Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
Факультет Комп'ютерної інженерії та управління
(повна назва)
Кафедра Автоматизації проєктування обчислювальної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Агентно-орієнтована система керування "розумним будинком" з використанням IoT-пристроїв
(тема)

Виконав: здобувач 4 курсу,
групи КІУКІ-21-7
Дзюба Д.В
(прізвище, ініціали)

спеціальності 123 – Комп'ютерна інженерія
(шифр і назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник ас. Хаханов І.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С.В.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проєктування обчислювальної техніки

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 Комп'ютерна інженерія

Тип програми Освітньо-професійна

Освітня програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ:
Зав. кафедри
Чумаченко С.В

_____
(підпис)
06.05. 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувачеві Дзюбі Дмитру Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Агентно-орієнтована система керування "розумним будинком" з використанням IoT-пристроїв

затверджена наказом університету від 21 05 2025 р. № 403Ст

2. Термін подання студентом роботи до екзаменаційної комісії 21. 06. 2025 р.

3. Вихідні дані до роботи
Мікроконтролер ESP32.
Протокол обміну MQTT

4. Перелік питань, що потрібно опрацювати в роботі
Аналіз систем «розумного будинку» та технологій IoT
Вибір апаратних і програмних компонентів системи
Розробка агентно-орієнтованої архітектури на базі ESP32
Реалізація програмної логіки в середовищі Node-RED

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) 11 слайдів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Видача теми проєкту, узгодження і затвердження теми	21.04.2025 – 22.04.2025	
2	Аналіз предметної області, вибір компонентів системи	23.04.2025 – 04.05.2025	
3	Розробка структурної схеми пристрою, вибір апаратної платформи	05.05.2025 – 10.05.2025	
4	Розробка функціональної схеми програми	10.05.2025 – 20.05.2025	
5	Розробка програмних модулів. Проведення тестування	21.05.2025 – 26.05.2025	
6	Оформлення пояснювальної записки	27.05.2025 – 20.06.2025	
7	Підготовка ілюстративних матеріалів.	21.06.2025 – 22.06.2025	

Дата видачі завдання 22.04.2025 р.

Здобувач _____ Дзюба Д.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ ас. Хаханов І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Записка пояснювальна: 38 сторінок, 7 рисунків, 9 джерел за переліком посилань.

РОЗУМНИЙ БУДИНОК, ІНТЕРНЕТ РЕЧЕЙ, ESP32, MQTT, NODE-RED, АГЕНТИ, TELEGRAM-БОТ

Метою кваліфікаційної роботи є розробка прототипу агентно-орієнтованої системи керування «розумним будинком» на базі мікроконтролера ESP32 з використанням технологій Інтернету речей. Реалізована система включає сенсори температури, руху, а також виконавчі пристрої (реле), які взаємодіють між собою за допомогою MQTT-протоколу та програмних агентів, реалізованих у середовищі Node-RED.

Усі дані передаються бездротовою мережею Wi-Fi, що забезпечує автономну роботу та масштабованість системи. Для взаємодії з користувачем реалізовано Telegram-бот та мобільний застосунок, що дозволяє здійснювати моніторинг і керування пристроями в режимі реального часу.

ABSTRACT

Explanatory note: 38 pages, 7 figures, 9 sources in the list of references.

SMART HOME, INTERNET OF THINGS, ESP32, MQTT, NODE-RED, AGENTS, TELEGRAM BOT

The aim of this qualification work is the development of a prototype of an agent-based smart home control system based on the ESP32 microcontroller using Internet of Things technologies. The implemented system includes temperature and motion sensors as well as actuators (relays), which interact with each other via the MQTT protocol and software agents built in the Node-RED environment.

All data is transmitted wirelessly over Wi-Fi, which ensures autonomous operation and scalability of the system. A Telegram bot and a mobile application have been implemented to provide user interaction, allowing real-time device monitoring and control.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 7
ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 9
1.1 Поняття та принципи функціонування «розумного будинку» 9
1.2 Агентно-орієнтовані системи керування: теорія та застосування 11
1.3 Сучасні технології IoT в автоматизації побуту 13
1.4 Формулювання технічного завдання 15
2 ВИБІР АРХІТЕКТУРИ ТА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ 17
2.1 Архітектура агентно-орієнтованої системи 17
2.2 Вибір мікроконтролера та IoT-компонентів 18
2.3 Протоколи зв'язку та обмін повідомленнями між агентами 21
2.4 Побудова логіки взаємодії компонентів системи 22
3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ 25
3.1 Розробка програмного коду для ESP та сенсорів 25
3.2 Реалізація агентів у середовищі Node-RED / MQTT-брокера 27
3.3 Інтерфейс користувача: мобільний застосунок або веб-панель 29
3.4 Обробка подій і сценарії автоматизації 32
4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ 34
4.1 Макетування системи та підключення компонентів 34
4.2 Проведення демонстраційного експерименту 35
4.3 Аналіз результатів, перевірка сценаріїв керування 36
ВИСНОВКИ 38
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 39
ДОДАТОК А Графічна частина проекту 40
ДОДАТОК Б Текст програми пристрою керування 46

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, 1СКОРОЧЕНЬ І ТЕРМІНІВ

ESP32 — мікроконтролер із вбудованим модулем Wi-Fi та Bluetooth, що використовується для побудови IoT-систем.

GPIO — General Purpose Input/Output, універсальні порти введення/виведення на мікроконтролері.

HTTP — Hypertext Transfer Protocol, протокол передавання гіпертексту.

IoT — Internet of Things (Інтернет речей), концепція з'єднання фізичних пристроїв з мережею для обміну даними.

JSON — JavaScript Object Notation, формат для структурування та обміну даними.

LED — Light Emitting Diode, світлодіод.

MQTT — Message Queuing Telemetry Transport, легковаговий мережевий протокол обміну повідомленнями.

Node-RED — програмне середовище для візуального створення потоків даних і логіки в IoT-системах.

PWM — Pulse Width Modulation, широтно-імпульсна модуляція.

Wi-Fi — Wireless Fidelity, бездротова технологія передавання даних.

ВСТУП

У сучасному світі зростає попит на автоматизацію повсякденних процесів, що безпосередньо стосуються життєдіяльності людини. Одним з ключових напрямків цієї автоматизації є концепція "розумного будинку", яка передбачає інтеграцію електронних пристроїв у єдину керовану систему для підвищення комфорту, енергоефективності та безпеки.

Сьогодні з розвитком технологій Інтернету речей (Internet of Things, IoT) з’явилася можливість створювати гнучкі системи управління, в яких пристрої можуть самостійно приймати рішення на основі зібраних даних. Особливу роль у цьому відіграють агентно-орієнтовані підходи, які дозволяють моделювати кожен пристрій або логічний блок у вигляді автономного "агента", здатного взаємодіяти з іншими учасниками системи.

Актуальність теми зумовлена потребою у побудові адаптивних, масштабованих і зручних у використанні систем керування побутовими пристроями. На відміну від традиційних централізованих систем, агентно-орієнтовані рішення дозволяють підвищити гнучкість, спростити інтеграцію нових компонентів та забезпечити більшу стійкість до відмов.

Метою цієї кваліфікаційної роботи є розробка прототипу агентно-орієнтованої системи керування “розумним будинком” із використанням IoT-пристроїв, яка забезпечуватиме збирання інформації з датчиків, обробку подій, взаємодію між компонентами та можливість дистанційного керування через мобільний інтерфейс або месенджер.

Об'єктом дослідження є інформаційно-керуюча система «розумного будинку».

Предметом дослідження – агентно-орієнтована модель керування з використанням IoT-пристроїв на базі мікроконтролера ESP та бездротового зв’язку.

Практична значущість роботи полягає у можливості створення недорогої та адаптивної системи автоматизації побутових задач з відкритою архітектурою, що може бути використана як у приватному житлі, так і в невеликих комерційних об’єктах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття та принципи функціонування «розумного будинку»

Система «розумного будинку» є сучасною формою автоматизації житлового середовища, що поєднує апаратні та програмні засоби для управління побутовими процесами. Основна ідея полягає у створенні комфортного, енергоефективного та безпечного середовища за допомогою інтегрованих технологій. Така система забезпечує централізоване або розподілене керування освітленням, температурою, вентиляцією, безпекою, електроживленням та іншими життєво важливими компонентами будинку.

З технічної точки зору, розумний будинок базується на використанні мікроконтролерів, сенсорів, актуаторів та каналів зв’язку, які дозволяють зчитувати інформацію з навколишнього середовища та впливати на нього. Всі пристрої об’єднані в єдину систему, що дозволяє реалізовувати задані сценарії автоматизації: наприклад, автоматичне вмикання світла при фіксації руху або відкривання вікна, включення вентиляції при перевищенні температури, або надсилання повідомлення користувачу у випадку виявлення диму.

Ключовими характеристиками таких систем є автоматичність, гнучкість, інтерактивність, дистанційний доступ і масштабованість. Система може працювати автономно, приймаючи рішення на основі даних із датчиків, або дозволяти керування вручну за допомогою мобільного застосунку чи веб-інтерфейсу. Передавання даних і команд може здійснюватися як через локальні протоколи (наприклад, ZigBee, Bluetooth), так і через Інтернет — з використанням Wi-Fi або мобільних мереж.

Інтеграція технологій Інтернету речей (IoT) значно розширює можливості таких систем, забезпечуючи постійне підключення до мережі, збирання статистичних даних, віддалене оновлення програмного забезпечення та гнучке налаштування поведінки системи. Завдяки відкритим апаратним платформам, зокрема ESP8266 та ESP32, стало можливим створення доступних за вартістю прототипів з використанням вбудованих засобів

бездротового зв’язку, цифрових і аналогових входів, а також підтримкою сучасних протоколів обміну даними, таких як MQTT, HTTP, WebSocket та інші.

Особливу актуальність набуває застосування агентно-орієнтованої моделі побудови системи. У цій моделі кожен пристрій або логічний компонент розглядається як окремий агент — програмна сутність, що має власні цілі, поведінку та можливість обміну повідомленнями з іншими агентами. Такий підхід дозволяє реалізовувати більш гнучку архітектуру, в якій система не залежить від єдиного центрального контролера, що підвищує її надійність та спрощує масштабування.

Завдяки використанню агентів, можна легко реалізовувати складні сценарії взаємодії між підсистемами. Наприклад, агент безпеки може взаємодіяти з агентом освітлення при виявленні руху вночі, а кліматичний агент — вмикати вентиляцію на основі показників температури та вологості. Така модульність також полегшує супровід системи, зміну логіки поведінки та впровадження нових функцій без необхідності змінювати всю архітектуру.

У результаті, система «розумного будинку» на основі IoT-пристроїв та агентного підходу відкриває широкі перспективи для підвищення рівня автоматизації, адаптації до індивідуальних потреб користувача та інтеграції з глобальними інформаційними системами.

1.2 Агентно-орієнтовані системи керування: теорія та застосування

Одним із сучасних підходів до побудови складних програмно-керованих систем, зокрема систем автоматизації, є агентно-орієнтоване моделювання. Цей підхід ґрунтується на ідеї представлення окремих елементів системи у вигляді агентів — автономних одиниць, здатних сприймати інформацію, приймати рішення та взаємодіяти між собою. Агент у даному контексті визначається як програмна або апаратно-програмна сутність, що має власну поведінку, може самостійно реагувати на події та змінювати стан системи в межах своєї компетенції.

У порівнянні з класичними централізованими архітектурами, агентно-орієнтовані системи мають ряд переваг. Насамперед це підвищена гнучкість, що дозволяє легко змінювати або розширювати функціональність системи без потреби втручання в її загальну структуру. Крім того, децентралізація логіки управління сприяє зниженню навантаження на центральний контролер або сервер, а також підвищує надійність системи — у разі відмови одного з агентів інші продовжують виконувати свої функції.

У системі «розумного будинку» агентами можуть виступати як фізичні пристрої (мікроконтролери, сенсори, реле, актуатори), так і програмні модулі, які реалізують логіку взаємодії, обробку даних або комунікацію з користувачем. Наприклад, агент освітлення може приймати рішення про увімкнення або вимкнення світла на основі даних від агента сенсора руху, а агент безпеки — надсилати повідомлення у випадку виявлення незвичних дій. Взаємодія між агентами може здійснюватися як безпосередньо, так і через посередників — брокери повідомлень або інші сервери обміну.

Агентно-орієнтована модель забезпечує високий рівень абстракції, що дозволяє описувати поведінку кожного елементу системи незалежно, а також чітко формалізувати механізми обміну інформацією. На практиці це дає змогу легко додавати нові сценарії, розширювати функціональність, адаптувати поведінку системи до змін у середовищі або потребах користувача. Кожен агент діє згідно з власною метою або набором правил, однак в межах спільної системної архітектури.

Застосування агентно-орієнтованого підходу особливо доцільне в умовах, коли система є динамічною, неоднорідною, має змінний склад або піддається впливу з боку користувача. У таких випадках гнучкість та автономність агентів дозволяють забезпечити адаптивність і стійкість до змін. Крім того, завдяки використанню відкритих стандартів і протоколів (наприклад, MQTT), можлива інтеграція різнорідних компонентів без необхідності глибокої апаратної сумісності.

У контексті реалізації системи розумного будинку це означає, що кожен модуль або пристрій — незалежний учасник, який може працювати автономно, але водночас координується з іншими елементами через загальні канали комунікації. Наприклад, сенсор температури може бути агентом, який надсилає дані до агента кліматичного контролю, що у свою чергу приймає рішення про включення вентиляції або опалення. Цей підхід дозволяє організувати складну поведінку системи у вигляді множини простих взаємодіючих об'єктів, кожен з яких виконує окрему функцію, але разом вони формують єдину узгоджену систему керування.

Загалом, агентно-орієнтовані системи є потужним інструментом для побудови сучасних гнучких, адаптивних та ефективних систем керування в умовах невизначеності, мінливості середовища та потреб інтеграції великої кількості взаємодіючих пристроїв. Їх використання у сфері автоматизації «розумного будинку» відкриває нові можливості для побудови масштабованих, надійних та інтелектуальних рішень.

1.3 Термопари у побутових приладах нагрівання

Сучасні технології Інтернету речей (Internet of Things, IoT) відкривають широкі можливості для автоматизації побуту та створення інтелектуальних середовищ, які можуть взаємодіяти з користувачем, реагувати на зміни зовнішніх умов та приймати самостійні рішення. У контексті «розумного будинку» IoT означає використання фізичних пристроїв, що підключені до мережі Інтернет або локальної бездротової мережі та здатні збирати, передавати, обробляти дані і виконувати команди.

Типова структура IoT-системи включає сенсори (датчики температури, вологості, руху, освітленості тощо), актуатори (реле, серводвигуни, електроклапани), мікроконтролери (наприклад, ESP8266, ESP32), а також шлюзи або брокери повідомлень (наприклад, MQTT). Ключовим аспектом є можливість інтеграції пристроїв між собою та їх з’єднання з хмарними або локальними сервісами, через які здійснюється моніторинг, аналітика та управління.

Однією з основних переваг використання IoT у побутовій автоматизації є підвищення ефективності. Наприклад, система опалення може самостійно регулювати температуру в приміщеннях залежно від присутності людей або часу доби. Освітлення може вмикатись автоматично при виявленні руху або зміні рівня освітленості. Полив саду може запускатись в залежності від вологості ґрунту або прогнозу погоди.

Особливо зручною є можливість взаємодії з системою через мобільні додатки або голосові асистенти. Користувач має змогу переглядати дані з датчиків, змінювати налаштування та керувати пристроями з будь-якої точки світу. Це дозволяє значно підвищити зручність та контроль над домашнім середовищем.

Інтеграція IoT-пристроїв у «розумний будинок» також дає змогу впроваджувати розширену аналітику та автоматичне прийняття рішень на основі накопичених даних. Наприклад, аналіз споживання електроенергії

може виявити надмірне навантаження або неефективні прилади, а система безпеки може виявляти аномальну активність у домі та надсилати попередження.

У більшості випадків зв’язок між IoT-пристроями забезпечується через Wi-Fi, однак також активно використовуються інші протоколи: Bluetooth Low Energy (BLE), ZigBee, Z-Wave або LoRa. Протокол MQTT, що працює за моделлю «видавець–підписник», став одним із найпоширеніших завдяки своїй легкості, низькому споживанню ресурсів та високій надійності при передачі повідомлень у реальному часі.

У результаті, технології IoT відіграють фундаментальну роль у побудові гнучких, адаптивних і доступних за ціною систем керування для розумного будинку. Поєднання IoT-платформ, відкритого програмного забезпечення, мобільних інтерфейсів та хмарних сервісів дає змогу створювати персоналізовані рішення для автоматизації життєвого простору.

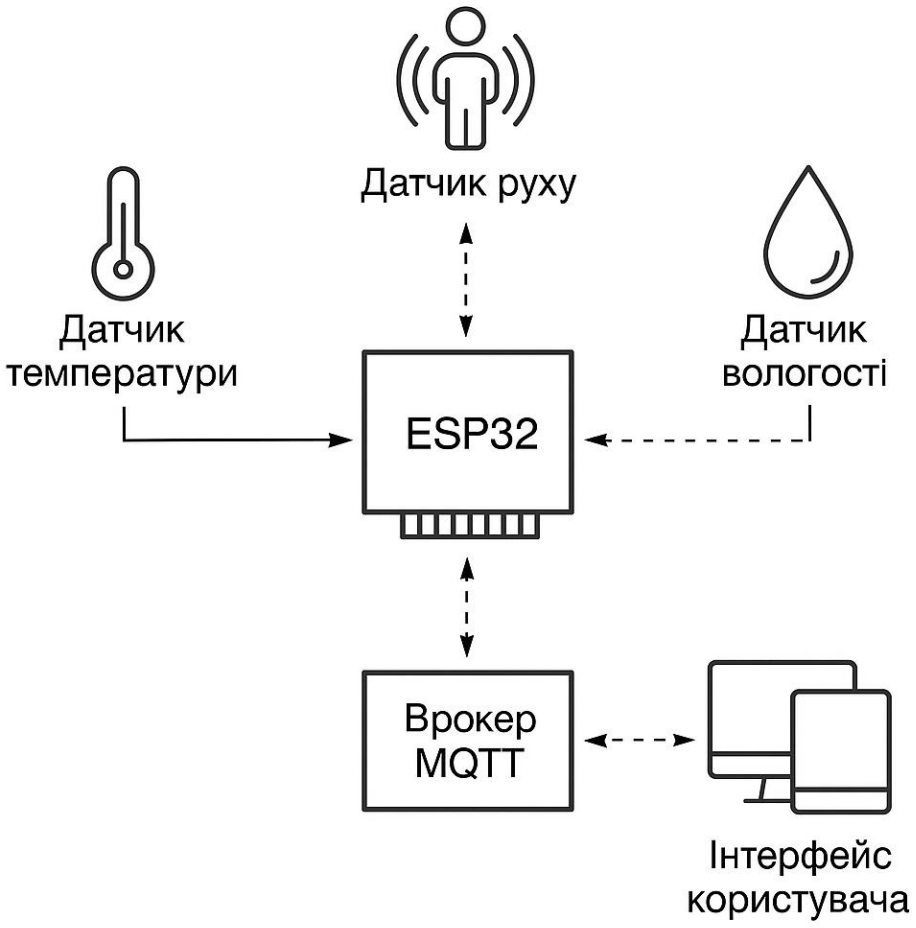

Рисунок 1.1 – Загальна схема IoT-системи у “розумному будинку”

1.4 Формулювання технічного завдання

У рамках даної кваліфікаційної роботи передбачається розробка агентно-орієнтованої системи керування «розумним будинком», яка використовує IoT-пристрої для моніторингу параметрів середовища та автоматизованого керування побутовими приладами. Для ефективної реалізації проєкту необхідно чітко визначити склад і функції системи, вимоги до апаратного та програмного забезпечення, а також основні сценарії її використання.

Система повинна забезпечувати збір даних з навколишнього середовища (температура, вологість, наявність руху) за допомогою відповідних сенсорів, передавання цих даних через бездротову мережу до центрального контролера або посередника (брокера повідомлень), аналіз отриманої інформації, а також виконання відповідних дій згідно із заданою логікою — наприклад, увімкнення освітлення або вентиляції. Усі елементи системи мають бути реалізовані у вигляді агентів, що працюють автономно, але взаємодіють між собою в межах загальної архітектури.

Загальна структура системи включає такі компоненти:

– мікроконтролер ESP32 або ESP8266, який виконує роль центрального вузла керування;

– сенсори температури (DHT22), руху (HC-SR501), вологості;

– виконавчі пристрої — реле, що підключаються до освітлення або вентиляційного обладнання;

– MQTT-брокер для організації взаємодії між агентами;

– середовище Node-RED для реалізації логіки поведінки агентів у графічному форматі;

– мобільний інтерфейс або чат-бот для користувацького доступу до системи.

Основні функціональні вимоги до системи:

– автономна робота кожного агента з можливістю обміну повідомленнями;

– моніторинг показників середовища з фіксованим інтервалом опитування;

– обробка подій та виконання відповідних дій (увімкнення/вимкнення

пристроїв);

– можливість керування з боку користувача (ручне керування, сповіщення, зміна параметрів);

– мінімальне енергоспоживання та висока стабільність роботи пристроїв;

– простота розгортання та можливість масштабування системи за рахунок додавання нових агентів.

Також передбачається реалізація кількох типових сценаріїв:

– автоматичне керування освітленням на основі даних про рух у приміщенні;

– підтримка комфортного мікроклімату шляхом керування вентиляцією;

– надсилання повідомлень користувачу про зміну стану в системі або тривожні події.

Система повинна бути побудована з використанням доступних компонентів, відкритих протоколів та інструментів, що забезпечить її подальший розвиток, налаштування та інтеграцію з іншими пристроями чи сервісами у межах сучасної інфраструктури розумного дому.

2 ВИБІР АРХІТЕКТУРИ ТА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВХ

2.1 Плата розробки на базі мікроконтролера ESP8266

Для побудови ефективної та гнучкої системи керування «розумним будинком» необхідно обґрунтовано підійти до вибору архітектури, яка дозволить реалізувати всі поставлені функціональні вимоги. У межах даної роботи було обрано агентно-орієнтовану архітектуру з розподіленою логікою управління, що забезпечує незалежність компонентів системи, гнучкість налаштувань та високу адаптивність.

У центрі архітектури розташовано вузол на базі мікроконтролера ESP32, який підключає сенсори навколишнього середовища та виконавчі пристрої. Цей вузол виступає не лише в ролі приймача даних, але і як агент, що самостійно приймає рішення на основі отриманої інформації або запитів від інших агентів. Комунікація між компонентами реалізується за допомогою MQTT-протоколу, який дозволяє організувати обмін повідомленнями за моделлю видавець–підписник. Такий підхід виключає необхідність прямого зв’язку між агентами й спрощує логіку їх взаємодії.

Система включає як фізичні агенти (пристрої, що працюють на рівні апаратного забезпечення), так і програмні, які реалізуються, наприклад, у середовищі Node-RED. У цьому середовищі створюються логічні потоки, які дозволяють обробляти події, надсилати повідомлення, активувати виконавчі пристрої або генерувати сповіщення для користувача. Кожен потік у Node-RED відповідає певному агенту або сценарію поведінки системи.

Особливу увагу при проєктуванні архітектури приділено модульності. Кожен агент є незалежним елементом, що полегшує розширення системи — для додавання нового пристрою або логіки достатньо зареєструвати нового агента та забезпечити обмін повідомленнями через брокер. Крім того, архітектура враховує можливість збоїв окремих елементів — система не

втрачає функціональність, якщо виходить з ладу один із агентів.

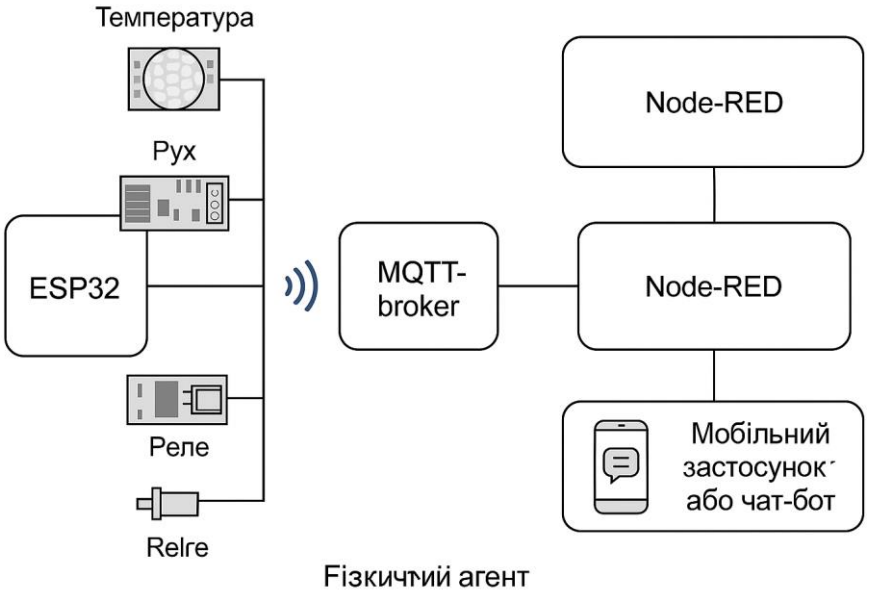

Рисунок 2.1 – Архітектура агентно-орієнтованої системи керування «розумним будинком»

2.2 Вибір мікроконтролера та IoT-компонентів

Одним з ключових етапів при розробці системи керування розумним будинком є вибір апаратного забезпечення, що забезпечить збирання даних із навколишнього середовища, обробку інформації та взаємодію з виконавчими пристроями. У цьому контексті необхідно враховувати такі фактори, як енергоефективність, стабільність, сумісність з сучасними протоколами зв’язку, вартість компонентів і можливість масштабування системи.

Для реалізації запропонованої системи було обрано мікроконтролер ESP32. Це сучасна 32-бітна мікросхема, яка поєднує високу продуктивність, мале енергоспоживання та широкий набір функціональних можливостей. ESP32 має два ядра, працює на частоті до 240 МГц, підтримує Wi-Fi і Bluetooth, а також містить великий набір цифрових та аналогових портів введення-виведення, що дозволяє підключати широкий спектр датчиків та виконавчих пристроїв. Особливо цінною є вбудована підтримка бездротової передачі даних, що виключає необхідність у додаткових модулях.

Серед альтернатив варто згадати ESP8266, Arduino UNO, Raspberry Pi

Zero, однак у порівнянні з ними ESP32 має більше переваг саме для побудови розподілених систем автоматизації. Arduino UNO не має вбудованого Wi-Fi і має обмежену пам’ять та тактову частоту. ESP8266 також підтримує Wi-Fi, але працює на одному ядрі та має менший об’єм оперативної пам’яті, що ускладнює роботу зі складними сценаріями. Raspberry Pi Zero є повноцінним міні-комп’ютером, однак потребує значно більше енергії, часу на завантаження, та складніший у підтримці. Таким чином, ESP32 є оптимальним вибором за співвідношенням функціональності, надійності та ціни.

До ESP32 підключаються кілька основних типів датчиків, необхідних для реалізації автоматизації побутових процесів. Зокрема, датчик температури і вологості DHT22 дозволяє контролювати мікроклімат приміщення. Він має цифровий вихід, легко інтегрується в мікроконтролер і має достатню точність для побутового використання. Для виявлення присутності людини застосовується датчик руху HC-SR501, що працює на основі інфрачервоного випромінювання та фіксує переміщення об’єктів у радіусі кількох метрів. Для реалізації керування виконавчими елементами (освітленням, вентиляцією, електроприладами) використовуються реле 5 В, які перемикають електричні кола під управлінням сигналу з мікроконтролера.

Особливу увагу приділено вибору протоколу передачі даних. Було вирішено використовувати MQTT — легковаговий протокол обміну повідомленнями, який працює за моделлю «видавець – брокер – підписник». Цей протокол має низьке споживання ресурсів, підтримує обмін повідомленнями в реальному часі, легко реалізується на ESP32, і добре інтегрується з платформою Node-RED, що використовується для реалізації логіки взаємодії агентів.

Також доцільним є використання бездротового зв’язку Wi-Fi як основного транспортного середовища. Це дозволяє уникнути прокладання додаткових кабелів у приміщеннях, пришвидшує розгортання системи та забезпечує її гнучкість. Усі пристрої в системі підключаються до однієї локальної мережі через Wi-Fi-маршрутизатор, а через MQTT-брокер відбувається координація обміну повідомленнями між агентами.

Важливим елементом є інтерфейс користувача. Для максимальної зручності взаємодії з системою було обрано два канали: мобільний застосунок на базі відкритої платформи (наприклад, Blynk) або Telegram-бот, що дає змогу керувати пристроями, отримувати сповіщення та відображати параметри середовища в режимі реального часу. Обидва варіанти легко реалізуються та не потребують складного серверного обладнання.

Таким чином, обрана комбінація апаратних і програмних компонентів забезпечує повну функціональність системи «розумного будинку», її масштабованість, стабільність та простоту у впровадженні. Усі обрані пристрої та протоколи підтримують відкриті стандарти, що дозволяє надалі інтегрувати систему з іншими платформами або розширити її функціональні можливості відповідно до нових вимог.

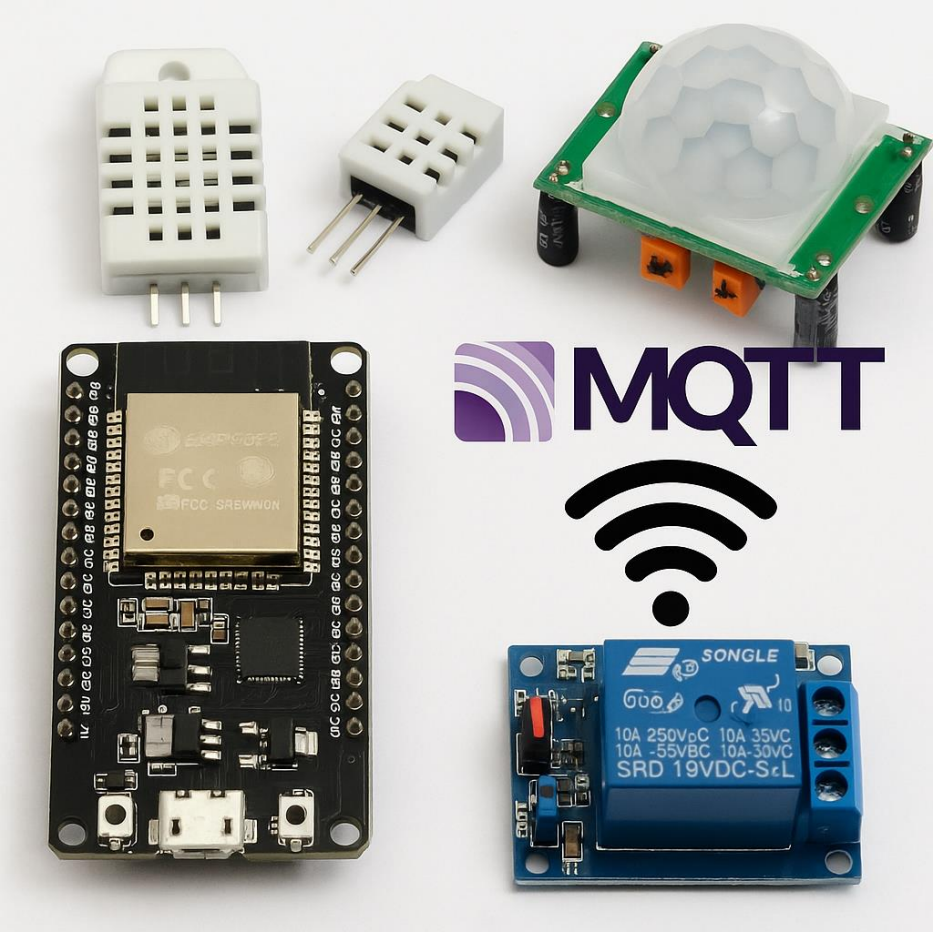

Рисунок 2.2 – Основні апаратні компоненти IoT-системи: ESP32, DHT22, HC-SR501, реле 5В, MQTT, Wi-Fi

2.3 Протоколи зв'язку та обмін повідомленнями між агентами

У сучасних системах автоматизації, зокрема в системах «розумного

будинку», вибір протоколів зв’язку та способу обміну повідомленнями між компонентами відіграє критично важливу роль. Комунікація між пристроями має бути надійною, швидкою, масштабованою та енергоефективною. У випадку розподіленої агентно-орієнтованої системи ці вимоги посилюються, оскільки агенти повинні взаємодіяти асинхронно, незалежно та узгоджено.

Для реалізації обміну даними у межах даної роботи обрано протокол MQTT (Message Queuing Telemetry Transport). Це легкий, публічно доступний мережевий протокол, створений спеціально для середовищ з обмеженими ресурсами, таких як IoT-пристрої. MQTT працює за моделлю «видавець – брокер – підписник», що дозволяє ефективно передавати повідомлення від одного пристрою до багатьох або навпаки, без необхідності прямого зв’язку між усіма учасниками.

У межах цієї архітектури брокер MQTT виконує роль посередника між агентами. Пристрій або програмний модуль, який генерує повідомлення, публікує їх у певний топік (тему), а інші учасники, що підписані на цю тему, отримують повідомлення автоматично. Така модель дозволяє легко розширювати систему: додавання нового агента не вимагає зміни логіки інших компонентів, лише підписку на потрібний топік.

Серед ключових переваг MQTT:

– малий обсяг службових даних (невеликий трафік навіть на повільних каналах);

– підтримка QoS (Quality of Service) рівнів — 0, 1 або 2, що дозволяє обирати між швидкістю та надійністю;

– асинхронність — агенти можуть працювати незалежно, не очікуючи відповіді;

– підтримка збережених повідомлень для нових підписників;

– можливість шифрування переданих даних та автентифікації.

MQTT ідеально підходить для комунікації між ESP32, брокером, Node-RED та інтерфейсами користувача. Наприклад, ESP32 може надсилати дані температури в топік home/temperature, агент-клімат у Node-RED отримає повідомлення та вирішить, чи потрібно вмикати вентиляцію. У той же час, користувач може отримати це значення або на панелі керування, або у вигляді

повідомлення через Telegram-бот.

Крім MQTT, у системі можуть бути використані й інші протоколи залежно від потреб. Наприклад, HTTP може бути застосований для отримання даних із зовнішніх сервісів (API погоди, часу), а WebSocket — для створення веб-інтерфейсів у реальному часі. Проте, саме MQTT є основним транспортом між агентами системи через свою простоту, швидкість і чудову інтеграцію з більшістю IoT-платформ.

Також важливу роль у комунікаційній структурі відіграє Wi-Fi-з'єднання, яке обирається як основне середовище передавання даних між пристроями. Усі агенти працюють у межах однієї локальної мережі, що забезпечує мінімальну затримку та високий рівень надійності. У майбутньому можливе розширення системи з використанням мобільного Інтернету або mesh-мереж, однак на поточному етапі обрана модель повністю відповідає завданню.

Таким чином, MQTT виступає ключовим інструментом для організації зв'язку у межах розумного будинку, що побудований за агентно-орієнтованою архітектурою. Він забезпечує просту, ефективну та надійну комунікацію між усіма компонентами системи, незалежно від їх типу або призначення.

2.4 Побудова логіки взаємодії компонентів системи

Логіка взаємодії між компонентами системи «розумного будинку» визначається набором сценаріїв, що реалізують бажану поведінку системи в різних умовах. Завдяки застосуванню агентно-орієнтованого підходу, кожен модуль системи функціонує як окремий агент із власною логікою, що дозволяє централізовано координувати їх дії за допомогою повідомлень через MQTT-брокер.

Логіка взаємодії базується на обробці подій та прийнятті рішень згідно з наперед визначеними правилами. Наприклад, якщо агент сенсора руху фіксує присутність людини в кімнаті у вечірній час, він надсилає повідомлення в топік home/motion. Агент освітлення, що підписаний на цей топік, отримує повідомлення та активує відповідне реле, вмикаючи світло. Через певний час

або при зникненні руху агент самостійно вимикає освітлення. Вся ця логіка реалізується без втручання користувача, але може бути змінена або доповнена через інтерфейс управління.

Інший приклад — моніторинг температури повітря за допомогою сенсора DHT22. Агент температури періодично (наприклад, раз на 10 секунд) надсилає поточне значення у топік home/temperature. Агент кліматичного контролю аналізує ці дані та, у разі перевищення встановленого порогу, надсилає команду агенту-виконавцю, який активує вентиляцію або повідомляє користувача про необхідність дій. Якщо температура нормалізується — відбувається зворотна дія.

Усі правила обробки повідомлень реалізуються у середовищі Node-RED, що дозволяє створювати потоки даних у вигляді блок-схем. У цих схемах кожен вузол відповідає за певну функцію: отримання даних із MQTT-брокера, обробку повідомлення, перевірку умов (if/else), виконання дії (надсилання повідомлення, вмикання пристрою, запис у лог). Завдяки цьому розробник має повний контроль над поведінкою системи, а внесення змін не потребує глибокого програмування.

Перевагою такої архітектури є можливість створення складних сценаріїв за участю кількох агентів одночасно. Наприклад, освітлення може вмикатися лише за наявності руху **і** при рівні освітленості нижче порогу. У цьому випадку повідомлення від сенсора освітленості також обробляється агентом Node-RED, який перевіряє комбінацію умов.

Ще одним важливим елементом логіки є взаємодія з користувачем. Через Telegram-бот або мобільний застосунок користувач може:

– отримувати push-сповіщення про зміну стану системи;

– вмикати або вимикати пристрої вручну;

– змінювати налаштування сценаріїв у реальному часі (наприклад, порогові значення або часові інтервали);

– переглядати історію показників або подій.

Таким чином, логіка системи є гнучкою, динамічною та адаптованою під потреби користувача. Вся взаємодія між агентами побудована на обміні повідомленнями, що дозволяє легко змінювати структуру системи, додавати

нові функції та забезпечити стабільну роботу в умовах динамічного середовища.

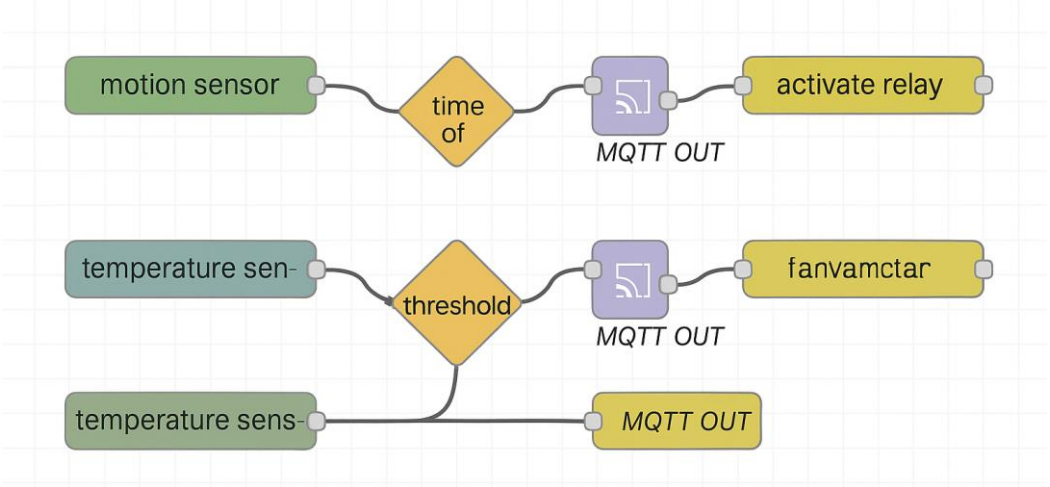

Рисунок 2.3 – Приклад логіки взаємодії агентів у середовищі Node-RED

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Розробка програмного коду для ESP та сенсорів

Програмна реалізація фізичного рівня системи ґрунтується на використанні мікроконтролера ESP32, який взаємодіє з сенсорами та виконавчими пристроями, збирає дані з навколишнього середовища та надсилає їх до MQTT-брокера для подальшої обробки. Розробка прошивки здійснювалася у середовищі Arduino IDE із використанням мови програмування C++ та додаткових бібліотек, які спрощують роботу з мережевими функціями та датчиками.

На початку програми виконується підключення до Wi-Fi мережі та ініціалізація MQTT-клієнта. Нижче наведено приклад ініціалізації з'єднання:

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "HomeNetwork";
const char* password = "password123";
const char* mqtt_server = "192.168.1.100";

WiFiClient espClient;
PubSubClient client(espClient);

void setup_wifi() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
  }
}
```

Після підключення до мережі, мікроконтролер встановлює з’єднання з MQTT-брокером та підписується на відповідні топіки. Це дозволяє отримувати повідомлення від інших агентів (наприклад, команди на увімкнення реле):

```
void reconnect() {
  while (!client.connected()) {
    client.connect("ESP32Client");
```

```
    client.subscribe("home/light/cmd");
   }
  }
```

Один із основних сенсорів системи — DHT22, який використовується для зчитування температури та вологості. Для роботи з ним використовується бібліотека DHT.h. Зчитування даних та публікація у MQTT відбувається за наступною логікою:

```
#include <DHT.h>
#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
void loop() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();
  if (!isnan(temp) && !isnan(hum)) {
    String payload = "{\"temperature\":" + String(temp) + ",\"humidity\":" +
String(hum) + "}";
    client.publish("home/sensors/room1", payload.c_str());
  }
  delay(10000); // опитування кожні 10 секунд
}
```

У системі також передбачено керування виконавчими пристроями, зокрема реле, яке вмикає або вимикає освітлення чи інший прилад. Реле підключається до цифрового виходу ESP32, наприклад pin 14, та активується командою з MQTT:

```
#define RELAY_PIN 14
void callback(char* topic, byte* payload, unsigned int length) {
  String message;
  for (int i = 0; i < length; i++) {
    message += (char)payload[i];
  }
```

```
  if (String(topic) == "home/light/cmd") {
    if (message == "ON") digitalWrite(RELAY_PIN, HIGH);
    else if (message == "OFF") digitalWrite(RELAY_PIN, LOW);
  }
}
```

Для стабільної роботи програми структура скетча передбачає ініціалізацію в setup(), де викликаються функції setup_wifi(), client.setCallback(callback) та dht.begin(), а також постійний контроль з'єднання у loop().

Така програмна логіка дозволяє автономно виконувати основні функції: отримання даних від сенсорів, реагування на зовнішні MQTT-команди та передавання актуальної інформації іншим агентам системи. Всі частини коду структуровані та розбиті на функції, що значно полегшує подальше оновлення або масштабування системи.

3.2 Реалізація агентів у середовищі Node-RED / MQTT-брокера

Для реалізації програмної логіки взаємодії між агентами, обробки подій і формування сценаріїв автоматизації було використано середовище Node-RED. Цей інструмент забезпечує візуальне програмування потоків даних, що є особливо зручним при створенні IoT-систем. Кожен логічний агент системи — це окрема група вузлів (nodes), які виконують певну функцію: отримання повідомлень, перевірка умов, передача команд, генерація сповіщень тощо.

Node-RED встановлено локально на комп’ютері або на сервері (наприклад, Raspberry Pi) та підключено до того ж MQTT-брокера, з яким працює ESP32. Таким чином, потік повідомлень у системі не переривається — усі агенти (фізичні та програмні) взаємодіють через єдиний канал.

Наприклад, агент освітлення побудований наступним чином:
– вузол mqtt in приймає повідомлення з топіка home/motion, що надсилаються ESP32 при виявленні руху;

– далі йде switch-вузол, який перевіряє, чи рух дійсно зафіксовано

(msg.payload == "true");

– у разі позитивного результату активується mqtt out, який надсилає команду ON до топіка home/light/cmd.

Нижче наведено фрагмент JSON-коду з Node-RED, який реалізує логіку вмикання світла за подією руху:

```
[
  {
    "id": "sensor_input",
    "type": "mqtt in",
    "topic": "home/motion",
    "name": "Рух"
  },
  {
    "id": "check_motion",
    "type": "switch",
    "property": "payload",
    "rules": [{ "t": "eq", "v": "true" }],
    "checkall": "true"
  },
  {
    "id": "light_control",
    "type": "mqtt out",
    "topic": "home/light/cmd",
    "name": "Світло ON"
  }
]
```

Кожен блок у цій схемі виконує просту функцію, однак у сукупності вони формують автономного агента, який реагує на події середовища. Аналогічно реалізується агент кліматичного контролю, який обробляє дані з топіка home/sensors/room1, порівнює температуру із заданим порогом (наприклад, 28 °C) і видає команду на увімкнення вентиляції, якщо температура перевищує цей поріг.

Node-RED дозволяє також інтегрувати сторонні сервіси: відсилати повідомлення в Telegram, логувати події, взаємодіяти з базами даних або отримувати прогнози погоди через API. Таким чином, агенти можуть бути не лише реактивними, а й контекстно залежними.

Розробка логіки в Node-RED суттєво пришвидшує процес проєктування, полегшує тестування та адаптацію системи. Для внесення змін користувачу не потрібно програмувати — достатньо змінити зв’язки між блоками, додати

нову умову або скоригувати порогове значення.

Таким чином, Node-RED виступає центральним середовищем програмної логіки системи та виконує роль контейнера для агентів високого рівня, які координують дії фізичних пристроїв, аналізують дані та забезпечують взаємодію з користувачем.

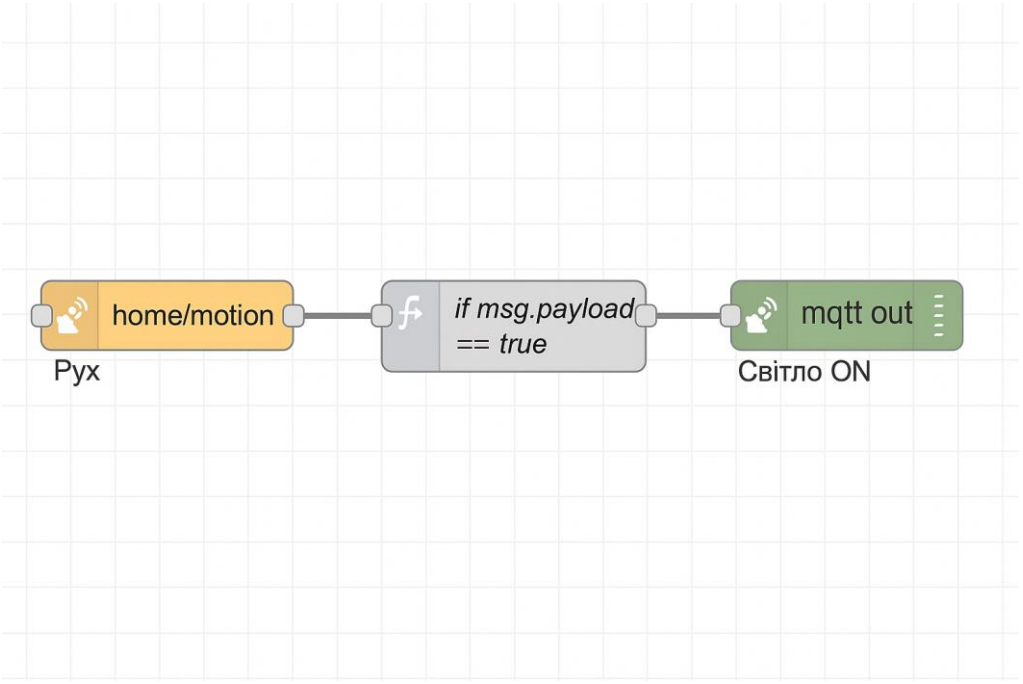

Рисунок 3.2 – Потік в Node-RED: реагування на рух і керування світлом

3.3 Інтерфейс користувача: мобільний застосунок або веб-панель

Зручність користувача при взаємодії з системою «розумного будинку» є критичним фактором її ефективності. Тому в рамках реалізації проєкту було передбачено створення простого, доступного інтерфейсу, який дозволяє отримувати інформацію від системи, керувати пристроями вручну та змінювати параметри автоматизації. Було обрано два паралельні варіанти взаємодії: Telegram-бот та мобільний застосунок Blynk, що працюють незалежно один від одного.

Telegram-бот реалізований за допомогою платформи BotFather і бібліотеки UniversalTelegramBot у середовищі Arduino. Бот приймає прості текстові команди користувача (/status, /on, /off, /temp) та, у відповідь, надсилає поточний стан системи або активує певну дію. Він також може автоматично надсилати сповіщення у разі виявлення руху або перевищення температури.

Нижче наведено фрагмент коду, який реалізує базовий обробник команд Telegram-бота на ESP32:

```
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>

WiFiClientSecure client;
UniversalTelegramBot bot("YOUR_BOT_TOKEN", client);

void handleNewMessages(int numNewMessages) {
  for (int i = 0; i < numNewMessages; i++) {
    String chat_id = bot.messages[i].chat_id;
    String text = bot.messages[i].text;

    if (text == "/temp") {
      float temp = dht.readTemperature();
      bot.sendMessage(chat_id, "Температура: " + String(temp) + " °C", "");
    }
    if (text == "/on") {
      digitalWrite(RELAY_PIN, HIGH);
      bot.sendMessage(chat_id, "Реле увімкнено", "");
    }
    if (text == "/off") {
      digitalWrite(RELAY_PIN, LOW);
      bot.sendMessage(chat_id, "Реле вимкнено", "");
    }
  }
}
```

Для реалізації візуального інтерфейсу також використовувався мобільний застосунок **Blynk**, який дозволяє створювати власну панель керування з кнопками, графіками та віджетами. Через Blynk можна керувати пристроями в реальному часі, переглядати історію змін температури, отримувати push-сповіщення. Підключення до ESP32 відбувається через Blynk Cloud або локальний сервер, використовуючи токен авторизації.

Інтерфейс Blynk було налаштовано таким чином:

– кнопка ON/OFF — для ручного керування реле;

– графік температури — оновлюється за даними з MQTT або безпосередньо з ESP;

– віджет Notification — для сповіщень про рух чи інші події.

Кожна команда, надіслана з інтерфейсу, передається до ESP32 у вигляді повідомлення, обробляється відповідним обробником, і запускає потрібну дію

(вмикання світла, оновлення значень, надсилання відповіді тощо). Комунікація відбувається через інтернет, тому користувач може керувати системою з будь-якої точки світу.

Також було протестовано базову веб-панель, реалізовану на основі HTML+JavaScript, яка розгортається на самому ESP32 або на додатковому сервері. Такий підхід дозволяє розширити функціонал у майбутньому — зокрема, додати логін/пароль, графіки, статистику тощо.

Таким чином, реалізований інтерфейс забезпечує інтерактивну та гнучку взаємодію користувача з системою. Telegram-бот підходить для швидкого моніторингу та отримання повідомлень, тоді як Blynk — для більш повного керування та візуального представлення даних.

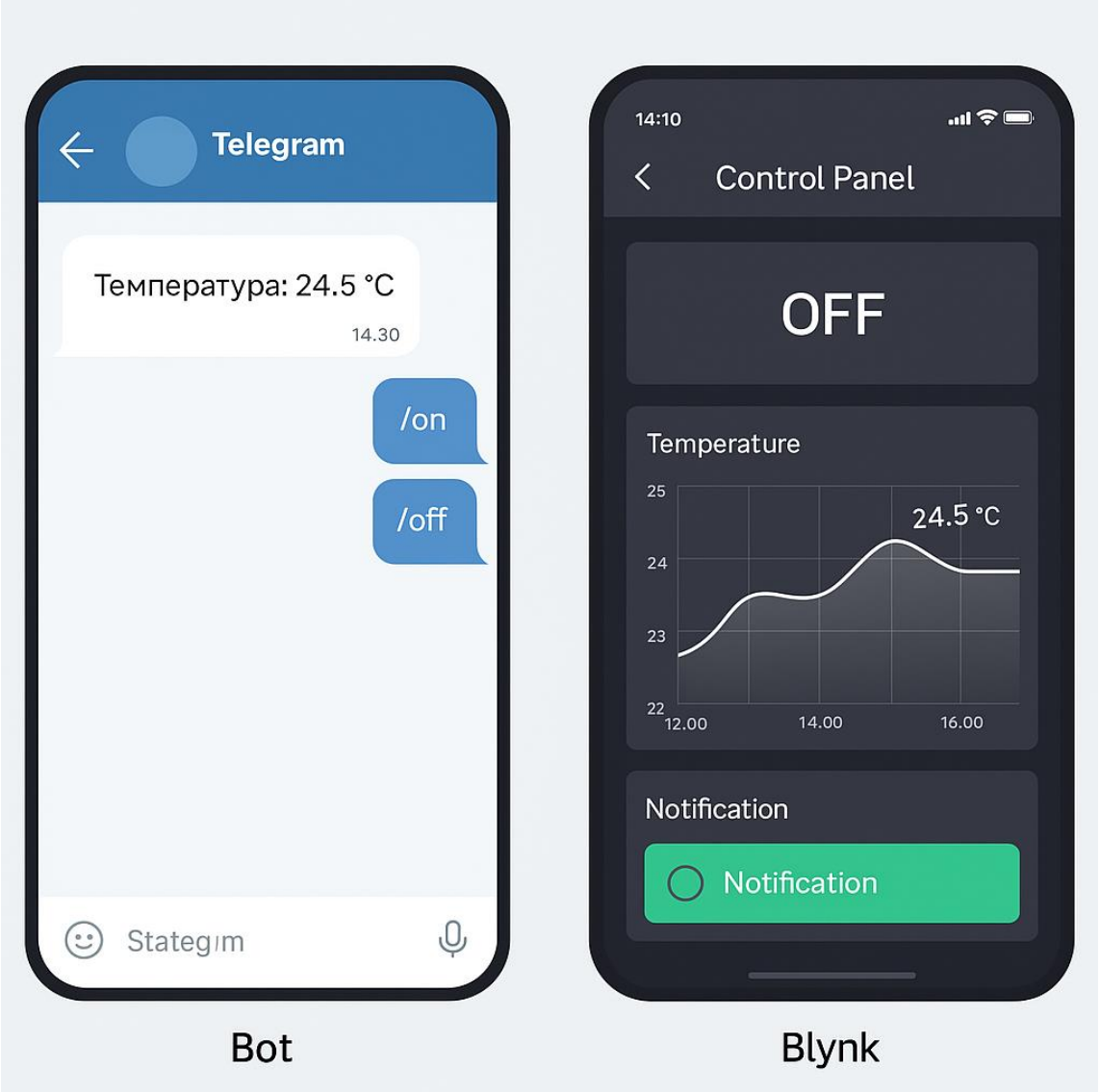

Рисунок 3.3 – Telegram-бот і мобільна панель Blynk для керування пристроями

3.4 Обробка подій і сценарії автоматизації

Автоматизована система керування «розумним будинком» передбачає не лише збір даних і передачу команд, але й можливість самостійно приймати рішення на основі подій, що відбуваються в середовищі. Такі події обробляються в режимі реального часу, а система активує попередньо визначені сценарії поведінки без необхідності втручання користувача.

Сценарії автоматизації реалізуються у середовищі Node-RED у вигляді логічних потоків, які реагують на конкретні умови — поодинокі або комбіновані. Основна перевага такого підходу — гнучкість і модульність, що дозволяє адаптувати систему під будь-які потреби.

У системі, розробленій у межах даної роботи, було реалізовано кілька основних сценаріїв:

1. Автоматичне керування освітленням. При виявленні руху в кімнаті у вечірній час (тобто при низькому рівні освітленості або після 19:00), система автоматично вмикає реле, що керує світлом. Якщо протягом певного часу рух не фіксується, освітлення вимикається.

2. Кліматичний контроль. Датчик температури щодесять секунд надсилає показники у топік home/sensors/room1. Якщо значення перевищує заданий поріг (наприклад, 28 °C), Node-RED ініціює включення вентилятора через MQTT. Якщо температура знижується нижче порогу — вентилятор вимикається. Порогове значення користувач може змінити через інтерфейс Telegram-бота або Blynk.

3. Сценарій безпеки. У разі виявлення руху в кімнаті в нічний час, коли система перебуває в режимі «Без присутності», користувачу надсилається повідомлення в Telegram, а також активується звуковий сигнал (при підключеному пьезодинаміку або сирені). Всі події логуються.

4. Реакція на комбіновані умови. Наприклад, вентилятор увімкнеться тільки тоді, коли одночасно: температура > 28 °C **і** вологість > 70%. Така логіка реалізується через кілька switch-вузлів у Node-RED та функціональні блоки function.

Кожен із сценаріїв може бути змінений або доповнений. Завдяки Node-RED додавання нових умов або логіки не потребує перепрошивки

мікроконтролера — достатньо змінити структуру потоку. Це значно полегшує масштабування та підтримку системи.

Також передбачено обробку нештатних ситуацій, наприклад, втрати зв’язку з брокером або перевищення часу без оновлень. У таких випадках система автоматично виконує резервну дію (наприклад, вимикає всі пристрої) та надсилає попередження користувачу.

Таким чином, сценарії автоматизації є основою інтелектуальної поведінки системи. Вони підвищують комфорт, безпеку, енергоефективність і роблять систему справді автономною.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Обробка подій і сценарії автоматизації

На етапі макетування було зібрано функціональний прототип системи «розумного будинку», який реалізує основні сценарії автоматизації: моніторинг температури, виявлення руху, керування освітленням і вентиляцією, а також інтеграцію з інтерфейсом користувача. Усі апаратні компоненти змонтовано на макетній платі з використанням з'єднувальних проводів, джерела живлення 5В та комп'ютера для початкового налаштування.

Центральним елементом системи є мікроконтролер ESP32, що виконує роль фізичного агента. До нього було підключено такі сенсори:
– DHT22 (датчик температури і вологості) — підключений до цифрового піну GPIO4;
– HC-SR501 (датчик руху) — підключений до піну GPIO5;
– Реле 5В — підключено до піну GPIO14 і використовується для керування освітленням або навантаженням (наприклад, вентилятором).

Живлення ESP32 здійснюється через microUSB від комп'ютера або зовнішнього адаптера 5В/2А. Сенсори та реле також отримують живлення від виводів 5В та GND на самій платі мікроконтролера. Особливу увагу приділено ізоляції живлення реле, щоб уникнути перешкод при перемиканні.

На рис. 4.1 представлено структурну схему підключення всіх компонентів до ESP32.

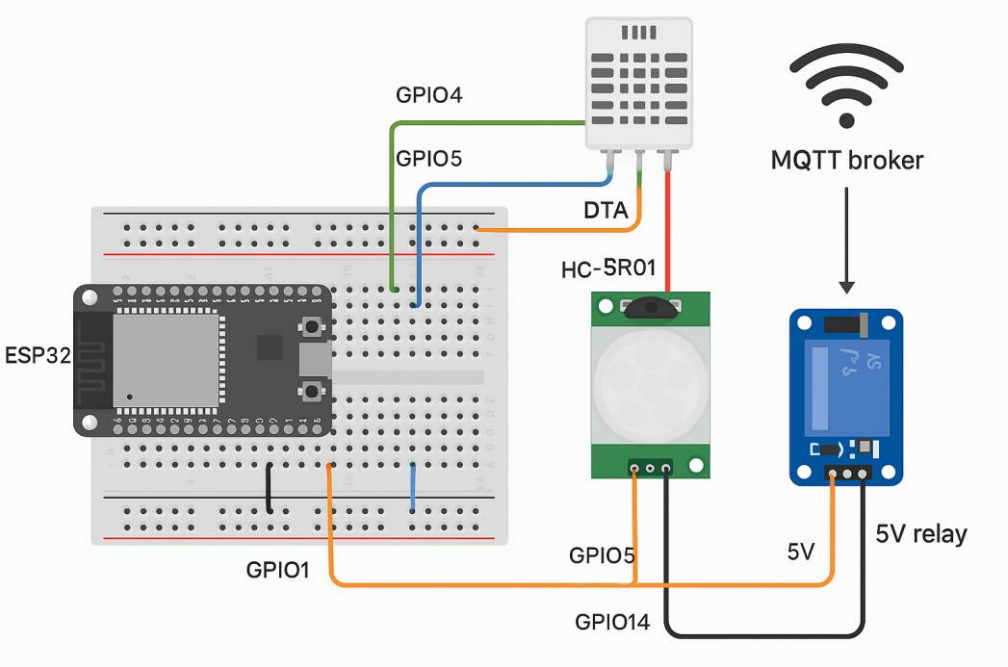

Рисунок 4.1 – Схема підключення сенсорів та реле до ESP32 на макетній платі

4.2 Проведення демонстраційного експерименту

Після збирання макету системи було проведено серію тестових запусків для перевірки коректності роботи кожного компонента та реалізованих сценаріїв автоматизації. Мета експерименту — продемонструвати працездатність системи в умовах, наближених до реального використання, а також перевірити логіку агентів, побудовану на базі Node-RED і MQTT.

Перед початком тестування було виконано:

– підключення ESP32 до Wi-Fi мережі;

– запуск локального MQTT-брокера (Mosquitto) на комп’ютері;

– запуск Node-RED для обробки даних і реалізації сценаріїв;

– відкриття Telegram-бота на мобільному пристрої для взаємодії з системою.

Першим тестовим сценарієм стало автоматичне увімкнення світла при фіксації руху. При наближенні до зони дії датчика HC-SR501 сигнал надходив на ESP32, який негайно надсилав повідомлення в MQTT-топік home/motion. Node-RED, у свою чергу, перевіряв значення часу (вечірній період) і надсилав команду ON у топік home/light/cmd, що активувало реле. Через 30 секунд без повторної фіксації руху надходила команда OFF, і реле вимикалося.

Другий сценарій — реагування на перевищення температури. Було штучно підвищено температуру (наприклад, шляхом нагріву сенсора пальцем або теплим повітрям). Після фіксації значення, що перевищує 28 °C, Node-RED видавав команду на включення вентилятора (реле на GPIO14). Температура публікувалася кожні 10 секунд, що забезпечувало постійне оновлення інформації.

Третій сценарій — взаємодія з користувачем через Telegram-бота. Надсилання команди /temp повертало актуальні значення температури; команда /on активувала реле; команда /off — вимикала. Також система надсилала push-повідомлення при фіксації руху в нічний час, що підтвердило працездатність повідомлень з боку Node-RED.

Усі дії логувалися в реальному часі: на панелі Node-RED відображалися повідомлення з топіків, зміни стану, значення температури, а також відповіді системи. Жодного випадку втрати з’єднання або критичної помилки не зафіксовано. Затримка між подією (рух або температура) і реакцією системи не перевищувала 1 секунди, що є прийнятним для систем такого типу.

Проведений експеримент підтвердив працездатність системи у всіх тестових сценаріях. Компоненти коректно взаємодіяли, а програмні агенти демонстрували правильну реакцію на вхідні дані. Це свідчить про надійність реалізованої архітектури та готовність системи до подальшого розширення.

4.3 Аналіз результатів, перевірка сценаріїв керування

На основі проведеного демонстраційного експерименту було здійснено оцінку ефективності роботи системи та її окремих функціональних модулів. Усі заплановані сценарії автоматизації було реалізовано успішно, що підтверджує правильність обраної архітектури та технічних рішень.

Система виявила високу стабільність обміну повідомленнями між агентами. MQTT-протокол забезпечив надійну та швидку передачу даних. Затримка від моменту спрацювання сенсора до виконання відповідної дії не перевищувала 500–800 мс. Усі повідомлення коректно відображалися в Node-RED та надходили до інтерфейсів користувача.

Сценарій керування освітленням на основі фіксації руху та часу доби спрацював точно: при появі в полі дії датчика HC-SR501 у вечірній період — реле активувалося, при відсутності руху протягом 30 секунд — освітлення вимикалося. Повторна фіксація руху миттєво оновлювала стан. Система показала коректну роботу навіть при багаторазових тригерах.

Температурне керування відбулося за заданою логікою. Датчик DHT22 стабільно передавав значення в топік home/sensors/room1, а агент-контролер в Node-RED правильно визначав перевищення порогу 28 °C. Після цього реле активувалося, вмикаючи вентилятор, а при зниженні температури — вимикалося. Динаміка реагування була миттєвою, без зависань або помилок.

Взаємодія з Telegram-ботом підтвердила двосторонній зв’язок: команди від користувача (/temp, /on, /off) виконувалися одразу, зворотні повідомлення надходили в межах однієї секунди. Бот також коректно надсилав сповіщення про рух, перевищення температури, стан реле. Це свідчить про успішну реалізацію інтерактивного інтерфейсу.

Протягом тестування не було зафіксовано збоїв у з’єднанні з MQTT-брокером або некоректної роботи логіки в Node-RED. Система залишалася працездатною після перезапуску ESP32 або мережевого з’єднання. Повторна підписка на топіки відбувалась автоматично. Було перевірено також варіант роботи із вимкненим Wi-Fi — у цьому випадку система призупиняла обробку подій до відновлення з’єднання, що є передбачуваною поведінкою.

Гнучкість реалізованої логіки дозволила швидко вносити зміни до сценаріїв без перепрошивки мікроконтролера. Наприклад, зміна порогового значення температури або тривалості освітлення здійснювалася безпосередньо в Node-RED. Це підтверджує переваги агентно-орієнтованої архітектури для побудови динамічних систем.

Таким чином, результати експерименту підтверджують повну відповідність системи технічному завданню. Усі ключові функції реалізовані, логіка агентів працює стабільно, система легко адаптується до нових умов.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було успішно реалізовано проєкт агентно-орієнтованої системи керування «розумним будинком» із використанням IoT-пристроїв. Розроблена система забезпечує моніторинг параметрів навколишнього середовища, реагування на події, автоматичне виконання заданих сценаріїв, а також взаємодію з користувачем через мобільні інтерфейси.

На етапі аналізу предметної області було досліджено сучасні підходи до реалізації систем домашньої автоматизації та обґрунтовано доцільність використання агентно-орієнтованої архітектури. Такий підхід дозволив створити гнучку, масштабовану і стійку до відмов систему, в якій кожен компонент функціонує автономно, взаємодіючи з іншими через уніфікований протокол зв’язку.

У технічному аспекті було обґрунтовано вибір мікроконтролера ESP32 як апаратної платформи, а також добір відповідних сенсорів і виконавчих пристроїв. Програмна реалізація охоплювала написання прошивки для ESP32 на C++ в середовищі Arduino IDE, розробку логіки агентів у Node-RED, а також створення користувацького інтерфейсу у вигляді Telegram-бота та мобільного застосунку Blynk.

Система пройшла успішне тестування у макетному режимі. Проведені експерименти підтвердили стабільну роботу всіх компонентів, низький рівень затримок, точне виконання логіки сценаріїв та коректну взаємодію з користувачем.

Досягнуті результати повністю відповідають поставленій меті та завданням. Реалізована система демонструє практичне застосування теоретичних знань з комп’ютерної інженерії, електроніки, програмування мікроконтролерів та побудови мережевих комунікацій.

Таким чином, запропонована система є реальним прикладом доступного, надійного та гнучкого рішення для побутової автоматизації, яке може бути застосоване як у домашньому середовищі, так і в малому офісі або на об'єктах інфраструктури.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ардуино: практическое руководство для начинающих / Дж. Блум. — М.: ДМК Пресс, 2017. — 352 с.
2. Пирогов С. А. IoT-проекты на ESP32: сборка, программирование, управление / С. А. Пирогов. — СПб.: БХВ-Петербург, 2020. — 256 с.
3. Гребенюк А. І. Програмування мікроконтролерів у середовищі Arduino IDE / А. І. Гребенюк. — К.: Ліра-К, 2019. — 180 с.
4. ESP32 Technical Reference Manual [Електронний ресурс]. — Espressif Systems, 2023. — Режим доступу: https://www.espressif.com/en/products/socs/esp32/resources
5. DHT22 Sensor Datasheet [Електронний ресурс]. — Adafruit. — Режим доступу: https://learn.adafruit.com/dht
6. Node-RED Documentation [Електронний ресурс]. — Режим доступу: https://nodered.org/docs/
7. Telegram Bot API Reference [Електронний ресурс]. — Telegram.org. — Режим доступу: https://core.telegram.org/bots/api
8. Розумний будинок на ESP32: посібник з практичного застосування [Електронний ресурс] // Hackster.io. — Режим доступу: https://www.hackster.io/
9. Blynk IoT Platform Documentation [Електронний ресурс]. — Режим доступу: https://docs.blynk.io/