МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій
(повне найменування інституту, назва факультету)

Кафедра комп’ютерних та інформаційних технологій
(повне найменування інституту, назва факультету)

Дипломна магістерська робота

на тему Комп’ютерно-інтегрована система керування теплицею для вирощування рослин в безґрунтовому середовищі

Виконав: студент групи МгАк-21
спеціальності
151 - Автоматизація та комп’ютерно-інтегровані технології
за освітньою програмою
Комп’ютерно-інтегровані технологічні процеси і виробництва
Павло КУПРИЄНКО
Керівник к.ф-м.н.,доц. Юрій ПИЛИПЕНКО
Рецензент д.т.н., проф. Віктор ЧУПРИНКА

Київ - 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра інформаційних та комп'ютерних технологій

Спеціальність 151 - автоматизація та комп'ютерно-інтегровані технології

Освітня програма – комп'ютерно-інтегровані технологічні процеси та виробництва

ЗАТВЕРДЖУЮ

Завідувач кафедри ІКТ

_____ Владислава СКІДАН

"___" _____ 2022 р.

З А В Д А Н Н Я

НА ДИПЛОМНУ МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Куприєнко Павлу Дмитровичу

1. Тема роботи «Комп'ютерно-інтегрована система керування теплицею для вирощування рослин в безгрунтовому середовищі», науковий керівник роботи ПИЛИПЕНКО Юрій, к.фм.н., доц, затверджені наказом вищого навчального закладу від 29 вересня 2022 року, № 180.

2. Строк подання студентом роботи - 8 листопада 2022 р.

3. Вихідні дані до роботи: температура °С від 24 до 27; вологість від 60 до 98 %; від 50 до 80% рівень наповнення ємності з водою; кислотність від 5,5 до 6,5 pH. Всі значення вихідних параметрів потрапляють до системи автоматизації у вигляді електричних сигналів, напруга яких може змінюватися від 0 до 5В, в залежності від значення відповідного вихідного параметру.

4. Зміст дипломної роботи (перелік питань, які потрібно розробити): Вступ. Аналіз процесу тепличного вирощування сільськогосподарських культур. Апаратне забезпечення моделі, що проектується. Алгоритми систем керування у дипломній роботі. Побудова моделі та можливості її втілення на практиці. Загальні висновки.

5. Консультанти розділів роботи (проекту)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Вступ	Пилипенко Ю.М., доцент		
Розділ 1	Пилипенко Ю.М., доцент		
Розділ 2	Пилипенко Ю.М., доцент		
Розділ 3	Пилипенко Ю.М., доцент		
Розділ 4	Пилипенко Ю.М., доцент		
Висновки	Пилипенко Ю.М., доцент		

6. Дата видачі завдання 12 вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного роботи (проекту)	Строк виконання етапів роботи	Примітка
1	Вступ	17.09.2022	
2	Розділ 1. ТЕПЛИЧНЕ ВИРОЩУВАННЯ СІЛЬСКОГОСПОДАРСЬКИХ КУЛЬТУР	22.09.2022	
3	Розділ 2. АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ МОДЕЛІ, ЩО ПРОЕКТУЄТЬСЯ	30.09.2022	
4	Розділ 3. АЛГОРИТМИ СИСТЕМ КЕРУВАННЯ У ДИПЛОМНІЙ РОБОТІ	10.10.2022	
5	Розділ 4. ПОБУДОВА МОДЕЛІ ТА МОЖЛИВОСТІ ЇЇ ВТІЛЕННЯ НА ПРАКТИЦІ	20.10.2022	
6	Висновки	25.10.2022	
7	Оформлення магістерської роботи (чистовий варіант)	01.11.2022	
9	Здача дипломної магістерської роботи на кафедру для рецензування (за 14 днів до захисту)	08.11.2022	
10	Перевірка дипломної магістерської роботи на наявність ознак плагіату (за 10 днів до захисту)	11.11.2022	
11	Подання дипломної магістерської роботи на затвердження завідувачу кафедри (за 7 днів до захисту)	15.11.2022	

Студент _____ **Павло КУПРИЄНКО**
(підпис) **(прізвище та ініціали)**

Керівник проекту (роботи) _____ **Юрій ПИЛИПЕНКО**
(підпис) **(прізвище та ініціали)**

Директор НМЦПФ _____ **Олена ГРИГОРЕВСЬКА**
(підпис) **(прізвище та ініціали)**

АНОТАЦІЯ

Куприєнко П.Д. Комп'ютерно-інтегрована система керування теплицею для вирощування рослин в безґрунтовому середовищі

Дипломна магістерська робота за спеціальністю 151 – «Автоматизація та комп'ютерно-інтегровані технології», Київський національний університет технологій та дизайну, Київ, 2022 рік.

Дипломна магістерська робота присвячена розробці моделі системи автоматичного керування теплицею для вирощування рослин в безґрунтовому середовищі.

Запропоновано розглянути безґрунтову технологію вирощування рослин та визначити параметри, які впливають на швидкість росту та розвитку рослини при тепличному вирощуванні. На основі отриманої інформації розробити модель системи керування, яка буде забезпечувати слідкування та автоматичне регулювання значень вимірювальних параметрів.

Результатом проведеної роботи є розроблена модель системи автоматичного контролю керування теплицею з використанням безґрунтової технології вирощування рослин.

Ключові слова: система автоматичного керування, безґрунтове вирощування рослин, алгоритм, мікроконтролер, мікропроцесорна платформа Arduino, програмне забезпечення проекту.

АННОТАЦИЯ

Куприенко П.Д. Компьютерно-интегрированная система управления теплицей для выращивания растений в беспочвенной среде

Дипломная магистерская работа по специальности 151 – Автоматизация и компьютерно-интегрированные технологии, Киевский национальный университет технологий и дизайна, Киев, 2022 год.

Дипломная магистерская работа посвящена разработке модели системы автоматического управления теплицей для выращивания растений в беспочвенной среде.

Предложено рассмотреть беспочвенную технологию выращивания растений и определить параметры, влияющие на скорость роста и развития при тепличном выращивании. На основе полученной информации разработать модель системы управления, которая будет обеспечивать слежение и автоматическую регулировку значений измерительных параметров.

Результатом проведенной работы является разработаная модель системы автоматического контроля управления теплицей с использованием беспочвенной технологии выращивания растений.

Ключевые слова: система автоматического управления, беспочвенное выращивание растений, алгоритм, микроконтроллер, микропроцессорная платформа Arduino, программное обеспечение проекта.

ANNOTATION

Kupryienko P.D. Computer-integrated greenhouse control system for growing plants in a soilless environment

Master's thesis in specialty 151 - "Automation and computer-integrated technologies", Kyiv National University of Technology and Design, Kyiv, 2022.

The master's thesis is devoted to the development of a model of the automatic control system of a greenhouse for growing plants in a soilless environment.

It is proposed to consider the soilless technology of growing plants and to determine the parameters that affect the rate of growth and development of plants during greenhouse cultivation. On the basis of the received information, develop a control system model that will provide monitoring and automatic adjustment of the values of the measurement parameters.

The result of the work is a developed model of the automatic control system for greenhouse management using soilless plant growing technology.

Keywords: automatic control system, soilless plant cultivation, algorithm, microcontroller, Arduino microprocessor platform, project software.

ЗМІСТ

ВСТУП

Актуальність теми магістерської дипломної роботи. Протягом останніх років людство дедалі більше стикається з нетиповими кліматичними аномаліями, які призводять до зміни природного балансу планети. Територія України також не є виключенням, все частіше спостерігаються аномальні погодні явища, які раніше відбувалися раз 50 – 100 років. Значні коливання температур протягом коротких проміжків часу, паводків, ураганів, штормів, посух, тривалих ливнів в першу чергу впливають на врожай та врожайність сільськогосподарських культур.

Для забезпечення стабільного врожаю все більш стає актуальним питання про створення штучного середовища – теплиці, в якому забезпечується оптимальні умови для вирощування сільськогосподарських культур.

Вирощування рослин в повністю контрольованих умовах має, як переваги так і недоліки. Безсуперечно головною перевагою теплиці є можливість забезпечити оптимальні умови для вирощування будь якої сільськогосподарської культури протягом всього календарного року. Використання такого підходу дає змогу розраховувати на стабільний та прогнозований врожай в незалежності від погодних умов навколишнього середовища.

Недоліком використання теплиць є обмежена ефективна посівна площа, звісно при необмеженому фінансуванні можна побудувати теплицю протяжністю в декілька десятків гектар, але це не є вигідним розподілом фінансового ресурсу. На сьогоднішній день найефективнішим способом вирішення проблеми обмеженості площі теплиці є відмова від ґрунтового середовища на корить альтернативних, безґрунтових, методів вирощування рослин.

Через те, що безґрунтові способи вирощування не залежать від ґрунтового середовища, це дозволяє збільшити ефективну площу теплиці, в вертикальному напрямку. Тобто створювати багатоярусні системи розташування рослин, дозволяє економити простір теплиці та ефективніше

використовувати енергетичні ресурси для підтримання оптимальних температурних умов для росту та розвитку рослин.

Безґрунтові методи вирощування є ефективними, але дуже залежними від рівня технічного забезпечення, тому даний спосіб вирощування набуває все більшої популярності через те, що сучасний рівень технічного та програмного забезпечення дозволяє розроблювати повністю автоматичну систему керування теплицею, яка в реальному часі отримує та аналізує великий масив даних і на основі отриманої інформації миттєво реагує на відхилення від заданих значень.

В даному проекті використовується аеропонний спосіб вирощування рослин на основі якого розроблена модель системи автоматичного керування теплицею.

Мета досліджень. Метою роботи є розробка моделі системи автоматичного керування теплицею з аеропонним способом вирощування рослин з імітацією реального технологічного процесу на базі мікропроцесорної плати Arduino.

Завдання дослідження.

- проаналізувати існуючі системи автоматично керування теплицею;
- визначити головні параметри, що потребують контролю в теплиці аеропонним способом вирощування рослин;
- розробити алгоритми роботи системи керування теплицею з аеропонним способом вирощування рослин;
- розробити модель системи автоматичного керування теплицею з аеропонним способом вирощування рослин;
- створити імітаційну модель роботи системи керування теплицею на базі мікропроцесорної плати Arduino;

Об’єкт дослідження – автоматична система керування теплицею з аеропонним способом вирощування рослин.

Предмет дослідження – методи, моделі та апаратно-програмні засоби для побудови автоматичної системи керування теплицею з аеропонним способом вирощування рослин.

Методи дослідження. Для вирішення поставлених задач використовувались методи побудови автоматичних систем керування, методи імітаційного моделювання, математичного аналізу та також об’єктно-орієнтованого програмування, функціонального проектування та відлагодження роботи системи.

Наукова новизна одержаних результатів полягає в тому, що розроблено метод багатовекторного автоматизованого контролю та розроблена модель системи автоматичного керування теплиці з аеропонним способом вирощування рослин на базі мікропроцесорної плати Arduino.

Практичне значення одержаних результатів полягає в тому, що використання запропонованої системи автоматичного керування теплиці з аеропонним способом вирощування рослин дозволить ефективно використовувати площу теплиці, отримувати прогнозований та стабільний врожай напротязі всього календарного року, а також зменшення собівартості продукту через ефективне використання водного та енергетичного ресурсу.

Апробація результатів магістерської дипломної роботи. Основні положення та результати роботи були представлені та обговорені на VI Міжнародній науково-практичній конференції «Мехатронні системи: інновації та інжиніринг» - «MSIE-2021 (КНУТД, Київ, 24 листопада 2022 р.).

РОЗДІЛ 1. ТЕПЛИЧНЕ ВИРОЩУВАННЯ СІЛЬСЬКОГОСПОДАРСЬКИХ КУЛЬТУР.

1.1 Огляд та характеристика основних видів теплиць.

Протягом останніх років людство дедалі більше стикається з нетиповими кліматичними аномаліями, які призводять до зміни природного балансу планети. Територія України також не є виключенням, все частіше спостерігаються аномальні погодні явища, які раніше відбувалися раз на 50 – 100 років. Значні коливання температур протягом коротких проміжків часу, паводків, ураганів, штормів, посух, тривалих ливнів в першу чергу впливають на врожай та врожайність сільськогосподарських земель.

Теплиця – це штучно створене середовище для забезпечення оптимальних умов для вирощування рослин. Сучасні теплиці можуть розрізнятися за режимами експлуатації, формою та розмірами, конструкцією, технологію вирощування рослин [1].

Режим експлуатації теплиці, визначає період в який здійснюється робота теплиці. Експлуатація теплиці може здійснюватися, як обмежений період, та так не обмежений період часу. Теплиці, що мають обмежений період експлуатації який найчастіше починається в березня і закінчується пізньою осінню мають назву – сезонні. Такий вид теплиць найчастіше використовується в домашньому господарстві. В більшості сезонних теплицях практично не використовується засоби автоматизації це означає те, що рівень забезпечення оптимальних умов для вирощування рослин в повному об’ємі лягає на плечі людини, що відповідальна за кінцевий продукт.

На противагу сезонним теплицям, які використовуються протягом обмеженого періоду часу, цілорічні теплиці можуть використовуватись на протязі всього календарного року. Такий тип теплиць використовується для вирощування рослин в промисловому масштабі. Розмір споруджень може вварюватися залежності від масштабів планового виробництва. В теплицях, що використовуються на протязі року є необхідність забезпечувати та підтримувати оптимальні умови для росту та розвитку рослин за будь яких

зовнішніх погодних умов. Рівень автоматизації в таких теплицях значно вище в порівнянні з сезонними, що дозволяє будувати багатоконтурні системи для слідкування за кожним аспектом процесу вирощування рослини [2].

Форма теплиць визначає кількість природного світла, що досягає рослини та впливає на енергоефективність. Найчастіше для промислових цілей вибирають теплиці прямої форми (рис 1.1), оскільки такий тип має просту та зрозумілу конструкцію [3].

а) б)

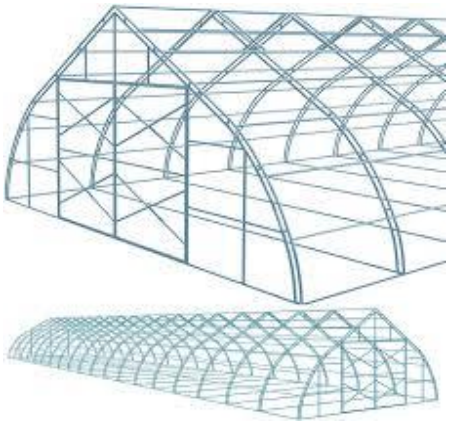
Рис 1.1 – а) Приклад теплиці прямої форми;
б) Приклад реалізації теплиці прямої форми в реальний практиці

Арочні форма також не є рідкістю, адже мають відмінну стійкість бічним вітрам, витримують велику вагу осадків (рис 1.2). Та меншу площу, яку необхідно покрити матеріалу для покриття, не зменшуючи площу споруди.

а) б)

Рис 1.2 – а) Приклад теплиці арочної форми;
б) Приклад реалізації теплиці арочної форми в реальний практиці

Стрілчаста форма відрізняється від арочної загостреним дахом, яка не дозволяє опадам накопичуватися і не перешкоджає проникненню сонячного світла (рис 1.3).

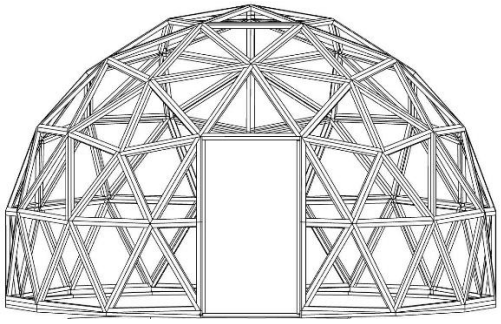


а)

б)

Рис 1.3 – а) Приклад теплиці стрілчастої форми;
б) Приклад реалізації теплиці стрілчастої форми в реальний практиці

Теплиці з формою купола забезпечують напевно найкращу проникність природнього світлового потоку, та забезпечують великий простір, який можна організувати багатьма способами (рис 1.4). Недоліком такої форми є велика вартість та швидкість зведення такої конструкції в порівнянні з іншими.



а)

б)

Рис 1.4 – а) Приклад теплиці купольної форми;
б) Приклад реалізації теплиці купольної форми в реальний практиці

У конструкції теплиці визначальну роль відграє покриття. Найчастіше для теплиць використовують два види покриття: скло та стільниковий полікарбонат.

Теплиці зі скла відрізняються хорошою світлопроникністю, яка позитивно впливає на рослини та економить значну частину коштів дозволяючи не вмикати штучне освітлення протягом світлового дня. Перевагою скла перед іншими матеріалами полягає в хороших теплоізоляційних здібностях даного матеріалу і його порівняно невеликій ціні. Однак скло має низку істотних недоліків. По перше це важка вага, яку витримає не кожен каркас. Усередині будівель зі скла повітря нагрівається дуже швидко, що є перевагою лише взимку, в будь-яку іншу пору року така властивість скла може бути недоліком, якщо використовувати недостатню потужну систему вентиляції оскільки наслідком перегріву може бути втрата врожаю частково або цілком [4].

Стільниковий полікарбонат (рис 1.5) є найбільш популярним матеріалом, адже забезпечує теплоізоляцію, розсіює шкідливі для рослин ультрафіолетові промені та є довговічним. Даний матеріал набув великої популярності через свою невелику вагу, внаслідок чого швидкість монтажних робіт в рази швидша та дешевша, ніж зі склом. Недоліком є менша світлопропускна здатність матеріалу в 85 – 90%, що трохи нижче, ніж у скла.

Рис 1.5 – Приклад стільникового полікарбонатного матеріалу

За технологією вирощування теплиці можна поділити на декілька видів:

– ґрунтові;

– стелажні;

– безґрунтові.

У теплицях, що використовують ґрунтові спосіб вирощування використовуються, природне ґрунтове середовище, або ґрунтові суміші з урахуванням особливостей і потреб різних видів рослин.

Рис 1.6 – Теплиця з ґрунтовим методом вирощування рослин

У стелажному варіанті культури вирощуються на поличках, оснащених бортами (рис 1.7). Даний варіант облаштування внутрішнього простору дозволяє більш раціонально використовувати робочу площу, оскільки рослини розташовуються в декілька поверхів. Єдиним обмеженням є неможливість вирощувати в них високорослі рослини [5].

Рис 1.7 - Теплиця з стелажним методом вирощування рослин

Теплиці, що використовують безґрунтові методи вирощування рослин, можна поділи на два типи в залежності від технології вирощування: гідропонний, або аеропонний спосіб [6]. Використовуючи гідропонну тепличну технологію, рослини культивують у водних розчинах з поживних сумішей, якими просочують спеціальний підтримуючий субстрат, який виконує функції ґрунту (рис 1.8). [7]

Рис 1.8 – Приклад гідропонного вирощування рослин

Аеропонний метод не передбачає занурення кореневої системи рослини в субстрат, навпаки корені рослин вільно звисають в закритому середовищі в яке у вигляді туману, що окутує кореневу систему рослин, поступають поживні речовини. Створюючи тим самим середовище багате киснем, що сприяє ефективному розвитку рослини [8].

Рис 1.9 – Приклад вертикальної аеропонної ферми

У зв`язку з тим, що в теплицях рослини не відчувають ніяких впливів ззовні крім світла, є необхідність забезпечення всіх умов за допомогою спеціального теплично обладнання. Для підтримання температурного режиму використовують обладнання для опалення, але після досягнення в приміщенні температури 40 °С рослини стають млявими і незабаром гинуть, тому є необхідність встановити систему провітрювання для запобігання такої проблеми. Сучасні системи вентиляції дозволяють поєднати систему опалювання з системою провітрювання, тим самим ефективно підтримувати заданий температурний режим, незважаючи на зовнішні погодні фактори [9].

Правильний полив є одним з факторів гарного врожаю, при використанні ґрунтового способу вирощування рослин необхідно забезпечити систему поливу рослин, яка повинна включати автоматичний крапельний або внутрішньогрунтовий полив теплиці.

Устаткування теплиць має також включати систему штучного освітлення, без якого неможлива правильна життєдіяльність рослин. Як правило, штучне світло встановлюється над молодими рослинами і використовується ввечері, вранці і в похмуру погоду. Сучасні лампи освітлення для теплиць це світлодіодні лампи з двома світловими спектрами, які найбільше впливають на фотосинтез рослини, саме червоний та синій (рис 1.10).

Рис 1.10 – Система штучного освітлення в теплиці

Скорочення тривалості світлового дня восени погано впливає на рослини, тому в цей період штучне світло використовується більш активно. Освітлення в промислових теплицях включають, коли рослинам не вистачає природного світла, тим самим збільшуючи світловий день [10].

1.2 Аналіз безґрунтових технологій вирощування рослин

Технологія безґрунтового вирощування рослин не є чимось новим для світу. На протязі всієї історії починаючи з древнього Єгипту, де єгипетські ієрогліфи описують вирощування рослин таких як, папірус чи лотос у водному середовищі, до наших днів де за допомогою аеропоніки проводять експерименти по вирощуванню рослин на космічних станціях.

За останні декілька десятків років увага до безґрунтових способів вирощування рослин зросла через декілька причин, основною з яких є потреба в забезпеченні великого попиту на високоякісну овочеву продукцію через постійне зростання населення планети. Високий рівень науки та техніки, дозволяють забезпечити найоптимальніші витрати водного та енергетичного ресурсів для отримання максимального врожаю.

Традиційні сільськогосподарські системи використовують велику кількість прісної води та добрив з відносно незначною прибутковістю. Гідропоніка та аеропоніка – це сучасні системи вирощування сільськогосподарських культур, які використовують багаті поживними речовинами середовища, а не ґрунт для живлення рослин. Такі системи є ефективними через те, що потребують менше водного ресурсу та місця в просторі в порівнянні зі звичайними сільськогосподарськими системами.

Різниця між аеропонікою та гідропонікою полягає в тому, як коріння піддаються впливу живильного розчину. У гідропоніці коріння занурюють у поживний розчин, тоді як в аеропоніці коріння підвішені в повітрі і періодично зрошуються поживним розчином. Зрошення відбувається за допомогою форсунок, що розпилюють розчин перетворюючі потік води в туман, який окутує корені рослин [11].

Часто в телицях, що використовують аеропонний спосіб вирощування рослин, практикувалося використання вертикальних колон в яких знаходяться рослини (рис 1.11) і розпилення живильного розчину відбувається в верхній частини колони, та каскадом спускається в низу покриваючи всі корені поживним розчином. Розчин, що не був поглинутий рослиною накопичується

в нижній частині колони та конденсує, тобто переходить в рідкий стан. В нижній частині як правило встановлена дренажна система, яка дозволяє відводити зайву вологу. Розчин, що був зібраний дренажною системою повторно задіються для зрошення рослин.

Рис 1.11 – Приклад вирощування рослин аеропонним способом

Гідропоніка у порівнянні зі звичайним методом вирощування використовує лише 10% водних ресурсів. Системи аеропоніки можуть зменшити використання води на 98%, використання добрив на 60%, і використання пестицидів на 100%, і все це при максимальному збільшенні врожайності. При використанні системи аеропоніки, рослини, які вирощуються, поглинають більше мінералів і вітамінів, роблячи рослини здоровішими на зовнішній вигляд. Ще одна перевага полягає в тому, що рослини легше пересаджувати, оскільки вони не страждають від трансплантаційного шоку [12].

При використанні аеропонної технології вирощування в воду, що поглинають рослини додається відносно мала частка добрив в порівнянні не тільки з ґрунтовим способом вирощування, але і з гідропонікою. Економія у використанні поживних речовин, відбувається оскільки в такій системі відбуваються мінімальні втрати водного ресурсу. Крім того, коріння поглинає кисень, і в рослинах, які вирощуються в ґрунті, кисень може бути обмежений.

Системи гідропоніки та аеропоніки забезпечують гнучкість і контроль за кожним етапом вирощування рослин, але обидві системи потребують значного технічного забезпечення, яке ефективно реалізується тільки в тепличному середовищі. Система гідропоніки може бути дещо простішою в реалізації та дозволяє дещо мати більші межі похибки, в порівнянні з аеропонікою. Однак і те, і інша системи набагато ефективніші, ніж ґрунтове землеробство, і обидві вони є майже однакові можливості щодо гнучкості контролю зрошення та поживних речовин програми [13].

Аеропоніка також дозволяє спостерігати за рослинами, не турбуючи їх. Швидкий доступ та регулювання на складу та кислотності живильного розчину, дозволяє усунути будь-які проблеми що хтось можуть мати рослини, перш ніж проблема справді стане непоправною.

Гідропонна система не має такої переваги тому, що живильний розчин є однією водяною масою в якій знаходяться рослини і хвороба, що передається водяним шляхом, може швидко поширюватися між ними.

До недоліків обох систем можна віднести залежність від електрики та при проектуванні таких систем вимагають наявності резервного генератора, щоб компенсувати перебої в електропостачанні.

Система аеропоніки має перевагу перед гідропонікою, тому що підвищена аерація розчину доставляє більше кисню до коренів рослин, стимулюючи ріст і запобігаючи утворенню патогенів [14].

Для розроблюваного проекту в якому створюється модель автоматичної системи керування теплицею обраний саме аеропонний спосіб вирощування рослин. Даний спосіб дозволяє краще використовувати простір теплиці. Він

гнучкий, що дозволяє використовувати його для вирощування багатьох сільськогосподарських культур без витрачання великих фінансових ресурсів на переобладнання. Саме даний спосіб найбільше підходить для створення вертикальних ферм завдяки чому коріння контактують саме з тими поживними речовинами, які їм потрібні, а також забезпечується середовище багате киснем, який рослини можуть поглинути. Це призводить до того, що рослини ростуть набагато швидше, що дозволить розраховувати на отримання декількох зборів врожаю за календарний рік.

1.3 Визначення та аналіз параметрів що регулюються автоматичною системою керування теплицею

Дотримання оптимального та стабільного температурного режиму в теплиці є однією з найголовніших запорук вдалого врожаю. Кожна плодоносна сільськогосподарська культура має свої оптимальні значення температури при дотриманні яких досягається найпродуктивніший рівень росту та розвитку рослини. Температура в теплиці повинна триматись в заданих значеннях в незалежності від перепадів нічних та денних температур, а також не залежати від різких погодних змін.

Технологія вирощування аеропонним способом для стабільної роботи системи потребує запасу розчину, що використовується для кореневої системи рослин. Для цього використовується ємність в якій може зберігатись рідини на декілька циклів поливу, в залежності від об’єму ємності, кількість запасу рідини може бути різним. Недостатній рівень заповнюваності ємності може стати проблемою у разі екстрених ситуацій, що можуть призвести до значних втрат продукції. Тому рівень наповненості є ще одним параметром, що контролюється автоматичною системою.

Рівень кислотності живильного розчину, яким здійснюються зрошення кореневої системи рослин також є дуже важливим. Саме цей показник впливає на швидкість та якість поглинання корисних речовин рослиною. Недотримання заданого значення рівня кислотності може призвести до

підвищення концентрації окремих елементів, і розчин для зрошення може стати токсичним для кореневої системи.

Ще одним параметром, що контролюється системою є рівень вологості в середовищі в якому знаходяться кореневі системи рослин. Інформація про актуальний рівень вологості дозволяє ефективно використовувати водний та енергетичний ресурс, через включення системи зрошення тільки при опусканні рівня вологості до нижньої точки при якому весь туман з попереднього циклу поливу конденсував в рідкий стан. Відключення системи зрошення відбувається при досягненні верхньої межі рівня вологості.

Висновки до розділу 1

1. Проведено аналіз теплиць, як системи для вирощування рослин розглянуті існуючі режими експлуатації теплиць протягом окремого сезонного періоду та на протязі всього календарного року. Проаналізовано існуючі форми тепличних споруд їх переваги та недоліки не тільки між собою, але і як форма теплиці впливає на організацію простору всередині. Розглянуті види зовнішнього покриття теплиці. Розглянуто, які параметри повинне забезпечувати тепличне обладнання

2. Були проаналізовані методи безґрунтового вирощування рослин. Виявленні їх переваги та недоліки перед ґрунтовим способом вирощування росли. На основі отриманої інформації був обраний аеропонний тип безґрунтового вирощування росли для розроблюваної автоматичної системи керування теплицею.

3. Визначенні основні параметри, які контролюються системою автоматичного керування теплицею з аеропонним способом вирощування рослин

РОЗДІЛ 2. АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ МОДЕЛІ, ЩО ПРОЕКТУЄТЬСЯ

2.1 Основні можливості та характеристики платформи Arduino

Arduino – це апаратна та програмна платформа з відкритим вихідним кодом, яка дуже проста у використанні. Ця платформа захопила уяву ентузіастів електроніки та спільноти розробників по всьому світу та дала змогу за доступну ціну експериментувати з електронними прототипами та втілювати їхні проекти у життя. Ці проекти можуть змінюватись від простого миготіння світлодіода до складних систем керування роботами [15].

Існує велика кількість видів плат Arduino, які можна розділити на дві групи: контролери та плати розширення. Контролери – це найважливіша частина плати, в яку записується виконувана програма (рис 2.1а). Плати розширення, є додатком до основної плати та містять ту чи іншу периферію, керовану контролером (рис 2.1б).

а) б)

Рис 2.1 – а) Плата Arduino Uno;
б) Плата розширення Arduino Wi-Fi

Серед широкого вибору плат з контролерами для даного проекту була обрана одна з найпопулярніших моделей Arduino Uno R3 (рис 2.2). Данна модель зроблена на основі мікроконтролера ATmega328. До його складу входить все необхідне для зручної роботи з мікроконтролером: 14 цифрових входів/виходів (з них 6 можуть використовуватися як ШІМ-виходи), 6 аналогових входів, кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм

живлення, роз'єм для внутрішньо-схемного програмування (ICSP) та кнопка скидання. В таблиці 2.1 наведенні технічні характеристики плати [16].

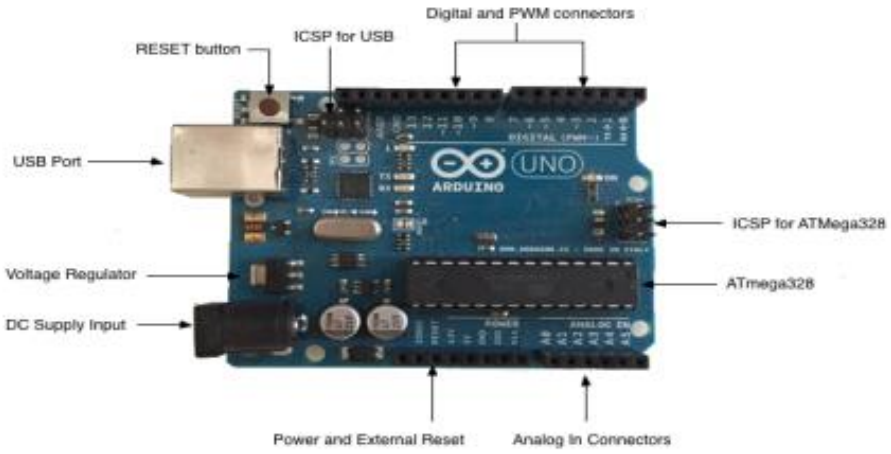


Рис 2.2 – Спрощена принципова схема плати Arduino Uno R3

Таблиця 2.1

Технічні характеристики Arduino Uno R3

Мікропроцесор	ATmega328
Робоча напруга	5 В
Вхідна напруга (рекомендована)	7 – 12 В
Вхідна напруга (максимальна)	6 – 20 В
Цифрові Входи/Виходи	14
Аналогові входи	6
Максимальний струм одного виходу	40 мА
Flash-пам'ять	32 КБ (ATmega328), 0.5 КБ зарезервовано завантажувачем
SRAM	2 КБ (ATmega328)
EEPROM	1 КБ (ATmega328)
Тактова частота	16 МГц

Живлення плати може відбуватись декількома способами: через контакти Vin/GND, через роз’єм зовнішнього живлення або USB-порт.

Виходи плати Vin і GND у роз'ємі живлення та зовнішнього скидання можуть використовуватися для живлення від зовнішньої батареї. Також для живлення може використовуватись мережевий AC/DC – адаптер із штекером діаметром 2.1 мм. Напруга зовнішнього джерела живлення може бути в межах від 6 до 20 В. Однак зменшення напруги живлення нижче 7В призводить до зменшення напруги на виведенні 5V, що може стати причиною нестабільної роботи пристрою. Використання напруги більше 12В може призводити до перегріву стабілізатора напруги та виходу плати з ладу. З огляду на це рекомендується використовувати джерело живлення з напругою в діапазоні від 7 до 12В.

Найпростішим та найменш затратним способом живлення плати є через USB – роз’єм, який може напряму підключатись до порту комп’ютера або через переносний акумулятор з USB виходом [17].

Більшість виходів плати можна налаштувати для виконання різних функцій. На рис 2.3 показана схема для чого можна використовувати різні виходи плати.

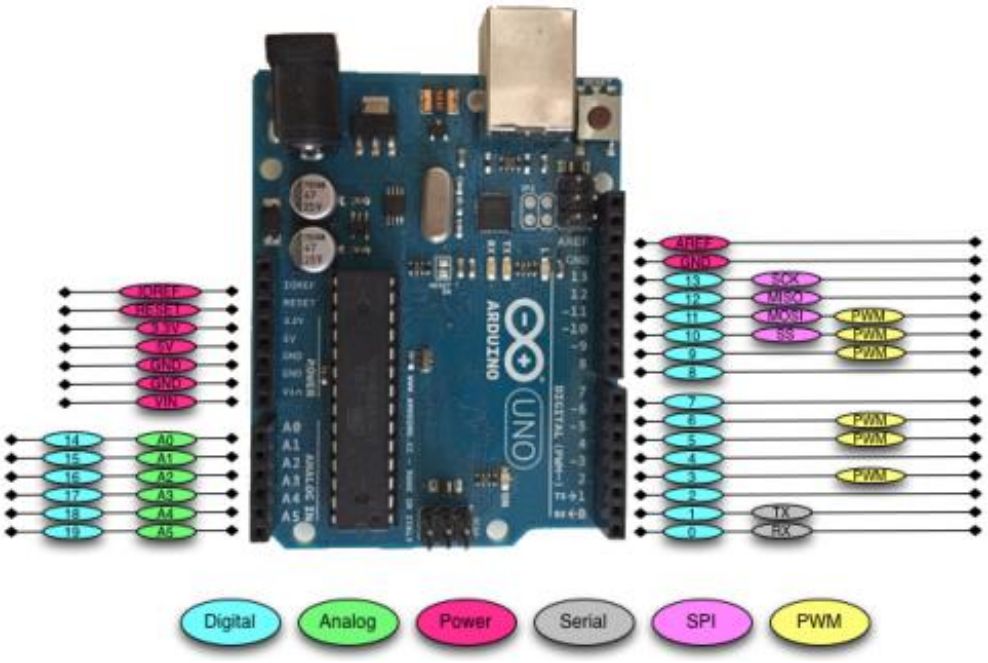


Рис 2.3 – Схема виходів плати Arduino Uno R3

Цифрові виходи Arduino найчастіше використовуються при підключенні зовнішніх датчиків. Ці виходи можуть бути налаштовані як для введення, так і для виводу інформації. Замовчуванням данні виходи перебувають у стані введення інформації. Цифрові виходи сприймають тільки одне з двох значень: HIGH (1), що дорівнює 5V, або LOW (0), яке одно 0V.

Плата може сприймати аналогові значення завдяки вбудованому аналогово-цифровому перетворювачу, що перетворює аналоговий сигнал в цифровий. Якщо аналогового виходи призначені для зчитування аналогових датчиків (вхід), виходи ШІМ призначені для виведення аналогової інформації через цифрові виходи.

Всі компоненти можуть напряму підключатись в плату, але це може призвести до поломки роз’єму або схема швидко перетвориться на безлад з купою проводів та елементів. Для уникнення подібного сценарію та для зручності роботи з Arduino використовується безпайкові макетні плати (рис 2.4).[18]

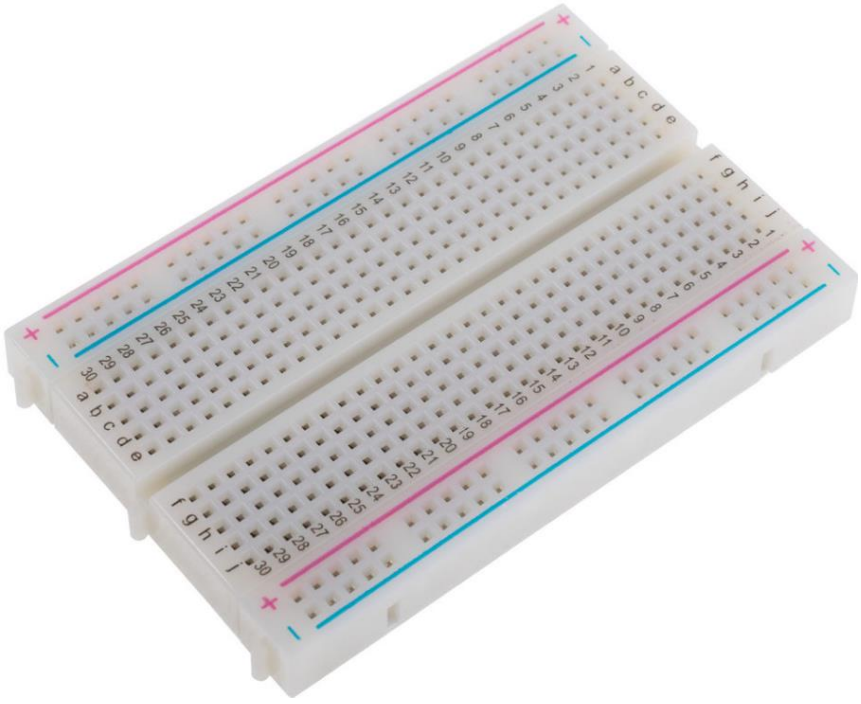


Рис 2.4 – Приклад безпайкової макетної плати

Щоб запрограмувати Arduino використовується програмне середовище Arduino IDE (рис 2.5), що є частиною безкоштовного програмного забезпечення, яке дозволяє програмувати мовою, яку розуміє Arduino. Мова програмування заснована на C/C++ і навіть може бути розширена за допомогою бібліотек C++ дозволяє створювати програми, що є набором інструкцій, які потім завантажуються в Arduino [19].

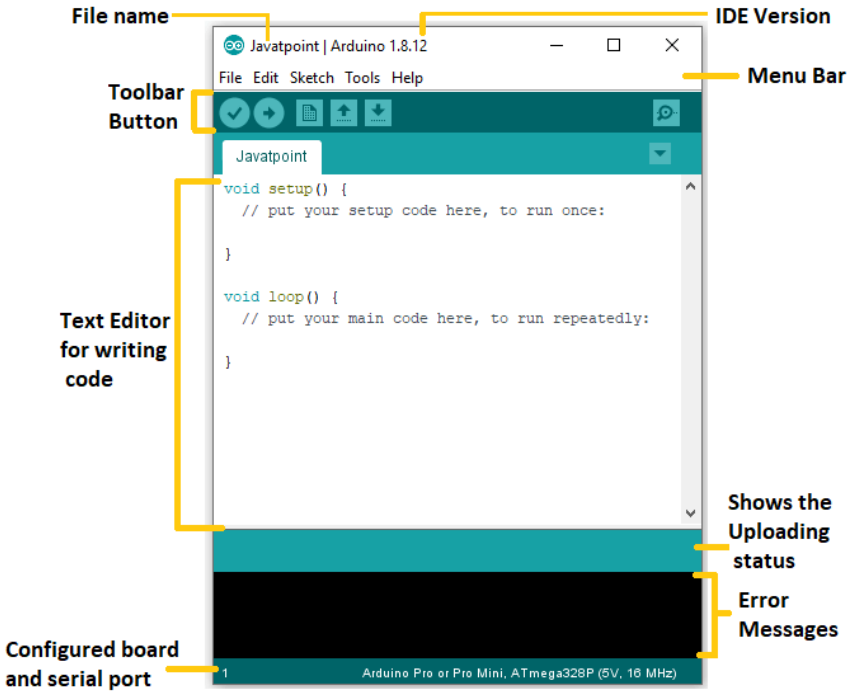


Рис 2.5 – Програмне середовище Arduino IDE

Програмне середовище складається з декількох частин, в верхній частині розташовані меню файлів, що мають стандартний набір параметрів для роботи з файлами. Наступним йде панель інструментів, які забезпечують швидкий доступ до функцій, що найчастіше використовуються для роботи з файлом.

Основною та найбільшою частиною програмного середовища є текстовий редактор для написання коду програми. Поля, що розташовані під текстовим полем показують статус успішності або неуспішності проведення компіляції при зберіганні написаного коду, а також показується повідомлення про помилку та інформацію про плату, яка підключена до комп’ютера.

2.2 Опис та аналіз датчика температури

Одним із завдань системи автоматичного керування є забезпечення заданого рівня температурного значення в середині теплиці. Вимірювання поточного значення температури відбувається за допомогою датчика температури. У якості чутливих елементів датчиків температури найчастіше застосовуються, тепломеханічні елементи, термометр опору та термопара.

Тепломеханічні датчики (термобіметалевий, дилатометричні) використовуються як датчики, що перетворюють зміну фактичного значення регульованої температури в переміщення.

Біметалевий елемент (рис 2.6) являє собою дві вузькі металеві пластинки 1 і 2 з різними коефіцієнтами температурного розширення, жорстко скріплених між собою по всій площині торкання (спаяні). Дія біметалевого датчика ґрунтується на відмінності температурних коефіцієнтів розширення різних металів. Коефіцієнт температурного розширення пластини 1 більше, ніж пластини 2 в 10-20 разів. При нагріванні біметалева смуга прогинається і контакт замикається. Датчики застосовуються для фіксації граничних значень температури та використовуються для двопозиційного регулювання «ввімкнуто-вимкнуто» температури [20].

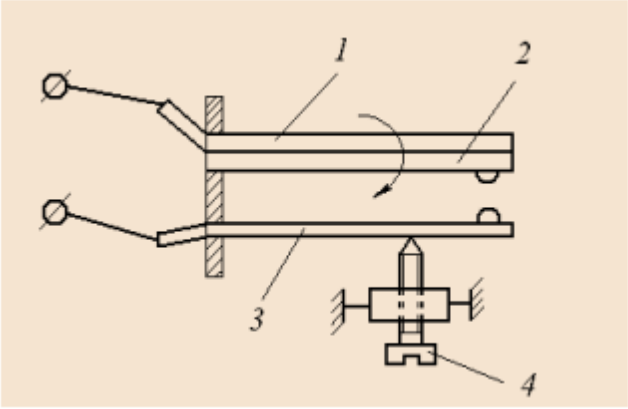


Рис 2.6 – Схема біметалевого датчика температури:
1 – пластина з заліза, міді або латуні; 2 – сплав заліза з нікелем;
3 – нерухомий контакт; 4 – регулюючий гвинт

Термометр опору – це пристрій, опір якого залежить від його температури. Такі термометри широко використовуються для вимірювання температур в діапазонах від -260 до 750 °С. В залежності від типу металу, що використовується при збільшенні температури опір може, як збільшуватись так і зменшуватись. Більшість чистих металів нагріванні на 1 °С збільшує свій опір в середньому на 0,4 – 0,6%, а напівпровідники при нагріванні, навпаки, можуть зменшувати свій опір. Причому зміна опору напівпровідників від температури відбувається в 5-10 більше, ніж у чистих металів. За рахунок цієї властивості напівпровідникових матеріалів такі термометри отримали назву “терморезистори” та широко використовуються в електроніці [21].

Термометр опору має наступну будову (рис 2.7) тонкий дріт намотують на каркас, який поміщають у металічний кожух таким чином, щоб запобігти електричного контакту вихідних дротів та самої спіралі з корпусом. Для цього корпус засипають порошком з діелектричного матеріалу, щоб запобігти електричного контакту дротів та самої спіралі з корпусом. Дріт намотується біфілярно, тобто дві близько розташовані, паралельні обмотки.

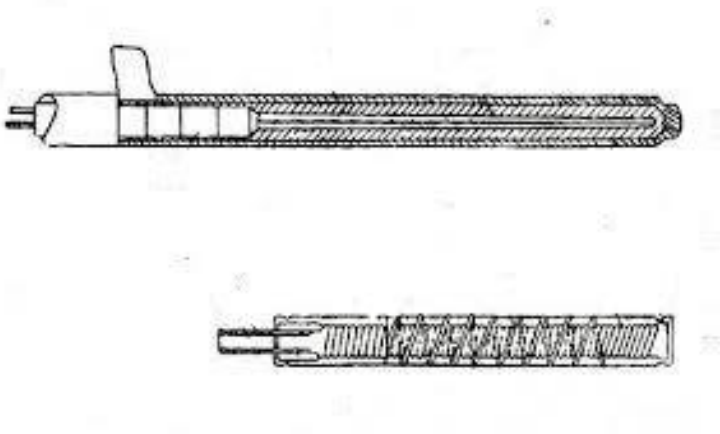


Рис 2.7 – Схема термометра опору

Термометри опору мають декілька схем підключення в залежності від необхідної точності та допустимої похибки обирається двопровідна, трипровідна або чотирипровідна схеми підключень. Найбільш розповсюджена є трипровідна схема, яка дозволяє отримувати похибку в 0,5°С на 100 °С.

Термопара – це пара провідників з різних матеріалів, з'єднаних на одному кінці і формують частину пристрою, що використовує термоелектричний ефект для вимірювання температури (рис 2.8).

Принцип дії заснований на ефекті Зеєбека або, інакше, термоелектричному ефекті. Коли кінці провідника знаходяться при різних температурах, між ними виникає різниця потенціалів, пропорційна різниці температур. Коефіцієнт пропорційності називають коефіцієнтом термо-ЕРС.

У різних металів коефіцієнт термо-ЕРС різний і, відповідно, різниця потенціалів, що виникає між кінцями різних провідників, буде різна. Поміщаючи спай з металів з відмінними коефіцієнтами в середовище з температурою Т1, то отримаємо напругу між протилежними контактами, які перебувають при іншій температурі Т2, яка буде пропорційна різниці температур Т1 і Т2 [22].

Рис 2.8 – Приклад термопари

Використовуються термопари для вимірювання температури різних типів об'єктів і середовищ, а також в автоматизованих системах управління і контролю. До переваг можна віднести широкий діапазон вимірювання, просту та дешевизну в виробництві та велику надійність.

При впровадженні розроблюваного контуру для промислової теплиці, на основі проведеного аналізу переваг та недоліків доступних датчиків вимірювання температури, я би відав перевагу, саме термометру опору. Адже такий тип датчика повністю забезпечує необхідну точність та надійність при вимірюванні температури навколишнього середовища.

Для побудови моделі контуру вимірювання температури за допомогою плати Arduino було обрано датчик температури DS18B20 (рис 2.9).

Даний датчик забезпечує вимірювання температури в діапазоні від -10 до +85 °C з похибкою в 0.5 °C, що оптимально підходить для перевірки роботи контору що забезпечує підтримання заданого температурного режиму.

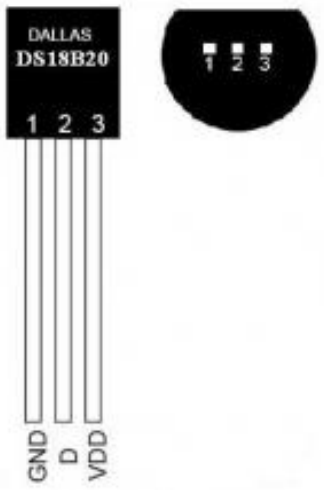


Рис 2.9 – Датчик температури DS18B20

2.3 Опис та аналіз датчика вологості

Вирощування рослин методом аеропоніки є дуже економним процесом, через ефективне використання водного ресурсу. Висока ефективність досягається завдяки точному визначенню моменту для початку та закінчення процесу зрошення коренів рослин. Початок та закінчення процесу поливу відбувається при досягненні заданих значень вологості. Для вимірювання поточного значення використовується датчик вологості, що встановлений в середовищі, де знаходяться кореневі системи рослин.

Види датчика вологості можуть бути різними, а саме: ємнісний, оптичний, резистивний, електронний та термісторний [23].

Ємнісний тип датчика вологості, являє собою повітряний конденсатор. При зміні вологи повітря діелектрична проникність змінюється, ця зміна вимірюється та можна отримати інформацію про насиченість повітря парами води. До переваг можна віднести простоту конструкції та низьку вартість, а недоліком є низка точність вимірювання.

На противагу ємнісному, оптичний датчик є найбільш точним приладом для вимірювання вологості. Принцип його роботи заснована саме на вимірюванні вмісту парів води в повітрі методом точки роси. Цей спосіб включає охолодження поверхні, як правило, металевого дзеркала, до температури, при якій вода на поверхні дзеркала знаходиться в рівновазі. При цій температурі маса води на поверхні дзеркала ні збільшується, ні зменшується, тобто пар над дзеркалом знаходиться в динамічній рівновазі з водяним конденсатом на дзеркалі [24].

Недоліком такого способу вимірювання вологості є велика ціна датчика. Використовуються датчики такого типу переважно в при роботі в лабораторіях.

Принцип дії резистивного датчика вологості полягає в вимірі опору металу, який знаходиться між двома електродами. Метал, що знаходиться між двома електродами називається оксид алюмінію. Даний метал дуже чутливий до зміни вологості, в залежності від чого його питомий опір змірюється. До переваг можна віднести їх невелику вартість.

Електронний датчик вологості повітря використовує властивості електроліту змінювати напругу при зміні вологості. Датчики такого типу дуже компактні та забезпечують хорошу точність в порівнянні з іншими типами датчиків вологості.

Термісторний датчик вологості в своїй будові має два термістори. Один з них знаходиться в герметичній камері з сухим повітрям, а інший має прямий доступ до повітряного середовища Пари вологи при попаданні на другий термістор на ньому частково конденсуються і випаровуються. При цьому змінюється його температура, а слідом опір. Порівняння опору двох термісторів і дає уявлення про ступінь зволоженості повітря.

Для модуляції контору зрошення кореневої системи рослин за допомогою плати Arduino обраний датчик DHT22 (рис 2.10). Даний датчик характеризується низьким енергоспоживанням має заводське калібрування, що забезпечує високу точність вимірювання з похибкою в 2% RH.

Рис 2.10 – Датчик вимірювання вологості DHT22

2.4 Опис та аналіз датчика кислотності

При використанні безґрунтової технології вирощування рослин забезпечення оптимального рівня кислотності розчину, є одним з основних параметрів для отримання стабільного врожаю. Оптимальний рівень pH дозволяє рослині отримувати всі доступні поживні речовини, що містяться в розчині, яким зрощують кореневу систему рослин. При недотримані заданого рівня кислотності в кращому випадку частина корисних речовин просто не буде сприйматися рослиною, а гіршому розчин з неправильним рівнем pH може нашкодити рослинам.

Для запобігання втрати вирощуваного врожаю методом аеропоніки в розроблюваній системі забезпечується вимірювання та контроль рівня кислотності розчинну, який поступає для зрошення кореневих систем рослин. Вимірювання рівня кислотності здійснюється pH-метром.

pH-метр – це прилад, призначення якого полягає в вимірюванні водневих показників різних матеріалів. Рівень pH характеризує ступінь активності іонів водню в кислотних, сольових і лужних, зокрема висококонцентрованих розчинах, H_2O (питній, акваріумній, технічній), агресивних середовищах, ґрунті, біосередовищах і рідинах людського організму та готових продуктах харчової галузі і системах виробництв, що безперервно контролюють технологічні нюанси [25].

Принцип роботи полягає в вимірі напруги між двома електродами зануреними в рідину (рис 2.11). Але один з електродів є індиканом та знаходиться в закритому корпусі зі боросилікатного скла, яке не боїться окислення та заповнене розчином pH відомого значення. Другий електрод називається вимірювальним. Корпус його також виготовлений з електропровідного боросилікатного скла з видутою на кінці кулькою. Корпус заповнений суспензією хлориду срібла в розчині соляної кислоти, в який занурений срібний дріт. При опусканні електродів в рідину електричне коло замикається [26].

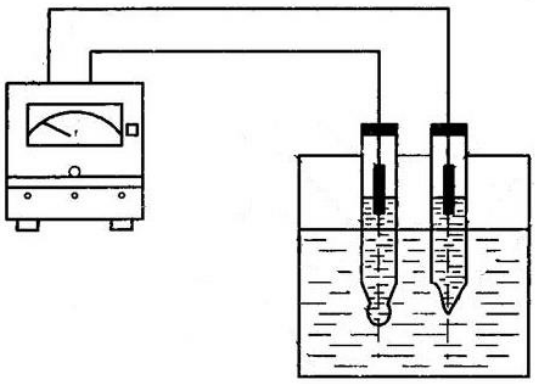


Рис 2.11 – Спрощена схема виміру рівня pH рідини

Для моделювання контуру забезпечення підтримання заданого рівня кислотності за допомогою плати Arduino обраний аналоговий датчик вимірювання рівня pH (рис 2.12)

Рис 2.12 – Датчик вимірювання рівня pH для Arduino

2.5 Опис та аналіз датчика рівня

Вирощування рослин аеропонним методом хоча і потребує в рази менше водного ресурсу в порівнянні з звичайним ґрунтовим методом вирощування, але вимагає забезпечення запасу розчину для поливу рослин на декілька циклів зрошення. Запас необхідний для забезпечення стабільного та безперебійного поливу рослин. Тому для вимірювання поточного рівня в ємності використовується датчик рівня.

За принципом роботи датчики рівня, можуть відрізнятись адже засновані на різних фізичних властивостях. Найбільшої популярності набули: візуальні, поплавкові, електричні та ультразвукові рівноміри.

Візуальні рівноміри є найпростішими вимірювачами рівня води та засновані на принципі дії сполучених посудин. Приклад візуальних рівнемірів показаний на рис. 2.13. Спостерігаючи за положенням рівня рідини в скляній трубці, можна визначити зміну поточний рівень в ємності. Перевагою таких рівнемірів є їх дешевизна, але можуть використовуватись тільки в системах де не потребується великої точності та швидкої реакції на зміну рівня в ємності.

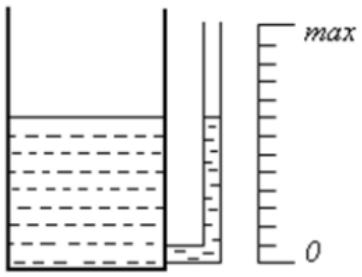


Рис 2.13 – Схема візуального рівнеміра

Принцип роботи поплавкових рівнемірів заснований на законі Архімеда. Розрізняють поплавкові рівнеміри широких та вузьких діапазонів. Рівнеміри широкого діапазону (рис 2.14) представляють собою поплавок 1, зв’язаний з противагою 2 за допомогою троса, перекинутого через ролики. В нижній частині противаги закріплено стрілку, яка показує на шкалі значення рівня в резервуарі [27].

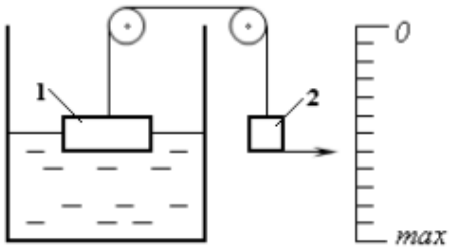


Рис 2.14 – Схема поплавкового рівнеміра широкого діапазону

Поплавкові рівнеміри вузького діапазону (рис. 2.15) зазвичай являють собою пристрої з кулеподібним поплавком діаметром 80/100мм, виконаним з нержавіючої сталі. Поплавок плаває на поверхні рідини і через штангу і спеціальні сальникові ущільнення з'єднується зі стрілкою вимірювального пристрою, або з перетворювачем кутових переміщень в уніфікований електричний або пневматичний сигнал.

Рис 2.15 – Схема поплавкового рівнеміра вузького діапазону

Рівнеміри вузького діапазону найчастіше використовуються для сигналізації про граничні значення рівня рідин. Вони мають необхідну плавучість, що дозволяє їм у незакріпленому стані перебувати на поверхні рідини в строго горизонтальному положенні. Процес перемикання запускається, коли давач відхиляється від горизонтального положення в будь-якому напрямку.

Найрозповсюдженішими електричними рівномірами є ємнісні та кондуктрометричні. У ємнісних рівнемірах використовуються діелектричні властивості рідин. Первинний вимірювальний перетворювач (ПВП) ємнісного рівнеміра являє собою електричний конденсатор, який перетворює зміну рівня рідини на пропорційне змінювання ємності. ПВП являє собою електрод або

електроди (циліндричні або у вигляді пластин), що опускаються у вимірюване за рівнем середовище.

Принцип ємнісних ПВП ґрунтується на різниці між діелектричною проникністю рідини та повітря і відповідно на залежності електричної ємності датчика від зміни рівня рідини. Для кожного значення рівня, ємність датчика визначається як ємність двох паралельно з'єднаних конденсаторів, один з яких утворюється частиною електродів перетворювача і рідиною, рівень якої вимірюється, а другий — іншою частиною електродів перетворювача і повітрям або парою рідини.

Кондуктрометричний метод вимірювання рівня є простим та надійним. Зазвичай використовується для сигналізатора рівня. Принцип роботи заснований на вимірюванні опору між загальним та сигнальними електродами [28]. На рис 2.16 показана схема роботи датчика.

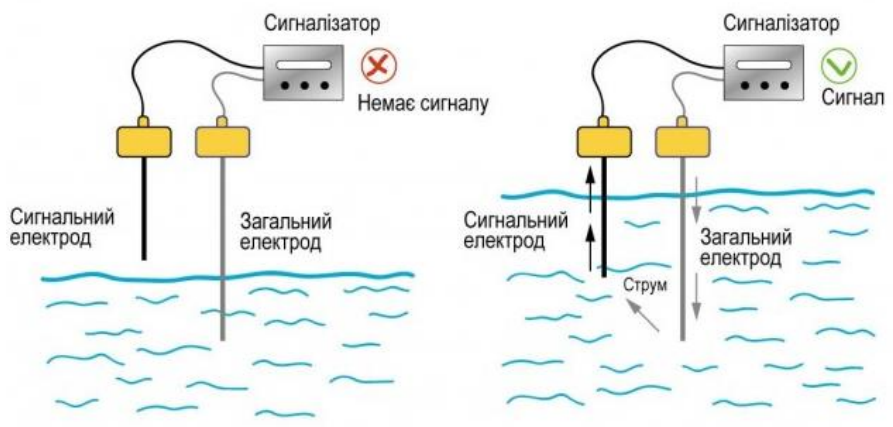


Рис 2.16 – Схема роботи кондуктрометричного датчика рівня

Загальний електрод повинен постійно знаходитися в контакті з рідиною. У випадку, якщо резервуар в якому здійснюється вимір рівня металевий, то загальним електродом можуть виступати стінки резервуару. Але такий спосіб вимірювання рівня доступний лише для струмопровідних рідин.

Принцип дії ультразвукових рівнемірів заснований на вимірі часу проходження ультразвукового імпульсу від випромінювача до поверхні рідини та назад. До переваг можна віднести те, що конструкція рівнеміра є

простою та надійною через відсутність рухомих частин. Подібні рівнеміри найчастіше встановлюють в верхню частину резервуару, що дає змогу уникнути прямого контакту з вимірюваною рідиною (рис 2.17). При відсутності рухомих частин та прямого контакту з рідиною датчики не потребують частого обслуговування. До недоліків можна віднести ціну, в порівнянні з іншими рівномірами [29].

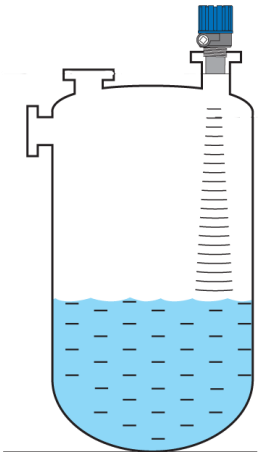


Рис 2.17 – Схема роботи ультразвукового рівнеміра

При впровадженні розроблюваного контуру вимірювання рівня в реальній теплиці на мою думку вибір ультразвукового рівнеміра буде найоптимальнішим рішенням. Тому що даний рівнемір задовольняє всі параметри, він надійний та не вимагає частого обслуговування.

Для модуляції контору підтримання заданого рівня в ємності за допомогою плати Arduino обраний ємнісний датчик рівня (рис 2.18).

Рис 2.18 – Датчик рівня для Arduino

Висновки до розділу 2

1. Були розглянуті основні можливості мікропроцесорної плати Arduino, її характеристики та розглянуто програмне середовище для програмування.

2. Проаналізовано доступні способи вимірювання температури та обраний датчик, що вибуде використовуватись при побудові моделі контуру системи.

3. Проаналізовано доступні способи вимірювання вологості та обраний датчик, що вибуде використовуватись при побудові моделі контуру системи.

4. Проаналізовано доступні способи вимірювання кислотності та обраний датчик, що вибуде використовуватись при побудові моделі контуру системи.

5. Проаналізовано доступні способи вимірювання рівня та обраний датчик, що вибуде використовуватись при побудові моделі контуру системи.

РОЗДІЛ 3. АЛГОРИТМИ СИСТЕМ КЕРУВАННЯ У ДИПЛОМНІЙ РОБОТІ

3.1 Розробка та побудова контуру клімат-контролю

Забезпечення дотримання заданого температурного режиму в теплиці є важливим фактором для отримання стабільного та прогнозованого кінцевого врожаю.

Контур клімат-контролю в своїй основі має наступний принцип роботи, що при змінні температури середовища вмикається або вимикається виконавчий механізм, що своєю роботою впливає на температурний показник. У ході розробки контуру була складена блок-схема (рис 3.1), яка ілюструє базовий принцип роботи за яким здійснюється робота контуру.

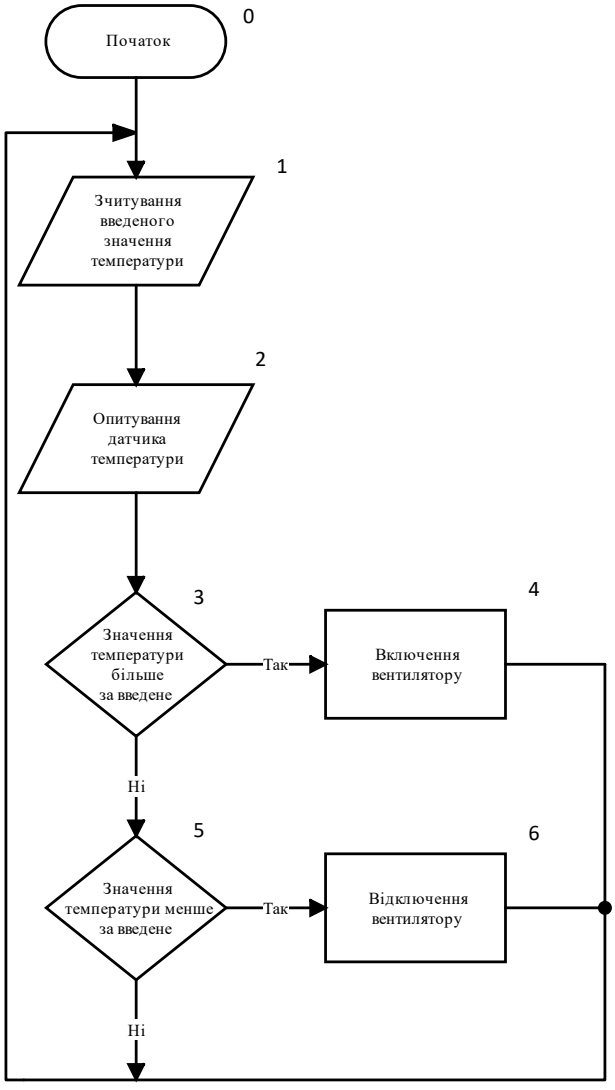


Рис 3.1 – Спрощена блок-схема контуру клімат-контролю

Маючи базове уявлення про роботу контуру та готовий алгоритм його роботи для перевірки його роботоспроможності за допомогою Arduino була зібрана перша модель контуру. В якості датчика температури використовується обраний DS18B20. У якості виконавчого елемента був використовується двигун постійного струму, який модулює роботу вентилятора. Схема підключення показана на рис. 3.2.

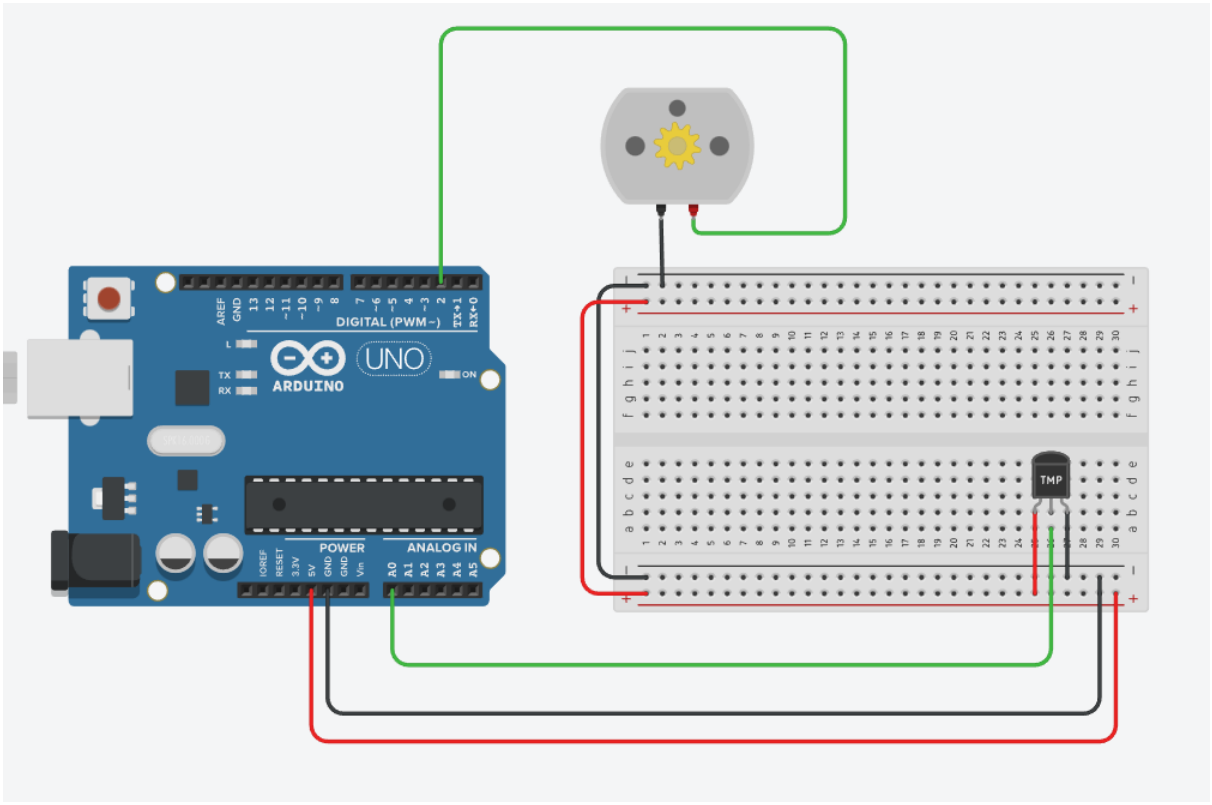


Рис 3.2 – Схема підключення датчика та виконавчого елемента

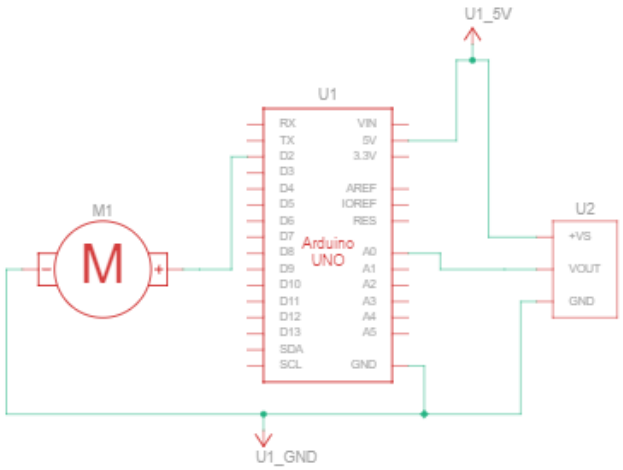


Рис 3.3 – Схема електрична схема підключення датчика та виконавчого елемента

У даній схемі підключення двигуна відбувається напряму до цифрового виходу плати, який може видавати тільки 0 або 5В, це означає що робота двигуна може бути тільки в двох режимах: виключений та включений. При включенні в роботу двигуна швидкість становиться максимальною, адже дана схема не має потреби в керуванні швидкістю двигуна.

Датчик вимірювання температури підключений до аналогового входу. Завдяки АЦП, що встановленого в плату, виміряна напруга поступає у вигляді числа від 0 до 1023. Для отримання вимірного значення в градусах Цельсія необхідно отримане значення перевести в мВ за формулою (3.1). В залежності від моделі датчика помножити на коефіцієнт, для обраного датчика температури коефіцієнт дорівнює 30.

$$U_{датчика} = \left(\left(ArduinoSensor / 1023 \right) \cdot 5000 \right) \tag{3.1}$$

$$t = U_{датчика} \cdot 30 \tag{3.2}$$

Для виведення інформації про поточний стан вимірюваної величини використовується Serial Monitor в вікні якого відображається поточна виміряна величина в даному випадку – це температура навколишнього середовища (рис 3.3).

Рис 3.3 – Відображення виміряної температури

Складена схема добре ілюструє базовий принцип роботи контуру клімат-контролю, але цього є не достатнім, адже метою реалізації контурів системи автоматичного керування теплицею за допомогою платформи Arduino є не тільки перевірка роботоспроможності, а і можливість досліди роботу системи в стресових ситуаціях. Причинами появи нетипових ситуацій при роботі системи можуть бути різні обставини, адже чим складніша система, тим більше критичних вузлів вона має.

Для розроблюваного контуру найбільшою загрозою є недотримання вимірюваного технологічного параметру в заданому діапазоні. Для контуру клімат-контролю таким параметром є температура, адже велике відхилення вимірюваного значення від заданого, як в більшу так і в меншу сторону може сильно нашкодити рослинам, що знаходяться в теплиці.

Для плавного та контрольованої зміну вимірювального значення на плату додатково встановлений потенціометр, але додавання тільки даного елемента в схему може призвести до небажаного конфлікту між інформацією, яка отримується від датчика та від потенціометра. Для уникнення такої проблеми була на плату встановлена кнопка, яка відповідає за зміну режиму роботи розробленої системи між автоматичним та ручним. Для індикації зміни режиму управління використовується зелений світлодіод, що загоряється при активації ручного режиму управління та гасне при його відключенні. За замовчуванням схема працює в автоматичному режимі.

Ще одним важливим фактором є сповіщення персоналу або користувача про виявленні системою проблеми. Це може бути візуальна, звукова чи комбінація з обох сигналізація. Тому для створення повноцінної моделі контуру клімат контролю є необхідним втілення системи сигналізації.

Постає питання про що саме повинна сповіщавати сигналізація? Для даного контуру критичним є зміна значень вимірюваного параметру від якого залежить кінцевий результат, тому логічним є висновок, що сигналізація повинна сповіщати про нетипові коливання виміряного значення. Для контуру

використано два типи сигналізації: візуальну та звукову. Сповіщення має два рівні роботи: попереджувальний та критичний.

Перший рівень є сповіщенням для персоналу або користувача про те, що температура вийшла за задані межі, але не досягла критичного значення. Та реалізований тільки візуальною сигналізацією за допомогою жовтого та білого світлодіодів. Жовтий світлодіод загоряється при не критичному перевищенні температури, а білий навпаки при опусканні вимірюваного значення.

Другий рівень є аудіо-візуальною сигналізацією який сповіщає про досягнення критичного рівня вимірювального значення. Візуальна частина сигналізації реалізована завдяки червоному та синьому світлодіоду, а звукова завдяки п’єзо динаміку. Червоний світлодіод спрацьовує при досягненні верхньої контрольної точки, а синій навпаки при опусканні вимірювального значення до нижньої критичної точки.

Для контуру клімат-контролю критичні точки встановленні на рівні в задане значення температури + 50% для верхньої межі та задане значення температури - 50% для нижньої.

Маючи уявлення про повноцінну роботу контуру та перелік усіх задіяних елементів електронних компонентів, було розроблено алгоритм роботи контуру клімат-контролю (рис 3.4) та зібрано модель за допомогою плати Arduino (рис 3.8).

Блоки, що зображені під порядковими номерами 5, 7, 8 обозначають підпрограми, що виконуються при задіяні блока. Дані блоки призначенні для візуального спрощення та облегшення зчитування інформації. Блок схеми підпрограм показані на рис. 3.5, 3.6, 3.7.

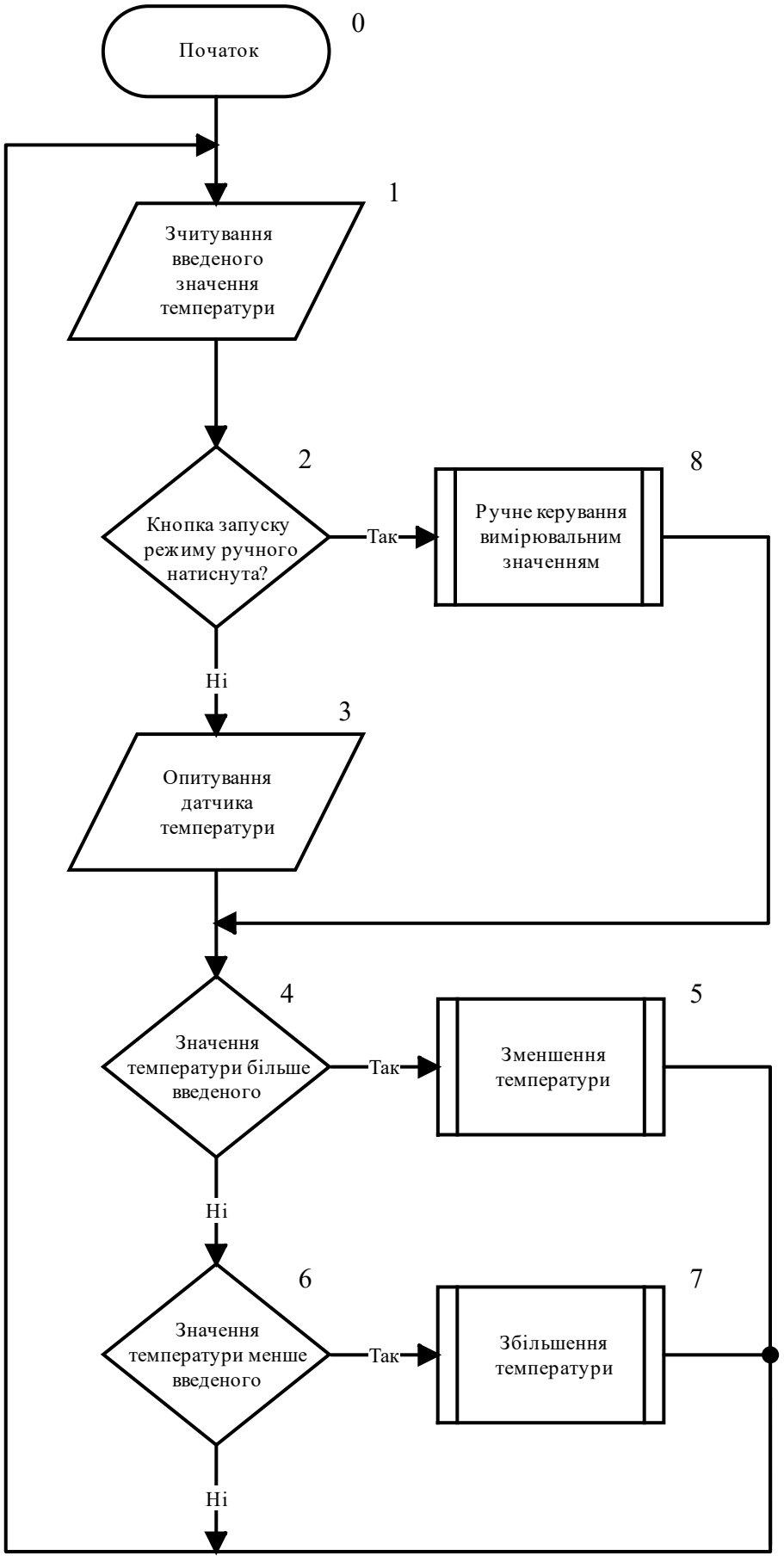


Рис 3.4 – Блок схема роботи контуру клімат-контролю

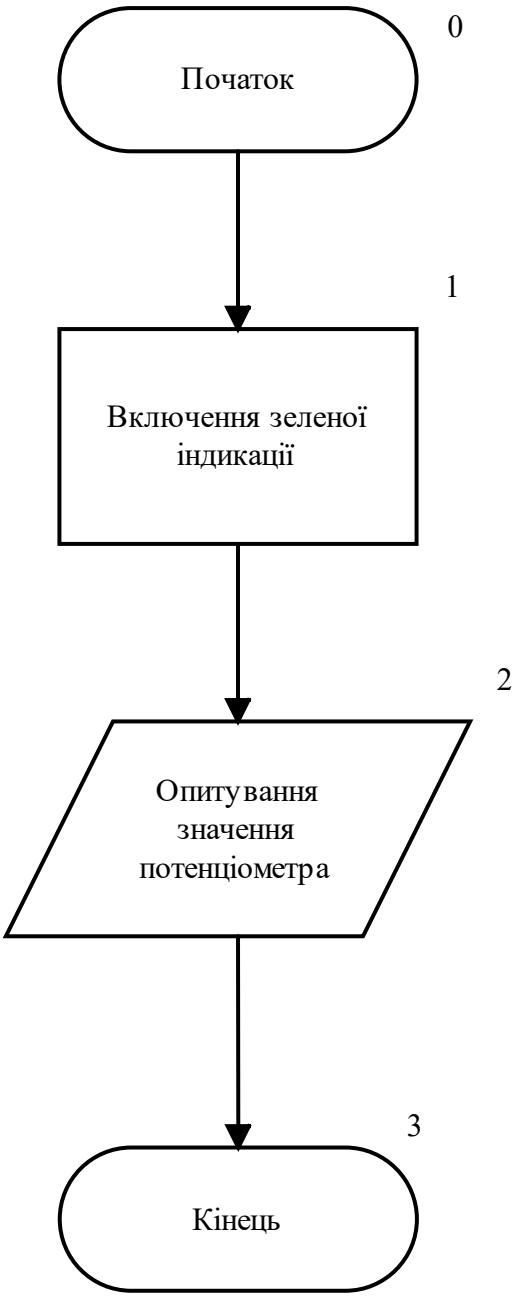


Рис 3.5 – Блок схема ручного керування значенням вимірювальної величини

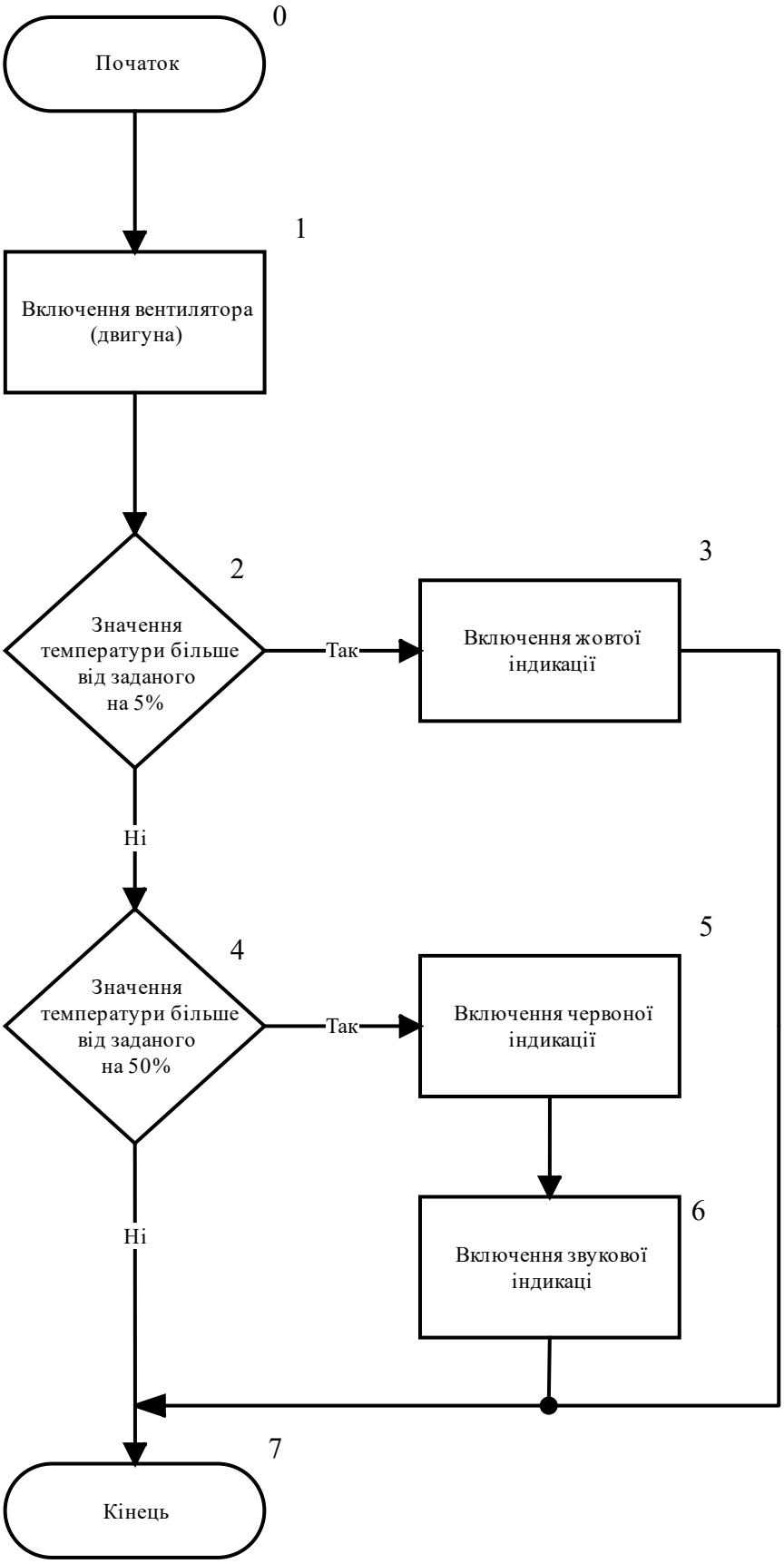


Рис 3.6 – Блок схема реагування
контуру на підвищення температури

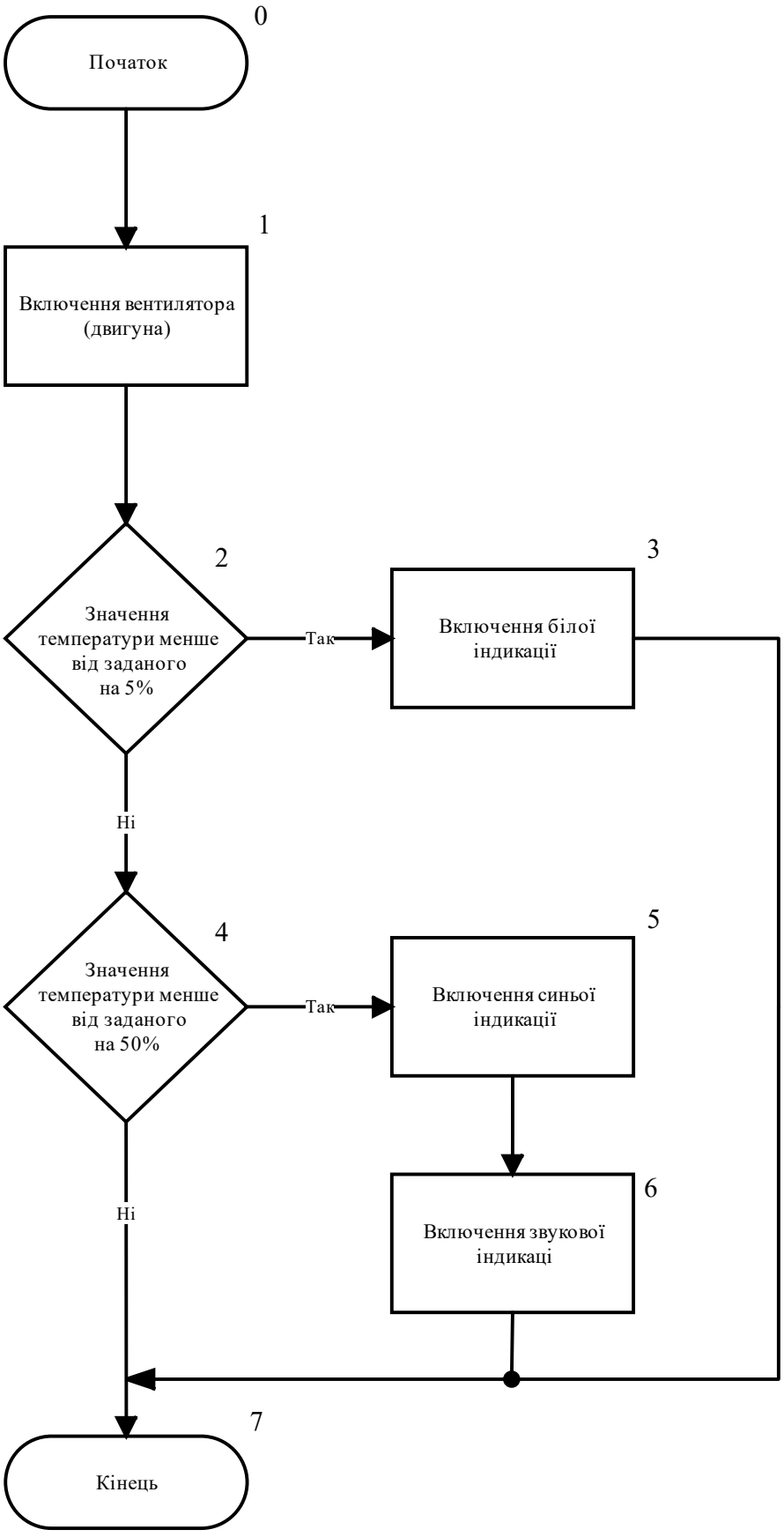


Рис 3.7 – Блок схема реагування
контуру на зниження температури

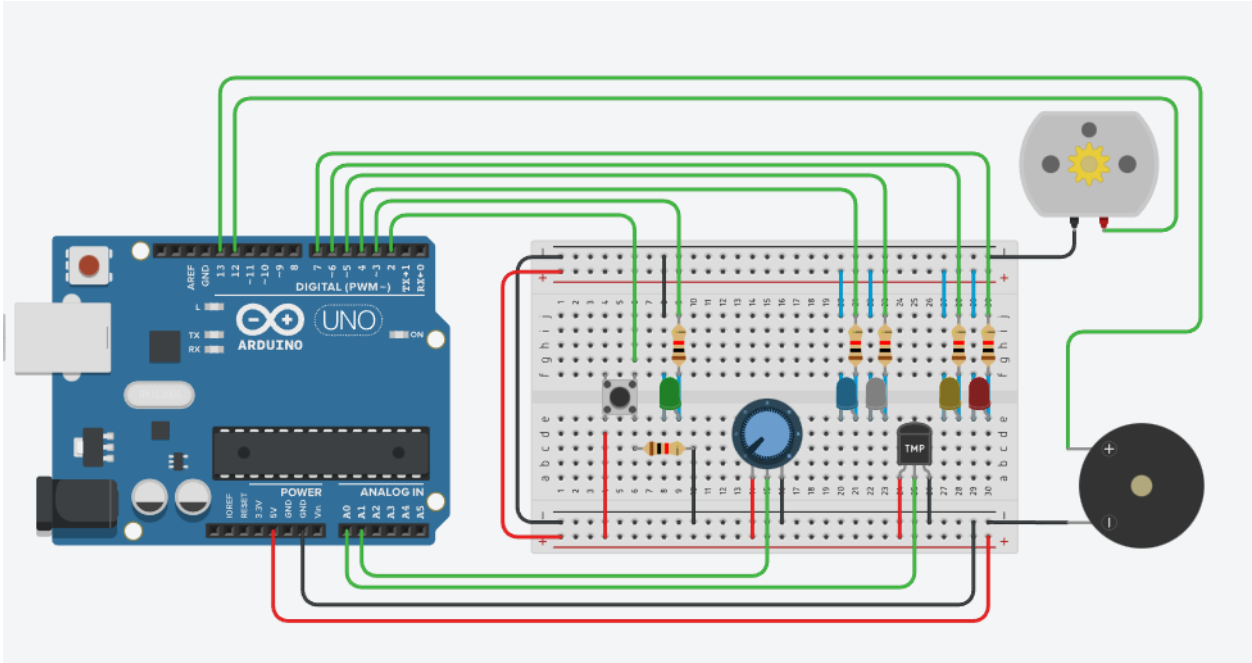


Рис 3.8 – Схема підключення компонентів до плати Arduino

Складена схема є тільки частиною виконаного завдання, адже для запуску в роботу необхідний програмний код в якому прописаний порядок роботи схеми.

Першим чином для підключених виходів плати присвоюється назви за якими одразу зрозуміло який елемент під’єднано до виходу (рис 3.9).

```
const int temperatureSensor = 0;
const int potentiometer = 1;
const int btn = 2;
const int greenLed = 3;
const int blueLed = 4;
const int whiteLed = 5;
const int yellowLed = 6;
const int redLed = 7;
const int DCMotor = 12;
const int piezo = 13;
```

Рис 3.9 – Опис виходів плати

Після цього вказуються змінні, що будуть використовуватись в проекті (рис 3.10) та в налаштуваннях за допомогою команди pinMode(), вказується назва входу та його режим роботи (рис 3.11).

```
//Змінні
int userTemperature = 25;
int temperature;
//верхня межа
float highLimit = (userTemperature*150)/100;
//нижня межа
float lowLimit = (userTemperature*50)/100;
// вихід за дозволені межі
float highLevelOfAllowedDiapazon = (userTemperature*105)/100;
float lowLevelOfAllowedDiapazon = (userTemperature*95)/100;
//кнопка
boolean isGreenLedOn = LOW;
int buttonState;
int lastButtonState = LOW;
//усунення брязкоту кнопки
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;
```

Рис 3.10 – Змінні, що будуть використовуватись в проекті

```
void setup() {
  Serial.begin(9600);
  pinMode(temperatureSensor, INPUT);
  pinMode(potentiometer, INPUT);
  pinMode(btn, INPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  pinMode(whiteLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(redLed, OUTPUT);
  pinMode(DCMotor, OUTPUT);
  pinMode(piezo, OUTPUT);
}
```

Рис 3.11 – Присвоєння режиму роботи виходів плати

При першому запуску всі виконавчі та індикаторні елементи знаходяться в вимкненому стані. Задана температура, яку повинна підтримуватись встановлена на рівні 25 °C.

На платі використовується кнопка, що означає необхідність в усуненні такого явища, як брязкіт кнопки. На рис 3.12 показано як вирішується ця задача за допомогою функції delay(100) (затримка в 100 мс). Після 100 мс після натиснення кнопки ефекту брязкоту кнопки не існує – кнопка у стабільному

стані. Функція debounce повертає інформацію про те, натискувалася кнопка чи ні, але вже без ефекту брязкоту кнопки!

```
boolean debounce(int last){
  int reading = digitalRead(btn);

  if (reading != lastButtonState) {
    delay(100);
    reading = digitalRead(btn);
  }

  if (reading != buttonState) {
    buttonState = reading;
    if (buttonState == HIGH) {
      isGreenLedOn = !isGreenLedOn;
    }
  }
  digitalWrite(greenLed, isGreenLedOn);
  lastButtonState = reading;
  return isGreenLedOn;
}
```

Рис 3.12 – Функція усунення брязкоту кнопки

Для зчитування інформації з аналогових входів використовується команда analogRead(), але значення отримуються цифрами від 0 до 1023. Для переведення в градуси Цельсія для значення від датчика задіяна формула, а для отриманих значень від потенціометра використана команда map(), яка перетворює один діапазон в заданий. Тому діапазон значень потенціометра від 0 до 50 °C.

```
if(isManualMode == LOW){
    int sensorData = analogRead(temperatureSensor);
    float mV = (sensorData/1023.0)*5000;
    temperature = mV/30;
  } if(isManualMode == HIGH){
    float readPotentiometer = analogRead(potentiometer);
    temperature = map(readPotentiometer, 0, 1023, 0, 50);
  }
```

Рис 3.13 – Зчитування даних в ручному та автоматичному режимах

Після отриманий даних в залежності від значень, що набула змінна temperature, вище або нижче заданого значення userTempetature спрацьовує одна з двох умов (рис 3.14).

```
if(temperature > userTemperature){
  digitalWrite(DCMotor, HIGH);
  //Відключення сигналізації про пониження температури
  digitalWrite(whiteLed, LOW);
  digitalWrite(blueLed, LOW);
  digitalWrite(piezo, LOW);
  //
  if(temperature > highLevelOfAllowedDiapazon){
    digitalWrite(yellowLed, HIGH);
  }
  if(temperature > highLimit){
    digitalWrite(redLed, HIGH);
    digitalWrite(piezo, HIGH);
  }
}

if(temperature < userTemperature){
  digitalWrite(DCMotor, LOW);
  //Відключення сигналізації про підвищення температури
  digitalWrite(yellowLed, LOW);
  digitalWrite(redLed, LOW);
  digitalWrite(piezo, LOW);
  //
  if(temperature < lowLevelOfAllowedDiapazon){
    digitalWrite(whiteLed, HIGH);
  }
  if(temperature < lowLimit){
    digitalWrite(blueLed, HIGH);
    digitalWrite(piezo, HIGH);
  }
}
```

Рис 3.14 – Лістинг тіла програми

В залежності від умови обов’язково спрацьовує команда по включенню або відключення двигуна, це можливо завдяки команді digitalWrite() та значень HIGH для включення, LOW для виключення. Для коректної роботи сигналізації, є необхідним попередньо впевнитись про те що сигналізація протилежного значення відсутня. Сигналізація про поточний стан спрацьовує коли значення змінної temperature вище, або менше за значення змінних highLimit, lowLimit, які являються критичними межами та highLevelOfAllowedDiapazon, lowLevelOfAllowedDiapazon, які являються попередженням про вихід вимірюваного значення за дозволенні межі.

3.2 Розробка та побудова контуру рівня наповненості ємності

При застосуванні аеропонної технології вирощування рослин є необхідність мати запас розчину для зрошення кореневої системи рослин. Для цього використовується ємність в якій може зберігатись рідини на декілька циклів поливу. Забезпечення належного рівня заповнюваності ємності є важливим фактором в досяганні стабільного та прогнозованого продукту, адже недостатній рівень наповненості може стати проблемою в разі екстреної ситуації, що може призвести до значних втрат серед рослин.

Метою розробки контуру рівня для даної системи є забезпечення автоматичного підтримання рівня в ємності для зберігання розчину. Для цього в ємність встановлений датчик рівня, який забезпечує вимірювання поточного рівня. Також визначенні контрольні точки по досягнені яких відбувається запуск процесу наповнення, або навпаки його відключення.

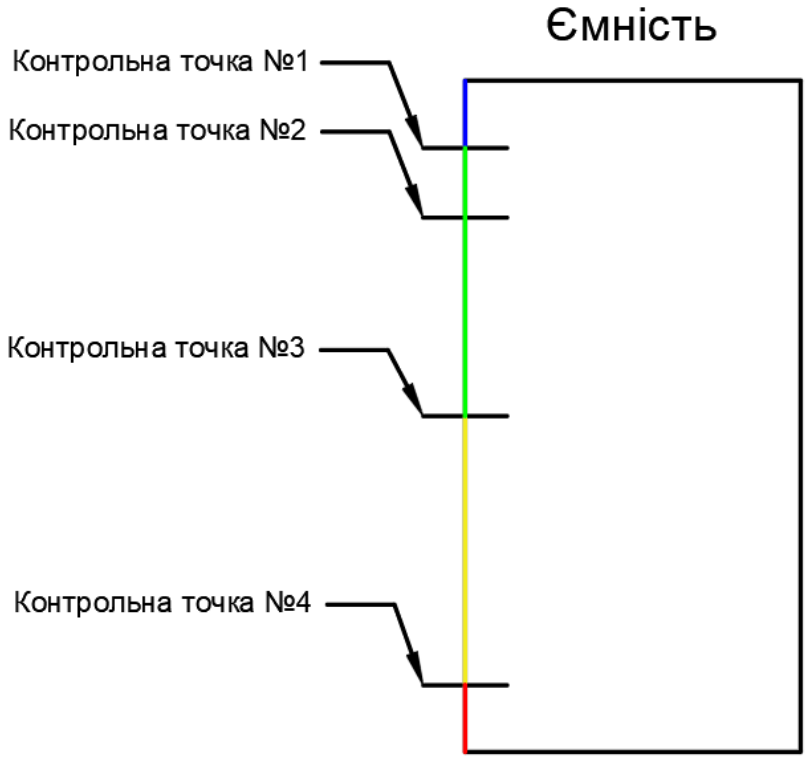


Рис 3.15 – Спрощена схема зон ємності

На рис 3.15 показано схему розташування визначених точок, всього їх чотири. Точки під порядковими номерами 1 та 4 є критичними, адже по

досягненні їх автоматично вимикаються засоби сповіщення, які сповіщають про небезпеку переповнення ємності, або навпаки про її критично малий рівень наповненості. При досягненні рівнем точки під номером 3 запускається система наповнення резервуару, яке відбувається до тих пір, поки рівень наповненості ємності не досягне контрольної точки №2. Проміжок між точками 1 та 2 є необхідним для уникнення ризику переповнення ємності, адже технологія аеропоніки передбачає, що розчин який не був поглинутий рослиною транспортується в ємність для повторного використання.

Визначенні контрольні точки також розбивають ємність на декілька зон, що дозволяє створити світлодіодну індикацію, яка відображатиме поточний рівень наповненості резервуару. Зона, межами якої є точки з порядковими номерами 1 та 3 означає, що рівень наповненості знаходиться в безпечній зоні та для індикації обраний зелений світлодіод. Зона з межами, що знаходиться між контрольними точками з номерами 3 та 4 є небажаною, адже означає, що рівень в ємності опустився нижче половини, про що буде сигналізувати жовтий світлодіод. Про опускання рівня нижче точки в під номером 4 означає, що ємність майже порожня, про це буде сигналізувати червоний світлодіод. При піднятті рівня вище точки під номером 1 засвітиться синій світлодіод, що означитиме небезпечний рівень наповненості.

У ході роботи на розробкою роботи контуру було створено блок схему, що повністю описує принцип роботи контуру (рис. 3.16). Блок схеми підпрограм, що використовуються в блок-схемі показані на рис 3.17 , рис 3.18, рис 3.19. Також зібрано модель за допомогою плати Arduino, схема підключень, якої показана на рис 3.20.

В даній схемі не використовується ручний режим роботи плати при якому вимірюванні значення задаються та змінюються за допомогою потенціометра. Розмір обраного датчика рівня забезпечує комфортну роботу в невеликій ємності в яку поміщений датчик. Зміна рівня відбувається шляхом додавання води в ємність або її виливання з неї.

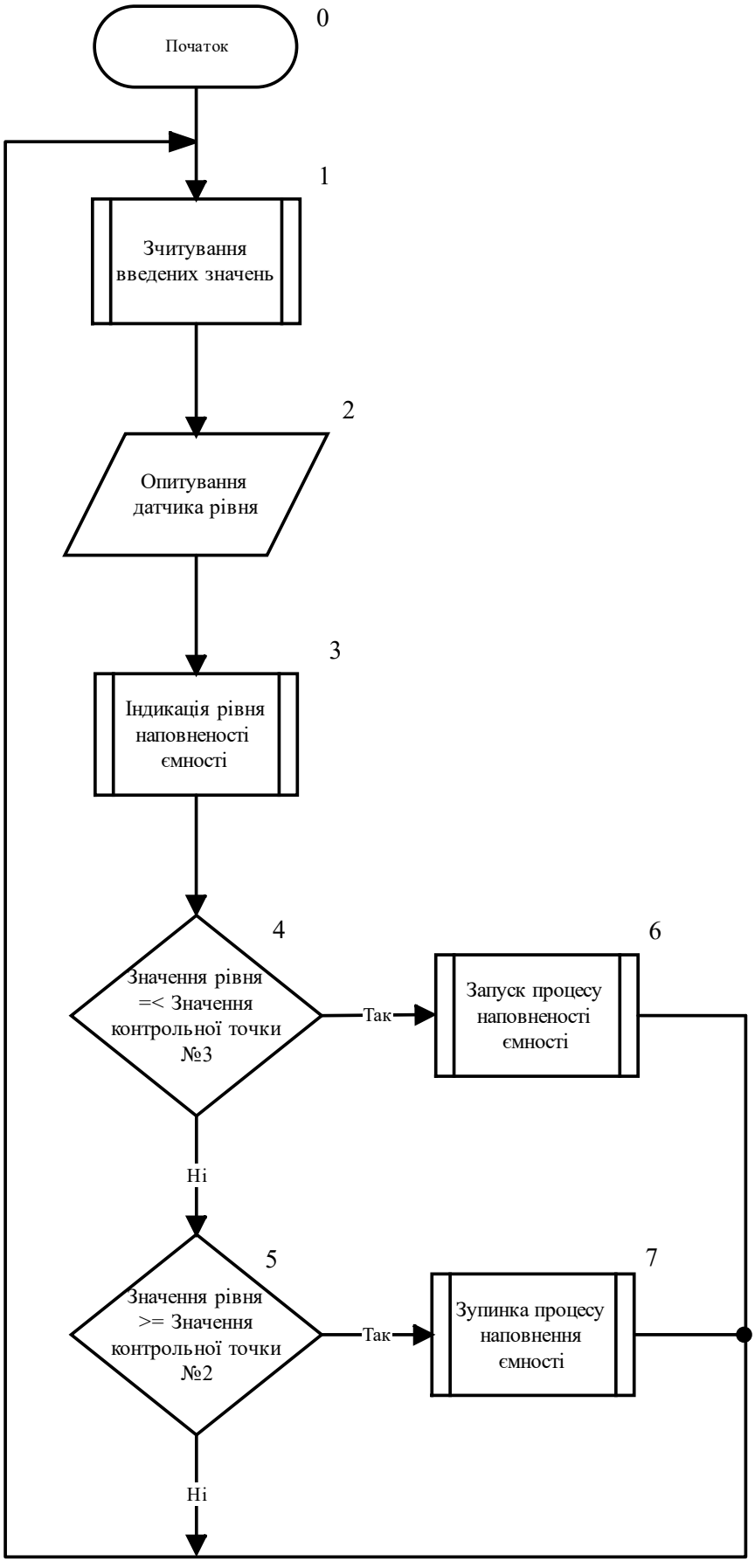


Рис 3.16 – Блок-схема роботи контуру підтримання рівня

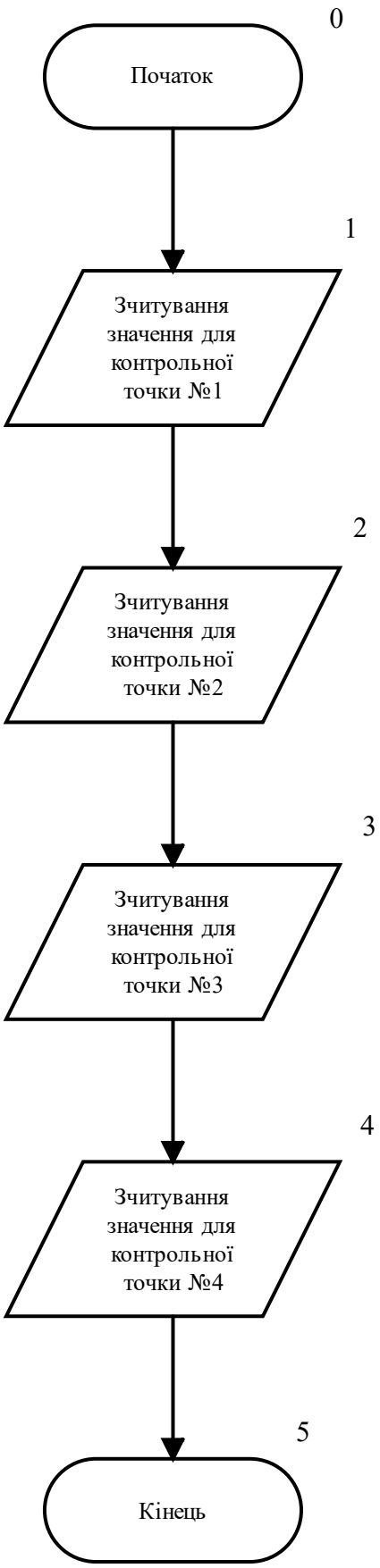


Рис 3.17 – Блок-схема підпрограми “Зчитування заданих значень”

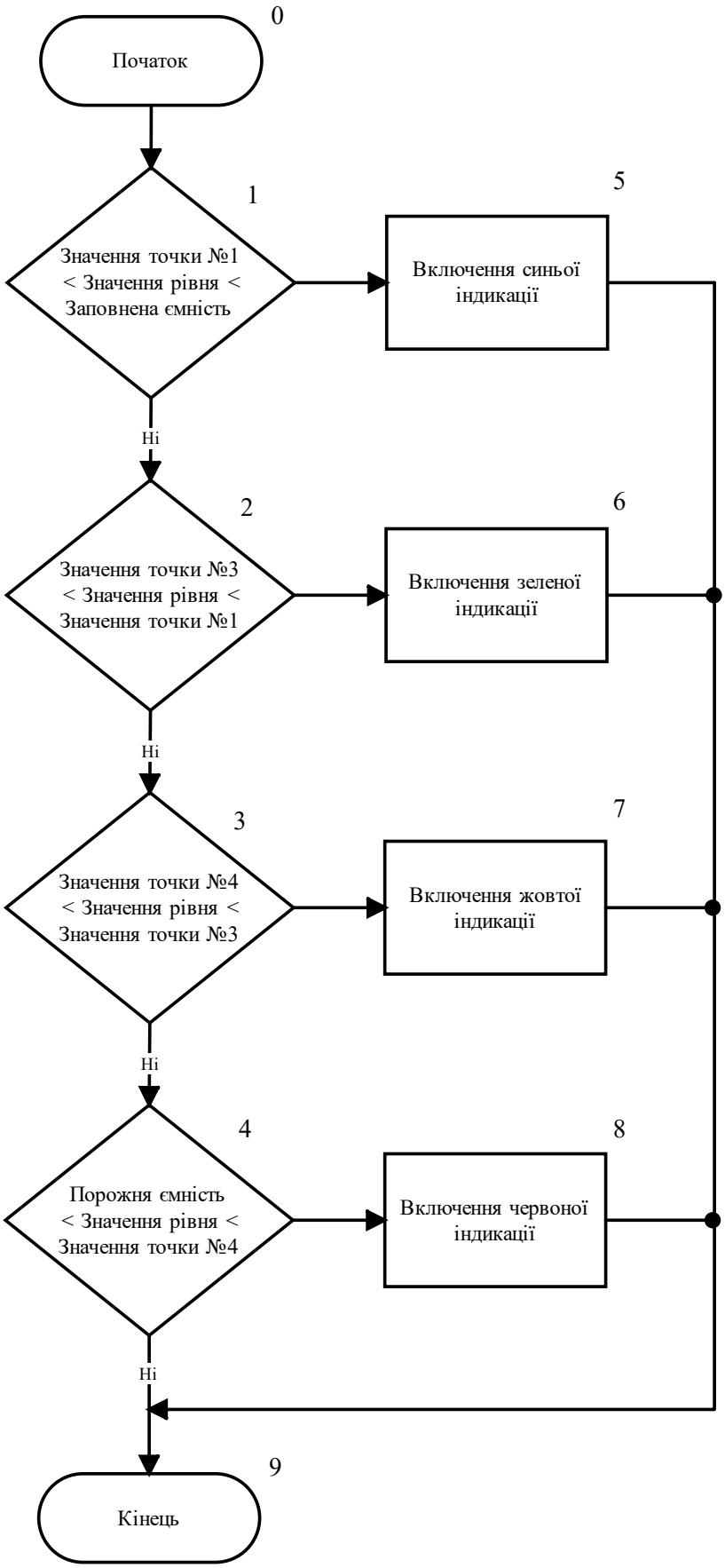


Рис 3.18 – Блок-схема підпрограми

“Індикація рівня наповненості”

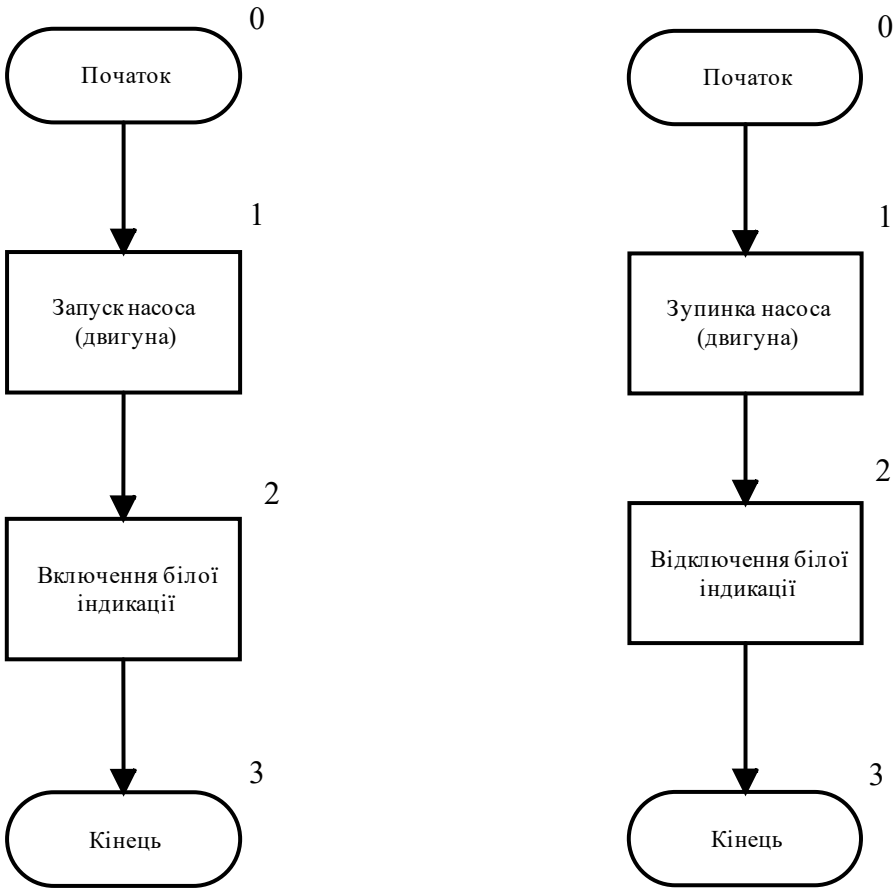


Рис 3.19 – Блок-схема підпрограми:

а) “Запуск процесу наповнення”; б) “Запуск процесу наповнення”

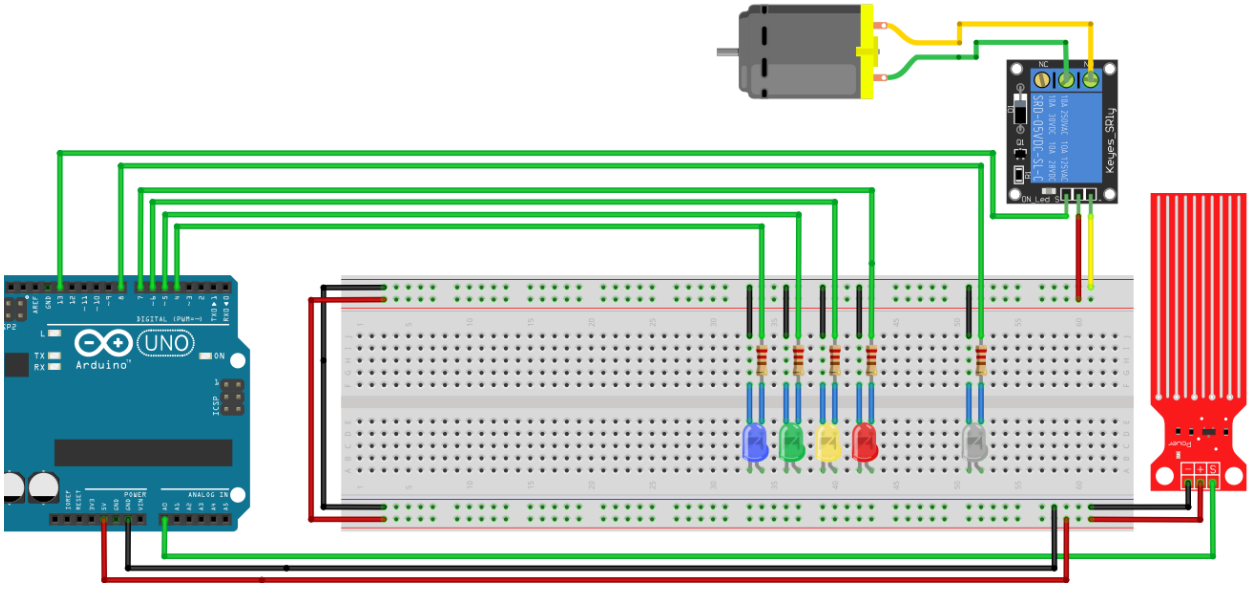


Рис 3.20 – Схема підключень компонентів

до плати Arduino

В схему входять наступні компоненти: двигун постійного струму, що підключений до 13 виходу плати через модуль реле, оранжевий діод підключений до 8 цифрового виходу плати та є індикацією роботи двигуна, група діодів, що призначені для індикації поточного рівня наповненості ємності підключені до виходів плати під номером 4, 5, 6, 7. Датчик рівня в даній схемі є єдиним компонентом для введення значень та підключений до аналогового входу під номером 0.

Маючи розроблений алгоритм роботи контуру та зібрану схему, останнім кроком є написання програми.

Першим кроком є опис задіяних виходів плати (рис 3.21).

```
const int waterLevelSensor = 0;
const int whiteLed = 3;
const int blueLed = 4;
const int greenLed = 5;
const int yellowLed = 6;
const int redLed = 7;
const int DCMotor = 13;
```

Рис 3.21 – Опис виходів для плати Arduino

Після цього задаються змінні, які будуть використовуватись в проекті (рис 3.22). Значення об’єму ємності задається в літрах, а критичні точки вираховуються в відсотковому значені.

```
//Змінні
int currentWaterLevel;
//контрольна точка №1 встановлена на висоті в 90% від об`єму ємності
float controlPoint__1 = 90;
//контрольна точка №2 встановлена на висоті в 80% від об`єму ємності
float controlPoint__2 = 80;
//контрольна точка №3 встановлена на висоті в 50% від об`єму ємності
float controlPoint__3 = 50;
//контрольна точка №3 встановлена на висоті в 10% від об`єму ємності
float controlPoint__4 = 10;

boolean autoMode = true;
boolean firstStart = true;
boolean isPumpOn = false;
extern volatile unsigned long timer0_millis;
```

Рис 3.22 – Опис змінних проекту

```
void setup() {
  Serial.begin(9600);
  pinMode(waterLevelSensor, INPUT);
  pinMode(whiteLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(redLed, OUTPUT);
  pinMode(DCMotor, OUTPUT);
}
```

Рис 3.23 – Налаштування режимів роботи описаних виходів плати

На рис 3.23 показано, що для кожного задіяного виходу за допомогою команди pinMode() прописаний режим роботи. Для даної схеми введення інформації відбувається тільки через аналоговий датчик рівня, тому він єдиний має режим роботи INPUT.

```
void loop() {
  if (autoMode == false) {
    //опитування датчика рівня
    float waterLevelSensorData = analogRead(waterLevelSensor);
    currentWaterLevel = map(waterLevelSensorData, 0, 1023, 0, 100);
  } else if (autoMode == true) {
    //Модуляція зміни вимірюваного значення
      modulateWaterLevelBehavior();
    }

  //функція для вимикання індикації
  offIndicate();
  //функція для вимикання індикації поточного рівня
  waterLevelLedIndication(currentWaterLevel);

  if (currentWaterLevel <= controlPoint__3) {
    //Змінна для початку процесу поливу
    isPumpOn = true;
  }

  if(currentWaterLevel >= controlPoint__2){
    //Змінна для зупинки процесу поливу
    isPumpOn = false;
  }
  //Насос
  DCwork()
  Serial.println(currentWaterLevel);
}
```

Рис 3.24 – Тіло програми роботи моделі контуру рівня

На рис 3.24 показано тіло програми за якою працює зібрана схема. За допомогою команди analogRead() відбувається зчитування інформації від датчика рівня, значення якого мають значення від 0 до 1023. Для переведення поточного значення рівня з числового діапазону від 0 до 1023 в відсотковий від 0 до 100% використовується команда map(). Далі в залежності від умови змінюється значення змінної isPumpOn, що відповідає за початок або зупинку процесу наповнення. Процес запуску або зупинки насосу здійснюється за допомогою команди digitalWrite() та винесений в окрему функцію DCwork() (рис 3.28).

Для модулювання поведінки вимірювального значення рівня наповненості ємності використовується функція modulateWaterBehavior() (рис 3.27). При роботі контуру, зміна значення рівня набуває лінійної поведінки, яку модулює дана функція. У ході роботи над функцію модулювання зміни вимірювального значення в часі, було побудовано графік в якому було визначено період за який відбувається повний цикл роботи контуру, а також показано як змінюється вимірювальне значення в часі (рис 3.25).

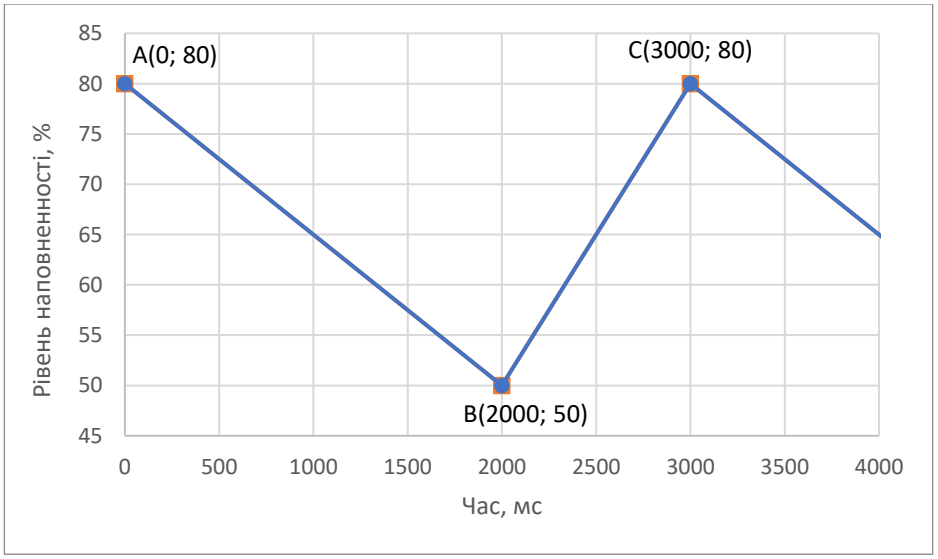


Рис 3.25 – Графік зміни значення рівня в часі

Для даного контуру враховуючи, що умови в теплиці не змінні період функції визначений в 3000 мс, найвищою та найнижчою точками є значення заданого діапазону, що дорівнюють 80 та 50 відсотків рівня наповненості.

Для визначення проміжного значення рівня в конкретний момент часу було визначено рівняння прямої за формулою 3.1 для прямих A(0; 80) B(2000; 50) та B(2000; 50) C(3000; 80).

$$y = kx + b \tag{3.1}$$

Для прямої AB рівняння має наступний вигляд:

$$y_1 = \left(-\frac{30}{2000} \cdot x \right) + 80 \tag{3.2}$$

Для прямої BC рівняння має наступний вигляд:

$$y_2 = \frac{30}{1000} \cdot x - 10 \tag{3.3}$$

Відстежування часу роботи розробленої схеми відбувається за допомогою функції millis(), яка в даному випадку повертає значення від 0 до 3000 мс. Блок-схема алгоритму за яким працює функція модлювання зміни вимірювального значення рівня показана на рис 3.26.

За даним алгоритмом робота функції полягає в обрахунку поточного значення рівня наповненості ємності в конкретний момент часу. Якщо значення часу x знаходиться в діапазоні від 0 до 2000 мілісекунд, то значення рівня є спадаючим, отже визначається за формулою 3.2. При отримані значення часу x, яке знаходиться в межах від 2000 до 3000 мілісекунд, поточне значення визначається за формулою 3.3.

Режим роботи зібраної схеми залежить від значення змінної autoMode. За замовчуванням значення змінної дорівнює false, що означає схема працює отримує інформацію про поточне значення рівня наповненості від датчика рівня.

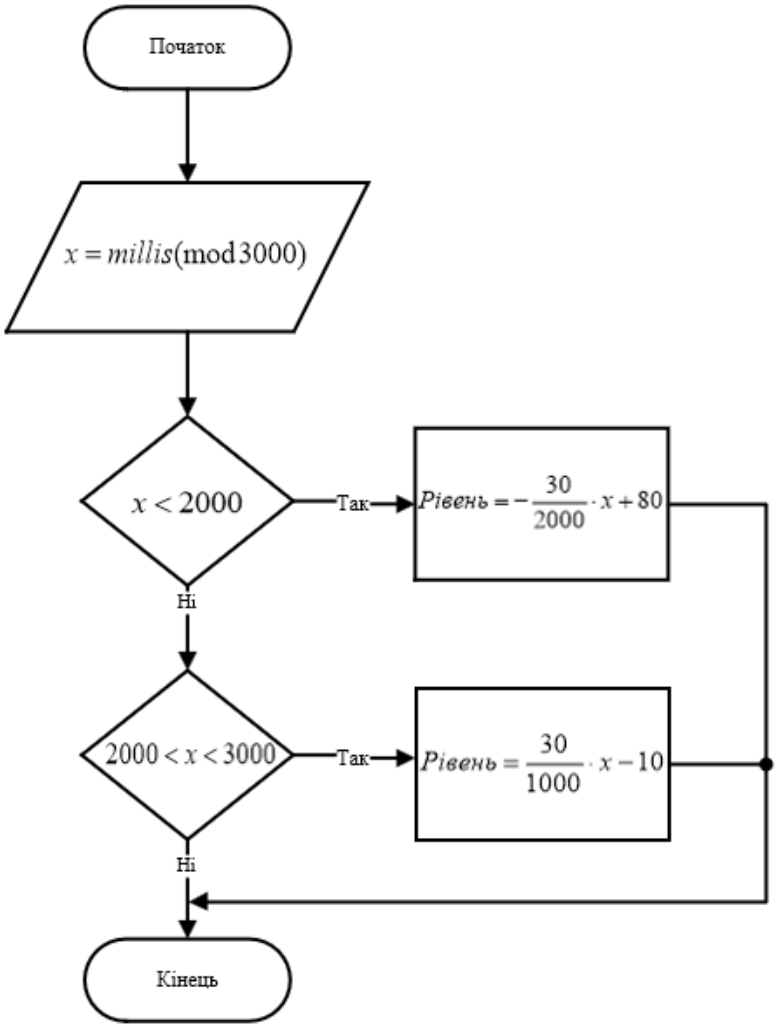


Рис 3.26 – Блок-схема алгоритму роботи функції модулювання

```
float modulateWaterLevelBehavior(){
  float x = millis();

  if(x < 2000){
    currentWaterLevel = (-30*x/2000) + 80;
  } else if(x > 2000 && x <= 3000){
    currentpHLevel = (30*x/1000) - 10;
  }

//обнулення millis
  if(x >= 3000){
        noInterrupts ();
        timer0_millis = 0;
        interrupts ();
  }
}
```

Рис 3.27 – Функція modulateWaterBehavior

```
boolean DCwork (){
  if(isPumpOn == true){
    //включення двигуна
    digitalWrite(DCMotor, HIGH);
    //включення індикації роботи дивигуна
    digitalWrite(whiteLed, HIGH);
  } else if (isPumpOn == false){
    //відключення двигуна
    digitalWrite(DCMotor, LOW);
    //відключення індикації роботи дивигуна
    digitalWrite(whiteLed, LOW);
  }
}
```

Рис 3.28 – Функція керування роботи насосу DCwork

В даній програмі присутні дві функції, які відповідають за індикацію поточного рівня наповнення ємності рис 3.29.

```
boolean offIndicate(){
  digitalWrite(blueLed, LOW);
  digitalWrite(greenLed, LOW);
  digitalWrite(yellowLed, LOW);
  digitalWrite(redLed, LOW);
}

boolean waterLevelLedIndication(int waterLevel){
  if(waterLevel <= 100 && waterLevel >= controlPoint__1){
    digitalWrite(blueLed, HIGH);
  } if(waterLevel <= controlPoint__1 && waterLevel >= controlPoint__3) {
    digitalWrite(greenLed, HIGH);
  } if(waterLevel <= controlPoint__3 && waterLevel >= controlPoint__4) {
    digitalWrite(yellowLed, HIGH);
  } if(waterLevel <= controlPoint__4 && waterLevel >= 0) {
    digitalWrite(redLed, HIGH);
  }
}
```

Рис 3.29 – Функції програми індикації поточного рівня

Функція waterLevelLedIndication() повністю спирається на розроблену блок-схему, що зображена на рис 3.18, де в залежності від значень поточного рівня виконується одна з заданих умов.

Функція offIndicate() необхідна для переведення світлодіодів в початковий стан. Це необхідно для коректного відображення поточного рівня за допомогою світлодіодів при переході від однієї вимірюваної зони ємності до іншої.

3.3 Розробка алгоритму побудови підтримки заданого рівня кислотності в середовищі

Рівень кислотності середовища в якому здійснюється ріст та розвиток рослини є важливим параметром, що впливає на кінцеву якість та кількість отриманого врожаю. Важливість такого параметру полягає в тому, що значення рівня pH середовища впливає на швидкість та якість поглинання рослиною поживних речовин для свого подальшого розвитку.

Вимірювання рівня кислотності здійснюється в діапазоні від 0 до 14, де 0 це найкисліше середовище, а 14 найбільш лужне середовище. Кожне значення pH має свій колір при перевірці його лакмусовою стрічкою, так кисле середовище має червоний колір, а лужне середовище темно фіолетовий. Проміжні значення показані на шкалі кислотності рис 3.28.

Оптимальні значення рівня кислотності можуть дещо відрізнятись для різних культур рослин, але в середньому підтримання рівня кислотності середовища в діапазоні від 5,5 до 6,5 pH забезпечить максимальну продуктивність росту для більшості сільськогосподарських культур, які вирощуються в теплицях.

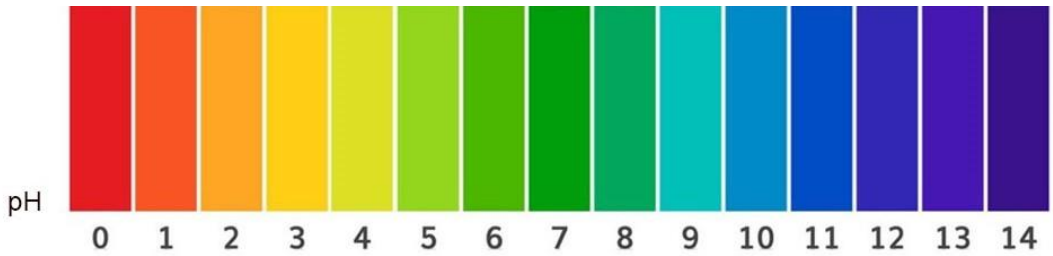


Рис 3.30 – Шкала індикації кислотності середовища

Завдяки тому, що в теплиці використовується аеропонний спосіб вирощування рослин кислотність середовища залежить від кислотності розчину, яким здійснюється зрошення кореневої системи рослин. Саме тому, вимірювання та контроль рівня pH здійснюється в ємності для зберігання рідини для зрошення рослин. Для цього в неї встановлений датчик

кислотності, а корегування рівня pH відбувається шляхом додавання концентрату кислот та лугів. Подача концентрату в ємність відбувається завдяки дозатору встановленому в верхній частині ємності.

При розробці моделі контуру на базі плати Arduino реалізація дозатора виконана за допомогою двох насосів RS-360SH. Принцип роботи такого дозатора полягає в подачі порції концентрату за визначений час. Тобто при зміні рівня pH в сторону збільшення кислотності насос 1 включається в роботу та працює напротязі визначеного часу за який лужний концентрат поступає в ємність впливаючи на значення кислотності. За таким же принципом працює насос 2, відмінний тільки концентрат, що поступає в ємність.

Індикація поточного рівня кислотності реалізована за допомогою світлової індикації. Якщо значення рівня pH знаходиться в заданому діапазоні про це буде сигналізувати зелений світлодіод. При отриманні від датчика кислотності значень вище від заданого діапазону це означатиме, що середовище стає біль лужним про що буде сповіщати блакитний світлодіод. При досягнені критичного рівня лужності спрацює синій світлодіод. При опусканні значень рівня pH нижче заданого діапазону означатиме, що середовище становиться більш кислим про що буде сигналізувати жовтий світлодіод діод. Про небезпечний рівень кислотності середовища буде сповіщати червоний світлодіод.

В даній схемі також відсутній ручний режим роботи плати при якому вимірюванні значення задаються та змінюються за допомогою потенціометра. Зміна рівня pH в ємності відбувається шляхом додавання води або концентрату кислот та лугів.

У ході роботи над контуром було створено блок схему, на якій показаний алгоритм роботи контуру (рис. 3.31). Блок схеми алгоритмів підпрограм, що використовуються в блок-схемі програми показані на рис 3.32 , рис 3.33, рис 3.34. Також зібрано модель за допомогою плати Arduino, схема підключень, якої показана на рис 3.35.

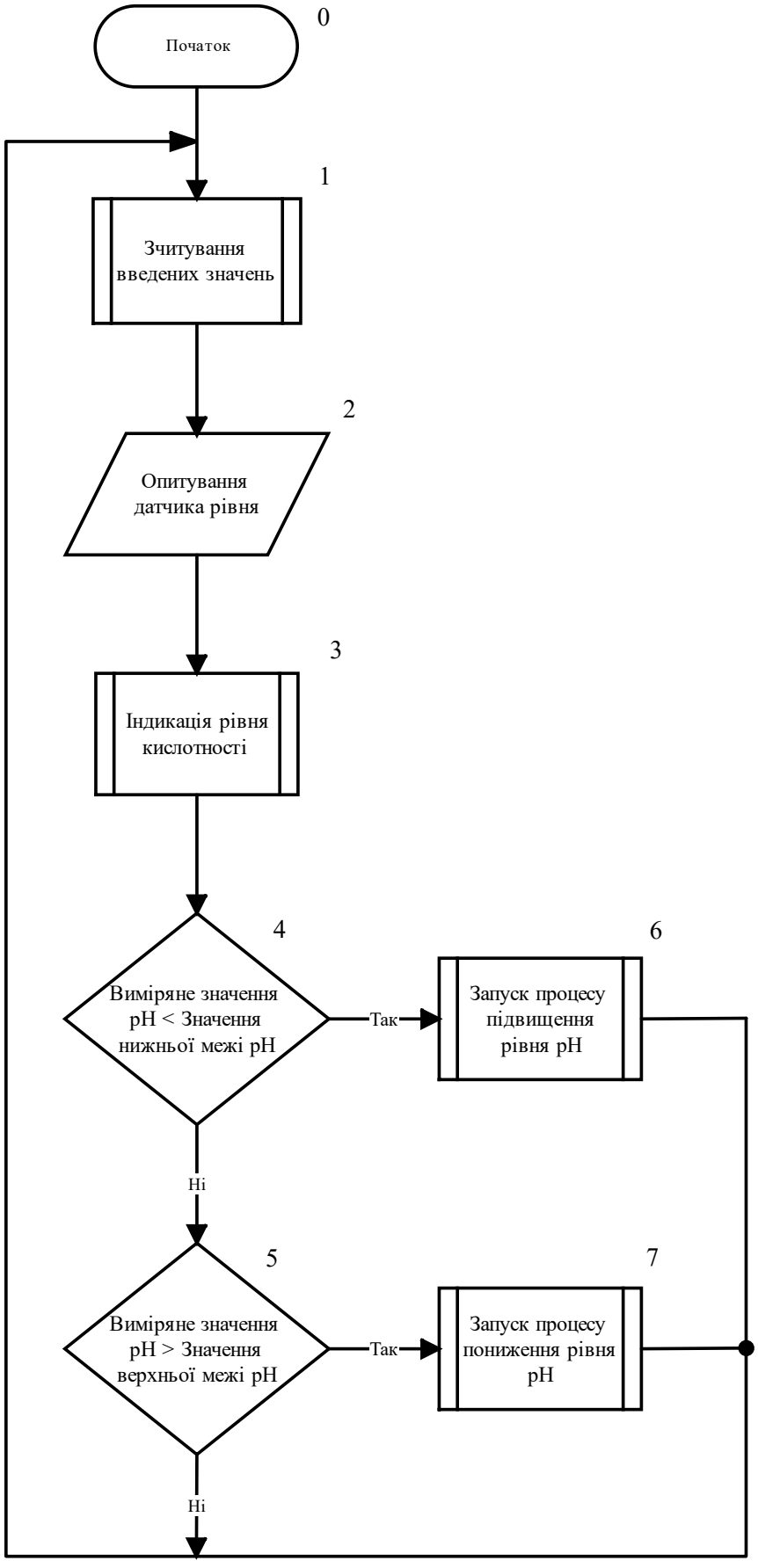


Рис 3.31 – Блок-схема роботи контуру підтримки рівня pH

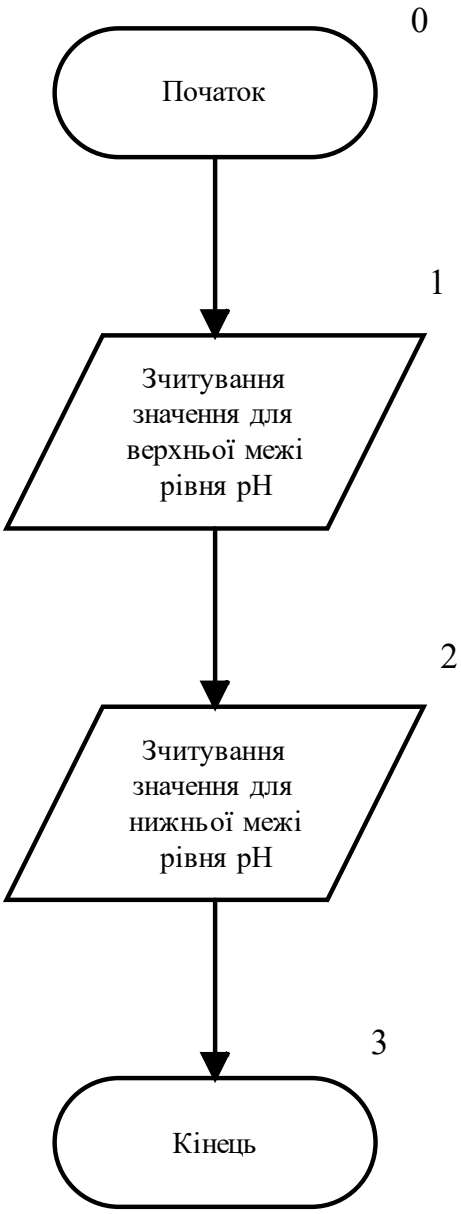


Рис 3.32 – Блок-схема алгоритму підпрограми “Зчитування заданих значень”

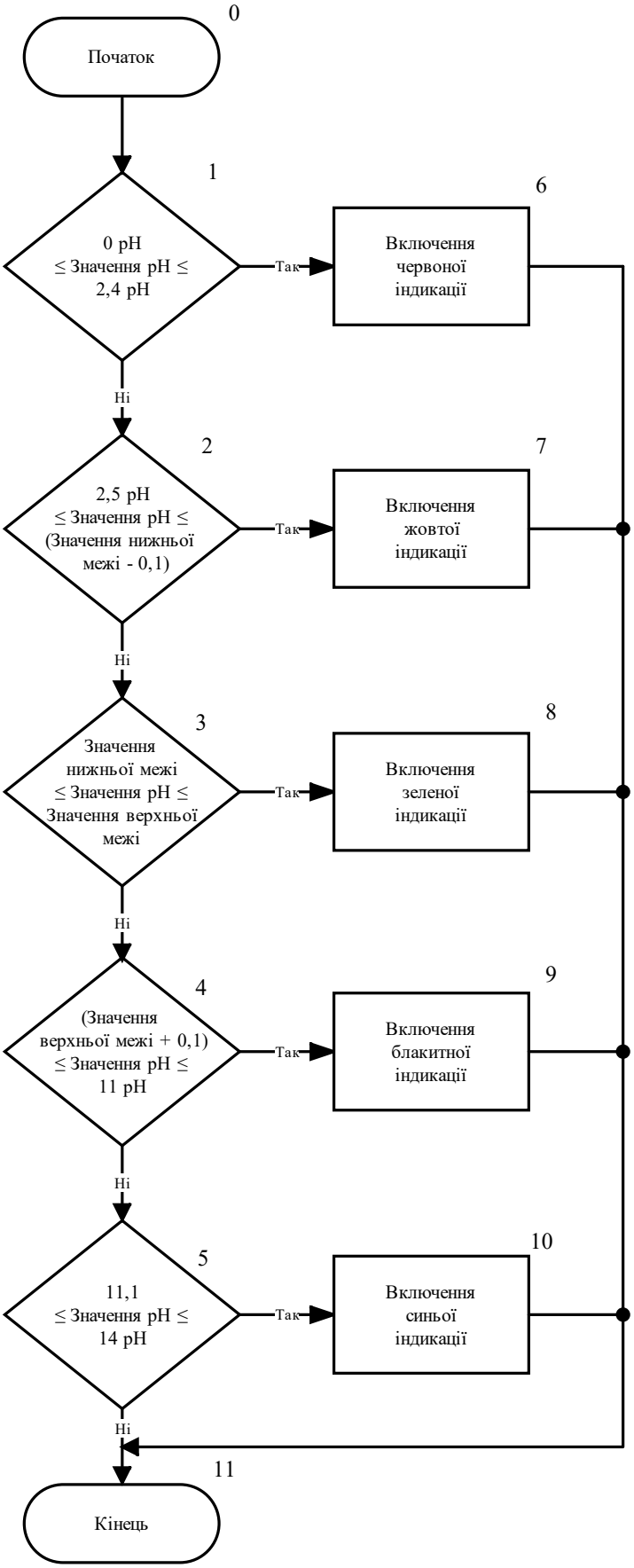
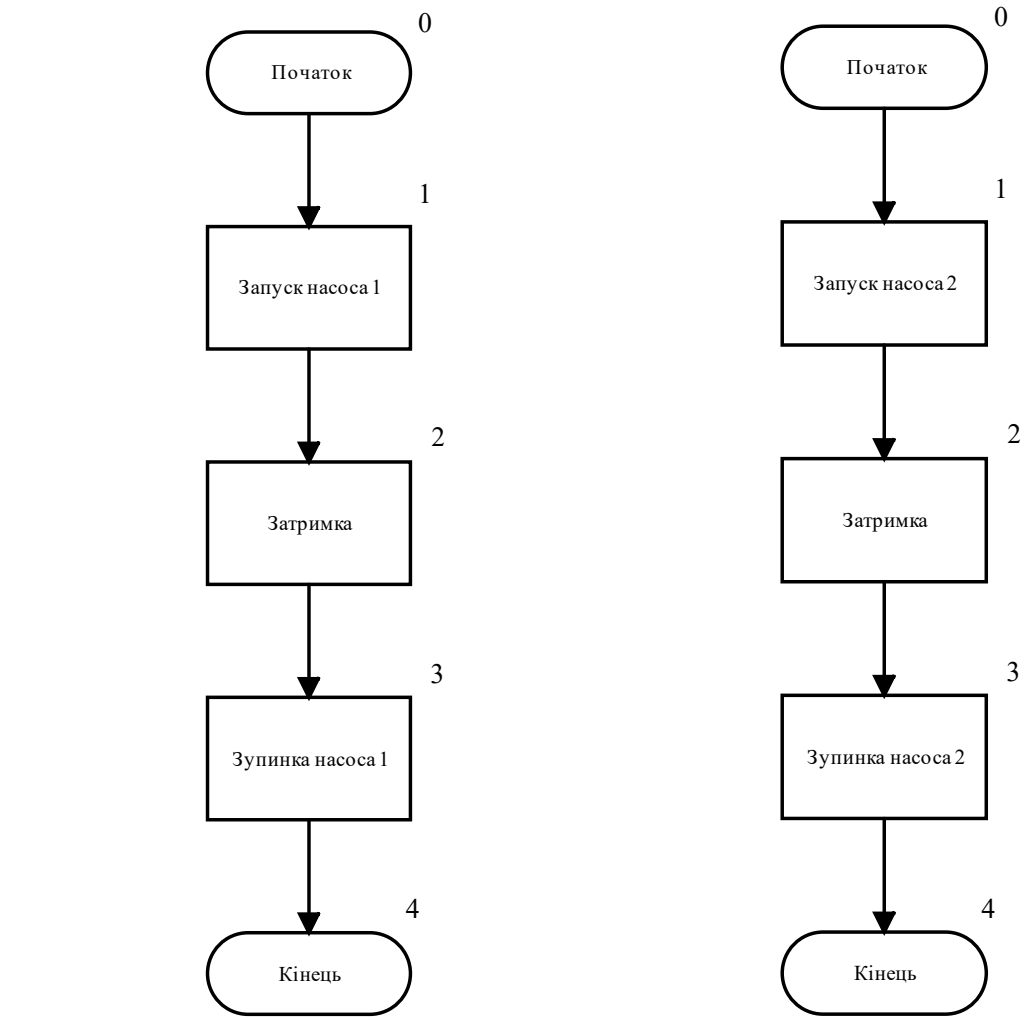


Рис 3.33 – Блок-схема алгоритму підпрограми

“Індикація поточного рівня pH”



а) б)

Рис 3.34 - Блок-схема алгоритму підпрограм:

а) “Процес підвищення рівня pH”; б) “Процес пониження рівня pH”

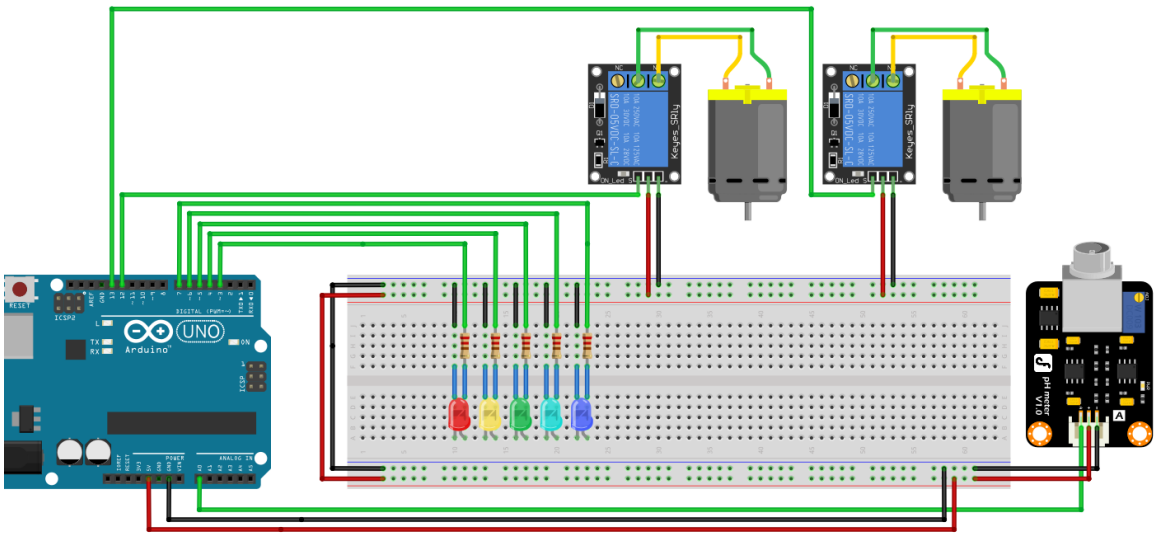


Рис 3.35 – Схема підключення компонентів до плати Arduino

На рис 3.36 показна схема підключення компонентів, які були використанні в для побудови моделі контору в підтримання заданого рівня кислотності. В схему входять аналоговий датчик, який підключений до аналогового входу 0, два насоси RS-360SH підключенні до цифрових виходів 13 та 12 та п’ять світлодіодів підключених до цифрових входів з 3 по 7.

Обрані насоси для коректної роботи потребують дещо більшого значення струму, ніж може видати вихід плати, отже для забезпечення їх робочим струмом було використано модуль реле, який може забезпечити струм комутації до 10А при напрузі 250В.

Маючи готовий алгоритм роботи та зібрану схему залишилось написати програму за якою буде працювати схема.

Першим кроком є опис задіяних виходів плати (рис 3.37) та опис змінних (рис 3.38), а також задання режимів роботи задіяних виходів плати.

```
const int pHSensor = 0;
const int redLed = 3;
const int yellowLed = 4;
const int greenLed = 5;
const int skyBlueLed = 6;
const int blueLed = 7;
const int DCMotor1 = 12;
const int DCMotor2 = 13;
```

Рис 3.37 – Опис виходів для плати Arduino

```
//Змінні
float currentpHLevel;
//Допустимий діапазон, pH
float highLine = 6.5;
float lowLine = 5.5;

boolean autoMode = true;
extern volatile unsigned long timer0_millis;
```

Рис 3.38 – Задання змінних для проекту

```
void setup() {
  Serial.begin(9600);
  pinMode(pHSensor, INPUT);
  pinMode(redLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(skyBlueLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  pinMode(DCMotor1, OUTPUT);
  pinMode(DCMotor2, OUTPUT);
}
```

Рис 3.39 – Налаштування режимів роботи описаних виходів плати

```
void loop() {
  if(autoMode == false) {
    //опитування датчика кислотності
    float pHSensorData = analogRead(pHSensor);
    currentpHLevel = map(pHSensorData, 0, 1023, 0, 14);
  } else if(autoMode == true){
      modulatepHBehavior();
  }

  //функція для вимикання індикації
  offIndicate();
  //функція для вимикання індикації поточного рівня
  pHLedIndication(currentpHLevel);

  if (currentpHLevel < lowLine) {
    //включення двигуна
    digitalWrite(DCMotor1, HIGH);
    //затримка
    delay(3000);
    //відключення двигуна
    digitalWrite(DCMotor1, LOW);
  }

  if(currentpHLevel > highLine){
    //включення двигуна
    digitalWrite(DCMotor2, HIGH);
    //затримка
    delay(3000);
    //відключення двигуна
    digitalWrite(DCMotor2, LOW);
  }

  Serial.println(currentpHLevel);
}
```

Рис 3.40 – Тіло програми роботи моделі контуру підтримки рівня кислотності

На рис 3.40 показано тіло програми за якою працює зібрана схема. За допомогою команди analogRead() відбувається зчитування інформації від датчика кислотності, значення якого знаходяться в діапазоні від 0 до 1023. Перетворення отриманого числа відбувається завдяки використанню команди map(), яка перетворює один діапазон в інший, в даному випадку перетворення здійснюється в рівень кислотності середовища, який знаходиться в діапазоні від 0 до 14 pH. Після отримання даних від датчика в залежності від умови виконується запуск одного з двох насосів, який працює протязі заданого періоду часу після спливу якого відключається. Час роботи насосу задається командою delay(), для даної схеми цей час становить 3 секунди. Запуск та зупинка роботи насосів відбуваються за допомогою команди digitalWrite().

Модулювання зміни вимірюваного значення в процесі роботи контуру відбувається завдяки функції modulatepHBehavior() (рис 3. 43). Зміна значення рівня кислотності є тривалим процесом та лінійним процесом тому період функції визначений в 5000 мілісекунд.

У ході роботи над функцією складений графік на якому показані, як значення кислотності змінюється в часі (рис 3.41). Також визначенні точки в яких значення рівень pH приймає найбільше на найменше значення.

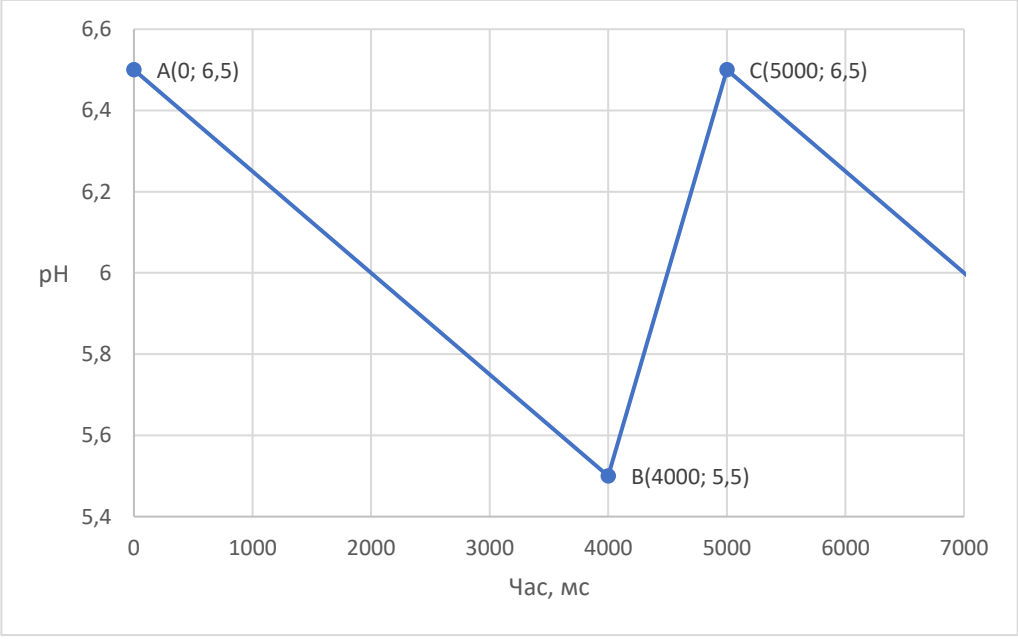


Рис 3.41 – Графік зміни значення pH в часі

Найбільше та найменше значення рівня кислотності приймає в точках 5,5 та 6,5, що є заданим діапазоном в якому підтримується рівень pH в поточному контурі. Для визначення проміжних значень рівня pH в конкретний момент часу, період розбито на дві прямі на якій значення спадає A(0; 6,5) B(4000; 5,5) та на якій зростає B(4000; 5,5) C(5000; 6,5).

За допомогою формули 3.1 визначено рівняння прямої для обох прямих.

Для прямої AB рівняння має наступний вигляд:

$$y_1 = \left(-\frac{1}{4000} \cdot x \right) + 6{,}5 \tag{3.4}$$

Для прямої BC рівняння має наступний вигляд:

$$y_2 = \frac{1}{1000} \cdot x + 1{,}5 \tag{3.5}$$

Для реалізації функції в час використана функція millis(), яка для даного контуру повертає значення від 0 до 5000 мілісекунд. Виходячи з графіка на рис 3.38 значення рівня кислотності в діапазоні від 0 до 4000 мілісекунд є спадаючим, а на проміжку від 4000 до 5000 мілісекунд зростає. Це означає, що для визначення проміжного значення рівня pH при значенні x часу, що належить першому спадаючому діапазону необхідно використати формулу 3.4. Для визначення проміжного значення з x часу, яка належить другому діапазону необхідно скористатись формулою 3.5.

Маючи інформацію про поведінку вимірювального значення рівня кислотності середовища було побудовано блок-схему алгоритму за яким працюватиме функція модулювання вимірювального значення. Блок-схема показана на рис. 3.39

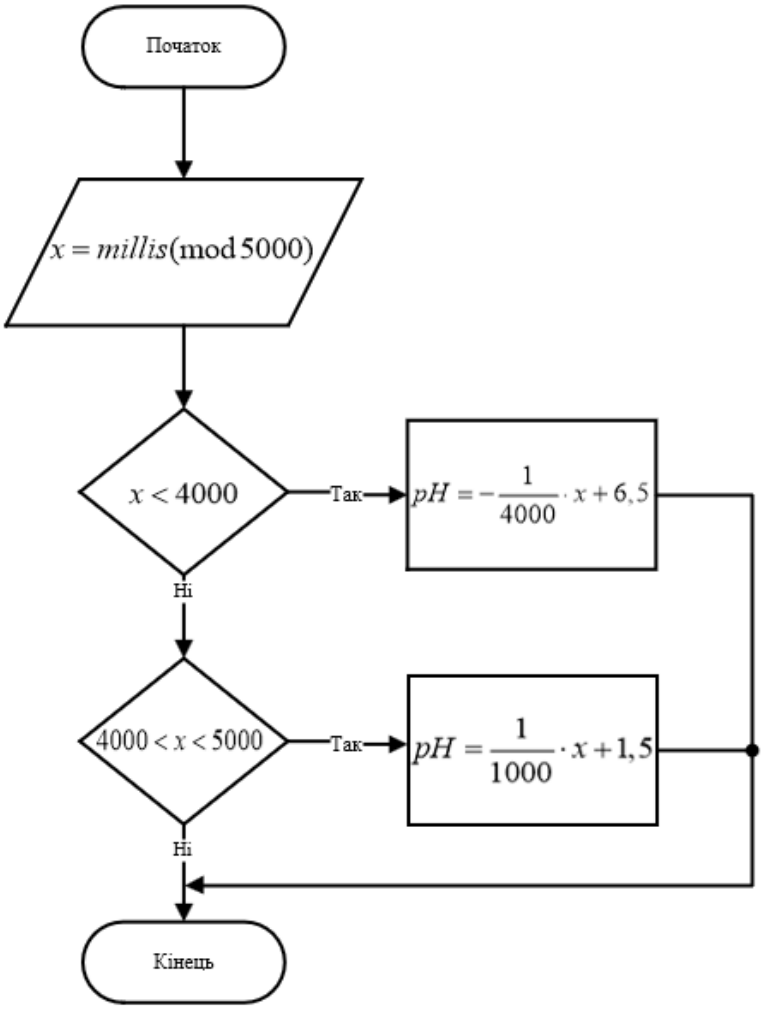


Рис 3.42 – Блок-схема алгоритму роботи функції модулювання

```
float modulatepHBehavior(){
  float x = millis();

  if(x < 4000){
    currentpHLevel = (-x/4000)+6.5;
  } else if(x > 4000 && x <= 5000){
    currentpHLevel = (x/1000)+1.5;
  }

  if(x >= 5000){
        noInterrupts ();
        timer0_millis = 0;
        interrupts ();
  }
}
```

Рис 3.43 – Функція modulatepHBehavior

Індикації поточного рівня pH відбувається завдяки двом функціям, які показані на рис 3.44

```
29  boolean offIndicate(){
30    digitalWrite(blueLed, LOW);
31    digitalWrite(greenLed, LOW);
32    digitalWrite(yellowLed, LOW);
33    digitalWrite(redLed, LOW);
34    digitalWrite(skyBlueLed, LOW);
35  }
36
37  boolean pHLedIndication(float pHLevel){
38    if(pHLevel >= 0 && pHLevel <= 2.4){
39      digitalWrite(redLed, HIGH);
40    } if(pHLevel >= 2.5 && pHLevel <= (lowLine - 0.1)) {
41      digitalWrite(yellowLed, HIGH);
42    } if(pHLevel >= lowLine && pHLevel <= highLine) {
43      digitalWrite(greenLed, HIGH);
44    } if(pHLevel >= (highLine + 0.1)  && pHLevel <= 11) {
45      digitalWrite(skyBlueLed, HIGH);
46    } if(pHLevel >= 11.1  && pHLevel <= 14) {
47      digitalWrite(blueLed, HIGH);
48    }
49  }
50
```

Рис 3.44 – Функції програми індикації поточного рівня pH

Функція pHLedIndication() повністю спирається блок-схему, що зображена на рис 3.33, де в залежності від значень поточного рівня кислотності виконується одна з заданих умов.

Функція offIndicate() необхідна для переведення світлодіодів в початковий стан. Це необхідно для коректного відображення поточного рівня за допомогою світлодіодів при переході від однієї вимірюваної зони ємності до іншої.

3.4 Розробка та побудова контуру підтримки заданого діапазону вологості середовища

Головною відмінністю аеропонного способу вирощування рослин від ґрунтового полягає в середовищі в якому знаходяться рослини та способу зрошення. В аеропоніці корені рослин вільно звисають в закритому повітряному середовищі, а процес поливу здійснюється через насичення повітряного середовища водною дисперсією, яка утворюється при проходженні рідини через форсунки. Беззаперечною перевагою такого способу зрошення в тому, що вологість розподіляється рівномірно окутуючи корені рослин.

Витрата водного ресурсу при такому способі поливу є дуже незначною в порівнянні з ґрунтовим вирощуванням. Додаткова економія також досягається завдяки тому що старт та тривалість процесу зрошення залежать від значення вологості середовища в якому знаходяться корені вирощуваних рослин. Особливістю такого туманного зрошення в тому що волога якою насичене повітря з часом опускається в низ та конденсує, тим самим значення рівня вологи зменшується. Недостатній рівень вологи може призвести до висихання коренів рослини та втрати частини врожаю. Тому метою розробки даного контуру є забезпечити рівень вологості в певному діапазоні значень.

Вимірювання поточного значення вологості в середовищі, де знаходиться коренева система рослин здійснюється завдяки встановленому датчику вологості. Діапазон вимірювання задається двома контрольним точками: нижньою та верхньою. При опусканні рівня вологості нижче значення нижньої контрольної точки відбувається запуск системи зрошення, який триває до поки рівень вологості не досягне значення верхньої контрольної точки.

Для індикації роботи режиму поливу обраний білий світлодіод, робота якого сповіщає про те що в даний момент відбувається процес зрошення. Робота зеленого світлодіоду означає про те що рівень вологості середовища знаходиться в значеннях заданого діапазону. Спрацювання жовтого світлодіоду буде сповіщати про те ще рівень вологості опустився нижче

значення нижньої межі заданого діапазону. Про небезпечно низький рівень вологості буде сповіщати комбінована сигналізація, яка складається з червоного світлодіоду та звукового п’єзо динаміка.

У ході роботи на розробкою контуру зрошення було складено блок-схему (рис 3.45), яка ілюструє алгоритм за яким працює контур. Блок-схеми підпрограм показані на рис 3.46, рис 3.47, рис 3.48.

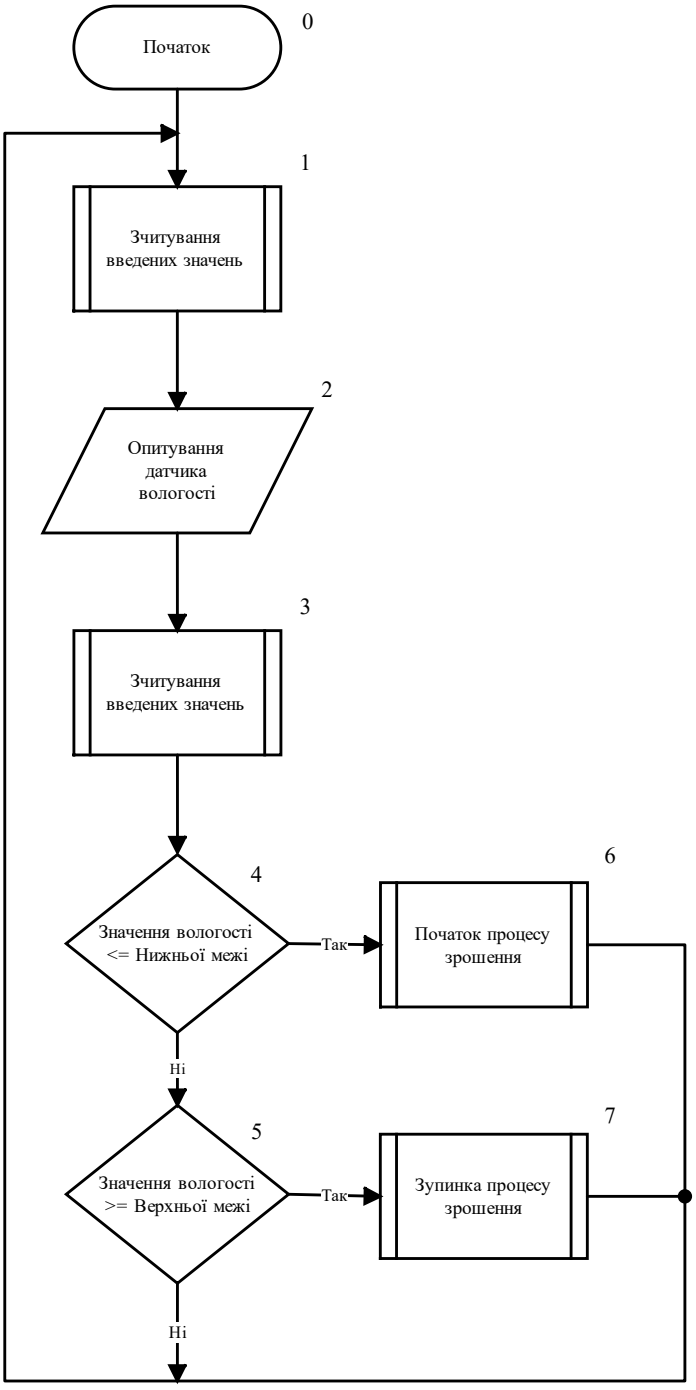


Рис 3.45 – Блок-схема роботи контуру зрошення

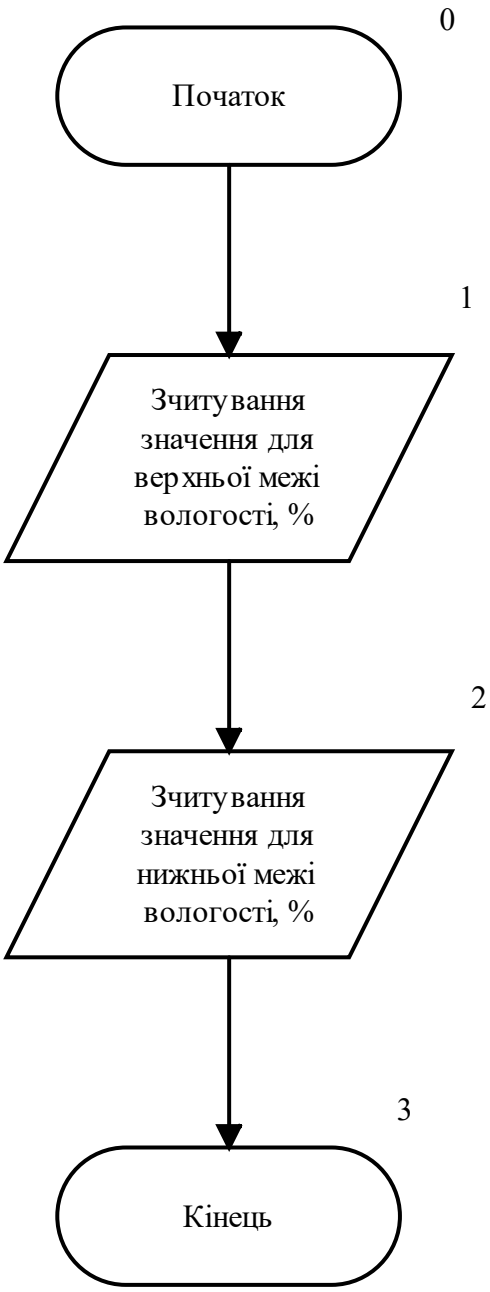


Рис 3.46 – Блок-схема алгоритму підпрограми “Зчитування заданих значень”

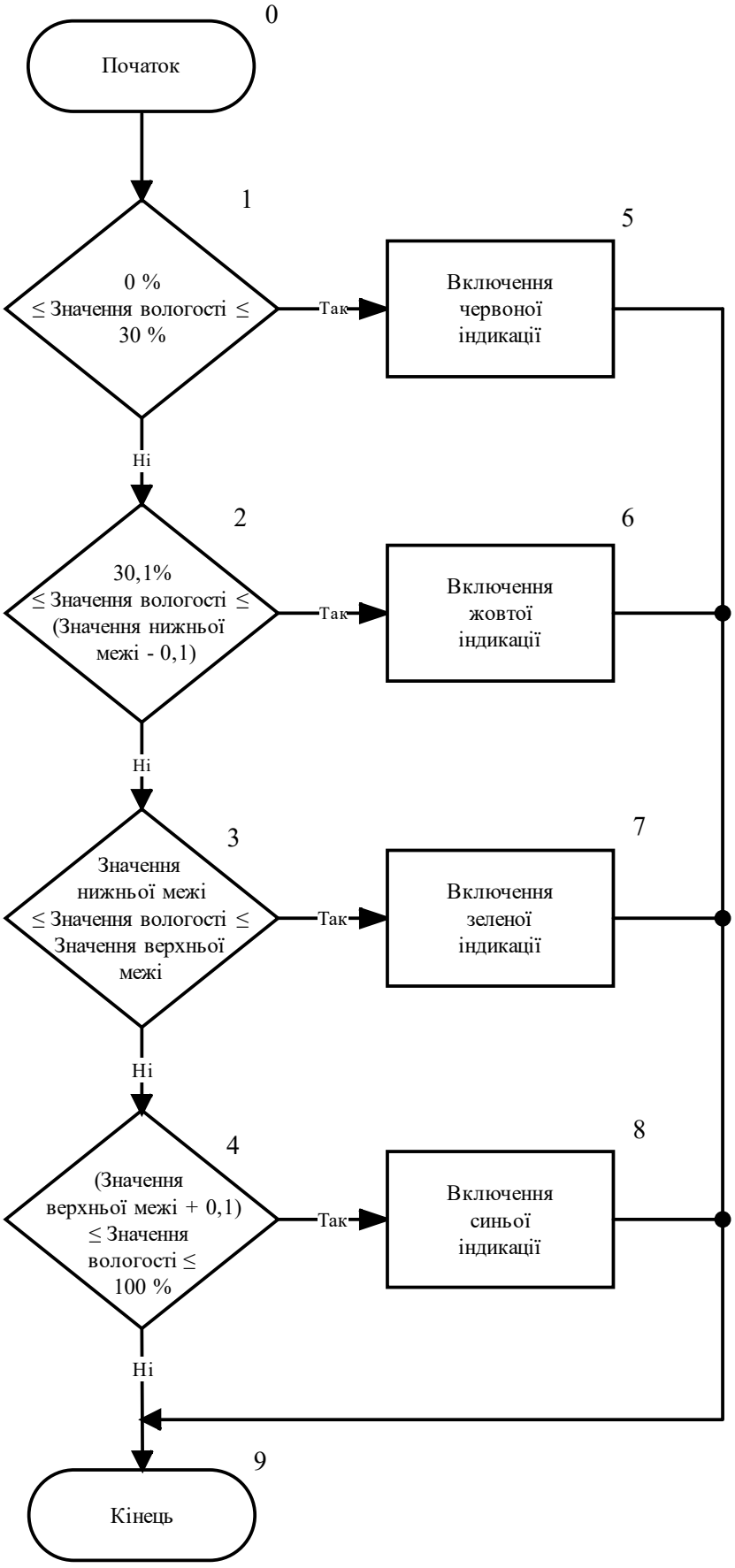
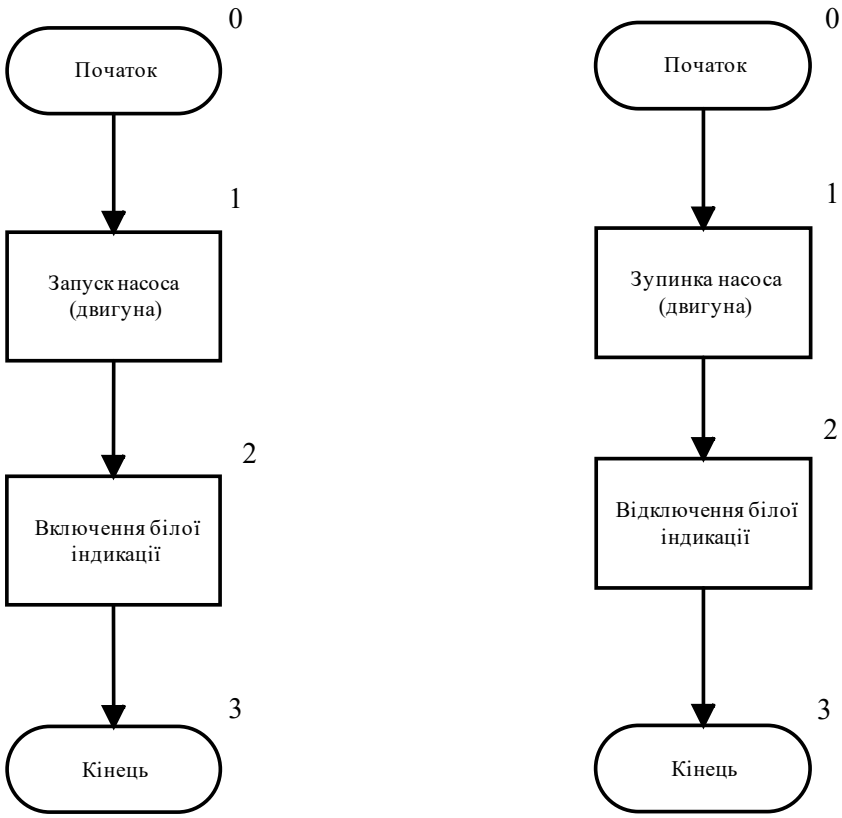


Рис 3.47 – Блок-схема алгоритму підпрограми

“Індикація поточного рівня ”



а) б)

Рис 3.48 – Блок-схема алгоритму підпрограми

а) “Початок процесу зрошення”; б) “Зупинка процесу зрошення”

За допомогою платформи Arduino зібрана модель розроблюваного контуру, схема підключень якого показана на рис 3.47.

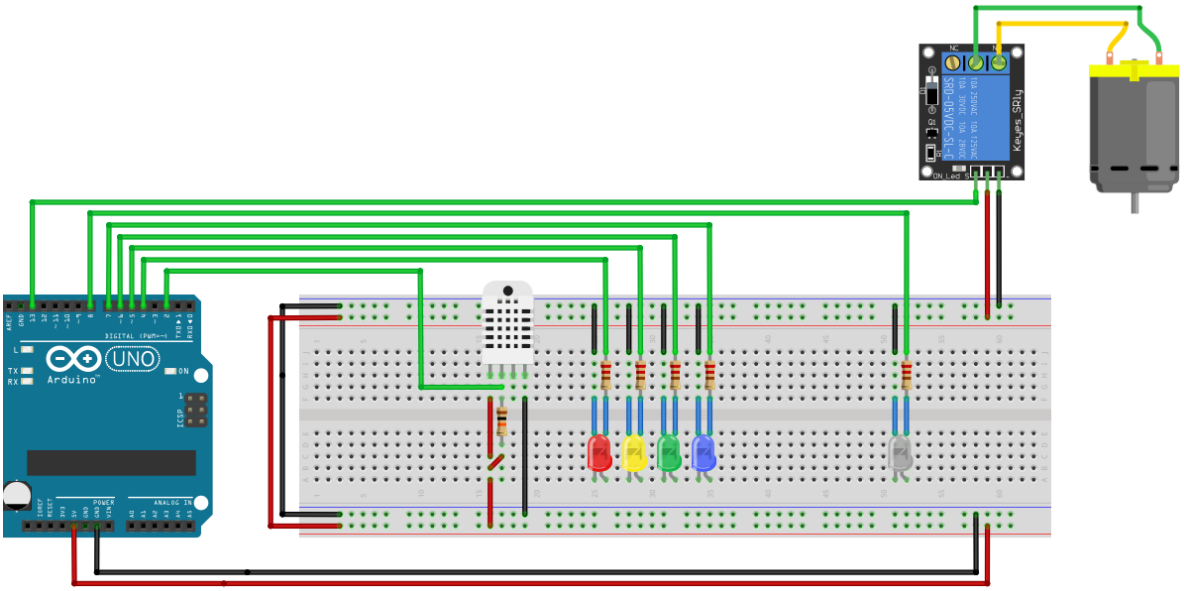


Рис 3.49 – Схема підключень компонентів до плати

Для побудови схеми було використано наступні компоненти: датчик вологості підключення, якого здійснюється до цифрового виходу плати 2, група світлодіодів, що призначена для індикації поточного рівня вологості підключено до виходів 4 – 7, світлодіод для індикації роботи насосу підключений до виходу 8, підключення самого насосу до виходу плати здійснюється через модуль реле, яке підключене до виходу плати.

Маючи розроблений алгоритм роботи та зібрану схему можна написати програмний код за яким схема буде працювати.

Першим чином в програмі описується задіяні входи плати (рис 3.50). Для підключення обраного датчика вологості DHT22 використовується бібліотека DHT sensor library в якій описані всі алгоритми та механізми роботи обраного датчика.

```
//Датчик вологості
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE)
//
const int potentiometer = 0;
const int btn = 3;
const int greenLed = 4;
const int whiteLed = 8;
const int DCMotor = 13;
```

Рис 3.50 – Опис виходів для плати Arduino

```
//Змінні
float currentHumidity;
//Верхня межа, %
float highLine = 98;
//Нижня межа, %
float lowLine = 60;

boolean autoMode = true;
boolean isPumpOn = false;
extern volatile unsigned long timer0_millis;
```

Рис 3.51 – Опис змінних для проекту

```
void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(redLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  pinMode(whiteLed, OUTPUT);
  pinMode(DCMotor, OUTPUT);
}
```

Рис 3.52 – Налаштування режимів роботи описаних виходів плати

```
void loop() {
  //Поточне значення
  if (autoMode == false) {
    currentHumidity = dht.readHumidity();
  } else if (autoMode == true) {
      modulateHumidityBehavior();
  }

  //Індикація
  offIndicate();
  humidityLedIndication(currentHumidity);

  if (currentHumidity <= lowLine) {
    //Змінна для старту процесу поливу
    isPumpOn = true;
  }

  if (currentHumidity >= highLine) {
    //Змінна для зупинки процесу поливу
    isPumpOn = false;
  }

  //Насос
  DCwork()

  //Вивід поточного значення на екран
  Serial.println(currentHumidity);
}
```

Рис 3.53 – Тіло програми

На рис 3.53 Показано тіло програми за якою працює зібрана схема. Програма має два режими роботи в залежності від того присутній датчик вимірювання вологості або ні. За замовчуванням датчик присутній, отже його опитування здійснюється за допомогою команди dht.readHumidity(). Данна команда повертає поточне значення вологості в відсотковому значенні, тому отримане значення одразу може використовуватись в подальшій програмі. Після отримання значень від датчика, яке задовольняє одну з двох умов відбувається зміна значення змінної isPumpOn, яка відповідає включення або відключення насосу в роботу. Запуск та зупинка роботи насосів відбуваються за допомогою функції DCwork() (рис 3.57) та безпосередньо команди digitalWrite().

Для модулювання зміни вимірювального значення вологості використовується функція modulateHumidityBehavior() (рис 3.56) В даному контурі вимірювальне значення є значенням вологості, яке змінюється лінійно, тому період даної функції складає 2000. Тому в ході роботи було побудовано графік (рис 3.54) на якому показано, як вимірювальне значення змінюється в часі, його найвищі та найнижчі значення, а також показаний період за який відбувається повний цикл роботи контуру.

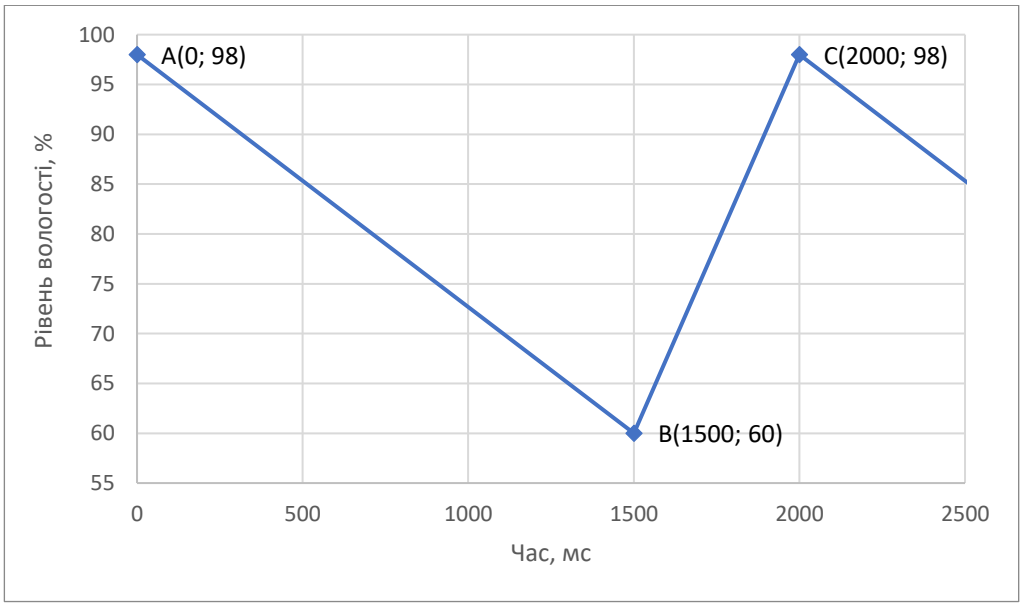


Рис 3.54 – Графік зміни вимірювального значення в часі

Значення періоду для даного контуру дорівнює 2000 мілісекунд. Цей період розбитий двома прямими A(0; 98) B(1500; 60) та B(1500; 60) C(2000; 98). Найвищим значенням рівня вологості є 98, а найнижчим 60 відсотків, що і є заданим діапазоном. Для визначення проміжного значення рівня в конкретний момент часу було використано рівняння прямої за формулою 3.1.

Для прямої AB рівняння має наступний вигляд:

$$y_1 = \left(-\frac{38}{1500} \cdot x \right) + 98 \tag{3.6}$$

Для прямої BC рівняння має наступний вигляд:

$$y_2 = \frac{38}{500} \cdot x - 54 \tag{3.7}$$

Зміна вологості є частим процесом тому значення періоду зміни вологості є найменшим з усіх розроблених контурів. Для слідкування за часом роботи контуру використана функція millis(), яка повертає значення для даного контуру від 0 до 2000 мілісекунд.

Маючи графік на якому показано поведінку вимірювального значення вологості та рівняння двох прямих на яких значення є спадаючим та зростаючим було складено блок-схему алгоритму за яким здійснюється обрахунок поточного значення в конкретним момент часу (рис 3.55).

Якщо значення часу x знаходиться в діапазоні від 0 до 1500 мілісекунд, то проміжне значення визначається за формулою 3.6. При отримані значення часу x, яке знаходиться в межах від 1500 до 2000 мілісекунд, поточне значення визначається за формулою 3.7.

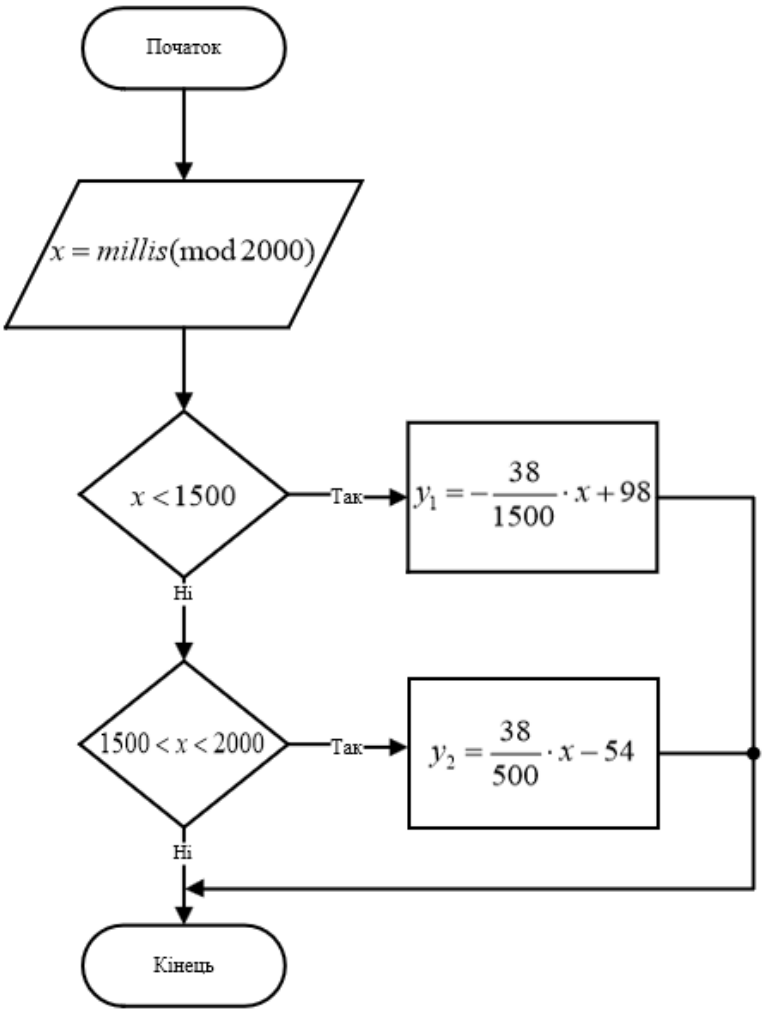


Рис 3.55 – Блок-схема алгоритму роботи функції модулювання

```
float modulateHumidityBehavior(){
  float x = millis();

  if(x < 1500){
    currentpHLevel = (-38*x/1500) + 98;
  } else if(x > 1500 && x <= 2000){
    currentpHLevel = (38*x/500)- 54;
  }

  if(x >= 2000){
        noInterrupts ();
        timer0_millis = 0;
        interrupts ();
  }
}
```

Рис 3.56 – Функція modulateHumidityBehavior

```
boolean DCwork (){
  if(isPumpOn == true){
    //включення двигуна
    digitalWrite(DCMotor, HIGH);
    //включення індикації роботи дивигуна
    digitalWrite(whiteLed, HIGH);
  } else if (isPumpOn == false){
    //відключення двигуна
    digitalWrite(DCMotor, LOW);
    //відключення індикації роботи дивигуна
    digitalWrite(whiteLed, LOW);
  }
}
```

Рис 3.57 – Функція керування роботою насосу DCwork

```
boolean offIndicate() {
  digitalWrite(blueLed, LOW);
  digitalWrite(greenLed, LOW);
  digitalWrite(yellowLed, LOW);
  digitalWrite(redLed, LOW);
}

boolean humidityLedIndication(float humidityLevel) {
  if (humidityLevel >= 0 && humidityLevel <= 30) {
    digitalWrite(redLed, HIGH);
  }
  if (humidityLevel >= 30.1 && humidityLevel <= (lowLine - 0.1)) {
    digitalWrite(yellowLed, HIGH);
  }
  if (humidityLevel >= lowLine && humidityLevel <= highLine) {
    digitalWrite(greenLed, HIGH);
  }
  if (humidityLevel >= highLine && humidityLevel <= 100) {
    digitalWrite(blueLed, HIGH);
  }
}
```

Рис 3.58 – Функції для індикації поточного рівня вологості

На рис 3.58 показні дві функції humidityLedIndication() та offIndicate(), що відповідають за роботу індикації поточного рівня вологості. Функція humidityLedIndication() отримує поточне значення рівня вологості та в залежності від нього спрацює один з світлодіодів.

Функція offIndicate() необхідна для відключення світлодіодів. Це необхідно для коректного відображення поточного рівня при переході від однієї вимірюваної зони ємності до іншої.

Висновки до розділу 3

1. Розроблено алгоритм роботи контуру клімат-контролю. За допомогою платформи Arduino зібрано модель розроблюваного контуру, а написаний програмний код роботи зібраної схеми.

2. Розроблено алгоритм роботи контуру слідкування за рівнем наповненості ємності. За допомогою платформи Arduino зібрано модель розроблюваного контуру, а також написаний програмний код за яким працює зібрана схема. А також розроблений алгоритм та написана програма, які імітують процес зміни рівня наповненості ємності.

3. Розроблено алгоритм роботи контуру підтримки заданого рівня кислотності в середовищі. За допомогою платформи Arduino зібрано модель розроблюваного контуру, а також написаний програмний код за яким працює зібрана схема. А також розроблений алгоритм та написана програма, які імітують процес зміни рівня кислотності середовища.

4. Розроблено алгоритм роботи контуру підтримки заданого рівня вологості в середовищі. За допомогою платформи Arduino зібрано модель розроблюваного контуру, та написаний програмний код роботи зібраної схеми. А також розроблений алгоритм та написана програма, які імітують процес зміни вологості середовища.

РОЗДІЛ 4. ПОБУДОВА МОДЕЛІ ТА МОЖЛИВОСТІ ЇЇ ВТІЛЕННЯ НА ПРАКТИЦІ

4.1 Реалізація побудованої моделі на мікропроцесорній платформі Arduino

Для розроблюваної моделі системи автоматичного керування теплицею з безґрунтовим способом вирощування рослин визначенні основні контури та параметри, що контролюються системою, для кожного контуру створений алгоритм його роботи, визначенні елементи схеми, які будуть задіяні при реалізації роботи контуру на базі плати Arduino та складена схема їх підключень до плати. Також для кожного контуру написаний програмний код, який дає можливість отримувати поточне значення вимірювального параметру, як від підключеного датчика, так і за допомогою функції моделювання процесу зміни вимірювального параметру в часі.

Наступним кроком в створенні розроблюваної моделі є об’єднання всіх розроблених контурів моделі в одну систему та тестування її роботи. Для створення такої схеми обрана плата Arduino Uno, яка використовувалась для побудови контурів системи, більше не підходить, тому що не забезпечує належну кількість виходів плати. Для її заміни обрана плата Arduino Mega (рис 4.1), яка має 53 цифрових виходів та 15 аналогових, що повністю задовольняють всі потреби.

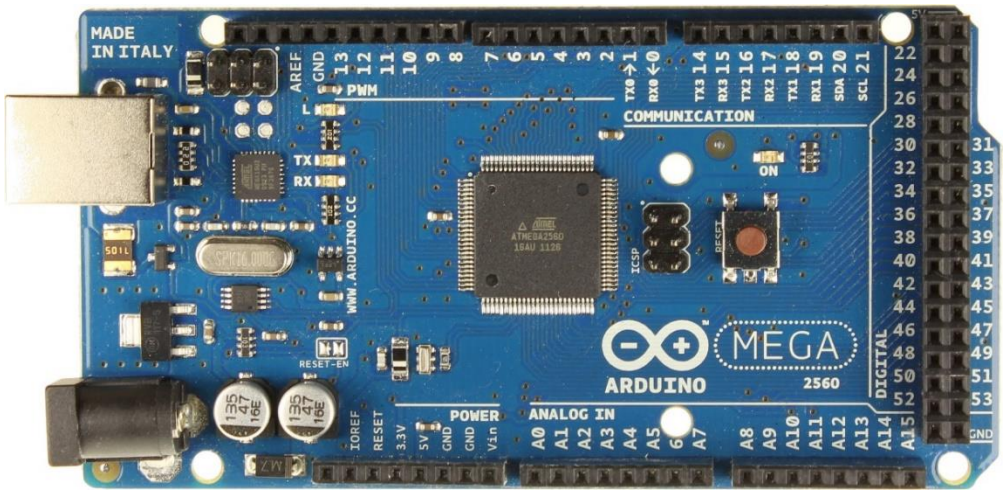


Рис 4.1 – Плата Arduino Mega

Для кожного контуру розроблена функція, яка імітує поведінку вимірювального значення в часі, також в схемах присутня світова індикація, яка забезпечує не тільки відображення поточного значення вимірювального параметру, але й дублює роботу виконавчих елементів. Це означає, що для побудови та тестування схеми розроблюваної системи можна скористатись тільки світлодіодами, робота, яких буде імітувати роботу контурів.

Для створення схеми підключення та імітації роботи отриманої схеми використано програмне забезпечення WOKWI, інструменти якого дозволяють забезпечити виконання поставлений задач.

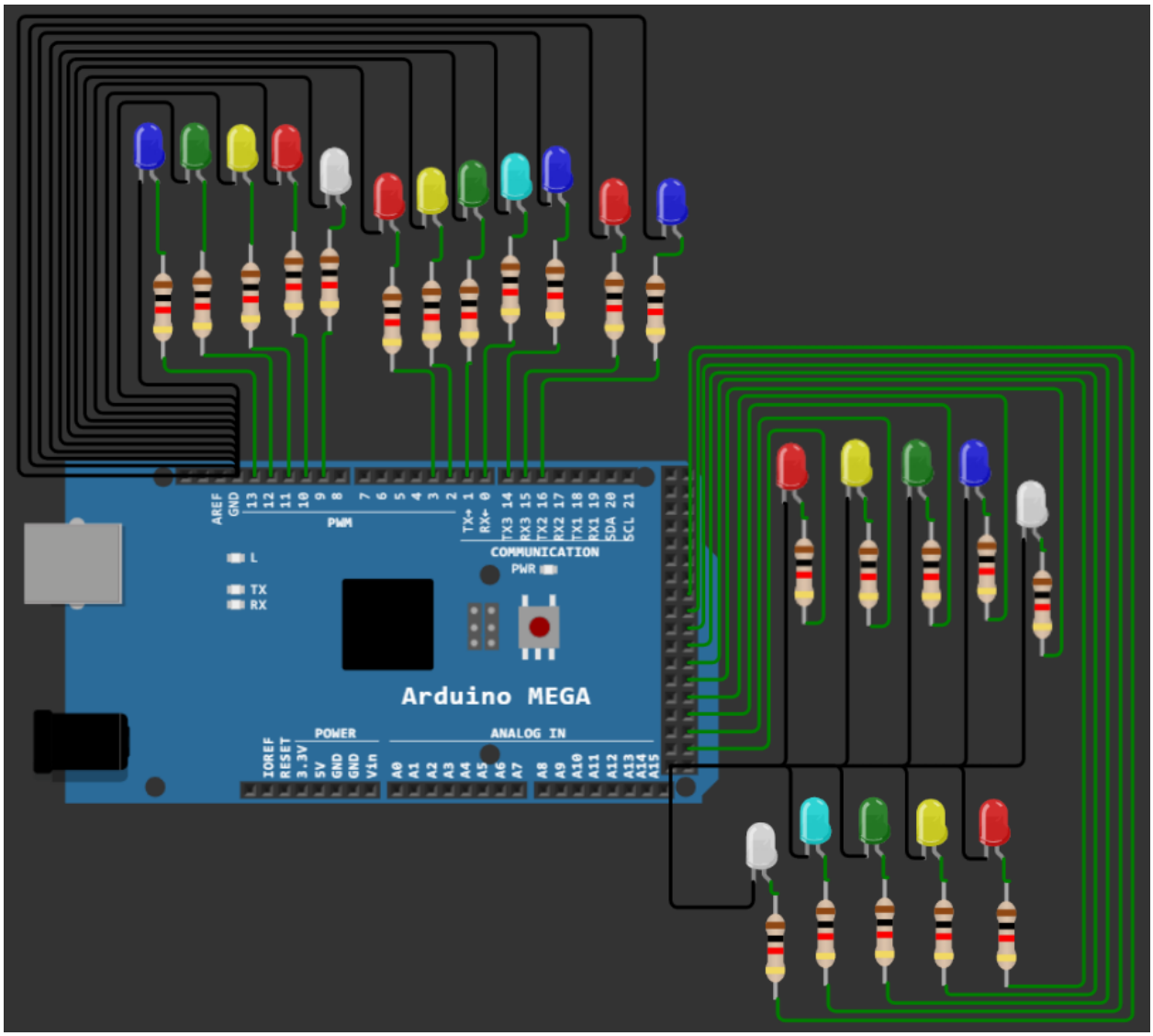
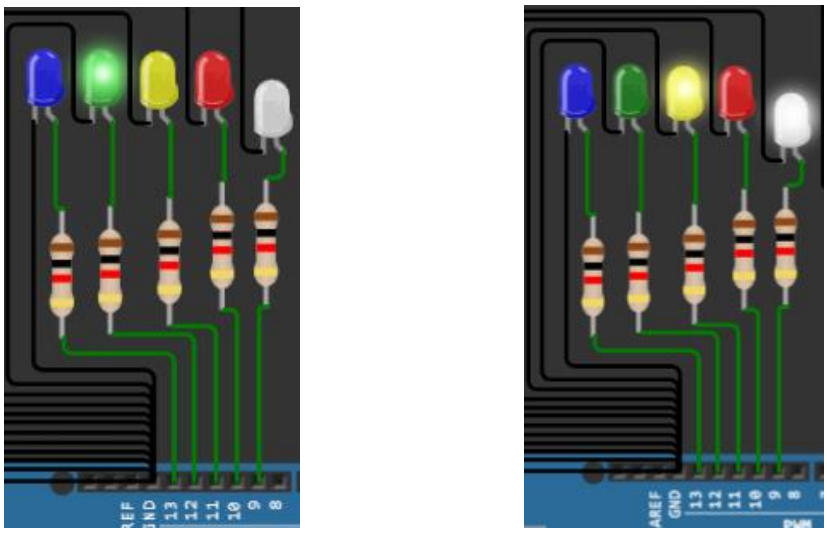


Рис 4.2 – Схема підключення компонентів до плати Arduino

На рис 4.2 показано схему підключення компонентів до плати Arduino. Дані компоненти, в основному світлодіоди, розташовані по групах для показу роботи розроблених контурів в одній системі. Кожна група має, що найменше один світлодіод, який розташований нижче за інших. Це розташування означає, що даний світлодіод призначений для індикації роботи виконавчого пристрою, наприклад насосу.

Група світлодіодів підключених до виходів з 9 по 13 призначенні для імітації роботи контуру підтримання заданого рівня наповненості ємності. Спрацювання зеленого світлодіоду вказує на те що поточний рівень наповненості знаходиться в заданому рівні (рис 4.3а). Спрацювання жовтого світлодіоду означає, що рівень наповненості опустився нижче половини об’єму ємності та є тригером для запуску в роботу системи наповнення ємності про що буде сповіщати робота білого світлодіоду (рис 4.3б).



а) б)

Рис 4.3 – Робота контуру підтримки рівня наповненості в імітаційному режимі роботи

Синій та червоний світлодіоди призначенні для індикації значення поточного рівня наповненості, яке знаходиться в небезпечних зонах. Робота червоного світлодіоду означає, що рівень в ємності є небезпечно низьким. Робота синього світлодіоду навпаки означає про небезпечний рівень наповненості.

Група світлодіодів, що імітують роботу контуру підтримання вологості підключенні до виходів плати під номерами 45, 47, 49, 51 та 53. Принцип роботи спрацювання світлодіодів є схожим до попереднього контуру. В даному робота зеленого сповіщає про те, що поточне значення вологості знаходиться заданому діапазоні та не потребує ввімкнення системи поливу для підвищення вологи в середовищі (рис 4.4).

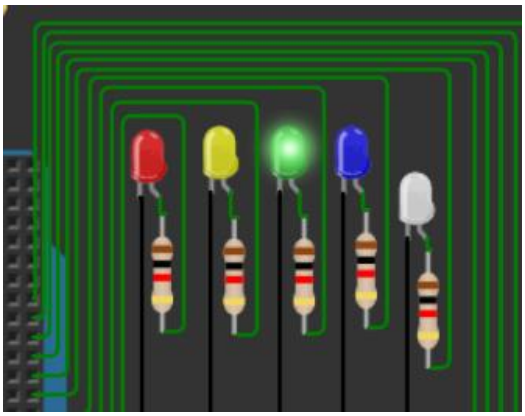
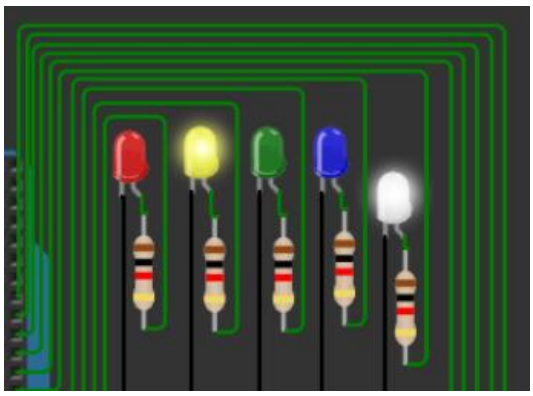
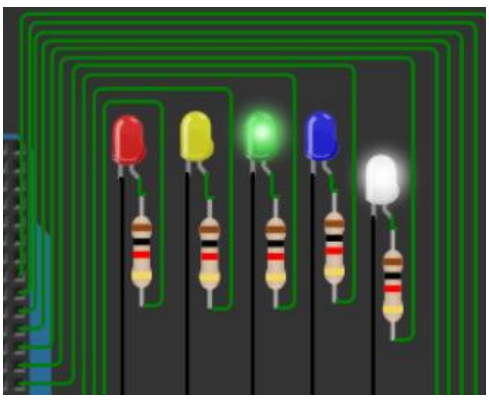


Рис 4.4 – Робота контуру підтримки вологості
в імітаційному режимі плати

Система зрошення буде автоматично включена в роботу при опусканні поточного рівня за встановлену нижню контрольну точку, про що буде сповіщати робота жовтого світлодіоду. Робота виконавчого елементу, яким забезпечується процес підняття вологості дублюється білим світлодіодом (рис 4.5).



а)



б)

Рис 4.5 – Робота контуру підтримки вологості
в імітаційному режимі плати

Процес підняття вологості є інертним процесом, а верхня контрольна точка по досяганні значення якої відбувається зупинка процесу поливу встановлена на рівні в 98 відсотків, тому поодиноке спрацювання синього діоду, робота якого сповіщає про те що рівень вологості є надмірним є передбачуваним (рис 4.6). На відміну від спрацювання червоного діоду, яке буде означати, що рівень вологості середовища знаходиться на небезпечно низькому рівні.

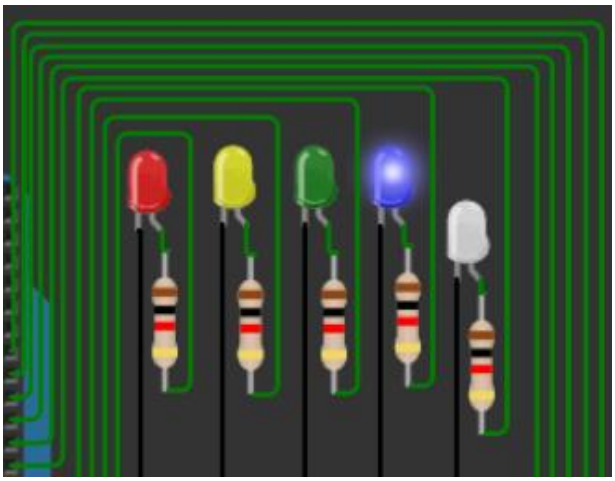


Рис 4.5 – Робота контуру підтримки вологості
в імітаційному режимі плати

Група світлодіодів, що імітують роботу контуру підтримання заданого рівня кислотності середовища підключенні до цифрових виходів плати під номерами 3, 2, 1, 0, 14, 15 та 16. В даному контурі для імітації роботи контуру використано дещо більше світлодіодів в порівняні з іншими, але головний принцип роботи залишається незмінним. Робота зеленого світлодіоду означає, що поточний рівень значення pH знаходиться в заданому діапазоні (рис 4.6).

Значення кислотності може коливатись, як в сторону кислотності, так і в сторону лужності. Жовтий світлодіод означає, що баланс змістився в стону кислотності, що призводить до запуску виконавчого елемента для вирівнювання рівня pH та спрацювання червоного світлодіоду (рис 4.7а). Блакитний діод означає, що середовище стало лужним та корекція середовища роботою синього світлодіоду (рис 4.7б).

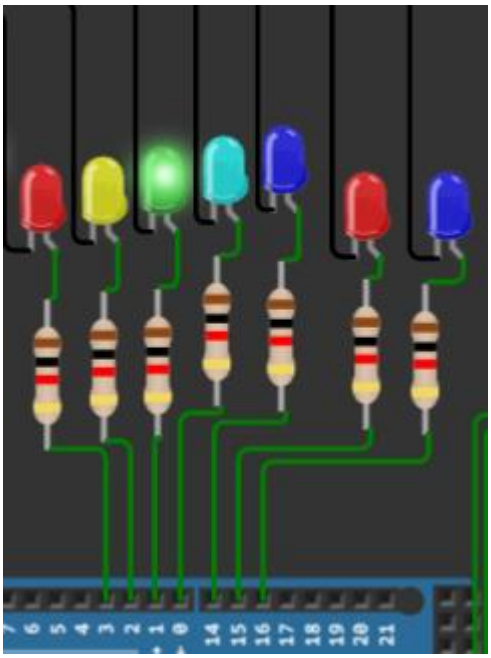
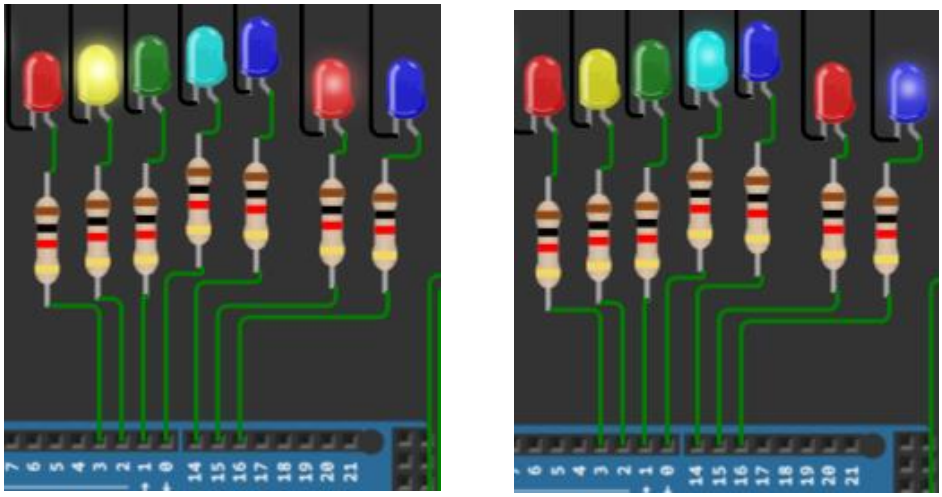


Рис 4.6 – Робота контуру підтримки рівня кислотності в імітаційному режимі плати



а) б)

Рис 4.7 – Робота контуру підтримки рівня кислотності в імітаційному режимі плати

Робота червоного та синього світлодіоду означають, що поточне значення рівня pH знаходиться в небезпечному рівні. Червоний діод сповіщає, що кислотність середовища є високою, а синій навпаки про те є середовище є надто лужним.

Останнім контуром є контур підтримання заданого значення температури в теплиці. Діоди, що імітують роботу даного контуру підключенні до наступних виходів плати: 35, 37, 39, 41, 43. Принцип роботи даного контуру залишається не зміним, а саме робота зеленого світло діоду сповіщає про те що вимірюване значення температури знаходиться в заданих межах (рис 4.8).

При опусканні значення температури нижче заданого діапазону спрацьовує блакитний світлодіод та автоматично вмикається виконавчий елемент, що впливає на вимірюване значення повертаючи його в задані межі. Робота виконавчого елементу дублюється білим світлодіодом (рис 4.9). Робота жовтого означає проте, що значення температури дещо вище за встановлені значення. Спрацювання червоного світлодіоду означає небезпечний рівень температури.

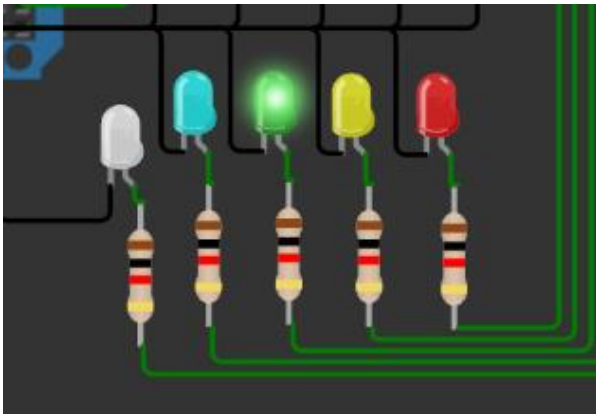


Рис 4.8 – Робота контуру клімат-контролю
в імітаційному режимі плати

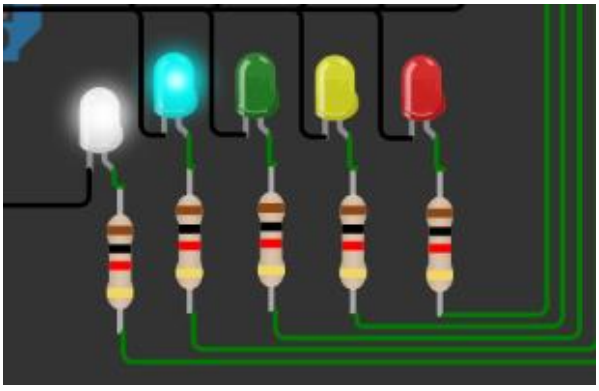


Рис 4.9 – Робота контуру клімат-контролю
в імітаційному режимі плати

4.2 Засоби, що дозволять створити систему автоматизації для безґрунтового вирощування врожаю в реальному житті.

Для реалізація розроблюваної автоматичної системи керування теплицею з безґрунтовим способом вирощування рослин в умовах наближених до промислових використання платформи Arduino, буде недоцільним. Адже при таких умовах є необхідність не тільки збору та обробки інформації від вимірювальних датчиків, але й забезпечувати керування великою кількістю потужних виконавчих елементів. Тому для керування такою системою найкращим вибором буде застосування повноцінного програмованого логічного контролера (ПЛК).

Перевагою ПЛК є те, що даний пристрів майже не потребує обслуговування, а налаштування здійснюється за допомогою персонального комп’ютера.

Проаналізувавши ринок контролерів за характеристиками та вартістю для побудови розроблюваної системи був обраний контролер фірми “ОВЕН” ПЛК160 (рис 4.10). Даний ПЛК обладнаний с дискретними та аналоговими входами/виходами та застосовується для побудови систем автоматичного керування.

Обраний контролер має наступні переваги та характеристики:

– наявність вбудованих дискретних входів/виходів на борту;

– наявність вбудованих аналогових входів/виходів;

– швидкісні входи для оброблення енкодерів;

– ведення архіву роботи, при підключенні до контролера usb-накопичувачів;

– просте та зручне програмування у системі codesys v. 2;

– передавання даних на верхній рівень через Ethernet або GSM-мережі;

– 3 послідовних портів (RS-232, RS 485);

– живлення: 220 В та 24 В.

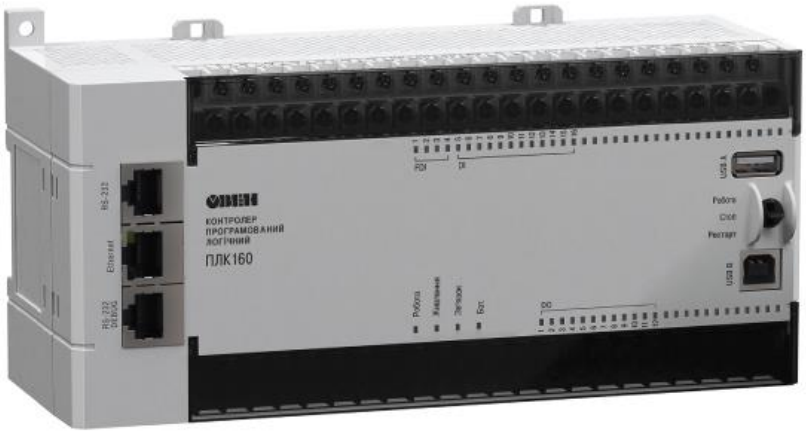


Рис 4.10 – Програмований логічний контролер ПЛК160

Вимірювальні прилади є також важливою частиною в реалізації розроблюваної системи автоматичного керування. Тому в додачу до обраного програмованого логічного контролера компанії “ОВЕН” також було розглянуто їх вимірювальну продукцію.

Компанія має широкий вибір датчиків для вимірювання температури, тому для розроблюваної системи обраний термометр опору ДТС034 з кабельним виводом (рис 4.11).

Рис 4.11 – Термоопір з кабельним виводом ДТС034

До характеристик даного датчику можна віднести:

– діапазон температур, що вимірюються: -50…+150°С;

– діапазон допустимих відхилень: ± 0,30 °С

– ступінь захисту: IP54;

Для вимірювання рівня вологості середовища обраний датчик волгості також від фірми “ОВЕН” під назвою ПВТ100 (рис 4.12). Даний датчик Здійснює безперервне перетворення відносної вологості робочого середовища в уніфіковані сигнали струму 4…20 мА. Які можуть передаватися через інтерфейс RS-485 за протоколом Modbus RTU.

Рис 4.12 – Датчик вологості ПВТ100

Для вимірювання значення поточного рівня заповненості ємності обраний ультразвуковий рівнемір компанії Vega під назвою VEGASON 62 (рис 4.13). Датчик має діапазон вимірювання від 0,4 до 8 м з точністю ±10мм, що повністю покриває необхідні потребності для розроблюваної системи.

Рис 4.13 – Ультразвуковий рівнемір VEGASON 62

Висновки до розділу 4

1. Зібрано повноцінну схему моделі розроблюваної системи автоматичного керування теплицею та проведено тестування її роботи в імітаційному режимі роботи плати Arduino за допомогою програмного забезпечення wokwi.

2. Проведений аналіз сучасних програмованих логічних контролерів та обраний підходящий для побудови розроблюваної системи. Проведений огляд вимірювальної продукції та обрані датчики, які можуть бути використані для побудови системи.

ЗАГАЛЬНІ ВИСНОВКИ

1. Проведено аналіз тепличного вирощування рослин. Розглянуто основні види теплиць, основні види конструкцій та способи вирощування рослин в них. Проаналізовано безґрунтові способи вирощування рослин їх види та описано їх переваги та недоліки. Визначені основні контури та параметри що будуть вимірюватися та контролюватися під час роботи системи керування.

2. В результаті проведеного аналізу сучасних платформ, що дозволять зібрати модель розроблюваної системи керування теплицею була обрана плата Arduino. Для кожного вимірювального параметру підібрано оптимальні датчики вимірювання для створення моделі на обраній платформі.

3. Для кожного контуру системи розроблено алгоритм його роботи. Розроблено схему підключення для кожного розроблюваного контуру, а також написаний програмний код за яким працює зібрані схеми. А також розроблений алгоритм та написана програми, які імітують процес зміни вимірювальної величини.

4. Розроблена імітаційна модель схеми роботи автоматичної системи керування телицею та протестована її робота. Розглянуті компоненти, які можуть бути використані при промисловій побудові системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Green Div. Greenhouse Gardening For Beginners / Green Div // Independently Published, 2021. - 5 с.
2. Jason Johns. Greenhouse Gardening - A Beginners Guide To Growing Fruit and Vegetables All Year Round / Jason Johns // Createspace Independent Publishing Platform // 2017. – 20 с.
3. Roger Marshall. The Greenhouse gardener`s manual / Roger Marshall // Timber Press, 2017. – 9 - 15 с.
4. Green Div. Hydroponics Growing System: A Complete guide for Beginners to build your own inexpensive Hydroponics system / Green Div // Charlie Creative Lab, 2021. – 30 с.
5. Технологія ярусного вирощування в теплиці. [Електронний ресурс]. – 2019. – Режим доступу: https://teplitca.kiev.ua/ua/a474653-tehnologiya-yarusnogo-vyraschivaniya.html
6. Lara Darling. Hydroponics, Aquaponics, Aeroponics (3 Books In 1) / Lara Darling // Amazon Digital Services LLC - KDP Print US, 2020 – 38 с.
7. М. Руденко. Чудесная гидропоника / М. Руденко // Vivat 2017. – 143 с.
8. Аеропоніка [Електронний ресурс]. – 2020 – Режим доступу: https://www.renovablesverdes.com/uk/aeroponia/
9. Обігрів теплиці [Електронний ресурс]. – 2021 – Режим доступу: https://plastok.com.ua/obihriv-teplitsi-v-zimovii-period-vidi-ta-osoblivosti
10. Освітлення в теплицях [Електронний ресурс]. – 2019 – Режим доступу: https://polikarbonatvs.com.ua/ua/articles/article9-htm/
11. Cerreto Rossouw. Automating Hydroponics: The Complete Guide To Food Security At Home / Cerreto Rossouw // CreateSpace 2018. – 64 с.
12. Thomas W. Gurley Aeroponics: Growing Vertical / Thomas W. Gurley // CRC Press, 2020. - 221 с.
13. Howard M. Resh Hydroponic Food Production / Howard M. Resh // CRC Press, 2022 - 39 с.

14. Ferdinand H. Quinones MD. Aeroponics: The Complete Guide About aeroponics / Ferdinand H. Quinones MD. Aeroponics // Independently published 2019 – 7 с.
15. Джон Хофман. Освоєння Arduino. Проектний підхід до електроніки, схемам та програмуванню / Д. Хофман // Самвидав 2018. - 11 с.
16. Документація для мікроконтролера ArduinoUno. [Електронний ресурс]. – Режим доступу: https://doc.arduino.ua/ru/hardware/Uno
17. Петин В. А., Биняковский А. А., Практическая энциклопедия Arduino / В. А. Петин, А. А Биняковский - изд. ДМК-Пресс, 2017 г.стр.158.
18. Майк МакРобертс. Основи Arduino / Майк МакРобертс // Самвидав 2018. – 15 с.
19. Белов А.В. Программирование ARDUINO. Создаем практические устройства / Белов А.В // Наука и Техника, 2018. – 45 с.
20. Гуржій А. М., Нельга А. Т., Співак В. М., Ітякін О. С. Основи автоматики та робототехніки / Гуржій А. М., Нельга А. Т., Співак В. М., Ітякін О. С. – Дніпро – Гарант СВ, 2021 – 61 с.
21. Датчики температури: термометр опору [Електронний ресурс] – 2020. – Режим доступу: https://corelamps.com/elektormontazhne-obladnannia/datchyk-temperatury/
22. Термопара та принципи її застосування [Електронний ресурс] – 2021. – Режим доступу: https://www.kvota.com.ua/statti/termopara/
23. Датчики вологості види та принцип роботи [Електронний ресурс] – 2019. Режим доступу: https://sitemasters.com.ua/elektroobladnannja/datchiki-vologosti-i-temperaturi-dlja/
24. Гідрометр для вимірювання вологості повітря [Електронний ресурс] – 2017. – Режим доступу: https://poradumo.com.ua/89808-kondensaciinii-gigrometr-gigrometr-dlia-vimiruvannia-vologosti-povitria/
25. pH-метр: призначення та принцип роботи [Електронний ресурс] – 2021. – Режим доступу: https://www.systopt.com.ua/article-rn-metr-naznachenye-y-pryncyp-raboty

26. pH-метр: робота, зберігання та каліброва [Електронний ресурс] – 2019. – Режим доступу: http://surl.li/dskga
27. Технологічні вимірювання і прилади. Вимірювання рівня та витрат / С. Г. Бондаренко, Д. М. Складанний, А. О. Абрамова // Київ: КПІ ім. Ігоря Сікорського, 2020. – 8 с.
28. Кондуктометричний метод вимірювання рівня [Електронний ресурс] – 2021. – Режим доступу: https://owen.ua/ua/news/konduktometrychnyj-metod-vymirjuvannja-rivnja
29. Акустичні рівнеміри [Електронний ресурс] – 2017. – Режим доступу: https://helpiks.org/6-52246.html
30. Шаховська Н.Б. , Голощук Р.О. Алгоритми та структури даних / Шаховська Н.Б. , Голощук Р.О. // Магнолія 2006 – 2021. – 15 – 33 с.
31. Дасгупта Санджой. Алгоритми / С. Дасгупта, У. Вазирани, Х. Пападимитриу // МЦНМО – 2019. – 25 – 32 с.
32. З. Шпак. Програмування мовою С /
33. О. Васильєв Програмування С++ в прикладах і задачах / О. Васильєв // Ліра-К – 2017. – 106 с.
34. Г. Галісєєв Системне програмування / Г. Галісєєв // Університет "Україна" -2019. – 11 – 45 с.

ДОДАТОК А

```
//Контур клімат-контролю
int currentTemperature;
//

const int temperatureSensor = 0;
const int potentiometer = 1;
const int btn = 2;
const int greenLed = 3;
const int blueLed = 4;
const int whiteLed = 5;
const int yellowLed = 6;
const int redLed = 7;
const int DCMotor = 12;
const int piezo = 13;

//Змінні
int userTemperature = 25;
int temperature;
//верхня межа
float highLimit = (userTemperature*150)/100;
//нижня межа
float lowLimit = (userTemperature*50)/100;
// вихід за дозволені межі
float highLevelOfAllowedDiapazon = (userTemperature*105)/100;
float lowLevelOfAllowedDiapazon = (userTemperature*95)/100;
//кнопка
boolean isGreenLedOn = LOW;
int buttonState;
int lastButtonState = LOW;
```

```
void setup() {
  Serial.begin(9600);
  pinMode(temperatureSensor, INPUT);
  pinMode(potentiometer, INPUT);
  pinMode(btn, INPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  pinMode(whiteLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(redLed, OUTPUT);
  pinMode(DCMotor, OUTPUT);
  pinMode(piezo, OUTPUT);
}

boolean debounce(int last){
  int reading = digitalRead(btn);

  if (reading != lastButtonState) {
    delay(100);
    reading = digitalRead(btn);
  }

  if (reading != buttonState) {
    buttonState = reading;
    if (buttonState == HIGH) {
      isGreenLedOn = !isGreenLedOn;
    }
  }
  digitalWrite(greenLed, isGreenLedOn);
```

```
  lastButtonState = reading;
  return isGreenLedOn;
}


void loop() {
  boolean isManualMode = debounce(lastButtonState);

  if(isManualMode == LOW){
    int sensorData = analogRead(temperatureSensor);
    float mV = (sensorData/1023.0)*5000;
    temperature = mV/30;
  } if(isManualMode == HIGH){
    float readPotentiometer = analogRead(potentiometer);
    temperature = map(readPotentiometer, 0, 1023, 0, 50);
  }

  if(temperature > userTemperature){
    digitalWrite(DCMotor, HIGH);
    //Відключення сигналізації про пониження температури
    digitalWrite(whiteLed, LOW);
    digitalWrite(blueLed, LOW);
    digitalWrite(piezo, LOW);
    //
    if(temperature > highLevelOfAllowedDiapazon){
      digitalWrite(yellowLed, HIGH);
    }
    if(temperature > highLimit){
      digitalWrite(redLed, HIGH);
      digitalWrite(piezo, HIGH);
```

```
    }
  }

  if(temperature < userTemperature){
    digitalWrite(DCMotor, LOW);
    //Відключення сигналізації про підвищення температури
    digitalWrite(yellowLed, LOW);
    digitalWrite(redLed, LOW);
    digitalWrite(piezo, LOW);
    //
    if(temperature < lowLevelOfAllowedDiapazon){
      digitalWrite(whiteLed, HIGH);
    }
    if(temperature < lowLimit){
      digitalWrite(blueLed, HIGH);
      digitalWrite(piezo, HIGH);
    }
  }

  delay(1000);
}
//рівень наповненості
const int waterLevelSensor = 0;
const int whiteLed = 3;
const int blueLed = 4;
const int greenLed = 5;
const int yellowLed = 6;
const int redLed = 7;
const int DCMotor = 13;
```

```
//Змінні
int currentWaterLevel;
//контрольна точка №1 встановлена на висоті в 90% від об`єму ємності
float controlPoint__1 = 90;
//контрольна точка №2 встановлена на висоті в 80% від об`єму ємності
float controlPoint__2 = 80;
//контрольна точка №3 встановлена на висоті в 50% від об`єму ємності
float controlPoint__3 = 50;
//контрольна точка №3 встановлена на висоті в 10% від об`єму ємності
float controlPoint__4 = 10;

boolean autoMode = true;
boolean isPumpOn = false;
extern volatile unsigned long timer0_millis;

void setup() {
  Serial.begin(9600);
  pinMode(waterLevelSensor, INPUT);
  pinMode(whiteLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(redLed, OUTPUT);
  pinMode(DCMotor, OUTPUT);
}

boolean offIndicate(){
  digitalWrite(blueLed, LOW);
  digitalWrite(greenLed, LOW);
  digitalWrite(yellowLed, LOW);
```

```
  digitalWrite(redLed, LOW);
}

boolean waterLevelLedIndication(int waterLevel){
  if(waterLevel <= 100 && waterLevel >= controlPoint__1){
    digitalWrite(blueLed, HIGH);
  } if(waterLevel <= controlPoint__1 && waterLevel >= controlPoint__3) {
    digitalWrite(greenLed, HIGH);
  } if(waterLevel <= controlPoint__3 && waterLevel >= controlPoint__4) {
    digitalWrite(yellowLed, HIGH);
  } if(waterLevel <= controlPoint__4 && waterLevel >= 0) {
    digitalWrite(redLed, HIGH);
  }
}

float modulateWaterLevelBehavior(){
  float x = millis();

  if(x < 2000){
    currentWaterLevel = (-30*x/2000) + 80;
  } else if(x > 2000 && x <= 3000){
    currentpHLevel = (30*x/1000) - 10;
  }

//обнулення millis
  if(x >= 3000){
        noInterrupts ();
        timer0_millis = 0;
        interrupts ();
  }
```

```
}

boolean DCwork (){
  if(isPumpOn == true){
    //включення двигуна
    digitalWrite(DCMotor, HIGH);
    //включення індикації роботи дивигуна
    digitalWrite(whiteLed, HIGH);
  } else if (isPumpOn == false){
    //відключення двигуна
    digitalWrite(DCMotor, LOW);
    //відключення індикації роботи дивигуна
    digitalWrite(whiteLed, LOW);
  }
}



void loop() {
  if (autoMode == false) {
    //опитування датчика рівня
    float waterLevelSensorData = analogRead(waterLevelSensor);
    currentWaterLevel = map(waterLevelSensorData, 0, 1023, 0, 100);
  } else if (autoMode == true) {
    //Модуляція зміни вимірюваного значення
      modulateWaterLevelBehavior();
    }

  //функція для вимикання індикації
  offIndicate();
  //функція для вимикання індикації поточного рівня
```

```
  waterLevelLedIndication(currentWaterLevel);

  if (currentWaterLevel <= controlPoint__3) {
    //Змінна для початку процесу поливу
    isPumpOn = true;
  }

  if(currentWaterLevel >= controlPoint__2){
    //Змінна для зупинки процесу поливу
    isPumpOn = false;
  }
  //Насос
  DCwork()
  Serial.println(currentWaterLevel);
}

//контур кислотності
const int pHSensor = 0;
const int redLed = 3;
const int yellowLed = 4;
const int greenLed = 5;
const int skyBlueLed = 6;
const int blueLed = 7;
const int DCMotor1 = 12;
const int DCMotor2 = 13;

//Змінні
float currentpHLevel;
//Допустимий діапазон, pH
float highLine = 6.5;
```

```
float lowLine = 5.5;

boolean autoMode = true;
extern volatile unsigned long timer0_millis;

void setup() {
  Serial.begin(9600);
  pinMode(pHSensor, INPUT);
  pinMode(redLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(skyBlueLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  pinMode(DCMotor1, OUTPUT);
  pinMode(DCMotor2, OUTPUT);
}

boolean offIndicate(){
  digitalWrite(blueLed, LOW);
  digitalWrite(greenLed, LOW);
  digitalWrite(yellowLed, LOW);
  digitalWrite(redLed, LOW);
  digitalWrite(skyBlueLed, LOW);
}

boolean pHLedIndication(float pHLevel){
  if(pHLevel >= 0 && pHLevel <= 2.4){
    digitalWrite(redLed, HIGH);
  } if(pHLevel >= 2.5 && pHLevel <= (lowLine - 0.1)) {
    digitalWrite(yellowLed, HIGH);
```

```
  } if(pHLevel >= lowLine && pHLevel <= highLine) {
    digitalWrite(greenLed, HIGH);
  } if(pHLevel >= (highLine + 0.1)  && pHLevel <= 11) {
    digitalWrite(skyBlueLed, HIGH);
  } if(pHLevel >= 11.1  && pHLevel <= 14) {
    digitalWrite(blueLed, HIGH);
  }
}

float modulatepHBehavior(){
  float x = millis();

  if(x < 4000){
    currentpHLevel = (-x/4000)+6.5;
  } else if(x > 4000 && x <= 5000){
    currentpHLevel = (x/1000)+1.5;
  }

  if(x >= 5000){
        noInterrupts ();
        timer0_millis = 0;
        interrupts ();
  }
}

void loop() {
  if(autoMode == false) {
    //опитування датчика кислотності
    float pHSensorData = analogRead(pHSensor);
    currentpHLevel = map(pHSensorData, 0, 1023, 0, 14);
```

```
  } else if(autoMode == true){
     modulatepHBehavior();
  }

  //функція для вимикання індикації
  offIndicate();
  //функція для вимикання індикації поточного рівня
  pHLedIndication(currentpHLevel);

  if (currentpHLevel < lowLine) {
    //включення двигуна
    digitalWrite(DCMotor1, HIGH);
    //затримка
    delay(3000);
    //відключення двигуна
    digitalWrite(DCMotor1, LOW);
  }

  if(currentpHLevel > highLine){
    //включення двигуна
    digitalWrite(DCMotor2, HIGH);
    //затримка
    delay(3000);
    //відключення двигуна
    digitalWrite(DCMotor2, LOW);
  }

  Serial.println(currentpHLevel);
}
```

```
//Контур вологості
//Датчик вологості
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
//
const int redLed = 4;
const int yellowLed = 5;
const int greenLed = 6;
const int blueLed = 6;
const int whiteLed = 8;
const int DCMotor = 13;

//Змінні
float currentHumidity;
//Верхня межа, %
float highLine = 98;
//Нижня межа, %
float lowLine = 60;

boolean autoMode = true;
boolean isPumpOn = false;
extern volatile unsigned long timer0_millis;

void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(redLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
```

```
  pinMode(greenLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  pinMode(whiteLed, OUTPUT);
  pinMode(DCMotor, OUTPUT);
}

boolean offIndicate() {
  digitalWrite(blueLed, LOW);
  digitalWrite(greenLed, LOW);
  digitalWrite(yellowLed, LOW);
  digitalWrite(redLed, LOW);
}

boolean humidityLedIndication(float humidityLevel) {
  if (humidityLevel >= 0 && humidityLevel <= 30) {
    digitalWrite(redLed, HIGH);
  }
  if (humidityLevel >= 30.1 && humidityLevel <= (lowLine - 0.1)) {
    digitalWrite(yellowLed, HIGH);
  }
  if (humidityLevel >= lowLine && humidityLevel <= highLine) {
    digitalWrite(greenLed, HIGH);
  }
  if (humidityLevel >= highLine && humidityLevel <= 100) {
    digitalWrite(blueLed, HIGH);
  }
}

float modulateHumidityBehavior(){
  float x = millis();
```

```
  if(x < 1500){
    currentpHLevel = (-38*x/1500) + 98;
  } else if(x > 1500 && x <= 2000){
    currentpHLevel = (38*x/500)- 54;
  }

  if(x >= 2000){
        noInterrupts ();
        timer0_millis = 0;
        interrupts ();
  }
}

boolean DCwork (){
  if(isPumpOn == true){
    //включення двигуна
    digitalWrite(DCMotor, HIGH);
    //включення індикації роботи дивигуна
    digitalWrite(whiteLed, HIGH);
  } else if (isPumpOn == false){
    //відключення двигуна
    digitalWrite(DCMotor, LOW);
    //відключення індикації роботи дивигуна
    digitalWrite(whiteLed, LOW);
  }
}

void loop() {
  //Поточне значення
```

```
  if (autoMode == false) {
    currentHumidity = dht.readHumidity();
  } else if (autoMode == true) {
      modulateHumidityBehavior();
  }

  //Індикація
  offIndicate();
  humidityLedIndication(currentHumidity);

  if (currentHumidity <= lowLine) {
    //Змінна для старту процесу поливу
    isPumpOn = true;
  }

  if (currentHumidity >= highLine) {
    //Змінна для зупинки процесу поливу
    isPumpOn = false;
  }

  //Насос
  DCwork()

  //Вивід поточного значення на екран
  Serial.println(currentHumidity);
}
```

ДОДАТОК Б

```
//Імітація роботи системи
//WaterLevel -- WL
const int WL_blueLed = 13;
const int WL_greenLed = 12;
const int WL_yellowLed = 11;
const int WL_redLed = 10;
const int WL_whiteLed = 9;

//Змінні
float currentWaterLevel;
//контрольна точка №1 встановлена на висоті в 90% від об`єму ємності
float controlPoint__1 = 90;
//контрольна точка №2 встановлена на висоті в 80% від об`єму ємності
float controlPoint__2 = 80;
//контрольна точка №3 встановлена на висоті в 50% від об`єму ємності
float controlPoint__3 = 55;
//контрольна точка №3 встановлена на висоті в 10% від об`єму ємності
float controlPoint__4 = 10;

boolean WL_isPumpOn = false;
float WL_y1;
float WL_y2;

//pH -- PH
const int PH_redLed = 3;
const int PH_yellowLed = 2;
const int PH_greenLed = 1;
const int PH_skyBlueLed = 0;
const int PH_blueLed = 14;
```

```
const int PH_IndicateredLed = 15;
const int PH_IndicateBlueLed = 16;

//змінні ph
//Змінні
float currentpHLevel;
//Допустимий діапазон, pH
float PH_highLine = 6.4;
float PH_lowLine = 5.7;
float PH_y1;
float PH_y2;

//humidity -- HUM
const int HUM_blueLed = 47;
const int HUM_greenLed = 49;
const int HUM_yellowLed = 51;
const int HUM_redLed = 53;
const int HUM_whiteLed = 45;
//Змінні
float currentHumidity;
//Верхня межа, %
float HUM_highLine = 95;
//Нижня межа, %
float HUM_lowLine = 65;

boolean HUM_isPumpOn = false;
float HUM_y1;
float HUM_y2;

//Temperature
```

```
const int TEM_SkyblueLed = 37;
const int TEM_greenLed = 39;
const int TEM_yellowLed = 41;
const int TEM_redLed = 43;
const int TEM_whiteLed = 35;
//Змінні
float currentTemperature;
//Верхня межа, %
float TEM_highLine = 27;
//Нижня межа, %
float TEM_lowLine = 25;

boolean TEM_isPumpOn = false;
float TEM_y1;
float TEM_y2;

void setup() {
  // wl
  pinMode(WL_blueLed, OUTPUT);
  pinMode(WL_greenLed, OUTPUT);
  pinMode(WL_yellowLed, OUTPUT);
  pinMode(WL_redLed, OUTPUT);
  pinMode(WL_whiteLed, OUTPUT);
  //ph
  pinMode(PH_redLed, OUTPUT);
  pinMode(PH_yellowLed, OUTPUT);
  pinMode(PH_greenLed, OUTPUT);
  pinMode(PH_skyBlueLed, OUTPUT);
  pinMode(PH_blueLed, OUTPUT);
  pinMode(PH_IndicateredLed, OUTPUT);
```

```
  pinMode(PH_IndicateBlueLed, OUTPUT);
  //humidity
  pinMode(HUM_redLed, OUTPUT);
  pinMode(HUM_yellowLed, OUTPUT);
  pinMode(HUM_greenLed, OUTPUT);
  pinMode(HUM_blueLed, OUTPUT);
  pinMode(HUM_whiteLed, OUTPUT);
  //Tem
  pinMode(TEM_SkyblueLed, OUTPUT);
  pinMode(TEM_greenLed, OUTPUT);
  pinMode(TEM_yellowLed, OUTPUT);
  pinMode(TEM_redLed, OUTPUT);
  pinMode(TEM_whiteLed, OUTPUT);
  //Serial.begin(9600);
}

//temperature
boolean temperatureLedIndication(int temperature){
  if(temperature <= TEM_lowLine){
    digitalWrite(TEM_SkyblueLed, HIGH);
  } if(temperature >= TEM_highLine){
    digitalWrite(TEM_yellowLed, HIGH);
  } if(temperature > TEM_lowLine && temperature < TEM_highLine){
    digitalWrite(TEM_greenLed, HIGH);
  }
}
boolean TEM_offIndicate(){
  digitalWrite(TEM_SkyblueLed, LOW);
  digitalWrite(TEM_greenLed, LOW);
  digitalWrite(TEM_yellowLed, LOW);
```

```
  digitalWrite(TEM_redLed, LOW);
}
boolean TEM_DCwork (){
  if(TEM_isPumpOn == true){
    //включення індикації роботи дивигуна
    digitalWrite(TEM_whiteLed, HIGH);
  } else if (TEM_isPumpOn == false){
    //відключення індикації роботи дивигуна
    digitalWrite(TEM_whiteLed, LOW);
  }
}
float modulateTemperatureBehavior(){
  float x = millis();
  TEM_y1 = x - WL_y2 ;
  if(TEM_y1 < 2000){
    currentTemperature = (-1*TEM_y1/2000) + 27;
  } else if(TEM_y1 > 2000 && TEM_y1 <= 3000){
    currentTemperature = (-2*TEM_y1/1000) + 21;
  }

  //обнулення
  if(TEM_y1 >= 3000){
    TEM_y2 = TEM_y1 + TEM_y2;
    TEM_y1 = 0;
  }
}

int Temperaturecontur(){
    //моделювання контуру рівня
  modulateTemperatureBehavior();
```

```
  //функція для вимикання індикації
  TEM_offIndicate();
  //функція для вимикання індикації поточного рівня
  temperatureLedIndication(currentTemperature);
  if (currentTemperature <= TEM_lowLine) {
    //Змінна для початку процесу поливу
    TEM_isPumpOn = true;
  }

  if(currentTemperature >= TEM_highLine){
    //Змінна для зупинки процесу поливу
    TEM_isPumpOn = false;
  }
  TEM_DCwork();
}

//wl
boolean waterLevelLedIndication(int waterLevel){
  if(waterLevel <= 100 && waterLevel > controlPoint__1){
    digitalWrite(WL_blueLed, HIGH);
  } if(waterLevel <= controlPoint__1 && waterLevel >= controlPoint__3) {
    digitalWrite(WL_greenLed, HIGH);
  } if(waterLevel <= controlPoint__3 && waterLevel >= controlPoint__4) {
    digitalWrite(WL_yellowLed, HIGH);
  } if(waterLevel <= controlPoint__4 && waterLevel >= 0) {
    digitalWrite(WL_redLed, HIGH);
  }
}

boolean WL_offIndicate(){
```

```
  digitalWrite(WL_blueLed, LOW);
  digitalWrite(WL_greenLed, LOW);
  digitalWrite(WL_yellowLed, LOW);
  digitalWrite(WL_redLed, LOW);
}

boolean WL_DCwork (){
  if(WL_isPumpOn == true){
    //включення індикації роботи дивигуна
    digitalWrite(WL_whiteLed, HIGH);
  } else if (WL_isPumpOn == false){
    //відключення індикації роботи дивигуна
    digitalWrite(WL_whiteLed, LOW);
  }
}

float modulateWaterLevelBehavior(){
  float x = millis();
  WL_y1 = x - WL_y2 ;
  if(WL_y1 < 2000){
    currentWaterLevel = (-30*WL_y1/2000) + 80;
  } else if(WL_y1 > 2000 && WL_y1 <= 3000){
    currentWaterLevel = (30*WL_y1/1000) - 10;
  }

  //обнулення millis
  if(WL_y1 >= 3000){
    WL_y2 = WL_y1 + WL_y2;
    WL_y1 = 0;
  }
```

```
}

int WaterLevelcontur(){
   //моделювання контуру рівня
  modulateWaterLevelBehavior();
  //функція для вимикання індикації
  WL_offIndicate();
  //функція для вимикання індикації поточного рівня
  waterLevelLedIndication(currentWaterLevel);
  if (currentWaterLevel <= controlPoint__3) {
    //Змінна для початку процесу поливу
    WL_isPumpOn = true;
  }

  if(currentWaterLevel >= controlPoint__2){
    //Змінна для зупинки процесу поливу
    WL_isPumpOn = false;
  }
  WL_DCwork();
}

//pH
boolean PH_offIndicate(){
  digitalWrite(PH_redLed, LOW);
  digitalWrite(PH_greenLed, LOW);
  digitalWrite(PH_yellowLed, LOW);
  digitalWrite(PH_blueLed, LOW);
  digitalWrite(PH_skyBlueLed, LOW);
}
boolean pHLedIndication(float pHLevel){
```

```
  if(pHLevel >= 0 && pHLevel <= 2.4){
    digitalWrite(PH_redLed, HIGH);
  } if(pHLevel >= 2.5 && pHLevel <= (PH_lowLine - 0.1)) {
    digitalWrite(PH_yellowLed, HIGH);
  } if(pHLevel >= PH_lowLine && pHLevel <= PH_highLine) {
    digitalWrite(PH_greenLed, HIGH);
  } if(pHLevel >= (PH_highLine + 0.1)  && pHLevel <= 11) {
    digitalWrite(PH_skyBlueLed, HIGH);
  } if(pHLevel >= 11.1  && pHLevel <= 14) {
    digitalWrite(PH_blueLed, HIGH);
  }
}
float modulatepHBehavior(){
  long x = millis();
  PH_y1 = x - PH_y2;

  if(PH_y1 < 4000){
    currentpHLevel = (-PH_y1/4000)+6.5;
  } else if(PH_y1 > 4000 && PH_y1 <= 5000){
    currentpHLevel = (PH_y1/1000)+1.5;
  }

  if(PH_y1 >= 5000){
    PH_y2 = PH_y2 + PH_y1;
    PH_y1 = 0;
  }
}
int pHcontur() {
  modulatepHBehavior();
  PH_offIndicate();
```

```
  pHLedIndication(currentpHLevel);
  if (currentpHLevel < PH_lowLine) {
     //включення двигуна
     digitalWrite(PH_IndicateredLed, HIGH);
     //відключення двигуна
     digitalWrite(PH_IndicateredLed, LOW);
    }

  if(currentpHLevel > PH_highLine){
     //включення двигуна
     digitalWrite(PH_IndicateBlueLed, HIGH);
     //відключення двигуна
     digitalWrite(PH_IndicateBlueLed, LOW);
    }
}

//humidity
float modulateHumidityBehavior(){
  long x = millis();
  HUM_y1 = x - HUM_y2;
  if(HUM_y1 < 1500){
    currentHumidity = (-38*HUM_y1/1500) + 98;
  } else if(HUM_y1 > 1500 && HUM_y1 <= 2000){
    currentHumidity = (38*HUM_y1/500)- 54;
  }

  if(HUM_y1 >= 2000){
    HUM_y2 = HUM_y2 + HUM_y1;
    HUM_y1 = 0;
  }
```

```
}

boolean HUM_DCwork(){
  if(HUM_isPumpOn == true){
    //включення індикації роботи дивигуна
    digitalWrite(HUM_whiteLed, HIGH);
  } else if (HUM_isPumpOn == false){
    //відключення індикації роботи дивигуна
    digitalWrite(HUM_whiteLed, LOW);
  }
}

boolean HUM_offIndicate() {
  digitalWrite(HUM_blueLed, LOW);
  digitalWrite(HUM_greenLed, LOW);
  digitalWrite(HUM_yellowLed, LOW);
  digitalWrite(HUM_redLed, LOW);
}

boolean humidityLedIndication(float humidityLevel) {
  if (humidityLevel >= 0 && humidityLevel <= 30) {
    digitalWrite(HUM_redLed, HIGH);
  }
  if (humidityLevel >= 30.1 && humidityLevel <= (HUM_lowLine - 0.1)) {
    digitalWrite(HUM_yellowLed, HIGH);
  }
  if (humidityLevel >= HUM_lowLine && humidityLevel <= HUM_highLine)
{
    digitalWrite(HUM_greenLed, HIGH);
  }
```

```
  if (humidityLevel >= HUM_highLine && humidityLevel <= 100) {
    digitalWrite(HUM_blueLed, HIGH);
  }
}
int humiditycountur(){
  modulateHumidityBehavior();
  HUM_offIndicate();
  humidityLedIndication(currentHumidity);

  if (currentHumidity <= HUM_lowLine) {
    //Змінна для старту процесу поливу
    HUM_isPumpOn = true;
  }

  if (currentHumidity >= HUM_highLine) {
    //Змінна для зупинки процесу поливу
    HUM_isPumpOn = false;
  }

  HUM_DCwork();
}

void loop() {
  WaterLevelcontur();
  pHcontur();
  humiditycountur();
  Temperaturecontur();
}
```