НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
імені адмірала Макарова

Навчально-науковий інститут автоматики та електротехніки
(повне найменування інституту, факультету)

Кафедра комп'ютеризованих систем управління
(повна назва кафедри)

«Допущений до захисту»
Завідувач кафедри
_____
«___» _____ 2024 р.

Пояснювальна записка

до кваліфікаційної роботи
на здобуття другого (магістерського) рівня вищої освіти

за спеціальністю 174– «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
ОПП - «Комп'ютеризовані системи управління та автоматика»

на тему: **Вбудована система комплексної автоматизації технологічних процесів вирощування рослин у теплиці**

Виконав студент **6** курсу, **6341м** групи
Сироватка А. В.
(прізвище та ініціали)
Керівник **Герасін О. С.**
(прізвище та ініціали)

Миколаїв – 2024

Національний університет кораблебудування імені адмірала Макарова

ННІ автоматики та електротехніки

Кафедра комп'ютеризованих систем управління

Рівень вищої освіти **другий (магістерський)**

Спеціальність **174– «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»**

ОПП **«Комп'ютеризовані системи управління і автоматика»**

ЗАТВЕРДЖУЮ
Завідувач кафедри КСУ
_____Черно О.О.
«____»_____2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

С и р о в а т к а А р т е м В і т а л і й о в и ч
(прізвище, ім'я, по батькові)

1. Тема роботи: *вбудована система комплексної автоматизації технологічних процесів вирощування рослин у теплиці*

керівник роботи Герасін Олександр Сергійович, к.т.н., доцент

затверджені наказом вищого навчального закладу від "20" грудня 2024 р. № 1075-уч

2. Строк подання роботи *19.12.2024 р.*

3. Вихідні дані до роботи *система має містити контролер Arduino Nano 33 IoT та мобільний керуючий додаток, датчики температури та вологості повітря і ґрунту, вентилятор, нагрівач, лампи освітлення, датчик концентрації CO_2, датчик потоку води, годинник реального часу та аварійну сирену.*

4. Мета дослідження *полягає в розробці вбудованої системи комплексної автоматизації технологічних процесів вирощування рослин у теплиці, яка забезпечить оптимізацію умов для росту рослин, підвищить ефективність управління теплицею та зменшить вплив людського фактору на процеси контролю та моніторингу.*

5. Зміст розрахунково-пояснювальної записки (основні задачі дослідження)
Аналіз систем автоматизації технологічних процесів вирощування рослин у теплиці
Технічне забезпечення для реалізації системи комплексної автоматизації технологічних процесів вирощування рослин у теплиці
Програмне забезпечення для впровадження системи комплексної автоматизації технологічних процесів вирощування рослин у теплиці

Впровадження засобів штучного інтелекту в систему комплексної автоматизації технологічних процесів вирощування рослин у теплиці

Охорона праці та навколишнього середовища

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

Актуальність та постановка задач розробки

Функціональна та принципова схеми системи

Вибір мов та апаратної платформи

Розробка алгоритмів керування та керуючої програми

Розробка мобільного додатку

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Охорона праці та навколишнього середовища	*Герасін О.С., доцент каф. КСУ*	29.11.2024	09.12.2024

Дата видачі завдання: «02» вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	*Аналіз систем автоматизації технологічних процесів вирощування рослин у теплиці*	*14.10.2024*	*Виконано*
2	*Технічне забезпечення для реалізації системи комплексної автоматизації технологічних процесів вирощування рослин у теплиці*	*04.11.2024*	*Виконано*
3	*Програмне забезпечення для впровадження системи комплексної автоматизації технологічних процесів вирощування рослин у теплиці*	*26.11.2024*	*Виконано*
4	*Впровадження засобів штучного інтелекту в систему комплексної автоматизації технологічних процесів вирощування рослин у теплиці*	*02.12.2024*	*Виконано*
5	*Охорона праці та навколишнього середовища*	*09.12.2024*	*Виконано*
6	*Оформлення пояснювальної записки*	*16.12.2024*	*Виконано*

Студент _____ Сироватка А.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Герасін О.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці у галузі систем керування для автоматизації технологічних процесів у аграрному секторі. Метою роботи є розробка вбудованої системи автоматизації технологічних процесів вирощування рослин у теплиці.

В ході роботи проведено аналіз сучасних систем керування теплицями, а також засобів та методів розробки вбудованої системи комплексної автоматизації технологічних процесів вирощування рослин у теплиці. Розроблено принципову схему системи, вибрано апаратну платформу, датчики та виконавчі механізми для реалізації вбудованої системи автоматизації технологічних процесів вирощування рослин у теплиці.

В результаті розроблено систему автоматизованого керування теплицею за допомогою контролеру Arduino Nano 33 IoT та мобільного застосунка.

Обсяг кваліфікаційної роботи становить 133 сторінки, робота містить 73 рисунків, 4 таблиці та 36 використаних джерел.

ABSTRACT

The qualification work is devoted to the development of control systems for the automation of technological processes in the agricultural sector. The purpose of the work is to develop an embedded system for automating technological processes for growing plants in a greenhouse.

During the work, an analysis of modern greenhouse control systems was carried out, as well as means and methods for developing an embedded system for automating technological processes for growing plants in a greenhouse. A schematic diagram of the system was developed, a hardware platform, sensors and actuators were selected for the implementation of an embedded system for complex automating technological processes for growing plants in a greenhouse.

As a result, an automated greenhouse control system was developed using the Arduino Nano 33 IoT controller and a mobile application.

The volume of the qualification work is 133 pages, the work contains 73 figures, 4 tables and 36 references.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

АСКТ– автоматизована система керування теплицею;

БД – база даних;

ЗВП – звуковідтворюючі пристрої;

ІБД – інтегрована база даних;

ІІТ – інтелектуальні інформаційні технології;

НМ – нейронна мережа;

МК – мікроконтролер;

ПАК – програмно-апаратний комплекс;

ПЛК – програмовані логічні контролери;

СДК – система дистанційного керування;

СУБД – система управління базою даних.

ВСТУП

В наш час автоматизовані системи керування (АСК) відіграють ключову роль у процесах автоматизації. Вони дозволяють здійснювати контроль над різними пристроями завдяки застосуванню технологій штучного інтелекту. Основними елементами таких систем є мікроконтролери — компактні й спеціалізовані обчислювальні пристрої, що використовуються для управління та моніторингу різних процесів.

Мікроконтролер (Microcontroller) — це невеликий електронний пристрій, який включає мікропроцесор, пам’ять та периферійні компоненти, призначений для управління конкретними функціями або пристроями. Його часто використовують у вбудованих системах для контролю й управління певними процесами. Мікроконтролери мають широке застосування: від промислової автоматизації й автомобільної галузі до побутової електроніки, медичних приладів і робототехніки.

Сучасна автоматизація технологічних процесів є надзвичайно важливою, хоча створення оптимальної моделі часто викликає труднощі через складну динаміку, неоднорідність і непередбачуваність реальних процесів. Більшість проектів автоматизації розробляється поетапно, з поступовим розширенням функціональних можливостей і використанням статичних методів контролю. У всьому світі активно розвиваються методи комплексної автоматизації, засновані на теорії оптимізації складних систем. Сучасні комп'ютерні обчислювальні технології допомагають вирішувати такі багатомірні задачі, хоч вони і потребують значних обчислювальних ресурсів.

У сільському господарстві існують особливі проблеми, пов'язані з регулюванням теплових процесів у теплицях. Недостатній контроль за теплообміном у тепличних спорудах часто обумовлений затримкою у транспорті регулюючих сигналів, множинними збуреннями, нерівномірним прогріванням повітря та ґрунту через систему труб тощо. Це обумовлено

відсутністю точних математичних моделей та систем автоматизації, які б охоплювали всі етапи розробки й експлуатації таких систем.

РОЗДІЛ 1

АНАЛІЗ СИСТЕМ АВТОМАТИЗАЦІЇ ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ ВИРОЩУВАННЯ РОСЛИН У ТЕПЛИЦІ

1.1. Ознайомлення з системами автоматизованого керування

Автоматизована система керування (АСК) призначена для забезпечення автоматичного контролю, моніторингу, оптимізації та керування різними процесами і системами. Вона реалізується використанням різних систем управління для робочого обладнання, такого як машини, котли і печі для термообробки, літальні апарати та інші об’єкти та транспортні засоби з мінімальним або зменшеним втручанням людини [19].

Варто зазначити, що АСК охоплює найрізноманітніші програми: від побутового термостата, керуючого котлом, до великої промислової системи управління з десятками тисяч вхідних вимірів і вихідних керуючих сигналів. За складністю управління вона може варіюватися від простого двохпозиційного управління до високорівневих алгоритмів з декількома змінними.

У найпростішому типі автоматичного управління контролер порівнює виміряне значення процесу з бажаним заданим значенням і обробляє отриманий сигнал помилки, щоб змінити деякі вхідні дані для процесу таким чином, щоб процес залишався усталеним. Це управління зі зворотним зв’язком являє собою додаток зворотного зв’язку до системи. Математичні основи теорії керування започатковано у 18 столітті, вона швидко розвивалася в 20-му [14].

Відомо, що автоматизація досягається за допомогою різних засобів, включаючи механічні, гідравлічні, пневматичні, електричні, електронні пристрої та комп’ютери, зазвичай в комбінації. У складних системах, таких як сучасні заводи, літаки і кораблі, зазвичай використовуються всі ці

комбіновані методи. Перевага автоматизації включає економію робочої сили, економію витрат на електроенергію, економію матеріальних витрат і підвищення якості і точності.

По суті, існує два типи контуру управління, а саме [26]:

1) управління без зворотного зв'язку;

2) управління зі зворотним зв'язком.

При управлінні без зворотного зв'язку керуючий вплив контролера не залежить від «виходу процесу» (або «регульованої змінної процесу»). Хорошим прикладом цього є котел центрального опалення, керований тільки таймером, так що тепло подається протягом постійного часу, незалежно від температури в приміщенні. Наприклад, керуючим впливом є виключення і включення котла, вихідний сигнал процесу-це температура будівлі.

При управлінні зі зворотним зв'язком вплив на контролер залежить від вихідного сигналу процесу. У разі аналогії з котлом, це буде включати датчик температури для контролю температури в приміщенні і тим самим подавати сигнал назад на контролер, щоб гарантувати, що він підтримує в приміщенні температуру, встановлену на термостаті.

Отже, контролер зі зворотним зв'язком має контур зворотного зв'язку, який гарантує, що контролер забезпечує вихідний сигнал процесу, рівний «еталонному входу» або «уставці». З цієї причини контролери отримали назву контролерів зі зворотним зв'язком.

Управління зі зворотним зв'язком – це принцип управління, за яким система прагне підтримувати задане відношення однієї системної змінної до іншої, порівнюючи функції цих змінних і використовуючи різницю в якості засобу контролю. Вдосконалений тип автоматизації, який зробив революцію у виробництві, авіабудуванні, зв'язку та інших галузях, - це управління зі зворотним зв'язком, яке зазвичай є безперервним і включає в себе виконання вимірювань за допомогою датчика і виконання розрахункових коригувань для підтримки вимірюваної змінної в заданому діапазоні. Теоретичною основою автоматизації зі зворотним зв'язком є теорія регулювання [3].

Крім того, послідовне управління може бути або фіксованим, або логічним, яке буде виконувати різні дії в залежності від різних станів системи. Прикладом регульованого управління в фіксованій послідовності є таймер орошення газону. Такий таймер дозволяє програмувати час і тривалість поливу газону, що дозволяє ефективно використовувати воду і запобігати її втратам. Стан зрошення газону відносяться до різних умов, які можуть виникнути в сценарії використання або послідовності дій системи. Прикладом є ліфт, який використовує логіку, засновану на станах системи, для виконання певних дій у відповідь на його стан і введення оператора.

Наприклад, якщо оператор натискає кнопку поверху n, система буде реагувати в залежності від того, зупинився ліфт або рухається, піднімається або опускається, відкриті або закриті двері, а також від інших умов (рис.1.1).

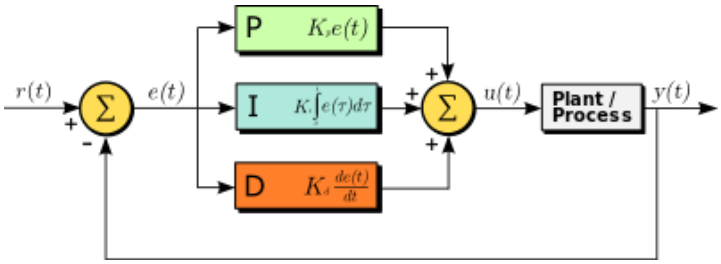

Рисунок 1.1 - Блок-схема контролера ПІД-регулятора в контурі зворотного зв’язку, де r(t) - значення процесу або «задане значення» і y(t) - виміряне значення процесу [18]

Комп’ютери можуть виконувати як послідовне управління, так і управління зі зворотним зв’язком, і, як правило, один комп’ютер виконує і те, і інше в промисловому додатку. Програмовані логічні контролери - це типи мікропроцесора спеціального призначення, які замінюють багато апаратних компонентів, такі як таймери і барабанні секвенсори, використовувані в системах релейної логіки [3].

Комп’ютери загального призначення для управління технологічними процесами все частіше замінюють автономні контролери за допомогою одного комп’ютера, здатного виконувати операції сотень контролерів.

Водночас комп’ютери управління технологічним процесом можуть обробляти дані з мережі логічних контролерів, приладів і контролерів, щоб реалізувати типове (наприклад, PID) управління багатьма окремими змінними або, в деяких випадках, реалізувати складні алгоритми управлення з використанням декількох входів і математичних маніпуляцій. Вони також можуть аналізувати дані і створювати графічні дисплеї в реальному часі для операторів, а також створювати звіти для операторів, інженерів і керівників.

Потрібно акцентувати, що принцип релейної логіки був введений з електрифікацією заводів, яка зазнала швидку адаптацію з 1900 по 1920-і роки. Центральні електростанції також переживали швидке зростання, і експлуатація нових котлів високого тиску, парових турбін і електричних підстанцій викликала великий попит на прилади та засоби управління. Центральні диспетчерські стали поширеними у 1920-х роках і використовувались для керування технологічними процесами. Проте, до початку 1930-х років, більшість систем управління технологічними процесами складалися з простих механізмів, які були здатні лише включатись або виключатись.

Оператори зазвичай відстежували графіки, складені самописцями, які відображали дані з інструментів. Щоб внести виправлення, оператори вручну відкривали або закривали клапани або включали або виключали перемикачі. У диспетчерських також використовувалися кольорові вогні, щоб посилати сигнали робітником на заводі, щоб вручну внести певні зміни. Контролери, які могли проводити розрахункові зміни у відповідь на відхилення від заданого значення, а не на двопозиційне управління, почали впроваджуватися в 1930-х роках. Контролери дозволили виробництву продовжити зростання продуктивності [1].

Починаючи з 1958 року, різні системи, засновані на твердотільних модулях цифрової логіки для жорстко запрограмованих логічних контролерів (попередників програмованих логічних контролерів), з'явилися на зміну електромеханічній релейній логіці в промислових системах управління. Для управління процесами і автоматизації, включаючи ранні версії Telefunken / AEGLogistat, SiemensSimatic, Philips / Mullard / Valvo [de] Norbit, BBCSigmatronic, ACEC Системи Logacec, Akkord [de] Estacord, KroneMibakron, Bistat, Datapac, Norlog, SSR або Procontic.

Існують різні засоби автоматизації. Серед них [3]:

- ШНМ - штучна нейронна мережа;
- DCS - розподілена система управління;
- HMI - людино-машинний інтерфейс;
- RPA - роботизована автоматизація процесів;
- SCADA - диспетчерський контроль і збір даних;
- PLC - Програмований логічний контролер;
- прилади;
- система управління рухом;
- робототехніка.

Промислова автоматизація в першу чергу пов'язана з автоматизацією виробництва, контролю якості та процесів обробки матеріалів. Контролери загального призначення для промислових процесів включають програмовані логічні контролери, автономні модулі введення-виведення і комп'ютери. Промислова автоматизація повинна замінити прийняття рішень людьми і ручну діяльність з використанням механізованого устаткування і команд логічного програмування.

Однією з тенденцій є більш широке використання машинного зору для забезпечення функцій автоматичного контролю і управління роботами, а інший - триваюче збільшення використання роботів. Промислова автоматизація просто необхідна в промисловості [7].

З іншого боку, енергоефективність в виробничих процесах стала більш пріоритетною. Компанії-виробники напівпровідників, такі як Infineon Technologies, пропонують програми для 8-бітних мікроконтролерів, наприклад, використовувані в системах управління двигунами, насосах загального призначення, вентиляторах і електровелосипедах, щоб знизити споживання енергії та, таким чином, підвищити ефективність.

Слід додати, що промислова автоматизація включає в себе впровадження програмованих логічних контролерів у виробничий процес.

Програмовані логічні контролери (ПЛК) використовують систему обробки, яка дозволяє варіювати управління входами і виходами за допомогою простого програмування. В ПЛК використовується програмована пам’ять, в якій зберігаються інструкції та функції, такі як логіка, послідовність, синхронізація, підрахунок. Застосування логічної мови програмування в ПЛК дозволяє забезпечити гнучкість, швидкість і надійність в реалізації логічного керування в автоматизованих системах. Залежно від логіки програми, ПЛК може обробляти вхідні дані, порівнювати їх з певними умовами та генерувати відповідні логічні вихідні сигнали.

Програмовані логічні контролери мають деяку схожості з комп'ютерами, хоча вони призначені для специфічних завдань автоматизації та керування процесами. Вони побудовані таким чином, що необхідні тільки базові знання програмування на основі логіки для їхнього ефективного використання, можуть працювати надійно і неперервно в різних важких умовах, включаючи високі температури, вологість, вібрації та електромагнітні перешкоди.

Найбільшою перевагою ПЛК є їх гнучкість. З тими ж базовими контролерами, ПЛК може управляти рядом різних систем управління. ПЛК позбавляють від необхідності перепрограмувати систему для зміни системи управління. Ця гнучкість призводить до створення економічних систем для складних і різноманітних об’єктів управління. ПЛК можуть варіюватися від невеликих пристроїв «будівельної цегли» з десятками вводів/виводів в

корпусі, інтегрованому з процесором, до великих монтованих в стійку модульних пристроїв з кількістю тисяч вводів/виводів, які часто підключаються до інших ПЛК та системи SCADA (рис.1.2).

Рисунок 1.2 - Система Siemens Simatic S7- 400 в стійці, зліва направо: блок живлення, інтерфейсний модуль та комунікаційний процесор [32]

Потрібно підкреслити, що найбільшою перевагою застосування АСК в промисловості є те, що вона пов’язана з більш швидким виробництвом і більш дешевими витратами на робочу силу.

Ще однією перевагою може бути те, що АСК замінюють важку фізичну або монотонну роботу.

Крім того, завдання, які виконуються в небезпечних середовищах або які іншим чином виходять за рамки людських можливостей, можуть виконуватися машинами, оскільки вони здатні працювати навіть при екстремальних температурах або в радіоактивній чи токсичній атмосфері. Машини також можна підтримувати за допомогою простих перевірок якості [3].

Однак в даний час не всі завдання можна автоматизувати, а деякі завдання дорожче автоматизувати, ніж інші. Початкові витрати на установку обладнання в заводських умовах високі, а відмова від обслуговування системи може призвести до втрати самого продукту.

Існують дослідження, які можуть свідчити про можливі шкідливі наслідки промислової автоматизації, включаючи втрату робочих місць та негативний вплив на навколишнє середовище. Проте, важливо врахувати, що

ці результати є складними і суперечливими, і можуть бути подолані шляхом відповідального планування та управління.

Перерахуємо основні переваги АСК на підприємстві [10]:

- підвищена пропускна здатність або продуктивність;
- підвищена якість або підвищена передбачуваність якості;
- підвищена надійність (узгодженість) процесів або продукту;
- підвищена узгодженість виведення;
- зниження прямих витрат і витрат на людську працю;
- установка в експлуатації скорочує час циклу;
- може виконувати завдання, що вимагають високої точності;
- замінює людей-операторів в задачах, пов’язаних з важкою фізичною або монотонною роботою (наприклад, використання одного вилочного навантажувача з одним водієм замість групи з декількох робочих для підйому важкого предмета) ;
- знижує деякі виробничі травми (наприклад, меншу напругу спини при піднятті важких предметів) ;
- замінює людей в задачах, що виконуються в небезпечному середовищі (наприклад, вогонь, космос, вулкани, ядерні об’єкти) ;
- виконує завдання, що виходять за рамки людських можливостей: розмір, вага, швидкість, витривалість. Значно скорочує час роботи і час виконання роботи;
- звільняє робітників для виконання інших функцій;
- забезпечує високорівневі завдання по розробці, розгортання, обслуговування та запуску автоматизованих процесів.

Водночас АСК мають недоліки, а саме:

- можливі загрози та уразливості безпеки можуть виникати внаслідок підвищеної вразливості до помилок, що можуть бути скоєні через збільшену автоматизацію;
- непередбачувані або надмірні витрати на розробку;
- висока початкова вартість;

– витісняє робітників у зв’язку з заміною роботи.

Крім того, існують певні обмеження АСК:

1. Сучасні технології не здатні автоматизувати всі бажані завдання. Багато операцій з використанням автоматизації вимагають великих вкладень капіталу і виробляють великі обсяги продукції, що робить несправності надзвичайно дорогими і потенційно небезпечними. Отже, необхідний деякий персонал для забезпечення правильного функціонування всієї системи і підтримки безпеки і якості продукції. Протягом того, як процес стає все більш автоматизованим, стає все менше і менше трудовитрат або поліпшення якості. Це приклад спадної прибутковості і логістичної функції.

2. Оскільки все більше і більше процесів стають автоматизованими, залишається все менше неавтоматизованих процесів. Це приклад вичерпання можливостей. Однак нові технологічні парадигми можуть встановлювати нові межі, які перевершують колишні.

3. Розпізнавання образів на людському рівні, розуміння мови і здатність відтворювати мову набагато перевершують можливості сучасних механічних і комп’ютерних систем.

4. Завдання, що вимагають суб’єктивної оцінки або синтезу складних сенсорних даних, таких як запахи і звуки, а також завдання високого рівня, такі як стратегічне планування, в даний час вимагають людського досвіду.

1.2 Аналіз сучасних систем автоматизації технологічних процесів вирощування рослин у теплиці

Розглянемо сучасні системи керування теплицями. Вони мають таку класифікацію:

– автономна теплиця, працює за рахунок сонячної чи теплової енергії. Повільно реагує на перепади температури, при різкому перепаді температури рослини можуть постраждати;

– енергозалежна модель, підключаються до електромережі, підтримка постійних параметрів більш оптимальна, можна створити для рослин ідеальні умови.

Комплекс керування промисловою теплицею показаний на рис. 1.3 [34].

Рисунок 1.3 – Теплиця Екотек

Теплиця має додаткове освітлення, вбудоване в свою систему, яке дозволяє продовжувати процес фотосинтезу і розвитку рослин, навіть при обмеженому освітленні під час короткого світлового дня. Окрім того, система оснащена автоматичним поливом, що дозволяє забезпечувати рослини необхідною кількістю вологи, навіть при відсутності нагляду над теплицею. Теплиця з такими функціональними можливостями може бути цінною для комерційних виробників рослин або осіб, які зацікавлені в самостійному вирощуванні овочів та інших рослин для власного споживання. Завдяки її розміру, захисній плівці, додатковому освітленню та автоматичному поливу, така теплиця може стати ефективним інструментом для досягнення високих врожаїв і забезпечення надійного росту рослин навіть у непридатних умовах.

Каркас теплиці виготовлений з оцинкованої сталі, що забезпечує довговічність, а покриття з полікарбонату товщиною 4–16 мм гарантує теплоізоляцію та рівномірний розподіл сонячного світла. Теплиця має модульну конструкцію, що дозволяє змінювати розміри в залежності від потреб — від 3 м ширини до 30 м довжини.

Основною особливістю є автоматизовані системи, включаючи клімат-контроль, полив, вентиляцію та освітлення. Клімат-контроль підтримує оптимальні температурні та вологості умови, а система поливу автоматично регулює кількість води, враховуючи вологість ґрунту. Вентиляційні люки автоматично відкриваються для контролю вмісту CO_2 і кисню. Освітлення активується для підтримки рослин в зимовий період.

Теплиця оснащена Wi-Fi модулем, що дозволяє користувачам віддалено контролювати та налаштовувати всі параметри роботи системи через мобільний додаток або веб-інтерфейс. Завдяки цьому модулю, користувач може моніторити температуру, вологість, рівень освітленості та інші важливі показники безпосередньо з телефону або комп'ютера, незалежно від того, де він знаходиться.

Переваги теплиці: підвищена енергоефективність, зниження впливу людського чинника, гнучкість у використанні, підтримка круглорічного вирощування.

Недоліки теплиці: висока вартість початкової інвестиції, залежність від енергії, необхідність технічного обслуговування, обмеженість простору.

Теплиця 2agrocloud зображена на рис. 1.4.[33]

Рисунок 1.4 – Теплиця 2agrocloud

Сучасні системи управління теплицями, як 2agrocloud, являють собою комплекс програмного та апаратного забезпечення, що забезпечує

автоматизацію вирощування рослин. Основні компоненти таких систем включають:

Блок управління містить наступні компоненти:

- центральний модуль системи, що координує роботу всіх підключених пристроїв;
- забезпечує доступ до інтернету та можливість дистанційного керування через додаток на смартфоні чи комп'ютері.

Датчики:

- температури та вологості повітря: контролюють мікроклімат всередині теплиці;
- вологості та температури ґрунту: визначають оптимальні умови для росту рослин;
- освітленості, відстежують рівень природного світла для ввімкнення додаткового освітлення.

Виконавчі механізми:

- електроприводи для автоматичного відкриття вікон, дверей чи фрамуг;
- системи поливу (крапельного) із дозуванням води;
- кліматичне обладнання: вентилятори, обігрівачі, зволожувачі.

Інтерфейс управління:

- зручний мобільний або веб-додаток із функціями моніторингу, ручного та автоматичного керування.

Для досягнення максимального врожаю та економії ресурсів сучасні теплиці використовують адаптивні алгоритми управління. Алгоритм підтримання мікроклімату передбачає збір даних із датчиків температури, вологості та освітленості, порівняння поточних параметрів із заданими оптимальними значеннями та активацію відповідного обладнання, такого як відкриття вікон, ввімкнення вентиляції, обігріву чи поливу. Алгоритм автоматичного поливу базується на визначенні рівня вологості ґрунту за допомогою датчиків, розрахунку необхідного об’єму води для поливу та

дозованому поданні води через крапельну систему. Алгоритм керування освітленням включає моніторинг природного освітлення, автоматичне ввімкнення фітоламп у разі недостатньої кількості світла та контроль тривалості світлового дня відповідно до потреб рослин. Алгоритм прогнозування використовує історичні дані та погодні прогнози для оптимізації роботи системи, наприклад, зниження інтенсивності поливу перед дощем або завчасне ввімкнення обігріву перед похолоданням.

Переваги теплиці: економія ресурсів, підвищення врожайності завдяки оптимальному догляду, зменшення людського втручання та мінімізація ризику помилок, можливість віддаленого моніторингу та управління.

Недоліки теплиці: ризик виходу з ладу обладнання, залежність від інтернету, складність інтеграції з наявними системами, витрати на абонентське обслуговування, недосконаліть алгоритмів, можливість технічних збоїв.

Мінітеплиця «Парничок» зображена на рис. 1.5.

Рисунок 1.5 – Мінітеплиця Парничок [14]

Запропонована міні-теплиця є проміжним вибором між великою повнорозмірною промисловою теплицею та мікротеплицею для однієї рослини. Вона має компактні розміри, що дозволяє зручно розташовувати її на прибудинковій ділянці. Однак, на відміну від повноцінних теплиць, ця міні теплиця не має автоматизованої системи підтримки мікроклімату.

Єдиним елементом теплиці є захисна плівка, яка створює ефект парника, утримуючи тепло всередині завдяки сонячному випромінюванню.

Переваги мінітеплиці: невеликі розміри, низька ціна.

Недоліки мінітеплиці: відсутність додаткового освітлення, поливу, підігріву та провітрювання.

Мікротеплиця Aerogarden Classic 6-Pod наведена на рис. 1.6.

Така мікротеплиця є прикладом мікротеплиці без стінок, що дозволяє основним джерелам світла (сонцю) проникати всередину. Однак, ця особливість обмежує можливість регулювання температури для рослин, що робить цю теплицю не придатною для умів з низькими температурами, оскільки рослина може постраждати або загинути. Крім того, дана система не має автоматизованого поливу, що ускладнює залишання рослин на тривалий час без нагляду.

Рисунок 1.6 - Мікротеплиця інтернет-сайту megazakaz.com [28]

Переваги мікротеплиці вмикають сучасний дизайн, невеликі розміри та наявність освітлення, що поліпшує умови росту рослин.

Недоліки мікротеплиці полягають у відсутності підігріву і автоматизованого поливу, що може обмежувати її функціональні можливості. Також, враховуючи обмежений функціонал, ціна даної теплиці може здатися високою.

Теплиця гроубокс Led 50 з автоматичним крапельним поливом показана на рис. 1.7.

Дана теплиця є найбільш автоматизованою серед розглянутих варіантів. Вона має штучне освітлення у фіто спектрі, автоматизований полив та вентиляцію. Також вона обладнана захисними стінками, що дозволяють захистити рослини від холодного повітря.

Переваги даної теплиці включають невеликі розміри, наявність освітлення, поливу та вентиляції, що сприяє оптимальним умовам для росту рослин.

Недоліки включають відсутність підігріву, високу ціну та невеликий корисний об’єм.

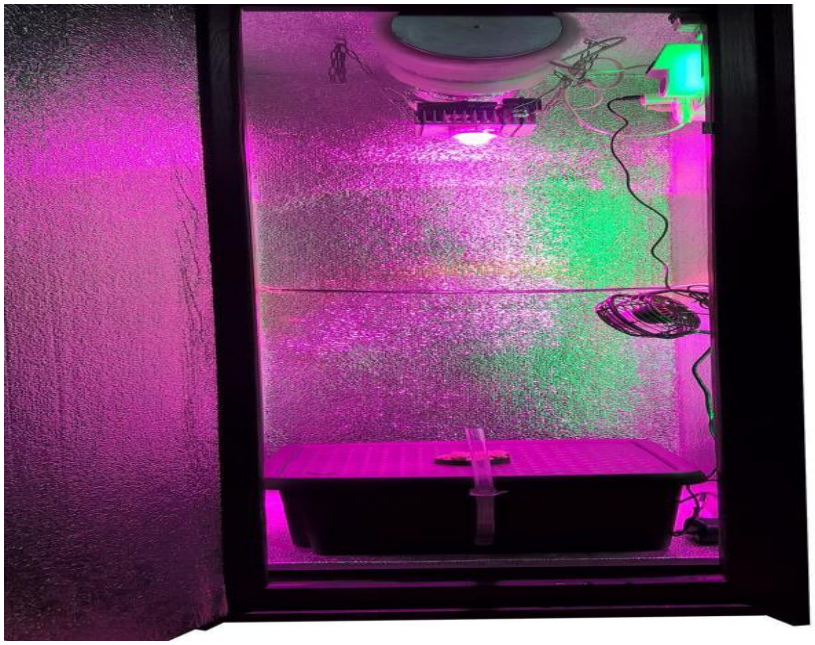

Рисунок 1.7 – Теплиця гроубокс Led 50 інтернет-сайту growboxua.com.ua [13]

Загалом, різні моделі теплиць мають свої переваги та недоліки. Наприклад, у мінітеплиці Парничок відсутнє додаткове освітлення та полив, а Мікротеплиця Aerogarden Classic 6-Pod не має належної теплоізоляції та вентиляції. Найбільш повним функціоналом володіє модель теплиці 2agrocloud, однак має свої недоліки.

Враховуючи це, було прийнято рішення розробити власний варіант автоматизованого керування теплицею, що має максимальний функціонал та доступну ціну.

1.3. Постановка задач розробки

Метою кваліфікаційної роботи є розробка та реалізація вбудованої системи автоматизації технологічних процесів вирощування рослин у теплиці через мобільний додаток на Android. Для досягнення поставленої мети необхідно виконати наступні задачі:

1. Вибір апаратної платформи, датчиків та виконавчих механізмів — визначити оптимальне обладнання, яке забезпечить ефективну роботу системи та відповідає вимогам автоматизації теплиці, зокрема для контроля температури, вологості та освітлення.
2. Розробка принципової схеми системи — створити схему, що відображає зв'язок між датчиками, виконавчими механізмами та контролером для чіткого розуміння структури системи.
3. Розробка алгоритму керування функціонуванням теплиці — створити алгоритм, який забезпечить автоматичне регулювання параметрів середовища, враховуючи показники датчиків та налаштування користувача.
4. Тестування моделі вбудованої системи автоматизації технологічних процесів вирощування рослин у теплиці — провести тестування для перевірки працездатності системи, її надійності та відповідності очікуваним вимогам.
5. Реалізація даних етапів дозволить створити систему, яка спрощує процес догляду за рослинами, підвищує ефективність використання ресурсів і забезпечує стабільне середовище для вирощування рослин.

Висновки за розділом 1

Підсумовуючи перший розділ, можна зробити такі висновки:

1. Визначено, що АСК - це технологія призначена для забезпечення автоматичного контролю, моніторингу, оптимізації та керування різними процесами і системами. Вона має на увазі використання різних систем управління для робочого обладнання, такого як: машини, процеси на заводах, котли і печі для термообробки, включення телефонних мереж, управління і стабілізація судів, літальні апарати та інші додатки та транспортні засоби з мінімальним або зменшеним втручанням людини.

2. Проаналізовано особливості сучасних систем автоматизації технологічних процесів вирощування рослин у теплиці, виявлено їх переваги, недоліки та ефективні шляхи розвитку розглянутих технологій.

3. Сформульовано мету та окреслено задачі розробки вбудованої системи для комплексної автоматизації технологічних процесів вирощування рослин у теплиці.

РОЗДІЛ 2

ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ КОМПЛЕКСНОЇ АВТОМАТИЗАЦІЇ ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ ВИРОЩУВАННЯ РОСЛИН У ТЕПЛИЦІ

2.1. Розробка функціональної схеми системи

Для систематизування та спрощення розробки алгоритмів програмного забезпечення автоматизованих систем керування процесами багаторівневою структури з високим рівнем розподілу елементів використовують функціональні схеми.

Функціональні схеми є основою розробки та подальшого вдосконалення програмного забезпечення систем керування процесами. Вони дозволяють оцінити витрати на розробку, налагодження і тестування програмного забезпечення та процесу в цілому. Вони візуалізують функціональну архітектуру, взаємодію компонентів системи як одного рівні так і між рівнями. На блок-схемах відображають функціональну логіку системи, зв'язок між вхідними даними та вихідним впливом.

Схема функціонування розроблюваної автоматизованої системи керування теплицею (далі за текстом АСКТ) наведена на рисунку 2.1. На схемі відображено взаємодію контролера зі середовищем теплиці, устаткуванням та інтерфейсом людина-машина (далі за текстом ЛМІ).

Взаємодія з середовищем теплиці здійснюється через давачі: температури та вологості повітря та ґрунту, тиску води у поливній системі, складу повітря, кислотності ґрунту, інтенсивності освітлення, тензодатчики.

Контролер взаємодіє з устаткуванням за прямими та непрямими показниками. До прямих показників відноситься струм споживання, наявність та величина напруги (електромережа, акумулятори та сонячні панелі). До непрямих показників відноситься зміна показників датчиків

середи при впливі на неї виконавчих пристроїв. Наприклад при поливі вологість ґрунту повинна збільшуватися, при включенні штучного освітлення, давач освітленості повинен це зареєструвати.

Взаємодія з користувачем здійснюється за допомогою ЛМІ. Користувач моє змогу як контролювати параметри теплиці тик і контролювати їх.

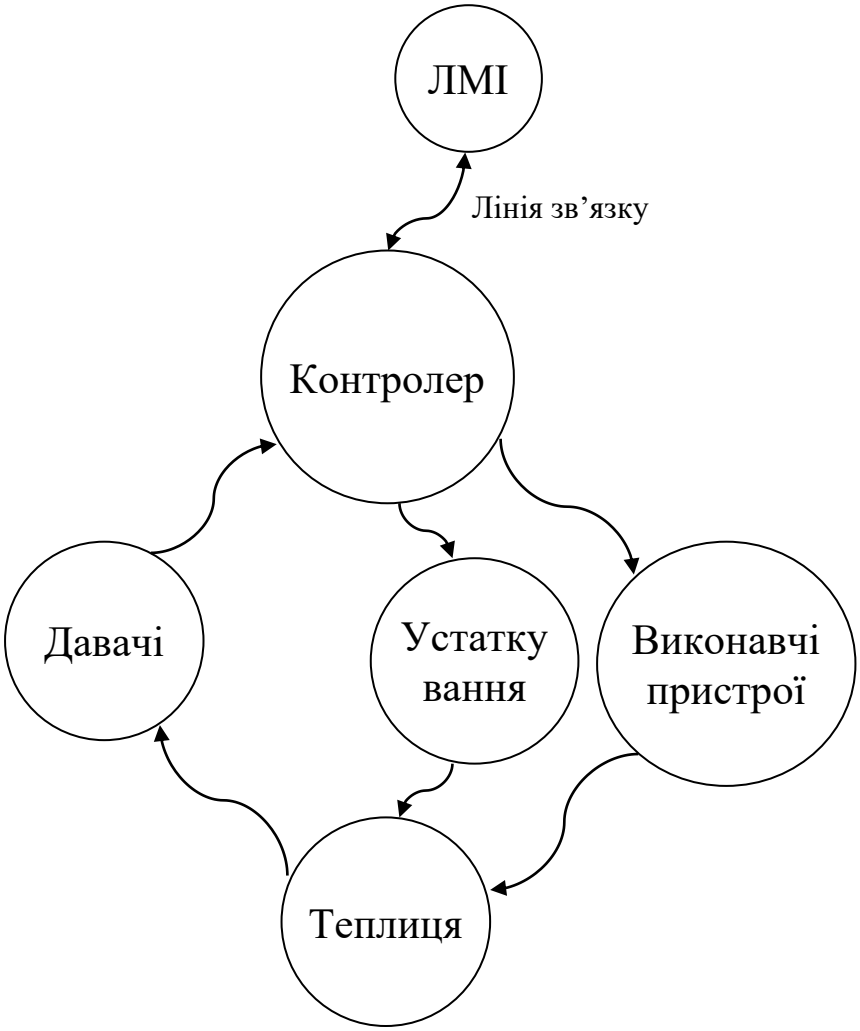

Рисунок 2.1 – Функціональна схема АСКТ

Схема АСКТ (рис. 2.1) закладається з:

- ЛМІ;
- лінії зв’язку;
- контролера;
- давачів;

- устаткування;
- виконавчих пристроїв;
- теплиці.

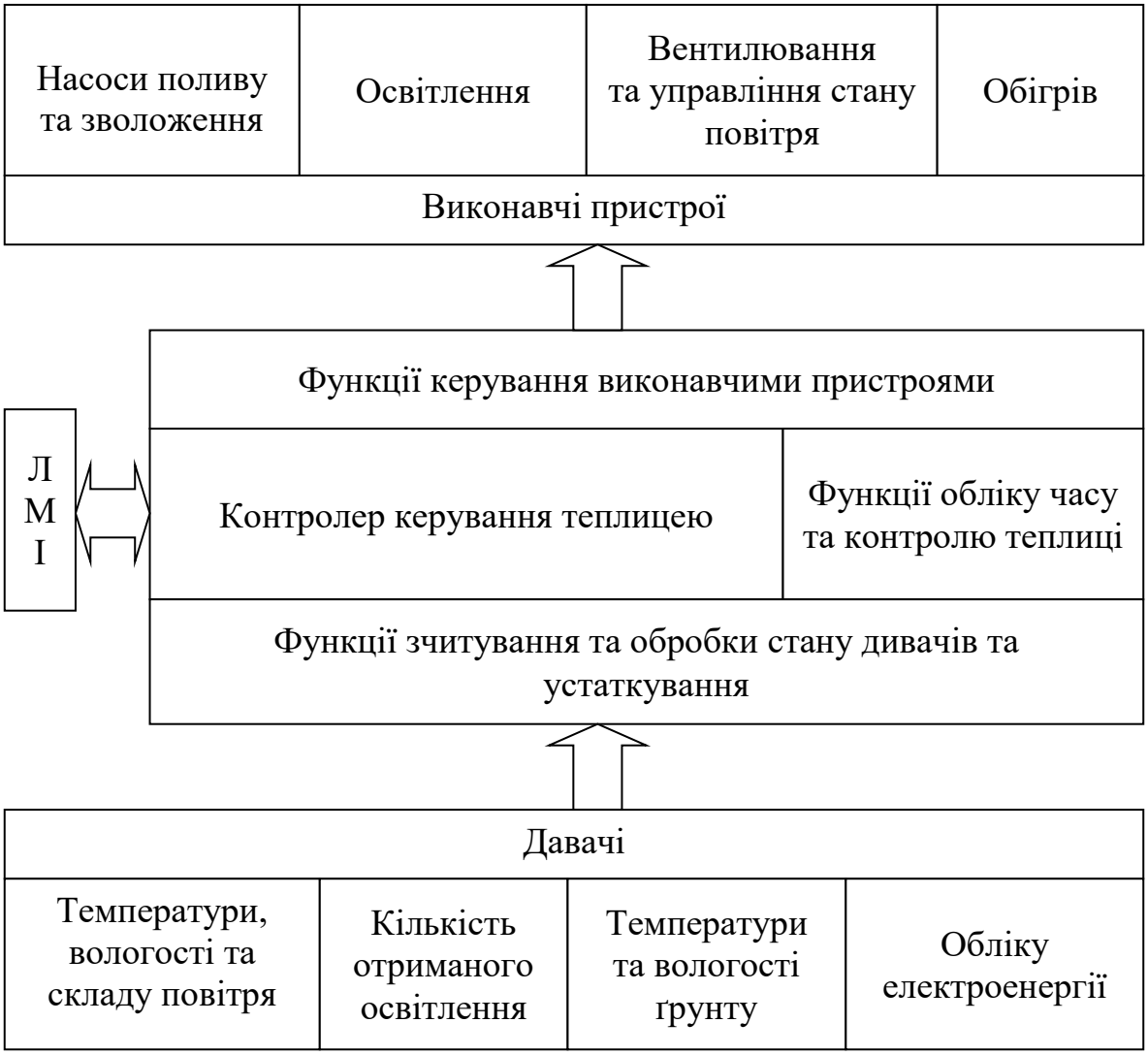

Рисунок 2.2 – Розширена функціональна схема АСКТ

Розширена функціональна схема АСКТ наведена на рисунку 2.2, відображає взаємодію складових функцій системи. Основою системи є контролер керування теплицею, який виконує функції керування виконавчими пристроями: включення/вимкнення насосів крапельного поливу та зрошування повітря; керування кількістю освітлення; складом та температурою повітря. Для коректного підтримання мікроклімату теплиці контролер виконує функції збору стану давачів: температури, вологості та

стану повітря; заміру та накопичення даних освітленості; температури, вологості та кислотного стану ґрунту; обліку використаної електроенергії. Також, контролер виконує функції обміну даними з ЛМІ.

Далі доцільно описати докладніше зв'язок функцій зі станом відповідних давачів.

Для контролю стану ґрунту (вологість, температура, кислотність) використовують відповідні давачі, отримавши їх стан функція контролю стану ґрунту приймає відповідне рішення: список, якщо вологість ґрунт нижче встановленого порога, включається насос крапельного поливу і вимикається по досягненню встановленого порогу, або за часом поливу, якщо після поливу вологість не зросла – встановлюється ознака помилки системи поливу; температура та кислотність ґрунту є показниками, які передаються у ЛМІ і у разі не усунення невідповідності встановлених меж за відповідний час, буде встановлена ознака помилки стану ґрунту.

Для контролю освітленості рослин у теплиці використовується давач освітленості. Значення освітленості накопичується та розраховується сумарне освітлення за світловий день. Відповідно до встановлених меж, функція системи приймає рішення про додаткове освітлення рослин у теплиці. Вимкнення додаткового освітлення виконується при досягненні встановлених меж.

За контроль стану повітря (вологість, температура та кількість CO_2) використовуються відповідні давачі. Функція контролю стану повітря обробляє дані з давачів, розраховує середнє значення та порівнює з встановленими межами. Якщо якийсь з параметрів вийшов за межі, вмикається відповідний виконавчий пристрій. Вимкнення пристрою відбувається при поверненні параметра у встановлені межі. Якщо змін параметру не відбулося за відведений час, то встановлюється ознака помилки стану повітря. Також ознака помилки буде встановлена під час неконтрольованому зміні параметрів. Наприклад підвищення кількості CO_2 при вимкненому виконавчому пристрої. Також функція контролю повітря

стежить за порушенням цілісності покриття теплиці (неконтрольоване зниження температури) або виникнення пожежі. При виникненні пожежі буде увімкнена система зрошування.

Контролер наділений функцією обліку витраченої електроенергії як з мережі, так і отриманої/витраченої з автономної системи живлення (акумулятори з сонячними панелями).

Завдяки зазначеним вище функціям, АСКТ стає повністю незалежною автономною системою з мінімальним втручанням людини.

2.2. Вибір апаратної платформи

Для реалізації АСКТ була вибрана, найбільш зручна, проста та надійна апаратно-програмна платформа Arduino, призначена для простої та надійної реалізації проектів у галузі автоматики, автоматизації та робототехніки. платформа має дуже розвинену апаратну складову, що представляє собою набори змонтованих друкованих плат (модулів) контролерів, сумісних датчиків і виконавчих пристроїв, дисплеїв та інтерфейсів, що виробляються як офіційним так і сторонніми виробниками. Відкрита апаратна платформа дозволяє легко реалізовувати, модифікувати, доповнювати та удосконалювати існуючі модулі. Нарощування системи додатковими функціональними модулями здійснюється через стандартизовані інтерфейси модулів.

Апаратно-програмна платформа Arduino має безкоштовну програмну оболонку (IDE) для написання програм, компіляції та програмування без додаткових програматорів. Програмна частина реалізована таким чином, щоб мати найменший поріг входу для програмування та реалізації системи автоматизації.

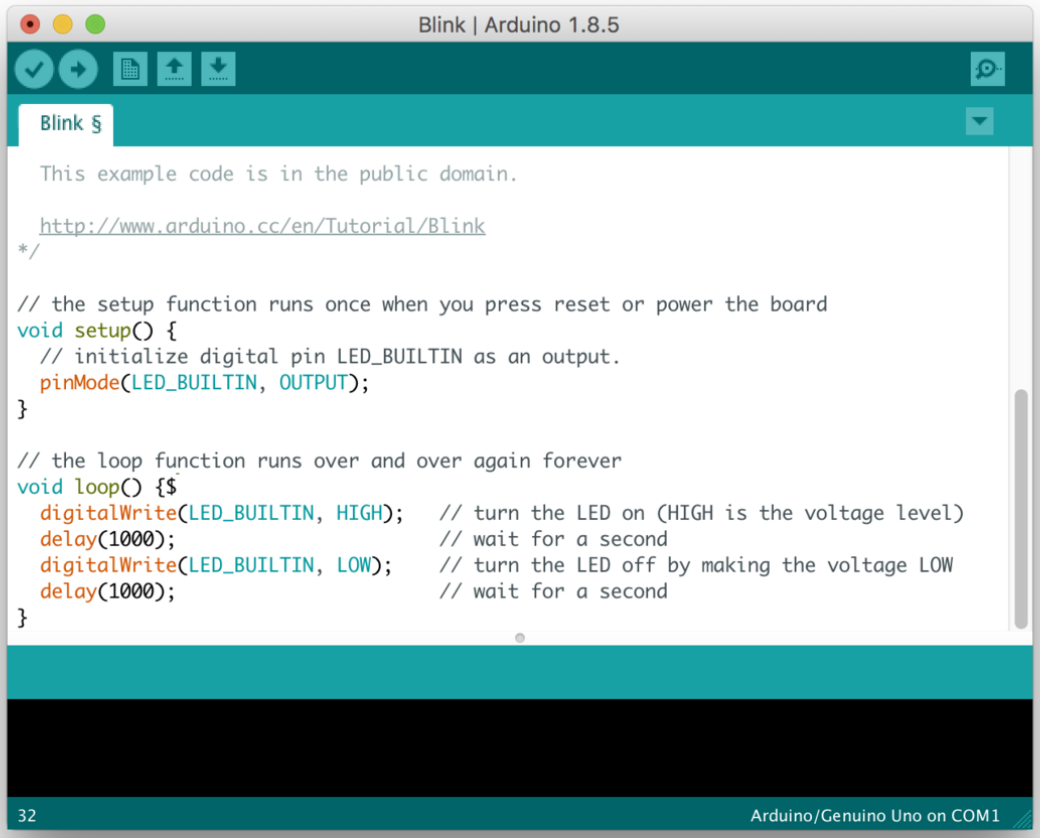

Рисунок 2.3 - Arduino IDE із текстом простої програми

Arduino поставляється з власним інтегрованим середовищем розробки (IDE), заснованим на Wiring IDE. Ця програма Java доступна безкоштовно для поширених операційних систем Windows , Linux і macOS. Він заснований на Processing IDE, середовищі розробки, що спеціалізується на графіку, моделюванні та анімації. Arduino IDE постачається з редактором коду і інтегрує gcc як компілятор. Крім того, бібліотека avr-gcc та інші бібліотеки Arduino («бібліотеки»), які значно спрощують програмування на C та C++.

Класичні Arduino та Arduino-сумісні плати спроектовані для монтажу в стопки через штирьові роз'єми. Таким чином, базову мікропроцесорну плату доповнюють необхідною периферією і зовнішніми підключеннями [20].

Існують плати Uno, Pro, Leonardo, Mega 2560, Due та плати, наприклад Zero, з розширеним набором штирьових роз'ємів для них. Плати розширення

стандартної довжини можуть встановлюватися і розширені процесорні плати (рис. 2.4).

Arduino сприймає навколишнє середовище, отримуючи дані від безлічі датчиків, і впливає на навколишнє середовище, керуючи освітленням, двигунами та іншими виконавчими механізмами [20].

Основне призначення платформи Arduino полягає у створенні і програмуванні електронних пристроїв та систем. Вона може використовуватися для розробки різноманітних проектів.

Рисунок 2.4 - Класичний конструктив Arduino з платами розширення

Arduino застосовується для організації взаємодії з іншими пристроями та виконавчими механізмами, де не потрібна повноцінна операційна система. У таких випадках, головним завданням є отримання сигналів з сенсорів та реагування на них і ця платформа ідеально впорається з цим завданням (рис.2.5).

Апаратне забезпечення типової плати Arduino засноване на мікроконтролері Microchip AVR із серії megaAVR, такому як ATmega328 [20].

Відхилення від цього можна знайти, наприклад, у платах Arduino Due (32-бітний процесор Atmel SAM3X8E Arm Cortex-M3), Yun, Tre, Gemma та Zero, де використовуються інші мікроконтролери Atmel.

Мікроконтролер (МК) являє собою невеликий комп’ютер на одному чіпі інтегральної схеми (IC). Мікроконтролер містить один або кілька

процесорів (процесорних ядер), а також пам’ять та програмовані периферійні пристрої вводу/виводу. Програмна пам’ять у вигляді фероелектричної ОЗУ, флеш-пам’яті NOR (Read-Only Memory) або OTP (One-Time Programmable) також часто включається до мікросхеми, а також невеликий обсяг ОЗУ.

Мікроконтролери розроблені для вбудованих додатків, на відміну від мікропроцесорів, що використовуються в комп’ютерах або програмах загального призначення, які складаються з різних дискретних інтегральних схем.

У сучасній термінології мікроконтролер нагадує систему на кристалі (SoC), але менш складний. SoC може підключати зовнішні мікросхеми мікроконтролера як компоненти материнської плати, але SoC зазвичай поєднує передові периферійні пристрої, такі як графічний процесор (GPU) і контролер інтерфейсу Wi-Fi, як внутрішні схеми блоку мікроконтролера [24].

Мікроконтролери використовуються в продуктах і пристроях з автоматичним керуванням, таких як системи керування автомобільними двигунами, медичні пристрої, що імплантуються, пульти дистанційного керування, офісні машини, побутова техніка, електроінструменти, іграшки та інші вбудовувані системи.

Завдяки зменшенню розміру та вартості в порівнянні з конструкцією, в якій використовується окремий мікропроцесор, пам’ять та пристрої введення/виводу, мікроконтролери роблять цифрове управління ще більшою кількістю пристроїв та процесів економічним.

Широко поширені мікроконтролери зі змішаними сигналами, що поєднують аналогові компоненти, необхідні для управління нецифровими електронними системами.

Ряд інших мікроконтролерів, таких як ESP8266, ESP32, STM32 або MSP430 також можна запрограмувати через Arduino IDE через розширення.

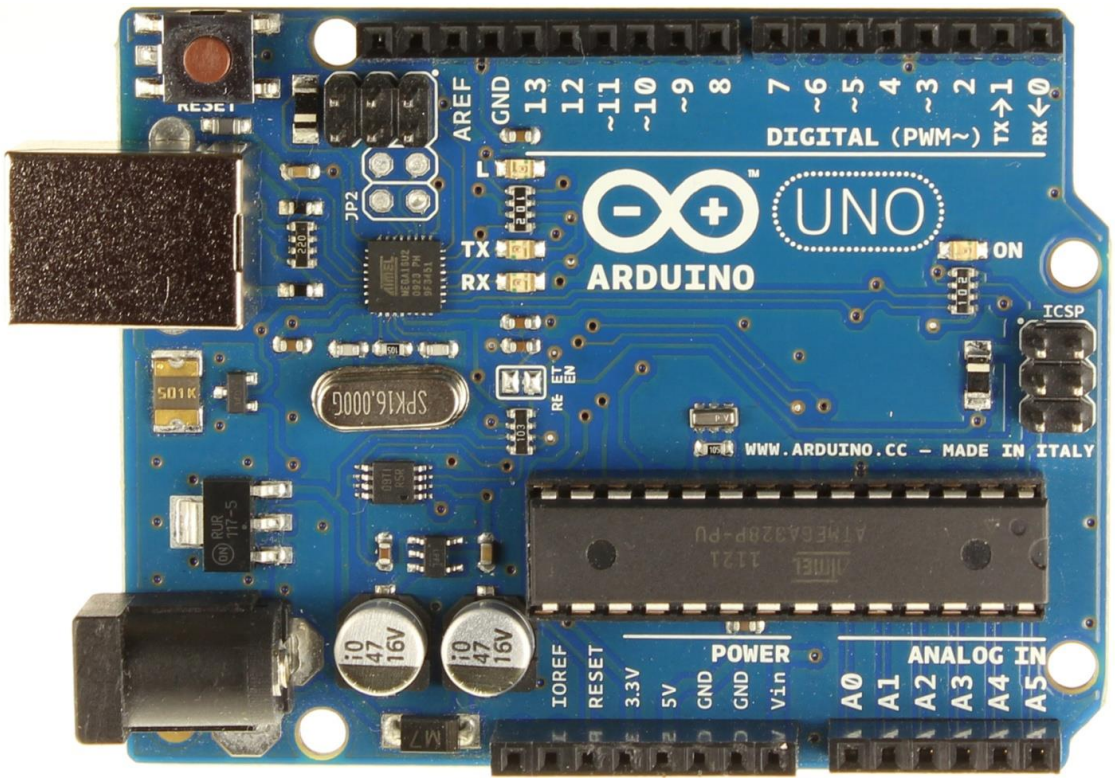

Рисунок 2.5 - Arduino UNO R3 - версія SMD з інтерфейсом USB та мікроконтролером ATmega 328

Для найбільшої гнучкості і простоти впровадження АСКТ, що реалізується, були розглянуті готові модулі, що мають достатню кількість портів вводу/виводу і мають апаратно реалізований модуль бездротової локальної мережі WiFi. Було розглянуто такі Arduino сумісні модулі:

- ESP8266;
- ESP32 ROM;
- Arduino Nano 33 IoT.

ESP8266 є мікроконтролером, виробленим компанією Espressif Systems, який має вбудований інтерфейс Wi-Fi. ESP8266 забезпечує широкі можливості для підключення до бездротових мереж Wi-Fi і передачі даних через Інтернет. Він має вбудований процесор, флеш-пам'ять, а також TCP/IP стек, що дозволяє йому здійснювати мережеву комунікацію [17].

У 2014 році ESP8266 здобув значно популярність через свою надзвичайно низьку ціну, що зробило його доступним для широкого кола

розробників. У 2016 році з'єднання явився ESP8285, який є вдосконаленим варіантом ESP8266, оскільки поєднує в собі мікроконтролер ESP8266 та флеш-пам'ять об'ємом 1 МБайт. Восени 2015 року Espressif представила свою наступну розробка - мікросхему ESP32. ESP32 є більш потужним та функціональним мікроконтролером, ніж ESP8266. Він має двоядерний процесор, підтримку Wi-Fi та Bluetooth.

Мікроконтролер ESP32 має:

- двоядерний мікропроцесор Xtensa LX6, який працює на основній частоті 240 МГц;
- вбудовану систему зв'язку Wi-Fi що підтримує стандарти бездротової мережі IEEE 802.11 b/g/n та підтримує протоколи шифрування WEP, WPA і WPA2 для забезпечення безпеки мережі;
- 14 портів вводу-виводу, з яких можна використовувати 11;
- інтерфейси SPI, I2S та UART, а також 10-бітний аналого-цифровий перетворювач (ADC).

ESP32 працює з живленням в діапазоні від 2,2 до 3,6 вольт.

Споживання енергії у режимі передачі становить до 215 мА, в режимі прийому - до 100 мА, а в режимі очікування - до 70 мА. Є також три режими зниженого споживання: Modem sleep (15 мА), Light sleep (0,4 мА) і Deep sleep (15 мкА).

Мікросхема ESP32 не має вбудованої користувальницької енергонезалежної пам'яті. Програмний код виконується з зовнішній SPI-флеш-пам'яті шляхом динамічного завантаження відповідних ділянок коду у кеш інструкцій. Підтримується до 16 МБ зовнішньої програмної пам'яті. Інтерфейс може бути налаштований на режими Standard, Dual або Quad SPI.

Інформацію про електричні параметри, цоколівки та схеми включення мікроконтролера ESP8266 можна знайте в документах 0A-ESP8266EX__Datasheet і 0B-ESP8266__System_Description, що надаються в рамках Espressif SDK.

Стан портів GPIO0, GPIO2 і GPIO15 в момент закінчення сигналу Reset (під час подачі живлення) визначає джерело програми для мікроконтролера ESP8266.

У даній системі передбачено два режими роботи. Перший режим - режим виконання коду UART - використовується для перепрошивки підключеної флеш-пам'яті. Цей режим дозволяє змінювати програмне забезпечення, що зберігається на флеш-пам'яті, шляхом передачі спеціальних команд через UART інтерфейс. Другий режим - штатний режим роботи - призначений для нормальної роботи системи. У цьому режимі мікроконтролер виконує свої основні функції і взаємодіє зі зовнішніми пристроями та периферійними пристроями, не пов'язаними зі зміною флеш-пам'яті.

Розглянемо засоби розробки.

Програмні засоби розробки (програмний комплект розробника, SDK) складаються з:

- компілятор та компонувальник перетворює програмний код, написаний у мові програмування, на виконуваний бінарний файл, який може бути завантажений на ESP8266.
- бібліотеки: Набір готових функцій і класів, які спрощують розробка програмного забезпечення для ESP8266. Ці бібліотеки можуть включати функції для керування Wi-Fi, GPIO, UART, SPI, I2C та іншими інтерфейсами, а також функції для роботи з різними протоколами і пристроями;
- засобів завантаження файлу, що виконується, в пам’ять програм мікроконтролера.

Espressif безкоштовно розповсюджує свої пакети розробки. Набір інструментів включає компілятор GCC, бібліотеку Espressif і завантажувач XTCOM. Бібліотеки постачаються як скомпільовані бібліотеки без вихідного тексту. Espressif підтримує дві версії SDK: одну на основі RTOS і іншу на основі зворотних викликів.

Мережева інфраструктура.

ESP8266 та ESP32 є популярними апаратними рішеннями для розробки систем Інтернету промов (IoT) у будинках та офісах. Обидва мікроконтролери можуть бути підключені до домашньої або офісної локальної мережі та отримувати доступ до Інтернету через роутер. Користувач може контролювати пристрої, що працюють на базі ESP8266 або ESP32, за допомогою планшета або комп'ютера, як локально, так і віддалено через Інтернет.

ESP32 є розширеною версією ESP8266 і має додаткові можливості. Він має в собі інтегровані контролери Wi-Fi, Bluetooth і Thread, що дозволяє реалізувати різноманітні бездротові комунікації. ESP32 доступний у різних серіях, таких як ESP32, ESP32-S, ESP32-C і ESP32-H, з різними процесорними ядрами, включаючи ядра компанії Tensilica та ядра з відкритою архітектурою RISC-V. В мікросхему ESP32 також вбудовано радіочастотний тракт з симетричним трансформатором, антенними комутаторами, малошумним підсилювачем, підсилювачем потужності, фільтрами та модулями керування живленням.

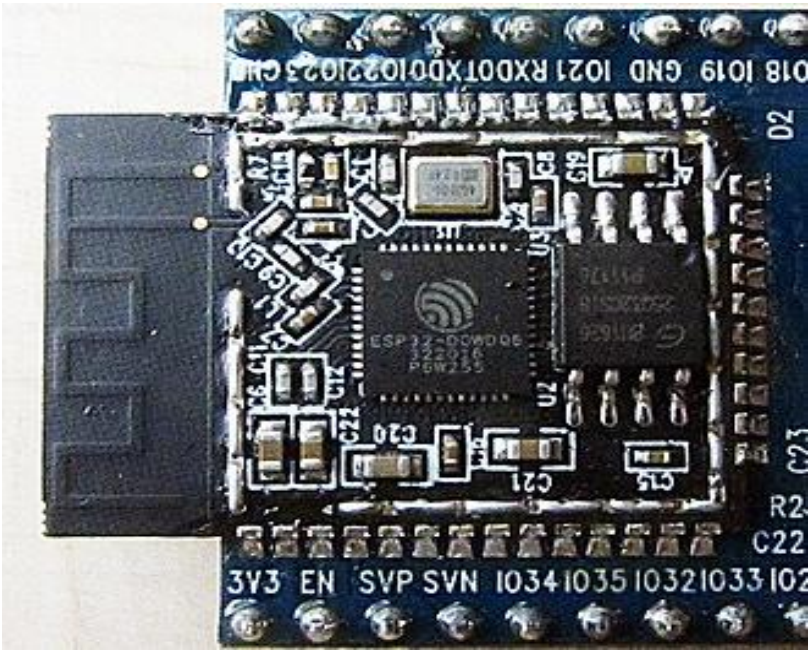

Рисунок 2.6 - Зовнішній вигляд мікросхеми ESP32

Особливості ESP32 [17]:

- ESP32 може отримати доступ до флеш-пам'яті QSPI (Quad serial peripheral interface) і SRAM (static random access memory) за допомогою високошвидкісного каналу;
- до 16 Мб зовнішньої флеш-пам’яті зіставлені з кодовим простором ЦП, що підтримує 8, 16 і 32-біт доступу;
- до 8 Мб зовнішньої flash/SRAM карти пам’яті на ЦП простору даних, підтримка 8, 16 та 3 2-біт доступу, читання даних підтримується на флеш-пам’яті та SRAM. Запис даних підтримується SRAM;
- ESP32-WROVER інтегрує 4-16 Мб зовнішньої SPI flash може бути картка пам’яті на процесор простір, що підтримують 8, 16 та 32 біт доступу;
- підтримується виконання коду;
- крім 4-16 МБ SPI flash, ESP32-WROVER також інтегрує 4-8 Мб PSRAM (Dynamic random-access memory) для більшого простору пам’яті;
- мікропрограма ESP32 Wi-Fi/BT може підтримувати лише кварцовий генератор 40 МГц.

RTC (Real-Time Clock) відповідає за вимірювання та відстеження часу у реальному світі в системі. Вона забезпечує точний розрахунок годин, хвилин, секунд та дати. RTC зазвичай використовується для синхронізації різних процесів, планування подій та відстеження часу в пристроях і системах.

Управління низьким споживанням енергії (Power Management) включає в себе набір технологій і стратегій, що дозволяють зменшити споживання енергії пристроєм або системою в режимах очікування або невикористання. Це може включати режими сну, зменшення тактової частоти процесора, відключення непотрібних пристроїв та оптимізацію роботи апаратно-програмного комплексу для забезпечення ефективного використання енергії.

RTC та управління низьким споживанням часто використовуються разом для забезпечення ефективного використання енергії в пристроях. Наприклад, RTC може використовуватись для переведення пристрою в

режим сну або очікування, коли його робота не потрібна, забезпечуючи енергозбереження. Крім того, управління низьким споживанням може контролювати живлення RTC і інших компонентів пристрою, забезпечуючи оптимальне використання енергії та продовжуючи тривалість роботи пристрою від батареї або іншого джерела електроживлення.

За допомогою сучасних технологій керування живленням ESP32 може перемикатися між різними режимами живлення.

Потужність режими або Power modes відносяться до різних станів роботи пристрою або системи, які впливають на їх споживання енергії. Ці режими дозволяють змінювати рівень споживання енергії в залежності від потреб і вимог до пристрою, що може бути корисним для економії енергії та продовження тривалості роботи на батареї.

Active mode / Активний режим: чіп радіо увімкнено. Чіп може отримувати, передавати чи слухати.

Modem-sleep mode / Режим сну модему: ЦП працює і годинник налаштовується. Базова смуга Wi-Fi/Bluetooth I радіо вимкнено.

Arduino Nano 33 IoT – це універсальне рішення для багатьох завдань. базові сценарії програм IoT з вбудованим модулем бездротової локальної мережі WiFi. Основним процесором плати є 32-розрядний процесор Arm® Cortex®-M0 із низьким енергоспоживанням SAMD21. З'єднання Wi-Fi і Bluetooth® здійснюється за допомогою u-blox модуля NINA-W10 - чіпсету з низьким енергоспоживанням, що працює в діапазоні 2.4 ГГц. Крім того, безпечний зв'язок забезпечується за допомогою крипточипу Microchip ECC608.

В Arduino IoT легко підключити до існуючої мережі Wi-Fi або використовувати її для створення власної точки доступу Arduino. Модуль має великий набір прикладів та бібліотек (WiFiNINA). Також модуль має можливість підключення до різних хмарних послуг:

- хмарний сервіс IoT від Arduino - простий та швидкий спосіб забезпечити безпечний зв'язок для всіх ваших підключених речей;

- хмарний сервіс Blynk - простий проект спільноти, що підключається до Blynk для керування платою з телефону з невеликою кількістю коду;
- хмарний сервіс IFTTT;
- хмарний сервіс AWS IoT Core (містить приклад підключення до Amazon Web Services);
- хмарний сервіс Azure (має детальний опис на репозиторії GitHub, де пояснюється, як підключити датчик температури до хмари Azure);
- хмарний сервіс Firebase, що з легкістю інтегрується до портативних пристроїв на базі Android.

Комунікаційний чіпсет Nano 33 IoT може бути як клієнтом, так і хост-пристроєм Bluetooth® та Bluetooth® Low Energy. Щось досить унікальне у світі платформ мікроконтролерів (все необхідне реалізовано у бібліотеці ArduinoBLE).

Nano 33 IoT – це двопроцесорний пристрій що дозволяє використовувати на платі одночасно Wi-Fi, Bluetooth® та Bluetooth® Low Energy. Ще одна можливість - використовувати надлегку версію Linux, що працює на модулі, в той час як основний мікроконтролер керує пристроями низького рівня, такими як двигуни або екрани.

Характеристики:

- Arduino Nano 33 IoT заснований на мікроконтролері SAMD21 (Cortex®-M0+ 32-бітний мікроконтролер ARM з низьким енергоспоживанням);
- радіомодуль u-blox NINA-W102;
- безпечний елемент ATECC608A;
- робоча напруга 3.3;
- максимальна вхідна напруга живлення: 21 В;
- постійний струм на контакт введення-виведення: 7 мА;
- тактова частота: 48 МГц;
- флеш-пам'ять ЦП: 256 КБ;
- оперативна пам'ять: 32 КБ;
- контакти цифрового введення/виводу: 14;

- ШИМ-контакти: 11 (2, 3, 5, 6, 9, 10, 11, 12, 16/A2, 17/A3, 19/A5)
- UART: 1;
- SPI: 1;
- I2C: 1;
- аналогові вхідні контакти: 8 (АЦП 8/10/12 біт) ;
- аналогові вихідні контакти: 1 (ЦАП 10 біт) ;
- зовнішні переривання: Усі цифрові контакти (всі аналогові контакти також можуть використовуватися як контакти переривання, але будуть мати дублюючі номери переривань);
- LED_BUILTIN: 13;
- USB: Вбудований у процесор SAMD21;
- розмір плати: 45 x 18 мм.

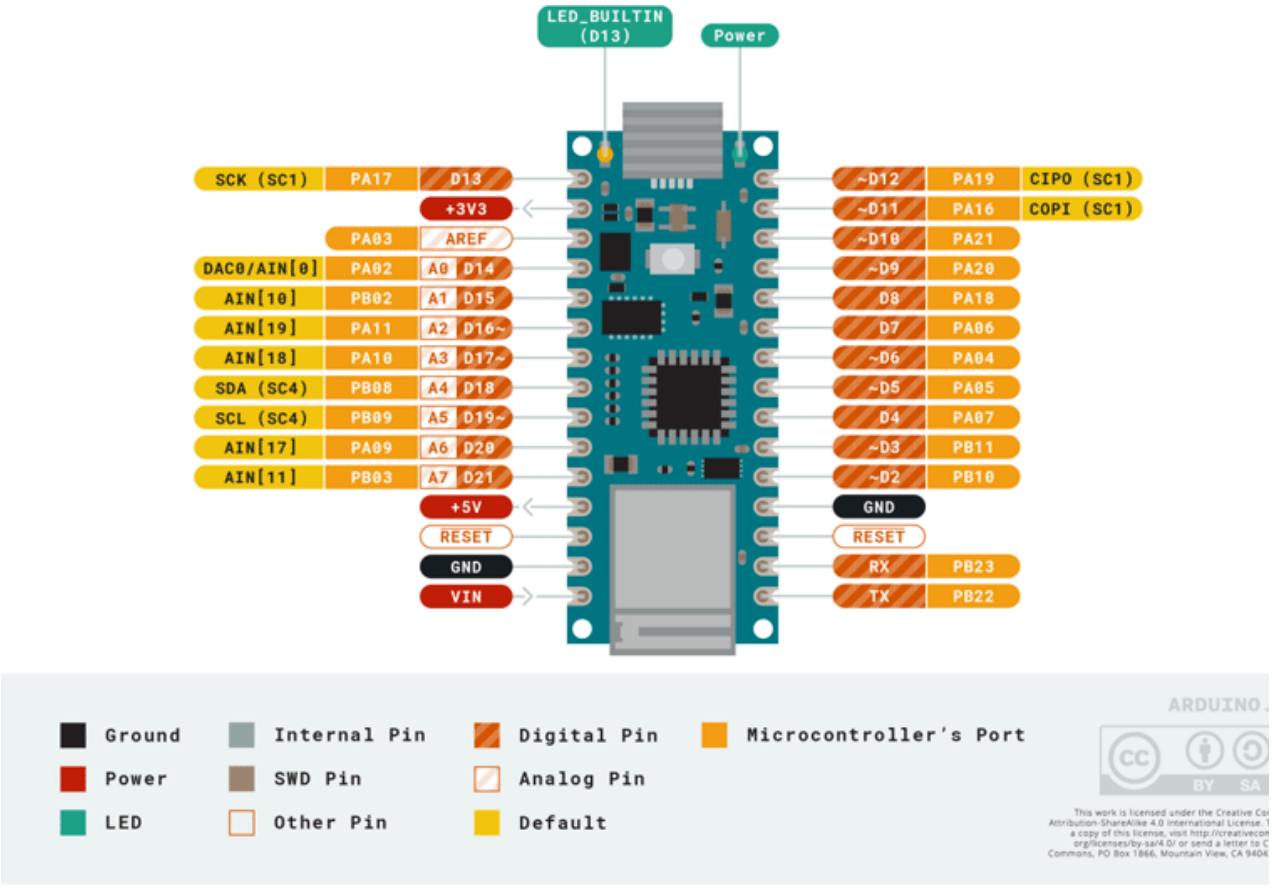

Рисунок 2.7 - Назва та призначення виводів модуля Arduino Nano 33 IoT

Мікроконтролер Arduino Nano 33 IoT було обрано для реалізації проекту АСКТ як найбільш розвинутий модуль з великим набором функцій реалізованих у супутніх бібліотеках.

У таблиці 2.1 наведено параметри розглянутих модулів. Відповідно параметрів, модуль Arduino Nano 33 IoT має такий самий процесор як і ESP32, але частота роботи ядра процесора вища і складає 240 МГц, що стане у нагоді при обробці даних та виведенні їх до інтерфейсу користувача, а також у подальшому вдосконаленні програмного забезпечення та розвинення функціональності АСКТ.

Таблиця 2.1. Параметри розглянутих модулів ESP8266, ESP32, Arduino Nano 33 IoT

Модуль / Параметр	ESP8266	ESP32	Arduino Nano 33 IoT
Процесор	Xtensa ® Single-Core 32 bit L106	Xtensa ® Dual-Core 32 bit LX6	Xtensa ® Dual-Core 32 bit LX6
Кількість ядер, частота, МГц	1, 80	2, 160	2, 240
WiFi	так, HT20	так, HT40	так, HT40
Bluetooth	ні	так, V4.2	так, V4.2
ОЗП, кБайт	160	512	520
ПЗУ, МБайт	16	16	2
Портів В/В	17	36	22
SPI/I2C/I2S/UART/LIN	2/1/2/2/0	4/2/2/2/0	6/6/2/6/1
АЦП, біт	10	12	12
CAN	ні	1	ні
Ethernet	ні	1	ні
Сенсор дотику	ні	1	ні
Давач температури	ні	1	ні

2.3. Датчики та виконавчі механізми

Для взаємодії АСКТ з середою теплиці застосовані наступні давачі:

1. Давач вологості та температури повітря DHT22, який містить у собі високоточний цифровий перетворювач та здатний вимірювати температуру у діапазоні -40..80 градусів з роздільною здатністю 0.1 °C, вологість у діапазоні 0..99.9 % з роздільною здатністю 0.1%, модуль містить всі необхідні додаткові компоненти для підключення датчика до мікроконтролеру [36], зовнішній вигляд наведено на рисунку 2.8.

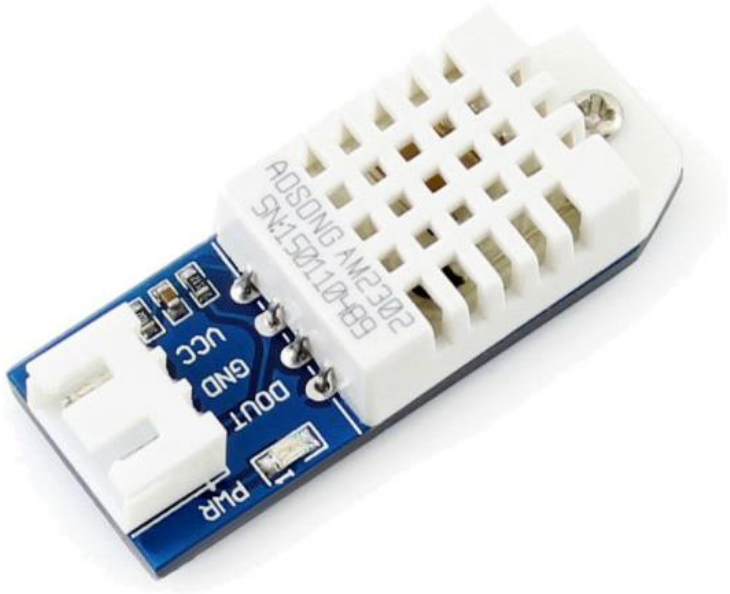

Рисунок 2.8 - Зовнішній вигляд давача вологості та температури DHT22

2. Ємнісний давач вологості ґрунту (Capacitive Soil Moisture Sensor) має великий строк експлуатації бо не має безпосереднього контакту з вологою та активною складовою ґрунту, корозійностійке покриття електродів робить їх практично вічними [36]. Також це позитивно впливає на стабільність показань та точність вимірювання. Модуль давача має вбудований стабілізатора напруги живлення, що також позитивно впливає на стабільність показань і дає можливість живити датчик напругою від 3,3 до 5,5В. У комплекті є кабель підключення, що дозволяє дуже просто і швидко

підключити датчик до контролера або плати розширення. Зовнішній вигляд ємнісного давача вологості ґрунту Capacitive Soil Moisture Sensor наведено на рисунку 2.9.

Рисунок 2.9 - Зовнішній вигляд ємнісного давача вологості ґрунту Capacitive Soil Moisture Sensor

3. Цифровий давач температури ґрунту DS18B20 в захисному водонепроникному корпусі з пиловологозахистом IP67, має діапазон вимірювання температури від -55 °C до + 100 °C (захисна оболонка датчика зроблена з ПВХ тому рекомендований верхній діапазон виміру обмежений 100 °C). Перетворювач має 12-бітну розрядну здатність [36]. Сам вимірювальний елемент DS18B20 розміщений в герметичному пиловологозахищеному корпусі, що забезпечує максимальний ступінь захисту давача і дозволяє проводити вимірювання температури в будь-яких умовах вологості, запиленості, а також при повному зануренні датчика в рідину або ґрунт. Робоча напруга дані/живлення від +3 до +5.5 В. Зовнішній вигляд цифрового давача температури ґрунту DS18B20 в захисному водонепроникному корпусі з пиловологозахистом IP67 наведено на рисунку 2.10.

Рисунок 2.10 - Зовнішній вигляд цифрового давача температури ґрунту DS18B20 в захисному водонепроникному корпусі з пиловологозахистом IP67

4. Давач SEN0159 концентрації CO_2 з високочутливим датчиком MG-811. Датчик має низьку чутливість до парів води та алкоголю, має промислове виконання та просте підключення до контролера [36]. Зовнішній вигляд давача концентрації CO_2 SEN0159 наведено на рисунку 2.11.

Рисунок 2.11 - Зовнішній вигляд давача концентрації CO_2 SEN0159

5. Давач освітленості OPT101. OPT101 - це фотодіод великої площі, інтегрований з оптимізованим операційним підсилювачем, простий у використанні перетворювач інтенсивності світла у напругу. Фотодіод має дуже велику площу вимірювання, яка збирає значну кількість світла, що дозволяє проводити вимірювання з високою чутливістю. Фотодіод має широкий спектральний відгук з максимальним піком в інфрачервоному спектрі та корисний діапазон від 300 нм до 1100 нм. Широкий діапазон живлення від 2,7 В до 36 В дозволяє застосовувати перетворювач у різноманітних архітектурах: від повністю аналогових схем до базових схем перетворення даних. Вбудоване джерело напруги підтримує підсилювач у оптимальному стані робочій зоні, навіть при слабкому освітленні. Вихідна напруга OPT101 є добутком струму фотодіода на резистор зворотного зв’язку (IDRF). Струм фотодіода пропорційний потужності випромінювання, що падає на фотодіод. Високоточний резистор внутрішнього зворотного зв'язку підстроюється лазером до 1 МОм. Використовуючи цей резистор, чутливість

вихідної напруги становить приблизно 0,45 В/мкВт на довжині хвилі 650 нм. Залежність вихідної напруги від освітленості наведена на рисунку 2.12 [36], а зовнішній вигляд давача освітленості OPT101 наведено на рисунку 2.13.

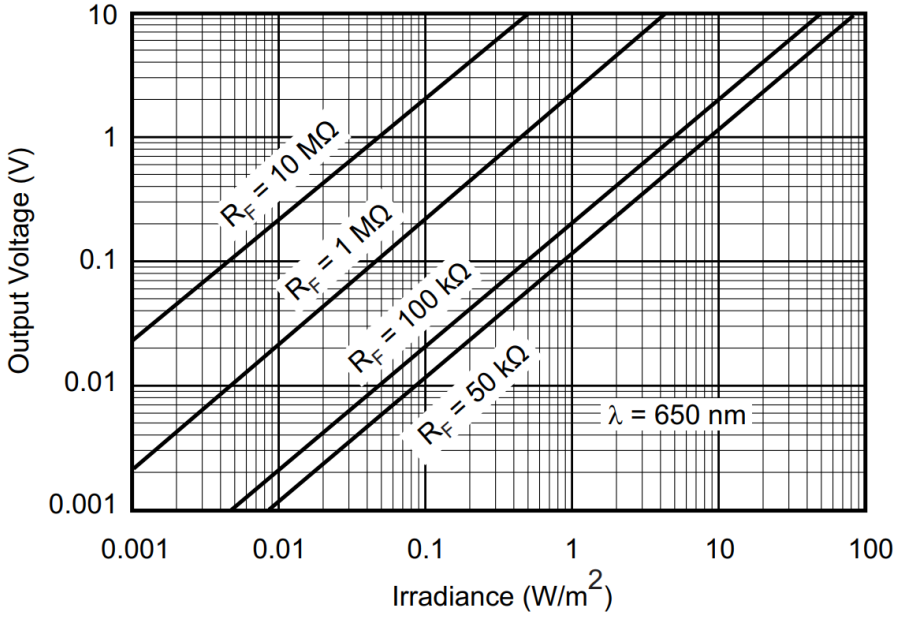

Рисунок 2.12 - Залежність вихідної напруги від освітленості давача OPT101

6. Модуль годинника реального часу (RTC) побудований на мікросхемі DS3231SN. Вона має високу точність ходу годинника завдяки вбудованому кварцовому резонатору в корпус мікросхеми, температурній компенсації та цифровій корекції частоти генератора [36].

Основні характеристики:

- висока точність годинникового генератора з термокомпенсацією та корекцією ходу;
- лічильники секунд, хвилин, годин, днів тижня, днів, місяців та років з календарем з корекцією високосного року до 2100 року;
- стабільність генератора ±2 ppm в діапазоні температур від 0°C to +40 °C;
- простий та поширений інтерфейс підключення;
- мале споживання від резервного джерела.

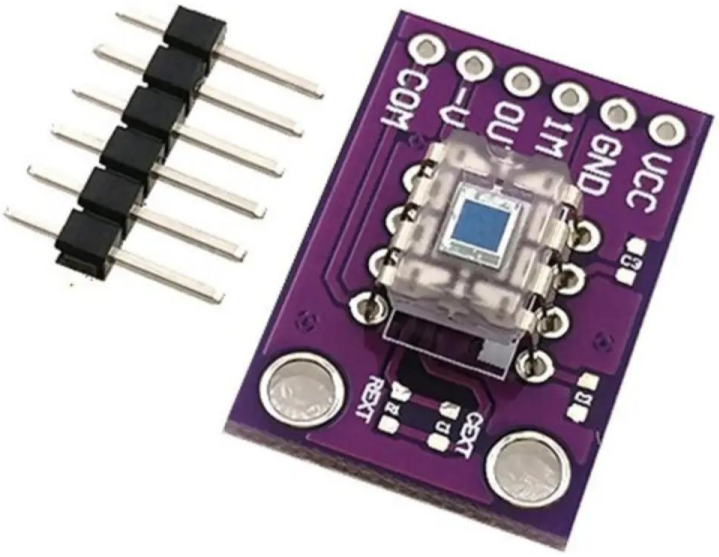

Рисунок 2.13 - Зовнішній вигляд давача освітленості OPT101

Зовнішній вигляд модуля годинника реального часу DS3231SN наведено на рисунку 2.14.

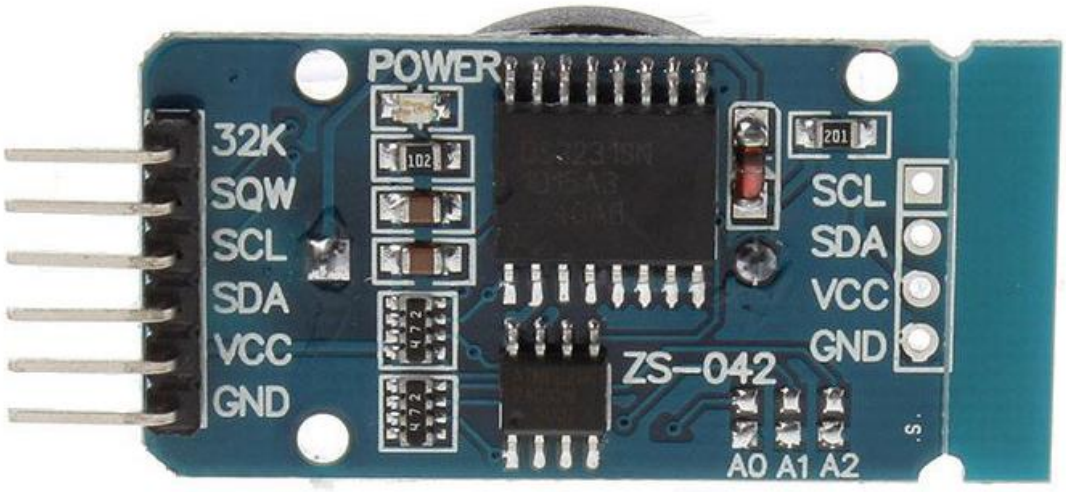

Рисунок 2.14 - Зовнішній вигляд модуля годинника реального часу DS3231SN

7. Датчик потоку води FS400A складається із пластикового клапана з двома штуцерами з різьбленням 1 дюйма, водного ротора та датчика Холла [36]. Коли вода проходить крізь датчик, ротор обертається із швидкістю пропорційної швидкості потоку води. Датчик Холла фіксує кожен оберт і

передає отриманий сигнал. Робоча напруга давача становить 5-24 В, вимірюваний діапазон складає 1~30 літрів за хвилину та 450 імпульсів на літр. Зовнішній вигляд датчика потоку води наведено на рисунку 2.15.

Рисунок 2.15 - Зовнішній вигляд датчика потоку води

8. У якості датчика ваги використані спеціалізований модуль 24-х бітного АЦП HX711(зовнішній вигляд відображено на рисунку 2.16) та тензодатчик ваги (зовнішній вигляд відображено на рисунку 2.17).

Модуль АЦП HX711 дозволяє дуже точно зчитувати показання з тензодатчиків. Модуль простий у підключенні, має два роз'єми, один для підключення до мікроконтролеру та управління самим датчиком, а інший для подачі на датчик живлення. Тезодатчики підключаються до плати розробника через роз'єм J1. Живлення на датчики подається за допомогою роз'єму JP2. Є два види каналів A і B. Каналу A можна задати більший або менший коефіцієнт посилення, що дозволяє використовувати тензодатчики розраховані на вагу від 1 кг до 500 кг [36].

Технічні характеристики:

- робоча напруга: 5В;

- частота оновлення: 80Гц;
- точність перетворення: 24біт (D конвертер);
- робочий струм: менше 10мА;
- диференціальний вхід з напругою: приблизно 40 мВ;
- можливість вибору коефіцієнта підсилення: 2, 64 або 128;
- швидкість виведення даних (частота): 10Гц або 80Гц.

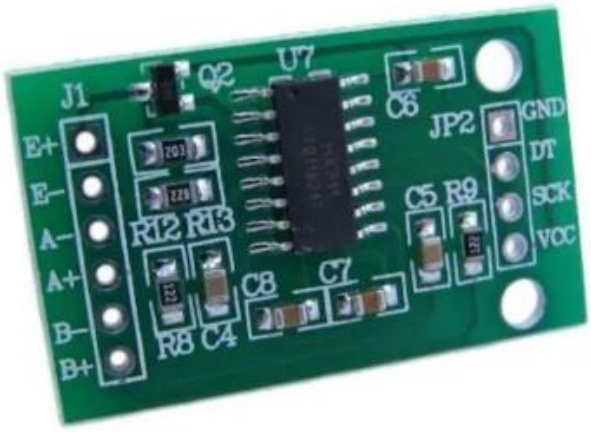

Рисунок 2.16 - Зовнішній вигляд модуля 24-х бітного АЦП HX711

Для виміру ваги було обрано два тензодатчики розраховані на вагу 20 кг. Що дозволить зчитувати вагу шпалери до 40 кг з точністю 2,4 мг.

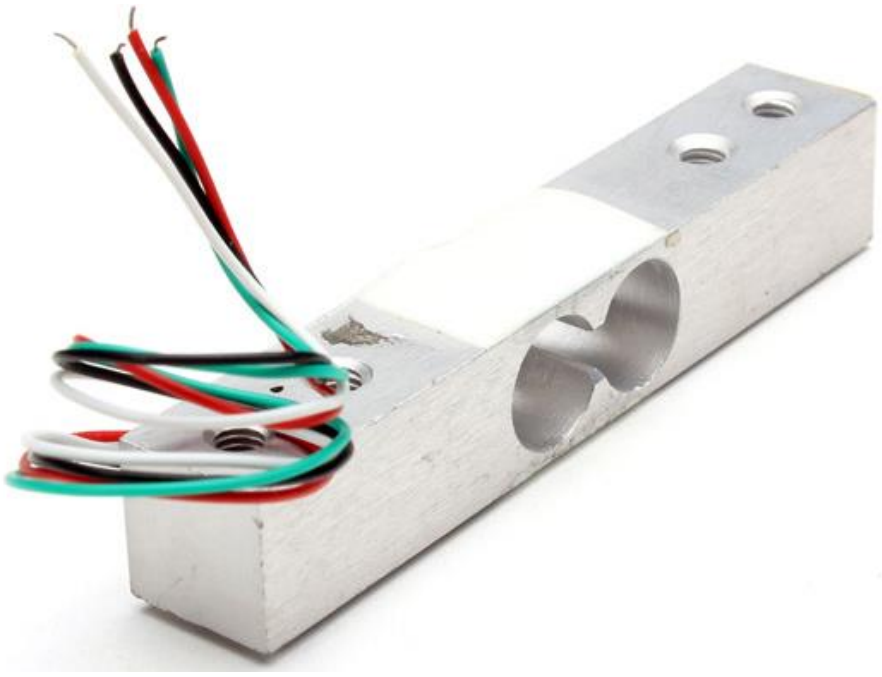

Рисунок 2.17 - Зовнішній вигляд тензодатчика розрахованого на вагу 20 кг

9. Датчик pH DFRobot Gravity - аналоговий датчик/вимірювач pH V2 спеціально розроблений для вимірювання pH розчину і відображення кислотності або лужності. Це зазвичай використовується в різних додатках, таких як аквапоніка, аквакультура і тестування води в навколишньому середовищі. Вбудована мікросхема стабілізатора напруги дозволяє використовувати джерело напруги від 3,3 В до 5,5 В, який сумісний з основною платою управління 5 В і 3,3 В. Вихідний сигнал, відфільтрований апаратно, має низький рівень цифрового шуму.

Зазвичай значення pH становить від 0 до 14. У стандартних термодинамічних умовах pH = 7, що означає, що розчин є нейтральним; pH <7, що означає, що розчин є кислим; pH> 7, що означає, що розчин є лужним [36]. Зовнішній вигляд датчика та модуль підсилювача зображено на рисунках 2.18, 2.19.

Рисунок 2.18 - Зовнішній вигляд датчика pH

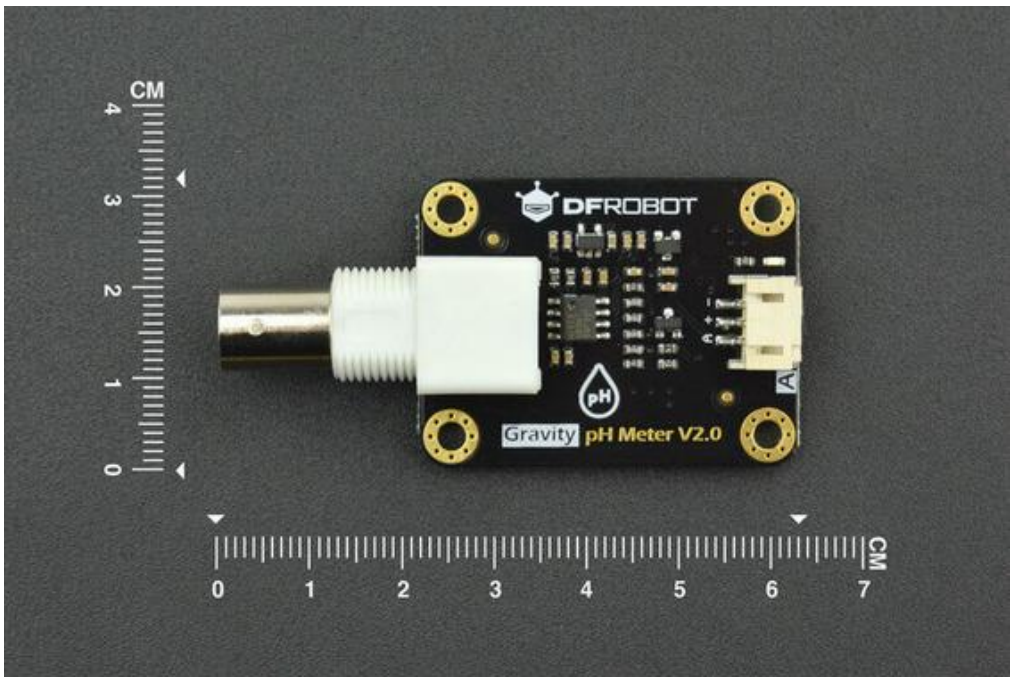

Рисунок 2.19 - Зовнішній вигляд модуля підсилення датчика pH

До виконавчих механізмів відносяться клапани води, газу, освітлення, підігрів, вентилювання, аварійна сирена. Усі виконавчих механізми керуються за допомогою модуля ізольованого реле з напругою керування 3-5 В. Така реалізація дозволяє з'єднувати будь які елементи незалежно від струму живлення [36]. Електрична схема модуля реле наведена на рисунку 2.20, а зовнішній вигляд – на рисунку 2.21.

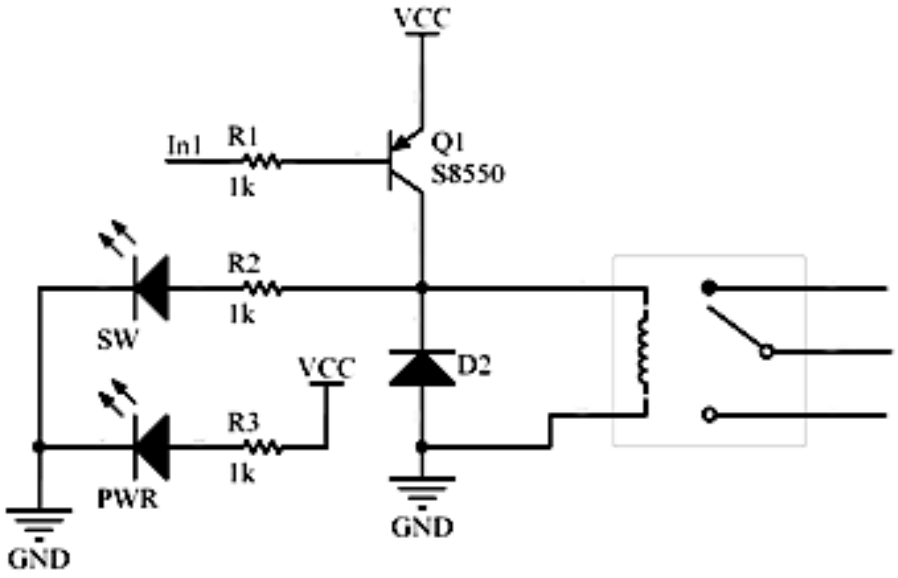

Рисунок 2.20 - Схема електрична принципова модуля ізольованого реле

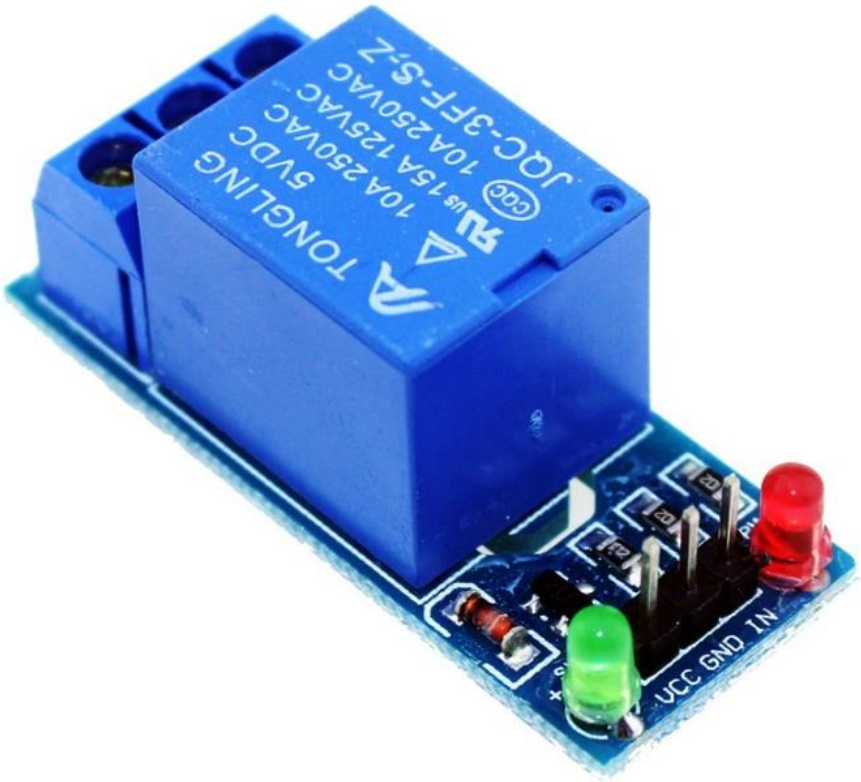

Рисунок 2.21 - Зовнішній вигляд модуля ізольованого реле

2.4. Схема системи автоматизації технологічних процесів вирощування рослин у теплиці

Для реалізації АСКТ була розроблена електрична схема, яка з'єднує контролер з усіма необхідними давачами, елементами керування, виконавчими пристроями і елементами живлення. Схема наведена на рисунку 2.22. Основою схеми є модуль Arduino Nano IoT 33 з інтегрованим модулем зв'язку WiFi. Давачі, такі як: давач вологості та температури повітря DHT22, ємнісні давач вологості ґрунту, давач SEN0159 концентрації CO_2, давач освітленості OPT101 під'єднані до аналогових входів модуля. Датчик потоку води FS400A під'єднано до цифрового входу. Модуль годинника реального часу приєднано до інтерфейсу I2C.

Модулі ізольованих реле, що керують виконавчими механізмами, приєднано до цифрових виходів модуля Arduino Nano IoT 33.

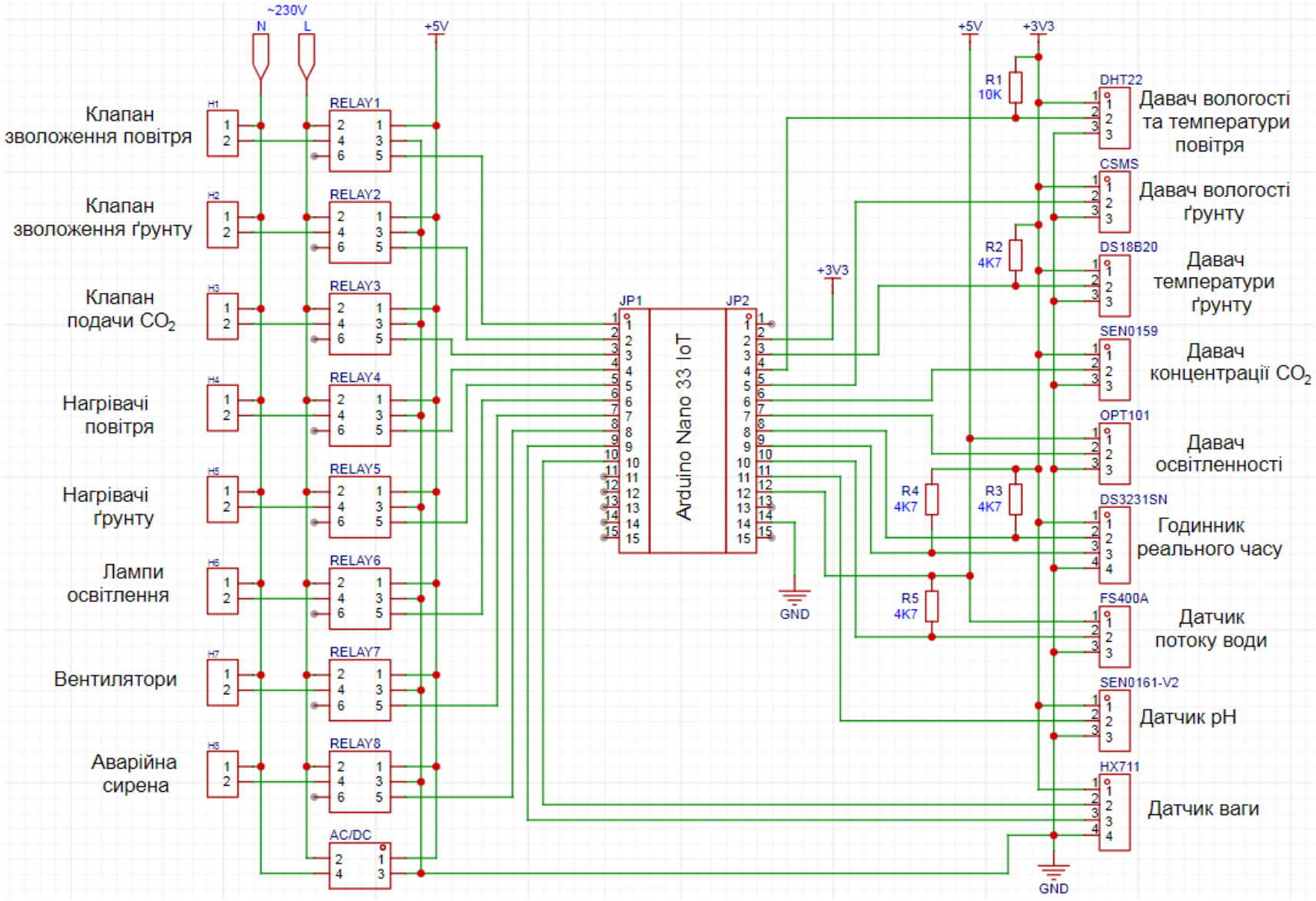

Рисунок 2.22 - Схема електрична принципова АСКТ

Живлення всієї схеми здійснюється імпульсним блоком живлення AC/DC OV-R5-2A (рис. 2.21). Блок забезпечує перетворення змінної напруги 220 В у постійну 5 В. Потужність блока складає 10 Вт, що при вихідній напрузі 5 В дозволяє отримати до 2 А струму.

Рисунок 2.23 - Перетворювач змінної напруги 220 В у постійну 5 В потужністю 10 Вт OV-R5-2A

Висновки за розділом 2

Підсумовуючи другий розділ, можна зробити наступні висновки:

1) розроблено функціональну та принципову електричну схеми АСКТ, яка виконує всі необхідні функції для автономної підтримки мікроклімату теплиці з мінімізацією втручання користувача. Це було досягнуто завдяки використанню сучасної елементної бази (контролер, необхідні здавачі та виконавчі механізми) та, як прямого так і непрямого, контролю поточних процесів. Користувач лише на початку задає необхідні параметри клімату

під вибрані рослини і АСКТ підтримує їх весь час зрощування, та сигналізує користувачеві про помилки та аварійні випадки.

2) Зроблено обґрунтований вибір давачів та виконавчих механізмів для АСКТ. Для відстеження параметрів температурі, вологості, освітленості, складу повітря вибрані:

- давач вологості та температури повітря DHT22;
- ємнісний давач вологості ґрунту (Capacitive Soil Moisture Sensor);
- цифровий давач температури ґрунту DS18B20;
- давач концентрації CO_2 SEN0159;
- давач освітленості OPT101;
- модуль годинника реального часу (RTC) DS3231SN;
- датчик потоку води FS400A;
- датчик ваги на модулі АЦП HX711 з тензодатчиком.

Для керування виконавчими механізмами обрано модулі реле з ізольованими контактами, що дозволяє підключати зовнішнє устаткування незважаючи на напругу їх живлення.

РОЗДІЛ 3

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВПРОВАДЖЕННЯ СИСТЕМИ КОМПЛЕКСНОЇ АВТОМАТИЗАЦІЇ ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ ВИРОЩУВАННЯ РОСЛИН У ТЕПЛИЦІ

3.1. Складання алгоритмів керування функціонуванням теплиці

Обрано для реалізації проекту АСКТ програмно-апаратна платформа Arduino дозволяє дуже легко та якісно розробляти автоматизовані системи керування. Це досягається не тільки великим вибором сумісних давачів, та механізмів, але ще і простою системою програмування, що абстрагує розробника від доволі складної реалізації задіяного у модулі мікроконтролера та периферії і пропонує інтегрувати програмний код розробника у доволі простий алгоритм програмного забезпечення що складається з двох послідовних функцій setup() та loop(). Тіло функцій надається для редагування розробником. Функція setup() дозволяє налаштувати задіяну периферію мікроконтролера та бібліотеки задіяних модулів що забезпечують їх функціонування. Функція loop() являє собою нескінчений цикл. У цьому циклі розробник розташовує функції, яки виконуються протягом всього часу роботи алгоритму. Блок-схема базового алгоритму платформи Arduino наведено на рисунку 3.1.

На рисунку 3.2 наведена блок-схема алгоритму налаштування периферійних модулів: портів вводу/виводу, що дозволяють обробляти як вхідні сигнали від давачів, так і керувати усіма виконавчими механізмами; модуля WiFi, що забезпечує зв'язок для передачі/прийняття параметрів теплиці для взаємодії з користувачем.

Також алгоритми містить налаштування бібліотеки використаних давачів: вологості та температури повітря DHT22; вологості та

температури ґрунту; концентрації CO_2; освітленості; годинника реального часу; потоку води.

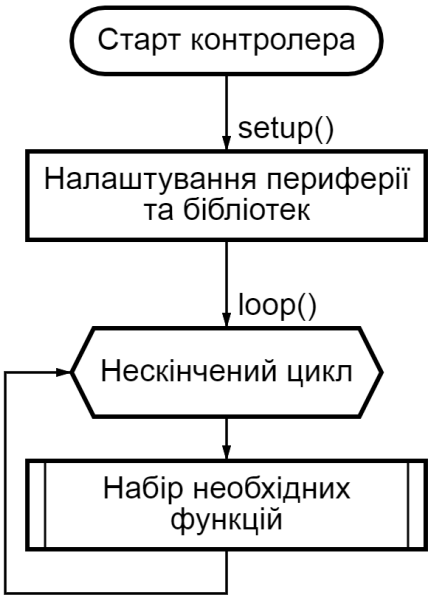

Рисунок 3.1 - Блок-схема базового алгоритму платформи Arduino

Ці модулі дозволяють слідкувати за станом мікроклімату теплиці та працездатністю виконавчих пристроїв.

На рисунку 3.3 наведено блок-схема алгоритму основного циклу програми що при необхідності зчитує параметри мікроклімату теплиці, виконує порівняння зчитаних значень параметрів з граничними з урахуванням задіяних виконавчих механізмів, обробляє запити користувач та формує відповідь (передає всі необхідні параметри у інтерфейс користувача).

Зчитування давачів та порівняння зчитаних значень параметрів з граничними з урахуванням задіяних виконавчих механізмів більш детально відображено у блок-схемі наведеної на рисунку 3.4.

Кожен отриманий параметр з давачів відображає стан мікроклімату теплиці. На основі цих параметрів оцінюються відхилення від встановлених меж і прийметься рішення на вплив за допомогою виконавчих механізмів.

Рисунок 3.2 - Блок-схема алгоритму налаштування периферійних модулів

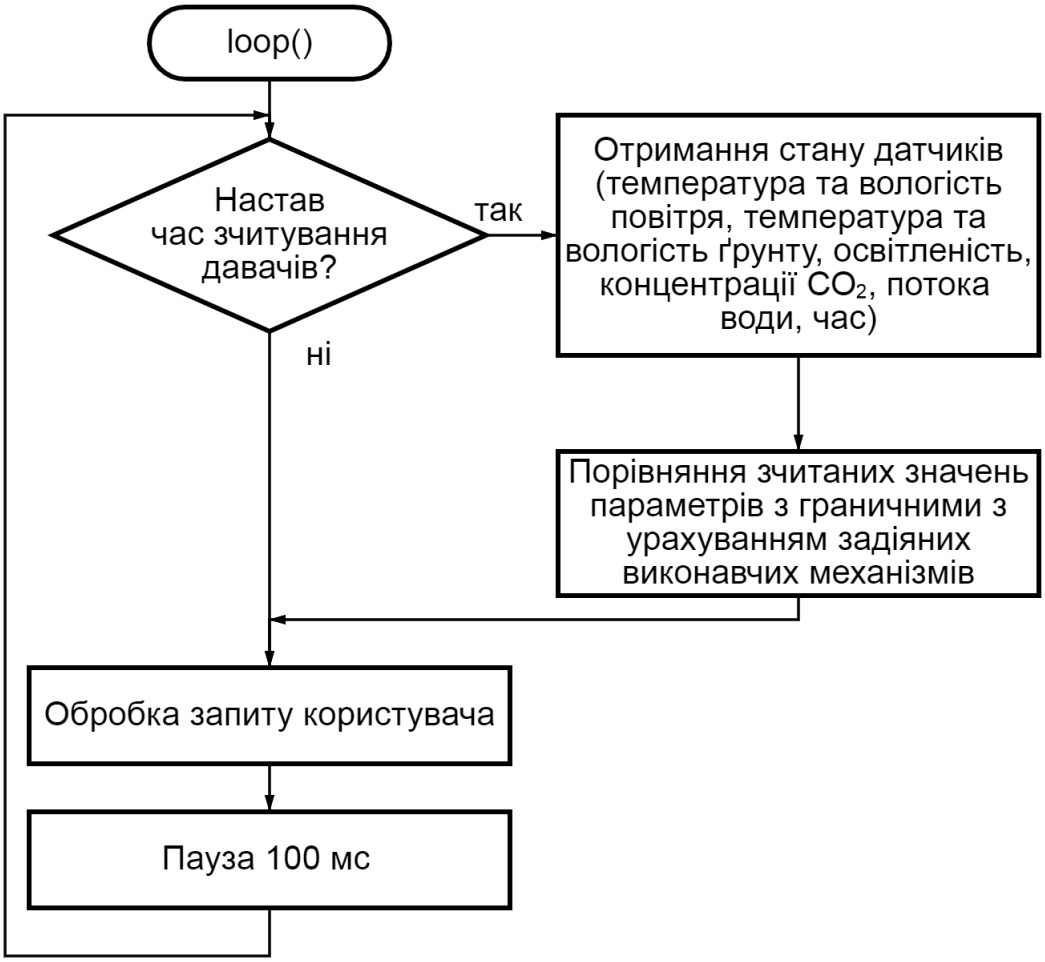

Рисунок 3.3 - Блок-схема алгоритму основного циклу (обробляються запити користувача та запускається зчитування параметрів мікроклімату теплиці)

Більш детально алгоритми для кожного з давачів відображені на рисунках 3.5-3.7.

На рисунку 3.5 відображено обробку давача температури та вологості. З початку перевіряються вихід температури за встановлені межі. Якщо температура нижче, то вмикається підігрів та контролюється підвищення температури. Якщо підвищення не відбувається встановлюється ознака помилки нагрівача і якщо температура буде нижчою за аварійну межу, включиться аварійна сирена. А якщо

температура вище встановленої межі, нагрівач вимикається, але якщо температура продовжить зростати (наприклад від сонячного тепла) буде увімкнено вентилятор провітрювання теплиці. Але якщо температура перевищить аварійну межу (наприклад у наслідок виникнення пожежі) буде увімкнено систему зволоження повітря та аварійну сирену.

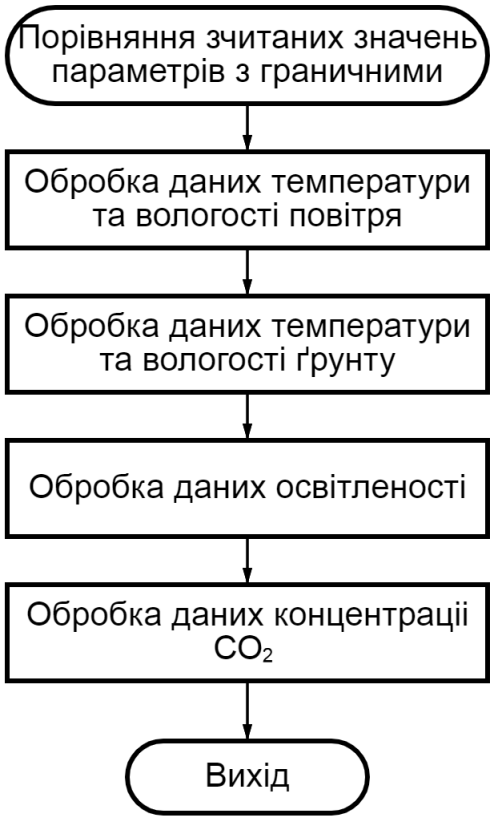

Рисунок 3.4 - Блок-схема алгоритму

Також контролюється вихід за встановлені межі вологості повітря. Якщо вологість нижча за встановлену межу, вмикається система зволоження повітря і контролюється зростання показника. Якщо значення показника не зростає, буде встановлено ознаку помилки, що буде відображено у інтерфейсі користувача. Якщо вологість досягла верхньої межі, система зволоження повітря буде вимкнена.

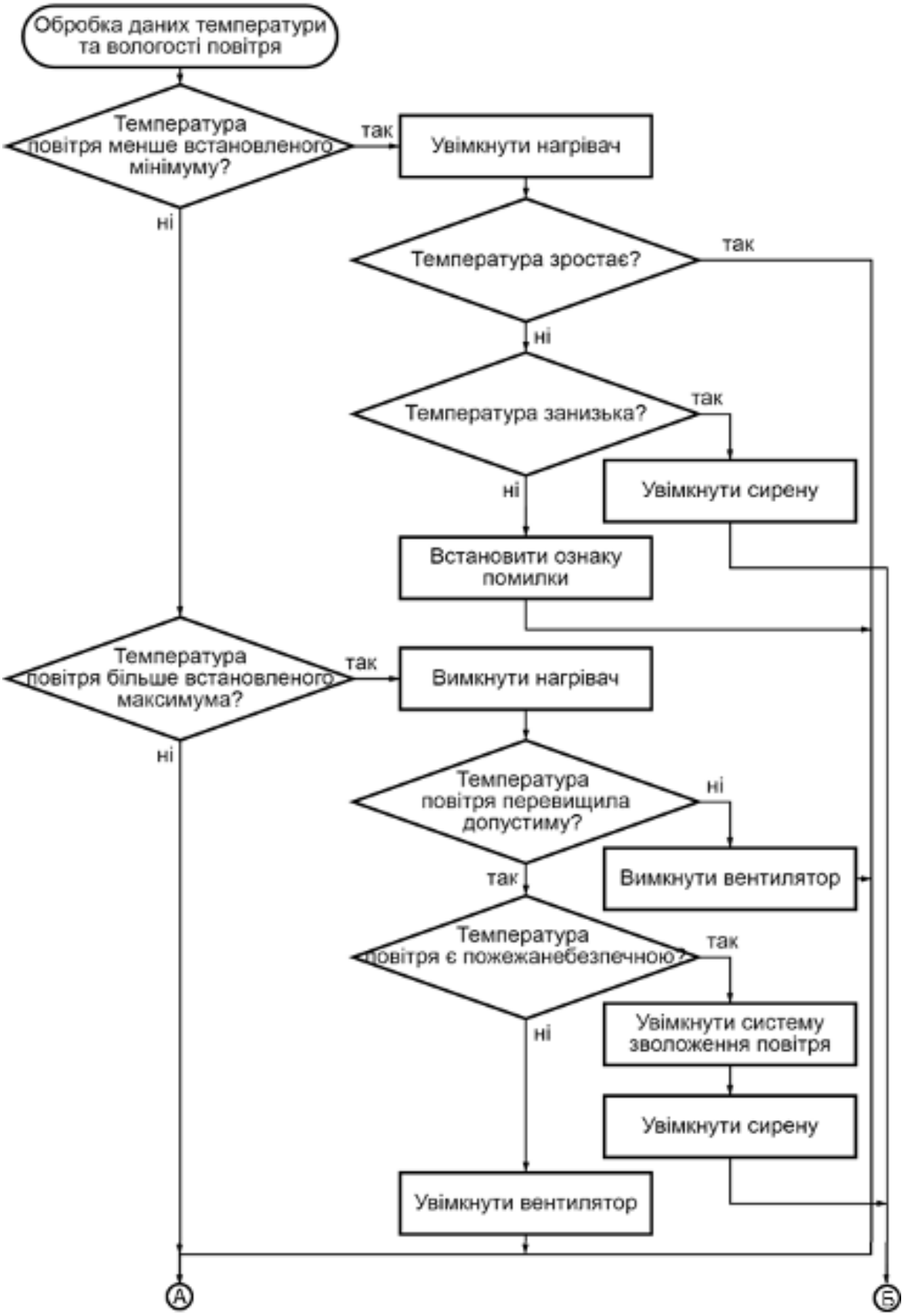

Рисунок 3.5.1 - Блок-схема алгоритму (початок) контролю температури та вологості

Рисунок 3.5.2 - Блок-схема алгоритму (кінець) контролю температури та вологості повітря в теплиці

На рисунку 3.6 відображено блок-схему алгоритму контролю температури та вологості ґрунту. Якщо температура ґрунту нижче встановленої межі, вмикається нагрівач і контролюється зростання температури. У випадку якщо температура не зростає, встановлюється ознака помилки, а якщо температура продовжить знижуватися і досягне аварійної межі, буде увімкнено аварійну сирену. Вологість ґрунту також контролюється в встановлених межах. Якщо вологість замала – вмикається насос поливу і контролюється збільшення вологості, якщо цього не відбувається, встановлюється ознака помилки. Якщо вологість досягла верхньої межі, насос поливу вимикається.

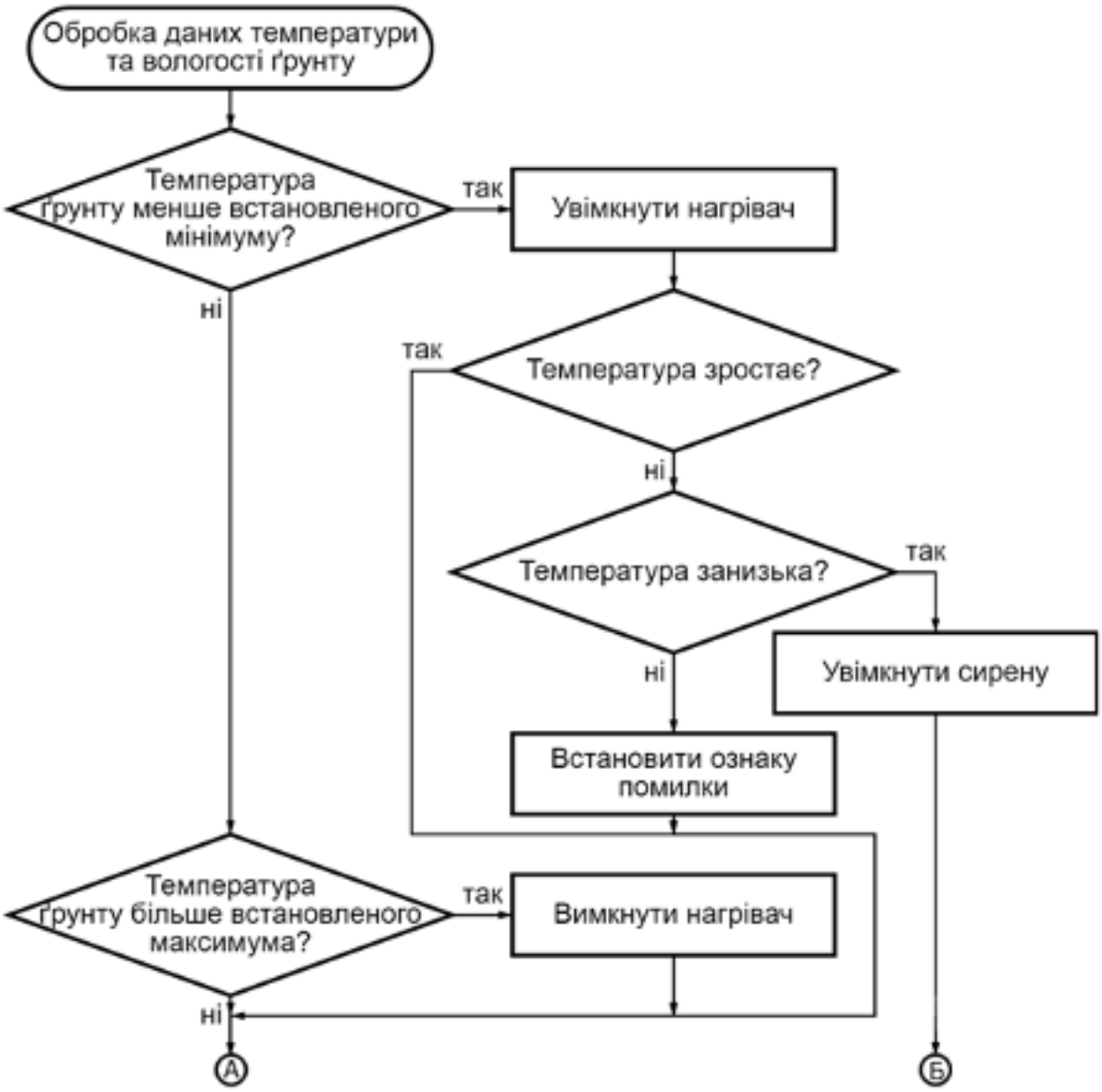

Рисунок 3.6.1 - Блок-схема алгоритму (початок) контролю температури та вологості ґрунту

На рисунку 3.7 наведено блок-схема алгоритму контролю освітленості.

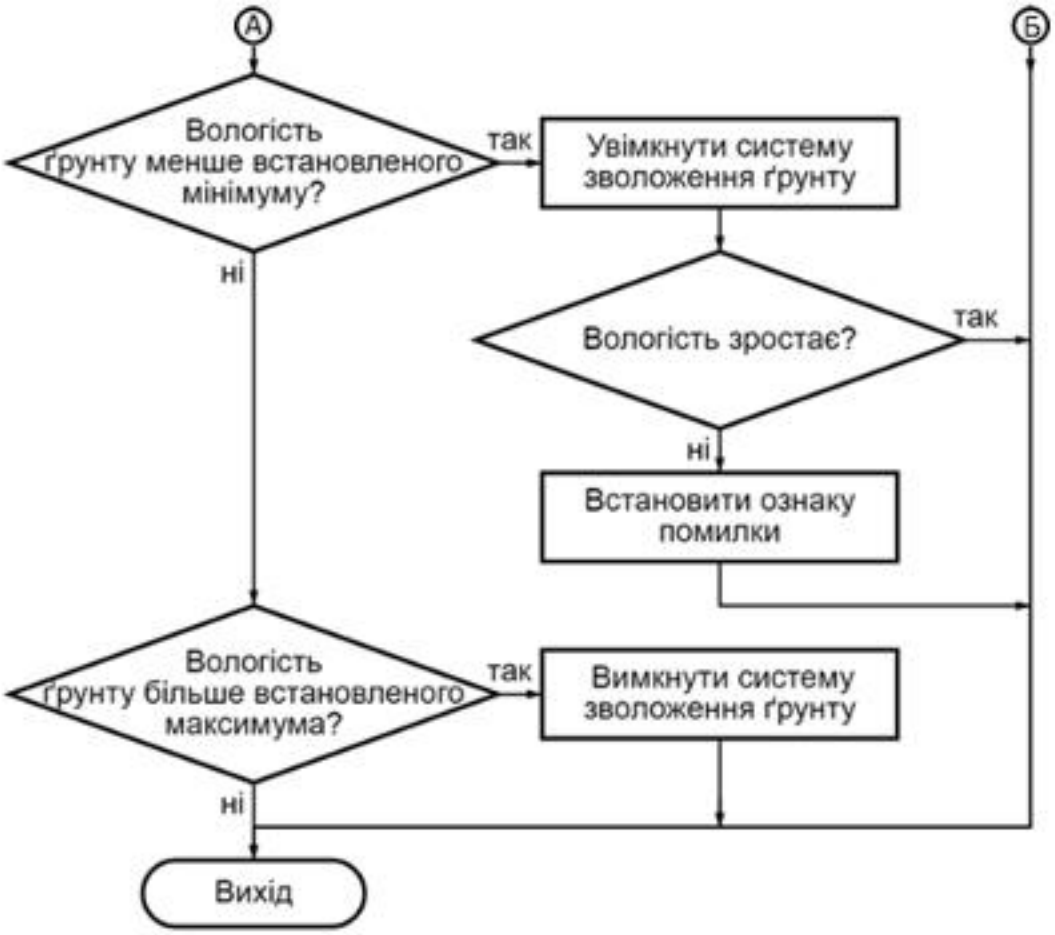

Рисунок 3.6.2 - Блок-схема алгоритму (кінець) контролю температури та вологості ґрунту

Впродовж дня накопичуються значення інтенсивності освітлення, та контролюється інтенсивність. Якщо в вечорі інтенсивність знижується нижче встановленої межі, а за світловий день не накопичено достатньо світла для рослин, буде увімкнено штучне освітлення, накопичення інтенсивності освітлення продовжиться і при досягненні встановленої межі, буде вимкнено. Якщо після увімкнення штучного освітлення не буде спостерігатися зростання накопиченої інтенсивності освітлення, буде встановлено ознаку помилки.

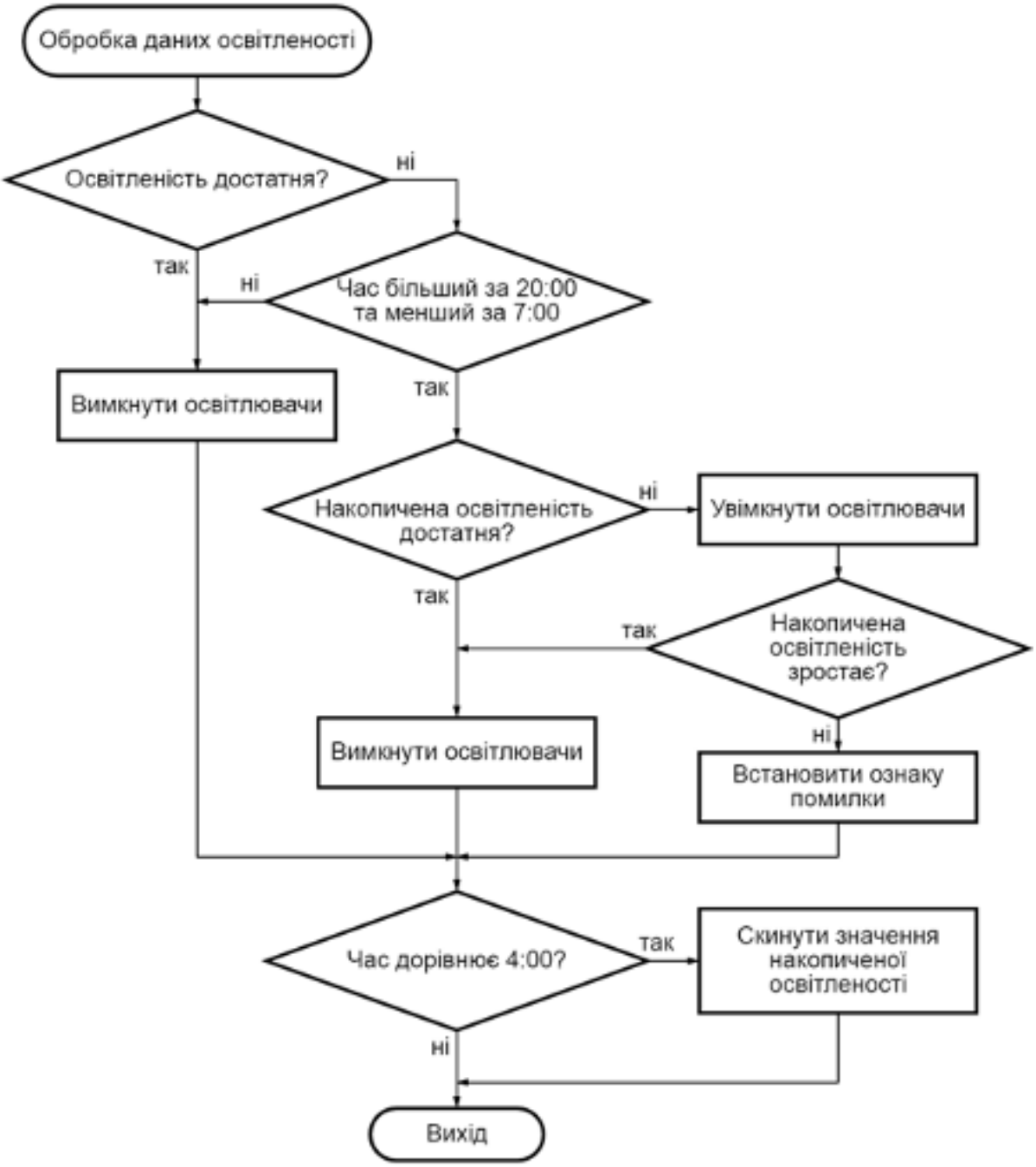

Рисунок 3.7 - Блок-схема алгоритму контролю освітленості

Концентрація CO_2 у теплиці утримується у встановлених межах завдяки алгоритму відображеному на рисунку 3.7. Якщо концентрація CO_2 занизька, то вмикається клапан подачі CO_2 у теплиці. Та якщо

концентрація не буде зростати встановлюється ознака помилки. А якщо концентрація CO_2 досягла верхньої межі, клапан подачі вимикається.

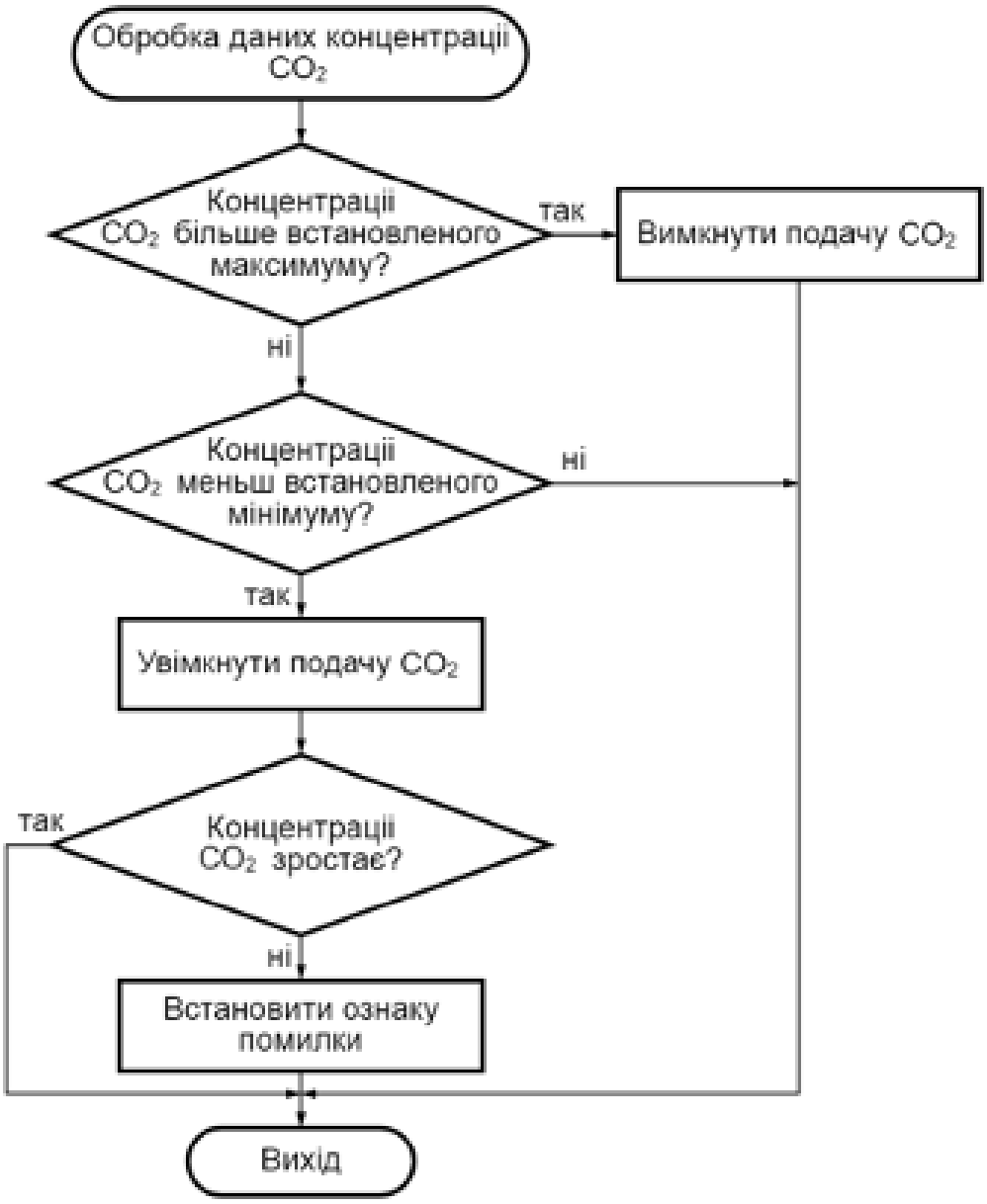

Рисунок 3.8 - Блок-схема алгоритму керування концентрації CO_2 у теплиці

Для роботи контролера теплиці було розроблено алгоритм підтримки інтерфейсу користувача (див. рисунок 3.9). Алгоритм повністю забезпечує користувача всією необхідною інформацією то дозволяє

змінювати всі необхідні параметри роботи теплиці за допомогою бездротової мережі WiFi:

- передача показників теплиці (реальний час, температура, вологість повітря та ґрунту, освітленість, концентрацію CO_2, розходу води, маси шпалер);

- прийняття параметрів налаштування, які задають: час, мінімальну та максимальну температуру повітря, вологість повітря та ґрунту, необхідну освітленість та час освітлення, концентрацію CO_2.

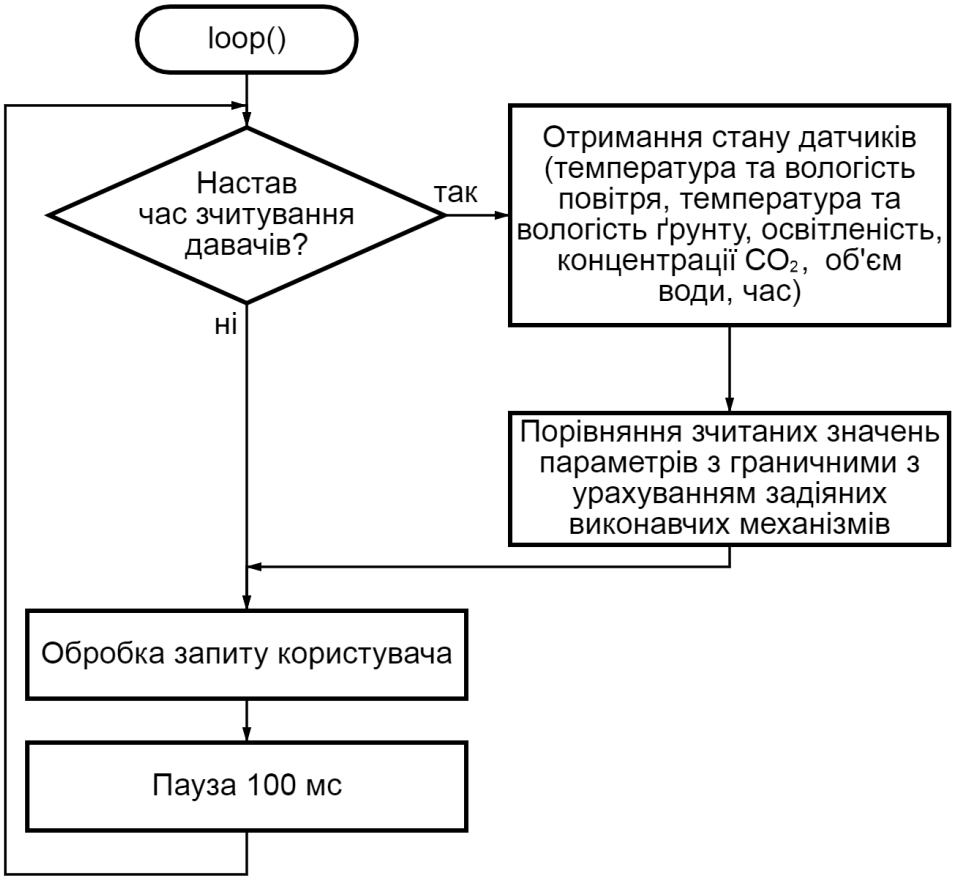

Рисунок 3.9 - Блок-схема алгоритму взаємодії з користувачем через мережу WiFi

Обмін даними з інтерфейсом користувача виконується за допомогою сталого формату JSON файлу.

Лістинг програми наведено у додатку 1.

Також було розроблено додаток для моніторингу, та налаштування параметрів теплиці.

Ця програма працює на пристроях Android. Мінімальна версія системи 4.1. Остання версія, на якій тестували додаток, — Android 11, що відповідно до статистики Google, охоплює близько 99,8% активних пристроїв з операційною системою, тобто пристроїв, випущених після 2012-2013 років.

Додаток для Android платформи, універсальна версія системи 4.1. Остання версія, на якій тестували додатки, є Android 11. Згідно з офіційною інформацією від компанії Google, покриває приблизно 99,8% активних пристроїв, які використовують операційну систему Android. Це включає пристрої, які були випущені з 2012-2013 року й надалі. Під час проектування системи складено діаграма послідовностей (рис. 3.2). Послідовність операцій, які відбуваються під час використання системи відображено на діаграмі (рис. 3.3-3.9).

3.2. Вибір програмного середовища розробки

Flutter - це потужний кросплатформний фреймворк для створення інтерфейсів користувача, розроблений компанією Google. Він дозволяє створювати елегантні та високоінтерактивні додатки для мобільних пристроїв, вебу та настільних систем за допомогою одного спільного коду.

Головною перевагою Flutter є можливість використовувати один і той самий код для різних платформ. Однак, оскільки кожна операційна система має свої особливості, деякі частини програми можуть потребувати адаптації для конкретної платформи, наприклад, для iOS. Водночас більша частина коду залишається спільною, що дозволяє значно зменшити час та витрати на розробку додатків для всіх підтримуваних платформ.

Мова програмування Dart є основним інструментом для розробки у Flutter.

Flutter забезпечує підтримку для багатьох цільових платформ, таких як Android, iOS, веб і настільні операційні системи, включаючи Windows, macOS та Linux. Завдяки AOT-компіляції на мові Dart, програми, розроблені на Flutter, працюють швидко та ефективно, наближаючись до продуктивності нативних додатків.

IntelliJ IDEA — це інтегроване середовище розробки (IDE), створене компанією JetBrains для розробки програмного забезпечення на різних мовах, таких як Java, Kotlin, Python, JavaScript, PHP та інших. Завдяки своїм потужним функціям та інтуїтивно зрозумілому інтерфейсу, IntelliJ IDEA є однією з найпопулярніших IDE серед розробників.

Версія IntelliJ IDEA 6.0 надала інтегроване рішення для створення графічних інтерфейсів користувача. Середовище також має гарну інтеграцію з такими популярними віртуальними інтерфейсами для розробників, як CVS, Ubuntu, ApacheAnt та Maven. У 2007 році IntelliJ анонсувала підтримку плагінів для програмування на Ruby.

З версії 9.0 IntelliJ IDEA доступна в двох варіантах: Community Edition та Ultimate Edition. Community Edition є безкоштовною і підтримує Java SE, Kotlin, Groovy та Scala, а також інтегрується з популярними системами контролю версій. Ultimate Edition, що доступна за комерційну ліцензію, додатково підтримує Java EE, UML-діаграми, аналіз покриття коду та інші фреймворки та системи контролю версій.

Android Studio — це потужне середовище розробки, яке надає розробникам усі необхідні інструменти для створення високоякісних додатків для Android. Від написання коду до налагодження та тестування, Android Studio забезпечує широкий спектр функцій, що значно полегшують процес розробки. Однак через велику кількість доступних

опцій навігація та досягнення максимальної ефективності можуть бути складними завданнями.

Середовище розробки призначене для вирішення ключових завдань, що виникають при створенні Android-додатків. Серед його функцій — інструменти для прискорення тестування програм, сумісних з різними версіями платформи, а також інструменти для створення додатків, що підтримують різноманітні дисплеї пристроїв.

Для швидшого створення програм Android Studio надає набір стандартних елементів інтерфейсу та візуальний редактор, який дозволяє попередньо переглядати різні стани інтерфейсу програми. Крім того, є майстер для створення кастомних елементів дизайну, який підтримує використання шаблонів для створення власних інтерфейсів. Платформа також має вбудовані можливості для завантаження зразків коду з GitHub.

Android Studio включає потужні інструменти для рефакторингу коду. Розробники можуть використовувати їх для полегшення змін у структурі коду, перейменування змінних, методів або класів, а також для витягування повторюваних частин коду в окремі функції. Це покращує читабельність коду, знижує його дублювання та підвищує підтримуваність.Представимо системні вимоги Android Studio у вигляді табл. 3.1.

Таблиця 3.1. Системні вимоги Android Studio

Windows	OS X/macOS	Linux	Chrome OS
Версія OS	Microsoft Windows 10/8/7 (32- or 64-bit)	Mac OS X 10.10 або вище, аж до 10.11.6 (ElCapitan) чи 10.14 (Mojave)	графічне середовище GNOME чи KDE
RAM	4 GB RAM мінімум, 8 GB RAM рекомендовано		
Diskspace	Дисковий простір 500 МБ для AndroidStudio, принаймні 4		

	ГБ для SDK, емульовані образи системи та кеш-пам’ять
Версія Java	JavaDevelopmentKit (JDK) 8
Роздільність дисплею	1280x800 мінімум

Xcode — це потужне та багатофункціональне середовище розробки (IDE), яке надає розробникам всі необхідні інструменти для створення програмного забезпечення для платформ Apple. Воно забезпечує зручну та продуктивну роботу, охоплюючи весь процес — від написання коду до складання готових проектів.

Xcode (IDE) — це абревіатура від Integrated Development Environment, що в перекладі означає "інтегроване середовище розробки". Такі середовища включають всі інструменти, необхідні для написання коду і складання проекту. Xcode розроблено спеціально для macOS, але проекти, створені в цьому середовищі, можна також запускати на iOS, tvOS та watchOS. Воно підтримує різноманітні мови програмування, такі як Swift, Objective-C, C, C++, а також AppleScript, Python, Ruby та Java. Крім того, сторонні розробники додали підтримку таких мов, як Haskell, Pascal, Ada та інших.

3.3. Розробка керуючої програми

На рис. 3.10 продемонстровано макет розробленого мобільного додатку.

Ознайомившись із макетом, переходимо до створення проєкту. У середовищі розробки IntelliJ IDEA створимо новий проєкт, підключивши Android SDK та Flutter SDK (рис. 3.11).

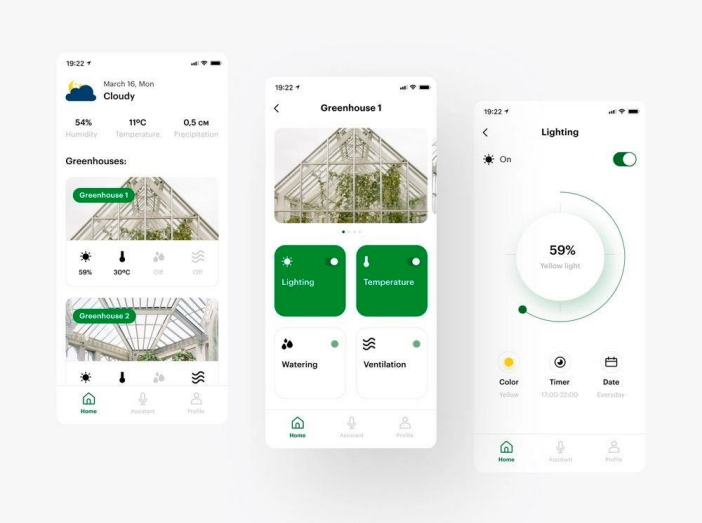

Рисунок 3.10 – Макет додатку для АСКТ

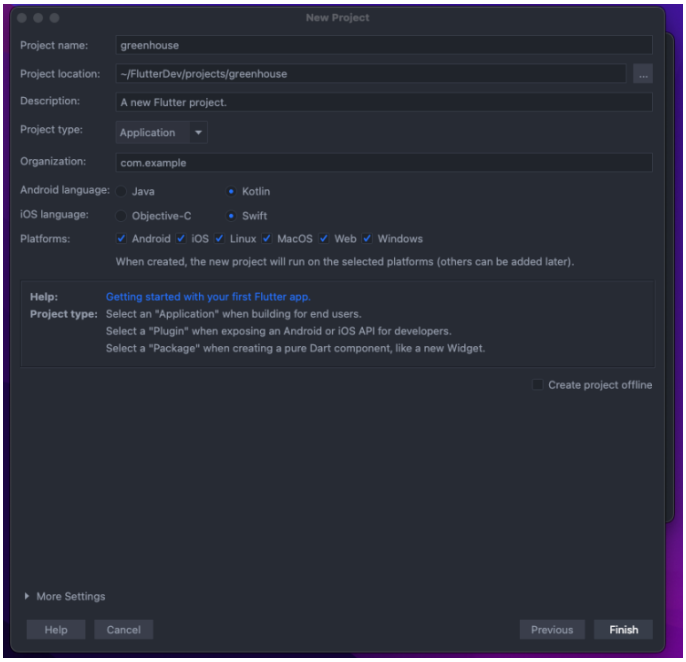

Рисунок 3.11 – Створення проекту

Спершу створимо папку та розмістимо в ній усі SVG та PNG зображення, які плануємо використовувати для верстки додатка (рис. 3.12).

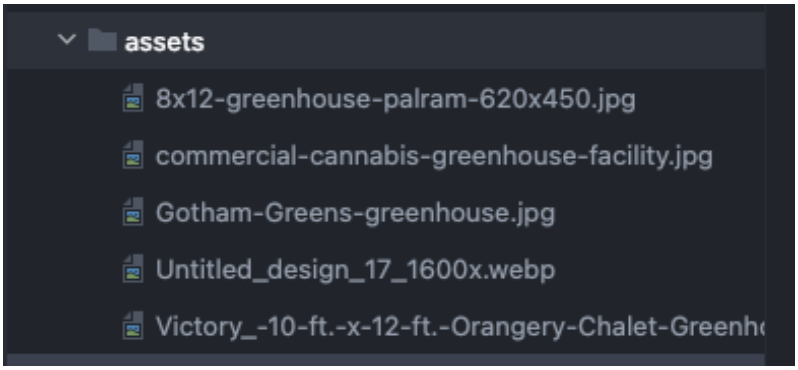

Рисунок 3.12 – Папка Assets

Щоб додати зображення до проекту, зазначимо шлях до папки з ними у файлі pubspec.yaml. Після аналізу функціоналу додатку внесемо до цього ж файлу сторонні бібліотеки, які будемо використовувати у проекті. Далі, для завантаження необхідних бібліотек, виконаємо команду “flutter pub get” у терміналі (рис. 3.13).

Рисунок 3.13 – Завантаження бібліотек

Keystore — це захищене сховище конфіденційної інформації, яке використовується для шифрування даних, аутентифікації та встановлення

HTTPS-з’єднання. Наприклад, для забезпечення SSL-з’єднання між клієнтом і сервером потрібні приватний ключ та сертифікат.

У разі односторонньої автентифікації Keystore застосовується лише на стороні сервера. Для двосторонньої автентифікації здійснюється обмін сертифікатами, тобто обидві сторони мають приватні ключі. Простими словами, Keystore використовується для швидкої ідентифікації власника ключа.

Java підтримує кілька форматів зберігання даних у Keystore:

– jks: Тип зберігання за замовчуванням, який найчастіше використовується;
– jceks: Альтернатива jks з покращеним шифруванням на основі Triple DES. Якщо спочатку використовувався jks, сховище можна оновити до jceks за допомогою утиліти keytool.
– pkcs12: Формат зберігання, який підходить для транспортування закритих ключів.

Для створення ключа, який буде використовуватися для підписання додатка, виконаємо команду - "keytool -genkey -v -keystore D:\webs\key.jks -storetype JKS -keyalg RSA -keysize 2048 -validity 10000 -alias key". У процесі виконання команди необхідно буде вказати пароль та шлях до збереження ключа (рис. 3.14).

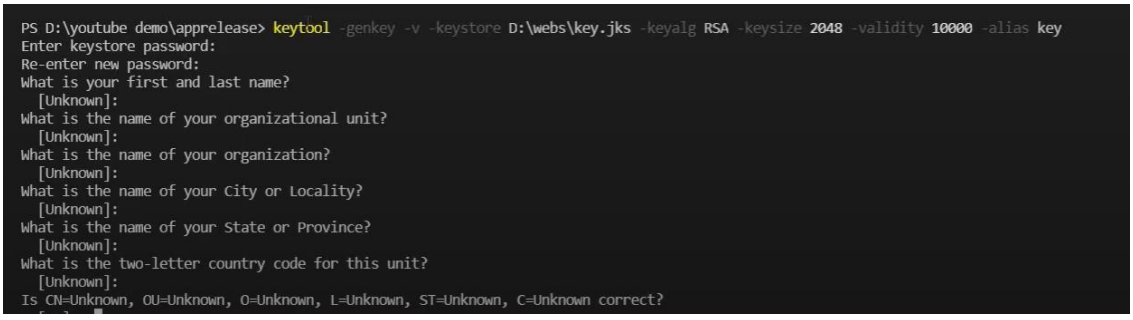

Рисунок 3.14 – Створення ключа

У папці Android створимо файл key.properties і додамо в нього дані ключа. Після цього, у файлі build.gradle налаштуємо підпис додатка за допомогою створеного ключа та виконаємо такі зміни, встановимо мінімальну та максимальну версії Android SDK, визначимо версію додатка та версію білда, додамо плагіни, які будуть використовуватись в майбутньому.

Наступним кроком відредагуємо файл AndroidManifest.xml, де змінемо ім’я та іконку додатка, а також додамо всі необхідні дозволи для користувача, такі як доступ до геолокації та інші (рис. 3.15–3.16).

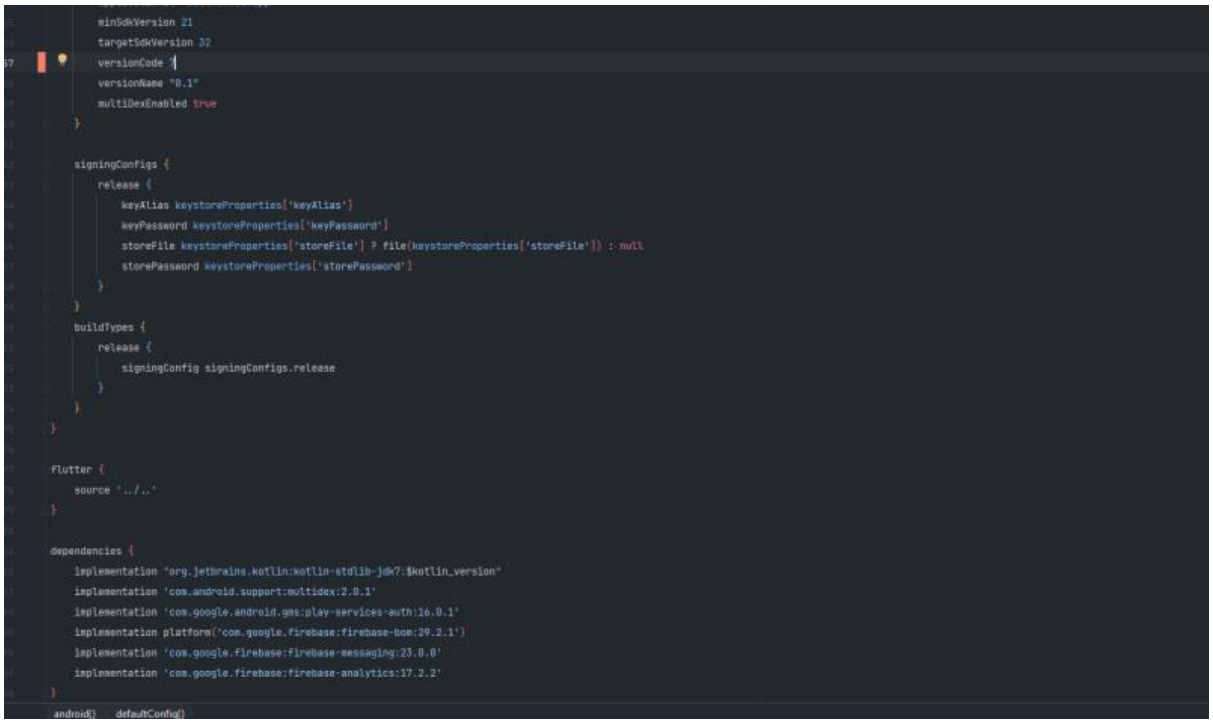

Рисунок 3.15 – Файл build.gradle

AndroidManifest.xml — це ключовий файл конфігурації для додатків Android. Він містить важливу інформацію про додаток і визначає його основні характеристики, зокрема компоненти додатка, такі як активності, служби, приймачі та постачальники контенту. Цей файл знаходиться в кореневій папці проекту Android і є обов'язковим для кожного додатка

Android. Система Android використовує AndroidManifest.xml для ідентифікації додатка та його налаштування під час встановлення та запуску. Зробимо там стандартні налаштування для конфігурацію додатку, виставимо назву додатку і проекту, та додамо запрос на дозвіл певних функцій, дозвіл на знаходження місцезнаходження, доступу в інтернет, часу телефона та збереження інформації на телефон.

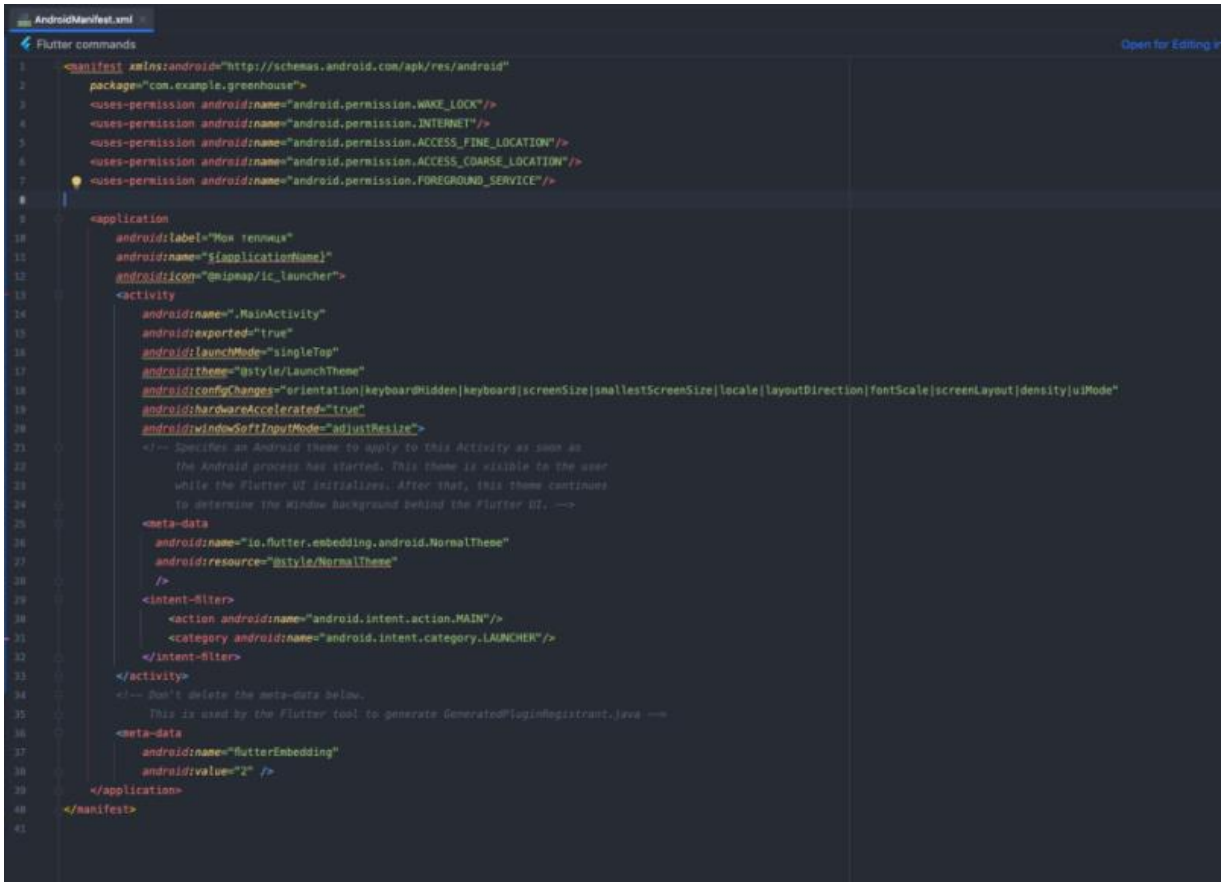

Рисунок 3.16 – Файл Android Manifest.xml

Використовуючи макет, створимо головний екран, на якому відобразимо всі дані, отримані з HTTP запитів до контролера Arduino Nano 33 IoT, та покажемо теплиці за допомогою стандартного StatefulWidget (рис. 3.17). StatefulWidget є одним з основних класів у Flutter, фреймворку для розробки мобільних та веб-додатків. Він призначений для створення віджетів, які можуть змінювати свій стан (state) під час виконання

програми. Коли використовують StatefulWidget, враховуються два основні об'єкти, StatefulWidget визначає конфігурацію віджета, State відповідає за зберігання та зміну стану цього віджета. згідно з цим принципом, ми розробимо наступні сторінки відповідно до макету.

```
class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  @override
  void initState() {
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    double height = MediaQuery.of(context).size.height;
    double width = MediaQuery.of(context).size.width;
    double topSafeArea = MediaQuery.of(context).padding.top;
    double bottomSafeArea = MediaQuery.of(context).padding.bottom;
    return Scaffold(
      backgroundColor: const Color(0xFFFAFAFA),
      body: SingleChildScrollView(
        child: Center(
          child: Padding(
            padding: const EdgeInsets.symmetric(horizontal: 20.0),
            child: Column(
              children: [
                const SizedBox(
                  height: 15,
                ), // SizedBox
                Row(
                  mainAxisAlignment: MainAxisAlignment.spaceBetween,
                  children: [
                    Row(
                      children: [
                        CachedNetworkImage(
                          imageUrl:
                              BlocProvider.of<UserBloc>(context).state.image,
                        ), // CachedNetworkImage
                        const SizedBox(
                          width: 15,
                        ), // SizedBox
                        Column(
```

Рисунок 3.17 – Частина коду головного екрану додатку

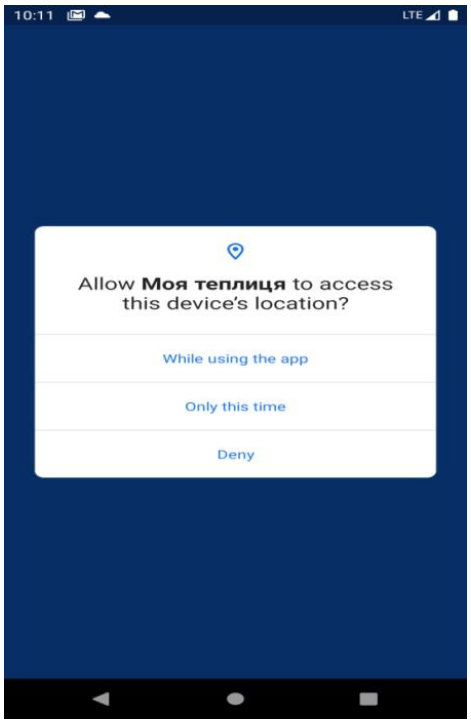

Рисунок 3.18 – Запит на передачу геолокаційних даних для отримання детальної інформації про погоду та місце знаходження

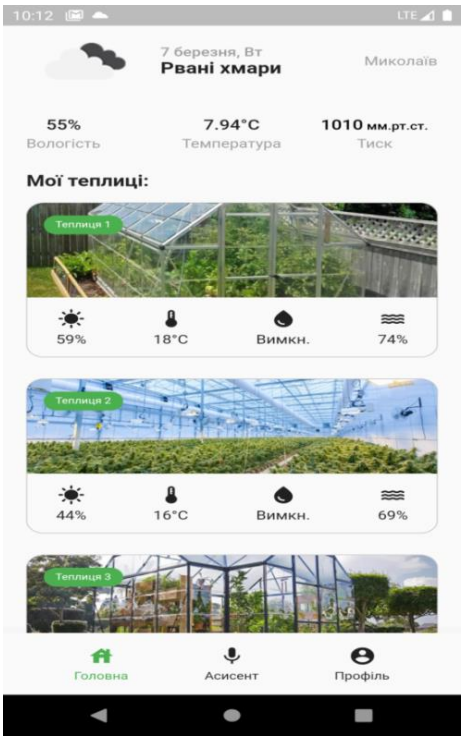

Рисунок 3.19 – Головний екран додатку

Використовуючи безкоштовне API OpenWeatherMap та BigDataCloud, реалізуємо систему для отримання даних про погоду та назву міста користувача. Спочатку отримуємо дозвіл на доступ до геолокації (довгота та широта), після чого відправляємо HTTP запити та отримуємо JSON з інформацією про погоду та місто. Ці дані відображаємо на головному екрані додатку (рис. 3.18-3.20).

```dart
Future<dynamic> getWeather(lat, lng) async {
  const String apiKey = 'bd5e378503939ddaee76f12ad7a97608';
  final String url = 'https://api.openweathermap.org/data/2.5/weather';
  var response = await Dio().get(
    url,
    queryParameters: {
      'lat': lat,
      'lon': lng,
      'appid': apiKey,
      'units': 'metric',
      'lang': 'ua',
    },
  );
  return response;
}

Future<dynamic> getCity(lat, lng) async {
  var response = await Dio().get(
    "https://api.bigdatacloud.net/data/reverse-geocode-client?latitude=$lat&longitude=$lng&localityLanguage=uk",
  );
  return response;
}
```

Рисунок 3.20 – Функції get запитів на отримання погоди і міста користувача

Створимо екран керування теплицею, на якому розробимо інтерфейс для повного налаштування. Для цього створимо віджети, які дозволяють передавати дані на сервер та запускати апаратну частину теплиці. Користувач зможе включати та вимикати освітлення, змінювати температуру і яскравість світла, контролювати та підтримувати температуру повітря та ґрунту, а також підтримувати вологість повітря та ґрунту. Крім того, інтерфейс дозволить включати полив, провітрювання та

насичувати приміщення CO_2. Все це буде реалізовано через зручний інтерфейс, що дозволяє налаштовувати параметри теплиці через додаток (рис. 3.21-3.22).

```dart
class GreenHousePage extends StatefulWidget {
  const GreenHousePage({
    Key? key,
    required this.title,
  }) : super(key: key);
  final String title;

  @override
  State<GreenHousePage> createState() => _GreenHousePageState();
}

class _GreenHousePageState extends State<GreenHousePage> {
  @override
  void initState() {
    super.initState();
  }

  final _controller = PageController();

  bool lighting = false;
  bool thermostat = false;
  bool watering = false;
  bool humidity = false;
  bool wind = false;
  int temp = 18;
  int vol = 55;

  @override
  Widget build(BuildContext context) {
    double height = MediaQuery.of(context).size.height;
    double width = MediaQuery.of(context).size.width;
    double topSafeArea = MediaQuery.of(context).padding.top;
    double bottomSafeArea = MediaQuery.of(context).padding.bottom;
    return Scaffold(
      appBar: AppBar(
        elevation: 0,
        backgroundColor: Colors.transparent,
        centerTitle: true,
        leading: IconButton(
          icon: const Icon(
            Icons.arrow_back_ios_rounded,
```

Рисунок 3.21 – Частина коду екрану програми яка відповідає за головну сторінку додатка

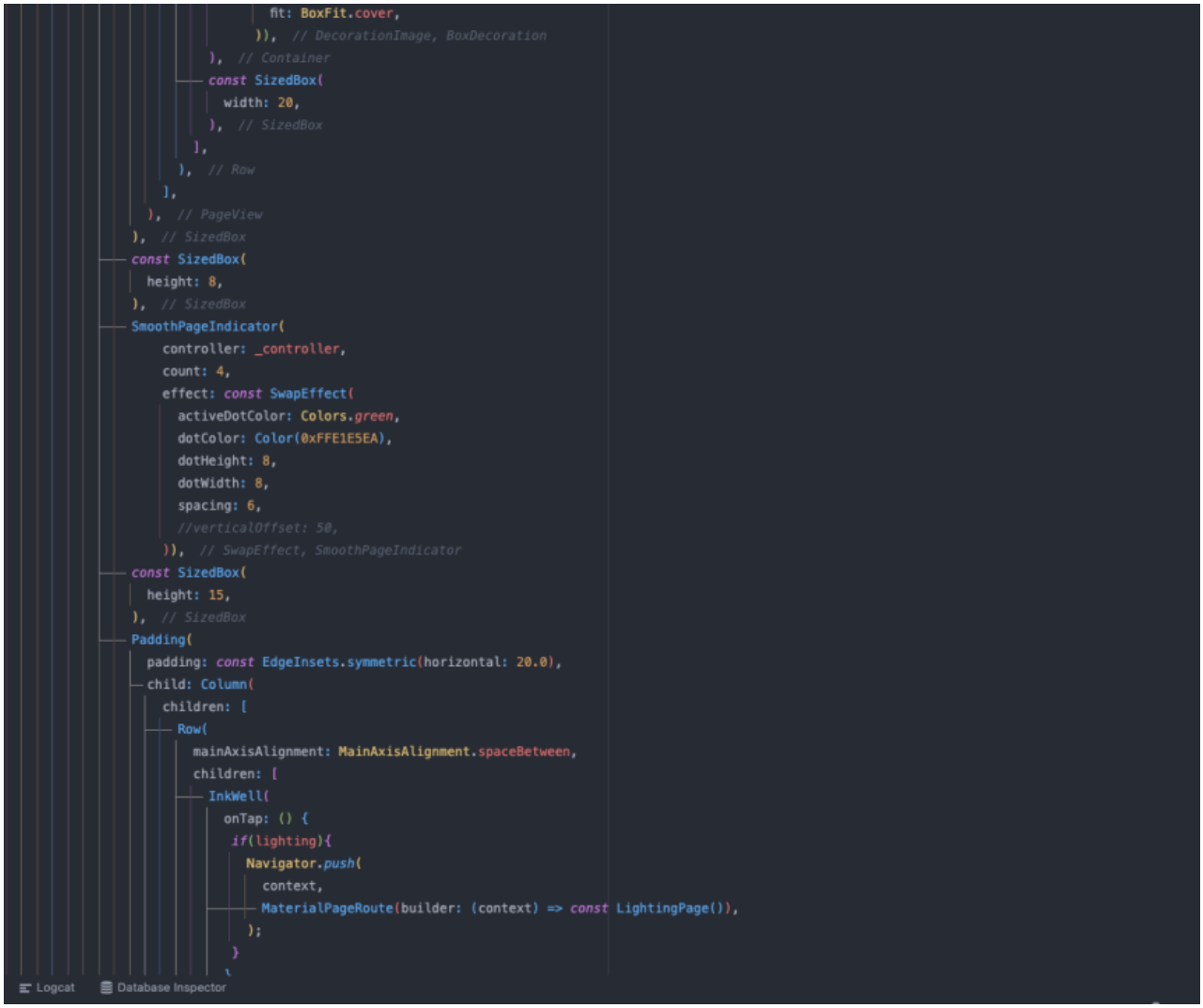

Рисунок 3.22 – Частина коду екрану програми яка відповідає за сторінку управління теплицею

Також було розроблено код для відображення в теплиці статусу помилок та аварій. Ця інформація приходить з Arduino контролеру і буде виводитись на екрані управління теплицею, де користувач зможе побачити актуальні повідомлення про можливі несправності або неполадки в системах теплиці. Таким чином, додаток не тільки дозволить контролювати параметри теплиці, але й оперативно інформуватиме користувача про будь-які проблеми, що виникають, що сприятиме швидкому реагуванню і підтримці належного стану теплиці (рис. 3.23-3.24).

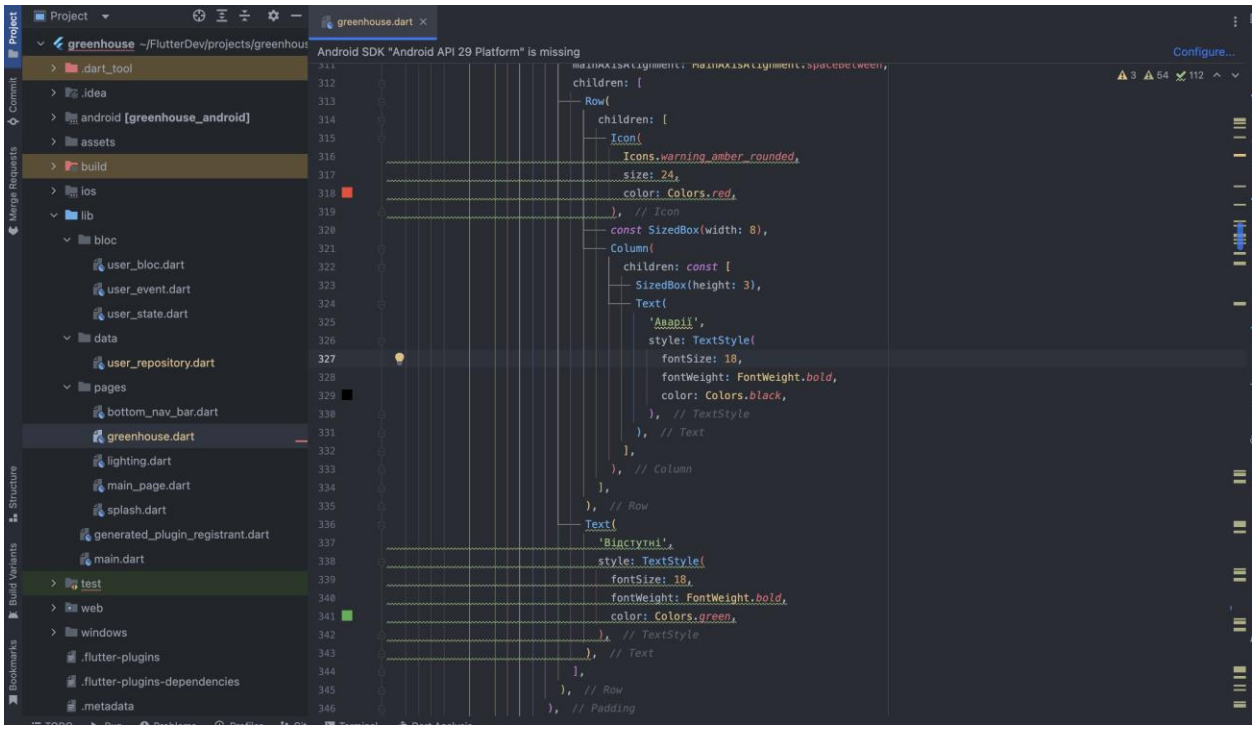

Рисунок 3.23– Частина коду екрану програми яка відповідає за сповіщення про помилки та аварію

Рисунок 3.24 – Екран додатку з відображенням інформації про помилки та аварії

Для реалізації системи моніторингу за вагою рослин та кислотністю ґрунту було розроблено код, який отримує від Arduino контролера дані

про загальну масу в грамах, виміряну з усіх тензодатчиків, а також інформацію з датчиків для вимірювання pH кислотності та кислотності ґрунту. Ці дані надходять і зберігаються при кожному новому відкритті додатку, раз на день. Після цього користувач отримує графік, який відображає 7-денну історію цих показників, що дозволяє відслідковувати прогрес або регрес у змінах маси, кислотності ґрунту та рівня pH. Цей підхід дає можливість користувачеві своєчасно коригувати умови в теплиці, покращуючи умови для росту рослин (рис. 3.25-3.27).

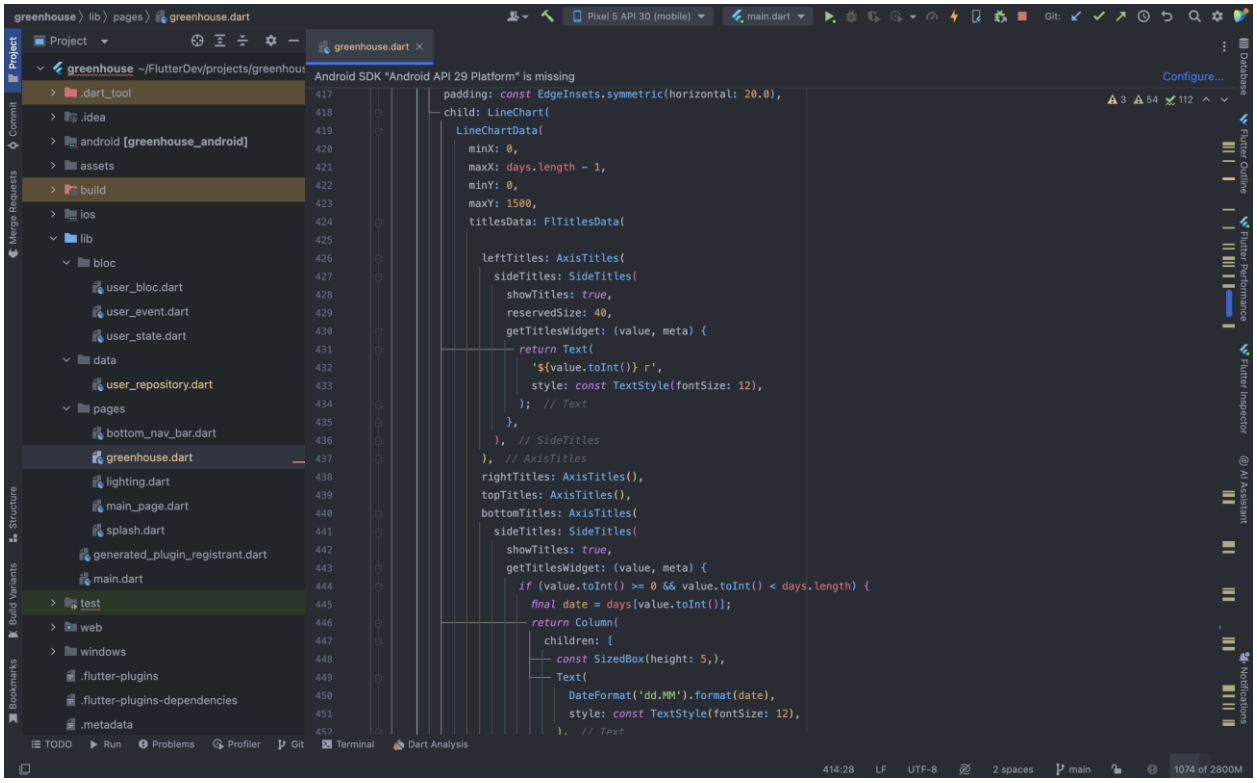

Рисунок 3.25 – Частина коду екрану програми яка відповідає за систему моніторингу

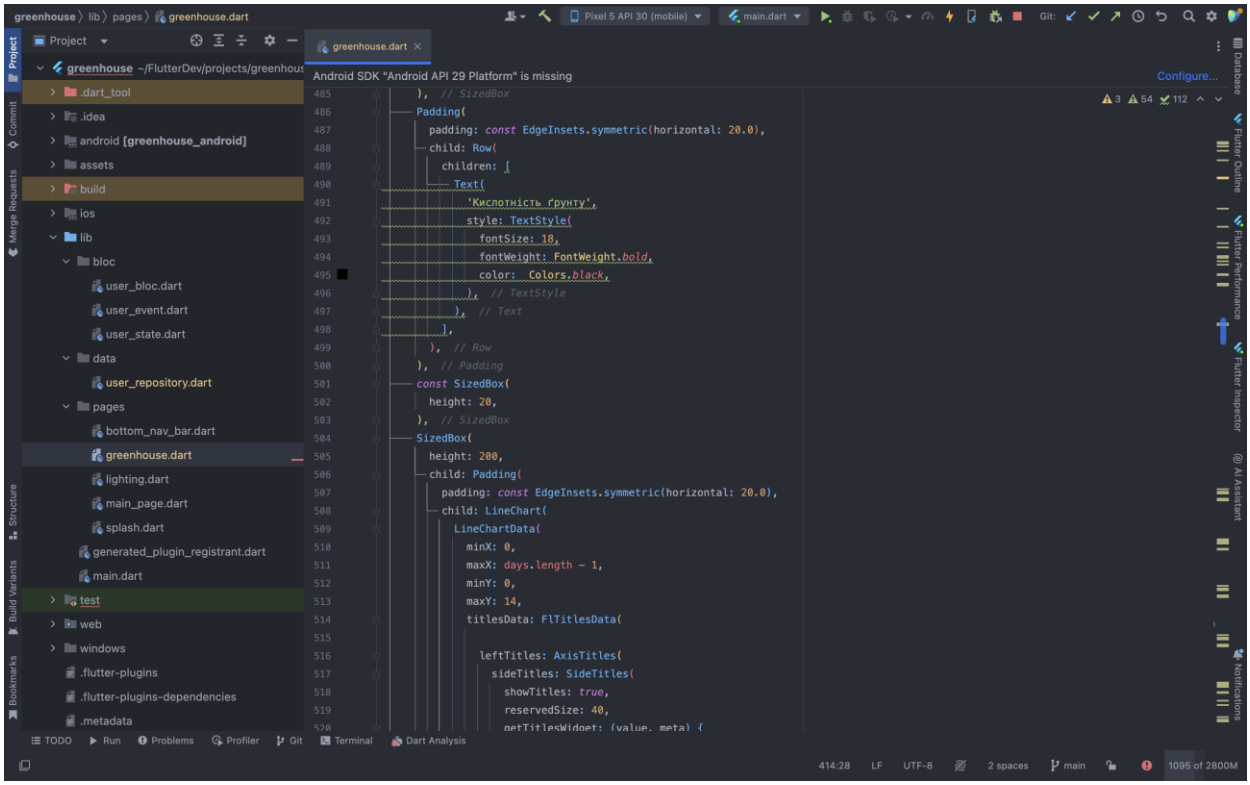

Рисунок 3.26 – Частина коду екрану програми яка відповідає за систему моніторингу

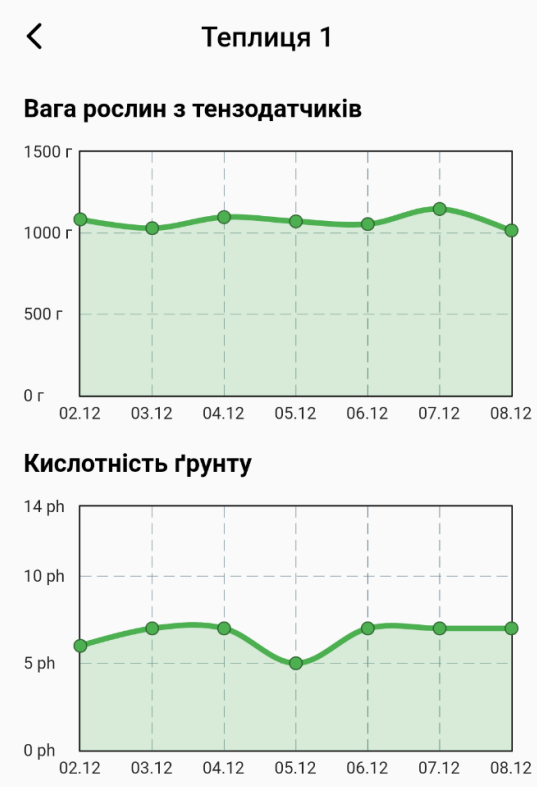

Рисунок 3.27 – Екран додатку з системою моніторингу

Висновки за розділом 3

Підсумовуючи третій розділ, можна зробити наступні висновки:

1. Розроблено алгоритм для управління роботою теплиці.
2. Створено програму для початкового налаштування контролера.

2. Вибрано програмне середовище для розробки системи.

3. Реалізовано програму управління.

РОЗДІЛ 4

ВПРОВАДЖЕННЯ ЗАСОБІВ ШТУЧНОГО ІНТЕЛЕКТУ В СИСТЕМУ КОМПЛЕКСНОЇ АВТОМАТИЗАЦІЇ ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ ВИРОЩУВАННЯ РОСЛИН У ТЕПЛИЦІ

4.1. Інтеграція штучного інтелекту в мобільний додаток

В додатку також була розроблена кнопка, яка дозволяє користувачеві вибирати тип рослини, яку він вирощує в теплиці. Після того, як користувач обирає тип рослини, додається можливість отримати рекомендації та аналіз від штучного інтелекту, що допомагає оптимізувати умови для вирощування.

Для цього був розроблений код, що передає на сервер інформацію про стан теплиці, включаючи параметри, такі як температура повітря, вологість, температура і вологість ґрунту, кислотність ґрунту, вага рослин із тензодатчиків, а також інші параметри (освітлення, вентиляція, рівень CO2, тип ґрунту, автоматичний полив і тип вирощуваної рослини). Штучний інтелект обробляє ці дані і надає рекомендації щодо оптимальних умов для вирощування, з урахуванням прогресу чи регресу в умовах теплиці.

Цей процес дозволяє користувачеві отримувати інтелектуальні рекомендації, що можуть допомогти в налаштуванні і поліпшенні умов для росту рослин, а також забезпечує більш ефективне управління теплицею.

Для цієї операції було вирішено використати штучний інтелект Gemini, він надає безкоштовне API і має доволі простий та гнучкий спосіб реалізації. Gemini – це високорозвинена мовна модель, створена компанією Google AI. Вона є представником сімейства моделей Gemini,

які відзначаються значною потужністю та здатністю виконувати широкий спектр мовних завдань.

Точна кількість шарів у моделі Gemini є комерційною таємницею. Однак відомо, що це надзвичайно глибока нейронна мережа, що складається з багатьох шарів, кожен з яких відповідає за обробку інформації на своєму рівні. Кількість шарів безпосередньо впливає на здатність моделі розуміти контекст, генерувати креативний контент та виконувати складні логічні міркування.

Модель Gemini навчалася за допомогою величезних обсягів текстових даних, що охоплюють різноманітні теми та стилі. Процес навчання включає:

- самонавчання: модель навчається на власних прогнозах, що дозволяє їй розвивати більш глибоке розуміння мови;
- підсилювальне навчання: модель навчається на основі зворотного зв'язку, оптимізуючи свої відповіді для досягнення бажаних результатів;
- навчання з учителем: модель навчається на великих наборах даних, де кожен приклад має правильну відповідь.

Модель Gemini заснована на найсучасніших технологіях глибокого навчання, включаючи:

- Трансформери: Це архітектура нейронної мережі, яка особливо ефективна для обробки послідовних даних, таких як текст;
- Атеншн-механізми: Ці механізми дозволяють моделі зосереджуватися на найважливіших частинах вхідних даних;
- Розподілене обчислення: Для навчання таких великих моделей використовуються потужні обчислювальні кластери.

Модель Gemini – це результат значних інвестицій в дослідження штучного інтелекту. Вона представляє новий рівень можливостей для мовних моделей, демонструючи здатність до глибокого розуміння мови,

генерації творчого контенту та виконання складних завдань, таких як аналіз даних з теплиці та надання рекомендацій (рис. 4.1-4.7) [35].

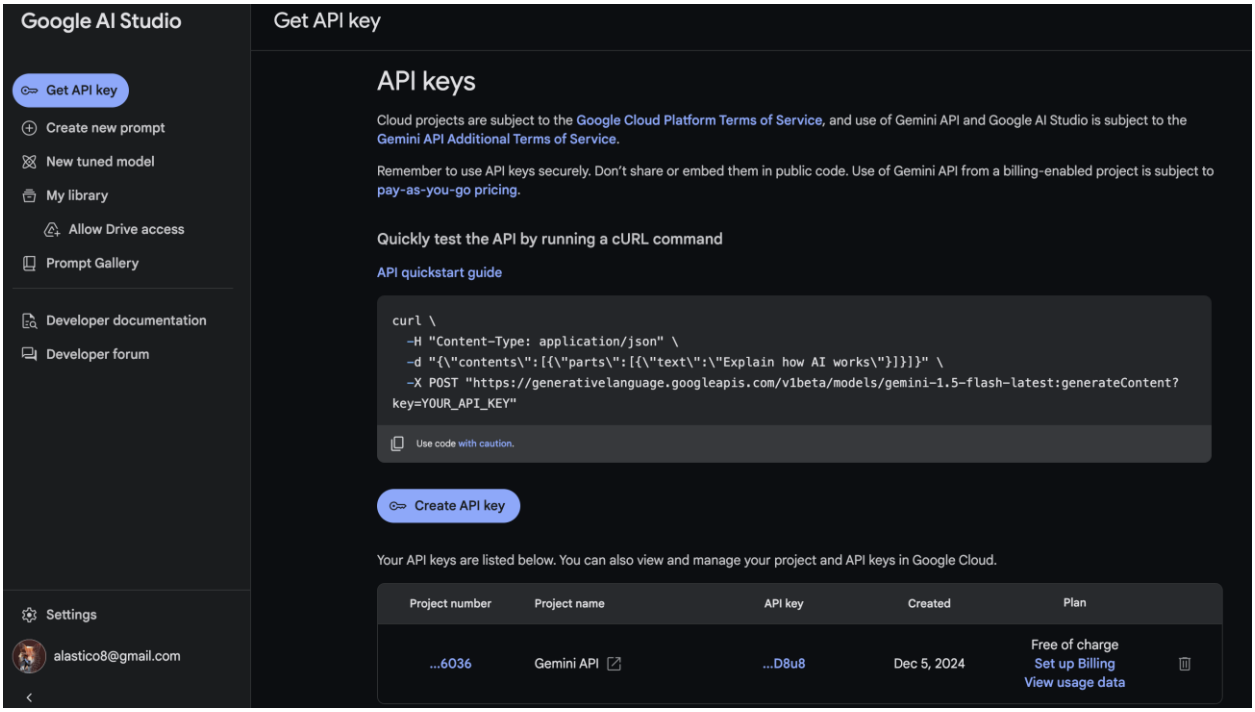

Рисунок 4.1 – Генерація API ключа для штучного інтелекту та приклад використання з документації проекту

```
Future<dynamic> getAiInfo(text) async {
 var response = await Dio().post(
    'https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-flash-latest:generateContent?key=AIzaSyBmE3_FZ9PQT0x3OFfz0pnHmwwLt
   data: {
     "contents": [
       {
         "parts": [
           {
             "text": text,  // Запрос от пользователя
           }
         ]
       }
     ]
   },
   options: Options(
     headers: {
       'Content-Type': 'application/json',
     },
   )
 );
 return response;
}
```

Рисунок 4.2 – Частина коду зі створенням функції для post запиту до штучного інтелекту

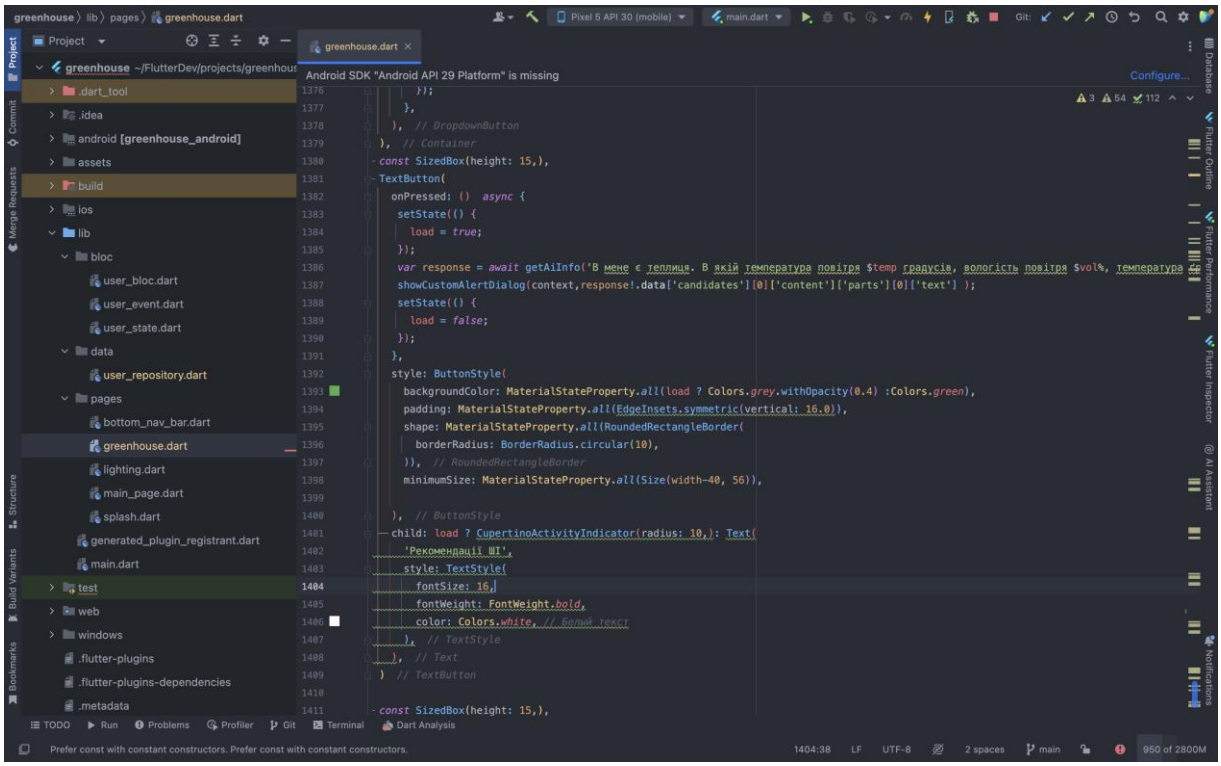

Рисунок 4.3 – Частина коду з кнопкою надання рекомендацій і аналізу від штучного інтелекту

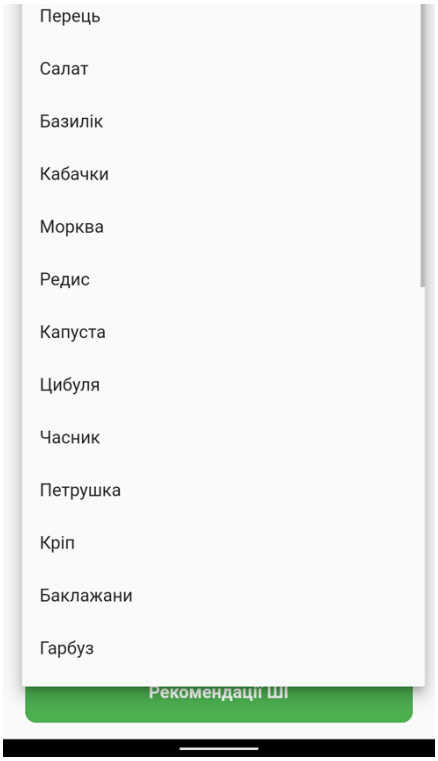

Рисунок 4.4 – Екран додатку з вибором рослини яку вирощують в теплиці

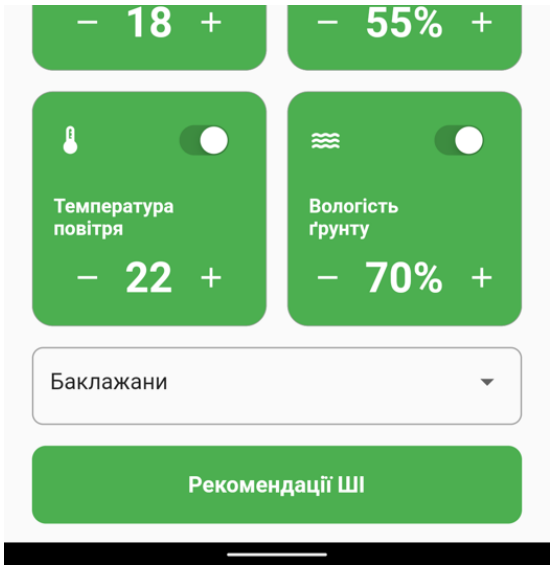

Рисунок 4.5 – Екран додатку з кнопкою отримання рекомендацій від Gemini

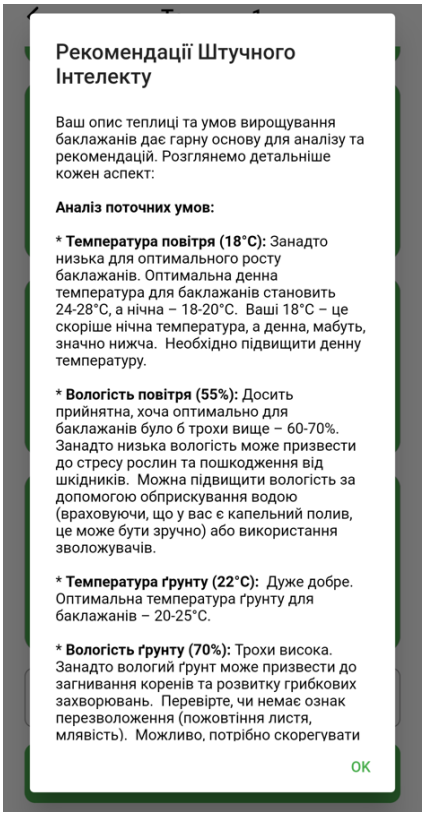

Рисунок 4.6 – Екран додатку з прикладом відповіді від штучного інтелекту

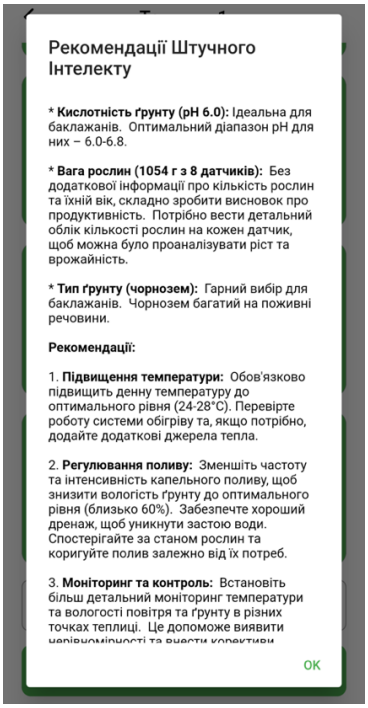

Рисунок 4.7 – Екран додатку з прикладом рекомендацій від штучного інтелекту

Для встановлення з'єднання з пристроєм виконуються HTTP запити з різними параметрами, деталі яких наведені в додатку 1. Параметри отримуються з HTML відповіді сервера (контролера), де вони представлені у вигляді рядкового списку. Додаток обробляє цю відповідь, парсить її та оновлює дані про теплицю. Завдяки цьому, користувач може отримувати актуальні показники стану теплиці або виконувати різноманітні операції, такі як включення освітлення, запуск поливу або коригування температури та інших умов. Цей процес забезпечує зручне і ефективне управління теплицею в реальному часі через додаток.

4.2. Тестування моделі вбудованої системи автоматизації технологічних процесів вирощування рослин у теплиці

Для тестування модульної системи використовувався пакет MSTest. У наступному прикладі (рисунок 4.8) показані результати тестової установки.

MSTest, Microsoft Testing Framework - це платформа тестування для додатків .NET. Він дозволяє створювати та виконувати тести, а також надавати набори тестів з інтеграцією з оглядачем Visual Studio та Visual Studio Code Test, .NET CLI та багатьма конвеєрами CI. MSTest - це повністю підтримувана платформа з відкритим вихідним кодом і кросплатформова платформа тестування, яка працює з усіма підтримуваними цільовими об'єктами .NET (платформа .NET Framework, .NET Core, .NET, .NET, UWP, WinUI тощо) розміщеними на GitHub.

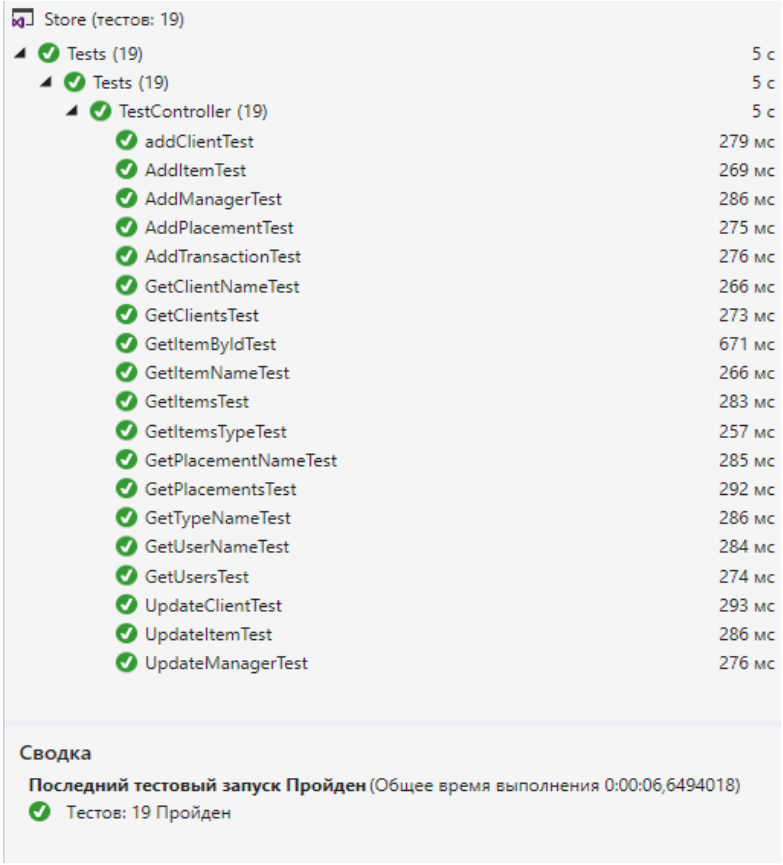

Рисунок 4.8 – Тестування розробленої системи

У процесі тестування системи кожен учасник повинен ретельно перевірити, що програмне забезпечення не має дефектів, перш ніж його буде випущено для кінцевих користувачів або замовників. Тестуванню

підлягають не лише функціональні можливості програмного забезпечення, а й усі бізнес-процеси та інтеграційні системи, що були реалізовані.

Системне тестування – це складний процес, який вимагає створення спеціалізованих методів і інструментів для оцінки ефективності та стабільності програмного забезпечення. Тестування повинно охоплювати всі аспекти функціонування системи, перевіряючи її здатність задовольняти вимоги користувачів та замовників. Для цього необхідно провести тестування з реалістичними даними, які імітують реальні умови використання. Це завдання може бути досить складним, особливо якщо система працює з великими обсягами даних або взаємодіє з численними зовнішніми системами.

Підготовка тестових сценаріїв, які покривають усі можливі функціональні можливості системи, є важливим етапом. Тестові випадки повинні охоплювати різні варіанти взаємодії користувачів із системою, щоб забезпечити її безпеку, ефективність і відповідність специфікаціям. На рисунках 4.9-4.13 відображено тестування додатку.

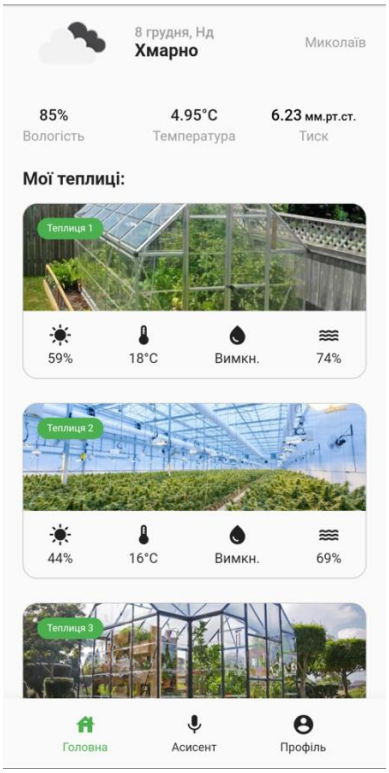

Рисунок 4.9 – Скріншот з тестування головного екрану теплиці

Рисунок 4.10 – Скріншот з тестування екрану обраної теплиці

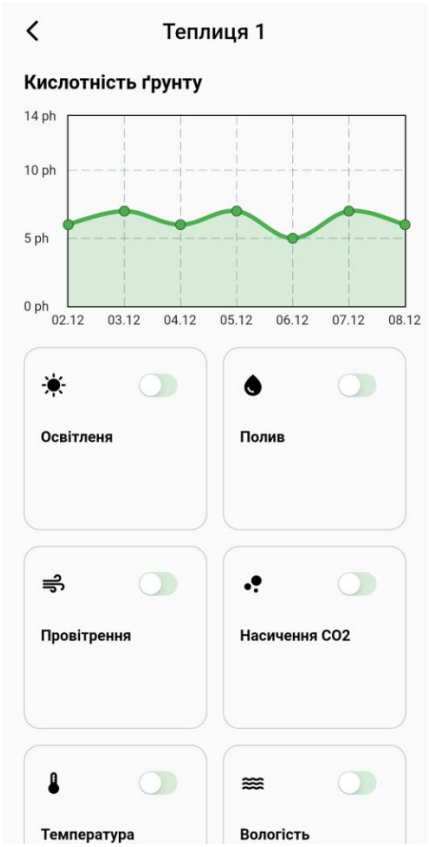

Рисунок 4.11 – Скріншот з тестування екрану налаштування теплиці

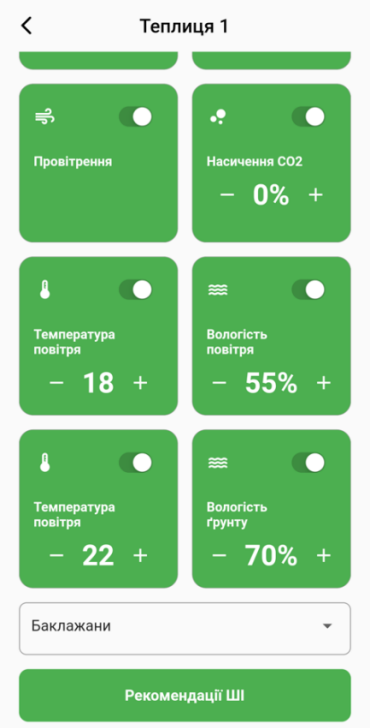

Рисунок 4.12 – Скріншот з тестування екрану конфігурації теплиці

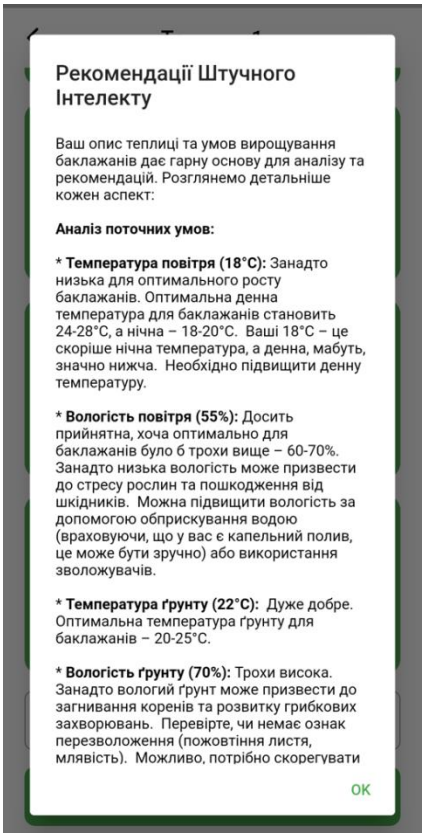

Рисунок 4.13 – Скріншот з тестування відповіді штучного інтелекту

Висновки за розділом 4

Підсумовуючи четвертий розділ, можна зробити наступні висновки:

1. Виконано інтеграцію штучного інтелекту у додаток для надання рекомендацій щодо вирощування рослин.
2. Проведено тестування моделі вбудованої системи автоматизації технологічних процесів вирощування рослин у теплиці.

РОЗДІЛ 5

ОХОРОНА ПРАЦІ ТА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

5.1. Основні поняття про охорону праці.

Охорона праці — це інтегрована система, яка включає правові, соціально-економічні, організаційно-технічні, гігієнічні та профілактичні заходи, спрямовані на збереження здоров'я та працездатності людей під час виконання роботи [16].

Захист здоров’я співробітників і забезпечення безпечних умов праці є ключовим елементом розвитку суспільства. Наразі велика увага приділяється впровадженню сучасних наукових підходів до організації роботи та мінімізації застосування низькокваліфікованої фізичної праці. Громадськість акцентує увагу на створенні умов праці, які виключають можливість виникнення професійних хвороб і травм на робочих місцях. Це реалізується за допомогою інноваційних методів і технологій, які гарантують безпеку працівників.

У цьому розділі висвітлено тему ідентифікації потенційних небезпек, аналізу й оцінки ризиків, а також розробки заходів для підвищення рівня безпеки працівників, залучених до створення веборієнтованого програмного забезпечення.

Розділ підготовлений відповідно до таких нормативних документів:

1. Закон України «Про охорону праці» від 16.10.2020 року.
2. Кодекс законів про працю України від 25.10.2020 року.
3. Постанова Кабінету Міністрів України «Про порядок проведення атестації робочих місць за умовами праці» від 28.10.2016 року.
4. Наказ ДСНС України «Про загальні вимоги до забезпечення роботодавцями охорони праці співробітників» від 25.01.2012 року.

5. Типова інструкція з охорони праці для ІТ-спеціалістів.

5.2. Аналіз небезпечних та шкідливих факторів, що впливають на людину при розробці системи керування

Мета аналізу потенційно небезпечних і шкідливих факторів, що впливають на людину під час розробки системи управління, полягає у виявленні та оцінці ризиків, які виникають у процесі створення та експлуатації цієї системи, з урахуванням таких аспектів:

- умови функціонування компанії;
- параметри виробничих процесів підприємства;
- конструктивні особливості, реальний стан обладнання об’єкта управління та умови його експлуатації;
- наявність технічних і організаційних заходів, спрямованих на запобігання аварійним ситуаціям і мінімізацію їх наслідків.

Аналіз базується на даних, отриманих з нормативно-правових актів, що регламентують вимоги безпеки, а також на проектній, експлуатаційній і технічній документації, наданій керівництвом [16].

На великих виробництвах підвищується ймовірність виникнення пожеж і вибухів. Статистичні дослідження в різних країнах показують, що матеріальні втрати від таких подій у технічно розвинених країнах постійно збільшуються. Однак, завдяки аналізу ризиків, можна запобігти або обмежити негативні наслідки, виявивши та оцінивши потенційні загрози для ІТ-компаній.

Аналіз ризиків включає збір і вивчення інформації про загрози об’єкта, що дозволяє визначити вхідні дані для оцінки ризиків, розробити заходи управління ними, а також вибрати ефективні стратегії мінімізації негативних наслідків.

Таким чином, структурований список потенційних небезпек і джерел ризику під час розробки системи управління представлений у таблиці 4.1.

Таблиця 4.1 Структурований перелік потенційних небезпек і джерел ризику при розробці системи контролю

№ з/п	Показник небезпеки	Джерело
1. Потенційні небезпеки та джерела ризику при нормальному режимі функціонування об’єкту		
1.1	Підвищена запиленість і гази на робочому місці	На робочому місці працівників (шкідливий пил від кулерів ПК)
1.2	Температура повітря робочої зони не за показниками норми	Обслуговування криогенного обладнання
1.3.	Вологість повітря робочої зони не за показниками норми	Тепловиділення від центральних процесорів
1.4	Високий рівень статичної електрики досягає небезпечного рівня	Під час перепаду температур на рабочому місці
1.5	Відсутнє або недостатнє природне освіта вдень	Вимоги до облаштування робочого місця працівника
1.6	Яскравість світла, що визначає зорову роботу	Мерехтливе зображення при роботі монітору
	Продовження табл. 4.1	
1.7	Зменшується контрастність, що характеризує зорову роботу	Неправильна робота екрану монітора.
1.8	Фізичне перевантаження:	
1.8.1	занадто багато статичних змін: - тривала багатогодинна (понад 8 годин)	Умови праці на робочому місці

	робота в статичному положенні тіла. - мала рухова активістка при значних навантаженнях на кістково-м’язовий апарат кистей рук.	
1.8.2	Нервово-психічні перевантаження, що супроводжуються професійною діяльністю: - постійно приймати відповідальні рішення в умовах дефіциту часу; - безпосереднє спілкування з людьми різного темпераменту; - навантаження на елементи зорової системи; - монотонність роботи.	Робота програмістів пов'язана з монотонністю
2. Потенційні небезпеки та джерела ризикув умовах виникнення надзвичайної ситуації		
2.1 Надзвичайні ситуації техногенного характеру		
2.1.1	Пожежа у будівлі чи комунікаціях	Вихід з ладу пристроїв електроживлення
2.1.2	Вибухи в будівлях або комунікаціях	У разі пожежі обладнання
2.2 Особливі соціальні обставини		
2.2.1	Вчинення психічного впливу на людину (мобінг, булінг, шантаж)	Тиск з боку керівництва або колективу
2.2.2	Вчинення фізичного впливу на людину (нанесення побоїв)	Тиск з боку керівництва або колективу

В Україні для оцінки ризиків об’єктів підвищеної небезпеки та забезпечення їхньої безпеки Міністерством праці та соціальної політики було розроблено «Методику визначення ризику та його допустимого рівня» (наказ від 12 грудня 2002 р. № 637). Цей документ регламентує підходи до аналізу потенційних небезпек, визначення аварійних ситуацій, оцінки ризиків та ймовірності відмов, а також до оцінки можливих наслідків аварій.

Відповідно до цієї методики, оцінка ймовірності виникнення несприятливої події здійснюється на основі аналізу характеристик об’єкта тестування, джерел загрози та ситуаційного аналізу.

Ймовірність виникнення несприятливої події можна розрахувати за відповідною формулою, яка враховує вхідні дані з таблиці 4.1, а саме параметри ризику, значення небезпеки та ймовірності відмов:

$$P_d = 1 - \prod (1 - P_i),$$

де і – фактори, що обумовлюють потенційну небезпеку; Рі – вірогідність реалізації індексом і небезпеки.

Цей підхід дозволяє комплексно оцінювати ризики, визначати їх прийнятний рівень і розробляти ефективні заходи для мінімізації небезпеки.

Для визначення вірогідності використовуємо наступну шкалу (табл. 4.2).

Таблиця 4.2 Аналіз джерел небезпеки під час розробки АСКТ

№ з/п	Найменування потенційної небезпеки	*Pa*	*Q*	*R*
1	Підвищена запиленість і гази на робочому місці	5	1	5
2	Температура повітря робочої зони не за показниками норми	7	5	35
3	Підвищена або знижена вологість повітря робочої зони	7	5	35
4	Підвищений рівень статичної електрики, що досягає небезпечних рівнів	8	2	16
5	Відсутність або нестача природного світла у світлий час доби	8	1	8
6	Підвищена яскравість світла, що характеризує зорову роботу	7	3	21
7	Знижена контрастність, що характеризує зорову роботу	7	2	14
8	Фізичне перевантаження:			

8.1	занадто багато статичних змін: - тривала багатогодинна (понад 8 годин) робота в статичному положенні тіла. - мала рухова активістка при значних навантаженнях на кістково-м’язовий апарат кистей рук.	8	2	16
Продовження табл. 4.2				
8.2	Нервово-психічні перевантаження, що супроводжуються професійною діяльністю: - постійно приймати відповідальні рішення в умовах дефіциту часу; - безпосереднє спілкування з людьми різного темпераменту; - навантаження на елементи зорової системи; - монотонність роботи.	8	3	21
9	Пожежа у будівлі чи комунікаціях	7	9	63
10	Вибухи в будівлях або комунікаціях	5	10	50
11	Дослідження психічного впливу на людину (мобінг, булінг, шантаж)	6	2	12
12	Дослідження фізичного впливу на людину (нанесення побоїв)	6	3	18

RA (Risk Assessment) — це процес, який включає систематичний аналіз і оцінку потенційних загроз і ризиків, що виникають під час розробки певної системи чи проекту. Цей підхід дозволяє ідентифікувати небезпеки, оцінити можливі наслідки їх виникнення, а також ймовірність цих подій, формуючи базу для ухвалення зважених управлінських рішень.

Q (Quality) — якість. У рамках оцінки ризиків це поняття стосується якості кожного аспекту розроблюваної системи, зокрема коду, документації та тестування. Забезпечення високої якості знижує ймовірність помилок і, відповідно, ризики, пов'язані з розробкою.

R (Risk) — ризик. Це ключовий параметр оцінки, що дозволяє кількісно чи якісно описати виявлені загрози. Аналіз ризиків передбачає

врахування не тільки ймовірності небезпек, а й масштабів їх потенційного впливу.Отже, згідно з розрахунків даних в таблиці 4.2 видно що загрози під час розробки та встановлення АСКТ найбільш імовірною і небезпечною є: виникнення пожежі в теплиці або комунікації, вибух в теплиці або комунікації, підвищення або зниження температури повітря в робочій зоні, підвищення або зниження температури повітря в робочій зоні.

Розробка систем автоматичного управління вимагає комплексного підходу, що включає роботу з електричними компонентами, дротами та їх підключенням до джерел живлення. Неправильний монтаж або використання низькоякісних елементів може стати причиною короткого замикання, перегріву або навіть пожежі. Для уникнення таких ситуацій необхідно строго дотримуватись встановлених електротехнічних норм, ретельно проєктувати схеми та застосовувати надійні захисні пристрої. При розробці важливо враховувати фізичну безпеку як самих компонентів, так і всієї системи. До цього належать правильне розташування електронних елементів, організація заземлення, встановлення засобів захисту від ударів, вологи, пилу та інших зовнішніх впливів. Окрему увагу потрібно приділяти безпеці персоналу, який буде працювати з обладнанням, забезпечивши їх необхідними знаннями, інструкціями та тренінгами.

Недоліки на етапах проектування, програмування або експлуатації можуть негативно позначитися на роботі системи автоматичного управління. Випадки недостатнього контролю, некоректної реакції на позаштатні ситуації чи відмови компонентів можуть не лише порушити роботу системи, але й завдати значної шкоди, наприклад, втрату врожаю чи порушення кліматичних умов. Тому критично важливо проводити всебічні випробування, ретельне тестування та перевірку функціональності перед введенням системи в експлуатацію.

5.3. Заходи з техніки безпеки для зменшення або усунення впливу шкідливих виробничих факторів при розробці системи керування

При проектуванні та встановленні обладнання важливо враховувати низку небезпечних та шкідливих виробничих факторів, зокрема [11]:

- рух машин і механізмів, які активно залучені до монтажу обладнання;
- транспортування вантажів;
- можливість руйнування конструкцій, що використовуються під час монтажу;
- підвищення напруги в електричних колах, коротке замикання, що можуть виникнути внаслідок впливу на організм людини. Усі пристрої, що використовуються при монтажі обладнання, повинні відповідати встановленим вимогам безпеки, визначеним стандартами та технічними умовами для супутніх пристроїв.

При монтажі пристроїв у вибухонебезпечних умовах важливо дотримуватися суворих вимог безпеки, а також використовувати спеціалізовані інструменти, пристрої та обладнання, які не здатні спричинити іскріння чи інші джерела запалення. Це обумовлено тим, що в таких умовах навіть найменша іскра може призвести до вибуху.

Згідно з вимогами СНіП ІІІ-4-80, відстані, які визначають небезпечні зони, повинні бути розраховані з урахуванням характеристик пристроїв, обладнання та конструкцій, а також типу виконуваних будівельних робіт. Ці відстані повинні забезпечувати безпеку працівників, що знаходяться поблизу місць переміщення устаткування, і захищати їх від можливого падіння предметів. При швидкості вітру 15 м/с і більше встановлення обладнання на відкритих територіях може стати небезпечним, оскільки високі швидкості вітру здатні спричинити падіння предметів або

непередбачений рух обладнання, що може призвести до травм чи пошкоджень.

Проект виробничого регламенту (ПВР) є важливим документом при монтажі обладнання, оскільки він містить вказівки та технічні рішення з питань техніки безпеки, пожежної безпеки і виробничої санітарії. Під час його розробки необхідно враховувати специфічні умови роботи та встановлення устаткування.

Організація безпеки на виробництві включає в себе встановлення небезпечних зон, позначення цих зон та огородження. Ці заходи повинні відповідати чинним нормативним актам України, що регламентують вимоги до безпеки на робочих місцях. Перед початком робіт необхідно провести огляд місця роботи, щоб оцінити умови виконання робіт, виявити потенційні ризики і визначити заходи для їх усунення. Забезпечення необхідними матеріалами і перевірка їх відповідності ПВР є ключовими кроками для гарантування безпеки під час монтажних робіт.

У випадку управління кранами застосовуються умовні позначення, які забезпечують передачу команд операторам і всім учасникам процесу роботи з кранами. ОСТ 36-93-83 (Основні системи та технічні засоби. Умовні Знаки.

Умовні знаки для управління кранами і порядок їх застосування) регламентує стандарти щодо умовних знаків, їх значень та порядку використання при управлінні кранами.

Перед підйомом і монтажем обладнання, деталей та вузлів необхідно перевірити приєднувальні розміри та збіг посадкових місць. Це важливо для забезпечення правильного монтажу та безпечної експлуатації техніки.

Закріплення виконуються відповідно до ПВР, який містить настанови та поради щодо встановлення трубопроводів і майданчиків, включаючи належну фіксацію, використання підходящих кріпильних елементів та відповідних технологій монтажу.

Очищення устаткування перед установкою є необхідним етапом для забезпечення безпеки та надійності роботи. Сніг, бруд і лід можуть спричиняти небезпечні умови, зокрема збільшувати тертя, ускладнювати переміщення частин та монтаж, а також сприяти корозії.

Згідно з ГОСТ 12.2.062-81 "Захисні огородження. Загальні технічні вимоги", обертові та рухомі частини монтуємого обладнання повинні бути оснащені захисними огородженнями.

Монтажна організація не має права здійснювати демонтаж обладнання та підключати встановлені установки до вже використовуваних.

Перевірка міцності з'єднань і виявлення можливих ушкоджень є важливою для забезпечення безпеки праці. У разі виявлення деформацій, руху або інших дефектів, їх слід усунути, а стан з'єднань перевірити перед подальшим використанням. СНіП III-4-80 встановлює вимоги до безпечних процедур роботи, у тому числі до збереження, розпакування, розконсервації та підключення устаткування.

Ці правила спрямовані на уникнення непередбачених ситуацій, забезпечення безпеки працівників і правильне виконання робіт з технікою.

Установку обладнання на підприємстві чи в цеху слід виконувати після отримання дозволу відповідно до СНіП III-4-80.

При розміщенні монтажного майданчика на діючому підприємстві необхідно вжити заходів для запобігання пожежам та зниження впливу небезпечних виробничих факторів на працівників до допустимого рівня.

Прийняті заходи повинні враховувати конкретні умови діючого підприємства для забезпечення безпеки працівників і запобігання нещасним випадкам.

З точки зору соціальної підтримки працівників цієї категорії особливо важливим є затвердження комплексної угоди, що передбачає додаткові пільги та винагороди для спеціалістів.

5.4. Охорона навколишнього природнього середовища.

Охорона навколишнього середовища на підприємстві, що займається вирощуванням рослин у теплиці, є важливим аспектом для запобігання негативному впливу виробничої діяльності на екосистему. Система охорони навколишнього середовища в теплиці включає комплекс заходів, спрямованих на мінімізацію шкідливих викидів та забезпечення безпечних умов праці. В процесі вирощування рослин у теплиці здійснюються наступні заходи для зниження рівня забруднення:

- розробка нормативно-правових актів та комплексних природоохоронних заходів;
- ідентифікація, оцінка та моніторинг викидів шкідливих елементів в атмосферу, зокрема від обладнання для освітлення та системи вентиляції.

Крім того, для забезпечення безпеки життєдіяльності працівників на підприємстві здійснюються організаційні та технічні заходи, спрямовані на мінімізацію впливу виробничих факторів на здоров'я працівників. Зокрема, важливими є питання технічних вимог щодо обладнання теплиці, а також підтримання санітарно-гігієнічних норм.

Для цього розробляється Екологічний паспорт підприємства, в якому фіксуються всі дані, що стосуються охорони навколишнього середовища та охорони праці. В паспорті містяться такі дані:

- схеми очищення стічних вод та викидів в атмосферу;
- інформація про технології, що забезпечують найкращі екологічні показники в процесі вирощування рослин;
- опис технологічних схем вирощування рослин та їх післяобробка;
- дані про тверді та інші відходи, що виникають у процесі вирощування;
- загальна інформація про підприємство.

Для ефективного впровадження екологічних норм у теплиці важливо залучати працівників екологічного контролю, які проводять інвентаризацію викидів та відходів, а також контролюють дотримання вимог екологічної безпеки. Враховуються допустимі рівні концентрацій шкідливих речовин на території теплиці, в атмосферному повітрі та у водних об'єктах.

Для забезпечення мінімізації екологічних ризиків у теплиці необхідно виконати наступні заходи:

1. Оформлення дозвільних документів:
    - інвентаризація викидів від технологічного обладнання теплиці;
    - оформлення дозволів на викиди шкідливих речовин;
    - ведення паспортів та реєстраційних карток на кожен вид відходів;
    - інвентаризація виробничих відходів.
2. Співпраця з органами контролю та екологічними організаціями:
    - міністерство охорони здоров'я;
    - міністерство екології та природних ресурсів;
    - державні органи, що займаються питаннями охорони навколишнього середовища.
3. Контроль за рівнем енергетичних викидів. Використання енергоефективного обладнання та технологій.
4. Контроль за кількістю та складом забруднюючих речовин, що викидаються в атмосферу. Використання спеціальних газоочисних установок для мінімізації впливу на атмосферу.
5. Використання організованих джерел викидів. Встановлення газоочисних пристроїв для забезпечення прийнятного рівня впливу на навколишнє середовище.

6. Забезпечення санітарно-гігієнічних норм. Контроль за рівнем забруднення та підтримка належних умов праці.
7. Обладнання робочих місць. Встановлення газосигналізаторів, вентиляційних систем та фільтрів, що забезпечують безпечні умови праці.
8. Організація спеціально відведених місць для зберігання промислових відходів. Створення умов для зберігання та утилізації відходів, що виникають під час технологічних процесів у теплиці.

Таким чином, впровадження системи охорони навколишнього середовища у процесі вирощування рослин у теплиці є важливим кроком для забезпечення сталого розвитку підприємства, збереження природних ресурсів та здоров'я працівників.

Висновки за розділом 5

Провівши аналіз умов праці при розробці та монтажу АСК, можна зазначити наявність шкідливих і небезпечних факторів, що впливають на працю. Розробка та монтаж таких систем можуть бути супроводжені високим рівнем шуму, зокрема при роботі з обладнанням; створювати несприятливі умови мікроклімату, такі як надмірна або занадто низька температура під час ремонтних робіт; також можуть мати ризик впливу як іонізуючого, так і неіонізуючого випромінювання, такого як рентгенівські промені або електромагнітні хвилі. Працівники повинні бути належним чином ознайомлені з правилами безпеки і використовувати відповідні засоби захисту. Недостатнє природне або штучне освітлення може погіршувати зорову функцію працівників, призводячи до втрати чіткості зору, втому очей та напругу. Надмірна яскравість, контрастність, мерехтіння екранів і відблиски також можуть негативно впливати на зорові функції і комфорт працівників. Важливо забезпечити належне

освітлення і створити оптимальні умови для перегляду інформації, аби зменшити негативний вплив візуальних факторів.

У межах системи управління охороною праці необхідно враховувати "трикутник" складових, що включають правову, організаційну та управлінську компоненти. Правова складова оцінюється на 8,1 балів з 10 можливих, організаційна - на 8,0, а управлінська - на 7,5 балів. Підвищення рівня інформаційної культури та врахування цих складових сприятимуть удосконаленню системи охорони праці та забезпеченню безпечних умов на підприємствах.

З метою покращення охорони праці, фахівці з інформаційних технологій розробили системні заходи, що спрямовані на правову оптимізацію, соціальний захист працівників та оцінку їх професійної придатності. Також необхідно обґрунтувати диференційовані норми праці при використанні нових комп'ютерних технологій, з урахуванням таких чинників, як вік, стать, рівень зорової діяльності, режими праці та відпочинку (протягом робочого дня, тижня та річного режиму відпусток).

Для підвищення продуктивності і забезпечення здоров'я фахівців в інформаційних технологіях необхідно впровадити ергономічну оптимізацію в рамках системи "оператор-термінал". Це сприятиме покращенню розумової працездатності та зорової діяльності, а також відновленню психосоматичного здоров'я працівників. Особлива увага має бути звернена на створення кімнат психофізіологічного розвантаження, що допоможе підвищити ефективність роботи і зберегти здоров'я працівників. Крім того, варто враховувати досвід зарубіжних країн у створенні комфортних умов для зорової роботи та дотриманні виробничої естетики. Не менш важливим є дотримання норм рівня виробничого шуму та забезпечення акустичної тиші поза межами офісних приміщень. Реалізація цих заходів допоможе покращити умови праці, знизити

негативний вплив на здоров'я працівників та підвищити їхню ефективність у галузі інформаційних технологій.

Додатково важливим є застосування функціональної музики в офісах і кабінетах психофізіологічного розвантаження. Це сприяє попередженню перевтоми, підтримці необхідного рівня розумової працездатності фахівців в комп'ютерній галузі. Застосування цих заходів у сфері охорони праці фахівців інформаційних технологій дозволяє підвищити ефективність роботи з новими комп'ютерними технологіями на 10-15% порівняно з традиційними моделями охорони праці. На підприємстві були реалізовані пропозиції та заходи щодо вдосконалення охорони праці для фахівців в інформаційних технологіях, а також розроблена інструкція з охорони праці при роботі за комп'ютером. Впровадження цих заходів сприяє поліпшенню інформаційної культури працівників, формуванню стратегій безпеки, запобіганню перевтоми та зменшенню ризику психофізіологічного навантаження при роботі з новими комп'ютерними технологіями, що позитивно впливає на умови праці, рівень безпеки та здоров'я працівників цієї сфери.

ВИСНОВКИ

Підсумовуючи загальний зміст роботи, можна зробити такі висновки:

1. Визначено, що наразі важливе місце в автоматизаційних процесах належить системам керування, які дозволяють здійснювати контроль за різноманітними пристроями. Зокрема для керування функціонуванням теплиць використовуються мікроконтролери, з яких складається велика кількість сучасної побутової та виробничо-промислової електронної техніки. Теплиці забезпечують до 30% свіжих овочів та фруктів у міжсезоння, а вихід з ладу чи некоректна робота системи керування призводить о значних негативних наслідків та, навіть, втрати врожаю.

2. Проаналізувавши доступні рішення, розроблено власний варіант автоматизованого керування теплицею з широким функціоналом та невеликою ціною. Запропоновано алгоритми обробки інформації для технології автоматизованого керування теплицею на базі контролера Arduino Nano 33 IoT. Для налаштування роботи контролера була створена програма на базі Arduino IDE. Телефон з додатком під'єднується до плати завдяки одній мережі WIFI, виконуючи http-запити, здійснює керування теплицею. Створено екран керування теплицею, де розроблено інтерфейс для повного налаштування. Розроблено віджети, які передають інформацію на сервер та запускають апаратну частину теплиці. Основні робочі функції системи: включення освітлення, зміна теплоти та яскравості світла, контроль та підтримання температури повітря і ґрунту, включення поливу, контроль та підтримання вологості повітря та ґрунту, провітрювання, насичення CO_2. Додатково були створені програмні засоби побудови графіків з 7-денними історіями змін по кислотності ґрунту та

ваги рослин з тензодатчиків. Система дозволяє здійснювати аналіз цих даних та отримувати рекомендації від засобів штучного інтелекту для оптимізації умов вирощування рослин.

3. В роботі також розглянуто основні питання охорони праці та навколишнього середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Александреску О. В. Сучасне проектування на C++. / О. В. Александреску – Вільямс, 2014. – 336 с.

2. Ардатський С. Н., Бартунов О. С. Управління доступом у складних інформаційних системах. – Освітні портали. Випуск 1. - 2005. - 187 с.

3. Бабинова Н. В., Гризов А. И., Сальников Д. М., Сидоренко М. С., Смородинов О. В. Нові платіжні технології. Информаційно-справочну видання./Під. ред. А. И. Гризова. М.: АОЗТ «Рекон», 2001. - 272с.

4. Басманов А. С., Широков Ю. Ф. Мікропроцесори і однокристальних мікро ЕОМ: Номенклатура і функціональні можливості / За ред. В. Г. Домрачева. М .: Вища школа, 1988. - 127 с.

5. Бодрова Г. А., Логвін В. І. Позиціонування та взаємодія у бездротових сенсорних мережах / Г. А. Бодрова, В. І. Логвін // Молодий вчений. – 2015. – № 6. – С. 129-132.

6. Вислоух С. П. Інформаційні технології в задачах технологічної підготовки приладо- та машинобудівного виробництва : монографія / С. П. Вислоух. – К. : НТУУ «КПІ», 2011. – 488 с.

7. Ворожко В. П., Корченко О. Г. Захист інформації з обмеженим доступом. Збірник нормативних документів. – К.: КУЦА, 1999. – 283 с

8. Вуд А. Мікропроцесори в питаннях і відповідях / Пер. з англ. М .: Вища школа, 1985. - 185 с.

9. Гайкович В. Ю., Єршов Д. В. Основи безпеки інформаційних технологій. М.: МІФІ, 1995. – 285с.

10. Гілль Л.С. Сучасні технології овочівництва закритого і відкритого ґрунту. Частина 1. Закритий ґрунт. Вінниця : Нова книга, 2008, 364 с.

11. Геврик Є О. Охорона праці. - К.: Ельга; Ніка-Центр, 2003. - 280 с.

12. Глушков В. М., Амосов М. М., Артеменко І. О. Енциклопедія кібернетики. Том 2. Київ, - 1974. – С. 33-54.

13. Голощапов О. Google Android. Програмування для мобільних пристроїв, 2011. - 438 с.

14. Джошуа, Блох Java. Ефективне програмування / Блох Джошуа. - М.: ЛОРІ, 2014. - 292 с.

15. Домарьов В. В. Безпека І. Т. Методологія створення систем захисту. – М. – СПб. – Київ, 2002. – 688 с.

16. Желібо Є П., Заверуха Н. М., Зацарний В, В. Безпека життедіяльності / За ред. Є П. Желібо. - К.: Каравела, 2010. - 328 с.

17. Інноваційні технології в управлінні складними біотехнологічними об’єктами агропромислового комплексу / А.П. Ладанюк, В.М. Решетюк, В.Д. Кишенько, Я.В. Смітюх. Київ : Центр учбової літератури, 2014. - 280 с.

18. Інтернет-ресурс. АСУ. – Режим доступу: https://uk.wikipedia.org/wiki.

19. Інтернет-ресурс. Arduino. Вікіпедія. – Режим доступу: https://de.wikipedia.org/wiki/Arduino_(Plattform).

20. Ліпунцов Ю. П. Управління процесами. М: Компанія АйТі, 2003. – С. 33-42.

21. Лисенко В.П., Дудник А.О. Оптимальне управління: стан та перспективи розвитку в тепличній галузі. Науковий вісник Національного університету біоресурсів і природокористування України. 2011. - Вип. 166. - С. 53–56.

22. Лотов О. В., Поспелова І. І. Багатокритеріальні задачі прийняття рішень: навч.посіб. М.: МАКС Прес, 2008. – С. 77-89.

23. Офіційний сайт Arduino. Інтернет-ресурс. – Режим доступу: https://www.arduino.cc.

24. Прокопенко Т.О. Інтелектуальна система керування температурно-вологісним режимом у теплиці. Науковий вісник Національного університету біоресурсів і природокористування України. Серія «Техніка та енергетика АПК». 2015. - Вип. 209. Ч. І. - С. 140–147.

25. Прокопенко Т.О., Мірошніченко М.С., Зубенко В.О. Комп'ютерно-інтегрована система автоматизації мікроклімату у теплиці з використанням нейромереж. Вісник Харківського національного технічного університету сільського господарства імені Петра Василенка. Технічні науки. «Проблеми енергозабезпечення та енергозбереження в АПК України». 2014. - Вип. 154. С. 79–82.

26. Самохвалов Ю. Я., Темпіков В. О., Хорошко В. О. Організаційно-технічне забезпечення захисту інформації: Навчальний посібник / За ред. проф. Хорошка В.О. – К.: НАУ, 2002. – С. 207.

27. Соколов В. Ю. Інформаційні системи і технології: Навчальний посібник – Київ ДУІКТ. - 2010. – С. 33-49.

28. Сташин В.В. Проектування цифрових пристроїв на однокристальних мікроконтролерах / Сташин В.В. - М.:Енергоатомвидав., 2010. – 224 с.

29. Усков О. О. Методичні вказівки з виконання курсового проекту по курсу «Розробка і стандартизація програмних засобів інформаційних технологій». Смоленськ: СФ АНО ВПО ЦС РФ «РУК»,2007. – С. 15- 44.

30. Уокерлі Дж. Архітектура та програмування мікро ЕОМ: У 2-х кн./Пер. з англ. М .: Світ, 1984. - Кн. 2. - 359 с.

31. Хосе М. Ангуло. Мікропроцесори: Архітектура, програмування та проектування систем. Тбілісі: Ганатлеба, 1989. – С. 45-54.

32. Щелкунов М. М. Мікропроцесорні засоби та системи / Щелкунов М. М., Діанов О. Н. - М.:Радіо та зв'язок, 2009. – 360 с.

33. Магазин з продажу теплиць – Режим доступу: https://2agrocloud.com/lp/gh/.

34. Магазин з продажу теплиць Екотек – Режим доступу: https://teplitca.kiev.ua/ua/p631717548-avtomaticheskaya-teplitsa.html.
35. Документація API Gemini – Режим доступу: https://ai.google.dev/gemini-api/docs.
36. Вебсайт Arduino UA – Режим доступу: https://arduino.ua/.

ДОДАТОК 1

Лістинг програми

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>
#include <Arduino_JSON.h>
#include <RTClib.h>
#include <DHT22.h>
#include <BH1750.h>
#include <Wire.h>
#include <DS18B20.h>
#include <DFRobot_HX711.h>
#include <PulseFlowMeter.h>
#include "DFRobot_PH.h"

#define IN_PH 21
float voltage,phValue,temperature = 25;
DFRobot_PH pH;

PulseFlowMeter meter;

#define PIN 35 // Input pin
#define K 25.613 // Pulses per liter parameter of flowmeter
#define MIN_FLOW 5 // Minimal flowrate treshold
#define MAX_FLOW 50 // Maximal flowrate treshold

BH1750 lightMeter;

#define ZERO_POINT_VOLTAGE  0.324 //define the output of the sensor in volts
when the concentration of CO2 is 400PPM
#define REACTION_VOLTGAE    0.020 //define the voltage drop of the sensor when
move the sensor from air into 1000ppm CO2
#define DC_GAIN             8.5 //define the DC gain of amplifier
#define READ_SAMPLE_INTERVAL 50 //define how many samples you are going to
take in normal operation
#define READ_SAMPLE_TIMES   5 //define the time interval(in milisecond)
between each samples in
float CO2Curve[3] = {2.602,ZERO_POINT_VOLTAGE,(REACTION_VOLTGAE/(2.602-3))};

//DS1302 rtc;
DS1302 rtc(25, 33, 32);
DFRobot_HX711 MyScale(3, 4);

#define IN_SOIL_MOISTURE    15
#define IN_CO2              16
#define IN_OPT101           17
#define DATA_SOIL_TEMP      13
```

```
#define DATA_AIR_TEMP       14 // SDA, or almost any other I/O pin

#define OUT_AIR_HUMIDIFIER  12
#define OUT_SOIL_MOISTURE   11
#define OUT_CO2             10
#define OUT_AIR_HEATER      9
#define OUT_SOIL_HEATER     8
#define OUT_LIGHTING        7
#define OUT_AIRING          6
#define OUT_ALARM           5

DS18B20 ds(DATA_SOIL_TEMP);
DHT22 dht22(DATA_AIR_TEMP);


struct StructRealTime {
  uint8_t hour;
  uint8_t minute;
  uint8_t second;
  uint8_t day;
  uint8_t month;
  uint8_t year;
};

struct StructMinCurMaxInt8 {
  int8_t min;
  int8_t current;
  int8_t last;
  int8_t max;
  uint8_t error;
};

struct StructMinCurMaxUint8 {
  uint8_t min;
  uint8_t current;
  uint8_t last;
  uint8_t max;
  uint8_t error;

};

struct StructMinCurMaxUint16 {
  uint16_t min;
  uint16_t current;
  uint16_t last;
  uint16_t max;
  uint8_t error;
};
```

```
struct StructBool {
  bool current;
};

struct {
  StructRealTime realTime;
  StructMinCurMaxInt8 airTemperature;
  StructMinCurMaxUint8 airHumidity;
  StructMinCurMaxUint8 soilMoisture;
  StructMinCurMaxUint8 soilTemperature;
  StructMinCurMaxUint16 illumination;
  StructMinCurMaxUint8 CO2;
  StructMinCurMaxUint8 pH;
  StructMinCurMaxUint16 Weight;
  StructMinCurMaxUint16 Water;
  StructBool Error;
  StructBool Alarm;
} allParams;

// Set these to your desired credentials.
const char *ssid = "";
const char *password = "";

WiFiServer server(80);


void setup() {

pinMode(OUT_AIR_HUMIDIFIER, OUTPUT);
pinMode(OUT_SOIL_MOISTURE, OUTPUT);
pinMode(OUT_CO2, OUTPUT);
pinMode(OUT_AIR_HEATER, OUTPUT);
pinMode(OUT_SOIL_HEATER, OUTPUT);
pinMode(OUT_LIGHTING, OUTPUT);
pinMode(OUT_AIRING, OUTPUT);
pinMode(OUT_ALARM, OUTPUT);


// Initialize the I2C bus (BH1750 library doesn't do this automatically)
  Wire.begin();
  lightMeter.begin(BH1750::ONE_TIME_HIGH_RES_MODE);
  while (!lightMeter.measurementReady(true)) {
    yield();
  }

  pH.begin();
  rtc.begin();
```

```
  if (!rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date & time this sketch was compiled
    rtc.adjust(DateTime(__DATE__, __TIME__));
  }

  Serial.begin(115200);
  Serial.println();
  Serial.println("Configuring access point...");

  // You can remove the password parameter if you want the AP to be open.
  // a valid password must have more than 7 characters
  if (!WiFi.softAP(ssid, password)) {
    log_e("Soft AP creation failed.");
    while (1)
      ;
  }
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);
  server.begin();

  Serial.println("Server started");
}

void loop() {
  static uint8_t timeCounter = 100;
  if (timeCounter++ >= 100) {
    getAllParameters();
    checkingConditions();
    timeCounter = 0;
  }
  WiFiThread();
  delay(100);
}

void getAllParameters() {
  readTime(&allParams.realTime);
  readAirTemperature(&allParams.airTemperature);
  readAirHumidity(&allParams.airHumidity);
  readSoilMoisture(&allParams.soilMoisture);
  readIllumination(&allParams.illumination);
  readpH(&allParams.pH);
}

void WiFiThread() {
  WiFiClient client = server.available();  // listen for incoming clients
  if (client) {                            // if you get a client,
```

```
    Serial.println("New Client.");          // print a message out the serial
port
    String currentLine = "";                // make a String to hold incoming
data from the client
    while (client.connected()) {            // loop while the client's
connected
      if (client.available()) {             // if there's bytes to read from
the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        if (c == '\n') {                    // if the byte is a newline
character

          // if the current line is blank, you got two newline characters in a
row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1
200 OK)
            // and a content-type so the client knows what's coming, then a
blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();

            // the content of the HTTP response follows the header:
            client.println("Connection: close");  // закрыть сессию после
ответа
            client.println();                     // пустая строка отделяет
тело сообщения
            client.println("<!DOCTYPE HTML>");    // тело сообщения
            client.println("<html>");
            client.println("<p>Time = " + String(allParams.realTime.hour) +
":" + allParams.realTime.minute + " " + allParams.realTime.day + "." +
allParams.realTime.month + "." + allParams.realTime.year + "<br>"
                           + "Air Temperature : min = " +
allParams.airTemperature.min + " current = " +
allParams.airTemperature.current + " max = " + allParams.airTemperature.max +
" °C" + " Err = " + allParams.airTemperature.error + "<br>"
                           + "Air Humidity : min = " +
allParams.airHumidity.min + " current = " + allParams.airHumidity.current + "
max = " + allParams.airHumidity.max + " °C" + " Err = " +
allParams.airHumidity.error + "<br>"
                           + "Soil Moisture : min = " +
allParams.soilMoisture.min + " current = " + allParams.soilMoisture.current +
" max = " + allParams.soilMoisture.max + " %" + " Err = " +
allParams.soilMoisture.error + "<br>"
```

```
                            + "Soil Temperature : min = " +
allParams.soilTemperature.min + " current = " +
allParams.soilTemperature.current + " max = " + allParams.soilTemperature.max
+ " °C" + " Err = " + allParams.soilTemperature.error + "<br>"
                            + "Illumination : min = " +
allParams.illumination.min + " current = " + allParams.illumination.current +
" max = " + allParams.illumination.max + " W/m2" + " Err = " +
allParams.illumination.error + "<br>"
                            + "Water :  current = " + allParams.Water.current +
" L" + "<br>"
                            + "CO2 : min = " + allParams.CO2.min + " current =
" + allParams.CO2.current + " max = " + allParams.CO2.max + " %" + " Err = " +
allParams.CO2.error + "<br>"
                            + "Weight :  current = " + allParams.Weight.current
+ "<br>"
                            + "pH = " + allParams.pH.current + "<br>"
                            + "Alarm = " + allParams.Alarm.current + "<br>"
                            + "</p>");
            client.println("</html>");

            // The HTTP response ends with another blank line:
            client.println();
            // break out of the while loop:
            break;
          } else {  // if you got a newline, then clear currentLine:
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything else but a carriage
return character,
          currentLine += c;      // add it to the end of the currentLine
        }

        JSONVar myObject = JSON.parse(currentLine);
        if (JSON.typeof(myObject) != "undefined") {
          //JSONVar keys = myObject.keys();
          for (int x = 0; x < myObject.keys().length(); x++) {
            if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("TimeHour")) {
              allParams.realTime.hour = (int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("TimeMinute")) {
              allParams.realTime.minute =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("TimeSecond")) {
              allParams.realTime.second = (int)myObject.keys()[x];
              writeTime(&allParams.realTime);
```

```
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("TimeDay")) {
              allParams.realTime.day = (int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("TimeMonth")) {
              allParams.realTime.month =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("TimeYear")) {
              allParams.realTime.year = (int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("AirTemperatureMin")) {
              allParams.airTemperature.min =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("AirTemperatureMax")) {
              allParams.airTemperature.max =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("AirHumidityMin")) {
              allParams.airHumidity.min =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("AirHumidityMax")) {
              allParams.airHumidity.max =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("SoilMoistureMin")) {
              allParams.soilMoisture.min =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("SoilMoistureMax")) {
              allParams.soilMoisture.max =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("SoilTemperatureMin")) {
              allParams.soilTemperature.min =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("SoilTemperatureMax")) {
              allParams.soilTemperature.max =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("IlluminationMin")) {
              allParams.illumination.min =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("SoilMoistureMax")) {
```

```
              allParams.illumination.max =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("CO2Max")) {
              allParams.illumination.max =
(int)myObject.keys()[x];
            }else if
((JSON.typeof(myObject[myObject.keys()[x]])).equals("CO2Min")) {
              allParams.illumination.min =
(int)myObject.keys()[x];
            }
          }
        }
      }
    }
    // close the connection:
    client.stop();
    Serial.println("Client Disconnected.");
  }
}

void readTime(StructRealTime *time) {
  DateTime now = rtc.now();
  (*time).hour = now.hour();
  (*time).minute = now.minute();
  (*time).day = now.day();
  (*time).month = now.month();
  (*time).year = now.year();
}

void writeTime(StructRealTime *time) {
  rtc.adjust(DateTime((*time).year, (*time).month, (*time).day, (*time).hour,
(*time).minute, (*time).second));
}

void readAirTemperature(StructMinCurMaxInt8 *temp) {
  (*temp).current = (int)dht22.getTemperature();
}

void readAirHumidity(StructMinCurMaxUint8 *humi) {
  (*humi).current = dht22.getHumidity();
}

void readSoilMoisture(StructMinCurMaxUint8 *moisture) {
  (*moisture).current = (int)map(analogRead(IN_SOIL_MOISTURE), 330, 630, 0,
100);
}
```

```
void readSoilTemperature(StructMinCurMaxUint8 *soilTemp) {
  (*soilTemp).current = ds.getTempC();
}

void readIllumination(StructMinCurMaxUint16 *illuminatio) {
  (*illuminatio).current += map(analogRead(IN_OPT101), 0, 1024, 300, 1100);
}

void readWeight(StructMinCurMaxUint8 *Weight) {
  (*Weight).current = MyScale.readWeight();
  if ((*Weight).current < (*Weight).min) (*Weight).min = (*Weight).current;
  else if ((*Weight).current > (*Weight).max) (*Weight).max =
(*Weight).current;
}

void readWaterFlow(StructMinCurMaxUint16 *water) {
  (*water).current = meter.getTotalVolume();
}

void readpH(StructMinCurMaxUint8 *ph) {
  float voltage = analogRead(IN_PH)/1024.0*5000;
  (*ph).current = pH.readPH(voltage, allParams.soilTemperature.current);
}

// -- CO2 --
void readCO2(StructMinCurMaxUint8 *CO2) {
  (*CO2).current = MGGetPercentage(MGRead(IN_CO2), CO2Curve);
}

float MGRead(int mg_pin)
{
  int i;
  float v=0;
  for (i=0;i<READ_SAMPLE_TIMES;i++) {
    v += analogRead(mg_pin);
    delay(READ_SAMPLE_INTERVAL);
    }
    v = (v/READ_SAMPLE_TIMES) *5/1024 ;
    return v;
}

int MGGetPercentage(float volts, float *pcurve)
{
  if ((volts/DC_GAIN )>=ZERO_POINT_VOLTAGE) {
    return -1;
  }
  else {
     return pow(10, ((volts/DC_GAIN)-pcurve[1])/pcurve[2]+pcurve[0]);
```

```
  }
}
//-----


void checkingConditions(){
  checkAirTemperatureHumidity();
  checkSoilTemperatureMoisture();
  checkIllumination();
  checkCO2();
}

void checkAirTemperatureHumidity(){
  static int countT = 0, countH = 0;
  if(allParams.airTemperature.current < allParams.airTemperature.min){
    digitalWrite(OUT_AIR_HEATER, HIGH);
    if (countT == 0){
      allParams.airTemperature.last = allParams.airTemperature.current;
    }else if (countT >= 1000){
      countT = -1;
      if(allParams.airTemperature.current > allParams.airTemperature.last){
        allParams.Error.current = 0;
      }else if(allParams.airTemperature.current <
allParams.airTemperature.min){
        allParams.Alarm.current = 1;
        digitalWrite(OUT_ALARM , HIGH);
      }else{
        allParams.Error.current = 1;
      }
    }
    countT++;
  }else if(allParams.airTemperature.current >= allParams.airTemperature.max){
    digitalWrite(OUT_AIR_HEATER, LOW);
    countT = 0;
    if (allParams.airTemperature.current >= allParams.airTemperature.max +
10){
      digitalWrite(OUT_AIRING, HIGH);
      if (allParams.airTemperature.current >= allParams.airTemperature.max +
50){
        digitalWrite(OUT_AIR_HUMIDIFIER, HIGH);
        digitalWrite(OUT_ALARM, HIGH);
        allParams.Alarm.current = 1;
      }
    }else{
      digitalWrite(OUT_AIRING, LOW);
    }
  }

  if(allParams.airHumidity.current < allParams.airHumidity.min){
```

```
    digitalWrite(OUT_AIR_HUMIDIFIER, HIGH);
    if (countH == 0){
      allParams.airHumidity.last = allParams.airHumidity.current;
    }else if (countH >= 1000){
      countH = -1;
      if(allParams.airHumidity.current > allParams.airHumidity.last){
        allParams.Error.current = 0;
      }else{
        allParams.Error.current = 1;
      }
    }
    countH++;
  }else if(allParams.airHumidity.current >= allParams.airHumidity.max){
    digitalWrite(OUT_AIR_HUMIDIFIER, LOW);
    allParams.Error.current = 0;
    countH = 0;
  }
}

void checkSoilTemperatureMoisture(){
  static int countT = 0, countM = 0;
  if(allParams.soilTemperature.current < allParams.soilTemperature.min){
    digitalWrite(OUT_SOIL_HEATER, HIGH);
    if (countT == 0){
      allParams.soilTemperature.last = allParams.soilTemperature.current;
    }else if (countT >= 1000){
      countT = -1;
      if(allParams.soilTemperature.current > allParams.soilTemperature.last){
        allParams.Error.current = 0;
      }else if(allParams.soilTemperature.current <
allParams.soilTemperature.min){
        allParams.Alarm.current = 1;
        digitalWrite(OUT_ALARM , HIGH);
      }else{
        allParams.Error.current = 1;
      }
    }
    countT++;
  }else if(allParams.soilTemperature.current >=
allParams.soilTemperature.max){
    digitalWrite(OUT_SOIL_HEATER, LOW);
    countT = 0;
  }

  if(allParams.soilMoisture.current < allParams.soilMoisture.min){
    digitalWrite(OUT_SOIL_MOISTURE, HIGH);
    if (countM == 0){
      allParams.soilMoisture.last = allParams.soilMoisture.current;
```

```
    }else if (countM >= 1000){
      countM = -1;
      if(allParams.airHumidity.current > allParams.airHumidity.last){
        allParams.Error.current = 0;
      }else{
        allParams.Error.current = 1;
      }
    }
    countM++;
  }else if(allParams.airHumidity.current >= allParams.airHumidity.max){
    digitalWrite(OUT_AIR_HUMIDIFIER, LOW);
    countM = 0;
  }
}

void checkIllumination(){
  static int count = 0;
  if (allParams.illumination.current < allParams.illumination.min &&
(allParams.realTime.hour > 20 || allParams.realTime.hour < 3)){
    digitalWrite(OUT_LIGHTING, HIGH);
    if (count == 0){
      allParams.illumination.last = allParams.illumination.current;
    }else if (count >= 1000){
      count = -1;
      if(allParams.illumination.current <= allParams.illumination.last){
        allParams.illumination.error = 1;
      }
      else{
        allParams.illumination.error = 0;
      }
    }
    count++;
  }else if (allParams.illumination.current >= allParams.illumination.max &&
(allParams.realTime.hour > 20 || allParams.realTime.hour < 3)){
    digitalWrite(OUT_LIGHTING, LOW);
    count = 0;
  }
  if (allParams.realTime.hour == 4){
    digitalWrite(OUT_LIGHTING, LOW);
    allParams.illumination.current = 0;
  }
}

void checkCO2(){
  static int count = 0;
  if (allParams.CO2.current < allParams.CO2.min){
    digitalWrite(OUT_CO2, HIGH);
    if (count == 0){
```

```
      allParams.CO2.last = allParams.CO2.current;
    }else if (count >= 1000){
      count = -1;
      if(allParams.CO2.current <= allParams.CO2.last){
        allParams.CO2.error = 1;
      }
      else{
        allParams.CO2.error = 0;
      }
    }
    count++;
  }else if (allParams.CO2.current >= allParams.CO2.max){
    digitalWrite(OUT_CO2, LOW);
    count = 0;
  }

}
```