Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

КВАЛІФІКАЦІЙНА РОБОТА

за освітнім ступенем «бакалавр»

на тему:

«Моніторинг і управління системою «Розумний дім» на базі WebIOPi»

Виконав:
студент IV-го курсу
групи ПМ-41
спеціальності 113 «Прикладна математика»
Кравчук Іван Васильович
Керівник:
к.т.н., доцент Шинкарчук Н.В.

Рівне – 2025

АНОТАЦІЯ ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ

Кравчук І.В. Моніторинг і управління системою «Розумний дім» на базі WebIOPi. Кваліфікаційна робота на здобуття ступеня «бакалавр» за спеціальністю 113 «Прикладна математика» – Рівненський державний гуманітарний університет. Рівне, 2025. 71 с.

Кваліфікаційна робота присвячена розробці вебінтерфейсу для системи «Розумний дім» з використанням WebIOPi як основного інструменту на базі одноплатного комп'ютера Raspberry Pi 3 Model B. Основна увага приділена можливостям WebIOPi для створення інтерактивних вебінтерфейсів, які забезпечують віддалене керування та моніторинг пристроїв у локальній мережі. У роботі розглянуто теоретичні основи систем «Розумний дім», включаючи їхню історію, компоненти, принципи роботи, стандарти зв'язку, безпеку та енергоефективність. Досліджено архітектуру Raspberry Pi 3 Model B, його апаратні та програмні засоби, а також процес проектування вебінтерфейсу для керування компонентами, такими як датчики температури та вологості (DHT11), RGB LED (KY-016), RFID-читач (MFRC522), датчик звуку (LM393) та пасивний зумер.

Ключові слова: розумний дім, WebIOPi, Raspberry Pi, домашня автоматизація, вебінтерфейс, GPIO, датчики.

ЗМІСТ

ВСТУП **5**

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТА ПРИНЦИПИ ФУНКЦІОНУВАННЯ СИСТЕМИ «РОЗУМНИЙ ДІМ» **7**

1.1. Визначення та історія системи «Розумний дім» **7**

1.2. Основні компоненти системи «Розумний дім» **9**

1.3 Принципи роботи та функціональні можливості **11**

1.4. Технологічні стандарти та протоколи зв’язку **13**

1.5. Безпека та захист даних у системах «Розумний дім» **15**

1.6. Енергоефективність системи «Розумний дім» **17**

РОЗДІЛ 2. ВИКОРИСТАННЯ WEBIOPI ДЛЯ РОЗРОБКИ ВЕБІНТЕРФЕЙСІВ І УПРАВЛІННЯ СИСТЕМОЮ «РОЗУМНИЙ ДІМ» **19**

2.1. WebIOPi: суть технології та її місце в системах «Розумний дім» **19**

2.2. Обґрунтування вибору WebIOPi для реалізації вебінтерфейсів у системах домашньої автоматизації **20**

2.3. Технічні можливості WebIOPi для створення вебінтерфейсів **21**

2.4. Принципи розробки вебінтерфейсів із використанням WebIOPi **22**

2.5. Інтеграція WebIOPi з компонентами системи «Розумний дім» **25**

2.6. Безпека використання WebIOPi у системах «Розумний дім» **27**

2.7. Переваги та недоліки WebIOPi **30**

РОЗДІЛ 3. АРХІТЕКТУРА І ПРОГРАМНІ ЗАСОБИ ОДНОПЛАТНОГО КОМП’ЮТЕРА RASPBERRY PI 3 MODEL B **33**

3.1. Одноплатний комп’ютер Raspberry Pi 3 Model B **33**

3.2. Архітектура Raspberry Pi 3 Model B **34**

3.3. Операційні системи для Raspberry Pi **36**

3.4. Робота з GPIO на Raspberry Pi **37**

3.5. Інтеграція з WebIOPi **39**

РОЗДІЛ 4. ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБІНТЕРФЕЙСУ МОНІТОРИНГУ І УПРАВЛІННЯ СИСТЕМОЮ «РОЗУМНИЙ ДІМ» НА БАЗІ WEBIOPI **41**

4.1. Загальний опис системи «Розумний дім» на базі WebIOPi **41**

4.2. Технологічна основа системи «Розумний дім» на базі WebIOPi **44**

4.3. Моніторинг температури та вологості **46**

4.4. Керування LED **47**

4.5. Система доступу на основі RFID **49**

4.6. Виявлення звуку **51**

4.7. Відтворення звукових сигналів **52**

ВИСНОВКИ **55**

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **56**

ДОДАТОК А **59**

ДОДАТОК Б **61**

ДОДАТОК В **65**

ВСТУП

Актуальність роботи. Швидкий розвиток Інтернету речей (IoT) та зростаючий попит на автоматизацію побутових процесів роблять розробку систем «Розумний дім» надзвичайно актуальною. Сучасні комерційні рішення для «Розумного дому» часто є пропрієтарними, дорогими та обмеженими у можливостях кастомізації, що створює потребу в доступних альтернативах. Використання відкритих інструментів, таких як WebIOPi, дозволяє створювати економічно вигідні та гнучкі системи, які забезпечують віддалене керування та моніторинг пристроїв через вебінтерфейс.

Робота з WebIOPi сприяє розвитку навичок програмування на Python, створення вебінтерфейсів за допомогою HTML, CSS, JavaScript та управління GPIO.

Система, розроблена в даній кваліфікаційній роботі, включає компоненти, які інтегруються через WebIOPi для створення інтерактивного вебінтерфейсу. Цей інтерфейс дозволяє користувачам віддалено керувати пристроями та моніторити їхній стан у локальній мережі, що робить систему зручною та адаптивною. Проєкт демонструє, як WebIOPi забезпечує просту інтеграцію апаратного забезпечення з вебтехнологіями.

Мета роботи: розробити вебінтерфейс моніторингу та управління системою «Розумний дім» на базі WebIOPi.

Об’єктом дослідження є системи домашньої автоматизації.

Інструмент дослідження. Фреймворк WebIOPi для створення вебінтерфейсів, що забезпечує інтеграцію апаратного забезпечення Raspberry Pi з інтерактивними вебдодатками для віддаленого керування та моніторингу «Розумного дому».

Предмет дослідження. Створення та інтеграція вебінтерфейсів для управління та моніторингу пристроїв «Розумного дому» з використанням WebIOPi.

Завдання дослідження:

1. Провести аналіз принципів функціонування систем «Розумний дім» та їх компонентів.
2. Дослідити можливості WebIOPi для створення вебінтерфейсів.
3. Описати архітектуру та програмні засоби Raspberry Pi 3 Model B.
4. Розробити вебінтерфейс для моніторингу та керування пристроями «Розумного дому».

Апробація кваліфікаційної роботи. Результати виконання кваліфікаційної роботи, окремі її аспекти та одержані узагальнення і висновки були оприлюднені на XVIII Всеукраїнській науково-практичній конференції здобувачів вищої освіти та молодих учених «Наука, освіта, суспільство очима молодих» (м. Рівне, 2025), звітній науковій конференції викладачів, співробітників і здобувачів вищої освіти Рівненського державного гуманітарного університету за 2024 рік (м. Рівне, 2025).

Публікації. Результати, які були отримані в ході кваліфікаційного дослідження частково опубліковані у вигляді тез XVIII Всеукраїнської науково-практичної конференції здобувачів вищої освіти та молодих учених «Наука, освіта, суспільство очима молодих» у Рівненському державному гуманітарному університеті (Додаток А).

Структура роботи. Робота складається зі вступу, чотирьох розділів, висновків та списку використаних джерел. Перший розділ присвячений аналізу теоретичних основ систем «Розумний дім», включаючи їхню історію, принципи функціонування, компоненти, стандарти зв’язку, безпеку та енергоефективність. У другому розділі описано можливості WebIOPi для розробки вебінтерфейсів, принципи інтеграції з апаратним забезпеченням, заходи безпеки, переваги й недоліки фреймворку. У третьому розділі розглянуто архітектуру та програмні засоби Raspberry Pi 3 Model B. В четвертому розділі подано процес проектування та розробки вебінтерфейсу для моніторингу та управління пристроями «Розумного дому». Список літератури містить тридцять два джерела.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ТА ПРИНЦИПИ ФУНКЦІОНУВАННЯ СИСТЕМИ «РОЗУМНИЙ ДІМ»

1.1. Визначення та історія системи «Розумний дім»

Розумний дім (розумний дім, smart home) – система об’єднаних пристроїв, здатних виконувати дії та вирішувати повсякденні завдання без участі людини [1]. Основною метою такої системи є підвищення комфорту й ефективності життя завдяки використанню різноманітних пристроїв – датчиків температури, освітлення, розумних розеток тощо. Їхня координація забезпечує виконання численних комплексних завдань, які часто відбуваються автоматично. Наприклад, система може вимикати опалення в порожній кімнаті на основі даних із датчиків руху чи звуку. Підключення до мережі значно розширює можливості, дозволяючи дистанційно керувати системою через смартфон, планшет або комп’ютер із будь-якої точки світу. Усі можливості системи «Розумний дім» спрямовані на створення зручного, безпечного та енергоефективного житлового середовища, адаптованого до потреб користувача.

Історія розумного дому сягає кінця XIX століття, коли почалися перші спроби автоматизації. У 1898 році Нікола Тесла продемонстрував радіокерований пристрій, який став прообразом технологій дистанційного керування. Ці ідеї, хоча й не застосовувалися в побуті, заклали фундамент для майбутніх розробок.

Ідеї більш розвинені до понять сучасних систем автоматизації будинку були продемонстровані на ярмарках у Чикаго (1934) та Нью-Йорку. У «великому яблуці» трохи пізніше (1964–1965), представили плани електрифікованих та автоматизованих приміщень. У решті-решт перший серйозний аналог розумного дому з'явився у 1966 році. Це була експериментальна система домашньої автоматизації – «домашній комп'ютер Echo IV». Його винахідник – Джим Сазерленд, інженер компанії Westinghouse Electric. Його технологія була приватним, некомерційним проектом [2].

У 1975 році з’явилася система X10 – технологія, що дозволяла керувати освітленням і простими приладами через електропроводку. Вона була революційною, але мала проблеми з надійністю, адже сигнали часто губилися. У 1984 році Американська асоціація будівельників офіційно ввела термін «розумний дім», описуючи будівлі, оснащені мікроконтролерами для керування світлом, опаленням чи охоронними системами. Однак через високу ціну такі системи встановлювали переважно в офісах чи будинках заможних людей [1].

У 1990-х роках розвиток електроніки та комп’ютерів прискорив прогрес. Інженери почали створювати більш складні системи, які могли об’єднувати кілька функцій, наприклад, керування освітленням і безпекою одночасно. На початку 2000-х з’явилися бездротові протоколи зв’язку, такі як Zigbee та Z-Wave. Вони дозволили створювати мережі з десятків пристроїв, які працювали з мінімальним енергоспоживанням і не залежали від проводів. Це зробило розумні будинки дешевшими та зручнішими для встановлення. Наприклад, датчики руху чи температури могли спілкуватися з центральним контролером, створюючи прості автоматизовані сценарії.

2010-ті роки стали переломними завдяки масовому поширенню смартфонів і технологій Інтернету речей. У 2011 році компанія Nest Labs випустила розумний термостат Nest, який не лише регулював температуру, а й аналізував звички користувачів, оптимізуючи енергоспоживання. Цей пристрій став символом нового покоління розумних будинків, доступних звичайним людям. У 2014 році Amazon представила голосовий помічник Alexa, а в 2016 році з’явився Google Assistant. Вони дозволили керувати розумними пристроями простими голосовими командами, наприклад, вмикати світло чи змінювати температуру. У цей період розумні доми почали інтегруватися з хмарними сервісами, що забезпечило віддалений доступ через інтернет і можливість зберігати дані.

Сьогодні розумні будинки стають дедалі розумнішими завдяки штучному інтелекту. Сучасні системи можуть передбачати потреби користувачів, наприклад, автоматично налаштовувати освітлення для вечірнього відпочинку чи вмикати опалення перед поверненням додому. Крім того, розумні доми підтримують

екологічні ініціативи: вони інтегруються з сонячними панелями та оптимізують енергоспоживання, допомагаючи зменшити вуглецевий слід. За оцінками експертів, до 2030 року розумні технології стануть стандартом для нових будинків у багатьох країнах, адже їхня ціна знижується, а можливості зростають.

1.2. Основні компоненти системи «Розумний дім»

Датчики та виконавчі пристрої є основою функціональності розумного дому, адже вони забезпечують збір даних і виконання команд.

Датчики (сенсори) збирають інформацію про стан навколишнього середовища або будинку, дозволяючи системі реагувати на зміни. Найпоширеніші типи датчиків включають:

- *датчики руху*. Виявляють присутність людей у приміщенні, активуючи освітлення чи системи безпеки. Наприклад, датчик може увімкнути світло, коли людина заходить у кімнату;
- *датчики температури та вологості*. Контролюють клімат у будинку, передаючи дані до термостатів чи кондиціонерів для регулювання температури;
- *датчики диму та витоку газу*. Забезпечують безпеку, попереджаючи про небезпечні ситуації, такі як пожежа чи витік газу, і активуючи оповіщення чи автоматичні дії, наприклад, перекриття газопостачання;
- *датчики освітлення*. Вимірюють рівень природного світла, дозволяючи системі регулювати яскравість ламп або відкривати жалюзі.

Датчики працюють як «очі» системи, надаючи дані для аналізу та прийняття рішень. Вони зазвичай компактні, енергоефективні та можуть працювати від батарей протягом кількох років [3].

Виконавчі пристрої (реле, актуатори) реагують на команди системи, виконуючи фізичні дії. До них належать:

- *реле.* Керують увімкненням і вимкненням електроприладів, наприклад, світла чи побутової техніки. Розумні реле дозволяють дистанційно вмикати кавоварку чи вимикати праску;
- *актуатори.* Виконують механічні дії, такі як відкривання дверей, регулювання жалюзі чи перекриття водопостачання при виявленні протікання;
- *розумні розетки.* Контролюють подачу електроенергії до приладів, дозволяючи дистанційно вимикати їх для економії енергії або безпеки.

Датчики та виконавчі пристрої працюють у тандемі: датчики надають інформацію, а виконавчі пристрої реалізують відповідні дії, створюючи автоматизоване середовище.

Центральний контролер, або хаб, є «мозком» системи «Розумний дім», координуючи роботу всіх компонентів. Він отримує дані від датчиків, обробляє їх і надсилає команди до виконавчих пристроїв, забезпечуючи автоматизацію та віддалене керування.

Функції хаба:

1. *Координація пристроїв.* Хаб об’єднує пристрої, які використовують різні протоколи (Wi-Fi, Zigbee, Z-Wave), дозволяючи їм працювати разом. Наприклад, Samsung SmartThings може підключати Zigbee-датчики та Wi-Fi-камери до однієї системи.
2. *Обробка даних.* Хаб аналізує інформацію від датчиків і виконує запрограмовані сценарії. Наприклад, якщо датчик руху виявляє активність уночі, хаб може увімкнути світло та надіслати сповіщення на смартфон.
3. *Віддалений доступ.* Хаб підключається до інтернету, дозволяючи користувачам керувати системою через мобільні додатки чи вебінтерфейси з будь-якої точки світу.
4. *Інтеграція з голосовими помічниками.* Багато хабів, таких як Amazon Echo, підтримують Alexa чи Google Assistant, що дозволяє керувати системою голосом.

Хаб відіграє ключову роль у забезпеченні сумісності та автоматизації. Без нього пристрої з різними протоколами не могли б взаємодіяти, а система втратила б здатність до складних сценаріїв [4].

1.3. Принципи роботи та функціональні можливості

Система «Розумний дім» – це об'єднані пристрої, які автоматизують повсякденні завдання, роблять життя комфортнішим і безпечнішим, а також допомагають економити ресурси. Її робота базується на трьох принципах: автоматизації процесів, віддаленому керуванні та інтеграції з Інтернетом речей і хмарними сервісами.

Автоматизація дозволяє розумному дому брати на себе рутинні завдання, використовуючи датчики, виконавчі пристрої та хаб. Система сама аналізує дані й виконує потрібні дії. Одна з головних функцій будь-якого «розумного дому» – створення сценаріїв, коли натискання на одну кнопку дозволяє виконувати послідовність дій для тієї чи іншої ситуації. Ось як це виглядає:

- *освітлення*. Система налаштовує світло для різних ситуацій. Наприклад, режим «Вечір» приглушує лампи й вмикає підсвітку, а режим «Прибирання» робить світло максимально яскравим. Датчики руху вмикають світло, коли ви заходите в кімнату, і вимикають, коли її покидаєте;
- *клімат-контроль*. Розумний дім тримає температуру й вологість на комфортному рівні. Якщо датчик показує спеку, вмикається кондиціонер, а в холод – опалення. Наприклад, система може вимкнути опалення в порожніх кімнатах, щоб зекономити енергію;
- *безпека*. Датчики руху, диму чи витоку газу захищають будинок. Камери відеоспостереження, датчики руху і об'єму дозволяють відслідковувати появу непрошених гостей. Якщо система виявляє протікання, вона може перекрити воду, а при підозрі на чужинців – увімкнути сигналізацію [5].

Автоматизація економить час і зусилля. Наприклад, розумний дім може сам відкрити жалюзі вранці чи вимкнути всі розетки, коли ви йдете з дому.

Для наочності роботи системи «Розумний дім» та взаємодії її компонентів наведено схему (рис 1.1).

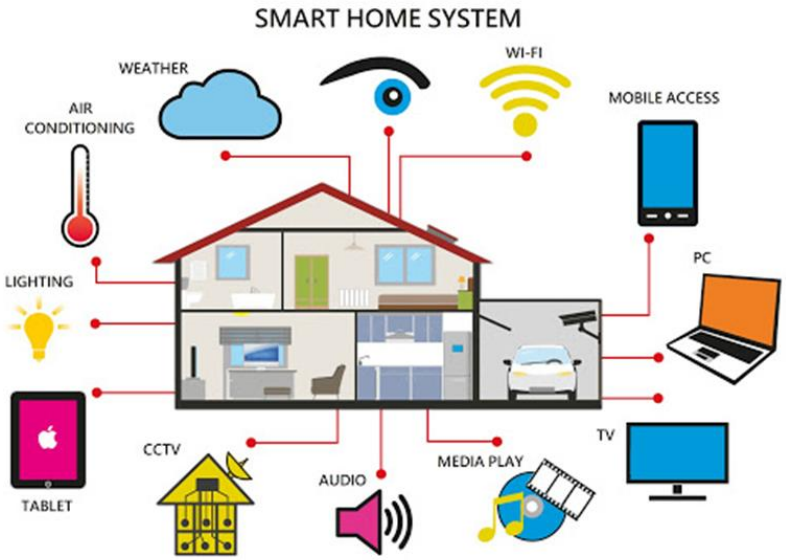

Рисунок 1.1. Схема роботи системи «Розумний дім»

Віддалене керування дає змогу контролювати розумний дім із будь-якої точки світу через смартфон чи комп’ютер, якщо є Інтернет.

- *Мобільні додатки*. Через додатки, як-от Philips Hue, можна вмикати світло, перевіряти камери чи змінювати температуру. Наприклад, ви можете увімкнути обігрівач, їдучи додому, щоб було тепло.
- *Вебінтерфейси*. Система дозволяє керувати будинком через браузер. Можна це робити, наприклад, через сторінку в Internet, виходячи на неї з будь-якого комп’ютера і використовуючи індивідуальний пароль [5]. Але такі сторінки потрібно добре захищати від хакерів.
- *Моніторинг*. Датчики й камери надсилають дані в реальному часі. Якщо датчик руху спрацьовує вночі, ви отримуєте сповіщення з відео з камери безпеки на телефон.

Віддалене керування додає зручності: ви завжди можете перевірити, чи вимкнули праску, чи зачинені двері. Але важливо, щоб система була захищена паролями й шифруванням.

Інтеграція з Інтернетом речей (IoT) дозволяє пристроям розумного дому «спілкуватися» між собою та з іншими сервісами через інтернет. Остаточно концепція «Інтернет речей» була сформована лише у 1999 р. Суть даної концепції полягає в тому, щоб за допомогою Інтернету, надати можливість взаємодії різних пристроїв, кожен з яких отримає свою власну унікальну IP-адресу [6]. У розумному домі це працює так:

- *взаємодія пристроїв*. Завдяки IoT датчик руху може сказати розумній лампі ввімкнути світло, а термостату – підняти температуру. Наприклад, коли ви заходите додому, система сама вмикає світло й музику;
- *хмарні сервіси*. Хмара зберігає дані, як-от відео з камер, і дає доступ до них із будь-якого місця. Хмарні сервіси також підтримують голосові помічники, як-от Alexa, які керують будинком через інтернет;
- *адаптація*. IoT і хмара допомагають системі вчитися. Наприклад, розумний дім може запам’ятати, коли ви приходите додому, і вмикати опалення заздалегідь.

Інтеграція з IoT робить систему гнучкою: ви можете додати нові пристрої, і вони легко впишуться в мережу.

1.4. Технологічні стандарти та протоколи зв’язку

Система «Розумний дім» залежить від технологічних стандартів і протоколів зв’язку, які забезпечують взаємодію між її компонентами, такими як датчики, хаб і пристрої керування. Ці протоколи визначають, як пристрої обмінюються даними, і впливають на швидкість, надійність і сумісність системи. У цьому підпункті розглянемо популярні протоколи MQTT, HTTP, CoAP, а також сучасні ініціативи, які покращують сумісність розумних домів.

Розумний дім використовує різні протоколи зв’язку, кожен із яких має свої особливості та призначення. Ось огляд трьох популярних протоколів:

- *MQTT (Message Queuing Telemetry Transport)*. Був розроблений компанією IBM у 1999 році, який є протоколом публікації/підписки. Він працює

поверх TCP/IP. Клієнт, що виступає в ролі видавця/підписника(наприклад, датчик) підключається до сервера, який є брокером повідомлень для отримання сповіщень. MQTT використовується для віддалених місць, де потрібна обмежена пропускна здатність мережі [7]. MQTT економить енергію і добре підходить для IoT-пристроїв, таких як розумні датчики чи термостати, адже він швидко передає невеликі повідомлення.

- *HTTP (HyperText Transfer Protocol).* Цей протокол знайомий усім через використання в інтернеті. У розумному домі HTTP дозволяє пристроям спілкуватися через вебінтерфейси. Наприклад, ви можете увімкнути світло через додаток на смартфоні, який надсилає HTTP-запит до хаба. Проте HTTP споживає більше ресурсів і не завжди ефективний для маленьких пристроїв, як-от датчики.
- *CoAP (Constrained Application Protocol).* Протокол, який дозволяє зв'язуватися з більш швидким Інтернетом, використовуючи подібні протоколи для обмежених пристроїв IoT, які називаються «вузлами». Він найкраще підходить для пристроїв, які знаходяться в одній або різних обмежених мережах. CoAP можна розглядати як заміну HTTP, оскільки він побудований на UDP а не TCP, що робить його швидшим і менш вимогливим до енергії [8].

Крім цих протоколів, у розумних домах використовують і інші стандарти зв’язку, які мають свою історію та застосування.

- *Bluetooth*. Один з найстаріших бездротових технологій, що використовується на всіх мобільних девайсах [9]. Він з’явився ще в 1990-х і став популярним для локального зв’язку на коротких відстанях. У розумному домі Bluetooth підключає пристрої, як-от смартфон і розумний замок, але його радіус дії обмежений – до 10–15 метрів, що робить його менш універсальним для великих будинків.
- *Z-Wave*. Даний протокол був розроблений компанією ZWave Alliance. Z-Wave створює сітчасту мережу, де пристрої передають сигнали один одному, що розширює покриття. Він використовується в розумних замках,

датчиках і лампах, адже споживає мало енергії й має менше перешкод, ніж Wi-Fi. Наприклад, Z-Wave-датчик може передати сигнал через інший пристрій, якщо хаб далеко [10].

Як подано у таблиці 1.1, кожен із цих протоколів має свої сильні й слабкі сторони, які впливають на їхнє застосування в розумному домі.

Таблиця 1.1. Порівняння протоколів зв'язку

Протокол	Переваги	Недоліки
MQTT	Легкий, економить енергію	Потребує брокера для обміну
HTTP	Універсальність, вебінтерфейси	Високе споживання ресурсів
CoAP	Швидкість, підходить для IoT	Менш поширений, ніж HTTP
Bluetooth	Простота, енергоефективність	Малий радіус дії
Z-Wave	Надійність, сітчаста мережа	Обмежена сумісність

Однією з проблем розумних домів є несумісність між пристроями, які використовують різні протоколи. Щоб вирішити це у 2019 році Apple, Amazon, Google і Zigbee Alliance об'єднали зусилля для створення проєкту Connected Home over IP (CHIP), який має на меті розробити єдиний стандарт для розумних домів . Цей стандарт базується на протоколі IP, що дозволяє пристроям від різних виробників працювати разом. Наприклад, розумна лампа від Philips Hue зможе легко підключитися до системи Google Home. CHIP також підтримує безпеку, використовуючи шифрування, і обіцяє спростити налаштування розумного дому для користувачів [11].

1.5. Безпека та захист даних у системах «Розумний дім»

IoT-пристрої, які є основою розумного дому (датчики, камери, розумні розетки), часто стають мішенню для кібератак через їхню простоту та підключення до інтернету. Загрози інформаційної безпеки IT-системи «розумний дім» в першу чергу залежать від обраних способів і технологій побудови даної системи, так як

на визначення можливих загроз впливає склад обладнання. Основні проблеми включають:

- *ненадійні паролі*. Багато IoT-пристроїв мають слабкі паролі за замовчуванням, як-от «admin» чи «1234», які легко зламати. Наприклад, хакер може отримати доступ до камери безпеки й стежити за будинком;
- *відсутність оновлень*. Деякі пристрої не отримують оновлення безпеки, що робить їх вразливими до нових атак. Наприклад, старий розумний термостат може мати уразливість, яку вже давно виправлено в нових моделях;
- *порушення цілісності даних*. Цілісність інформації – це достовірність і повнота інформації, що отримується системою від різних датчиків і пристроїв, встановлених в системі, наприклад, при отриманні невірної інформації системою про наявність в приміщенні людини може привести до помилкового спрацьовування системи контролю доступу [11]. Якщо хакер підробить дані датчика руху, система може відкрити двері, думаючи, що це господар;
- *централізовані системи*. Для оцінки ризиків інформаційної безпеки «розумного дому» розглянемо найбільш ймовірні загрози, реалізація яких може призвести до порушення інформаційної безпеки «розумного дому», побудованого з централізованої технології. Якщо хакер зламає хаб, він отримує контроль над усією системою, наприклад, може вимкнути сигналізацію чи ввімкнути всі прилади.

Ці вразливості можуть призвести до серйозних наслідків, як-от крадіжка даних, порушення приватності чи навіть фізична небезпека для мешканців.

Щоб захистити розумний дім від кібератак, використовують методи шифрування та автентифікації, які забезпечують безпеку даних і доступу. Ось основні підходи:

- *шифрування даних*. Шифрування захищає дані, які передаються між пристроями та хабом. Наприклад, коли ви вмикаєте світло через смартфон, команда шифрується за допомогою протоколу TLS (Transport Layer

Security), щоб ніхто не міг її перехопити. Це особливо важливо для бездротових протоколів. Бездротові протоколи дозволяють зручно реалізувати мережеву взаємодію, але вони вразливі, якщо не захищені шифруванням [12];

- *двофакторна автентифікація (2FA).* Для доступу до системи, наприклад, через мобільний додаток, можна налаштувати 2FA. Це означає, що крім пароля потрібно ввести код, який надсилається на ваш телефон. Так, навіть якщо хтось дізнається пароль, він не зможе увійти без другого фактора;
- *біометрична автентифікація*. Деякі розумні замки використовують відбитки пальців чи розпізнавання обличчя. Наприклад, двері відкриваються лише для господаря, що зменшує ризик несанкціонованого доступу;
- *сегментація мережі*. Щоб зменшити ризик централізованих атак, пристрої розумного дому можна підключити до окремої мережі, ізольованої від основної домашньої мережі. Якщо хакер зламає розумну розетку, він не зможе дістатися до вашого комп'ютера.

1.6. Енергоефективність системи «Розумний дім»

Автоматизація в розумному домі дозволяє пристроям самостійно приймати рішення про використання енергії, базуючись на даних від датчиків, виконавчих механізмів і програмованих сценаріїв. Впровадження технології розумного дому сприяє підвищенню енергоефективності, що призводить до зменшення споживання енергії, зниження витрат та покращення екологічної сталості [13]. Розглянемо основні аспекти, де автоматизація дає найбільший ефект.

- *освітлення*. Розумний дім контролює світло в кожній кімнаті, використовуючи датчики руху та природного світла. Наприклад, якщо ви вийшли з кухні, система одразу вимикає лампи, а якщо ввечері заходите в коридор, світло вмикається саме, але лише там, де ви є. Також автоматика регулює яскравість залежно від сонячного світла: у сонячний день лампи

стають тьмянішими, а в похмуру погоду – яскравішими. Це дозволяє економити до 15–20% електроенергії, адже світло не горить даремно в порожніх приміщеннях чи вдень, коли сонце дає достатньо освітлення;

- *клімат-контроль*. Розумні термостати й кондиціонери оптимізують температуру в будинку. Наприклад, коли ви йдете на роботу, система знижує опалення до 16–18°C, щоб не гріти порожній будинок, а за годину до вашого повернення поступово підвищує температуру до комфортних 22°C. Улітку кондиціонер вимикається, якщо вікна відкриті чи температура на вулиці знижується. Датчики вологості також допомагають увімкнути зволожувач лише тоді, коли повітря занадто сухе, що економить енергію й воду. Така автоматика може знизити споживання енергії на опалення чи охолодження на 20–30%;
- *розумні розетки*. Їх використання дозволяє автоматично вимикати невикористовуване обладнання, коли воно тривалий час не використовується. Це дає змогу зекономити до 10% енергії на таких пристроях [14];
- *електроприлади та побутова техніка*. Автоматизація дозволяє оптимізувати роботу пральних машин, холодильників і бойлерів. Наприклад, система може запускати прання чи нагрів води вночі, коли тарифи на електроенергію нижчі, або коли сонячні панелі накопичують достатньо енергії. Холодильник може регулювати температуру залежно від того, скільки продуктів у ньому, і вимикати компресор, коли це не потрібно, що економить до 5–10% електрики;
- *інтеграція з відновлюваними джерелами енергії*. Розумний дім може працювати з сонячними панелями чи вітрогенераторами. Система стежить за погодними умовами й заряджає батареї вдень, коли сонце активне, а ввечері використовує накопичену енергію для освітлення чи зарядки гаджетів. Якщо сонячної енергії бракує, система перемикається на мережу лише частково, економлячи до 40% витрат на електрику в сонячні дні.

РОЗДІЛ 2
ВИКОРИСТАННЯ WEBIOPI ДЛЯ РОЗРОБКИ ВЕБІНТЕРФЕЙСІВ І УПРАВЛІННЯ СИСТЕМОЮ «РОЗУМНИЙ ДІМ»

2.1. WebIOPi: суть технології та її місце в системах «Розумний дім»

WebIOPi – це програмний фреймворк, який створювався спеціально для роботи з одноплатними комп'ютерами Raspberry Pi, щоб забезпечити віддалене керування пристроями. Він дозволяє реалізувати принципи Інтернету речей (IoT), інтегруючи апаратне забезпечення з програмними рішеннями. Завдяки WebIOPi можна розробляти різноманітні користувацькі додатки, які забезпечують зручне управління системами «Розумний дім» через вебінтерфейси [15]. Цей інструмент став популярним серед розробників завдяки своїй простоті та орієнтованості на автоматизацію побутових процесів.

Основна суть WebIOPi полягає в тому, що він виступає посередником між апаратною частиною (наприклад, сенсорами чи реле, підключеними до Raspberry Pi) і користувачем, який може керувати цими пристроями через браузер. Фреймворк підтримує створення вебінтерфейсів, які дозволяють віддалено моніторити стан системи та управляти пристроями в режимі реального часу. Наприклад, WebIOPi дає змогу створювати вебінтерфейси для віддаленого моніторингу та керування пристроями в системах «Розумний дім», використовуючи Raspberry Pi як центральний елемент [16]. Це робить його цінним інструментом для невеликих проєктів домашньої автоматизації, де важливі простота та доступність.

У контексті систем «Розумний дім» WebIOPi займає особливе місце, адже він забезпечує легкий доступ до апаратного забезпечення через мережу, що дозволяє автоматизувати повсякденні задачі – від увімкнення світла до моніторингу температури. Його інтеграція з Raspberry Pi робить технологію економічно вигідною, адже Raspberry Pi – це недорогий і універсальний пристрій, який підтримує підключення різноманітних сенсорів і модулів. Таким чином, WebIOPi

допомагає створювати гнучкі та функціональні системи, які підвищують комфорт, безпеку та енергоефективність у домі.

2.2. Обґрунтування вибору WebIOPi для реалізації вебінтерфейсів у системах домашньої автоматизації

Вибір WebIOPi як основного інструменту для реалізації вебінтерфейсів у системах домашньої автоматизації був обґрунтований цілою низкою причин, які враховують як технічні можливості фреймворка, так і практичні аспекти його застосування в умовах обмежених ресурсів. Одним із ключових факторів стало те, що використання WebIOPi у поєднанні з Raspberry Pi забезпечує гнучкість і економічність для реалізації систем домашньої автоматизації через Інтернет речей (IoT) [16]. Ця комбінація забезпечує розробку ефективних і економічно доступних рішень для автоматизації, які не передбачають значних фінансових затрат на дороге обладнання чи складне програмне забезпечення. Raspberry Pi, виступаючи платформою для роботи з WebIOPi, було обрано як логічний і доцільний вибір для реалізації проєкту, з огляду на наявність цих пристроїв у розпорядженні навчального закладу. Це дало змогу уникнути додаткових витрат на закупівлю апаратного забезпечення та зосередитися безпосередньо на розробці та тестуванні системи, що значно спростило процес роботи над проєктом.

Ще однією важливою причиною стала висока адаптивність WebIOPi до різноманітних апаратних конфігурацій. Зокрема, фреймворк підтримує гнучкі конфігурації для інтеграції сенсорів, через GPIO-порти Raspberry Pi, дозволяючи відображати дані в реальному часі на вебінтерфейсі [15]. Ця особливість робить WebIOPi надзвичайно зручним для створення систем моніторингу та керування, де потрібно швидко налаштувати підключення до сенсорів і забезпечити їхню інтеграцію з мережею. Наприклад, завдяки цьому можна легко організувати відображення температури чи тиску в реальному часі на вебсторінці, що є важливим для систем «Розумний дім». Така гнучкість дозволяє адаптувати систему

під різні потреби, від простого увімкнення світла до складнішого аналізу даних із датчиків, що робить WebIOPi привабливим інструментом для розробників.

Окрім технічних переваг, варто відзначити й практичну сторону вибору цього фреймворка. WebIOPi орієнтований на роботу з Raspberry Pi, що забезпечує високу сумісність і стабільність у процесі експлуатації. До того ж, фреймворк пропонує широкий набір бібліотек, які спрощують створення вебінтерфейсів навіть для тих, хто не має глибоких знань у програмуванні. Це стало важливим фактором, адже у навчальному процесі часто доводиться шукати рішення, які дозволяють швидко приступити до реалізації ідей без тривалих підготовчих етапів. Завдяки наявності детальної документації та прикладам коду, WebIOPi дозволив мені ефективно налаштувати систему, використовуючи вже доступні ресурси університету.

2.3. Технічні можливості WebIOPi для створення вебінтерфейсів

WebIOPi вирізняється широким набором технічних можливостей, які роблять його потужним інструментом для розробки вебінтерфейсів у системах домашньої автоматизації. Однією з ключових переваг цього фреймворка є його здатність забезпечувати віддалене керування пристроями через вебінтерфейс, що дозволяє використовувати Raspberry Pi як своєрідний файловий сервер або навіть медіа-центр. Це означає, що користувач може отримати доступ до системи з будь-якого пристрою, підключеного до Інтернету, незалежно від того, де він перебуває – чи то вдома, чи в іншій точці світу. Така функціональність значно розширює можливості для створення систем «Розумний дім», адже дозволяє організувати зручний контроль над пристроями без фізичного доступу до апаратного забезпечення [17].

Окрім віддаленого доступу, WebIOPi забезпечує глибоку інтеграцію з апаратними компонентами через GPIO-порти Raspberry Pi. Це дає змогу підключати різноманітні пристрої – від простих світлодіодів до складніших модулів, таких як сенсори руху чи температури і керувати ними безпосередньо через вебінтерфейс. Наприклад, можна налаштувати систему так, щоб дані з

датчиків відображалися на вебсторінці в реальному часі, а користувач мав можливість увімкнути чи вимкнути певний пристрій одним натисканням кнопки у браузері. Така інтеграція дозволяє створювати не лише функціональні, а й інтуїтивно зрозумілі рішення, які легко адаптувати під конкретні потреби.

Ще однією важливою особливістю WebIOPi є його вбудована підтримка вебсервера, реалізованого на основі Python. Це забезпечує швидке розгортання системи без необхідності встановлення додаткових серверних компонентів, що економить час і ресурси під час розробки. До того ж, фреймворк підтримує створення динамічних вебінтерфейсів за допомогою бібліотеки JavaScript, що входить до його складу. Завдяки цьому розробник може створювати сторінки з інтерактивними елементами, такими як кнопки, графіки чи поля для введення даних, які дозволяють користувачу не лише спостерігати за станом системи, а й активно взаємодіяти з нею. Наприклад, можна реалізувати інтерфейс, де користувач задає певні параметри роботи системи, а WebIOPi автоматично виконує відповідні дії.

2.4. Принципи розробки вебінтерфейсів із використанням WebIOPi

Розробка вебінтерфейсів для систем «Розумний дім» за допомогою фреймворку WebIOPi є ключовим аспектом створення інтуїтивно зрозумілих і функціональних систем управління. WebIOPi, розроблений для роботи з одноплатними комп’ютерами Raspberry Pi, дозволяє реалізувати віддалене керування пристроями через веббраузер, забезпечуючи взаємодію між користувачем і апаратними компонентами. Цей процес є важливим для автоматизації побутових процесів, таких як регулювання освітлення, клімату чи безпеки, і вимагає дотримання певних принципів для забезпечення ефективності та зручності.

Розробка вебінтерфейсу з WebIOPi передбачає кілька послідовних етапів, які забезпечують логічну структуру та функціональність системи. Першим етапом є планування структури інтерфейсу, що включає визначення основних елементів і їх

розташування. На цьому етапі розробник аналізує потреби користувача та визначає, які функції будуть доступні через інтерфейс. Наприклад, для системи «Розумний дім» інтерфейс може включати кнопки для вмикання/вимикання світла, перемикачі для регулювання температури або графіки для відображення даних із датчиків (температура, вологість). Планування також охоплює створення макета сторінки, який визначає розміщення елементів управління для забезпечення зручності використання.

Другий етап – вибір елементів управління, що залежить від типу інформації та дій, які потрібно реалізувати. WebIOPi дозволяє створювати різноманітні елементи, такі як:

- кнопки для виконання миттєвих дій (наприклад, увімкнення лампи);
- перемикачі для зміни стану пристрою (увімкнено/вимкнено);
- слайдери для регулювання параметрів (яскравість світла, температура);
- графіки та діаграми для візуалізації даних у реальному часі (наприклад, зміна температури протягом дня).

Ці елементи мають бути інтуїтивно зрозумілими та відповідати принципам дизайну інтерфейсів. Створення інтерфейсів може бути логічним і простим процесом, але вимагає інтуїтивного розуміння того, чому певні шаблони поведінки є найкращими, коли ми використовуємо вебмови, що забезпечують плавну та легку взаємодію [18]. Третій етап – тестування та оптимізація. Після створення прототипу інтерфейсу проводяться тести для перевірки його функціональності, зручності та швидкості реакції. Наприклад, перевіряється, чи коректно відображаються дані з датчиків і чи швидко виконуються команди. Оптимізація може включати зменшення розміру графічних елементів або спрощення JavaScript-коду для підвищення швидкодії на мобільних пристроях.

Роль клієнт-серверної взаємодії. Клієнт-серверна взаємодія є основою функціонування вебінтерфейсів, створених за допомогою WebIOPi. Фреймворк використовує вбудований HTTP-сервер, який забезпечує зв’язок між вебінтерфейсом (клієнтом) і апаратними компонентами (сервером, що працює на Raspberry Pi). WebIOPi реалізує REST API, що дозволяє клієнту надсилати запити

до сервера для виконання дій або отримання даних. Наприклад, користувач натискає кнопку в браузері, що генерує запит до сервера, який, у свою чергу, активує GPIO-пін для увімкнення пристрою.

Офіційна документація WebIOPi пояснює цей процес: «WebIOPi включає HTTP-сервер, який надає як HTML-ресурси, так і REST API для керування пристроями. Браузер спочатку завантажує HTML-файл, після чого включений JavaScript виконує асинхронні виклики до REST API для керування та оновлення інтерфейсу. Цей метод є дуже ефективним, оскільки не потребує оновлення та завантаження всієї сторінки» [19]. Такий підхід забезпечує швидку реакцію системи та мінімізує навантаження на мережу, що особливо важливо для систем із обмеженими ресурсами, таких як Raspberry Pi.

Клієнт-серверна взаємодія також дозволяє реалізувати моніторинг у реальному часі. Наприклад, датчик температури надсилає дані через GPIO до сервера WebIOPi, який передає їх клієнту для відображення на вебсторінці. Асинхронні запити (AJAX) забезпечують оновлення лише потрібних елементів інтерфейсу, що підвищує зручність і ефективність.

Використання стандартних вебтехнологій. WebIOPi спирається на стандартні вебтехнології – HTML, CSS і JavaScript – для створення інтуїтивно зрозумілого дизайну. HTML формує структуру сторінки, визначаючи розташування елементів управління, таких як кнопки чи текстові поля. CSS забезпечує візуальний стиль, роблячи інтерфейс привабливим і зручним, наприклад, задаючи кольори, шрифти та відступи. JavaScript додає динамічність, дозволяючи оновлювати вміст сторінки без перезавантаження, що критично для систем у реальному часі.

Наприклад, JavaScript може періодично запитувати дані з датчиків через REST API WebIOPi і оновлювати графіки на сторінці, забезпечуючи користувачу актуальну інформацію про стан системи. Фронтенд-розробка включає створення візуальної частини вебсайту, з якою взаємодіють користувачі, використовуючи технології, такі як HTML, CSS і JavaScript.

Адаптивний дизайн дозволяє доступ до контенту через кілька роздільних здатностей пристроїв. Оскільки все більше людей взаємодіють з вебсайтами через

мобільні пристрої, користувачі тепер очікують, що вебсайти будуть адаптивними [20]. Ця особливість робить систему «Розумний дім» більш доступною і зручною для користувачів, незалежно від того, яким пристроєм вони користуються.

У таблиці 2.1. подано, як адаптивний дизайн враховує особливості взаємодії з різними пристроями.

Таблиця 2.1. Порівняння елементів вебінтерфейсу за типом пристрою

Елемент	ПК	Смартфон
Кнопки	Компактні, горизонтально	Більші, вертикально
Графіки	Деталізовані, повний екран	Спрощені, адаптивні
Навігація	Верхнє меню	Нижнє меню (доступ для пальця)

2.5. Інтеграція WebIOPi з компонентами системи «Розумний дім»

Інтеграція WebIOPi з компонентами системи «Розумний дім» є ключовим аспектом створення ефективних рішень для домашньої автоматизації. WebIOPi, як фреймворк для веббазованого керування апаратним забезпеченням, забезпечує гнучку взаємодію з різноманітними пристроями, дозволяючи підключати датчики, виконавчі механізми та інші компоненти, обробляти їхні сигнали та передавати дані через вебінтерфейс. У цьому підрозділі розглянуто основи взаємодії з апаратним забезпеченням, принципи підключення пристроїв, механізми передачі даних і можливості розширення системи, уникаючи повторів із попередніми розділами та зосереджуючись на унікальних технічних аспектах.

WebIOPi підтримує стандартні інтерфейси, такі як GPIO (General Purpose Input/Output), I2C (Inter-Integrated Circuit) і SPI (Serial Peripheral Interface), які є основою для взаємодії з апаратним забезпеченням у системах «Розумний дім» [21].

- *GPIO* дозволяє керувати цифровими сигналами, що необхідно для роботи з простими датчиками (наприклад, датчиками руху) та виконавчими пристроями (наприклад, реле для керування освітленням).

- *I2C* застосовується для зв’язку з пристроями, які потребують більш складної взаємодії, такими як датчики температури чи тиску.
- *SPI* використовується для швидкісного обміну даними з пристроями, такими як аналогово-цифрові перетворювачі або дисплеї, де потрібна висока пропускна здатність.

WebIOPi надає бібліотеки, які спрощують роботу з цими інтерфейсами, дозволяючи розробникам легко інтегрувати апаратне забезпечення в свою систему. Наприклад, за допомогою WebIOPi можна налаштувати скрипт, який зчитує дані з датчика температури через I2C і відображає їх на вебінтерфейсі [21]. Цей підхід забезпечує гнучкість і доступність для створення систем, де важлива взаємодія з різними типами пристроїв.

У системах «Розумний дім» датчики використовуються для моніторингу навколишнього середовища, а виконавчі пристрої – для автоматизації дій на основі отриманих даних. WebIOPi дозволяє інтегрувати ці компоненти, забезпечуючи їхню взаємодію через вебінтерфейс.

- *Датчики*. Підключаються через GPIO, I2C або SPI залежно від типу. Наприклад, датчик температури DS18B20 підключається через GPIO, тоді як датчик вологості DHT22 може використовувати I2C. WebIOPi дозволяє читати значення з цих датчиків і передавати їх до вебінтерфейсу для відображення, наприклад, показуючи поточну температуру на екрані смартфона.
- *Виконавчі пристрої*. Такі як реле, сервомотори чи мотори, керуються через GPIO. Наприклад, реле може бути використане для вмикання/вимикання світла, а сервомотор – для автоматизації жалюзі, регулюючи їхнє положення залежно від рівня освітленості.

Обробка сигналів від датчиків та передача команд до виконавчих пристроїв здійснюється через скрипти Python, які є частиною WebIOPi. Ці скрипти можуть бути налаштовані для автоматизації процесів, таких як увімкнення опалення при зниженні температури або вмикання світла при виявленні руху. Наприклад, за допомогою WebIOPi можна створити систему, яка автоматично регулює освітлення

залежно від рівня освітленості в кімнаті, використовуючи фотодатчик . Цей підхід дозволяє створювати адаптивні системи, які реагують на зміни в навколишньому середовищі.

WebIOPi використовує HTTP-сервер для обміну даними між клієнтом (веббраузером) та сервером (апаратним забезпеченням). Основним механізмом є REST API, який дозволяє надсилати запити для отримання даних з датчиків або для керування виконавчими пристроями.

Однією з переваг WebIOPi є його гнучкість щодо розширення системи. Додавання нових пристроїв, таких як датчики клімату, є простим завданням завдяки підтримці стандартних інтерфейсів.

Ця гнучкість дозволяє системі «Розумний дім» адаптуватися до змін у вимогах користувача чи навколишнього середовища, роблячи її більш універсальною та ефективною. Наприклад, за допомогою WebIOPi можна легко інтегрувати нові пристрої, такі як камери спостереження чи датчики безпеки, без значних змін у існуючій архітектурі системи [22].

2.6. Безпека використання WebIOPi у системах «Розумний дім»

Використання WebIOPi для управління системами «Розумний дім» забезпечує зручність і гнучкість, але водночас вимагає ретельного підходу до безпеки. Оскільки система дозволяє віддалене керування пристроями через вебінтерфейс, вона може бути вразливою до різних типів атак, якщо не забезпечити належний захист. У цьому розділі розглянуто ключові аспекти безпеки, які потрібно враховувати при впровадженні WebIOPi в системах «Розумний дім», уникаючи повторів із попередніми розділами, такими як технічні можливості чи інтеграція з апаратним забезпеченням. Текст написаний у формальному академічному тоні, з природними переходами між пунктами, щоб забезпечити цілісність, ніби його писала одна людина.

Основні аспекти безпеки.

1. *Захист вебсервера*. WebIOPi використовує вбудований вебсервер, який є основним інтерфейсом для взаємодії користувача з системою. Для захисту від несанкціонованого доступу необхідно налаштувати автентифікацію користувачів, використовувати сильні паролі та обмежити доступ до сервера лише з довірених IP-адрес. Наприклад, можна налаштувати файервол, щоб дозволяти доступ лише з локальної мережі або з конкретних IP-адрес, що зменшує ризик атак ззовні. Це особливо важливо, адже, як зазначається у дослідженні, багато IoT-пристроїв, включаючи ті, що використовуються в системах «Розумний дім», вразливі до атак типу ман-ін-те-мідл, коли зловмисники перехоплюють комунікацію між пристроями [23]. Такий підхід допомагає зменшити ймовірність несанкціонованого доступу до системи через вебінтерфейс.
2. *Шифрування даних.* Для захисту даних, що передаються між клієнтом (браузером) і сервером, необхідно використовувати HTTPS замість HTTP. HTTPS забезпечує шифрування трафіку, що запобігає перехопленню конфіденційної інформації, такої як паролі чи команди керування пристроями. Встановлення SSL/TLS-сертифікату на вебсервері WebIOPi є обов’язковим кроком для забезпечення безпеки зв’язку. Це особливо актуально, адже, як підкреслюють дослідники, смарт-домашні пристрої часто стають джерелом цифрових шкод, зокрема порушення конфіденційності, коли особиста інформація користувачів збирається без їхньої згоди [24]. Шифрування даних є ключовим для захисту від таких ризиків.
3. *Керування доступом.* Важливо налаштувати права доступу так, щоб лише авторизовані користувачі могли керувати системою. Це можна досягти за допомогою систем автентифікації, таких як локальні облікові записи або інтеграція з зовнішніми сервісами автентифікації (наприклад, OAuth). Крім того, можна впровадити багатостадійну автентифікацію (MFA), щоб додатково захистити доступ до системи.

4. *Оновлення програмного забезпечення.* Регулярне оновлення WebIOPi та інших компонентів системи є критичним для запобігання експлуатації відомих вразливостей. Важливо відстежувати випуски оновлень і встановлювати їх якомога швидше, оскільки багато атак на IoT-пристрої виникають через застаріле програмне забезпечення.
5. *Фізична безпека.* Оскільки WebIOPi працює на фізичному пристрої, необхідно забезпечити його фізичну безпеку. Це включає розміщення пристрою в недоступному місці, використання захисних корпусів та блокування фізичного доступу до порту USB чи інших інтерфейсів. Також варто розглянути використання безпечного живлення, щоб уникнути ризиків, пов’язаних із перериванням електропостачання.
6. *Моніторинг і аудит.* Регулярний моніторинг системи на наявність підозрілих активностей та ведення журналів подій можуть допомогти виявити і реагувати на спроби несанкціонованого доступу. Наприклад, можна налаштувати систему на відправку сповіщень у разі спроб входу з невідомих IP-адрес чи помічених аномалій у трафіку.

Крім специфічних заходів для WebIOPi, варто звернути увагу на загальні рекомендації щодо безпеки IoT-пристроїв, оскільки система «Розумний дім» на базі WebIOPi є частиною більшої IoT-екосистеми. До таких рекомендацій належать:

- використання сертифікованих пристроїв, які пройшли тестування на безпеку;
- регулярне оновлення фірмвару для всіх пристроїв, включаючи датчики, актуатори та хаби;
- забезпечення того, щоб усі пристрої були належним чином налаштовані і інтегровані в мережу, з мінімальними ризиками несумісності чи конфліктів;
- використання сегментації мережі, щоб ізолювати IoT-пристрої від основної мережі, зменшуючи ризик поширення атак.

Ці заходи допомагають створити багатошарову систему захисту, яка зменшує ймовірність успішних атак на систему «Розумний дім».

2.7. Переваги та недоліки WebIOPi

WebIOPi – це фреймворк, який дозволяє створювати вебінтерфейси для керування апаратним забезпеченням на базі Raspberry Pi, що робить його популярним серед ентузіастів IoT і початківців у сфері домашньої автоматизації. Проте, як і будь-яка технологія, він має свої сильні та слабкі сторони, які потрібно враховувати при виборі інструментів для розробки. Аналіз базується на доступній документації, досвіді користувачів і технічних характеристиках, з урахуванням того, що проект більше не активно розвивається, що впливає на його актуальність.

Переваги WebIOPi.

1. *Доступність і простота використання.* WebIOPi є вільним і відкритим програмним забезпеченням, що дозволяє будь-якому користувачеві з доступом до Raspberry Pi почати створювати вебінтерфейси для керування апаратним забезпеченням без додаткових витрат. Це робить його ідеальним для студентів чи невеликих проєктів, де бюджет обмежений [19].
2. *Підтримка різноманітних інтерфейсів.* Фреймворк підтримує стандартні інтерфейси Raspberry Pi, такі як GPIO, I2C і SPI, що дозволяє підключати широкий спектр пристроїв – від простих датчиків руху чи температури до складніших модулів, таких як реле чи сервомотори. Ця гнучкість робить WebIOPi універсальним інструментом для інтеграції різноманітних компонентів у систему «Розумний дім».
3. *Веббазований інтерфейс.* Використання вебінтерфейсу дозволяє керувати системою з будь-якого пристрою, підключеного до мережі, без необхідності встановлення додаткового програмного забезпечення. Користувач може отримати доступ до системи через браузер на ПК, смартфоні чи планшеті, що робить управління зручним і доступним. Наприклад, можна вимкнути світло в кімнаті з будь-якої точки світу, маючи доступ до Інтернету. Це особливо корисно для тих, хто часто подорожує або хоче мати контроль над домом на відстані.

4. *Гнучкість.* WebIOPi дозволяє створювати власні скрипти на Python для кастомізації функціональності системи. Це дає розробникам можливість адаптувати систему до специфічних потреб користувача.
5. *Інтеграція з IoT.* Завдяки підтримці мережевих протоколів, таких як HTTP та MQTT, WebIOPi забезпечує легку інтеграцію з Інтернетом речей. Це дозволяє пристроям взаємодіяти один з одним і з користувачем через мережу, створюючи повноцінну IoT-екосистему. Наприклад, можна налаштувати систему так, щоб вона отримувала погодні дані з Інтернету і автоматично регулювала клімат у будинку. Це сприяє створенню адаптивних систем, які реагують на зовнішні умови, що є важливим для сучасних розумних домів.

Недоліки WebIOPi.

1. *Безпека.* Як веббазована система, WebIOPi вразлива до вебатак, якщо не налаштована правильно. Наприклад, відсутність автентифікації чи шифрування може дозволити несанкціонованому користувачеві отримати доступ до системи. Для забезпечення безпеки необхідно ретельно налаштувати автентифікацію, використовувати шифрування (HTTPS) та обмежувати доступ до системи лише з довірених IP-адрес.
2. *Обмежена продуктивність.* Оскільки WebIOPi працює на Raspberry Pi, який має обмежені обчислювальні ресурси, він може не справлятися з дуже складними системами або великими обсягами даних. Наприклад, якщо в системі задіяно десятки датчиків і виконавчих пристроїв, які постійно передають дані, Raspberry Pi може перегріватися чи сповільнюватися, що негативно вплине на стабільність системи. Для таких випадків варто розглянути більш потужні платформи, такі як Raspberry Pi 4 або навіть десктопні комп’ютери, що обмежує його використання для великих проєктів.
3. *Масштабованість.* WebIOPi не підходить для великих систем з сотнями пристроїв, оскільки може виникнути проблеми з продуктивністю і керуванням. Наприклад, у багатокімнатному будинку з десятками датчиків

і пристроїв керування, управління через вебінтерфейс може стати громіздким і повільним. Для таких масштабних проєктів краще використовувати більш просунуті рішення, такі як Home Assistant чи Node-RED, які пропонують кращу масштабованість і управління, що є значним недоліком для амбітних систем.

4. *Залежність від мережі.* Для віддаленого керування потрібне стабільне з'єднання з мережею, що може бути проблемою в регіонах з нестабільним Інтернетом чи у випадках відключення електроенергії.
5. *Відсутність активної підтримки.* Проект WebIOPi більше не розвивається, останній коміт на GitHub датований 2014 роком, що означає, що будь-які помилки або вразливості не будуть виправлені. Це робить його ризикованим для використання в критичних системах, оскільки він може не бути сумісним з новішими версіями апаратного забезпечення чи операційних систем.
6. *Обмежена документація і спільнота.* Хоча є деяка документація, вона може бути застарілою, а спільнота не дуже активна, що ускладнює пошук рішень для проблем.

У таблиці 2.2 наведено порівняння WebIOPi з популярними альтернативами, що підкреслює його обмеження для складних систем.

Таблиця 2.2. Порівняння WebIOPi з альтернативами

Критерій	WebIOPi	Home Assistant	Node-RED
Вартість	Безкоштовно	Безкоштовно	Безкоштовно
Продуктивність	Обмежена	Висока	Висока
Масштабованість	Низька	Висока	Висока
Безпека	Потребує налаштування	Вбудована, сильна	Потребує налаштування
Активна підтримка	Немає, застарілий	Є, активна спільнота	Є, активна спільнота

РОЗДІЛ 3
АРХІТЕКТУРА І ПРОГРАМНІ ЗАСОБИ ОДНОПЛАТНОГО КОМП'ЮТЕРА RASPBERRY PI 3 MODEL B

3.1. Одноплатний Raspberry Pi 3 Model B

Raspberry Pi – це серія одноплатних комп'ютерів, створена британським фондом Raspberry Pi Foundation з метою популяризації вивчення програмування та комп'ютерних наук. Raspberry Pi – одноплатний комп'ютер, розроблений британським фондом Raspberry Pi Foundation [25].

Перша модель Raspberry Pi Model B, представлена в лютому 2012 року, стала початком нової ери в доступних комп'ютерних технологіях. Вона оснащувалася одноядерним процесором із тактовою частотою 700 МГц і 256 МБ оперативної пам'яті. Ці характеристики були достатніми для базових освітніх завдань, таких як навчання програмуванню чи створення простих проєктів, наприклад, медіацентрів або ретрогеймінгових консолей. Однак її обмеження, зокрема низька продуктивність і мала кількість пам'яті, не дозволяли ефективно виконувати складніші завдання, такі як багатозадачність чи обробка великих обсягів даних.

Ця модель заклала фундамент для подальшого розвитку, демонструючи, що навіть із мінімальними ресурсами можна створити платформу, яка буде доступною для широкої аудиторії. Її ціна, близько 35 доларів США, зробила її особливо привабливою для шкіл і ентузіастів, що сприяло швидкому поширенню платформи.

У лютому 2015 року була випущена Raspberry Pi 2 Model B, яка стала значним кроком у підвищенні продуктивності. Ця модель отримала чотириядерний процесор із частотою 900 МГц і 1 ГБ оперативної пам'яті, що дозволило значно розширити можливості платформи. Завдяки багатоядерному процесору Raspberry Pi 2 могла ефективно виконувати кілька завдань одночасно, що зробило її придатною для більш вимогливих проєктів, таких як серверні застосунки, розробка IoT-систем чи запуск складніших операційних систем.

Цей крок уперед зберіг доступну ціну (залишився на рівні 35 доларів), що підкреслило філософію платформи – поєднання потужності та економічності. Raspberry Pi 2 Model B стала популярною серед користувачів, які потребували більшої продуктивності, але не хотіли переходити на дорожчі рішення.

Найвищим досягненням цього етапу розвитку стала модель Raspberry Pi 3 Model B (рис 3.1), випущена в лютому 2016 року. Вона не лише підвищила тактову частотою процесора до 1.2 ГГц, але й додала вбудовані модулі Wi-Fi (802.11n) і Bluetooth (4.1).

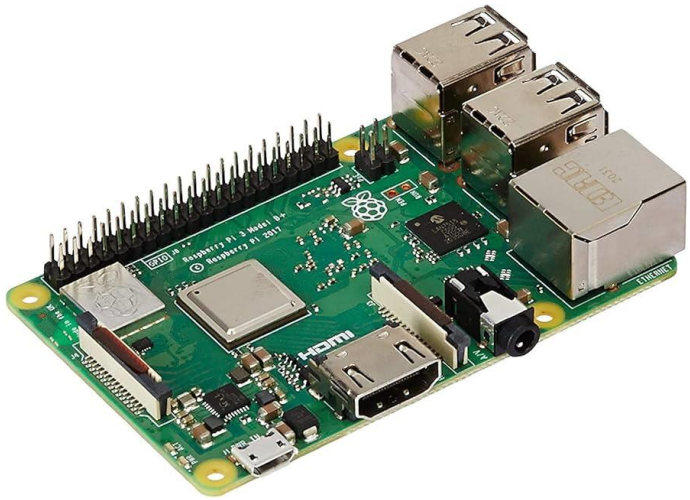

Рисунок 3.1. Одноплатний комп’ютер Raspberry Pi 3 Model B

Ці покращення зробили Raspberry Pi 3 Model B однією з найпопулярніших моделей, адже вона поєднувала в собі доступність, потужність і універсальність. Її вбудовані бездротові модулі спростили створення мережевих проєктів, таких як системи моніторингу чи віддаленого керування.

3.2. Архітектура Raspberry Pi 3 Model B

Одноплатний комп’ютер Raspberry Pi 3 Model B випущений у 2016 році є однією з найпопулярніших моделей у лінійці одноплатних комп’ютерів, розроблених Raspberry Pi Foundation. Її архітектура забезпечує баланс між продуктивністю, компактністю та доступною ціною, що робить її ідеальним вибором для створення систем «Розумний дім».

Система-на-чипі та процесор. Raspberry Pi побудований на системі-на-чипі (SoC) Broadcom BCM2835, яка включає в себе процесор ARM із тактовою частотою 700 МГц, графічний процесор VideoCore IV і 512 чи 256 мегабайтів оперативної пам'яті. Жорсткий диск відсутній, натомість використовується SD карта. Така апаратна начинка дозволяє відтворювати відео формату H.264 в роздільній здатності 1080p [25], що корисно для виведення інформації на монітор чи телевізор у проєктах «Розумний дім».

Оперативна пам'ять. Платформа оснащена 1 ГБ оперативної пам'яті, що дозволяє одночасно обробляти кілька програм, наприклад, сервер WebIOPi для керування пристроями та обробку даних із датчиків. 1 ГБ пам'яті забезпечує достатню продуктивність для більшості освітніх і хобі-проєктів, включаючи IoT-застосунки [26]. Це робить Raspberry Pi 3 Model B придатною для складних сценаріїв автоматизації, де потрібна стабільна робота кількох компонентів.

Мережеві можливості. Однією з ключових інновацій Raspberry Pi 3 Model B є вбудовані модулі Wi-Fi (802.11n) і Bluetooth (4.1). Raspberry Pi 3 Model B став першим у лінійці з інтегрованими бездротовими технологіями, що усунуло потребу в зовнішніх адаптерах [27]. Ці модулі дозволяють легко підключати платформу до локальної мережі чи інших пристроїв, що є критично важливим для систем «Розумний дім». Наприклад, Wi-Fi забезпечує віддалений доступ до вебінтерфейсу для керування освітленням чи моніторингу температури, а Bluetooth може використовуватися для зв'язку з мобільними пристроями.

GPIO-піни та периферійні порти. Raspberry Pi 3 Model B має 40 GPIO-пінів, які дозволяють підключати різноманітні датчики (наприклад, температури, руху) та актуатори (реле, мотори). Ці піни підтримують протоколи, такі як I²C і SPI, що забезпечує гнучкість у взаємодії з апаратними компонентами.

Додатково оснащення. Чотири порти USB 2.0 для підключення периферійних пристроїв, таких як камери чи зовнішні накопичувачі. Етернет-порт зі швидкістю 100 Мбіт/с для стабільного мережевого з'єднання. HDMI-порт, який підтримує вивід відео у роздільній здатності до 1080p. Слот для microSD-карти, який слугує основним сховищем даних, замінюючи жорсткий диск.

3.3. Операційні системи для Raspberry Pi

Вибір операційної системи для Raspberry Pi є одним із ключових етапів у розробці проєктів, особливо для систем «Розумний дім», де потрібна стабільна взаємодія між апаратним забезпеченням і програмним, таким як WebIOPi. Операційна система визначає, наскільки ефективно пристрій може обробляти дані, керувати GPIO-пінами та підтримувати мережеві функції, необхідні для віддаленого керування. У цьому підпункті розглядаються основні операційні системи, їхні переваги та недоліки, з акцентом на придатність для проєктів із використанням WebIOPi.

Raspberry Pi OS: оптимальний вибір. Raspberry Pi OS, раніше відома як Raspbian, є офіційною операційною системою, розробленою Raspberry Pi Foundation. Вона базується на дистрибутиві Debian і спеціально адаптована для апаратного забезпечення Raspberry Pi, що забезпечує високу продуктивність і сумісність. Raspberry Pi OS є рекомендованою операційною системою для більшості користувачів, забезпечуючи стабільну та швидку роботу [28]. Основні переваги включають:

- *оптимізація для апаратного забезпечення*. Система максимально використовує ресурси Raspberry Pi, що дозволяє ефективно обробляти дані від датчиків і керувати пристроями в реальному часі;
- *підтримка GPIO*. Raspberry Pi OS включає бібліотеки, такі як RPi.GPIO, які спрощують програмування для роботи з датчиками та актуаторами, наприклад, для керування освітленням чи моніторингу температури;
- *сумісність із WebIOPi.* Легко встановлюється та налаштовується на цій системі, що є критично важливим для створення вебінтерфейсів у проєктах «Розумний дім»;
- *спільнота та ресурси*. Велика спільнота користувачів надає численні посібники, приклади коду та форуми, що полегшує розробку та вирішення проблем.

Недоліком може бути менш інтуїтивний інтерфейс для користувачів, які звикли до інших дистрибутивів Linux, таких як Ubuntu. Проте для проєктів, орієнтованих на WebIOPi, Raspberry Pi OS залишається найкращим вибором.

Ubuntu MATE: альтернатива для знайомого інтерфейсу. Ubuntu MATE – це легковаговий дистрибутив Ubuntu з графічним середовищем MATE, який підходить для Raspberry Pi. Він пропонує знайомий інтерфейс для користувачів, які мають досвід роботи з Ubuntu на персональних комп'ютерах. Ubuntu MATE забезпечує зручне робоче середовище, але може бути менш оптимізованим для Raspberry Pi порівняно з Raspberry Pi OS [29]. Переваги включають:

- *зручність для користувачів Ubuntu*. Інтерфейс MATE є інтуїтивним і схожим на традиційні настільні системи, що полегшує перехід для новачків;
- *доступ до репозиторіїв Ubuntu*. Користувачі можуть встановлювати широкий спектр програмного забезпечення, що корисно для розробки складних проєктів.

Однак Ubuntu MATE має певні обмеження. Вона менш оптимізована для апаратного забезпечення Raspberry Pi, що може призводити до зниження продуктивності, особливо при виконанні багатозадачних операцій. Крім того, встановлення WebIOPi на Ubuntu MATE може вимагати додаткових налаштувань, а в деяких випадках користувачі стикаються з проблемами сумісності, що ускладнює її використання в проєктах, подібних до цього.

Інші операційні системи. Окрім Raspberry Pi OS і Ubuntu MATE, для Raspberry Pi доступні й інші операційні системи, такі як Windows 10 IoT Core і Home Assistant OS.

3.4. Робота з GPIO на Raspberry Pi

Універсальні піни вводу-виводу (General Purpose Input/Output, GPIO) є однією з ключових особливостей одноплатного комп'ютера Raspberry Pi, що забезпечує його здатність взаємодіяти з апаратними компонентами. У контексті

систем «Розумний дім» GPIO-піни дозволяють підключати датчики, актуатори та інші пристрої, створюючи основу для автоматизації.

Raspberry Pi 3 Model B оснащений 40 GPIO-пінами (рис 3.2), які можна налаштувати як вхідні або вихідні залежно від потреб проєкту. Вхідні піни використовуються для зчитування сигналів від зовнішніх пристроїв, таких як датчики руху чи температури, тоді як вихідні піни надсилають сигнали для керування пристроями, наприклад, реле чи світлодіодами.

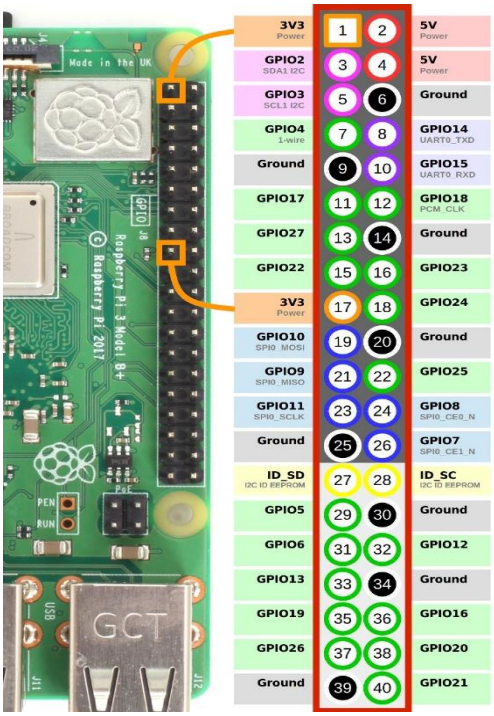

Рисунок 3.2. Схема розташування та нумерації GPIO-пінів Raspberry Pi 3 Model B

GPIO-піни дозволяють Raspberry Pi взаємодіяти з фізичним світом, роблячи його ідеальним для створення IoT-пристроїв [30]. Ці піни підтримують цифрові сигнали, а також протоколи, такі як I²C і SPI, що розширює можливості підключення складніших компонентів.

У системах «Розумний дім» GPIO-піни є основою для реалізації автоматизованих функцій. Наприклад, датчик температури DHT11 може передавати дані через вхідний пін, а реле, підключене до вихідного піна, вмикає опалення при зниженні температури. Аналогічно, RFID-читач може бути підключений для розпізнавання тегів і керування доступом до будинку.

Для роботи з GPIO-пінами необхідно правильно їх налаштувати. Кожен пін може бути сконфігурований як:

- *вхідний*. Для зчитування стану (високий або низький рівень сигналу). Наприклад, датчик руху видає високий сигнал при виявленні активності;
- *вихідний*. Для надсилання сигналу (встановлення високого або низького рівня). Наприклад, вихідний пін може активувати реле для вмикання світла.

Налаштування здійснюється програмно, найчастіше за допомогою бібліотеки RPi.GPIO у мові Python. Перед використанням пінів необхідно визначити їхню нумерацію: фізичну (за розташуванням на платі) або BCM (за номерами процесора Broadcom). У цьому проєкті використовується BCM-нумерація, оскільки вона є стандартною для бібліотеки RPi.GPIO і полегшує інтеграцію з WebIOPi.

3.5. Інтеграція з WebIOPi

У контексті розробки систем «Розумний дім» на базі Raspberry Pi 3 Model B важливу роль відіграє фреймворк WebIOPi, який забезпечує створення вебінтерфейсів для керування апаратними компонентами через мережу. WebIOPi дозволяє користувачам віддалено взаємодіяти з GPIO-пінами, що робить його ключовим інструментом для автоматизації та моніторингу. Цей підрозділ присвячено опису функціональних можливостей WebIOPi, процесу його інтеграції з Raspberry Pi, а також прикладу практичного застосування в системах «Розумний дім». Основні можливості включають:

- *віддалене керування GPIO*. Користувачі можуть вмикати або вимикати пристрої, підключені до GPIO-пінів, такі як реле для освітлення чи мотори, з будь-якого пристрою, підключеного до мережі;
- *моніторинг датчиків*. WebIOPi дозволяє зчитувати дані з датчиків, наприклад, температури чи руху, і відображати їх у реальному часі;
- *REST API*. Надає програмний доступ до GPIO, що дозволяє інтегрувати систему з іншими додатками чи мобільними програмами;

- *кастомізація інтерфейсу*. Користувачі можуть створювати власні вебсторінки з кнопками, графіками чи індикаторами для керування системою.

Ці можливості роблять WebIOPi ідеальним інструментом для проєктів «Розумний дім», де потрібен зручний і гнучкий доступ до апаратного забезпечення через мережу.

Інтеграція WebIOPi з Raspberry Pi потребує виконання кількох етапів налаштування, що забезпечують стабільну роботу фреймворку. Основні кроки включають:

1. *Встановлення WebIOPi*. Фреймворк встановлюється на Raspberry Pi OS через менеджер пакетів або завантаження з офіційного репозиторію. Наприклад, команда *sudo pip install webiopi* встановлює необхідні компоненти [32]. Важливо переконатися, що Python і pip встановлені заздалегідь.
2. *Налаштування скриптів Python*. Для керування GPIO через WebIOPi створюються скрипти Python, які визначають функції для роботи з пінами. Наприклад, функція для вмикання реле може бути позначена як макрос WebIOPi для виклику через вебінтерфейс.
3. *Конфігурація вебсерверу*. WebIOPi запускає вебсервер, доступний за локальною адресою (наприклад, http://<IP-адреса>:8000). Для віддаленого доступу необхідно налаштувати порт (за замовчуванням 8000) і, за потреби, забезпечити безпечне з’єднання через HTTPS.
4. *Створення інтерфейсу*. WebIOPi дозволяє використовувати HTML, JavaScript і CSS для створення кастомізованих вебсторінок. Наприклад, можна створити кнопку для вмикання світла чи графік для відображення температури.

WebIOPi спрощує створення мережевих проєктів, дозволяючи керувати апаратним забезпеченням без глибоких знань веброзробки [30]. Після налаштування система стає доступною через браузер, що забезпечує зручність використання.

РОЗДІЛ 4
ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБІНТЕРФЕЙСУ МОНІТОРИНГУ І УПРАВЛІННЯ СИСТЕМОЮ «РОЗУМНИЙ ДІМ» НА БАЗІ WEBIOPI

4.1. Загальний опис системи «Розумний дім» на базі WebIOPi

Розроблена в рамках цієї кваліфікаційної роботи система «Розумний дім» є практичним рішенням для автоматизації побутових процесів, побудованим на базі одноплатного комп'ютера Raspberry Pi 3 Model B. Основною метою системи є створення зручного вебінтерфейсу, який дозволяє користувачам віддалено моніторити стан будинку та керувати його пристроями через локальну мережу. Центральним елементом системи є фреймворк WebIOPi, який забезпечує інтеграцію апаратного забезпечення з вебтехнологіями, дозволяючи створювати інтерактивні рішення для автоматизації. Система охоплює моніторинг мікроклімату, керування візуальними та звуковими сигналами, а також контроль доступу, що робить її універсальним інструментом для підвищення комфорту та безпеки в побуті.

Метою розробки системи є створення функціонального рішення для автоматизації будинку, яке дозволяє користувачам відстежувати параметри середовища та керувати пристроями через веббраузер. Система призначена для спрощення повсякденних задач, таких як контроль температури та вологості, управління освітленням, забезпечення безпеки через ідентифікацію користувачів і сповіщення про зовнішні події. WebIOPi відіграє ключову роль, дозволяючи створювати вебінтерфейс, який забезпечує доступ до апаратних компонентів Raspberry Pi через локальну мережу. Це дозволяє користувачам взаємодіяти з системою з будь-якого пристрою, підключеного до мережі, наприклад, зі смартфона чи комп'ютера, що підвищує зручність і практичність використання.

Основні компоненти системи. Система «Розумний дім» інтегрує кілька апаратних компонентів, кожен із яких виконує специфічні функції:

- *датчик температури та вологості DHT11*. Використовується для вимірювання температури та вологості в приміщенні, що дозволяє моніторити мікроклімат і відображати дані в реальному часі через вебінтерфейс;
- *RGB LED (KY-016)*. Служить для візуальних сповіщень, дозволяючи змінювати кольори (червоний, зелений, синій) або створювати ефект переливання для індикації стану системи;
- *RFID-читач MFRC522*. Забезпечує контроль доступу шляхом зчитування RFID-тегів, що дозволяє ідентифікувати користувачів і забезпечувати безпечний доступ до будинку;
- *датчик звуку LM393*. Виявляє звукові сигнали в приміщенні, що може бути використано для моніторингу безпеки чи автоматизації реакцій на шум;
- *пасивний зумер*. Відтворює звукові сигнали для сповіщень, наприклад, при дозволі чи забороні доступу через RFID або для інших подій.

Ці компоненти підключені до GPIO-пінів Raspberry Pi 3 Model B (рис 4.1), що дозволяє програмно керувати ними через WebIOPi. Код для HTML, CSS розміщено в додатку Б, а для Python, JavaScript у додатку В.

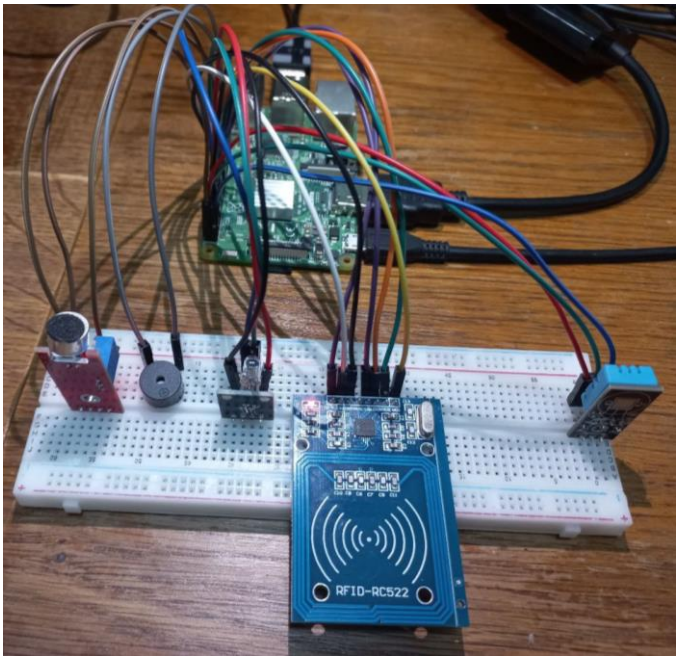

Рисунок 4.1. Схема підключення датчиків
до одноплатного комп’ютера Raspberry Pi 3 Model B

Функціональність системи. Система пропонує набір функцій, які забезпечують її універсальність і практичність:

- *моніторинг температури та вологості*. Відображення поточних даних із датчика DHT11 та історії змін у вигляді графіка для аналізу мікроклімату;
- *керування освітленням*. Можливість увімкнення/вимкнення RGB LED, вибору кольору або активації режиму переливання через вебінтерфейс;
- *контроль доступу*. Використання RFID-читача для ідентифікації користувачів із відтворенням відповідних звукових сигналів через зумер;
- *виявлення звуку*. Моніторинг звукових подій у приміщенні з відображенням статусу в реальному часі;
- *звукові сповіщення*. Відтворення мелодій через зумер для сигналізації подій, таких як доступ чи виявлення звуку;
- *відображення часу*. Показ поточного системного часу Raspberry Pi для зручності користувача.

Ці функції реалізовані через вебінтерфейс (рис 4.2), який забезпечує інтуїтивне керування та моніторинг у реальному часі.

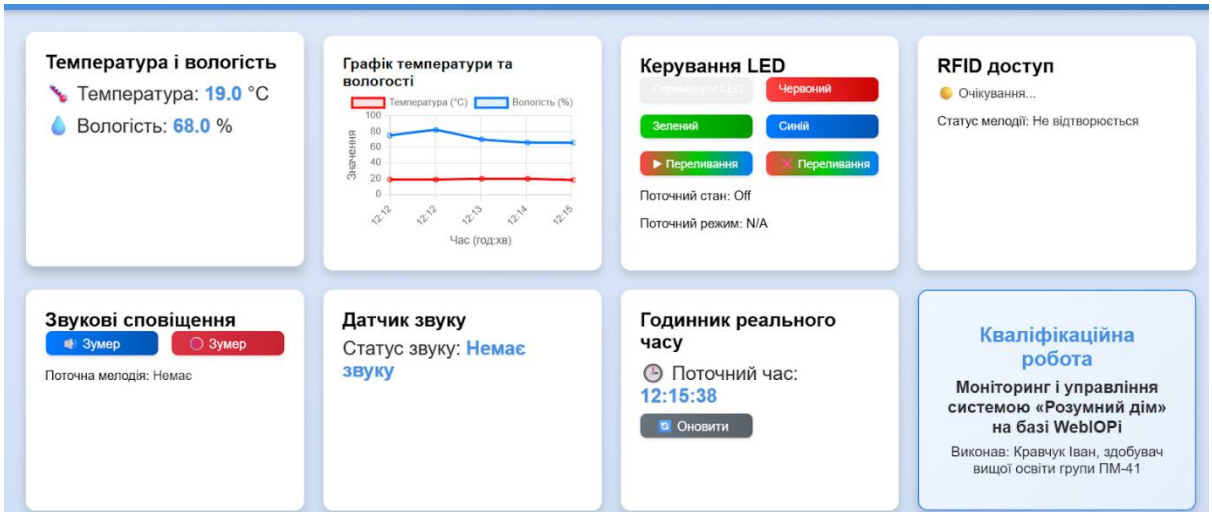

Рисунок 4.2. Вебінтерфейс для моніторингу та управління системою «Розумний дім» на базі WebIOPi

Інтеграція з WebIOPi. WebIOPi є центральним елементом системи, забезпечуючи зв’язок між апаратним забезпеченням і вебінтерфейсом. Фреймворк

дозволяє створювати макроси в Python, які викликаються через HTTP-запити з фронтенду, побудованого на HTML, CSS і JavaScript. Наприклад, макроси для зчитування даних із датчика DHT11 чи керування RGB LED дозволяють користувачам взаємодіяти з системою через браузер. WebIOPi працює як вебсервер на Raspberry Pi, доступний через локальну мережу, що забезпечує віддалений доступ до всіх функцій системи. Це дозволяє користувачам моніторити стан будинку та керувати пристроями з будь-якої точки локальної мережі.

4.2. Технологічна основа системи «Розумний дім» на базі WebIOPi

Система побудована на базі одноплатного комп'ютера Raspberry Pi 3 Model B, який виконує роль основної платформи для обробки даних і керування пристроями. Його апаратні можливості, зокрема 40 GPIO-пінів, вбудовані модулі Wi-Fi та Bluetooth, дозволяють ефективно підключати зовнішні пристрої та забезпечувати мережевий доступ. До складу системи входять:

- *Raspberry Pi 3 Model B.* Забезпечує обчислювальну потужність для виконання програмного коду, обробки даних із датчиків і хостингу вебсервера WebIOPi. Wi-Fi-модуль дозволяє користувачам отримувати доступ до системи через локальну мережу;
- *датчики та актуатори*. Включають датчик температури та вологості DHT11 (GPIO 14), RGB LED (KY-016, GPIO 17, 27, 22), RFID-читач MFRC522 (SPI-піни), датчик звуку LM393 (GPIO 6) і пасивний зумер (GPIO 26). Ці компоненти забезпечують моніторинг і керування в системі «Розумний дім».

Апаратне забезпечення підібрано з урахуванням його доступності, сумісності з Raspberry Pi та здатності виконувати задачі автоматизації.

Програмне забезпечення. Програмна частина системи складається з інструментів, які забезпечують створення вебінтерфейсу, обробку даних і взаємодію з апаратним забезпеченням:

- *WebIOPi*. Фреймворк, який дозволяє створювати вебінтерфейси для керування GPIO-пінами Raspberry Pi. Він працює як вебсервер у локальній мережі, забезпечуючи доступ до макросів Python через HTTP-запити. WebIOPi є ключовим елементом, що об’єднує апаратне та програмне забезпечення, дозволяючи користувачам взаємодіяти з системою через браузер.
- *Python*. Використовується для реалізації бекенд-логіки, зокрема макросів WebIOPi, які керують апаратними компонентами. Застосовуються бібліотеки:
  - *RPi.GPIO*. Для керування GPIO-пінами, наприклад, для активації RGB LED чи зумера.
  - *Adafruit_DHT*. Для зчитування даних із датчика DHT11, що забезпечує моніторинг температури та вологості.
  - *MFRC522*. Для обробки даних із RFID-читача MFRC522, що дозволяє реалізувати контроль доступу.
- *HTML та CSS*. HTML визначає структуру вебінтерфейсу, включаючи секції для відображення даних і керування пристроями. CSS забезпечує сучасний дизайн із градієнтними кнопками та адаптивною верстkою для різних пристроїв.
- *JavaScript*. Відповідає за інтерактивність, обробляючи дії користувача, такі як натискання кнопок, і викликаючи макроси WebIOPi для оновлення даних у реальному часі.
- *Chart.js*. Бібліотека для створення графіків, яка використовується для візуалізації історичних даних температури та вологості, що додає зручності для аналізу мікроклімату.

Ці програмні інструменти забезпечують створення інтуїтивного та функціонального вебінтерфейсу, який відповідає потребам системи «Розумний дім».

Інтеграція компонентів через WebIOPi. WebIOPi є центральним елементом системи, що забезпечує зв’язок між апаратним і програмним забезпеченням.

Фреймворк дозволяє створювати макроси в Python, які обробляють запити від фронтенду, побудованого на HTML, CSS і JavaScript. WebIOPi працює як вебсервер на Raspberry Pi, доступний у локальній мережі, що забезпечує віддалений доступ до всіх функцій системи, таких як моніторинг температури, керування світлом чи контроль доступу.

4.3. Моніторинг температури та вологості

Моніторинг температури та вологості дозволяє користувачам відстежувати мікроклімат у приміщенні через вебінтерфейс. Ця функція реалізована за допомогою датчика DHT11 (рис. 4.3), інтегрованого з Raspberry Pi 3 Model B, і фреймворку WebIOPi, який забезпечує віддалений доступ до даних через локальну мережу.

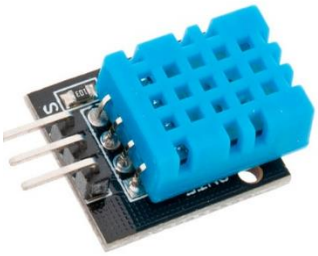

Рисунок 4.3. Датчик температури та вологості DHT11

А*паратна частина.* Для моніторингу температури та вологості використано датчик DHT11, який підключений до GPIO 14 (фізичний пін 8) Raspberry Pi 3 Model B. Цей датчик є економічним і надійним рішенням для побутових проєктів, забезпечуючи вимірювання температури в діапазоні від 0 до 50 °C і вологості від 20% до 90%. Підключення датчика включає:

- пін VCC до 3.3V (фізичний пін 1) для живлення;
- пін GND до заземлення (фізичний пін 6);
- пін DATA до GPIO 14 для передачі даних.

Датчик DHT11 передає цифрові дані, які обробляються Raspberry Pi для подальшого відображення у вебінтерфейсі.

Програмна реалізація моніторингу температури та вологості базується на використанні бібліотеки Adafruit_DHT для Python, яка забезпечує зчитування даних із датчика DHT11. У системі створено два макроси WebIOPi для обробки даних:

- *макрос getTemperatureHumidity*. Зчитує поточні значення температури та вологості з датчика DHT11 і повертає їх у форматі рядка, розділеного крапкою з комою. Цей макрос викликається з вебінтерфейсу для оновлення даних у реальному часі;
- *макрос getDHTHistory*. Зберігає історію вимірювань (час, температура, вологість) у масиві та повертає її у форматі JSON для побудови графіка.

Вебінтерфейс для моніторингу температури та вологості створено за допомогою HTML, CSS і JavaScript. У додатку Б (HTML/CSS) визначено секцію з двома елементами:

- *поточні дані*. Відображаються значення температури (у °C) і вологості (у %) у реальному часі;
- *графік*. Використовується бібліотека Chart.js для створення лінійного графіка, який відображає історію змін температури та вологості за останні 6 годин (360 записів із періодичністю раз на хвилину).

JavaScript-код у додатку В періодично викликає макроси WebIOPi для оновлення даних. Наприклад, функція refreshDHT викликає getTemperatureHumidity кожну хвилину, оновлюючи значення на сторінці, а функція updateDHTChart викликає getDHTHistory для оновлення графіка. Це забезпечує динамічне відображення даних у вебінтерфейсі, доступному через локальну мережу.

4.4. Керування LED

Керування світлодіодом (LED) забезпечує користувачам зручність у керуванні освітленням через вебінтерфейс. Ця функція реалізована за допомогою модуля KY-016 (рис. 4.4).

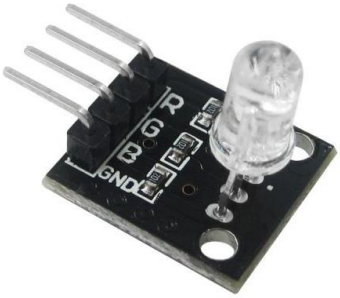

Рисунок 4.4. Модуль RGB світлодіода KY-016

Апаратна частина. Для реалізації керування освітленням використано RGB LED модуль KY-016, який підключений до GPIO-пінів Raspberry Pi 3 Model B, а саме: червоний канал підключено до GPIO 17 (фізичний пін 11), зелений канал підключено до GPIO 27 (фізичний пін 13), синій канал підключено до GPIO 22 (фізичний пін 15).

Підключення RGB LED до GPIO-пінів дозволяє програмно керувати його станом, що є основою для реалізації функції через WebIOPi.

Програмна реалізація керування LED базується на використанні бібліотеки RPi.GPIO для Python, яка забезпечує доступ до GPIO-пінів Raspberry Pi. У системі створено кілька макросів WebIOPi для управління LED, які викликаються через вебінтерфейс:

- *макрос toggle_led*. Перемикає стан LED між ввімкненим і вимкненим. Наприклад, якщо LED вимкнено, цей макрос увімкне його, встановивши відповідні рівні напруги на каналах;
- *макрос set_red*. Встановлює LED на червоний колір, активуючи канал GPIO 17 і вимикаючи канали GPIO 27 і 22;
- *макрос set_green*. Встановлює LED на зелений колір, активуючи канал GPIO 27 і вимикаючи інші;
- *макрос set_blue*. Встановлює LED на синій колір, активуючи канал GPIO 22.
- *макрос start_transition*. Запускає режим переливання кольорів, змінюючи інтенсивність каналів у циклі для створення ефекту плавного переходу між кольорами;

- *макрос stop_transition*. Зупиняє режим переливання, повертаючи LED до статичного стану, наприклад, останнього вибраного кольору.

Ці макроси інтегровані з WebIOPi, що дозволяє викликати їх через HTTP-запити з фронтенду. Наприклад, макрос set_red використовує PWM для встановлення максимальної яскравості на червоному каналі, забезпечуючи чітке відображення кольору. Режим переливання реалізований через циклічне зміну рівнів напруги на всіх трьох каналах, що створює ефект зміни кольорів, який можна зупинити в будь-який момент.

Вебінтерфейс для керування LED створений за допомогою HTML, CSS і JavaScript, що забезпечує інтуїтивне та зручне управління через браузер. У додатку Б (HTML/CSS) визначено секцію з кнопками для керування LED:

- кнопка «Перемкнути LED» викликає макрос toggle_led, дозволяючи користувачам вмикати чи вимикати світло;
- кнопки «Червоний», «Зелений», «Синій» викликають відповідні макроси set_red, set_green, set_blue, дозволяючи вибрати бажаний колір;
- кнопки «▶ Переливання» і «✕ Переливання» викликають макроси start_transition і stop_transition, забезпечуючи активацію та зупинку режиму переливання кольорів.

JavaScript-код у додатку В обробляє кліки по цих кнопках і викликає відповідні макроси WebIOPi через HTTP-запити. Наприклад, при натисканні на кнопку «Червоний» JavaScript викликає макрос set_red, який встановлює LED на червоний колір, оновлюючи стан у реальному часі. Це забезпечує інтерактивність і зручність у керуванні світлодіодом через браузер, доступний у локальній мережі.

4.5. Система доступу на основі RFID

Контроль доступу через RFID забезпечує безпеку шляхом ідентифікації користувачів за допомогою RFID-тегів. Ця функція дозволяє визначати, чи дозволений доступ, відображати статус у вебінтерфейсі та супроводжувати результат звуковими сповіщеннями через зумер.

Апаратна частина. Для реалізації контролю доступу використано RFID-читач MFRC522 (рис 4.5), який підключений до SPI-пінів Raspberry Pi 3 Model B.

Рисунок 4.5. RFID модуль з карткою доступу

Конкретна конфігурація підключення:

- пін SDA (SS) до GPIO 24 (фізичний пін 18);
- пін MOSI до GPIO 10 (фізичний пін 19);
- пін MISO до GPIO 9 (фізичний пін 21);
- пін SCK до GPIO 11 (фізичний пін 23);
- пін VCC до 3.3V (фізичний пін 1);
- пін GND до заземлення (фізичний пін 6).

RFID-читач MFRC522 зчитує унікальні ідентифікатори (UID) RFID-тегів, що дозволяє ідентифікувати користувачів для надання або заборони доступу, наприклад, для відкриття дверей у будинку. Зумер, підключений до GPIO 26, використовується для відтворення звукових сигналів, які вказують на дозвіл або заборону доступу.

Програмна реалізація контролю доступу базується на бібліотеці mfrc522 для Python, яка забезпечує зчитування даних із RFID-тегів. У системі створено макрос WebIOPi get_rfid_status, який виконує наступні дії:

- зчитує UID тегу, наближеного до читача;
- перевіряє чи UID відповідає дозволеному значенню;
- у разі дозволеного UID активує мелодію дозволу через зумер і встановлює статус « Доступ дозволено »;

- у разі невідповідного UID відтворює мелодію заборони і встановлює статус « ● Доступ заборонено ».

Мелодії генеруються функцією play_melody, яка керує зумером на GPIO 26, створюючи звукові сигнали шляхом швидкого перемикання високого та низького рівнів напруги. Макрос get_rfid_status повертає статус у форматі JSON, що містить інформацію про доступ і стан мелодії, для подальшого відображення у вебінтерфейсі.

4.6. Виявлення звуку

Виявлення звуку забезпечують додатковий рівень безпеки та автоматизації. Виявлення звуку дозволяє моніторити середовище на наявність шуму, що може бути корисним для виявлення незвичайної активності чи для автоматизації реакцій на звукові події.

Апаратна частина. Для реалізації виявлення звуку використано датчик звуку LM393 (рис. 4.6). Підключений до GPIO 6 (фізичний пін 31) Raspberry Pi 3 Model B. Цей датчик виявляє звукові сигнали в оточуючому середовищі, надсилаючи сигнал на Raspberry Pi при виявленні шуму. Він працює як цифровий вхід, змінюючи стан піна на LOW при виявленні звуку.

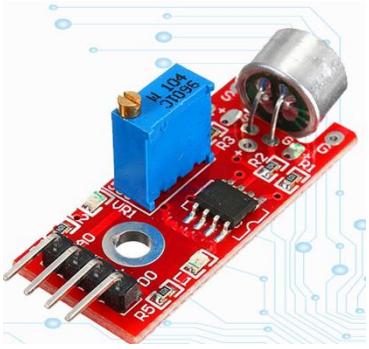

Рисунок 4.6. Датчик звуку LM393

Програмна реалізація виявлення звуку та звукових сповіщень базується на бібліотеці RPi.GPIO для Python. Для виявлення звуку використовується функція

monitor_sound, яка налаштовує GPIO 6 на вхідний режим і додає обробник подій для виявлення змін стану піна. Коли датчик виявляє звук (пін стає LOW), встановлюється прапорець sound_detected на True, а після затримки в 3 секунди – на False. Це дозволяє системі реагувати на звукові події, наприклад, вмикати світло чи активувати інші пристрої, із врахуванням тимчасового характеру звуку.

У системі створено макрос WebIOPi getSoundStatus, який повертає поточний стан виявлення звуку (виявлено чи не виявлено), дозволяючи оновлювати вебінтерфейс у реальному часі. Наприклад, якщо звук виявлено, статус змінюється на «Звук виявлено», а після затримки – на «Немає звуку».

Вебінтерфейс для виявлення звуку (рис 4.7) створений за допомогою HTML, CSS і JavaScript.

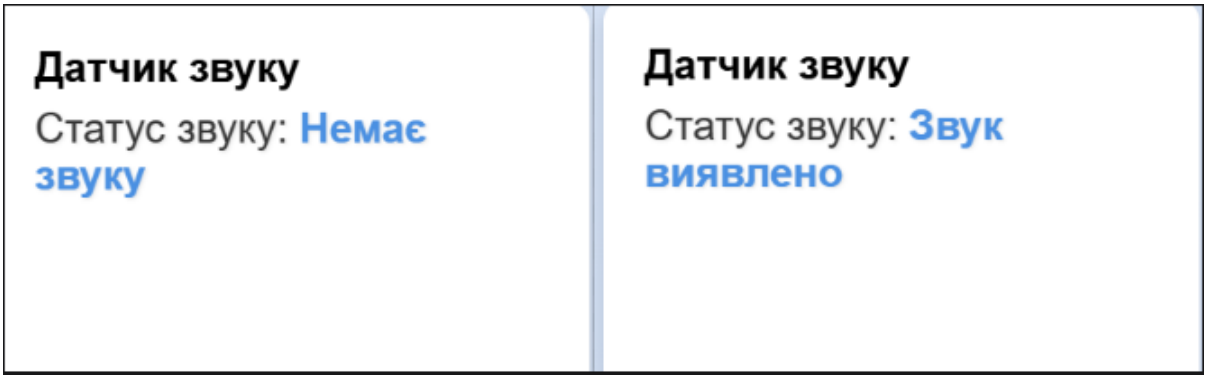

Рисунок 4.7. Вебінтерфейс для виявлення звуку

4.7. Відтворення звукових сигналів

Відтворення звукових сигналів забезпечує аудіопідтвердження подій, таких як контроль доступу через RFID або ручна активація сповіщень. Ця функція реалізована за допомогою пасивного зумера, інтегрованого з вебінтерфейсом, що дозволяє користувачам віддалено керувати звуковими сповіщеннями через локальну мережу.

Апаратна частина. Для відтворення звукових сигналів використано пасивний зумер (рис 4.8), підключений до GPIO 26 (фізичний пін 37) Raspberry Pi 3 Model B. На відміну від активного зумера, пасивний зумер потребує програмного

керування частотою сигналу для створення звуку, що дозволяє генерувати мелодії різної тональності.

Рисунок 4.8. Пасивний зумер

Підключення включає:

- позитивний пін зумера до GPIO 26 для передачі сигналу;
- негативний пін до заземлення (фізичний пін 39) для завершення електричного кола.

Така конфігурація дозволяє Raspberry Pi керувати зумером, створюючи звукові сповіщення для різних подій у системі, наприклад, підтвердження доступу чи сигналізації про виявлення звуку.

Програмна реалізація відтворення звукових сигналів базується на бібліотеці RPi.GPIO для Python, яка забезпечує керування GPIO-пінами. Основна функція play_melody відповідає за генерацію звуку, приймаючи масив частот нот (наприклад, CL, CM, CH) і тривалостей (у секундах). Функція працює шляхом:

- перемикання GPIO 26 між високим і низьким рівнями напруги для створення звукової хвилі;
- обчислення періоду сигналу на основі частоти ноти (наприклад, період = 1/частота);
- виконання циклів для створення звуку протягом заданої тривалості.

Для інтеграції з вебінтерфейсом створено два макроси WebIOPi:

- *макрос buzzer*. Запускає відтворення мелодії дозволу (ноти [CM[1], CM[3], CM[5]] з тривалістю [0.5, 0.5, 0.5]) у окремому потоці, щоб не блокувати основний процес. Повертає повідомлення «Buzzer OK» у разі успіху;

- *макрос stop_buzzer*. Зупиняє відтворення мелодії, встановлюючи прапорець buzzer_running на False і вимикаючи зумер (GPIO 26 на LOW). Повертає повідомлення «Buzzer stopped» або «Buzzer not running».

Ці макроси дозволяють викликати функції відтворення звуку через HTTP-запити з вебінтерфейсу, забезпечуючи віддалене керування.

Вебінтерфейс для керування звуковими сповіщеннями створено за допомогою HTML, CSS і JavaScript. У додатку Б визначено секцію з елементами:

- кнопка «🔊 Зумер» для активації мелодії;
- кнопка « Зумер» для зупинки відтворення.

JavaScript-код у додатку В обробляє кліки по кнопках, викликаючи відповідні макроси WebIOPi:

- при натисканні на «🔊 Зумер» викликається макрос buzzer, який запускає мелодію, і відображається повідомлення про успішну активацію;
- при натисканні на « Зумер» викликається макрос stop_buzzer, який зупиняє мелодію, із відповідним повідомленням.

Це забезпечує інтерактивне керування зумером через браузер, із відображенням поточного стану відтворення.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було досягнуто значних результатів у вивченні теоретичних основ, освоєнні інструментальних засобів та розробці практичного рішення для автоматизації побутових процесів. Основні підсумки роботи такі:

1. Проведено аналіз літературних джерел, що дозволило сформувати цілісне уявлення про принципи функціонування систем «Розумний дім». Це сприяло розумінню ключових аспектів архітектури таких систем і викликів, пов’язаних з їхньою реалізацією.

2. Опановано інструментальні та функціональні засоби фреймворка WebIOPi для створення вебінтерфейсів, які забезпечують керування портами GPIO одноплатного комп’ютера Raspberry Pi.

3. Детально розглянуто і досліджено апаратне та програмне забезпечення одноплатного комп’ютера Raspberry Pi 3 Model B. Налаштовано операційну систему Raspberry Pi OS і реалізовано програмні рішення на мові програмування Python для взаємодії з датчиками та виконавчими пристроями, що стало основою для функціонування системи «Розумний дім».

4. Спроектовано та реалізовано повнофункціональний вебінтерфейс для моніторингу й управління системою «Розумний дім» на базі WebIOPi, який є ключовим результатом роботи. Інтерфейс забезпечує зручне відстеження стану датчиків і керування пристроями в реальному часі, пропонуючи інтуїтивно зрозумілу взаємодію для користувачів. Застосовано принципи адаптивного дизайну, що ґрунтуються на сучасних стандартах веброзробки для забезпечення оптимальної сумісності з різними пристроями, включаючи настільні комп’ютери, планшети та смартфони.

Розроблена система «Розумний дім» є економічно ефективним, масштабованим і гнучким рішенням, яке може бути адаптоване до реальних умов експлуатації та вдосконалене для подальшого використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розумний дім – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Розумний_дім (дата звернення: 20.10.2024).

2. Spicer D. If You Can’t Stand the Coding, Stay Out of the Kitchen: Three Chapters in the History of Home Automation. Dr. Dobb’s Journal. 2000. Aug. 12. URL: https://www.drdobbs.com/architecture-and-design/if-you-cant-stand-the-coding-stay-out-of/184404040 (дата звернення: 23.10.2024).

3. Tanenbaum A.S., Wetherall D.J. Computer Networks: 5th ed. Boston: Pearson, 2011. 960 с.

4. Weiser M. The Computer for the 21st Century. Scientific American. 1991. Т. 265, № 3. С. 94–104.

5. Дужак І.О. «Розумний дім». Автоматизація технологічних і бізнес-процесів. 2013. № 13–14. С. 31.

6. Goodin D. Guerilla researcher created epic botnet to scan billions of IP addresses. Ars Technica. 2013. March 20. URL: https://arstechnica.com/information-technology/2013/03/guerilla-researcher-created-epic-botnet-to-scan-billions-of-ip-addresses/ (дата звернення: 01.11.2024).

7. MQTT. Wikipedia. URL: https://en.wikipedia.org/wiki/MQTT (дата звернення: 06.11.2024).

8. Sethi P., Sarangi S. Internet of Things: Architectures, Protocols, and Applications. Journal of Electrical and Computer Engineering. 2017. DOI: 10.1155/2017/9324035.

9. Galeev M.T.Catching the Z-Wave.Embedded Systems Design.2006. T.19, № 10.

10. Crist R. Apple, Amazon, Google, and others want to create a new standard for smart home tech. CNET. 2019. Dec. 18. URL: https://www.cnet.com/home/smart-home/apple-amazon-google-and-others-want-to-create-a-new-standard-for-smart-home-tech/ (дата звернення: 07.11.2024).

11. Liu F., Zhao H. The Design of WIFI-Based Smart Home Communication Hardware Adapter. 2015 Fifth International Conference on Instrumentation and

Measurement, Computer, Communication and Control (IMCCC), Qinhuangdao, 2015. С. 1193–1197.

12. Mario B., Candid W. Insecurity in the Internet of Things. SECURITY RESPONSE. 2015. С. 9–14.

13. Hercegova K., Baranovskaya T. Smart technologies for energy consumption management. SHS Web of Conferences. 2021. Т. 128. Art. 02005. DOI: 10.1051/shsconf/202112802005.

14. Edimax SP-2101W Smart Plug. URL: https://www.edimax.com/edimax/merchandise/merchandise_detail/data/edimax/in/home_automation_smart_plug/sp-2101w/ (дата звернення: 02.12.2024).

15. Мясищев А.А. Використання одноплатних мінікомп'ютерів і фреймворка WebIOPi для віддаленого доступу до сенсорів. Сучасні інформаційні технології. Комп'ютерна інженерія. 2016. URL: https://elar.khmnu.edu.ua/server/api/core/bitstreams/d8090d13-7319-4d98-9223-6bdc4f0b8214/content (дата звернення: 04.12.2024).

16. Sumi L. Components of Communications Based on IoT-Technologies in Smart Homes Enhancement by Raspberry Pi. Journal of the Asiatic Society of Mumbai. 2022. Т. XCV, № 2. С. 243.

17. Використання веб-інтерфейсу для дистанційного керування Raspberry Pi. Заняття 13-14. mikrotik.kpi.ua. 2015. URL: https://mikrotik.kpi.ua/index.php/courses-list/category-raspberry/76-using-the-web-interface-for-remote-control-raspberry-pi-session-13-14 (дата звернення: 08.12.2024).

18. Скотт Б., Ніл Т. Принципи та шаблони дизайну вебінтерфейсів для багатих взаємодій. Sebastopol: O'Reilly Media, 2009. 352 с. ISBN 978-0596155353.

19. WebIOPi: Tutorial – Framework Basis. URL: https://webiopi.trouch.com/Tutorial_Basis.html (дата звернення: 14.01.2025).

20. Маркотт Е. Адаптивний дизайн: найкращі практики. Interaction Design Foundation. URL: https://www.interaction-design.org/literature/article/responsive-design-let-the-device-do-the-work (дата звернення: 17.01.2025).

21. WebIOPi: Tutorial – Devices. URL: https://webiopi.trouch.com/Tutorial_Devices.html (дата звернення: 21.01 .2025).

22. Chakraborty A., Islam M., Shahriyar F., Islam S., Zaman H., Hasan M. Smart Home System: A Comprehensive Review. Journal of Electrical and Computer Engineering. 2023. DOI: 10.1155/2023/7616683.

23. Smart Home: Threats and Countermeasures. URL: https://www.rambus.com/iot/smart-home/ (дата звернення: 23.02.2025).

24. Buil-Gil D., Kemp S., Kuenzel S., Coventry L., Zakhary S., Tilley D., Nicholson J. The digital harms of smart home devices: A systematic literature review. Computers in Human Behavior. 2023. DOI: 10.1016/j.chb.2023.107770.

25. Raspberry Pi. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Raspberry_Pi (дата звернення: 15.03.2025).

26. Upton E., Halfacree G. Raspberry Pi User Guide. Hoboken, NJ: Wiley, 2016. 312 с. ISBN 9781119264361.

27. Raspberry Pi Versions. Pi My Life Up. URL: https://pimylifeup.com/raspberry-pi-versions/ (дата звернення: 16.03.2025).

28. Operating Systems. Raspberry Pi. URL: https://www.raspberrypi.com/software/operating-systems/ (дата звернення: 20.03.2025).

29. Best Operating Systems for Raspberry Pi. Pi My Life Up. URL: https://pimylifeup.com/raspberry-pi-best-operating-systems/ (дата звернення: 21.03.2025).

30. Documentation. Raspberry Pi. URL: https://www.raspberrypi.com/documentation/ (дата звернення: 03.04.2025).

31. Anderson J. 3 popular programming languages you can learn with Raspberry Pi. Opensource.com. URL: https://opensource.com/article/19/3/programming-languages-raspberry-pi (дата звернення: 10.04.2025).

32. Raspberry Pi GPIO with WebIOPi. Pi My Life Up. URL: https://pimylifeup.com/raspberry-pi-webiopi/ (дата звернення: 11.04.2025).

ДОДАТОК А

МОНІТОРИНГ І УПРАВЛІННЯ СИСТЕМАМИ «РОЗУМНОГО ДОМУ» ЗАСОБАМИ WEBIOPI

Кравчук І.В., здобувач ступеня вищої освіти «бакалавр»
Шинкарчук Н.В., кандидат технічних наук, доцент кафедри інформаційних технологій та моделювання
Рівненський державний гуманітарний університет, м. Рівне, Україна

У сучасних умовах активної цифровізації суспільства значну популярність здобули системи автоматизації побуту, відомі як «розумний дім». Це інноваційні програмно-апаратні комплекси, які забезпечують централізоване або дистанційне керування різноманітними електронними пристроями в житлових чи комерційних приміщеннях. Такі системи дозволяють автоматизувати управління освітленням, мікрокліматом, системами безпеки, електропостачанням, мультимедійними пристроями та іншими компонентами в автоматичному або напівавтоматичному режимі. Завдяки інтеграції сучасних технологій, «розумний дім» стає не лише зручним інструментом для підвищення комфорту, але й ефективним рішенням для оптимізації ресурсів і забезпечення безпеки.

Основна мета впровадження технологій «розумного дому» полягає в підвищенні рівня комфорту, енергоефективності та безпеки, а також у мінімізації необхідності людського втручання в рутинні процеси. Наприклад, системи можуть автоматично регулювати температуру в приміщенні залежно від погодних умов, вимикати освітлення в порожніх кімнатах або надсилати сповіщення про несанкціонований доступ. Крім того, «розумний дім» сприяє економії електроенергії, що є важливим аспектом у контексті зростання цін на енергоносії та турботи про екологію [1]. Технології «розумного дому» стрімко розвиваються, стаючи дедалі доступнішими завдяки поширенню недорогих одноплатних комп’ютерів, таких як Raspberry Pi, широкого асортименту сенсорів, а також відкритого програмного забезпечення, яке дозволяє створювати гнучкі та масштабовані рішення.

Однією з найперспективніших платформ для реалізації проєктів «розумного дому» є WebIOPi – вебфреймворк, який забезпечує зручне управління електронними пристроями через веббраузер. Цей інструмент дозволяє створювати повноцінні вебінтерфейси для дистанційного керування сенсорами, виконавчими пристроями, а також для моніторингу фізичних параметрів середовища, таких як температура, вологість чи рівень освітленості. WebIOPi базується на мові програмування Python і добре працює на одноплатному комп’ютері Raspberry Pi, який завдяки своїм компактним розмірам і достатнім обчислювальним можливостям ідеально підходить для реалізації невеликих IoT-систем.

WebIOPi підтримує прямий доступ до портів GPIO (General Purpose Input/Output) Raspberry Pi, що дозволяє програмно зчитувати дані з датчиків або надсилати команди до виконавчих елементів, таких як реле, світлодіоди, зумери чи сервоприводи. Це відкриває широкі можливості для автоматизації: наприклад, увімкнення освітлення при виявленні руху, активація сигналізації у разі спрацювання датчика безпеки або автоматичне керування системами вентиляції залежно від рівня вологості. Завдяки зручному REST API, WebIOPi дозволяє досить швидко створювати власні клієнтські інтерфейси без глибоких знань у веброзробці, що робить його доступним навіть для початківців. Крім того, WebIOPi забезпечує віддалене керування GPIO-пінами Raspberry Pi через Інтернет, що значно спрощує та підвищує ефективність дистанційного управління побутовими та промисловими пристроями. Користувачі можуть налаштовувати вебінтерфейс за допомогою модифікацій CSS або створювати власні вебдодатки, використовуючи REST API. Встановлення WebIOPi є простим і передбачає виконання кількох команд, що робить цей фреймворк зручним і доступним для реалізації систем автоматизації в розумних будинках чи промислових об’єктах [2].

Розробка вебінтерфейсу для системи «розумного дому» з використанням WebIOPi базується на стандартних вебтехнологіях, таких як HTML, CSS і JavaScript [3]. Це забезпечує створення легких і адаптивних вебсторінок, які коректно відображаються на різних пристроях – від настільних комп’ютерів до смартфонів і планшетів. Такий підхід дозволяє користувачам зручно взаємодіяти з системою з будь-якої точки світу, де є доступ до Інтернету. Наприклад, вебінтерфейс може відображати поточні показники температури і вологості, надавати можливість увімкнення чи вимкнення освітлення за допомогою кнопок або повзунків, а також підтримувати функції авторизації для захисту від несанкціонованого доступу.

Типова архітектура системи «розумного дому» включає кілька ключових компонентів. Центральним вузлом управління виступає Raspberry Pi, який обробляє дані від підключених сенсорів (температури, вологості, руху, освітленості, звуку тощо) і керує виконавчими пристроями (реле, сервоприводи, світлодіоди, зумери). Взаємодія між компонентами може відбуватися через дротові (наприклад, через GPIO) або бездротові протоколи, такі як Wi-Fi, Bluetooth або Zigbee. Користувацький вебінтерфейс, побудований на основі WebIOPi, забезпечує зручний доступ до всіх функцій системи в режимі реального часу. На рис. 1 подано приклад такого вебінтерфейсу, який дозволяє користувачу вмикати або вимикати окремі модулі, налаштовувати параметри середовища або переглядати актуальні дані з сенсорів. Інтерфейс працює без затримок, що дає змогу оперативно реагувати на зміни в системі.

GPIO #	GPIO Description	Status	Action	Edit
4			Turn On	Edit
17	Red LED		Turn On	Edit
18			Turn On	Edit
21	Green LED		Turn On	Edit
22			Turn On	Edit
23			Turn On	Edit
24			Turn On	Edit
25			Turn On	Edit

Рис. 1 Вебінтерфейс WebIOPi для керування елементами системи «розумного дому»

На рис. 2 зображено типову схему підключення сенсорів і виконавчих пристроїв до Raspberry Pi. Ця схема демонструє модульність і гнучкість системи: завдяки простим з'єднанням і підтримці бібліотек Python, підключення нових компонентів, таких як додаткові сенсори чи модулі не вимагає складного програмування.

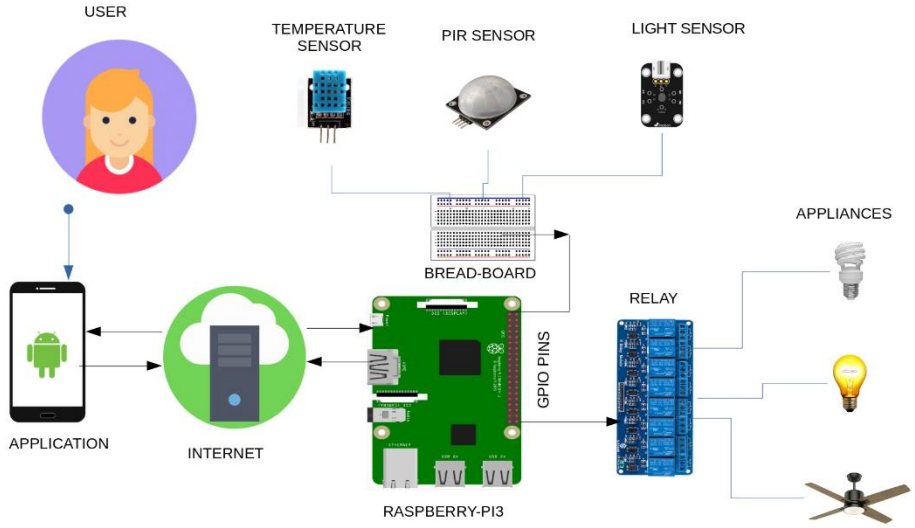

Рис. 2 Схематичне підключення сенсорів і виконавчих пристроїв до Raspberry Pi

Однією з ключових переваг WebIOPi є можливість масштабування системи. При потребі до системи можна додати нові функціональні модулі, такі як розпізнавання облич для автоматичного розблокування дверей, голосове керування через інтеграцію з голосовими асистентами, підтримка мобільних застосунків для зручного доступу або взаємодія з хмарними сервісами для зберігання даних і аналізу.

Таким чином, WebIOPi є потужним, доступним і гнучким інструментом для створення веборієнтованих систем автоматизації. Його використання дозволяє швидко розгортати функціональні рішення для моніторингу та керування середовищем, які можна легко адаптувати під конкретні потреби користувача. У поєднанні з Raspberry Pi, широким вибором сенсорів і активною підтримкою спільноти, WebIOPi відкриває можливості для реалізації інноваційних систем «розумного дому». Легкість використання, відкрита архітектура, гнучкість налаштувань і доступність роблять цей фреймворк ідеальним вибором як для ентузіастів, так і для професійних розробників, які прагнуть створювати сучасні автоматизовані рішення без значних фінансових вкладень.

Список використаних джерел:

1. Chakraborty, A., Islam, M., Shahriyar, F., Islam, Sh., Zaman, H. U., Hasan, M. Smart Home System: A Comprehensive Review. Journal of Electrical and Computer Engineering. 2023. С. 1-30. DOI: 10.1155/2023/7616683.

2. Kalyani, M. N. L., Sree Sita Mahalakshmi, V., Vijaya Ramavardhan, A., Gowri, Y. Smart Home Security And Automated System Using Raspberry Pi And Wireless Device. International Journal of Computer Science & Mechatronics. 2017. Т. 3, № 2. С. 34-38. ISSN 2455-1910.

3. The Raspberry Pi Internet of Things Toolkit - Now in two flavors. URL: https://webiopi.trouch.com/ (дата звернення: 15.04.2025).

ДОДАТОК Б

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Розумний дім – WebIOPi</title>
  <link rel="stylesheet" href="styles.css">
  <script src="/webiopi.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/chart.js@4.4.3/dist/chart.umd.min.js"></script>
  <script src="script.js" defer></script>
</head>
<body>
  <header>
    <h1>Центр керування розумним домом</h1>
  </header>
  <main class="dashboard">
    <section class="card temperature-card">
      <h2>Температура і вологість</h2>
      <p class="sensor-data">  Температура: <span id="temp">--</span> °C</p>
      <p class="sensor-data">● Вологість: <span id="humidity">--</span> %</p>
    </section>
    <section class="card dht-graph-card">
      <h3>Графік температури та вологості</h3>
      <canvas id="dhtChart"></canvas>
    </section>
    <section class="card led-control-card">
      <h2>Керування LED</h2>
      <div class="led-buttons">
        <button class="btn led-control" data-macro="toggle_led">Перемкнути LED</button>
        <button class="btn led-control red" data-macro="set_red">Червоний</button>
        <button class="btn led-control green" data-macro="set_green">Зелений</button>
        <button class="btn led-control blue" data-macro="set_blue">Синій</button>
        <button class="btn led-control trans" data-macro="start_transition">▶ Переливання</button>
        <button class="btn led-control trans" data-macro="stop_transition">✕ Переливання</button>
      </div>
      <p>Поточний стан: <span id="ledState">Off</span></p>
      <p>Поточний режим: <span id="ledMode">N/A</span></p>
    </section>
    <section class="card rfid-card">
      <h2>RFID доступ</h2>
      <p id="rfidStatus">  Очікування...</p>
      <p>Статус мелодії: <span id="melodyStatus">Не відтворюється</span></p>
    </section>
    <section class="card notification-card">
      <h2>Звукові сповіщення</h2>
      <div class="sound-buttons">
        <button id="triggerBuzzerBtn" class="btn btn-primary">🔊 Зумер</button>
        <button id="stopBuzzerBtn" class="btn btn-danger">  Зумер</button>
      </div>
      <p>Поточна мелодія: <span id="buzzerMelody">Немає</span></p>
    </section>
    <section class="card sound-sensor-card">
      <h2>Датчик звуку</h2>
      <p>Статус звуку: <span id="soundStatus">Немає звуку</span></p>
    </section>
    <section class="card clock-card">
      <h2>Годинник реального часу</h2>
      <p>🕒 Поточний час: <span id="clock">--:--:--</span></p>
      <button id="refreshClockBtn" class="btn btn-secondary">🔄 Оновити</button>
    </section>
    <section class="card author-card">
      <h2>Кваліфікаційна робота</h2>
      <h3>Моніторинг і управління системою «Розумний дім» на базі WebIOPi</h3>
      <p>Виконав: Кравчук Іван, здобувач вищої освіти групи ПМ-41</p>
    </section>
  </main>
  <footer>
    <p>© 2025 Розумний дім на базі WebIOPi. Виконав: Кравчук Іван, здобувач вищої освіти групи ПМ-41</p>
  </footer>
</body>
</html>
```

```
* {
  box-sizing: border-box;
}
```

```
body {
  font-family: 'Roboto', 'Helvetica', 'Arial', sans-serif;
  background: linear-gradient(to bottom, #e0eafc, #cfdef3);
  margin: 0;
  padding: 0;
}

header, footer {
  text-align: center;
  background: linear-gradient(90deg, #4a90e2, #357abd);
  color: white;
  padding: 1.5rem 0;
  box-shadow: 0 4px 8px rgba(0,0,0,0.2);
}

h1, h2, h3 {
  margin: 0;
}

h1 {
  font-size: 2rem;
}

.dashboard {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
  gap: 1.5rem;
  padding: 2rem;
}

.card {
  background: white;
  border-radius: 12px;
  padding: 1.5rem;
  box-shadow: 0 6px 12px rgba(0,0,0,0.15);
  transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.card:hover {
  transform: translateY(-5px);
  box-shadow: 0 8px 16px rgba(0,0,0,0.2);
}

.btn {
  padding: 12px 20px;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-size: 1rem;
  font-weight: 500;
  display: inline-flex;
  align-items: center;
  gap: 0.5rem;
  transition: background-color 0.3s ease, transform 0.2s ease, box-shadow 0.2s ease;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  white-space: nowrap;
  min-width: 140px;
}

.btn:hover {
  transform: translateY(-2px);
  box-shadow: 0 4px 8px rgba(0,0,0,0.2);
}

.btn-primary {
  background: linear-gradient(90deg, #007bff, #0056b3);
  color: white;
}

.btn-primary:hover {
  background: linear-gradient(90deg, #0056b3, #003087);
}

.btn-danger {
  background: linear-gradient(90deg, #dc3545, #c82333);
  color: white;
}
```

```
.btn-danger:hover {
  background: linear-gradient(90deg, #c82333, #a71d2a);
}

.btn-secondary {
  background: linear-gradient(90deg, #6c757d, #5a6268);
  color: white;
}

.btn-secondary:hover {
  background: linear-gradient(90deg, #5a6268, #4a4e53);
}

.led-buttons {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(140px, 1fr));
  gap: 1rem;
}

.sound-buttons {
  display: flex;
  justify-content: space-around;
  flex-wrap: wrap;
  gap: 1rem;
  margin-bottom: 1rem;
}

.led-control {
  font-size: 0.9rem;
  padding: 10px 16px;
}

.led-control.red {
  background: linear-gradient(90deg, #ff4444, #cc0000);
  color: white;
}

.led-control.red:hover {
  background: linear-gradient(90deg, #cc0000, #990000);
}

.led-control.green {
  background: linear-gradient(90deg, #00cc00, #009900);
  color: white;
}

.led-control.green:hover {
  background: linear-gradient(90deg, #009900, #006600);
}

.led-control.blue {
  background: linear-gradient(90deg, #007bff, #0056b3);
  color: white;
}

.led-control.blue:hover {
  background: linear-gradient(90deg, #0056b3, #003087);
}

.led-control.trans {
  background: linear-gradient(90deg, #ff4444, #00cc00, #007bff);
  color: white;
}

.led-control.trans:hover {
  background: linear-gradient(90deg, #cc0000, #009900, #0056b3);
}

.temperature-card .sensor-data, .sound-sensor-card p, .clock-card p {
  font-size: 1.5rem;
  margin: 0.5rem 0;
  color: #333;
  text-shadow: 1px 1px 2px rgba(0,0,0,0.1);
}

.temperature-card .sensor-data span, .sound-sensor-card p span, .clock-card p span {
  font-weight: bold;
  color: #4a90e2;
}
```

```
#rfidStatus, #melodyStatus, #buzzerMelody {
  font-size: 1rem;
  margin: 0.5rem 0;
  color: #333;
}

#melodyStatus span, #buzzerMelody span {
  font-weight: bold;
  color: #4a90e2;
}

#dhtChart {
  max-height: 200px;
}

.author-card {
  background: linear-gradient(135deg, #f0f8ff, #e6f0fa);
  border: 2px solid #4a90e2;
  box-shadow: 0 6px 12px rgba(0,0,0,0.2);
  text-align: center;
  padding: 2rem;
}

.author-card h2 {
  font-size: 1.6rem;
  color: #4a90e2;
  margin: 0.5rem 0;
}

.author-card h3 {
  font-size: 1.3rem;
  color: #333;
  margin: 0.5rem 0;
}

.author-card p {
  font-size: 1.1rem;
  color: #333;
  margin: 0.5rem 0;
}

footer {
  font-size: 0.9rem;
  padding: 1rem;
  text-align: center;
  background: linear-gradient(90deg, #4a90e2, #357abd);
  color: white;
  box-shadow: 0 -4px 8px rgba(0,0,0,0.2);
}

@media (max-width: 768px) {
  h1 { font-size: 1.5rem; }
  .dashboard { grid-template-columns: 1fr; padding: 1rem; gap: 1rem; }
  .card { padding: 1rem; }
  .temperature-card .sensor-data, .sound-sensor-card p, .clock-card p { font-size: 1.2rem; }
  .btn { padding: 10px 16px; font-size: 0.9rem; min-width: 120px; }
  .led-control { font-size: 0.8rem; padding: 8px 12px; }
  .sound-buttons { flex-direction: column; align-items: center; }
  .sound-buttons .btn { margin: 0.5rem 0; width: 100%; }
  .author-card { padding: 1.5rem; }
  .author-card h2 { font-size: 1.4rem; }
  .author-card h3 { font-size: 1.1rem; }
  .author-card p { font-size: 1rem; }
}

@media (max-width: 480px) {
  h1 { font-size: 1.2rem; }
  .temperature-card .sensor-data, .sound-sensor-card p, .clock-card p { font-size: 1rem; }
  .btn { padding: 8px 12px; font-size: 0.8rem; min-width: 100px; }
  .led-control { font-size: 0.7rem; }
  .author-card h2 { font-size: 1.2rem; }
  .author-card h3 { font-size: 1rem; }
  .author-card p { font-size: 0.9rem; }
}
```

ДОДАТОК В

```
webiopi().ready(function() {
    const tempElement = document.getElementById("temp");
    const humidityElement = document.getElementById("humidity");
    const rfidStatusElement = document.getElementById("rfidStatus");
    const melodyStatusElement = document.getElementById("melodyStatus");
    const triggerBuzzerBtn = document.getElementById("triggerBuzzerBtn");
    const stopBuzzerBtn = document.getElementById("stopBuzzerBtn");
    const ledStateElement = document.getElementById("ledState");
    const ledModeElement = document.getElementById("ledMode");
    const ledControlButtons = document.querySelectorAll(".led-control");
    const soundStatusElement = document.getElementById("soundStatus");
    const clockElement = document.getElementById("clock");
    const refreshClockBtn = document.getElementById("refreshClockBtn");
    let soundDetectedTime = null;
    let previousSoundStatus = "not detected";
    let dhtChart = null;

    function refreshDHT() {
        webiopi().callMacro("getTemperatureHumidity", [], function(macro, args, response) {
            if (response === "Error;Error") {
                tempElement.textContent = humidityElement.textContent = "Помилка";
            } else {
                const [temperature, humidity] = response.split(";");
                tempElement.textContent = temperature;
                humidityElement.textContent = humidity;
                updateDHTChart();
            }
        });
    }

    function updateDHTChart() {
        webiopi().callMacro("getDHTHistory", [], function(macro, args, response) {
            try {
                const history = JSON.parse(response);
                if (history.length === 0) return; // Якщо даних немає, виходимо

                if (!dhtChart) {
                    const ctx = document.getElementById('dhtChart').getContext('2d');
                    dhtChart = new Chart(ctx, {
                        type: 'line',
                        data: {
                            labels: history.map(item => item.time), // Формат часу: HH:MM
                            datasets: [
                                {
                                    label: 'Температура (°C)',
                                    data: history.map(item => item.temperature),
                                    borderColor: '#ff0000',
                                    backgroundColor: 'rgba(255, 0, 0, 0.1)',
                                    fill: false,
                                    tension: 0.1
                                },
                                {
                                    label: 'Вологість (%)',
                                    data: history.map(item => item.humidity),
                                    borderColor: '#007bff',
                                    backgroundColor: 'rgba(0, 123, 255, 0.1)',
                                    fill: false,
                                    tension: 0.1
                                }
                            ]
                        },
                        options: {
                            responsive: true,
                            maintainAspectRatio: false,
                            scales: {
                                x: {
                                    display: true,
                                    title: {
                                        display: true,
                                        text: 'Час (год:хв)',
                                        font: { size: 14 }
                                    },
                                    ticks: {
                                        maxRotation: 45,
                                        minRotation: 45,
                                        maxTicksLimit: 10
                                    }
```

```
                    },
                    y: {
                        display: true,
                        title: {
                            display: true,
                            text: 'Значення',
                            font: { size: 14 }
                        }
                    }
                },
                plugins: {
                    legend: {
                        labels: {
                            font: { size: 12 }
                        }
                    }
                }
            }
        });
    } else {
        dhtChart.data.labels = history.map(item => item.time);
        dhtChart.data.datasets[0].data = history.map(item => item.temperature);
        dhtChart.data.datasets[1].data = history.map(item => item.humidity);
        dhtChart.update();
    }
  } catch (error) {
    console.error("Помилка при оновленні графіку:", error);
  }
 });
}

function refreshSoundStatus() {
 webiopi().callMacro("getSoundStatus", [], function(macro, args, response) {
  if (response === "detected") {
    soundDetectedTime = Date.now();
    soundStatusElement.textContent = "Звук виявлено";
  } else {
    if (soundDetectedTime && (Date.now() - soundDetectedTime > 5000)) {
      soundStatusElement.textContent = "Немає звуку";
      soundDetectedTime = null;
    }
  }
  previousSoundStatus = response;
 });
}

function refreshRFIDStatus() {
 webiopi().callMacro("get_rfid_status", [], function(macro, args, response) {
  try {
    const status = JSON.parse(response);
    rfidStatusElement.textContent = status.status;
    melodyStatusElement.textContent = status.melody;
  } catch (error) {}
 });
}

function updateLEDStatus() {
 webiopi().callMacro("get_led_status", [], function(macro, args, response) {
  try {
    const status = JSON.parse(response);
    ledStateElement.textContent = status.led_on ? "On" : "Off";
    ledModeElement.textContent = status.led_on
      ? status.mode === "static" ? `Static - ${status.current_color}` : "Transitioning"
      : "N/A";
  } catch (error) {}
 });
}

function callMacroAndUpdateStatus(macroName) {
 if (!macroName) return;
 webiopi().callMacro(macroName, [], function(macro, args, response) {
  updateLEDStatus();
 });
}

function refreshClock() {
 webiopi().callMacro("getCurrentTime", [], function(macro, args, response) {
  clockElement.textContent = response;
 });
```

```
    }

    // Ініціалізація графіку при завантаженні
    if (typeof Chart !== 'undefined') {
        updateDHTChart();
    } else {
        console.error("Chart.js не завантажено!");
    }

    ledControlButtons.forEach(button => {
        button.addEventListener("click", () => {
            const macro = button.getAttribute("data-macro");
            if (macro) callMacroAndUpdateStatus(macro);
        });
    });

    triggerBuzzerBtn.addEventListener("click", () => {
        webiopi().callMacro("buzzer", [], function(macro, args, response) {
            alert(response === "Buzzer OK" ? "Зумер активовано" : "Помилка при вмиканні зумера");
        });
    });

    stopBuzzerBtn.addEventListener("click", () => {
        webiopi().callMacro("stop_buzzer", [], function(macro, args, response) {
            alert(response === "Buzzer stopped" ? "Зумер зупинено" : "Помилка зупинки зумера");
        });
    });

    refreshClockBtn.addEventListener("click", refreshClock);

    // Оновлення даних
    refreshDHT();
    setInterval(refreshDHT, 60000); // Оновлення раз на хвилину (60 секунд)
    updateLEDStatus();
    refreshSoundStatus();
    setInterval(refreshSoundStatus, 500);
    refreshRFIDStatus();
    setInterval(refreshRFIDStatus, 1000);
    refreshClock();
    setInterval(refreshClock, 60000);
});

# # -*- coding: utf-8 -*-
import webiopi
import json
import time
import RPi.GPIO as GPIO
import Adafruit_DHT
from mfrc522 import SimpleMFRC522
from datetime import datetime
import threading

GPIO.setwarnings(False)

# ---- Піни ----
DHT_PIN = 14                # GPIO4 для DHT11
RGB_PINS = {"R": 17, "G": 27, "B": 22}  # GPIO17, GPIO27, GPIO22 для KY-016
BUZZER_PIN = 26             # GPIO26 для зумера
SOUND_PIN = 6               # GPIO6 для LM393 (DO, фізичний пін 31)

# ---- Ініціалізація ----
sensor = Adafruit_DHT.DHT11
led_on = False
mode = "static"
current_color = "blue"
transition_running = False
rfid_reader = None
dht_history = []            # Історія для температури та вологості
buzzer_running = False
last_rfid_status = "    Очікування..."
last_melody_status = "Не відтворюється"
sound_detected = False
sound_monitor_running = True
sound_thread = None

# Частоти нот для мелодій
CL = [0, 131, 147, 165, 175, 196, 211, 248]
CM = [0, 262, 294, 330, 350, 393, 441, 495]
CH = [0, 525, 589, 661, 700, 786, 882, 990]
```

```
# Мелодії
allowed_melody = [CM[1], CM[3], CM[5]]
allowed_beat = [0.5, 0.5, 0.5]
denied_melody = [CM[5], CM[1]]
denied_beat = [0.3, 0.3]
def monitor_sound():
    global sound_detected, sound_monitor_running
    def sound_callback(channel):
        global sound_detected
        if GPIO.input(channel) == GPIO.LOW:
            sound_detected = True
            time.sleep(3)
            sound_detected = False
    GPIO.setup(SOUND_PIN, GPIO.IN)
    GPIO.add_event_detect(SOUND_PIN, GPIO.BOTH, callback=sound_callback, bouncetime=300)
    while sound_monitor_running:
        time.sleep(0.1)
def play_melody(song, beat):
    global buzzer_running
    buzzer_running = True
    try:
        GPIO.output(BUZZER_PIN, GPIO.LOW)
        for i in range(len(song)):
            if not buzzer_running:
                break
            if song[i] == 0:
                GPIO.output(BUZZER_PIN, GPIO.LOW)
                time.sleep(beat[i])
            else:
                duration = beat[i]
                period = 1.0 / song[i]
                cycles = int(duration / period)
                for _ in range(cycles):
                    if not buzzer_running:
                        break
                    GPIO.output(BUZZER_PIN, GPIO.HIGH)
                    time.sleep(period / 2)
                    GPIO.output(BUZZER_PIN, GPIO.LOW)
                    time.sleep(period / 2)
            time.sleep(0.05)
        GPIO.output(BUZZER_PIN, GPIO.LOW)
    except:
        pass
    finally:
        buzzer_running = False
def setup():
    global rfid_reader, pwmR, pwmG, pwmB, sound_thread, sound_monitor_running
    GPIO.setmode(GPIO.BCM)
    for pin in RGB_PINS.values():
        try:
            GPIO.setup(pin, GPIO.OUT)
            GPIO.output(pin, GPIO.LOW)
        except:
            pass
    try:
        GPIO.setup(BUZZER_PIN, GPIO.OUT)
        GPIO.output(BUZZER_PIN, GPIO.LOW)
    except:
        pass
    try:
        pwmR = GPIO.PWM(RGB_PINS["R"], 100)
        pwmG = GPIO.PWM(RGB_PINS["G"], 100)
        pwmB = GPIO.PWM(RGB_PINS["B"], 100)
        pwmR.start(0)
        pwmG.start(0)
        pwmB.start(0)
    except:
        pass
    try:
        rfid_reader = SimpleMFRC522()
    except:
        pass
    try:
        sound_monitor_running = True
        sound_thread = threading.Thread(target=monitor_sound)
        sound_thread.setDaemon(True)
        sound_thread.start()
    except:
        pass
```

```
def destroy():
    global transition_running, buzzer_running, sound_monitor_running
    transition_running = False
    buzzer_running = False
    sound_monitor_running = False
    try:
        if sound_thread:
            sound_thread.join(timeout=1)
        pwmR.stop()
        pwmG.stop()
        pwmB.stop()
        GPIO.cleanup()
    except:
        pass
def set_color(color):
    try:
        if color == "red":
            pwmR.ChangeDutyCycle(100)
            pwmG.ChangeDutyCycle(0)
            pwmB.ChangeDutyCycle(0)
        elif color == "green":
            pwmR.ChangeDutyCycle(0)
            pwmG.ChangeDutyCycle(100)
            pwmB.ChangeDutyCycle(0)
        elif color == "blue":
            pwmR.ChangeDutyCycle(0)
            pwmG.ChangeDutyCycle(0)
            pwmB.ChangeDutyCycle(100)
        else:
            pwmR.ChangeDutyCycle(0)
            pwmG.ChangeDutyCycle(0)
            pwmB.ChangeDutyCycle(0)
    except:
        pass
@webiopi.macro
def getTemperatureHumidity():
    try:
        humidity, temperature = Adafruit_DHT.read_retry(sensor, DHT_PIN)
        if humidity is not None and temperature is not None:
            current_time = datetime.now().strftime("%H:%M")  # Формат HH:MM
            dht_history.append({"time": current_time, "temperature": float("%.1f" % temperature), "humidity": float("%.1f" % humidity)})
            if len(dht_history) > 360:  # Зберігаємо 360 записів (6 годин при оновленні раз на хвилину)
                dht_history.pop(0)
            return "%.1f;%.1f" % (temperature, humidity)
        else:
            return "Error;Error"
    except:
        return "Error;Error"
@webiopi.macro
def getDHTHistory():
    try:
        return json.dumps(dht_history)
    except:
        return "[]"
@webiopi.macro
def getSoundStatus():
    global sound_detected
    try:
        return "detected" if sound_detected else "not detected"
    except:
        return "error"
@webiopi.macro
def toggle_led():
    global led_on, mode, current_color
    try:
        if led_on:
            set_color("off")
            led_on = False
            mode = "static"
            return "LED off"
        else:
            led_on = True
            set_color(current_color)
            return "LED on"
    except:
        return "Error"
@webiopi.macro
def set_red():
    global mode, current_color
```

```
    try:
        mode = "static"
        current_color = "red"
        if led_on:
            set_color("red")
        return "Set to red"
    except:
        return "Error"
@webiopi.macro
def set_green():
    global mode, current_color
    try:
        mode = "static"
        current_color = "green"
        if led_on:
            set_color("green")
        return "Set to green"
    except:
        return "Error"
@webiopi.macro
def set_blue():
    global mode, current_color
    try:
        mode = "static"
        current_color = "blue"
        if led_on:
            set_color("blue")
        return "Set to blue"
    except:
        return "Error"
@webiopi.macro
def start_transition():
    global mode, transition_running
    try:
        if mode != "transition":
            mode = "transition"
            transition_running = True
            for x in range(1, 101):
                if not transition_running:
                    break
                pwmG.ChangeDutyCycle(x)
                time.sleep(0.05)
            for x in range(1, 101):
                if not transition_running:
                    break
                pwmR.ChangeDutyCycle(101-x)
                time.sleep(0.025)
            for x in range(1, 101):
                if not transition_running:
                    break
                pwmG.ChangeDutyCycle(101-x)
                pwmB.ChangeDutyCycle(x)
                time.sleep(0.025)
            for x in range(1, 101):
                if not transition_running:
                    break
                pwmB.ChangeDutyCycle(x)
                time.sleep(0.025)
            transition_running = False
            mode = "static"
            if led_on:
                set_color(current_color)
            return "Transition started"
        else:
            return "Already in transition"
    except:
        return "Error"
@webiopi.macro
def stop_transition():
    global mode, transition_running
    try:
        if mode == "transition":
            mode = "static"
            transition_running = False
            if led_on:
                set_color(current_color)
            return "Transition stopped"
        else:
            return "Not in transition"
```

```
    except:
        return "Error"
@webiopi.macro
def get_led_status():
    try:
        status = {
            "led_on": led_on,
            "mode": mode,
            "current_color": current_color
        }
        return json.dumps(status)
    except:
        return "{}"
@webiopi.macro
def get_rfid_status():
    global rfid_reader, last_rfid_status, last_melody_status
    try:
        if rfid_reader is None:
            return json.dumps({"status": "Помилка", "melody": "Помилка"})
        id, text = rfid_reader.read_no_block()
        if id is not None:
            uid = str(id)
            if uid == "923474830100":
                thread = threading.Thread(target=play_melody, args=(allowed_melody, allowed_beat))
                thread.setDaemon(True)
                thread.start()
                last_rfid_status = "    Доступ дозволено"
                last_melody_status = "Мелодія дозволу"
            else:
                thread = threading.Thread(target=play_melody, args=(denied_melody, denied_beat))
                thread.setDaemon(True)
                thread.start()
                last_rfid_status = "● Доступ заборонено"
                last_melody_status = "Мелодія заборони"
        return json.dumps({"status": last_rfid_status, "melody": last_melody_status})
    except:
        return json.dumps({"status": "Помилка", "melody": "Помилка"})
@webiopi.macro
def buzzer():
    try:
        thread = threading.Thread(target=play_melody, args=(allowed_melody, allowed_beat))
        thread.setDaemon(True)
        thread.start()
        return "Buzzer OK"
    except:
        return "Error"
@webiopi.macro
def stop_buzzer():
    global buzzer_running
    try:
        if buzzer_running:
            buzzer_running = False
            GPIO.output(BUZZER_PIN, GPIO.LOW)
            return "Buzzer stopped"
        else:
            return "Buzzer not running"
    except:
        return "Error"
@webiopi.macro
def getCurrentTime():
    try:
        now = datetime.now()
        return now.strftime("%H:%M:%S")
    except:
        return "Error"
if __name__ == "__main__":
    try:
        setup()
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        pass
    finally:
        destroy()
```