Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

# КВАЛІФІКАЦІЙНА РОБОТА

Програмно-технічний комплекс моніторингу якості водопровідної води в режимі реального часу
Назва теми

Рівень вищої освіти *перший (бакалаврський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ. 022043.22.01.100 ПЗ

Виконав здобувач IV курсу, група КІ2-22-1 Підпис Тарас КРАВЧУК Ініціали, прізвище

Керівник Науковий ступінь, учене звання Підпис В'ячеслав АСКЕРОВ Ініціали, прізвище

Нормоконтролер канд.фіз.-мат.наук, доц. Науковий ступінь, учене звання Підпис Тетяна КИСІЛЬ Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС Підпис Ольга ПАВЛОВА Ініціали, прізвище
«09» червня 2026 р.
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Рівень вищої освіти перший (бакалаврський)

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІІС
_____ Ольга ПАВЛОВА
" 10 " 01 2026 р.

**ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Кравчуку Тарасу Сергійовичу
Прізвище, ім'я, по батькові студента

1. Тема проєкту (роботи) Прогрмано-технічний комплекс моніторингу якості водопровідної води в режимі реального часу
Керівник проекту (роботи) Аскеров В'ячеслав Васильович
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7

2. Термін подання здобувачем роботи на кафедру 01.06.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка задачі.
Проєктування програмно-технічного комплексу.
Програмно-апаратна реалізація та тестування програмно-технічного засобу

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Архітектура програмно-технічного комплексу
Алгоритм роботи ПЗ ESP32
Апаратне забезпечення проєкту

6. Консультанти розділів кваліфікаційної роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |

7. Дата видачі завдання «_10_»_____01_____2026 р.

## КАЛЕНДАРНИЙ ПЛАН

| №з/п | Назва етапів (розділів) дипломного проекту (роботи) | Термін виконання етапів проекту (роботи) | Примітка |
|---|---|---|---|
| 1 | Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником | 10.01.2026 | виконано |
| 2 | Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження | 01.02.2026 | виконано |
| 3 | Робота над розділом 1 – дослідження предметної області та постановка задачі | 01.03.2026 | виконано |
| 4 | Робота над розділом 2 – проєктування програмно-технічного комплексу | 01.04.2026 | виконано |
| 5 | Робота над розділом 3 – програмно-апаратна реалізація та тестування програмно-технічного засобу | 29.04.2026 | виконано |
| 6 | Оформлення пояснювальної записки згідно вимог | 25.05.2026 | виконано |
| 7 | Попередній захист ВКР | 26.05.2026 | виконано |
| 8 | Захист ВКР на засіданні ЕК | Червень 2026 року | |

Здобувач _____ Тарас КРАВЧУК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ В'ячеслав АСКЕРОВ
Підпис Ім'я, ПРІЗВИЩЕ

| № рядка | формат | Позначення | Найменування | Кіл. листів | № екз | Примітка |
|---|---|---|---|---|---|---|
| | | | Текстові документи | | | |
| 1 | | КвРКІ. 022043.22.01.100 ПЗ | Пояснювальна записка | 75 | | |
| | | | | | | |
| | | | Графічні матеріали | | | |
| 2 | | КвРКІ. 022043.22.01.100 Е8 | Архітектура програмно- | 1 | | |
| | | | технічного комплексу | | | |
| 3 | | КвРКІ. 022043.22.01.100 Е8 | Алгоритм роботи ПЗ | 1 | | |
| | | | ESP32 | | | |
| 4 | | КвРКІ. 022043.22.01.100 Е8 | Апаратне забезпечення | 1 | | |
| | | | проєкту | | | |

| Зм | Арк | № докум | Підпис | Дата | КвРКІ. 022043.22.01.100 ВП | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Кравчук | | 01.06 | Відомість проекту | Літера | Аркуш | Аркушів |
| Перевір. | | Аскеров | | | | У | 1 | 1 |
| Н. контр. | | Кисіль | | | | ХНУ, КІ2-22-1 | | |
| Затв. | | Павлова | | 01.06 | | | | |

# АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмно-технічний комплекс моніторингу якості водопровідної води в режимі реального часу».

Автор роботи: Тарас КРАВЧУК.

Керівник роботи: В'ячеслав АСКЕРОВ.

Пояснювальна записка: 75 с., 27 рис., 3 табл., 3 дод., 45 джерел.

Графічна частина: 3 креслення.

АВТОМАТИЗАЦІЯ, ДАТЧИК, ESP32, FIREBASE, ІНТЕРНЕТ РЕЧЕЙ, МОНІТОРИНГ, ЯКІСТЬ ВОДИ.

Метою роботи є розробка програмно-технічного комплексу для безперервного моніторингу якості водопровідної води. Апаратна частина системи побудована на базі мікроконтролера ESP32 та датчиків pH, загальної мінералізації (TDS) і температури. Для обробки телеметричних даних у реальному часі використано хмарну платформу Firebase. Розроблено веб-панель та мобільний додаток, які забезпечують візуалізацію статистики та оперативне сповіщення користувачів про відхилення показників від встановлених норм.

У результаті створено діючий прототип із високою стабільністю передачі даних. Практична значимість розробки полягає у забезпеченні ефективного побутового контролю якості водопостачання, що сприяє підвищенню безпеки споживання води. Перспективи розвитку об'єкта передбачають інтеграцію додаткових сенсорів (наприклад, рівня хлору чи тиску) та розширення функціоналу аналітичного модуля для прогнозування обслуговування фільтраційних систем.

Підпис здобувача — 30.05.2026 Дата

## ЗМІСТ

# ВСТУП

Забезпечення належної якості питної води є фундаментальним чинником підтримки здоров'я населення та забезпечення екологічної безпеки. В умовах сучасної урбанізації та фізичного зношення інфраструктури водопровідних мереж споживачі часто стикаються з проблемою невідповідності води санітарним нормам. Традиційні методи лабораторного контролю якості води, хоч і вирізняються високою точністю, є фінансово затратними та не дозволяють здійснювати безперервний моніторинг у режимі реального часу. Швидкий розвиток технологій Інтернету речей (IoT) відкриває принципово нові можливості для створення доступних, автоматизованих та масштабованих систем контролю, що зумовлює високу актуальність розробки програмно-технічних комплексів моніторингу якості води на базі сучасних мікроконтролерів.

Метою цієї кваліфікаційної роботи є розробка та практична реалізація програмно-технічного комплексу, призначеного для збору, обробки, зберігання та візуалізації даних про фізико-хімічні показники якості водопровідної води, зокрема рівня pH, загальної мінералізації TDS та температури, з можливістю дистанційного сповіщення користувача про критичні зміни стану системи.

Для досягнення поставленої мети в роботі було вирішено низку взаємопов'язаних завдань, що охоплюють весь цикл розробки, починаючи від аналізу предметної області та існуючих рішень у сфері IoT-моніторингу, і закінчуючи практичним впровадженням системи. Важливим етапом стало обґрунтування вибору апаратної платформи ESP32, яка завдяки вбудованим модулям Wi-Fi, низькому енергоспоживанню та широким можливостям периферії є оптимальною для завдань збору телеметричних даних. Проєктування архітектури системи було виконано з використанням методу віртуального прототипування у середовищі Wokwi. Застосування цього інструменту дозволило провести повну верифікацію схемотехнічних рішень, відлагодити

логіку взаємодії датчиків із мікроконтролером та усунути потенційні програмні помилки ще до етапу фізичної збірки пристрою, що значно скоротило час на розробку.

У межах роботи розроблено алгоритми збору та цифрової фільтрації даних, налаштовано програмну інфраструктуру на базі хмарної платформи Firebase. Вибір Firebase зумовлений її здатністю забезпечувати синхронізацію даних у реальному часі (Realtime Database), що є критично важливим для оперативного відображення стану води. На основі хмарної бази даних створено кросплатформовий інтерфейс користувача у вигляді веб-панелі та мобільного додатку, що забезпечує зручну візуалізацію статистики та інтуїтивно зрозумілу систему сповіщень.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі наведено аналітичний огляд існуючих систем моніторингу та обґрунтовано вибір засобів реалізації. Другий розділ присвячено апаратному проектуванню та віртуальному моделюванню системи. У третьому розділі описано програмну архітектуру та реалізацію хмарного серверного середовища. Четвертий розділ містить результати експериментального тестування розробленого прототипу.

Об’єктом дослідження є процес автоматизованого моніторингу параметрів якості водопровідної води, а предметом дослідження виступають апаратно-програмні засоби збору, передачі та обробки даних на базі мікроконтролера ESP32 та хмарних технологій. Методологічну основу роботи склали методи системного аналізу для вибору архітектурних рішень, метод віртуального прототипування для проектування електричних схем, методи об’єктно-орієнтованого програмування для розробки клієнт-серверної частини системи, а також методи експериментального тестування для верифікації точності отриманих результатів.

Наукова новизна роботи полягає у запропонованій архітектурі інтегрованого комплексу, що гармонійно поєднує низьковартісну апаратну базу

на базі ESP32 з гнучкими хмарними сервісами Firebase, забезпечуючи надійність моніторингу при мінімальних витратах на впровадження. Практичне значення результатів роботи підтверджується створенням діючого прототипу, який може бути ефективно використаний у побутових системах водопостачання для автоматичного контролю якості води та завчасного виявлення відхилень від нормативних значень, при цьому розроблені програмні модулі мають значний потенціал до адаптації для ширших завдань розумного будинку.

# 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Забезпечення безпеки та якості питної води є фундаментальним завданням сучасних систем життєзабезпечення урбанізованих територій. В умовах стрімкого розвитку міської інфраструктури та фізичного старіння водопровідних мереж виникає гостра потреба у впровадженні ефективних засобів контролю фізико-хімічних параметрів води. Як свідчить аналіз техногенних аварій останнього десятиліття, більшість інцидентів пов’язана саме із вторинним забрудненням води в розподільчих мережах, а не з недоліками підготовки води на станціях. Основними чинниками, що впливають на якість води у кінцевих точках споживання, є корозійні процеси в трубопроводах, наявність відкладень, зміна концентрації солей, коливання pH, а також неконтрольовані зміни хімічного складу джерел водопостачання внаслідок сезонних або техногенних факторів.

Сучасні інформаційні процеси у цій сфері вимагають переходу від періодичного лабораторного контролю, що дає дискретні зрізи даних з латентністю до декількох діб, до постійного моніторингу в режимі реального часу. Такий перехід дозволяє мінімізувати ризики споживання неякісної води, оперативно реагувати на аварійні ситуації (прориви, гідроудари, залпові скиди) та формувати обґрунтовані графіки профілактичних робіт. Крім того, безперервний збір даних утворює підґрунтя для побудови прогнозних моделей змін якості води за показниками pH, загальної мінералізації (TDS) та температури, що особливо важливо для систем раннього попередження [1].

Важливим технічним аспектом функціонування систем моніторингу є забезпечення високої метрологічної надійності даних, які отримуються від сенсорних модулів. Електрохімічні (pH-електроди), кондуктометричні (датчики

TDS) та терморезистивні датчики, які використовуються для аналізу води, є надзвичайно чутливими до впливу зовнішніх факторів[24, 25, 26]. Серед основних завад слід виділити температурну залежність вимірювань, зміну гідростатичного тиску, дрейф нуля через старіння електродів, а також вплив мікрозабруднень (органічних плівок) на поверхні чутливих елементів. Це створює об’єктивну необхідність розробки спеціалізованих програмних алгоритмів для автоматичної температурної компенсації показників [14], цифрової фільтрації шумів (наприклад, ковзним середнім або медіанним фільтром) у аналогових ланцюгах, а також алгоритмів калібрування «на льоту». Без належної апаратно-програмної обробки сигнали втрачають свою достовірність, що робить неможливим використання отриманих даних для прийняття об'єктивних управлінських рішень у системах автоматизації.

Цифровізація сектору водопостачання, яка є невід'ємною складовою концепції «інтелектуальних міст» (Smart Cities), вимагає не лише фізичного вимірювання параметрів, але й їхньої оперативної передачі, централізованої систематизації, валідації та візуалізації. Відсутність єдиної інформаційної платформи для моніторингу якості води на побутовому рівні значно обмежує доступ споживачів до актуальної інформації про стан водопровідної мережі. Більшість існуючих рішень (наприклад, промислові аналізатори) залишаються закритими системами, де отримані дані не інтегруються з сучасними мобільними (iOS/Android) або вебзастосунками. Наслідком є позбавлення кінцевого користувача можливості віддаленого контролю, побудови історії вимірювань та отримання push-сповіщень про критичні зміни складу води, що може бути критичним для людей із хронічними захворюваннями або родин із малими дітьми.

Таким чином, проблема полягає у відсутності доступних, масштабованих та інтегрованих програмно-технічних інструментів для автоматизованого збору, обробки й представлення даних про якість води саме в побутовому секторі. Традиційні методи опрацювання інформації про стан води, що ґрунтуються на

ручному відборі проб та їх подальшому лабораторному аналізі, мають суттєві обмеження щодо:

– оперативності (результат доступний лише через години/дні).

– репрезентативності (одиничний зріз не відображає динаміки).

– вартості (лабораторні аналізи потребують кваліфікованого персоналу та реактивів).

Автоматизація цього процесу за допомогою сучасних мікроконтролерних платформ (наприклад, Arduino або ESP32) та хмарних сервісів IoT-платформ дозволяє не лише забезпечити безперервний контроль, але й створити стійку базу даних для побудови прогнозних моделей та систем підтримки прийняття рішень. Вирішення цієї задачі лежить у площині інтеграції трьох компонентів: сенсорного вузла, електрохімічні та кондуктометричні датчики, мікроконтролерної системи збору та попередньої обробки: фільтрація, компенсація та хмарної аналітичної платформи: збереження, візуалізація, сповіщення. Саме така трирівнева архітектура відкриває нові можливості для створення інтелектуальних систем сповіщення та управління якістю води без безпосереднього втручання людини, реалізуючи принципи Індустрії 4.0 у житлово-комунальному господарстві.

## 1.2 Аналіз наявного програмно-апаратного забезпечення предметної області

Сучасний ринок засобів моніторингу якості води характеризується інтенсивним розвитком та поступовим переходом від класичних лабораторних методів до систем автоматизованого збору даних. Дослідження досвіду провідних виробників та розробників дозволяє класифікувати існуючі рішення на три основні категорії, кожна з яких має свою нішу застосування, технічні обмеження та економічні особливості.

Перша група представлена промисловими аналізаторами, які традиційно використовуються на великих очисних спорудах, об'єктах централізованого водопостачання та промислових підприємствах. Такі системи базуються на складних архітектурах АСУ ТП, виділяються високою метрологічною точністю та надійністю, а також часто підлягають обов'язковій державній сертифікації. Проте, їх впровадження у приватних або малих системах є економічно недоцільним через надзвичайно високу вартість обладнання, складність монтажу та необхідність підключення до спеціалізованих індустріальних мереж.

Друга група охоплює портативні побутові прилади, такі як TDS-метри та pH-метри, які набули широкої популярності завдяки доступності. Їхньою головною перевагою є мобільність та миттєве отримання результату, що робить їх ідеальними для швидкої перевірки води в побутових умовах. Водночас вони мають недолік – відсутність архівації даних та можливості дистанційного моніторингу. Користувач змушений вручну фіксувати результати, що унеможливлює проведення довгострокового аналізу якості води та виявлення прихованих тенденцій погіршення показників.

Третя група – це сучасні IoT-рішення, що базуються на використанні мікроконтролерів ESP32 [2] та інтеграції з хмарними сервісами, зокрема Firebase[4]. Це найдинамічніший сегмент, який дозволяє перетворити звичайні датчики на інтелектуальні системи з дистанційним доступом. Типову архітектуру таких систем наведено на рисунку 1.1. Такі рішення забезпечують безперервний збір даних, миттєву передачу вимірювань до хмарних сховищ та візуалізацію показників через веб-інтерфейси або мобільні застосунки. Користувач отримує можливість віддаленого контролю за станом води з будь-якого місця, де є доступ до інтернету, а також автоматичне сповіщення про критичні відхилення [5]. Крім того, накопичення історичних даних у хмарі створює передумови для побудови прогнозних моделей та аналізу довгострокових трендів зміни якості води [16]. Це дозволяє не лише оперативно реагувати на перевищення гранично допустимих концентрацій, але й виявляти

приховані закономірності погіршення якості води, пов'язані, наприклад, із сезонними змінами або зносом водопровідних мереж.

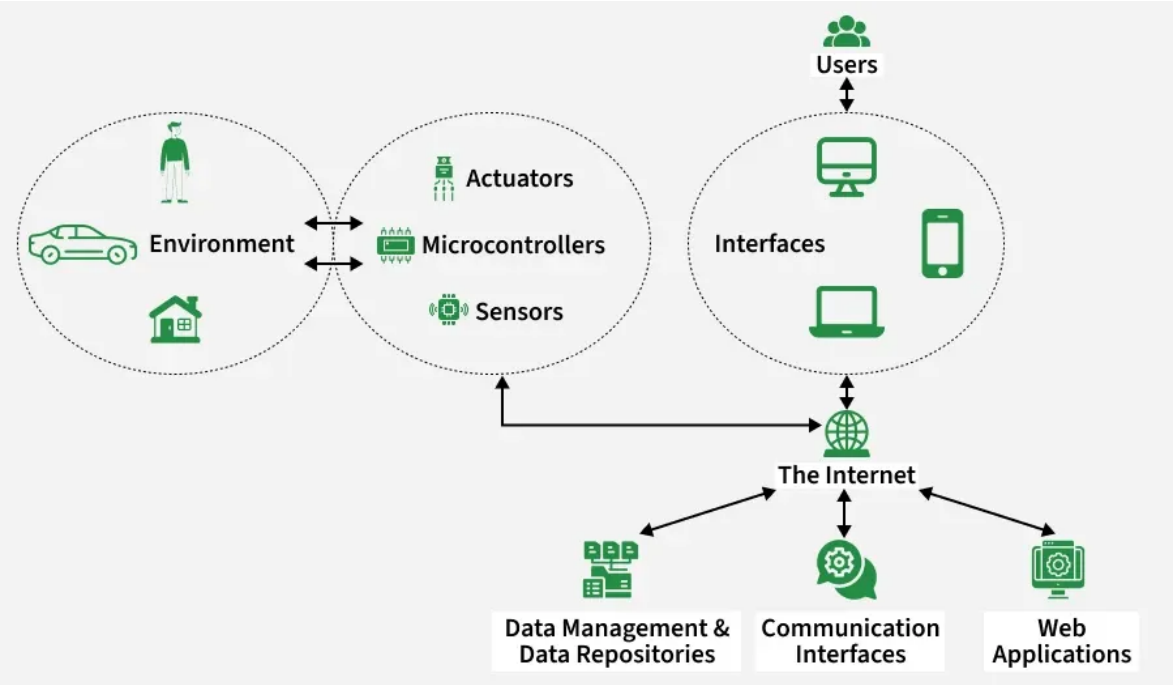



Рисунок 1.1 – Архітектура IoT системи

Проте навіть комерційні аналоги в цьому сегменті часто мають недоліки, такі як висока вартість через «бренд» або залежність від закритих екосистем виробника.

Порівняльний аналіз зазначених рішень наведено у таблиці 1.1, яка дозволяє оцінити доцільність впровадження розроблюваного комплексу. Порівняльний аналіз зазначених рішень наведено у таблиці 1.1, яка дозволяє оцінити доцільність впровадження розроблюваного комплексу. Як видно з таблиці, більшість існуючих систем або орієнтовані на промислове застосування з високою вартістю обладнання, або є закритими рішеннями з обмеженими можливостями для адаптації під конкретні потреби користувача.

Таблиця 1.1 – Порівняльний аналіз існуючих засобів моніторингу якості води

| Тип системи | Особливості | Переваги | Недоліки |
|---|---|---|---|
| Промислові системи | Висока точність, складна архітектура АСУ ТП | Надійність, сертифікація за стандартами | Висока ціна, складність впровадження для дому |
| Портативні прилади | Автономні датчики без зв'язку | Низька ціна, мобільність | Відсутність архівації даних, ручне керування |
| IoT-рішення (аналоги) | Програмно-апаратні комплекси на базі МК | Дистанційний доступ, автоматизація | Потреба у калібруванні, залежність від Wi-Fi |
| Розроблюваний комплекс | IoT-система на ESP32 з хмарним інтерфейсом | Оптимальна ціна, Real-time моніторинг | Потребує налаштування інфраструктури |

Аналізуючи наведені дані, можна дійти висновку, що жодна з існуючих груп не забезпечує ідеального балансу між функціональністю та доступністю для широкого кола користувачів. Промислові системи є надмірно складними, портативні прилади – занадто примітивними, а існуючі комерційні IoT-аналоги часто нав’язують користувачеві надлишкову залежність від пропрієтарного програмного забезпечення.

Саме тому розроблюваний програмно-апаратний комплекс на базі мікроконтролера ESP32 має суттєву перевагу. Використання відкритої архітектури дозволяє реалізувати гнучкий моніторинг у режимі реального часу при мінімальних фінансових витратах на обладнання. Ключовою відмінністю запропонованого підходу є можливість повної адаптації хмарного інтерфейсу Firebase під конкретні потреби користувача, що забезпечує не тільки автоматизацію збору даних, а й повну прозорість алгоритмів обробки телеметрії. Таким чином, розроблюваний проект заповнює технологічний розрив між дорогими промисловими рішеннями та функціонально обмеженими побутовими засобами, пропонуючи оптимальний інструментарій для моніторингу якості води.

## 1.3 Визначення вимог до системи автоматизації та розробка технічного завдання

Формування технічних вимог до системи автоматизації моніторингу якості води є критично важливим етапом проектування, оскільки саме на цьому рівні визначаються функціональні межі комплексу та способи взаємодії користувача з програмно-апаратним середовищем. Головною метою проекту є створення інтелектуальної автоматизованої системи, здатної забезпечувати безперервний збір, опрацювання, архівування та візуалізацію телеметричних даних з високим рівнем достовірності. Аналіз предметної області дозволив декомпозувати загальні завдання системи на конкретні функціональні блоки, структуру яких відображено на карті функціональних можливостей, представленій на рисунку 1.2.

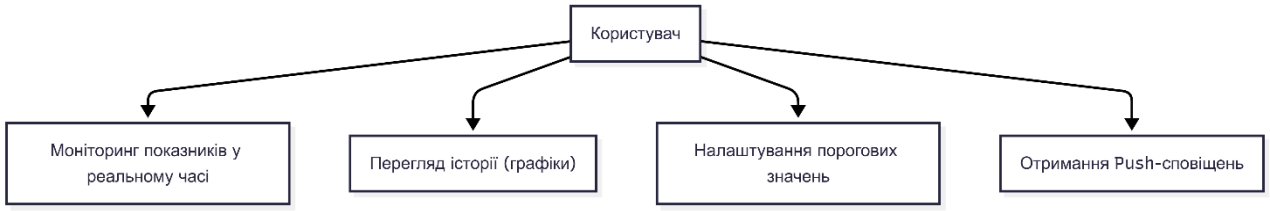



Рисунок 1.2 – Функціональна структура програмно-технічного комплексу

Згідно з цією схемою, першим фундаментальним функціональним напрямком роботи системи є безперервний моніторинг показників якості води у реальному часі. Реалізація цієї вимоги передбачає інтеграцію сенсорів для вимірювання рівня pH, концентрації розчинених речовин TDS та температури середовища. Критично важливим аспектом тут є програмна реалізація алгоритмів цифрової фільтрації сигналів та температурної компенсації, що дозволяє нівелювати похибки вимірювань, спричинені тепловими коливаннями середовища, та гарантувати відповідність результатів реальним фізичним показникам.

Паралельно з процесом збору даних, система повинна забезпечувати повноцінну візуалізацію та перегляд історії показників. Для цього програмне забезпечення реалізує механізм архівування телеметрії у хмарній базі даних Firebase. Вимогою до цього модуля є можливість відображення динаміки змін параметрів за довільні часові проміжки у вигляді інтерактивних графіків, що дозволяє користувачеві самостійно проводити аналітичну оцінку стану води, виявляти довгострокові тренди забруднення або відхилень від нормативних значень.

Окремим функціональним блоком виступає механізм налаштування порогових значень, який забезпечує гнучкість системи та її адаптивність до різних умов використання. Користувач отримує інструментарій для встановлення граничних меж для кожного окремого показника, що дозволяє системі самостійно ідентифікувати критичні стани середовища. Цей функціонал тісно інтегрований з підсистемою автоматичної генерації сповіщень. У разі виходу будь-якого з параметрів за встановлені межі, система автоматично ініціює відправку push-повідомлень або оновлює індикацію в інтерфейсі користувача, забезпечуючи оперативне реагування на позаштатні ситуації та запобігаючи небезпечним наслідкам.

Для моделювання взаємодії користувача з розроблюваним програмно-апаратним комплексом та формалізації сценаріїв використання було побудовано

діаграму прецедентів (UML Use Case), яку наведено на рисунку 1.3. Вона відображає основні сценарії взаємодії користувача із системою, включаючи моніторинг параметрів, роботу алгоритмів компенсації, аналіз історичних даних та налаштування оповіщень.

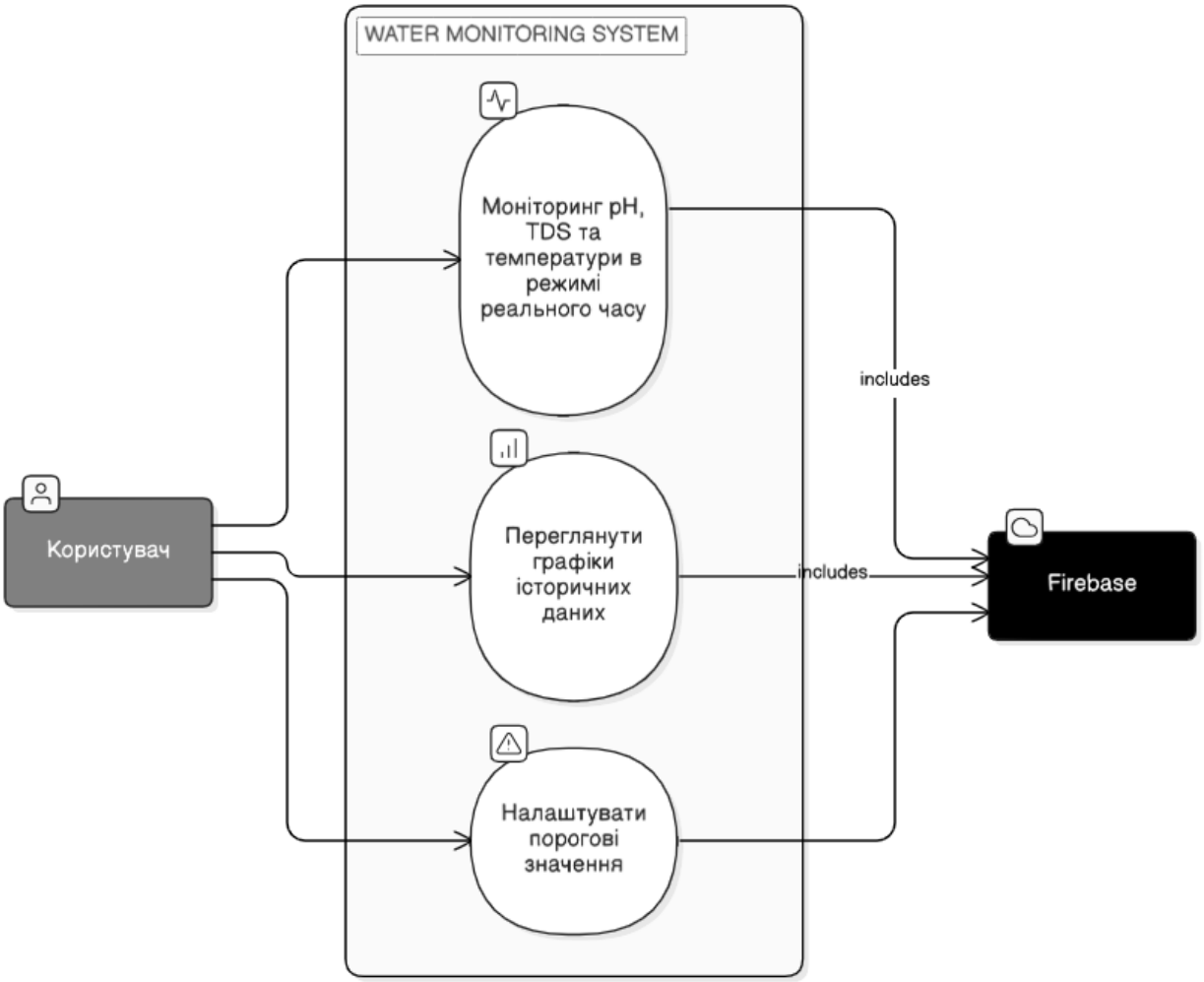



Рисунок 1.3 – Діаграма прецедентів системи моніторингу

Окрім функціональних вимог, до системи висуваються суворі нефункціональні вимоги, які визначають експлуатаційну надійність комплексу. Враховуючи специфіку роботи IoT-пристроїв, архітектура повинна володіти високою стійкістю до розривів мережевого з'єднання, що досягається шляхом впровадження механізмів локальної буферизації даних та їх подальшої

синхронізації з хмарою після відновлення стабільного доступу до інтернету. Важливою нефункціональною вимогою також є енергоефективність програмного коду, яка дозволяє оптимізувати цикли роботи мікроконтролера ESP32 та підтримувати автономність системи при живленні від обмежених джерел енергії. Масштабованість системи забезпечується модульною структурою програмного забезпечення, що дозволяє у майбутньому інтегрувати додаткові типи сенсорів або підключати комплекс до розгалужених екосистем розумного будинку без необхідності радикальних змін в базовій архітектурі.

Також, в контексті розробки системи шляхом віртуального прототипування, висуваються особливі вимоги до програмної частини, яка повинна бути повною мірою адаптована до особливостей емуляції периферійних інтерфейсів у середовищі Wokwi. Це гарантує ідентичність логіки роботи віртуальної моделі порівняно з фізичними зразками, що є запорукою успішного подальшого впровадження. Деталізований виклад усіх технічних параметрів, протоколів обміну даними, специфікацій інтерфейсів та програмних модулів зафіксовано у документі «Технічне завдання», який оформлено відповідно до державних стандартів та розміщено у відповідних додатках до цієї дипломної роботи для забезпечення повноти технічної документації проекту.

## 1.4 Висновки до розділу 1

У першому розділі було здійснено комплексний аналіз предметної області, який підтвердив критичну необхідність переходу від періодичного лабораторного контролю до безперервного автоматизованого моніторингу якості води. Дослідження ринку програмно-апаратних засобів виявило суттєвий технологічний розрив: промислові аналізатори мають надлишкову вартість та складність для індивідуального користувача, тоді як побутові прилади не забезпечують можливості архівації даних та дистанційного оповіщення.

За результатами проведеного порівняльного аналізу обґрунтовано вибір мікроконтролера ESP32 як оптимальної платформи, що забезпечує необхідний баланс між продуктивністю, енергоефективністю та наявністю вбудованих засобів бездротового зв’язку. Такий вибір створює передумови для реалізації гнучкого віртуального прототипування, що дозволяє мінімізувати часові та фінансові витрати на етапі відладки.

На основі детального вивчення фізико-хімічних принципів роботи сенсорів pH, TDS та температури, було сформовано перелік функціональних вимог, які є основою для забезпечення достовірності телеметричних даних. Побудова діаграми прецедентів дозволила формалізувати сценарії взаємодії користувача з системою, чітко розмежувавши функціональні межі програмного забезпечення та зовнішніх хмарних сервісів (Firebase).

Таким чином, розв'язана задача декомпозиції вимог та визначення постановки завдання створює надійний теоретико-методичний фундамент для наступного етапу роботи. Отримані результати дозволяють перейти до безпосереднього проектування програмно-апаратної архітектури, розробки алгоритмів обробки сигналів та створення віртуальної моделі системи, яка відповідатиме сучасним вимогам до IoT-комплексів.

# 2 ПРОЄКТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО КОМПЛЕКСУ

## 2.1. Вибір та обґрунтування архітектури програмно-технічного комплексу

На етапі проєктування програмно-технічного комплексу моніторингу якості водопровідної води першочерговим завданням є визначення загальної архітектури системи, яка б забезпечувала виконання всіх функціональних вимог, сформульованих у технічному завданні. Архітектура ПТК визначає склад основних підсистем, характер їх взаємодії, способи обміну даними та розподіл обчислювального навантаження. Для систем моніторингу та збору даних у реальному часі традиційно використовуються три основні типи архітектур: монолітна, локальна клієнт-серверна та хмарно-орієнтована.

Монолітна архітектура передбачає, що всі компоненти системи – збір даних, обробка, зберігання та інтерфейс користувача - виконуються на одному пристрої. У контексті задачі моніторингу якості води це означає, що мікроконтролер ESP32 не лише зчитує показники з датчиків, але й зберігає історію вимірювань на підключеній SD-карті, а також відображає дані на вбудованому дисплеї або через веб-сервер, запущений безпосередньо на ESP32 [2]. Такий підхід є простим у реалізації, не потребує додаткових серверів та мінімізує затримки, оскільки всі операції виконуються локально. Проте, як видно з прикладу реалізації монолітної архітектури, наведеного на рисунку 2.1, вона має суттєві обмеження: неможливість віддаленого доступу до даних за межами локальної мережі, обмежений обсяг пам'яті для зберігання історії вимірювань (об'єм SD-карти або flash-пам'яті ESP32), а також складність масштабування та оновлення програмного забезпечення. Крім того, веб-інтерфейс, запущений на ESP32, є доволі обмеженим за функціональністю через невеликий обсяг оперативної пам'яті пристрою. З огляду на зазначені обмеження, монолітна архітектура є непридатною для створення сучасних систем моніторингу, що потребують цілодобового віддаленого доступу та накопичення великих масивів історичних даних. Саме тому в даній роботі вона розглядається лише як базовий

орієнтир для порівняння, тоді як основний акцент зроблено на більш гнучких та масштабованих архітектурних рішеннях.

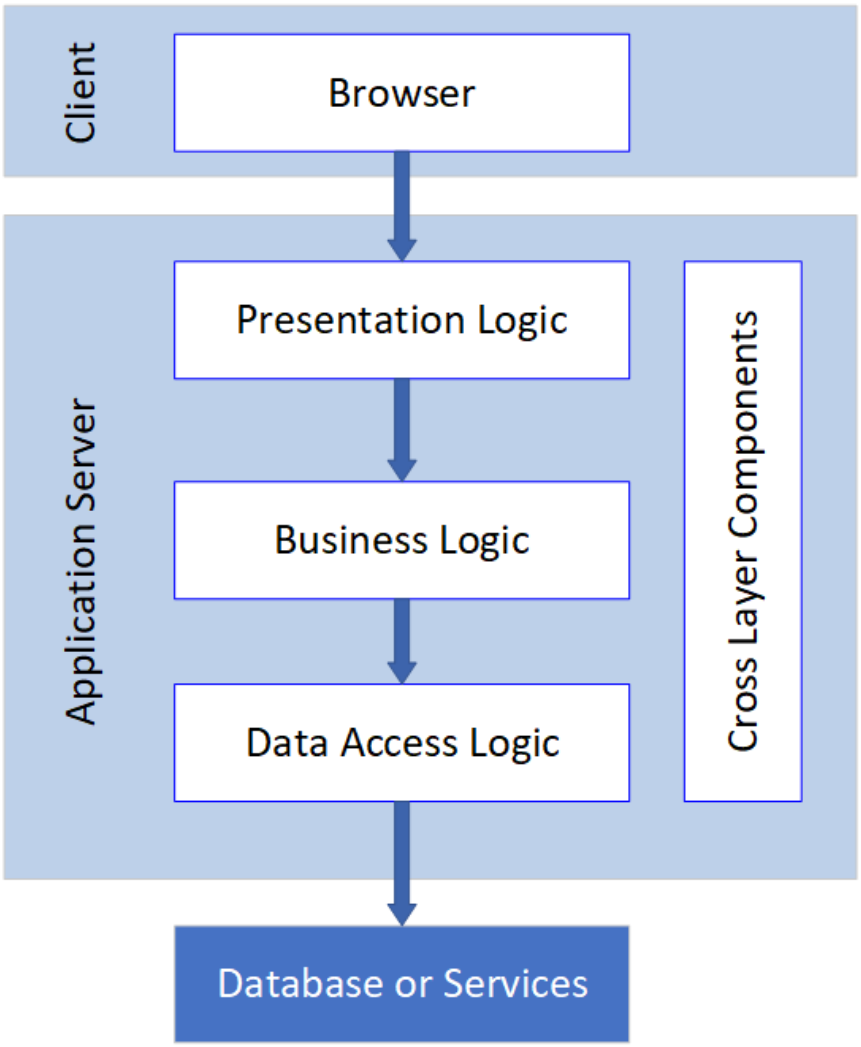



Рисунок 2.1 – Приклад монолітної архітектури

Локальна клієнт-серверна архітектура працює за наступним принципом: дані з датчиків, підключених до ESP32, надсилаються через локальну мережу на центральний сервер, який може бути реалізований на одноплатному комп'ютері Raspberry Pi або на звичайному персональному комп'ютері. Сервер виконує функції зберігання даних у базі даних (наприклад, MySQL або SQLite)[33], обробки запитів від клієнтів та генерування звітів. Користувачі отримують доступ до системи через веб-інтерфейс, який також розгорнуто на цьому сервері.

Приклад реалізації такої архітектури наведено на рисунку 2.2. Локальна клієнт-серверна архітектура забезпечує кращу продуктивність порівняно з монолітною, дозволяє зберігати практично необмежену історію вимірювань (обмежену лише об'ємом дискового простору сервера) та надає зручний веб-інтерфейс з повноцінною графікою. Однак вона вимагає постійно увімкненого серверного обладнання, налаштування мережевої безпеки (порти, брандмауери), а також не забезпечує зручного доступу до даних поза межами локальної мережі без додаткових налаштувань, таких як VPN або проброс портів. Адміністрування власного сервера потребує додаткових знань та часу, що не завжди є доцільним у межах дипломного проєкту.

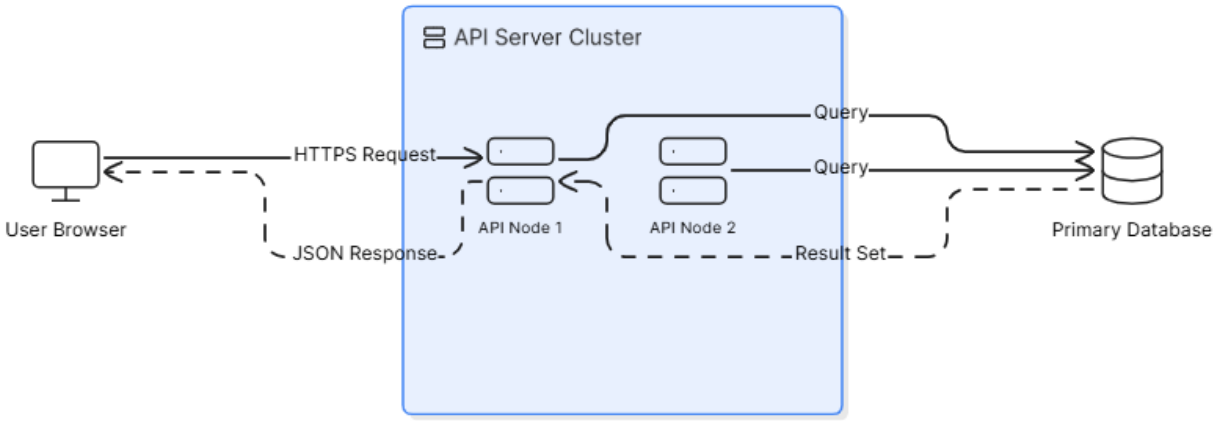



Рисунок 2.2 – Приклад локальної клієнт-серверної архітектури

Хмарно-орієнтована архітектура, яка також відома як Backend-as-a-Service (BaaS), працює наступним чином: пристрій збору даних (ESP32) передає вимірювання через інтернет до хмарної платформи, яка бере на себе всі серверні функції – зберігання даних, забезпечення безпеки, автентифікацію користувачів та синхронізацію в реальному часі. Клієнтські застосунки – веб-панель та мобільний додаток – підключаються безпосередньо до цієї ж хмарної платформи, отримуючи оновлення миттєво завдяки механізму WebSockets. Приклад реалізації такої архітектури наведено на рисунку 2.3. Цей підхід усуває

необхідність утримання власного сервера, забезпечує глобальну доступність даних з будь-якої точки світу, де є доступ до інтернету, та надає автоматичне масштабування без додаткових зусиль з боку розробника. Крім того, хмарні платформи, такі як Firebase, пропонують готові SDK для всіх популярних платформ, що значно прискорює розробку.

Додатковою перевагою є вбудована підтримка автентифікації користувачів, що дозволяє розмежувати права доступу до даних між різними користувачами без необхідності розробки власної системи логіну. Хмарні платформи також надають засоби для надсилання push-сповіщень на мобільні пристрої, що є критично важливим для оперативного інформування користувача про виявлені відхилення показників якості води. Важливою особливістю є можливість автоматичного резервного копіювання даних та забезпечення їхньої цілісності без участі розробника – ці функції повністю делегуються хмарному провайдеру. Завдяки використанню готових програмних інтерфейсів (SDK) значно спрощується інтеграція з клієнтськими застосунками на різних платформах, включаючи веб-браузери, мобільні пристрої на iOS та Android, а також сам мікроконтролер.

Окрім того, хмарні рішення, як правило, пропонують безкоштовні тарифні плани з обмеженнями, достатніми для навчальних та невеликих комерційних проєктів, що робить їх особливо привабливими для дипломного дослідження. Масштабованість таких платформ дозволяє легко розширювати систему – додавати нові точки моніторингу (додаткові ESP32) або нових користувачів без необхідності змінювати інфраструктуру чи писати додатковий серверний код. Вбудовані механізми безпеки, такі як правила доступу до бази даних та шифрування каналів зв'язку, забезпечують захист конфіденційних даних користувачів на рівні, який важко досягти при самостійному розгортанні сервера.

З точки зору надійності, хмарні провайдери гарантують високий рівень доступності сервісів (зазвичай 99.9% і вище), що критично для систем

моніторингу, які мають працювати безперервно. Технічне обслуговування, оновлення програмного забезпечення та захист від DDoS-атак повністю покладаються на провайдера, що дозволяє розробнику зосередитися виключно на прикладній логіці своєї системи. Крім того, хмарні платформи надають зручні веб-інтерфейси для адміністрування, моніторингу використання ресурсів та аналізу трафіку, що спрощує управління системою навіть для користувачів без глибоких технічних знань.

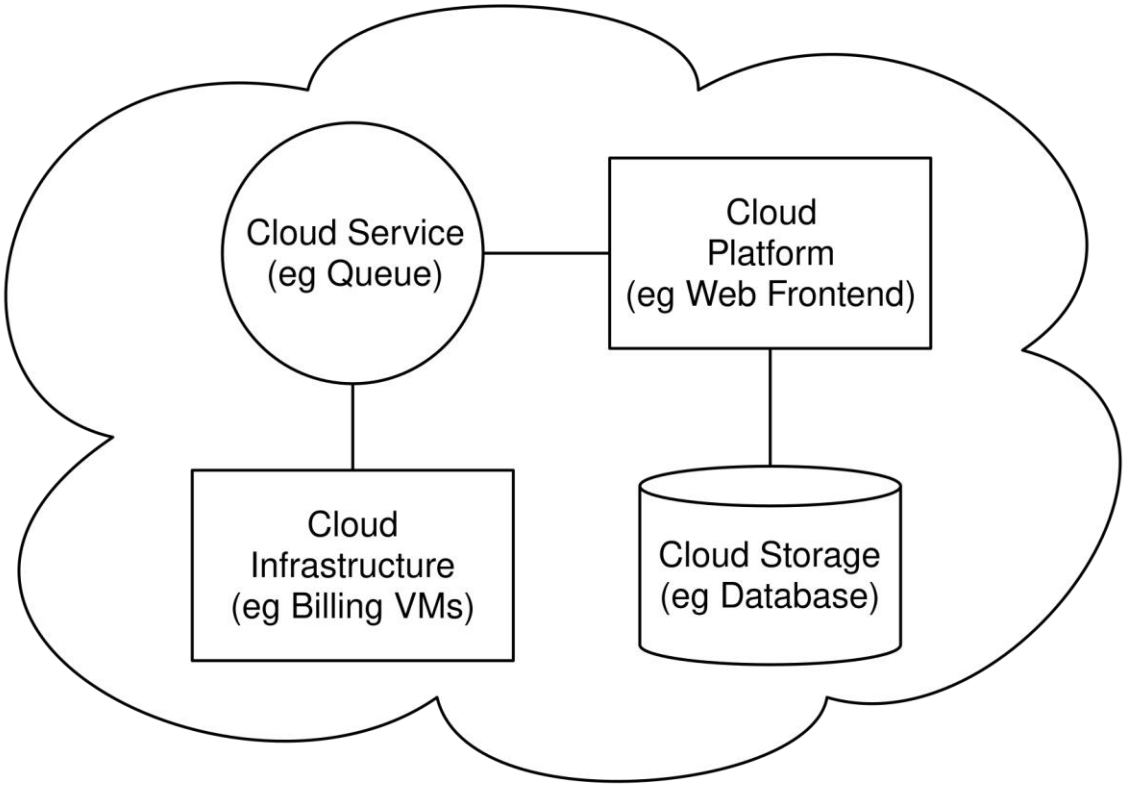



Рисунок 2.3 – Приклад хмарно-орієнтованої архітектури

Для обґрунтованого вибору архітектури було проведено порівняльний аналіз трьох розглянутих підходів за ключовими критеріями, які є найбільш важливими для програмно-технічного комплексу моніторингу якості води. До цих критеріїв належать: можливість віддаленого доступу до даних, обсяг зберігання історії вимірювань, підтримка роботи в реальному часі, складність

налаштування безпеки, вартість розгортання та масштабованість. Результати порівняння зведено у таблицю 2.1.

Таблиця 2.1 – Порівняння архітектурних підходів для ПТК моніторингу якості води

| Критерій | Монолітна архітектура | Локальна клієнт-серверна архітектура | Хмарно-орієнтована архітектура |
|---|---|---|---|
| Віддалений доступ до даних | Відсутній | Обмежений | Повний |
| Обсяг зберігання історії | Обмежений | Практично необмежений | Необмежений у межах тарифного плану |
| Підтримка реального часу | Так | Так | Так |
| Складність налаштування безпеки | Низька | Висока (потребує налаштування брандмауера, SSL-сертифікатів) | Низька (правила доступу Firebase) |
| Вартість розгортання | Низька | Середня | Безкоштовно |

Кінець таблиці 2.1

| | | | |
|---|---|---|---|
| Масштабованість | Низька | Середня | Висока |
| Енергоспоживання ESP32 | Низьке (немає постійного Wi-Fi) | Високе (постійне підключення до Wi-Fi) | Високе (постійне підключення до Wi-Fi) |

На основі проведеного аналізу для даного проєкту обрано хмарно-орієнтовану архітектуру з використанням Firebase як Backend-as-a-Service. Цей вибір обумовлений наступними факторами. По-перше, система повинна забезпечувати моніторинг у реальному часі – автоматичне знімання показників, миттєве формування звітів про відхилення та сповіщення користувачів. Firebase Realtime Database та Firestore нативно підтримують синхронізацію даних у реальному часі через механізм WebSockets, що дозволяє клієнтським застосункам отримувати нові вимірювання без необхідності періодичних HTTP-запитів, так званого polling. По-друге, вимога до зберігання історії вимірювань повністю забезпечується Firebase завдяки автоматичному зберіганню даних у хмарі з можливістю подальшої фільтрації, сортування та аналізу. По-третє, глобальна доступність даних є критичною, оскільки веб-панель та мобільний додаток повинні мати можливість відображати статистику споживання та якості води з будь-якого місця, де є доступ до інтернету. Четвертим фактором є спрощення розробки: Firebase надає готові программні інтерфейси для всіх використовуваних платформ - Web SDK для веб-панелі на React, React Native SDK для мобільного застосунку та бібліотеку urequests для ESP32 на MicroPython. Це значно скорочує час на реалізацію мережевої взаємодії та автентифікації порівняно з написанням власного серверного коду на мові JavaScript з використанням Express та реляційної бази даних на кшталт PostgreSQL. П'ятим фактором є безкоштовний тарифний план Spark, обсягів

якого (до 1 ГБ сховища, 10 ГБ трафіку на місяць, 20 тисяч операцій запису на день) більш ніж достатньо для одного пристрою ESP32 з періодичністю вимірювань раз у десять-п'ятнадцять хвилин. Додатковою перевагою є інтеграція Firebase Cloud Messaging з React Native, яка дозволяє надсилати push-сповіщення на мобільний пристрій при виявленні відхилень якості води від норми, наприклад, якщо показник pH перевищує 8.5 або загальна мінералізація TDS стає більшою за 500 частинок.

На основі обраної архітектури розроблено загальну структурну схему програмно-технічного комплексу, яка відображає всі основні компоненти та зв'язки між ними. Схема наведена на рисунку 2.4. Апаратна підсистема складається з мікроконтролера ESP32, який працює під керуванням інтерпретатора MicroPython, та трьох підключених датчиків pH-метра, TDS-метра та датчика температури. Програмне забезпечення на ESP32 виконує періодичне зчитування даних з цих датчиків, попередньо фільтрує аномальні значення, компенсує показники загальної мінералізації відповідно до поточної температури води, формує JSON-структуру та надсилає її до Firebase за допомогою HTTP-запиту, використовуючи бібліотеку urequests. Важливо зазначити, що вся апаратна частина реалізована виключно у симуляційному середовищі Wokwi, про що детальніше йтиметься нижче.

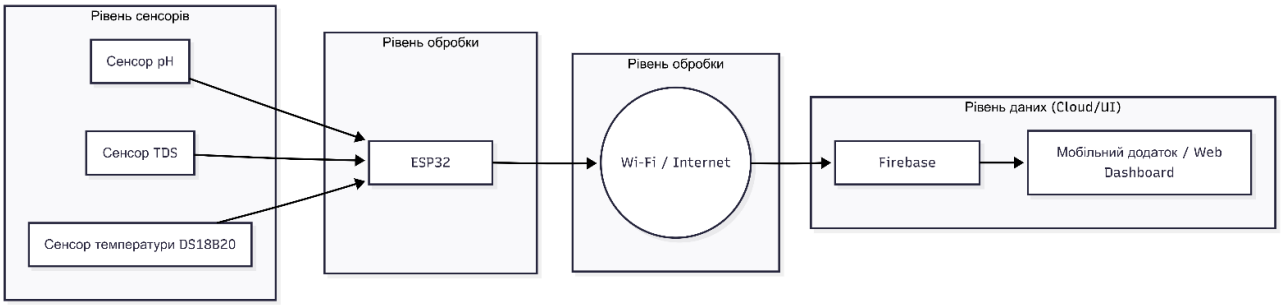



Рисунок 2.4 – Архітектурна схема програмно-технічного комплексу

Хмарна підсистема, яка представлена сервісами Firebase, виконує роль центрального сховища даних та брокера синхронізації. Показники якості води

зберігаються у колекції logs, кожен запис якої містить поля для значення pH, загальної мінералізації TDS, температури, часової мітки та статусу, що вказує на відповідність показників нормі. Правила безпеки налаштовані таким чином, що запис нових даних дозволений тільки авторизованим пристроям, тобто самому ESP32, а читання даних дозволене будь-якому авторизованому користувачеві системи. Веб-панель є односторінковим застосунком, реалізованим з використанням бібліотеки React [6]. Веб-застосунок підключається до Firebase через Web Software Development Kit, підписується на зміни в sensor_data і відображає актуальні показники у вигляді цифрових індикаторів та графіків, для побудови яких використовується бібліотека Chart.js або Recharts [34, 35]. Розробка веб-панелі велася у середовищі Visual Studio Code (рисунок 2.5). Мобільний застосунок є крос-платформним додатком, створеним з використанням фреймворку React Native [7] та середовища швидкого прототипування Expo Snack [10] (рисунок 2.6). Він забезпечує ті ж самі функції, що й веб-панель – перегляд поточних показників, історії вимірювань та звітів про відхилення. Додатковою можливістю мобільного застосунку є підтримка push-сповіщень через Firebase Cloud Messaging [32], що дозволяє користувачеві миттєво дізнаватися про погіршення якості води коли застосунок не відкритий.


Рисунок 2.5 – Редактор коду VS Code

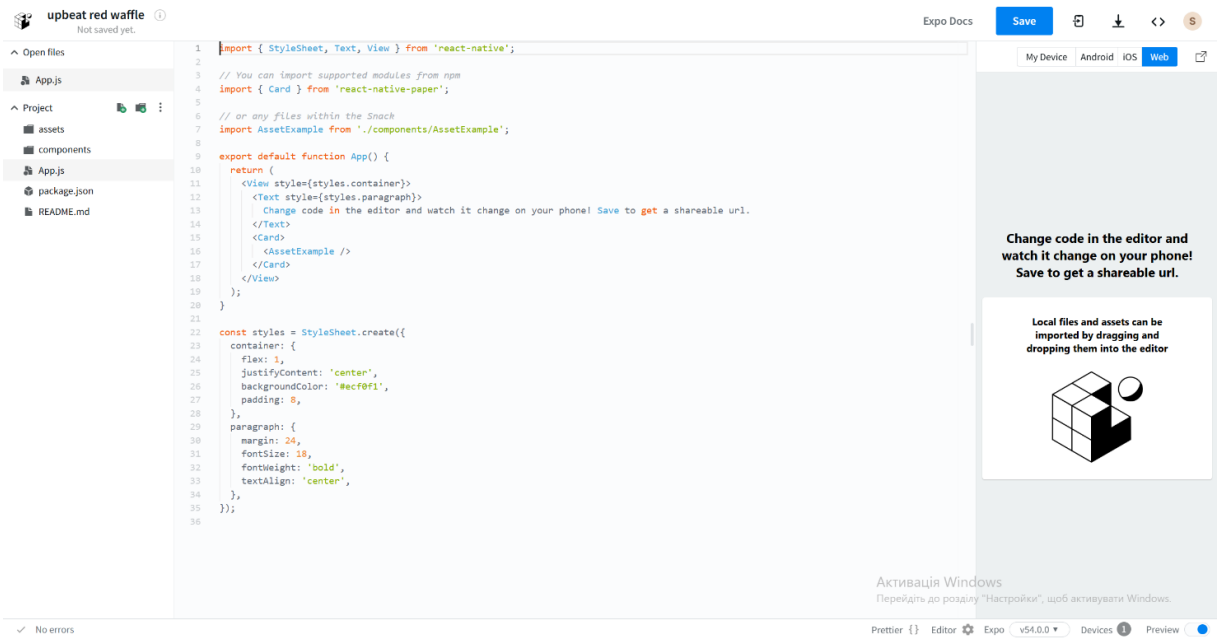



Рисунок 2.6 – Онлайн редактор Expo Snack

Використання Expo Snack дозволило швидко прототипувати та тестувати додаток без необхідності локального налаштування середовища розробки для операційних систем iOS та Android.

Окремо варто обґрунтувати використання симуляційного середовища Wokwi замість фізичного макету ESP32 з датчиками. Відсутність фізичного прототипу не є обмежуючим фактором для досягнення мети дипломної роботи з кількох причин. По-перше, Wokwi дозволяє повністю емулювати роботу мікроконтролера ESP32, виконувати код MicroPython, симулювати аналогові сигнали з датчиків pH та TDS, а також показники температури. Цього набору можливостей цілком достатньо для відлагодження алгоритмів збору даних, фільтрації викидів, компенсації температури та формування HTTP-запитів до Firebase. По-друге, код, написаний для симуляції в Wokwi, після незначних змін у конфігурації виводів GPIO може бути безпосередньо використаний для реального пристрою, що робить проєкт готовим до фізичної реалізації у майбутньому. По-третє, симуляція дозволяє виконувати десятки тестових запусків за хвилину, змінювати параметри навколишнього середовища, такі як температура або pH, без фізичної присутності біля обладнання, що значно

прискорює процес розробки. По-четверте, Wokwi є безкоштовним веб-інструментом, що не вимагає встановлення додаткового програмного забезпечення або придбання компонентів, що є критично важливим для студентського проєкту. І нарешті, згідно з вимогами до розділу проєктування, на цьому етапі не обов'язково мати фізичний прототип – достатньо обґрунтувати вибір компонентів та продемонструвати працездатність ключових алгоритмів. Wokwi повноцінно виконує цю функцію, тому його використання є повноцінною заміною фізичному макету на етапі проєктування та дозволяє сконцентруватися на розробці програмної частини – веб-панелі, мобільного застосунку та серверної логіки у Firebase.

## 2.2 Проєктування апаратної підсистеми

Апаратна підсистема є фундаментом програмно-технічного комплексу моніторингу якості водопровідної води, оскільки саме вона безпосередньо взаємодіє з об'єктом моніторингу – водою, та виконує первинне перетворення фізичних величин (кислотності, загальної мінералізації, температури та каламутності) в електричні сигнали, які можуть бути оброблені мікроконтролером. У цьому підрозділі обґрунтовується вибір кожного компонента апаратної підсистеми, наводиться їх технічна специфікація, описується схема підключення у симуляційному середовищі Wokwi, а також пояснюються причини вибору мови програмування MicroPython замість традиційної для ESP32 мови C++.

### 2.2.1 Вибір мікроконтролера

Центральним елементом апаратної підсистеми є мікроконтролер, який виконує опитування датчиків, обробку отриманих даних та передачу результатів до хмарної платформи Firebase. Після аналізу доступних на ринку рішень було

розглянуто три найбільш поширені варіанти для IoT-проєктів: платформа Arduino Uno, мікроконтролер ESP8266 та мікроконтролер ESP32.

Arduino Uno є найпростішою та найбільш розповсюдженою платформою для початківців. Вона має достатню кількість аналогових входів (шість) для підключення датчиків pH, TDS, а також цифрових виводів для датчика температури DS18B20. Проте Arduino Uno не має вбудованого Wi-Fi модуля, тому для передачі даних до Firebase потрібно додатково придбати та підключити зовнішній модуль на кшталт ESP8266 або Ethernet-шилд. Це ускладнює схему підключення, збільшує енергоспоживання та вартість кінцевого пристрою. Крім того, обчислювальна потужність мікроконтролера ATmega328P, тактовна частота 16 МГц, 2 КБ оперативної пам'яті, є недостатньою для роботи зі стеком TCP/IP та виконання HTTPS-запитів без додаткових модулів.

Мікроконтролер ESP8266 є більш сучасним рішенням, оскільки має вбудований Wi-Fi модуль, що дозволяє передавати дані безпосередньо до інтернету без додаткових компонентів. Його тактова частота становить 80 МГц (з можливістю розгону до 160 МГц), а обсяг оперативної пам'яті сягає 80 КБ. ESP8266 підтримує роботу з протоколами TCP/IP, HTTP та HTTPS, що робить його придатним для взаємодії з Firebase. Однак ESP8266 має лише один аналогово-цифровий перетворювач (ADC), що є критичним обмеженням для даного проєкту, оскільки потребує підключення трьох аналогових датчиків одночасно – pH-метра, TDS-метра та датчика турбідності. Використання лише одного ADC змусило б застосовувати зовнішній мультиплексор або комутувати канали зовнішніми реле, що значно ускладнює схему та знижує частоту опитування.

Мікроконтролер ESP32 є еволюційним продовженням ESP8266 і позбавлений його основних недоліків. Він має вбудований Wi-Fi модуль для передачі даних, Bluetooth для можливого розширення функціоналу, два 12-розрядних аналогово-цифрових перетворювачі (ADC1 та ADC2), які забезпечують до 18 каналів аналогового введення. Цього достатньо для

одночасного підключення всіх трьох аналогових датчиків pH, TDS. Тактова частота ESP32 може сягати 240 МГц, а обсяг оперативної пам'яті становить 520 КБ, що дозволяє виконувати складні обчислення, працювати з JSON-структурами та використовувати інтерпретатор MicroPython без суттєвих обмежень. Крім того, ESP32 підтримує роботу з протоколом OneWire на апаратному рівні, що спрощує підключення цифрового датчика температури DS18B20. На основі цих переваг для розроблюваного комплексу обрано мікроконтролер ESP32 у конфігурації DevKit V4, який є стандартною платою для розробки та повністю підтримується середовищем симуляції Wokwi.

### 2.2.2 Вибір та специфікація датчиків

Для виконання задачі моніторингу якості водопровідної води необхідно вимірювати три ключові параметри: кислотність (pH), загальну мінералізацію (TDS) та температуру. Кожен з цих параметрів потребує окремого датчика з відповідними характеристиками.

Датчик pH використовується для вимірювання кислотності води, яка є одним з найважливіших показників якості питної води. Нормальне значення pH для водопровідної води знаходиться в діапазоні від 6.5 до 8.5. У проєкті застосовується аналоговий pH-метр, який видає напругу в діапазоні від 0 до 3.3 В, пропорційну значенню pH. Цей датчик підключається до аналогового входу мікроконтролера, де за допомогою 12-розрядного АЦП значення напруги перетворюється у цифровий код від 0 до 4095. Подальше перетворення у значення pH виконується програмно за лінійною формулою, яка у коді реалізована у вигляді функції get_ph з файлу calculations.py, що повертає значення pH, округлене до двох десяткових знаків.

Датчик TDS призначений для вимірювання загальної концентрації розчинених твердих речовин у воді, що вимірюється в частинках на мільйон ppm. Для питної води допустимим вважається значення TDS до 500 ppm. Принцип

роботи TDS-метра базується на вимірюванні електропровідності води, чим більше розчинених солей та мінералів, тим вища провідність і, відповідно, вища вихідна напруга датчика. Як і pH-метр, TDS-метр є аналоговим датчиком з вихідним діапазоном 0-3.3 В, що підключається до окремого аналогового входу ESP32. Перетворення сирого значення АЦП у ppm виконується функцією get_tds, що повертає показник у частинках на мільйон. Важливою особливістю TDS-вимірювань є їх залежність від температури – при підвищенні температури електропровідність води зростає, що призводить до завищення показників. Тому у подальшій реалізації планується додати температурну компенсацію з використанням показників датчика DS18B20.

Датчик температури DS18B20 є цифровим датчиком з інтерфейсом OneWire, що дозволяє підключати кілька датчиків до одного цифрового виводу мікроконтролера. Діапазон вимірювання DS18B20 становить від -55°C до +125°C з точністю ± 0.5°C в діапазоні від -10°C до +85°C, що є цілком достатнім для моніторингу водопровідної води. Датчик має 64-бітний унікальний ідентифікатор, що дозволяє однозначно ідентифікувати його при роботі з кількома пристроями на одній шині. У коді робота з DS18B20 реалізована за допомогою бібліотек onewire та ds18x20. Функція get_temperature(roms, ds_sens or) виконує перетворення температури командою convert_temp, очікує 750 мілісекунд, необхідних для завершення вимірювання, та зчитує результат для кожного знайденого датчика. Характеристики датчиків можна переглянути в табличці 2.2.

Таблиця 2.2 – Технічні характеристики датчиків апаратної підсистеми

| Параметр | pH-метр | TDS-метр | DS18B20 |
|---|---|---|---|
| Тип сигналу | Аналоговий | Аналоговий | Цифровий |

Кінець таблиці 2.2

| | | | |
|---|---|---|---|
| Діапазон вимірювання | 0 – 14 pH | 0 – 1000 ppm | -55 – +125 °C |
| Точність | ±0.1 pH | ±10 ppm | ±0.5 °C |
| Вихідна напруга | 0 – 3.3 В | 0 – 3.3 В | цифровий протокол |

2.2.3. Схема підключення у середовищі Wokwi

Оскільки фізичний прототип на даному етапі розробки не виготовлявся, всі експерименти та налагодження виконувались у симуляційному середовищі Wokwi. Це середовище дозволяє створювати віртуальні схеми, підключаючи до плати ESP32 різноманітні компоненти, в тому числі потенціометри для емуляції аналогових датчиків та спеціалізовані компоненти на кшталт DS18B20.

На основі аналізу файлу конфігурації diagram.json було встановлено, що у проєкті використовуються наступні компоненти та підключення. Мікроконтрол ер ESP32 DevKit V4 працює у середовищі MicroPython версії 1.21.0. Для емуляції датчика pH використовується повзунковий потенціометр з ідентифікатором pot3, підключений до аналогового виводу GPIO34. Центральний контакт потенціометра (SIG) з'єднано з виводом 34, живлення (VCC) підключено до виводу 3V3 мікроконтролера, а земля (GND) до загального виводу GND. На схемі цей компонент позначено текстовою міткою «pH Sensor».

Для емуляції датчика TDS використовується повзунковий потенціометр з ідентифікатором pot1, підключений до аналогового виводу GPIO35. Вибір повзункового потенціометра замість звичайного обумовлений зручністю зміни значень у симуляції - повзунок дозволяє більш плавно регулювати опір та

відстежувати зміну показників. Аналогічно до pH-метра, цей компонент отримує живлення 3.3 В та підключений до землі. На схемі він позначений текстовою міткою «TDS Sensor».

Для емуляції датчика температури DS18B20 використовується спеціалізований симуляційний компонент з ідентифікатором temp1, який підключений до цифрового виводу GPIO4 за протоколом OneWire. Цей компонент має атрибут temperature, який задає поточне значення температури у симуляції у даному проєкті встановлено 18.5°C. Підключення виконано стандартно для DS18B20: виведення DQ (дані) підключено до GPIO4, живлення до 3V3, GND до загальної землі. На рисунку 2.7 наведено схему підключення компонентів у середовищі Wokwi.

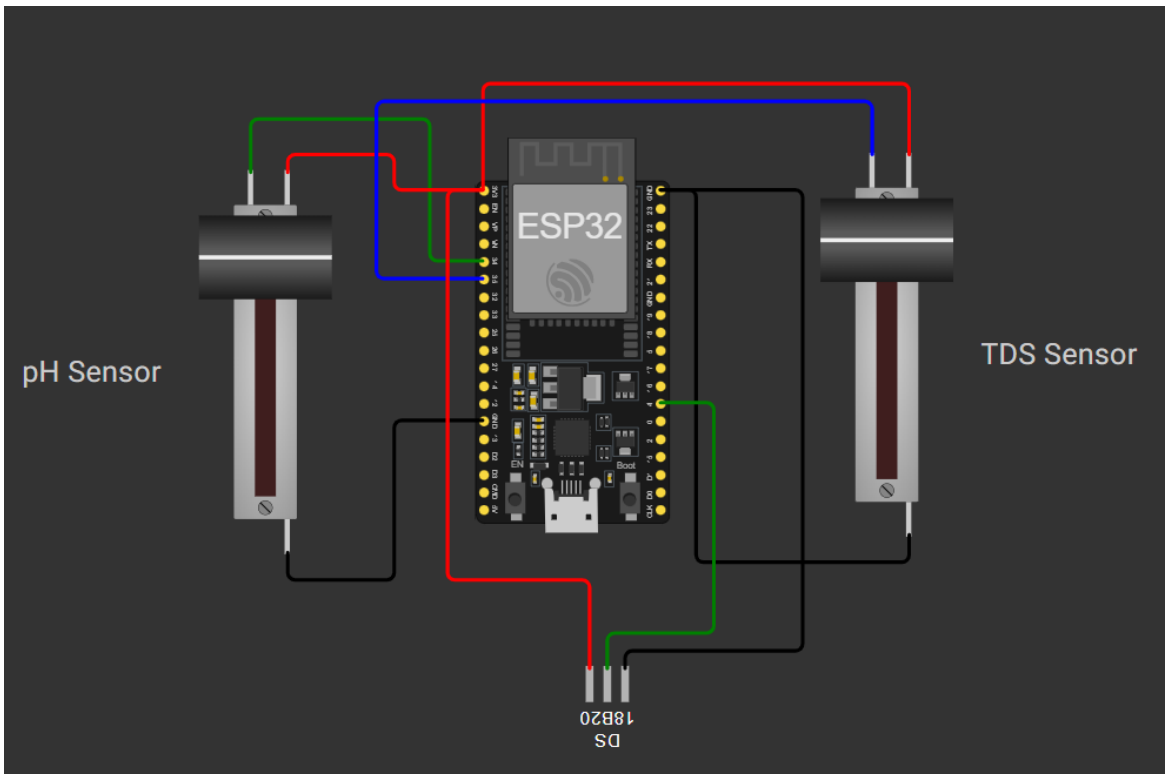



Рисунок 2.7 – Загальна принципова схема програмно-технічного комплексу в середовищі Wokwi

Слід зазначити, що у наведеній схемі використовуються потенціометри як замінники реальних аналогових датчиків. Це допустимий підхід на етапі

проєктування, оскільки дозволяє перевірити коректність роботи програмного забезпечення при зміні вхідних сигналів у всьому діапазоні значень.

### 2.2.4. Обґрунтування вибору MicroPython замість C++

Традиційно для мікроконтролерів ESP32 використовується мова C++ з фреймворком Arduino [14, 27]. Однак у даному проєкті було прийнято рішення використовувати MicroPython, що обумовлено декількома факторами.

Першим фактором є швидкість розробки та простота написання коду. MicroPython є високорівневою мовою з динамічною типізацією та великою кількістю вбудованих бібліотек. Код на MicroPython є більш лаконічним та читабельним порівняно з C++. Наприклад, підключення до Wi-Fi мережі, реалізоване у файлі boot.py, займає близько п'ятнадцяти рядків з використанням вбудованого модуля network, тоді як на C++ потрібно писати обробники подій та керувати станами підключення вручну.

Другим фактором є наявність інтерактивної консолі REPL (Read-Eval-Print-Loop). MicroPython надає можливість підключитися до ESP32 та виконувати команди в реальному часі, миттєво бачачи результат. Це значно спрощує налагодження: можна окремо перевірити роботу кожного датчика, змінити параметри на льоту, протестувати функції перетворення без перекомпіляції та перепрошивки пристрою. У середовищі Wokwi консоль REPL доступна безпосередньо у веб-інтерфейсі.

Третім фактором є легкість роботи з JSON та HTTP. Для передачі даних до Firebase необхідно формувати JSON-структури та виконувати HTTP-запити. MicroPython має вбудовані модулі json та urequests, які дозволяють виконувати ці операції ліченнями рядками коду. У C++ для роботи з JSON потрібно підключати сторонні бібліотеки на кшталт ArduinoJson, які споживають значну частину обмеженої пам'яті мікроконтролера.

Звісно, використання MicroPython має і недоліки, головним з яких є нижча продуктивність порівняно з компільованим C++ кодом, оскільки MicroPython є

інтерпретованою мовою. Крім того, MicroPython споживає більше оперативної пам'яті – ядро інтерпретатора займає близько 100-150 КБ flash-пам'яті та 20-30 КБ RAM. Однак для даного проєкту ці обмеження не є критичними: період опитування датчиків у тестовому режимі становить 3 секунди, а в робочому режимі планується інтервал у 10-15 хвилин, тому навіть повільніше виконання коду не впливає на загальну працездатність. Обсяг використовуваної пам'яті не перевищує доступних ресурсів ESP32. Таким чином, вибір MicroPython є виправданим пріоритетом швидкості розробки над максимальною продуктивністю.

## 2.3 Проєктування людино-машинного інтерфейсу

### 2.3.1. Загальні вимоги до інтерфейсу

На етапі проєктування сформульовано наступні вимоги до людино-машинного інтерфейсу. Інтерфейс має забезпечувати відображення поточних показників якості води (pH, TDS, температура) у наочній формі, бажано з використанням кольорової індикації безпеки. Історія вимірювань повинна бути представлена у вигляді хронологічного списку з можливістю фільтрації за датою. Для аналізу динаміки змін необхідно передбачити графічне відображення трендів. Інтерфейс має бути адаптивним однаково коректно відображатися на екранах різної роздільної здатності (від смартфонів до моніторів). Час реакції на дії користувача не повинен перевищувати однієї секунди, а оновлення даних має відбуватися автоматично без необхідності перезавантаження сторінки.

Для мобільного застосунку додатковими вимогами є підтримка push-сповіщень при виявленні відхилень, а також адаптація елементів керування до сенсорного введення кнопки мають бути достатньо великими, жести інтуїтивно зрозумілими.

### 2.3.2. Проєктування веб-панелі

Веб-панель проєктується як односторінковий застосунок, що складається з трьох основних функціональних блоків: верхньої інформаційної панелі з поточними показниками, блоку графіків історії та блоку сповіщень про відхилення.

Верхня інформаційна панель містить три картки, розташовані горизонтально. Кожна картка відповідає одному параметру якості води. Для pH-метра проєктується круговий вимірювальний прилад з діапазоном від 0 до 14. На шкалі приладу кольором виділяється безпечна зона від 6.5 до 8.5 зеленим кольором, тоді як небезпечні зони позначаються червоним. Стрілка приладу вказує на поточне значення, а в центрі круга великим шрифтом виводиться числове значення pH. Нижче приладу розташовується текстовий індикатор стану: «Безпечно» або «Небезпечно».

Для TDS-метра проєктується аналогічний круговий прилад, але з діапазоном від 0 до 1000 ppm. Безпечною зоною вважається діапазон від 0 до 500 ppm, тому зеленим кольором виділяється нижня половина шкали, а червоним – верхня (понад 500 ppm). Для температури пропонується використовувати не круговий прилад, а стовпчикову діаграму, оскільки для цього параметра важливішим є порівняння з фіксованою шкалою, аніж відносне відхилення. Стовпчик синього кольору зростає пропорційно температурі, а під ним виводиться числове значення в градусах Цельсія. Під кожною карткою розташовується невеликий текстовий блок з часом останнього оновлення даних. Нижче інформаційної панелі розташовується блок графіків історії, який відображає динаміку зміни показників за поточний день. Для кожного параметра будується окремий лінійний графік, де вісь X відповідає часу вимірювання, а вісь Y значенню параметра. Лінія графіка для pH має зелений колір, для TDS синій, для температури жовтогарячий, що забезпечує візуальне розрізнення параметрів навіть при побіжному погляді. Область під лінією графіка пропонується

зафарбовувати напівпрозорим кольором, що покращує читабельність та дозволяє оцінити загальний тренд змін. При наведенні курсора миші на будь-яку точку графіка передбачається відображення спливаючої підказки з точним значенням параметра та часом вимірювання, що забезпечує можливість детального аналізу окремих спостережень. На рисунку 2.8 наведено макет веб-панелі.

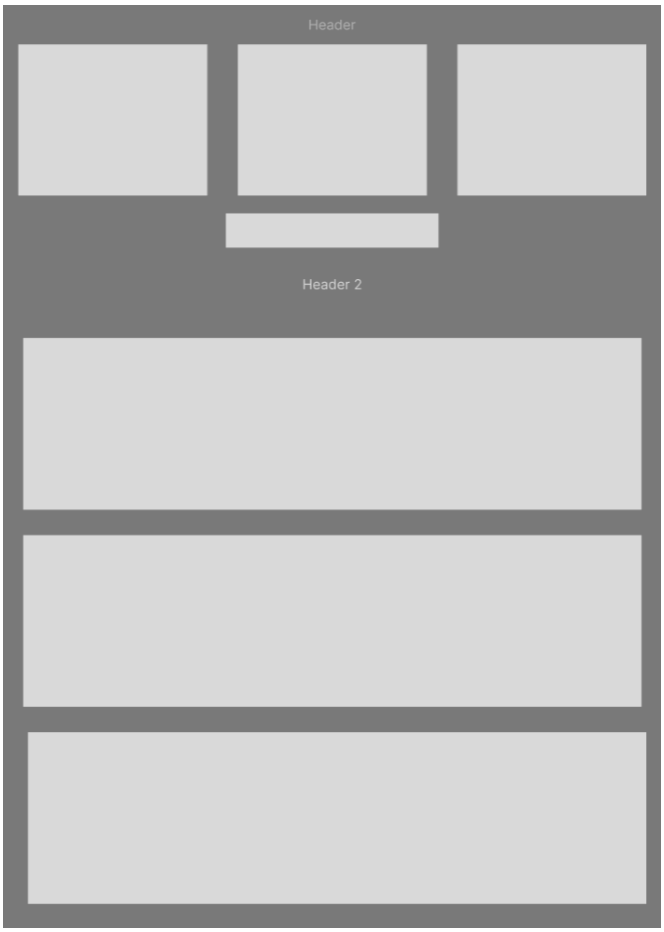



Рисунок 2.8 – Макет веб-панелі моніторингу якості води

Нижче інформаційної панелі розміщується блок графіків історії. Для кожного параметра будується окремий лінійний графік, на якому вісь X відображає час (години та хвилини), а вісь Y значення параметра. Графіки мають відображати дані за поточний день. Лінія графіка для pH має зелений колір, для TDS синій, для температури жовтогарячий. Область під лінією графіка пропонується зафарбовувати напівпрозорим кольором для кращої візуальної помітності. При наведенні курсора миші на будь-яку точку графіка має

з'являтися спливаюча підказка з точним значенням параметра та часом вимірювання.

### 2.3.3. Проєктування мобільного застосунку

Мобільний застосунок проєктується з урахуванням обмежень, характерних для смартфонів: менший розмір екрана, сенсорне керування, необхідність економії трафіку та енергії акумулятора. Навігація в застосунку організована за допомогою нижнього таб-меню, яке містить три основні розділи: «Моніторинг», «Історія» та «Аналітика». Така структура є стандартною для сучасних мобільних додатків і забезпечує швидкий доступ до всіх функцій одним дотиком.

Екран автентифікації є першим екраном, який бачить користувач після встановлення застосунку. Він містить два поля введення (електронна пошта та пароль), кнопку входу та посилання на екран реєстрації для нових користувачів. Екран реєстрації додатково містить поле для введення повного імені. Обидва екрани мають темне тло та підсвічування активних полів блакитним кольором. Після успішної автентифікації користувач автоматично переходить до головного екрана моніторингу.

Екран моніторингу є головним екраном застосунку. Через обмежений розмір екрана смартфона три картки з показниками розташовуються не в ряд, а у вигляді сітки 2 на 2: дві картки pH та TDS, пліч-о-пліч у верхньому ряді, а третя картка – температура на всю ширину екрана в нижньому ряді. Через те, що кругові прилади займають багато місця на маленькому екрані, пропонується спростити їх до великих чисел з підписами та кольоровою індикацією безпеки. Наприклад, значення pH виводиться великим шрифтом, а під ним напис «Безпечно» зеленим кольором або «Небезпечно» червоним. Додатково над цим блоком розташовується велика картка із загальним статусом води «БЕЗПЕЧНО» або «НЕБЕЗПЕЧНО» яка займає верхню частину екрана і привертає увагу своїм кольором.

Екран історії призначений для перегляду журналу вимірювань. Основним елементом цього екрана є вертикальний список, де кожен запис подається у вигляді окремої картки. Картка містить дату та час вимірювання, у верхньому лівому куті, іконку стану у верхньому правому куті, а також три числові показники: pH, TDS, температура в рядок унизу. Якщо вимірювання було небезпечним (хоча б один показник вийшов за норму), картка має легкий червонуватий фон. Проєктується механізм посторінкового завантаження: спочатку завантажуються записи лише за поточний день, а внизу списку розташовується кнопка «Завантажити попередній день». Користувач також може оновити список жестом «потягнути вниз».

Екран аналітики призначений для візуалізації трендів зміни показників. Через невеликий розмір екрана три графіки розташовуються один під одним. Кожен графік є лінійним і відображає останні десять вимірювань. Вісь часу не підписується, оскільки на маленькому екрані підписи створювали б візуальний шум. Під кожним графіком розташовуються три статистичні показники: мінімальне, максимальне та середнє значення за обраний період. Таке компонування дозволяє користувачеві швидко оцінити загальну тенденцію, не вчитуючись у дрібні деталі.

Система сповіщень є важливою відмінністю мобільного застосунку від веб-панелі. Проєктується, що при виявленні відхилення якогось показника від норми застосунок генерує push-сповіщення. Сповіщення з'являється у вигляді банера у верхній частині екрана, навіть якщо застосунок згорнуто або екран заблоковано. Сповіщення має містити короткий текст, наприклад: «Увага! pH води перевищує норму». Одночасно зі сповіщенням має спрацьовувати вібрація пристрою, щоб гарантовано привернути увагу користувача. Користувач може налаштувати, чи отримувати йому сповіщення, у налаштуваннях телефону. На рисунку 2.9 наведено схему навігації між екранами мобільного застосунку, яка показує як користувач переміщується між екранами.

На рисунку 2.9 наведено схему навігації між екранами мобільного застосунку, яка показує, як користувач переміщується між екранами. Така архітектура забезпечує інтуїтивно зрозумілий та зручний досвід використання, мінімізуючи кількість дій, необхідних для отримання актуальної інформації про якість води. Завдяки поєднанню автентифікації, посторінкового завантаження історії, аналітичних графіків та миттєвих сповіщень, розроблений мобільний застосунок повністю задовольняє потреби сучасного користувача в оперативному моніторингу стану водопровідної води.

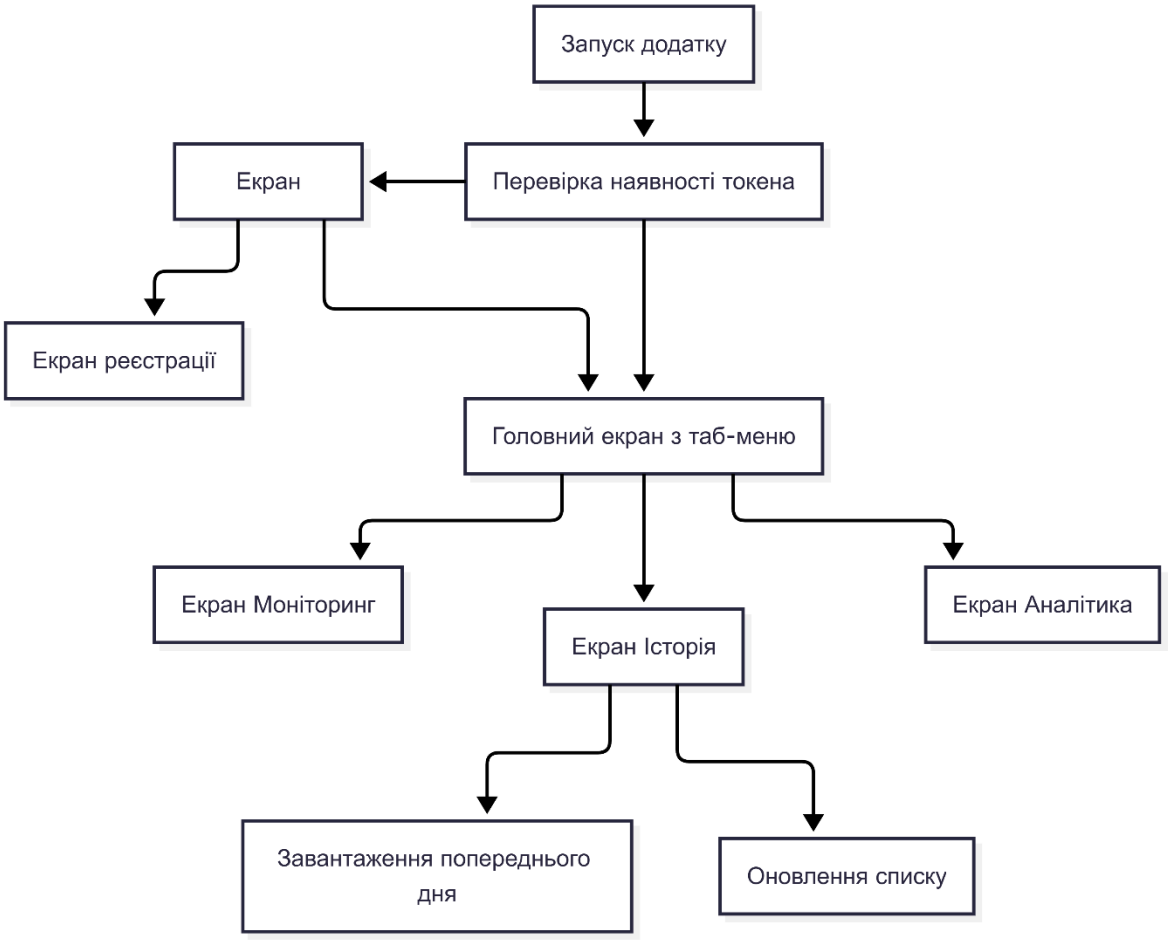



Рисунок 2.9 – Схема навігації мобільного застосунку

## 2.4 Висновки до розділу 2

У другому розділі пояснювальної записки виконано проєктування програмно-технічного комплексу моніторингу якості водопровідної води в режимі реального часу. На основі аналізу предметної області та сформульованих вимог до системи розроблено архітектурне, апаратне та програмне рішення, яке забезпечує автоматичне знімання показників з датчиків, зберігання історії вимірювань у хмарній базі даних, аналіз відхилень від норм та надання даних користувачеві через веб-панель і мобільний застосунок.

Проведено порівняльний аналіз трьох типів архітектур: монолітної, локальної клієнт-серверної та хмарно-орієнтованої. На основі таких критеріїв, як необхідність віддаленого доступу до даних, зберігання історії вимірювань, робота в реальному часі, складність налаштування безпеки та вартість розгортання, обрано хмарно-орієнтовану архітектуру з використанням Firebase Realtime Database. Це рішення забезпечує глобальну доступність даних, автоматичну синхронізацію між клієнтами, спрощує розробку завдяки готовим програмним інтерфейсам і не потребує утримання власного сервера.

Апаратна підсистема. Обґрунтовано вибір мікроконтролера ESP32, який має вбудований Wi-Fi модуль, достатню кількість аналогових входів для підключення датчиків pH та TDS, а також підтримує протокол OneWire для цифрового датчика температури DS18B20. Для кожного датчика наведено технічні характеристики та обґрунтовано доцільність його використання. Розроблено схему підключення компонентів у симуляційному середовищі Wokwi, де аналогові датчики емулюються за допомогою потенціометрів. Окремо обґрунтовано вибір мови програмування MicroPython замість традиційної C++ на користь швидкості розробки, простоти налагодження через інтерактивну консоль REPL та зручності роботи з JSON і HTTP.

Програмне забезпечення ESP32. Спроектовано алгоритм роботи мікроконтролера, який включає ініціалізацію Wi-Fi, синхронізацію часу через

NTP, основний цикл опитування датчиків, порогову фільтрацію даних та передачу результатів до Firebase через HTTP PUT та POST запити. Такий підхід дозволяє уникнути засмічення бази даних незначними коливаннями показників.

Серверна частина. Спроектовано структуру Firebase Realtime Database, яка включає вузол sensor_data для зберігання поточного вимірювання, вузол logs з організацією за датами для зберігання історії та вузол users для майбутньої автентифікації. Розроблено правила безпеки для контролю доступу до даних та індекси для оптимізації запитів до історії вимірювань.

Людино-машинний інтерфейс. Розроблено проєкти веб-панелі на React та мобільного застосунку на React Native. Веб-панель містить кругові шкали для pH та TDS, стовпчикову діаграму для температури та лінійні графіки для відображення історії. Мобільний застосунок адаптовано до сенсорного керування з нижнім навігаційним меню на три розділи, а також доповнено push-сповіщеннями при виявленні відхилень. Обидва інтерфейси витримано в єдиній темній кольоровій гамі.

Методи комп'ютерної інженерії. Проведено аналіз методів фільтрації сигналів, калібрування датчиків та контролю цілісності даних при передачі. Обрано порогову фільтрацію як найбільш просту та ефективну для даного застосування, лінійне калібрування для перетворення сигналів датчиків та серверні часові мітки для коректного датування подій.

Усі спроектовані рішення враховують специфіку розробки та створюють основу для подальшої програмної реалізації та тестування системи. Розроблений проєкт є готовим до фізичного втілення, схеми підключення та структура бази даних спроектовані таким чином, що при переході від симуляції до реального обладнання знадобляться лише мінімальні зміни.

# 3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ

## 3.1 Проєктування програмного забезпечення для ESP32

Програмне забезпечення для ESP32 складається з трьох основних файлів, кожен з яких виконує певну функцію у загальній архітектурі системи. Файл boot.py відповідає за початкову ініціалізацію системи, підключення до бездротової мережі Wi-Fi та реалізує функції для взаємодії з Firebase Realtime Database. Файл calculations.py містить допоміжні математичні функції для перетворення сирих значень аналогово-цифрового перетворювача у фізичні величини - pH, TDS та температуру, а також функцію формування рядка з поточною датою та часом. Файл main.py є головним модулем, який виконує нескінченний цикл опитування датчиків, порівняння отриманих значень з попередніми, прийняття рішення про необхідність передачі даних до хмари та власне передачу. Така модульна структура забезпечує легкість підтримки коду, можливість багаторазового використання функцій та спрощує налагодження, оскільки кожен модуль може бути протестований окремо.

Файл boot.py виконується автоматично при старті мікроконтролера. У ньому визначено константи SSID та PASSWORD, які зберігають ім'я та пароль бездротової мережі. Функція connect_wifi створює об'єкт WLAN у режимі клієнта, активує його та намагається встановити з'єднання із заданою точкою доступу. Якщо з'єднання не вдається встановити одразу, функція виконує до десяти спроб з інтервалом в одну секунду. Після успішного підключення IP-адреса присвоюється автоматично за протоколом DHCP. Ця функція викликається з головного файлу main.py перед початком основного циклу вимірювань. Крім того, boot.py містить дві функції для роботи з Firebase. Функція put_data виконує HTTP PUT-запит до вузла sensor_data.json у базі даних Firebase, повністю замінюючи вміст цього вузла новими даними. Такий підхід є доцільним, оскільки у даній реалізації важливими є лише найактуальніші

показники якості води. Функція put_logs виконує HTTP POST-запит до вузла logs.json, що додає новий запис до масиву логів. Це дозволяє зберігати історію вимірювань з часовими мітками, які формуються на стороні Firebase за допомогою спеціального об'єкта ".sv". Обидві функції після виконання запиту закривають з'єднання викликом response.close та перевіряють статус-код відповіді, значення 200 свідчить про успішне збереження даних у хмарі.

Файл calculations.py містить виключно чисті функції перетворення, які не мають побічних ефектів. Функція get_temperature отримує список ROM-ідентифікаторів датчиків DS18B20 та об'єкт сенсора, після чого ініціює вимірювання температури викликом ds_sensor.convert_temp, очікує 750 мілісекунд, необхідних для завершення перетворення, та зчитує значення для першого датчика у списку. Функція get_ph отримує сире значення з аналогового входу та перетворює його у значення pH. Ділення на 4095 пов'язане з роздільністю 12-розрядного АЦП, а множення на 14 приводить діапазон до шкали pH від 0 до 14. Результат округлюється до двох десяткових знаків. Функція get_tds працює за аналогічним принципом, але множення виконується на 1000, оскільки діапазон TDS становить від 0 до 1000 ppm. Функція get_full_date формує рядок з поточною датою та часом у форматі РРРР-ММ-ДД ГГ:ХХ:СС з урахуванням часової зони UTC+2, що відповідає Києву. Зсув на дві години реалізований додаванням константи 2 * 3600 секунд до значення, отриманого від функції time.

Файл main.py є головним модулем, який організовує нескінченний цикл опитування датчиків. Після імпорту необхідних бібліотек та модулів він ініціалізує аналогові входи на пінах 34, 35 та 32 (температура, не використовується у фінальній версії, оскільки температура зчитується цифровим датчиком), налаштовує їх атенюатор на рівень ADC.ATTN_11DB, що дозволяє вимірювати напругу до 3.3 В, та ініціалізує шину OneWire на піні 4 для підключення DS18B20. Після цього виконується сканування пристроїв на шині методом scan, який повертає список ROM-кодів усіх знайдених датчиків. Далі

викликаються функції connect_wifi з модуля boot для підключення до інтернету та ntptime.settime для синхронізації внутрішнього годинника ESP32 з серверами точного часу через протокол NTP.

Основним циклом програми є конструкція while True, яка виконується безперервно після завершення всіх ініціалізацій. У тілі циклу відбувається зчитування значень з трьох аналогових входів за допомогою методу read(), який повертає ціле число від 0 до 4095. Отримані сирі значення передаються до відповідних функцій перетворення з модуля calculations, які повертають фізичні величини - pH, TDS у ppm та температуру у градусах Цельсія. Особливістю реалізації є те, що зчитування температури виконується окремо через цифровий датчик DS18B20, а аналоговий вхід на піні 32, позначений як turb_pin, у поточній версії коду не використовується для вимірювань, але залишений для можливого майбутнього розширення функціоналу.

Ключовим механізмом, що мінімізує кількість запитів до Firebase та зменшує навантаження на мережу і хмарну базу даних, є система порогових значень. У коді визначено три константи: ph_threshold, tds_threshold та temp_threshold. Змінні last_ph, last_tds та last_temp зберігають значення попер еднього успішного вимірювання, яке було відправлено до Firebase. Після кожного нового циклу вимірювань обчислюється абсолютна різниця між поточним та попереднім значенням за допомогою функції abs. Якщо ця різниця перевищує відповідне порогове значення хоча б для одного з трьох параметрів, фіксується факт значної зміни показників якості води. У цьому випадку формується JSON-об'єкт data, який містить усі поточні показники та часову мітку у текстовому форматі, після чого викликаються функції put_data() та put_logs(). Перша з них оновлює поточний стан системи у Firebase, а друга додає новий запис до історії вимірювань з серверною часовою міткою. Після успішної передачі змінні last_ph, last_tds та last_temp оновлюються поточними значенням и. Якщо ж жоден з параметрів не змінився більше, ніж на заданий поріг, передача даних не виконується, що запобігає засміченню бази даних дублюючими або

несуттєво різними вимірюваннями. Незалежно від того, чи були передані дані, значення виводяться в послідовний порт за допомогою функції print() для візуального контролю у консолі. Наприкінці кожної ітерації циклу виконується пауза тривалістю 3 секунди, яка у тестовому режимі дозволяє спостерігати за змінами показників при ручному регулюванні потенціометрів. У робочому режимі цей інтервал планується збільшити до 10-15 хвилин. На рисунку 3.8 та 3.9 наведено блок-схеми алгоритмів роботи основного циклу програми на ESP32, яка візуалізує послідовність дій від ініціалізації до прийняття рішення про передачу даних до хмари.

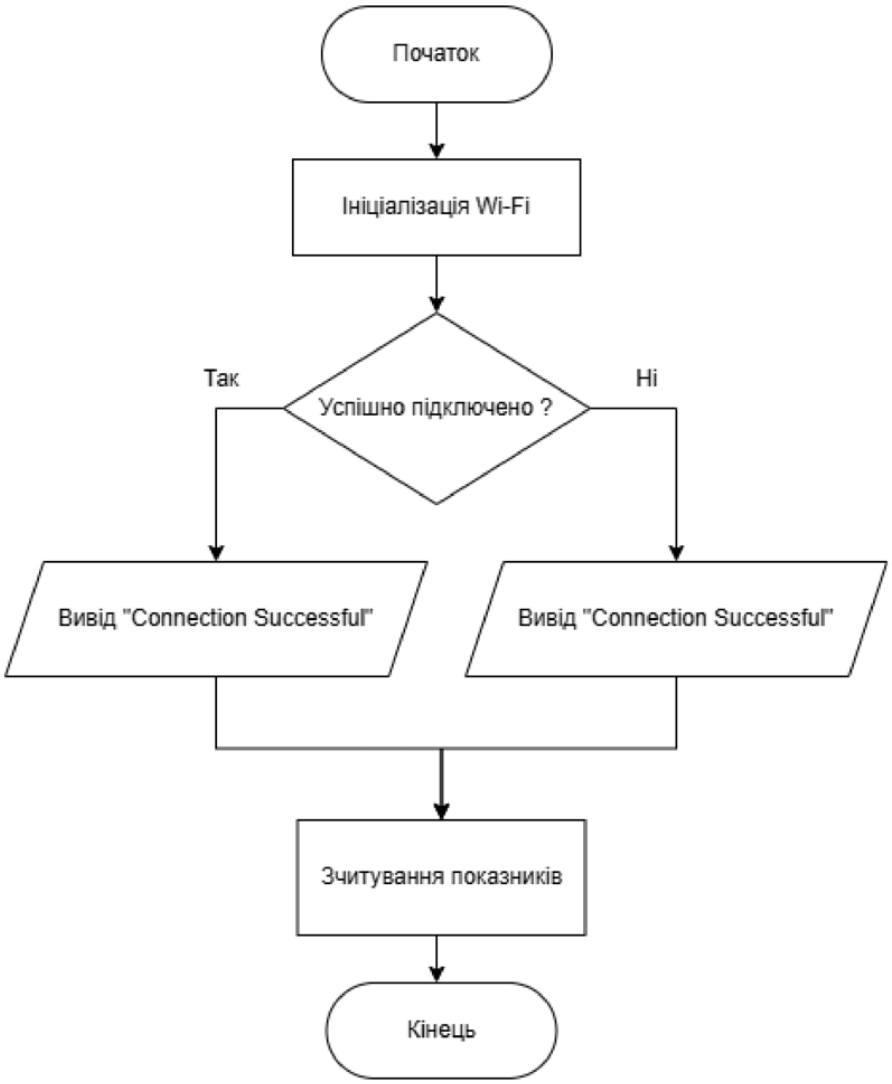



Рисунок 3.8 – Блок-схема процесу ініціалізації Wi-Fi з’єднання та обробки мережевих виключень

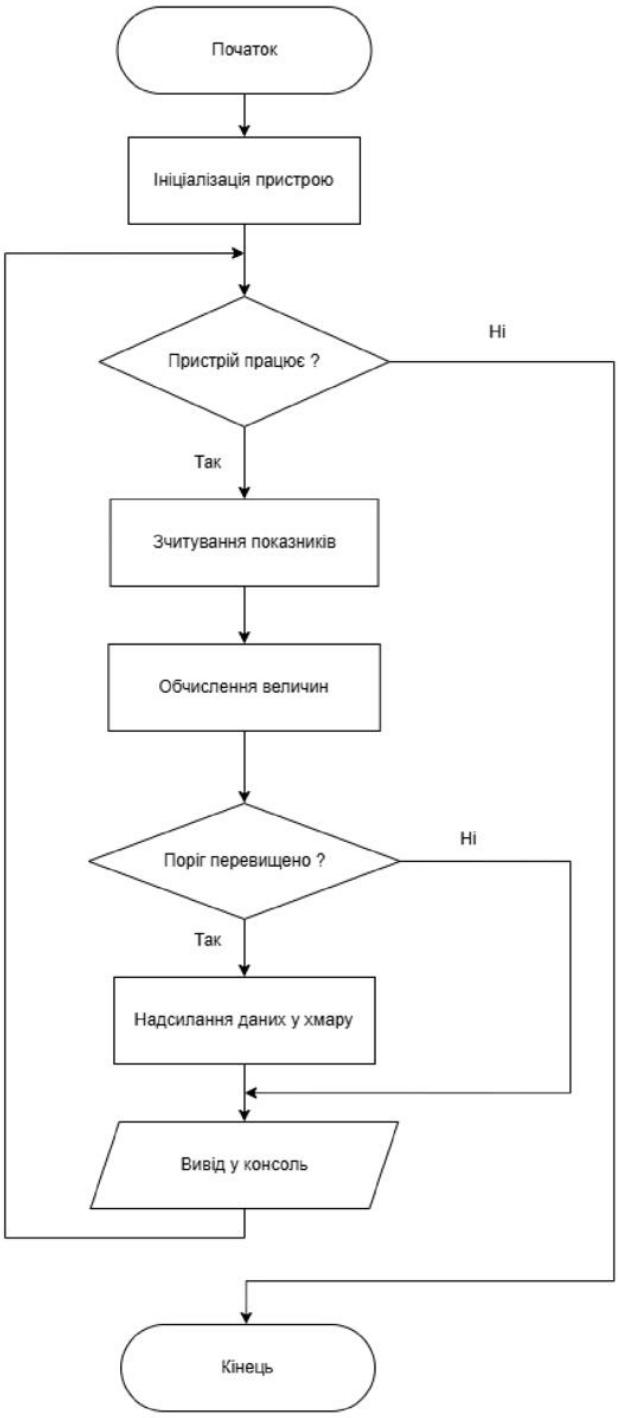



Рисунок 3.9 – Блок-схема алгоритму оновлення показників датчиків

Взаємодія з Firebase Realtime Database реалізована через простий протокол HTTP REST, який підтримується бібліотекою urequests у MicroPython. Як зазначалося раніше, запис даних виконується двома різними способами залежно від типу інформації.

Для зберігання поточного стану системи використовується вузол sensor_data.json. Функція put_data виконує HTTP PUT-запит до цього вузла, передаючи у тілі запиту JSON-об'єкт з поточними показниками pH, TDS, температури та часу. PUT-метод обрано тому, що він повністю замінює вміст вузла новими даними, що відповідає вимогам зберігання лише найактуальнішого вимірювання. Після виконання запиту відповідь сервера містить статус-код, який перевіряється на рівність 200. Якщо статус-код підтверджує успіх, у консоль виводиться повідомлення "Saved successfully".

Для зберігання історії вимірювань використовується окремий вузол logs.json. Функція put_logs виконує HTTP POST-запит до цього вузла. На відміну від PUT, метод POST додає новий об'єкт до кінця масиву (якщо вузол є масивом) або створює новий дочірній елемент з унікальним ключем, згенерованим Firebase. У даній реалізації до логів передається лише значення pH та TDS, а також спеціальний об'єкт timestamp, який інструктує Firebase автоматично підставити час отримання запиту у форматі Unix timestamp. Це є більш надійним способом фіксації часу, ніж генерація часової мітки на стороні клієнта, оскільки виключає похибки, пов'язані з неточною синхронізацією годинника ESP32 або затримками при передачі даних.

## 3.2 Проєктування серверної частини та бази даних у Firebase

Серверна частина програмно-технічного комплексу реалізована з використанням хмарної платформи Firebase, а саме її компонента Realtime Database. Цей сервіс забезпечує зберігання даних у реальному часі, синхронізацію між усіма підключеними клієнтами, а також надає засоби для захисту інформації через систему правил безпеки. Вибір Firebase Realtime Database обумовлений декількома факторами: зберігання даних у вигляді єдиного JSON-дерева є природним форматом для обміну з ESP32 на MicroPython, Realtime Database надає простий REST API [28, 29], що використовується у

функціях put_data та put_logs з файлу boot.py, має меншу затримку порівняно з Firestore, а безкоштовний тариф Spark надає 1 ГБ сховища та 10 ГБ трафіку, чого достатньо для даного проєкту.

Структура бази даних складається з трьох основних вузлів, що підтверджується скріншотом консолі Firebase на рисунку 3.10. Вузол sensor_data призначений для зберігання поточного вимірювання. Він є одиничним об'єктом, який повністю перезаписується при кожній передачі даних за допомогою HTTP PUT-запиту. Структура цього об'єкта включає поля: кислотність, загальна мінералізація, температура та дата і час вимірюванн я. Вузол logs призначений для зберігання історії вимірювань. Записи додаються за допомогою HTTP POST-запиту, причому кожен запис містить поля ph, tds та спеціальне поле timestamp з об'єктом ".sv", який повідомляє Firebase автоматич-но підставити серверний час. Вузол users призначений для майбутнього розшир ення функціоналу – додавання автентифікації користувачів та персоналізації доступу.


Рисунок 3.10 – Структура вузлів Firebase Realtime Database

Для захисту даних від несанкціонованого доступу у Firebase Realtime Database використовуються правила безпеки. На поточному етапі розробки, з метою спрощення налагодження, для вузлів sensor_data та logs надано право на читання та запис для всіх клієнтів. Це дозволяє ESP82 вільно передавати дані, а

веб-панелі та мобільному застосунку їх зчитувати. Для вузла users встановлено заборону на читання та запис до моменту впровадження системи автентифікації. Оскільки з часом кількість записів у вузлі logs зростатиме, для забезпечення швидкої вибірки даних налаштовано індексацію за полями timestamp, ph та tds, що дозволяє виконувати фільтрацію та сортування без повного перебору всіх записів.

## 3.3 Проєктування клієнтських застосунків

### 3.3.1. Веб-панель на React для відображення статистики якості води

Веб-панель є односторінковим застосунком, розробленим з використанням бібліотеки React. Вона призначена для візуалізації поточних показників якості води у реальному часі, відстеження історії вимірювань та аналізу відхилень від норми. Веб-панель безпосередньо підключається до Firebase Realtime Database, отримуючи оновлення через механізм WebSockets, що забезпечує миттєву синхронізацію даних без необхідності перезавантаження сторінки.

Основним компонентом веб-панелі є ControlPanel.jsx, який виконує роль головного контролера всього застосунку. У цьому компоненті за допомогою хука useState визначено два стани: waterData для зберігання поточних показників pH, TDS, температури, часу та historyLogs для зберігання історії вимірювань за поточну дату. Початкові значення встановлені як нульові для числових показників та поточний час для поля time. Хук useEffect використовується для налаштування підписок на зміни в базі даних Firebase одразу після монтування компонента. Перша підписка створюється на вузол sensor_data за допомогою функції onValue, де sensorDataRef є посиланням на шлях sensor_data у базі даних, отриманим з файлу конфігурації firebase.js. Кожного разу, коли ESP32 оновлює поточне вимірювання через PUT-запит, функція зворотного виклику отримує новий знімок даних, з якого вилучається значення методом val, після чого оновлюється

стан waterData через setWaterData. Друга підписка створюється на вузол logs/today, де today формується у форматі РРРР-ММ-ДД. Такий підхід дозволяє завантажувати лише логи за поточну дату, що зменшує обсяг переданих даних та прискорює роботу графіків. Обидві підписки повертають функції відписки, які викликаються у функції очищення, що запобігає витокам пам'яті при демонтуванні компонента. Використання саме onValue замість одноразового get забезпечує роботу у реальному часі, панель автоматично оновлюється при кожній зміні даних без додаткових запитів.

Файл firebase.js містить конфігурацію підключення до Firebase. Об'єкт firebaseConfig зберігає унікальні ідентифікатори проєкту apiKey, authDo main, databaseURL, projectId та інші. Функція initializeApp ініціалізує з'єднання з Firebase, після чого getDatabase створює об'єкт бази даних. Далі створюються посилання на кореневі вузли: sensorDataRef та для логів. Використання окремого файлу конфігурації дозволяє централізовано керувати налаштуваннями підключення та перевикористовувати їх у будь-якому компоненті проєкту.

Інтерфейс веб-панелі складається з трьох основних секцій, що відображаються у компоненті ControlPanel. Перша секція містить три картки з візуалізацією поточних показників у вигляді вимірювальних приладів та діаграм. Друга секція відображає час останнього оновлення даних, що дозволяє користувачеві оцінити актуальність інформації. Третя секція містить графіки історії вимірювань за поточний день для кожного параметру окремо.

Компонент PHCard.jsx відповідає за відображення показника кислотності. Він отримує значення pH через пропси та визначає змінну isSafe, яка є true, якщо pH знаходиться у безпечному діапазоні від 6.5 до 8.5, та false в іншому випадку. Для візуалізації використовується компонент Gauge з бібліотеки mui. Параметр value встановлює поточне значення, valueMax визначає максимальне значення шкали, а параметри startAngle та endAngle задають кутові межі дуги приладу. Колір дуги змінюється динамічно: зелений для безпечних значень та червоний для небезпечних за допомогою умовного оператора у властивості fill.

Під графіком розташовано дві інформаційні панелі: одна показує поточне числове значення pH, інша текстовий висновок про безпеку або небезпеку з відповідним кольоровим маркуванням.

Компонент TDSCard.jsx має аналогічну структуру, але призначений для відображення загальної мінералізації у ppm. Безпечним вважається значення TDS нижче 500 ppm, тому умова isSafe перевіряє, чи не перевищує tds цього порогу. Максимальне значення шкали встановлено на 1000 ppm, що є типовим діапазоном для побутових TDS-метрів. Колір дуги також змінюється залежно від безпеки показника.

Компонент TemperatureCard.jsx дещо відрізняється від попередніх двох - замість кругової шкали тут використовується стовпчикова діаграма BarChart з тієї ж бібліотеки mui. Вісь X має єдину категорію «Temperature», вісь Y налаштована на діапазон від 0 до 50 градусів Цельсія. Колір стовпчика залежить від температури: синій для нормальних значень та червоний для підвищених значень. Такий вибір візуалізації обумовлений тим, що для температури більш інформативним є порівняння з фіксованою шкалою, аніж зміна у відносних одиницях.

Компонент HistoryChart.jsx відповідає за побудову графіків історії вимірювань. Він приймає чотири параметри: об'єкт з логами від Firebase, заголовок графіка, назва поля для відображення, та колір лінії графіка. У тілі компонента виконується перетворення вхідного об'єкта у масив, придатний для побудови графіка. Функція Object.values перетворює об'єкт Firebase у масив значень, після чого викликається метод map, який для кожного елемента створює новий об'єкт з полями time та value –числове значення відповідного параметра. Потім виконується сортування за часовою міткою для коректного відображення хронології. Для побудови графіка використовується компонент LineChart з параметрами: xAxis отримує масив відформатованих часів, series отримує масив значень та колір лінії, а параметр area: true додає підсвічування області під лінією графіка. Додатково налаштовано стилізацію

осей через властивість sx колір тексту підписів встановлено білим, а лінії осей сірими.

У головному файлі App.jsx відбувається лише рендеринг компонента Cont rolPanel, що робить структуру застосунку максимально простою та зручною для подальшого розширення. Файл panel.module.css містить стилі для всіх компонентів. Тут застосовано сучасний дизайн з напівпрозорими картками, що досягається за допомогою властивості backdrop-filter, яка створює ефект розмиття заднього фону. Градієнтний фон карток виконаний з переходом від темно-синього до майже прозорого. Всі картки мають округлі кути, тінь та білий текст.

Таким чином, веб-панель на React забезпечує повноцінний моніторинг якості води у реальному часі[44]. Вона автоматично отримує оновлення з Firebase, візуалізує поточні показники за допомогою кругових шкал та стовпчикових діаграм, колір яких сигналізує про відхилення від норми, а також відображає історію вимірювань за поточний день у вигляді лінійних графіків. Використання бібліотеки mui забезпечує високу якість візуалізації та адаптивність до різних розмірів екрану. На рисунках 3.11 та 3.12 наведено зовнішній вигляд веб-панелі з усіма основними елементами інтерфейсу.

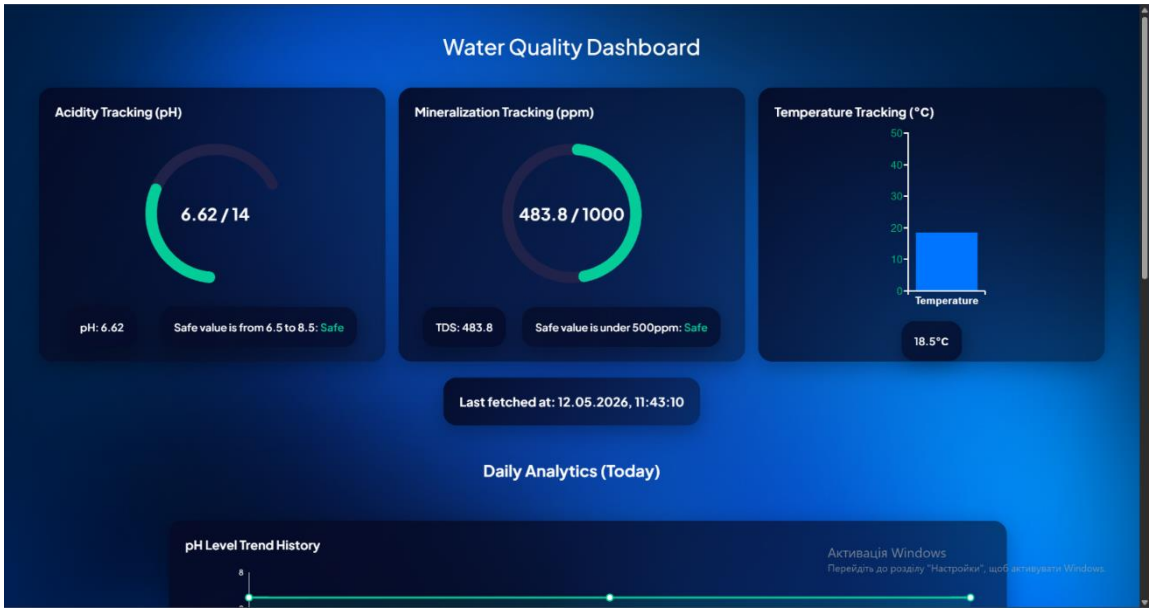



Рисунок 3.11 – Поточні показники води у веб-панелі


Рисунок 3.12 – Графіки історії вимірювань за поточний день

### 3.3.2. Мобільний застосунок на React Native (Expo Snack)

Мобільний застосунок розроблено з використанням фреймворку React Native та середовища Expo Snack, що дозволяє створювати кросплатформні додатки для операційних систем iOS та Android з єдиною кодовою базою. Основне призначення застосунку – забезпечити користувачеві зручний доступ до даних моніторингу якості води з будь-якого місця, а також інформувати про критичні відхилення через push-сповіщення. На відміну від веб-панелі, яка передбачає роботу зі стаціонарного комп'ютера, мобільний застосунок орієнтований на використання поза домом, користувач може отримати сповіщення про погіршення якості води, навіть перебуваючи на роботі або у відрядженні.

Архітектура застосунку побудована навколо стекової навігації з бібліотеки React Navigation. До того, як користувач увійде в систему, він бачить лише екрани входу та реєстрації. Після успішної автентифікації з'являється головна частина застосунку з нижнім навігаційним меню, яке містить три основні розділи: «Моніторинг», «Історія» та «Аналітика». Така структура є типовою для сучасних мобільних додатків і забезпечує інтуїтивно зрозумілу навігацію

користувач завжди знає, де він знаходиться, і може миттєво перемикатися між розділами одним дотиком.

Перед початком роботи з застосунком користувач має автентифікуватися. Екран входу містить поля для введення електронної пошти та пароля, а також кнопку «Увійти». Для нових користувачів передбачено екран реєстрації з додатковим полем для повного імені. Автентифікація реалізована через Firebase Authentication – застосунок надсилає POST-запит до REST API Firebase, а у відповідь отримує токен, який надає доступ до захищених даних. Після успішного входу або реєстрації користувач автоматично потрапляє до головного екрану моніторингу. Дані про нових користувачів додатково зберігаються у вузлі users Realtime Database для можливої персоналізації в майбутньому, наприклад, для налаштування індивідуальних порогових значень або збереження історії переглядів. На екрані входу (рис. 3.13) передбачено обробку помилок, якщо користувач ввів неправильну пошту або пароль, застосунок показує відповідне повідомлення. Під час виконання запиту до сервера кнопка входу блокується та відображається індикатор завантаження, що запобігає повторним натисканням та покращує користувацький досвід. Екран реєстрації (рис. 3.14) перевіряє, чи заповнені всі поля, та повідомляє про помилки, якщо, наприклад, пароль занадто короткий або електронна пошта має невірний формат.

Важливою особливістю мобільного застосунку є система push-сповіщень, яка реалізована через бібліотеку expo-notifications. Під час першого запуску додаток запитує дозвіл на надсилання сповіщень, і у разі згоди реєструє пристрій у системі. Коли хук useDashboard отримує нові дані з Firebase, спеціальна функція перевіряє показники на відповідність нормам безпеки. Якщо виявлено відхилення, застосунок миттєво генерує локальне сповіщення з текстом попередження та викликає вібрацію пристрою. Сповіщення з'являється навіть тоді, коли застосунок згорнуто або екран телефону заблоковано, що гарантує, що користувач не пропустить критичну інформацію.

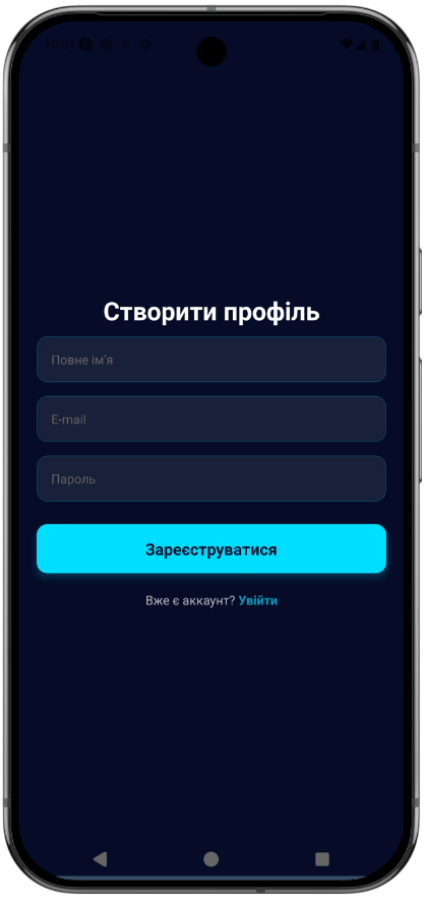



Рисунок 3.13 – Екран входу

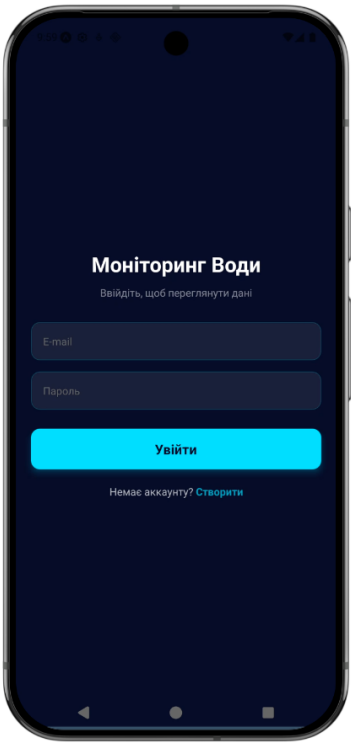



Рисунок 3.14 – Екран реєстрації

Після автентифікації користувач потрапляє на екран «Моніторинг», який є основним робочим екраном застосунку. Тут відображаються поточні показники якості води в режимі, наближеному до реального часу. Замість підписки на зміни через WebSockets, як це зроблено у веб-панелі, мобільний застосунок використовує періодичне опитування сервера кожні три секунди. Це рішення було обрано з кількох причин: по-перше, постійне WebSocket-з'єднання споживає більше енергії акумулятора, що критично для мобільних пристроїв; по-друге, спрощується керування з'єднанням – не потрібно обробляти розриви зв'язку та перепідключення; по-третє, три секунди є достатньо малим інтервалом, щоб оновлення виглядали плавними для ока, але не створювали надмірного навантаження на батарею. Якщо отримані дані відрізняються від попередніх, інтерфейс оновлюється лише при реальній зміні показників, що додатково економить ресурси.

У верхній частині екрана розташована велика картка із загальним статусом води «БЕЗПЕЧНО» (рис. 3.15) або «НЕБЕЗПЕЧНО» (рис. 3.16). Ця картка є візуально домінуючим елементом, оскільки саме вона дає миттєву оцінку ситуації. Колір картки змінюється залежно від показників: зелений, коли всі параметри в нормі, та червоний, коли є відхилення. Крім того, межа картки теж змінює колір, що створює ефект «світіння» і додатково привертає увагу. Під карткою статусу розташовані дві невеликі картки пліч-о-пліч для показника pH та мінералізації TDS, а під ними окрема картка для температури. Кожна картка містить назву параметра, числове значення та одиницю вимірювання. Якщо якийсь параметр виходить за межі норми, відповідне число підсвічується червоним кольором. Це дозволяє користувачеві миттєво визначити, який саме параметр викликає занепокоєння, не вчитуючись у всі цифри.

Важливою особливістю мобільного застосунку, яка відрізняє його від веб-панелі, є система push-сповіщень. Під час першого запуску додаток запитує дозвіл на надсилання сповіщень. Без цього дозволу сповіщення не працюватимуть, тому застосунок пояснює користувачеві, навіщо потрібен доступ. Коли нові дані

надходять з Firebase, спеціальна функція перевіряє показники на відповідність нормам. Якщо виявлено відхилення, застосунок миттєво генерує локальне сповіщення з текстом попередження, наприклад: «Увага! pH води перевищує норму» або «Високий рівень мінералізації!». Одночасно зі сповіщенням викликається вібрація пристрою, що гарантує, що користувач не пропустить важливе повідомлення, навіть якщо телефон перебуває в кишені або сумці. Сповіщення з'являються навіть тоді, коли застосунок згорнуто або телефон заблоковано – це дозволяє користувачеві дізнаватися про проблеми з якістю води, навіть не відкриваючи додаток.


Рисунок 3.15 – Екран моніторингу (безпечний стан)


Рисунок 3.16 – Екран моніторингу (небезпечний стан)

Екран «Історія» призначений для перегляду журналу всіх збережених вимірювань. Це важливий функціональний елемент, оскільки дозволяє відстежувати зміни якості води в часі та аналізувати, чи є певні закономірності наприклад, погіршення показників у вечірній час або після дощу. Логи у Firebase організовані за структурою logs/рік-місяць-день, тобто кожен день зберігається окремим об'єктом. Така організація є дуже зручною, оскільки дозволяє завантажувати дані порціями, спочатку лише за поточну дату, а потім, за потреби, додавати попередні дні. Це значно пришвидшує роботу застосунку,

особливо коли історія вимірювань налічує вже сотні або тисячі записів. Якщо записів за поточний день немає, то буде відображатися відповідне повідомлення (рис. 3.17).


Рисунок 3.17 – Екран історії вимірювань без даних

Користувач може оновити список звичайним жестом «потягнути вниз» - це стандартна поведінка для багатьох мобільних додатків, тому вона інтуїтивно зрозуміла. Під час оновлення відображається анімація завантаження, а після завершення список показує найактуальніші дані. Якщо користувач хоче побачити старіші записи, він може натиснути кнопку «Завантажити попередній

день» внизу списку. Кожне таке натискання додає до списку записи ще за один день у минуле. Така реалізація дозволяє переглядати архів без необхідності завантажувати одразу всі дані, що економить трафік, час та ресурси пристрою.

Кожен запис у журналі відображається у вигляді окремої картки (рис. 3.18). У верхній частині картки показано дату та час вимірювання, а також іконку щит із галочкою для безпечних вимірювань або знак оклику в трикутнику для небезпечних. Нижче розташовано три показники в рядок – pH, TDS та температура. Якщо вимірювання хоча б один показник вийшов за норму, вся картка має легкий червонуватий відтінок, а відповідні числа підсвічуються червоним кольором. Така візуалізація дозволяє швидко знайти проблемні моменти в історії: користувач може просто гортати список і звертати увагу на червоні картки, не вчитуючись у кожне число окремо.

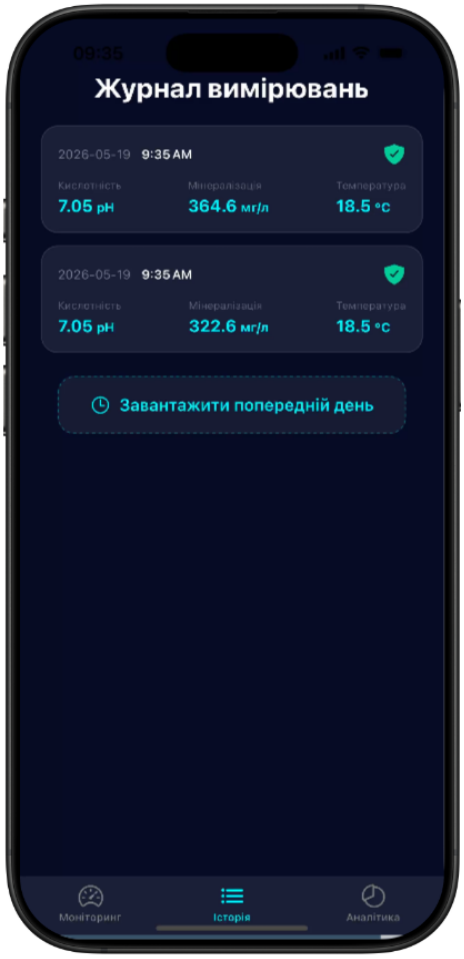



Рисунок 3.18 – Екран історії вимірювань з даними

Екран «Аналітика» призначений для оцінки динаміки зміни показників якості води. Якщо екран історії показує окремі вимірювання одне за одним, то аналітика дає змогу побачити загальну картину, чи покращується якість води, чи погіршується, або залишається стабільною. На основі останніх десяти вимірювань будуються компактні лінійні графіки для pH, TDS та температури. Саме десять вимірювань було обрано як оптимальне значення цього достатньо, щоб побачити тенденцію, але графік залишається читабельним на невеликому екрані смартфона.

Під кожним графіком розміщено три статистичні показники: мінімальне, середнє та максимальне значення (рис.3.19). Середнє значення виділено кольором, що відповідає лінії графіка, щоб привернути до нього увагу. Наприклад, для pH середнє значення буде зеленим, для TDS синім, для температури жовтогарячим. Ці статистичні дані дозволяють швидко оцінити, чи знаходяться показники в нормі в середньому, навіть якщо окремі вимірювання виходили за межі.


Рисунок 3.19 – Графік мінералізації

Візуалізація графіків виконана лаконічно та мінімалістично. Прибрано підписи осей, оскільки на маленькому екрані вони займають багато місця і створюють візуальний шум, прибрано точки даних, вони потрібні лише для точного аналізу, а для оцінки тренду достатньо лінії, залишено лише згладжену лінію, яка показує загальний напрямок змін. Користувач одразу бачить, чи лінія йде вгору, вниз або залишається горизонтальною. Кольори ліній узгоджено із загальною стилістикою застосунку, зелений для pH, синій для TDS, жовтогарячий для температури. Загальний вигляд сторінки аналітики зображений на рисунку 3.20.

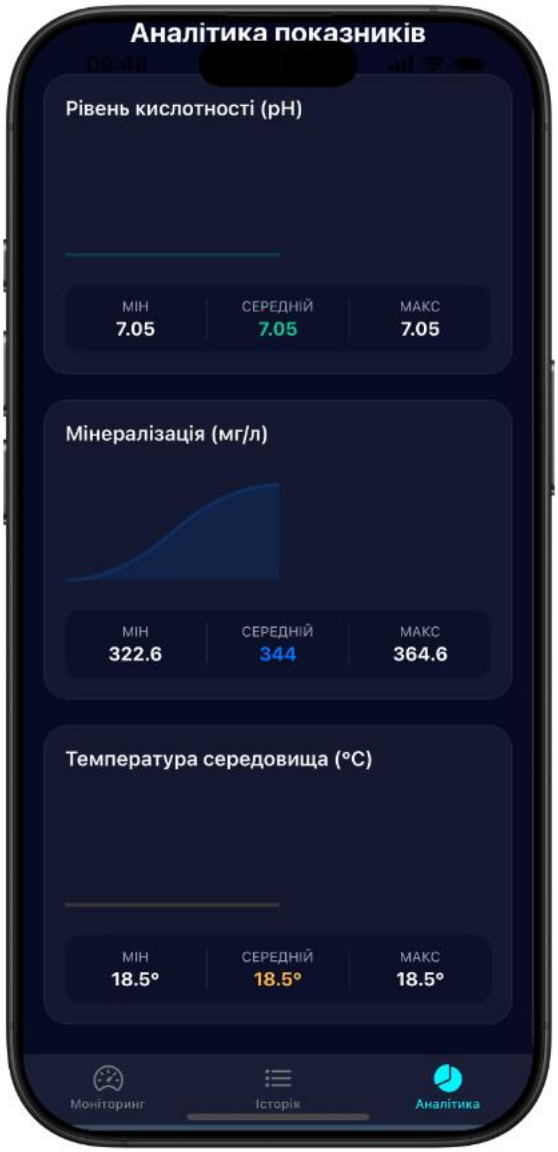



Рисунок 3.20 – Графіки всіх показників

Якщо даних ще недостатньо для побудови графіків наприклад, після першого запуску застосунку, екран показує відповідне повідомлення та індикатор завантаження. Переглянути цей екран можна на рисунку 3.21. Як тільки накопичується достатня кількість вимірювань, графіки з'являються автоматично.


Рисунок 3.21 – Сторінка аналітики без даних

Переміщення між трьома основними екранами здійснюється за допомогою нижнього навігаційного меню (рис. 3.22). Цей елемент інтерфейсу завжди видимий на всіх екранах захищеної частини застосунку. Активна вкладка підсвічується блакитним кольором, неактивні сірим. Кожна вкладка має іконку

та текстовий підпис: для «Моніторингу» іконка спідометра, для «Історії» список, для «Аналітики» кругова діаграма. Таке поєднання іконки та тексту робить навігацію зрозумілою навіть для нових користувачів, які вперше відкрили застосунок.

Уся стилізація мобільного застосунку виконана в єдиному стилі з веб-панеллю, що створює цілісне враження від програмно-технічного комплексу. Темно-синій фон є основним для всіх екранів. Картки мають напівпрозорий темно-синій колір з розмиттям, округлі кути та легку тінь, що створює ефект «скляної» поверхні. Текст виконано білим кольором для основних написів та сірим для другорядних. Акцентний блакитний колір використовується для активних елементів активної вкладки, посилань, деяких декоративних елементів. Незалежно від того, чи користувач переглядає дані з комп'ютера через веб-панель, чи з телефону через мобільний застосунок, візуальне оформлення залишається впізнаваним, що підкреслює єдність системи.

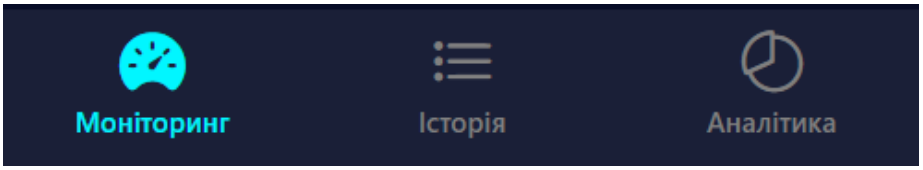



Рисунок 3.22 – Нижнє навігаційне меню

## 3.4. Висновки до розділу 3

У ході роботи над програмно-технічним комплексом моніторингу якості водопровідної води було виконано повний цикл розробки від вибору архітектури та компонентів до реалізації програмного забезпечення для всіх підсистем. У результаті створено діючий зразок системи, який успішно виконує поставлені завдання: автоматичне знімання показників, зберігання історії вимірювань, аналіз відхилень від норм та надання даних користувачеві через веб-панель і мобільний застосунок.

Апаратна підсистема, реалізована у симуляційному середовищі Wokwi, включає мікроконтролер ESP32, емулятори датчиків pH, TDS та DS18B20. Вибір ESP32 обґрунтований наявністю вбудованого Wi-Fi, достатньої кількості аналогових входів та підтримкою протоколу OneWire. Використання MicroPython замість традиційної C++ дозволило значно прискорити розробку завдяки інтерактивній консолі REPL, вбудованим бібліотекам для роботи з JSON та HTTP, а також лаконічності коду.

Програмне забезпечення для ESP32 реалізує алгоритм періодичного опитування датчиків та передачу результатів до Firebase Realtime Database через HTTP PUT та POST запити. Такий підхід дозволяє уникнути засмічення бази даних незначними коливаннями показників.

Серверна частина реалізована на Firebase Realtime Database, яка забезпечує зберігання даних у реальному часі, синхронізацію з усіма клієнтами та захист через правила безпеки. Структура бази даних включає вузол sensor_data для поточних вимірювань, вузол logs з організацією за датами для зберігання історії та вузол users для автентифікації користувачів. Використання серверних часових міток забезпечує точне датування подій незалежно від налаштувань годинника на пристроях.

Веб-панель на React підписується на зміни в базі даних через onValue та відображає поточні показники у вигляді кругових шкал для pH та TDS, а також стовпчикової діаграми для температури. Лінійні графіки з бібліотеки MUI X Charts показують історію змін кожного параметра за поточний день. Колірна індикація дозволяє миттєво оцінювати ситуацію [12].

Мобільний застосунок на React Native забезпечує ті ж функції з додатковою підтримкою автентифікації через Firebase Authentication, push-сповіщень та вібрації при відхиленнях від норми, а також посторінкового завантаження історії (спочатку за поточну дату, потім за попередні дні). Екран аналітики демонструє графіки останніх десяти вимірювань з агрегатними метриками.

Тестування системи проводилося як на рівні окремих модулів, перевірка функцій перетворення даних у calculations.py, тестування HTTP-запитів до Firebase, так і на рівні комплексної взаємодії всіх компонентів. У симуляційному середовищі Wokwi було перевірено коректну роботу ESP32 при зміні вхідних сигналів (регулювання потенціометрів), своєчасну передачу даних до Firebase, а також автоматичне оновлення веб-панелі та мобільного застосунку. Механізм порогових значень успішно відфільтровує незначні коливання, передаючи до бази даних лише суттєві зміни показників.

Усі підсистеми інтегровані в єдиний комплекс, який повністю відповідає вимогам технічного завдання.

# ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було розроблено програмно-технічний комплекс моніторингу якості водопровідної води в режимі реального часу. Система забезпечує автоматичне знімання показників з датчиків pH, загальної мінералізації TDS та температури, зберігання історії вимірювань у хмарній базі даних Firebase, аналіз відхилень від нормативних значень, формування звітів та надання даних користувачеві через веб-панель на React та мобільний застосунок на React Native.

У першому розділі проведено аналіз предметної області, розглянуто основні параметри якості води (pH, TDS, температура) та їх нормативні значення згідно з санітарними нормами. Проаналізовано існуючі аналоги систем моніторингу якості води, виявлено їх переваги та недоліки. На основі цього аналізу сформульовано вимоги до програмно-технічного комплексу, визначено функціональні та нефункціональні вимоги, а також обмеження проєкту. Результатом першого розділу стало чітке формулювання завдань, які необхідно вирішити при проєктуванні та реалізації системи.

У другому розділі проведено проєктування програмно-технічного комплексу. Виконано порівняльний аналіз трьох типів архітектур (монолітної, локальної клієнт-серверної та хмарно-орієнтованої) та обрано хмарно-орієнтовану архітектуру з використанням Firebase Realtime Database, що забезпечує глобальну доступність даних та синхронізацію в реальному часі. Обґрунтовано вибір мікроконтролера ESP32, аналогових датчиків pH та TDS, цифрового датчика температури DS18B20, а також мови програмування MicroPython замість традиційної C++. Спроектовано структуру бази даних Firebase з вузлами sensor_data для поточного стану та logs для історії вимірювань, розроблено правила безпеки та індекси для оптимізації запитів. Розроблено проєкт людино-машинного інтерфейсу, веб-панелі на React та мобільного застосунку на React Native з нижнім навігаційним меню на три розділи:

Моніторинг, Історія, Аналітика. Проведено аналіз методів комп'ютерної інженерії, обрано порогову фільтрацію сигналів датчиків, лінійне калібрування та серверні часові мітки для контролю цілісності даних. Результатом другого розділу стала повна проєктна документація, достатня для переходу до етапу програмної реалізації.

У третьому розділі здійснено програмну реалізацію всіх компонентів системи та їх тестування. Розроблено прошивку для ESP32 на MicroPython, яка реалізує підключення до Wi-Fi мережі, синхронізацію часу через NTP, періодичне опитування датчиків, порогову фільтрацію даних та передачу результатів до Firebase через HTTP PUT та POST запити. Розроблено веб-панель на React з використанням бібліотеки MUI X Charts для кругових шкал та лінійних графіків, яка підписується на зміни в базі даних та автоматично оновлює відображення. Розроблено мобільний застосунок на React Native з підтримкою автентифікації через Firebase Authentication, push-сповіщень та вібрації при виявленні відхилень, посторінкового завантаження історії та аналітики з агрегатними метриками. Виконано тестування системи в симуляційному середовищі Wokwi, перевірено коректність передачі даних до Firebase, роботу веб-панелі та мобільного застосунку, а також функціонування механізму сповіщень при виявленні відхилень. Результатом третього розділу став повністю функціонуючий програмно-технічний комплекс, готовий до розгортання.

Розроблена система має низку практичних переваг. Автоматизація процесу контролю якості води усуває потребу в ручних вимірюваннях, що економить час та виключає людські помилки. Цілодобовий моніторинг дозволяє виявити погіршення якості води одразу після його виникнення. Миттєві push-сповіщення на мобільний пристрій гарантують, що користувач дізнається про проблему навіть перебуваючи поза домом. Доступ до історії вимірювань та графіків трендів дозволяє аналізувати довгострокові зміни якості води.

Розроблений комплекс може бути використаний не лише для моніторингу водопровідної води, але й у суміжних галузях: для контролю якості води в

приватних свердловинах та колодязях; для моніторингу стану води в акваріумах, де pH та TDS є критичними для виживання гідробіонтів; для контролю якості води в системах зворотного осмосу; для навчальних цілей у закладах освіти при вивченні IoT-технологій та вбудованих систем.

Напрямками подальшого розвитку системи є: додавання додаткових датчиків; впровадження інтелектуальних алгоритмів для прогнозування погіршення якості води; інтеграція з системами «розумний дім» для автоматичного відключення води при критичних відхиленнях; фізична реалізація пристрою з реальними датчиками та виготовлення корпусу.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ


1. Про затвердження Державних санітарних норм та правил «Гігієнічні вимоги до води питної, призначеної для споживання людиною» : Наказ Міністерства охорони здоров'я України від 12.05.2022 р. № 452. URL: https://zakon.rada.gov.ua/go/z0723-22 (дата звернення: 22.03.2026).

2. ESP32 Series Datasheet. Espressif Systems. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (дата звернення: 15.02.2026).

3. MicroPython Documentation / D. P. George et al. URL: https://docs.micropython.org/en/latest/ (дата звернення: 20.02.2026).

4. Firebase Realtime Database Documentation. Google. URL: https://firebase.google.com/docs/database (дата звернення: 25.02.2026).

5. Firebase Authentication Documentation. Google. URL: https://firebase.google.com/docs/auth (дата звернення: 28.02.2026).

6. React Documentation. Meta Platforms. URL: https://react.dev/ (дата звернення: 02.03.2026).

7. React Native Documentation. Meta Platforms. URL: https://reactnative.dev/ (дата звернення: 05.03.2026).

8. Wokwi Simulator Documentation. Wokwi. URL: https://docs.wokwi.com/ (дата звернення: 10.03.2026).

9. DS18B20 Programmable Resolution 1-Wire Digital Thermometer Datasheet. Analog Devices. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/DS18B20.pdf (дата звернення: 12.03.2026).

10. Expo Snack Documentation. Expo. URL: https://docs.expo.dev/workflow/snack/ (дата звернення: 10.05.2026).

11. React Navigation Documentation. React Navigation. URL: https://reactnavigation.org/docs/getting-started/ (дата звернення: 15.03.2026).

12. MUI X Charts Documentation. MUI. URL: https://mui.com/x/react-charts/ (дата звернення: 18.03.2026).

13. React Native Chart Kit. GitHub. URL: https://github.com/indiespirit/react-native-chart-kit (дата звернення: 20.03.2026).

14. Plauska I., Liutkevičius A., Janavičiūtė A. Performance Evaluation of C/C++, MicroPython, Rust and TinyGo Programming Languages on ESP32 Microcontroller. *Electronics*. 2023. Vol. 12, No 1. URL: https://doi.org/10.3390/electronics12010143 (дата звернення: 25.03.2026).

15. An IoT-Based System for Water Quality Monitoring and Notification System of Aquaculture Prawn Pond. 2022 *International Conference on ICT for Smart Society (ICISS)*. 2022. URL: https://ieeexplore.ieee.org/document/9994388 (дата звернення: 30.03.2026).

16. Hasib A., Akib A. S. M. A. S., Giri A. HydroSense: A Dual-Microcontroller IoT Framework for Real-Time Multi-Parameter Water Quality Monitoring with Edge Processing and Cloud Analytics. 2025. URL: https://arxiv.org/abs/2601.21595 (дата звернення: 01.04.2026).

17. An Optimal Internet of Things-Driven Intelligent Decision-Making System for Real-Time Fishpond Water Quality Monitoring and Species Survival. *Sensors*. 2024. Vol. 24, No 23. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC11644897/ (дата звернення: 03.04.2026).

18. Hasib A., Akib A. S. M. A. S. Cloud-Enabled IoT System for Real-Time Environmental Monitoring and Remote Device Control Using Firebase. 2026. URL: https://arxiv.org/html/2601.17414v1 (дата звернення: 05.04.2026).

19. Dewi Puspa Ayu Lestari. Rancang bangun sistem monitoring pengolahan air menggunakan ESP32 berbasis Android : дис. ... бакалавра / Politeknik Negeri Jakarta, 2024. URL: https://repository.pnj.ac.id/id/eprint/21306/ (дата звернення: 09.04.2026).

20. Saputra G. A. Sistem Kontrol dan Monitoring PH Air dan Kekeruhan Air Akuarium Ikan Komet Berbasis IoT dan ESP32 : дис. ... бакалавра / Politeknik Negeri Bali, 2025. URL: https://repository.pnb.ac.id/id/eprint/18724/ (дата звернення: 11.04.2026).

21. Compact Smart Water Meter Development for Smart City. 2023 *International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation* (ICAMIMIA). 2023. URL: https://ieeexplore.ieee.org/document/10420528 (дата звернення: 13.04.2026).

22. Saputra I. G. A. Sistem Kontrol dan Monitoring Budidaya Ikan Nila Berbasis Mikrokontroler ESP32 dan PLC Omron CP1E : дис. ... бакалавра / Politeknik Negeri Bali, 2025. URL: https://repository.pnb.ac.id/id/eprint/17648/ (дата звернення: 15.04.2026).

23. Ahmad A., Irfani A. Sistem Kontrol dan Monitoring Kualitas Air pada Air Kolong Bekas Tambang Timah для Budidaya Ikan Air Tawar Berbasis IoT : дис. ... бакалавра / Politeknik Manufaktur Negeri Bangka Belitung, 2025. URL: http://repository.polman-babel.ac.id/id/eprint/2048/ (дата звернення: 17.04.2026).

24. pH Meter using Arduino. Circuit Digest. URL: https://circuitdigest.com/microcontroller-projects/arduino-ph-meter (дата звернення: 19.03.2026).

25. TDS Sensor with ESP32. Random Nerd Tutorials. URL: https://randomnerdtutorials.com/esp32-tds-water-quality-sensor/ (дата звернення: 19.03.2026).

26. DS18B20 Temperature Sensor with ESP32. Random Nerd Tutorials. URL: https://randomnerdtutorials.com/micropython-ds18b20-esp32-esp8266/ (дата звернення: 19.04.2026).

27. ESP32 MicroPython Tutorial. Random Nerd Tutorials. URL: https://randomnerdtutorials.com/micropython-esp32-esp8266/ (дата звернення: 20.04.2026).

28. Sending Sensor Data to Firebase Realtime Database with ESP32. Random Nerd Tutorials. URL: https://randomnerdtutorials.com/esp32-firebase-realtime-database/ (дата звернення: 10.04.2026).

29. ESP32 with Firebase Web App. Firebase. URL: https://firebase.google.com/learn/ (дата звернення: 10.04.2026)

30. Mahmoud A. A., Aljumaily A. W. Real-Time Water Quality Monitoring System Using IoT and Firebase Cloud. *International Journal of Interactive Mobile Technologies*. 2023. Vol. 17, No 12. P. 45–58. URL: https://www.researchgate.net/publication/372507886_Water_Quality_Monitoring_System (дата звернення: 10.05.2026).

31. Singh R., Baz M., Gehlot A. Water Quality Monitoring Using IoT and Machine Learning: A Review. *2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*. 2023. P. 345–350. URL: https://ieeexplore.ieee.org/document/10175322 (дата звернення: 10.05.2026).

32. Firebase Cloud Messaging Documentation. Google. URL: https://firebase.google.com/docs/cloud-messaging (дата звернення: 10.05.2026).

33. Google Cloud IoT Core Documentation. Google. URL: https://cloud.google.com/iot/docs (дата звернення: 10.05.2026).

34. JavaScript Chart Library for Developers – Chart.js. Chart.js. URL: https://www.chartjs.org/ (дата звернення: 10.05.2026).

35. Recharts Documentation. Recharts. URL: https://recharts.org/ (дата звернення: 01.05.2026).

36. OneWire Protocol Specification. Analog Devices / Maxim Integrated. URL: https://www.analog.com/en/technical-articles/1wire-communication-through-software.html (дата звернення: 02.05.2026).

37. Expo Notifications Documentation. Expo. URL: https://docs.expo.dev/versions/latest/sdk/notifications/ (дата звернення: 03.05.2026).

38. React Native Firebase Documentation. Invertase. URL: https://rnfirebase.io/ (дата звернення: 10.05.2026).

39. Async Storage for React Native. React Native Community. URL: https://react-native-async-storage.github.io/async-storage/ (дата звернення: 03.05.2026).

40. TDS Sensor V1.0 for Arduino – Technical Datasheet. DFRobot. URL: https://wiki.dfrobot.com/TDS_Sensor_V1.0_SKU_SEN0244 (дата звернення: 02.05.2026).

41. pH Sensor V2.0 for Arduino – Technical Datasheet. DFRobot. URL: https://wiki.dfrobot.com/pH_meter_V2_SKU_SEN0161 (дата звернення: 01.05.2026).

42. NodeMCU ESP32 Pinout Reference. Last Minute Engineers. URL: https://lastminuteengineers.com/esp32-pinout-reference/ (дата звернення: 09.05.2026).

43. MQTT Protocol Specification Version 5.0. OASIS Standard. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html (дата звернення: 11.05.2026).

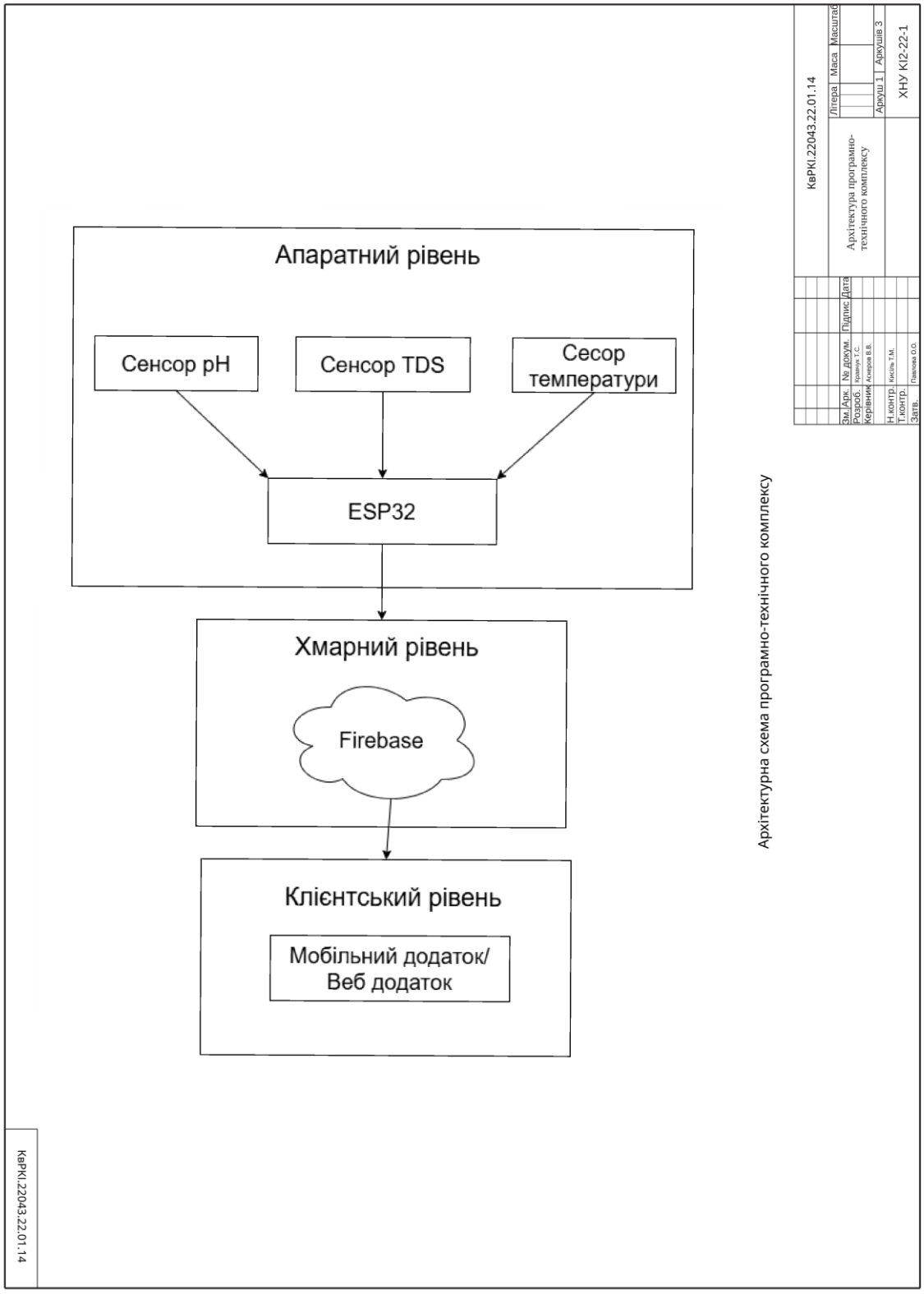
44. Войтко В. В., Ковальчук А. М. Інтелектуальні системи моніторингу якості води на базі IoT-технологій. *Вісник Вінницького політехнічного інституту*. 2024. № 3. С. 22–29. URL: https://visnyk.vntu.edu.ua/index.php/visnyk/article/view/2456 (дата звернення: 12.05.2026).

45. Петренко О. І., Савченко В. П. Застосування Firebase для зберігання та обробки даних в IoT-системах реального часу. *Комп'ютерні науки та кібербезпека*. 2025. № 2. URL: https://kpi.ua/ (дата звернення: 11.02.2026).

# ДОДАТОК А

(обов’язковий)

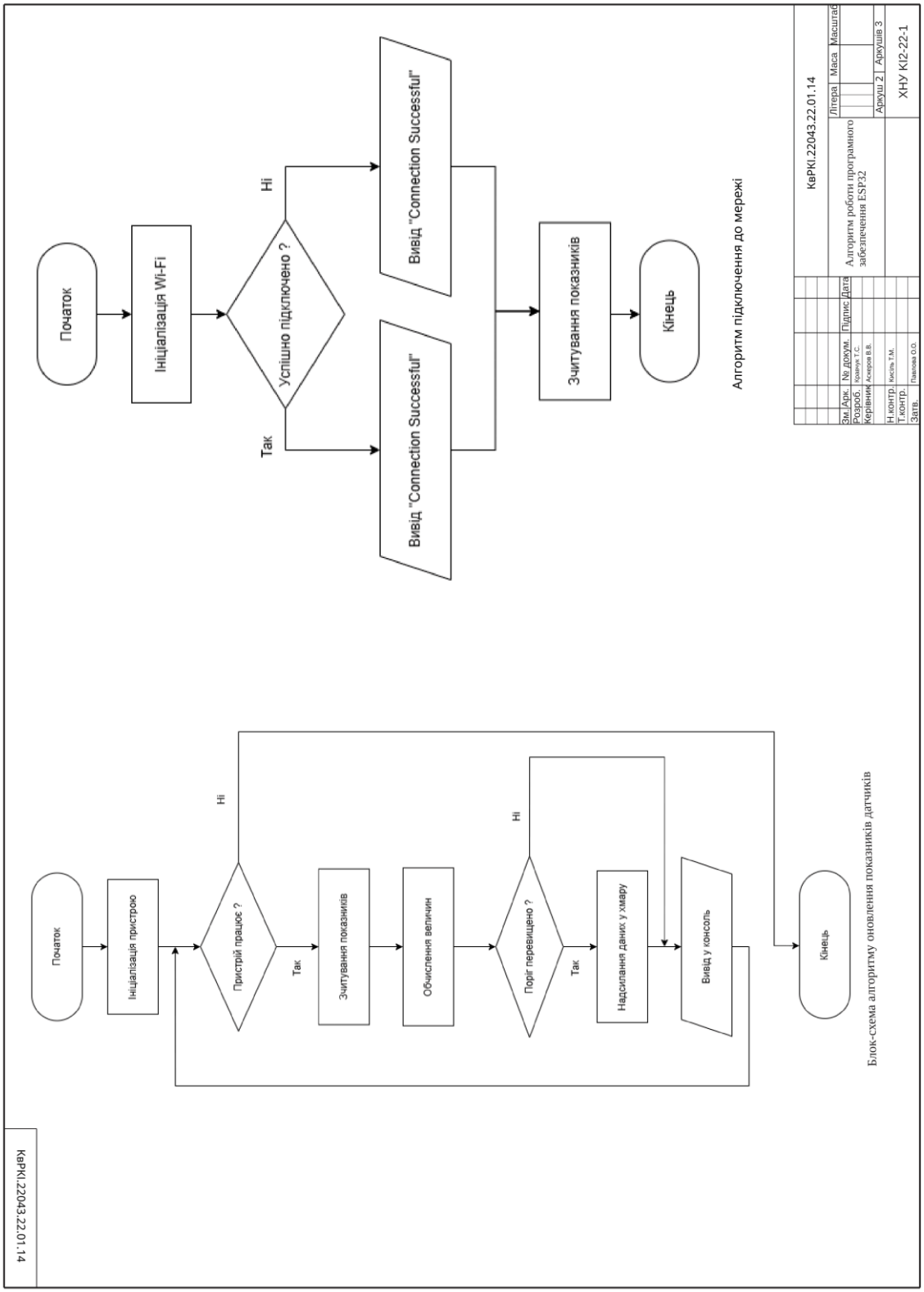
Копія креслення «Архітектура програмно-технічного комплексу»



# ДОДАТОК Б

(обов’язковий)

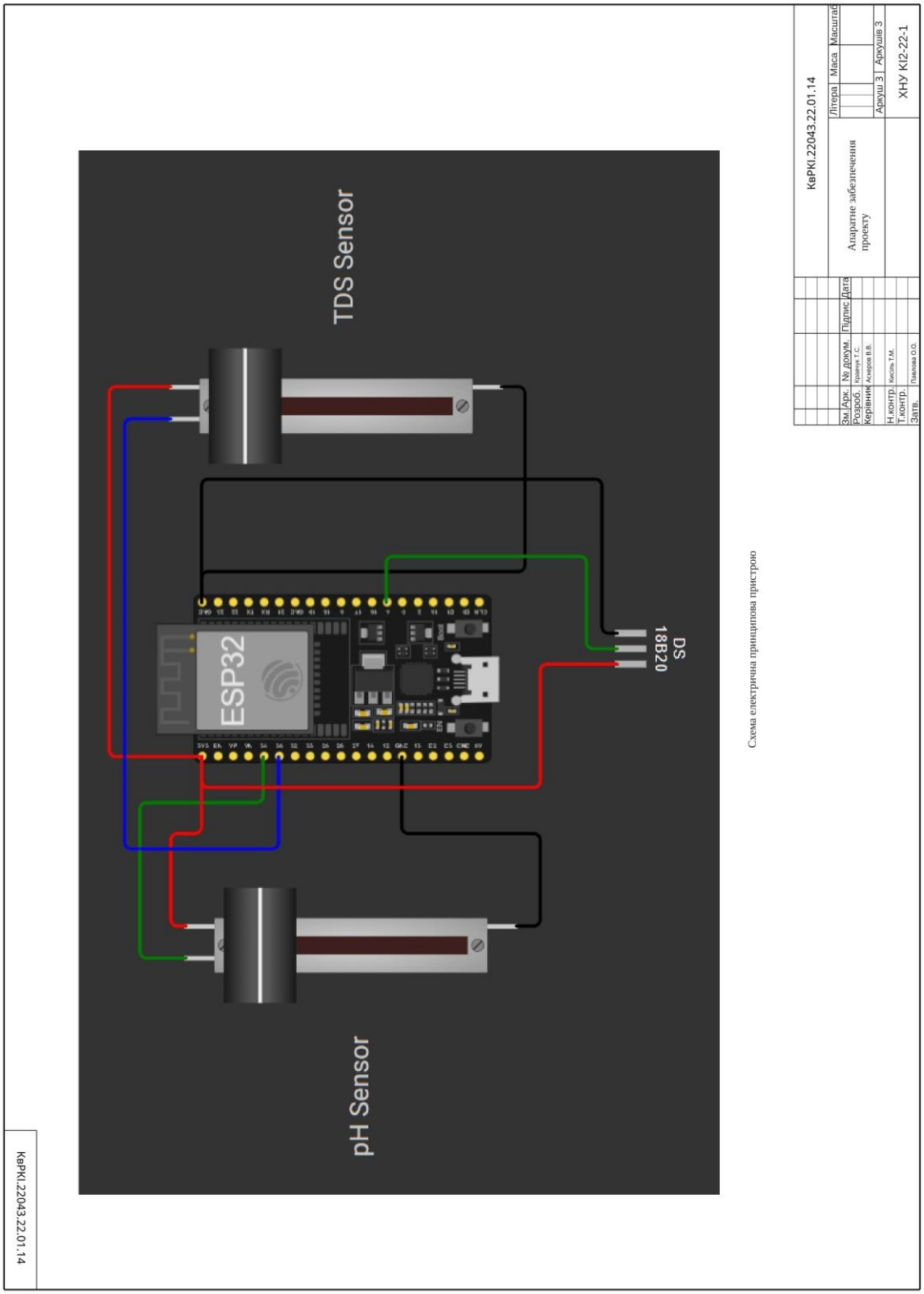
Копія креслення «Алгоритм роботи ПЗ ESP32»



**ДОДАТОК В**

(обов’язковий)

Копія креслення «Апаратне забезпечення проєкту»



**Протокол аналізу звіту подібності експертом**

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

**Автор:** Тарас КРАВЧУК

**Співавтор:**

**Назва:** Програмно-технічний комплекс моніторингу якості водопровідної води в режимі реального часу

**Експерт:** В'ячеслав АСКЕРОВ

**Підрозділ:** Кафедра комп'ютерної інженерії та інформаційних систем

**Коефіцієнт подібності 1:**2.32%

**Коефіцієнт подібності 2:**0.48%

**Мікропробіли:** 300

**Заміна букв:** 83

**Інтервали:** 0

**Білі знаки:** 0

**Дата створення звіту:** 2026-05-27 19:48:47.0

**Після аналізу Звіту подібності констатую наступне:**

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-05-28 Доцент Андрій Нічепорук

*Дата* експерт

Thu May 28 08:34:19 EEST 2026, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

# Anti-Plagiarism (http://ap.km.ua) v-15.701

**Максимальне співпадіння з одним документом 1.0%**

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 13%**

| | Документ | | Сумарний збіг по Базі Даних | |
|---|---|---|---|---|
| ID: 272532<br>Назва: БКР Програмно-технічний комплекс моніторингу якості водопровідної води в режимі реального часу<br>Додано в БД: 2026-05-28<br>Автора: Тарас КРАВЧУК<br>Керівники: В’ячеслав АСКЕРОВ<br>Консультанти:<br>Опоненти: | Символи | Лексеми | Символи | Лексеми |
| | 98693 | 715 | 2194 (2%) | 30 (4%) |

Джерело плагіату

| ID | Опис | Наявність плагіату в документі | |
|---|---|---|---|
| | | Символи | Лексеми |

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

# РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Кравчук Тарас Сергійович

Тема: Програмно-технічний комплекс моніторингу якості водопровідної води в режимі реального часу

Спеціальність**:** 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___75___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є розробка програмно-технічного комплексу для автоматизованого моніторингу якості водопровідної води в режимі реального часу. Було спроектовано апаратну частину на базі ESP32, створено серверну частину на платформі Firebase, а також розроблено веб-панель та мобільний застосунок.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області, розглянуто основні параметри якості води (Ph, TDS, Температура) та їх нормативні значення, а також сформульовано вимоги до програмно-технічного комплексу. В другому розділі кваліфікаційної роботи проведено проєктування програмно-технічного комплексу. Проведено порівняльний аналіз архітектурних підходів та обрано хмарно-орієнтовану архітектуру з використанням Firebase. Обґрунтовано вибір мікроконтролера ESP32, датчиків, а також мови програмування MicroPython. Спроектовано структуру бази даних, розроблено алгоритм роботи мікроконтролера. В третьому розділі кваліфікаційної роботи виконано програмно-апаратну реалізацію та тестування системи. Розроблено прошивку для ESP32 на MicroPython з функціями підключення до Wi-Fi, синхронізації часу через NTP, опитування датчиків та передачі даних до Firebase. Реалізовано веб-панель на React. Розроблено мобільний

застосунок на React Native з підтримкою автентифікації через Firebase Authentication. Проведено тестування системи в середовищі Wokwi, підтверджено працездатність усіх компонентів.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: апаратна частина реалізована в середовищі симуляції .

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Праворська Наталія Іванівна, доцент кафедри ТПЗ

"01" червня 2026 р. _____ (підпис)

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Тарас КРАВЧУК

ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-22-1

# ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

ТКеер

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Програмно-технічний комплекс моніторингу якості водопровідної води в режимі реального часу
Автор Тарас КРАВЧУК
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти перший (бакалаврський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: В'ячеслав АСКЕРОВ

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

| № | Висновок | Позначка про відповідність |
|---|---|---|
| 1 | Ознаки академічного плагіату | |
| 1.1 | Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту. | відповідає |
| 1.2 | Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. | |
| 1.3 | Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат. | |
| 1.4 | Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту. | |
| 2 | Інші види порушень академічної доброчесності | |

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 2,32%; та системою Anti-Plagiarism складає 1.0%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

01.06.2026

Завідувач кафедри _____ Підпис Ольга ПАВЛОВА Ім'я, ПРІЗВИЩЕ

Гарант освітньої програми _____ Підпис Андрій НІЧЕПОРУК Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Підпис В'ячеслав АСКЕРОВ Ім'я, ПРІЗВИЩЕ