Міністерство освіти і науки України

Державний університет «Одеська політехніка»

Інститут штучного інтелекту та робототехніки

Кафедра «Комп'ютерні системи»

Закордонець Олександр Геннадійович,

студент групи УК-161

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

ДОСЛІДЖЕННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ ПОЛИВУ НА СМАРТ ДАТЧИКАХ

Спеціальність: 123 – "Комп'ютерна інженерія"

Спеціалізація: Спеціалізовані комп'ютерні системи

Керівник:

Ступень Павло В'ячеславович,

кандидат техн. наук, доцент

Одеса — 2021

Міністерство освіти і науки України

Державний університет «Одеська політехніка»

Інститут штучного інтелекту та робототехніки
Кафедра комп'ютерних систем

Рівень вищої освіти другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
(шифр і назва)

Спеціалізація / освітня програма Спеціалізовані ком'ютерні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

“____” _____ 2021 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Закордонцю Олександру Геннадійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Дослідження комп’ютерної системи поливу на смарт датчиках

Керівник роботи Ступень Павло В’ячеславович, кандидат техн. наук, доцент ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ректора ОНПУ від “ 01 ” жовтня 2021 року № 346-в
2. Зміст роботи
Вступ. Роль сучасних технологій у розвитку сільського господарства. Структурний аналіз і вибір компонентів системи. Реалізація інтелектуальної системи поливу.

3. Перелік ілюстративного матеріалу
Презентація – 11 слайдів

4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

5. Дата видачі завдання 01.10.21

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Затвердження теми кваліфікаційної роботи	01.10.21	Виконано
2	Аналіз предметної області	02.10.21 – 07.10.21	Виконано
3	Аналіз сучасниз технологій у розвитку сільського господарства	08.10.21 – 12.10.21	Виконано
4	Структурний аналіз	13.10.21 – 17.10.21	Виконано
5	Вибір компонентів системи	18.10.21 – 28.10.21	Виконано
6	Реалізація інтелектуальної системи поливу	29.10.21 – 10.11.21	Виконано
7	Загальна архітектура	11.11.21 – 16.11.21	Виконано
8	Функціональні можливості системи та висновки	17.11.21 – 21.11.21	Виконано
9	Оформлення пояснювальної записки	22.11.21 – 30.11.21	Виконано
10	Оформлення презентації	01.12.21 – 11.12.21	Виконано
11	Захист кваліфікаційної роботи	29.12.21	Виконано

Здобувач вищої освіти _____ Закордонець О. Г.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ступень П.В.
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Актуальність теми дослідження

Сучасні системи поливу зазвичай розподілять воду рівномірним способом по всьому полю, який одночасно не є ефективним і може нашкодити рослинам. Інтернет речей (IoT) - нова технологія, що передбачає використання датчиків і приводів для побудови складних систем зі зворотним зв’язком, дає можливість створити розумне рішення для зрошення.

Поточна робота ілюструє дослідження польових компонентів моніторингу систем поливу, що використовує готові та недорогі компоненти, в той же час досліджуючи, наскільки це складно використовувати апаратні компоненти та відповідні інструменти в області IoT.

Проблематика спочатку була структурована через класичний процес зверху вниз, щоб ідентифікувати такі компоненти, як мікрокомп'ютери, мікроконтролери, датчики та мережеві з’єднання, які знадобляться для створення рішення. Було прийнято: Raspberry Pi, Arduino Uno, гігрометр серії MH-Sensor, протокол обміну повідомленнями MQTT і протокол зв'язку ZigBee, як реалізовано в XBee S2C. Після того, як компоненти були визначені, дослідження проводилося за принципом - знизу вгору. Дослідження компонентів проводилося ізольовано відносно один одного, через структурований ряд робіт, при цьому кожен пункт стосується певного компонента та перевіряє, наскільки складним було б використання певного компонента. При цьому кожен етап дозволяв автору набути і поглибити розуміння кожного компонента, і поступово створювати більш досконалу систему, до прийняття остаточних рішень. Було прийнято переважну більшість ідентифікованих компонентів і інструментів, які повинні бути простими у використанні, добре задокументованими і, головне, функціональними для використання цільовим користувачем. Автор дійшов висновку, що реалізація MQTT-SN (MQTT-Sensor Network), є не такою досконалою, як інші. Це призвело до того, що постала необхідність у розробці концепції легкого, так званого, шлюзу - ZigBee/MQTT, що є одним із ключових елементів інтелектуальної системи поливу.

Роль зрошення в сільському господарстві

Вода, що подається населеним пунктам, потім використовується для різних цілей, у тому числі для використання у побуті, використання в містах, гірничодобувній, та інших видах промисловостей. У даній роботі основна увага зосереджена на сільському господарстві.

Сільськогосподарська промисловість значним чином залежить від ефективного зрошення рослин і сільськогосподарських культур. Вода відіграє ключову роль у життєвому циклі рослин. Рослинам потрібна вода для росту і високої врожайності, тобто вода є необхідною для транспортування поживних речовин, потрібних рослині для росту, з ґрунту до листя рослини. Крім того, вода відіграє значну роль під час фотосинтезу. Отже, важливо переконатися, що рослини мають потрібну кількість води, не занадто багато або надто мало, адже надмірний полив пригнічує рослини, і може значно нашкодити. З іншого боку, якщо рослинам не вистачає води, вони не отримають поживні речовини з ґрунту, що також потенційно приносить шкоду.

Очевидно, що природним джерелом води для рослин є дощова вода, однак у засушливі періоди, коли мало атмосферних опадів, рослини потрібно поливати штучно, за допомогою систем поливу.

Інтернет речей (IoT)

Інтернет речей – це мережа пристроїв, які мають можливість ідентифікувати та активуватися шляхом підключення до мережі Інтернет. Вони здатні спілкуватися та обмінюватися інформацією через уніфіковану структуру. Хоча концепцію IoT вперше ввів Кевін Ештон в рамках управління ланцюгом поставок, протягом багатьох років IoT застосовувався в багатьох областях. Наприклад, IoT зіграв значну роль у впровадженні ідеї розумних будинків і розумних міст. Продукти IoT також були створені в галузі охорони здоров’я, освіти та сільського господарства.

IoT педставляє наступну еволюцію Інтернету. Спочатку основна функціональність Інтернету полягала у підключенні комп’ютерів. Ця функція перемістилася в бік об’єднання людей. І зараз, IoT – це передусім об’єднання речей. В основі IoT лежить фізичне обладнання з можливостями керування датчиками, які передають зібрані дані в більшу мережу - Інтернет. Ці пристрої приєднуються до

фізичних об'єктів, що дозволяє трансформувати ці фізичні об’єкти в «розумні», які «усвідомлюють» своє оточення: вони можуть бачити, відчувати, спілкуватися, взаємодіяти та обмінюватися даними, знаннями та інформацією. Технології IoT може ідентифікувати, знаходити, контролювати ці об’єкти та автоматично запускати відповідні події в режимі реального часу. Датчики також часто підключаються до програмованих пристроїв, таких як мікро-контролери, які здатні виконувати збір даних з чутливих елементів, і керувати поведінкою виконавчих елементів. Робота з цими апаратними елементами вимагає розуміння та знання того, як керувати датчиками та виконавчими механізмами. Тому побудова IoT системи вимагає розуміння як інформатики, так і електронної техніки.

Мета дослідження

Основна мета роботи – дослідити основні аспекти проблематики створення інтелектуальної системи зрошення, яка контролює точну кількість води, що подається на частину поля, виходячи з вмісту води, яка вже наяна в ґрунті. Для досягнення поставленої мети необхідно реалізувати наступні підцілі:

- обрати готові та недорогі компоненти, які можна використовувати для виконання дослідження за архітектурою «зверху вниз».

- застосувати підхід «знизу вгору» для навчання, а потім інтегрувати кожен компонент до фінального рішення, фіксуючи рівень складності на кожному кроці.

Задачі дослідження

- проаналізувати публікації за темою роботи;

- провести системний аналіз і вибір компонентів;

- вивчити взаємодію Arduino та XBee;

- розглянути аспекти взаємодії бездротового зв’язоку між Raspberry Pi та Arduino;

- дослідити протоколи MQTT і MQTT-SN;

- сформулювати архітектуру засобу інформаційної комунікації;

- скласти висновки за результатами роботи.

Об'єкт дослідження – процес поливу сільськогосподарських земель.

Предмет дослідження – багатокомпонентна інтелектуальна система зрошення земель сільськогосподарського призначення.

Методологія дослідження

Побудова системи відбувалася за конструктивною методологією в тому сенсі, що проводився пошук комплексного рішення, для реалізації в реальних умовах. Перевірялась складність проведення кожного етапу. Проблема спочатку була розбита за допомогою класичного підходу «зверху вниз», для того щоб визначити готові та недорогі компоненти, такі як мікроконтролери, датчики та мережеві підключення, які знадобляться для створення системи. Коли компоненти були прийняті, робота проводилась за підходом «знизу вгору». Компоненти системи досліджувались ізольовано один від одного. Це було зроблено за допомогою структурованої серії досліджень, кожне з яких стосується певного компонента та ідентифікує складність і доцільність його застосування. Цей процес проходив до набуття розроблюваною системо остаточного вигляду.

Наукова новизна дослідження зосереджується на підході до побудови комплексної структури інтелектуальної системи поливу. У даній роботі досліджується взаємодія не тільки вузлів вимірювання рівня вологості грунту та прогнозування атмосферних опадів, а і додатково – принцип визначення пріоритету у зрошенні кожної окремої ділянки на полі. Такий підхід дає можливість не тільки знизити витрату води та електроенергії, під час поливу, а й підвищити врожайність сільськогосподарських культур на відповідних ділянках.

Практичне значення одержаних результатів

Через нерегулярність або дефіцит атмосферних опадів у багатьох районах, часто потрібне штучне зрошення. Методи загального поливу, як правило, наносять воду на поле рівномірно. Однак різні частини поля мають різні характеристики, такі як склад ґрунту або вплив сонячного випромінювання. Все більш досконалі зрошувальні системи були розроблені комерційними організаціями, щоб пом'якшити проблеми аж до побудови найновіших розумних зрошувальних систем. Про такі

дослідження повідомляється в цій роботі. Зокрема, досліджуються два основних, взаємопов’язаних питання:

- які є наявні готові, переважно недорогі компоненти, для створення інтелектуальної системи зрошення?

- які основні аспекти проблематики розробки подібних рішень?

Третє пов’язане питання стосується «досконалості» різних компонентів, де ступінь придатності кожного компонента залежить від кількості супроводжувальної документації та супутніх програмних рішень.

Особистий внесок магістра. Автор даної роботи досліджує саме раціональність побудови та застосування багатокомпонентної інтелектуальної системи поливу, що включає в себе окремі раніше вже відомі елементи, а саме: комплекси вимірювання вологості грунту, системи поливу, що враховують прогнозування атмосферних опадів та принципи зрошування, з урахуванням пріоритетності в поливі окремих ділянок.

РОЗДІЛ 1. РОЛЬ СУЧАСНИХ ТЕХНОЛОГІЙ У РОЗВИТКУ СІЛЬСЬКОГО ГОСПОДАРТВА

1.1 Четверта промислова революція (4IR) та Інтернет речей (IoT)

Як визначив Клаус Шваб, засновник і виконавчий голова World Economic Форум, четверта промислова революція характеризується технологічним прогресом і інноваціями, які руйнують бар'єри між фізичним, цифровим та біологічним світами [1].

Революція використовує нові технології, такі як штучний інтелект (AI), робототехніка, хмара комп’ютери, інтернет речей (IoT) та 3D-друк серед інших [1; 2; 3; 4].

1.1.1 Технологічні драйвери 4IR

Інтернет речей (IoT)

Цифрові технології є основною рушійною силою 4IR. В основі цих технологій лежить Інтернет речей (IoT) [3]. IoT визначається як мережа об’єктів, пристроїв із зондуванням і привідною здатністю. Ці пристрої під’єднані до Інтернету та можуть спілкуватися, передавати та обмінюватися інформацією через уніфіковану структуру [5; 6; 7]. Концепцію IoT вперше запровадив Кевін Ештон в рамках управління ланцюгом поставок.

Протягом багатьох років IoT застосовувався в кількох доменах [7]. Наприклад, IoT зіграли значну роль у впровадженні ідеї розумних будинків та розумних міст. Продукти IoT також спостерігаються у сфері охорони здоров’я, освіти та сільського господарства [7]. IoT представляє наступну еволюцію до Інтернету. Спочатку основною функцією Інтернету було підключення комп'ютерів. Ця функція перемістилася в бік об’єднання людей. І зараз, перш за все, IoT про з’єднання «речей».

Основою IoT є фізичні апаратні елементи з датчиками та активаційні можливості, які передають зібрані дані в більшу мережу - Інтернет. Ці пристрої є прикріпленими до фізичних об'єктів і дозволяють трансформувати ці традиційні фізичні об'єкти в «розумні» об’єкти, які «усвідомлюють» своє оточення. Вони можуть бачити, відчувати, спілкуватися, взаємодіяти та обмінюватися даними, знаннями та інформацією [5]. Технології IoT можуть ідентифікувати, знаходити, контролювати ці об’єкти та запускати відповідні події автономно в режимі реального часу [3].

Датчики також часто підключаються до програмованих пристроїв, таких як мікроконтролери, які здатні виконувати збір даних з чутливих елементів і керувати поведінкою виконавчих елементів. Тому робота з цими апаратними елементами вимагає розуміння та знання того, як керувати датчиками та виконавчими механізмами. Крім того, елементи є зібраними в електронну схему і керовані програмним забезпеченням мікроконтролера.

Штучний інтелект

Штучний інтелект (ШІ) має на меті імітувати та перевершувати людський інтелект за допомогою обчислювальної техніки. Він запрограмований думати як люди, а також імітувати їхні дії [3]. ШІ складається з численних підгалузей, включаючи машинне навчання (МН), яке на даний момент є основною галуззю. [3].

Машинне навчання використовує алгоритми для автоматичного навчання, пошуку прихованих ідей та покращення наявних даних без програмування на це [3]. Машинне навчання також дозволяє це робити системам адаптації та прийняття повторюваних і надійних рішень, при впливі на нові дані [3]. Машинне навчання широко використовується для розпізнавання зображень, текстового аналізу, біометричної ідентифікація тощо [9].

1.1.2 4IP-технології в сільськогосподарському секторі

4IR надає потенційно багато можливостей для створення нових форм виробництва, які викликають період зростання кількості ресурсів [10]. Використання датчиків, великих даних, машинне навчання може трансформувати продуктивність

сільського господарства скрізь і зокрема в Україні. Ефект 4IR в сільському господарстві реалізується через оптимізацію прийняття рішень [2]. Прийняття сільськогосподарських рішень передбачає використання сільськогосподарських пристроїв та обладнання для збору даних про стан ґрунту, а також інформацію про зростання, клімат та навколишнє середовище [11; 2; 10]. При цьому, не тільки покращується точність сільськогосподарські виробництва, але й досягаються його необхідні показники [13]. Інформація про конкурентоспроможні ціни, інформація про моніторинг посівів, профілактика захворювань, поради та підтримка, пом'якшення наслідків стихійного лиха також створює виняткову можливість реконструювати сільськогосподарський сектор для підвищення доходів, виробництва та попиту в посушливих регіонах [2; 12].

1.1.3 Вимоги користувача до програм Інтернету речей

Автори [14] провели обсерваційне дослідження, метою якого було виявлення проблем, які виникли у користувачів, коли вони взаємодіють із додатками IoT. Мета дослідження була розбита на два основних питання дослідження. Перше питання було зосереджено на тому, як користувачі могли вільно працювати, взаємодіяти з фізичними атрибутами Інтернету речей і не звертатися за допомогою до технічних команд підтримки. Другим питанням було визначення проблем, з якими стикалися користувачі під час моніторингу додатку IoT, коли він проходить самоадаптацію.

Дослідження визначило та заявило можливі вдосконалення послуг і додатків IoT через відгуки користувачів. У навчанні використовували вибірку з п'яти учасників з різними професійними та технічними профілями. Перший користувач був інженером-програмістом, який не мав жодних знань про IoT, другий був досвідченим програмістом, який має основне уявлення про те, що таке IoT, але ніколи не використовував його для себе. Третім і четвертим учасниками були власник ресторану і офіціант, відповідно мав розуміння того, що таке IoT, а останній не мав жодних знань навіть про технології в загальному. П'ятим і останнім учасником був розробник програмного забезпечення зі знаннями про IoT та його застосування.

Учасникам було надано стартовий набір «Розумний дім», що складається з трьох пристроїв: один, з яких виявив присутність або рух людини, детектор відкриття/закриття, який сигналізує, коли двері були відкриті чи зачинені, а також присутність, яка вказує на місцезнаходження GPS об'єкта, коли він був прикріплений до учасників. Потім учасники повинні були встановити програму, розташувавши датчики в правильному положенні об’єкта або людини. Загальні результати дослідження показали, що здатність користувачів вивчати, як використовувати IoT, залежить виключно від їхніх особистісних якостей. Користувачі, які не мають досвіду роботи з IT, розчарував цей досвід і переважною, окрім зручності для навчання додатків, користувачам було важко приступили до налаштування елементів програми, вони не розуміли з чого потрібно починати, і вони покладалися на методи «випробовування» та «проби та помилки».

Введення предметів IoT до існуючої навчальної програми, Ізраїльського Технологічного інституту Холона запровадив курс IoT за вибором у їхній існуючій програмі бакалавра з управління технологіями [15]. Курс структурований так, щоб складатися з обох теоретичних тем, які викладаються на офіційних лекціях і презентаціях, а також на практичних заняттях.

Теоретичні теми охоплюють історію IoT від M2M до систем IoT. Вони також покривають основні концепції, визначення, архітектура, пристрої та додатки в промисловості, в тому числі сервісно-орієнтована архітектура, що складається з чотирьох рівнів (відправлення, мережа, служба та користувач), що використовується для проектування багатьох систем IoT. На курсі практичної роботи студенти розробляють план проекту IoT і продукт на основі архітектури IoT з використанням датчиків, хмарного обчислення, модулів та програмних послуг. Проект реалізований на комп’ютері Raspberry Pi, що використовують мову програмування Python. Під час практичних занять студенти навчаються щоб встановити операційну систему Raspbian Linux на Raspberry Pi. Вони також дізнаються про різні контакти GPIO на Raspberry Pi, що підключають різні типи датчиків (наприклад, температуру, світло) до Raspberry Pi та взаємодію з датчиками (наприклад, виконання вимірювання) за

допомогою Python. Студенти також навчаються встановленню локальної бази даних (SQLite) та використанню хмарних баз даних, таких як Amazon Relational Database Service.

До моменту написання роботи курс уже викладався двічі, загалом 32 студентам. Більшість із цих студентів не мали попереднього досвіду роботи з електронікою, і так вони вперше працювали з інтеграцією обладнання та апаратного забезпечення. Однак під час курсу, було виявлено, що Raspberry Pi був «дружнім» одноплатним комп'ютером, за допомогою якого користувачі могли легко створювати продукти IoT. Крім того, на сайті Raspberry Pi міститься детальна інформація, та інструкції щодо роботи з мікрокомп'ютером.

У першому випадку на курсі програмування викладалося на Java, що студентам було досить складним. В результаті вони витратили більшу частину часу на розробку синтаксису мови, замість фактичного проекту. У другому випадку мова програмування курсу була змінена на Python. Студентам було легше працювати з Python, вони змогли одночасно вивчайти нову мову та зосередитись на розробці програми. Було зроблено висновок, що мова Python більше підходить для людей, які не мають досвіду в розробці програмного забезпечення.

Майбутніми цілями установи було створення передової лабораторії IoT для прискорення передових проектів як для дослідницьких, так і для освітніх цілей [15]. Як результат, лабораторія буде оснащена не тільки тим обладнанням, яке вже є (датчики, двигуни, підсвічування, кнопки тощо), а також передовими компонентами для створення роботів, які будуть реалізовані з додатками IoT. Також заклад планує оновити існуючий курс «Введення в програмування» у програмі бакалавра для навчання Python, а також додати новий курс «Поглиблені теми програмування». Таким чином вони сподіваються забезпечити студентів кращим початковим курсом IoT [15].

Навчальна програма бакалавра для вивчення IoT на факультеті комп'ютерних наук факультету комп'ютерних наук національного університету Сан-Маркос

пропонує навчальний план бакалавра для вивчення тем IoT, щоб підготувати випускників високої кваліфікації, які можуть розробляти додатки IoT, що відповідають потребам національної промисловості [16]. Пропонований курс передбачав впровадження нових курсів, які охоплюють основні теми, що стосуються IoT. Організація та курс з інженерного управління містять теми про сенсорні мережі.

У навчальний план також входить курс із розподілених систем, який складається з тем про рівень управління проміжним програмним забезпеченням. Навчальний план завершується програмною інженерією та курсом «Інтелектуальні системи», де студенти дізнаються про розробку додатків IoT та подібні продукти [16]. Під час узгодження з новим навчальним планом університет впровадив дослідження IoT в лабораторії. В результаті у вітчизняній науці взяли участь студенти магістратури, де вони представили два IoT-додатки, одне направлений на розробку середовища за допомогою Arduino та управління хмарними даними [16]. У другому проекті йшлося про інтерфейс мозку/комп'ютера, який керує автомобілем для людей з обмеженими фізичними можливостями. Крім того, їм вдалося встановити міцніші відносини між дослідницькою лабораторією IoT і різними галузями, де компанії фінансують деякі з проектів, і наймають кращих студентів магістратури.

Навчальна програма IoT підготувала кваліфікованих випускників із покращеною продуктивністю для розробка нових додатків, що включають Інтернет речей.

1.2 Інтелектуальні системи зрошування

У [17] автори розробили автоматизовану інтелектуальну систему поливу, яка керує потоком води на основі умов навколишнього середовища (температура і вологість) на ділянці за допомогою датчиків. Система є оснащеною платою Arduino, інтегрованою з датчиками вологості ґрунту та температури, які вимірюють вологість ґрунту та температуру на ділянці. Мікроконтролер вмикає/вимикає потік води на ділянці на основі цих показань. Крім того, Arduino підключається до модуля GSM, що дозволяє підключати систему до смарт-системи, телефону. Потім система

підключається до спеціального додатка для Android, за допомогою якого фермер може бачити роботу системи, включаючи значення датчика температури та вологості. Додаток дозволяє фермерам вручну керувати зрошенням, тобто перемикання ON/OFF стану водяного клапану на основі відображених значень датчика. Роботою системи зрошення можна дистанційно керувати з розробленої веб-сторінки, доступної з www.sulamadenetim.com, також використовуючи значення датчика.

Використання датчика вологості ґрунту запобігає надмірному зрошенню. Це не тільки економить воду, а й підвищує врожайність. Можливість керувати системою за допомогою смартфона робить його зручним для використання, оскільки сьогодні смартфони мають високий рівень застосування. Зручність використання збільшується завдяки можливості дистанційного керування зрошенням на веб-сайті.

1.2.1 Реалізація автономної системи зрошення на IoT Raspberry Pi

Автор запропонував конструкцію автоматичної системи водопостачання, здатної виявляти відповідний час для поливу рослин на сільськогосподарських угіддях і постійно контролювати рівень води, запобігати накопиченню води навколо гнилих саджанців [18]. Система складається з центральної плати Raspberry Pi, мікроконтролера Arduino, різних датчиків, модуля WiFi, GSM модуля, релейної колодки.

Мікроконтролери Arduino розміщені в різних місцях на сільськогосподарських угідь і збирають дані з датчиків. Система використовує датчик вологості ґрунту, датчик денного світла (або фотоелемент) і датчик рівня води. Датчик вологості ґрунту взаємодіє з ґрунтом для вимірювання сухості/вологості ґрунту, а датчик денного світла виявляє сонячні промені. Крім того, датчик рівня води використовувався для вимірювання рівня води в полі. Усі зібрані дані з датчиків передаються до мікроконтролера Arduino. Після отримання даних датчика мікроконтролер бездротовим шляхом передає дані на центральну плату Raspberry Pi через WiFi. Raspberry Pi продовжує роботу, завантажуючи кожен Arduino за IP-адресою,

зазначеною на Raspberry Pi, і запитує дані датчика від кожного Arduino. Коли Raspberry Pi отримує дані датчика від Arduino, він порівнює їх для попереднього встановлення порогових значень вологості ґрунту, освітленості та рівня води. Тоді Raspberry Pi надсилає команди Arduino, щоб увімкнути/вимкнути подачу води на певний період часу, і вказує Arduino вимкнути подачу води, коли тривалість перевищена.

Коли Arduino отримує ці команди, він просто виконує відповідні дії, відкриває або закриває водяний клапан на конкретній ділянці. Система також складається з датчика рівня води у резервуарі, який визначає кількість води в баку, і повідомляє про це адміністратора, коли рівень води нижче встановленого значення. Arduino використовує GSM модуль, оснащений мобільною SIM-картою для зв'язку з адміністратором за допомогою SMS-сервісу.

Запропонована система є масштабованою, її можна розміщувати в невеликих горщиках, у саду на задньому дворі, і на більшій сільськогосподарській ділянці. Це також забезпечує науковий та системний підхід, рослини отримують потрібну кількість води, тим самим покращуючи врожайність, а також забезпечуючи більш розумне використання води.

Система може включати прогноз погоди, що може допомогти покращити прийняття рішень і надалі уникнути надмірних втрат води. Систему також можна повторно розробити для використання інструменту, який вимірює поживні речовини в ґрунті. Такий інструмент міг би дозволить системі точно подавати добрива в землю.

1.2.2 Інтелектуальна зрошувальна система, як механізм управління водними ресурсами

Автори [19] також розробили систему управління водними ресурсами для оптимізації доступності води у водоймах у певних районах, що стикаються з проблемами дефіциту води, щоб забезпечити ефективне її використання. Система

керує зрошенням на основі вологості ґрунту та кількості води, наявної у водосховищі. Система складається з датчика вологості ґрунту та ультразвукового датчика. Датчик вологості грунту вимірює кількість води в ґрунті. Встановлюють ультразвуковий датчик на водоймі, а потім вимірюють рівень води. Значення з датчиків перетворюються на електронний сигнал, який потім надсилається на мікроконтролер Atmega 328 в Arduino Uno. Отримавши значення, мікроконтролер вмикає/вимикає водяні насоси, коли показники перевищують задані порогові значення. Крім того, система використовує значення рівня води в резервуарі, щоб визначити пріоритети ділянок для зрошення та визначити кількість насосів, на які потрібно ввімкнути у будь-який момент часу.

Система забезпечує зрошення рослин відповідно до потреб ґрунту у воді в свою чергу забезпечує ріст рослин і запобігає втратам води. Можливість розставляти пріоритети роботи насосів на основі рівня води у водоймах гарантують довговічність поливного обладнання, а також запобігає витраті води. Система була перевірена в лабораторії з трьома різними зразками ґрунту: сухі, вологі та перезволожені. Результати експериментів показали хороші результати продуктивності та гнучкості роботи системи [19].

1.3 Система моніторингу з використанням мережі речей у точному землеробстві

У [20] автор запропонував систему оповіщення, яка контролює водний стан у рослин за допомогою технології IoT. Система була розроблена як трирівнева програма, що складається з рівня збору даних, рівня комунікації і рівня IoT.

Рівень збору даних складається з сенсорної мережі, розгорнутої в полях. Кожен сенсорний вузол у мережі складається з мікроконтролера Atmega 128, IEEE 802.15.4 трансивера ZigBee, карти SDRAM і датчика вологості ґрунту. Сенсорний вузол вставлено у ґрунті для вимірювання об’єму води. Сенсорні вузли спілкуються за допомогою 802.15.4 - протоколу, який безпосередньо не підтримується в Інтернеті.

Тоді комунікатор виступає як міст між мережею 802.15.4 і мережею GPRS, щоб дозволити з'єднання між ними сенсорної мережі та Інтернет. Багатопротокольний маршрутизатор Meshlium підтримує 5 інтерфейсів бездротового підключення, включаючи ZigBee і 3G/GPRS [21]. Маршрутизатор також вважається дуже зручним для користувачів веб-додатків. Він розміщений у водонепроникному корпусі-футлярі, що дозволяє встановлювати його на відкритому повітрі, збирає дані вузла датчика, а потім зберігає його в базі даних локально або зовнішньо.

Останній рівень програми — «хмара», що дозволяє користувачам взаємодіяти з системою. Для вивчення була обрана платформа Ubidots, датчик із маршрутизатором Meshlium. Дані розміщуються на хмарній платформі. Ubidots - це платформа, яка дозволяє розробникам отримувати дані датчиків і обробляти їх у зручному вигляді. Платформа також використовується для передачі даних датчиків у хмару IoT із внутрішнього пристрою. Крім того, Ubidots дозволяє розробникам автоматично визначати сповіщення та тригери відповіді від заданого порогово значення. Огляд в інтерфейсі користувача складається з Google - карт, на яких відображається поле ферми з синіми точками, які вказують розташування кожного сенсорного вузла в польових умовах. Додаток також вказує значення вологості ґрунту, пов’язане з кожним датчиком, та хронологію зміни вологості кожної ділянки. Крім того, система надсилає SMS сповіщення, коли рівень води перевищує критичне порогове значення.

Відображення стану вологості ґрунту на кожній ділянці дозволяє фермерам оцінити кількість води, необхідної на земельній ділянці, що може підвищити ефективність використання води, і також підвищити врожайність сільськогосподарських культур. Крім того, хронологічна інформація про зміни вологості ґрунту на кожній ділянці, насиченість або посуху кожної ділянки може допомагати вирішувати, яку культуру посадити на кожній ділянці, виходячи з вологості ділянки та вимог до особливостей зрошування різних рослин.

Повідомлення, коли рівень води досягає критичної точки запобігає водному стресу у рослин. Однак систему можна вдосконалити до більш складного рівня, наприклад, автоматизувати полив, щоб система могла перемикатися

увімкнення/вимкнення водопостачання на ділянці на основі показань датчика без участі людського втручання. Система також може включати прогноз погоди, щоб покращити прийняття рішень.

1.3.1 Недорога система інтелектуального контролю зрошення

У [22] автор запропонував прототип недорогої розумної системи зрошення, яку фермер середнього класу може використовувати на своїй фермі. Система складається з сенсорних вузлів, розміщених у різних точках на полі ферми та вузла контролера. Кожен вузол датчика оснащений загальним показником вологості ґрунту - датчик і платою Arduino Uno, яка складається з мікросхеми ATMEGA-328 - контролер. Датчик вологості ґрунту взаємодіє з ґрунтом і вимірює значення вологості, яке потім передається на мікроконтролер через пристрій бездротової мережі. Отримавши це значення вологості, мікроконтролер розраховує відсоток сухості ґрунту. Далі, це значення надсилається на вузол контролера, де він далі аналізується. Вузол контролера оснащений мікрокомп'ютером Raspberry Pi, який керує клапанами водяного двигуна в різних точках ферми на основі відсотку сухості, отриманого від вузлів датчика. Якщо сухість або вологість перевищує необхідні значення, то клапан відповідно відкритий або закритий. Крім того, Raspberry Pi підключається до Інтернету та передає такі дані, як стан водяного насосу (вкл./вимк.) на зареєстрований номер мобільного телефону.

Експериментальні результати прототипу показали, що він здатний автоматично контролювати поливні заходи на основі показань вологості ґрунту. Зрошення поля на основі його потреби у воді зменшують витрати води. Крім того, автоматичний полив зменшує необхідність втручання людини. Автори відзначають, що системні пристрої споживають мало електроенергії і тому можуть працювати від акумулятора протягом тривалого часу. Прототип проходив випробування у віддаленому районі, і було показано, що система є вигідною для фермерів у відповідних районах. Крім того,

систему можна використовувати з усіма видами поливу, напр. крапельний полив, канал, зрошувач.

1.3.2 Ефективне впровадження недорогих інтелектуальних систем зрошення

Автори [15] розробили автоматизоване та недороге рішення для розумного зрошення разом із можливістю моніторингу погодних параметрів та забезпечення безпеки поля ферми. Система використовує NodeMCU як мікроконтролер, інтегрований з датчиками PH, вологості ґрунту та PIR, а також модулем WiFi, який також може діяти як маршрутизатор, через який система передає дані в хмару.

NodeMCU збирає таку інформацію про погоду, як температура, вологість і хмарність навколо місця розташування ділянки. Ця інформація про погоду зберігається в хмарі. Датчик PH вимірює значення ph ґрунту, що вказує на кислотність, а отже, підказує фермеру тип культури, яку доцільніше висаджувати на даній ділянці у певний період часу. Датчик вологості ґрунту контролює вологість ґрунту. Система використовує значення датчика для вмикання/вимкнення подачі води в полі, коли показник нижче, або перевищує порогове значення відповідно. Дані про вологість ґрунту також передаються до хмари, де вона відстежується за допомогою веб-програми та може бути отримана у форматі електронної таблиці для подальшого аналізу.

Датчик PIR (пасивний інфрачервоний) постійно контролює поле, і визначає порушників (тварин чи людей). При виявленні порушника фермеру повідомляється через електронну пошту.

Було доведено, що система має дуже низьку вартість і споживає менше енергії, оскільки використовує NodeMCU як мікроконтролер. Крім того, система ефективна у сфері зрошення: можливість стежити за полем, без необхідності фізичного обходу території є дуже корисною для користувача. Здатність передбачити врожай для окремого поля підвищує врожайність і продуктивність. PIR - датчик не має обмежень на тип об’єкта, який він виявляє, будь то тварина чи людини.

Висновки до розділу 1

Цей розділ охоплює змістовний контекс поточного дослідження. Розглянуто поняття 4IR, її технологічні основи та вплив на сільськогосподарський сектор. Також досліджено впровадження IoT в країнах, що мають посушливі регіони, зручність використання його компонентів, а також деякі з існуючі інтелектуальні системи зрошення IoT.

Нарешті представлено проекти, побудовані за допомогою Arduino і Raspberry Pi, а також численні програмні засоби, які будуть використовуватись у подальших розділах роботи.

Наступний розділ зосереджується на початковому проектуванні інтелектуальної системи зрошення за допомогою класичного підходу до побудови системи - зверху вниз із використанням низки сценаріїв і безпосередньо на основі матеріалу, розглянутого в цьому розділі.

РОЗДІЛ 2. СТРУКТУРНИЙ АНАЛІЗ І ВИБІР КОМПОНЕНТІВ СИСТЕМИ

2.1 Функціональна архітектура

2.1.1 Функціональність вузлів

Можна розглянути поле, яке має досить велике розміри, щоб врахувати різні властивості, наприклад тип грунту, вплив сонця, а також різні градієнти рельєфу. У цьому випадку ми хочемо контролювати вологість ґрунту кожної окремої ділянки відповідно до її специфічних потреб у зрошенні. Це означає, що ми повинні розмістити елементи або «вузли», як показано стрілками на рисунку 2.1, визначати вологість ґрунту на кожній ділянці та автоматично вмикати/вимикати водяні клапани.

Виходячи з функціональності, кожен вузол повинен бути обладнаний як мінімум датчиком вологості ґрунту, електроклапаном, як виконавчий механізм, і програмованим блоком, який періодично обробляє інформацію від датчика, і вирішіть, відкрити водяний клапан чи закрити.

Рисунок 2.1 - Ілюстрація розташування вузлів, розміщених у полі

Можна припустити, що дві ділянки одночасно переходять в режим підсушування, але немає достатньої кількості води для поливу обох ділянок. Потрібно прагнути, щоб система могла поливати ділянки на основі пріоритетності, враховуючи тип рослин у кожній частині поля та встановлювати пріоритети рослин, які є найбільшими чутливими до нестачі води. У цьому випадку замість того, щоб кожен вузол приймав рішення ізольовано, необхідно, щоб вузли могли спілкуватися з якимось інтелектуальним «супервузлом», який керує загальним процесом зрошування (див. рис. 2.2).

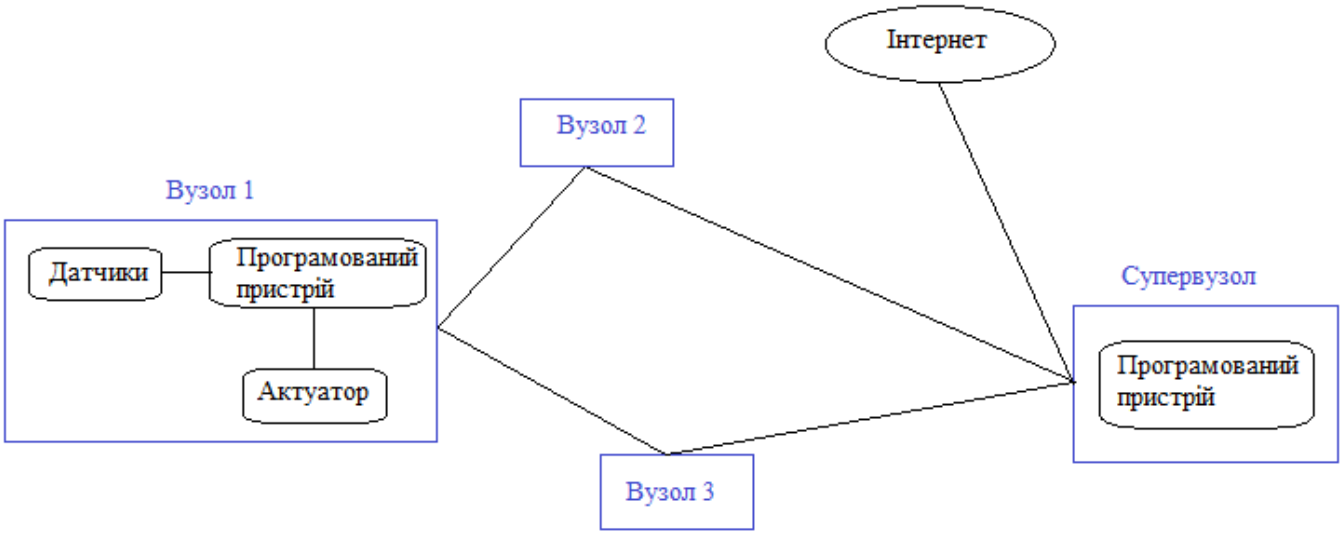

Рисунок 2.2 - Невелика сітчаста мережа

Завдяки такій архітектурі вузлів, система вимірює дані про вологість ґрунту, і передайте дані супервузлу, який потім приймає рішення про найбільш коректні дії поливу, врахувати фактори, які виходять за межі досяжності конкретного вузла.

2.1.2 Системний інтелект

Природні погодні умови, звичайно, є фактором, якого ми не можемо уникнути, будуючи архітектуру системи зрошення. Отже, давайте далі уявимо, що вузол переходить у «сухий» режим безпосередньо перед сильною зливою. Полив ділянки призведе до втрати водних ресурсів і, можливо, навіть до ерозії ґрунту і затоплення рослин. Крім того, можуть бути місцеві обмеження, такі як низький рівень греблі, яка

повинна бути керованою. Тому необхідно, щоб система могла використовувати підключення до більшого Інтернету, маючи доступ до прогнозування погоди (наприклад, опадів) для забезпечення покращеної підтримки прийняття рішень для зрошення, щоб система приймала рішення не тільки на основі вологості ґрунту та потреби рослин у воді, але й враховувати прогноз погоди в цьому районі. Таким чином, коли вузол повідомляє про сухий стан безпосередньо перед дощем, система повинна відкласти зрошення за допомогою активації, щоб тримати клапан вузла закритим, поки вузол знову не досягне сухого стану.

Зпрощена ілюстрація того, як клієнт керує зрошенням вузла на основі обох датчиків: місцеві дані та прогноз погоди приведено на рисунку 2.3.

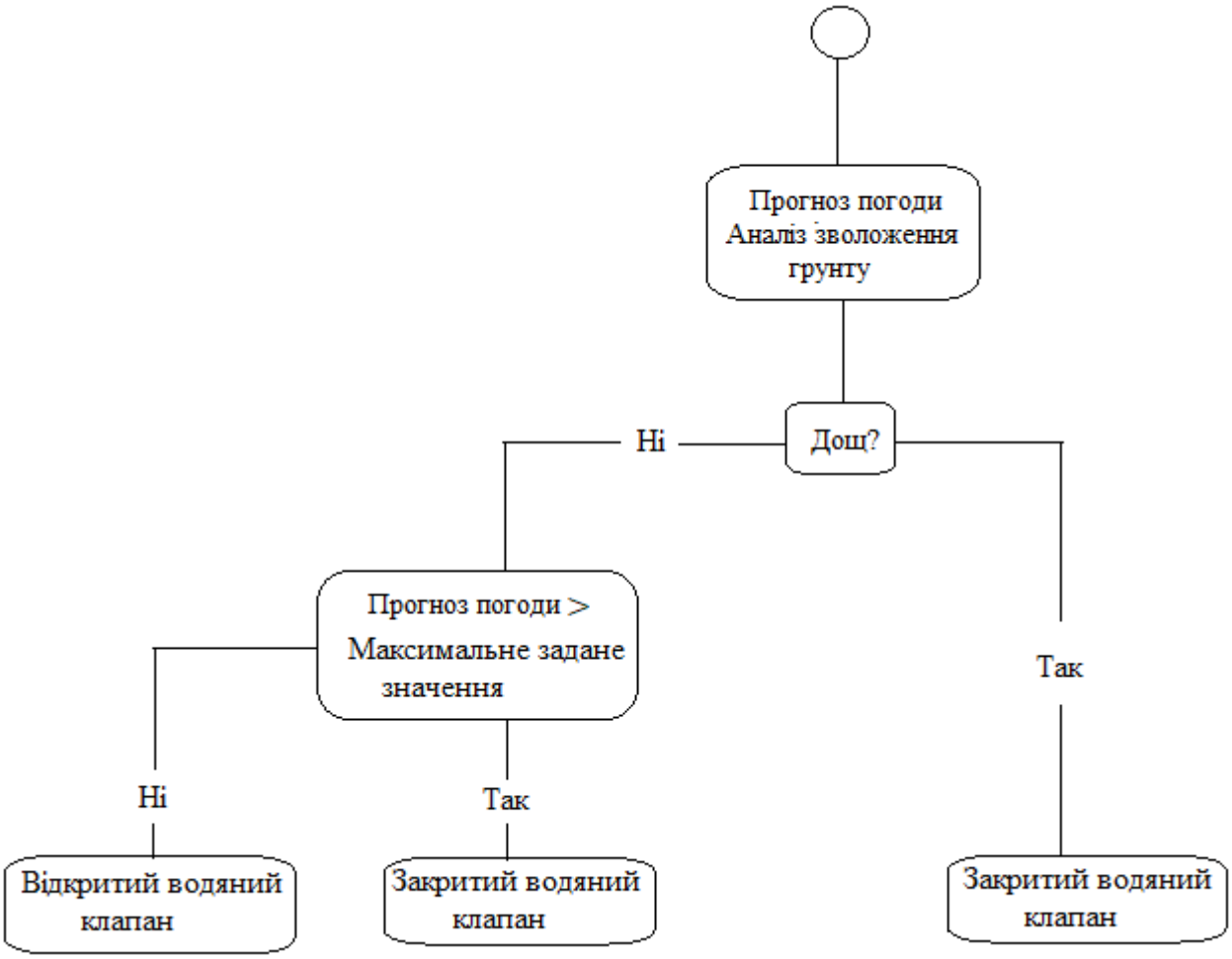

Рисунок 2.3 - Спрощена ілюстрація системного інтелекту

Додавання цієї можливості до розроблюваної системи для зв’язку з більшою мережею реалізується повною мірою за допомогою архітектури IoT, додавши компонент хмарних обчислень. Можна застосувати хмарні обчислення аналізу даних, наприклад схеми візуалізації, щоб показати продуктивність пристрою, виявити неефективність, і придумати способи оптимізації нашої системи. IoT також надає можливості будувати схеми та кореляції з хронологічними даними, які можуть додатково сприяти створенню алгоритмів для покращення автоматизації нашої системи за допомогою машинного навчання.

2.1.3 Зв'язок між вузлами

Зв'язок між системними вузлами, необхідний для першого розглянутого сценарію, може бути досягнено в дротовій або бездротовій мережі. У дротових мережах для підключення пристроїв до мережі використовуються фізичні дроти, причому кожен пристрій під’єднано окремим проводом і передає дані з однаковою швидкістю.

Дротові мережі вважаються безпечними: вимагають фізичного втручання в мережу для отримання доступу до неї, що означало б порушення фізичних кордонів. Однак, проводові з'єднання дорогі в установці, враховуючи вартість проводів, монтажу і виконання робіт.

Можна уявити, що є необхідність додати в мережу ще один вузол, щоб зрошувати нову ділянку. Щоб розширити дротову мережу (тобто додати новий вузол), знадобляться додаткові дроти, і витрати на оплату праці. Аналогічно, якщо, наприклад, два вузли на двох різних ділянках постійно дають ті самі показники вологості грунту, немає сенсу тримати вузли в тому ж положенні. Так, один вузол стає зайвим, і тому є потреба фізично перемістити один вузол, щоб задіяти іншу ділянку. Таке завдання було б складним у дротовій мережі: для зміни розташування будь-якого пристрою потрібні додаткові кабелі для підключення пристрою до мережі.

З іншого боку, бездротові мережі дозволяють бездротовим пристроям спілкуватися через радіосигнали, таким чином усуваючи потребу у фізичних

проводах. Вони потребують менше обладнання що робить їх загалом дешевшими, легшими та швидшими у встановленні, порівняно з дротовими мережами. Вони не обмежують свої підключені пристрої фіксованою зоною, пристрої можуть отримати доступ до мережі будь-де в межах дії відповідного сигналу.

2.1.4 Меш-комунікація

Підключення вузлів до «супервузла» в бездротовій мережі ставить питання про те, як зробити підключення кожного вузла до супервузла. Одним із способів зробити це, було б розмістити вузли так, щоб всі вони були «видимі» супервузлу (видимий у цьому випадку означає радіо доступний, враховуючи частоту та потужність використовуваних радіоприймачів). Однак це означало б що розмір бездротової мережі може досягати лише кола, що оточує супервузол з припущенням, що в колі немає перешкод. Якщо переміщувати вузол за межі кола супер-вузла, через, наприклад, причини, які ми вказали раніше, то вузол більше не матиме змоги спілкуватися з супервузлом і більше не зможе надсилати до нього показання вологості грунту, або отримувати команди спрацьовування від супервузла.

Бездротова мережа – це рішення, що підтримує «сітчасту мережу», де кожен із системних вузлів здатний зв’язуватися з вузлом, розташованим поруч з ним, щоб досягти супервузла. Ілюстрація цієї ідеї сітчастої мережі можна побачити на рисунку 2.2, де вузол 1 хоче спілкуватися з супервузлом, і вузол 1 не видимий супервузлу. Потім дані можуть передаватися до супервузла через проміжний вузол 2.

Рішення, яке пропонує можливості автоматичного виявлення маршрутів, також буде корисним. Маршрут відкриття встановлює шлях, що складається з одного або кількох проміжних вузлів, звідки надходить повідомлення. Вихідний вузол переміщується між вузлами, поки не досягне вузла призначення. Маршрут передачі даних складається з одного або кількох вузлів, які діють як проміжні вузли.

Рішення, яке пропонує автоматичне виявлення маршруту означатиме, що саме рішення відповідає за встановлення маршруту між вихідним і кінцевим вузлом через доступні проміжні вузли. Що, якщо проміжний вузол не діє (вузол 2 у даному

випадку)? Автоматичне виявлення маршруту може бути корисним, роблячи можливою реалізацію «самовідновлення»: новий шлях буде встановлений через інший проміжний вузол (тобто вузол 3) (якщо такий вузол існує в системі).

2.1.5 Управління даними

Відомо, що завдяки тому, як будується система, буде кілька джерел даних і споживачів. Наприклад, є вузли, які передають свої дані про вологість ґрунту до супервузола, що робить їх джерелами даних, а супервузол – споживачем даних. Коли супервузол передає команди активації вузлам, супервузол стає джерелом даних, а вузли стають споживачами даних. Коли супервузол отримує інформацію про прогноз погоди, він стає споживачем даних, а хмара — джерелом даних. Тому триває пошук рішення, яке керуватиме потоками даних, тобто забезпечуватиме правильні дані, що будуть передаватися правильному споживачеві. Рішення, яке відокремлюватиме вузли системи від супервузла було б досить корисним.

2.2 Вибір компонентів системи

У цьому пункті визначається та обговорюються загальні категорії компонентів, необхідних для вирішення задачі на основі функціонального аналізу в розділі 1. Ілюстрація цієї інтеграції компонентів приведена на рисунку 2.4.

2.2.1 Вибір виконуючих компонентів

Датчик вологості ґрунту – це недорогий пристрій, який використовується для визначення значення вологості ґрунту [23; 24]. Датчик може вимірювати вміст води в ґрунті на основі змін опору або ємності [25; 26]. Резистивні датчики вологості ґрунту використовують співвідношення між опором і вологістю ґрунту для вимірювання рівня вологості. Такий датчик складається з двох зондів які безпосередньо вносяться

в ґрунт [26]. Коли води більше, то вище значення струму, що проходить ґрунтом, це призводить до нижчого опору та вищого рівня вологості [27].

Ємнісні датчики, з іншого боку, працюють, вимірюючи зміну ємності, що пов'язано з різною діелектричною проникністю, яка відноситься до вмісту води ґрунті [26; 28]. Датчик складається з позитивної та негативної пластин, розділених діелектриком посередині [26]. Ємнісні датчики не піддаються корозії, що забезпечує краще зчитування рівня вологості, але вони, як правило, дорожчі, ніж резистивні [27].

Датчики зазвичай включають аналого-цифровий перетворювач (АЦП), який перетворює аналогові сигнали в двійкову систему, що дозволяє їм передавати значення вологості ґрунту як число.

Привод

Іншим компонентом, необхідним у кожному вузлі, є привод. Привод - це пристрій, який виконує фізичні рухи на основі енергії та відповідних сигналів [29]. Лінійні приводи здійснюють рух вперед або назад в заданій лінійній площині. Поворотні приводи, з іншого боку, обертається в круговій площині і не обмежується заданим шляхом, вони можуть обертатися в одному і тому ж напряму на стільки, скільки нам потрібно [29].

Приводи також класифікуються на основі їх джерела живлення: електричні, пневматичні та гідравлічні [31]. Пневматичні та гідравлічні приводи мають поршень всередині циліндра, і в основному підходять для досить великої потужностей, пов'язаних з великими амплітудами руху, чого не передбачено в системі, яку ми будуємо. З іншого боку, електричні приводи використовують електроенергію для активації електричних пристроїв, таких як клапани та двигуни. Ці приводи зазвичай використовуються для керуючої дії типу включення-вимкнення за допомогою електричних сигналів. Це водяні електроклапани, які зазвичай використовуються в автоматичному зрошенні.

2.2.2 Програмовані елементи

Мікроконтролер

Для вузла потрібний програмований блок у вигляді мікроконтролера. Мікроконтролер - це невеликий, недорогий і автономний комп'ютер на мікропроцесорі. Мікроконтролери часто вбудовуються всередині виконавчих пристроїв для управління функціями останніх [31]. Мікроконтролери коштують дешевше та мають меншу потужність, ніж мікрокомп’ютери (тому вони можуть працювати від батареї, якщо необхідно), і можуть використовуватися з широким спектром електронного обладнання через: особливості їх введення/виводу; датчик і виконавчий механізм, у нашому випадку [31; 32]. На відміну від мікрокомп'ютерів, мікроконтролер може працювати без операційної системи, що дає повний контроль над системою програмісту та зменшує шанси до відмов.

Мікроконтролери класифікуються за категоріями відповідно до їх арифметично-логічного розміру (у бітах), пам'яті, архітектури та характеристик набору функцій [33]. Арифметичний розмір мікроконтролера вказує на максимальний розмір двійкових даних, який може обробляти мікроконтролер як єдине ціле [32]. Ці відмінності висвітлюються під час математичних операцій. А більша кількість бітів покращує продуктивність і точність мікроконтролера [32]. Однак наша система не вимагає високих обчислювальних можливостей, а отже, навіть мікроконтролера з мінімальним розміром арифметичної логіки (8 біт) нам буде достатньо.

Мікроконтролери підтримують набори машинних інструкцій за допомогою Complex Instruction Set Com-комп’ютер (CISC), або комп’ютер зі скороченим набором команд (RISC) [33]. Мета архітектури CISC - виконати завдання з мінімальними рядками коду і дозволяє програмістам використовувати одну інструкцію для виконання кількох низькорівневих операцій (наприклад, завантаження з пам'яті або зберігання в пам'яті) [34].

З архітектурою CISC швидкість процесора обмежена одним сегментом. Для виконання може знадобитися більше одного розміру тактової частоти, а також архітектура розширює складність програми [35]. RISC, з іншого боку, використовує прості команди, які можна розбити на кілька сегментів, які досягають низькорівневих операцій за один такт [34].

CISC - архітектура збільшує швидкість процесора, однак вона має велику кеш-пам'ять на процесорі [35]. Знову ж таки, оскільки система, яку ми будуємо, не вимагає великих обчислень, архітектурні відмінності для нас не суттєві.

Архітектура пам'яті мікроконтролера буває двох видів: Гарвардська та Прінстонська [33].

Гарвардська архітектура пам'яті фізично відокремлює інструкції та дані, шляхи від пам'яті, це означає, що центральний процесор може читати інструкції та здійснювати доступ до пам'яті даних одночасно [36]. Ця архітектура покращує швидкість виконання програми та потребує більшої апаратної складності [36].

Прінстонська архітектура, з іншого боку, зберігає інструкції та дані в одній пам’яті, тому використовує той самий шлях для отримання інструкцій і даних [36]. Це означає, що мікроконтролер не може одночасно читати інструкцію і виконувати доступ до пам'яті даних, що робить Гарвардську архітектуру пам'яті швидше ніж Прінстонська архітектура [36]. Ці архітектурні відмінності важливі в конкретному налаштування, але не в системі, що будується у даному випадку.

Мікро-комп'ютер

Хоча використання мікроконтролера є хорошим вибором для системних вузлів, мікроконтролери працюють на обмежених потужності та пам’яті, і зазвичай можуть одночасно запускати лише одну програму. З іншого боку, для супервузла потрібен програмований блок, потужніший за мікроконтролер, з більшою потужністю обробки та операційною системою. Тому вирішено використовувати мікрокомп'ютер як програмований блок у супервузлі.

Мікрокомп'ютер - це компактний, відносно недорогий комп'ютер. Як і звичайний комп’ютер, мікрокомп’ютер складається з центрального процесора (CPU), блока пам'яті та блока введення/виводу [37]. CPU мікрокомп'ютера називається мікропроцесором і виконує всі арифметичні та логічні операції. А блок пам'яті мікрокомп'ютера реалізований за допомогою певної форми пам'яті лише для читання (ROM) і мікросхеми оперативної пам’яті (RAM) [37]. Оперативна пам'ять зберігає дані CPU і програм, і він непостійний. ПЗУ - це енергонезалежна пам'ять, яка постійно зберігає інструкції, надані її виробником [37]. Блок введення-виведення забезпечує

інтерфейс для введення та пристрої виведення, такі як клавіатура, миша та екран HDMI [37]. На відміну від мікроконтролерів, мікрокомп’ютери можуть працювати з повноцінною операційною системою [37].

2.3 Протоколи обміну даними

2.3.1 Протокол ZigBee

Як зазначалося раніше, потрібно, щоб системні вузли могли спілкуватися з супервузлом максимально легко і надійно. Вівся пошук рішення для бездротового зв’язку, яке пропонує можливості сітчастої мережі, а також автоматичне виявлення маршрутів і розширені можливості самовідновлення.

Хорошим кандидатом для такого рішення є протокол ZigBee або мережа WiFi. Однак Wi-Fi роутери дорогі в порівнянні з пристроями ZigBee. Крім того, Wi-Fi пристрої в цілому споживають більше енергії, ніж пристрої ZigBee, що робить їх менш придатними для використання в обмеженому середовищі, як те, з яким проводиться робота зараз. Ці характеристики виключають мережевий WiFi як варіант для даної системи. Це залишає доцільним протокол ZigBee.

ZigBee - це протокол бездротового зв’язку малої дальності на основі стандарту IEEE 802.15.4, щоб реалізувати безпечний, недорогий, малопотужний бездротовий зв’язок «машина-машина» (M2M) і мережі Інтернету речей (IoT) [38]. ZigBee працює на частоті 2,4 ГГц з швидкістю передачі даних 250 кбіт/с і охоплює діапазон 10-100 метрів [39; 40; 41].

Важливою особливістю ZigBee є можливість підтримувати сітчасту мережу, додаткову до початкової, і топологію мережі, що підтримується стандартом IEEE 802.15.4 [38; 39; 41]. ZigBee пропонує автоматичне визначення маршруту можливості, де маршрут складається з одного або кількох проміжних вузлів, як я вже обговорювалося у розділі 1 [41]. ZigBee також пропонує розширені можливості «самовідновлення», які додатково вирішують проблему зв'язку вузлів із супервузлом, навіть якщо проміжний вузол виходить з ладу або відокремлюється від супервузла,

встановлюючи новий маршрут через доступний проміжний вузол, де це можливо [39]. Протокол забезпечує режим сну з низьким енергоспоживанням. Пристрої в цій технології можуть переходити в режим сну з низьким енергоспоживанням, коли вони не задіяні в обміні даними [41]. Ця можливість корисна для реалізації ідеї пристроїв, які передають між собою дані про вологість ґрунту та отримують команди увімкнення від супервузла.

2.3.2 Протокол MQTT

MQTT – це протокол передачі даних, який зазвичай виконується поверх стеку TCP/IP, і є вихідний від ранніх робіт IoT [41]. MQTT був розроблений як легкий протокол обміну повідомленнями зменшеної пропускної спроможності мережі для обмежених середовищ, таких як низька потужність, обмежена комп'ютерна здатність та пам'ять, а також обмежена пропускна здатність [42]. MQTT реалізує публікацію / шаблонів підписки, який базується на брокері/сервері повідомлень і всіх інших вузлах, розміщених навколо брокера у топології. Шаблон опублікувати/підписатися означає, що абоненти діють незалежно від видавців і навпаки, реалізуючи роз'єднання між даними джерела та споживачі, які обговорювались у розділі 1 [43].

Видавці можуть публікувати дані на сервер, навіть якщо абоненти недоступні, наприклад, сплять [39]. Так само, абоненти можуть отримати опубліковане від брокера, навіть якщо видавці недоступні. [39]. Це встановлює асинхронну комунікацію між видавцями та абонентами [43]. З іншого боку, зв'язок абонентів і видавців із сервером синхронний [39].

Ще одна важлива особливість шаблону повідомлень опублікувати/підписатись, що один видавець може надсилати дані різним абонентам [43]. Тому що, дані опубліковані на сервер, можуть бути доступні кільком абонентам без потреби адресно надсилати дані кожному абоненту окремо [39]. Один абонент має доступ до даних кількох видавців (рис. 2.4 [43]).

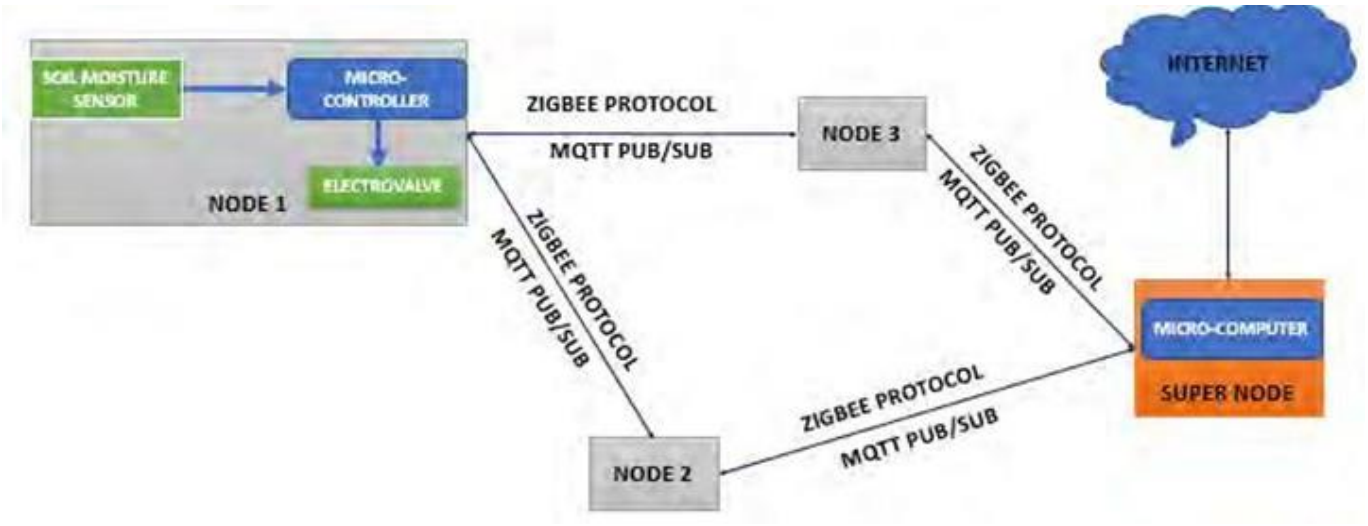

Рисунок 2.4 – Загальна ілюстрація компонентів системи

2.4 Електроклапан, допоміжні елементи системи

2.4.1 Датчик вологості ґрунту серії MH

В даній роботі обрано комерційно доступний датчик вологості ґрунту серії MH, як показано на рисунку 2.5. Датчик є резистивним датчиком вологості ґрунту, тобто він використовує співвідношення між електричним опором і вмістом води для вимірювання рівня вологості ґрунту. Модуль вологості грунту складається з двох частин, основного датчика та електронної частини, яка містить аналого-цифровий перетворювач (АЦП). Датчик складається з двох зондів, які взаємодіють з ґрунтом для вимірювання об’ємного вмісту води в ґрунті. Два зонди подають струм у ґрунт і визначають опір ґрунту, який потім перетворюється в значення вологості [44]. АЦП датчика використовує 10 біт, тому значення відображаються від 0 до 1023[44].

Датчик оснащений як цифровим, так і аналоговим виходом, завдяки чому він може працювати і в тому, і в іншому цифровому та аналоговому режимі [45].

2.4.2 Електроклапан

Вирішено використовувати електроклапан в якості приводного компонента, щоб контролювати полив, тобто перемикач увімкнкти/вимкнути водяний клапан на

основі показань датчика вологості ґрунту. Електроклапан може частково або повністю блокувати трубу, щоб змінити кількість води, яка протікає через неї.

Наразі використовується ідея повного вмикання/вимкнення клапана. Для реалізації поставлених задач, однак, буде застосовуватись імітація електроклапана з використанням двох світлодіодів, де світиться один (верхній) світлодіод, коли ґрунт висохне, а інший (нижній) світиться, коли грунт вологий.

2.4.3 Arduino Uno

Вирішено попрацювати з сімейством мікроконтролерів Arduino. Arduino є платформа з відкритим вихідним кодом розробки апаратних засобів [46]. Платформа є простою та доступною для користувача через її апаратні та програмні компоненти [47]. Наприклад, програмне забезпечення Arduino зроблено простим і легким для початківців, водночас достатньо гнучким для просунутих користувачів [47].

Платформа Arduino широко використовується в Maker Movement і в русі відкритого коду, це означає, що має бути велика підтримка, а інструменти повинні бути достатньо розвиненими для використання, як описано в розділі 1. Як вже зазначалося, природа системи, що будується, не потребує великих обчислювальних можливостей або навіть багато пам'яті, оскільки очікується мати досить невеликі програми Arduino, і тому просто потрібен мікроконтролер, у якому буде достатньо інструментів. У сімействі Arduino з мікроконтролерів, вибрано Arduino Uno. Arduino Uno дуже рекомендує спільнота Arduino, вона є найбільш документованою з усього сімейства Arduino. Він має велику підтримку та допомогу в побудові системи з цим мікроконтролером [48].

Arduino Uno складається з 8-розрядного процесора RISC, 32 КБ флеш-пам'яті, 0,5 КБ використовується для завантажувача [46]. Він також включає 14 цифрових контактів вводу/виводу (з яких 6 можна використовувати як виходи широтно-імпульсної модуляції (ШІМ), які можна використовувати для підключення датчика вологості ґрунту та датчика електроклапан до мікроконтролера [46]. Плата також містить 1 послідовний порт UART, який може використовуватися для послідовного

зв’язку з пристроєм XBee (див. рис 2.5) [46]. Плата може живитися від акумулятора 9 В, що робить його портативним у різних експериментальних налаштуваннях і нарешті в експлуатації за призначенням.

2.4.4 Raspberry Pi

Вирішено використовувати Raspberry Pi як мікрокомп’ютер для даної системи. Raspberry Pi - це сімейство недорогих апаратних платформ або мікрокомп'ютерів з відкритим вихідним кодом [91]. Raspberry Pi - мікрокомп'ютер може працювати з різними операційними системами, включаючи Raspbian - безкоштовний відкритий вихідний дистрибутив Linux, який потім став корисним для підтримки доступної платформи [49]. Як і мікроконтролер Arduino, сімейство Raspberry Pi зробило свій напрямок Maker Movement, що означає хорошу підтримку для рішення такого типу проблеми [50]. У сімействі Raspberry Pi вирішено працювати з Raspberry Pi 3. Raspberry Pi 3 базується основі 64-розрядного чотирьохядерного процесора з тактовою частотою 1,2 ГГц [51]. Він включає в себе 1 ГБ оперативної пам'яті, 4 порти USB, які можна використовувати з периферійними пристроями, такими як клавіатура, миша та порт HDMI для використання комп'ютера з екраном [51]. Raspberry Pi 3 також включає два послідовних порти UART, які можна використовувати для послідовного зв’язоку із пристроєм XBee, як показано на рисунку 2.5 [51]. Він також включає порт Ethernet, через який можна легко підключити комп’ютер до Інтернету, що є основною вимогою до супервузла, як детально описано раніше [51].

2.4.5 ZigBee XBee

Прийнято використовувати широко доступне в продажу сімейство радіочастотних модулів XBee, вироблене міжнародною компанією Digi. Ці модулі дійсно зазвичай використовуються для забезпечення підключення до електронних пристроїв, таких як мікроконтролери та датчики [52]. Модулі Digi XBee можуть бути

легко підключеними до інтелектуального пристрою, мікроконтролера Arduino UNO і Raspberry Pi у даному випадку через послідовний інтерфейс UART [52]. Крім того, вони працюють як автономні через їх програмовані варіанти, що усуває потребу в мікроконтролері [52].

Ці модулі не вимагають додаткової розробки або взагалі не потребують сторонніх втручань і легко налаштовуються [52]. Вони також залишили слід у великій спільноті Maker і відкритих вихідних кодів, це, звісно, означає, що існує велика підтримка, яка допоможе розпочати роботу, і, знову ж таки, доцільно залишатися в межах цих напрямків [52].

Модулі Digi XBee RF класифікуються з серії від 1 до 3, з різними можливостями [53]. РЧ-модулі XBee Series 1, також відомі як XBee 802.15.4 реалізують стек IEEE 802.15.4 і підтримують зв'язок у запатентованій сітчастій мережі [53]. Ці модулі добре підходять для простих систем невеликого розміру, вони діють як проста заміна кабелю [53]. Модулі XBee Series 2 або XBee ZigBee модулі створені для більш складних і великих мереж, вони реалізують повний стек ZigBee який побудований на основі стека IEEE 802.15.4 і підтримують сітчасту мережу ZigBee з покращеним діапазонjv і меншим споживанням енергії, в порівнянні з модулями серії 1 [53].

Модулі XBee серії 3 додають нові можливості, такі як структура безпеки Digi TrustFence, низький рівень Bluetooth energy (BLE) локального введення в експлуатацію та впроваджено MicroPython, що усуває потребу в зовнішньому мікроконтролері [52]. Вирішено працювати з радіочастотними модулями Digi XBee Series 2, оскільки їх достатньо для реалізації сітчастої мережі ZigBee, виявлення маршрутів та можливості самовідновлення [53].

2.4.6 Протокол обміну повідомленнями MQTT

Необхідно знайти рішення MQTT для реалізації моделі видавця/передплатника MQTT. Бажано, щоб платформа могла підтримувати Raspberry Pi і Arduino UNO мікроконтролери. Хорошим кандидатом для цього є брокер Eclipse Mosquitto MQTT [54]. Mosquitto реалізує брокер MQTT і бібліотеку клієнтів MQTT, виконуючи обмін

повідомленнями за допомогою моделі опублікувати/підписатися [54]. Mosquitto є відкритим вихідним кодом, а це означає, що має бути доступна підтримка, а інструменти мають бути достатньо розвиненими для використання [26]. Mosquitto легкий і може працювати з широким спектром пристроїв, від малопотужних одиночних плат до повноцінних комп'ютерів, включаючи Raspberry Pi [54].

Для MQTT потрібні складні протоколи транспортного рівня наприклад, TCP/IP, не передбачений протоколом ZigBee, який заплановано використовувати. З'явився кандидат, який буде хорошим варіантом MQTT, відомим як MQTT-SN. MQTT-SN був спеціально розроблений для малопотужних і недорогих пристроїв, щоб додати можливість використання сенсорних мереж з мінімальні ресурсами енергоспоживання та обмеженими можливостями обробки, такі як ZigBee, без перспективних шаблонів обміну повідомленнями абонента/видавця MQTT [55]. Протокол вводить шлюз між клієнтами MQTT-SN і брокером MQTT [55]. Шлюз виконує протокольне перетворення між клієнтами MQTT-SN, які працюють через стек TCP/IP, і MQTT брокером, що працює поверх стеку TCP/IP [55]. Тому вирішено вивчити цей MQTT-SN протокол, зокрема, щоб визначити, наскільки зрілими будуть інструменти та наскільки легко було б створити проектовану систему з ним. Як розглядалося в розділі 1, цей вибір довелося змінити, і завершувати створення розроблюваного шлюзу.

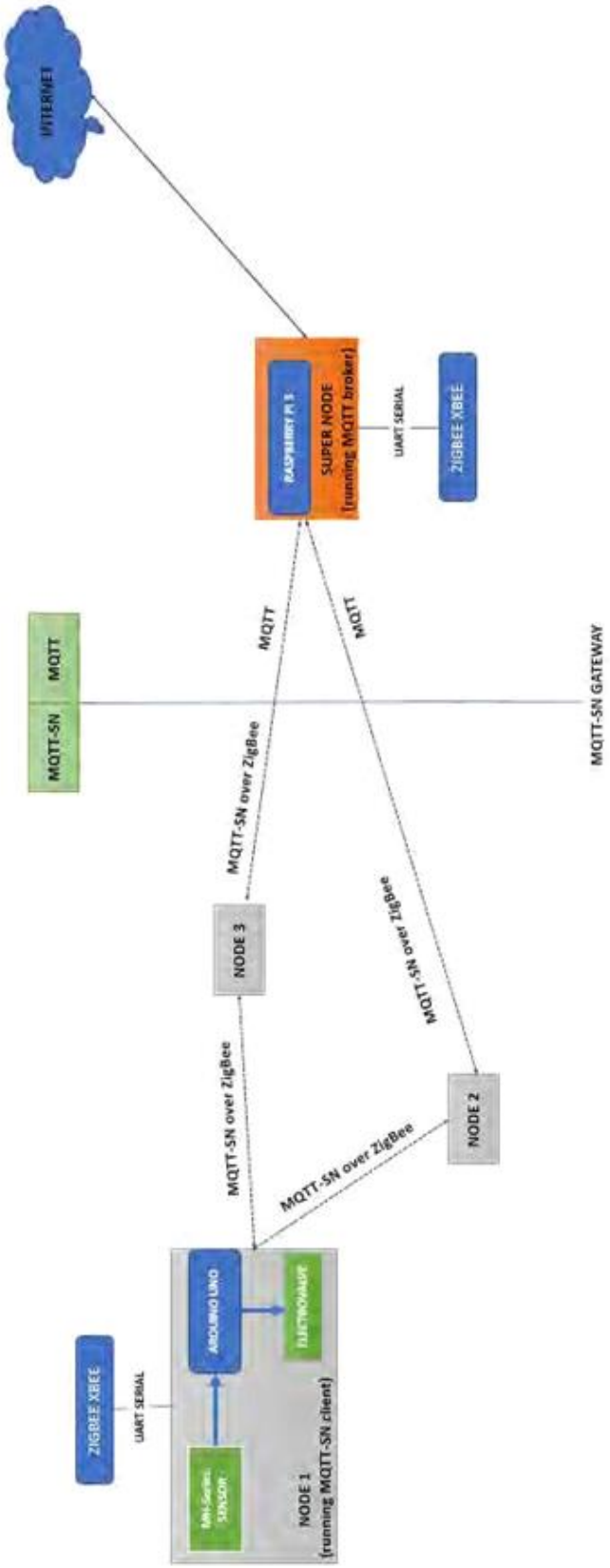

Рисунок 2.5 - Ілюстрація повних компонентів системи

Висновки до розділу 2

У цьому розділі йде мова про початок розробки за архітектурою - зверху вниз. У цьому дослідженні рішення було прийнято через серію експериментів. У цій роботі були визначені необхідні загальні компоненти для побудови інтелектуальної системи зрошення. Після цього, готові і добре- підтримувані апаратні та програмні засоби, що реалізують визначені загальні компоненти, була обрана система. У наступному розділі буде розглянуто впровадження інтелектуальної системи зрошення з вибраними апаратними та програмними засобами.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПОЛИВУ

3.1 Інструменти та компоненти

Реалізується шлюз і додаткові компоненти за допомогою апаратних і програмних засобів які було обрано і досліджено у попередніх розділах. З інструментів обирались ті, які вважаються простішими у використанні. Ці апаратні та програмні засоби перераховані нижче.

3.1.1 Апаратне забезпечення

- Raspberry Pi 3;
- Плата Arduino Uno;
- Віддалений комп’ютер;
- 2 пристрої XBee S2C, налаштовані в мережі ZigBee.

3.1.2 Програмне забезпечення

- Arduino IDE;
- IDE Python;
- Бібліотека Arduino XBee;
- Бібліотека Python XBee;
- Бібліотека Python MQTT;
- Бібліотека MongoDB;
- Брокер Mosquitto.

Рисунок 3.1 ілюструє різні компоненти системи, які описано у попередньому розділі.

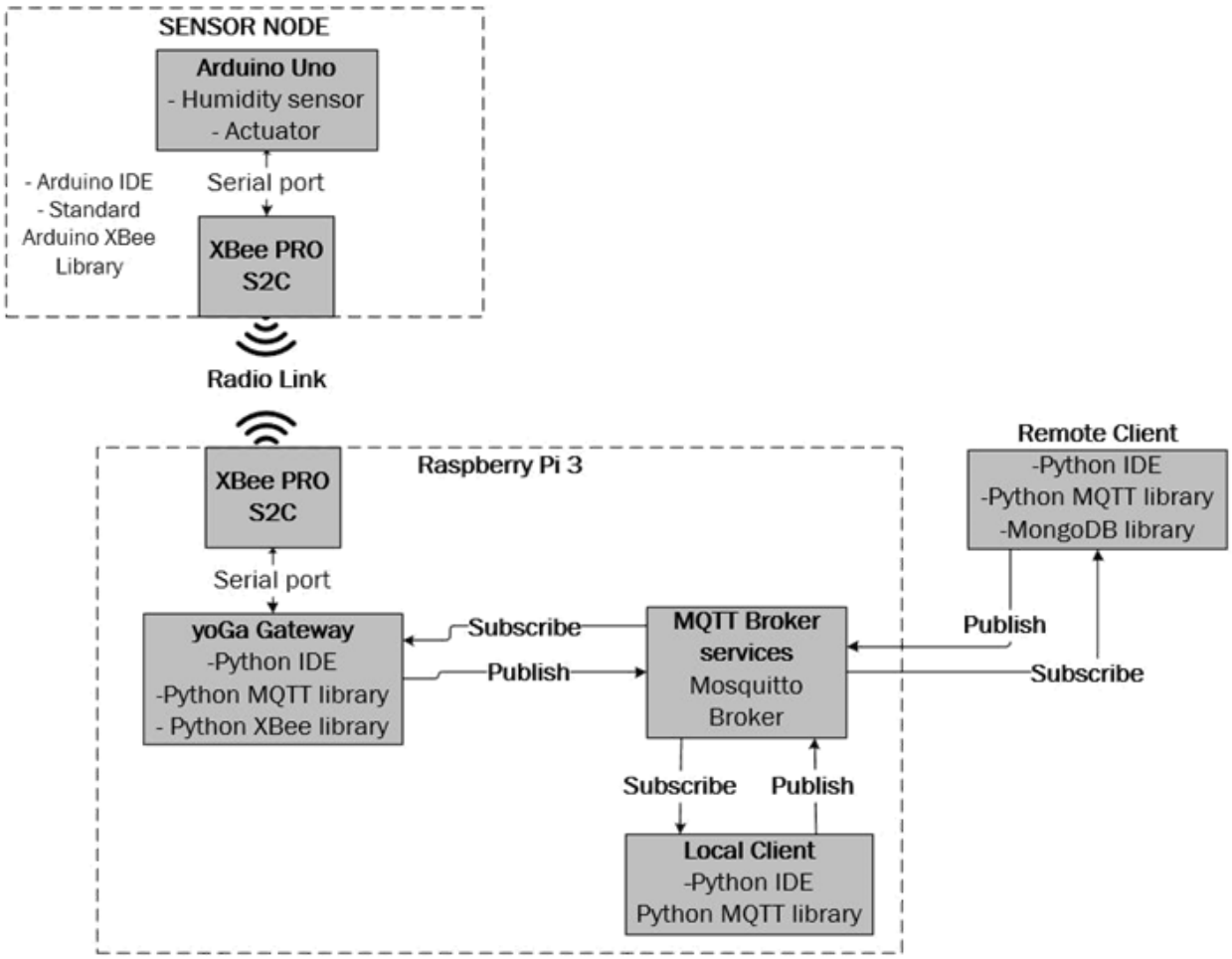

Рисунок 3.1 - Ілюстрація інструментів і компонентів реалізації системи

3.2 Загальна архітектура

Шлюз складається з двох інтерфейсів, інтерфейсу ZigBee та інтерфейсу MQTT. Інтерфейс ZigBee прослуховує повідомлення (публікації або запити), що надходять від вузлів у мережі ZigBee. Через цей інтерфейс шлюз також посилає на вузли команди активації, які він отримує від клієнтів MQTT через інтерфейс MQTT. Шлюз MQTT інтерфейс публікує дані датчика від імені вузлів брокеру MQTT. У цьому інтерфейсі шлюз також прослуховує повідомлення від клієнтів MQTT, які потім надсилаються на датчик вузлів через інтерфейс ZigBee, як уже обговорювалося раніше.

Сенсорні вузли складаються з одного інтерфейсу ZigBee, який надсилає повідомлення про звіт клієнтам MQTT через шлюз, і потім надсилає запит на дію від

шлюзу. Локальний контролер складається з MQTT інтерфейсу. У цьому інтерфейсі контролер прослуховує повідомлення від шлюзу та публікує команди дій до шлюзу через брокер MQTT. Панель бази даних складається з інтерфейсу MQTT, а також інтерфейсу серверної бази даних. Інтерфейс MQTT - це блокування виклику, який прослуховує повідомлення від інших клієнтів MQTT. Внутрішній інтерфейс просто зберігає інформацію, яку передають повідомлення MQTT, у запущену базу даних MongoDB.

3.2.1 Структури даних

Представлення вузлів у шлюзі

Щоб представити вузли у шлюзі, вирішено використати бібліотеку. Бібліотека загалом є прийнятним варіантом, коли є список унікальних ключів, які відповідають значенням, де записів немає в жодному конкретному запиті. У мережі ZigBee вузли ідентифікуються за їх унікальною MAC-адресою. Це також важливо для шлюзу, решти систем, якими можуть бути вузли однозначно ідентифіковані. В результаті структура бібліотеки була більш придатною, ніж масив чи проста структура списку.

Шлюз використовує ці бібліотеки для тимчасового збереження вхідних даних вузлів від клієнтів MQTT, поки вузол не запитає дію. Значення цієї ідеї полягає у передачі даних вузлам лише на запит від вузла, тому що він імітує «абонента» в протоколі MQTT.

Представлення вузлів у локальному контролері

Щоб представити вузли в локальному контролері, вирішено знову використати бібліотеки з тих причин, які обговорювались у попередньому підрозділі. У бібліотеці контролера зберігається сільськогосподарська інформація про вузли, де кожен вузол також ідентифікується своєю MAC - адресою. Ця інформація про вузол включає інформацію про GPS, пріоритет, стан клапана, тип посіву тощо. Значення для бібліотеки в цьому контролері полягає в тому, що він зберігає стан вузлів. Таким чином, контролер знає, які вузли в даний момент поливаються, і в яких вузлах відключена вода. Для обсягу цієї реалізації та тестування, бібліотека створюється у

спрощеному форматі та заповнюється прикладними значеннями. Однак це можна відтворити на основі фактичної інформації від фермерів, а потім зберігати в базі даних.

Представлення значень вузлів у Database Saver

Бібліотека використовує накопичувач бази даних, який зберігає значення вологості вхідних вузлів. Ця перевага для бібліотеки полягає в тому, що значення вологості та дії вузла не надходять до накопичувача бази даних одночасно. Значення вологості вузла публікується шлюзом до MQTT брокера, а потім зчитується засобом збереження бази даних і локальним контролером. Далі локальний контролер публікує відповідну дію вузла в брокері MQTT, і тільки потім дію вузла зчитує засіб збереження бази даних (а також шлюз). В результаті, бібліотека вузлів використовується для тимчасового збереження значення вологості вузла до моменту прийняття дій вузлом. І лише тоді значення вузла зберігається в базі даних як єдиний запис.

Щоб розпочати процес проектування, складено схему, за якою працюватиме шлюз, як показано на рисунку 3.2. Схема заснована на дослідженнях, про які було описано у попередніх розділах роботи. Як видно, шлюз буде розташований у супер-вузлі.

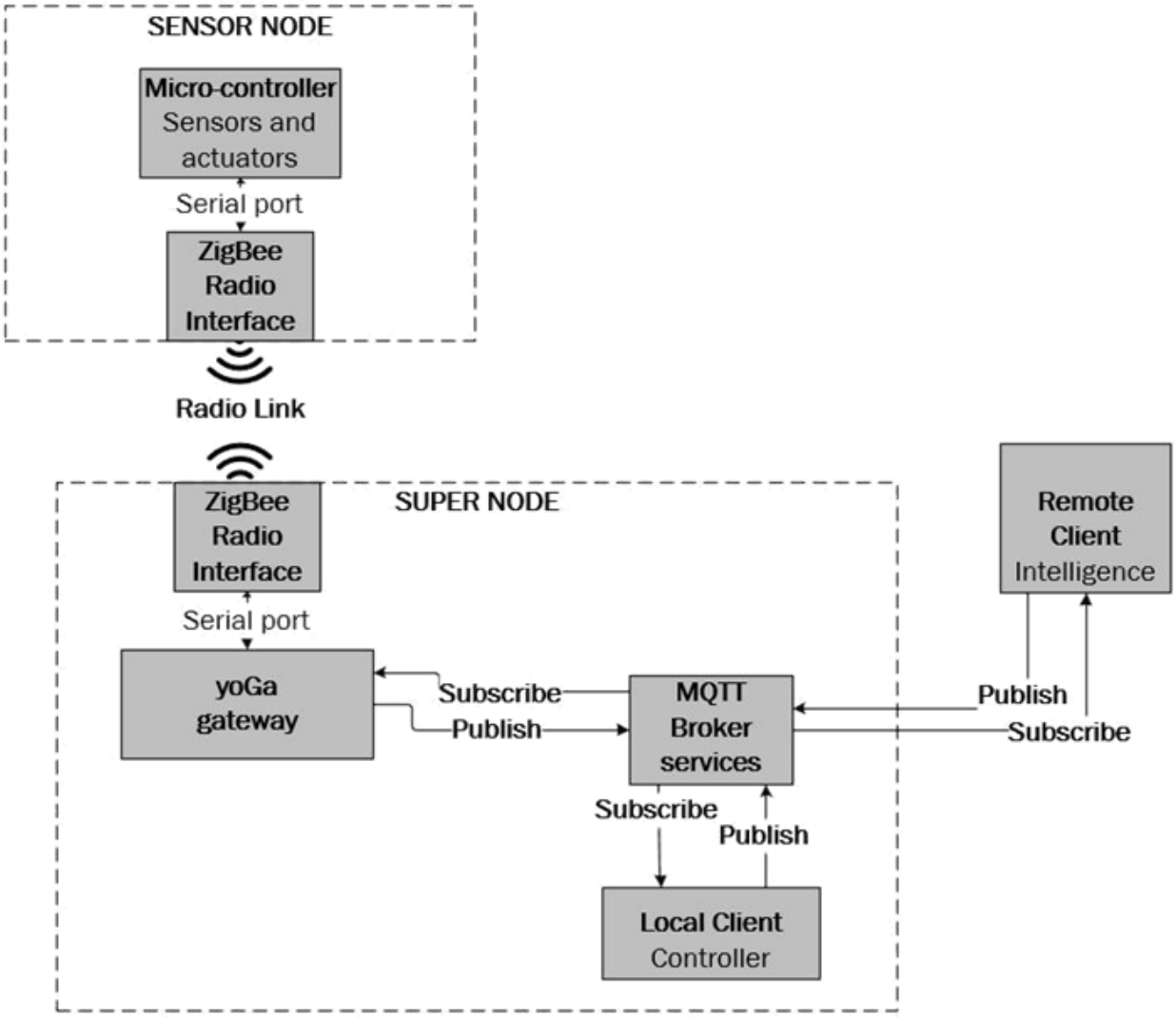

Рисунок 3.2 – Схематичне зображення розроблюваної системи

3.2.2 Архітектура високого рівня

Як показано на малюнку 3.2, шлюз можна розглядати як приймач і передавач ZigBee, пов’язаних з клієнтом абонента та видавцев MQTT. Інтерфейс ZigBee взаємодіє з датчиками вузлів мережі ZigBee. З іншого боку, інтерфейс MQTT взаємодіє зі стандартним протоколом MQTT.

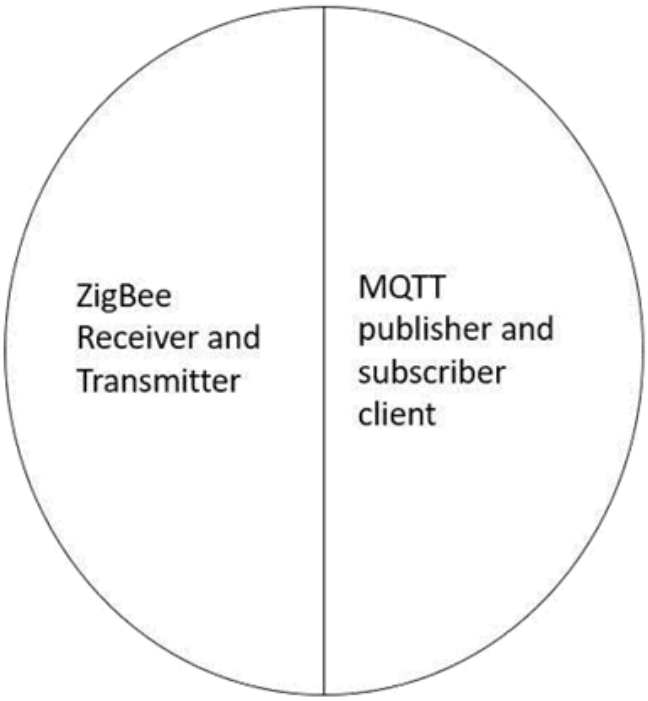

Рисунок 3.2 - Архітектура високого рівня системи

3.2.3 Проектні обмеження

Архітектура шлюзу характеризується наступним.

По-перше, потрібно, щоб шлюз забезпечував простоту коду в сенсорних вузлах, щоб зробити його використання простим і для новачків, використовувуючи обчислювальні пристрої на вузлі без зайвих ускладнень.

По-друге, потрібно гарантувати можливість проведення більшої частину часу у сні для вузлів, які не є маршрутизаторами.

Хостом шлюзу є мікрокомп’ютер Raspberry Pi, який працює з обмеженою пам’яттю (менше 1 ГБ і зовнішньою SD-картою).

Наявність шлюзу як протоколу без стану означає, що коли він спілкується з вузлами, він не зберігає стан сеансу з попередніх запитів. Це в подальшому зменшує вимоги за об’ємом пам’яті до шлюзу. Наявність шлюзу без стану досить корисна для випадків, де шлюз спілкується з більшою мережею вузлів. У такому випадку, якщо шлюз було реалізовано як із збереженням стану, тоді він повинен був би зберегти стан сеансу для кожного з’єднання з кожним вузлом, що є не дуже доцільним варіантом

для середовища з обмеженою пам'яттю. Крім пам'яті, реалізація шлюзу, як протоколу без стану спрощує роботу відновлення після помилки, яка змушує перезапустити шлюз, на відміну від протоколу з визначенням стану, де сервер втрачає весь свій нестабільний стан у стані помилки, у протоколі без стану.

Наслідки збою та відновлення сервера майже непомітні і полегшують відновлення від стану помилки, яка змушує перезапустити шлюз.

3.3 Вузли датчика

Як обговорювалося раніше, вузол надсилає повідомлення, що містить дані датчика, і запит на повідомлення для дії приводу над ZigBee. Для виконання цього завдання сенсорний вузол використовує бібліотеку Arduino XBee для спілкування з радіопристроями XBee. У бібліотеці XBee вузол використовує об'єкти наступних класів:

- XBee, який містить функції надсилання та читання вхідних пакетів API;
- ZBTx який представляє пакет API запиту ZigBee;
- ZBRx який представляє пакет API відповіді ZigBee;
- XBeeAddress64 який представляє 64-розрядну мережеву адресу пристрою XBee і встановлюється щоб утримувати MAC-адресу віддаленого пристрою XBee, підключеного до Raspberry Pi.

У своїй функції налаштування вузол датчика ініціалізує послідовний порт, який потім встановлюється як послідовний порт об’єкта XBee, який був створений раніше за допомогою функції SetSerial. Крім того, вузол встановлює адресу віддаленого пристрою XBee, запитуючи пакет до об'єкта XBeeAddress64, який представляє пристрій XBee в Raspberry Pi, використовуючи встановлену функцію SerialAddress64. Нарешті, вузол налаштовує цифровий контакт 13 плати Arduino як вихідний контакт, який буде представляти собою актуатор. Для цілей простоти тестування вузол використовує світлодіод як активатор (вмикаючи його в залежності від отриманої дії) замість фактичного водяного клапана.

Перед функцією циклу створюються три змінні, одна змінна для утримання значення вологості грунту вузла та дві змінні для утримання значення вологості MSB та LSB відповідно. Знову для простоти значення вологості вузла не засноване на показаннях фактичного датчика вологості, але використовує призначені показники.

Функція циклу – це місце, де вузол датчика оновлює значення вологості та надсилає повідомлення і обробляє вхідні повідомлення від шлюзу через мережу ZigBee. Вузол використовує логіку заглушки для оновлення значення зволоження вузла, просто збільшуючи змінну вологості на 1 кожен раз, коли вводиться цикл. Один раз змінна перевищує 1023, потім вона скидається до 0. Вузол використовує ці значення від 0 до 1023, оскільки датчик вологості ґрунту зчитує 10-бітове значення вологості від 0 до 1023 регулярність підвищення значення вологості була обрана для спрощення згодом випробувань.

Вузол надсилає до шлюзу два типи повідомлень: повідомлення про публікацію та повідомлення із запитом, як обговорювалося в попередньому розділі. Вузол створює масив даних для зберігання корисного навантаження, яке буде надіслано в повідомленні. Масив складається з трьох елементів.

Перший елемент масиву даних вказує тип повідомлення, яке передається, «0» вказує на публікацію, а «1» означає запит на дії. Другий і третій елементами є найбільш значущими байтами (MSB) і найменш значущими байтами (LSB) значення вологості. Цей масив даних потім встановлюється як корисне навантаження для пакета запиту, який був створений раніше за допомогою функції Payload. А потім, нарешті відправити, використавши функцію для надіслання повідомлення звіту до шлюзу.

Після відправлення повідомлення про публікацію або звіт, вузол намагається отримати пакет підтвердження ZigBee (який слід відрізняти від відповіді шлюзу). Вузол використовує функцію readPacket для читання та скасування підтвердження пакету edgement, який не несе релевантної інформації, із зазначеним тайм-аутом 4 секунди. Після скасування пакета вузол затримує програму на 2 секунди, перш ніж перейти до надсилання повідомлення із запитом на дію. Ці затримки є лише орієнтовними. Загальне правило полягає в тому, щоб надати достатньо часу решті

системи, щоб обчислити дію приводу у відповідь на публікацію значення вологості. Щоб надіслати запит, вузол встановлює перший байт масиву даних в «1», а потім надсилає повідомлення запиту.

Знову ж таки, вузол очікує пакет підтвердження ZigBee, який також зчитується і відкидуэться. Після відправки повідомлення запиту вузол чекає 2 секунди і прослуховує відповідь від шлюзу протягом 4 секунд, використовуючи функцію readPacket. Потім вузол декодує пакет корисного навантаження, яке містить дію вузла ("0" або "1") і відповідним чином спрацьовує. Вузол запалює світлодіод, щоб «відкрити» клапан, і гасить світлодіод, щоб «закрити» клапан.

3.3.1 Отримання та обробка повідомлення від датчика

Вузол

Якщо вузол датчика (вузол 1) надсилає повідомлення з даними датчика якомусь клієнту MQTT через шлюз ZigBee/MQTT, то потім очікує відповіді через деякий час. Один із способів такої реалізації є те, що вузол надсилає повідомлення зі значенням датчика до шлюзу, і потім дочекавшись відповіді, що містить команду для відповідного спрацьовування, за встановлену кількість часу. Іншим варіант - щоб вузол надсилав окремі повідомлення, перше з даними датчика, а друге - запит команду для активації. Останній метод (як показано на рис. 3.3) імітує систему публікації/підписки та негайно виконує два обмеження збільшення часу сну для вузлів і шлюзу без адреси.

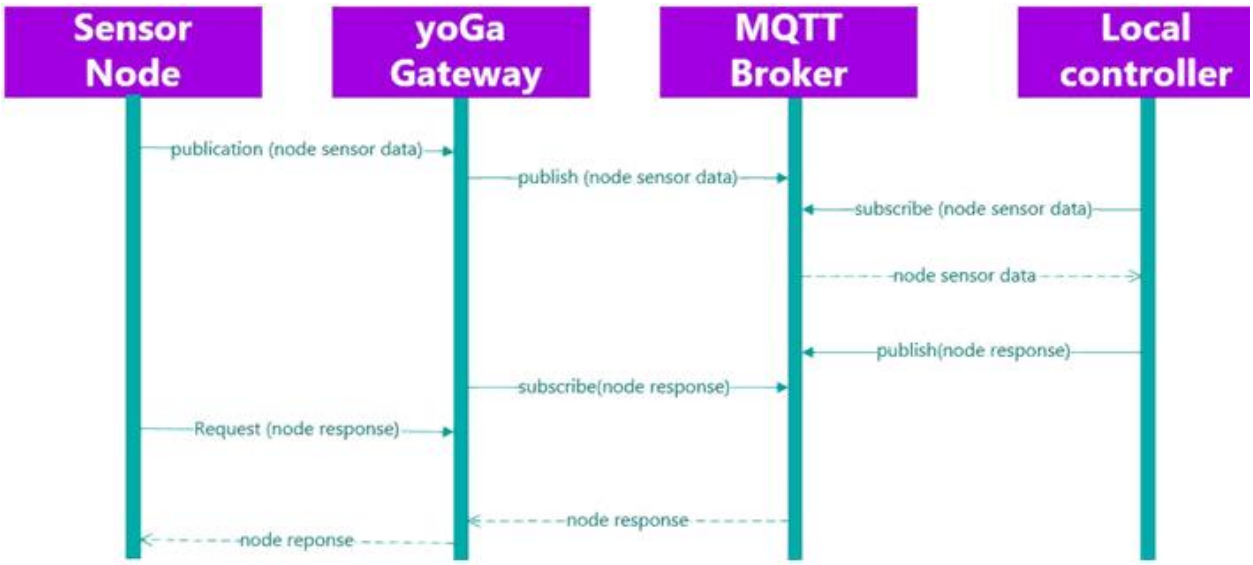

Рисунок 3.3 - Потоки даних від сенсорного вузла до клієнта MQTT через шлюз

3.3.2 Пакети підтвердження ZigBee

Коли два пристрої ZigBee спілкуються, протокол ZigBee використовує механізм підтвердження. Протокол генерує та надсилає відправникові повідомлення-підтвердження пристрою після успішної передачі повідомлення на пристрій-отримувач. Пристрої ZigBee зберігають повідомлення у невеликому обсязі пам'яті, і буфер працює так, як найстаріше повідомлення знаходиться в передній частині черги, а всі вхідні повідомлення додаються в кінці. Це означає, що для обробки видно лише найстаріше повідомлення. Тому що визначення не містить жодної інформації, що стосується програми, вона просто відкидається.

На рисунку 3.4 представлена діаграма стану, що описує зв'язок сенсорного вузла з шлюзом, ілюструючи різні стани вузла від надсилання повідомлення про публікацію до шлюзу, до моменту, коли вузол отримає відповідь від шлюзу та вжиття відповідних заходів.

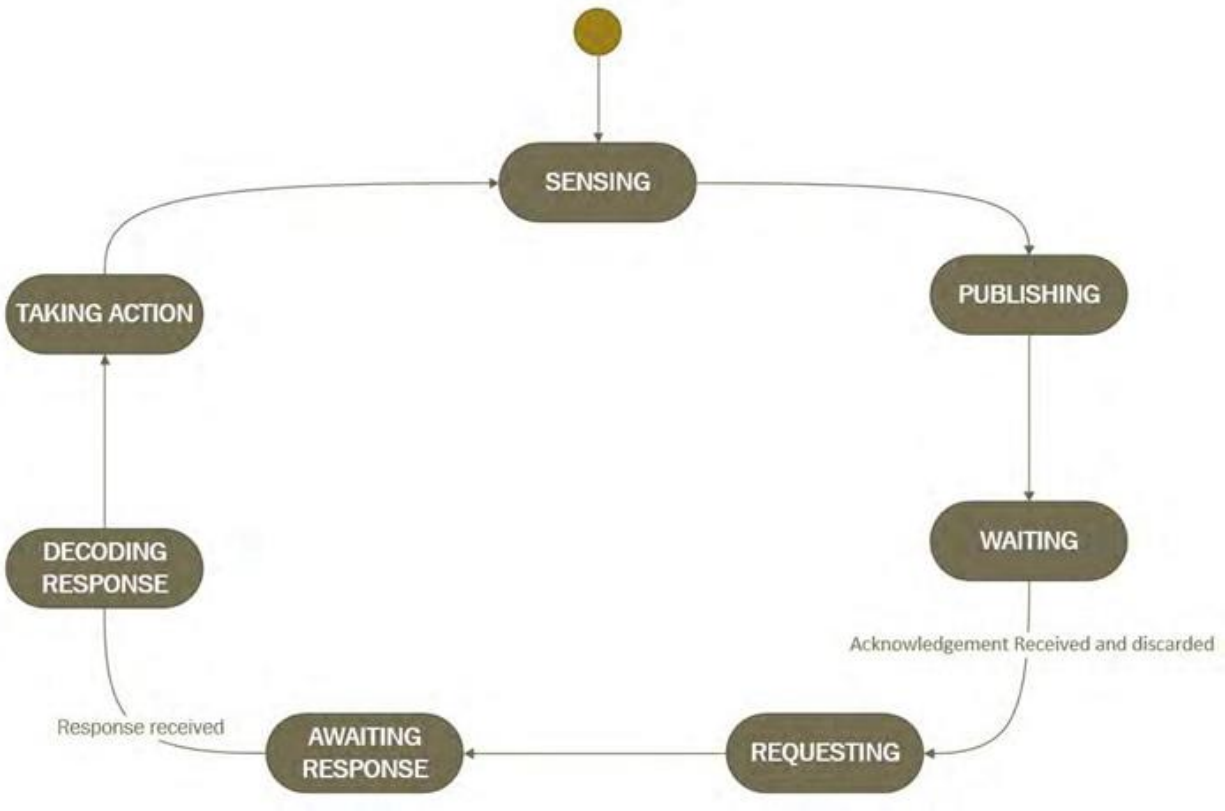

Рисунок 3.4 - Скінченна діаграма вузла датчика

3.3.3 Теми шлюзу MQTT

Як показано на рисунку 3.5, шлюз отримує два типи повідомлень від сенсорного вузла: публікацію та запит (який, з точки зору MQTT, слід розглядати як підписку). Можна припустити, що шлюз отримує повідомлення про публікацію від сенсорного вузла. У такому випадку, шлюз просто публікує повідомлення брокеру MQTT від імені датчика вузола, де опубліковані дані доступні для використання будь-яким іншим клієнтом MQTT. Брокери MQTT організують дані в темах. В результаті при отриманні публікації повідомлення від датчика вузола, шлюз створює тему в брокері MQTT, а потім публікує отримані дані датчика по темі. Щоб будь-який інший клієнт MQTT отримав опубліковані дані, клієнт повинен підписатися на тему в брокері, відповідно.

Можна прийняти, що шлюз отримує дані датчика від більш ніж одного сенсорного вузла. Як дані кожного сенсорного вузла публікуються в брокері MQTT, щоб їх було легко ідентифікувати іншими клієнтами MQTT? У цьому випадку шлюз 2 має створити унікальну тему для кожного вузла. Щоб задовольнити цю унікальну ідентифікацію кожної теми вузла, файл шлюза може використовувати MAC-адресу пристрою ZigBee, оскільки в мережі ZigBee немає вузлів, які, можливо, можуть використовувати ту саму MAC-адресу. І так щоразу шлюз отримує повідомлення публікації від сенсорного вузла, шлюз готує тему вузла, включаючи MAC-адресу вузла, а потім публікує тему даних датчика вузла.

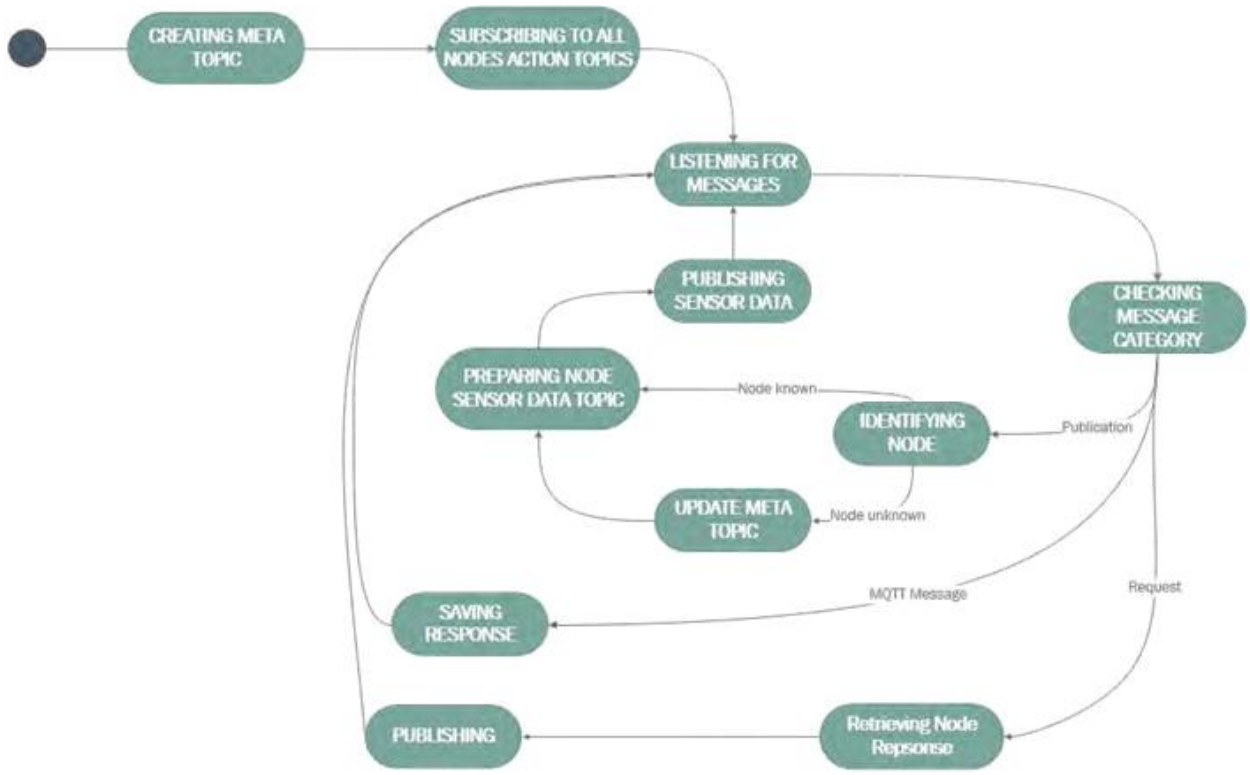

Рисунок 3.5 - Скінченна діаграма шлюзу

3.3.4 Отримання та обробка даних від клієнтів MQTT

До цього моменту досліджувалось, як шлюз отримує та обробляє дані від сенсорних вузлів, де шлюз просто публікує дані датчика у відповідній темі вузла в

брокері MQTT від імені сенсорного вузла, а дані споживаються іншими клієнтами MQTT. Тепер можна припустити, що якийсь клієнт MQTT хотів би використовувати опубліковані дані датчика, і відповідати вузлу відправника через шлюз. Для отримання таких даних клієнтом MQTT, він повинен був би підписатися на тему даних датчика вузла, створену шлюзом, як обговорювалось раніше. Однак вузли датчиків видимі лише для шлюзу, клієнти MQTT не мають інформації про вузли, наприклад, загальна кількість сенсорних вузлів у мережі, їх MAC-адреси та інша важлива інформація про вузли. В результаті було вирішено щоб створити мета-тему, яка передаватиме інформацію про вузли датчика від шлюзу до інших клієнтів MQTT.

Таким чином, шлюз повинен створити цю мета-тему в брокері MQTT, а потім опублікувати інформацію про вузли, наприклад, загальну кількість вузлів датчика системи до теми. Крім того, шлюз оновлює мета-тему щоразу, коли отримує повідомлення від нового сенсорного вузла. Коли справа доходить до відправки, або відповіді на вузол датчика через шлюз, діє той самий принцип підписки та публікації. Клієнт повинен потім опублікувати повідомлення-відповідь у відповідній темі в брокері MQTT, як показано на рисунку 3.6.

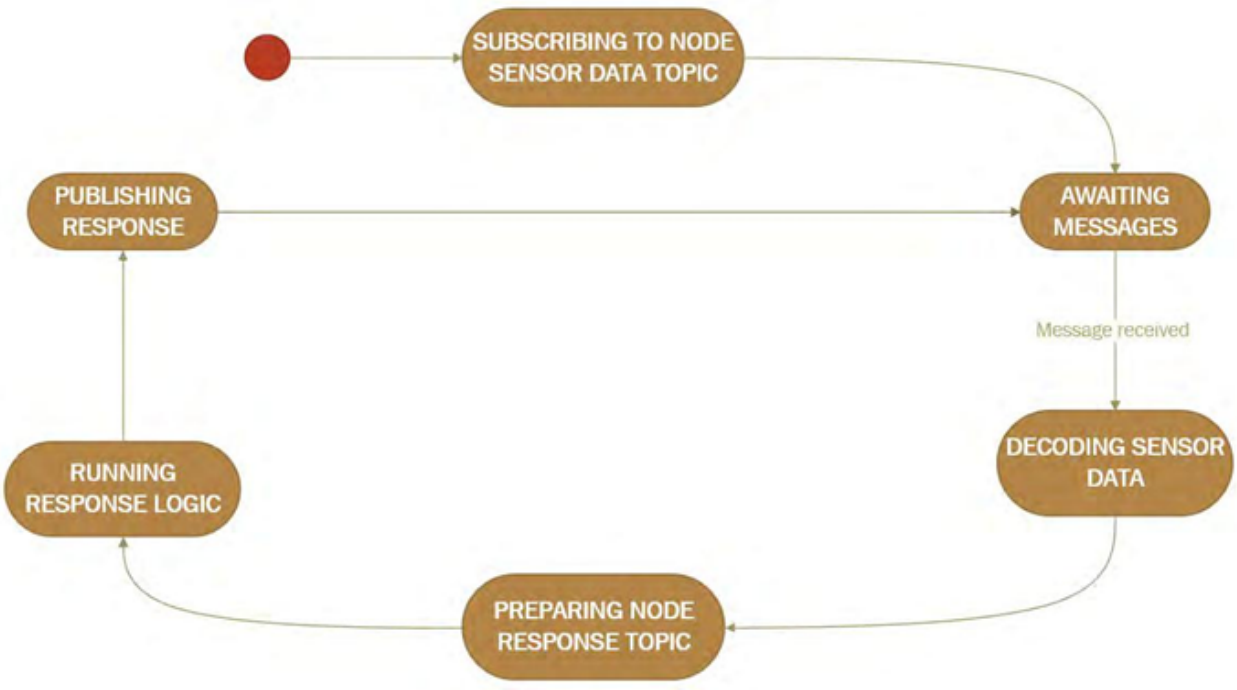

Рисунок 3.6 - Діаграма кінцевих станів локального клієнта

3.3.5 Передача відповіді клієнта MQTT на датчик

Вузол

Якщо припустити, що клієнт хоче надіслати відповідь сенсорному вузлу через шлюз. Щоб це сталося, шлюз повинен підписатися на тему відповіді вузла створеної клієнтом MQTT. Отже, шлюз отримує публікації від клієнта MQTT яку необхідно доставити до сенсорного вузла. Однак, як згадувалося раніше, тільки шлюз надсилає відповідь вузлу після отримання повідомлення запиту від цього вузла (див. рис. 3.3). В результаті шлюз повинен мати механізм, щоб утримувати відповідь вузла до самого шлюзу, отримуючи повідомлення запиту від вузла, і тільки потім передає його. Це показано на рисунку 3.5.

3.3.6 Клієнт локального контролера

Локальний контролер — це локальний клієнт MQTT, який простим способом реалізується локально «точка прийняття рішення» в системі, яка надає команду на відкриття або закриття клапана води (інші точки прийняття рішень можуть бути розташовані в мережі Інтернет). Контролер отримує дані від датчиків вузлів через брокер MQTT із шлюзу ZigBee/MQTT, а потім відповідає дією для керування потоком води вузла. Для досягнення цього завдання контролер використовує наступні класи:

- mqtt.client;
- mqtt.subscriber, який містить функції, що дозволяють здійснювати пряму підписку та обробку повідомлень, і буде використовуватися для підписки клієнта на теми MQTT і реєструвати функцію зворотного виклику.

Контролер використовує бібліотеку, яка зберігає інформацію кожного вузла щодо прийняття актуальних рішень, при цьому кожен вузол ідентифікується своєю MAC-адресою. Ця інформація про вузол включає інформацію про GPS, пріоритет, стан клапана, тип культури і т. д. Для обсягу цієї реалізації та тестування створюється бібліотека у спрощеному форматі та заповнюється прикладними значеннями.

Насправді її можна створити на основі фактичної інформації від польових розробників і фермерів, а потім зчитувати з бази даних.

Для обробки всіх повідомлень від брокера MQTT визначено функцію зворотного виклику. У функції контролер використовує повідомлення теми методу, який повертає в тему повідомлення у вигляді рядка, а потім витягує адресу рядку теми вузла за допомогою індексації рядків.

Після того, як адреса вузла витягується, контролер перевіряє якщо вузол відомий, тобто якщо він існує в бібліотеці вузлів. Якщо вузол існує в бібліотеці, значення стану клапана вузла, яке вказує на дію вузла («O» для відкриття та «C» для закриття) завантажується із бібліотеки, а дія вузла публікується в брокері MQTT. Щоб виконати публікацію, контролер спочатку підключає об'єкт клієнта MQTT локальний брокер MQTT за допомогою функції підключення, а потім готує тему дії вузла у цьому форматі дія/адреса вузла. Нарешті, дія для вузла публікується в брокері в створеній темі, використовуючи функцію публікації. Після публікації дії вузла контролер запускає логіку-заглушку, щоб оновити статус вузла в бібліотеці вузлів, просто змінивши його з «O» до «C», і навпаки. Якщо вузол не існує в бібліотеці вузлів, то контролер надсилає повідомлення про помилку, що вказує на те, що вузол невідомий контролеру.

Контролер використовує функцію підписки зворотного виклику, яка пропонує простий і зрозумілий спосіб підписки клієнта MQTT на набір тем і процесів вхідних повідомлень з використанням визначеної користувачем функції зворотного виклику.

Функція приймає три аргументи, в функції зворотного виклику onmessage, яка обробляє всі вхідні повідомлення. Другим аргументом, який приймає функція, є набір тем, на які підписується клієнт. Досліджуваний клієнт підписується на всі теми про вологість вузлів від брокера MQTT. Нарешті, останнім аргументом є ім'я хоста MQTT, яке представляє брокер MQTT. Доцільно використовувати локально-запущеного брокера. Виклик функції зворотного виклику підписки блокується, це означає, що він зчитує і повертається до потоку програми лише тоді, коли клієнт отримує повідомлення від брокера MQTT. Це позбавляє необхідності встановлювати цикл, який зчитує брокер MQTT, поки клієнт не отримає інформацію.

Хоча функція пропонує простий спосіб підписки клієнта на набір тем і обробки вхідних повідомлень, він не може використовуватися програмою шлюзу, оскільки не було виклику блокування, в функції readPacket.

3.3.7 Віддалений клієнт зберігання бази даних

Сховище бази даних — це програма, яка просто отримує значення вологості та дії сенсорних вузлів і локальних контролерів відповідно через брокера MQTT, а потім переліку значення вузлів у базі даних. Це просто приклад того, як відносно легко можна додати нову логіку системи. Клієнт використовує такі класи:

- MongoDB;
- mqtt.client;
- mqtt.subscriber.

Клієнт використовує функцію зворотного виклику для обробки вхідних повідомлень MQTT. У функції спочатку перевіряють категорію повідомлення, чи воно має значення вологості, або ні. Для отримання цієї інформації використовується тема повідомлення.

Як було сказано раніше, теми є у формі /вологість/адреса вузла або дія/адреса вузла, і тому включає інформацію про адресу вузла і категорію кількості даних про вузол. Клієнт використовує повідомлення теми, функції вилучення повідомлення з рядку тем, а потім використовує індексування символів рядка для вилучення другого символу рядка, який є або "h" або "a", що вказує значення вологості та дії відповідно. Якщо категорією повідомлення є вологість, тоді значення вологості вузла тимчасово зберігається у пам’яті вологості вузла, яка була створена як глобальна змінна. Щоб зберегти значення вологості, клієнт спочатку витягує рядок теми вузла, а потім використовує повідомлення для функції отримання значення вологості вузла як двійкових даних. Потім двійкові дані перетворюються в рядок за допомогою функції декодування, яка потім нарешті зберігається як значення вологості вузла в бібліотеку вологості.

З іншого боку, якщо категорією повідомлення є дія, тобто вона містить дію для вузла, значення вологості вузла, то дії зберігаються в MongoDB. Щоб зберегти ці значення, клієнт готує рядок списку, який буде використовуватися пізніше під час виконання «вставки». Рядок списку використовує адресу вузла, тому клієнт витягує адресу вузла з рядка теми повідомлення. Після цього вузол завантажує значення вологості вузла з бібліотеки вологості вузлів. Потім значення дії вузла завантажується з корисного навантаження повідомлення як двійкові дані, а потім перетворені в рядок за допомогою функції декодування. Нарешті, вузол вологості і значення відповідної дії зберігаються в базі даних.

Як показано на рисунку 3.5, клієнт потім ініціалізує об’єкт клієнта MongoDB, підключений до локально-запущеної MongoDB. Після цього створюється база даних. Ця база даних є місцем, де фактична вологість вузла та значення дії зберігаються як у вигляді списку. Потім клієнт MQTT підписується на всі теми вологості вузла та дій, а потім реэструє функцію зворотного виклику.

3.4 Функціональні можливості системи

Шлюз - це двонаправлений комунікатор MQTT/ZigBee, який отримує повідомлення від датчика вузла за протоколом ZigBee, а потім публікує дані, що передаються повідомленнями, у MQTT сервер. Дані потім використовуються іншими клієнтами MQTT, які також можуть надсилати зворотні повідомлення до сенсорних вузлів через сервер MQTT до шлюзу.

3.4.1 Обробка звернень MQTT

Шлюз визначає функцію зворотного виклику для обробки отримання повідомлень MQTT від клієнтів MQTT. Повідомлення містять відповіді, які шлюз надсилає до сенсорних вузлів за запитом вузла. Функція приймає три аргументи, в клієнта наприклад, дані клієнта, і фактично отримане повідомлення, яке є зразком з

MQTT Message. Повідомлення MQTT складається з таких атрибутів, як повідомлення topic, тіло повідомлення qos, тощо. Потім функція витягує адресу вузла, до якого спрямована дія, використовуючи рядок теми повідомлення, і зберігає повідомлення в попередньо ініціалізованій бібліотеці вузлів. Теми MQTT мають ієрархічну структуру, подібну до папок та файли у файловій системі, використовуючи косу риску як роздільник. Шлюз спирається на цю ієрархічну структуру з використанням адреси вузла та категорії даних для формування шляху до теми.

Адреса вузла визначає вузол датчика, якому належать дані, і передбачає категорії даних, що один вузол може мати більше однієї категорії даних. Такий підхід спрощує процес підписки на всі теми вузлів під тією ж категорією. Для простоти та збереження загальних налаштувань цього дослідження реалізація шлюзу знаходиться в контексті розумного зрошення. У результаті теми у шлюзі знаходяться в формі /вологість/адреса вузла і /дія/адреса вузла, які вказують на вологість вузлів і теми дій, відповідно.

Щоб витягти адресу вузла, шлюз спочатку витягує тему повідомлення, в якій повідомлення було опубліковано, використовуючи повідомлення, функцію теми, яка повертає тему у вигляді рядка. Після вилучення адреси вузла функція перевіряє, чи відомий вузол шлюзу, тобто чи вузол знаходиться в бібліотеці вузлів. Якщо вузол відомий, то функція завантажує відповідь вузла в бібліотеку.

Корисне навантаження повідомлення MQTT – це двійкові дані, які бібліотека зберігає у вигляді рядка. В результаті функція перетворює відповідь в рядок, використовуючи функцію декодування, а потім завантажує її в бібліотеку для передачі до вузла за запитом. Якщо вузол не існує в бібліотеці вузлів, то це означає що вузол не знаходиться в мережі ZigBee, що стосується шлюзу. Зазвичай ця ситуація може виникнути через те, що цей вузол ще не передавався після перезапуску шлюзу. Ситуація нормалізується протягом періоду часу, ідентичного циклу передачі сенсорного вузла. У гіршому випадку все одно це аномалія, тому вона реєструється. Потім шлюз ініціалізує об’єкти імпортованих класів, починаючи з MQTT клієнта об'єкт за допомогою функції клієнта. Слідом за цим шлюз підключає клієнта до локального брокера MQTT, у даному випадку працює на Raspberry Pi,

використовуючи функцію підключення. Потім шлюз використовує ієрархічну структуру тем для підписки теми відповідей клієнта на вузли.

Оскільки шлюз реалізовується в контексті розумного зрошення, де локальний контролер надсилає повідомлення про відповідь вузла через шлюз, то для цілей тестування шлюз підписує клієнта MQTT на всі теми дій вузлів. Після підписки клієнта на тему, визначена функція обробника пов’язана з клієнтом onmessage зворотнім зверненням, щоб визначена функція обробила кожне отримане повідомлення MQTT.

3.4.2 Читання повідомлень ZigBee

Переходячи до частини ZigBee, шлюз ініціалізує локальний об’єкт XBee на AMA послідовний порт Raspberry Pi і відкриває підключення пристрою. Як зазначалося раніше, шлюз отримує два типи повідомлень від сенсорного вузола, публікацію або запит на відповідь. Публікація містить дані датчиків від сенсорних вузлів, які шлюз потім публікує у відповідну тему в брокера MQTT. З іншого боку, повідомлення-запит має отримати відповідь. Отже, шлюз надсилає на вузол датчика дію, яку він отримав від клієнта MQTT, який у нашому випадку діє як локальний контролер.

Шлюз використовує функцію read_data читання вхідних повідомлень на локальному пристрої XBee. Виклик функції блокує, тобто повертає до логіки програми, коли вузол отримує повідомлення від шлюзу або досягає часу очікування. У даному випадку шлюз встановлює час очікування на 2 години, що означає, що він очікує повідомлення протягом 2 годин. Для інших умов можуть знадобитися інші часові проміжки. Функція повертає XBeeMessage, який серед багатьох атрибутів має Remote XBee Device, який містить інформацію про вузол надісланого повідомлення та корисне навантаження повідомлення у вигляді масиву байтів. Потім шлюз використовує віддалений пристрій get_64_bitaddr, функцію вилучення адреси віддаленого пристрою XBee. Функція tion повертає адресу локального пристрою відправника як властивість об'єкта XBee64BitAddress. Об’єкт XBee64BitAddress

містить адресу вузла в масиві байтів, який не можна використовувати з серверу MQTT. В результаті шлюз використовує функцію в класі banscii, до перетворення адреси вузла з масиву байтів у шістнадцятковий вигляд, який є рядком байтів, і декодується в рядок, який можна використовувати на сервері MQTT за допомогою функції декодування.

Після того, як адреса відправника витягується у вигляді рядка, шлюз перевіряє, чи відомий вузол, якщо він існує в бібліотеці вузлів. Якщо вузол не відомий, то просто додається enod до бібліотеки, як обговорюється в пункті про архітектуру. Після цього шлюз перевіряє тип отриманого повідомлення, яке є опублікованим, або запит на дію.

Розрізняючи публікацію та відповідь запиту повідомлення, встановлюється перший байт корисного навантаження повідомлення від сенсорних вузлів на «0» і «1» відповідно. В результаті шлюз перевіряє перший байт масиву даних корисного навантаження повідомлення. Якщо перший байт не є ні нулем, ні 1, то повідомлення не підлягає публікації, воно читається та відхиляється шлюзом. Якщо перший байт дорівнює нулю, повідомлення є публікацією, шлюз опублікує файл даних датчика, які повідомленням передає брокеру MQTT. Опублікувавши дані датчика, шлюз спочатку готує тему даних, під якою буде повідомлення опубліковано у формі вологість/адреса вузла. Після цього витягуються байти даних датчика вузла (які в даному контексті є вологістю) масив байтів корисного навантаження повідомлення XBee з використанням індексування символів. Байти потім перетворюються на цілі значення, які є фактичним значенням вологості. Після цього підключається клієнт MQTT до локального брокера MQTT, що працює на Raspberry Pi, а потім значення вологості вузла публікується як рядок у підготовленій темі. З іншого боку, якщо перший байт є "1", повідомлення є запитом на дію, шлюз отримує відповідь вузла (це дію на клапан, який може бути відкритим або закритим у цьому контексті) з бібліотеки дій вузла. Потім дія надсилається на віддалений XBee вузол, який представлений віддаленим пристроєм, завантажений з теми повідомлення раніше, коли повідомлення було прочитано. Однак, якщо в бібліотеці немає дії вузла, то знову ж таки, шлюз видає повідомлення про помилку, що вказує на те, що вузла немає в бібліотеці, як обговорювалося раніше. Основний цикл повторюється доки програма завершується.

3.4.3 Інтелектуальність системи

Важливо зазначити, що в даному розділі лише реалізовано і проаналізовано функціональність шлюзу, і те наскільки легко вузли датчиків можуть передавати дані через шлюз без використання будь-яких складних програм, як це було у випадку протоколу MQTT-SN. Крім того, немає ні впровадження, ні тестування інтелекту системи, і відповідно до сфери застосування. Через роз'єднання компонентів, можна легко додати клієнта, який підпишеться на датчик теми вузлів у брокері MQTT, а потім використовувати штучний інтелект, машинне навчання та прогнозування погоди для покращення прийняття рішень щодо контролю витрати води в датчику вузла. Потім клієнт може опублікувати цю інформацію в брокері MQTT, де вона може бути легко використана шлюзом.

Враховуючи особливості функціоналу розроблюваної системи, та беручи до уваги результати досліджень за поточною темою інших авторів [17; 18; 19], можна зробити висновок, що багатокомпонентна інтелектуальна система зрошення є актуальним напрямком технічних впроваджень у сільськогосподарському секторі. Застосування такої системи дає змогу зекономити близько 16% затрачених водних ресурсів, у порівнянні з класичними системами поливу. Цей показник можна покращити, шляхом налагодження роботи системи у реальних польових умовах.

Висновки до розділу 3

У цьому розділі йдеться мова про архітектуру легкого, спрощеного шлюзу, який публікує і запитує дані до та від клієнтів MQTT від імені сенсорних вузлів, працюючи у мережі ZigBee. Шлюз був розроблений таким чином, що мережевий компонент ZigBee імітує MQTT шаблон видавець/користувач, надсилаючи окремі повідомлення для публікації та запиту даних.

Також, у цьому розділі розповідається про реалізацію шлюзу з використанням додаткових компонентів щоб проілюструвати функціональність шлюзу в контексті розумної системи зрошення у відносно простий спосіб. У цьому розділі досить зпрощено реалізовано розумну систему поливу, усунувши більш складні компоненти запропонованої системи у розділі 2.

Шлюз реалізує комунікатор ZigBee/MQTT, який з'єднує вузли датчиків ZigBee у стандартний MQTT простим способом. Архітектура шлюзу включає вузли датчиків для надсилання двох різних повідомлень для публікації та запиту даних до та від шлюзу, імітуючи шаблон MQTT видавець/абонент. Ця архітектура повністю роз'єднує сенсорні вузли від клієнтів MQTT, що дозволяє використовувати всі різні компоненти системи, діяти незалежно один від одного. Шлюз реалізований як протокол без стану, що дозволяє різним компонентам легко підключатися та повторно підключатися без процедури додаткового введення. Шлюз оптимізує інтерфейс MQTT до стандартного MQTT, який дозволяє додавати компоненти в повну систему досить просто. Компоненти це прості клієнти MQTT, які отримують дані датчиків від шлюзу через брокера MQTT, і можуть керувати сенсорними вузлами через шлюз. Спочатку було припущення, що буде раціональною ідеєю реалізувати мета-тему, яка передає мета-інформацію про різні вузли мережі ZigBee, як зазначалося раніше. Однак пізніше стало зрозуміло, що це не є основним у функціональності шлюзу. В результаті такої реалізації не було тем у шлюзі.

ВИСНОВКИ

У ході виконання роботи початковим етапом був аналіз матеріалів та наукових публікацій за напрямком дослідження. Викладено масштаб роботи, методологію та структуру дослідження. Далі вивчено особливості 4IR і дві її технологічні основи, які можна використовувати для створення розумних система зрошення, а саме Інтернет речей та штучний інтелект. Також повторно переглянуто існуючі подібні роботи, по-перше, проекти розумного зрошення з використанням компонентів IoT, а по-друге проекти, що досліджують впровадження систем IoT у посушливі регіони, та загальну проблематику створення систем IoT для користувачів, які вперше зтикаються з такими цілями.

Наступним етапом був аналіз проблеми за класичною схемою «зверху вниз», виконуючи серію сценаріїв. Подібний аналіз дозволив розібратися в проблемі, щоб визначити основні компоненти, які знадобляться для побудови подібної системи.

Рішення було розроблене як ряд сенсорних вузлів, розміщених на полі, з можливістю вимірювання вологості ґрунту та передачею значення вологості центральному «супервузлу». Далі, супервузол керує кожним датчиком витрати води вузла, передаючи відповідну команду спрацьовування назад до вузла.

Було необхідним, щоб системні вузли могли спілкуватися з супервузлом якомога простішим і надійнішим шляхом. В результаті, вівся пошук рішення для бездротового зв’язку який пропонує можливості мережі, а також автоматичне виявлення маршрутів і розширені можливості самовідновлення. Також потрібно було отримати протокол передачі даних, який відокремлює сенсорні вузли від супервузла, щоб усунути залежності між вузлами та дозволити їм діяти незалежно один від одного, тобто надсилати повідомлення незалежно від того, активний інший вузол чи ні. Тому, було обрано протокол зв'язку ZigBee і протокол передачі даних MQTT. Виходячи з цього, досліджувались готові апаратні та програмні компоненти, які підходять для реалізації визначених загальних елементів системи. Вирішено скористатися сімейство мікроконтролерів Arduino з відкритим вихідним кодом і мікрокомп'ютером Raspberry Pi. Також було обрано комерційно доступний датчик

вологості ґрунту серії MH та електроклапан як виконавчий компонент для управління потоком води. Досліджувалось сімейство RF модулів XBee, які реалізують протокол ZigBee, а також відкритий вихідний код проекту Mosquitto MQTT, який реалізує брокер MQTT і бібліотеку клієнтів MQTT.

Було знайдено більшість інструментів з великою кількістю підтримки, доступної в мережі Інтернет. Було визначено, що протокол MQTT вимагає складного рівня передачі, такого як TCP в мережу IP, і тому MQTT не можна було використовувати з сенсорними вузлами, які підтримують лише протокол ZigBee. В результаті вирішено використовувати комбінацію MQTT, MQTT-SN, який був спеціально розроблений для підтримки ZigBee і UDP, зберігаючи властивості MQTT. Проаналізовано бібліотеку MQTT-SN, доступну на GitHub. Бібліотека реалізує Arduino MQTT-SN клієнт через протокол ZigBee і шлюз MQTT-SN Linux. Потім проводились дослідження цієї бібліотеки, і було зроблено висновок, що MQTT-SN недостатньо підходить для цільового користувача.

За результатами проведеної роботи можна зробити висновок, що створення багатоелементної інтелектуальної системи зрошення має ряд складнощів, які пов’язані з реалізацією взаємодії елементів системи, зокрема, у питанні обміну інформацією. У той же час, впровадження інтелектуальної системи зрошення, подібної до тої, яка розглядається у даній роботі дає змогу знизити надмірну витрату водних запасів, та підвищити об’єми та якість урожаю. Насамперед це досягається за рахунок виконання зрошувальних робіт на окремих ділянка за принципом пріоритетності.

Враховуючи особливості функціоналу розроблюваної системи, та беручи до уваги результати досліджень за поточною темою інших авторів [17; 18; 19], можна зробити висновок, що багатокомпонентна інтелектуальна система зрошення є актуальним напрямком технічних впроваджень у сільськогосподарському секторі. Застосування такої системи дає змогу зекономити близько 16% затрачених водних ресурсів, у порівнянні з класичними системами поливу. Цей показник можна покращити, шляхом налагодження роботи системи у реальних польових умовах.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

4IR - четверта промислова революція

IoT - Інтернет речей

3D -поняття тривимірності

ШІ – штучний інтелект

МН – машинне навчання

PIR - пасивний інфрачервоний датчик

АЦП – аналогово-цифровий перетворювач

CPU - центральний процесор

ROM - постійна пам'ять

RAM - оперативна пам'ять

MSB – найбільш важливі байти

LSB – найменш важливі байти

ШІМ – широтно-імпульсна модуляція

СПИСОК ЛІТЕРАТУРИ

1. Klaus Schwab. The Fourth Industrial Revolution: What it Means, How to Respond. On-line. [Last Accessed: May2020]. url: https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/.

2. Jehoon Sung. \The Fourth Industrial Revolution and Precision Agriculture". In: Automation in Agriculture: Securing Food Supplies for Future Generations. BoD{Books on Demand, 2018, pp. 3{15.

3. Guoping Li, Yun Hou, and Aizhi Wu. \Fourth Industrial Revolution: Technological Drivers, Impacts and Coping Methods". In: Chinese Geographical Science 27.4 (2017), pp. 626{637.

4. Min Xu, Jeanne M David, and Suk Hi Kim. \The Fourth Industrial Revolution: Opportunities and Challenges". In: International Journal of Financial Research 9.2 (2018), pp. 90{95.

5. Feng Xia et al. \Internet of Things". In: International Journal of Communication Systems 25.9 (2012), pp. 101{1102.

6. Ovidiu Vermesan et al. \Internet of Things Strategic Research Roadmap". In: Internet of Things- Global Technological and Societal Trends 1.2011 (2011), pp. 9{52.

7. Jayavardhana Gubbi et al. \Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions". In: Future Generation Computer Systems 29.7 (2013), pp. 1645{ 1660.

8. Oracle South Africa. What is the Internet of Things? Online. [Last Accessed: Jan 2019]. url: https://www.oracle.com/za/internet-of-things/what-is-iot.html.

9. Klaus Schwab. The Fourth Industrial Revolution. USA: Crown Publishing Group, 2017.

10. Desmond Tutu Ayentimi and John Burgess. \Is the Fourth Industrial Revolution Relevant to sub-Sahara Africa?" In: Technology Analysis & Strategic Management 31.6 (2019), pp. 641{652.

11. James E Addicott. \The Precision Farming Revolution". In: The Precision Farming Revolution. Springer, 2020, pp. 1{35.

12. Ilaria Zambon et al. \Revolution 4.0: Industry vs. Agriculture in a Future Development for SMEs". In: Processes 7.1 (2019), p. 36.

13. DD Dasig. \Implementing IoT and Wireless Sensor Networks for Precision Agriculture". In: Internet of Things and Analytics for Agriculture, Volume 2. Springer, 2020, pp. 23{ 44.

14. Victoria Namirimu. \User Requirements for Internet of Things (IoT) Applications: An Observational study". MA thesis. Sweden: Blekinge Institute of Technology, 2015.

15. Soa Amador Nelke and Michael Winokur. \Introducing IoT Subjects to an Existing Curriculum. An Ongoing Experience at the Faculty of the Technology Management - HIT". In: Cyber Physical Systems. Model-Based Design. Springer, 2018, pp. 193{196.

16. Jorge Guerra Guerra and Armando Fermin Perez. \Alignment of Undergraduate Curriculum for Learning IoT in a Computer Science Faculty". In: Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education. 2017, pp. 362{362.

17. Levent Sey, Ertan Akman, and Tugrul C Topak. \Smart Irrigation System". In: World Academy of Science, Engineering and Technology, International Journal of Electrical and Computer Engineering 2 (2015).

18. Ahmed Imteaj et al. \IoT Based Autonomous Percipient Irrigation System using Raspberry Pi". In: 2016 19th International Conference on Computer and Information Technology (ICCIT). IEEE. 2016, pp. 563{568.

19. Olugbenga Kayode Ogidan, Abiodun Emmanuel Onile, and Oluwabukola Grace Adegboro. \Smart Irrigation System: A Water Management Procedure". In: Agricultural Sciences 10.01 (2019), pp. 25{31.

20. Foughali Karim, Karim Fathallah, and Ali Frihida. \Monitoring System Using Web of Things in Precision Agriculture". In: Procedia Computer Science 110 (2017), pp. 402{ 409.

21. Meshlium -The IoT Gateway-Waspmote Technical Guide. Online. [Last Accessed: January 2021]. url: https://development.libelium.com/waspmote-technical-guide /meshlium.

22. Chandan Kumar Sahu and Pramitee Behera. \A Low Cost Smart Irrigation Control System". In: 2015 2nd International Conference on Electronics and Communication Systems (ICECS). IEEE. 2015, pp. 1146{1152.

23. S Darshna et al. \Smart Irrigation System". In: IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) 10.3 (2015), pp. 32{36.

24. John E Walsh. \Soil moisture sensor". US Patent 4,540,936. 1985. url: https://patents.google.com/patent/US4540936.

25. Sharmila Nath, Jayanta kumar Nath, and Kanak Chandra Sarma. \IoT Based System for Continuous Measurement and Monitoring of Temperature, Soil Moisture and Relative Humidity". In: Technology 9.3 (2018), pp. 106{113.

26. Soil Moisture Sensor { How to choose and use with Arduino. Online. [Last accessed:March 2020]. 2019. url: https://www.seeedstudio.com/blog/2020/01/10/what-is - soil-moisture-sensor-and-simple-arduino-tutorial-to-get-started/.

27. P Aravind et al. \A Wireless Multi-Sensor System for Soil Moisture Measurement". In: 2015 IEEE SENSORS. IEEE. 2015, pp. 1{4.

28. Roger T Koide et al. \Plant Water Status, Hydraulic Resistance and Capacitance". In: Plant Physiological Ecology. Springer, 1989, pp. 161{183.

29. Ammar Rayes and Samer Salam. \The Things in IoT: Sensors and Actuators". In: Internet of Things From Hype to Reality. Springer, 2017, pp. 57{77.

30. Ala Al-Fuqaha et al. \Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications". In: IEEE Communications Surveys & Tutorials 17.4 (2015), pp. 2347{2376.

31. Warren Snyder and Monte Mar. \Programmable Microcontroller Architecture (Mixed Analog/Digital)". US Patent 6,724,220. 2004. url: https://patents.google.com/patent/US7221187.

32. James O Mergard et al. Flexible Microcontroller Architecture. US Patent 6,415,348. 2002. url: https://patents.google.com/patent/US6415348B1/en?q= Flexible+Microcon troller+Architecture&oq=Flexible+Microcontroller+Architecture.

33. RASS Bannatyne and Greg Viot. \Introduction to Microcontrollers". In: WESCON/97 Conference Proceedings. IEEE. 1997, pp. 564{574.

34. Korbin S Van Dyke, Paul Campbell, and Don Alan Van Dyke. Computer for Execution of RISC and CISC Instruction Sets. US Patent 7,047,394. 2006. url: https://patents.google.com/patent/US7047394B1/en.

35. Tariq Jamil. \RISC versus CISC". In: IEEE Potentials 14.3 (1995), pp. 13{16.

36. Liberios Vokorokos et al. \A Multicore Architecture Focused on Accelerating Computer Vision Computations". In: Acta Polytechnica Hungarica 10.5 (2013), pp. 29{43.

37. Dogan Ibrahim. PIC32 Microcontrollers and the Digilent chipKIT: Introductory to Advanced Projects. Oxford: Newnes, 2015.

38. Ala Al-Fuqaha et al. \Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications". In: IEEE Communications Surveys & Tutorials 17.4 (2015), pp. 2347{2376.

39. Sethi Pallavi and Smruti R Sarangi. \Internet of Things: Architectures, Protocols and Applications". In: Journal of Electrical and Computer Engineering 2017.1 (2017), pp. 1{ 25.

40. Burak H C orak et al. \Comparative Analysis of IoT Communication Protocols". In: 2018 International Symposium on Networks, Computers and Communications (ISNCC). IEEE. 2018, pp. 1{6.

41. Nisha Ashok Somani and Yask Patel. \Zigbee: A Low Power Wireless Technology for Industrial Applications". In: International Journal of Control Theory and Computer Modelling (IJCTCM) 2.3 (2012), pp. 27{33.

42. Bernstein Corinne. MQTT (MQ Telemetry Transport). Online. [Last Accessed: June 2020]. url: https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport.

43. What is Publish/Subscribe? Is it Useful for Me. Online. [Last accessed 15-January-2020]. url: https://vernemq.com/intro/mqtt-primer/publish-subscribe.html#publish-subscribe.

44. Suprabha Jadhav and Shailesh Hambarde. \Android Based Automated Irrigation System using Raspberry Pi". In: International Journal of Science and Research 5.6 (2016), pp. 2345{51.

45. S Darshna et al. \Smart Irrigation System". In: IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) 10.3 (2015), pp. 32{36.

46. Alessandro D'Ausilio. \Arduino: A Low-Cost Multipurpose Lab Equipment". In: Behavior Research Methods 44.2 (2012), pp. 305{313.

47. Nikola Zlatanov. \Arduino and Open Source Computer Hardware and Software". In: International Research Journal of Engineering and Technology (IRJET) 4 (2017).

48. Arduino.cc. Getting Started:Introduction. Online. [Last Accessed: February 2019]. 2019. url: https://www.arduino.cc/en/Guide/Introduction.

49. Mirjana Maksimovic et al. \Raspberry Pi as Internet of Things Hardware: Performances and Constraints". In: Design Issues 3.8 (2014).

50. Wei-Meng Lee and Clarence Chng. Introduction to IoT Using the Raspberry Pi. Online. [Last Accessed: October 2019]. 2020. url: https://www.codemag.com/Article/1607071/Introduction-to-IoT-Using-the-Raspberry-Pi.

51. Bernadette Johnson. How the Raspberry Pi Works. Online. [Last Accessed: October 2019]. 2019. url: https://computer.howstuffworks.com/raspberry-pi2.htm.

52. Digi XBee ZigBee. Online. [Last Accessed: February 2019]. url: https://www.digi.com/products/embedded-systems/digi-xbee/rf-modules/2-4-ghz-modules/xbeezigbee.

53. Robert Faludi. Building wireless sensor networks: with ZigBee, XBee, arduino, and processing. O'Reilly Media, Inc., 2010.

54. Eclipse Foundation. Eclipse Mosquitto. Online. [Last Accessed: February 2019]. url: https://mosquitto.org/.

55. Andy Stanford-Clark and Hong Linh Truong. \MQTT For Sensor Networks (MQTT-SN) Protocol Speci cation Version 1.2". In: International business machines (IBM) Corporation 1.2 (2013), pp. 1{27.