

Міністерство освіти і науки України
Державний заклад
«Луганський національний університет імені Тараса Шевченка»

Навчально-науковий інститут математики та інформаційних технологій

Кафедра інформаційних технологій та систем

Коробка Олег Сергійович

**РОЗРОБКА БАЗИ ДАНИХ ДЛЯ МОНІТОРИНГУ І ВІДСТЕЖЕННЯ
ЕКОЛОГІЧНИХ ПОКАЗНИКІВ МІСТА**

**кваліфікаційна робота
здобувача вищої освіти другого (магістерського) рівня
освітньої програми «Комп'ютерні мережі»
за спеціальністю 123 Комп'ютерна інженерія**

Особистий підпис _____ Олег КОРОБКА

Науковий керівник _____, Микола СЕМЕНОВ,
кандидат педагогічних наук, доцент
кафедри інформаційних технологій
та систем

Завідувач кафедри _____ Микола СЕМЕНОВ,
кандидат педагогічних наук, доцент
кафедри інформаційних технологій
та систем

АНОТАЦІЯ

Тема: Розробка бази даних для моніторингу і відстеження екологічних показників міста.

Спеціальність: 123 «Комп'ютерна інженерія».

Установа: ЛНУ імені Тараса Шевченка, 2024р.

Магістерська робота містить: 75 с., 24 рис., 4 табл., 46 джерел.

Об'єкт дослідження – база даних моніторингу та відстеження екологічних показників міста.

Предмет дослідження – проєктування бази даних моніторингу та відстеження екологічних показників міста.

Мета роботи – розробити базу даних моніторингу та відстеження екологічних показників міста, яка зберігатиме дані про стан навколишнього середовища міста, зібрані з різних джерел у структурованому вигляді з можливістю обробки та аналізу даних.

Результати роботи – в роботі проаналізовано наявні бази даних моніторингу екологічних показників міста. За підсумками аналізу визначено вимоги та розроблено структуру, алгоритми збирання, зберігання й обробки інформації у базі даних, яка буде зберігати дані про екологічні показники міста. Розроблено інтерфейс користувача для доступу до бази даних.

Ключові слова: ЕКОЛОГІЧНІ ПОКАЗНИКИ, БАЗА ДАНИХ, МОНІТОРИНГ, АНАЛІЗ ДАНИХ, ДАТЧИК, ЕКОТЕХНОЛОГІЇ, ІНФОРМАЦІЙНА СИСТЕМА.

ANNOTATION

Topic: Development of a database for monitoring and tracking ecological indicators of a city.

Speciality: 123 "Computer Engineering".

Institution: Luhansk Taras Shevchenko National University (LTSNU), 2024 year.

Master's thesis consists of: 75 p., 24 im., 4 tables, 46 sources.

Object of research – the database for monitoring and tracking ecological indicators of a city.

Subject of research – developing a database for monitoring and tracking ecological indicators of a city.

Objective of the study – to develop a database for monitoring and tracking ecological indicators of a city that will store data on the city's environmental status gathered from various sources in a structured format, enabling data processing and analysis.

Results of the study - analysis of existing databases monitoring the ecological indicators of a city has been conducted. Based on the analysis, requirements and the structure have been developed, along with algorithms for collecting, storing, and processing information within the database, which will contain data on the city's ecological indicators. A user interface has been developed for accessing the database.

Keywords: ECOLOGICAL INDICATORS, DATABASE, MONITORING, DATA ANALYSIS, SENSOR, ECOTECHNOLOGIES, INFORMATION SYSTEM.

ЗМІСТ

ВСТУП	5
РОЗДІЛ 1. СИСТЕМИ МОНІТОРИНГУ ЕКОЛОГІЧНИХ ПОКАЗНИКІВ	9
1.1. Системи контролю повітряного середовища	9
1.2. Системи контролю якості води	11
1.3. Системи збору даних про стан ґрунту	13
1.4. Моніторинг шумового забруднення	14
1.5. Інтегровані системи моніторингу.....	16
РОЗДІЛ 2. РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ЕКОЛОГІЧНИХ ПОКАЗНИКІВ МІСТА	20
2.1. Побудова бази даних та визначення архітектури системи	20
2.2. Інтеграція датчиків у систему.....	25
2.3. Розробка програмного забезпечення для оперування базою даних	44
2.4. Забезпечення захисту від втрати даних	54
РОЗДІЛ 3. АНАЛІЗ ТА ТЕСТУВАННЯ СИСТЕМИ	61
3.1. Визначення критеріїв тестування.....	61
3.2. Дослідження ефективності та надійності системи	64
ВИСНОВКИ.....	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	71

ВСТУП

Урбанізація є одним з найпотужніших процесів, які впливають на економічний, соціальний і екологічний розвиток людства. За даними ООН, у 2018 році 55% населення планети проживало у міських поселеннях, а до 2050 року ця частка зросте до 68% [12]. Міста є центрами інновацій, культури і освіти, а також рушіями економічного зростання. Однак, міста також стикаються з численними викликами, пов'язаними з негативним впливом урбанізації на навколишнє середовище. Забруднення повітря, води і ґрунту, виснаження природних ресурсів, втрата біорізноманіття, зміна клімату, погіршення здоров'я та якості життя мешканців міст – це лише деякі з проблем, які потребують ефективного управління екологічними аспектами урбанізованих територій.

Ефективне управління екологічними аспектами урбанізованих територій стає важливою складовою сталого розвитку міст, який передбачає гармонійне поєднання економічного прогресу, соціальної справедливості і охорони довкілля. Для досягнення цієї мети необхідно мати достовірну, актуальну і доступну інформацію про стан екології міста, яка дозволить виявляти проблеми, аналізувати їх причини та наслідки, визначати пріоритети та цілі, розробляти та реалізовувати заходи щодо їх вирішення та відстежувати їх ефективність. Одним з ключових факторів для забезпечення такої інформації є комплексне використання баз даних про екологію міста.

Бази даних екологічних показників міста – це сукупність структурованих даних, які відображають різні аспекти взаємодії міста з навколишнім середовищем, такі як якість повітря, води, ґрунту, рівень шуму, викиди забруднюючих речовин, виробництво та утилізація відходів, споживання енергії та ресурсів, рослинний та тваринний світ, зелені зони, кліматичні параметри тощо. Бази даних про екологію міста можуть бути створені та підтримувані різними суб'єктами, такими як державні органи, наукові установи, громадські організації, приватні компанії, медіа тощо. Вони можуть мати різні формати, рівні деталізації, частоту оновлення, способи

збору, обробки, зберігання, аналізу та візуалізації даних та бути використані для різних цілей, таких як моніторинг, дослідження, освіта, підтримка прийняття рішень, інформування, контроль тощо.

Для забезпечення ефективного моніторингу екології міста необхідно мати надійну, актуальну та доступну базу даних, яка зберігає, обробляє та аналізує інформацію про стан довкілля у структурованому вигляді. База даних повинна відповідати таким критеріям:

- Повнота, тобто включати всі необхідні параметри та показники, які характеризують екологію міста.
- Точність, тобто відображати реальний стан довкілля з врахуванням похибок та невизначеностей вимірювань.
- Актуальність, тобто відповідати сучасному стану довкілля та оновлюватися в режимі реального часу або з заданою періодичністю.
- Доступність, тобто бути відкритою та зручною для різних категорій користувачів, таких як органи державного управління, наукові установи, громадські організації, ЗМІ, громадяни тощо.
- Зрозумілість, тобто мати чітку та зрозумілу структуру, номенклатуру, формат, метадані, документацію та інтерфейс для роботи з даними.
- Корисність, тобто мати можливість використання даних для різних цілей, таких як моніторинг, дослідження, освіта, підтримка прийняття рішень, інформування, контроль, участь, захист прав та інтересів тощо.

Використання бази даних для моніторингу і відстеження екологічних показників міста сприяє підвищенню рівня усвідомленості, відповідальності, прозорості та ефективності управління екологічними аспектами урбанізованих територій [14]. Тому використання такої бази даних може стати ключовим фактором для розуміння та покращення якості навколишнього середовища та підтримки сталого розвитку міста.

Об'єкт дослідження – база даних моніторингу та відстеження екологічних показників міста.

Предмет дослідження – проєктування бази даних моніторингу та відстеження екологічних показників міста.

Мета роботи – розробити базу даних моніторингу та відстеження екологічних показників міста, яка зберігатиме дані про стан навколишнього середовища міста, зібрані з різних джерел у структурованому вигляді з можливістю обробки та аналізу даних.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- 1) провести огляд та проаналізувати наявні бази даних, що використовуються для моніторингу екологічних показників у містах.
- 2) Визначити вимоги до функціональності та можливостей бази даних для ефективного моніторингу та відстеження екологічних показників міста.
- 3) розробити структуру бази даних, включаючи визначення сутностей, зв'язків і атрибутів, які відображатимуть екологічні дані міста.
- 4) розробити алгоритми для збору, зберігання, обробки та аналізу отриманих даних про екологічні показники.
- 5) розробити інтерфейс користувача, який забезпечить зовнішній доступ до інформації, збереженої в базі даних моніторингу екологічних показників міста.

Новизна результатів дослідження полягає в застосуванні такого підходу до розробки бази даних, який об'єднує методи моніторингу та відстеження екологічних показників міста з різних джерел у структуровану систему з обробкою та аналізом даних. Цей підхід відкриває можливості для комплексного аналізу та використання екологічних даних міста з метою удосконалення стратегій управління середовищем та прийняття більш обґрунтованих рішень у сфері екології та розвитку території громади.

Методи дослідження. У даній роботі були використані такі методи: аналітичний метод – для ознайомлення з теоретичними основами, аналізу наявних баз даних та формування вимог до бази моніторингу екологічних

показників міста; метод моделювання – для розробки структури, алгоритмів обробки даних та створення інтерфейсу користувача бази; метод експерименту – для апробації та впровадження розробленої бази даних у практику.

У першому розділі проведено огляд систем моніторингу контролю екологічних показників міста, зокрема, засобів контролю параметрів повітря, води, ґрунту та шумового забруднення.

У другому розділі описано процес побудови бази даних та визначення архітектури системи моніторингу екологічних показників міста. Реалізовано фізичне підключення та інтеграцію датчиків у програмне забезпечення, описано програмну імплементацію модулів для зчитування й аналізу даних, представлено систему комплексного захисту системи від втрати даних.

У третьому розділі визначено критерії для аналізу й тестування побудованої системи і виконано її дослідження на основі обраних критеріїв. Зроблено висновки про ефективність системи та доцільність її практичного використання.

РОЗДІЛ 1. СИСТЕМИ МОНІТОРИНГУ ЕКОЛОГІЧНИХ ПОКАЗНИКІВ

Здійснення ефективного моніторингу екологічних показників у місті є важливим для забезпечення сталого розвитку та здоров'я мешканців. Наразі існують різноманітні системи, спрямовані на вимірювання та аналіз екологічних параметрів, які дозволяють здійснювати контроль за станом довкілля у міській зоні. Для подальшої розробки комплексної та ефективної бази даних моніторингу екологічних показників міста необхідно розглянути вже наявні системи моніторингу, що надасть уявлення про різноманітність методів, їхню ефективність та існуючі обмеження.

1.1. Системи контролю повітряного середовища

Моніторинг рівня забруднення повітря в конкретних містах та регіонах здійснюють локальні системи контролю повітряного середовища. Ці системи використовують різноманітні датчики та сенсори, щоб надати детальну інформацію про якість повітря в місцях, де може бути підвищений ризик для здоров'я або довкілля. Для збору даних про склад та рівень забруднення атмосфери використовують такі технології [3, 24, 36]:

1. Датчики газів:

- Електрохімічні сенсори: Вони вимірюють рівні окремих газів, таких як оксиди азоту (NO_x), оксиди сірки (SO_x), вуглеводні та інші. Ці сенсори досить поширені та використовуються для моніторингу на вулицях та в промислових районах.

- Фотоіонізаційні детектори (PID): Вони здатні виявляти різноманітні леткі органічні сполуки, такі як розчинники та летючі органічні речовини (VOC). Ці датчики широко використовуються для виявлення потенційно небезпечних речовин у повітрі.

2. Лазерні датчики: використовуються для вимірювання конкретних газів за допомогою лазерного променя, що поглинається цими газами. Наприклад, лазерні датчики CO₂.

3. Датчики твердих часток: лазерні дифузійні датчики PM: Вони вимірюють рівень твердих часток в повітрі (PM_{2.5} та PM₁₀) за допомогою лазерної дифракції або розсіяння світла.

4. Метеорологічні станції. Ці станції не лише вимірюють рівні забруднень, але й збирають метеорологічні дані, такі як температура, вологість, напрямок та швидкість вітру, що дозволяє краще розуміти розподіл забруднень у просторі та часі.

5. Інтернет речей (IoT) та смарт-технології: розташування сенсорів на об'єктах, будівлях або вулицях міста для моніторингу рівня забруднення та зв'язку з центральною системою для збору та аналізу даних.

Локальні системи моніторингу встановлюються у найбільш чутливих або потенційно небезпечних зонах, таких як індустриальні райони, місця з високим рівнем автомобільного руху, а також поруч з фабриками або заводами, що виділяють в атмосферу шкідливі речовини. Вони можуть бути розташовані на будівлях, дорогах, в парках або інших громадських місцях [29].

Ці системи дозволяють місцевим органам та населенню миттєво отримувати дані про рівень забруднення повітря. Інформація, отримана від цих систем, може використовуватися для вжиття заходів захисту здоров'я населення, прийняття екологічно орієнтованих рішень у місцевому масштабі та спрямування належних ресурсів на вирішення проблем екології в конкретних місцевостях.

Одним з прикладів сучасного проєкта, де встановлюються сенсори для моніторингу рівня забруднення повітря в місті, є **Airly** – польська компанія, яка розробляє та виробляє інтелектуальні датчики якості повітря, що вимірюють концентрацію частинок PM_{2.5}, PM₁₀, NO₂, O₃, SO₂, CO та інших забруднювачів. Датчики Airly встановлюються на вуличних ліхтарях, дахах будинків, фасадах магазинів та інших місцях, опісля чого дані передаються

через мобільний інтернет на хмарну платформу Airly, де вони обробляються та аналізуються. Користувачі можуть отримувати доступ до даних про якість повітря через вебсайт, мобільний додаток або API Airly. Крім того, Airly надає прогнози та рекомендації щодо покращення якості повітря, а також інтегрується з різними сервісами, такими як Google Assistant, Amazon Alexa, IFTTT, Twitter тощо.

1.2. Системи контролю якості води

Системи контролю якості води в містах відіграють ключову роль у забезпеченні безпечного та доступного водопостачання для споживачів. Ці системи відстежують різноманітні параметри води, від хімічного складу до мікробіологічних показників, щоб забезпечити відповідність стандартам якості та безпеки. Ось деякі з головних технологій та методів, які застосовуються в сучасних системах контролю якості води:

1. Сенсорні технології. Передбачають використання сенсорів та датчиків для безпосереднього вимірювання параметрів води. Вони забезпечують швидке та точне збирання даних, що дозволяє вчасно реагувати на забруднення та зберігати природні водні ресурси. Ось деякі з найпоширеніших та передових сенсорів та датчиків [24, 27]:

- рН-метри. Датчики рН вимірюють концентрацію воднісної іонізації та визначають кислотність чи лужність води. Вони важливі для визначення потенційно шкідливих рівнів рН у водних системах.
- Датчики розчиненого кисню. Вимірюють кількість розчиненого кисню у воді, що дозволяє оцінювати оксигенацію водойм, необхідну для підтримки життя риб і інших організмів.
- Датчики температури. Вимірюють температуру води і є важливими для встановлення відповідних температурних умов для різних екосистем та забезпечення оптимальних умов для життя водних організмів.

- Турбідиметри. Вимірюють ступінь турбідності води, що вказує на наявність часток та забруднень у воді, що може бути ключовим показником її якості.
- Електропровідність (ЕС). Датчики ЕС вимірюють провідність води, що дозволяє оцінити наявність розчинених солей та мінералів.
- Датчики рівня води. Використовуються для вимірювання рівня води в водоймах та ставках, дозволяючи вчасно реагувати на можливі зміни та уникнути ризику затоплення.

2. Хроматографія. Дозволяє розділяти та аналізувати різні речовини у воді. Використання хроматографічних систем дозволяє визначати концентрацію хімічних сполук, таких як різні типи забруднювачів або хімічні елементи.

3. Спектроскопія. Цей метод базується на використанні спектральних характеристик речовин у воді для їх ідентифікації та аналізу. Інфрачервона спектроскопія, ультрафіолетова та видима спектроскопія використовуються для визначення різних складових речовин у воді.

4. Мас-спектрометрія. Дозволяє визначати хімічний склад води на основі маси та структури молекул, що дозволяє виявляти й аналізувати різноманітні хімічні сполуки.

5. Мікробіологічні тести. Включають в себе методи оцінки мікробного складу води, виявлення наявності та кількості бактерій, вірусів, грибків та інших мікроорганізмів, що можуть впливати на якість води та її безпеку.

Ці методи можуть використовуватися окремо або в комбінації [46] для отримання більш повної та точної інформації про якість води у містах. Обрані методи зазвичай залежать від конкретних параметрів, які моніторяться, а також від доступності та обладнання, яке застосовується в місцях моніторингу.

Однією з технологічних розробок для моніторингу якості води в місті є проєкт Water Quality Sensors [25], який надає інтелектуальні датчики для різноманітних потреб моніторингу води. Датчики є OEM-версіями, що означає, що користувачі можуть їх брендувати та налаштовувати. Такі датчики

готові до Інтернету речей, тож їх можна жити від батарей та підключати до будь-якого стороннього пристрою чи мережі. Вони здатні вимірювати рН, провідність, розчинений кисень, каламутність, температуру тощо. Датчики від Water Quality Sensors можна використовувати для моніторингу відходів і питної води, аквакультури, промислових і технологічних вод, ґрунтових і поверхневих вод.

1.3. Системи збору даних про стан ґрунту

Аналіз екологічного стану ґрунту використовує різноманітні технології, спрямовані на вимірювання фізико-хімічних та біологічних параметрів для оцінки його якості та виявлення забруднень. Ось деякі з них:

1. Датчики вимірювання параметрів ґрунту. Визначають різні характеристики ґрунту, такі як рівень вологості, рН, концентрація різних речовин (наприклад, азот, фосфор), електропровідність. Ці датчики можуть бути встановлені у ґрунті або використовувати безконтактні методи вимірювання [31].

2. Геоінформаційні системи (ГІС) та супутникові зйомки. Використовуються для картографування і візуалізації стану ґрунту, виявлення змін у ландшафті та розподілі різних типів ґрунтів у місцевості [5, 8, 9].

3. Біосенсори та біоіндикатори. Передбачають використання організмів (наприклад, рослин або бактерій) або їхніх чутливих до забруднень реакцій у якості індикаторів для виявлення наявності та рівня забруднення ґрунту.

4. Електроніка та сенсорна техніка. Розробка електронних пристроїв, які вимірюють параметри ґрунту та транслюють отримані дані до систем збору та обробки інформації [45].

5. Аналізатори молекулярного рівня. Використання методів аналізу на молекулярному рівні для виявлення хімічних сполук та забруднювачів у ґрунті, що може включати мас-спектрометрію, газову хроматографію та інші техніки.

Ці технології відіграють критичну роль у забезпеченні точного та об'єктивного аналізу екологічного стану ґрунту [24]. Вони дозволяють не лише виявляти наявність забруднень, але й встановлювати їхнє джерело, динаміку змін, а також передбачати можливі наслідки для екосистеми та здоров'я населення.

Технології аналізу ґрунту стають дедалі більш точними та чутливими завдяки поєднанню різних методик та використанню наукових розробок у сфері хімії, біології та інженерії. Завдяки цьому, вдало аналізуючи ґрунт, можна отримати цінні дані для прийняття рішень у сфері охорони навколишнього середовища, землекористування та планування розвитку.

1.4. Моніторинг шумового забруднення

Систематичний збір, обробка та аналіз даних про шумове забруднення в місті дозволяє приймати обґрунтовані рішення для поліпшення якості життя мешканців міста та збереження екологічної рівноваги. Цей процес включає кілька ключових етапів.

1. Встановлення датчиків із вимірювальними пристроями.

Розміщення датчиків на стратегічних пунктах у місті, які охоплюють різні джерела шуму (автомобільні дороги, промислові зони, житлові райони тощо). Існує декілька типів датчиків та технологій, які використовуються для моніторингу шуму в містах [16, 20, 41]:

- Іонізаційні мікрофони. Ці мікрофони здатні вимірювати шум у великому діапазоні частот та мають високу чутливість. Вони ефективні для вимірювання різних джерел шуму, від транспорту до промислових об'єктів.
- Цифрові шумоміри. Ці прилади використовуються для вимірювання рівня шуму у вигляді числових значень. Вони можуть бути підключені до мережі IoT (інтернету речей) для надсилання даних у реальному часі на сервери для подальшого аналізу.

- Акустичні камери. Ці технології використовують мікрофони та спеціальне програмне забезпечення для візуалізації джерел шуму на карті. Вони дозволяють точно локалізувати джерела шуму та його інтенсивність.
- Акустичні датчики на основі штучного інтелекту (ШІ). Системи, що використовують ШІ, можуть аналізувати звукові дані та розпізнавати різні типи шуму (наприклад, шум від транспорту, будівельні роботи тощо), використовуючи алгоритми машинного навчання.

2. Запис і зберігання даних. Збір даних про рівень шуму в реальному часі та їх зберігання у відповідних базах даних.

3. Оцінка рівня шуму та ідентифікація джерел. Обробка отриманих даних для визначення середньодобових, середньогодинних або інших характеристик рівня шуму. Це дозволяє встановити найшумніші райони та виділити основні джерела шуму в місті.

4. Картографування та візуалізація даних. Створення карт або графіків, які ілюструють рівень шуму в різних частинах міста для кращого розуміння і представлення результатів.

5. Інтерпретація та застосування результатів. Аналіз впливу шумового забруднення на здоров'я людей та екологічну ситуацію в місті на основі отриманих даних та використання результатів для розробки планів та стратегій зменшення шумового рівня у місті, таких як зміна транспортних маршрутів, звукова ізоляція доріг, зонування міста тощо [26].

Після збору та обробки інформації показники можуть бути внесені до бази даних, яка може мати ряд таблиць для зберігання інформації про рівні шуму в різних частинах міста, часові марки вимірювань, характеристики джерел шуму та інші параметри, що дозволять здійснювати аналізи за різними критеріями. Така база даних може підтримувати функції виконання запитів для отримання конкретних даних про рівень шуму за певний період часу, визначення середніх значень, аналізу варіації шуму в різний час та розподіл

шуму у просторі. Завдяки цьому аналізу уможлиблюється визначення гарячих точок з високим рівнем шуму та ідентифікація основних джерел шуму в місті.

Одним із прикладів технологічного проекту моніторингу шумового забруднення в місті є проєкт Sounds of New York City (SONYC) [42]. Цей проєкт використовує інтелектуальну мережу акустичних датчиків, громадських науковців і співпрацю з міськими установами для моніторингу та пом'якшення рівня міського шумового забруднення. В рамках проєкту було розгорнуто 75 фіксованих датчиків і зібрано понад 150 мільйонів 10-секундних аудіозаписів. В проєкті також було задіяно машинне навчання для розрізнення джерел шуму, таких як будівництво, рух, постріли та музика.

1.5. Інтегровані системи моніторингу

Інтегровані системи моніторингу екології – це комплексні платформи або програмні засоби, які забезпечують можливість одночасного або послідовного спостереження, збору, аналізу та управління даними, пов'язаними з різними аспектами навколишнього середовища в місті [12]. Основними характеристиками таких систем є:

- **Комплексність.** Інтегровані системи охоплюють не лише один аспект екології, але й об'єднують декілька важливих параметрів, таких як якість повітря, води, стан ґрунту, рослинність, шумове забруднення та інші.
- **Збір та аналіз даних.** Вони здатні обробляти інформацію з різних джерел, таких як датчики, супутникові дані, метеорологічні станції, та виводити комплексну інформацію для аналізу.
- **Інтеграція і взаємодія.** Ці системи забезпечують можливість інтеграції різних видів даних і їхню взаємодію, що дозволяє здійснювати аналіз і прийняття рішень на основі комплексного підходу.
- **Управління та прийняття рішень.** Вони надають можливість управлінням містами та екологічними організаціями мати доступ до важливої інформації для прийняття обґрунтованих рішень у сфері охорони довкілля та здоров'я мешканців.

Системи, які охоплюють різні аспекти екологічного моніторингу одночасно, мають численні переваги, які сприяють удосконаленню процесів контролю та управління станом навколишнього середовища в містах [23, 43]. Ось деякі з них:

- Комплексність даних: Такі системи надають можливість отримувати повніше уявлення про стан екології в місті, оскільки збирають інформацію про різні аспекти, такі як якість повітря, води, ґрунту, шумове забруднення тощо.
- Актуальність інформації. Оскільки дані збираються одночасно з різних джерел, це дозволяє оперативно отримувати інформацію про зміни в екологічних показниках та швидко реагувати на можливі загрози довкіллю.
- Здатність до аналізу взаємодій між факторами. Інтегровані системи моніторингу дозволяють виявляти взаємозв'язки між різними аспектами довкілля, що є критичним для правильного розуміння і управління екологічними проблемами.
- Ефективне прийняття рішень. Комплексна інформація, отримана від інтегрованих систем, допомагає приймати обґрунтовані рішення для збереження середовища та забезпечення здоров'я мешканців.
- Ефективність ресурсного використання. Інтегровані системи дозволяють оптимізувати використання ресурсів при зборі та аналізі даних, оскільки вони забезпечують узгодженість та системність у процесі моніторингу.

Інтегровані системи моніторингу екологічних показників міста складаються з наступних основних елементів [4]:

1. Мережа спостережень – це сукупність стаціонарних і мобільних постів моніторингу, які вимірюють різні параметри довкілля, такі як якість повітря, води, ґрунту, рівень шуму, радіації тощо. Мережа спостережень може використовувати різні типи датчиків, засобів зв'язку, пристроїв зберігання і передачі даних.

2. Центр обробки даних – це вузол, який отримує, зберігає, обробляє і аналізує дані від мережі спостережень. Центр обробки даних може використовувати різні методи та алгоритми для перетворення сирової інформації в корисну інформацію, таку як індекси якості довкілля, тренди, прогнози, попередження тощо.
3. База даних – це сховище, яке зберігає дані від центру обробки даних в структурованому і доступному форматі. База даних може використовувати різні моделі та системи керування базами даних для забезпечення надійності, безпеки, швидкості і гнучкості доступу до даних.
4. Інтерфейс користувача – це засіб, який надає дані від бази даних до кінцевих користувачів, таких як органи влади, науковці, громадськість тощо. Інтерфейс користувача може використовувати різні формати та канали для візуалізації і поширення даних, такі як вебсайти, мобільні додатки, інформаційні табло, звіти тощо.

Для створення інтегрованих систем моніторингу екологічних показників міста необхідно враховувати ряд факторів [1, 35]:

- Мета та завдання моніторингу – визначення цілей, показників, критеріїв, стандартів та індикаторів моніторингу, що відповідають потребам та інтересам зацікавлених сторін.

- Об'єкти та суб'єкти моніторингу – визначення джерел, складових, параметрів та зон впливу забруднення довкілля, а також учасників, відповідальних за проведення моніторингу.

- Методи та засоби моніторингу – вибір найбільш адекватних, надійних, точних та економічних методів та засобів вимірювання, обробки, аналізу та надання даних.

- Організація моніторингу – розробка організаційної структури, функціональних обов'язків, правил та процедур, а також забезпечення необхідних ресурсів та коштів для реалізації моніторингу.

Так, у роботі гонгконгських розробників [44] описана система, що складається з мобільного пристрою та серверної платформи з дев'ятьма високоточними мікродатчиками для оцінки впливу на навколишнє середовище, наприклад РМ-2,5, РМ-10, СО, SO₂, УФ-індекс та шум. Система була застосована в Гонконзі, і вона може виявити просторово-часові варіації якості навколишнього середовища та вплив різних факторів, таких як тайфуни та рослинний покрив. Система також може підтримувати стає планування та дослідження, надаючи індикатори та моніторинг стійкості міст до клімату та екологічної депривації.

РОЗДІЛ 2. РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ЕКОЛОГІЧНИХ ПОКАЗНИКІВ МІСТА

2.1. Побудова бази даних та визначення архітектури системи

Бази даних для зберігання екологічних показників міста є важливим інструментом для ефективного управління екологічною ситуацією в місті. Вони забезпечують централізоване зберігання, достовірність та прозорість інформації про стан довкілля.

Переваги використання бази даних для зберігання екологічних показників міста [6, 10, 20]:

- Централізоване зберігання даних: Бази даних дозволяють зберігати дані про стан довкілля в одному місці. Це забезпечує їхню уніфікацію та полегшує доступ до них.
- Забезпечення достовірності даних: Бази даних дозволяють контролювати точність даних та своєчасність їхнього внесення.
- Покращення якості аналізу даних: Бази даних дозволяють використовувати сучасні методи аналізу даних, що дає можливість отримати більш точні та інформативні висновки про стан довкілля.
- Забезпечення прозорості інформації: Бази даних дозволяють зробити інформацію про стан довкілля в місті доступною для громадськості.

Основні функції бази даних для зберігання екологічних показників міста [2, 7]:

- Збір даних: Бази даних дозволяють збирати дані про стан довкілля з різних джерел, таких як лабораторні дослідження, моніторингові станції, статистичні звіти підприємств та установ.
- Зберігання даних: Бази даних дозволяють зберігати дані про стан довкілля в структурованому вигляді. Це забезпечує їхній швидкий пошук та аналіз.
- Аналіз даних: Бази даних дозволяють використовувати сучасні методи аналізу даних для отримання інформації про стан довкілля.

- Відображення даних: Бази даних дозволяють відображати дані про стан довкілля у вигляді звітів, карт, графіків та інших форм.

Для створення моделі бази даних моніторингу екологічних показників міста, таких як забруднення повітря, стан води, рівень шуму та стан ґрунту, було ухвалене рішення використати реляційну базу даних з низкою таблиць, які представлятимуть різні аспекти системи моніторингу. Перша таблиця буде включати дані про типи датчиків, їхнє розташування та технічні параметри. Інші таблиці будуть зберігати зібрані дані, наприклад, таблиця з даними про забруднення повітря в конкретних точках міста з відомостями про рівень забруднення для різних речовин. Окремі таблиці можуть бути призначені для зберігання історичних даних, що дозволять проводити аналізи змін показників в часі. Крім того, зв'язуючі таблиці забезпечать зв'язок між різними видами даних та їхню логічну організацію для забезпечення ефективного доступу та обробки інформації у системі моніторингу [11, 19].

Схема бази даних містить такі таблиці:

1. Таблиця **measuring_points** – містить інформацію про пункти моніторингу екологічних показників міста.

Таблиця 2.1 – Структура таблиці measuring_points

Поле	Опис
id	Ідентифікатор пункту моніторингу
name	Назва пункту моніторингу
location	Географічне положення пункту моніторингу

Ідентифікатор пункту моніторингу (поле id) є унікальним ідентифікатором пункту моніторингу. Він використовується для ідентифікації пункту моніторингу в інших таблицях бази даних.

Назва пункту моніторингу (поле name) є текстовим полем, яке містить назву пункту моніторингу.

Географічне положення пункту моніторингу (поле location) є географічною точкою, яка містить координати пункту моніторингу.

2. Таблиця **measurements** – містить інформацію про вимірювання екологічних показників, які проводяться в пунктах моніторингу міста.

Таблиця 2.2 – Структура таблиці measurements

Поле	Опис
id	Ідентифікатор вимірювання
measuring_point_id	Ідентифікатор пункту моніторингу, в якому було проведено вимірювання
measurement_date	Дата та час проведення вимірювання
temperature	Температура повітря або води
humidity	Вологість повітря
ammonia	Рівень амоніаку в повітрі
carbon_dioxide	Рівень вуглекислого газу в повітрі
hydrogen_sulfide	Рівень сірководню в повітрі
nitrogen_oxides	Рівень оксидів азоту в повітрі
dust_concentration	Концентрація пилу в повітрі
water_ph	pH води
water_temperature	Температура води
dissolved_oxygen_level	Рівень розчиненого кисню у воді
soil_ph	pH ґрунту

Ідентифікатор вимірювання (поле id) є унікальним ідентифікатором вимірювання. Він використовується для ідентифікації вимірювання в інших таблицях бази даних.

Ідентифікатор пункту моніторингу (поле measuring_point_id) є ідентифікатором пункту моніторингу, в якому було проведено вимірювання. Це поле пов'язує таблицю **measurements** з таблицею **measuring_points**.

Дата та час проведення вимірювання (поле measurement_date) містять дату та час, коли було проведено вимірювання.

Екологічні показники (поля temperature, humidity, ammonia, carbon_dioxide, hydrogen_sulfide, nitrogen_oxides, dust_concentration, water_ph, water_temperature, dissolved_oxygen_level, soil_ph) містять значення екологічних показників, які були виміряні в пункті моніторингу.

3. Таблиця **sensors** – містить інформацію про сенсори, які використовуються для вимірювання екологічних показників.

Таблиця 2.3 – Структура таблиці sensors

Поле	Тип	Опис
id	integer	Ідентифікатор сенсора
name	string	Назва сенсора
type	string	Тип сенсора
measuring_point_id	integer	Ідентифікатор пункту моніторингу, в якому встановлено сенсор
location	point	Географічне положення сенсора
calibration_date	timestamp	Дата калібрування сенсора

Ідентифікатор сенсора (поле id) є унікальним ідентифікатором сенсора. Він використовується для ідентифікації сенсора в інших таблицях бази даних.

Назва сенсора (поле name) є текстовим полем, яке містить назву сенсора.

Тип сенсора (поле type) є текстовим полем, яке містить тип сенсора. Наприклад, "термометр", "гігрометр", "датчик амоніаку" тощо.

Ідентифікатор пункту моніторингу (поле measuring_point_id) є ідентифікатором пункту моніторингу, в якому встановлено сенсор. Це поле пов'язує таблицю sensors з таблицею measuring_points.

Географічне положення сенсора (поле location) є географічною точкою, яка містить координати сенсора.

Дата калібрування сенсора (поле calibration_date) містить дату, коли сенсор був відкоригований.

Отже, на основі наданих даних можна побудувати графічну схему зв'язків між таблицями. На рис. зображена схема, яка відображає зв'язки між таблицями `measuring_points`, `measurements` та `sensors`.

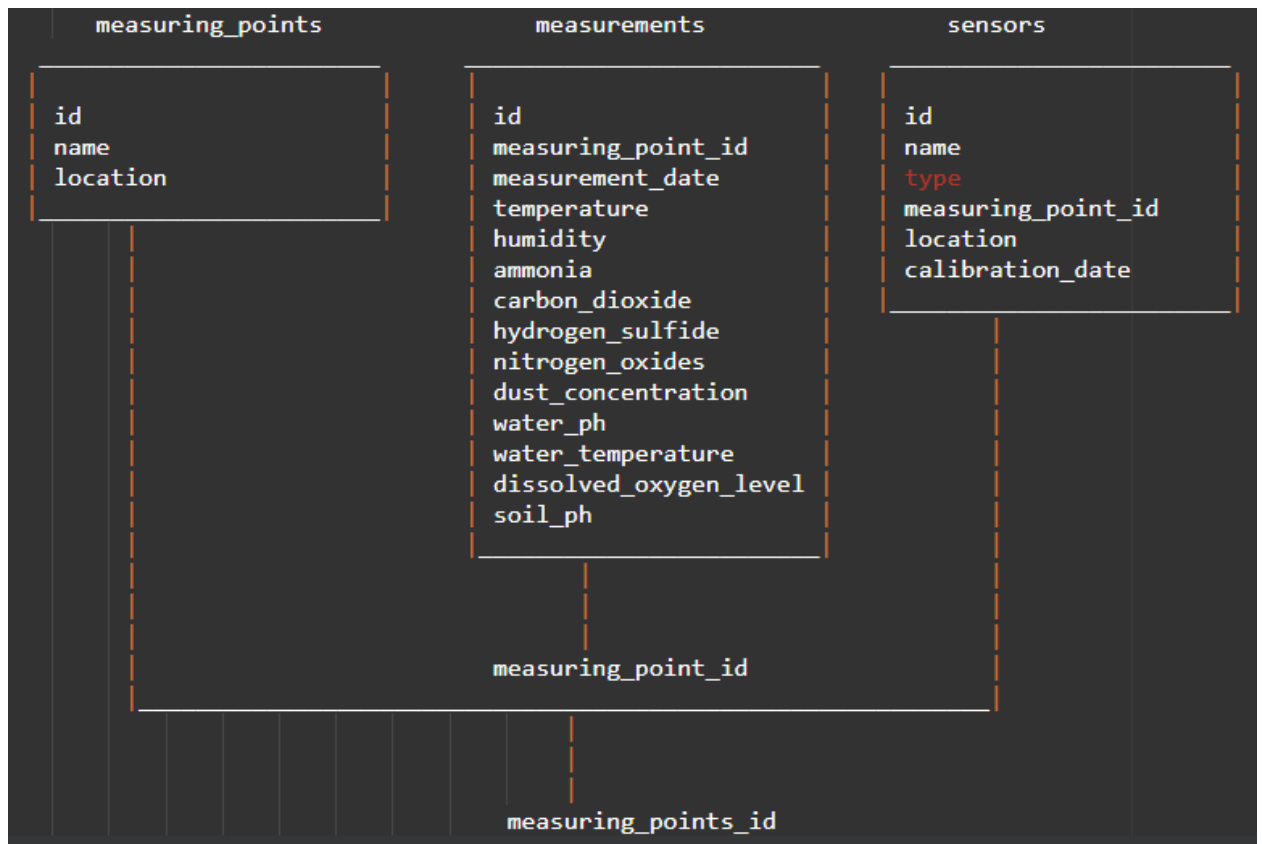


Рис. 2.1 – Взаємозв'язок між таблицями в БД системи

У цій схемі зв'язків:

- `measuring_points_id` у таблиці `sensors` вказує на `id` у таблиці `measuring_points`, що вказує, до якого пункту моніторингу прикріплений сенсор.
- `measuring_point_id` у таблиці `measurements` вказує на `id` у таблиці `measuring_points`, вказуючи, в якому пункті моніторингу проводилися вимірювання.
- Зв'язок між `measurements` та `sensors` відбувається через спільний `measuring_point_id`, оскільки вимірювання проводяться у пунктах моніторингу, де розташовані сенсори.

Ця схема ілюструє, як дані про вимірювання (**measurements**) пов'язані з пунктами моніторингу (**measuring_points**) і сенсорами (**sensors**), що дозволяє структурувати та організувати дані для моніторингу екологічних показників.

2.2. Інтеграція датчиків у систему

Обрання відповідних датчиків для моніторингу стану повітря є критично важливою складовою для забезпечення точності та повноти отриманих даних. Вибір датчиків для вимірювання параметрів повітря, таких як рівень газів, температура, вологість, згідно з рекомендаціями [10, 38, 40], вимагає аналізу функціональних можливостей, точності та надійності пристроїв.

Датчик температури і вологості.

Датчик DHT22, також відомий як AM2302, є цифровим датчиком температури та вологості, який забезпечує високу точність вимірювань та зручність у використанні. Він використовується в системах моніторингу клімату, в промисловості та в домашніх проектах і сумісний з системами автоматизації та управління, де важливо вимірювати температуру та вологість. Далі представлений детальний огляд цього датчика.

Вимірювана температура: Діапазон вимірювання температури зазвичай становить від -40°C до $+80^{\circ}\text{C}$.

Вимірювана вологість: Датчик здатний вимірювати вологість у діапазоні від 0% до 100%.

Точність: Зазвичай має дуже хорошу точність для температури (зазвичай $\pm 0.5^{\circ}\text{C}$) та вологості ($\pm 2-5\%$).

Інтерфейс зв'язку: Використовує простий інтерфейс для зчитування даних, який може бути легко підключений до мікроконтролерів, таких як Arduino або Raspberry Pi.

Цифровий сенсор: DHT22 використовує цифровий сенсор для вимірювання температури та вологості повітря.

Передача даних: Датчик використовує протокол одночасної передачі даних для передачі вимірів через один пін зв'язку.

Калібрування: Зазвичай не потребує додаткового калібрування та забезпечує стабільні та точні результати.

На рис. 2.2 показана схема з такими компонентами:

- DHT22 – сам датчик.
- VCC – живлення датчика (5 В або 3,3 В).
- GND – загальний провід.
- DATA – вихідний сигнал датчика.

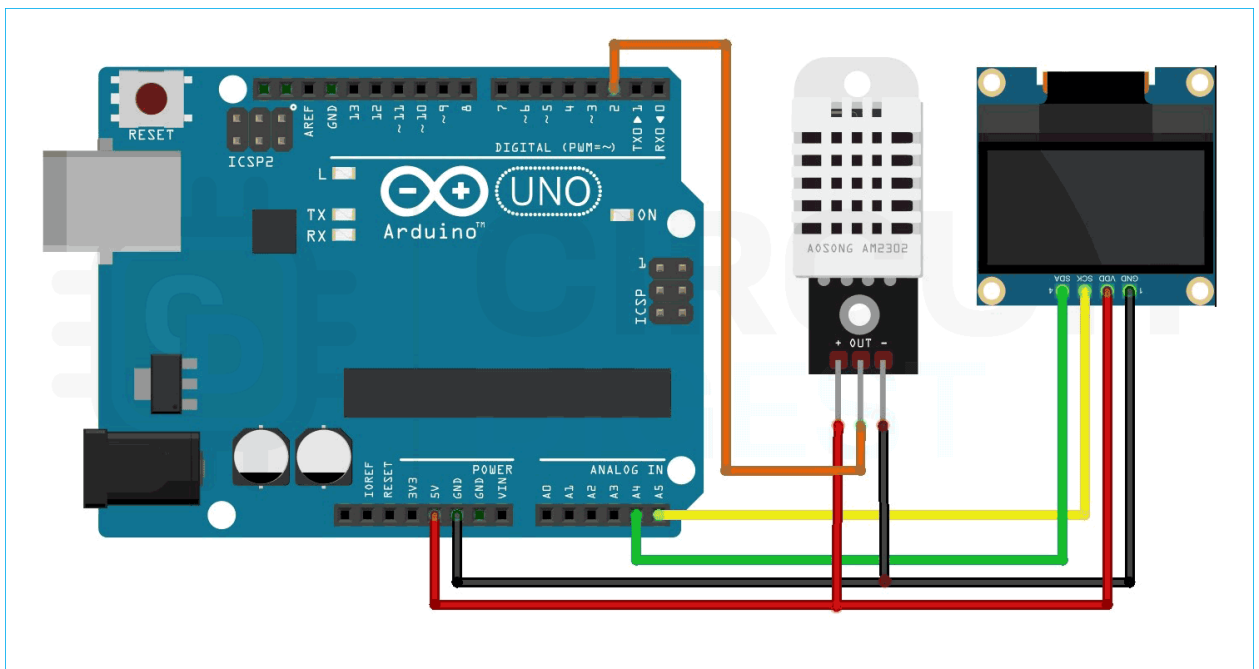


Рис. 2.2 – Схема підключення датчика DHT22

VCC і GND підключаються до джерела живлення. DATA підключається до мікроконтролеру або іншого пристрою, який буде приймати вихідні дані датчика.

NTC-термометр – це датчик, який використовує опір NTC-резистора для вимірювання температури. Опір NTC-резистора зменшується зі збільшенням температури. Датчик DHT22 використовує NTC-резистор з номінальним опором 10 кОм.

Ємнісний датчик вологості – це датчик, який використовує ємність плівки з кераміки для вимірювання вологості. Ємність плівки збільшується зі збільшенням вологості. Датчик DHT22 використовує плівку з кераміки з номінальною ємністю 22 пікофарада.

Для роботи датчика DHT22 необхідно виконати наступні кроки:

1. Підключити датчик до джерела живлення.
2. Задати частоту тактового сигналу датчика.
3. Відправити команду на датчик для початку вимірювання.
4. Отримати вихідні дані датчика.

Для отримання вихідної інформації від датчика DHT22 необхідно передати йому команду з восьми біт. Вихідні дані датчика DHT22 містять код помилки (0 або 1), значення температури та значення вологості.

Код помилки вказує на те, чи успішно завершилося вимірювання. Якщо він дорівнює 0, то вимірювання пройшло успішно. Якщо код помилки дорівнює 1, то вимірювання невдале. Значення температури і вологості представлені у вигляді 16-бітних чисел, які необхідно конвертувати в градуси Цельсія і відсотки. На рис. 2.3 наведений код для зчитування температури й вологості повітря і виведення значень на моніторі серійного порту.

```
#include <DHT.h>

#define DHTPIN 2      // Пін, на якому підключений сенсор DHT22
#define DHTTYPE DHT22 // Вказуємо тип сенсора (DHT22)

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600); // Ініціалізація зв'язку з монітором (швидкість 9600 bps)
  dht.begin();        // Ініціалізація сенсора
}

void loop() {
  float humidity = dht.readHumidity(); // Зчитування вологості
  float temperature = dht.readTemperature(); // Зчитування температури

  // Перевірка на наявність помилок у зчитуванні
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Помилка зчитування даних з датчика DHT!");
    return;
  }

  // Виведення результатів на монітор
  Serial.print("Вологість: ");
  Serial.print(humidity);
  Serial.print("%\tТемпература: ");
  Serial.print(temperature);
  Serial.println("°C");
}
```

Рис. 2.3 – Реалізація зчитування температури й вологості повітря

Алгоритм роботи складається з таких кроків:

1. Ініціалізація. Підключення датчика до плати Arduino і його налаштування.
2. Зчитування даних: Кожні 2 секунди датчик зчитує значення температури та вологості з оточуючого повітря.
3. Перевірка даних: Перевірка правильності зчитаних значень. Якщо дані не вдалося зчитати, програма виведе повідомлення про помилку.
4. Виведення результатів: Отримані значення температури та вологості виводяться на моніторі порту в Arduino IDE у форматі: "Вологість: [значення]% Температура: [значення]°C".

Датчик газів.

Датчик MQ-135 є газовим датчиком, призначеним для виявлення різних видів газів у повітрі. Основною особливістю цього датчика є його здатність виявляти гази, такі як амоніак, діоксид вуглецю, азотні оксиди та багато інших, що робить його корисним у вимірюванні якості повітря [32, 33]. Нижче подано детальний огляд датчика MQ-135.

Вимірювані гази: амоніак (NH_3), діоксид вуглецю (CO_2), сірководень (H_2S), бензен (C_6H_6), ацетон ($\text{C}_3\text{H}_6\text{O}$), толуол (C_7H_8), ацетилентріхлорид ($\text{C}_2\text{H}_3\text{Cl}_3$), метанол (CH_4O), ацетонітрил ($\text{C}_2\text{H}_3\text{N}$), диметиламін ($\text{C}_2\text{H}_7\text{N}$).

Напруга живлення: Зазвичай працює при напрузі 5V DC.

Точність: Точність вимірювання може варіюватися в залежності від газу, але в загальному, для багатьох газів, точність не є дуже високою.

Температурний діапазон: Здатний працювати у широкому діапазоні температур.

Провідна плівка: MQ-135 має провідну плівку, яка змінює свою провідність при зміні концентрації різних газів.

Зміна опору: При зустрічі з певними газами, опір датчика змінюється, що може бути виміряно іншими пристроями для визначення концентрації газів у повітрі.

На рис. 2.4 зображена схема датчика, підключеного до мікроконтролеру Arduino. Мікроконтролер використовується для управління датчиком та для отримання вихідного сигналу від датчика. Схема містить такі компоненти:

- MQ135 – сам датчик.
- VCC – живлення датчика (5 В або 3,3 В).
- GND – загальний провід.
- AOUT – вихідний сигнал датчика.

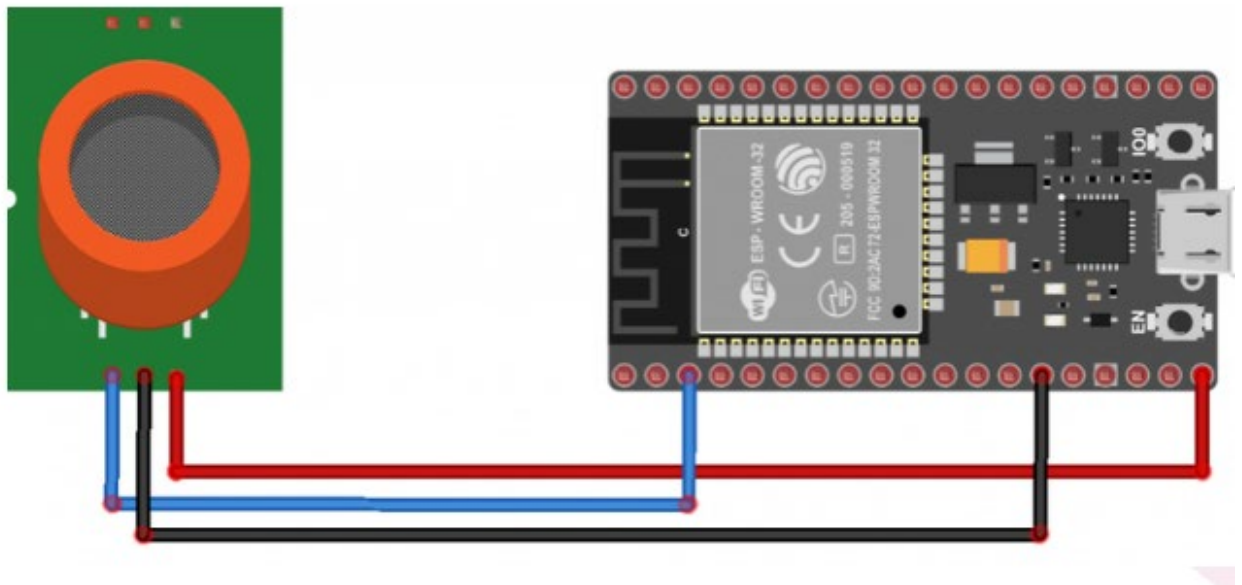


Рис. 2.4 – Схема підключення датчика MQ135

VCC і GND підключаються до джерела живлення. AOUT підключається до мікроконтролеру або іншого пристрою, який буде приймати вихідний сигнал датчика.

MQ135 є чутливим електродом, покритим шаром металоксиду. Коли газові сполуки потрапляють на цей шар, вони утворюють електричний струм. Цей струм є пропорційним концентрації газів у повітрі.

Для роботи датчика MQ135 необхідно виконати наступні кроки:

1. Підключити датчик до джерела живлення.
2. Задати частоту тактового сигналу датчика.
3. Відправити команду на датчик для початку вимірювання.
4. Отримати вихідний сигнал датчика.

Для отримання вихідного сигналу від датчика MQ135 необхідно передати йому команду з восьми біт.

Вихідний сигнал датчика MQ135 є пропорційним концентрації газових сполук у повітрі. Цей сигнал можна конвертувати в концентрацію, використовуючи відповідну таблицю або модель.

Для зчитування значень з датчика MQ-135, обчислення опору датчика та конвертації його у частку на мільйон (ppm) для різних газів, таких як амоніак, вуглекислий газ, сірководень та оксиди азоту, використовувався код, зображений на рис. 2.5.

```
int sensorPin = A0; // Пін, до якого підключено датчик MQ-135
float R0 = 10000; // Значення R0 (опір датчика при чистому повітрі)
float ppm; // Значення ppm (частка на мільйон) для газу

void setup() {
  Serial.begin(9600); // Ініціалізація зв'язку з монітором
}

void loop() {
  int sensorValue = analogRead(sensorPin); // Зчитування значення з датчика
  float voltage = sensorValue * (5.0 / 1023.0); // Конвертування в напругу

  float RS = ((5.0 * R0) / voltage) - R0; // Обчислення опору RS
  float ratio = RS / R0; // Обчислення співвідношення RS/R0

  // Корекція значення ppm для різних газів
  float ammonia = 0.53; // Корекція для амоніаку
  float carbonDioxide = 0.18; // Корекція для вуглекислого газу
  float hydrogenSulfide = 0.02; // Корекція для сірководню
  float nitrogenOxides = 0.01; // Корекція для оксидів азоту

  ppm = (1.0 / ratio - 1.0) / ammonia; // Обчислення рівня амоніаку в ppm
  // Аналогічно для інших газів: ppm для вуглекислого газу, сірководню та оксидів азоту

  Serial.print("Ammonia (NH3): ");
  Serial.print(ppm);
  Serial.println(" ppm");

  delay(1000); // Затримка перед наступним вимірюванням
}
```

Рис. 2.5 – Реалізація зчитування даних з датчика MQ135

Так, значення опору датчика MQ-135 при зміні концентрації амоніаку (або іншого газу) змінюється, і це використовується для визначення показника. Наведені в коді операції виконують зчитування аналогових значень

з датчика, конвертацію їх у напругу та розрахування відношення опору датчика до опору при чистому повітрі. На основі цього розрахунку визначається концентрація амоніаку в ppm, використовуючи відповідні коефіцієнти корекції для різних газів. Результат виводиться через порт монітора Arduino, дозволяючи спостерігати за змінами рівня амоніаку у реальному часі. Для вимірювання різних типів газів зазначаються відповідні зміни до коефіцієнтів корекції та розрахунків.

Датчик MQ-135, завдяки своїй здатності виявляти різні гази, є корисним інструментом для моніторингу якості повітря у різних ситуаціях та проектах. Однак, для точних вимірювань, важливо враховувати його особливості та обмеження.

Датчик ідентифікації пилу та часток у повітрі.

У системах моніторингу якості повітря в приміщеннях, вуличних мережах моніторингу, пристроях очищення повітря та інших пристроях, які потребують вимірювання концентрації пилу в атмосфері, широко використовується датчик GP2Y1010AU0F [38]. Принцип його роботи полягає у використанні інфрачервоного світла. Датчик має вбудований фоторезистор, який змінює свою опірність в залежності від кількості пилу, який потрапляє на його поверхню. Інфрачервоне світло, що випромінюється датчиком, проходить через повітря і відбивається від часток пилу. Фоторезистор вимірює рівень світла, що повернулося, і на основі цього визначає концентрацію пилу. Його інші особливості включають:

- **Наявність вентилятора.** Датчик часто використовується з вентилятором для забезпечення потоку повітря через нього, щоб забезпечити більш точне вимірювання [13, 36].

- **Аналоговий вихід.** Він має аналоговий вихід, який може бути підключений до мікроконтролерів, таких як Arduino або Raspberry Pi, для отримання значень концентрації пилу.

На рис. 2.6 показана схема датчика GP2Y1010AU0F, який підключений до мікроконтролеру Arduino. Мікроконтролер використовується для управління датчиком та для отримання вихідного сигналу від датчика.

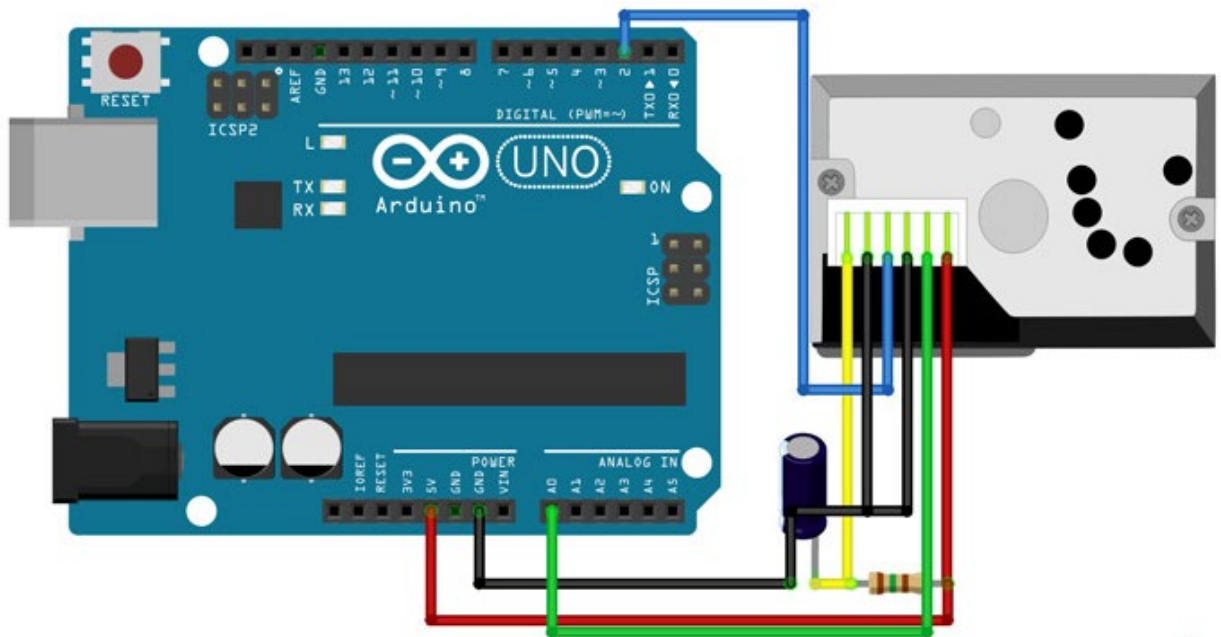


Рис. 2.6 – Схема підключення датчика GP2Y1010AU0F

Датчик складається з таких компонентів:

- GP2Y1010AU0F – сам датчик.
- VCC – живлення датчика (5 В або 3,3 В).
- GND – загальний провід.
- OUT – вихідний сигнал датчика.

На схемі є також кілька необов'язкових компонентів:

- R1 – резистор, який обмежує струм, що протікає через датчик.
- C1 – конденсатор, який допомагає згладжувати вихідний сигнал датчика.
- LED – світлодіод, який індикуює, що датчик працює.

VCC і GND підключаються до джерела живлення. OUT підключається до мікроконтролеру або іншого пристрою, який буде приймати вихідний сигнал датчика.

GP2Y1010AU0F використовує інфрачервоний лазер для вимірювання концентрації пилу в повітрі. Коли пил потрапляє на шлях лазера, він поглинає

частину світла. Ця зміна в інтенсивності світла є пропорційною концентрації пилу в повітрі.

Для роботи датчика GP2Y1010AU0F необхідно виконати такі кроки:

1. Підключити датчик до джерела живлення.
2. Задати частоту тактового сигналу датчика.
3. Відправити команду на датчик для початку вимірювання.
4. Отримати вихідний сигнал датчика.

Вихідний сигнал датчика GP2Y1010AU0F є пропорційним концентрації пилу в повітрі. Цей сигнал можна конвертувати в концентрацію пилу, використовуючи відповідну таблицю або модель.

Код для Arduino з використанням датчика пилу GP2Y1010AU0F, зображений на рис. 2.7, використовується для зчитування концентрації пилу в атмосфері. Датчик підключений до аналогового піна A0 плати Arduino.

```
int dustPin = A0; // Пін для зчитування аналогових даних від датчика пилу
int LEDPin = 13; // Пін світлодіода для візуальної індикації

void setup() {
    Serial.begin(9600); // Ініціалізація з'єднання з монітором порту (Serial Monitor)
    pinMode(LEDPin, OUTPUT);
}

void loop() {
    int dustVal = analogRead(dustPin); // Зчитування значення з датчика пилу
    digitalWrite(LEDPin, HIGH); // Увімкнути світлодіод для візуальної індикації

    delayMicroseconds(280); // Затримка для стабілізації сигналу
    dustVal = analogRead(dustPin); // Повторне зчитування значення пилу

    digitalWrite(LEDPin, LOW); // Вимкнути світлодіод

    // Конвертування значення аналогового сигналу в концентрацію пилу
    float dustConcentration = (float)dustVal * (5.0 / 1024.0);

    // Виведення значення концентрації пилу в Serial Monitor
    Serial.print("Значення пилу: ");
    Serial.println(dustConcentration);

    delay(1000); // Затримка для плавного оновлення даних
}
```

Рис. 2.7 – Програмна реалізація зчитування даних про концентрацію пилу з датчика GP2Y1010AU0F

Основні етапи роботи коду:

1. Ініціалізація пінів та з'єднання з монітором порту (Serial Monitor).
2. Отримання значень від датчика пилу через аналоговий вхід A0.
3. Увімкнення світлодіода на платі для візуальної індикації процесу вимірювання.
4. Затримка та повторне зчитування значень для отримання точніших даних.
5. Конвертація аналогових значень у концентрацію пилу.
6. Виведення отриманих даних про концентрацію пилу в Serial Monitor.

Таким чином уможлиблюється моніторинг рівня пилу в атмосфері та отримання числових даних, які можна використовувати для аналізу якості повітря в конкретному середовищі. Важливо враховувати, що калібрування є необхідним для точних вимірювань в різних умовах експлуатації [22].

Датчик рН води.

pH Sensor Kit – це набір, який зазвичай включає в себе датчик рН, а також додаткові компоненти, які дозволяють вимірювати рівень кислотності (рН) рідини або водного середовища. Цей тип набору широко використовується в наукових дослідженнях, проектах IoT, а також у вивченні хімії та біології.

Основні компоненти pH Sensor Kit можуть включати:

1. Датчик рН: Це електрод, який генерує електричний сигнал, пропорційний рівню рН розчину. Датчики можуть мати різні формфактори та типи інтерфейсів (аналогові або цифрові).
2. Калібрувальні рідини: Рідини різних рН, які використовуються для налаштування та калібрування датчика перед використанням.
3. Адаптери та кабелі: Деякі набори можуть включати адаптери та кабелі для зручного підключення датчика до мікроконтролера, такого як Arduino або Raspberry Pi.

Датчики рН зазвичай працюють шляхом вимірювання потенціалу водню (рН) у рідині. Вони можуть бути використані для вимірювання рівня

кислотності води у водоймах для аналізу якості води у виробництві або в агропромисловому секторі.

pH Sensor Kit може взаємодіяти з мікроконтролерами, такими як Arduino або Raspberry Pi, для зчитування та обробки даних про рівень кислотності (pH) рідини. Ця взаємодія зазвичай відбувається через аналогові або цифрові піни на датчику pH, які підключаються до відповідних портів на платформі Arduino або Raspberry Pi.

При використанні Arduino датчик pH підключається до аналогового або цифрового входу за допомогою відповідних пінів (наприклад, для аналогового зчитування використовується пін Analog Input). Деякі датчики можуть вимагати додаткового обробника сигналу або аналогового-цифрового перетворювача для забезпечення правильного читання даних.

При використанні Raspberry Pi застосовується аналогово-цифровий перетворювач (ADC) або модуль, що підтримує аналогові сигнали, для зчитування даних з аналогових вхідних пінів датчика pH. Потім використовуються програмні бібліотеки або інтерфейси для зчитування даних через ці піни.

На рис. 2.8 показаний pH Sensor Kit, підключений до Arduino, який використовувався в рамках цього дослідження. Комплект включає в себе датчик pH, плату Arduino та проводи для підключення датчика до плати.

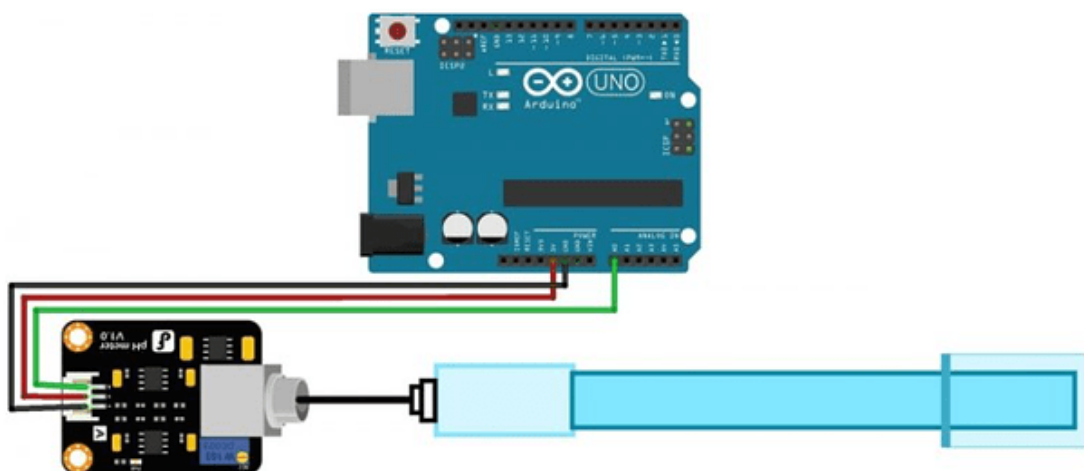


Рис. 2.8 – Схема підключення pH Sensor Kit

На рисунку показано, що датчик рН підключений до плати Arduino за допомогою трьох проводів:

- Червоний провід – підключений до порту VCC плати Arduino.
- Чорний провід – підключений до порту GND плати Arduino.
- Зелений провід – підключений до порту A0 плати Arduino.

Порт VCC забезпечує живлення датчика рН, а порт GND - заземлення датчика. Порт A0 використовується для отримання вихідного сигналу від датчика рН. Датчик рН працює за принципом потенціометрії. Коли два електроди занурюються в рідину, між ними виникає електричний потенціал.

Після підключення датчика рН до мікроконтролера необхідно виконати програмний код, який зчитує дані з датчика. Це включає в себе калібрування датчика (використовуючи відомі рідини різних рН значень), зчитування аналогового або цифрового сигналу, та перетворення цих значень на відповідні показники рівня рН, як показано на рис. 2.9.

```
const int analogPin = A0; // Пін, до якого підключено датчик рН на Arduino

void setup() {
  Serial.begin(9600); // Ініціалізація Serial для виводу результатів
}

void loop() {
  // Зчитування значення з аналогового піна
  int sensorValue = analogRead(analogPin);

  // Перетворення значення на напругу (5V - напруга Arduino)
  float voltage = sensorValue * (5.0 / 1023.0);

  // Перетворення напруги на рівень рН за допомогою калібрувальних значень
  float calibrated_pH = map(voltage, 0.0, 5.0, 0.0, 14.0);

  // Вивід результату на монітор
  Serial.print("Raw ADC Value: ");
  Serial.print(sensorValue);
  Serial.print(", Voltage: ");
  Serial.print(voltage);
  Serial.print(", pH Value: ");
  Serial.print(calibrated_pH); // Виведення виміряного рН на монітор

  Serial.println();

  delay(1000); // Затримка для зручності відображення результатів
}
```

Рис. 2.9 – Реалізація зчитування даних про кислотність з pH Sensor Kit

У цьому коді використовується функція `analogRead()` для зчитування значення з аналогового піна (де A0 – номер аналогового піна на Arduino, до якого підключено датчик рН). Після зчитування проводиться перетворення цього значення на напругу, яка потім може бути перетворена на відповідний рівень рН, використовуючи калібрувальні значення для конкретного датчика.

Після отримання цих значень можна зберігати показники в базі даних або на сервері для подальшого аналізу або візуалізації.

Датчик температури води.

DS18B20 Waterproof Temperature Sensor – це водонепроникний цифровий датчик температури, який широко використовується для вимірювання температури в різних середовищах, включаючи воду. Ось огляд основних характеристик цього датчика:

1. Наявність захисту від вологості: DS18B20 має герметичну оболонку, яка дозволяє йому працювати в умовах вологості або під водою. Це робить його ідеальним для використання в системах моніторингу температури в водних дослідженнях, акваріумах, басейнах або інших водоймах.

2. Цифровий інтерфейс: DS18B20 забезпечує зручний цифровий інтерфейс, що спрощує підключення до мікроконтролерів, таких як Arduino, Raspberry Pi або інші пристрої. Він використовує протокол OneWire, що дозволяє підключати кілька датчиків до одного входу.

3. Висока точність: Цей датчик забезпечує високу точність вимірювання температури з роздільною здатністю до 0.0625°C . Він має широкий діапазон вимірювання температур, зазвичай від -55°C до $+125^{\circ}\text{C}$.

4. Працездатність у різних умовах: DS18B20 може працювати в широкому діапазоні температур та вологості, що робить його придатним для застосування в різних середовищах.

5. Наявність додаткових функцій: DS18B20 може виконувати різні додаткові функції, такі як відправка алармів при перевищенні заданого порогу температури або використання унікального ідентифікатора для кожного датчика.

На рис. 2.10 зображена схема підключення датчика температури DS18B20 до мікроконтролера Arduino. Вона містить такі компоненти:

- DS18B20 – сам датчик.
- VCC – живлення датчика (3,3 В або 5 В).
- GND – загальний провід.
- DQ – вихідний сигнал датчика.

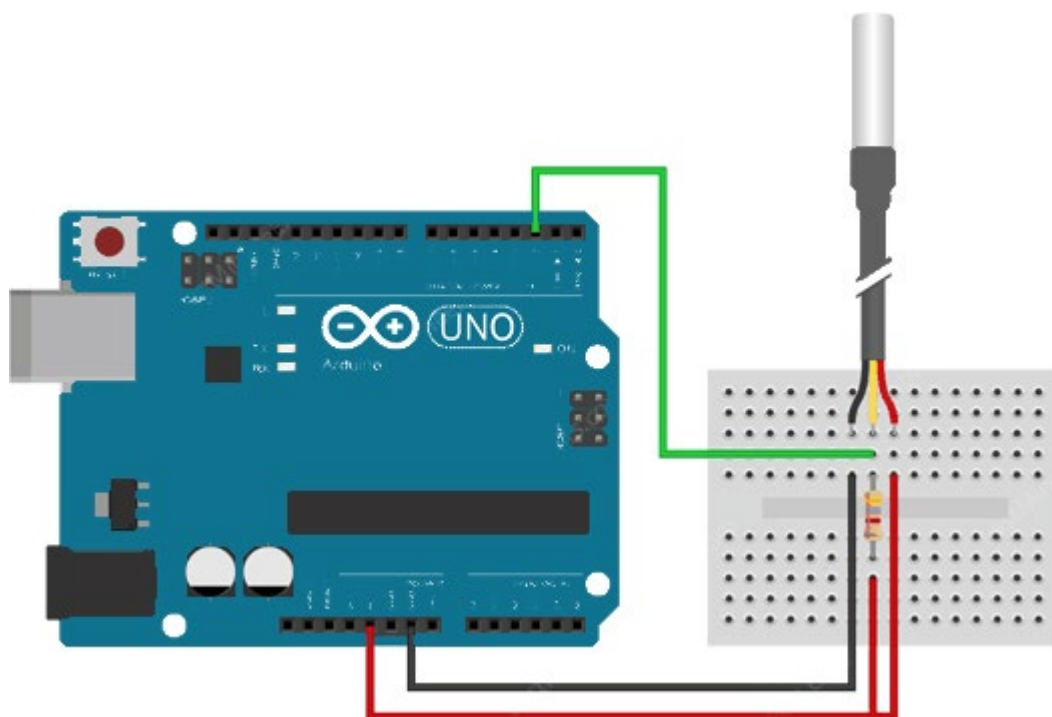


Рис. 2.10 – Схема підключення датчика DS18B20

Датчик DS18B20 має вбудований терморезистор. Коли датчик занурюється в рідину, його опір змінюється в залежності від температури рідини. DS18B20 використовує 1-Wire протокол для передачі даних, тому він підключений до мікроконтролера Arduino за допомогою спеціального штифта 1-Wire. На схемі також показано, що датчик заземлений, що необхідно для забезпечення безпечної роботи.

На рис. 2.11 наведений приклад коду для відправки аларма на Arduino при перевищенні заданого порогу температури. Цей код використовує бібліотеки OneWire та DallasTemperature для комунікації з температурним датчиком DS18B20. Основна мета – виявлення температури води чи іншого

середовища та відправлення аларму, якщо температура перевищує заданий поріг.

```
OneWire oneWire(ONE_WIRE_BUS_PIN);
DallasTemperature sensors(&oneWire);

float alarmTemperature = 25.0; // Заданий поріг температури для аларму

void setup() {
  Serial.begin(9600);
  sensors.begin();
}

void loop() {
  // Отримання поточної температури
  sensors.requestTemperatures();
  float currentTemperature = sensors.getTempCByIndex(0);

  Serial.print("Поточна температура: ");
  Serial.print(currentTemperature);
  Serial.println(" °C");

  if (currentTemperature > alarmTemperature) {
    // Якщо поточна температура перевищує заданий поріг, відправляємо аларм
    Serial.println("Температура перевищила поріг!");
  }

  // Включення світлодіода (світлодіод підключений до піна 13)
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Рис. 2.11 – Відправка аларма при перевищенні порогового значення

Цей код є придатним для відправки повідомлень або активації конкретної сигналізації при перевищенні температурного порогу, наприклад, використовуючи Wi-Fi модуль для відправлення повідомлення або вмикання світлодіоду для сигналізації.

Таким чином, датчик DS18B20 є доречним для використання в різних системах, де потрібно вимірювати температуру в середовищах з вологістю, і він широко використовується у проектах із збору даних про температуру води в екологічних дослідженнях та системах моніторингу.

Датчик рівня розчиненого кисню.

Однією з моделей датчиків розчиненого кисню є Atlas Scientific EZO Dissolved Oxygen Sensor, який може бути інтегрований з мікроконтролерами через UART, I2C або використовується з модулем адаптера, щоб забезпечити його сумісність з Arduino або Raspberry Pi.

Atlas Scientific пропонує декілька варіантів датчиків розчиненого кисню, таких як EZO-DO Embedded Dissolved Oxygen Circuit або Dissolved Oxygen Kit. Ці датчики можуть використовуватися для моніторингу рівня кисню у воді в різних умовах, включаючи акваріуми, водойми, дослідження водних систем тощо.

Основні особливості Dissolved Oxygen Kit:

- Діапазон вимірювання: 0-20 мг/л або 0-200% насичення.
- Точність: $\pm 0,2$ мг/л або $\pm 2\%$ насичення.
- Калібрування: одноточкове або двоточкове.
- Компенсація температури: автоматична або ручна.
- Компенсація солоності та атмосферного тиску: ручна.
- Зберігання даних: 50 наборів даних.

На рис. 2.12 зображена схема, де датчик EZO Dissolved Oxygen підключений до мікроконтролеру Arduino за допомогою плати розширення.

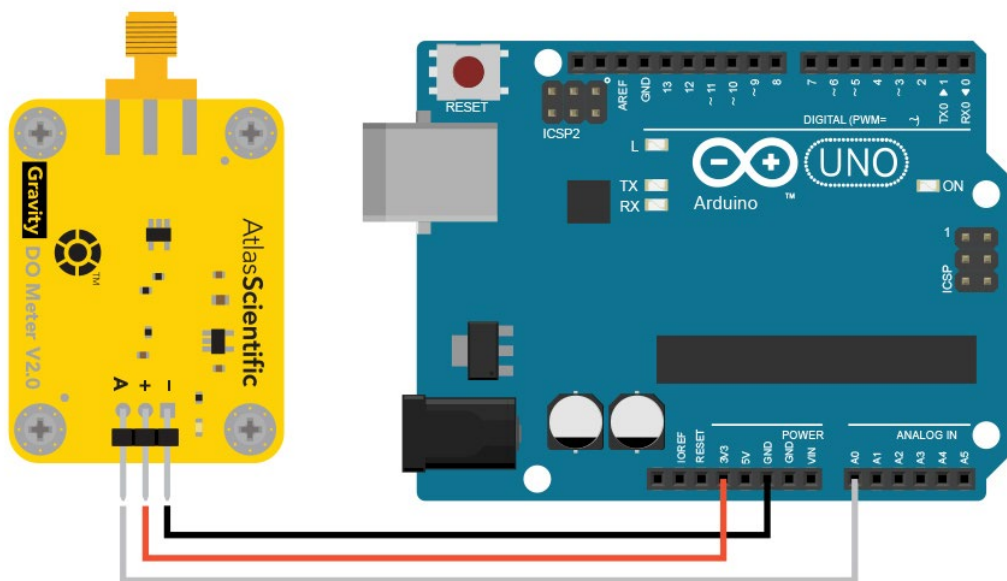


Рис. 2.12 – Схема підключення датчика EZO Dissolved Oxygen

EZO Dissolved Oxygen використовує інтерфейс UART для передачі даних та підключений до таких дротів:

- VCC – живлення датчика (5 В або 3,3 В).
- GND – загальний провід.
- TX – вихідний сигнал датчика.

VCC і GND підключаються до джерела живлення. TX підключається до мікроконтролеру Arduino.

EZO Dissolved Oxygen працює за принципом електрохімічного потенціометру. Електрохімічний потенціометр – це пристрій, який перетворює електричний потенціал в електричний струм. Коли датчик занурюється в рідину, його електричний потенціал змінюється в залежності від концентрації розчиненого кисню в рідині.

Мікроконтролер Arduino використовується для управління датчиком та для отримання вихідного сигналу. Мікроконтролер може використовуватися для вимірювання концентрації розчиненого кисню в рідині та для виведення результату на дисплей або в комп'ютер.

Для отримання показників з датчика розчиненого кисню (Dissolved Oxygen Kit) з Arduino використовується бібліотека, надану Atlas Scientific для взаємодії з їхніми датчиками. Фрагмент коду для зчитування показників з датчика Dissolved Oxygen Kit наведений на рис. 2.13.

```
#include <Wire.h> // Додаткова бібліотека для зв'язку по шині I2C
#include <SoftwareSerial.h> // Бібліотека для роботи з UART

SoftwareSerial mySerial(10, 11); // RX, TX піни для спілкування з датчиком

void setup() {
  Serial.begin(9600); // Ініціалізація серійного порту
  mySerial.begin(9600); // Ініціалізація спілкування з датчиком
}

void loop() {
  if (mySerial.available()) {
    String input = mySerial.readStringUntil('\n'); // Отримання даних від датчика
    Serial.println(input); // Виведення даних до монітора порту
  }
  delay(1000); // Затримка між зчитуванням даних
}
```

Рис. 2.13 – Отримання показників з датчика EZO Dissolved Oxygen

Наведений код демонструє можливість комунікації з датчиком розчиненого кисню від Atlas Scientific за допомогою Arduino. Він використовує SoftwareSerial для емуляції додаткового UART з'єднання, оскільки деякі плати Arduino мають обмежену кількість апаратних UART-портів.

Отримані дані з датчика передаються на монітор порту Arduino IDE через вбудований UART-порт. При кожній ітерації loop() він очікує наявності даних від датчика через SoftwareSerial і виводить їх на монітор порту зі швидкістю 9600 bps (біт в секунду).

Датчик рН ґрунту.

DFRobot Gravity: Analog pH Sensor / Meter Kit – це комплект, який включає аналоговий датчик рН та модуль для вимірювання рівня кислотності (рН) рідини або ґрунту. На рис. 2.14 показана схема підключення рН сенсора до плати Arduino.

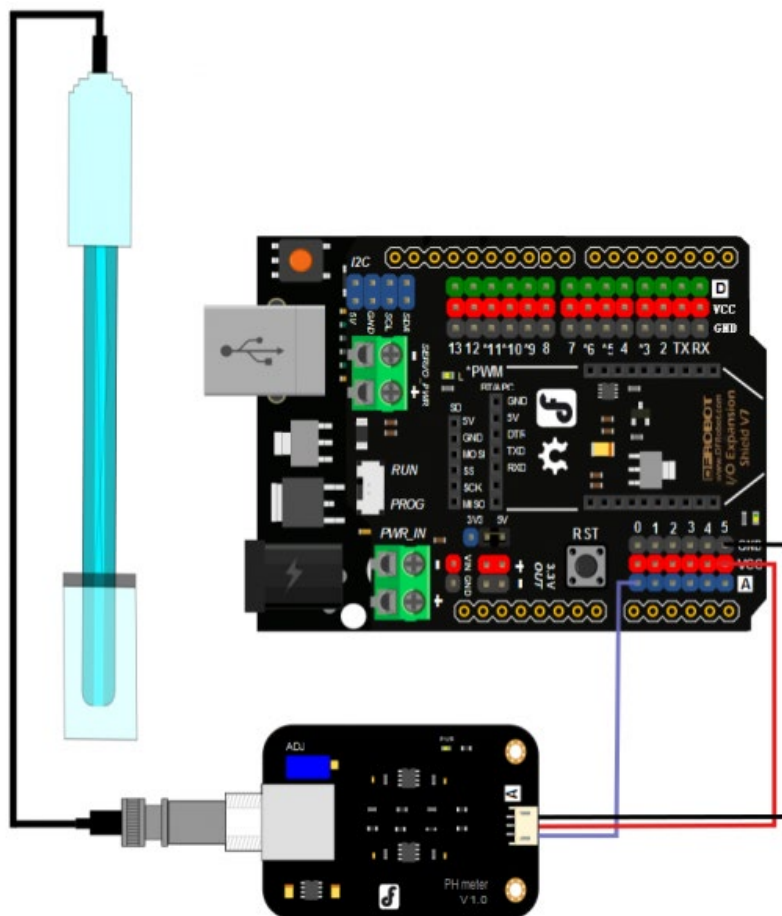


Рис. 2.14 – Схема підключення Analog pH Sensor / Meter Kit

Ось деякі ключові особливості цього комплекту:

- Діапазон вимірювань: від 0 до 14 рН.
- Спосіб вимірювання: Аналоговий вихід, що дозволяє підключити його до аналогового входу мікроконтролера (наприклад, Arduino).
- Сумісність: Може працювати з багатьма мікроконтролерами, включаючи Arduino та Raspberry Pi.
- Структура комплекту: сенсор рН у вигляді електрода, який можна занурити у рідину або ґрунт для вимірювання рівня рН, та модуль перетворення, який приймає сигнал від сенсора рН і перетворює його на аналоговий сигнал для зручного підключення до мікроконтролера.

Принцип роботи сенсора полягає в вимірюванні концентрації водневих іонів у розчині, відображаючи його у вигляді числового значення в діапазоні рН. Після цього модуль перетворення обробляє сигнал та передає його на мікроконтролер для подальшої обробки або аналізу [17].

Цей комплект може бути використаний для вимірювання рівня кислотності в різних середовищах, таких як ґрунт для сільського господарства, водойми для екологічного моніторингу або навіть в лабораторних умовах. Утім, аналогові датчики рН можуть бути схильними до змін у точності через знос або калібрування, яке може знадобитися періодично для збереження точності вимірювань [39].

На рис. 2.15 наведений код для Arduino, який дозволяє зчитувати дані з аналогового датчика рН, підключеного до піну A0. Після зчитування аналогового сигналу, отримане значення конвертується в рівень рН (для спрощення припускається діапазон від 0 до 14). Отримане значення рН виводиться через портальне вікно монітора в Arduino IDE кожну секунду.

```

// Підключення аналогового датчика pH
int sensorPin = A0; // Підключення до аналогового входу A0

void setup() {
  Serial.begin(9600); // Ініціалізація серійного з'єднання
}

void loop() {
  // Зчитування значення з датчика
  int sensorValue = analogRead(sensorPin);
  // Конвертація в рівень pH (припустимо, що діапазон 0-14 pH)
  float pHValue = (float)sensorValue * 5.0 / 1024;

  Serial.print("Значення pH: ");
  Serial.println(pHValue); // Вивід значення pH у вікно монітора порту

  delay(1000); // Затримка для читабельності виводу
}

```

Рис. 2.15 – Реалізація зчитування даних з Analog pH Sensor / Meter Kit

Використання цього коду є необхідним для перевірки та відлагодження датчика pH в реальному часі через серійний монітор Arduino.

2.3. Розробка програмного забезпечення для оперування базою даних

У процесі проектування системи моніторингу екологічних показників міста та вибору відповідних датчиків для збору цих даних було зроблено значний крок у створенні інструменту, спрямованого на отримання важливої та актуальної інформації про екологічні показники міста. Проектування програмного забезпечення спрямована на оптимізацію процесу отримання, збереження та обробки цих даних для подальшого аналізу та використання. Ретельне планування структури програми передбачає забезпечення надійності, ефективності та точності збору і зберігання інформації, враховуючи якість датчиків та особливості бази даних. У цьому контексті, варто розглянути ключові аспекти структури ПЗ та взаємозв'язок між його компонентами, що сприятиме оптимізації та ефективному використанню

програмного інструменту для моніторингу і відстеження екологічних показників.

1. Встановлення з'єднання з базою даних, де знаходяться таблиці `measuring_points`, `measurements` та `sensors`.

Розробка з'єднання з базою даних у програмі відіграє ключову роль у забезпеченні доступу до інформації та виконанні операцій з даними. Для взаємодії з базою даних використовується MySQL Connector/C++, як показано на рис. 2.16.

```
#include <iostream>
#include <mysql_driver.h>
#include <mysql_connection.h>

int main() {
    sql::mysql::MySQL_Driver *driver;
    sql::Connection *connection;

    try {
        driver = sql::mysql::get_mysql_driver_instance();
        connection = driver->connect("tcp://127.0.0.1:3306", "username", "password");

        connection->setSchema("measuring_points");

        std::cout << "Connected to database successfully!" << std::endl;

        delete connection;
    } catch (sql::SQLException &e) {
        std::cout << "SQL Exception: " << e.what() << std::endl;
    }

    return 0;
}
```

Рис. 2.16 – Реалізація з'єднання з базою даних

У цьому фрагменті коду:

mysql_driver.h та **mysql_connection.h** – це частини бібліотеки MySQL Connector/C++;

MySQL_Driver – інтерфейс для отримання драйвера підключення до MySQL;

Connection – об'єкт підключення до бази даних;

"tcp://127.0.0.1:3306" – адреса сервера MySQL та порт для з'єднання;

username та **password** – ідентифікаційні дані для доступу до бази даних;
measuring_points – назва бази даних, до якої відбувається підключення.

Доцільність цього компонента полягає в забезпеченні можливості взаємодії з базою даних MySQL у програмному застосунку. Це важливо для збереження, отримання та обробки даних, які надходять від датчиків, які відстежують екологічні показники. З'єднання з базою даних є ключовою складовою для програми моніторингу, оскільки це дає змогу зберігати зібрані дані та здійснювати з ними різноманітні операції: від простих запитів на вибірку даних до складних аналітичних операцій.

При побудові системи моніторингу екологічних показників цей код є першим кроком у взаємодії з базою даних. Він відкриває можливості для зберігання інформації про вимірювання та місцезнаходження датчиків, забезпечуючи можливість аналізу та моніторингу екологічних даних з метою прийняття обґрунтованих рішень щодо збереження навколишнього середовища.

Доцільність використання MySQL Connector/C++ або подібних бібліотек виявляється зручності з'єднання з базою даних, а також у можливості працювати з SQL-запитами для обробки та аналізу отриманих даних. Це створює базову інфраструктуру для впровадження програмного забезпечення для моніторингу екологічних показників міста.

2. Ініціалізація засобів для взаємодії з базою даних.

Взаємодія з базою даних у системі реалізована за допомогою бібліотеки SQLAPI++, яка спрощує взаємодію програмного коду з базою даних. Фрагмент коду на C++ з використанням SQLAPI++, зображений на рис. 2.17, демонструє ініціалізацію засобів для взаємодії з базою даних MySQL і виконує запит до бази даних для вибірки даних з таблиці `measuring_points` і виводу їх на монітор.

Використання SQLAPI++ спрощує процес взаємодії програми з базою даних, забезпечуючи високий рівень абстракції і дозволяючи писати менше коду для досягнення бажаних результатів. Також спрощується процес роботи

з базою даних, завдяки інтерфейсу для підключення до бази даних, виконання запитів та обробки результатів, що забезпечує швидку взаємодію зі сховищем.

```
int main() {
    SAConnection connection;

    try {
        connection.Connect(
            "MySQL",
            "localhost",
            "username",
            "password",
            SA_MySQL_Client);

        std::cout << "Connected to database successfully!" << std::endl;

        SACommand select(&connection, "SELECT * FROM measuring_points");
        select.Execute();

        while (select.FetchNext()) {
            std::cout << "ID: " << select.Field("id").asLong() << std::endl;
            std::cout << "Name: " << select.Field("name").asString() << std::endl;
            std::cout << "Location: " << select.Field("location").asString() << std::endl;
        }

        connection.Disconnect();
        std::cout << "Disconnected from database!" << std::endl;

    } catch (SAException &ex) {
        std::cout << "Database Exception: " << ex.ErrText().GetMultiByteChars() << std::endl;
    }

    return 0;
}
```

Рис. 2.17 – Ініціалізація засобів взаємодії з базою даних

У цьому коді:

- SAConnection – об'єкт, що представляє з'єднання з базою даних.
- SACommand – об'єкт для виконання SQL-запитів.
- SELECT * FROM measuring_points – SQL-запит на вибірку всіх даних з таблиці measuring_points.
- Execute() виконує запит до бази даних.
- FetchNext() переходить до наступного рядка результатів запиту.
- Field("column_name").asType() отримує значення поля з результатів запиту.

- `Connect()` – метод для підключення до бази даних з вказанням драйвера, адреси сервера, імені користувача, пароля та типу клієнта (у цьому випадку – для MySQL).
- `Disconnect()` – метод для розірвання з'єднання з базою даних.
- `SAException` – виключення, яке може виникнути під час роботи з базою даних.

Здійснення з'єднання з базою даних є ключовою складовою для збереження інформації, отриманої від датчиків, та надання можливості подальшої обробки цих даних, і застосування SQLAPI++ спростило процес взаємодії з БД, дозволяючи більше уваги приділяти розробці функціональності ПЗ, яке опрацьовує та аналізує отримані дані, замість витрат на рутинні операції.

3. Конфігурація підключення до сенсорів та зчитування даних з датчиків – важливий етап, який передбачає встановлення спільної комунікації між програмою та самими датчиками. Для цього використовуються спеціалізовані бібліотеки, які дозволяють взаємодіяти з різноманітними типами датчиків.

- **DHT22:**

- `DHT.h` – стандартна бібліотека для датчиків DHT від Adafruit.
- `Adafruit_Sensor.h` – бібліотека для датчиків від Adafruit.

- **MQ-135:**

- `MQ135.h` – бібліотека для датчиків MQ-135 від SparkFun.

- **GP2Y1010AU0F:**

- `GP2Y1010AU0F.h` – бібліотека для датчиків GP2Y1010AU0F від SparkFun.

- **pH Sensor Kit:**

- `DFRobot_PH.h` – бібліотека для датчиків pH від DFRobot.

- **DS18B20 Waterproof Temperature Sensor:**

- `OneWire.h` – бібліотека для датчиків температури OneWire.
- `DallasTemperature.h` – бібліотека для датчиків температури Dallas.

- **Atlas Scientific EZO Dissolved Oxygen Sensor:**
 - AtlasScientific.h – бібліотека для датчиків Atlas Scientific.
- **DFRobot Gravity: Analog pH Sensor / Meter:**
 - DFRobot_PH.h – бібліотека для датчиків pH від DFRobot.

Більшість бібліотек для датчиків, що використовуються в проєкті, доступні в стандартній бібліотеці Arduino. Для деяких датчиків, наприклад, MQ-135, потрібна бібліотека від стороннього розробника. Бібліотеки для датчиків температури OneWire і Dallas часто використовуються разом. Бібліотеки для датчиків Atlas Scientific і DFRobot Gravity є аналогічними.

Після вибору бібліотеки програма має встановити зв'язок з датчиком. Це може включати підключення датчика до конкретних пінів мікроконтролера, ініціалізацію необхідних налаштувань та інструкцій для взаємодії з датчиком через обрану бібліотеку. Далі для кожного конкретного типу датчика налаштовуються параметри, які впливають на зчитування даних. Це може включати частоту зчитування, точність вимірювань, обробку помилок тощо.

Після конфігурації програма проводить тестування зв'язку з датчиками. Це включає зчитування тестових даних, перевірку їх правильності та відповідність отриманих даних очікуваним значенням.

Зчитування даних з датчиків (температура, вологість тощо) відбувається у декілька етапів, орієнтованих на кожен конкретний тип датчика. Залежно від конкретного типу датчика, програма виконує відповідні команди для отримання значень. На прикладі, наведеному в рис. 2.18, виконується аналіз рівня амоніаку за допомогою газового сенсора MQ-135. Програмно зчитати дані з цього сенсора можна через аналоговий вхід мікроконтролера Arduino.

```

int sensorPin = A0; // Підключення датчика до аналогового піна A0

void setup() {
  Serial.begin(9600); // Ініціалізація передачі даних через Serial Monitor
}

void loop() {
  int ammoniaLevel = analogRead(sensorPin); // Зчитування аналогового сигналу з датчика
  Serial.print("Рівень амоніаку: ");
  Serial.println(ammoniaLevel); // Вивід отриманого значення в Serial Monitor
  delay(1000); // Затримка для читання кожну секунду (можна змінювати)
}

```

Рис. 2.18 – Аналіз даних, зчитаних з газового сенсора

Цей фрагмент коду використовує `analogRead()` для зчитування аналогового сигналу з піна, до якого підключений датчик. Значення, отримане з `analogRead()`, представляє аналогове значення, що відповідає рівню амоніаку. Ці дані виводяться у Serial Monitor через `Serial.print()` та `Serial.println()` для подальшого моніторингу або збереження.

Отримані дані можуть бути у вигляді аналогових сигналів, цифрових значень або специфічних кодів [30]. Програма виконує конвертацію цих даних у зручний формат для подальшої обробки та аналізу. Вона починає з визначення типу даних, отриманих від датчиків, що є важливим для правильної конвертації у формат, придатний для зберігання в базі даних. Наприклад, числові дані, такі як температура, вологість та концентрація CO₂, можуть бути збережені у форматі плаваючої точки. Після цього дані піддаються очищенню, оскільки вони можуть містити шум або помилки, що може вплинути на точність результатів. Наприклад, застосування фільтрів допомагає в прибиранні шумів, аномальних значень або помилок, що можуть вплинути на точність отриманих даних. Це може включати застосування різних методів фільтрації, таких як середнє значення, медіана, експоненціальне згладжування тощо, для виявлення та видалення аномалій або шумів, які можуть змінити точність отриманих даних. Такі фільтри допомагають забезпечити більш стабільні та точні вимірювання перед подальшою обробкою даних. Після очищення даних вони конвертуються у формат, придатний для зберігання в базі даних, такий як JSON або XML, щоб

мати можливість зберігати та використовувати їх у подальших аналітичних процесах.

У рис. 2.19 наведений фрагмент коду, у якому температура отримується за допомогою функції `getTempCByIndex()` бібліотеки `DallasTemperature.h`. і конвертується у формат JSON за допомогою функції `String()`. Окрім бібліотеки `DallasTemperature.h`, включаються бібліотеки `OneWire.h` та `ArduinoJson.h` для отримання значення температури через датчик температури, підключений до піна 2. Після отримання значення температури, воно зберігається у змінній `tempC`, а потім додається до JSON-об'єкта. Функція `serializeJson()` з `ArduinoJson.h` використовується для конвертації цього об'єкта JSON в рядок формату JSON (`jsonString`), який потім виводиться у Serial Monitor.

```
#include <DallasTemperature.h>
#include <OneWire.h>
#include <ArduinoJson.h>

#define ONE_WIRE_BUS 2 // Пін для зчитування даних датчика температури

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup() {
  Serial.begin(9600);
  sensors.begin(); // Ініціалізація датчиків температури
}

void loop() {
  sensors.requestTemperatures(); // Запит на оновлення значень температури
  float tempC = sensors.getTempCByIndex(0); // Отримання температури

  // Створення об'єкта JSON
  StaticJsonDocument<100> jsonDoc;
  jsonDoc["temperature"] = tempC;

  String jsonString;
  serializeJson(jsonDoc, jsonString); // Конвертація в JSON-стрічку

  Serial.println(jsonString); // Виведення JSON-стрічки у Serial Monitor

  delay(1000); // Затримка для зчитування кожну секунду
}
```

Рис. 2.19 – Отримання температури та конвертація в формат JSON

4. Отримання географічного положення сенсорів та пунктів моніторингу. Для цього використовується таблиця `measuring_points`, яка містить ідентифікатори, назви та географічні координати пунктів моніторингу.

Пункти моніторингу:

- Ідентифікатори: В таблиці `measuring_points` є унікальні ідентифікатори для кожного пункту моніторингу.
- Географічне положення: Координати цих пунктів можуть бути збережені в окремих стовпцях таблиці, які представляють широту та довготу місця розташування сенсорів.

Сенсори:

- Прикріплення до пунктів моніторингу: У таблиці `sensors` існує зв'язок з `measuring_points` через ідентифікатор пункту моніторингу. Це дозволяє прикріпити сенсор до певного місця моніторингу.
- Географічне положення сенсорів: Додатково до координат пунктів моніторингу, зберігаються координати самого сенсора, що допомагає визначити точне місцезнаходження сенсора на пункті моніторингу.

Таким чином, таблиця `measuring_points` використовується як джерело географічних координат для пунктів моніторингу, а саме ідентифікатори цих точок використовуються для прикріплення сенсорів. Географічні дані пунктів моніторингу можуть бути використані для картографічного відображення результатів моніторингу або для аналізу просторових взаємозв'язків між показниками, отриманими від різних сенсорів на різних місцях.

5. Збереження отриманих вимірювань в таблицю `measurements` разом із відповідними ідентифікаторами пункту моніторингу. При додаванні нових записів необхідно включати відповідний ідентифікатор пункту моніторингу, щоб вказати місце, де вони були здійснені. Це може виглядати, як на рис. 2.20.

```
INSERT INTO measurements (measuring_point_id, measurement_date,  
temperature, humidity, ammonia, carbon_dioxide, ...)  
VALUES (1, '2023-11-28 08:00:00', 25.5, 60, 0.02, 450, ...);
```

Рис. 2.20 – Фрагмент збереження вимірювань в базу даних

У цьому прикладі `measuring_point_id` вказує на ідентифікатор конкретного пункту моніторингу (в даному випадку, це пункт з ідентифікатором 1). Окрім ідентифікатора пункту моніторингу, зберігаються

дані про дату вимірювання та конкретні показники, такі як температура, вологість, рівень амоніаку, вуглекислого газу тощо.

Цей підхід дозволяє відстежувати, які саме вимірювання були проведені у певному пункті моніторингу та в який час, що спрощує подальший аналіз та використання цих даних.

6. Налаштування програми для регулярного (за потребою) зчитування даних з датчиків та їх збереження у базі даних. Для регулярного зчитування даних з датчиків та збереження їх у базі даних програма спочатку встановлює певний розклад опитування датчиків. Цей розклад може бути встановлений за допомогою таймера або використанням вбудованих функцій часу в залежності від необхідності зчитування. У випадку системи, що розроблювалася, реалізовано регулярне зчитування даних, реалізація чого показана на рис. 2.21.

```
#define DHTPIN 2      // Пін, на якому підключений сенсор DHT22
#define DHTTYPE DHT22 // Вказуємо тип сенсора

DHT dht(DHTPIN, DHTTYPE);
unsigned long previousMillis = 0;
const long interval = 300000; // Інтервал зчитування (5 хвилин)

MySQL_Connection conn((Client *)&Serial);
MySQL_Cursor *cursor;

void setup() { ...
}

void loop() {
    unsigned long currentMillis = millis();

    if (currentMillis - previousMillis >= interval) {
        previousMillis = currentMillis;

        float humidity = dht.readHumidity();
        float temperature = dht.readTemperature();

        char INSERT_SQL[100];
        sprintf(INSERT_SQL,
            "INSERT INTO measurements (temperature, humidity) VALUES (%f, %f)", temperature, humidity);

        cursor->execute(INSERT_SQL);
        Serial.println("Data saved to database!");
    }
}
```

Рис. 2.21 – Реалізація ітеративного зчитування даних

У цьому фрагменті коду відбувається зчитування температури та вологості з сенсора DHT22 (AM2302) кожні 5 хвилин (300000 мс). Отримані

дані (температура та вологість) вставляються в таблицю `measurements` бази даних MySQL, яка має поля `temperature` та `humidity`.

Програма спочатку встановлює з'єднання з базою даних, а потім у циклі `loop()` перевіряє, чи пройшов заданий інтервал. Якщо так, вона зчитує дані з датчика, формує SQL-запит для вставки цих даних у таблицю бази даних та виконує його.

Після встановлення графіку опитування виконуються заплановані дії зчитування даних з датчиків. Ця операція включає активацію сенсорів та отримання вимірювань їхніх показників.

Отримані дані проаналізовані та готові до збереження у базі даних. Далі виконуються процедури збереження цих даних у відповідних таблицях з дотриманням взаємозв'язків між різними показниками та пунктами моніторингу. Після збереження, програма готова повторити цей процес зчитування та збереження даних з датчиків відповідно до заданого розкладу опитування.

2.4 Забезпечення захисту від втрати даних

Забезпечення захисту від втрати даних у випадку відключення датчиків або помилок під час зчитування та збереження інформації важливе для надійності системи моніторингу. У даному проєкті реалізована низка способів, які запобігають втраті інформації.

1. Використання механізмів перевірки доступності датчиків перед зчитуванням. Перевірка доступності датчиків перед зчитуванням даних є важливою складовою надійності системи моніторингу. Цей механізм дозволяє переконатися у наявності та працездатності датчиків перед початком зчитування інформації. Основна мета цього підходу – уникнути некоректних чи недостовірних вимірювань та запобігти виникненню помилок через недоступність датчиків.

Перевірка доступності датчиків включає такі етапи:

- Перевірка стану з'єднання з датчиками: перевірка фізичного з'єднання між датчиками та мікроконтролером Arduino, виявлення підключених пристроїв.

- Перевірка ініціалізації датчиків: перевірка успішності початкової ініціалізації датчиків, що може включати в себе відповідний повідомлення про успішну ініціалізацію чи виявлення можливих проблем.

- Визначення готовності до зчитування: перевірка готовності датчиків до передачі даних, наприклад, через готовність до зчитування нових вимірювань чи статус готовності датчика до роботи.

Цей механізм дозволяє системі перед початком збору даних переконатися, що всі необхідні датчики працюють коректно та доступні для зчитування, зменшуючи ризик виникнення помилок та забезпечуючи надійність отриманих вимірювань [28].

2. Обробка винятків та помилок. Під час зчитування інформації з сенсорів можуть виникати ситуація, коли відсутня відповідь від датчика або передаються зіпсовані дані. Обробка винятків та помилок, пов'язаних із зчитуванням інформації з сенсорів, здійснюється за допомогою таких методів:

- Try-Catch блоки. Використовуються для обробки виняткових ситуацій. Код, який може викликати помилки, розміщується в ``try`` блоку, а обробка помилок – у ``catch`` блоку.

- Перевірка стану сенсорів. Перед спробою зчитування даних з сенсорів рекомендується перевіряти їх стан (наприклад, чи підключені вони до системи, чи правильно працюють тощо).

- Використання індикаторів помилок чи флагів. Записування прапорців або індикаторів, які вказують на те, що під час зчитування відбулася помилка. Це може допомогти в подальшому визначенні причини та відновленні роботи.

- Логування помилок. Фіксація подій та помилок у лог-файлах для подальшого аналізу та вирішення проблем.

- Перевірка коректності даних. Після зчитування даних виконується перевірка їх на коректність та відповідність очікуваним значенням чи

формату. Якщо дані виявляються некоректними, вони ігноруються або відзначаються як помилкові.

- Автоматичне відновлення. Реалізація механізму, який дозволяє спробувати знову зчитати дані після виникнення помилки, щоб дати можливість системі автоматично відновити роботу після короточасних помилок.

3. Використання буферів зберігання даних. При отриманні нових даних з датчиків, ці вимірювання зберігаються у внутрішній буфер пам'яті мікроконтролера. Перед записом у базу даних, програма перевіряє наявність нових даних у буфері та формує запити на вставку в базу лише для цих нових даних. Це дозволяє відправляти дані пакетами, що зменшує кількість запитів до бази даних, підвищуючи ефективність.

У випадку відключення з'єднання з базою даних, програма тимчасово зберігає нові вимірювання в EEPROM пам'яті. EEPROM (Electrically Erasable Programmable Read-Only Memory) – це тип пам'яті, яка забезпечує збереження даних після вимкнення живлення. Це відмінна від RAM, яка втрачає дані при вимкненні живлення. EEPROM здатна зберігати дані навіть після відключення живлення пристрою та може бути перепрограмована для зберігання нових даних. На рис. 2.22 показано, що коли з'єднання відновлюється, тимчасово збережені дані витягуються з носія та відправляються в базу даних.

Після відправлення тимчасово збережених даних в базу даних, програма виконує перевірку на успішність передачі. Якщо дані успішно вставлені, вони видаляються з EEPROM пам'яті, звільняючи місце для нових вимірювань. У випадку, якщо передача була невдалою, програма зберігає ці дані у спеціальному резервному буфері, щоб уникнути втрати інформації.

У випадку, якщо виникає помилка при відновленні з'єднання з базою даних під час спроби передачі тимчасово збережених даних, програма автоматично намагається відновити з'єднання визначену кількість разів з певним інтервалом. Це робиться для забезпечення надійності передачі та запобігання втраті даних.


```

#include <EEPROM.h>

void saveDataToEEPROM(float temperature, float humidity) {
    int address = 0; // Початкова адреса в EEPROM для збереження даних

    // Запис температури у EEPROM
    EEPROM.put(address, temperature);
    address += sizeof(float);

    // Запис вологості у EEPROM
    EEPROM.put(address, humidity);
}

void retrieveDataFromEEPROM(float &temperature, float &humidity) {
    int address = 0; // Початкова адреса в EEPROM, де зберігаються дані

    // Зчитування температури з EEPROM
    EEPROM.get(address, temperature);
    address += sizeof(float);

    // Зчитування вологості з EEPROM
    EEPROM.get(address, humidity);
}

```

Рис. 2.22 – Реалізація аварійного збереження даних

У наведеному фрагменті коду функція `saveDataToEEPROM()` приймає температуру та вологість як параметри та записує їх у пам'ять EEPROM, а функція `retrieveDataFromEEPROM()` зчитує температуру та вологість з EEPROM та повертає їх через посилання. Такий підхід значно підвищує надійність системи, особливо у випадках, коли можуть виникати переривання в зв'язку або проблеми під час збереження інформації.

3. Логування помилок та проблем. Запис подій, помилок та виняткових ситуацій у лог-файли є важливою частиною забезпечення надійності системи моніторингу [34]. Це дозволяє відстежувати роботу програми, виявляти проблеми та вчасно реагувати на них. Лог-файли містять інформацію про:

- Дату та час виникнення події, що дозволяє точно визначити момент виникнення проблеми.

- Опис події, помилки чи виняткової ситуації з достатньою деталізацією, щоб зрозуміти суть проблеми.
- Деталі про стан системи в момент виникнення події, наприклад, інформація про стан датчиків, з'єднання з базою даних чи стан програми.
- Дії, які були виконані перед виникненням проблеми, що може допомогти в реконструкції послідовності подій.
- Лог-файли повинні бути структуровані та деталізовані для спрощення подальшого аналізу. Рівень деталізації логів може відрізнятися залежно від важливості подій та вимог до аналізу. Кожен запис у лог-файлі повинен містити достатньо інформації для того, щоб легко ідентифікувати проблему та зрозуміти її причини. Приклад запису в лог-файлі наведений на рис. 2.23.

```
[2023-10-15 08:45:22] Помилка під час спроби зчитування температури з сенсора DHT22.
[2023-10-16 14:20:11] Сенсор DHT22 недоступний. Перевірте підключення та стан сенсора.
[2023-10-18 09:10:05] Не вдалося встановити з'єднання з базою даних. Перевірте параметри підключення.
[2023-10-20 17:30:55] Додано новий запис в таблицю measurements: температура = 25°C, вологість = 60%.
```

Рис. 2.23 – Реалізація логування подій

Такі записи дозволяють операторам або розробникам визначати причини проблеми та вживати відповідних заходів для виправлення недоліків у системі моніторингу.

4. Реалізація механізму резервного копіювання. Регулярне створення резервних копій бази даних – важливий елемент захисту інформації від втрати у випадку серйозних проблем [15]. Для цього автоматизується процес, який періодично створює копії бази даних і зберігає їх на іншому носії, що відокремлений від основної системи. Це забезпечує можливість відновлення інформації у разі її втрати через фізичні чи програмні проблеми.

Процес резервного копіювання виглядає наступним чином: модуль, що відповідає за цей процес бекапу, запускається за розкладом чи після виникнення певних подій, які вказують на необхідність створення резервної копії [37]. Спочатку здійснюється з'єднання з базою даних, а потім здійснюється виконання команд та запитів на створення резервної копії, що продемонстровано на рис. 2.24.

```

IPAddress serverAddr(193, 45, 82, 246); // IP-адреса сервера бази даних
char user[] = "username"; // Логін користувача бази даних
char password[] = "password"; // Пароль користувача бази даних
char db[] = "measuring_points"; // Назва бази даних

MySQL_Connection conn((Client *)&Serial);
MySQL_Cursor *cursor;

void setup() { ...
}

void loop() {
    // Резервне копіювання запускається за запитом або за розкладом
    if (needBackup()) {
        backupDatabase();
    }
    delay(10000); // Перевірка резервного копіювання кожні 10 секунд
}

void backupDatabase() {
    Serial.println("Creating database backup...");
    char BACKUP_SQL[] = "BACKUP DATABASE TO '/backup/measuring_points_backup.sql'";
    cursor->execute(BACKUP_SQL);
    Serial.println("Backup created!");
}

bool needBackup() {
    return true; // Повертаємо true, щоб завжди створювати резервну копію
}

```

Рис. 2.24 – Виконання команд на створення резервної копії

У наведеному фрагменті коду використовується бібліотека MySQL_Connector для з'єднання з базою даних MySQL та створення резервної копії. При цьому:

- У функції loop() виконується перевірка умови для створення резервної копії (функція needBackup()). В даному прикладі ця умова завжди повертає true, що означає, що резервна копія створюватиметься завжди.

- Якщо умова для створення резервної копії виконується, запускається функція backupDatabase(), яка виконує SQL-запит для створення резервної копії бази даних.

- Механізм функції needBackup() може бути налаштований для визначення умови створення резервної копії на основі реальних умов, таких як

час, кількість записів чи інші параметри. За замовчуванням функція повертає значення true.

Отримана резервна копія зберігається на окремому сервері, щоб у разі втрати основних даних через непередбачувані обставини, інформація залишалась доступною для відновлення. Для ефективного захисту даних підтримується регулярність процесу створення резервних копій, з урахуванням обсягу даних, частоти змін і вимог до часу відновлення системи.

РОЗДІЛ 3. АНАЛІЗ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1. Визначення критеріїв тестування

Тестування системи моніторингу екологічних показників є критичним етапом розробки, оскільки визначає надійність, ефективність та точність системи в реальних умовах експлуатації. Ця важливість пояснюється кількома ключовими аспектами.

По-перше, екологічні дані, які система збирає та обробляє, мають велике значення для здоров'я і безпеки людей, а також стану природи. Недостовірні чи неточні вимірювання можуть призвести до невірних висновків і, в кінцевому підсумку, до прийняття неправильних рішень в галузі охорони навколишнього середовища та здоров'я.

Другим важливим аспектом є те, що система моніторингу повинна бути надійною та стійкою до різноманітних умов, таких як зміни в навколишньому середовищі, атмосферні впливи, атаки з боку зловмисників чи технічні збої. Тестування дозволяє визначити, наскільки система може витримати ці впливи та як ефективно вона реагує на непередбачені ситуації.

Не менш важливим фактором є й те, що в процесі експлуатації система взаємодіє з різними компонентами, такими як датчики, бази даних, засоби звітування тощо. Тестування дозволяє визначити сумісність цих компонентів, виявити можливі конфлікти та удосконалити їх взаємодію для забезпечення безперебійної роботи системи.

Отже, тестування системи моніторингу екологічних показників визначає її надійність, точність та функціональність, гарантуючи, що зібрані дані є достовірними, а сама система в змозі ефективно функціонувати в реальних умовах з врахуванням динаміки природних та технічних впливів.

Тестування ефективності роботи різних типів сенсорів у системі моніторингу екологічних показників включає ряд критеріїв, які оцінюють їхню точність, стійкість та здатність працювати в різних умовах. Ось критерії для кожного типу сенсора:

1. Датчики температури та вологості (наприклад, DHT22).

- Точність вимірювань: Перевірка відповідності вимірювань дійсним значенням температури та вологості.
- Стійкість до змін умов: Тестування реакції на стрімкі зміни температури або вологості.
- Споживана потужність: Вимірювання енергоспоживання для оптимізації використання енергії.

2. Датчики рівня амоніаку, вуглекислого газу, сірководню та інших газів.

- Точність виявлення газів: Перевірка точності виявлення та вимірювання концентрації різних газів.
- Стійкість до зовнішніх впливів: Оцінка роботи сенсорів під впливом інших газів, температурних та вологості змін.
- Час відгуку: Визначення часу, необхідного сенсору для реагування на зміну концентрації газу.

3. Датчики рівня шуму.

- Точність вимірювань рівня шуму: Перевірка відповідності вимірювань реальному рівню шуму в навколишньому середовищі.
- Стійкість до інтерференцій: Оцінка впливу інших звуків та шумів на роботу сенсора.
- Час відгуку: Вимірювання часу, необхідного для сенсора реагувати на зміни рівня шуму.

4. Датчики рівня забруднення повітря (пилу, оксидів азоту тощо).

- Точність вимірювань: Перевірка відповідності вимірювань реальному рівню забруднення повітря.
- Стійкість до зовнішніх факторів: Оцінка впливу погодних умов, інших газів чи частинок на роботу сенсора.
- Час відгуку: Визначення часу, необхідного для сенсора реагувати на зміни рівня забруднення повітря.

5. Датчики водних параметрів (рН, температура, рівень розчиненого кисню тощо).

- Точність вимірювань: Перевірка відповідності вимірювань реальним значенням параметрів води.
- Стійкість до змін хімічного складу: Оцінка роботи сенсорів під впливом різних речовин у воді.
- Час відгуку: Вимірювання часу, необхідного для сенсора реагувати на зміни параметрів води.

Тестування за такими критеріями дозволяє забезпечити оптимальну ефективність та надійність роботи системи моніторингу екологічних показників у різних умовах та враховувати специфіку кожного типу сенсора.

Загальні критерії для ефективності системи моніторингу екологічних показників охоплюють широкий спектр аспектів, що визначають її функціональність та надійність в реальних умовах. Перш за все, система повинна забезпечувати точне та достовірне вимірювання різноманітних екологічних показників, щоб забезпечити надійну основу для прийняття рішень у сфері охорони навколишнього середовища та здоров'я.

Ефективність системи також оцінюється її здатністю працювати стійко та безперервно в різних умовах, включаючи зміни в погоді, технічні або програмні збої, а також вплив інтерференцій та зовнішніх факторів на роботу сенсорів.

Час відгуку системи на зміни та здатність швидко реагувати на важливі екологічні події є ще одним важливим аспектом. Оптимізація часу обробки та передачі даних дозволяє забезпечити оперативність системи та негайне інформування стосовно змін у середовищі.

Додатково, ефективність вимагає гнучкості та сумісності з різними типами сенсорів та пристроїв, щоб система могла адаптуватися до різноманітних вимог та умов моніторингу.

Забезпечення безпеки та захисту від несанкціонованого доступу до даних є також важливими аспектами ефективності системи, особливо в контексті чутливих екологічних даних.

Таким чином, загальні критерії визначаються високою точністю вимірювань, стійкістю до змін умов, оперативністю реакції, гнучкістю та безпекою для системи моніторингу екологічних показників.

3.2. Дослідження ефективності та надійності системи

Для оцінки ефективності програми моніторингу екологічних показників застосовано комплексний підхід, який включає в себе ряд параметрів та методів вимірювання.

Дослідження швидкодії. У табл. 3.1 продемонстровано результати дослідження швидкодії програми моніторингу екологічних показників через вимірювання часу відгуку системи на різні типи запитів, часу обробки та збереження даних у базі даних. Запуск програми здійснювався під навантаженням різної складності та об'єму даних для вимірювання часу відгуку, часу обробки даних та їх збереження.

Таблиця 3.1 – Дослідження швидкодії системи моніторингу

Тест №	Тип запиту	Час відгуку (ms)	Час обробки (ms)	Час збереження в БД (ms)
1	Отримання температури повітря	25	10	15
2	Вимірювання вологості повітря	35	15	20
3	Аналіз рівня забруднення повітря	28	12	16
4	Концентрація пилу	30	14	16
5	Вимірювання рН води	40	18	22

Ця таблиця відображає результати проведених тестів, під час яких вимірювалися час відгуку системи на різні типи запитів, час обробки та збереження даних у базі даних.

Аналіз результатів:

1. **Час відгуку:** Цей параметр відображає час, який потрібен програмі для надання відповіді на запит. На основі таблиці видно, що середній час відгуку системи на запити становить приблизно 32 мс.
2. **Час обробки:** Цей показник показує час, який програма витрачає на обробку отриманих даних перед їхнім збереженням. З таблиці видно, що середній час обробки становить близько 13 мс.
3. **Час збереження в БД:** Цей параметр відображає час, необхідний для збереження отриманих даних у базі даних. У середньому програмі потрібно приблизно 18 мс для збереження даних.

На підставі аналізу швидкодії вдалося дійти висновку, що загальний час обробки запиту, який складається з часу відгуку, часу обробки та часу збереження в БД, є невеликим (така оцінка ґрунтується на порівняльному аналізі цього часу зі стандартами й очікуваними значеннями в контексті подібних систем моніторингу), що свідчить про достатню швидкість відповіді системи на зовнішні запити. Однак, можливості для оптимізації присутні у часі обробки та збереження в БД, де можна спробувати покращити продуктивність програми шляхом оптимізації операцій обробки та збереження даних.

Дослідження відмовостійкості. У цьому тесті визначалася частота виникнення помилок під час зчитування даних з датчиків, а також стійкість програми до втрати з'єднання з базою даних чи пристроями збору інформації.

В табл. 2 наведено показано результати тестів з надійності системи під час стрес-тестування. Частота помилок при зчитуванні даних з датчиків була виміряна для всіх типів датчиків.

Таблиця 3.2 – Частота виникнення помилок під час зчитування даних з датчиків

Тип датчика	Кількість тестів	Кількість помилок	Частота помилок
Датчик температури повітря	200	8	4%
Датчик вологості повітря	200	4	2%
Датчик часток пилу в повітрі	200	0	0%
Датчик концентрації газових сполук	200	12	6%
Датчик рівня рН води	200	0	0%
Датчик температури води	200	4	2%
Датчик рівня розчиненого кисню	200	0	0%
Датчик рівня рН ґрунту	200	8	4%

Окрім того, фіксувалася стійкість системи до втрати з'єднання з базою даних та датчиками, що показано в табл. 3.3.

Таблиця 3.3 – Стійкість програми до втрати з'єднання

Умова тесту	Кількість тестів	Кількість відмов	Відсоток відмов
Втрата з'єднання з базою даних	200	9	4,5%
Втрата з'єднання з датчиками	200	0	0%

На основі результатів стрес-тестування системи моніторингу з різних типів датчиків можна зробити наступні висновки:

1. Висока стабільність багатьох датчиків: Більшість датчиків, таких як датчики вологості повітря, часток пилу, рівня рН води, рівня розчиненого кисню у воді, демонструють низьку частоту помилок (0-2%). Це свідчить про їхню стабільність та надійність під час зчитування даних.

2. Потенційні проблеми з надійністю окремих датчиків: Деякі типи датчиків, такі як датчики концентрації газових сполук та рівня рН ґрунту, показують високу частоту помилок (4-8%). Це може вказувати на потенційні

проблеми з їхньою надійністю під час роботи або чутливість до певних умов.

3. Значна варіабельність результатів. Різниця у частоті помилок між різними типами датчиків вказує на варіабельність їхньої надійності. Це може бути пов'язане з якістю самого датчика, умовами використання чи його технічними особливостями.

4. Потреба у додатковому аналізі. Виявлені помилки можуть потребувати додаткового дослідження для визначення їхніх причин. Деякі помилки можуть бути пов'язані з технічними аспектами зчитування, умовами вимірювання або специфікою даних, що надходять від датчиків.

На основі результатів стрес-тестування системи моніторингу з різних типів датчиків виокремлюються ключові висновки. Виявлено високу стабільність багатьох датчиків, таких як датчики вологості повітря, часток пилу, рівня рН води та рівня розчиненого кисню у воді, що свідчить про їхню надійність та стабільність в процесі зчитування даних. Такі результати є обнадійливими, оскільки вони вказують на ефективність цих ключових компонентів системи.

З іншого боку, виявлено потенційні проблеми з надійністю окремих датчиків, зокрема датчиків концентрації газових сполук та рівня рН ґрунту, які показали високу частоту помилок. Це виглядає як сигнал для подальшого вивчення та можливих корекцій у функціонуванні цих конкретних компонентів системи.

Загалом, виявлена значна варіабельність результатів стрес-тестування вказує на потребу у додатковому аналізі. Різниця у частоті помилок між різними типами датчиків може бути пов'язана з різними факторами, такими як якість датчика, умови використання чи технічні особливості. Детальне дослідження цих аспектів є важливим для подальшого вдосконалення ефективності та надійності системи моніторингу екологічних показників.

ВИСНОВКИ

Сучасні міста стикаються з численними екологічними проблемами, які впливають на якість життя населення та стан навколишнього середовища. Для вирішення цих проблем необхідно мати достовірну та актуальну інформацію про стан екології міста, яка може бути отримана за допомогою систем моніторингу. Системи моніторингу екології міста - це комплексні технічні та програмні рішення, які дозволяють вимірювати, збирати, обробляти та аналізувати дані про різні параметри екології міста, такі як якість повітря, води, ґрунту, шуму тощо.

У першому розділі дослідження проведено докладний огляд існуючих систем моніторингу, спрямованих на контроль екологічних показників міста, визначено та виконано аналіз систем контролю параметрів повітря, води, ґрунту та шуму виявили різноманітні підходи та технології, які використовуються в сучасних системах моніторингу.

За результатами огляду були визначені переваги існуючих систем. Деякі з них відзначаються високою точністю вимірювань та здатністю надавати деталізовану інформацію про різні аспекти екологічного стану міста. Деякі системи також використовують передові технології для автоматизації та забезпечення швидкого реагування на зміни в показниках. З іншого боку, виявлено істотні недоліки, які можуть обмежити ефективність і надійність деяких систем. Деякі з систем можуть стикатися з проблемами в калібруванні, виявляти недостатню стійкість до зовнішніх факторів або обмежену гнучкість у вимірюванні різноманітних параметрів.

В другому розділі магістерської роботи було представлено процес побудови бази даних та вибору архітектури для системи моніторингу екологічних показників міста. Процес побудови бази даних включав в себе визначення структури таблиць, розробку зв'язків між ними та вибір оптимальних типів даних для зберігання різноманітних параметрів екологічного моніторингу.

Особлива увага була приділена вибору архітектури системи, що допомагає забезпечити ефективне зберігання та обробку великої кількості даних, а також забезпечити швидкий доступ до необхідної інформації.

Аналіз фізичного підключення та інтеграції датчиків у програмне забезпечення виявив важливі аспекти, такі як стійкість до втрат даних, ефективність передачі інформації та забезпечення безперебійної роботи системи навіть у випадках технічних або програмних неполадок.

Подання результатів програмної реалізації системи включало в себе опис розроблених функцій для зчитування та аналізу даних. Ця частина дослідження дозволила не лише визначити ефективність розробленого програмного забезпечення, а й відобразити його можливості у забезпеченні високоякісного та надійного моніторингу екологічних показників міста.

У третьому розділі було визначено й застосовано критерії для тестування системи моніторингу екологічних показників міста. Цей етап дозволив об'єктивно оцінити різні аспекти функціонування системи та визначити її ефективність відповідно до поставлених завдань.

Аналіз результатів тестування надав глибоке розуміння роботи системи в умовах реального використання. Була врахована реакція системи на зміни в параметрах моніторингу, її стійкість до внутрішніх і зовнішніх факторів, а також час відгуку на запити та точність передачі даних.

Результати, отримані під час аналізу, дозволили зробити об'єктивний висновок щодо ефективності системи моніторингу. Оцінка виконання системи була здійснена на основі обраних критеріїв, що враховували важливі аспекти, такі як точність вимірювань, швидкість реакції та надійність передачі даних.

Загалом, система моніторингу продемонструвала задовільну ефективність та придатність для практичного використання в умовах міського екологічного контролю. Однак, з урахуванням отриманих результатів, рекомендується подальше вдосконалення системи для оптимізації її роботи та розширення можливостей для подальшого використання у реальних умовах.

Розглядаючи можливості розвитку та вдосконалення нашої системи моніторингу екологічних показників міста, визначається кілька ключових напрямків, які можуть покращити її ефективність та розширити функціонал. Впровадження технологій, таких як штучний інтелект, машинне навчання та аналіз великих даних, може значно підвищити точність та швидкість обробки інформації. Це сприятиме більш деталізованому аналізу екологічних показників та покращить здатність системи передбачати можливі зміни.

Інтеграція нових типів датчиків, спрямованих на вимірювання специфічних показників, може розширити спектр моніторингу. Наприклад, впровадження сучасних датчиків для виявлення забруднення повітря новими речовинами або вдосконалення систем для вимірювання біологічних показників може забезпечити більш повну картину екологічної ситуації.

Отримані результати дозволяють стверджувати, що розроблена система є ефективним інструментом для моніторингу низки екологічних параметрів у місті. Вона не лише забезпечує точний та швидкий збір даних, але й є гнучкою та готовою до впровадження нових технологій та методик у майбутньому.

Перспективи для розвитку системи включають у себе вдосконалення алгоритмів обробки даних, використання передових технологій та нових типів датчиків, а також активну участь у великих міжнародних дослідженнях з охорони навколишнього середовища.

Таким чином, розроблена система відповідає завданням дослідження та може слугувати важливим інструментом для влади та громадськості у здійсненні контролю та прийнятті обґрунтованих рішень щодо екологічної ситуації у місті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Антонова, Г. В., А. В. Кедич, О. В. Ковирьова. "Інтернет речей та бездротові смарт-мережі в точному землеробстві." Комп'ютерні засоби, мережі та системи (2019).
2. Боголюбов В., Клименко М., Мокін В. Моніторинг довкілля. Вінниця, 2010. 232 с.
3. Бордюг, Н. С., А. В. Ращенко, and А. В. Лесь. "Розробка проєкту системи моніторингу атмосферного повітря." Екологічні науки 2 (2021): 35.
4. Зацерковний В.І. Геоінформаційні системи і бази даних: Монографія. Кн. 2 / В.І. Зацерковний, В.Г. Бурачек, О.О. Железняк, А.О. Терещенко. – Ніжин: НДУ ім. М. Гоголя, 2017. – 237 с.
5. Каменева І. П. Комплексний аналіз екологічної безпеки міста на основі сучасних ГІС-технологій /І. П. Каменева, Я. В. Яцишин, Д. О. Полішко, О. О. Попов // Екологія довкілля та безпека життєдіяльності. – 2008. –№ 5. – С. 41–46.
6. Караєва, Н. В., Л. О. Левченко, and Я. М. Трохименко. "Методологія розробки системи моніторингу рівня сталого розвитку та економічної безпеки України." Управління розвитком складних систем 5 (2011): 111-116.
7. Ломазов, Павло Костянтинович. "Удосконалення системи екологічного моніторингу атмосферного повітря міста Дніпро." (2020).
8. Мкртчян О. Геоінформаційне моделювання в конструктивній географії. Львів, 2010. 119 с.
9. Мухін В., Крижановський Є. Геоінформаційні системи в екології. Вінниця, 2014. 192 с.
10. Николук, Ольга, Андрій Лапін, and Інна Грінчук. "Модель системи екологічного моніторингу міста." Наукові перспективи (Naukovі perspektivi), 7 (25) (2022).
11. Павлова С. О., Сарибога Г. В. Віртуальна лабораторія екологічного моніторингу довкілля. З використанням роботизованих систем. Електронна та

акустична інженерія. 2020. Т. 3, № 4. С. 10–13. DOI: <https://doi.org/10.20535/2617-0965.2020.3.4.198592>.

12. Сергієнко, Л. В. "Екологічні наслідки урбанізації в системі загроз безпеці урбанізованим територіям." *Право та державне управління* 4 (2021): 147-158.

13. Удовенко, С. Г., et al. "Екологічний моніторинг ландшафтних ділянок з використанням регуляризованих штучних нейронних мереж." *Біоніка інтелекту* 1.94 (2020): 13-22.

14. Чмуж, В. С. "Методи розподілу обчислювальних ресурсів в інтелектуальній системі екологічного моніторингу." (2023).

15. Шевченко, Р. Ю. «Мобільна геоінформаційна система екологічного моніторингу міста Києва як науково-методологічна модель запобігання ризикам антропогенного впливу» – Режим доступу: <http://www.ecoj.dea.kiev.ua/archives/2019/2/11.pdf> (дата звернення: 09.03.2023).

16. Шкрібляк, Юрій Миколайович. "Модель та засоби програмно-апаратної системи контролю шумового забруднення розумного міста." (2019).

17. Що таке IoT-платформа або Інтернет речей для аграрія | Агробізнес-Україна [Електронний ресурс] // Агробізнес-Україна. – Режим доступу: <https://agrobusiness.com.ua/shcho-take-iot-platforma-abo-internet-rechei-dlia-ahrariia> (дата звернення: 24.02.2023).

18. Яновський, Ф. Й. "Сучасні та перспективні локаційні датчики в системах Інтернету речей (IoT)." (2023).

19. Akhter, Fowzia, et al. "Design and development of an IoT enabled pedestrian counting and environmental monitoring system for a smart city." 2019 13th International Conference on Sensing Technology (ICST). IEEE, 2019.

20. A. Siamwala, Z. Lochhead and W. Abdulla, "Environmental Noise Monitoring Using Distributed IoT Sensor Nodes," 2019 International Conference on Electronics, Information, and Communication (ICEIC), Auckland, New Zealand, 2019, pp. 1-10, doi: 10.23919/ELINFOCOM.2019.8706473.

21. Bacco, Manlio, et al. "Environmental monitoring for smart cities." *IEEE Sensors Journal* 17.23 (2017): 7767-7774.
22. Brun-Laguna, Keoma, et al. "SOL: An end-to-end solution for real-world remote monitoring systems." 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC). IEEE, 2016.
23. B. Siregar, A. B. Azmi Nasution and F. Fahmi, "Integrated pollution monitoring system for smart city," 2016 International Conference on ICT For Smart Society (ICISS), Surabaya, Indonesia, 2016, pp. 49-52, doi: 10.1109/ICTSS.2016.7792847.
24. Buelvas, J., Múnera, D., Tobón V., D.P. et al. Data Quality in IoT-Based Air Quality Monitoring Systems: a Systematic Mapping Study. *Water Air Soil Pollut* 234, 248 (2023). <https://doi.org/10.1007/s11270-023-06127-9>
25. de Camargo, Edson Tavares, et al. "Low-Cost Water Quality Sensors for IoT: A Systematic Review." *Sensors* 23.9 (2023): 4424.
26. D. Gunatilaka, "An IoT-Enabled Acoustic Sensing Platform for Noise Pollution Monitoring," 2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 2021, pp. 0383-0389, doi: 10.1109/UEMCON53757.2021.9666534.
27. Geetha, S., Gouthami, S. Internet of things enabled real time water quality monitoring system. *Smart Water* 2, 1 (2016). <https://doi.org/10.1186/s40713-017-0005-y>
28. Jurdak, Raja, Abdelhamid Nafaa, and Alessio Barbirato. "Large scale environmental monitoring through integration of sensor and mesh networks." *Sensors* 8.11 (2008): 7493-7517.
29. Kiyan, Amirhossein, et al. "A comprehensive platform for air pollution control system operation in smart cities of developing countries: A case study of Tehran." *Environmental Industry Letters* 1.1 (2023): 10-27.

30. Kousis, I., M. Manni, and A. L. Pisello. "Environmental mobile monitoring of urban microclimates: A review." *Renewable and Sustainable Energy Reviews* 169 (2022): 112847.
31. K. Sakthi, Y. Mohamed Sarim Zain, S. Manoj Shakthi Raj and M. Manickavasakar, "IoT based Soil Monitoring and Control Systems," 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2023, pp. 516-522, doi: 10.1109/ICSSIT55814.2023.10061141.
32. Malche, Timothy, Priti Maheshwary, and Rakesh Kumar. "Environmental monitoring system for smart city based on secure Internet of Things (IoT) architecture." *Wireless Personal Communications* 107.4 (2019): 2143-2172.
33. Marques, Gonçalo, et al. "Internet of things and enhanced living environments: measuring and mapping air quality using cyber-physical systems and mobile computing technologies." *Sensors* 20.3 (2020): 720.
34. M. Bacco, F. Delmastro, E. Ferro and A. Gotta, "Environmental Monitoring for Smart Cities," in *IEEE Sensors Journal*, vol. 17, no. 23, pp. 7767-7774, 1 Dec.1, 2017, doi: 10.1109/JSEN.2017.2722819.
35. Montori, Federico, Luca Bedogni, and Luciano Bononi. "A collaborative internet of things architecture for smart cities and environmental monitoring." *IEEE Internet of Things Journal* 5.2 (2017): 592-605.
36. Moursi, A.S., El-Fishawy, N., Djahel, S. et al. An IoT enabled system for enhanced air quality monitoring and prediction on the edge. *Complex Intell. Syst.* 7, 2923–2947 (2021). <https://doi.org/10.1007/s40747-021-00476-w>
37. Rödl, A., Arlati, A. A general procedure to identify indicators for evaluation and monitoring of nature-based solution projects. *Ambio* 51, 2278–2293 (2022). <https://doi.org/10.1007/s13280-022-01740-0>
38. Saini, Jagriti, Maitreyee Dutta, and Gonçalo Marques. "Indoor air quality monitoring systems based on internet of things: A systematic review." *International journal of environmental research and public health* 17.14 (2020): 4942.

39. Soil Monitoring with IoT - Smart Agriculture [Electronic resource] // Manx Technology Group. – Mode of access: <https://www.manxtechgroup.com/soil-monitoring-with-iot-smart-agriculture/> (date of access: 23.02.2023).
40. Solecki, W., Rosenzweig, C. Indicators and monitoring systems for urban climate resiliency. *Climatic Change* 163, 1815–1837 (2020). <https://doi.org/10.1007/s10584-020-02947-4>
41. Soni, P., Joshi, K., Vyas, A. (2022). IoT-Based Unique Air and Noise Pollution Monitoring System. In: Moh, M., Sharma, K.P., Agrawal, R., Garcia Diaz, V. (eds) *Smart IoT for Research and Industry*. EAI/Springer Innovations in Communication and Computing. Springer, Cham. https://doi.org/10.1007/978-3-030-71485-7_2
42. SONYC – Sounds of New York City [Electronic resource]. – Mode of access: <https://wp.nyu.edu/sonyc/> (date of access: 10.02.2023).
43. What is a smart environmental monitoring? Definition and Details [Electronic resource] // Paessler - The Monitoring Experts. – Mode of access: <https://www.paessler.com/it-explained/smart-environmental-monitoring> (date of access: 08.03.2023).
44. Wong, Man Sing, et al. "Towards a smart city: Development and application of an improved integrated environmental monitoring system." *Sustainability* 10.3 (2018): 623.
45. Wu, Yin, Zenan Yang, and Yanyi Liu. "Internet-of-Things-Based Multiple-Sensor Monitoring System for Soil Information Diagnosis Using a Smartphone." *Micromachines* 14.7 (2023): 1395.
46. Zulkifli, Che Zalina, et al. "IoT-based water monitoring systems: a systematic review." *Water* 14.22 (2022): 3621.