Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 20__ р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Рисований М.О.
«____» _____ 20 р.

Керівник проекту
кандидату технічних наук, доцент
_____ Босько В. В.
«____» _____ 20 р.
Рецензент _____
_____

м. Кропивницький

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *123 Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« __ » _____ 20__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Рисованому Максиму Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією*

керівник роботи *Босько Віктор Васильович, к-д техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу *№ 131-02 від 01.04 .2024 року*

2. Строк подання студентом роботи до захисту *01.04.2024 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмного забезпечення для системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією на основі платформи Arduino UNO.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

6. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	19.05.2024 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

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

АНОТАЦІЯ

Рисований М.О. Програмне забезпечення системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

У даній кваліфікаційній бакалаврській роботі розроблено програмне забезпечення системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією.

Метою роботи є розробка програмного забезпечення для системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією на основі платформи Arduino UNO.

Результат роботи – програмна реалізація системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією.

В процесі роботи над реалізацією системи виконано дослідження існуючих методів, алгоритмів та програмних засобів. Розроблено та реалізовано власне програмне забезпечення, здійснено опис всіх його компонентів.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено на мові програмування Arduino.

Ключові слова: інтернет речей, моніторинг, екстраполяція, метеостанція, Arduino.

ABSTRACT

Rysovanyi M.O. Software of the system for remote monitoring of microclimatic changes with their extrapolation. 123 Computer Engineering. Central Ukraine National Technical University. Kropyvnytskyi. 2024.

This bachelor's qualification work develops software for a remote monitoring system of microclimate changes with their extrapolation.

The aim of the work is to develop software for a remote monitoring system of microclimate changes with their extrapolation based on the Arduino UNO platform.

The result of the work is the software implementation of a remote monitoring system of microclimate changes with their extrapolation.

In the process of working on the system implementation, research on existing methods, algorithms, and software tools was conducted. The software was developed and implemented, and a description of all its components was provided.

A user-friendly interface is developed. The instructions for working with the software are provided.

The program can be used on an IBM PC running Windows 10/11.

The program is developed in the Arduino programming language.

Keywords: Internet of Things, monitoring, extrapolation, weather station, Arduino.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БР	–	Бакалаврська робота
IoT	–	Інтернет речей (Internet of Things)
OLED	–	Органічний світлодіодний дисплей
IDE	–	Інтегроване середовище розробки
і т.д.	–	і так далі
DIY	–	Зроби сам
I2C	–	Шина двоїчної адресації
LCD	–	Рідкокристалічний дисплей
ПЗ	–	Програмне забезпечення
ОС	–	Операційна система
ПЕОМ	–	Персональна обчислювальна машина
ПК	–	Персональний комп’ютер
ПХ	–	Порти хосту
ОС	–	Операційна система
ТЗ	–	Технічне завдання
хв.	–	хвилина
SPI		Інтерфейс перифірерійних пристроїв
ICS	–	Internet Component Suite

ВСТУП

Актуальність теми. В сучасному світі, де зростає значення ефективного використання ресурсів та збереження навколишнього середовища, системи віддаленого моніторингу мікрокліматичних змін стають все більш актуальними та необхідними. Зокрема, розвиток технологій у сфері інтернету речей (IoT) відкриває широкі можливості для створення ефективних систем моніторингу, які дозволяють в реальному часі отримувати дані про кліматичні умови в різних просторах та пристосовувати дії відповідно до цих даних.

Україна, як країна з різноманітним кліматом та великим сільськогосподарським потенціалом, особливо зацікавлена у вдосконаленні систем моніторингу мікроклімату для забезпечення оптимальних умов для сільськогосподарських культур, збереження врожаю та підвищення його якості. Однак існуючі рішення не завжди відповідають сучасним вимогам ефективності, точності та доступності.

Метою роботи є розробка програмного забезпечення для системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією на основі платформи Arduino UNO.

Основними завданнями, що необхідно вирішити для досягнення поставленої мети, включають наступне:

озробка апаратної складової системи, включаючи підключення та інтеграцію різних датчиків (вологості, температури, барометра), датчика точного часу та OLED дисплея для візуалізації даних.

озробка програмного забезпечення для збору, обробки та відображення отриманих даних про мікроклімат, з використанням мови програмування

роведення експериментів для перевірки працездатності та точності системи, а також оцінка її ефективності у реальних умовах.

Отримані результати мають практичне значення для широкого кола застосувань. Зокрема, система може бути використана для моніторингу мікроклімату в сільськогосподарських господарствах, оранжереях, складах зберігання продуктів харчування та інших об'єктах, де важливо забезпечити оптимальні умови для збереження та розвитку рослинного світу. Крім того, отримані дані можуть бути використані для наукових досліджень у галузі агрокліматології, екології та інших споріднених областях.

Отже, дана робота відіграє важливу роль у подальшому розвитку відповідної галузі науки та виробництва. Крім того, вона може сприяти покращенню якості життя населення та збереженню природних ресурсів.

РИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

ризначення системи

Система, що розробляється, має на меті забезпечити віддалений моніторинг мікрокліматичних параметрів в різних просторах та навколишніх середовищах. Головним призначенням системи є збір, обробка та передача даних про температуру, вологість та атмосферний тиск за допомогою встановлених датчиків. Крім того, система надає можливість відображення цих даних на OLED дисплеї для зручності користувача.

Основні функції системи включають:

бір та реєстрацію даних про мікрокліматичні параметри, що дозволяє в реальному часі контролювати та відстежувати зміни в середовищі.

наліз та обробку отриманих даних.

ідображення інформації на дисплеї для візуалізації даних та надання користувачеві зрозумілої інформації про поточний стан середовища.

огування даних у окремий файл з метою відстеження змін у середовищі за певний час, що дозволяє зберігати історію змін для подальшого аналізу та використання.

бласть застосування

Розроблена система може бути успішно використана в різних областях, де важливо контролювати та забезпечувати оптимальні мікрокліматичні умови. Деякі з можливих областей застосування включають:

ільське господарство: Система може бути використана для моніторингу та контролю вирощування сільськогосподарських культур в оранжереях,

теплицях або на відкритому ґрунті, забезпечуючи оптимальні умови для зростання рослин та збереження врожаю.

ромислові об'єкти: Система може бути використана на промислових об'єктах, де необхідно контролювати температуру та вологість, таких як склади зберігання продуктів харчування, склади для зберігання медикаментів або технологічні приміщення.

фісні та житлові приміщення: Система може бути використана для контролю мікроклімату в офісах, будинках та інших приміщеннях, забезпечуючи комфортні умови для праці та проживання.

аукові дослідження: Система може бути використана для проведення наукових досліджень у галузі агрокліматології, екології та інших споріднених областях, де важливо вивчати вплив мікроклімату на різні процеси та явища.

ндустрія**:** Система може бути використана в промислових установках для контролю мікроклімату в приміщеннях, де важливо забезпечити оптимальні умови для виробничих процесів та збереження обладнання.

хорона довкілля: Система може бути використана для моніторингу середовища в природних резерватах, парках та інших природних об'єктах для вивчення впливу людської діяльності на навколишнє середовище та вжиття заходів щодо його збереження.

удівництво: Система може бути використана в будівельній галузі для моніторингу мікроклімату на будівельних майданчиках та в приміщеннях під час будівництва, що дозволяє забезпечити оптимальні умови для робітників та ефективне використання будівельних матеріалів.

едична галузь: може використовуватися для контролю та збереження оптимальних умов у медичних закладах, лабораторіях та аптеках. Це особливо важливо для збереження лікарських засобів та медичного обладнання, які можуть бути чутливими до змін у мікрокліматі.

світня галузь: може бути використана для проведення практичних занять з різних дисциплін в галузі екології або агрокліматології.

Розроблена система віддаленого моніторингу мікрокліматичних змін має широкі можливості застосування в різних галузях діяльності. Вона забезпечує контроль та управління мікрокліматом у різних умовах, починаючи від промислових об'єктів і закінчуючи приватними оселями. Розроблена система може бути використана в сільському господарстві для оптимізації умов вирощування рослин, в медичних закладах для збереження лікарських засобів та в освітніх установах для проведення навчальних та наукових досліджень. Таким чином, вона відкриває нові можливості для контролю мікроклімату та забезпечення комфорту та безпеки в різних сферах життя та діяльності.

ЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

Огляд існуючих систем моніторингу мікрокліматичних змін

Перед тим як розглянути конкретні системи та технології, важливо зазначити, що на ринку існує ряд існуючих систем моніторингу мікроклімату та погодних умов. Кожна з цих систем має свої переваги та недоліки, які варто враховувати при обговоренні їхнього застосування. Відомо, що деякі системи можуть бути високою вартістю, інші можуть мати обмежений функціонал, а деякі можуть не завжди забезпечувати стабільну та надійну роботу. Отже, проведення аналізу існуючих систем допоможе краще зрозуміти їхні можливості та обмеження, а також визначити причини, які можуть вплинути на необхідність розробки нової системи віддаленого моніторингу мікроклімату.

Існує велика кількість вже готових метеостанції, які були розроблені компаніями, які на цьому спеціалізуються. Однією з найвідоміших та широко використовуваних є метеостанція Davis Vantage Pro2 [1]. Ця метеостанція відома своєю надійністю та точністю вимірювань. Вона має широкий набір датчиків, включаючи температуру, вологість, атмосферний тиск та опади.

Основні області застосування Davis Vantage Pro2:

етеорологічні дослідження: Метеостанція використовується для проведення наукових досліджень у галузі метеорології. Вона дозволяє отримувати точні дані про погодні умови та мікроклімат, що необхідно для аналізу кліматичних тенденцій та прогнозування погоди.

ільське господарство: використовується у сільському господарстві для моніторингу погодних умов та визначення оптимального часу для сівби, поливу, обробки рослин тощо. Це допомагає фермерам оптимізувати виробничі процеси та підвищити врожайність.

удівництво та будівельна індустрія: Davis Vantage Pro2 використовується для контролю та моніторингу погодних умов на будівельних майданчиках. Вона дозволяє планувати та керувати будівельними процесами з урахуванням погодних факторів, таких як опади, вітер, температура тощо.

ромислові застосування: Метеостанція використовується у промисловості для моніторингу погодних умов, що може бути важливим у виробництві та транспортуванні товарів, а також для безпеки працівників у відкритих промислових зонах.

собисте використання: Деякі користувачі також встановлюють метеостанцію Davis Vantage Pro2 у своїх домівках або садах для моніторингу погоди та мікроклімату для особистого використання та розваг.

Метеостанція Davis Vantage Pro2 також має можливість бездротового зв'язку та підтримує роботу з комп'ютером або мобільними пристроями через спеціалізоване програмне забезпечення.

2 - це потужний інструмент для моніторингу погоди та мікроклімату, проте важливо враховувати його вартість. На момент написання цієї роботи, найдешевша версія метеостанції Davis Vantage Pro2 коштує приблизно 595 доларів. Ця ціна може бути важливим фактором для багатьох користувачів, які розглядають різні варіанти метеостанцій для своїх потреб.

Варто розглянути одну з найпопулярніших метеостанцій, яка має назву

популярна метеостанція завдяки простоті встановлення та використання. Розглянемо детально її особливості, плюси та мінуси, а також причини використання та її доступність.

Особливості метеостанції AcuRite 01036M:

ростота в установці: AcuRite 01036M легко встановлюється і налаштовується, користувачі часто звертають на це увагу.

ездротовий зв'язок: Завдяки технології бездротового зв'язку, ця метеостанція може бути підключена до монітора або базової станції за допомогою

радіочастот, що спрощує процес і дозволяє розташувати датчики в будь-якому місці.

оніторинг через смартфон: Деякі моделі AcuRite 01036M мають можливість підключення до смартфону через спеціальний додаток, що дозволяє користувачам переглядати дані про погоду в режимі реального часу з будь-якої точки з'єднання.

Переваги:

оступність: Однією з основних переваг AcuRite 01036M є його доступність. В порівнянні з іншими метеостанціями на ринку, вона зазвичай має більш прийнятну вартість.

ростота використання: Інтуїтивний і легкий у використанні інтерфейс робить AcuRite 01036M ідеальним варіантом для користувачів.

Недоліки:

бмежений функціонал: У порівнянні з більш дорогими та продвинутими моделями, AcuRite 01036M може мати обмежений функціонал.

е завжди стабільний зв'язок: Деякі користувачі повідомляли про нестабільність зв'язку між датчиками та монітором, особливо в умовах великих відстаней.

Загалом AcuRite 01036M є непоганим варіантом метеостанції для особистого використання. Таку модель можна використовувати у своїх домівках, садах і т.д. Вартість такої станції на момент написання роботи приблизно 200 долларів.

Наступною альтернативою є метеостанція **La Crosse Technology V40-**

є добре відомою та широко використовуваною метеостанцією, має багато функціоналу та є досить надійною.

Особливості:

- Багатофункціональність: La Crosse Technology V40-PRO-INT може вимірювати різні параметри погоди, включаючи температуру, вологість, вітер,

атмосферний тиск та дощ. Це робить її ідеальним інструментом для повного моніторингу погоди.

CD-дисплей: Метеостанція оснащена великим LCD-дисплеєм, який зручно відображає дані про погоду та іншу інформацію для користувача.

ездротовий зв'язок: Завдяки технології бездротового зв'язку, La Crosse Technology V40-PRO-INT може безперервно передавати дані між датчиками та монітором.

Переваги:

- Надійність: La Crosse Technology є надійною та стабільнільною метеостанції, користувачі відзначають це у відгуках.
- Зручне керування: Інтуїтивний і легкий у використанні інтерфейс робить La Crosse Technology V40-PRO-INT дуже зручною у використанні.
- Доступність: В порівнянні з іншими моделями метеостанцій, La Crosse Technology V40-PRO-INT зазвичай має прийнятну вартість, що робить її доступною для більш широкого кола користувачів.

Недоліки:

- Обмежений функціонал: У порівнянні з деякими дорожчими моделями, La Crosse Technology V40-PRO-INT може мати обмежений функціонал.
- Проблеми зі зв’язком: Деякі користувачі повідомляли про можливі проблеми зі стабільністю зв'язку між датчиками та монітором.

La Crosse Technology V40-PRO-INT є надійним інструментом для моніторингу погоди та мікроклімату з багатофункціональними можливостями та доступною ціною.

Альтернативою цієї станції є досить схожа станція - **Oregon Scientific** Ця метеостанція має датчики вологості, температури, вітру, атмосферного тиску та дощу. Також має дисплей, зовнішні датчики для більш точних вимірювань у природі.

З переваг можна зазначити:

- Доступність і вартість: Oregon Scientific WMR86 є досить доступною моделлю метеостанції, яка пропонує широкий функціонал за прийнятну ціну. ручне використання: Простий інтерфейс та зручна конструкція може бути значним плюсом для звичайного користувача.

Недоліки:

- Обмежена точність: У порівнянні з деякими більш дорогими моделями, точність вимірювань Oregon Scientific WMR86 може бути обмеженою.

естійкий зв'язок**:** Така сама проблема, що у більшості попередніх метеостанції. Причиною цієї проблеми може бути погана якість датчиків або ж дуже велика відстань між зовнішним датчиком та основною станцією. На офіційному сайті зазначається що дистанція використання сягає 100 метрів.

Більшість метеостанцій мають живлення 3AA батарейками, які у свою чергу хоч і є досить довгоживучими для такої нескладної системи, але все ж, як і будь-яка річь може бути неякісною або просто батарейка вийшла з ладу. Багато користувачів пишуть відгуки зазначаючи про нестійкий зв’язок або ненадійність пристроїв, але у свою чергу не перевіряють такі очевидні речі. Звичайно, деякі метеостанції мають очевидні проблеми зі зв’язком, просто треба завжди орієнтуватись на похибку у відгуках користувачів та зазначених характеристик на сайті.

Багато з метеостанцій має досить примітивний дизайн, але щоб виділятися на ринку для звичайного користувача метеостанція Netatmo розробила більш футуристичний вигляд та в більшості випадків через це має свого покупця.

Рисунок 2.1 – Зовнішний вигляд метеостанції Netatmo[5]

Особливості Netatmo:

ідключення до Інтернету: Netatmo Weather Station може бути підключена до Інтернету, що дозволяє користувачам віддалено отримувати дані про погоду через смартфон або комп'ютер.

нтелектуальний аналіз: Ця метеостанція має можливість інтелектуального аналізу погодних умов та навіть пропонує рекомендації щодо дій на основі зібраних даних.

тильний дизайн: Netatmo Weather Station має сучасний та стильний дизайн, який добре вписується в будь-який інтер'єр.

Переваги метеостанції:

- Це одна з перших метеостанцій, яка пропонує синхронізацію зі смартфоном, що робить процес моніторингу погоди ще зручнішим.

- Netatmo Weather Station може інтегруватися з різними погодними службами, що дозволяє користувачам отримувати більше інформації про погодні умови.

- Метеостанція Netatmo Weather Station використовується для вимірювання різних параметрів навколишнього середовища, що дозволяє користувачам бути більш екологічно свідомими.

Недоліки метеостанції:

- Висока вартість: Порівняно з іншими моделями метеостанцій, Netatmo Weather Station може бути дещо дорожчою, що робить її менш доступною для деяких користувачів.

алежність від Інтернету: Оскільки метеостанція підключається до Інтернету, вона може бути менш ефективною в разі відсутності Інтернет-з'єднання.

N

e

t Датчик звуку є пристроєм, який призначений для перетворення a звукових коливань у електричні сигнали. Основна функція є зафіксувати t акустичні події в оточуючому середовищі і перетворити їх в електричні сигнали. Датчики звуку в метеостанціях використовуються для вимірювання рівню шуму в навколишньому середовищі. Вони також широко використовуються у вимірювальних системах безпеки, аудіозаписувальних пристроях, системах розпізнавання голосу та багатьох інших пристроях, де необхідно акустичне взаємодіяти з оточуючим середовищем.

t Багато з зазначених метеостанцій використовуються в побуті звичайних користувачів, але варто зауважити, що є й такі метеостанції, які використовуються професіоналами. Основна відмінність дорогих метеостанцій від дешевих в якості датчиків та габаритності. Для більш професійного користування повинні бути найкращі компоненти, тому що на відміну від побутового використання має бути точна інформація, так як від цієї інформації бути залежати робота промислових та аграрних підприємств,

t

діяльність метеорологічних служб, а також безпека в авіації та мореплавстві. Дорогі метеостанції, як правило, мають більш точні датчики, які забезпечують надійні вимірювання погодних параметрів навіть у найскладніших умовах. Вони також можуть мати більше можливостей для підключення до додаткових сенсорів та розширених систем збору даних.

Професійні метеостанції також зазвичай мають більшу габаритність, оскільки їх потрібно розміщувати у відкритих просторах для отримання максимально точних даних. Вони можуть мати такі додаткові функції, як вбудовані системи реєстрації даних, програмне забезпечення для аналізу даних та засоби дистанційного моніторингу.

Одна з таких метеостанція, яка в 99 відсотках випадках використовуються професіонально це **Campbell Scientific CR1000.** [6]. 1000 від Campbell Scientific має модульну конструкцію, що дозволяє користувачам налаштовувати систему відповідно до своїх потреб. CR1000 забезпечує високу точність вимірювань завдяки використанню датчиків високої якості та технології автоматичної калібрування. Завдяки вбудованому засобу збору та передачі даних, користувачі можуть отримувати доступ до даних про погоду в реальному часі через Інтернет з будь-якого місця, де є Інтернет-з'єднання, тобто метеостанція має дистанційний доступ. Також, одна з головних переваг на яку звертають увагу це стійкість до середовища. Метеостанція має міцний корпус та стійкість до екстремальних погодних умов, що дозволяє використовувати її в різних кліматичних умовах та навіть у важкодоступних місцях, таких як гірські райони або пустелі. Campbell 1000 застосовувалася у різних наукових дослідженнях та дослідницьких проектах.

C

a

m

p

b

e

l

як наукових досліджень, так і промислових застосувань, забезпечуючи високу точність та надійність даних. Наприклад, крім метеостанцій, Campbell також пропонує датчики для моніторингу рівня води, швидкості та напрямку вітру, сонячної радіації, якості ґрунту. Одна з таких моделей LW110 є станцією моніторингу електричного поля і попереджає населення про небезпеку появи блискавки та грози. Особливість такої станції в тому що попередження засновані на вимірах електричного поля, а не попередніх ударів, тому система здатна виявляти небезпеку виникнення блискавки зазделегідь. Станція широко застосовується у США, Великій Британії, Франції для попередження населення про подальшу небезпеку, і через високу точність вимірювань люди мають достатньо часу для організації своєї власної безпеки.

Також варто привести у приклад метеостанції, які використовуються у навчальних закладах та любителями метеорології. Такі станції відрізнятися тим, що мають різноманітні датчики, щоб навчити людей користуватися метеорологічним обладнанням і розуміти погодні явища. Наприклад, метеостанції, призначені для навчальних цілей, часто оснащені додатковими датчиками, які вимірюють різні параметри атмосфери, такі як рівень UV-випромінювання, радіація, а також здатність вимірювати вологість та температуру ґрунту. Ці метеостанції можуть бути використані для навчання студентів у школах та університетах, а також для організації метеорологічних гуртків для любителів. Вони надають можливість вивчати погоду та кліматичні процеси у навчальних цілях, а також сприяють вихованню інтересу до науки та збільшенню метеорологічної грамотності серед широкого загалу. Одна з

т

а

к

и

х Особливості Ambient Weather WS-2902C:

ездротове підключення: WS-2902C використовує бездротовий зв'язок для

м

е

збирання та передачі даних, що дозволяє легко встановлювати та налаштовувати метеостанцію без необхідності проводів.

ирокий спектр функцій: Ця метеостанція вимірює температуру, вологість, тиск повітря, швидкість та напрямок вітру, а також кількість опадів. Всі ці дані можуть бути відображені на вбудованому дисплеї або передані через Інтернет для подальшого аналізу.

оступ до даних через Інтернет: За допомогою підключення до Інтернету, користувачі можуть отримувати доступ до даних про погоду у реальному часі з будь-якого пристрою, що підключений до мережі, наприклад, смартфону або комп'ютера.

егка установка та використання: Ambient Weather WS-2902C поставляється з докладною інструкцією з налаштування та використання, що робить її дружньою до користувача навіть для початківців у метеорології.

Переваги:

етеостанція WS-2902C вимірює широкий спектр погодних параметрів, включаючи температуру, вологість, тиск повітря, швидкість та напрямок вітру, а також кількість опадів. Це дозволяє користувачам отримувати повну картину погодних умов у своєму регіоні.

S-2902C має інтуїтивно зрозуміле і просте в управлінні інтерфейс, що робить налаштування та користування метеостанцією легким та зручним.

етеостанція може підключатися до Інтернету, що дозволяє користувачам отримувати дані про погоду у реальному часі з будь-якого місця, де є Інтернет-з'єднання.

є надійною та довговічною, що робить її привабливим вибором для домашнього використання.

Недоліки:

- Відсутність підтримки налаштування за допомогою мобільного додатка: На відміну від деяких інших метеостанцій, WS-2902C не має мобільного додатка для зручного налаштування та управління.

- Обмежений дальній зв'язок: Деякі користувачі відзначають, що бездротовий зв'язок між сенсорами та базовою станцією може бути обмеженим в залежності від умов навколишнього середовища.

- Обмежені можливості аналізу даних: Деякі користувачі вважають, що програмне забезпечення, яке постачається з метеостанцією, має обмежені можливості для аналізу та візуалізації даних.

Обгрунтування необхідності розробки системи за темою випускної кваліфікаційної роботи з урахуванням проведеного аналізу виявляє низку вагомих аргументів щодо актуальності та цілеспрямованості цього проекту.

Перш за все, в аналізі було виявлено, що більшість готових метеостанцій на ринку мають високу якість та функціональність, але часто вони не надають користувачам можливості змінювати або модифікувати компоненти залежно від їх потреб. Це обмеження може бути неприйнятним для деяких користувачів, які мають специфічні вимоги щодо функціональності та хочуть мати можливість розширити або модифікувати свою метеостанцію в майбутньому. Крім того, важливо зазначити, що багато з вище згаданих готових метеостанцій не надають користувачам можливості перепрограмування або модифікації за допомогою програмування. Вони поставляються на ринок як готові пристрої з вбудованим програмним забезпеченням, яке не підлягає змінам або налаштуванням користувачем. Це може бути обмеженням для тих, хто бажає налаштувати метеостанцію під свої потреби або розробити власні програмні функції для збору та аналізу даних.

Розробка метеостанції на базі Arduino пропонує гнучкий та модульний підхід до побудови метеорологічного обладнання. Завдяки використанню Arduino та відкритого програмного забезпечення, користувачі отримують можливість легко модифікувати та адаптувати систему до своїх потреб,

вносячи зміни в апаратне забезпечення та програмне забезпечення згідно зі своїми вимогами. Це особливо важливо для використання метеостанції в освітніх закладах, де студенти можуть вчитися експериментувати з різними датчиками та модулями, розширюючи свої знання в галузі електроніки та програмування. Розробка власної метеостанції на базі Arduino може бути важливим кроком у розвитку дослідницьких навичок та інтересу до науки в галузі метеорології. Цей проект надає можливість студентам та дослідникам займатися практичними дослідженнями та експериментами у галузі метеорології, що сприяє поглибленню їх розуміння погодних процесів та кліматичних змін.

Також важливо відзначити, що на сучасному ринку практично немає готових метеостанцій, які можна було б розглядати як "умовний пазл", тобто систему, яку можна складати з окремих компонентів, адаптуючи її під власні потреби та вимоги. Багато з готових метеостанцій продаються як вбудовані рішення з фіксованим набором функцій та компонентів, і користувачі мають обмежені можливості змінювати або розширювати їх. У контексті вищезазначеного, розробка метеостанції на базі Arduino надає значну перевагу, оскільки мікроконтролер Arduino працює на відкритому програмному забезпеченні та має широкі можливості для перепрограмування та налаштування. Користувачі можуть створювати власні програмні рішення за допомогою Arduino IDE (Integrated Development Environment) на мові програмування C/C++, що відкриває широкі можливості для розвитку індивідуальних функціональних можливостей та адаптації метеостанції до своїх потреб. Такий підхід сприяє розширенню функціональності метеостанції та її потенційному використанню в різних сферах досліджень та застосування.

Отже, на відміну від готових метеостанцій, розробка власної системи на базі Arduino пропонує унікальні можливості для користувачів у плані гнучкості, адаптивності та навчання, що робить цей проект важливим та актуальним для використання в освітніх та дослідницьких цілях.

Обґрунтування вибору засобів для побудови системи та мови програмування

Почнемо з основного компонента Arduino UNO. Arduino Uno є одним з найпопулярніших та найширше використовуваних мікроконтролерів для DIY проектів. Arduino Uno – пристрій на основі мікроконтролера ATmega328. Основні характистики пристрою:

- Цифрові входи/виходи: Плата має 14 цифрових входів/виходів, з яких 6 можуть використовуватися як виходи з широтно-імпульсною модуляцією (PWM).

налогові входи: Є 6 аналогових входів для зчитування аналогових сигналів.

- Резонатор: Вбудований керамічний резонатор з частотою 16 МГц.

SB-підключення: Має USB-порт для з’єднання з комп’ютером.

оз’єм живлення: Може живитися як від USB, так і від зовнішнього джерела живлення (наприклад, батарейки).

CSP-роз’єм: Для програмування мікроконтролера через SPI.

нопка скидання: Для скидання мікроконтролера.

фера застосування: Arduino Uno використовується для розробки різноманітних електронних пристроїв. Це може бути від простих світлодіодних маячків до складних систем автоматизації.

обоча напруга 5В.

апруга живлення 7-12В.

RAM 2КБ.

EPROM 1КБ.

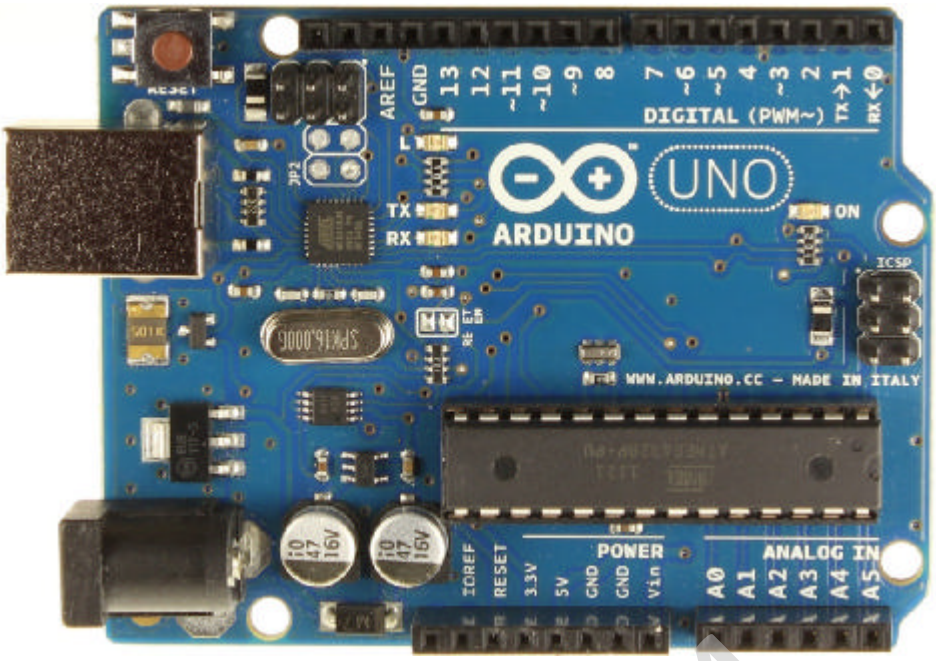

Рисунок 2.2 – Arduino UNO[42] з видом з переду

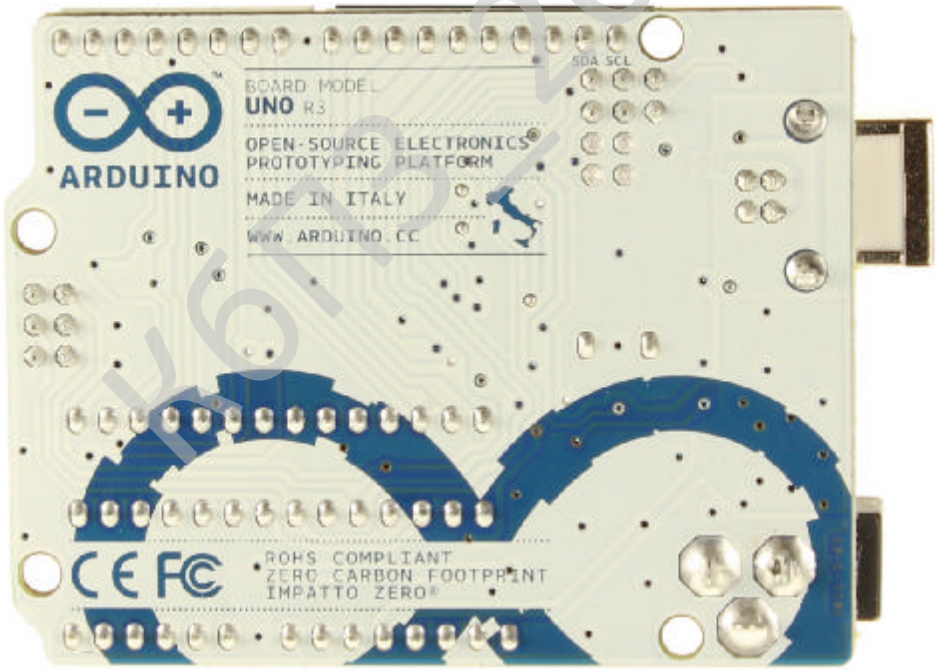

Рисунок 2.3 – Arduino UNO[42] з видом ззаду

Обираючи Arduino Uno для розробки метеостанції, враховано декілька важливих факторів, які зробили цю плату оптимальним вибором для проекту.

Перш за все, Arduino Uno є добре відомою та широко використовуваною платою, що забезпечує простоту у використанні та розробці. Її популярність означає наявність великої кількості документації, прикладів коду та підтримки спільноти, що полегшує процес розробки. Крім того, Arduino Uno пропонує значну кількість цифрових та аналогових входів/виходів, що відкриває можливості для підключення різноманітних сенсорів та модулів для вимірювання погодних параметрів. Це робить Arduino Uno досить гнучкою для наших потреб. Проте, важливо відзначити, що Arduino Uno не має вбудованого модуля Wi-Fi, що може бути недоліком для деяких застосувань, зокрема для передачі даних до хмарних сервісів або моніторингу з використанням інтернету.

У проекті буде використано декілька датчиків для вимірювання температури та вологості, одним з найпопулярніших датчиків є DHT-11. [8]

Представляє собою електронний пристрій, що вимірює температуру та вологість у цифровому форматі, і забезпечує калібрування цифрового сигналу на виході. Він складається з двох основних компонентів - ємнісного датчика вологості та термістора. Додатково, в датчику знаходиться аналого-цифровий перетворювач (АЦП), який перетворює аналогові значення температури та вологості у цифровий формат.

Характеристики DHT-11:

атчик DHT-11 здатен вимірювати температуру від 0°C до 50°C і відносну вологість повітря від 20% до 90%.

очність вимірювання температури становить ±2°C, а вимірювання вологості -

- Інтерфейс підключення: Датчик DHT-11 підключається до мікроконтролера за допомогою цифрового входу/виходу. Він використовує простий протокол однобітового сигналу для передачі даних.

З переваг можна зазначити низьку вартість датчика, легка інтеграція та використання з мікроконтролером Arduino, довга та надійна робота у

ширикому діапазоні температур. Недоліком може бути обмежена точність вимірювань, в порівнянні з більш дорогими альтернативами.

Наступним датчиком буде розглянуто GY-21 [9], це також датчик вологості та температури, але коштує дорожче та більш точно визначає вимірюванні дані.

На кристалі розташовані компоненти для аналого-цифрового перетворення, обробки сигналу та використання інтерфейсу I2C. Ця технологія є предметом патентування та застосовується у промисловості. Вбудований CMOS датчик і система управління характеризуються низьким дрейфом і гістерезисом, а також демонструють відмінну довгострокову стабільність і мінімальне споживання енергії. Калібрувальні дані, які надаються заводом-виробником, зберігаються в окремій пам'яті, що забезпечує повну сумісність датчиків без необхідності у додатковому калібруванні або зміні програмного забезпечення.

Опис датчика Gy-21 включає наступні характеристики: він має точність виміру вологості в діапазоні від 0 до 80% RH не більше ± 3%, а також точність вимірювання температури від -10 до +85 °C не більше ± 0.4 °C. Робочий діапазон вологості цього датчика становить від 0 до 100% RH, а температури - від -40 до +125 °C. Датчик працює при напрузі від 1.9 до 3.6 В, а розміри модуля складають 13 х 10 х 2 см. Маючи інтегрований нагрівач на чіпі, він забезпечує велику точність вимірювань та можливість використання в широкому діапазоні умов. Підключення до мікроконтролерів можливе через інтерфейси I2C або SPI. Однак, варто відзначити, що цей датчик має вищу вартість порівняно з іншими, менш точними моделями, такими як DHT-11.

Застосовуються у системах опалення, кондиціювання, термостати, автомобільні клімат-контролі, мобільні телефоні, планшети, навігатори.

- це електронний пристрій, який надає точний відлік часу і дати. Він має вбудований кварцовий генератор, що забезпечує високу точність вимірювання. Модуль підтримує комунікацію з мікроконтролером через різні інтерфейси,

такі як I2C або SPI, і має можливість зберігати інформацію про час навіть при відключенні живлення, завдяки вбудованій батареї або суперконденсатору.

Цей модуль призначений для роботи в широкому діапазоні робочих напруг від 2 до 5,5 В, що робить його сумісним з різними джерелами живлення та мікроконтролерами. З використанням стандартного елемента живлення CR2032, він забезпечує надійну та тривалу роботу в будь-яких умовах. Завдяки вбудованій пам'яті об'ємом 31 байт, модуль здатен зберігати значну кількість даних, що робить його ідеальним вибором для зберігання налаштувань, логів часу або будь-якої іншої інформації, яка потребує постійного збереження. Модуль відповідає вимогам щодо енергоефективності, максимізуючи продуктивність та тривалість роботи при максимальному споживанні струму до 300 мА при напрузі 2,5 В. Це робить його економічним у використанні та відповідальним за довготривалу роботу. Завдяки робочому діапазону температур від 0 до +70 °C, модуль може працювати в широкому спектрі кліматичних умов, що робить його відмінним вибором як для внутрішніх, так і для зовнішніх застосувань.

Для виводу інформації з датчиком буде використано OLED дисплей з жовто-синім кольором [11]. Ці два кольори матриці дозволяють зручно розділити загальну і службову області відображення інформації.

Характеристики цього дисплея наступні: його розмір становить 0.96", тип - OLED, з роз'ємом підключення 4-pin (VCC/GND/SCL/SDA), і роздільною здатністю 128*64. Щодо розмірів, вони складають 27x27x4 мм. Специфікація включає драйвер OLED модуля SSD1306, кут огляду більше 160 градусів, напруга живлення знаходиться в діапазоні від 3.3 до 6 В, а рівні вхідних сигналів - 3.3 В / 5 В.

O

L

E

D

підключається до мікроконтролера за допомогою I2C інтерфейсу, що спрощує процес комунікації та зменшує кількість потрібних контактів.

Наступний компонент - Модуль барометра BMP280 3.3B[12], який є важливим компонентом для вимірювання атмосферного тиску та температури в різноманітних застосунках.

Один із основних функціональних елементів модуля - вимірювання атмосферного тиску. Це важливий параметр, який використовується для прогнозування погодних умов, визначення висоти над рівнем моря та аналізу змін у кліматичних умовах. Модулі BMP280 широко використовуються у метеорологічних станціях для збору даних про погоду та створення погодних прогнозів. Вони допомагають вимірювати тиск у реальному часі, що дозволяє здійснювати точніші прогнози та моніторинг кліматичних змін. Модуль барометра BMP280 3.3B є продуктом компанії BOSCH, яка відома своїми високоякісними електронними пристроями та сенсорами. Завдяки надійності та точності продукції BOSCH, модулі BMP280 здатні забезпечити надійне та точне вимірювання атмосферного тиску та температури в різних застосунках. BOSCH відома своєю інноваційною розробкою та технологічними досягненнями у різних сферах.

Характеристики цього модуля наступні: напруга живлення може коливатися від 1.71 В до 3.6 В. Максимальна швидкість інтерфейсу I2C становить 3.4 МГц. Споживаний струм складає 2.7 мкА при частоті відліків в 1 Гц. Інтерфейс включає I2C, SPI (4 провід), та SPI (3 провід). Модуль має заводське калібрування. Рівень шуму досягає до 0.2 Па (1.7 см) і 0.01 градуса за Цельсієм. Діапазон вимірюваного тиску становить від 300 hPa до 1100 hPa (від 9000 м до -500 м). Його розміри складають 21 мм на 18 мм.

Нова модель датчика, порівняно з попередніми версіями (BMP085 і BMP180), пропонує три режими роботи:

ежим "Сон" (SLEEP) - є режимом зниженого енергоспоживання, спроектованим для заощадження електроенергії.

ежим "Примусовий" (FORCED) - аналогічний до режиму роботи датчиків команді від контролера, виконує вимірювання, передає результати вимірювання контролеру і повертається до режиму зниженого енергоспоживання.

ежим "Нормальний" (NORMAL) - є унікальним у даного датчика. У цьому режимі датчик власноруч прокидається та проводить замірювання тиску і температури у середовищі, а потім переходить у режим сну. Всі характеристики режиму Normal виконуються незалежно один від одного, і дані можна зчитувати в будь-який момент.

Також альтернативою OLED дисплею може слугувати LCD дисплей 1602 з підключенням I2C.

Дисплей LCD 1602 I2C[13] - це розширений варіант класичного дисплея LCD 1602, який використовує шину I2C для зв'язку з мікроконтролером. Він має два рядки по 16 символів кожен, що дозволяє виводити текстову інформацію з більшою чіткістю.

Характеристики цього індикатора включають наявність I2C інтерфейсу на мікросхемі PCF85741 та контролер дисплея HD44780. Також передбачено можливість регулювання контрастності. Дисплей має 2 рядки по 16 символів у кожному, загалом - 32 символи. Розмір пікселя становить 0,5 х 0,5 мм, а розміри самої плати індикатора складають 80 х 36 х 15 мм. Видима область екрану становить 64,5 х 14 мм, і кожен символ має 40 пікселів. Індикатор оснащений синім тлом і білим підсвічуванням. Напруга живлення становить 5 В, а діапазон робочих температур - від 0 до +60ºС.

Деякі особливості та переваги:

- Розширене відображення: Дисплей LCD 1602 забезпечує можливість відображення тексту та символів на двох рядках по 16 символів кожен, що дає більше місця для відображення інформації порівняно з OLED дисплеєм.

изька вартість: LCD 1602 I2C дисплей є більш доступним в порівнянні з OLED дисплеєм, що може зменшити витрати на проект.

обра читабельність: Хоча дисплей LCD може бути менш яскравим за OLED дисплей, він все ще забезпечує добру читабельність тексту навіть при яскравому світлі.

ростота використання: Використання інтерфейсу I2C спрощує підключення дисплею до мікроконтролера Arduino та зменшує кількість необхідних проводів.

табільність: LCD дисплей має добру стабільність роботи та менше схильний до виникнення проблем з відображенням даних порівняно з OLED технологією.

Одним з найпопулярніших датчків вологості та температури у більш дорогому сегменті є датчик DHT-22[14]. Датчик DHT22 (також відомий як AM2302) є цифровим датчиком вологості та температури, який відрізняється від свого попередника DHT11 вищою точністю та ширшим діапазоном вимірювань.

Характеристики цифрового датчика AM2302 від компанії ASAIR включають точність вимірювань температури на рівні 0.1 ºC та діапазон вимірювання вологості від 0 до 100%. Цей датчик може працювати в широкому діапазоні температур від -40 до 80 ºC з точністю вимірювання вологості ± 2% RH та точністю вимірювання температури ± 0.5 градуса. Він живиться від напруги 3.6-6 В, має 4 виводи, а також характеризується низьким споживанням енергії. Додатково, цей датчик не вимагає обв'язки і може працювати при використанні досить довгих дротів.

Особливості та переваги:

ифровий вихідний сигнал: Він має цифровий вихідний сигнал, що спрощує його підключення до мікроконтролера або мікропроцесора. Для зчитування даних може використовуватися протокол однопровідникового інтерфейсу.

исока стабільність та надійність: DHT22 відомий своєю високою стабільністю

та надійністю вимірювань. Він має вбудований датчик температури і вологості, який дозволяє здійснювати точні вимірювання в різних умовах.
ирокий діапазон застосування: DHT22 може бути використаний у багатьох областях, включаючи автоматизацію, системи контролю клімату, агрономію, вимірювання та багато іншого.

Обґрунтування вибору мови програмування є ключовою складовою при розробці будь-якого проекту. Вибір мови програмування повинен враховувати потреби проекту, його складність, доступність ресурсів та навички команди розробників. Для розробки системи віддаленого моніторингу мікроклімату було вибрано мову програмування Arduino і також середовище розробки ПЗ від розробника Arduino Software. Мова Arduino базується на мові C. Це означає, що багато синтаксичних правил та підходів до програмування аналогічні до мови C. Arduino IDE - це безкоштовне середовище розробки, яке надає простий інтерфейс для програмування плат Arduino, мова Arduino спрощена в порівнянні з іншими, що дозволяє швидко створювати прості програми. Arduino широко використовується в спільноті хобістів, студентів, інженерів та розробників по всьому світу, завдяки цьому легко знайти велику кількість відеогайдів, документації, прикладів коду та підтримки спільноти. Також, варто зауважити, що Arduino надає широкий вибір готових бібліотек з різними датчиками, пристроями, модулями. Багато бібліотек створено розробниками-любителями, які в більшості випадках виявляються краще ніж базові бібліотеки для звичайних датчиків. Більшість з цих бібліотек можна знайти на Github або ж на сайтах, які продають датчики та модулі. Arduino підтримує велику кількість апаратних платформ, такі як Arduino Uno, Nano, та інші. Це дозволяє обирати оптимальну для кожного користувача платформу для проекту залежно від потреб.

Додатково до обгрунтувань вибору мови програмування Arduino варто відзначити його гнучкість та розширюваність. Arduino може бути використаний для розробки широкого спектру проектів, починаючи від

простих прототипів до складних систем управління та автоматизації. Крім того, наявність великої кількості готових компонентів та модулів, які сумісні з , робить його найкращим вибором для швидкої реалізації ідей. Це дозволяє швидко втілити концепцію проекту в життя та швидко переходити до тестування та вдосконалення. Arduino сприяє інтеграції з іншими технологіями, такими як IoT (Internet of Things), що робить його ідеальним вибором для проектів, які потребують зв'язку та взаємодії з іншими пристроями та хмарними сервісами.

Враховуючи ці переваги, вибір мови програмування Arduino є раціональним та ефективним для реалізації проекту системи віддаленого моніторингу мікроклімату.

2.3 Розгорнута постановка завдання

В процесі розробки БР необхідно виконати наступний обсяг робіт:

изначити та обґрунтувати актуальність теми БР;

изначити призначення системи та область її впровадження;

наліз існуючих метеостанцій, аналоговічні системи з метою виявлення їх позитивних і негативних властивостей. Результати аналізу будуть враховані при подальшій розробці;

ідбір та підключення необхідних датчиків, зокрема датчика вологості, температури та барометра, для забезпечення точного та комплексного моніторингу мікрокліматичних умов;

творення програмного коду для зчитування даних з датчиків, обробки і аналізу цих даних, а також відображення результатів на OLED або LCD дисплеї;

еалізація функціоналу для збереження отриманих даних в окремий файл для подальшого використання та аналізу;

роведення тестування системи з метою виявлення та усунення можливих помилок та недоліків у роботі;

адати перелік скорочень, символів і спеціальних термінів, що будуть використані при розробці БР;

авести список літератури, що використалась при розробці БР;

озробити анотацію на українській та англійській мовах.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Темою БР вже було визначено розробка програмного забезпечення системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією.

Після проведення аналізу існуючих систем аналогів, програм-аналогів та виявлення їх позитивних сторін було вирішено розробити ПЗ з урахуванням виявлених позитивних якостей та за виключенням недоліків для рішення поставленої технічним завданням задачі.

Оцінюючи задачі, необхідні для досягнення поставленої мети розробки системи віддаленого моніторингу мікроклімату, виділимо основні критерії її функціонування:

истема повинна забезпечувати достовірні вимірювання мікрокліматичних параметрів з високою точністю, що є ключовим критерієм для надійного моніторингу середовища.

истема має мати потенціал для реалізації додаткових функцій, таких як сповіщення про зміни параметрів, автоматизація процесів, зберігання історичних даних тощо.

ля довготривалого функціонування системи важливо, щоб вона була енергоефективною, забезпечуючи мінімальне споживання енергії.

истема повинна бути готовою до масштабування, щоб можна було розширювати її функціональність або додавати нові компоненти за потреби.

раховуючи обмежені бюджетні ресурси, важливо, щоб реалізація системи була економічно ефективною і доступною.

Одна з основних необхідностей розробки даного приладу є вартість розробки, на ринку є досить багато дорогих метеостанцій, які

використовуються професійно у різних сферах, а дешеві альтернативи в основному не сильно відрізняються за наповненням датчиками від самостійної розробки власноруч на базі Arduino. Але варто зазначити, що багато модулів та датчиків є вартісними, тому для розробки були використані не дуже дорогі компоненти, що дозволить зменшити загальну вартість проекту. Більш того, використання Arduino UNO як основи для розробки системи дозволяє з легкістю масштабувати функціональність та замінювати компоненти за потреби. Розробка даного пристрою власноруч не лише забезпечить економію коштів, але й надасть можливість власноручно адаптувати систему під конкретні потреби користувача.

Проектні рішення, які будуть використовуватися, повинні виконувати всі перераховані вище функції. Крім того, вони повинні бути компактними і зрозумілими і відповідати вимогам до систем моніторингу.

Використання бібліотек є ключовим аспектом у розробці системи віддаленого моніторингу мікроклімату на базі Arduino. Багато функціональності, яка потрібна для реалізації конкретних завдань, вже реалізована у вигляді готових бібліотек. Наприклад, для роботи з датчиками вологості та температури, такими як DHT11, DHT22, або BMP280, можна використовувати відповідні бібліотеки, які вже містять функції для зчитування даних з цих датчиків та їх обробки. Використання бібліотек дозволяє значно спростити процес програмування, знизити його складність, а також забезпечити більшу стабільність та надійність програмного забезпечення.

Вибір датчиків в системі моніторингу стану атмосфери базується на їхній здатності надати достовірні та точні вимірювання необхідних параметрів. Датчик DHT-11 був обраний через свою простоту використання, доступність та здатність вимірювати температуру та вологість з задовільною точністю.

Альтернативою цього датчика, буде також розглянуто датчик GY-21, він є не таким популярним у застосуванні, але теж доступний за ціною та здатний вимірювати показники з адекватною точністю.

Також буде використовуватися датчик BMP280, який забезпечує вимірювання тиску та температури з високою точністю. Враховуючи різноманітність умов середовища, в яких може застосовуватися система моніторингу, важливо мати можливість користуватися різними типами датчиків, що дозволить отримувати комплексну та надійну інформацію про мікроклімат. Кожен з обраних датчиків має свої переваги та обмеження, і їхній вибір базується на вимогах до точності, доступності та можливості вимірювань у конкретних умовах експлуатації.

Для визначення часу та дати використовується модуль Real Time Clock (RTC), що дозволяє точно визначати поточну дату та час. Цей модуль підключений до Arduino і забезпечує систему актуальними даними про час, що є важливим для точного реєстрування моментів вимірювання параметрів мікроклімату. За допомогою RTC забезпечується незалежність від зовнішніх джерел часу, так що система може правильно функціонувати навіть у випадку відключення від мережі або перезавантаження. Такий підхід гарантує стабільність та надійність системи в умовах різних експлуатаційних умов.

Для виведення інформації для користувача використовується OLED дисплей. Завдяки своїй яскравості, контрастності та можливості відображення інформації на темному фоні, він забезпечує зручне та ефективне сприйняття даних про мікроклімат без надмірного навантаження на очі користувача. Відображення на OLED дисплеї може відображати інформацію про температуру, вологість, атмосферний тиск та дату й час. OLED дисплей енергоефективний і забезпечує зручне відображення інформації навіть у нічний час.

Система віддаленого моніторингу мікрокліматичних змін базується на використанні датчиків для вимірювання температури, вологості та тиску атмосфери. Кожен датчик розташований на відповідному модулі, який підключений до мікроконтролера Arduino Uno. Мікроконтролер зчитує дані з датчиків та відображає їх на OLED або LCD дисплеї для зручного перегляду

користувачем. Крім того, дані, що збираються з датчиків, також логуються в окремий файл за допомогою мови програмування Python. Програма на Python працює в фоновому режимі на комп'ютері, до якого підключений Arduino через USB-порт. Вона перехоплює дані, які надсилаються з Arduino через серійний порт, та записує їх у відповідний файл. Зв'язок із серійним портом встановлюється за допомогою бібліотеки **serial**. Після встановлення зв'язку програма відкриває файл **data.txt** для запису. У циклі while програма чекає на отримання даних з серійного порту. Коли дані надходять, вони зчитуються, декодуються з байтового формату в рядок, видаляються зайві пробіли і символи нового рядка за допомогою методу strip(), та записуються до файлу

Після запису даних у файл, вони також виводяться на екран за допомогою функції print(), щоб вони були видимі у вікні консолі.

Цей код постійно чекатиме на нові дані, які надходять зі з'єднання через COM порт, і записуватиме їх до файлу data.txt.

Такий підхід дозволяє зберігати історичні дані про мікроклімат, що сприяє подальшому аналізу та моніторингу змін у середовищі.

Таким чином, маємо всі необхідні дані для побудови структурної і функціональної схем підсистеми.

3.2 Розробка структурної схеми

У цьому розділі буде надано детальний опис структурної схеми пристрою метеостанції. Призначенням цієї схеми є візуалізація взаємозв'язків між різними компонентами системи та розкриття їх функціональних завдань у контексті реалізації задуманого проекту.

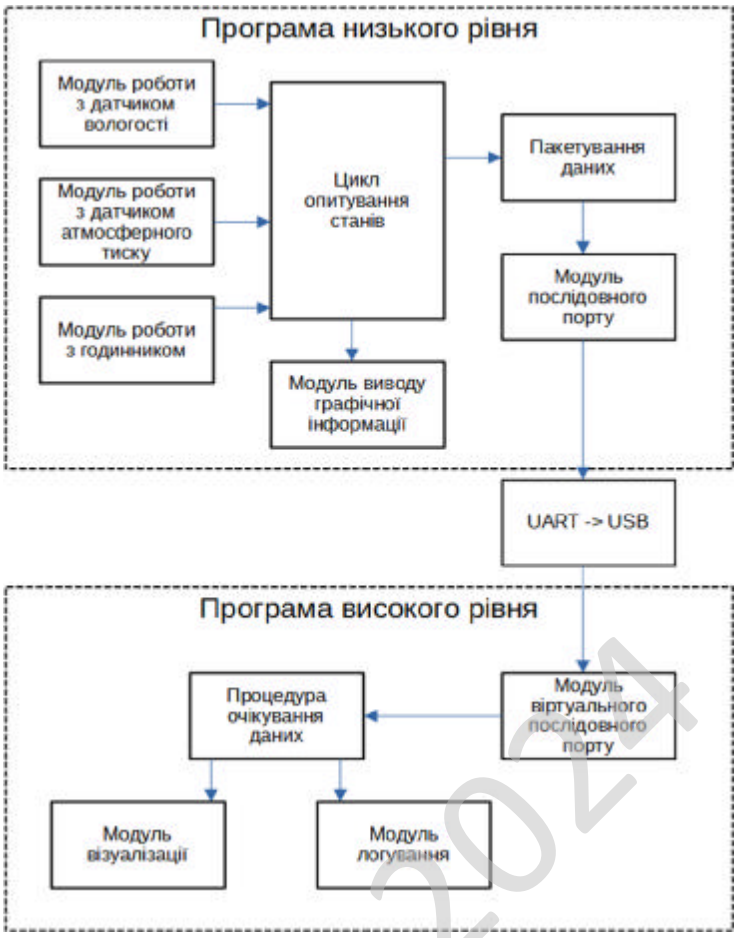

Рисунок 3.1 – Структурна схема пристрою

Ця структурна схема відображає взаємозв'язки між різними компонентами нашої метеостанції. Основним компонентом є мікроконтролер Arduino Uno, який виконує обробку даних від датчиків та управляє роботою системи.

Датчики вимірюють температуру, вологість та атмосферний тиск. Отримані дані передаються до мікроконтролера для подальшої обробки.

Arduino Uno виконує обробку вхідних даних від датчиків. Він регулює роботу датчиків, виконує аналіз даних та приймає рішення щодо відображення інформації на дисплеї.

Дисплей відображає поточні показники температури, вологості та атмосферного тиску. Це дозволяє користувачу швидко переглянути стан атмосферних умов.

Ця структурна схема демонструє, як мікроконтролер Arduino Uno взаємодіє з датчиками та дисплеєм для забезпечення правильної роботи метеостанції та надання користувачам необхідної інформації про погоду.

Розробка функціональної схеми

У цьому розділі розглядається розробка функціональної схеми системи метеостанції. Подано огляд функціональних блоків та їх взаємозв'язків, описано роль кожного блоку у виконанні завдань системи. З'ясовано, як вони взаємодіють між собою для забезпечення правильної роботи метеостанції.

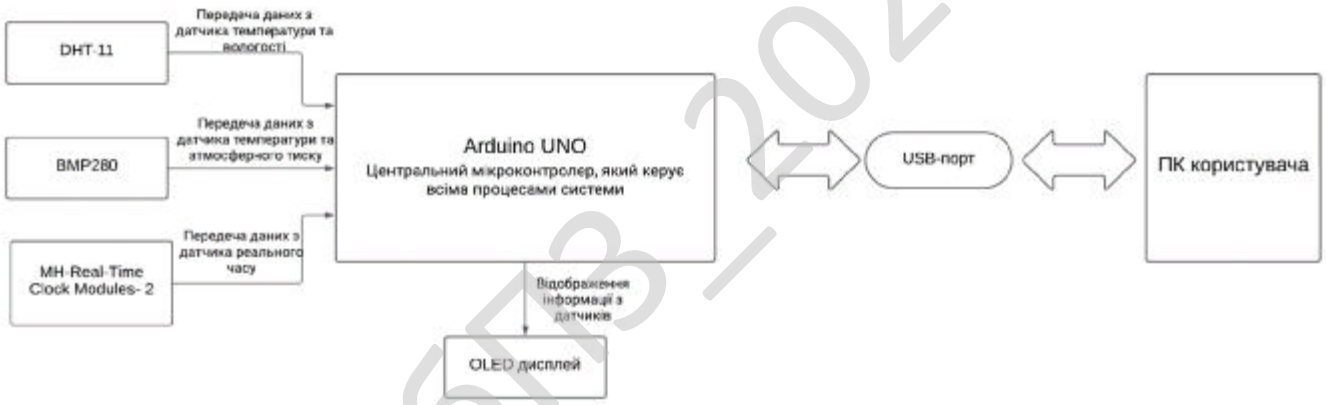

Рисунок 3.2 – Функціональна схема пристрою

У цьому розділі детально розглядається функціональна схема метеостанції з урахуванням ролі кожного функціонального блоку у процесі її роботи. Принципи взаємодії між компонентами системи висвітлені з точки зору їх внутрішньої взаємодії та спільної роботи для забезпечення надійної та ефективної роботи метеостанції. Додатково, в цьому розділі розглядаються важливі аспекти функціонування системи, такі як обмін даними між компонентами, обробка та аналіз отриманих вимірювань, а також управління системою з боку користувача через ПК та USB-порт.

Мікроконтролер Arduino Uno:

сновний керуючий модуль, який керує всіма процесами метеостанції.

читує дані з датчиків.

ерує виведенням інформації на дисплей.

заємодіє з ПК користувача через USB-порт для передачі даних.

Датчик DHT-11:

дійснює вимірювання температури та вологості повітря.

адає отримані дані мікроконтролеру для подальшої обробки.

Дисплей OLED 0.96" 128x64:

ідображає вимірювані параметри (температура, вологість тощо) для користувача.

ідображає іншу інформацію, таку як дату та час, статус підключення і т.д.

Датчик BMP280:

икористовується для вимірювання атмосферного тиску та температури.

адає дані про тиск та температуру мікроконтролеру для подальшої обробки.

ПК користувача та USB-порт:

икористовується для підключення метеостанції до ПК.

озволяє користувачу отримувати дані з метеостанції.

оже використовуватися для налаштування та управління параметрами метеостанції.

Ці компоненти взаємодіють між собою, забезпечуючи збір, обробку та відображення інформації про погодні умови для користувача.

3.4 Розробка діаграми процесів

У цьому розділі надається детальний опис принципової схеми підключень для реалізації метеостанції. З використанням інтерактивного інструменту від Circuito.io була створена принципова схема, яка відображає всі компоненти та їх з'єднання.

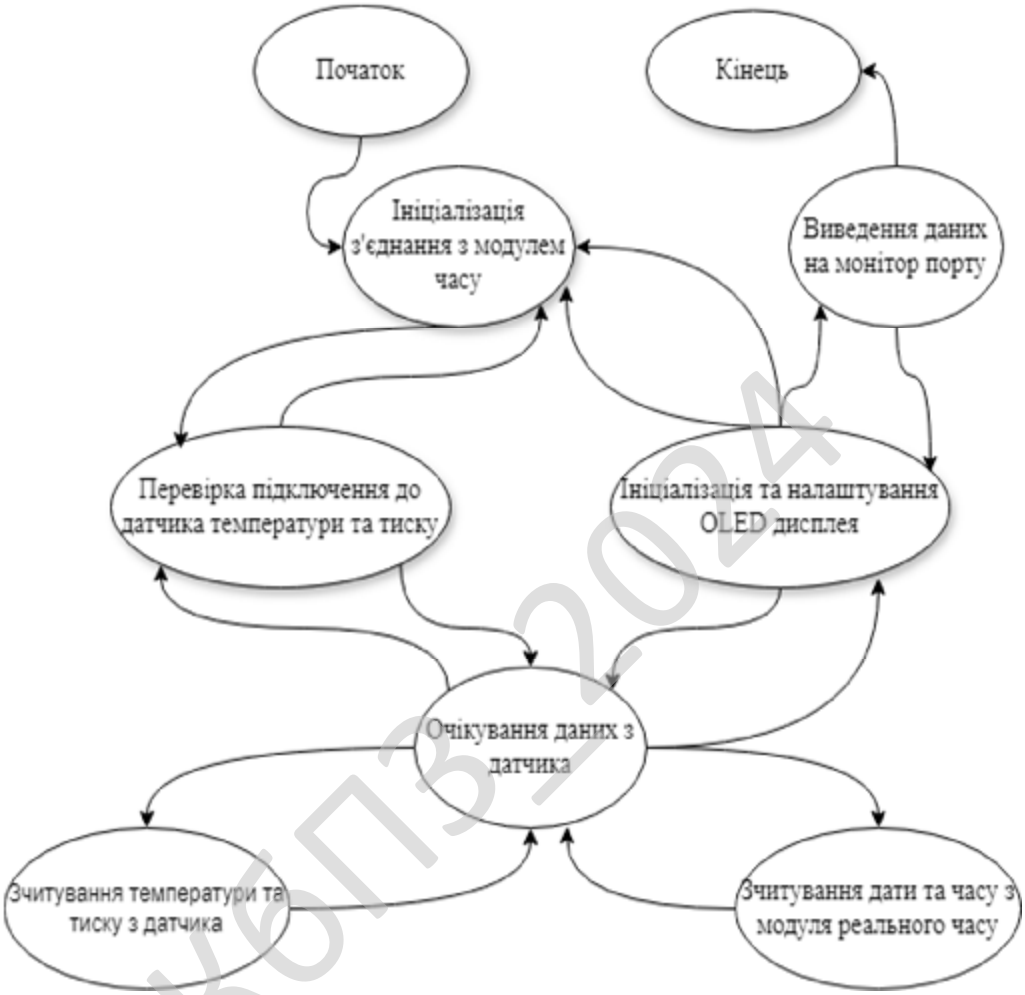

Рисунок 3.3 – Діаграма взаємодії процесів

Принципова схема включає в себе різноманітні компоненти, які забезпечують функціональні можливості метеостанції. Кожен з цих компонентів підключений до мікроконтролера Arduino Uno за допомогою макетної плати. Принцип підключення базується на використанні цифрових та аналогових входів/виходів Arduino, а також спеціальних протоколів зв'язку, таких як I2C та SPI.

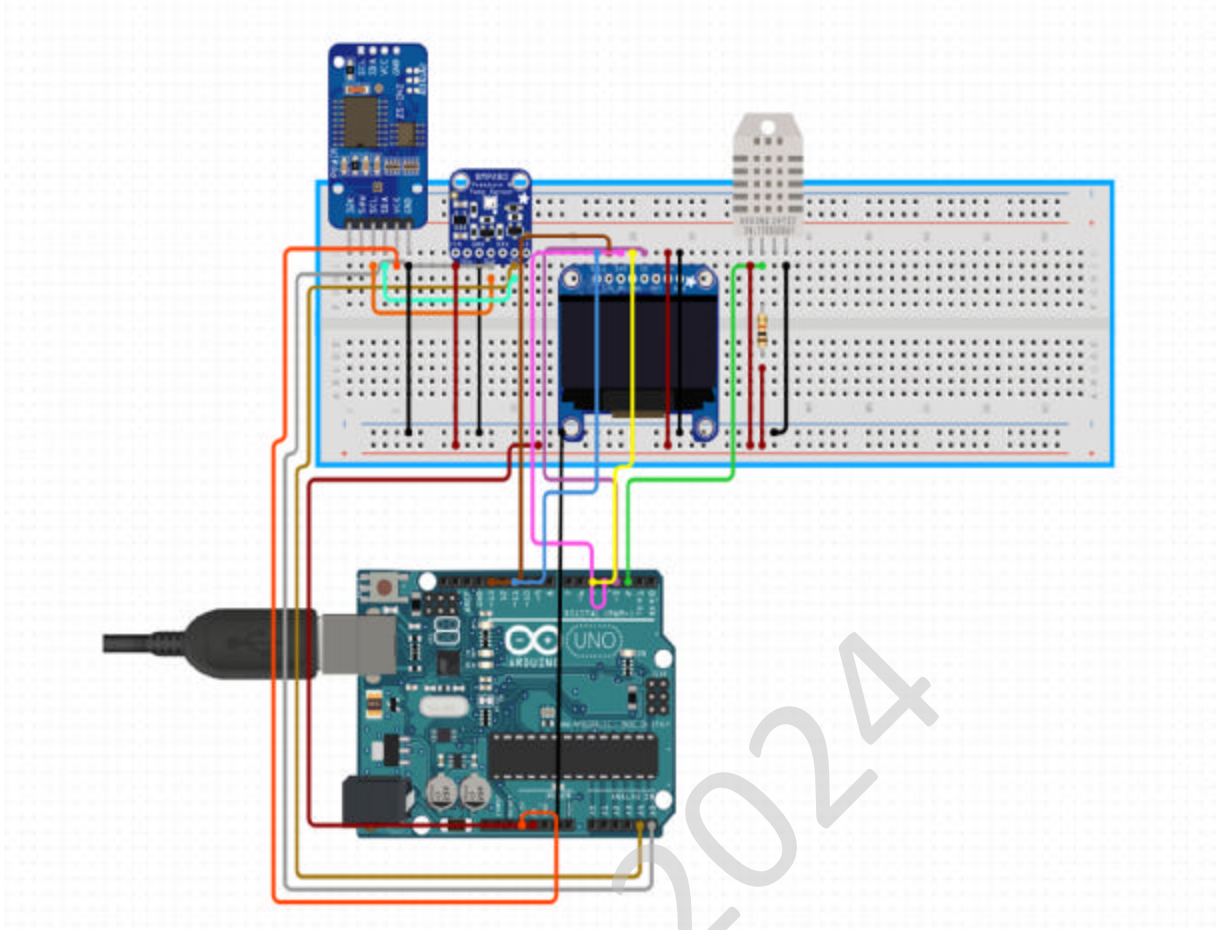

Рисунок 3.4 – Схема пристрою метеостанції з використанням датчика

У макетній схемі, кожен компонент з'єднаний з відповідними портами Arduino, які забезпечують передачу даних та сигналів керування. Наприклад, датчики температури та вологості підключені до аналогових входів Arduino для зчитування аналогових сигналів, тоді як OLED дисплей та барометр підключені до цифрових портів для обміну даними через інтерфейси I2C. Макетна плата дозволяє зручно розміщати та підключати компоненти, забезпечуючи надійне з'єднання та легкість у відладці та модифікації пристрою. Крім того, вона дозволяє ефективно використовувати простір та ресурси Arduino для оптимальної роботи всієї системи метеостанції.

Альтернативною схемою пристрою може бути використання датчика не буде.

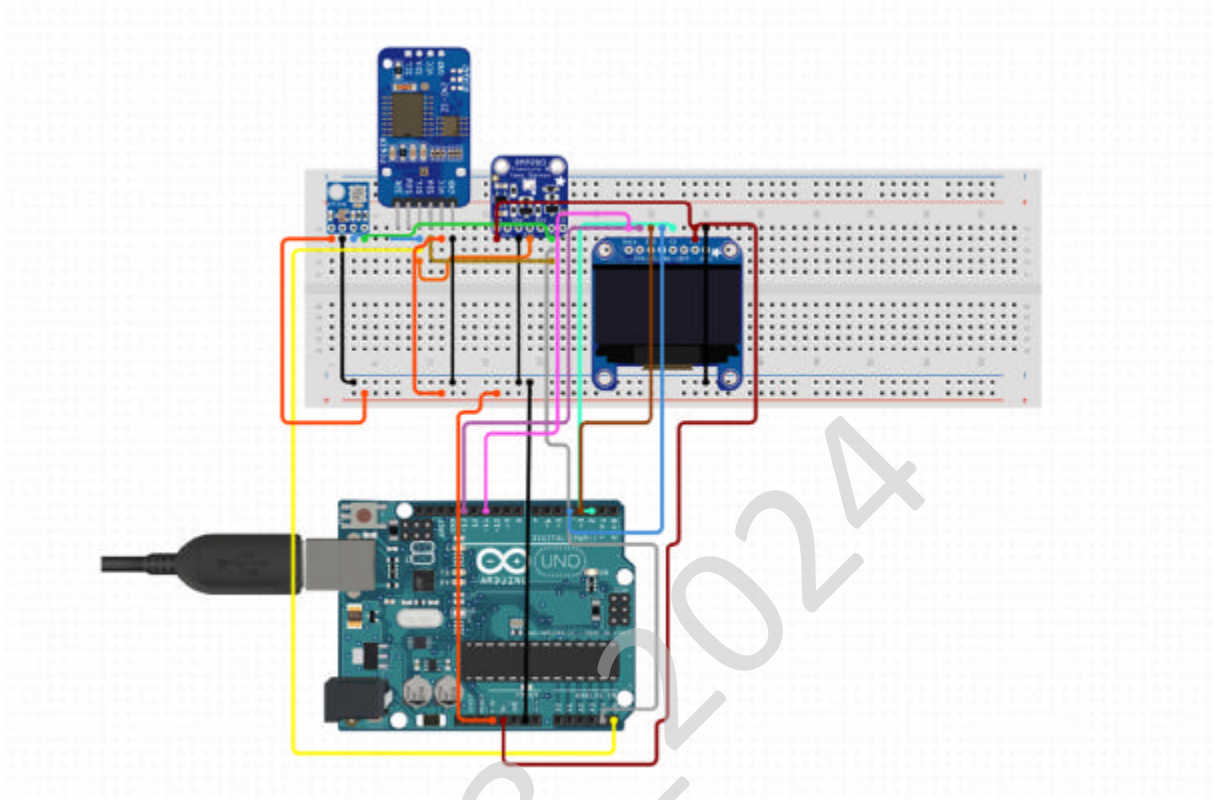

Рисунок 3.5 – Схема пристрою метеостанції з використанням датчика

Було представлено дві альтернативні схеми підключення компонентів до мікроконтролера Arduino Uno. Одна з них використовує датчик температури та вологості DHT11, тоді як інша замінює його на більш точний датчик GY-21.

Перша схема з DHT11, хоч і має меншу точність вимірювання, є більш доступною та простою у використанні. DHT11 підключається за допомогою одного цифрового піну, що полегшує його інтеграцію у систему.

Друга схема, з використанням GY-21, надає більш точні вимірювання температури та вологості, але вимагає додаткових підключень та конфігурації.

Це може бути вигідним у випадку, якщо точність вимірювань є критичним фактором для проекту.

Обидві схеми дозволяють ефективно використовувати ресурси Arduino та забезпечують надійне та зручне підключення компонентів метеостанції.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

У цьому розділі надається опис блок-схем алгоритмів роботи пристрою та системи. Розглядаються кроки та послідовність операцій, необхідних для виконання основних функцій метеостанції. Кожен блок алгоритму детально аналізується та пояснюється, що дозволяє отримати чітке уявлення про функціонування системи в цілому.

Так як, при розробці було виявлено, що деякі датчики не можуть працювати одночасно через різні проблеми, було прийнято рішення розділити датчики на окремі програми, які в цілому не відрізняються, тому що у багатьох датчиків майже однаковий функціонал.

Розглянемо поетапно алгоритм роботи програми з датчиком BMP280 та інших обов’язкових компонентів:

Крок 1: Початок програми.

Крок 2: Ініціалізація з'єднання з модулем реального часу (RTC).

Крок 3: Перевірка підключення до датчика температури та тиску

Крок 4: Ініціалізація та налаштування OLED дисплея.

Крок 5: Очікування зчитування даних з датчика.

Крок 6: Зчитування температури та тиску з датчика BMP280.

Крок 7: Зчитування дати та часу з модуля реального часу (RTC).

Крок 8: Оновлення відображення даних на OLED дисплеї.

Крок 9: Виведення даних на монітор порту швидкості 9600 бод.

Крок 10: Завершення циклу.

Рисунок 4.1 – Блок-схема роботи метеостанції з датчиком BMP280

Також, розглянемо алгоритм роботи з датчиком температури та вологості DHT11 та іншими обов’язковими компонентами:

Крок 1: Початок програми.

Крок 2: Ініціалізація датчика вологості і температури DHT11.

Крок 3: Ініціалізація модуля реального часу DS1302.

Крок 4: Перевірка з'єднання з OLED дисплеєм.

Крок 5: Очікування зчитування даних з датчиків.

Крок 6: Зчитування вологості і температури з датчика DHT11.

Крок 7: Зчитування дати та часу з модуля реального часу DS1302.

Крок 8: Оновлення відображення даних на OLED дисплеї.

Крок 9: Виведення даних на монітор порту швидкості 9600 бод.

Крок 10: Завершення циклу.

Рисунок 4.2 – Блок-схема роботи метеостанції з датчиком DHT-11

Як раніше і зазначав, що логування даних з метеостанції відбувається за допомогою невеликої програми на мові програмування Python, так як Arduino IDE не має прямої підтримки запису тексту у файли на пристрої зберігання, оскільки мікроконтролери Arduino зазвичай не мають файлової системи, як комп'ютери або мобільні пристрої.

Алгоритм роботи програми для логування інформації:

Крок 1: Початок програми.

Крок 2: Встановлення з'єднання з COM-портом (COM3) і встановлення швидкості передачі даних (9600 бод).

Крок 3: Відкриття файлу "data.txt" для запису.

Крок 4: Зчитування рядка даних з COM-порту.

Крок 5: Декодування зчитаних даних з байтового формату в текстовий формат і видалення зайвих пробілів.

Крок 6: Виведення считаних даних в консоль.

Крок 7: Запис считаних даних у файл "data.txt".

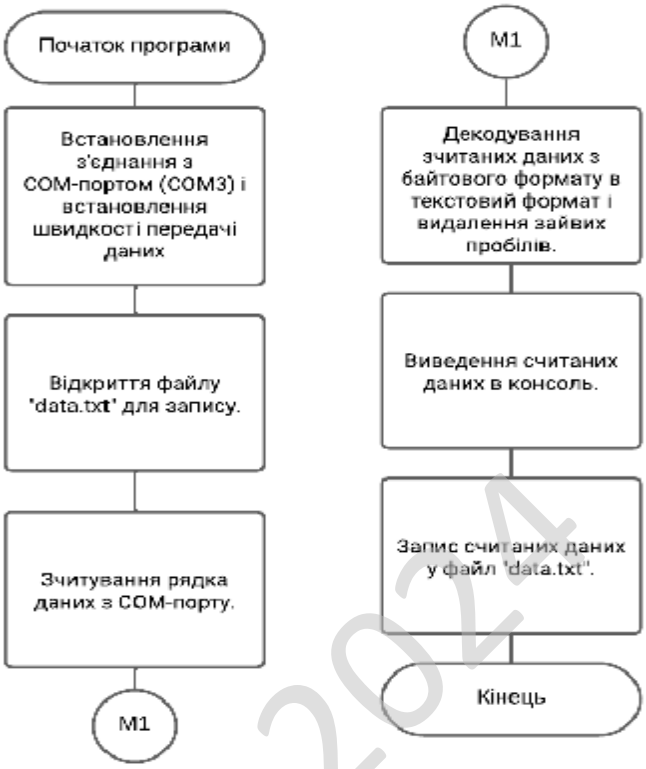

Рисунок 4.3 – Блок-схема програми для записування даних у файл

Програмний код блок-схеми наведеного на рисунку 4.1. відображає практичне застосування використання датчика температури і тиску BMP280, модуля реального часу DS1302 та OLED дисплею для створення пристрою метеостанції.

У функції setup() відбувається початкова ініціалізація модулів. Спочатку відбувається ініціалізація Serial для відладки та зчитування даних. Потім ініціалізуються модулі BMP280 і DS1302 для зчитування температури, тиску та поточного часу відповідно. На останньому кроці відбувається ініціалізація OLED дисплею.

У функції loop() відбувається цикл зчитування даних. Кожну секунду зчитуються дані температури і тиску за допомогою датчика BMP280, а також

поточний час за допомогою модуля DS1302. Після зчитування даних вони відображаються на OLED дисплеї. Температура і тиск відображаються разом з відповідними піктограмами, які допомагають користувачу легше розуміти відомості. Також відображається поточний час та дата.

Усі зчитані дані також виводяться у Serial Monitor. Це допомагає відслідковувати та аналізувати дані в реальному часі.

Рисунок 4.4 – Відображення даних на OLED дисплеї

При роботі даної програми та метеостанції на базі Arduino UNO маємо такі дані впродовж деякого часу:

Програмний код, який відображений блок-схемою на рисунку 4.2. виконує функції метеостанції за допомогою Arduino та певних датчиків.

Код використовує бібліотеки Wire.h, Adafruit_GFX.h, датчиками DHT11 та DS1302 для вимірювання температури, вологості та часу

відповідно. У функції setup(), код ініціалізує Serial Monitor, датчик DHT11 та зациклюється. У функції loop(), затримка 1 секунда дозволяє зчитати нові дані.

За допомогою об'єкта RtcDateTime, програма отримує поточний час з модуля DS1302. Дані про вологість і температуру зчитуються з датчика DHT11. Зчитані дані відображаються на OLED дисплеї за допомогою відповідних значків і тексту. Всі дані також виводяться у Serial Monitor.

При роботі цього варіанту метеостанції маємо такі результати:

Humidity: 14.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:17

Humidity: 14.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:18

Humidity: 14.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:20

Humidity: 14.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:21

Humidity: 14.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:22

Humidity: 14.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:23

Humidity: 14.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:24

Humidity: 14.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:25

Humidity: 37.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:26

Humidity: 37.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:27

Humidity: 20.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:28

Humidity: 20.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:29

Humidity: 17.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:30

Humidity: 17.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:31

Humidity: 16.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:32

Humidity: 16.00% Temperature: 21.00C

Date: 4/3/2024 Time: 17:4:33

Результати вимірювань датчиків BMP280 та DHT11 показують різницю в точності та якості вимірювань. Вимірювання температури та тиску за допомогою BMP280 відображаються з більшою точністю та стабільністю. Наприклад, температура, виміряна BMP280, коливається в діапазоні від 25.52°C до 25.55°C, що свідчить про високу точність вимірювання. У той же час, датчик вологості та температури DHT11 відображає стабільні вимірювання температури приблизно 21.00°C, проте навіть при стабільних умовах вимірювання вологості показують значні коливання.

Це свідчить про те, що датчик BMP280 працює якісніше і надає більш точні та надійні вимірювання температури та тиску. Така різниця в якості вимірювань може бути важливою для застосувань, де потрібна висока точність, наприклад, в системах контролю клімату або вимірювання погодних умов.

Також варто розписати, як працюють бібліотеки, які найбільше допомогли в розробці даного пристрою, одна з таких бібліотек є

Бібліотека Adafruit для сенсора орієнтації BMP280 містить ряд корисних функцій для роботи з цим сенсором. Функція begin() використовується для ініціалізації сенсора та перевірки його коректного

підключення. setSampling() встановлює параметри зчитування, такі як режим роботи та роздільна здатність температури та тиску. Дві ключові функції - readTemperature() та readPressure() - зчитують температуру та тиск відповідно. Функція readAltitude() обчислює приблизну висоту над рівнем моря на основі зчитаних значень тиску та переданого атмосферного тиску на рівні моря. Крім того, takeForcedMeasurement() дозволяє виконати нове вимірювання в режимі примусового зчитування. Всі ці функції використовуються для отримання та обробки даних з сенсора BMP280 у програмах Arduino.

Бібліотека для RTC (Real-Time Clock) DS1302 містить набір функцій для взаємодії з годинником. Функція Begin() використовується для ініціалізації зв'язку з годинником. GetIsWriteProtected() та SetIsWriteProtected(bool) визначають і встановлюють захист від запису. IsDateTimeValid() перевіряє, чи є поточні дані дати та часу дійсними. GetIsRunning() та SetIsRunning(bool) визначають і встановлюють стан роботи годинника. Функції GetTrickleChargeSettings() та SetTrickleChargeSettings(uint8_t) визначають і встановлюють параметри струмового заряду годинника. SetDateTime(const RtcDateTime&) та GetDateTime() встановлюють і отримують дату та час з годинника відповідно. SetMemory(uint8_t, uint8_t) та GetMemory(uint8_t) встановлюють і отримують дані з пам'яті годинника.

Бібліотека Wire реалізує TWI/I2C для Wiring та Arduino. Вона дозволяє взаємодіяти з пристроями, що підтримують протокол TWI/I2C. Клас TwoWire містить функції для роботи з цим протоколом. Функція begin() ініціалізує бібліотеку, а end() завершує роботу з нею. Метод setClock() встановлює швидкість зв'язку. Функція requestFrom() використовується для отримання даних від пристрою за його адресою. Метод beginTransmission() вказує, що починається передача даних до пристрою, а endTransmission() завершує цю передачу. Методи write() та read() відповідають за запис та читання даних відповідно. Функції onReceive() та onRequest() встановлюють функції

зворотного виклику, які викликаються при отриманні або запиті даних від майстра.

Також, бібліотека Adafruit-GFX - це потужний і простий у використанні інтерфейс для візуального відображення даних на графічних дисплеях. Вона надає базові класи та методи для створення та керування графічними об'єктами, такими як лінії, кола, прямокутники, текст і т. д., на різних графічних дисплеях, зокрема LCD та OLED.

Основні можливості бібліотеки Adafruit-GFX включають:

творення графічних об'єктів: Вона дозволяє створювати графічні об'єкти, такі як лінії, кола, прямокутники, трікутники, за допомогою відповідних методів.

обота з кольорами і шрифтами: Adafruit-GFX дозволяє встановлювати кольори ліній та областей, а також виводити текст з різними шрифтами і розмірами.

асштабування і трансформація: Вона підтримує масштабування, поворот та інші трансформації графічних об'єктів.

ідтримка різних типів графічних дисплеїв: Бібліотека може використовуватися з LCD та OLED дисплеями різних роздільної здатності та інтерфейсів, таких як SPI або I2C.

ростота використання: Adafruit-GFX має простий та зрозумілий інтерфейс, що робить її ідеальним вибором для початківців у графічному програмуванні.

Ця бібліотека часто використовується разом з бібліотеками конкретних графічних дисплеїв від Adafruit, такими як бібліотека Adafruit-SSD1306 для OLED дисплеїв або бібліотека Adafruit-ST7735 для TFT LCD дисплеїв. Завдяки Adafruit-GFX, користувачі можуть легко створювати графічні інтерфейси для своїх проектів на Arduino та інших платформах.

Бібліотека Adafruit_SSD1306 - це інтерфейс для використання графічних OLED дисплеїв з контролером SSD1306 в мікроконтролерних проектах на платформі Arduino. Ця бібліотека дозволяє легко керувати графічними елементами на OLED дисплеї та виводити текст, графіки та інші зображення.

Основні можливості бібліотеки Adafruit_SSD1306 включають:

ивід тексту: Завдяки бібліотеці можна виводити текст різними шрифтами та розмірами на OLED дисплей.

творення графічних об'єктів: Adafruit_SSD1306 дозволяє створювати графічні об'єкти, такі як лінії, кола, прямокутники, трикутники та інші, і відображати їх на дисплеї.

асштабування та трансформація: Бібліотека підтримує масштабування, поворот та інші трансформації графічних об'єктів.

росте керування дисплеєм: Adafruit_SSD1306 надає методи для управління яскравістю, включення / вимикання дисплею та інші опції керування дисплеєм.

ідтримка анімації і прокрутки: Бібліотека дозволяє створювати анімаційні ефекти та прокручувати вміст на OLED дисплеї.

Бібліотека Adafruit_SSD1306 легко інтегрується з Arduino IDE та іншими середовищами розробки, що дозволяє розробникам швидко створювати графічні інтерфейси для своїх проектів. Вона часто використовується для створення інформаційних панелей, годинників, моніторів систем та багатьох інших пристроїв з відображенням на OLED дисплеях.

Файл DHT.h є частиною бібліотеки для Arduino, яка використовується для взаємодії з датчиками вологості та температури DHT11, DHT21 (AM2301), DHT22 (AM2302), та подібними. Основна функція цієї бібліотеки - зручний збір даних від датчика та їх подальше використання у вашому проекті.

Основні можливості DHT.h включають:

читування даних: Бібліотека дозволяє зчитувати температуру та вологість повітря з датчика DHT.

ідтримка різних моделей: Вона підтримує різні моделі датчиків, такі як DHT11, DHT21 (AM2301) та DHT22 (AM2302), що дозволяє використовувати один і той же код для різних пристроїв.

ростота використання: Інтерфейс бібліотеки досить простий, що дозволяє швидко і легко інтегрувати датчики в ваш проект.
табільність та надійність: DHT.h надійно працює з датчиками вологості та температури, забезпечуючи стабільне зчитування даних.
умісність з Arduino IDE: Бібліотека працює з Arduino IDE та іншими середовищами розробки для мікроконтролерів Arduino, що спрощує її використання.

Використання DHT.h дозволяє легко і швидко отримувати дані про температуру та вологість, що дозволяє створювати різноманітні проекти, які вимагають моніторингу цих параметрів, такі як кліматичні системи, стан внутрішнього середовища, сільське господарство та багато інших.

Бібліотеки, виконують важливу роль у спрощенні роботи з датчиками та виведенні інформації на дисплеї у проектах Arduino. Вони надають розробникам зручний інтерфейс для взаємодії з апаратним обладнанням, таким як датчики вологості та температури, та графічними дисплеями. Це дозволяє швидко та легко інтегрувати ці функціональності в проекти без необхідності вдаватися до складних низькорівневих операцій. Розробники цих бібліотек заслуговують на вдячність за їхню роботу та внесок у спільноту Arduino. Широке поширення цих бібліотек, їхній відкритий доступ та активна підтримка сприяють розвитку та поширенню знань у сфері вбудованих систем та IoT. Це сприяє прискоренню процесу розробки, забезпечуючи розробникам доступ до готових рішень та сприяючи інноваціям у цьому сегменті.

4.2 Захист розробленого програмного забезпечення

Згідно закону України від 23.12.93 р. № 3792-XII “Про авторське право і суміжні права” в редакції від 16.07.01 року визначено терміни “комп’ютерна програма”. Згідно цього ж Закону, комп’ютерна програма є об’єктами авторського права, тобто інтелектуальною власністю. Основними аспектами

захисту програмного забезпечення є забезпечення конфіденційності, цілісності та доступності даних та функціональності пристрою.

Але варто зауважити, що було використано бібліотеки, які є загальнодоступними. У процесі розроблення метеостанції були використані різноманітні бібліотеки для забезпечення роботи з різними компонентами та модулями.

Основні бібліотеки:

- Wire.h: Ця бібліотека використовується для забезпечення зв'язку по шині I2C між Arduino та різними пристроями, такими як датчики температури, тиску та модуль часу.

- Adafruit_GFX.h: Бібліотека Adafruit_GFX дозволяє легко керувати графічними об'єктами та малюнками на дисплеї за допомогою Arduino.
dafruit_SSD1306.h: Ця бібліотека використовується для роботи з OLED дисплеєм SSD1306 від Adafruit. Вона дозволяє відображати текст, графіку та інші елементи на дисплеї.

- A

h: Ця бібліотека використовується для роботи з модулем часу DS1302. Вона дозволяє отримувати поточний час з модуля та встановлювати новий час.

h: Бібліотека Adafruit_BMP280 використовується для роботи з датчиком тиску та температури BMP280. Вона дозволяє отримувати дані про тиск та температуру з датчика.

Використання цих бібліотек дозволило ефективно і зручно керувати різними аспектами метеостанції, такими як вимірювання температури, тиску, вологості та відображення цих даних на OLED дисплеї. Також, використання цих бібліотек спростило процес програмування та забезпечило стабільну та надійну роботу пристрою.

Варто зауважити, що програма та метеостанція не мають авторизації, бо як, я вважаю, не потрібно [illegible] програма та пристрій мають бути [illegible] у використанні [illegible] користувачем [illegible] має бути функція заміни датчиків та компонентів.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

У цьому розділі детально розглядається процес інтеграції метеостанції в промислову експлуатацію. Метеостанція може бути використана в різних галузях промисловості та ділових приміщеннях, де контроль погодних умов є важливим аспектом безпеки та ефективності виробництва.

Перед впровадженням системи необхідно провести аналіз потреб бізнесу або виробництва для визначення оптимального місця розташування метеостанції. Важливо врахувати фактори, такі як тип будівлі, географічне положення та специфічні вимоги щодо метеоданих.

Після вибору місця розташування проводиться процес конфігурації та встановлення метеостанції. Це включає в себе монтаж сенсорів, підключення до електромережі або інших джерел живлення, а також налаштування програмного забезпечення для збору та аналізу даних. Крім того, важливо забезпечити необхідну підтримку та навчання персоналу, який буде відповідати за експлуатацію та моніторинг метеостанції. Це може включати навчання з використання програмного забезпечення, інструкції щодо обслуговування обладнання та документацію з процедур у разі виникнення проблем.

Для перегляду короткої довідки про програму було розроблено застосунок з інформацією про авторське право. Зазначено університет, кафедру, тему, спеціальність та інша інформація розробника програмного забезпечення.

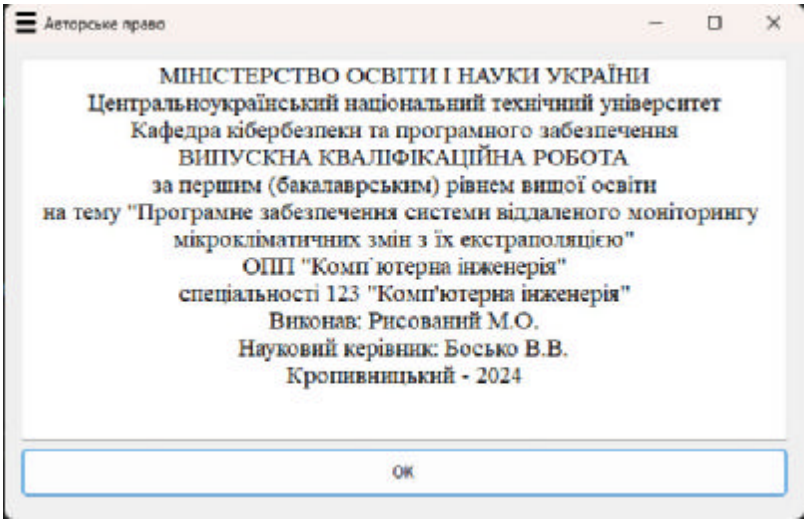

Рисунок 5.1 – Вікно розробника ПЗ

Після встановлення метеостанція готова до використання. Вона забезпечує надійний та точний збір метеоданих, які можуть використовуватися для прийняття рішень щодо безпеки, планування виробничих процесів та оптимізації ресурсів. Результати впровадження системи можуть бути оцінені з точки зору зменшення ризиків, підвищення продуктивності та забезпечення ефективного управління промисловими процесами.

Метеостанції успішно використовуються в різних галузях та сферах діяльності. Метеостанції допомагають сільськогосподарським підприємствам в оптимізації виробничих процесів шляхом забезпечення точних даних про погоду та кліматичні умови. Вони дозволяють вчасно виявляти погодні аномалії, встановлювати оптимальні терміни посівів і збору врожаю, а також вести моніторинг захворювань та шкідників. Метеостанції використовуються для збору даних про клімат та погодні явища для наукових досліджень, спостережень та аналізу змін клімату. Метеостанції допомагають управляти логістичними процесами, шляхом надання інформації про погодні умови на дорогах, морських та повітряних маршрутах. Це дозволяє зменшити ризики аварій та затримок, а також оптимізувати маршрути та час доставки. У енергетичній промисловості метеостанції використовуються для

прогнозування виробництва енергії, оптимізації роботи вітрових та сонячних електростанцій, а також для моніторингу та прогнозу споживання енергії.

Роль метеостанцій у застосуванні полягає в забезпеченні точних та надійних даних про погоду та клімат, що дозволяє приймати обґрунтовані рішення щодо безпеки, ефективності та оптимізації різних процесів у вищезазначених галузях.

6. ОСНОВНІ ВИСНОВКИ

У ході виконання випускної кваліфікаційної роботи була розроблена метеостанція на базі мікроконтролера Arduino UNO. Основними компонентами програмного забезпечення є сенсори температури та вологості (DHT11) та датчик атмосферного тиску та температури (BMP280), також альтернативним датчиком температури та вологості був GY-21.

У процесі розробки було розглянуто велику кількість альтернативних метеостанцій, що охоплюють широкий спектр цінових категорій. Починаючи від високоякісних та дорогих моделей, таких як Vantage Pro 2, які відрізняються високою точністю та надійністю вимірювань, і закінчуючи найдешевшими моделями, доступними для широкого кола користувачів. Метеостанції класу Vantage Pro 2 від компанії Davis Instruments є одними з найпопулярніших та надійних на ринку. Вони володіють широким функціоналом, включаючи вимірювання температури, вологості, атмосферного тиску, швидкості вітру та напрямку вітру, а також можуть бути додатково комплектовані різноманітними сенсорами та аксесуарами.

Однак існують також доступніші альтернативи, такі як метеостанції з вбудованими датчиками температури та вологості, що працюють на базі Arduino або Raspberry Pi. Ці моделі можуть бути менш коштовними у вигляді придбання та експлуатації, але все ще забезпечують досить точні вимірювання для багатьох застосувань. Вибір конкретної метеостанції залежить від потреб користувача, його бюджету та вимог до точності та функціональності пристрою.

Під час досліджень виявлено, що датчик BMP280 продемонстрував більшу точність та стабільність вимірювань у порівнянні з DHT11, рекомендується використання датчика BMP280, так як завжди потрібна висока точність даних про температуру та атмосферний тиск.

Метеостанція може бути використана в різних галузях, таких як сільське господарство, енергетика, транспорт, будівництво та наукові дослідження. Вона дозволяє приймати обґрунтовані рішення щодо оптимізації виробничих процесів, планування та управління ресурсами, а також моніторингу та прогнозування погодних явищ. Метеостанція обладнана дисплеєм OLED, який використовується для відображення зібраних даних про погоду. Дисплей OLED забезпечує чітке та яскраве відображення інформації про температуру, вологість, час та дату, що робить користування метеостанцією простим та зручним для звичайного користувача.

Загалом, виконана робота дозволила створити функціональну та ефективну метеостанцію, яка може бути успішно використана в різних сферах діяльності для збору та аналізу даних про погоду та кліматичні умови.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

vantage Pro2 – Davis Instruments.

Professional Weather Center model 01036

V40-PRO Complete Personal Remote Monitoring Weather Station

– V40 Pro Complete Personal Remote Monitoring

e

t- Метеостанція Met Pro Campbell

a

ĐHT11

m

b

O

BMP280

ŠCD1602

D

Дисплей

Матеріал Вікіпедії [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/Електронний_каталог

Денісова О. О. Інформаційні системи і технології в юридичній діяльності : навч. посіб. / Денісова О. О. – К. : КНЕУ, 2003. – 315 с.

Банзі М., Шіло М. Make: Getting Started with Arduino. — Sebastopol: Maker Media, Inc., 2014. — 262 с.

Блюм Дж. Exploring Arduino: Tools and Techniques for Engineering Wizardry. — Indianapolis: Wiley, 2013. — 352 с.

Боксолл Дж. Arduino Workshop, 2nd Edition: A Hands-on Introduction with 65 Projects. — San Francisco: No Starch Press, 2019. — 392 с.

Характеристики

Purdum, Jack J. Beginning C for Arduino: Learn C Program. — New York: Apress, 2012. — 280 p.

Craft, Brock. Arduino Projects For Dummies. — Hoboken, NJ: For Dummies, 2013. — 408 p.

Смірнов С.А./ Проектування комп’ютерних систем та мереж./ Смірнов С.А., Смірнов О.А., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. / Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

Смірнов С.А., Дрєєва Г.М. / Системне програмне забезпечення. Методичні рекомендації до виконання лабораторних робіт з предмету «Системне Програмне забезпечення/ ЦНТУ, 2022 – 160 с.

Смірнов С.А./ Розробка методу передтестової компіляції й розподілу доступу./ Смірнов С.А., Смірнов О.А., Коваленко О.В., Коваленко А.С. / Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницький. 19-20 квітня 2018р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215

Бормен Томас Г. Вступ до алгоритмів: Переклад з англійської третього видання [укр.] = Introduction to algorithms : Third Edition : [пер. з англ.] / Томас Г. Кормен, Чарлз Е. Лейзерсон, Роналд Л. Рівест, Кліфорд Стайн, — К: К.І.С., 2019. — 1288 с.

Грокаючи алгоритми. Ілюстрований посібник для програмістів та допитливих [Текст] / Е. Маттес ; Пер. з англ. О. Бєлова: Львів : Вид-во Старого Лева, 2021, 600 с.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms. The MIT Press. 2022 1677 с.

Marcello La Rocca. Advanced Algorithms and Data Structures. Manning Publications Co. 2021. 769 с.

John Paul Mueller, Luca Massaron. Algorithms for Dummies. John Wiley & Sons, Inc. 2022. 451 с.

Tom Jenkyns, Ben Stephenson. Fundamentals of Discrete Math for Computer Science. Springer. 2018. 514 с.

ндруник В.А., Висоцька В.А., Пасічник В.В., Чирун Л.Б., Чирун Л.В. Чисельні методи в комп‘ютерних науках: навчальний посібник – Львів: Видавництво «Новий світ – 2000», 2020. – 470 с.

адачин В. М. Чисельні методи : навчальний посібник – Х. : Вид. ХНЕУ ім. С. Кузнеця, 2014. – 180 с.

ason Turner. C++ Best Practices. ISBN 979-8690792589. Independently published. 2022. 143 с.

zejla Medjedovic, Emin Tahirovic. Algorithms and Data Structures for Massive Datasets. Manning Publications Co. 2022. 280 с.

teven C. Chapra, David E. Clough. Applied Numerical Methods with Python for Engineers and Scientists. McGraw Hill LLC. 2022. 1474 с.

assimo Bertaccini. Cryptography Algorithms. Packt Publishing. 2022. 358 с.

o

n 42. Arduino. Arduino Uno. URL: https://doc.arduino.ua/hardware/Uno

ehonathan Sharvit. Data Oriented Programming. Manning Publications. 2022. 447

a

c.

t

лгоритми та методи обчислень: система дистанційної освіти ЦНТУ : веб-сайт.

h

– Режим доступу: https://moodle.kntu.kr.ua/course/view.php?id=738 . - Назва з

a

екрана.

n

artlett J. Programming from the Ground Up. — http://

www.freebookcentre.net/ComputerScienceBooksDownload/Programming -

S

fromthe-Ground-Up-(J.-Bartlett).html

a

Світ електроних схем. [Електронний ресурс] – Режим доступу:

n

d

артін Р. Чиста архітектура. – Фабула, 2019. – 368 с

e

.

арарака В.Д. Архітектура комп'ютерних систем: навчальний посібник. – Житомир: ЖДТУ, 2018. –383с

учасні напрямки комп’ютерної та мікропроцесорної техніки Розділ 1. Основні тенденції розвитку комп'ютерної і мікропроцесорної техніки. Розділ 2 Характеристики ARM і Cortex процесорів: конспект лекцій. [Електронний ресурс]: для студ. спеціальності 171 Електроніка, спеціалізації «Електронні компоненти та системи» /Т. О. Терещенко, Ю.С. Ямненко; КПІ ім. Ігоря Сікорського; уклад,– Електронні текстові данні 1 файл: 5,248 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2020. – 68 с.

айт Української команди розподілених обчислень.– Режим доступу:

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу № 131-02 від 01.04.2024 року, видане на кафедрі кібербезпеки та програмного забезпечення.

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

- Збір та реєстрацію даних про мікрокліматичні параметри, що дозволяє в реальному часі контролювати та відстежувати зміни в середовищі.
- Аналіз та обробку отриманих даних.
- Відображення інформації на дисплеї для візуалізації даних та надання користувачеві зрозумілої інформації про поточний стан середовища простий, інтуїтивно зрозумілий інтерфейс.
- Логування даних у окремий файл з метою відстеження змін у середовищі за певний час, що дозволяє зберігати історію змін для подальшого аналізу та використання.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel Core i7/16 ГБ DDR4 /1 Tb/ GeForce 1060 3GB або сумісні з ним.

5.8.2 Мова програмування

Програму розроблено на мові програмування Arduino.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД.

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 65 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 23.05.2024 р.

9.2 Подання кваліфікаційної бакалаврської роботи на захист 04.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи

за першим (бакалаврським) рівнем вищої освіти

_____ В.В. Босько

Програмне забезпечення системи віддаленого моніторингу мікрокліматичних змін з їх екстраполяцією

Лістинг програми

Код документу 12

Загальна кількість аркушів: 22

Літера: РП

Кропивницький – 2024 року

Лістинг програми BMP280Clock.ini:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_BMP280.h> // Включення необхідних бібліотек

#include <RtcDS1302.h> // Включення бібліотеки для роботи з модулем часу DS1302

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET    -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); // Ініціалізація дисплею SSD1306

#define BMP_SCK  (13)
#define BMP_MISO (12)
#define BMP_MOSI (11)
#define BMP_CS   (10)

Adafruit_BMP280 bmp(BMP_CS, BMP_MOSI, BMP_MISO, BMP_SCK); // Ініціалізація датчика BMP280

ThreeWire myWire(5, 6, 4);
RtcDS1302<ThreeWire> Rtc(myWire);  // Створюємо об'єкт Rtc для роботи з модулем часу DS1302

// Ініціалізація зовнішнього модуля часу DS1302 через 3 провідники

void setup() {
  Serial.begin(9600); // Ініціалізація з'єднання з монітором серійного порту
  RtcDateTime now = Rtc.GetDateTime(); // Отримання поточного часу від модуля DS1302

  if (!bmp.begin()) { // Перевірка успішного підключення датчика BMP280
    Serial.println("Could not find a valid BMP280 sensor, check wiring!"); // Виведення повідомлення про помилку у випадку невдалого підключення
    while (1); // Зациклювання програми в разі виникнення помилки
  }

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Перевірка успішного ініціалізації дисплею SSD1306
    Serial.println(F("SSD1306 allocation failed")); // Виведення повідомлення про помилку
    for (;;); // Зациклювання програми в разі виникнення помилки
  }

  display.clearDisplay(); // Очищення дисплею перед початком виводу даних
}

void loop() {
  delay(1000); // Затримка 1 секунда

  float temperature = bmp.readTemperature(); // Отримання значення температури від датчика BMP280
  float pressure = bmp.readPressure() / 100.0; // Отримання значення тиску в атмосферах від датчика BMP280
  RtcDateTime now = Rtc.GetDateTime(); // Отримання поточного часу від модуля DS1302

  display.clearDisplay(); // Очищення дисплею перед виведенням нових даних
  display.setTextSize(1); // Встановлення розміру шрифту
  display.setTextColor(SSD1306_WHITE); // Встановлення кольору тексту
  display.setCursor(0, 0); // Встановлення позиції курсору для виводу тексту на дисплей

  // Виведення даних температури та тиску на дисплей
```

```
  display.print("Temperature: ");
  display.print(temperature);
  display.println("°C");
  display.print("Pressure: ");
  display.print(pressure);
  display.println(" hPa");

  // Виведення поточної дати та часу на дисплей
  display.print("Date: ");
  display.print(now.Month());
  display.print("/");
  display.print(now.Day());
  display.print("/");
  display.print(now.Year());
  display.println(" ");
  display.print("Time: ");
  display.print(now.Hour());
  display.print(":");
  display.print(now.Minute());
  display.print(":");
  display.print(now.Second());

  display.display(); // Відображення зображення на дисплеї

  // Виведення даних температури, тиску, дати та часу через монітор серійного порту
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println("°C");
  Serial.print("Pressure: ");
  Serial.print(pressure);
  Serial.println(" hPa");
  Serial.print("Date: ");
  Serial.print(now.Month());
  Serial.print("/");
  Serial.print(now.Day());
  Serial.print("/");
  Serial.print(now.Year());
  Serial.print(" ");
  Serial.print("Time: ");
  Serial.print(now.Hour());
  Serial.print(":");
  Serial.print(now.Minute());
  Serial.print(":");
  Serial.println(now.Second());
}
```


Лістинг програми BMP280Clock_2.ini:

```
#include <Wire.h> // Бібліотека для роботи з шиною I2C
#include <Adafruit_GFX.h> // Загальна бібліотека для графічного інтерфейсу
#include <Adafruit_SSD1306.h> // Бібліотека для керування OLED дисплеєм SSD1306
#include <Adafruit_BMP280.h> // Бібліотека для роботи з датчиком тиску BMP280
#include <RtcDS1302.h> // Бібліотека для роботи з годинником реального часу DS1302

// Константи для розміру дисплея
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

// Константа для визначення контакту reset дисплея (у цьому випадку не підключено)
#define OLED_RESET  -1

// Створення об'єкта дисплея
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// Константи для визначення контактів датчика тиску BMP280
```

```
#define BMP_SCK (13)
#define BMP_MISO (12)
#define BMP_MOSI (11)
#define BMP_CS  (10)

// Створення об'єкта датчика тиску
Adafruit_BMP280 bmp(BMP_CS, BMP_MOSI, BMP_MISO, BMP_SCK);

// Об'єкт для роботи з годинником реального часу (використовується шина I2C з іншими номерами контактів)
ThreeWire myWire(5, 6, 4);
RtcDS1302<ThreeWire> Rtc(myWire);

// Піктограми для датчиків та часу в форматі бітових масивів
const unsigned char PROGMEM temperature_icon[] =
{ B00011000, B00111100, B00111100, B00011000, B00011000, B00011000, B00011000, B00011000 };

const unsigned char PROGMEM pressure_icon[] =
{ B00000000, B00111100, B01000010, B01000010, B01000010, B01000010, B00111100, B00000000 };

const unsigned char PROGMEM clock_icon[] =
{ B00000000, B00011000, B00100100, B01000010, B01000010, B01000010, B00111100, B00000000 };

const unsigned char PROGMEM calendar_icon[] =
{ B00000000, B00011000, B00100100, B01000010, B01000010, B01000010, B00111100, B00000000 };


void setup() {
  Serial.begin(9600); // Ініціалізація послідовного порту для виведення даних
  Rtc.Begin(); // Ініціалізація годинника реального часу
  if (!bmp.begin()) {
    Serial.println("Не знайдено датчик BMP280, перевірте підключення!");
    while (1); // Нескінчений цикл у разі помилки
  }
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("Помилка виділення пам'яті для SSD1306"));
    for (;;); // Нескінчений цикл у разі помилки
  }
  display.clearDisplay(); // Очищення дисплея
}

void loop() {
    delay(1000); // Затримка на 1 секунду

  // Зчитування показників датчика тиску
  float temperature = bmp.readTemperature();
  float pressure = bmp.readPressure() / 100.0; // Перетворення тиску в hPa

  // Отримання даних про дату та час з годинника реального часу
  RtcDateTime now = Rtc.GetDateTime();

  // Очищення дисплея перед оновленням даних
  display.clearDisplay();

  // Встановлення розміру та кольору шрифту
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);

  // Відображення піктограми та температури
  display.drawBitmap(5, 5, temperature_icon, 16, 16, SSD1306_WHITE);
  display.setCursor(25, 5);
  display.print(temperature);
  display.println("C");

  // Відображення піктограми та тиску
```

```
  display.drawBitmap(60, 5, pressure_icon, 16, 16, SSD1306_WHITE);
  display.setCursor(80, 5);
  display.print(pressure);
  display.println(" hPa");

  // Відображення піктограми та часу
  display.drawBitmap(5, 30, clock_icon, 16, 16, SSD1306_WHITE);
  display.setCursor(25, 30);
  display.print(now.Hour());
  display.print(":");
  display.print(now.Minute());
  display.print(":");
  display.println(now.Second());

  // Відображення піктограми та дати
  display.drawBitmap(60, 30, calendar_icon, 16, 16, SSD1306_WHITE);
  display.setCursor(80, 30);
  display.print(now.Month());
  display.print("/");
  display.print(now.Day());
  display.print("/");
  display.println(now.Year());

  // Відображення даних на дисплеї
  display.display();

  // Виведення даних в послідовний порт для моніторингу
  Serial.print("Температура: ");
  Serial.print(temperature);
  Serial.println("°C");
  Serial.print("Тиск: ");
  Serial.print(pressure);
  Serial.println(" hPa");
  Serial.print("Дата: ");
  Serial.print(now.Month());
  Serial.print("/");
  Serial.print(now.Day());
  Serial.print("/");
  Serial.print(now.Year());
  Serial.print(" ");
  Serial.print("Час: ");
  Serial.print(now.Hour());
  Serial.print(":");
  Serial.print(now.Minute());
  Serial.print(":");
  Serial.println(now.Second());
}
```


Лістинг програми DHT_1.ini:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>
#include <RtcDS1302.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET  -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define DHTPIN 7    // Пін підключення датчика DHT11
#define DHTTYPE DHT11 // Тип датчика (DHT11)
DHT dht(DHTPIN, DHTTYPE); // Створюємо об'єкт dht для роботи з датчиком DHT11

ThreeWire myWire(5, 6, 4);
RtcDS1302<ThreeWire> Rtc(myWire); // Створюємо об'єкт Rtc для роботи з модулем
часу DS1302
```

```
void setup() {
  Serial.begin(9600); // Ініціалізація Serial Monitor
  dht.begin();     // Ініціалізація датчика DHT11

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }

  display.clearDisplay();
}

void loop() {
  delay(1000);

  RtcDateTime now = Rtc.GetDateTime();
  float humidity = dht.readHumidity();    // Зчитуємо вологість
  float temperature = dht.readTemperature(); // Зчитуємо температуру в градусах
Цельсія

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);

  // Виводимо температуру і вологість
  display.print("Вологість: ");
  display.print(humidity);
  display.print("% ");
  display.print("Температура: ");
  display.print(temperature);
  display.println("C");

  // Виводимо дату і час
  display.print("Дата: ");
  display.print(now.Month());
  display.print("/");
  display.print(now.Day());
  display.print("/");
  display.print(now.Year());
  display.print(" ");
  display.print("Час: ");
  display.print(now.Hour());
  display.print(":");
  display.print(now.Minute());
  display.print(":");
  display.print(now.Second());

  display.display();

  // Виводимо всю інформацію також в Serial Monitor
  Serial.print("Вологість: ");
  Serial.print(humidity);
  Serial.print("% Температура: ");
  Serial.print(temperature);
  Serial.println("C");
  Serial.print("Дата: ");
  Serial.print(now.Month());
  Serial.print("/");
  Serial.print(now.Day());
  Serial.print("/");
  Serial.print(now.Year());
  Serial.print(" ");
  Serial.print("Час: ");
  Serial.print(now.Hour());
  Serial.print(":");
  Serial.print(now.Minute());
  Serial.print(":");
```

```
  Serial.println(now.Second());
}
```

Лістинг програми DHT_2.ini:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>
#include <RtcDS1302.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET  -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define DHTPIN 7    // Пін підключення датчика DHT11
#define DHTTYPE DHT11 // Тип датчика (DHT11)
DHT dht(DHTPIN, DHTTYPE); // Створюємо об'єкт dht для роботи з датчиком DHT11

ThreeWire myWire(5, 6, 4);
RtcDS1302<ThreeWire> Rtc(myWire); // Створюємо об'єкт Rtc для роботи з модулем
часу DS1302

// Значки в форматі бітових масивів (bitmap) для піктограм
const unsigned char icon_temperature[] PROGMEM = {
  B00011000,
  B00111100,
  B01111110,
  B01000010,
  B01000010,
  B01000010,
  B01111110,
  B01111110,
  B01111110,
  B01000010,
  B01000010,
  B01111110
};

const unsigned char icon_humidity[] PROGMEM = {
  B00000000,
  B00011000,
  B00111100,
  B01000010,
  B01111110,
  B11111110,
  B11111110,
  B01000010,
  B11111110,
  B01000010,
  B01000010,
  B11111110
};

const unsigned char icon_clock[] PROGMEM = {
  B00011000,
  B00111100,
  B01111110,
  B01000010,
  B01011010,
  B01000110,
  B01000110,
  B01011010,
  B01000010,
  B01111110,
  B00111100,
  B00011000
```

```
};

const unsigned char icon_calendar[] PROGMEM = {
  B01111110,
  B01111110,
  B01111110,
  B01111110,
  B01111110,
  B01111110,
  B01111110,
  B01111110,
  B01111110,
  B01111110,
  B01111110,
  B01111110
};

void setup() {
  Serial.begin(9600); // Ініціалізація Serial Monitor
  dht.begin();     // Ініціалізація датчика DHT11


  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }

  display.clearDisplay();
}

void loop() {
  delay(1000);

    RtcDateTime now = Rtc.GetDateTime();
  float humidity = dht.readHumidity();    // Зчитуємо вологість
  float temperature = dht.readTemperature(); // Зчитуємо температуру в градусах
Цельсія

  display.clearDisplay();
  display.setTextSize(1); // Встановлюємо стандартний розмір тексту
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);

  // Відображення піктограм часу та дати зверху
  display.drawBitmap(0, 0, icon_clock, 12, 12, SSD1306_WHITE);
  display.setCursor(15, 0);
  display.print(now.Hour());
  display.print(":");
  display.print(now.Minute());
  display.print(":");
  display.println(now.Second());

  display.drawBitmap(64, 0, icon_calendar, 12, 12, SSD1306_WHITE);
  display.setCursor(79, 0);
  display.print(now.Month());
  display.print("/");
  display.print(now.Day());
  display.print("/");
  display.println(now.Year());

  // Відображення значень вологості та температури посередині
  display.drawBitmap(0, 24, icon_humidity, 12, 12, SSD1306_WHITE);
  display.setCursor(15, 24);
  display.print(humidity);
  display.println("%");

  display.drawBitmap(64, 24, icon_temperature, 12, 12, SSD1306_WHITE);
  display.setCursor(79, 24);
  display.print(temperature);
```

```
  display.println("C");

  display.display();

  // Виведення всієї інформації також в Serial Monitor
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print("% Temperature: ");
  Serial.print(temperature);
  Serial.println("C");
  Serial.print("Date: ");
  Serial.print(now.Month());
  Serial.print("/");
  Serial.print(now.Day());
  Serial.print("/");
  Serial.print(now.Year());
  Serial.print(" ");
  Serial.print("Time: ");
  Serial.print(now.Hour());
  Serial.print(":");
  Serial.print(now.Minute());
  Serial.print(":");
  Serial.println(now.Second());
}
```

Wire.cpp:

```
extern "C" {
  #include <stdlib.h>
  #include <string.h>
  #include <inttypes.h>
  #include "utility/twi.h"
}

#include "Wire.h"

// Ініціалізація змінних класу ////////////////////////////////////////////////

uint8_t TwoWire::rxBuffer[BUFFER_LENGTH];
uint8_t TwoWire::rxBufferIndex = 0;
uint8_t TwoWire::rxBufferLength = 0;

uint8_t TwoWire::txAddress = 0;
uint8_t TwoWire::txBuffer[BUFFER_LENGTH];
uint8_t TwoWire::txBufferIndex = 0;
uint8_t TwoWire::txBufferLength = 0;

uint8_t TwoWire::transmitting = 0;
void (*TwoWire::user_onRequest)(void);
void (*TwoWire::user_onReceive)(int);

// Конструктори ///////////////////////////////////////////////////////////////

TwoWire::TwoWire()
{
}

// Публічні методи
//////////////////////////////////////////////////////////////

void TwoWire::begin(void)
{
  rxBufferIndex = 0;
  rxBufferLength = 0;

  txBufferIndex = 0;
  txBufferLength = 0;

  twi_init();
```

```
  twi_attachSlaveTxEvent(onRequestService); // має існувати стандартний
зворотний виклик
  twi_attachSlaveRxEvent(onReceiveService); // має існувати стандартний
зворотний виклик
}

void TwoWire::begin(uint8_t address)
{
  begin();
  twi_setAddress(address);
}

void TwoWire::begin(int address)
{
  begin((uint8_t)address);
}

void TwoWire::end(void)
{
  twi_disable();
}

void TwoWire::setClock(uint32_t clock)
{
  twi_setFrequency(clock);
}

/***
 * Встановлює тайм-аут TWI.
 *
 * Це обмежує максимальний час очікування апаратного TWI. Якщо пройшло більше
часу, шина припускається
 * залишається заблокованою (наприклад, через помилки, що виникають від шуму або
несправних рабів), і транзакція переривається.
 * Опціонально, апарат TWI також скидається, що може бути необхідно для
подальших транзакцій,
 * успішно завершити в деяких випадках (зокрема, коли шум змусив апарат TWI
думати, що є другий
 * майстер, який взяв шину).
 *
 * Коли спрацьовує тайм-аут, встановлюється прапор, який можна запитати за
допомогою `getWireTimeoutFlag ()` і очищається
 * коли викликається `clearWireTimeoutFlag ()` або `setWireTimeoutUs ()`.
 *
 * Зауважте, що цей тайм-аут також може спрацьовувати під час очікування
розтягування годин або очікування другого майстра
 * завершити свою транзакцію. Тому переконайтеся, що тайм-аут адаптується для
врахування цих випадків, якщо потрібно.
 * Типовим тайм-аутом буде 25 мс (це максимальне розтягування годин, дозволене
протоколом SMBus),
 * але (набагато) коротші значення зазвичай також працюватимуть.
 *
 * У майбутньому тайм-аут буде ввімкнено за замовчуванням, тому якщо вам
потрібно вимкнути тайм-аут, то рекомендується вимкнути його за замовчуванням за
допомогою `setWireTimeoutUs (0)`, навіть якщо це зараз
 * за замовчуванням.
 *
 * @param timeout значення тайм-ауту в мікросекундах, якщо нуль, тоді перевірка
тайм-ауту вимкнена
 * @param reset_with_timeout якщо true, тоді інтерфейс TWI автоматично
скидається при тайм-ауті
 *                           якщо false, тоді інтерфейс TWI не скидається при
тайм-ауті

 */
void TwoWire::setWireTimeout(uint32_t timeout, bool reset_with_timeout){
  twi_setTimeoutInMicros(timeout, reset_with_timeout);
}
```

```
/***
 * Повертає прапор тайм-ауту TWI.
 *
 * @return true, якщо тайм-аут стався після того, як прапор був останній раз
очищений.
 */
bool TwoWire::getWireTimeoutFlag(void){
  return(twi_manageTimeoutFlag(false));
}

/***
 * Очищає прапор тайм-ауту TWI.
 */
void TwoWire::clearWireTimeoutFlag(void){
  twi_manageTimeoutFlag(true);
}

uint8_t TwoWire::requestFrom(uint8_t address, uint8_t quantity, uint32_t
iaddress, uint8_t isize, uint8_t sendStop)
{
  if (isize > 0) {
  // відправити внутрішню адресу; цей режим дозволяє відправляти повторні
запуски для доступу
  // до внутрішніх регістрів деяких пристроїв. Ця функція виконується апаратним
  // модулем TWI на інших процесорах (наприклад, регістрами TWI_IADR та TWI_MMR
Due)

  beginTransmission(address);

  // максимальний розмір внутрішньої адреси - 3 байти
  if (isize > 3){
    isize = 3;
  }

  // записати адресу внутрішнього регістра - спочатку найбільш значущий байт
  while (isize-- > 0)
    write((uint8_t)(iaddress >> (isize*8)));
  endTransmission(false);
  }

  // обмежити до довжини буфера
  if(quantity > BUFFER_LENGTH){
    quantity = BUFFER_LENGTH;
  }
  // виконати блокуюче читання в буфер
  uint8_t read = twi_readFrom(address, rxBuffer, quantity, sendStop);
  // встановити змінні ітератора rx буфера
  rxBufferIndex = 0;
  rxBufferLength = read;

  return read;
}

uint8_t TwoWire::requestFrom(uint8_t address, uint8_t quantity, uint8_t
sendStop) {
      return requestFrom((uint8_t)address, (uint8_t)quantity, (uint32_t)0,
(uint8_t)0, (uint8_t)sendStop);
}

uint8_t TwoWire::requestFrom(uint8_t address, uint8_t quantity)
{
  return requestFrom((uint8_t)address, (uint8_t)quantity, (uint8_t)true);
}

uint8_t TwoWire::requestFrom(int address, int quantity)
{
  return requestFrom((uint8_t)address, (uint8_t)quantity, (uint8_t)true);
}
```

```
uint8_t TwoWire::requestFrom(int address, int quantity, int sendStop)
{
  return requestFrom((uint8_t)address, (uint8_t)quantity, (uint8_t)sendStop);
}

void TwoWire::beginTransmission(uint8_t address)
{
  // позначити, що ми передаємо
  transmitting = 1;
  // встановити адресу цільового раба
  txAddress = address;
  // скинути ітератори буфера передачі
  txBufferIndex = 0;
  txBufferLength = 0;
}

void TwoWire::beginTransmission(int address)
{
  beginTransmission((uint8_t)address);
}

//
//    Спочатку 'endTransmission' була функцією f (void).
//    Вона була змінена для прийняття одного параметра, що вказує
//    чи має виконатися STOP на шині.
//    Виклик endTransmission (false) дозволяє малюнку
//    виконати повторний запуск.
//
//    ПОПЕРЕДЖЕННЯ: В бібліотеці нічого не відстежується, чи
//    чи закінчено було правильно переможення автобуса з STOP. Це
//    дуже можливо залишити шину в увішному стані, якщо
//    немає виклику endTransmission (true). Деякі I2C
//    пристрої будуть дивно себе вести, якщо вони не бачать STOP.
//
uint8_t TwoWire::endTransmission(uint8_t sendStop)
{
  // передати буфер (блокуюче)
  uint8_t ret = twi_writeTo(txAddress, txBuffer, txBufferLength, 1, sendStop);
  // скинути ітератори буфера передачі
  txBufferIndex = 0;
  txBufferLength = 0;
  // позначити, що ми закінчили передачу
  transmitting = 0;
  return ret;
}

//    Це забезпечує зворотню сумісність з оригінальним
//    визначенням та очікуваним поведінкою endTransmission
//
uint8_t TwoWire::endTransmission(void)
{
  return endTransmission(true);
}

// має бути викликано в:
// зворотний виклик події передачі раба
// або після beginTransmission (адреса)
size_t TwoWire::write(uint8_t data)
{
  if(transmitting){
  // в режимі майстра передавача
    // не марно, якщо буфер повний
    if(txBufferLength >= BUFFER_LENGTH){
      setWriteError();
      return 0;
    }
    // покласти байт у буфер передачі
    txBuffer[txBufferIndex] = data;
    ++txBufferIndex;
```

```
    // оновлення кількості в буфері
    txBufferLength = txBufferIndex;
  }else{
  // в режимі раба надсилання
    // відповісти майстеру
    twi_transmit(&data, 1);
  }
  return 1;
}

// має бути викликано в:
// зворотний виклик події передачі раба
// або після beginTransmission (адреса)
size_t TwoWire::write(const uint8_t *data, size_t quantity)
{
  if(transmitting){
  // в режимі майстра передавача
    for(size_t i = 0; i < quantity; ++i){
      write(data[i]);
    }
  }else{
  // в режимі раба надсилання
    // відповісти майстеру
    twi_transmit(data, quantity);
  }
  return quantity;
}

// має бути викликано в:
// зворотний виклик події отримання раба
// або після requestFrom (адреса, numBytes)
int TwoWire::available(void)
{
  return rxBufferLength - rxBufferIndex;
}

// має бути викликано в:
// зворотний виклик події отримання раба
// або після requestFrom (адреса, numBytes)
int TwoWire::read(void)
{
  int value = -1;

  // отримати кожен послідовний байт при кожному виклику
  if(rxBufferIndex < rxBufferLength){
    value = rxBuffer[rxBufferIndex];
    ++rxBufferIndex;
  }

  return value;
}

// має бути викликано в:
// зворотний виклик події отримання раба
// або після requestFrom (адреса, numBytes)
int TwoWire::peek(void)
{
  int value = -1;

  if(rxBufferIndex < rxBufferLength){
    value = rxBuffer[rxBufferIndex];
  }

  return value;
}

void TwoWire::flush(void)
{
  // XXX: реалізувати.
```

```
}

// функція, яка викликається за кадром, коли дані отримані
void TwoWire::onReceiveService(uint8_t* inBytes, int numBytes)
{
  // не звертайте уваги, якщо користувач не зареєстрував зворотний виклик
  if(!user_onReceive){
    return;
  }
  // не звертайте увагу, якщо буфер rx використовується операцією майстра
requestFrom()
  // я знаю, що це втрачає дані, але це дозволяє незначну глупоту
  // це означає, що вони можуть ще не прочитати всі дані від майстра
requestFrom()
  if(rxBufferIndex < rxBufferLength){
    return;
  }
  // скопіювати буфер прийому TWI в локальний буфер читання
  // це дозволяє виконувати нові читання паралельно
  for(uint8_t i = 0; i < numBytes; ++i){
    rxBuffer[i] = inBytes[i];
  }
  // встановити змінні ітератора rx
  rxBufferIndex = 0;
  rxBufferLength = numBytes;
  // сповістити користувацьку програму
  user_onReceive(numBytes);
}

// функція, яка викликається за кадром, коли дані запитані
void TwoWire::onRequestService(void)
{
  // не звертайте увагу, якщо користувач не зареєстрував зворотний виклик
  if(!user_onRequest){
    return;
  }
  // скинути ітератори буфера передачі
  // !!! це знищить будь-яку очікуван

// якщо користувач хоче на нього відповісти, він повинен відновити відповідь
  txBufferIndex = 0;
  txBufferLength = 0;
  // запустити користувальницьку програму
  user_onRequest();
}

// Приватні методи //////////////////////////////////////////////////////////////

// Оповіщає інші методи, що їх зворотній виклик не повинен викликатися
void TwoWire::setWriteError(void)
{
}

// Загальна ініціалізація
void TwoWire::initialize(void)
{
  rxBufferIndex = 0;
  rxBufferLength = 0;

  txBufferIndex = 0;
  txBufferLength = 0;

  twi_init();
}
```

Rtc.cpp:

```
#pragma once
```

```
#include <Arduino.h>
#include "RtcUtility.h"
#include "RtcDateTime.h"
#include "ThreeWire.h"


//Адреси регістрів DS1302
const uint8_t DS1302_REG_TIMEDATE   = 0x80;
const uint8_t DS1302_REG_TIMEDATE_BURST = 0xBE;
const uint8_t DS1302_REG_TCR    = 0x90;
const uint8_t DS1302_REG_RAM_BURST = 0xFE;
const uint8_t DS1302_REG_RAMSTART   = 0xc0;
const uint8_t DS1302_REG_RAMEND     = 0xfd;
// розмір оперативної пам'яті змінних
const uint8_t DS1302RamSize = 31;


// Біти регістру керування зарядом DS1302
enum DS1302TcrResistor
{
    DS1302TcrResistor_Disabled = 0,
    DS1302TcrResistor_2KOhm = 0b00000001,
    DS1302TcrResistor_4KOhm = 0b00000010,
    DS1302TcrResistor_8KOhm = 0b00000011,
    DS1302TcrResistor_MASK  = 0b00000011,
};

enum DS1302TcrDiodes
{
    DS1302TcrDiodes_None = 0,
    DS1302TcrDiodes_One      = 0b00000100,
    DS1302TcrDiodes_Two      = 0b00001000,
    DS1302TcrDiodes_Disabled = 0b00001100,
    DS1302TcrDiodes_MASK     = 0b00001100,
};

enum DS1302TcrStatus
{
    DS1302TcrStatus_Enabled  = 0b10100000,
    DS1302TcrStatus_Disabled = 0b01010000,
    DS1302TcrStatus_MASK     = 0b11110000,
};

const uint8_t DS1302Tcr_Disabled = DS1302TcrStatus_Disabled |
DS1302TcrDiodes_Disabled | DS1302TcrResistor_Disabled;

// Регістр і біти зупинки годинника DS1302
const uint8_t DS1302_REG_CH = 0x80; // біт у регістрі секунд
const uint8_t DS1302_CH     = 7;

// Регістр та біти захисту від запису
const uint8_t DS1302_REG_WP = 0x8E;
const uint8_t DS1302_WP = 7;

template<class T_WIRE_METHOD> class RtcDS1302
{
public:
    RtcDS1302(T_WIRE_METHOD& wire) :
        _wire(wire)
    {
    }

    void Begin()
    {
        _wire.begin();
    }

    bool GetIsWriteProtected()
    {
```

```
    uint8_t wp = getReg(DS1302_REG_WP);
    return !!(wp & _BV(DS1302_WP));
}

void SetIsWriteProtected(bool isWriteProtected)
{
    uint8_t wp = getReg(DS1302_REG_WP);
    if (isWriteProtected)
    {
        wp |= _BV(DS1302_WP);
    }
    else
    {
        wp &= ~_BV(DS1302_WP);
    }
    setReg(DS1302_REG_WP, wp);
}

bool IsDateTimeValid()
{
    return GetDateTime().IsValid();
}

bool GetIsRunning()
{
    uint8_t ch = getReg(DS1302_REG_CH);
    return !(ch & _BV(DS1302_CH));
}

void SetIsRunning(bool isRunning)
{
    uint8_t ch = getReg(DS1302_REG_CH);
    if (isRunning)
    {
        ch &= ~_BV(DS1302_CH);
    }
    else
    {
        ch |= _BV(DS1302_CH);
    }
    setReg(DS1302_REG_CH, ch);
}

uint8_t GetTrickleChargeSettings()
{
    uint8_t setting = getReg(DS1302_REG_TCR);
    return setting;
}

void SetTrickleChargeSettings(uint8_t setting)
{
    if ((setting & DS1302TcrResistor_MASK) == DS1302TcrResistor_Disabled)
    {
        // недійсна настройка резистора, встановити на вимкнено
        setting = DS1302Tcr_Disabled;
    }
    else if ((setting & DS1302TcrDiodes_MASK) == DS1302TcrDiodes_Disabled ||
        (setting & DS1302TcrDiodes_MASK) == DS1302TcrDiodes_None)
    {
        // недійсна настройка діодів, встановити на вимкнено
        setting = DS1302Tcr_Disabled;
    }
    else if ((setting & DS1302TcrStatus_MASK) != DS1302TcrStatus_Enabled)
    {
        // недійсна настройка статусу, встановити на вимкнено
        setting = DS1302Tcr_Disabled;
    }

    setReg(DS1302_REG_TCR, setting);
```

```
    }

    void SetDateTime(const RtcDateTime& dt)
    {
        // встановити дату та час
        _wire.beginTransmission(DS1302_REG_TIMEDATE_BURST);

        _wire.write(Uint8ToBcd(dt.Second()));
        _wire.write(Uint8ToBcd(dt.Minute()));
        _wire.write(Uint8ToBcd(dt.Hour())); // лише 24-годинний режим
        _wire.write(Uint8ToBcd(dt.Day()));
        _wire.write(Uint8ToBcd(dt.Month()));

        // RTC Hardware Day of Week is 1-7, 1 = Monday
        // перетворюємо наш день тижня в Rtc день тижня
        uint8_t rtcDow = RtcDateTime::ConvertDowToRtc(dt.DayOfWeek());

        _wire.write(Uint8ToBcd(rtcDow));
        _wire.write(Uint8ToBcd(dt.Year() - 2000));
        _wire.write(0); // без захисту запису, оскільки все це ігнорується, якщо
воно захищено

        _wire.endTransmission();
    }

    RtcDateTime GetDateTime()
    {
        _wire.beginTransmission(DS1302_REG_TIMEDATE_BURST | THREEWIRE_READFLAG);

        uint8_t second = BcdToUint8(_wire.read() & 0x7F);
        uint8_t minute = BcdToUint8(_wire.read());
        uint8_t hour = BcdToBin24Hour(_wire.read());
        uint8_t dayOfMonth = BcdToUint8(_wire.read());
        uint8_t month = BcdToUint8(_wire.read());

        _wire.read();  // викидуємо день тижня, оскільки ми обчислюємо його

        uint16_t year = BcdToUint8(_wire.read()) + 2000;

        _wire.read();  // викидуємо прапорець захисту запису

        _wire.endTransmission();

        return RtcDateTime(year, month, dayOfMonth, hour, minute, second);
    }

    void SetMemory(uint8_t memoryAddress, uint8_t value)
    {
        // адреси пам'яті переплітаються з адресами читання та запису
        // тому нам потрібно обчислити зміщення
        uint8_t address = memoryAddress * 2 + DS1302_REG_RAMSTART;
        if (address <= DS1302_REG_RAMEND)
        {
            setReg(address, value);
        }
    }

    uint8_t GetMemory(uint8_t memoryAddress)
    {
        uint8_t value = 0;
        // адреси пам'яті переплітаються з адресами читання та запису
        // тому нам потрібно обчислити зміщення
        uint8_t address = memoryAddress * 2 + DS1302_REG_RAMSTART;
        if (address <= DS1302_REG_RAMEND)
        {
            value = getReg(address);
        }
        return value;
    }
```

```
    uint8_t SetMemory(const uint8_t* pValue, uint8_t countBytes)
    {
        uint8_t countWritten = 0;

        _wire.beginTransmission(DS1302_REG_RAM_BURST);

        while (countBytes > 0 && countWritten < DS1302RamSize)
        {
            _wire.write(*pValue++);
            countBytes--;
            countWritten++;
        }

        _wire.endTransmission();

        return countWritten;
    }

    uint8_t GetMemory(uint8_t* pValue, uint8_t countBytes)
    {
        uint8_t countRead = 0;

        _wire.beginTransmission(DS1302_REG_RAM_BURST | THREEWIRE_READFLAG);

        while (countBytes > 0 && countRead < DS1302RamSize)
        {
            *pValue++ = _wire.read();
            countRead++;
            countBytes--;
        }

        _wire.endTransmission();

        return countRead;
    }

private:
    T_WIRE_METHOD& _wire;

    uint8_t getReg(uint8_t regAddress)
    {
        _wire.beginTransmission(regAddress | THREEWIRE_READFLAG);
        uint8_t regValue = _wire.read();
        _wire.endTransmission();
        return regValue;
    }

    void setReg(uint8_t regAddress, uint8_t regValue)
    {
        _wire.beginTransmission(regAddress);
        _wire.write(regValue);
        _wire.endTransmission();
    }
};

#include <Adafruit_BMP280.h>

/*!
 * @brief  Конструктор BMP280, використовуючи i2c
 * @param  *theWire
 *         Необов'язковий об'єкт wire
 */
Adafruit_BMP280::Adafruit_BMP280(TwoWire *theWire) {
  _wire = theWire;
  temp_sensor = new Adafruit_BMP280_Temp(this);
  pressure_sensor = new Adafruit_BMP280_Pressure(this);
}
```

```
/*!
 * @brief  Конструктор BMP280, використовуючи апаратний SPI
 * @param  cspin
 *         Номер піна cs
 * @param  theSPI
 *         Необов'язковий об'єкт SPI
 */
Adafruit_BMP280::Adafruit_BMP280(int8_t cspin, SPIClass *theSPI) {
  spi_dev = new Adafruit_SPIDevice(cspin, 1000000, SPI_BITORDER_MSBFIRST,
                                   SPI_MODE0, theSPI);
  temp_sensor = new Adafruit_BMP280_Temp(this);
  pressure_sensor = new Adafruit_BMP280_Pressure(this);
}

/*!
 * @brief  Конструктор BMP280, використовуючи бітбенг SPI
 * @param  cspin
 *         Пін для CS/SSEL
 * @param  mosipin
 *         Пін для MOSI
 * @param  misopin
 *         Пін для MISO
 * @param  sckpin
 *         Пін для SCK
 */
Adafruit_BMP280::Adafruit_BMP280(int8_t cspin, int8_t mosipin, int8_t misopin,
                                 int8_t sckpin) {
  spi_dev = new Adafruit_SPIDevice(cspin, sckpin, misopin, mosipin);
  temp_sensor = new Adafruit_BMP280_Temp(this);
  pressure_sensor = new Adafruit_BMP280_Pressure(this);
}

Adafruit_BMP280::~Adafruit_BMP280(void) {
  if (spi_dev)
    delete spi_dev;
  if (i2c_dev)
    delete i2c_dev;
  if (temp_sensor)
    delete temp_sensor;
  if (pressure_sensor)
    delete pressure_sensor;
}

/*!
 * Ініціалізує сенсор.
 * @param addr
 *        Адреса I2C (за замовчуванням = 0x77)
 * @param chipid
 *        Очікуваний ідентифікатор чіпа (використовується для перевірки
з'єднання).
 * @return Істину, якщо ініціалізація була успішною, в іншому випадку - false.
 */
bool Adafruit_BMP280::begin(uint8_t addr, uint8_t chipid) {
  if (spi_dev == NULL) {
    // Режим I2C
    if (i2c_dev)
      delete i2c_dev;
    i2c_dev = new Adafruit_I2CDevice(addr, _wire);
    if (!i2c_dev->begin())
      return false;
  } else {
    // Режим SPI
    if (!spi_dev->begin())
      return false;
  }

  // перевірка, чи сенсор, тобто ідентифікатор чіпа, правильний
  _sensorID = read8(BMP280_REGISTER_CHIPID);
  if (_sensorID != chipid)
```

```
    return false;

  readCoefficients();
  // write8(BMP280_REGISTER_CONTROL, 0x3F); /* потрібно? */
  setSampling();
  delay(100);
  return true;
}

/*!
 * Встановлює конфігурацію зразка для пристрою.
 * @param mode
 *        Режим роботи сенсора.
 * @param tempSampling
 *        Схема вибірки для температурних вимірювань.
 * @param pressSampling
 *        Схема вибірки для вимірювань тиску.
 * @param filter
 *        Режим фільтрації для застосування (якщо є).
 * @param duration
 *        Тривалість вимірювання.
 */
void Adafruit_BMP280::setSampling(sensor_mode mode,
                                  sensor_sampling tempSampling,
                                  sensor_sampling pressSampling,
                                  sensor_filter filter,
                                  standby_duration duration) {
  if (!_sensorID)
    return; // функцію begin() ще не викликано
  _measReg.mode = mode;
  _measReg.osrs_t = tempSampling;
  _measReg.osrs_p = pressSampling;

  _configReg.filter = filter;
  _configReg.t_sb = duration;

  write8(BMP280_REGISTER_CONFIG, _configReg.get());
  write8(BMP280_REGISTER_CONTROL, _measReg.get());
}

/**************************************************************************/
/*!
    @brief  Записує значення 8 бітів через I2C/SPI
*/
/**************************************************************************/
void Adafruit_BMP280::write8(byte reg, byte value) {
  byte buffer[2];
  buffer[1] = value;
  if (i2c_dev) {
    buffer[0] = reg;
    i2c_dev->write(buffer, 2);
  } else {
    buffer[0] = reg & ~0x80;
    spi_dev->write(buffer, 2);
  }
}

/*!
 *  @brief  Зчитує значення 8 бітів через I2C/SPI
 *  @param  reg
 *          вибраний регістр
 *  @return значення з вибраного регістру
 */
uint8_t Adafruit_BMP280::read8(byte reg) {
  uint8_t buffer[1];
  if (i2c_dev) {
    buffer[0] = uint8_t(reg);
    i2c_dev->write_then_read(buffer, 1, buffer, 1);
  } else {
```

```
    buffer[0] = uint8_t(reg | 0x80);
    spi_dev->write_then_read(buffer, 1, buffer, 1);
  }
  return buffer[0];
}

/*!
 *  @brief  Зчитує значення 16 бітів через I2C/SPI
 */
uint16_t Adafruit_BMP280::read16(byte reg) {
  uint8_t buffer[2];

  if (i2c_dev) {
    buffer[0] = uint8_t(reg);
    i2c_dev->write_then_read(buffer, 1, buffer, 2);
  } else {
    buffer[0] = uint8_t(reg | 0x80);
    spi_dev->write_then_read(buffer, 1, buffer, 2);
  }
  return uint16_t(buffer[0]) << 8 | uint16_t(buffer[1]);
}

uint16_t Adafruit_BMP280::read16_LE(byte reg) {
  uint16_t temp = read16(reg);
  return (temp >> 8) | (temp << 8);
}

/*!
 *   @brief  Зчитує підписане значення 16 бітів через I2C/SPI
 */
int16_t Adafruit_BMP280::readS16(byte reg) { return (int16_t)read16(reg); }

int16_t Adafruit_BMP280::readS16_LE(byte reg) {
  return (int16_t)read16_LE(reg);
}

/*!
 *  @brief  Зчитує значення 24 бітів через I2C/SPI
 */
uint32_t Adafruit_BMP280::read24(byte reg) {
  uint8_t buffer[3];

  if (i2c_dev) {
    buffer[0] = uint8_t(reg);
    i2c_dev->write_then_read(buffer, 1, buffer, 3);
  } else {
    buffer[0] = uint8_t(reg | 0x80);
    spi_dev->write_then_read(buffer, 1, buffer, 3);
  }
  return uint32_t(buffer[0]) << 16 | uint32_t(buffer[1]) << 8 |
         uint32_t(buffer[2]);
}

/*!
 *  @brief  Зчитує заводські коефіцієнти
 */
void Adafruit_BMP280::readCoefficients() {
  _bmp280_calib.dig_T1 = read16_LE(BMP280_REGISTER_DIG_T1);
  _bmp280_calib.dig_T2 = readS16_LE(BMP280_REGISTER_DIG_T2);
  _bmp280_calib.dig_T3 = readS16_LE(BMP280_REGISTER_DIG_T3);

  _bmp280_calib.dig_P1 = read16_LE(BMP280_REGISTER_DIG_P1);
  _bmp280_calib.dig_P2 = readS16_LE(BMP280_REGISTER_DIG_P2);
  _bmp280_calib.dig_P3 = readS16_LE(BMP280_REGISTER_DIG_P3);
  _bmp280_calib.dig_P4 = readS16_LE(BMP280_REGISTER_DIG_P4);
  _bmp280_calib.dig_P5 = readS16_LE(BMP280_REGISTER_DIG_P5);
  _bmp280_calib.dig_P6 = readS16_LE(BMP280_REGISTER_DIG_P6);
  _bmp280_calib.dig_P7 = readS16_LE(BMP280_REGISTER_DIG_P7);
  _bmp280_calib.dig_P8 = readS16_LE(BMP280_REGISTER_DIG_P8);
```

```
  _bmp280_calib.dig_P9 = readS16_LE(BMP280_REGISTER_DIG_P9);
}
```