Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *«Комп'ютеризована система вирощування рослин за технологічними картами»*

Виконав(ла): студент(ка) *IV* курсу, групи *CIc-42*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

	(підпис)	Демчан Н.І. (прізвище та ініціали)
Керівник	(підпис)	Жаровський Р.О. (прізвище та ініціали)
Нормоконтроль	(підпис)	Луцик Н.С. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Осухівська Г.М. (прізвище та ініціали)
Рецензент	(підпис)	Мудрик І.Я. (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Осухівська Г.М.*
(підпис) (прізвище та ініціали)
« ____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Демчану Назару Івановичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютеризована система вирощування рослин за технологічними картами*

Керівник роботи *Жаровський Руслан Олегович к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*26*» *04* *2024* року № *4/7-468*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи *структурна схема, вихідний код мікроконтролера, функціональна схема,блок схема алгоритму роботи, діаграма послідовностей повідомлень*

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналіз технічного завдання
2. Проектна частина
3. *Практична частина*
4. *Безпека життєдіяльності, основи охорони праці*
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Структурна схема (А1)
Функціональна схема (А1)
Блок схема алгоритму роботи (А1)
Діаграма послідовностей (А1)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	*Пилипець М.І., д.т.н., професор кафедри МТ*		

7. Дата видачі завдання *26.04.2024*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Розробка та затвердження технічного завдання*		
2	*Аналіз технічного завдання*		
3	*Аналіз вимог та принципів організації систем вирощування рослин за технологічними картами*		
4	*Проєктування системи вирощування рослин за технологічними картами*		
5	*Розробка схем і програмного забезпечення системи вирощування рослин за технологічними картами*		
6	*Розробка інструкцій з використання системи*		
7	*Безпека життєдіяльності, основи охорони праці*		
8	*Оформлення кваліфікаційної роботи*		
9	*Попередній захист кваліфікаційної роботи*		
10	*Захист кваліфікаційної роботи*		

Студент _____ *Демчан Н.І.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Жаровський Р.О.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Комп’ютеризована система вирощування рослин за технологічними картами // Кваліфікаційна робота на здобуття освітнього ступеня бакалавр // Демчан Назар Іванович // Тернопільський національний технічний університет імені Івана Пулюя. Факультет комп’ютерно-інформаційних систем та програмної інженерії. Кафедра комп’ютерних систем та мереж. // спеціальність 123 «Комп’ютерна інженерія» // СІс-42 // Тернопіль. 2024 рік // с. – 56, рис – 39, таблиць – 3 лістингів – 14, бібліогр – 15, аркушів А1 – 4, додатки - 3.

Ключові слова: мікроконтролер, алгоритм, програма, автоматичний, догляд, рослини, arduino, esp-01s, технологічні карти.

У кваліфікаційній роботі бакалавра розроблено комп’ютеризовану систему вирощування рослин за технологічними картами. Пояснювальна записка складається із змісту, вступу, чотирьох розділів, висновків та переліку посилань.

У першому розділі була аргументована актуальність обраної теми КР, аналіз ТЗ та аналіз конкурентних систем на ринку України.

У розділі два було виконано: опис і обґрунтування вибору елементної бази, розробку електрично-принципової схеми пристрою, розробку алгоритму роботи КС із описом ключових функцій коду та опис взаємодії контролера з сервером Node.js та MQTT Брокером.

У розділі практичної частини було розроблено інструкція з експлуатації електронного пристрою і методики перевірки, функціонування (контролю, випробування) електронного пристрою, спроектовано прототип КС вирощування рослин за технологічними картами.

У розділі безпеки життєдіяльності та основи охорони праці було описано домедичну допомогу при отруєнні пестицидами та агрохімікатами;вимоги до режимів праці і відпочинку при роботі з ВДТ

ANNOTATION

Computerized plant growing system based on technological maps // Bachelor's thesis // Demchan Nazar Ivanovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information System and Software Engineering, Department of Computer Systems and Networks, group CIs-42 // Ternopil, 2024 //

p. – 56, pictures – 39, tables – 3, listings - 14, bibliogr – 15, sheets A1 – 4, supplements – 3.

Keywords: microcontroller, algorithm, program, automatic, care, plants, arduino, esp-01s, technological maps.

In the bachelor's thesis, a computerized plant growing system based on technological maps was developed. The explanatory note consists of a table of contents, introduction, four chapters, conclusions, and a list of references.

The first section argued the relevance of the chosen topic of the QW, analyzed the TT and analyzed competitive systems in the Ukrainian market.

Section two included: description and justification of the choice of the element base, development of the device's electrical schematic diagram, development of the CS algorithm with a description of the key functions of the code, and description of the controller's interaction with the Node.js server and MQTT Broker.

In the practical part, we developed an instruction manual for the operation of the electronic device and methods for checking, functioning (control, testing) of the electronic device, designed a prototype of the computerized plant growing system based on technological maps.

In the section on life safety and the basics of labor protection, pre-medical assistance in case of poisoning by pesticides and agrochemicals is described; requirements for work and rest modes when working with VDT

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, СИМВОЛІВ І СПЕЦІАЛЬНИХ ТЕРМІНІВ 7
ВСТУП 8
РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ 9
1.1 Аналіз вимог до комп’ютерної системи 9
1.2 Аналітичний огляд існуючих рішень 11
РОЗДІЛ 2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ 16
2.1 Опис і обґрунтування вибору елементної бази 16
2.2 Розробка функціональної схеми пристрою 21
2.3 Розробка алгоритму роботи системи 21
2.4 Зв'язок між Arduino та ESP-01S 30
2.5 Взаємодія системи з MQTT Брокером та Node.js сервером. 31
РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА 35
3.1 Розробка експлуатаційної інструкції пристрою 35
3.2 Реалізація проектних рішень 36
3.3 Розробка методики перевірки, функціонування (контролю, випробування) системи 41
3.4 Розробка прототипу системи 45
РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ 48
4.1 Домедична допомога при отруєнні пестицидами та агрохімікатами 48
4.2 Вимоги до режимів праці і відпочинку при роботі з ВДТ 51
ВИСНОВКИ 54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 55
ДОДАТОК А 57
ДОДАТОК Б 63
ДОДАТОК В 65

ПЕРЕЛІК СКОРОЧЕНЬ, СИМВОЛІВ І СПЕЦІАЛЬНИХ ТЕРМІНІВ

ТЗ – технічне завдання

ПЗ – пояснювальна записка

КС – комп’ютерна система

ЕС – електронна система

ЕП – електронний пристрій

НДР – науково-дослідницька розробка

ОЗУ – оперативний запам’ятовуючий пристрій

ПК – персональний комп’ютер

ПЗ – програмне забезпечення

МК – Мікроконтролер

КР – кваліфікаційна робота

SPA – Single Page Application

ВСТУП

Ми живемо в епоху цифрових технологій та віддаленого зв’язку. За останнє століття людство навчилось ефективно і швидко обмінюватись інформацією між різними пристроями, що відкрило безліч нових можливостей. Тому вже не дивно,що ми не уявляємо своє життя без smart гаджетів і розумних технологій. Щодня мільйони розумних машин покращують життя людини та роблять її роботу ефективнішою.

Сьогодні навіть найпростіші прилади, наприклад кавоварки, тостери, холодильники можуть контролюватися дистанційно за допомогою смартфона або веб-застосунку, що зручно та економить час. Більше того, величезний перелік техніки може виконувати свої функції без участі користувача, тобто автоматично. Ця особливість стає все більш популярною і отримує схвальні відгуки від користувачів. Прилади можуть не тільки керуватися «по кнопці», а і самі надсилати дані про стан системи.

В той же час, нерідко трапляються ситуації, коли в шаленому ритмі життя ви забуваєте доглядати за рослинами. “Комп’ютеризована система вирощування рослин за технологічними картами” не лише дозволяє швидко і зручно керувати поливанням рослин. Система дозволяє вмикати УФ-лампу, на певний час, а також прискорює процес росту, а й автоматично вмикає помпу з водою. Вона є досить легкою у користуванні, а за сайту, ви можете вибирати режими роботи системи(автоматичний, по таймеру, ручний) та технологічну карту для специфічної рослини. За допомогою сайту зручно переглядати метрики системи, а також керування процесом роботи. Цей пристрій чудово підійде для будинків та домашніх ферм, завдяки малим габаритам та простоті у використанні.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Аналіз вимог до комп’ютерної системи

У сучасному світі ведення високоефективного сільського господарства вимагає точного керування процесами вирощування рослин. Технологічні карти стають основою для планування та виконання агротехнічних операцій. З метою автоматизації цих процесів необхідна комп'ютеризована система, яка забезпечить точне виконання технологічних вимог.

У галузі автоматичного догляду за рослинами, система виявляє значний потенціал для використання в приватних житлових будинках та невеликих господарствах. Ця технологія не лише полегшує процес догляду за рослинами, але й забезпечує дистанційний доступ та управління параметрами середовища для домашніх рослин, через веб-сайт. Використовуючи датчики вологості, система автоматично реагує на потреби рослин або ж може бути програмована за допомогою таймера через веб сайт, для доступ з будь якої точки світу, для належного поливу та освітлення, всі ці параметри задаються за допомогою створення та вибору певної технологічної карти для рослини. Це значно спрощує процес ведення домашнього саду та допомагає забезпечити оптимальні умови для росту та розвитку для будь-якої рослини.

Така система може стати невід'ємною частиною домашнього середовища, де власники можуть насолоджуватися красою своїх рослин, не турбуючись про їхні потреби у догляді. Переваги використання автоматичного догляду за рослинами в приватних будинках охоплюють ефективність використання води, енергії та часу, сприяючи створенню здорового та приємного середовища для ветегації рослин.

Основним завданням кваліфікаційної роботи є системи на базі Arduino із можливістю моніторингу параметрів вологості, а також температури рослини та збереження цих даних та відправку на сервер, вибору режиму роботи, а також створення та вибору технологічних карт для різних рослин.

Комп'ютеризована система вирощування рослин за технологічними картами буде базуватися на використанні Arduino UNO та ESP-01S, MQTT протоколу для обміну даними між фізичною складовою системи і Node.js сервера для обробки та аналізу цих даних.

Система призначена для виконання таких функцій:

а) початок/припинення подачі простої води та води з органічним добривом;

б) вкл/викл УФ-лампи;

в) можливість автоматизувати процеси а,б;

г) можливість поставити таймер для процесів а,б;

д) можливість ручного включення для процесів а,б;

е) керування процесами а,б,в,г через сайт;

ж) створення та вибору технологічних карт для рослин та автоматизувати процеси а,б,в,г через сайт.

На рисунку 1.1 продемонстровано структура та функціональні блоки мікропроцесорної системи.

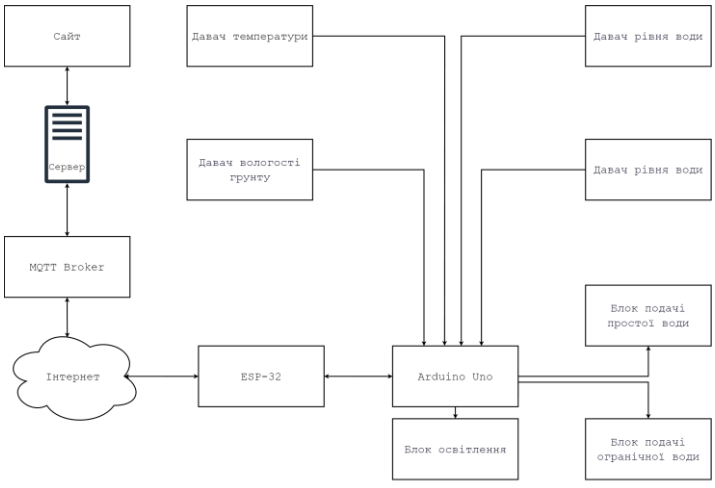

Рисунок 1.1 – Структура та функціональні блоки КС

Дана система складається з Arduino Uno та ESP32-01. Складові системи :

– vікроконтролер Arduino Uno (призначений для зчитування даних з датчиків, керування реле та передачі інформації на WiFi модуль ESP32-01);

– WiFi модуль ESP32-01 (для зв’язку з MQTT Брокером, відповідно сервером Node.js);

– блок подачі органічної та вологої води – RS-360SH помпа на основі моторчика зміного струму для підкачування води, що контролюється за допомогою Arduino;

– датчик вологості грунту – YL-69 [8];

– датчик рівня води – T1592 [9];

– DHT11 – датчик температури та вологості повітря;

– блок освітлення УФ-лампи через реле.

1.2 Аналітичний огляд існуючих рішень

Розроблювальна система, має вузьку спеціалізацію, але вона дуже популярна серед користувачів. Порівнюючи її з аналогами, можна згадати такі рішення, як Wi Drop M12, Dripping Pro 001 . Розглянемо їх більш детально.

а) Wi Drop M12 – достатньо запрограмувати пристрій на певний інтервал та інтенсивність полива. Крапельний полив Wi Drop M12 (рис. 1.2) працює від вбудованого акумулятора. Можливість заддти на пристрої інтервал та інтенсивність поливу. В середньому, одного заряду хватає на 1-2 місяці.

Рисунок 1.2 – Wi Drop M12

Особливості:

- робота від акумулятора на 2200 мА/год;
- робота у режимі енергозбереження;
- робота в двох режимах:крапельне розпилення, полив;
- одночасних полив 12 горшків;
- автоматичне виявлення низького заряду акумулятора;
- система виявлення нехватки води та наступне автоматичне відключення.

б) Dripping Pro 001 – автоматична система поливу для кімнатних рослин, оснащена автономним насосом, здатним обслуговувати до 10 рослин одночасно. Пристрій (рис. 1.3) забезпечує своєчасний і точний полив, підтримуючи оптимальний рівень вологи в ґрунті. Це запобігає проблемам, пов'язаним із пересиханням. Потужний насос дозволяє ефективно поливати навіть ті рослини, які потребують багато води. Система вирізняється можливістю одночасного обслуговування 10 рослин і зручним управлінням за допомогою кнопок на корпусі.

Рисунок 1.3 – Dripping Pro 001

Особливості:

– можливість періодичного, автоматичного поливу домашніх рослин;

– налаштування системи за допомогою кнопок керування системою;

– наявність занурюваного насоса для забору рідини з резервуару;

– живлення всієї системи від батарейок (бажано алкалінові, акумуляторні);

– проста у використанні та надійна конструкція всієї системи автоматичного поливу;

– для роботи потрібен доступ до резервуару з водою та елементи живлення.

в) Jakemy JM-G01 – поливальна система JM-G01(рис. 1.4) – це високоякісний товар, призначений для ефективного та зручного поливу домашніх рослин. Виготовлена з надійних матеріалів, система забезпечує тривалу службу та стабільну роботу, оскілкьий має потужний насос, що забезпечує поливом від 10 до 15 рослин.

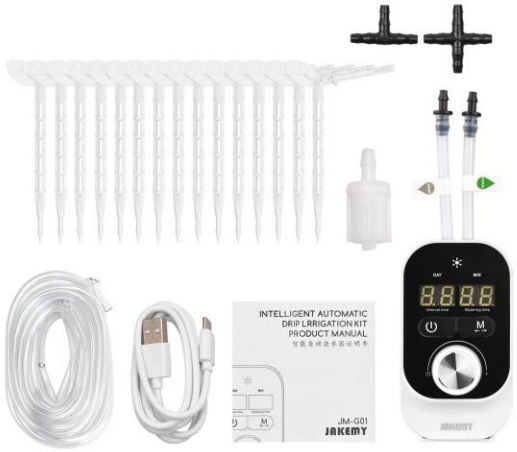

Рисунок 1.4 – Jakemy JM-G01

Основні характеристики:

– широкий спектр застосування: підходить для поливу офісних рослин, балконних квітів, клумб та горшкових рослин;

– висока потужність: може поливати від 10 до 15 горщиків одночасно, що економить час і забезпечує ефективний полив;

– просте встановлення: система постачається з висувною застібкою для кріплення до відра, що дозволяє зручно встановити та зберегти місце;

– гнучкі налаштування: інтервал поливу можна встановити в діапазоні від 1 до 23 годин або від 1 до 30 днів. Один полив може тривати від 10 до 59 секунд або від 1 до 5 хвилин;

– точний полив: конструкція стрілки з листям забезпечує більш точний полив, ніж ручний метод, що сприяє повному використанню та економії води;

– функція пам'яті: оснащена функцією пам'яті, яка зберігає налаштування і дозволяє відновити їх у разі відключення електроенергії, забезпечуючи безперервний догляд за рослинами;

– виявлення відсутності води: якщо у відрі немає води, через 30 секунд після припинення подачі води система активує функцію відсутності

води і на екрані з'являється повідомлення про помилку, сповіщаючи користувача про необхідність поповнити запаси води.

Тому, проаналізувавши вище перелічені продукти, було взято та зважено всі плюси та мінуси цих систем. Було вирішено проектувати мікропроцесорну систему з такими особливостями:

- можливості функції автоматичного поливу по датчикам вологості ґрунту;
- сповіщати користувача про нестачу води у резервуарах;
- автоматичне ввімкнення/вимкнення фітолампи для росту рослин;
- різні режими роботи, такі як: автоматичний, ручний та по таймеру;
- дистанційне управління, за допомогою сайту;
- введення технологічних карт для рослин;
- вибір технологічних карт;
- зберігання метрики.

Ця система дозволить оптимізувати догляд за рослинами, забезпечуючи ефективне використання ресурсів та зручне управління процесом.

РОЗДІЛ 2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ

2.1 Опис і обґрунтування вибору елементної бази

Для розробки цієї системи можна використовувати різні програмовані пристрої. Основні з них включають персональні комп'ютери та програмовані контролери, такі як платформи Arduino, STM, Raspberry Pi та інші аналогічні пристрої.

В нашому випадку, для реалізації проекту нам потрібно буде використати, як платформу Arduino [6] для взаємодії з датчиками і давачами, так і персональний комп`ютер. В подальшому з використання хмарних технології, такі як AWS, Microsoft Azure, Google Cloud, для хостингу реляційної бази даних та роботи нашого Node.js сервера і SPA.

В якості основної плафторми для реалізації була вибрана Arduino. Особлива простота в експлуатації, доступність, гнучкість та можливість локального розташування пристрою забезпечують більш точний контроль та управління умовами, що сприяють зростанню рослин.

Arduino Uno виступатиме у ролі керуючого модуля, а ESP-01S [7] у ролі інформаційного, для передачі даних в інтернет.

Мікроконтролер Atmega-328-P 8-біт буде відповідати за обробку даних у системі. Arduino Uno має 14 цифрових виходів та 6 аналогових інтерфейсів, працює на частоті 16 МГц, має USB-інтерфейс, інтерфейс живлення, контакти ICSP та кнопку скидання.

Система може оптимально функціонувати за умови використання зовнішнього джерела живлення з напругою від 6 В до 20 В.

Проте, важливо відзначити, що при живленні менше 7 В вихідний сигнал 5 В може бути зниженим, що може вплинути на стабільність роботи платформи. Треба бути обережним у таких ситуаціях.

З іншого боку, при використанні напруги понад 12 В внутрішній регулятор може перегрітися, що може спричинити пошкодження плати. Тому рекомендується уникати застосування напруги більше 12 В для забезпечення безпечної та надійної роботи платформи. Технічні параметри Arduino Uno подано у таблиці 2.1

В якості редактора написання коду для програмування мікрокотролера була вибрана Arduino IDE. Arduino IDE - це спеціальне програмне забезпечення для розробки програм для платформ Arduino. Вона дозволяє легко писати, компілювати та завантажувати програми на мікроконтролери Arduino.

Для написання серверної та клієнтської частини була вибрана мова пограмування TypeScript з фреймоврками Nest.js [11] та React.js [12] відповідно.

Таблиця 2.1 – Технічні параметри Arduino Uno

Назва	Характеристика
Мікроконтролер	Atmega328P
Робоча напруга	5 В
Вхідна напруга (рекомендована)	7-12 В
Вхідна напруга (гранична)	6-20 В
Цифрові Входи / Виходи	14
Аналогові входи	8
Постійний струм через вхід / вихід	40 мА
Постійний струм для виведення 3.3В	50 мА
Флеш-пам’ять	32 Кб (ATmega328P), з яких 0.5 Кб
ОЗУ	2 Кб (ATmega328P)
EEPROM	1 Кб (ATmega328P)
Тактова частота	16 МГц

Як описувалось вище модуль ESP-01 із чіпом ESP8266 використовується для бездротового зв'язку через Wi-Fi. На його платі розташована 2-мегабайтна Flash-пам'ять, чіп ESP8266EX, п'єзоелектричний резонатор, два світлодіодні індикатори і компактна антена, яка розміщена у вигляді зигзагу на поверхні плати. Flash-пам'ять використовується для зберігання програмного забезпечення. Під час включення мікроконтролер автоматично завантажує програмне забезпечення на ESP8266EX.

Модуль призначений для роботи з "АТ-командами". Керуючий контролер відправляє команди, які виконуються ESP. В самому чіпі ESP8266 міститься вбудований мікроконтролер, що діє як самостійний пристрій. Технічні параметри ESP8266 подано у таблиці 2.2.

Таблиця 2.2 – Технічні параметри ESP8266

Назва	Характеристика
Флеш-пам'ять	до 16мб максимум (512кб)
Робоча напруга	3,3В
Процесор	Tensilica L106, 32б
Швидкість процесора	80-160МГц
ОЗУ	32кб, 80кб

Для підключення цього модуля використовують послідовні порти - Tx та Rx.

Tx - це вихідний порт, а Rx - вхідний порт. Вони підключаються до відповідних портів на Arduino. Найпростіший спосіб підключення - використовувати USB-UART перехідник для з'єднання з комп'ютером. Це дозволяє надсилати команди з терміналу і миттєво отримувати відповіді від модуля.

Але також можна під’єднувати модуль через Arduino,у вигляді програматора. Для цього потрібно з’єднати Rx - Rx, Tx - Tx, спільну землю, а також замкнути землю пін RESET на Arduino, як це зображено на рисунку 2.1.

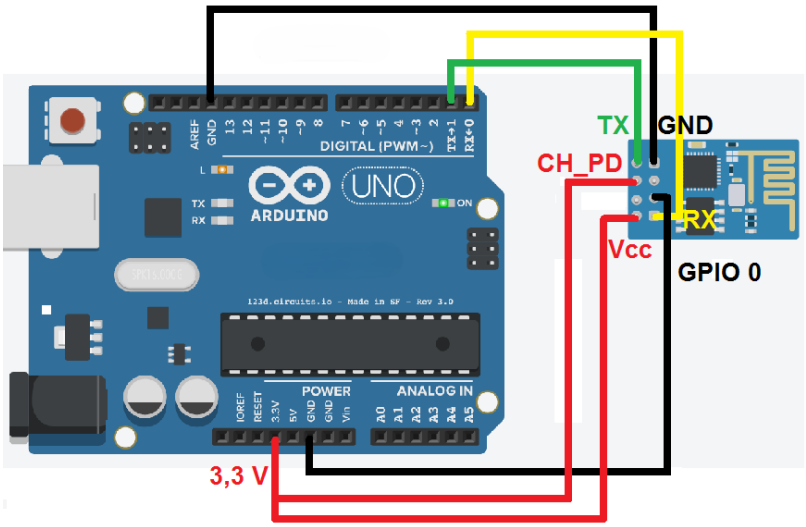

Рисунок 2.1 – Підключення ESP-01 до Arduino у вигляді програматора

Для вимірювання вологості ґрунту в проектованій системі використовується датчик, що складається з модулів YL-69 та YL-38.

На електродах зонда YL-69 генерується слабка напруга. У сухому ґрунті опір високий, що зменшує струм, тоді як у вологому ґрунті опір знижується, збільшуючи струм. Аналізуючи цей аналоговий сигнал, можна визначити рівень вологості ґрунту. Шуп YL-69 підключений до датчика YL-38 через два дроти характеристики давача YL-69 наведено у таблиці 2.3.

Таблиця 2.3 – Технічні характеристики YL-69

Назва	Характеристика
Діапазон вимірюваної вологості	20 - 95%
Робоча напруга	5V-3.3V
Точність	-5%
Робоча температура	0-60°C
Можливість встановлювати порогове значення вологості	Є
Керуючий чіп	LM393

Мініатюрний водяний насос - це компактний погружний пристрій, призначений для перекачування води між різними ємностями. Він здатний перекачувати рідину зі швидкістю до 120 літрів на годину. Використовується для поливу рослин, в акваріумах, фонтанах тощо.

Конструкція електродвигуна постійного струму подібна до конструкції генератора постійного струму. В електродвигуні завдяки взаємодії струму в обмотці ротора (якоря) з основним магнітним полем виникає електромагнітна індукція, яка створює обертальний момент.

Технічні характеристики RS-360SH:

- номінальна напруга: 7.2V;
- середня продуктивність: 1 літр - 75 сек. (При харчуванні 6V);
- витрата води: 30 л / год;
- діаметр вхідного і вихідного патрубків: 4 мм;
- тиск води: 0.15 bar (при використанні 12V);
- загальний розмір: 65х44мм;
- матеріал: метал + пластик;
- вага: 75гр.

Для отримання інформації про ємкість резервуарів води було використано T1592[9] - аналоговий датчик. Для цих типів датчиків різні конструкції, а саме: поплавцеві, занурені, врізні. Даний датчик води – занурений.

Внутрішній опір датчика змінюється залежно від площі контакту з рідиною, що впливає на напругу. Ці коливання напруги дозволяють визначати рівень води. Чим глибше занурений датчик, тим менший опір між двома сусідніми проводами. Датчик має три контакти для підключення до контролера: живлення, землю та аналоговий сигнал, який підключається до аналогового входу. Під час зчитування даних із цього входу значення збільшується зі зростанням рівня води. Крім того, датчик оснащений червоним світлодіодом, що сигналізує про наявність живлення.

2.2 Розробка функціональної схеми пристрою

Розробка принципової схеми здійснюється відповідно до структури функціональних вузлів і елементів системи, представленої на рисунку 1.1. Функціональна схема системи наведена на рисунку 2.2.

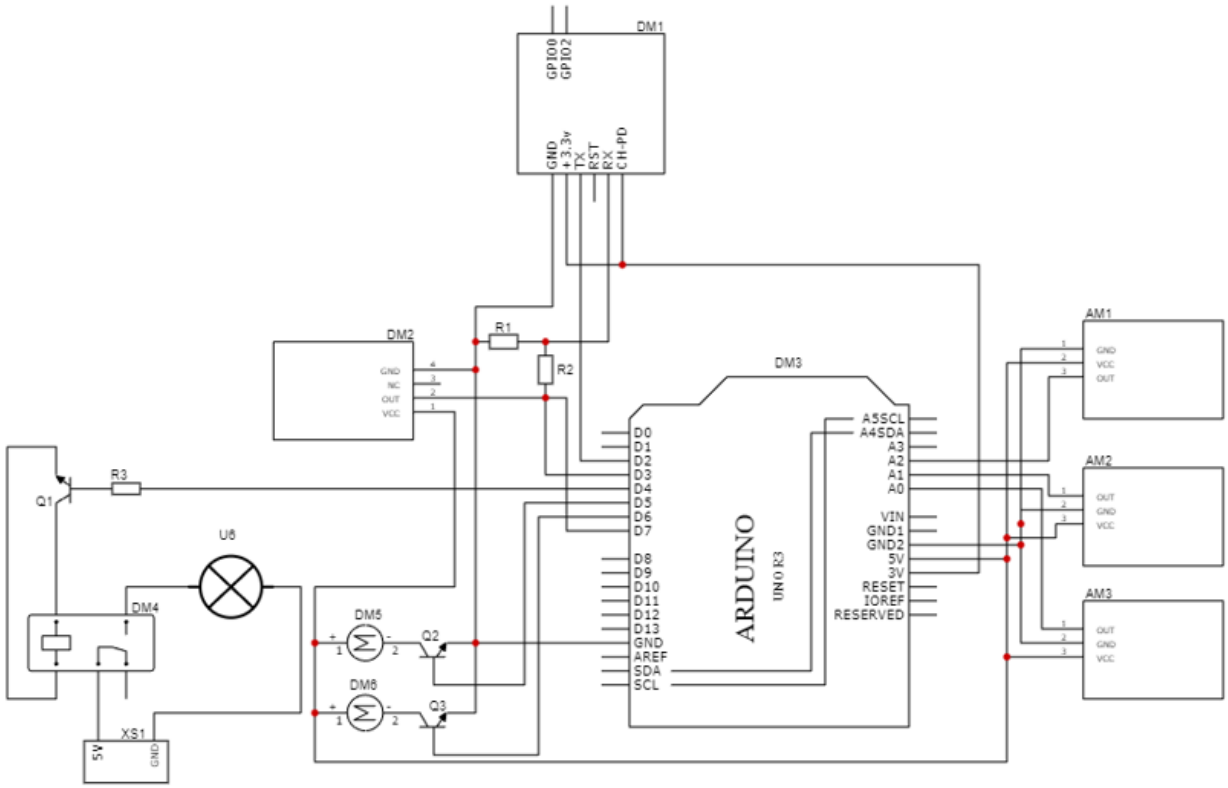

Рисунок 2.2 – Функціональна схема системи

2.3 Розробка алгоритму роботи системи

Аналізуючи поставлені задачі, які має виконувати ЕС складено алгоритм роботи виконавчого модуля:

Алгоритм роботи виконавчого модуля:

а) ініціалізація бібліотек і датчиків;

б) оголошення змінних: waterTanksLvl, currentTime типу integer призначена для запису даних рівня води в посудинах з водою та часу роботи ардуїно, для відліку часу роботи насосів для лампи;

в) отримання даних з ESP-01S, що передаються через UART через MQTT Broker;

г) оголошення змінних та приймання значень з ESP-01S, а саме:

- durationBulp, durationSimple;
- durationOrganic, humidityTarget;
- humidityTargetOrganic типу integer;
- turnOnBulp, turnOnOrganic, turnOnSimple типу bool;
- mode типу String.

ґ) якщо waterTanksLvl < 5, відсилається повідомлення на ESP-01S та в свою чергу відсилає на MQTT Broker, сервер в свою чергу слухає повідомлення та відсилає на сайт, повідомлення;

д) якщо mode =="Auto", вибраний режим автоматичний, тоді якщо:

– currentTime<=durationBulp, включити лампу, в іншому випадку виключити її;

– humidityValue < humidityTargetOrganic && waterTanksLvl > 5, включити насос з органічною водою, в іншому випадку виключити його;

– humidityValue < humidityTarget && waterTanksLvl > 5, включити насос з простою водою, в іншому випадку виключити його;

е) якщо mode =="Manual", вибраний режим ручний, отримуємо дані з ESP-01S, тоді якщо:

– turnOnBulp, включити лампу, в іншому випадку виключити її;

– turnOnOrganic && waterTanksLvl > 5 включити насос з органічною водою, в іншому випадку виключити його;

– turnOnSimple && waterTanksLvl > 5 включити насос з простою водою, в іншому випадку виключити його;

є) Якщо mode =="Timer", вибраний режим автоматичний, тоді якщо:

– turnOnBulp, включити лампу, в іншому випадку виключити її;

– turnOnOrganic && waterTanksLvl > 5, включити насос з органічною

водою, в іншому випадку виключити його;

– tunrOnSimple && waterTanksLvl > 5, включити насос з простою водою, в іншому випадку виключити його;

Блок-схема алгоритму роботи виконавчого модуля представлено у вигляді блок-схеми на рисунках 2.3 – 2.6.

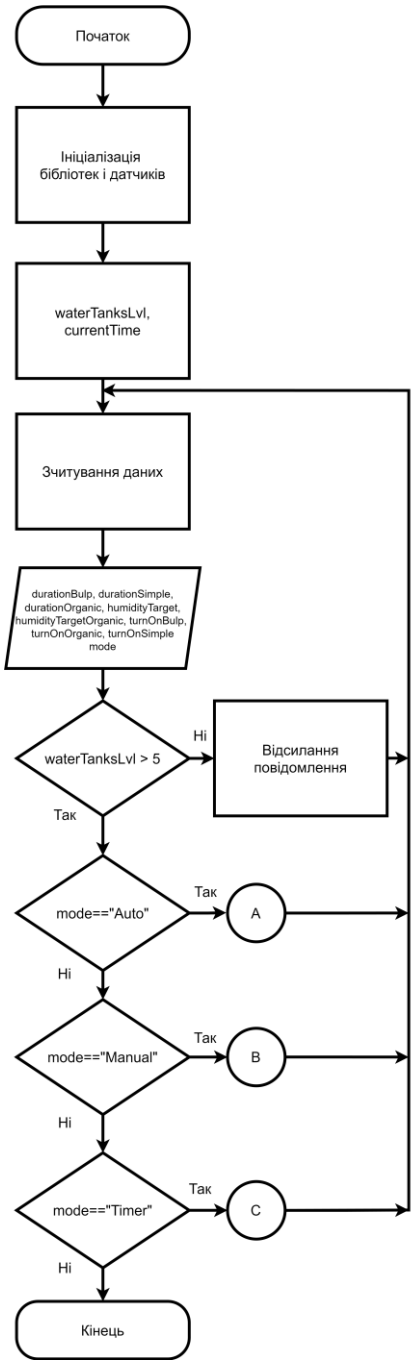

Рисунок 2.3 – Алгоритм роботи виконавчого модуля

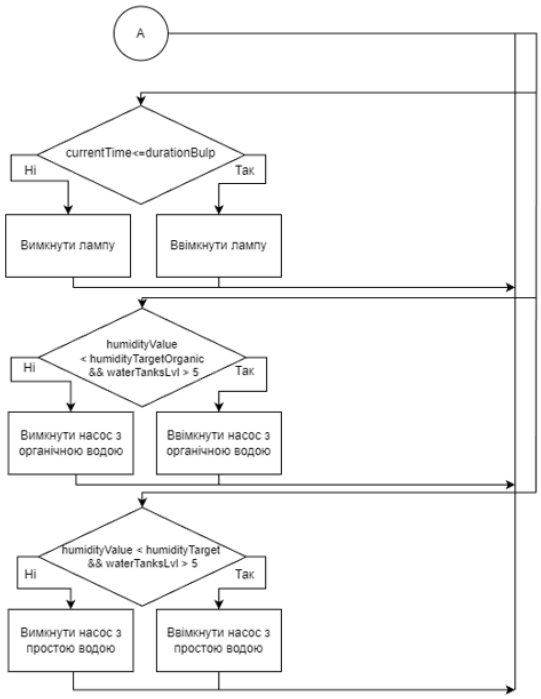

Рисунок 2.4 – Алгоритм роботи виконавчого модуля при виборі автоматичного режиму

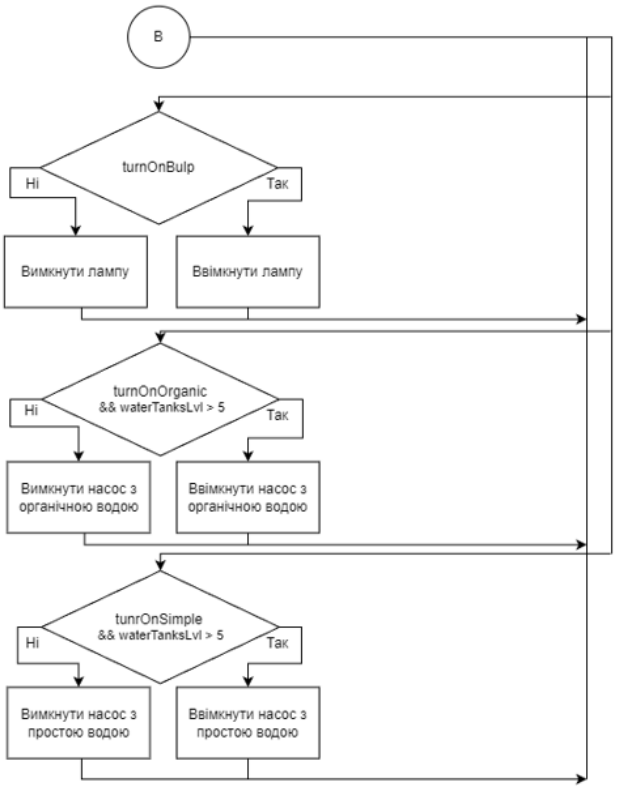

Рисунок 2.5 – Алгоритм роботи виконавчого модуля при виборі ручного режиму

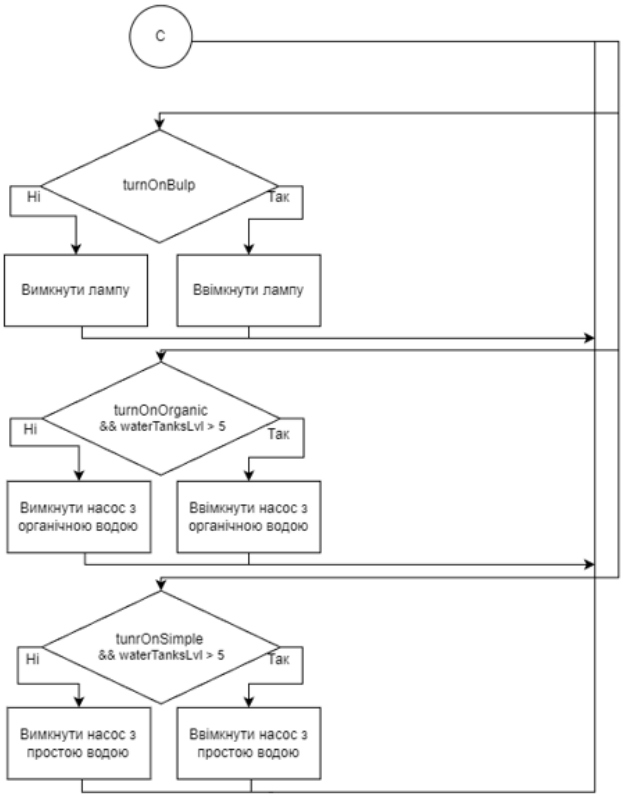

Рисунок 2.6 – Алгоритм роботи виконавчого модуля при виборі режиму по таймеру

Алгоритм роботи інформаційного модуля:

а) ініціалізація бібліотек і оголошення змінних;

б) підключення до WI-FI мережі;

в) підписка на топіки;

г) слухання топіків;

д) отримання даних;

е) перевірка на отримання даних з MQTT / відправка їх на керуючий модуль;

ж) перевірка на отримання даних з керуючого модуля / відправка їх на MQTT брокер.

Блок-схема алгоритму роботи інформаційного модуля представлено у вигляді блок-схеми на рисунку 2.7.

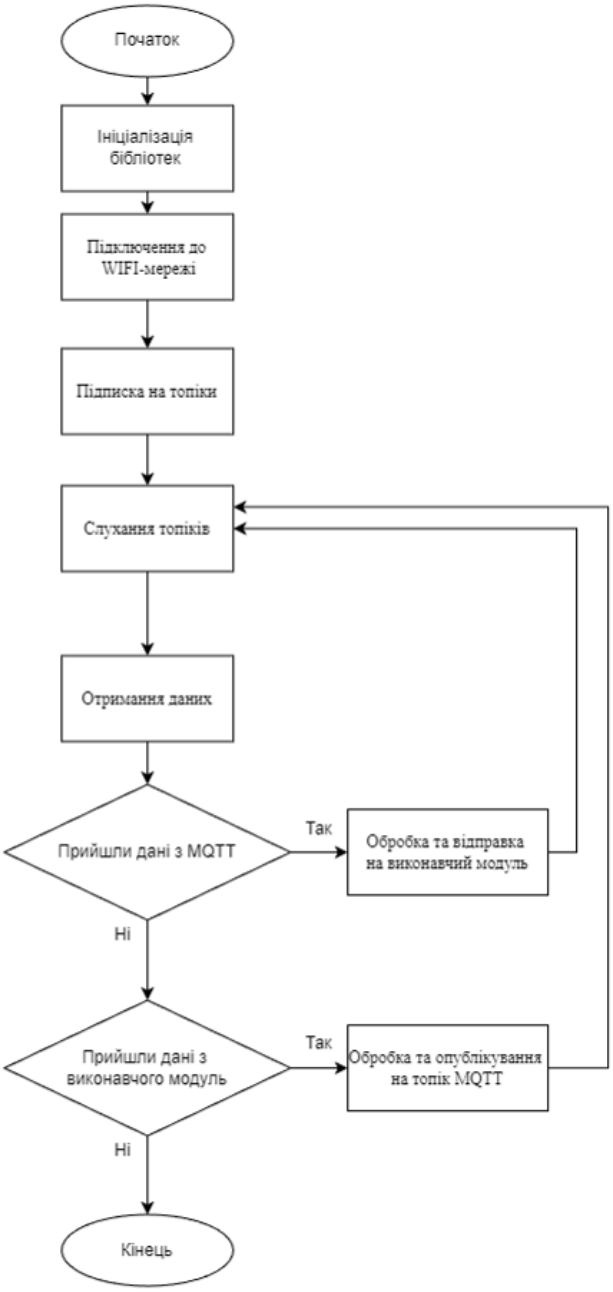

Рисунок 2.7 – Алгоритм роботи інформаційного модуля

Відповідно до вище зазначених алгоритмів був написаний код програми виконавчого модуля та інформаційного та код до них наведено в додатку А та Б відповідно.

На рисунку 2.8. зображеного лістинг коду ініціалізації бібліотек та оголошення змінних виконавчого модуля.

```
#include <SoftwareSerial.h> // Підключення бібліотеки для програмного послідовного порту

#include <Stepper.h> // Підключення бібліотеки для керування кроковим двигуном

#include <DHT.h> // Підключення бібліотеки для роботи з датчиком DHT

#include <ArduinoJson.h> // Підключення бібліотеки для роботи з JSON-форматом

#define STEPS 2038 // Визначення кількості кроків для повного оберту крокового двигуна

#define DHTTYPE DHT11 // Визначення типу датчика DHT як DHT11

#define dthPin 7 // Визначення піна, до якого підключений датчик DHT

Stepper stepper(STEPS, 8, 9, 10, 11); // Ініціалізація крокового двигуна з вказаною кількістю кроків та пін-кодами для підключення

DHT sens(dthPin, DHTTYPE); // Ініціалізація об'єкта датчика DHT з вказаним пін-кодом та типом датчика

#define photo A0 // Визначення піна для фоторезистора

#define moisture A2 // Визначення пінадля датчика вологості грунту

#define dht2 A3 // Визначення піна для другого датчика DHT

#define dht3 A4 // Визначення піна для третього датчика DHT

#define LED_PIN 6 // Визначення піна для керування світлодіодом

int photo_value; // Змінна для зберігання значення з фоторезистора

int tmp_value; // Змінна для зберігання значення температури

int humidity_value; // Змінна для зберігання значення вологості

bool turnOnBulb; // Логічна змінна для керування ввімкненням/вимкненням лампи

bool turnOnOrganic; // Логічна змінна для керування ввімкненням/вимкненням органічного режиму

bool turnOnSimple; // Логічна змінна для керування ввімкненням/вимкненням простого режиму

bool isWater = true; // Логічна змінна для перевірки наявності води

String MIN_TMP; // Змінна для зберігання мінімальної температури (як строка)

String MIN_MOISTURE; // Змінна для зберігання мінімальної вологості (як строка)

    int humidityValueTarget; // Цільове значення вологості для простого режиму

    int humidityValueTargetOrganic; // Цільове значення вологості для органічного режиму

    unsigned long timerTarget; // Цільове значення таймера

    unsigned long currentMillis = 0; // Поточний час у мілісекундах

    unsigned long rememberTimeSimple = 0; // Змінні для зберігання часу останнього включення різних режимів

    unsigned long rememberTimeOrganic = 0;

    unsigned long rememberTimeBulp = 0;

    unsigned long durationBulp = 0; // Змінні для зберігання тривалості включення різних режимів

    unsigned long durationSimple = 0;

    unsigned long durationOrganic = 0;

    SoftwareSerial serialEsp(2, 3); // Ініціалізація програмного послідовного порту з вказаними пін-кодами для RX та TX

    const long onDuration = 300; // Визначення тривалості включення у мілісекундах
```

Рисунок 2.8 – Лістинг ініціалізація бібліотек та оголошення змінних

На рисунку 2.9 зображено лістинг коду конфігурація пінів

```
void setup() {
  stepper.setSpeed(5); // Встановлює швидкість крокового двигуна на 5 RPM (обертів на хвилину)
  sens.begin(); // Ініціалізує датчик DHT для початку вимірювання температури та вологості

  pinMode(photo, INPUT); // Встановлює режим піна для фоторезистора як вхідний

  digitalWrite(dthPin, HIGH); // Встановлює високий рівень на пін-коді датчика DHT (можливо для його живлення)

  pinMode(LED_PIN, OUTPUT); // Встановлює режим піна для світлодіода як вихідний

  Serial.begin(19200); // Ініціалізує апаратний послідовний порт з швидкістю 19200 бод
  serialEsp.begin(19200); // Ініціалізує програмний послідовний порт з швидкістю 19200 бод
}
```

Рисунок 2.9 – Лістинг конфігурації пінів

На рисунку 2.10 зображено лістинг оримання перевірки надходження даних та парсинг з інформаційного модуля.

```
if (serialEsp.available() > 0) { // Перевіряє, чи є дані, доступні для читання з програмного послідовного порту
  String incomingString = serialEsp.readString(); // Зчитує вхідний рядок з програмного послідовного порту
  Serial.println("Received String TO ARDUINO FROM ESP: " + incomingString); // Друкує отриманий рядок на апаратний послідовний порт
  DynamicJsonDocument doc(1024); // Створює об'єкт JSON документа з розміром 1024 байти
  deserializeJson(doc, incomingString); // Десеріалізує отриманий рядок JSON в об'єкт doc
  String data22 = (const char*)doc["data"]; // Зберігає значення поля "data" в рядок data22
  mode = (const char*)doc["data"]["mode"]; // Зберігає значення поля "mode" в змінну mode
  Serial.println("data22: " + data22); // Друкує значення data22 на апаратний послідовний порт
```

Рисунок 2.10 – Лістинг надходження даних та парсинг з інформаційного модуля

На рисунку 2.11 – 2.13 зображено лістинг перевірка вибраного режиму та присвоєння значення зміних автоматичного, ручного та режиму по таймеру відповідно.

```
  if (mode == "Auto") { // Якщо режим "Auto"

    resetAll(); // Викликає функцію resetAll для скидання всіх
параметрів

    rememberTimeSimple = 0; // Скидає час останнього включення
простого режиму

    rememberTimeOrganic = 0; // Скидає час останнього включення
органічного режиму

    rememberTimeBulp = 0; // Скидає час останнього включення лампи

    humidityValueTarget = 0; // Скидає цільове значення вологості

    humidityValueTarget = doc["data"]["humidity target"]; // Зберігає
цільове значення вологості з JSON

    humidityValueTargetOrganic =
doc["data"]["humidity target organic"]; // Зберігає цільове значення
вологості для органічного режиму з JSON

    durationBulp = doc["data"]["duration bulb"]; // Зберігає
тривалість включення лампи з JSON

    durationBulp = durationBulp * 3600000; // Конвертує тривалість
включення лампи з годин в мілісекунди

  }
```

Рисунок 2.11 – Лістинг надходження автоматичного режиму

```
  if (mode == "Manual") { // Якщо режим "Manual"

    resetAll(); // Викликає функцію resetAll для скидання всіх
параметрів

    rememberTimeSimple = 0; // Скидає час останнього включення
простого режиму

    rememberTimeOrganic = 0; // Скидає час останнього включення
органічного режиму

    rememberTimeBulp = 0; // Скидає час останнього включення лампи

    humidityValueTarget = 0; // Скидає цільове значення вологості

    humidityValueTargetOrganic = 0; // Скидає цільове значення
вологості для органічного режиму

    turnOnBulb = doc["data"]["turn on bulb"]; // Зберігає значення
керування лампою з JSON

    turnOnOrganic = doc["data"]["turn on organic water"]; // Зберігає
значення керування органічним поливом з JSON

    turnOnSimple = doc["data"]["turn on simple water"]; // Зберігає
значення керування простим поливом з JSON

  }
```

Рисунок 2.12– Лістинг надходження ручного режиму

```
if (mode == "Timer") { // Якщо режим "Timer"

    resetAll(); // Викликає функцію resetAll для скидання всіх
параметрів

    rememberTimeSimple = 0; // Скидає час останнього включення
простого режиму

    rememberTimeOrganic = 0; // Скидає час останнього включення
органічного режиму

    rememberTimeBulp = 0; // Скидає час останнього включення лампи

    humidityValueTargetOrganic = 0; // Скидає цільове значення
вологості для органічного режиму

    durationBulp = doc["data"]["duration_bulb"]; // Зберігає
тривалість включення лампи з JSON

    durationBulp = durationBulp * 3600000; // Конвертує тривалість
включення лампи з годин в мілісекунди

    durationSimple = doc["data"]["duration_simple"]; // Зберігає
тривалість простого поливу з JSON

    durationSimple = durationSimple * 60000; // Конвертує тривалість
простого поливу з хвилин в мілісекунди

    durationOrganic = doc["data"]["duration_organic"]; // Зберігає
тривалість органічного поливу з JSON

    durationSimple = durationOrganic * 60000; // Конвертує тривалість
органічного поливу з хвилин в мілісекунди

  }
```

Рисунок 2.13 – Лістинг надходження режиму по таймеру

2.4 Зв'язок між Arduino та ESP-01S

Для передачі даних між цими двома контролерами, реалізовано через протокол передачі даних UART.

UART – це протокол послідовної передачі даних, який дозволяє двом пристроям обмінюватися інформацією через послідовний інтерфейс. Для зв'язку між Arduino та ESP-01S через UART необхідно налаштувати обидва пристрої таким чином, щоб вони могли обмінюватися даними, детальна схема підключення зображенна на рисунку 2.14.

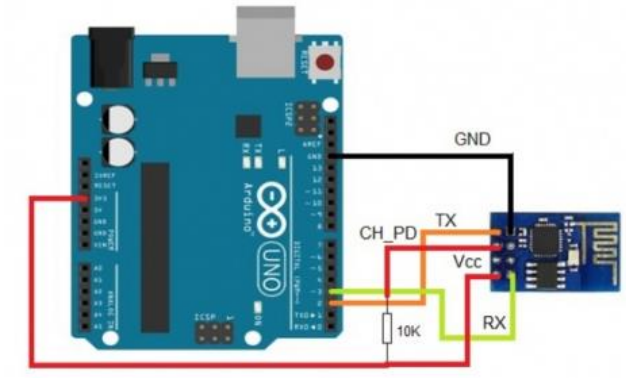

Рисунок 2.14 – Реалізація підключення UART

UART є асинхронним протоколом, що означає, що передача даних не потребує спільного тактового сигналу. Однак обидва пристрої повинні бути налаштовані на однакову швидкість передачі даних, також протокол забезпечує двонаправлений зв'язок, що дозволяє обом пристроям передавати та приймати дані одночасно.

Варто замітити, що UART використовує буфери для зберігання вхідних та вихідних даних, що дозволяє обробляти дані асинхронно. Кожен пристрій зберігає отримані дані в буфері, поки вони не будуть прочитані програмою.

Передача даних здійснюється побітово. TX пін одного пристрою відправляє дані, які приймаються RX піном іншого пристрою.

Перед відправкою даних встановлюється стартовий біт, який сигналізує про початок передачі, потім передаються біти даних і в кінці передається стоповий біт, як це зображено на рисунку 2.15.

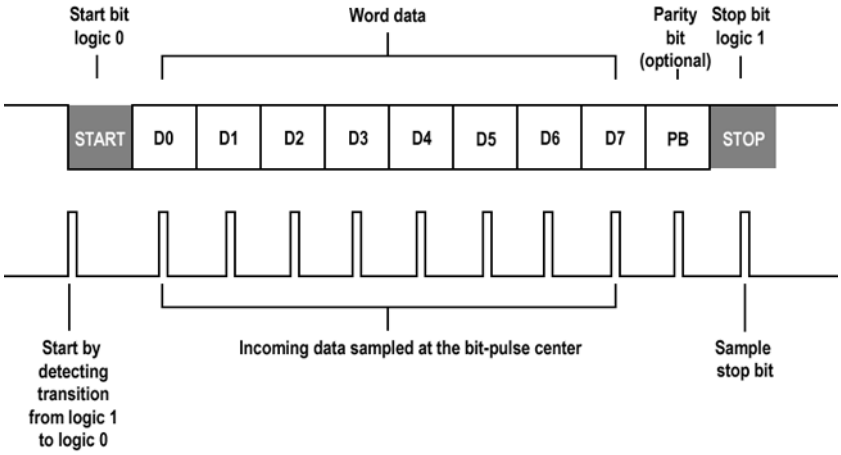

Рисунок 2.15 – Відправка даних через UART

2.5 Взаємодія системи з MQTT Брокером та Node.js сервером.

Для віддаленого моніторингу даних із давачів та зворотньою взаємодію використовується сервер Node.js, база даних PostrgeSQL та протокол передачі даних MQTT.

MQTT – це протокол, заснований на стандартах, або набір правил для обміну повідомленнями, який використовується для взаємодії між комп'ютерами. Розумні датчики, носимі пристрої та інші пристрої Інтернету речей (IoT) зазвичай передають і отримують дані через мережі з обмеженими ресурсами та пропускною здатністю. Ці пристрої IoT використовують MQTT для передачі даних, оскільки його легко реалізувати і він може ефективно передавати дані IoT. MQTT підтримує обмін повідомленнями між пристроями та хмарою в обох напрямках, приклад передачі повідомлень зображено на рисунку 2.16 на мікроконтролер та на рисунку 2.17 з мікроконтролера на сервер.

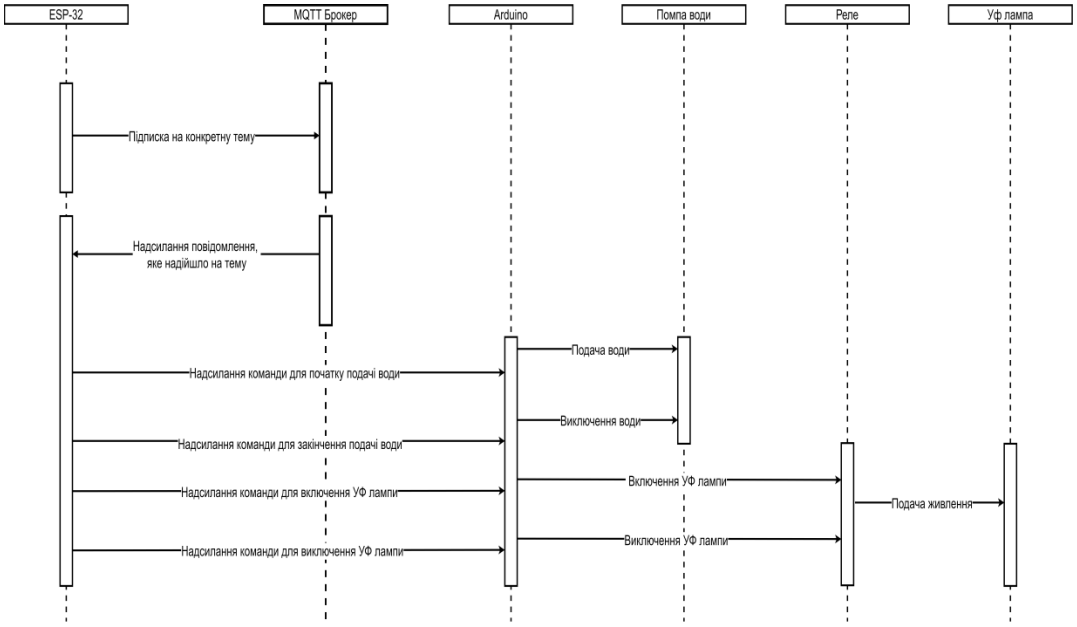

Рисунок 2.16 – Діаграма послідовностей отримання повідомлень

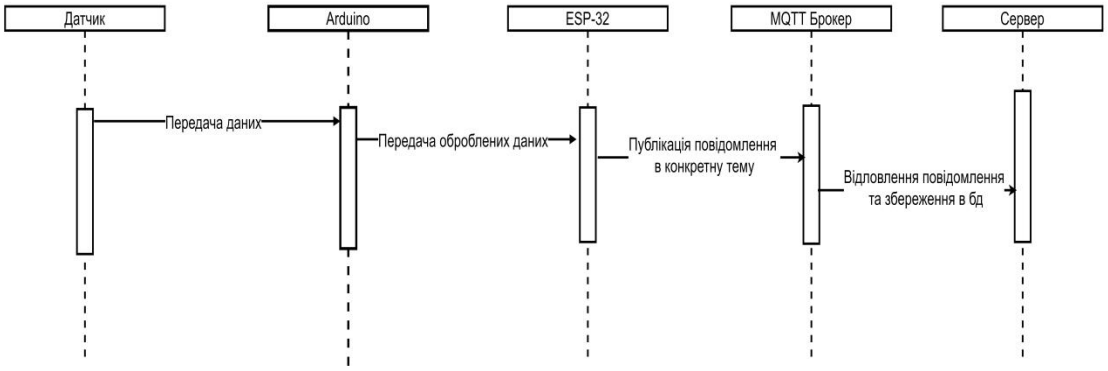

Рисунок 2.17 – Діаграма послідовностей відсилання повідомлень

На рисунку 2.18 зображено лістинг з`єднання до MQTT Брокера та конфігурація WIFI з’єднання.

```
#include <ESP8266WiFi.h> // Підключення бібліотеки для роботи з Wi-Fi на ESP8266
#include <PubSubClient.h> // Підключення бібліотеки для роботи з MQTT
#include <ArduinoJson.h> // Підключення бібліотеки для роботи з JSON
#include <SoftwareSerial.h> // Підключення бібліотеки для програмного послідовного порту
#define DEBUG true // Визначення макросу для режиму налагодження
// Дані для підключення до Wi-Fi
#define ssid "" // SSID (назва) Wi-Fi мережі
#define password "" // Пароль Wi-Fi мережі
WiFiClient client; // Ініціалізація об'єкта WiFiClient для з'єднання з мережею
// Дані про MQTT брокер
String device_id = "Device0001"; // Ідентифікатор пристрою
const char *mqtt_server = "192.168.0.78"; // IP-адреса MQTT брокера
const int mqtt_port = 1883; // Порт MQTT брокера
const char *mqtt_user = "testing"; // Ім'я користувача для підключення до MQTT брокера
const char *mqtt_password = "testing"; // Пароль для підключення до MQTT брокера
const char *mqtt_clientId = "Deivce_Device0001"; // Ідентифікатор клієнта для MQTT
const char *topic_publish = "notifications"; // Тема для публікації повідомлен
const char *topic_publish_sensors = "sensors"; // Тема для публікації даних з сенсорів
const char *topic_publish_lvl_of_water_1 = "tank_lvl_water_1"; // Тема для публікації рівня води в баку 1
const char *topic_publish_lvl_of_water_2 = "tank_lvl_water_2"; // Тема для публікації рівня води в баку 2
const char *topic_subscribe = "setup"; // Тема для підписки на налаштування
void callback(char *topic, byte *payload, unsigned int length); // Прототип функції зворотного виклику для обробки повідомлень MQTT
PubSubClient mqtt_client(mqtt_server, mqtt_port, callback, client);
```

Рисунок 2.18 – Лістинг з`єднання до MQTT Брокера та конфігурація WIFI з’єднання

Також варто зазначити всі переваги цього протоколу:

– легкість та ефективність - реалізація MQTT на пристрої IoT вимагає мінімальних ресурсів, тому його можна використовувати на маленьких мікроконтролерах. Наприклад, мінімальне керуюче повідомлення MQTT може складатися всього лише з двох байтів даних. Заголовки повідомлень MQTT також малі, що дозволяє оптимізувати пропускну здатність мережі;

– масштабованість - для реалізації MQTT потрібна мінімальна

кількість коду, який потребує дуже мало енергії. Крім того, протокол має вбудовані функції для забезпечення взаємодії з великою кількістю пристроїв IoT. Тому ви можете використовувати протокол MQTT для підключення мільйонів таких пристроїв;

– надійність - багато пристроїв IoT підключаються через ненадійні стільникові мережі з низькою пропускною здатністю та високою затримкою. MQTT має функції, що зменшують час, необхідний для відновлення підключення пристрою IoT до хмари. Також протокол визначає три рівні якості обслуговування, щоб забезпечити надійність: максимум один раз (0), мінімум один раз (1) і рівно один раз (2) ;

– безпека - MQTT полегшує розробникам завдання шифрування повідомлень та автентифікації пристроїв і користувачів за допомогою сучасних протоколів аутентифікації, таких як OAuth, TLS1.3;

– хороша підтримка - деякі мови програмування, наприклад Python, C#, C++, JS забезпечують відмінну підтримку протоколу MQTT. Тому розробники можуть швидко реалізувати його з мінімальною кількістю коду в додатку будь-якого типу.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Розробка експлуатаційної інструкції пристрою

Щоб коректно виконувала свої функції, до Arduino необхідно підключити такі датчики/модуля: WiFi модуль, датчик DHT22, датчик T1592, щупи для вимірювання вологості ґрунту або так званий YL-69, реле та УФ лампу. Підключення слід виконувати відповідно до функціональної схеми.

Спершу, щоб почати користуватися пристроєм, необхідно забезпечити живлення Arduino та модулю реле, а також підключити їх разом, а також приєднати ESP-01S, а також прописати пароль та назву вашої WIFI мережі.

Після активації живлення на платі відбуваються наступні дії:

а) WiFi модуль встановлює зв'язок з мереженю, після успішного підєднання встановлюється зв'язок із брокером

б) далі відбувається зчитування даних із давачів, а саме:

- з датчика DHT22 зчитується температура;
- з датчика вологості YL-69 зчитується вологість ґрунту;
- кожні 5 секунд дані з цих датчиків збираються і потім повідомлення публікується через брокер. Сервер приймає це повідомлення, одночасно зберігає його в базу даних та передає на клієнтську сторону.

в) вибір або створення технлогічної карти рослии та режиму роботи системи

г) в залежності від вибранго режиму йде, перевірка умов, а саме:

- вологості грунту, якщо вони нищі за норми, подається сигнал на увімкнення помпи з простою водою та з органічною водою;
- перевірка часу тривалості ввімкнення УФ-лампи;
- перевірка наявності води у ємкостях з водою;
- перевірка вибраного режиму роботи.

Щоб переглянути зібрані дані, та керування вибраним режимом та вибором технологічною картою рослини, потрібно відкрити браузер і перейти на наш веб-сайт за посиланням localhost:3000. Система спроектована таким чином, щоб мінімізувати людське втручання: просто підключіть живлення,вкрутіть УФ-лампочку та наберіть резервуари з водою і більше нічого не потрібно.

3.2 Реалізація проектних рішень

Для розробки серверної частини, для взаємодії з клієнтською частиною, був використаний Node.js.

Node.js сервер - це серверна програма, яка використовує Node.js, платформу для виконання JavaScript поза браузером. Вона може бути створена для обробки запитів і відповідей HTTP, обробки даних, взаємодії з базою даних, виконання операцій вводу-виводу та інших завдань.

Node.js сервер може служити як основа для веб-додатків, API, веб-сервісів та інших типів програмного забезпечення. Він здатний обробляти одночасні запити з великою кількістю відвідувачів, завдяки асинхронній моделі програмування Node.js. Через свою швидкодію та простоту у використані і був вибраний сервер на Node.js, а також через єдину мову як для сервера так і для клієнта. Для клієнта ж використаний простий та зрозумілий фреймворк React.js.

React.js – це популярна JavaScript бібліотека для побудови інтерфейсів користувача. Вона була розроблена командою Facebook і вперше випущена в 2013 році. За допомогою цього фреймворку розробнику дозволяє створювати

складні інтерфейси користувача з мінімальним кодом та максимальною продуктивністю, підтримую компонентий підхід та віртуальне DOM дерево, що дозволяє React ефективно визначати зміни та мінімізувати маніпуляції з реальним DOM тим самим оптимізовувати ререндер сторінки, при зміні компонентів.

Під час розробки серверної частини, важливо заздалегідь визначити, який функціонал пропонуватиме створений сайт, як будуть передаватися дані, змінюватися значення та як це все взаємодіятиме між собою. Але перш за все потрібно написати реалізацію підє'днання до бази даних PostgreSQL зробити це можливовикористовуючи TypeORM пакунок. Приклад коду під’єднання до бази данних зображено на рис. 3.1.

```
const typeOrmConfigEnvs = async (
  withEntities: boolean,
  configService: ConfigService,
): Promise<TypeOrmModuleOptions> => {
  if (withEntities) {
    return {
      host: configService.get<string>('POSTGRES_HOST'),
      port: configService.get<number>('POSTGRES_PORT_DB'),
      logging: ['error'],
      type: 'postgres',
      entities: ['dist/**/*.entity.{ts,js}'],
      migrations: ['dist/migration/**/*.{ts,js}'],
      subscribers: ['dist/**/*.subscriber.{ts,js}'],
      database: configService.get<string>('POSTGRES_DB'),
      username: configService.get<string>('POSTGRES_USER'),
      password: configService.get<string>('POSTGRES_PASSWORD'),
      ssl: getSSLConfig(configService.get<string>('SERVER_MODE')),
      synchronize: true,
    };
  }
  return {
    host: configService.get<string>('POSTGRES_HOST'),
    port: configService.get<number>('POSTGRES_PORT_DB'),
    type: 'postgres',
    database: configService.get<string>('POSTGRES_DB'),
    username: configService.get<string>('POSTGRES_USER'),
    password: configService.get<string>('POSTGRES_PASSWORD'),
  };
};
```

Рисунок 3.1 – Приклад підключення до Бази Даних

Настуним кроком потрібно реалізувати отримання даних для графіків з давачів вологості грунту та температури. Код реалізації наведено в лістингу 3.2 та результат отримання даних на рисунку 3.3.

```
@MessagePattern('sensors')
getTmp(@Payload() data: IMessage, @Ctx() context: MqttContext) : void  {
  this.sensorsRepository.create({
    time: new Date(),
    temperature: data.tmp,
    moisture: data.moisture,
  });
  this.eventEmitter.emit(
    SENSORS.TMP_SEND,
    new TemperatureSendEvent(Number(data.tmp)),
  );
  this.eventEmitter.emit(
    SENSORS.MOISTURE_OF_GROUND,
    new MoistureSendEvent(Number(data.moisture)),
  );
}
```

Рисунок 3.2 – Реалізація отримання даних з датчиків та відправка на клієнтську частину

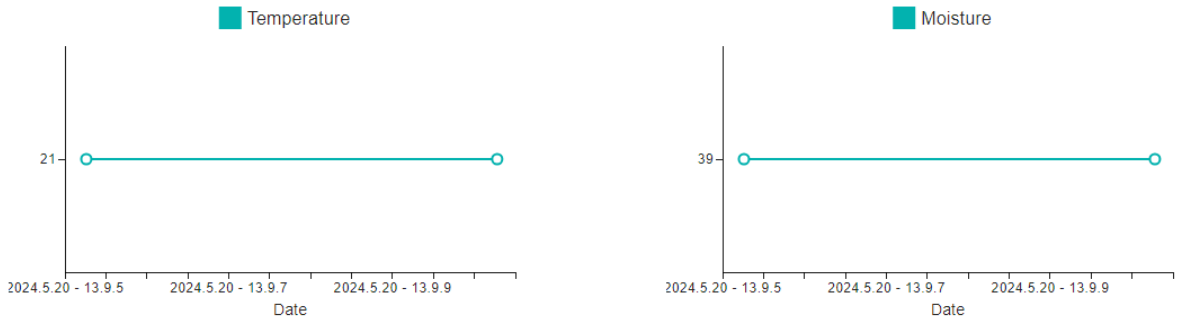

Рисунок 3.3 – Відображення отриманих даних з датчиків на клієнті

Важливий крок, потрібно написати головний функціонал системи, що відрізняє її від інших, а саме створення та редагування технологічних карт рослин. Для цього нам потрібно задати такі атрибути як: назву, опис, повторюваність - необхідно для роботи режиму по таймеру (кожного дня,

кожного тижня, кожного місяця), день повторення, година повторення, час ключення УФ-лампи, необхідна вологість грунту, час роботи помпи з органічною водою/простою. Інтерфейс створення зображено на рисунку 3.4 та реалізація серверної частини на рисунку 3.5.

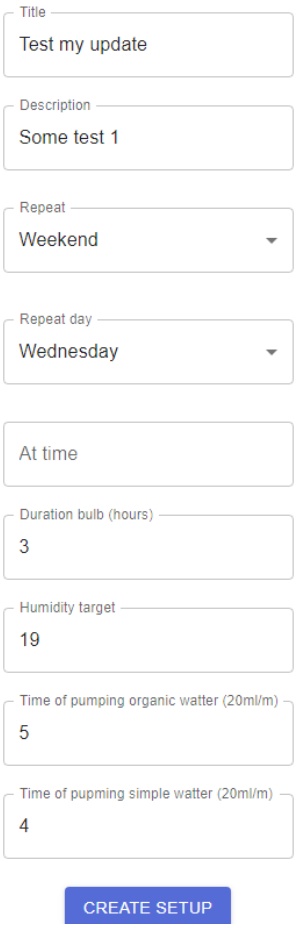

Рисунок 3.4 – Відображення створення за заповнення необхідних атрибутів

```
public async createRoutine(createRoutineDto: CreateRoutineDto) : Promise<void> {
  try {
    await this.routineRepository.createOrUpdate(createRoutineDto);
  } catch (e) {
    throw new BadRequestException( objectOrError: 'Db error');
  }
}
```

Рисунок 3.5 – Код створення технологічної карти рослини

Останній крок, це написання функціоналу вибирання режиму та технологічної карти рослини. Перш за все, нам потрібно вибрати один з трьох

режимів та вибір технологічної карти рослини, реалізований інтерфейс зображений на рисунку 3.6. З серверної частини потрібно здійснити опублікування за топіком необхідних атрибутів до MQTT Брокера, реалізація якого наведена в лістину 3.7.

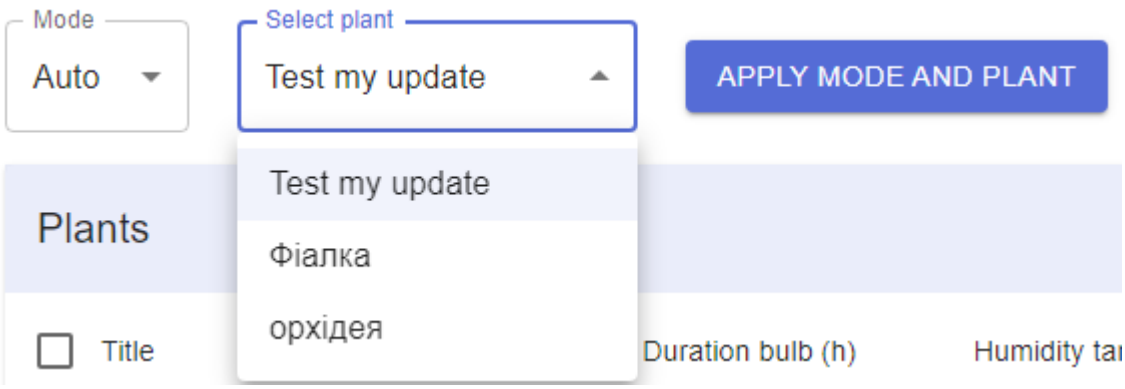

Рисунок 3.6 – Інтерфейс вибору режиму та технологічної карти рослини

```
public async selectRoutine(selectRoutineAndModeDto: SelectRoutineAndModeDto) : Promise<void> {
const { id : string , mode : ModeEnum } = selectRoutineAndModeDto;
const routine : RoutineEntity = await this.routineRepository.getById(id);
if (!routine) {
  throw new ConflictException( objectOrError: 'Not found');
}
await this.routineRepository.updateManyNotSelect();
await this.routineRepository.createOrUpdate( payload: {
  ...routine,
  is_selected: true,
  selected_mode: mode,
});
await this.routineRepository.createOrUpdate( payload: {
  ...routine,
  is_selected: true,
  selected_mode: mode,
});
if (mode !== ModeEnum.TIMER) {
  const schedule : boolean = this.schedulerRegistry.doesExist( type: 'cron', name: 'schedule');
  if (schedule) {
    this.schedulerRegistry.deleteCronJob( name: 'schedule');
  }
}
if (mode === ModeEnum.TIMER) {
  return this.createSchedule(routine);
}
this.mqttService.sendSetup( routine: {
  duration_bulb: routine.duration_bulb,
  humidity_target: routine.humidity_target,
  mode,
});
```

Рисунок 3.7 – Код опублікування в MQTT Брокер повідомлення

Після написання цих трьох функціональностей, можна переглянути повну реалізацію системи на рисунку 3.8.

Рисунок 3.8 – Кінцевий інтерфейс системи

3.3 Розробка методики перевірки, функціонування (контролю, випробування) системи

Етап тестування є критично важливим при створенні пристрою, програмного додатку, електричної схеми тощо.

Основні аспекти тестування:

– дозволяє перевірити правильність реалізації усіх вимог проєкту;

– демонструє, що створений проєкт відповідає специфікаціям і визначеним вимогам до продуктивності;

– гарантує зацікавленим сторонам, що проєкт працюватиме відповідно до очікувань;

– допомагає виявити та усунути дефекти й помилки, включаючи критичні;

– перевіряє належну інтеграцію та взаємодію всіх компонентів системи.

Перш за все, треба перевірити чи всі елементи системи підключені до Arduino коректно, усі підключення потрібно проводити згідно з фунцкіональною схемою на рисунку 2.1.

При подачі живлення світлодіоди на платі почнуть миготіти, що означає готовність системи до роботи, а також має світитись червоний світлотіод на ESP-01S сигналізуючи про з'єднання з Wi-Fi і готовність системи для передачі даних, як на рисунку 3.9.

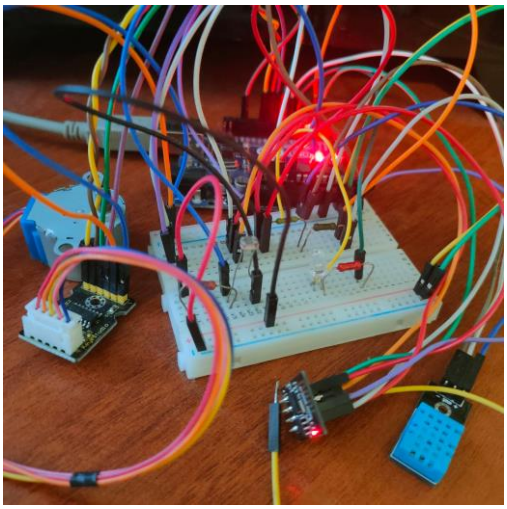

Рисунок 3.9 – Готовність системи

Наступним кроком треба переконатись чи є доступ до WI-FI, сервера та до MQTT Брокера, якщо все гаразд з мережевим підключенням отримаємо таке повідомлення в Serial Port, на рисунку 3.10.

```
16:24:52.265 -> Attempting MQTT connection..
16:24:52.265 -> MQTT Client Connected
16:24:52.298 -> Subscribe "setup" ok
```

Рисунок 3.10 – Успішний результат підключення до мережі

Під час першого запуску, треба вибрати режим роботи та технологічну карту або створити нову для нової рослини. За допомогою датчика вологості та датчика температури, ми отримуємо актуальні метрики та зберігаємо їх. Також користувач може включити або ж виключити УФ-лампу та підкачку органічної або простої води.

У системі можуть бути такі види несправностей. Одна з яких, дані на сайті не відображаються коректно – перевірте правильність з'єднання та справність усіх елементів системи, перевірте чи датчик справний або замініть його на аналогічний. Якщо з підключенням все гаразд, спробуйте перезапустити систему.

Інформація на сайті не оновлюються – слід перевірити чи правильно вказаний пароль та логін для з’єднання з WI-FI мережею і чи є самий інтернет. Переконатись у працездатності сервера та чи надходять дані на MQTT Брокер.

Контролер не реагує на зміни стану кнопок – необхідно перевірити правильність підключення та справність мікроконтролера і модуля ESP, вимірявши напругу на їхніх виходах за допомогою тестера, знову ж таки перевірити чи надходяться дані на MQTT Брокер, можна це зробити за допомогою програми MQTTX, створити нове зєднання та підписатись на отримання декількох тем, наприклад: sensors, waters, setup, як це зображено на рисунку 3.11.

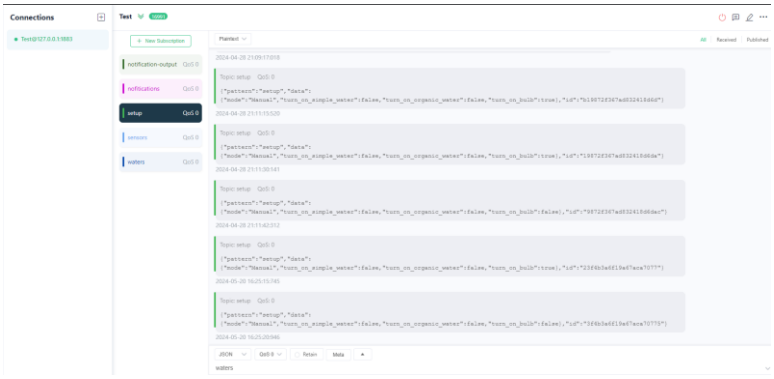

Рисунок 3.11 – Перегляд находженних даних за допомогою програми MQTTX

Також розглянем тест кейси успішної роботи системи, такі як, вручне включення лампи – після вибору ручного режиму та натискання на кнопки “Turn on Bulb” зображеного на рисунку 3.12.

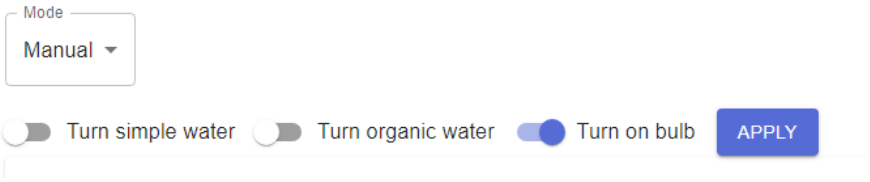

Рисунок 3.12 – Інтерфейс включення лампи

Відбувається успішне включення лампи, як це зображено на рисунку 3.13.

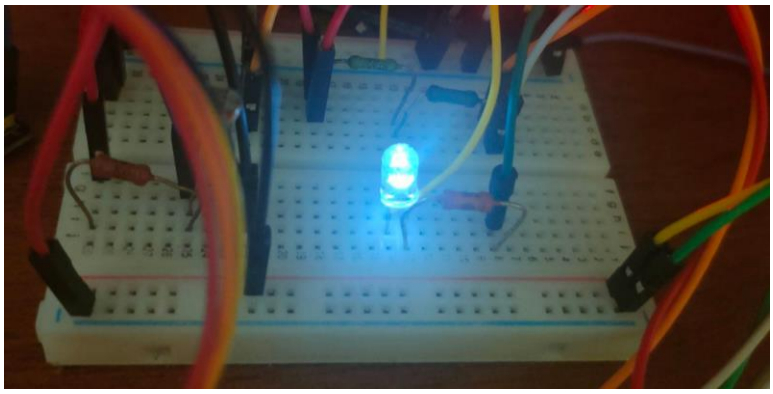

Рисунок 3.13 – Успішне включення лампи

Вручне включення помпи з водою – після вибору ручного режиму та натискання на кнопки “Turn on simple water” або “Turn on organic water” зображеного на рисунку 3.14. відбувається включення помпи з простою водою або органічною відповідно зображено на рисунку 3.15.

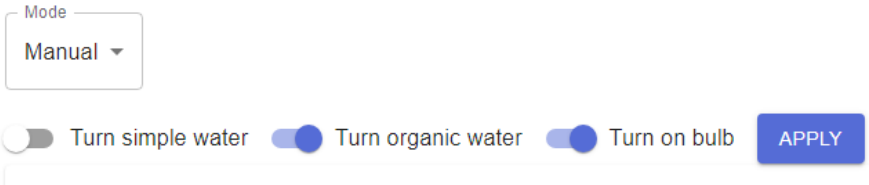

Рисунок 3.14 – Вибір ручного режиму та включення помпи з водою

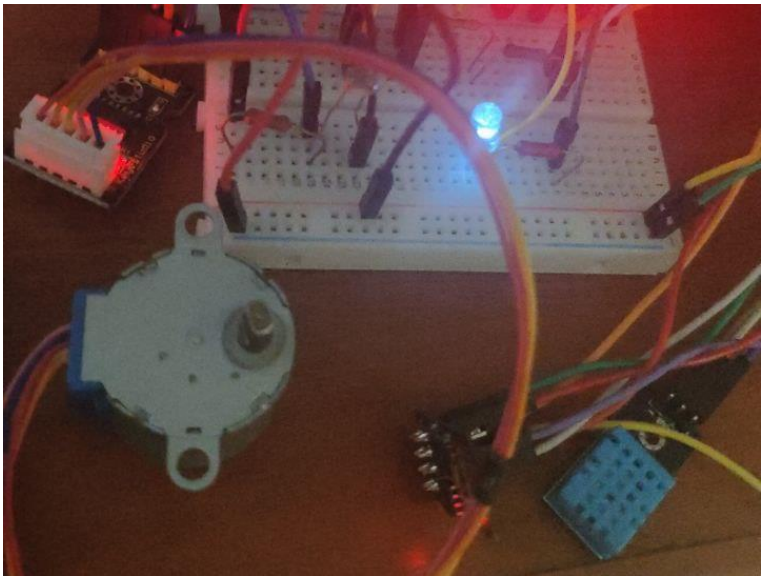

Рисунок 3.15 – Успішне включення помпи

Перевірка на наявності води у ємкостях з водою – при недостатній кількості води, на клієнтську частину відсилається повідомлення для перевірки води, зображено на рисунку 3.16.

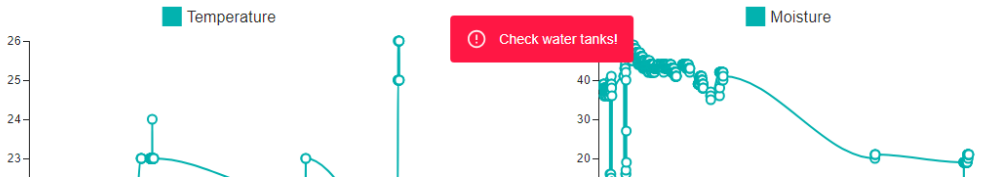

Рисунок 3.16 – Успішне отримання повідомлення

3.4 Розробка прототипу системи

Під час розробки комп'ютеризованої системи був створений також робочий прототип, який наочно демонструє функціональність пристрою. Прототип дозволяє керувати такими аспектами, як освітлення і полив, а також збирати дані про вологість ґрунту і температуру. Для реалізації цих функцій було встановлено датчики, виконавчі елементи та керуючі пристрої, розташовані відповідно до функціональної схеми системи.

Прототип складається з різних елементів, розташованих відповідно до

їхньої функціональної ролі. На макетні платі встановлені датчики, що забезпечують збір даних, замість датчиків вологості грунту і рівня води, які б мали бути використані, були застосовані аналоги, такі як датчики освітлення та датчик вологості повітря. Після зняття показників дані передаються на ESP-01S через протокол UART і в подальшому робиться публікація повідомлення у потрібну тему в MQTT брокер.

Збоку використанний один моторчик зміного струму який імітує помпу, а також замість лампи використаний світлодіод, для простої демонстрації виконання логіки системи.

Для роботи виконавчих елементів та давачів, підключення здійснюється як і у ТЗ. Такий принцип прототипування дозволяє візуально спостерігати за активністю роботи елементів та всієї системи, а також контролювати їхню активацію та вимкнення. На рисунку 3.17 зображено прототип у запущеному стані.

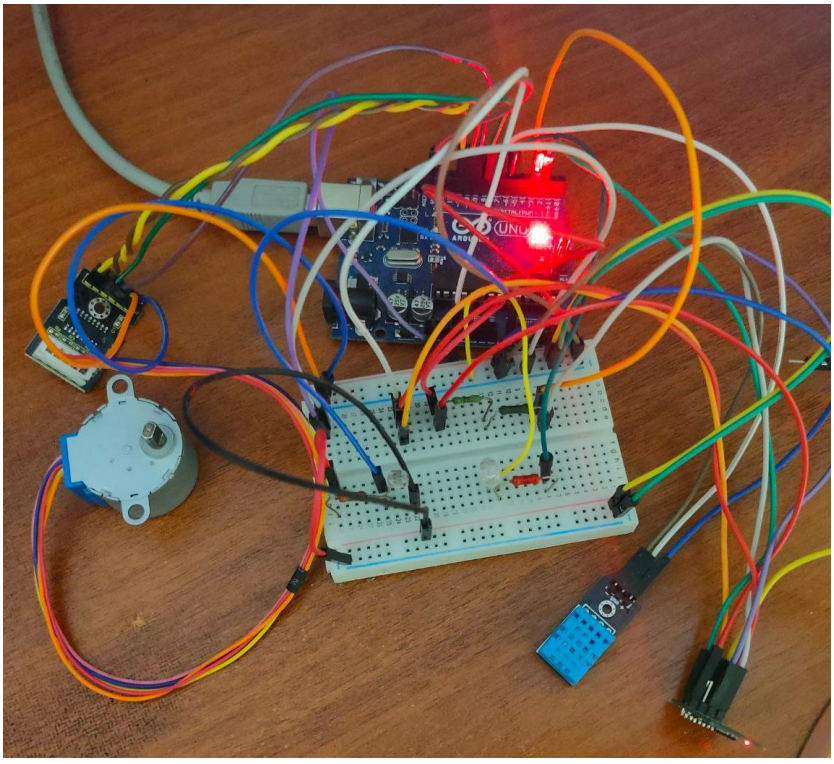

Рисунок 3.17 – Прототип системи у запущеному стані

Будь-який прототип є обов'язковим при побудові КС, оскільки він дає змогу розробнику тестувати всі ключові функції в реальному часі.

Цей прототип ілюструє потенціал автоматизації та вдосконалення управління процесами в практичних ситуаціях. Прототип слугує основою для подальших досліджень і вдосконалення системи догляду за рослинами, за допомогою технологічних карт, спрямованих на покращення вегетації та врожайності рослин, енергоефективності та мінімізації участі людини.

Представлений прототип включає датчики, виконавчі елементи та керуючі пристрої, розташовані відповідно до їхніх функцій. У кваліфікаційній роботі запропоновано підхід, який дозволяє багаторазово використовувати цю конструкцію, що дає можливість обслуговувати будь-яку кількість рослин та будь який вид рослин завдяки технологічним картам за допомогою одного пристрою.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Домедична допомога при отруєнні пестицидами та агрохімікатами

При використанні комп’ютеризованої системи вирощування рослин за технологічними картами, працівники повинні знати про небезпеку отруєнням пестицидами та агрохімікатами.

Пестициди – це токсичні речовини хімічного або біологічного походження, призначені для знищення шкідливих мікроорганізмів, гризунів, бур’янів, чагарників, засмічуючих видів риб. Залежно від виробничого призначення пестициди поділяють на:

– інсектициди – засоби проти шкідливих комах;

– фунгіциди – засоби проти збудників грибкових хвороб рослин;

– немациди – проти молюсків, слизняків;

– гербіциди – проти бур’янів, для протруювання насіння.

В залежності від хімічної структури пестициди поділяються на фосфорорганічні, хлорорганічні, ртутьорганічні сполуки, препарати, які містять миш’як, мідь. Більшість з них токсичні для людини.

Фосфорорганічні сполуки (ФОС). Найбільшого практичного застосування у вигляді інсектицидів і акарицидів знайшли карбофос, хлорофос, дихлофос, фосфамід, метилмеркаптофос, авенін.

ФОС потрапляють в організм людини усіма відомими шляхами.

Розрізняють три ступені гострої інтоксикації:

– легкий ступінь супроводжується головним болем, головокружінням, швидкою втомою, нудотою, блюванням, болем в животі, звуженням зіниць (міоз), зниженням артеріального тиску, сповільненням пульсу, слинотечею, посиленим потовиділенням;

– середній ступінь характеризується посиленням головного болю, головокружінням, задишкою, проносом, депресією;

– важкий ступінь супроводжується непогамовним блюванням, мимовільними сечовипусканням та дефекацією, судомами м'язів кінцівок і тіла, набряком легень з наступною зупинкою дихання.

Хронічному отруєнню властиві: головний біль, особливо у скронях, послаблення пам'яті, роздратованість, потовиділення, розлади сну, галюцинації, депресія.

Перша допомога при гострому отруєнні ФОС:

– при потраплянні на шкіру – промити її 10% розчином питної соди або 5% розчином нашатирного спирту;

– при потраплянні в очі – промити їх водою;

– при потраплянні через рот – промити шлунок 2% розчином питної соди, застосувати активоване вугілля.

Хлорорганічні сполуки (ХОС). До цієї групи входять дихлордифенілтрихлорметилметан (ДДТ), хлорбензол, гексахлорциклогексан, гептахлор, поліхлоркамфен. ХОС блокують дихальний фермент, через що виникає киснева недостатність тканин. При потраплянні ХОС на шкіру виникають дерматити; при вдиханні – ознаки ураження дихальної системи, аж до пневмонії; при попаданні через рот – ознаки гострого отруєння: відчуття втоми, слабкості в кінцівках, потовиділення, втрата апетиту, нудота, блювання, біль в ділянці шлунку. У важких випадках – порушення свідомості, судоми, аритмія серцевої діяльності, зниження артеріального тиску. Через 1-2 години може наступити смерть.

Перша допомога: старанно миють відкриті ділянки шкіри водою з милом, промивають шлунок, дають випити активоване вугілля, призначають сольове проносне.

Препарати, що містять мідь (мідний купорос, хлорокис міді, бордоська суміш),мають загальноотруйну дію з переважанням токсичного впливу на капіляри. У разі потрапляння їх в травний тракт з'являється металевий присмак

у роті, слинотеча, біль у животі, блювання із синьо-зеленим забарвленням блювотних мас, зниження температури тіла, жовтяниця. У важких випадках порушується дихання та серцева діяльність, наступає коматозний стан і смерть.

Якщо отрута потрапила через дихальні шляхи, то виникає тривала лихоманка, кашель з виділенням зеленуватого мокротиння, носові кровотечі, біль в ділянці шлунку, пронос.

При хронічному отруєнні виникають біль в ділянці шлунку, слинотеча, пронос, кашель, першіння в горлі, зеленувате забарвлення шкіри обличчя та волосся.

Перша допомога: промивання шлунку 0,1% розчином перманганату калію, вживання молока, активованого вугілля.

Миш'як та його сполуки (арсенат натрію, арсенат кальцію, арсенат натрію) використовують проти шкідників сільськогосподарських культур.

Розрізняють три форми гострих отруєнь:

– шлунково-кишкова форма – супроводжується відчуттям металевого присмаку в роті, печінням у зіві, болем у животі, блюванням, сильним проносом (випорожнення у вигляді рисового відвару), зниженням температури тіла, судомами у литках;

– паралітична форма – розвивається при попаданні в організм великої кількості миш'яку – при цьому виникають загальна слабкість, судоми, непритомність, параліч дихального і серцево-рухового центрів;

– форма подразливої дії – при попаданні миш'яку на слизові оболонки очей, верхніх дихальних шляхів. При цьому виникають чхання, кашель, набряк слизових оболонок очей.

Агрохімікати – це органічні, мінеральні і бактеріальні добрива, регулятори росту, що застосовуються для підвищення родючості грунтів, урожайності сільськогосподарських культур. Серед агрохімікатів найбільш вживані азотні, фосфорні та калійні мінеральні добрива.

При отруєнні азотними добривами виникають подразнення верхніх дихальних шляхів, шкіри, слизових оболонок очей.

Калійні добрива викликають нежить, ураження шкіри; фосфорні – запалення верхніх дихальних шляхів, носові кровотечі.

Після роботи з агрохімікатами слід вимити відкриті ділянки тіла водою з милом, прополоскати ніс, рот.

4.2 Вимоги до режимів праці і відпочинку при роботі з ВДТ

При використанні комп’ютеризованої системи вирощування рослин за технологічними картами, потрібно враховувати вимоги до режимів праці і віпдочинку, пов'язаної з використанням ВДТ ЕОМ і ПЕОМ, для збереження здоров'я оператора, запобігання професійним захворюванням і підтримки працездатності передбачаються внутрішньозмінні регламентовані перерви для відпочинку.

Внутрішньозмінні режими праці і відпочинку містять додаткові нетривалі перерви в періоди, що передують появі об'єктивних і суб'єктивних ознак стомлення і зниження працездатності.

При виконанні робіт, що належать до різних видів трудової діяльності, за основну роботу з ВДТ слід вважати таку, що займає не менше 50% робочого часу[14]. Впродовж робочої зміни передбачаються:

- перерви для відпочинку і вживання їжі (обідні перерви);
- перерви для відпочинку і особистих потреб (згідно з трудовими нормами);
- додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

За характером трудової діяльності розрізняють три професійні групи, згідно з діючим класифікатором професій (ДК-003-95 і Зміна N1 до ДК-003-95):

а) розробники програм (інженери-програмісти) виконують роботу переважно з відеотерміналом та документацією при необхідності інтенсивного обміну інформацією з ЕОМ і високою частотою прийняття рішень. Робота характеризується інтенсивною розумовою творчою працею з підвищеним

напруженням зору, концентрацією уваги на фоні нервово-емоційного напруження, вимушеною робочою позою, загальною гіподинамією, періодичним навантаженням на кисті верхніх кінцівок. Робота виконується в режимі діалогу з ЕОМ у вільному темпі з періодичним пошуком помилок в умовах дефіциту часу;

б) оператори електронно-обчислювальних машин виконують роботу, пов'язану з обліком інформації, одержаної з ВДТ за попереднім запитом, або тієї, що надходить з нього, супроводжується перервами різної тривалості, пов'язана з виконанням іншої роботи і характеризується напруженням зору, невеликими фізичними зусиллями, нервовим напруженням середнього ступеня та виконується у вільному темпі;

в) оператор комп'ютерного набору виконує одноманітні за характером роботи з документацією та клавіатурою і нечастими нетривалими переключеннями погляду на екран дисплея, з введенням даних з високою швидкістю. Робота характеризується як фізична праця з підвищеним навантаженням на кисті верхніх кінцівок на фоні загальної гіподинамії з напруженням зору (фіксація зору переважно на документи), нервово-емоційним напруженням.

Правилами встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні в залежності від характеру праці:

– для розробників програм із застосуванням ЕОМ слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за ВДТ [15];

– для операторів із застосуванням ЕОМ слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години [15];

– для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за ВДТ [15];

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 години [15].

При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин [15].

Для зниження нервово-емоційного напруження, втомлення зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі, робити деякі перерви, використовувати для виконання комплекс вправ, згідно Державних санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98 [15].

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розроблено комп'ютеризовану систему для вирощування рослин за технологічними картами, використовуючи Arduino як основний модуль керування та ESP-01S для обробки інформації. Система оснащена датчиками температури, вологості та рівня води. На основі даних, отриманих з цих датчиків, система керує такими виконавчими механізмами, як насос змінного струму та УФ освітлення. В роботі:

– проаналізовано актуальність обраної теми, технічне завдання та конкурентні системи на українському ринку;

– детально описано та обґрунтовано вибір елементної бази та розроблено принципову схему пристрою;

– створено алгоритм роботи системи з описом ключових функцій програмного коду, а також написаний сайт та сервер для керуванням мікроконтролером;

– підготовлено інструкцію з експлуатації електронного пристрою та методику його тестування і перевірки;

– спроектовано та зібрано прототип системи.

Розроблена система має переваги, які полягають в наступному:

– збільшену обчислювальну потужність і розширені можливості для контролю та управління завдяки використанню Arduino;

– додавання модуля WiFi дозволяє здійснювати контроль на відстані.

– додавання технологічних карт рослин як основна перевага перед аналогами.

Користувач може переглядати відповідні параметри через ПК або телефон за допомогою власноруч створеного сайту, а також зберігати такі метрики, як температура та вологість ґрунту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Осухівська Г.М., Тиш Є.В., Луцик Н.С., Паламар А.М. Методичні вказівки до виконання кваліфікаційних робіт здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 123 «Комп'ютерна інженерія» усіх форм навчання. Тернопіль, ТНТУ. 2022. 28 с.

2. Паламар М.І., Стрембіцький М.О., Паламар А.М. Проектування комп'ютеризованих вимірювальних систем і комплексів. Навчальний посібник. Тернопіль: ТНТУ. 2019. 150 с.

3. Оконський М.В., Лупенко С.А., Паламар А.М. Інформаційно-вимірювальна система для контролю метеорологічних параметрів на основі Інтернету речей. Матеріали IX науково-технічної конференції "Інформаційні моделі, системи та технології" Тернопільського національного технічного університету імені Івана Пулюя (Тернопіль, 8–9 грудня 2021 року), Тернопіль: ТНТУ, 2021. С. 118.

4. Yatsyshyn V., Pastukh O., Palamar A., Zharovskyi R. Technology of relational database management systems performance evaluation during computer systems design. Scientific Journal of TNTU (Tern.). Vol. 109. No 1. 2023. pp. 54–65.

5. Yatsyshyn V., Pastukh O., Zharovskyi R., Shabliy N. Software tool for productivity metrics measure of relational database management system. Mathematical Modeling. No 1 (48). 2023. P. 7-17.

6. Arduino Uno. URL: https://www.arduino.cc/ (дата звернення: 10.06.2024).

7. ESP-01S характеристика та принцип роботи. URL: https://www.alldatasheet.com/datasheet-pdf/pdf/1148023/ESPRESSIF/ESP32.html (дата звернення: 10.06.2024).

8. Датчик вологості грунту YL-69. URL: https://imrad.com.ua/userdata/modules/productFiles/P013t9pW_YL-69.pdf (дата звернення: 10.06.2024).

9. Датчик рівня води T1592. URL: https://www.alldatasheet.com/datasheetpdf/pdf/19243/PHILIPS/TDA1592.html (дата

звернення: 10.06.2024)

10. Node.js. URL: https://nodejs.org/en (дата звернення: 10.06.2024).

11. What is MQTT. URL: https://aws.amazon.com/what-is/mqtt (дата звернення: 10.06.2024).

12. React.js.URL: https://uk.legacy.reactjs.org (дата звернення: 10.06.2024).

13. Nest.js. URL: https://nestjs.com/ (дата звернення: 10.06.2024).

14. Роз'яснення щодо набуття права на перерву у зв'язку з роботою за комп'ютером. URL: https://zakon.rada.gov.ua/rada/show/v0455282-07 (дата звернення: 10.06.2024).

15. Про затвердження Примірної інструкції з охорони праці під час експлуатації електронно-обчислювальних машин. URL: https://zakon.rada.gov.ua/rada/show/v0443810-13 (дата звернення: 10.06.2024).

ДОДАТОК А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя
Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра комп'ютерних систем та мереж

«Затверджую»
Завідувач кафедри КС
_____ Осухівська Г.М.
«____»_____ 2024 р.

КОМП'ЮТЕРИЗОВАНА СИСТЕМА ВИРОЩУВАННЯ РОСЛИН ЗА ТЕХНОЛОГІЧНИМИ КАРТАМИ

ТЕХНІЧНЕ ЗАВДАННЯ
на __4__ листках

На здобуття освітньо-кваліфікаційного рівня *бакалавр*
Напрям *123 Комп'ютерна інженерія*
Спеціальність *123 Комп'ютерна інженерія*

«УЗГОДЖЕНО»
Керівник кваліфікаційної роботи
_____ к.н.т., доц. Жаровський Р.О.
«____»_____2024 р.

«ВИКОНАВЕЦЬ»
Студент групи СІс-42
_____ Демчан Н.І.
«____»_____2024 р.

Тернопіль 2024

1. Назва та підстава для виконання роботи.

1.1. Комп’ютеризована система вирушування рослин за технологічними картами.

1.2. Підставою для виконання кваліфікаційної роботи бакалавра (КРБ) є Наказ по Університету (№ 4/7-468 від 26.04.2024 р.)

2. Виконавець.

2.1. Студент групи СІс-42 кафедри КС Тернопільського національного технічного університету ім. І. Пулюя Демчан Назар Іванович.

3. Мета роботи.

3.1. Метою роботи є розробити структуру та апаратне забезпечення комп’ютеризованої системи вирущування рослин за технологічними картами.

4. Склад виробу.

4.1. До складу виробу повинні входити:

1) мікроконтролер;
2) wifi модуль;
3) давач вологості грунту;
4) давач вологості/температури повітря;
5) давач рівня води;
6) модуль реле;
7) два NPN - транзистори ;
8) два DC - двигуна;

5. Технічні вимоги.

5.1. Вимоги по призначенню.

5.1.1. Вбудована система повинна мати наступні параметри:

1) Діапазон вимірюваної температури, С -40 … +125
2) Діапазон вимірюваної відносної вологості, % 0…100
3) Точність вимірювання температури, °С ±0,1
4) Точність вимірювання вологості, % ±0,5

5.1.2. Система повинна живитись напругою постійного струму, В +5 ±1.5

5.2. Вимоги до умов експлуатації:

5.2.1. По умовам експлуатації виріб повинен відповідати вимогам ДСТУ 15150 для УХЛ4.1

5.2.2. Температура експлуатації від 0 до +50^{o}С

5.2.3. Відносна вологість до 100% при t=25^{o}С

5.3.Конструктивні вимоги.

5.3.1. Конструювання корпусу приладу в КРБ не передбачено.

5.3.2. Для побудови системи має бути використана сучасна компоненти з можливістю поверхневого монтажу друкованого вузла

5.3.3. При побудові системи необхідно передбачити наявність роз’ємів живлення і обміну даними.

5.3.4. Габаритні розміри при макетуванні, мм, не більше:

довжина 200
ширина 200
висота 150

5.3.5. Маса макету, кг, не більше 2

5.3.6. Конструкція макету повинна забезпечувати доступ до всіх комплектуючих виробів при тестуванні.

5.4. Вимоги до надійності.

5.4.1. Система повинна відповідати вимогам ДСТУ 2862-94.

5.4.2. Наробка на відмову, не менше 4500 год.

5.5. Вимоги метрології.

5.5.1. Вимірювання параметрів системи при моделюванні повинно виконуватись на універсальних вимірювальних приладах.

6. Економічні показники.

6.1. Собівартість системи повинна бути не більше 2300 грн.

7. Вимоги до документації.

7.1. Конструкторська документація повинна відповідати вимогам ЄСКД та ДСТУ.

7.2. До складу документації повинно входити:

1) ПЗ
2) Структурна схема
3) Функціональна схема
4) Блок схема алгоритму роботи
5) Діаграма послідовностей роботи

8. Стадії та етапи розробки КРБ

8.1 Стадії та етапи виконання КРБ наведенні в таблиці 1.

Таблиця 1

№ з/п	Назва етапів роботи	Термін виконання етапів роботи
1	*Розробка та затвердження технічного завдання*	
2	*Аналіз технічного завдання*	
3	*Аналіз вимог та принципів організації систем вирощування рослин за технологічними картами*	
4	*Проєктування системи вирощування рослин за технологічними картами*	
5	*Розробка схем і програмного забезпечення системи вирощування рослин за технологічними картами*	
6	*Розробка інструкцій з використання системи*	
7	*Безпека життєдіяльності, основи охорони праці*	
8	*Оформлення кваліфікаційної роботи*	
9	*Попередній захист кваліфікаційної роботи*	
10	*Захист кваліфікаційної роботи*	

9. У дане ТЗ можуть вноситись зміни по узгодженню сторін.

ДОДАТОК Б

Перелік елементів функціональної схеми

Позн.	Найменування	К-сть	Примітка
	Мікроконтролери		
DM1	ESP8266-1, LILYGO	1	
DM3	Arduino UNO R3	1	
	Модулі		
DM2	DHT22, ROBOTDYN	1	
AM1	YL-69, HONEY	1	
DM4	SRD05VDCSLC	1	
AM2, AM3	T1592 , EKITS	2	
DM5, DM6	RS-360SH-16260	2	
	Резистори		
R1, R2	МЛТ-0.125 1 кОм, YAGEO	2	
R3	МЛТ-0.125 220 Ом, YAGEO	1	
	Транзистори		
Q1, Q2, Q3	2N5551	3	
	Роз'єми		
XS1	Клема блока живлення 5 В	1	

Змн.	Арк.	№ докум.	Підпис	Дата	КС КРБ 123.314.00.00 ПЕ		
Розроб.		Демчан Н. І.			Комп'ютеризована система вирощування рослин за технологічними картами Перелік елементів	Літ.	Арк. 59 / Аркушів 18
Перевір.		Жаровський Р.О.					
Реценз.		Мудрик І.Я.				ТНТУ, каф. КС, гр. СІс-42	
Н. Контр.		Луцик Н.С.					
Затверд.		Осухівська Г.М.					

ДОДАТОК В

Лістинг коду системи

```
#include <SoftwareSerial.h>
#include <Stepper.h>
#include <DHT.h>
#include <ArduinoJson.h>
#define STEPS 2038

#define DHTTYPE DHT11
#define dthPin 7
Stepper stepper(STEPS, 8, 9, 10, 11);
DHT sens(dthPin, DHTTYPE);

#define photo A0
#define moisture A2
#define dht2 A3
#define dht3 A4

#define LED_PIN 6

int photo_value;
int tmp_value;
int humidity_value;

String mode = "";
bool turnOnBulb;

bool turnOnOrganic;
bool turnOnSimple;

bool isWater = true;



String MIN_TMP;
String MIN_MOISTURE;
```

```
int humidityValueTarget;
int humidityValueTargetOrganic;
unsigned long timerTarget;
unsigned long currentMillis = 0;

unsigned long rememberTimeSimple = 0;
unsigned long rememberTimeOrganic = 0;
unsigned long rememberTimeBulp = 0;

unsigned long durationBulp = 0;
unsigned long durationSimple = 0;
unsigned long durationOrganic = 0;

SoftwareSerial serialEsp(2, 3);
const long onDuration = 300;

void resetAll() {
  turnOnBulb = false;
  turnOnOrganic = false;
  turnOnSimple = false;
  digitalWrite(LED_PIN, LOW);
  stepper.step(0);
}
void sendData(int interval) {
  static long next_time = 0;          // remember the next_time for
next action (note static)
  if (millis() < next_time) return;  // if time has not arrived,
return
  next_time += interval;              // else update next_time by
period

  StaticJsonDocument<200> doc;

  doc["topic"] = "sensors";
  doc["moisture"] = photo_value;
  doc["tmp"] = tmp_value;
```

```
  char out[128];
  serializeJson(doc, out);
  serialEsp.write(out);

  Serial.print("Send to esp photo:");
  Serial.println(photo_value);
  Serial.print("Send to esp tmp:");
  Serial.println(tmp_value);

  Serial.print("Send to esp humidity:");
  Serial.println(humidity_value);
}


void sendDataNoWaters(int interval) {
  static long next_time = 0;         // remember the next_time for
next action (note static)
  if (millis() < next_time) return;  // if time has not arrived,
return
  next_time += interval;
  if (photo_value < 10) {
    isWater = false;
    StaticJsonDocument<200> doc;
    doc["topic"] = "waters";
    doc["is_water"] = false;
    char out[128];
    serializeJson(doc, out);
    serialEsp.write(out);
  }

  if (photo_value > 10) {
    isWater = true;
    StaticJsonDocument<200> doc;
    doc["topic"] = "waters";
    doc["is_water"] = true;
    char out[128];
    serializeJson(doc, out);
```

```
    serialEsp.write(out);
  }
}

void setup() {
  stepper.setSpeed(5);
  sens.begin();

  pinMode(photo, INPUT);

  digitalWrite(dthPin, HIGH);

  pinMode(LED_PIN, OUTPUT);

  Serial.begin(19200);
  serialEsp.begin(19200);
}
void loop() {
  tmp_value = sens.readTemperature();
  humidity_value = sens.readHumidity();
  int valRAW_photoresistor = analogRead(photo);
  photo_value = map(valRAW_photoresistor, 0, 1023, 0, 100);

  sendDataNoWaters(6000);
  sendData(5000);

  if (serialEsp.available() > 0) {
    String incomingString = serialEsp.readString();
    Serial.println("Received String TO ARDUINO FROM ESP: " +
incomingString);

    DynamicJsonDocument doc(1024);

    deserializeJson(doc, incomingString);


    mode = (const char*)doc["data"]["mode"];
```

```
    if (mode == "Auto") {
      resetAll();
      rememberTimeSimple = 0;
      rememberTimeOrganic = 0;
      rememberTimeBulp = 0;
      humidityValueTarget = 0;
      humidityValueTarget = doc["data"]["humidity_target"];
      humidityValueTargetOrganic =
doc["data"]["humidity_target_organic"];
      durationBulp = doc["data"]["duration_bulb"];
      durationBulp = durationBulp * 3600000;
    }

    if (mode == "Manual") {
      resetAll();
      rememberTimeSimple = 0;
      rememberTimeOrganic = 0;
      rememberTimeBulp = 0;
      humidityValueTarget = 0;
      humidityValueTargetOrganic = 0;
      turnOnBulb = doc["data"]["turn_on_bulb"];
      turnOnOrganic = doc["data"]["turn_on_organic_water"];
      turnOnSimple = doc["data"]["turn_on_simple_water"];
    }
    if (mode == "Timer") {
      resetAll();
      rememberTimeSimple = 0;
      rememberTimeOrganic = 0;
      rememberTimeBulp = 0;
      humidityValueTargetOrganic = 0;
      durationBulp = doc["data"]["duration_bulb"];
      durationBulp = durationBulp * 3600000;
      durationSimple = doc["data"]["duration_simple"];
      durationSimple = durationSimple * 60000;
      durationOrganic = doc["data"]["duration_organic"];
      durationSimple = durationOrganic * 60000;
    }
```

```
  Serial.println("obj-mode: " + mode);
  Serial.println("turnOnOrganic: " + turnOnOrganic);
  Serial.println("durationBulp: " + durationBulp);
  Serial.println("humidityValueTarget: " + humidityValueTarget);
}
if (mode == "Auto") {
  if ((millis() - rememberTimeBulp) <= durationBulp) {
    rememberTimeBulp = millis();
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(LED_PIN, LOW);
  }
  if (humidity_value < humidityValueTarget && isWater) {
    //simple
    stepper.step(1038);
  } else {
    stepper.step(0);
  }
  if (humidity_value < humidityValueTarget && isWater) {
    stepper.step(2038);
  } else {
    stepper.step(0);
  }
  // stepper.step(2038);
}
if (mode == "Timer") {
  if ((millis() - rememberTimeBulp) <= durationBulp) {
    rememberTimeBulp = millis();
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(LED_PIN, LOW);
  }
  if ((millis() - rememberTimeSimple) <= durationSimple) {
    if (isWater) {
      rememberTimeSimple = millis();
      stepper.step(2038);
    }
```

```
    } else {
      stepper.step(0);
    }
    if ((millis() - rememberTimeOrganic) <= durationOrganic) {
      if (isWater) {
        rememberTimeOrganic = millis();
        stepper.step(2038);
      }
    } else {
      stepper.step(0);
    }
  }
  if (mode == "Manual") {
    if (turnOnBulb) {
      digitalWrite(LED_PIN, HIGH);
    } else {
      digitalWrite(LED_PIN, LOW);
    }
    if (turnOnOrganic && isWater) {
      stepper.step(1038);
    }
    if (turnOnSimple && isWater) {
      stepper.step(2038);
    }
  }
}
```

Скетч конфігурації ESP-32:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <SoftwareSerial.h>
#define DEBUG true
// Wi-Fi Credentials
#define ssid "ND_2.4"
#define password ""
```

```
WiFiClient client;
// MQTT Broker Details
String device_id = "Device0001";
const char *mqtt_server = "192.168.0.78";
const int mqtt_port = 1883;
const char *mqtt_user = "testing";
const char *mqtt_password = "testing";
const char *mqtt_clientId = "Deivce_Device0001";
const char *topic_publish = "notifications";
const char *topic_publish_sensors = "sensors";
const char *topic_publish_lvl_of_water_1 = "tank_lvl_water_1";
const char *topic_publish_lvl_of_water_2 = "tank_lvl_water_2";

const char *topic_subscribe = "setup";
void callback(char *topic, byte *payload, unsigned int length);
PubSubClient mqtt_client(mqtt_server, mqtt_port, callback, client);
char c;
void setup() {
  Serial.begin(19200);
  Serial.println("\n\nWelcome to SMART HOUSE ESP15\n");
  setup_wifi();
  mqtt_connect();
  delay(3000);
}
void loop() {
  if (Serial.available() > 0) {
    String incomingData = Serial.readString()

    StaticJsonDocument<200> doc;

    deserializeJson(doc, incomingData);

    const char* topic = doc["topic"].as<const char*>();

    char out[128];
    serializeJson(doc, out);
    mqtt_publish((char *)out, (char *)topic);
```

```
  }

  if (!mqtt_client.loop())
    mqtt_connect();
}
void setup_wifi() {
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println("\"" + String(ssid) + "\"");

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void mqtt_connect() {
  // Loop until we're reconnected
  while (!mqtt_client.connected()) {
    Serial.println("Attempting MQTT connection...");
    // Attempt to connect
    if (mqtt_client.connect(mqtt_clientId, mqtt_user,
mqtt_password)) {
      if (mqtt_client.connect(mqtt_clientId)) {
        Serial.println("MQTT Client Connected");
        // Subscribe
        mqtt_subscribe(topic_subscribe);
      }
    } else {
      Serial.print("failed, reconnect=");
      Serial.print(mqtt_client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
```

```
      delay(5000);
    }
  }
}

void mqtt_publish(char *data, char *topic) {
  mqtt_connect();
  mqtt_client.publish(topic, data);
}

void mqtt_subscribe(const char *topic) {
  if (mqtt_client.subscribe(topic))
    Serial.println("Subscribe \"" + String(topic) + "\" ok");
  else
    Serial.println("Subscribe \"" + String(topic) + "\" failed");
}

void callback(char *topic, byte *payload, unsigned int length) {
  String command;
  for (int i = 0; i < length; i++)
    Serial.write((char)payload[i]);
}
```