Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» грудня 2022 р.

Пояснювальна записка

до кваліфікаційної роботи
магістра
на тему: «**Модель «розумної» науково-дослідницької лабораторії»**

Захищено на засіданні
Атестаційної комісії № 44
протокол № __ від __.12.2022 р.
Оцінка _____ / _____
Голова Атестаційної комісії
д.т.н., професор
_____**МІНУХІН**
Сергій Володимирович

Виконав:

студент 2 курсу, групи КІ– 61
за спеціальністю 121 – Комп'ютерна інженерія.
Галузь знань: 12 – Інформаційні технології
ВОЛИНСЬКИЙ Валентин Володимирович

Керівник:

д.т.н., професор кафедри теоретичної та прикладної системотехніки
МПРОШНИК Марина Анатоліївна

Рецензент:

д.т.н., доцент кафедри спеціалізованих комп'ютерних систем
ДОЦЕНКО Сергій Ілліч

Харків – 2022

АНОТАЦІЯ

Кваліфікаційна робота містить вступ, 3 розділи, висновок, список літератури та додатки. Об'єм роботи складає 93 сторінок, з яких 70 сторінок складають основну частину, що містить 91 рисунків, 30 використаних джерел для дослідження на 3 сторінки і 4 додатки на 16 сторінок.

Мета наукової роботи полягає в створенні «розумної» науково-дослідницької лабораторії

Об'єкт дослідження: технологія Інтернету речей та «розумного дому».

Результат дослідження: модель науково-дослідницької лабораторії, що може імітувати поведінку в реальних системах та здатна самостійно приймати рішення, в залежності від впливу зовнішніх факторів та сценаріїв роботи, що можуть бути налаштовані дистанційно.

В процесі дослідження була створена модель розумної науково-дослідницької лабораторії на основі біологічної та хімічної лабораторій з використанням пристроїв, випущених Cisco. Для моделювання був використаний додаток Cisco Packet Tracer, який дозволяє в реальному часі відстежувати поведінку IoT пристроїв в залежності від сценаріїв роботи, на які впливають фактори зовнішнього середовища. Емулятор дозволяє легко створювати специфічні пристрої Інтернету речей, які направлені на вирішення конкретних вузьконаправлених задач та інтегрувати такі пристрої в уже створену мережу.

Ключові слова: *Internet of Things, DHCP, DNS, Python, JavaScript, IP, hoisting, модель, router, CName, WiFi, 3G/4G, Cisco Packet Tracer, smart home*

ABSTRACT

The work has an introduction, three sections, conclusion, lists of sources and four appendices. The total volume is 93 pages, of which 70 are the pages of the main part, which contains 91 pictures, 30 used sources on 3 pages and 4 appendices on 16 pages.

The purpose of scientific work is to create a «smart» research laboratory

Research object: Internet of Things and «smart home» technology.

The result of the study: a model of a research laboratory that can simulate the behavior of real systems and is able to make decisions independently, depending on the influence of external factors and work scenarios that can be configured remotely.

In the process of research, a model of a smart research laboratory was created based on biological and chemical laboratories using devices produced by Cisco. The Cisco Packet Tracer application was used for simulation, which allows real–time monitoring of the behavior of IoT devices depending on work scenarios that are affected by environmental factors. The emulator allows you to easily create specific devices of the Internet of Things, which are aimed at solving specific, narrowly focused tasks, and dynamically integrate such devices into an already created network.

Keywords: *Internet of Things, DHCP, DNS, Python, JavaScript, IP, hoisting, model, router, CName, WiFi, 3G/4G, Cisco Packet Tracer, smart home*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

.py – Python файл.

.js – JavaScript файл.

IoT – Internet of Things.

LAN – Local Area Network.

Fa – Fast Ethernet.

DNS – Domain Name Server.

HTTP – HyperText Transfer Protocol.

TCP – Transmission Control Protocol.

GNS – Graphical Network Simulator

Wi–Fi – Wireless Fidelity.

4G – 4 Generation.

UDP – User Datagram Protocol.

CO – Carbon monoxide.

DHCP – Dynamic Host Configuration Protocol.

ВСТУП

Сучасні інформаційні технології розвиваються з надзвичайної швидкістю. З'являються нові чи покращуються уже існуючі протоколи передачі даних, зростає швидкість та безпека передачі інформації дистанційно. Це все сприяє розвитку сучасних технологій. Яскравим результатом такого розвитку є концепція Internet of Things. Ця концепція і різні типи автоматизації, які існують саме завдяки IoT допомагають автоматично, на рівні мікроконтролерів чи мікропроцесорів вирішувати повсякденні задачі як в житті кожної людини, так і системах державного чи промислового рівня.

Системи, розроблені за допомогою Інтернету речей характеризуються зручним та швидким локальним чи віддаленим керування, інтуїтивно-зрозумілим інтерфейсом керування, який може налаштовуватися під конкретного користувача чи групи користувачів, адаптивністю та масштабованістю, забезпеченням автоматизації потреб різних систем чи людей, що економить не тільки сили, а і людино–години роботи, що критично важливо для підприємств чи державних структур. Такі системи забезпечують автоматичне прийняття рішень навіть в найкритичніші ситуації шляхом використання найсучасніших інформаційних технологій.

Для створення таких складно організованих систем найчастіше використовують підходи «розумний дім» чи «індустріальний Інтернет речей». Звичайно, максимального результату можна досягти в комбінації цих двох принципів. Створення загальноприйнятих протоколів взаємодії пристроїв по одній чи по розподілених мережах, впровадження 5G та Wi-Fi 6 мереж здешевлює використання підходів в повсякденному житті.

Створення програмних моделей систем домашньої автоматизації дозволяє ще на етапі моделювання знайти всі проблеми, з якими користувач може зіткнутися вже у готових розгорнутих реальних системах. Саме для таких цілей було створено мережеві додатки, такі як Cisco Packet Tracer чи GNS3. Який з

додатків використовувати слід вирішувати після дослідження, який тип моделі буде проектуватися.

Можна сказати, що створення програмної моделі «розумної» науково-дослідницької лабораторії актуальна задача. Тому що постійний розвиток сенсорних та мікросервісних технологій, машинного навчання, 5G робить технологію Інтернету речей все більш доступною і дає їх змогу виконувати задачі не тільки по домашній автоматизації, а і бути більш гнучкою та адаптивною, дозволяючи інтегрувати себе практично в будь-яку сфера науки чи навчання і візуально показати, як сучасні технології можуть покращити процеси в наукових чи експериментальних дослідження, збільшуючи напрацювання і рухаючи інформаційний і технічний розвиток суспільства вперед.

Мета дослідження: покращити процес проведення наукових досліджень та експериментів в різних типах лабораторій з використання концепції Інтернету речей.

Завдання дослідження:

- Аналіз сучасного Інтернету речей;
- моделювання науково-дослідницької лабораторії з використанням додатку по створенню комп'ютерних мереж;
- створення необхідних пристроїв для моделі та розробити методику їх тестування;
- комплексне тестування розробленої моделі і розробити інструкції по користуванню моделлю на основі показаних результатів;

Об'єкт дослідження: концепція Інтернету речей та система домашньої автоматизації.

Предмет дослідження: модель «розумної» науково-дослідницької лабораторії, змодельованої за допомогою сучасного додатку для створення комп'ютерних мереж.

РОЗДІЛ І
АНАЛІЗ КОНЦЕПЦІЇ INTERNET OF THINGS

1.1 Особливості використання Internet of Things

Інтернет речей створений для того, щоб об'єднувати пристрої в одну стабільну мережу, забезпечувати передавання даних між компонентами через спеціальне програмне забезпечення, додавати чи видаляти практично будь-яке технічне обладнання. Роком народження Інтернету речей можна вважати 1999 рік. Але бурхливий розвиток та всесвітню відомість концепція отримала тільки в 2010 році. Спеціалісти з США вважають, що IoT став однією з шести проривних технологій за останні 20 років[12]. Ось уже 12 років щорік проводиться науково-технічна конференція в Брюсселі, де найбільші корпорації демонструють найновітніші напрацювання в області IoT.

Принцип роботи Інтернету речей простий – це система, яка здатна до самомасштабування і працює автономно. Такі системи здатні самостійно обробляти великі об’єми даних, які вони отримують від серверів, або від один одного, займаючись обміном даних в рамках однієї системи. Вони здатні працювати, використовуючи дані з хмарних сервісів (Amazon, Alibaba, Azure) за допомогою підключення по Wi-Fi, Bluetooth, MQTT чи будь–якого іншого типу зв’язку, який підтримує розгорнута система.

Компоненти, з яких складається Інтернет речей можна записати як абревіатуру ABCDE. Розглянемо її значення.

1) A - analytics– вважається головною частиною Інтернету речей, яка здатна об’єднати пристрою в рамках спільної бізнес-логіки;
2) B - BigData – стосується інформації, яку може зберігати системи, наприклад в хмарі. Ці дані можна використовувати в навчанні системи чи в майбутньому прийнятті рішень.
3) C - connection – описуються канали та способи передачі інформації в рамках конкретної працюючої системи;

4) D - devices – характеризуються пристрої, які здатні функціонувати і приймати рішення в залежності від факторів зовнішнього середовища чи налаштованих сценаріїв роботи;
5) E - experience – пристрої здатні зберігати та обмінюватися даними, на основі який рішення можуть прийматися групою пристроїв.

Концепція Інтернету речей використовується всюди. Кожен з нас стикався з нею в магазині, в транспорті, на вулиці і навіть вдома, не усвідомлюючи цього. На рисунку 1.1 зображено розподіл ринку Інтернету речей на 2021 рік[12].

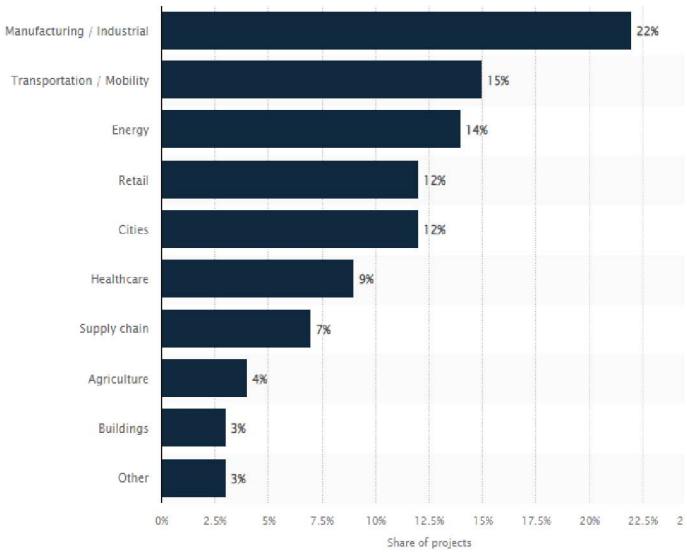


Рисунок 1.1 – Сучасний ринок Інтернету речей.

Отже можна сказати, що індустріальний Інтернет речей наразі являється найпопулярнішим. І це не дивно, адже це дозволяє автоматизувати безліч процесів, зменшити відходи, підвищити ефективність та швидкість виробу продукції, здешевити логістику, покращити умови працювання, мінімізувати людський фактор в небезпечних чи важконавантажених системах, організувати контроль електропередач чи цілих підстанцій.

Але є недоліки в концепції Інтернету речей. Перш за все, це зловмисники, які хочуть скористатися тим, шо технологія ще розвивається, отже в ній обов’язкову знайдуться вразливості. Атаки хакерів можуть негативно впливати навіть на цілу систему IoT. Тому у країнах, де Інтернет речей використовується

дуже активно хакерські атаки по типу: DDos атак, фішинг, XSS атак – це не новина. Саме через загрози втрати даних чи порушення цілісності системи, компанії, які займаються розробкою пристроїв IoT в першу чергу звертають увагу саме на організацію безпеки. Іншим недоліком є те, шо поки що компанії не використовують однакові протоколи при створенні мереж. Тобто пристрої від різних виробників можуть конфліктувати або взагалі не працювати через несумісність ПО. І найбільшим недоліком Інтернету речей є те, що інтеграція таких технологій обов'язково призведе до втрати робочих місць звичайних людей. Машини часто краще справляють з тривіальними задачами, тому доцільно та дешевше поставити керувати і приймати рішення в таких системах не людей, в машини чи програми. Та незважаючи на це все, технологія IoT має дуже високі перспективи. Ріст популярності можна побачити на рисунку 1.2 [13].

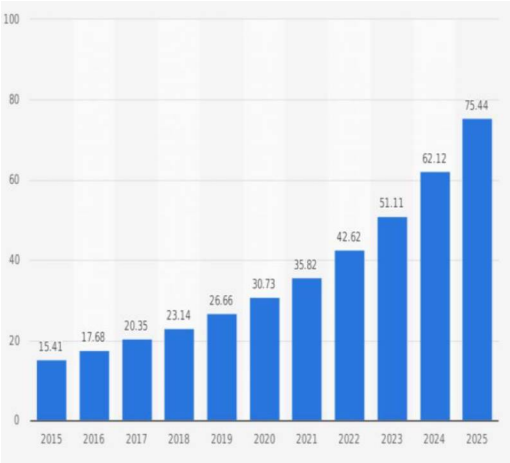


Рисунок 1.2 – Прогнозований ріст Інтернету речей.

Тобто, кожен рік популярність росте приблизно на 20%. Це чи не найвищий показник росту серед сучасних інформаційних технологій.

Отже, можна сказати, що через свою «молодість» концепція Інтернету речей має надзвичайно великий потенціал розвитку і інтеграції в усі сфери життя в майбутні роки. Тому вивчати, принципи побудови, передачі інформації, способи створення своїх «розумних» пристроїв варто почати вже зараз. Адже настане час, що технологія стане настільки доступною, що кожна людина зможе

по максимуму використовувати всі переваги систем домашньої автоматизації наповну.

1.2 Загальні вимоги до сучасних систем домашньої автоматизації

Згідно з науковими висновками стосовно Інтернету речей, система домашньої автоматизації повинна задовольняти наступні критерії[13]:

1) масштабованість. Під цим поняттям розуміється, що будь-куди можна додати чи видалити будь-який пристрій без втрати функціоналу або з ціллю додати щось нове;
2) фінансова доступність – говорить про те, що таку систему може дозволити практично будь-хто;
3) адаптивність всіх компонентів системи та додатку для керування цими системами – мається на увазі, системи не мають бути заточені тільки під одного клієнта. Будь-яку систему можна налаштувати на потреби клієнта чи на фактори зовнішнього середовища, де ця система буде монтуватися;
4) підтримка надійних та швидких протоколів передачі інформації;
5) можливості інтеграції продукції декількох виробників в одну систему – наразі з цим є проблеми, тому що розробники пока що не використовують однакові підходи щодо розробки свої товарів;
6) інтерфейс має бути зрозумілий всім користувачам і адаптуватися під його потреби;
7) розділення системи на модулі, тобто на частини, які виконують однакові задачі;
8) стаціонарне чи віддалене керування, швидке реагування та оповіщення для клієнта про надзвичайні ситуації;
9) багатомовна технічна підтримка онлайн, чи офлайн за допомогою добре структурованих розділів FAQ чи технічного бота.

Тобто, компанії, які розробляють пристрої концепції Інтернету речей, необхідно чітко слідувати тим трендам та потребам, які виставив для них сьогоднішній світ. Важливо слідувати за динамікою змін потреб і у випадку, коли

щось змінилося, компанія потрібно оперативно на це реагувати, щоб зайняти передову нішу у розробці IoT.

1.3 Протоколи та стандарти передачі даних в системах автоматизації

Для того, щоб правильно змоделювати та розгорнути проект «розумних» систем, потрібно знати, коли та як використовувати різні типи підключень. Такі складні системи зачасту комбінують і дротові, і бездротові типи підключень, які можуть в критичні моменти взаємно замінити один одного. Системи з комбінованим підключенням працюють стабільніше та швидше і не залежать від перешкод, які можуть бути на одному типі підключення. Але для передачі даних, необхідно розумітися на протоколах передачі даних. Саме ці протоколи дають змогу убезпечити користувацькі дані, передавати дані без втрат навіть по нестабільній мережі, передавати різні типи даних по одному каналу.

Розглянемо основні типи протоколів, які можуть бути використані при моделювання «розумних» моделей та при створенні реальних систем в помешканнях, на робочих місцях чи в промислових підприємствах.

1.3.1 Протоколи передачі даних UDP, HTTP, MQTT

UDP – один із основних протоколів передачі даних в мережі Інтернет. Головна характеристика – це можливість посилати пакети по мережі іншим хостам по їх IP-адресам. Для цього не обов'язково налаштовувати на початковому етапі особливі канали передачі даних. UDP працює досить просто і в ньому не передбачено використання сигналів стосовно того, що датаграма успішно доставлено. Тому іноді виникають такі проблеми, що пакети приходят не по порядку, бувають дубльовані. Але протокол гарантує, що якщо пакет дістався адресата, то в повністю цілісному стані. Наразі UPD часто використовується в DNS-серверах, VoIP пристроях, де потрібно відповідати на маленькі запити від тисяч клієнтів. Структуру датаграми UDP можна побачити на рисунку 1.1 [3].

Біти	0–15	16–31
0–31	Порт відправника	Порт отримувача
32–63	Довжина датаграми	Контрольна сума
64 – …	Дані	

Рисунок 1.3 – Структура датаграми UPD.

Порт відправника вказує, з якого порта було відправлено датаграму. Очікується, що саме на цей порт прийде відповідь у випадку успіху чи не успіху передачі даних. Порт може бути як статичним, так і динамічним.

Порт отримувача – майже повна аналогія з портом відправника. Також може бути динамічним чи статичним (наприклад для сервера).

Довжина датаграми задає довжину всієї переданої інформації в байтах. Містить в собі довжину заголовку та даних. Максимальний розмір даних – 65535 байт[3]. Але при такому розмірі, можлива фрагментація пакету і він может прийти не повністю, прийти з затримкою чи взагалі різні фрагменти можуть прийти на різні пристрої. Тому посилання великих розмірів даних через UDP не рекомендується.

Контрольна сума використовується для перевірки даних та заголовку на предмет помилок. Може бути пустим і (наприклад в IPv4) не обов'язковим. Розрахунок контрольної суми описується стандартом RFC 1071.

Важливо пам'ятати, що протокол не є стовідсотково надійним. Але в сучасному світі така організація надійності може навпаки нашкодити протоколу передачі даних. Оскільки UDP використовується там, де втрата якихось пакетів не є великою проблемою і її можна проігнорувати (наприклад VoIP, чи багатокористувацькі ігри).

HTTP - Hypertext Transfer Protocol – протокол прикладного рівня. Основна початкова ціль – передача документів в форматі HTML. Але з розвитком технологій протокол зміг передавати будь-який тип інформації. Працює по принципу «клієнт-сервер». Пристрої для роботи HTTP розділяються на три

групи: клієнти, сервера та проксі, які займаються виконанням транспортних зобов'язків (наприклад nginx).

Структура HTTP складається з обов'язкових трьох частин, які повинні слідувати виключно в строгому порядку[4]:

1) Початкові дані, які характеризують тип відправленого повідомлення. Тут повинна вказуватися метод передачі даних, URI та версія використаного протоколу.
2) Заголовки, основна задача яких охарактеризувати тіло відправленого повідомлення, основні параметри передачі даних та додаткові параметри, указані в метаданих.
3) Тіло повідомлення – це те, що відправляється за допомогою протоколу. Зазвичай можна передавати захищені дані (паролі, файли). Тіло не являється обов'язковим для окремих версій HTTP.

При використанні обов'язково повинен повертатися якийсь HTTPResponse. Якщо запит неуспішний, то буде відправлятися помилка зі спеціальним кодом. При успішному, передається код про успішність і дані, на які поступив запит (при необхідності). Response код можна поділити на наступні групи [3]:

1) Коди, які починаються з 1 (100, 103) – це інформаційні коди, наприклад можна передат статус опрацювання запиту;
2) 2xx (200, 201) – це коди про успішність виконання запиту;
3) 3xx (300, 304) – це інформація про перенаправлення на окремий ресурс;
4) 4xx (404, 405) – говорять про помилку клієнта;
5) 5xx (500, 502) – помилки, які трапляються на стороні сервера.

Наведемо декілька прикладів типів запитів через HTTP

1) GET – використовується для передачі деякої інформації. Не є безпечним, тому не варто передавати конфіденційні дані. Такі запити кешуються і зберігаються в закладках браузера;
2) POST – використовується для передачі дани. Повідомлення зберігається в тілі HTTP запиту, тому можна вважати метод передачі безпечним;

3) DELETE – використовується для видалення конкретного ресурсу;
4) PUT – зазвичай використовують для зміни існуючих даних. Якщо дані не знайдено, вони будуть створені і повернеться 201 код про успіх.

На сьогоднішній день вийшла версія HTTP/3. Вона підтримується в усіх сучасних браузерах, але ще не є розповсюдженою масово.

MQTT – це один з основних протоколів, який використовується для передачі даних між пристроями в Інтернеті речей. Основна задача – передати невеликі фрагменти інформації. Активно підтримує QoS - Quality of Service. QoS використовується для того, щоб активно передавати багато інформації одночасно. Особливістю являється те, що дані гарантовано будуть передані, навіть якщо якість сигналу впаде чи з'являються штучні перешкоди. Оскільки MQTT – протокол на основі TCP/IP, то зв'язок відбувається в кілька етапів. Розглянемо їх.

1) Встановлення з'єднання TCP відбувається шляхом передачі декількох пакетів від клієнта до сервера;
2) встановлення з'єднання MQTT. Якщо успішно пройшов етап 1, то надсилається пакет connect. В цьому пакеті міститься інформація про права підключення клієнта до сервера. Якщо права на підключення є, то сервер відправляє клієнту повідомлення про успішність і дає дозвіл про обмін даними;
3) коли дані передано, то TCP з'єднання переривається разом з MQTT. Це відбувається завдяки відправки серверу спеціального пакету, де є змінна FIN = 1.

Тобто до переваг протоколу можна віднести наступне:

- MQTT містить в собі функцію Last Will and Testament. Вона потрібна для нестабільної мережі, щоб інформувати про аварійне відключення всіх або частини пристроїв;
- використання TCP/IP для організації захищених зв'язків в мережі і забезпечує стовідсоткову відправку даних від клієнта до сервера при використанні порту 1883 по замовчуванні;

- оскільки протокол може функціонувати навіть при низькому якості зв'язку, то дані передаються в простому і компактному форматі;

MQTT працює в мережі, де є три обов'язкові учасники – брокери, підписники та видавець. Видавець – це пристрій, який можуть відправляти повідомлення в мережі IoT. Брокер – це вузол, який відповідає за взаємодію між видавцем та підписником. Брокер займається обробкою даних і перевірку на валідність. Підписник – це кінцевий отримувач даних, може виступати в ролі окремого сервера чи хмарних сервісів.

Тобто, MQTT ідеально підходить для проектування та розвернення систем Інтернету речей завдяки організації обов'язкової передачі даних за допомогою функції QoS навіть при слабкому зв'язку між компонентами.

1.3.2 Протокол Z-Wave для бездротової передачі даних

Z-Wave – радіопротокол передачі даних, який використовується зачасту в системах домашньої автоматизації. Використовується Xiaomi, Apple, Amazon. Важливою характеристикою протоколу э те, шо він повністю стандартизований. Тобто, протокол покриває абсолютно всі рівні OSI[2]. Така особливість дає змогу протоколу працювати на різних пристроях від різних виробників. Саме тому він такий поширений. Загалом, Z-Wave розроблений для того, щоб функціонувати в невеликих домах чи квартирах. Підтримує підключення до 150 пристроїв по принципу «все сам». Тобто практично кожен може розгорнути таку систему самостійно без спеціалістів. Крім того, протокол Z-Wave витрачає мало енергії. Часто пристрої, які працюють в цьому протоколі, використовують батарейки для живлення і можуть так працювати роки.

Z-Wave використовує комірчату топологію. Це означає, що можна створити мережу з двох пристроїв: з керуючого та керованого. І таких пристроїв можна додавати все більше і більше, але потрібно слідкувати, щоб сигнал не став надто слабкий. Комірчата топологія – це топологія, яка може самоорганізуватися. Її маршрутизація залежить від багатьох зовнішніх факторів,

наприклад якщо між двома пристроями з'явилась перешкода і сигнал не може бути переданий, він під в обхід з використанням інших вузлів.

Протокол Z-Wave має наступні переваги:

1) використання шифрування AES-128 для передачі даних;
2) не потребує прокладки дротів;
3) масштабованість;
4) моніторинг через Інтернет

Але є і недоліки. Розглянемо їх:

1) низька швидкість передачі даних;
2) через низьку швидкість, часто можуть бути потрібні підсилювачі сигналу, якщо в систему більше 30 пристроїв;
3) немає можливості передачі звуку та зображень.

Тобто, протокол Z-Wave має широке використання і на 2020 рік його підтримують близько 1000 виробників по всьому світу[2]. Він є основним для невеликих домашніх систем автоматизації.

1.3.3. Способи передачі інформації з використанням мережі Wi-Fi

Wi-Fi – технологія бездротової передачі даних. Будь-яка система Wi-Fi мусить містити не менше однієї точки доступа і не менше одного клієнта. Можливе підключення в режимі Ad-hoc, вони говорить про те, що іноді клієнти можуть не використовувати точку доступа, а підключатися за допомогою мережевих адаптерів[3]. Точки доступа можуть бути публічними або приватними. Сьогодні Wi-Fi широко використовується в Інтернеті речей. Яскравим прикладом є не тільки домашні системи автоматизації, а і так названі «розумні» міста, які повністю покриті мережею Wi-Fi. Прикладом є Барселона та Стокгольм де використовують стандарт IEEE 802.11n через його швидкість та надійність.

До переваг Wi-Fi можна віднести наступне:

- можливо створити мережу без дротів, що дає змогу кращого масштабування;

- висока швидкість передачі даних;
- адаптивність в побудові та реконфігурація;
- сумісність з усіма пристроями на ринку.

Проте наявну певні недоліка, а саме:

- часто адаптери дорогі;
- швидкість залежить від середовища та кількості користувачів;
- можуть використовуватися не повністю безпечні протоколи шифрування переданих даних;
- в багатьох країнах Wi-Fi працює на різних частотах;
- висока енергопотребність;
- часто сигнали можуть «глушити» один одного, тому що в діапазоні , наприклад 2.4 ГГц працює багато пристроїв.

За роки існування Wi-Fi було розроблено досить багато стандартів. Але найбільш розповсюджені наступні:

1) IEEE 802.11a – зазвичай працює в діапазоні 5 ГГц. Максимальна швидкість при прямій передачі – 50 Мбіт/с
2) IEEE 802.11b/g – працює в діапазоні 2.4 ГГц. Максимальна швидкість також 54 Мбіт/с
3) IEEE 802.11 Super f – покращений протокол, який працює в діапазоні 2.4 ГГц, але видає швидкість в 108 Мбіт/с
4) IEEE 802.11n – працює в діапазоні 2.4-2.5 ГГц і 5 ГГц і забезпечує швидкість до 450 Мбіт/с

Зараз активно почали розвиватися так названі Wi-Fi 6, тобто розповсюдження сигналу на частоті 6 ГГц. Але ця технологія ще в процесі розвитку, а маршрутизатори, які здатні використовувати частоту 6 ГГц наразі коштують дуже дорого. Але без сумніву, що з розвитком Wi-Fi 6 він буде проникати все більше і більше в наше життя

Тобто можна сміливо сказати, що технологія Wi-Fi стада найбільш використовуваний звичайною людиною технологією. Але створення нових стандартів, можливість передавати сигнал на кілометри, безпечне шифрування

даних та впровадження підсилювачів сигналу говорить про те, що технологія Wi-Fi в найближчому майбутньому бути найбільш розповсюдженою в світі Інтернету речей.

1.3.4 Мобільні мережі 4G/5G

Сучасні стандарти 4G/5G являються нащадками стандарту 1G. G – означає покоління. Всі мережі стандарту 1G була аналоговими і дозволяли передавати лише звуковий сигнал[4]. Частота, на якій працювали ці мережі була в діапазоні 120-900 МГц. В 1992 році з'явився стандарт 2G. Почала вводитися цифрова передача даних і стало можливо передавати текст. Для цього використовуються протоколи GPRS та EDGE. Через 8 років з'явився стандарт 3G. Через введення деяких оновлень (HSDPA) для стандарту вдалося десятикратно збільшити швидкість передачі даних (до 7.2 Мбіт/с). Це оновлення забезпечило передачу даних з використанням широкосмугових швидкостей[4].

Розробки 4G велися ще до того, як був прийнятий 3G. Але вперше стандарт був представлений в 2009 році. Максимальна заявлена швидкість – до 1000 Мбіт/с. Однією з переваг 4G – це так названа Hand-off система. Основна її суть полягає в тому, шо на одному пристрої можна почати роботу, а потім перемкнутися на інший пристрій і почати роботу звідти, звідки ви перемкнулися. Це забезпечується за допомогою підключення до двох вишок одночасно. Діапазон, в якому працює 4G складає 2000-8000 МГц. Для стандарту четвертого покоління було розроблено ще один стандарт, який називається Long Term Evolution. Завдяки цьому, забезпечується висока швидкість передачі даних і пристрої з LTE можуть працювати з додатками, які потребують великої пропускної здатності (наприклад ігри високої якості, HD – відео).

5G розробки почались в 2008 році (тобто ще до введення 4G). Оскільки стандарт ще не всесвітньо поширений, то пропонується частота від 3.4 до 3.8 ГГц. Але також говорять, що стандарт буде працювати і на наднизьких, і на надвисоких частотах (до 70 ГГц). І це не випадково, адже спектр нижче 6 ГГц зазвичай зайнятий військовими, екстреними службами і так далі. В основі

протоколів буде лежати концепція Network slicing[4]. Вона говорить те, шо розподіл ресурсів буде проводитися в залежності від потреб сегментів мережі в користувачів, які є в цьому сегменті. Серед основних переваг стандарту 5G над 4G можна виділити наступне:

- потреба в меншій кількості електроенергії. Якщо порівнювати з LTE, то 5G буде споживати від 17% до 23% менше електроенергії;
- передача сигналу оптимізована, тому що вишки 5G працюють точечно, вони відправляють сигнал не на площу, а до конкретного пристрою, який потребує цього;
- кількість пристроїв, які можуть одночасно підключитися може збільшитися від 7 до 10 разів. А це зроблено для того, щоб стандарт 5G міг максимально ефективно працювати з Інтернетом речей. Тобто, за допомогою 5G одночасно можна підключити до мільйона пристроїв без втрати якості сигналу.

Тобто, стандарт 5G називають стандартом майбутнього. Адже основна ціль – це не просто підвищити швидкість, а заложити підґрунтя для майбутнього ефективного функціонування Інтернету речей, який безумовно буде присутній у кожній сфері людського життя.

1.4 Існуючі системи та комплекти для побудови «розумних» будинків

Оскільки технологія активно розвивається, не відстаючи від технологічного світу, то багато компаній почали розробляти свої повноцінні комплекти, для того щоб їх можна було швидко та безпечно розгорнути в рамках дому, офісу чи окремого робочого місця. Розглянемо, які компанії являються передовими в розробці та впровадженні технології Інтернету речей в сфери людського життя.

1.4.1 «Розумний» дім від Xiaomi

Розумний дім Xiaomi – це мережа, в яку об’єднані всі пристрої і керується штучним інтелектом, налаштування може відбуватися через спеціальний

офіційний додаток[8]. Структура складається з декількох модулів, які пов'язані між собою за допомогою наступних протоколів передачі даних:

1) ZigBee – в такому випадку, пристрої працюють від акумуляторів і інтегруються за допомогою шлюза керування;
2) Wi-Fi – пристрої працюють в одній мережі і передача даних залежить від якості сигналу і навантаженості на W-iFi модуль;
3) через Bluetooth – робота багато в чому схожа на роботу по Wi-Fi

В систему Xiaomi можуть входити наступні пристрої:

1) датчики;
2) розумні розетки та вимикачі;
3) різні розумні пристрої. починаючи від чайника і закінчуючи відеокамерою.

Приклад пристроїв можна побачити на рисунку 1.4[25]

Рисунок 1.4 – «Розумні» пристрої від компанії Xiaomi.

Серед переваг «розумного» дому Xiaomi можна виділити те, що все керування можна робити через спеціальний додаток MiHome. Невеликі розміри, швидке встановлення, техпідтримка 24/7, створення самостійних сценаріїв роботи, зручне API – це все користувач отримує при покупці такої системи. Але є декілька недоліків. По-перше, система китайська і керування відбувається через китайські сервера чи хмари Alibaba. Через це можуть виникати затримки в прийнятті рішень. Крім того, українська мова не підтримується системою, тому все меню, так само як і інструкції по користуванні написані англійською мовою. Але незважаючи на це, система варта уваги, а через поступове здешевлення, інтегрувати її можна в дім.

1.4.2 «Розумний» дім від Ajax Systems

Вітчизняна компанія, яка займається дротові та бездротові системи і має власне виробництво повного циклу. На 2022 рік система складається з 36 пристроїв для різних сфер життя[6]. Особливість цих пристроїв в тому, що датчики та пристрої можуть працювати автономно від 2 до 7 років просто на своїх батареях. Для зв'язку пристроїв в системі, використовується двосторонній радіопротокол Jeweller, який розроблений власними силами компанії. Пристрої можна побачити на рисунку 1.5[26]

Рисунок 1.5 – Пристрої від Ajax Systems.

Центр керування – хаб. До нього можна підключити до 200 різних пристроїв. Пристрої можуть подавати сигнал до 2 км. Хаб може використовувати декілька каналів[6]: Ethernet, Wi-Fi, SIM. Хаб безперервно в контакті з хмарою компанії і дуже швидко може відправляти та приймати дані.

Домашньою системою можна керувати за допомогою додатку. Він дозволяє створювати групи пристроїв, виключати чи підключати нові, керувати правами користувачів, додавати нових, переглядати історію подій та логи помилок. Важливо пам’ятати, що системам Ajax присуджено найвищий ступінь надійності. Використовує наступні технології:

- Fibra – дротова технологія передачі даних з високим рівнем захищеності і з мінімуму обмежень на типи передачі даних;
- ELSA – трьохетапних алгоритм, який використовується в інфочервоних сенсорах;

- Wings – радіопротокол, який здатний передавати дані в реальному часі навіть при нестабільному підключенні чи в навмисно створених радіоперешкодах.

Тобто, Ajax Systems є серйозним гравцем серед компаній, які займаються розробкою «розумних» пристрої. А з розвитком сенсорних, ML – технологій, продукція компанії буде також стрімко зростати.

1.5 Програмні засоби створення моделей з використанням Інтернету речей

Для створення моделей з використанням Інтернету речей на сьогодні є багато додатків. Загалом, такі додатки розділяють на емулятори та симулятори. Розглянемо різницю між цими поняттями.

Емуляція – це виконання задач програми чи системи зі збереженням її характеристик та принципів роботи[9]. Емуляція виконує програмний код в середовищі, яка складається з тих же компонентів, що і об'єкт емуляції. Тобто створюється майже точна модель пристрою чи системи.

Симуляція – це відтворення роботи програми оригінала виключно віртуально. Для цього використовується спеціальна програма. Симуляція лише імітує виконання кода[10]. Зачасту симулюються лише окремі характеристики, можливості чи функції програми, при чому часто не в повному об’ємі. Створюється ілюзія, наче ми працюємо в справжній програмі, хоч насправді, функціонал «фальшивий».

Серед найпопулярніших додатків для створення моделей з використанням домашньої автоматизації – це Cisco Packet Tracer, GNS3, NetSim. Кожен має свої плюси та мінуси. Розглянемо кожен з них.

1.5.1 Додаток GNS3

GNS – Graphical Network Simulator – кросплатформний графічний симулятор мережі, який дозволяє створювати віртуальні мережі. Для

повноцінної роботи середовище GNS3 використовує наступні програмні продукти[11]:

- WinPCAP – драйвер та бібліотека функцій, що дозволяє отримати доступ до інтерфейсів фізичного комп'ютера і інформації, що передається.
- Wireshark – графічний аналізатор мережевого трафіку. Візуально відображає інформацію про трафік. Використовується в середовищі GNS3.
- SolarWinds Response – аналізує трафік, підготовлений за допомогою Wireshark
- PUTTY – система терміналу, дозволяє віддалено підключатися до пристроїв та керувати ними.

Розглянемо принцип конфігурації маршрутизатора. Основні параметри можна побачити на рис. 1.6

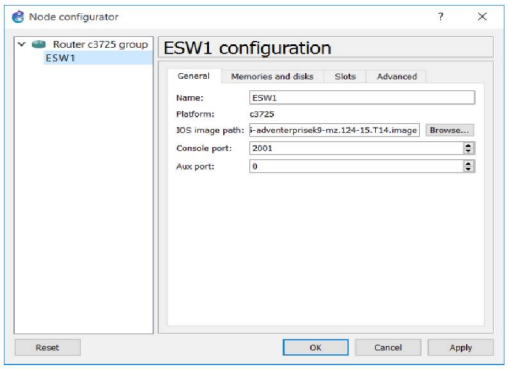


Рисунок 1.6 – Налаштування маршрутизатора в GNS3.

Серед переваг даного додатку можна виділити:

1) можливість створення складних топологій;
2) симуляція Cisco маршрутизаторів;
3) використання зручних программ для контролю трафіку;
4) безкоштовна;
5) моделювання ATM, Ethernet підключень.

Але є і недоліки. Основні – це:

1) Відсутність самостійного створення пристроїв IoT;
2) слабка підтримка концепції Інтернету речей;
3) черезмірне використання CPU комп'ютера;

4) слабке налаштування L2 мереж[12].

Загалом додаток добре себе показує при побудові складних кросплатформних мереж з використанням пристроїв різних компаній. Через свою простоту та доступність часто використовуються для навчання та отримання сертифікації CCNA. Але для побудови мереж з використанням Інтернету речей не підходить, через слабку інтеграцію з цією концепцією.

1.5.2 Додаток NetSim

NetSim — це інструмент моделювання мережі, який дозволяє створювати мережеві сценарії, моделювати трафік, проектувати протоколи та аналіз продуктивності мережі. Користувачі можуть досліджувати поведінку мережі за допомогою тесту комбінації мережевих параметрів[12]. Приклад логічної топології мережі NetSim можна побачити на рисунку 1.7

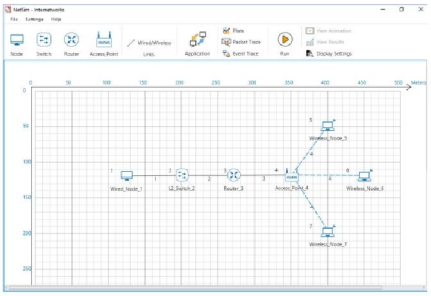


Рисунок 1.7 – Логічна топологія мережі NetSim.

Додаток має свої плюси та мінуси, які слід розглянути. Серед переваг слід виділити:

1) інтуїтивно зрозумілий інтерфейс;
2) широкий вибір пристроїв для побудови мережі (наприклад Cisco);
3) можливість створення складних топологій з використанням пристроїв від різних виробників;
4) зручний моніторинг трафіку, який зображено в графічному виді

Але є деякі мінуси. Розглянемо їх:

1) вразливість мереж через проблеми в безпеці;

2) проблеми підтримки C/C++ скриптів;
3) низька підтримка інтернету речей та відсутність створення власних вузьконаправлених пристроїв;
4) присутність незрозумілих багів, які можна скинути тільки перезавантаженням додатку.

В додатку присутній зручний спосіб тестування передачі пакетів по мережі. Це дозволяє вивчати сценарії, коли пакет не пройшов, дізнатися, в чому була причина затримки та налаштування автоматичного посилання пакетів у випадку їх втрати чи не отримання адресатом[13]. Через те, що в додатку є багато пристроїв різних виробників, то вони працюють абсолютно одинаково, тому буває важко знайти різницю в пристроях на етапі емулювання.

1.5.3 Додаток Cisco Packet Tracer

Безумовно найпопулярнішим додатком для створення комп'ютерних мереж є Cisco Packet Tracer. Продукт являється кросплатформним і можна завантажити та встановити на Windows, Linux, Unix, Mint[14]. Для створення такої складної моделі, як модель науково-дослідницької лабораторії слід використати саме цей додаток. Цьому є наступні причини:

1) додаток має можливість розділення мережі на логічну та фізичну топології, що зручно для моніторингу та контролю пристроїв в рамках кожної мережі;
2) повна інтеграція з концепцією Інтернет речей. В наявності є розумний дім, індустріальний IoT. Також присутні програмні компоненти, які можна змінювати під себе. Крім того, є можливість створювати свої пристрої практично будь-якої складності та успішно інтегрувати їх в нову чи уже створену мережу;
3) широкі можливості задання параметрів впливу на мережу за допомогою стандартних функцій. Якщо цього мало, є можливість програмно задати параметри впливу на всю мережу чи на окремі компоненти;

4) широка підтримка різноманітних пристроїв, таких як комутатори, транслятори, хаби, різні типи серверів, VoIP-пристрої
5) можливість створення своїх кластерів;
6) підтримка мов програмування Python, JavaScript останніх версій;
7) зручне налаштування пристроїв на локальне чи віддалене управління та моніторинг

Серед недоліків слід виділити:

1) слабка підтримка IOS;
2) проблемні протоколи STP та MQTT
3) складні мережі можуть навантажувати сильно навантажувати додаток, що спричиняє баги;
4) складність розподілення коду по окремим файлам;
5) відсутня можливість спробувати unit-тести для окремих модулів коду.

Крім того, додаток дозволяє графічно аналізувати трафік та в реальному часі показувати, як пакети даних передаються мережею. Розглянемо деякі компоненти додатку. На рисунку 1.8 зображено панель симуляції передачі пакетів по мережі.

Рисунок 1.8 – Симуляція передачі пакетів по мережі.

Можна фільтрувати по протоколам передачі даних чи вибрати конкретний, який ми хочемо використати для передачі пакетів від пристрою А до пристрою Б. Наглядно демонструється статус пакета (успішно чи неуспішно).

На рисунку 1.9 зображено всі фактори зовнішнього середовища, на основі яких можна створювати умови функціонування умови функціонування окремих пристроїв чи всієї моделі.

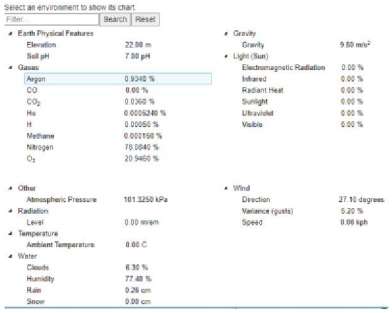


Рисунок 1.9 – Фактори зовнішнього середовища для керування моделлю.

Якщо таких факторів недостатньо, то ми легко можемо створити кастомні за допомогою мови програмування Python, Basic чи JavaScript. Важливо пам’ятати, що код може імітувати не тільки зовнішнє середовище. а і повноцінні «розумні» пристрої. При запуску коду, він починає працювати на фоні і зупинити його (при відсутності помилок в коді) можна вручну

Якщо пристрої мають однакову логіку, наприклад забезпечують віддалене керування, то доцільно розділити їх на окремі кластери зі своїми задачами.
Для того, щоб візуально побачити, в які порти, як пов’язані пристрої один з одним, слід використати фізичне відображення створеною моделі.

Важливою особливістю додатку є те, що кожну мережу можна розділити на локацію. До цього слід віднестися серйозно, тому що кожна модель, створена в додатку, повинна знаходитися в тій локації, в якій запланований оригінал. На рисунку 1.10 зображено розділ на фізичні локації.

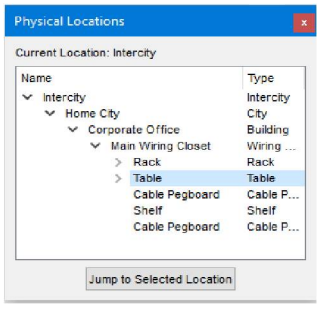


Рисунок 1.10 – Фізичні локації додатку.

Загалом, всі пристрої додатку можна розділити на наступні категорії

1) кінцеві пристрої. Сюди входять маршрутизатори, сервери, хмарні пристрої, VoIP пристрої та комутатори (рисунок 1.11).

Рисунок 1.11 – Network devices для побудови мереж.

2) кінцеві пристрої, за допомогою яких можна впливати на функціонування моделі. Саме сюди входять пристрої інтернету речей (рисунок 1.12).

Рисунок 1.12– End devices для керування мережею.

Приклад таких пристроїв зображено на рисунку 1.13

Рисунок 1.13 – Приклади кінцевих пристроїв додатку.

3) окремі компоненти, які можна програмувати для імітації впливу зовнішнього середовища чи справжній «розумних» пристроїв. Саме вони підтримують використання мов програмування Python, JavaScript (рисунок 1.14).

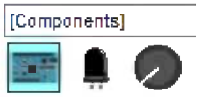


Рисунок 1.14 – Програмовані компоненти додатку.

4) для поєднання пристроїв моделі, які потребують дротове підключення, слід використати різні типи дротів, в залежності від специфікації пристрою.

Підключення поділяються на звичайні підключення та структурні, які дозволяють на фізичному рівні комбінувати різні типи підключень для пристроїв, які підтримують це (наприклад сервер).

5) Коли над моделлю працює декілька людей, слід використати багатокористувацькі підключення. На рисунку 1.15 зображено процес конфігурації такого підключення.

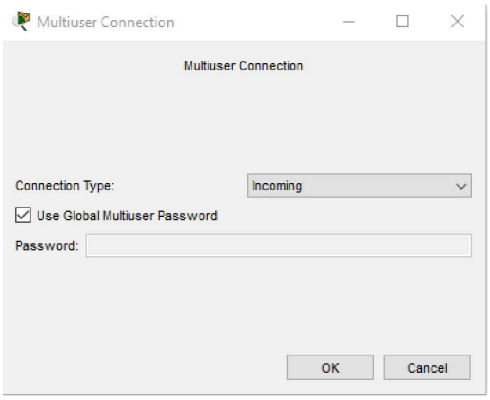


Рисунок 1.15 – Налаштування багатокористувацького підключення для керування мережею.

Тому можна сказати, шо завдяки своєму широко направленому функціоналу, додаток Cisco Packet Tracer найкраще підходить для проектування «розумної» науково-дослідницької лабораторії. Крім того, додаток постійно оновлюється, додаються нові можливості та пристрої, які не зламають існуючу модель при їх інтегруванні[15].

1.6 Постановка задачі по моделюванню

На основі концепції Інтернет речей, в якій компоненти легко масштабуються, легко інтегруються в нові чи уже існуючі системи, можна зробити висновок, що Інтернет речей ідеально підходить для того, щоб створити модель «розумної» науково-дослідницької лабораторії. Така модель повинна мати наступні переваги: зручний віддалений чи локальний контроль підсистемами, моніторинг протікаючих процесів, інформування про надзвичайні ситуації, забезпечення безпеки персоналу лабораторій та устаткуванню для того,

щоб зберегти напрацювання, надання права приймати рішення в залежності від впливу зовнішнього середовища чи від заданих параметрів керування. Для того, щоб створити таку модель необхідно:

1) розглянути та вивчити, як технологія Інтернету речей працює в сучасних мережах;
2) проаналізувати протоколи передачі даних для швидкої і безпечної передачі даних між пристроями мережі;
3) розглянути принципи використання «розумного» дому, промислового Інтернету речей та способи їх комбінування;
4) на основі вибраного додатку для емулювання мереж створити власну модель «розумної» науково-дослідницької лабораторії;
5) «навчити» модель самостійно приймати рішення в залежності від заданих сценаріїв роботи чи впливу навколишнього середовища на протікання процесів;
6) протестувати створені пристрої за допомогою бібліотек тестування та на пристроях в моделі;
7) при необхідності створити власні «розумні» пристрої чи пристрої, які будуть імітувати зовнішні фактори;
8) розділити модель на фізичні рівні та задати логіки їх взаємодії;
9) за допомогою мережевих пристроїв налаштувати віддалене та локальне керування пристроями моделі;
10) протестувати модель загалом, задаючи чи змінюючи як параметри зовнішнього середовища, так і умови, за яких приймаються рішення;

Тобто, виконавши всі завдання, повинна бути створена легко масштабована модель науково-дослідницької лабораторії, яка здатна самостійно приймати рішення, забезпечуючи процес дослідження та вивчення більш безпечним, продуктивним та швидким. Сценарії роботи можна змінювати та слідкувати за ними локально, безпосередньо в рамках моделі, чи віддалено за допомогою мережі Інтернет.

1.7 Висновки до розділу І

В даному розділі була описана концепція Інтернету речей, основні вимоги, які є в сучасному світі для домашньої автоматизації «розумний дім». Описано причини росту популярності технології та майбутні перспективи. Розглянуто основні протоколи та стандарти передачі даних в мережах, які побудовані на основі Інтернету речей. Наведено приклади уже існуючих комплектів для побудови «розумних» систем.

Проаналізовано основні сучасні додатки для побудови складних моделей з використанням концепції Інтернету речей. Розглянуто їх основні можливості та особливості використання, способи створення та інтеграції нових пристроїв з використанням мов програмування Python та JavaScript. На основі проведеного аналізу вирішено використати додаток від компанії Cisco, який називається Cisco Packet Tracer.

РОЗДІЛ ІІ
СТВОРЕННЯ МОДЕЛІ

2.1 Загальна модель

Для того, щоб про фізично розділити пристрої по різним лабораторіям, потрібно створити фізичну топологію моделі. Вона показана на рисунку 2.1

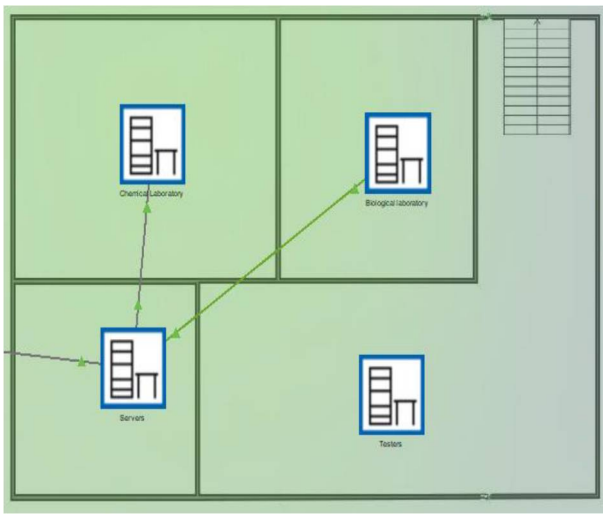


Рисунок 2.1 – Фізична топологія моделі.

Загалом, науково-дослідницька лабораторія складається з 4 компонентів:

1) Біологічна лабораторія;
2) хімічна лабораторія;
3) серверна частина;
4) прилади тестування;

Для демонстрації повного виду лабораторії, зі всією логікою варто перейти в розділ «Logical». Всі пристрої знаходяться в своїх логічний рівнях і пов’язані один з одним тільки через єдиний сервер керування, де можна задати логіку їх поведінки та прийняття рішень в тих чи інших ситуаціях. Логічна топологія моделі науково-дослідницької лабораторії показана на рисунку 2.2.

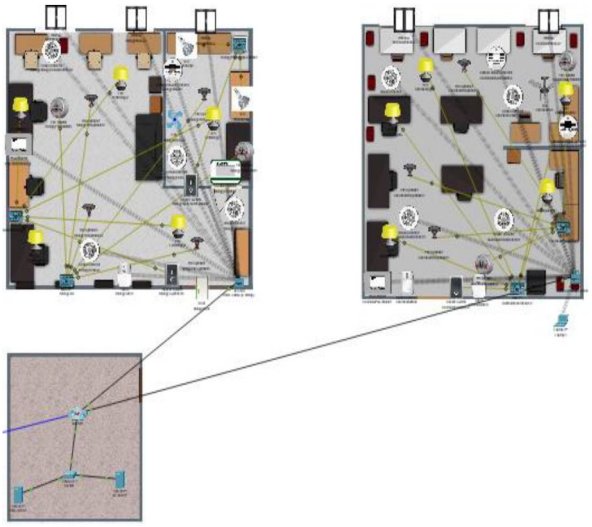


Рисунок 2.2. Логічна модель «розумної» науково-дослідницької лабораторії.

Тобто, модель «розумної» науково-дослідницької лабораторії складається з 3 частин, які мають ізольовану логіку взаємодії з користувачем та принципами прийняття рішень і пов’язані з серверною частиною для віддаленого доступу за допомогою мережі Інтернет

2.2 Проектування серверної частини моделі

Для правильного функціонування моделі і можливості відстежувати сигнали від мережі, коригувати, налаштовувати чи видаляти правила взаємодії пристроїв з користувачами чи один з одним в залежності від параметрів зовнішнього середовища слід налаштувати віддалений доступ до компонентів мережі. Фізичну топологію серверної частини можна побачити на рисунку 2.3

Рисунок 2.3 Фізична топологія серверної частини моделі.

Доступ повинен бути з будь-якої точки світу з використанням 3G/4G/5G чи Wi-Fi. Для виконання цих завдань варто налаштувати сервери та маршрутизатори. Конфігурацію можна побачити на рисунку 2.4

2.2.1 Налаштування DHCP та DNS

DHCP сервер потрібен для того, щоб динамічно видавати IP-адреси всім пристроям мережі. Зазвичай він працює в парі з DNS сервером, який змінює IP в звичні для нас URI.

Для створення DHCP сервера потрібно вибрати тип сервера WMP300N. Він підтримує підключення через FastEthernet та GigabitEthernet. Спочатку потрібно налаштувати IP адресацію для DNS-сервера. Процес налаштування можна побачити на рисунку 2.4

```
Router>en
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int g0/1
Router(config-if)#ip address 10.0.0.1 255.255.255.0
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up

Router(config-if)#
```

Рисунок 2.4 – Налаштування адресації DNS-серверу.

Далі потрібно налаштувати DNS-сервер для зміни IP на зрозумілий URI. Схему налаштування можна побачити на рисунку 2.5

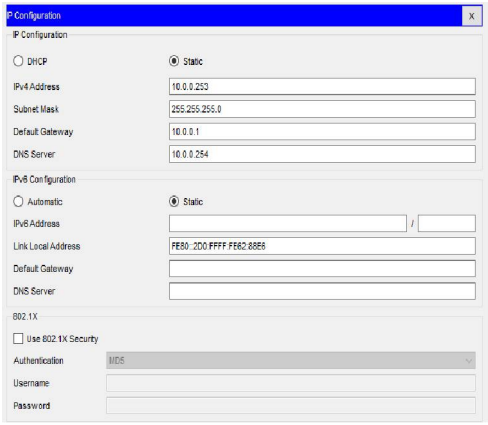


Рисунок 2.5 – Налаштування IP-конфігурації для DNS-сервера.

Далі нам потрібно прив’язати IP-адрес і URI для того, щоб можна було перейти за посиланням, зайти в аккаунт і мати можливість керувати пристроями Інтернету речей. Налаштування можна побачити на рисунку 2.6

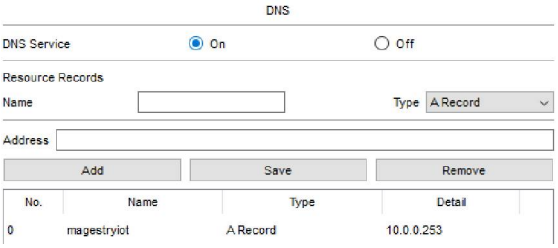


Рисунок 2.6 – Прив’язка IP адреси до URI.

2.2.2 Організація керування пристроями моделі

Для налаштування безпечного віддаленого доступу до керування мережею локально чи віддалено потрібно налаштувати IP-адресацію для віддаленого серверу, де буде зберігатися вся інформація стосовно пристроїв Інтернету речей та логіка їх взаємодії та функціонування. Потрібно вибрати сервер по типу Central Office і провести налаштування IP, як показано на рисунку 2.7

```
Router(config-if)#int g0/2
Router(config-if)#ip address 209.165.201.255 255.255.255.224
Bad mask /27 for address 209.165.201.255
Router(config-if)#ip address 209.165.201.225 255.255.255.224
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/2, changed state to up

Router(config-if)#
```

Рисунок 2.7 – IP-адресація серверу керування.

Також потрібно налаштувати хмару для обміну даними віддалено. Потрібно вибрати хмару типу Cloud-PT. Налаштування показано на рисунку 2.8

```
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int g0/0
Router(config-if)#ip address 209.165.200.225 255.255.255.224
Router(config-if)#not shutdown
                  ^
% Invalid input detected at '^' marker.

Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up
```

2.8 – Налаштування хмари для віддаленого доступу.

Далі потрібно налаштувати DHCP-пули, в рамках яких DHCP – сервер зможе роздавати IP-адреси для конкретних пристроїв. Загалом, пули можна розділити на три види: для кластера (де знаходяться сервера та хмара, для біологічної лабораторії та для хімічної).

Розглянемо схему налаштування пула для кластера. Потрібно вибрати потрібний маршрутизатор (вибрано Router-PT-Empty, тому що в нього найкраща масштабованість в можливістю додавати нові порти різних типів. Процес додавання нових портів для пристроїв в додатку Cisco Packet Tracer показано на рисунку 2.9.

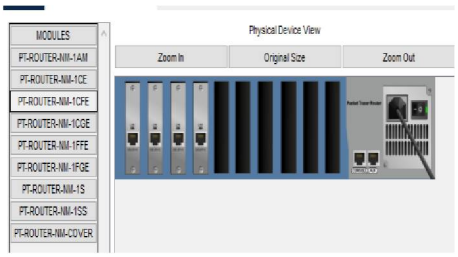


Рисунок 2.9 – Процес додавання нових портів до пристроїв Cisco.

В консолі маршрутизатора потрібно вказати стандартні адреси для серверів та адрес DNS-сервера. І виключити з пулу адреси, які будуть використовуватися для інших пулів. Процес налаштування показано на рисунку 2.10

```
Router$conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)$ip dhcp excluded-address 209.165.201.225 209.165.201.229
Router(config)$ip dhcp pool CELL
Router(dhcp-config)$network 209.165.201.224 255.255.255.224
                            ^
% Invalid input detected at '^' marker.

Router(dhcp-config)$network 209.165.201.224 255.255.255.224
Router(dhcp-config)#default-router 209.165.201.225
Router(dhcp-config)#dns-server 10.0.0.254
```

Рисунок 2.10 – Налаштування DHCP-pool для кластера.

Тепер за аналогією потрібно налаштувати пули для біологічної та хімічної лабораторії. Налаштування DHCP-pool під назвою Biology для автоматичної роздачі IP-адрес пристроям біологічної лабораторії, показано на рисунку 2.11

```
Router(config)#ip dhcp excluded-address 209.165.202.225 209.165.202.229
Router(config)#ip dhcp pool Biology
Router(dhcp-config)#default-router 209.165.202.225
Router(dhcp-config)#network 209.165.202.224 255.255.255.224
Router(dhcp-config)#dns-server 10.0.0.254
Router(dhcp-config)#exit
Router(config)#
```

2.11 Налаштування Biology pool для біологічної лабораторії.

Конфігурацію DHCP-pool під назвою Chemical для хімічної лабораторії показано на рисунку 2.12.

```
Router(config)#ip dhcp excluded-address 209.165.200.225 209.165.200.229
Router(config)#ip dhcp pool
% Incomplete command.
Router(config)#ip dhcp pool Chemical
Router(dhcp-config)#default-router 209.165.200.225
Router(dhcp-config)#network 209.165.200.224 255.255.255.224
Router(dhcp-config)#dns-server 10.0.0.254
Router(dhcp-config)#exit
Router(config)#
```

2.12 Налаштування Chemical pool для хімічної лабораторії.

Всі сервера та маршрутизатори налаштовані, логічну топологію системи для віддаленої та локального керування пристроями моделі можна побачити на рисунку 2.13.

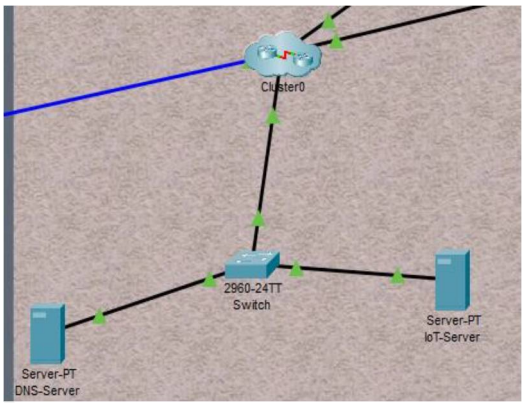


Рисунок 2.13 Логічна топологія системи налаштування віддаленого та локального доступу до пристроїв моделі.

Перевіримо правильність виконання системою своїх задач. Виберемо будь-який пристрій моделі для перевірки правильності отримання IP-адреси. Результат можна побачити на рисунку 2.14

Рисунок 2.14 – Отримання пристроєм IP-адреси від DHCP-сервера.

Спочатку перевіримо правильність роботи DNS-сервера. За адресом www.magestryiot.com і при заданні правильного логіну та паролю повинен з'явитися інтерфейс керування пристроями. Результат на рисунку 2.15

Рисунок 2.15 – Робота DNS-серверу.

Перевіримо доступність зареєстрованих в мережі пристроїв і доступність до них (рисунок 2.16).

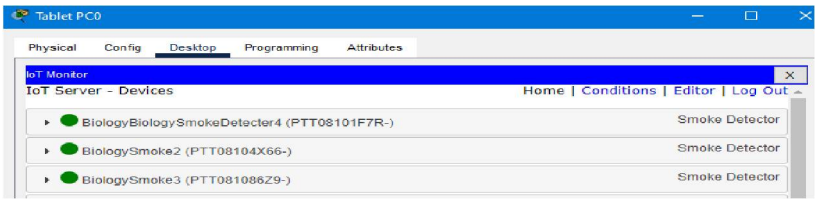


Рисунок 2.16 – сервіс керування IoT пристроями моделі.

Тобто, всі пристрої в моделі мають свій IP і можуть обмінюватися даними в мережі і приймати зміни як локально, так і віддалено. Налаштування IP-адресації за допомогою DHCP-серверу зробило задачу адресації легше, адже тепер IP пристрої отримують автоматично, без потреби статичного задання.

2.3 Проектування біологічної лабораторії

Біологічна лабораторія має свої особливості, на які варто звертати увагу при проектуванні. Насамперед, це специфіка пристроїв та параметрів

зовнішнього середовища, що впливає як на пристрої, так і на саму систему в цілому. Модель біологічної лабораторії можна побачити на рисунку 2.17

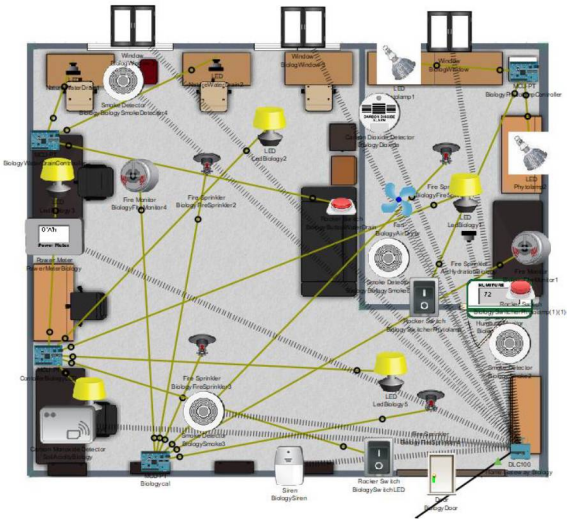


Рисунок 2.17 – Модель біологічної лабораторії.

Фізичне представлення моделі показано на рисунку 2.18

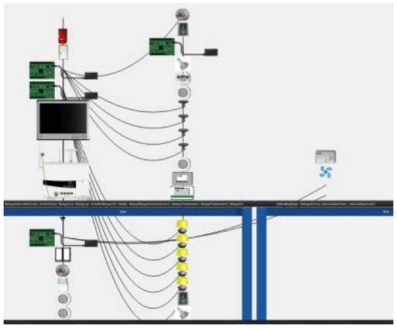


Рисунок 2.18 – Фізичне відображення моделі.

Розглянемо ближче підсистеми, які присутні в моделі

2.3.1 Система осушення повітря

Бля біологічної лабораторії система осушення повітря дуже важлива. Коли ми говоримо про різні типи біологічних дослідів, наприклад над рослинами, то потрібно звертати увагу на стан повітря на місці проведення дослідів чи спостережень. Схема підсистеми осушення повітря показана на рисунку 2.19.

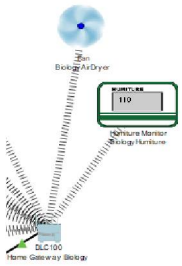


Рисунок 2.19 – Система осушення повітря біологічної лабораторії.

Вона складається з двох частин: від вимірювача вологості повітря та спеціального пристрою, який в залежності від рівня вологості розганяю та підігріває повітря на робочому місці. Логіка роботи системи показана на рисунку 2.20

Edit Remove	Yes	BiologyDryerOnLow	Match all: • BiologyHumiture Humitor >= 75 • BiologyHumiture Humitor <= 89	Set BiologyAirDryer Status to Low
Edit Remove	Yes	BiologyDryerOnHigh	BiologyHumiture Humitor >= 89	Set BiologyAirDryer Status to High
Edit Remove	Yes	BiologyDryerOff	BiologyHumiture Humitor <= 74	Set BiologyAirDryer Status to Off

Рисунок 2.20 – Умови роботи системи осушення повітря.

Тобто можна сказати, що коли рівень вологості зависокий, то осушувач вмикається на повну потужність. Коли вологість падає до нормального рівня, пристрій вимикається. Для цього, потрібно створити сам пристрій осушення повітря. Зконфігурований пристрій можна побачити на рисунку 2.21

```
import {valumeRate,globalState,dryFast,dryMiddle} from Environment
const setup = () => {

    IoEClient.setup({
        type: "Air dryer",
        states: [
        {
            name: "Status",
            type: "options",
            options: {
                "0": "Off",
                "1": "Low",
                "2": "High"
            },
            controllable: true
        }
        ]
    });

    IoEClient.onInputReceive =incomeData=> processData(incomeData, true);


    attachInterrupt(0, () => processData(customRead(0), false);


    dryerState = restoreProperty("state", 0);
    setState(dryerState);
}
const restoreProperty = (incomeProps, initialData)=>{
```

Рисунок 2.21 – Конфігурація пристрої осушення повітря.

Загалом, логіку роботи пристрою можна описати наступним чином: на вхід контролера подається сигнал про рівень вологості повітря. Сервер порівнює

вхідні дані з параметрами, які задані для нього стандартними адміністратором. В залежності від рівня, який сервер отримав від пристрою моніторингу рівня вологості – подається сигнал на вихід, який приймає пристрій осушення, котрий в свою чергу починає працювати в тому режимі, який переданий йому у відповіді від сервера.

2.3.2 Система освітлення за допомогою фітоламп

Система освітлення в виді фітоламп потрібна для того, щоб прискорити ріст та розвиток рослин в штучних умовах. Така система повинна бути в кожній біологічній лабораторії, але зазвичай вона стаціонарна і не залежить від Інтернету речей. «Розумну» систему освітлення за допомогою фітоламп показано на рисунку 2.22.

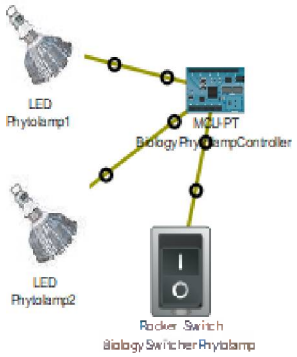


Рисунок 2.22 – Система освітлення фітолампами.

Головна частина цієї системи – це контролер, який делегує задачі між сервером, лампами та вимикачем. Конфігурація показана на рисунку 2.22.

```
from gpio import *
from time import *

pinMode(0,INPUT)
pinMode(1,OUTPUT)
pinMode(2,OUTPUT)

while True:
    if digitalRead(0)==HIGH:
        digitalWrite(1,HIGH)
        digitalWrite(2,HIGH)
    else:
        digitalWrite(1,LOW)
        digitalWrite(2,LOW)
```

Рисунок 2.22 – Конфігурація контролера для керування системою.

Але фітолампа не являється стандартним пристроєм в додатку. Його створено самостійно. Код пристрою «фітолампа» показано на рисунку 2.23

```
import {maxlight,maxRate,type,lastSeconds,timeDelay} form MyEnvVariables;
const setup = () => {
    setComponentOpacity(type.toString(), 1);
    attachInterrupt(0, isr);
    isr();
}

const isr = () => {
    inputDataSignal = analogRead(0);
    valueEnteredOn = map(inputDataSignal, 0, 1023, 0, 1);

    setComponentOpacity(type.toString(), 1-valueEnteredOn);
    setDeviceProperty(getName(), "level",input);
}

const loop = timeDelay => {
    updateEnvironment();
    delay(timeDelay);
}

const updateEnvironment = () =>{
    let countedRate = +(valueEnteredOn*maxlight*maxRate) / +(Environment.getVolume());
    Environment.setContribution("Visible Light", countedRate, countedRate, false);
}
```

Рисунок 2.23 – Код для роботи фітолампи.

Тобто, аналізуючи код можна сказати, що принцип роботи фітолампи в наступному: вона вмикається, якщо подано сигнал про вмикання з контролеру. Але якщо в приміщенні вже є світло, то фітолампа подасть зворотній сигнал до мікроконтролера, який в свою чергу передасть відповідь на сервер про те, що неможливо виконати команду, потрібно організувати лише один вид освітленості на робочому місці. І адміністратор відразу побачить, в чому проблема і виконає потрібні кроки для активізації системи.

2.3.3 Система зволоження повітря лабораторії

В цілому, система зволоження повітря схожа на систему осушення (рисунок 2.24). Вона також повинна бути в кожній біологічній лабораторії і повинна бути легко масштабована і адаптивна.

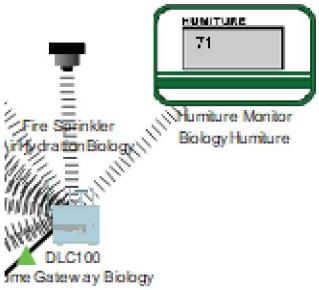


Рисунок 2.24 – Система зволоження повітря.

Крім того, самого пристрою зволоження немає в додатку, а пристрої для пожежогасіння чи поливу газона нам не підходять через надмірну подачу води. Конфігурацію пристрою можна побачити на рисунку 2.25

```
var VOLUME_AT_RATE = 10;


const setup = ()=> {

    IoEClient.onInputReceive = function(input) {
        processData(input, true);
    };

    attachInterrupt(0, function() {
        processData(customRead(0), false);
    });

    setState(state);
}

const processData=(data, bIsRemote)=>{return ? data.length<=0:se
```

Рисунок 2.25 – Пристрій зволоження повітря.

Логіка прийняття рішень системою показана на рисунку 2.26.

Edit	Remove	Yes	AirHydrationBiologyOn	BiologyHumiture Humitor <= 60	Set AirHydrationBiology Status to true
Edit	Remove	Yes	AirHydrationBiologyOff	BiologyHumiture Humitor >= 70	Set AirHydrationBiology Status to false

Рисунок 2.26 – Логіка прийняття рішень підсистеми зволоження повітря.

Якщо рівень вологості низький, потрібно увімкнути пристрій. Якщо рівень досягнув достатнього, вимкнути

2.3.4 Система попередження про пожежу

Однією з найважливіших систем є система контролю пожежної безпеки. Вона складається з двох частин: з фіксації задимленості та фіксації підвищення температури в приміщенні через вогонь. Розглянемо систему контролю за рівнем диму. Логіку роботи можна побачити на рисунку 2.27

Actions	Enabled	Name	Condition	Actions
Edit Remove	Yes	BiologySkomeSystemOn	Match any: • BiologyBiologySmokeDetector4 Level >= 0.21 • BiologySmoke2 Level >= 0.21 • BiologySmoke3 Level >= 0.21 • BiologyBiologySmokeDetector Level >= 0.21	Set BiologySiren On to true Set BiologyWindow3 On to true Set BiologyWindow2 On to true Set BiologyWindow On to true
Edit Remove	Yes	BiologySmokeSystemOff	Match any: • BiologySmoke2 Level <= 0.2 • BiologySmoke2 Level <= 0.2 • BiologyBiologySmokeDetector Level <= 0.2 • BiologyBiologySmokeDetector4 Level <= 0.2	Set BiologySiren On to false Set BiologyWindow3 On to false Set BiologyWindow2 On to false Set BiologyWindow On to false

Рисунок 2.27 Логіка роботи системи контролю рівня диму.

Тобто, якщо стандартний пристрій контролю (в нашому випадку smoke detector) диму фіксує перевищення заданої норми (близько 0.18-0.22), вмикається сирена, яка сповіщає локально та відправляє сигнал до віддалених керуючих пристроїв. Також відкриваються вікна для провітрення та виведення надлишків. Налаштування пристроїв для віддаленого контролю показано на рисунку 2.28

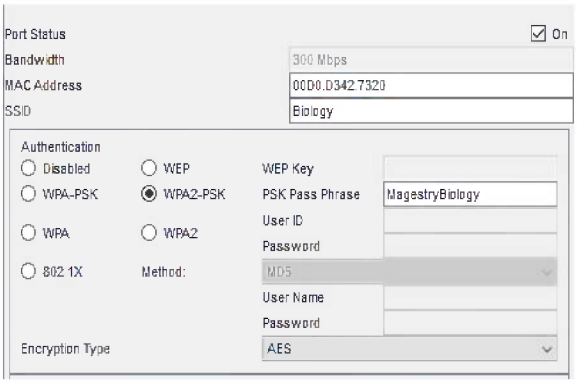


Рисунок 2.28 – Типове налаштування пристроїв контролю за димом.

Оскільки, система має працювати разом зі системою гасіння пожежі, налаштуємо її також. Для цього потрібно налаштувати контролер для правильного розподілення сигналів між компонентами двох систем. Конфігурація плати керування системою гасіння пожежі показано на рисунку 2.29.

```
def handleSensorData_second():
    second_value = digitalRead(5)
    if second_value == 0:
        customWrite(1, '0')
        customWrite(2, '0')
        customWrite(3, '0')
        customWrite(4, '0')
    else:
        customWrite(1, '1')
        customWrite(2, '1')
        customWrite(3, '1')
        customWrite(4, '1')
def main():
    add_event_detect(0, handleSensorData_zero)
    add_event_detect(5, handleSensorData_second)
    while True:
        delay(1000)
if __name__ == "__main__":
    main()
```

Рисунок 2.29 – Контролер для керування пожежогасінням.

Тобто, контролер зчитує дані на вхідних портах, до яких підімкнуті всі пристрої, перевіряє який це сигнал (0 - немає вогню, 1 - є) і відсилає їх на виходи, до яких під'єднані пристрої гасіння (1-увімкнутися, 0 - вимкнутися). Варто зазначити, що пристрої системи гасіння пожежі працюють на батарейках. Такі пристрої можуть працювати автономно від 2 до 5 років. Це зроблено для того, щоб не створити короткого замикання в системі під час гасіння водою, адже вона негативно впливає на дроти та дротове підключення в цілому.

2.3.5 Система автоматичного поливу рослин

Система автоматичного поливу повинна працювати як автоматично, в залежності від часу, так і напряму після команди від користувача. На рисунку 2.35 можна побачити систему автоматичного поливу.

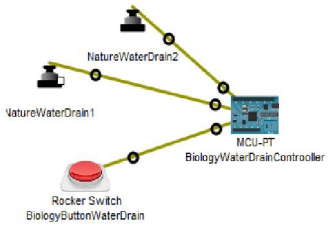


Рисунок 2.35 Система автоматичного поливу рослин.

Плата керування може отримувати команди як з кнопки, так і від серверу. Конфігурація роботи кнопки для поливу показана на рисунку 2.36

```
let is_open = false;
let P1 = 0;
let PImage = 1;

const setup= ()=> update(is_open);

const mouseEvent=(pressed, x, y, firstPress)=> {
    if (firstPress)
        update(!is_open);
}

const update=open=>{
    is_open = open;
    digitalWrite(PImage, is_open ? HIGH : LOW);
    digitalWrite(P1, is_open ? HIGH : LOW);
    setDeviceProperty(getName(), "state", is_open);
}
```

Рисунок 2.36 – Конфігурація кнопки для контролю системи автоматичного поливу рослин.

Тобто, кнопка задає два стани: увімкнути або вимкнути систему поливу. Кнопка потрібна для того, щоб системою міг керувати персонал лабораторії, якщо в нього немає доступу до віддаленого керування. Крім того, система поливу має працювати при пожежі, для того, щоб зберегти рослини. І працюватиме допоки не закінчиться вода або не подасться сигнал про відміну.

2.3.6 Система для контролю рівня кислотності грунту

Система контролю кислотності грунту потрібна для того, шо ефективно моніторити стан грунту, особливо для вибагливих рослин. Таку систему

потрібно робити адаптивною під різні структури грунту. Саму систему можна побачити на рисунку 2.37.

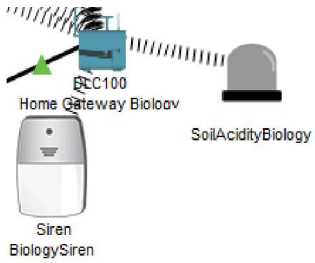


Рисунок 2.37 – Система контролю кислотності грунту.

Налаштуємо специфічний пристрій, якого немає в додатку. Програмна конфігурація показана на рисунку 2.38

```
const restoreProperty=(propertyName, defaultValue)=>
{
    let value = getDeviceProperty(getName(), propertyName);
    if ( !(value === "" || value == "undefined") ){
        if ( typeof(defaultValue) == "number" )
            value = Number(value);

        setDeviceProperty(getName(), propertyName, value);
        return value;
    }

    return defaultValue;
}


const loop=()=> detect();delay(1000);


const  detect=()=>
{
    let value = Environment.get(ENVIRONMENT_NAME);
    if (value >= 0 )
        setLevel( Environment.get(ENVIRONMENT_NAME));
}
```

Рисунок 2.38 – Пристрій моніторингу кислотності грунту.

Логіка роботи його наступна: коли стан грунту незадовільний, пристрій надсилає сигнал кожну секунду до сирени, щоб сповістити про умови. А сам моргає зеленим світлом кожну секунду для того, щоб візуально можна було зрозуміти, де умови стали незадовільними.

2.4 Проектування хімічної лабораторії

Проектування хімічної лабораторії має свої особливості, які характерні тільки таким типам лабораторій. Насамперед – забезпечити максимальний рівень безпеки персоналу, зручну систему оповіщення про надзвичайні випадки,

масштабованість та візуалізацію всіх процесів. Модель хімічної лабораторії можна побачити на рисунку 2.39

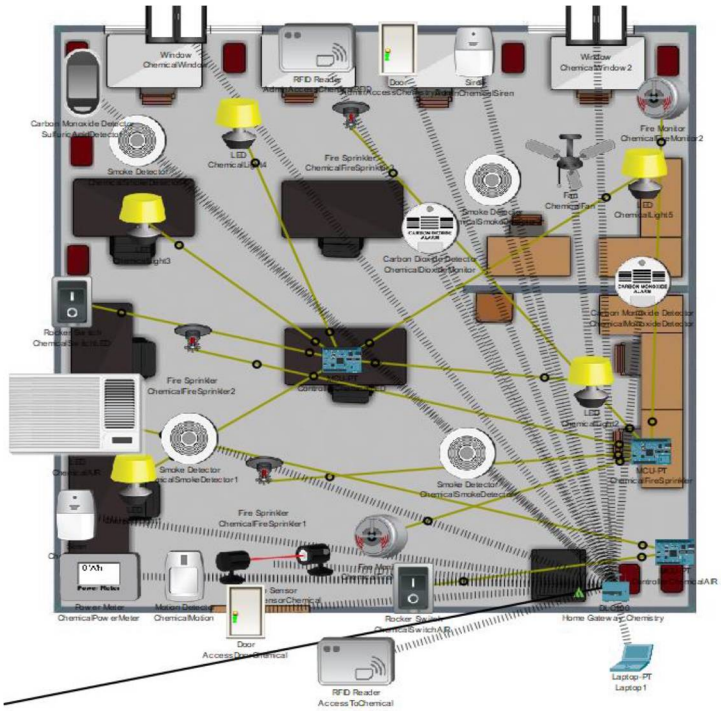


Рисунок 2.39 – Модель «розумної» хімічної лабораторії.

Фізичне відображення моделі продемонстровано на рисунку 2.40.

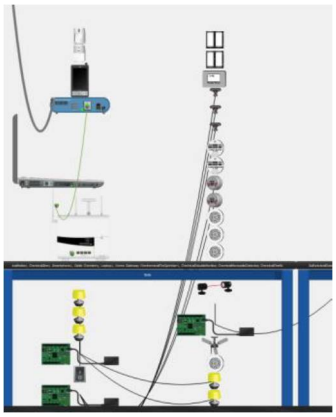


Рисунок 2.40 – Фізична топологія моделі.

Модель складається з низки підсистем, які виконують задачі, які притаманні тільки хімічній лабораторії. Розглянемо їх.

2.4.1 Системи контролю за рівнем чадного газу

Система моніторингу та сповіщення стосовно рівня CO важлива для хімічної лабораторії. Часто реагенти при змішуванні можуть виділяти надлишки CO, які є дуже шкідливими для людини. Тому доцільно створити «розумну» систему, яка зможе вчасно реагувати на різке підвищення рівня CO на робочому місці. Загалом, систему можна побачити на рисунку 2.41

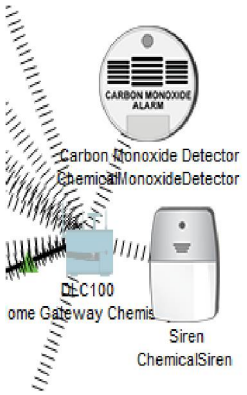


Рисунок 2.41 – Система контролю рівня CO.

Пристрій не є стандартним для додатку. Його потрібно створити (рисунок 2.42).

```
import {monooxide,maximumRate} from MyEnvVeriables
class Monoxide {
    static const InitialState=0;

    const setupInitialEnviroment=()=>{
        if (InitialState == 1 ){
            let data = maximumRate / Environment.getVolume();
            Environment.setContribution("CO", monooxide*data);
        }
        else Environment.setContribution("CO", 0);

    }

    const setupMonoxidProject = ()=> setState( ()=>restoreProperty("state", 0) );

    const clearProps = (property, initialValue)=>{
        let propertyData = getDeviceProperty(getName(), property);
        if ( !(propertyData == "undefined") ){
            if ( typeof(initialValue) == "number" )
                propertyData = Number(propertyData);

            setDeviceProperty(getName(), property, propertyData);
            return propertyData;
        }

        return initialValue;
    }


    const systemMouseIsPressed =(..., isPressed)=> isPressed ? setState(InitialState ? 0 : 1):undefined


    const setState=(createdNewStateSystem)=>{
```

Рисунок 2.42 – Пристрій контролю за рівнем чадного газу.

Якщо рівень в приміщенні перевищує норму, то вмикається сирена про сповіщення і персонал повинен виконати кроки, які прописані в техніці безпеки хімічної лабораторії для того, щоб видалити газ з робочого простору.

2.4.2 Система контролю за рівнем вуглекислого газу

Рівень вуглекислого газу може підвищуватися в лабораторії в двох випадках: під час пожежі та під час проведення хімічних реакцій. Тому доцільно створити таку систему з використання концепції Інтернету речей, роблячи її «розумною». Система складається з трьох частин: серверу, пристрою ідентифікації та сирени (рисунок 2.43).

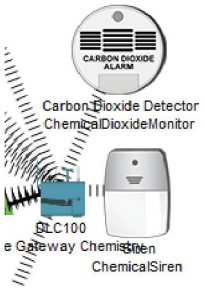


Рисунок 2.43 – Система контролю за рівнем вуглекислого газу.

Такий пристрій вже є в додатку, але він налаштований для того, щоб працювати у зв’язці з датчиком вогню, тому може не помітити явні, але замалі підвищення вуглекислого газу не в приміщенні, а на робочому місці загалом. Тому його потрібно модифікувати, розбивши більш «чутливим». Код переналаштування пристрою можна побачити на рисунку 2.44.

Рисунок 2.44 – Покращення пристрою фіксування вуглекислого газу.

Тобто, змінивши код і початкові параметри, переналаштувавши методи на нові задані показники, пристрій повинен фіксувати навіть мінімальні перевищення норми не тільки в приміщення, а і на кожному робочому місці, де можуть проводитися хімічні реакції.

2.4.3 Система очищення повітря

Часто буває, що повітря забруднене випарами, неприємними запахами чи жаром від сухого спирту для дослідів. Тому важливо організувати систему очищення та подачу свіжого повітря до приміщення. Така система повинна складатися з трьох компонентів – кнопки управління, контролером, який може самостійно обробляти вхідні і вихідні сигнали, та пристрою кондиціонування. Систему можна побачити на рисунку 2.45.

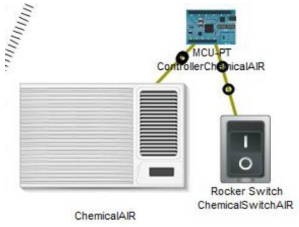


Рисунок 2.45 – Система очищення повітря.

Якщо персонал лабораторії захоче очистити приміщення, то потрібно увімкнути пристрій через кнопку. Але контролер функціонує таким способом, що не в залежності від бажань персоналу, він сам ввімкне пристрій очищення в наступних випадках:

1) коли завеликий рівень чадного газу;
2) коли завеликий рівень вуглекислого газу;
3) коли завеликий рівень кислотний випарів;
4) при перевищенні норми концентрації диму.

Тобто, використовуючи мови програмування, контролер та стандартний пристрій кондиціонування можна зробити систему «розумною», яка здатна

самостійно приймати рішення в залежності від параметрів зовнішнього середовища.

2.4.4 Система організації безпеки хімічної лабораторії

Хімічна лабораторія містить в собі рідкісні, часто дорогі і що найважливіше – небезпечні реактиви. Інколи досліди можуть проходити пасивно годинами. Для того, щоб убезпечити реактиви та дорогостоячі пристрої проведення хімічних досліджень, потрібно організувати безпеку лабораторії від зловмисників. Зробити це можна використати настінний датчик руху в парі з датчиком руху, який працює з використанням направленого лазера. Система показана на рисунку 2.46.

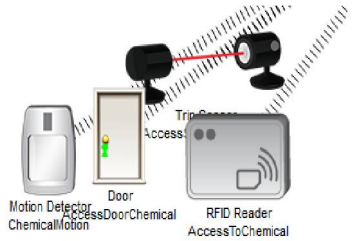


Рисунок 2.46 – Система безпеки лабораторії.

Для того, щоб розрізнити, что є свій, а хто чужий, можна використати датчики RFID. Якщо ключ вірний, датчики виключаються, а працівнику відкривається доступ до лабораторії. Умови прийняття рішень можна побачити на рисунку 2.47

Edit Remove	Yes	AccessLaboratoryOn	AccessToChemical Card ID = 222	Set AccessDoorChemical Lock to Unlock
Edit Remove	Yes	AccessLaboratoryOff	Match any: AccessToChemical Status is Waiting AccessToChemical Status is Valid	Set AccessDoorChemical Lock to Lock

Рисунок 2.47 – Умови прийняття рішень при доступі в приміщення.

В іншому випадку при несанкційному доступі, вмикаються всі наявні сирени і надсилається сигнал–тривога до органів правопорядку.

2.4.5 Система доступу до реагентів

Деякі реагенти небезпечні для життя, тому доступ до них має бути тільки у висококваліфікованих спеціалістів. Потрібно зробити систему, яка могла би по ключу ідентифікувати спеціаліста і дати йому доступ до потрібних складових. Для організації такого доступу ідеально підходить RFID датчики, які являються одними з перших пристроїв Інтернету речей. Систему доступу можна побачити на рисунку 2.48.

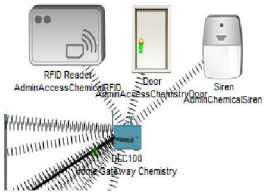


Рисунок 2.48 – Система доступу до небезпечних реагентів.

Логіка роботи в наступному: кожен працівник, який має доступ і розуміє всі ризики має при собі спеціальний електронний ключ, який має свій ідентифікатор. Якщо значення RFID ключа, який заданий в системі і ключа, який є в працівника співпадають, йому відкривається доступ. Якщо ключі не співпадають, звучить попереджуюча сирена про спробу несанкціонованого доступу (рисунок 2.49)

Edit Remove	Yes	AdminAccessChemicalOn	AdminAccessChemicalRFID Card ID = 225	Set AdminChemicalSiren On to false Set AdminAccessChemistryDoor Lock to Unlock Set AdminAccessChemicalRFID Status to Valid
Edit Remove	Yes	AdminChemicalAccessOff	AdminAccessChemicalRFID Status is Waiting	Set AdminAccessChemistryDoor Lock to Lock
Edit Remove	Yes	AdminAccessInvalid	AdminAccessChemicalRFID Status is Invalid	Set AdminChemicalSiren On to true Set AdminAccessChemistryDoor Lock to Lock

Рисунок 2.49 – Правила роботи доступу до реагентів.

Пристрій RFID є стандартним і його можна використовувати практично в будь-яких моделях, де потрібно організовувати доступ. Цей принцип працює в замках для під'їздів багатоквартирних будинків.

2.4.6 Система контролю рівня сірчаних випарів

Випари сірчаної кислоти є смертельно небезпечними для людини. Такі випари можуть бути тільки при складних хімічних реакціях, в яких можуть

використовуватися небезпечні чи агресивні реагенти. Тому потрібно створити високочутливий пристрій, який міг би ловити навіть незначні коливання, які перевищують норму парів на робочому місці науковця. Схема системи показана на рисунку 2.50.

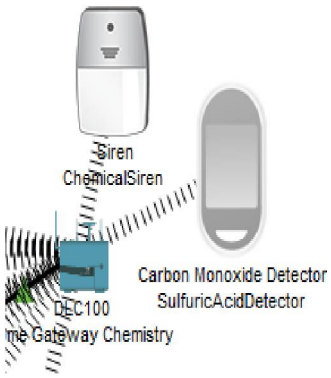
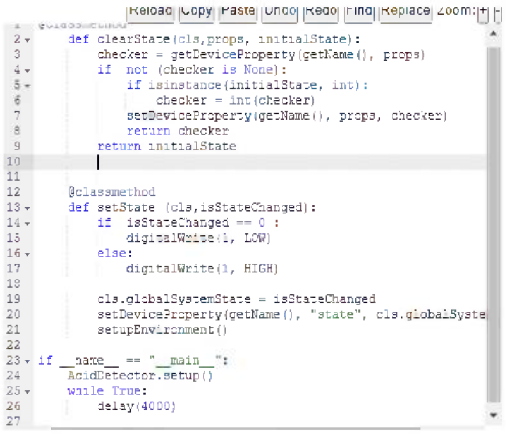


Рисунок 2.50 – Система контролю норми парів сірчаної кислоти.

Пристрій створено за допомогою мови програмування Python з використанням ООП підходу для формування більшої чутливості пристрою до впливів навколишньої середи. Код пристрою можна побачити на рисунку 2.51.



```
    def clearState(cls,props, initialState):
        checker = getDeviceProperty(getName(), props)
        if  not (checker is None):
            if isinstance(initialState, int):
                checker = int(checker)
            setDeviceProperty(getName(), props, checker)
            return checker
        return initialState

    @classmethod
    def setState (cls,isStateChanged):
        if  isStateChanged == 0 :
            digitalWrite(1, LOW)
        else:
            digitalWrite(1, HIGH)

        cls.globalSystemState = isStateChanged
        setDeviceProperty(getName(), "state", cls.globalSyste
        setupEnvironment()

if __name__ == "__main__":
    AcidDetector.setup()
    while True:
        delay(4000)
```

Рисунок 2.51 – Програмна реалізація пристрою контролю за нормою випарів.

За допомогою концепції Інтернету речей було створено новий пристрій, який значно спростить робочий процес та проведення небезпечних хімічних дослідів. При різкому чи повільному зростанні рівня парів кислоти, детектор спрацює, увімкнеться сирена, яка відразу сповістить робочий персонал про

надзвичайну ситуацію і дозволить оперативно відреагувати і прийняти правильні рішення.

Роблячи висновок, можна сказати, що за допомогою технології Інтернету речей можна створити комплексні системи, які здатні працювати синхронно чи асинхронно, в залежності від поставленої задачі. Системи можуть бути адаптивними, масштабировании, керуватися віддалено і самостійно приймати рішення в залежності від встановлений правил та параметрів. Використання мов програмування робить пристрої більше гнучкими та дає дозвіл покращувати уже існуючі пристрої, додаючи в них новий функціонал, при цьому зберігаючи старий.

2.5 Висновки до розділу II

На основі додатку Cisco Packet Tracer було створено модель «розумної» науково-дослідницької лабораторії. Вона складається з трьох частин – хімічної, біологічної лабораторій та серверної частини.

Керувати моделлю можна як віддалено так і локально. Кожна частина складається зі своїх підсистем. Кожна підсистема може працювати як незалежно, так і синхронно з іншими підсистемами.

Для специфічних систем кожної лабораторії, за відсутності стандартних, було створено власні пристрої для своїх задач за допомогою мов програмування, які підтримує додаток. Пристрої були протестовані як за допомогою зміни параметрів зовнішнього середовища, так і за допомогою unit-тестів. Для кожної системи були налаштовані свої параметри зовнішнього середовища. Вся модель розділена на різні фізичні рівні для зручності масштабування та налаштування.

РОЗДІЛ ІІІ
ТЕСТУВАННЯ РОЗРОБЛЕНОЇ МОДЕЛІ

3.1 Системні та технічні вимоги до розробленої моделі

Для того, щоб модель «розумної» науково-дослідницької лабораторії працювала на персональному комп'ютері, потрібно встановити додаток Cisco Packet Tracer. Скачати його можна з офіційного сайту компанії. Документація до додатку[14] говорить, що мінімальні вимоги до системи мають бути наступними:

1) Операційна система Windows 8/10/11 (32/64bit) або Linux Mint, або Ubuntu не раніше 18 версії, або MacOS;
2) місце на диске – 1.5 Гб;
3) ОЗУ – 4 ГБ;
4) тип процесору – Intel (не раніше 9 покоління) або AMD;
5) мінімум двоядерний процесор.

Крім того, для максимально ефективного використання додатку слід мати стабільне підключення до мережі з використанням Wi–Fi чи Ethernet підключення. Також, зі зростанням складності розробленої моделі, будуть потрібні більше сильні технічні характеристики пристрою, адже при використанні багатьох пристроїв одночасно, налаштуванні та створенні нових пристроїв, розділенні моделі на підсистеми та різні топології, завданні десятків параметрів зовнішньої середи, навантаження на процесор буде зростати.

3.2 Тестування серверної частини

Для того, щоб протестувати роботу серверної частини моделі, яка відповідає за віддалений чи локальний контроль за пристроями та прийняттям рішень, потрібно використати будь-який керуючий пристрій, наприклад Laptop. В його меню є IoT server (рисунок 3.1)

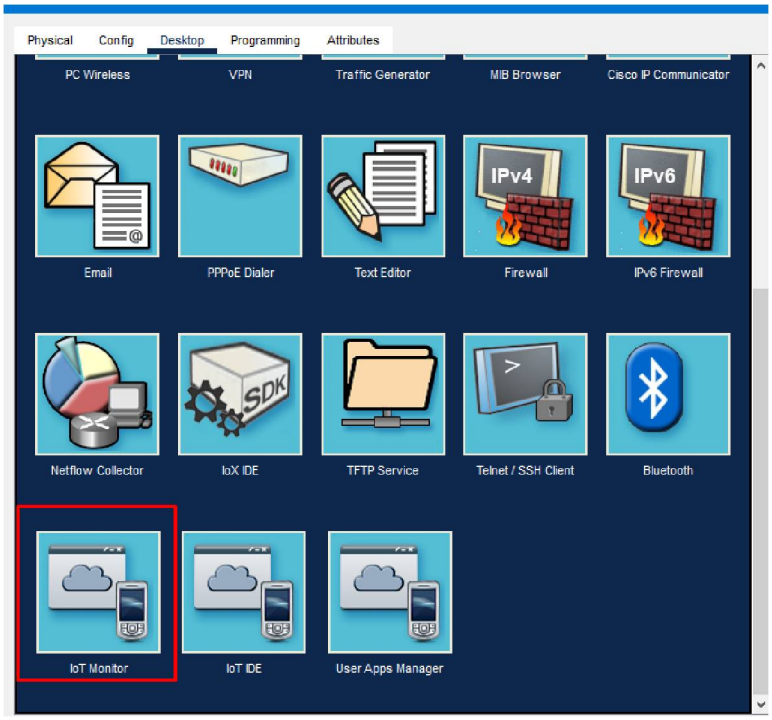


Рисунок 3.1 Можливості керуючого пристрою.

При виборі даного меню в користувача запросять ввести адрес, логін та пароль. Пароль та логін стандартні – admin. Сервер було налаштовано за допомогою DNS і його адрес www.magestryiot.com. Приклад дивитися на рисунку 3.1.

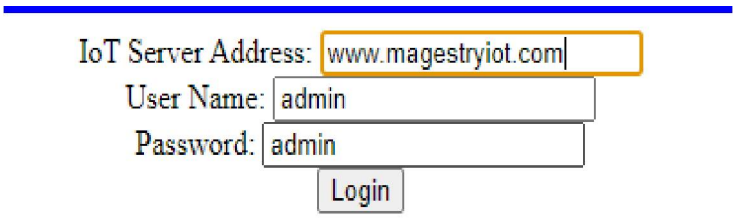


Рисунок 3.2 – Авторизація для керування пристроями.

Якшо введені невірні логін чи пароль, з'явиться наступне меню (рисунок 3.3)

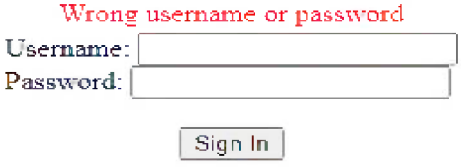


Рисунок 3.3 – Неправильні дані для авторизації.

В іншому випадку, коли логін, пароль та адрес серверу правильні, користувачу відкриється доступ до можливості керувати моделлю, налаштовувати пристрої чи логіку прийняття рішень (рисунок 3.4).

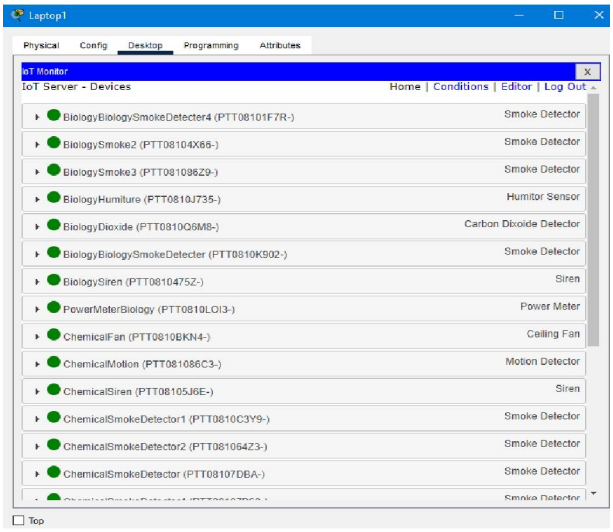


Рисунок 3.4 – Меню керування для авторизованих користувачів.

Крім того, перевіримо роботу DHCP-серверу. Візьмемо будь-який пристрій моделі і перевіримо, чи є в нього автоматично отриманий IP-адрес(рисунок 3.5).

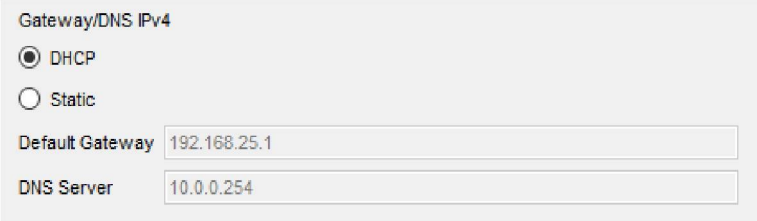


Рисунок 3.5 – Автоматичне отримання пристроєм IP-адреси.

Тобто, вся серверна частина працює коректно. Є можливість ідентифікувати пристрої за допомогою автоматично отриманого IP-адресу, є можливість керування моделлю віддалено за допомогою кінцевих пристроїв. Авторизація проходить успішно на сервері.

3.3 Тестування хімічної лабораторії

Розділимо модель хімічної лабораторії на підсистеми та протестуємо їх окремо.

1. Розглянемо систему контролю доступу до небезпечних реактивів. Для тестування потрібно використати стандартний пристрій, який називається RFID Card, який генерує ключ, що має співпасти з ключем, який заданий в RFID Reader. Якщо ключі співпадають, користувач отримує доступ (дивитися рисунок 3.6). Зелене світло означає, що ідентифікація пройшла успішно.

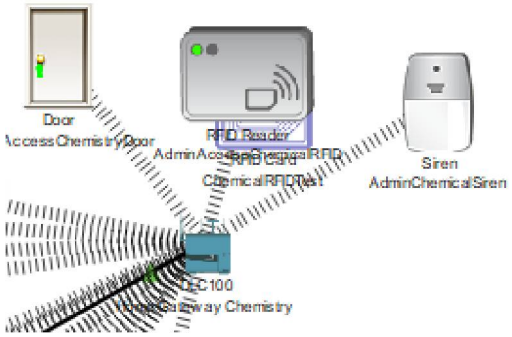


Рисунок 3.6 – Успішна ідентифікація користувача.

В іншому випадку, коли ключ не вірний, чи є місце навмисному злонамірному втручанню, ввімкнеться сирена, а доступу до місця зберігання не буде (рисунок 3.7).

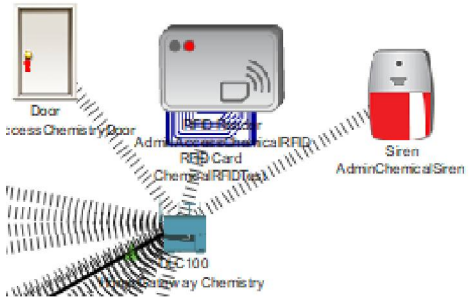


Рисунок 3.7 – Неуспішна ідентифікація користувача.

Тобто, система доступу до небезпечних реагентів працює правильно і здатна самостійно вирішувати, чи ідентифікація успішна, чи ні і в залежності від результату приймати подальші дії.

2. Протестуємо роботу системи контролю за випарами сірчаної кислоти. Для цього нам потрібно створити пристрій, який імітує збільшення парів. Код пристрою можна побачити на рисунку 3.8

```
const restoreProperty = (incomeProps, initialData)=>{
    let getProp = getDeviceProperty(getName(), incomeProps);
    if ( typeof(initialData) == "number" ) {
        getProp = Number(getProp);
    }
    else{
        setDeviceProperty(getName(), incomeProps, getProp);
        return getProp;
    }
    return initialData;
}

const mouseEvent = (pressed, x, y, firstPress)=> if (firstPress) toggleState();

const processData = (incomeData, isRemote=false)=>{
    if ( incomeData.length <= 0 ) return;
    setState(parseInt(incomeData.split(",")[0]));
}

const sendReport =()=>{
    customWrite(0, globalState);
    IoEClient.reportStates(globalState);
    setDeviceProperty(getName(), "state", globalState);
}

const setState(newIncomeState) =>{
    analogWrite(A1, newIncomeState);
```

Рисунок 3.8 – Код пристрою імітації збільшення парів сірчаної кислоти.

Якщо показник кількості парів вище встановленої технікою безпекою норми, то вмикається сирена, яка відразу сповіщає персонал. Результат можна побачити на рисунку 3.8.

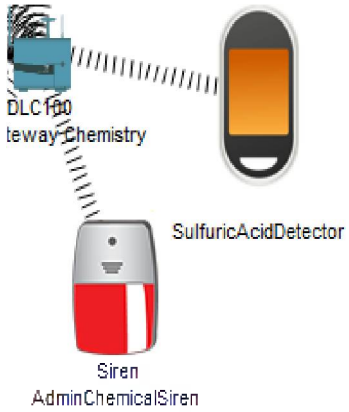


Рисунок 3.8 – Тестування система контролю за парами сірчаної кислоти.

Тобто, в залежності від рівня парів, система може в автоматичному режимі сповіщати про надзвичайну ситуацію, а в іншому випадку бути активною, але не подавати сигналів до користувачів та пристроїв.

3. Проведемо тестування системи фільтрації повітря. Скажімо, що персонал після хімічного експерименту з високим виділенням різних парів хоче очистити приміщення. Для цього можна активувати систему за допомогою кнопки. Результат можна побачити на рисунку 3.9

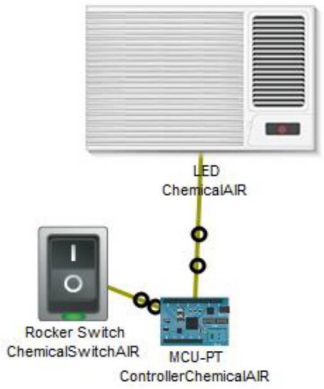


Рисунок 3.9 – Ручна активація системи очищення повітря.

Але бувають випадки, коли модель сама повинна прийняти рішення, наприклад коли персоналу немає, чи слід очищати повітря (наприклад при розбитті реагентів чи невеликій пожежі). На рисунку 3.10 показано активацію системи при перевищенні норми диму в приміщення

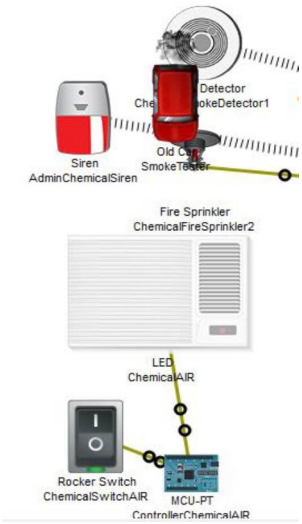


Рисунок 3.10 – Автоматична активація системи очищення повітря.

Можна сказати, що система «розумна» тому що можна працювати як в стаціонарному режимі, шляхом активації при натискання на кнопку, так і автоматично, базуючись на оброблених даних, які приходять з навколишнього середовища.

4. Зробимо тестування системи оповіщення про надмірність чадного газу в лабораторії. Для цього потрібно створити пристрій за допомогою мов програмування, який здатний імітувати ріст рівня CO в приміщенні. Код можна побачити на рисунку 3.11

```
import {monooxide,maximumRate} from MyEnvVeriables
class Mooxide {
    static const InitialState=0;

    const setupInitialEnviroment=()=>{
        if (InitialState == 1 ){
            let data = maximumRate / Environment.getVolume();
            Environment.setContribution("CO", monooxide*data);
        }
        else Environment.setContribution("CO", 0);

    }

    const setupMonooxidProject = ()=> setState( ()=>restoreProperty("state", 0) );
```

Рисунок 3.11 – Програмування пристрою імітації підвищення рівня чадного газу.

Якщо рівень високий, ввімкниться сирена, яка відразу сповістить працівників про небезпечну ситуації. Результати тестування можна побачити на рисунку 3.12

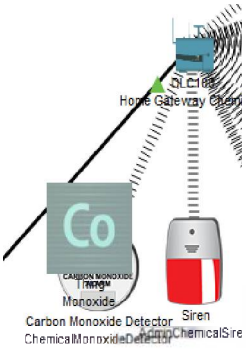


Рисунок 3.12 – Результати тестування системи оповіщення про надмірність чадних газів.

5. Проведемо тест для перевірки правильності роботи системи сповіщення про надлишок вуглекислого газу в лабораторії, чи то після хімічних дослідів, чи через пожежу. Для цього потрібно створити пристрій, який при активації зможе підвищувати рівень вуглекислого газу в приміщення. Код продемонстровано на рисунку 3.13

```
from ioeclient import IoEClient
from My_env_variables import *

class DioxideDetector:
    globalSystemState = 0

    @classmethod
    def setupEnvironment (cls):
        if  cls.globalSystemState == 1 :
            volume = 100000 / Environment.getVolume()
            Environment.setContribution("Dioxide", dioxide*volume, maximumRate, True)
        else:
            Environment.setContribution("Dioxide", 0, 0, True)

    @classmethod
    def setupDioxide (cls):
        cls.globalSystemState = restoreProperty("state", 0);
        setState(cls.globalSystemState)
```

Рисунок 3.13 – Конфігурація пристрою фіксації для створення надлишку CO2.

Коли рівень стане більше за нормальний безпечний рівень, то сирена сповістить про ситуацію і у працівників буде час для виправлення надзвичайної ситуації, шляхом фільтрації повітря як приклад. Результати проведених тестів відображені на рисунку 3.14

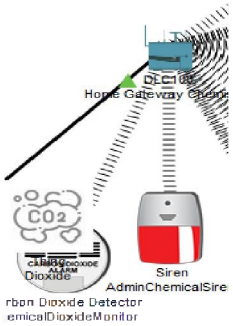


Рисунок 3.14 – Результати тестів системи контролю за рівнем вуглекислого газу.

3.4 Тестування біологічної лабораторії

1. Фітолампи потрібні для більше ефективного росту рослин. Протестуємо роботу системи. Її можна ввімкнути вручну, за допомогою кнопки, якщо при цьому немає альтернативного джерела світла. Результат на рисунку 3.15

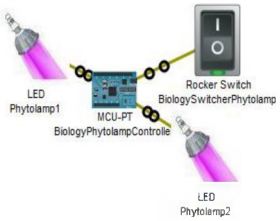


Рисунок 3.15 – Активація фітоламп.

Коли є альтернативне світло, наприклад звичайна лампа, спрацює попередження і фітолампа не буде активна. Результат на рисунку 3.16

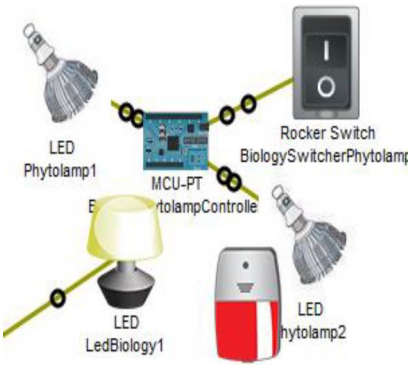


Рисунок 3.16 – Система не активна при присутності іншого типу освітленості.

На рисунку LED – це звичайна лампа, яка активована вручну. Як ми бачимо, логіка системи активації фітоламп працює як і було задумано.

2. Продемонструємо систему поливу грунту. Вона може бути активована вручну за допомогою кнопки керування. Проте система за одиницю часу видає в багато разів менше води, ніж наприклад пристрій тушіння пожежі. Результат активації системи вручну можна побачити на рисунку 3.17

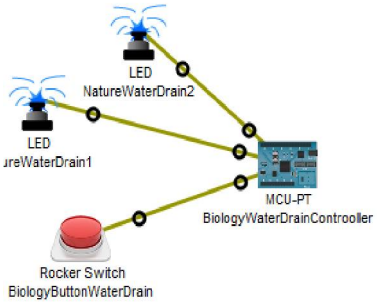


Рисунок 3.17 – Активація поливу рослин вручну.

Розглянемо як систему активується при пожежі в приміщенні. Результат тестування показано на рисунку 3.18.

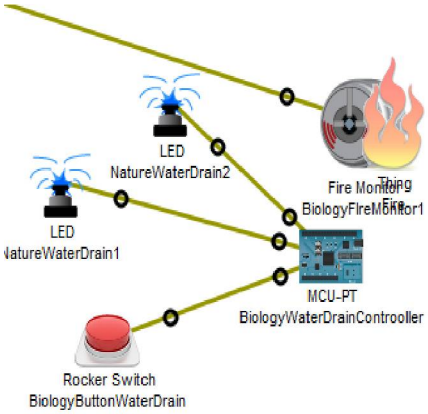


Рисунок 3.18 – Автоматична активація системи поливу грунту.

Тобто, коли спрацьовує один із датчиків вогню, він подає сигнал на сервер, а сервер в свою чергу подає сигнал на контроллер для того, щоб ввімкнути систему поливу грунту.

3. Протестуємо систему осушення повітря для забезпечення кращих умов для росту та розвитку екосистем біологічної лабораторії. Якщо рівень вологості вище 75, то вмикається осушувач. Принцип роботи показано на рисунку 3.19

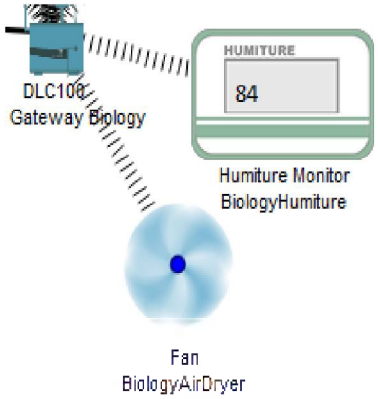


Рисунок 3.19 – Роботи системи осушення повітря.

Пристрій Fan налаштований так, щоб підсушувати повітря на коефіцієнт за одиницю часу. Логіку автоматичного прийняття рішень було задано за на сервері, з якого пристрою системи і отримують вказівки щодо режимів роботи. Роботу системи, коли вологість в рамках норми показано на рисунку 3.20.

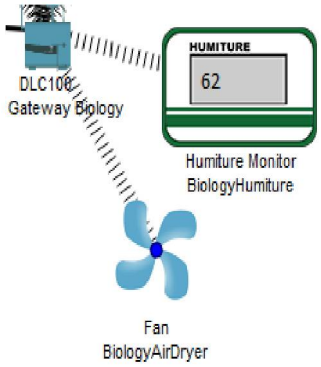


Рисунок 3.20 – Вид неактивної системи осушення повітря.

4. Система зволоження повітря повинна працювати як противовіс осушення. Коли повітря засухе (нижче вказаної для спеціальних умов норми), система активується (рисунок 3.21).

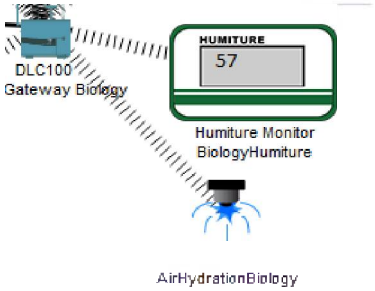


Рисунок 3.21 – Активація системи зволоження при занадто сухому повітрі.

Але оскільки система може за одиницю часу розбризкувати воду, то логічно її покращити, додавши можливість активації при пожежі і не потрібно враховувати загальний рівень вологості. Активацію системи при вогні можна побачити на рисунку 3.22.

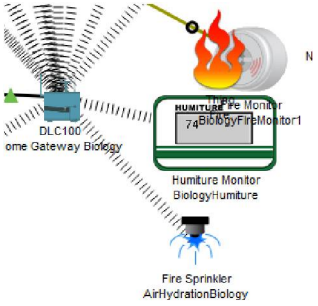


Рисунок 3.22 – Активація системи при пожежі в приміщенні.

Тобто створена система може функціонувати в двох випадках – при прямому призначення для зволоження повітря і розбризкувати воду при пожежі. 5. Однією з найважливіших систем будь-якої «розумної» моделі – це система пожежогасіння. Вона повинна спрацьовувати, коли будь-який датчик фіксує вогонь. При цьому вмикається стандартний пристрій (Fire sprinkler) для гасіння і вмикається сирена, щоб повідомити про критичну ситуацію. Спочатку требо створити пристрій, який може імітувати наявність вогню в системі. Код можна побачити на рисунку 3.23.

```
class fire{
    setup=()=>setDeviceProperty(getName(), 'IR', 900);
}

fire= new Fire()
fire.setup()
```

Рисунок 3.23 – Створення пристрою імітації вогню.

Тепер можна протестувати систему (рисунок 3.24)

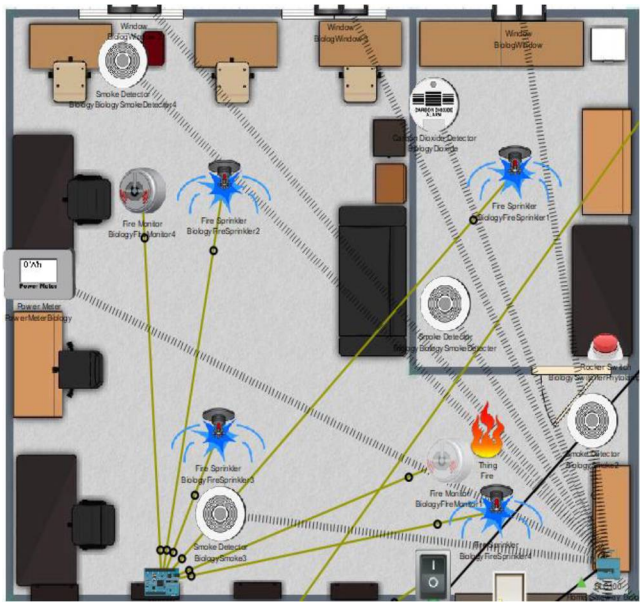


Рисунок 3.24 – Система гасіння пожежі біологічної лабораторії.

Крім того, система працює синхронно разом з системою фіксації надлишків диму в приміщенні. Коли рівень диму стає більшим норми рівно в 2

рази, то це є сигналом того, що скоріше за все сталася пожежа. Для імітації диму можна використати спеціальний стандартний пристрій CPT – old car. Результат тестування синхронної роботи двох систем показано на рисунку 3.25

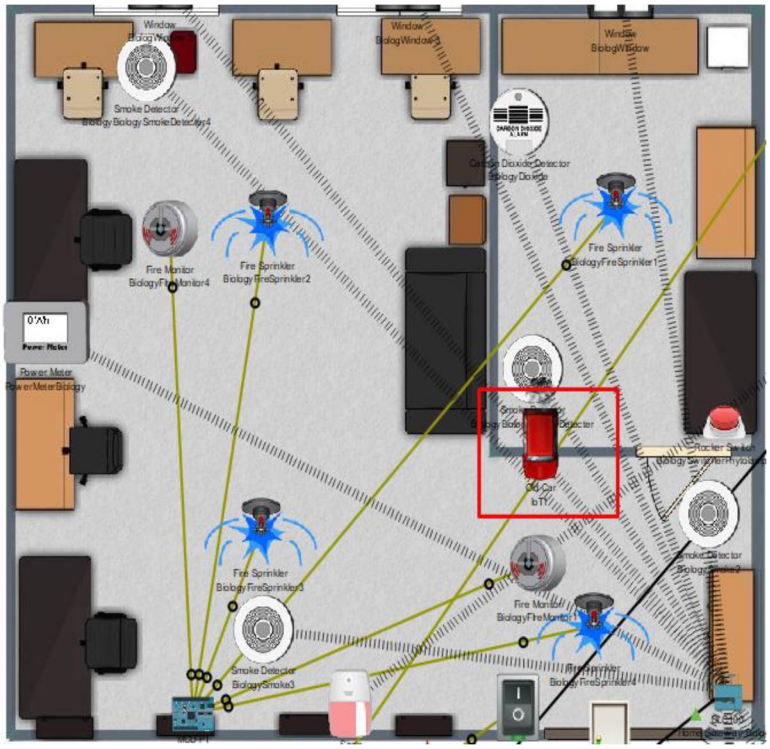


Рисунок 3.25 – Активація системи пожежогасіння при перевищенні рівня диму в два рази.

Тобто, за допомогою стандартних та новостворених пристроїв за допомогою мов програмування Python та JavaScript вдалося успішно протестувати біологічну та хімічну лабораторію і на основі тестів зробити висновок, що кожна підсистема виконує ті задачі, які перед нею поставлені і має здатність без втручання людини приймати рішення в тих чи інших ситуаціях. Важливо пам’ятати, що кожен тип лабораторії має свої особливості, тому деякі підсистеми можуть функціонувати не зовсім коректно для інших типів. Для того, щоб уникнути таких проблем можна додати більше «розумних» пристроїв або, маючи навички програмування, покращити вже існуючі, зробити їх більш «чутливими» до сигналів з навколишнього середовища, від користувача чи від інших пристроїв поточної чи іншої лабораторії

3.5 Висновки до розділу III

В розділі були описані методи тестування моделі як комплексно, так і окремо кожну підсистему. Для тестування використовувалися як стандартні пристрої та зовнішнього середовища. Для деяких підсистем було створено пристрої-тестування, які можуть імітувати поведінку деяких специфічних параметрів зовнішнього середовища, наприклад рівень парів сірчаної кислоти.

Розглянуто системі вимоги до пристроїв, де модель може налаштуватися. Проведено тестування DNS та DHCP- серверів та можливість ідентифікації пристроїв в рамках моделі. За допомогою кінцевих пристроїв є можливість керувати пристроями та налаштовувати сценарії та умови прийняття рішень в залежності від сигналів пристроїв, які надсилають їх за в залежності від впливу та параметрів середовища, де вони знаходяться.

ВИСНОВКИ

В кваліфікаційній роботі розглянуті особливості сучасної концепції Інтернету речей. Причини її популярності, активного розвитку та майбутній розвиток. Розділення ринку IoT, загальні вимоги до сучасних систем домашньої автоматизації, які потрібно виконувати для створення найбільш конкуретних пристроїв автоматичного прийняття рішень.

Розглянуто способи передачі даних в системах домашньої автоматизації з використанням різних протоколів, що використовуються найбільшими розробниками пристроїв Інтернету речей. Проведено порівняння протоколів та проаналізовано способи передачі даних, які активно будуть використовуватися в майбутньому, наприклад 5G.

Проведено порівняння найсучасніших додатків для творення моделей з використанням IoT. На основі отриманих результатів було прийнято рішення використовувати додаток Cisco Packet Tracer. Розглянуто його основні можливості стосовно створення моделей та можливості до масштабування чи адаптивності шляхом додавання своїх пристроїв.

В процесі дослідження було розроблено модель «розумної» науково-дослідницької лабораторії, яка складається з трьох частин: біологічної та хімічної лабораторії та серверної частини, яка відповідає за створення умов для керування моделлю віддалено та локально за допомогою будь-яких кінцевих пристроїв, якщо користувач має відповідні права. Налаштовано DHCP та DNS-сервери для динамічної ідентифікації пристроїв в моделі. Показано системні вимоги для створення такого типу моделей.

Для кожної підсистеми моделі було створено свої як залежні, так і незалежні сценарії роботи, які базуються на отриманих сигналах з навколишнього середовища. Модель здатна самостійно приймати рішення в залежності від отриманих сигналів. Адміністратор може простежувати, додавати, видаляти чи змінювати сценарії роботи.

Для специфічних підсистем, які характерні тільки окремим типам лабораторій, наприклад хімічній, було створено та інтегровано свої пристрої автоматизації, які працюють синхронно з уже існуючими. Для відтворення деяких зовнішніх факторів також були створені окремі пристрої з використанням мов програмування. Кожна частина моделі «розумної» науково-дослідницької лабораторії характерна своїми особливостями, сценаріями роботи та прийняттям автоматичних рішень. Системи можуть працювати синхронно, не впливаючи одна на одну, але при критичних ситуаціях, можуть обмінюватися сигналами для забезпеченням максимальної безпеки персоналу та самої системи.

На основі проведеного тестування моделі можна зробити висновок, що система показала максимальну роботоздатність. Всі процеси протікають так, що можна легко зрозуміти, як працює модель. Можна вносити практично будь-які зміни як в умови прийняття рішень, так і в самі системи шляхом додавання чи видалення пристроїв, тобто модель характеризуються високою адаптивністю та масштабованістю. Такі типи моделей можливо інтегрувати в уже існуючі лабораторії різних типів та характеристик практично без втрати функціоналу.

Тобто, використовуючи найсучасніші технології, можна розробити системи, які зможуть покращити, пришвидшити, спростити проведення будь-яких наукових дослідів, експериментів та розробку наукових напрацювань. За допомогою Інтернету речей можна зробити таку залежність – Інтернет речей допомагає розвитку технологій, які в свою чергу допомагають розвитку Інтернету речей.

Дана тема дослідження була апробована на щорічній міжнародній науково-технічній конференції «Комп'ютерне моделювання в наукоємних технологіях» (КМНТ-2022).

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hassan, Q.F.; Khan, A. ur R.; Madani, S.A. Internet of Things: Challenges, Advances, and Applications. – 2017. – 130c.
2. Bastos, D.; Shackleton, M.; El–Moussa, F. (2018). «Internet of Things: A Survey of Technologies and Security Risks in Smart Home and City Environments». Living in the Internet of Things: Cybersecurity of the IoT – 2018. – 215c.
3. Silvio Ereno Quincozes – MQTT Protocol: Fundamentals, Tools and Future Directions. 2019. – 170c.
4. Postel, J. User Datagram Protocol. Internet Engineering Task Force. 2014. – 45c.
5. Postel, Jonathan. «Internet Protocol Version 2». 2022. – 226c.
6. Bill Cerveny «IPv6 Fragmentation». Arbor Networks. 2016. – 98c.
7. Catalin Cimpanu «Cloudflare, Google Chrome, and Firefox add HTTP/3 support». 2019. – 458c.
8. Zhang, Yongbin; Liang, Ronghua; Ma, Huiling (2012). «Teaching Innovation in Computer Network Course for Undergraduate Students with Packet Tracer». 2018. – 518c.
9. Roy Want ; Schilit, Bill N.; Jenson, Scott. «Enabling the Internet of Things». 2015. – 87c.
10. Amy Nordrum: «Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated». 2018. – 113c.
11. Anthony Brush: «Home Automation in the Wild: Challenges and Opportunities». 2015. – 318c.
12. Kenneth Lange: The Little Book on REST Services. 2019. – 78c.
13. Roy Fielding: Hypertext Transfer Protocol : Semantics and Content, Section 4. 2018. – 154c.
14. Kristin Masters. The Impact of Industry 4.0 on the Automotive Industry. 2017. –58c.

15. Nitin Dahad. «Designer's Guide to IIoT Security. 2019. – 67с.
16. Energy Management Framework. [Електронний ресурс]. Режим доступу до ресурсу: https://datatracker.ietf.org/doc/html/rfc7326. (Дата звернення 25.09.2022).
17. Tech pages/IoT systems. [Електронний ресурс]. Режим доступу до ресурсу: https://xmpp.org/extensions/xep–0323.html . (Дата звернення 03.06.2022).
18. IoT data security vulnerable as connected devices proliferate. [Електронний ресурс]. Режим доступу до ресурсу: https://www.techtarget.com/iotagenda/definition/IoT–security–Internet–of–Things–security. (Дата звернення 08.08.2022).
19. Douglas Crockford. «Prototypal Inheritance in JavaScript». 2017. – 764с.
20. The Cain Gang Ltd. Python Metaclasses. 2020. – 21с.
21. Cisco Packet Tracer. [Електронний ресурс]. Режим доступу до ресурсу: https://web.archive.org/web/20120914231610/https://www.cisco.com/web/offer/emea/7130/docs/PT5.pdf (Дата звернення 18.07.2022).
22. Download Cisco Packet Tracer 8.2 & GNS3. [Електронний ресурс]. Режим доступу до ресурсу: https://web.archive.org/web/20220524235332/https://www.packettracernetwork.com/download/download–packet–tracer.html. (Дата звернення 13.10.2022).
23. Chris Richardson. Microservice architecture pattern.2022.– 145с.
24. How Does RFID Technology Work?.[Електронний ресурс]. Режим доступу до ресурсу: https://www.makeuseof.com/tag/technology–explained–how–do–rfid–tags–work. (Дата звернення 29.09.2022).
25. Xiaomi unveils Wi-Fi Amplifier, Bedside Smart Lamp & A Bluetooth Headset. [Електронний ресурс]. Режим доступу до ресурсу: http://www.gizmochina.com/2015/06/10/xiaomi-unveils-wi-fi-amplifier-bedside-smart-lamp-a-bluetooth-headset/. (Дата звернення 03.09.2022).
26. Обладнання по-українськи: як Ajax Systems розробляє і виробляє гаджети для безпеки. [Електронний ресурс]. Режим доступу до ресурсу: https://ain.ua/ru/2016/08/18/hardware-po-ukrainski-kak-ajax-systems-

razrabatyvaet-i-proizvodit-gadzhety-dlya-bezopasnosti/. (Дата звернення 23.10.2022).

27. Побудуйте свій дім разом з BroadLink. [Електронний ресурс]. Режим доступу до ресурсу: https://www.broadlink.ae/build-your-smart-home/. (Дата звернення 23.10.2022).
28. 21 Open Source Projects for IoT. [Електронний ресурс]. Режим доступу до ресурсу: https://www.linux.com/NEWS/21-OPEN-SOURCE-PROJECTS-IOT/. (Дата звернення 12.10.2022).
29. Smart City Coupe test results. [Електронний ресурс]. Режим доступу до ресурсу: https://www.euroncap.com/en/ratings-rewards/latest-safety-ratings/. (Дата звернення 17.09.2022).
30. Cisco Systems Summary. [Електронний ресурс]. Режим доступу до ресурсу: https://finance.yahoo.com/quote/csco?ltr=1. (Дата звернення 09.11.2022).

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) Магістр
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 – Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної
та прикладної системотехніки
д.т.н., проф. Шматков С. І.
«10» _____11_____2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Волинського Валентина Володимировича
(прізвище, ім'я, по батькові студента)

1. Тема роботи: Модель "розумної" науково-дослідницької лабораторії
керівник роботи Мірошник Марина Анатоліївна, доктор технічних наук, професор ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "___"_____20__року №_____

2. Строк подання студентом роботи _____30 листопада 2022_____

3. Перелік питань, які потрібно розробити.

1) Аналіз технології управління "розумним домом"

2) Постановка задачі моделювання "розумної" навчально-дослідницької лабораторії

3) Розробка моделі Smart-лабораторії на основі додатку Cisco Packet Tracer

4) Дослідження розробленої моделі

5) Розробка рекомендацій по впровадженню технології "розумного" дому в навчально-дослідницьку лабораторію

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання виконання етапів роботи
1.	Аналіз і підбір літератури та створення літературної бази для для розробки моделі	19.10.2021 - 09.01.2022
2.	Аналіз сучасних прикладів індустріального Інтернету речей, розумного дому	10.01.2022 - 01.02.2022
3.	Розробка програмної моделі “розумної” науково-дослідницької лабораторії	02.02.2022 - 30.03.2022
4.	Розробка та тестування власних “розумних” пристроїв для науково-дослідницької лабораторії	01.04.2022 - 01.05.2022
5.	Розробка технічного завдання на розроблену науково-дослідницьку лабораторію	01.05.2022 - 25.05.2022
6.	Розробка програми і методики тестування розробленої моделі	26.05.2022 - 20.06.2022
7.	Розробка інструкцій для користувача	21.06.2022 - 15.07.2022
8.	Створення пояснювальної записки кваліфікаційної роботи	16.07.2022 - 01.09.2022
9.	Оформлення звіту за результатами науково-дослідницької практики	01.09.2022 - 30.09.2022
10.	Оформлення звіту за результатами переддипломної практики	01.10.2022 - 15.10.2022
11.	Підготовка доповіді на тему кваліфікаційної роботи на науково-технічну конференцію	16.10.2022 - 01.11.2022
12.	Підготовка до попереднього захисту роботи	02.11.2022 - 14.11.2022
13.	Представлення кваліфікаційної роботи науковому керівнику	15.11.2022 - 23.11.2022
14.	Представлення кваліфікаційної роботи науковому рецензенту	24.11.2022 - 30.11.2022

5. Дата видачі завдання _____10. 12. 2021_____

Студент Волинський В.В. підпис
ініціали, прізвище

Керівник роботи Мірошник М.А. підпис
ініціали, прізвище

Додаток Б

Затверджую

_____

«____»_____2022р.

Технічне завдання

на розробку програмного виробу

«Модель «розумної» науково-дослідницької лабораторії»

Назва розділу	Назва і зміст підрозділу
1. Введення	1.1. Назва програмного виробу: модель «розумної» науково-дослідницької лабораторії 1.2. Галузь застосування: хімія, біологія, комп'ютерні технології моделювання.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – «Комп'ютерна інженерія». 2.2. Завдання на кваліфікаційну роботу магістра затверджено наказом ректора No 0210-05/1804 від 10.09.2022.
3. Призначення розробки	3.1. Мета розробки програмного виробу: збільшення ефективності проведення досліджень, наукових напрацювань в наукових біологічних та хімічних лабораторіях за допомогою інтегрування технології Інтернету речей, «розумного дому» та промислового Інтернету речей. 3.2. Призначення виробу: автоматизація процесу контролю, прийняття рішень, забезпечення комфорту та безпеки в наукових лабораторіях різного типу.

4. Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: швидке реагування на зовнішні фактори, автоматизація процесів, віддалене керування, масштабуємість. 4.2. Вимоги до надійності: зменшити ризик злому системи, захистити акаунти адміністраторів, фільтрувати помилки. 4.3. Вимоги до умов експлуатації: обладнання Cisco в лабораторіях; інтуїтивно зрозуміле керування; 4.4. Вимоги до складу і параметрів технічних засобів: доступ до інтернету пристроїв віддаленого керування; наявність комп'ютера з 32 або 64-розрядним процесором з тактовою частотою більше 1 ГГц; операційна система Windows, Linux, MacOs; 4.5. Вимоги до інформаційної та програмної сумісності: сумісність з різними операційними системами Windows, Linux, MacOs. 4.6. Вимоги до маркування та упаковки: відсутні. 4.7. Вимоги до транспортування і зберігання: відсутні. 4.8. Спеціальні вимоги: відсутні.
5. Вимоги до програмної документації.	Програмною документацією до виробу «Програмна модель «розумної» науково-дослідницької лабораторії» вважати: 1) Справжнє Технічне завдання на розробку програмного вироби (представити у вигляді Додатку А до пояснювальної записки до кваліфікаційної роботи). 2) Програму і методику випробувань розробленого програмного вироби (представити у вигляді Додатку В до пояснювальної записки до кваліфікаційної роботи). 3) Опис програмного вироби (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи). 4) Текст програми (представити в Додатку Г до

6. Техніко-економічні показники	1) Можливі витрати грошових коштів на розробку програмного виробу: не потрібні. 2) Порівняння з існуючими зарубіжними та вітчизняними аналогами представити в розділі 2	
7. Стадії і етапи розробки	Дата етапу	Назва етапу
	19.10.2021 - 09.01.2022	1. Аналіз і підбір літератури та створення літературної бази для розробки моделі
	10.01.2022 - 01.02.2022	2. Аналіз сучасних прикладів індустріального Інтернету речей, розумного дому
	02.02.2022 - 30.03.2022	3. Розробка програмної моделі «розумної» науково-дослідницької лабораторії
	01.04.2022 - 01.05.2022	4. Розробка та тестування власних «розумних» пристроїв для науково-дослідницької лабораторії
	01.05.2022 - 25.05.2022	5. Розробка технічного завдання на розроблену науково-дослідницьку лабораторію
	26.05.2022 - 20.06.2022	6. Розробка програми і методики тестування розробленої моделі
	21.06.2022 - 15.07.2022	7. Розробка інструкцій для користувача
	16.07.2022 - 01.09.2022	8. Створення пояснювальної записки дипломної роботи
	01.09.2022 - 30.09.2022	9. Оформлення звіту за результатами науково-дослідницької практики
	01.10.2022 - 15.10.2022	10. Оформлення звіту за результатами переддипломної практики

	16.10.2022 - 01.11.2022	11. Підготовка доповіді на тему дипломної роботи на науково-технічну конференцію
	02.11.2022 - 14.11.2022	12. Підготовка до попереднього захисту роботи
	15.11.2022 - 23.11.2022	13. Представлення кваліфікаційної роботи науковому керівнику
	24.11.2022 - 30.11.2022	14. Представлення кваліфікаційної роботи науковому рецензенту
8.Порядок контролю і приймання	Загальні вимоги до приймання розробленого виробу: 1) Перевірка ходу розробки моделі керівнику дипломної роботи виконувати 1 раз в 3 тижні. 2) Випробування виробу відповідно до Програми і методики випробувань провести на різних типів лабораторій. 3) Захист розробленого програмного вироби провести на засіданні атестаційної комісії. 4) Пояснювальну записку уявити на паперових носіях в одному примірнику, в електронному вигляді	

Виконавець:
студент групи КІ-61
Волинський В.В.

Замовник:
д.т.н., професор кафедри теоретичної та прикладної системотехніки.
Мірошник М.А.

Додаток В

Затверджую

_____

«____»_____2022р.

Програма і методика випробувань

програмного виробу

«Модель «розумної» науково-дослідницької лабораторії»

1 Об'єкт випробувань

1.1 Найменування випробуваного програмного виробу: модель «розумної» науково-дослідницької лабораторії.

1.2 Область його застосування: процеси проведення наукових дослідів та напрацювань в різних типах лабораторій

2. Мета випробувань

2.1 Перевірка працездатності програмної моделі та її функціоналу в залежності від впливів зовнішніх факторів.

2.2 Підтвердження характеристик розробленого виробу вимогам, які сформульовані в ТЗ (Додаток Б).

3. Загальні положення

3.1 Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії та перевірку даного виробу.

3.2 Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі біологічної та хімічної лабораторій.

3.3 Обсяг випробувань

Приймальні випробування програмного виробу проводяться в обсязі

відповідному цієї програми і методики випробувань.

3.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю замовника, виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна реагувати на змінні навколишнього середовища, створені пристрої повинні ефективно виконувати запрограмовані дії, автоматично приймати рішення в залежності від створених сценаріїв роботи для покращення процес проведення дослідів.

5. Вимоги до програмної документації

Склад програмної документації, що подається на випробування, включає:

1. Технічне завдання на розробку моделі (наведене в Додатку А пояснювальної записка до дипломної роботи).
2. Програму і методику випробувань розробленого програмного виробу (наведено в Додатку В пояснювальної записка до дипломної роботи)
3. Опис виробу (представлено в Розділі 2 пояснювальної записка до дипломної роботи).
4. лістинги налаштування приладів на взаємодію (наведено в Додатку Г пояснювальної записка до дипломної роботи).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Необхідна присутність персонального комп'ютера з операційною системою не нижче Windows 8/10/11, Linux або MacOS. Також наявність додатку Cisco Packet Tracer версії не нижче 7.2.1 для створення працюючої мережі.

6.2 Порядок проведення випробувань

1. Перевірка програмної документації

1) Перевірка відповідності програмної документації за критерієм наявності зазначеної в ТЗ документації.

2)Перевірка якості програмної документації виконується на відповідність вимогам стандартів Єдиної Системи Проектної Документації (ЄСПД).

2. Перевірка працездатності моделі:

Тест 1: Запустити файл в формату .pkt за допомогою додатку Cisco Packet Tracer.

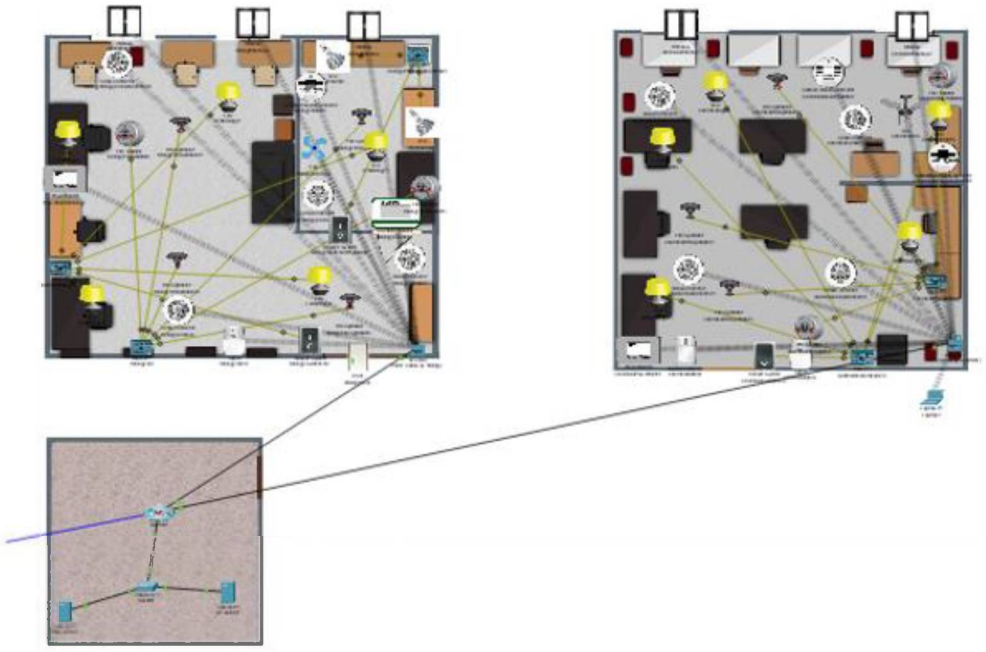


Рисунок 2.1 – Модель «розумної» науково-дослідницької лабораторії.

Якщо на екрані з'явилася модель з логічною топологією, то тест 1 пройдений.

3. Власне випробування програмного виробу.

Тест 1: Протестуємо роботу системи осушення повітря в біологічній лабораторії.

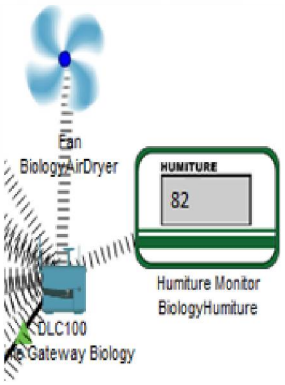


Рисунок 3.1 – Тестування системи осушення повітря біологічної лабораторії.

Якщо Humidity Monitor показав значення вище 75, то повинен увімкнутися пристрій осушення. Якщо це відбулося, тест можна вважати успішним

Тест 2: протестуємо систему доступу до реагентів тільки для експертів. Для цього використаємо стандартний пристрій - RFID Card, який генерує ключ доступу.

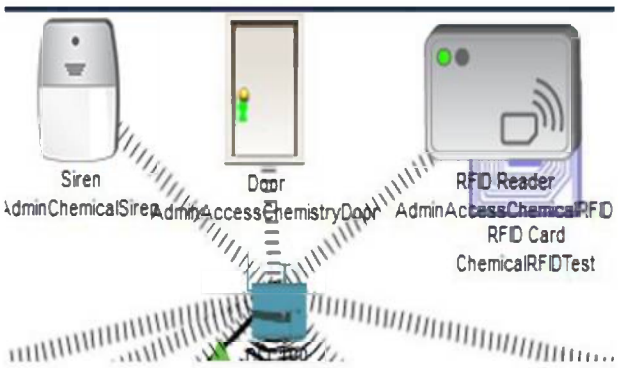


Рисунок 3.2 – Тестування системи фіксації задимленості.

Якщо RFID загорівся зеленим, то ключ валідний. Тоді мають відкритися двері сейфу і загорітися зеленим.

Тест 3: проведемо тестування віддаленого доступу до пристроїв лабораторії. Заходимо по адресу www.magestryiot.com. Результат показано на рисунку 3.3.

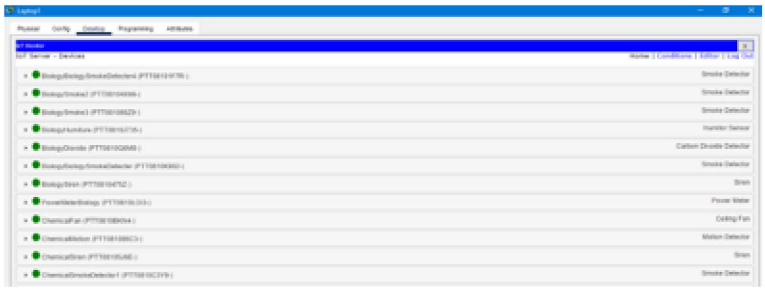


Рисунок 3.3 - Серверна частина системи управління пристроями.

Якщо на екрані з'являються розумні пристрої, то URI працює і ми можемо моніторити статус пристроїв і налаштовувати принципи прийняття рішень. Тоді тест можна вважати виконаним.

Випробування моделі можна вважаються успішними, якщо виконалися тести 1, 2 та 3.

Виконавець:

Студент групи КІ-61

Волинський В.В.

Додаток Г

Предоставлення коду для моделі «розумної» науково-дослідницької лабораторії

Лістинг Г.1 – Налаштування детектора вуглекислого газу

```
from My_env_variables import *
class DioxideDetector:
        globalSystemState = 0
        @classmethod
        def setEnv(cls):
                if  cls.globalSystemState == 1 :
                        volume = 100000 / Environment.getVolume()
                        Environment.setContrib(«Dioxide», dioxide*volume,
maximumRate, True)
                else:
                        Environment.setContrib(«Dioxide», 0, 0, True)
        @classmethodso
        def setupDioxide (csl):
                cls.globalSystemState = recheckProp(«state», 0);
                cls.__setState(cls.globalSystemState)
        @classmethod
        def clearState(cls,props, initialState):
                checker = getProp(getName(), props)
                if  not (checker is None):
                        if isinstance(initialState, int):
                                checker = int(checker)
                        setProp(getName(), props, checker)
                        return checker
                return initialState
```

```
@classmethod
def onClick(cls,isPressed):
        if isPressed:
            setState(0 if cls.globalSystemState else 1)
    @classmethod
    def __setState (cls,isStateChanged):
        if  isStateChanged == 0 :
            digitalWrite(1, LOW)
        else:
            digitalWrite(1, HIGH)
        cls.globalSystemState = isStateChanged
        setProp(getName(), «state», cls.globalSystemState)
        setupEnv()
if __name__ == «__main__»:
  DioxideDetector.setup()
  while True:
    delay(4000)
```

Лістинг Г.2 – Налаштування пристрою осушення повітря

```
import {valumeRate,globalState,dryFast,dryMiddle} from Environment
const setup = () => {
    IoT.setup({
        type: «Air dryer»,
        ]
    });
    IoT.dispatch =incomeData=> dataChecker(incomeData, true);
    attrInterrupt(0, () => dataChecker(customRead(0), false);
    dryerState = recheckProp(«state», 0);
    static setState(dryerState);
}
```

```
const recheckProp= (incomeProps, initialData)=>{
        let getProp = getGlobalProp(getName(), incomeProps);
        if ( typeof(initialData) == «number» ) {
                getProp = Number(getProp);
        }
        else{
                setDeviceProperty(getName(), incomeProps, getProp);
                return getProp;
        }
        return initialData;
}
const  onClick= (pressed, x, y, firstPress)=> if (firstPress) toggleState();
const dataChecker= (incomeData, isRemote=false)=>{
        if ( incomeData.length <= 0 ) return;
        static setState(parseInt(incomeData.split(«,»)[0]));
}
const sendReport =()=>{
        myWriter(0, globalState);
        IOT.logState(globalState);
        setProp(getName(), «state», globalState);
}
```

```
  const setState(newIncomeState) =>{
        analogWrite(A1, newIncomeState);
        senderReporter();
        updateEnv();
  }
  const toggleState=checkerSate=>{
        if ( checkerSate >= 3 )


globalState = 0;
      setState(globalState);
}
const updateEnv=()=>{
      let volume = +valumeRate / +Environment.getVolume();
      if ( globalState === 0){
            Environment.setContrib(«Dry speed», 0, 0);
      }
      else if ( globalState == 1 ){
            Environment.setContribution(«Dry speed»«, dryMiddle,
dryMiddle,false);
      }
      else if ( globalState == 2) Environment.setContrib(«Dry speed»«, dryFast,
dryFast, false);
}
```

Лістинг Г.3 – Пристрій ручної активація роботи фіголамп

```
from gpio import *
from time import *
pinMode(0,INPUT)
pinMode(1,OUTPUT)
pinMode(2,OUTPUT)
while True:
        if digitalRead(0)==HIGH:
                digitalWrite(1,HIGH)
                digitalWrite(2,HIGH)
        else:
                digitalWrite(1,LOW)
                digitalWrite(2,LOW)
```

Лістинг Г.4 – Пристрій імітації вогню

```
class Fire{
        const setup=()=>setProps(getName(), 'IR', 900);
}
fire= new Fire()
fire.setup()
```

Лістинг Г.5 – Приклад задання системних констант

```
const  monoxide = 1/3600;
const maximumRate = 100000;
export monoxide,maximumRate;
```

Лістинг Г.6 – Приклад налаштування DHCP

```
Router(config)#ip dhcp excluded–address 179.145.222.225  179.145.222.229
Router(config)#ip dhcp pool Chemical
Router(dhcp–config)#default–router 179.145.221.225
Router(dhcp–config)#network 179.145.221.225
Router(dhcp–config)#dns–server 10.0.0.234
Router(dhcp–config)#exit
```

Лістинг Г.7 – Код фітолампи

```
import {maxlight,maxRate,type,lastSeconds,timeDelay} form MyEnvVariables;
const setup = () => {
        OpacitySetting(type.toString(), 1);
        setInterrupt(0, I_S_R);
        I_S_R();
}

const isr = () => {
        inputDataSignal = analogRead(0);


       valueEnteredOn = map(inputDataSignal, 0, 1023, 0, 1);

       OpacitySetting(type.toString(), 1–valueEnteredOn);
       setProp(getName(), «level»,input);
}
const loop = delaying=> {
       setEnv();
       delay(delaying);
}
const setEnv= () =>{
       Environment.setContribution(«Visible Light», countedRate, countedRate,
false);
}
```