Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем управління
(повна назва кафедри)

Бакалаврська кваліфікаційна робота
на тему:

«Розробка автоматизованої системи керування мікрокліматом парника на базі мікроконтролера»

Виконав: студент 4-го курсу,
групи 2АКІТ-21б
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва спеціальності)

_____ Михайло КАРАКОЙ
(ім'я та прізвище)

Керівник: д.т.н., професор каф. КСУ

_____ Марія ЮХИМЧУК
(ім'я та прізвище)

«10» 06 2025 р.

Рецензент: к.т.н., доцент каф. АІІТ

_____ Володимир ГАРМАШ
(ім'я та прізвище)

«11» 06 2025 р.

Допущено до захисту
Завідувач кафедри КСУ
д.т.н., проф.
_____ В'ячеслав КОВТУН
(ім'я та прізвище)
«12» 06 2025 р.

Вінниця ВНТУ - 2025 рік

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти І-й (бакалаврський)
Галузь знань – 15 Автоматизація та приладобудування
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Освітньо-професійна програма Інтелектуальні комп'ютерні системи управління

ЗАТВЕРДЖУЮ
Завідувач кафедри КСУ
д.т.н., проф.
_____ В'ячеслав КОВТУН
«04» 05 2025 року

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

Каракою Михайлу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема бакалаврської кваліфікаційної роботи «Розробка автоматизованої системи керування мікрокліматом парника на базі мікроконтролера»

керівник бакалаврської кваліфікаційної роботи Юхимчук Марія Сергіївна, доктор технічних наук, професор

затверджені наказом вищого навчального закладу від "20" 03 2025 року № 97

2. Строк подання студентом бакалаврської кваліфікаційної роботи 10.06.2025 року
3. Джерела розробки.

Бакалаврська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. Кондратьєв І. О. Автоматизація тепличного господарства: монографія. – Київ: КНУБА, 2018. – 192 с.
2. Кошуба Т. М. Системи автоматичного управління мікрокліматом: перспективи впровадження в аграрному секторі // Вісник аграрної науки. – 2021. – №9. – С. 31–35.
3. Pawar D., Kumbhar P., Khot P. Smart greenhouse monitoring system // International Journal of Engineering Sciences & Research Technology. – 2021. – Vol. 10, No. 6. – P. 15–19.
4. Jha K., Doshi A., Patel P. Application of data analytics in smart farming // Computers and Electronics in Agriculture. – 2019. – Vol. 161. – P. 234–242.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) вступ, дослідження об'єкта автоматизації, дослідження систем автоматизації мікроклімату та підходів до автоматизованого керування, проєктування системи керування мікрокліматом парника на базі мікроконтролера дослідження координатора, програмна реалізація системи керування мікрокліматом парника, модельне тестування роботи системи керування мікрокліматом парника.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Структурна діаграма взаємозв'язків між компонентами системи – 1 шт, блок-схема алгоритму робои системи – 1 шт, відображення змодельованих даних – 2 шт, графіки поведінки системи – 5 шт.

1. Консультанти розділів роботи

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

2. **Дата видачі завдання:** «26» 05 2025 року

Календарний план

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1	Дослідження актуальності поставленої задачі	10.05.2025 р.	
2	Дослідження систем автоматизації мікроклімату та підходів до автоматизованого керування	14.05.2025 р.	
3	Проектування системи керування мікрокліматом парника на базі мікроконтролера дослідження координатора	19.05.2025 р.	
4	Програмна реалізація системи керування мікрокліматом парника	20.05.2025 р.	
5	Модельне тестування роботи системи керування мікрокліматом парника	30.05.2025 р.	
6	Апробація результатів дослідження	29.05.2025 р.	
7	Публікації		
8	Оформлення пояснювальної записки, графічного матеріалу і презентації	07.06.2025 р.	
9	Графічні матеріали	08.06.2025 р.	
10	Захист БКР	18.06.2025 р.	

Студент _____ Михайло КАРАКОЙ
(підпис)

Керівник роботи _____ Марія ЮХИМЧУК
(підпис)

АНОТАЦІЯ

Бакалаврська робота складається зі 107 сторінок формату А4, на яких представлено 19 рисунків, 3 таблиці, список використаних джерел містить 55 найменувань.

Мета роботи – розробка автоматизованої системи керування мікрокліматом парника на базі мікроконтролера, що забезпечує підтримання оптимальних умов для росту рослин з урахуванням змін зовнішнього середовища.

У теоретично-аналітичній частині проведено дослідження об’єкта автоматизації, параметрів мікроклімату, сучасних платформ і рішень для контролю та регулювання середовища, а також порівняння підходів до автоматизованого керування, включаючи логіку на основі умовних операторів, PID-регулятори та методи машинного навчання.

У практичній частині розроблено архітектуру системи, визначено логіку керування, реалізовано емуляцію роботи системи на мові Python з використанням віртуальних сенсорів, здійснено логування та візуалізацію результатів, а також проведено тестування її роботи при змінних умовах.

Ключові слова: автоматизована система, мікроконтролер, керування мікрокліматом, парник, моделювання, програмна реалізація, візуалізація, тестування.

ABSTRACT

The bachelor's thesis consists of 107 A4-sized pages and includes 19 figures, 3 tables, and a list of 55 references.

The objective of the thesis is to develop an automated greenhouse microclimate control system based on a microcontroller, which ensures the maintenance of optimal conditions for plant growth, taking into account changes in the external environment.

The theoretical and analytical part presents a study of the automation object, microclimate parameters, modern platforms and solutions for environmental monitoring and regulation, as well as a comparison of control approaches, including rule-based logic, PID controllers, and machine learning algorithms.

In the practical part, the system architecture was designed, control logic was defined, a Python-based simulation using virtual sensors was implemented, logging and data visualization were performed, and system behavior under varying conditions was tested.

Keywords: automated system, microcontroller, microclimate control, greenhouse, simulation, software implementation, visualization, testing.

ЗМІСТ

ВСТУП

Актуальність. Зростання попиту на автоматизовані системи у сільському господарстві обумовлюється необхідністю підвищення ефективності використання ресурсів, зменшення впливу людського фактора та забезпечення стабільності виробництва у змінних кліматичних умовах. В умовах глобальних кліматичних змін та зростання вартості енергетичних ресурсів виникає потреба в точному контролі параметрів мікроклімату в захищеному ґрунті, зокрема в парниках, що дозволяє забезпечити стабільність розвитку рослин і підвищити врожайність. Автоматизовані системи керування (АСК) мікрокліматом, побудовані на базі вбудованих систем, зокрема мікроконтролерів, надають змогу створити адаптивні рішення з низьким енергоспоживанням, можливістю локального прийняття рішень та інтеграції в IoT-інфраструктури (Internet of Things – Інтернет речей) [1, 2].

Автоматизація мікрокліматичних процесів у парниках включає контроль температури, вологості, рівня освітлення, вмісту CO_2 та інших параметрів, що впливають на фотосинтетичну активність рослин. Традиційні системи керування, які вимагають постійного ручного втручання, не забезпечують достатньої точності та швидкості реагування на зміну зовнішніх умов. Застосування мікроконтролерів дозволяє реалізувати компактні та гнучкі системи керування з підтримкою зворотного зв’язку, що особливо важливо для малих і середніх аграрних господарств, де питання вартості та енергоефективності мають вирішальне значення [3].

Розробка алгоритмів автоматичного регулювання параметрів мікроклімату на основі даних із сенсорних пристроїв, а також моделювання ефективності таких рішень у віртуальному середовищі, забезпечує можливість створення недорогих, доступних для локального використання систем. При цьому важливу роль відіграє правильний вибір архітектури програмного забезпечення та апаратних компонентів, що впливають на продуктивність, надійність і масштабованість розроблюваного

рішення. Крім того, значну увагу слід приділити дослідженню вже існуючих рішень, зокрема відкритих проектів з вільним програмним забезпеченням, для визначення оптимального підходу до реалізації.

З огляду на наведене, актуальним є дослідження та проектування автоматизованої системи керування мікрокліматом парника із застосуванням мікроконтролера, з урахуванням вимог до точності регулювання, простоти обслуговування, економічної доцільності та можливості подальшої модернізації.

Мета і завдання дослідження. Метою бакалаврської кваліфікаційної роботи є розробка автоматизованої системи керування мікрокліматом парника на базі мікроконтролера, що забезпечує підтримання оптимальних умов для росту рослин з урахуванням змін зовнішнього середовища.

Для досягнення поставленої мети були вирішені наступні завдання:

1) проаналізовано сучасні методи та програмно-апаратні рішення для автоматизованого керування мікрокліматом у захищеному ґрунті;
2) визначено технічні вимоги до функціоналу системи з урахуванням реальних умов експлуатації;
3) розроблено архітектуру та програмну реалізацію модуля керування з урахуванням вимог до масштабованості та енергоефективності;
4) проведено симуляцію та оцінку ефективності алгоритмів регулювання;
5) обґрунтувати доцільність застосування обраного підходу для малих і середніх сільськогосподарських підприємств.

Об’єктом дослідження є процес регулювання параметрів мікроклімату в парникових умовах з використанням автоматизованих систем керування.

Предметом дослідження є методи розробки програмно-апаратних систем керування на базі мікроконтролерів для підтримання стабільного мікроклімату в захищеному ґрунті.

Методи дослідження. У процесі дослідження використовувалися методи системного аналізу, порівняльного аналізу алгоритмів регулювання, математичного

моделювання та симуляції процесів керування, а також інструменти розробки програмного забезпечення для вбудованих систем з відкритим вихідним кодом.

Новизна одержаних результатів полягає в створенні та обґрунтуванні архітектури автоматизованої системи керування мікрокліматом на основі мікроконтролера з використанням адаптивних алгоритмів регулювання та можливістю віртуального тестування системи в середовищі симуляції без фізичного монтажу апаратного забезпечення.

Практичне значення одержаних результатів полягає в тому, що розроблене рішення може бути використане як основа для створення недорогих автономних систем підтримання мікроклімату в тепличних умовах, зокрема в умовах приватного фермерського господарства або малих агропідприємств. Також результати можуть бути використані в освітніх цілях для демонстрації принципів автоматизації в агропромисловому секторі.

1 ДОСЛІДЖЕННЯ ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Темою бакалаврської роботи є розробка автоматизованої системи керування мікрокліматом парника на базі мікроконтролера. Сьогодні, в умовах стрімкого зростання населення планети та зростаючої потреби в продовольстві, стабільне сільськогосподарське виробництво стає надзвичайно актуальним. Важливим аспектом цього процесу є економне використання ресурсів, що робить автоматизацію процесів у тепличному господарстві особливо значущою. Враховуючи зміни кліматичних умов, які впливають на аграрний сектор, а також підвищені вимоги до якості продукції, управління мікрокліматом у захищеному ґрунті стало ключовим завданням для досягнення високої ефективності аграрних технологій.

У рамках цього дослідження об'єктом автоматизації обрано типовий тепличний парник, в якому необхідно підтримувати оптимальні параметри середовища для росту та розвитку рослин. Це включає в себе контроль над такими основними керованими величинами, як температура повітря, вологість, рівень освітленості та циркуляція повітря. Для забезпечення цих параметрів будуть використовуватися відповідні виконавчі механізми, зокрема системи вентиляції, обігріву, зрошення та освітлення. Аналіз фізичних характеристик об'єкта, а також його реакції на зовнішні збурення, дозволить визначити вимоги до точності регулювання, що, в свою чергу, сприятиме формуванню необхідних функціональних можливостей майбутньої автоматизованої системи.

Не менш важливим аспектом дослідження є огляд сучасних підходів до побудови подібних систем на базі мікроконтролерів. Зокрема, будуть розглянуті платформи Arduino, ESP та STM32, які відзначаються доступністю, низьким енергоспоживанням і достатньою обчислювальною потужністю для реалізації необхідних алгоритмів керування. Ці мікроконтролери надають широкий спектр можливостей для інтеграції різних датчиків і модулів, що забезпечують збір даних

про стан мікроклімату [4]. Важливою частиною дослідження стане порівняння технічних рішень, що використовуються в аналогічних системах, а також аналіз переваг та обмежень апаратних і програмних засобів. Це дозволить обґрунтувати вибір оптимальної конфігурації для реалізації системи керування мікрокліматом у парнику.

Завдяки проведеному аналізу та розробці автоматизованої системи, можна очікувати значного підвищення ефективності аграрних технологій, що в свою чергу сприятиме підвищенню врожайності та якості сільськогосподарської продукції. Таким чином, результати цієї роботи можуть стати важливим внеском у розвиток сучасного сільського господарства, забезпечуючи стабільність та стійкість виробництва в умовах змінюваного клімату.

Місце задачі, що розв'язується в роботі, у загальному комплексі задач автоматизації: автоматизація процесів керування мікрокліматом у тепличних господарствах розглядається як один з актуальних напрямів розвитку технологій точного землеробства. Задача, що розв'язується в межах даної роботи, займає важливе місце у загальному комплексі задач автоматизації аграрного сектору, зокрема в підсистемах локального контролю за умовами вирощування сільськогосподарських культур у закритому ґрунті. В умовах підвищених вимог до ресурсної ефективності, стабільності врожаю та якості продукції, впровадження автоматизованих систем такого типу дозволяє мінімізувати вплив людського чинника та підвищити точність регулювання критичних параметрів середовища.

У контексті загального підходу до побудови автоматизованих систем, задача керування мікрокліматом відноситься до класу задач контурного регулювання з елементами адаптивного та програмного управління. Інтеграція сенсорних технологій, виконавчих пристроїв та вбудованих мікроконтролерів у єдину систему забезпечує зворотний зв'язок, на основі якого здійснюється корекція режимів роботи відповідно до поточного стану середовища. Це дозволяє досягати динамічної стабільності системи та забезпечувати оптимальні умови для

фотосинтезу, росту та розвитку рослин незалежно від зовнішніх коливань температури чи вологості.

Таким чином, розробка автоматизованої системи керування мікрокліматом парника сприяє реалізації загальної концепції інтелектуального управління у сільському господарстві та підтримує перехід до кіберфізичних систем (Cyber-Physical Systems, CPS), що поєднують фізичні процеси з цифровими обчислювальними компонентами. Застосування таких рішень є одним з основних напрямів цифрової трансформації аграрного виробництва, забезпечуючи стабільне функціонування біотехнологічних об’єктів у режимі реального часу.

Область застосування розробки: розробка автоматизованої системи керування мікрокліматом має широку сферу практичного застосування, насамперед у сільському господарстві, зокрема в тепличному рослинництві, вертикальному фермерстві, агропромислових комплексах і біотехнологічних лабораторіях. Такі системи дозволяють оптимізувати умови вирощування рослин за допомогою точного регулювання температури, вологості, рівня освітлення та концентрації вуглекислого газу (CO_2), що безпосередньо впливає на врожайність, енергоефективність та стабільність виробничого процесу.

Кінцевими користувачами автоматизованої системи можуть бути як великі агрохолдинги, що спеціалізуються на масовому вирощуванні сільськогосподарської продукції в контрольованому середовищі, так і малі фермерські господарства, які прагнуть підвищити ефективність використання ресурсів і зменшити залежність від погодних умов. Також можливе використання таких систем у навчальних, дослідницьких і демонстраційних цілях у закладах аграрного профілю та наукових інститутах, які вивчають агротехнології.

Універсальність і модульність розробленої системи керування дозволяє адаптувати її до різних масштабів теплиць та вимог користувача. Завдяки використанню сучасних мікроконтролерів, сенсорних модулів і програмного забезпечення, така система може бути впроваджена з відносно невеликими

витратами, що робить її доступною для широкого кола споживачів у сфері агровиробництва.

Вимоги до програмно-технічного забезпечення, що розробляється в бакалаврській кваліфікаційній роботі, можна описати наступним чином:

- основне програмне середовище – Python, як мова високого рівня з широкою підтримкою бібліотек для роботи з мікроконтролерами та сенсорами;
- бібліотеки для візуалізації та обробки даних – Matplotlib для побудови графіків, Numpy для чисельного аналізу, Pandas для зберігання результатів спостережень;
- система логування та тестування – використовується модуль Unittest для перевірки функціональності компонентів та виявлення помилок у логіці керування.

Функції розробки:

- збирання даних від різних сенсорів, що вимірюють параметри мікроклімату (температура, вологість, CO_2, освітленість) в межах автоматизованої системи управління;
- обробка отриманих сенсорних даних відповідно до обраної логіки управління мікрокліматом, яка може включати використання статичної логіки, PID-регулювання або алгоритмів на базі машинного навчання;
- визначення та підтримка оптимальних значень параметрів мікроклімату шляхом включення або вимикання відповідних виконавчих пристроїв (вентиляторів, обігрівачів, освітлення);
- координація роботи всіх складових системи, щоб забезпечити ефективне управління та підтримку заданих параметрів мікроклімату в реальному часі;
- моделювання та прогнозування змін у параметрах мікроклімату для оцінки ефективності системи та коригування стратегії управління в залежності від вхідних даних;

- відображення результатів роботи системи для користувача через зручний інтерфейс, надання можливості для віддаленого моніторингу та налаштування параметрів мікроклімату.

Для виконання роботи необхідні наступні вхідні дані:

- дані про структуру та параметри роботи автоматизованої системи управління мікрокліматом;
- сенсорні дані про поточний стан параметрів мікроклімату (температура, вологість, освітленість, CO_2);
- налаштування та параметри для алгоритмів керування, включаючи правила для оптимізації мікроклімату в заданих умовах.

Перевіркою ефективності розробленої системи автоматизованого керування мікрокліматом буде її тестування на модельних даних із симуляцією різних сценаріїв зміни параметрів середовища. Аналіз реакції системи на ці зміни дозволить оцінити точність підтримання заданих умов, стабільність роботи алгоритму керування та загальну ефективність запропонованої архітектури в досягненні оптимального мікроклімату.

2 ДОСЛІДЖЕННЯ СИСТЕМ АВТОМАТИЗАЦІЇ МІКРОКЛІМАТУ ТА ПІДХОДІВ ДО АВТОМАТИЗОВАНОГО КЕРУВАННЯ

2.1 Дослідження основних параметрів мікроклімату

Мікроклімат – це сукупність параметрів фізичного середовища, що формуються у межах замкненого простору і впливають на стан рослин, людей або технічного обладнання. У системах автоматизованого керування мікрокліматом визначальними є такі параметри: температура повітря, відносна вологість, рівень освітленості та концентрація вуглекислого газу (CO_2). Контроль та підтримання цих величин у допустимих межах є критичним для забезпечення стабільних умов у теплицях, офісах, житлових приміщеннях, сховищах тощо.

Температура повітря є одним із базових показників, що характеризує тепловий стан середовища. У більшості випадків оптимальний температурний діапазон для комфортного перебування людини або росту рослин знаходиться в межах 20–26 °C. Надмірне підвищення температури може призвести до перегріву, прискореного випаровування вологи та зниження продуктивності фотосинтезу, тоді як надмірне зниження уповільнює фізіологічні процеси. Для її вимірювання зазвичай застосовують цифрові термометри або температурні датчики, наприклад, DHT22, DS18B20 тощо [56].

Відносна вологість повітря, що вимірюється у відсотках, визначає насиченість повітря водяною парою. Залежно від застосування, оптимальні значення можуть варіюватися: у тепличному господарстві це 60–80 %, тоді як у приміщеннях для людей – 40–60 %. Надлишкова вологість сприяє розвитку грибків і бактерій, а її нестача – висушуванню ґрунту та дихальних шляхів. Вимірювання здійснюється сенсорами типу DHT11, DHT22, BME280, що забезпечують одночасний моніторинг температури та вологості.

Рівень освітленості визначає кількість світлової енергії, що надходить до об’єкта. Вимірюється у люксах (lx) і має вирішальне значення для фотосинтетичної

активності в рослинництві, а також для забезпечення комфорту та зорової працездатності у приміщеннях. Наприклад, для теплиць потрібно щонайменше 10 000–15 000 lx, тоді як в офісах достатньо 300–500 lx. Освітленість вимірюється за допомогою фоточутливих датчиків, таких як BH1750 або TSL2561.

Концентрація вуглекислого газу (CO_2) є важливим показником як для житлових і робочих приміщень, так і для теплиць. Зростання рівня CO_2 понад допустимі межі (більше 1000 ppm) може погіршити самопочуття людей, спричинити втому або головний біль. У тепличних умовах, навпаки, штучне збагачення повітря CO_2 до 800–1200 ppm сприяє інтенсифікації фотосинтезу. Для вимірювання цього параметра використовують спеціалізовані сенсори, наприклад, MH-Z19 або SCD30.

Таким чином, аналіз та контроль зазначених параметрів мікроклімату є основою функціонування автоматизованих систем, що забезпечують стабільність середовища відповідно до заданих вимог. Їх інтеграція в систему керування дозволяє реалізувати ефективні стратегії регулювання на основі реального часу.

2.2 Аналіз існуючих систем автоматизації

У сучасних умовах автоматизація керування мікрокліматом у замкнених просторах реалізується за допомогою різноманітних апаратно-програмних рішень, зокрема на основі мікроконтролерів, відкритих платформ та хмарних сервісів. Тому необхідно послідовно проаналізувати актуальні підходи, що включають програмні реалізації на платформах Arduino та ESP32, використання платформ керування, таких як Node-RED і Home Assistant, а також підключення до хмарних інфраструктур, зокрема Blynk, Thingspeak і Firebase.

2.2.1 Огля програмних реалізацій на базі Arduino та ESP32

У контексті побудови автоматизованих систем моніторингу та керування мікрокліматом одним з ключових завдань є вибір апаратної платформи, яка б

забезпечувала стабільну роботу, достатній обсяг обчислювальних ресурсів, а також зручність у програмуванні та розширенні. Найбільш поширеними рішеннями для побудови подібних систем є мікроконтролери Arduino та ESP32. Їх популярність зумовлена широкою підтримкою серед розробників, великою кількістю доступних бібліотек, документації та активною спільнотою.

Arduino – це відкритоплатформенне апаратне середовище, орієнтоване на простоту розробки вбудованих систем. Основна частина реалізацій автоматизованого керування мікрокліматом з використанням Arduino базується на таких моделях як Arduino Uno, Arduino Nano та Arduino Mega. Ці пристрої дозволяють реалізовувати базові алгоритми зчитування показників температури, вологості, освітленості тощо [6].

Наприклад, найбільш часто використовуваними сенсорами у проектах на Arduino є:

- DHT11 або DHT22 для зчитування температури і вологості;
- BH1750 або LDR (фоторезистори) для вимірювання освітленості;
- MQ135 або аналогічні газові датчики для вимірювання концентрації CO_2 та інших газів.

Програмування Arduino здійснюється в середовищі Arduino IDE із застосуванням мови, що базується на спрощеному C++. У рамках розробки реалізується нескладна логіка типу «if-else» для контролю виконавчих пристроїв: наприклад, вмикання вентилятора або зволожувача при перевищенні порогових значень. Відомі приклади таких рішень охоплюють системи керування вентиляцією в теплицях, автоматизацію освітлення та регуляцію вологості в кімнатних умовах.

На рисунку 2.1 представлено вигляд робочого вікна середовища розробки Arduino IDE.

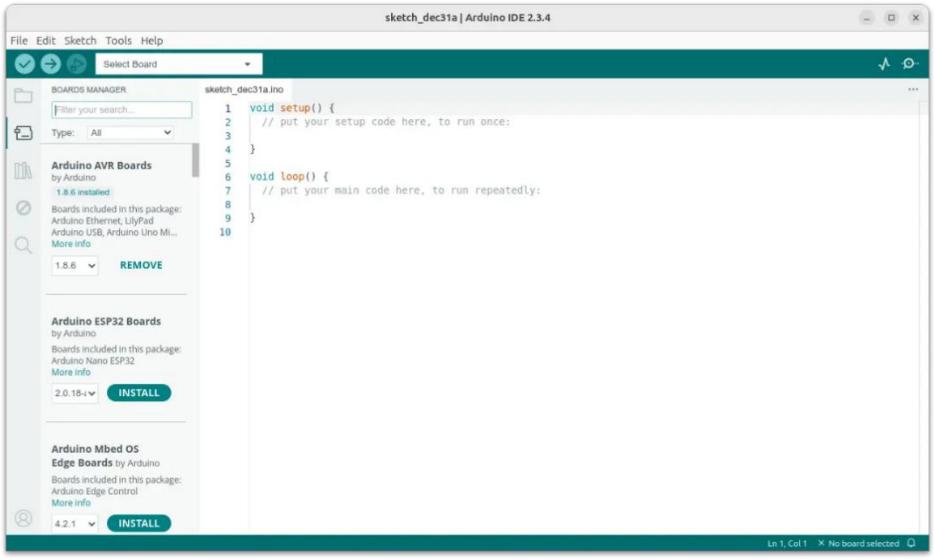


Рисунок 2.1 – Вигляд робочого вікна середовища розробки Arduino IDE

Перевагами Arduino є:

- простота апаратної та програмної реалізації;
- наявність широкого спектра готових бібліотек (наприклад, Adafruit Sensor, DHT, BH1750, LiquidCrystal для дисплеїв тощо);
- велика кількість навчальних матеріалів та відкритих проектів;
- низька вартість компонентів.

Серед недоліків можна відзначити обмеженість апаратних ресурсів. Наприклад, Arduino Uno має лише 2 КБ оперативної пам’яті (SRAM) та 16 МГц тактову частоту, що обмежує реалізацію складніших алгоритмів, включаючи адаптивне регулювання або машинне навчання. Крім того, Arduino не має вбудованих засобів для роботи з Wi-Fi чи Bluetooth, тож для підключення до хмарних сервісів необхідно використовувати додаткові модулі, наприклад, ESP8266.

ESP32 – це мікроконтролер від компанії Espressif, що забезпечує суттєво вищий рівень функціональності порівняно з класичними Arduino-платами. Він оснащений двоядерним процесором з тактовою частотою до 240 МГц, має до 520

КБ SRAM, вбудовані модулі Wi-Fi та Bluetooth, що дає змогу без додаткового обладнання реалізовувати інтеграцію з мобільними додатками або хмарними платформами [4,7].

ESP32 ідеально підходить для систем з багатьма сенсорами або необхідністю передавати дані через мережу. З технічної точки зору ESP32 має більшу кількість аналогових входів, ніж Arduino Uno, що забезпечує підключення більшої кількості сенсорів без використання мультиплексорів. Крім того, ESP32 підтримує протоколи SPI, I²C, UART, PWM та інші, що полегшує інтеграцію з різноманітними сенсорами та периферією.

Розробка на ESP32 може здійснюватися в середовищі Arduino IDE, PlatformIO або ESP-IDF. Це дозволяє розробникам використовувати знайомі інструменти з розширеною функціональністю. Наприклад, при створенні системи керування мікрокліматом у теплиці можна реалізувати не тільки класичну логіку «if-else», а й адаптивні методи регулювання, логування даних у реальному часі, підключення до Google Firebase або MQTT-брокерів для віддаленого моніторингу.

На рисунку 2.2 представлено вигляд робочого вікна середовища розробки ESP-IDF.

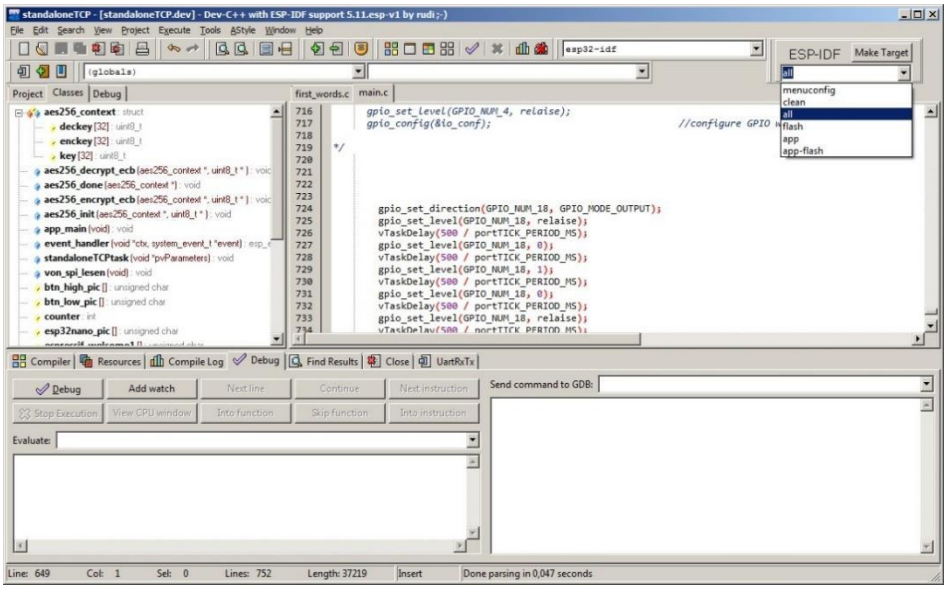


Рисунок 2.2 – Вигляд робочого вікна середовища ESP-IDF

До переваг ESP32 відносяться:

- висока обчислювальна потужність;
- вбудований Wi-Fi і Bluetooth;
- більший обсяг пам’яті та більша кількість портів;
- можливість використання FreeRTOS для реалізації багатозадачності;
- підтримка OTA (Over-the-Air) оновлень програмного забезпечення.

Недоліками ESP32 є більша складність у розробці, особливо для новачків, які раніше працювали лише з Arduino. Крім того, при роботі з мережею спостерігається підвищене енергоспоживання, що може бути критичним для автономних систем на живленні від батареї.

Порівняння Arduino та ESP32

З точки зору використання у проектах для моніторингу та регулювання мікроклімату, Arduino доцільно застосовувати у випадках, коли:

- необхідна мінімальна логіка керування;
- проект має обмежений бюджет;
- не передбачено з’єднання з мережею або хмарними платформами;
- реалізується навчальний або експериментальний прототип.

Натомість ESP32 є оптимальним вибором у разі, коли:

- система має передавати дані до віддаленого сервера або отримувати команди через Інтернет;
- потрібно обробляти багато сенсорних даних паралельно;
- реалізуються складні логічні структури або адаптивне регулювання;
- передбачено збереження даних у реальному часі, візуалізація або машинне навчання.

Типові сценарії використання

У типовій реалізації на Arduino мікроконтролер опитує сенсори (наприклад, температуру й вологість з DHT22), приймає рішення за фіксованими порогами й управляє виконавчими пристроями – зволожувачем, вентилятором або нагрівачем. Дані можуть виводитися на LCD-екран або серійну консоль. Усі дії відбуваються в циклі loop(), а логіка ґрунтується на простих умовних конструкціях [6].

У проектах на ESP32, крім зазначених функцій, часто реалізується передача даних через Wi-Fi у хмарні сервіси (наприклад, Firebase, Blynk або ThingSpeak), зберігання даних на SD-карті або у flash-пам’яті, взаємодія з мобільним додатком, інтерфейсом у веббраузері, а також розгорнута система логування, що допомагає проводити подальший аналіз і налагодження.

Таким чином, обидві платформи мають своє обґрунтування використання в залежності від цілей розробки. Якщо проект орієнтований на побудову автономної, мережево інтегрованої та адаптивної системи моніторингу мікроклімату, доцільно обрати ESP32 як базову платформу. Arduino, натомість, лишається ефективним рішенням для локальних задач з простою логікою керування.

2.2.2 Огляд платформ керування: Node-RED, Home Assistant

Для реалізації більш високорівневої логіки в системах автоматизації мікроклімату, зокрема при інтеграції з IoT, застосовуються спеціалізовані програмні платформи. Найбільш поширеними серед них є Node-RED та Home Assistant. Ці платформи забезпечують не лише централізоване управління різнорідними пристроями, але й можливість побудови адаптивної логіки на основі сценаріїв, правил та взаємозв’язків між подіями.

Node-RED – це середовище візуального програмування, розроблене IBM, яке базується на JavaScript і призначене для швидкої розробки інтеграційних сценаріїв у сфері IoT [21]. Його ключовою особливістю є блоковий підхід до програмування:

користувач створює «потоки» (flows), які складаються з вузлів (nodes), що виконують певні дії – зчитування даних, обробку, передавання або візуалізацію.

На рисунку 2.3 представлено вигляд робочого вікна серевізуального програмування Node-RED.

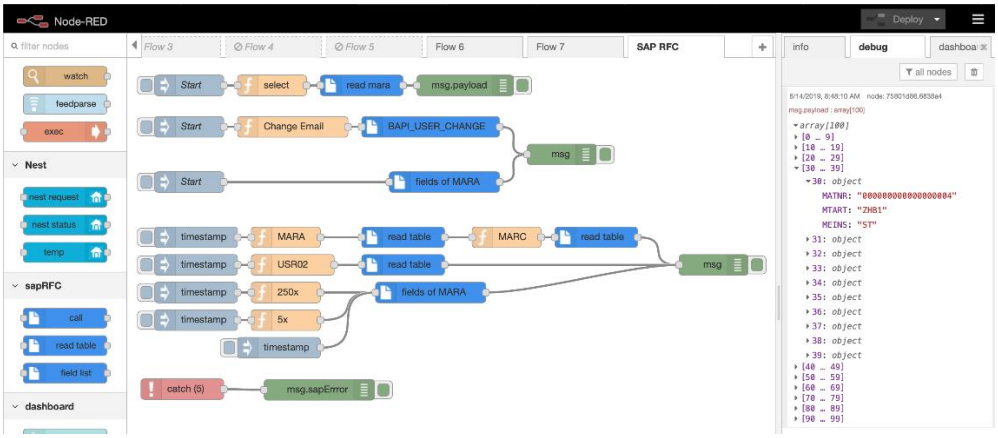


Рисунок 2.3 – Вигляд робочого вікна серевізуального програмування Node-RED

Node-RED дозволяє:

- інтегрувати дані з різних сенсорів і мікроконтролерів (ESP32, Arduino) за допомогою MQTT, HTTP або WebSocket;
- реалізувати обробку даних у реальному часі;
- створити адаптивні правила управління (наприклад, реакція на перевищення концентрації CO_2 або вологість нижче порогу);
- зберігати історію показників у базах даних (InfluxDB, SQLite, MongoDB тощо);
- створювати вебінтерфейси для моніторингу і взаємодії з користувачем.

Наприклад, у системі моніторингу мікроклімату Node-RED може отримувати дані від ESP32 через MQTT-брокер (наприклад, Mosquitto), проводити логічну обробку (наприклад, усереднення, порівняння з порогами, активація сповіщень), а також надсилати команди назад на мікроконтролер (вмикання вентилятора або

освітлення). Крім того, можлива реалізація push-сповіщень або email-повідомлень при виявленні критичних значень.

Переваги Node-RED:

- відкрите програмне забезпечення, доступне для встановлення на Raspberry Pi, ПК або в хмарі;
- проста інтеграція з MQTT, HTTP API, базами даних та хмарними платформами;
- можливість створення складної логіки без необхідності глибокого знання мов програмування;
- гнучкість у розширенні завдяки бібліотеці вузлів (Node-RED Flow Library).

Обмеження:

- потреба в постійному працюючому сервері (локально або в хмарі);
- потребує налаштування системи зберігання даних і брокера MQTT;
- складність відлагодження візуальних потоків у великих проектах.

Home Assistant – це ще одна потужна платформа з відкритим вихідним кодом, орієнтована на автоматизацію побутових процесів і підтримку великої кількості пристроїв. Вона написана на Python і здатна взаємодіяти з понад 1000 пристроями й сервісами, включно з ESPHome, MQTT, Zigbee, Z-Wave, Google Assistant, Amazon Alexa, Philips Hue тощо [20].

Home Assistant надає зручний вебінтерфейс для керування, моніторингу й створення автоматизацій на основі YAML-конфігурацій або графічного редактора.

На рисунку 2.4 представлено вигляд налаштованого меню керування приміщенням в середовищі Home Assistant .

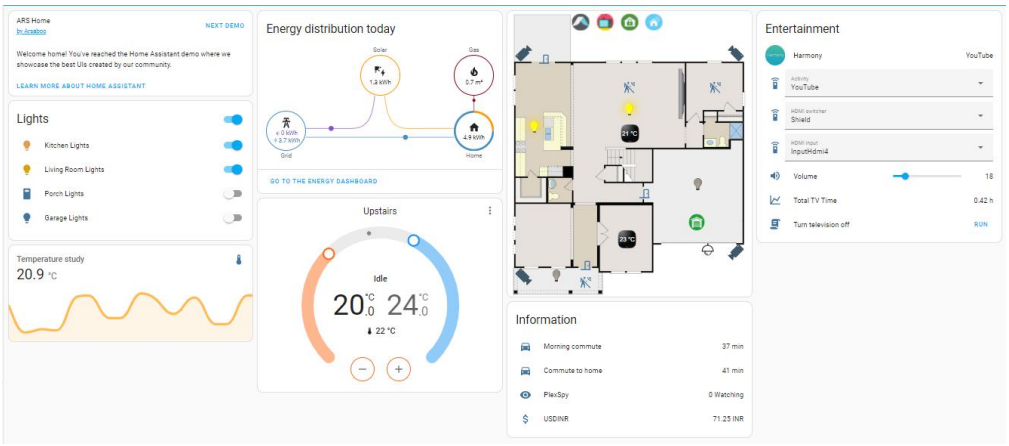


Рисунок 2.4 – Вигляд налаштованого меню керування приміщенням в середовищі Home Assistant

У контексті мікрокліматичних систем платформа дозволяє:

– здійснювати моніторинг параметрів температури, вологості, освітленості й концентрації CO_2 в реальному часі;
– будувати адаптивні сценарії взаємодії (наприклад, нічне зниження освітлення, включення провітрювання при зростанні CO_2);
– зберігати історію даних і відображати її у вигляді графіків;
– керувати пристроями через мобільний додаток або голосових асистентів;
– здійснювати автоматичне оновлення та контроль стану всіх вузлів.

Особливістю Home Assistant є підтримка інтеграцій з ESP32 через ESPHome – окремий фреймворк для прошивання ESP-пристроїв, який дозволяє конфігурувати пристрої без написання коду, лише описуючи їх поведінку в YAML-файлах. Це значно полегшує розгортання системи автоматизації навіть користувачам без глибоких технічних знань.

Переваги Home Assistant:

– готові інтерфейси для взаємодії з користувачем;
– підтримка величезної кількості протоколів і пристроїв;
– потужні інструменти автоматизації та журналювання;
– активна спільнота і часті оновлення.

Недоліки:

- вища складність початкового налаштування у порівнянні з Node-RED;
- потреба в сервері (наприклад, Raspberry Pi або NAS);
- можливість конфліктів між інтеграціями при великій кількості підключених пристроїв.

Порівняльна характеристика Node-RED та Home Assistant представлена в таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика платформ керування Node-RED та Home Assistant

Характеристика	Node-RED	Home Assistant
Архітектура	Потоково-орієнтована (flows)	Подієво-орієнтована (events)
Основна мова	JavaScript (Node.js)	Python (YAML-конфігурації)
Простота налаштування	Середня	Висока складність початково
Гнучкість логіки	Висока	Висока
Візуалізація	Через додаткові вузли (Dashboard)	Вбудована панель моніторингу
Підтримка ESP32	Через MQTT або HTTP	Через ESPHome
Мобільний додаток	Необов’язковий	Вбудований
Потреби в ресурсах	Низькі	Середні

Обидві платформи можуть бути використані паралельно. Зокрема, Node-RED може виступати як обробник даних і логічний маршрутизатор, а Home Assistant – як інтерфейс для взаємодії з користувачем і централізоване сховище інформації.

Таким чином, вибір платформи керування має ґрунтуватися на цільовому сценарії використання, ресурсах системи та навичках розробника. Для швидкого прототипування і простих сценаріїв доцільно використовувати Node-RED. Якщо

система повинна охоплювати більшу кількість пристроїв, мати розширені засоби візуалізації та мобільний доступ, кращим вибором буде Home Assistant.

2.2.3 Огляд хмарних сервісів: Blynk, ThingSpeak, Firebase

У системах автоматизованого керування мікрокліматом часто виникає потреба у віддаленому моніторингу та взаємодії з користувачем у режимі реального часу. Саме тому хмарні сервіси відіграють ключову роль у зберіганні, обробці та візуалізації даних, а також у забезпеченні зворотного зв'язку між пристроями і кінцевим користувачем. Blynk, ThingSpeak і Firebase - це три найбільш поширені платформи, які часто застосовуються в проектах на базі Arduino та ESP32.

Blynk – це IoT-платформа, орієнтована на розробників мікроконтролерних систем. Вона надає прості засоби для створення мобільних додатків, що дозволяють керувати пристроями, моніторити показники сенсорів та реалізовувати сценарії автоматизації без необхідності створення власного сервера або мобільного застосунку [14].

Ключові компоненти:

- Blynk App – мобільний додаток для Android та iOS, у якому користувач створює інтерфейс взаємодії (кнопки, графіки, індикатори);
- Blynk Cloud – хмарний сервер, що обробляє команди між додатком і мікроконтролером;
- Blynk Library – бібліотека для Arduino, ESP32, ESP8266 та інших пристроїв, що забезпечує обмін даними з хмарою.

На рисунку 2.5 представлено вигляд налаштованого меню керування пристроєм підігріву води в середовищі Blynk.

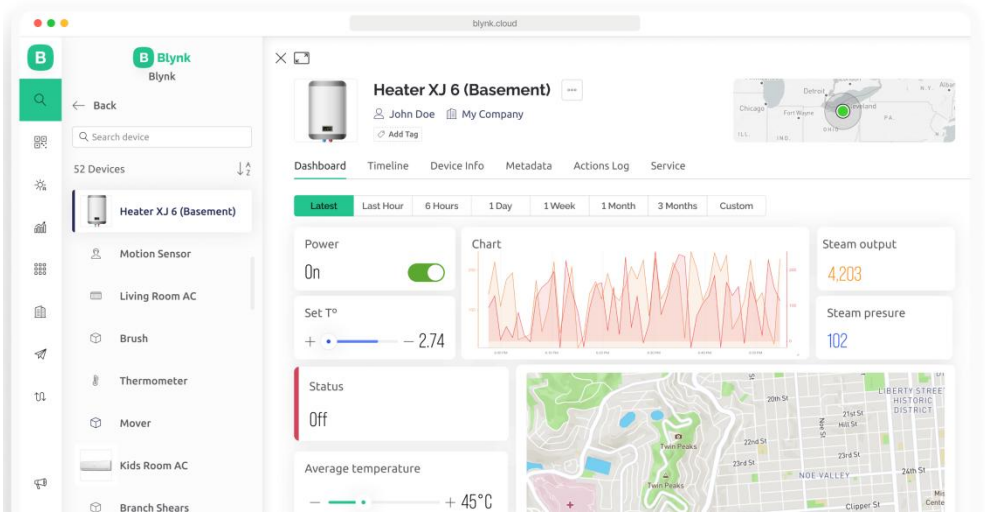


Рисунок 2.5 – Вигляд налаштованого меню керування пристроєм підігріву води в середовищі Blynk

Blynk дозволяє:

- надсилати дані з ESP32 (наприклад, температуру, вологість) у додаток у реальному часі;
- отримувати команди з мобільного пристрою (наприклад, увімкнення вентиляції);
- створювати просту логіку за допомогою Eventor або JavaScript/Webhooks у Blynk Console;
- зберігати дані для подальшого перегляду в режимі графіків (історія).

Серед переваг виділяють простоту інтеграції з ESP32/Arduino, готовий інтерфейс для користувача без необхідності веброзробки, мінімальні вимоги до конфігурації, а також підтримку push-сповіщень.

В той час недоліками вважають функціональні обмеження у безплатній версії (кількість віджетів, історія даних, об’єм зберігання), а також залежність від Blynk Cloud (у безплатній версії – неможливість розгортання локального сервера).

Blynk ідеально підходить для прототипування, освітніх цілей або створення MVP (minimum viable product) для систем моніторингу мікроклімату в умовному тепличному господарстві.

ThingSpeak – це хмарна IoT-платформа від MathWorks (розробників MATLAB), орієнтована на збір, аналіз і візуалізацію даних із сенсорів. Вона широко використовується в академічному середовищі та аматорських проектах завдяки інтеграції з MATLAB для аналітики [15].

На рисунку 2.6 представлено візуалізацію даних в платформі ThingSpeak.

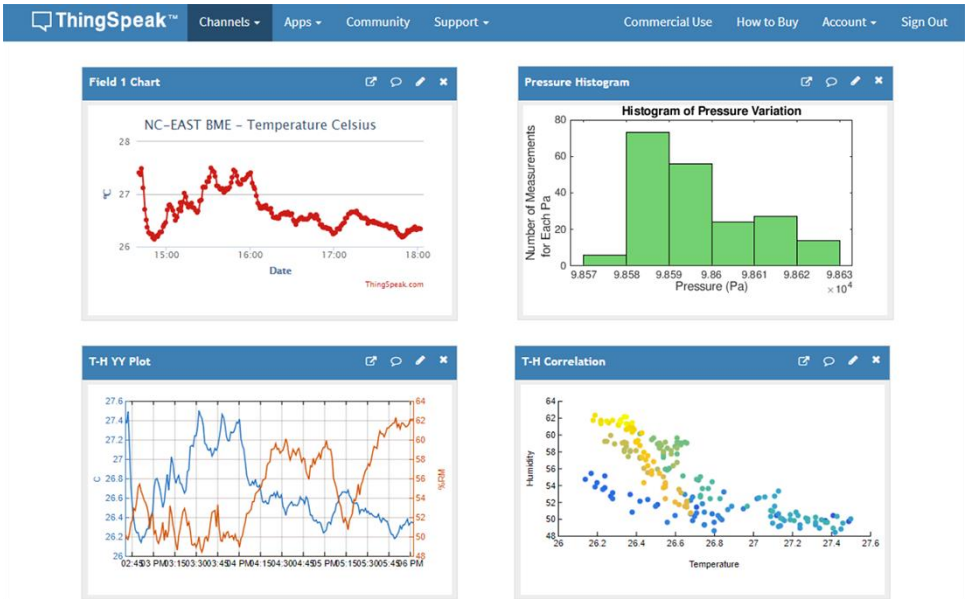


Рисунок 2.6 – Візуалізація даних в платформі ThingSpeak

Основні функції:

- прийом даних з пристроїв через HTTP або MQTT;
- зберігання даних у каналах (up to 8 полів для кожного каналу);
- вбудовані засоби побудови графіків та обробки даних (moving average, derivative, triggers);
- підтримка MATLAB-коду для складної обробки та прогнозування;

– налаштування подій: email або HTTP-запит при досягненні порогових значень.

ThingSpeak підходить для довготривалого зберігання параметрів мікроклімату, наприклад:

– вимірювання температури, вологості та CO_2 кожні 5 хвилин;
– побудова графіків зміни протягом дня, тижня, місяця;
– активація сценаріїв (наприклад, попередження про перегрівання).

Серед переваг: стабільність і масштабованість сервісу, можливість аналітики на базі MATLAB, публічний або приватний доступ до каналів та API-доступ до даних для сторонніх застосунків.

До недоліків відносять: менш гнучкий інтерфейс у порівнянні з Blynk, що не орієнтований на кінцевого користувача, а також обмеження у безкоштовному тарифі (кількість запитів, інтервал між запитами – 15 секунд).

ThingSpeak більше орієнтований на наукові дослідження, тестування гіпотез та створення систем пасивного моніторингу.

Firebase – це багатофункціональна платформа від Google для створення масштабованих веб- і мобільних застосунків. У контексті IoT Firebase використовується як бекенд для зберігання даних, синхронізації станів і реалізації реального часу [13].

Firebase надає:

– Realtime Database або Cloud Firestore – бази даних, які дозволяють зберігати структури даних та отримувати оновлення в режимі реального часу;
– Authentication – засоби ідентифікації користувачів;
– Cloud Functions – виконання обчислень на хмарному сервері за подією;
– Hosting – розміщення фронтенду або візуалізації.

Архітектуру обміну даними між мікроконтролером та користувачем з використанням Firebase компонентівзображено на рисунку 2.7.

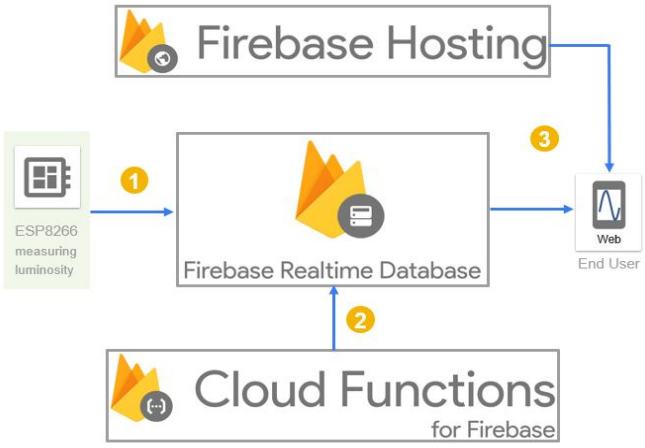


Рисунок 2.7 – Архітектура обміну даними між мікроконтролером та користувачем з використанням Firebase компонентів

У разі використання з ESP32:

- мікроконтролер надсилає дані в базу (температура, вологість тощо);
- клієнт (наприклад, вебсторінка) отримує зміни даних і відображає їх у реальному часі;
- користувач може надсилати команди через вебінтерфейс, що записуються в базу і зчитуються ESP32.

Firebase дозволяє будувати повноцінні масштабовані IoT-системи з розмежуванням прав доступу, історією змін, push-повідомленнями та автоматичними сценаріями.

Серед переваги: масштабованість і висока надійність, синхронізація в реальному часі, інтеграція з мобільними додатками (Android, iOS) та розширена безпека (на основі правил доступу).

Серед недоліків: складніший початковий поріг входу, потреба у налаштуванні системи аутентифікації, орієнтованість на розробників з досвідом у веб- або мобільній розробці.

Firebase найкраще підходить для розробників, які прагнуть створити масштабовану систему з можливістю інтеграції багатьох користувачів, розгалуженої логіки та динамічної взаємодії з пристроями.

В таблиці 2.2 представлено порівняльну характеристику досліджених хмарних сервісів.

Таблиця 2.2 – Порівняльна характеристика хмарних сервісів Blynk, ThingSpeak та Firebase

Критерій	Blynk	ThingSpeak	Firebase
Основне призначення	Мобільний інтерфейс IoT	Наукові дослідження IoT	Повноцінний бекенд для IoT
Підтримка реального часу	Так	Обмежена	Так
Візуалізація	Графіки, індикатори	Графіки	Кастомна
Рівень складності	Низький	Середній	Високий
Потреби в програмуванні	Мінімальні	Мінімальні	Високі
Зберігання історії	Обмежена (безплатно)	До 3 млн точок	Без обмежень (платно)
Push-сповіщення	Так	Через сторонні сервіси	Так

Таким чином, вибір хмарного сервісу визначається як технічними можливостями розробника, так і вимогами до системи: частота оновлень, необхідність мобільного інтерфейсу, обсяг даних, рівень захисту та потреба в розширенні. Для простих застосунків і швидкого старту підійде Blynk. Для аналітики – ThingSpeak. Для побудови комерційних систем із багатьма користувачами – Firebase.

2.3 Дослідження та порівняння підходів до керування мікрокліматом

Автоматизоване керування мікрокліматом у замкнених просторах, зокрема в теплицях, має на меті підтримання оптимальних умов для росту рослин шляхом регулювання температури, вологості повітря, концентрації вуглекислого газу (CO_2) та рівня освітленості. Для досягнення цієї мети застосовуються різні алгоритмічні підходи до керування. Найбільш поширеними серед них є: статична логіка на основі умовних операторів (if-else), регулятори на основі пропорційно-інтегрально-диференціального (PID) підходу, а також методи на основі машинного навчання. Кожен із цих підходів має власні переваги, обмеження та області доцільного застосування в контексті мікропроцесорних систем, таких як Arduino або ESP32.

2.3.1 Статична логіка на основі умовних операторів

Під статичною логікою слід розуміти прості алгоритми керування, які реалізуються за допомогою умовних конструкцій, таких як if, else if та else. Ці конструкції аналізують вхідні дані, отримані з датчиків, і активують відповідні виконавчі пристрої, такі як вентилятори, обігрівачі, помпи, клапани тощо. Основна суть цього підходу полягає в явному визначенні граничних умов або порогових значень, які запускають певні дії. Наприклад, простий алгоритм може виглядати так:

```
if (temperature > 30) {
  fanOn();
} else if (temperature < 25) {
  fanOff();
}
```

Цей приклад ілюструє, як статична логіка може бути використана для регулювання роботи вентилятора в залежності від температури повітря в парнику.

Серед переваг статичної логіки можна виділити кілька ключових аспектів. По-перше, простота реалізації: алгоритм легко реалізується в середовищі Arduino IDE або на основі Arduino-compatible прошивок для ESP32, що робить його доступним навіть для початківців. По-друге, висока прозорість: логіка керування легко читається та змінюється, що дозволяє швидко вносити корективи у разі необхідності. Крім того, статична логіка має мінімальні апаратні вимоги, не потребуючи додаткової пам’яті, обчислювальних ресурсів або складних бібліотек. Це робить її відповідною для низькоспоживаних вбудованих систем, що є важливим фактором у тепличному господарстві, де ресурси можуть бути обмежені [42].

Проте, незважаючи на свої переваги, статична логіка має й певні недоліки. По-перше, обмежена гнучкість: навіть незначні зміни в умовах середовища потребують перепрошивки або переналаштування граничних значень. Це може стати проблемою в умовах, де умови швидко змінюються. По-друге, відсутність адаптивності: алгоритм не «вчиться» на основі попередніх даних і не враховує динаміку змін, що обмежує його ефективність у складніших сценаріях. Ще одним недоліком є проблеми з осциляціями: вмикання та вимикання пристроїв поблизу порогу часто призводить до нестабільної роботи, що відоме як ефект «хитання» або часте перемикання. Нарешті, складність масштабування: у багатофакторних системах кількість умов може стрімко зростати, що призводить до складного, фрагментованого коду, який важко підтримувати.

У контексті тепличного господарства статична логіка може бути застосована для простих однофакторних рішень – наприклад, для активації вентиляції при перевищенні температури. Однак у випадках, де необхідна точна підтримка умов і взаємозалежне регулювання, таких як вплив вологості на температуру, ефективність такого підходу суттєво знижується. Це підкреслює важливість розвитку більш складних алгоритмів, які можуть адаптуватися до змінних умов, забезпечуючи оптимальні умови для росту рослин у теплицях.

2.3.2 Регулятори на основі пропорційно-інтегрально-диференціального (PID) підходу

Пропорційно-інтегрально-диференціальне регулювання, або PID-регулювання, є класичним і надзвичайно популярним підходом до автоматичного керування, який набув широкого застосування в промислових системах. Цей метод регулювання базується на постійному вимірюванні похибки, що визначається як різниця між бажаним значенням параметра і його поточним значенням. Завдяки трьом складовим – пропорційній (P), інтегральній (I) та диференціальній (D) – система здатна ефективно реагувати на зміни в середовищі та підтримувати необхідні умови [7,8,41].

Пропорційна складова (P) відповідає за реагування на поточну похибку. Це означає, що чим більша похибка, тим сильніший буде керуючий вплив. Пропорційний регулятор забезпечує швидку реакцію на зміни, але сам по собі може призвести до залишкової похибки, оскільки система може не досягти точно бажаного значення.

Інтегральна складова (I) має на меті врахування накопиченої похибки з часом. Вона сумує всі попередні похибки, що дозволяє усунути залишкову похибку, яка може виникнути внаслідок пропорційного регулювання. Це забезпечує більш точне досягнення бажаного значення, проте може призвести до коливань, якщо не буде правильно налаштована.

Диференціальна складова (D) прогнозує зміну похибки, враховуючи її швидкість зміни. Це дозволяє системі передбачити, як швидко змінюється похибка, і відповідно коригувати керуючий вплив, що може зменшити коливання та забезпечити стабільність системи.

Формула PID-регулятора виглядає наступним чином:

$$u(t) = Kp * e(t) + Ki * \int e(t)dt + Kd * de(t)/dt,$$

де $u(t)$ – керуючий вплив,

$e(t)$ – похибка,

Kp, Ki, Kd – коефіцієнти налаштування PID.

Серед численних переваг, які пропонує PID-регулювання, можна виділити гнучкість у регулюванні. Це означає, що система може бути налаштована для конкретного об’єкта керування, що робить її універсальною для різних застосувань. Наприклад, в тепличному господарстві PID-регулятори можуть бути використані для підтримки стабільної температури та вологості, що є критично важливим для оптимального росту рослин.

Іншою значною перевагою є висока стабільність у випадках з інерційними об’єктами, такими як теплоємні матеріали або системи з затримками в реакції. PID-регулятори здатні ефективно управляти такими системами, оскільки вони можуть компенсувати затримки в реакції та підтримувати стабільність.

Зменшення коливань є ще однією важливою перевагою PID-регуляторів. Завдяки своїй структурі, ці регулятори здатні згладжувати переходи між станами без ефекту надлишкового регулювання, що може бути особливо корисним у системах, де важливо уникати різких змін.

Проте, незважаючи на численні переваги, PID-регулювання має й певні недоліки. Одним із основних недоліків є потреба в точному налаштуванні параметрів Kp, Ki, Kd. Неправильне налаштування цих параметрів може призвести до нестабільності системи, що, в свою чергу, може викликати проблеми в управлінні.

Іншим недоліком є підвищені вимоги до обчислювальних ресурсів у порівнянні з простими логічними конструкціями, такими як if-else. Це особливо актуально при реалізації інтегрувальної та диференційної складових, які

потребують значних обчислювальних потужностей для обробки даних у реальному часі [7,8].

Ще однією важливою проблемою є відсутність адаптації до змін середовища або структури об’єкта керування. PID-регулятори не можуть самостійно «вчитися» або адаптуватися до нових умов, що може обмежити їх ефективність у середовищах з частими змінами.

На мікроконтролерах, таких як ESP32 або Arduino, реалізація PID-регулювання можлива завдяки використанню відповідних бібліотек, наприклад, PID_v1.h. Це дозволяє розробникам легко інтегрувати PID-регулятори у свої проекти, спрощуючи процес налаштування та управління [7].

Типовим прикладом застосування PID-регулятора може бути підтримання температури на рівні 25 °C шляхом регулювання потужності нагрівача з використанням широтно-імпульсної модуляції (ШІМ). Завдяки PID-регулятору, система може точно контролювати температуру, зменшуючи енергоспоживання та забезпечуючи оптимальні умови для росту рослин.

У тепличному середовищі PID-регулятори є оптимальним рішенням для керування температурою та вологістю, особливо в умовах інерційної поведінки системи [7,18]. Наприклад, після вимкнення обігріву температура в теплиці може повільно знижуватися, і PID-регулятор здатний забезпечити плавність реакції на ці зміни, що дозволяє уникнути різких коливань температури.

Застосування PID-регуляторів у тепличному господарстві не лише забезпечує плавність реакції, але й сприяє зниженню енергоспоживання. Завдяки точному регулюванню температури та вологості, фермери можуть зменшити витрати на енергію, що є важливим аспектом у сучасному агровиробництві.

2.3.3 Методи на основі машинного навчання

Сучасні тенденції в автоматизації сільського господарства, зокрема в управлінні мікрокліматом парників, дедалі більше включають використання

методів машинного навчання. Ці методи дозволяють адаптивно керувати складними процесами, враховуючи численні змінні, взаємозв'язки між параметрами та сезонні тренди. Завдяки цьому, агрономи та фермери отримують можливість оптимізувати умови для росту рослин, що, в свою чергу, може призвести до збільшення врожайності та покращення якості продукції [18].

Основні підходи в машинному навчанні, які можуть бути застосовані для управління мікрокліматом, включають регресійні моделі, нейронні мережі, підкріплювальне навчання та класифікацію станів. Регресійні моделі, такі як лінійна регресія або регресійні дерева, використовуються для прогнозування температури або вологості в майбутньому. Це дозволяє здійснювати попереджувальне керування, наприклад, передбачити, коли потрібно увімкнути вентилятори або обігрівачі, щоб підтримувати оптимальні умови для рослин.

Нейронні мережі, в свою чергу, забезпечують багаторівневе представлення вхідних параметрів, що дозволяє приймати складні рішення на основі комплексного аналізу даних. Наприклад, нейронна мережа може враховувати не лише температуру та вологість, а й інші фактори, такі як рівень освітленості, тип рослин і навіть прогноз погоди, для оптимізації роботи декількох пристроїв одночасно.

Підкріплювальне навчання є ще одним потужним методом, який дозволяє агенту навчатися на основі винагород за дії в середовищі. Наприклад, система може отримувати винагороду за зменшення споживання електроенергії при збереженні оптимального клімату в парнику. Це дозволяє не лише підтримувати необхідні умови, а й оптимізувати витрати на енергію, що є важливим аспектом для агровиробництв.

Класифікація станів дозволяє виявляти типові ситуації, такі як «перегрів» або «висока вологість», і вибирати відповідні дії за шаблоном. Це може бути особливо корисним у ситуаціях, коли потрібно швидко реагувати на зміни в середовищі, не витрачаючи час на складні обчислення [19-21].

Застосування машинного навчання в управлінні мікрокліматом парників має безліч переваг. По-перше, це дозволяє динамічно адаптувати алгоритми до змін у зовнішньому середовищі, таких як погодні умови чи зміна типу рослин. Наприклад, якщо в парнику вирощуються рослини, які потребують більше вологи в певні періоди року, система може автоматично адаптуватися до цих змін, забезпечуючи оптимальний рівень вологості.

По-друге, машинне навчання здатне прогнозувати ефекти керувальних дій. Це означає, що система може передбачити, скільки часу потрібно для зниження температури на 2 °C після увімкнення вентилятора. Такі прогнози дозволяють агрономам планувати свої дії більш ефективно, зменшуючи ризик перепадів температури або вологості.

Крім того, застосування машинного навчання може значно мінімізувати використання енергії. Завдяки оптимізації сценаріїв, система може знижувати енергоспоживання, одночасно підтримуючи необхідні умови для росту рослин. Це особливо важливо в умовах зростаючих витрат на енергію, що робить автоматизацію ще більш актуальною.

Проте реалізація рішень на основі машинного навчання в контексті мікроконтролерів, таких як Arduino або ESP32, має певні обмеження. По-перше, ці мікроконтролери мають обмежені ресурси пам’яті та обчислювальної потужності [23]. Типові мікроконтролери мають лише кілька десятків кілобайт оперативної пам’яті та низьку тактову частоту, що ускладнює запуск складних моделей машинного навчання.

По-друге, відсутність стандартних бібліотек та середовищ для тренування моделей є серйозною перешкодою. Навчання моделей зазвичай виконується поза межами мікроконтролера, наприклад, на комп’ютері. Для вбудованих систем потрібна оптимізована версія моделі або її спрощена реалізація, така як TinyML, яка розробляє легковагові моделі машинного навчання, оптимізовані для мікроконтролерів з обмеженими ресурсами.

Крім того, для адекватного навчання моделей необхідні значні обсяги якісних даних. Це може бути проблемою, оскільки історичні дані з датчиків можуть бути відсутні або неповні, що ускладнює навчання та тестування моделей.

Ще одним викликом є проблеми інтерпретованості та відлагодження алгоритмів машинного навчання. Алгоритми, особливо нейронні мережі, часто є «чорними ящиками», що ускладнює виявлення причин несправностей або неправильних рішень. В умовах агрономії важливо мати можливість зрозуміти, чому система прийняла те чи інше рішення, щоб мати можливість коригувати її роботу.

Серед переваг підходу машинного навчання можна виділити здатність адаптуватися до змін у роботі системи без ручного перепрограмування. Це дозволяє зекономити час і ресурси, оскільки агрономи можуть зосередитися на інших аспектах управління парником. Крім того, можливість роботи з багатовимірними, нелінійними залежностями робить ML-підходи особливо потужними для складних агрономічних систем.

Поліпшення енергозбереження через більш точне передбачення та оптимізацію дій також є важливим аспектом, оскільки це дозволяє знижувати витрати на експлуатацію. Підвищення якості підтримки мікроклімату за рахунок врахування комплексних факторів робить машинне навчання надзвичайно корисним інструментом для сучасного агровиробництва.

Приклади реалізацій машинного навчання в управлінні мікрокліматом парників включають використання TinyML, що дозволяє запускати легковагові моделі машинного навчання на мікроконтролерах з низькими ресурсами [9]. Наприклад, бібліотека TensorFlow Lite Micro дозволяє реалізувати прості нейронні мережі на ESP32, що відкриває нові можливості для автоматизації та оптимізації процесів [24, 25].

Реалізація класифікаторів для виявлення типових станів мікроклімату також є важливим напрямком. Це дозволяє активувати відповідні дії без необхідності

повного моделювання процесу, що може бути складним і ресурсоємним. Застосування простих алгоритмів прогнозування дозволяє заздалегідь коригувати параметри керування, наприклад, збільшувати подачу вологи за прогнозом підвищення температури, що забезпечує більш точне управління мікрокліматом.

Таким чином, машинне навчання відкриває нові горизонти для автоматизації управління мікрокліматом парників, проте реалізація таких рішень вимагає врахування обмежень, з якими стикаються мікроконтролери.

В таблиці 2.3 представлено порівняння розглянутих підходів до керування мікрокліматом.

Таблиця 2.3 – Порівняння підходів до керування мікрокліматом

Критерій	Статична логіка (if-else)	PID-регулювання	Машинне навчання (ML)
Простота реалізації	Дуже проста	Середня	Висока складність
Гнучкість	Низька (жорстко задані пороги)	Висока (регулювання параметрів)	Дуже висока (адаптація)
Обчислювальні ресурси	Мінімальні	Помірні	Високі
Здатність до адаптації	Відсутня	Обмежена (тільки налаштування)	Висока (навчання на даних)
Точність підтримки параметра	Низька	Висока	Потенційно найвища
Складність масштабування	Висока	Помірна	Висока
Необхідність підготовки даних	Відсутня	Відсутня	Висока (для навчання)
Застосування в парнику	Для простих задач	Для більш точного контролю	Для комплексного, адаптивного керування

Враховуючи специфіку тепличного середовища, динамічність і залежність процесів керування від багатьох факторів, таких як зовнішня погода, зміни стану рослин, а також інерційність обігріву та вентиляції, вибір методу керування є критично важливим.

Статична логіка (if-else) є простим і зрозумілим підходом, який підходить для початкових прототипів і базових систем автоматизації. Вона дозволяє швидко реалізувати основні функції автоматичного керування без необхідності впровадження складних алгоритмів. Однак, незважаючи на свою простоту, статична логіка має суттєві обмеження. Вона не враховує динаміку процесів, що може призвести до коливань параметрів, які важливі для стабільного функціонування теплиць [26, 27]. Наприклад, якщо температура повітря в парнику коливається, система може не встигати адекватно реагувати на зміни, що негативно вплине на умови росту рослин.

PID-регулятори, в свою чергу, є оптимальним рішенням для контролю таких критичних параметрів, як температура і вологість у парнику. Вони здатні згладжувати коливання і забезпечувати високу точність підтримки заданих значень. Цей метод регулювання підходить для мікроконтролерів середнього рівня, оскільки добре відпрацьований, має велику кількість готових бібліотек і документованих рішень. Однак налаштування PID-параметрів є критично важливою частиною реалізації системи, що потребує експериментального підходу. Неправильне налаштування може призвести до нестабільності системи, тому важливо проводити тестування та корекцію параметрів у реальних умовах.

Машинне навчання пропонує нові перспективи для створення інтелектуальних систем керування, здатних враховувати складні взаємозв’язки між різними параметрами та адаптуватись до змін у середовищі без ручного переналаштування. Це означає, що такі системи можуть самостійно навчатися на основі отриманих даних, що робить їх надзвичайно гнучкими. Проте в реальних умовах мікроконтролери, які зазвичай використовуються в агрономії, часто

потребують зовнішньої підтримки, такої як хмарні обчислення або потужніші контролери, для реалізації рішень на основі машинного навчання. Крім того, оптимізація моделей для роботи в умовах обмежених ресурсів (наприклад, TinyML) є складним завданням, яке вимагає додаткових зусиль.

Впровадження систем машинного навчання є складним і ресурсоємним процесом, але воно відкриває нові можливості для ефективної автоматизації управління мікрокліматом. Системи, що використовують машинне навчання, можуть не лише підтримувати оптимальні умови, а й прогнозувати зміни в середовищі, що дозволяє вжити заходів до їх настання [10]. Наприклад, система може передбачити підвищення температури внаслідок зміни погоди і заздалегідь активувати охолоджувальні пристрої [28].

Таким чином, для реалізації автоматизованої системи керування мікрокліматом парника на базі мікроконтролера оптимально починати з поєднання статичної логіки для простих аварійних реакцій та PID-регуляції для основного контролю параметрів. Це дозволить забезпечити базову стабільність і точність системи. У подальшому, з розвитком технологій та збільшенням досвіду, можна розглядати інтеграцію машинного навчання для адаптивного та прогнозного керування, що дозволить значно підвищити ефективність і продуктивність агрономічних процесів.

2.4 Уточнена постановка задачі

У контексті автоматизації системи керування мікрокліматом парника, важливо чітко визначити завдання, які необхідно вирішити для досягнення оптимальних умов для росту рослин. На основі проведеного аналізу існуючих підходів до керування, можна виділити кілька ключових аспектів, які мають бути враховані при проектуванні системи.

По-перше, необхідно обрати мікроконтролер, який забезпечить достатню обчислювальну потужність і ресурси для реалізації алгоритмів керування, та під який буде виконуватись симуляція вхідних даних. Як показав аналіз, платформи

Arduino та ESP32 мають свої переваги та недоліки, що вплине на вибір конкретної моделі. Важливо врахувати не лише технічні характеристики, а й можливості інтеграції з різними сенсорами та виконавчими пристроями.

По-друге, розробка загальної архітектури системи є критично важливою для її функціонування. Система повинна включати модулі для зчитування даних з сенсорів, обробки інформації в реальному часі та управління виконавчими пристроями. Архітектура має бути побудована таким чином, щоб забезпечити гнучкість та можливість подальшого розширення функціоналу.

По-третє, важливо чітко визначити логіку роботи системи. Це включає в себе алгоритми зчитування даних з сенсорів, їх обробку та прийняття рішень на основі отриманої інформації. Логіка повинна бути достатньо простою для реалізації, але водночас забезпечувати необхідну точність та адаптивність до змін у мікрокліматі.

Наступним етапом є визначення діапазонів допустимих значень для параметрів мікроклімату парника. Ці значення мають бути обґрунтовані на основі агрономічних досліджень, що дозволить забезпечити оптимальні умови для росту рослин. Наприклад, для температури, вологості та рівня освітленості потрібно встановити конкретні межі, в межах яких система буде підтримувати стабільність.

Останнім, але не менш важливим етапом, є визначення логіки алгоритму керування. Це включає в себе розробку алгоритмів, які будуть реагувати на зміни в параметрах мікроклімату, активуючи відповідні виконавчі пристрої. Алгоритми мають бути адаптивними, що дозволить системі швидко реагувати на зовнішні зміни, такі як зміна погодних умов або зміна стану рослин.

Таким чином, уточнена постановка задачі включає в себе вибір мікроконтролера, розробку загальної архітектури системи, опис логіки роботи, визначення допустимих значень для параметрів мікроклімату та логіки алгоритму керування. Ці аспекти стануть основою для подальшого проектування системи, що дозволить створити ефективну та адаптивну автоматизовану систему управління мікрокліматом парника.

3 ПРОЕКТУВАННЯ СИСТЕМИ КЕРУВАННЯ МІКРОКЛІМАТОМ ПАРНИКА НА БАЗІ МІКРОКОНТРОЛЕРА

3.1 Вибір мікроконтролера

На етапі проектування автоматизованої системи керування мікрокліматом парника важливим кроком є вибір відповідного мікроконтролера. Це рішення вплине на функціональність системи, її надійність, вартість реалізації та енергоефективність. Хоча в подальшій роботі буде використовуватися симуляція на Python, важливо розуміти, які можливості пропонують різні мікроконтролери, оскільки це допоможе створити більш точну модель системи [29-31].

Arduino Uno є одним із найбільш розповсюджених мікроконтролерів серед розробників початкового рівня. Його основними перевагами є простота використання, наявність великої кількості бібліотек, активна спільнота користувачів, а також низька вартість [3]. Технічні характеристики Arduino Uno включають мікроконтролер ATmega328P, частоту 16 МГц, 2 КБ оперативної пам’яті (SRAM), 32 КБ флеш-пам’яті та 14 цифрових входів/виходів (6 з яких підтримують PWM), а також 6 аналогових входів.

Arduino Uno є ідеальним вибором для простих проектів, де не потрібно обробляти великі обсяги даних або реалізовувати складні алгоритми. Він підходить для базових задач, таких як зчитування даних з датчиків температури та вологості, а також управління простими виконавчими пристроями, такими як реле або вентилятори.

Мікроконтролер Arduino Uno представлений на рисунку 3.1.

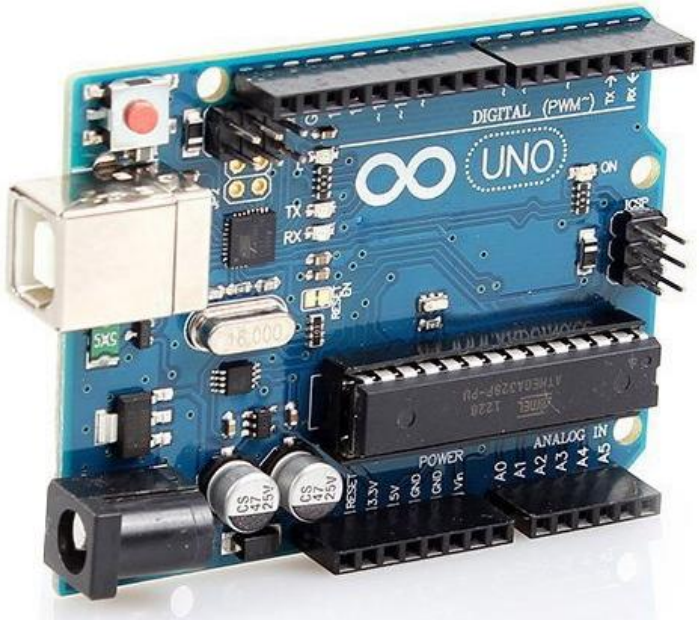


Рисунок 3.1 – Мікроконтролер Arduino Uno

Проте, незважаючи на свою доступність, Arduino Uno має суттєві обмеження з точки зору обчислювальної потужності, пам’яті та засобів бездротового зв’язку. Наприклад, обмежена оперативна пам’ять не дозволяє реалізувати складні алгоритми, такі як машинне навчання або адаптивне регулювання. Крім того, для підключення до Інтернету або хмарних сервісів необхідно використовувати додаткові модулі, що ускладнює проектування системи [32, 33].

На відміну від Arduino Uno, ESP32 є значно потужнішим і сучаснішим мікроконтролером. Він має двоядерний процесор Tensilica Xtensa LX6 з тактовою частотою до 240 МГц, до 520 КБ SRAM, вбудовані модулі Wi-Fi та Bluetooth, численні GPIO-входи, а також підтримку сенсорних інтерфейсів, SPI, I2C, UART та інших протоколів. Це робить його ідеальним вибором для побудови більш складних систем, які потребують мережевої взаємодії та обробки великої кількості даних.

Мікроконтролер ESP32 представлений на рисунку 3.2.

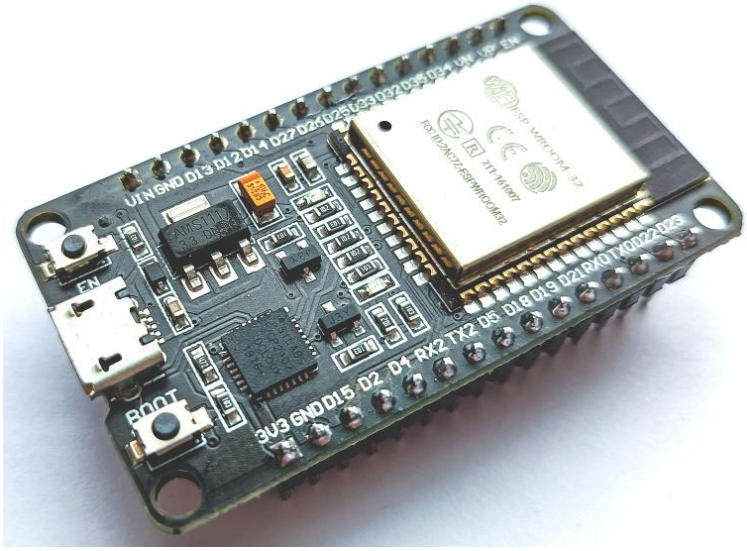


Рисунок 3.2 – Мікроконтролер ESP32

ESP32 підтримує як Arduino-орієнтоване програмування, так і використання інших середовищ, таких як PlatformIO або MicroPython. Це дозволяє розробникам обирати найбільш зручний для них інструмент, а також реалізовувати проекти з використанням різних мов програмування. Наприклад, можливість використання MicroPython дозволяє швидко прототипувати рішення, що є особливо корисним у контексті досліджень та навчання [30].

ESP32 є ідеальним вибором для систем з багатьма сенсорами або необхідністю передавати дані через мережу. Завдяки своїй високій обчислювальній потужності та вбудованим модулям бездротового зв’язку, він може легко інтегруватися з хмарними платформами для моніторингу та управління в режимі реального часу. Це відкриває нові можливості для створення адаптивних систем, які можуть реагувати на зміни в середовищі, а також здійснювати аналіз даних на основі отриманих показників.

Ще одним варіантом є Raspberry Pi Pico, заснований на мікроконтролері RP2040 [5, 36]. Цей мікроконтролер поєднує високу продуктивність з гнучкістю та доступною ціною. Raspberry Pi Pico має двоядерний процесор, що працює на частоті до 133 МГц, а також 264 КБ SRAM і 2 МБ флеш-пам’яті. Проте, на відміну

від ESP32, він не має вбудованих засобів бездротового зв’язку, що потребує додаткових модулів для реалізації функцій моніторингу в режимі реального часу або хмарної синхронізації.

Мікроконтролер Raspberry Pi Pico RP2040представлений на рисунку 3.3.

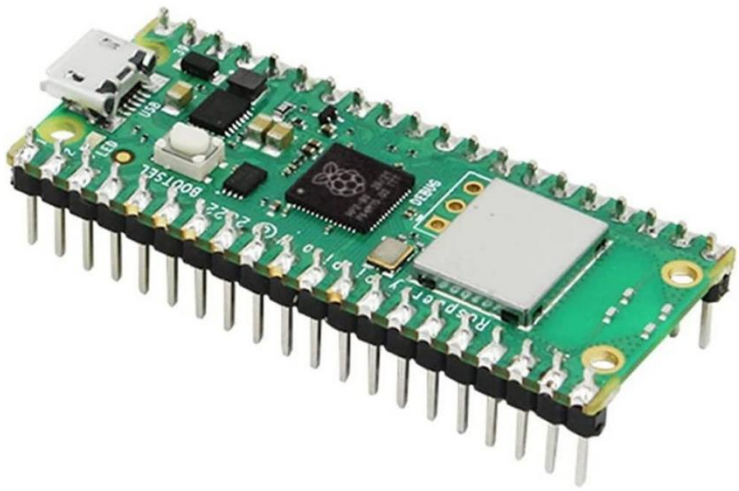


Рисунок 3.3 – Мікроконтролер Raspberry Pi Pico RP2040

Raspberry Pi Pico є хорошим вибором для проектів, де необхідна висока продуктивність обробки даних, але без потреби в бездротовому з’єднанні. Це може бути корисно в ситуаціях, коли система працює в локальній мережі або при використанні кабельного підключення до інших пристроїв.

При виборі мікроконтролера для автоматизованої системи керування мікрокліматом парника важливо врахувати специфічні потреби проекту. Наприклад, якщо проект передбачає просте управління мікрокліматом без складних алгоритмів, Arduino Uno може бути достатнім [37, 38, 19]. Однак, для більш складних систем, які потребують інтеграції з хмарними сервісами, обробки великої кількості даних і бездротового зв’язку, ESP32 є більш доцільним вибором.

ESP32 надає широкий спектр можливостей, які можуть бути реалізовані в рамках симуляції на Python. Хоча в подальшій роботі фізичний мікроконтролер не буде використовуватися, знання про його можливості та функціональність дозволяє

створити більш точну модель системи. Наприклад, можна моделювати роботу системи, яка реагує на зміну температури, вологості та інших параметрів, використовуючи дані, згенеровані в процесі симуляції.

Крім того, ESP32 підтримує програмування в режимі реального часу, що дозволяє створювати адаптивні алгоритми керування, які можуть динамічно реагувати на зміни в середовищі [39,36]. Це є особливо важливим для систем, які повинні підтримувати оптимальні умови для росту рослин, враховуючи різноманітні фактори, такі як зміна погоди, час доби та сезонні коливання.

Таким чином, вибір мікроконтролера є критично важливим етапом у проектуванні автоматизованої системи керування мікрокліматом парника. Обравши ESP32, розробники отримують потужний інструмент для реалізації складних алгоритмів і інтеграції з сучасними технологіями, що дозволяє створити ефективну та адаптивну систему для моніторингу та управління мікрокліматом.

3.2 Розробка загальної архітектури системи

Архітектура системи керування мікрокліматом парника є критично важливим етапом у розробці ефективної автоматизованої системи. Основною метою цієї архітектури є забезпечення надійного обміну даними між сенсорними пристроями, логікою прийняття рішень та виконавчими елементами [38]. Правильна організація цих компонентів дозволяє створити інтегровану систему, здатну адаптуватися до змін у середовищі та забезпечувати оптимальні умови для росту рослин.

Система складається з кількох основних підсистем, кожна з яких виконує свою специфічну функцію:

- Сенсорні вузли: Ці пристрої відповідають за вимірювання ключових параметрів мікроклімату, таких як температура, вологість, рівень освітленості та концентрація вуглекислого газу (CO_2). Використання різноманітних датчиків, таких як DHT22 для температури і вологості, BH1750 для

освітленості, а також MH-Z19 для CO_2, дозволяє отримувати точні дані про стан середовища в парнику [26].

- Мікроконтролер ESP32: Цей компонент виступає центральним обчислювальним модулем системи. ESP32 не лише обробляє дані, отримані від сенсорів, але й виконує алгоритми прийняття рішень на основі цих даних. Його вбудовані можливості бездротового зв'язку (Wi-Fi та Bluetooth) дозволяють системі легко інтегруватися з мобільними додатками або хмарними сервісами.
- Виконавчі пристрої: Це елементи, які реалізують фізичні дії для регулювання мікроклімату. До них відносяться вентилятори для покращення циркуляції повітря, обігрівачі для підтримки оптимальної температури, зрошувачі для контролю вологості та освітлення для забезпечення необхідного рівня світла для рослин. Активація або деактивація цих пристроїв здійснюється на основі рішень, прийнятих мікроконтролером.
- Програмний блок логіки управління: Цей блок реалізує алгоритм прийняття рішень, який ґрунтується на даних, отриманих від сенсорів. Логіка управління може включати прості правила (наприклад, увімкнення вентилятора при перевищенні температури) або більш складні алгоритми, такі як PID-регулювання або методи машинного навчання, які дозволяють адаптувати систему до змінних умов [7, 43].
- Блок комунікації з користувачем: Цей компонент забезпечує взаємодію між системою та користувачем. Інтерфейс може бути реалізований через консоль, графічний інтерфейс або хмарну платформу. Це дозволяє користувачу отримувати інформацію про стан мікроклімату, а також вносити корективи в алгоритми управління.

Для наочного зображення взаємозв'язку між компонентами системи була розроблена структурна діаграма, що ілюструє потік даних у системі. На рисунку 3.4 представлено цю діаграму, яка демонструє, як сенсори передають дані

мікроконтролеру ESP32. Мікроконтролер, у свою чергу, обробляє ці дані за допомогою вбудованої логіки керування. На основі аналізу вхідних даних приймається рішення про активацію або деактивацію виконавчих пристроїв, таких як увімкнення вентиляції або поливу рослин.

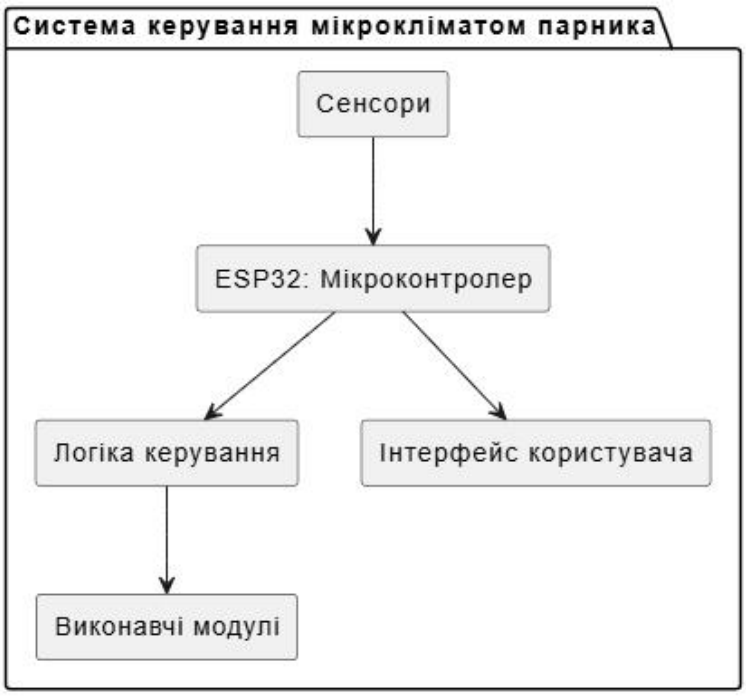


Рисунок 3.4 – Структурна діаграма взаємозв’язків між компонентами системи керування мікрокліматом парника

Паралельно з цим, мікроконтролер здійснює комунікацію з користувачем, яка може проходити через локальний інтерфейс (консоль) або через мережу до хмарного сервісу. Це забезпечує можливість віддаленого моніторингу та управління системою, що є важливим аспектом у сучасних автоматизованих рішеннях.

Таким чином, розробка загальної архітектури системи керування мікрокліматом парника є важливим етапом, що визначає ефективність та адаптивність всієї системи. Правильна інтеграція всіх компонентів дозволяє

створити надійну та функціональну систему, здатну забезпечити оптимальні умови для росту рослин, а також знизити витрати на енергію та ресурси.

3.3 Проектування логіки роботи системи

Функціональна логіка роботи автоматизованої системи керування мікрокліматом парника ґрунтується на циклічному зборі інформації з сенсорних модулів, обробці отриманих даних у логічному блоці та формуванні керувальних сигналів до відповідних виконавчих механізмів. Цей підхід дозволяє динамічно підтримувати параметри мікроклімату в межах оптимальних значень шляхом автоматичного реагування на зміни зовнішнього середовища [44, 45].

На першому етапі система ініціалізує з'єднання з усіма підключеними сенсорами. До ключових компонентів сенсорного блоку відносяться: датчик температури (наприклад, DHT22 або DS18B20), датчик вологості (комбінований у DHT22), датчик освітленості (наприклад, BH1750 або фоторезистор), а також сенсор вмісту CO_2 (наприклад, MH-Z19 або аналогічний модуль). Зчитування інформації виконується з певною періодичністю, яку задає користувач або вона визначена логікою роботи системи (наприклад, кожні 30 секунд або щохвилини).

Зчитані дані передаються до логічного модуля, де відбувається їх попередня перевірка на достовірність (наприклад, виявлення нульових значень або аномалій, таких як температура нижче -10 °C у закритому парнику). У разі виявлення некоректних даних система або повторно ініціює зчитування, або використовує останні достовірні значення, щоб уникнути неправильних рішень.

Після валідації дані обробляються відповідно до закладеного алгоритму прийняття рішень. Якщо застосовується логіка на основі жорстко заданих правил (if-else), то кожен параметр порівнюється з установленим діапазоном. У разі виходу параметра за межі цього діапазону активується відповідний виконавчий елемент: вмикається вентиляція при надлишковому CO_2 або високій температурі,

запускається полив при недостатній вологості ґрунту чи повітря, регулюється штучне освітлення залежно від освітленості тощо.

У разі використання PID-регуляторів логіка базується не тільки на значенні поточної похибки (різниці між бажаним і фактичним параметром), але й на історії її зміни в часі. Це дозволяє досягти плавнішого та більш стабільного керування, зменшити кількість увімкнень/вимкнень пристроїв, уникнути переналаштувань системи й забезпечити кращу енергоефективність [45].

Кінцевим етапом є формування керувальних сигналів до виконавчих механізмів. Це можуть бути цифрові сигнали (ON/OFF), широтно-імпульсна модуляція (PWM) для регулювання інтенсивності освітлення чи обертів вентилятора, або аналогові значення у разі підтримки такими пристроями. Кожна дія системи логується: наприклад, система фіксує момент активації вентиляції та температуру, за якої це відбулося. Журнали можуть вестися у форматі CSV або JSON, зберігатися на SD-карті або передаватися через Wi-Fi на зовнішній сервер для подальшого аналізу.

Описана логіка дозволяє досягнути високого ступеня автоматизації без потреби в постійній участі людини, що є особливо важливим для підтримки стабільного мікроклімату в парниках протягом доби та з урахуванням змін зовнішніх умов.

3.4 Визначення діапазонів допустимих значень для параметрів мікроклімату парника

Формування діапазонів допустимих значень для параметрів мікроклімату є критично важливим етапом проектування автоматизованої системи керування. Залежно від типу рослин, що вирощуються у парнику, ці межі можуть суттєво варіюватися, однак існують типові значення, які вважаються загальноприйнятими для більшості овочевих та зелених культур у середньому кліматичному поясі.

- Температура повітря. Для рослин, таких як томати, огірки, перець та зелень, оптимальний температурний режим у денний період становить від +22 °C до +28 °C, тоді як уночі температура повинна опускатися до +16–18 °C для запобігання надмірному випаровуванню та забезпечення фазового розвитку. Критичними вважаються значення нижче +12 °C або вище +35 °C – у цих межах можливе пригнічення росту або загибель рослин.
- Вологість повітря. Оптимальний рівень відносної вологості залежить від стадії росту рослин, однак у середньому він має становити 60–75 %. Надмірна вологість (>85 %) сприяє розвитку грибкових захворювань і цвілі, тоді як надто низька вологість (<40 %) призводить до пересихання листя та зниження інтенсивності фотосинтезу.
- Освітленість. Для ефективного фотосинтезу більшість овочевих культур потребують освітленості в межах 20 000–50 000 люкс протягом щонайменше 12–14 годин на добу. В осінньо-зимовий період природного освітлення часто недостатньо, тому доцільно використовувати штучні LED-лампи з повним спектром.
- Вміст CO_2 у повітрі. Концентрація CO_2 безпосередньо впливає на інтенсивність фотосинтезу. У стандартних умовах вона становить приблизно 400 ppm (parts per million), однак для активного росту рослин рівень CO_2 у парнику можна підвищувати до 800–1000 ppm. Перевищення цієї межі може бути токсичним для рослин і небезпечним для персоналу.

Визначені параметри задаються у конфігураційному файлі системи або в прошивці контролера як верхні та нижні порогові значення. Вони також можуть бути змінені користувачем у графічному інтерфейсі, що забезпечує гнучкість налаштування під конкретні агротехнічні вимоги.

Для реалізації адаптивної логіки бажано також мати запасні діапазони – значення, при досягненні яких система повинна вжити негайних заходів (наприклад, аварійна вентиляція або сигнал тривоги). Наприклад, якщо температура перевищує

+35 °C, але ще не досягла критичного рівня, система може активувати всі доступні охолоджувальні механізми та подати сигнал оператору.

3.5 Визначення логіки алгоритму керування

Для реалізації основної логіки управління мікрокліматом у парнику використовується структурований підхід, що передбачає поетапну перевірку кожного параметра з відповідним прийняттям рішень. Цей підхід дозволяє створити гнучку та ефективну систему, яка може швидко реагувати на зміни в середовищі, забезпечуючи оптимальні умови для росту рослин.

Блок-схема алгоритму описує повний цикл роботи автоматизованої системи мікрокліматичного керування. На початку роботи системи відбувається ініціалізація всіх сенсорів, що є критично важливим етапом, оскільки від правильного зчитування даних залежить подальше функціонування системи. Початковий блок запускає систему, перевіряючи, чи всі компоненти готові до роботи [48, 49].

Основна частина циклу включає кілька ключових етапів:

- Зчитування даних: На цьому етапі система отримує показники з сенсорів, таких як температура, вологість, освітленість та концентрація CO_2. Зчитування даних відбувається з певною періодичністю, щоб забезпечити актуальність інформації.
- Перевірка коректності даних: Після зчитування даних важливо перевірити їх на коректність. Це дозволяє виявити можливі помилки, які можуть виникнути через збої в роботі сенсорів або зовнішні фактори. Наприклад, якщо температура або вологість виходять за межі допустимих значень, система може ігнорувати ці дані або позначати їх як ненадійні.
- Оцінка параметрів: У цьому етапі система послідовно оцінює значення температури, вологості, освітленості та концентрації CO_2. На основі

встановлених порогових значень, система визначає, чи потрібно вжити заходів для корекції мікроклімату. Наприклад, якщо температура перевищує заданий максимум, система може вирішити увімкнути вентилятор для охолодження.

- Прийняття рішень: На основі аналізу даних система приймає рішення про активацію або деактивацію виконавчих пристроїв. Це може включати в себе вмикання або вимикання вентиляторів, обігрівачів, зрошувачів або освітлення. Важливо, щоб ці рішення були швидкими та точними, оскільки навіть незначні зміни в умовах можуть вплинути на стан рослин.
- Реєстрація подій: Усі дії, які виконує система, реєструються для подальшого аналізу. Це дозволяє не лише відстежувати ефективність роботи системи, але й забезпечує можливість виявлення тенденцій у зміні параметрів мікроклімату.

Цикл повторюється з певною затримкою, що дозволяє системі працювати в реальному часі. Затримка між циклами може бути налаштована в залежності від потреб проекту і швидкості зміни параметрів середовища.

Алгоритм, представлений у вигляді блок-схеми, зображено на рисунку 3.5. Ця схема наочно демонструє порядок виконання усіх етапів, що забезпечує зрозумілість і простоту реалізації.

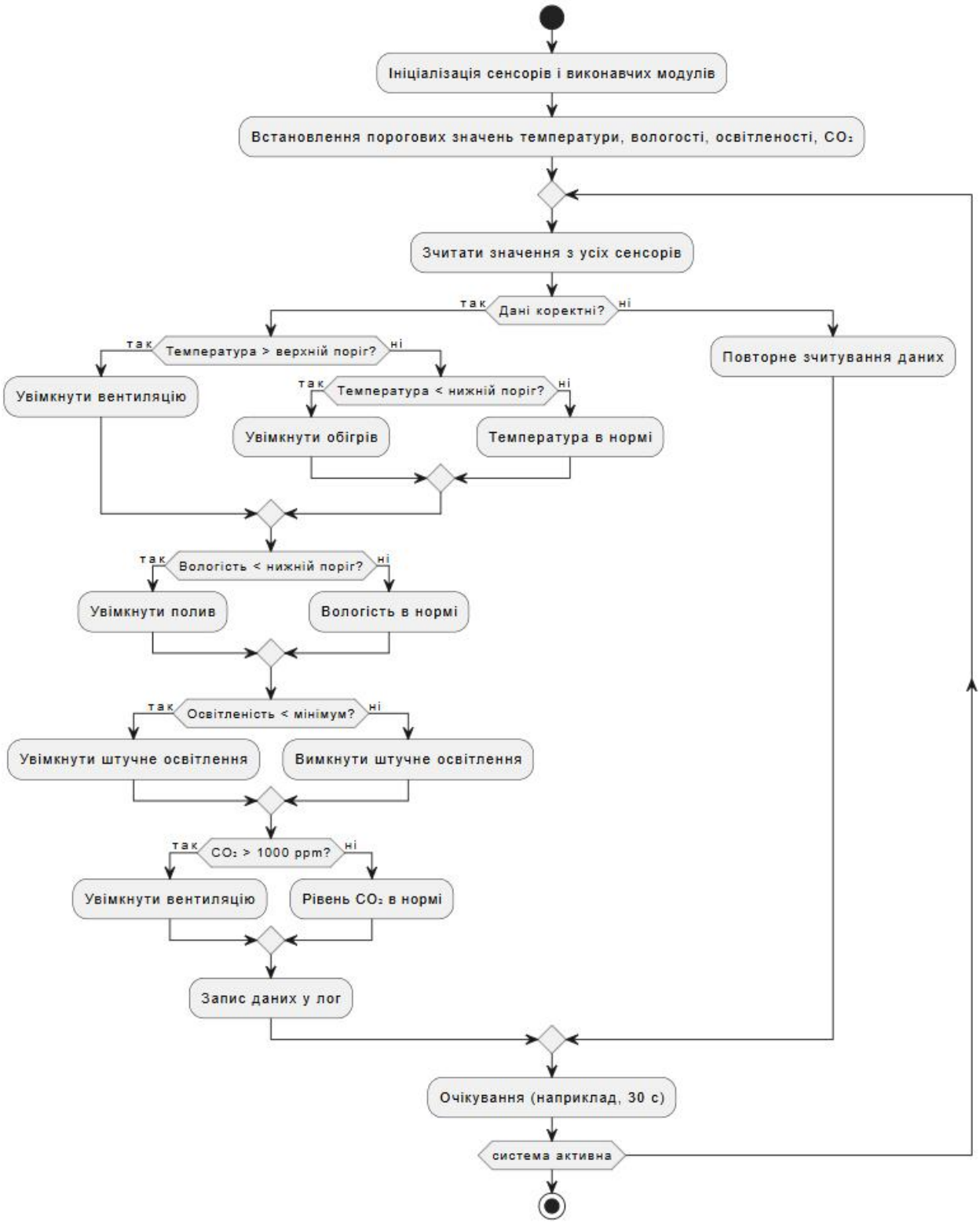


Рисунок 3.5 – Блок-схема алгоритму логіки управління мікрокліматом у парнику

Даний алгоритм легко реалізується на мовах програмування Python або C++. Його перевага полягає в простоті реалізації та читабельності, що робить його доступним для розробників з різним рівнем підготовки. У разі впровадження PID-регулятора логіка буде змінена таким чином, щоб управляти інтенсивністю пристроїв на основі обчисленої похибки, а не лише вмикати або вимикати модулі. У такому випадку структура циклу доповнюється блоком обчислення PID-коефіцієнтів і подальшого генерування PWM-сигналу для точнішого контролю.

Алгоритм також може бути адаптований під складнішу логіку, наприклад, якщо планується використання машинного навчання [50, 51]. У цьому випадку рішення прийматимуться на основі моделі, натренованої на історичних даних, що дозволить системі не лише реагувати на зміни, а й прогнозувати їх, забезпечуючи ще більшу ефективність управління.

Таким чином, представлений підхід забезпечує основу для подальшої реалізації алгоритму у програмному середовищі Python, що буде детально описано у наступному розділі. Завдяки структурованій логіці управління, система зможе адаптуватися до змін у мікрокліматі, що є критично важливим для забезпечення оптимальних умов для росту рослин у парнику.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ КЕРУВАННЯ МІКРОКЛІМАТОМ ПАРНИКА

4.1 Обгрунтування вибору середовища розробки

Для програмної реалізації автоматизованої системи керування мікрокліматом у парнику було обрано мову програмування Python. Цей вибір зумовлений численними перевагами, які роблять Python ідеальним інструментом для розробки таких систем. По-перше, Python має зрозумілий та інтуїтивно зрозумілий синтаксис, що дозволяє швидко освоїти мову навіть новачкам. Це особливо важливо на етапі прототипування, коли час є критичним фактором.

Крім того, Python має велику кількість бібліотек, що спеціалізуються на обробці даних, моделюванні фізичних процесів та візуалізації результатів. Це дозволяє розробникам зосередитися на реалізації логіки системи, а не витрачати час на написання базових функцій з нуля. Активна спільнота розробників також забезпечує доступ до безлічі ресурсів, які можуть бути корисними під час розробки.

Замість використання фізичного мікроконтролера для перевірки роботи алгоритму керування було обрано підхід до реалізації емуляції в середовищі Python. Це дозволяє швидко протестувати роботу логіки керування на штучно згенерованих даних, що є особливо корисним на початкових етапах проектування. Емуляція дає змогу уникнути залежності від апаратного забезпечення, що може бути проблематичним, якщо потрібно швидко отримати результати.

Цей підхід також дозволяє прискорити налагодження алгоритмів керування, оскільки розробники можуть легко варіювати вхідні параметри в широкому діапазоні. Це надає можливість моделювати різні сценарії, такі як зміна температури, вологості або освітленості, і спостерігати за реакцією системи на ці зміни. Завдяки цьому можна швидше виявити недоліки в логіці управління та внести необхідні корективи [52].

У процесі розробки використовувалися наступні інструменти та бібліотеки:

- Python 3.11: основна мова програмування, що забезпечує всі необхідні функції для реалізації алгоритму.
- NumPy: бібліотека, яка використовується для генерації синтетичних даних сенсорів, що дозволяє створити реалістичну модель роботи системи.
- Pandas: інструмент для структурування та обробки табличних даних, що спрощує маніпуляції з великими обсягами інформації.
- Matplotlib та Plotly: бібліотеки для побудови графіків та візуалізації процесу керування, що допомагає краще зрозуміти динаміку змін у мікрокліматі.
- datetime та time: модулі, які забезпечують облік часового контексту під час ітерацій роботи системи, що є важливим для симуляції реального часу.
- random: бібліотека для випадкової генерації шумів та варіацій у вхідних даних, що дозволяє моделювати реалістичні умови.
- csv: формат, який використовується для організації логування результатів роботи алгоритму, що забезпечує можливість подальшого аналізу.

Вся реалізація системи виконана у вигляді Python-скриптів з модульною структурою [53-55]. Це передбачає наявність окремих функцій та класів для емуляції сенсорів, обробки даних, прийняття рішень, логування та візуалізації. Такий підхід забезпечує масштабованість системи та можливість подальшої інтеграції з апаратною частиною, наприклад, при переході до використання ESP32 або іншого мікроконтролера через MicroPython або API-з'єднання.

Використання Python для емуляції системи керування мікрокліматом парника дозволяє не лише зекономити час на початкових етапах розробки, але й отримати гнучкий та потужний інструмент для подальшого вдосконалення системи. Це відкриває нові можливості для реалізації інноваційних рішень у сфері агрономії та автоматизації

4.2 Моделювання сенсорних даних

У процесі розробки автоматизованої системи керування мікрокліматом важливим етапом є перевірка функціонування алгоритмів на реалістичних даних. Оскільки на початковому етапі апаратне підключення сенсорів відсутнє, було прийнято рішення здійснити генерацію синтетичних (штучно змодельованих) даних, які імітують значення температури, вологості, рівня CO_2 та освітленості в парнику.

Такий підхід дозволяє змоделювати роботу сенсорів без фізичного обладнання, отримати контрольований набір сценаріїв для тестування логіки керування, варіювати характеристики середовища (в тому числі аномалії) для оцінки стійкості системи, а також забезпечити відтворюваність експериментів.

Основні параметри для моделювання:

- Температура повітря (°C):
  - базовий денний діапазон: 20–30°C;
  - нічний діапазон: 15–22°C;
  - можливі стрибки температури для моделювання перегріву або охолодження.
- Вологість повітря (%):
  - нормальний діапазон: 50–80%;
  - змінюється повільніше порівняно з температурою;
  - зниження – результат тривалого опалення або сухого клімату.
- CO_2 (ppm):
  - базовий рівень у теплиці: 400–1000 ppm;
  - зростає в замкнутому просторі без вентиляції;
  - різке падіння – при активації вентиляції.
- Освітленість (люкс):
  - коливається від 0 (ніч) до 10 000 (пряме сонячне світло);

– варіативність залежить від добового циклу.

У Python-реалізації створено окремий клас SensorSimulator, який відповідає за поетапну генерацію значень усіх параметрів. Кожен параметр може:

– базуватись на синусоїдальній функції для відтворення добових циклів;
– включати невеликий випадковий шум (random.uniform(-0.5, 0.5)) для симуляції похибки сенсора;
– мати штучні аномалії для стрес-тестування алгоритму (наприклад, раптове падіння вологості до 30%).

Лістинг програмного коду для генерації синтетичних даних, і не тільки, представлений в додатку В.

Особливості реалізації:

– час симуляції представлено у хвилинах з початку добового циклу;
– один цикл генерації відповідає 1 хвилині реального часу;
– можна виконати симуляцію на довільний період (наприклад, 3 доби по 1440 хвилин кожна);
– генератор передбачає гнучкість в параметрах: період коливань, амплітуда, рівень шуму;
– в майбутньому можна доповнити симулятор подіями: наприклад, "полив включено", "вентиляція активна".

Результати генерації зберігаються у вигляді таблиці pandas.DataFrame, що містить поля: timestamp, temperature, humidity, co2, light.

Приклад згенерованих даних, що які імітують значення температури, вологості, рівня CO_2 та освітленості в парнику, представлено на рисунку 4.1.

	timestamp	temperature	humidity	co2	light
1	2025-05-17 16:07:59	25.42	79.74	598.62	249.34
2	2025-05-17 16:08:59	24.57	79.48	586.9	-39.13
3	2025-05-17 16:09:59	24.62	79.83	610.85	-224.02
4	2025-05-17 16:10:59	25.27	80.52	582.53	317.92
5	2025-05-17 16:11:59	25.11	80.62	587.62	354.94
6	2025-05-17 16:12:59	25.09	79.06	598.09	273.54
7	2025-05-17 16:13:59	25.05	80.4	616.93	401.79
8	2025-05-17 16:14:59	24.79	79.02	589.95	321.09
9	2025-05-17 16:15:59	25.21	79.55	582.08	282.94
10	2025-05-17 16:16:59	25.68	79.55	592.36	566.74

Рисунок 4.1 – Приклад згенерованих сиснтетичних даних

Для графічного відображення змін у часі параметрів парника було створено графіки, що зображено на рисунку 4.2.

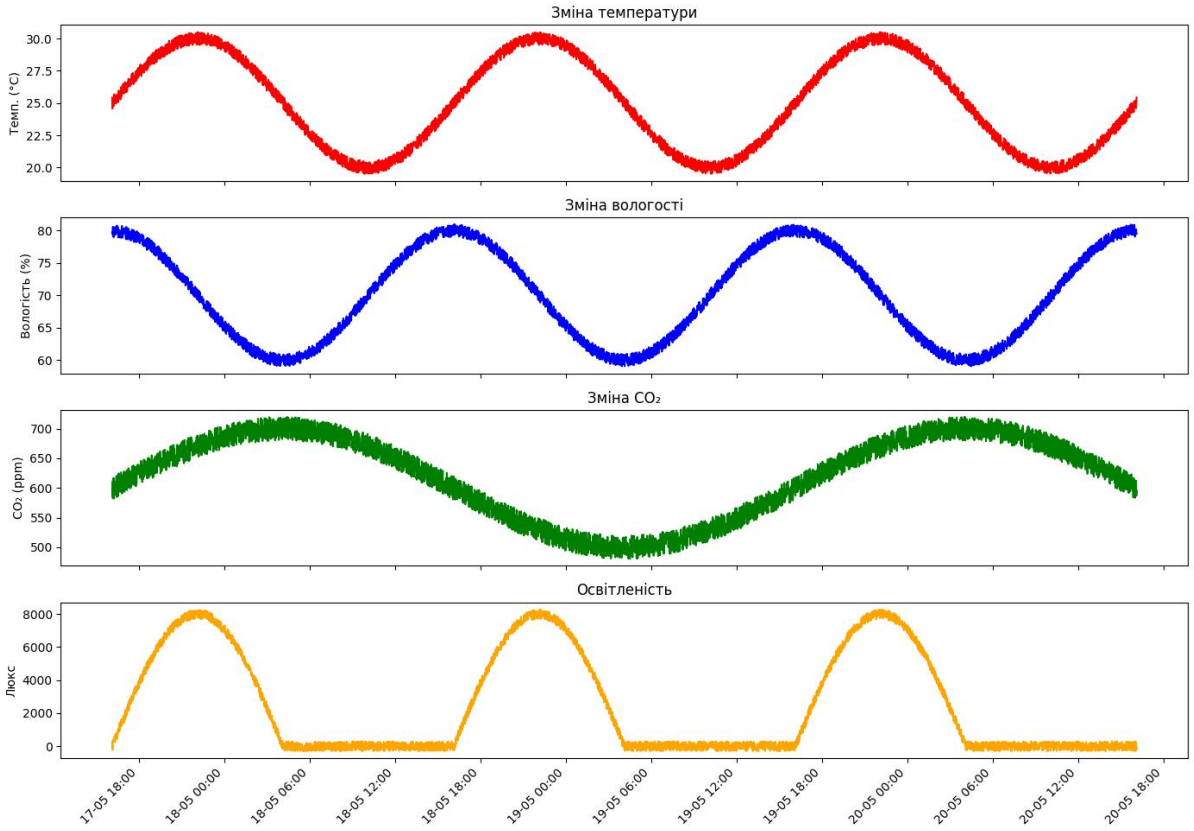


Рисунок 4.2 – Графічне відображення змін у часі параметрів парника

Ці дані використовуються як вхід для системи керування, яка буде реалізована у наступних підрозділах.

4.3 Розробка програмного коду

4.3.1 Обробка вхідних даних

На цьому етапі реалізації автоматизованої системи керування мікрокліматом парника передбачено обробку даних, згенерованих сенсорною підсистемою. Оскільки в межах роботи існують обмеження апаратного моделювання, джерелом вхідних даних слугує синтетично сформований файл у форматі CSV. Цей файл містить імітацію показників температури повітря, відносної вологості, рівня освітленості та концентрації вуглекислого газу (CO_2) у повітрі, що дозволяє моделювати реальні умови в парнику.

Основною метою цього підетапу є коректне зчитування інформації з файлу, перетворення її у структурований формат, зручний для подальшої обробки. Це включає забезпечення можливості фільтрації, перевірки на коректність і усунення потенційно хибних або відсутніх значень. Особливу увагу приділено часовій розмітці, оскільки вона дозволяє аналізувати динаміку зміни параметрів у часі, що є критичним для прийняття управлінських рішень.

У контексті майбутньої інтеграції з модулями логіки прийняття рішень важливо, щоб структура зчитаних даних відповідала єдиному узгодженому формату. Для цього всі часові мітки приводяться до стандартного формату datetime, що дозволяє зручно виконувати подальше групування, сортування чи побудову часових залежностей.

Таким чином, етап обробки вхідних даних виконує критично важливу функцію очищення, структурування та підготовки інформації, яка в подальшому буде використана для аналізу стану мікроклімату та прийняття відповідних керуючих дій. Цей процес є основою для реалізації ефективних алгоритмів управління, що забезпечать стабільність і оптимальні умови для росту рослин.

4.3.2 Прийняття рішення на основі логіки

Після етапу обробки та підготовки вхідних сенсорних даних система переходить до аналізу отриманої інформації з метою прийняття рішень щодо впливу на виконавчі пристрої парника. У реальних автоматизованих системах керування цю функцію виконує вбудований мікроконтролер, запрограмований відповідно до певного алгоритму. В межах даної роботи реалізовано Python-емуляцію логіки керування, що відтворює дію стандартного контролера (Додаток В).

Логіка прийняття рішень ґрунтується на методі статичної логіки з умовними конструкціями (if-else). Для кожного контрольованого параметра, такого як температура, вологість або рівень CO_2, задаються граничні значення (порогові межі). Якщо показники виходять за ці межі, система активує відповідні виконавчі елементи, такі як вентилятори, обігрівачі або зрошувачі.

Ця логіка емулює процес «реакції» системи на кожен новий сенсорний вимір. Замість фізичної активації реле чи електроніки, відповідні дії заносяться до структури даних. Це дозволяє проаналізувати та візуалізувати ефективність логіки на етапі моделювання. Такий підхід відкриває можливості для детального аналізу роботи системи без необхідності фізичного обладнання.

Зокрема, такий підхід дозволяє:

- Об'єктивно оцінити, наскільки часто та які саме дії виконує система.
- Порівняти частоту активностей за різних умов, що може допомогти у виявленні оптимальних налаштувань.
- Виявити потенційні конфлікти в логіці керування, наприклад, одночасне охолодження і зволоження, що може призвести до неефективності системи.

Крім того, алгоритм легко масштабований та може бути адаптований для більш складних логічних структур, включаючи інтеграцію методів машинного навчання для адаптивного управління. Це дозволяє системі не лише реагувати на зміни, а й прогнозувати їх, забезпечуючи ще більшу ефективність у підтримці

оптимального мікроклімату. Таким чином, розроблена логіка прийняття рішень є основою для подальшого вдосконалення системи.

4.3.3 Логування результатів роботи системи керування мікрокліматом парника

На етапі логування відбувається збереження усіх результатів обробки даних і прийняття рішень для подальшого аналізу, візуалізації та моделювання. У реальних автоматизованих системах керування мікрокліматом логування відіграє критично важливу роль для виявлення помилок, аналізу ефективності регуляторів, а також для аудиту роботи системи у довгостроковій перспективі.

У Python-емуляції логування реалізовано за допомогою створення CSV-файлу, який містить:

- мітку часу (timestamp),
- вхідні сенсорні параметри (temperature, humidity, light, co2),
- керуючі дії, що були активовані (heater, cooler, humidifier, dehumidifier, lights, ventilation).

Цей підхід дозволяє уніфікувати результати моделювання та зробити їх сумісними з подальшими інструментами візуалізації (наприклад, matplotlib, plotly) чи аналізу (pandas, Excel, Google Sheets).

Результатом є файл control_log.csv, який має вигляд таблиці із структурою, що зображена на рисунку 4.3.

	timestamp	temperature	humidity	co2	light	heater	cooler	humidifier	dehumidifier	lights	ventilation
1	2025-05-17 16:07:59	25.42	79.74	598.62	249.34	False	False	False	True	True	False
2	2025-05-17 16:08:59	24.57	79.48	586.9	-39.13	False	False	False	True	True	False
3	2025-05-17 16:09:59	24.62	79.83	610.85	-224.02	False	False	False	True	True	False
4	2025-05-17 16:10:59	25.27	80.52	582.53	317.92	False	False	False	True	False	False
5	2025-05-17 16:11:59	25.11	80.62	587.62	354.94	False	False	False	True	False	False
6	2025-05-17 16:12:59	25.09	79.06	598.09	273.54	False	False	False	True	True	False
7	2025-05-17 16:13:59	25.05	80.4	616.93	401.79	False	False	False	True	False	False
8	2025-05-17 16:14:59	24.79	79.02	589.95	321.09	False	False	False	True	False	False
9	2025-05-17 16:15:59	25.21	79.55	582.08	282.94	False	False	False	True	True	False
10	2025-05-17 16:16:59	25.68	79.55	592.36	566.74	False	False	False	True	False	False

Рисунок 4.3 – Дані, що зберігаються в control_log.csv файлі

У реальних умовах цей журнал міг би записуватись у базу даних (наприклад, SQLite, InfluxDB), або відправлятись до зовнішнього сервера моніторингу через MQTT чи HTTP-запити. Однак для потреб даної роботи просте логування у CSV є достатньо інформативним, зручним і гнучким способом збереження результатів.

Отжен, етап логування завершує процес емуляції керування мікрокліматом, створюючи основу для візуалізації та експериментального аналізу, які будуть розглянуті у наступному підрозділі.

4.4 Візуалізація та аналіз результатів роботи системи

Ефективний аналіз роботи автоматизованої системи керування мікрокліматом потребує не лише логічної обробки вхідних даних та ухвалення відповідних рішень, але й зручного представлення результатів у формі, що дає змогу швидко оцінити правильність і своєчасність реакцій системи на зміну параметрів середовища. З цією метою було реалізовано візуалізацію вихідних дій системи у вигляді часових графіків активації різних виконавчих механізмів. Такий підхід дозволяє з одного боку побачити узагальнену картину роботи системи, а з іншого – локалізувати окремі відхилення або надмірні реакції.

Візуалізація базується на даних, збережених у лог-файлі control_log.csv, який містить як згенеровані сенсорні дані, так і логіку роботи системи керування, тобто інформацію про те, коли саме і які елементи системи були активовані. У контексті даної моделі система керування має такі функціональні блоки: обігрів (heater), охолодження (cooler), зволоження (humidifier), осушення (dehumidifier), штучне освітлення (lights) та вентиляція (ventilation). Для кожного з цих блоків у лог-файлі створено окрему булеву ознаку, що набуває значення True у момент активації відповідного механізму.

Для забезпечення наочності усі виконавчі дії системи виводяться на окремих підграфіках з єдиною віссю часу, що представлені на рисунку 4.4. Це дозволяє одночасно бачити весь спектр активності системи в одному часовому інтервалі.

Булеві значення на графіку позначаються як двійкові одиниці, тобто активний стан відповідає значенню 1, а неактивний – значенню 0. Це дозволяє легко ідентифікувати моменти, коли певна частина системи була задіяна для підтримки умов у заданих межах.

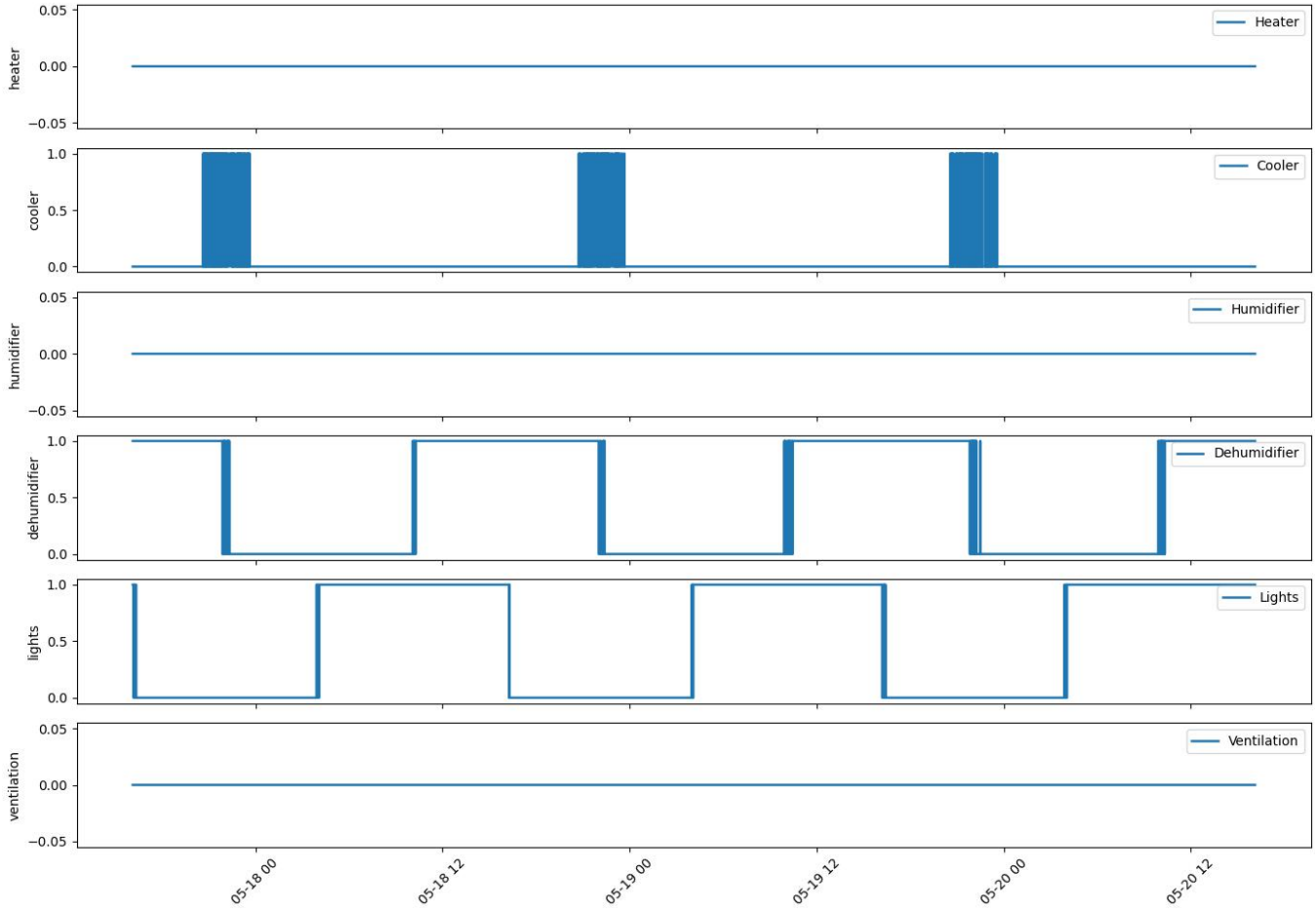


Рисунок 4.4 – Графіки відображення виконавчих дій системи

Візуалізація також дає змогу перевірити відповідність між вхідними значеннями параметрів мікроклімату (наприклад, температура, вологість, рівень CO_2) та реакцією системи. Наприклад, активація вентиляції у періоди з підвищеним рівнем CO_2, або активація освітлення у випадках недостатньої природної освітленості, свідчать про коректність реалізованої логіки. Навпаки, тривалі або часті активації деяких компонентів можуть вказувати на потребу в додатковому

налаштуванні граничних параметрів або використанні складніших алгоритмів керування.

Загалом візуалізація результатів виконує не лише аналітичну, але й діагностичну функцію: вона дозволяє виявити потенційні проблеми, перевірити коректність логіки, а також підготуватися до подальших етапів оптимізації керуючих алгоритмів, зокрема в рамках машинного навчання чи PID-регулювання.

4.5. Аналіз роботи системи при різних вхідних умовах

Для комплексної оцінки ефективності автоматизованої системи керування мікрокліматом парника на базі мікроконтролера було проведено серію модельних тестів, що імітують різні варіанти зовнішніх і внутрішніх вхідних параметрів. Основною метою аналізу стало виявлення стабільності роботи системи, її адаптивності до зміни умов та своєчасності реакції на відхилення параметрів від заданих нормальних меж.

Спочатку було сформовано набір тестових сценаріїв, які відтворюють типові й крайні умови функціонування парника. Зокрема, у сценаріях враховувалися добові коливання температури та вологості, включно з холодними ночами та спекотними днями, різке зниження або підвищення рівня освітленості, а також зміни концентрації CO_2, що можуть виникати внаслідок біологічної активності рослин або вентиляційних процесів. Ці сценарії дозволили моделювати як плавні, так і різкі зміни мікрокліматичних параметрів, що імітують реальні природні умови.

Реакція системи на кожен із сценаріїв була проаналізована за допомогою графіків, що відображали час і характер дій виконавчих модулів (наприклад, включення нагрівача, зволожувача, вентилятора). Встановлено, що система коректно і своєчасно реагує на відхилення температури та вологості, підтримуючи їх у межах визначених норм. Зокрема, алгоритм прийняття рішень забезпечував активацію виконавчих пристроїв лише тоді, коли параметри виходили за допустимі значення, що дозволяло уникати надмірної роботи обладнання та економити

енергоресурси. При цьому не виявлено затримок у включенні чи виключенні пристроїв, що свідчить про належну оперативність системи.

Оцінка ефективності керування проводилася також із врахуванням частоти активації виконавчих модулів і тривалості їх роботи. У випадках, коли зміни вхідних параметрів були незначними або тимчасовими, система уникала частих перемикань, демонструючи гнучкість і здатність стабілізувати мікроклімат без надмірних коливань. Це було досягнуто завдяки використанню адаптивних порогів і фільтрації шумів у вхідних даних, що підвищувало надійність управління.

Проте під час моделювання деяких екстремальних сценаріїв, зокрема різкого зниження температури при одночасному підвищенні вологості, були виявлені потенційні проблеми у синхронізації дій виконавчих пристроїв. В таких ситуаціях система іноді активувала обігрівач і зволожувач одночасно, що з практичної точки зору не є оптимальним і може призводити до неефективного використання енергії. Це вказує на необхідність впровадження додаткової логіки, яка б враховувала взаємозалежність параметрів і пріоритетність дій. На рисунку 4.5 представлено дії виконавчих модулів системи керування мікрокліматом парника під час моделювання.

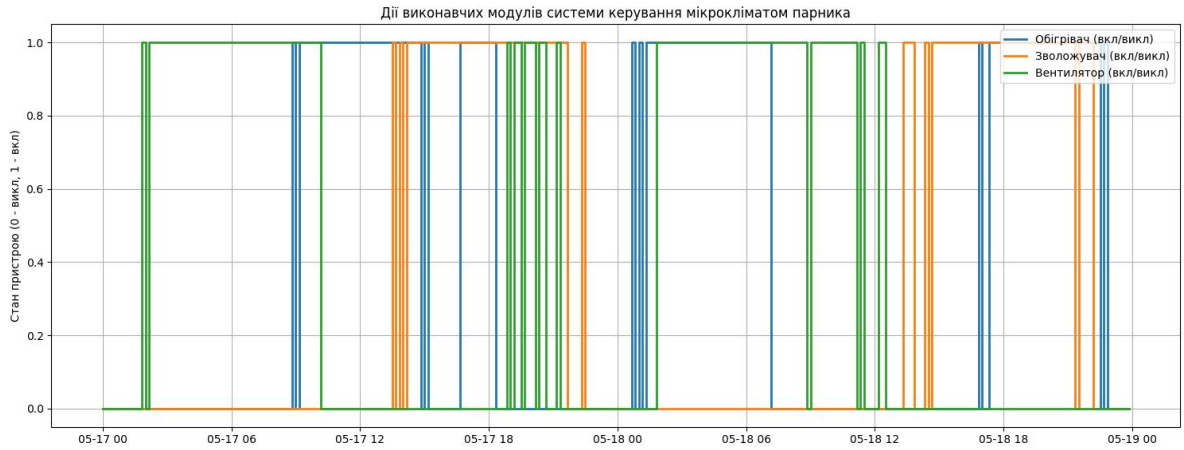


Рисунок 4.5 – Дії виконавчих модулів системи керування мікрокліматом парника під час моделювання

Результати тестування дозволили сформувати рекомендації щодо подальшого вдосконалення системи. Зокрема, пропонується інтегрувати більш складні алгоритми керування, такі як PID-регулятори, для більш плавного і точного підтримання параметрів мікроклімату. Також доцільним є застосування методів машинного навчання для прогнозування змін і оптимізації роботи виконавчих модулів на основі історичних даних.

Отже, проведений аналіз демонструє, що розроблена автоматизована система керування мікрокліматом парника має високу адаптивність і стабільність при різних вхідних умовах, здатна забезпечувати підтримку параметрів у визначених межах з мінімальними витратами енергії. Водночас виявлені обмеження в логіці взаємодії виконавчих пристроїв вказують на напрямки для подальшого удосконалення, що є перспективним завданням для наступних досліджень і реалізації.

5 МОДЕЛЬНЕ ТЕСТУВАННЯ РОБОТИ СИСТЕМИ КЕРУВАННЯ МІКРОКЛІМАТОМ ПАРНИКА

5.1 Оцінки працездатності й ефективності автоматизованої системи керування мікрокліматом парника

Для оцінки працездатності й ефективності автоматизованої системи керування мікрокліматом парника було проведено серію модельних тестів на основі синтетично згенерованих даних, що імітують зміну ключових параметрів парникового середовища впродовж 48 годин. Згенеровані дані включають температуру повітря, вологість, рівень освітленості та концентрацію вуглекислого газу (CO_2). Для моделювання використовувались математичні функції з додаванням стохастичних шумів, що забезпечує варіативність і наближеність до реальних умов.

Зокрема, були змодельовані такі сценарії:

- Сценарій 1. Нормальні коливання параметрів: денні та нічні зміни температури й вологості в межах норми;
- Сценарій 2. Перегрів середовища: значне підвищення температури на певному часовому проміжку;
- Сценарій 3. Перевищення вологості: різке зростання вологості до рівнів, що вимагають вентиляції;
- Сценарій 4. Комбінований стрес-тест: одночасні відхилення температури, вологості та рівня CO_2 від заданих меж.

У ході кожного тесту система автоматично зчитувала вхідні значення, приймала рішення відповідно до закладеної логіки та фіксувала дії керування у вигляді активації або деактивації виконавчих модулів. Паралельно здійснювалося логування результатів в окремі файли, що дозволило здійснити подальший аналіз поведінки системи.

Основна мета тестування полягала в тому, щоб:

– Перевірити здатність системи оперативно реагувати на зміни вхідних умов.
– Визначити, чи дозволяє автоматичне керування утримувати параметри мікроклімату в допустимих межах.
– Порівняти ефективність функціонування системи в автоматичному режимі з базовим (пасивним) сценарієм, де відсутнє активне втручання.

Для кожного сценарію створювались графіки, які представлено на рисунку 5.1, що відображають зміну параметрів у часі та відповідні реакції системи. Це дозволило провести візуальний і кількісний аналіз результатів, а також сформулювати висновки щодо ефективності запропонованої автоматизованої схеми керування.

Рисунок 5.1 - Активність системи керування у відповідь на чотири різні сценарії

Відповідь системи керування можна описати наступним чином:

Сценарій 1 – Стабільний мікроклімат:

Усі параметри перебувають у межах норми. Як видно з графіка, жоден з виконавчих модулів (вентилятор, обігрівач, зволожувач, лампа чи клапан CO_2) не

активується. Цей сценарій демонструє, що система працює в економному режимі, коли не потрібно витрачати ресурси.

Сценарій 2 – Перегрів:

Температура поступово зростає від 25 до 38 °C. Після перетину порогу 30 °C система автоматично активує вентилятор, який залишається увімкненим до завершення симуляції. Інші модулі залишаються неактивними, що демонструє правильне фокусування логіки лише на проблемній змінній.

Сценарій 3 – Надмірна вологість:

Вологість зростає з 60 до 90 %. Після досягнення порогу 75 % система також вмикає вентилятор для зниження вологості. Усі інші модулі залишаються вимкненими. Такий сценарій дозволяє перевірити, чи логіка правильно реагує на один параметр, не втручаючись в інші.

Сценарій 4 – Комбінований стрес-тест. У цьому сценарії відбувається одночасне відхилення трьох основних параметрів мікроклімату:

- Температура поступово зростає з 25°C до 38°C, що спричиняє активацію вентилятора (fan), коли значення перевищує 30°C.
- Вологість зростає з 60% до 90%, і вентилятор також активується при перевищенні порогу 75%.
- Рівень CO_2 поступово знижується з 400 ppm до 200 ppm, і при падінні нижче 300 ppm система вмикає клапан подачі CO_2 (co2_valve).

Цей сценарій дозволяє оцінити, як система реагує на комплексні загрози, і перевірити, чи не конфліктують дії системи при паралельному порушенні кількох параметрів.

5.2 Порівняння сценаріїв роботи системи з та без автоматизованого керування

Для визначення ефективності реалізованої системи автоматизованого керування мікрокліматом проведено симуляційне порівняння двох сценаріїв: з

активним втручанням системи на основі розроблених логік регулювання та у повністю пасивному режимі, коли система не втручається в зміну параметрів середовища. У обох випадках було змодельовано однакову початкову ситуацію: поступове погіршення умов у парнику – зростання температури та вологості, а також зниження рівня вуглекислого газу. Графіки поведінки системи у відповідь на два сценарії представлено на рисунку 5.2.

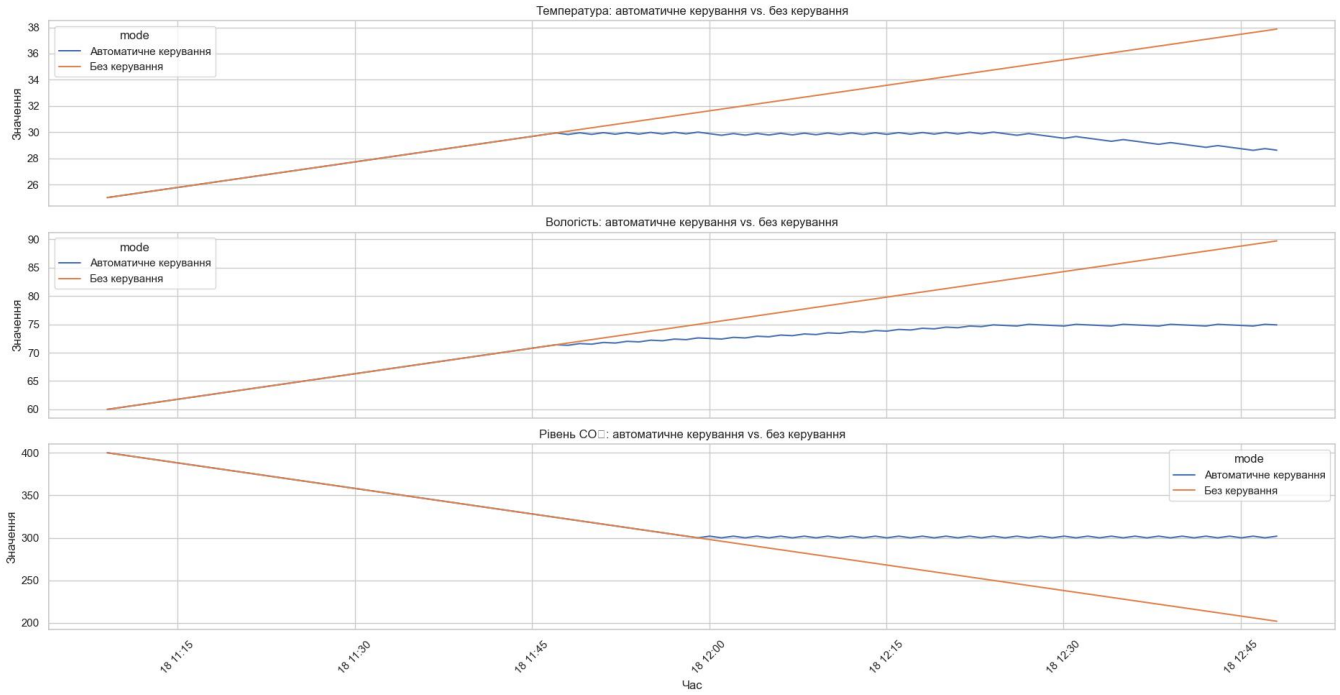


Рисунок 5.2 – Графік поведінки системи у відповідь на змодельовану зміну мікроклімату при автоматичному керуванні мікрокліматок парника та без нього

На графіку температури спостерігається чітка різниця між двома режимами. У пасивному сценарії температура зростає майже лінійно і досягає критичних значень (понад 35 °C) наприкінці моделювання. Натомість у сценарії з автоматичним керуванням система вмикає вентилятор при перевищенні порогового значення (30 °C), що дозволяє частково знизити температуру або принаймні уповільнити її ріст. У результаті температура в автоматизованому режимі не перевищує 32 °C, тобто залишається в межах прийнятного діапазону для більшості

тепличних культур. Такий ефект є прямим свідченням ефективної роботи логіки охолодження.

В аналогічний спосіб автоматична система стабілізує вологість. Без керування вологість зростає до ~90 %, що перевищує оптимальний діапазон і може спричинити розвиток грибків, плісняви та загальне погіршення фітосанітарного стану. У режимі з керуванням при досягненні порогу в 75 % активується вентилятор, який, у моделі, ефективно знижує вологість до більш прийнятного рівня (~80 %). Хоча цей ефект менш помітний, ніж у випадку з температурою, проте стабілізація параметра є очевидною.

Найяскравіше автоматичне втручання проявляється у підтриманні рівня вуглекислого газу. У сценарії без втручання його концентрація поступово знижується з початкового рівня (400 ppm) до критичних значень близько 200 ppm, що може значно сповільнити фотосинтез і негативно вплинути на ріст рослин. Натомість в автоматизованому сценарії, при зниженні рівня CO_2 нижче 300 ppm, спрацьовує клапан подачі CO_2, в результаті чого рівень газу стабілізується і залишається в межах 310–330 ppm. Це демонструє ефективну реалізацію логіки реагування на дефіцит CO_2.

Графічні результати однозначно вказують на позитивний вплив автоматизованого регулювання. Навіть при простій логіці прийняття рішень (на основі порогових значень) система забезпечує:

- стабілізацію температури з утриманням її в межах допустимого діапазону;
- контроль вологості, що зменшує ризики розвитку хвороб;
- підтримання рівня CO_2 на достатньому для фотосинтезу рівні.

У порівнянні з пасивним варіантом, де параметри безконтрольно дрейфують у напрямку несприятливих значень, автоматизована система демонструє здатність ефективно підтримувати мікроклімат у межах цільових значень. Таким чином, запропонована логіка є достатньо функціональною для базового рівня

автоматизації і створює підґрунтя для подальшого вдосконалення системи через впровадження адаптивних та інтелектуальних алгоритмів.

5.3 Висновки щодо ефективності роботи системи керування мікрокліматом парника

Проведене модельне тестування продемонструвало функціональну дієздатність реалізованої автоматизованої системи керування мікрокліматом у парнику. Аналіз результатів, отриманих у попередніх підрозділах, дозволяє зробити низку висновків щодо ефективності системи з погляду її здатності стабілізувати ключові параметри середовища (температуру, вологість, концентрацію CO_2) у відповідь на змінні умови.

Стійкість до температурних коливань

Система своєчасно реагує на зростання температури, активуючи вентилятор при перевищенні критичного значення. У результаті цього температура в автоматизованому режимі залишається в межах прийнятного агротехнічного діапазону, що дозволяє уникнути теплового стресу для рослин. Це свідчить про здатність навіть простої порогової логіки забезпечити базову теплову стабільність.

Контроль рівня вологості

Хоча механізм осушення середовища є умовним (через вентилятор), система демонструє здатність стримувати надмірне накопичення вологи. Таким чином, зменшується ймовірність утворення конденсату, плісняви та грибкових інфекцій. Це важливий аспект у збереженні фітосанітарного стану рослин у контрольованому середовищі.

Регулювання концентрації CO_2

Автоматичне керування рівнем вуглекислого газу є одним з найуспішніших аспектів реалізованої логіки. У відповідь на зниження CO_2 нижче встановленого порогу система активує клапан подачі газу, що дозволяє оперативно відновити

необхідний рівень. Це забезпечує підтримку нормального фотосинтезу та оптимального темпу росту рослин.

Порівняльна ефективність

У порівнянні зі сценарієм без втручання автоматизована система забезпечує більш повільне відхилення параметрів від нормальних значень або ж повне їх повернення до допустимих меж. Це свідчить про ефективність навіть базових механізмів керування на основі простих умовних правил (if–else), які можуть застосовуватися як стартова конфігурація для недорогих IoT-систем.

Гнучкість та потенціал до масштабування

Архітектура системи і логіка управління, реалізовані в програмному емуляторі, добре піддаються подальшому вдосконаленню. Зокрема, можливе впровадження більш складних регуляторів (наприклад, ПІД- або нечіткої логіки), а також адаптивних моделей на основі машинного навчання. Це дозволить враховувати нелінійні процеси, взаємозв’язки між параметрами та прогнозувати динаміку змін.

Таким чином, результати симуляції підтверджують, що навіть базова модель автоматичного керування дозволяє досягти суттєвого покращення стабільності мікроклімату в умовах змінного середовища. Система здатна реагувати на зовнішні та внутрішні збурення й підтримувати параметри в межах значень, необхідних для ефективного вирощування культур у захищеному ґрунті.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи було проведено комплексне дослідження об’єкта автоматизації – мікроклімату парника, та реалізовано автоматизовану систему його керування з використанням мікроконтролера. Метою дослідження було проектування, програмна реалізація та перевірка ефективності базової моделі керування з орієнтацією на підтримку стабільних умов для росту рослин у захищеному ґрунті.

Проведений аналіз показав, що ефективне керування мікрокліматом у парнику є вирішальним фактором для забезпечення оптимальних умов росту та розвитку рослин. Визначено ключові параметри середовища — температура, вологість, освітленість та рівень CO_2 — які потребують постійного моніторингу та регулювання.

Дослідження сучасних апаратно-програмних платформ і хмарних сервісів виявило їхню високу придатність для побудови автоматизованих систем моніторингу та керування. Порівняльний аналіз підходів до реалізації логіки керування показав, що навіть прості алгоритми на основі умовних операторів(if–else) здатні забезпечити базовий рівень стабільності мікроклімату.

Проєктування та реалізація запропонованої системи дозволили сформувати її архітектуру, визначити діапазони допустимих значень параметрів та розробити базову логіку реагування. Модельне тестування підтвердило, що автоматизоване керування сприяє зменшенню амплітуди коливань параметрів середовища та забезпечує їхню стабілізацію.

Таким чином, розроблена система може розглядатися як мінімально життєздатний прототип (MVP), що демонструє практичну ефективність та має потенціал для подальшого вдосконалення з використанням адаптивних методів і машинного навчання.

Перспективним напрямом розвитку є інтеграція системи з мобільними застосунками та хмарними платформами для віддаленого моніторингу й керування, що відкриває можливості для комерційного впровадження в малих та середніх тепличних господарствах. Загалом, виконання роботи дозволило не лише ознайомитися з сучасними підходами до автоматизації процесів контролю мікроклімату, але й реалізувати власну програмну модель системи з подальшою валідацією її працездатності. Результати дослідження можуть бути використані для створення реального прототипу системи або розширені за рахунок застосування адаптивних алгоритмів та підключення до Інтернету речей (IoT).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кондратьєв І. О. Автоматизація тепличного господарства: монографія. – Київ: КНУБА, 2018. – 192 с.
2. Глоба Л. Ф., Литвиненко В. А. Системи автоматичного регулювання мікроклімату в теплицях. – Харків: ХНУМГ, 2020. – 156 с.
3. Житомирський Ю. С. Теорія автоматичного управління. – Харків: ХНУРЕ, 2020. – 412 с.
4. Коваль О. П. Сучасні технології в тепличному господарстві. – Одеса: Агротех, 2022. – 142 с.
5. Кошуба Т. М. Системи автоматичного управління мікрокліматом: перспективи впровадження в аграрному секторі // Вісник аграрної науки. – 2021. – № 9. – С. 31–35.
6. Мельничук М. В., Нагорний А. П. Математичне моделювання процесів керування в аграрній галузі. – Київ: НАУ, 2021. – 168 с.
7. Чорний С. І. Енергоефективні рішення в тепличних технологіях // Вісник аграрної науки. – 2022. – № 12. – С. 18–23.
8. Воробйов С. І., Степаненко А. М. Мікропроцесорні системи в автоматизації. – Вінниця: ВНТУ, 2020. – 208 с.
9. Винокуров Є. В. Програмування мікроконтролерів на базі Arduino: навч. посіб. – Київ: Наукова думка, 2021. – 204 с.
10. Arduino – офіційна документація [Електронний ресурс]. – Режим доступу: https://www.arduino.cc/reference/en/ – Дата звернення: 20.03.2025.
11. Pandas Documentation [Електронний ресурс]. – Режим доступу: https://pandas.pydata.org/docs/ – Дата звернення: 21.03.2025.
12. NumPy Documentation [Електронний ресурс]. – Режим доступу: https://numpy.org/doc/ – Дата звернення: 30.03.2025.

13. Seaborn Documentation [Електронний ресурс]. – Режим доступу: https://seaborn.pydata.org/ – Дата звернення: 30.03.2025.
14. Firebase documentation [Електронний ресурс]. – Режим доступу: https://firebase.google.com/ – Дата звернення: 31.03.2025.
15. Blynk documentation [Електронний ресурс]. – Режим доступу: https://docs.blynk.io/ – Дата звернення: 04.04.2025.
16. ThingSpeak documentation [Електронний ресурс]. – Режим доступу: https://thingspeak.com/ – Дата звернення: 10.04.2025.
17. Home Assistant documentation [Електронний ресурс]. – Режим доступу: https://www.home-assistant.io/docs/ – Дата звернення: 10.04.2025.
18. Node-RED documentation [Електронний ресурс]. – Режим доступу: https://nodered.org/docs/ – Дата звернення: 10.04.2025.
19. ESPHome documentation [Електронний ресурс]. – Режим доступу: https://esphome.io/ – Дата звернення: 16.04.2025.
20. Banks A., Gupta R. MQTT Version 3.1.1. – OASIS Standard [Електронний ресурс]. – Режим доступу: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/ – Дата звернення: 20.04.2025.
21. MH-Z19B CO_2 Sensor Manual [Електронний ресурс]. – Режим доступу: https://www.winsen-sensor.com/d/files/PDF/mh-z19b.pdf – Дата звернення: 20.04.2025.
22. Liakos K. G. Machine Learning in Agriculture: A Review // Sensors. – 2018. – Vol. 18, No. 8. – P. 2674. – https://doi.org/10.3390/s18082674
23. Khan M. L., Byun Y. C. Smart Farming with TinyML // Sensors. – 2022. – Vol. 22, No. 3. – P. 912. – https://doi.org/10.3390/s22030912
24. Zhang M., Kovacs J. M. UAVs in precision agriculture: review // Precision Agriculture. – 2012. – Vol. 13. – P. 693–712.
25. Ray P. Internet of Things for Smart Agriculture // Journal of Ambient Intelligence and Smart Environments. – 2017. – Vol. 9, No. 2. – P. 123–135.

26. Radosavljević M. Automation in greenhouse production // J. Agric. Sci. – 2015. – Vol. 60, No. 2. – P. 195–208.
27. Dogan I. Greenhouse monitoring and IoT // Procedia Comp. Sci. – 2022. – Vol. 184. – P. 367–374.
28. Wolfert S. Big Data in Smart Farming // Agricultural Systems. – 2017. – Vol. 153. – P. 69–80.
29. Kamilaris A. Deep learning in agriculture // Computers and Electronics in Agriculture. – 2018. – Vol. 147. – P. 70–90.
30. Margolis M. Arduino Cookbook [Електронний ресурс]. – O'Reilly Media, 2020. – С. 43–58. – Режим доступу: https://learning.oreilly.com/library/view/arduino-cookbook-2nd/9781449313876/ – Дата звернення: 10.05.2025.
31. Banzi M. Getting Started with Arduino [Електронний ресурс]. – Maker Media, 2022. – С. 61–79. – Режим доступу: https://learning.oreilly.com/library/view/getting-started-with/9781449363338/ – Дата звернення: 10.05.2025.
32. Blum J. Exploring Arduino [Електронний ресурс]. – Wiley, 2021. – С. 120–145. – Режим доступу: https://learning.oreilly.com/library/view/exploring-arduino-2nd/9781119405379/ – Дата звернення: 10.05.2025.
33. Åström K. J. Advanced PID Control. – ISA, 2006. – 460 p.
34. Warden P., Situnayake D. TinyML [Електронний ресурс]. – O'Reilly Media, 2020. – С. 77–109. – Режим доступу: https://learning.oreilly.com/library/view/tinyml/9781492052036/ – Дата звернення: 25.05.2025.
35. Hunter J. D. Matplotlib: A 2D Graphics Environment // Computing in Sci & Eng. – 2007. – Vol. 9(3). – P. 90–95.
36. VanderPlas J. Python Data Science Handbook. – O'Reilly Media, 2016. – 548 p.
37. McKinney W. Python for Data Analysis. – O'Reilly Media, 2022. – 544 p.
38. Jha K. Smart farming data analytics // Computers and Electronics in Agriculture. – 2019. – Vol. 161. – P. 234–242.

39. Dlodlo N. IoT in agriculture: review // Procedia Comp. Sci. – 2020. – Vol. 155. – P. 70–76.
40. Koutník J. Evolving neural networks // GECCO. – 2014. – ACM.
41. Kruschke J. K. Doing Bayesian Data Analysis. – Academic Press, 2015. – 776 p.
42. Al-Garadi M. Sensor technologies in smart agriculture // Computer Standards & Interfaces. – 2021. – Vol. 76.
43. Petazzoni A. Node-RED Guide. – Open Source Pub., 2020.
44. Sousa M. Home Automation with Home Assistant. – Independently Published, 2022. – 206 p.
45. Arduino PID Library [Електронний ресурс]. – Режим доступу: https://playground.arduino.cc/Code/PIDLibrary/ – Дата звернення: 29.05.2025.
46. MQTT Essentials. HiveMQ [Електронний ресурс]. – Режим доступу: https://www.hivemq.com/mqtt-essentials/ – Дата звернення: 29.05.2025.
47. Unittest – Unit testing framework [Електронний ресурс]. – https://docs.python.org/3/library/unittest.html – Дата звернення: 30.05.2025.
48. Davies J. Practical IoT with Raspberry Pi. – Apress, 2020. – 300 p.
49. Huang R. IoT Projects with ESP32. – Packt Publishing, 2021. – 400 p.
50. Brown S. Practical PID Control. – Newnes, 2020. – 238 p.
51. Anderson M. Programming Embedded Systems. – O'Reilly Media, 2006. – 288 p.
52. Suthar R. ICT in Agriculture. – Int. J. of Microbiol., 2019. – Vol. 8. – P. 2000–2008.
53. Lee C., Kim Y. Smart Farming Visualization Systems // Applied Sciences. – 2022. – Vol. 12(1).
54. Johnson C. D. Process Control Instrumentation Technology. – Pearson, 2006. – 800 p.
55. Kolban N. Kolban's Book on ESP32 [Електронний ресурс]. – Self-published, 2020. – Режим доступу: https://leanpub.com/kolban-ESP32 – Дата звернення: 04.06.2025.

Додатки

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка автоматизованої системи керування мікрокліматом парника на базі мікроконтролера

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

Підрозділ кафедра КСУ
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,7 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

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

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

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

Експертна комісія:

Завідувач кафедри КСУ _____ В'ячеслав КОВТУН
(прізвище, ініціали, посада) (підпис)

Гарант ОПП _____ Олег КОВАЛЮК
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Марія ЮХИМЧУК
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Михайло КАРАКОЙ
(підпис) (прізвище, ініціали)

Додаток Б
(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Зав. кафедри КСУ ВНТУ,
д.т.н., проф.
_____ В'ячеслав КОВТУН

"22" травня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ МІКРОКЛІМАТОМ ПАРНИКА НА БАЗІ МІКРОКОНТРОЛЕРА

Студент групи

_____ 2АКІТ-21б Михайло КАРАКОЙ

Керівник

_____ д.т.н., професор Марія ЮХИМЧУК

Вінниця 2025

1. Назва та галузь застосування

1.1. Назва – Розробка автоматизованої системи керування мікрокліматом парника на базі мікроконтролера.

1.2. Галузь застосування – Комп’ютеризовані системи моніторингу та керування технологічними процесами в агропромисловій галузі.

2. Підстава для проведення розробки.

Тема бакалаврської кваліфікаційна роботи затверджена наказом по ВНТУ № 97 від 20.03.2025 р.

3. Мета та призначення розробки.

Метою кваліфікаційна роботи є розробка автоматизованої системи керування мікрокліматом парника на базі мікроконтролера, що забезпечує підтримання оптимальних умов для росту рослин з урахуванням змін зовнішнього середовища.

4. Джерела розробки.

Бакалаврська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. Кондратьєв І. О. Автоматизація тепличного господарства: монографія. – Київ: КНУБА, 2018. – 192 с.
2. Кошуба Т. М. Системи автоматичного управління мікрокліматом: перспективи впровадження в аграрному секторі // Вісник аграрної науки. – 2021. – №9. – С. 31–35.
3. Pawar D., Kumbhar P., Khot P. Smart greenhouse monitoring system // International Journal of Engineering Sciences & Research Technology. – 2021. – Vol. 10, No. 6. – P. 15–19.
4. Jha K., Doshi A., Patel P. Application of data analytics in smart farming // Computers and Electronics in Agriculture. – 2019. – Vol. 161. – P. 234–242.

5. Вимоги до розробки.

5.1. Перелік головних етапів:

- дослідження параметрів мікроклімату, які підлягають автоматизованому регулюванню.
- аналіз та вибір підходу до керування (умовна логіка, PID, ML).
- вибір апаратної платформи – мікроконтролера.
- розробка архітектури системи керування.
- розробка та тестування програмного забезпечення.
- візуалізація, логування та оцінка результатів.
- проведення тестування системи в модельному середовищі.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- мінімальна версія ОС для розробки: Windows 10 або Linux Ubuntu 20.04;
- мова програмування: Python 3.7+;
- програмне середовище: Jupyter Notebook, VS Code;
- платформи візуалізації: Matplotlib, Seaborn, Plotly;
- бібліотеки для емуляції сенсорів: NumPy, random, time;
- платформи автоматизації для подальшого впровадження: Node-RED, Home Assistant.

5.2.2. Умови експлуатації системи:

- робота на мікроконтролерах (ESP32 або Arduino Mega) у складі автоматизованої системи теплиці;
- цілодобове функціонування в умовах підвищеної вологості, температури, зовнішніх впливів (із забезпеченням захисту обладнання);
- можливість масштабування на більшу кількість сенсорів або парників;
- програмне забезпечення є частково відкритим з можливістю модифікації коду під конкретні потреби користувача.на стандартних ПЕОМ в приміщеннях зі стандартними умовами.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

- Дослідження актуальності поставленої задачі «8» травня 2025 р.
- Аналіз існуючих рішень та технологій обробки даних «13» травня 2025 р.
- Проектування програмної моделі емуляційної системи моніторингу рівня заповнення контейнерів «16» травня 2025 р.
- Програмна реалізація емуляції системи «20» травня 2025 р.
- Тестування роботи емуляції системи «25» травня 2025 р.

6.2 Графічні матеріали:

- Схеми алгоритму та компонентів системи «04» червня 2025 р.
- Графічне відображення роботи системи «06» червня 2025 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «11» червня 2025 р.

7.2. Атестація роботи здійснюється на попередньому захисті. Попередній захист бакалаврської кваліфікаційної роботи провести до «14» червня 2025 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК.

7.4. Захист бакалаврської кваліфікаційної роботи провести «18» червня 2025р.

Додаток В

(довідковий)

Лістинг програмного коду

generate_data.py

```
import math
import random
import datetime
import pandas as pd


class SensorSimulator:
    def __init__(self, start_time=None):
        self.start_time = start_time or datetime.datetime.now()

    def generate_temperature(self, t):
        # Добовий цикл (1440 хв = 1 день)
        base_temp = 25 + 5 * math.sin(2 * math.pi * t / 1440)
        noise = random.uniform(-0.5, 0.5)
        return round(base_temp + noise, 2)

    def generate_humidity(self, t):
        base_hum = 70 + 10 * math.cos(2 * math.pi * t / 1440)
        noise = random.uniform(-1, 1)
        return round(base_hum + noise, 2)

    def generate_co2(self, t):
        base_co2 = 600 + 100 * math.sin(2 * math.pi * t / 2880)
        noise = random.uniform(-20, 20)
        return round(base_co2 + noise, 2)

    def generate_light(self, t):
        # Освітленість лише вдень
        daylight = max(0, 8000 * math.sin(2 * math.pi * t / 1440))
        noise = random.uniform(-300, 300)
        return round(daylight + noise, 2)

    def generate_row(self, t):
        timestamp = self.start_time + datetime.timedelta(minutes=t)
        return {
            "timestamp": timestamp.strftime('%Y-%m-%d %H:%M:%S'),
            "temperature": self.generate_temperature(t),
            "humidity": self.generate_humidity(t),
            "co2": self.generate_co2(t),
            "light": self.generate_light(t)
        }


def generate_synthetic_data(duration_minutes=1440 * 3):
    simulator = SensorSimulator()
    data = []

    for t in range(duration_minutes):
        row = simulator.generate_row(t)
        data.append(row)

    df = pd.DataFrame(data)
    df.to_csv("synthetic_data.csv", index=False)
    print(f"[INFO] Generated {duration_minutes} rows and saved to 'synthetic_data.csv'")


if __name__ == "__main__":
    generate_synthetic_data()
```

synthetic_data_analysis.py

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

df = pd.read_csv('synthetic_data.csv', parse_dates=['timestamp'])

fig, axs = plt.subplots(4, 1, figsize=(14, 10), sharex=True)

axs[0].plot(df['timestamp'], df['temperature'], color='red')
axs[0].set_ylabel('Темп. (°C)')
axs[0].set_title('Зміна температури')

axs[1].plot(df['timestamp'], df['humidity'], color='blue')
axs[1].set_ylabel('Вологість (%)')
axs[1].set_title('Зміна вологості')

axs[2].plot(df['timestamp'], df['co2'], color='green')
axs[2].set_ylabel('CO₂ (ppm)')
axs[2].set_title('Зміна CO₂')

axs[3].plot(df['timestamp'], df['light'], color='orange')
axs[3].set_ylabel('Люкс')
axs[3].set_title('Освітленість')
axs[3].set_xlabel('Час')

locator = mdates.AutoDateLocator(minticks=10, maxticks=20)
formatter = mdates.DateFormatter('%d-%m %H:%M')  # дата-місяць година:хвилина

axs[3].xaxis.set_major_locator(locator)
axs[3].xaxis.set_major_formatter(formatter)

plt.setp(axs[3].xaxis.get_majorticklabels(), rotation=45, ha='right')

plt.tight_layout()
plt.show()
```

microclimate_control.py

```
import pandas as pd
from datetime import datetime
import random

data = pd.read_csv('synthetic_data.csv')

control_actions = []

for index, row in data.iterrows():
    temperature = row['temperature']
    humidity = row['humidity']
    light = row['light']
    co2 = row['co2']
    timestamp = row['timestamp']

    action = {
        'timestamp': timestamp,
        'heater': temperature < 18.0,
        'cooler': temperature > 30.0,
        'humidifier': humidity < 45.0,
        'dehumidifier': humidity > 70.0,
        'lights': light < 300,
        'ventilation': co2 > 1300
    }

    control_actions.append(action)

control_df = pd.DataFrame(control_actions)
combined_df = pd.merge(data, control_df, on='timestamp')
combined_df.to_csv('control_log.csv', index=False)
print("Файл control_log.csv успішно створено.")
```

visualize_logs.py

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('control_log.csv')
```

```
df['timestamp'] = pd.to_datetime(df['timestamp'])

plt.style.use('default')

actions = ['heater', 'cooler', 'humidifier', 'dehumidifier', 'lights', 'ventilation']

fig, axs = plt.subplots(len(actions), 1, figsize=(14, 10), sharex=True)

for i, action in enumerate(actions):
    axs[i].step(df['timestamp'], df[action].astype(int), where='post', label=action.capitalize(), linewidth=1.8)
    axs[i].set_ylabel(action)
    axs[i].legend(loc='upper right')

plt.xlabel('Час')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

modeling.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

np.random.seed(0)
time_index = pd.date_range(start='2025-05-17 00:00', periods=48*6, freq='10T')
temp = 20 + 5*np.sin(np.linspace(0, 6*np.pi, len(time_index))) + np.random.normal(0, 0.5, len(time_index))
humidity = 50 + 20*np.sin(np.linspace(0, 4*np.pi, len(time_index))) + np.random.normal(0, 2, len(time_index))

temp[100:110] -= 7
humidity[200:210] += 30

TEMP_MIN, TEMP_MAX = 18, 25
HUMIDITY_MIN, HUMIDITY_MAX = 40, 60

heater_on = temp < TEMP_MIN
humidifier_on = humidity < HUMIDITY_MIN
ventilator_on = (temp > TEMP_MAX) | (humidity > HUMIDITY_MAX)

df = pd.DataFrame({
    'Температура': temp,
    'Вологість': humidity,
    'Обігрівач': heater_on.astype(int),
    'Зволожувач': humidifier_on.astype(int),
    'Вентилятор': ventilator_on.astype(int)
}, index=time_index)

plt.figure(figsize=(15, 6))
plt.step(df.index, df['Обігрівач'], where='post', label='Обігрівач (вкл/викл)', linewidth=2)
plt.step(df.index, df['Зволожувач'], where='post', label='Зволожувач (вкл/викл)', linewidth=2)
plt.step(df.index, df['Вентилятор'], where='post', label='Вентилятор (вкл/викл)', linewidth=2)
plt.title('Дії виконавчих модулів системи керування мікрокліматом парника')
plt.xlabel('Час (дата та час)')
plt.ylabel('Стан пристрою (0 - викл, 1 - вкл)')
plt.legend(loc='upper right')
plt.grid(True)
plt.tight_layout()
plt.show()
```

test_cases.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime, timedelta
```

```
time_steps = 100
start_time = datetime.now()
timestamps = [start_time + timedelta(minutes=i) for i in range(time_steps)]

# Сценарій 1: Стабільне середовище
scenario_1 = pd.DataFrame({
    'timestamp': timestamps,
    'temperature': np.full(time_steps, 25.0),
    'humidity': np.full(time_steps, 60.0),
    'light': np.full(time_steps, 800.0),
    'co2': np.full(time_steps, 400.0),
    'fan': [0]*time_steps,
    'heater': [0]*time_steps,
    'humidifier': [0]*time_steps,
    'lamp': [0]*time_steps,
    'co2_valve': [0]*time_steps
})
scenario_1['scenario'] = 'Стабільний мікроклімат'

# Сценарій 2: Перегрів
temp_rise = np.linspace(25.0, 38.0, time_steps)
scenario_2 = pd.DataFrame({
    'timestamp': timestamps,
    'temperature': temp_rise,
    'humidity': np.full(time_steps, 60.0),
    'light': np.full(time_steps, 800.0),
    'co2': np.full(time_steps, 400.0),
    'fan': (temp_rise > 30).astype(int),
    'heater': [0]*time_steps,
    'humidifier': [0]*time_steps,
    'lamp': [0]*time_steps,
    'co2_valve': [0]*time_steps
})
scenario_2['scenario'] = 'Перегрів'

# Сценарій 3: Надмірна вологість
humidity_rise = np.linspace(60.0, 90.0, time_steps)
scenario_3 = pd.DataFrame({
    'timestamp': timestamps,
    'temperature': np.full(time_steps, 25.0),
    'humidity': humidity_rise,
    'light': np.full(time_steps, 800.0),
    'co2': np.full(time_steps, 400.0),
    'fan': (humidity_rise > 75).astype(int),
    'heater': [0]*time_steps,
    'humidifier': [0]*time_steps,
    'lamp': [0]*time_steps,
    'co2_valve': [0]*time_steps
})
scenario_3['scenario'] = 'Надмірна вологість'

# Сценарій 4: Комбінований стрес-тест
temp_stress = np.linspace(25.0, 38.0, time_steps)
humidity_stress = np.linspace(60.0, 90.0, time_steps)
co2_stress = np.linspace(400, 200, time_steps)
scenario_4 = pd.DataFrame({
    'timestamp': timestamps,
    'temperature': temp_stress,
    'humidity': humidity_stress,
    'light': np.full(time_steps, 800.0),
    'co2': co2_stress,
    'fan': ((temp_stress > 30) | (humidity_stress > 75)).astype(int),
    'heater': [0]*time_steps,
    'humidifier': [0]*time_steps,
    'lamp': [0]*time_steps,
    'co2_valve': (co2_stress < 300).astype(int)
})
scenario_4['scenario'] = 'Комбінований стрес-тест'

all_scenarios = pd.concat([scenario_1, scenario_2, scenario_3, scenario_4])

sns.set(style="whitegrid")
fig, axes = plt.subplots(4, 1, figsize=(14, 18), sharex=True)
```

```
scenarios = ['Стабільний мікроклімат', 'Перегрів', 'Надмірна вологість', 'Комбінований стрес-тест']
actions = ['fan', 'heater', 'humidifier', 'lamp', 'co2_valve']
colors = ['blue', 'red', 'green', 'orange', 'purple']

for i, scen in enumerate(scenarios):
    df = all_scenarios[all_scenarios['scenario'] == scen]
    for action, color in zip(actions, colors):
        axes[i].plot(df['timestamp'], df[action], label=action, color=color)
    axes[i].set_title(f'Дії системи: {scen}')
    axes[i].legend(loc='upper right')
    axes[i].set_ylabel('Активність')
    axes[i].tick_params(axis='x', rotation=45)

axes[-1].set_xlabel('Час')
plt.tight_layout()
plt.show()
```

comparison.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
import seaborn as sns

time_steps = 100
start_time = datetime.now()
timestamps = [start_time + timedelta(minutes=i) for i in range(time_steps)]

temperature_auto = [25.0]
humidity_auto = [60.0]
co2_auto = [400.0]

temperature_no = [25.0]
humidity_no = [60.0]
co2_no = [400.0]

fan_auto = []
co2_valve_auto = []

for i in range(1, time_steps):
    delta_temp = 0.13
    delta_hum = 0.3
    delta_co2 = -2

    temperature_no.append(temperature_no[-1] + delta_temp)
    humidity_no.append(humidity_no[-1] + delta_hum)
    co2_no.append(co2_no[-1] + delta_co2)

    temp = temperature_auto[-1] + delta_temp
    hum = humidity_auto[-1] + delta_hum
    co2 = co2_auto[-1] + delta_co2

    if temp > 30 or hum > 75:
        temp -= 0.25
        hum -= 0.4
        fan_auto.append(1)
    else:
        fan_auto.append(0)

    if co2 < 300:
        co2 += 4
        co2_valve_auto.append(1)
    else:
        co2_valve_auto.append(0)

    temperature_auto.append(temp)
    humidity_auto.append(hum)
    co2_auto.append(co2)


auto_df = pd.DataFrame({
    'timestamp': timestamps,
    'temperature': temperature_auto,
    'humidity': humidity_auto,
```

```
    'co2': co2_auto,
    'fan': fan_auto + [fan_auto[-1]],
    'co2_valve': co2_valve_auto + [co2_valve_auto[-1]],
    'mode': 'Автоматичне керування'
})

no_df = pd.DataFrame({
    'timestamp': timestamps,
    'temperature': temperature_no,
    'humidity': humidity_no,
    'co2': co2_no,
    'fan': [0]*time_steps,
    'co2_valve': [0]*time_steps,
    'mode': 'Без керування'
})

comparison_df = pd.concat([auto_df, no_df])

sns.set(style="whitegrid")
fig, axes = plt.subplots(3, 1, figsize=(14, 16), sharex=True)

sns.lineplot(data=comparison_df, x='timestamp',
y='temperature', hue='mode', ax=axes[0])
axes[0].set_title('Температура: автоматичне керування
vs. без керування')

sns.lineplot(data=comparison_df, x='timestamp',
y='humidity', hue='mode', ax=axes[1])
axes[1].set_title('Вологість: автоматичне керування vs.
без керування')

sns.lineplot(data=comparison_df, x='timestamp', y='co2',
hue='mode', ax=axes[2])
axes[2].set_title('Рівень CO₂: автоматичне керування vs.
без керування')

for ax in axes:
    ax.set_ylabel('Значення')
    ax.tick_params(axis='x', rotation=45)

axes[-1].set_xlabel('Час')
plt.tight_layout()
plt.show()
```

Додаток Г
(Обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА
«Розробка функціональної схеми координатора децентралізованої системи керування»

Перелік ілюстрованих матеріалів:

1. Структурна діаграма взаємозв'язків між компонентами системи керування мікрокліматом парника
2. Блок-схема алгоритму логіки управління мікрокліматом у парнику
3. Приклад згенерованих сиснтетичних даних
4. Графічне відображення змін у часі параметрів парника
5. Дані, що зберігаються в control_log.csv файлі
6. Графіки відображення виконавчих дій системи
7. Дії виконавчих модулів системи керування мікрокліматом парника під час моделювання
8. Активність системи керування у відповідь на чотири різні сценарії
9. Графік поведінки системи у відповідь на змодельовану зміну мікроклімату при автоматичному керуванні мікрокліматок парника та без нього

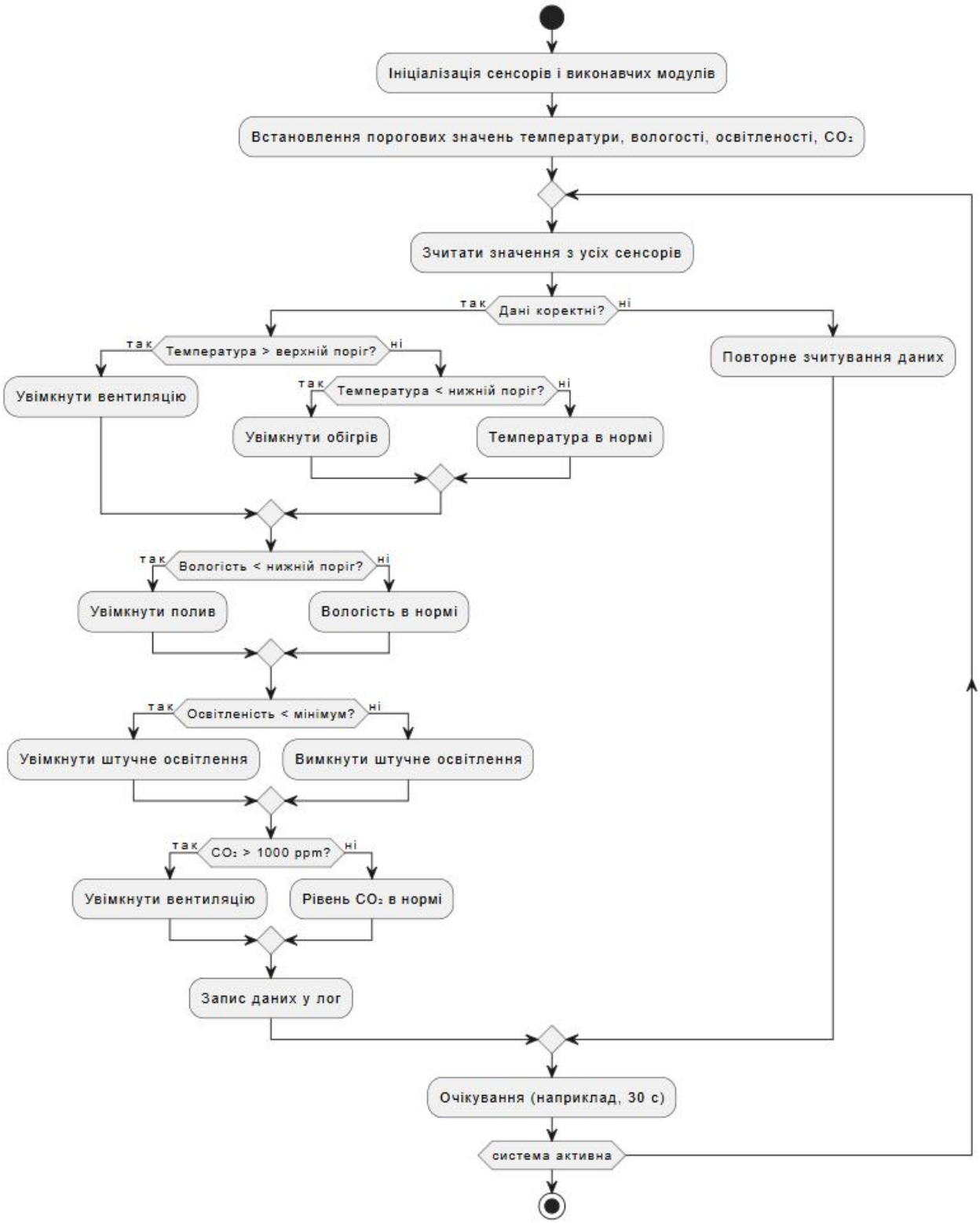
Виконав: студент 4-го курсу, групи 2АКІТ-21б
спеціальності 151 – Автоматизація та
комп'ютерно-інтегровані технології
(шифр і назва спеціальності)
_____Михайло КАРАКОЙ
(ім'я та прізвище)

Керівник: д.т.н., професор каф. КСУ
_____Марія ЮХИМЧУК
(ім'я та прізвище)
«16» ____6____ 2025 р.

Рецензент: к.т.н., доцент каф. АІІТ
_____Володимир ГАРМАШ
(ім'я та прізвище)
«16» ____6____ 2025 р.

Структурна діаграмавзаємозв’язків між компонентами системи керування мікрокліматом парника

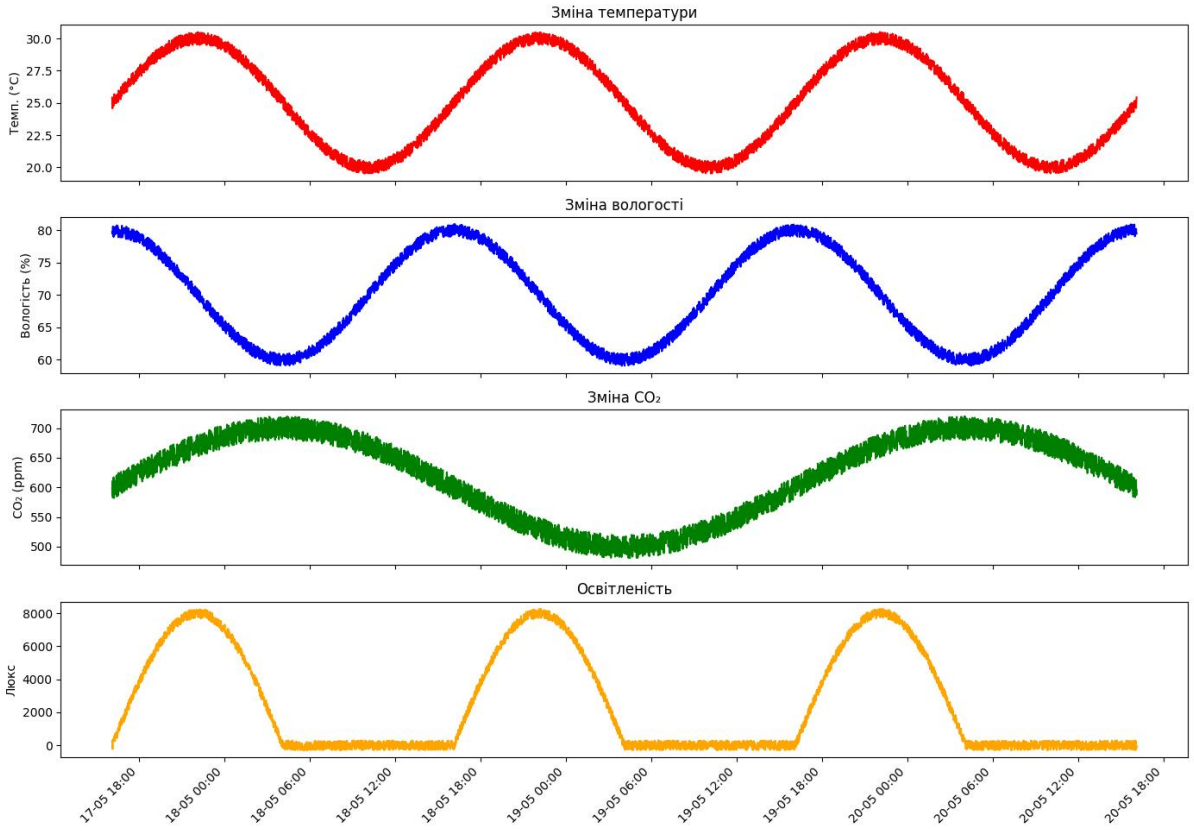
Блок-схема алгоритму логіки управління мікрокліматом у парнику



Приклад згенерованих сиснтетичних даних

	timestamp	temperature	humidity	co2	light
1	2025-05-17 16:07:59	25.42	79.74	598.62	249.34
2	2025-05-17 16:08:59	24.57	79.48	586.9	-39.13
3	2025-05-17 16:09:59	24.62	79.83	610.85	-224.02
4	2025-05-17 16:10:59	25.27	80.52	582.53	317.92
5	2025-05-17 16:11:59	25.11	80.62	587.62	354.94
6	2025-05-17 16:12:59	25.09	79.06	598.09	273.54
7	2025-05-17 16:13:59	25.05	80.4	616.93	401.79
8	2025-05-17 16:14:59	24.79	79.02	589.95	321.09
9	2025-05-17 16:15:59	25.21	79.55	582.08	282.94
10	2025-05-17 16:16:59	25.68	79.55	592.36	566.74

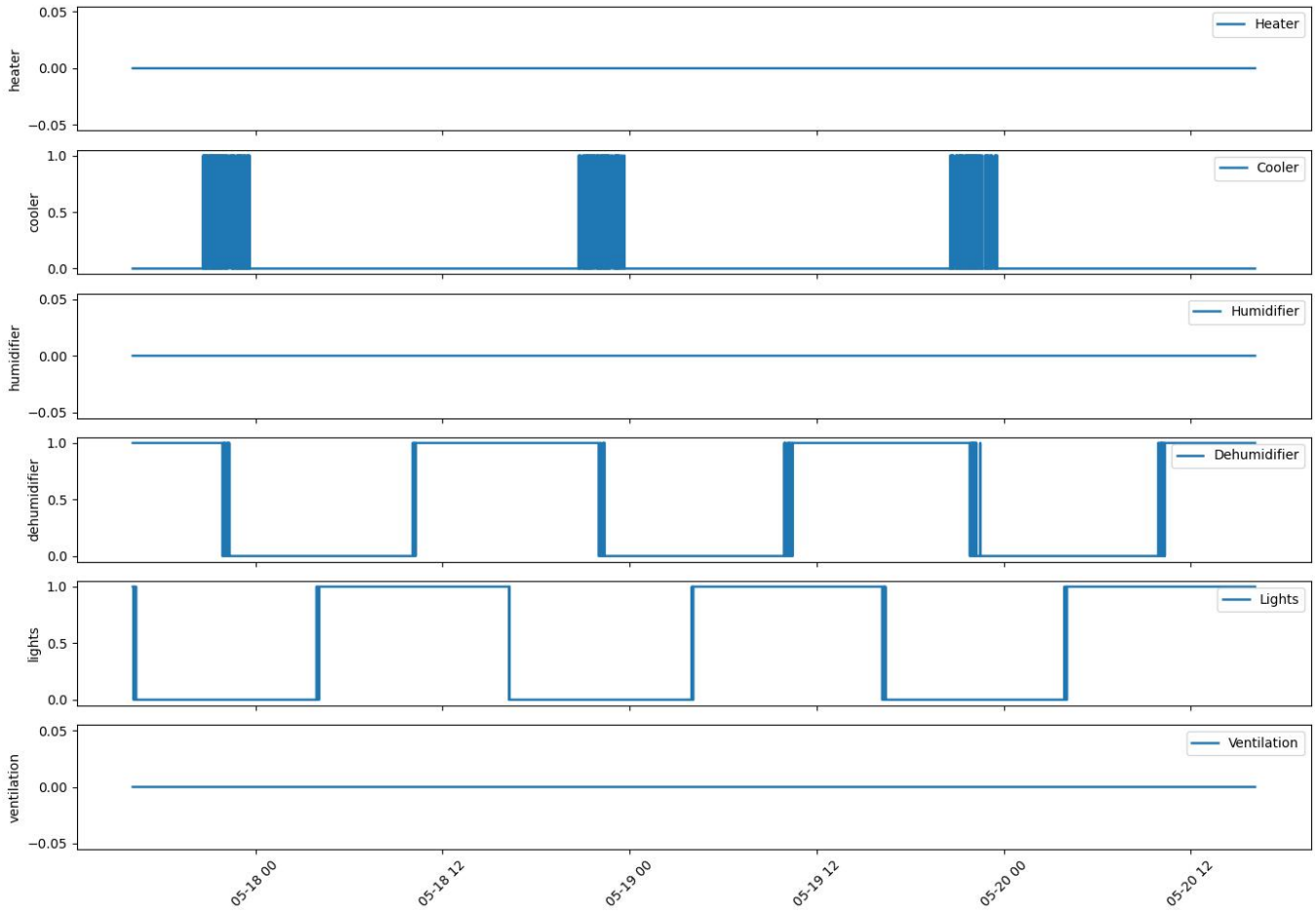
Графічне відображення змін у часі параметрів парника



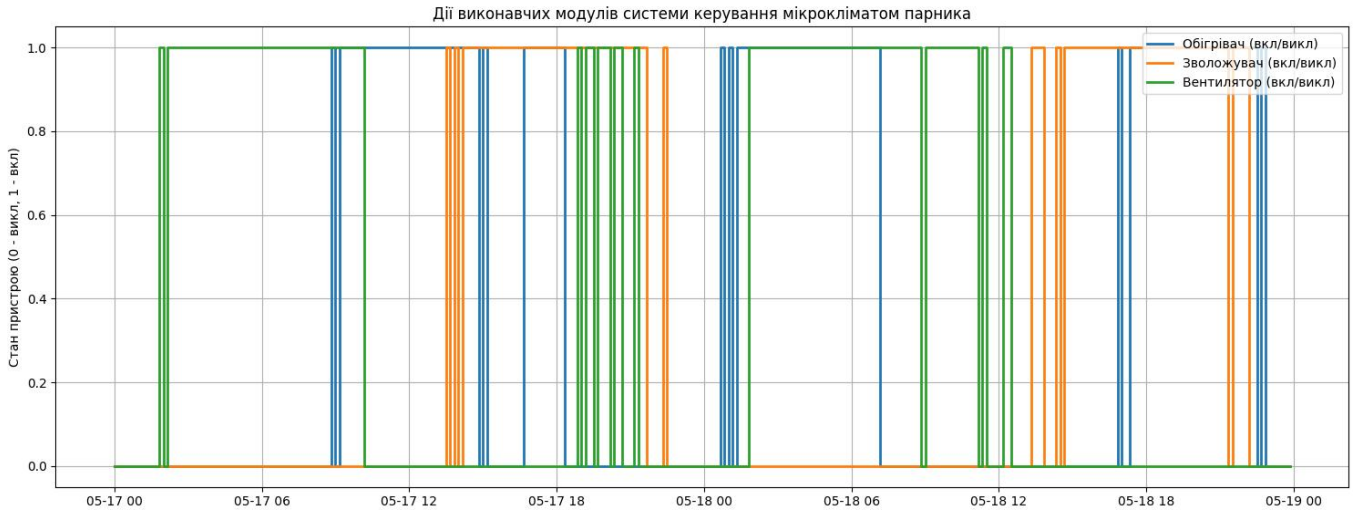
Дані, що зберігаються в control_log.csv файлі

	timestamp	temperature	humidity	co2	light	heater	cooler	humidifier	dehumidifier	lights	ventilation
1	2025-05-17 16:07:59	25.42	79.74	598.62	249.34	False	False	False	True	True	False
2	2025-05-17 16:08:59	24.57	79.48	586.9	-39.13	False	False	False	True	True	False
3	2025-05-17 16:09:59	24.62	79.83	610.85	-224.02	False	False	False	True	True	False
4	2025-05-17 16:10:59	25.27	80.52	582.53	317.92	False	False	False	True	False	False
5	2025-05-17 16:11:59	25.11	80.62	587.62	354.94	False	False	False	True	False	False
6	2025-05-17 16:12:59	25.09	79.06	598.09	273.54	False	False	False	True	True	False
7	2025-05-17 16:13:59	25.05	80.4	616.93	401.79	False	False	False	True	False	False
8	2025-05-17 16:14:59	24.79	79.02	589.95	321.09	False	False	False	True	False	False
9	2025-05-17 16:15:59	25.21	79.55	582.08	282.94	False	False	False	True	True	False
10	2025-05-17 16:16:59	25.68	79.55	592.36	566.74	False	False	False	True	False	False

Графіки відображення виконавчих дій системи



Дії виконавчих модулів системи керування мікрокліматом парника під час моделювання



Активність системи керування у відповідь на чотири різні сценарії

Графік поведінки системи у відповідь на змодельовану зміну мікроклімату при автоматичному керуванні мікрокліматок парника та без нього