Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ НА МЕТЕОСТАНЦІЇ»

Виконав: студент IV курсу, групи АКІТР-23мс
спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(шифр і назва спеціальності)
Богдан ПАЛАШ
(прізвище та ініціали)
Керівник: к.т.н., доцент кафедри АІІТ
Володимир ГАРМАШ
«16» 06 2025 р.
Консультант: к.т.н., доцент кафедри АІІТ
Ярослав КУЛИК
«16» 06 2025 р.
Рецензент: к.т.н, проф. каф КСУ
Микола БИКОВ
«17» 06 2025 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«18» 06 2025 р.

Вінниця ВНТУ - 2025

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти перший (бакалаврський)

Галузь знань 17 - Електроніка, автоматизація та електронні комунікації

(шифр і назва)

Спеціальність 174 – автоматизація, комп'ютерно-інтегровані технології та робототехніка

Освітньо-професійна програма Інтелектуальні комп'ютерні системи управління

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ

д.т.н., проф. Олег БІСІКАЛО

«18» 06 2025 року

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Палашу Богдану Руслановичу

(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка програмного забезпечення для автоматимазації процесів на метеостанції»

Керівник роботи: Гармаш В. В. к.т.н, доцент кафедри АІІТ

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «20» березня 2025 р. №97

2. Термін подання студентом роботи «10» червня 2025 р.

3. Вихідні дані до роботи:

Розроблене програмне забезпечення повинне запускатись на будь-якій ОС, де встановлений Arduino IDE. Модель програмного забезпечення об'єктно-орієнтована. Мова програмування С++

4. Зміст текстової частини

Аналіз існуючих систем моніторингу метеоданих та визначення вимог до проектованої метеостанції. Розробка структурної схеми метеорологічної станції та вибір елементної бази. Розробка програмного забезпечення для автоматизації процесів на метеостанції.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)

Схема організації метеостанції на базі Arduino Uno. Схема етапів ініціалізації системи. Схема алгоритму відображення даних на дисплеї. Схема етапів процесу зчитування даних з датчиків.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Кулик Я.А., доц. каф. АІІТ	21.03.25	13.05.25

7. Дата видачі завдання «20» березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітки
		початок	закінчення	
1	Вибір, узгодження та затвердження теми БКР	10.03.2025	20.03.2025	Вик.
2	Аналіз літературних джерел. Попередня розробка основних розділів	21.03.2025	02.04.2025	Вик.
3	Проектування структурної схеми метеостанції, обґрунтування вибору елементної бази	03.04.2025	15.04.2025	Вик.
4	Огляд існуючих програмних засобів для реалізації поставленої задачі	16.04.2025	27.04.2025	Вик.
5	Розробка програмного забезпечення	28.04.2025	11.05.2025	Вик.
6	Опис алгоритму функціонування програми	12.05.2025	25.05.2025	Вик.
7	Оформлення пояснювальної записки та графічної частини	26.05.2025	08.06.2025	Вик.
8	Попередній захист БКР, доопрацювання, рецензування БКР	09.06.2025	16.06.2025	Вик.
9	Захист БКР ЕК	18.06.2025	18.06.2025	Вик.

Студент _____ Богдан ПАЛАШ
(підпис) (прізвище та ініціали)

Керівник роботи _____ Володимир ГАРМАШ
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 79 сторінок формату А4 в тому числі і додатків, на яких є 11 рисунків, 5 таблиць, список використаних джерел містить 31 найменування.

Ця робота присвячена розробці програмного забезпечення для автоматизованої метеостанції на базі Arduino Uno, що забезпечує моніторинг ключових метеорологічних параметрів. Основна мета полягає в розробці ефективного інструменту для збору, обробки та відображення даних про температуру, вологість та атмосферний тиск у реальному часі. У процесі дослідження було визначено ключові вимоги до метеостанції, включаючи високу точність вимірювань, автоматизацію збору даних та зручність експлуатації. Вибір мікроконтролера ATmega328 та сенсора BME280 забезпечив оптимальне поєднання продуктивності та енергоефективності. Програмне забезпечення, розроблене на C++ в Arduino IDE, дозволяє зчитувати, обробляти та відображати дані з датчиків, забезпечуючи надійний моніторинг метеорологічних параметрів.

Ключові слова: автоматизована метеостанція, мікроконтролер, інтернет речей (iot), моніторинг метеоданих, прогнозування погоди, c/c++ програмування, arduino ide, вбудовані системи, безпровідний зв'язок, збір даних, обробка даних, візуалізація даних.

ANOTATION

The bachelor's thesis consists of 79 A4 pages, including appendices with 11 figures and 5 tables. The list of references includes 31 titles.

This work is dedicated to the development of software for an automated weather station based on Arduino Uno, which provides monitoring of key meteorological parameters. The main goal is to develop an effective tool for collecting, processing, and displaying data on temperature, humidity, and atmospheric pressure in real-time. During the research process, key requirements for the weather station were defined, including high measurement accuracy, automated data collection, and ease of operation. The choice of the ATmega328 microcontroller and the BME280 sensor ensured an optimal combination of performance and energy efficiency. The software, developed in C++ within the Arduino IDE, allows for the reading, processing, and displaying of data from sensors, ensuring reliable monitoring of meteorological parameters.

Keywords: automated weather station, microcontroller, internet of things (iot), weather data monitoring, weather forecasting, c/c++ programming, arduino ide, embedded systems, wireless communication, data collection, data processing, data visualization.

ЗМІСТ

ВСТУП

Важливу роль у сферах, де важливим є контроль та аналіз метеопоказників, відіграють станції, які дозволяють одержувати інформацію про температуру навколишнього середовища, вологості, атмосферного тиску, тощо. Ця інформація має вирішальне значення в промисловості для планування виробничих процесів, у сільському господарстві для оптимізації вирощування рослин та тварин, у наукових дослідженнях для розуміння кліматичних змін, а також у повсякденному житті.

Актуальність роботи. Ефективне спостереження за погодними умовами дозволяє отримувати цінну інформацію для прогнозування, прийняття рішень та забезпечення безпеки. Зростаючий попит на оперативні та точні метеорологічні дані зумовлює необхідність використання автоматизованих метеостанцій, здатних збирати, обробляти та передавати інформацію безперервно та з мінімальним втручанням людини. Автоматизація процесів на метеостанції не тільки підвищує ефективність збору даних, але й забезпечує їхню надійність та частоту оновлення.

Метою бакалаврської кваліфікаційної роботи є підвищення ефективності та точності збору метеорологічних даних та прогнозування погоди шляхом впровадження автоматизованого програмного забезпечення для метеорологічних станцій**,** що забезпечить якісний контроль параметрів клімату.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

– визначити вимоги та проаналізувати існуючі рішення щодо проектування метеостанцій;

– розробити структурну схему метеорологічної станції та вибрати елементну базу;

– створити програмне забезпечення для автоматизації процесів на метеостанції.

Об’єктом роботи є процес створення автоматизованої системи моніторингу метеоданих.

Предметом роботи є засоби та методи розробки програмного забезпечення для автоматизації процесів на метеостанції.

Методи дослідження включають аналіз і синтез, порівняння та узагальнення, системний підхід, а також методи проектування та розробки програмного забезпечення, апробація та виступ на конференції.

Апробація результатів роботи здійснювалася шляхом публікації наукової тези "Розробка програмного забезпечення для автоматизації процесів на метеостанції" на LIV Всеукраїнській науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації (2025), у секції автоматизації та інтелектуальних інформаційних технологій, що відбулася у Вінницькому національному технічному університеті 24-25 квітня 2025 року.[1]

Науково-практичний результат роботи полягає в розробці програмного забезпечення для автоматизації процесів на метеостанції, розроблене на C++ в Arduino IDE, що дозволяє зчитувати, обробляти та відображати дані з датчиків, забезпечуючи надійний моніторинг ключових метеорологічних параметрів.

У підсумку, результати даної бакалаврської роботи сприятимуть розвитку вимірювальних приладів та метеорологічних систем, що матиме важливе значення для наукових досліджень, промисловості та повсякденного життя. Отримані результати та розроблені рішення можуть бути використані як основа для подальших розробок та вдосконалення систем вимірювання параметрів навколишнього середовища з використанням новітніх технологій та методів.

1 ВИЗНАЧЕННЯ ВИМОГ ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ЩОДО ПРОЕКТУВАННЯ МЕТЕОСТАНЦІЙ

1.1 Огляд існуючих систем моніторингу метеоданих

Спостереження за атмосферними явищами, їх правильне визначення, фіксація часу початку та припинення, коливань інтенсивності тощо здійснюються на метеорологічних станціях. За технічними можливостями метеостанції можна розділити на дві категорії: аналогові та цифрові.

Аналогові метеостанції обладнані механічними кліматичними пристроями, які спроможні лише реєструвати поточні зміни атмосферних показників. Функціональні можливості даних пристроїв є досить обмеженими. Цифрові метеостанції можуть не лише фіксувати поточні параметри, а й складати прогноз погоди на найближчі дні. [2]

Головними складовими будь-якої метеостанції є датчики. Датчик слугує надійним приймачем та перетворювачем вимірюваних величин, який до того ж володіє високою точністю. На сьогодні існує широкий вибір датчиків, які використовуються у метеостанціях.

В результаті обробки отриманої інформації, вимірюючи і аналізуючі, можливо передбачати погоду. Передбачати погоду на короткий термін можливо зі значно більшою точністю за допомогою портативних метеорологічних станцій, в тому числі і автоматичних.

Метеостанції дозволяють отримувати точні значення температури, вологості та атмосферного тиску. На підставі аналізу динаміки змін показників можна самостійно робити прогноз погоди. Дані спостережень портативних метеорологічних станцій дають уявлення про характеристику погодних умов на певній, зазвичай досить невеликій ділянці місцевості. [3]

До інформації про погодні умови висуваються наступні вимоги:

– оперативність збору інформації, а отже її доцільність;

– мінімізація технічних засобів для її отримання;

– точність інформації, що надходить;

– зручність у використанні та мобільність технічних засобів збору інформації, тощо.

Дані, що надходять зі звичайних метеорологічних станцій, далеко не у всіх випадках можуть бути використані: по-перше, збирання метеорологічних даних є узагальнюючим; по-друге, ці метеостанції є зазвичай стаціонарними, тобто розташовані на певних ділянках місцевості; по-третє, не всі ділянки земної поверхні знаходяться в зонах дії метеорологічних станцій.

Тому для реалізації всіх цих вимог потрібна компактна автоматична система збору метеорологічної інформації, яка надає оперативні та точні дані, необхідні для прийняття виважених рішень та попередження негативних наслідків внаслідок змін погодних умов.

Розглянемо декілька аналогів цифрових метеостанцій, метою яких є збір та збереження певних метеопоказників.

1.1.1 Побутова метеостанція HamaEWS-800

HamaEWS-800 [4] – представляє собою побутову метеостанцію, яка здатна вимірювати значення відносної вологості, температури та рівня атмосферного тиску.

Зовнішній вигляд пристрою представлено на рис. 1.1.

Рисунок 1.1. Зовнішній вигляд HamaEWS-800

Пристрій має барометричний індикатор з атмосферним тиском для точного відстеження змін погоди. Однією з ключових *переваг* є наявність

регульованого температурного діапазону (з заданням максимальних і мінімальних значень), який при активації сигналу тривоги допомагає своєчасно реагувати на зміни та підтримувати комфортні умови.

Метеорологічна станція має монохромний дисплей, що має світлодіодне підсвічування. Живлення здійснюється за рахунок батарейок, які знаходяться у спеціальному відсіку задньої панелі базового блоку. Важливою особливістю метеостанції є також можливість підключення додаткових бездротових сенсорів, які можуть працювати дистанційно. Характеристики метеостанції наведено у таблиці 1.1.

Таблиця 1.1. Характеристики HamaEWS-800

№	Назва характеристики	Значення
1	Діапазон вимірювання температури	–40 °C … +65 °C
2	Діапазон вимірювання вологи	20%…95%
3	Діапазон вимірювання тиску	350…1000 гПа
4	Живлення	від батарейок (3xAA)
5	Додаткові функції	годинник, календар
6	Ціна	2500 грн.

Недоліками метеостанції є недостатня точність вимірювання температури (у якості сенсора температури використовується звичайний кремнієвий діод), низька контрастність зображення внаслідок нерівномірної підсвітки та не дуже зручні одиниці вимірювання тиску (замість міліметрів використовуються дюйми р.с).

1.1.2 Метеостанція La Crosse WT137 Aluminum

Метеостанція La Crosse WT137 Aluminum [5] – представляє собою дешевий та простий варіант, що може отримувати дані про температуру і вологість повітря. Зовнішній вигляд приладу представлено на рис. 1.2.

Метеостанція La Crosse WT137 Aluminum проста у використанні та забезпечує чітке відображення на своєму цифровому монохромному дисплеї. *Додатковою перевагою* є інтуїтивно зрозуміла розфарбована шкала рівня комфорту, яка миттєво дає змогу оцінити сприятливість мікроклімату.

Рисунок 1.2 – Зовнішній вигляд La Crosse WT137 Aluminum

Корисною особливістю є функція фіксації мінімальних і максимальних значень, що дозволяє відстежувати зміни протягом часу. Живлення від однієї батарейки робить її автономною та зручною у розміщенні.

Характеристики приладу наведено у таблиці 1.2.

Таблиця 1.2. Характеристики La Crosse WT137 Aluminum

№	Назва характеристики	Значення
1	Діапазон вимірювання температури	0 °C … +50 °C
2	Діапазон вимірювання вологи	20%…95%
3	Живлення	батарейки (1xAA)
4	Додаткові функції	годинник, будильник
5	Ціна	450 грн.

Недоліками метеорологічної станції є її досить обмежена функціональність: вимірювання АТ не передбачена, відсутня можливість

підключення додаткових сенсорів, неможливість проведення вимірів при низьких температурах.

1.1.3 Цифрова метеостанція TFA Dostmann Spring Breeze 35114001

Цифрова метеостанція TFA Dostmann Spring Breeze 35114001 [6] складається з базової станції, зовнішнього датчика 30.3222.02 з кронштейном та блока живлення (рисунок 1.3).

Рисунок 1.3 – Зовнішній вигляд TFA Dostmann Spring Breeze 35114001

Перевагами такої метеостанцій є точність вимірювань, яка забезпечується завдяки використанню високоякісних сенсорів і каліброваних модулів. Вона зручна у використанні, надійна та довговічна в експлуатації.

Незважаючи на суттєві функціональні переваги в порівнянні з іншими аналогами, дана метеостанції також має *певні недоліки*: висока вартість та обмежені можливості кастомізації: зазвичай комерційні рішення не дозволяють користувачам вносити значні зміни до програмного забезпечення або апаратної частини, що обмежує їхню гнучкість.

Характеристики приладу наведено у таблиці 1.3.

Таблиця 1.3. Характеристики TFA Dostmann Spring Breeze 35114001

№	Назва характеристики	Значення
1	Діапазон вимірювання температури	-40 °C … +60 °C
2	Діапазон вимірювання вологи	1%…99%
3	Частота передачі даних (від зовнішніх датчиків)	433 MHz
4	Живлення базової станції	блок живлення 230 VAC/5.0 VDC; батарейки (2xAAA)
5	Живлення зовнішнього датчика	батарейки (2xC)
6	Додаткові функції	годинник, календар на шести мовах
7	Ціна	7200 грн.

Отже, зібрані дані про метеостанції виявляють низку питань, що потребують уваги для покращення їхньої функціональності та зручності використання. Однією із ключових проблем є недостатня точність вимірювання температури, що в окремих моделях, як-от HamaEWS-800, зумовлена застосуванням простих кремнієвих діодів як сенсорів. Це може призводити до значних похибок у відображенні реальних температурних показників, що є критично важливим для метеорологічних спостережень та прогнозів. Вирішення цієї проблеми полягає у впровадженні більш досконалих та каліброваних температурних датчиків, які забезпечуватимуть вищу точність вимірювань.

Іншою суттєвою проблемою є обмежена функціональність деяких метеостанцій. Наприклад, модель La Crosse WT137 Aluminum не передбачає вимірювання атмосферного тиску, що є важливим параметром для розуміння погодних процесів. Крім того, відсутність можливості підключення додаткових сенсорів обмежує спектр даних, які може збирати станція, роблячи її менш універсальною. Також, нездатність проводити виміри при низьких

температурах робить такі станції непридатними для використання в холодних кліматичних умовах.

Розв'язання цих обмежень вимагає розширення базового набору функцій метеостанцій, включення можливості підключення зовнішніх датчиків для вимірювання різноманітних параметрів (опадів, швидкості та напрямку вітру, вологості ґрунту, сонячної радіації тощо) та забезпечення їхньої надійної роботи в широкому діапазоні температур.

Проблеми також існують у сфері інтерфейсу користувача. Низька контрастність зображення на дисплеї, як у HamaEWS-800, особливо при нерівномірній підсвітці, може ускладнювати зчитування інформації. Незручні одиниці вимірювання, такі як використання дюймів ртутного стовпчика (замість більш звичних міліметрів) для відображення атмосферного тиску, також створюють додаткові незручності для користувачів. Вирішення цих проблем передбачає використання якісних дисплеїв з високою контрастністю та регульованою яскравістю, а також надання користувачам можливості вибору зручних одиниць вимірювання для всіх відображуваних параметрів.

Нарешті, варто відзначити проблему високої вартості та обмежених можливостей кастомізації деяких, особливо комерційних, метеостанцій, як було згадано щодо цифрової метеостанції TFA Dostmann Spring Breeze, незважаючи на її функціональні переваги. Висока ціна робить їх менш доступними для широкого кола користувачів, а обмеження у внесенні змін до програмного забезпечення чи апаратної частини знижує їхню гнучкість та адаптивність до індивідуальних потреб.

Подолання цієї проблеми може полягати у розробці більш економічних рішень, а також у наданні користувачам більшої свободи у налаштуванні та розширенні функціоналу метеостанцій, можливо, через відкриті програмні інтерфейси або модульну конструкцію пристроїв.

На основі наведеного аналізу існуючих метеостанцій, очевидною є нагальна *потреба у власній розробці*, оскільки існуючі рішення мають низку суттєвих недоліків, які можна усунути. Зокрема, власна розробка дозволить

підвищити точність вимірювань, розширити функціональність метеостанції, забезпечити надійну роботу в широкому діапазоні температур та запропонувати більш економічне рішення порівняно з деякими комерційними аналогами. До того ж, метеостанція слугує перехідним етапом для подальших досліджень взаємодії погоди та екосистем, а також для розробки прогнозів щодо впливу погодних змін на здоров'я.

Таким чином, власна розробка метеостанції є необхідним кроком для створення більш досконалого, функціонального, зручного та доступного рішення, яке враховуватиме виявлені недоліки існуючих аналогів та задовольнятиме реальні потреби користувачів.

1.2 Визначення вимог до проектованої системи

Аналіз аналогів цифрових метеорологічних станцій показав, що на сьогоднішній день на ринку наявне досить велике їх різноманіття. Проте, з огляду на виявлені недоліки існуючих метеостанцій, таких як недостатня точність вимірювань, обмежена функціональність, незручний інтерфейс користувача та висока вартість при обмежених можливостях кастомізації, стає очевидною доцільність проектування нової метеостанції та розробки спеціалізованого програмного забезпечення для автоматизації її роботи.

Такий підхід дозволить створити прилад, позбавлений багатьох з перелічених недоліків, забезпечуючи тим самим значно вищу ефективність збору, обробки та аналізу метеорологічних даних. Автоматизація процесів, від збору первинних даних до їхньої візуалізації та передачі, мінімізує людський фактор, підвищить оперативність отримання інформації та спростить експлуатацію приладу.

Крім того, перспективним напрямком у розробці метеостанцій є використання універсальних сенсорів, здатних одночасно вимірювати декілька ключових метеорологічних чинників. Застосування таких інтегрованих рішень може призвести до зменшення загальної кількості сенсорів, спрощення конструкції приладу, зниження його вартості та

енергоспоживання, а також підвищення надійності системи в цілому. Розробка та впровадження подібних універсальних сенсорів є важливим кроком у напрямку створення більш компактних, ефективних та економічно вигідних метеорологічних станцій майбутнього.

Отже, для проектування метеостанції та розробки програмного забезпечення для автоматизації процесів на ній визначено ряд **вимог**:

- висока точність збору та обробки даних, що дозволяє отримувати достовірну та вичерпну інформацію про метеопоказники;
- автоматизація збору даних, що забезпечує їх оперативне опрацювання;
- відображення інформації про вимірювальні параметри на метеоприладі;
- забезпечення можливості підключення до мережі Інтернет (опціонально) для віддаленого доступу до метеоданих;
- використання доступних за вартістю компонентів для забезпечення прийнятної загальної вартості проекту.

Визначення вимог до проектованої системи є важливим етапом, що базується на аналізі існуючих метеостанцій та потребах потенційних користувачів. Враховуючи виявлені недоліки, такі як недостатня точність вимірювань, обмежена функціональність, незручний інтерфейс та висока вартість при обмежених можливостях кастомізації, стає очевидною доцільність проектування нової метеостанції та розробки спеціалізованого програмного забезпечення для автоматизації її роботи.

Такий підхід дозволить створити прилад, позбавлений багатьох з перелічених недоліків, забезпечуючи тим самим значно вищу ефективність збору, обробки та аналізу метеорологічних даних. Автоматизація процесів від збору первинних даних до їхньої візуалізації та передачі мінімізує людський фактор, підвищить оперативність отримання інформації та спростить експлуатацію приладу.

1.3 Висновки до 1 розділу

Створення інженерної системи, призначеної для моніторингу природних параметрів, вимагає попереднього глибокого аналізу існуючих рішень. У першому розділі роботи було проведено систематичний огляд існуючих систем моніторингу метеоданих. Цей етап є фундаментальним для проєктування, оскільки дозволяє ідентифікувати поточні технологічні підходи та виявити їхні функціональні та структурні особливості. Шляхом порівняльного аналізу комерційних та відкритих метеостанцій було оцінено їхню функціональність, діапазони вимірювань, точність, надійність, інтерфейси користувача та цінову категорію.

Порівняння аналогів охоплювало широкий спектр рішень, що представляють різні рівні функціональності та цінові сегменти. Такий підхід дозволив визначити оптимальний рівень складності та точності для проектованої системи, забезпечивши компроміс між вартістю, функціональністю та вимогами до точності вимірювань. Надмірна складність або завищена точність можуть спричинити необґрунтовані ресурсні витрати, тоді як їхня недостатність – незадовільні результати. На цьому етапі було сформовано попередні вимоги до майбутньої метеостанції, враховуючи специфічні потреби та технічні обмеження.

Обґрунтування необхідності створення власної автоматизованої метеостанції базується на виявлених недоліках існуючих комерційних рішень. Досвід експлуатації або аналіз доступних систем часто виявляє компроміси, реалізовані виробниками з метою оптимізації вартості або спрощення конструкції. Це може стосуватися точності сенсорів, ергономіки інтерфейсу, можливостей розширення або загальної економічності.

Власна розробка надає низку суттєвих переваг. По-перше, вона дозволяє підвищити точність вимірювань шляхом інтеграції більш досконалих та індивідуально каліброваних сенсорів. Комерційні рішення часто використовують стандартні компоненти, які можуть не відповідати найвищим вимогам до метрологічних характеристик, тоді як власна розробка забезпечує вибір оптимальних датчиків та їхню прецизійну калібровку.

По-друге, власна розробка дає можливість створити інтуїтивно зрозумілий інтерфейс користувача з підвищеною контрастністю дисплея, що значно підвищує зручність експлуатації. Стандартні інтерфейси комерційних пристроїв не завжди є повністю адаптованими до конкретних потреб користувача.

По-третє, власна розробка може призвести до створення більш економічного рішення порівняно з деякими комерційними аналогами. Це досягається за рахунок прямого закупу компонентів та відсутності націнок за бренд або надлишковий функціонал.

Крім того, ключовою перевагою є підвищена гнучкість у налаштуванні та розширенні функціоналу системи. Комерційні пристрої, як правило, мають фіксовані функціональні можливості, тоді як власна станція може бути адаптована під будь-які індивідуальні потреби, від інтеграції додаткових сенсорів до взаємодії з іншими інформаційними системами.

Нарешті, автоматизація процесів, від збору первинних даних до їхньої візуалізації та передачі, мінімізує вплив людського фактора. Це усуває необхідність у ручному зборі та обробці даних, що підвищує оперативність отримання інформації та спрощує експлуатацію приладу.

Чітке визначення вимог на цьому етапі є критично важливим для успішного проектування та розробки автоматизованої метеостанції. Без ясно сформульованих вимог проєкт може зіткнутися з проблемами на етапах реалізації, що призведе до затримок, додаткових витрат та незадовільного кінцевого результату. Встановлені вимоги враховують як бажану функціональність (параметри вимірювань, точність, методи відображення даних), так і практичні обмеження.

До цих обмежень відносяться: використання платформи Arduino, що обумовлено її простотою програмування, доступністю та широкою базою ресурсів; вартість, оскільки бюджет проєкту є визначальним фактором для оптимізації витрат на компоненти; а також ергономіка експлуатації, щоб

система була простою у використанні, не вимагаючи від кінцевого користувача глибоких технічних знань.

Власне проектування дозволяє інтегрувати найбільш підходящі технічні рішення та програмні алгоритми, враховуючи специфічні потреби та завдання користувачів метеорологічної інформації. Це означає, що система може бути максимально адаптована під конкретні умови використання, чи то локальний моніторинг, чи для агротехнічних потреб, чи для освітніх цілей.

Таким чином, розробка власної автоматизованої метеостанції є обґрунтованим та актуальним кроком. Вона дозволяє створити більш ефективний, функціональний та доступний інструмент для моніторингу атмосферних параметрів. Цей підхід забезпечує не лише високу точність та зручність, а й надає можливість адаптувати рішення до індивідуальних вимог, що часто є недосяжним для готових комерційних продуктів. Такий проєкт сприяє поглибленню знань у галузі електроніки, програмування та системної інтеграції.

2 РОЗРОБКА СТРУКТУРНОЇ СХЕМИ МЕТЕОРОЛОГІЧНОЇ СТАНЦІЇ ТА ВИБІР ЕЛЕМЕНТНОЇ БАЗИ

2.1 Проектування структури метеостанції

Архітектура розробленої автоматизованої метеостанції ґрунтується на взаємодії кількох ключових *функціональних блоків* (рисунок 2.1). Ці блоки спільно забезпечують повний цикл операцій. До цього циклу входять збір, обробка, візуалізація та передача метеорологічних даних.

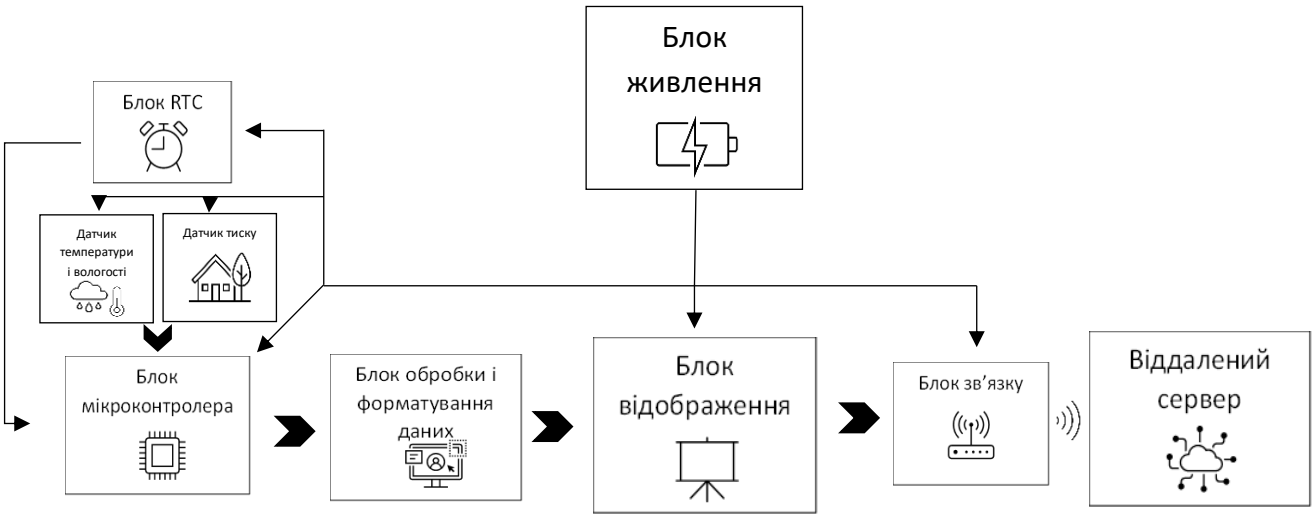


Рисунок 2.1 – Схема метеостанції

Серцем системи є центральний *обчислювальний блок*. Він виконує роль головного координатора та керуючого всіма процесами. Цей модуль відповідає за зчитування інформації від підключених датчиків. Він також проводить необхідні розрахунки для обробки первинних даних. Крім того, блок керує відображенням актуальної інформації на локальному дисплеї. За наявності відповідного модуля, він організовує обмін даними із зовнішніми пристроями через мережу Інтернет.

Для ефективної роботи цього блоку доцільно використовувати популярну платформу мікроконтролера. Вона має бути добре підтримуваною та відзначатися простотою програмування. Важливою є наявність великої кількості готових бібліотек. Також перевагою є активна спільнота розробників. Обраний мікроконтролер повинен мати достатню кількість

портів вводу/виводу. Це необхідно для підключення всіх потрібних сенсорів та периферійних пристроїв. Ще однією важливою характеристикою є адекватна обчислювальна потужність. Вона забезпечить швидку та ефективну обробку отриманих даних. [7]

Інформація про погодні умови надходить до центрального блока від *комплексу сенсорів*. На початковому етапі планується використання комбінованого датчика. Він здатен одночасно вимірювати температуру та відносну вологість повітря. Також передбачено окремий високоточний датчик для визначення атмосферного тиску.

Для забезпечення безпосереднього доступу до поточної метеорологічної інформації передбачено *блок візуалізації*. Він відповідає за відображення даних у режимі реального часу на локальному дисплеї. Простий та рідко-кристалічний дисплей є оптимальним рішенням. Такі дисплеї забезпечують хорошу читабельність тексту та символів за умови достатнього зовнішнього освітлення. Він забезпечить наочне представлення основних погодних параметрів.

Можливість віддаленого доступу до зібраних даних забезпечується *опціональним блоком комунікації*. Він також забезпечує інтеграцію з онлайн-сервісами. За допомогою компактного модуля бездротового зв'язку метеостанція зможе підключатися до Інтернету. Це дозволить користувачам відстежувати погодні умови з будь-якої точки світу. Також з'явиться можливість використовувати різноманітні онлайн-інструменти. Ці інструменти призначені для аналізу та візуалізації даних.

Стабільне та надійне живлення всіх компонентів метеостанції забезпечує окремий *блок живлення*. Він може використовувати зовнішнє джерело живлення. Також можливе використання акумулятора для автономної роботи. Важливою складовою цього блоку є схема стабілізації напруги. Вона гарантує належний рівень живлення для кожного електронного компонента.

Для забезпечення точності фіксації часу вимірювань передбачено використання *блоку годинника реального часу* (RTC). Це потрібно незалежно від підключення до Інтернету чи стабільності основного живлення. Застосування енергоефективної мікросхеми з вбудованим кварцовим резонатором є важливим. Також необхідне резервне живлення від батарейки. Це гарантує збереження точного часу навіть у випадку відключення основного живлення. Це критично важливо для подальшого аналізу та коректної інтерпретації зібраних метеорологічних даних. Взаємодія всіх цих модулів забезпечує повноцінне функціонування автоматизованої метеостанції. Вона буде здатна ефективно збирати, обробляти, відображати та передавати важливу інформацію про погодні умови.

Принцип роботи автоматизованої метеостанції починається зі збору метеорологічних даних різними датчиками, які безперервно фіксують зміни температури, вологості, тиску та інших параметрів. Зібрана інформація передається до центрального блоку, яким є мікроконтролер. Отримавши первинні дані, мікроконтролер виконує їхню обробку, здійснюючи необхідні обчислення та приводячи їх до зручного формату для подальшого використання. Оброблені дані потім спрямовуються до блоку відображення, де вони візуалізуються на локальному дисплеї, надаючи користувачеві оперативну інформацію про поточні погодні умови.

Крім локального відображення, передбачена можливість передачі оброблених даних на зовнішні ресурси. За допомогою блоку зв'язку, який може використовувати, наприклад, технологію Wi-Fi, дані можуть надходити на віддалений сервер або спеціалізовану онлайн-платформу для централізованого зберігання, подальшого аналізу або інтеграції з іншими сервісами.

Стабільну та надійну роботу всіх електронних компонентів метеостанції забезпечує блок живлення, який підтримує необхідний рівень енергії для кожного модуля системи. Важливу роль у функціонуванні станції відіграє блок годинника реального часу (RTC). Він надає мікроконтролеру точну

інформацію про поточний час, що є критично важливим для коректної реєстрації зібраних даних та своєчасного виконання запланованих завдань.

Таким чином, розроблена структурна схема метеостанції чітко визначає взаємодію між усіма ключовими апаратними компонентами, починаючи від збору даних датчиками і закінчуючи їхнім відображенням та передачею. Ця детальна схема є фундаментальною основою для наступного етапу розробки програмного забезпечення, яке керуватиме всіма процесами станції та забезпечить ефективну обробку отриманої метеорологічної інформації.

2.2 Обґрунтування вибору технічних засобів автоматизації

2.2.1 Вибір мікроконтролера

Для проектування автоматизованої метеостанції було розглянуто два основних варіанти мікроконтролерів: Arduino Uno R3 та ESP32 (рисунок 2.2).

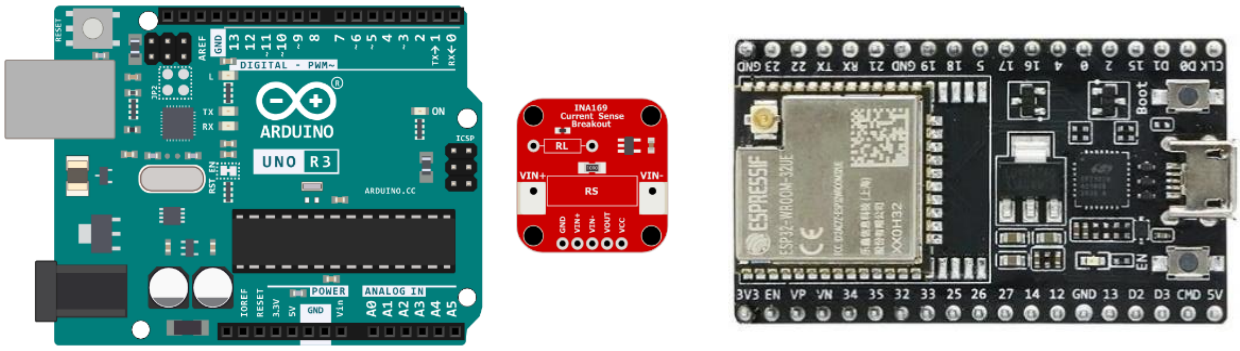


Рисунок 2.2 – Мікроконтролери Arduino Uno R3 та ESP32

Arduino Uno R3 має велику та активну спільноту користувачів по всьому світу, безліч підручників, прикладів коду та бібліотек, які значно спрощують процес розробки. Це означає, що можна завжди знайти відповіді на свої запитання, саме завдяки цьому поріг входження в світ електроніки стає набагато нижчим.

Однією з ключових переваг Arduino Uno R3 є його 5V логічний рівень. Це робить його сумісним з величезною кількістю поширених датчиків без необхідності використовувати додаткові перетворювачі рівнів напруги. Багато електронних компонентів працюють саме з 5V, що суттєво спрощує

підключення та зменшує кількість потенційних джерел помилок. Розробнику не потрібно турбуватися про складні схеми перетворення напруги, що дозволяє зосередитися на логіці проєкту.

Хоча ESP32 безперечно пропонує більше обчислювальної потужності та вбудовані можливості Wi-Fi та Bluetooth, для таких базових проєктів, як, наприклад, проста метеостанція з локальним відображенням даних, Arduino Uno R3 є цілком достатнім і значно простішим у використанні. Використання ESP32 для такого завдання схоже на використання суперкомп'ютера для вирішення простого арифметичного прикладу. Це призводить до зайвих витрат ресурсів (як фінансових, так і часових) та потенційного ускладнення коду без відчутних переваг.

Порівняльна характеристика мікроконтролерів Arduino Uno R3 та ESP32 наведена в таблиці 2.1.

Таблиця 2.1. Порівняння Arduino Uno R3 та ESP32 [8]

Характеристика	**Arduino Uno R3**	**ESP32**
Процесор	8-бітний ATmega328P	Tensilica Xtensa LX6 Dual-Core
ОЗП	2 КБ	520 КБ
Флеш-пам'ять	32 КБ	4 МБ або більше
Тактова частота	16 МГц	До 240 МГц
Цифрові виводи/вводи	14	До 36
Аналогові входи	6	До 18
Wi-Fi	Зовнішній модуль	Вбудований 2.4 ГГц
Робоча напруга	5 V	3.3 V
Вартість	~5 $	~5-10 $
Використання	Дуже просто	Просто

Для проєктів, де простота використання та велика кількість доступних ресурсів є пріоритетом, Arduino Uno R3 є ідеальним вибором. Ця плата стала своєрідним стандартом для входу в електроніку та програмування завдяки своїй доступності та величезній підтримці спільноти.

Arduino Uno R3 розроблений так, щоб бути максимально інтуїтивно зрозумілим. Його інтегроване середовище розробки (IDE) та спрощена мова програмування (заснована на C++) дозволяють швидко написати та завантажити код. ESP32, хоч і потужний, вимагає глибшого розуміння архітектури та налаштувань, що може бути надмірним.

У разі потреби підключення до Інтернету, до Arduino Uno R3 можна легко підключити зовнішній модуль Wi-Fi, такий як ESP8266. Це дозволяє додати функціонал, схожий на ESP32, але без складнощів, пов'язаних з інтеграцією всієї потужності ESP32 з самого початку. Такий підхід дає можливість поступово розширювати функціонал проєкту, додаючи лише ті компоненти, які дійсно потрібні, і тоді, коли саме потрібні. Це дозволяє краще контролювати складність та вартість проєкту.

Питання енергоспоживання є ще одним ключовим аспектом, особливо для проєктів, які працюватимуть від батарей. Хоча ESP32 має режими глибокого сну для економії енергії, активне використання Wi-Fi та його потужний процесор можуть призвести до значно більшого енергоспоживання порівняно з простим 8-бітним мікроконтролером ATmega328P, який є серцем Arduino Uno R3. Для базового проєкту, де енергоефективність не є критичною (наприклад, при живленні від мережі), простота Arduino Uno R3 переважає.

Отже, для проєктів, де простота та велика кількість доступних ресурсів є пріоритетом, Arduino Uno R3 є беззаперечним лідером. Його простота використання, велика спільнота, сумісність з поширеними датчиками та гнучкість у розширенні функціоналу роблять його ідеальним інструментом для швидкої та ефективної реалізації ідей. Хоча ESP32 пропонує більше потужності, його використання для простих завдань є надмірним і може ускладнити процес розробки. Таким чином, роблячи вибір для свого проєкту,

не варто гнатися за надмірною потужністю. Часто оптимальним рішенням є те, що найбільш просте, доступне та має найкращу підтримку. І в цьому контексті Arduino Uno R3 займає провідну позицію.

2.2.2 Вибір датчиків температури, вологості та атмосферного тиску

Параметри ТП, ВП та АТ є основними характеристиками клімату. Ці параметри можуть бути легко виміряні і контрольовані за допомогою датчиків (сенсорів). Ринок датчиків дуже стрімко розвивається, виробники сумлінно працюють над створенням нових та вдосконаленням вже існуючих моделей. Важливою особливістю даних пристроїв є їхня стабільність та працездатність, вони не потребують сну і відпочинку, тому постійно придатні до роботи.

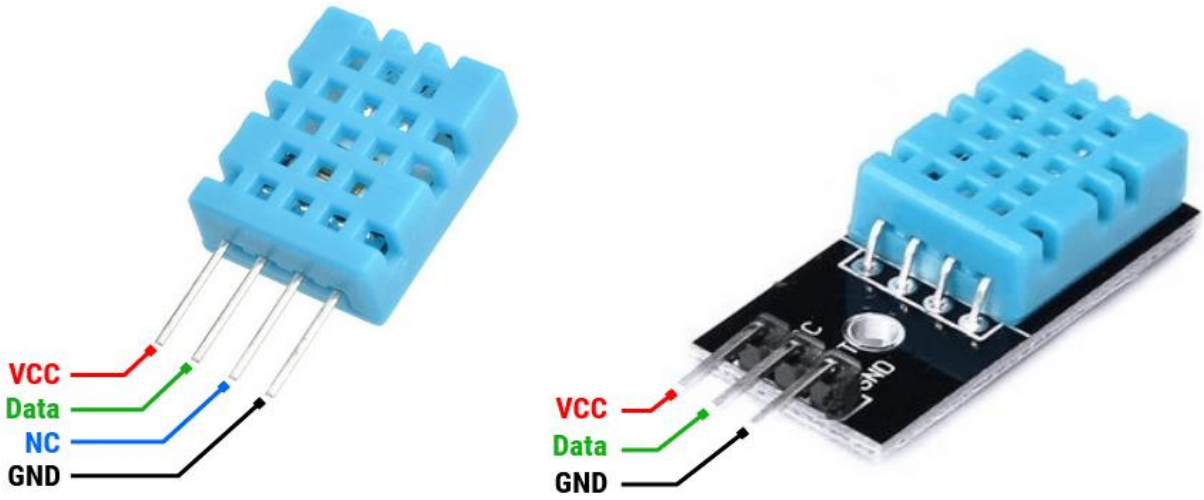


Рисунок 2.3 – Датчик DHT11

Датчик DHT11(див. рисунок 2.3) є бюджетним рішенням для вимірювання температури (0-50°C, ±2°C) та вологості (20-80% RH, ±5%). Він поєднує термістор і ємнісний датчик, має вбудований АЦП для перетворення аналогових сигналів у цифрові. Для зв'язку використовує однодротовий протокол (потрібен підтягувальний резистор). Живиться від 3,3В або 5В, споживає 2,5 мА під час роботи та 50 мкА в стані спокою.

Серед недоліків: обмежений діапазон вимірювань, висока похибка, низька роздільна здатність (1°C/1%) та низька швидкість оновлення (1 Гц). Підключення просте (три виводи), але вимагає використання спеціальних

бібліотек для зчитування даних. Однодротовий протокол менш ефективний порівняно з I²C.

Завдяки низькій ціні та простоті інтеграції, DHT11 підходить для базових застосувань з невисокими вимогами до точності, таких як освітні проекти та моніторинг у стабільних умовах. Не рекомендується для проектів, що потребують високої точності в широкому діапазоні умов.

Датчик DHT22 (рисунок 2.4), також відомий як AM2302, є покращеною версією DHT11, зовні дуже схожою, але значно перевершує її за ключовими характеристиками, незважаючи на дещо вищу ціну. Він так само використовує ємнісний сенсор вологості та термістор для вимірювання температури та вологості.

Рисунок 2.4 – Датчик DHT22

Однією з головних переваг DHT22 є значно ширший діапазон вимірювань температури від –40°C до +80°C та вологості від 0 до 100% RH. Його точність вимірювання температури становить ±0,5°C, а вологості – ±2% (іноді до ±5%). Роздільна здатність температури становить 0,1°C. Проте, частота оновлення даних у DHT22 нижча, лише 0,5 Гц. Живлення та споживання струму (3,3-5В, до 2,5 мА) аналогічні DHT11, як і однодротовий інтерфейс з підтягувальним резистором та трьома виводами для підключення.

Програмування та спосіб передачі даних також ідентичні DHT11. Основними недоліками DHT22 є нижча швидкість оновлення, можлива інерційність, потенційні розбіжності в показаннях між екземплярами та іноді некоректні зчитування. Завдяки вищій точності та ширшому діапазону, DHT22 є кращим вибором для метеорологічних станцій та інших проектів, де важлива точність вимірювань, включаючи зовнішній моніторинг.

Датчик BME280 (рисунок 2.5) являє собою високоточний сучасний сенсор, який, на відміну від DHT11 та DHT22, здатний вимірювати не лише температуру та вологість, але й атмосферний тиск. Для вимірювання температури та вологості він використовує традиційні термістор та ємнісний датчик, а для визначення тиску – п'єзорезистивний датчик з деформуючоюся мембраною. Компактний корпус LGA робить його ідеальним вибором для використання в мобільних та портативних пристроях, де важливі малі розміри та низьке енергоспоживання.

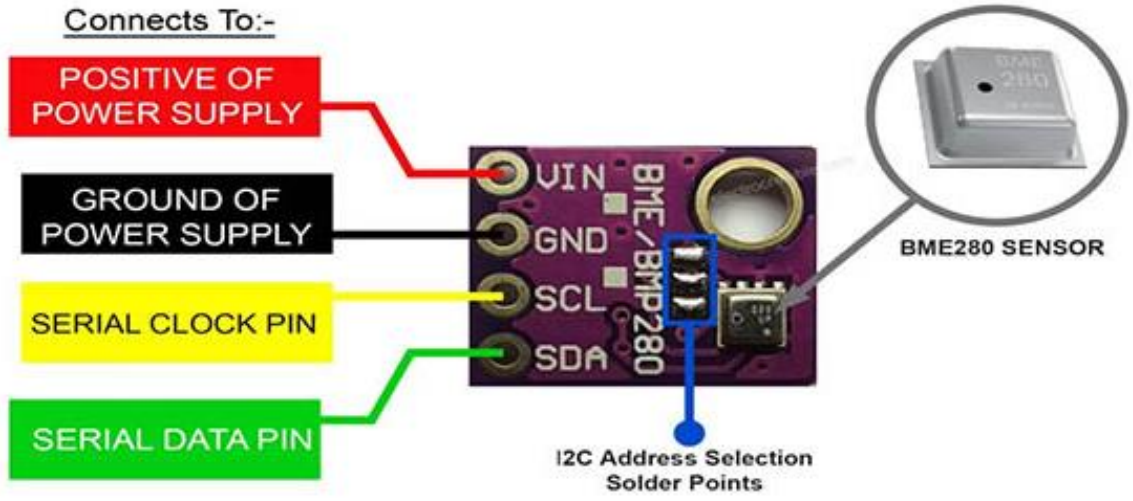


Рисунок 2.5 – Датчик BME280

Датчик BME280 вирізняється широким діапазоном вимірювання температури від –40°C до +85°C та повним діапазоном вологості від 0% до

100% RH. Його високу точність підтверджує типова похибка: ±1,0°C для температури та ±3% для вологості.

Однією з ключових переваг BME280 є його надзвичайно низьке енергоспоживання. В активному режимі вимірювання він споживає лише близько 3,6 мкА, а в режимі сну цей показник падає до мізерних 0,1 мкА. Разом із широким діапазоном напруги живлення від 1,8 В до 3,6 В (який у готових модулях часто розширюється до 5 В), це робить BME280 чудовим вибором для автономних пристроїв, що працюють від батарей і потребують тривалого часу роботи.

Для обміну даними BME280 використовує стандартні інтерфейси I²C або SPI, що є значно ефективнішим за однодротові протоколи DHT-датчиків. Підключення є зручним: для I²C потрібно чотири дроти, а для SPI – шість. Для програмування існують численні готові бібліотеки, наприклад, Adafruit BME280 для Arduino, які спрощують процес зчитування даних про температуру, тиск та вологість, а також забезпечують гнучкість при підключенні кількох датчиків через шину I²C.

Загалом, BME280 виступає як найкращий вибір серед розглянутих датчиків, пропонуючи розширені можливості та високу продуктивність. Його ключові переваги включають здатність вимірювати три ключові параметри: температуру, вологість і атмосферний тиск. Це робить його універсальним рішенням для комплексних метеостанцій та систем моніторингу. Крім того, BME280 вирізняється високою точністю вимірювань, швидкою реакцією на зміни у навколишньому середовищі та дуже низьким енергоспоживанням. Останнє є критично важливим для автономних пристроїв, які працюють від батарей і потребують тривалого часу роботи без підзарядки.

Однак, як і будь-який пристрій, BME280 має свої недоліки. Основним з них є вища вартість порівняно з простішими DHT-датчиками, що може бути фактором при обмеженому бюджеті проєкту. Ще одним потенційним недоліком є можливе самонагрівання датчика. Це явище може призводити до незначного завищення показань температури, що варто враховувати при

проектуванні та калібруванні системи. Незважаючи на ці незначні мінуси, переваги BME280 роблять його ідеальним вибором для професійних та вимогливих застосувань.

Порівняння трьох вищевказаних датчиків наведені в таблиці 2.2.

Таблиця 2.2- Порівняльна характеристика датчиків [9].

Характеристика	DHT11	DHT22 (AM2302)	BME280
Вимірювані величини	Температура, вологість	Температура, вологість	Температура, вологість, тиск
Температурний діапазон	Від 0 до 50 °C	Від -40 до 80 °C	Від -40 до 85 °C
Похибка температури	±2 °C (0…50 °C)	±0,5 °C (у всьому діапазоні)	±1,0 °C (типова)
Вологісний діапазон	~20…80 % RH	0…100 % RH	0…100 % RH
Похибка вологості	±5 % RH	±2 % RH (типова), до ±5 %	±3 % RH
Роздільна здатність	~1 °C; ~1 % RH	~0,1 °C; ~0,1 % RH	~0,01 °C; ~0,008 % RH (16-bit ADC)
Частота оновлення	≤1 Hz (1 вимір/с)	≤0,5 Hz (1 вимір/2 с)	До ~100 Hz
Інтерфейс зв’язку	Однодротовий (протокол DHT)	Однодротовий (протокол DHT)	I²C або SPI (цифрова шина)
Живлення	3,0–5,5 В	3,0–6,0 В	1,71–3,6 В (чип); 3,3–5 В (модуль)
Споживання струму	~50 μA (очікування); до 2,5 мА при вимірі	~50 μA (очікування); до 2,5 мА при вимірі	~3,6 μA(при 1 Hz); 0,1 μA (сон)

Для вимірювання температури та вологості *оптимальним вибором* є датчик DHT22 завдяки його вищій точності та ширшому діапазону вимірювань порівняно з DHT11. Для додаткового вимірювання атмосферного тиску, необхідного для базового прогнозування погоди, може бути використаний датчик BME280. Слід зазначити, що, згідно з дослідженнями,

датчик BME280 може показувати температуру на 1-2°C вище за фактичну через самонагрів або особливості калібрування. Тому для найбільш точних вимірювань температури та вологості рекомендується використовувати DHT22, а BME280 – для вимірювання атмосферного тиску. Обидва датчики сумісні з Arduino Uno R3.

2.2.3 Вибір дисплея

Розглядаючи варіанти відображення метеорологічних даних на локальному рівні, ми детально проаналізували переваги та недоліки двох основних типів дисплеїв: рідкокристалічних (РК) та органічних світлодіодних (OLED). Кожен з цих типів має свої унікальні характеристики, які впливають на зручність використання, вартість та складність інтеграції в наш проект автоматизованої метеостанції.

Рідкокристалічні (РК) дисплеї є добре відомою та широко використовуваною технологією. Особливу увагу ми звернули на алфавітно-цифрові РК-дисплеї, зокрема модель з конфігурацією 16x2 (рисунок 2.6). Однією з ключових переваг цього типу дисплеїв є їхня економічна вигідність. У порівнянні з OLED-дисплеями, РК-екрани мають значно нижчу вартість, що є важливим фактором при розробці бюджетного, але функціонального рішення. Крім того, РК-дисплеї є широко доступними на ринку електроніки, що полегшує їхнє придбання та заміну у разі потреби. [10]

Рисунок 2.6 – РК-дисплей LCD1602A рідкокристалічний

Важливим аспектом для нашого проекту, який використовує платформу Arduino, є наявність великої кількості готових бібліотек, таких як добре відома

LiquidCrystal. Ці бібліотеки значно спрощують процес програмування та керування дисплеєм, дозволяючи розробникам швидко інтегрувати його в систему без необхідності глибокого розуміння низькорівневих протоколів. Простота підключення РК-дисплеїв до мікроконтролера Arduino також є значною перевагою. Зазвичай, для їхнього підключення потрібно лише кілька з'єднувальних проводів.

Щодо якості відображення, алфавітно-цифрові РК-дисплеї забезпечують відмінну читабельність тексту та символів, особливо за умови достатнього зовнішнього освітлення. Ці дисплеї ідеально підходять для завдань, що вимагають чіткого представлення текстової інформації.

Для нашого проєкту, який полягає у відображенні основних метеорологічних параметрів у текстовому форматі, таких як "Температура: 25°C" та "Вологість: 60%", можливостей РК-дисплея 16x2 (16 символів у 2 рядках) буде цілком достатньо. Цей тип дисплея дозволяє ефективно виводити необхідні дані, забезпечуючи при цьому простоту використання та підключення.

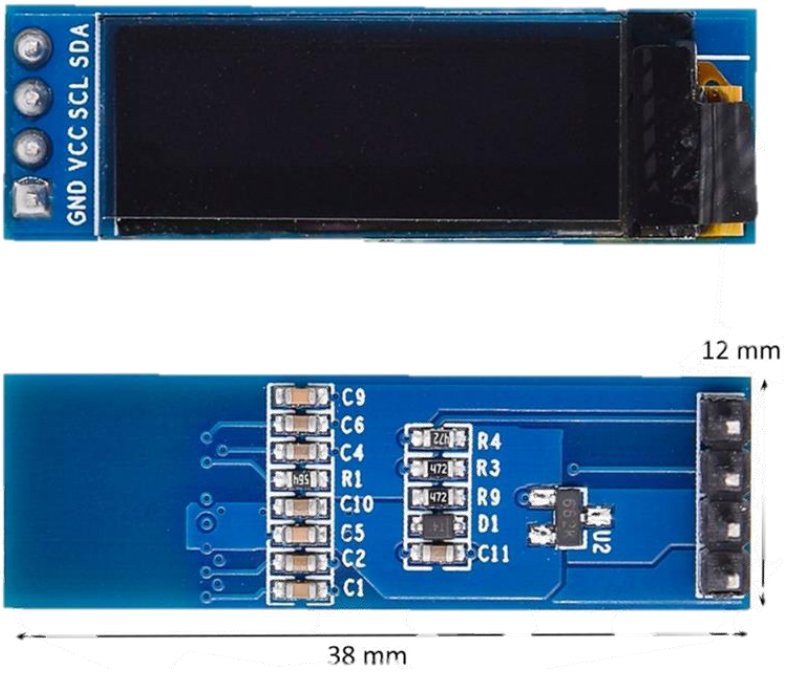


Рисунок 2.7 – OLED LCD РК-дисплей

З іншого боку, *OLED (органічні світлодіодні) дисплеї*, такі як моделі з роздільною здатністю 128x64, пропонують значно вищу контрастність

зображення, що значно покращує візуальне сприйняття інформації (див. рисунок 2.7). OLED-дисплеї також мають кращі кути огляду порівняно з багатьма типами РК-екранів, що дозволяє користувачеві чітко бачити інформацію незалежно від кута, під яким він дивиться на дисплей. [11]

Однак, ці переваги OLED-дисплеїв супроводжуються певними недоліками. Перш за все, вони можуть бути дорожчими за РК-дисплеї, що може збільшити загальну вартість проекту. Крім того, для їхнього використання часто вимагається встановлення додаткових бібліотек для Arduino, які можуть бути менш поширеними та мати складнішу структуру порівняно з бібліотеками для РК-дисплеїв.

Обґрунтовуючи вибір на користь РК-дисплея 16x2, керуємось кількома ключовими міркуваннями. Низька вартість робить його оптимальним варіантом з точки зору бюджету проекту. Простота використання з платформою Arduino Uno R3, а також наявність великої кількості навчальних матеріалів та бібліотек, значно полегшують процес розробки та інтеграції дисплея в проектовану метеостанцію. Враховуючи, що основною метою локального відображення є надання користувачеві базової інформації про поточні погодні умови у текстовому форматі, РК-дисплей 16x2 чудово справляється з цим завданням, забезпечуючи достатню читабельність та надійність. Хоча OLED-дисплеї мають певні технічні переваги, їхня вища вартість та потенційна складність інтеграції роблять РК-дисплей більш практичним та економічно ефективним вибором для даного проекту.

2.2.4 Wi-Fi модуль ESP8266 ESP-01

У нашому проекті автоматизованої метеостанції передбачена можливість підключення до мережі Інтернет для розширення функціональності системи, зокрема для віддаленого доступу до зібраних даних та їхньої інтеграції з різноманітними онлайн-платформами або сервісами. [12]

Для реалізації цієї можливості з використанням мікроконтролера Arduino Uno R3 оптимальним рішенням є застосування зовнішнього модуля *Wi-Fi ESP8266* (рисунок 2.8). Цей модуль є досить популярним вибором серед розробників завдяки своїй невисокій вартості, що робить його економічно привабливим для нашого проекту.

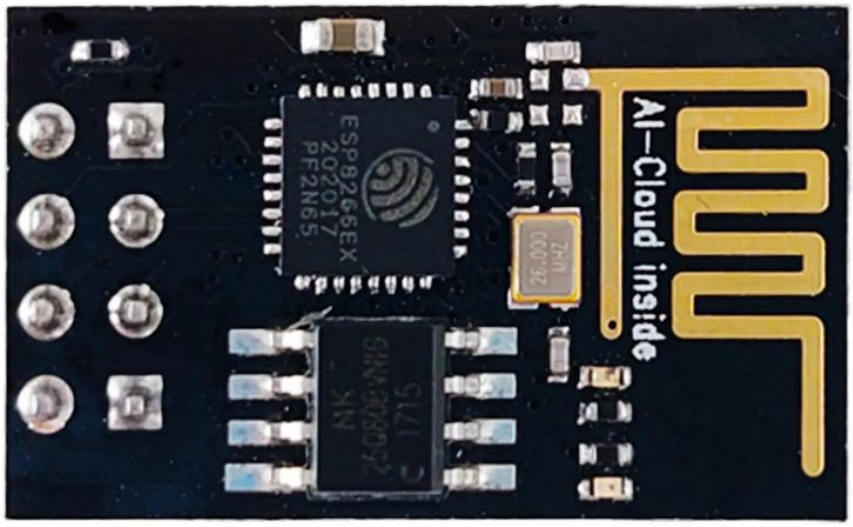


Рисунок 2.8 – Wi-Fi модуль ESP8266 ESP-01

Не менш важливою перевагою модуля ESP8266 є його активна підтримка спільнотою Arduino. Для цього модуля існує велика кількість добре розроблених та протестованих бібліотек Arduino, які значно спрощують процес встановлення з'єднання з мережею Wi-Fi та передачі даних. Завдяки цим бібліотекам, розробники можуть відносно легко реалізувати необхідний функціонал без глибокого занурення в низькорівневі деталі роботи бездротового зв'язку.

Використання модуля ESP8266 надає нашій метеостанції можливість підключатися до існуючих мереж Wi-Fi. Після встановлення з'єднання, станція отримує доступ до Інтернету, що відкриває широкі можливості для передачі зібраних метеорологічних даних на віддалені сервери для централізованого зберігання та подальшого аналізу. Крім того, з'являється можливість інтеграції з різноманітними онлайн-платформами, які можуть надавати послуги візуалізації даних, прогнозування погоди або інші корисні функції.

Таким чином, застосування зовнішнього Wi-Fi модуля ESP8266 є *ефективним та економічно обґрунтованим* способом забезпечення підключення нашої автоматизованої метеостанції до мережі Інтернет, відкриваючи шлях для розширення її функціональних можливостей та інтеграції з глобальною мережею.

2.2.5 Модуль годинника реального часу DS3231

Використання модуля годинника реального часу (RTC), *такого як DS3231* (рисунок 2.9), є надзвичайно важливим доповненням до нашої автоматизованої метеостанції, особливо з огляду на необхідність точної фіксації часу кожного вимірювання. Цей модуль забезпечує абсолютно точний відлік часу, який не залежить від наявності зовнішнього живлення або підключення до мережі Інтернет. Ця незалежність є критично важливою для метеостанції, оскільки коректна часова мітка кожного зафіксованого параметра є основою для подальшого аналізу погодних тенденцій, побудови графіків та будь-якої іншої обробки даних. [13]

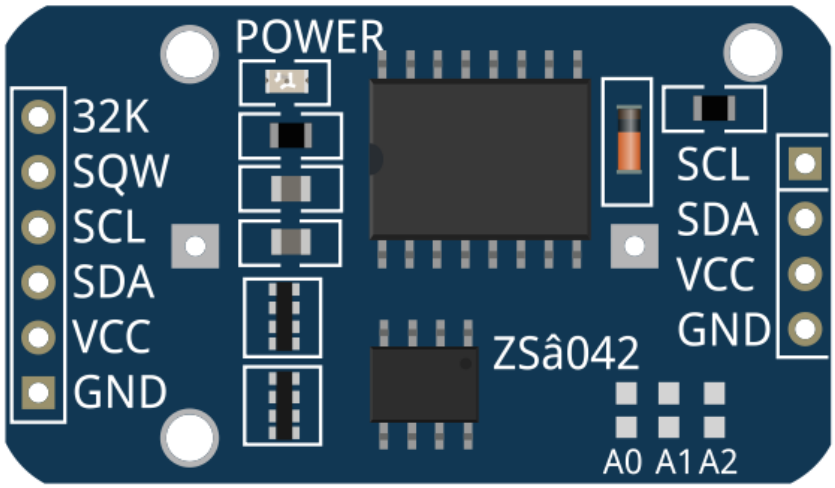


Рисунок 2.9 – Модуль годинника реального часу DS3231

Модуль DS3231 вирізняється своєю високою точністю завдяки вбудованому температурно-компенсованому кварцовому генератору (TCXO). Це забезпечує мінімальну похибку у відліку часу навіть при змінах температури навколишнього середовища, що є характерним для зовнішніх

умов експлуатації метеостанції. Крім того, модуль DS3231 є недорогим компонентом, що відповідає нашому прагненню до економічно ефективного рішення.

Ще однією значною перевагою використання DS3231 є хороша підтримка бібліотеками Arduino. Існує велика кількість готових до використання бібліотек, які значно спрощують процес інтеграції модуля з мікроконтролером Arduino Uno R3 та забезпечують зручні функції для зчитування та встановлення поточного часу й дати. Це дозволяє розробникам легко реалізувати функціонал точної часової реєстрації даних без необхідності глибокого розуміння принципів роботи RTC.

Таким чином, інтеграція модуля годинника реального часу DS3231 є обґрунтованим та важливим кроком у розробці проектованої метеостанції. Він гарантує точність часових міток для всіх зібраних метеорологічних даних, забезпечує незалежність від зовнішніх факторів живлення та підключення до мережі, є економічно вигідним і легко інтегрується з платформою Arduino завдяки наявності якісних бібліотек.

2.2.6 Підключення компонентів метеостанції

Для підключення датчика температури та вологості DHT22 необхідно з'єднати його вивід VCC з 5V Arduino, оскільки цей датчик зазвичай працює від 3.3V до 5V, і таке підключення забезпечує достатнє живлення. Вивід ДАНІ слід підключити до виводу 2 Arduino, тому що DHT22 використовує однопровідний послідовний протокол зв'язку, якому потрібен лише один сигнальний провід, а вибір саме цього виводу є довільним. Вивід GND датчика потрібно з'єднати із GND Arduino, що є обов'язковим для забезпечення однакового рівня опорної напруги.

Підключаючи датчик температури, вологості та тиску BME280, який використовує протокол I2C, його вивід VCC слід з'єднати з 3.3V Arduino, оскільки робоча напруга цього датчика становить 3.3V, і підключення до 5V може призвести до його пошкодження. Вивід GND BME280 необхідно

підключити до GND Arduino. Вивід SCL (Serial Clock Line) потрібно з'єднати з A5 Arduino, оскільки це одна з двох ліній протоколу I2C, що відповідає за синхронізацію передачі даних, а на платах Arduino Uno виводи A4 та A5 за замовчуванням використовуються для I2C. Вивід SDA (Serial Data Line) BME280 слід підключити до A4 Arduino, оскільки ця лінія використовується для безпосередньої передачі даних.

Для підключення РК-дисплея 16x2 з адаптером I2C його вивід VCC необхідно з'єднати з 5V Arduino, оскільки адаптери часто мають вбудований регулятор напруги. Вивід GND РК-дисплея слід підключити до GND Arduino. Вивід SCL адаптера потрібно з'єднати з A5 Arduino, оскільки він використовує протокол I2C. Вивід SDA адаптера необхідно підключити до A4 Arduino для передачі даних.

Підключаючи модуль годинника реального часу DS3231, який також використовує протокол I2C, його вивід VCC слід з'єднати з 5V Arduino, оскільки цей модуль може працювати від 3.3V до 5V. Вивід GND DS3231 потрібно підключити до GND Arduino. Вивід SCL модуля необхідно з'єднати з A5 Arduino. Вивід SDA модуля слід підключити до A4 Arduino для передачі даних часу.

Для підключення Wi-Fi модуля ESP8266 його вивід VCC необхідно з'єднати з 3.3V Arduino, оскільки цей модуль чутливий до напруги і працює лише при 3.3V. Вивід GND ESP8266 слід підключити до GND Arduino. Вивід RX (Receive) модуля потрібно підключити до піна 10 Arduino через дільник напруги, оскільки RX вивід ESP8266 розрахований на 3.3V, а TX вивід Arduino працює на 5V. Вивід TX (Transmit) ESP8266 слід підключити до виводу 11 Arduino, оскільки RX вивід Arduino є толерантним до 5V, хоча для надійності іноді рекомендується використовувати зсув рівнів напруги.

Слід зазначити, що BME280, РК-дисплей з адаптером I2C та DS3231 використовують один і той самий протокол зв'язку I2C, тому вони підключаються до одних і тих самих пінів Arduino: A4 для SDA та A5 для SCL.

Протокол I2C дозволяє підключати кілька пристроїв до одних і тих самих пінів завдяки унікальним адресам кожного пристрою на шині I2C.

Отже, на РК-дисплеї 16x2 можна побачити по черзі три різних екрани, кожен з яких відображає певну комбінацію метеорологічних даних та поточної дати/часу. Дані будуть оновлюватися кожні 10 секунд (згідно з updateInterval), а зміна екрану відбуватиметься кожні 5 секунд (згідно з displayInterval).

Повню схему підключень для всієї метеостанції можна побачити в **Додатку Б**.

2.3 Висновки до 2 розділу

У другому розділі роботи було детально розроблено структурну схему автоматизованої метеостанції. Ця схема є ключовим елементом проєктування, оскільки вона докладно визначає взаємодію між усіма апаратними компонентами системи, охоплюючи весь її життєвий цикл – від початкового збору даних за допомогою сенсорів до їхнього подальшого відображення та, у разі потреби, передачі. Деталізація цієї схеми забезпечує повне розуміння функціональних зв'язків та інформаційних потоків у межах станції.

Вибір конкретних електронних компонентів для реалізації проєкту не був довільним, а ґрунтувався на глибокому порівняльному аналізі доступних на ринку рішень. Метою цього аналізу було обґрунтування оптимального вибору, який би найкраще відповідав поставленим вимогам до функціональності та ефективності автоматизованої метеорологічної станції.

Порівняння зосереджувалося на кількох ключових критеріях: перш за все, оцінювалася простота інтеграції компонентів у загальну систему. Це важливий аспект, оскільки складність підключення та налаштування може значно збільшити час і вартість розробки. Компоненти, які легко взаємодіють між собою та з обраною мікроконтролерною платформою, є пріоритетними.

По-друге, аналізувалася здатність кожного компонента задовольняти функціональні вимоги станції. Ці вимоги охоплюють не лише базовий збір метеоданих (температура, вологість, тиск), а й ефективну їхню обробку та

зручне відображення для користувача. Важливо було переконатися, що обрані сенсори забезпечуватимуть необхідну точність та діапазони вимірювань, а засоби відображення – адекватну читабельність.

Третій критерій стосувався наявності необхідної документації та підтримки спільноти розробників. Цей фактор є критично важливим, особливо для проєктів, що реалізуються в освітніх або аматорських цілях. Наявність великої кількості навчальних матеріалів, прикладів коду, готових бібліотек та активної спільноти, що може надати допомогу, безпосередньо впливає на швидкість та якість розробки, дозволяючи оперативно вирішувати виникаючі питання.

Кінцевою метою всіх цих порівнянь була оптимізація метеостанції за ключовими критеріями. Це включало досягнення високої точності вимірювань (для отримання достовірних метеорологічних даних), забезпечення можливості підключення до мережі Інтернет (для віддаленого доступу до метеоданих), зниження собівартості (для забезпечення економічної доцільності проєкту) та підвищення загальної надійності пристрою (що гарантує його стабільну та довготривалу роботу).

Виходячи з проведеного аналізу, для реалізації проєкту були обрані наступні компоненти: мікроконтролер Arduino Uno R3, вибір якого пояснюється винятковою простотою використання та широкою підтримкою спільноти, що робить його ідеальним для базових завдань та швидкого прототипування; датчики DHT22 та BME280, обрані за їхню високу точність вимірювань та можливість отримання комплексних даних про атмосферні умови; економічний РК-дисплей 16x2, визнаний достатнім для відображення основної інформації у текстовому форматі, що забезпечує чіткість та читабельність; модуль Wi-Fi ESP8266, обраний для забезпечення можливості передачі даних через інтернет; та модуль реального часу DS3231, який є критично важливим для точної реєстрації даних, забезпечуючи їх коректну прив'язку до часу незалежно від стану живлення системи.

Таким чином, ретельне порівняння різних електронних компонентів та їхній обґрунтований вибір є не просто переліком їхніх характеристик, а стратегічним інструментом в проєктуванні. Цей підхід дозволяє приймати обґрунтовані інженерні рішення, спрямовані на створення функціональної та економічно доцільної автоматизованої метеорологічної станції, яка повністю відповідає поставленим цілям та вимогам проєкту. Розроблена детальна структурна схема метеостанції з чітко визначеними компонентами та докладним описом їхнього підключення є фундаментальною основою для наступного етапу розробки програмного забезпечення, яке керуватиме всіма процесами станції та забезпечить ефективну обробку отриманої метеорологічної інформації.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ НА МЕТЕОСТАНЦІЇ

3.1 Огляд платформи Arduino

Платформа Arduino – це цілісна екосистема з відкритим вихідним кодом, яка об'єднує як апаратне забезпечення, представлене широким розмаїттям плат мікроконтролерів, так і програмне забезпечення у вигляді інтегрованого середовища розробки Arduino IDE. Ця інтеграція апаратних і програмних компонентів забезпечує зручний та доступний інструментарій для розробки електронних проєктів. Ключовою особливістю Arduino є її фундаментальна орієнтованість на зручність використання для користувачів з різним рівнем технічної підготовки, що значно знижує поріг входження у світ мікроконтролерів та програмування [14]

Середовище розробки Arduino IDE є крос-платформним застосунком, доступним для основних операційних систем, таких як Windows, macOS та Linux. Воно надає інтуїтивно зрозумілий інтерфейс, призначений для написання програмного коду, відомого в екосистемі Arduino як "скетч", його компіляції та подальшого завантаження на плати Arduino. Серед ключових функціональних можливостей IDE варто виділити: редактор коду з підсвічуванням синтаксису, що покращує читабельність та зручність написання; інтегровані кнопки для швидкої перевірки та завантаження розробленого коду на плату, що спрощує процес розробки; вбудований послідовний монітор, який є незамінним інструментом для налагодження роботи програми та відображення даних у реальному часі; а також менеджер бібліотек, що значно спрощує процес встановлення та керування зовнішніми бібліотеками, суттєво розширюючи функціональні можливості платформи.

Мова програмування Arduino ґрунтується на потужних і широко розповсюджених мовах C та C++, проте вона має значно спрощений синтаксис, що робить її доступнішою для початківців. Крім того, вона включає додаткові вбудовані функції та спеціалізовані бібліотеки, розроблені

виключно для платформи Arduino. Такий підхід дозволяє розробникам абстрагуватися від деяких складних та низькорівневих аспектів програмування мікроконтролерів, таких як пряма робота з регістрами, що значно спрощує процес розробки та підвищує його доступність. Базова структура будь-якого скетчу Arduino строго визначена та включає дві основні функції: setup(), яка виконується лише один раз при запуску або скиданні плати для ініціалізації всіх необхідних параметрів та компонентів, та loop(), яка виконується безперервно та циклічно, забезпечуючи постійну роботу програми та її реакцію на зовнішні подразники.

Однією з найбільш вагомих переваг екосистеми Arduino є наявність великої та постійно зростаючої колекції бібліотек. Ці бібліотеки розробляються як активною та розгалуженою спільнотою користувачів, так і офіційно підтримуються розробниками Arduino. Вони містять попередньо написаний, оптимізований та протестований код, призначений для взаємодії з різноманітними апаратними компонентами, такими як широкий спектр датчиків (температури, вологості, тиску, освітленості тощо), різноманітні типи дисплеїв (РК, OLED), комунікаційні модулі (Wi-Fi, Bluetooth, LoRa) та інші периферійні пристрої. Крім того, бібліотеки охоплюють реалізацію загальних функціональних можливостей, наприклад, роботу з мережевими протоколами або файловими системами. Така широка підтримка бібліотек значно спрощує та прискорює процес розробки, дозволяючи інженерам та ентузіастам використовувати готові, перевірені рішення, замість написання коду з нуля, що економить час та зменшує ймовірність помилок [15]

Підсумовуючи, можна з упевненістю констатувати, що простота використання платформи Arduino, її інтуїтивно зрозуміле інтегроване середовище розробки, спрощена мова програмування, що базується на C/C++, та величезна підтримка різноманітних бібліотек роблять її ідеальним вибором для розробки програмного забезпечення для нашої автоматизованої метеостанції. Ці фундаментальні переваги забезпечують швидке прототипування, що дозволяє оперативно перевіряти концепції; ефективну

розробку, завдяки наявності готових інструментів та компонентів; та легке масштабування проєкту, дозволяючи додавати новий функціонал та розширювати можливості системи у майбутньому.

3.2 Обґрунтування середовища та мови програмування для автоматизації процесів на метеостанції

Вибір Arduino IDE як основного середовища розробки для проєкту автоматизованої метеостанції є цілком виправданим і ґрунтується на низці значущих переваг. Насамперед, крос-платформна сумісність Arduino IDE є критично важливою, оскільки вона дозволяє здійснювати розробку програмного забезпечення незалежно від використовуваної операційної системи. Будь то Windows, macOS, чи Linux – розробники отримують гнучкість у виборі робочого середовища, що сприяє ефективності та доступності процесу розробки.

Іншою значною перевагою є зручний та інтуїтивно зрозумілий інтерфейс середовища розробки. Ця особливість істотно спрощує процес написання програмного коду, відомого як "скетчі", та його подальшого завантаження на плату мікроконтролера Arduino. Завдяки цій простоті, навіть розробники-початківці можуть швидко освоїти основні функції IDE та ефективно розпочати роботу над проєктом, мінімізуючи час на навчання та адаптацію.

Особливо цінним для нашого проекту є інтегрований менеджер бібліотек Arduino IDE. Цей інструмент забезпечує легкий та зручний доступ до широкого спектра вже готових бібліотек, спеціально розроблених для взаємодії з різноманітними апаратними компонентами, які ми плануємо використовувати в метеостанції. Це включає бібліотеки для роботи з різними типами датчиків (температури, вологості, тиску тощо), дисплеями (РК, OLED) та комунікаційними модулями (Wi-Fi ESP8266, RTC DS3231). Легкість встановлення та керування цими бібліотеками значно прискорює процес

розробки та дозволяє використовувати вже перевірений та оптимізований код. [16]

Обрана для програмування платформа Arduino мова, що базується на C/C++, також є оптимальним вибором для нашого проекту. Вона пропонує вдалий баланс між високорівневими абстракціями, які полегшують написання коду та роблять його більш зрозумілим, та низькорівневим контролем над апаратними ресурсами мікроконтролера. Ефективність мови C/C++ є критично важливою для вбудованих систем, які часто мають обмежені обчислювальні ресурси та обсяг пам'яті. Крім того, велика кількість вже існуючих бібліотек C/C++ для Arduino надає значну перевагу з точки зору доступного готового коду та реалізації різноманітних функціональних можливостей, необхідних для роботи з метеорологічними датчиками та протоколами зв'язку. [17]

Хоча у світі розробки вбудованих систем існують потужні та функціональні альтернативні середовища розробки, такі як PlatformIO, яке часто використовується у поєднанні з багатофункціональним редактором коду VS Code, і які пропонують значно розширені можливості, особливо корисні для великомасштабних та складних проєктів, Arduino IDE зберігає свої унікальні переваги. Ці переваги проявляються насамперед у її спрощеному та орієнтованому безпосередньо на платформу Arduino робочому процесі. PlatformIO, безперечно, надає розробникам доступ до широкого спектра мікроконтролерних платформ, інструментів для автоматизації збірки, розширених можливостей налагодження та інтеграції з системами контролю версій. Це робить його привабливим для професійних розробників та команд, які працюють над комплексними комерційними проєктами з суворими вимогами до оптимізації та управління кодом. Однак, саме ця багатофункціональність може стати перешкодою для тих, хто тільки починає свій шлях у програмуванні мікроконтролерів або працює над проєктами з помірною складністю.

У контексті нашого проєкту автоматизованої метеостанції, масштаб якого є помірним, а основний фокус зосереджений саме на екосистемі Arduino, простота та зручність Arduino IDE є вагомою та вирішальною перевагою. Arduino IDE розроблено таким чином, щоб максимально спростити вхідний поріг для користувачів, надаючи інтуїтивно зрозумілий інтерфейс для написання коду, його компіляції та завантаження на плату. Відсутність надлишкових функцій дозволяє зосередитися на основних завданнях розробки без відволікання на складні налаштування чи конфігурації. Вона пропонує такий досвід, де підключення плати, вибір її типу та завантаження першого скетчу займають лічені хвилини. Хоча PlatformIO може запропонувати вищу гнучкість і контроль для досвідчених користувачів, для нашого проєкту, який тісно інтегрований з філософією Arduino, простота та специфічна адаптація Arduino IDE забезпечують оптимальний баланс між функціональністю та легкістю використання, що у підсумку прискорює процес розробки та підвищує його ефективність.

Щодо візуальних мов програмування, таких як XOD, вони пропонують принципово альтернативний підхід до розробки програмного забезпечення, відходячи від традиційного написання коду в текстовому редакторі. Ці середовища дозволяють створювати логіку програми шляхом перетягування та з'єднання графічних блоків, що представляють функції, компоненти або цілі алгоритми. Такий підхід може бути надзвичайно привабливим для користувачів, оскільки він візуалізує потік даних та виконання операцій, роблячи процес розробки більш інтуїтивним та доступним. Це ідеально підходить для швидкого створення простих прототипів, демонстрації концепцій або для освітніх цілей, де акцент робиться на розумінні логіки, а не на деталях синтаксису чи структури коду.

Однак, ця ж надмірна абстракція апаратних деталей, яка є перевагою для початківців, може стати суттєвим обмеженням при роботі над більш складними проєктами або коли потрібен глибокий контроль над апаратним забезпеченням. Візуальні мови програмування, за своєю суттю, створюють

шар абстракції між розробником і мікроконтролером, що потенційно може обмежити рівень контролю, необхідний для реалізації специфічних або складних функцій.

Для нашого проєкту метеостанції, де необхідні точні вимірювання, можливі розширення функціоналу та проведення оптимізацій, які вимагають специфічного апаратного доступу, обмеження, що накладаються візуальними мовами програмування, можуть бути значними. Тому, вибір традиційного текстового програмування на C/C++ в Arduino IDE є більш виправданим з точки зору гнучкості та можливості реалізації всіх необхідних оптимізацій та складних функцій.

Висновок полягає в тому, що вибір Arduino IDE в поєднанні з мовою програмування на основі C/C++ забезпечує надійну, доступну та ефективну платформу для розробки програмного забезпечення для нашої автоматизованої метеостанції. Ми зможемо повною мірою скористатися перевагами екосистеми Arduino, включаючи велику кількість бібліотек та активну спільноту, для ефективної та результативної реалізації всіх запланованих функціональних можливостей.

3.3 Розробка програмної частини системи

3.3.1 Вибір програмних засобів для реалізації задачі

Для успішної та ефективної реалізації нашої автоматизованої метеостанції критично важливим етапом є вибір відповідних програмних бібліотек. Ці бібліотеки відіграють центральну роль, оскільки вони забезпечують не лише безперебійну взаємодію з різноманітними апаратними компонентами (такими як датчики, дисплеї та комунікаційні модулі), але й дозволяють реалізувати весь необхідний функціонал системи. Правильний вибір бібліотек дозволяє значно спростити процес розробки, мінімізувати кількість помилок та оптимізувати використання ресурсів мікроконтролера, що є ключовим для створення надійної та функціональної метеостанції.

Для роботи з обраним нами високоточним датчиком BME280 найбільш оптимальним вибором є добре підтримувана бібліотека Adafruit BME280. Ця бібліотека надає повний набір функцій для ініціалізації датчика, зчитування точних даних про температуру, вологість та атмосферний тиск. Вона також забезпечує підтримку обох основних протоколів зв'язку, які може використовувати BME280 – I²C та SPI. Варто зазначити, що для коректної роботи Adafruit BME280 часто вимагає встановлення залежної бібліотеки Adafruit Unified Sensor.

У випадку використання більш простих датчиків DHT11 або DHT22, чудово підійде спеціалізована бібліотека DHT sensor library від Adafruit. Вона забезпечує зручний інтерфейс для зчитування даних температури та вологості з цих датчиків.

Для керування блоком відображення також існують зручні бібліотеки. У випадку використання стандартного РК-дисплея, достатньо буде вбудованої в Arduino IDE бібліотеки LiquidCrystal. Вона надає базові функції для відображення текстової метеорологічної інформації. Якщо ж використовується РК-дисплей з інтерфейсом I²C, більш зручною альтернативою є бібліотека LiquidCrystal_I2C, яка значно спрощує процес підключення дисплея, використовуючи меншу кількість контактів мікроконтролера.

Для роботи з OLED-дисплеєм рекомендується використовувати потужну бібліотеку Adafruit SSD1306, яка часто використовується в поєднанні з графічною бібліотекою Adafruit GFX. Цей комплект бібліотек надає широкі можливості для виведення не лише тексту, але й різноманітних геометричних фігур та навіть бітових зображень на OLED-екрані, що може бути корисним для створення більш інформативного та візуально привабливого інтерфейсу.

У випадку реалізації підключення до Інтернету за допомогою модуля ESP8266, незамінною стане бібліотека ESP8266WiFi. Вона надає необхідні функції для підключення до бездротових мереж Wi-Fi та встановлення різноманітних мережевих з'єднань, що є основою для передачі даних на

віддалені сервери або онлайн-платформи. Для безпосередньої передачі даних можуть знадобитися додаткові бібліотеки, що підтримують конкретні протоколи обміну даними. Наприклад, для роботи з протоколом MQTT можуть використовуватися бібліотеки PubSubClient або ArduinoMqttClient, а для здійснення HTTP-запитів існують відповідні бібліотеки, що полегшують взаємодію з веб-сервісами. [18]

Якщо в проекті передбачається локальне зберігання даних на карті пам'яті SD, стандартна бібліотека SD (SD.h) надає необхідну функціональність для читання та запису файлів на SD-карті, підключеній до мікроконтролера Arduino Uno R3 через відповідний SD-кардридер.

Нарешті, для забезпечення точної реєстрації часу за допомогою модуля годинника реального часу (RTC), такого як DS3231, популярним та легким у використанні варіантом є бібліотека RTClib. Вона надає зручні функції для ініціалізації модуля RTC, зчитування поточного часу та дати, а також їх встановлення, що є критично важливим для коректної часової мітки зібраних метеорологічних даних.

Вибір саме цих програмних засобів для розробки автоматизованої метеостанції був обґрунтований кількома ключовими факторами. По-перше, їхня надійність є пріоритетом, оскільки вона гарантує стабільну та безперебійну роботу системи моніторингу впродовж тривалого часу. По-друге, широка підтримка спільнотою Arduino відіграє вирішальну роль; це забезпечує доступ до величезної кількості ресурсів, таких як форуми, блоги, приклади коду та готові рішення, що є неоціненним для вирішення можливих проблем та оптимізації функціоналу. По-третє, наявність зрозумілої документації значно спрощує процес вивчення та використання обраних інструментів, дозволяючи розробникам швидко опановувати їхні можливості. Нарешті, велика кількість прикладів використання дозволяє адаптувати вже існуючі рішення до конкретних потреб проєкту, що суттєво полегшує та прискорює процес розробки програмного забезпечення для нашої автоматизованої метеостанції.

3.3.2 Опис програмних функцій та модулів

Програмне забезпечення автоматизованої метеостанції розроблено як комплексна система, що складається з кількох взаємопов'язаних модулів, кожен з яких відповідає за певний етап обробки інформації або взаємодію з апаратними компонентами (блок-схема алгоритму роботи системи представлена в **Додатку Б**).

Основний принцип роботи програми полягає в безперервному циклі, який включає збір метеоданих, їхню обробку, відображення для локального користувача та, за наявності відповідного обладнання, передачу на віддалені сервери для зберігання та аналізу.

Першим і надзвичайно важливим етапом у роботі нашої автоматизованої метеостанції є ***ініціалізація***. Ця фаза виконується лише один раз — при першому запуску мікроконтролера Arduino. Її основна мета — підготувати всі необхідні апаратні та програмні компоненти до подальшої коректної роботи.

У межах ініціалізації відбувається налаштування послідовного порту. Цей порт слугує основним інструментом як для налагодження системи, так і для виведення діагностичної інформації, що дозволяє контролювати стан пристрою та оперативно виявляти потенційні проблеми.

Крім того, на етапі ініціалізації відбувається повний запуск та підготовка всіх датчиків, відповідальних за збір метеорологічних даних. Це включає датчик температури та вологості DHT22, а також високоточний датчик атмосферного тиску BME280.

Для кожного з цих сенсорів система визначає відповідні піни підключення на мікроконтролері Arduino, забезпечуючи фізичний зв'язок. Після цього відбувається запуск відповідних програмних бібліотек, які надають зручні та оптимізовані функції для взаємодії з цими датчиками, дозволяючи легко зчитувати дані та керувати їхньою роботою. Таким чином, ініціалізація створює стабільне та функціональне середовище для подальших вимірювань та обробки інформації.

Окрім сенсорів, на етапі ініціалізації налаштовується локальний дисплей, який може бути представлений РК-екраном (з паралельним або I²C підключенням) або більш сучасним OLED-дисплеєм. Для керування дисплеєм використовуються спеціалізовані бібліотеки, які дозволяють виводити текстову та графічну інформацію.

Якщо в проекті передбачено підключення до мережі Інтернет, ініціалізується Wi-Fi модуль ESP8266, здійснюється підключення до заданої бездротової мережі. Для забезпечення точної часової прив'язки зібраних даних ініціалізується модуль годинника реального часу DS3231, який зберігає точний час незалежно від основного живлення.

Після завершення етапу ініціалізації програма переходить до основного циклу ***збору даних***, який виконується безперервно у функції `loop()`. На цьому етапі відбувається періодичне опитування всіх підключених датчиків. З датчика DHT22 зчитуються поточні значення температури та вологості повітря.

Датчик BME280 надає більш повний набір даних, включаючи температуру, вологість та атмосферний тиск. Для кожного отриманого набору метеорологічних даних з модуля RTC зчитується точний час, що дозволяє зафіксувати момент часу кожного вимірювання. У випадку, якщо в системі передбачено локальне зберігання даних, зібрана інформація може бути записана у файла на SD-карту.

На основі отриманих метеорологічних даних реалізується простий алгоритм ***прогнозування погоди***. На поточному етапі основним критерієм для прогнозу є тенденція зміни атмосферного тиску. Програма аналізує динаміку тиску за певний проміжок часу.

Зокрема, різке падіння тиску може розцінюватися як ознака погіршення погодних умов, наприклад, наближення шторму чи циклону. Натомість, повільне зростання тиску може свідчити про покращення погоди та встановлення стабільних антициклональних умов. Цей алгоритм, хоча й

базовий, дозволяє надати користувачеві елементарний прогноз на основі ключового метеорологічного параметра.

Зібрані та оброблені дані про метеорологічні параметри, а саме значення температури, вологості, атмосферного тиску та поточний час, готуються до ***відображення на локальному дисплеї***. Ці дані ретельно форматуються у вигляді чітких текстових рядків.

Після форматування, вони виводяться на екран РК- або OLED-дисплея за допомогою спеціалізованих функцій відповідних програмних бібліотек. Такий підхід забезпечує користувачеві безпосередній та зручний доступ до актуальної метеорологічної інформації, дозволяючи йому швидко оцінити поточні погодні умови.

Весь процес збору, обробки та відображення даних відбувається в автоматичному режимі в нескінченному циклі `loop()`. Частота оновлення даних регулюється за допомогою введення невеликих затримок у виконанні програми. [19]

У випадку наявності підключення до мережі Інтернет через модуль ESP8266, реалізується функціонал ***передачі даних на віддалений сервер***. Зібрані метеорологічні дані та час періодично відправляються на зовнішній сервер або спеціалізовану онлайн-платформу для зберігання, візуалізації та подальшого аналізу.

Для передачі даних використовуються стандартні інтернет-протоколи, такі як HTTP або MQTT, та відповідні бібліотеки Arduino. Дані перед відправкою форматуються у зручний для обробки формат, наприклад, JSON або CSV. При використанні певних онлайн-платформ, таких як ThingSpeak, надіслані дані можуть автоматично відображатися у вигляді графіків на веб-інтерфейсі користувача. [20]

Вибір ***ThingSpeak*** для нашого проєкту обґрунтований його численними перевагами, які роблять цю платформу оптимальним рішенням для демонстрації можливостей автоматизації збору даних та їх інтеграції з IoT-платформами. По-перше, ThingSpeak вирізняється простотою інтеграції з

мікроконтролерами завдяки зручному HTTP API. Це дозволяє легко надсилати дані з Arduino та інших мікроконтролерів без складних налаштувань.

По-друге, доступність безкоштовного тарифного плану для освітніх та некомерційних проєктів робить ThingSpeak надзвичайно привабливим, дозволяючи реалізувати функціональність без додаткових фінансових витрат. Крім того, платформа має вбудовані інструменти візуалізації даних, що дозволяє швидко та наочно представляти зібрану метеорологічну інформацію у вигляді графіків та діаграм без необхідності використання стороннього програмного забезпечення.

ThingSpeak також пропонує потужні можливості моніторингу та аналізу даних, включаючи функції обробки даних у реальному часі та налаштування сповіщень. Його широка доступність як хмарної платформи та розлога документація з численними прикладами значно полегшують процес розробки та налагодження. Всі ці фактори у сукупності роблять ThingSpeak ідеальним вибором для даного проєкту, що має на меті продемонструвати ефективність збору та аналізу даних у контексті Інтернету речей.

Для забезпечення зручності читання, розуміння та подальшої підтримки програмного коду, його було написано з урахуванням принципів чіткого структурування. Це включає логічне групування функціональних блоків, використання значущих імен змінних та функцій, що відображають їхнє призначення, а також додавання детальних коментарів, які пояснюють призначення різних фрагментів коду. Такий підхід значно полегшує аналіз коду іншими розробниками, а також його модифікацію та налагодження у майбутньому.

Хоча основною мовою програмування для Arduino є C/C++, використання високорівневих бібліотек дозволяє істотно спростити процес розробки. Ці бібліотеки надають абстракції над низькорівневими апаратними операціями, дозволяючи розробнику зосередитися на логіці проєкту. Крім того, застосування стандартизованих бібліотек забезпечує певну міру переносимості коду між різними моделями плат Arduino, оскільки бібліотеки

адаптують функції до конкретної архітектури мікроконтролера, мінімізуючи необхідність у переписуванні коду при зміні апаратної платформи.

3.3.3 Алгоритм функціонування програмного коду

Автоматизована метеостанція розпочинає свою роботу з підключення необхідних бібліотек, таких як DHT.h для датчика DHT22, Wire.h для I2C комунікації з BME280, РК-дисплеєм та RTC, Adafruit_Sensor.h як базової бібліотеки для сенсорів Adafruit, Adafruit_BME280.h для датчика BME280, LiquidCrystal_I2C.h для керування РК-дисплеєм через I2C, RTClib.h для модуля реального часу DS3231 та SoftwareSerial.h для зв'язку з ESP8266.

Далі визначаються піни Arduino та константи, включаючи номери пінів датчиків, їх типи, I2C адреси пристроїв та розміри РК-дисплея. Створюються об'єкти для кожного підключеного компонента: dht для DHT22, bme для BME280, lcd для РК-дисплея, rtc для DS3231 та esp8266 для Wi-Fi модуля. Оголошуються змінні для зберігання метеорологічних даних, таких як температура, вологість, тиск, попередній тиск та прогноз погоди. Також визначаються таймери для періодичного оновлення даних та дисплея, а змінна displayState використовується для циклічної зміни відображуваної інформації.

Функція `setup()` виконується одноразово при запуску Arduino і відповідає за ініціалізацію всіх компонентів. Вона ініціалізує послідовний порт, РК-дисплей, датчики DHT22 та BME280 (з перевіркою на помилки), модуль RTC (з можливістю встановлення початкового часу), зчитує початкове значення тиску та виводить повідомлення про готовність системи. У додатковій частині викликається функція `setupWiFi()` для налаштування Wi-Fi модуля ESP8266. [21]

Основна логіка роботи метеостанції міститься у функції loop(), яка виконується циклічно. Вона отримує поточний час і перевіряє, чи минув встановлений інтервал для оновлення метеоданих (10 секунд) або оновлення дисплея (5 секунд). Якщо час оновлення даних настав, викликаються функції `updateMeasurements()`, `updateForecast()` та `printDebugInfo()`.

Функція updateMeasurements() зчитує дані з датчиків DHT22 та BME280, обробляючи можливі помилки зчитування з DHT22, та оновлює значення тиску. Функція `updateForecast()` виконує простий прогноз погоди на основі зміни атмосферного тиску. Функція `printDebugInfo()` виводить поточні метеодані та час у послідовний порт для налагодження. Якщо настав час оновлення дисплея, викликається функція `updateDisplay()`, яка очищає дисплей та відображає різну інформацію (температуру/вологість, тиск/прогноз, дату/час) залежно від значення змінної displayState, яка циклічно змінюється.

Додатковий код для Wi-Fi включає функції `setupWiFi()`, `sendCommand()`, `sendDataToThingSpeak()` та `processWiFi()`. `setupWiFi()` налаштовує зв'язок з ESP8266 та підключається до Wi-Fi мережі. `SendCommand()` використовується для відправки AT-команд на ESP8266. `SendDataToThingSpeak()` формує HTTP GET запит з метеоданими та відправляє його на платформу ThingSpeak. processWiFi() періодично викликає `sendDataToThingSpeak()` для передачі даних. (Приклад програмного коду наведений у **Додатку Б**)

Таким чином, функціональна логіка програми забезпечує безперервне зчитування даних з датчиків, що є основою для подальшої обробки інформації. Одночасно з цим, програма оновлює прогноз погоди, ґрунтуючись на аналізі зібраних метеорологічних параметрів. Для цілей налагодження та моніторингу системи, вся релевантна інформація виводиться у налагоджувальний інтерфейс.

Паралельно відбувається циклічне відображення метеоданих на РК-дисплеї, що забезпечує користувачеві доступ до актуальної інформації. Крім того, за умови інтеграції відповідного програмного коду, зібрані та оброблені дані також передаються на онлайн-платформу через Wi-Fi, розширюючи можливості моніторингу та аналізу інформації.

3.4 Висновки до 3 розділу

У третьому розділі було розроблене програмне забезпечення на базі Arduino Uno, а також здійснений опис програмних функцій та модулів автоматизованої метеостанції, наведений алгоритм функціонування програмного коду. Обрані для програмування середовище Arduino IDE та мова програмування C/C++ як оптимальні для реалізації задачі, а також обґрунтований вибір програмних засобів.

Розроблена програмна частина для автоматизованої метеостанції успішно реалізує всі вимоги, зазначені у першому розділі.

Висока точність збору та обробки метеорологічних даних у системі метеостанції забезпечується інтеграцією спеціалізованих датчиків: DHT22 для основних вимірювань температури та вологості, та BME280 для атмосферного тиску. BME280 також слугує резервним джерелом даних про температуру і вологість, враховуючи його потенційний самонагрів, який може впливати на точність температури. Для синхронізації кожного вимірювання використовується модуль RTC, що забезпечує точну фіксацію часу.

Автоматизація збору даних відбувається через безперервний цикл програми, де команда loop() постійно опитує датчики, забезпечуючи автоматичний збір метеорологічних даних.

Для відображення інформації на метеоприладі використовуються бібліотеки для РК (LiquidCrystal, LiquidCrystal_I2C) та OLED (Adafruit SSD1306, Adafruit GFX) дисплеїв. Зібрані дані форматуються та виводяться на локальний дисплей функцією updateDisplay(), яка оновлює інформацію про температуру, вологість, тиск, прогноз і час.

Можливість підключення до Інтернету для віддаленого доступу забезпечується бібліотекою ESP8266WiFi для Wi-Fi та протоколами HTTP/MQTT (PubSubClient, ArduinoMqttClient). Дані форматуються в JSON/CSV та передаються на платформи на кшталт ThingSpeak або Arduino

Cloud функціями setupWiFi(), sendCommand(), sendDataToThingSpeak() та processWiFi().

Використання *доступних за вартістю* компонентів робить проект економічно вигідним завдяки платформі Arduino.

Отже, розроблене програмне забезпечення успішно відповідає всім вимогам першого розділу завдяки вибору компонентів та ефективній програмній реалізації на базі Arduino.

ВИСНОВКИ

Розробка програмного забезпечення для автоматизації процесів на метеостанції з використанням платформи Arduino є складним, але цілком здійсненним завданням. Цей звіт охопив ключові етапи цього процесу, починаючи з аналізу існуючих рішень та визначення вимог до проектованої системи, і закінчуючи вибором апаратних компонентів та детальним описом розробки програмної частини.

Автоматизація метеорологічного моніторингу надає значні переваги, включаючи безперервний збір даних, аналіз у реальному часі та можливість віддаленого доступу до інформації. Платформа Arduino, особливо у поєднанні з мікроконтролером Arduino Uno R3, виявилася вдалим вибором для реалізації такого проекту завдяки своїй простоті використання, широкій підтримці бібліотек та великій спільноті користувачів.

Для подальшого розвитку проекту можна розглянути можливість інтеграції більш просунутих датчиків, таких як датчики швидкості та напрямку вітру, а також кількості опадів. Також можна розширити функціональність системи шляхом реалізації більш складних алгоритмів прогнозування погоди, розробки інтерактивного веб-інтерфейсу (з використанням ESP8266), та впровадження більш просунутих методів аналізу та візуалізації даних.

На завершення слід зазначити, що використання платформи Arduino є ефективним та перспективним підходом для створення автоматизованих систем метеорологічного моніторингу, придатних для широкого спектра застосувань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Палаш Б. Р. Розробка прогамного забезпечення для автоматизації процесів на метеостанції. Ukraine, 2025. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/24249 (дата звернення: 29.05.2025).
2. Основні види метеостанцій. URL: https://vencon.ua/ua/articles/kak-vybrat-meteostanciyu (дата звернення: 11.05.2025)
3. Ройко О. Ю. Сучасна наука та освіта волині : зб. матеріалів наук.-практ. онлайн-конф., 20 лист. 2020 р., Луцьк, 2020. URL: https://salo.li/fB50025. (дата звернення: 11.05.2025)
4. Hama EWS-800 інструкція користувача. URL: https://salo.li/28Ecf67 (дата звернення: 11.05.2025)
5. WT-137 Comfort Meter with Temp and Humidity. URL: https://www.lacrossetechnology.com/products/wt-137u (дата звернення: 11.05.2025)
6. Wireless weather station with anemometer SPRING BREEZE 35.1140. URL: https://www.tfa-dostmann.de/en/product/wireless-weather-station-with-anemometer-spring-breeze-35-1140/ (дата звернення: 11.05.2025)
7. Грищук Ю.С. Мікроконтролери: архітектура, програмування та застосування в електромеханіці. Харків: НТУ «ХПІ», 2019. 384 с.
8. ESP32 vs Arduino : Definition & the Main Differences. URL: https://acebott.com/stem-blogs/esp32-vs-arduino-what-is-different/ (дата звернення: 11.05.2025)
9. Мій проект. URL: https://myproject.com.ua/porivniannia-datchykiv-temperatury-ta-volohosti-dht11-vs-dht22-vs-bme280.html (дата звернення: 11.05.2025)
10. Liquid Crystal Displays (LCD) with Arduino. URL: https://docs.arduino.cc/learn/electronics/lcd-displays/ (дата звернення: 11.05.2025)

11. Arduino – OLED. URL: https://arduinogetstarted.com/tutorials/arduino-oled#google_vignette (дата звернення: 11.05.2025)
12. Kusriyanto M., Anggara Putra A. Weather station design using IoT platform based on Arduino mega. 2018 IEEE International Symposium on Electronics and Smart Devices (ISESD). 2018. P. 1–4.
13. DS3231. URL: https://docs.arduino.cc/libraries/ds3231/ (дата звернення: 11.05.2025)
14. IoT Based Weather Monitoring System Using Arduino-UNO. URL: https://ieeexplore.ieee.org/document/9357748 (дата звернення: 11.05.2025).
15. What is Arduino? URL: https://www.arduino.cc/en/Guide/Introduction (дата звернення: 11.05.2025).
16. Libraries. URL: https://doc.arduino.ua/ru/prog/Libraries (дата звернення: 11.05.2025).
17. Arduino / C++. URL: https://docs.arduino.cc/arduino-cloud/guides/arduino-c/ (дата звернення: 11.05.2025).
18. Колодій В., Слюсар В. Особливості програмування мікроконтролерів з вбудованим Wi-Fi модулем в середовищі Arduino IDE. 2017. 122 с.
19. Функція Loop. URL: https://doc.arduino.ua/ru/prog/Loop (дата звернення: 11.05.2025).
20. ThingSpeak. URL: https://www.thethingsnetwork.org/docs/applications/thingspeak/ (дата звер нення: 11.05.2025).
21. Функція Setup: https://doc.arduino.ua/ru/prog/Setup (дата звернення: 11.05.2025).
22. Дзендзелюк О., Мусійчук І., Рабик В. Автоматизована система моніторингу параметрів довкілля. *Теоретична електротехніка.* 2010. – №61. – С. 90–98.

23. Євтушенко, Ю. В. (2024). Дослідження параметрів точності мініатюрної метеорологічної станції : пояснюв. зап. до дипл. роботи магістра : радіоелектр. компютериз. засоби. Харків, 2024. 123 с.
24. Оконський М., Лупенко С., Паламар А. Інформаційно-вимірювальна система для контролю метеорологічних параметрів на основі Інтернету речей : матеріали IX наук.-техн. конф. "Інформаційні моделі, системи та технології". Тернопіль, 2021. 118 с.
25. Рисований М. О. Програмне забезпечення системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією. 2024. 97 с.
26. Глухов О.В., Кравчук О.О., Левченко Є.В. Вивчення властивостей мікроконтролерів і електронних систем на базі платформи Ардуіно : навч. посібник для студентів ВНЗ. Харків : ХНУРЕ, 2019. 192 с.
27. Кужель В. В. Мікроконтролерні платформи в IoT мережах : робота на здобуття кваліфікаційного ступеня бакалавра : спец. 171 - електроніка / наук. кер. К. В. Тищенко. Суми : Сумський державний університет, 2023. 36 с.
28. Колосова К. С. Метеостанция на Arduino : матеріали XII Всеукр. студент. наук.-тех. конф. "Сталий розвиток міст". Харків. 2019.
29. Ігнатенко Д., Волощенко І. Автоматизована система вимірювання метеорологічних показників : збірник студент. наук. робіт "Автоматизація та приладобудування". Харків : ADED, 2019. С. 51–54.
30. Рогожин, А. В. Дослідження та розробка системи автоматичного керування метеостанцією : матеріали VII Всеукр. наук-тех. конф. молодих вчених, аспірантів та студентів "Автоматизація, контроль та управління: пошук ідей та рішень". Покровськ. 2021.
31. Цирульник С. М. Мобільні додатки та онлайн-платформи моніторингу даних WI-FI метеостанції : електрон. наук. фах. видання "Відкрите освітнє Е-середовище". 2020.

ДОДАТКИ

Вінницький національний технічний університет
Кафедра автоматизації та інтелектуальних інформаційних технологій

ЗАТВЕРДЖУЮ
д.т.н., професор Олег БІСІКАЛО
_____
(підпис)
«___» _____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на бакалаврську кваліфікаційну роботу
«Розробка програмного забезпечення для автоматизації процесів на метеостанції»
08-31.БКР.011.02.000 ТЗ

Керівник бакалаврської кваліфікаційної роботи
к.т.н., доц. Володимир Гармаш
_____
(підпис)
«___» _____ 2025 р.

Розробив студент гр. АКІТР-23мс
Богдан Палаш
_____
(підпис)
«___» _____ 2025 р.

.

Вінниця ВНТУ – 2025 рік

Додаток А (обов'язковий)

Технічне завдання на бакалаврську кваліфікаційну роботу

1. Підстава для проведення робіт

Підставою для виконання бакалаврської кваліфікаційної роботи на тему: «Розробка програмного забезпечення для автоматизації процесів на метеостанції» є наказ № 97 від 20.03.2025 р.

Термін виконання робіт:

початок 07.03.2025 р.
кінець 10.06.2025 р.

2. Мета та вихідні дані для проведення робіт

Метою бакалаврської кваліфікаційної роботи є автоматизація процесів моніторингу метеорологічних даних з мінімальним втручанням людини, забезпечення достовірності та доступності зібраної інформації.

Вихідними даними для проведення робіт є індивідуальне завдання на бакалаврську кваліфікаційну роботу від 20.03.2025 р.

3. Етапи виконання робіт

Виконавцем всіх перерахованих в даному розділі етапів є: студент групи **АКІТР-23мс Палаш Богдан Русланович** факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№ Етапу	Зміст етапу	Строки виконання
Е1	Аналіз існуючих рішень щодо проектування метеостанцій	07.03 – 19.03
Е2	Визначення вимог до проектованої метеостанції	20.04 – 26.04
Е3	Розробка структурної схеми метеорологічної станції та вибір елементної бази	27.04 – 09.04
Е4	Розробка програмного забезпечення для автоматизації процесів на метеостанції	10.05 – 01.06
Е5	Аналіз результатів роботи та підготовка роботи до захисту	02.06 – 18.06

4. Призначення і галузь застосування

Система призначена для автоматизованого збору, обробки та передачі даних про погодні умови (температура, вологість, атмосферний тиск) з метеостанції до хмарного сервісу ThingSpeak для подальшого моніторингу та аналізу. Метеостанція функціонуватиме в умовах, що передбачають вплив змінних погодних факторів.

5. Технічні дані

Умови експлуатації метеостанції:

Фізичне середовище: Метеостанція функціонуватиме в умовах, що передбачають вплив змінних погодних факторів (температура, вологість, атмосферний тиск).

Живлення: Забезпечується стабільним джерелом живлення для мікроконтролера та датчиків.

Підключення: Метеостанція потребує доступу до мережі Інтернет для передачі даних на ThingSpeak. Використовується модуль ESP8266 для Wi-Fi з'єднання.

Розміщення: Передбачається стаціонарне розміщення метеостанції.

Точність вимірювання:

Датчик температури та вологості DHT22:

- Діапазон вимірювання температури: −40…80°C
- Точність вимірювання температури: ±0.5°C
- Діапазон вимірювання вологості: 0…100 %RH
- Точність вимірювання вологості: ±2−5 %RH

Датчик температури, вологості та атмосферного тиску BME280:

- Діапазон вимірювання температури: −40…85°C
- Точність вимірювання температури: ±1°C
- Діапазон вимірювання вологості: 0…100 %RH
- Точність вимірювання вологості: ±3 %RH
- Діапазон вимірювання тиску: 300…1100 гПа
- Точність вимірювання тиску: ±1 гПа (абсолютна), ±0.12 гПа (відносна)

Склад системи

- Мікроконтролер: Плата Arduino Uno R3
- Wi-Fi модуль: ESP-01 Wi-Fi модуль.
- Датчик температури та вологості повітря: DHT22.
- Датчик температури, вологості та атмосферного тиску: BME280.
- РК-дисплей: LiquidCrystal-I2C 16x2.
- Модуль годинника реального часу (RTC): RTC_DS3231.
- З'єднувальні елементи: Проводи, резистори, макетна плата.

– Джерело живлення: Стабілізоване джерело живлення 5В для Arduino та окреме стабілізоване живлення 3.3В для ESP-01.

6 Джерела розробки

6.1 IoT Based Weather Monitoring System Using Arduino-UNO. URL: https://ieeexplore.ieee.org/document/9357748 (дата звернення: 11.05.2025).
6.2 Глухов О.В., Кравчук О.О., Левченко Є.В. Вивчення властивостей мікроконтролерів і електронних систем на базі платформи Ардуіно: навч. посібник для студентів ВНЗ. Харків: ХНУРЕ, 2019. – 192 с.
6.3 ДСТУ EN ISO 10012:2022 Системи керування вимірюванням. Вимоги до процесів вимірювання та вимірювального обладнання (EN ISO 10012:2003, IDT; ISO 10012:2003, IDT). – Введ. наказом № 285 від 28.12.2022. – К. : 2022.
6.4 ДСТУ EN 60730-2-9:2017 "Пристрої автоматичні електричні керувальні побутової та аналогічної призначеності. Частина 2-9. Додаткові вимоги до температурочутливих керувальних пристроїв (EN 60730-2-9:2010, IDT; IEC 60730-2-9:2008, MOD). – Введ. 01.01.2019. – К. : 2019. 24 с.
6.5 Методичні вказівки до виконання бакалаврських дипломних робіт (проектів) для студентів спеціальностей 126 – «Інформаційні системи та технології», 151 – «Автоматизація та комп’ютерно-інтегровані технології» / Уклад. Р. Н. Квєтний, О. М. Бевз, О. В. Бісікало. – Вінниця : ВНТУ, 2019. – 26 с.

.

Здобувач ст. гр. АКІТР-23мс _____ Богдан ПАЛАШ

Додаток Б (обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИМАЗАЦІЇ ПРОЦЕСІВ НА МЕТЕОСТАНЦІЇ

Схема організації метеостанції

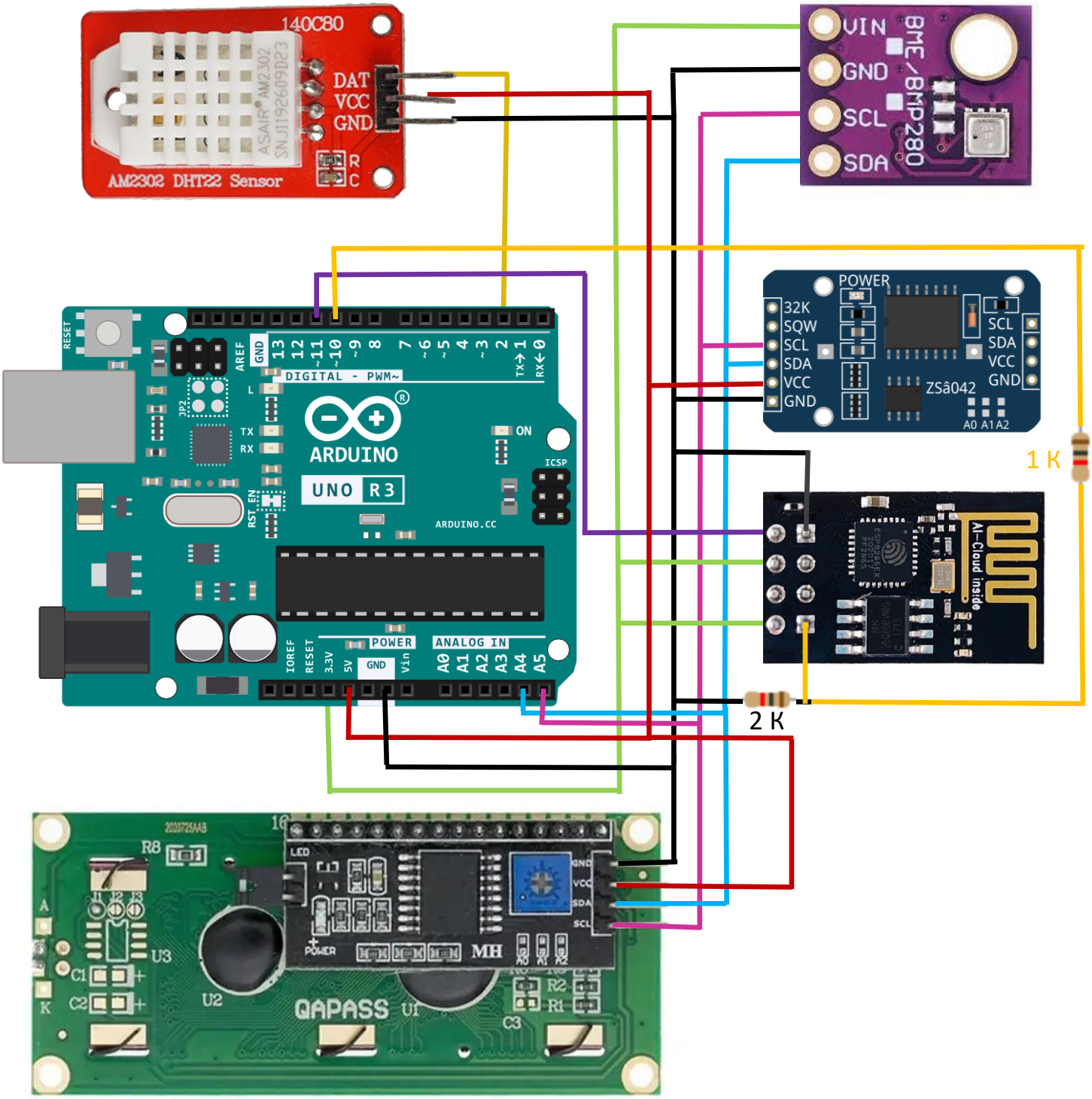


Рисунок Б.1 – Схема організації метеостанції

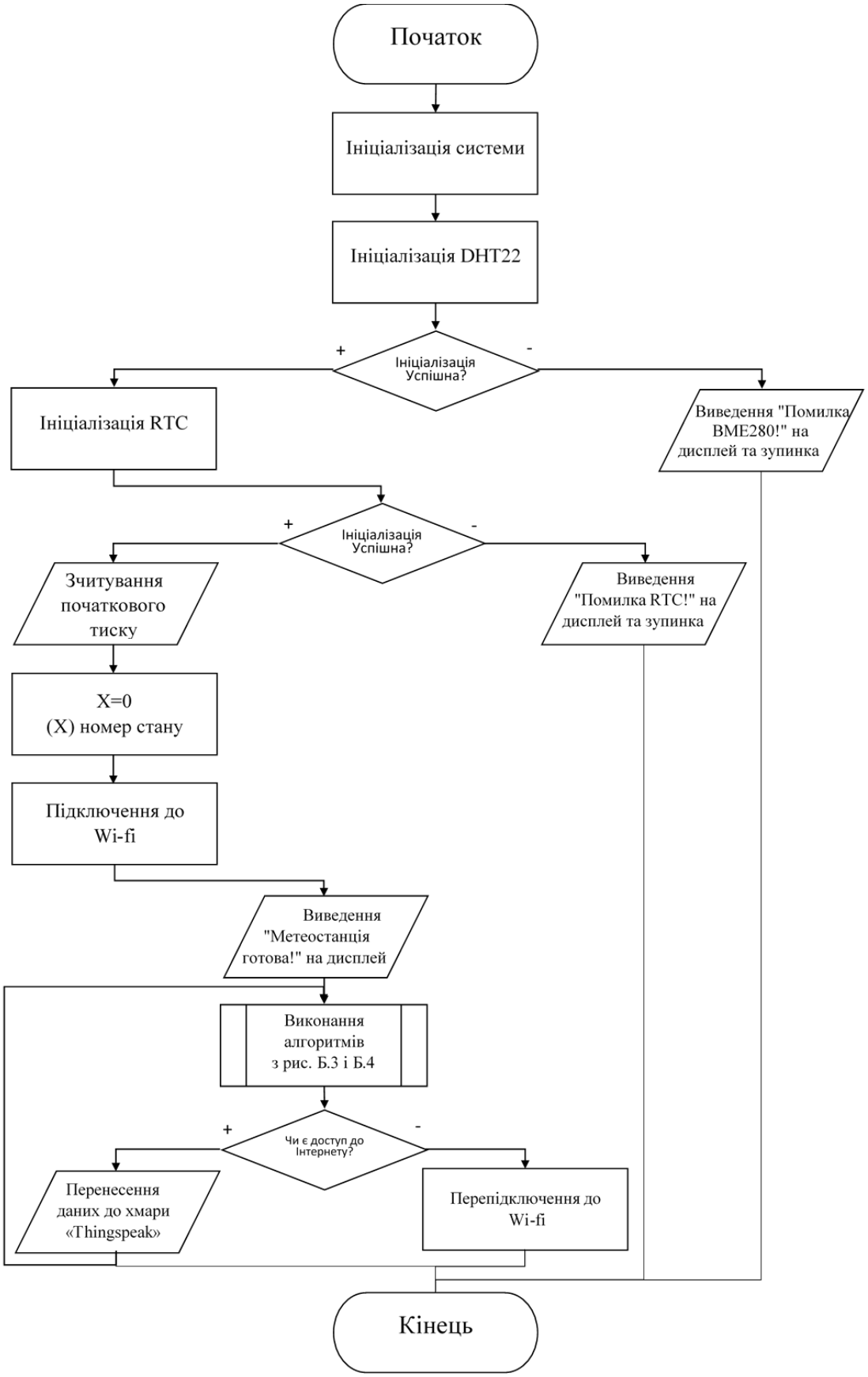
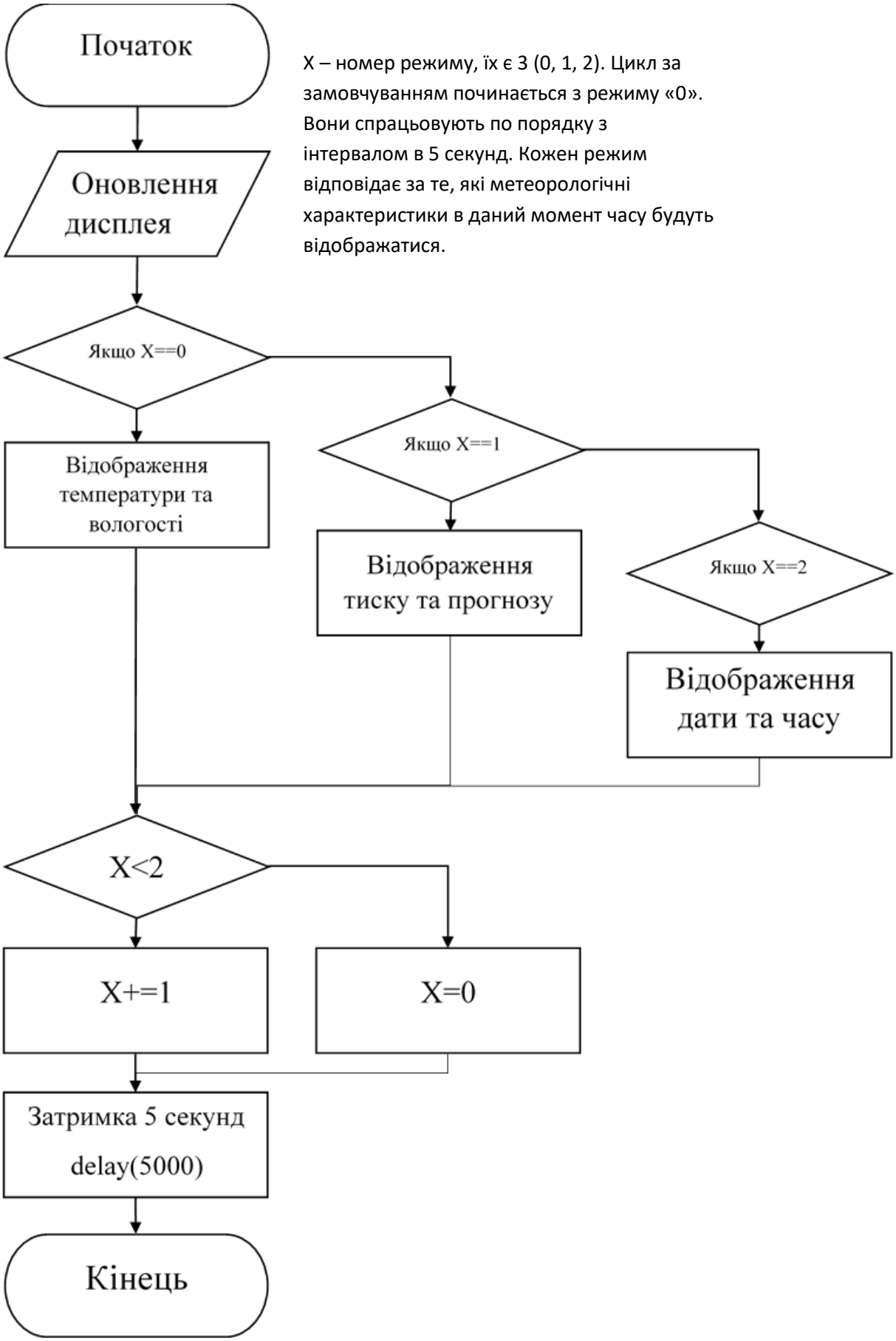


Рисунок Б.2 – Етапи ініціалізації системи



X – номер режиму, їх є 3 (0, 1, 2). Цикл за замовчуванням починається з режиму «0». Вони спрацьовують по порядку з інтервалом в 5 секунд. Кожен режим відповідає за те, які метеорологічні характеристики в даний момент часу будуть відображатися.

Рисунок Б.3 – Алгоритм відображення даних на дисплеї

Рисунок Б.4 – Етапи процесу зчитування даних з датчиків

Додаток В (Обов’язковий)

Лістинг основних функції програми

```
// Базовий скетч для автоматизованої метеостанції
// Включає датчики DHT22, BME280, РК-дисплей 16x2 і модуль RTC DS3231

// Підключення бібліотек
#include <DHT.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <LiquidCrystal_I2C.h>
#include <RTClib.h>

// Визначення пінів і констант
#define DHTPIN 2 // DHT22 підключений до цифрового піну 2
#define DHTTYPE DHT22 // Вказуємо тип датчика DHT
#define BME_ADDR 0x76 // I2C адреса BME280 (може бути 0x77)
#define LCD_ADDR 0x27 // I2C адреса дисплея (стандартно 0x27)
#define LCD_COLS 16 // Кількість стовпців дисплея
#define LCD_ROWS 2 // Кількість рядків дисплея

// Ініціалізація об'єктів
DHT dht(DHTPIN, DHTTYPE);
Adafruit_BME280 bme;
LiquidCrystal_I2C lcd(LCD_ADDR, LCD_COLS, LCD_ROWS);
RTC_DS3231 rtc;

// Змінні для зберігання метеоданих
float temperature = 0;
float humidity = 0;
float pressure = 0;
float previousPressure = 0;
char forecast[10] = "Стабільно";

// Таймери для періодичних завдань
unsigned long lastUpdateTime = 0;
unsigned long lastDisplayTime = 0;
```

```
const long updateInterval = 10000; // Інтервал оновлення даних (10 секунд)
const long displayInterval = 5000; // Інтервал зміни відображення (5 секунд)
int displayState = 0; // Змінна для поточного стану дисплея

void setup() {
 // Ініціалізація послідовного порту для відлагодження
 Serial.begin(9600);
 Serial.println(F("Автоматизована метеостанція"));

 // Ініціалізація дисплея
 lcd.init();
 lcd.backlight();
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Ініціалізація...");

 // Ініціалізація DHT22
 dht.begin();
 Serial.println(F("DHT22 ініціалізовано"));

 // Ініціалізація BME280
 if (!bme.begin(BME_ADDR)) {
 Serial.println(F("Помилка ініціалізації BME280!"));
 lcd.setCursor(0, 1);
 lcd.print("Помилка BME280!");
 while (1);
 }
 Serial.println(F("BME280 ініціалізовано"));

 // Ініціалізація RTC
 if (!rtc.begin()) {
 Serial.println(F("Помилка ініціалізації RTC!"));
 lcd.setCursor(0, 1);
 lcd.print("Помилка RTC!");
 while (1);
 }

 // Встановлення часу RTC, якщо його втрачено (закоментувати після першого запуску)
```

```
 // rtc.adjust(DateTime(F(DATE), F(TIME)));

 Serial.println(F("RTC ініціалізовано"));

 // Зчитування початкового значення тиску
 pressure = bme.readPressure() / 100.0F;
 previousPressure = pressure;

 // Повідомлення про успішну ініціалізацію
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Метеостанція");
 lcd.setCursor(0, 1);
 lcd.print("готова!");
 delay(2000);

 Serial.println(F("Система готова до роботи!"));
}

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

 // Оновлення метеоданих з інтервалом updateInterval
 if (currentMillis - lastUpdateTime >= updateInterval) {
 lastUpdateTime = currentMillis;
 updateMeasurements();
 updateForecast();
 printDebugInfo();
 }

 // Оновлення дисплея з інтервалом displayInterval
 if (currentMillis - lastDisplayTime >= displayInterval) {
 lastDisplayTime = currentMillis;
 updateDisplay();
 }
}

// Функція зчитування даних з датчиків
```

```
void updateMeasurements() {
 // Зчитування даних з DHT22
 float dhtTemp = dht.readTemperature();
 float dhtHum = dht.readHumidity();

 // Перевірка на помилки зчитування DHT22
 if (isnan(dhtTemp) || isnan(dhtHum)) {
 Serial.println(F("Помилка зчитування з DHT22!"));
 } else {
 temperature = dhtTemp;
 humidity = dhtHum;
 }

 // Зчитування даних з BME280
 float bmeTemp = bme.readTemperature();
 float bmeHum = bme.readHumidity();
 float bmePres = bme.readPressure() / 100.0F;

 // Використання даних BME280, якщо DHT22 дав помилку
 if (isnan(dhtTemp)) temperature = bmeTemp;
 if (isnan(dhtHum)) humidity = bmeHum;

 // Оновлення значення тиску
 previousPressure = pressure;
 pressure = bmePres;
}
// Функція прогнозування погоди на основі зміни тиску
void updateForecast() {
 float pressureDiff = pressure - previousPressure;

 if (pressureDiff > 1.0) {
 strcpy(forecast, "Покращення");
 } else if (pressureDiff < -1.0) {
 strcpy(forecast, "Погіршення");
 } else if (pressureDiff > 0.5) {
 strcpy(forecast, "Прояснення");
 } else if (pressureDiff < -0.5) {
 strcpy(forecast, "Дощ");
```

```
 } else {
 strcpy(forecast, "Стабільно");
 }
}

// Функція виведення відлагоджувальної інформації в серійний порт
void printDebugInfo() {
 DateTime now = rtc.now();

 Serial.println(F("----------------------------"));
 Serial.print(F("Дата і час: "));
 Serial.print(now.day());
 Serial.print(F("/"));
 Serial.print(now.month());
 Serial.print(F("/"));
 Serial.print(now.year());
 Serial.print(F(" "));
 Serial.print(now.hour());
 Serial.print(F(":"));
 Serial.print(now.minute());
 Serial.print(F(":"));
 Serial.println(now.second());

 Serial.print(F("Температура: "));
 Serial.print(temperature);
 Serial.println(F(" °C"));

 Serial.print(F("Вологість: "));
 Serial.print(humidity);
 Serial.println(F(" %"));

 Serial.print(F("Тиск: "));
 Serial.print(pressure);
 Serial.println(F(" hPa"));

 Serial.print(F("Прогноз: "));
 Serial.println(forecast);
 Serial.println(F("----------------------------"));
```

```
}

// Функція оновлення дисплея з циклічною зміною відображення
void updateDisplay() {
 lcd.clear();
 DateTime now = rtc.now();

 switch (displayState) {
 case 0:
 // Відображення температури та вологості
 lcd.setCursor(0, 0);
 lcd.print("Темп.: ");
 lcd.print(temperature, 1);
 lcd.print((char)223); // Символ градуса
 lcd.print("C");

 lcd.setCursor(0, 1);
 lcd.print("Вологість: ");
 lcd.print(humidity, 0);
 lcd.print("%");
 break;

 case 1:
 // Відображення тиску та прогнозу
 lcd.setCursor(0, 0);
 lcd.print("Тиск: ");
 lcd.print(pressure, 1);
 lcd.print(" hPa");

 lcd.setCursor(0, 1);
 lcd.print("Прогноз: ");
 lcd.print(forecast);
 break;

 case 2:
 // Відображення дати та часу
 lcd.setCursor(0, 0);
 lcd.print(now.day());
```

```
 lcd.print("/");
 lcd.print(now.month());
 lcd.print("/");
 lcd.print(now.year());

 lcd.setCursor(0, 1);
 if (now.hour() < 10) lcd.print("0");
 lcd.print(now.hour());
 lcd.print(":");
 if (now.minute() < 10) lcd.print("0");
 lcd.print(now.minute());
 lcd.print(":");
 if (now.second() < 10) lcd.print("0");
 lcd.print(now.second());
 break;

 default:
 displayState = 0;
 return;
 }

 // Циклічна зміна стану дисплея
 displayState = (displayState + 1) % 3;
}
// Додатковий код для підключення до Wi-Fi і відправки даних на ThingSpeak
// Це доповнення до основного скетчу
#include <SoftwareSerial.h>

// Налаштування SoftwareSerial для ESP8266
SoftwareSerial esp8266(10, 11); // RX, TX

// Налаштування Wi-Fi
const char* ssid = "YOUR_WIFI_NETWORK"; // Ім'я вашої Wi-Fi мережі
const char* password = "YOUR_WIFI_PASSWORD"; // Пароль від Wi-Fi мережі
const char* thingSpeakApiKey = "YOUR_API_KEY"; // API ключ ThingSpeak

// Таймер для відправки даних
unsigned long lastSendTime = 0;
```

```
const long sendInterval = 300000; // Інтервал відправки даних (5 хвилин)

void setupWiFi() {
 // Налаштування комунікації з ESP8266
 esp8266.begin(9600);
 delay(1000);

 Serial.println(F("Налаштування ESP8266..."));

 // Скидання модуля ESP8266
 sendCommand("AT+RST", 5000);

 // Налаштування режиму роботи
 sendCommand("AT+CWMODE=1", 2000);

 // Підключення до Wi-Fi
 String cmd = "AT+CWJAP=\"";
 cmd += ssid;
 cmd += "\",\"";
 cmd += password;
 cmd += "\"";
 sendCommand(cmd, 10000);

 Serial.println(F("ESP8266 налаштовано"));
}

// Функція відправки команд на ESP8266
String sendCommand(String command, const int timeout) {
 String response = "";
 esp8266.println(command);

 long int time = millis();
 while ((time + timeout) > millis()) {
 while (esp8266.available()) {
 char c = esp8266.read();
 response += c;
 }
 }
```

```
 Serial.print(command);
 Serial.print(F(" -> "));
 Serial.println(response);

 return response;
}

// Функція відправки даних на ThingSpeak
void sendDataToThingSpeak() {
 // Формування тіла запиту
 String httpRequest = "GET /update?api_key=";
 httpRequest += thingSpeakApiKey;
 httpRequest += "&field1=";
 httpRequest += String(temperature);
 httpRequest += "&field2=";
 httpRequest += String(humidity);
 httpRequest += "&field3=";
 httpRequest += String(pressure);
 httpRequest += " HTTP/1.1\r\nHost: api.thingspeak.com\r\nConnection: close\r\n\r\n";

 // Підключення до сервера ThingSpeak
 sendCommand("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80", 10000);

 // Відправка запиту
 sendCommand("AT+CIPSEND=" + String(httpRequest.length()), 2000);
 sendCommand(httpRequest, 5000);

 // Закриття з'єднання
 sendCommand("AT+CIPCLOSE", 2000);

 Serial.println(F("Дані успішно відправлені на ThingSpeak"));
}

// Функція, що викликається в основному циклі loop()
void processWiFi() {
 unsigned long currentMillis = millis();
```

```
 // Відправка даних на ThingSpeak з інтервалом sendInterval
 if (currentMillis - lastSendTime >= sendInterval) {
 lastSendTime = currentMillis;
 sendDataToThingSpeak();
 }
}
```

Додаток Г (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Розробка програмного забезпечення для автоматизації процесів на метеостанції»

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ: кафедра АІІТ, ФІІТА. АКІТР-23мс
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 9,5 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☐ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Олег БІСІКАЛО, зав. кафедри АІІТ _____
(прізвище, ініціали, посада) (підпис)

Любов ЯНЧУК, секретар кафедри АІІТ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Овчинников К. В.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Гармаш В. В.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Палаш Б. Р.
(підпис) (прізвище, ініціали)