МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп’ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувачка кафедри,
д-р техн. наук, проф.

_____Ірина ЖУРАВСЬКА

« __ » _____ 202__ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Система керування розумним будинком на базі ESP32

Спеціальність 123 Комп’ютерна інженерія

Освітня програма «Комп’ютерна інженерія»

Здобувач _____Микола АВШЕНЮК
підпис

«__» _____ 202__ р.

Керівник PhD, ст. викладач каф. КІ _____ Євген ДАРНАПУК
підпис
«__» _____ 202__ р.

Миколаїв – 2025

Факультет	Комп’ютерних наук
Кафедра	Комп’ютерної інженерії
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	123 Комп’ютерна інженерія
Освітня програма	Комп’ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри комп’ютерної інженерії
_____ Ірина ЖУРАВСЬКА
«_____» _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача
Авшенюку Миколі Миколайовичу
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Система керування розумним будинком на базі ESP32

Затверджена наказом по ЧНУ ім. Петра Могили від 30.10.2024 № 297.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом кваліфікаційної роботи є розроблена апаратна частина на базі ESP32 та програмна частина для забезпечення функціонування системи керування розумним будинком.

4. Перелік питань, що підлягають розробці:
1) проаналізувати існуючі рішення систем «розумного будинку», їх керування, функціональність та переваги;
2) дослідити та вивчити технічні характеристики й особливості пристроїв для керування системами «розумного будинку», враховуючи можливості інтеграції;
3) здійснити проєктування та розробку апаратної та програмної частин загальної системи для забезпечення керування «розумним будинком»;
4) дослідити можливості подальшого удосконалення системи керування.

5. Перелік графічних матеріалів
Презентація

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Євген ДАРНАПУК
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Микола АВШЕНЮК
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «_____» _____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної бакалаврської роботи

Тема: Система керування розумним будинком на базі ESP32

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	30.10.2024	30.10.2024	Виконано
2	Огляд літератури за темою роботи	01.03.2025	14.03.2025	Виконано
3	Складання календарного плану КБР	15.03.2025	15.03.2025	Виконано
4	Аналіз предметної області	16.03.2025	30.03.2025	Виконано
5	Розробка проєктних рішень	01.04.2025	18.04.2025	Виконано
6	Моделювання та конструювання АПЗ	19.04.2025	01.05.2025	Виконано
7	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування	02.05.2025	25.05.2025	Виконано
8	Відгук керівника КБР	26.05.2025	11.06.2025	Виконано
9	Оформлення КБР та презентації	10.05.2025	03.06.2025	Виконано
10	Попередній захист	21.05.2025	04.06.2025	Виконано
11	Рецензування	06.06.2025	13.06.2025	Виконано
12	Завершення оформлення КБР та презентації	14.06.2025	17.06.2025	Виконано
13	Захист кваліфікаційної бакалаврської роботи	24.06.2025	24.06.2025	Виконано

Керівник роботи _____ Євген ДАРНАПУК

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Микола АВШЕНЮК

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

«____» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«Система керування розумним будинком на базі ESP32»
Студент 405 гр.: Авшенюк Микола Миколайович
Керівник: PhD, ст. викладач Дарнапук Євген Сергійович

Актуальність напрямку кваліфікаційної бакалаврської роботи (КБР) полягає у тому, що нині системи керування будинком дозволяють здійснювати автономне керування власним будинком з будь-якого місця, потенційно зменшуючи споживання електроенергії та підвищуючи безпеку, водночас забезпечуючи комфортніші умови проживання.

У даній КБР **об'єктом** дослідження є системи керування «розумними» будинками та автоматизації. **Предметом** дослідження є проведення аналізу характеристик, функціональних можливостей, переваг, недоліків, наявних аналогів систем «розумного будинку», включаючи дослідження та аналіз існуючих рішень та моделей.

Метою кваліфікаційної роботи є дослідження технологій і функціональних можливостей систем «розумного будинку», розробка функціональної системи керування «розумним будинком», побудованої на основі мікроконтролера ESP32 з можливістю бездротового моніторингу та керування.

Пояснювальна записка складається зі вступу, трьох розділів основного матеріалу, висновків та 2 додатків.

У першому розділі КБР здійснено огляд наукової та публіцистичної літератури відповідного напрямку, включаючи тенденції використання систем «розумного будинку», інтеграцію та потенціал їх впровадження у майбутньому. У другому розділі розглянуто аспекти функціонального моделювання, а саме визначено основні функції системи, побудовано загальний алгоритм АПЗ. У третьому розділі КБР описано етапи розробки апаратної та програмної частин системи керування «розумним» будинком на базі ESP32. Робота також містить перелік джерел посилання та додатки.

Результатом проведеного дослідження в рамках кваліфікаційної роботи бакалавра, є розроблене АПЗ керування системою «розумного будинку» на базі мікроконтролера ESP32.

Кваліфікаційна бакалаврська робота викладена на 73 сторінках (без додатків), містить 3 розділи, 35 рисунків, 5 таблиць та 20 джерел посилання, 2 додатки.

Ключові слова: *розумний будинок, ESP32, датчики, контроль параметрів, керування системою, Інтернет Речей, freeRTOS.*

ABSTRACT
of the Bachelor's Thesis
"Smart home control system based on ESP32"
Student: Mykola Avsheniuk
Supervisor: PhD, Senior Lecturer Yevhen Darnapuk

The relevance of the bachelor's thesis is that today's home control systems allow to control home autonomously from anywhere, potentially reducing energy consumption and increasing security, while providing a more comfortable living environment.

In this bachelor's thesis, **the object of study** is smart home control and automation systems. **The subject of the study** is to analyse the characteristics, functionality, advantages, disadvantages, and existing analogues of smart home systems, including research and analysis of existing solutions and models.

The purpose of the bachelor's thesis is to study the technologies and functionality of smart home systems to determine their potential and prospects for use, to develop a hardware and software smart home control system based on ESP32.

The bachelor's thesis includes a professional section. The explanatory note consists of an introduction, three sections of the main material, conclusions and 2 appendices.

The first chapter of the bachelor's thesis provides an analytical review of the scientific and journalistic literature in the relevant area, including trends in the use of smart home systems, integration and the overall potential for their implementation in the future. The second chapter deals with the aspects of functional modelling, namely, the main functions of the system and the general algorithm of the APP. The third chapter of the bachelor's thesis describes the stages of development of the hardware and software parts of the smart home system based on ESP32. The work also contains a list of references and appendices.

The result of the research carried out as part of the bachelor's thesis is the developed hardware and software system for controlling the smart home system based on the ESP32 microcontroller.

The paper consists of 73 pages (excluding appendices), 5 tables, 35 figures, 20 references and 2 appendices.

Keywords: *smart home, ESP32, sensors, parameter control, system management, Internet of Things, freeRTOS.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АПЗ – апаратно-програмне забезпечення
КБР – кваліфікаційна бакалаврська робота
ООП – об’єктно-орієнтовне програмування
ОС – операційна система
ОСРЧ – операційна система реального часу
ПЗ – програмне забезпечення

HTTP – Hypertext Transfer Protocol
IoT – Internet of Things
LDR – Light Dependent Resistor
MQTT – Message Queuing Telemetry Transport
SoC – System-on-Chip
SPI – Serial Peripheral Interface
UART – Universal Asynchronous Receiver/Transmitter

ВСТУП

Актуальність теми кваліфікаційної бакалаврської роботи (КБР) роботи зумовлена стрімким розвитком систем автоматизації у повсякденному житті. У сучасних умовах все більше уваги приділяється підвищенню комфорту, енергоефективності та безпеки в оселях, що призводить до зростання інтересу до систем керування «розумним будинком». Одним із важливих напрямів розвитку таких систем є інтеграція мікроконтролерів з можливостями бездротового зв’язку. Особливої популярності набуває платформа ESP32, що поєднує низьке енергоспоживання, вбудовані Wi-Fi та Bluetooth-модулі, у тому числі достатню обчислювальну потужність для реалізації складних функціональних сценаріїв. Впровадження таких рішень дозволить здійснювати дистанційне керування освітленням, кліматичними системами, безпекою, побутовою технікою тощо. У поєднанні з концепцією Інтернету речей (англ. Internet of Things, IoT), система керування «розумним будинком» на базі ESP32 може стати ще одним кроком у напрямі автоматизації побуту.

Об’єктом дослідження є методи та технології автоматизованого керування побутовими пристроями.

Предмет дослідження є архітектура, принципи побудови та реалізації системи керування «розумним будинком» на базі модуля ESP32 з використанням бездротових технологій зв’язку.

Метою роботи є дослідження технологій і функціональних можливостей систем «розумного будинку», розробка функціональної системи керування «розумним будинком», побудованої на основі мікроконтролера ESP32 з можливістю бездротового моніторингу та керування.

З урахуванням поставленої мети, завдання роботи є наступними:

– провести аналіз існуючих рішень у сфері систем «розумного будинку» та архітектурних особливостей їх побудови;

– дослідити технічні та програмні можливості модуля ESP32;

– розробити архітектуру системи керування з урахуванням типових побутових пристроїв;

– реалізувати апаратну частину системи «розумний будинок»;

– реалізувати програмну частину для забезпечення керування пристроями за допомогою мікроконтролера ESP32;

– здійснити тестування функціональності системи.

Таким чином, виконання даної роботи передбачає розробку як апаратної, так і програмної частини системи, включаючи вибір сенсорів і виконавчих пристроїв, побудову схеми підключення, написання коду на мові програмування C++ в середовищі Arduino IDE.

Практичне значення результатів роботи полягає у створенні універсальної, масштабованої та енергоефективної платформи для автоматизації житлових приміщень, яка може бути використана як основа для подальших досліджень або впровадження в реальні проєкти систем «розумного будинку».

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ

1.1 Актуальність теми КБР

1.1.1 Тенденції автоматизації побутових процесів

Автоматизація побутових процесів є одним із ключових напрямів розвитку сучасних інформаційних та електронних технологій, що визначають обличчя інтелектуального середовища проживання людини у XXI столітті. Системи автоматизації, що ще донедавна використовувались переважно в промисловості, впевнено переходять у сферу повсякденного життя, трансформуючи звичайні житлові приміщення в інтерактивні, керовані середовища – так звані «розумні будинки» (англ. smart home) [1].

Загальносвітова тенденція до зростання урбанізації, зростання енергоспоживання, а також потреба у підвищенні ефективності та зручності побуту стимулюють розвиток рішень, що дозволяють мінімізувати участь людини в рутинних завданнях. До таких завдань належать автоматичне керування освітленням залежно від присутності людини, підтримання заданого мікроклімату в приміщенні, контроль за роботою побутових приладів, а також відеоспостереження, сигналізація тощо.

Останніми роками стрімко розвиваються не лише окремі автоматизовані пристрої, а й інтегровані рішення, що об’єднують у собі функціональність кількох систем і забезпечують централізоване або розподілене управління. Така інтеграція сприяє більш ефективному використанню ресурсів та відкриває нові можливості для адаптивної поведінки системи залежно від умов навколишнього середовища, часу доби, сценаріїв користувача чи аналізу зібраних даних.

Додатково варто зазначити зростання ролі мобільних застосунків, хмарних сервісів, що розширюють функціональність систем і роблять їх доступними. Завдяки розвитку бездротових технологій, таких як Wi-Fi, Bluetooth, ZigBee, Z-Wave тощо, спрощується встановлення та масштабування домашніх систем автоматизації [2].

Таким чином, автоматизація побутових процесів із пасивного застосунку перетворюється на активного учасника повсякденного життя людини. Вона дозволяє не лише підвищити рівень комфорту, а й зменшити навантаження на користувача, оптимізувати витрати, підвищити безпеку та створити передумови для формування сталого, енергоефективного та інтерактивного життєвого простору.

1.1.2 Інтеграція технологій Інтернету речей у побутові системи

Інтернет речей є однією з ключових технологій, що кардинально змінює підходи до побутової автоматизації. Концепція Internet of Things (IoT) полягає у створенні єдиного інформаційного середовища, в якому фізичні пристрої можуть обмінюватися даними, взаємодіяти між собою, приймати рішення або передавати інформацію на сервери для обробки. Завдяки цьому звичайні побутові пристрої – від ламп і термостатів до холодильників і замків – перетворюються на інтелектуальні об’єкти, здатні до адаптації, навчання та віддаленого керування [3].

В основі інтеграції IoT у побутові системи лежить ідея повної взаємодії між користувачем, інфраструктурою житла та хмарними або локальними сервісами. Це дозволяє реалізувати як прості сценарії (наприклад, увімкнення світла за присутності людини), так і складні багаторівневі моделі (наприклад, оптимізацію роботи системи опалення на основі прогнозу погоди, часу доби та звичок мешканців). Технічною основою для такої інтеграції виступають датчики, виконавчі пристрої, контролери, мережеві інтерфейси та програмне забезпечення (ПЗ). Кожен пристрій в IoT-мережі має унікальний ідентифікатор, що дозволяє йому бути частиною мережі та реагувати на команди або змінювати свій стан відповідно до сценарію.

Одним з ключових факторів поширення IoT у побуті є доступність бездротових технологій зв’язку:

– Wi-Fi забезпечує високу швидкість та сумісність з більшістю домашніх роутерів;

– Bluetooth є енергоефективним варіантом для близького зв’язку з мобільними пристроями;

– ZigBee і Z-Wave є спеціалізованими протоколами з низьким енергоспоживанням для створення стабільних домашніх мереж IoT [2].

Важливою перевагою IoT-систем є можливість масштабування та адаптації під потреби конкретного користувача. Кожна система може починатися з кількох базових пристроїв і поступово розширюватися до повноцінної мережі з десятками сенсорів, керованих реле, камер спостереження та інших елементів. З погляду безпеки та збереження даних, інтеграція IoT вимагає належної реалізації протоколів шифрування, автентифікації користувачів, а також механізмів захисту від несанкціонованого доступу. Це особливо актуально у контексті «розумного будинку», оскільки такі системи контролюють доступ до приміщення, камери, системи безпеки тощо.

Таким чином, технології Інтернету речей виступають основою для побудови сучасних побутових систем, забезпечуючи підвищення ефективності, зручності, безпеки та інтелектуального управління ресурсами житла.

1.1.3 Потенціал використання модуля ESP32 у системах «розумний будинок»

Мікроконтролер ESP32 є одним із найпоширеніших рішень у сфері побудови систем «розумного будинку» завдяки своїй універсальності, високій обчислювальній потужності, енергоефективності та вбудованій підтримці бездротових інтерфейсів зв’язку. Розроблений компанією Espressif Systems [4], цей модуль вирізняється низькою вартістю, компактністю та широкими функціональними можливостями, що робить його оптимальним вибором для реалізації як окремих інтелектуальних пристроїв, так і повноцінних автоматизованих систем.

Основними перевагами ESP32 є:

– подвійне ядро Tensilica Xtensa LX6 з тактовою частотою до 240 МГц, що дозволяє виконувати декілька завдань одночасно (наприклад, обробку сигналів від сенсорів та передачу даних мережею);

– підтримка Wi-Fi та Bluetooth без потреби у зовнішніх модулях, що дозволяє реалізувати віддалене управління пристроями через мобільні застосунки, вебінтерфейси чи хмарні сервіси;

– наявність великої кількості GPIO-портів, що забезпечує легке підключення різноманітних датчиків і виконавчих механізмів (реле, мотори, сенсори температури, вологості, руху тощо);

– підтримка протоколів зв’язку, таких як MQTT, HTTP, WebSocket, I²C, SPI, UART, що дає змогу інтегрувати ESP32 у більші системи автоматизації [4].

Модуль ESP32 активно використовується як ядро для реалізації контролерів розумного освітлення, клімат-контролю, охоронних систем, управління шторами, побутовою технікою, відеоспостереженням та іншими елементами «розумного будинку». Його відкритість до програмування, сумісність з Arduino IDE, ESP-IDF та іншими середовищами розробки дає змогу як професіоналам, так і ентузіастам швидко створювати індивідуальні проєкти.

Ще одним важливим аспектом є підтримка енергозберігаючих режимів роботи, що робить ESP32 придатним для автономних пристроїв з живленням від акумуляторів. Завдяки цьому забезпечується тривала безперебійна робота пристроїв, наприклад, сенсорів руху чи контролерів мікроклімату.

ESP32 виступає ключовим апаратним елементом у проєктуванні гнучких, доступних та масштабованих систем «розумного будинку», здатних забезпечити ефективну автоматизацію житлового простору з високим рівнем інтелектуальності та зручності для користувача.

1.2 Огляд існуючих рішень систем «розумного будинку»

1.2.1 Комерційні системи: функціональні можливості та обмеження

Сучасний ринок систем «розумного будинку» представлений низкою комерційних рішень, які забезпечують автоматизацію та централізоване керування побутовими пристроями. Серед найбільш популярних екосистем можна виокремити лінійку Google Nest, Amazon Alexa, Apple HomeKit, Samsung SmartThings Hub, Xiaomi Mi Home та інші. Ці платформи зазвичай пропонують інтеграцію різних типів пристроїв – від освітлювальних приладів і термостатів до побутової техніки, відеоспостереження та охоронних систем – в єдине кероване середовище. Однією з ключових переваг таких рішень є підтримка мобільних застосунків та голосових асистентів, що забезпечує зручний інтерфейс взаємодії з користувачем, а також можливість віддаленого доступу через хмарні сервіси [5]. Крім того, комерційні системи підтримують налаштування автоматизованих сценаріїв, що дозволяє реалізувати складну логіку взаємодії пристроїв без необхідності прямого втручання користувача.

Так продукти «розумного будинку» Google Nest пропонують різні варіанти пристроїв для забезпечення керуванням будинком, у тому числі колонки Nest Hub, що можна використовувати для голосових команд і перегляду інформації.

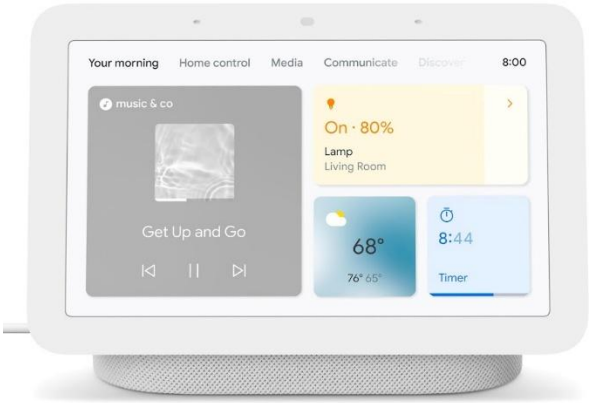

Рисунок 1.1 – Дисплей керування «розумним будинком» Google Nest Hub [6]

Інші продукти лінійки Nest включають також «розумні дисплеї» (рис. 1.1), термостати, детектори диму, чадного газу, камери безпеки та «розумні замки». Керування такою системою виконується шляхом використання голосового керування командами, наприклад, увімкнення світла, відтворення музики тощо.

Іншим комерційним прикладом є центр «розумного будинку» Samsung SmartThings Hub (рис. 1.2), що виконує підключення та керування різними пристроями в будинку, пропонуючи рішення централізованого та автоматизованого керування таким.

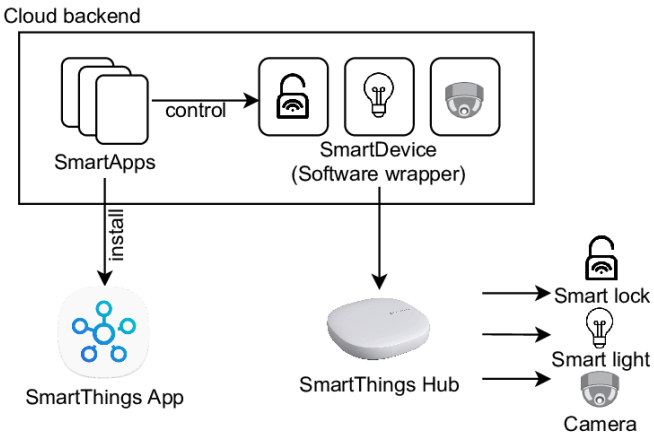

Рисунок 1.2 – Пристрій керування системою «розумного будинку» Samsung SmartThings Hub [7]

Такий пристрій дозволяє керувати такими пристроями, як «розумне освітлення», замки, термостати, шляхом використання спеціалізованого застосунку SmartThings. У тому числі, такий центр дозволяє дистанційно керувати безпекою будинку, керувати споживанням електроенергії, а також є функція отримувати сповіщення про підозрілу активність в будинку. Як у аналогічній системі Google Nest Hub, керування так само можна здійснювати шляхом голосових команд, налаштовувати завдання на основі тригерів, завдань, визначеного часу, для окремих або всіх підключених пристроїв одночасно.

Також, розглянемо третій альтернативний варіант, що пропонується на комерційному ринку. Платформа та застосунок, Xiaomi Mi Home, є системою

керування пристроями загальної системи «розумного будинку». Дана платформа підтримує широкий спектр різних датчиків, у тому числі датчики, освітлення, побутову техніку, камери відеоспостереження, підключаючись до пристроїв через Wi-Fi, ZigBee або Bluetooth.

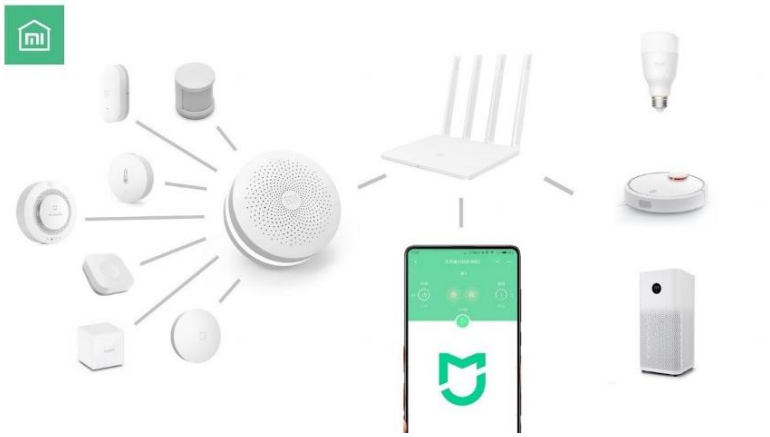

Рисунок 1.3 – Пристрій керування «розумним» будинком Xiaomi Mi Home [8]

У даній системі користувач може здійснювати дистанційне керування пристроями через окремий спеціальний застосунок Mi Home, а також налаштовувати правила автоматизації процесів, створювати «розумні сцени» для керування будинком персоналізовано. Такий застосунок надає єдиний інтерфейс для керування усіма «розумними» пристроями дистанційно, що підключені до платформи.

Попри широкі функціональні можливості, комерційні платформи мають низку обмежень, що впливають на їхню гнучкість і ефективність у специфічних умовах використання. По-перше, більшість таких систем є закритими або обмежено відкритими, що ускладнює або унеможливлює інтеграцію пристроїв сторонніх виробників. Це суттєво обмежує масштабованість і кастомізацію системи. По-друге, робота багатьох рішень залежить від постійного доступу до хмарної інфраструктури виробника, що створює ризики щодо надійності та безперервності роботи у разі збоїв зв’язку або сервісів. Також слід враховувати питання конфіденційності, адже передача даних на зовнішні сервери потенційно створює загрозу витоку персональної інформації. Вартість таких систем, особливо у разі комплексного

впровадження, також є доволі високою, що може бути критичним чинником для користувачів з обмеженим бюджетом.

Таким чином, хоча комерційні системи є зручними для масового користувача завдяки інтуїтивно зрозумілому інтерфейсу, технічній підтримці та широкому функціоналу, їхня закритість, залежність від інтернет-з’єднання та висока вартість є факторами, що обмежують можливості гнучкого налаштування та подальшого розширення. У зв’язку з цим усе більшого поширення набувають відкриті програмні платформи та апаратні рішення на базі мікроконтролерів, зокрема ESP32, що дозволяють створювати індивідуалізовані та незалежні системи автоматизації побуту.

1.2.2 Порівняння протоколів зв’язку

Одним з ключових аспектів побудови ефективної системи «розумного будинку» є вибір протоколу бездротового зв’язку, який визначає швидкість обміну даними, енергоспоживання, надійність з’єднання та здатність до масштабування мережі. Найпоширенішими технологіями у цій сфері є ZigBee, Z-Wave, Wi-Fi та Bluetooth [9].

ZigBee – це бездротовий протокол, заснований на стандарті IEEE 802.15.4, який функціонує в діапазоні частот 2,4 ГГц, а також у деяких регіонах – 868 МГц або 915 МГц. Основними перевагами ZigBee є низьке енергоспоживання, підтримка сітчастої топології та здатність об’єднувати до 65’000 пристроїв у єдину мережу. Завдяки цьому технологія ідеально підходить для пристроїв з батарейним живленням та для реалізації великих систем автоматизації. Проте через роботу на перевантаженому частотному діапазоні 2,4 ГГц можливі перешкоди з боку інших пристроїв, що використовують цей спектр (наприклад, Wi-Fi).

Z-Wave є комерційним протоколом, який працює у ліцензованих діапазонах нижчих частот (у Європі – 868 МГц), що забезпечує меншу ймовірність перешкод і більший радіус дії в приміщеннях. Як і ZigBee, Z-Wave підтримує сітчасту топологію, але обмежує кількість пристроїв до 232 на одну мережу. Його перевагами є стабільність з’єднання та хороша сумісність

пристроїв між собою завдяки сертифікації. До недоліків можна віднести вищу вартість модулів та менш розвинене open-source середовище порівняно з ZigBee.

Wi-Fi, що базується на стандарті IEEE 802.11, забезпечує високу швидкість передавання даних і широке розповсюдження інфраструктури. Він активно використовується у «розумному будинку» для підключення до Інтернету камер спостереження, голосових асистентів та інших пристроїв, що потребують великої пропускної здатності. Однак Wi-Fi характеризується відносно високим енергоспоживанням, що обмежує його використання у пристроях з автономним живленням. Крім того, одночасне підключення великої кількості пристроїв до одного маршрутизатора може знизити загальну стабільність мережі.

Bluetooth, також використовується в системах автоматизації, передусім для локальної взаємодії на короткій відстані. Основними перевагами є низьке енергоспоживання, швидке встановлення з’єднання та підтримка широкого спектру мобільних пристроїв. Проте обмежений радіус дії та невелика кількість одночасно підключених пристроїв обмежують масштабованість таких рішень.

Узагальнюючи, вибір протоколу залежить від конкретних вимог до системи: енергоефективність і масштабованість краще забезпечують ZigBee та Z-Wave, а для передачі великих обсягів даних доцільним є використання Wi-Fi, а найкращим для локальної взаємодії з мінімальним споживанням енергії – BLE. В умовах створення відкритих, масштабованих систем на базі ESP32 пріоритет часто надається Wi-Fi або Bluetooth, які вже інтегровані у цей мікроконтролер.

1.2.3 Відкриті програмні платформи: Home Assistant та OpenHAB

У контексті розвитку систем «розумного будинку» особливої уваги заслуговують відкриті програмні платформи, що забезпечують гнучкість, масштабованість та можливість повного контролю над архітектурою рішення. До найбільш поширених і підтримуваних спільнотою рішень належать Home

Assistant та OpenHAB – системи з відкритим вихідним кодом, які дозволяють інтегрувати велику кількість пристроїв, протоколів і сервісів в єдине кероване середовище [10].

Home Assistant є платформою для локального керування пристроями, написана на мові програмування Python. Вона орієнтована на приватність користувачів, адже вся логіка автоматизації та обробка даних можуть працювати без доступу до хмарних сервісів. Система підтримує понад 2000 інтеграцій з різними пристроями та сервісами, включаючи ZigBee, Z-Wave, MQTT, Bluetooth, Wi-Fi, а також голосові помічники (Google Assistant, Amazon Alexa). Home Assistant відзначається активним розвитком, регулярними оновленнями, простим інтерфейсом керування та наявністю великої спільноти розробників. Крім того, існує офіційний дистрибутив Home Assistant OS, який може бути розгорнутий на таких пристроях, як Raspberry Pi або міні-ПК [11].

OpenHAB – платформа з відкритим кодом, що створена на мові Java та працює на принципах кросплатформності. OpenHAB підтримує велику кількість застосунків («bindings»), що дозволяють керувати різноманітними пристроями через MQTT, Modbus, KNX, Z-Wave, ZigBee та інші протоколи. Платформа орієнтована як на початківців, так і на досвідчених користувачів, пропонуючи широкий набір інструментів для розробки складних сценаріїв автоматизації [12].

Обидві платформи мають свої переваги та особливості. Home Assistant більше підходить для користувачів, які цінують простоту налаштування, зручний графічний інтерфейс та швидкий доступ до функціоналу. OpenHAB, своєю чергою, забезпечує високу стабільність, гнучкість у налаштуванні, що важливо для реалізації масштабних або специфічних рішень у сфері автоматизації.

Відкриті платформи можуть слугувати або основою для розгортання кінцевої системи, або інструментом для тестування сумісності та інтеграції пристрою з іншими компонентами «розумного будинку». Наявність великої

кількості готових бібліотек та документації значно полегшує процес розробки та впровадження функціоналу керування побутовими приладами через бездротові мережі.

1.2.4 Оцінка доцільності використання відкритих рішень

Застосування відкритих програмних рішень у сфері реалізації систем «розумного будинку» набуває все більшої актуальності у зв’язку з потребою користувачів у гнучкості, незалежності від комерційних платформ та доступі до повного функціоналу без додаткових витрат. У цьому контексті особливої уваги заслуговують платформи з відкритим вихідним кодом, такі як Home Assistant та OpenHAB, які, на відміну від закритих комерційних систем, забезпечують широкий простір для кастомізації, інтеграції нестандартних пристроїв та оптимізації під індивідуальні потреби користувача.

Однією з ключових переваг використання відкритих рішень є відсутність прив’язки до конкретного виробника або екосистеми, що дозволяє уникнути обмежень, характерних для багатьох комерційних продуктів, таких як необхідність платної підписки, фіксований перелік сумісного обладнання або обмежений доступ до внутрішньої логіки системи. У відкритих системах користувач може повністю контролювати, які пристрої підключати, які сценарії автоматизації реалізовувати, як зберігати та обробляти дані [13].

Також важливою є економічна доцільність впровадження відкритих рішень. Використання безкоштовного ПЗ, широкої підтримки від спільноти та доступності бібліотек дозволяє зменшити витрати на розробку та обслуговування системи. Це особливо актуально у випадках побудови експериментальних або навчальних макетів систем «розумного будинку», а також для індивідуальних користувачів, які прагнуть автоматизувати власне житло без суттєвих фінансових витрат.

Не менш важливою є і безпекова складова: більшість відкритих платформ підтримують локальне зберігання даних, що зменшує ризик витоку персональної інформації через сторонні хмарні сервіси. Можливість ручного налаштування дозволяє реалізувати власну політику безпеки, обирати рівень

доступу до пристроїв і даних, використовувати шифрування та багаторівневу автентифікацію.

Разом з тим, варто зазначити і певні обмеження, пов’язані з використанням відкритих рішень. Це, насамперед, необхідність наявності базових знань у галузі мережевих технологій, програмування та електроніки, що може ускладнити процес розгортання системи для пересічного користувача. Крім того, відсутність централізованої технічної підтримки передбачає потребу самостійного вирішення технічних проблем, що виникають під час експлуатації.

З огляду на вищезазначене, можна зробити висновок, що доцільність використання відкритих рішень у системах «розумного будинку» є високою в контексті індивідуальних, навчальних та дослідницьких проєктів, а також для користувачів, які цінують гнучкість, контроль та прозорість систем. У межах КБР, відкриті програмні платформи можуть бути використані як інструмент для інтеграції апаратної частини, розробленої на базі модуля ESP32, з високим ступенем адаптації до конкретних функціональних вимог системи.

1.3 Обґрунтування та вибір підходів щодо виконання завдань

1.3.1 Вибір архітектури керування

Вибір архітектури керування системою «розумного будинку» є ключовим етапом, що визначає загальну ефективність, надійність та масштабованість розроблюваного рішення. Архітектура визначає спосіб взаємодії між апаратними компонентами, модулями введення-виведення, обчислювальними блоками та інтерфейсами користувача, а також впливає на швидкодію та стабільність системи.

У межах даного проєкту було обґрунтовано доцільність використання розподіленої архітектури з централізованим логічним управлінням. Така модель дозволяє поєднувати гнучкість локального контролю на рівні окремих пристроїв із можливістю централізованого моніторингу та налаштування через головний контролер, яким виступає мікроконтролер ESP32. Подібна

організація забезпечує підвищену відмовостійкість: у разі втрати зв’язку з центральним вузлом окремі модулі можуть продовжувати автономне виконання своїх функцій.

Застосування ESP32 як центрального елемента системи дозволяє реалізувати одночасну підтримку кількох бездротових інтерфейсів, що забезпечує зручність інтеграції з іншими пристроями або хмарними сервісами. Передбачається використання Wi-Fi для зв’язку з користувацьким інтерфейсом, а також для обміну даними з сенсорними модулями та виконавчими пристроями. Локальні вузли системи можуть включати різноманітні датчики (температури, вологості, руху тощо) та виконавчі елементи, що з’єднуються із центральним вузлом безпосередньо або через шину.

Вибрана архітектура керування також враховує принципи модульності та масштабованості. Це дозволяє користувачу поступово розширювати функціональні можливості системи шляхом додавання нових вузлів без необхідності модифікації основного програмного коду або структури даних. Комунікація між модулями може здійснюватися за допомогою MQTT або HTTP-протоколу, що є популярними у сфері IoT і підтримуються більшістю відкритих програмних платформ.

Таким чином, обрана архітектура забезпечує баланс між централізованим управлінням і децентралізованою обробкою подій, що є ефективним рішенням для побудови гнучкої, надійної та зручно масштабованої системи «розумного будинку».

1.3.2 Вибір апаратної платформи

Одним із ключових етапів розробки системи керування «розумним будинком» є вибір апаратної платформи, яка виступає основою для реалізації логіки керування, збору даних із сенсорів, обміну інформацією між модулями та організації інтерфейсу з користувачем. У контексті сучасних вимог до таких систем важливими критеріями є енергоефективність, обчислювальні

можливості, наявність вбудованих засобів бездротового зв’язку, підтримка розширень, доступність документації та програмного забезпечення.

На основі аналізу ринку мікроконтролерів та модулів для IoT-проєктів було прийнято рішення використати ESP32 як головну апаратну платформу проєкту. Мікроконтролер ESP32, розроблений компанією Espressif Systems, представляє собою потужний двоядерний SoC з архітектурою Tensilica Xtensa LX6, який працює на частоті до 240 МГц. Його головними перевагами є:

– інтегровані модулі бездротового зв’язку: ESP32 підтримує Wi-Fi (802.11 b/g/n) та Bluetooth, що спрощує інтеграцію з мережевими протоколами та мобільними пристроями без додаткового зовнішнього обладнання;

– наявність численних цифрових і аналогових входів/виходів, інтерфейсів UART, SPI, I2C, ADC, DAC та PWM, що забезпечує гнучкість під час підключення датчиків і виконавчих пристроїв;

– підтримка енергозберігаючих режимів: ESP32 підтримує декілька рівнів енергозбереження, що є критичним при використанні автономних джерел живлення або батарей.

Крім цього, ESP32 відзначається високим співвідношенням вартість/функціональність, що робить його економічно доцільним варіантом як для прототипування, так і для промислового застосування. Його відкритий програмний стек та активна спільнота розробників дозволяють гнучко адаптувати систему до змін вимог та інтегрувати сторонні рішення, зокрема програмні платформи типу Home Assistant або OpenHAB.

Таким чином, вибір ESP32 як апаратної основи системи керування «розумним будинком» є технічно обґрунтованим і стратегічно доцільним, оскільки він дозволяє задовольнити як поточні функціональні вимоги, так і забезпечити потенціал для подальшої модернізації системи.

1.3.3 Обґрунтування вибору програмного забезпечення

Проєктування системи автоматизації житла на базі мікроконтролера ESP32 потребує виваженого підходу до вибору програмного забезпечення,

здатного забезпечити ефективну реалізацію завдань реального часу, стабільність функціонування, розширюваність та підтримку бездротового зв’язку. У межах даного проєкту ключовою вимогою є здатність мікроконтролера паралельно обробляти численні завдання, зокрема обробку сигналів від сенсорів, керування виконавчими пристроями та підтримку зв’язку з мобільним застосунком. З огляду на це, основним середовищем виконання було обрано FreeRTOS – операційну систему реального часу з підтримкою багатозадачності, таймерів, міжпроцесної взаємодії та синхронізації [14].

FreeRTOS є офіційно підтримуваним рішенням для мікроконтролерів ESP32 розробником платформи Espressif, що забезпечує повну сумісність із апаратними ресурсами модуля та надає можливість гнучкої оптимізації використання пам’яті, пріоритетів завдань та механізмів взаємодії між компонентами системи. Такий підхід дозволяє досягти високої надійності системи за умов одночасного виконання кількох процесів, що є критично важливим для систем «розумного будинку».

Комунікація з користувачем у системі реалізується за допомогою Bluetooth-з’єднання, що забезпечує локальне керування пристроями без потреби в доступі до Wi-Fi або хмарних сервісів. Для взаємодії з мікроконтролером розроблено спеціалізований Android-застосунок, що дозволяє виконувати контроль і моніторинг роботи системи через зручний інтерфейс.

Прошивка ESP32 розроблялася за допомогою Arduino IDE у поєднанні з мовою програмування C++. Такий вибір обумовлений широкою підтримкою ESP32 у середовищі Arduino, наявністю великої кількості готових бібліотек для роботи з Bluetooth, FreeRTOS і периферійними модулями, а також низьким порогом входу, що дозволяє зосередитися на логіці системи. Arduino IDE також дозволяє інтегрувати FreeRTOS у вигляді окремих завдань із використанням відповідних директив, що спрощує реалізацію багатопоточності.

Таким чином, комбінація FreeRTOS як операційної системи, Arduino IDE й C++ як засобів розробки та Bluetooth як каналу зв’язку з користувачем через Android-застосунок створює збалансовану програмну архітектуру, що забезпечує масштабованість, надійність та ефективність функціонування системи «розумного будинку» на базі ESP32.

1.4 Формування вимог до АПЗ

1.4.1 Загальні технічні вимоги до системи

Розроблювана система автоматизації житлового простору має відповідати сукупності загальних технічних вимог, які забезпечують її ефективну функціональність, надійність, масштабованість та зручність експлуатації. Основною вимогою є підтримка роботи в режимі реального часу з використанням FreeRTOS як середовища керування завданнями, що дозволяє гарантувати своєчасну обробку подій від сенсорів і виконавчих пристроїв. Система повинна забезпечувати безперебійну взаємодію з мобільним застосунком через бездротовий інтерфейс Bluetooth, що потребує стабільного двостороннього зв’язку з мінімальною затримкою передачі даних.

З огляду на специфіку застосування системи в умовах житлового середовища, важливою вимогою є низьке енергоспоживання, особливо за наявності автономного живлення. Передбачено реалізацію режимів енергозбереження мікроконтролера ESP32 та інших апаратних компонентів. Архітектура системи має бути модульною, що дозволяє адаптувати її до потреб конкретного користувача шляхом додавання або вилучення окремих функціональних блоків без необхідності внесення змін до основної логіки керування.

Оскільки система взаємодіятиме з користувачем через Android-застосунок, особливе значення має забезпечення сумісності інтерфейсів передачі даних та стабільна підтримка з’єднання. Також важливою є вимога до надійності функціонування – система повинна бути здатною до відновлення після збоїв у живленні або переривання з’єднання без втрати поточного стану.

Окремо слід зазначити потребу в базовому рівні захисту від несанкціонованого доступу через Bluetooth-з’єднання для підвищення безпеки експлуатації. Крім того, система має передбачати можливість оновлення програмного забезпечення з мінімальними витратами часу та ресурсів.

1.4.2 Вимоги до модуля керування та периферійних пристроїв

Модуль керування системою «розумного будинку» виконує роль центрального елементу, відповідального за координацію роботи всіх підключених пристроїв, обробку сигналів від сенсорів, виконання логіки автоматизації та забезпечення зв’язку з користувачем.

У якості апаратної основи було обрано мікроконтролер ESP32, що відповідає сучасним вимогам до вбудованих систем, зокрема завдяки наявності двоядерного процесора, підтримці FreeRTOS, а також інтегрованих модулів бездротового зв’язку – Wi-Fi та Bluetooth. Мікроконтролер повинен підтримувати одночасне виконання кількох завдань у режимі реального часу, з гарантованим пріоритетом обробки критичних подій. Крім того, необхідно забезпечити наявність достатньої кількості цифрових та аналогових входів/виходів для підключення сенсорів (наприклад, температури, вологості, руху) та виконавчих пристроїв. Надійна взаємодія з периферією повинна бути реалізована через стандартні інтерфейси: GPIO, I^2C, UART, SPI тощо.

Усі периферійні пристрої, що входять до складу системи, мають забезпечувати стабільну роботу у побутових умовах експлуатації. Сенсори повинні характеризуватися високою точністю вимірювання та низьким енергоспоживанням. Виконавчі пристрої мають забезпечувати швидке реагування на команди, надійність у циклічному режимі роботи та електричну безпеку при тривалому навантаженні.

З метою забезпечення гнучкості системи та можливості її адаптації під індивідуальні потреби користувача, апаратна архітектура повинна бути модульною: всі елементи мають легко підключатися до основного модуля без необхідності перепроектування плати. Також важливо передбачити можливість розширення функціоналу шляхом додавання нових типів сенсорів

або виконавчих елементів у рамках наявного ресурсу обчислювальної потужності та пам’яті ESP32.

Висновки до розділу 1

У першому розділі було розглянуто теоретичні засади концепції «розумного будинку», актуальність якої зростає в умовах стрімкого розвитку технологій автоматизації побутових процесів та широкого впровадження Інтернету речей. Проаналізовано основні напрями розвитку таких систем, зокрема орієнтацію на підвищення енергоефективності, зручності користування, безпеки, а також інтеграцію з хмарними сервісами.

Проведено огляд сучасних комерційних рішень у галузі автоматизації житла, у межах якого було виявлено як їхні переваги, так і обмеження. Здійснено порівняння основних бездротових протоколів зв’язку – ZigBee, Z-Wave, Wi-Fi та Bluetooth – за критеріями енергоспоживання, дальності дії, вартості реалізації та сумісності з побутовими пристроями.

Окрема увага приділялася відкритим програмним платформам, таким як Home Assistant та OpenHAB, що забезпечують гнучке налаштування, підтримку широкого спектра пристроїв. У підрозділах 1.3 та 1.4 обґрунтовано вибір архітектури системи на базі мікроконтролера ESP32, з урахуванням його технічних характеристик: багатозадачності, низького енергоспоживання, підтримки FreeRTOS та інтегрованих модулів бездротового зв’язку. Сформульовано комплекс вимог до апаратного та програмного забезпечення, що забезпечують функціональність, надійність та розширюваність системи.

Проведений аналіз заклав основу для наступного етапу розробки: проєктування апаратно-програмної частини забезпечення, створення алгоритмів її функціонування та реалізації користувацького інтерфейсу.

2 ПРОЄКТУВАННЯ СИСТЕМИ КЕРУВАННЯ «РОЗУМНИМ БУДИНКОМ»

2.1 Функціональне моделювання системи

У першу чергу, функціональне моделювання системи керування «розумним будинком» передбачає визначення її основних функцій, побудову сценаріїв взаємодії користувача із системою, а також опис потоків інформації між компонентами. Відповідно, у даному розділі буде проведено аналіз та виконати перехід від загального опису системи до її структурованої функціональної моделі, що дозволить окреслити функції кожного компонента системи та їх взаємодію в рамках єдиної архітектури.

2.1.1 Визначення основних функцій та сценаріїв взаємодії

Процес функціонального моделювання інтелектуальної системи управління побутовим середовищем («розумного будинку») передбачає ідентифікацію ключових функціональних можливостей, що забезпечують виконання системою покладених на неї завдань Основною метою цього етапу є визначення чітко окреслених функцій та можливих сценаріїв взаємодії користувача із системою або її підсистемами.

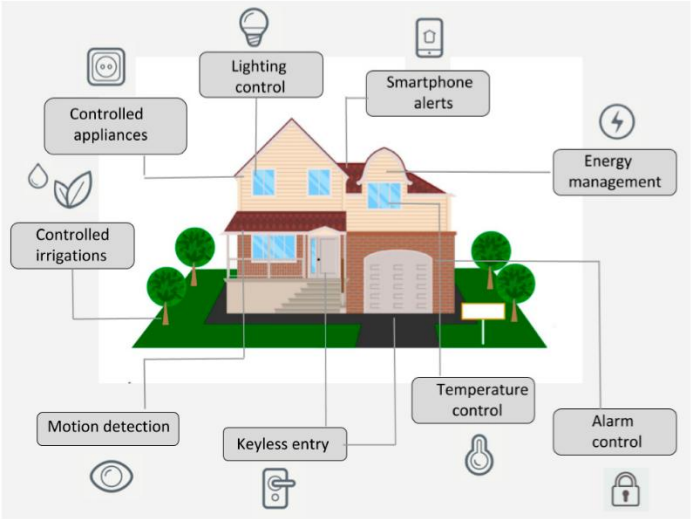

Рисунок 2.1 – Загальні функції системи «розумного будинку» [12]

На основі аналізу потреб користувача, стандартів автоматизації побуту (наприклад, KNX, Zigbee Home Automation, Matter), а також вимог безпеки, комфорту та енергоефективності, до системи висуваються такі функціональні вимоги:

1) моніторинг параметрів середовища:
   – збір даних із сенсорів температури, вологості, освітленості, газу, диму, руху тощо;
   – періодичне оновлення показників на інтерфейсі користувача;
2) автоматичне реагування на події:
   – активація освітлення при виявленні руху;
   – увімкнення вентиляції при перевищенні допустимої вологості;
   – сигналізація про виявлення диму або витік газу.
3) керування системою вручну або дистанційно:
   – взаємодія з системою через локальну панель керування, мобільний застосунок або вебінтерфейс;
   – підтримка авторизації та різних рівнів доступу.
4) аналіз та протоколювання подій:
   – зберігання історії подій у внутрішній базі даних;
   – формування звітів щодо температурних режимів, енергоспоживання, кількості спрацювань тощо.
5) сценарне управління:
   – реалізація користувацьких сценаріїв (наприклад, «нічний режим», «я вдома», «аварійна ситуація»);
   – можливість динамічного налаштування правил реагування.
6) самодіагностика та відновлення роботи:
   – виявлення несправностей окремих підсистем;
   – перезавантаження вузлів у разі збоїв;
7) відправка повідомлень про критичні помилки.

Таким чином, функціональне моделювання управління побутовим середовищем дозволяє сформувати цілісне уявлення про ключові можливості

та поведінку в типових ситуаціях. Визначені вимоги охоплюють як базові операції зі збору та обробки даних від сенсорів, так і високорівневі механізми автоматичного реагування, сценарного керування, захисту та самодіагностики, що забезпечує гнучкість, адаптивність та надійність системи, створює основу для побудови ефективної взаємодії з користувачем і підвищення загального рівня безпеки та комфорту в побутовому середовищі.

2.1.2 Опис інформаційних потоків у системі

Інформаційні потоки в системі «розумного будинку» є ключовим елементом забезпечення її функціональної цілісності та узгодженої взаємодії між компонентами. Вони охоплюють процеси збору, передачі, обробки, зберігання та відображення даних, що надходять від сенсорів, виконавчих пристроїв, зовнішніх джерел та користувача. Структурований аналіз таких потоків дозволяє визначити архітектурні особливості системи, оптимізувати обмін даними і забезпечити надійність управління.

У системі можна виділити такі основні типи інформаційних потоків:

– вхідні інформаційні потоки: дані надходять від датчиків температури, вологості, освітленості, диму, газу, руху тощо. Потоки можуть бути безперервними або подійно-орієнтованими. Ці потоки формують основу для прийняття рішень у системі;

– внутрішні інформаційні потоки: включають обмін повідомленнями між контролерами, мікросхемами введення/виведення, базою даних та логічними модулями обробки. Наприклад, після отримання даних про перевищення порогу температури система передає команду на активацію вентиляції або кондиціонера;

– вихідні інформаційні потоки: формуються у вигляді команд, що спрямовуються до виконавчих пристроїв – реле, моторів, світильників тощо. Такі потоки забезпечують реалізацію автоматичних або ручних сценаріїв;

– потоки взаємодії з користувачем: дані від користувача надходять через інтерфейси – мобільний застосунок, вебінтерфейс, локальну панель

керування. У відповідь система повертає інформацію про поточні параметри, статуси пристроїв, історію подій або пропонує дії за сценаріями;

– журнали подій та діагностичні потоки: формуються у вигляді логів та звітів, що зберігаються у внутрішній базі даних. Вони містять інформацію про всі події, включаючи аварійні ситуації, реакції системи та дії користувача. Такі потоки необхідні для діагностики, оптимізації сценаріїв та ретроспективного аналізу;

– мережеві потоки: у разі використання хмарної платформи (наприклад, AWS IoT) забезпечується обмін інформацією між локальною системою та зовнішніми інтерфейсами – для зберігання резервних копій, дистанційного моніторингу та оновлення конфігурацій.

У системі «розумного будинку» надходження даних від сенсорних пристроїв є фундаментальним процесом, що забезпечує реальне уявлення про поточний стан середовища. Дані з датчиків температури, вологості, освітленості, диму та руху надходять у систему у вигляді сигналів, що формуються внаслідок фізичних змін у середовищі.

Рисунок 2.2 – Алгоритм роботи пристрою з термодатчиком

Відповідно, у даній системі термодатчики вимірюватимуть температуру повітря, перетворюючи аналоговий сигнал у цифровий, що надсилатиметься до центрального контролера через інтерфейси (рис. 2.2).

У типових реалізаціях використовується сенсор DHT11, що здатен вимірювати температуру в діапазоні від –20 до +60 °C з точністю ±2 °C. Цей сенсор також одночасно вимірює вологість, що робить його універсальним для побутових застосувань (рис. 2.3).

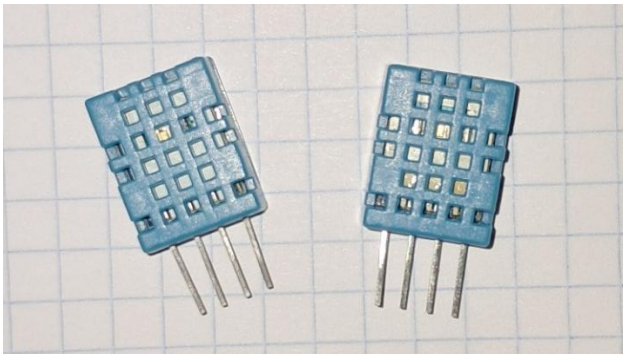

Рисунок 2.3 – Фото датчика DHT11

Пристрій DHT11 є недорогим рішенням, що можна використовувати в широкому спектрі застосувань від домашньої автоматизації то промислового управління системами. Датчик використовує два основні компоненти для виконання вимірювання параметрів температури та вологості: ємнісний датчик вологості та термістор. Загальні технічні характеристики датчику наведені у табл. 2.1.

Таблиця 2.1 – Технічні характеристики сенсора DHT11

Характеристика	**Значення**
Діапазон вимірювання вологості, %	20–90
Діапазон виміру температури, °C	-20–60
Точність вимірювання вологості, RH	± 2 % RH
Точність вимірювання температури, °C	± 0,5
Напруга живлення, В	5
Кількість виводів	4

Аналогічно, гігрометри передають інформацію про відносну вологість у приміщенні, дозволяючи системі аналізувати комфортні або критичні рівні вологості (рис. 2.4).

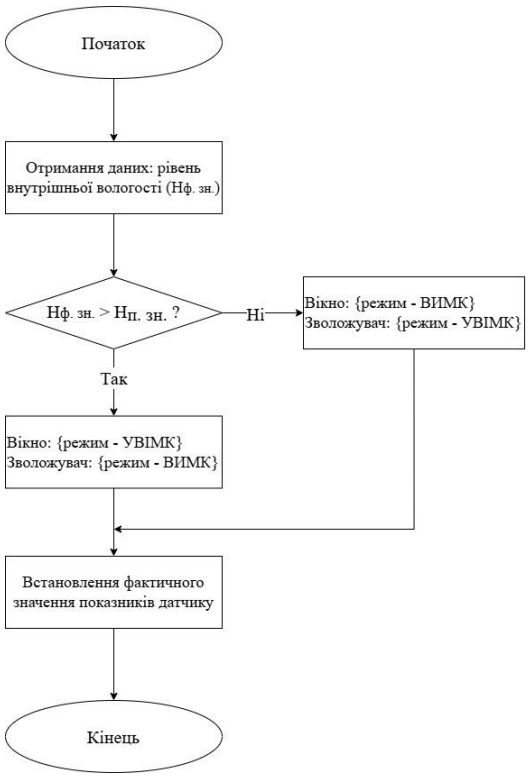

Рисунок 2.4 – Алгоритм роботи датчику вологості

Окрім DHT11, популярним є AM2301, який має цифровий вихід, працює з високою стабільністю, і застосовується у приміщеннях, де необхідний постійний контроль клімату, наприклад, у дитячих кімнатах, теплицях, серверних (рис. 2.5).

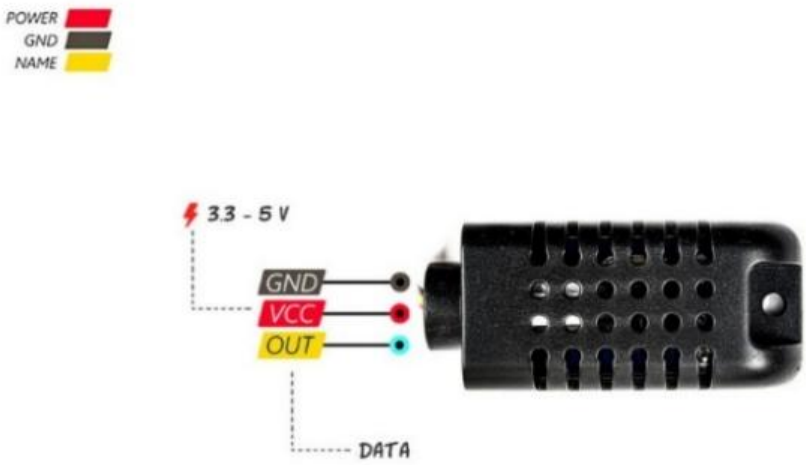

Рисунок 2.5 – Зображення датчику AM2301 [15]

Загалом, AM2301 є датчиком, що широко застосовується для моніторингу температури та вологості в домашніх та промислових умовах. Він

забезпечує точні вимірювання та може інтегруватися у системи «розумного будинку» для автоматичного контролю мікроклімату. Датчик використовує ємнісний сенсор вологості та термістор для визначення температури. Вбудований АЦП перетворює аналоговий сигнал у цифровий, що дозволяє передавати дані через цифровий інтерфейс. Загальні технічні характеристики датчику наведені у табл. 2.2.

Таблиця 2.2 – Технічні характеристики сенсора AM2301

Характеристика	**Значення**
Точність, °C	0,1
Точність вимірювання	0,3–1
Діапазон вимірювання вологості, %	0–100
Діапазон виміру температури, °C	-40–80
Точність вимірювання вологості, RH	± 2%
Точність вимірювання температури, °C	± 0,5
Напруга живлення, В	3,3–5,2
Кількість виводів	3

Датчики освітленості використовують фоточутливі елементи, що реагують на рівень світла в приміщенні, і дозволяють динамічно регулювати штучне освітлення (рис. 2.6).

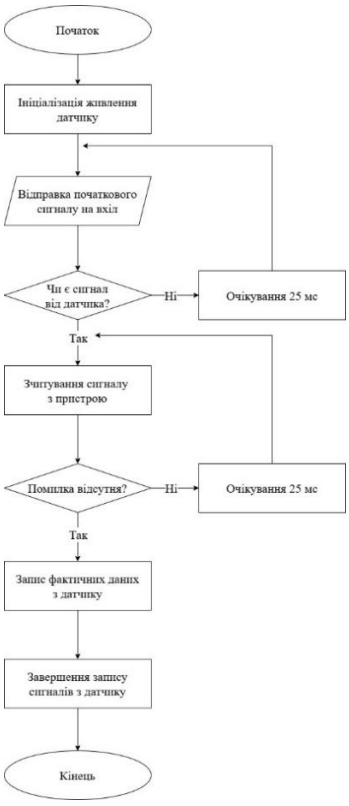

Рисунок 2.6 – Алгоритм роботи датчику освітленості

Серед них найчастіше використовується BH1750, що працює через інтерфейс I²C, має широкий діапазон вимірювання освітленості від 1 до 65535 люксів та високу чутливість (рис. 2.7).

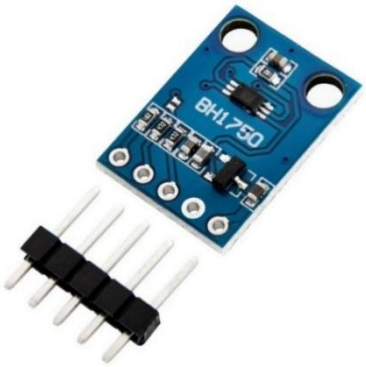

Рисунок 2.7 – Зображення сенсору BH1750 [16]

Також, BH1750 є цифровим датчиком освітленості, що широко використовується в системах «розумного будинку» для автоматичного регулювання освітлення та моніторингу навколишнього середовища. Датчик використовує фотодіодний масив для вимірювання рівня освітленості та передає дані у люксах через I²C інтерфейс. Це дозволяє легко інтегрувати його в Arduino, ESP8266, ESP32 та інші мікроконтролери. Загальні технічні характеристики датчику наведені у табл. 2.3.

Таблиця 2.3 – Технічні характеристики модуля BH1750

Характеристика	**Значення**
Точність, люкс	1
Напруга живлення, В	3–5
Діапазон даних, лк	0–65635
Інтерфейс	I^2C
Кількість виводів	5

Завдяки низькому енергоспоживанню, сенсор добре підходить для автономних пристроїв.

Сенсори диму й газу, побудовані на основі хімічної реактивності або оптичного принципу, виявляють наявність потенційно небезпечних речовин у повітрі та передають тривожний сигнал при перевищенні допустимих концентрацій (рис. 2.8).

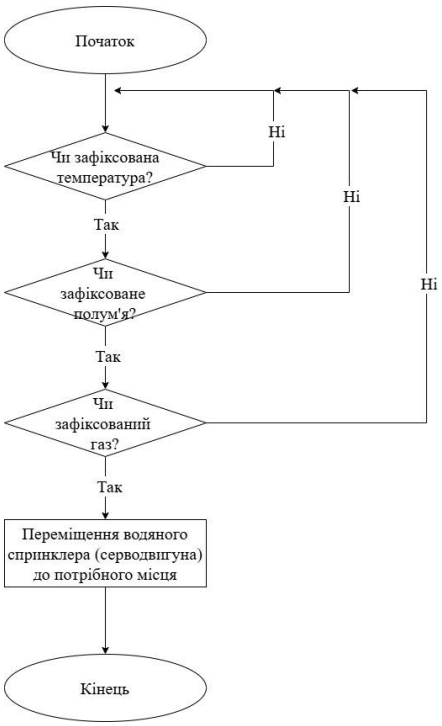

Рисунок 2.8 – Загальний алгоритм роботи датчику газу

Для побутового застосування часто обирають MQ-2 (рис. 2.9) або MQ-135. MQ-2 чутливий до диму, пропану, бутану та метану, і забезпечує аналоговий вихід для калібрування порогів спрацювання. MQ-135 краще підходить для виявлення чадного газу, бензолу та аміаку, що особливо важливо для кухонь і гаражів.

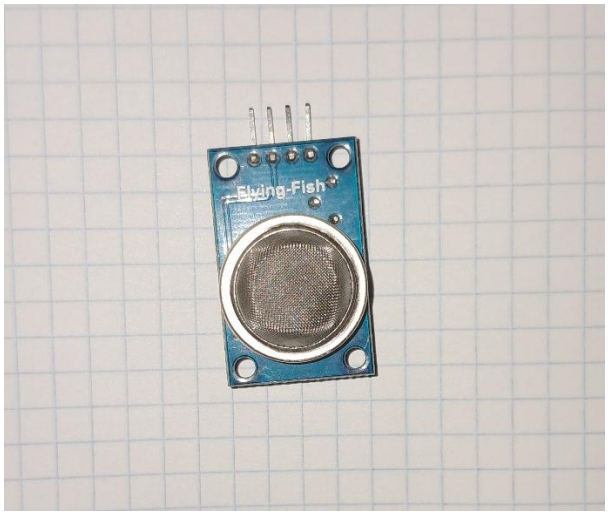

Рисунок 2.9 – Зображення датчику газу MQ-2

Широко у системах «розумного будинку» використовується саме датчик MQ-2 для виявлення витоків газу, контролю якості повітря та забезпечення безпеки. Сенсор MQ-2 використовує метал-оксидний

напівпровідниковий сенсор, який змінює свій опір при контакті з певними газами. Це дозволяє виявляти LPG, пропан, метан, водень, алкоголь, дим та чадний газ. Загальні технічні характеристики датчику наведені у табл. 2.4.

Таблиця 2.4 – Технічні характеристики сенсору MQ-2

Характеристика	**Значення**
Діапазон чутливості, мд	300–10000
Опір чутливого елемента, кОм	1–20
Час відгуку, с	менше 10
Чутливість, с	більше 5
Опір нагрівача, Ом	31
Струм нагрівача, мА	менше 180
Напруга живлення, В	5
Потужність нагрівача, мВт	менше 900
Кількість виводів	4

Датчики руху, зазвичай побудовані на інфрачервоній або ультразвуковій технології, фіксують наявність руху в зоні покриття й генерують подійно-орієнтовані повідомлення. Найбільш поширеною моделлю є HC-SR501 – PIR-сенсор, що реагує на зміну теплового фону людини, має можливість регулювання чутливості й часу спрацювання. У випадках, коли потрібне точніше зондування (наприклад, контроль проходу в коридорі), застосовується ультразвуковий сенсор HC-SR04 (рис. 2.10), що вимірює відстань до об’єкта і визначає факт руху або наближення.

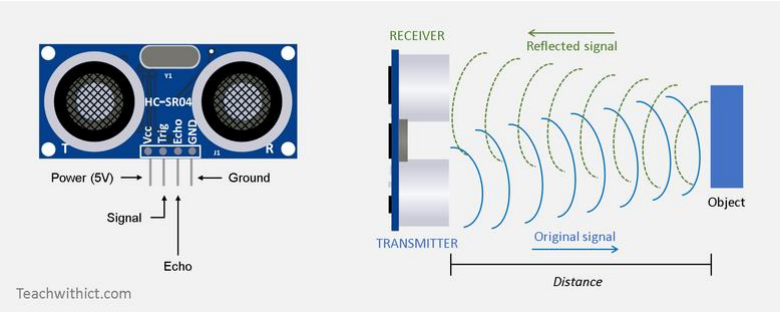

Рисунок 2.10 – Принцип роботи датчику ультразвуку HC-SR04 [17]

Ультразвуковий датчик відстані HC-SR04 широко використовується в системах «розумного будинку» для контролю простору, автоматизації та безпеки. Датчик використовує ультразвукові хвилі для вимірювання відстані до об’єкта. Він випромінює сигнал, який відбивається від перешкоди, а потім

приймач фіксує час повернення сигналу, що дозволяє розрахувати відстань. Загальні технічні характеристики датчику наведені у табл. 2.5.

Таблиця 2.5 – Технічні характеристики датчику HC-SR04

Характеристика	**Значення**
Робоча напруга, В	3,8–5,5
Струм, мА	8
Частота, кГц	40
Максимальна дистанція, мм	1500
Роздільна здатність, мм	3
Ширина імпульсів, мкс	10
Кут огляду	15°
Кількість виводів	4

Усі ці сенсорні дані надходять у центральну обчислювальну одиницю або шлюз, де попередньо обробляються, фільтруються та агрегуються. Далі інформація використовується як основа для прийняття рішень системою, ініціювання автоматичних дій (наприклад, включення світла, запуск вентиляції) або оновлення інформації на інтерфейсі користувача в режимі реального часу. Надійність і точність цього процесу безпосередньо впливають на ефективність функціонування всієї системи та рівень комфорту і безпеки у середовищі проживання.

2.2 Алгоритми взаємодії підсистем

Алгоритми взаємодії між підсистемами середовища «розумного будинку» відіграють ключову роль у забезпеченні узгодженої, надійної та адаптивної роботи системи керування таким. Підсистеми, такі як моніторинг параметрів середовища, управління освітленням, вентиляцією, безпекою, енергоспоживанням та комунікацією з користувачем, функціонують як незалежні модулі, які взаємодіють через єдиний центр керування – контролер або програмний шлюз.

Розробка алгоритмів ґрунтується на принципах подійно-орієнтованої архітектури (англ. event-driven architecture), де будь-яка зміна стану середовища (температура, освітленість, рух, дим тощо) генерує подію, яка

передається до центрального контролера. У контролері ця подія обробляється відповідно до заздалегідь визначених сценаріїв або динамічно адаптованих правил логіки. Таким чином, відбувається реакція системи – запуск виконавчих механізмів або зміна конфігурації підсистем.

Кожен алгоритм взаємодії визначається у вигляді алгоритмічного дерева, що включає наступні етапи:

– ініціація події (наприклад, перевищення температурного порогу);

– виклик відповідного обробника у модулі логіки;

– передача повідомлень у залежні підсистеми (наприклад, вентиляцію, повідомлення користувачу);

– зміна стану системи (вмикання пристрою, реєстрація дії в журналі подій).

Наприклад, у разі виявлення підвищеної вологості датчиком AM2301, система активує підсистему вентиляції. Якщо рівень вологості не знижується протягом визначеного інтервалу, алгоритм переходить до наступної гілки: надсилається повідомлення користувачу та відбувається повторне сканування параметрів з пріоритетним опитуванням.

Ключова особливість розроблених алгоритмів полягає в підтримці пріоритетного мультипроцесного планування завдань, коли критичні події (дим, газ, несанкціонований рух) переривають звичайний цикл обробки та мають доступ до швидкого каналу реакції (англ. fast response pipeline). Крім того, враховано механізми взаємного підтвердження станів між підсистемами (наприклад, для активації пожежної сигналізації необхідно, щоб тривогу одночасно підтвердили як сенсор диму MQ-2, так і температурний сенсор DHT11). Такий підхід зменшує ймовірність хибних спрацювань і забезпечує більшу достовірність рішень.

Обробка подій здійснюється у черзі подій з пріоритетами. Для цього використовуються структури черг типу FIFO з можливістю переваги критичних подій. Усі події, що генеруються підсистемами, підлягають

періодичному аналізу задля виявлення нештатних ситуацій та забезпечення самодіагностики.

2.2.1 Загальний алгоритму роботи системи

Проєктування системи управління побутовим середовищем потребує розробки узагальненого алгоритму її функціонування, що забезпечуватиме стабільну, адаптивну та ефективну взаємодію всіх підсистем у реальному часі. Такий алгоритм визначає послідовність дій, що виконує система в процесі моніторингу, аналізу даних, прийняття рішень і виконання керуючих дій.

Алгоритм роботи системи базується на концепції подійно-орієнтованого управління з постійним циклом обробки інформації. Його структура включає декілька ключових етапів, що відображають логіку функціонування типових «розумних середовищ».

На першому етапі відбувається ініціалізація системи: здійснюється автоматичне виявлення та перевірка доступності підключених сенсорів і виконавчих пристроїв, зчитування попередньо збережених налаштувань користувача, а також запуск внутрішніх сервісів моніторингу. У разі виявлення критичних помилок активується підсистема діагностики.

Далі, на регулярній основі або за тригерною умовою виконується опитування сенсорів. Зокрема, у системі реалізується зчитування таких параметрів:

– температури (на основі сенсора DHT11, що забезпечує точність ±0.5°C);

– вологості (використовується AM2301 з цифровим виходом та інтерфейсом UART);

– освітленості (застосовується фотометричний модуль BH1750, що працює через шину I2C);

– диму та газу (датчик MQ-2, який реагує на LPG, CO та CH_4);

– руху (ультразвуковий сенсор HC-SR04).

Після отримання сигналів, система виконує попередню обробку даних, що включає фільтрацію перешкод, усереднення значень (наприклад, методом ковзного середнього) та нормалізацію у прийнятний для аналізу формат. Цей етап спрямований на зменшення похибок вимірювання та виключення хибно позитивних спрацювань.

Наступним етапом є оцінка поточної ситуації. Центральний контролер порівнює поточні значення з визначеними порогами або умовами сценаріїв. Наприклад:

– якщо вологість перевищує 70 %, активується сценарій увімкнення вентиляції;

– при фіксації руху в нічний час: активується підсвітка приміщення;

– у разі виявлення диму чи газу: подається тривожне повідомлення та активується сигналізація.

На основі результатів аналізу запускається виконання відповідних дій, що передбачає передавання команд на виконавчі пристрої (реле, освітлення, клапани, сирени), оновлення стану в інтерфейсі користувача (мобільному застосунку або вебпанелі), а також, за потреби, надсилання push-повідомлень або email-сповіщень.

Паралельно з цим, система виконує логування подій. Вся інформація щодо сенсорних показників, прийнятих рішень і активованих сценаріїв зберігається у внутрішній базі даних SQLite, що дозволяє надалі здійснювати статистичний аналіз.

На завершальному етапі відбувається підготовка до наступного циклу: система повертається у стан очікування нових даних або подій, продовжуючи безперервний моніторинг параметрів середовища.

Таким чином, розроблений алгоритм забезпечує модульну, масштабовану та надійну взаємодію між усіма підсистемами «розумного будинку», створюючи основу для динамічного реагування на зміну умов середовища, підтримання безпеки, комфорту та енергоефективності в побутовому просторі.

2.2.2 Алгоритм обробки подій від сенсорів

Обробка подій, що надходять від сенсорів, є ключовим компонентом функціонування системи керування «розумним будинком». Алгоритм обробки забезпечує своєчасне виявлення змін у параметрах середовища та формування відповідних реакцій системи з урахуванням пріоритетів і сценаріїв роботи.

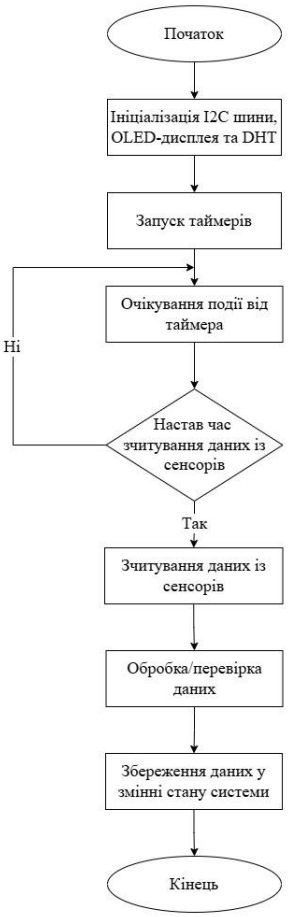

Рисунок 2.11 – Блок-схема алгоритму подій від сенсорів у системі

Початковим кроком алгоритму є прийом сигналів від сенсорів у вигляді цифрових або аналогових даних, які передаються по комунікаційних інтерфейсах (наприклад, I2C, UART, Zigbee, Wi-Fi). Оскільки сенсорні дані можуть надходити як періодично, так і подійно (при виникненні певної події, наприклад, виявленні руху або диму), алгоритм передбачає механізм подійної обробки, що оптимізує реагування та економить ресурси системи.

Далі дані проходять етап попередньої обробки, який включає перевірку коректності та достовірності отриманих значень, фільтрацію шумів та усунення аномалій (за допомогою методів, наприклад, ковзного середнього

або порогового фільтра). Це дозволяє зменшити кількість хибних спрацювань і забезпечує стабільність роботи системи.

Основною логікою алгоритму є порівняння отриманих показників із заздалегідь встановленими пороговими значеннями або умовами, що відповідають нормальному або критичному стану. Для кожного сенсора визначені унікальні межі, наприклад, температура – від 18 до 26 °C, вологість – до 60 %, рівень освітленості – не менше 300 люкс тощо. Блок-схема алгоритму порівняння показників з датчиків наведена у додатку А.

У разі виявлення перевищення порогів або подій, що потребують реагування (наприклад, рух у забороненій зоні, задимлення або витік газу), алгоритм генерує відповідне подійне повідомлення, яке передається на центральний контролер для подальшої обробки. При цьому застосовується пріоритезація подій, що дозволяє оперативно реагувати на критичні ситуації, зокрема аварійні, з одночасним ігноруванням менш значущих або дублюючих сигналів.

Подальшим кроком є ініціація відповідних дій – автоматичне керування виконавчими механізмами (вмикання/вимикання освітлення, вентиляції, подача тривожних сигналів) або сповіщення користувача через інтерфейс управління або мобільний застосунок.

Для забезпечення надійності роботи алгоритму реалізовано механізм журналювання подій, який дозволяє зберігати історію отриманих даних і реакцій системи. Це створює умови для аналізу ефективності, усунення помилок і подальшого удосконалення логіки управління.

Загалом, запропонований алгоритм обробки подій від сенсорів є основою для побудови адаптивної та безпечної системи керування «розумним будинком», що гарантує своєчасне виявлення змін у середовищі та адекватну реакцію на них.

2.3 Формалізація програмної моделі з використанням FreeRTOS

Для реалізації програмної частини системи керування «розумним будинком» було обрано операційну систему реального часу FreeRTOS, що забезпечує ефективне управління багатозадачністю та високу надійність роботи вбудованих систем. Використання FreeRTOS дозволяє формалізувати програмну модель системи шляхом розподілу функціональних блоків на окремі завдання (англ. tasks), які виконуються паралельно або з певним пріоритетом.

Програмна модель системи структурована як набір завдань, кожна з яких відповідає за обробку конкретного функціонального напрямку: збір даних із сенсорів, обробка подій, керування виконавчими механізмами, комунікація з користувачем, протоколювання подій та діагностика стану. Такий підхід забезпечує ізоляцію функціональних підсистем, що підвищує модульність, зручність тестування та масштабування системи.

В основі моделі лежить механізм планування завдань з використанням пріоритетів, де більш критичні процеси (наприклад, обробка тривожних сигналів датчиків диму або газу) отримують вищий пріоритет виконання порівняно з фоновими операціями (збір статистики або оновлення інтерфейсу). Це гарантує своєчасне реагування на аварійні події та підтримує стабільність роботи системи (рис. 2.12).

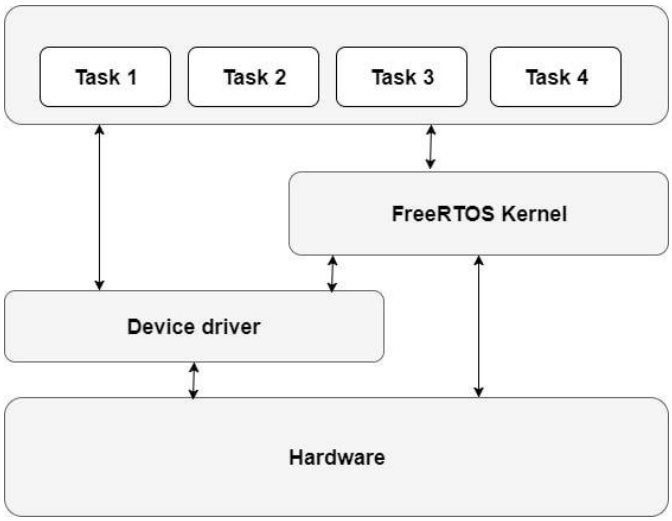

Рисунок 2.12 – Архітектура моделі FreeRTOS [18]

Для взаємодії між завданнями використовується набір засобів синхронізації, які надає FreeRTOS: черги (англ. queues) для передачі повідомлень і даних, семафори (англ. semaphores) та м’ютекси (англ. mutexes) для захисту спільних ресурсів від одночасного доступу. Такий підхід мінімізує ризики гонок даних і підвищує надійність.

Програмна модель передбачає також використання таймерів FreeRTOS для реалізації періодичних операцій, таких як опитування сенсорів або оновлення статусу системи. Таймери дозволяють забезпечити точність та регулярність виконання цих завдань без додаткового навантаження на основний цикл обробки.

Завдяки вбудованим засобам налагодження та моніторингу FreeRTOS забезпечує прозорість виконання завдань, що є критично важливим на етапах розробки та тестування. Інструменти відслідковування стану завдань, часу їх виконання та пріоритетів сприяють оптимізації продуктивності та стабільності системи.

Отже, формалізація програмної моделі на базі FreeRTOS є основою для побудови високонадійної, масштабованої та ефективної системи керування «розумним будинком», що відповідає сучасним вимогам до автоматизації побутових середовищ.

2.3.1 Використання FreeRTOS у системі

Використання операційної системи реального часу FreeRTOS у системі керування «розумним будинком» забезпечує надійність, гнучкість та ефективність роботи програмного забезпечення. FreeRTOS є широко застосовуваною вбудованою ОС, що має малий обсяг коду, мінімальні вимоги до ресурсів та підтримує багатозадачність із пріоритетним плануванням.

У контексті розробленої системи FreeRTOS виконує функції керування кількома паралельно працюючими завданнями, що відповідають за різні функціональні компоненти: збір та обробку даних з сенсорів (температури, вологості, освітленості, диму, руху), керування виконавчими пристроями

(освітлення, вентиляція, сигналізація), взаємодію з користувачем через інтерфейси, а також діагностику і логування подій.

Завдяки використанню механізму пріоритетів FreeRTOS система має можливість оперативно реагувати на критичні події, наприклад, спрацювання датчика диму чи руху, забезпечуючи при цьому паралельне виконання фонових завдань без втрати продуктивності. Кожне завдання ізольованк у власному контексті, що підвищує стійкість до збоїв і полегшує тестування та підтримку коду.

Для забезпечення синхронізації та обміну даними між завданнями застосовуються стандартні механізми FreeRTOS – черги, семафори та м’ютекси. Це дозволяє уникнути проблем з одночасним доступом до спільних ресурсів, а також забезпечує коректну послідовність обробки подій.

Крім того, FreeRTOS підтримує використання таймерів, які застосовуються для реалізації періодичних опитувань сенсорів і оновлення інформації у користувацькому інтерфейсі. Такий підхід дозволяє підтримувати актуальний стан системи без зайвого навантаження на центральний процесор.

2.3.2 Механізми обміну даними

Для забезпечення ефективної та надійної взаємодії між окремими підсистемами та завданнями в системі керування «розумним будинком», побудованій на базі FreeRTOS, використовуються стандартні механізми обміну даними, передбачені цією операційною системою реального часу. Основними інструментами для організації комунікації та синхронізації між завданнями є черги, семафори та м’ютекси.

Черги у FreeRTOS слугують для передачі повідомлень або структурованих даних між завданнями або між перериваннями і завданнями. Вони гарантують послідовний та безпечний обмін інформацією, запобігаючи одночасному доступу, що особливо важливо при обробці сигналів від сенсорів, де точність і цілісність даних є критичними. Використання черг

дозволяє реалізувати модель «виробник-споживач», де завдання, що збирають дані, передають їх у чергу для подальшої обробки або збереження.

Семафори застосовуються для сигналізації подій і контролю доступу до ресурсів. Вони забезпечують механізм блокування завдань, які очікують на певні умови або на звільнення ресурсу, тим самим уникаючи станів гонки (англ. race conditions) та підвищуючи стабільність роботи системи. У системі «розумного будинку» семафори можуть використовуватись для синхронізації між обробниками апаратних переривань від сенсорів і відповідними завданнями обробки.

М’ютекси є спеціалізованим видом семафорів, які дозволяють забезпечити ексклюзивний доступ до спільних ресурсів, наприклад, до змінних конфігурації або журналів подій, що редагуються одночасно кількома завданнями. Впровадження м’ютексів дозволяє уникнути пошкодження даних і забезпечити коректність функціонування програмної логіки.

Крім цих основних механізмів, у системі також застосовуються таймери FreeRTOS, що дозволяють запускати періодичні або відкладені функції, які оновлюють стан системи, опитують сенсори чи ініціюють інші регламентовані дії.

Загалом, застосування вбудованих у FreeRTOS механізмів обміну даними і синхронізації забезпечує ефективну взаємодію складових системи, високу реактивність до подій, а також стабільність і масштабованість програмної архітектури «розумного будинку».

2.3.3 Формування логічної моделі взаємодії завдань

Формування логічної моделі взаємодії завдань у системі керування «розумним будинком» базується на структурованому розподілі функціональних обов’язків між окремими завданнями операційної системи реального часу FreeRTOS. Кожне завдання відповідає за виконання певного набору операцій, що дозволяє забезпечити паралельність обробки даних,

оптимізувати використання апаратних ресурсів і підвищити надійність системи.

Основою логічної моделі є визначення ключових завдань: збір даних із сенсорів, обробка подій, керування виконавчими механізмами, оновлення інтерфейсу користувача та ведення журналу подій. Кожне з цих завдань взаємодіє з іншими через механізми обміну даними, описані раніше, що забезпечує синхронізацію та обмін інформацією без конфліктів.

Завдання збору даних виконується періодично або на основі подій від апаратних переривань. Вона отримує сигнали від сенсорів температури, вологості, освітленості, диму та руху, формує відповідні повідомлення та передає їх у черги для подальшої обробки. Завдання обробки подій зчитує інформацію з цих черг, аналізує отримані дані, приймає рішення щодо необхідних дій і активує відповідні виконавчі підсистеми.

Взаємодія завдань керування виконавчими механізмами полягає у прийманні команд від завдання обробки подій і безпосередньому керуванні зовнішніми пристроями – наприклад, увімкненням освітлення, вентиляції або сигналізації. Паралельно завдання оновлення інтерфейсу користувача забезпечує відображення актуальної інформації в режимі реального часу, отримуючи дані через відповідні черги або спільні структури.

Журналювання подій виконується спеціалізованим завданням, що записує історію змін у внутрішню базу даних, забезпечуючи можливість подальшого аналізу та формування звітів.

Реалізація логічної моделі у вигляді чітко структурованої взаємодії завдань сприяє підвищенню модульності програмного забезпечення, спрощенню відлагодження та подальшому масштабуванню системи. Такий підхід відповідає сучасним принципам розробки вбудованих систем і забезпечує надійну роботу «розумного будинку» у реальних умовах експлуатації.

Висновки до розділу 2

У другому розділі здійснено проєктування системи керування «розумним будинком», що базується на використанні мікроконтролера ESP32, датчиків навколишнього середовища.

Розроблено алгоритми реакції на зміни температури, вологості, освітленості, наявності диму, дотику та руху. Основною логікою управління є порівняння поточних показників із заздалегідь встановленими пороговими значеннями: наприклад, температура в межах 18–26 °C, вологість до 60 %, рівень освітленості не нижче 300 люкс тощо. При виявленні відхилень система приймає відповідне рішення: активує або вимикає виконавчі пристрої (вентилятор, освітлення), подає сигнали тривоги тощо.

Особливістю реалізованої системи є використання операційної системи реального часу FreeRTOS, яка дозволяє ефективно реалізувати багатозадачну обробку даних. Завдяки створенню окремих завдань для кожного функціонального блоку забезпечується високий рівень паралельності й стабільності роботи системи. Крім того, для синхронізації між завданнями застосовано черги, що дозволяє безпечно передавати дані між завданнями, уникати конфліктів доступу до спільних ресурсів і зберігати модульність системи.

Для підвищення мобільності реалізовано модуль Bluetooth-зв’язку, що забезпечує передачу інформації про стан системи до віддаленого користувача. Також передбачено відображення ключової інформації на OLED-дисплеї, що дозволяє локально контролювати параметри навколишнього середовища та поточний статус роботи системи.

У результаті проведеного проєктування визначено загальну архітектуру системи керування, що поєднує датчики, керувальні алгоритми та інтерфейси зв’язку.

3 АПАРАТНО-ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ «РОЗУМНОГО БУДИНКУ»

3.1 Розробка архітектури АПЗ

Розробка архітектури АПЗ є одним із ключових етапів створення системи «розумного будинку», що забезпечує ефективну, надійну та масштабовану роботу. Архітектурне проєктування визначає структуру системи, взаємозв’язки між її компонентами, принципи їх взаємодії, а також розподіл функціональних завдань між апаратною та програмною складовими.

Правильна архітектура АПЗ дозволяє забезпечити модульність, гнучкість та простоту подальшої модернізації системи, а також створює передумови для інтеграції додаткових компонентів. Враховуючи особливості управління «розумним будинком», архітектура розроблюваної системи повинна гарантувати надійність збору, обробки та передачі даних від сенсорних елементів, оперативне реагування на події, а також зручність взаємодії користувача з інтерфейсом.

У даному підрозділі наведено опис основних структурних елементів АПЗ, визначено їх функціональне призначення, а також розглянуто схеми взаємодії між модулями системи.

3.1.1 Загальна структура системи

АПЗ розробленої системи «розумного будинку» базується на платформі ESP32, що забезпечує інтеграцію широкого спектру сенсорів та виконавчих пристроїв із бездротовим керуванням через Bluetooth. Центральним елементом системи виступає мікроконтролер ESP32, що виконує функції збору даних, їх обробки, реалізації логіки управління та комунікації з користувачем.

Система включає наступні апаратні компоненти: датчик температури і вологості DHT11, Light Dependent Resistor (LDR) для вимірювання інтенсивності освітлення, ультразвуковий датчик для виявлення присутності

людини біля дверей, сенсор диму MQ2 для моніторингу потенційних загроз пожежі, а також вбудований сенсор дотику ESP32 для виявлення дотику до дверей із сигналізацією тривоги. Виконавчі пристрої – вентилятор та освітлення – керуються у трьох режимах: вручну, автоматично (на основі показників сенсорів) та вимкнено.

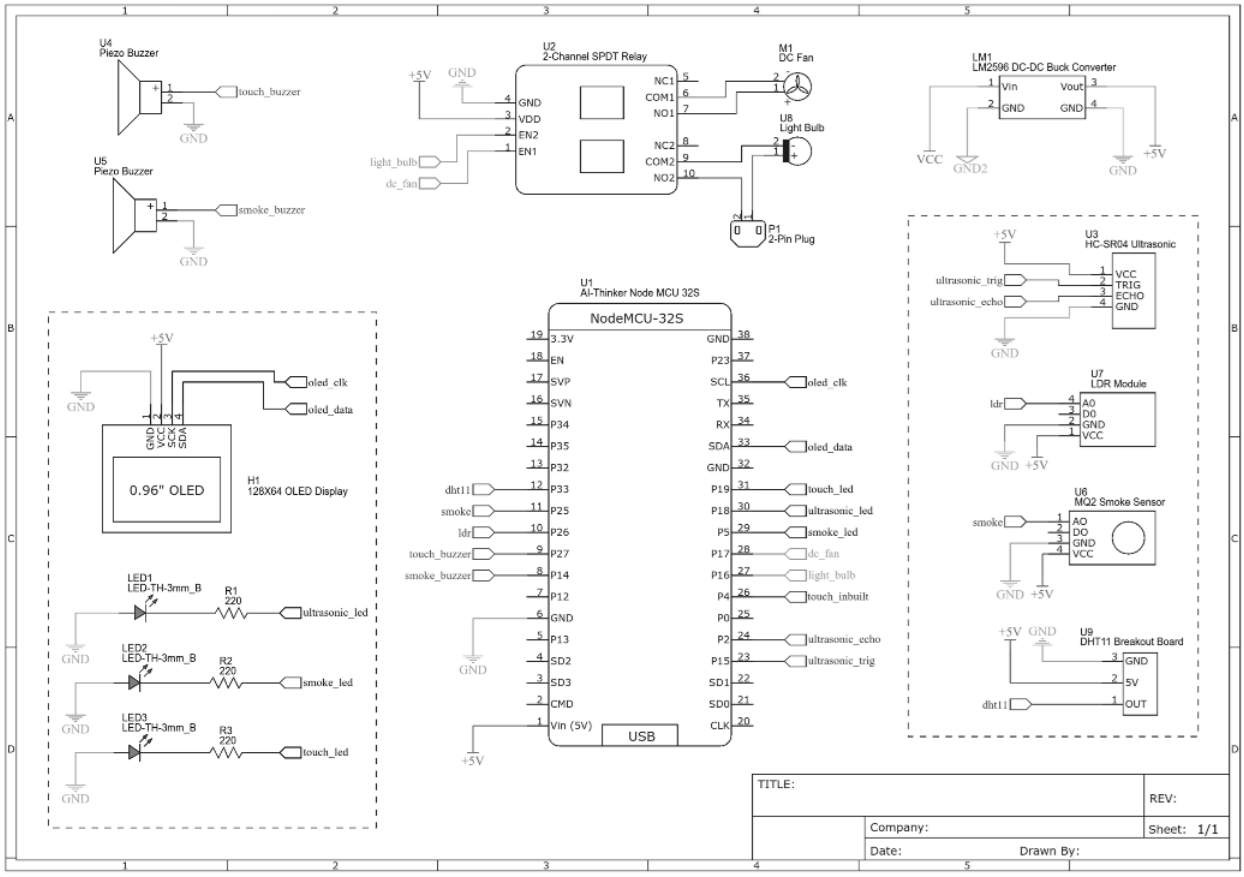

Рисунок 3.1 – Схема системи

Збір і обробка даних від сенсорів відбувається в режимі реального часу, що дозволяє системі швидко реагувати на зміни у навколишньому середовищі. Відображення інформації здійснюється на OLED-дисплеї, а також у спеціальному мобільному застосунку для Android, який забезпечує зручний інтерфейс для дистанційного моніторингу та керування.

Безпровідна комунікація за допомогою Bluetooth забезпечує гнучкість розміщення компонентів і швидкий доступ користувача до системи без необхідності складного налаштування мережі.

Отже, загальна структура системи характеризується високою інтеграцією апаратних модулів, ефективною організацією програмного

управління на базі ESP32, а також зручними інтерфейсами взаємодії, що разом гарантує надійну, масштабовану та комфортну систему «розумного будинку».

3.1.2 Опис основних модулів і їх взаємодії

АПЗ системи складається з низки функціональних модулів, кожен з яких виконує специфічні завдання, що забезпечують комплексне керування побутовим середовищем. Основними модулями системи є:

– модуль збору даних із сенсорів – включає датчики температури та вологості (DHT11), освітленості (LDR), диму (MQ2), ультразвуковий сенсор (HC-SR04) для виявлення руху та центр керування ESP32;

– модуль обробки даних і прийняття рішень – реалізований на базі мікроконтролера ESP32, цей модуль обробляє отримані від сенсорів дані, аналізує поточний стан середовища та приймає рішення про включення або вимкнення виконавчих пристроїв, таких як вентилятор чи освітлення. Логіка управління підтримує три режими роботи: ручний, автоматичний (залежно від показників сенсорів) та вимкнений;

– модуль інтерфейсу користувача – забезпечує відображення актуальної інформації на OLED-дисплеї, а також взаємодію з користувачем через мобільний застосунок для Android. Цей модуль відповідає за прийом команд від користувача, а також за відправку повідомлень про критичні події, наприклад, виявлення диму або спробу дотику до дверей;

– комунікаційний модуль – реалізує бездротовий зв’язок через Bluetooth, що дозволяє системі обмінюватися даними з мобільним застосунком, а також забезпечує дистанційне керування та моніторинг.

Взаємодія між модулями відбувається у реальному часі за допомогою обміну повідомленнями та подіями. Модуль збору даних регулярно передає виміри модулю обробки, який аналізує інформацію і, залежно від заданих правил і сценаріїв, активує відповідні виконавчі пристрої. Інформація про стан системи оновлюється на OLED-дисплеї і передається в мобільний застосунок через комунікаційний модуль.

3.2 Вибір технологій та мов програмування

Вибір відповідних технологій та мов програмування є ключовим етапом розробки АПЗ, оскільки визначає ефективність, надійність та зручність подальшої реалізації системи. У цьому підрозділі обґрунтовується вибір інструментів розробки, що відповідають функціональним вимогам поставленого завдання з розробки системи керування «розумним будинком», апаратним ресурсам обраного мікроконтролера, а також забезпечують підтримку необхідних протоколів зв’язку та можливість інтеграції з мобільним застосунком.

3.2.1 Огляд можливих технологій і платформ

На етапі вибору технологій для розробки системи «розумний будинок» було розглянуто декілька апаратних платформ та програмних середовищ, які здатні забезпечити необхідний рівень функціональності, продуктивності та гнучкості. Серед основних апаратних рішень аналізувалися мікроконтролери сімейства Arduino (наприклад, Arduino Uno, Arduino Mega), платформи на базі ESP32 та Raspberry Pi.

Arduino має широку спільноту, численні бібліотеки та простоту у використанні, проте обмежені можливості бездротового зв’язку та обчислювальної потужності. ESP32 виділяється вбудованим модулем Bluetooth і Wi-Fi, двоядерним процесором, що робить його ідеальним для реалізації бездротового управління та обробки кількох сенсорів у реальному часі. Raspberry Pi надає ще більшу обчислювальну потужність і можливість запуску повноцінної операційної системи, проте його апаратна складність і енергоспоживання можуть бути надмірними для завдань простого сенсорного моніторингу.

Щодо програмних технологій, розглядалися мови програмування C/C++ (зокрема для Arduino та ESP32), Python (переважно для Raspberry Pi), а також платформи розробки мобільних застосунків для Android (Java/Kotlin) та iOS (Swift). Для організації бездротового зв’язку обирали протоколи Bluetooth Low

Energy (BLE), Wi-Fi та Zigbee, які мають різні переваги залежно від сценаріїв використання.

Порівняльний аналіз дозволив визначити, що ESP32 є оптимальним вибором для апаратної частини проєкту завдяки поєднанню продуктивності, вбудованих засобів бездротового зв’язку та підтримці необхідних периферійних пристроїв. Для програмування контролера використано мову C++ із застосуванням фреймворку ESP-IDF. Мобільний застосунок розроблено на платформі Android із використанням мови Kotlin, що забезпечує зручний і сучасний інтерфейс користувача.

3.2.2 Обґрунтування вибору мов програмування

Вибір мов програмування для розробки АПЗ є критично важливим для забезпечення ефективності, надійності та масштабованості системи «розумний будинок». Основним завданням було визначити мову, яка забезпечить максимальний контроль над апаратними ресурсами, низький рівень затримок у обробці подій, а також підтримку необхідних апаратних інтерфейсів.

Для програмування мікроконтролера ESP32 розглядалися дві основні мови: C і C++. Обидві мови забезпечують високу продуктивність і є стандартом у вбудованому програмуванні. Однак C++ має суттєві переваги у порівнянні з C, зокрема:

– підтримка об’єктно-орієнтованого програмування (ООП), що дозволяє краще структурувати код, моделювати апаратні компоненти у вигляді класів та створювати гнучкіші і розширювані архітектури програмного забезпечення.

– наявність стандартної бібліотеки STL, яка значно полегшує роботу зі структурами даних, алгоритмами та контейнерами, скорочуючи час розробки.

– підтримка абстракцій високого рівня, що дозволяє ізолювати апаратно-залежний код від логіки управління, сприяючи кращій підтримці та масштабованості проєкту.

При цьому C++ зберігає можливість прямого доступу до низькорівневих апаратних ресурсів, що критично для реалізації функцій роботи з сенсорами та бездротовими інтерфейсами.

У порівнянні з іншими мовами, такими як Python або Java, які можуть використовуватися у деяких вбудованих рішеннях, C++ суттєво випереджає їх за показниками швидкодії, розміром коду та споживанням ресурсів. Наприклад, Python, хоч і має більшу швидкість розробки, є інтерпретованою мовою з великим обсягом ресурсів, що неприйнятно для ресурсозалежних систем на ESP32. Java має високі вимоги до середовища виконання та споживання пам’яті, що також ускладнює її використання у малоресурсних пристроях.

Таким чином, вибір C++ для реалізації АПЗ ESP32 базується на компромісі між високою продуктивністю, гнучкістю архітектури та широкою підтримкою апаратних інтерфейсів. Це дозволяє забезпечити стабільність роботи системи, ефективну обробку подій від сенсорів та інтеграцію з Bluetooth-модулем для бездротового керування.

3.2.3 Використання операційної системи реального часу

Для забезпечення надійної та своєчасної обробки подій у системі «розумний будинок» було прийнято рішення застосувати операційну систему реального часу (ОСРЧ) – FreeRTOS. Використання ОСРЧ є обґрунтованим у зв’язку з вимогами до синхронізації роботи численних апаратних модулів, гарантованої обробки пріоритетних подій від сенсорів та підтримки багатозадачності на ресурсно-обмеженому мікроконтролері ESP32.

FreeRTOS забезпечує низку переваг, які критично важливі для вбудованих систем:

– детермінованість виконання завдань – час виконання кожного завдання можна контролювати, що гарантує обробку критичних подій у реальному часі без затримок;

– підтримка пріоритетів завдань – дозволяє реалізувати ефективне планування, в якому, наприклад, обробка сигналів диму або руху матиме вищий пріоритет ніж періодичне оновлення інтерфейсу користувача;

– механізми синхронізації та комунікації між завданнями – семафори, черги повідомлень та м’ютекси забезпечують безпечний обмін даними між різними підсистемами без ризику станів гонитви чи блокувань;

– невеликі розміри ядра та низькі системні накладні витрати, що відповідає ресурсним обмеженням мікроконтролера ESP32.

FreeRTOS має широку підтримку на платформі ESP32, а також добре документований API, що прискорює розробку і тестування програмного забезпечення. Впровадження ОСРЧ дозволяє структурувати програмне забезпечення у вигляді окремих завдань, кожне з яких відповідає за певний функціонал – збір даних з сенсорів, обробку сигналів тривоги, керування виконавчими механізмами, взаємодію з Bluetooth-модулем та оновлення інтерфейсу користувача.

Таким чином, застосування FreeRTOS у проєкті забезпечує необхідну гнучкість, масштабованість та надійність системи, що є основою для реалізації інтелектуального керування побутовим середовищем у режимі реального часу.

3.3 Вибір компонентів АПЗ

На цьому етапі розробки системи «розумного будинку» було здійснено аналіз та підбір апаратних і програмних компонентів, що забезпечать виконання поставлених функціональних вимог з урахуванням технічних, економічних та експлуатаційних факторів. Вибір компонентів базувався на критеріях сумісності, енергоспоживання, продуктивності, надійності та можливості інтеграції в єдину архітектуру системи.

Особливу увагу було приділено підбору сенсорів, виконавчих пристроїв, комунікаційних інтерфейсів, а також програмних бібліотек і фреймворків, що забезпечують ефективну роботу системи в режимі реального часу.

3.3.1 Апаратні елементи

У процесі розробки АПЗ системи «розумного будинку» було здійснено вибір апаратних компонентів, що відповідають технічним вимогам, можуть забезпечити надійність, точність вимірювань, а також інтегруватися у загальну систему з урахуванням енергоспоживання та вартості.

Основним обчислювальним вузлом системи обрано мікроконтролер ESP32 від компанії Espressif (рис. 3.2). Він поєднує в собі високу продуктивність, 32-бітний процесор з частотою до 240 МГц, великий обсяг оперативної пам’яті та широкий набір периферійних інтерфейсів (UART, SPI, I^2C, GPIO).

Рисунок 3.2 – Модуль ESP32

Найважливішою перевагою ESP32 є вбудована підтримка бездротових протоколів Wi-Fi і Bluetooth, що забезпечує гнучкість при впровадженні дистанційного керування та моніторингу системи через мобільний застосунок. FreeRTOS, що реалізована для ESP32, дозволяє ефективно управляти багатозадачністю й забезпечувати надійну роботу системи в реальному часі.

3.3.1.1 Вимірювання температури та вологості у приміщенні

Для вимірювання температури та вологості було обрано датчик DHT11. Хоча DHT11 не є найточнішим сенсором серед доступних на ринку, він володіє достатньою точністю для побутових застосувань (похибка температури ±2 °C, вологості ±5 %) і характеризується простотою

підключення, низькою вартістю та енергоефективністю. Ці характеристики роблять його оптимальним вибором для систем «розумного будинку», де точність вимірювань не є критичною, а на першому плані стоїть економія ресурсів.

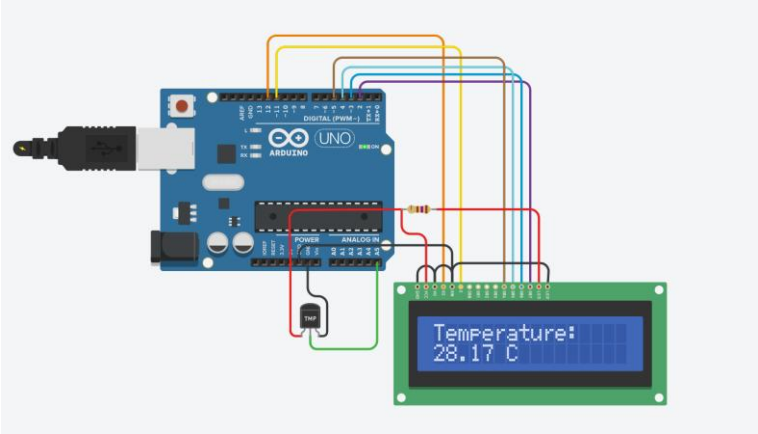

Рисунок 3.3 – Складання схеми та моделювання роботи температурного датчику у середовищі TinkerCad

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

#define temp A5
#define led 13

void setup()
{
  lcd.begin(16, 2);
  pinMode(led, OUTPUT);
  pinMode(temp, INPUT);
  Serial.begin(9600);
  lcd.clear();
  lcd.print("Temperature: ");
}

float pre_temp = 0;
void loop() {
  float temperature = 0;
  temperature = (analogRead(temp) * 0.48828125) - 49.95;
  if(pre_temp != temperature)
  {
    lcd.setCursor(0,1);
    lcd.print("                ");
  }
  lcd.setCursor(0,1);
  lcd.print(temperature);
  lcd.print(" C");
  pre_temp = temperature;
}
```

Рисунок 3.4 – Програмний код керування температурним датчиком

Даний програмний код (рис. 3.4) реалізує функцію моніторингу температури з виведенням даних на LCD-дисплей та реакції через світлодіод. У коді використовується бібліотека LiquidCrystal.h для керування LCD-дисплеєм, підключеним до цифрових пінів мікроконтролера. У функції *setup()* виконується початкова ініціалізація дисплея, задання пінів для світлодіода та

аналогового термодатчика. Також відкривається послідовний порт для можливого виведення діагностичних даних через комп’ютер Основна логіка виконується у циклі *loop()*, який повторюється безперервно. У кожній ітерації зчитується аналогове значення температури з термодатчика, яке перетворюється у фізичну температуру за допомогою калібрувальної формули.

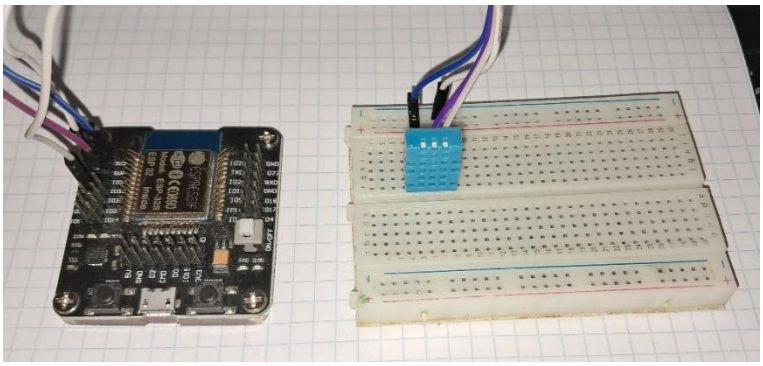

Рисунок 3.5 – Складання елементу системи з датчиком температури

Виконано також збірку модуля ESP32 з датчиком температури й вологості DHT11 на макетній платі. Така конфігурація є типовим прототипом, що дозволяє оперативно перевірити працездатність апаратної частини та налагодити передачу даних між компонентами та використовувати у якості елемента системи «розумного будинку».

3.3.1.2 Керування освітленістю у системі «розумного будинку»

Для оцінки рівня освітленості застосовується резистивний фотодіод LDR. Цей датчик змінює свій опір залежно від інтенсивності світла, що робить його простим і дешевим рішенням для виявлення змін у природному та штучному освітленні. Використання LDR дозволяє реалізувати автоматичне регулювання освітлення приміщень, що сприяє підвищенню енергоефективності системи.

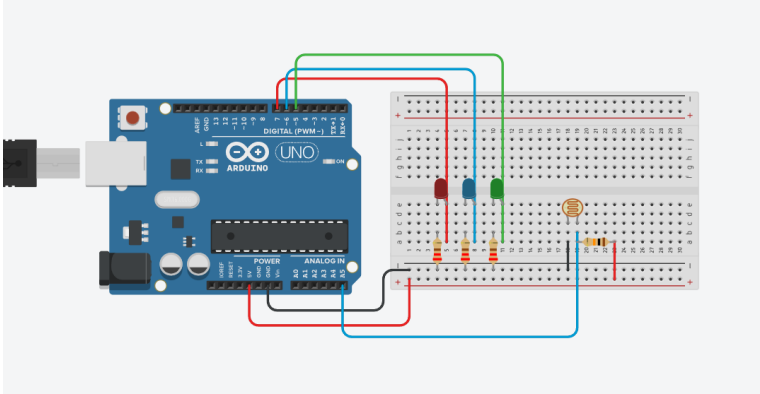

Рисунок 3.6 – Складання схеми з використанням резистивного фотодіоду

Для реалізації базової системи контролю рівня освітленості з використанням фоторезистора та трьох світлодіодів, які сигналізують про різні рівні освітлення, реалізовано програмний код (рис. 3.7). Логіка роботи базується на зчитуванні аналогового сигналу з датчика освітленості та вмиканні відповідного світлодіода залежно від рівня освітлення в навколишньому середовищі.

У блоці *setup()* здійснюється початкова конфігурація: задаються три цифрові порти, як виходи для керування світлодіодами, а також порт вхід для зчитування даних з фоторезистора. Додатково ініціалізується послідовний порт (Serial.begin(9600)) для виведення інформації про зчитані значення.

```
int ledRed = 7;
int ledBlue = 6;
int ledGreen = 5;

int sensorLDR = A5;

void setup()
{
  pinMode(ledRed, OUTPUT);
  pinMode(ledBlue, OUTPUT);
  pinMode(ledGreen, OUTPUT);
  pinMode(sensorLDR, INPUT);
  Serial.begin(9600);
}

void loop()
{
  int leitura = analogRead(sensorLDR);
  Serial.print("Reading: ");
  Serial.println(leitura);

  if(analogRead(sensorLDR) > 600){
    digitalWrite(ledRed, HIGH);
    digitalWrite(ledBlue, LOW);
    digitalWrite(ledGreen, LOW);
  }
  else if (analogRead(sensorLDR) >= 85 && analogRead(sensorLDR) <= 170){
    digitalWrite(ledBlue, HIGH);
    digitalWrite(ledRed, LOW);
    digitalWrite(ledGreen, LOW);
  }
  else{
    digitalWrite(ledBlue, LOW);
    digitalWrite(ledRed, LOW);
    digitalWrite(ledGreen, HIGH);
  }
```

Рисунок 3.7 – Програмний код керування роботою фотодіоду

Основна логіка реалізована у циклі *loop()*, який постійно виконується. На кожній ітерації за допомогою функції *analogRead(sensorLDR)* зчитується

поточне значення освітленості. Далі реалізовано умовну структуру прийняття рішень:

– якщо значення освітленості більше за 600, система вважає, що рівень світла високий, тому вмикається червоний світлодіод, а синій та зелений вимикаються;

– якщо значення знаходиться в межах від 85 до 170 включно, тобто освітленість помірна, активується синій світлодіод, а інші вимикаються;

– в усіх інших випадках (наприклад, при дуже низькому або середньо-низькому рівні освітленості) вмикається зелений світлодіод, а червоний і синій залишаються вимкненими.

Таким чином, система забезпечує візуальну індикацію стану освітлення в середовищі.

3.3.1.3 Виявлення диму та газу у приміщенні

Датчик MQ-2 був обраний для детекції диму та газів, що є критичною функцією безпеки у будь-якій побутовій системі. MQ-2 забезпечує чутливість до широкого спектру газів, включаючи чадний газ, пропан, бутан та дим, що дає змогу своєчасно виявляти потенційні загрози і активувати сигналізацію. Крім того, MQ-2 має простий аналоговий вихід, що дозволяє легко інтегрувати його з мікроконтролером без потреби у складних перетворювачах.

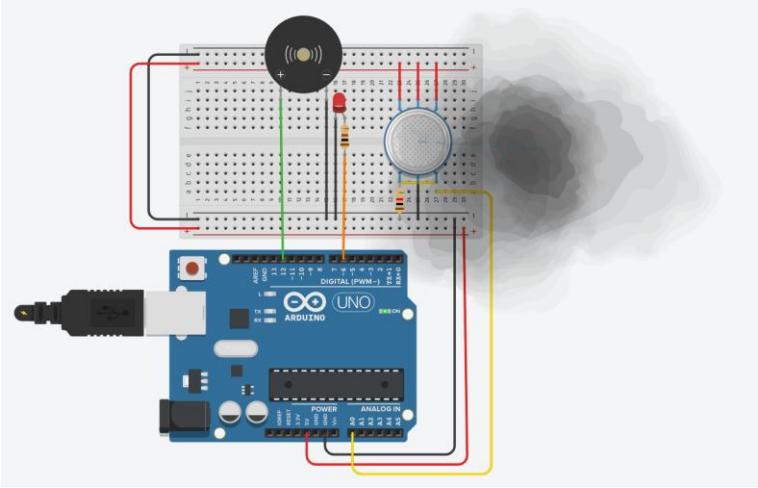

Рисунок 3.8 – Складання схеми та моделювання роботи з датчиком виявлення диму та газу

Програмний код роботи з датчиком MQ-2 (рис. 3.9) реалізує базову систему виявлення концентрації газу з використанням аналогового газового сенсора, світлодіода та звукового сигналізатора. Основним завданням є в режимі реального часу моніторити рівень газу в повітрі та відповідно до цього змінювати яскравість світлодіода і гучність/частоту звукового сигналу. У функції *setup()* виконується ініціалізація виводів: порт 6 використовується для керування світлодіодом, а порт 12 – для керування базером.

```
const int gas_input = A0;
int gas = 0;
const int led = 6;
const int buzzer = 12;

void setup()
{
  pinMode(led,OUTPUT);
  pinMode(buzzer,OUTPUT);

  Serial.begin(9600);
}

void loop()
{
  gas  = analogRead(gas_input);

  Serial.println(gas);

  int led_out = map(gas, 80, 400, 0, 255);

  tone(buzzer,led_out,100);

  analogWrite(led,led_out);

  delay(100);
}
```

Рисунок 3.9 – Програмний код для роботи з датчиком

Також запускається послідовний порт для виводу показників датчика в монітор серійного порту (Serial Monitor). У циклі *loop()* реалізована основна логіка:

– зчитується поточне значення з газового сенсора, підключеного до аналогового входу. Зчитане значення є пропорційним концентрації газу;

– отримане значення виводиться в консоль за допомогою *Serial.println(gas)* для моніторингу;

– за допомогою функції *map()* зчитане значення масштабуються з діапазону 80–400 у новий діапазон 0–255, де 80 – умовний поріг низької концентрації газу, а 400 – умовний поріг високої концентрації;

– застосовується функція *tone(buzzer, led_out, 100)*, що генерує звуковий сигнал на базері з частотою, відповідною рівню газу – чим вища концентрація, тим вищий тон сигналу;

– одночасно відбувається аналогове керування світлодіодом за допомогою *analogWrite(led, led_out)*. Яскравість світлодіода також залежить від рівня газу.

На завершення кожного циклу реалізується затримка в 100 мс, що дозволяє стабілізувати частоту оновлення даних.

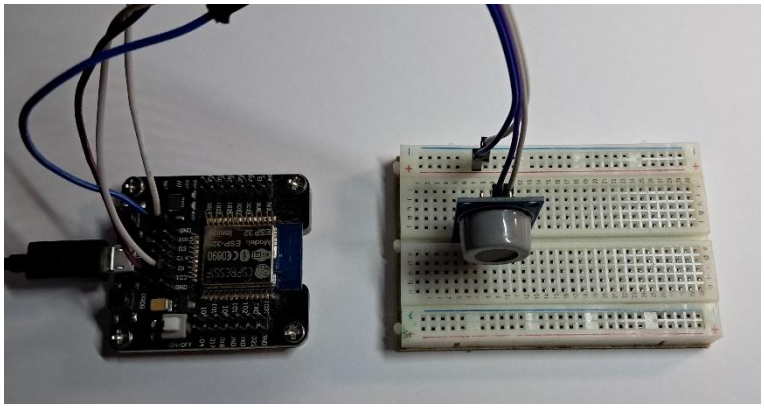

Рисунок 3.10 – Прототип зібраного компонента з датчиком MQ-2

Виконано також прототипування компонента з датчиком MQ-2, що підключений до модуля мікроконтролера ESP32. Дана збірка дозволяє провести попереднє тестування роботи сенсора з виявлення диму та газу. У подальшому конструкція може бути адаптована для постійного використання в системі моніторингу повітря в межах «розумного будинку».

3.3.1.4 Виявлення присутності та руху людей в будинку

Для виявлення присутності людини біля дверей було обрано ультразвуковий датчик HC-SR04. Цей сенсор є широко розповсюдженим та перевіреним рішенням для безконтактного визначення відстані до об’єктів. HC-SR04 забезпечує точність вимірювання до 1500 мм, що цілком відповідає вимогам системи охорони та автоматизації входу.

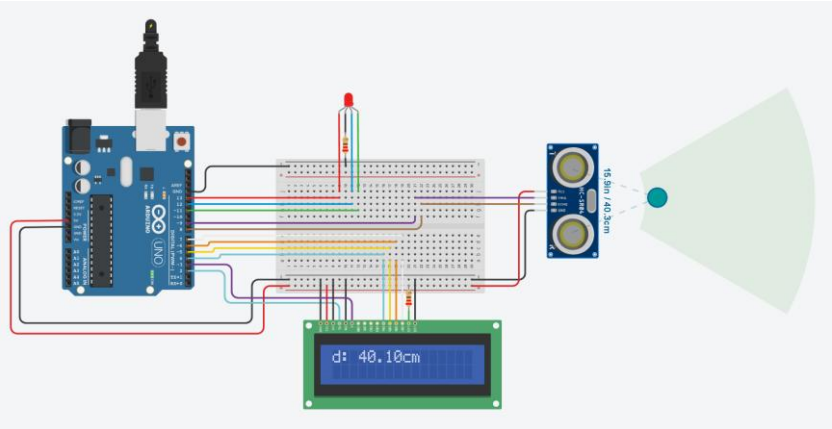

Рисунок 3.11 – Збірка схеми та моделювання роботи компонентів у TinkerCad

Для роботи із датчиком ультразвуку реалізовано програмний код, що забезпечує функціональність модуля відстеження присутності людей у приміщенні на основі сенсора HC-SR04 та візуальної індикації за допомогою світлодіодів. Такий модуль є складовою частиною системи керування «розумним будинком» і може використовуватися, зокрема, для автоматичного керування освітленням, охоронними системами або контролем доступу.

```
#include <LiquidCrystal.h>

const int triggerPin = 9;
const int echoPin    = 8;

const int greenLedPin  = 11;
const int blueLedPin   = 12;
const int redLedPin    = 13;

const int registerSelectPin = 2;
const int enablePin = 3;

const int DB4 = 4;
const int DB5 = 5;
const int DB6 = 6;
const int DB7 = 7;

LiquidCrystal lcd(
  registerSelectPin,
  enablePin,
  DB4,
  DB5,
  DB6,
  DB7
);

void setup() {
  Serial.begin(9600);
  pinMode(triggerPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(greenLedPin, OUTPUT);
  pinMode(blueLedPin, OUTPUT);
  pinMode(redLedPin, OUTPUT);

  lcd.begin(16, 2);
}
```

Рисунок 3.12 – Програмний код роботи з ультразвуковим датчиком у TinkerCad

У функції *setup()* виконується ініціалізація:

– ультразвукового датчика HC-SR04: пін 9 використовується для генерації ультразвукового імпульсу, а пін 8 – для приймання відбитого сигналу;

– світлодіодів трьох кольорів, що сигналізують про наявність або відсутність руху (відповідно до відстані до об’єкта);

– LCD-дисплея, на якому відображається відстань до об’єкта в сантиметрах.

```
void loop() {
  triggerUltraSound();

  double distance = calculateDistance();

  printDistanceInSerialPort(distance);

  setLedColorBasedOnDistance(distance);

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("d: ");
  lcd.print(distance);
  lcd.print("cm");

  delay (200);
}

void triggerUltraSound() {
  digitalWrite(triggerPin,LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin,HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin,LOW);
}

double calculateDistance() {
  double duration = pulseIn(echoPin, HIGH);
  double distance = duration * 0.0343 / 2;
}

void printDistanceInSerialPort(double distance) {
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
}
```

Рисунок 3.13 – Програмний код моделювання роботи датчику у TinkerCad

У циклі *loop()* реалізовано основну логіку:

– генерується імпульс ультразвукового сигналу (triggerUltraSound()), який спрямовується в простір;

– функція *calculateDistance()* вимірює час проходження сигналу до об’єкта і назад, після чого обчислюється відстань до нього. Виміряне значення виводиться в монітор серійного порту (Serial Monitor) для діагностики;

– відповідно до отриманої відстані, функція *setLedColorBasedOnDistance(distance)* активує певний світлодіод, який візуально сигналізує про рівень наближення:

– якщо відстань понад 200 см, включається зелений світлодіод – приміщення вільне;

– якщо відстань у межах 100–200 см, включається жовте світло (поєднання зеленого і червоного) – можлива наявність об'єкта;

– якщо відстань менше 100 см, включається червоний світлодіод – виявлено рух або присутність людини.

На LCD-дисплей виводиться поточне значення відстані, що дає змогу користувачеві спостерігати за динамікою змін у реальному часі. Функція *delay(200)* задає інтервал між циклами опитування сенсора, що дозволяє стабільно оновлювати інформацію кожні 200 мілісекунд.

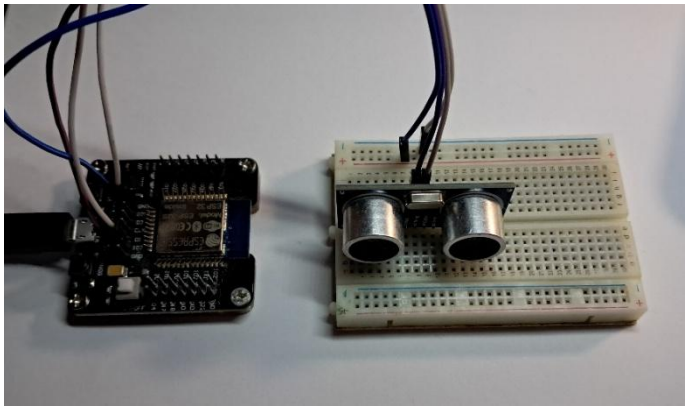

Рисунок 3.14 – Прототип зібраного компонента з датчиком HC-SR04

У тому числі, у системах керування «розумним будинком» використовується PIR-сенсор для виявлення присутності та руху людей у приміщенні (рис. 3.15). Даний датчик виявляє інфрачервоне випромінювання, що природно випромінюють живі істоти, зокрема люди. Він не випромінює сигналів самостійно, а лише реєструє зміни температурного фону у зоні спостереження. Якщо виявляється різниця в інфрачервоному випромінюванні (наприклад, людина проходить повз), то сенсор генерує цифровий сигнал високого рівня.

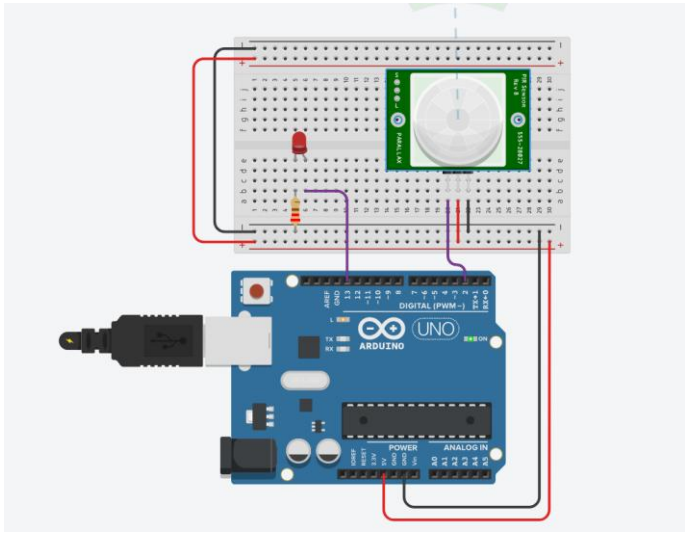

Рисунок 3.15 – Збірка схеми та моделювання роботи PIR-сенсору

Отриманий сигнал від PIR-сенсора передається до мікроконтролера або центрального модуля керування системою, який інтерпретує його як наявність руху або присутності людини. На основі цього можуть виконуватись різні дії: автоматичне вмикання або вимикання освітлення, активація системи безпеки, запуск вентиляції чи кондиціонування.

```
int sensorState = 0;

void setup()
{
  pinMode(2, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()

  sensorState = digitalRead(2);

  if (sensorState == HIGH) {
    digitalWrite(LED_BUILTIN, HIGH);
  } else {
    digitalWrite(LED_BUILTIN, LOW);
  }
  delay(10);
}
```

Рисунок 3.16 – Програмний код керування датчиком PIR

Програмний код реалізує базову логіка виявлення руху за допомогою PIR-сенсора. У функції *setup()* встановлюється режим роботи пінів: 2-й пін конфігуровано як вхідний, оскільки він приймає сигнал від PIR-сенсора, а вбудований світлодіод – як вихідний, адже він буде вмикатися чи вимикатися в залежності від сигналу з сенсора.

Далі, у циклі *loop()* зчитується стан сенсора за допомогою функції та зберігається в окремій для цього змінній *sensorState*. Якщо стан високий, тобто сенсор зафіксував рух, то на контролері вмикатиметься вбудований світлодіод. У протилежному випадку, якщо рух не зафіксовано, то світлодіод вимикається. Також, враховано невелику затримку в 10 мс, що забезпечує стабільність зчитування даних.

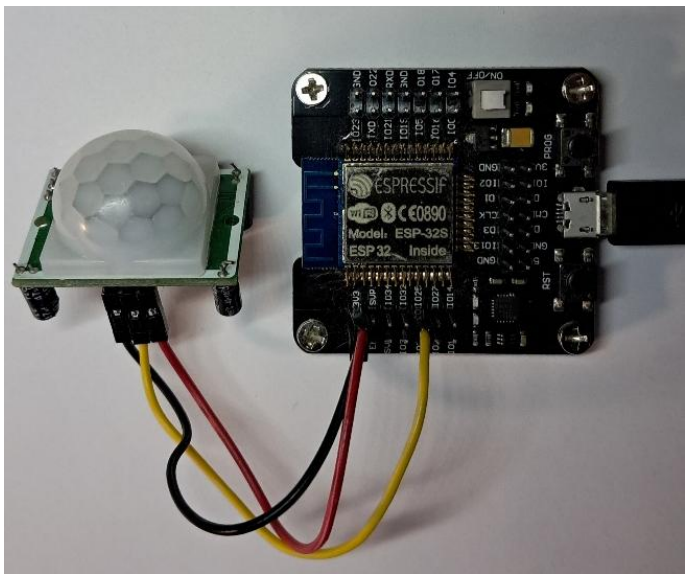

Рисунок 3.17 – Прототипування зібраного компонента

Усі компоненти були обрані з урахуванням сумісності з платформою ESP32, можливості живлення від одного джерела, а також легкості інтеграції в загальну систему через стандартні інтерфейси I^2C, UART та GPIO. Такий комплексний підхід забезпечує не лише високу функціональність та точність системи, а й її масштабованість та можливість подальшого розвитку.

3.3.2 Використані бібліотеки та фреймворки

Розробка АПЗ здійснювалась із застосуванням ряду спеціалізованих бібліотек і фреймворків, що забезпечують стабільну та ефективну взаємодію між апаратними компонентами та програмним забезпеченням. Вибір цих інструментів ґрунтувався на критеріях сумісності з апаратною платформою ESP32, відкритості, документованості, а також можливості реалізації багатозадачності і забезпечення реального часу.

1) Бібліотека BluetoothSerial.h використовується для реалізації бездротового зв’язку за протоколом Serial Port Profile (SPP) через Bluetooth. Дана бібліотека є офіційною частиною ESP-IDF (Espressif IoT Development Framework) і забезпечує високий рівень інтеграції з мікроконтролером ESP32, що дозволяє організувати стабільну послідовну передачу даних без додаткових апаратних засобів. Вибір BluetoothSerial обґрунтований

необхідністю забезпечення мобільного управління та моніторингу системи через Android-застосунок, що підвищує гнучкість використання розробленої системи.

2) Бібліотека Wire.h є стандартною Arduino-бібліотекою для роботи з шиною I^2C (Inter-Integrated Circuit). Вона відповідає за ініціалізацію та обмін даними між мікроконтролером і периферійними пристроями, такими як OLED-дисплей та інші датчики з інтерфейсом I^2C. Використання Wire.h обумовлене широким поширенням I^2C у вбудованих системах завдяки простоті підключення і можливості роботи з багатьма пристроями на одній шині.

3) Бібліотека DHT.h призначена для роботи з цифровими датчиками температури і вологості типу DHT11 та DHT22. Вона забезпечує коректне опитування сенсора, обробку сигналів і вивід точних значень температури і вологості. Використання цієї бібліотеки гарантує стабільність зчитування навіть за умови наявності шумів або нестабільних сигналів, що є критично важливим для систем контролю клімату і автоматизації.

4) Графічні бібліотеки Adafruit_SSD1306.h і Adafruit_GFX.h використовуються для роботи з OLED-дисплеєм на основі контролера SSD1306. Бібліотека Adafruit_GFX забезпечує універсальний графічний інтерфейс для виводу тексту, графіки та простих анімацій, а Adafruit_SSD1306 реалізує специфічні драйвери для дисплея розміром 128x64 пікселів. Таке поєднання дозволяє реалізувати зручний і наочний інтерфейс користувача для відображення стану системи в реальному часі.

5) Бібліотека Ticker.h використовується для організації таймерів із періодичним викликом функцій, що дозволяє реалізувати неблокуючий таймерний механізм. Це критично для багатозадачних систем, де необхідно виконувати регулярні операції (наприклад, опитування датчиків) без затримок і блокувань основного потоку виконання.

6) FreeRTOS – це операційна система (ОС) реального часу, інтегрована в середовище розробки ESP32. Вона забезпечує управління

багатозадачністю, пріоритетне планування та синхронізацію процесів, що дозволяє ефективно розподіляти ресурси мікроконтролера і реалізовувати паралельне виконання завдань, таких як читання сенсорів, керування виконавчими механізмами і взаємодія з користувачем. Використання FreeRTOS є критично важливим для підвищення продуктивності та стабільності роботи системи, особливо в умовах реального часу.

7) Заголовочні файли soc/soc.h та soc/rtc_cntl_reg.h містять низькорівневі визначення і функції, необхідні для управління системними регістрами мікроконтролера ESP32, включаючи контроль живлення і відключення функції brownout reset. Це дозволяє підвищити надійність роботи пристрою в умовах нестабільного живлення, що є важливим для забезпечення безперервної роботи системи без несподіваних перезавантажень.

Вибір бібліотек і фреймворків базувався на комплексній оцінці їх функціональних можливостей, стабільності, а також рівня підтримки та сумісності з апаратною платформою ESP32.

3.4 Реалізація системи керування компонентами «розумного будинку»

Для забезпечення керування системою «розумного будинку», пропонується розглянути рішення даного завдання шляхом реалізації застосунку на базі Android. Важливо відзначити, що використання ОС Android у системі керування «розумним будинком» є доцільним рішенням, з огляду на відкритість та можливості інтеграції з іншими інструментами керування. Android – є відкритою платформою розробки мобільних застосунків, що дозволяє створити адаптивний і функціональний інтерфейс для взаємодії з апаратною частиною системи.

Розроблений застосунок дозволить в режимі реального часу отримувати дані з сенсорів, контролювати стан освітлення у будинку, температури, рівня вологості, виявлення руху, газу тощо. Відповідно, користувач матиме змогу

здійснювати моніторинг і керування будинком через мобільний пристрій з будь-якого місця.

Для реалізації застосунку, у першу чергу необхідно розробити use case діаграму з метою визначення та візуалізації потрібних функціональних вимог системи, демонструючи взаємодію між користувачем і системою, у даному випадку системою керування «розумним будинком» (рис. 3.18).

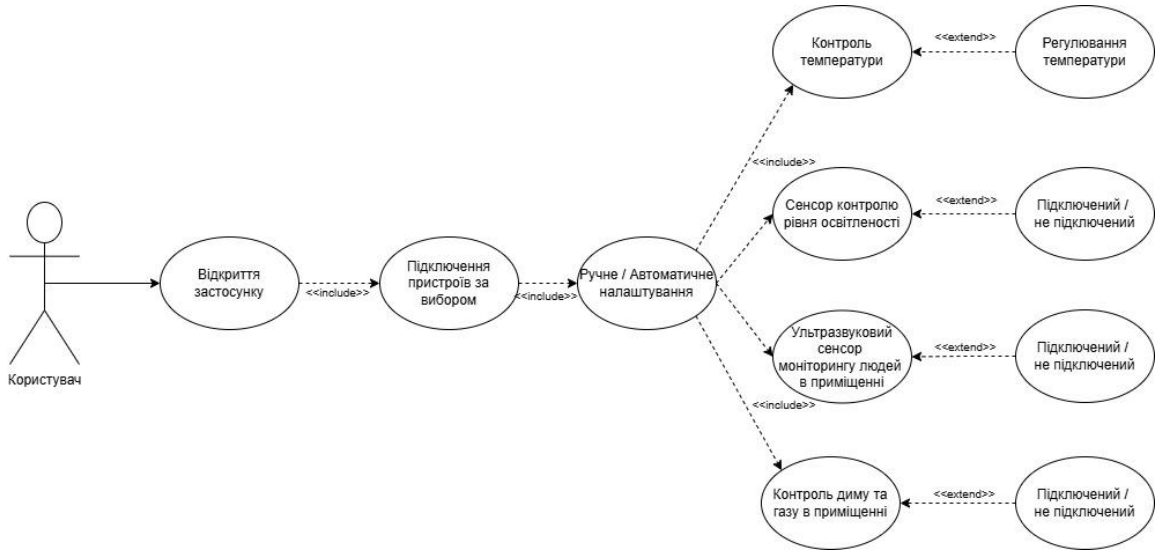

Рисунок 3.18 – Use Case діаграма користувача із застосунком

Основна логіка мобільного застосунку базується на комунікації з апаратною частиною системи через бездротовий інтерфейс Wi-Fi та Bluetooth. Застосунок може надсилати команди до мікроконтролера ESP32 для увімкнення або вимкнення світла, запуску сигналізації або зміни режиму роботи кондиціонера. У зворотному напрямку застосунок приймає дані від сенсорів і відображає їх у зручному графічному вигляді: користувач, значення вимірювання температури, стан підключення датчиків системи, режими налаштування (автоматичне, ручне), контроль сповіщення.

Для розробки застосунку, також було виконано дизайн користувацького інтерфейсу для визначення розмітки та розташування елементів, з якими користувач системи керування «розумним будинком» буде взаємодіяти при використанні (рис. 3.19–3.20).

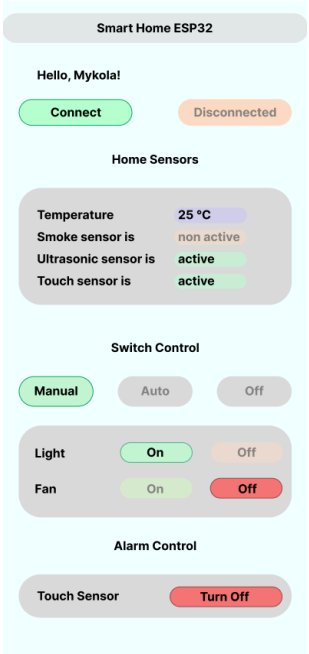

Рисунок 3.19 – Дизайн користувацького інтерфейсу перед підключенням до ESP32

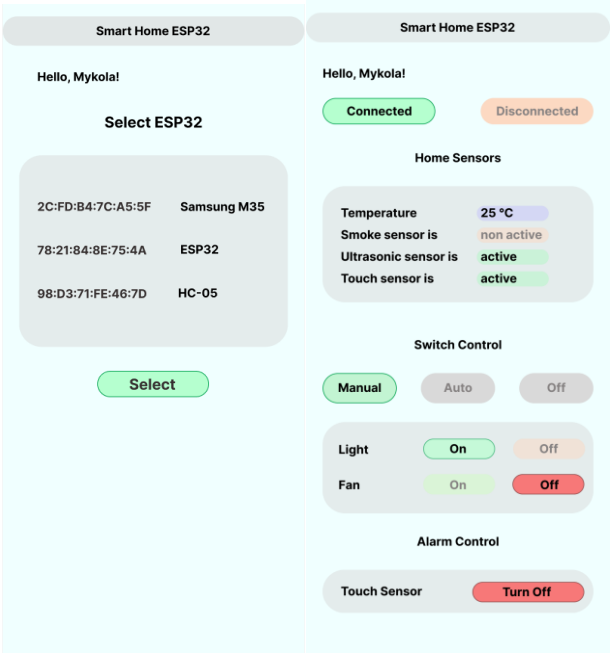

Рисунок 3.20 – Дизайн користувацького інтерфейсу після підключення ESP32

Отже, загалом функціональні можливості застосунку є наступними:

– підключення до контролера ESP32: у верхній частині інтерфейсу користувач бачить повідомлення-привітання та кнопку Connect, що дозволяє встановити зв’язок із контролером ESP32 по Bluetooth. Також, можна відслідкувати статус з’єднання;

– моніторинг сенсорів. У розділі Home Sensors у режимі реального часу виводиться: температура в кімнаті; статус датчика диму (активний / неактивний); статус ультразвукового сенсора (для відстеження руху або наявності об’єктів). Активні елементи підсвічені кольором (наприклад, «active» – зелений, «non active» – червоний), що дозволяє швидко оцінити ситуацію;

– керування режимами перемикання. Дає змогу обрати: ручне керування (вмикання/вимикання пристроїв вручну); автоматичний режим (на основі показників сенсорів або часу); повне вимкнення автоматичного керування;

– керування приладами. Можна незалежно вмикати/вимикати: освітлення та вентилятор;

– керування тривожною сигналізацією. Якщо система зафіксувала дотик (наприклад, вторгнення), користувач може вимкнути тривогу натисканням кнопки Turn Off у розділі Touch Sensor.

Така структура системи надає змогу користувачу дистанційно контролювати основні прилади в системі «розумного будинку».

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи було детально розглянуто розробку апаратної частини системи «розумного будинку», включаючи реалізацію різних пристроїв у Tinkercad, використання їх з ESP32 як центром керування в апаратному комплексі. У даному розділі, основна увага була приділена датчикам різних типів та іншим компонентам, що забезпечують автоматизацію процесів у системі «розумного будинку».

У даній роботі, датчик температури та вологості DHT11 використовується для моніторингу кліматичних умов усередині приміщення. Цей сенсор забезпечує регулярне зчитування значень температури та вологості, що дозволяє підтримувати комфортні умови в будинку та створює основу для автоматичного керування опаленням або кондиціонуванням.

Фотодіод у системі керування «розумним будинком» використовується для автоматичного регулювання освітлення залежно від рівня зовнішньої освітленості. Його робота базується на зміні електричної провідності під впливом світла. Це дозволяє вмикати або вимикати освітлення без участі користувача, знижуючи енергоспоживання та підвищуючи комфорт.

Газовий датчик MQ-2 застосовується для виявлення диму та горючих газів, що є критично важливим для безпеки мешканців. MQ-2 реагує на зміну концентрації газів у повітрі й надсилає сигнал на мікроконтролер, що може активувати сирену або сповістити користувача через застосунок.

Для виявлення присутності людей розглянуто використання двох типів датчиків: ультразвуковий HC-SR04 та інфрачервоний PIR-сенсор. HC-SR04 вимірює відстань до об’єктів, що дозволяє фіксувати наближення до дверей або вікон, тоді як PIR-сенсор виявляє рух у зоні спостереження за допомогою інфрачервоного випромінювання.

Для реалізації роботи системи використано низку бібліотек і фреймворків Arduino, зокрема для роботи із сенсорами, дисплеєм та зв’язком із Android-застосунком.

Варто зауважити, що мобільний застосунок, створений в Android Studio, є центральним елементом взаємодії користувача із системою керування, оскільки дозволяє переглядати стан сенсорів, керувати освітленням, вентиляцією та іншими пристроями в ручному або автоматичному режимі.

Розглянуті компоненти та сенсори формують комплексну систему апаратного забезпечення «розумного будинку», яка забезпечує достатній рівень контролю та зручності для користувачів.

ВИСНОВКИ

У ході виконання кваліфікаційної бакалаврської роботи, було запропоноване та розроблене АПЗ системи керування «розумним будинком», що має низку переваг у порівнянні з існуючими на комерційному ринку системами. У першу чергу, до них належать нижча собівартість реалізації, можливість масштабування та оптимізації системи відповідно до потреб користувача, у тому числі адаптивність системи.

Практична значущість результатів даної роботи полягає у тому, що даний комплекс дозволяє розроблювати економічно вигідні системи керування «розумними будинками» без використання дороговартісних інструментів і програмних рішень.

Відповідно до сформованих завдань, досягнуто основні результати в ході виконання КБР, а саме:

– проведено аналіз існуючих рішень у сфері систем «розумного будинку» та архітектурних особливостей їх побудови;

– досліджено технічні та програмні можливості модуля ESP32;

– розроблено архітектуру системи керування «розумним будинком»;

– реалізовано апаратну частину системи «розумний будинок»;

– реалізовано програмну частину для забезпечення керування пристроями.

У результаті, розроблено цілісну та функціонуючу систему керування «розумним будинком», що сенсори моніторингу навколишнього середовища та мобільний застосунок для взаємодії користувача з пристроями.

У подальшому розвиток предмету дослідження може полягати в розширенні функціональних можливостей системи «розумного будинку» шляхом інтеграції додаткових сенсорів, таких як датчики якості повітря (CO_2, пилу), камери відеоспостереження, сенсори відкриття дверей, вікон і контролери енергоспоживання. Це дозволить створити більш комплексну та адаптивну систему, що забезпечить не лише автоматизацію побутових процесів, але й підвищити рівень енергоефективності та безпеки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Taiwo O., Gabralla L. A., Ezugwu A. E. Smart Home Automation: Taxonomy, Composition, Challenges and Future Direction. *Computational Science and Its Applications – ICCSA 2020: 20th International Conference*. Cagliari, Italy. Jul. 1–4, 2020. P. 878–894. DOI: 10.1007/978-3-030-58817-5_62.

2. Smart Home Protocols: How Your Devices Talk to Each Other. *smartgeekhome.com*. URL: https://smartgeekhome.com/smart-home-protocols/ (Last accessed: 29.04.2025).

3. Introduction to the Internet of Things: Concepts, Technologies, and Applications. *eicta.iitk.ac.in*. URL: https://eicta.iitk.ac.in/knowledge-hub/internet-of-things/introduction-to-the-internet-of-things-concepts-technologies-and-applications/ (Last accessed: 29.04.2025).

4. ESP32. *www.espressif.com*. URL: https://www.espressif.com/en/products/socs/esp32 (Last accessed: 30.04.2025).

5. Smart Home Guide: Get the Most Out of Your Amazon Alexa, Google Home and Apple Homekit. *www.cnet.com*. URL: https://www.cnet.com/home/smart-home/smart-home-guide-get-the-most-out-of-your-amazon-alexa-google-home-and-apple-homekit/ (Last accessed: 30.04.2025).

6. Nest Hub Max. *store.google.com*. URL: https://store.google.com/us/product/google_nest_hub_max?hl=en-US (Last accessed: 01.05.2025).

7. Chen X., Zhang X., Elliot M., Wang X., Wang F. Fix the leaking tap: A survey of Trigger-Action Programming (TAP) security issues, detection techniques and solutions. *Computers & Security*. June, 2022. DOI 10.1016/j.cose.2022.102812.

8. Система Xiaomi Mi Home на практиці: чи можуть безпека і комфорт бути недорогими? URL: https://homehub.com.ua/uk/news/systema-xiaomi-mi-home-na-praktyke-mohut-ly-bezopasnost-y-komfort-bt-nedorohymy/ (дата звернення: 01.05.2025).

9. Danbatta S., Varol A. Comparison of Zigbee, Z-Wave, Wi-Fi, and Bluetooth Wireless Technologies Used in Home Automation. *2019 7th International*

Symposium on Digital Forensics and Security (ISDFS). Barcelos, Portugal, Jun. 10–12 June 2019. DOI: 10.1109/ISDFS.2019.8757472.

10. openHAB vs. Home Assistant. *www.wundertech.net*. URL: https://www.wundertech.net/openhab-vs-home-assistant/ (Last accessed: 01.05.2025).

11. Home Assistance. *www.home-assistant.io*. URL: https://www.home-assistant.io/getting-started/ (Last accessed: 01.05.2025).

12. Introduction OpenHAB. *www.openhab.org*. URL: https://www.openhab.org/docs/ (Last accessed: 01.05.2025).

13. The best smart home systems: Top ecosystems for home automation explained. *the-ambient.com*. URL: https://www.the-ambient.com/explainers/smart-home-ecosystems-152/ (Last accessed: 02.05.2025).

14. FreeRTOS. URL: https://www.osrtos.com/rtos/freertos/ (Last accessed: 04.05.2025).

15. Interfacing AM2301/DHT21 Temperature & Humidity Sensor Module with Arduino. URL: https://electropeak.com/learn/interfacing-am2301-dht21-temperature-humidity-sensor-module-with-arduino/ (Last accessed: 05.05.2025).

16. GY-302 BH1750 light intensity module. URL: https://www.pixelelectric.com/sensors/light-sound/light-color-optical-sensor/gy-302-bh1750-light-intensity-module/ (Last accessed: 06.05.2025).

17. Samuel R., Kumar G., Vignesh V., Vigneshwaran V. Empowering farmers for a prosperous india: iot enabled automatic irrigation and prevention of animal intrusion for better crop yield. *Advances in Mathematics: Scientific Journal*. 2020. DOI: 10.37418/amsj.9.9.14.

18. Qutqut M., Al-Sakran A., Almasalha F., Hassanein H. Comprehensive Survey of the IoT Open Source OSs. *IET Wireless Sensor Systems*. 2018. DOI 10.1049/iet-wss.2018.5033.

19. Stolojescu-Crisan C., Crisan C., Butunoi B.-P. An IoT-Based Smart Home Automation System. *Sensors*. 2021. DOI: 10.3390/s21113784.

20. Mustofa A., Dagnew Y., Prabhakar G., Idrisi Dr. M. SECHA: A Smart Energy-Efficient and Cost-Effective Home Automation System for Developing Countries. *Journal of Computer Networks and Communications*. 2023. DOI: 10.1155/2023/8571506.

ДОДАТОК А

Блок-схема алгоритму роботи з порівнянням показників

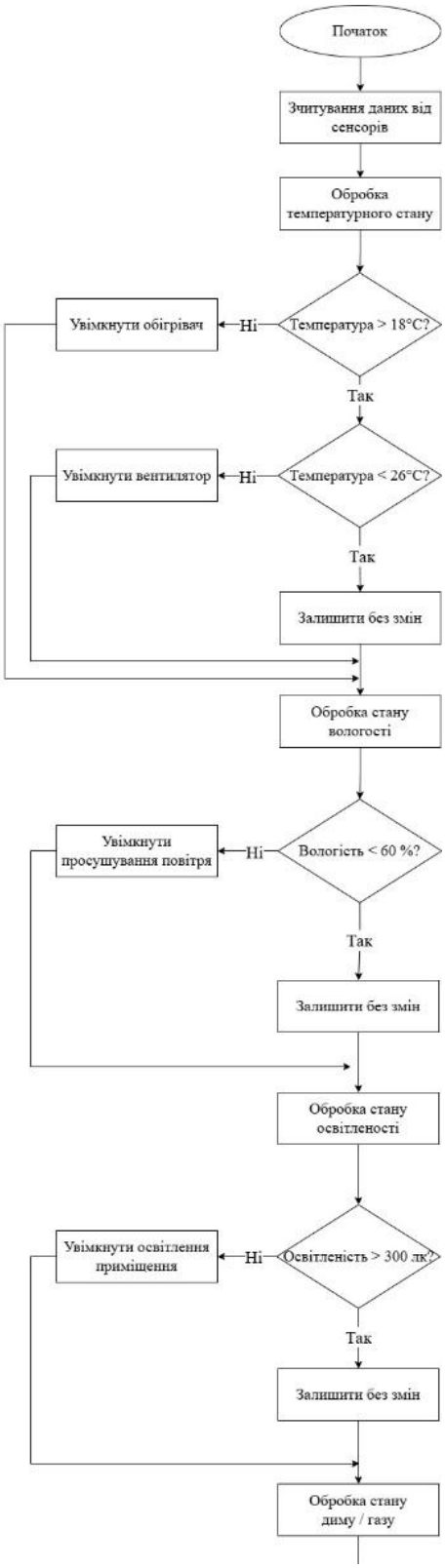

Рисунок А.1 – Алгоритм обробки показників

Рисунок А.2 – Алгоритм обробки показників

ДОДАТОК Б

Код програми

```
#include "BluetoothSerial.h"
#include <Wire.h>
#include "DHT.h"
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
#include <Ticker.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

#if CONFIG_FREERTOS_UNICORE
static const BaseType_t app_cpu = 0;
#else
static const BaseType_t app_cpu = 1;
#endif

#define DHTPIN 33
#define DHTTYPE DHT11
#define lightSensor 26
#define smokeSensor 25
#define touchSensor 4
#define echo 2
#define trigger 15

#define fanRelay 17
#define lightRelay 16

#define smokeBuzzer 14
#define touchBuzzer 27

#define smokeLed 5
#define touchLed 19
#define ultrasonicLed 18

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define SCREEN_ADDRESS 0x3C
#define OLED_RESET 4

DHT dht(DHTPIN, DHTTYPE);
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
BluetoothSerial SerialBT;
Ticker ultrasonic;

static QueueHandle_t tempReading;
static QueueHandle_t lightReading;
static QueueHandle_t smokeAlarm;
static QueueHandle_t touchAlarm;

TaskHandle_t autoFan_handle = NULL;
TaskHandle_t autoLight_handle = NULL;

bool fanStatus = false;
bool lightStatus = false;
bool smokeStatus = false;
bool touchStatus = false;
bool ultrasonicStatus = false;

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);

  Serial.begin(115200);
  Wire.begin();
  SerialBT.begin("ESP32");
```

```
  Serial.println("The device started, now you can pair it with bluetooth!");
  dht.begin();
  ultrasonic.attach(1, ultrasonicDetect);
  display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);

  pinMode(fanRelay, OUTPUT);
  pinMode(lightRelay, OUTPUT);
  pinMode(smokeLed, OUTPUT);
  pinMode(touchLed, OUTPUT);
  pinMode(ultrasonicLed, OUTPUT);
  pinMode(smokeBuzzer, OUTPUT);
  pinMode(touchBuzzer, OUTPUT);
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);

  digitalWrite(fanRelay, HIGH);
  digitalWrite(lightRelay, HIGH);

  digitalWrite(touchBuzzer, LOW);
  digitalWrite(smokeBuzzer, LOW);

  digitalWrite(smokeLed, LOW);
  digitalWrite(touchLed, LOW);
  digitalWrite(ultrasonicLed, LOW);

  introDisplay();

  tempReading = xQueueCreate(10, sizeof(int));
  lightReading = xQueueCreate(10, sizeof(int));

  xTaskCreatePinnedToCore (tempRead, "Temp read", 2048, NULL, 1, NULL, app_cpu);
  xTaskCreatePinnedToCore (autoFan, "Auto fan", 2048, NULL, 1, &autoFan_handle, app_cpu);
  xTaskCreatePinnedToCore (lightRead, "Light read", 1024, NULL, 1, NULL, app_cpu);
  xTaskCreatePinnedToCore (autoLight, "Auto light", 1024, NULL, 1, &autoLight_handle, app_cpu);
  xTaskCreatePinnedToCore (smokeDetect, "Smoke detect", 1024, NULL, 1, NULL, app_cpu);
  xTaskCreatePinnedToCore (touchDetect, "Touch read", 1024, NULL, 1, NULL, app_cpu);
  xTaskCreatePinnedToCore (switchControl, "Switch control", 4096, NULL, 1, NULL, app_cpu);
  xTaskCreatePinnedToCore (indicatorDisplay, "OLED display", 4096*2, NULL, 1, NULL, app_cpu);

  vTaskSuspend (autoFan_handle);
  vTaskSuspend (autoLight_handle);
}

void loop() {
  vTaskDelay(500 / portTICK_PERIOD_MS);
}

void tempRead(void *parameter) {
 int t = 0;

  while (true) {
    t = dht.readTemperature();

    if (isnan(t)) {
      Serial.println(F("Failed to read from DHT sensor!"));
      return;
    }

    SerialBT.print("#");
    SerialBT.print(t);
    SerialBT.print("?");

    Serial.print("Temperature: ");
    Serial.print(t);
    Serial.println(" °C");

    xQueueSend (tempReading, (void*)&t, 10);
    vTaskDelay(2000 / portTICK_PERIOD_MS);
  }
}

void autoFan(void *parameter) {
  int tempValue;

  while (true) {
    xQueueReceive(tempReading, (void *)&tempValue, portMAX_DELAY);
```

```
    if (tempValue >= 33) {
      SerialBT.print ("Fan on?");
      digitalWrite(fanRelay,LOW); ;
      fanStatus = true;
    }
    else if (tempValue < 33) {
      SerialBT.print ("Fan off?");
      digitalWrite(fanRelay,HIGH);
      fanStatus = false;
    }
    vTaskDelay(200 / portTICK_PERIOD_MS);
  }
}

void lightRead(void *parameter) {
  int lightValue;

  while (true) {
    lightValue = analogRead(lightSensor);
    Serial.print("Light intensity: ");
    Serial.println(lightValue);

    xQueueSend (lightReading, (void*)&lightValue, 10);
    vTaskDelay(2000 / portTICK_PERIOD_MS);
  }
}

void autoLight(void *parameter) {
  int lightValue;

  while (true) {
    xQueueReceive(lightReading, (void *)&lightValue, portMAX_DELAY);

    if (lightValue >= 2200) {
      SerialBT.print("Bulb on?");
      digitalWrite(lightRelay,LOW);
      lightStatus = true;
    }
    else if (lightValue < 2200) {
      SerialBT.print("Bulb off?");
      digitalWrite(lightRelay,HIGH);
      lightStatus = false;
    }
    vTaskDelay(200 / portTICK_PERIOD_MS);
  }
}

void smokeDetect(void *parameter) {
  int smokeValue;

  while (true) {
    smokeValue = analogRead(smokeSensor);
    Serial.print("Smoke: ");
    Serial.println(smokeValue);

    if (smokeValue >= 3200) {
      SerialBT.print("Smoke active?");
      digitalWrite(smokeLed, HIGH);
      digitalWrite(smokeBuzzer, HIGH);
      smokeStatus = true;
    }
    else if (smokeValue < 3200) {
      SerialBT.print("Smoke inactive?");
      digitalWrite(smokeLed, LOW);
      digitalWrite(smokeBuzzer, LOW);
      smokeStatus = false;
    }

    vTaskDelay(1000 / portTICK_PERIOD_MS);
  }
}

void touchDetect(void *parameter) {
  int touchValue;

  while (true) {
    touchValue = (touchRead(touchSensor));
```

```
    if (touchValue < 20) {
      SerialBT.print("Touch active?");
      digitalWrite(touchLed, HIGH);
      digitalWrite(touchBuzzer, HIGH);
      touchStatus = true;
    }
  }
}

void ultrasonicDetect() {
  int distance;
  int duration;

  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);

  duration = pulseIn(echo, HIGH);
  distance = (duration / 2) * 0.0343;

  Serial.print("Distance: ");
  Serial.println(distance);

  if (distance > 20) {
    SerialBT.print("Ultrasonic inactive?");
    digitalWrite(ultrasonicLed, LOW);
    ultrasonicStatus = false;
  }
  else if (distance <= 20) {
    SerialBT.print("Ultrasonic active?");
    digitalWrite(ultrasonicLed, HIGH);
    ultrasonicStatus = true;
  }
}

void switchControl(void *parameter) {
  char input;

  while (true) {
    if (SerialBT.available() > 0) {
      input = SerialBT.read();
      switch (input) {
        case 'M': {
          vTaskSuspend (autoFan_handle);
          vTaskSuspend (autoLight_handle);
          break;
        }
        case 'F': {
          SerialBT.print("Fan on?");
          digitalWrite(fanRelay, LOW);
          fanStatus = true;
          break;
        }
        case 'Y': {
          SerialBT.print("Fan off?");
          digitalWrite(fanRelay, HIGH);
          fanStatus = false;
          break;
        }
        case 'L': {
          SerialBT.print("Bulb on?");
          digitalWrite(lightRelay, LOW);
          lightStatus = true;
          break;
        }
        case 'Z': {
          SerialBT.print("Bulb off?");
          digitalWrite(lightRelay, HIGH);
          lightStatus = false;
          break;
        }
        case 'A': {
          vTaskResume(autoFan_handle);
          vTaskResume(autoLight_handle);
          break;
```

```
        }
        case 'O': {
          vTaskSuspend(autoFan_handle);
          vTaskSuspend(autoLight_handle);
          SerialBT.print("Fan off?");
          SerialBT.print("Bulb off?");
          digitalWrite(fanRelay, HIGH);
          digitalWrite(lightRelay, HIGH);
          fanStatus = false;
          lightStatus = false;
          break;
        }
        case 'T': {
          SerialBT.print("Touch inactive?");
          digitalWrite(touchBuzzer, LOW);
          digitalWrite(touchLed, LOW);
          touchStatus = false;
          break;
        }
      }
    }
  }
}

void indicatorDisplay(void *parameter) {
  int tempValue;

  const unsigned char fanIndicator [] PROGMEM = {
    0x01, 0xf8, 0x00, 0x07, 0x0e, 0x00, 0x0d, 0xc3, 0x00, 0x1b, 0xe1, 0x80, 0x11, 0xe0, 0x80, 0x30,
          0xe0, 0xc0, 0x30, 0x66, 0xc0, 0x30, 0x7f, 0xc0, 0x33, 0xff, 0xc0, 0x33, 0xde, 0xc0, 0x17, 0xc0,
          0x80, 0x1b, 0x81, 0x80, 0x0f, 0x83, 0x00, 0x06, 0x06, 0x00, 0x03, 0xfc, 0x00, 0x00, 0x00, 0x00,
          0x00, 0xf0, 0x00, 0x01, 0xf8, 0x00, 0x07, 0xfe, 0x00, 0x07, 0xfe, 0x00
  };

  const unsigned char lightIndicator [] PROGMEM = {
    0x00, 0x60, 0x00, 0x01, 0xfc, 0x00, 0x07, 0xc6, 0x00, 0x07, 0xc2, 0x00, 0x0f, 0xf1, 0x00, 0x0f,
        0xf1, 0x00, 0x0f, 0xfb, 0x00, 0x0f, 0xff, 0x00, 0x0f, 0xff, 0x00, 0x07, 0xfe, 0x00, 0x07, 0xfe,
        0x00, 0x03, 0xfc, 0x00, 0x01, 0xfc, 0x00, 0x01, 0xf8, 0x00, 0x01, 0x00, 0x00, 0x01, 0x80, 0x00,
        0x01, 0xf8, 0x00, 0x01, 0xf8, 0x00, 0x00, 0xf0, 0x00, 0x00, 0x00, 0x00
  };

  const unsigned char smokeIndicator [] PROGMEM = {
    0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x30, 0x00, 0x00, 0x38, 0x00, 0x00, 0x38, 0x00, 0x02,
          0x78, 0x00, 0x07, 0xf8, 0x00, 0x07, 0xfa, 0x00, 0x07, 0x3e, 0x00, 0x07, 0x3f, 0x00, 0x03, 0x0f,
          0x00, 0x17, 0x0f, 0x80, 0x1e, 0x0f, 0x80, 0x1e, 0x07, 0x80, 0x1e, 0x07, 0x80, 0x1e, 0x07, 0x80,
          0x0e, 0x07, 0x80, 0x0f, 0x0f, 0x00, 0x03, 0xfc, 0x00, 0x00, 0x60, 0x00
  };

  const unsigned char touchIndicator [] PROGMEM = {
    0x00, 0x60, 0x00, 0x01, 0xf8, 0x00, 0x03, 0xfc, 0x00, 0x07, 0x8e, 0x00, 0x07, 0x0e, 0x00, 0x06,
          0x06, 0x00, 0x06, 0x06, 0x00, 0x00, 0x06, 0x00, 0x00, 0x06, 0x00, 0x00, 0x0f, 0x00, 0x1f, 0xff,
          0x80, 0x1f, 0xff, 0x80, 0x1f, 0xff, 0x80, 0x1f, 0xff, 0x80, 0x1f, 0xff, 0x80, 0x1f, 0xff, 0x80,
          0x1f, 0xff, 0x80, 0x1f, 0xff, 0x80, 0x00, 0x00, 0x00, 0x1f, 0xff, 0x80
  };

  const unsigned char ultrasonicIndicator [] PROGMEM = {
    0x00, 0x00, 0x00, 0x00, 0xf0, 0x00, 0x01, 0xf8, 0x00, 0x03, 0xfc, 0x00, 0x03, 0xfc, 0x00, 0x03,
        0xfc, 0x00, 0x03, 0xfc, 0x00, 0x03, 0xfc, 0x00, 0x03, 0xfc, 0x00, 0x01, 0xf8, 0x00, 0x01, 0xf8,
        0x00, 0x01, 0xf0, 0x00, 0x00, 0xf0, 0x00, 0x00, 0xf0, 0x00, 0x07, 0xfe, 0x00, 0x1f, 0xff, 0x80,
          0x3f, 0xff, 0xc0, 0x7f, 0xff, 0xe0, 0xff, 0xff, 0xf0, 0xff, 0xff, 0xf0
  };

  while (true) {
    xQueueReceive(tempReading, (void*)&tempValue, portMAX_DELAY);

    display.clearDisplay();

    display.setTextColor(WHITE);
    display.setTextSize(2);
    display.setCursor(10,10);
    display.print("Temp ");
    display.print(tempValue);
    display.print((char)247);
    display.print("C");

    if (fanStatus == true) {
      display.drawBitmap(2, 36, fanIndicator, 20, 20, WHITE);
```

```
    }
    else if (fanStatus == false) {
      display.setCursor(6, 38);
      display.print("-");
    }

    if (lightStatus == true) {
      display.drawBitmap(28, 36, lightIndicator, 20, 20, WHITE);
    }
    else if (lightStatus == false) {
      display.setCursor(32, 38);
      display.print("-");
    }

    if (smokeStatus == true) {
      display.drawBitmap(54, 36, smokeIndicator, 20, 20, WHITE);
    }
    else if (smokeStatus == false) {
      display.setCursor(58, 38);
      display.print("-");
    }

    if (touchStatus == true) {
      display.drawBitmap(80, 36, touchIndicator, 20, 20, WHITE);
    }
    else if (touchStatus == false) {
      display.setCursor(84, 38);
      display.print("-");
    }

    if (ultrasonicStatus == true) {
      display.drawBitmap(106, 36, ultrasonicIndicator, 20, 20, WHITE);
    }
    else if (ultrasonicStatus == false) {
      display.setCursor(110, 38);
      display.print("-");
    }

    display.display();
    vTaskDelay(200 / portTICK_PERIOD_MS);
  }
}

void introDisplay() {
  display.clearDisplay();

  const unsigned char intro [] PROGMEM = {
  };

  display.drawBitmap(0, 0, intro, 128, 64, WHITE);

  display.display();
  delay(4000);
  display.clearDisplay();

  display.setTextColor(WHITE);
  display.setTextSize(2);
  display.setCursor(6,10);
  display.print("Smart Home");
  display.setCursor(6,40);
  display.print("Automation");

  display.display();
  delay(4000);
  display.clearDisplay();
}
```