*-модуль і джерело живлення;
- не торкатися роз'ємів і контактів мокрими руками;
- не переміщувати пристрій за провідники або кабель електрода;
- контролювати показники на дисплеї та в мобільному додатку;
- не залишати пристрій увімкненим без нагляду.

Після закінчення роботи необхідно:

- вимкнути пристрій і від'єднати його від джерела живлення;
- обережно вийняти рН-електрод зі зразка;
- промити електрод дистильованою водою та провести дезінфекцію відповідно до інструкції виробника;
- утилізувати одноразові ємності, серветки та рукавички як потенційно забруднені матеріали;
- очистити й продезінфікувати робочу поверхню;

– зберігати пристрій у сухому місці, окремо від рідин і хімічних розчинів.

У разі аварійної ситуації необхідно:

- при потраплянні рідини на електроніку негайно вимкнути живлення та не вмикати пристрій до повного висихання і перевірки;
- при появі запаху гару, диму або перегріванні від'єднати пристрій від джерела живлення;
- при пошкодженні рН-електрода припинити роботу, прибрати уламки без контакту голими руками і не використовувати електрод повторно;
- при потраплянні буферного чи дезінфекційного розчину на шкіру або в очі промити уражену ділянку великою кількістю води;
- у разі займання діяти відповідно до інструкції з пожежної безпеки приміщення та використовувати первинні засоби пожежогасіння, придатні для електрообладнання.

4. Особливості обслуговування даної системи для аналізу рН слини:

Оскільки система має електронні компоненти та вимірювальну частину, яка контактує з біологічною рідиною, технічне обслуговування повинно включати регулярний огляд корпусу, кабелю, роз'ємів, рН-електрода та стану ізоляції провідників. Перед кожним використанням необхідно перевіряти відсутність тріщин на скляному електроді, сухість електронної частини, справність дисплея та коректність підключення *BLE*-модуля.

Технічне обслуговування доцільно проводити після кожного тривалого циклу експлуатації або при появі ознак нестабільної роботи: значних коливань показників рН, втрати *Bluetooth*-з'єднання, перегрівання, пошкодження кабелю або появи механічних дефектів корпусу. У випадку виявлення несправностей експлуатація системи припиняється до повного усунення причини несправності.

Висновок до розділу 5

У даному розділі проведено аналіз основних компонентів пристрою для аналізу рН слини, визначено характер взаємодії користувача з пристроєм та оцінено потенційні небезпеки, які можуть виникати під час експлуатації.

Встановлено, що основними факторами ризику є біологічна небезпека через контакт зі слиною, хімічна небезпека під час роботи з буферними та дезінфекційними розчинами, електрична небезпека при пошкодженні кабелів або потраплянні рідини на електронні компоненти, а також пожежна небезпека внаслідок короткого замикання чи перегріву джерела живлення.

Для усунення або зменшення зазначених ризиків запропоновано технічні, організаційні, режимні та експлуатаційні заходи: розміщення електронних модулів у корпусі, розділення вологої та сухої зон, використання стабілізованого джерела живлення 5 В, перевірка справності провідників і рН-електрода, застосування одноразових рукавичок, очищення і дезінфекція електрода після вимірювання. Додатково розроблено інструкцію з техніки безпеки при експлуатації системи. За умови дотримання запропонованих заходів експлуатація пристрою у лабораторних або навчально-дослідних умовах є безпечною.

ВИСНОВКИ

Пристрій реєстрації та аналізу рівня кислотності слини є актуальною розробкою у напрямі портативних біомедичних систем моніторингу. Він дозволяє контролювати рН слини як показник кислотно-лужного стану ротової порожнини та може використовуватися для спостереження за умовами, що впливають на процеси демінералізації та ремінералізації емалі. Завдяки поєднанню апаратної частини, бездротового передавання даних і мобільного додатку система забезпечує не лише реєстрацію поточного значення рН, а й фіксацію результатів, перегляд історії, побудову графіка та статистичне оброблення даних.

Поставлені задачі виконано у повному обсязі:

1. Проведено огляд фізіологічного значення рН слини, методів його вимірювання та існуючих пристроїв-аналогів. У результаті встановлено, що рН слини є динамічним показником, який залежить від харчування, гігієнічних процедур, активності мікрофлори, швидкості слиновиділення та інших чинників. Проаналізовано індикаторні, електрохімічні та оптичні методи вимірювання рН, а також визначено, що для створення прототипу найбільш доцільним є використання електрохімічного рН-сенсора з мікроконтролерною обробкою сигналу.
2. Розроблено апаратну частину пристрою на основі рН-сенсора, мікроконтролера Arduino Uno, LCD-дисплея з I2C-інтерфейсом, Bluetooth-модуля HM-10 та автономного блока живлення. Сформовано функціональну схему пристрою, виконано підбір компонентів, створено макетну схему та перевірено працездатність основних вузлів системи. Пристрій забезпечує зчитування сигналу з рН-модуля, його оброблення, локальне відображення результату на дисплеї та передавання даних до мобільного додатку.
3. Створено програмне забезпечення для мікроконтролера Arduino, яке виконує зчитування аналогового сигналу, усереднення результатів,

розрахунок значення рН, виведення даних на LCD-дисплей та передавання результатів через Bluetooth. Для взаємодії з мобільним додатком реалізовано текстовий формат передавання даних типу РН:значення, що спрощує подальше оброблення результатів у програмній частині системи.

4. Спроектовано корпус пристрою, підготовлено його деталі до 3D-друку та виконано збірку прототипу. Корпус було змодельовано з урахуванням габаритів Arduino Uno, рН-модуля, Bluetooth-модуля, дисплея, елементів живлення та з'єднувальних проводів. Передбачено вікно для LCD-дисплея, місце для розміщення електронних компонентів і можливість виведення рН-електрода назовні. Виготовлений корпус забезпечує механічний захист апаратної частини та надає пристрою завершеного конструктивного вигляду.
5. Розроблено мобільний додаток для підключення до пристрою, фіксації результатів вимірювання, перегляду історії, побудови графіка та розрахунку статистичних показників. Додаток приймає дані від Bluetooth-модуля, відображає поточне значення рН, зберігає результати з датою, часом і умовами вимірювання, а також дозволяє оцінювати зміну рН у динаміці. Завдяки цьому система переходить від разового вимірювання до накопичення й аналізу результатів моніторингу.

Також у роботі було розглянуто питання охорони праці під час експлуатації пристрою. Визначено основні потенційні небезпеки, пов'язані з контактом із біологічною рідиною, використанням хімічних розчинів, низьковольтним живленням та можливим коротким замиканням. Запропоновано технічні, організаційні, режимні та експлуатаційні заходи безпеки, а також розроблено інструкцію з техніки безпеки при роботі з пристроєм.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Timpel J., Klinghammer S., Riemenschneider L., Ibarlucea B., Cuniberti G., Hannig C., Sterzenbach T. Sensors for in situ monitoring of oral and dental health parameters in saliva. *Clinical Oral Investigations*. 2023. Vol. 27. P. 5719–5736. DOI: 10.1007/s00784-023-05206-9. URL: <https://doi.org/10.1007/s00784-023-05206-9>.
2. Sgiea E. D., Cristache C. M., Mihut T., Drafta S., Beuran I. A. The Integration of Salivary pH Meters and Artificial Intelligence in the Early Diagnosis and Management of Dental Caries in Pediatric Dentistry: A Scoping Review. *Oral*. 2025. Vol. 5. Article 12. DOI: 10.3390/oral5010012. URL: <https://doi.org/10.3390/oral5010012>.
3. Lopes P. C., Carvalho T., Gomes A. T. P. C., Veiga N., Blanco L., Correia M. J., Volpi Mello-Moura A. C. White spot lesions: diagnosis and treatment – a systematic review. *BMC Oral Health*. 2024. Vol. 24. Article 58. DOI: 10.1186/s12903-023-03720-6. URL: <https://doi.org/10.1186/s12903-023-03720-6>.
4. Roberts W. E., Mangum J. E., Schneider P. M. Pathophysiology of Demineralization, Part II: Enamel White Spots, Cavitated Caries, and Bone Infection. *Current Osteoporosis Reports*. 2022. Vol. 20. P. 106–119. DOI: 10.1007/s11914-022-00723-0. URL: <https://doi.org/10.1007/s11914-022-00723-0>.
5. Enax J., Fandrich P., Schulze zur Wiesche E., Epple M. The Remineralization of Enamel from Saliva: A Chemical Perspective. *Dentistry Journal*. 2024. Vol. 12. Article 339. DOI: 10.3390/dj12110339. URL: <https://doi.org/10.3390/dj12110339>.
6. Bowen W. H. The Stephan Curve revisited. *Odontology*. 2013. Vol. 101. P. 2–8. DOI: 10.1007/s10266-012-0082-z.
7. Jolim95. The Stephan's Curve.jpg. Wikimedia Commons. 2016. URL: https://commons.wikimedia.org/wiki/File:The_Stephan%27s_Curve.jpg (дата звернення: 01.06.2026).
8. Xu S., Mumuni A. N., Tuason R. T. S., Maki K. A. Methodological Considerations in Saliva-Based Biomarker Research: Addressing Patient-Specific Variability in Translational Research Protocols. *Current Protocols*. 2025. Vol. 5. Article e70235. DOI: 10.1002/cpz1.70235. URL: <https://doi.org/10.1002/cpz1.70235>.
9. Colorimetric and Potentiometric pH Measurement. Control.com. URL: <https://control.com/textbook/continuous-analytical-measurement/ph-measurement/> (дата звернення: 01.06.2026).
10. Psodox. PH-Elektrode.svg. Wikimedia Commons. 2007. URL: <https://commons.wikimedia.org/wiki/File:PH-Elektrode.svg> (дата звернення: 01.06.2026).
11. HORIBA. LAQUAtwin pH-22 Pocket Water Quality Meter. URL: <https://www.horiba.com/int/water-quality/products/detail/action/show/Product/laquatwin-ph-22-874/> (дата звернення: 31.05.2026).

12. Hanna Instruments. HALO2 Wireless pH Meter. URL: <https://hannainst.com/halo2/> (дата звернення: 31.05.2026).
13. Milwaukee Instruments. Milwaukee PH55 PRO Waterproof pH / Temperature Tester with ATC Replaceable Probe. URL: <https://milwaukeeinstruments.com/milwaukee-ph55-pro-waterproof-ph-temperature-tester-with-atc-replaceable-probe/> (дата звернення: 31.05.2026).
14. Apera Instruments. PH60 Premium pH Pocket Tester. URL: <https://aperainst.de/produkt/ph60-premium-pocket-ph-meter/?lang=en> (дата звернення: 31.05.2026).
15. Vernier. Go Direct pH Sensor. URL: <https://www.vernier.com/product/go-direct-ph-sensor/> (дата звернення: 31.05.2026).
16. DFRobot. Gravity: Analog pH Sensor / Meter Kit V2. URL: <https://wiki.dfrobot.com/sen0161-v2/> (дата звернення: 31.05.2026).
17. CircuitDigest. DIY Arduino pH Meter using Gravity pH Sensor. URL: <https://circuitdigest.com/microcontroller-projects/arduino-ph-meter> (дата звернення: 31.05.2026).
18. Arduino. Arduino UNO Rev3: Product Reference and Technical Specifications. URL: <https://docs.arduino.cc/hardware/uno-rev3> (дата звернення: 31.05.2026).
19. SunFounder. I2C LCD1602 Module. URL: https://docs.sunfounder.com/projects/ultimate-sensor-kit/en/latest/components_basic/21-component_i2c_lcd1602.html (дата звернення: 31.05.2026).
20. HM-10 Bluetooth 4.0 BLE Module Datasheet. Jinan Huamao Technology Co., Ltd. URL: <https://people.ece.cornell.edu/land/courses/ece4760/PIC32/uart/HM10/DSD%20TECH%20HM-10%20datasheet.pdf> (дата звернення: 31.05.2026).
21. Makerlab Electronics. 2 × 18650 Lithium Battery Shield V8 Mobile Power Expansion Board Module. URL: <https://www.makerlab-electronics.com/products/dual-18650-lithium-battery-shield-v8-mobile-power-expansion-board-module> (дата звернення: 31.05.2026).
22. Fritzing. Fritzing — electronics made easy: Getting Started. URL: <https://fritzing.org/learning/get-started/> (дата звернення: 31.05.2026).
23. SolidWorks. SOLIDWORKS 3D CAD. URL: <https://www.solidworks.com/product/solidworks-3d-cad> (дата звернення: 31.05.2026).
24. UltiMaker. UltiMaker Cura. URL: <https://ultimaker.com/software/ultimaker-cura/> (дата звернення: 31.05.2026).
25. Android Developers. Bluetooth Low Energy overview. URL: <https://developer.android.com/develop/connectivity/bluetooth/ble/ble-overview> (дата звернення: 31.05.2026).
26. Android Developers. Bluetooth permissions. URL: <https://developer.android.com/develop/connectivity/bluetooth/bt-permissions> (дата звернення: 31.05.2026).

27. Android Developers. DataStore. URL: <https://developer.android.com/topic/libraries/architecture/datastore> (дата звернення: 31.05.2026).
28. Android Developers. Save simple data with SharedPreferences. URL: <https://developer.android.com/training/data-storage/shared-preferences> (дата звернення: 31.05.2026).
29. Android Developers. Save data in a local database using Room. URL: <https://developer.android.com/training/data-storage/room> (дата звернення: 31.05.2026).
30. Android Developers. Guide to app architecture. URL: <https://developer.android.com/topic/architecture> (дата звернення: 31.05.2026).
31. Android Developers. Layouts in Views: Declare a layout. URL: <https://developer.android.com/develop/ui/views/layout/declaring-layout> (дата звернення: 31.05.2026).
32. Каштанов С. Ф., Демчук Г. В. Охорона праці в дипломних проектах (роботах): настанови до виконання розділу : навч. посіб. Київ : КПІ ім. Ігоря Сікорського, 2019. 29 с.
33. Про охорону праці : Закон України від 14.10.1992 № 2694-XII. URL: <https://zakon.rada.gov.ua/go/2694-12> (дата звернення: 31.05.2026).
34. ДСТУ ІЕС 60529:2019. Ступені захисту, забезпечувані корпусами (ІР-код) (ІЕС 60529:2013, IDT). Київ : ДП «УкрНДНЦ», 2019.
35. ДСТУ EN 61010-1:2014. Вимоги щодо безпечності контрольно-вимірювального та лабораторного електричного устаткування. Частина 1. Загальні вимоги.
36. Rajasekaran J. J., Krishnamurthy H. K., Bosco J., Jayaraman V., Krishna K., Wang T., Bei K. Oral Microbiome: A Review of Its Impact on Oral and Systemic Health. *Microorganisms*. 2024. Vol. 12. Article 1797. DOI: 10.3390/microorganisms12091797. URL: <https://doi.org/10.3390/microorganisms12091797>.
37. ДСТУ EN 61140:2019. Захист проти ураження електричним струмом. Загальні аспекти щодо установок та обладнання.
38. Про затвердження Правил пожежної безпеки в Україні : наказ МВС України від 30.12.2014 № 1417. URL: <https://zakon.rada.gov.ua/go/z0252-15> (дата звернення: 31.05.2026).
39. Про затвердження Технічного регламенту низьковольтного електричного обладнання : постанова Кабінету Міністрів України від 16.12.2015 № 1067. URL: <https://zakon.rada.gov.ua/go/1067-2015-%D0%BF> (дата звернення: 31.05.2026).

ДОДАТОК А

Код програми для роботи пристрою у середовищі розробки *Arduino*

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>

// Піни
#define PH_PIN A0

// HM-10:
// TX HM-10 -> D10 Arduino
// RX HM-10 -> D11 Arduino
#define BT_RX 10
#define BT_TX 11

SoftwareSerial BT(BT_RX, BT_TX);

// Якщо дисплей не працює з адресою 0x27
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Налаштування pH
float calibration_value = 21.34;

// Інтервал автоматичного надсилання pH у додаток
const unsigned long SEND_INTERVAL = 1000;
unsigned long lastSendTime = 0;

// false – пристрій очікує команду START з додатку
// true – пристрій одразу починає передавати pH
bool streaming = false;

// Буфер для команд з Bluetooth
String btCommand = "";
unsigned long lastCommandCharTime = 0;

// Налаштування
void setup() {
  Serial.begin(9600);
  BT.begin(9600);

  lcd.init();
  lcd.backlight();

  printLCDLine(0, "Analiz pH slyny");
  printLCDLine(1, "Pidkluch dodat.");
  delay(2000);

  printLCDLine(0, "pH: --");
```

```

printLCDLine(1, "Start u dodatku");

// Технічна інформація тільки для Serial Monitor
Serial.println("Arduino pH meter started");
Serial.println("HM-10 pins: TX->D10, RX->D11");
Serial.println("Commands: PING, READ, START, STOP");
Serial.println("Bluetooth app data format: PH:6.35");

// У Bluetooth надсилаємо статус без чисел,
// щоб додаток не сприйняв службове повідомлення як pH.
sendBTStatus("READY");
}

// Основний цикл
void loop() {
    readBluetoothCommands();

    // Обробка команди без Enter / New line
    if (btCommand.length() > 0 && millis() - lastCommandCharTime > 300) {
        processCommand(btCommand);
        btCommand = "";
    }

    if (streaming && millis() - lastSendTime >= SEND_INTERVAL) {
        lastSendTime = millis();
        sendPHData();
    }
}

// Зчитування pH з усередненням
float readPHVoltage(int &rawAverage) {
    int buffer_arr[10];

    for (int i = 0; i < 10; i++) {
        buffer_arr[i] = analogRead(PH_PIN);
        delay(30);
    }

    // Сортування значень
    for (int i = 0; i < 9; i++) {
        for (int j = i + 1; j < 10; j++) {
            if (buffer_arr[i] > buffer_arr[j]) {
                int temp = buffer_arr[i];
                buffer_arr[i] = buffer_arr[j];
                buffer_arr[j] = temp;
            }
        }
    }
}

```

```

// Відкидаємо 2 найменші та 2 найбільші значення
long avgval = 0;
for (int i = 2; i < 8; i++) {
    avgval += buffer_arr[i];
}

rawAverage = avgval / 6;

float voltage = rawAverage * 5.0 / 1023.0;
return voltage;
}

// Розрахунок pH
float calculatePH(float voltage) {
    float pHValue = -5.70 * voltage + calibration_value;
    return pHValue;
}

// Передача pH-даних
void sendPHData() {
    int raw = 0;
    float voltage = readPHVoltage(raw);
    float pHValue = calculatePH(voltage);

    unsigned long timeSeconds = millis() / 1000;

    // 1. Детальний технічний вивід тільки в Serial Monitor
    Serial.print("time_s=");
    Serial.print(timeSeconds);
    Serial.print(", raw=");
    Serial.print(raw);
    Serial.print(", voltage_V=");
    Serial.print(voltage, 3);
    Serial.print(", PH=");
    Serial.println(pHValue, 2);

    // 2. Чистий формат для мобільного додатку
    BT.print("PH:");
    BT.println(pHValue, 2);

    // 3. Зручний користувацький вивід на LCD без додаткових статусів
    String pHLine = "pH:";
    pHLine += String(pHValue, 2);

    printLCDLine(0, pHLine);

    if (streaming) {
        printLCDLine(1, "Vymiriuвання...");
    } else {

```

```

        printLCDLine(1, "Start u dodatku");
    }
}

```

```

// Прием команд з телефона
void readBluetoothCommands() {
    while (BT.available()) {
        char c = BT.read();

        lastCommandCharTime = millis();

        if (c == '\n' || c == '\r') {
            if (btCommand.length() > 0) {
                processCommand(btCommand);
                btCommand = "";
            }
        } else {
            btCommand += c;
        }
    }
}

```

```

// Обработка команд
void processCommand(String cmd) {
    cmd.trim();
    cmd.toUpperCase();

    Serial.print("BT command: ");
    Serial.println(cmd);

    if (cmd == "PING") {
        sendBTStatus("OK");
    }

    else if (cmd == "READ") {
        sendPHData();
    }

    else if (cmd == "START") {
        streaming = true;
        sendBTStatus("STARTED");
        sendPHData();
        lastSendTime = millis();
    }

    else if (cmd == "STOP") {
        streaming = false;
        sendBTStatus("STOPPED");
    }
}

```

```

        printLCDLine(1, "Start u dodatku");
    }

    else {
        sendBTStatus("UNKNOWN_COMMAND");
    }
}

// Службові повідомлення у Bluetooth
void sendBTStatus(String status) {
    Serial.print("STATUS:");
    Serial.println(status);

    BT.print("STATUS:");
    BT.println(status);
}

// Зручний вивід рядка на LCD 16x2
void printLCDLine(byte row, String text) {
    lcd.setCursor(0, row);

    if (text.length() > 16) {
        text = text.substring(0, 16);
    }

    lcd.print(text);

    for (int i = text.length(); i < 16; i++) {
        lcd.print(" ");
    }
}

```

ДОДАТОК Б



Рисунок 1 – Креслення кришки

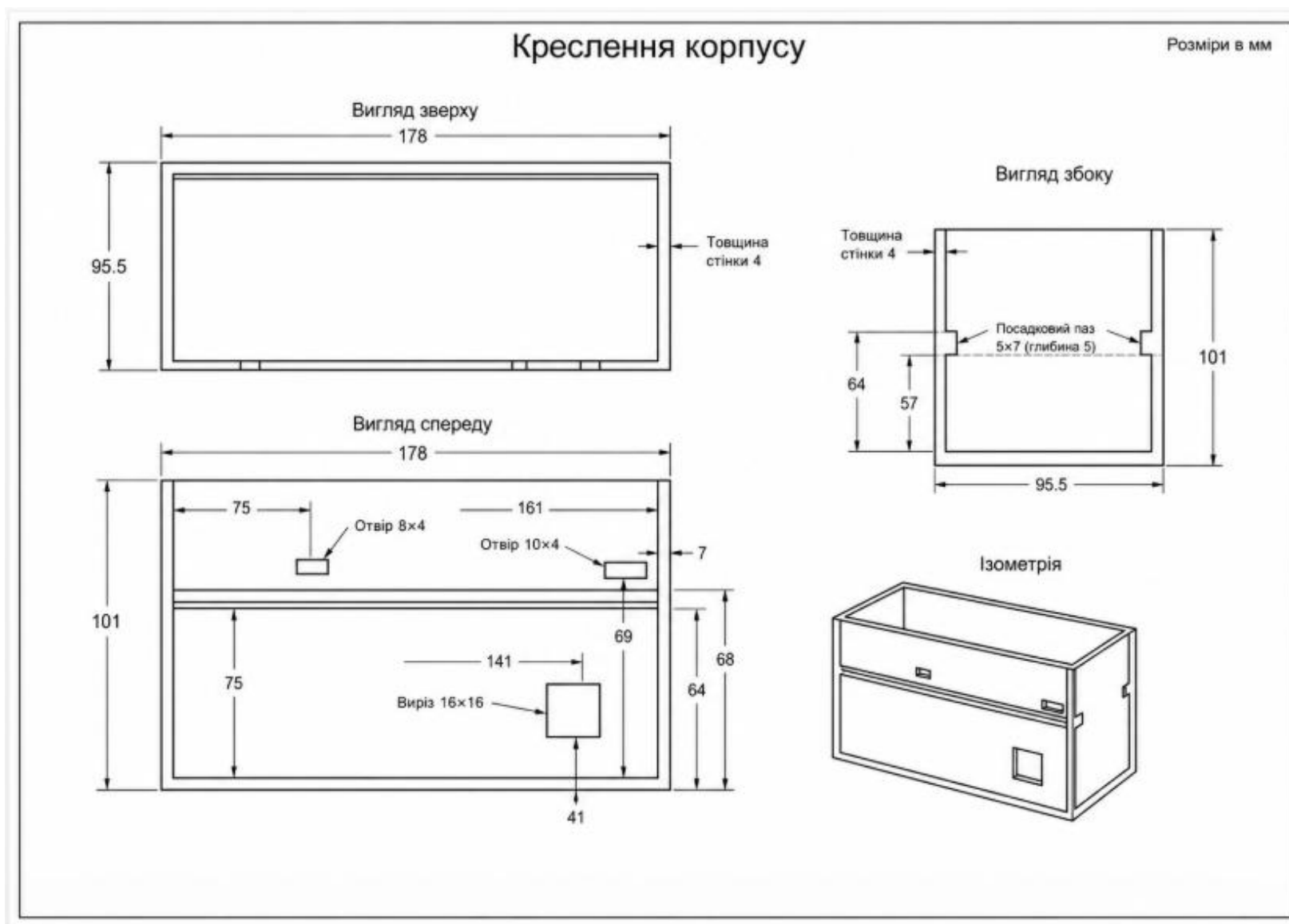


Рисунок 2 – Креслення корпусу

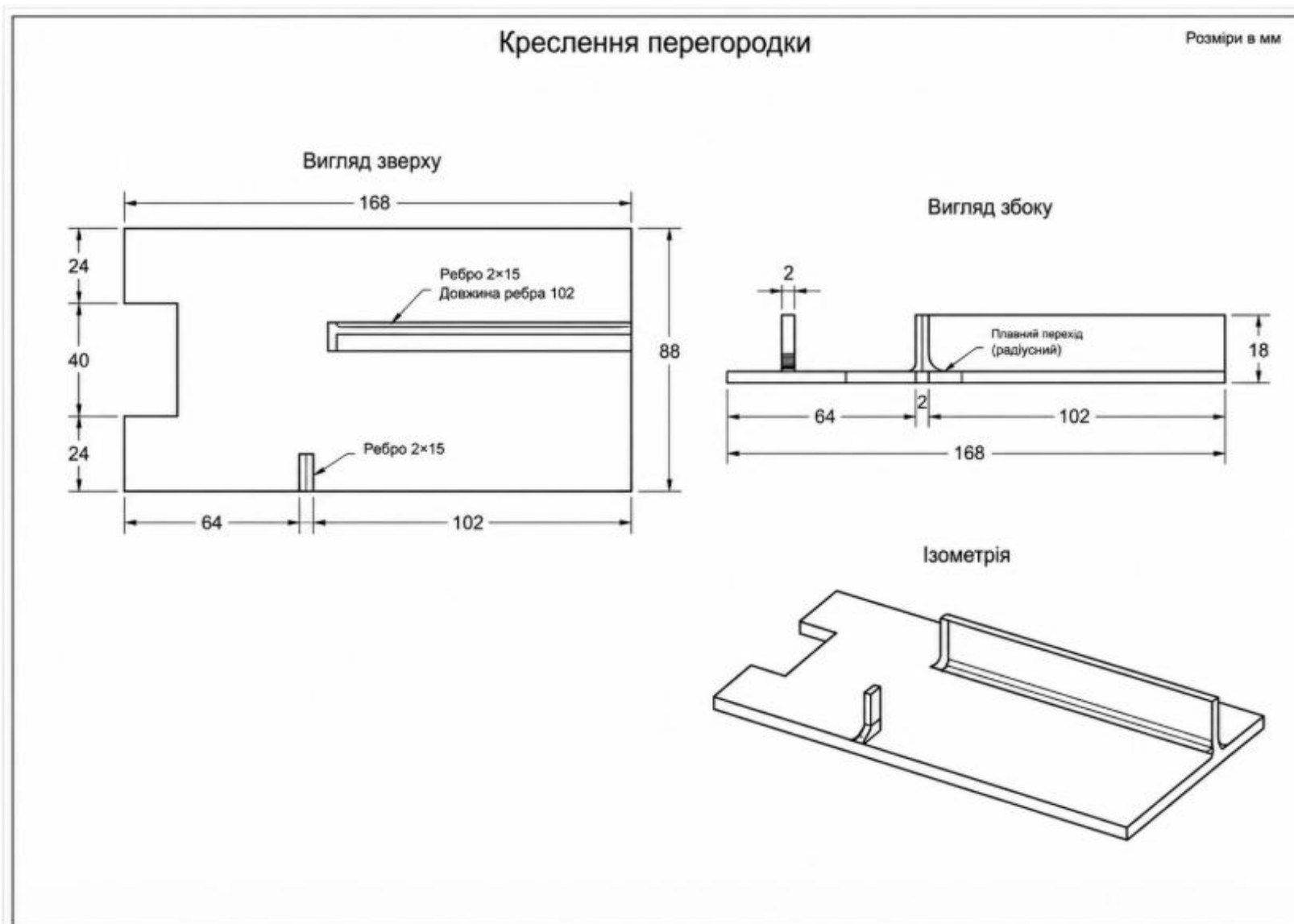


Рисунок 3 – Креслення перегородки

ДОДАТОК В

Код програми додатку для фіксації і обробки даних отриманих з пристрою

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#F3F7FA"
    android:fillViewport="true">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="20dp">

        <TextView
            android:id="@+id/titleTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Моніторинг рН слини"
            android:textSize="26sp"
            android:textStyle="bold"
            android:gravity="center"
            android:textColor="#1F2933"
            android:layout_marginBottom="8dp" />

        <TextView
            android:id="@+id/subtitleTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Система вимірювання, збереження та аналізу рН"
            android:textSize="14sp"
            android:gravity="center"
            android:textColor="#52616B"
            android:layout_marginBottom="20dp" />

        <TextView
            android:id="@+id/statusTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Статус: пристрій не підключено"
            android:textSize="16sp"
            android:gravity="center"
            android:textColor="#1F2933"
            android:background="#E3F2FD"
            android:padding="12dp"
            android:layout_marginBottom="18dp" />

        <LinearLayout
            android:id="@+id/phCardLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:gravity="center"
            android:background="FFFFFF"
            android:padding="20dp"
            android:layout_marginBottom="20dp">
```

```

<TextView
    android:id="@+id/phLabelTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Поточне значення pH"
    android:textSize="17sp"
    android:textColor="#52616B"
    android:gravity="center" />

<TextView
    android:id="@+id/phValueTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="-"
    android:textSize="60sp"
    android:textStyle="bold"
    android:textColor="#00796B"
    android:gravity="center"
    android:layout_marginTop="6dp"
    android:layout_marginBottom="4dp" />

<TextView
    android:id="@+id/phHintTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Після вимірювання тут з'явиться усереднений
результат"
    android:textSize="13sp"
    android:textColor="#7B8794"
    android:gravity="center" />

</LinearLayout>

<TextView
    android:id="@+id/instructionTitleTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Порядок роботи"
    android:textSize="18sp"
    android:textStyle="bold"
    android:textColor="#1F2933"
    android:layout_marginBottom="6dp" />

<TextView
    android:id="@+id/instructionTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="1. Увімкніть пристрій.\n2. Підключіться через
Bluetooth.\n3. Помістіть сенсор у зразок.\n4. Натисніть «Старт вимірювання».\n5.
Після завершення збережіть результат."
    android:textSize="14sp"
    android:textColor="#52616B"
    android:lineSpacingExtra="3dp"
    android:background="#FFFFFF"
    android:padding="14dp"
    android:layout_marginBottom="18dp" />

<Button
    android:id="@+id/scanButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Підключити пристрій"

```

```

        android:layout_marginBottom="10dp" />

<Button
    android:id="@+id/startMeasurementButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Старт вимірювання"
    android:enabled="false"
    android:layout_marginBottom="10dp" />

<Button
    android:id="@+id/saveResultButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Зафіксувати результат"
    android:enabled="false"
    android:layout_marginBottom="10dp" />

<Button
    android:id="@+id/historyButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Історія вимірювань"
    android:layout_marginBottom="10dp" />

<Button
    android:id="@+id/statisticsButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Статистика"
    android:layout_marginBottom="10dp" />

<Button
    android:id="@+id/graphButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Графік pH"
    android:layout_marginBottom="10dp" />

<Button
    android:id="@+id/exportButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Експорт CSV"
    android:layout_marginBottom="10dp" />

<Button
    android:id="@+id/aboutButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Про додаток"
    android:layout_marginBottom="20dp" />

<TextView
    android:id="@+id/devicesTitleTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Інформація"
    android:textSize="18sp"
    android:textStyle="bold"
    android:textColor="#1F2933"
    android:layout_marginBottom="8dp" />

```

```

        <TextView
            android:id="@+id/devicesTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Поки що пристрій не підключено"
            android:textSize="15sp"
            android:textColor="#52616B"
            android:background="#FFFFFF"
            android:padding="14dp"
            android:layout_marginBottom="24dp" />

    </LinearLayout>
</ScrollView>

```

AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-feature
        android:name="android.hardware.bluetooth_le"
        android:required="true" />

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

    <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
    <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="Saliva pH Monitor"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.SalivaPHMonitor">

        <activity
            android:name=".MainActivity"
            android:exported="true">

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

        </activity>

    </application>
</manifest>

```

MainActivity.kt

```

package com.example.salivaphmonitor

import android.Manifest
import android.annotation.SuppressLint
import android.app.AlertDialog
import android.bluetooth.BluetoothAdapter
import android.bluetooth.BluetoothDevice
import android.bluetooth.BluetoothGatt
import android.bluetooth.BluetoothGattCallback
import android.bluetooth.BluetoothGattCharacteristic
import android.bluetooth.BluetoothGattDescriptor
import android.bluetooth.BluetoothGattService
import android.bluetooth.BluetoothManager
import android.bluetooth.BluetoothProfile
import android.bluetooth.le.ScanCallback
import android.bluetooth.le.ScanResult
import android.bluetooth.le.ScanSettings
import android.content.Context
import android.content.pm.PackageManager
import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import android.graphics.Path
import android.os.Build
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.text.InputType
import android.view.View
import android.widget.AdapterView
import android.widget.Button
import android.widget.EditText
import android.widget.LinearLayout
import android.widget.ScrollView
import android.widget.Spinner
import android.widget.TextView
import androidx.activity.enableEdgeToEdge
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import androidx.core.content.ContextCompat
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import org.json.JSONArray
import org.json.JSONObject
import java.text.SimpleDateFormat
import java.util.Calendar
import java.util.Date
import java.util.Locale
import java.util.UUID
import kotlin.math.abs
import kotlin.math.max
import kotlin.math.min

class MainActivity : AppCompatActivity() {

    private lateinit var statusTextView: TextView
    private lateinit var phValueTextView: TextView
    private lateinit var devicesTextView: TextView
    private lateinit var scanButton: Button
    private lateinit var startMeasurementButton: Button
    private lateinit var saveResultButton: Button
    private lateinit var historyButton: Button

```

```

private lateinit var statisticsButton: Button
private lateinit var graphButton: Button
private lateinit var exportButton: Button
private lateinit var aboutButton: Button

private var bluetoothAdapter: BluetoothAdapter? = null
private var bluetoothGatt: BluetoothGatt? = null

private var selectedDevice: BluetoothDevice? = null
private var notifyCharacteristic: BluetoothGattCharacteristic? = null
private var writeCharacteristic: BluetoothGattCharacteristic? = null

private val handler = Handler(Looper.getMainLooper())

private val scanPeriod: Long = 12000
private var isScanning = false
private var isConnected = false

private val measurementDurationMs: Long = 5000
private var isMeasuring = false
private val phReadings = mutableListOf<Double>()
private var finalPhValue: Double? = null
private var finalStabilityText: String = ""

private val foundDevices = mutableListOf<FoundBleDevice>()
private val incomingBuffer = StringBuilder()

private val savedMeasurements = mutableListOf<SavedMeasurement>()

private val preferencesName = "saliva_ph_preferences"
private val measurementsKey = "saved_measurements"
private var pendingCsvText: String? = null

private val hm10ServiceUuid: UUID =
    UUID.fromString("0000ffe0-0000-1000-8000-00805f9b34fb")

private val hm10CharacteristicUuid: UUID =
    UUID.fromString("0000ffe1-0000-1000-8000-00805f9b34fb")

private val clientCharacteristicConfigUuid: UUID =
    UUID.fromString("00002902-0000-1000-8000-00805f9b34fb")

private data class FoundBleDevice(
    val device: BluetoothDevice,
    val name: String,
    val address: String,
    val rssi: Int
)

private data class SavedMeasurement(
    val phValue: Double,
    val date: String,
    val time: String,
    val condition: String,
    val note: String,
    val stability: String
)

private val requestPermissionsLauncher =
registerForActivityResult(ActivityResultContracts.RequestMultiplePermissions())
{ permissions ->

```

```

        val allGranted = permissions.values.all { it }

        statusTextView.text = if (allGranted) {
            "Статус: дозволи надано, можна сканувати BLE-пристрої"
        } else {
            "Статус: не всі дозволи надано"
        }
    }

    private val createCsvFileLauncher =
registerForActivityResult(ActivityResultContracts.CreateDocument("text/csv")) {
uri ->
    if (uri == null) {
        statusTextView.text = "Статус: експорт скасовано"
        return@registerForActivityResult
    }

    val csvText = pendingCsvText

    if (csvText == null) {
        statusTextView.text = "Статус: немає даних для експорту"
        return@registerForActivityResult
    }

    try {
        contentResolver.openOutputStream(uri)?.use { outputStream ->
            outputStream.write(csvText.toByteArray(Charsets.UTF_8))
        }

        statusTextView.text = "Статус: CSV-файл збережено"
    } catch (_: Exception) {
        statusTextView.text = "Статус: помилка експорту CSV"
    }
}

    private val scanCallback = object : ScanCallback() {

        @SuppressWarnings("MissingPermission")
        override fun onScanResult(callbackType: Int, result: ScanResult) {
            super.onScanResult(callbackType, result)

            val device = result.device
            val address = device.address
            val name = device.name
                ?: result.scanRecord?.deviceName
                ?: "Без назви"

            val existingIndex = foundDevices.indexOfFirst { it.address ==
address }

            val item = FoundBleDevice(
                device = device,
                name = name,
                address = address,
                rssi = result.rssi
            )

            if (existingIndex >= 0) {
                foundDevices[existingIndex] = item
            } else {
                foundDevices.add(item)
            }
        }
    }

```

```

    }

    runOnUiThread {
        showFoundDevicesText()
    }
}

override fun onScanFailed(errorCode: Int) {
    super.onScanFailed(errorCode)

    runOnUiThread {
        statusTextView.text = "Статус: помилка BLE-сканування, код:
$errorCode"
    }
}

private val gattCallback = object : BluetoothGattCallback() {

    @SuppressWarnings("MissingPermission")
    override fun onConnectionStateChange(gatt: BluetoothGatt, status: Int,
newState: Int) {
        super.onConnectionStateChange(gatt, status, newState)

        if (newState == BluetoothProfile.STATE_CONNECTED) {
            isConnected = true

            runOnUiThread {
                statusTextView.text = "Статус: підключено, пошук
сервісів..."
            }

            gatt.discoverServices()
        } else if (newState == BluetoothProfile.STATE_DISCONNECTED) {
            isConnected = false
            isMeasuring = false

            runOnUiThread {
                statusTextView.text = "Статус: пристрій відключено"
                startMeasurementButton.text = "Підключитись"
                startMeasurementButton.isEnabled = true
                saveResultButton.isEnabled = false
            }

            closeGatt()
        }
    }

    override fun onServicesDiscovered(gatt: BluetoothGatt, status: Int) {
        super.onServicesDiscovered(gatt, status)

        if (status == BluetoothGatt.GATT_SUCCESS) {
            setupGattCharacteristics(gatt)
        } else {
            runOnUiThread {
                statusTextView.text = "Статус: не вдалося знайти BLE-
сервіси"
            }
        }
    }
}

@Deprecated("Deprecated in Java")

```

```

        override fun onCharacteristicChanged(
            gatt: BluetoothGatt,
            characteristic: BluetoothGattCharacteristic
        ) {
            super.onCharacteristicChanged(gatt, characteristic)

            val value = characteristic.value ?: return
            handleIncomingBytes(value)
        }

        override fun onCharacteristicChanged(
            gatt: BluetoothGatt,
            characteristic: BluetoothGattCharacteristic,
            value: ByteArray
        ) {
            super.onCharacteristicChanged(gatt, characteristic, value)

            handleIncomingBytes(value)
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
view, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            view.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom)
            insets
        }

        statusTextView = findViewById(R.id.statusTextView)
        phValueTextView = findViewById(R.id.phValueTextView)
        devicesTextView = findViewById(R.id.devicesTextView)
        scanButton = findViewById(R.id.scanButton)
        startMeasurementButton = findViewById(R.id.startMeasurementButton)
        saveResultButton = findViewById(R.id.saveResultButton)
        historyButton = findViewById(R.id.historyButton)
        statisticsButton = findViewById(R.id.statisticsButton)
        graphButton = findViewById(R.id.graphButton)
        exportButton = findViewById(R.id.exportButton)
        aboutButton = findViewById(R.id.aboutButton)

        loadMeasurementsFromStorage()

        val bluetoothManager =
            getSystemService(Context.BLUETOOTH_SERVICE) as BluetoothManager

        bluetoothAdapter = bluetoothManager.adapter

        requestBluetoothPermissions()

        scanButton.setOnClickListener {
            startBleScan()
        }

        startMeasurementButton.setOnClickListener {

```

```

        if (!isConnected) {
            connectToSelectedDevice()
        } else {
            startMeasurement()
        }
    }

    saveResultButton.setOnClickListener {
        val ph = finalPhValue

        if (ph != null) {
            showSaveMeasurementDialog(ph)
        } else {
            statusTextView.text = "Статус: спочатку виконайте вимірювання"
        }
    }

    historyButton.setOnClickListener {
        showHistoryDialog()
    }

    statisticsButton.setOnClickListener {
        showStatisticsPeriodDialog()
    }

    graphButton.setOnClickListener {
        showGraphPeriodDialog()
    }

    exportButton.setOnClickListener {
        exportMeasurementsToCsv()
    }

    aboutButton.setOnClickListener {
        showAboutDialog()
    }

    startMeasurementButton.isEnabled = false
    saveResultButton.isEnabled = false
}

private fun requestBluetoothPermissions() {
    val permissions = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
        arrayOf(
            Manifest.permission.BLUETOOTH_SCAN,
            Manifest.permission.BLUETOOTH_CONNECT,
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.ACCESS_COARSE_LOCATION
        )
    } else {
        arrayOf(
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.ACCESS_COARSE_LOCATION
        )
    }

    requestPermissionsLauncher.launch(permissions)
}

private fun hasBluetoothPermissions(): Boolean {
    val fineLocationPermission = ContextCompat.checkSelfPermission(
        this,

```

```

        Manifest.permission.ACCESS_FINE_LOCATION
    ) == PackageManager.PERMISSION_GRANTED

    val coarseLocationPermission = ContextCompat.checkSelfPermission(
        this,
        Manifest.permission.ACCESS_COARSE_LOCATION
    ) == PackageManager.PERMISSION_GRANTED

    return if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
        val scanPermission = ContextCompat.checkSelfPermission(
            this,
            Manifest.permission.BLUETOOTH_SCAN
        ) == PackageManager.PERMISSION_GRANTED

        val connectPermission = ContextCompat.checkSelfPermission(
            this,
            Manifest.permission.BLUETOOTH_CONNECT
        ) == PackageManager.PERMISSION_GRANTED

        scanPermission && connectPermission && fineLocationPermission &&
coarseLocationPermission
    } else {
        fineLocationPermission && coarseLocationPermission
    }
}

@SuppressLint("MissingPermission")
private fun startBleScan() {
    if (!hasBluetoothPermissions()) {
        statusTextView.text = "Статус: спочатку надайте Bluetooth-дозволи"
        requestBluetoothPermissions()
        return
    }

    if (bluetoothAdapter == null) {
        statusTextView.text = "Статус: Bluetooth не підтримується на цьому
телефоні"
        return
    }

    if (bluetoothAdapter?.isEnabled == false) {
        statusTextView.text = "Статус: увімкніть Bluetooth на телефоні"
        return
    }

    if (isScanning) {
        return
    }

    foundDevices.clear()
    selectedDevice = null
    startMeasurementButton.isEnabled = false
    saveResultButton.isEnabled = false
    finalPhValue = null
    finalStabilityText = ""

    devicesTextView.text = "Сканування BLE-пристроїв..."
    statusTextView.text = "Статус: пошук BLE-пристроїв"

    val settings = ScanSettings.Builder()
        .setScanMode(ScanSettings.SCAN_MODE_LOW_LATENCY)
        .build()

```

```

        isScanning = true

        bluetoothAdapter?.bluetoothLeScanner?.startScan(
            null,
            settings,
            scanCallback
        )

        handler.postDelayed({
            stopBleScan()

            if (foundDevices.isEmpty()) {
                statusTextView.text = "Статус: BLE-пристрої не знайдено"
                devicesTextView.text =
                    "BLE-пристрої не знайдено.\nПеревірте, чи Arduino/HM-10
увімкнений."
            } else {
                statusTextView.text = "Статус: сканування завершено"
                showDeviceSelectionDialog()
            }
        }, scanPeriod)
    }

    @SuppressWarnings("MissingPermission")
    private fun stopBleScan() {
        if (isScanning && hasBluetoothPermissions()) {
            bluetoothAdapter?.bluetoothLeScanner?.stopScan(scanCallback)
            isScanning = false
        }
    }

    private fun showFoundDevicesText() {
        if (foundDevices.isEmpty()) {
            devicesTextView.text = "Поки що пристрої не знайдено"
            return
        }

        val text = StringBuilder()
        text.append("Знайдені BLE-пристрої:\n\n")

        foundDevices.forEachIndexed { index, item ->
            text.append("${index + 1}. ${item.name}\n")
            text.append("MAC: ${item.address}\n")
            text.append("RSSI: ${item.rssi} dBm\n\n")
        }

        devicesTextView.text = text.toString()
    }

    private fun showDeviceSelectionDialog() {
        val deviceNames = foundDevices.map { item ->
            "${item.name}\n${item.address}"
        }.toTypedArray()

        AlertDialog.Builder(this)
            .setTitle("Оберіть Bluetooth LE пристрій")
            .setItems(deviceNames) { _, which ->
                val selected = foundDevices[which]
                selectedDevice = selected.device

                statusTextView.text =

```

```

        "Статус: обрано ${selected.name}"

        devicesTextView.text =
            "Обраний пристрій:\nНазва: ${selected.name}\nMAC:
${selected.address}"

        startMeasurementButton.isEnabled = true
        startMeasurementButton.text = "Підключитись"
    }
    .setNegativeButton("Скасувати", null)
    .show()
}

@SuppressLint("MissingPermission")
private fun connectToSelectedDevice() {
    if (!hasBluetoothPermissions()) {
        statusTextView.text = "Статус: спочатку надайте Bluetooth-дозволи"
        requestBluetoothPermissions()
        return
    }

    val device = selectedDevice

    if (device == null) {
        statusTextView.text = "Статус: спочатку натисніть Сканувати й
оберіть пристрій"
        return
    }

    statusTextView.text = "Статус: підключення до BLE-пристрою..."
    startMeasurementButton.isEnabled = false

    closeGatt()

    bluetoothGatt = device.connectGatt(
        this,
        false,
        gattCallback
    )
}

@SuppressLint("MissingPermission")
private fun setupGattCharacteristics(gatt: BluetoothGatt) {
    val hm10Service = gatt.getService(hm10ServiceUuid)
    val hm10Characteristic =
hm10Service?.getCharacteristic(hm10CharacteristicUuid)

    if (hm10Characteristic != null) {
        notifyCharacteristic = hm10Characteristic
        writeCharacteristic = hm10Characteristic
    } else {
        findSuitableCharacteristics(gatt.services)
    }

    val notifyChar = notifyCharacteristic

    if (notifyChar == null) {
        runOnUiThread {
            statusTextView.text =
                "Статус: не знайдено характеристики для отримання даних"
            startMeasurementButton.isEnabled = true
        }
    }
}

```

```

        return
    }

    enableNotifications(gatt, notifyChar)

    runOnUiThread {
        statusTextView.text = "Статус: підключено, можна почати вимірювання"
        startMeasurementButton.text = "Старт вимірювання"
        startMeasurementButton.isEnabled = true
        saveResultButton.isEnabled = false
    }
}

private fun findSuitableCharacteristics(services:
List<BluetoothGattService>) {
    notifyCharacteristic = null
    writeCharacteristic = null

    for (service in services) {
        for (characteristic in service.characteristics) {
            val properties = characteristic.properties

            val canNotify =
                properties and BluetoothGattCharacteristic.PROPERTY_NOTIFY
!= 0 ||
                properties and
BluetoothGattCharacteristic.PROPERTY_INDICATE != 0

            val canWrite =
                properties and BluetoothGattCharacteristic.PROPERTY_WRITE !=
0 ||
                properties and
BluetoothGattCharacteristic.PROPERTY_WRITE_NO_RESPONSE != 0

            if (canNotify && notifyCharacteristic == null) {
                notifyCharacteristic = characteristic
            }

            if (canWrite && writeCharacteristic == null) {
                writeCharacteristic = characteristic
            }
        }
    }
}

@SuppressLint("MissingPermission")
private fun enableNotifications(
    gatt: BluetoothGatt,
    characteristic: BluetoothGattCharacteristic
) {
    val notificationEnabled =
gatt.setCharacteristicNotification(characteristic, true)

    if (!notificationEnabled) {
        runOnUiThread {
            statusTextView.text = "Статус: не вдалося увімкнути
notification"
        }
        return
    }

    val descriptor =

```

```

characteristic.getDescriptor(clientCharacteristicConfigUuid)

    if (descriptor != null) {
        val properties = characteristic.properties

        val descriptorValue =
            if (properties and BluetoothGattCharacteristic.PROPERTY_INDICATE
!= 0) {
                BluetoothGattDescriptor.ENABLE_INDICATION_VALUE
            } else {
                BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE
            }

        @Suppress("DEPRECATION")
        descriptor.value = descriptorValue

        @Suppress("DEPRECATION")
        gatt.writeDescriptor(descriptor)
    }
}

private fun startMeasurement() {
    if (!isConnected) {
        statusTextView.text = "Статус: пристрій не підключено"
        return
    }

    if (isMeasuring) {
        return
    }

    phReadings.clear()
    finalPhValue = null
    finalStabilityText = ""
    incomingBuffer.clear()

    isMeasuring = true

    phValueTextView.text = "-"
    statusTextView.text = "Статус: вимірювання триває..."
    startMeasurementButton.text = "Вимірювання..."
    startMeasurementButton.isEnabled = false
    saveResultButton.isEnabled = false

    sendCommandToDevice("START\n")

    handler.postDelayed({
        finishMeasurement()
    }, measurementDurationMs)
}

private fun finishMeasurement() {
    if (!isMeasuring) {
        return
    }

    isMeasuring = false

    sendCommandToDevice("STOP\n")

    if (phReadings.isEmpty()) {
        statusTextView.text = "Статус: pH-дані не отримано, повторіть

```

```

вимірювання"
    phValueTextView.text = "-"
    startMeasurementButton.text = "Старт вимірювання"
    startMeasurementButton.isEnabled = true
    saveResultButton.isEnabled = false
    return
}

val average = phReadings.average()
val min = phReadings.minOrNull() ?: average
val max = phReadings.maxOrNull() ?: average
val range = abs(max - min)

finalPhValue = average

val stabilityText = if (range <= 0.15) {
    "стабільне"
} else {
    "нестабільне"
}

finalStabilityText = stabilityText

phValueTextView.text = String.format(Locale.US, "%.2f", average)

statusTextView.text =
    "Статус: вимірювання завершено, результат $stabilityText"

devicesTextView.text =
    "Останнє вимірювання:\n" +
    "Кількість значень: ${phReadings.size}\n" +
    "Середнє рН: ${String.format(Locale.US, "%.2f", average)}\n"
+
    "Мінімум: ${String.format(Locale.US, "%.2f", min)}\n" +
    "Максимум: ${String.format(Locale.US, "%.2f", max)}\n" +
    "Розкид: ${String.format(Locale.US, "%.2f", range)}"

startMeasurementButton.text = "Старт вимірювання"
startMeasurementButton.isEnabled = true
saveResultButton.isEnabled = true
}

@SuppressLint("MissingPermission")
private fun sendCommandToDevice(command: String): Boolean {
    val gatt = bluetoothGatt
    val characteristic = writeCharacteristic

    if (gatt == null || characteristic == null) {
        statusTextView.text = "Статус: не знайдено характеристики для
надсилання команди"
        return false
    }

    val properties = characteristic.properties

    characteristic.writeType =
        if (properties and
BluetoothGattCharacteristic.PROPERTY_WRITE_NO_RESPONSE != 0) {
            BluetoothGattCharacteristic.WRITE_TYPE_NO_RESPONSE
        } else {
            BluetoothGattCharacteristic.WRITE_TYPE_DEFAULT
        }
}

```

```

@Suppress("DEPRECATION")
characteristic.value = command.toByteArray(Charsets.UTF_8)

@Suppress("DEPRECATION")
return gatt.writeCharacteristic(characteristic)
}

private fun handleIncomingBytes(value: ByteArray) {
    val text = value.toString(Charsets.UTF_8)
    incomingBuffer.append(text)

    var normalized = incomingBuffer.toString().replace("\r", "\n")

    while (normalized.contains("\n")) {
        val lineEndIndex = normalized.indexOf("\n")
        val line = normalized.substring(0, lineEndIndex).trim()

        processIncomingLine(line)

        normalized = normalized.substring(lineEndIndex + 1)
    }

    incomingBuffer.clear()
    incomingBuffer.append(normalized)

    if (incomingBuffer.length > 100) {
        incomingBuffer.clear()
    }
}

private fun processIncomingLine(line: String) {
    if (line.isBlank()) {
        return
    }

    val phMatch = Regex("""PH\s*[:=]\s*(\d+(?:[.]\d+)?)""").find(line)

    if (phMatch != null) {
        val ph = phMatch.groupValues[1]
            .replace(',', '.')
            .toDoubleOrNull()

        if (ph != null) {
            if (isMeasuring) {
                phReadings.add(ph)
            }

            runOnUiThread {
                phValueTextView.text = String.format(Locale.US, "%.2f", ph)

                if (isMeasuring) {
                    statusTextView.text =
                        "Статус: вимірювання... отримано ${phReadings.size}
знач."
                } else {
                    statusTextView.text =
                        "Статус: отримано pH = ${String.format(Locale.US,
                        "%.2f", ph)}"
                }
            }
        }
    }
}

```

```

    }
}

private fun showSaveMeasurementDialog(ph: Double) {
    val now = Date()

    val dateFormat = SimpleDateFormat("dd.MM.yyyy", Locale.getDefault())
    val timeFormat = SimpleDateFormat("HH:mm", Locale.getDefault())

    val dateEditText = EditText(this).apply {
        hint = "Дата"
        setText(dateFormat.format(now))
        inputType = InputType.TYPE_CLASS_TEXT
    }

    val timeEditText = EditText(this).apply {
        hint = "Час"
        setText(timeFormat.format(now))
        inputType = InputType.TYPE_CLASS_TEXT
    }

    val conditions = listOf(
        "До їжі",
        "Після їжі",
        "Після чищення зубів",
        "Після ремінералізуючого засобу",
        "Контрольне вимірювання",
        "Інше"
    )

    val adapter = ArrayAdapter(
        this,
        android.R.layout.simple_spinner_item,
        conditions
    ).apply {

setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
    }

    val conditionSpinner = Spinner(this).apply {
        this.adapter = adapter
    }

    val noteEditText = EditText(this).apply {
        hint = "Примітка"
        inputType = InputType.TYPE_CLASS_TEXT or
InputType.TYPE_TEXT_FLAG_MULTI_LINE
        minLines = 2
    }

    val container = LinearLayout(this).apply {
        orientation = LinearLayout.VERTICAL
        setPadding(48, 16, 48, 0)

        addView(TextView(this@MainActivity).apply {
            text = "pH: ${String.format(Locale.US, "%.2f", ph)}"
            textSize = 20f
        })

        addView(TextView(this@MainActivity).apply {
            text = "Стабільність: ${finalStabilityText.ifBlank { "не
визначено" }}"

```

```

        textSize = 14f
    })

    addView(dateEditText)
    addView(timeEditText)

    addView(TextView(this@MainActivity).apply {
        text = "Умова вимірювання:"
        textSize = 14f
        setPadding(0, 16, 0, 0)
    })

    addView(conditionSpinner)
    addView(noteEditText)
}

AlertDialog.Builder(this)
    .setTitle("Фіксація результату")
    .setView(container)
    .setPositiveButton("Зберегти") { _, _ ->
        val measurement = SavedMeasurement(
            phValue = ph,
            date = dateEditText.text.toString(),
            time = timeEditText.text.toString(),
            condition = conditionSpinner.selectedItem.toString(),
            note = noteEditText.text.toString(),
            stability = finalStabilityText
        )

        saveMeasurement(measurement)
    }
    .setNegativeButton("Скасувати", null)
    .show()
}

private fun saveMeasurement(measurement: SavedMeasurement) {
    savedMeasurements.add(measurement)
    saveMeasurementsToStorage()

    statusTextView.text = "Статус: результат збережено"

    devicesTextView.text =
        "Останній збережений результат:\n" +
        "pH: ${String.format(Locale.US, "%.2f",
measurement.phValue)}\n" +
        "Дата: ${measurement.date}\n" +
        "Час: ${measurement.time}\n" +
        "Умова: ${measurement.condition}\n" +
        "Стабільність: ${measurement.stability.ifBlank { "не
визначено" }}\n" +
        "Примітка: ${measurement.note.ifBlank { "немає" }}\n\n" +
        "Усього збережено: ${savedMeasurements.size}"

    saveResultButton.isEnabled = false
}

private fun saveMeasurementsToStorage() {
    val jsonArray = JSONArray()

    savedMeasurements.forEach { measurement ->
        val jsonObject = JSONObject()
        jsonObject.put("phValue", measurement.phValue)
    }
}

```

```

        jsonObject.put("date", measurement.date)
        jsonObject.put("time", measurement.time)
        jsonObject.put("condition", measurement.condition)
        jsonObject.put("note", measurement.note)
        jsonObject.put("stability", measurement.stability)

        jsonArray.put(jsonObject)
    }

    getSharedPreferences(preferencesName, Context.MODE_PRIVATE)
        .edit()
        .putString(measurementsKey, jsonArray.toString())
        .apply()
}

private fun loadMeasurementsFromStorage() {
    val jsonText = getSharedPreferences(preferencesName,
Context.MODE_PRIVATE)
        .getString(measurementsKey, "[]") ?: "[]"

    savedMeasurements.clear()

    try {
        val jsonArray = JSONArray(jsonText)

        for (i in 0 until jsonArray.length()) {
            val jsonObject = jsonArray.getJSONObject(i)

            val measurement = SavedMeasurement(
                pHValue = jsonObject.optDouble("pHValue", 0.0),
                date = jsonObject.optString("date", ""),
                time = jsonObject.optString("time", ""),
                condition = jsonObject.optString("condition", ""),
                note = jsonObject.optString("note", ""),
                stability = jsonObject.optString("stability", "")
            )

            savedMeasurements.add(measurement)
        }
    } catch (_: Exception) {
        savedMeasurements.clear()
    }
}

private fun showHistoryDialog() {
    if (savedMeasurements.isEmpty()) {
        AlertDialog.Builder(this)
            .setTitle("Історія вимірювань")
            .setMessage("Збережених вимірювань поки що немає.")
            .setPositiveButton("OK", null)
            .show()

        return
    }

    val historyText = StringBuilder()

    savedMeasurements.asReversed().forEachIndexed { index, measurement ->
        historyText.append("${index + 1}. pH: ")
        historyText.append(String.format(Locale.US, "%.2f",
measurement.pHValue))
        historyText.append("\n")
    }
}

```

```

        historyText.append("Дата: ${measurement.date}\n")
        historyText.append("Час: ${measurement.time}\n")
        historyText.append("Умова: ${measurement.condition}\n")
        historyText.append("Стабільність: ${measurement.stability.ifBlank {
"не визначено" }}\n")

        if (measurement.note.isNotBlank()) {
            historyText.append("Примітка: ${measurement.note}\n")
        }

        historyText.append("\n")
    }

    val textView = TextView(this).apply {
        text = historyText.toString()
        textSize = 15f
        setPadding(32, 24, 32, 24)
    }

    val scrollView = ScrollView(this).apply {
        addView(textView)
    }

    AlertDialog.Builder(this)
        .setTitle("Історія вимірювань")
        .setView(scrollView)
        .setPositiveButton("Закрити", null)
        .setNegativeButton("Очистити історію") { _, _ ->
            confirmClearHistory()
        }
        .show()
}

private fun confirmClearHistory() {
    AlertDialog.Builder(this)
        .setTitle("Очистити історію?")
        .setMessage("Усі збережені вимірювання буде видалено.")
        .setPositiveButton("Очистити") { _, _ ->
            savedMeasurements.clear()
            saveMeasurementsToStorage()

            statusTextView.text = "Статус: історію очищено"
            devicesTextView.text = "Історія вимірювань очищена."
        }
        .setNegativeButton("Скасувати", null)
        .show()
}

private fun showStatisticsPeriodDialog() {
    val options = arrayOf(
        "За весь період",
        "За останні 7 днів",
        "За останні 30 днів"
    )

    AlertDialog.Builder(this)
        .setTitle("Оберіть період статистики")
        .setItems(options) { _, which ->
            when (which) {
                0 -> showStatisticsDialog(null, "за весь період")
                1 -> showStatisticsDialog(7, "за останні 7 днів")
                2 -> showStatisticsDialog(30, "за останні 30 днів")
            }
        }
}

```

```

    }
    }
    .setNegativeButton("Скасувати", null)
    .show()
}

private fun showStatisticsDialog(periodDays: Int?, periodTitle: String) {
    val filteredMeasurements = getMeasurementsForPeriod(periodDays)

    if (filteredMeasurements.isEmpty()) {
        AlertDialog.Builder(this)
            .setTitle("Статистика")
            .setMessage("Немає збережених вимірювань $periodTitle.")
            .setPositiveButton("ОК", null)
            .show()

        return
    }

    val count = filteredMeasurements.size
    val averagePh = filteredMeasurements.map { it.phValue }.average()
    val minPh = filteredMeasurements.minOf { it.phValue }
    val maxPh = filteredMeasurements.maxOf { it.phValue }

    val stableCount = filteredMeasurements.count {
        it.stability.equals("стабільне", ignoreCase = true)
    }

    val unstableCount = filteredMeasurements.count {
        it.stability.equals("нестабільне", ignoreCase = true)
    }

    val statisticsText =
        "Період: $periodTitle\n\n" +
        "Кількість вимірювань: $count\n\n" +
        "Середнє рН: ${String.format(Locale.US, "%.2f",
averagePh)}\n\n" +
        "Мінімальне рН: ${String.format(Locale.US, "%.2f",
minPh)}\n\n" +
        "Максимальне рН: ${String.format(Locale.US, "%.2f",
maxPh)}\n\n" +
        "Стабільні результати: $stableCount\n\n" +
        "Нестабільні результати: $unstableCount"

    AlertDialog.Builder(this)
        .setTitle("Статистика")
        .setMessage(statisticsText)
        .setPositiveButton("ОК", null)
        .show()
}

private fun showGraphPeriodDialog() {
    val options = arrayOf(
        "За весь період",
        "За останні 7 днів",
        "За останні 30 днів"
    )

    AlertDialog.Builder(this)
        .setTitle("Оберіть період графіка")
        .setItems(options) { _, which ->
            when (which) {

```

```

        0 -> showGraphDialog(null, "за весь період")
        1 -> showGraphDialog(7, "за останні 7 днів")
        2 -> showGraphDialog(30, "за останні 30 днів")
    }
}
.setNegativeButton("Скасувати", null)
.show()
}

private fun showGraphDialog(periodDays: Int?, periodTitle: String) {
    val filteredMeasurements = getMeasurementsForPeriod(periodDays)

    if (filteredMeasurements.size < 2) {
        AlertDialog.Builder(this)
            .setTitle("Графік рН")
            .setMessage("Для побудови графіка потрібно щонайменше 2 збережені вимірювання $periodTitle.")
            .setPositiveButton("ОК", null)
            .show()

        return
    }

    val values = filteredMeasurements.map { it.pHValue }

    val minPh = values.minOrNull() ?: 0.0
    val maxPh = values.maxOrNull() ?: 0.0
    val averagePh = values.average()

    val summaryText =
        "Період: $periodTitle\n" +
        "Кількість вимірювань: ${values.size}\n" +
        "Середнє рН: ${String.format(Locale.US, "%.2f",
averagePh)}\n" +
        "Мінімум: ${String.format(Locale.US, "%.2f", minPh)} " +
        "Максимум: ${String.format(Locale.US, "%.2f", maxPh)}"

    val container = LinearLayout(this).apply {
        orientation = LinearLayout.VERTICAL
        setPadding(24, 16, 24, 8)

        addView(TextView(this@MainActivity).apply {
            text = summaryText
            textSize = 14f
            setPadding(0, 0, 0, 16)
        })

        addView(PhChartView(this@MainActivity, values).apply {
            layoutParams = LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.MATCH_PARENT,
                dp(320)
            )
        })

        addView(TextView(this@MainActivity).apply {
            text = "Вісь X – послідовність збережених вимірювань\nВісь Y –
значення рН"
            textSize = 12f
            setPadding(0, 12, 0, 0)
        })
    }
}

```

```

        val scrollView = ScrollView(this).apply {
            addView(container)
        }

        AlertDialog.Builder(this)
            .setTitle("Графік pH")
            .setView(scrollView)
            .setPositiveButton("Закрити", null)
            .show()
    }

    private fun showAboutDialog() {
        val message =
            "Додаток призначений для моніторингу pH слини за допомогою пристрою  
на базі Arduino.\n\n" +
            "Основні функції:\n" +
            "• підключення до вимірювального пристрою через Bluetooth  
LE;\n" +
            "• запуск вимірювання pH;\n" +
            "• усереднення отриманих значень;\n" +
            "• оцінка стабільності вимірювання;\n" +
            "• збереження результатів з датою, часом, умовами та  
приміткою;\n" +
            "• перегляд історії, статистики та графіка pH;\n" +
            "• експорт результатів у CSV-файл.\n\n" +
            "Порядок використання:\n" +
            "1. Увімкніть пристрій.\n" +
            "2. Натисніть «Підключити пристрій».\n" +
            "3. Оберіть Bluetooth LE-модуль.\n" +
            "4. Помістіть сенсор у зразок.\n" +
            "5. Натисніть «Старт вимірювання».\n" +
            "6. Після завершення зафіксуйте результат.\n\n" +
            "Увага: додаток не встановлює медичний діагноз. Він  
призначений для навчального, дослідницького та моніторингового використання."

        AlertDialog.Builder(this)
            .setTitle("Про додаток")
            .setMessage(message)
            .setPositiveButton("Зрозуміло", null)
            .show()
    }

    private fun exportMeasurementsToCsv() {
        if (savedMeasurements.isEmpty()) {
            AlertDialog.Builder(this)
                .setTitle("Експорт CSV")
                .setMessage("Немає збережених вимірювань для експорту.")
                .setPositiveButton("ОК", null)
                .show()

            return
        }

        pendingCsvText = buildCsvText()
        createCsvFileLauncher.launch("saliva_ph_measurements.csv")
    }

    private fun buildCsvText(): String {
        val csv = StringBuilder()

        csv.append('\uFEFF')
        csv.append("Дата;Час;pH;Умова вимірювання;Стабільність;Примітка\n")
    }

```

```

        savedMeasurements.forEach { measurement ->
            csv.append(csvField(measurement.date)).append(";")
            csv.append(csvField(measurement.time)).append(";")
            csv.append(String.format(Locale.US, "%.2f",
measurement.phValue)).append(";")
            csv.append(csvField(measurement.condition)).append(";")
            csv.append(csvField(measurement.stability.ifBlank { "не визначено"
}))).append(";")
            csv.append(csvField(measurement.note.ifBlank { "немає"
}))).append("\n")
        }

        return csv.toString()
    }

    private fun csvField(value: String): String {
        val escaped = value.replace("\"", "\\\"")
        return "\"$escaped\""
    }

    private fun getMeasurementsForPeriod(periodDays: Int?):
List<SavedMeasurement> {
        if (periodDays == null) {
            return savedMeasurements
        }

        val startCalendar = Calendar.getInstance().apply {
            set(Calendar.HOUR_OF_DAY, 0)
            set(Calendar.MINUTE, 0)
            set(Calendar.SECOND, 0)
            set(Calendar.MILLISECOND, 0)
            add(Calendar.DAY_OF_YEAR, -(periodDays - 1))
        }

        val startDate = startCalendar.time

        return savedMeasurements.filter { measurement ->
            val measurementDate = parseMeasurementDate(measurement.date)
            measurementDate != null && !measurementDate.before(startDate)
        }

    }

    private fun parseMeasurementDate(dateText: String): Date? {
        return try {
            val dateFormat = SimpleDateFormat("dd.MM.yyyy", Locale.getDefault())
            dateFormat.isLenient = false
            dateFormat.parse(dateText)
        } catch (_: Exception) {
            null
        }
    }

    private fun dp(value: Int): Int {
        return (value * resources.displayMetrics.density).toInt()
    }

    @SuppressLint("MissingPermission")
    private fun closeGatt() {
        try {
            bluetoothGatt?.disconnect()
            bluetoothGatt?.close()
        }
    }

```

```

    } catch (_: Exception) {
    }

    bluetoothGatt = null
    notifyCharacteristic = null
    writeCharacteristic = null
    isConnected = false
}

override fun onDestroy() {
    super.onDestroy()

    stopBleScan()
    sendCommandToDevice("STOP\n")
    closeGatt()
}

private class PhChartView(
    context: Context,
    private val values: List<Double>
) : View(context) {

    private val axisPaint = Paint(Paint.ANTI_ALIAS_FLAG).apply {
        color = Color.DKGRAY
        strokeWidth = 3f
        style = Paint.Style.STROKE
    }

    private val gridPaint = Paint(Paint.ANTI_ALIAS_FLAG).apply {
        color = Color.LTGRAY
        strokeWidth = 1.5f
        style = Paint.Style.STROKE
    }

    private val linePaint = Paint(Paint.ANTI_ALIAS_FLAG).apply {
        color = Color.rgb(0, 120, 160)
        strokeWidth = 5f
        style = Paint.Style.STROKE
    }

    private val pointPaint = Paint(Paint.ANTI_ALIAS_FLAG).apply {
        color = Color.rgb(0, 120, 160)
        style = Paint.Style.FILL
    }

    private val textPaint = Paint(Paint.ANTI_ALIAS_FLAG).apply {
        color = Color.DKGRAY
        textSize = 28f
        style = Paint.Style.FILL
    }

    override fun onDraw(canvas: Canvas) {
        super.onDraw(canvas)

        if (values.size < 2) {
            canvas.drawText("Недостатньо даних для графіка", 40f, height /
2f, textPaint)
            return
        }

        val leftPadding = 80f
        val rightPadding = 30f

```

```

val topPadding = 40f
val bottomPadding = 70f

val chartLeft = leftPadding
val chartRight = width - rightPadding
val chartTop = topPadding
val chartBottom = height - bottomPadding

val chartWidth = chartRight - chartLeft
val chartHeight = chartBottom - chartTop

if (chartWidth <= 0 || chartHeight <= 0) {
    return
}

val rawMin = values.minOrNull() ?: 0.0
val rawMax = values.maxOrNull() ?: 14.0

var minY = rawMin - 0.2
var maxY = rawMax + 0.2

minY = max(0.0, minY)
maxY = min(14.0, maxY)

if (abs(maxY - minY) < 0.5) {
    minY = max(0.0, rawMin - 0.5)
    maxY = min(14.0, rawMax + 0.5)
}

val yRange = maxY - minY

val gridLines = 4

for (i in 0..gridLines) {
    val ratio = i.toFloat() / gridLines
    val y = chartBottom - ratio * chartHeight
    val labelValue = minY + ratio * yRange

    canvas.drawLine(chartLeft, y, chartRight, y, gridPaint)
    canvas.drawText(
        String.format(Locale.US, "%.1f", labelValue),
        8f,
        y + 10f,
        textPaint
    )
}

axisPaint)
canvas.drawLine(chartLeft, chartTop, chartLeft, chartBottom,
axisPaint)
canvas.drawLine(chartLeft, chartBottom, chartRight, chartBottom,

val path = Path()

values.forEachIndexed { index, value ->
    val xRatio = index.toFloat() / (values.size - 1).toFloat()
    val yRatio = ((value - minY) / yRange).toFloat()

    val x = chartLeft + xRatio * chartWidth
    val y = chartBottom - yRatio * chartHeight

    if (index == 0) {

```

```
        path.moveTo(x, y)
    } else {
        path.lineTo(x, y)
    }

    canvas.drawCircle(x, y, 7f, pointPaint)
}

canvas.drawPath(path, linePaint)

canvas.drawText("1", chartLeft - 5f, chartBottom + 42f, textPaint)
canvas.drawText(
    values.size.toString(),
    chartRight - 20f,
    chartBottom + 42f,
    textPaint
)

canvas.drawText("pH", 8f, chartTop - 10f, textPaint)
}
}
```