КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

До захисту допущено

В.о. завідувача кафедри ІСТ

_____ Олексій КОЛЕСНІКОВ

(підпис) (ім'я, ПРІЗВИЩЕ)

"___"_____2021р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

спеціальності 126 «Інформаційні системи та технології»

освітньої програми «Програмні технології інтернет речей»

на тему: Створення системи розумного зрошування у фермерському господарстві "Дісті" для підвищення врожайності.

Виконав (-ла): студент (-ка) __4__ курсу, групи ІР-41

(шифр групи)

Михайло ШУСТОВ_____ _____

(Ім'я, ПРІЗВИЩЕ) (підпис)

Керівник к.т.н доцент Ростислав ЛІСНЕВСЬКИЙ _____

(посада, науковий ступінь, вчене звання, Ім'я, ПРІЗВИЩЕ) (підпис)

Консультант нормо контроль к.т.н доц. Ростислав ЛІСНЕВСЬКИЙ _____

(назва розділу) (посада, вчене звання, науковий ступінь, Ім'я, ПРІЗВИЩЕ) (підпис)

Рецензент _____ _____

(посада, науковий ступінь, вчене звання, науковий ступінь, Ім'я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у кваліфікаційна робота немає запозичень з праць інших авторів без відповідних посилань.

Здобувач освіти _____

(підпис)

Київ - 2021 року

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра: Інформаційних систем та технологій
Освітньо-кваліфікаційний рівень: Бакалавр
Спеціальність: 126 – Інформаційних систем та технологій
Програма: Програмних технологій інтернет речей

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
Д.т.н., Колесніков О.Є.

_____
“____” _____2021 року

ЗАВДАННЯ
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент: *Михайло ШУСТОВ*
Група: *41*

1. Тема кваліфікаційної роботи: *Створення системи розумного зрошування у фермерському господарстві “Дісті” для підвищення врожайності.*

Затверджена протоколом засідання кафедри ІСТ №16/20 від «09» листопада 2020 р.

Керівник проекту: *к.т.н доцент Ростислав ЛІСНЕВСЬКИЙ*

2. Строк подання студентом готової роботи – «24» червня 2021 р.

3. Цільова установка та вихідні дані до роботи:

Побудувати систему розумного поливу для підприємства. Розробити веб-інтерфейс та Телеграм-бот для моніторингу роботи системи:

- *врахувати виявлені недоліків у аналогічних систем і успішно їх нівелювати;*
- *розумне зрощення має працювати на секторах і підтримувати певну вологість;*

До функціоналу проектованої системи можна віднести:

- *логіка зрошування та підтримка вологості в секторі де ростуть рослини;*
- *веб-інтерфейс, що надає змогу переглядати дані з датчиків та роботу системи;*
- *телеграм-бот для зручного перегляду показників датчиків системи;*

4. Зміст роботи:

- *постановка задачі та аналіз рішення;*
- *створення системи з обраного обладнання;*
- *розробка бази даних для розумного зрошування;*
- *створення логіки роботи системи;*
- *розробка веб-інтерфейсу та Телеграм-боту для моніторингу;*

5. Перелік графічних матеріалів (слайдів):

- *дослідження готових рішень розумного зрошування;*
- *побудова системи розумного зрошування;*
- *підбір обладнання для системи;*
- *створення веб-інтерфейсу для перегляду роботи;*
- *створення Телеграм-боту;*

6. Календарний план виконання роботи:

№ з/п	Назва частин роботи	Дати виконання роботи за планом	Виконання роботи за планом
1	Вивчення літературних джерел з розробки розумного зрошування	05.03.2021 – 12.03.2021	Виконано
2	Збір і вивчення матеріалів на базі ТОВ «Дісті-ПРО»	26.03.2021 – 10.04.2021	Виконано
3	Складання розгорнутого плану роботи	11.04.2021 – 13.04.2021	Виконано
4	Підготовка розділу « аналіз та опис сфери застосування системи автоматичного поливу »	14.04.2021 – 17.04.2021	Виконано

5	Підготовка розділу « розробка проекту та створення системи розумного поливу »	18.04.2021 – 20.04.2021	Виконано
6	Підготовка розділу « розробка програмного коду для розумного зрошування та створення додатків для моніторингу. »	26.04.2021 – 05.05.2021	Виконано
7	Оформлення кваліфікаційної роботи	07.05.2021 – 15.05.2021	Виконано
8	Передача кваліфікаційної роботи рецензенту для рецензування	20.05.2021 – 22.05.2021	Виконано
9	Передача кваліфікаційної роботи науковому керівникові	25.05.2021 – 28.05.2021	Виконано
10	Попередній захист кваліфікаційної роботи	31.05.2021	Виконано
11	Захист роботи	24.06.2021	

Дата видачі завдання «10» листопада 2021 р.

Керівник роботи: *к.т.н доцент кафедри інформаційних систем та технологій Ростислав ЛІСНЕВСЬКИЙ*

_____
(підпис)

Завдання прийняв до виконання:
студент групи ІР-41: Михайло ШУСТОВ

_____

(підпис)

АНОТАЦІЯ

Дипломна робота на тему: « Система розумного зрошування у фермерському господарстві “Дісті” для підвищення врожайності. » складається зі вступу, де описано мету, актуальність, методи дослідження та завдання. Складається з 3 розділ і висновку до них.

У першому розділі досліджується актуальність системи, проходить дослідження готових рішень, типів зв’язку у системах розумного зрошування та дослідження мікроконтролерів.

У другому розділі створюється теоретична частина системи, проходить підбір обладнання для створення системи, підключення частин системи та створення бази даних.

У третьому розділі створюється логіка роботи системи зрошування, програмується веб-інтерфейс та створюється Телеграм-бот для моніторингу за зрошуванням.

Робота складається з 70 сторінок, містить 49 рисунки. При написані роботи було використано 50 інформаційних джерел. Ключові слова: IoT, розумний полив, зрошення, WiFi, веб-інтерфейс, Телеграм-бот, датчики, соленоїд.

SUMMARY

Thesis on the topic: “Smart irrigation system in the farm" Disti "to increase yields.”Consists of an introduction, which describes the purpose, relevance, research methods and objectives. It consists of 3 sections and a conclusion to them.

The first section examines the relevance of the system, examines ready-made solutions, types of communication in intelligent irrigation systems and the study of microcontrollers.

In the second section the theoretical part of the system is created, the equipment for system creation, connection of system parts and creation of a database is selected.

In the third section, the logic of the irrigation system is created, the web interface is programmed and a Telegram bot is created to monitor irrigation.

The work consists of 70 pages, contains 49 drawings. When writing the work was used 50 information sources. Keywords: IoT, smart watering, irrigation, WiFi, web interface, Telegram bot, sensors, solenoid.

ЗМІСТ

ВСТУП

Зараз автоматизовані системи та розумне приладдя грають дуже велику роль у аграрному секторі. Теплиці вимагають розумного кліматичного контролю, полив повинен бути автоматичним та налаштованим під певну рослину та її вегетаційний стан, її особливості і під рівень вологості, який повинен постійно підтримуватися для отримання максимально комфортних умов для рослини, щоб отримати найкращий врожай. Звичайні автоматичні таймери поливу вирішують проблему з нормованою подачею води для рослин, але вони не можуть бути налаштовані під специфічність ґрунту та не здатні підтримувати постійну вологу через погодні умови. Також ефективність таймерів не можливо відслідкувати за допомогою веб-інтерфейсів чи додатків, так як вони не підтримують ці розширення. Саме для цього і створюється система розумного поливу, щоб підтримувати постійне значення вологи згідно з потребами рослин, для економії води при використанні системи, а також для можливостей моніторингу за процесом роботи системи за допомогою графічних інтерфейсів.

Метою цієї роботи є створення системи розумного зрошування, а також дослідження обладнання яке використовується при створенні розумних систем та сфери застосування розумного поливу ,а також створення додатків для моніторингу роботи системи зрошування.

Об’єктом дослідження є система розумного автоматичного поливу, яка працює в теплиці на підприємстві “Дісті”.

Предметом дослідження цієї роботи є IoT впровадження в агросфері, обладнання для створення системи розумного зрошення, датчики вологості та температури.

Методи дослідження: створення системи, експерименти з нею, аналіз існуючих аналогів.

РОЗДІЛ 1. АНАЛІЗ ТА ОПИС СФЕРИ ЗАСТОСУВАННЯ СИСТЕМИ АВТОМАТИЧНОГО ПОЛИВУ

1.1 Аналіз сфери автоматичного поливу

У сучасному світі зберігання води та отримання максимальної кількості врожаю - є дуже важливим завданням. Популяція людей зростає кожного дня і рішення які використовувались століття тому, вже не можна використовувати у сучасному світі. Розумні автоматизовані системи вирішують такі проблеми як - надмірне використання води, точне зрошування, моніторинг поливу, дистанційне керування процесом. Кожна рослина отримує рівно стільки води, скільки їй потрібно, це запобігає гниттю коріння, появі захворювань у рослин та запобіганню росту бур'яну навколо саджанця. Моніторинг поливу допомагає слідкувати за зрошуванням рослин, та аналізувати ситуацію при вирощуванні. За допомогою дистанційного керування можна керувати зрощенням віддалено, цим може займатися одна людина у будь-якій точці світу, також є можливість моніторити декілька об'єктів одночасно, що дає змогу вести господарство, яке не прив'язано до одного місця. Близько 50% води витрачається через перезволоження, спричинене неефективністю традиційних методів зрошення та недосконалих систем. Розумні зрошувальні системи автоматично пристосовують графік поливу та час роботи відповідно до конкретних ландшафтних потреб. Контролери на основі Arduino, Node MCU, NOD irrigation підвищують ефективність використання води при поливі. На відміну від звичайних контролерів зрошення, які працюють за заздалегідь запрограмованим графіком та таймерами, розумні контролери аналізують погоду, стан ґрунту, випаровування та використання води рослинами, щоб автоматично регулювати графік поливу до фактичних умов сектору. Наприклад, по мірі збільшення зовнішньої температури або зменшення кількості опадів, розумна плата враховує специфічні для ділянки зміни, такі як тип ґрунту, рівень опадів тощо, а потім вже сама регулює тривалість або графік поливу.

Розумний полив має явні переваги перед таймерами зрошування. Проводились дослідження, які показували значну економію воду у 30% - 50% при використанні розумного поливу. Тести Асоціації зрошення (IA) та Міжнародного центру водних технологій при Університеті штату Каліфорнія у Фресно показали, що розумні контролери зрошення дозволяють економити на 20 відсотків води більше, ніж звичайні таймери зрошення [1]. В іншому дослідженні випробувано прототип системи таймера-приймача, що складається з таймера поливу, модифікованого для прийому сигналу, що транслюється через супутник. Економія води на відкритому повітрі була розрахована на основі 2-річного використання перед установкою та скоригована з урахуванням погодних умов. Зафіксована середня економія на відкритому повітрі становить 16 відсотків, а також повідомляється, що це становить 85 відсотків потенційної економії, заснованої на еталонному ET. У штаті Вашингтон, штат Пьюджет-Саунд, було проведено дослідження ефективного зрошення водних ресурсів партнерства Saving Water Partnership, коаліції з 24 постачальників води [1]. Економія води була розрахована на основі історичного споживання та зроблено поправки на погодні умови. Заощаджена економія води становила 20 735 галонів на рік на ділянку для секторів з контролерами датчиків дощу та 10 071 галонів на рік на ділянки, що використовують таймери.

1.2 Огляд існуючих розумних рішень

NOD irrigation від IT-LYNX. Контролер NOD irrigation (рис.1.2.1) побудований на архітектурі базового універсального контролера UCPT-3-12. Універсальний програмований логічний контролер UCPT-3-12 (іншими словами, контролер з програмованої логікою) модульної архітектури призначений для вирішення завдань телематики, контролю та управління, а саме:

- управління електронним і електрично-механічними пристроями;
- автоматизація і контроль технологічних процесів;
- збір, обробка, зберігання і передача даних;

Основною відмінною особливістю контролера є можливість тісної інтеграції з веб-орієнтованими платформами, призначеними для обробки і відображення, отриманих з нього даних, а так само, для налаштування та управління контролером. Логіка роботи контролера описується за допомогою мови програмування Lua і виконується з використанням вбудованого в контролер інтерпретатора, що дозволяє адаптувати його під будь-яку предметну область без участі висококваліфікованих інженерів. Цей інноваційний продукт дозволить стежити за кліматичними умовами і мікрокліматом на конкретному об'єкті (в саду, теплиці, ферми, на полі) увесь рік в режимі реального часу з будь-якої точки світу. Все це допоможе ефективно спланувати проведення сільськогосподарських робіт. Контролер призначений для безперервного збору всіх метеорологічних даних і передачі їх на централізований сервер, де всі дані обробляються і стають доступні для користувача в програмній частині. [2]

Рисунок 1.2.1 Nod irrigation

WaterECO від Aifro. Контролер WaterEco від компанії Aifro був розроблений для автоматизації і керування зрошувальними системами. Також компанія Aifro надає доступ до додатка, за допомогою якого можна керувати контролером. WaterEco пропонує можливість легко керувати зрошенням та контролювати його з будь-якої точки світу. Підключення пристроїв та клапанів може здійснюватися через різні мережі, такі як Wi-Fi або хмарні технології. У системі також є датчики дощу, які повідомляють, чи потрібно поливати рослини, чи буде дощ. Ця система має здатність переходити в сплячий режим і вимикати всю систему [3].

BaseStation3200 від BaseLine. Контролер BaseStation3200 створений для автоматизації зрошення на великих площах. Керування відбувається вручну, або за допомогою веб-сайту. Також на сайті можна переглядати всю інформацію яка надходить з датчиків, лічильників та інших приладів в системі.

Він підтримує запатентовану провідну технологію від Baseline, а також традиційні дротові та рішення модернізації. Його вдосконалені функції управління потоком та гнучкі опції зв'язку дозволяють користувачам керувати пристроями через хмару або за допомогою локальної мережі. BaseStation 3200 поєднує в собі інтелектуальну технологію поливу на основі датчика вологості ґрунту від Baseline та найкращі в галузі практики зрошення на основі погоди в єдиний користувальницький інтерфейс. BaseStation 3200 може допомогти вам зменшити споживання води до 62 відсотків. BaseStation 3200 використовує унікальну чутливу технологію та компоненти продуктивності, для того щоб підтримувати до 99 секторів та легко керувати їм. Програмування та функції BaseStation 3200 дозволяють вам точно контролювати воду на вашому інтерфейсі. Ви можете керувати кожним аспектом потоку, починаючи від джерела води до контрольної точки (головного клапана, насоса або ареометра) до магістралі та вниз до зони. Програмне забезпечення контролера зрошення Baseline вивчає потік для кожної зони, максимізує кількість зон, які він може включити одночасно. Базові датчики вологості ґрунту вимірюють фактичний вміст вологи в ґрунті там, де це дійсно важливо - в кореневій зоні. Можна точно

визначити, скільки зрошувати і до куди вистачає потужності. Це дозволяє запускати, зупиняти або призупиняти програми зрошення відповідно до дуже точних даних.

WeatherAccess для BaseManager забезпечує платформу, яка поєднує інтелектуальну технологію поливу на основі датчика вологості ґрунту з найкращими практиками зрошення на основі погоди в єдиний користувальницький інтерфейс.

Кожен BaseStation 3200 включає вбудований порт Ethernet, тому контролер готовий до підключення в інтернет. Варіанти бездротового підключення включають Wi-Fi, стільниковий модем та радіо з розширеним спектром [4].

Модуль ADAM-6100 від Advantech. ADAM-6100 на (рис. 2) є модулем вводу/виводу Industrial Ethernet і Industrial PROFINET. У цієї серії є вбудовані протоколи PROFINET або EtherNet / IP, він має можливість послідовного підключення, що дозволяє значно швидше передавати дані в процесі управління технологічним процесом і іншими додатками промислової автоматизації. Послідовне підключення також забезпечує більш масштабовану систему і допомагає підвищити стійкість до перешкод, характерним для заводських налаштувань. Ізольована конструкція запобігає руйнуванню серії ADAM -6100 і допомагає краще працювати на відкритих ділянках, з різними температурами, та в різних умовах [5].

Рисунок 1.2.2 ADAM-6100

1.3 Типи зв’язку у системах розумного поливу

Окрім звичайного дротового підключення, у системах розумного поливу можна використовувати бездротові технології для передачі даних. Вони зручні у використанні, дають можливість встановити датчики у важкодоступних місцях та керувати ними на відстані. Технології бездротової передачі даних потрібні при автоматизації великих та промислових теплиць, а також на полях які потребують розумного поливу. Також, за допомогою бездротових технологій стає можливо керувати процесом поливу і переглядати дані, які надходять з датчиків, статус системи, помилки які виникли при роботі.

Технологія Wi-Fi. WiFi чи 802.11 - це бездротовий протокол, який був побудований з метою заміни Ethernet за допомогою бездротового зв'язку через неліцензійні смуги. Його метою було забезпечити готовий, простий у впровадженні та у використанні бездротовий зв’язок короткого діапазону з взаємодією між постачальниками. При нульовій вартості спектра мало уваги зосереджувалось на спектральній ефективності, а з очікуваним використанням настільних пристроїв, енергоефективність не була критичною. Стандартний Wi-Fi, хоч і підходить для системи IoT, але він має обмеження як в діапазоні, так і в енергоефективності. IEEE усунули ці недоліки, опублікувавши специфікації для 802.11ah та 802.11ax:

WiFi HaLow або 802.11ah - технологія яка була заснована на стандарті IEEE802.11ah, затвердженому в жовтні 2016 року. Вона була введена спеціально для вирішення проблем, що стосуються діапазону та потужності систем IoT. 802.11ah використовує діапазон 900 МГц для забезпечення розширеного діапазону частот з низькими потребами до споживання енергії. Енергоспоживання додатково оптимізується за допомогою попередньо визначених періодів пробудження / сна і забезпечує радіус дії понад один кілометр. Але WiFi HaLow потребує спеціальних точок доступу і клієнтського обладнання.

POC або 802.11ax - стандарт високоефективного бездротового зв'язку (IEEE802.11ax) додає ряд зручних для функцій IoT. Він має такі ж функції з пробудженням та групуванням станцій як і 802.11ah, для того, щоб дозволити клієнтам зберігати енергію та уникати колізій зв'язку. Крім того, багатокористувацькі можливості MIMO в поєднанні з меншим інтервалом сигналу (78,125 кГц) дозволяють 18 клієнтам одночасно надсилати дані в межах каналу 40 МГц.

WiFi не є найкращим рішенням для систем інтернету речей, але часто його можна використовувати при створенні розумного буднику, систем безпеки, автоматизації поливу та освітлення в невеликих теплицях. В таких випадках розумні пристрої можна підключати до розеток і не турбуватися про енергозатратність. WiFi HaLow - розроблений спеціально для IoT, але вимагає спеціальної інфраструктури та спеціальних клієнтів.

Постачальники WiFi продовжують покращувати IoT, починають з'являтися корпоративні точки доступу з модулями (ZigBee, Bluetooth та іншими) [6].

Технологія Bluetooth. Розроблений компанією Ericsson в 1994 році, Bluetooth використовує короткохвильові УВЧ-хвилі між 2402 і 2480 МГц. Частина цієї технології, іменована Bluetooth LE (Bluetooth Low Energy, вона ж Bluetooth Smart, також відома як BLE) прямо позиціонує себе як ідеальний вибір для IoT (Internet of things). BLE вже вміє маршрутизувати Internet трафік, визначати координати в приміщеннях, підключати промислові програмовані логічні контролери, підтримувати WEB сервери, підключати ваги, термометри, пульсометри, оксиметри, тонометри і масу інших речей. С BLE автоматично вирішується безліч проблем властивих рішенням з використанням Wi-Fi. Вже скоро, пристрої з BLE зможуть організовуватися в MESH мережі, за технологією схожою з ZigBee. Це вже відображено в специфікації Bluetooth 5.0. Bluetooth технологію можна використовувати при створенні розумного будинку, автоматизації невеликих теплиць, також при створенні навчальних проектів

рекомендують використовувати Bluetooth модулі, через їх, відносну, легкість у підключенні та простоту у використанні [7].

Технологія ZigBee. Zigbee - це бездротова технологія створена як відкритий стандарт зв'язку, для того, щоб задовільнити потреби у низькій вартості та низькому електроспоживанню у IoT мережах. Стандарт працює на специфікації IEEE 802.15.4 фізичного зв'язку, а також, на неліцензійних смугах - 2,4 ГГц, 900 МГц, 868 МГц. Специфікація 802.15.4, за якою працює стек Zigbee, отримала ратифікацію Інститутом інженерів електротехніки та електроніки (IEEE) в 2003 році. Протокол дозволяє пристроям обмінюватися даними в різноманітних мережевих топологіях і має термін служби батареї протягом декількох років.

Протокол ZigBee 3.0 призначений для передачі даних через RF середовище, яке поширене у використані в комерційних та промислових додатках. Ціль ZigBee 3.0 в тому, щоб всі пристрої були підключенні без дротів до однієї мережі, незважаючи на їх призначення та функції. Схема сертифікації ZigBee забезпечує взаємодію продуктів від різних виробників. Підключення мереж ZigBee 3.0 до IP-домену дозволяє отримувати можливість моніторингу і контролю за допомогою персональних комп'ютерів, смартфонів і планшетів, тим самим створювати справжній інтернет речей.

Особливості протоколу ZigBee:

- Підтримка різних мережевих топології (точка-точка, точкові - багатоточкові та коміркові мережі) (рис. 1.3.1).
- Невеликий робочій цикл, який забезпечує тривалий час роботи джерела живлення .
- Малі затримки.
- Спектр поширення прямої послідовності (DSSS).
- Підтримує до 65000 вузлів на одну мережу.
- 128-битне шифрування AES.
- Уникнення колізій, повторних спроб і підтвердження.

Ключовим компонентом протоколу Zigbee є можливість підтримувати коміркові мережі. У комірковій мережі вузли взаємопов’язані з іншими вузлами, так що кожен вузол з'єднує кілька шляхів. Зв'язки між вузлами динамічно оновлюються та оптимізуються за допомогою вбудованої таблиці маршрутизації. Коміркові мережі мають децентралізований характер; кожен вузол здатний самостійно знаходитись у мережі. Крім того, коли вузли залишають мережу, топологія сітки дозволяє вузлам переконфігурувати шляхи маршрутизації на основі нової структури мережі. Характеристики топології комірок та спеціальної маршрутизації забезпечують стабільну роботу в умовах, де щось постійно змінюються [8].

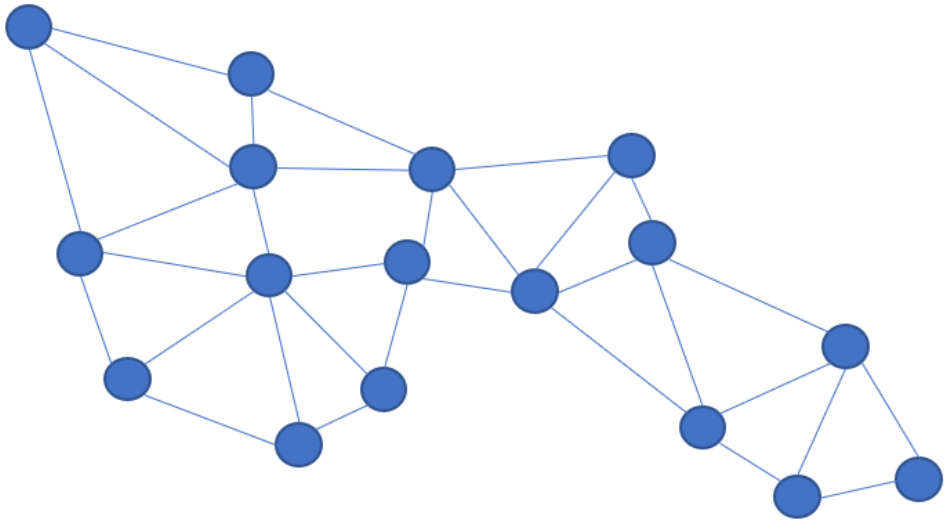

Figure 1 Mesh network

Рисунок 1.3.1 коміркова мережа

Технологія LoRa. LoRaWAN - це протокол LPWAN (Low Power Wide Area Network), побудований на базі бездротового RF LoRa Semtech та просунутий Альянсом LoRa, який працює на довгих дистанціях, має низьку ціну, мобільний, енергозберігаючий. Працюючи на неліцензійних частотах ISM у всьому світі, це підключення спеціально призначене для використання у системах IoT та дозволяє економічно та ефективно розгортати та виконувати операції як для державних, так і для приватних мереж. Протокол забезпечує надійний зв’язок на великі відстані (до 40 км у сільській місцевості та до 3 км у

міському середовищі, залежно від характеристик шлюзів) між датчиками на місцях та радіомережевими станціями. LoRaWAN був розроблений спеціально для зменшення енергоспоживання та продовження терміну служби батареї підключених датчиків до 10+ років. Він пропонує безпечну геолокацію на основі мережі, щоб знайти будь-який виправлений або мобільний кінцевий пристрій із підтримкою LoRa. LoRa використовує різницю в часі щодо прибуття та інші гібридні методи для визначення місця розташування без використання додаткової обробної потужності на стороні кінцевого пристрою. Ця вбудована можливість, на відміну від GPS, не збільшує споживання енергії і не потребує додаткових дорогих апаратних компонентів. Мережі LoRaWAN працює на безліцензійних, економічних та безкоштовних ISM смугах - EU 868 МГц, AS 923 МГц, US 915 МГц - де надає доступ будь-якому постачальнику послуг або компанії для розгортання та експлуатації мереж без отримання ліцензії від регулятора. Стандарт LoRaWAN використовує підхід з відкритим протоколом на чолі з Альянсом LoRa, який об'єднує специфікацію, розробку та розгортання стандарту, розробляє керівництво з сертифікації та забезпечує взаємодію між усіма постачальниками радіоінфраструктури, постачальниками основних мереж та розробниками та виробниками кінцевих пристроїв. Відкритий стандарт LoRaWAN, інфраструктура високої продуктивності та послуги з підключенням з доданою вартістю дозволяють державним або приватним операторам швидко розгортати та експлуатувати мережі за допомогою оптимізованих капітальних витрат (Capex) та операційних витрат (Opex) , а також забезпечувати швидкий час виходу на ринок, щоб отримати вигоду від розширення IoT у різних сегментах. LoRaWAN пропонує два надійні рівні наскрізної безпеки.

Один для мережі, який забезпечує взаємну автентифікацію між кінцевим пристроєм та мережею як частину процедури приєднання до мережі. Це гарантує, що мережевий трафік не був змінений, він призначений лише для законного кінцевого пристрою, уповноваженого приєднуватися до автентичної мережі.

Один для програми, яка гарантує, що оператор мережі не має доступу до даних програми кінцевого користувача. LoRaWAN - одна з небагатьох мереж IoT, що реалізує наскрізне шифрування корисних навантажень додатків, якими обмінюються кінцеві пристрої та сервери додатків. У традиційних стільникових мережах трафік зашифровується по повітряному інтерфейсу, але передається як звичайний текст в основній мережі оператора, вимагаючи від користувача вибрати та розгорнути додатковий рівень безпеки (VPN або спеціальний додаток для шифрування рівня безпеки).

LoRaWAN пропонує мережі з високою масштабованістю та пропускною здатністю для підтримки тисячі підключених кінцевих пристроїв та мільйонів переданих повідомлень. Ця висока продуктивність досягається використанням адаптивної швидкості передачі даних (ADR) та використанням багатоканального багатомодемного приймача в шлюзі, щоб можна було приймати одночасні повідомлення на декілька каналів. Мережа LoRaWAN розгорнута в архітектурі мережі «зірка зірок», завдяки якій кінцеві пристрої не пов’язані з певним шлюзом, а передають дані на безліч шлюзів у радіусі дії [9].

1.4 Дослідження контролюючих плат для розумного поливу

Під час розробки системи розумного поливу, з’явилася ціль у тому, щоб підібрати найкращу керуючу плату до даного проекту, так як плата є мікрокомп’ютером, головним елементом, що отримує дані з датчиків і приймає рішення по поливу згідно з закладеною в нього програмою. Керуюча плата повинна бути не вразливою як до зовнішніх чинників так і від внутрішніх тому, що від її працездатності залежать кількість і якість врожаю яку отримає підприємець. Також, для неї важливо мати доступ до інтернету та підтримувати бездротові протоколи передачі даних, бо це дає змогу лише одному оператору контролювати процес роботи багатьох автоматизованих теплиць і вирішує питання з місцерозташуванням цього оператора. При проведенні роботи було

досліджено чотири види плат - NodeMCU, Arduino UNO, Arduino NANO та Ukraino UNO.

Arduino UNO. Arduino Uno – це мікроконтролерна плата яка заснована на мікроконтролері ATmega328 AVR із вбудованим подвійним пакетом DIP.(рис.1.4.1) Контролер має 20 цифрових входів/виводів з яких 6 можна використовувати для широко-імпульсної модуляції та 6 аналогових входів. Програми до плати можна завантажувати за допомогою простої у використанні програмної середи Arduino IDE. Arduino IDE має широку спільноту підтримки і велику кількість бібліотек, що робить його простим у використанні та легким у навчанні. Uno відрізняється від усіх попередніх плат тим, що він не використовує чіп драйвера FTDI USB-to-serial. Натомість, він використовує ATmega16U2 який запрограмований як USB-to-serial перетворювач. Цей допоміжний мікроконтролер має власний завантажувач USB, що дозволяє досвідченим користувачам перепрограмувати його. Перший проект Arduino був започаткований в Інституті дизайну Ivrea у 2003 році Девідом Куартіелем та Массімо Банзі з метою забезпечити дешевий та гнучкий спосіб для студентів та професіоналів для управління низкою пристроїв у реальному світі. Отже, Arduino Uno має дуже велику підтримку серед спільноти і зручність у використані, але із негативних аспектів у UNO можна виділити її більшу ціну порівняно з іншими аналогами і те, що для цієї плати потрібно докупляти різні модулю, такі як: модуль WiFi, модуль SSD, ZigBee модуль, тощо [10].

Рисунок 1.4.1 Arduino Uno

Arduino Nano. Arduino Nano - це невелика і зручна у використанні плата на базі ATmega328 (Arduino Nano 3.x)(рис.1.4.2). Вона функціонує як Arduino Duemilanove, але в іншому форматному пакеті. У контролера відсутній роз'єм живлення постійного струму і він працює з USB-кабелем Mini-B замість стандартного кабелю. Кожен з 14 цифрових контактів на Nano можна використовувати як вхід або вихід, використовуючи функції pinMode (), digitalWrite () та digitalRead (). Вони працюють при напрузі 5 вольт. Кожен штифт може забезпечувати або приймати максимум 40 мА і має внутрішній підтягуючий резистор (відключений за замовчуванням) 20-50 кОм. Крім того, деякі шпильки мають спеціальні функції:

- Послідовні 0(RX) та 1(TX) які потрібні для отримання (RX) та передачі (TX) послідовних даних TTL. Ці виводи підключені до відповідних вводів послідовної мікросхеми FTDI USB-to-TTL.

- Зовнішні переривання на пінах 2 і 3. Ці піни налаштовані на переривання на низькому значенні, зростаючому або спадаючому фронті або зміні значення.
- Піни широко імпульсної модуляції 3, 5, 6, 9, 10, 11. Вони забезпечують 8-бітний вихід ШІМ за допомогою функції analogWrite ().
- SPI піни - 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Ці піни підтримують комунікацію SPI.
- Світлодіод на піні 13. Вбудований світлодіод підключений до цифрового піна 13.

Arduino Nano має ряд засобів для зв'язку з комп'ютером, іншим Arduino або іншими мікроконтролерами. ATmega328 забезпечує послідовний зв'язок UART TTL (5 В), який доступний на цифрових контактах 0 (RX) і 1 (TX). FTDI FT232RL на платі складає цей послідовний зв’язок через USB, а драйвери FTDI (входять до програмного забезпечення Arduino) забезпечують віртуальний com-порт для програмного забезпечення на комп’ютері. Програмне забезпечення Arduino включає послідовний монітор, який дозволяє надсилати прості текстові дані на плати та з плат. Світлодіоди RX і TX на платі будуть блимати, коли дані передаються за допомогою мікросхеми FTDI та USB-з'єднання на комп'ютер. Бібліотека SoftwareSerial дозволяє здійснювати послідовний зв'язок на будь-якому з цифрових контактів Arduino Nano. Також ця плата має корисну особливість у тому, що в ній є функція програмного скидання. Замість того, щоб фізично натисканням кнопки скидати конфігурацію на платі, Nano дозволяє скидувати його за допомогою програмного забезпечення, що працює на підключеному комп’ютері. Отже, Arduino Nano є чудовим вибором при створенні проекту, вона компактна та зручна у використанні. Хоч вона і коштує більше ніж деякі її аналоги, але якість і деякі цікаві функції роблять її фаворитом при створенні систем розумного поливу або інших розумних систем [11].

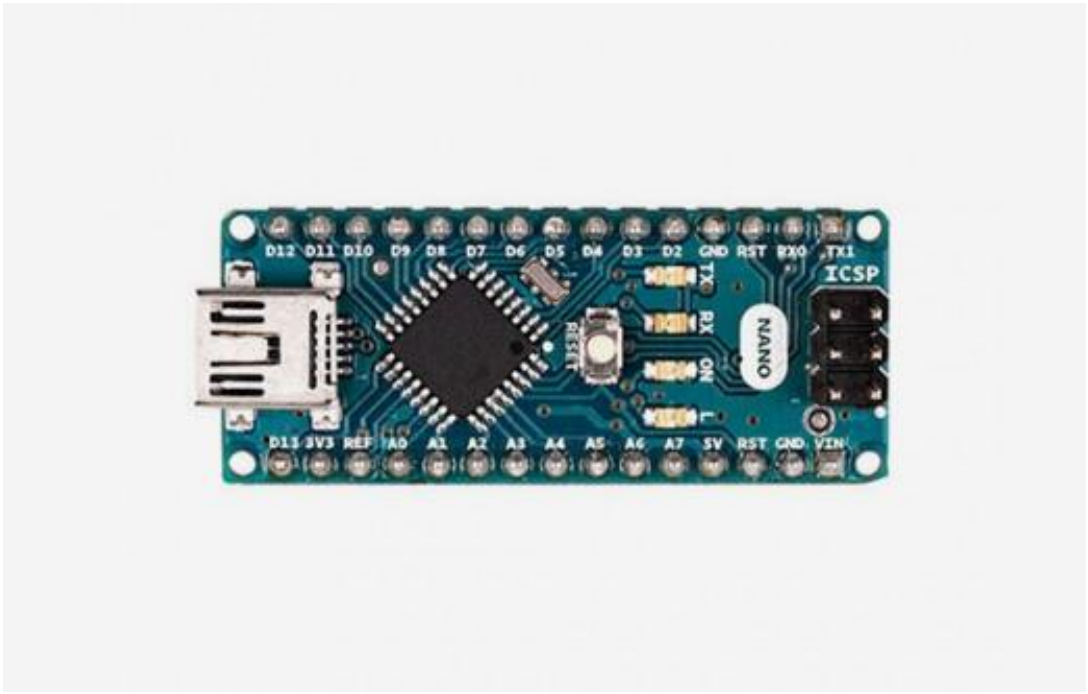

Рисунок 1.4.2 Arduino Nano

Ukraino Uno. При аналізі розумних контролерів не можна не згадати український аналог плати Arduino Uno - Ukraino Uno. Ця плата ідентична своєму аналогу, але має кілька доповнень яких немає у Uno R3 (див. Рис. 1.4.3). Апаратно, плата повністю сумісна з усіма шилдами і модулями для Arduino Uno, а також, з програмної сторони - з усіма її бібліотеками і програмами.

Основні відмінності Ukraino Uno від оригінальної Uno R3:

- В якості USB-UART конвертера замість чіпу ATMega16U2 використовують більш надійний CP2102.
- Має перемикач напруги логічних рівнів на виходах плати - 5 чи 3.3В.
- Діапазон вхідних напруг збільшений до 7...25В.
- Встановлений стабілізатор 78M05 замість AMS1117.
- Має якісні танталові конденсатори, які дозволяють отримувати живлення для модулів прямо з плати.
- Вибір джерела живлення між USB та за зовнішнім блоком живлення здійснюється за допомогою перемикача Ext/USB.
- В схемі використані деталі широко поширені.

Ukraino Uno є оптимальним варіантом для вивчення мікроконтролерів. Її потужності, обсягу пам'яті і кількості висновків досить для більшості завдань, які виникають в аматорській електроніці [12].

Рисунок 1.4.3 плата Ukraino

Node MCU. Node MCU - це середовище розробки програмного та апаратного забезпечення з відкритим кодом, побудоване на недорогій системі на чіпі (SoC) під назвою ESP8266(рис.1.4.4). ESP8266, розроблений та виготовлений компанією Espressif Systems, містить найважливіші елементи комп'ютера: процесор, оперативну пам'ять, WiFi модуль і навіть сучасну операційну систему та SDK. Це робить його чудовим вибором для проектів інтернету речей усіх видів. У кожну плату NodeMCU входить USB-послідовний перетворювач. В офіційній версії плата заснована на наборі мікросхем CP2102 і пропонує найкращу сумісність. Справжні плати використовують набір мікросхем CP2102, включаючи офіційно ліцензовані модулі Amica NodeMCU.

Іншим поширеним перетворювачем USB у послідовний пристрій є CH340G, який часто зустрічається у недорогих модулів, включаючи блоки LoLin. Інші конструкції можуть використовувати драйвери, включаючи набір мікросхем FTDI, але ці конструкції зустрічаються не часто. Залежно від операційної системи, яка використовується з NodeMCU, повинен бути встановлений відповідний драйвер. Як правило, Windows 10 відразу розпізнає набір мікросхем CP2102, тоді як CH340G може вимагати окремого встановлення.

Для підключення більшості модулів і датчиків, при роботі з Node MCU, може знадобитися breadboard, а програмувати плату можна в середовищі Arduino IDE, хоча можна зазначити, що це не я самим зручним середовищем для програмування. Вбудований WiFi модуль є дуже корисною особливістю, так як можна одразу починати роботу с веб-інтерфейсом, що дуже зручно при появі задачі в створенні системи моніторингу. Контролер коштує набагато дешевше ніж його аналоги, проте, має якісні запчастини, компактний розмір і легкість при вивчанні. Під час створення практичного проекту було обрано саме його через вище перераховані властивості [13].

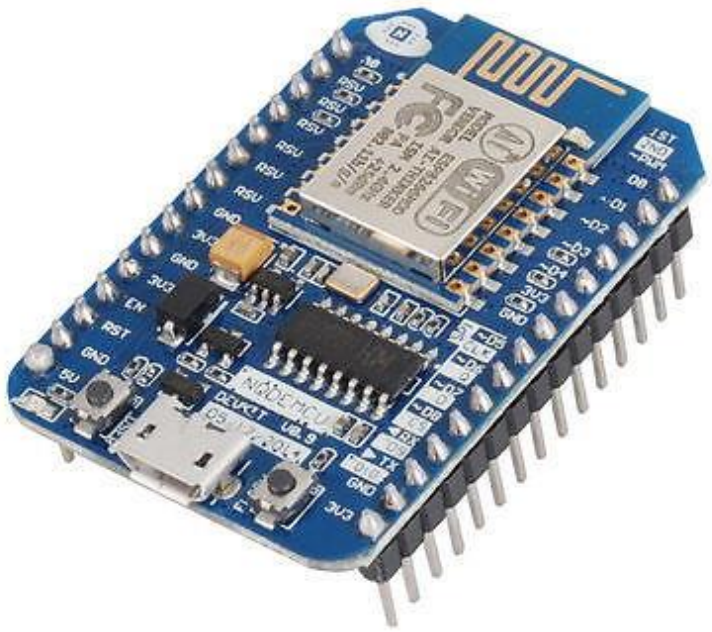

Рисунок 1.4.4 плата NodeMCU

1.5 Висновок по розділу

В цьому розділі було проведено аналіз актуальності розумного поливу, проведено дослідження готових рішень як від іноземних так і від українських виробників(таких як IT-LYNX). Також було досліджено види бездротового зв’язку, такі як: WiFi, Bluetooth, ZigBee, LoRA. Були приведені їх сильні та слабкі сторони. Були досліджені розумні мікроконтролери які можна використовувати при створенні розумних систем. Були дослідженні їх характеристики, властивості та особливості. Після виконання розділу було зроблено висновки щодо необхідності створення систем розумного поливу у сучасному світі, було обрано найкращий тип бездротового зв’язку для створення системи і також було відібрано мікроконтролерну плату яка підійде для керування та моніторингу поливу.

РОЗДІЛ 2. РОЗРОБКА ПРОЕКТУ ТА СТВОРЕННЯ СИСТЕМИ РОЗУМНОГО ПОЛИВУ

2.1 Постановка задачі та створення концепції розумного поливу

Цілю роботи є створення системи розумного поливу. Для такої системи потрібно підібрати керуючу мікросхему для контролю процесів у проекті, також, треба вибрати датчики температури та вологості для секторів у теплиці для глибокого контролю поливу. Ще треба підібрати електромагнітні клапани, які будуть під'єднані до мікроконтролера і керування якими - буде відбуватися за допомогою силового ключа. Отже, спочатку було прийнято рішення у програмному середовищі Visio створити концептуальну модель системи, для кращого розуміння при створенні.

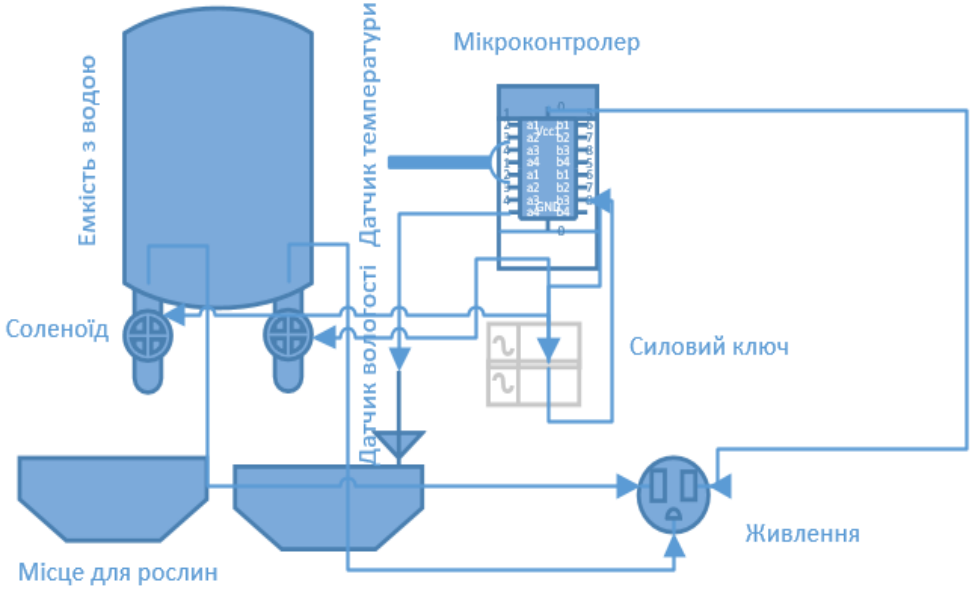

Рисунок 2.1.1 Концептуальна модель розумного поливу в фермерському господарстві "Дісті"

Як видно на (рис. 2.1.1), до ємності з водою треба під'єднати електромагнітні клапани і підключити їх до силового ключа, а також до живлення, так як вони працюють від мережі 220V. До мікроконтролера треба підключити два датчика: вологості і температури і теж під'єднати його до

мережі. Для поливу секторів, де будуть рости рослини, треба зробити систему крапельного поливу для подачі води.

Після того, як було створено модель системи розумного поливу, треба провести дослідження обладнання та обрати найкраще для створення проекту.

2.2 Аналіз і підбір обладнання для створення проекту

Насамперед треба обрати керуючу плату, так як це - головний елемент системи, який буде приймати на себе інформацію з датчиків температури і вологи, і згідно до своєї налаштованої логіки, буде вирішувати - відкривати соленоїдні клапани або зачиняти їх. Також з мікроконтролера можна буде переглядати дані за допомогою веб-інтерфейсу, а також проводити моніторинг з Телеграм-боту.

Керуюча плата. Під час розробки проекту, було обрано дві плати для порівняння - Arduino Uno R3 та Node MCU v0.9. Arduino Uno - .більш зручний для початку. Завдяки рейкам для пінів до нього, можна одразу підключити датчик, реле, модуль, тощо і починати проводити дослідити і створювати проекти. А в інтернеті є дуже багато різних сайтів та бібліотек від спільноти, які допоможуть на початку роботи [див розділ 1.4]. Але у Arduino є свої мінуси. По-перше це ціна і необхідність докупати модулі для бездротового з’єднання. Оригінальна плата коштує дорожче ніж аналоги і для того, щоб створити програми для моніторингу - потрібну докупляти WiFi модулі для бездротового підключення. Також, цей мікроконтролер має досить великий розмір і потребує більше місця при комплектуванні системи (рис.2.2.1).

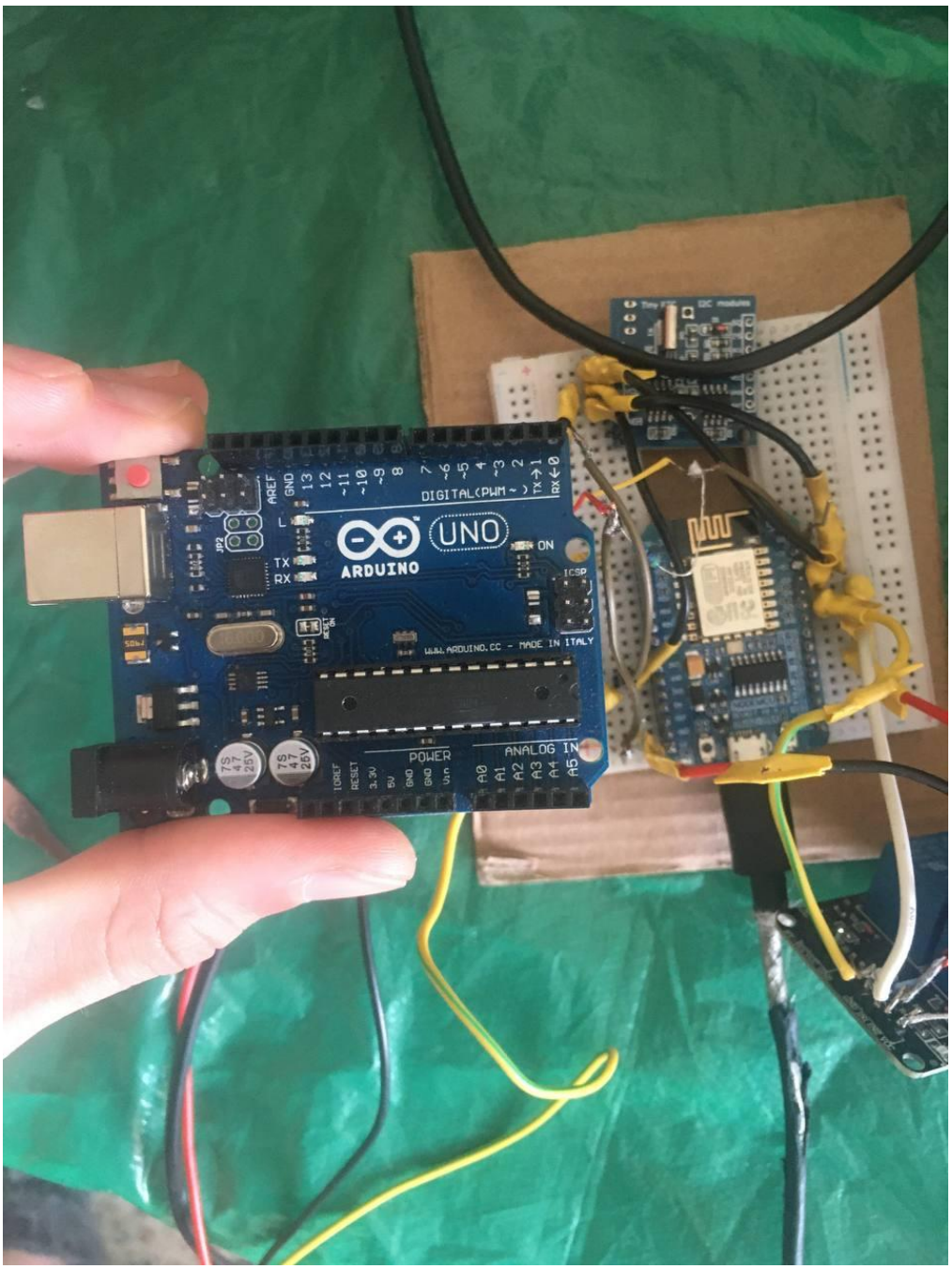

Рисунок 2.2.1 порівняння Arduino Uno R3 та Node MCU v0.9

При цьому, розумний контролер Node MCU обходить плату Arduino по всім параметрам. Коштує він в два рази дешевше ніж версія Uno R3 і має вбудований модуль WiFi, який необхідний для створення веб-інтерфейсу для моніторингу [див. Розділ 1.4]. Також ця плата більш компактна, що додає зручності при комплектуванні системи.

Для того, щоб підключати датчики та модулі до мікроконтролера, нам знадобиться breadboard, (рис.2.2.2) а для підключення до живлення і до комп’ютера - потрібен дріт USB – micro [14].

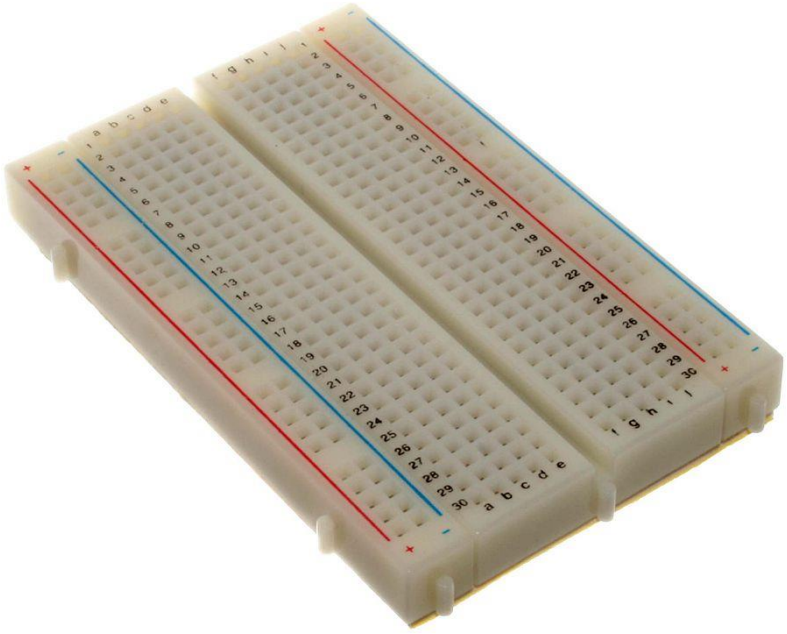

Рисунок 2.2.2 - Breadboard для підключення дротів

Після того, як було обрано керуючу плату, треба вирішити які датчики вологи і температури підійдуть до системи.

Датчик вологи. Під час проведення аналізу датчиків вологості було зроблено висновки, що датчики по типу Soil Hydrometer (Рис.2.2.3) та його аналоги не підходять для проекту[15], тому що, їх термін придатності не більше ніж кілька місяців. Це все через те, що зубці між якими проходить струм - зроблені з неякісного метала, що дуже швидко ржавіє від вологості ґрунту і перестає працювати. При створені системи розумного поливу потрібно мати надійне обладнання, яке буде працювати більше ніж рік і забезпечувати якісну роботу.

Рисунок 2.2.3 Датчик вологості Soil Hydrometer

Тому для проекту було обрано датчик Capacitive Soil Moisture Sensor v1.2.(рис.2.2.4). Це сенсор вологості ґрунту зі стійким до корозії покриттям [16]. Він приймає на себе напругу живлення в 3,3 і 5 Вольт. Вихідний аналоговий сигнал від 0 до 3 Вольт. Має конектор типу PH2.0-3P. Та розміри - 99 на 16 мм. Сенсор дуже зручний у використані та дає досить зручні показники.

Рисунок 2.2.4 Capacitive Soil Moisture Sensor v1.2

Датчик температури. Для вимірювання температури було обрано цифровий датчик DS1820 в металевій захисній гільзі [17]. Це провідний датчик з довжиною кабелю в 1 метр, і напругою живлення в 5 Вольт. Робоча температура для нього від -55 градусів по Цельсію до + 125 градусів по Цельсію. Цей сенсор підходить до умов, де треба працювати в вологому середовищі. Металева захисна гільза захищає механізм вимірювання (рис.2.2.5).

Рисунок 2.2.5 датчик температури DS1820

Також при виборі сенсора температури було розглянуто FiB_DS-001, але було вирішено не брати його, так не зручний у підключенні та є дуже велика вірогідність купити підробку.

Електромагнітний клапан. Для того, щоб керувати подачею води, нам потрібні електромагнітні клапани чи, як вони ще відомі, соленоїди. Соленоїди повинні бути встановленими в ємкість з водою і бути підключеними до мережі 220 або 20 Вольт, а також до силового ключа, за допомогою якого, розумний контролер буде керувати ними (рис.2.2.6).

Під час аналізу електромагнітних клапанів було звернено увагу на - Латунний електромагнітний клапан ¾ [18]. Він зроблений з латуні, працює з напругою у 220 Вольт, пропускає тиск в 0,7 МПа та працю є при температурі від - 10 до 100 градусів Цельсію. Але він має досить велику ціну, під своєю вагою -

клапан може зірвати різьбу, а форма вихідного отвору досить не зручна для підключення шлангу для води.

Рисунок 2.2.6 Латунний електромагнітний клапан ¾

Тому було знайдено універсальний електромагнітний клапан 1/180, корпус якого вироблений з пластику, [19] має вихідний отвір 6 мм , який підходить до підключення трубки для поливу. Також цей соленоїд має гарну ціну і чудово підходить до ємності з водою. Цей клапан має фільтрову сітку для запобігання попадання важких частинок до рослин (рис.2.2.7).

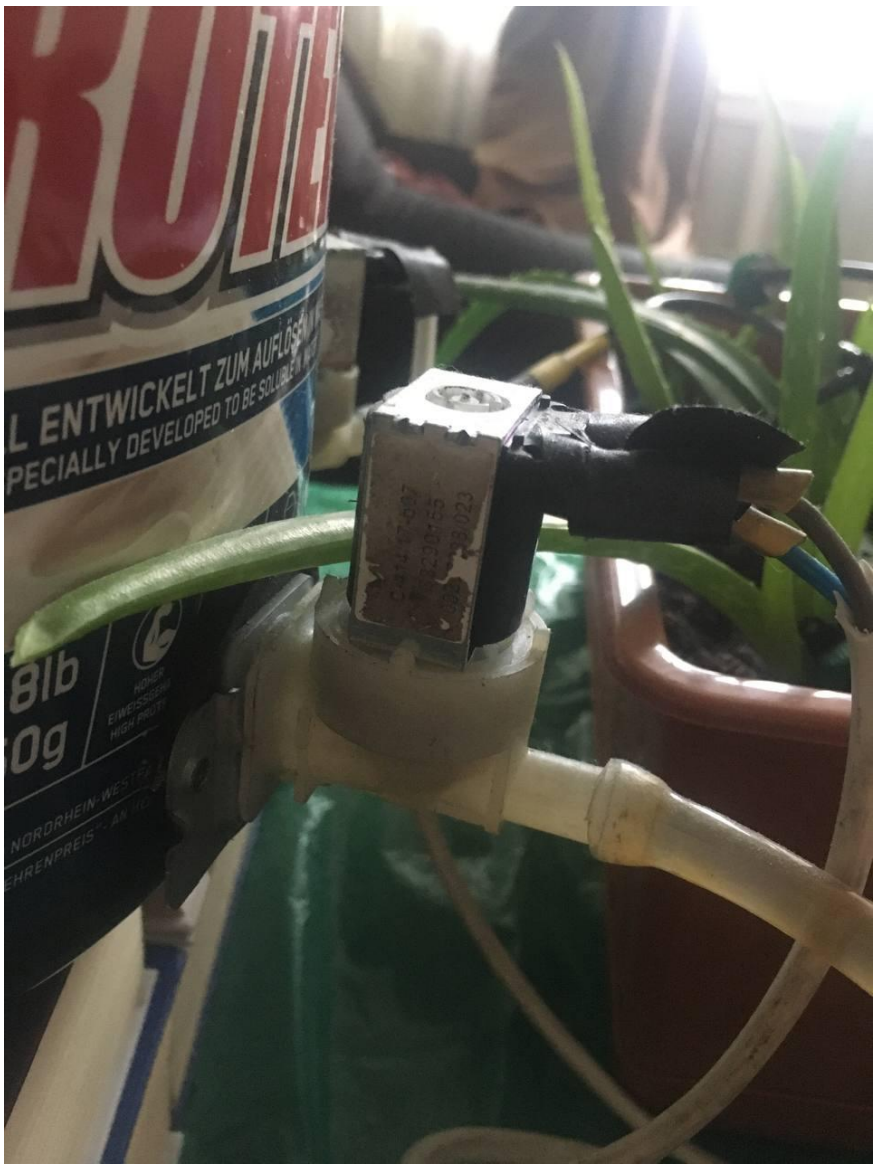

Рисунок 2.2.7 соленоїд 1/180

Силовий ключ. Силовий ключ, або реле - обов’язковий елемент для роботи з соленоїдними клапанами. Плата подає сигнал на цей ключ (0 або 1), і залежно від цієї фази, силове реле подає струм до соленоїдів.

Спочатку, для системи було обрано AQZ404 Power Relay, його було підключено до системи і проведено кілька дослідів, після який, виявилось, що цей ключ, під час роботи, дуже сильно нагрівається, тому, для справної роботи, його треба встановлювати на радіатор. Також під час роботи з ним було виявлено деякі помилки коли плата передавала напругу на нього.

Тому, після цих дослідів, було вирішено взяти пару силових реле JQC-3FF-S-Z (рис.2.2.8) які вже були встановлені на модуль з усіма потрібними речами для їх роботи.

Рисунок 2.2.8 Силовий ключ JQC

2.3 Підключення елементів, створення системи розумного зрошування

Після того, як було підібрано всі компоненти, їх треба правильно підключити до мікроконтролера і між собою для того, щоб вони могли працювати згідно з їх призначенням.

Спочатку, плату було встановлено на breadboard , для зручного підключення інших елементів до неї. Для зручності, breadboard був розділений на дві частини через особливості розташування вусиків на платі (див.рис.2.3.1).

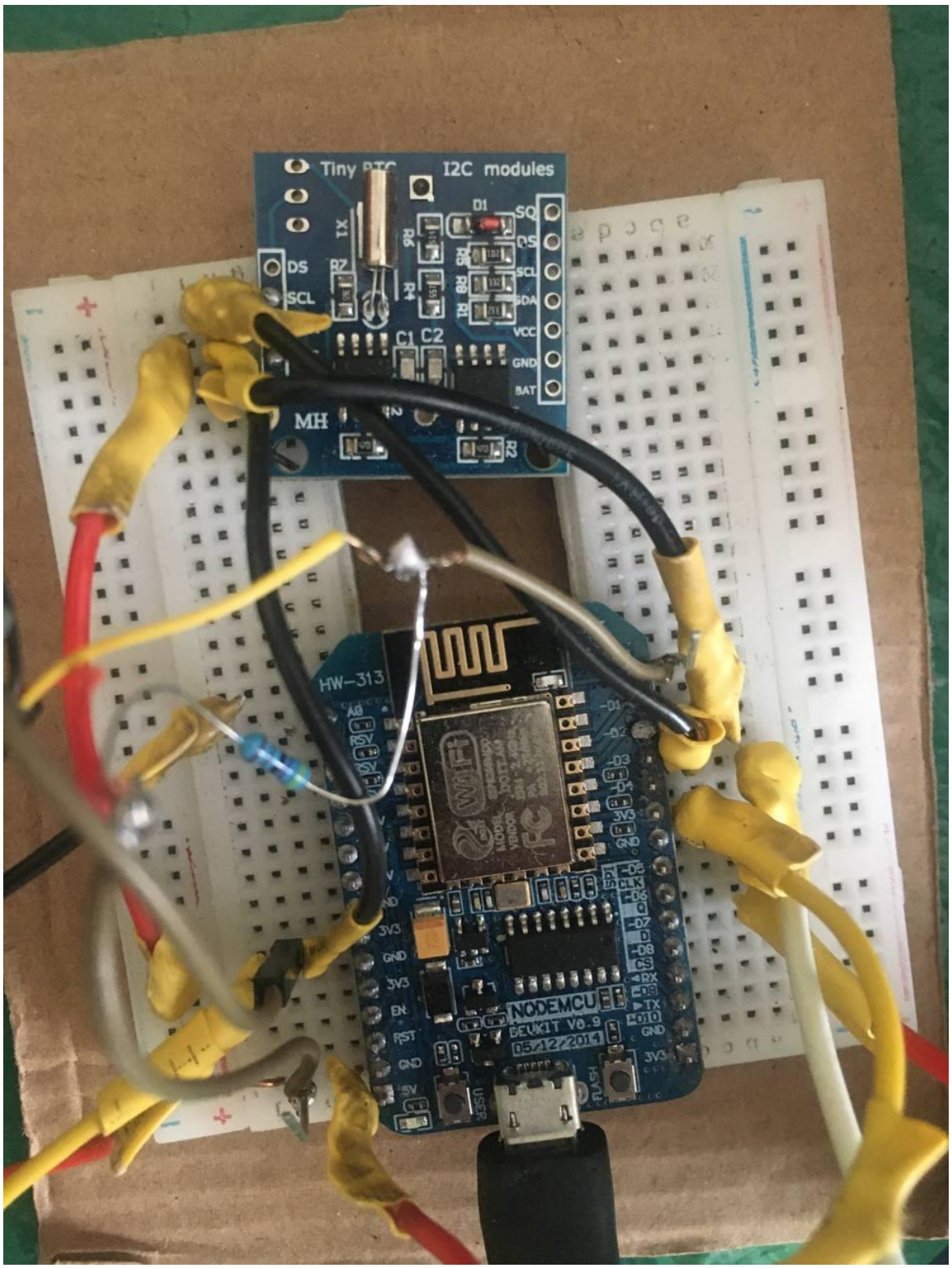

Рисунок 2.3.1 Плата на breadboard

Після цього, до плати було під’єднано датчики вологості і температури. Датчик вологості мав досить короткий дріт, тому було прийнято рішення подовжити дроти які були на ньому для зручності у використані (рис. 2.3.2).

Рисунок 2.3.2 подовжені дроти у датчика вологості

Також голі дроти були покриті гідроізоляційною трубкою, для безпечного використання при роботі системи. Датчик вологості має три дроти які треба підключити до плати: GND (Жовтий), VCC (Червоний), AOUT (Чорний). GND (Жовтий) був підключений до GND на платі, VCC (Червоний) був підключений до 5V, а AOUT (Чорний) був підключений до аналогового порту A0.

До датчика температури було під’єднано резистор, згідно з інструкцією (рис.2.3.3).

Рисунок 2.3.3 Датчик температури з резистором.

Датчик температури також має три дроти, GND (Чорний), VCC (Червоний), DOUT (Жовтий). GND (Чорний) був підключений до GND на платі, VCC був підключений до 5V, а DOUT (Жовтий) до цифрового порту D0.

Після того, як датчики вологості і температури були під’єднані, почалась робота по підключенню силового ключа. Спереду силового реле було під’єднано 4 дроти для контролю за роботою електромагнітних клапанів (рис.2.3.4).

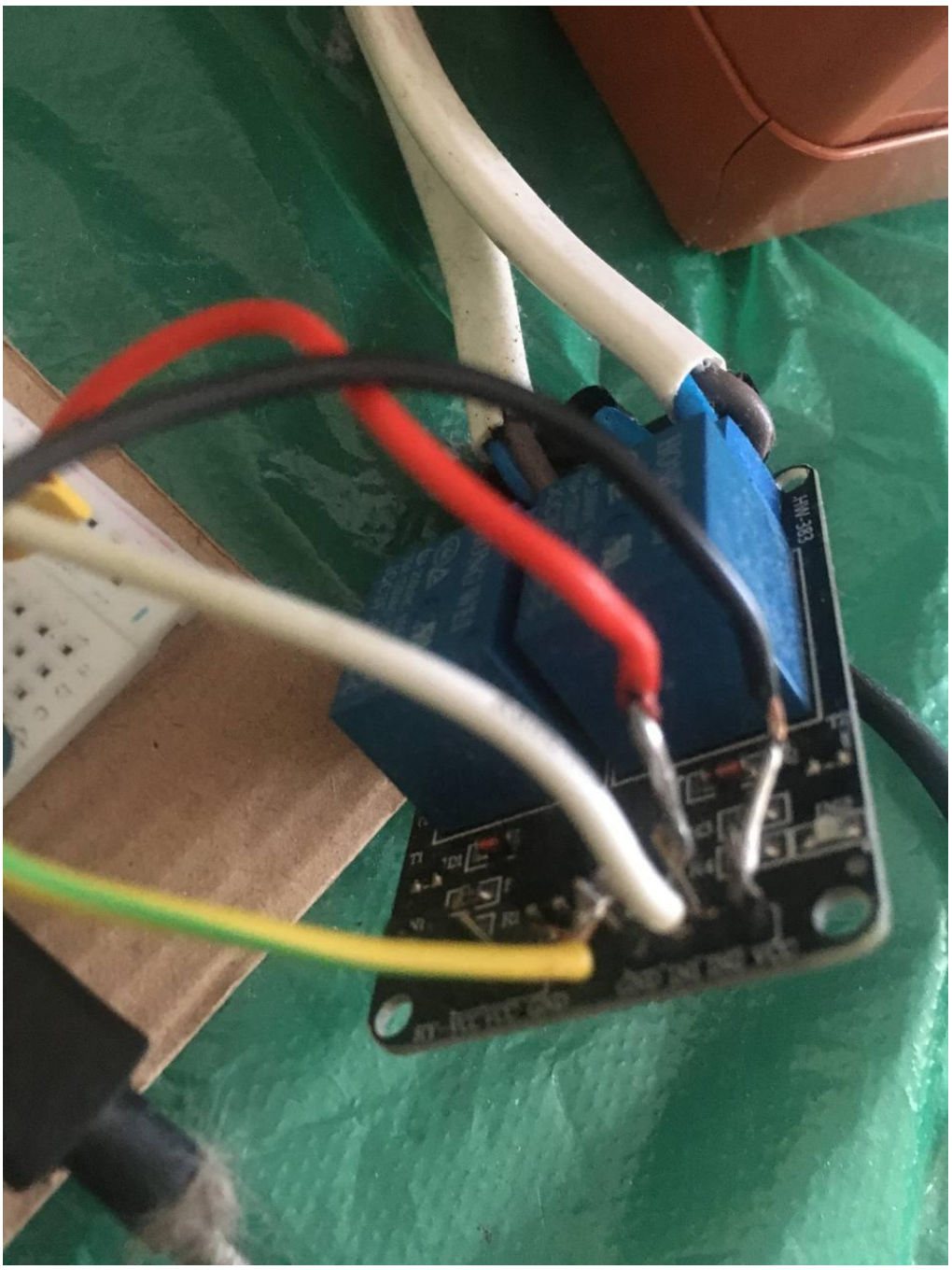

Рисунок 2.3.4 Передня частина силового реле

GND (Жовтий) було підключено до GND на платі, IN1 (Білий) який використовується для контролю за першим соленоїдом, було підключено до цифрового порту D4. IN2 (Червоний) який використовується для контролю за другим соленоїдом, його було підключено до цифрового порту D7 на платі. Також VCC (Чорний) було під’єднано до 5V на мікроконтролері.

До задньої сторони силового ключа було під’єднано великі дроти для робити с електромагнітними клапанами та напругою у 220V (рис.2.3.5).

Рисунок 2.3.5 задня частина силового ключа

Дроти були підключенні до свої полюсів, а також були підведені і підключенні до електромагнітних клапанів (рис. 2.3.6). Також голі дроти були обгорнуті в ізоляційну стрічку, для безпечної роботи із системою.

Рисунок 2.3.6 підключення електромагнітного клапана

Живлення до соленоїдів подається через мережу 220V, тому було під’єднано дріт який з’єднує силовий ключ з мережею 220V (див.рис.2.3.7).

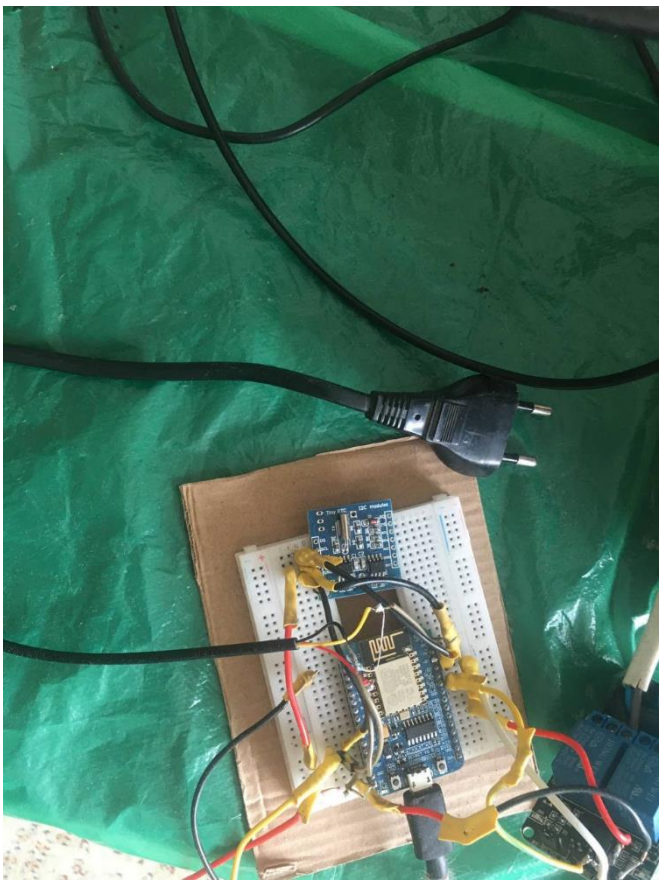

Рисунок 2.3.7 Дріт 220V

Після того як система була зібрана, треба вирішити питання з постачанням води і середовищем, куди вода буде надходити. Було обрано крапельний тип поливу, а спосіб подачі води - через самополив. Для цього було обрано ємність в 10 літрів (Рис. 2.3.8). В цій ємності було зроблено два отвори для підключення соленоїдів. Клапани були закріпленні за допомогою гайки ¾ та зажаті резиновою прокладкою, для герметичності.

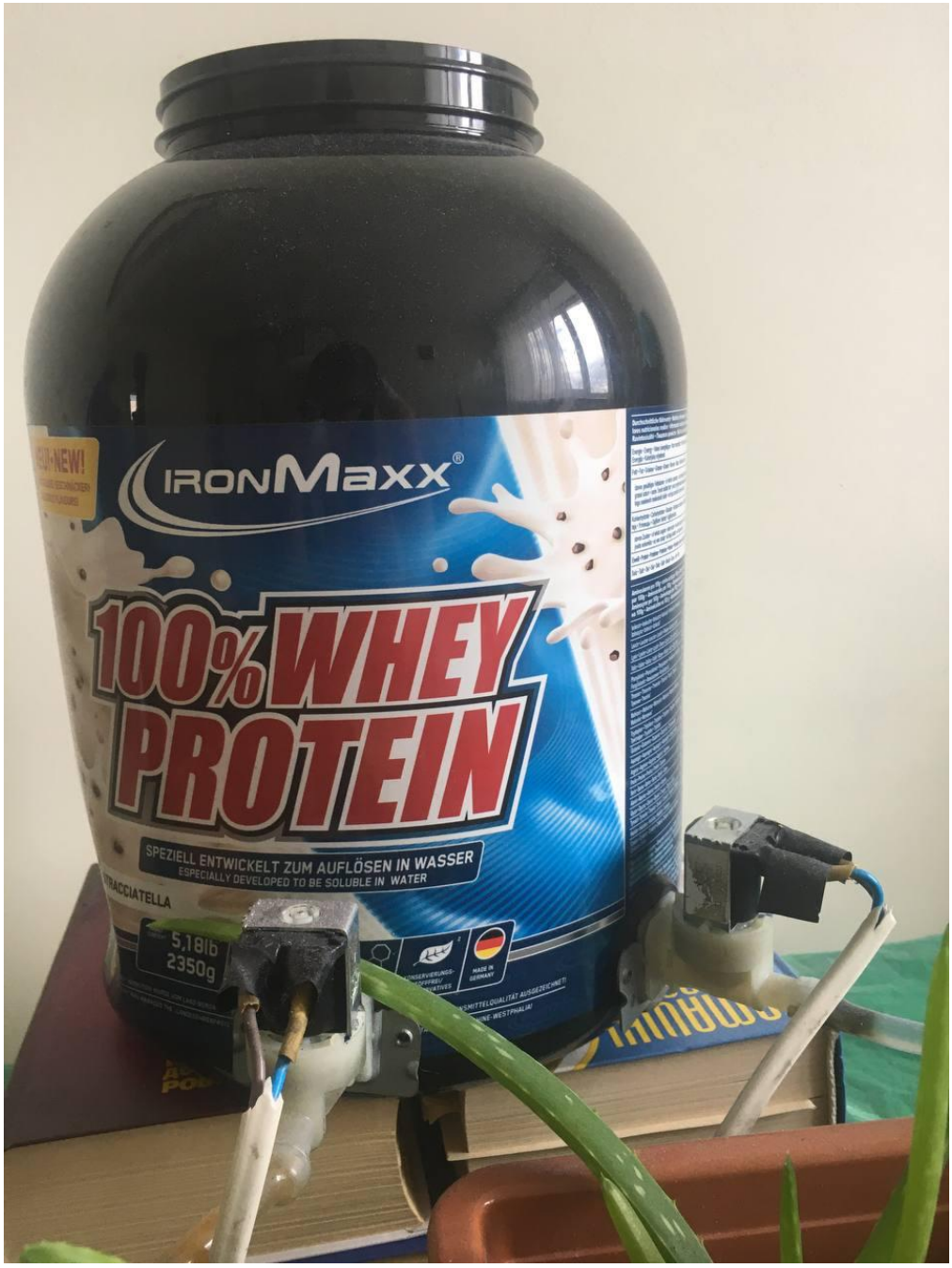

Рисунок 2.3.8 Ємність з водою та підключенні соленоїди.

Для крапельного поливу було зроблено крапельну лінію з крапельницями на кінцях для точкового поливу рослин (див. Рис. 2.3.9). Але при під’єднані лінії до клапана виявилось, що розміри не співпадають. Тому було прийнято рішення зробити перехідник з 6мм харчової трубки. 4мм шланг лінії намастили клеєм та засунули в харчову трубку, а зверху встановили гідроізоляцію яку раніше використовували при роботі з дротами. Перехідник виправдав очікування - не протікав і через кольорове забарвлення отримав назву - “Бджілка”.

Рисунок 2.3.9 Крапельний полив з перехідником до клапану

Після цього, система була повністю готова для роботи і поливу, залишилося тільки налаштувати її.

2.4 Створення логічної і фізичної бази даних для системи

Логічна модель була створена у середовищі Oracle SQL Developer Data Modeler (рис.2.4.1).

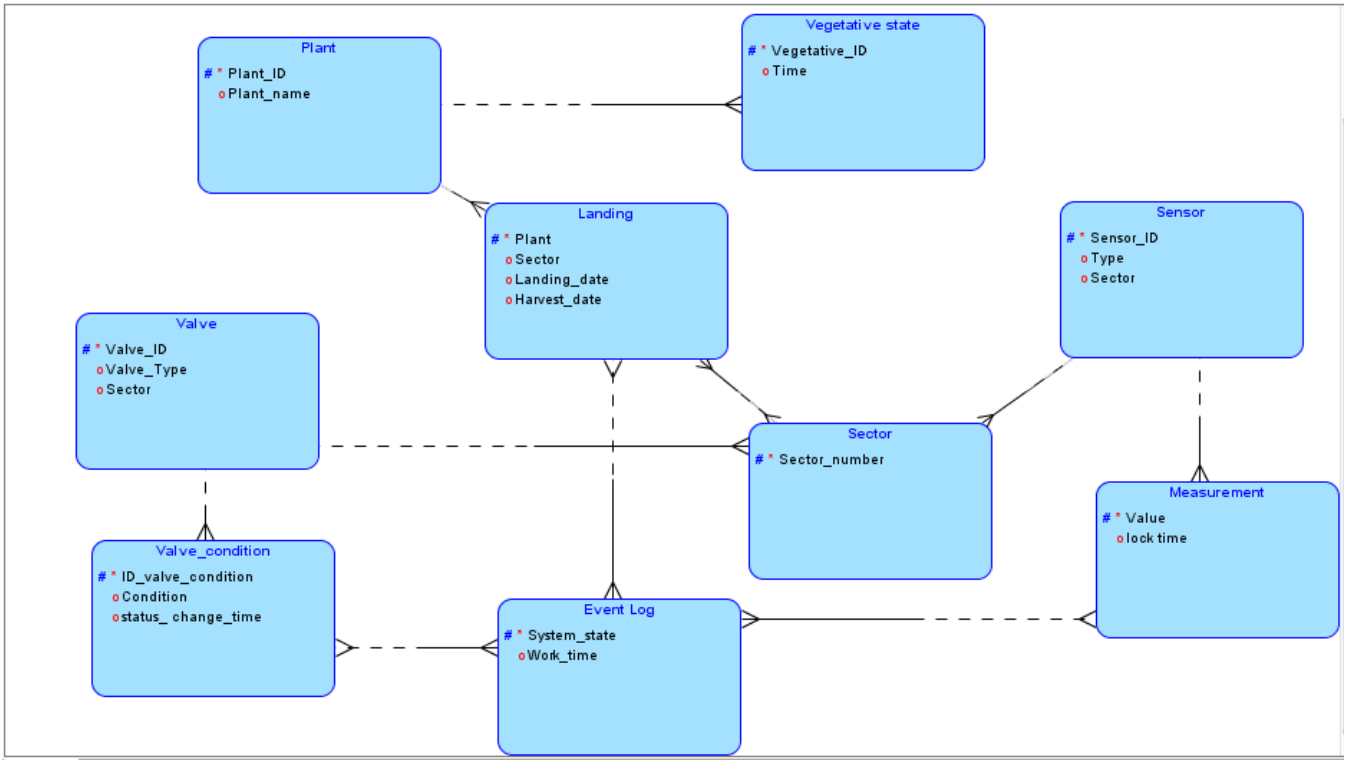

Рисунок 2.4.1 Логічна модель для контролю системи розумного поливу

Логічна модель бази даних містить у собі 9 таблиць (див. Рис.2.4.1):

1. Plant (Рослини).
2. Vegetative state (Вегетаційний стан).
3. Landing (Посадки).
4. Sector (Сектор).
5. Sensor (Датчик).
6. Measurement (Показники).
7. Valve (Клапан).
8. Valve condition (Стан клапану).
9. Event Log (Журнал подій).

Таблиця Plant містить у собі два поля - Plant_ID, Plant_name, де Plant_ID це первинний ключ.Тип даних для Plant_ID- integer, Plant_name- varchar(50).

Таблиця Vegetative state містить у собі три поля - Vegetative_ID, Time, Plant_ID де Vegetative_ID це первинний ключ. Тип даних для Vegetative_ID- integer, Time- Time, Plant_ID- integer.

Таблиця Landing містить у собі 5 полів - Plant, Sector, Landing_date, Harvest_date, Plant_ID де Plant це первинний ключ. Тип даних для Plant-

varchar50, Sector- integer, Landing_date- Datetime, Harvest_date- Datetime, Plant_ID- integer.

Таблиця Sector містить у собі 3 поля- Sector_number, Sensor_ID, Valve_ID, де Sector_number це первинний ключ. Тип даних для Sector_number- integer, Sensor_ID- integer, Valve_ID- integer.

Таблиця Sensor містить у собі 3 поля- Sensor_ID, Type, Sector де Sensor_ID це первинний ключ. Тип даних для Sensor_ID- integer, Type- varchar(50), Sector- integer.

Таблиця Measurement містить у собі 3 поля- Value, lock_time, Sensor_ID, де Value це первинний ключ. Тип даних для Value- integer, lock_time- time, Sensor_ID- integer.

Таблиця Valve містить у собі 3 поля- Valve_ID, Valve_Type, Sector, де Valve_ID це первинний ключ. Тип даних для Valve_ID- integer, Valve_Type- varchar(50), Sector- integer.

Таблиця Valve_condition містить у собі 4 поля- ID_valve_condition, Condition, status_change_time, Valve_ID, де ID_valve_condition це первинний ключ. Тип даних для ID_valve_condition- integer, Condition- integer, status_change_time- time, Valve_ID- integer.

Таблиця Event log містить у собі 2 поля- System_state, Work_time де це первинний ключ. Тип даних для System_state- integer, Work_time- time .

Зв’язки між таблицями :

Plant – Vegetative state - багато до одного.

Plant – Landing - багато до одного.

Landing - Sector - багато до багато.

Landing – Event Log - багато до багато.

Sector- Sensor - багато до одного.

Sector – Valve - багато до одного.

Valve- Valve_condition - багато до одного.

Sensor – Measurement - багато до одного.

Valve_condition – Event Log - багато до багато.

Measurements – Event Log - багато до багато.

Фізична модель була створена з логічної за допомогою інструмента “Engineer” Фізична модель бази даних містить у собі 13 таблиць (див. Рис.2.4.2):

1. Plant.
2. Vegetative_state.
3. Landing.
4. Sector.
5. Sensor.
6. Measurement.
7. Valve.
8. Valve_condition.
9. Event_Log.
10. Relation_3.
11. Relation_4.
12. Relation_9.
13. Relation_10.

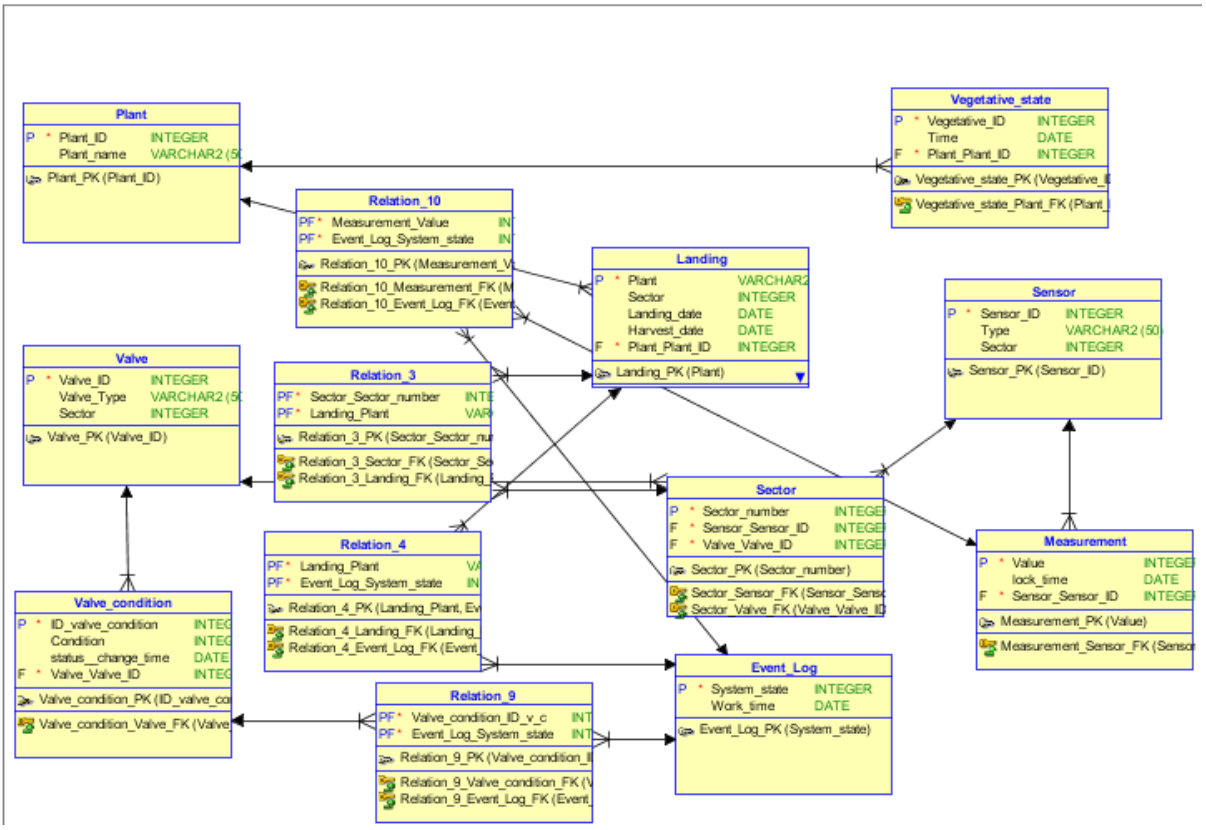

Рисунок 2.4.2 Фізична модель для контролю системи розумного поливу

При створені фізичної моделі з логічної були додані декілька суміжних таблиць Relation.

А саме: таблиця Relation_3 містить у собі 2 поля- Sector_Sector_number, Landing_Plant . Тип даних для Sector_Sector_number- integer, Landing_Plant- varchar2 (50) .

Таблиця Relation_4 містить у собі 2 поля- Landing_Plant, Event_Log_state. Тип даних для Landing_Plant- varchar (50), Event_Log_state- integer .

Таблиця Relation_9 містить у собі 2 поля- valve_condition_ID_v_c, Event_Log_System_state. Тип даних для alve_condition_ID_v_c- integer, Event_Log_System_state- integer.

Таблиця Relation_10 містить у собі 2 поля- Measurement_Value, Event_Log_System_state. Тип даних для Measurement_Value- integer, Event_Log_System_state- integer.

Зв’язки між таблицями :

Plant – Vegetative_state - багато до одного.

Plant – Landing - багато до одного.

Landing – Relation_3 - багато до одного.

Landing – Relation_4 - багато до одного.

Relation_3 – Sector - багато до одного.

Sector- Sensor - багато до одного.

Sector – Valve - багато до одного.

Valve- Valve_condition - багато до одного.

Sensor – Measurement - багато до одного.

Valve_condition – Relation_9 - багато до одного.

Measurements – Relation_10 - багато до одного.

Relation_4 – Event_Log - багато до одного.

Relation_9 – Event_Log - багато до одного.

Relation_10 – Event_Log - багато до одного.

Після цього, завдяки внутрішнім функціям Oracle Apex було створено веб-інтерфейс для перегляду цієї бази даних (див. Рис. 2.4.3).

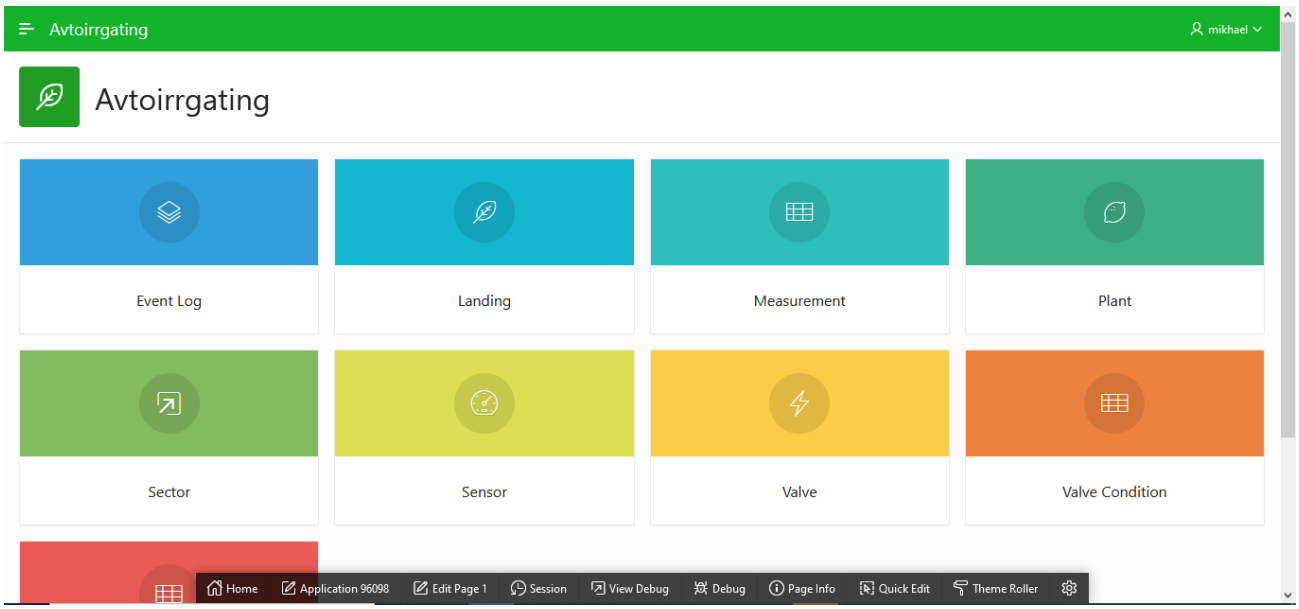

Рисунок 2.4.3 Створення веб-інтерфейсу для перегляду бази даних

Головне меню містить такі таблиці:

1. Event Log.
2. Landing.
3. Measurement.
4. Plant.
5. Sector.
6. Sensor.
7. Valve.
8. Valve Condition.
9. Vegetative State.

Також у лівій верхній частині меню можна знайти навігатор за допомогою якого буде зручніше знайти потрібну таблицю (Див. Рис. 2.4.4).

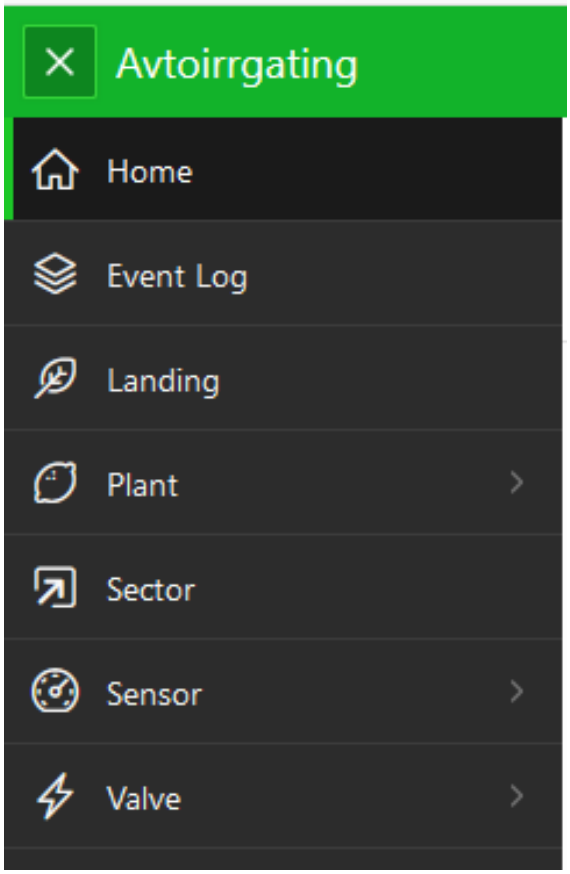

Рисунок 2.4.4 Навігатор

2.5 Створення алгоритмів роботи

Перед тим як програмувати систему, треба створити логіку її роботи. Її можна візуалізувати і зробити алгоритм роботи системи.

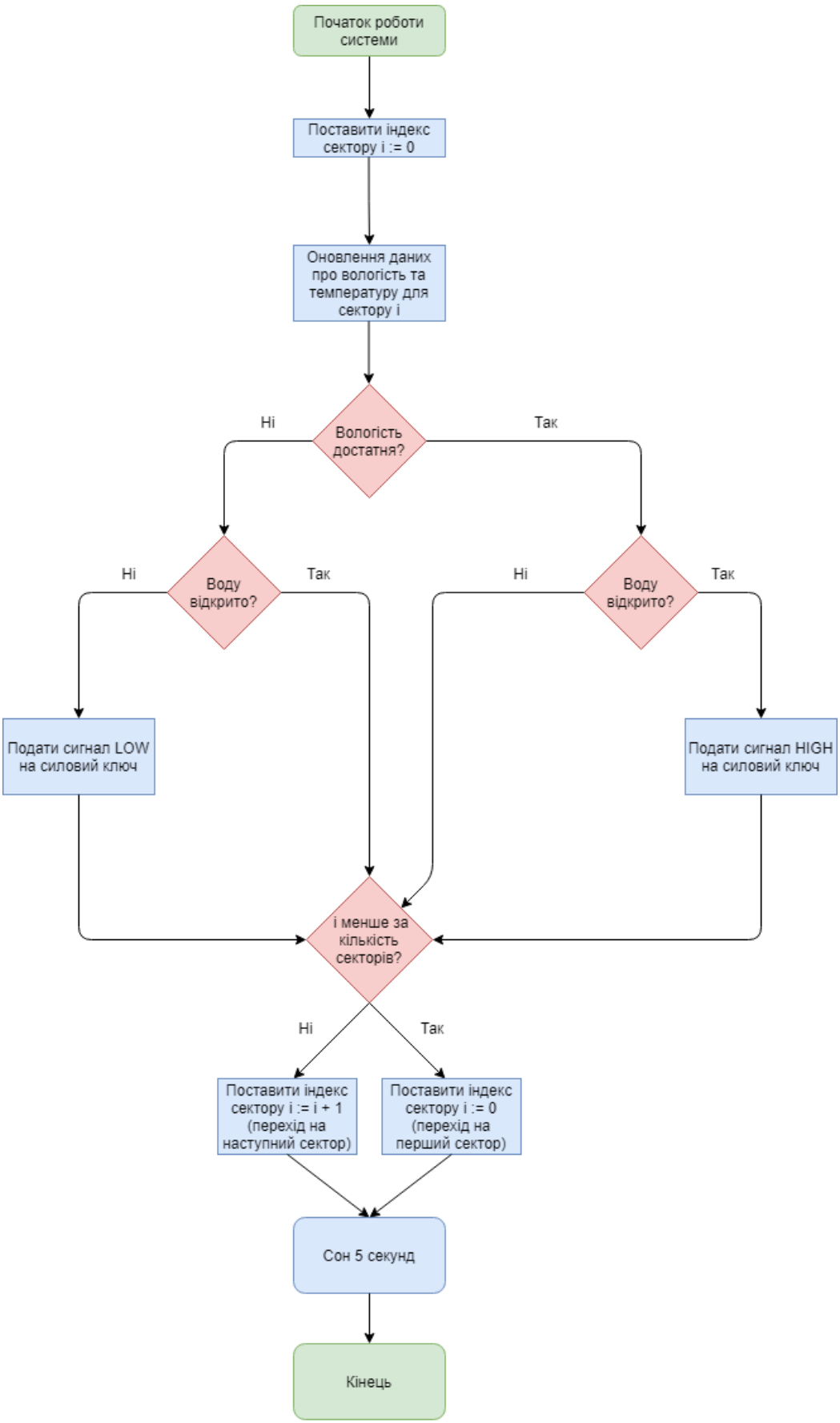

Рисунок 2.5.1 Алгоритм роботи розумного зрошування системи в компанії “Дісті”

Система працює на основі секторів. Кожен сектор має свій номер, датчик вологості і електромагнітний клапан, який подає воду на нього. Система працює по черзі для кожного сектору. Після початку роботи, система збирає дані з датчиків. З їх показників вона вирішує, що треба зробити для сектора (рис. 2.5.1). Після того, як всі операції були виконані в одному секторі - система переходить до іншого і так в циклі з певними затримками у часі.

Також був створений алгоритм роботи серверу з веб-інтерфейсом (рис.2.5.2).

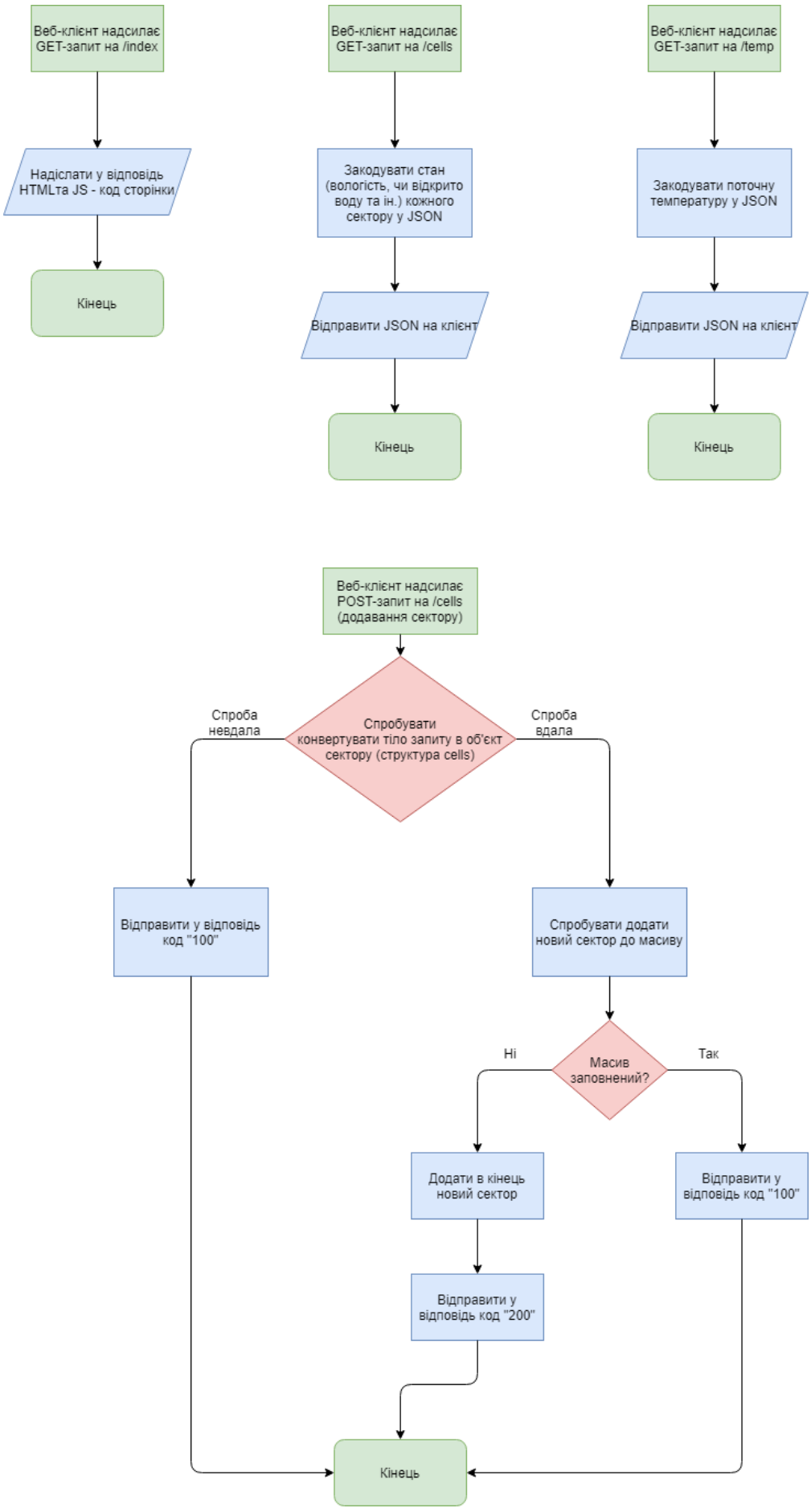

Рисунок 2.5.2 Алгоритм роботи сервера для контролю поливу

А ще був зроблений алгоритм роботи Телеграм-бота. Графічно було зображено логіку роботи бота (рис.2.5.3).

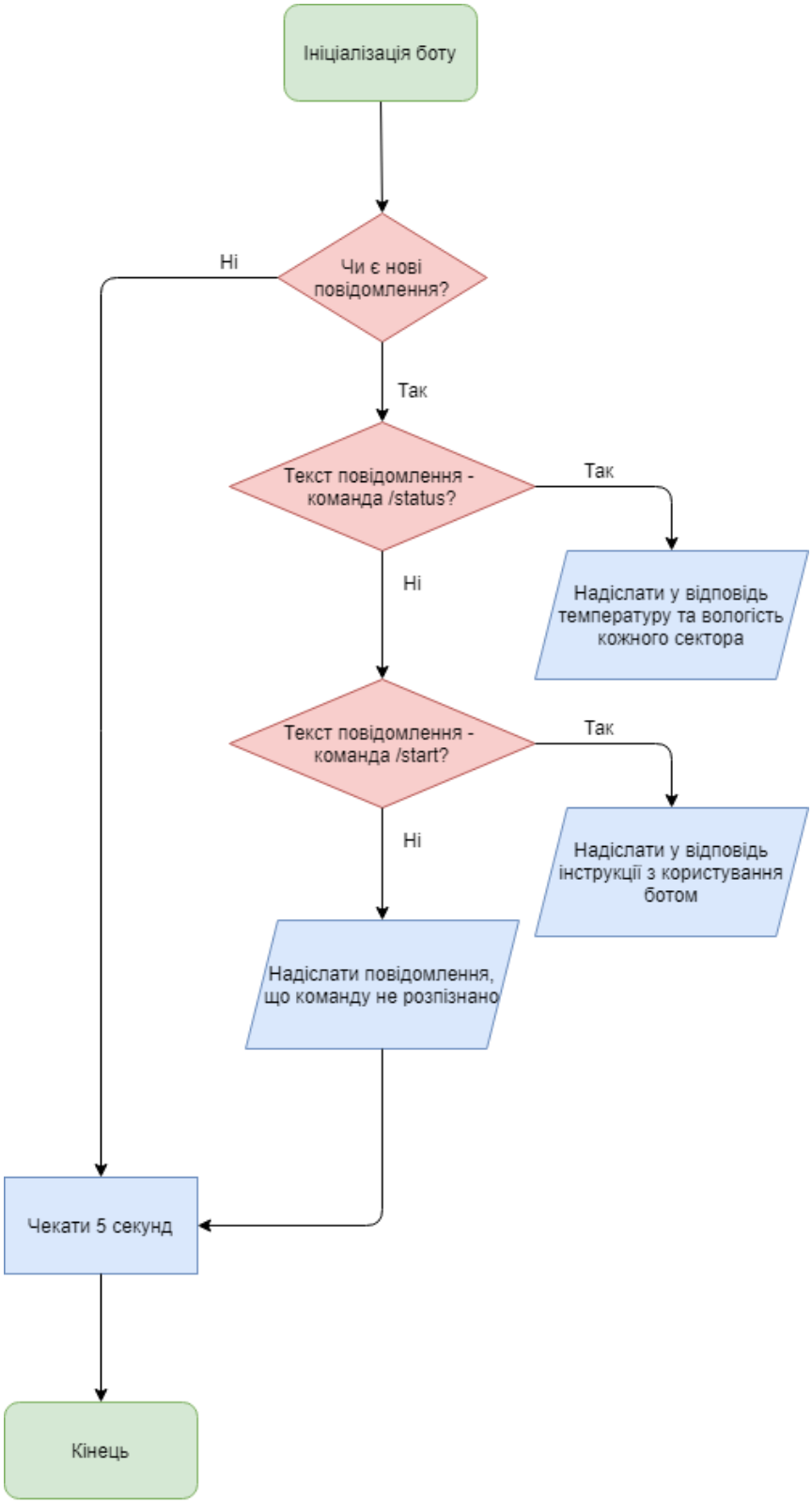

Рисунок 2.5.3 Алгоритм роботи Телеграм-боту для системи розумного зрошення

Бот аналізує чи поступили повідомлення на нього. Якщо ні, то через 5 секунд він перевіряє знову. Якщо на нього надійшло повідомлення /status, то він, у відповідь, надсилає температуру та вологість кожного сектора. Якщо було надіслано не вірне повідомлення, то він у відповідь надсилає, що команду не розпізнано.

2.6 Висновок по розділу

В другому розділі було створено концепт системи та поставлені цілі та задачі по її проектуванню. Було підібрано обладнання, яке найкраще підійшло для створення системи, а саме - була обрана керуюча плата для контролю зрошування, обрано датчики температури та вологи для збирання показників, обрано електромагнітний клапан для керування подачею води та силовий ключ, щоб керувати соленоїдами через мікроконтролер. Після того, як обладнання було підібрано, його було під'єднано в єдину систему. Плата була встановлена на breadboard, до неї було підключено датчики та силовий ключ. До силового ключа було підключено два електромагнітних клапана, які були підключені до мережі 220V. Клапани встановлені до ємності з водою, була прокладена крапельна лінія для крапельного поливу рослин. Також було створено базу даних для системи розумного поливу, створено логічну та реляційну модель в середовищі Oracle SQL Developer Data Modeler, а за допомогою додатка Oracle Apex було зроблено інтерфейс для зручного керування базою. Також були створені графічні алгоритми де відображається логіка роботи розумного поливу, серверу з веб-інтерфейсом та Телеграм-боту.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО КОДУ ДЛЯ РОЗУМНОГО ЗРОШУВАННЯ ТА СТВОРЕННЯ ДОДАТКІВ ДЛЯ МОНІТОРИНГУ.

3.1 Створення логіки роботи системи

В цьому розділі постає задача в створені програмного коду для отримання показників за датчиків вологості та температури, в створені логіки роботи силового ключа і електромагнітних клапанів та в створені додатків для моніторингу роботи системи. В програмі кожен сектор теплиці представляється екземпляром структури PlantCell на (рис. 3.1.1).

```
struct PlantCell {
  short id;
  short hSensorPin;
  short waterValve;
  short criticalPercent;
  short enoughPercent;
  bool valveIsOpen;
  short humidityPercent;
  PlantCell() {}
  PlantCell(short _id, short _hs, short _wv, bool _tr, short crP, short enP) {
    id = _id;
    hSensorPin = _hs;
    waterValve = _wv;
    valveIsOpen = false;
    humidityPercent = 0;
    criticalPercent = crP;
    enoughPercent = enP;
  }
  // true if humidity has changed, false otherwise
  bool ReadHumidity() {
    short h_buff = humidityPercent;
    humidityPercent = map(analogRead(hSensorPin), AIR_VALUE, WATER_VALUE, 0, 100);
    return h_buff != humidityPercent;
  }
};
```

Рисунок 3.1.1 структура PlantCell

Кожен сектор має свій унікальний ідентифікатор, свій датчик вологості, свої налаштування оптимальної вологості та свій соленоїд, який під’єднаний до неї. В структурі зберігається ця інформація, а також інформація про поточне значення вологості та про те, чи відкритий зараз клапан. Вона містить метод для

зчитування вологості з датчика (метод ReadHumidity на рисунку 3.1.1). Теплицю в цілому представлено масивом структури PlantCell.

Після успішного налаштування датчика вологості ґрунту постала задача програмування датчика температури (див. Рис. 3.1.2). Для підключення цього датчика було використано такі бібліотеки як: OneWire, DallasTemperature.

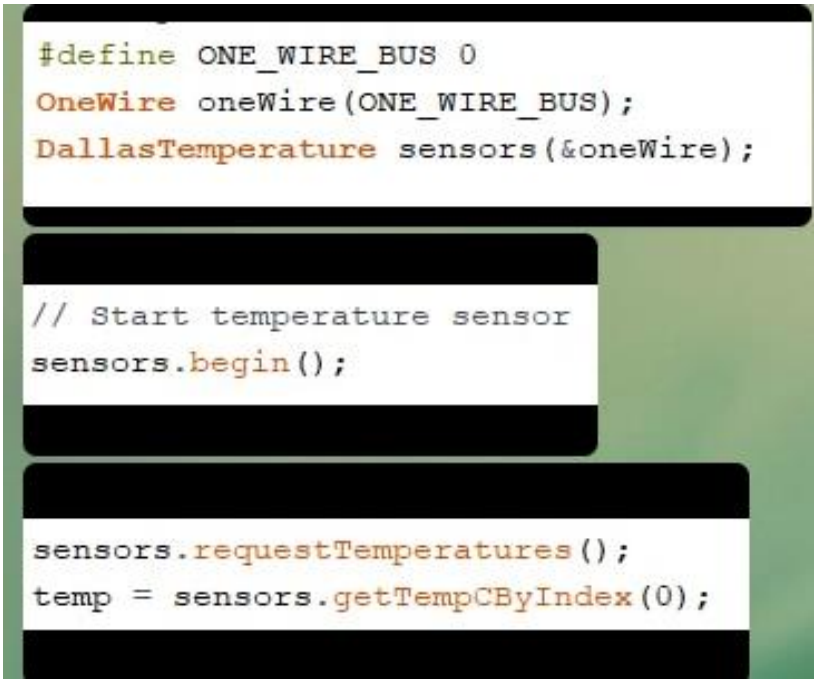

Рисунок 3.1.2 код для роботи датчика температури

Перші два фрагменти - це налаштування та запуск датчика відповідно, а останній - це зчитування температури .

Наступним кроком було створено програмний код для керування електромагнітними клапанами через силовий ключ (рис. 3.1.3). Силовий ключ подає воду, коли вологість у секторі замала і припиняє подачу, коли вологість досягає зазначеного у відповідному об’єкті PlantCell рівня.

```
static long currentMillis;
static int interval = 5000;
static float temp = 0;
// every 5 sec
if (millis() - currentMillis >= interval)
{
  for (short i = 0; i < cellCount; i++)
  {
    // update humidity
    cells[i].ReadHumidity();
    // if soil moisture too low and valve is closed, open valve
    if (!cells[i].valveIsOpen && cells[i].humidityPercent <= cells[i].criticalPercent) {
      Serial.print("Opened ");
      Serial.print(i);
      Serial.print(" with humidity ");
      Serial.println(cells[i].humidityPercent);
      digitalWrite(cells[i].waterValve, LOW);
      cells[i].valveIsOpen = true;
    }
    // if valve open and moisture is high enough, close valve
    if (cells[i].valveIsOpen && (cells[i].humidityPercent > cells[i].enoughPercent)) {
      Serial.print("Closed ");
      Serial.print(i);
      Serial.print(" with humidity ");
      Serial.println(cells[i].humidityPercent);
      digitalWrite(cells[i].waterValve, HIGH);
      cells[i].valveIsOpen = false;
    }
  }
  currentMillis = millis();
}
```

Рисунок 3.1.3 фрагмент коду для роботи силового ключа

3.2 Створення веб-інтерфейсу

Також при створені системи розумного зрошування треба було зробити код для підключення керуючої плати до інтернету по мережі WiFi (див. Рис. 3.2.1) для подальшого створення веб-інтерфейсу для моніторингу. Для створення цього фрагменту було використанні бібліотеки: ESP8266WiFi, ArduinoJson, ESPAsyncWebServer.

```
WiFi.begin(ssid, password);
// Wait for connection
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected..!");
Serial.print("Got IP: ");
Serial.println(WiFi.localIP());
```

Рисунок 3.2.1 фрагмент коду для підключення плати до WiFi.

Для моніторингу системи розумного зрошування було створено веб-інтерфейс. Він зручний у використанні і його можна увімкнути без зайвих завантажень з інтернету. Для того, щоб почати створювати треба підключити плату до мережі інтернету. Після того, як плата була підключена треба на ній створити локальний сервер за допомогою бібліотеки ESPAsyncWebServer (див. Рис. 3.2.2). Код для налаштування серверу був винесений в окрему функцію configServer ().(див. Рис 3.2.3).

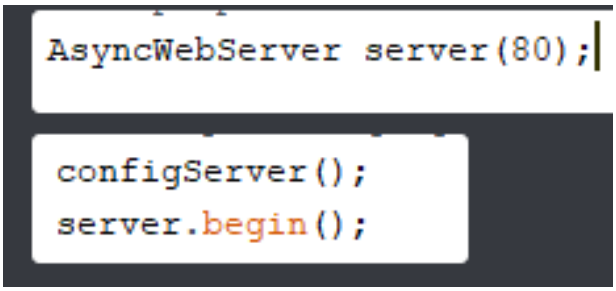

Рисунок 3.2.2 фрагмент коду для створення серверу.

```
// on loading the page, send HTML
server.on("/index", HTTP_ANY, [](AsyncWebServerRequest * request) {
  Serial.println("Got index request");
  request->send_P(200, "text/html", index_html);
  Serial.println("Sent HTML");
});

// on request, send cell updates
server.on("/cells", HTTP_GET, [](AsyncWebServerRequest * request) {
  request->send(200, "text/plain", createUpdatesJson(cellCount, cells));
});

// on request, send temperature updates
server.on("/temp", HTTP_GET, [](AsyncWebServerRequest * request) {
  sensors.requestTemperatures();
  request->send(200, "text/plain", String(sensors.getTempCByIndex(0)));
});

// on POST request, parse incoming data (a cell) and try to add the cell
server.on(
  "/",
  HTTP_POST,
[](AsyncWebServerRequest * request) {},
NULL,
[](AsyncWebServerRequest * request, uint8_t *data, size_t len, size_t index, size_t total) {
  Serial.println("POST request");
  String res = String((char *)data);
  PlantCell newCell = createCellFromString(res, cellCount);
  if (addCell(newCell)) {
    Serial.println("added: ");
    Serial.println(res);
    request->send(200);
  }
  else {
    Serial.println("add failed");
    request->send(100);
  }
});
```

Рисунок 3.2.3 функція для подальшого налаштування сервера.

Як видно на (рис. 3.2.3), при певному запиті на сервер відправляється код веб-сторінки (див. рис. 3.2.4 - 3.2.5).

```
        }
        function postCell(cell) {
            const jsonBody = JSON.stringify(cell);
            console.log("Sending POST with: ");
            console.log(jsonBody);
            fetch("/", {
                method: "POST",
                body: jsonBody
            }).then(response => {
                console.log("Response: ", response);
            });
        }
        function constructCell(_hPin, _wPin, _tReg, _critPercent, _enoughPercent) {
            const newCell = { hPin: _hPin, wPin: _wPin, tReg: _tReg, crP: _critPercent, enP: _enoughPercent };
            return newCell;
        }
        function loadTemperature() {
          fetch("/temp").then(function(response) {
            response.text().then((text) => {
              temp = parseInt(JSON.parse(text)) + 150;
              hdrTemp = document.querySelector('#temp');
              hdrTemp.innerHTML = temp + String.fromCharCode(176) + "C";
            });
          });
        }
        function loadCells() {
            const list = document.querySelector('#cells');
            var cells = [];
            fetch("/cells").then(function(response) {
                response.text().then(function(text) {
                    json = JSON.parse(text);
                    cells = json.data;
                    cells.forEach(cell => {
                        var currentCell = document.getElementById("cell" + cell.id);
                        cellElement = createHtmlCell(cell.id, cell.hPin, cell.hVal, cell.on, cell.crP, cell.wPin);
                        if(currentCell == null)
                        {
                            list.appendChild(cellElement);
                        }
                        else
                        {
                            list.replaceChild(cellElement, currentCell);
                        }
                    });
                });
            });
        }
        setInterval(loadCells, 5000);
        setInterval(loadTemperature, 5000);
    </script>
```

Рисунок 3.2.4 Javascript код сторінки.

```
<body>
    <header>
        <h2 id="temp" style="height: fit-content; margin-top: 0px;"></h3>
    </header>
    <div class="content">
        <div id="cells" class="cell-container">
            <form id = "addForm" class="cell" method="POST" action="/">
                <input type="text" name="hPin" id="hPin">
                <input type="text" name="wPin" id="wPin">
                <input type="text" name="tReg" id="tReg">
                <input type="text" name="crP" id="crP">
                <input type="text" name="hEn" id="hEn">
                <input type="submit" value="Submit">
            </form>
        </div>
    </div>
    <script>
       document.getElementById('addForm').addEventListener('submit', function(e) {
          e.preventDefault(); //to prevent form submission
          const hPin = document.getElementById('hPin').value;
          const wPin = document.getElementById('wPin').value;
          const tReg = document.getElementById('tReg').value;
          const crP = document.getElementById('crP').value;
          const hEn = document.getElementById('hEn').value;
          postCell(constructCell(hPin, wPin, tReg, crP, hEn));
       });
    </script>
</body>
```

Рисунок 3.2.5 html код сторінки

Також для роботи з веб-сервером було розроблено декілька допоміжних функцій: для додавання та видалення секторів (рис. 3.2.6) та для перетворення JSON-даних (рис. 3.2.7).

```
bool removeCell(short& id) {
  // Find cell with this id in array
  short i = 0;
  for (; i < cellCount && cells[i].id != id; i++) {
  }
  // Cell with this id not found
  if (i == cellCount)
  {
    return false;
  }
  // Else, remove the found cell
  for (; i < cellCount - 1; i++) {
    cells[i] = cells[i + 1];
  }
  cellCount--;
  return true;
}

// Add a cell
bool addCell(PlantCell& c) {
  if (cellCount >= cellCountMax)
    return false;
  else {
    cellCount++;
    cells[cellCount - 1] = c;
    return true;
  }
}
```

Рисунок 3.2.6 додавання і видалення секторів

```
// Parse JSONDocument and return a new PlantCel
PlantCell createCellFromString(String& json, short& id) {
  StaticJsonDocument<200> doc;
  deserializeJson(doc, json);
  short humidityPin = doc["hPin"];
  short waterValvePin = doc["wPin"];
  short timeRegulated = doc["tReg"];
  short criticalHumidity = doc["crP"];
  short enoughHumidity = doc["enP"];
  Serial.println(humidityPin);
  Serial.println(waterValvePin);
  Serial.println(timeRegulated);
  Serial.println(criticalHumidity);
  Serial.println(enoughHumidity);
  return PlantCell(id, humidityPin, waterValvePin, timeRegulated, criticalHumidity, enoughHumidity);
}

// Create a JSONDocument from PlantCell object
char * createUpdatesJson(short &cellCount, PlantCell* cells) {
  DynamicJsonDocument doc(2048);
  JsonArray data = doc.createNestedArray("data");
  for (short i = 0; i < cellCount; i++)
  {
    JsonObject cell = data.createNestedObject();
    cell["id"] = cells[i].id;
    cell["crP"] = cells[i].criticalPercent;
    cell["hEn"] = cells[i].enoughPercent;
    cell["hPin"] = cells[i].hSensorPin;
    cell["hVal"] = cells[i].humidityPercent;
    cell["wPin"] = cells[i].waterValve;
    cell["on"] = cells[i].valveIsOpen;
  }
  char res[2048];
  serializeJson(doc, res);
  return res;
}
```

Рисунок 3.2.7 перетворення JSON-даних

Після того, як всі налаштування були зроблені, було запущено сам веб-інтерфейс для перегляду (див. Рис. 3.2.8).

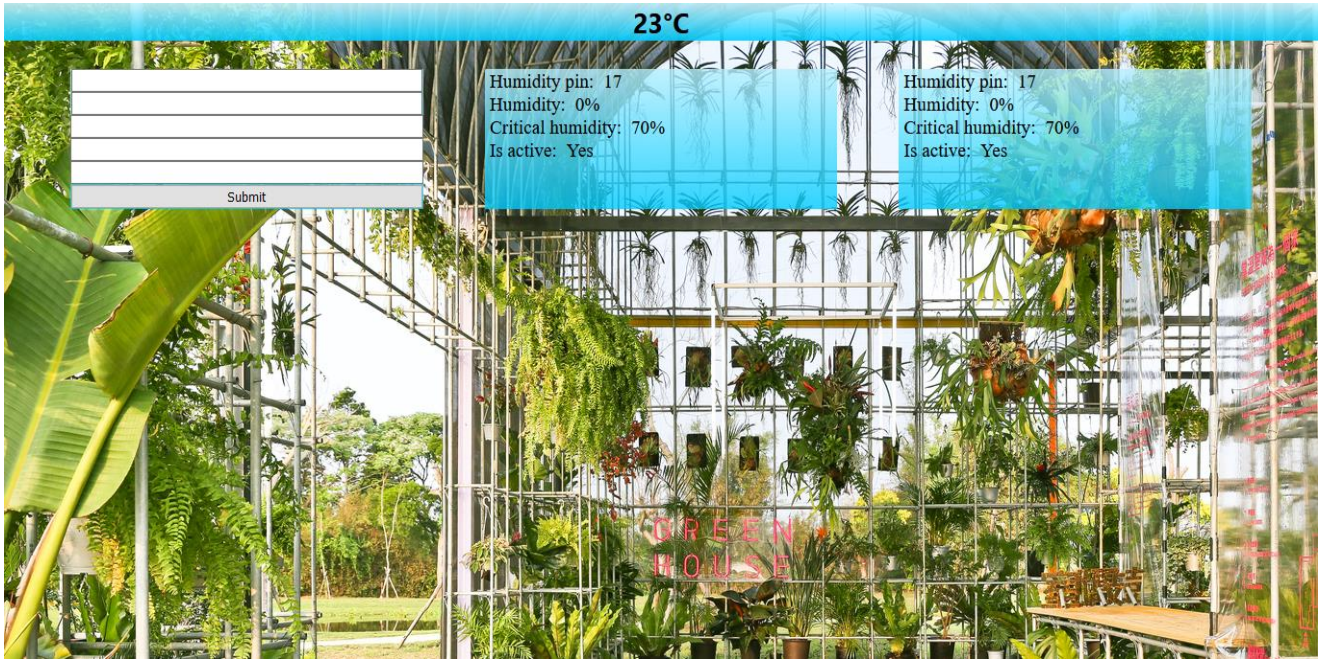

Рисунок 3.2.8 - вигляд веб-інтерфейсу

Зверху знаходиться значення температури всього об’єкту, а під ним - клітинки з секторами. Перша клітинка - це поле для створення нового сектору, а інші дві - вже робочі сектори. Humidity pin - це пін на платі, до якого підключений

датчик вологості, Humidity - рівень вологості ґрунту у секторі, Critical humidity - рівень вологості, на якому зрошування буде зупинятися, Is active - статус роботи електромагнітного клапана.

3.3 Створення Телеграм-боту

Для зручного моніторингу стану роботи системи, було вирішено створити бота в мессенджері “Telegram”. В порівнянні з веб-інтерфейсом, бот має свої переваги та недоліки. Наприклад, Телеграм-бот дуже зручний у використанні - для того, щоб почати роботу з ним, достатньо ввести команду /start у відповідному чаті з ботом. З іншого боку, Телеграм-бот не може передати ту кількість інформації, яку можна передати через веб-інтерфейс. Також бот обмежений в своїх можливостях, порівняно з веб-інтерфейсом.

Для початку, треба створити Телеграм-бота в BotFather чаті в Telegram. Треба ввести /start і в повідомлені обрати /newbot (див. Рис. 3.3.1).

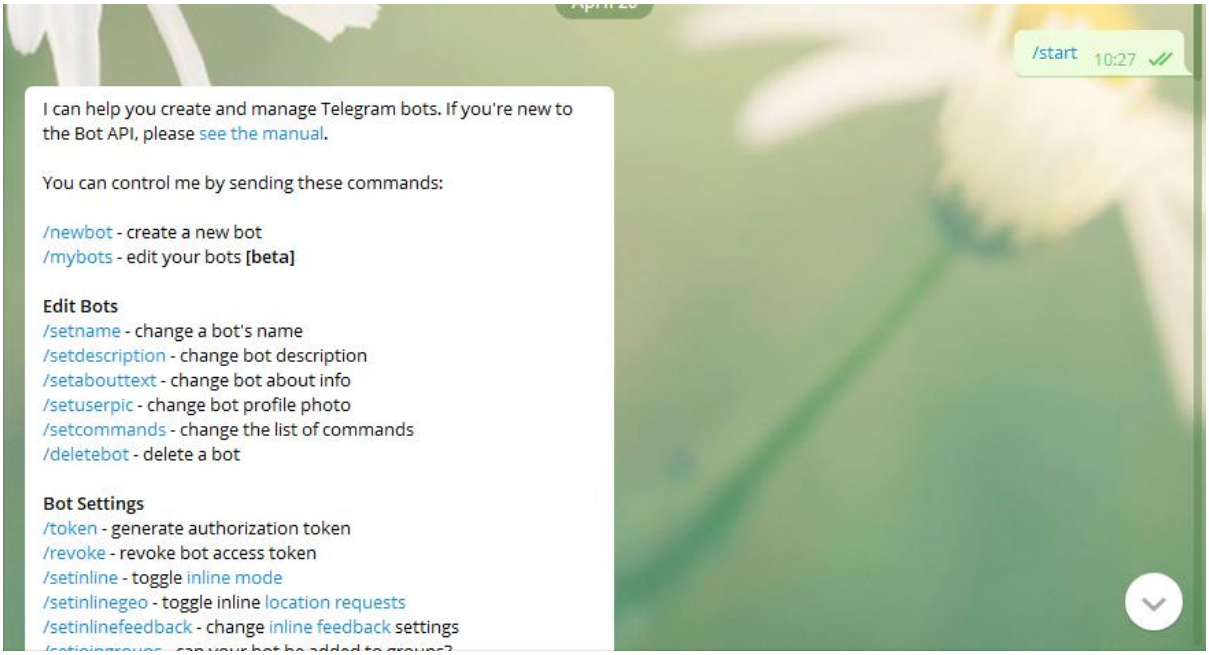

Рисунок 3.3.1 створення бота в BotFather

Після цього, треба дати ім’я цьому боту та username, за допомогою якого, користувачі зможуть підключатися до нього (див. Рис. 3.3.2).

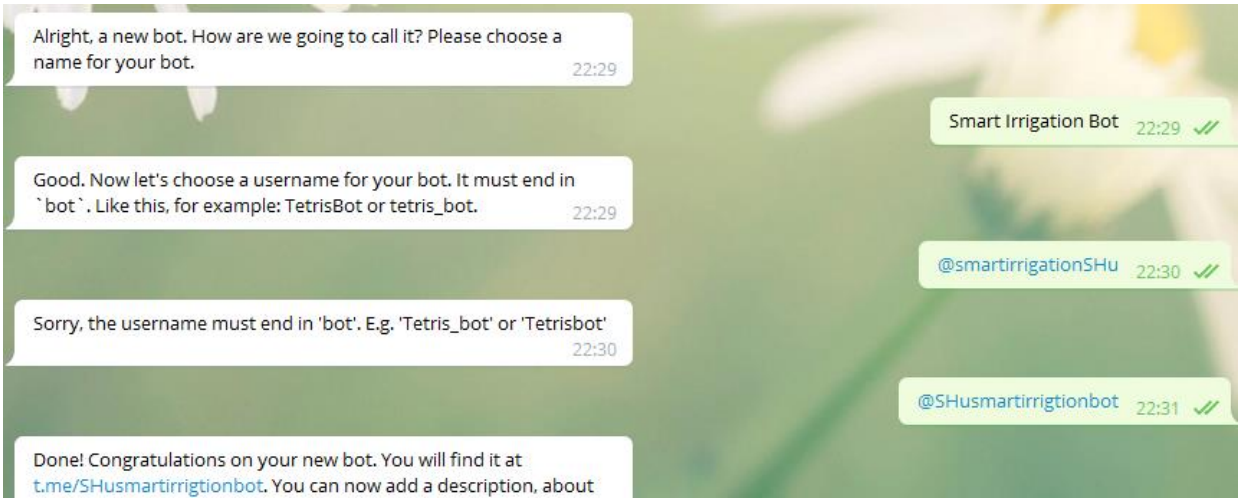

Рисунок 3.3.2 створення Username для бота

Після того, як боту було дано ім'я @SHusmartirrigtionbot, BotFather створив HTTP API token, за допомогою якого і буде під'днано плату Node MCU до телеграму.

Для створення Телеграм-боту на платі було використано бібліотеку UniversalTelegramBot та WiFiClientSecure. Спочатку було налаштовано бота - задано токен, який отриманий від BotFather (див. Рис. 3.3.3).

```
X509List cert(TELEGRAM_CERTIFICATE_ROOT);
WiFiClientSecure secured_client;
UniversalTelegramBot bot(BOT_TOKEN, secured_client);
```

Рисунок 3.3.3 перші налаштування бота

Після цього, було створено код з алгоритмом роботи бота. Створено логіку відправки повідомлень (див. Рис. 3.3.4).

```
if (millis() - bot_lasttime > BOT_MTBS)
{
  int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

  while (numNewMessages)
  {
    Serial.println("got response");
    handleNewMessages(numNewMessages);
    numNewMessages = bot.getUpdates(bot.last_message_received + 1);
  }

  bot_lasttime = millis();
}
```

Рисунок 3.3.4 логіка відправки повідомлень

Потім було створено код для реагування на команди користувача бота разом з командами у мессенджирі (Рис. 3.3.5).

```
void handleNewMessages(int numNewMessages)
{
  static String status_response;
  static int temp;
  Serial.print("handleNewMessages ");
  Serial.println(numNewMessages);

  for (int i = 0; i < numNewMessages; i++)
  {
    String chat_id = bot.messages[i].chat_id;
    String text = bot.messages[i].text;

    String from_name = bot.messages[i].from_name;
    if (from_name == "")
      from_name = "Guest";

    if (text == "/status")
    {
        sensors.requestTemperatures();
        temp = sensors.getTempCByIndex(0) + 150;
        status_response = "Temperature: ";
        status_response += temp;
        status_response += "°C\nHumidity:\n";
        for(short i = 0; i < cellCount; i++)
        {
          status_response += "Cell ";
          status_response += i;
          status_response += ":";
          status_response += " ";
          status_response += cells[i].humidityPercent;
          status_response += "%\n";
        }
        bot.sendMessage(chat_id, status_response, "");
    }

    if (text == "/start")
    {
      String welcome = "Welcome, " + from_name + ".\n";
      welcome += "This is my bachelor's thesis project, Smart Irrigation Bot.\n\n";
      welcome += "/ledon : to switch the Led ON (for diagnostics)\n";
      welcome += "/ledoff : to switch the Led OFF (for diagnostics)\n";
      welcome += "/status : get current irrigation status\n";
      bot.sendMessage(chat_id, welcome, "Markdown");
    }
  }
```

Рисунок 3.3.5 Код для реагування на команди користувача

Після всіх налаштувань бота у самому Telegram тепер можна використовувати його для моніторингу за роботою системи (див. Рис. 3.3.6).

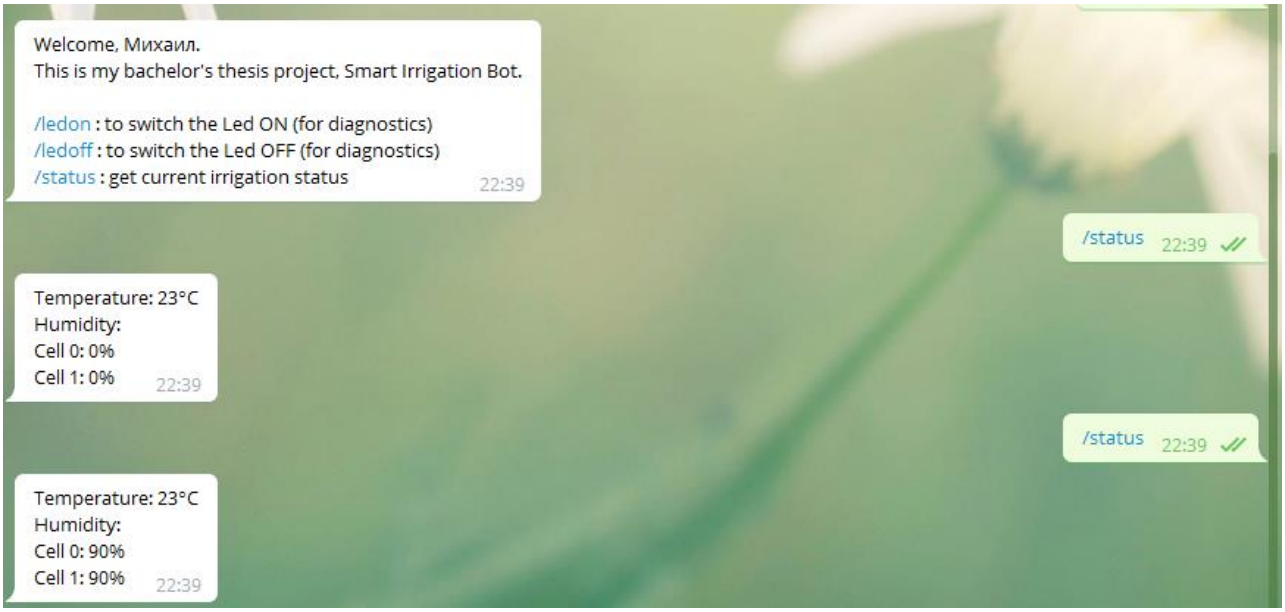

Рисунок 3.3.6 Команди і відповіді бота

3.4 Висновок по розділу

В третьому розділі було прописано програмний код для роботи системи. Було налаштовано програми для роботи датчиків вологості там температури. Була створена логіка для роботи силового ключа залежно від показань датчика вологості. В другому підрозділі плату було підключено до інтернету за допомогою технології WiFi та створено на ній веб-сервер в якому буде працювати веб-інтерфейс. В інтерфейсі зображено сектори, де відбувається полив, у вигляді клітинок. В клітинці відображається рівень вологості, до якого піна підключений датчик вологості, критичний рівень вологи та статус поливу сектора. Також зліва є поле для додавання нових секторів для поливу, а зверху відображується температура в об’єкті. Також створено Телеграм-бот для зручного перегляду показників датчику і роботи системи. В BotFather було дано ім’я боту і взято його токен, за допомогою якого, плату було під’єднано до нього. За допомогою бібліотек UniversalTelegramBot та WiFiClientSecure створено логіку роботи повідомлень в ньому.

ВИСНОВОК

1. У даній роботі було створено систему розумного зрошування у фермерському господарстві “Дісті” для підвищення врожайності. Перед тим як почати розробку було сформовано актуальність за задачу роботи.

2. В першому розділі було проаналізовано готові рішення в сфері автоматичного та розумного поливу. Були представлені найкращі розробки які існують на даний час у цій сфері. Також було проаналізовано типи зв’язку розумних речей у цій сфері, бо це є дуже важливим фактором при створенні системи, так як ці зв’язки мають свої недоліки і переваги, а також область застосування, яке не завжди є глобальним рішенням для всього. Після цього, досліджено сучасні мікроконтролери для створення системи розумного поливу. Були відібрані та порівняні, ті плати, які найкраще підходять для реалізації цього проекту.

3. У другому розділі було створено концептуальну модель системи розумного поливу. Після створення моделі було підібрано обладнання для того, щоб зібрати цю систему. Було обрано плату, датчики температури та вологості, електромагнітні клапани та силові ключі для їх контролю. Було створено базу даних в середовищі Oracle Data Modeler та веб-інтерфейс для неї в Oracle APEX. Потім, ці предмети було з’єднано між собою і, фізично, система була зібрана. Також були створені алгоритми: логіки роботи розумного зрошування, логіки роботи серверу та логіки роботи Телеграм-бота.

4. В третьому розділі було створено програмний код для збирання показників з датчиків, та створено логіку роботи системи. Після цього, треба було створити веб-інтерфейс для моніторингу за системою. Він має у собі сектори, які відображаються як клітинки де відбуваються всі процеси. Зверху інтерфейсу відображається температура на об’єкті, а зліва є поле для створення нового сектору. Після цього було створено Телеграм-бота. Він видає, в режимі реального часу, інформацію щодо показників датчиків та стану поливу, за допомогою команди /status в чаті з ботом.

5. Під час виконання роботи, всі задачі були виконані, система розумного поливу була створена, та працює самостійно, без людського втручання. Всі прилади були підключенні, створені додатки для моніторингу та налаштована система зрошування.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. HidroPoint [Електронний ресурс] / Режим доступу: https://www.hydropoint.com/what-is-smart-irrigation/#:~:text=Smart%20Irrigation%20Controllers%20Save%20Water%20and%20Money&text=Several%20controlled%20research%20studies%20indicate,from%2030%20to%2050%20percent.&text=Outdoor%20water%20savings%20were%20calculated,were%20adjusted%20for%20weather%20conditions (дата звертання 20.04.2021).

2. it_lynx [Електронний ресурс] / Режим доступу: http://www.it-lynx.com/products/design-construction-and-automation-of-irrigation-systems/ (дата звертання 21.04.2021).

3. Amazon, Aifro WaterEco [Електронний ресурс] / Режим доступу: https://www.amazon.com/Aifro-WaterEco-Irrigation-Controller-screen/product-reviews/B017223KIM (дата звертання 22.04.2021).

4. Baselines systems [Електронний ресурс] / Режим доступу: https://www.baselinesystems.com/products.php/basestation-3200 (дата звертання 22.04.2021).

5. Proxis.ua [Електронний ресурс] / Режим доступу: https://www.proxis.ua/ru/product/fieldbus-ethernet-io-modules-advantech-ADAM-6100 / (дата звертання 23.04.2021).

6. Networkworld WiFi tech [Електронний ресурс] / Режим доступу: https://www.networkworld.com/article/3196191/wifi-s-evolving-role-in-iot.html#:~:text=Role%20of%20WiFi%20in%20IoT,connectivity%20with%20cross%2Dvendor%20interoperability (дата звертання 23.04.2021).

7. Aeris, bluetooth for iot [Електронний ресурс] / Режим доступу: https://www.aeris.com/news/post/bluetooth-for-iot/ (дата звертання 24.04.2021).

8. Digi, ZigBee solution [Електронний ресурс] / Режим доступу: https://www.digi.com/solutions/by-technology/zigbee-wireless-standard#:~:text=Zigbee%20is%20a%20wireless%20technology,low%2Dpower%20

wireless%20IoT%20networks.&text=The%20protocol%20allows%20devices%20to, battery%20life%20lasting%20several%20years (дата звертання 25.04.2021).

9. Kerlink LoRaWAN [Електронний ресурс] / Режим доступу: https://www.kerlink.com/lora/ (дата звертання 25.04.2021).

10. Arduino uno [Електронний ресурс] / Режим доступу: https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-uno.html (дата звертання 26.04.2021).

11. Arduino Nano [Електронний ресурс] / Режим доступу: https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardNano (дата звертання 26.04.2021).

12. Ukraino Uno [Електронний ресурс] / Режим доступу: https://www.mini-tech.com.ua/ukraino-uno (дата звертання 26.04.2021).

13. Node MCU [Електронний ресурс] / Режим доступу: https://arduinomaster.ru/platy-arduino/esp8266-nodemcu-v3-lua/ (дата звертання 27.04.2021).

14. GeekMatic [Електронний ресурс] / Режим доступу: http://geekmatic.in.ua/ (дата звертання 28.04.2021).

15. GeekMatic soil moisture sensor [Електронний ресурс] / Режим доступу: http://geekmatic.in.ua/soil_hygrometer (дата звертання 28.04.2021).

16. GeekMatic soil moisture sensor [Електронний ресурс] / Режим доступу: http: http://geekmatic.in.ua/datchik_vlazhnosti_pochvi_emkostniy (дата звертання 28.04.2021).

17. DS1820 [Електронний ресурс] / Режим доступу: http://tec.org.ru/publ/stati_po_ehlektroinke/mikroskhemy/o_termodatchikakh_ds1820_ds18s20_ds18b20/8-1-0-14 (дата звертання 29.04.2021).

18. Prom, латунний клапан [Електронний ресурс] / Режим доступу: https://prom.ua/ua/p1310788076-latunnyj-elektromagnitnyj-klapan.html?utm_source=google_product&utm_medium=cpc&utm_term=&utm_content=da&utm_campaign=KT_cpc_1,1_1_&gclid=CjwKCAjwqcKFBhAhEiwAfEr7zQFZwEXwsOL4nISB-

03rYqUa1MBeaYfqZdYV_nI349m_AAowpbn43xoCxBUQAvD_BwE (дата звертання 30.04.2021).

19. Domel, електромагнітний клапан [Електронний ресурс] / Режим доступу: https://domel.com.ua/zapchasti-dlya-bytovoy-tekhniki/zapchasti-dlya-stiralnykh-posudomoechnykh-mashin/elektromagnitnye-klapany/elektromagnitnyy-klapan-stiralnoy-mashiny-universalnyy-1-180-c00194396/ (дата звертання 30.04.2021).

20. AmperMarket, power relay електромагнітний клапан [Електронний ресурс] / Режим доступу: https://ampermarket.kz/relay/1-channel-relay-module/ (дата звертання 01.05.2021).

21. Arduino Telegram bot [Електронний ресурс] / Режим доступу: https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot (дата звертання 02.05.2021).

22. Dalas Temperature [Електронний ресурс] / Режим доступу: https://www.arduino.cc/reference/en/libraries/dallastemperature/ (дата звертання 02.05.2021).

23. WiFi libary [Електронний ресурс] / Режим доступу: https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/ESP8266WiFi.h (дата звертання 02.05.2021).

24. OneWire [Електронний ресурс] / Режим доступу: https://github.com/PaulStoffregen/OneWire (дата звертання 03.05.2021).

25. Liquid Crystal [Електронний ресурс] / Режим доступу: https://github.com/arduino-libraries/LiquidCrystal (дата звертання 06.05.2021).

26. Network Protocols Handbook 2nd Edition. 2004 - 2005 Javvin Technoiosies Inc.

27. Ли П. Архитектура интернета вещей / Перри Ли. – М. : ДМК Пресс, 2018. – 454 с.

28. Serpanos D. Internet-of-Things (IoT) Systems: Architectures, Algorithms, Methodologies / Dimitrios Serpanos, Marilyn Wolf. – New York City : Springer, 2018. – 107 p.

29. Milenkovic M. Internet of Things: Concepts and System Design / Milan Milenkovic. – New York City: Springer, 2020. – 315 p.

30. Javed A. Building Arduino Projects for the Internet of Things. Experiments with Real-World Applications / Adeel Javed. – 1st ed. – New York City: Apress, 2016. – 307 p.

31. Monk S. Programming Arduino: Getting Started with Sketches / Simon Monk. – 2nd Ed. – New York City: McGraw-Hill Education, 2016. – 192 p.

32. Knight S. Arduino for Beginners: Step-by-Step Guide to Arduino (Arduino Hardware & Software) [e-book] / Simon Knight. – 2018. – 139 p.

33. Лисенко В. П. Інтернет речей: метод. вказівки / Лисенко В. П., Лендєл Т. І., Грищенко В. О., Удовенко О. О., Кіктєв М. О. – К.: НУБіП, 2020. – 50 с.

34. Олещенко Л. М. Програмування пристроїв інтернету речей : лабор. практикум / Л. М. Олещенко, Я. В. Хіцко. – К.: КПІ ім. Ігоря Сікорського, 2019. – 47 с.

35. Rose D. Enchanted Objects: Design, Human Desire, and the Internet of Things / David Rose. – New York City: Scribner, 2014. – 320 p.

36. Грінгард С. Інтернет речей / Семюель Грінгард. – Х.: Клуб сімейного дозвілля, 2018. – 176 с.

37. Rose K. The Internet of Things: An Overview [Електрон. ресурс] / Karen Rose, Scott Eldridge, Lyman Chapin; The Internet Society (ISOC). – 2015. – 80 p. – Режим доступу: https://www.internetsociety.org/wp-content/uploads/2017/08/ISOC-IoT-Overview-20151221-en.pdf.

38. Evans D. The Internet of Things [Електрон. ресурс] / Dave Evans; Cisco IBSG. – 2011. – 11 p. – Режим доступу: http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

39. Офіційний сайт фірми Texas Instruments / [Електронний ресурс]. – Режим доступу: http://www.ti.com/.

40. Характеристики мікроконтролерів TM4C123 / [Електронний ресурс]. – Режим доступу: http://www.ti.com/tool/EK-TM4C123GXL.

41. Офіційний сайт операційної системи реального часу FreeRTOS / [Електронний ресурс]. – Режим доступу: http://www.freertos.org/.

42. Форкун Ю. В. Інформатика: навч. посіб. / Ю. В. Форкун, Н. А. Длугунович. - – Львів: Видавництво «Новий світ – 2000», 2018. – 464 с.

43. Комп'ютери та комп'ютерні технології: навч. посіб. / Ю. Б. Бродський, К. В. Молодецька, О. Б. Борисюк,І. Ю. Гринчук. – Житомир : Вид-во «Житомирський національний агроекологічний університет», 2016. – 186 с.

44. Інформаційні технології [навчальний посібник] /О.Г. Кузьмінська, С.Г. Литвинова , Т.П. Саяпіна//-К: ЦП «Компрінт», 2017.-290 с. Видання друге - перероблене і доповнене.

45. Інформатика та інформаційні технології у цивільній безпеці: Практикум / [Маляров М.В, Гусева Л.В., Паніна О.О. та ін.]; Під заг. ред. М.В. Малярова. - Харків: НУЦЗ України, 2015. - 330 с.

46. Інформатика та інформаційні технології у цивільній безпеці: Практикум / [Маляров М.В, Гусева Л.В., Паніна О.О. та ін.]; Під заг. ред. М.В. Малярова. - Харків: НУЦЗ України, 2015. - 336 с.

47. Рожко Г. "Операційні системи та їх обслуговування"/Г Рожко. — Тернопіль: Технічний коледж ТНТУ, 2020 р. — 72 с.

48. The Java Tutorials [Електронний ресурс] – Режим доступу: https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html.

49. Галкін О.В., Катеринич Л.О., Шкільняк О.С. Програмування на Java 8. Навчальний посібник. – Київ: Вид-во «Логос», 2017. – 186 с.

50. Кадомський К.К. Java. Теорія і практика: навчальний посібник для студентів природничих спеціальностей університетів /, Ніколюк П.К. – Вінниця: Донну, 2019. – 197 с.

ДОДАТОК А

Рисунок А 1 Слайд 1 Титульний лист

Рисунок А 2 Мета роботи

Актуальність роботи

У сучасному світі зберігання води та отримання максимальної кількості врожаю - є дуже важливим завданням. Популяція людей зростає кожного дня і рішення які використовувались століття тому, вже не можна використовувати у сучасному світі.

Розумні автоматизовані системи вирішують такі проблеми як - надмірне використання води, точне зрошування, моніторинг поливу, дистанційне керування процесом. Кожна рослина отримує рівно стільки води, скільки їй потрібно, це запобігає гниттю коріння, появі захворювань у рослин та запобіганню росту бур'яна навколо саджанця.

Моніторинг поливу допомагає слідкувати за зрошуванням рослин, та аналізувати ситуацію при вирощуванні. За допомогою дистанційного керування можна керувати зрощенням віддалено, цим може займатися одна людина у будь-якій точці світу, також є можливість моніторити декілька об'єктів одночасно, що дає змогу вести господарство, яке не прив'язано до одного місця.

Рисунок А 3 Актуальність роботи

Дослідження готових рішень

Adam 6220 від Advantech

WaterECO від Aifro

NOD irrigation від IT-LYNX

BaseStation3200 від BaseLine

Рисунок А 4 Готові рішення

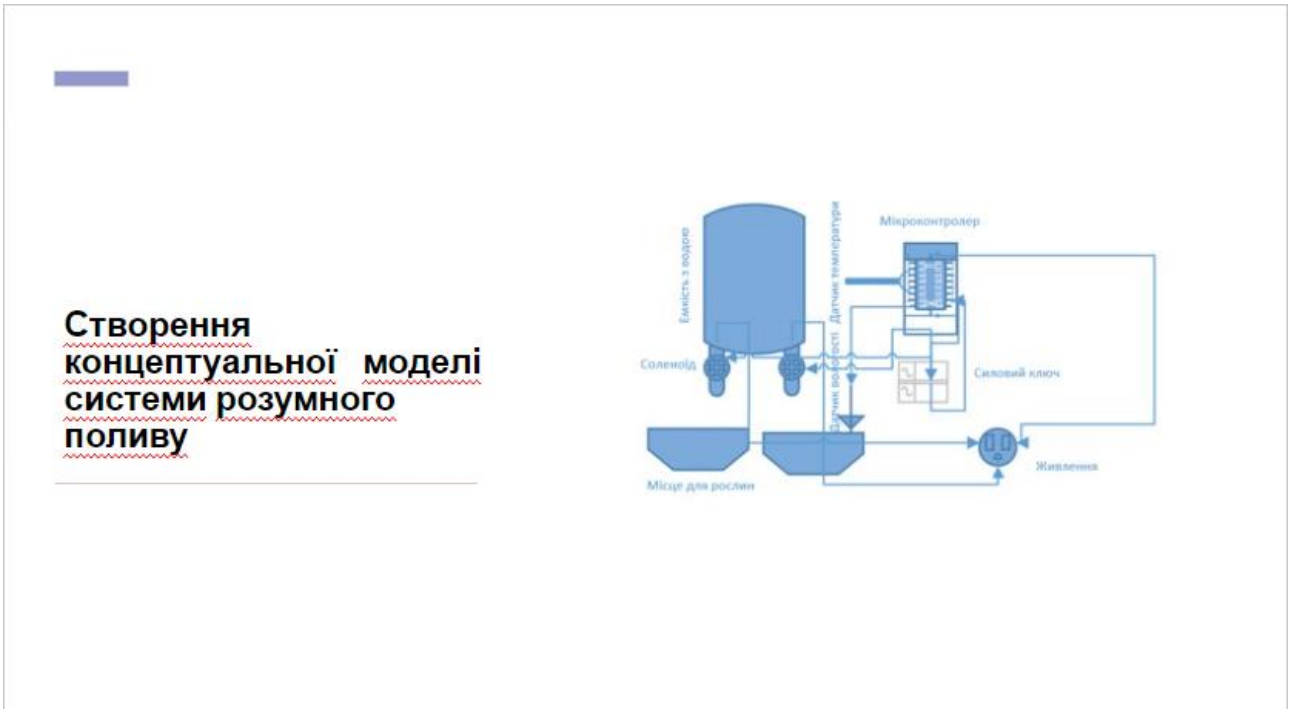

Рисунок А 5 Модель в Visio

Рисунок А 6 Підбір Обладнання

Рисунок А 7 Створення системи

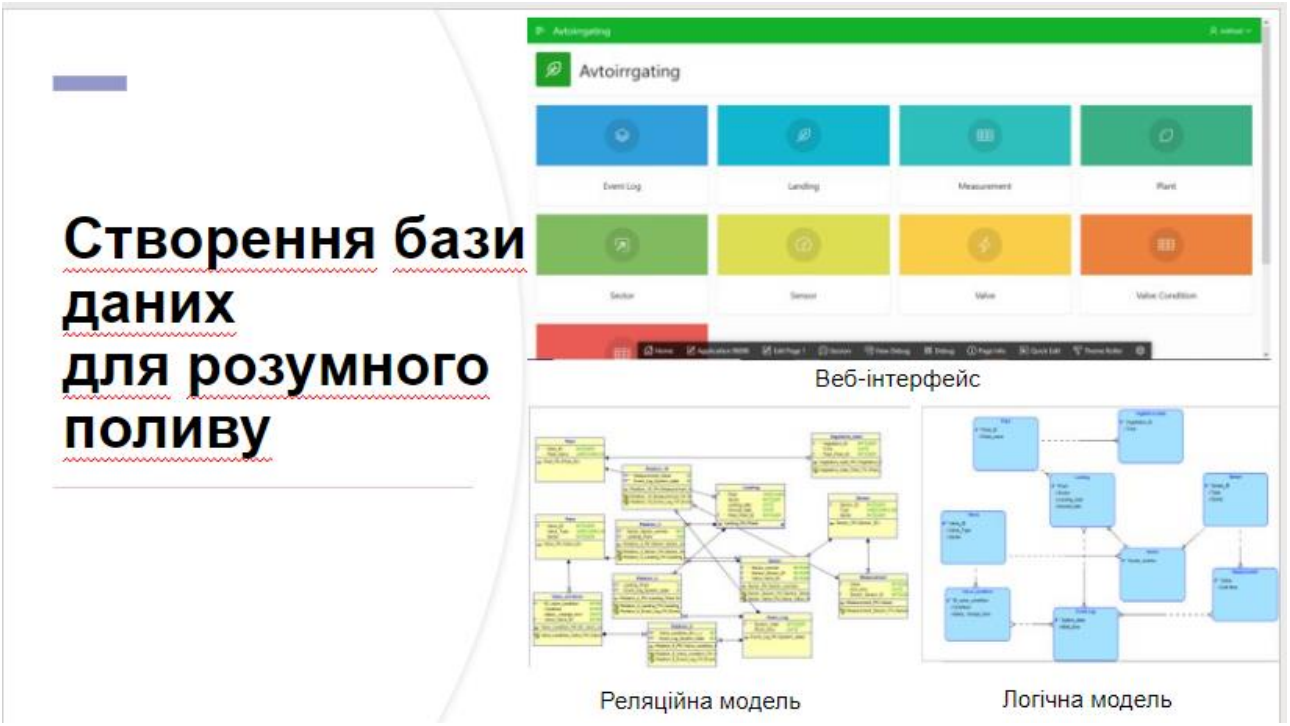

Рисунок А 8 Створення бази даних

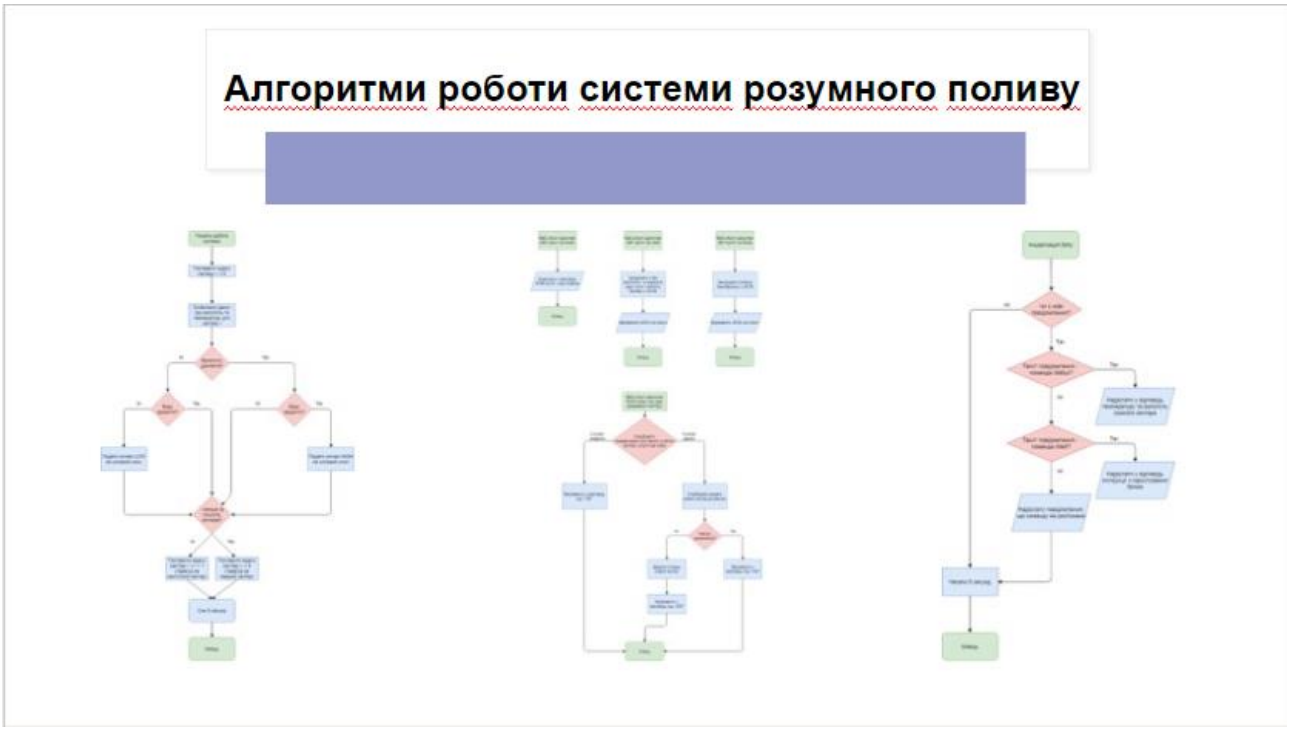

Рисунок А 9 Алгоритми

Рисунок А 10 Створення веб-інтерфейсу

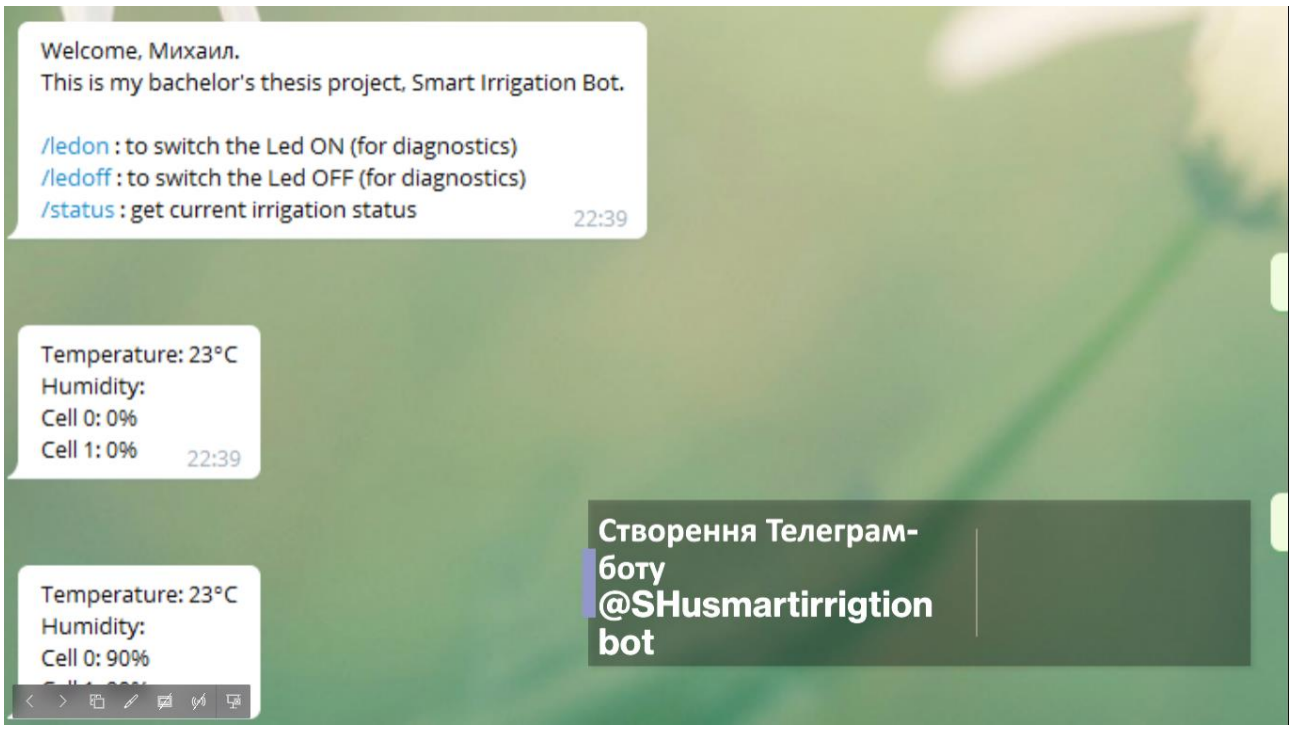

Рисунок А 11 Створення Телеграм-боту

Висновок

- У даній роботі було створено систему розумного зрошування у фермерському господарстві "Дісті" для підвищення врожайності.
- Було досліджено готові рішення в сфері автоматичного та розумного поливу.
- Було створено концептуальну схему системи зрощування.
- Було підібрано обладнання для того, щоб зібрати цю систему. Було обрано плату, датчики температури та вологості, електромагнітні клапани та силові ключі для їх контролю.
- Було створено логічну та реляційну модель бази даних розумного поливу та веб-інтерфейс для керування та перегляду за таблицями бази.
- Створено алгоритми роботи системи, а саме: алгоритм роботи розумного зрошування, алгоритм роботи сервера та алгоритм роботи Телеграм-бота.
- Після цього, було створено веб-інтерфейс для моніторингу за системою. Він має у собі сектори, які відображаються як клітинки де відбуваються всі процеси. Зверху інтерфейсу відображається температура на об'єкті, а зліва є поле для створення нового сектору.
- Після цього було створено Телеграм-бота. Він видає, в режимі реального часу, інформацію щодо показників датчиків та стану поливу, за допомогою команди /status в чаті з ботом.
- Під час виконання роботи, всі задачі були виконані, система розумного поливу була створена, та працює самостійно, без людського втручання. Всі прилади були підключенні, створені додатки для моніторингу та налаштована система зрошування.

Рисунок А 12 Висновок

ДОДАТОК Б

Фрагмент коду для Телеграм-боту

```
#include <ESP8266WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <UniversalTelegramBot.h>
#include <WiFiClientSecure.h>

#define WIFI_SSID "TP-LINK_3E2D8C"
#define WIFI_PWD ""
#define BOT_TOKEN ""
#define BOT_MTBS 1000

X509List cert(TELEGRAM_CERTIFICATE_ROOT);
WiFiClientSecure secured_client;
UniversalTelegramBot bot(BOT_TOKEN, secured_client);

bool ledStatus = 0;

// Temperature sensor
#define ONE_WIRE_BUS 0
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// WiFi
const char* ssid = "TP-LINK_3E2D8C";
const char* password = "";
```

```
// value of humidity sensor in air and water
#define AIR_VALUE 1024
#define WATER_VALUE 640


#define DEFAULT_CRITICAL_PERCENT 50
#define DEFAULT_ENOUGH_PERCENT 80


struct PlantCell {
  short id;

  short hSensorPin;
   short waterValve;
  short criticalPercent;
  short enoughPercent;
    bool valveIsOpen;
  short humidityPercent;
  PlantCell() {}
  PlantCell(short _id, short _hs, short _wv, bool _tr, short crP, short enP) {
    id = _id;
    hSensorPin = _hs;
    waterValve = _wv;
    valveIsOpen = false;
    humidityPercent = 0;
    criticalPercent = crP;
    enoughPercent = enP;
  }
    bool ReadHumidity() {
```

```
    short h_buff = humidityPercent;
    humidityPercent = map(analogRead(hSensorPin), AIR_VALUE, WATER_VALUE, 0, 100);
    return h_buff != humidityPercent;
  }
};

short cellCount = 0;
const short cellCountMax = 15;
PlantCell cells [cellCountMax] = {};

// бот
/*WiFiClientSecure net_ssl;
  TelegramBot bot (BotToken, net_ssl);*/

void setup() {

  cells[0] = PlantCell(0, A0, D4, false, 70, 81);
  cells[1] = PlantCell(1, A0, D7, false, 70, 81);
  cellCount = 2;

  // Initialize all to 0 (HIGH because idk the machine spirit is angry and it works that way)
    for (short i = 0; i < cellCount ; i++)
  {
    pinMode(cells[i].waterValve, OUTPUT);
    digitalWrite(cells[i].waterValve, HIGH);
  }
```

```
  Serial.begin(115200);
  // Wait for Serial to start
  while (!Serial);
  // Connect to WiFi
  Serial.println("Connecting to ");
  Serial.println(ssid);
  configTime(0, 0, "pool.ntp.org");      // get UTC time via NTP
  secured_client.setTrustAnchors(&cert); // Add root certificate for api.telegram.org
  WiFi.begin(ssid, password);
  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");
  Serial.println(WiFi.localIP());

  // Start temperature sensor
  sensors.begin();

}

void loop() {
  static long currentMillis;
  static int interval = 5000;
  static float temp = 0;
  unsigned static long bot_lasttime;
```

```
if (millis() - bot_lasttime > BOT_MTBS)
{
  int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

  while (numNewMessages)
  {
    Serial.println("got response");
    handleNewMessages(numNewMessages);
    numNewMessages = bot.getUpdates(bot.last_message_received + 1);
  }

  bot_lasttime = millis();
}

// every 5 sec
if (millis() - currentMillis >= interval)
{
  // Read temperature
  sensors.requestTemperatures();
  temp = sensors.getTempCByIndex(0);
  /* sensors.requestTemperatures();
    temp = sensors.getTempCByIndex(0);
    Serial.print(temp);
    Serial.println("ºC"); */
  for (short i = 0; i < cellCount; i++)
  {
    // update humidity
    cells[i].ReadHumidity();
```

```
      // if soil moisture too low and valve is closed, open valve
      if (!cells[i].valveIsOpen && cells[i].humidityPercent <= cells[i].criticalPercent) {
        Serial.print("Opened ");
        Serial.print(i);
        Serial.print(" with humidity ");
        Serial.println(cells[i].humidityPercent);
        digitalWrite(cells[i].waterValve, LOW);
        cells[i].valveIsOpen = true;
      }
      // if valve open and moisture is high enough, close valve
      if (cells[i].valveIsOpen && (cells[i].humidityPercent > cells[i].enoughPercent)) {
        Serial.print("Closed ");
        Serial.print(i);
        Serial.print(" with humidity ");
        Serial.println(cells[i].humidityPercent);
        digitalWrite(cells[i].waterValve, HIGH);
        cells[i].valveIsOpen = false;
      }
    }
    currentMillis = millis();
  }
}

void printHumidity(short &i, short &value) {
  Serial.print(i);
  Serial.print(": ");
  Serial.print(value);
  Serial.println("%");
}
```

```
// TODO: delete cell
bool removeCell(short& id) {
  // Find cell with this id in array
  short i = 0;
  for (; i < cellCount && cells[i].id != id; i++) {
  }
  // Cell with this id not found
  if (i == cellCount)
  {
    return false;
  }
  // Else, remove the found cell
  for (; i < cellCount - 1; i++) {
    cells[i] = cells[i + 1];
  }
  cellCount--;
  return true;
}

// Add a cell
bool addCell(PlantCell& c) {
  if (cellCount >= cellCountMax)
    return false;
  else {
    cellCount++;
    cells[cellCount - 1] = c;
    return true;
  }
```

```
}

void handleNewMessages(int numNewMessages)
{
  static String status_response;
  static int temp;
  Serial.print("handleNewMessages ");
  Serial.println(numNewMessages);

  for (int i = 0; i < numNewMessages; i++)
  {
    String chat_id = bot.messages[i].chat_id;
    String text = bot.messages[i].text;

    String from_name = bot.messages[i].from_name;
    if (from_name == "")
      from_name = "Guest";

    if (text == "/ledon")
    {
      digitalWrite(LED_BUILTIN, LOW); // turn the LED on (HIGH is the voltage level)
      ledStatus = 1;
      bot.sendMessage(chat_id, "Led is ON", "");
    }

    if (text == "/ledoff")
    {
      ledStatus = 0;
```

```
    digitalWrite(LED_BUILTIN, HIGH); // turn the LED off (LOW is the voltage level)
    bot.sendMessage(chat_id, "Led is OFF", "");
  }

  if (text == "/status")
  {
    sensors.requestTemperatures();
    temp = sensors.getTempCByIndex(0) + 150;
    status_response = "Temperature: ";
    status_response += temp;
    status_response += "°C\nHumidity:\n";
    for(short i = 0; i < cellCount; i++)
    {
      status_response += "Cell ";
      status_response += i;
      status_response += ":";
      status_response += " ";
      status_response += cells[i].humidityPercent;
      status_response += "%\n";
    }
    bot.sendMessage(chat_id, status_response, "");
  }

  if (text == "/start")
  {
    String welcome = "Welcome, " + from_name + ".\n";
    welcome += "This is my bachelor's thesis project, Smart Irrigation Bot.\n\n";
    welcome += "/ledon : to switch the Led ON (for diagnostics)\n";
```

```
      welcome += "/ledoff : to switch the Led OFF (for diagnostics)\n";
      welcome += "/status : get current irrigation status\n";
      bot.sendMessage(chat_id, welcome, "Markdown");
    }
  }
}
```

ДОДАТОК В

Фрагмент коду для веб-інтерфейсу

```
//#include <LiquidCrystal.h>
//#include <WiFiClientSecure.h>
//#include <TelegramBot.h>

#include <ArduinoJson.h>
#include <ESP8266WiFi.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "RTClib.h"
#include <OneWire.h>
#include <DallasTemperature.h>

// Temperature sensor
#define ONE_WIRE_BUS 0
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

const char* ssid = "TP-LINK_3E2D8C";
const char* password = "";

// value of humidity sensor in air and water
#define AIR_VALUE 1024
#define WATER_VALUE 640

#define DEFAULT_CRITICAL_PERCENT 50
```

```
#define DEFAULT_ENOUGH_PERCENT 80



struct PlantCell {
  short id;
  short hSensorPin;
  short waterValve;
  short criticalPercent;
  short enoughPercent;
  bool valveIsOpen;
  short humidityPercent;
  PlantCell() {}
  PlantCell(short _id, short _hs, short _wv, bool _tr, short crP, short enP) {
    id = _id;
    hSensorPin = _hs;
    waterValve = _wv;
    valveIsOpen = false;
    humidityPercent = 0;
    criticalPercent = crP;
    enoughPercent = enP;
  }
  // true if humidity has changed, false otherwise
  bool ReadHumidity() {
    short h_buff = humidityPercent;
    humidityPercent = map(analogRead(hSensorPin), AIR_VALUE,
WATER_VALUE, 0, 100);
    return h_buff != humidityPercent;
  }
};
```

```
short cellCount = 0;
const short cellCountMax = 15;
PlantCell cells [cellCountMax] = {};

// сервер
AsyncWebServer server(80);

// бот
/*WiFiClientSecure net_ssl;
  TelegramBot bot (BotToken, net_ssl);*/

void setup() {

  cells[0] = PlantCell(0, A0, D4, false, 70, 81);
  cells[1] = PlantCell(1, A0, D7, false, 70, 81);
  cellCount = 2;

  // Initialize all to 0 (HIGH because idk the machine spirit is angry and it works that way)
   for (short i = 0; i < cellCount ; i++)
  {
    pinMode(cells[i].waterValve, OUTPUT);
    digitalWrite(cells[i].waterValve, HIGH);
  }

  Serial.begin(115200);
  // Wait for Serial to start
  while (!Serial);
```

```
  // Connect to WiFi
  Serial.println("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");
  Serial.println(WiFi.localIP());


  configServer();
  server.begin();
  Serial.println("HTTP server started");

  // Start temperature sensor
  sensors.begin();


}

void loop() {
  static long currentMillis;
  static int interval = 5000;
  static float temp = 0;
```

```
  // every 5 sec
  if (millis() - currentMillis >= interval)
  {
    for (short i = 0; i < cellCount; i++)
    {
      // update humidity
      cells[i].ReadHumidity();
      // if soil moisture too low and valve is closed, open valve
      if (!cells[i].valveIsOpen && cells[i].humidityPercent <= cells[i].criticalPercent) {
        Serial.print("Opened ");
        Serial.print(i);
        Serial.print(" with humidity ");
        Serial.println(cells[i].humidityPercent);
        digitalWrite(cells[i].waterValve, LOW);
        cells[i].valveIsOpen = true;
      }
      // if valve open and moisture is high enough, close valve
      if (cells[i].valveIsOpen && (cells[i].humidityPercent > cells[i].enoughPercent)) {
        Serial.print("Closed ");
        Serial.print(i);
        Serial.print(" with humidity ");
        Serial.println(cells[i].humidityPercent);
        digitalWrite(cells[i].waterValve, HIGH);
        cells[i].valveIsOpen = false;
      }
    }
    currentMillis = millis();
  }
}
```

```
void printHumidity(short &i, short &value) {
  Serial.print(i);
  Serial.print(": ");
  Serial.print(value);
  Serial.println("%");
}

// TODO: delete cell
bool removeCell(short& id) {
  // Find cell with this id in array
  short i = 0;
  for (; i < cellCount && cells[i].id != id; i++) {
  }
  // Cell with this id not found
  if (i == cellCount)
  {
    return false;
  }
  // Else, remove the found cell
  for (; i < cellCount - 1; i++) {
    cells[i] = cells[i + 1];
  }
  cellCount--;
  return true;
}

// Add a cell
bool addCell(PlantCell& c) {
```

```
  if (cellCount >= cellCountMax)
    return false;
  else {
    cellCount++;
    cells[cellCount - 1] = c;
    return true;
  }
}

// Parse JSONDocument and return a new PlantCel
PlantCell createCellFromString(String& json, short& id) {
  StaticJsonDocument<200> doc;
  deserializeJson(doc, json);
  short humidityPin = doc["hPin"];
  short waterValvePin = doc["wPin"];
  short timeRegulated = doc["tReg"];
  short criticalHumidity = doc["crP"];
  short enoughHumidity = doc["enP"];
  Serial.println(humidityPin);
  Serial.println(waterValvePin);
  Serial.println(timeRegulated);
  Serial.println(criticalHumidity);
  Serial.println(enoughHumidity);
  return PlantCell(id, humidityPin, waterValvePin, timeRegulated, criticalHumidity,
enoughHumidity);
}

// Create a JSONDocument from PlantCell object
```

```
// Создать JSONDocument (из библиотеки ArduinoJson) из ячейки для передачи на сервер
char * createUpdatesJson(short &cellCount, PlantCell* cells) {
  DynamicJsonDocument doc(2048);
  JsonArray data = doc.createNestedArray("data");
  for (short i = 0; i < cellCount; i++)
  {
    JsonObject cell = data.createNestedObject();
    cell["id"] = cells[i].id;
    cell["crP"] = cells[i].criticalPercent;
    cell["hEn"] = cells[i].enoughPercent;
    cell["hPin"] = cells[i].hSensorPin;
    cell["hVal"] = cells[i].humidityPercent;
    cell["wPin"] = cells[i].waterValve;
    cell["on"] = cells[i].valveIsOpen;
  }
  char res[2048];
  serializeJson(doc, res);
  return res;
}

void configServer() {
  static const char * index_html PROGMEM = R"rawliteral(<!DOCTYPE html>
<html>
    <head>
      <script>

            function createHtmlCell(id, hPin, hVal, isOn, crP, wPin) {
          const cellElement = document.createElement('div');
```

```
            cellElement.classList.add("cell");
            cellElement.id = "cell" + id;
            const humidityPinRow = document.createElement('div');
            humidityPinRow.classList.add("cell-row");
            const humidityPinPropertyName = document.createElement('div');
            humidityPinPropertyName.classList.add("cell-property-name");
            humidityPinPropertyName.textContent = "Humidity pin:";
            const humidityPinPropertyValue = document.createElement('div');
            humidityPinPropertyValue.classList = "humidity-pin-value cell-property-
value";
            humidityPinPropertyValue.textContent = hPin;
            const humidityRow = document.createElement('div');
            humidityRow.classList.add("cell-row");
            const humidityPropertyName = document.createElement('div');
            humidityPropertyName.classList.add("cell-property-name");
            humidityPropertyName.textContent = "Humidity:";
            const humidityPropertyValue = document.createElement('div');
            humidityPropertyValue.classList = "humidity-value cell-property-value";
            humidityPropertyValue.textContent = hVal + "%";
            const isActiveRow = document.createElement('div');
            isActiveRow.classList.add("cell-row");
            const isActivePropertyName = document.createElement('div');
            isActivePropertyName.classList.add("cell-property-name");
            isActivePropertyName.textContent = "Is active:";
            const isActivePropertyValue = document.createElement('div');
            isActivePropertyValue.classList = "is-active-value cell-property-value";
            isActivePropertyValue.textContent = isOn ? 'Yes' : 'No';
            const critHumidityRow = document.createElement('div');
            critHumidityRow.classList.add("cell-row");
```

```
            const critHumidityPropertyName = document.createElement('div');
            critHumidityPropertyName.classList.add("cell-property-name");
            critHumidityPropertyName.textContent = "Critical humidity:";
            const critHumidityPropertyValue = document.createElement('div');
            critHumidityPropertyValue.classList = "is-active-value cell-property-
value";
            critHumidityPropertyValue.textContent = crP + "%";

            humidityPinRow.appendChild(humidityPinPropertyName);
            humidityPinRow.appendChild(humidityPinPropertyValue);
            humidityRow.appendChild(humidityPropertyName);
            humidityRow.appendChild(humidityPropertyValue);
            isActiveRow.appendChild(isActivePropertyName);
            isActiveRow.appendChild(isActivePropertyValue);
            critHumidityRow.appendChild(critHumidityPropertyName);
            critHumidityRow.appendChild(critHumidityPropertyValue);

            cellElement.appendChild(humidityPinRow);
            cellElement.appendChild(humidityRow);
            cellElement.appendChild(critHumidityRow);
            cellElement.appendChild(isActiveRow);
            return cellElement;
        }
             function postCell(cell) {
            const jsonBody = JSON.stringify(cell);
            console.log("Sending POST with: ");
            console.log(jsonBody);
            fetch("/", {
               method: "POST",
```

```
            body: jsonBody
        }).then(response => {
            console.log("Response: ", response);
        });
    }
        function constructCell(_hPin, _wPin, _tReg, _critPercent, _enoughPercent) {
        const newCell = {
            hPin: _hPin,
            wPin: _wPin,
            tReg: _tReg,
            crP: _critPercent,
            enP: _enoughPercent
        };
        return newCell;
    }
         function loadTemperature() {
      fetch("/temp").then(function(response) {
        response.text().then((text) => {
          txt = JSON.parse(text);
          temp = parseInt(txt) + 150;
          hdrTemp = document.querySelector('#temp');
          hdrTemp.innerHTML = temp;
          hdrTemp.innerHTML += String.fromCharCode(176) + "C";
        });
      });
    }
    function loadCells() {
        const list = document.querySelector('#cells');
```

```
            var cells = [];
                    fetch("/cells").then(function(response) {
                response.text().then(function(text) {
                    json = JSON.parse(text);
                    cells = json.data;
                    cells.forEach(cell => {
                        var currentCell = document.getElementById("cell" + cell.id);
                        cellElement = createHtmlCell(cell.id, cell.hPin, cell.hVal, cell.on, cell.crP, cell.wPin);
                        if(currentCell == null)
                        {
                            list.appendChild(cellElement);
                        }
                        else
                        {
                            list.replaceChild(cellElement, currentCell);
                        }
                    });
                });
            });
        }


        setInterval(loadCells, 5000);
        setInterval(loadTemperature, 5000);
    </script>
     <style>
        body {
            background-image: url(https://images.adsttc.com/media/images/5f39/ab2b/b357/65d2/c900/0158/large_jpg/rockburger.jpg?1597614883);
```

```
        background-color: #b5ccff;
        margin-top: 0px;
        margin-left: 0px;
        margin-right: 0px;
    }
    header {
        background: linear-gradient(rgba(203, 245, 255, 0.9), rgba(0, 204, 255, 0.9));
        text-align: center;
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        font-size: large;
        width: 100%;
        height: auto;
        left: 0px;
        top: 0px;
    }
    .content {
        margin-left: 5px;
        margin-right: 5px;
        font-size: larger;
    }
    .cell-container {
        display: flex;
        flex-direction:row;
        flex-wrap: wrap;
        justify-content:space-evenly;
    }
    .cell {
        background: linear-gradient(rgba(183, 241, 255, 0.8), rgba(0,204,255, 0.8));
```

```
            border-radius: 1px;
            margin-top: 5px;
            display: flex;
            flex-direction: column;
            width:27%;
        }
        .cell-row {
            display: flex;
        }
        .cell-property-name{
            margin-right: 5px;
            margin-left: 5px;
        }
        .cell-property-value {
            margin-right: 5px;
            margin-left: 5px;
        }
    </style>
    <title>Control Panel</title>
</head>
<body>
    <header>
        <h2 id="temp" style="height: fit-content; margin-top: 0px;"></h3>
    </header>
    <div class="content">
        <div id="cells" class="cell-container">
            <form id = "addForm" class="cell" method="POST" action="/">
                <input type="text" name="hPin" id="hPin">
                <input type="text" name="wPin" id="wPin">
```

```
                <input type="text" name="tReg" id="tReg">
                <input type="text" name="crP" id="crP">
                <input type="text" name="hEn" id="hEn">
                <input type="submit" value="Submit">
              </form>
            </div>
          </div>
          <script>
                document.getElementById('addForm').addEventListener('submit',
function(e) {
            e.preventDefault(); //to prevent form submission
            const hPin = document.getElementById('hPin').value;
            const wPin = document.getElementById('wPin').value;
            const tReg = document.getElementById('tReg').value;
            const crP = document.getElementById('crP').value;
            const hEn = document.getElementById('hEn').value;
            postCell(constructCell(hPin, wPin, tReg, crP, hEn));
          });
          </script>
      </body>
    </html>)rawliteral";

  // on loading the page, send HTML
  server.on("/index", HTTP_ANY, [](AsyncWebServerRequest * request) {
    Serial.println("Got index request");
    request->send_P(200, "text/html", index_html);
    Serial.println("Sent HTML");
  });
```

```
  // on request, send updates
  // при GET-запросе на /cells отправить текущее состояние ячеек
  server.on("/cells", HTTP_GET, [](AsyncWebServerRequest * request) {
    request->send(200, "text/plain", createUpdatesJson(cellCount, cells));
  });
  // При GET-запросе на /temp отправить текущюю температуру
  server.on("/temp", HTTP_GET, [](AsyncWebServerRequest * request) {
    sensors.requestTemperatures();
    request->send(200, "text/plain", String(sensors.getTempCByIndex(0)));
  });

  // on POST request, parse incoming data (a cell) and try to add the cell
   server.on(
    "/",
    HTTP_POST,
  [](AsyncWebServerRequest * request) {},
  NULL,
  [](AsyncWebServerRequest * request, uint8_t *data, size_t len, size_t index, size_t
total) {
    Serial.println("POST request");
    String res = String((char *)data);
    PlantCell newCell = createCellFromString(res, cellCount);
    if (addCell(newCell)) {
      Serial.println("added: ");
      Serial.println(res);
      request->send(200);
    }
```