Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий інститут електроенергетики
(інститут)
Електротехнічний факультет
(факультет)
Кафедра кіберфізичних та інформаційно-вимірювальних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню магістра

студента Погрібняк Ірини Олегівни
(П.І.Б.)

академічної групи 151м-19-1.
(шифр)

спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
(код і назва спеціальності)

за освітньо-професійною програмою 151 Автоматизація та комп'ютерно-інтегровані технології
(офіційна назва)

на тему Автоматизація процесу керування мікрокліматом в кімнатній теплиці
(назва за наказом ректора)

Консультанти	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинг.	інституційною	
Керівник кваліфікаційної роботи	проф. Ткачов В.В.			
Провідний консультант	ст.викл. Бойко О.О.			
Синтез системи керування	доц. Бубліков А.В.			
Експериментальний розділ	ст.викл. Бойко О.О.			
Економічна частина	ст. викл. Яремчук І.О.			
Охорона праці та безпека в надзвичайних ситуаціях	проф. Чеберячко Ю.І.			
Рецензент				
Нормоконтролер	ас. Славінський Д.В.			

Дніпро
2020

ЗАТВЕРДЖЕНО:
завідувач кафедри
кіберфізичних та інформаційно-вимірювальних систем
(повна назва)
_____ Ткачов В.В.
(підпис) (прізвище, ініціали)

«_____»_____2020 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістра

студенту Погрібняк І. О. **академічної групи** 151м-19-1
(прізвище та ініціали) (шифр)

спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

за освітньо-професійною програмою 151 Автоматизація та комп'ютерно-інтегровані технології
(офіційна назва)

на тему Автоматизація процесу керування мікрокліматом в кімнатній теплиці**,**

затверджену наказом ректора НТУ «Дніпровська політехніка» від 20.11.2020 № 965-с

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання. Теоретичний розділ	Вступ. Опис технологічного процесу для об'єкта автоматизації. Огляд існуючих систем автоматизації. Стан питання. Вибір напрямку створення автоматизованої системи. Визначення моделі об'єкта керування.	12.10.2020
Синтез системи керування	Обрання структури системи керування та регулятора. Розрахунок параметрів регулятора. Дослідження функціонування системи керування на базі обраного регулятора.	02.10.2020
Експериментальний розділ	Розробка програмного забезпечення системи керування на підставі обраного регулятора та його налаштувань.	23.11.2020
Економічна частина	Економічне обґрунтування доцільність витрат на створення системи керування.	30.11.2020
Охорона праці	Розробка організаційно-технічних заходів, щодо реалізації правил безпеки при експлуатації системи.	07.12.2020

Завдання видано _____ ст. викл., Бойко О.О.
(підпис п.конс.) (прізвище, ініціали)

Дата видачі 01.09.2020

Дата подання до атестаційної комісії 14.12.2020

Прийнято до виконання _____ Погрібняк І. О.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить: 126 сторінок, 59 рисунки, 19 таблиці, 21 джерел, 3 додатків.

Об’єкт дослідження: процес вирощування рослин в кімнатній теплиці.

Предмет дослідження: автоматизація процесу керування мікрокліматом в кімнатній теплиці.

Мета кваліфікаційної роботи: забезпечення підтримки оптимальних умов мікроклімату в кімнатній теплиці.

Основними методами дослідження використаними для досягнення поставленої мети були: аналіз літературних джерел, декомпозиція, імітаційне моделювання, теорії автоматичного керування.

Згідно до вимог в якості пристрою керування обрано одноплатний контролер WEMOS D1. Для контролера обрані модулі та джерела живлення. Виконано розробку схеми електричної принципової. На підставі обраного апаратного забезпечення та схеми виготовлено контролер системи керування мікрокліматом в кімнатній теплиці.

На підставі аналітичних засад в графічному середовищі імітаційного моделювання Simulink математичного пакету MATLAB розроблено модель об’єкта керування. За результатами аналізу моделі об’єкта обрано структуру системи керування та розроблено її модель на базі пропорційно-інтегрального регулятора з обмеженням керуючого впливу та запобіганням перенасичення інтегральної складової за методом “защіпки”. Проведено дослідження та обґрунтування параметрів регулятора за яким розроблено програмне забезпечення системи керування з людино-машинним інтерфейсом.

Подальшим напрямком розвитку роботи є розробка людино-машинного інтерфейсу системи керування для пристроїв операційної системою Android та iOS.

Ключові слова: КІМНАТНА ТЕПЛИЦЯ, WEMOS D1, SCADA СИСТЕМА ZENON, ПІ-РЕГУЛЯТОР, ІМПУЛЬСНИЙ ПЕРЕТВОРЮВАЧ.

ЗМІСТ

ВСТУП

В кваліфікаційній роботі об’єктом дослідження є процес вирощування рослин в кімнатній теплиці, предметом дослідження є автоматизація процесу керування мікрокліматом в кімнатній теплиці. Метою кваліфікаційної роботи є забезпечення підтримки оптимальних умов мікроклімату в кімнатній теплиці. В якості об’єкта керування виступає кімнатна теплиця.

В роботі проаналізовано існуючи пристрої керування котрі можуть бути використані при розробці контролера системи керування. За результатами аналізу в якості пристрою керування обрано одноплатний контролер WEMOS D1. Для контролера обрані модулі та джерела живлення. Виконано розробку схеми електричної принципової. На підставі обраного апаратного забезпечення та схеми виготовлено контролер системи керування мікрокліматом в кімнатній теплиці.

В графічному середовищі імітаційного моделювання Simulink математичного пакету MATLAB розроблено модель об’єкта керування. За результатами аналізу моделі об’єкта обрано структуру системи керування та розроблено її модель на базі пропорційно-інтегрального регулятора з обмеженням керуючого впливу та запобіганням перенасичення інтегральної складової за методом “защіпки”.

Використовуючи модель системи керування виконано дослідження впливу параметрів регулятора на якість функціонування системи керування. У результаті чого отримані залежності показників якості від коефіцієнта підсилення та часу інтегрування. Виконано обґрунтування обрання параметрів отриманого регулятора. Для регулятора розроблено імпульсний перетворювач.

На підставі безперервної моделі регулятора розроблено функцію реалізації пропорційно-інтегрального регулятора на мові програмування C++ для контролера WEMOS D1, яка була використана при створені програмного забезпечення системи керування. Для SCADA системи zenon розроблено людино-машинний інтерфейс який відображає режим роботи системи керування, час доби, стан датчиків та виконавчих пристроїв. Крім того він дозволяє у ручному режимі змінювати стан виконавчих пристроїв.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Галузь промисловості

Сільське господарство – галузь економіки, що призначена для забезпечення населення продовольством і отримання сировини для промисловості. Дана галузь представлена практично у всіх країнах світу, у ній зайнято близько 1,1 млрд економічно активного населення. В галузях сільського та лісового господарства України за даними 2019 року зайнято 1,8 млн. чоловік (рис. 1.1).

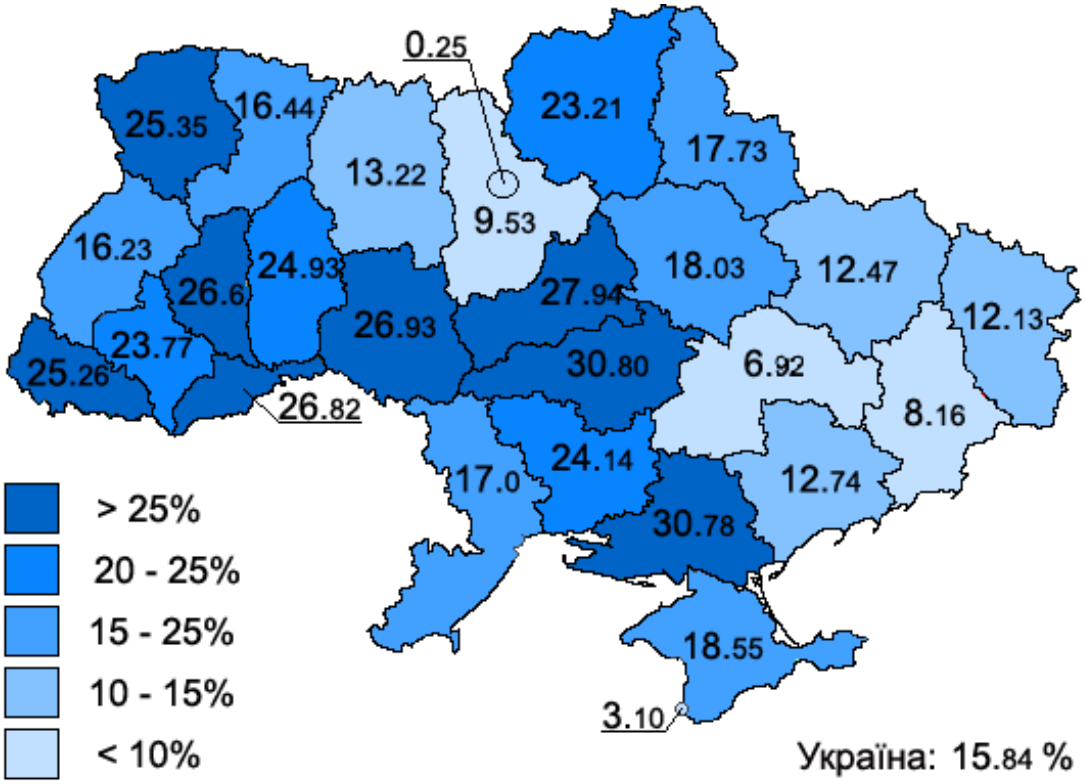


Рисунок 1.1 – Розподіл зайнятості у населення у сільському господарстві по регіонам України

Сільськогосподарські угіддя України займають 42 млн. гектарів, або 70 % загального фонду країни. 78,9 % сільськогосподарських угідь — орні землі та багаторічні насадження, 13,0 % — пасовища, 8,4 % — сіножаті. Найвища частка орних земель — у степових районах (70–80 %) і лісостеповій зоні. Пасовища зосереджені, в основному, в Карпатах, на Поліссі та в південно-східних степових областях, сіножаті — в долинах річок лісової і лісостепової зон.

Станом на 2015 рік українські сільгоспвиробники постачають свою продукцію в 190 країн світу.

За даними уряду, станом на 2019 рік агросектор займає майже 17 % ВВП України і приносить майже 38 % валютної виручки.

Сільське господарство складається з рослинництва і тваринництва. За вартістю продукції рослинництво перевищує тваринництво.

Рослинництво включає вирощування зернових, технічних, кормових, овочевих, баштанних культур і картоплі, садівництво, виноградарство і квіткарство. Однією з найпоширенішої є овочівництво.

Овочівництво поширене по всій території України і має в основному азональний характер. Найбільша концентрація посівів овочевих культур характерна для господарств, розташованих навколо великих міст, для забезпечення населення свіжою продукцією. На Поліссі вирощують переважно огірки, моркву, столовий буряк, капусту, у Лісостепу — огірки, помідори, цибулю, у Степу — помідори, перець, баклажани.

Територія України лежить переважно у помірна континентальній області помірного кліматичного поясу із зростанням континентальності з північного заходу на південний схід. Південний берег Криму виділяється в окремий регіон субтропічного середземноморського клімату. В Українських Карпатах і Кримських горах висота місцевості та експозиція схилів зумовлюють вертикальну зональність клімату.

Середньорічна температура повітря в Україні коливається від +11 °С… +13 °С на півдні до +5 °С… +7 °С на півночі. Пересічна середня температура найхолоднішого місяця (січня) змінюється від −7 °С… −8 °С на північному сході до 0 °С у степовому Криму і +2 °С… +4 °С на Південному узбережжі Криму. У найтеплішому місяці (липні) середньомісячна температура змінюється від +17 °С… +19 °С на півночі та північному заході країни до +22 °С… +23 °С у південних районах і +25 °С — на Південному узбережжі Криму.

Через кліматичні умови Україна не може вирощувати сільськогосподарські культури весь рік. Щоб вирішити цю проблему використовується декілька способів зберігання продуктів і сільськогосподарські теплиці.

1.2 Технологічний процес

Теплиця – конструкція зі стінами та дахом, зроблена з прозорого матеріалу, такого як скло, в яких вирощують рослини, які потребують регульованих кліматичних умов. Температура всередині теплиці за рахунок сонячного світла стає вищою, ніж температура навколишнього середовища назовні, захищаючи приміщення в холодну погоду. При вирощуванні сільськогосподарської продукції використовуються різні види теплиць.

Багато скляних теплиць або теплиць є високотехнологічними виробничими приміщеннями для вирощування овочів або квітів. Скляні теплиці заповнені обладнанням, включаючи екрануючі установки, опалення, охолодження, освітлення і можуть контролюватися комп'ютером для оптимізації умов росту рослин. Для оцінки оптимальності ступенів і коефіцієнта комфорту парникового мікро-клімату (тобто температури повітря, відносної вологості і дефіциту тиску пари) використовують різні методи, щоб зменшити ризик виробництва до культивування конкретної культури.

Теплиці та парники в Україні користуються великим попитом. Парники в Україні використовують на різних етапах виробництва, для підготовки розсади до сезону, для вирощування впередвесняний сезон, для отримання більшої кількості врожаїв за сезон, так як дозрівання в парнику відбувається набагато швидше. До того ж, якщо обладнати спеціалізовану теплицю, то з’явитися можливість отримувати урожай круглий рік. Виходячи з цього питання розробки систем керування процесами підтримки мікроклімату в теплицях є важливим для сільського господарства.

Теплиці поділяються на (рис. 1.2):

− кімнатні міні теплиці (невеликі споруди висотою до 1.5 м)

− теплиці дачні, які мають розміри шириною 2÷4 м, довжиною 3÷10 м і висотою від 2÷3 м, розраховані для вирощування продукції для однієї невеликої сім'ї. Такі теплиці використовуються на приватних присадибних ділянках;

− фермерські теплиці, мають розміри шириною 4÷8 м, довжиною 6÷20 м і висотою 2,5÷3,5 м. Вони використовуються круглий рік для вирощування продукції в основному під реалізацію;

− теплиці термоси, які стають одним з найперспективніших напрямків в сількогосподарський галузі, так як дозволяють значно знизити тепловтрати, а значить економити на обігріві і знижувати початкову вартість продукції;

− промислові теплиці, які мають площу не менше 500 м2. Такі об’єкти повинні мати навчений персонал і спрямовані суто на комерційну діяльність. Мають розвинену інфраструктуру і високий ступінь автоматизації. Автоматична теплиця дозволяє значно збільшити ефективність роботи самого комплексу.

Вирощування сільгосппродукції в тепличних умовах вимагає підтримки мікроклімату, до основних параметрів якого відносяться: температура і вологість повітря в теплиці та вологість і температура ґрунту.

Мікроклімат – клімат приземного шару повітря, обумовлений мікромасштабними відмінностями земної поверхні усередині місцевого клімату. Наприклад, в місцевому кліматі лісового масиву розрізняють мікроклімат лісових полян, узлісь тощо; в місцевому кліматі міста – мікроклімат площ, провулків, скверів, дворів.

Фітоклімат – атмосферні умови в середовищі поширення рослин: в травостої, в кронах дерев тощо.

З віддаленням від земної поверхні відмінності мікроклімату швидко нівелюються. Вони сильно залежать і від погоди, посилюючись в ясну тиху погоду і згладжуючись в похмуру погоду, у відсутності інсоляції та руху повітря.

Параметри мікроклімату визначаються типом вирощуваної культури. В рамках кваліфікаційної роботи розглядається процес підтримки мікроклімату кімнатної теплиці в якій вирощується сільсько-господарська культура – огірок (таб. 1.1).

Кімнатні

Дачні

Термоси

Фермерські

Промислові

Рисунок 1.2 – Класифікація теплиць

Таблиця 1.1 – Вимоги до мікроклімату при вирощуванні огірків

Сорт	Глибина висадки	Температура повітря для проростання	Температура повітря для росту в денний час	Температура повітря для росту в нічний час
Роял F1	2,5÷3 см	+12÷13 °С	+25 °С	+20 °С
Руфус F1	2,5÷3 см	+12÷13 °С	+18÷20 °С	+15 °С

Оптимальна відносна вологість повітря складає 80÷90 %, при точність підтримки ± 3 %, температура повітря 24÷25 °С вдень і 17÷18 °С вночі, температура ґрунту +20 °С.

Тривалість світлового дня 12÷16 годин на добу. Вирощування огірків в теплиці зі штучним освітленням організовується таким чином, щоб між темним і світлими періодами перепад температури повітря становив не більше 6÷8 °С. В період вегетативного росту молодим рослинам рекомендується забезпечувати вплив синьої частини спектра (довжина хвилі 400÷500 нм), а в період цвітіння і утворення зав’язі – червоною (600÷700 нм). Типова потужність освітлення складає 300 Вт/м2.

Для освітлення в сучасних теплицях використовуються світлодіодні світильники або комбіновані випромінювачі різних спектрів. За рахунок малої потужність вони майже не нагріваються і розташовується достатньо близько до рослин. Огірки в теплицях, обладнаних таким освітленням, ростуть швидше і дають більше плодів навіть з мінімальним використанням добрив.

Основна маса коренів огірків розташовується на глибині 3÷7 см від поверхні ґрунту, тому потребує достатнього зволоження. Гарний врожай потребує вологості ґрунту 90÷95 % від повної волого-ємності. Для підтримки вологості ґрунту використовується крапельне зрошення. При даному типі поливу вода потрапляє прямо до кореневої системи, дозволяючи рослинам швидко набирати силу, скорочуючи витрати води. Як правило, при поливі одночасно здійснюється і підгодівля огірків добривами. За весь вегетативний період для повноцінного догляду за огірками здійснюється 2÷3 підживлення, особливо на початку цвітіння і в період бутонізації. До початку періоду цвітіння на палив витрачається 4÷5 л/м2 водою з температурою +20÷25 °С.

1.3 Об’єкт керування

1.3.1 Загальна характеристика об’єкта керування

Корпус кімнатної теплиці виготовлено з алюмінієво-композитних панелей (АКП), які використовуються при облицюванні будівель. Панелі складаються з двох попередньо пофарбованих алюмінієвих листів товщиною до 0,5 мм, між якими розташовується середній шар – полімерна композиція на основі поліолефінів (рис. 1.3). Загальна товщина панелі складає 3÷5 мм, маса 5,5÷7.8 кг/м2. Головна перевага облицювання алюмінієвими композитними панелями є довговічність ізолюючих матеріалів.

Основні технічні характеристики алюмінієво композитних панелей:

− термічний опір 0.0103 м^2·К/Вт;

− відхилення температури 115 °С

− температурний коефіцієнт 5.54 Вт·м^2;

− температурний діапазон -50÷+80 °С.

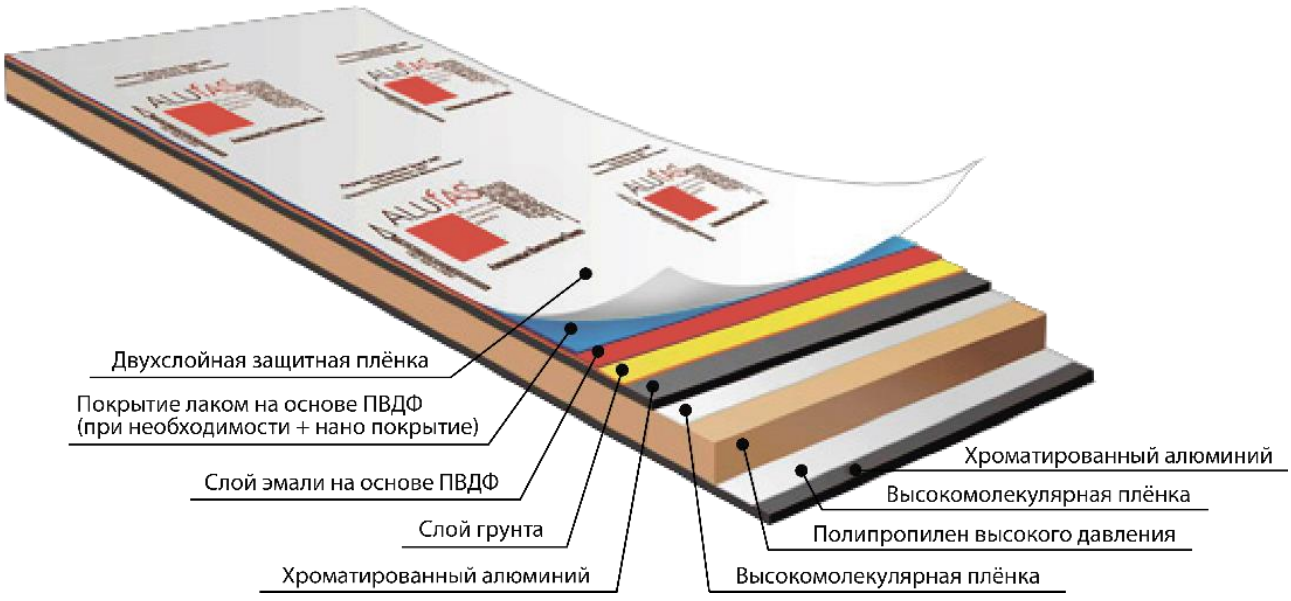


Рисунок 1.3 – Структура алюмінієво композитної панелі

Структура розробленої кімнатної теплиці наведена на рисунку 1.4.

Фізичні характеристики теплиці:

− ширина корпусу 0,42 м;

− довжина корпусу 0,84 м;

− висота корпусу 0,9 м;

– ширина контейнера з рослинами 0,027 м;

– довжина контейнера з рослинами 0,041 м;

– висота контейнера з рослинами 0,095 м

– об’єм повітря 0,32 м3;

– об’єм ґрунту 0,0004 м3;

– щільність ґрунту 0,375 кг/л.

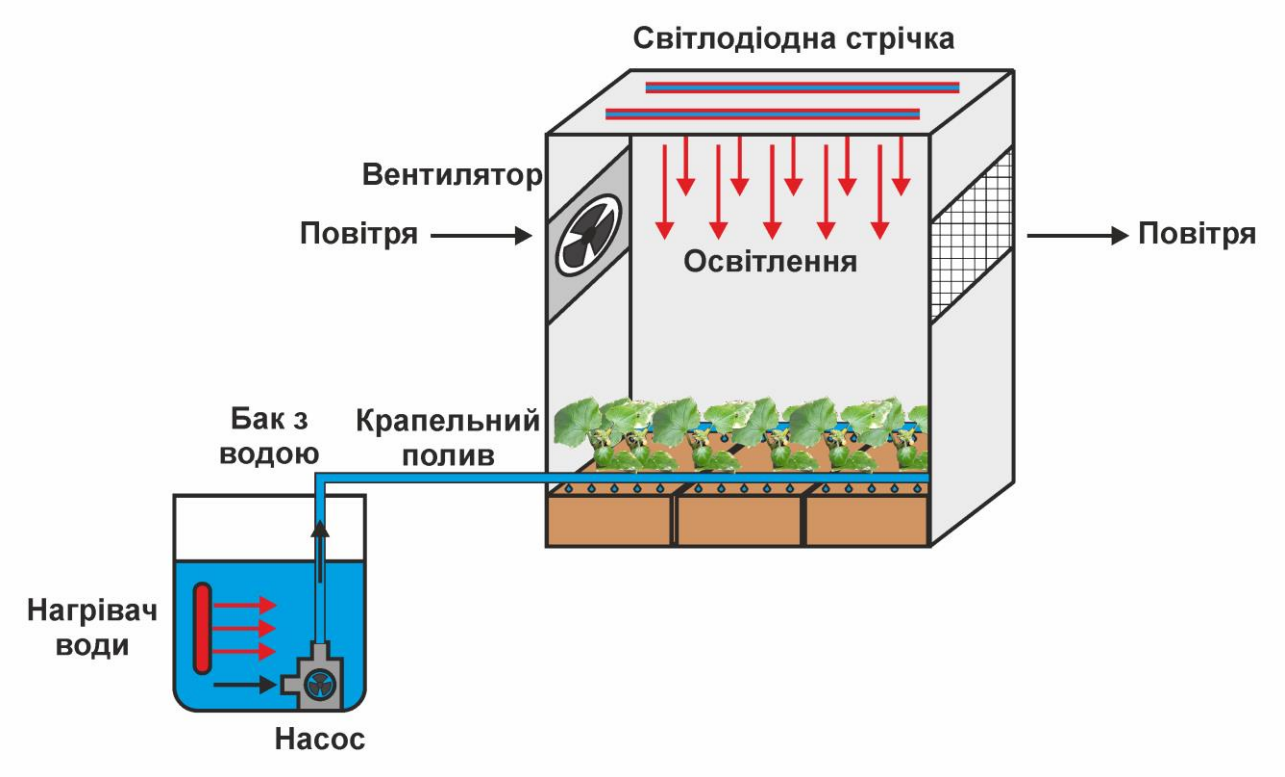


Рисунок 1.4 – Об’єкт керування

В якості ґрунту використані легка земляна суміш, яка складається з дернової землі, перегною, торфу, піску. Дана суміш забезпечує рослини азотом, фосфором, калієм та має кислотність 6÷7 pH.

1.3.2 Структура об’єкта керування

Структура об’єкту керування наведена на рисунку 1.5. В якості вхідних параметрів об’єкту керування виступають освітленість, подача повітря і води та її підігрів. В якості вихідних параметрів виступають температури повітря, води, ґрунту, вологість повітря і ґрунту, рівень води.

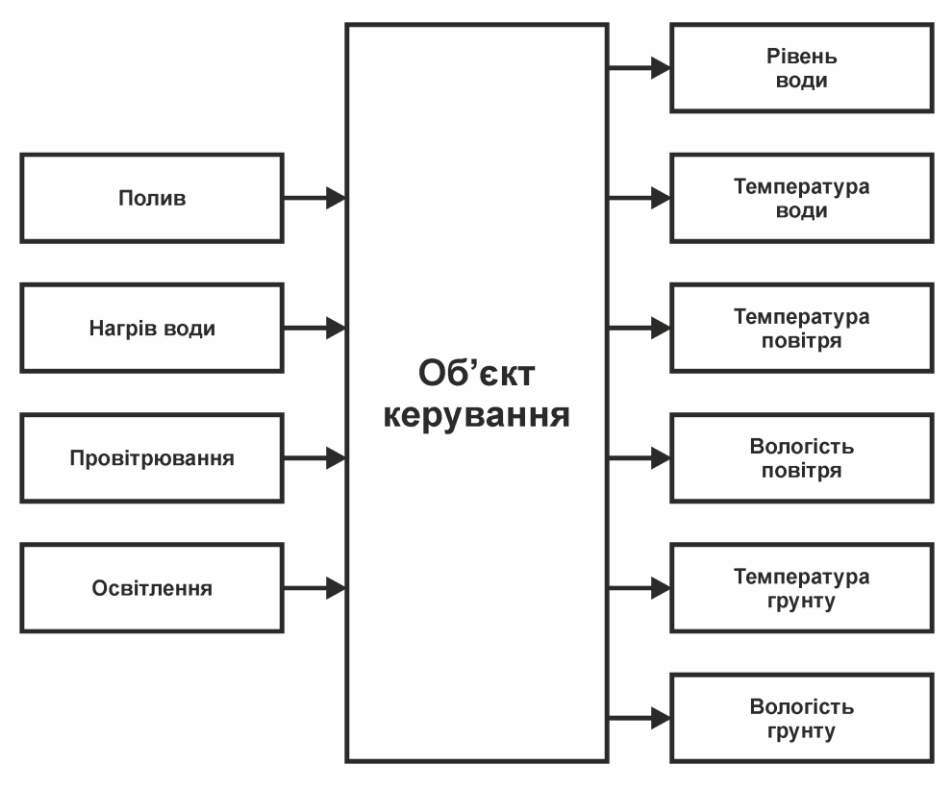


Рисунок 1.5 – Структура об’єкту керування

1.3.3 Принцип функціонування об’єкта керування

Об’єкт керування функціонує в двох режимах «ніч» (22÷6 годин) та «день» (6÷22 годин), який обираються відповідно до часу доби. В режимі «день» включається освітлення та при низькому рівні вологості ґрунту виконується полив. В режимі «ніч» виключається освітлення.

Згідно до технологічного процесу в режимі «ніч»:

− температура повітря 15÷20 °С;

− вологість повітря 70÷80 %;

− вологість ґрунту 80÷90 %.

В режимі «день»:

− температура повітря 20÷25 °С;

− вологість повітря 80÷90 %;

− вологість ґрунту 90÷95 %.

Температура води полива 20÷25 °С, температура ґрунту 15÷20 °С.

За рахунок широких діапазонів підтримки параметрів об’єкту керування їх зміну можливо виконувати за допомогою дискретних виконавчих пристроїв, таким чином об’єкт керування є дискретним і отже необхідно розробити автоматизовану систему керування процесом підтримки мікроклімату в кімнатній теплиці.

1.4 Структура системи керування

В якості об’єкта керування виступає кімнатна теплиця, для якої виконується розробка системи автоматизованого керування підтримкою мікроклімату. Вхідними параметрами об’єкта є сигнали керування підігрівом і подачею води, провітрюванням та освітленням. Вихідними параметрами об’єкту керування є рівень і температура води, температура і вологість гнуту, температура і вологість повітря.

Виходячи з цього, система керування повинна включати датчики рівня та температури води, датчики температури та вологості грунту, датчики температури та вологості повітря та пристрої керування насосом, нагрівачем води, вентилятором та освітленням (рис. 1.6).

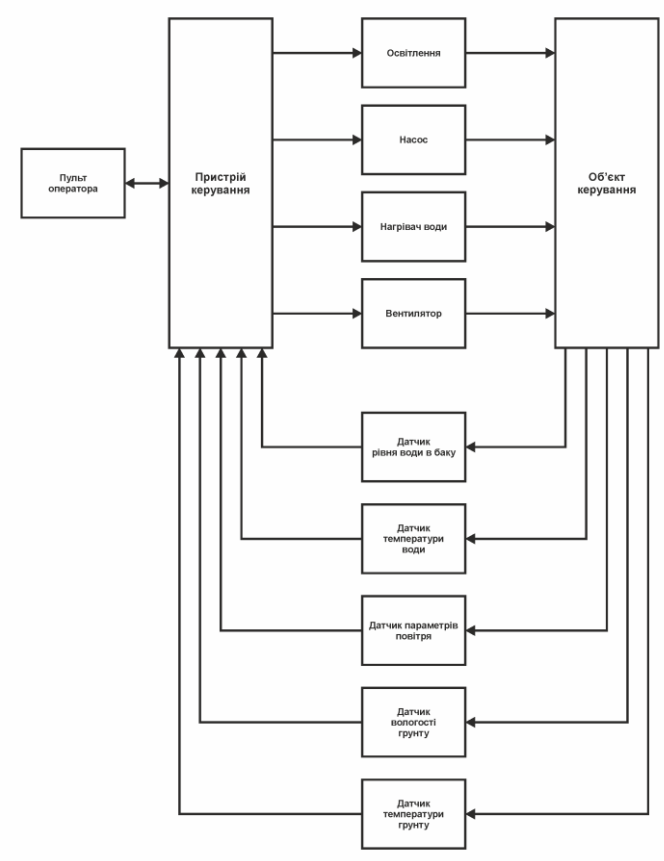


Рисунок 1.6 – Структурна схема системи керування

Виходячи з вимог система повинна забезпечувати керування об’єктом та отже включати підсистему керування технологічним обладнанням. Дана підсистема складається з пристроїв збору інформації (датчики рівня води, температури та вологості), еталонів стану обладнання (рівень включення насосу, температури та вологості), системи автоматичного контролю стану обладнання, програми керування яка повинна реалізувати керування насосом, нагрівачем, вентилятором та освітленням.

Крім того система повинна забезпечувати візуалізацію та контроль, за технологічним процесом, тому до неї повинна входити підсистема інформаційного забезпечення роботи оператора. Дана підсистема складається з реєстрації параметрів процесу, людино-машинного інтерфейсу та сигналізації досягнення параметрами заданих значень.

Також системою повинно забезпечуватися архівування технологічних процесів які відбуваються, цьому в неї повинна бути підсистема ведення архівів параметрів та подій, включно з базою даних та резервним сховищем.

Розроблена структурна схема інформаційних потоків наведена на рисунку 1.7. Дана структура забезпечує керування подачею води, температурою та вологістю, сбір даних про технологічний процес, візуалізацію отриманих даних, збереження їх в базі даних та створення їх резервних копій. Крім того дана структура забезпечує контроль, за обладнанням та сигналізацію досягнення параметрами керування заданих значень.

Таким чином згідно з вимогами підсистема автоматизованого керування технологічним обладнанням представляє собою апаратно програмний комплекс до якого входять датчики рівня води, датчики температури та вологості, об’єкт керування, пристрої керування насосом, освітленням, температурою.

Підсистема інформаційного забезпечення роботи оператора представляє собою апаратно програмний комплекс з сервером на персонального комп’ютері SCADA системи zenon Supervisor.

Підсистема ведення архівів параметрів та подій представляє собою окремий програмний модуль SCADA системи zenon.

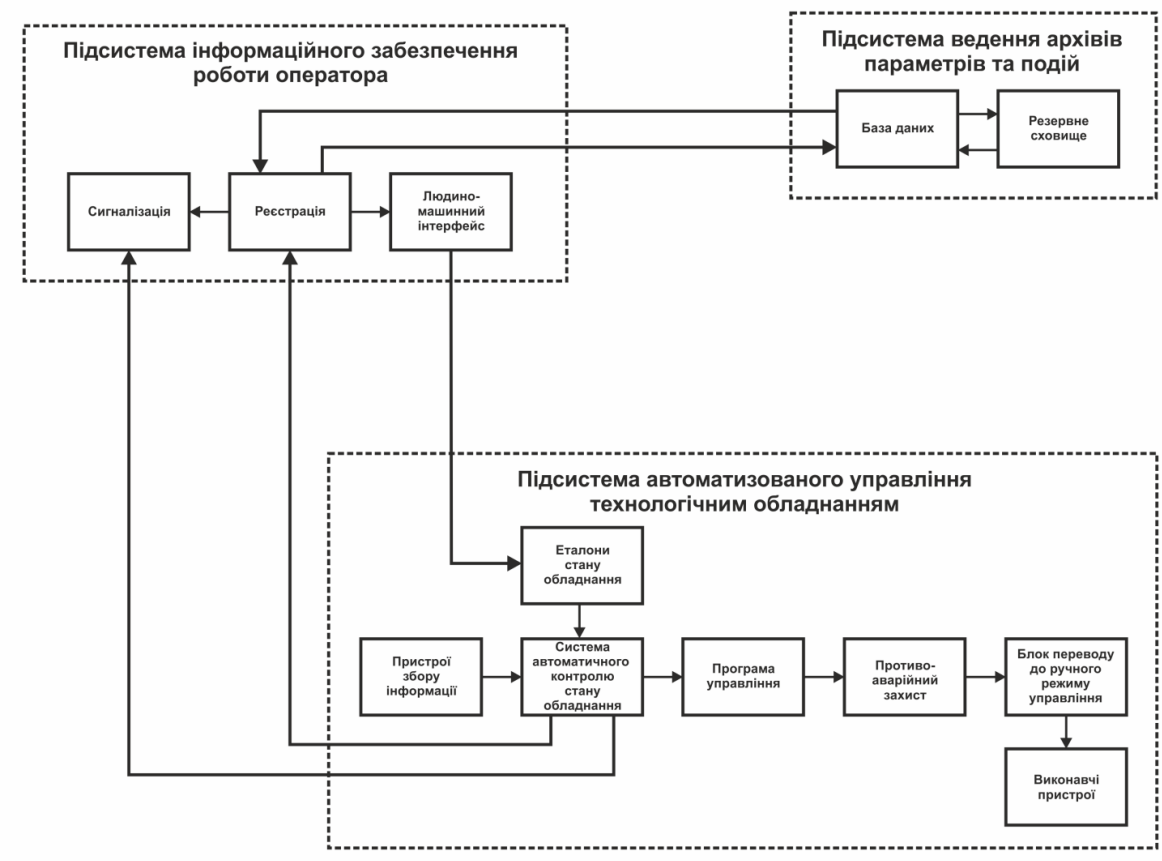


Рисунок 1.7 – Структурна схема інформаційних потоків

Зв’язок між наведеним апаратним комплексом згідно з вимогами забезпечується за допомогою інтерфейсу Wi-Fi, а між програмним забезпеченням за допомогою стандартних протоколів.

Однією з задач вирішуваних системою керування є крапельний полив паростків. Даний процес виконується водою температурою від 20÷25 °С. Для запобігання перегріву нагрівача води необхідно контролювати її рівень. Для вирішення даного завдання обрано поплавковий датчик рівня води (рис. 1.8). Технічні характеристики датчика наведені в таблиці 1.2.

Рисунок 1.8 – Датчик рівня води

Таблиця 1.2 – Технічні характеристики датчика рівня води

№	Найменування параметра	Значення
1	Тип	Релейний
2	Максимальна комутована напруга, В	~220
3	Максимальний комутований струм, А	0,5
4	Температура контрольованої середи, °С	-20÷80

Виходячи з того, що необхідно контролювати температуру води у баку, яка знаходиться в діапазоні 5÷30 °С для її вимірювання обрано датчик температури Dallas DS18B20 в герметичному виконанні та цифровим інтерфейсом 1-Wire (рис. 1.9). Технічні характеристики датчика наведені в таблиці 1.3.

Рисунок 1.9 – Датчик температури води DS18B20

Таблиця 1.3 – Технічні характеристики датчика температури води DS18B20

№	Найменування параметра	Значення
1	Тип	Аналоговий
2	Діапазон вимірюваної температури, °С	-55÷125
3	Точність вимірювання, °С	±0,5
4	Розрядність вимірювання, біт	9/12
5	Інтерфейс передачі інформації	1-Wire
6	Максимальний час перетворювання, мс	750
7	Напруга живлення, В	3÷5,5
8	Максимальний споживаний струм, мА	1,5

Обраний датчик температури також використано для вимірювання температури ґрунту.

Для вимірювання температури повітря та його вологості обрано комбінований датчик DHT22 (рис. 1.10), який є цифровим датчиком температури та вологості, що дозволяє калібрувати цифровий сигнал на виході. Складається з ємнісного датчика вологості і термістора. Також, датчик містить в собі АЦП для перетворення аналогових значень вологості і температури (табл. 1.4).

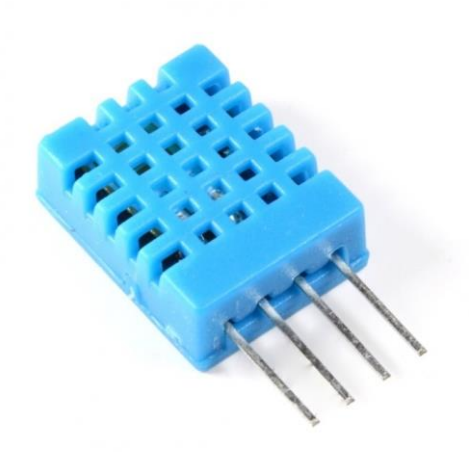


Рисунок 1.10 – Датчик температури та вологості DHT22

Таблиця 1.4 – Технічні характеристики датчика температури та вологості DHT22

№	Найменування параметра	Значення
1	Тип	Аналоговий
2	Діапазон вимірюваної температури, °С	-40÷80
3	Точність вимірювання температури, °С	±0,5
4	Діапазон вимірювання вологості, %	0÷100
5	Точність вимірювання вологості, %	±5
6	Максимальний час перетворювання, Гц	0,5
7	Напруга живлення, В	3÷5
8	Максимальний споживаний струм, мА	2,5

Для вимірювання вологості ґрунту обрано електродний датчик YL-69 який на виході видає логічний сигнал 1 або 0 відповідні до налаштувань вимірювання заданих потенціометром або аналоговий сигнал (рис. 1.11). Технічні характеристики датчика наведені в таблиці 1.5.

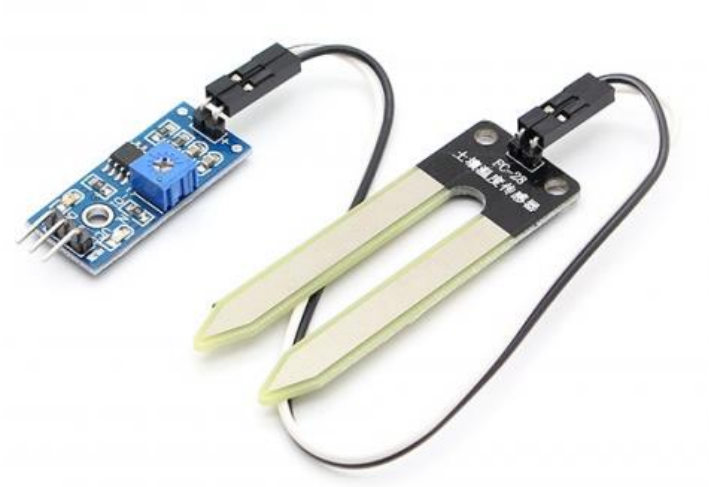


Рисунок 1.11 – Датчик вологості ґрунту YL-69

Таблиця 1.5 – Технічні характеристики датчика вологості ґрунту YL-69

№	Найменування параметра	Значення
1	Тип	Аналоговий
2	Діапазон вимірювання вологості, %	20÷90
3	Точність вимірювання вологості, %	±5
4	Максимальний час перетворювання, мс	1000
5	Напруга живлення, В	3÷5
6	Максимальний споживаний струм, мА	20

1.5 Формулювання задачі керування

Відповідно до принципу функціонування об’єкту система керування процесом підтримки мікроклімату в кімнатній теплиці повинна працювати в двох режимах ручному та автоматичному. В ручному та автоматичному режимі за допомогою людино-машинного інтерфейсу повинна забезпечуватися візуалізація температур води, повітря та ґрунту, вологості повітря та ґрунту, рівень води в ємності для поливу. Доступ до людино-машинного інтерфейсу реалізується за допомогою бездротової мережі Wi-Fi з використанням мобільного додатку операційної системі Android.

Крім того в ручному режимі користувач повинен мати можливість за допомогою інтерфейсу керувати освітленням, подачею повітря і води та її підігрівом.

Система також повинна забезпечувати функціонування в двох автоматичних режимах «ніч» і «день» та зміну між ними відповідно до часу доби. В «нічному» режимі система підтримує мікроклімат відповідно до налаштувань:

− температура повітря 15÷20 °С;

− вологість повітря 70÷80 %;

− вологість ґрунту 80÷90 %.

В свою чергу в «денному» режимі система підтримує мікроклімат:

− температура повітря 20÷25 °С;

− вологість повітря 80÷90 %;

− вологість ґрунту 90÷95 %.

При низькій вологість ґрунту здійснюється крапельний полив водою температурою 20÷25 °С за допомогою насосу. Нагрівання води виконується коли її температура падає нижче 18 °С. При нагріванні води насос не працює. При низько-

му рівні води в ємності спрацьовує попереджувальний аварійний сигнал, нагрівання не відбувається.

1.6 Висновки по розділу

За результатами аналізу технологічного процесу, об’єкта керування та структури системи керування встановлено:

– об’єктом дослідження є процес вирощування рослин в кімнатній теплиці;

– предметом дослідження є автоматизація процесу керування мікрокліматом в кімнатній теплиці;

– метою кваліфікаційної роботи є забезпечення підтримки оптимальних умов мікроклімату в кімнатній теплиці;

– об’єктом керування є кімнатна теплиця;

– вхідними параметром об’єкта керування є рівень освітленості, рівень зрошення, потужність нагріву води, швидкість повітря;

– вихідними параметром об’єкта керування є рівень води у баку, температура води, температура та вологість повітря, вологість ґрунту, температура ґрунту;

– для розробки контролера системи керування, необхідно обрати пристрій керування та його модулі;

– на підставі обраного пристрою керування та його модулів потрібно розробити контролер та виготовити його;

– для розробки моделі об’єкта керування потрібно виконати його аналіз;

– на підставі проведеного аналізу необхідно обрати метод отримання моделі об’єкта керування;

– відповідно до обраного методу потрібно виконати дослідження об’єкта керування;

– за результатами дослідження об’єкта керування необхідно розробити його модель;

– остаточна модель об’єкта керування повинна бути отримана у вигляді моделі графічному середовища імітаційного моделювання Simulink;

– при визначенні моделі об’єкта керування повинні бути враховані вимоги технологічного процесу;

– для обрання методу керування об’єктом потрібно проаналізувати технологічний процес та отриману модель об’єкта керування;

– за результатами аналізу необхідно обрати метод керування на базі котрого буде синтезовано систему керування;

– на підставі аналізу необхідно розробити модель системи керування в графічному середовища імітаційного моделювання Simulink;

– отриману систему керування необхідно дослідити на стійкість та робастність;

– з урахуванням отриманої моделі системи керування потрібно розробити програмне забезпечення системи керування для обраного пристрою керування;

– для програмного забезпечення системи керування необхідно розробити людино-машинний інтерфейс для SCADA системи zenon та перевірити його функціонування;

– потрібно розглянути економічну доцільність розробки та впровадження системи керування;

– потрібно розглянути шкідливі та небезпечні фактори у приміщенні автоматизованого робочого місця оператора та інженерно-технічні заходи щодо охорони праці.

2 ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Обрання пристрою керування

Як було зазначено раніше контролер системи керування повинен бути побудований на базі пристрою керування який має можливість підключення обраних датчиків і виконавчих пристроїв, обладнаний інтерфейсом Wi-Fi та має низьку вартість.

Даним вимогам відповідають два варіанта рішення. Перший використання одноплатного контролера Arduino Uno R3 з зовнішнім модулем Wi-Fi ESP-01 який підключається за допомогою послідовного інтерфейсу UART (рис. 2.1). Даний контролер має 14 дискретних входів/виходів, 6 аналогових та 1 інтерфейс UART (табл. 2.1).

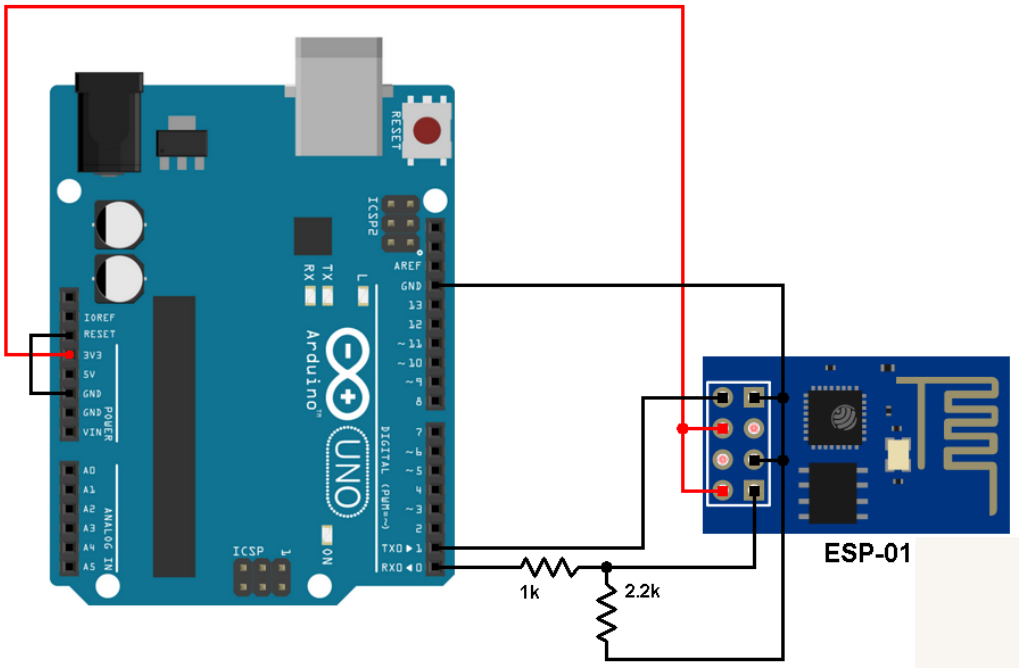


Рисунок 2.1 – Контролер Arduino Uno R3 з модулем ESP-01

Головним недоліком такого рішення є взаємодія з модулем Wi-Fi. При використанні стандартної прошивки модуля ESP-01 обмін виконується за допомогою AT інструкцій, що потребує розробки окремих програмних елементів для контролера та реалізації на їх базі протоколу який підтримується SCADA системою zenon. Крім того передача великих об’ємів інформації за рахунок цього способу є не стабільним.

Таблиця 2.1 – Технічні характеристики контролера Arduino Uno R3

№	Найменування параметра	Значення
1	Мікроконтролер	ATmega328
2	Тактова частота, МГц	16
3	Флеш-пам’ять, КБайт	32
4	ОЗП, КБайт	2
5	EEPROM, КБайт	1
6	Робоча напруга, В	5
7	Вхідна напруга, В	7-12
8	Кількість цифрових входів/виходів, шт.	14
9	Кількість аналогових входів, шт.	6
10	Максимальний вхідний струм, мА	40
11	Максимальний вихідний струм, мА	50
12	Кількість UART, шт.	1
13	Кількість I2C, шт.	1
14	Середній споживаний струм, мА	50
15	Ціна, грв.	114

Присутня можливість зміни прошивки модуля ESP-01 для використання протоколу MQTT або Modbus TCP. Однак SCADA система zenon не має прямого драйвера праці з протоколом MQTT, а стандартного варіанта підтримки Modbus TCP на сьогодні не існує. Таким чином необхідно модернізувати вже існуючу прошивку модуля, або розробити повністю нову.

Альтернативним рішенням є використання контролера повністю реалізованого на базі контролера ESP8266 та сумісного з контролером Arduino Uno R3. Таким контролером є WEMOS D1 (рис. 2.2). Даний контролер має 8 дискретних входів/виходів, 1 аналоговий та 1 Wi-Fi інтерфейс.

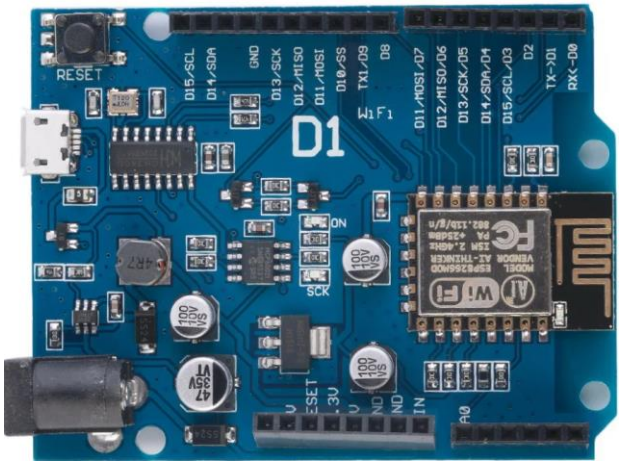


Рисунок 2.2 – Контролер WEMOS D1

Таблиця 2.2 – Технічні характеристики контролера WEMOS D1

№	Найменування параметра	Значення
1	Мікроконтролер	ES8266
2	Тактова частота, МГц	160
3	Флеш-пам’ять, КБайт	512
4	ОЗП, КБайт	32
5	EEPROM, КБайт	4
6	Робоча напруга, В	5
7	Вхідна напруга, В	7-12
8	Кількість цифрових входів/виходів, шт.	8
9	Кількість аналогових входів, шт.	1
10	Максимальний вхідний струм, мА	1
11	Максимальний вихідний струм, мА	12
12	Кількість UART, шт.	1
13	Кількість I2C, шт.	1
14	Wi-Fi інтерфейс, шт.	1
15	Середній споживаний струм, мА	100
16	Ціна, грв.	140

Контролер WEMOS D1 програмується аналогічно Arduino Uno R3, але на відміну від нього має прямий доступ до Wi-Fi інтерфейсу, що дозволяє розробляти програмне забезпечення системи керування та комунікації з зовнішніми системами як один елемент. Крім того прямий доступ до Wi-Fi інтерфейсу не потребує використання AT команд та передачі їх по інтерфейс UART. Крім того обидва контролера мають приблизно однакову ціну.

На підставі вище названих переваг у якості пристрою керування контролера системи підтримки мікроклімату в кімнатній теплиці обрано контролер WEMOS D1.

2.2 Обрання модулів пристрою керування

Виходячи з вимог до функціонування системи керування до складу контролера повинні входити релейний модуль для комутації виконавчих пристроїв, годинник реального часу та пристрій виводу стану системи керування.

Враховуючи кількість пристроїв керування у якості релейного модуля обрано 4 Relay Module (рис. 2.3). Модуль має чотири незалежні релейні канали з нор-

мально розімкненими та нормально замкнутими контактами напругою навантаження ~250 В та струмом навантаження 10 А (табл. 2.3).

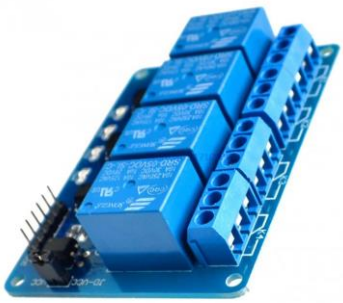


Рисунок 2.3 – Модуль 4 Relay Module

Таблиця 2.3 – Технічні характеристики модуля 4 Relay Module

№	Найменування параметра	Значення
1	Кількість каналів, шт.	4
2	Керуюча напруга, В	5
3	Керуючий струм, мА	5
4	Напруга навантаження, В	~250
5	Струм навантаження, А	10

У якості модуля годинника реального часу обрано Real Time Clock на базі мікросхеми DS3231SN (рис. 2.4) високої точності з термокомпенсацією ходу, зовнішнім елементом живлення CR2032 та інтерфейсом I2C (табл. 2.4).

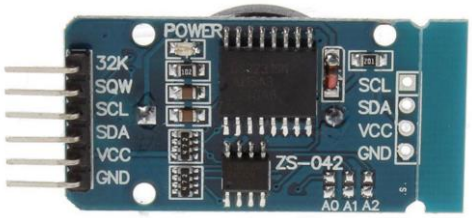


Рисунок 2.4 – Модуль Real Time Clock

Таблиця 2.4 – Технічні характеристики модуля Real Time Clock

№	Найменування параметра	Значення
1	Стабільність генератора 0÷+40 С, імп./хв.	±2
2	Стабільність генератора -40÷+85 С, імп./хв.	±3
3	Інтерфейс	I2C
4	Робоча напруга, В	3,0÷5,5
5	Споживаний струм у сплячому режимі, мкА	1,0
6	Споживаний струм, мА	1,0

У якості пристрою виводу стану системи керування обрано рідкокристалічний індикатор LCD1602 з контролером PCF8574 який реалізує інтерфейс I2C (рис. 2.5). Індикатор має підсвічування, два ряди, кожний ряд складається з 16 символів (табл. 2.5).

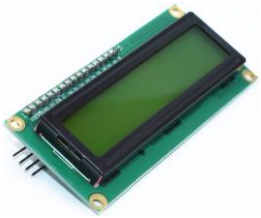


Рисунок 2.5 – Індикатор LCD1602

Таблиця 2.5 – Технічні характеристики індикатор LCD1602

№	Найменування параметра	Значення
1	Кількість рядів, шт.	2
2	Кількість символів у ряду, шт.	16
3	Підсвічування	Жовте
4	Колір символів	Чорний
5	Робоча напруга, В	5
6	Споживаний струм, мА	5,5

2.3 Обрання пристроїв живлення

Виходячи з обраного апаратного забезпечення в системі керування використовуються три типи напруги +5 В, +12 В та ~220 В. Від +5 В живиться контролер, його модулі та датчики, від +12 В живиться освітлення, насос та вентилятор, від ~220 В живиться нагрівач води. Згідно з цим необхідно обрати два блока живлення.

Споживаний струм контролера, його модулів та датчиків:

$$I = 150 + 4 \cdot 5 + 1 + 5{,}5 + 1 + 0{,}5 + 2 \cdot 1{,}5 + 2{,}5 + 20 = 150 \text{ (мА)}. \qquad (2.1)$$

Відповідно до (2.1) у якості пристрою живлення обрано блок живлення CHD0510 з напругою живлення 5 В та максимальним струмом 1000 мА (рис. 2.6).

Рисунок 2.6 – Блок живлення CHD0510

Потужність споживана виконавчими пристроями:

$$P = 30 + 15 + 10 = 55 \text{ (Вт)}. \quad (2.2)$$

Згідно з (2.2) в якості блока живлення обрано PWR110, з напруго живлення 12 В та потужністю 60 Вт (рис. 2.7).

Рисунок 2.7 – Блок живлення PWR110

2.4 Схема електрична принципова системи керування

На підставі обраного апаратного забезпечення розроблено схему електричну принципову системи керування (рис. 2.8).

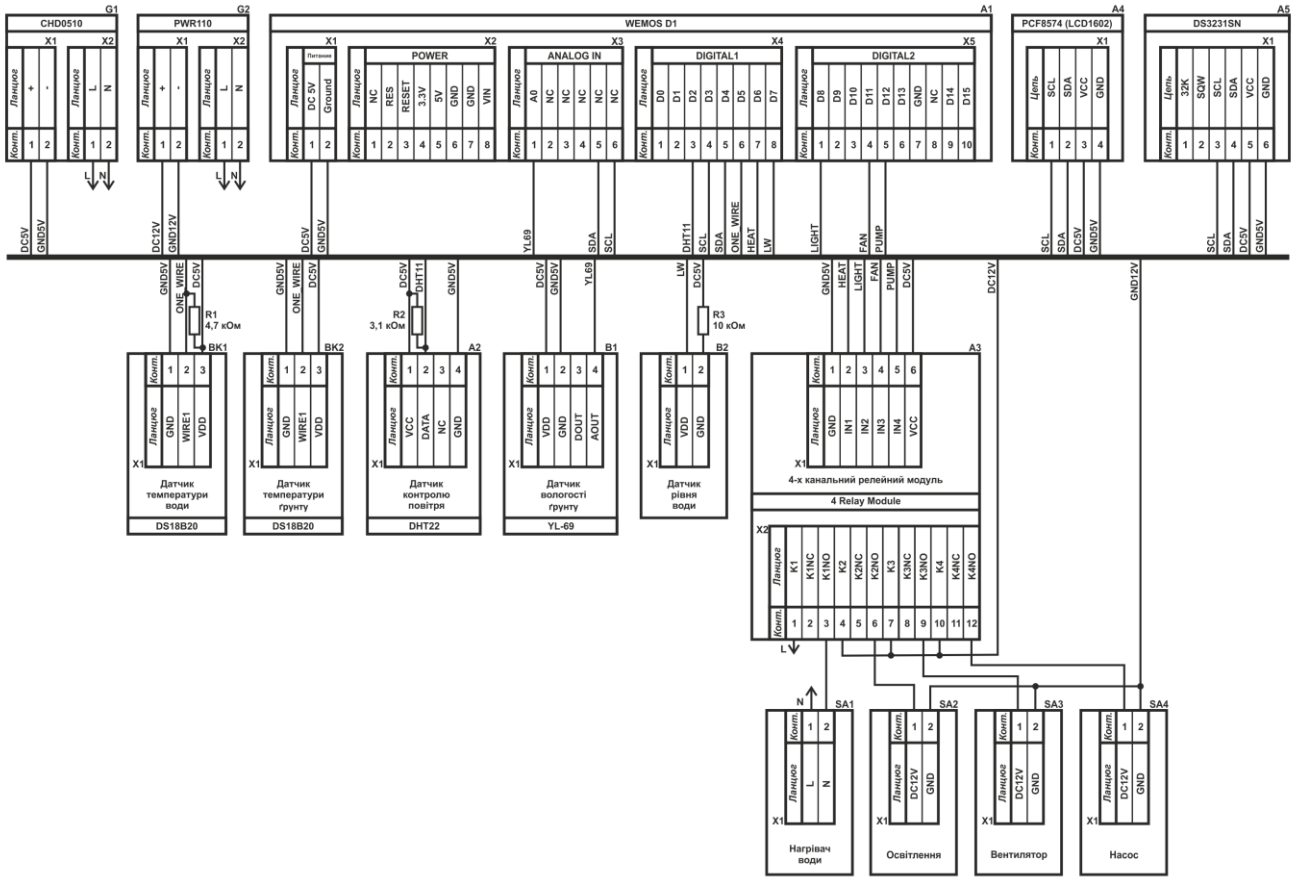


Рисунок 2.8 – Схема електрична принципова системи керування

В якості пристрою керування використано контролер WEMOS D1 (A1), який живиться від 5 В блока живлення CHD0510 (G1). До контролера підключаються модуль рідкокристалічного індикатора (A4) і модуль годинника реального часу (A5) за допомогою інтерфейсу I2C та чотирьох канальний релейний модуль (A3).

Релейний модуль (A3) комутує напругу ~220 В для нагрівача води (SA1) і напругу 12 В від блока живлення PWR110 (G2) для світлодіодного освітлення (S?-2), вентилятора (SA3) та насоса (SA4).

Датчики температури води (BK1) та ґрунту (BK2) підключаються до контролера (A1) через дискретний вхід/вихід. Обмін даними між контролером та датчиками виконується за допомогою програмного інтерфейсу 1-Wire.

Датчик контролю повітря (A2) вимірює температуру і вологість та підключається до контролера (A1) через дискретний вхід/вихід. Обмін даними між контролером та датчиком виконується за допомогою послідовного програмного інтерфейсу.

Датчик вологості ґрунту (B1) підключається до аналогового входу контролера (A1), за допомогою плати узгодження, а датчик рівня води (B2) підключається через струмообмежувальний резистор (R3) до дискретного входу/виходу.

2.5 Розробка контролера системи керування

Для створення контролера системи керування мікрокліматом в кімнатній теплиці було обрано корпус Z74 (рис. 2.9) розмірами 176 х 126 х 57,4 мм.

Рисунок 2.9 – Корпус контролера

В корпусі було розмішено одноплатний контролер WEMOS D1 з платою розширення Arduino Prototype Shild, модулем годинника реального часу Real time clock, релейним модулем 4 Relay Module, платою перетворювача датчика вологості ґрунту YL-69, платою підключення датчиків, та клемником підключення виконавчих пристроїв з напругою живлення 12 В (рис. 2.10).

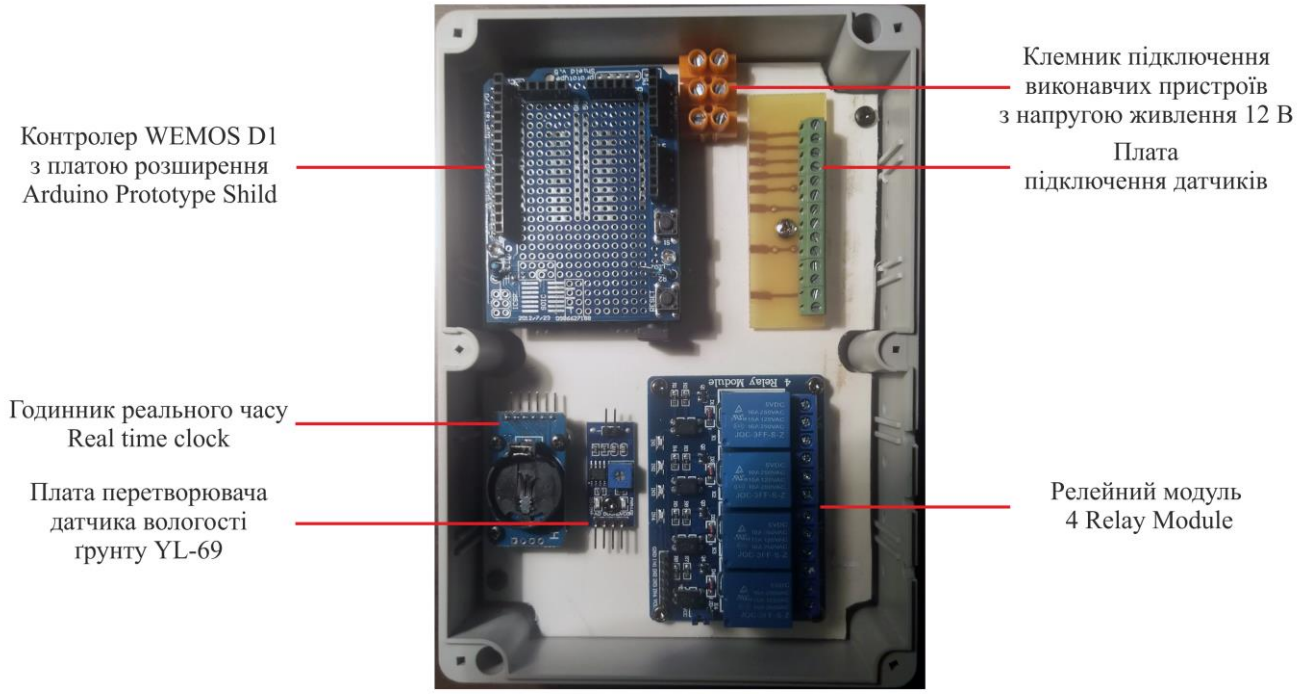


Рисунок 2.10 – Розмішення елементів контролера

Усі елементи між собою були підключені за допомогою шлейфів та пайки оловом. На передній поверхні контролера розмішено рідкокристалічний індикатор з контролером PCF8574 який реалізує інтерфейс I2C (рис. 2.11).

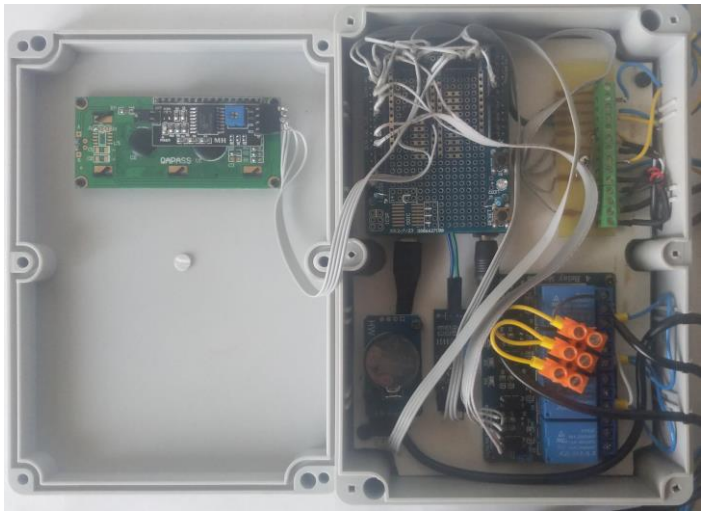


Рисунок 2.11 – Підключення елементів контролера

До контролера були підключені датчики та блоки живлення (рис. 2.12).

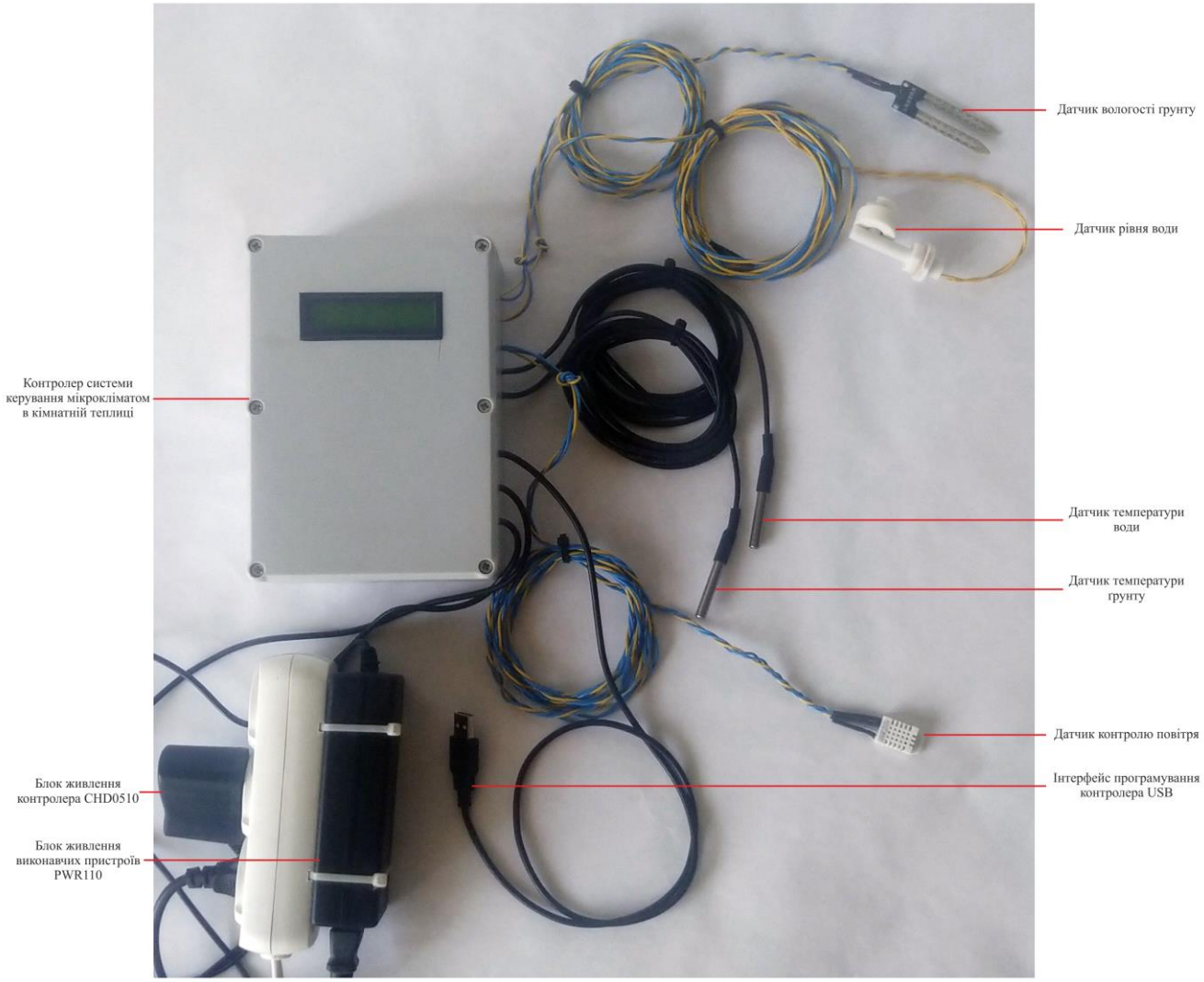


Рисунок 2.12 – Контролер системи керування мікрокліматом в кімнатній теплиці

2.6 Висновки по розділу

1. Відповідно до вимог у якості пристрою керування обрано контролер WEMOS D1 з інтерфейсами Wi-Fi і I2C, 1 аналоговим входом та 8 дискретними входами/виходами сумісний з контролером Arduino Uno R3 за моделями та середою розробки програмного забезпечення.

2. Для контролера обрано модуль годинника реального часу Real time clock з інтерфейсом I2C, релейний модуль 4 Relay Module, 2 рядковий 16 символьний рідкокристалічний індикатор з контролером PCF8574 і інтерфейсом I2C та два блока живлення +5 В і +12 В.

3. На підставі обраного апаратного забезпечення розроблено схему електричну принципову системи керування за якою виготовлено плату підключення датчиків та обрано плату розширення Arduino Prototype Shield. Згідно з схемою електричною принциповою виготовлено контролер системи керування мікрокліматом в кімнатній теплиці.

4. Отриманий контролер з підключеними датчиками подалі буде використано при розробці програмного забезпечення системи керування та перевірки його функціонування.

3 СИНТЕЗ СИСТЕМ КЕРУВАННЯ

Відповідно до завдання виконується розробка систему керування мікрокліматом в кімнатній теплиці. У даній системі безперервним є процес підтримки температури в теплиці. Виходячи з цього потрібно розробити модель об’єкта керування, синтезувати систему керування та дослідити її функціонування.

3.1 Модель об’єкта керування

В якості об’єкта керування виступає кімнатна теплиця ізольована від кімнати за допомогою алюмінєво композитних панелей. Обмін повітрям між теплицею та кімнатою виконується за допомогою отвору вентилятора та вентиляційного отвору. Таким чином йде постійний обмін повітрям між теплицею та кімнатою. Температура повітря в кімнаті складає 20 °С, вологовміст 7,492 г/кг, об’єм повітря в теплиці 0,32 м^3. Витрата повітря природна 0,64 м^3/год, витрата при працюючому вентиляторі 6,4 м^3/год. Температура повітря в теплиці складає 22 °С, вологовміст 12,719, потужність освітлення 30 Вт, втрати тепла 5 Вт, прихід вологості за рахунок зрощення $2.5 \cdot 10^{-5}$ л/с.

Таким чином в теплицю постійно притікає холодне повітря яке заміщує тепле повітря та змішується з ним. Виходячи з цього потрібно розрахувати параметри зміни температури, вологовмісту та ентальпії повітря в теплиці.

Усі наведені параметри залежать від зміни об’єму повітря, яка відповідає витраті повітря за 1 секунду:

$$\Delta V = \frac{G}{3600}, \tag{3.1}$$

де ΔV – зміна об’єму повітря (м^3), G – витрата повітря (м^3/год).

Зміна параметрів повітря залежить від співвідношення кімнатного повітря та повітря в теплиці:

$$k = \frac{\Delta V}{V - \Delta V}, \tag{3.2}$$

де k– співвідношення кількості кімнатного повітря та повітря в теплиці, V – об’єм повітря в теплиці (м^3).

Враховуючи (3.2) зміна вологовмісту повітря в теплиці є різницею:

$$\Delta d = k \cdot (d_{\text{ТП}} - d_{\text{КП}}), \tag{3.3}$$

де Δd – зміна вологовмісту повітря (кг/кг), $d_{\text{ТП}}$ – вологовміст повітря в теплиці (кг/кг), $d_{\text{КП}}$ – вологовміст кімнатного повітря (кг/кг).

Зміна ентальпії повітря в теплиці є різницею:

$$\Delta I = k \cdot (I_{\text{ТП}} - I_{\text{КП}}), \tag{3.4}$$

де ΔI – зміна ентальпії повітря (кДж/кг), $I_{\text{ТП}}$ – ентальпія повітря в теплиці (кДж/кг), $I_{\text{КП}}$ – ентальпія кімнатного повітря (кДж/кг).

Зміна температури повітря в теплиці є різницею:

$$\Delta t = \frac{\Delta I - l_p \cdot \Delta d}{C_{\text{СП}} + \text{C}_{\text{П}} \cdot \Delta d}, \tag{3.5}$$

де Δt – зміна температури повітря (°С), l_p – питома теплота пароутворення при 0 °С (кДж/кг), $C_{\text{СП}}$ – теплоємність сухого повітря 1,005 (кДж/(кг·°С)), $\text{C}_{\text{П}}$ – теплоємність пару 1,8 (кДж/(кг·°С)).

Для (3.4) ентальпія повітря розраховується зворотно пропорційно до температури:

$$I = (C_{\text{СП}} + \text{C}_{\text{П}} \cdot \Delta d) \cdot t + l_p \cdot \Delta d, \tag{3.6}$$

де I – ентальпія повітря (кДж/кг).

Відповідно до аналітичного опису розрахунку зміни температури, вологовмісту та ентальпії повітря в кімнатній теплиці в математичному пакеті MATLAB розроблено функцію для графічного середовища імітаційного моделювання Simulink:

```
function [dG, dt, dd, dI] = airMix(fan, G1, G1f, t1, d1, V, t2, d2)

% fan - Робота вентилятора
% G1 - Витрата повітря, м3/год
% G1f - Витрата повітря вентилятором, м3/год
% t1 - Температура кімнатного повітря, °C
% d1 - Вологовміст, г/кг
% V - Об'єм повітря в теплиці, м3
% t2 - Температура повітря, °C
% d2 – Вологовміст повітря в теплиці, г/кг
% dG - Зміна об'єма повітря в теплиці, м3
% dt - Зміна температури повітря в теплиці, °C
% dd - Зміна вологовміста повітря в теплиці, кг/кг
% dI - Зміна ентальпії повітря в теплиці, кДж/кг

cycle = 0.01; % Час виклику функції, с
```

```
d1 = d1 / 1000; % г/кг перетворення до кг/кг
d2 = d2 / 1000; % г/кг перетворення до кг/кг

if (fan == 0) % Вентилятор вимкнено
    dG = G1; % Природна витрата повітря, м3/год
else % Вентилятор включено
    dG = G1f * fan;
    if (dG < G1)
        dG = G1; % Мінімальне значення витрати повітря для вентилятора, м3/год
    end
end

dV = dG / (3600 / cycle); % Зміна об'єму повітря, м3/с
k = min(dV, V - dV) / max(dV, V - dV); % Співвідношення повітря

I1 = (1.005 + 1.8 * d1) * t1 + 2500 * d1; % Ентальпія повітря в кімнаті, кДж/кг
I2 = (1.005 + 1.8 * d2) * t2 + 2500 * d2; % Ентальпія повітря в теплиці, кДж/кг

dd = (d1 - d2) * k; % Зміна вологовмісту повітря, кг/кг
dI = (I1 - I2) * k; % Зміна ентальпії повітря, кДж/кг
dt = (dI - 2500 * dd) / (1.005 + 1.8 * dd); % Зміна температури повітря, °C
```

Таким чином температуру повітря в теплиці, вологовміст та ентальпія:

$$t_{\text{Т}} = t_{\text{Т}(-1)} + \Delta t,$$
$$d_{\text{Т}} = d_{\text{Т}(-1)} + \Delta d + d_3, \qquad (3.7)$$
$$I_{\text{Т}} = I_{\text{Т}(-1)} + \Delta I,$$

де $t_{\text{Т}}, t_{\text{Т}(-1)}$ – температура повітря в теплиці на поточному та попередньому інтервалі часу (°С), $d_{\text{Т}}, d_{\text{Т}(-1)}$ – вологовміст повітря в теплиці на поточному та попередньому інтервалі часу (кг/кг), $I_{\text{Т}}, I_{\text{Т}(-1)}$ – ентальпія повітря в теплиці на поточному та попередньому інтервалі часу (кДж/кг).

Крім заміщення повітря в теплиці виконується його підігрів за рахунок освітлення та втрата тепла через огорожу таким чином необхідно розрахувати зміну кількості тепла у повітря з урахуванням його маси та ентальпії. Для чого потрібно визначити відносну вологість повітря та його щільність.

Відносна вологість повітря:

$$\varphi = \frac{B \cdot d_{\text{Т}}}{(d_{\text{Т}} \cdot P_{\text{НП}} + 0{,}623 \cdot P_{\text{НП}})}, \qquad (3.8)$$

де φ – відносна вологість повітря, B – барометричний (атмосферний) тиск 101325 (Па), $P_{\text{НП}}$ – тиск насиченого пара (Па).

Тиск насиченого пару:

$$P_{\text{НП}} = 479 + (11.52 + 1.62 \cdot t_{\text{Т}})^2. \qquad (3.9)$$

Щільність вологого залежить від відносної вологості та температури:

$$\rho = \rho_{\text{СП}} - \frac{P}{R \cdot T} \cdot (M_{\text{СП}} - M_{\text{П}}), \tag{3.10}$$

де $\rho_{\text{СП}}$ – щільність сухого повітря (кг/м^3), Р – Парціальний тиск водяної пари (Па), R – універсальна газова стала 8,314, кДж/(моль·К), T – температура повітря (К), $M_{\text{СП}}$ – молярна вага сухого повітря (кг/моль), $M_{\text{П}}$ – молярна вага водяного пара (кг/моль).

Щільність сухого повітря залежить від барометричного тиску, парціального тиску та температури:

$$\rho_{\text{СП}} = \frac{M_{\text{СП}} \cdot P}{R \cdot T}. \tag{3.11}$$

Парціальний тиск водяної пари залежить від температури та відносної вологості:

$$P = \varphi \cdot P_{\text{НП}}. \tag{3.12}$$

Температура повітря для розрахунку щільності повітря:

$$T = t_{\text{Т}} + 273. \tag{3.13}$$

Маса повітря в теплиці є добутком щільності вологого повітря та об’єму повітря:

$$m_{\text{Т}} = \rho \cdot V, \tag{3.14}$$

де $m_{\text{Т}}$ – маса повітря в теплиці (кг).

Таким чином кількість теплоти повітря в теплиці з урахуванням освітлення та втрат:

$$Q_{\text{Т}} = m_{\text{Т}} \cdot I_{\text{Т}} + P_{\text{О}} - Q_{\text{В}}, \tag{3.15}$$

де $Q_{\text{Т}}$ – кількість теплоти повітря в теплиці (кДж), $P_{\text{О}}$ – потужність освітлення (кВт), $Q_{\text{В}}$ – втрати тепла (кДж).

Ентальпія отримана на підставі кількості теплоти:

$$I_{\text{Т}} = \frac{Q_{\text{Т}}}{m_{\text{Т}}}. \tag{3.16}$$

Виходячи з отриманого значення ентальпії температура повітря визначається відповідно до (3.5), а відносна вологість повітря відповідно до (3.8).

Відповідно до аналітичного опису розрахунку температури, вологовмісту та відносної вологості повітря в кімнатній теплиці в математичному пакеті MATLAB

розроблено функцію для графічного середовища імітаційного моделювання Simulink:

```
function [tOut, dOut, fi] = greenHouse(dG, dt, dd, dI, V, t, d, lP, Qv, pd)

% dG - Зміна об'єма повітря в теплиці, м3
% dt - Зміна температури в теплиці, °C
% dd - Зміна вологовмісту в теплиці, кг/кг
% dI - Зміна ентальпії повітря в теплиці, кДж/кг
% V - Об'єм повітря в теплиці, м3
% t - Температура повітря в теплиці, °C
% d - Вологовміст повітря в теплиці, г/кг
% lP - Потужність освітлення, Вт
% Qv - Втрати енергії через ізолятор, Вт
% pd - Приплив вологи за рахунок зрошення, г/с
% tOut - Температура повітря в теплиці, °C
% dOut - Вологовміст повітря в теплиці, г/кг
% fi - Відносна вологість повітря в теплиці, %

persistent sI; % Ентальпія повітря в теплиці, кДж/кг
persistent st; % Температура повітря в теплиці, °C
persistent sd; % Вологовміст повітря в теплиці, кг/кг
if (isempty(sI))
    sd = d / 1000; % Початковий вологовміст, г/кг перетворення до кг/кг
    sI = (1.005 + 1.8 * sd) * t + 2500 * sd; % Початкова ентальпія, кДж/кг
    st = t; % Початкова температура, °C
end

cycle = 0.01; % Час виклику функції, с

sI = sI + dI; % Врахування зміни ентальпії в теплиці, кДж
st = st + dt; % Врахування зміни температури в теплиці, °C
sd = sd + dd + pd * cycle; % Врахування зміни вологовмісту, кг/кг

B = 101325; % Атмосферний тиск, Па
Pnp = 479 + (11.52 + 1.62 * st)^2; % Тиск насиченого водяного пару, Па
fi = (B * sd) / (sd * Pnp + 0.623 * Pnp); % Відносна вологість, %

Msv = 29; % Молярна маса сухого повітря, кг/моль
Mp = 18; % Молярна маса водяної пари, кг/моль
R = 8.314 * 10^3; % Універсальна газова стала, Дж / (моль * К)

P = fi * Pnp; % Парціальний тиск водяної пари, Па
T = t + 273; % Температура повітря в теплиці, К

rosv = (B * Msv) / (R * T); % Щільність сухого повітря, кг/м3
Ro = rosv - P / (R * T) * (Msv - Mp); % Щільність вологого повітря, кг/м3
m = Ro * V; % Маса повітря в теплиці, кг
sQ = m * sI; % Кількість теплоти повітря в теплиці, кДж

% Врахування зміни кількості теплоти повітря в теплиці, кДж
sQ = sQ + lP / 1000 * cycle - Qv / 1000 * cycle;
sI = sQ / m; % Ентальпія повітря в теплиці, кДж/кг

Csv = 1.005; % Теплоємність сухого повітря, кДж/(кг*К)
Cp = 1.8; % Теплоємність водядого пара, кДж/(кг*К)
lp = 2500; % Питома теплота пароутворення при 0 °C, кДж/кг
Cv = Csv + Cp * sd; % Теплоємність повітря, кДж/(кг*К)
st = (sI - lp * sd) / Cv; % Температура, °C

Pnp = 479 + (11.52 + 1.62 * st)^2; % Тиск насиченого водяного пару, Па
fi = (B * sd) / (sd * Pnp + 0.623 * Pnp); % Відносна вологість, %
```

```
tOut = st; % Остаточна температура повітря в плиці, °C
dOut = sd * 1000; % Остаточний вологовміст, кг/кг перетворення до г/кг
```

На підставі отриманих моделей в графічному середовищі імітаційного моделювання Simulink математичного пакету MATLAB розроблено модель об’єкта керування (рис. 3.1).

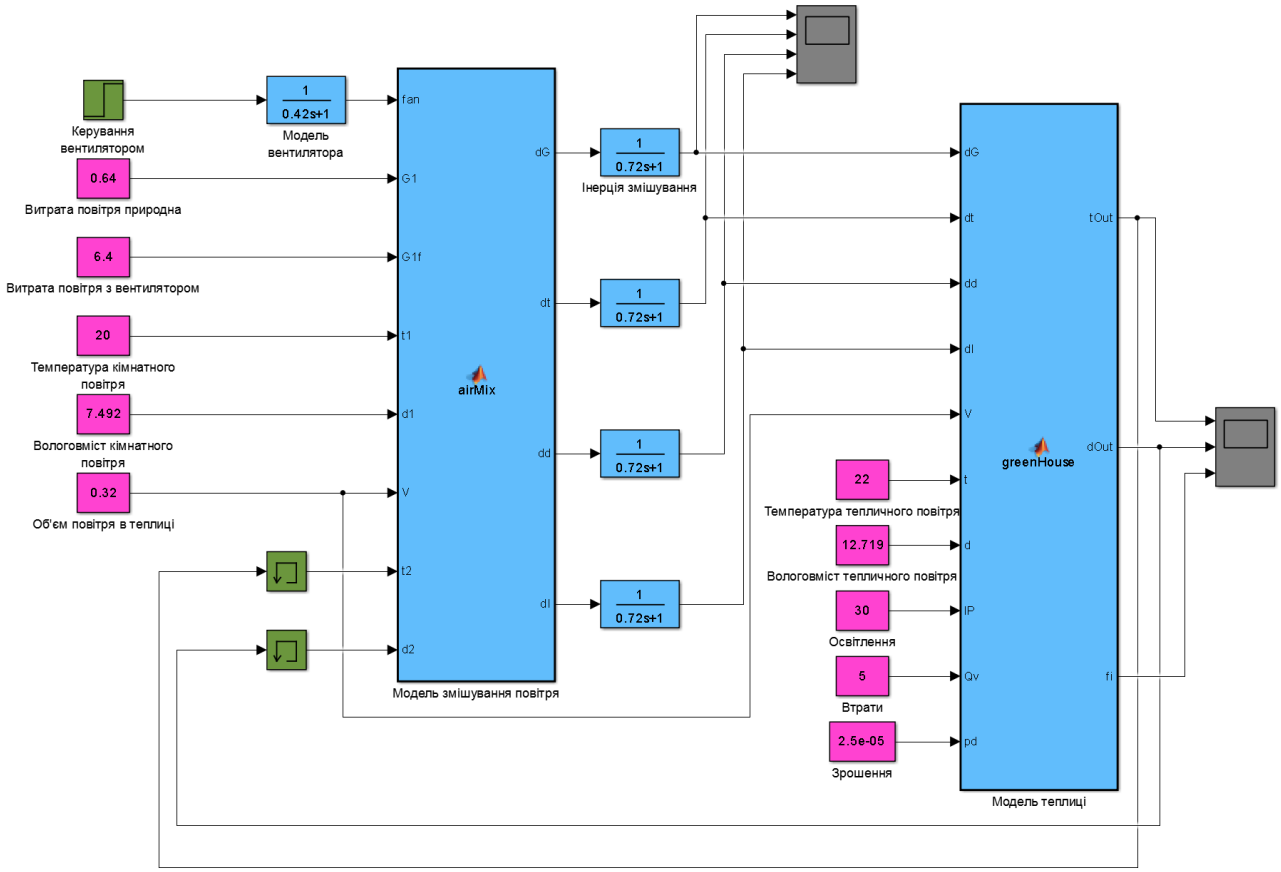


Рисунок 3.1 – Модель об’єкта керування

Виходячи з технічних характеристик вентилятор постійного струму має час розгону 2 с, на підставі чого для опису зміни потоку обрано передавальну функцію:

$$W_{\text{В}}(s) = \frac{1}{0{,}42s + 1}, \tag{3.17}$$

де $W_{\text{В}}(s)$ – передавальна функція вентилятора.

Виходячи з об’єму теплиці 0,32 м^3 та максимальної витрати повітря 6,4 м^3/год в якості передавальних функцій зміщення повітря обрано:

$$W_{\text{З}}(s) = \frac{1}{0{,}72s + 1}, \tag{3.18}$$

де $W_{\text{З}}(s)$ – передавальна функція змішування.

В моделі для реалізації передавальних функцій змішення використані блоки з начальними параметрами, з налаштуваннями відповідними до початкових параметрів моделі теплиці та моделі змішування повітря.

Перевірка функціонування об’єкта керування показала (рис. 3.2), що модель функціонує відповідно до фізичних закономірностей. Початкова температура повітря в телиці складає 22 °С, вологовміст 12,7 г/кг, відносна вологість 75 %. Робота вентилятора який подає повітря з кімнати температурою 20 °С та вологовмістом 7,492 призводить до зменшення температури в теплиці до 20,4 °С, а вологовмісту до 12 г/кг. Відповідно до зменшення температури зменшується тиск насиченої пари і відносна вологість збільшується до 77,5 %. Наявність різниці між параметрами повітря в квартирі та теплиці викликані освітленням та крапельним поливом.

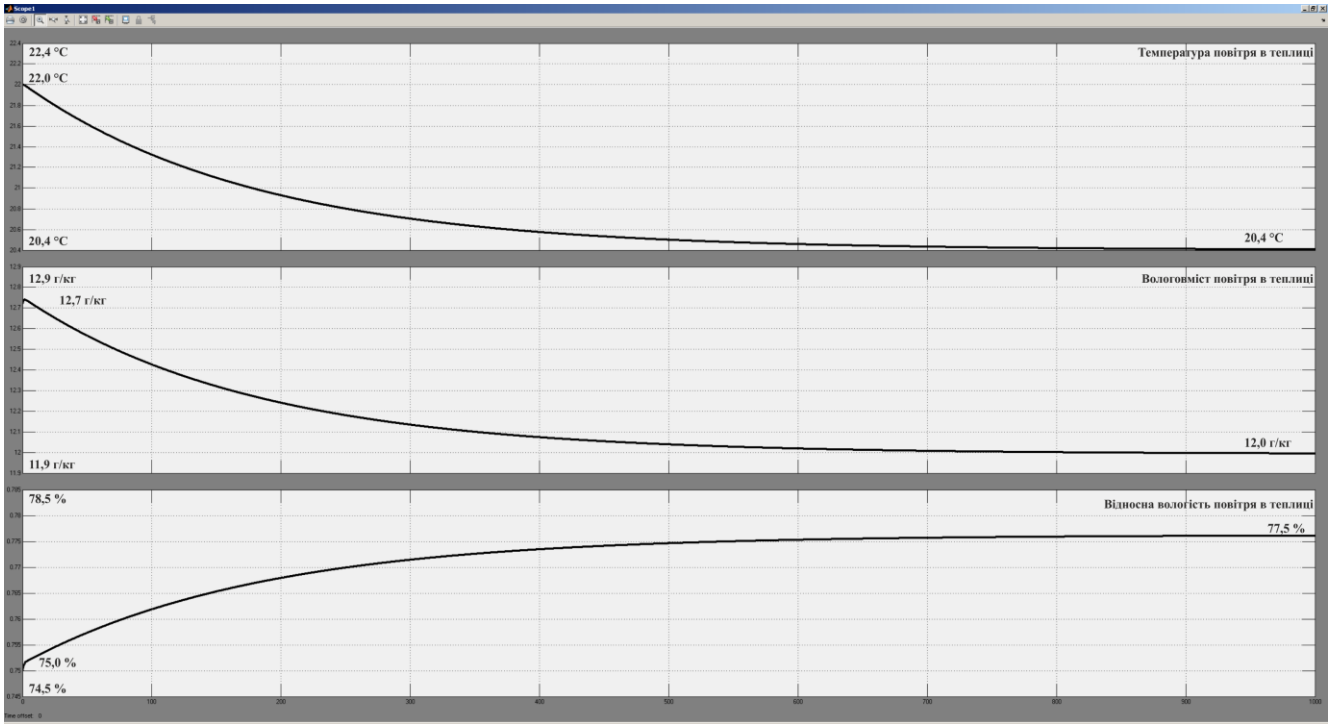


Рисунок 3.2 – Моделювання об’єкта керування

3.2 Синтез системи керування

Відповідно до завдання система керування мікрокліматом в кімнатній теплиці повинна підтримувати температуру повітря на рівні 22 °С у межах ±2 °С. Враховуючи наявність збурень таких як зміна температури в кімнаті, потужності

освітлення, температури води крапельного паливу, продуктивності насосу система повинна контролювати температуру повітря. Даним вимогам відповідає структура системи керування з від'ємним зворотнім зв'язком.

Типовим рішенням для теплових об'єктів є використання в якості регулятора в системі з від'ємним зворотнім зв'язком пропорційного, пропорційно-інтегрального або пропорційно-інтегрально-диференційного регулятора. Враховуючи вимоги до системи керування про відсутність наявності статичної похибки відносно до діапазону керування та роботу системи керування в режимі стабілізації коли відсутня швидка зміна параметрів зовнішнього середовища в якості пристрою керування обрано пропорційно-інтегральний регулятор.

На підставі обраних рішень розроблено модель системи керування мікрокліматом в кімнатній теплиці на базі пропорційно-інтегрального регулятора (рис. 3.3).

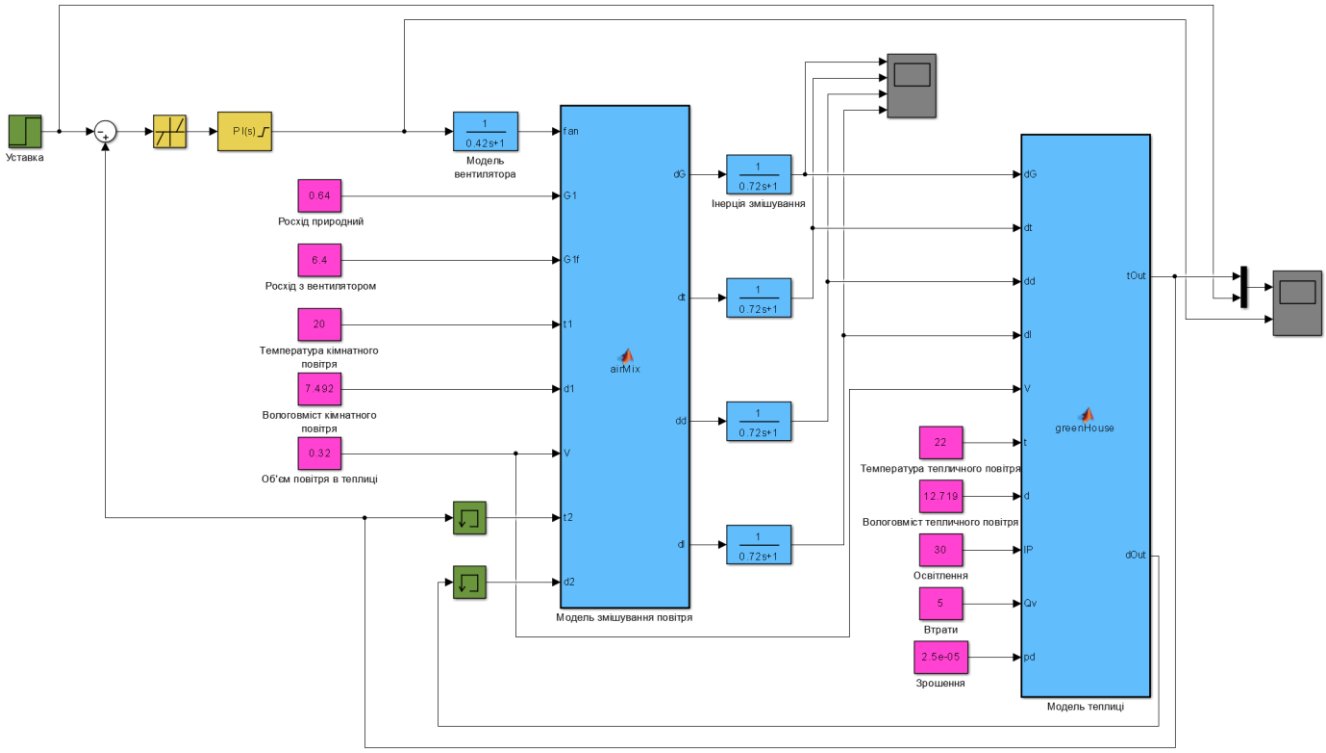


Рисунок 3.3 – Модель системи керування

В моделі використано пропорційно-інтегральний регулятор з обмеженням керуючого впливу від 0 до 1, методом запобігання перенасичення інтегральної складової "Защіпка", та зменшеною зоною керування за помилкою ±0,5 °C.

Початкове налаштування пропорційно-інтегрального регулятора виконано емпірично за результатами моделювання, що пов’язано з складністю моделі об’єкта керування та таким чином неможливістю отримання його апроксимації динамічними ланками і використання розрахункових методів ($K = 0{,}25, Ti = 100$ (с)).

За результатами моделювання встановлено (рис. 3.4), що система керування функціонує відповідно до вимог та забезпечує находження керуючого впливу в заданих межах. Система починає функціонувати при досягненні температурою рівня 22,5 °С, що пов’язано з наявністю зони нечутливості.

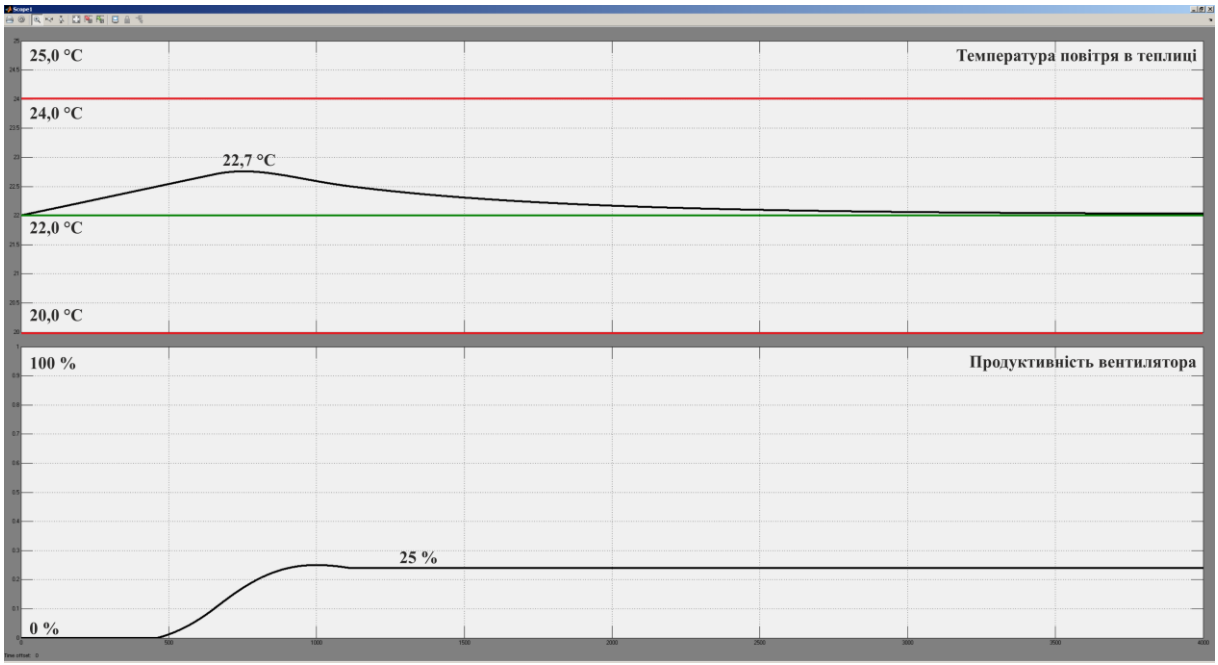


Рисунок 3.4 – Моделювання системи керування

Таким чином отримана система керування задовольняє вимогам технологічного процесу та бути використана у подальших дослідженнях з метою обґрунтування параметрів регулятора.

3.3 Обґрунтування обрання параметрів регулятора

Враховуючи неможливість обрання параметрів регулятора аналітичним шляхом виконано ряд експериментів з їх варіюванням у широких діапазонах для чого використовувалося програмне забезпечення Analysis Work v1.1. На підставі отриманих даних проведено дослідження їх впливу на якість функціонування сис-

теми керування. В якості показників використані перерегулювання та інтегральна оцінка квадрату помилки керування.

Для візуалізації та обробки даних було використано функцію аналізу написану на мові MATLAB:

```
function AnalyzeK(fileName)

    fprintf('\nАналіз впливу коефіцієнта підсилення v1.0\n\n');

    load(fileName);
    figure('Name', 'Аналіз впливу коефіцієнта підсилення');
    hold('on');
    setPoint = data.modeledData.data.parameters.K.setPoint.values(1);
    startTime = data.modeledData.data.parameters.K.processValues(1).Time(1);
    stopTime = data.modeledData.data.parameters.K.processValues(1).Time(end);
    plot([startTime, stopTime], [setPoint, setPoint], 'LineWidth', 3.0, 'Color',
[0.0, 0.7, 0.0]);

    result = struct();
    result.K = data.modeledData.data.parameters.K.parameter.values;
    result.overshoot = zeros(1, length(result.K));
    result.integral = zeros(1, length(result.K));
    for nIndex = 1 : length(data.modeledData.data.parameters.K.processValues)
        plot(data.modeledData.data.parameters.K.processValues(nIndex),
'LineWidth', 3.0, 'Color', [0.1 * nIndex /
length(data.modeledData.data.parameters.K.processValues) * 10, 0.0, 0.0]);
        result.overshoot(nIndex) =
max(data.modeledData.data.parameters.K.processValues(nIndex).Data) - setPoint;
        result.integral(nIndex) =
trapz((data.modeledData.data.parameters.K.processValues(nIndex).Data -
setPoint).^2);
    end
    hold('off');
    ylim([22 23]);
    fprintf('\nКоефіцієнт підсилення:\n'); fprintf('%.3f\t', result.K);
    fprintf('\nПеререгулювання:\n'); fprintf('%.3f\t', result.overshoot);
    fprintf('\nІнтегральна оцінка:\n'); fprintf('%.2d\t', result.integral);
end
```

Функція завантажує файл даних виводить отримані дані у вигляді графіків та розраховує абсолютне перерегулювання та інтегральну оцінку. На графіках найменшому значенню коефіцієнта підсилення відповідає чорний колір, а найбільшому червоний.

Варіювання коефіцієнта підсилення від 1 до 10 призвело до виникнення статичної помилки керування (рис. 3.5, табл. 3.1), на підставі цього були отримані дані для коефіцієнтів підсилення від 0,1 до 1,0 (рис. 3.6, табл. 3.2). Виходячи з цих даних інтегральна оцінка зменшується з зменшенням коефіцієнта підсилення, а таким чином якість функціонування системи керування покращується. Найменше значення інтегральної оцінки відповідає діапазону від 0,2 до 0,5 згідно з цим отримані дані (рис. 3.7, табл. 3.3).

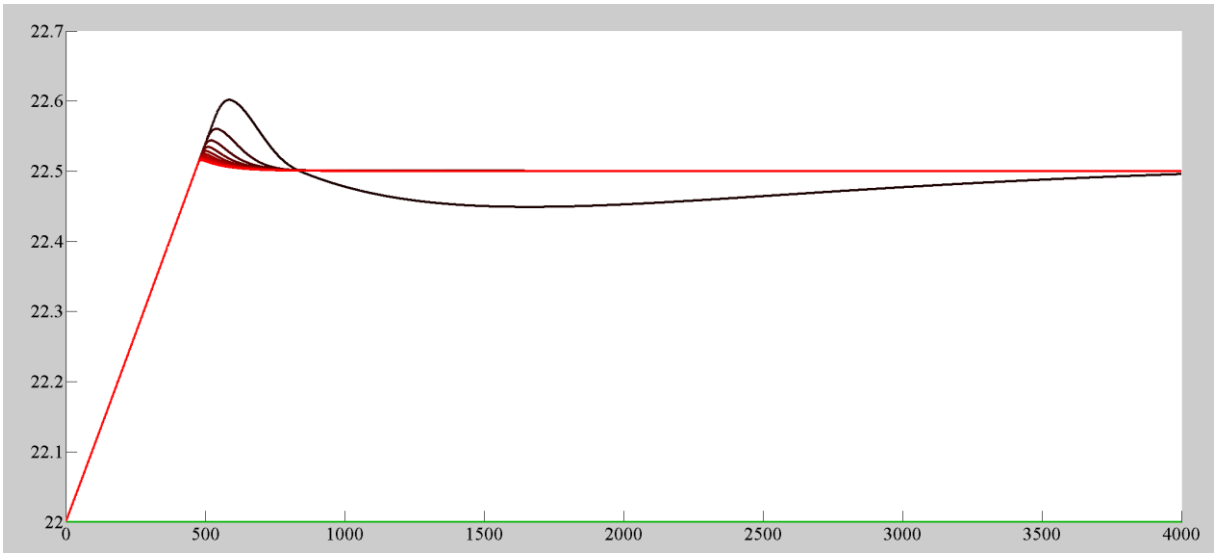


Рисунок 3.5 – Зміна коефіцієнта підсилення від 1 до 10

Таблиця 3.1 – Показники якості коефіцієнта підсилення

K	1	2	3	4	5	6	7	8	9	10
Перерегулювання, °C	0,602	0,560	0,544	0,535	0,529	0,525	0,522	0,519	0,517	0,516
Інтег. оцінка, 10^4	8,53	9,33	9,29	9,28	9,27	9,26	9,25	9,25	9,25	9,25

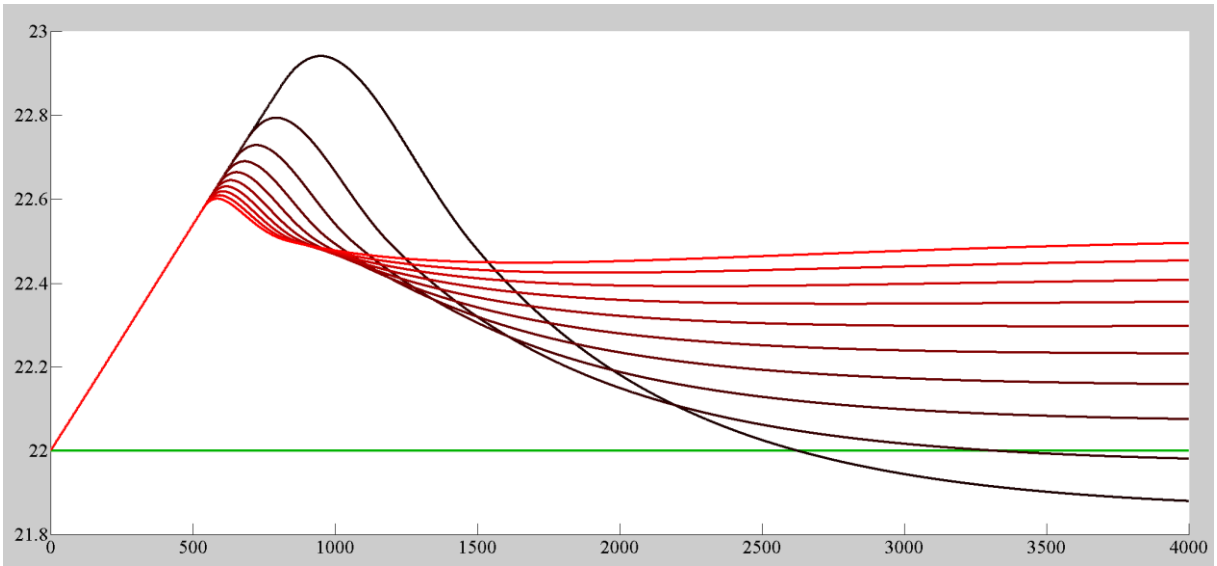


Рисунок 3.6 – Зміна коефіцієнта підсилення від 0,1 до 1,0

Таблиця 3.2 – Показники якості коефіцієнта підсилення

K	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
Перерегулювання, °C	0,942	0,795	0,729	0,691	0,665	0,645	0,631	0,619	0,610	0,602
Інтег. оцінка, 10^4	7,08	4,54	4,05	4,21	4,70	5,37	6,12	6,92	7,73	8,53

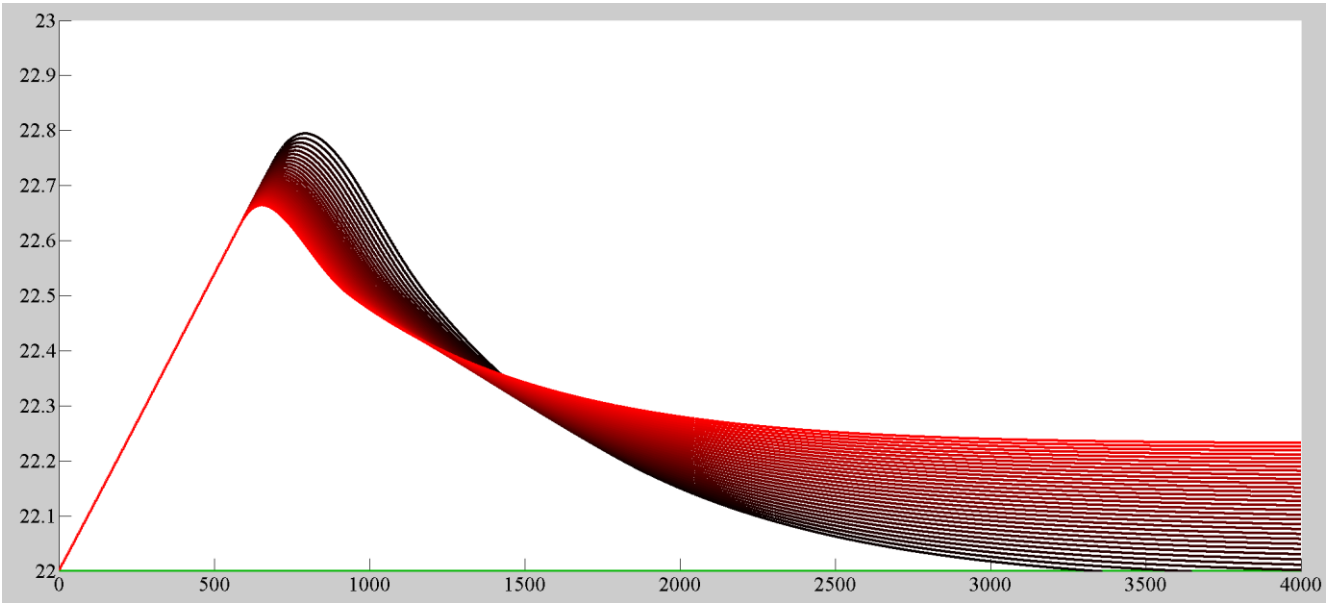


Рисунок 3.7 – Зміна коефіцієнта підсилення від 0,2 до 0,5

Таблиця 3.3 – Показники якості коефіцієнта підсилення

K	0,200	0,210	0,220	0,230	0,240	0,250	0,260	0,270	0,280	0,290
Перерегулювання, °C	0,795	0,786	0,778	0,770	0,763	0,757	0,751	0,745	0,739	0,734
Інтег-на оцінка, 10^4	4,54	4,44	4,36	4,28	4,22	4,17	4,13	4,10	4,07	4,06
K	0,300	0,310	0,320	0,330	0,340	0,350	0,360	0,370	0,380	0,390
Перерегулювання, °C	0,729	0,725	0,720	0,716	0,712	0,708	0,704	0,701	0,697	0,694
Інтег-на оцінка, 10^4	4,05	4,04	4,04	4,05	4,06	4,08	4,10	4,12	4,15	4,18
K	0,400	0,410	0,420	0,430	0,440	0,450	0,460	0,470	0,480	0,490
Перерегулювання, °C	0,691	0,688	0,685	0,682	0,679	0,676	0,674	0,671	0,669	0,667
Інтег. оцінка, 10^4	4,21	4,25	4,29	4,34	4,38	4,43	4,48	4,53	4,59	4,65

Для більш наочного аналізу розроблено функцію візуалізації отриманих даних:

```
function ShowAnalyze()

    fprintf('\nЗалежності показників якості від коефіцієнта підсилення v1.0\n\n');

    K = [0.200    0.210 0.220 0.230 0.240 0.250 0.260 0.270 0.280 0.290 0.300 0.310
      0.320 0.330 0.340 0.350 0.360 0.370 0.380 0.390 0.400 0.410 0.420 0.430 0.440
      0.450 0.460 0.470 0.480 0.490 0.500];
    Overshoot = [0.795  0.786 0.778 0.770 0.763 0.757 0.751 0.745 0.739 0.734 0.729
      0.725 0.720 0.716 0.712 0.708 0.704 0.701 0.697 0.694 0.691 0.688 0.685 0.682
      0.679 0.676 0.674 0.671 0.669 0.667 0.665];
    Integral = [4.54e+04      4.44e+04    4.36e+04    4.28e+04    4.22e+04
      4.17e+04    4.13e+04    4.10e+04    4.07e+04    4.06e+04    4.05e+04
      4.04e+04    4.04e+04    4.05e+04    4.06e+04    4.08e+04    4.10e+04
      4.12e+04    4.15e+04    4.18e+04    4.21e+04    4.25e+04    4.29e+04
      4.34e+04    4.38e+04    4.43e+04    4.48e+04    4.53e+04    4.59e+04
      4.65e+04    4.70e+04];

    figure();
    plot(K, Overshoot, 'LineWidth', 3.0, 'Color', 'k');
    xlabel('Коефіцієнт підсилення');
    ylabel('Перерегулювання, °C');
    figure();
```

```
    plot(K, Integral, 'LineWidth', 3.0, 'Color', 'k');
    xlabel('Коефіцієнт підсилення');
    ylabel('Інтегральна оцінка');

    fprintf('Мінімальне перерегулювання\n');
    [~, nIndex] = min(Overshoot);
    fprintf('K: %.3f\n', K(nIndex));
    fprintf('Перерегулювання: %.3f\n', Overshoot(nIndex));
    fprintf('Інтегральна оцінка: %d\n', Integral(nIndex));

    fprintf('\nМінімальна інтегральна оцінка\n');
    [~, nIndex] = min(Integral);
    fprintf('K: %.3f\n', K(nIndex));
    fprintf('Перерегулювання: %.3f\n', Overshoot(nIndex));
    fprintf('Інтегральна оцінка: %d\n', Integral(nIndex));
end
```

В результаті використання функції отримані графіки (рис. 3.8) та найкращі показники якості.

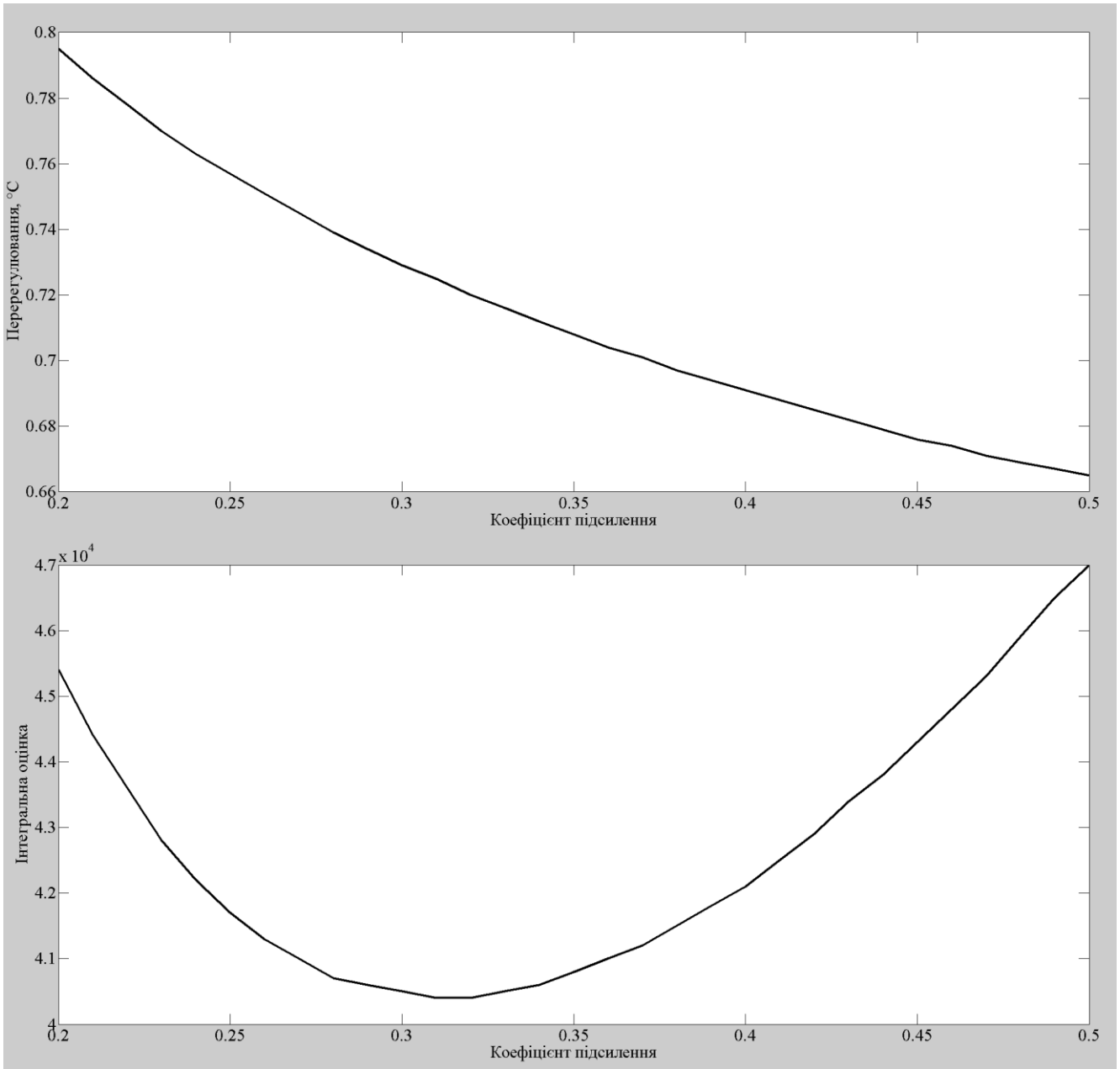


Рисунок 3.8 – Залежність показників якості від коефіцієнта підсилення

```
Залежності показників якості від коефіцієнта підсилення v1.0

Мінімальне перерегулювання
K: 0.500
Перерегулювання: 0.665
Інтегральна оцінка: 47000

Мінімальна інтегральна оцінка
K: 0.310
Перерегулювання: 0.725
Інтегральна оцінка: 40400
```

Відповідно до отриманих результатів мінімальній інтегральній оцінці відповідає коефіцієнт підсилення 0,31. Враховуючи те, що перерегулювання в даному випадку має менш вагомий пріоритет, так як максимальна перерегулювання складає 2, а отримане 0,725 та те, що показники якості для раніше отриманого коефіцієнта підсилення 0,3 близькі до найкращих, в якості коефіцієнта підсилення обрано останній.

Варіювання час інтегрування від 10 до 500 призвело до виникнення статичної помилки керування (рис. 3.93.5, табл. 3.4), на підставі цього були отримані дані для часу інтегрування від 50 до 140 (рис. 3.10, табл. 3.5). Виходячи з цих даних інтегральна оцінка зменшується з зменшенням часу інтегрування, однак при часі інтегрування меншім за 80 усталений режим стає меншим за уставку. Найменше значення інтегральної оцінки при позитивному перерегулювання відповідає діапазону від 75 до 85 згідно з цим отримані дані (рис. 3.11, табл. 3.6).

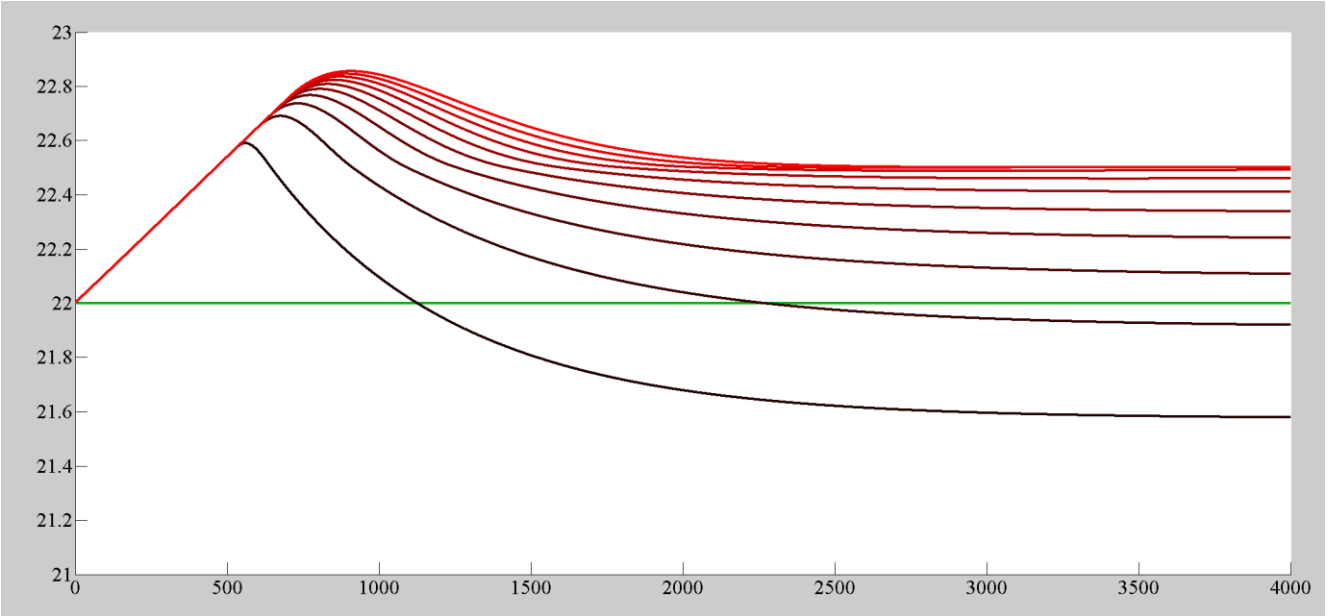


Рисунок 3.9 – Зміна часу інтегрування від 10 до 500 с

Таблиця 3.4 – Показники якості часу інтегрування

Ti	10	60	110	160	210	260	310	360	410	460
Перерегу-ня, °C	0,591	0,691	0,737	0,767	0,790	0,808	0,823	0,836	0,846	0,856
Інтег. оцінка, 10^4	4,80	2,83	4,43	6,44	8,31	9,89	10,11	10,20	10,26	10,31

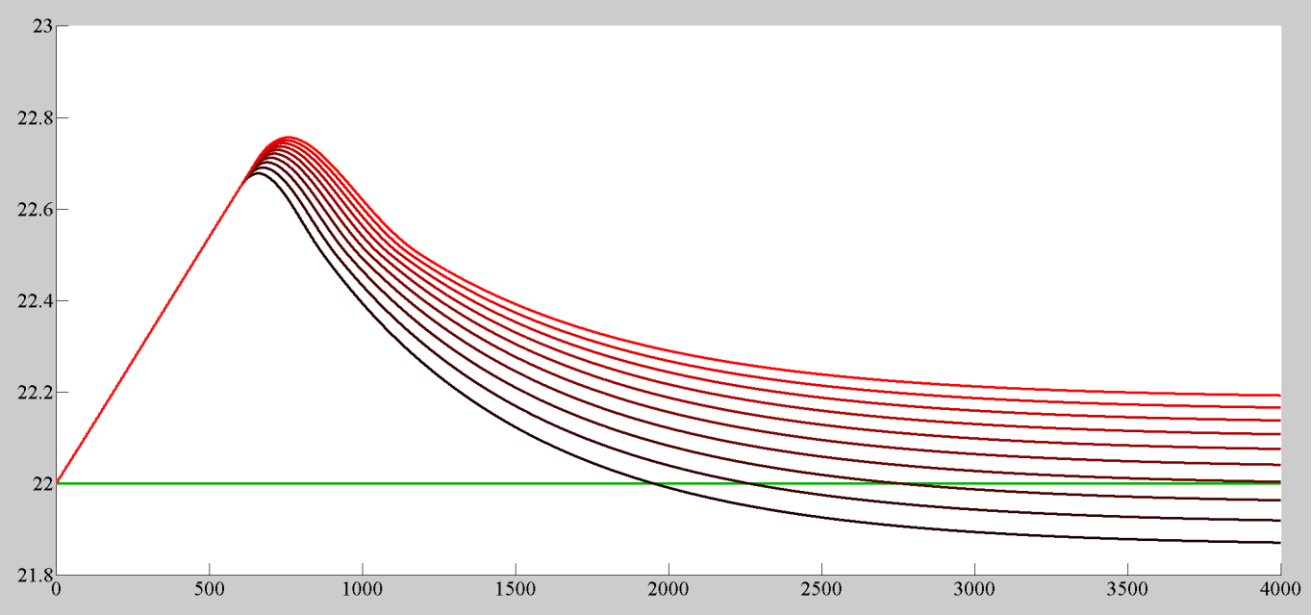


Рисунок 3.10 – Зміна часу інтегрування від 50 до 140 с

Таблиця 3.5 – Показники якості часу інтегрування

Ti	50	60	70	80	90	100	110	120	130	140
Перерегу-ня, °C	0,678	0,691	0,702	0,712	0,721	0,729	0,737	0,744	0,750	0,756
Інтег. оцінка, 10^4	2,69	2,83	3,06	3,35	3,68	4,05	4,43	4,83	5,23	5,63

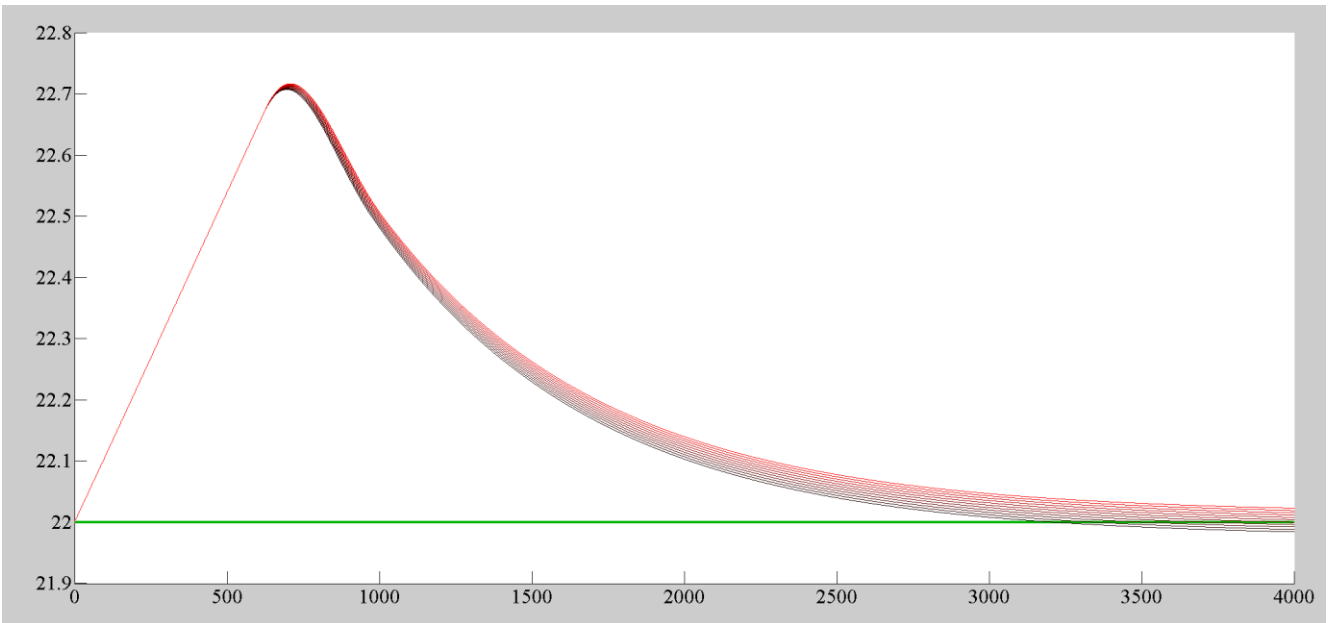


Рисунок 3.11 – Зміна часу інтегрування від 75 до 85 с

Таблиця 3.6 – Показники якості часу інтегрування

Ti	75	76	77	78	79	80	81	82	83	84
Перерегу-ня, °C	0.707	0.708	0.709	0.710	0.711	0.712	0.713	0.714	0.715	0.716
Інтег. оцінка, 10^4	3.20	3.23	3.26	3.29	3.32	3.35	3.38	3.41	3.45	3.48

На підставі отриманих даних побудовані графіки залежності перерегулювання та інтегральної оцінки від часу інтегрування (рис. 3.12).

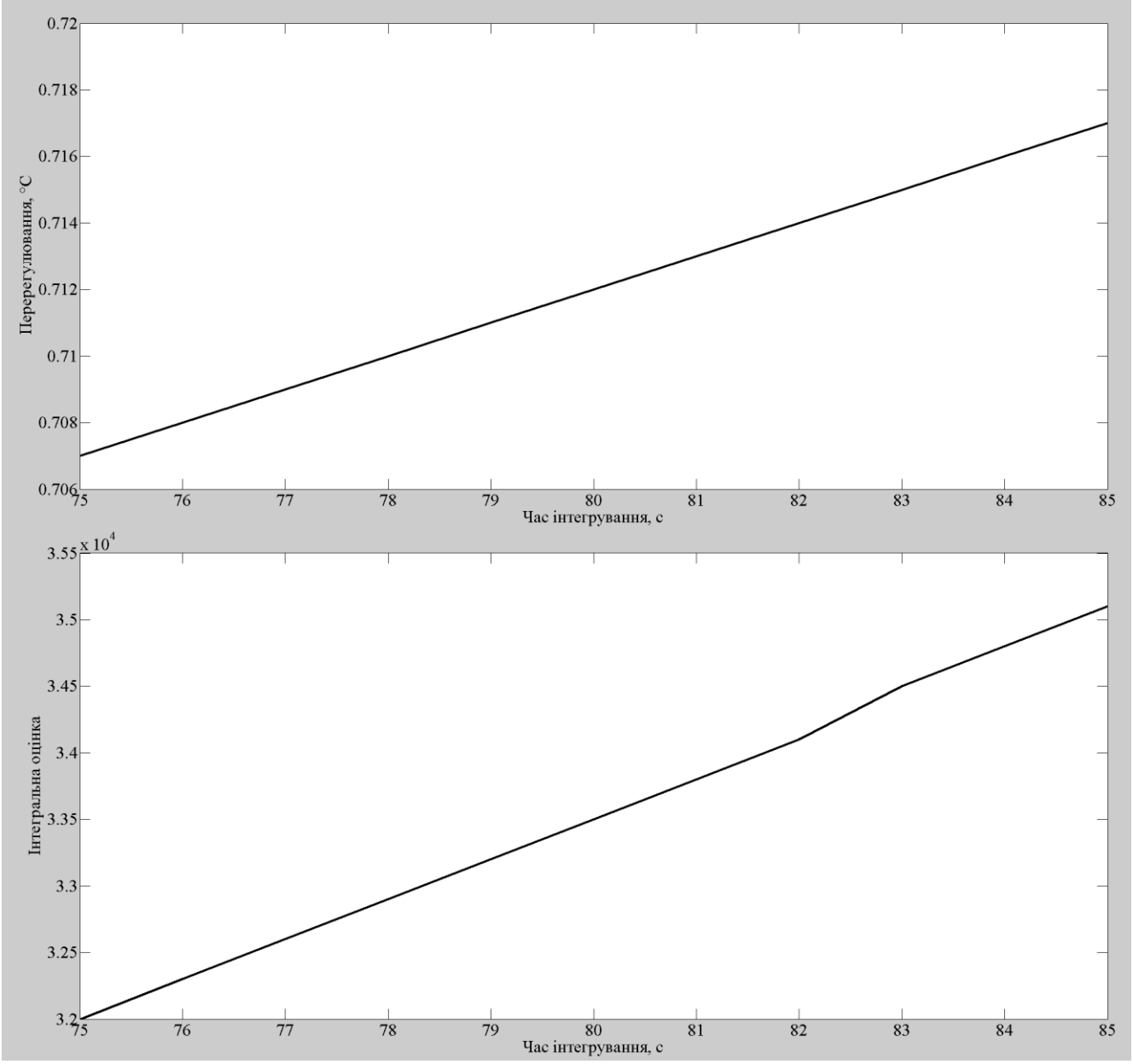


Рисунок 3.12 – Залежність показників якості від часу інтегрування

```
Залежності показників якості від часу інтегрування v1.0

Мінімальне перерегулювання
Ti: 75.000
Перерегулювання: 0.707
Інтегральна оцінка: 32000

Мінімальна інтегральна оцінка
```

```
Ti: 75.000
Перерегулювання: 0.707
Інтегральна оцінка: 32000
```

Враховуючи наявність від’ємної складової при значенні часу інтегрування меншому за 80 та наявність більшої складової інтегральної оцінки для більших значень часу інтегрування, в якості часу інтегрування пропорційно-інтегрального регулятора обрано 80 с. Результат налаштування системи керування наведено на рисунку 3.13.

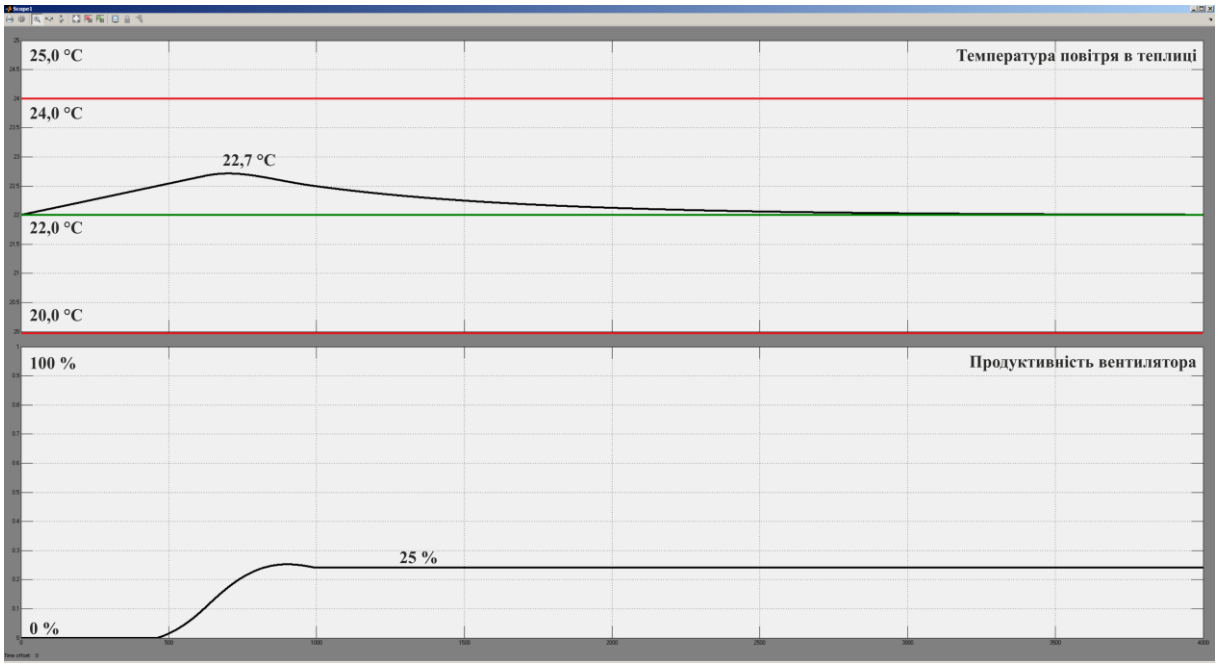


Рисунок 3.13 – Моделювання системи керування

Таким чином шляхом поступового наближення були встановленні діапазони зміни параметрів регуляторів в яких показники якості приймають оптимальні значення. Встановлено, що інтегральна оцінка має більший пріоритет за перерегулювання так як у всіх випадках абсолютне значення перерегулювання не виходить за межі обмеження ±2 °C. За результатами моделювання отримані залежності показників якості від коефіцієнта підсилення та часу інтегрування. На підставі аналізу залежностей та форм перехідних процесів виконано обґрунтування обрання параметрів пропорційно-інтегрального регулятора.

3.4 Модель імпульсного перетворювача

При розробці моделі системи керування та обґрунтуванні параметрів регулятора керування вентилятором виконувалося безперервно. Однак в системі, що

розробляється керування вентилятором виконується дискретно (ввімкнути/виключити). Рішенням цієї проблеми є використання імпульсного керуючого впливу, коли на підставі керуючого сигналу який формується регулятором імпульсний перетворювач формує імпульси довжина яких пропорційна до керуючого сигналу при незмінному періоді.

На підставі описаного алгоритму розроблено модель імпульсного перетворювача на мові програмування MATLAB:

```
function output = pulseGenerator(value, period)

% setPoint - Уставка
% period - Период, с
% output - Вихід

cycle = 0.01;

persistent sTime;                                                    % Час,
с
persistent sSwitchTime;                                              % Час
перемикання, с
persistent sOutput;                                                  % Вихі-
дне значення

if (isempty(sTime))
    sTime = 0.0;
    sSwitchTime = 0.0;
    sOutput = 0.0;
end

sTime = sTime - cycle;
if (sTime < 0.0)
    sTime = 0.0;
end

if (sTime == 0.0)
    sTime = period;
    sSwitchTime = period - period * value;
    sOutput = 1.0;
end

if (sSwitchTime >= sTime)
    sOutput = 0.0;
end

output = sOutput;
```

Використовуючи отриману модель імпульсного перетворювача розроблено модель системи керування мікрокліматом в кімнатній теплиці (рис. 3.14). У якості налаштування періоду сигналу обрано 10 секунд. Результати моделювання наведені на рисунку 3.15.

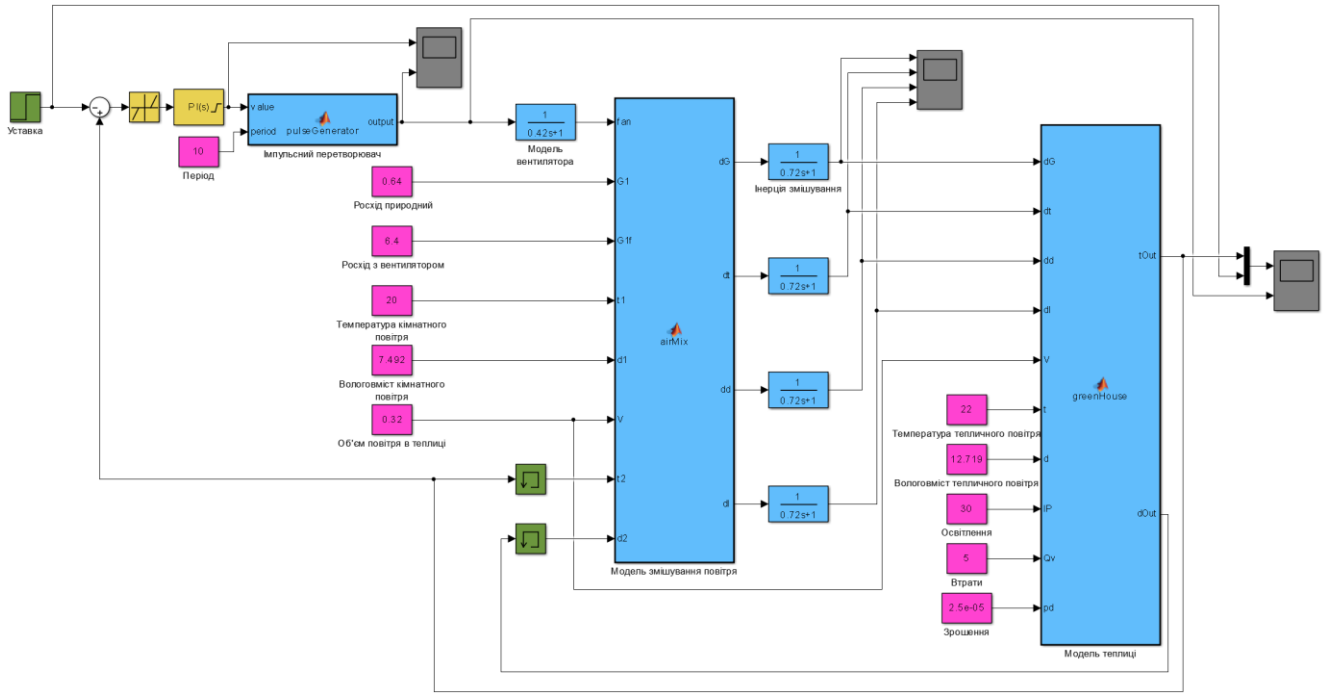


Рисунок 3.14 – Модель системи керування

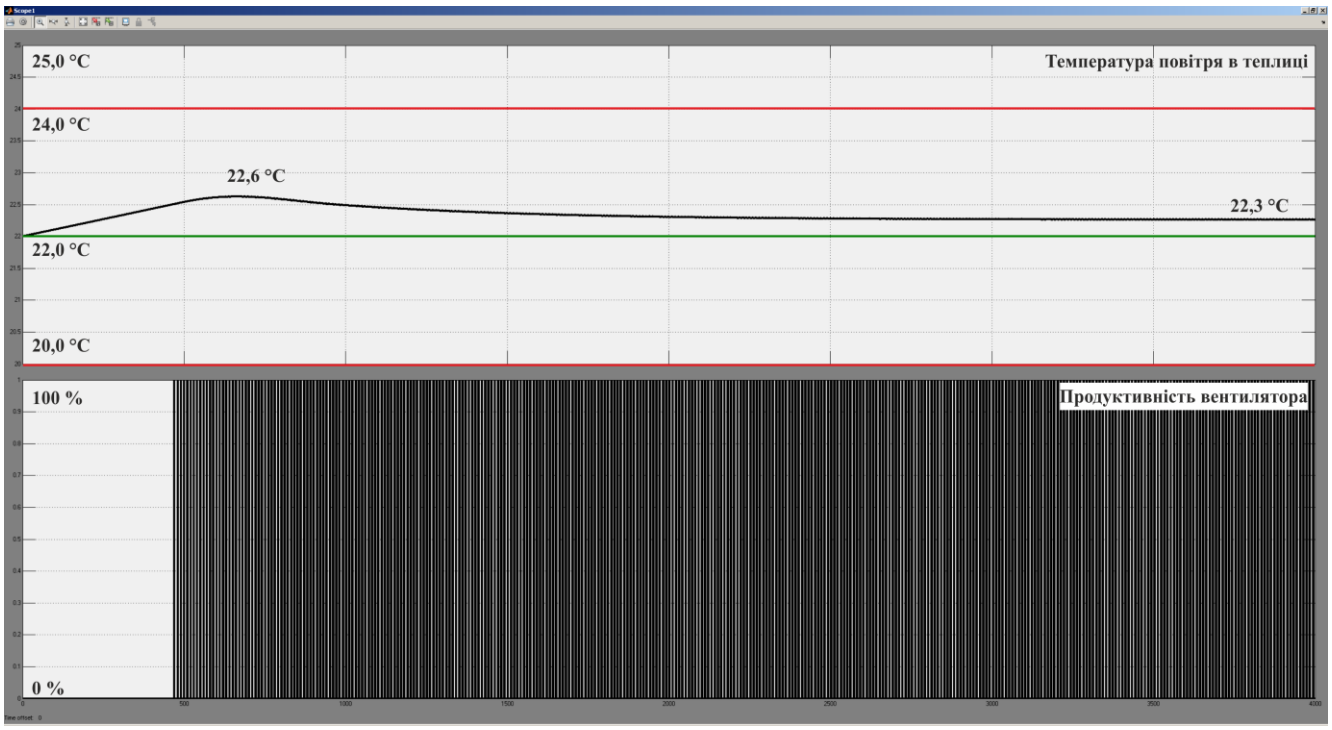


Рисунок 3.15 – Моделювання системи керування

Виходячи з отриманих даних присутня статична помилка 0,3 °C. Формування керуючого впливу виконується відповідно до алгоритму (рис. 3.16). Збільшення керуючого сигналу приводить к пропорційному збільшенню довжини імпульсу керуючого впливу.

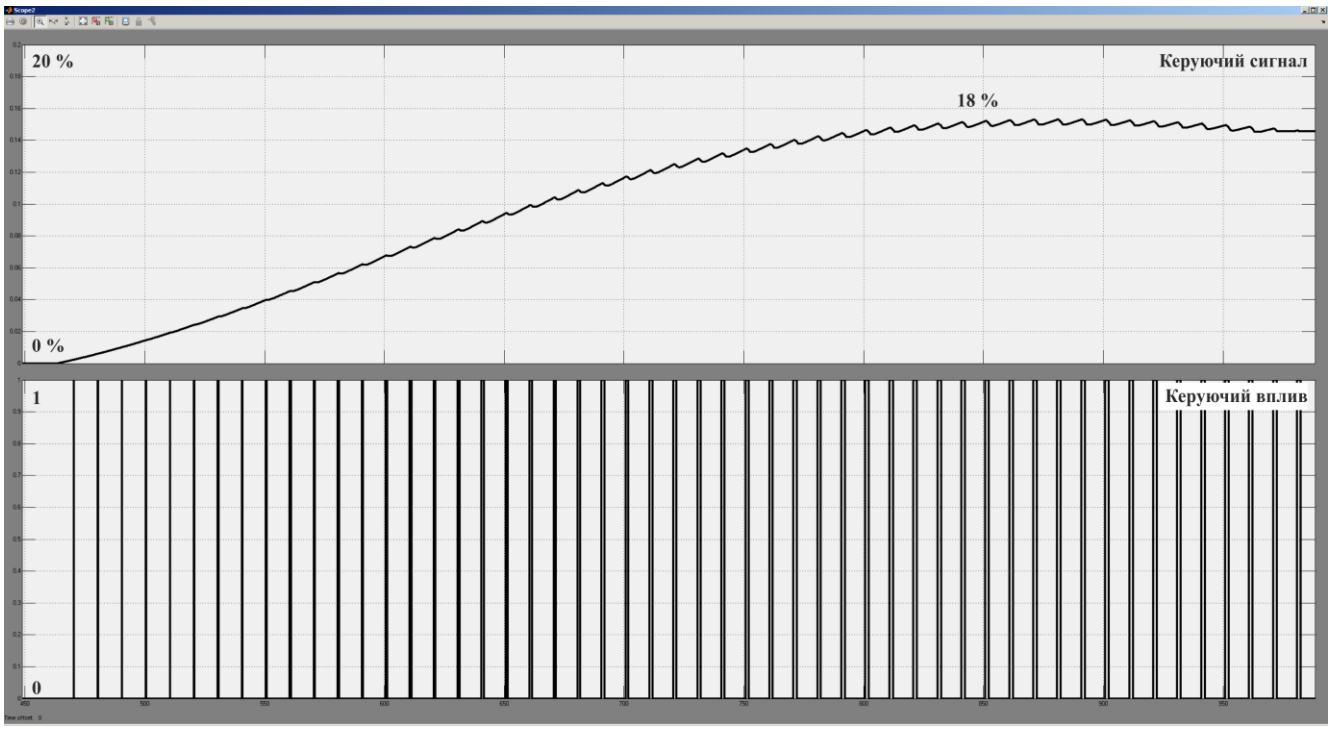


Рисунок 3.16 – Формування керуючого впливу

Враховуючи те що зменшення періоду формування керуючого впливу повинно призводити до покращення якості керування та виходячи з обмежень апаратного забезпечення отримані дані для періодів довжиною від 1 до 10 с (рис. 3.17).

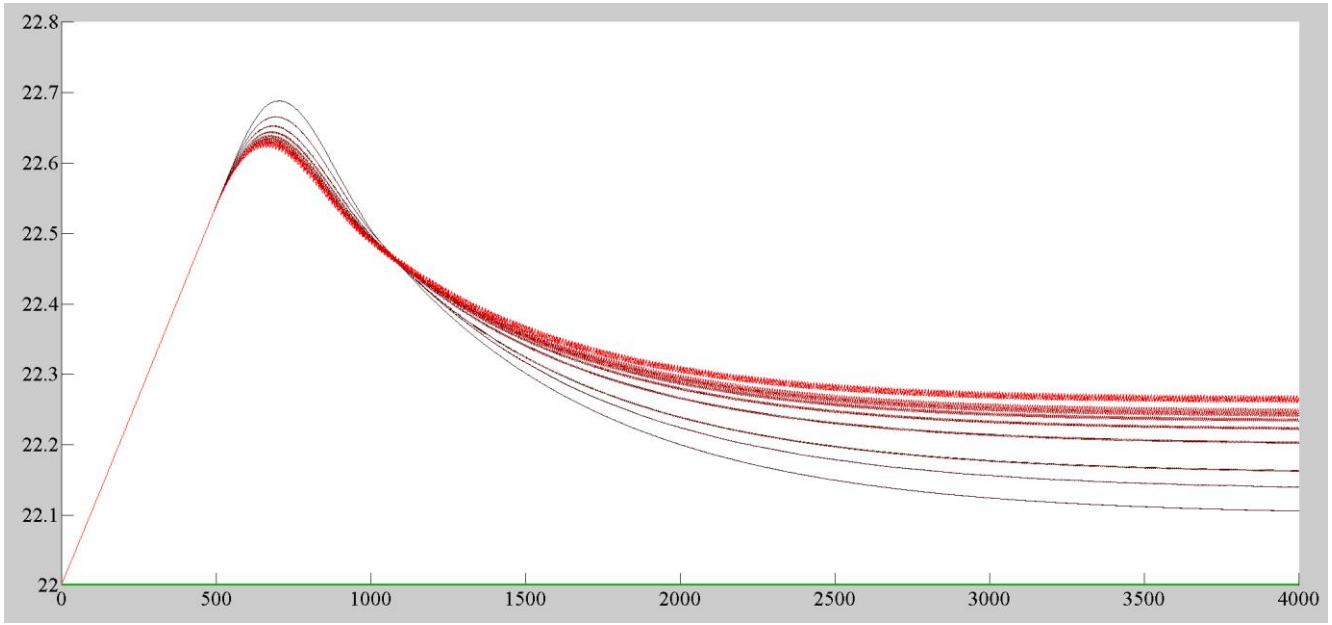


Рисунок 3.17 – Зміна періоду від 1 до 10 с

Як бачимо зменшення періоду призводить до зменшення статичної помилки. На підставі цього в якості значення періоду формування керуючого впливу

обрано час 1 с. Результат налаштування системи керування наведено на рисунку 3.18.

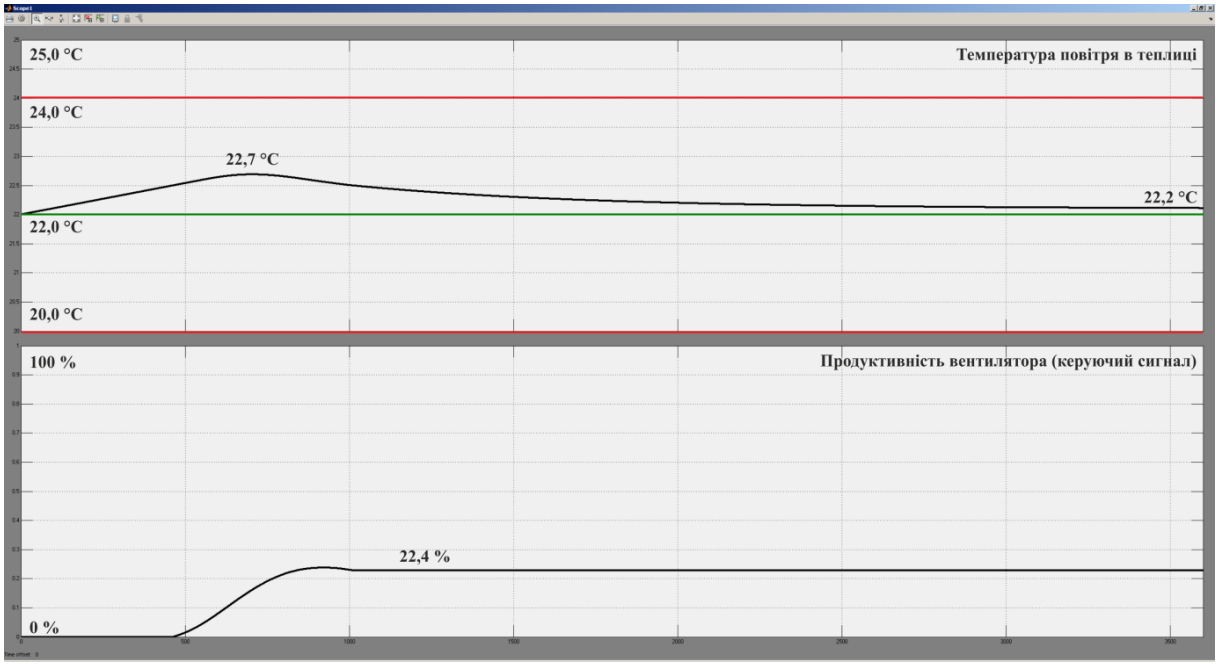


Рисунок 3.18 – Моделювання системи керування

Таким чином отримано модель імпульсного перетворювача який є частиною регулятора. Обґрунтовано обрання значення періоду формування керуючого впливу виходячи з обмежень апаратного забезпечення. Отримана система керування функціонує відповідно до вимог, модель регулятора може бути використана при розробці програмного забезпечення системи керування мікрокліматом в кімнатній теплиці.

3.5 Висновки по розділу

1. Запропоновані аналітичні засади на підставі котрих в графічному середовищі імітаційного моделювання Simulink математичного пакету MATLAB розроблено модель об’єкта керування. Отримана модель функціонує відповідно до фізичних закономірностей зміни температури в кімнатній теплиці.

2. На базі пропорційно-інтегрального регулятора з обмеженням керуючого впливу та запобіганням перенасичення інтегральної складової за методом “защіпки” розроблено систему керування. Виконано первинне ручне налаштування параметрів регулятора.

3. На підставі даних моделювання отриманих шляхом варіювання параметрів регулятора отримані залежності показників якості від параметрів. Враховуючи показники якості та графіки перехідних процесів обрані параметри пропорційно-інтегрального регулятора та виконано їх обґрунтування.

4. Враховано особливості обраного апаратного забезпечення за рахунок розробки імпульсного перетворювача. Досліджено вплив періоду формування керуючого впливу на якість функціонування системи керування, за результатами яких обрано його значення.

5. Отримано модель пропорційно-інтегрального регулятора з імпульсним перетворювачем яка може бути використана при розробці системи керування мікрокліматом в кімнатній теплиці.

4 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

4.1 Цифрова модель регулятора

Для отримання програмного забезпечення регулятора на підставі його безперервної моделі створено блок підсистеми. Пропорційно-інтегральний регулятор переведено до цифрової форми з часом виклику 0,01 с. Зона нечутливості є дискретним блоком, а імпульсний перетворювач написано на мові MATLAB. Таким чином отримано цифрову модель системи керування (рис. 4.1) з регулятором який має два входи та два виходи (рис. 4.2).

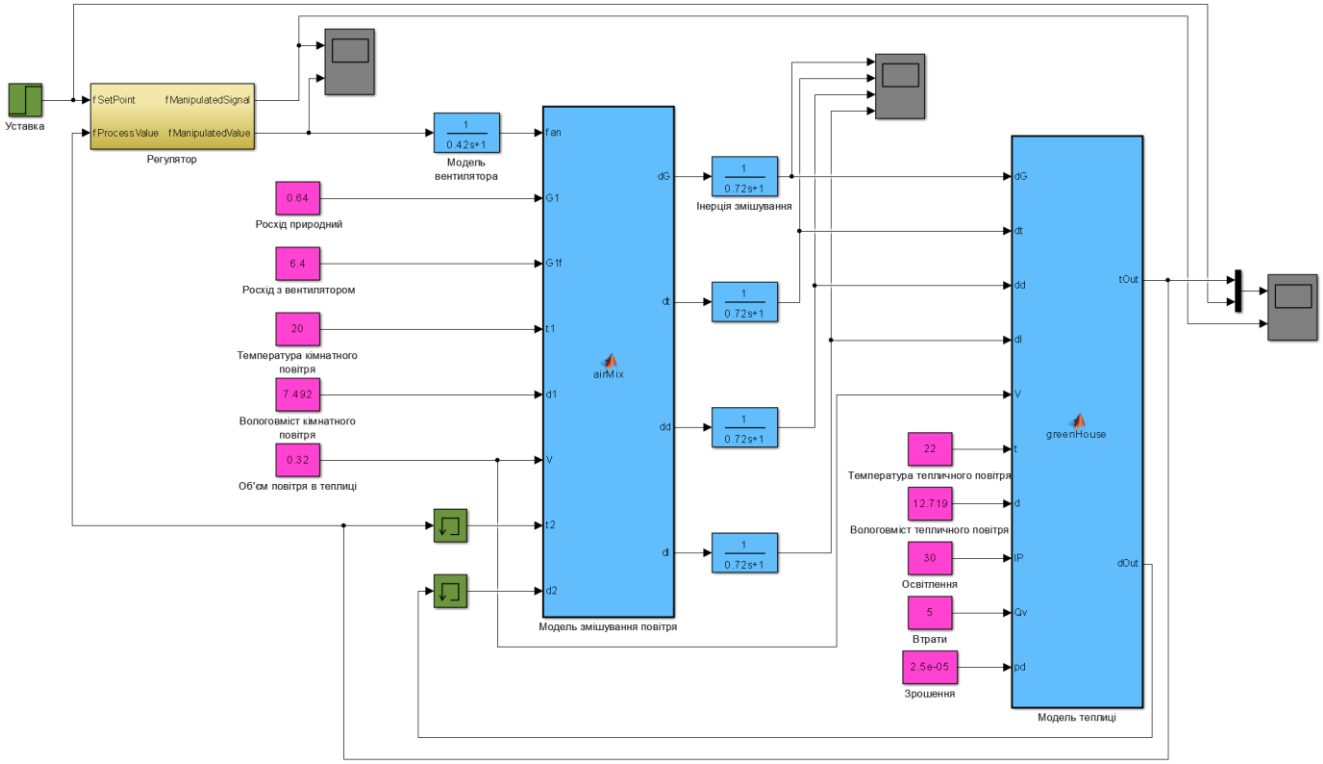


Рисунок 4.1 – Модель цифрової системи керування

На вхід регулятора задаються уставка температури в теплиці та дійсне температура в теплиці на підставі котрих обчислюється значення помилки керування. Зміна помилки керування в межах ±0,5 °С не приводить до зміни на вході регулятора. На підставі помилки керування регулятор розраховує значення керуючого сигналу. На підставі отриманого значення імпульсний перетворювач розраховує значення керуючого впливу на наступний період. Зміна значення керуючого впливу виконується кожну секунду. З блока отримуються значення керуючого сигналу та керуючого впливу.

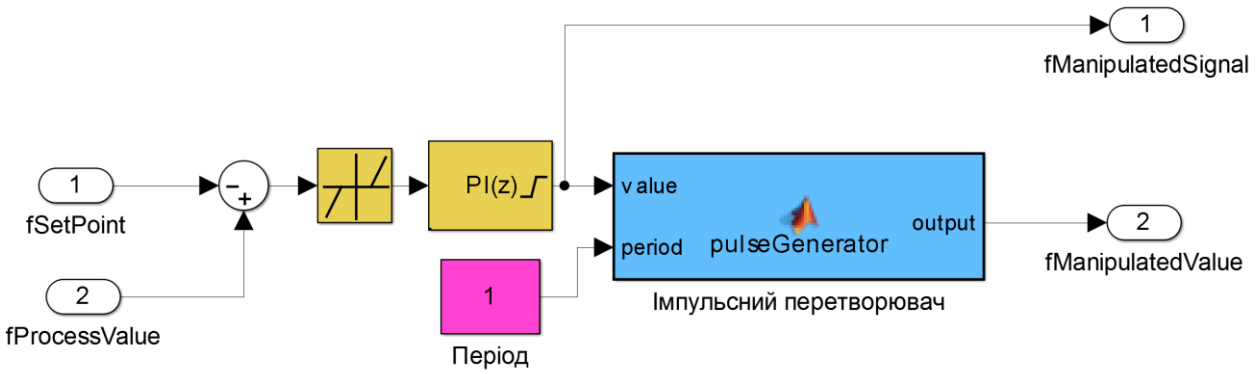


Рисунок 4.2 – Модель цифрового регулятора

Результати моделювання системи керування з цифровим регулятором наведені на рисунку 4.3.

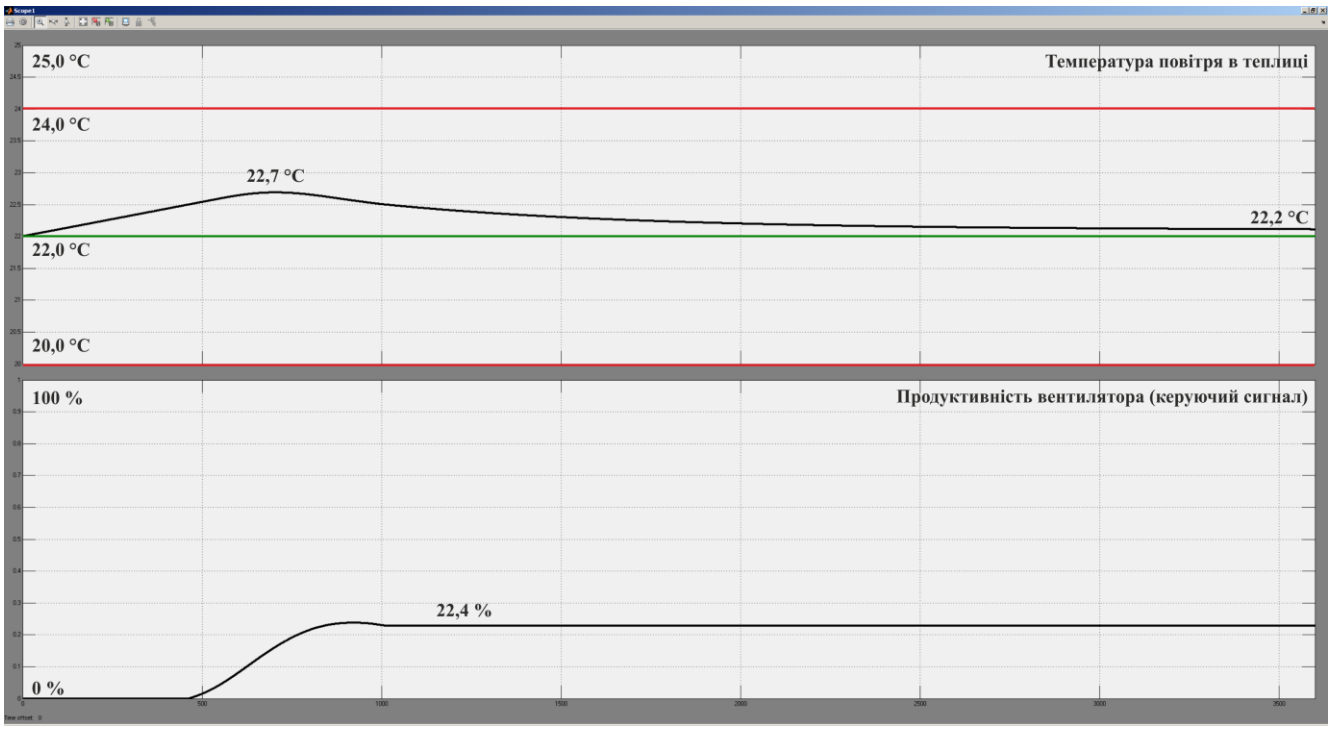


Рисунок 4.3 – Моделювання системи керування з цифровим регулятором

Для порівняння перехідного процесу системи керування з цифровим регулятором з системою керування з безперервним регулятором розроблено функцію на мові MATLAB:

```
function compareData()

    fprintf('\nПорівняння функціонування безперервного та цифрового регулятора
v1.0\n\n');

    load('data.mat');
    figure('Name', 'Порівняння функціонування безперервного та цифрового регулято-
ра');
    hold('on');
    plot(xref.Time, xref.Data, 'LineWidth', 3.0, 'Color', 'k');
    plot(x.Time, x.Data, 'LineWidth', 2.0, 'Color', 'r');
```

```
    hold('off');

    fprintf('Ступінь відповідності: %.2f\n', goodnessOfFit(xref.Data, x.Data,
'NRMSE') * 100);
end
```

Перевірка відповідності показала, що система керування з цифровим регулятором на 100 % відповідає безперервній (рис. 4.4), що пов’язано з однаковим шагом дискретизації 0,01 с.

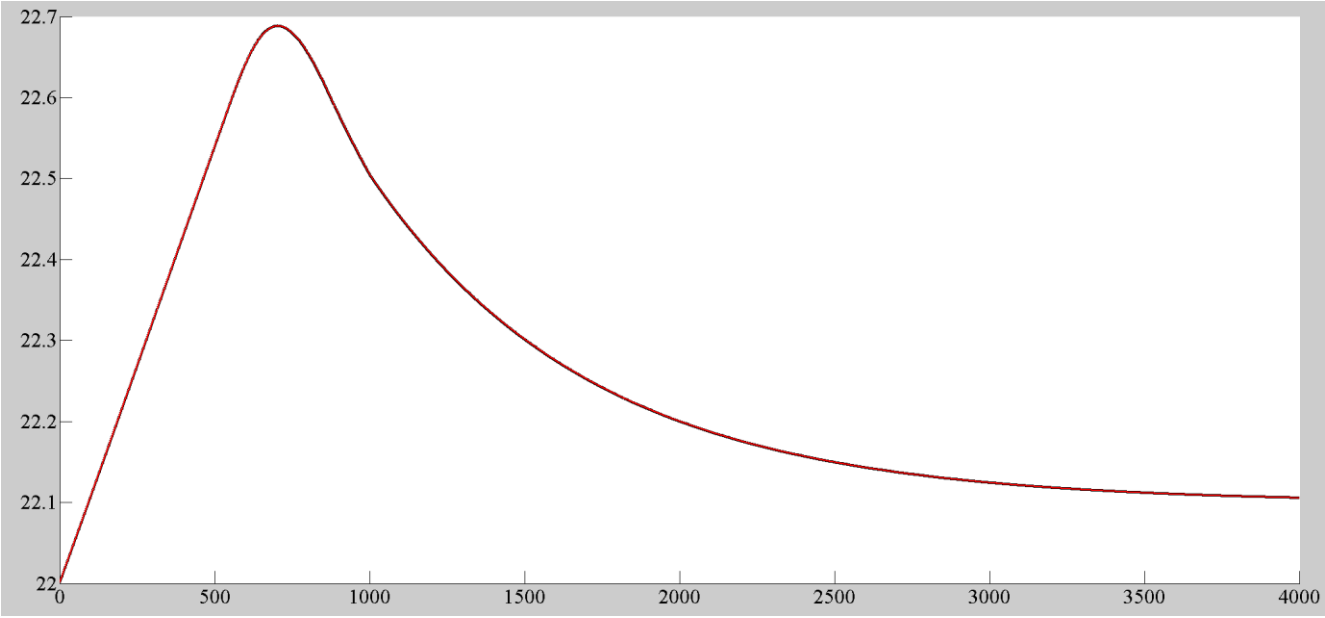


Рисунок 4.4 – Порівняння систем керування

На підставі отриманого цифрового регулятора розроблено функцію на мові C++ наведену в додатку А.3. Перед функцією об’являються структури відповідні виходу блоку та внутрішній пам’яті блока:

```
struct PIOutput // Результат розрахунку керуючого впливу
{
  float fManipulatedSignal; // Керуючий сигнал
  float fManipulatedValue; // Керуючий вплив
};

struct PIData // Дані регулятора
{
  float Integrator_DSTATE; // Стан інтегратора
  float sTime; // Поточний час
  float sSwitchTime; // Час перемикання
  float sOutput; // Стан виходу
} g_piData;
```

На початку функції об’являються тимчасові змінні:

```
  PIOutput result; // Результат розрахунку параметрів регулятора

  float rtb_Sum;
  float rtb_ProportionalGain;
  float rtb_IntegralGain;
  float rtb_Saturation;
  bool rtb_Equal1;
  bool rtb_Equal2;
```

```
float rtb_DeadZone;
float y;
float y_0;
```

Далі виконується розрахунок впливів регулятора:

```
rtb_IntegralGain = fProcessValue - fSetPoint; // Помилка керування

if (rtb_IntegralGain > 0.5) // Зона нечуствітельності
    rtb_IntegralGain = rtb_IntegralGain - 0.5f;
else if (rtb_IntegralGain >= -0.5)
    rtb_IntegralGain = 0.0;
else
    rtb_IntegralGain = rtb_IntegralGain - -0.5f;

rtb_Sum = rtb_IntegralGain + g_piData.Integrator_DSTATE; // Керуючий сигнал
rtb_ProportionalGain = 0.3f * rtb_Sum; // Пропорційна складова
rtb_IntegralGain = 0.0125f * rtb_IntegralGain; // Інтегральна складова
```

Після розрахунку обмежень керуючого сигналу та інтегральної складової виконується перетворення керуючого впливу до керуючого сигналу у імпульсній формі:

```
g_piData.sTime = g_piData.sTime - 0.01f; // Розрахунок часу
if (g_piData.sTime < 0.0)
    g_piData.sTime = 0.0;

if (g_piData.sTime = 0.0) // Час вийшов
{
    g_piData.sTime = 1.0;
    g_piData.sSwitchTime = 1.0f - rtb_Saturation;
    g_piData.sOutput = 1.0;
}

if (g_piData.sSwitchTime >= g_piData.sTime) // Імпульс закінчився
    g_piData.sOutput = 0.0;

result.fManipulatedSignal = rtb_Saturation; // Сигнал керування
result.fManipulatedValue = g_piData.sOutput; // Керуючий вплив
```

Перевірка отриманої функції було виконано в середовищі розробки програмного забезпечення Microsoft Visual Studio 2010 та Arduino 1.8.12 налаштованій для програмування контролера WEMOS D1. Перевірка показала, що програмне забезпечення регулятора функціонує відповідно до моделі регулятора.

4.2 Розробка програмного забезпечення системи керування

Програмне забезпечення системи керування розроблялося на базі програмного забезпечення системи керування отриманому при виконанні бакалаврської кваліфікаційної роботи з урахуванням наявності пропорційно-інтегрального регулятора температури повітря в теплиці (рис. 4.5).

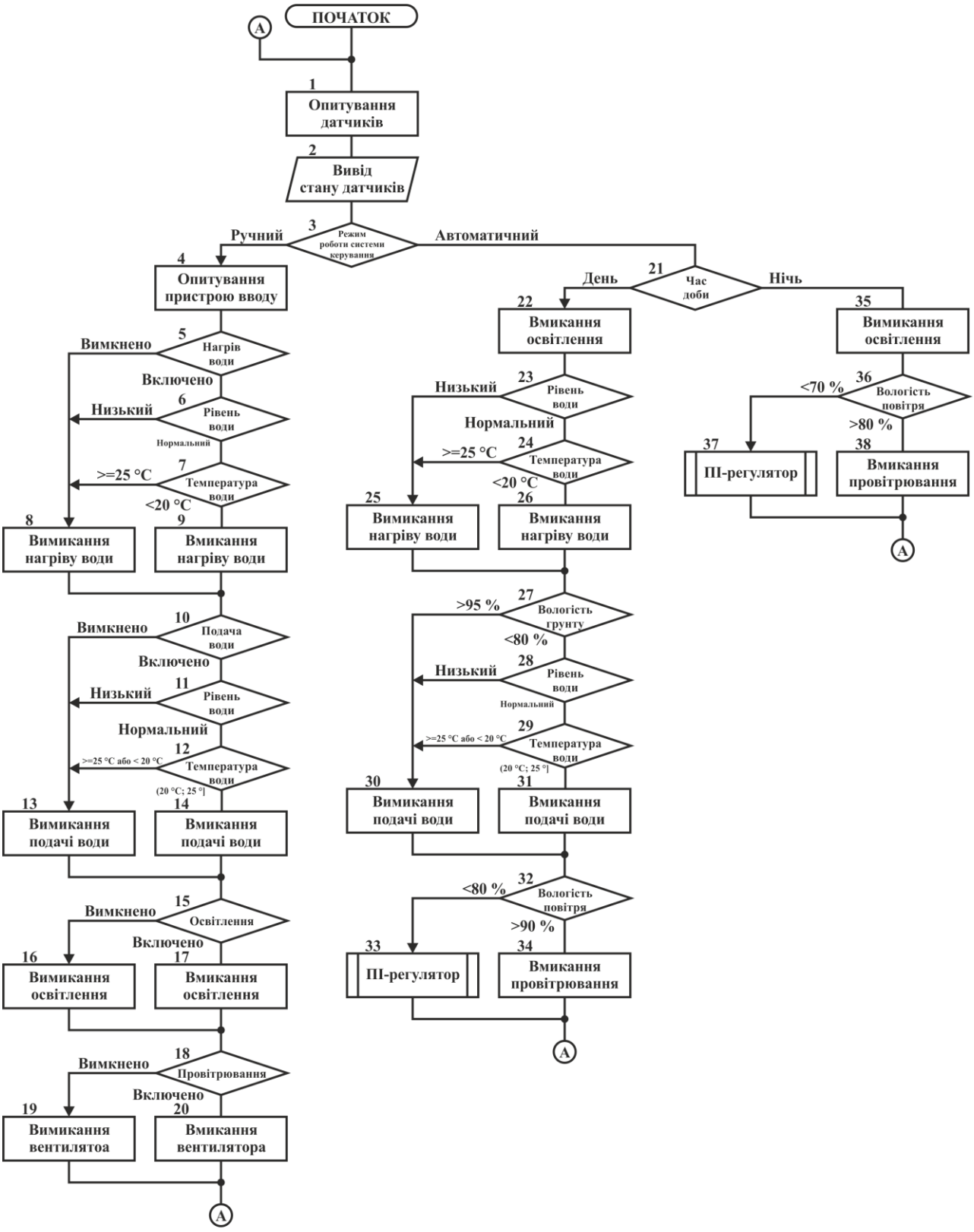


Рисунок 4.5 – Алгоритм керування

На початку кожного циклу виконується опитування усіх датчиків (1) та вивід інформації про їх стан (2) на рідкокристалічний індикатор та панель оператора

в якості котрої виступає персональний комп’ютер з SCADA системою zenon або пристрій з операційною системою Andorid чи iOS.

Відповідно до вимог система керування може функціонувати в двох режимах: ручному та автоматичному (3). В ручному режимі послідовно перевіряється виконувані дії: нагрів води (5), подача води (10), освітлення (15) та провітрювання (20).

Нагрів води (9) виконується якщо рівень води є нормальним (6), а її температура менша 20 °С (7). У випадку невиконання однієї з вимог нагрів води не відбувається (8).

Подача води (14) виконується якщо рівень води є нормальним (11), а її температура є більшою 20 °С та меншою 25 °С. У випадку невиконання однієї з вимог полив не відбувається (13).

Примусове вмикання і вимикання освітлення (16, 17) та провітрювання (19, 20) виконується без перевірки додаткових вимог.

При функціонуванні в автоматичному режимі система може знаходитися в двох станах: день та ніч. Стан день відповідає місцевому часу з 6 до 22 години та складає 16 годин. В даному стані вмикається освітлення (22) виконується підігрів води (26), полив (31) та провітрювання (33, 34).

Підігрів води виконується (26) коли рівень води є нормальним (23), а її температура є більшою 20 °С та меншою 25 °С. У свою чергу полив виконується (31) коли вологість ґрунту складає менше 80 % (27), її рівень є нормальним (28), а її температура є більшою 20 °С та меншою 25 °С. Полив вимикається (30) коли вологість ґрунту перевищує 95 % (27).

При вологості повітря більше 90 % (32) виконується провітрювання теплиці (34). Якщо вологість задовольняє вимогам вентилятором керує пропорційно-інтегральний регулятор (33) який стабілізує температуру повітря в теплиці.

Стан ніч відповідає місцевому часу з 22 до 6 годи та складає 8 годин. В даному стані освітлення вимикається (35). При вологості повітря більше 80 % (36) виконується провітрювання теплиці (38). Якщо вологість задовольняє вимогам

вентилятором керує пропорційно-інтегральний регулятор (37) який стабілізує температуру повітря в теплиці.

На підставі алгоритму керування розроблено відповідну функцію на мові програмування С++ наведену в додатку А.

Розроблена функція відповідно до алгоритму реалізує два режими роботи системи керування: ручний та автоматичний.

```
if (g_Hardware.getTypeWork() == 0)
{
    // Ручний режим
}
else
{
    // Автоматичний режим
}
```

Якщо в ручному режимі вмикається нагрів води перевіряється рівень води та її температура. У випадку коли рівень води нижчий за норму або температура води більша 25 °С нагрів води вимикається:

```
if (g_Hardware.getHeatWater() == 1) // Нагрів води
{
    if (g_Hardware.getWaterLevel() == 0) // Немає води
      g_Hardware.clearHeaterWater(); // Відключення нагріву

    if (g_Hardware.getWaterTemperature() >=
        g_Hardware.getSetPointDayWaterTemperatureMax()) // Вода нагріта
        g_Hardware.clearHeaterWater(); // Відключення нагріву
}
```

При включенні у ручному режимі поливу перевіряється рівень води та її температура. У випадку коли рівень води нижчий за норму або температура води менша 20 °С або більша 25 °С полив вимикається:

```
if (g_Hardware.getPump() == 1) // Працює насос
{
    if (g_Hardware.getWaterLevel() == 0) // Немає води
        g_Hardware.clearHeaterWater(); // Відключення нагріву

    if (g_Hardware.getWaterTemperature() <=
        g_Hardware.getSetPointDayWaterTemperatureMin()
        || g_Hardware.getWaterTemperature() >
        g_Hardware.getSetPointDayWaterTemperatureMax()) // Вода холодна чи гаряча
            g_Hardware.clearPump(); // Відключення насоса
    }
```

Згідно з алгоритмом програмне забезпечення в автоматичному режиму функціонує в двох станах: ніч та день. Розглянемо функціонування системи в режимі ніч.

Відповідно до вимог в режимі ніч вимикається освітлення. При вологість вище 80 % вмикається провітрювання, якщо вологість нижча вентилятором керує

пропорційно-інтегральний регулятор який підтримує температуру повітря в теплиці:

```
// Ніч
if ((g_Hardware.getHours() < g_Hardware.getSetPointDayHours()
     && g_Hardware.getMinutes() < g_Hardware.getSetPointDayMinutes())
    || (g_Hardware.getHours() >= g_Hardware.getSetPointNightHours()
     && g_Hardware.getMinutes() >= g_Hardware.getSetPointNightMinutes()))
{
    g_Hardware.clearLight(); // Вимкнення освітлення

    if (g_Hardware.getAirHumidity() >
        g_Hardware.getSetPointNightAirHumidityMax()) // Вологість повітря висока
        g_Hardware.setFan(); // Ввімкнення вентилятора
    else
    {
        PIOutput result = piController(
                         (float)g_Hardware.getSetPointDayAirTemperature(),
                         g_Hardware.getAirTemperatureFloat()); // ПІ-регулятор
        if (result.fManipulatedValue == 0) // Керування вентилятором
            g_Hardware.setFan();
        else
           g_Hardware.clearFan();
    }
}
```

В режимі день освітлення вмикається. Підігрів води виконується, якщо рівень води є нормальним, а її температура складає менше 20 °C. Полив виконується коли температура води складає від 20 до 25 °C, а рівень вологості ґрунту менше 80 %:

```
    g_Hardware.setLight(); // Ввімкнення освітлення

    if (g_Hardware.getWaterLevel() == 0) // Немає води
        g_Hardware.clearHeaterWater(); // Вимкнення нагріву
    else if (g_Hardware.getWaterTemperature() <
             g_Hardware.getSetPointDayWaterTemperatureMin()) // Вода холодна
      g_Hardware.setHeaterWater(); // Ввімкнення нагріву

    if (g_Hardware.getWaterTemperature() >
        g_Hardware.getSetPointDayWaterTemperatureMax()) // Вода гаряча
        g_Hardware.clearHeaterWater(); // Вимкнення нагріву

    // Немає води або вода холодно або гаряча
    if (g_Hardware.getWaterLevel() == 0
        || g_Hardware.getWaterTemperature() <
           g_Hardware.getSetPointDayWaterTemperatureMin()
        || g_Hardware.getWaterTemperature() >=
           g_Hardware.getSetPointDayWaterTemperatureMax())
      g_Hardware.clearPump(); // Вимкнення насоса
    // Вологість ґрунту низька
    else if (g_Hardware.getGroundHumidity() <
             g_Hardware.getSetPointDayGroundHumidityMin())
         g_Hardware.setPump(); // Ввімкнення насоса

    // Вологість ґрунту велика
    if (g_Hardware.getGroundHumidity() >
        g_Hardware.getSetPointDayGroundHumidityMax())
        g_Hardware.clearPump(); // Вимкнення насоса
```

При вологість вище 90 % вмикається провітрювання, якщо вологість нижча вентилятором керує пропорційно-інтегральний регулятор який підтримує температуру повітря в теплиці:

```
if (g_Hardware.getAirHumidity() >
    g_Hardware.getSetPointDayAirHumidityMax()) // Вологість повітря висока
    g_Hardware.setFan(); // Включення вентилятора
else // Вологість повітря нормальна
{
    PIOutput result = piController(
                    (float)g_Hardware.getSetPointDayAirTemperature(),
                     g_Hardware.getAirTemperatureFloat()); // ПІ-регулятор
    if (result.fManipulatedValue == 0) // Керування вентилятором
        g_Hardware.setFan();
    else
      g_Hardware.clearFan();
}
```

В якості протоколу зв’язку між контролером та зовнішнім програмним забезпеченням обрано Modbus TCP, для чого використано бібліотеку Modbu-?TCPSlave яка розповсюджується за ліцензією GNU. Аналіз тексту бібліотеки показав, що вона не повністю відповідає вимогам. Таким чином до неї було внесено ряд змін: видалена підтримка усіх контролерів крім основаних на контролері ESP8266, замінено об’явлення на перечислення, перероблено функцію запуску для роботи в режимі точки доступу:

```
void ModbusTCPSlave::begin(const char *csSSID, const char *csPassword, const
uint8_t nIP[4], const uint8_t nGateway[4], const uint8_t nSubnet[4])
{
    WiFi.mode(WIFI_AP_STA);
    WiFi.softAPConfig(IPAddress(nIP), IPAddress(nGateway), IPAddress(nSubnet));
    WiFi.softAP(csSSID, csPassword);

    m_Server.begin(); // Запуск сервера
}
```

Відповідно до цього були внесені зміни до функції зв’язку з Master пристроєм:

```
byte byteFN = MB_FC_NONE;
int Start;
int WordDataLength;
int ByteDataLength;
int MessageLength;

if (m_Server.hasClient()) // Перевірка наявності клієнта
{
    if (!m_Client || !m_Client.connected()) // Якщо клієнт вільний
    {
        if (m_Client)
            m_Client.stop();
        m_Client = m_Server.available();
    }
    else // Клієнт зайнятий, відмова в підключенні
    {
```

```
        WiFiClient serverClient = m_Server.available();
        serverClient.stop();
    }
}

// Читання даних з буфера
if (m_Client && m_Client.connected() && m_Client.available())
{
    unsigned int nIndex = 0;
    while(m_Client.available() && nIndex < sizeof(m_Buffer))
    {
        m_Buffer[nIndex] = m_Client.read();
        ++nIndex;
    }
}
else
    return;
```

Таким чином відповідно до вимог було розроблено програмне забезпечення системи керування мікрокліматом в кімнатній теплиці яке реалізує ручний та автоматичний режими роботи. Система забезпечує керування мікроклімату в день та вночі. В ручному режимі реалізуються блокування виконавчих пристроїв у випадку невідповідності команд оператора до вимог технологічного процесу. В автоматичному режимі система керує нагрівачем води, насосом, освітленням та вентилятором при цьому контролюються температура і вологість повітря, температура і вологість ґрунту, температура води і її наявність. Підчас перевірки функціонування програмного забезпечення на контролері було встановлено, що усі датчики та виконавчі пристрої функціонують, а на рідкокристалічному індикаторі відображається їх стан (рис. 4.6).

Рисунок 4.6 – Функціонування контролера

На індикаторі мнемоніки AH – відповідає вологості повітря, AT – відповідає температурі повітря, WT – температурі води, WL – наявності води, GH – вологості ґрунту, GT - температурі ґрунту, HLFP – стану нагрівача, освітлення, вентилятора та насоса. Зміна параметрів на індикаторі виконується кожні 5 секунд.

4.3 Розробка програмного забезпечення людино-машинного інтерфейсу

Відповідно до завдання програмне забезпечення системи керування повинно відображати процес керування мікрокліматом в кімнатній теплиці та функціонувати в двох режимах: ручному та автоматичному. Згідно з цим в SCADA систему додано драйвер інтерфейсу Modbus RTU “MODRTU32” та змінні датчиків і виконавчих пристроїв (рис. 4.7).

State	Name	Measur...	Driver
	Filter text	Filter...	Filter text
	AirTemperature	°C	MODRTU32 - Controller
	AirWet	%	MODRTU32 - Controller
	DayTime		MODRTU32 - Controller
	Fan		MODRTU32 - Controller
	GruntTemperature	°C	MODRTU32 - Controller
	GruntWet	%	MODRTU32 - Controller
	Illumination		MODRTU32 - Controller
	Pump		MODRTU32 - Controller
	TypeWork		MODRTU32 - Controller
	WaterHeater		MODRTU32 - Controller
	WaterLevel		MODRTU32 - Controller
	WaterTemperature	°C	MODRTU32 - Controller

Рисунок 4.7 – Змінні датчиків та виконавчих пристроїв

Згідно з вимогами розмір зображення людино-машинного інтерфейсу має бути 1920 на 1080 пікселей, для перемикання між зображеннями необхідно використовувати кнопки. На підставі цього розроблено шаблон наведений на рисунку 4.8.

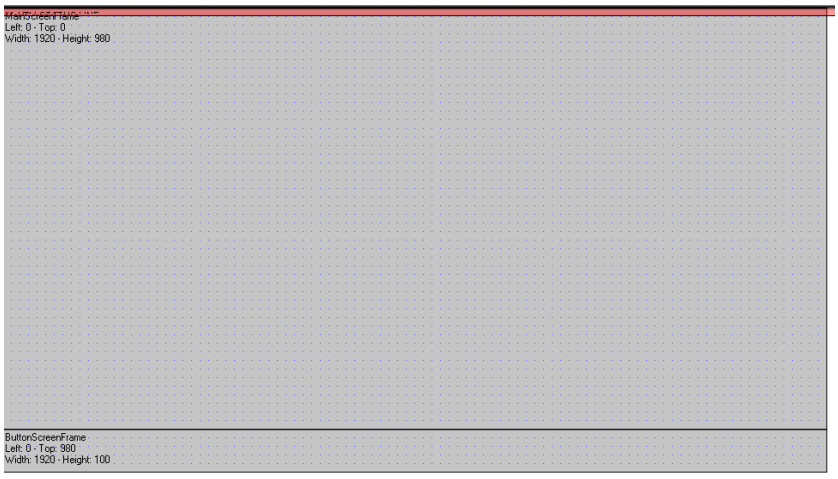


Рисунок 4.8 – Шаблон зображення

Згідно з шаблоном розроблено зображення кнопок (рис. 4.9) на якому розміщено кнопки перемикання між титульним аркушем, технологічним процесом та ручним режимом роботи. Крім того на зображенні розміщено часи, кнопку оновлення зображення та закриття людино-машинного інтерфейсу.

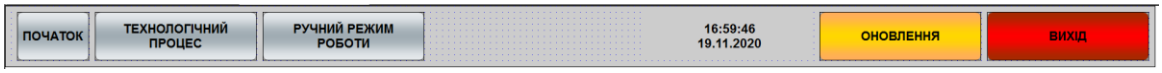


Рисунок 4.9 – Зображення кнопок

На початковому зображенні відображається інформація про кваліфікаційну роботу (рис. 4.10).

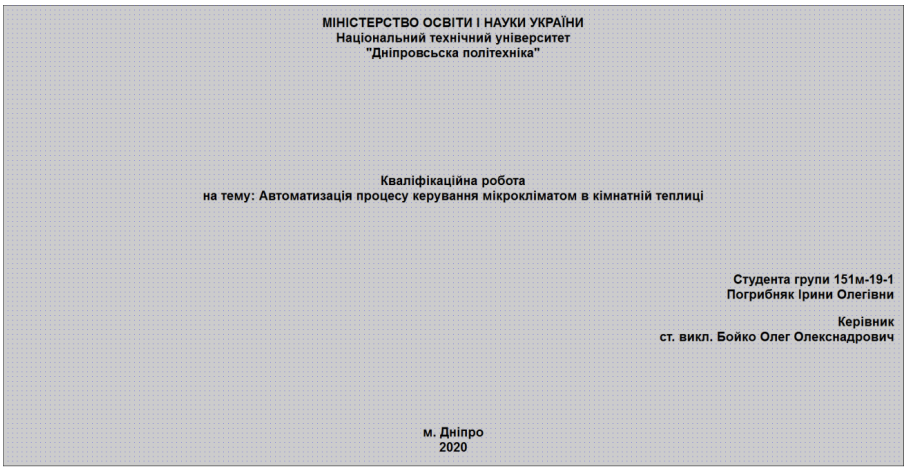


Рисунок 4.10 – Початкове зображення

Зображення технологічного процесу відображає об’єкт керування стан датчиків та виконавчих пристроїв до яких відносяться рівень води, температура води, температура та вологість повітря, освітлення, температура та вологість ґрунту, насос, нагрівач води та вентилятор (рис. 4.11).

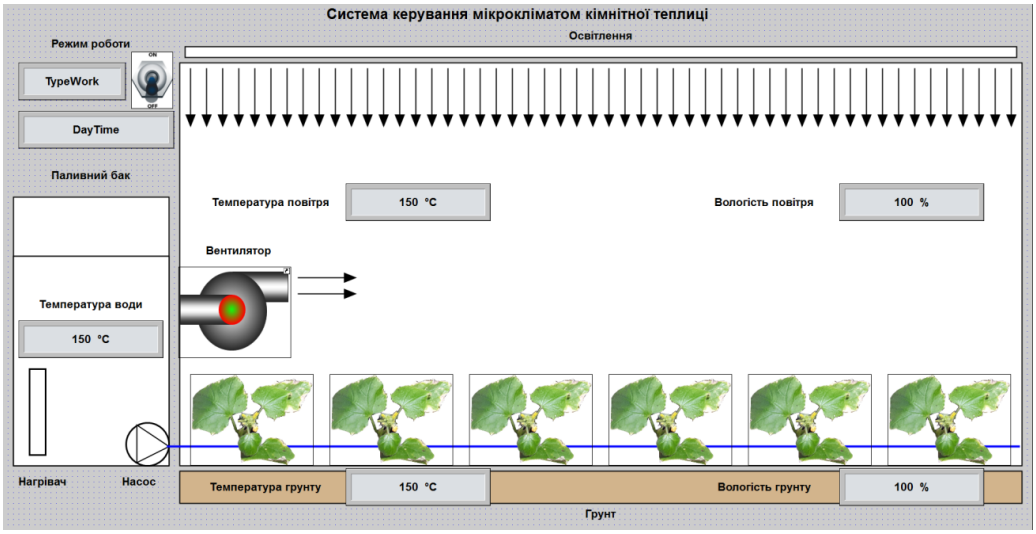


Рисунок 4.11 – Зображення технологічного процесу

Зображення ручного режиму дозволяє вмикати та вимикати пристрої керування до яких відносяться нагрівач, насос, вентилятор та освітлення (рис. 4.12).

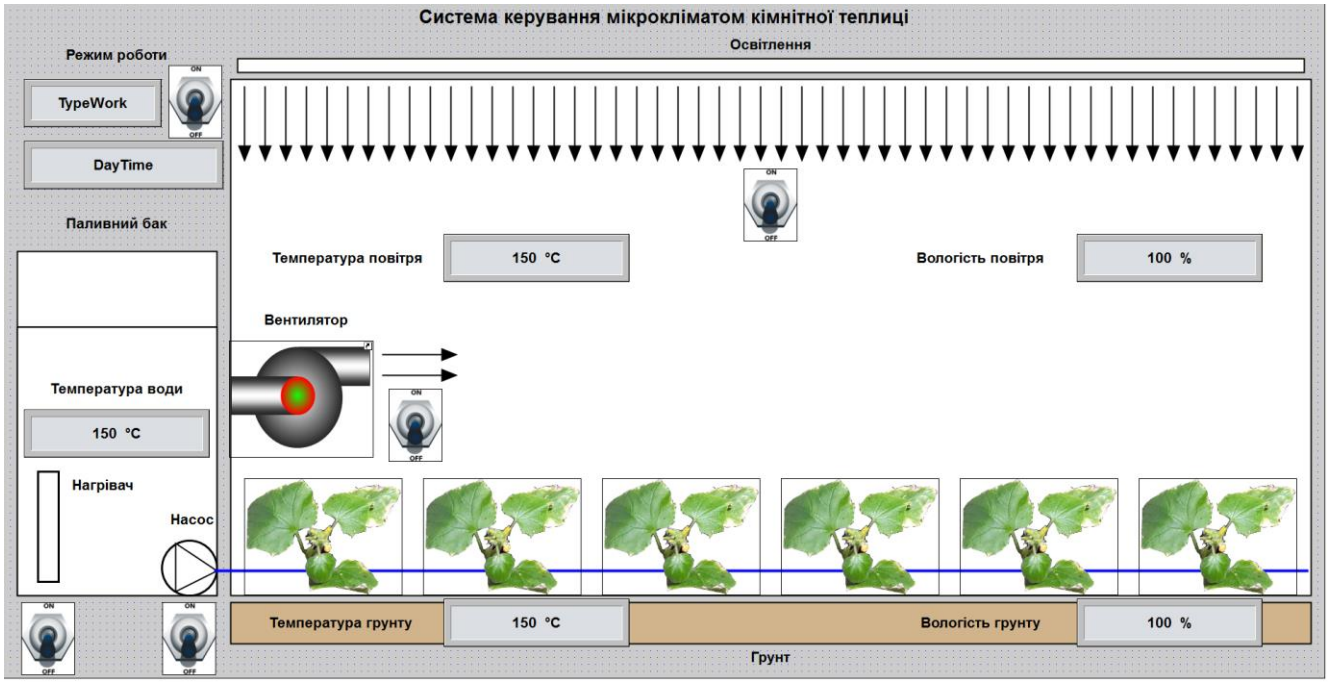


Рисунок 4.12 – Зображення ручного режиму роботи

Перевірка функціонування людино-машинного інтерфейсу показало, що воно функціонує відповідно до завдання та відповідає усім вимогам. Дозволяє контролювати стани датчиків та керувати виконавчими пристроями. Крім того воно має можливість відображатися на пристроях з операційними системами Android та iOS за допомогою програмного забезпечення ZenMobile for zenon.

4.4 Розробка бібліотеки зв’язку з системою керування

Виходячи з того, що за вимогами з системою керування повинні мати можливість зв’язуватися пристрої керування з операційними системами Android та iOS було перевірено можливість використання бібліотек EasymodbusTCP та JLibModbus для обміну даними з розробленим контролером та SCADA системою zenon. Встановлено, що вони не сумісні з драйвером MODRTU32 SCADA системи zenon, що пов’язано з невірним формуванням ідентифікатора транзакцій. Таким чином для збереження сумісності роботи програмного забезпечення системи керування з SCADA системою zenon на мові програмування Java розроблено бібліотеку зв’язку з системою керування (додаток В).

Бібліотека складається з двох класів ModbusTCPFactory та Modbu-?TCPPacket. Перший використовується для формування пакету Modbus TCP відповідно до його функції, а другий відповідає за розмішення даних в пакеті Modbus TCP.

Розгляним класс ModbusTCPPacket. Клас має два конструктори перший використовується для створення пакета за значеннями полів:

```
ModbusTCPPacket(short nTID, byte nUID, byte nFunction, byte[] data)
{
    // Конструктор пакета на підставі значень полів

    m_nTID = nTID;
    m_nUID = nUID;
    m_nLength = (short)(1 + 1 + data.length);
    m_nFunction = nFunction;
    m_Data = data;
}
```

Другий конструктор використовується для створення пакету на підставі отриманих даних:

```
ModbusTCPPacket(char[] dataBuffer)
{
    // Конструктор пакета на підставі отриманих даних

    if (dataBuffer.length < m_nLengthMin)
        return;

    m_nTID = (short)((((int)dataBuffer[m_nTIDAddress]) << 8)
                     | (dataBuffer[m_nTIDAddress + 1]));
    m_nUID = (byte)dataBuffer[m_nUIDAddress];
    m_nLength = (short)((((int)dataBuffer[m_nLengthAddress]) << 8)
                    | (dataBuffer[m_nLengthAddress + 1]));
    m_nFunction = (byte)dataBuffer[m_nFunctionAddress];

    m_Data = new byte[dataBuffer.length - m_nDataAddress];
    for (int nIndex = m_nDataAddress; nIndex < dataBuffer.length; ++nIndex)
        m_Data[nIndex - m_nDataAddress] = (byte)dataBuffer[nIndex];
}
```

Інформація в пакеті зберігаються у вигляд структури відповідної до протоколу Modbus TCP. Клас дозволяє перетворити її до даних які можуть бути відправлені відповідно до протоколу TCP:

```
byte[] getPacket()
{
    // Отримання готового пакета

    byte[] result = new byte[m_nMBAP + 1 + m_Data.length];
    result[m_nTIDAddress]       = (byte)(m_nTID >>> 8);
    result[m_nTIDAddress + 1]   = (byte)(m_nTID);
    result[m_nPIDAddress]       = 0;
    result[m_nPIDAddress + 1]   = 0;
    result[m_nLengthAddress]    = (byte)(m_nLength >> 8);
    result[m_nLengthAddress + 1]= (byte)(m_nLength);
    result[m_nUIDAddress]       = m_nUID;
```

```
    result[m_nFunctionAddress]  = m_nFunction;
    System.arraycopy(m_Data, 0, result, m_nDataAddress, m_Data.length);

    return result;
}
```

Як було зазначено вище клас ModbusTCPFactory реалізує формування пакетів відповідно до функцій протоколу Modbus TCP. Для отримання даних з системи керування використовується функції читання регістрів:

```
static ModbusTCPPacket ReadRegisters(int nTID, int nUID, int nAddress, int
nQuantity)
{
    byte[] data = new byte[]{(byte)(nAddress >>> 8), (byte)(nAddress),
                             (byte)(nQuantity >>> 8), (byte)(nQuantity)};
    return new ModbusTCPPacket((short)nTID, (byte)nUID,
                               m_nFunctionReadRegisters, data);
}
```

Для запису передачі даних до системи керування використовується функції запису регістрів:

```
static ModbusTCPPacket WriteMultipleRegisters(int nTID, int nUID,
                                              int nAddress, int[] values)
{
    byte[] data = new byte[2 + 2 + 1 + values.length * 2];
    data[0] = (byte)(nAddress >>> 8);
    data[1] = (byte)(nAddress);
    data[2] = (byte)(values.length >>> 8);
    data[3] = (byte)(values.length);
    data[4] = (byte)(values.length * 2);
    return new ModbusTCPPacket((short)nTID, (byte)nUID,
                               m_nFunctionWriteMultipleRegisters, data);
}
```

Перевірка функціонування розробленої бібліотеки показало, що зв’язок з системою керування є стабільним, обмін даних відбувається відповідно до протоколу Modbus TCP (рис. 4.13).

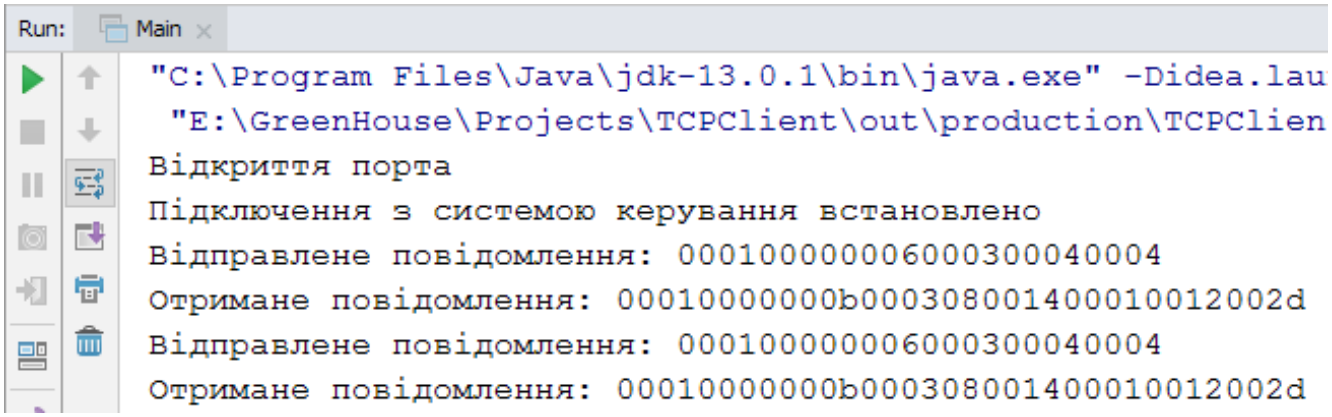


Рисунок 4.13 – Перевірка функціонування бібліотеки зв’язку з системою керування мікрокліматом в кімнатній теплиці

Таким чином запропонована бібліотека зв’язку за протоколом Modbus TCP може бути використана при розробці програмного забезпечення людино-машинного інтерфейсу для пристроїв з операційними системами Android та iOS.

4.5 Висновки по розділу

1. На підставі безперервної моделі системи керування отримана цифрова модель пропорційно-інтегрального регулятора з імпульсним перетворювачем. Порівняння результатів моделювання безперервної та цифрової системи показало відповідність 100 %. На підставі отриманої цифрової моделі регулятора розроблена функція на мові програмування С++.

2. На підставі модифікованого алгоритму керування та функції яка реалізує пропорційно-інтегральний регулятор розроблено програмне забезпечення системи керування, яке забезпечує керування мікроклімату в день та вночі, в ручному та автоматичному режимі. В ручному режимі реалізуються блокування виконавчих пристроїв у випадку невідповідності команд оператора до вимог технологічного процесу. В автоматичному режимі система керує нагрівачем води, насосом, освітленням та вентилятором при цьому контролюються температура і вологість повітря, температура і вологість ґрунту, температура води і її наявність.

3. Перевірка функціонування програмного забезпечення на контролері показала, що усі датчики та виконавчі пристрої функціонують, а на рідкокристалічному індикаторі відображається їх стан.

4. Для SCADA системи zenon розроблено людино-машинний інтерфейс який відображає режим роботи системи керування, час доби, стан датчиків та виконавчих пристроїв. Крім того він дозволяє у ручному режимі змінювати стан виконавчих пристроїв. Отриманий людино машинний інтерфейс за допомогою ZenMobile for zenon відображається на пристроях з операційними системи Android та iOS.

5. Перевірка бібліотек EasymodbusTCP, JLibModbus які реалізують протокол Modbus TCP на працю з SCADA системою zenon показала, що вони не сумісні з її драйвером MODRTU32, що пов’язано з невірним формуванням ідентифікатора

транзакцій. На мові програмування Java розроблено бібліотеку для зв’язку пристроїв з операційними системами Android та iOS з системою керування. Бібліотека може бути використана при розробці програмного забезпечення людино-машинного інтерфейсу системи керування для таких пристроїв, що може стати подальшим розвитком кваліфікаційної роботи.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Техніко-економічне обґрунтування впровадження автоматизації процесу підтримання мікроклімату кімнатної теплиці

У кваліфікаційній роботі розглядається економічна доцільність розробки та впровадження системи керування процесом підтримки мікроклімату кімнатної теплиці.

Розвиток технічних, апаратних і програмних засобів, дає можливість підприємствам зменшувати витрати, підвищувати якість продукції, збільшувати швидкість виробництва, автоматизувати процеси і безліч інших рішень.

У цьому розділі дипломного проекту приведено економічне обґрунтування доцільності використання автоматизованої системи керування процесу підтримання мікроклімату кімнатної теплиці

Система забезпечує керування мікроклімату в день та вночі. В ручному режимі реалізуються блокування виконавчих пристроїв у випадку невідповідності команд оператора до вимог технологічного процесу. В автоматичному режимі система керує нагрівачем води, насосом, освітленням та вентилятором при цьому контролюються температура і вологість повітря, температура і вологість ґрунту, температура води і її наявність.

Для того, щоб визначити економічну ефективність впровадження даної системи замість застарілої базової, проведемо розрахунки річних витрат та прибутків.

5.2 Розрахунок капітальних витрат пов'язаних з впровадженням системи керування

Розрахуємо капітальні витрати, що пов'язані з виготовленням та впровадженням комп’ютеризованої системи керування підтримання температури в головній колоні при видобутку бензолу з вугілля. Визначення проектних капітальних витрат проводиться за такою формулою:

$$К_{пр} = С_{об} + Д_{тр} + М_{мн} \quad (5.1)$$

де $С_{об}$ – витрати на комплектуючі вироби;

$Д_{тр}$ – витрати на транспортно-заготівельні витрати;

$М_{мн}$ – витрати на монтаж і налагодження системи;

Вартість комплектуючих деталей наведена в таблиці 5.1.

Таблиця 5.1 – Вартість комплектуючих системи

№	Найменування апаратного забезпечення	Одиниці виміру	Кількість	Оптова ціна за од., грн.	Сума, грн.
1	Arduino Uno R3	од.	1	125	125
2	Датчик контролю повітря	од.	1	29	29
3	4-х канальний релейний модуль	од.	1	83	83
4	Wi-Fi модуль ESP8266	од.	1	72	72
5	Рідкокристалічний екран LCD1602	од.	1	91	91
6	Датчик рівня води	од.	1	45	45
7	Датчик контролю вологості ґрунту	од.	1	30	30
8	Датчик температури води	од.	1	42	42
9	Датчик температури ґрунту	од.	1	42	42
10	Блок живлення +5В	од.	1	29	29
11	Блок живлення +12В	од.	1	185	185
12	Резистор 4,7 кОм	од.	2	1	2
13	Резистор 10 кОм	од.	2	1	2
14	Резистор 1 кОм	од.	2	1	2
15	Резистор 2 кОм	од.	1	1	1
16	Водо нагрівач	од.	1	110	110
17	Освітлення	од.	2	80	160
18	Вентилятор	од.	1	46	46
19	Насос	од.	1	158	158
Разом					1254

Витрати на транспортно-заготівельні і складські витрати визначаються по всіх розділах в залежності від вартості обладнання матеріалів, виробів, конструк-

ції, беруться 8 % від загальної вартості.

$$Д_{тр} = С_{об} \times 0,08 \quad (5.2)$$

де, $С_{об}$ – вартість комплектуючих, грн.

Таким чином витрати на транспортно-заготівельні і складські роботи складають

$$Д_{тр} = 1254 \times 0,08 = 100,32 \text{ грн}$$

Вартість монтажно-налагоджувальних робіт приймаємо на рівні 7 % від вартості обладнання.

$$М_{МН} = С_{об} \times 0,07 \quad (5.3)$$

Витрати на монтажно-налагоджувальні роботи складуть

$$М_{мн} = 1254,00 \times 0,07 = 87,78 \text{ грн.}$$

Таким чином капітальні витрати складаються:

$$К_{ПР} = 1254,00 + 100,32 + 87,78 = 1442,1 \text{ (грв.)}$$

5.3 Розрахунок капітальних витрат на програмне забезпечення

5.3.1 Розрахунок часу на розробку програмного забезпечення

Трудомісткість розробки програмного забезпечення розраховується за формулою:

$$t = t_o + t_u + t_a + t_n + t_{от} + t_g \quad (5.4)$$

де t_o - витрати праці на підготовку і опис поставленого завдання;

t_u - витрати праці на дослідження алгоритму рішення завдання;

t_a - витрати праці на обробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{от}$ - витрати праці на налаштування програм на ЕОМ;

t_g - витрати праці на підготовку документації по завданню.

Складові витрат праці визначаються на підставі умовної кількості оброблюваних операторів у програмному забезпеченні.

Умовне кількість операторів у програмі:

$$Q = q \times c(1 + p), \tag{5.5}$$

де q – кількість операторів, які у програмі, приймаємо q = 1 (виходячи з ПЗ на мові Matlab);

c – коефіцієнт складності програми;

p – коефіцієнт корекції програми в процесі її обробки.

Коефіцієнт складності «c» програми визначає відносну складність програми по відношенню до типового завданням, складність якого відповідає 1. Приймаємо c=1,35

Коефіцієнт корегування програми «p» визначає збільшення обсягу робіт за рахунок внесення змін в алгоритм або програму в результаті уточнення постановки завдання. Величина p приймемо рівною 0,1.

Таким чином, для програми, описаної в дипломному проекті:

$$Q = 1*1,35(1+0,1) =148,5$$

Оцінка витрат праці на підготовку і опис завдання в даному дипломному проекті складають 35 люд.-годин.

Витрати праці на вивчення опису завдання визначаються з урахуванням уточнення опису та кваліфікації програміста за формулою:

$$t_u = \frac{Q \cdot B}{(75...85) \cdot k}, \text{ люд.-годин} \tag{5.6}$$

де B - коефіцієнт збільшення витрат праці приймаємо B = 1,5;

k - коефіцієнт кваліфікації програміста, які визначається залежно від стажу роботи за спеціальністю.

У нашому випадку коефіцієнт кваліфікації програміста становить k=1,2. Для розроблювального програмного забезпечення:

$$t_u = \frac{148,5 * 1.5}{80 * 1.2} = 2 \text{ люд.} - \text{годин.}$$

Витрати на розробку алгоритму рішення завдання визначаються за формулою:

$$t_a = \frac{Q}{(20...25)\cdot k}, \text{ люд.-годин} \quad (5.7)$$

Для розроблювального програмного забезпечення:

$$t_a = \frac{148,5}{20 * 1.2} = 6 \text{ люд.} - \text{годин.}$$

Витрати праці на складання програми по готовій блок-схемі алгоритму визначаються за формулою:

$$t_n = \frac{Q}{(20...25)\cdot k}, \text{ люд.-годин} \quad (5.8)$$

Для розроблювального програмного продукту:

$$t_n = \frac{148,5}{20 * 1.2} = 6 \text{ люд.} - \text{годин.}$$

Витрати праці на налагодження програми на ЕОМ розраховуються за формулою:

$$t_{нал} = \frac{Q}{(4...5)\cdot k}, \text{ люд.-годин} \quad (5.9)$$

Для конкретного програмного продукту:

Витрати праці на підготовку документації по завданню визначаються за формулою:

$$t_Д = t_{ДР} + t_{ДО}, \text{ люд.-год,} \quad (5.10)$$

де $t_{ДР}$ – трудомісткість підготовки матеріалів до написання;

$t_{ДО}$ – трудомісткість редагування, друку та оформлення документації.

$$t_{ДР} = Q/(15...20)\cdot k, \quad (5.11)$$

$$t_{ДО} = 0,75\ t_{ДР} \quad (5.12)$$

$$t_{ДО} = 0,75 * 2,88 = 2 \text{ люд.-год.}$$

Для програмного забезпечення, що розроблено в дипломному проекті:

$$t_Д = 3 + 2 = 5 \text{ люд.-год.}$$

Трудомісткість розробки програмного забезпечення становитиме:

$$t = 30 + 1 + 3 + 3 + 10 + 5 = 57 \text{ людино-годин.}$$

5.3.2 Розрахунок витрат на розробку програмного продукту

Витрати на розробку програмного продукту включають витрати на заробітну плату розробника програми $З_{зп}$ і вартість машинного часу, необхідного для налаштування програми на ЕОМ $З_{мі}$

$$К_{пз} = З_{зп} + З_{мі}, \text{ грн.} \quad (5.13)$$

Заробітна плата розробника програмного забезпечення:

$$З_{зп} = t\,С_{пр}, \text{ грн.} \quad (5.14)$$

де t – загальна трудомісткість обробки програмного забезпечення;

$С_{пр}$ – середня годинна тарифна ставка програміста становить:

$$С_{пр} = 85 \text{ грн./час.}$$

Заробітна плата за розробку програмного забезпечення дорівнює:

$$З_{зп} = 57 * 85 = 4845 \text{ грн.}$$

Вартість машинного часу, необхідного для налаштування програми на ЕОМ:

$$З_{мв} = t_{нал}\ \ С_{мч}, \text{ грн.} \quad (5.15)$$

де $t_{нал}$ – трудомісткість налагодження програми на ЕОМ, людино-годин; $С_{Мч}$ - вартість машино-години ЕОМ, грн. / год. $С_{Мч} = 9$ грн. / год.

$$З_{МВ} = 10 * 9 = 90 \text{ грн.}$$

Витрати на розробку програмного забезпечення системи керування становитимуть:

$$К_{ПЗ} = 4845 + 90 = 4935 \text{ грн.}$$

Очікувана тривалість розробки програмного забезпечення:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.} \quad (5.16)$$

де $В_к$ – кількість розробників, так як програма в дипломному проекті розроблялася однією людиною, то Вк = 1;

$F_р$ – місячний фонд робочого часу (Fр = 176 годин).

Визначимо тривалість розробки ПЗ:

$$t_{\text{нал}} = \frac{57}{1*176} = 0{,}32 \text{ міс.}$$

Розрахувавши всі показники, використовуємо формулу 5.1 і розраховуємо капітальні витрати:

$$К_{пр} = 1254 + 100{,}32 + 87{,}78 + 4935 = 6\ 377{,}1 \text{ грн.}$$

Вартість системи керування що знаходиться в експлуатації становить 23005,5 грн. Використовувана система в розрахунках прийнята за базовий варіант.

$$\Delta К = К_{пр} – К_{баз} = 6\ 377{,}1 – 1442{,}1 = 4\ 929 \text{ грн.}$$

5.3.3 Розрахунок експлуатаційних витрат

Річні експлуатаційні витрати розраховуються за формулою:

$$C_e = C_a + C_з + C_с + C_{р.о.} + C_{ее} + C_{інш}\ , \qquad (5.17)$$

де C_e - річні поточні витрати, пов'язані із застосуванням системи керування;

C_a – амортизація основних фондів;

$C_з$ – заробітна плата обслуговуючого персоналу;

$C_с$ – відрахування на соціальні заходи;

$C_{р.о}$ – витрати на технічне обслуговування та поточний ремонт обладнання;

$C_{ее}$ – вартість електроенергії;

$C_{інш}$ – інші витрати.

Визначимо експлуатаційні витрати при впровадженні системи керування підтримки мікроклімату в кімнатній теплиці.

5.3.4 Амортизація основних фондів

Залежно від групи, до якої віднесено той, чи інший об’єкт основних засобів, встановлено мінімально-допустимі строки їх амортизації

Обладнання, розробленої в дипломному проекті системи керування, належить до 4 групи (машини та обладнання). Передбачуваний термін експлуатації системи становить 3 років.

При використанні методу прискореного зменшення залишкової вартості норма амортизації визначається за формулою:

$$H_a = (2 / T)*100\% \quad (5.18)$$

T – термін корисного використання об'єкта;

H_a – норма амортизації;

$$C_a = (ПВ * H_a)/100\%, \quad (5.19)$$

C_a – амортизація основних фондів (річна);

$ПВ$ – первинна вартість, дорівнює капітальним витратам $ПВ = К$;

Отже, норма амортизації для проектованої системи керування складе:

$$H_a = (2/5)*100\% = 40\%$$

Сума амортизації для проектованої системи становитиме:

$$C_{a.} = (1442,1* 40\%)/100\% = 576,84 \text{ грн.}$$

5.3.5 Розрахунок фонду заробітної плати

Номінальний річний фонд робочого часу одного працівника:

$$T_{ном.рік} =(T_к – T_{вих.св} – T_{відп})*T_{зм}, \text{ годин} \quad (5.20)$$

де, $T_к$ – календарний фонд робочого часу, 365 днів;

$T_{вих.св}$ – вихідні дні та свята, 114 дні;

$T_{відп}$ – відпустка, 21 день;

$T_{зм}$ – тривалість зміни, 8 год.

Таким чином, річний фонд робочого часу працівника складе:

$$T_{ном.рік} = (365 – 114 – 21)*8 = 1840 \text{ годин}$$

Для керування процесом задіяні 1 оператор ЛМІ, 2 технологи і 1 спеціаліст з електроустаткування.

Після впровадження проектованої системи керування штат персоналу не зміниться, отже заробітна плата і відрахування на соціальні заходи будуть однакові.

Розрахунок річного фонду заробітної плати виробничих робітників здійснюється у відповідності з формою, наведеною в таблиці 5.2.

Таблиця 5.2 - Розрахунок заробітної плати персоналу

№ п/п	Найменування професії робітників	Число працюючих, чол	Годинна тарифна ст грн. / ч.	Номінальний річний фонд робочого часу (годину)	Пряма заробітна плата, грн.	Додаткова заробітна плата	Доплати (7%), грн.	Всього заробітна плата, грн.
1	Оператор процесу	1	30	1840	55200	5520	3864	64584
2	Інженер з обслуговування	1	32	1840	58880	5888	4121,6	68889,6
	Разом							113473,6

$$C_{\text{с}} = 0{,}22 \cdot 113473{,}6 = 24\,964{,}192 \text{ (грн)}$$

5.3.6 Відрахування на соціальні заходи

Відрахування на соціальні заходи визначаються за формулою:

$$C_с = 0{,}22 * C_з \qquad (5.21)$$

$$C_{с.пр} = C_{с.баз} = 0{,}22 * 24964{,}192 = 5\,492{,}12 \text{ грн.}$$

5.3.7 Розрахунок витрат на технічне обслуговування та ремонт

Витрати на технічне обслуговування та поточний ремонт обладнання та мережі приймаємо на рівні 5% від величини капітальних витрат:

$$C_{то.тр} = 0{,}05 \square К \qquad (5.22)$$

$$C_{р.о.} = 0{,}05 * 1442{,}16 = 72.11 \text{ грн.}$$

5.3.8 Витрати на електроенергію

Розрахуємо вартість електроенергії, споживаної системою керування, розробленої у проекті:

$$C_{ее} = К_е * К_{др} * ds * Т \qquad (5.23)$$

де $К_е$ – кількість електроенергії, спожите проектованої системою керування за годину, 0,4 кВт * год;

$К_{др}$ – кількість робочих днів у році, $К_{др}$ = 365 –114= 251день;

ds – тривалість зміни, 8 годин;

Т – тариф на електроенергію для підприємств (Для користувачів електроенергії 2 класу тариф складає 2,26 грн. за кВт без ПДВ. З урахуванням ПДВ тариф Т = 2,26 * 1,2 = 2,712грн).

$$С_{ее.пр} = 0,1*251*8*2,712= 544,57 \text{ грн.}$$

5.3.9 Інші витрати

Інші витрати з експлуатації об'єкта проектування включають витрати з охорони праці, на спецодяг та інше згідно практиці, ці витрати визначаються в розмірі 4% від річного фонду заробітної плати обслуговуючого персоналу:

$$С_{інш} = С_з \cdot 0,04 \text{ грн.} \quad (5.24)$$

$$С_{інш.} = 113473,6* 0,04 = 4538,94\text{грн.}$$

За формулою 4.17 розраховуємо річні експлуатаційні витрати для проектного та базового варіантів:

Таблиця 5.3 – Експлуатаційні витрати

Назва покажчика	Проектний варіант
Амортизація, грн	576,84
Фонд заробітної плати, грн	113473,61
Відрахування на соц. виплати, грн	5492,12
Ремонт та тех. обслуговування, грн	72,11
Електроенергія, грн	403,61
Інше, грн	4538,94
Разом, грн	159915,96

Виходячи з розрахунків, видно, що капітальні витрати склали 1442,1 грн., а річні експлуатаційні витрати 159915,96 грн. Таким чином, впровадження нового обладнання та його експлуатація є дуже коштовними в матеріальному плані, але необхідними, оскільки встановлення нової системи керування дозволить зменшити витрати на всіх етапах технологічного процесу, а також підвищити відсоток продукту вищого ґатунку.

5.4 Визначення додаткового прибутку від впровадження системи керування

Середньодобове збільшення виходу готової продукції при впровадженні розробленої системи керування складе 7%, за рахунок зниження простоїв обладнання.

Визначимо додатковий прибуток від впровадження проектованої системи керування:

$$\Delta П = (Ц_{пр.} - S_{пр.}) \times Q_{пр} - (Ц_{баз.} - S_{баз.}) \times Q_{баз.},$$

де $S_{пр.}$, $S_{баз.}$ – Собівартість обробки 1 т. продукції за проектним і базовим варіантами, грн. / од. (1254*20%=1504,8 грн.);

$Ц_{пр.}$, $Ц_{баз.}$ – Ціна за 1 т. продукції, грн., З урахуванням торгової надбавки 20% складе 1442,1*20%=1730,52грн.;

$Q_{пр.}$, $Q_{баз}$ – обсяг виробництва продукції, кг. На даний момент продуктивність обробки становить 1 кг. на добу. При 251 робочому дні обсяг виробленої продукції складе:

$$Q_{баз.} = 1 * 251 = 251 \text{ кг/рік.}$$

При впровадженні системи керування:

$$Q_{пр.} = 1{,}07 * 1 * 251 = 268{,}57 \text{ кг/ рік}$$

Додатковий прибуток від впровадження системи керування

$\Delta П = (1730{,}52 - 1504{,}8) * 268{,}57 - (1442{,}1 - 1254) * 251 = 13408{,}52$ грн.

5.5 Оцінка економічної ефективності проекту

Визначимо показники економічної ефективності проектованої системи керування:

– річний економічний ефект:

$$Е = \Delta П - \Delta С - \Delta К * Е_п > 0 \qquad (5.26)$$

– економічна ефективність:

$$Е_г = \Delta П - \Delta С \qquad (5.27)$$

– і термін окупності розробки:

$$Т_{ок} = \Delta К / Е_г \qquad (5.28)$$

$$E_п = (N_{кр} – N_{інф}) /100 \quad (5.29)$$

де, $N_{кр}$ – річна процентна ставка,%;

$N_{інф}$ – річний рівень інфляції,%.

В якості нормативного значення приймемо величину банківської кредитної ставки $N_{кр}$ (18%) з урахуванням інфляції $N_{інф}$ (1,2%), тобто:

$$E_п = (18 – 1,2) /100 = 0.168$$

$$E = 13408,52– 295,2 – 0,168*4\ 929 = 12\ 285,25грн.$$

$$E_г = 13408,52– 295,2 = 13\ 113,32грн.$$

$$T_{ок} = 4\ 929 /576,84 = 8,54\ року$$

Коефіцієнт ефективності капітальних витрат Є показує, скільки гривень додаткового прибутку (економії) приносить одна гривня капітальних витрат:

$$Є = E_г / ΔK \quad (5.30)$$

Коефіцієнт ефективності становить:

$$Є = 576,84 /4\ 929 = 0,12$$

Отже, при впровадженні системи керування 1 грн. капітальних витрат приносить 2,5 грн. прибутку.

Економічні показники, що характеризують ефективність створення і використання розробленого проекту системи керування відображені в таблиці 5.4.

Таблиця 5.4 – Економічні показники

Найменування показників	Од. вимірювання	Показники базового варіанту системи	Показники проектного варіанту системи
1	2	3	4
Капітальні витрати	грн.	1442,1	1442,1
Експлуатаційні витрати, всього	грн.	159915,96	159915,96
В тому числі: - амортизація	грн.	576,84	576,84
- заробітна плата обслуговуючого персоналу	грн.	113473,6	113473,6
- відрахування на соціальні заходи	грн.	5 492,12	5 492,12
- технічне обслуговування та поточний ремонт системи керування	грн.	403,61	403,61
- вартість споживаної електроенергії	грн.	403,61	403,61
- інші витрати	грн.	4538,94	4538,94

1	2	3	4
Додатковий прибуток	грн.	-	576,84
Річний економічний ефект	грн.	-	13 113,32
Коефіцієнт ефективності			8,54
Термін окупності капітальних вкладень	р	-	0,12

ВИСНОВКИ

При впровадженні проектованої системи капітальні витрати складають 24575,5 грн. Річні експлуатаційні витрати, пов'язані з впровадженням системи 348290,898 грн. Проте, очікується додатковий прибуток у розмірі 576,84 грн та дуже швидкий термін окупності 8,54 року.

Виходячи з отриманих результатів, та звертаючи увагу на досить великий коефіцієнт ефективності, що дорівнює 0,12 можна зробити висновок, що впровадження проектованої комп’ютеризованої системи економічно вигідно.

6 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В кваліфікаційній роботі розробляється автоматизованого система керування кімнатною теплицею (АСК КТ). У якості об'єкта дослідження умов праці розглянуте робоче місце, у якому розташовано технологічне встаткування.

Завданням процесу керування є підтримка заданих параметрів (вологість та температура) в кімнатній теплиці, що дозволить підвищити якість товарної продукції та поліпшити інформаційне забезпечення АСК КТ.

6.1 Аналіз небезпечних і шкідливих факторів у приміщенні автоматизованого робочого місця оператора

Об'єктом дослідження в кваліфікаційній роботі є приміщення операторського пункту кімнатної теплиці. Приміщення, в якому знаходиться об'єкт дослідження, має одні двері й два віконні прорізи, а також відсутні внутрішні перегородки. Стіни та стеля побілені, підлога покрита ламінатом. Усередині приміщення розташовано одне робоче місце оператора АКСТ, що представляють собою робочий стіл з персональним комп'ютером. Приміщення має пожежну сигналізацію.

По небезпеці ураження електричним струмом приміщення належить до приміщень з підвищеною небезпекою, тому що існує можливість одночасного дотику до опалювальних батарей з'єднаних із землею та корпусами електричних апаратів.

У приміщенні операторського пункту присутні наступні шкідливі й небезпечні фактори:

– небезпека ураження електричним струмом при контакті зі струмоведучими проводами, корпусами електроустаткування, при переміщенні по виробничому приміщенню та корпусами персональних комп'ютерів, що опинилися під напругою в результаті пробою ізоляції;

– недостатня освітленість робочого місця, що приводить до зорового стомленню операторів;

– небезпека перегрівання блоків живлення, у разі яких може виникнути пожежа.

6.2 Інженерно-технічні заходи щодо охорони праці у приміщенні автоматизованого робочого місця оператора

Відповідно до правил і вимог по безпеці й охороні праці розроблені методи усунення шкідливих та небезпечних факторів.

Для забезпечення приміщення електричною енергією до нього підведена трифазна електрична мережа 50 Гц з напругою ~0,4 кВ с глухозаземленою нейтраллю. В приміщенні розташовані споживачі з напругою ~220 В, а усі електричні розетки, розташовані по периметру приміщення.

Споживачів електричної енергії (персональні комп’ютери) мають ймовірність пробою електричної фази на корпус та являють собою значну небезпеку для здоров’я й життя людей, які працюють із ними. Небезпеку, яку несуть електричні пристрої в аварійному режимі, можна усунути. Виходячи з того, що в мережах із глухозаземленої нейтраллю напругою до ~1 кВ захисне заземлення не ефективне (струм глухого замикання на землю залежить від опору заземлення), безпека забезпечується зануленням, яке зменшує тривалість режиму замикання на корпус. Для цього прокладається нульовий провід, який з'єднується із глухо заземленої нейтраллю джерела живлення, до якого й приєднують металеві корпуси всіх потенційних споживачів. При замиканні на корпус електроустановка переходить у режим короткого замикання, у результаті чого спрацьовує максимальний струмовий захист, який селективно відключає ушкоджену ділянку мережі. Крім того, занулення знижує потенціали корпусів, які з’являються в момент замикання на землю.

Струмовий захист реалізований з використанням автоматів, які розривають електричну мережу при високих струмових навантаженнях. У приміщенні є один центральний автомат, який дозволяє знеструмити все приміщення. При замиканні фази на корпус одного з комп’ютерів, відбувається відключення від мережі всіх споживачів.

У цілому для забезпечення електробезпечності в приміщенні регулярно проводяться інструктажі й перевірка всіх співробітників щодо знання техніки безпеки.

6.3 Розрахункова частина

Користуючись програмним забезпеченням DIALux evo 9.1 побудуємо приміщення в якому знаходиться об’єкт дослідження, зобразимо в ньому місце оператора та місцезнаходження АСК КТ.

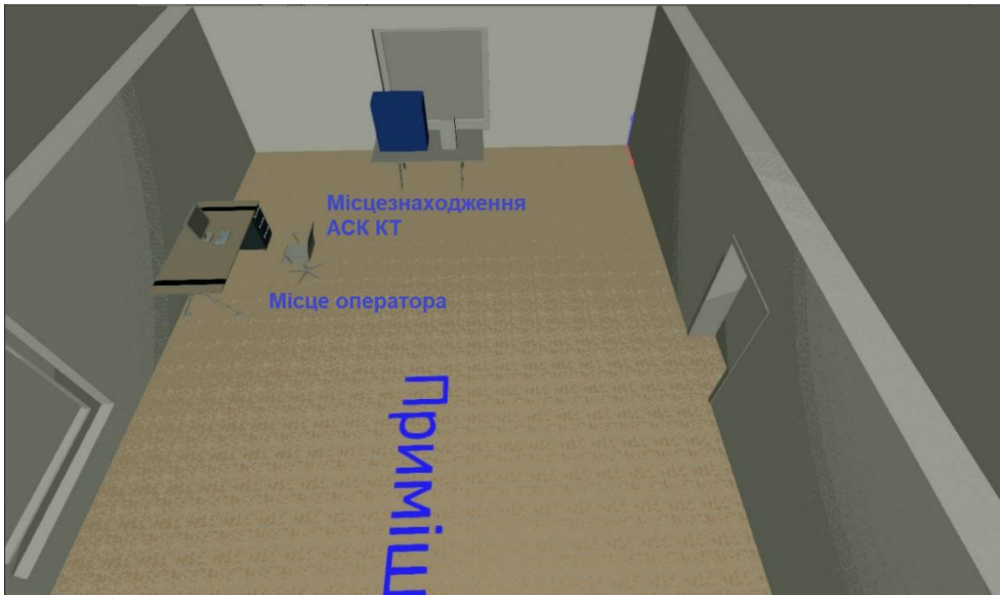


Рисунок 6.1 – Модель приміщення 3D

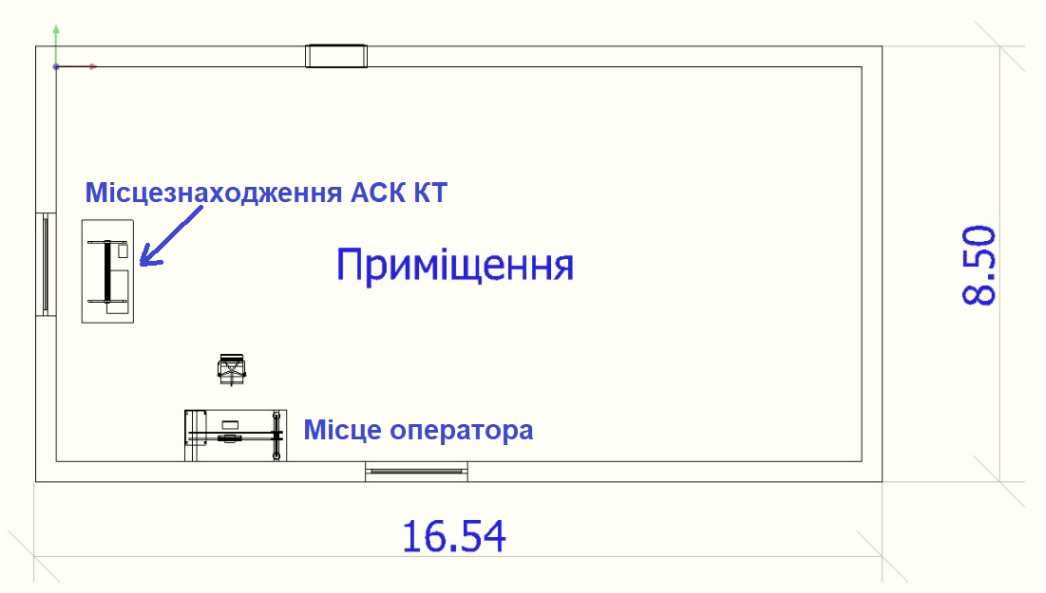


Рисунок 6.2 – Модель креслення приміщення

Для розрахунку освітленності у приміщенні робимо такі дії:

- Переходимо у вкладку "Світло" Вибираємо "Імпортувати файл світильника" Вибираємо на комп'ютері потрібний IES файл, підготовлений заздалегідь.
- Додаємо потрібний нам IES файл світильника в програму.
- Додаємо світильники в наше приміщення. Використовуємо інструмент "Автоматичний розподіл для зон" Даний інструмент працює в автоматичному режимі.
- Переглянемо наше приміщення у вигляді "3D візуалізація" для наочності.
- Програма автоматично розставляє кількість світильників під заданий рівень освітленості.

  У програмі автоматично встановлено 500лк
- Змінимо рівень освітленості. Кількість світильників скоротиться, отримаємо 3 світильника на 2х лінійках 1.546x0.067x0.064 м

  Програма автоматично розставляє світильники під заданий рівень освітленості. В нашому випадку це будуть активні світильники SRGSCB/1500 LED -Central.Line.Optic.

Інформація про "Світлодіодний лоток Regiolux SRGSCB / 1500 8000 830 DALI IP54 vw":

Рисунок 6.3 – SRGSCB/1500 LED -Central.Line.Optic

Зубчастий лоток з оцинкованої профільованої листової сталі, покритий поверхнею поліефірною смолою. Безінструментальне кріплення з інтегрованими в конструкцію затискачами забезпечує захист від крадіжки та демонтажу. Вбудова-

ні лицьові сторони з пластику з ущільнювальними губами та закритим по колу ущільненням для кріпильної шини для ступеня захисту до IP54 Прямий широкий розподіл світла за допомогою Central.Line.Optic з пластику ПММА. Оптична лінза містить прозорий ущільнювач, що забезпечує її легку збірку та зручність у обслуговуванні завдяки легко очищуваній поверхні. Щільність 1-рядового розташування лінз забезпечує злиття точок у лінію з однорідним виглядом на об’єкті. Електричне підключення за допомогою фіксованого 5-контактного роз'єму для швидкого монтажу з вільним вибором фази.

– Все готово для запуску розрахунків. Натиснувши на значок "Запуск розрахунків", запустимо процес розрахунку освітленості.
– Розрахунок завершено. Результати можна переглянути з правого боку, натиснувши на "трикутничок" розгорнуться результати освітленості на робочій площині.

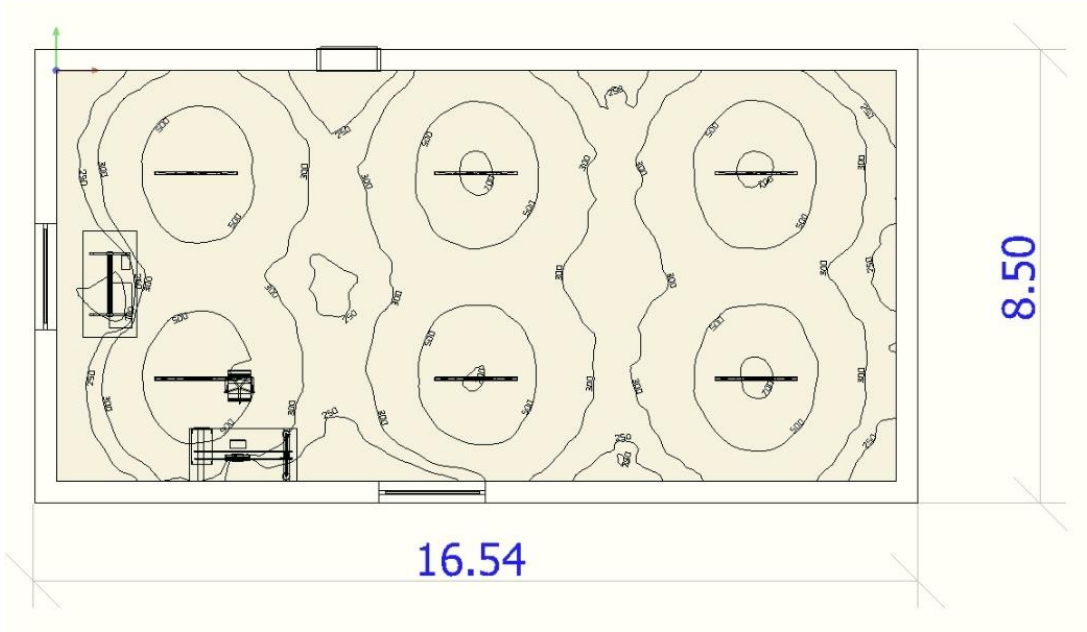


Рисунок 6.4 – Модель креслення освітленності приміщення

– Візуально в режимі 3D переглянемо розподіл світла. Відкриємо "Опції відображення" натиснувши на значок "монітора" В "Опціях відображення" виберемо режим "Відображення кривих розподілу світла"

После необходимо кликнуть на любой светильник, отобразится КСС светильника.

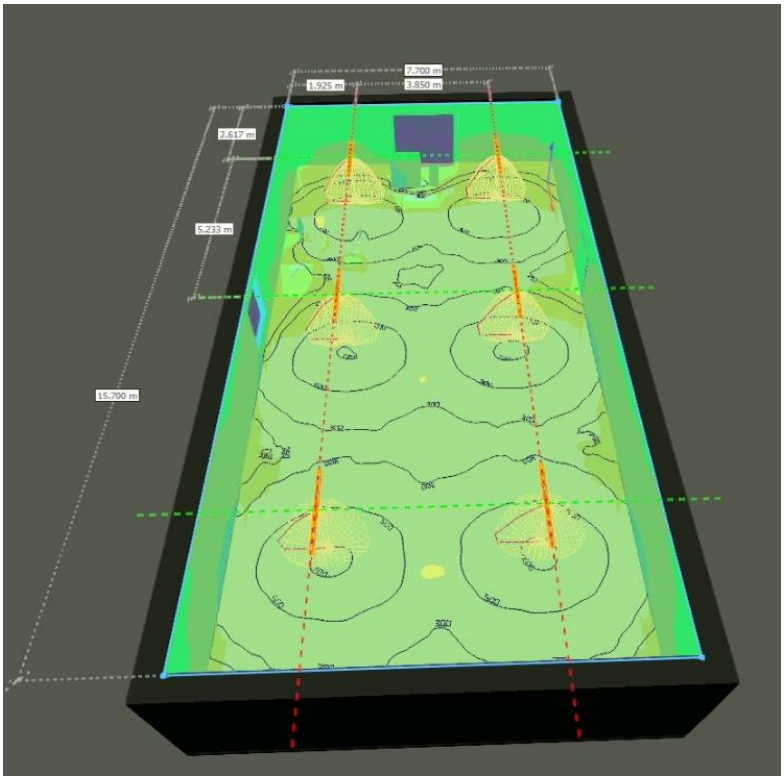


Рисунок 6.5 – Модель освітленності приміщення 3D

Згідно з розрахунками в приміщенні достатньо освітлення.

6.4 Пожежна профілактика

Ступінь вогнестійкості приміщення складає В-IIа. Основним засобом запобігання пожеж і вибухів від електроустаткування є правильний вибір і експлуатація встаткування.

Приміщення, у яких розташовуються персональні комп’ютери, повинні бути не нижче II ступеня вогнестійкості. У даному типі приміщень повинні бути медичні аптечки першої допомоги, система автоматичної пожежної сигналізації відповідно й пожежна сигналізація, з димовими пожежними оповіщувачами й переносними вуглекислотними вогнегасниками з розрахунків 2 штуки на кожні 20 м^2 площі приміщення з обліком гранично припустимих концентрацій вогнегасної рідини відповідно вимогам «Правил пожежної безпеки в Україні». В інших приміщеннях допускається встановлювати теплових пожежних оповіщувачів.

У якості засобу пожежогасіння обрано вуглекислотні вогнегасники ОУ-5. Вогнегасник ОУ-5 є переносним та має місткість балону 5 літрів, що відповідає 3,5 кілограма. Даний тип вогнегасників призначений для гасіння речовин горіння яких не може відбуватися без доступу повітря та електроустаткування, що працює під напругою менше 10 кВ. Перевагою вуглекислотних вогнегасників є відсутність слідів гасіння тому що вуглекислота після використання не залишає бруду. Вогнегасники не призначені для гасіння речовин, горіння яких може відбуватися без доступу повітря (алюміній, магній і їх сплави, натрій, калій).

Підходи до засобів пожежогасіння повинні бути вільними. У приміщенні передбачено 2 евакуаційних виходи.

При експлуатації встаткування неприпустимим вважається:

– експлуатація кабелів та проводів з ушкодженням або ти, що втратили захисні властивості за час експлуатації ізоляції; залишених під напругою кабелів та проводів з неізольованими провідниками;

– застосування для опалення приміщення нестандартного електронагрівального встаткування;

– використання електроапаратури й приладів в умовах, які не відповідають рекомендаціям підприємств-виготовлювачів.

У разі пожежі діяти згідно з планом евакуації

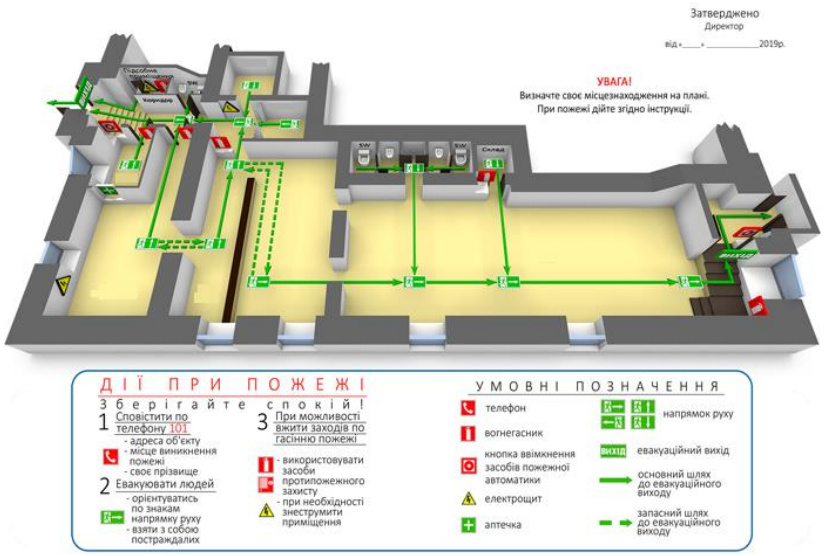


Рисунок 6.6 – Модель приміщення 3D

Негайно повідомити про пожежу оперативно-рятувальну службу цивільного захисту за телефоном 101. При цьому необхідно назвати адресу об’єкта, вказати кількість поверхів будівлі, характер та місце виникнення надзвичайної ситуації (напр., пожежі), обстановку, наявність людей, а також повідомити своє прізвище;

1. якщо поряд є ручний пожежний сповіщувач, привести його в дію натисканням на кнопку;
2. гучним голосом сповістити: «Тривога! Термінова евакуація!»;
3. сповістити керівництво закладу, повідомити чергового охоронця за телефоном;

за необхідності викликати інші аварійно-рятувальні служби (медичну, газорятувальну тощо)

Негайно і спокійно оголосити (повідомити) всьому персоналу закладу про необхідність термінової евакуації з усіх приміщень, використовуючи систему оповіщення будинку (за наявності). Перевірити, чи відкриті всі двері евакуаційних виходів та виходи до зовнішніх сходових кліток.

Усі евакуйованих із будівлі перевіряють за наявними поіменними списками груп. У разі виявлення відсутності когось із працівників, негайно з’ясовують, хто та де бачив його (її) востаннє, і передають цю інформацію представнику оперативно-рятувальної служби цивільного захисту, яка прибула до місця виклику.

Якщо немає безпосередньої загрози, організувати евакуацію матеріальних цінностей, згідно із заздалегідь розробленим планом.

6.5 Безпека при надзвичайних ситуаціях

Автоматизоване робоче місце оператора може бути піддане наступним видам надзвичайних ситуацій: пожежа; повінь; та інші.

6.5.1 Порядок дій при пожежі

Якщо пожежа застала вас у приміщенні, слід дотримуватись наступних правил:

- до дверей приміщення слід повзти підлогою під хмарою диму, але не відчиняти двері відразу;
- обережно доторкніться до дверей тильною стороною долоні, якщо двері не гарячі, то відчиніть їх та швидко виходьте;
- якщо двері гарячі, не відчиняйте їх – дим та полум’я не дозволять вам вийти;
- щільно закрийте двері, а всі щілини і отвори заткніть будь-якою тканиною, щоб уникнути подальшого проникнення диму. Повертайтесь поповзом у глибину приміщення і приймайте заходи до порятунку;
- присядьте та глибоко вдихніть повітря, розкрийте вікно, висуньтеся та спробуйте покликати за допомогою;
- якщо ви не в змозі розкрити вікно, розбийте віконне скло твердим предметом та зверніть увагу людей, які можуть викликати пожежну команду;
- якщо ви вибрались через двері, зачиніть їх та поповзом пересувайтесь до виходу із приміщення;
- обов’язково зачиняйте за собою всі двері;
- зверніть увагу, що під час пожежі заборонено користуватися ліфтами;
- якщо ви знаходитесь у висотному будинку, не біжіть вниз крізь вогнище, а користуйтеся можливістю врятуватися на даху будівлі.

Запам’ятайте: у всіх випадках, якщо ви в змозі, зателефонуйте «101» і викличте пожежну команду.

ВИСНОВКИ

1. В кваліфікаційній роботі об’єктом дослідження є процес вирощування рослин в кімнатній теплиці, предметом дослідження є автоматизація процесу керування мікрокліматом в кімнатній теплиці. Метою кваліфікаційної роботи є забезпечення підтримки оптимальних умов мікроклімату в кімнатній теплиці. В якості об’єкта керування виступає кімнатна теплиця.

2. Згідно до вимог проаналізовано існуючи пристрої керування котрі можуть бути використані при розробці контролера системи керування. За результатами аналізу в якості пристрою керування обрано одноплатний контролер WEMOS D1. Для контролера обрані модулі та джерела живлення. Виконано розробку схеми електричної принципової. На підставі обраного апаратного забезпечення та схеми виготовлено контролер системи керування мікрокліматом в кімнатній теплиці.

3. На підставі аналітичних засад в графічному середовищі імітаційного моделювання Simulink математичного пакету MATLAB розроблено модель об’єкта керування. За результатами аналізу моделі об’єкта обрано структуру системи керування та розроблено її модель на базі пропорційно-інтегрального регулятора з обмеженням керуючого впливу та запобіганням перенасичення інтегральної складової за методом “защіпки”.

4. Використовуючи модель системи керування виконано дослідження впливу параметрів регулятора на якість функціонування системи керування. У результаті чого отримані залежності показників якості від коефіцієнта підсилення та часу інтегрування. Виконано обґрунтування обрання параметрів отриманого регулятора.

5. Відповідно до технологічного процесу для регулятора розроблено імпульсний перетворювач, виконано дослідження впливу періоду формування керуючого впливу на якість функціонування системи керування та обґрунтовано обрання значення період.

6. На підставі безперервної моделі регулятора розроблено його цифрову модель, за якою розроблено функцію реалізації пропорційно-інтегрального регулятора на мові програмування C++ для контролера WEMOS D1, яка була використана при розробці програмного забезпечення системи керування.

7. Для SCADA системи zenon розроблено людино-машинний інтерфейс який відображає режим роботи системи керування, час доби, стан датчиків та виконавчих пристроїв. Крім того він дозволяє у ручному режимі змінювати стан виконавчих пристроїв. Отриманий людино машинний інтерфейс за допомогою ZenMobile for zenon відображається на пристроях з операційними системи Android та iOS.

8. На мові програмування Java розроблено бібліотеку для зв’язку пристроїв з операційними системами Android та iOS з системою керування. Бібліотека може бути використана при розробці програмного забезпечення людино-машинного інтерфейсу системи керування. Подальшим розвитком кваліфікаційної роботи є розробка людино-машинного інтерфейсу системи керування з використанням отриманої бібліотеки для таких пристроїв.

ПЕРЕЛІК ПОСИЛАНЬ

1. Положення про навчально-методичне забезпечення освітнього процесу Національного технічного університету «Дніпровська політехніка» / Укладачі: Ю.О. Заболотна, Є.А. Коровяка, В.О. Салов;М-во освіти і науки України, Нац. техн. ун-т. «Дніпровська політехніка» – Д. : НТУ «ДП», 2018. – 23 с.

2. Положення про організацію атестації здобувачів вищої освіти НТУ «Дніпровська політехніка» / Укладачі: Ю.О. Заболотна, О.О. Конопльова, В.О. Салова , В.О. Салов; М-во освіти і науки України, Нац. техн. ун-т. «Дніпровська політехніка» – Д. : НТУ «ДП», 2018. – 40 с.

3. Стандарт вищої освіти України. Рівень вищої освіти перший (бакалаврський) рівень. Ступінь вищої освіти бакалавр. Спеціальність 151 Автоматизації та комп'ютерно-інтегровані технології. МОН України. – Київ. – 2018. – 17 с.

4. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання / Нац. стандарт України. – Вид. офіц. – [чинний від 2017-07-01]. – Київ : ДП «УкрНДНЦ», 2016. – 27 с.

5. ДСТУ 1.5:2015. Правила розроблення. Викладання та оформлення національних нормативних документів оформлювання / Нац. стандарт України. – Вид. офіц. – [чинний від 2017-02-01]. – Київ: ДП «УкрНДНЦ», 2016. – 61 с.

6. ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання / Нац. стандарт України. – Вид. офіц. – [Уведено вперше ; чинний від 2016-07-01]. – Київ : ДП «УкрНДНЦ», 2016. – 17 с.

7. ДСТУ Б А.2.4-16:2008. Система проектної документації для будівництва. Автоматизація технологічних процесів. Зображення умовні приладів і засобів автоматизації в схемах / Нац. стандарт України. – Вид. офіц. – [Уведено вперше ; чинний від 2010-01-01]. Київ : ДП «УкрНДНЦ», 2008. – 10 с.

8. ГОСТ 2.710-81. Единая система конструкторской документации. Обозначения буквенно-цифровые в электрических схемах / Государственный комитет СССР по стандартам. – Вид. офіц. – [Уведено вперше ; чинний від 1981-07-01]. – М. : Издательство стандартов, 1989. – 17 с.

9. Камнев В.Н. Чтение схем и чертежей электроустановок: Практ. пособие для ПТУ. – 2-е. изд., перераб. и доп. / В.Н. Камнев – М.: Высш. шк. , 1990. – 144 с.

10. Дипломне проектування. Методичні рекомендації для студентів спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології / Упоряд.: В.В. Ткачов, , А.В. Бубліков, Л.І. Цвіркун, С.М. Проценко, О.О. Бойко. Д.В. Славінський.– Дніпро : НГУ, 2017. – 29 с.

11. Бойко О.О. Методичні вказівки до лабораторних робіт з теорії автоматичного управління для студентів напрямку підготовки «Комп'ютерна інженерія» / Укл.: О.О. Бойко – Д.: Державний ВНЗ «НГУ», 2017. – 107 с. – Режим доступу: https://goo.gl/nUMtFE. – Назва з домашньої сторінки Інтернету.

12. Ковалев, И.В. Автоматизированные системы управления: Учебное пособие для лекционных занятий для студентов специальности 210200 «Автоматизация технологических процессов и производств», всех форм обучения / И.В. Ковалев, Г.В. Волкова – Красноярск: СибГТУ. – 2006. – 179 с.100.

13. Токмаков Н. М. Математическая модель системы управления микроклиматом ангарных теплиц / Н. М. Токмаков, В. С. Грудинин // Гавриш №3. — М. : Научно-исследовательский институт овощеводства защищенного грунта (НИИОЗГ), 2008. — С. 28—32.

14. Ковалев, И.В. Автоматизированные системы управления: Учебное пособие для лекционных занятий для студентов специальности 210200 «Автоматизация технологических процессов и производств», всех форм обучения / И.В. Ковалев, Г.В. Волкова – Красноярск: СибГТУ. – 2006. – 179 с.

15. Токмаков Н. М. Математическая модель системы управления микроклиматом ангарных теплиц / Н. М. Токмаков, В. С. Грудинин // Гавриш №3. — М. : Научно-исследовательский институт овощеводства защищенного грунта (НИИОЗГ), 2008. — С. 28—32.

16. Парр Э. Програмируемые контролеры: руководство для инженера / Э. Парр. – М.: БИНОМ. Лаборатория знаний, 2007. – 516 с.

17. Парк Дж. Сбор данных в системах контроля и управления. Практическое руководство / Дж. Парк, С. Маккей. – М.: ООО «Группа ИДТ», 2006. – 504 с.

18. Минаев И.Г. Программируемые логические контроллеры. Практическое руководство для начинающего инженера / И.Г. Минаев, В.В. Самойленко. – Ставрополь: АРГУС, 2009. – 100 с.

19. Погрібняк І.О., Проектування та розробка програмного забезпечення промислових контролерів на базі графів станів та мови structured text / І.О. Погрібняк, О.О. Бойко. Міжнародна науково-практична конференція енергозбереження та енергоефективність 2019. 28-29 листопада 2019 року. – Дніпро: НТУ “ДП”, 2019. – С. 88-89.

20. Погрібняк І.О., Проектування програмного забезпечення системи керування процесом підтримки мікроклімату в кімнатній теплиці / І.О. Погрібняк, М.В. Козарь, О.О. Бойко. Міжнародна науково-практична конференція енергозбереження та енергоефективність 2019. 28-29 листопада 2019 року. – Дніпро: НТУ “ДП”, 2019. – С. 90-92.

21. Погрібняк І.О., Автоматизированная система управленияподдержани-ем микроклимата в складском помещении / І.О. Погрібняк, М.В. Козарь. Матеріали студентської науково-технічної конференції 2019 р. – Д.: Національний технічний університет «Дніпровська політехніка», 2019. – С. 98-100.

ДОДАТОК А – ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ КЕРУВАННЯ МІКРОКЛІМАТОМ В КІМНАТНІЙ ТЕПЛИЦІ

A.1 Greenhouse.ino

```
/*****************************************************************************
    Програма реалізації системи керування
*****************************************************************************/

#include "Controller.h" // Функція пристрою керування

void setup(void)
{
  g_Hardware.setup(); // Налаштування контролера
}

/*****************************************************************************
    Основна функція контролера
*****************************************************************************/

void loop(void)
{
  g_Hardware.update(); // Оновлення стану апаратних пристроїв
  controller(); // Виклик функції реалізації системи керування
}
```

A.2 Controller.h

```
/*****************************************************************************
    Функція реалізації системи керування
*****************************************************************************/

#include "PIController.h" // Функція пропорційно-інтегрального регулятора
#include "Hardware.h" // Налаштування апаратного забезпечення

Hardware g_Hardware; // Апаратне забезпечення

void controller()
{
  if (g_Hardware.getTypeWork() == 0) // Ручний режим
  {
    if (g_Hardware.getHeatWater() == 1) // Нагрів води
    {
        if (g_Hardware.getWaterLevel() == 0) // Немає води
          g_Hardware.clearHeaterWater(); // Відключення нагріву

        if (g_Hardware.getWaterTemperature() >=
            g_Hardware.getSetPointDayWaterTemperatureMax()) // Вода нагріта
            g_Hardware.clearHeaterWater(); // Відключення нагріву
    }

    if (g_Hardware.getPump() == 1) // Працює насос
    {
        if (g_Hardware.getWaterLevel() == 0) // Немає води
            g_Hardware.clearHeaterWater(); // Відключення нагріву

        // Вода холодна чи гаряча
        if (g_Hardware.getWaterTemperature() <=
            g_Hardware.getSetPointDayWaterTemperatureMin()
            || g_Hardware.getWaterTemperature() >
            g_Hardware.getSetPointDayWaterTemperatureMax())
            g_Hardware.clearPump(); // Відключення насоса
```

```
  }
}
else
{
  // Ніч
  if ((g_Hardware.getHours() < g_Hardware.getSetPointDayHours()
       && g_Hardware.getMinutes() < g_Hardware.getSetPointDayMinutes())
      || (g_Hardware.getHours() >= g_Hardware.getSetPointNightHours()
       && g_Hardware.getMinutes() >= g_Hardware.getSetPointNightMinutes()))
  {
    g_Hardware.clearLight(); // Вимкнення освітлення

    if (g_Hardware.getAirHumidity() >
        g_Hardware.getSetPointNightAirHumidityMax()) // Вологість повітря висока
      g_Hardware.setFan(); // Ввімкнення вентилятора
    else
    {
      PIOutput result = piController(
                      (float)g_Hardware.getSetPointDayAirTemperature(),
                      g_Hardware.getAirTemperatureFloat()); // ПІ-регулятор
      if (result.fManipulatedValue == 0) // Керування вентилятором
        g_Hardware.setFan();
      else
        g_Hardware.clearFan();
    }
  }
  // День
  else
  {
      g_Hardware.setLight(); // Ввімкнення освітлення

      if (g_Hardware.getWaterLevel() == 0) // Немає води
          g_Hardware.clearHeaterWater(); // Вимкнення нагріву
      else if (g_Hardware.getWaterTemperature() <
               g_Hardware.getSetPointDayWaterTemperatureMin()) // Вода холодна
        g_Hardware.setHeaterWater(); // Ввімкнення нагріву

      if (g_Hardware.getWaterTemperature() >
          g_Hardware.getSetPointDayWaterTemperatureMax()) // Вода гаряча
          g_Hardware.clearHeaterWater(); // Вимкнення нагріву

      // Немає води або вода холодно або гаряча
      if (g_Hardware.getWaterLevel() == 0
          || g_Hardware.getWaterTemperature() <
             g_Hardware.getSetPointDayWaterTemperatureMin()
          || g_Hardware.getWaterTemperature() >=
             g_Hardware.getSetPointDayWaterTemperatureMax())
        g_Hardware.clearPump(); // Вимкнення насоса
      // Вологість ґрунту низька
      else if (g_Hardware.getGroundHumidity() <
               g_Hardware.getSetPointDayGroundHumidityMin())
        g_Hardware.setPump(); // Ввімкнення насоса

      // Вологість ґрунту велика
      if (g_Hardware.getGroundHumidity() >
          g_Hardware.getSetPointDayGroundHumidityMax())
          g_Hardware.clearPump(); // Вимкнення насоса

      if (g_Hardware.getAirHumidity() >
          g_Hardware.getSetPointDayAirHumidityMax()) // Вологість повітря висока
        g_Hardware.setFan(); // Включення вентилятора
      else // Вологість повітря нормальна
      {
        PIOutput result = piController(
```

```
                            (float)g_Hardware.getSetPointDayAirTemperature(),
                             g_Hardware.getAirTemperatureFloat()); // ПІ-регулятор
          if (result.fManipulatedValue == 0) // Керування вентилятором
            g_Hardware.setFan();
          else
            g_Hardware.clearFan();
        }
    }
  }
}
```

A.3 PIController.h

```
/*************************************************************************
    Функція пропорційно-інтегрального регулятора
*************************************************************************/

struct PIOutput // Результат розрахунку керуючого впливу
{
  float fManipulatedSignal; // Керуючий сигнал
  float fManipulatedValue; // Керуючий вплив
};

struct PIData // Дані регулятора
{
  float Integrator_DSTATE; // Стан інтегратора
  float sTime; // Поточний час
  float sSwitchTime; // Час перемикання
  float sOutput; // Стан виходу
} g_piData;

PIOutput piController(float fSetPoint, float fProcessValue)
{
  PIOutput result;

  float rtb_Sum;
  float rtb_ProportionalGain;
  float rtb_IntegralGain;
  float rtb_Saturation;
  bool rtb_Equal1;
  bool rtb_Equal2;
  float rtb_DeadZone;
  float y;
  float y_0;

  rtb_IntegralGain = fProcessValue - fSetPoint; // Помилка керування

  if (rtb_IntegralGain > 0.5) // Зона нечуствітельності
      rtb_IntegralGain = rtb_IntegralGain - 0.5f;
  else if (rtb_IntegralGain >= -0.5)
      rtb_IntegralGain = 0.0;
  else
      rtb_IntegralGain = rtb_IntegralGain - -0.5f;

  rtb_Sum = rtb_IntegralGain + g_piData.Integrator_DSTATE; // Керуючий сигнал
  rtb_ProportionalGain = 0.3f * rtb_Sum; // Пропорційна складова
  rtb_IntegralGain = 0.0125f * rtb_IntegralGain; // Інтегральна складова

  if (rtb_ProportionalGain >= 1.0) // Обмеження керуючого сигналу
      rtb_Saturation = 1.0;
  else if(rtb_ProportionalGain > 0.0)
      rtb_Saturation = rtb_ProportionalGain;
  else
      rtb_Saturation = 0.0;
```

```
  if (rtb_ProportionalGain > 1.0) // Захист від перенасичення інтегратора
      rtb_DeadZone = rtb_ProportionalGain - 1.0f;
  else if(rtb_ProportionalGain >= 0.0)
      rtb_DeadZone = 0.0;
  else
      rtb_DeadZone = rtb_ProportionalGain;

  if (rtb_DeadZone < 0.0)
      y = -1.0;
  else if(rtb_DeadZone > 0.0)
      y = 1.0;
  else
      y = rtb_DeadZone;

  if (rtb_IntegralGain < 0.0)
      y_0 = -1.0;
  else if(rtb_IntegralGain > 0.0)
      y_0 = 1.0;
  else
      y_0 = rtb_IntegralGain;

  rtb_Equal1 = (y == y_0);

  if (rtb_ProportionalGain < 0.0)
      rtb_ProportionalGain = -1.0;
  else if(rtb_ProportionalGain > 0.0)
      rtb_ProportionalGain = 1.0;

  if (rtb_Sum < 0.0)
      rtb_Sum = -1.0;
  else if(rtb_Sum > 0.0)
      rtb_Sum = 1.0;
  rtb_Equal2 = (rtb_ProportionalGain == rtb_Sum);

  if ((0.0 != rtb_DeadZone) && ((rtb_Equal1 && rtb_Equal2)
       || ((!rtb_Equal1) && (!rtb_Equal2))))
      rtb_IntegralGain = 0.0;

  g_piData.sTime = g_piData.sTime - 0.01f; // Розрахунок часу
  if (g_piData.sTime < 0.0)
      g_piData.sTime = 0.0;

  if (g_piData.sTime = 0.0) // Час вийшов
  {
      g_piData.sTime = 1.0;
      g_piData.sSwitchTime = 1.0f - rtb_Saturation;
      g_piData.sOutput = 1.0;
  }

  if (g_piData.sSwitchTime >= g_piData.sTime) // Імпульс закінчився
      g_piData.sOutput = 0.0;

  result.fManipulatedSignal = rtb_Saturation; // Сигнал керування
  result.fManipulatedValue = g_piData.sOutput; // Керуючий вплив

  g_piData.Integrator_DSTATE = (0.01f * rtb_IntegralGain) +
  g_piData.Integrator_DSTATE; // Інтегральна складова

  return result; // Результат
}
```

A.4 Hardware.h

```
/*************************************************************************
    Настройки аппаратного обеспечения
*************************************************************************/

#include <DHT.h> // Бібліотека датчика температури та вологості повітря DHT22
#include <OneWire.h> // Бібліотека 1-Wire інтерфейса
#include <DallasTemperature.h> // Бібліотека датчика температури DS18B20
#include <LiquidCrystal_I2C.h> // Бібліотека LCD інтерфейса
#include <iarduino_RTC.h> // Бібліотека годинника реального часу DS3231
#include "ModbusTCPSlave.h" // Бібліотека Modbus TCP Slave
#include <eeprom.h> // Бібліотека роботи з EEPROM
#include <EEPROMAnything.h> // Бібліотека роботи з EEPROM

namespace HardwareConsts
{
  const unsigned int SensorGroundHumidity = A0; // Датчик вологості грунта
  const unsigned int SensorAirHumidityTemperature = D2; // Датчик температори та
вологості опвітря
  const unsigned int OneWire = D5; // 1-Wire інтерфейс
  const unsigned int SensorWaterLevel = D7; // Датчик рівня води
  const unsigned int ActuatorHeatWater = D6; // Нагрів води
  const unsigned int ActuatorLight = D8; // Освітлення
  const unsigned int ActuatorFan = D11; // Вентилятор
  const unsigned int ActuatorPump = D12; // Насос

  const unsigned int LCDAddress = 0x27; // Адреса LCD індикатора
  const unsigned int LCDSymbols = 16; // Кількість символів LCD індикатора
  const unsigned int LCDLines = 2; // Кількість рядів LCD індикатора

  const unsigned int ClockType = RTC_DS1307; // Тип годинника реального часа

  const char WiFiSSID[] = "GreenHouse"; // Назва точки доступу
  const char WiFiPassword[] = "1234567890"; // Пароль до точки доступу
  const byte WiFiIP[] = {192, 168, 1, 1}; // IP адреса
  const byte WiFiGateway[] = {192, 168, 1, 1}; // Шлюз
  const byte WiFiSubnet[] = {255, 255, 255, 0}; // Маска підмережі

  const unsigned int SerialSpeed = 115200; // Швидкість послідовного інтерфейсу

  const unsigned int AirHumidityTemperatureCounter = 5; // Кількість циклів до
опитування датчика повітря
  const unsigned int GroundWaterTemperaturesCounter = 5; // Кількість циклів до
опитування датчиків температури води та землі
  const unsigned int GroundHumidityCounter = 5; // Кількість циклів до опитування
датчика вологості грунту
  const unsigned int ClockCounter = 5; // Кількість циклів до опитування годин ре-
ального часу
  const unsigned int WaterLevelCounter = 10; // Кількість спрацьовувань датчика
рівня води
  const unsigned int LCDCounter = 10; // Кількість циклів до поновлення LCD
  const unsigned int CycleTime = 10; // Час циклу, мс
};

namespace LCDConsts
{
  const int LCDValueCounter   = 10; // Кількість відображень
  const int LCDValueCounter1  = 5; // Кількість відображень
  const int LCDValueCounter2  = 5; // Кількість відображень
  const int LCDState1         = 0; // Стан 1
  const int LCDState2         = 1; // Стан 2
};
```

```
class Hardware
{
  public:
    Hardware() :
      m_OneWire(HardwareConsts::OneWire),
      m_SensorsTemperature(&m_OneWire),
      m_LCD(HardwareConsts::LCDAddress, 2, 1, 0, 4, 5, 6, 7),
      m_Clock(HardwareConsts::ClockType),
      m_Modbus(0, 0, 32)
    {
      m_nAirHumidityTemperatureCounter = 0;
      m_nAirHumidity = 0;
      m_nAirTemperature = 0;
      m_nGroundWaterTemperaturesCounter = 0;
      m_nGroundTemperature = 0;
      m_nWaterTemperature = 0;
      m_nGroundHumidityCounter = 0;
      m_nGroundHumidity = 0;
      m_nClockCounter = 0;
      m_nHours = 0;
      m_nMinutes = 0;
      m_nWaterLevelCounter = 0;
      m_nWaterLevel = 0;
      m_nHeatWater = 0;
      m_nLight = 0;
      m_nFan = 0;
      m_nPump = 0;

      m_nTimeUpdate = 0;
      m_nTypeWork = 1;
      m_nTypeDay = 0;
      m_nSetPointDayHours = 6;
      m_nSetPointDayMinutes = 0;
      m_nSetPointDayAirHumidityMin = 80;
      m_nSetPointDayAirHumidityMax = 90;
      m_nSetPointDayAirTemperatureMin = 20;
      m_nSetPointDayAirTemperatureMax = 24;
      m_nSetPointDayAirTemperature = 22;
      m_nSetPointDayGroundHumidityMin = 80;
      m_nSetPointDayGroundHumidityMax = 95;
      m_nSetPointDayWaterTemperatureMin = 20;
      m_nSetPointDayWaterTemperatureMax = 25;
      m_nSetPointNightHours = 22;
      m_nSetPointNightMinutes = 0;
      m_nSetPointNightAirHumidityMin = 70;
      m_nSetPointNightAirHumidityMax = 80;
      m_nSetPointNightAirTemperatureMin = 18;
      m_nSetPointNightAirTemperatureMax = 22;
      m_nSetPointNightAirTemperature = 20;
      m_nSaveSetPoint = 0;

      m_nLCDCounter = 0;
      m_nLCDState = LCDConsts::LCDState1;
      m_nLCDValueCounter = 0;
      m_nLCDNeedUpdate = 1;

      int sizeValue = 2;
      EEPROM.begin(64);
      if (EEPROM_readAnything(0, sizeValue) != 0)
      {
        m_nSetPointDayHours = EEPROM_readAnything(0, sizeValue);
        m_nSetPointDayMinutes = EEPROM_readAnything(1, sizeValue);
        m_nSetPointDayAirHumidityMin = EEPROM_readAnything(2, sizeValue);
        m_nSetPointDayAirHumidityMax = EEPROM_readAnything(3, sizeValue);
```

```
        m_nSetPointDayAirTemperatureMin = EEPROM_readAnything(4, sizeValue);
        m_nSetPointDayAirTemperatureMax = EEPROM_readAnything(5, sizeValue);
        m_nSetPointDayAirTemperature = EEPROM_readAnything(6, sizeValue);
        m_nSetPointDayGroundHumidityMin = EEPROM_readAnything(7, sizeValue);
        m_nSetPointDayGroundHumidityMax = EEPROM_readAnything(8, sizeValue);
        m_nSetPointDayWaterTemperatureMin = EEPROM_readAnything(9, sizeValue);
        m_nSetPointDayWaterTemperatureMax = EEPROM_readAnything(10, sizeValue);
        m_nSetPointNightHours = EEPROM_readAnything(11, sizeValue);
        m_nSetPointNightMinutes = EEPROM_readAnything(12, sizeValue);
        m_nSetPointNightAirHumidityMin = EEPROM_readAnything(13, sizeValue);
        m_nSetPointNightAirHumidityMax = EEPROM_readAnything(14, sizeValue);
        m_nSetPointNightAirTemperatureMin = EEPROM_readAnything(15, sizeValue);
        m_nSetPointNightAirTemperatureMax = EEPROM_readAnything(16, sizeValue);
        m_nSetPointNightAirTemperature = EEPROM_readAnything(17, sizeValue);
      }
    }

    void setup()
    {
      Serial.begin(HardwareConsts::SerialSpeed); // Налаштування послідовного ін-
терфейсу

      pinMode(HardwareConsts::SensorWaterLevel, INPUT); // Ініціалізація контакту
датчика рівня води
      pinMode(HardwareConsts::SensorGroundHumidity, INPUT); // Ініціалізація кон-
татка датчика вологості грунту

      pinMode(HardwareConsts::ActuatorHeatWater, OUTPUT); // Ініціалізація контак-
ту виходу 1
      pinMode(HardwareConsts::ActuatorLight, OUTPUT); // Ініціалізація контакту
виходу 2
      pinMode(HardwareConsts::ActuatorFan, OUTPUT); // Ініціалізація контакту ви-
ходу 3
      pinMode(HardwareConsts::ActuatorPump, OUTPUT); // Ініціалізація контакту ви-
ходу 4

      m_SensorsTemperature.begin(); // Ініціалізація датчика температури DS18B20

      m_LCD.begin(HardwareConsts::LCDSymbols, HardwareConsts::LCDLines); // Ініці-
алізація LCD
      m_LCD.setBacklightPin(3, POSITIVE);
      m_LCD.setBacklight(HIGH);

      m_Clock.begin(); // Ініціалізація годинника реального часу

      m_Modbus.begin(HardwareConsts::WiFiSSID, // Ініціалізація точки доступу
             HardwareConsts::WiFiPassword,
             HardwareConsts::WiFiIP,
             HardwareConsts::WiFiGateway,
             HardwareConsts::WiFiSubnet);

      writeModbusData();
    }

    void update()
    {
      writeModbusData(); // Запис даних у буфер Modbus

      updateAirHumidityTemperature(); // Отримання параметрів повітря
      updateGroundWaterTemperatures(); // Отримання температури землі та води
      updateGroundHumidity(); // Отримання вологості грунту
      updateWaterLevel(); // Отримання рівня води
      updateTime(); // Отримання значення часу
      updateActuators(); // Оновлення стану виконавчих пристроїв
```

```
  updateLCD(); // Оновлення LCD
  m_Modbus.poll(); // Обробка запиту Modbus TCP

  readModbusData(); // Читання даних з буфера Modbus
}

unsigned int getAirHumidity() const
{
  return m_nAirHumidity;
}

unsigned int getAirTemperature() const
{
  return m_nAirTemperature;
}

float getAirTemperatureFloat() const
{
  return m_fAirTemperature;
}

unsigned int getGroundTemperature() const
{
  return m_nGroundTemperature;
}

unsigned int getWaterTemperature() const
{
  return m_nWaterTemperature;
}

unsigned int getGroundHumidity() const
{
  return m_nGroundHumidity;
}

unsigned int getHours() const
{
  return m_nHours;
}

unsigned int getMinutes() const
{
  return m_nMinutes;
}

unsigned int getWaterLevel() const
{
  return m_nWaterLevel;
}

unsigned int getHeatWater() const
{
  return m_nHeatWater;
}

void setHeaterWater()
{
  m_nHeatWater = 1;
}

void clearHeaterWater()
{
  m_nHeatWater = 0;
```

```
}

unsigned int getLight() const
{
  return m_nLight;
}

void setLight()
{
  m_nLight = 1;
}

void clearLight()
{
  m_nLight = 0;
}

unsigned int getFan() const
{
  return m_nFan;
}

void setFan()
{
  m_nFan = 1;
}

void clearFan()
{
  m_nFan = 0;
}

unsigned int getPump() const
{
  return m_nPump;
}

void setPump()
{
  m_nPump = 1;
}

void clearPump()
{
  m_nPump = 0;
}

unsigned int getTypeWork() const
{
  return m_nTypeWork;
}

unsigned int getSetPointDayAirTemperature() const
{
  return m_nSetPointDayAirTemperature;
}

unsigned int getSetPointNightAirTemperature() const
{
  return m_nSetPointNightAirTemperature;
}

unsigned int getSetPointDayWaterTemperatureMin() const
{
```

```
    return m_nSetPointDayWaterTemperatureMin;
  }

  unsigned int getSetPointDayWaterTemperatureMax() const
  {
    return m_nSetPointDayWaterTemperatureMax;
  }

  unsigned int getSetPointDayHours() const
  {
    return m_nSetPointDayHours;
  }

  unsigned int getSetPointDayMinutes() const
  {
    return m_nSetPointDayMinutes;
  }

  unsigned int getSetPointNightHours() const
  {
    return m_nSetPointNightHours;
  }

  unsigned int getSetPointNightMinutes() const
  {
    return m_nSetPointNightMinutes;
  }

  unsigned int getSetPointNightAirHumidityMin() const
  {
    return m_nSetPointNightAirHumidityMin;
  }

  unsigned int getSetPointNightAirHumidityMax() const
  {
    return m_nSetPointNightAirHumidityMax;
  }

  unsigned int getSetPointDayGroundHumidityMin() const
  {
    return m_nSetPointDayGroundHumidityMin;
  }

  unsigned int getSetPointDayGroundHumidityMax() const
  {
    return m_nSetPointDayGroundHumidityMax;
  }

  unsigned int getSetPointDayAirHumidityMin() const
  {
    return m_nSetPointDayAirHumidityMin;
  }

  unsigned int getSetPointDayAirHumidityMax() const
  {
    return m_nSetPointDayAirHumidityMax;
  }

private:
  void updateAirHumidityTemperature()
  {
    if (m_nAirHumidityTemperatureCounter != 0)
    {
      --m_nAirHumidityTemperatureCounter;
```

```
        return;
      }
      m_nAirHumidityTemperatureCounter = Hardw-
?reConsts::AirHumidityTemperatureCounter;

      if
(m_SensorAirHumidityTemperature.read22(HardwareConsts::SensorAirHumidityTemperatur
e) == DHTLIB_OK)
      {
        m_nAirHumidity = m_SensorAirHumidityTemperature.humidity;
        m_nAirTemperature = m_SensorAirHumidityTemperature.temperature;
        m_fAirTemperature = m_SensorAirHumidityTemperature.temperature;
      }
    }

    void updateGroundWaterTemperatures()
    {
      if (m_nGroundWaterTemperaturesCounter != 0)
      {
        --m_nGroundWaterTemperaturesCounter;
        return;
      }
      m_nGroundWaterTemperaturesCounter = Hardw-
?reConsts::GroundWaterTemperaturesCounter;

      m_SensorsTemperature.requestTemperatures(); // Отримання значень темератруи
від датчиків 1-Wire
      m_nGroundTemperature = m_SensorsTemperature.getTempCByIndex(0);
      m_nWaterTemperature = m_SensorsTemperature.getTempCByIndex(1);
    }

    void updateGroundHumidity()
    {
      if (m_nGroundHumidityCounter != 0)
      {
        --m_nGroundHumidityCounter;
        return;
      }
      m_nGroundHumidityCounter = HardwareConsts::GroundHumidityCounter;

      m_nGroundHumidity = analogRead(HardwareConsts::SensorGroundHumidity);
    }

    void updateTime()
    {
      String sTime = m_Clock.gettime("H:i"); // Отримання поточного часу
      char chHours[3] {sTime[0], sTime[1], '\0'};
      m_nHours = atoi(chHours);
      char chMinutes[3] {sTime[3], sTime[4], '\0'};
      m_nMinutes = atoi(chMinutes);
    }

    void updateWaterLevel()
    {
      if (digitalRead(HardwareConsts::SensorWaterLevel) == HIGH) // Вода на рівні
датчика
        ++m_nWaterLevelCounter;
      else // Вода нижче датчика
      {
        m_nWaterLevelCounter = 0;
        m_nWaterLevel = 0;
        return;
      }
```

```
      if (m_nWaterLevelCounter >= HardwareConsts::WaterLevelCounter) // Води до-
сить
      {
        m_nWaterLevelCounter = HardwareConsts::WaterLevelCounter;
        m_nWaterLevel = 1;
      }
    }

    void updateActuators()
    {
      digitalWrite(HardwareConsts::ActuatorHeatWater, !m_nHeatWater);
      digitalWrite(HardwareConsts::ActuatorLight, !m_nLight);
      digitalWrite(HardwareConsts::ActuatorFan, !m_nFan);
      digitalWrite(HardwareConsts::ActuatorPump, !m_nPump);
    }

    void updateLCD()
    {
      if (m_nLCDCounter != 0)
      {
        --m_nLCDCounter;
        return;
      }
      m_nLCDCounter = HardwareConsts::LCDCounter;

      switch (m_nLCDState)
      {
        case LCDConsts::LCDState1:
          if (m_nLCDValueCounter < LCDConsts::LCDValueCounter1)
          {
            ++m_nLCDValueCounter;

            if (m_nLCDNeedUpdate == 1)
            {
              m_LCD.clear();
              m_nLCDNeedUpdate = 0;
            }

            m_LCD.setCursor(0, 0);
            m_LCD.print("AH="); m_LCD.print(m_nAirHumidity); m_LCD.print("%");
            m_LCD.setCursor(10, 0);
            m_LCD.print("AT="); m_LCD.print(m_nAirTemperature); m_LCD.print("C");

            m_LCD.setCursor(0, 1);
            m_LCD.print("WL="); m_LCD.print(m_nWaterLevel);
            m_LCD.setCursor(10, 1);
            m_LCD.print("WT="); m_LCD.print(m_nWaterTemperature);
m_LCD.print("C");
          }
          else
          {
            m_nLCDState = LCDConsts::LCDState2;
            m_nLCDNeedUpdate = 1;
            m_nLCDValueCounter = 0;
          }
          break;

        case LCDConsts::LCDState2:
          if (m_nLCDValueCounter < LCDConsts::LCDValueCounter2)
          {
            ++m_nLCDValueCounter;

            if (m_nLCDNeedUpdate == 1)
            {
```

```
            m_LCD.clear();
            m_nLCDNeedUpdate = 0;
          }

          m_LCD.setCursor(0, 0);
          m_LCD.print("GH="); m_LCD.print(m_nGroundHumidity);
          m_LCD.setCursor(10, 0);
          m_LCD.print("GT="); m_LCD.print(m_nGroundTemperature);
m_LCD.print("C");

          m_LCD.setCursor(0, 1);
          m_LCD.print("HLFP="); m_LCD.print(m_nHeatWater); m_LCD.print(";");
m_LCD.print(m_nLight);
          m_LCD.print(";"); m_LCD.print(m_nFan); m_LCD.print(";");
m_LCD.print(m_nPump);
        }
        else
        {
          m_nLCDState = LCDConsts::LCDState1;
          m_nLCDNeedUpdate = 1;
          m_nLCDValueCounter = 0;
        }
        break;
    }
  }

  void writeModbusData()
  {
    m_Modbus.m_HoldingRegisters[0] = m_nAirHumidity;
    m_Modbus.m_HoldingRegisters[1] = m_nAirTemperature;
    m_Modbus.m_HoldingRegisters[2] = m_nGroundTemperature;
    m_Modbus.m_HoldingRegisters[3] = m_nGroundHumidity;
    m_Modbus.m_HoldingRegisters[4] = m_nWaterTemperature;
    m_Modbus.m_HoldingRegisters[5] = m_nWaterLevel;
    m_Modbus.m_HoldingRegisters[6] = m_nHours;
    m_Modbus.m_HoldingRegisters[7] = m_nMinutes;
    m_Modbus.m_HoldingRegisters[8] = m_nHeatWater;
    m_Modbus.m_HoldingRegisters[9] = m_nLight;
    m_Modbus.m_HoldingRegisters[10] = m_nFan;
    m_Modbus.m_HoldingRegisters[11] = m_nPump;
    m_Modbus.m_HoldingRegisters[12] = m_nTimeUpdate;
    m_Modbus.m_HoldingRegisters[13] = m_nTypeWork;
    m_Modbus.m_HoldingRegisters[14] = m_nTypeDay;
    m_Modbus.m_HoldingRegisters[15] = m_nSetPointDayHours;
    m_Modbus.m_HoldingRegisters[16] = m_nSetPointDayMinutes;
    m_Modbus.m_HoldingRegisters[17] = m_nSetPointDayAirHumidityMin;
    m_Modbus.m_HoldingRegisters[18] = m_nSetPointDayAirHumidityMax;
    m_Modbus.m_HoldingRegisters[19] = m_nSetPointDayAirTemperatureMin;
    m_Modbus.m_HoldingRegisters[20] = m_nSetPointDayAirTemperatureMax;
    m_Modbus.m_HoldingRegisters[21] = m_nSetPointDayAirTemperature;
    m_Modbus.m_HoldingRegisters[22] = m_nSetPointDayGroundHumidityMin;
    m_Modbus.m_HoldingRegisters[23] = m_nSetPointDayGroundHumidityMax;
    m_Modbus.m_HoldingRegisters[24] = m_nSetPointDayWaterTemperatureMin;
    m_Modbus.m_HoldingRegisters[25] = m_nSetPointDayWaterTemperatureMax;
    m_Modbus.m_HoldingRegisters[26] = m_nSetPointNightHours;
    m_Modbus.m_HoldingRegisters[27] = m_nSetPointNightMinutes;
    m_Modbus.m_HoldingRegisters[28] = m_nSetPointNightAirHumidityMin;
    m_Modbus.m_HoldingRegisters[29] = m_nSetPointNightAirHumidityMax;
    m_Modbus.m_HoldingRegisters[30] = m_nSetPointNightAirTemperatureMin;
    m_Modbus.m_HoldingRegisters[31] = m_nSetPointNightAirTemperatureMax;
    m_Modbus.m_HoldingRegisters[32] = m_nSetPointNightAirTemperature;
    m_Modbus.m_HoldingRegisters[33] = m_nSaveSetPoint;
  }
```

```
  void readModbusData()
  {
    m_nHeatWater = m_Modbus.m_HoldingRegisters[8];
    m_nLight = m_Modbus.m_HoldingRegisters[9];
    m_nFan = m_Modbus.m_HoldingRegisters[10];
    m_nPump = m_Modbus.m_HoldingRegisters[11];
    m_nTimeUpdate = m_Modbus.m_HoldingRegisters[12];
    m_nTypeWork = m_Modbus.m_HoldingRegisters[13];
    m_nTypeDay = m_Modbus.m_HoldingRegisters[14];
    m_nSetPointDayHours = m_Modbus.m_HoldingRegisters[15];
    m_nSetPointDayMinutes = m_Modbus.m_HoldingRegisters[16];
    m_nSetPointDayAirHumidityMin = m_Modbus.m_HoldingRegisters[17];
    m_nSetPointDayAirHumidityMax = m_Modbus.m_HoldingRegisters[18];
    m_nSetPointDayAirTemperatureMin = m_Modbus.m_HoldingRegisters[19];
    m_nSetPointDayAirTemperatureMax = m_Modbus.m_HoldingRegisters[20];
    m_nSetPointDayAirTemperature = m_Modbus.m_HoldingRegisters[21];
    m_nSetPointDayGroundHumidityMin = m_Modbus.m_HoldingRegisters[22];
    m_nSetPointDayGroundHumidityMax = m_Modbus.m_HoldingRegisters[23];
    m_nSetPointDayWaterTemperatureMin = m_Modbus.m_HoldingRegisters[24];
    m_nSetPointDayWaterTemperatureMax = m_Modbus.m_HoldingRegisters[25];
    m_nSetPointNightHours = m_Modbus.m_HoldingRegisters[26];
    m_nSetPointNightMinutes = m_Modbus.m_HoldingRegisters[27];
    m_nSetPointNightAirHumidityMin = m_Modbus.m_HoldingRegisters[28];
    m_nSetPointNightAirHumidityMax = m_Modbus.m_HoldingRegisters[29];
    m_nSetPointNightAirTemperatureMin = m_Modbus.m_HoldingRegisters[30];
    m_nSetPointNightAirTemperatureMax = m_Modbus.m_HoldingRegisters[31];
    m_nSetPointNightAirTemperature = m_Modbus.m_HoldingRegisters[32];
    m_nSaveSetPoint = m_Modbus.m_HoldingRegisters[33];

    if (m_nSaveSetPoint == 1)
    {
      EEPROM.begin(64);
      EEPROM_writeAnything(0, m_nSetPointDayHours);
      EEPROM_writeAnything(1, m_nSetPointDayMinutes);
      EEPROM_writeAnything(2, m_nSetPointDayAirHumidityMin);
      EEPROM_writeAnything(3, m_nSetPointDayAirHumidityMax);
      EEPROM_writeAnything(4, m_nSetPointDayAirTemperatureMin);
      EEPROM_writeAnything(5, m_nSetPointDayAirTemperatureMax);
      EEPROM_writeAnything(6, m_nSetPointDayAirTemperature);
      EEPROM_writeAnything(7, m_nSetPointDayGroundHumidityMin);
      EEPROM_writeAnything(8, m_nSetPointDayGroundHumidityMax);
      EEPROM_writeAnything(9, m_nSetPointDayWaterTemperatureMin);
      EEPROM_writeAnything(10, m_nSetPointDayWaterTemperatureMax);
      EEPROM_writeAnything(11, m_nSetPointNightHours);
      EEPROM_writeAnything(12, m_nSetPointNightMinutes);
      EEPROM_writeAnything(13, m_nSetPointNightAirHumidityMin);
      EEPROM_writeAnything(14, m_nSetPointNightAirHumidityMax);
      EEPROM_writeAnything(15, m_nSetPointNightAirTemperatureMin);
      EEPROM_writeAnything(16, m_nSetPointNightAirTemperatureMax);
      EEPROM_writeAnything(17, m_nSetPointNightAirTemperature);
      EEPROM.end();

      m_nSaveSetPoint = 0;
    }
  }

private:
  dht m_SensorAirHumidityTemperature; // Датчик температури та вологості повітря
  OneWire m_OneWire; // Інтерфейс 1-Wire
  DallasTemperature m_SensorsTemperature; // Датчики температури
  LiquidCrystal_I2C m_LCD; // LCD екран
  iarduino_RTC m_Clock; // Годинник реального часу
  ModbusTCPSlave m_Modbus; // Modbus
```

```
    unsigned int m_nAirHumidityTemperatureCounter; // Лічильник датчика повітря
    unsigned int m_nAirHumidity; // Вологість повітря
    unsigned int m_nAirTemperature; // Температура повітря
    unsigned int m_nGroundWaterTemperaturesCounter; // Лічильник датчиків темпера-
тури землі та води
    unsigned int m_nGroundTemperature; // Температура землі
    unsigned int m_nWaterTemperature; // Температура води
    unsigned int m_nGroundHumidityCounter; // Лічильник датчика вологості грунту
    unsigned int m_nGroundHumidity; // Вологість землі
    unsigned int m_nClockCounter; // Лічильник часу
    unsigned int m_nHours; // Години
    unsigned int m_nMinutes; // Хвилини
    unsigned int m_nWaterLevelCounter; // Кількість спрацьовувань датчика рівня,
постанова для
    unsigned int m_nWaterLevel; // Стан датчика рівня води
    unsigned int m_nHeatWater; // Нагрівання води
    unsigned int m_nLight; // Освітлення
    unsigned int m_nFan; // Вентилятор
    unsigned int m_nPump; // Насос

    unsigned int m_nTimeUpdate; // Оновлення часу 0 - Ні / 1 - Оновити час
    unsigned int m_nTypeWork; // Режим роботи 0 - Ручний / 1 - Автоматичний
    unsigned int m_nTypeDay; // Режим доби 0 - Ніч / 1 - День
    unsigned int m_nSetPointDayHours; // Уставка початку дня години
    unsigned int m_nSetPointDayMinutes; // Уставка початку дня хвилини
    unsigned int m_nSetPointDayAirHumidityMin; // Уставка дня вологість повітря
    unsigned int m_nSetPointDayAirHumidityMax; // Уставка дня вологість повітря
    unsigned int m_nSetPointDayAirTemperatureMin; // Уставка дня температура пові-
тря
    unsigned int m_nSetPointDayAirTemperatureMax; // Уставка дня температура пові-
тря
    unsigned int m_nSetPointDayAirTemperature; // Уставка дня температура повітря
    unsigned int m_nSetPointDayGroundHumidityMin; // Уставка дня вологість землі
    unsigned int m_nSetPointDayGroundHumidityMax; // Уставка дня вологість землі
    unsigned int m_nSetPointDayWaterTemperatureMin; // Уставка дня температура во-
ди
    unsigned int m_nSetPointDayWaterTemperatureMax; // Уставка дня температура во-
ди
    unsigned int m_nSetPointNightHours; // Уставка початку ночі години
    unsigned int m_nSetPointNightMinutes; // Уставка початку ночі хвилини
    unsigned int m_nSetPointNightAirHumidityMin; // Уставка ночі вологість повітря
    unsigned int m_nSetPointNightAirHumidityMax; // Уставка ночі вологість повітря
    unsigned int m_nSetPointNightAirTemperatureMin; // Уставка ночі температура
повітря
    unsigned int m_nSetPointNightAirTemperatureMax; // Уставка ночі температура
повітря
    unsigned int m_nSetPointNightAirTemperature; // Уставка ночі температура пові-
тря
    unsigned int m_nSaveSetPoint; // Збереження налаштувань 0 - Ні / 1 - Так

    unsigned int m_nLCDCounter; // Лічильник LCD
    unsigned int m_nLCDState; // Стан LCD
    unsigned int m_nLCDValueCounter; // Лічильник відображення
    unsigned int m_nLCDNeedUpdate; // Необхідність поновлення

    float m_fAirTemperature; // Температура повітря
};
```

A.5 ModbusTCPSlave.h

```
#include <ESP8266WiFi.h>

class ModbusTCPSlave
{
```

```
  public:
    ModbusTCPSlave(const unsigned int cnCoilsMax = 20, const unsigned int
cnInputRegistersMax = 20, const unsigned int cnHoldingRegistersMax = 20, const
unsigned int cnTCPPort = 502);

    void begin(const char *csSSID, const char *csPassword, const uint8_t nIP[4],
const uint8_t nGateway[4], const uint8_t nSubnet[4]);
    void poll();

    unsigned int  *m_Coils;
    unsigned int  *m_InputRegisters;
    unsigned int  *m_HoldingRegisters;

    const unsigned int m_cnCoilsMax;
    const unsigned int m_cnInputRegistersMax;
    const unsigned int m_cnHoldingRegistersMax;

  private:
    byte m_Buffer[270];
    WiFiServer m_Server;
    WiFiClient m_Client;
};
```

A.6 ModbusTCPSlave.cpp

```
#include "ModbusTCPSlave.h"

enum ModbusFunctions
{
  MB_FC_NONE = 0,
  MB_FC_READ_COILS = 1,
  MB_FC_READ_DISCRETE_INPUT = 2,
  MB_FC_READ_REGISTERS = 3,
  MB_FC_READ_INPUT_REGISTERS = 4,
  MB_FC_WRITE_COIL = 5,
  MB_FC_WRITE_REGISTER = 6,
  MB_FC_WRITE_MULTIPLE_COILS = 15,
  MB_FC_WRITE_MULTIPLE_REGISTERS = 16
};

enum ModbusOffsets
{
  MB_TCP_TID = 0,
  MB_TCP_PID = 2,
  MB_TCP_LEN = 4,
  MB_TCP_UID = 6,
  MB_TCP_FUNC = 7,
  MB_TCP_REGISTER_START = 8,
  MB_TCP_REGISTER_NUMBER = 10
};

ModbusTCPSlave::ModbusTCPSlave(const unsigned int cnCoilsMax, const unsigned int
cnInputRegistersMax, const unsigned int cnHoldingRegistersMax, const unsigned int
cnTCPPort) :
    m_cnCoilsMax(cnCoilsMax),
    m_cnInputRegistersMax(cnInputRegistersMax),
    m_cnHoldingRegistersMax(cnHoldingRegistersMax),
    m_Server(cnTCPPort)
{
  if (m_cnCoilsMax != 0)
    m_Coils = new unsigned int[m_cnCoilsMax];
  else
    m_Coils = 0;
```

```
  if (m_cnInputRegistersMax != 0)
    m_InputRegisters = new unsigned int[m_cnInputRegistersMax];
  else
    m_InputRegisters = 0;

  if (m_cnHoldingRegistersMax != 0)
    m_HoldingRegisters = new unsigned int[m_cnHoldingRegistersMax];
  else
    m_HoldingRegisters = 0;
}

void ModbusTCPSlave::begin(const char *csSSID, const char *csPassword, const
uint8_t nIP[4], const uint8_t nGateway[4], const uint8_t nSubnet[4])
{
    WiFi.mode(WIFI_AP_STA);
    WiFi.softAPConfig(IPAddress(nIP), IPAddress(nGateway), IPAddress(nSubnet));
    WiFi.softAP(csSSID, csPassword);

    m_Server.begin(); // Запуск сервера
}

void ModbusTCPSlave::poll()
{
    byte byteFN = MB_FC_NONE;
    int Start;
    int WordDataLength;
    int ByteDataLength;
    int MessageLength;

    if (m_Server.hasClient()) // Перевірка наявності клієнта
    {
      if (!m_Client || !m_Client.connected()) // Якщо клієнт вільний
      {
          if (m_Client)
            m_Client.stop();
          m_Client = m_Server.available();
      }
      else // Клієнт зайнятий, відмова в підключенні
      {
        WiFiClient serverClient = m_Server.available();
        serverClient.stop();
      }
    }

    if (m_Client && m_Client.connected() && m_Client.available()) // Читання даних
з буфера
    {
        unsigned int nIndex = 0;
        while(m_Client.available() && nIndex < sizeof(m_Buffer))
        {
            m_Buffer[nIndex] = m_Client.read();
            ++nIndex;
        }
    }
    else
      return;

    byteFN = m_Buffer[MB_TCP_FUNC];
    Start =
word(m_Buffer[MB_TCP_REGISTER_START],m_Buffer[MB_TCP_REGISTER_START+1]);
    WordDataLength =
word(m_Buffer[MB_TCP_REGISTER_NUMBER],m_Buffer[MB_TCP_REGISTER_NUMBER+1]);

    switch(byteFN)
```

```
    {
        case MB_FC_NONE:
            break;

        case MB_FC_READ_REGISTERS:  // 03 Read Holding Registers
            ByteDataLength = WordDataLength * 2;
            m_Buffer[5] = ByteDataLength + 3; //Number of bytes after this one.
            m_Buffer[8] = ByteDataLength;     //Number of bytes after this one (or
number of bytes of data).
            for(int i = 0; i < WordDataLength; i++)
            {
                m_Buffer[ 9 + i * 2] = highByte(m_HoldingRegisters[Start + i]);
                m_Buffer[10 + i * 2] =  lowByte(m_HoldingRegisters[Start + i]);
            }
            MessageLength = ByteDataLength + 9;
            m_Client.write((const uint8_t *)m_Buffer,MessageLength);

            byteFN = MB_FC_NONE;
            break;

        case MB_FC_READ_INPUT_REGISTERS:  // 04 Read Input Registers
            Start =
word(m_Buffer[MB_TCP_REGISTER_START],m_Buffer[MB_TCP_REGISTER_START+1]);
            WordDataLength =
word(m_Buffer[MB_TCP_REGISTER_NUMBER],m_Buffer[MB_TCP_REGISTER_NUMBER+1]);
            ByteDataLength = WordDataLength * 2;
            m_Buffer[5] = ByteDataLength + 3; //Number of bytes after this one.
            m_Buffer[8] = ByteDataLength;     //Number of bytes after this one (or
number of bytes of data).
            for(int i = 0; i < WordDataLength; i++)
            {
                m_Buffer[ 9 + i * 2] = highByte(m_InputRegisters[Start + i]);
                m_Buffer[10 + i * 2] =  lowByte(m_InputRegisters[Start + i]);
            }
            MessageLength = ByteDataLength + 9;
            m_Client.write((const uint8_t *)m_Buffer,MessageLength);
            byteFN = MB_FC_NONE;
            break;

        case MB_FC_WRITE_REGISTER:  // 06 Write Holding Register
            m_HoldingRegisters[Start] =
word(m_Buffer[MB_TCP_REGISTER_NUMBER],m_Buffer[MB_TCP_REGISTER_NUMBER+1]);
            m_Buffer[5] = 6; //Number of bytes after this one.
            MessageLength = 12;
            m_Client.write((const uint8_t *)m_Buffer,MessageLength);
            byteFN = MB_FC_NONE;
            break;

        case MB_FC_WRITE_MULTIPLE_REGISTERS:    //16 Write Holding Registers
            ByteDataLength = WordDataLength * 2;
            m_Buffer[5] = ByteDataLength + 3; //Number of bytes after this one.
            for(int i = 0; i < WordDataLength; i++)
            {
                m_HoldingRegisters[Start + i] =  word(m_Buffer[ 13 + i *
2],m_Buffer[14 + i * 2]);
            }
            MessageLength = 12;
            m_Client.write((const uint8_t *)m_Buffer,MessageLength);
            byteFN = MB_FC_NONE;
            break;
    }
}
```

ДОДАТОК Б – ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ЛЮДИНО-МАШИННОГО ІНТЕРФЕЙСУ

Б.1 Драйвери

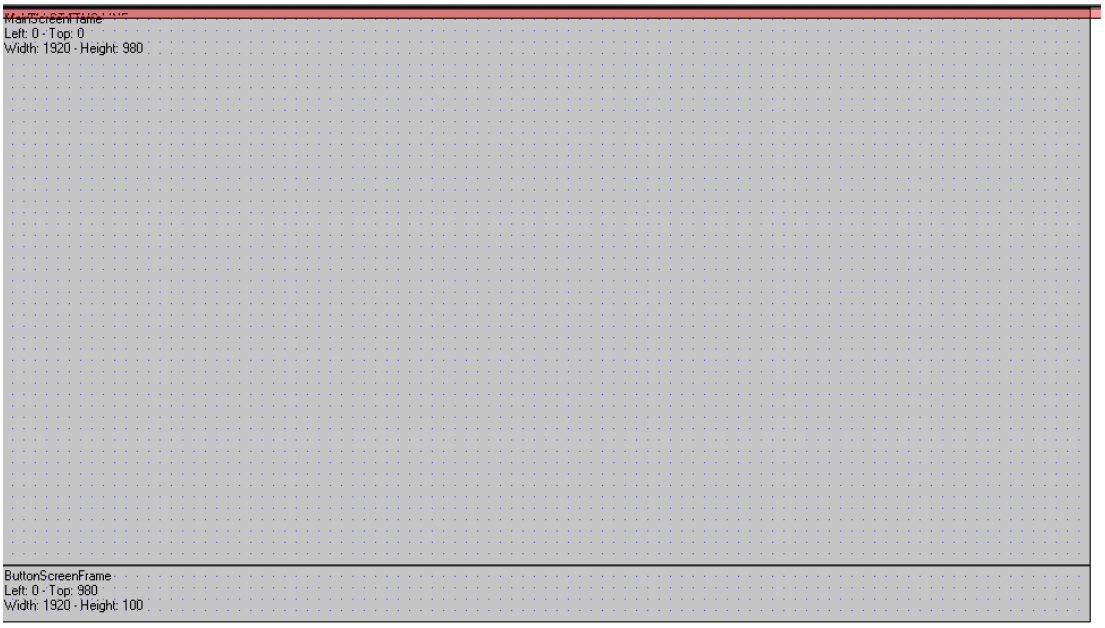
State	Identification	Description	File name
	Driver for internal variables		Intern
	Driver for mathematics variables		MATHDR32
	Driver for system variables		SYSDRV
	Controller		MODRTU32

Б.2 Перелік змінних

State	Name	Measur...	Driver
	AirTemperature	°C	MODRTU32 - Controller
	AirWet	%	MODRTU32 - Controller
	DayTime		MODRTU32 - Controller
	Fan		MODRTU32 - Controller
	GruntTemperature	°C	MODRTU32 - Controller
	GruntWet	%	MODRTU32 - Controller
	Illumination		MODRTU32 - Controller
	Pump		MODRTU32 - Controller
	TypeWork		MODRTU32 - Controller
	WaterHeater		MODRTU32 - Controller
	WaterLevel		MODRTU32 - Controller
	WaterTemperature	°C	MODRTU32 - Controller

Б.3 Шаблони зображень

State	Display...	Name	Background color	Freely defineable fram...	Left [pixels]	Top [pixels]	Right [pixels]	Bottom [pixels]
	☑	ALARM STATUS LINE	#FF0000	☑	0	0	2560	18
	☑	MainScreenFrame	#CCCCCC	☐	0	0	1920	980
	☑	ButtonScreenFrame	#CCCCCC	☐	0	980	1920	1080

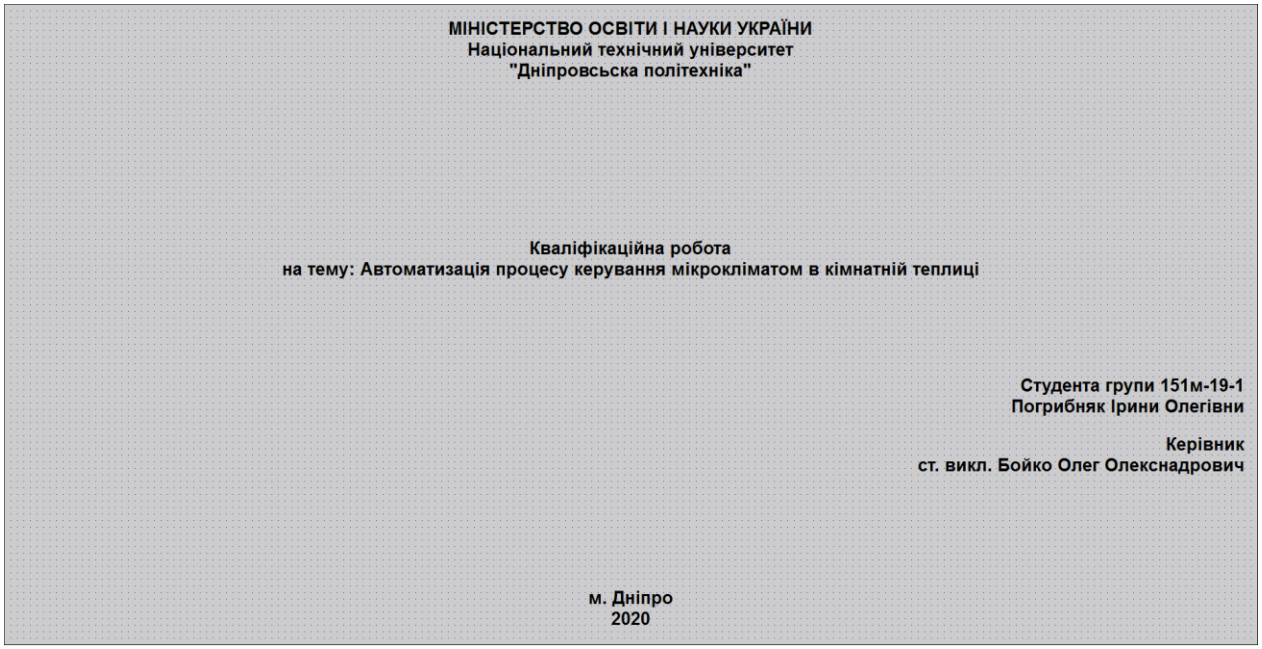


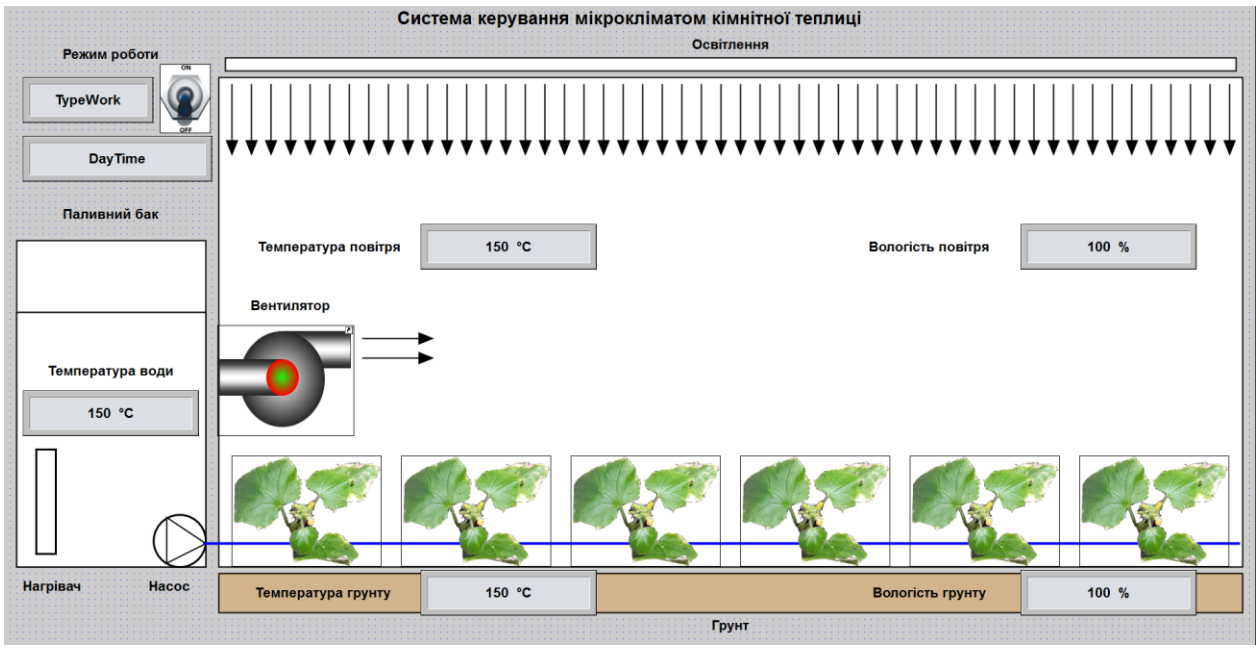
Б.4 Функції

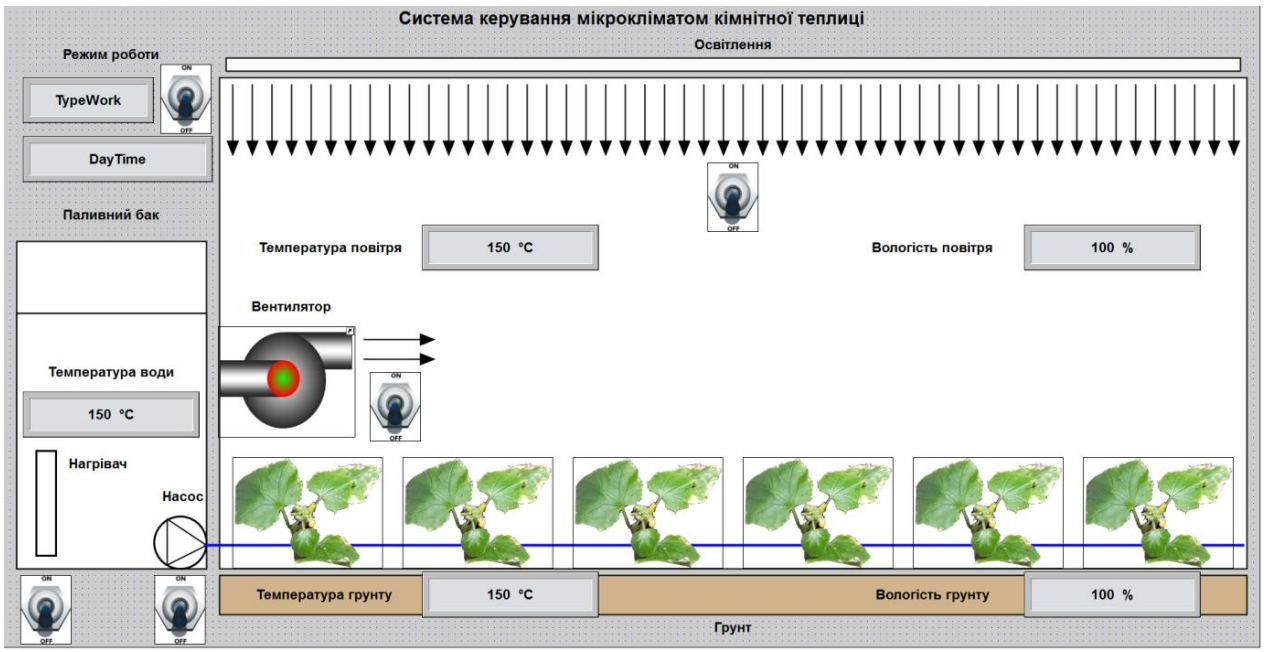
State	Name	Type	Parameter
	Filter text	Filter text	Filter text
	SwitchMainScreen	Screen switch	TechnologyScreen (Standard)
	SwitchButtonScreen	Screen switch	ButtonScreen (Standard)
	ReloadProject	Reload project online	changed objects
	ExitRuntime	Exit Runtime	
	SwitchHandleScreen	Screen switch	HandleScreen (Standard)
	SwitchTitleScreen	Screen switch	TitleScreen (Standard)

Б.5 Зображення

State	Name	Screen type	Frame	Background color	Start function	End function
	Filter text	Filter text	Filter text	Filter text	Filter text	Filter text
	HandleScreen	Standard	MainScreenFrame	#CCCCCC	< no function li...	< no function li...
	TechnologyScreen	Standard	MainScreenFrame	#CCCCCC	< no function li...	< no function li...
	ButtonScreen	Standard	ButtonScreenFrame	#CCCCCC	< no function li...	< no function li...
	TitleScreen	Standard	MainScreenFrame	#CCCCCC	SwitchButtonSc...	< no function li...




Система керування мікрокліматом кімнітної теплиці
Режим роботи
TypeWork
DayTime
Освітлення
Паливний бак
Температура повітря
150 °C
Вологість повітря
100 %
Вентилятор
Температура води
150 °C
Нагрівач
Насос
Температура грунту
150 °C
Вологість грунту
100 %
Грунт


Система керування мікрокліматом кімнітної теплиці
Режим роботи
TypeWork
DayTime
Освітлення
Паливний бак
Температура повітря
150 °C
Вологість повітря
100 %
Вентилятор
Температура води
150 °C
Нагрівач
Насос
Температура грунту
150 °C
Вологість грунту
100 %
Грунт

ДОДАТОК В – БІБЛІОТЕКА ЗВ'ЯЗКУ З СИСТЕМОЮ КЕРУВАННЯ

В.1 ModbusTCPFactory.java

```
public class ModbusTCPFactory
{
    // Формувач пакетів Modbus TCP

    ModbusTCPFactory()
    {
        setTID(1);
        setUID(0);
    }

    ModbusTCPFactory(int nTID, int nUID)
    {
        setTID(nTID);
        setUID(nUID);
    }

    ModbusTCPPacket ReadCoils(int nAddress, int nQuantity)
    {
        return ReadCoils(m_nTID, m_nUID, nAddress, nQuantity);
    }

    static ModbusTCPPacket ReadCoils(int nTID, int nUID, int nAddress, int
nQuantity)
    {
        byte[] data = new byte[]{(byte)(nAddress >>> 8), (byte)(nAddress),
(byte)(nQuantity >>> 8), (byte)(nQuantity)};
        return new ModbusTCPPacket((short)nTID, (byte)nUID, m_nFunctionReadCoils,
data);
    }

    ModbusTCPPacket ReadDiscreteInput(int nAddress, int nQuantity)
    {
        return ReadDiscreteInput(m_nTID, m_nUID, nAddress, nQuantity);
    }

    static ModbusTCPPacket ReadDiscreteInput(int nTID, int nUID, int nAddress, int
nQuantity)
    {
        byte[] data = new byte[]{(byte)(nAddress >>> 8), (byte)(nAddress),
(byte)(nQuantity >>> 8), (byte)(nQuantity)};
        return new ModbusTCPPacket((short)nTID, (byte)nUID,
m_nFunctionReadDiscreteInput, data);
    }

    ModbusTCPPacket ReadRegisters(int nAddress, int nQuantity)
    {
        return ReadRegisters(m_nTID, m_nUID, nAddress, nQuantity);
    }

    static ModbusTCPPacket ReadRegisters(int nTID, int nUID, int nAddress, int
nQuantity)
    {
        byte[] data = new byte[]{(byte)(nAddress >>> 8), (byte)(nAddress),
(byte)(nQuantity >>> 8), (byte)(nQuantity)};
        return new ModbusTCPPacket((short)nTID, (byte)nUID,
m_nFunctionReadRegisters, data);
    }

    ModbusTCPPacket ReadInputRegisters(int nAddress, int nQuantity)
```

```
    {
        return ReadInputRegisters(m_nTID, m_nUID, nAddress, nQuantity);
    }

    static ModbusTCPPacket ReadInputRegisters(int nTID, int nUID, int nAddress,
int nQuantity)
    {
        byte[] data = new byte[]{(byte)(nAddress >>> 8), (byte)(nAddress),
(byte)(nQuantity >>> 8), (byte)(nQuantity)};
        return new ModbusTCPPacket((short)nTID, (byte)nUID,
m_nFunctionReadInputRegisters, data);
    }

    ModbusTCPPacket WriteCoil(int nAddress, boolean bValue)
    {
        return WriteCoil(m_nTID, m_nUID, nAddress, bValue);
    }

    static ModbusTCPPacket WriteCoil(int nTID, int nUID, int nAddress, boolean
bValue)
    {
        byte[] data = new byte[]{(byte)(nAddress >>> 8), (byte)(nAddress),
(byte)(bValue ? 0xFF : 0x00), (byte)(0x00)};
        return new ModbusTCPPacket((short)nTID, (byte)nUID, m_nFunctionWriteCoil,
data);
    }

    ModbusTCPPacket WriteSingleRegister(int nAddress, int nValue)
    {
        return WriteSingleRegister(m_nTID, m_nUID, nAddress, nValue);
    }

    static ModbusTCPPacket WriteSingleRegister(int nTID, int nUID, int nAddress,
int nValue)
    {
        byte[] data = new byte[]{(byte)(nAddress >>> 8), (byte)(nAddress),
(byte)(nValue >>> 8), (byte)(nValue)};
        return new ModbusTCPPacket((short)nTID, (byte)nUID,
m_nFunctionWriteSingleRegister, data);
    }

    ModbusTCPPacket WriteMultipleCoils(int nAddress, boolean[] values)
    {
        return WriteMultipleCoils(m_nTID, m_nUID, nAddress, values);
    }

    static ModbusTCPPacket WriteMultipleCoils(int nTID, int nUID, int nAddress,
boolean[] values)
    {
        byte[] data = new byte[2 + 2 + 1 + values.length * 2];
        data[0] = (byte)(nAddress >>> 8);
        data[1] = (byte)(nAddress);
        data[2] = (byte)(values.length >>> 8);
        data[3] = (byte)(values.length);
        data[4] = (byte)(values.length * 2);
        int nIndex = 5;
        for (boolean value : values)
        {
            data[nIndex] = (byte)(value ? 0xFF : 0x00);
            data[nIndex + 1] = 0x00;
            nIndex += 2;
        }
        return new ModbusTCPPacket((short)nTID, (byte)nUID,
m_nFunctionWriteMultipleCoils, data);
```

```
    }

    ModbusTCPPacket WriteMultipleRegisters(int nAddress, int[] values)
    {
        return WriteMultipleRegisters(m_nTID, m_nUID, nAddress, values);
    }

    static ModbusTCPPacket WriteMultipleRegisters(int nTID, int nUID, int
nAddress, int[] values)
    {
        byte[] data = new byte[2 + 2 + 1 + values.length * 2];
        data[0] = (byte)(nAddress >>> 8);
        data[1] = (byte)(nAddress);
        data[2] = (byte)(values.length >>> 8);
        data[3] = (byte)(values.length);
        data[4] = (byte)(values.length * 2);
        return new ModbusTCPPacket((short)nTID, (byte)nUID,
m_nFunctionWriteMultipleRegisters, data);
    }

    void setTID(int nTID)
    {
        m_nTID = (short)nTID;
    }

    public int getTID()
    {
        return (int)m_nTID;
    }

    void setUID(int nUID)
    {
        m_nUID = (byte)nUID;
    }

    int getUID()
    {
        return (int)m_nUID;
    }

    private short m_nTID; // Ідентифікатор транзакції
    private byte m_nUID; // Ідентифікатор

    static private final byte m_nFunctionNone                       = 0;
// Невідома функція
    static private final byte m_nFunctionReadCoils                  = 1;
// Функція читання дискретних значень
    static private final byte m_nFunctionReadDiscreteInput          = 2;
// Функція читання дискретного входу
    static private final byte m_nFunctionReadRegisters              = 3;
// Функція читання регістрів
    static private final byte m_nFunctionReadInputRegisters         = 4;
// Функція читання вхідних регістрів
    static private final byte m_nFunctionWriteCoil                  = 5;
// Функція запису дискретного значення
    static private final byte m_nFunctionWriteSingleRegister        = 6;
// Функція запису дискретного регістра
    static private final byte m_nFunctionWriteMultipleCoils         = 15;
// Функція запису декількох дискретних значень
    static private final byte m_nFunctionWriteMultipleRegisters     = 16;
// Функція запису декількох регістрів
}
```

B.2 ModbusTCPPacket.java

```
class ModbusTCPPacket
{
    // Пакет Modbus TCP

    ModbusTCPPacket(short nTID, byte nUID, byte nFunction, byte[] data)
    {
        // Конструктор пакета на підставі значень полів

        m_nTID = nTID;
        m_nUID = nUID;
        m_nLength = (short)(1 + 1 + data.length);
        m_nFunction = nFunction;
        m_Data = data;
    }

    ModbusTCPPacket(char[] dataBuffer)
    {
        // Конструктор пакета на підставі отриманих даних

        if (dataBuffer.length < m_nLengthMin)
            return;

        m_nTID = (short)((((int)dataBuffer[m_nTIDAddress]) << 8) | (dataB-
?ffer[m_nTIDAddress + 1]));
        m_nUID = (byte)dataBuffer[m_nUIDAddress];
        m_nLength = (short)((((int)dataBuffer[m_nLengthAddress]) << 8) |
(dataBuffer[m_nLengthAddress + 1]));
        m_nFunction = (byte)dataBuffer[m_nFunctionAddress];

        m_Data = new byte[dataBuffer.length - m_nDataAddress];
        for (int nIndex = m_nDataAddress; nIndex < dataBuffer.length; ++nIndex)
            m_Data[nIndex - m_nDataAddress] = (byte)dataBuffer[nIndex];
    }

    byte[] getPacket()
    {
        // Отримання готового пакета

        byte[] result = new byte[m_nMBAP + 1 + m_Data.length];
        result[m_nTIDAddress]       = (byte)(m_nTID >>> 8);
        result[m_nTIDAddress + 1]   = (byte)(m_nTID);
        result[m_nPIDAddress]       = 0;
        result[m_nPIDAddress + 1]   = 0;
        result[m_nLengthAddress]    = (byte)(m_nLength >> 8);
        result[m_nLengthAddress + 1]= (byte)(m_nLength);
        result[m_nUIDAddress]       = m_nUID;
        result[m_nFunctionAddress]  = m_nFunction;
        System.arraycopy(m_Data, 0, result, m_nDataAddress, m_Data.length);

        return result;
    }

    String getPacketString()
    {
        // Отримання пакета у вигляді рядка

        StringBuilder sBuffer = new StringBuilder();
        byte[] packet = getPacket();

        for (byte nValue : packet)
        {
            sBuffer.append(Character.forDigit(nValue >>> 4, 16));
```

```
            sBuffer.append(Character.forDigit(nValue & 0x0F, 16));
        }

        return sBuffer.toString();
    }

    short m_nTID; // Ідентифікатор транзакції
    byte m_nUID; // Ідентифікатор блоку
    short m_nLength; // Довжина даних
    byte m_nFunction; // Функція
    byte[] m_Data; // Дані

    static private final byte m_nMBAP              = 7;
// Розмір заголовка
    static private final byte m_nLengthMin         = m_nMBAP + 1 + 2;
// Мінімальна довжина пакета
    static private final byte m_nTIDAddress        = 0;
// Адреса ідентифікатора транзакції
    static private final byte m_nPIDAddress        = 2;
// Адреса ідентифікатора протоколу
    static private final byte m_nLengthAddress     = 4;
// Адреса довжини даних рахованої від UID
    static private final byte m_nUIDAddress        = 6;
// Адреса ідентифікатор блоку
    static private final byte m_nFunctionAddress   = 7;
// Адреса функціонально
    static private final byte m_nDataAddress       = 8;
// Адреса початку даних
}
```