КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інформаційних систем та технологій

Спеціальність 126 – Інформаційні системи та технології
Освітня програма «Програмні технології інтернет речей»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

«Модель системи IoT рішень для процесу сироваріння»

Здобувач освіти на освітньому рівні «Магістр» 2-го курсу групи ІРма-21:

Руслан КУЧЕРЕНКО

(Власне Ім'я, ПРІЗВИЩЕ)

(підпис студента)

Науковий керівник:

к.т.н., доцент

(науковий ступінь, вчене звання)

Ольга КРАВЧЕНКО

(Власне Ім'я, ПРІЗВИЩЕ)

(дата) (підпис)

Попередній захист:

До захисту в Екзаменаційній комісії

(Висновок: "До захисту в Екзаменаційній комісії")

В.о. завідувача кафедри
Інформаційних систем та технологій

д.т.н., доцент

(науковий ступінь, вчене звання)

Олексій КОЛЕСНІКОВ

(Власне Ім'я, ПРІЗВИЩЕ)

(підпис)

(дата)

Київ 2021

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційні системи та технології
Освітній рівень Магістр
Спеціальність 126 Інформаційні системи та технології
Освітня програма Програмні технології інтернет речей

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри,
д.т.н., доцент
Олексій КОЛЕСНІКОВ
_____
«__» _____ 2021 року

ЗАВДАННЯ
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ МАГІСТРА

Здобувач освіти: Руслан КУЧЕРЕНКО

Група: ІРма-21

1. **Тема кваліфікаційна робота магістра**: «Модель системи IoT рішень для процесу сироваріння».
Затверджена протоколом засідання кафедри ІСТ №16/20 від «09» листопада 2020 р.
2. **Строк подання студентом готової роботи** – «20» травня 2021 р.
3. **Цільова установка та вихідні дані до роботи**: дослідження в області рішення кліматичного-контролю камери зберігання сирів. Програмно-апаратні рішення для оптимізації процесів в галузі виробництва сиру. Дані стану навколишнього середовища з давачів, методи їх контролю та передачі даних з використанням систем IoT.
4. Зміст роботи: РОЗДІЛ 1 АНАЛІЗ СУЧАСНОГО СТАНУ ІОТ РІШЕНЬ ДЛЯ МОДЕЛІ СИСТЕМИ ПРОЦЕСУ СИРОВАРІННЯ (аналіз аналогів, метод автоматичного управління технологічним процесом сироваріння); РОЗДІЛ 2 РОЗРОБКА ПРОЕКТУ ІОТ РІШЕННЯ ТА МЕТОДІВ ОБРОБКИ ДАНИХ В ІОТ (Розробка проекту IoT рішення, методи та засоби обробки даних, використання хмарних технологій для реалізації IoT рішень); РОЗДІЛ 3 РЕАЛІЗАЦІЯ АРХІТЕКТУРИ ІОТ РІШЕННЯ (архітектура проекту IoT рішення, комунікаційні технології та системи, аналіз захищеності інформації та методів і способів (протоколів) для її кіберзахисту); РОЗДІЛ 4 ОПИС РОЗРОБКИ ПРОГРАМНОГО

(програмування мікроконтролера Arduino, розробка програмного забезпечення та графічного інтерфейсу, аналіз результатів досліджень).

5. **Перелік графічного матеріалу**: Концептуальна модель архітектури системи IoT рішень для процесу сироваріння; алгоритм роботи системи та структурна схема; таблиця відповідність етапів виробництва сиру, показнику температури/вологості середовища та необхідних давачів; блок схема роботи програмного забезпечення; графічний інтерфейс системи для ОС Windows; зображення прототипу системи.

6. **Календарний план виконання роботи:**

Етапи виконання дипломних робіт	**Термін виконання**	**Результат виконання**
1. Вибір тематики кваліфікаційної роботи магістра	20.10.2019	виконано
2. Наказ про затвердження тем кваліфікаційної роботи магістра та призначення наукових керівників	09.11.2020	виконано
3. Формування переліку нормативних матеріалів, літератури з проблематики кваліфікаційної роботи магістра роботи	10.12.2020	виконано
4. Розробка плану кваліфікаційної роботи магістра і його погодження з науковим керівником	24.12.2020	виконано
5. Написання I розділу дипломної роботи	20.01.2021	виконано
6. Написання II розділу дипломної роботи	19.02.2021	виконано
7. Написання III розділу дипломної роботи	05.03.2021	виконано
8. Написання IV розділу дипломної роботи	19.03.2021	виконано
9. Підготовка висновків і пропозицій	05.04.2021	виконано
9. Попередній захист дипломної роботи	20.04.2021	виконано
10. Рецензування кваліфікаційної роботи магістра	до 20.05.2021	виконано
11. Захист кваліфікаційної роботи магістра	26.05.2021	

Дата видачі завдання «_10_» _листопада_ 2020 р.

Керівник роботи: к.т.н., доц. Ольга КРАВЧЕНКО _____ (підпис)

Завдання прийняв до виконання:

Здобувач освіти на освітньому рівні «Магістр» 2-го курсу групи ІРма-21

Руслан КУЧЕРЕНКО _____

(Власне Ім’я, ПРІЗВИЩЕ) (підпис)

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційних систем та технологій

Освітня програма «Програмні технології інтернет речей»

Кваліфікаційна робота магістра Руслана КУЧЕРЕНКА

Тема роботи: «Модель системи IoT рішень для процесу сироваріння».

Мета кваліфікаційної роботи магістра – розробка методу автоматичного управління технологічним процесом сироваріння засобами IoT, проектування і програмна реалізація на його основі високоефективної системи IoT рішень для процесу сироваріння з підтримкою автоматизованого управління процесами обігріву, охолодження, зволоження, осушування, вентиляції, освітлення та мінімальним використанням енергоресурсів мікроконтролера.

Об'єкт дослідження – система IoT рішень для процесів сироваріння.

Предмет дослідження – обробка даних стану навколишнього середовища з давачів, методи їх контролю та передачі даних з використанням систем IoT.

Методи дослідження – використано теоретичні та емпіричні методи дослідження: аналіз та узагальнення наукової літератури, розробка програмного коду для мікроконтролера в середовищі Arduino IDE, проектування програмного забезпечення та графічного інтерфейсу користувача в середовищі Qt Creator, верифікація даних за структурою SWOT аналізу.

Практичне значення одержаних результатів. Дана кваліфікаційна робота магістра має прикладне значення – розробка прототипу системи IoT рішень для процесів сироваріння. Запропоновано концептуальну модель архітектури системи, розроблено схемотехнічне рішення, спроектовано та

реалізовано прототип системи. Одержані результати рекомендовано впровадити на виробництво.

Апробація результатів. Основні положення і результати досліджень, викладені у проекті, пройшли апробацію на X міжнародній науково-практичній конференції «Topical issues of the development of modern science» (Софія, Болгарія 2020) [1], на VII інтернаціональній конференції «Information Technology and Interactions» (Satellite) (Київ, 2020) [2] та на сімнадцятій міжнародній науково-технічній конференції «Проблеми інформатизації» у 2021 році [3], опубліковані у журналі «Sciences of Europe» № 51 за 2020 р. [4].

Кваліфікаційна робота магістра складається зі змісту, вступу, основної частини, яка включає чотири розділи, висновків та списку використаних джерел. Всього 140 сторінок.

КЛЮЧОВІ СЛОВА: модель, архітектура IoT рішення, мікроконтроллер, програмний комплекс, схематичне рішення, SWOT-аналіз

ABSTRACT

TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV

Faculty of Information Technologies

Department of Information Systems and Technologies

Educational Program "Software Technologies of the Internet of Things"

Qualification work of master Ruslan KUCHERENKO.

Work topic: "Model of IoT solutions for cheese-making process".

The purpose of the master's qualification work is to develop a method of automatic control of the technological process of cheese-making by means of IoT, design and software implementation on its basis of highly efficient IoT solutions for the cheese-making process with the support of automated control of heating, cooling, humidification, dehumidification, ventilation, lighting and minimal.

The object of research is a system of IoT solutions for cheese-making processes.

The subject of research is the processing of environmental data from sensors, methods of their control and data transmission using IoT systems.

Research methods - used theoretical and empirical research methods: analysis and generalization of scientific literature, development of software code for the microcontroller in Arduino IDE, software design and graphical user interface in Qt Creator, data verification by SWOT analysis.

The practical significance of the obtained results. This qualification work of the master has an applied value - the development of a prototype system of IoT solutions for cheese-making processes. A conceptual model of the system architecture is proposed, a circuit design solution is developed, a prototype of the system is designed and implemented. It is recommended to implement the obtained results in production.

Approbation of results. The main provisions and research results presented in the project were tested at the X International Scientific and Practical Conference "Topical issues of the development of modern science" (Sofia, Bulgaria 2020) [1], at

the VII International Conference "Information Technology and Interactions" (Satellite) (Kyiv, 2020) [2] and at the seventeenth international scientific and technical conference "Problems of Informatization" in 2021 [3], published in the journal "Sciences of Europe" № 51 for 2020 [4].

The master's qualification work consists of the content, introduction, main part, which includes four sections, conclusions and a list of sources used. Total 140 pages.

KEY WORDS: model, IoT solution architecture, microcontroller, software package, schematic solution, SWOT-analysis

ЗМІСТ

ВСТУП

Актуальність теми. На сьогодні в Україні зростає популярність розвитку молочної промисловості. Дві основні причини – ефективний збір молока та збільшення прибутку для виробників, де певною мірою вплинули інформаційні технології. Поширення та безпека молока – це досить серйозне питання, оскільки воно стосується здоров'я 90% нашого суспільства. Споживання молока є однією з найосновніших потреб людини. Якість харчової продукції є головною умовою її конкурентоздатності. Висока цінність молока стосується продукту, одержаного в умовах суворого дотримання кліматичних та санітарно-гігієнічних норм.

Спеціалістами провідної компанії України маркетингових досліджень Pro-Consulting проведено аналіз стану кисломолочних продуктів. До уваги брали статистику виробництва, основних тенденцій розвитку ринку, імпорту, експорту та споживання молочної продукції. У 2017 році вироблено 811 млн. т. молока, у 2018 – 843 млн. т., а до 2024 року грошовий обсяг, за прогнозами, перевищить 703 млрд доларів. Дослідження показало, що ринок молочної продукції різноманітний і висококонкурентний, адже має приріст близько 5% на рік [5].

Цікаво, що сири займають близько 10% за об'ємом виготовлення серед молочної продукції. У всьому світі існує кілька сотень видів натурального сиру, що вважається одним із найголовніших корисних продуктів. Він складається з усіх необхідних людському організму речовин: білків, вуглеводів, мінеральних солей, вітамінів та жирів. У порівнянні за поживною цінністю сири суттєво перевищують м'ясні, рибні та мучні продукти. Найбільше ціниться в складі сирів кальцій. Оскільки існують екологічні проблеми, що впливають на руйнування кісток в людському організмі, їм як ніколи необхідно мати у своєму раціоні харчування солі кальцію. Особливої уваги це стосується районів України, що зазнали впливу радіації. Саме тому сир є необхідною та обов'язковою складовою у харчовому раціоні людини [6].

Одним із шляхів мінімізації проблеми несприятливого вибору та виснажливої людської роботи є використання інформаційних технологій. У багатьох молочних господарствах вже застосовується комп'ютерний контроль фізіологічних та санітарних параметрів, що призводить до підвищення продуктивності та усунення деяких енергозатратних операцій. Програмні технології інтернет речей (IoT) набирають нових обертів, і багато наявних знань вже є вражаючими. IoT системи відіграють важливу роль у нашому житті щодня, хоча вони можуть бути не обов'язково видимими. Стрімкий розвиток IoT рішень спонукає до більш детального вивчення меж їх впровадження: агрокультура, промисловість, енергетика, медицина, смарт-будинки. Особливо **актуальною**, на мою думку, є галузь промисловості, а саме система IoT рішень для процесу сироваріння. Адже необхідно вилучити можливість відхилення від норм та вимог під час виробничого процесу.

Сучасна література пропонує автоматизацію та впровадження IoT рішень у різних сферах. Особливої уваги приділяють напрямку Промислового Інтернету речей, де відбувається суттєве підвищення ефективності виробництва за рахунок автоматизації процесів шляхом використання систем вбудованих давачів та віддаленого контролю, а також виключення можливості фатальних наслідків через людський фактор. Розглядають успішні приклади реалізації систем Інтернету речей на виробництвах бритв для гоління, мотоциклів, переробці відходів, проектуванні розумних будинків та навіть цілих міст. При цьому відсутні програмно-апаратні рішення для оптимізації процесів саме в галузі виробництва сиру. Готові системи на ринку представляють поодинокі рішення кліматичного-контролю камери зберігання сирів, теплиць чи житлових приміщень. Вони мають цілий ряд недоліків в своїх характеристиках, основними з яких є відсутня реалізація програмно-апаратних технологій, низька потужність навантаження, ручне керування, недостатній функціонал, неможливість масштабування та висока ціна. Саме тому, задача розробки системи IoT рішень для процесу сироваріння є **актуальною**, а впровадження технологій IoT розглядаються як один з найкращих способів вирішити дану задачу.

Мета і завдання дослідження. Метою кваліфікаційної роботи магістра є методу автоматичного управління технологічним процесом сироваріння засобами IoT та проектування і програмна реалізація на його основі високоефективної системи IoT рішень для процесу сироваріння з підтримкою автоматизованого управління процесами обігріву, охолодження, зволоження, осушування, вентиляції, освітлення та мінімальним використанням енергоресурсів мікроконтролера.

Основні завдання, що потрібно виконати для досягнення мети:

1. Аналіз базових принципів роботи IoT систем.
2. Аналіз технологічних стадій виготовлення сиру.
3. Аналіз останніх наукових досліджень та практичних розробок IoT систем.
4. Аналіз існуючих систем IoT рішень для процесу сироваріння.
5. Дослідження комунікаційних систем та технологій.
6. Проектування логічних зв’язків вузлів системи.
7. Розробка схеми структурної системи IoT рішень для процесу сироваріння.
8. Розробка концептуальної моделі архітектурного рішення відповідно до проекту системи IoT.
9. Розробка методу автоматичного управління технологічним процесом сироваріння.
10. Розробка програмного забезпечення.
11. Проведення верифікації даних та SWOT аналізу.

Об’єктом дослідження є система IoT рішень для процесів сироваріння.

Предметом дослідження є обробка даних стану навколишнього середовища з давачів, методи їх контролю та передачі даних з використанням систем IoT.

Методи дослідження. Під час вирішення основних завдань при розробці системи IoT рішень для процесів сироваріння використано теоретичні та емпіричні методи дослідження: аналіз та узагальнення наукової літератури,

розробка програмного коду для мікроконтролера в середовищі Arduino IDE, проектування програмного забезпечення та графічного інтерфейсу користувача в середовищі Qt Creator, верифікація даних за структурою SWOT аналізу.

Практичне значення одержаних результатів. Дана кваліфікаційна робота магістра має прикладне значення – розробка прототипу системи IoT рішень для процесів сироваріння. Запропоновано концептуальну модель архітектури системи, розроблено схемотехнічне рішення, спроектовано та реалізовано прототип системи. Одержані результати рекомендовано впровадити на виробництво.

Апробація результатів. Основні положення і результати досліджень, викладені у проекті, пройшли апробацію на X міжнародній науково-практичній конференції «Topical issues of the development of modern science» (Софія, Болгарія 2020) [1], на VII інтернаціональній конференції «Information Technology and Interactions» (Satellite) (Київ, 2020) [2] та на XVII міжнародній науково-технічній конференції «Проблеми інформатизації» у 2021 році [3], опубліковані у журналі «Sciences of Europe» № 51 за 2020 р. [4].

РОЗДІЛ 1 АНАЛІЗ СУЧАСНОГО СТАНУ ІОТ РІШЕНЬ ДЛЯ МОДЕЛІ СИСТЕМИ ПРОЦЕСУ СИРОВАРІННЯ

У цьому розділі сформульовано та описано задачу, яка буде вирішуватись в кваліфікаційній роботі магістра на основі проведеного аналізу сучасного стану IoT рішень. Для аналізу і оцінки стану об'єкта дослідження проведено декілька основних аналізів: аналіз базових принципів роботи Internet of Things, аналіз технологічних стадій виготовлення сиру, аналіз останніх наукових досліджень та практичних розробок, аналіз готових систем на ринку. Описано вхідну інформацію, яка необхідна для вирішення поставленої задачі.

1.1 Постановка задачі кваліфікаційної роботи магістра

Метою кваліфікаційної роботи магістра є розробка моделі системи IoT рішень для процесу сироваріння, що має задовольняти наступні вимоги:

- високоефективна система з мінімальним використанням енергоресурсів мікроконтролера;
- підтримка можливості майбутніх програмно-апаратних оновлень;
- модульність системи для зручної заміни частини, що вийшла з ладу;
- висока надійність системи;
- використання сучасних конструкторських підходів;
- пристосованість до експлуатації у відповідних кліматичних умовах;
- доступна для заміни елементна база;
- відповідність технічним характеристикам, зазначеним у технічному завданні.

Для досягнення кінцевої мети необхідно виконати наступні завдання:

- аналіз базових принципів роботи IoT систем;
- аналіз технологічних стадій виготовлення сиру;
- аналіз останніх наукових досліджень та практичних розробок IoT систем;
- аналіз існуючих систем IoT рішень для процесу сироваріння;
- описати логічні зв'язки вузлів системи на етапі проектування;

- навести структурну схему роботи системи IoT рішень для процесу сироваріння;
- розробити архітектуру відповідно до проекту системи IoT;
- провести дослідження комунікаційних технологій для даних систем;
- аналіз кібербезпеки та способів захищення інформації;
- розробити програмний продукт та графічний інтерфейс для користувача;
- виконати верифікацію даних.

Таким чином, сформовано початкові вимоги для розробки моделі системи IoT рішень для процесу сироваріння та завдання для кваліфікаційної роботи магістра.

1.2 Аналіз базових принципів роботи Internet of Things

IoT відіграє важливу роль у технологічному прогресі, оскільки тут можна знайти вирішення більшості проблем. Це технології майбутнього, яке наступило вже сьогодні. Багато підприємств впроваджують IoT для підвищення їх продуктивності та ефективності.

Уявіть, що речі навколо нас починають розмовляти і надають корисну інформацію. Наприклад, парасолька, що інформує про наявність дощу сьогодні й необхідність взяти її з собою.

Концепція IoT виникла в 1999 році. До 2010 року цей напрям обріс різними технологіями, отримав практичне застосування і з того часу стрімко розвивається.

За прогнозами дослідницької компанії Gartner, у 2020 році відбудеться злиття IoT з декількома IT-трендами. Штучний інтелект, машинне навчання і периферійні обчислення, з'єднавшись з IoT, відкриють нові можливості та в найближчі 5 років серйозно вплинуть на ринок [7].

За оцінками IDC, до 2025 року 95% даних, що збираються в реальному часі, будуть оброблятись на базі IoT рішень [8].

Ідея IoT полягає у взаємодії речей з сервером і між собою, де участь людини зводиться до мінімуму. Наприклад, лічильники електроенергії, що

відсилають показники в керуючу компанію, GPS-трекери, які відстежують рух публічного транспорту або ж фітнес браслети – це все IoT.

Існує величезна кількість різних пристроїв, підключених до мережі інтернет, які вимагають наявність оптимальної архітектури. IoT технології дозволяють пристроям генерувати дані в режимі реального часу, які ми можемо потім проаналізувати та використати для створення та вдосконалення різних систем. IoT система формує розгалужену мережу з підключеними пристроями, і ці пристрої збирають та обмінюються даними про те, як вони використовуються та про середовище, в якому вони працюють.

Основні компоненти IoT екосистеми зображені на рисунку 1.1:

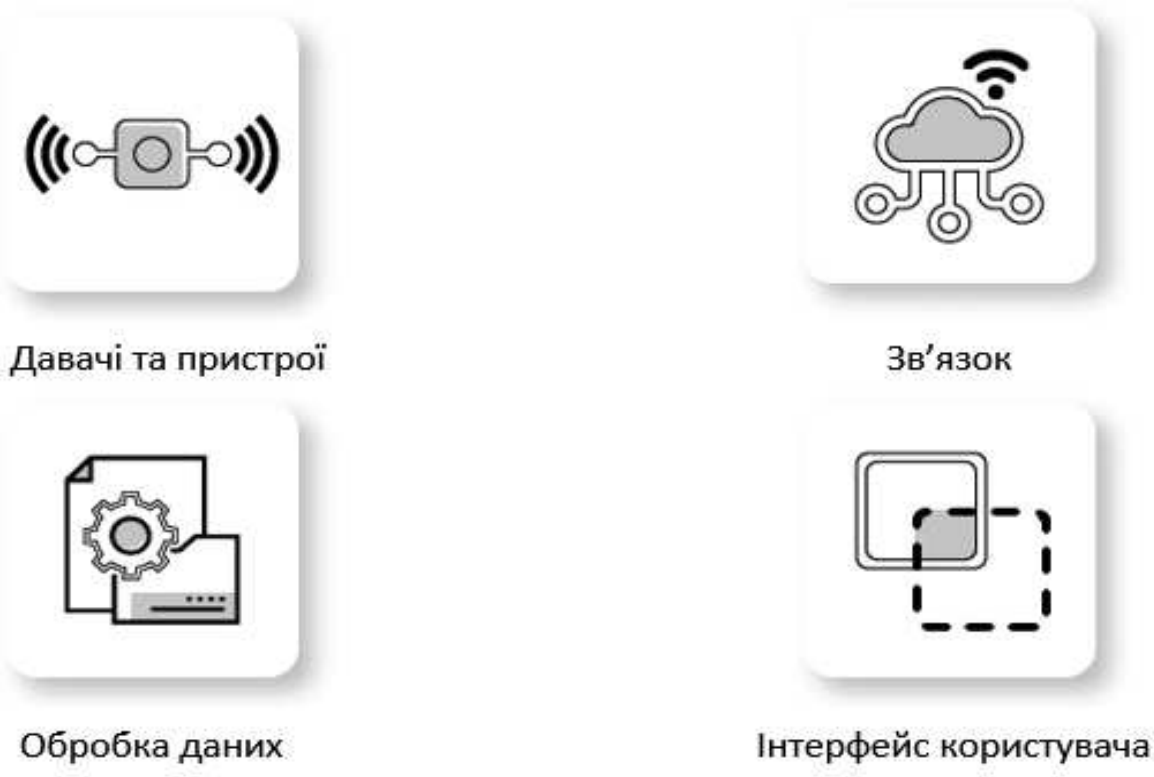


Рисунок 1.1 – Основні компоненти IoT екосистеми [9]

1) Давачі та пристрої

Найсуттєвіші складові, які слід враховувати в IoT технологіях – це давачі та пристрої. Вони використовуються для збору та обміну даних з навколишнього середовища. Можуть бути персональними, переносними чи вбудованими.

2) Зв’язок.

При підключенні до мережі, дані, зібрані давачами, надсилаються до хмарної інфраструктури, але для їх транспортування потрібен носій. Для цього використовують такі засоби, як супутникові мережі, Wi-Fi, Bluetooth, широкосмугові мережі (WAN) тощо. Ефективність безпеки IoT сильно залежить від швидкості й доступності цих засобів. Тому їх обирають відповідно до специфікацій, енергоспоживання, пропускної здатності та діапазону.

3) Обробка даних

Після досягнення хмарної інфраструктури дані необхідно проаналізувати, щоб вжити подальші правильні дії. Реалізується на базі віртуального сервера, реальної машини або за допомогою хмари. Аналіз може бути простим, як наприклад перевірка температури, або складним – виявлення крадія в будинку за допомогою камер.

4) Інтерфейс користувача

Останній крок – повідомлення користувачеві певної інформації. Це може бути сповіщення або ж звуковий сигнал, що надсилається на мобільний додаток. Таким чином забезпечується доступ до даних пристроїв і наочне зображення результатів аналізу, а також можливість ручного налаштування певної системи.

Усім відомо про розумні будинки та автомобілі, але IoT технології мають набагато ширший спектр застосування. Розглянемо галузі використання IoT у сучасному світі:

1) Охорона здоров’я

IoT відкривають нові можливості для медичних фахівців та пацієнтів. Нові технології дозволяють лікарям отримувати доступ у реальному часі до медичних даних пацієнтів, зберігати їх у хмарному сховищі й надсилати іншим фахівцям. Це скорочує час очікування пацієнта, допомагає перевірити наявність необхідного обладнання для лікування та спрощує процес виявлення хронічних захворювань, що набагато зменшує ризики. Також великим досягненням є використання медичних дронів, які можуть прилетіти на допомогу набагато швидше й у важкодоступні для людей місця [10,11].

2) Освіта

IoT технології також зробили революцію в галузі освіти. Вони поєднують людей у всьому світі, полегшують процес обміну знаннями, зменшують бар'єр у доступі до потрібних даних [10,11].

3) Роздрібна торгівля

Галузь IoT спрощує передачу персоналізованого досвіду та бази даних зацікавлених користувачів-покупців. Також автоматизує процеси оформлення замовлень та виконує обслуговування без особливих зусиль [10,11].

4) Економіка

Поєднання технологій IoT та Blockchain, тобто Blockchain of Things, суттєво змінює погляд на економіку. Подвійна технологія допомагає творити інтернет ринок, де можна зберігати та обмінюватися даними з компаніями та магазинами надійно та ефективно з використанням концепції розумних контрактів, що значно скорочує зусилля користувача [10,11].

5) Агрокультура

IoT технології дуже широко використовуються у сфері сільського господарства. На сьогодні доступно безліч інструментів, що використовуються для прогнозу погодних умов, перевірки стану ґрунту, отримання інформації про стан здоров'я домашніх тварин, відстежування їх місцезнаходження тощо [10].

6) Промисловість

У сфері промисловості навіть уведено спеціальний термін – Industrial Internet of Things (IIoT). Це об'єднана мережа багатьох пристроїв, що утворює єдину систему, яка відслідковує, збирає, аналізує та надає нові цінні дані. Наприклад, вбудовані давачі для моніторингу стану роботів на підприємництві, які символізують про потребу технічного обслуговування [10,11].

7) Розумне місто

Колаборація IoT технологій, таких як управління дорожнім рухом, освітленням, паркінгом, рівнем забруднення, публічним транспортом, та ін. утворюють екосистему розумного міста (рис.1.2). В розвинених країнах вже є безліч успішних прикладів впровадження таких екосистем [10,11].

Схема роботи екосистеми IoT зображена на рисунку 1.2.

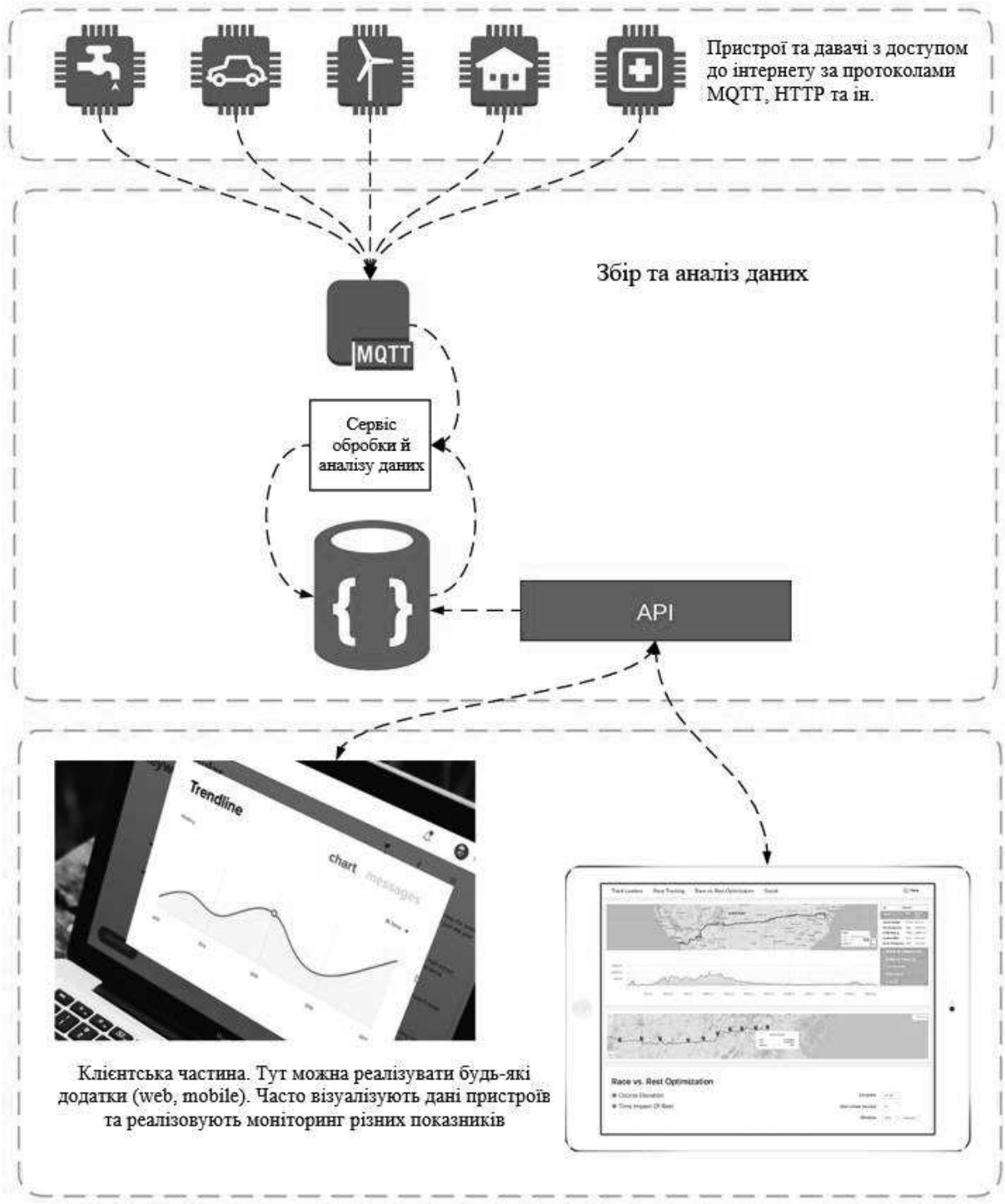


Рисунок 1.2 – Схема роботи IoT екосистеми [12]

Майбутнє IoT практично необмежене завдяки прогресу технологій. Можливості, що відкриваються, є вражаючими, а саме: підвищення продуктивності праці, збільшення доступності інформації, автоматизації

багатьох процесів та полегшення життя людей. З кожним днем перевага надається широкому впровадженню IoT технологій, незалежно від галузі застосування.

1.3 Аналіз технологічних стадій виготовлення сиру

Промислове виробництво сиру – це процес, який характеризується багатьма змінними. Сучасні заводи повністю автоматизовані для більшості операцій, що виконуються конвеєрним методом. За робочу добу переробляють від 30 до 300 т молока.

На рис. 1.3 наведена послідовність технологічних стадій виготовлення сирів.

Рисунок 1.3 – Технологічні стадії виробництва сирів [13]

Спочатку молоко охолоджують до 6-8 °С та відправляють на резервування. За нижчих температур (4-5 °С) погіршуються його технологічні властивості.

Свіже молоко має пройти етап дозрівання – витримка впродовж 12-16 год за температури 8-12 °С. Це пояснюється тим, що воно певний час зберігає бактерицидні властивості, що перешкоджає розвитку молочно-кислих бактерій.

Так як у свіжому молоці розвиваються різні організми, воно не є цілком стерильним. Збільшується рівень амінокислот, які створюють сприятливі умови для розвитку молочних бактерій. Внаслідок цього утворюється молочна кислота, що переводить частини кальцію в розчинний стан.

Сьогодні ж на сучасних заводах не проводять дозрівання молока. Воно проходить саме по собі в період доставки молока з різних куточків країни та під час резервування.

Для одержання сиру відповідної жирності, відбувається наступний крок – нормалізація молока по відсотку жирності. Для цього до нового молока додають знежирене молоко або вершки, що також сприяє зниженню шкідливої мікрофлори. Пастеризація відбувається за температури 71-72 °С з витримкою 20-25 с. У сироварінні прийнято не використовувати термообробку, так як це знижує його придатність. Популярним методом є бактеріофугування молока на спеціальних установках – бактеріофугах. Під час пастеризації також знижується й корисна мікрофлора. Тому після пастеризації додають бактеріальну закваску.

У молоко, також, додають хлористий кальцій (10-40 г на 100 кг молока) у вигляді водного розчину. Це сприяє підвищенню вмісту іонів кальцію та покращення процесу зсідання.

Під час процесу виготовлення сиру потрібно сповільнювати розвиток шкідливої мікрофлори, такої як кишкові палички та молочнокислі бактерії. Для цього у молоко додають калійну селітру (10-30г на 100кг молока).

Наступний етап – сичужне згортання молока. У молоко додають сичуговий порошок (2.5 г на 100 кг молока) й для твердого сиру тривалість 25-30 хв, а для м’якого – 40-90хв. Результатом цього процесу є сичуговий згусток, який потім зневоднюють, розрізають та дрібнять. Утворюється сирне зерно, розмір якого залежить від типу потрібного сиру. Наприклад для швейцарського це 2-3 мм, а для м’яких сирів – до 10-15 мм. Кінцевий стан сирного зерна

визначається його фізичними властивостями: міцність, клейкість та сухість. Для перевірки його стискають у долоні й розтирають. При належній обробці сирна кулька відразу розпадається на окремі зерна. Далі починають формувати сири. Це проводиться двома способами: насипом, для отримання пустотної форми та з пласту, для рівномірного малюнку.

У спеціальні форми насосом подається сирне зерно та сироватка. Потім сироватку видаляють, а сирну масу піддають пресуванню впродовж 10-15 хвилин. Утворюється сирний пласт, який розрізають на потрібні частини, загортають у серветки й поміщають від прес в тепле приміщення за температури 20-22 °С.

Для соління сирів існує декілька способів: використання сухої солі, соляної пудри та розсолу. Найбільш поширений – сири поміщають у бетонні басейни з розсолом.

Дозрівання сирів проходить у сховищах на стелажах-контейнерах. В залежності від типу сиру, підтримується потрібний температурно-вологий режим. В зв’язку з цим на сирних заводах присутні декілька приміщень з відповідними кліматичними умовами.

Фінальний етап – сортування, упаковка й транспортування готового продукту. Розсортовані сири обгортають в спеціальний папір та перевозять з дотриманням температурного режиму 8-12 °С за необхідної вологості повітря 85-87 % [13].

1.4 Аналіз останніх наукових досліджень та практичних розробок

Клаус Шваб пише у книзі [14] : «Одним з головних мостів між фізичною та цифровою реальністю, який створений четвертою промисловою революцією, є Інтернет речей». Дійсно, сьогодні виробничі системи перетворюються на цифрові екосистеми. У цій трансформації головну роль відіграють Інтернет речі та Великі дані. З цією метою промислові підприємства вступають в нову еру «великих даних», де обсяг, швидкість та різноманітність даних, якими вони керують, вибухають з дуже високими темпами [15]. Інтернет речі, що описують мережу взаємопов’язаних об’єктів за допомогою вбудованих технологій,

безпосередньо поєднуються з концепцією Великих даних, що дозволяє збирати ще більше інформації [16]. Все більше пристроїв, виробничого обладнання та інструментів, транспортних засобів стає оснащене різними давачами.

У статті [17] стверджують, що Інтернет Речей найбільше розвивається завдяки складовій у сфері промисловості (Промисловий Інтернет речей). Це суттєво підвищує ефективність виробництва завдяки вбудованим давачам з можливістю автоматизації та віддаленого контролю без участі людського фактору. Одним з яскравих прикладів є виробництво бритв Philips, що відбувається з використанням 128 роботів у темному приміщенні [18]. Інший приклад – виробництво мотоциклів Harley Davidson, на якому суттєво скоротили час простою. Після реконструкції виробничих майданчиків час на виготовлення зменшився з 28 днів до 16 годин [18]. Завдяки Промисловому Інтернету речей у виробників з'являється доступ до передових аналітичних інструментів, штучного інтелекту та машинного навчання, що пришвидшує прийняття рішень та поліпшує виробничі показники.

У статті [19] приведено основні принципи, що впливають на ріст та розповсюдження Інтернету Речей. Це дотримання безпеки та конфіденційності даних; головне – якість, а не кількість даних; розвиток розумних міст і будинків та колаборація з бізнесом. Прикладом провідного розумного міста можна привести Барселону, що з 2012 року впроваджує IoT [20]. Встановлено розумні світильники, які змінюють яскравість, коли пішохід поруч, вимірюють трафік, якість повітря, рівень шуму та натовпу людей і навіть пропонують безкоштовний доступ до міської Wi-Fi мережі. Також вбудовано давачі в автомобільні місця для паркування, впроваджено цифрові автобусні зупинки, розумні смітники та давачі для оптимізації зрошення паркової зони. За статистикою, що наведена у статті [20], близько 26% користувачів в інтернету в США володіють пристроєм розумного будинку. Це світильники, які відповідно до вашого графіку роботи вмикають та вимикають світло, холодильники, що аналізують свій вміст і створюють список покупок, кавоварки, які автоматично починають варити, коли

ви прокидаєтесь та багато іншого. Технологічне майбутнє, яке ніхто не міг уявити, наступило вже сьогодні.

Різні популярні та інноваційні рішення Інтернету речей, такі як переробка відходів, моніторинг якості повітря, розумне землеробство розглянуто в роботі [21] в контексті прогресу технологій та перспектив промислового ринку. На Всесвітньому економічному форумі представлений звіт щодо наслідків, можливостей, переваг та загроз Промислового Інтернету речей [22]. Це спричинить нову тенденцію розвитку, що дозволить приймати автоматизовані рішення та вживати заходи в режимі реального часу. У роботі [23] автор представив різні аспекти та технологічні перспективи в промисловому застосуванні Інтернету речей. Різниця Промислового Інтернету речей від споживаного в створенні переваг для бізнесу за рахунок підключення розумних девайсів на виробництві.

У роботі [24] представлено систему розумного будинку, яка керує побутовою технікою будинку за допомогою комп'ютера. Приладами можна керувати голосовими командами або таймером. У роботі [25] розроблено систему, де побутові прилади керувалися через веб-сторінку або доданок для Android смартфонів. Проте у цих системах використовувалась технологія Bluetooth, яка має обмежену дальність (на той час максимум 10 м) та низьку швидкість передачі даних.

У роботі [26] представлено доступну по ціні систему екологічного моніторингу. Використано вбудований мікро веб-сервер на базі мікроконтролера Arduino Mega 2560. Це дає змогу керування системою еко-моніторингу з можливістю підключення за IP-адресою для віддаленого доступу.

У роботах [27-28] розроблено систему автоматизації будинку, впровадивши для керування побутовою технікою мікроконтролер Arduino Due та модуль зв'язку ESP8266-01, що отримував дані зі смартфону через мережу WI-FI. Серед можливостей представлено управління світлом, вентиляцією та сигналізація витоку газу за допомогою відповідних давачів. Модуль ESP8266-01

дозволяє передавати дані зі швидкістю до 11 Мбіт/с та діапазоном зв’язку 150 метрів, що значно більше за наявних конкурентів [29].

Враховуючи видимі переваги, прийнято рішення обрати для розробки системи клімат-контролю камери дозрівання сирів модуль ESP8266-01.

1.5 Аналіз готових систем на ринку

Для пошуку аналогів технічних та програмних IoT рішень на тему кваліфікаційної роботи магістра потрібно розглянути запропоновані варіанти на ринку.

Першим досліджено найпопулярніший варіант клімат-контролю для камери дозрівання сирів «Модульна блок-установка» (рис. 1.4) [30].

Виробником розроблена канальна кліматична установка для забезпечення необхідних режимів температури та волості повітря в камері визрівання сирів.

Установка має наступні характеристики:

1) Підтримка температурного режиму від 10 ℃ до 20 ℃;
2) Підтримка вологості повітря до 99%;
3) Споживана потужність близько 2,2 кВт;
4) Об’єм камери від 30 м^3 до 60 м^3.

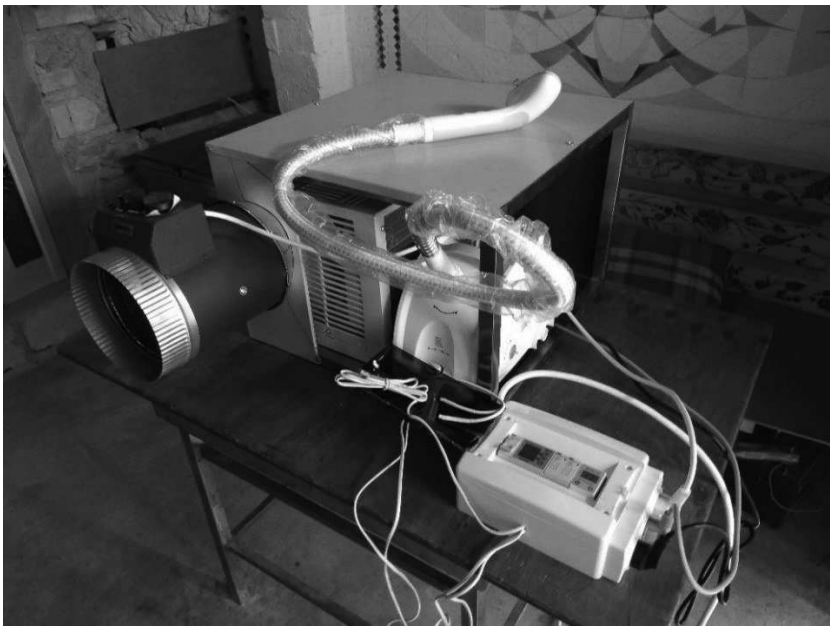


Рисунок 1.4 – Система клімат-контролю для камери дозрівання сирів [30]

Установка застосовується для обладнання приміщень з нестабільною температурою протягом різних пір року. Принцип роботи засновано на контролі датчиків терморегулятора температури всередині приміщення. У спекотну погоду працює система охолодження, в холодну – нагрівання. Для підтримки вологості повітря необхідне підключення до водопровідної мережі.

Ціна такої установки сягає близько 1000 доларів [30].

Основним недоліком є наявність одного варіанту реалізації без використання апаратно-програмних технологій (так як керування здійснюється за рахунок встановлення вручну температури на терморегуляторі) та висока цінова політика.

Наступною розглянемо систему управління клімат-контролю для теплиць «Терраформ» (рис. 1.5) [31].

Рисунок 1.5 – Система клімат контролю для теплиць [31]

Клімат-контроль для теплиць «Терраформ» дає змогу управляти опаленням, поливом, освітленням та вентиляцією. Головною особливістю є можливість віддаленого контролю комунікацій за допомогою будь-якого пристрою, підключеного до мережі інтернет.

Характеристики установки:

1) 8 реле для керування пристроями;
2) потужність навантаження реле 1 кВт;
3) 4 давачі температури ґрунту, рідини;

4) 4 давачі вологості ґрунту;

5) 4 давачі освітленості ;

6) 2 давачі концентрації CO_2.

Ціна системи управління близько 800 доларів [31].

Основним недоліком є низька потужність навантаження на реле – всього 1 кВт. Її явно не вистачить для охолоджувача чи підігрівача приміщення.

Також розглянуто кліматичний комплекс для житлових приміщень від виробника Zenet (рис. 1.6) [32].

Рисунок 1.6 – Система кліматичного комплексу Zenet [32]

Кліматичний комплекс Zenet має функції обігіріву, зволоження, охолодження, вентиляції та очищення повітря. Комплекс працює на «акумуляторах» з льоду, які встановлюються у відділення з водою. Охолоджена вода циркулює по системі фільтрів, через які проходить повітря й очищується, зволожується та охолоджується.

Характеристики установки:

1) споживана потужність 65 Вт;

2) зниження температури на 7 градусів;

3) мобільність.

Ціна кліматичного комплексу близько 150 доларів [32].

Недоліками даної установки є регулярна заміна «акумуляторів» з льоду, ручне налаштування та малий функціонал.

Для того, щоб порівняти системи клімат-контролю «Модульна блок-установка», «Терраформ», «Zenet» складемо таблицю порівнянь для цих систем (Табл. 1.1). Оберемо наступні критерії: призначення, можливості, характеристики, недоліки, ціна та кількість задач, що вирішуються. Дані критерії найкраще характеризують основні параметри систем клімат-контролю.

Таблиця 1.1 Порівняння систем клімат-контролю

		Назва системи		
		«Модульна блок-установка»	«Терраформ»	«Zenet»
1	**2**	**3**	**4**	**5**
Критерії порівняння	**Призначення**	Клімат-контроль для камери дозрівання сирів.	Клімат-контроль для теплиць	Кліматичний комплекс для житлових приміщень
	Можливості	Забезпечення необхідних режимів температури та вологості повітря.	Управління опаленням, поливом, освітленням та вентиляцією. Віддалений контроль.	Обігрів, зволоження, охолодження, вентиляція та очищення повітря. Працює на «акумуляторах» з льоду.
	Характеристики	1) Підтримка температурного режиму від 10 °C до 20 °C; 2) Підтримка вологості повітря до 99%; 3) Споживана потужність близько 2,2 кВт; 4) Об’єм камери від 30 м3 до 60 м3.	1) 8 реле для керування пристроями; 2) потужність навантаження реле 1 кВт; 3) 4 давачі температури ґрунту, рідини; 4) 4 давачі вологості ґрунту; 5) 4 давачі освітленості ; 6) 2 давачі концентрації CO2.	1) споживана потужність 65 Вт; 2) зниження температури на 7 градусів; 3) мобільність.

Продовження табл. 1.1

1	2	3	4	5
	Недоліки	Один варіанту реалізації без використання апаратно-програмних технологій.	Низька потужність навантаження на реле – всього 1 кВт.	Регулярна заміна «акумуляторів» з льоду, ручне налаштування та малий функціонал.
	Ціна	1000 доларів	800 доларів	150 доларів
	Вирішує	3 задачі	6 задач	4 задачі

Аналізуючи запропоновані варіанти на ринку України (акцент саме для цього регіону) можна зробити висновок, що система IoT рішень для процесу сироваріння взагалі не представлена. Існують поодинокі рішення для кліматконтролю камери зберігання сирів, теплиць чи житлових приміщень. Характеристики даних систем, приведених в таблиці 1.1, є недостатніми для моделі, що розробляється в кваліфікаційній роботі магістра, а недоліки вважаю суттєвими.

1.6 Визначення вхідних даних для вирішення поставленої задачі

Зважаючи на відсутність оптимального програмно-апаратного рішення та складність процесу виготовлення сирів вирішено розробити модель системи IoT рішень для процесу сироваріння на підприємстві. Враховуючи характеристики та недоліки систем з таблиці порівнянь (таблиця 1.1), визначимо вхідні данні для проектування нашої моделі:

- підтримка необхідного температурно-вологого режиму (відповідно до підрозділу 1.3);
- можливість підключення достатньої кількості модулів давачів до панелі керування;

- віддалений автоматичний контроль системою та можливість ручного налаштування;
- низька споживана потужність;
- використання апаратно-програмних технологій;
- підтримка керування декількома приміщеннями;
- використання елементної бази відповідно до розрахованого навантаження;
- низька вартість системи.

На основі проведених досліджень сформуємо метод автоматичного управління технологічним процесом сироваріння.

1.7 Метод автоматичного управління технологічним процесом сироваріння

Сир є одним із найбільш споживаних молочних продуктів у всьому світі. Його сприйнятливість кінцевим споживачем багато в чому залежить від конкретних органолептичних властивостей, включаючи вигляд, форму, прозорість, смак і аромат. Усі ці властивості сиру залежать від багатьох факторів, а особливо від чіткого дотримання технологічних норм під час кожного етапу сироваріння. Задля підвищення ефективності виробництва сиру спробуємо впровадити технологію промислового інтернету речей й автоматизувати процеси шляхом використання систем вбудованих давачів та віддаленого контролю, а також виключимо можливість впливу людського фактору на процес.

Опишемо метод автоматичного управління технологічним процесом сироваріння. Він складається з наступних кроків:

Крок 1. Ініціалізація номінальних значень. Відповідно до значень, уведених фахівцем в систему визначаються кліматичні умови приготування та зберігання сирів.

Крок 2. Зчитування значень з навколишнього середовища. Давачі зчитують значення, що характеризують середовище, в якому вони знаходяться. Для реалізації вузла камери зберігання сирів основними значеннями є:

температура, вологість, рівень CO_2, освітлення; для інших вузлів додатково планується зчитувати значення кислотності pH та електропровідності.

Крок 3. Аналіз температурного режиму. Якщо значення температури середовища менше за номінальну температуру, то зупиняється охолоджувач та запускається обігрівач. Якщо ж значення температури середовища більше за номінальну – зупиняється обігрівач та запускається охолоджувач. В залежності від типу сиру, значення температури коливається від -4 °C до 6 °C. За дотримання низької температури дозрівання сирів сповільнюється, а занадто висока температура провокує появу цвілі.

Крок 4. Аналіз вологості. Якщо значення вологості середовища менше за номінальну вологість, то зупиняється осушувач та запускається зволожувач. Якщо ж значення вологості середовища більше за номінальне – зупиняється зволожувач та запускається осушувач. Значення вологості повітря має бути рівним 80-90%. Дотримання високого значення вологості повітря заважає утворенню щільної скоринки на поверхні сиру, а за низького значення вологості сири пересихають.

Крок 5. Аналіз рівня CO_2. Якщо рівень вуглекислого газу середовища менший за номінальне значення, то зупиняється вентиляція й відбувається насичення газом. Якщо ж рівень вуглекислого газу середовища більший за номінальне значення – запускається вентиляція й концентрація газу в повітрі зменшується. Регулювання рівня вуглекислого газу впливає на швидкість старіння сиру та утворення цвілі.

Крок 6. Аналіз освітлення. Якщо рівень освітлення середовища менший за номінальне значення, то подається сигнал «Вимкнуто освітлення». Якщо ж рівень освітлення середовища більший за номінальне значення – сигнал «Увімкнуто освітлення». За допомогою ультрафіолетового випромінювання регулюється рівень бактерій, утворення різних цвілей та сирних грибків.

Крок 7. Повторення кроків 2-6.

Інтелектуальна система клімат-контролю для камери зберігання сирів реагує на зміни від впливу навколишнього середовища на відміну від аналогів, де необхідні процеси по нормалізації середовища запускаються за таймером.

1.8 Висновки до розділу 1

Відсутність різноманіття методів і засобів реалізації моделі системи IoT рішень для процесу сироваріння свідчить про те, що оптимального програмно-апаратного рішення не знайдено, тому вважаємо за доцільне розгляд нової технологічної пропозиції за темою кваліфікаційної роботи магістра. Проведений спектр аналізів у даному розділі та пошук аналогів є підґрунтям для розробки прототипу. Основна мета – створити систему IoT рішень для процесу сироваріння на підприємстві, відповідно до вхідних даних, задовольняючи вимоги та задачі, описані в даному розділі. При цьому застосувати технології IoT, спроектувати інтерфейс управління для користувача та мінімізувати ціну.

РОЗДІЛ 2 РОЗРОБКА ПРОЕКТУ ІОТ РІШЕННЯ ТА МЕТОДІВ ОБРОБКИ ДАНИХ В ІОТ

У цьому розділі описується принцип роботи та розробляється схема структурна проектованої системи. Обирається елементна база відповідно до поставленої мети. Будуються UML-діаграми, описуються методи та засоби обробки даних й обґрунтовується вибір створення програмного продукту та використання хмарних технології.

2.1 Розробка проекту IoT рішення

2.1.1 Розробка схеми структурної та принцип роботи пристрою

Відповідно до поставленої мети, опишемо алгоритм роботи системи IoT рішень для процесу сироваріння.

1. Фахівець вводить номінальні значення температури, вологості, кислотності, електропровідності, освітлення та рівня CO2 для кожного етапу процесу сироваріння (рис. 1.3).

2. Зняти показники, що характеризують навколишнє середовище. Головними є температура, вологість, рівень кислотності та електропровідності. Додатково зчитується концентрація CO_2 в повітрі та освітленість.

3. Порівняти значення номінальні та реальні. Відповідно до результату, подати сигнал для виконання процесів (обігрів, охолодження, зволоження, осушування, вентиляція, управління освітленням), а також індикації стану якості молока на кожному етапі, що характеризується показниками кислотності та електропровідності.

Згідно з описаним алгоритмом, пристрій має складатись з:

1) мікроконтролеру Arduino;
2) давачу освітленості;
3) давачу температури;
4) давачу вологості;
5) давачу рівня CO_2;
6) давачу значення електропровідності;

7) давачу рівня pH;

8) системи управління набором реле;

9) модулю WI-FI.

Схема структурна для автоматизованої системи IoT рішень для процесу сироваріння зображена на рис. 2.1.

Мікроконтролер Arduino зчитує необхідні дані про стан навколишнього середовища з вбудованих давачів та передає їх на ПК користувача.

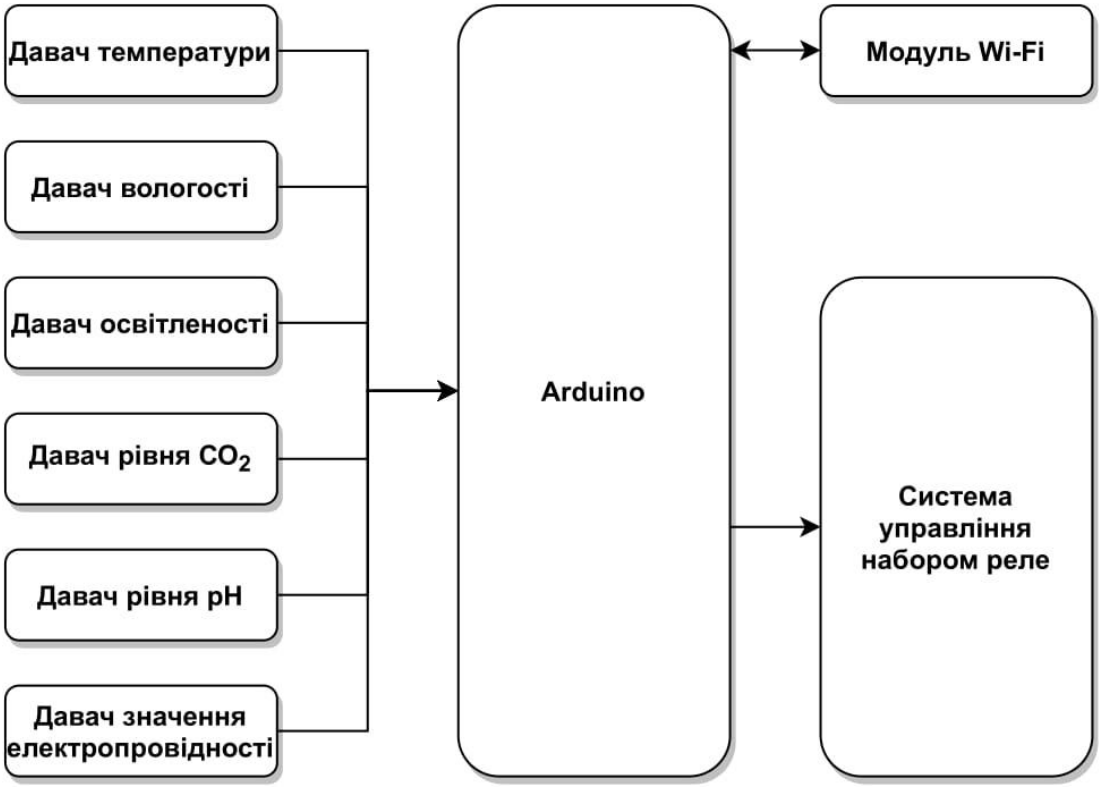


Рисунок 2.1 – Схема структурна пристрою

Відповідно до результату порівняння зчитаних даних та номінальних (оптимальні дані для зберігання сирів, встановлюються на персональний комп'ютер (ПК) в програмі керування), мікроконтролер (МК) подає сигнал для активації відповідного реле для виконання певної операції відносно нормалізації навколишнього середовища.

2.1.2 Вибір та обґрунтування елементної бази

Вибір елементної бази проводиться згідно з поставленими вимогами у технічному завданні відповідно до схеми структурної. Вагомими є мінімальні затрати при виготовленні пристрою.

Мікроконтролер

Основні вимоги для вибору моделі мікроконтролера Arduino є підтримка інтерфейсів UART для «спілкування» з ПК, I2C для підключення давача освітленості, температури й вологості та наявність достатньої кількості виводів.

Обрано мікроконтролер із серії Arduino Nano V3 на базі ATmega328P (рис.2.2) [33].

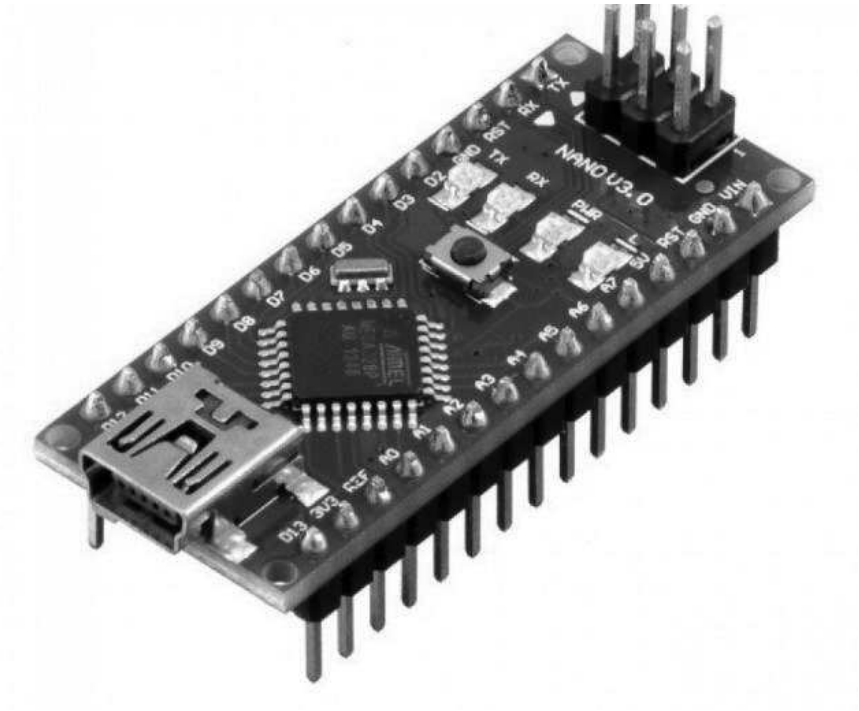


Рисунок 2.2 – Arduino Nano V3 [33]

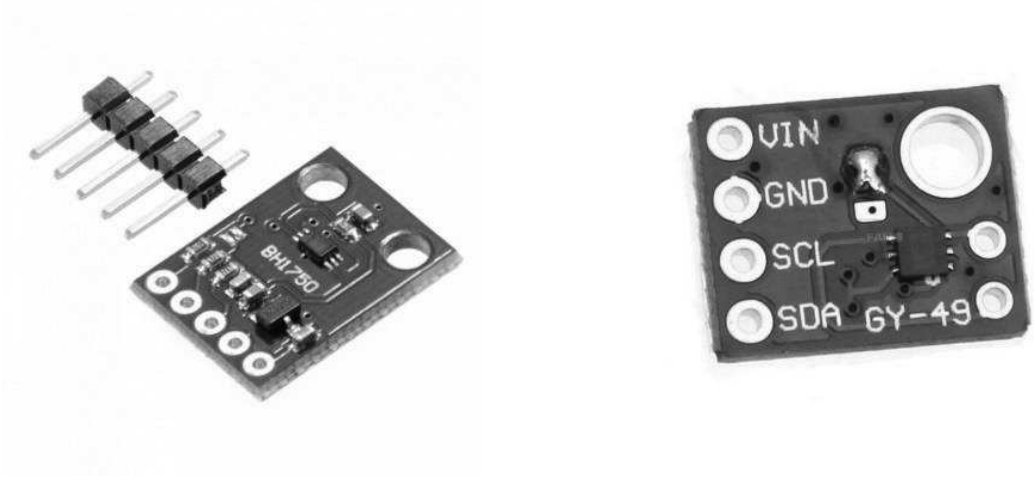
Основними перевагами МК є:

1) достатня кількість аналогових (8) та цифрових (14) пінів;
2) 32 кб flash-пам’яті;
3) 2 кб SRAM;
4) тактова частота 16 МГц;
5) наявність UART, I2C, SPI інтерфейсів;
6) маленький розмір;
7) доступна ціна.

Давач освітленості

Розглянемо наступні давачі: GY-302, GY-49 (рис.2.3) [34,35].

Вони всі призначені для вимірів освітленості та мають високу чутливість. Давач GY-302 має діапазон вимірюваних даних від 0 до 65535 Люкс та ціну 48 грн., а GY-49 - від 0 до 188000 Люкс та 78 грн.



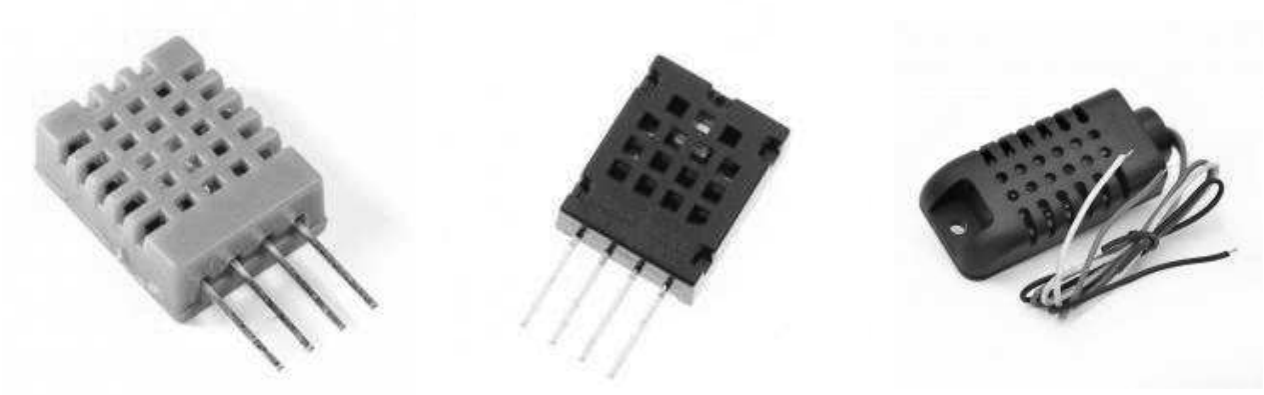
GY-302 GY-49

Рисунок 2.3 – Давачі освітленості [31,32]

Діапазон робочих температур однаковий – від -40℃ до 85℃ [34,35]. Обираємо GY-302, так як він дешевший та цілком задовольняє своїм діапазоном вимірів.

Давач температури та вологості

Під час пошуку давачів вологості виявлено, що вони йдуть в парі з давачами температури. Розглянемо наступні давачі: DHT11, AM2320, DHT21 (рис.2.4) [36-38].



DHT11 AM2320 DHT21

Рисунок 2.4 – Давачі температури та вологості [33-35]

Їх характеристики наведено в таблиці 2.1 [36-38].

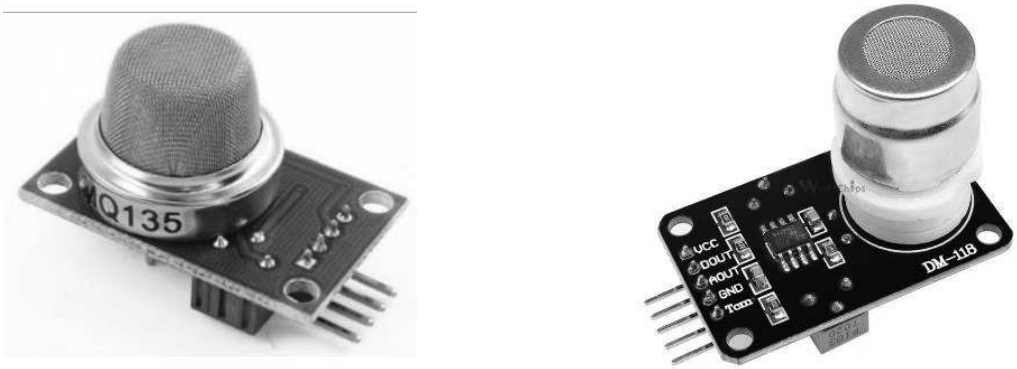
Таблиця 2.1 – Характеристики давачів температури та вологості [36-38]

	DHT11	AM2320	DHT21
Діапазон виміру температури	0 - 50℃	-40 - 80℃	-55 - 80℃
Точність виміру температури	±5%	±0.5%	±0.5%
Діапазон виміру вологості	20-90RH	0-100RH	0-100RH
Точність виміру вологості	±5%	±2%	±2%
Ціна, грн	29	64	110

Відповідно до таблиці 2.1 давач DHT11 має меншу точність вимірів (±5%), ніж інші та діапазон вимірюваних температур, що не відповідає заданому в ТЗ. Давачі AM2320 та DHT21 мають схожі характеристики. Але AM2320 дешевший, тому обираємо саме його.

Давач рівня CO_2

Розглянемо наступні давачі: MQ-135 та MG-811 (рис.2.5) [39,40].



MQ-135 MG-811

Рисунок 2.5 – Давачі рівня CO_2 [39,40]

Давач MQ-135 використовується для виміру вмісту шкідливих газів у повітрі таких як: аміак, окиси азоту, пари вуглецю, бензину, диму і т.ін. Ціна давача 67 грн. [39].

Давач MG-811 – високоякісний сенсор, що застосовується для визначення концентрації вуглекислого газу CO_2. Ціна давача 1592 грн. [40].

Обираємо давач MQ-135, так як він має ширший спектр можливостей та нижчу ціну.

Wi-Fi модуль

Модуль ESP8266-01 – мікроконтролер з інтерфейсом Wi-Fi. Дозволяє передавати дані зі швидкістю до 11 Мбіт/с та діапазоном зв'язку 150 метрів, що значно більше за конкурентів (рис.2.6) [41].

Рисунок 2.6 – Модуль ESP8266-01 [41]

Реле

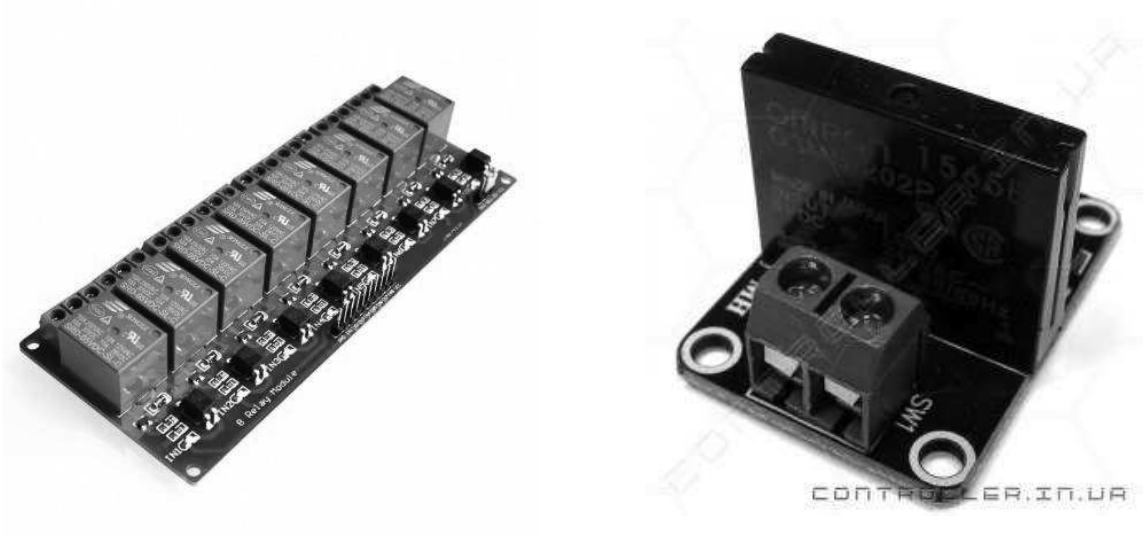
Для управління мікроконтролером Arduino потужним навантаженням використовують реле, що по суті являє ключ. Максимальний струм навантаження обраного реле для кожного каналу модуля 10 А, а напруга 250 В. Ми отримаємо потужність $P = I \cdot U = 10 \cdot 250 = 2.5$ (кВт) на кожен канал, чого має вистачити для керуючих приладів. Для реалізації проекту потрібно 6 каналів: обігрів, охолодження, зволоження, осушування, вентиляція, управління освітленням.

Розрізняють, також, твердотільне реле. Воно має ряд переваг, таких як:

1) висока швидкість перемикання;
2) тривалий термін експлуатації;
3) безшумність;
4) відсутність іскри та стрибка напруги;

5) економія електроенергії та ін.

Приклад таких реле приведено на рис 2.7 [42,43].



8-канальне реле Твердотільне реле

Рисунок 2.7 – Реле [42,43]

Для впровадження у виробництво рекомендується обрати твердотільне реле. Але для тестової реалізації проекту буде достатньо й звичайного.

Давач рівня pH

Рівень кислотності pH є важливою характеристикою молока. Показник кислотності свіжого молока рівний 6,6-6,75 pH. Значення pH змінюється під час концентрації або ж додавання домішок до молока, підігрівання чи сквашування в результаті біохімічних процесів. У промислових масштабах рівень pH визначають, коли сквашування істотно впливає на якість вихідного продукту. Відповідність стану молока та рівня pH приведено в таблиці 2.2 [44].

Таблиця 2.2 – Відповідність стану молока від рівня pH [44]

Характеристика молока	**Рівень pH**
Молоко від корів з захворюванням вимені	> 6,8
Нормальне свіже молоко	6,6-6,8
Молоко, яке починає скисати	6,3
Згортання молока при нагріванні	5,7
Згортання з утворенням густих згустків	5,3-5,5

Для виміру рівня кислотності в молочних продуктах необхідно обрати давач рівня pH. Розглянемо аналоговий недорогий давач рівня pH для Arduino (рис.2.8) [45].

Рисунок 2.8 – аналоговий давач рівня pH [45]

Він має просте, зручне і практичне підключення, а також володіє наступними характеристиками:

1) Напруга живлення: 5 ± 0,2 В
2) Робочий струм: 5-10 млА
3) Діапазон виявленої концентрації: 0-14 pH
4) Діапазон температури виявлення: 0-80 ℃
5) Ціна: 946 грн

Серед доступних варіантів по ціні/якості – це оптимальний варіант в Україні, що задовольняє всі наші потреби, тому обираємо саме його.

Давач значення електропровідності

Крім рівня кислотності важливим параметром якості молока є його електропровідність. При 20 ℃ значення електропровідності коливається від 3 мСм/см до 6 мСм/см і залежить від концентрації іонів у молоці. Для зменшення концентрації іонів додають воду, цукор, рідше протеїни або нерозчинні солі, таким чином зменшуючи провідність молока. При додаванні солей концентрація іонів навпаки збільшується, що спричиняє у свою чергу збільшення провідності

молока. Якщо при 18 °С електропровідність становить 6,5-13 мСм/см, це сигналізує про наявність небезпечного, стрімко розповсюджуваної інфекційно-запальної хвороби молочної залози – маститу. Для вимірювання електропровідності використовують – кондуктометр. Варто зазначити, що температура є важливим фактором на який необхідно звертати увагу, вона безпосередньо впливає на показники кислотності та електропровідність [44].

Розглянемо аналоговий давач виміру електропровідності для Arduino, що зображений на рисунку 2.9 [46].

Рисунок 2.9 – аналоговий давач електропровідності [46]

Цей давач розроблений спеціально для контролерів Arduino з вбудованими простими, зручними й практичними функціями. Має наступні характеристики:

1) Робоча напруга: 5 В
2) Діапазон вимірювання: 1мСм/см - 20мСм/см
3) Точність: ± 10%
4) Ціна: 1998 грн

Діапазон та точність вимірювань а також напруга живлення повністю задовольняють наші потреби. Обираємо цей варіант.

Отже, у моделі системи IoT рішень для процесу сироваріння мікроконтролер Arduino Nano V3 зчитує необхідні дані про стан навколишнього середовища (AM2320, GY-302, MQ-135) та якість молока (рівень pH та електропровідності) з вбудованих давачів та передає їх на ПК користувача. Відповідно до результату порівняння фактичних даних та номінальних (встановлюються фахівцем на ПК в програмі керування), МК подає сигнал для

активації реле, що запускає виконання певної операції відносно нормалізації стану середовища.

Схема мережі зображена на рисунку 2.10.

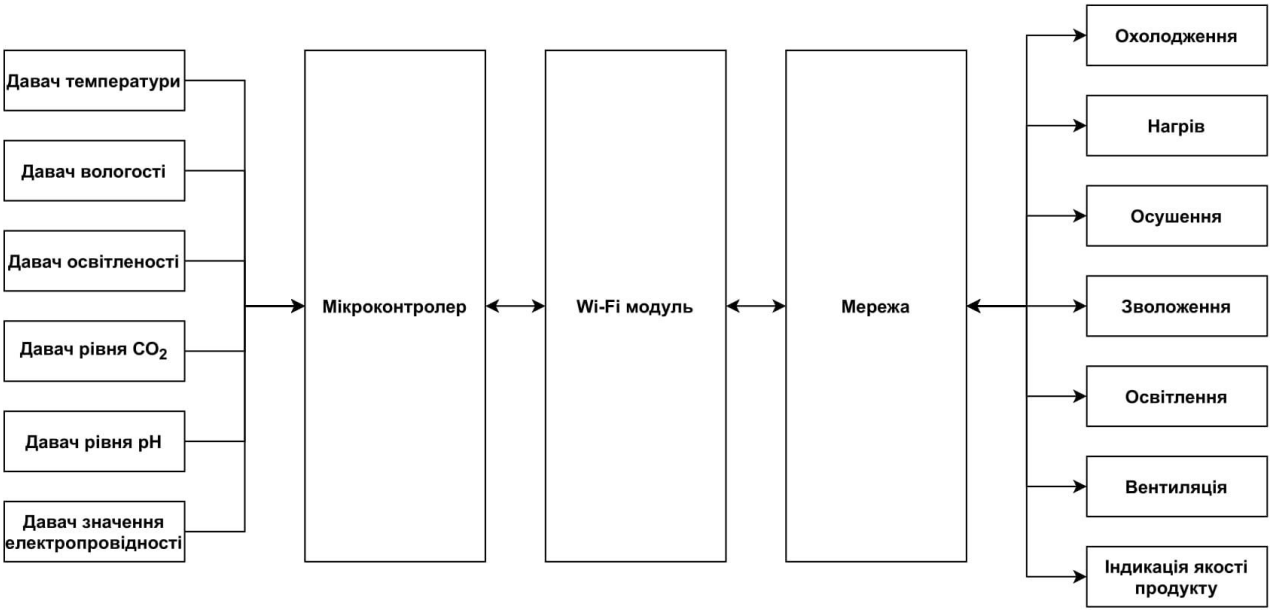


Рисунок 2.10 – Схема мережі моделі системи IoT рішень для процесу сироваріння

Усі компоненти обрано у низькому ціновому сегменті та відповідають поставленій меті.

2.1.3 Проект системи IoT рішень для процесу сироваріння

UML (Unified Modeling Language) – стандартизована мова моделювання, що дозволяє візуалізувати програмний продукт у вигляді різних типів діаграм [47]. Для реалізації проектної частини системи IoT рішень для процесу сироваріння побудуємо UML-діаграми використання, послідовності та стану.

Діаграма використання показує взаємозв’язок між користувачем та системою. Користувач встановлює номінальні значення в системі IoT рішень для процесу сироваріння. Мікроконтролер зчитує значення з давачів та відправляє їх на програмне забезпечення (ПЗ). ПЗ в свою чергу зчитує номінальні значення від користувача, отримує значення з давачів від МК, порівнює їх між собою та подає сигнал на МК для активації процесу. МК обробляє сигнал від ПЗ та активує обладнання для нормалізації стану середовища (обігрів, охолодження,

зволоження, осушування, вентиляція та освітлення). Діаграма використання системи IoT рішень для процесу сироваріння зображена на рисунку 2.11.

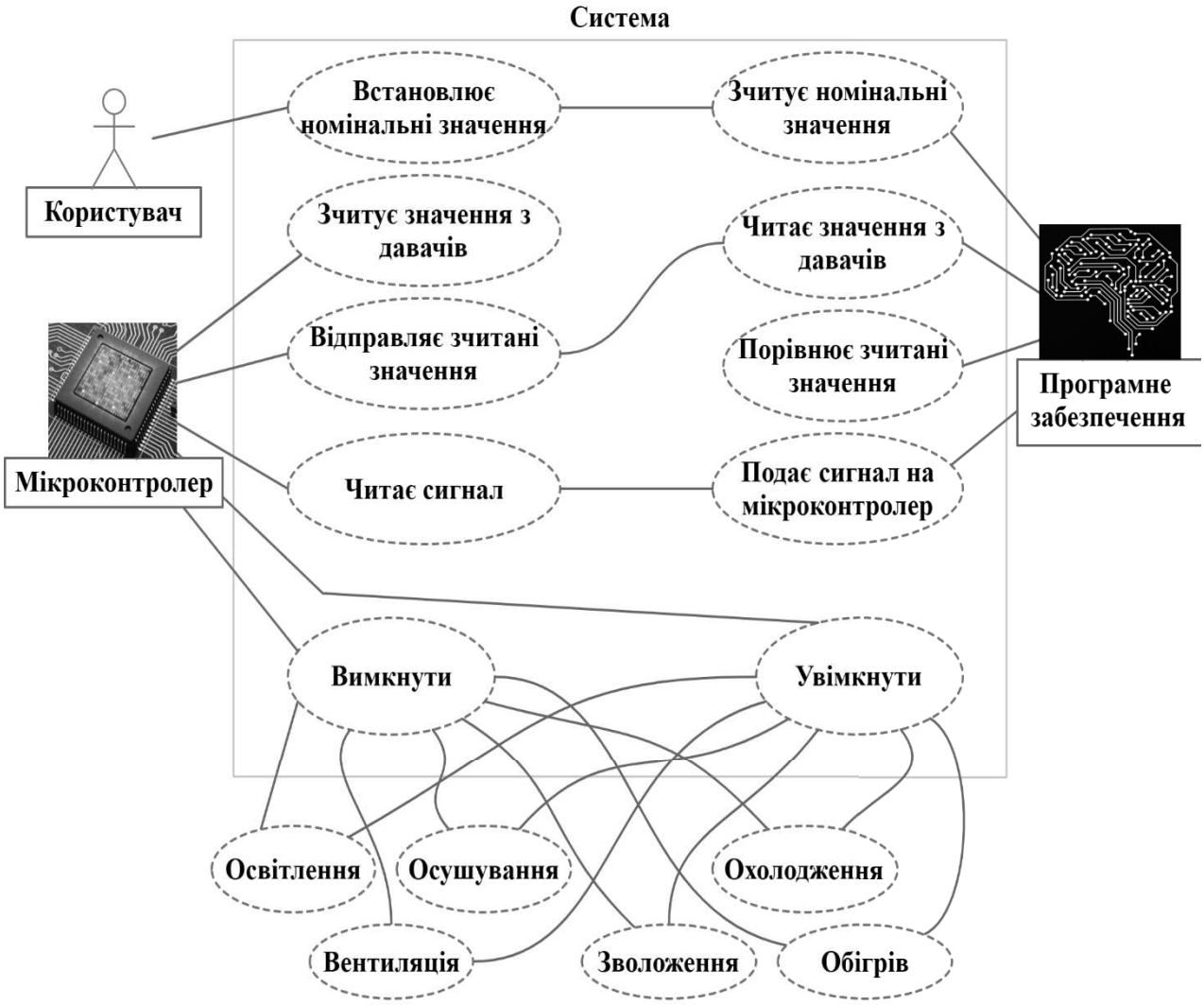


Рисунок 2.11 – Діаграма використання

Наступною побудуємо діаграму послідовності. Вона демонструє порядок дій в системі на одній часовій осі. Відповідно до алгоритму роботи системи IoT рішень для процесу сироваріння, розділимо процес на чотири часові осі. На першій розташуємо «Програмний інтерфейс», що взаємодіє з другою часовою віссю «Дані» за наступними процесами: взяти дані сенсорів, взяти оптимальні дані; а потім з третьою віссю «Мікроконтролер» за процесом – сигнал. В свою чергу «МК» взаємодіє з четвертою часовою віссю «Периферія» за процесами: охолодження, обігрів, зволоження, осушування, вентиляція, освітлення. На рисунку 2.12 зображена діаграма послідовності для нашої системи.

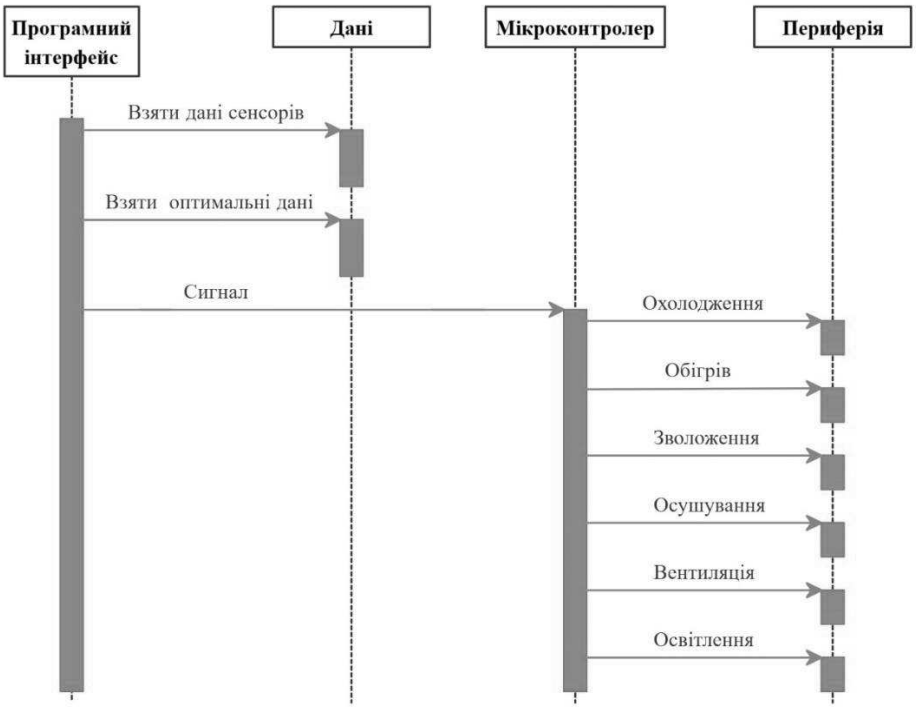


Рисунок 2.12 – Діаграма послідовності

Діаграма станів призначена для представлення різних змін між можливими станами системи і зображена на рисунку 2.13.

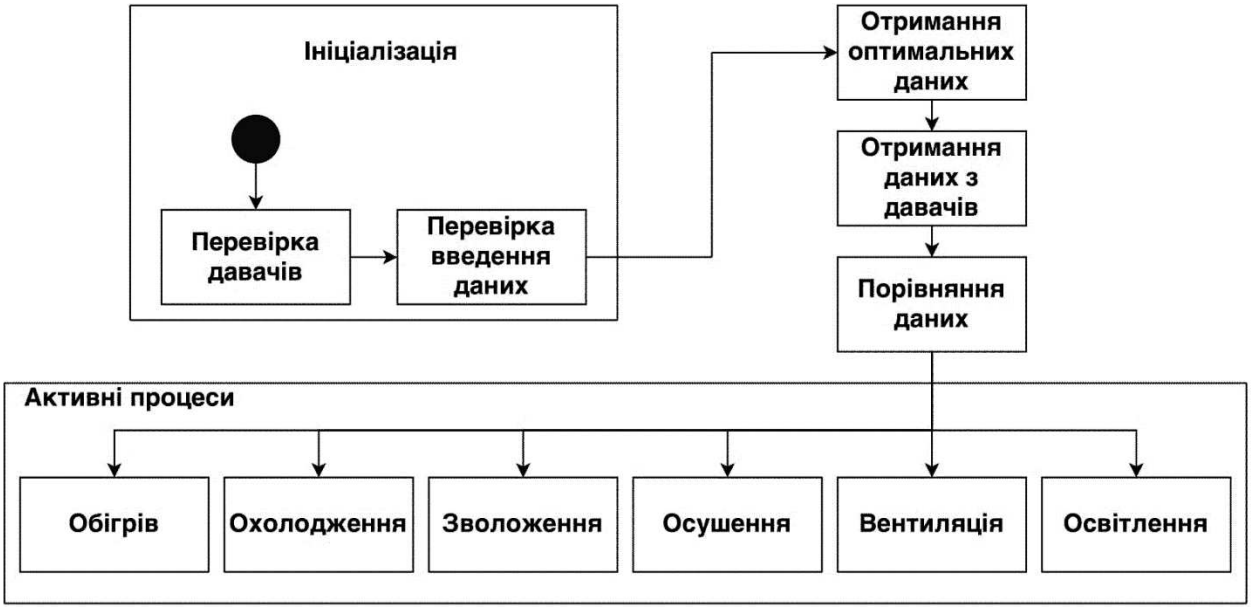


Рисунок 2.13 – Діаграма станів

Система IoT рішень для процесу сироваріння починає працювати з блоку ініціалізації даних, де відбувається перевірка підключення давачів та перевірка даних, які ввів користувач. Далі в системі відбуваються стани отримання

оптимальних даних, отримання даних з давачів та порівняння між собою цих даних. Наступним активується блок активних процесів, до якого ввійшли стани обігріву, охолодження, зволоження, осушення, вентиляції та освітлення.

2.2 Методи та засоби обробки даних

Згідно з алгоритмом роботи та обраною елементною базою у підрозділі 2.1, система IoT рішень для процесу сироваріння отримує данні з давачів, що характеризують навколишнє середовище, порівнює номінальні та реальні данні й подає відповідний сигнал.

Розглянемо інтерфейси обробки та передачі даних з давачів на мікроконтролер Arduino:

1) I2C – давач температури та вологості AM2320, освітленості GY-302;
2) ADC – давач значення електропровідності, рівня pH, рівня CO_2;
3) UART – WI-FI модуль ESP8266.

Першим проаналізуємо інтерфейс I2C (Inter-Integrated Circuits) – це послідовна шина для обміну даних, що використовує дві лінії живлення та дві лінії зв’язку: сигнал синхронізації SCL (Serial Clock) та сигнал даних SDA (Serial Data). Зазвичай вважають, що є два пристрої – ведучий (Master) та ведений (Slave). Ведучий пристрій відповідає за сигнал початку передачі даних та генерацію тактових імпульсів, а ведений – за передачу даних за запитом ведучого. Для кожного веденого пристрою існує унікальна адреса, за якою до нього може звертатись ведучий. Розглянемо принцип роботи шини I2C на прикладі часової діаграми станів сигналів SCL та SDA (рис. 2.14) [48].

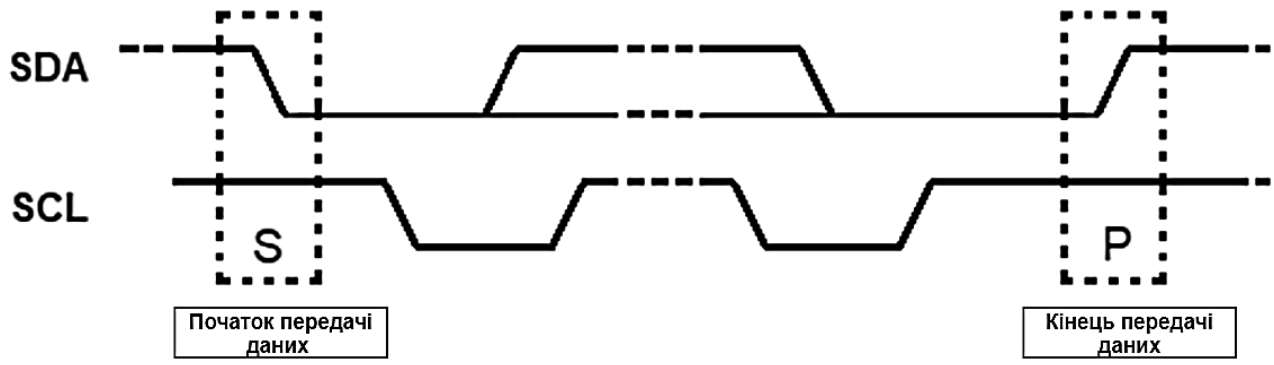


Рисунок 2.14 – Діаграма станів сигналів SCL та SDA [48]

Початковою умовою передачі даних є перехід в нижній логічний рівень (low) сигналу SDA. В цей момент часу сигнал SCL займає високий логічний рівень (high), що свідчить про готовність до передачі даних. Потім сигнал SCL також займає нижній логічний рівень, що свідчить про початок передачі даних. Протягом цього процесу ведучий пристрій генерує 9 тактових імпульсів по сигналу SCL, паралельно з цим ведений пристрій по сигналу SDA передає унікальну адресу давачу з 7 біт, 1 біт на визначення теперішнього стану (отримання чи передача даних) та ще 1 біт на підтвердження запиту (всього 9 біт). Як тільки дані передано, сигнал SDA займе високий логічний рівень, – це є умовою завершення передачі даних. А сигнал SCL також переходить у високий логічний рівень [48].

Наступним розглянемо ADC (analog-to-digital converter). Більшість інформативних сигналів, що можна отримати з навколишнього середовища є аналоговими. Вони безперервні на всьому проміжку часу та нагадують синусоїду. Виміряти такий сигнал можна величиною зміни напруги. Мікроконтролери дозволяють перевести еквівалент аналогового сигналу в цифрове значення. Для підключення аналогових давачів до мікроконтролерів Arduino та отримання з них даних використовують АЦП (аналого-цифровий перетворювач). В обраній раніше платі Arduino є 8 аналогових 10-бітних пінів та вбудований АЦП. Величина отриманих цифрових даних залежить від споживаної напруги та розрядності АЦП. Кількість можливих виміряних даних можна розрахувати за формулою 2.1 [49]:

$$n = 2^N, \tag{2.1}$$

де N – розрядність АЦП;

n – кількість можливих виміряних даних.

Розрахуємо кількість можливих виміряних даних за формулою 2.1:

$$n = 2^{10} = 1024$$

Точність виміру даних можна розрахувати за формулою 2.2 [49]:

$$D = \frac{U}{n} = \frac{U}{2^N}, \tag{2.2}$$

де U – напруга споживання;

D – точність виміру даних.

За умови, що напруга споживання рівна 5 В, розрахуємо точність виміру даних за формулою 2.2:

$$D = \frac{5}{1024} = 4.9 \text{ (мВ)}$$

Принцип роботи інтерфейсу UART (Universal Asynchronous Receiver-Transmitter) базується на двох лініях даних: прийому RX (Received Data) та передачі TX (Transmitted Data). Відповідно при підключенні двох пристроїв через послідовний порт, дані лінії з'єднуються навхрест, тобто RX першого пристрою до TX другого пристрою. Аналогічно, TX першого пристрою до RX другого. Завдяки двом лініям даних інтерфейс UART підтримує можливість обміну інформацією в дуплексному режимі. Для послідовної передачі даних використовується 10 біт: 1й та 10й біти для синхронізації, з 2го по 9й біти (1 байт) – данні. Швидкість передачі даних вимірюється в бодах (біт в секунду). Для мікроконтролера Arduino найчастіше використовують швидкість передачі даних 9600 бод, а в разі підключення модуля Wi-Fi – 115200 бод. Сигнали RX та TX відповідають логічним рівням TTL (Transistor–transistor logic) від 0 В до 5 В [50].

Для отримання інформації про навколишнє середовище з давачів необхідно запрограмувати мікроконтролер Arduino. Розглянуті вище інтерфейси обробки й передачі даних мають власні бібліотеки, за допомогою яких в середовищі розробки Arduino IDE можна створити необхідний програмний продукт.

2.3 Використання хмарних технологій для реалізації IoT рішень

Визначення терміну IoT можна описати як сукупність способів з'єднати кілька речей, що знаходяться в різних або однакових місцях через інтернет, щоб отримати інтегровану систему для розвитку продуктивності передачі інформації. Важливо ефективно збирати необроблені дані, аналізуючи та видобуваючи вихідні дані, зібрані з давачів, щоб отримати цінну інформацію та надати різні

супутні послуги. Послуги IoT відрізняються від послуг інтернету, – вимоги залежать від аспектів застосування.

Станом на 2020 рік у світі вже є 30 мільярдів IoT пристроїв, а в 2025 році їх кількість перевищить 75 мільярдів, повідомляє служба Statista [51]. Усі ці пристрої вироблятимуть величезні обсяги даних, які необхідно обробляти швидко та без втрат. Щоб задовольнити зростаючий попит на IoT рішення використовують хмарні та туманні обчислення.

Термін хмарні обчислення охоплює цілий спектр послуг, що надаються через інтернет хмарними провайдерами. Коли говорять про хмарні технології, користувачеві надаються обчислювальні ресурси (сервери, бази даних, мережі), інструменти (середовище розробки, операційна система, проміжне програмне забезпечення) та готові рішення (аналітика чи моніторинг). Хмарні обчислення мають двошарову архітектуру, зображену на рисунку 2.15 [52].

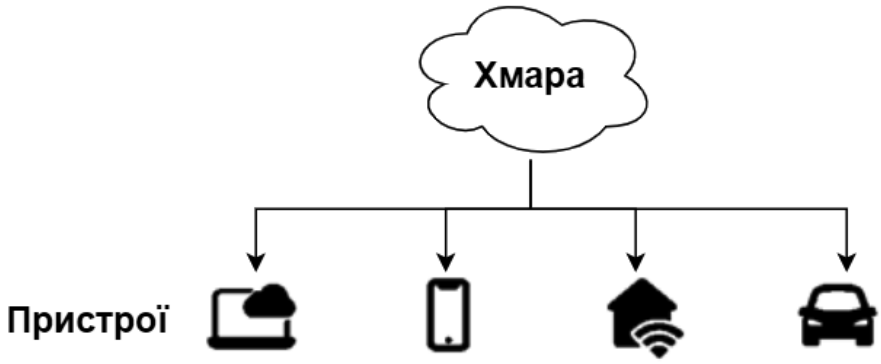


Рисунок 2.15 – Архітектура хмарних обчислень

Використання хмарних технологій означає, що вам не потрібно збирати та підтримувати власну фізичну інфраструктуру (сервери чи центри обробки даних) або встановлювати засоби розробки для створення програмного забезпечення. Усі ці необхідні технологічні послуги та обчислювальні потужності надаються сервісами хмарних послуг, які поділяються на три групи:

- IaaS (Інфраструктура як послуга) – віддалений центр обробки даних з такими ресурсами, як бази даних, обчислювальна потужність та мережеві зв'язки [52].

- PaaS (Платформа як послуга) – платформа для розробки з інструментами та компонентами для створення, тестування та запуску програм [52].
- SaaS (Програмне забезпечення як послуга) – готове програмне забезпечення, пристосоване до різноманітних бізнес-потреб [52].

Інтеграція Інтернету речей та хмарних технологій є економічно вигідним способом створення та моніторингу IoT систем. Зовнішні сервіси забезпечують необхідну масштабованість та гнучкість для управління та аналізу даних, зібраних підключеними пристроями, тоді як спеціалізовані платформи (наприклад, Microsoft Azure IoT Hub, SmartThings, AWS, Google OpenThread) надають розробникам можливість створювати системи IoT без великих інвестицій у апаратне та програмне забезпечення.

Плюси використання хмарних технологій для IoT рішень:

- Покращена продуктивність – зв’язок між давачами IoT та системами обробки даних відбувається швидше.
- Бази даних для зберігання інформації – дуже масштабований та необмежений простір для зберігання, здатність інтегрувати, агрегувати та обмінюватися величезною кількістю даних.
- Можливості обробки – віддалені центри обробки даних надають необмежені можливості віртуальної обробки.
- Зменшені витрати – вартість користування хмарними технологіями нижча за створення локального обладнання та його постійного обслуговування.

Мінуси використання хмарних технологій для IoT рішень:

- Висока затримка – все більше і більше застосунків IoT вимагають дуже низької затримки, але хмара не може цього гарантувати через суттєву відстань між клієнтськими пристроями та центрами обробки даних.
- Час простою – технічні проблеми та перебої в роботі мереж можуть виникати з будь-якої причини в будь-якій системі, що базується на якості інтернет зв’язку, і змушує клієнтів страждати від збою; багато

компаній використовують кілька каналів підключення з автоматичним переключенням, щоб уникнути технічних проблем.

- Безпека та конфіденційність – ваші приватні дані передаються через глобальну мережу інтернет, де підключені канали разом із тисячами гігабайт інформації інших користувачів; не дивно, що система вразлива до кібератак або втрати даних; проблему можна частково вирішити за допомогою використання гібридних або приватних хмар.

Туманні обчислення – це розширення хмарних обчислень за рахунок використання додаткових вузлів між підключеними фізичними пристроями та хмарою. Вони здатні забезпечувати миттєве з'єднання завдяки розміщенню вузлів безпосередньо біля підключених пристроїв. Значна обчислювальна потужність граничних вузлів дозволяє їм виконувати обчислення великої кількості даних самостійно, не відправляючи їх на віддалені сервери. Основна різниця між туманними та хмарними обчисленнями полягає в тому, що хмара це централізована система, тоді як туман – розподілена децентралізована інфраструктура. Туманні обчислення мають тришарову архітектуру, зображену на рисунку 2.16 [52].

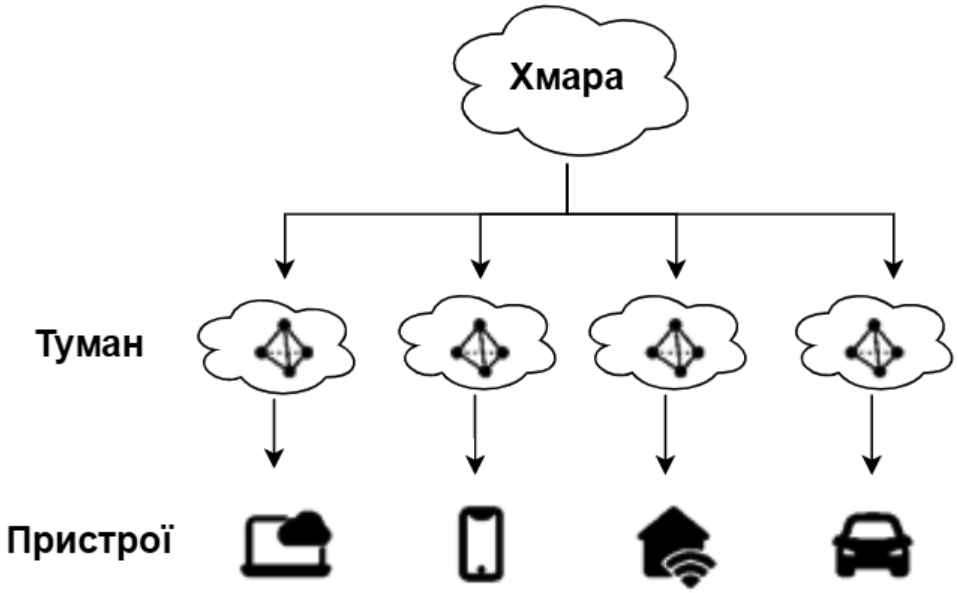


Рисунок 2.16 – Архітектура туманних обчислень [52]

Туманні обчислення виконують роль посередника між обладнанням та віддаленими серверами. Вони регулюють, яку інформацію слід надсилати на

сервер, а яку можна обробляти локально. Таким чином, туман є розумним шлюзом, який розвантажує хмари, забезпечуючи більш ефективне зберігання, обробку та аналіз даних. Слід зазначити, що мережа туману не є окремою архітектурою, і вона не замінює хмарних обчислень, а доповнює її, наближаючись якомога ближче до джерела інформації.

Плюси використання туманних обчислень для IoT рішень:

- Низька затримка – туманні вузли розташовані географічно ближче до користувачів і здатні надати миттєві відповіді.
- Відсутня проблема із пропускною здатністю – частини інформації збираються в різних точках, замість того, щоб надсилати їх разом до одного центру за одним каналом.
- Втрата зв'язку неможлива – через безліч взаємопов'язаних каналів.
- Високий рівень безпеки – адже дані обробляються величезною кількістю вузлів у складній розподіленій системі.
- Покращений інтерфейс користувача – миттєві відповіді та відсутність затримок задовольняють користувачів.
- Енергоефективність – крайові вузли використовують енергоефективні протоколи, такі як Bluetooth, Zigbee або Z-Wave.

Мінуси використання туманних обчислень для IoT рішень:

- Більш складна система – туман має додатковий рівень вузлів в системі обробки та зберігання даних.
- Додаткові витрати – компаніям слід купувати крайні пристрої: маршрутизатори, концентратори, шлюзи.
- Обмежена масштабованість – туман не такий масштабований, як хмара.

Розглянемо основні відмінності між хмарними та туманними обчисленнями (Таблиця 2.3).

1) Хмарна архітектура централізована і складається з великих центрів обробки даних, які можуть бути розташовані по всьому світу, за тисячу миль від клієнтських пристроїв. Архітектура туману розподілена і

складається з мільйонів невеликих вузлів, розташованих якомога ближче до клієнтських пристроїв.

2) Туман діє як посередник між центрами обробки даних і обладнанням і, отже, ближче до кінцевих користувачів. Якщо шар туману відсутній, хмара безпосередньо пов'язується з пристроями, що забирає багато часу.
3) У хмарних обчислення обробка даних відбувається у віддалених центрах. Обробка і зберігання в туманних обчисленнях виконуються на межі мережі поблизу з джерелом інформації, що має вирішальне значення для контролю в реальному часі.
4) Хмарні обчислення переважають туманні за рахунок обчислювальних можливостей і розміру сховищ баз даних.
5) Хмара складається з декількох великих серверних вузлів. Туман включає в себе мільйони дрібних вузлів.
6) Туман виконує короткочасний аналіз вузлів через миттєвий відгук, у той час як хмара націлена на довгостроковий глибокий аналіз через більший час зв’язку.
7) Туман забезпечує низьку затримку; хмара – високу затримку.
8) Хмарна система руйнується без підключення до Інтернету. В туманних обчисленнях використовуються різні протоколи і стандарти, тому ризик відмови набагато нижче.
9) Туман – більш безпечна система, ніж хмара, завдяки своїй розподіленій архітектурі.

Таблиця 2.3 – Порівняння туманних та хмарних обчислень

	Хмарні обчислення	**Туманні обчислення**
1	**2**	**3**
Архітектура	Централізована	Розподілена
Комунікація з пристроями	Здалеку	Безпосередньо біля пристроїв

Продовження табл. 2.3

1	2	3
Обробка даних	Далеко від джерела інформації	Близько до джерела інформації
Обчислювальні можливості	Вищі	Нижчі
Кількість вузлів	Декілька	Дуже багато
Час аналізу	Тривалий період	Короткий період
Латентність	Висока	Низька
Підключення	Інтернет	Різні протоколи та стандарти
Безпека	Нижча	Вища

Зважаючи на можливості, які відкриваються перед розробниками з використанням хмарних технологій пропоную обрати одну із доступних хмарних платформ для створення системи IoT рішень для процесу сироваріння. На перший погляд туманні обчислення більше підходять для контролю процесів сироваріння на підприємстві, ніж хмарні. Оскільки вони дають можливість швидкої обробки даних за рахунок розміщення додаткових вузлів поруч з давачами, а також мають кращу стійкість перед технічними збоями інтернет мережі. Крім того для IoT системи важливими факторами є високий рівень безпеки та енергоефективність, що відповідає цілям у поставленій меті.

Так як процес сироваріння є досить складним, вважаю за необхідність розробити власний програмний продукт, що дасть можливість гнучкості та більш детальної реалізації необхідного функціоналу.

2.4 Висновки до розділу 2

У розділі відповідно до поставленої мети описано принцип роботи пристрою та розроблено схему структурну. Вона складається з мікроконтролеру Arduino, системи управління набором реле, модулю WI-FI, давачів освітленості, температури, вологості, рівня CO_2, електропровідності та рівня кислотності pH. Відповідно до складових схеми структурної проведено вибір елементної бази,

характеристики якої повністю забезпечують виконання вимог з технічного завдання.

Побудовано UML-діаграми використання, послідовності та стану, що дозволили проаналізувати взаємозв’язок системи та користувача і дослідити функціонал IoT системи, що розробляється. Описані методи та засоби обробки даних й обґрунтовано вибір створення програмного продукту та використання хмарних технологій.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ АРХІТЕКТУРИ IOT РІШЕННЯ

У цьому розділі представлено архітектуру IoT рішення, визначено середовище передачі даних, протоколи, способи та методи захисту інформації. Проведено аналіз проблеми відсутності стандартизації систем IoT та порівняльний аналіз між сучасними платформами IoT рішень.

3.1 Архітектура проекту IoT рішення

Швидкий розвиток технологій в даний час суттєво обертається навколо Інтернету речей і очікується, що він відіграє важливу роль найближчим часом. IoT став центром уваги багатьох дослідників для виявлення проблем та викликів, що пов’язані з його дизайном та архітектурою. Однією з багатьох важливих проблем є безліч мов, протоколів та стандартів, а також відсутність згоди, що саме використовувати для різних рівнів IoT. IoT не має єдиної платформи стандартизації, – вона змінюється через неоднорідність пов’язаних речей.

Розмова про стандарти Інтернету речей розпочата на початку 2013 року, але на той момент це могло бути вже пізно. Технологічна галузь не може сидіти без діла, поки розробляються стандарти. Багато технологічних битв виграються або програються до того, як стандарти будуть навіть близькі до затвердження. Щоб розширений перегляд IoT поєднувався, потрібен додатковий час та більш глибока міжгалузева співпраця. За словами інсайдерів та досліджень Gartner, повсюдне або усталене розгортання IoT все ще перебуває в межах від п’яти до десяти років, ставлячи технологію на первинну ціль на 2021 рік.

Основною ідеєю є розробка стандарту фреймворку IoT, який охоплює всі стандарти роз’ємів, підтримуючи всі залучені технології, від пристроїв та мереж до машино-машинної взаємодії (M2M) та веб-стандартів.

IoT має багато архітектурних шаблонів. Розглянемо дві найбільш поширені та широко використовувані архітектури для бізнесу, промислових досліджень та додатків, відомі як тришарова та п'ятишарова архітектури.

Розглянемо структуру тришарової архітектури, що вважається важливою складовою IoT. Шаблон наведено на рисунку 3.1 [53].

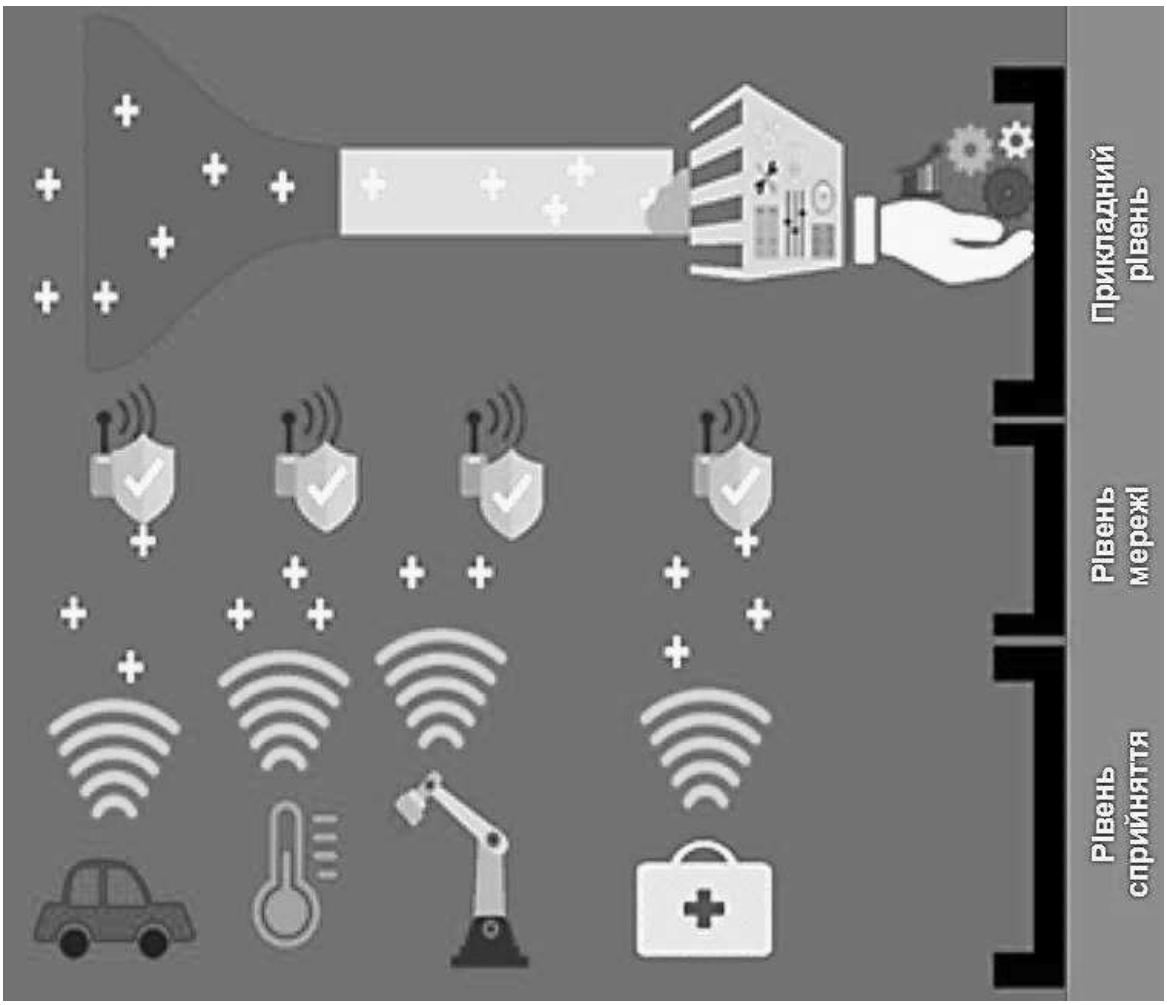


Рисунок 3.1 – Тришарова архітектура IoT [53]

Відповідно до назви, вона складається з трьох рівнів: сприйняття, мережі та прикладного. Розглянемо кожен з них. Рівень сприйняття відомий як фізичний та сенсорний рівень, оскільки він містить фізичні пристрої та вбудовані давачі відповідно. У цьому рівні вбудовані давачі збирають дані з пристроїв і надсилають їх на мережевий рівень, який в свою чергу забезпечує підключення рівня сприйняття до фізичного. Мережевий рівень повинен мати технології підключення, такі як дротове та бездротове захищене з'єднання. Прикладний рівень аналізує отримані дані та обробляє їх для надання послуг, прийняття рішень чи зворотного надсилання результатів на рівень сприйняття [53].

П'ятишарова архітектура складається з тришарової архітектури та двох додаткових рівнів, які називають рівень доступу та проміжного програмного забезпечення (рис. 3.2) [53].

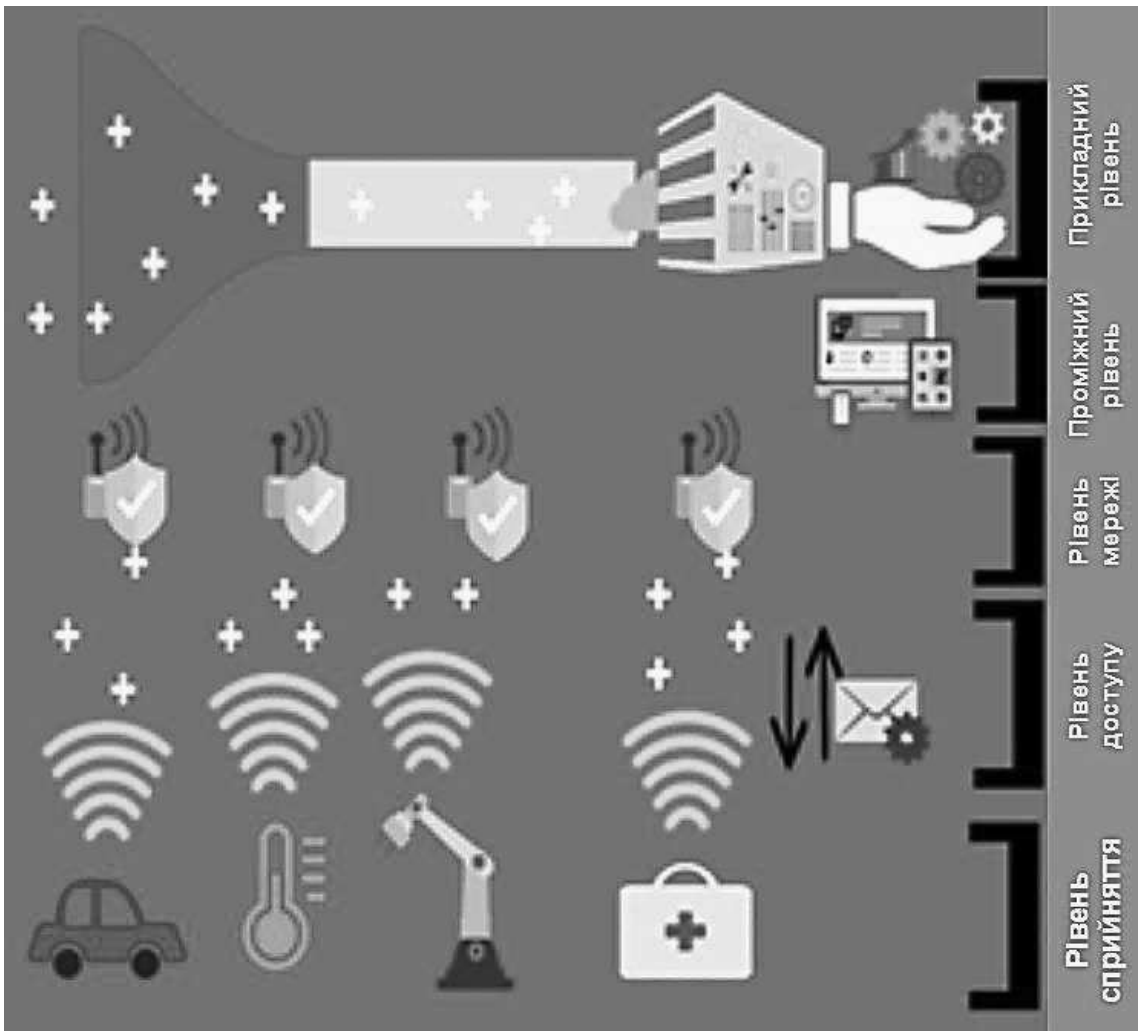


Рисунок 3.2 – П’ятишарова архітектура IoT [53]

Прикладний рівень, сприйняття та мережі виконують ті ж самі функції, що і в тришаровій архітектурі. Рівень доступу відповідає за управління зв’язком IoT в конкретному середовищі та обмін повідомленнями між об’єктами та системами. Рівень проміжного програмного забезпечення передбачає більш гнучкий зв’язок між апаратною та прикладною складовими [53].

У порівнянні з тришаровою, п’ятишарова архітектура забезпечує більшу масштабованість та конфігурованість за рахунок двох додаткових рівнів. Але зовсім різними ці дві архітектури назвати не можна, так як тришарова архітектура повністю є складовою частиною п’ятишарової.

Опишемо системну архітектуру моделі IoT рішень для процесу сироваріння. Система передбачає комунікацію машина-машина, яка може не потребувати взаємодії людини протягом процесу сироваріння, за винятком

початкового етапу налаштування. Розглянемо тришаровий тип архітектури, так як в нашому випадку він повністю задовольняє потреби розробки. Запропонована система має три домени, що являють собою домен вузла давачів (рівень сприйняття), домен зв’язку (рівень мережі) та домен доступу користувача (прикладний рівень). Концептуальна модель архітектури системи IoT рішень для процесу сироваріння зображена на рисунку 3.3.

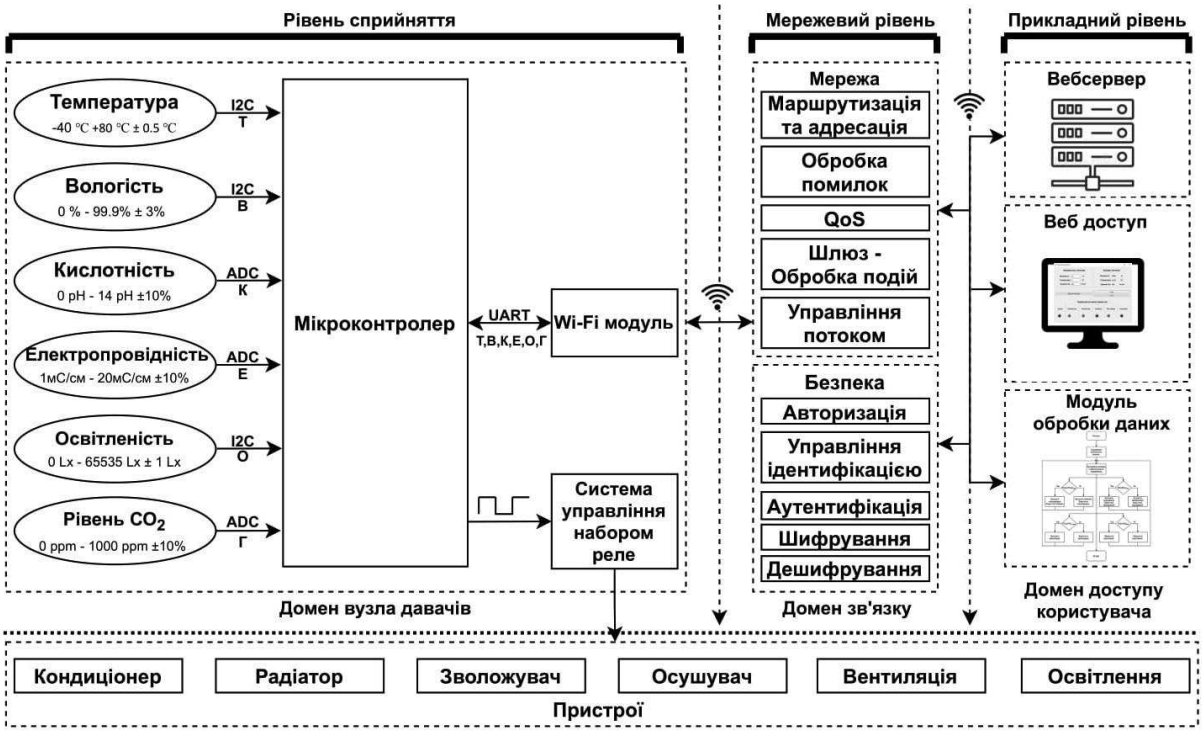


Рисунок 3.3 – Концептуальна модель архітектури системи IoT рішень для процесу сироваріння

Перший домен складається з багатьох вузлів, що можуть бути розташовані віддалено в різних місцях підприємства. Кожен вузол складається з давачів (сенсорів) як ресурсів, що передають фактичні данні. Давачі вимірюють такі фізичні значення, як: температура, вологість, кислотність, електропровідність, рівень освітлення та CO_2. Фактичні дані з давачів приймає мікроконтролер. Потім дані з вузла за допомогою шлюзу надсилаються через домен зв’язку на сервер у домен доступу користувача. На сервері дані зберігаються, а потім використовуються для аналізу стану та якості суміші та навколишнього середовища. Відповідно до результату аналізу, подається сигнал

на мікроконтролер для активації потрібного реле, що може запустити: обігрів, охолодження, зволоження, осушування, вентиляцію, управління освітленням, перехід по наступної стадії виготовлення сиру.

3.2 Комунікаційні технології та системи

3.2.1 Аналіз проблеми відсутності стандартизації IoT рішення

Існує багато проблем та завдань, які обмежують ефективність та продуктивність систем IoT, зокрема архітектури.

У 2014 році підключено 20 мільярдів девайсів, у 2020 році – 30 мільярдів, і крім того додатки розробляються з нескінченним потенціалом [54, 55]. Усі ці IoT системи створені від різних виробників та працюють на власних платформах. І навіть якщо вони виконують однакові функції, вони можуть мати кардинально різні формати даних, від чого саме зараз страждає фреймворк Hadoop: «Інтернет речей ніколи не зможе заговорити загальноприйнятою мовою» [56]. Це висловлювання узагальнює сутність IoT, що призводить до відсутності єдиної стандартизації.

За результатами голосування журналу Light Reading [57] виявилося, що найбільшою проблемою, яка стоїть перед IoT, є стандартизація, яку потрібно вирішити, оскільки вона негативно впливає на зростання взаємодії між системами IoT.

Охоплено три ключові міркування: бездротові протоколи, технології та стандарти даних [58, 59]. Дуже часто організації використовують свої стандарти, що призводить до створення пристроїв, які не можуть спілкуватися між собою.

Наявність стандартизації забезпечує взаємодію, що покращує успішну інтеграцію та обмін інформацією між розподіленими системами. Це означає, що існує потреба у стандартних протоколах та платформах, які можуть підключати різні пристрої від різних постачальників для спілкування між собою. Уніфікуючі стандарти IoT можуть також покращити загальну безпеку, де підключення пристроїв незалежно від виробника буде простіше захистити.

Розглянемо сучасні підходи до стандартизації. Першим є використання хмарної платформи Microsoft Azure з фреймворком Hadoop. Більшість

організацій використовують Hadoop як базову платформу для Інтернету речей. Щоб подолати проблему стандартизації, Hadoop потребує впровадження гібридної архітектури, яка зможе врахувати обсяг, різноманітність, швидкість та зберігання величезного об'єму даних. Цього може досягти платформа Azure з можливостями підключення мільйонів пристроїв різних марок, підтримання безлічі платформ і протоколів, використання служби Azure Data Lake для зберігання даних будь-якого розміру, форми та швидкості а також виконання всіх видів аналітики та обробки даних через безліч платформ та мов [60].

Наступний метод – використання платформи обміну повідомленнями IoT Chat, яка розроблена як доповнення до Azure, що дозволяє різним IoT пристроям взаємодіяти між собою, не залежно від постачальників. Основною метою є забезпечення простої та надійної у використанні платформи, що дозволяє розробникам IoT систем зосередитись на тому, що робить їх проекти унікальними. Ця платформа надає такі послуги: Azure API App, Azure Storage, Azure SQL та Redis Cache. На даний момент підтримується протокол передачі даних HTTP.

Розвиток технології IR Bridging на платформі Smart Thing Shield запровадив покращення можливості віддаленого керування ІЧ пультами, що відкрило набагато більше можливостей, ніж перемикання телевізійних каналів. ІЧ пульти підключаються до контролера хмари SmartThings за допомогою плати Arduino UNO з використанням протоколу ZigBee [61].

Наведені приклади сучасних підходів спрямовані на створення та розповсюдження єдиної IoT платформи. Зрозуміло, що конкуренція є досить високою. Розглянемо більш детально ефективні рішення, які пропонують світові лідери щодо вирішення проблеми стандартизації.

3.2.2 Платформа Azure IoT Hub

Microsoft Azure пропонує еталонну архітектуру IoT для підтримки широкого спектру пристроїв, середовищ, сценаріїв, шаблонів обробки та стандартів з високим рівнем безпеки та масштабованості. Azure IoT Hub - це послуга, що надається Azure IoT Suite, і це ключовий будівельний блок для

реалізації архітектури рішень IoT. Розглянемо архітектуру платформи Azure IoT Hub, що наведена на рисунку 3.4 [62].

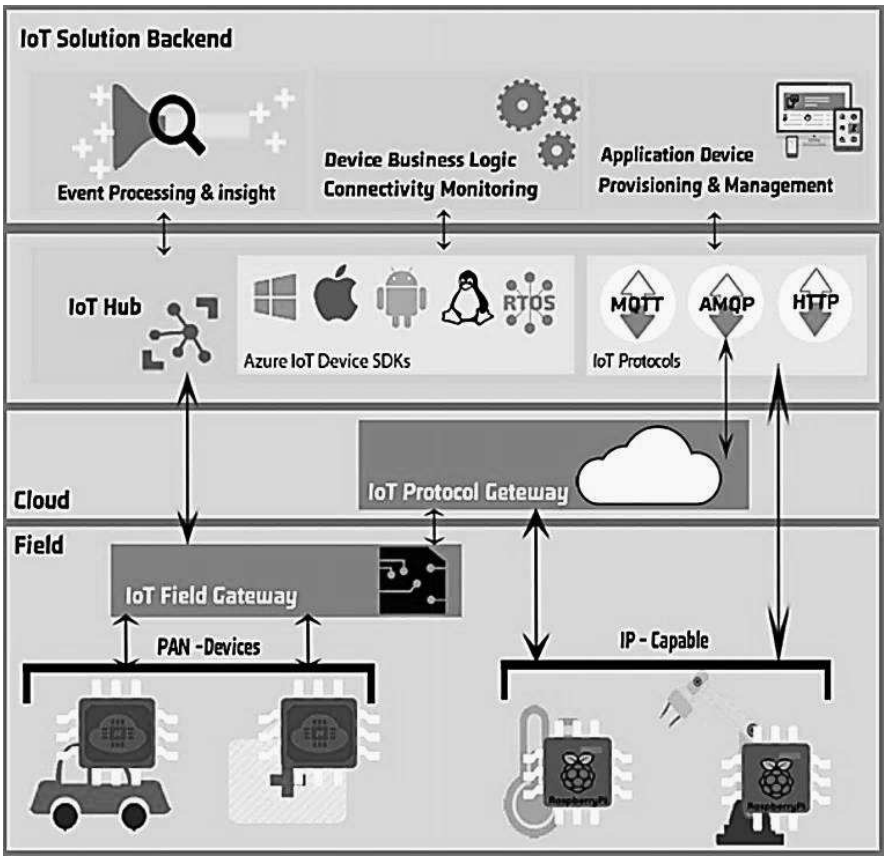


Рисунок 3.4 – Архітектура платформи Azure IoT Hub [62]

Впровадження даної архітектури дозволяє платформі Azure IoT Hub підтримувати важливі функції, які можуть подолати відсутність стандартизації в системах IoT.

1. Використання відомих платформ та протоколів

IoT Hub має високу масштабованість, щоб утримувати мільйони підключень та обробляти величезний обсяг даних. IoT Hub має великий набір бібліотек пристроїв, що підтримують різні мови, підключає існуючі пристрої та дозволяє додавати нові пристрої. SDK з відкритим кодом для декількох платформ, включаючи Windows, Linux та операційні системи реального часу, можна налаштувати та використовувати [63]. Якщо у вашому рішенні немає можливості використовувати бібліотеки пристроїв, IoT Hub надає загальнодоступний протокол, який дозволяє пристроям спочатку

використовувати протоколи AMQP 1.0, Message Queuing Telemetry Transport (MQTT v3.1.1), та HTTP 1.1. Протоколів HTTP, Advanced Message Queue Protocol (AMQP), та MQTT недостатньо для використання з усіма пристроями. Для IoT пристроїв можуть знадобитись кастомні, приватні або прикладні протоколи. Зважаючи на це, IoT Hub можна розширити для підтримки кастомних протоколів за допомогою шлюзу Cloud Protocol Gateway. Даний шлюз управляє необхідними службами на пристроях локально та відповідає за конвертування протоколів, наприклад MQTT в AMQP [64].

2. Дуплексний зв’язок із IoT пристроями

Azure IoT Hub - це повністю керована платформа, яка забезпечує безпечний та надійний двонаправлений зв’язок між мільйонами різнорідних пристроїв IoT. Кожному пристрою надається власний ключ безпеки для підключення до IoT Hub, і зберігається в реєстрі ідентифікаторів IoT Hub. Azure IoT Hub підтримує протоколи HTTP та AMQP, щоб забезпечувати зв'язок будь-яких пристроїв з підтримкою IP. Шлюз використовується для полегшення зв'язку з іншими пристроями, які не мають підтримки IP мережі (PAN). Він використовує хмарну телеметрію, щоб мати змогу зрозуміти стан пристроїв і бути готовим прийняти рішення або дію, наприклад, надсилати команди та сповіщення на підключені пристрої та відстежувати доставку повідомлень [65].

3. Моніторинг підключення пристрою

Дана функція дозволяє системам IoT-рішеннь виявляти проблеми з підключенням, отримуючи докладні журнали роботи [62].

4. Додаткові послуги

Для покращеної аналітики обробки подій та збору статистичних даних із доступних вихідних даних існують надійні служби, такі як Stream Analytics, Machine Learning, Notification Hubs, та PowerPi [65].

Переваги платформи Azure IoT Hub:

1. Підтримка декількох протоколів пристроїв: MQTT, AMQP та HTTP.
2. Визначений загальний формат повідомлень для підтримки різних протоколів та безперебійної взаємодії між ними.

3. Використання облікових даних та контролю доступу для кожного пристрою для безпечного зв'язку.
4. Надійний обмін повідомленнями від пристрою до хмари та від хмари до пристрою.
5. Просте підключення пристроїв до бібліотек для найбільш поширених мов та платформ.
6. IoT Hub дозволяє одночасно підключатись мільйонам пристроїв з простими в управлінні та налаштованими функціями.
7. Анти-спуфінг властивості – захист від хакерських атак.
8. Використовуйте те, що ви вже знаєте: Azure надає доступ до тих самих технологій, яким сьогодні довіряють і користуються мільйони підприємств, таких як Windows і Linux, віртуальні машини та контейнери та Active Directory.
9. Плавне підключення центру обробки даних до хмари: Azure надає користувачеві змогу легко створювати гібридні програми, що використовують ресурси у вашому центрі обробки даних, у постачальників послуг та в самому Azure без складних обхідних процедур та компромісів.
10. Мінімізуйте свій ризик: захист та конфіденційність даних та послуг, використовуючи життєвий цикл розробки безпеки (SDL), обов’язковий провідний в галузі гарантійний процес, який забезпечує безпеку Azure знизу вгору.
11. Протокол HTTP/1 добре працює для ледь підключених пристроїв.
12. Використання протоколів AMQP та MQTT використовуються, якщо важлива затримка доставки. Також протоколи AMQP та MQTT підтримують серверну передачу та негайне надсилання повідомлень з IoT Hub на пристрій [65, 66].

Недоліки платформи Azure IoT Hub:

1. У деяких випадках SDK не можуть підтримувати всі протоколи або всі методи автентифікації.

2. Протокол HTTP/1 неефективний як для пристроїв, так і для IoT Hub, де він не має способу реалізації push-сервера та опитування пристроїв.
3. Протоколи AMQP та MQTT є більш компактними, ніж HTTP/1, тому що є двійковими протоколами [65,66].

3.2.3 Платформа SmartThings

Платформа SmartThings – одне з найкращих доступних рішень для досягнення стандартизації, спроектованої та розробленої компанією Samsung. Вони створили розумний домашній вузол, який працює як конвертор, придатний для величезної кількості стандартів для різних систем розумного будинку, де він здатний змусити їх працювати разом. Крім того, це дозволяє контролювати будинок з будь-якої точки світу, і це ефективний спосіб перетворити ваш будинок на розумний будинок. Розглянемо архітектуру платформи SmartThings що наведена на рисунку 3.5 [67].

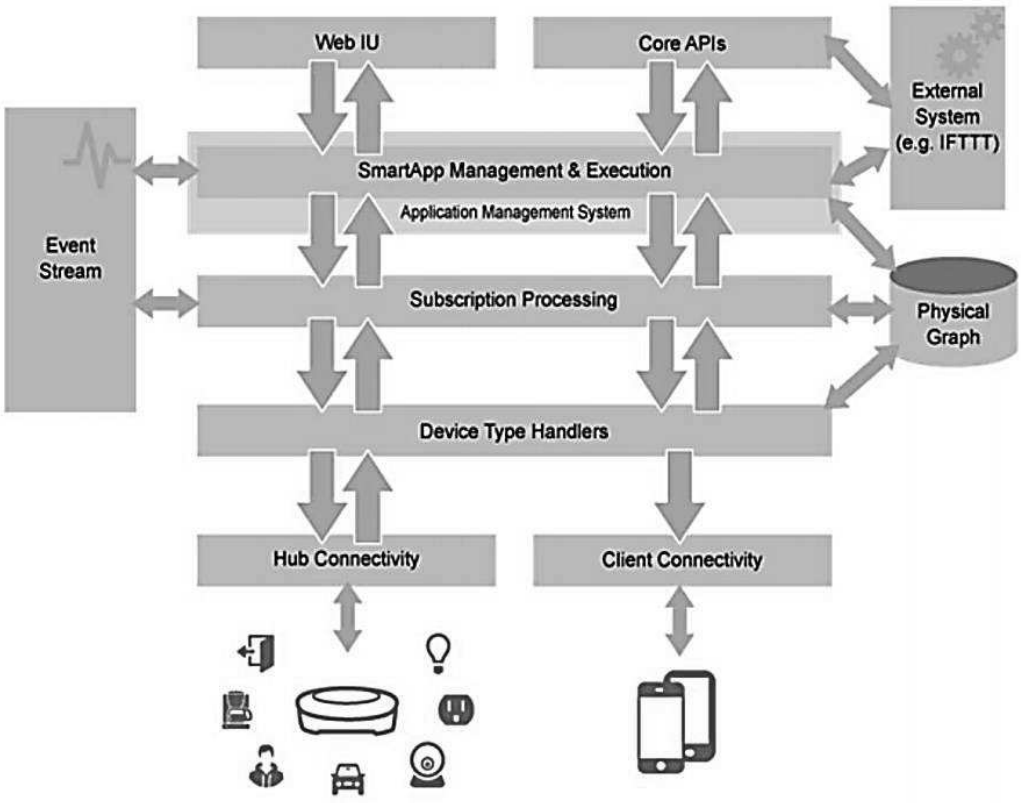


Рисунок 3.5 – Архітектура платформи SmartThings [67]

Ключовою функцією SmartThings є те, що системою можна керувати з багатьма пристроями для автоматизації будинку. Він забезпечує інтегрований

підхід, де може обмежити використання певного протоколу, пристрою чи технології. Значення терміну сумісності зосереджено на таких стандартах, як IP/Wi-Fi, ZigBee та Z-Wave, так як з ними можна працювати з сотнями пристроїв.

Пристрої: основні складові інфраструктури; вони встановлюють зв'язок між реальним світом та системою SmartThings.

Хаб: слід негайно підключити його до своєї мережі; встановлює зв'язок між хмарою SmartThings, всіма під'єднаними давачами та мобільним додатком.

Рівень управління підключенням: встановлює зв'язок між SmartThings Hub, клієнтськими пристроями із серверами. Існує два інтерфейси цього рівня:

1. Підключення клієнта – підключає клієнтські пристрої до хмари.
2. Підключення хабу – підключає хаб до хмари

Обробники типів пристроїв: визначають типи пристроїв.

SmartApps: на платформі SmartThings, коли створюються події, SmartApps розробляються з підписками на визначені події. Коли тригер ініціює записані процеси або кінцеві точки через підписки, SmartApp сповіщує про це. Він перестане працювати, коли завдання буде виконано.

Управління передплатою: Мета рівня управління передплатою полягає в координації подій, які активуються обробниками типу пристрою.

Веб-інтерфейс: він розміщений поверх усіх інших шарів, щоб забезпечити керування пристроями, концентраторами та багатьма установками системи SmartThings.

Вбудований симулятор: Він може імітувати будь-який тип пристроїв, тому не потрібно мати фізичних пристроїв [68, 69].

Переваги платформи SmartThing

1. Простота інсталяції та налаштування.
2. Сумісність: вбудовані давачі, що використовують однакові стандарти, SmartThings базується на продуктах Z-wave (наприклад: одна мова спілкування) і Zigbee (наприклад: може говорити іншою мовою), що забезпечує широкий спектр продуктів, які можна додати на систему.

Крім того, він підтримує принцип If This then That (IFTTT), що дозволяє автоматизувати та підключатись до всього, що можна собі уявити.

3. Працює зі сторонніми пристроями, такими як Hue.
4. Тривалий час автономної роботи давачів.
5. Пропонує багатофункціональні давачі.
6. Немає обмежень щодо кількості давачів, які можна додати [70].

Недоліки платформи SmartThing

1. Хаб не підтримує власну камеру або чистий протокол для підключень, і для цього потрібне дротове підключення Ethernet.
2. Якщо хаб знаходиться в автономному режимі більше 5 секунд, вся система перестає працювати, усі програми, усі обробки всіх станів заморожені.
3. Підключення хабу до програми відбувається відносно повільно.
4. Оновлення та інтеграція з SmartThings Hub непрості та нелегкі[70].

3.2.4 Платформа OpenThread

Автоматизація потоків даних для розумних будинків зазнає великої проблеми - відсутність стандартизації мережевих протоколів. Необхідність для спілкування пристроїв між собою, що говорять на одній і тій же мові бездротового зв'язку з однаковою частотою [71, 72]. Компанія Google розпочала пошук відкритого набору стандартних протоколів для домашньої автоматизації, де датчики, пристрої та механізми всередині будинку можуть обмінюватися даними. Вони виявили, що такого протоколу не існує. Проблема існуючих протоколів пов'язана з відсутністю стандартизації в Інтернеті речей, коли багато пристроїв від різних постачальників не можуть взаємодіяти між собою. Одне з рішень, запропонованих Google, відоме як Thread. Thread – це відкритий і стандартний набір мережевих протоколів, метою якого є створення стандартного зв'язку між підключеними пристроями [71].

Thread – це протокол бездротової мережі на основі IP із низьким споживанням енергії та високим рівнем безпеки, завдяки чому підключення

пристроїв стає кращим, ніж інші технології, такі як Bluetooth та Wi-Fi. Технологія Thread без проблем підключить пристрої вдома [71,72].

Протокол Thread долає відсутність стандартизації:

1. Призначений для підтримки широкого спектру товарів для дому: контроль доступу, прилади, клімат-контроль, енергоменеджмент та безпека.
2. Протокол відкритого стандарту: потік покладається на протокол бездротової мережі на основі IP, він передає пакети IPv6 через 6LowPAN (персональна мережа низького енергоспоживання).
3. Простий у використанні споживачами: встановлення мережі Thread є інтуїтивно зрозумілим та простим для користувачів. Окрім того, користувачі можуть додавати, видаляти та авторизувати пристрої в мережі, використовуючи смартфони чи комп'ютери за кілька простих кроків [71,72].

Розглянемо архітектуру платформи OpenThread (рис. 3.6) [71].

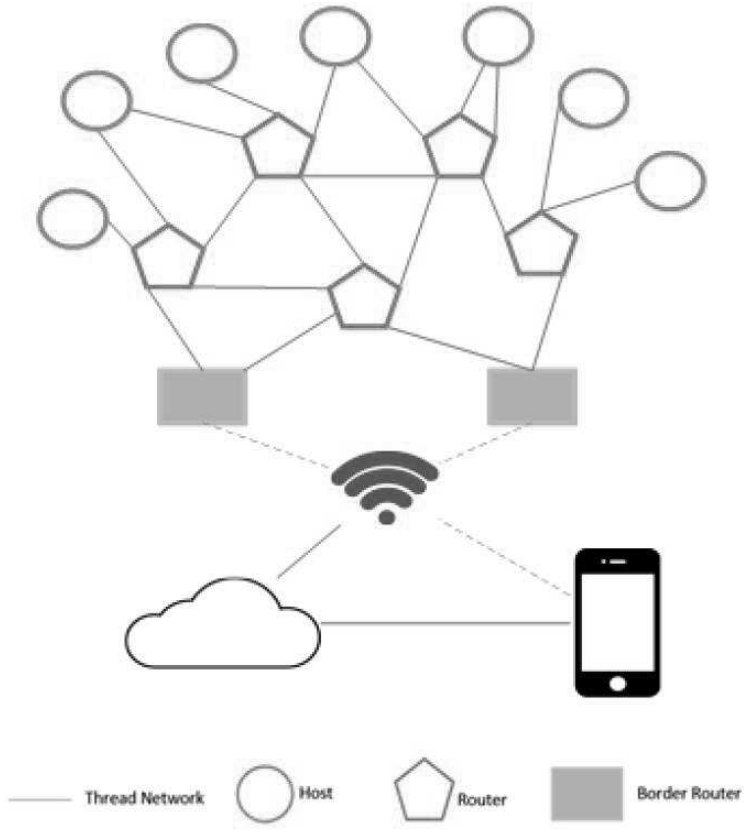


Рисунок 3.6 – Архітектура платформи OpenThread [71]

Архітектура Thread складається з сітчастої мережі і будується знизу вгору, щоб підтримувати комунікацію між пристроями. Потік на основі 6LoPAN (Low power Wireless Personal Area Networks) дозволяє надсилати та приймати пакети IPv6 через бездротові мережі. Це дає можливість навіть невеликим пристроям підключатися до Інтернету і, отже, створювати Інтернет речей. Thread розроблений з новою архітектурою безпеки, щоб зробити його надійним та простим у додаванні та видаленні продуктів, а також підтримує більше 250 пристроїв у мережі. У мережі Thread існує три типи пристроїв: крайній маршрутизатор, маршрутизатор та хост або кінцевий пристрій [72].

Крайній маршрутизатор: особливий тип маршрутизатора, який забезпечує підключення від мережі 802.15.4 до сусідніх мереж на інших фізичних рівнях (наприклад Wi-Fi та Ethernet). Він надає послуги для пристроїв усередині мережі 802.15.4, а також охоплює маршрутизацію для роботи поза мережею. У мережі Thread може бути один або кілька крайніх маршрутизаторів, що покращує відмовостійкість (рис 3.6). Коли крайній маршрутизатор падає, вся мережа потоку опускається. Отже, для гнучкості слід використовувати кілька крайніх маршрутизаторів. Коли крайній маршрутизатор не працює, інший все одно може працювати і Thread мережа продовжує функціонувати.

Маршрутизатор: надає послуги маршрутизації мережевим пристроям. Крім того, це забезпечує безпеку та послуги для пристроїв, які намагаються приєднатися до мережі. Він не призначений для режму сну, тоді як хости або кінцеві пристрої можуть переходити в режим сну.

Хост або кінцевий пристрій: Хости – це давачі та кінцеві пристрої, які підключені до мережі Thread. Вони можуть спілкуватися лише через батьківський маршрутизатор і не можуть пересилати повідомлення на інші пристрої.

Розглянемо відкритий стандарт Thread Network Stack для безпечного, малопотужного, економічного, бездротового зв'язку від пристрою до пристрою (D2D). Він розроблений спеціально для підключення побутових приладів, де

бажана мережа на основі стеку IP може використовувати різноманіття прикладних рівнів (рис.3.7) [73].

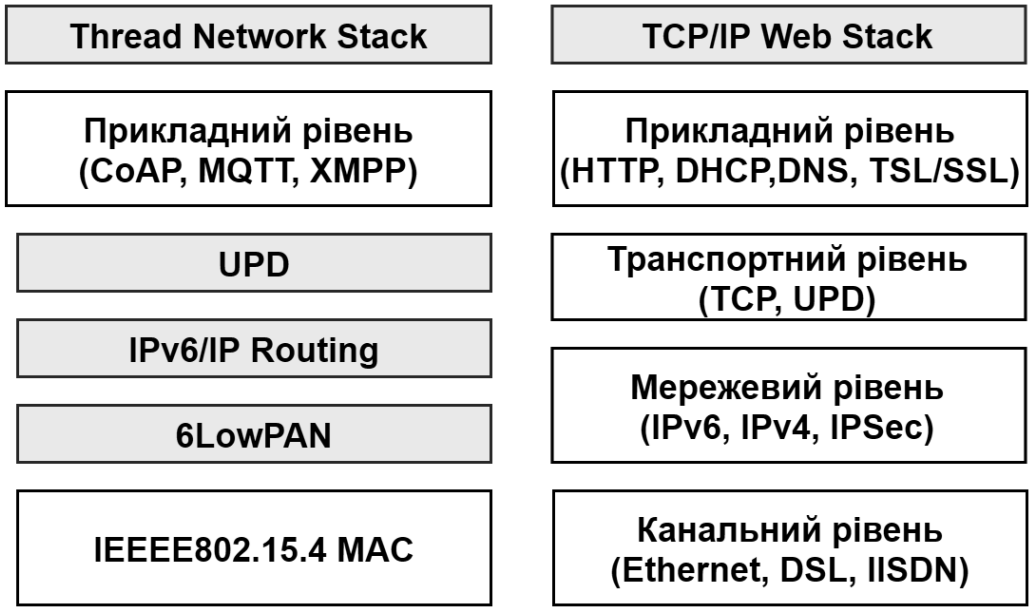


Рисунок 3.7 – Thread Network Stack [73]

1. Стандарт зв'язку IEEE 802.15.4 як протокол підключення: розроблений для бездротових персональних мереж із низькою швидкістю та низькою потужністю.
2. Підключення 6LoWPAN та IPv6: IPV6 дозволяє пристроям взаємодіяти з іншими пристроями, отримувати доступ до хмари або взаємодіяти з користувачем через мобільні програми Thread. 6LoWPAN – це відкритий стандарт, який розглядається як каталізатор, що дозволяє відправляти та приймати пакети IPv6 через реле IEEE 802.15.4 на бездротових PAN низької потужності. На рисунку 3.7 видно, що 6LoWPAN розміщується поверх IEEE 802.15.4 у мережевому стеку; він визначає механізми стиснення заголовка та інкапсуляції, які дозволяють надсилати та приймати пакети IPv6.
3. Використання UDP (User Datagram Protocol): для обміну повідомленнями між пристроями, зазвичай використовується в IoT. UDP є кращим ніж TCP для використання у веб-стеку [73].

Переваги платформи OpenThread

1. Просте використання клієнтами.

2. Завжди захищена.
3. Енергоефективна.
4. Відкритий протокол, який підтримує IPv6.
5. На основі надійної сітчастої мережі без єдиної точки поломки.
6. Працює через стандарт 802.15.4.
7. Розроблена для підтримки широкого спектру товарів для дому: побутова техніка, контроль доступу, клімат-контроль, енергоменеджмент, освітлення та безпека.
8. Масштабована.
9. Висока сумісність.
10. Менш дороге та простіше обладнання у використанні та обслуговуванні [74].

Недоліки платформи OpenThread

Враховуючи низьку потужність, існують два стандарти, що широко адаптовані промисловістю: Thread та Bluetooth. Дві технології мають свої позитивні сторони. Bluetooth існує вже давно, зрілий і широко використовується в обчислювальній техніці вже деякий час. Розглянемо вбудовані смартфони та переносні пристрої. Технологія Bluetooth в основному є на кожному смартфоні та на кожному переносному пристрої (розумні годинники, браслети, взуття тощо). Якщо певні розумні домашні пристрої підключаються через Bluetooth, ми можемо взаємодіяти з ними більш автономно та безперебійно. Наприклад, ви хочете керувати кондиціонером, що залежать від температури тіла, виміряної за допомогою переносного пристрою. Якщо кондиціонер використовує Bluetooth, він буде безпосередньо зв’язуватися з пристроєм, який можна носити, але коли він використовує технологію Thread – спочатку зв’язок проходитиме через шлюз, тож це зробить інтеграцію більш складною та менш плавною [74].

3.2.5 Порівняння між платформами IoT рішень

Для систематизації отриманої інформації після аналізу передових платформ IoT рішень проведемо всебічне порівняння між ними. Перше

порівняння між платформами Microsoft Azure та SmartThings, що наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння між Microsoft Azure та SmartThings

	Microsoft Azure	**SmartThings**
1	**2**	**3**
Використання	Зазвичай використовується в галузях для побудови професійних проектів.	Використовується для розробки проектів Розумного дому.
Протоколи	HTTPS, AMQP, MQTT, AMQP через WebSockets. Можливість запустити шлюз протоколу Azure IoT у хмарі або локально та розширити його	ZigBee, Z-Wave, Cloud-to-Cloud, Local Area Network.
Підключені пристрої	Підключайте, налаштовуйте і керуйте мільйонами пристроїв IoT.	Сумісність з сотнями пристроїв від постачальників.
Платформи і Операційні системи	Платформи ОС Linux / Unix, Android, Windows. Arduino. Mbed. TI-RTOS.	Працює тільки на смартфонах і планшетах, встановивши застосунок SmartApp. Підтримка: ОС Android, iOS. Має можливість підключитися до будь-якого з 240 каналів IFTTT і створити інструкцію IFTTT (If This Thin That), щоб зробити ваш будинок розумнішими.
Мови програмування та бібліотеки	Бібліотеки C. Бібліотеки Node.js. Бібліотеки Python. Javascript. C#.	Мова програмування Groovy.

Продовження табл. 3.1

1	2	3
Складність налаштування	Потрібно багато навичок в залежності від ступеня складності і ідеї проекту.	Менш складний, ніж Azure.

Розглянемо наступне порівняння між протоколами OpenThread та ZigBee, щоб підкреслити ключові відмінності, сильні та слабкі сторони цих рішень. Багато розробників бажають використовувати протокол ZigBee у своїх проектах розумних будинків, а не Thread. Одина із причин - підтримка SmartThings Hub протоколу ZigBee. Thread і ZigBee майже ідентичні, працюють з однаковим протоколом низького рівня MAC. Більше того, вони одночасно є відкритими стандартами та поєднують у собі локальні мережі. Крім того, вони націлені на порівнянні типи програм, подібні до розумного будинку. Тим не менше, деякі основні відмінності потребують обговорення при порівнянні двох мереж. Вони об'єднані у два основні аспекти – мережевий рівень та прикладний рівень. На мережевому рівні Thread використовує 6LoWPAN, який надає кожному вузлу IP-адресу, де ZigBee використовує крайній маршрутизатор, якому важко комунікувати при хмарному підключенні. Розглянемо таблицю 3.2.

Таблиця 3.2 – Порівняння між Thread та ZigBee

<table>
<tr><th>Модель OSI</th><th>6LoWPAN</th><th>ZigBee</th></tr>
<tr><td>Прикладний рівень</td><td>COAP, MQTT, HTTP, JSON, WebSocket</td><td>ZCL, ZDO</td></tr>
<tr><td>Транспортний рівень</td><td>TCP, UDP, ICMP</td><td>APS</td></tr>
<tr><td>Мережений рівень</td><td>IPv6, RPL</td><td>AODV, MTO/Source routing</td></tr>
<tr><td rowspan="2">Канальний рівень</td><td>6LoWPAN</td><td rowspan="3">IEEE802.15.4</td></tr>
<tr><td rowspan="2">IEEE802.15.4</td></tr>
<tr><td>Фізичний рівень</td></tr>
</table>

На прикладному рівні ZigBee визначає, як програми підключаються до нього та запускають його. Однак Thread не обмежує прикладний рівень, що

забезпечує універсальний спосіб спілкування з пристроями та кінцевими вузлами. Крім того, він пропонує можливість спілкуватися з кількома програмами, що робить Thread кращим, ніж ZigBee, з метою стандартизації [75].

Відсутність стандартизації негативно позначається на масштабованості та безпеці IoT систем. Цю проблему можна вирішити, запропонувавши інтегровану платформу Azure IoT Suite в межах IoT Azure Hub, де вона може масштабуватися для розміщення мільйонів існуючих пристроїв та додавання нових, підтримуючи різні мови програмування, загальнодоступні протоколи та операційні системи. Крім того, вона може обробляти, аналізувати та зберігати дані на основі зміни їх обсягу, різноманітності та швидкості, щоб отримати повне середовище розробки проектів та продуктів IoT.

Іншим запропонованим рішенням є платформа SmartThings; вона розглядається як платформа для розумних будинків, де її простіше встановити, і використовується для розробки менш складних проектів, ніж те, що може зробити Azure, підтримуючи меншу кількість сумісних пристроїв та протоколів, таких як ZigBee та Z-Wave, її проекти розробляє лише одна мова програмування.

Порівняння між Thread та ZigBee підтверджує можливість подолати відсутність проблеми стандартизації. Платформа SmartThings планує підтримувати протокол Thread, протокол передачі даних по мережі, який запропонований NEST та іншими, а також центр наступного покоління SmartThings.

3.3 Аналіз захищеності інформації та методів і способів (протоколів) для її кіберзахисту.

Технології інтернету речей стикаються з проблемами щодо створення масштабованих, безпечних та захищених систем, пов’язаних з обмеженнями ресурсів. Оскільки технології IoT взаємодіють з людьми, машинами та середовищем, збої в роботі можуть привести до дуже серйозних наслідків. Цей факт робить безпеку IoT особливо важливою. Забезпечення гарантій безпеки (наприклад захист від вторгнення або несанкціонованого доступу) можуть допомогти запобігти порушенню безпеки зловмисниками. Такі заходи безпеки,

як захист зони польоту Airbus, що забороняють пілотам виконувати ризиковані маневри, можуть допомогти запобігти нанесенню шкоди правопорушниками [76].

Безпека традиційного інтернету підвищена завдяки добре розробленим заходам безпеки, таким як набори протоколів SSL/TLS (Secure Socket Layer/Transport Layer Security) [77]. Однак технології IoT мають унікальні характеристики, які відрізняють їх від традиційного інтернету, що обумовлює особливі вимоги до безпечної мережевої архітектури IoT. Через специфічні вимоги широко використовувані заходи мережевої безпеки не адаптуються окремо до IoT. Існують пристрої IoT з обмеженими ресурсами, і очікується, що їх кількість буде швидко рости. Отже, захищена архітектурна мережа повинна добре працювати з нестабільним підключенням і пристроями з обмеженими ресурсами у великому масштабі. Однак з такими заходами безпеки, як TLS на основі сертифікатів, наданих центрами сертифікації, буде дуже складно контролювати авторизацію величезної кількості пристроїв з обмеженнями ресурсів. Безпечна мережева архітектура для IoT повинна забезпечувати гарантії безпеки на рівні, порівнянному з TLS, принаймні, для деяких пристроїв. Тому недостатньо просто адаптувати полегшені рішення безпеки для бездротових сенсорних мереж (WSN), які йдуть на компроміс з точки зору гарантій безпеки.

3.3.1 Вимоги до безпеки IoT рішень

Розглянемо вимоги до безпеки мережевої архітектури та систем управління ключами для і систем IoT рішень.

Часта авторизація та аутентифікація: внаслідок важливості деяких пристроїв в IoT потрібно сувора авторизація та аутентифікація. Машини працюють на більш високих швидкостях, ніж люди, а фізична доступність пристроїв робить їх більш уразливими. Динамічно змінюються ситуації, що виникають в результаті мобільності, можуть змінити авторизацію пристроїв. Більш того, часта авторизація може надати способи обмежити шкоду в разі злому критичних пристроїв.

Автоматична взаємна аутентифікація: користувачам не потрібно запам'ятовувати паролі для великої кількості пристроїв. Таким чином, пристрої IoT повинні мати можливість аутентифікувати себе без втручання користувача.

Переривчасте підключення: мобільність пристроїв дозволяє їм змінювати мережеве середовище, в якому вони працюють, що може привести до нестабільного підключення. Однак ми не можемо поставити під загрозу безпеку, коли мережеве з'єднання нестабільно, тому ми повинні мати можливість обробляти переривчасте підключення пристроїв.

Динамічна реєстрація об'єктів: на відміну від традиційного Інтернету, Інтернет речей включає пристрої з більш коротким життєвим циклом, ніж звичайні комп'ютери. Мобільні пристрої можна динамічно додавати та видаляти з систем автентифікації/авторизації. Отже, слід керувати додаванням та видаленням об'єктів в архітектурі захищеної мережі.

Підтримка функцій масштабованості: Існує безліч підходів до масштабованості мережі, які приносять користь пристроїв IoT, включаючи протоколи публікації-підписки [78], такі як MQTT [79]. Архітектура безпеки повинна вміти працювати разом з цими функціями масштабованості.

Врахування обмежень ресурсів: Деякі пристрої в IoT мають обмежені ресурси; наприклад, пристрої, що працюють від акумуляторів, мають обмежений енергетичний запас для обчислень та зв'язку. Посилені заходи безпеки зазвичай витрачають більше енергії; однак надмірне споживання енергії може зашкодити доступності. Отже, повинна бути можливість збалансувати безпеку та споживання енергії.

Конфіденційність: пристрої, такі як мобільні телефони, можуть легко збирати і передавати особисту інформацію своїх користувачів. Отже, міра безпеки також повинна забезпечувати захист конфіденційності користувача.

3.3.2 Заходи безпеки IoT рішень

Криптографічні протоколи Secure Socket Layer/Transport Layer Security (SSL/TLS), або просто TLS [80], забезпечують безпеку Інтернету. Для аутентифікації протокол TLS використовує сертифікати, що зазвичай надаються

центром сертифікації (ЦС). Однак, це не може бути найкращим варіантом для IoT систем через витрати на ЦС, щоб управляти величезною кількістю сертифікатів. Щоб масштабувати TLS з центрами сертифікації, проект Let's Encrypt запускає безкоштовні і автоматизовані центри сертифікації. Проте, для пристроїв з обмеженими ресурсами буде занадто вимогливо мати великі сертифікати і виконувати дорогі в обчислювальному відношенні операції з асиметричним ключем (криптографія з відкритим ключем) для кожного з'єднання TLS [81].

У статті [82] пропонують систему аутентифікації для Інтернету речей з використанням протоколу DTLS [83], варіант датаграми TLS. Протокол обмеженого застосування (CoAP) [84] захищений DTLS. Такий підхід може краще працювати з пристроями з обмеженими ресурсами, ніж TLS. Однак DTLS все ще базується на з'єднаннях «point-to-point», таких як TLS, що робить складним завдання захищати «one-to-many» комунікацій, поширених у IoT [78]. Крім того, сертифікат, який використовується в TLS і DTLS, містить унікальне значення для об'єкта. З’являються ризики розкрити особистість організації, що може привести до потенційної загрози конфіденційності.

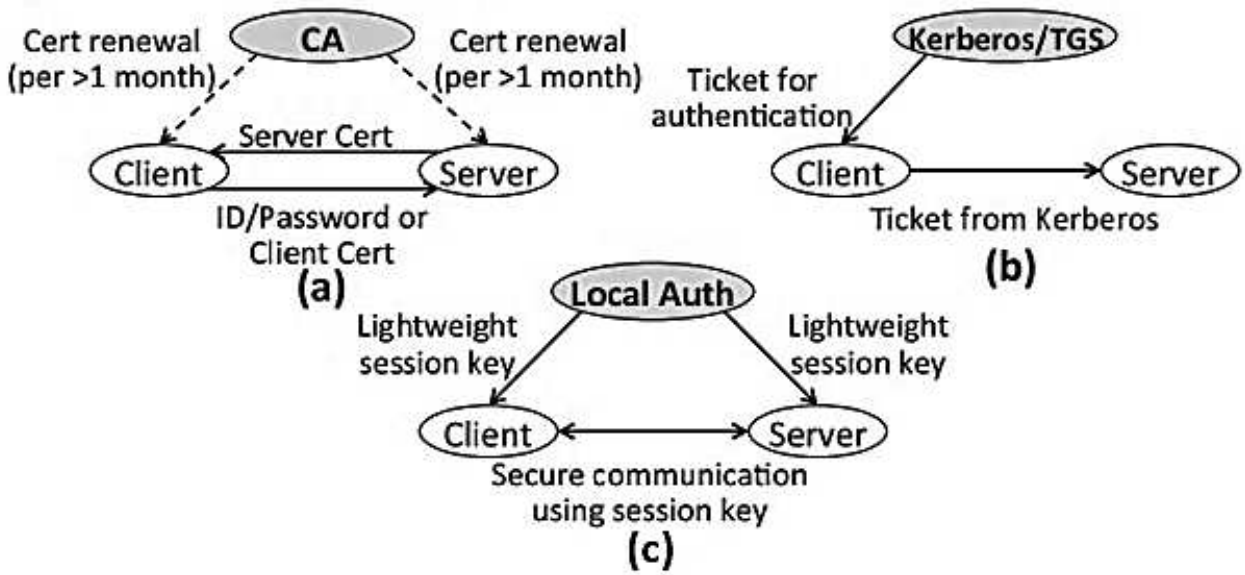


Рисунок 3.8 – Потоки аутентифікації/авторизації різних підходів: (a) ЦА (центр сертифікації) та Certs (сертифікати); (b) Kerberos (служба автентифікації)/TGS (надання ключів) (c) Локальна авторизація Auth [85]

Іншою проблемою підходів, заснованих на сертифікатах, є скасування аутентифікації. Як показано на рис. 3.8 (a) [85], центр сертифікації бере участь тільки у видачі сертифікатів, а клієнт і сервер аутентифікують один одного за допомогою сертифікатів, доки сертифікати є дійсними. Термін дії сертифікатів, як правило, перевищує декілька місяців через накладні витрати на управління сертифікатами, особливо для поновлення сертифікатів. Отже, складно скасувати аутентифікацію об'єктів, що використовують сертифікати. Це може бути потенційною загрозою безпеки критичних компонентів Інтернету речей в разі їх злому.

Для аутентифікації об'єктів система аутентифікації Kerberos [86] видає тимчасові ключі через свою службу видачі ключів (TGS), як показано на малюнку 3.8 (b) [85]. Таким чином, Kerberos забезпечує централізований контроль над терміном дії аутентифікації, вирішуючи проблеми скасування. Однак такі системи вимагають втручання користувача, наприклад для введення паролів. Це ускладнює підтримку автоматичної взаємної аутентифікації пристроїв IoT.

Існують розширення для Kerberos [87], які використовують криптографію з відкритим ключем для автентифікації, щоб замінити втручання людини. Однак навіть із цими розширеннями клієнти з переривчастим підключенням можуть зіткнутися з проблемами автентифікації, коли вони не підключені до Kerberos/TGS. Незважаючи на те, що ключі кешування можуть вирішити проблему періодичного підключення, це створює ще одну проблему – дозволити компрометованому клієнту з кешованими ключами автентифікуватися на сервері. Це тому, що квиток доставляється на сервер клієнтом, а не Kerberos/TGS, як показано на рис. 3.8 (b) [85], і сервер довіряє клієнту до тих пір, поки ключ діє.

Потоки перевірки автентичності для підходів на основі сертифікатів і системи перевірки достовірності Kerberos призначені для Інтернету з комп'ютерами загального призначення. Хоча ці підходи виявилися успішними для традиційного Інтернету, потоки аутентифікації, показані на рис. 3.8 (a) і (b)

[85], не можуть вирішити деякі з проблем безпеки і масштабованості, пов'язаних з IoT. Тому розглянемо мережеву архітектуру, яка має потоки автентифікації та авторизації, як показано на рис. 3.8 (c) [85].

У запропонованій мережевій архітектурі локальний об'єкт авторизації Auth призначає полегшені ключі сеансу об'єктів, які беруть участь в обміні даними. Об’єкт Auth контролює термін дії ключів сеансу і покриває авторизацію, а також аутентифікацію. Це можливо, тому що Auth знає про контекст зв'язку зареєстрованих об'єктів, визначаючи чи авторизований об'єкт для зв'язку з іншими. Хоча Auth служить локальною точкою авторизації, він також взаємодіє з іншими об’єктами Auth для управління обміном даними, зареєстрованими з різними Auth, локально розподіляючи накладні витрати авторизації. Потенційні цілі розгортання для Auth включають шлюзи Intel IoT3 і SwarmBox4 з проекту TerraSwarm [88].

Заходи безпеки для бездротових сенсорних мереж (WSN) підтримують автоматичну аутентифікацію для пристроїв з обмеженими ресурсами. Підходи з використанням випадкових або попарних попередньо розподілених ключів шифрування [89,90] забезпечують енергоефективні рішення. Був застосований підхід до управління статичними ключами, який дозволяє встановлювати ключі для знову доданих вузлів [91]. Однак системи управління статичними ключами можуть не підходити для критично важливих для безпеки пристроїв, що працюють у ворожих середовищах, через підвищену вірогідність атаки через криптографічного ключа з тривалим терміном служби.

У статті [92] вивчають динамічні системи управління ключами для мережі WSN, які можуть вирішити проблему з попередньо розподіленими ключами, підтримуючи механізми відкликання ключів. Ці підходи до WSN можна розділити на розподілені і централізовані. У розподілених підходах, включаючи EDDK [93] (Energy-efficient Distributed Deterministic Key management), сусідні вузли взаємодіють для динамічного встановлення ключів. Такі підходи дозволяють уникнути єдиної точки відмови в системах

аутентифікації; проте вони, як правило, більш уразливі для атак і схильні до помилок проектування.

У централізованих підходах центральна довірена третя сторона (наприклад, базова станція) відповідає за генерацію і поширення ключів для вузлів. Багато з цих підходів мають ті ж потоки аутентифікації, що і потік аутентифікації пропонованого підходу на рис. 2 (c). У статті [94] пропонують централізований підхід прямої аутентифікації для ієрархічних і різнорідних сенсорних мереж, що складаються з високопродуктивних і молодших сенсорних вузлів. Водночас у статті [95] забезпечують систему розподілу ключів з використанням безпілотного літального апарату (БПЛА) в якості центру розподілу і координації ключів для великомасштабних мереж WSN. Для підтримки додавання і видалення мобільних сенсорних вузлів у статті [96] пропонують систему управління ключами, яка використовує створення ключів до і після розгортання.

Підсумуємо запропонований підхід відповідно до вимог безпеки IoT рішень (Таблиця 3.3).

Таблиця 3.3 – Відповідність рішення до вимог

Вимоги безпеки IoT	**Запропонований підхід**
1	**2**
Часта аутентифікація та авторизація	Auth контролює кожне безпечне з'єднання і може забезпечити короткі терміни дії ключів сеансу
Автоматична взаємна аутентифікація	Auth забезпечує повністю автоматизовану аутентифікацію; втручання людини не потрібно, крім реєстрації юридичної особи
Переривчасте підключення	Auth дозволяє використовувати кешовані ключі сеансу
Підтримка функцій масштабованості	Ключі сеансу можуть спільно використовувати більше двох об'єктів для зв'язку one-to-many

Продовження табл. 3.3

1	2
Врахування обмежень ресурсів	Для аутентифікації використовуються невеликі і легкі симетричні сеансові ключі; Auth дозволяє використовувати різні криптографічні алгоритми для пристроїв з обмеженими ресурсами, включаючи ті, які не можуть дозволити собі криптографію з відкритим ключем.
Конфіденційність	Для аутентифікації не потрібен унікальний ідентифікатор завдяки використанню тимчасових ключів сеансу
Реєстрація динамічного об'єкта	Суб’єкт може бути легко зареєстрований в Auth, не перериваючи роботу інших

Системи динамічного управління ключами для WSN можуть задовольняти деяку частину вимог безпеки, пов'язаних з IoT. Однак у них все ще є обмежені результати, щоб охопити більшу неоднорідність пристроїв в IoT, від вузлів датчиків з обмеженими ресурсами до критичних компонентів, які вимагають частої аутентифікації і авторизації з метою безпеки. Підтримка динамічного середовища, такого як переривчастий зв'язок і реєстрація динамічних об'єктів, все ще є недостатньою.

Запропонований підхід підтримує часту автоматичну аутентифікацію і авторизацію з використанням локального об'єкта авторизації під назвою Auth. Auth авторизує зареєстровані об'єкти за допомогою розподілу сеансового ключа. Кешуючи ключі сеансу і використовуючи різні криптографічні алгоритми, можна ефективно авторизувати навіть об'єкти з переривчастим зв'язком або обмеженнями ресурсів. Для аутентифікації і авторизації об'єкту необхідно використовувати тільки тимчасові ключі сеансу, надані Auth. Таким чином, не потрібно ризикувати розкрити свою особистість, використовуючи своє унікальне значення, таке як сертифікат, зберігаючи свою конфіденційність. Даний підхід

має значно кращу масштабованість, ніж SSL/TLS, для сценаріїв, поширених в IoT, при забезпеченні порівнянного рівня безпеки з SSL/TLS.

3.4 Висновки до розділу 3

У цьому розділі проведено аналіз та запропоновано власний варіант архітектури системи IoT рішень для процесу сироваріння. Вона складається з трьох рівнів: сприйняття, мережевого та прикладного.

Проведено порівняння можливостей та характеристик різних хмарних сервісів. Прийнято рішення обрати сервіс Microsoft Azure за рахунок доступності та переваг над конкурентами.

На сьогодні питання безпеки мережі є досить суттєвим, задля захисту від дій зловмисників та компаній конкурентів. Проведено аналіз методів і способів захищеності інформації. Запропоновано використовувати покращений алгоритм криптографії, порівняний з протоколом TLS.

РОЗДІЛ 4 ОПИС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У цьому розділі розглядаються програмна частина, а саме: основні функції та бібліотеки для програмування мікроконтролера Arduino. Також, приводиться блок схема роботи програмного забезпечення, графічний інтерфейс системи та прототип системи.

4.1 Програмування мікроконтролера Arduino

Відповідно до проведеного аналізу технологічних стадій виготовлення сиру у розділі 1, складемо схему відповідності етапів виробництва, показнику температури/вологості середовища та необхідних давачів (рис. 4.1).

Етап	Показник температури/ вологості	Необхідні давачі
Приймання молока, визначення його якості	10 ^{O}C	Давач температури, рівня кислотності та електропровідності
Охолодження молока	6-8 ^{O}C	Давач температури, рівня кислотності та електропровідності
Резервування Етап дозрівання (12-16 год)	8-13 ^{O}C	Давач температури, рівня кислотності та електропровідності
Нормалізація молока по % жирності та пастеризація	71-72 ^{O}C	Давач температури, рівня кислотності та електропровідності
Сичужне згортання молока	25-45 ^{O}C	Давач температури, рівня кислотності та електропровідності
Формування форми сиру	20-22 ^{O}C	Давач температури, вологості, кислотності та електропровідності
Дозрівання сирів	12 - 15 ^{O}C 70-90 %	Давач температури, вологості, рівня CO_2
Зберігання сирів	-4 +6 ^{O}C 80-90 %	Давач температури, вологості, рівня CO_2

Рисунок 4.1 – Відповідність етапів виробництва сиру, показнику температури/вологості середовища та необхідних давачів

Згідно із запропонованою архітектурою системи IoT рішень для процесу сироваріння у розділі 3, на рівні сприйняття знаходиться домен вузлів давачів.

Тобто для кожного етапу виробництва сиру потрібно розмістити необхідну групу давачів, які будуть передавати інформацію щодо виміряних даних. У розділі 2 проведено вибір елементної бази для реалізації такої системи. На жаль на даному етапі розробки відсутнє фінансування, тому спробуємо реалізувати програмно-апаратну частину тільки одного вузла. Відповідно до обраної елементної бази у розділі 2, в наявності є: мікроконтролер Arduino Nano V3, давач освітленості GY-302, давач температури та вологості AM2320, давач рівня CO_2 MQ-135, та модуль WI-FI – ESP8266. Даний перелік компонентів найкраще підходить для реалізації вузла клімат-контролю для камери зберігання сирів.

Першим кроком є встановлення програмного забезпечення Arduino IDE та необхідних драйверів для розпізнання комп'ютером плати та емуляції її підключення через COM-порт. Наступний крок – правильно підключити обрані давачі до плати Arduino. Інтерфейси обробки та передачі даних з давачів на мікроконтролер розглянуто у 2му розділі. Пригадаємо, що використовується послідовна шина даних I2C для давачів температури, вологості та освітленості та вбудований аналого-цифровий перетворювач для давачу рівня CO_2. На початковому етапі розробки відмовимось від модуля Wi-Fi й передаватимемо дані на ПК користувача через цифровий порт. На рисунку 4.2 зображено конфігурацію пінів мікроконтролера Arduino Nano V3 [97].

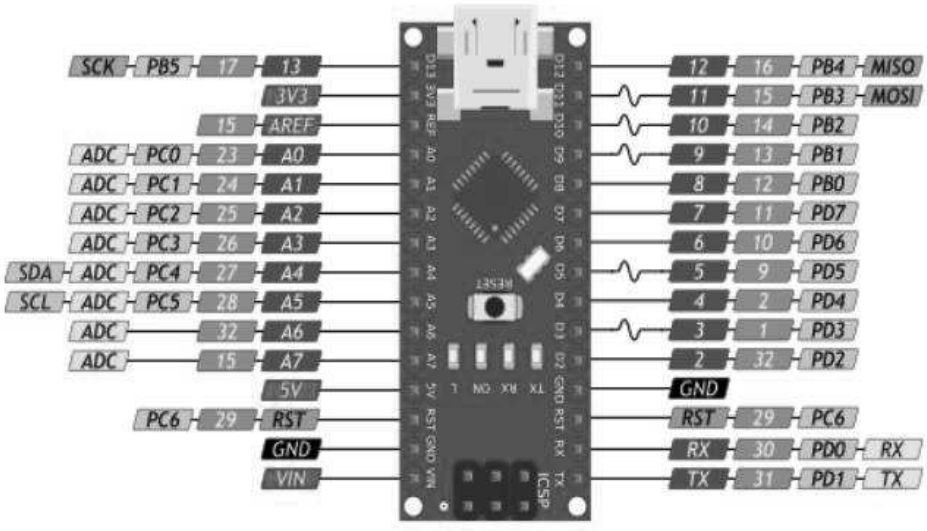


Рисунок 4.2 – Конфігурація пінів Arduino Nano V3 [97]

На рисунку 4.3 зображено схеми підключення давачів: (а) температури та вологості AM2320, (б) рівня газу CO_2 MQ-135, (с) освітленості GY-302.

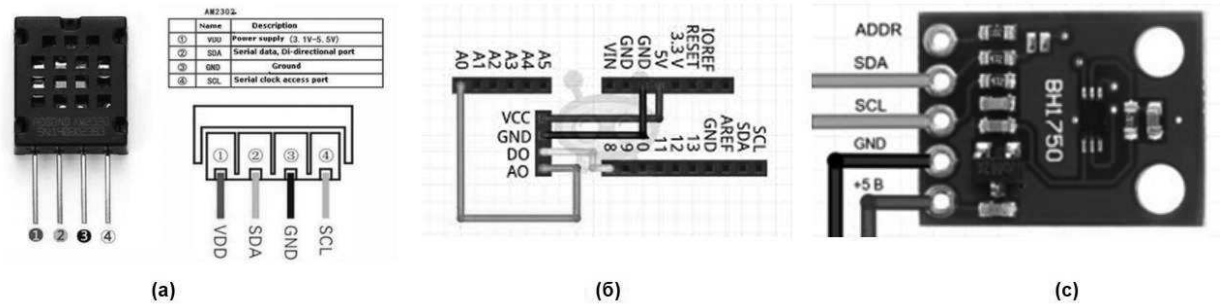


Рисунок 4.3 – Схема підключення давачів: (а) температури та вологості AM2320, (б) рівня газу CO_2 MQ-135, (с) освітленості GY-302

Зважаючи на те, що у нас відсутні прилади, якими в майбутньому буде керувати система, відтворимо індикацію активних процесів (обігрів, охолодження, зволоження, осушування, вентиляція, управління освітленням) за рахунок світлодіодів. Тобто коли процес має активуватись, загорається світлодіод.

На рисунку 4.4 зображено тестовий прототип вузла системи IoT рішень для клімат-контролю камери зберігання сирів.

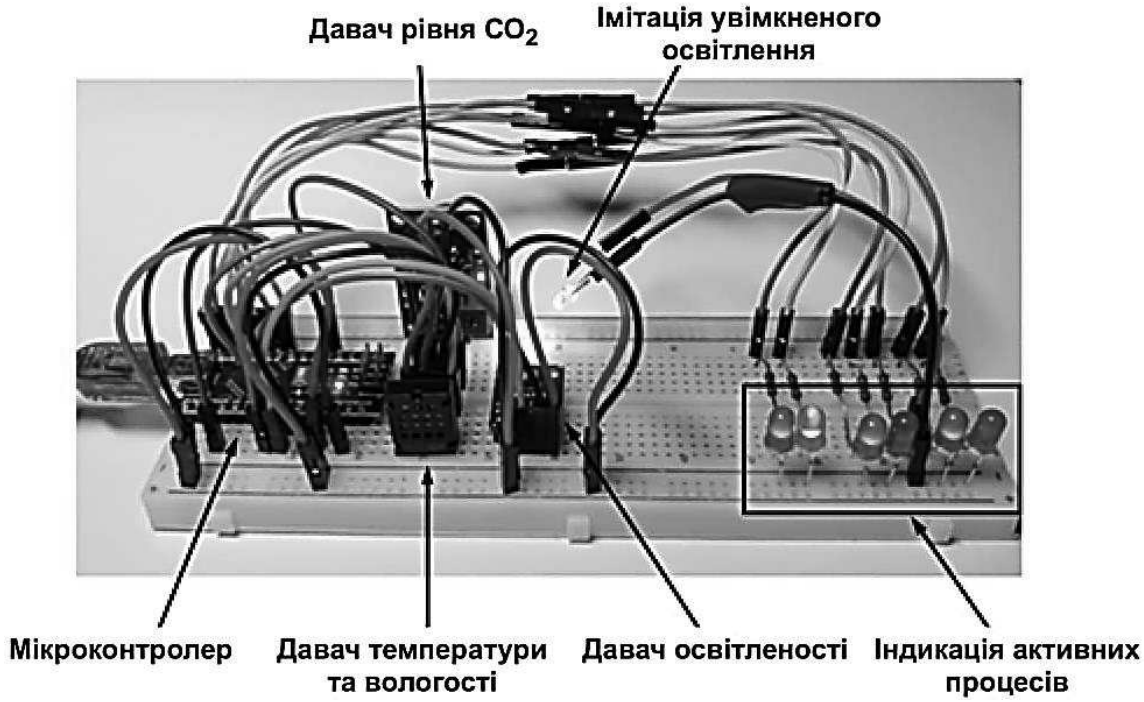


Рисунок 4.4 – Прототип вузла системи IoT рішень

Запрограмуємо мікроконтролер Arduino Nano V3 на базі ATmega328P за допомогою програмного забезпечення Arduino IDE.

Розглянемо основні моменти. Для підключення та отримання даних з давачів використаємо наступні бібліотеки:

1) Wire.h – інтерфейс I2C;

2) AM2320.h – бібліотека для роботи з давачем температури та вологості;

3) BH1750.h – бібліотека для роботи з давачем освітленості.

Підключаємо дані бібліотеки та створюємо об’єкти для подальшого звернення до них в проекті програми для мікроконтролеру Arduino (рис. 4.5).

```
// подключаем библиотеку I2C:
#include <Wire.h>
//подключаем библиотеку датчика AM2320:
#include <AM2320.h>
// объявляем объект th
AM2320 th;
// подключаем библиотеку датчика BH1750:
#include <BH1750.h>
// объявляем объект lightMeter:
BH1750 lightMeter;
```

Рисунок 4.5 – Підключення бібліотек давачів

Далі виконуємо ініціалізацію змінних на відповідність підключення давачів та діодів до пінів плати (рис. 4.6).

```
//gas sensor
const int gas_lvl = A0; //пин уровеня газа
const int gas_ind = 2; //пин наличия газа
boolean noGas; //переменная для хранения значения о присутствии газа
int gasValue = 0; //переменная для хранения количества газа
//инициализация управления
int led_heat = 12;
int led_cooling = 11;
int led_dehum = 10;
int led_hum = 9;
int led_gas_leak = 8;
int led_fan = 7;
int led_light = 6;
//оптимальные значения
int opt_hum = 70;
int opt_temp = 30;
```

Рисунок 4.6 – Ініціалізація змінних

При програмуванні мікроконтролеру Arduino важливою особливістю є наявність двох обов'язкових блоків: setup() – функції налаштування, виконується один раз, loop() – основна частина програми, виконується в нескінченному циклі. На рисунку 4.7 зображена конфігурація блоку setup(), де вказана швидкість зв'язку через послідовний порт – 9600 бод, налаштування для пінів (OUTPUT – сигнал на вихід, INPUT -сигнал на вхід) та ініціалізація давача освітлення.

```
void setup() {
  Serial.begin(9600);
  //лед управление
  pinMode (led_heat, OUTPUT);
  pinMode (led_cooling, OUTPUT);
  pinMode (led_dehum, OUTPUT);
  pinMode (led_hum, OUTPUT);
  pinMode (led_gas_leak, OUTPUT);
  pinMode (led_fan, OUTPUT);
  pinMode (led_light, OUTPUT);
  //индикатор газа
  pinMode(gas_ind, INPUT); //установка режима пина
  //датчик освещения
   lightMeter.begin(); //инициализация датчика BH1750
}
```

Рисунок 4.7 – конфігурація блоку setup()

Інший текст програми написаний в блоці loop(). Для зчитування значення рівня CO_2 в повітрі використаємо наступні функції:

1) digitalRead() – присутність газу CO_2 в повітрі;
2) analogRead() – кількість газу CO_2 в повітрі.

Для передачі даних на ПК користувача через цифровий порт використаємо наступні функції:

1) Serial.print() – для запису значень давачів у послідовний порт;
2) Serial.read() – для зчитування даних з послідовного порта.

Для індикації активних процесів використаємо наступні функції:

1) digitalWrite(10, HIGH) – вмикаємо світлодіод;
2) digitalWrite(10, LOW) – вимикаємо світлодіод.

Головна задача мікроконтролеру Arduino Nano V3 – зчитати необхідні дані з давачів, передати на ПК користувача, й відповідно до результату, подати сигнал на активацію певного процесу.

Обробка отриманих даних з давача газу зображена на рисунку 4.8. Давач передає на мікроконтролер значення вмісту газу в навколишньому середовищі. Якщо воно перевищує норму – з'являється сигнал тривоги й запускається вентиляція (у нашому випадку загораються відповідні світлодіоди).

```
void loop() {
  //модуль газа
  noGas = digitalRead(gas_ind); //считываем значение о присутствии газа
  gasValue = analogRead(gas_lvl); // и о его количестве
  //вывод
  if (noGas)
  {
      Serial.print("Утечки газа нету. ");
      Serial.print("Уровень газа: ");
      Serial.println(gasValue);
      digitalWrite (led_gas_leak, LOW); // выключить светодиод
      delay(1000);
      digitalWrite (led_fan, LOW); // включить светодиод
      delay(1000);
  }
  else
  {
      Serial.print("Внимание! Утечка газа!");
      digitalWrite (led_gas_leak, HIGH); // включить светодиод
      delay(1000);
      Serial.print("Уровень газа: ");
      Serial.println(gasValue);
      Serial.print("Запущена вентиляция");
      digitalWrite (led_fan, HIGH); // включить светодиод
      delay(1000);
```

Рисунок 4.8 – Обробка отриманих даних з давача рівня газу CO_2

Обробка отриманих даних з давача температури та вологості розпочинається з перевірки на стабільне підключення до мікроконтролеру Arduino (рис. 4.9).

```
//модуль температуры/влажности
switch(th.Read()) {
  case 2:
    Serial.println("CRC failed");
    break;
  case 1:
    Serial.println("Sensor offline");
    break;
  case 0:
    Serial.print("Влажность воздуха: ");
    Serial.print(th.h);
    Serial.print("%, Температура: ");
    Serial.print(th.t);
    Serial.println("*C");
```

Рисунок 4.9 – Перевірка підключення давача температури та вологості

Якщо відсутні повідомлення про помилку з'єднання, отримуємо інформацію з середовища й порівнюємо її з оптимальними даними, заданими в момент ініціалізації змінних. Приклад обробки даних температури наведено на рисунку 4.10. Данні про вологість середовища обробляються за подібним алгоритмом.

```
//проверка температуры
   if (th.t >= opt_temp)
  {
     Serial.println("Охлаждение помещения запущено");
     digitalWrite (led_heat, LOW); // включить светодиод
     digitalWrite (led_cooling, HIGH); // включить светодиод
     delay(200);
  }
  else
  {
    Serial.println("Нагрев помещения запущено");
    digitalWrite (led_cooling, LOW); // выключить светодиод
    digitalWrite (led_heat, HIGH); // включить светодиод
    delay(200);
  }
```

Рисунок 4.10 – Обробка даних температури

Прийнято рішення зробити імітацію освітлення камери зберігання за рахунок світлодіоду й розташувати його прямо над давачем освітленості. Після калібрування системи отримано значення освітленості в темну пору доби. І якщо в певний момент часу воно зросте, мікроконтролер сповістить про ввімкнене освітлення (рис. 4.11).

```
//проверка освещенности
  digitalWrite (led_light, HIGH); // включить светодиод
  delay(1000);
  //считываем показания с BH1750:
  uint16_t lux = lightMeter.readLightLevel();
  //выводим показания в послед. порт:
  Serial.print("Уровень освещенности:");
  Serial.println(String(lux) + " lx");
  if (lux>80)
  {
    Serial.println("Освещение хранилища включено!");
  }
  else
  {
    Serial.println("Освещение хранилища выключено.");
  }
```

Рисунок 4.11 – Обробка даних освітленості

Результат роботи зібраного прототипу вузла клімат-контролю для камери зберігання сирів на базі мікроконтролеру Arduino, обраних давачів та написаної програми зображено на рисунку 4.12.

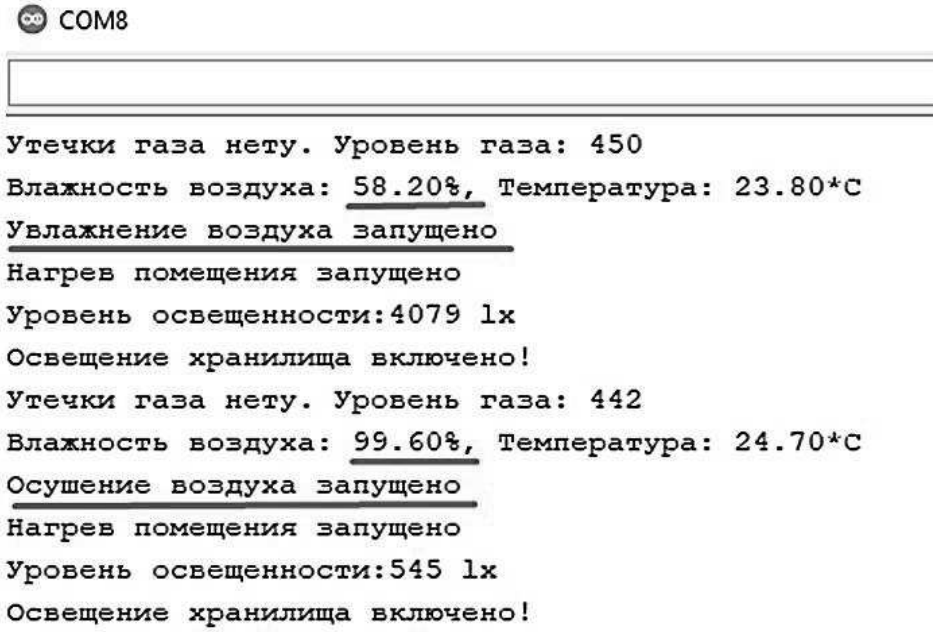


Рисунок 4.12 – Результат роботи системи клімат контролю

Наступним етапом розробки є додавання до нашої реалізації вузла клімат-контролю для камери зберігання сирів модулю WI-FI – ESP8266, реалізованого на платі ESP-07. На рисунку 4.13 зображено конфігурацію пінів даного модуля [98].

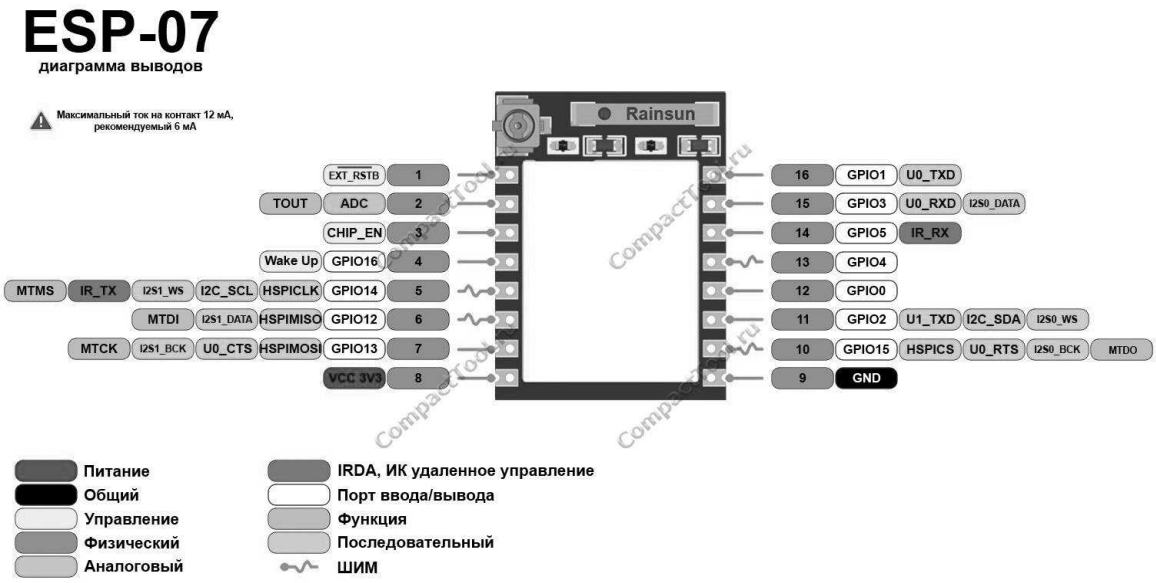


Рисунок 4.13 – Конфігурація пінів ESP-07 [98]

Для підключення модулю WI-FI до плати Arduino необхідно зібрати спеціальну схему з периферією, що забезпечить зручне прошивання модулю та стабільність роботи. Схема підключення зображена на рисунку 4.14.

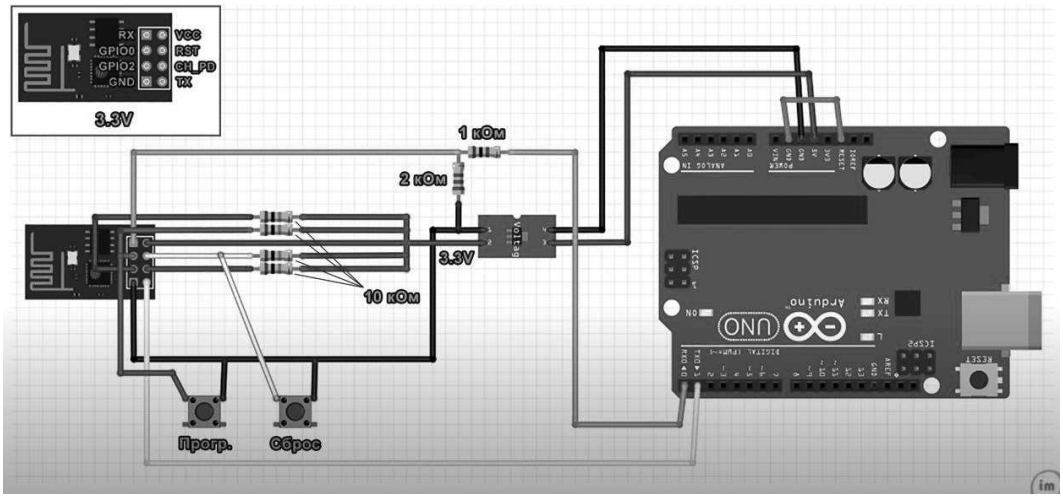


Рисунок 4.14 – Схемі підключення модулю ESP-07

На рисунку 4.15 зображено прототип підключення модулю Wi-Fi – ESP8266 до плати Arduino Nano v3.

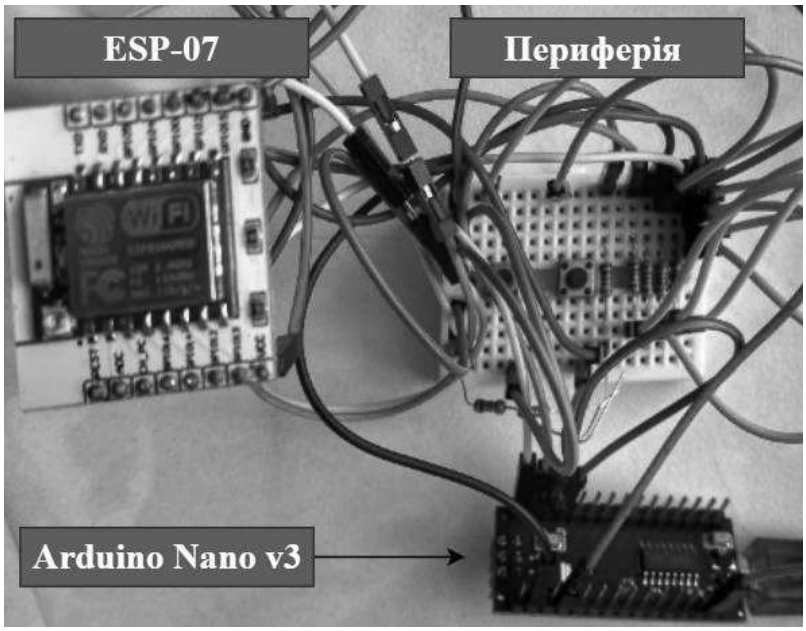


Рисунок 4.15 – Прототип підключення модулю ESP-07

Запрограмуємо підключений Wi-Fi модуль. Для цього скористаємось бібліотекою ESP8266WebServer. Опишемо основні етапи. Ініціалізуємо зміні, яким присвоїмо назву та пароль від локальної Wi-Fi мережі (рис.4.16).

```
#include <ESP8266WebServer.h>
/* Введите SSID и пароль от вашей сети*/
const char* ssid = "Sirius";   // SSID
const char* password = "luna_51a";  // пароль
ESP8266WebServer server(80);
```

Рисунок 4.16 – Ініціалізація змінних

Блок setup() включає задання швидкості зв'язку в 115200 бод, під'єднання до локальної Wi-Fi мережі за заданими раніше даними (ім'я мережі та пароль), перевірку на підключення та отримання IP адреси. Після цього сервер вважається запущеним (рис. 4.17).

```
void setup()
{
  Serial.begin(115200);
  delay(100);
  Serial.println("Connecting to ");
  Serial.println(ssid);
  // подключиться к вашей локальной wi-fi сети
  WiFi.begin(ssid, password);
  // проверить, выполнено ли подключение wi-fi сети
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");
  Serial.println(WiFi.localIP());
  server.on("/", handle_OnConnect);
  server.on("/getdata", handle_OnConnectData);
  server.onNotFound(handle_NotFound);
  server.begin();
  Serial.println("HTTP server started");
}
```

Рисунок 4.17 – Конфігурація блоку setup() для Wi-Fi мережі

У циклічному блоці loop() реалізовано обробку даних давачами, що характеризують навколишнє середовище. Функція handle_OnConnectData() передає повідомлення на сервер з інформацією від давачів (рис. 4.18).

```
void handle_OnConnectData()
{
  String ptr = "";
  ptr +="<h1>ESP8266 Climate Monitor</h1>\n";
  ptr +="<p>Air humidity: ";
  ptr +=HumCString;
  ptr +=" %</p>";
  ptr +="<p>Temperature: ";
  ptr +=TempCString;
  ptr +=" &deg;C</p>";
  ptr +="<p>Gas level: ";
  ptr +=CO2CString;
  ptr +=" ppm</p>";
  ptr +="<p>Illumination level: ";
  ptr +=LuxCString;
  ptr +=" lx</p>";
  server.send(200, "text/html", ptr);
}
```

Рисунок 4.18 – Реалізація функції handle_OnConnectData()

Для тестування роботи програми створимо web-сторінку та відправимо на неї дані з web-серверу. Оброблені значення від давачів, що характеризують навколишнє середовище відправились через мережу Wi-Fi. На рисунку 4.19 зображено скріншот сторінки браузера.

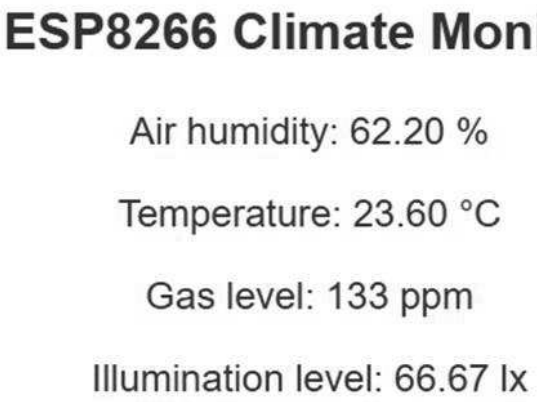


Рисунок 4.19 – Скріншот сторінки браузера

Web-сервер працює, отже можна переходити до наступного етапу – розробки спеціалізованого програмного забезпечення та графічного інтерфейсу користувача для вузла системи клімат-контролю для камери зберігання сирів.

Повний програмний код для мікроконтролеру Arduino приведений у додатку А.

4.2 Розробка програмного забезпечення та графічного інтерфейсу

Згідно з описом методу автоматичного управління технологічним процесом сироваріння (розділ 1), розробимо блок схему роботи програмного забезпечення для реалізації обраного вузла системи клімат-контролю для камери зберігання сирів (рис. 4.20).

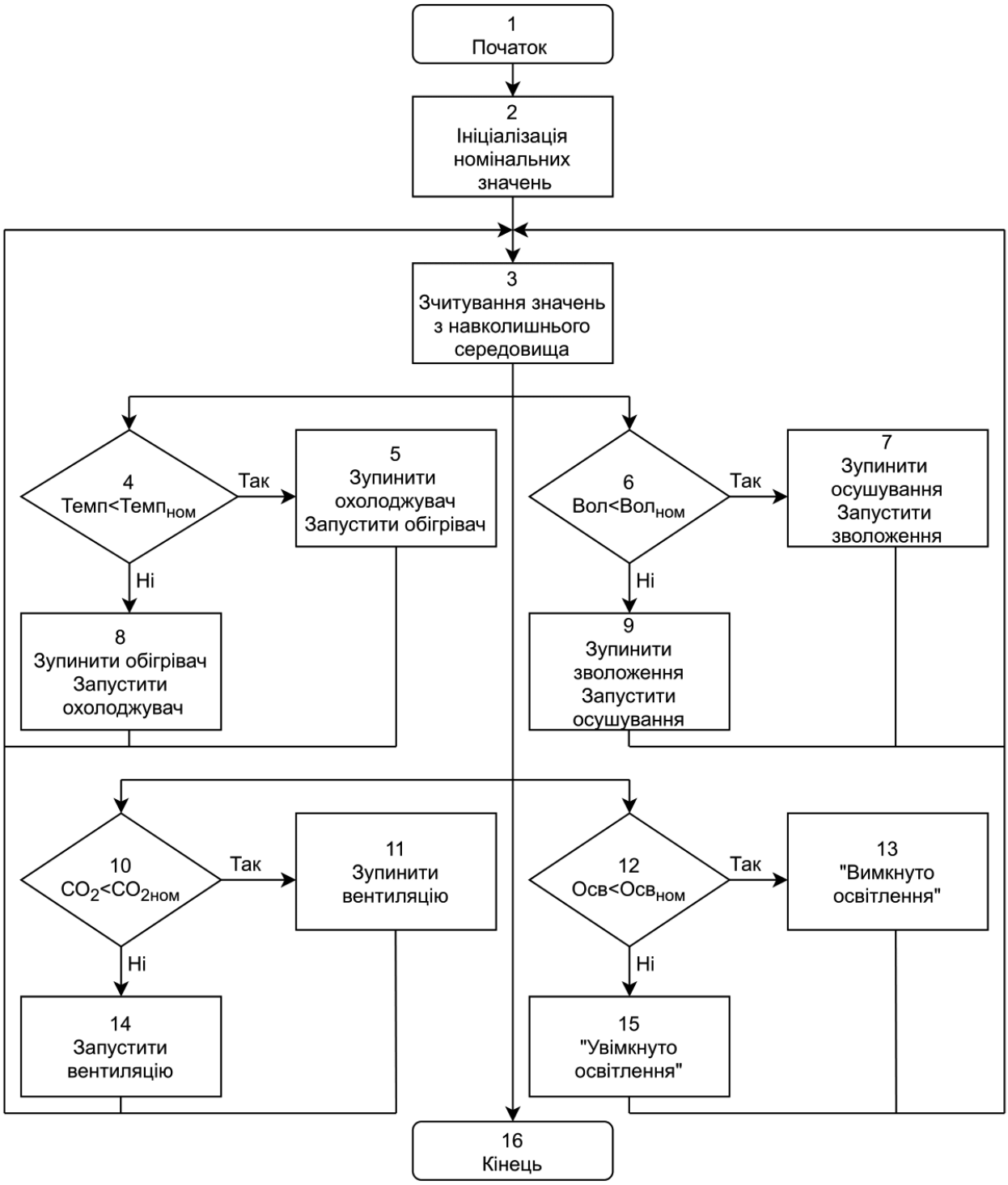


Рисунок 4.20 – Блок схема роботи програмного забезпечення

Відповідно до кроків методу автоматичного управління технологічним процесом сироваріння, спершу відбувається ініціалізація номінальних значень, потім давачі зчитують значення з навколишнього середовища, в якому вони знаходяться. Далі відбувається аналіз кожного режиму, відповідно до характеристик, які надає давач і запуск та зупинка необхідного процесу.

Головна задача програмного забезпечення – прийняти необхідні дані від мікроконтролера, проаналізувати їх за алгоритмом роботи (порівняння номінальних та теперішніх значень) й відповідно до результату, подати сигнал на активацію певного процесу. Для відображення усіх активних процесів використовується LED індикація.

Обираємо для розробки програмного забезпечення та створення графічного інтерфейсу IDE Qt Creator [99]. Дане ПЗ повністю задовольняє наші потреби, тому що має ряд переваг над іншими:

- підтримка декількох платформ розробки ПЗ та можливість адаптації під іншу платформу шляхом перекомпіляції проекту;
- за рахунок цього максимальна продуктивність розробки й зменшення часу витрат у майбутньому;
- використання особливої структури файлів, що дозволяє кодувати на повній швидкості без потреби використання віртуальної машини;
- зручний інтуїтивно зрозумілий інтерфейс;
- гнучкий інструмент розробки графічного інтерфейсу користувача.

Згідно з алгоритмом роботи програми, необхідно розподілити панель керування на умовні частини та реалізувати:

- область для введення фахівцем номінальних значень температури, вологості, рівня газу CO_2;
- область для відображення показників з давачів у реальному часі;
- область керування системою (завантаження значень, початок роботи, закінчення роботи);
- область індикації активних процесів (обігрів, охолодження, зволоження, осушування, вентиляція, управління освітленням).

На рисунку 4.21 зображено розроблений графічний інтерфейс користувача на ОС Windows вузла системи IoT рішень для клімат контролю камери дозрівання сирів.

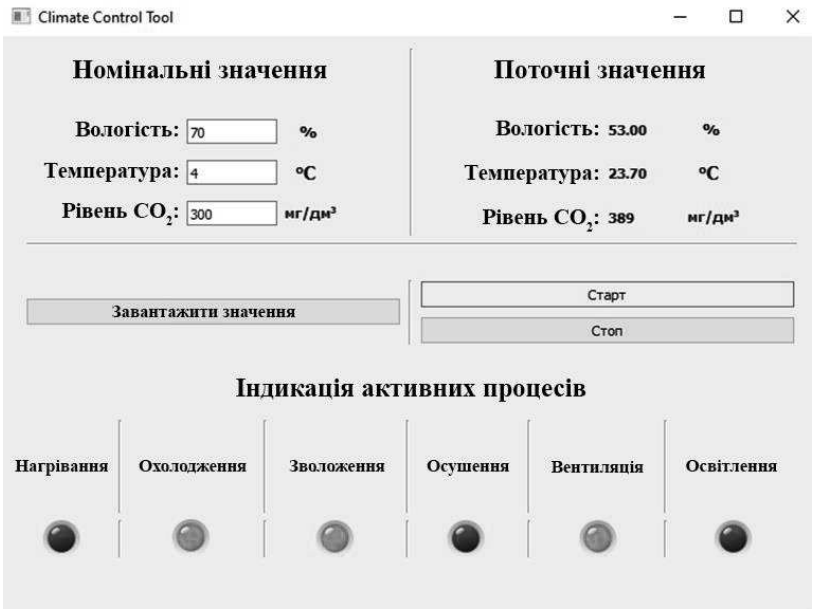


Рисунок 4.21 – Графічний інтерфейс системи для ОС Windows

За рахунок кросплатформенної особливості обраного середовища розробки Qt Creator, також пропоную розглянути концепт додатку на мобільний телефон на базі ОС Android (рис. 4.22).

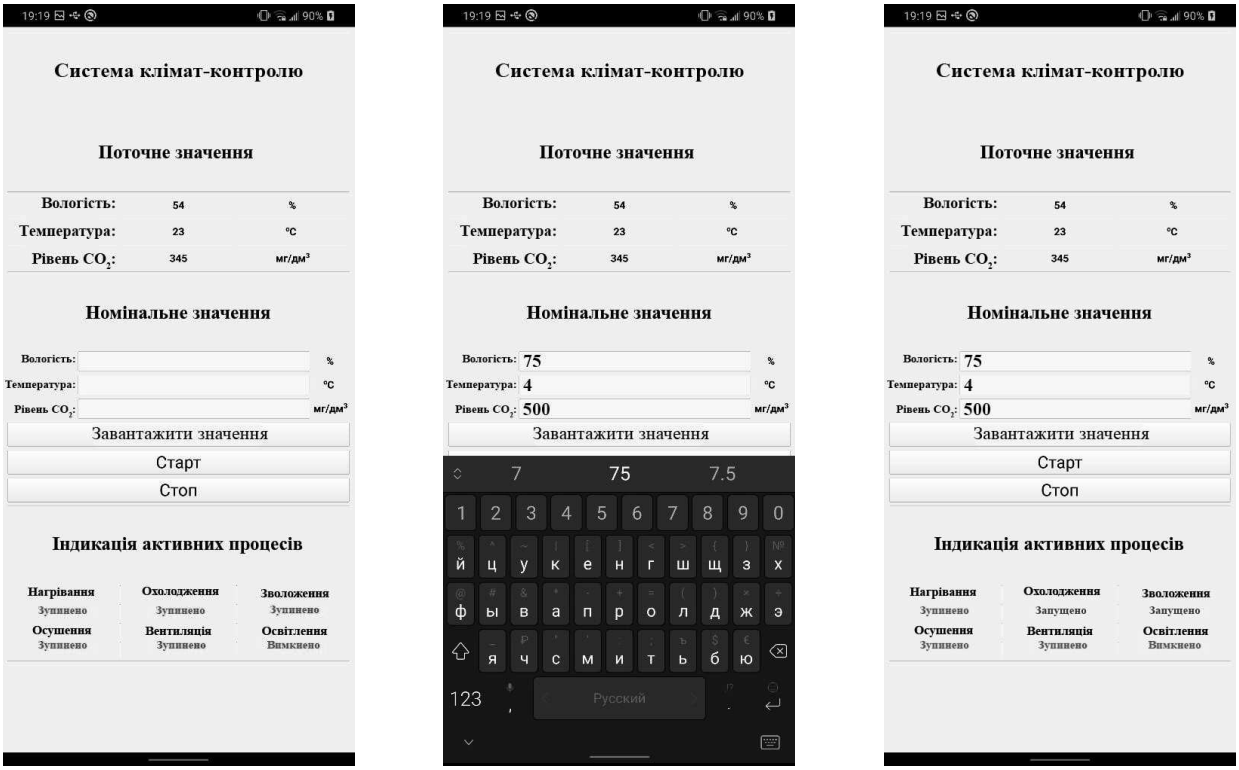


Рисунок 4.22 – Графічний інтерфейс системи для ОС Android

Повний текст програми розробленого спеціалізованого програмного забезпечення для ОС Windows приведений у додатку Б.

Інструкція для користувача наведена в додатку В.

4.3 Аналіз результатів досліджень

4.3.1 Розрахунок вартості системи

Розрахуємо вартість системи IoT рішень для процесу сироваріння. На апаратну реалізацію прототипу вузла витрачено 625 гривень. Виготовлення друкованої плати буде коштувати близько 600 гривень. Програмне забезпечення для мікроконтролера та графічний інтерфейс користувача – 10000 гривень. Технічна підтримка на рік, та можливість оновлень системи – 1500 в місяць. Відповідно до рисунку 4.1, звичайний процес сироваріння складається з 8 етапів – тобто це 8 комплектів давачів. У перших 6 етапах додатково потрібні давачі рівня електропровідності та кислотності pH, ціною в 3000 гривень за комплект. Повна вартість системи IoT рішень для процесу сироваріння приведена в таблиці 4.1.

Таблиця 4.1 – Розрахунок вартості системи

Найменування	**Ціна за 1 шт., грн**	**Кількість**	**Ціна за комплект, грн**
Друкована плата	600	8	4800
Давач рівня електропровідності	2000	6	12000
Давач кислотності pH	1000	6	6000
Елементна база	500	8	4000
$\Sigma_{\text{апаратне забезпечення}}$			23300
Програмне забезпечення	20000	1	20000
Технічна підтримка	1500	12	18000
$\Sigma_{\text{програмне забезпечення}}$			38000
$\Sigma_{\text{систему}}$			61300

Таким чином, вартість апаратного забезпечення складає близько 850 доларів, програмного забезпечення – 1350 доларів, а розробленої системи

складає 61300 гривень, або близько 2200 доларів, включаючи технічну підтримку на 1 рік.

4.3.2 Порівняння розробленої системи з аналогами

Складемо таблицю порівнянь розробленої системи IoT рішень для процесу сироваріння з аналогами, розглянутими в розділі 1 (таблиця 4.2).

Таблиця 4.2 Порівняння між системами

		Назва системи		
		«Модульна блок-установка»	«Терраформ»	Власна розробка
1	**2**	**3**	**4**	**5**
Критерії порівняння	**Призначення**	Клімат-контроль для камери дозрівання сирів.	Клімат-контроль для теплиць	Система IoT рішень для процесу сироваріння
	Можливості	Забезпечення необхідних режимів температури та вологості повітря.	Управління опаленням, поливом, освітленням та вентиляцією. Віддалений контроль.	Автоматизовані процеси: Обігрів, охолодження, зволоження, осушування, вентиляція, управління освітленням
	Характеристики	1) Підтримка температурного режиму від 10 ℃ до 20 ℃; 2) Підтримка вологості повітря до 99%; 3) Споживана потужність близько 2,2 кВт; 4) Об’єм камери від 30 м3 до 60 м3.	1) 8 реле для керування пристроями; 2) потужність навантаження реле 1 кВт; 3) 4 давачі температури ґрунту, рідини; 4) 4 давачі вологості ґрунту; 5) 4 давачі освітленості ; 6) 2 давачі концентрації CO2.	На один реалізований вузол: 1) Діапазон виміру вологості повітря до 100%; 2) Діапазон виміру температур -40 до 80℃ 3) Відслідковування рівня газу та керування вентиляцією 4) Керування освітленням приміщення У майбутньому: 5) Визначення якості молока (давачі рівня електропровідності та кислотності pH)

Продовження табл. 4.2

1	2	3	4	5
	Недоліки	Один варіанту реалізації без використання апаратно-програмних технологій.	Низька потужність навантаження на реле – всього 1 кВт.	Відсутність фінансування для подальшої розробки інших вузлів системи IoT рішень для процесу сироваріння
	Ціна	1000 доларів	800 доларів	2200 доларів
	Вирішує	3 задачі	6 задач	27 задач

Відповідно до порівняння в таблиці 4.2 можна зробити висновок, що спроектована система врахувала всі недоліки аналогів, має розширений діапазон можливостей та автоматизовану апаратно-програмну реалізацію.

4.3.3 SWOT-аналіз результатів

Strengths. Сильною стороною системи IoT рішень для процесу сироваріння є її структура. Завдяки розгалуженій структурі, в програмі паралельно отримуються та обробляються дані з давачів світла, температури та вологості.

Weaknesses. Недоліком створеного програмного продукту є початковий етап впровадження системи, що потребуватиме додаткових коштів на закупівлю давачів та монтування мережі зв’язку як апаратної частини забезпечення роботи системи IoT рішень для процесу сироваріння, але це окупиться в процесі експлуатації.

Opportunities. В подальшому дана система дозволяє додавати модулі для аналізу виготовленої продукції або систему зберігання.

Threats. Загрозами, що матимуть негативні наслідки для системи IoT рішень для процесу сироваріння можна вважати: недостатнє фінансування;

витрати на навчання персоналу; необхідність введення до персоналу заводу фахівця, що відповідатиме за супроводження програмного продукту.

Розроблений програмний продукт враховує переваги аналогів на ринку, але має дорожчу вартість та переваги в простоті використання, що необхідно для виробництва сирів.

4.4 Висновки до розділу 4

Відповідно до результату проробленої роботи у розділі 4, виконано наступні пункти:

1) написано програмний код для мікроконтролеру Arduino;
2) зібрано програмно-апаратний прототип вузла системи клімат-контролю для камери зберігання сирів;
3) розроблено блок-схему роботи програмного забезпечення;
4) спроектовано графічний інтерфейс користувача;
5) створено програмне забезпечення для ОС Windows.
6) проведено аналіз результатів дослідження, що включає розрахунок вартості системи, порівняння розробленої системи з аналогами, SWOT-аналіз результатів.

ВИСНОВКИ

У даній кваліфікаційній роботі магістра запропоновано метод автоматичного управління технологічним процесом сироваріння. На основі даного метода спроектовано та програмно реалізовано високоефективну систему IoT рішень для процесу сироваріння з підтримкою автоматизованого управління процесами обігріву, охолодження, зволоження, осушування, вентиляції, освітлення та мінімальним використанням енергоресурсів мікроконтролера.

Виконано:

1. Аналіз базових принципів роботи IoT систем.
2. Аналіз технологічних стадій виготовлення сиру.
3. Аналіз останніх наукових досліджень та практичних розробок IoT систем.
4. Аналіз існуючих систем IoT рішень для процесу сироваріння.
5. Дослідження комунікаційних систем та технологій.
6. Проектування логічних зв’язків вузлів системи.
7. Розроблено схему структурну системи IoT рішень для процесу сироваріння.
8. Розроблено концептуальну модель архітектурного рішення відповідно до проекту системи IoT.
9. Розроблено метод автоматичного управління технологічним процесом сироваріння.
10. Створено програмне забезпечення системи IoT рішень для процесу сироваріння з підтримкою автоматизованого управління процесами обігріву, охолодження, зволоження, осушування, вентиляції, освітлення та мінімальним використанням енергоресурсів мікроконтролера.
11. Проведення верифікації даних засобами прототипу на основі Arduino.

Під час створення даної системи отримано наступні наукові та практичні результати:

1. Проаналізовано базові принципи роботи технології Інтернету речей. Дана технологія з кожним роком все більше набирає обертів. Популярними

галузями впровадження IoT рішень є медицина, освіта, торгівля, економіка, агрокультура та промисловість. В основі архітектури IoT рішень є 4 вузли: давачі та пристрої, мережа, методи обробки даних та інтерфейс користувача. Основні задачі, які вирішує впровадження IoT: підвищення продуктивності праці, збільшення доступності інформації, автоматизації багатьох процесів та полегшення життя людей. З кожним днем перевага надається широкому впровадженню IoT технологій, незалежно від галузі застосування.

2. Проведено аналіз технологічних стадій виготовлення сиру й виявлено, що процес доволі складний, а смакові властивості готового продукту залежать від багатьох факторів. Можна виділити вісім основних технологічних стадій виготовлення сиру, які потребують чіткого відстежування якості вхідних складових продуктів на стану навколишнього середовища. Для автоматизації процесу сироваріння запропоновано встановити необхідну групу давачів в зонах проведення кожної стадії виготовлення сиру.

3. Аналіз останніх наукових досліджень підкреслює актуальність розвитку Промислового Інтернету речей. Наведено приклади впровадження даної технології в різних сферах виробництва: побутової техніки, транспорту, обладнання для розумних будинків та розумних міст. Особливу увагу приділяють екологічній ситуації та інноваційним IoT рішенням для перероблювання відходів і моніторингу якості повітря. При цьому відсутній варіант реалізації системи IoT рішень на виробництві молочних продуктів. Що підкреслює новизну та актуальність обраної теми дослідження.

4. Проаналізовано готові системи на ринку, серед яких: модульна блок-установка для камери зберігання сиру, система клімат-контролю теплиць «Терраформ» та система кліматичного комплексу житлових приміщень Zenet. Відповідно до отриманих результатів порівнянь характеристик цих систем, визначено вхідні дані для проектування моделі систему IoT рішень для процесу сироваріння.

5. Систематизуючи результати проведених аналізів, відсутність різноманіття методів і засобів реалізації моделі системи IoT рішень для процесу

сироваріння свідчить про те, що оптимального програмно-апаратного рішення не знайдено, тому вважаємо за доцільне розгляд нової технологічної пропозиції за темою кваліфікаційної роботи магістра. Прийнято рішення розробити систему, яка у свою чергу є простою в реалізації, підтримує автоматичну нормалізацію необхідних для виробництва сиру умов, модульність, масштабованість, низьку споживану потужність та є доступною по та ціні.

6. Реалізовано проектну частину системи: побудовані UML-діаграми використання, послідовності та стану, що показують взаємозв'язки, порядок дій та стани системи; розроблено алгоритм роботи системи та структурну схему. Проведено вибір елементної бази, де усі підібрані компоненти знаходяться у низькому ціновому сегменті та відповідають поставленій задачі. Мікроконтролер Arduino Nano V3 зчитує необхідні дані про стан навколишнього середовища з вбудованих давачів (температури, вологості, рівня CO_2, значення електропровідності та кислотності pH) та передає їх на ПК користувача. Відповідно до результату порівняння зчитаних даних та номінальних (встановлюються фахівцем на етапі налаштування програмного забезпечення), МК подає сигнал для активації відповідного реле для виконання операції відносно нормалізації навколишнього середовища. Також побудовано схему мережі зв'язку системи.

7. Розроблено концептуальну модель архітектурного рішення відповідно до проектної частини. За основу обрано тришарову архітектуру, що цілком розкриває сутність нашої системи. На рівні сприйняття знаходиться домен вузла давачів, які будуть знімати дані з навколишнього середовища. Мережевий рівень відповідає за передачу даних та сигналів управління системою між мікроконтролером та прикладним рівнем (сервер та панель управління користувача).

8. Проведено аналіз комунікаційних технологій та систем, що показав проблему відсутності стандартизації IoT рішень. Її вирішують шляхом використання єдиної хмарної платформи, що забезпечує покращення інтеграції та обміну інформацією між розподіленими системами. Проаналізовано плюси й

мінуси відомих хмарних сервісів, таких як: Microsoft Azure IoT Hub, SmartThings та Google OpenThread. Відповідно до аналізу та результату порівняння платформ, запропоновано обрати Microsoft Azure для майбутньої розробки системи IoT рішень для процесу сироваріння. Вона має ряд переваг над конкурентами та зазвичай використовується для реалізації серйозних проектів в різних галузях за рахунок можливості гнучкого налаштування.

9. Запропоновано безпечну мережеву архітектуру з використанням локальних автоматизованих об'єктів авторизації для задоволення вимог, пов'язаних з IoT, при забезпеченні гарантій безпеки високого рівня. Дана мережева архітектура забезпечує механізми управління ключами, які дуже добре масштабуються і добре працюють з пристроями з обмеженими ресурсами або нестабільним підключенням. Запропонована архітектура може використовувати алгоритми криптографії, які використовуються в протоколі TLS, для забезпечення відповідного рівня гарантій безпеки.

10. Підібрано набір давачів для кожного вузла системи. За відсутності фінансування, прийнято рішення виконати прототипування тільки останнього вузла, що відповідає за клімат-контроль для камери зберігання сирів. Написано програмний код для МК Arduino в спеціалізованому IDE. Головною задачею для МК є обробка даних з давачів, передача їх на ПК користувача, очікування сигналу на активацію певного процесу. Розроблено метод автоматичного управління технологічним процесом сироваріння. Також розроблено програмне забезпечення та графічний інтерфейс користувача в середовищі Qt Creator для ОС Windows. Головною задачею програмного забезпечення є прийняти необхідні дані від мікроконтролера, проаналізувати їх за алгоритмом порівняння номінальних та теперішніх значень й відповідно до результату, подати сигнал на активацію певного процесу. Зібрано прототип системи на основі відібраної елементної бази в розділі 2 та проведено успішне тестування. Проведено порівняння розробленої системи IoT рішень для процесу сироваріння з аналогами, розглянутими в розділі 1 та SWOT-аналіз результатів. Власна

розробка врахувала всі недоліки аналогів, має покращений функціонал, а також відповідає закладеним на початку проектування вимогам.

За матеріалами роботи опубліковано 4 друковані праці.

Отримані результати свідчать про виконання кваліфікаційної роботи магістра у повному обсязі. Досягнуто всіх поставлених у меті завдань. Планується впровадження системи на виробництво.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kravchenko O.V., Gladka M.V., Kucherenko R.Y. Development of IoT solutions for climate control of dairy production process // Topical issues of the development of modern science. Abstracts of the 10th International scientific and practical conference. Publishing House “Accent”. Sofia, Bulgaria. 2020. Pp. 48-50. URL: https://sci-conf.com.ua. (дата звертання 07.03.2021)

2. Kravchenko O. V., Kucherenko R. Y. IOT Solutions System For Climate Control Process of Making Cheese //Information Technology and Interactions (Satellite): Conference Proceedings, December 04, 2020, Kyiv, Ukraine / Taras Shevchenko National University of Kyiv and [etc]; Vitaliy Snytyuk (Editor). Kyiv: Stylos, 2020. Pp. 352 ISBN 978-966-2399-61-5

3. Кучеренко Р.Ю. Модель системи IoT рішень для процесу сироваріння// XVII міжнародна науково-технічна конференція «Проблеми інформатизації», 2021 р.

4. Kravchenko O. V., Kucherenko R. Y., Danchenko E.B., Besedina S.V. Розробка системи IoT рішень для клімат контролю процесу виготовлення молочної продукції // Sciences of Europe, Vol 2, No 51. Praha, Czech Republic. 2020. Pp. 69-75.

5. Дослідження ринку молока [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/UuosWK0 (дата звертання 07.03.2021)

6. Товажнянський Л. Л., Буккало С. І., Капустенко П. О. та ін. Загальна технологія харчової промисловості у прикладах і задачах. Підручник ЦУЛ 2011 р. 832 ст.

7. Gartner Says 4.9 Billion Connected "Things" Will Be in Use in 2015 [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/euodlAj (дата звертання 07.03.2021)

8. The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025 [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/Ouod7sG (дата звертання 07.03.2021)

9. What is Internet of Things (IoT) and How it Works? [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/FuofyPy (дата звертання 07.03.2021)

10. 10 найпопулярніших сфер використання інтернету речей [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/auofl4S (дата звертання 07.03.2021)

11. The IoT Evolution – Top 9 IoT Use Cases [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/5uoghEK (дата звертання 07.03.2021)

12. Как работает IoT: анализ технических решений [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/CuognIn (дата звертання 07.03.2021)

13. Товажнянський Л. Л., Бухкало С. І., Капустенко П. О. та ін. Загальна технологія харчової промисловості у прикладах і задачах. Підручник ЦУЛ 2011 р. 832 ст.

14. Клаус Шваб. Четвертая промышленная революция. – Из-во: Эксмо, 2016, 208 с. ISBN: 978-5-699-90556-0

15. D. Mourtzis, E.Vlachou, N.Milas. Industrial Big Data as a Result of IoT Adoption in Manufacturing // Procedia CIRP. – Volume 55, 2016, pp. 290-295.

16. ICTC, Big Data & The Intelligence Economy [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/Guog0EV (дата звертання 07.03.2021)

17. Промисловий інтернет речей [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/Rt2xC7i (дата звертання 07.03.2021)

18. Станок как сервис: от системы мониторинга к цифровой фабрике [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/pt2x6qC (дата звертання 07.03.2021)

19. How IoT and automation will transform how industries function [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/pt2crbj (дата звертання 07.03.2021)

20. How the Growing Impact of IoT Is Automating the World [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/7uohvIo (дата звертання 07.03.2021)

21. Perera C., Chi H. L. M. A survey on internet of things from industrial market perspective // IEEE Access. – Volume 2, 2014, pp. 1660–1679.

22. O'Halloran D., Kvochko E. Industrial internet of things: Unleashing the potential of connected products and services// World Economic Forums IT Governors (2015) [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/vuoh3Q8 (дата звертання 07.03.2021)

23. Debasish Mondal, The Internet of Thing (IOT) and Industrial Automation: a future perspective// World Journal of Modelling and Simulation (2019) [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/muojhUk (дата звертання 07.03.2021)

24. Anamul Haque S. M., Kamruzzaman S. M., Md. Ashraful Islam . A System for Smart Home Control of Appliances based on Timer and Speech Interaction// Proc. 4th International Conference on Electrical Engineering, The Institution of Engineers, Dhaka, Bangladesh (2006), pp. 128-131

25. Deepali Javale, Mohd. Mohsin, Shreerang Nandanwar. Home Automation and Security System Using Android ADK.// International Journal of Electronics Communication and Computer Technology (IJECCT) . – Volume 3, Issue 2, 2013

26. Nathan David, Abafor Chima, Aronu Ugochukwu, Edoga Obinna. Design of a Home Automation System Using Arduino// International Journal of Scientific & Engineering Research. (2015) – Volume 6, Issue 6, pp. 795. ISSN 2229-551

27. Shopan Dey, Ayon Roy and Sandip Das. Home Automation Using Internet of Thing// IRJET (2016), 2(3) pp. 1965-1970

28. Dr. Saritha Namboodiri1, Varsha T. G. IOT Based System For Smart And Secured Home// IRJET, – (2018) [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/NuojVn3 (дата звертання 07.03.2021)

29. Lalit Mohan Satapathy. Arduino based home automation using Internet of things (IoT)// International Journal of Pure and Applied Mathematics. – Volume 118 No. 17, 2018, pp 769-778

30. Климат контроль для камеры созревания сыров [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/Ouoj3Ie (дата звертання 07.03.2021)

31. Климат контроль для теплиц любых размеров и конструкций "Терраформ" [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/WuokoLf (дата звертання 07.03.2021)

32. Климатический комплекс Zenet ZET-472 [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/tuokgKA (дата звертання 07.03.2021)

33. Arduino nano v3 datasheet [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/4t2cjeu (дата звертання 14.03.2021)

34. GY-302 datasheet [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/Qt2czHG (дата звертання 14.03.2021)

35. GY-49 datasheet [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/LuolhHq (дата звертання 14.03.2021)

36. DHT11 datasheet [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/luolzGl (дата звертання 14.03.2021)

37. AM2320 datasheet [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/Dt2cQq5 (дата звертання 14.03.2021)

38. DHT21 datasheet [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/5uolQmN (дата звертання 14.03.2021)

39. MQ-135 datasheet [Electronic resource] - Resource access mode: https://cutt.ly/Zt2cR1L (дата звертання 14.03.2021)

40. MG-811 datasheet [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/muolUyx (дата звертання 16.06.2020)

41. ESP8266-01 datasheet [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/1uolP4U (дата звертання 14.03.2021)

42. 8-канальне реле [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/huol85k (дата звертання 14.03.2021)

43. Твердотільне реле [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/wuolCHB (дата звертання 14.03.2021)

44. How to determine milk quality with a pH meter and conductometer [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/xlX1luc (дата звертання 14.03.2021)

45. Аналоговий датчик PH для Arduino [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/hlX0nAl (дата звертання 14.03.2021)

46. Аналоговый датчик измерения электропроводности для Ардуино [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/FlBlX1x (дата звертання 14.03.2021)

47. Unified Modeling Language [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/UvUoXJM (дата звертання 14.03.2021)

48. Использование интерфейса I2C в Arduino [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/dvqMaZX (дата звертання 14.03.2021)

49. Подключение аналоговых датчиков к Ардуино [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/Fvwq7s1 (дата звертання 14.03.2021)

50. Последовательный порт UART в Ардуино [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/Lvwomht (дата звертання 14.03.2021)

51. Internet of Things (IoT) connected devices installed base worldwide [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/PvUaN3p (дата звертання 14.03.2021)

52. Fog Computing vs. Cloud Computing for IoT Projects [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/OvUajt6 (дата звертання 14.03.2021)

53. C. Wang, M. Daneshmand, M. Dohler, X. Mao, R. Q. Hu, and H. Wang, "Guest editorial - special issue on Internet of things (IoT): Architecture, protocols and services," IEEE Sensors Journal, vol. 13, no. 10, pp. 3505–3510, Oct. 2013.

54. Cognizant, "How the Internet of things will overcome a lack of standards," 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/RvUhWWv (дата звертання 25.03.2021)

55. Internet of things undermined by a lack of standards, warns Pentaho VP EMEA Paul Scholey 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/MvUhJ7Z (дата звертання 25.03.2021)

56. J. Newman, "Why the Internet of things might never speak A common language," in App Economy, Fast Company, 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/0vUjRjs (дата звертання 25.03.2021)

57. J. Hope, "The biggest problem with the Internet of things," in Tech.co 2015. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/svUj2ep (дата звертання 25.03.2021)

58. D. Price, "Solving Standardisation of the Internet of things," in cloudtweaks, 2015. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/kvUkf7n (дата звертання 25.03.2021)

59. "Architecture of an Interoperable IoT Platform Based on Microservices," MIPRO, 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/yvUkzAr (дата звертання 25.03.2021)

60. Microsoft, "Data lake," 2015. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/NvUkH9V (дата звертання 25.03.2021)

61. Building an IR bridge with the SmartThings ThingShield, Hackster.io, 2013. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/TvUlroh (дата звертання 25.03.2021)

62. Microsoft, "Microsoft Azure IoT Reference Architecture," Microsoft Corporation, 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/gvUzaa1 (дата звертання 25.03.2021)

63. "Azure IoT suite | Microsoft azure," in azure.microsoft, 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/DvUcaaO (дата звертання 25.03.2021)

64. D. Betts, "What is azure IoT hub?," in azure.microsoft, 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/kvUcbp0 (дата звертання 25.03.2021)

65. D. Betts, "Azure IoT Hub developer guide," in azure.microsoft, 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/AvUvZzG (дата звертання 25.03.2021)

66. "The best cloud platform - Microsoft azure vs. AWS," in azure.microsoft, 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/IvUbEWv (дата звертання 25.03.2021)

67. SmartThings, "How it works," SmartThings, 2016 [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/GvUbXmP (дата звертання 25.03.2021)

68. 6LoWPAN," in Wikipedia, Wikimedia Foundation, 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/UvUnqSq (дата звертання 25.03.2021)

69. "Intelligent living," in Smart home, Samsung, 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/BvUnfGk (дата звертання 25.03.2021)

70. "SmartThings recent failures in stability and support," in SmartThings, SmartThings Community, 2015. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/nvUnWh7 (дата звертання 25.03.2021)

71. A. Chakrabarti, "Thread – an open standard protocol for home automation," in infoQ, InfoQ, 2015. . [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/EvUnXy1 (дата звертання 25.03.2021)

72. D. Sung, "Thread, ZigBee, Z-Wave: Why smart home standards matter," Wareable, 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/mvUn0v7 (дата звертання 25.03.2021)

73. "Thread," in threadgroup, 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://www.threadgroup.org/ (дата звертання 25.03.2021)

74. W. Colitti, "The battle for Smart Home networking," in Walter Colitti, 2015. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/gvUQELz (дата звертання 25.03.2021)

75. L. LABS, "Thread vs. ZigBee (for IoT engineers)," in IOT Networks, Link Labs, 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/1vUQZB8 (дата звертання 25.03.2021)

76. P. Traverse, I. Lacaze, and J. Souyris, “Airbus Fly-By-Wire: A total approach to dependability,” in Building the Inform. Soc., ser. IFIP Intl. Federation for Inform. Process. Springer, 2004, no. 156, pp. 191–212.

77. The transport layer security (TLS) protocol [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/TLS (дата звертання 25.03.2021)

78. P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, “The many faces of publish/subscribe,” ACM Comput. Surv., vol. 35, no. 2, pp. 114–131, Jun. 2003.

79. A. Banks and R. Gupta, “MQTT version 3.1.1,” OASIS Standard [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/hvUUpiA (дата звертання 25.03.2021)

80. T. Dierks and E. Rescorla, “The transport layer security (TLS) protocol version 1.2,” RFC 5246, IETF, Aug. 2008.

81. Let`s Encrypt [Електронний ресурс] – Режим доступу до ресурсу: https://letsencrypt.org/ (дата звертання 25.03.2021)

82. T. Kothmayr, C. Schmitt, W. Hu, M. Br¨unig, and G. Carle, “DTLS based security and two-way authentication for the Internet of Things,” Ad Hoc Networks, vol. 11, no. 8, pp. 2710–2723, Nov. 2013.

83. E. Rescorla and N. Modadugu, “Datagram transport layer security version 1.2,” RFC 6347, IETF, Jan. 2012.

84. Z. Shelby, K. Hartke, and C. Bormann, “The constrained application protocol (CoAP),” RFC 7252, IETF, Jun. 2014.

85. A Secure Network Architecture for the Internet of Things Based on Local Authorization Entities // IEEE 4th International Conference on Future Internet of Things and Cloud Aug. 2016.

86. C. Neuman, T. Yu, S. Hartman, and K. Raeburn, “The Kerberos network authentication service (V5),” RFC 4120, IETF, Jul. 2005.

87. L. Zhu and B. Tung, “Public key cryptography for initial authentication in Kerberos (PKINIT),” RFC 4556, IETF, Jun. 2006.

88. TerraSwarm [Електронний ресурс] – Режим доступу до ресурсу: https://www.terraswarm.org/ (дата звертання 25.03.2021)

89. W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney, “A key management scheme for wireless sensor networks using deployment knowledge,” in INFOCOM 2004. Twenty-third Annu. Joint Conf. of the IEEE Comput. and Commun. Societies, vol. 1, Mar. 2004, p. 597.

90. W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, “A pairwise key predistribution scheme for wireless sensor networks,” ACM Trans. Inf. Syst. Secur., vol. 8, no. 2, pp. 228–258, May 2005.

91. F. Gandino, B. Montrucchio, and M. Rebaudengo, “Key management for static wireless sensor networks with node adding,” IEEE Trans. on Industrial Informatics, vol. 10, no. 2, pp. 1133–1143, May 2014.

92. X. He, M. Niedermeier, and H. de Meer, “Dynamic key management in wireless sensor networks: A survey,” Journal of Network and Computer Applications, vol. 36, no. 2, pp. 611–622, Mar. 2013.

93. X. Zhang, J. He, and Q. Wei, “EDDK: Energy-efficient distributed deterministic key management for wireless sensor networks,” EURASIP Journal on Wirel. Commun. Netw., vol. 2011, pp. 12:1–12:11, Jan. 2011.

94. J.-Y. Huang, I.-E. Liao, and H.-W. Tang, “A forward authentication key management scheme for heterogeneous sensor networks,” EURASIP Journal on Wirel. Commun. Netw., vol. 2011, pp. 6:1–6:10, Jan. 2011.

95. O. K. Sahingoz, “Large scale wireless sensor networks with multi-level dynamic key management scheme,” Journal of Systems Architecture, vol. 59, no. 9, pp. 801–807, Oct. 2013.

96. S. H. Erfani, H. H. Javadi, and A. M. Rahmani, “A dynamic key management scheme for dynamic wireless sensor networks,” Security and Communication Networks, vol. 8, no. 6, pp. 1040–1049, Apr. 2015.

97. Arduino Nano: распиновка, схема подключения и программирование [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/CuokKWV (дата звертання 12.04.2021)

98. WiFi ESP-07 чип ESP8266 [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/lvUAECX (дата звертання 12.04.2021)

99. Qt Creator [Електронний ресурс] – Режим доступу до ресурсу: https://www.qt.io/ (дата звертання 12.04.2021)

Додаток А

МОДЕЛЬ СИСТЕМИ ІОТ РІШЕНЬ ДЛЯ ПРОЦЕСУ СИРОВАРІННЯ

Текст програми для МК Arduino

Листів 3

Розробник _____ Руслан КУЧЕРЕНКО

Н

Київ, 2021

Текст програми для мікроконтролера Arduino Nano V3

```
// подключаем библиотеку I2C:
#include <Wire.h>
//подключаем библиотеку датчика AM2320:
#include <AM2320.h>
// объявляем объект th
AM2320 th;
// подключаем библиотеку датчика BH1750:
#include <BH1750.h>
// объявляем объект lightMeter:
BH1750 lightMeter;
//gas sensor
const int gas_lvl = A0; //пин уровеня газа
const int gas_ind = 2; //пин наличия газа
boolean noGas; //переменная для хранения значения о присутствии газа
int gasValue = 0; //переменная для хранения количества газа
//инициализация управления
#define led_heat 12
#define led_cooling 11
#define led_dehum 10
#define led_hum 9
#define led_fan 8
#define l_light 7
#define led_light 6
void setup() {
 Serial.begin(9600);
 //лед управление
 pinMode (led_heat, OUTPUT);
 pinMode (led_cooling, OUTPUT);
 pinMode (led_dehum, OUTPUT);
 pinMode (led_hum, OUTPUT);
 pinMode (l_light, OUTPUT);
 pinMode (led_fan, OUTPUT);
 pinMode (led_light, OUTPUT);
 //индикатор газа
 pinMode(gas_ind, INPUT); //установка режима пина
 digitalWrite(led_heat, LOW);
 digitalWrite(led_cooling, LOW);
 digitalWrite(led_dehum, LOW);
 digitalWrite(led_hum, LOW);
 digitalWrite(l_light, LOW);
 digitalWrite(led_fan, LOW);
 //датчик освещения
  lightMeter.begin();      //инициализация датчика BH1750
}
void loop() {
  //модуль газа
 gasValue = analogRead(gas_lvl); // и о его количестве
 Serial.print(gasValue);
 Serial.print(",");
 //модуль температуры/влажности
 switch(th.Read()) {
   case 2:
     Serial.println("CRC failed");
     break;
   case 1:
     Serial.println("Sensor offline");
     break;
   case 0:
     Serial.print(th.h);
     Serial.print(",");
     Serial.print(th.t);
     Serial.print(",");
     break;
 }
 //проверка освещенности
  digitalWrite (led_light, HIGH); // включить светодиод
  //считываем показания с BH1750:
  uint16_t lux = lightMeter.readLightLevel();
  //выводим показания в послед. порт:
  Serial.print(String(lux));
  Serial.print(",");
 //запись в порт
  if (Serial.available())
  {
    char main_signal = Serial.read();
     if (main_signal == '1')
      {
       digitalWrite (led_heat, LOW); // включить светодиод
       digitalWrite (led_cooling, HIGH); // включить светодиод
       }
    else if (main_signal == '2')
    {
     digitalWrite (led_cooling, LOW); // выключить светодиод
     digitalWrite (led_heat, HIGH); // включить светодиод
     }
    else if (main_signal == '3')
    {
     digitalWrite (led_hum, LOW); // включить светодиод
     digitalWrite (led_dehum, HIGH); // включить светодиод
    }
    else if (main_signal == '4')
    {
      digitalWrite (led_dehum, LOW); // выключить светодиод
      digitalWrite (led_hum, HIGH); // включить светодиод
    }
    else if (main_signal == '5')
    {
      digitalWrite (led_fan, HIGH); // включить светодиод
    }
    else if (main_signal == '6')
    {
       digitalWrite (led_fan, LOW); // включить светодиод
    }
    else if (main_signal == '7')
    {
       digitalWrite (l_light, HIGH); // включить светодиод
    }
    else if (main_signal == '8')
```

```
    {
        digitalWrite (l_light, LOW); // включить
светодиод
    }
    else if (main_signal == '9')
    {
     digitalWrite(led_heat, LOW);
     digitalWrite(led_cooling, LOW);
     digitalWrite(led_dehum, LOW);
     digitalWrite(led_hum, LOW);
     digitalWrite(l_light, LOW);
     digitalWrite(led_fan, LOW);
    }
    else
    {
Serial.print("Error");
     }
  }
delay(200);
}
```

Додаток Б

МОДЕЛЬ СИСТЕМИ ІОТ РІШЕНЬ ДЛЯ ПРОЦЕСУ СИРОВАРІННЯ

Текст програми для ОС Windows

Листів 9

Розробник _____ Руслан КУЧЕРЕНКО

Н

Київ, 2021

Текст програмного забезпечення для ОС Windows

main.pro

```
QT       += core gui serialport
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
CONFIG += c++11
# The following define makes your compiler emit warnings if you use
# any Qt feature that has been marked deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS
# You can also make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000
# disables all the APIs deprecated before Qt 6.0.0
SOURCES += \
    main.cpp \
    mainwindow.cpp
HEADERS += \
    mainwindow.h
FORMS += \
    mainwindow.ui
# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

main.cpp

```
#include "mainwindow.h"
#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    //w.setFixedSize(634,366);
    w.setWindowTitle("Climate Control Tool");
    w.show();
    return a.exec();
}
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QSerialPort>
#include <QSerialPortInfo>
#include <QDebug>
#include <QtWidgets>
#include <QString>
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    QPixmap pic("D:/Qt_proj/test_Ar/led_gr_off.png");
    MainWindow::ui->heat_im->setFixedSize(30,30);
    MainWindow::ui->heat_im->setScaledContents(true);
    MainWindow::ui->heat_im->setPixmap(pic);
    //
    MainWindow::ui->cool_im->setFixedSize(30,30);
    MainWindow::ui->cool_im->setScaledContents(true);
    MainWindow::ui->cool_im->setPixmap(pic);
    //
    MainWindow::ui->hum_im->setFixedSize(30,30);
    MainWindow::ui->hum_im->setScaledContents(true);
    MainWindow::ui->hum_im->setPixmap(pic);
    //
    MainWindow::ui->dehum_im->setFixedSize(30,30);
    MainWindow::ui->dehum_im->setScaledContents(true);
    MainWindow::ui->dehum_im->setPixmap(pic);
    //
    MainWindow::ui->fan_im->setFixedSize(30,30);
    MainWindow::ui->fan_im->setScaledContents(true);
    MainWindow::ui->fan_im->setPixmap(pic);
    //
    MainWindow::ui->light_im->setFixedSize(30,30);
    MainWindow::ui->light_im->setScaledContents(true);
    MainWindow::ui->light_im->setPixmap(pic);

    arduino_is_available = false;
    arduino_port_name = "";

    arduino = new QSerialPort;
    serialBuffer = "";

    foreach (const QSerialPortInfo &serialPortInfo, QSerialPortInfo::availablePorts()){
        if(serialPortInfo.hasVendorIdentifier() && serialPortInfo.hasProductIdentifier()){
            if(serialPortInfo.vendorIdentifier() == arduino_nano_vendor_id){
                if(serialPortInfo.productIdentifier() == arduino_nano_product_id){
                    arduino_port_name = serialPortInfo.portName();
                    arduino_is_available = true;
                }
            }
        }
    }

    if(arduino_is_available){
        //open and configure the serialport
        arduino->setPortName(arduino_port_name);
        arduino->open(QSerialPort::ReadWrite); //режим порта!
        arduino->setBaudRate(QSerialPort::Baud9600);
        arduino->setDataBits(QSerialPort::Data8);
        arduino->setParity(QSerialPort::NoParity);
        arduino->setStopBits(QSerialPort::OneStop);
        arduino->setFlowControl(QSerialPort::NoFlowControl);
```

```
        QObject::connect(arduino,
SIGNAL(readyRead()), this, SLOT(readSerial()));

    }else{
        //dive error message if not available
        QMessageBox::warning(this, "Port error",
"Couldn`t find the Arduino!");
    }

//connect(Load_Butt,SIGNAL(clicked()),this,SLOT(on
_Load_Butt_clicked()));
   //
connect(Start_Butt,SIGNAL(clicked()),this,SLOT(on_
Start_Butt_clicked()));
}
MainWindow::~MainWindow()
{
    if(arduino->isOpen()){
        arduino->close();
    }
    delete ui;
}
void MainWindow::on_Load_Butt_clicked()
{
    str_hum_opt = ui->line_hum_opt->text();
    str_temp_opt = ui->line_temp_opt->text();
    str_co2_opt = ui->line_co2_opt->text();
    qDebug() << "Hum: " << str_hum_opt;
    qDebug() << "T: " << str_temp_opt;
    qDebug() << "CO2: " << str_co2_opt;
}
void MainWindow::on_Start_Butt_clicked()
{
    QPixmap pic("D:/Qt_proj/test_Ar/led_gr_off.png");
    QPixmap pic1("D:/Qt_proj/test_Ar/led_gr_on.png");
    //temp
    if(bufferT>str_temp_opt.toFloat())
    {
        MainWindow::ui->heat_im->setPixmap(pic);
        MainWindow::ui->cool_im->setPixmap(pic1);
        led_data = "1";
        MainWindow::writeSerial(led_data);
        qDebug() << "Охлаждение " << led_data;
    }else{
        MainWindow::ui->heat_im->setPixmap(pic1);
        MainWindow::ui->cool_im->setPixmap(pic);
        led_data = "2";
        MainWindow::writeSerial(led_data);
        qDebug() << "Нагрев  " << led_data;
    }
    //hum
    if(bufferH>str_hum_opt.toFloat())
    {
        MainWindow::ui->hum_im->setPixmap(pic);
        MainWindow::ui->dehum_im->setPixmap(pic1);
        led_data = "3";
        MainWindow::writeSerial(led_data);
    }else{
        MainWindow::ui->hum_im->setPixmap(pic1);
        MainWindow::ui->dehum_im->setPixmap(pic);
        led_data = "4";
        MainWindow::writeSerial(led_data);
    }
    //co2
    if(bufferCO>str_co2_opt.toFloat())
    {
        MainWindow::ui->fan_im->setPixmap(pic1);
        led_data = "5";
        MainWindow::writeSerial(led_data);
    }else{
        MainWindow::ui->fan_im->setPixmap(pic);
        led_data = "6";
        MainWindow::writeSerial(led_data);
    }
    //lux
    if(bufferLx>100)
    {
        MainWindow::ui->light_im->setPixmap(pic1);
        led_data = "7";
        MainWindow::writeSerial(led_data);
    }else{
        MainWindow::ui->light_im->setPixmap(pic);
        led_data = "8";
        MainWindow::writeSerial(led_data);
    }
}
void MainWindow::on_Stop_Butt_clicked()
{
    QPixmap pic("D:/Qt_proj/test_Ar/led_gr_off.png");
    MainWindow::ui->heat_im->setPixmap(pic);
    MainWindow::ui->cool_im->setPixmap(pic);
    MainWindow::ui->hum_im->setPixmap(pic);
    MainWindow::ui->dehum_im->setPixmap(pic);
    MainWindow::ui->fan_im->setPixmap(pic);
    MainWindow::ui->light_im->setPixmap(pic);
    led_data = "9";
    MainWindow::writeSerial(led_data);
}
void MainWindow::writeSerial(QString data)
{
    if(arduino->isWritable()){
        arduino->write(data.toStdString().c_str());
    }else{
        qDebug() << "Couldn`t write to serial!";
    }
}
void MainWindow::readSerial()
{
    bufferSplit = serialBuffer.split(",");
    if(bufferSplit.length() < 5){
        serialData = arduino->readAll();
        serialBuffer +=
QString::fromStdString(serialData.toStdString());
    }else{
        //bufferSplit[1] is a good value
        qDebug() << bufferSplit;
        MainWindow::updateV_CO(bufferSplit[0]);
        MainWindow::updateV_H(bufferSplit[1]);
        MainWindow::updateV_T(bufferSplit[2]);
        serialBuffer = "";
        bufferCO = bufferSplit[0].toFloat();
        bufferT = bufferSplit[2].toFloat();
        bufferH = bufferSplit[1].toFloat();
        bufferLx = bufferSplit[3].toFloat();
    }
}
```

```
void MainWindow::updateV_T(const QString
sensor_reading)
{
   ui->temp_labe->setText(sensor_reading);
}
void MainWindow::updateV_H(const QString
sensor_reading)
{
   ui->hum_lable->setText(sensor_reading);
}
void MainWindow::updateV_CO(const QString
sensor_reading)
{
   ui->co2_lable->setText(sensor_reading);
}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QSerialPort>
#include <QPushButton>
#include <QLineEdit>
#include <QString>
#include <QPixmap>
QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE
class MainWindow : public QMainWindow
{
   Q_OBJECT
public:
   MainWindow(QWidget *parent = nullptr);
  ~MainWindow();
private slots:
   void on_Load_Butt_clicked();
   void on_Start_Butt_clicked();
   void on_Stop_Butt_clicked();
   void readSerial();
   void writeSerial(QString);
   void updateV_T(const QString);
   void updateV_H(const QString);
   void updateV_CO(const QString);
private:
   Ui::MainWindow *ui;
   QSerialPort *arduino;
   static const quint16 arduino_nano_vendor_id =
6790;
   static const quint16 arduino_nano_product_id =
29987;
   QString arduino_port_name;
   bool arduino_is_available;
   QByteArray serialData;
   QString serialBuffer;
   QLineEdit *line_hum_opt;
   QLineEdit *line_temp_opt;
   QLineEdit *line_co2_opt;
   QString str_hum_opt;
   QString str_temp_opt;
   QString str_co2_opt;
   QStringList bufferSplit;
   float bufferCO;
   float bufferT;
   float bufferH;
   float bufferLx;
   QString led_data;
   //QString *str_hum_opt;
};
#endif // MAINWINDOW_H
```

mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow"
name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>661</width>
    <height>470</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="Line" name="line_5">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>160</y>
      <width>621</width>
      <height>16</height>
     </rect>
    </property>
    <property name="orientation">
     <enum>Qt::Horizontal</enum>
    </property>
   </widget>
   <widget class="QLabel" name="label_10">
    <property name="geometry">
     <rect>
      <x>9</x>
      <y>264</y>
      <width>631</width>
      <height>51</height>
     </rect>
    </property>
    <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-size:11pt; font-weight:600;&quot;&gt;Индикация
активных
процессов&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/h
tml&gt;</string>
    </property>
   </widget>
   <widget class="QWidget"
name="gridLayoutWidget">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>310</y>
      <width>621</width>
      <height>111</height>
```

```
    </rect>
   </property>
   <layout class="QGridLayout"
name="gridLayout_4">
    <item row="1" column="8">
     <widget class="QLabel" name="fan_im">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;right&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;/&gt;&lt;/p&gt;&lt;/body&gt;&lt;/h
tml&gt;</string>
      </property>
     </widget>
    </item>
    <item row="1" column="10">
     <widget class="QLabel" name="light_im">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;right&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;/&gt;&lt;/p&gt;&lt;/body&gt;&lt;/h
tml&gt;</string>
      </property>
     </widget>
    </item>
    <item row="0" column="8">
     <widget class="QLabel" name="fan_lb">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;&gt;Вентиляция&lt;/span&gt;&lt;/
p&gt;&lt;/body&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="1" column="6">
     <widget class="QLabel" name="dehum_im">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;right&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;/&gt;&lt;/p&gt;&lt;/body&gt;&lt;/h
tml&gt;</string>
      </property>
     </widget>
    </item>
    <item row="0" column="1">
     <widget class="Line" name="line_10">
      <property name="orientation">
       <enum>Qt::Vertical</enum>
      </property>
     </widget>
    </item>
    <item row="1" column="2">
     <widget class="QLabel" name="cool_im">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;right&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;/&gt;&lt;/p&gt;&lt;/body&gt;&lt;/h
tml&gt;</string>
      </property>
     </widget>
    </item>
    <item row="0" column="3">
     <widget class="Line" name="line_9">
      <property name="orientation">
       <enum>Qt::Vertical</enum>
      </property>
     </widget>
    </item>
    <item row="0" column="7">
     <widget class="Line" name="line_14">
      <property name="orientation">
       <enum>Qt::Vertical</enum>
      </property>
     </widget>
    </item>
    <item row="0" column="6">
     <widget class="QLabel" name="dehum_lb">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;&gt;Осушение&lt;/span&gt;&lt;/p
&gt;&lt;/body&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="1" column="3">
     <widget class="Line" name="line_8">
      <property name="orientation">
       <enum>Qt::Vertical</enum>
      </property>
     </widget>
    </item>
    <item row="1" column="1">
     <widget class="Line" name="line_11">
      <property name="orientation">
       <enum>Qt::Vertical</enum>
      </property>
     </widget>
    </item>
    <item row="0" column="4">
     <widget class="QLabel" name="hum_lb">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;&gt;Увлажнение&lt;/span&gt;&lt;/
p&gt;&lt;/body&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="1" column="0">
     <widget class="QLabel" name="heat_im">
```

```
      <property name="text">
       <string/>
      </property>
     </widget>
    </item>
    <item row="0" column="9">
     <widget class="Line" name="line_2">
      <property name="orientation">
       <enum>Qt::Vertical</enum>
      </property>
     </widget>
    </item>
    <item row="0" column="0">
     <widget class="QLabel" name="light_lb_2">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;&gt;Нагрев&lt;/span&gt;&lt;/p&gt;
&lt;/body&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="1" column="4">
     <widget class="QLabel" name="hum_im">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;right&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;/&gt;&lt;/p&gt;&lt;/body&gt;&lt;/h
tml&gt;</string>
      </property>
     </widget>
    </item>
    <item row="0" column="2">
     <widget class="QLabel" name="cool_lb">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;&gt;Охлаждение&lt;/span&gt;&lt;/
p&gt;&lt;/body&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="0" column="10">
     <widget class="QLabel" name="light_lb">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;&gt;Освещение&lt;/span&gt;&lt;/p
&gt;&lt;/body&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="0" column="5">
     <widget class="Line" name="line_12">
      <property name="orientation">
       <enum>Qt::Vertical</enum>
      </property>
     </widget>
    </item>
    <item row="1" column="7">
     <widget class="Line" name="line_4">
      <property name="orientation">
       <enum>Qt::Vertical</enum>
      </property>
     </widget>
    </item>
    <item row="1" column="5">
     <widget class="Line" name="line_13">
      <property name="orientation">
       <enum>Qt::Vertical</enum>
      </property>
     </widget>
    </item>
    <item row="1" column="9">
     <widget class="Line" name="line_6">
      <property name="orientation">
       <enum>Qt::Vertical</enum>
      </property>
     </widget>
    </item>
   </layout>
  </widget>
  <widget class="QWidget" name="">
   <property name="geometry">
    <rect>
     <x>19</x>
     <y>9</y>
     <width>621</width>
     <height>41</height>
    </rect>
   </property>
   <layout class="QHBoxLayout"
name="horizontalLayout_3">
    <item>
     <widget class="QLabel" name="label_5">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-size:11pt; font-
weight:600;&quot;&gt;Номинальные
значения&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/ht
ml&gt;</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QLabel" name="label_6">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-size:11pt; font-weight:600;&quot;&gt;Текущие
значения
&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</s
tring>
      </property>
     </widget>
```

```
    </item>
   </layout>
  </widget>
  <widget class="QWidget" name="">
   <property name="geometry">
    <rect>
     <x>50</x>
     <y>60</y>
     <width>221</width>
     <height>101</height>
    </rect>
   </property>
   <layout class="QGridLayout"
name="gridLayout_3">
    <item row="1" column="0">
     <widget class="QLabel" name="label_23">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-size:10pt; font-
weight:600;&quot;&gt;Температура:&lt;/span&gt;&lt
;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="0" column="0">
     <widget class="QLabel" name="label_16">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
&gt;&lt;span style=&quot; font-size:10pt; font-
weight:600;&quot;&gt;Влажность:&lt;/span&gt;&lt;/
p&gt;&lt;/body&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="2" column="0">
     <widget class="QLabel" name="label_25">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-size:10pt; font-weight:600;&quot;&gt;Уровень
CO&lt;/span&gt;&lt;span style=&quot; font-size:10pt;
font-weight:600; vertical-
align:sub;&quot;&gt;2&lt;/span&gt;&lt;span
style=&quot; font-size:10pt; font-
weight:600;&quot;&gt;:&lt;/span&gt;&lt;/p&gt;&lt;/b
ody&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="2" column="2">
     <widget class="QLabel" name="label_26">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;&gt;мг/дм&lt;/span&gt;&lt;span
style=&quot; font-weight:600; vertical-
align:super;&quot;&gt;3
&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</s
tring>
      </property>
     </widget>
    </item>
    <item row="0" column="2">
     <widget class="QLabel" name="label_22">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-size:10pt; font-weight:600;&quot;&gt;%
&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</s
tring>
      </property>
     </widget>
    </item>
    <item row="1" column="1">
     <widget class="QLineEdit"
name="line_temp_opt"/>
    </item>
    <item row="1" column="2">
     <widget class="QLabel" name="label_24">
      <property name="text">
       <string>&lt;!DOCTYPE HTML PUBLIC &quot;-
//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-
html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta
name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot;
font-family:'MS Shell Dlg 2'; font-size:8pt; font-
weight:400; font-style:normal;&quot;&gt;
&lt;p align=&quot;center&quot; style=&quot; margin-
top:12px; margin-bottom:12px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-
indent:0px;&quot;&gt;&lt;span style=&quot; font-
size:11pt; font-weight:600; vertical-
align:super;&quot;&gt;o&lt;/span&gt;&lt;span
style=&quot; font-size:11pt; font-
weight:600;&quot;&gt;C
&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</s
tring>
      </property>
     </widget>
    </item>
    <item row="2" column="1">
     <widget class="QLineEdit"
name="line_co2_opt"/>
    </item>
    <item row="0" column="1">
     <widget class="QLineEdit"
name="line_hum_opt"/>
    </item>
   </layout>
  </widget>
  <widget class="QWidget" name="">
   <property name="geometry">
    <rect>
     <x>390</x>
     <y>60</y>
```

```
     <width>211</width>
     <height>101</height>
    </rect>
   </property>
   <layout class="QGridLayout" name="gridLayout">
    <item row="0" column="1">
     <widget class="QLabel" name="hum_lable">
      <property name="font">
       <font>
        <weight>75</weight>
        <bold>true</bold>
       </font>
      </property>
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;/&gt;&lt;/body&gt;&lt;/html
&gt;</string>
      </property>
     </widget>
    </item>
    <item row="1" column="2">
     <widget class="QLabel" name="label_20">
      <property name="text">
       <string>&lt;!DOCTYPE HTML PUBLIC &quot;-
//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-
html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta
name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot;
font-family:'MS Shell Dlg 2'; font-size:8pt; font-
weight:400; font-style:normal;&quot;&gt;
&lt;p align=&quot;center&quot; style=&quot; margin-
top:12px; margin-bottom:12px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-
indent:0px;&quot;&gt;&lt;span style=&quot; font-
size:11pt; font-weight:600; vertical-
align:super;&quot;&gt;o&lt;/span&gt;&lt;span
style=&quot; font-size:11pt; font-
weight:600;&quot;&gt;C
&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</s
tring>
      </property>
     </widget>
    </item>
    <item row="1" column="1">
     <widget class="QLabel" name="temp_labe">
      <property name="font">
       <font>
        <weight>75</weight>
        <bold>true</bold>
       </font>
      </property>
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;br/&gt;&lt;/p&gt;&
lt;/body&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="2" column="2">
     <widget class="QLabel" name="label_19">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-
weight:600;&quot;&gt;мг/дм&lt;/span&gt;&lt;span
style=&quot; font-weight:600; vertical-
align:super;&quot;&gt;3
&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</s
tring>
      </property>
     </widget>
    </item>
    <item row="2" column="0">
     <widget class="QLabel" name="label_17">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-size:10pt; font-weight:600;&quot;&gt;Уровень
CO&lt;/span&gt;&lt;span style=&quot; font-size:10pt;
font-weight:600; vertical-
align:sub;&quot;&gt;2&lt;/span&gt;&lt;span
style=&quot; font-size:10pt; font-
weight:600;&quot;&gt;:&lt;/span&gt;&lt;/p&gt;&lt;/b
ody&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="1" column="0">
     <widget class="QLabel" name="label_18">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-size:10pt; font-
weight:600;&quot;&gt;Температура:&lt;/span&gt;&lt
;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="0" column="0">
     <widget class="QLabel" name="label_15">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
&gt;&lt;span style=&quot; font-size:10pt; font-
weight:600;&quot;&gt;Влажность:&lt;/span&gt;&lt;/
p&gt;&lt;/body&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item row="0" column="2">
     <widget class="QLabel" name="label_21">
      <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;span style=&quot;
font-size:10pt; font-weight:600;&quot;&gt;%
```

```
&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
       </property>
      </widget>
     </item>
     <item row="2" column="1">
      <widget class="QLabel" name="co2_lable">
       <property name="font">
        <font>
         <weight>75</weight>
         <bold>true</bold>
        </font>
       </property>
       <property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;center&quot;&gt;&lt;br/&gt;&lt;/p&gt;&
lt;/body&gt;&lt;/html&gt;</string>
       </property>
      </widget>
     </item>
    </layout>
   </widget>
   <widget class="Line" name="line">
    <property name="geometry">
     <rect>
      <x>320</x>
      <y>10</y>
      <width>20</width>
      <height>151</height>
     </rect>
    </property>
    <property name="orientation">
     <enum>Qt::Vertical</enum>
    </property>
   </widget>
   <widget class="QWidget" name="">
    <property name="geometry">
     <rect>
      <x>19</x>
      <y>178</y>
      <width>621</width>
      <height>91</height>
     </rect>
    </property>
    <layout class="QGridLayout"
name="gridLayout_2">
     <item row="0" column="2">
      <layout class="QVBoxLayout"
name="verticalLayout_7">
       <item>
        <widget class="QPushButton"
name="Start_Butt">
         <property name="text">
          <string>Старт</string>
         </property>
        </widget>
       </item>
       <item>
        <widget class="QPushButton"
name="Stop_Butt">
         <property name="text">
          <string>Стоп</string>
         </property>
        </widget>
       </item>
      </layout>
     </item>
     <item row="0" column="0">
      <widget class="QPushButton"
name="Load_Butt">
       <property name="text">
        <string>Загрузить значения</string>
       </property>
      </widget>
     </item>
     <item row="0" column="1">
      <widget class="Line" name="line_3">
       <property name="orientation">
        <enum>Qt::Vertical</enum>
       </property>
      </widget>
     </item>
    </layout>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menubar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>661</width>
     <height>21</height>
    </rect>
   </property>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
 </widget>
 <resources/>
 <connections/>
</ui>
```

Додаток В

МОДЕЛЬ СИСТЕМИ ІОТ РІШЕНЬ ДЛЯ ПРОЦЕСУ СИРОВАРІННЯ

Інструкція користувачеві

Листів 3

Розробник _____ Руслан КУЧЕРЕНКО

Н

Київ, 2021

Інструкція користувачеві

1. Підключити систему клімат-контролю до ПК.
2. Запуск програмного забезпечення (рис.В1):

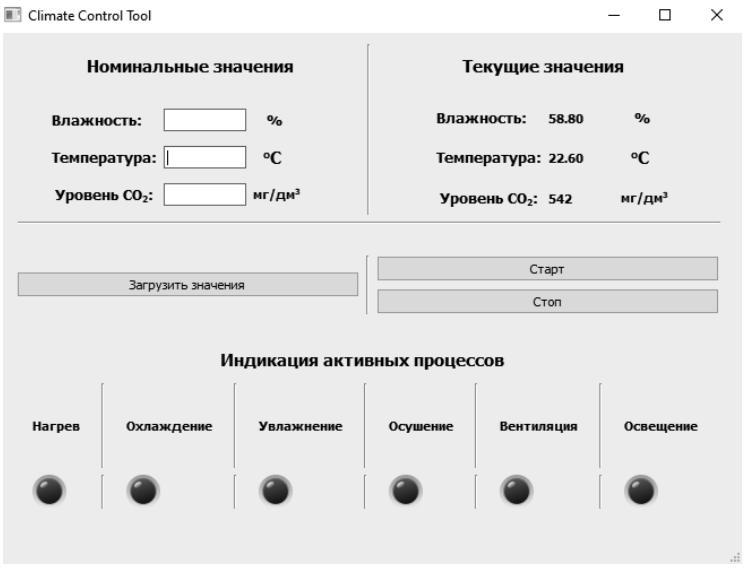


Рисунок В1 – Графічний інтерфейс програми клімат-контролю

3. Вказуємо номінальні значення температури, вологості та рівня CO_2 в залежності від типу сиру та натискаємо «Загрузить значения» (рис.В2):

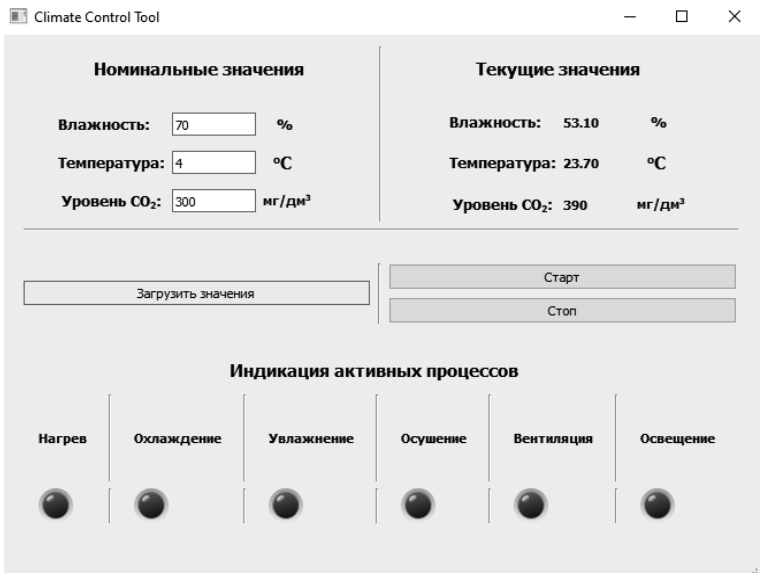


Рисунок В2 – Завантаження номінальних даних до програми

4. Натискаємо кнопку «Старт» (рис.В3):

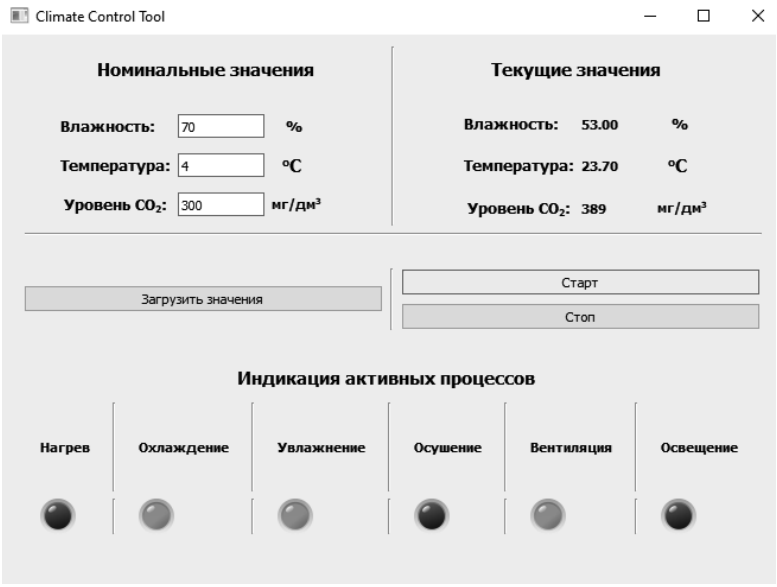


Рисунок В3 – Запуск програмного забезпечення

За необхідності зміни номінальних значень натиснути на кнопку «Стоп» на повторити пункти 3,4.

Додаток Г

МОДЕЛЬ СИСТЕМИ ІОТ РІШЕНЬ ДЛЯ ПРОЦЕСУ СИРОВАРІННЯ

Графічні зображення

Листів 11

Розробник _____ Руслан КУЧЕРЕНКО

Н

Київ, 2021

Зміст

Рисунок Г 1 – Слайд 1 Тема доповіді

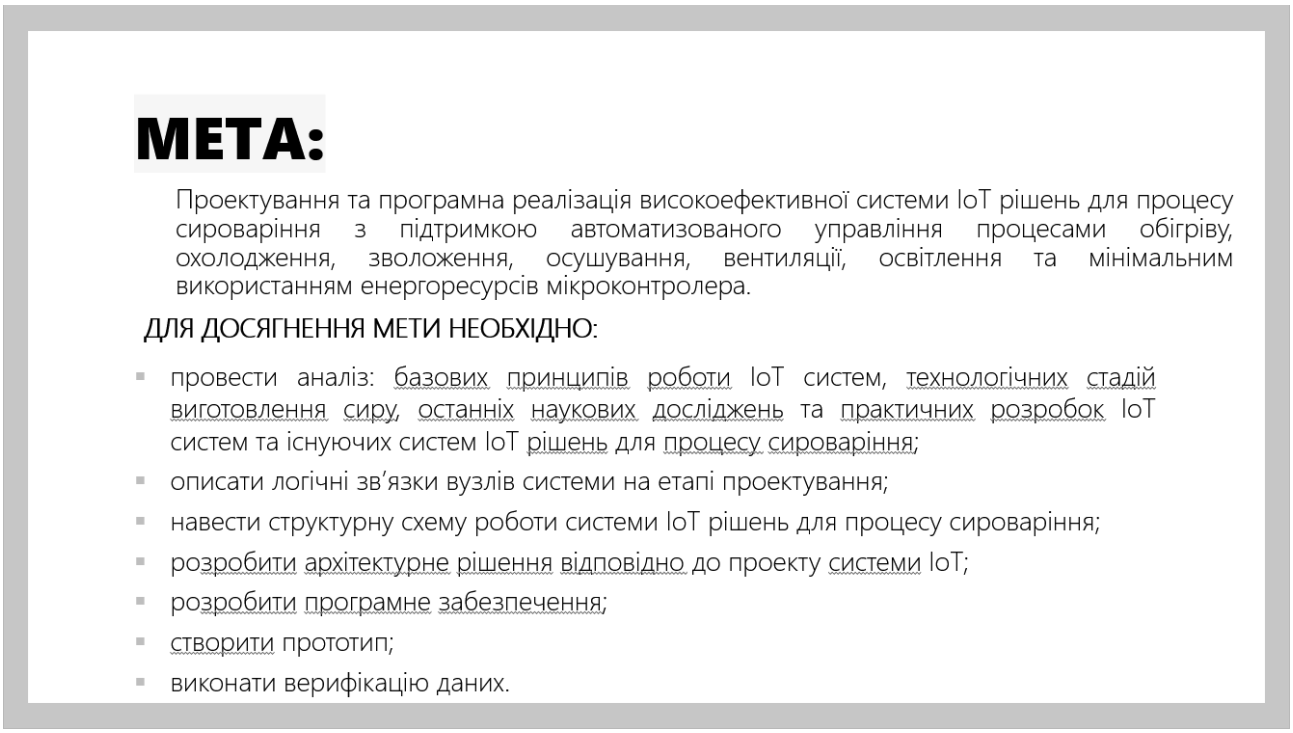


Рисунок Г 2 – Слайд 2 Мета роботи

ОГЛЯД АНАЛОГІВ

НАЗВА СИСТЕМИ	КРИТЕРІЇ ПОРІВНЯННЯ					
	ПРИЗНАЧЕННЯ	МОЖЛИВОСТІ	ХАРАКТЕРИСТИКИ	НЕДОЛІКИ	ЦІНА	Вирішує
«Модульна блок-установка»	Клімат-контроль для камери дозрівання сирів	Забезпечення необхідних режимів температури та вологості повітря.	• Підтримка температурного режиму від 10 °C до 20 °C • підтримка вологості повітря до 99%; • споживана потужність близько 2,2 кВт; • об'єм камери від 30 м3 до 60 м3	Один варіанту реалізації без використання апаратно-програмних технологій.	1000 доларів	3 задачі
«Терраформ»	Клімат-контроль для теплиць	Управління опаленням, поливом, освітленням та вентиляцією. Віддалений контроль.	• 8 реле для керування пристроями • потужність навантаження реле 1 кВт • 4 давачі температури ґрунту, рідини • 4 давачі вологості ґрунту • 4 давачі освітленості • 6)2 давачі концентрації CO_2	Низька потужність навантаження на реле – всього 1 кВт.	800 доларів	6 задач
«Zenet»	Кліматичний комплекс для житлових приміщень	Обігрів, зволоження, охолодження, вентиляція та очищення повітря. Працює на «акумуляторах» з льоду.	• Споживана потужність 65 Вт; • зниження температури на 7 градусів • мобільність	Регулярна заміна «акумуляторів» з льоду, ручне налаштування та малий функціонал.	150 доларів	4 задачі

Рисунок Г 3 – Слайд 3 Огляд існуючих аналогів

СТРУКТУРНА СХЕМА ПРИСТРОЮ

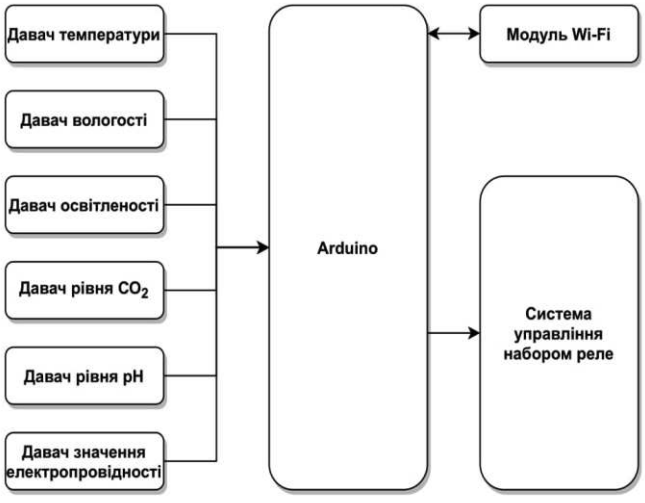


Рисунок Г4 – Слайд 4 Схема структурна пристрою

Рисунок Г 5 – Слайд 5 UML-діаграма використання

Рисунок Г 6 – Слайд 6 Архітектура системи

ХМАРНІ ТЕХНОЛОГІЇ

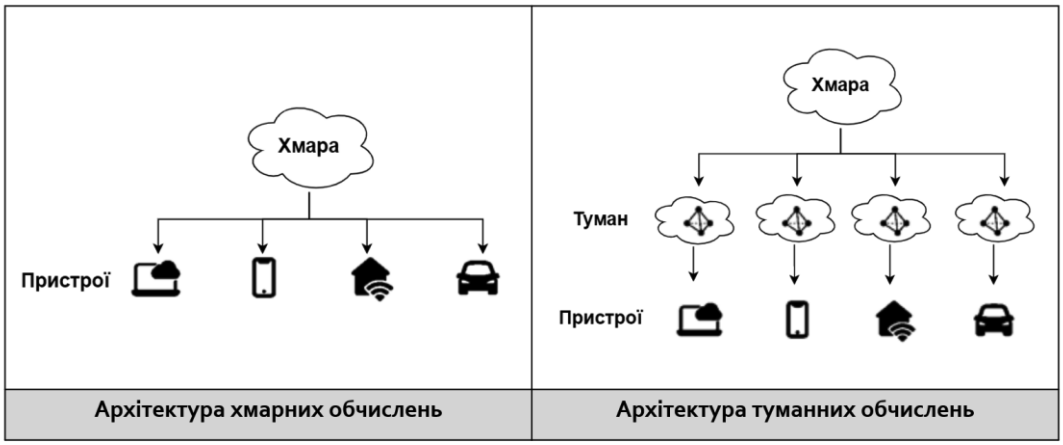


Рисунок Г 7 – Слайд 7 Хмарні технології

ПОРІВНЯННЯ

	Хмарні обчислення	Туманні обчислення
Архітектура	Централізована	Розподілена
Комунікація з пристроями	Здалеку	Безпосередньо біля пристроїв
Обробка даних	Далеко від джерела інформації	Близько до джерела інформації
Обчислювальні можливості	Вищі	Нижчі
Кількість вузлів	Декілька	Дуже багато
Час аналізу	Тривалий період	Короткий період
Латентність	Висока	Низька
Підключення	Інтернет	Різні протоколи та стандарти
Безпека	Нижча	Вища

Рисунок Г 8 – Слайд 8 Порівняння туманних та хмарних обчислень

Рисунок Г 9 – Слайд 9 Структура системи

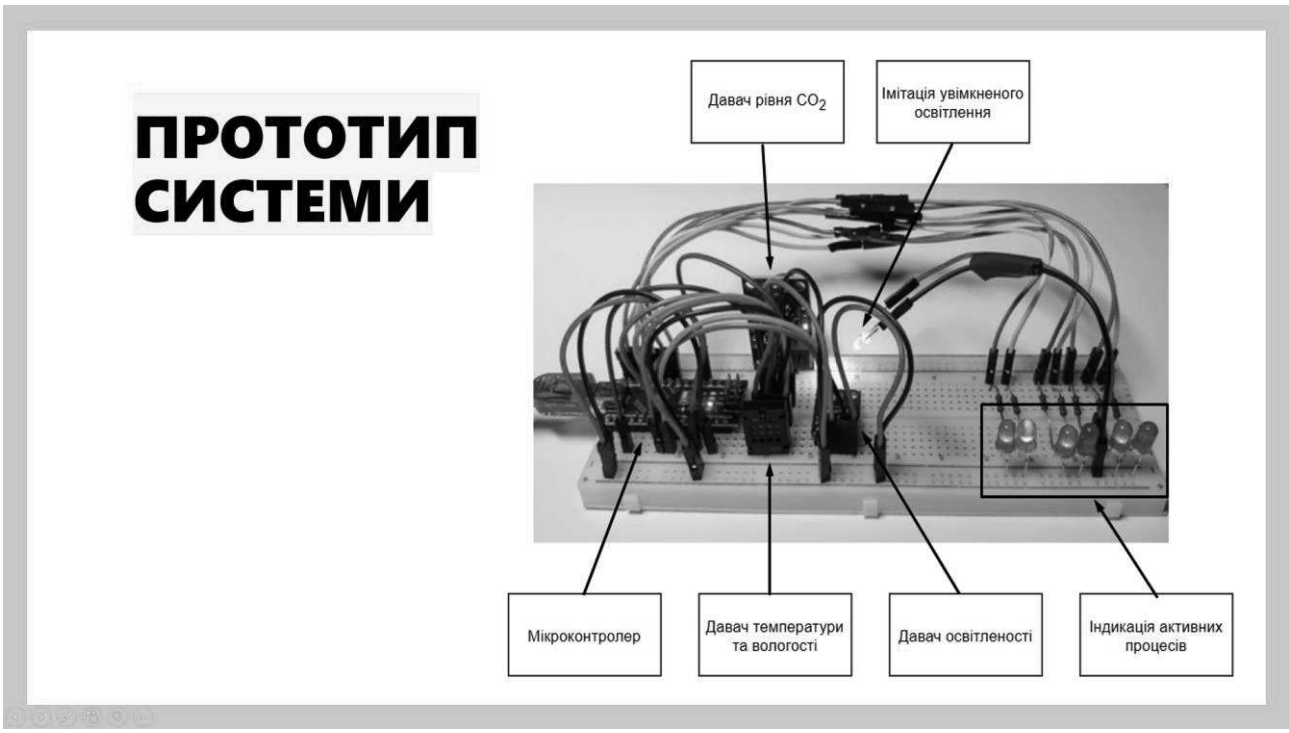


Рисунок Г 10 – Слайд 10 Прототип системи

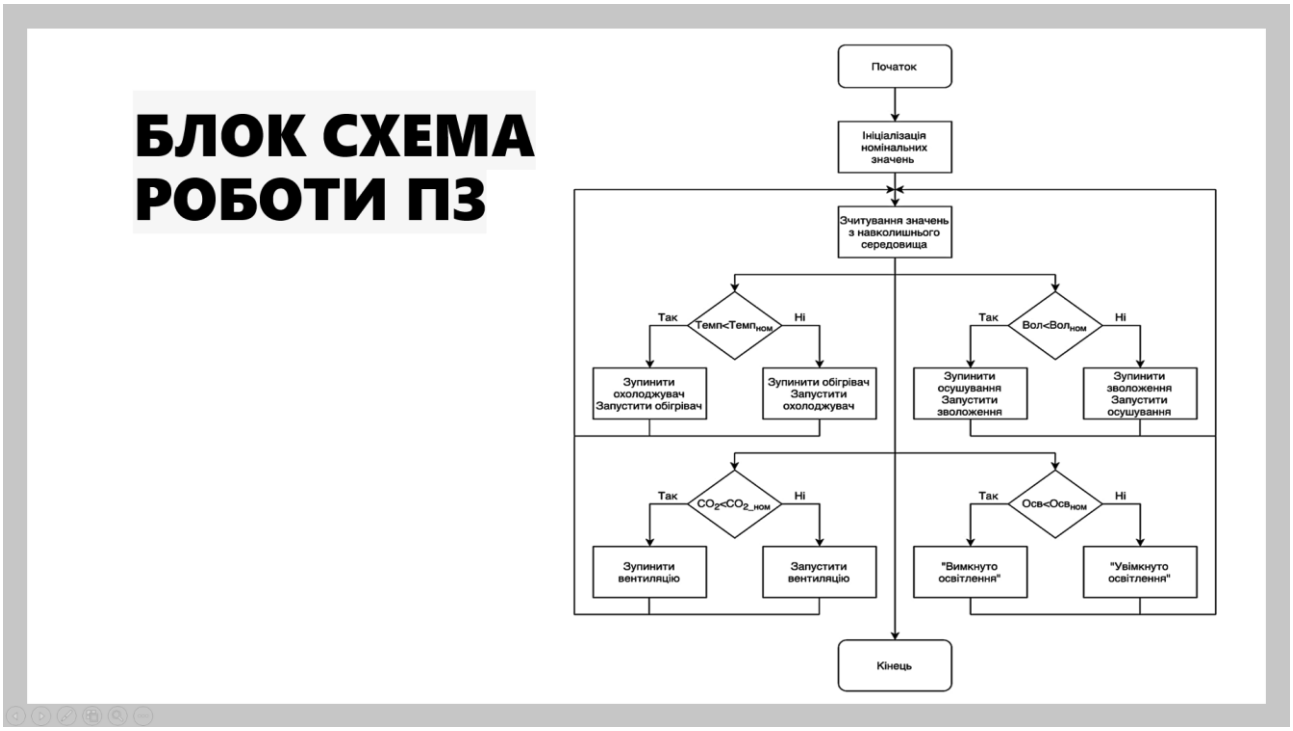


Рисунок Г 11 – Слайд 11 Блок-схема роботи ПЗ

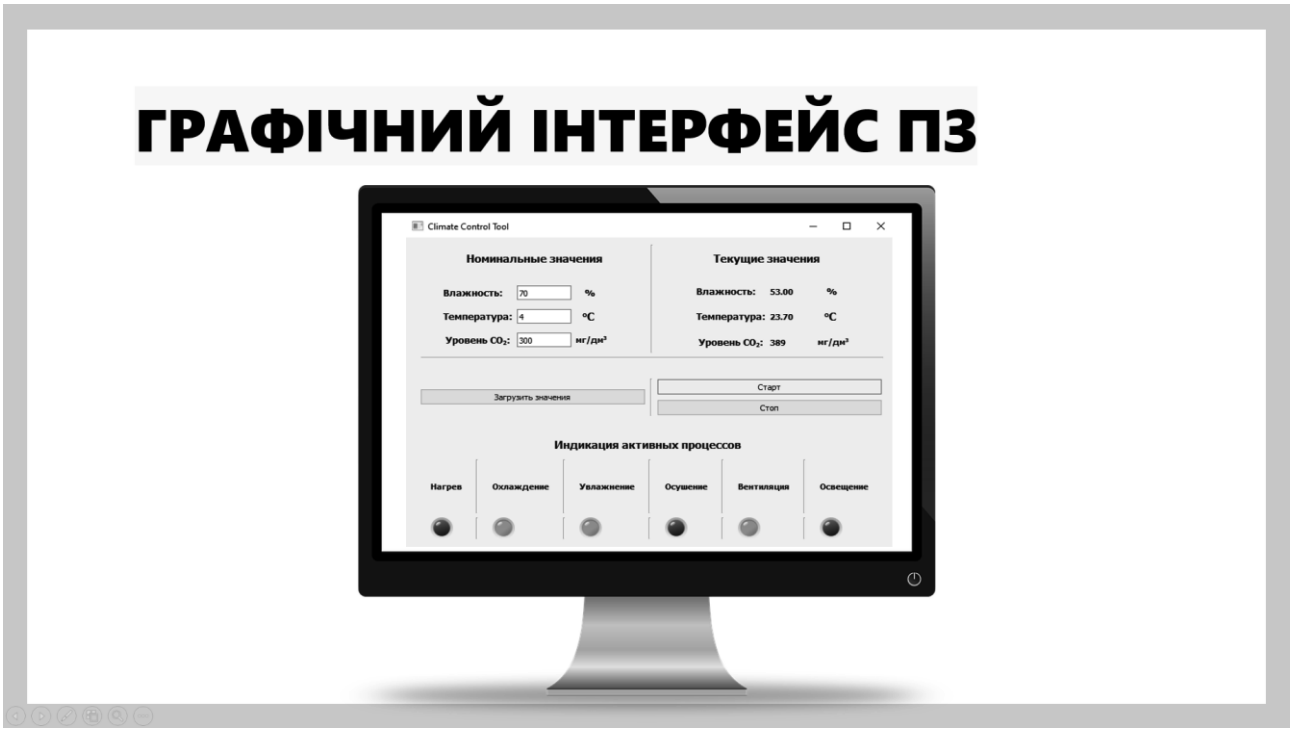
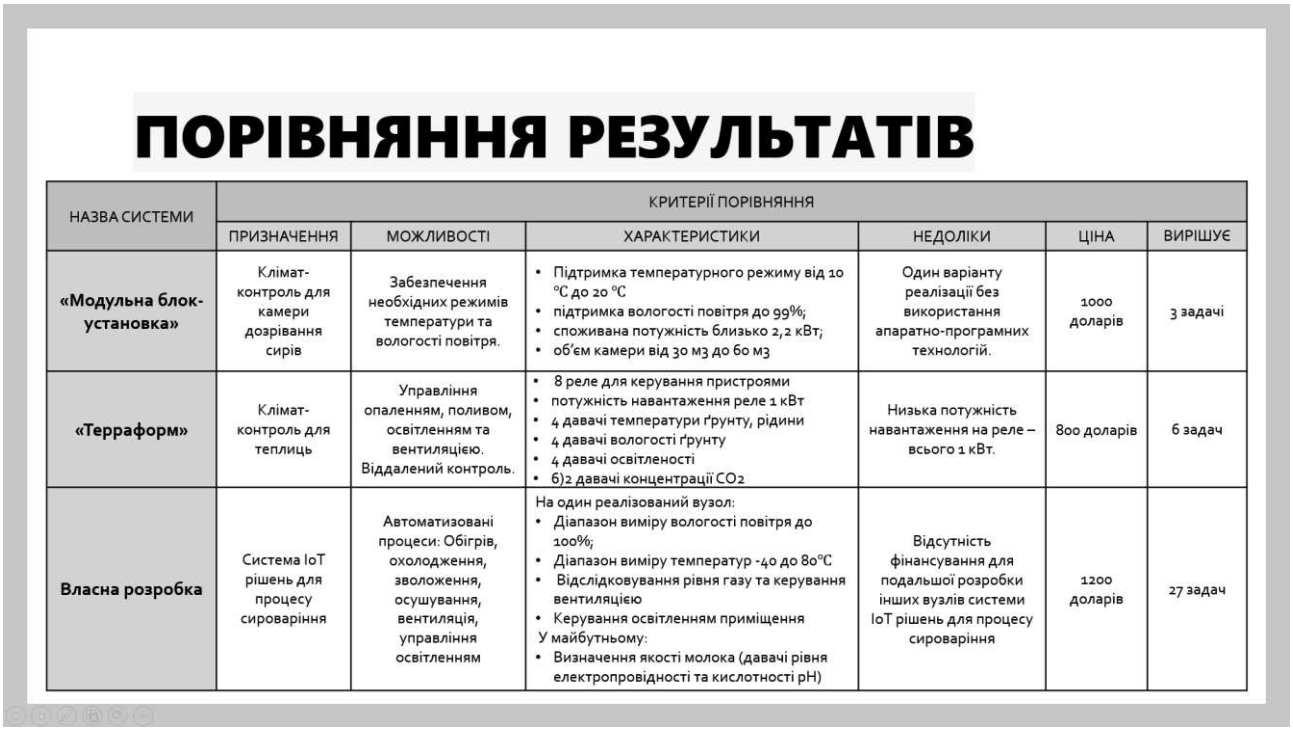


Рисунок Г 12 – Слайд 12 Графічний інтерфейс ПЗ



НАЗВА СИСТЕМИ	КРИТЕРІЇ ПОРІВНЯННЯ					
	ПРИЗНАЧЕННЯ	МОЖЛИВОСТІ	ХАРАКТЕРИСТИКИ	НЕДОЛІКИ	ЦІНА	ВИРІШУЄ
«Модульна блок-установка»	Клімат-контроль для камери дозрівання сирів	Забезпечення необхідних режимів температури та вологості повітря.	• Підтримка температурного режиму від 10 °С до 20 °С • підтримка вологості повітря до 99%; • споживана потужність близько 2,2 кВт; • об'єм камери від 30 м3 до 60 м3	Один варіанту реалізації без використання апаратно-програмних технологій.	1000 доларів	3 задачі
«Терраформ»	Клімат-контроль для теплиць	Управління опаленням, поливом, освітленням та вентиляцією. Віддалений контроль.	• 8 реле для керування пристроями • потужність навантаження реле 1 кВт • 4 давачі температури ґрунту, рідини • 4 давачі вологості ґрунту • 4 давачі освітленості • б)2 давачі концентрації CO_2	Низька потужність навантаження на реле – всього 1 кВт.	800 доларів	6 задач
Власна розробка	Система IoT рішень для процесу сироваріння	Автоматизовані процеси: Обігрів, охолодження, зволоження, осушування, вентиляція, управління освітленням	На один реалізований вузол: • Діапазон виміру вологості повітря до 100%; • Діапазон виміру температур -40 до 80°С • Відслідковування рівня газу та керування вентиляцією • Керування освітленням приміщення У майбутньому: • Визначення якості молока (давачі рівня електропровідності та кислотності pH)	Відсутність фінансування для подальшої розробки інших вузлів системи IoT рішень для процесу сироваріння	1200 доларів	27 задач

Рисунок Г 13 – Слайд 13 Порівняння власної розробки з аналогами

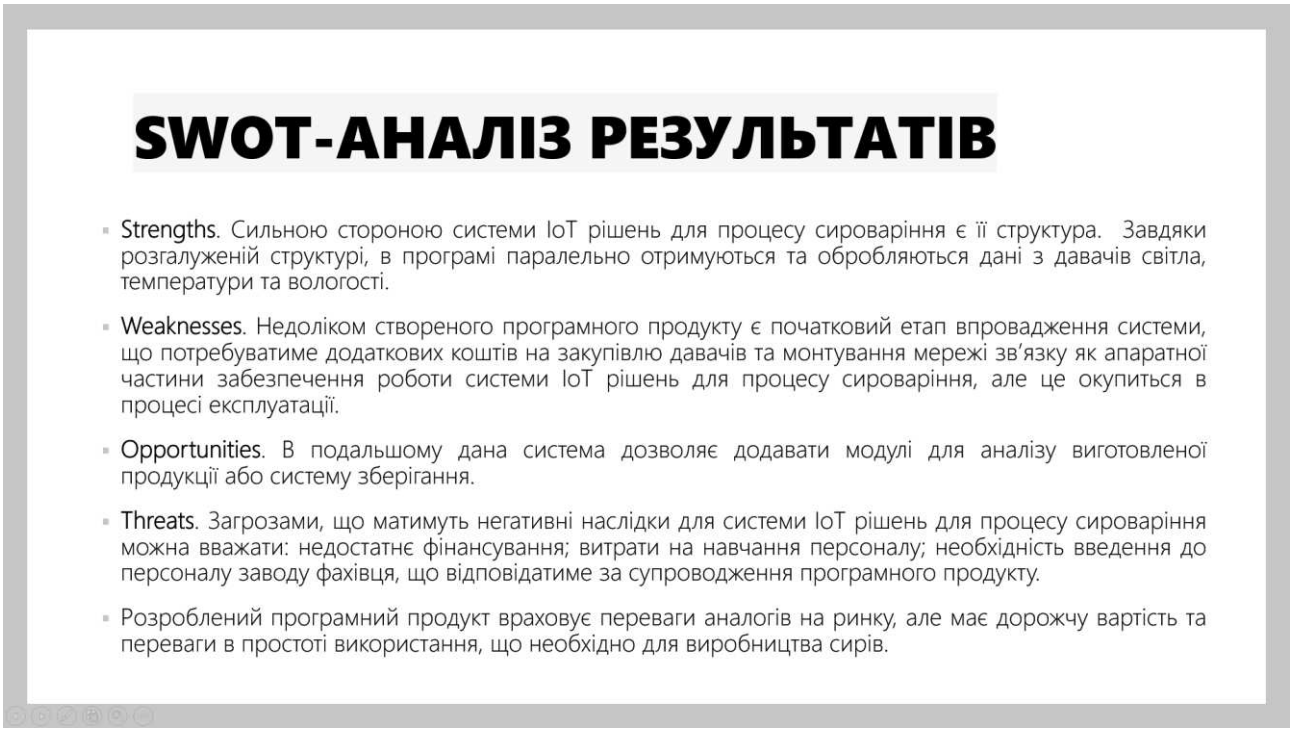


Рисунок Г 14 – Слайд 14 SWOT-аналіз результатів

Рисунок Г 15 – Слайд 15 Висновки

Рисунок Г 16 – Слайд 16 Публікації результатів

Рисунок Г 17 – Слайд 17 Дякую за увагу