ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА
(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерних систем та технологій
(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«Розробка системи Розумний будинок»

«Development of the Smart House system»

Виконав: здобувач денної форми навчання
Спеціальності 151-Автоматизація та комп'ютерно-інтегровані технології
(код, назва спеціальності)
Освітня програма Комп'ютерна обробка та аналіз даних
(назва)
Бригар Костянтин Олександрович
(прізвище, ім'я, по-батькові здобувача)

Керівник к. т. н., доцент Ларін Д. Г. _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)
Рецензент д.т. н., проф. Гунченко Ю.О.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Комп'ютерних систем та технологій
№ ___ від ___.___.20__ р.

Завідувач кафедри

_____ _____
(підпис) (прізвище, ім'я)

Захищено на засіданні ЕК № ____
протокол № __ від ___.___.20__ р.

Оцінка _____/_____/_____
(за національною шкалою / шкалою ECTS / бали)

Голова ЕК

_____ _____ _____
(підпис) (прізвище, ім'я)

Одеса 2025

АНОТАЦІЯ

В кваліфікаційній роботі розглянуті та проаналізовані сучасні системи побудови розумного будинку. Проведено аналіз існуючих рішень, систем управління розумними будинками. Виконано проектування плану розташування обладнання по зонах будинку. Визначено функціонал системи. Обране обладнання для реалізації функцій систем розумного будинку. Запропанована система управління на базі ArduinoNano та ESP8266. Побудовані принципові схеми пристроїв розумного будинку. Розроблені програми, яка керують пристроями розумного будинку та виконує команди з управляючого серверу. Проведене тестування системи розумного будинку.

ABSTRACT

The qualification work considered and analyzed modern systems for building a smart home. An analysis of existing solutions and smart home management systems was carried out. A design of the equipment layout plan for the zones of the house was carried out. The functionality of the system was determined. Equipment was selected to implement the functions of smart home systems. A control system based on ArduinoNano and ESP8266 was proposed. Schematic diagrams of smart home devices were built. Programs were developed that control smart home devices and execute commands from the control server. The smart home system was tested.

ЗМІСТ

ВСТУП

У сучасному світі стрімкий розвиток цифрових технологій суттєво впливає на всі сфери життя людини, зокрема на побут. Автоматизація, інтернет речей (IoT), штучний інтелект і мобільні додатки вже давно перестали бути атрибутами виключно виробництва або високотехнологічних галузей — вони дедалі частіше інтегруються у повсякденне життя звичайної людини. Однією з найбільш актуальних тенденцій у цьому напрямку є впровадження технологій «розумного будинку».

Розумний будинок (англ. *Smart Home*) — це комплекс технічних рішень, спрямованих на автоматизацію, оптимізацію та контроль над різними системами житла, такими як освітлення, опалення, вентиляція, безпека, енергозбереження та інше. Завдяки використанню датчиків, контролерів, мережевих технологій і мобільних інтерфейсів, система розумного будинку дозволяє значно підвищити рівень комфорту, безпеки та ефективності використання ресурсів.

У зв'язку з глобальними екологічними викликами, зростанням вартості енергоносіїв та необхідністю підвищення рівня безпеки житлових приміщень, попит на інтелектуальні системи управління будинком стрімко зростає. Крім того, пандемія COVID-19 посилила тренд на організацію зручного, автоматизованого домашнього простору, де люди працюють, навчаються і відпочивають. Відтак, автоматизовані системи управління житлом стають не розкішшю, а логічним етапом розвитку побутової інфраструктури.

Технології розумного будинку дозволяють вирішувати цілий спектр завдань, таких як зменшення витрат на електроенергію та тепло завдяки оптимізації роботи систем, підвищення рівня безпеки через використання датчиків руху, камер відеоспостереження, сигналізацій, забезпечення дистанційного керування елементами будинку через смартфон або інші пристрої, створення сценаріїв автоматизації (наприклад, вимкнення

освітлення при виході з будинку, регулювання температури залежно від часу доби тощо).

Однак, незважаючи на значний прогрес у галузі "розумних будинків", існує ряд невирішених проблем та напрямків для подальшого розвитку. Так існує проблема сумісності та інтероперабельності різних пристроїв та протоколів зв'язку, пов’язана з тим, що на ринку представлено велику кількість пристроїв від різних виробників, які часто використовують власні протоколи зв'язку, що ускладнює їх інтеграцію в єдину систему. Не до кінця вирішені питання безпеки та конфіденційності даних. Зі збільшенням кількості підключених пристроїв зростають ризики несанкціонованого доступу до особистих даних та управління системами будинку. Існує складність налаштування та використання для непідготовлених користувачів. Та, крім того, повноцінні системи "розумний будинок", що включають широкий спектр функціональності, можуть бути досить дорогими, що є суттєвим бар'єром для багатьох потенційних користувачів.

Таким чином, розробка ефективних, безпечних, зручних у використанні та доступних систем "розумний будинок" є актуальною науково-технічною задачею, що має значне практичне значення. Головна задача моєї роботи полягає у розробці недорогого, енергоефективного та гнучкого прототипу розумного будинку, який може бути масштабований та адаптований до різних умов експлуатації.

1 ОГЛЯД І АНАЛІЗ СУЧАСНИХ СИСТЕМ «РОЗУМНОГО БУДИНКУ»

1.1 Історія розвитку концепції розумного будинку

Концепція "розумного будинку", або інтелектуального житла, пройшла довгий та еволюційний шлях від футуристичних уявлень до реальних технологічних рішень, які сьогодні активно впроваджуються в повсякденне життя. Її історію можна поділити на кілька ключових етапів, кожен з яких характеризується певними технологічними досягненнями, зміною уявлень про можливості автоматизації та появою нових потреб користувачів.

Зародження ідеї автоматизованого житла сягає початку XX століття та тісно пов'язане з розвитком електрифікації та перших побутових електроприладів. У цей період з'являються перші механічні та електромеханічні пристрої, покликані полегшити домашню роботу. Хоча вони ще не були "розумними" у сучасному розумінні, вони заклали основу для автоматизації рутинних процесів.

Одним з ранніх прикладів можна вважати автоматичні пральні машини, пилососи та інші прилади, які звільняли час домогосподарок. У науково-фантастичній літературі того часу також почали з'являтися образи повністю автоматизованих будинків, здатних самостійно виконувати різноманітні функції. Варто згадати, наприклад, твори Жуля Верна, де описувалися технологічно розвинені житла майбутнього.

З появою перших комерційно доступних комп'ютерів та розвитком електроніки відкрилися нові можливості для автоматизації. У цей період починаються перші експерименти зі створення систем управління домашнім обладнанням за допомогою комп'ютерів. Ці проекти були переважно дослідницькими та демонстрували потенціал інтеграції різних пристроїв.

Одним з відомих ранніх проектів був ECHO IV (Electronic Computing Home Operator), розроблений Джимом Сазерлендом у 1966 році. Ця система

могла керувати освітленням, температурою та побутовими приладами за допомогою голосових команд. Хоча ECHO IV був досить громіздким та складним, він став важливим кроком у розвитку концепції "розумного будинку" [1].

У 1970-х роках з'являються перші спроби створення промислових стандартів для домашньої автоматизації, хоча вони ще не набули широкого поширення.

1980-ті та 1990-ті роки стали важливим етапом у розвитку "розумного будинку" завдяки появі перших стандартизованих протоколів зв'язку, призначених спеціально для домашньої автоматизації.

Одним з найвідоміших протоколів цього періоду є X10, розроблений у 1975 році. X10 використовує існуючу електропроводку для передачі сигналів управління між пристроями. Незважаючи на свою відносну простоту та низьку вартість, X10 мав обмеження у швидкості передачі даних та надійності зв'язку [2]. Проте, він став одним з перших широко використовуваних стандартів для домашньої автоматизації.

У цей період також з'являються перші комерційні системи "розумний будинок", хоча вони були досить дорогими та орієнтованими на вузький сегмент ринку. Ці системи часто включали централізований контролер, який керував освітленням, опаленням та системами безпеки.

Поява та широке поширення Інтернету та бездротових технологій, таких як Wi-Fi, Bluetooth та Zigbee, зробили революцію у розвитку "розумного будинку". Ці технології забезпечили більш гнучкі, надійні та прості у встановленні рішення для домашньої автоматизації.

У 2000-х роках з'являється велика кількість нових пристроїв та систем "розумний будинок", які використовують IP-мережі для зв'язку та управління. Стають популярними системи дистанційного керування через веб-інтерфейси та мобільні додатки.

З'являються нові протоколи зв'язку, такі як Z-Wave (розроблений у 2001 році), який забезпечує більш надійний та енергоефективний

бездротовий зв'язок порівняно з X10 [3]. Zigbee (стандартизований у 2003 році) стає ще одним популярним протоколом, особливо для низькопотужних пристроїв з mesh-мережевою топологією [4].

Особливо важливим етапом стає розвиток екосистем "розумний будинок" від великих технологічних компаній, таких як Amazon (з Alexa), Google (з Google Home/Nest) та Apple (з HomeKit). Ці екосистеми об'єднують велику кількість різноманітних пристроїв від різних виробників та надають користувачам зручні інструменти для керування ними за допомогою голосових команд та мобільних додатків.

Сьогодні концепція "розумного будинку" охоплює широкий спектр технологій та застосувань, включаючи:

- Керування яскравістю, кольором та розкладом освітлення.
- Автоматичне регулювання температури для економії енергії та забезпечення комфорту.
- Дистанційне керування доступом, відеоспостереження, датчики руху та сигналізації.
- Холодильники, пральні машини, кавоварки та інші прилади з можливостями дистанційного керування та автоматизації.
- Інтеграція аудіо- та відеотехніки, голосове керування мультимедіа.
- Моніторинг та оптимізація використання електроенергії.

Концепція "розумного будинку" продовжує активно розвиватися. Очікується подальша інтеграція штучного інтелекту та машинного навчання, що дозволить системам ставати більш проактивними та адаптивними до потреб користувачів. Розвиток технологій 5G та інших високошвидкісних мереж сприятиме більш надійному та швидкому обміну даними між пристроями.

Одним з ключових напрямків розвитку є також підвищення рівня безпеки та конфіденційності даних користувачів. Зі зростанням кількості

підключених пристроїв питання захисту від кіберзагроз стає все більш актуальним.

Очікується також подальше зниження вартості компонентів та спрощення процесу встановлення та налаштування систем "розумний будинок", що зробить їх доступнішими для ширшого кола споживачів [5,6].

Історія розвитку концепції "розумного будинку" є прикладом поступової еволюції від простих механічних пристроїв до складних інтегрованих систем, керованих за допомогою комп'ютерів, бездротових технологій та голосових помічників. Кожен етап розвитку був зумовлений технологічними проривами та зміною уявлень про можливості автоматизації житлового простору. Сьогодні "розумний будинок" перестає бути розкішшю та **стає** все більш доступним і функціональним інструментом для підвищення комфорту, безпеки та ефективності нашого життя.

1.2. Основні поняття і терміни

Система розумного будинку — це інтегрована система, яка використовує програмно-апаратні засоби для моніторингу, керування та автоматизації функцій житлового приміщення з метою забезпечення комфорту, безпеки, енергозбереження та зручності.

До складу типового розумного будинку можуть входити:

- Центральний контролер (Hub, Gateway, Controller) — Пристрій, який служить центральним вузлом для зв'язку та управління різними "розумними" пристроями в будинку. Він агрегує дані від датчиків, обробляє їх та передає команди на виконавчі пристрої.
- Кінцеві пристрої (End Devices) – Фізичні пристрої в будинку, які виконують певні функції та можуть бути керовані або контролюватися системою "розумний будинок".
- Датчики (Sensors)— Пристрої, які вимірюють певні фізичні параметри навколишнього середовища або стан об'єктів та перетворюють їх на

електричні сигнали для подальшої обробки. Фіксують зміну параметрів навколишнього середовища (рух, температура, вологість, дим, відкриття дверей тощо);

- Виконавчі механізми (Actuators**)** — Пристрої, які виконують певні дії на основі отриманих команд від центрального контролера або інших компонентів системи. Розумні реле (для керування освітленням або розетками), сервоприводи (для керування клапанами або жалюзі), динаміки, дисплеї.
- Комунікаційні інтерфейси (Communication Protocols) — Набори правил та процедур, які визначають формат та послідовність обміну даними між різними пристроями в мережі "розумного будинку". Як приклад - Wi-Fi (IEEE 802.11) (Поширений стандарт бездротового зв'язку для підключення до локальної мережі та Інтернету), Bluetooth (Протокол бездротового зв'язку малого радіусу дії, часто використовується для прямого підключення пристроїв до смартфонів або хабів), Zigbee (IEEE 802.15.4) (Низькопотужний бездротовий протокол, оптимізований для мереж з великою кількістю невеликих пристроїв), Z-Wave (Бездротовий протокол, розроблений спеціально для домашньої автоматизації, характеризується низьким енергоспоживанням та надійною mesh-мережею), X10 (Застарілий протокол, що використовує електропроводку для передачі сигналів), Thread (Ще один низькопотужний бездротовий протокол на основі IP, розроблений для IoT).
- Інтерфейс користувача (User Interface, UI) — Засіб, за допомогою якого користувач взаємодіє з системою "розумний будинок". Реалізується як мобільні додатки, веб-інтерфейси, голосові помічники, сенсорні панелі, фізичні пульти керування.мобільний додаток, вебінтерфейс або голосовий асистент.
- Безпека "розумного будинку" (Smart Home Security) -- Заходи, спрямовані на захист системи "розумний будинок" від

несанкціонованого доступу, кібератак та витоку особистих даних. Шифрування даних, надійні паролі, оновлення програмного забезпечення, захист мережі Wi-Fi.

1.3 Класифікація систем розумного будинку

За способом управління

- Автономні системи "Розумний будинок" (Standalone Systems):

Автономні системи "Розумний будинок" являють собою локальні рішення, які функціонують незалежно від зовнішніх мереж, зокрема від Інтернету. Управління такими системами зазвичай здійснюється безпосередньо в межах будинку за допомогою локальних інтерфейсів, таких як пульти дистанційного керування, настінні панелі або внутрішня бездротова мережа (наприклад, Bluetooth). Кожен пристрій або підсистема в автономній системі, як правило, є самодостатнім та виконує свої функції без постійного зв'язку з центральним контролером або зовнішніми серверами.

Прикладами автономних систем можуть бути прості системи автоматичного освітлення з датчиками руху, які вмикають світло при виявленні присутності людини та вимикають його через певний час. Також до цього типу можна віднести локальні системи безпеки з датчиками та сиренами, які спрацьовують при виявленні вторгнення, але не передають інформацію за межі будинку. Автономними можуть бути і деякі системи опалення або кондиціонування з програмованими термостатами, які працюють за заданим розкладом без зовнішнього керування.

Основною перевагою автономних систем є їхня незалежність від зовнішніх мереж, що підвищує їхню надійність у випадку відсутності інтернет-з'єднання. Вони також можуть бути більш простими в установці та налаштуванні, оскільки не потребують складних мережевих конфігурацій. Крім того, автономні системи можуть забезпечувати вищий рівень конфіденційності, оскільки дані про роботу пристроїв не передаються на

зовнішні сервери. Однак, їхні функціональні можливості часто обмежені локальним управлінням і вони не надають можливості дистанційного контролю або інтеграції з іншими онлайн-сервісами.

-- Мережеві системи "Розумний будинок" (Networked Systems):

Мережеві системи "Розумний будинок", на відміну від автономних, передбачають обов'язкове підключення до локальної мережі (часто з доступом до Інтернету) та централізоване управління через хаб або хмарну платформу. Всі пристрої в такій системі обмінюються даними між собою та з центральним контролером за допомогою різних протоколів зв'язку (Wi-Fi, Zigbee, Z-Wave тощо). Це забезпечує значно ширші функціональні можливості та гнучкість управління.

Мережеві системи дозволяють користувачам контролювати всі аспекти "розумного будинку" дистанційно за допомогою мобільних додатків, веб-інтерфейсів або голосових помічників. Вони також підтримують складні сценарії автоматизації, які можуть включати взаємодію між різними типами пристроїв. Наприклад, при спрацюванні датчика руху може вмикатися світло, активуватися камера відеоспостереження та надсилатися повідомлення на смартфон власника. Мережеві системи часто інтегруються з хмарними сервісами для зберігання даних, аналітики та розширення функціональності.

Перевагами мережевих систем є їхні широкі функціональні можливості, зручність дистанційного керування, можливість інтеграції з різними сервісами та гнучкість у налаштуванні складних сценаріїв автоматизації. Однак, вони залежать від стабільного інтернет-з'єднання для повноцінної роботи, а їхня складність може ускладнювати встановлення та налаштування. Крім того, питання безпеки та конфіденційності даних є більш актуальними для мережевих систем, оскільки інформація про роботу пристроїв може передаватися через Інтернет та зберігатися на зовнішніх серверах.

Таким чином, вибір між автономною та мережевою системою "Розумний будинок" залежить від конкретних потреб користувача, бюджету,

вимог до функціональності та рівня технічної підготовки. У багатьох сучасних рішеннях спостерігається тенденція до комбінування елементів обох підходів, створюючи гібридні системи, які поєднують надійність локального управління з гнучкістю та широкими можливостями мережевих технологій.

За типом комунікацій

-- Дротові системи "Розумний будинок" (Wired Systems)

Дротові системи "Розумний будинок" використовують фізичні кабельні з'єднання для передачі даних та керуючих сигналів між усіма компонентами системи. Це означає, що кожен пристрій, датчик та виконавець підключається до центрального контролера або один до одного за допомогою спеціальних кабелів. Для таких систем часто використовуються стандартизовані протоколи передачі даних, такі як KNX, Modbus або LonWorks, які забезпечують надійний та стабільний зв'язок.

Однією з головних переваг дротових систем є їхня висока надійність та стабільність передачі даних. Фізичне з'єднання практично виключає можливість втрати сигналу або перешкод, які можуть виникати у бездротових системах через радіочастотні перешкоди або велику кількість одночасно підключених пристроїв. Дротові системи також можуть забезпечувати вищу пропускну здатність, що є важливим для передачі великих обсягів даних, наприклад, від камер відеоспостереження високої роздільної здатності. Крім того, живлення пристроїв у дротових системах часто може здійснюватися по тих же кабелях, що й передача даних (Power over Ethernet, PoE), що спрощує встановлення та зменшує кількість необхідних розеток.

Однак, встановлення дротових систем зазвичай є більш складним та витратним, особливо якщо воно здійснюється в уже існуючому будинку. Прокладання кабелів може потребувати штроблення стін, що призводить до будівельних робіт та збільшує час і вартість проекту. Дротові системи також є менш гнучкими у плані розширення та модифікації. Додавання нових

пристроїв може вимагати прокладання нових кабелів, що може бути не завжди зручно або естетично прийнятно. Через ці особливості дротові системи частіше використовуються на етапі будівництва або капітального ремонту, коли є можливість закласти необхідну інфраструктуру.

- Бездротові системи "Розумний будинок" (Wireless Systems)

Бездротові системи "Розумний будинок" використовують різні радіотехнології для передачі даних та команд між пристроями. До найпоширеніших бездротових протоколів належать Wi-Fi, Bluetooth, Zigbee, Z-Wave та інші. Ці протоколи забезпечують зв'язок між пристроями та центральним контролером без необхідності фізичних кабельних з'єднань.

Основною перевагою бездротових систем є їхня гнучкість та простота встановлення. Додавання нових пристроїв зазвичай зводиться до їхнього підключення до існуючої бездротової мережі або сполучення з центральним контролером. Це робить бездротові системи ідеальними для модернізації вже існуючих будинків, де прокладання кабелів може бути складним або неможливим. Бездротові системи також пропонують більшу мобільність та зручність у переміщенні пристроїв у межах зони дії бездротової мережі.

Однак, бездротові системи можуть бути більш схильними до впливу радіочастотних перешкод від інших пристроїв або будівельних матеріалів, що може призводити до нестабільного зв'язку або втрати сигналу. Пропускна здатність бездротових мереж також може бути обмеженою, особливо при великій кількості одночасно підключених пристроїв. Крім того, енергоспоживання бездротових пристроїв може бути вищим, що потребує частішої заміни батарей або використання зовнішніх джерел живлення. Безпека бездротових мереж також є важливим аспектом, і користувачам необхідно вживати заходів для захисту своєї мережі від несанкціонованого доступу.

У сучасному світі спостерігається тенденція до використання комбінованих (гібридних) систем, які поєднують переваги як дротових, так і бездротових технологій. Наприклад, критично важливі елементи системи,

такі як центральний контролер та системи безпеки, можуть бути підключені дротовим способом для забезпечення максимальної надійності, тоді як менш критичні пристрої можуть використовувати бездротове з'єднання для гнучкості та зручності встановлення.

За функціональністю

-- Базові (локальні) системи "Розумний будинок" (Basic/Local Systems)

Базові, або локальні, системи "Розумний будинок" є найпростішим рівнем автоматизації та зазвичай орієнтовані на вирішення окремих, обмежених завдань у межах будинку. Такі системи часто складаються з незалежних пристроїв або невеликих груп пристроїв, які можуть працювати автономно або керуватися локально без складного центрального контролера чи інтеграції з іншими підсистемами. Прикладами базових систем можуть бути окремі комплекти розумного освітлення з власним хабом або без нього, розумні термостати, що працюють за заданим розкладом, або окремі камери відеоспостереження з локальним записом.

Основною характеристикою базових систем є їхня сфокусованість на конкретній функції. Вони можуть значно підвищити комфорт або безпеку в певній області, але не передбачають комплексного управління всіма інженерними системами будинку. Управління такими пристроями часто здійснюється за допомогою окремих мобільних додатків або локальних інтерфейсів, і можливість їхньої взаємодії з пристроями від інших виробників або іншими підсистемами може бути обмеженою або взагалі відсутньою.

Перевагами базових систем є їхня відносна простота встановлення та налаштування, нижча початкова вартість порівняно з інтегрованими рішеннями та можливість поступового впровадження автоматизації, починаючи з найважливіших для користувача функцій. Однак, їхнім головним недоліком є відсутність єдиної точки управління та обмежені можливості для створення складних сценаріїв автоматизації, які б об'єднували роботу різних пристроїв.

- Інтегровані системи "Розумний будинок" (Integrated Systems)

Інтегровані системи "Розумний будинок" являють собою комплексні рішення, які об'єднують управління багатьма інженерними системами та пристроями будинку в єдину екосистему. Такі системи зазвичай включають центральний контролер (хаб або сервер), який координує роботу всіх підключених пристроїв, забезпечує обмін даними між ними та надає користувачеві єдиний інтерфейс для управління. Інтегровані системи можуть контролювати освітлення, опалення, вентиляцію, кондиціонування, системи безпеки, мультимедійні пристрої, побутову техніку та багато іншого.

Основною характеристикою інтегрованих систем є їхня здатність до взаємодії між різними підсистемами та створення складних сценаріїв автоматизації, які реагують на різні події та умови. Наприклад, інтегрована система може автоматично регулювати температуру в залежності від часу доби та присутності мешканців, вмикати освітлення при відкритті дверей, активувати систему охорони при виході з дому та надсилати сповіщення на смартфон у разі небезпеки. Управління такою системою зазвичай здійснюється через єдиний мобільний додаток або веб-інтерфейс, що забезпечує зручність та централізований контроль.

Перевагами інтегрованих систем є їхні широкі функціональні можливості, гнучкість у налаштуванні складних сценаріїв автоматизації, можливість дистанційного керування всіма аспектами будинку та потенціал для значної економії енергії та підвищення рівня безпеки. Однак, їхнім недоліком є вища початкова вартість, складність встановлення та налаштування, а також залежність від надійності центрального контролера та сумісності різних пристроїв. При виборі інтегрованої системи важливим є врахування екосистеми та підтримки різних протоколів зв'язку для забезпечення сумісності з майбутніми розширеннями.

1.4 Архітектура системи розумного будинку

Архітектура системи "Розумний будинок" являє собою багаторівневу структуру, що забезпечує взаємодію між різними фізичними пристроями, мережевими технологіями та програмним забезпеченням для реалізації інтелектуального управління житловим простором. Розуміння цієї архітектури є ключовим для проектування, розробки та впровадження ефективних та масштабованих систем "розумний будинок". Традиційно, архітектуру такої системи можна розглядати на трьох основних рівнях: фізичний рівень, рівень передачі даних та логічний рівень.

Фізичний рівень (Physical Layer):

Фізичний рівень є найнижчим рівнем архітектури та включає всі фізичні пристрої, які безпосередньо взаємодіють з навколишнім середовищем та користувачем. До цього рівня належать різноманітні сенсори, виконавчі пристрої та контролери. Сенсори відповідають за збір інформації про стан будинку та його оточення, вимірюючи такі параметри як температура, вологість, освітленість, рух, відкриття/закриття дверей та вікон, рівень диму або газу. Виконавчі пристрої, навпаки, здійснюють фізичні дії на основі отриманих команд, наприклад, вмикають або вимикають світло, регулюють температуру опалення, відкривають або закривають замки, керують роботою побутової техніки.

Контролери на фізичному рівні можуть бути як простими (наприклад, мікроконтролери в окремих датчиках або виконавчих пристроях), так і більш складними (локальні контролери окремих підсистем). Вони відповідають за безпосереднє управління підключеними до них пристроями, збір даних від сенсорів та виконання базових команд. Важливою характеристикою пристроїв фізичного рівня є їхня здатність взаємодіяти з фізичним світом та перетворювати фізичні величини в електричні сигнали (для сенсорів) або навпаки (для виконавчих пристроїв).

Пристрої фізичного рівня можуть використовувати різні технології для зв'язку між собою та з вищими рівнями архітектури. Це можуть бути як дротові інтерфейси (наприклад, RS-485, Ethernet), так і бездротові технології (наприклад, Bluetooth Low Energy, Zigbee, Z-Wave). Вибір конкретної технології залежить від вимог до швидкості передачі даних, енергоспоживання, дальності зв'язку та вартості.

Рівень передачі даних (Data Link Layer):

Рівень передачі даних відповідає за надійну передачу інформації між пристроями фізичного рівня та логічним рівнем. На цьому рівні визначаються протоколи зв'язку, формати даних, механізми адресації та маршрутизації, які забезпечують коректну доставку даних між різними компонентами системи. Цей рівень абстрагує деталі фізичного рівня, надаючи вищим рівням стандартизований спосіб обміну даними.

На рівні передачі даних функціонують різні мережеві технології та протоколи, такі як Wi-Fi (IEEE 802.11), Ethernet (IEEE 802.3), Zigbee (IEEE 802.15.4), Z-Wave та інші. Кожен з цих протоколів має свої особливості щодо топології мережі (зірка, дерево, mesh), швидкості передачі даних, дальності зв'язку, енергоефективності та надійності. Вибір протоколу залежить від конкретних вимог системи "розумний будинок" та характеристик підключених пристроїв.

Центральні контролери (хаби, шлюзи) часто функціонують саме на рівні передачі даних, виступаючи посередниками між різними мережами та протоколами. Вони забезпечують перетворення протоколів, маршрутизацію даних та керування мережею "розумного будинку". На цьому рівні також можуть реалізовуватися базові механізми безпеки, такі як шифрування даних та аутентифікація пристроїв.

Логічний рівень (Logic Layer):

Логічний рівень є найвищим рівнем архітектури та відповідає за обробку даних, прийняття рішень, реалізацію інтелектуальних функцій та надання інтерфейсу користувача. На цьому рівні відбувається аналіз

інформації, отриманої з фізичного рівня через рівень передачі даних, визначаються правила та сценарії автоматизації, а також забезпечується взаємодія з користувачем через мобільні додатки, веб-інтерфейси або голосові помічники.

Основні компоненти логічного рівня включають центральний контролер (який може виконувати не лише функції рівня передачі даних, але й логічну обробку), хмарні сервіси (для зберігання даних, аналітики та віддаленого доступу), програмне забезпечення для управління та візуалізації даних, а також системи штучного інтелекту та машинного навчання для реалізації більш складних інтелектуальних функцій.

На логічному рівні реалізуються основні функції "розумного будинку", такі як автоматичне керування освітленням та опаленням на основі розкладів та даних від сенсорів, системи безпеки з розпізнаванням образів та оповіщеннями, голосове керування пристроями, створення персоналізованих сценаріїв для різних ситуацій, а також інтеграція з іншими онлайн-сервісами та екосистемами.

Інтерфейс користувача, що також належить до логічного рівня, є ключовим елементом для забезпечення зручної та інтуїтивно зрозумілої взаємодії з системою "розумний будинок". Він надає користувачеві можливість контролювати стан пристроїв, налаштовувати параметри роботи, створювати та редагувати сценарії автоматизації, а також отримувати інформацію про події, що відбуваються в будинку.

Таким чином, архітектура системи "розумний будинок" є складною ієрархічною структурою, де кожен рівень виконує свої специфічні функції, забезпечуючи безперебійну роботу та інтелектуальне управління житловим простором. Взаємодія між цими трьома рівнями – фізичним, рівнем передачі даних та логічним – є основою для реалізації всіх можливостей сучасної системи "розумний будинок".

Деякі архітектури побудовані на принципах розподіленої системи (наприклад, коли кожен мікроконтролер має свою логіку), інші — на централізованих рішеннях.

1.5 Огляд існуючих комерційних рішень

1.5.1 Google Nest

Система Google Nest являє собою комплексну екосистему пристроїв та сервісів, розроблену компанією Google для створення розумного будинку. В її основі лежить ідея інтеграції різних аспектів домашнього життя, забезпечуючи користувачам зручне управління, автоматизацію рутинних завдань, підвищення рівня безпеки та оптимізацію енергоспоживання. Екосистема Google Nest включає широкий спектр пристроїв, серед яких розумні колонки та дисплеї (Nest Mini, Nest Hub, Nest Hub Max, Nest Audio), термостати (Nest Thermostat), камери відеоспостереження (Nest Cam), дверні дзвінки (Nest Doorbell), а також пристрої для забезпечення бездротової мережі (Nest Wifi).

Центральним елементом управління екосистемою Google Nest виступає віртуальний помічник Google Assistant, інтегрований у більшість пристроїв Nest. Завдяки голосовим командам користувачі можуть керувати розумними пристроями, отримувати інформацію (про погоду, новини, тощо), встановлювати нагадування, відтворювати музику та подкасти, здійснювати дзвінки та багато іншого. Google Assistant також забезпечує персоналізацію взаємодії, розпізнаючи голоси різних членів сім'ї та надаючи їм відповідну інформацію та налаштування.

Для управління та налаштування пристроїв Google Nest використовується мобільний додаток Google Home, доступний для Android та iOS. Цей додаток служить єдиною платформою для підключення нових пристроїв, створення сценаріїв автоматизації (так званих "Routine"),

керування окремими пристроями, перегляду відео з камер спостереження в режимі реального часу та перегляду історії подій. Додаток Google Home також дозволяє інтегрувати пристрої Nest з пристроями розумного будинку від інших виробників, які підтримують екосистему Google.

Однією з ключових переваг системи Google Nest є її інтеграція з іншими сервісами Google, такими як Google Calendar, Google Maps, YouTube Music та іншими. Це забезпечує безшовну взаємодію між різними аспектами цифрового життя користувача та його розумним будинком. Наприклад, користувач може попросити Google Assistant додати подію до календаря, дізнатися про свій маршрут на роботу або відтворити улюблену музику на розумній колонці.

Безпека є одним з пріоритетних напрямків розвитку Google Nest. Камери та дверні дзвінки Nest оснащені функціями виявлення руху, розпізнавання облич (за наявності підписки Nest Aware), двостороннім аудіозв'язком та шифруванням відеоданих. Термостати Nest допомагають економити енергію за рахунок автоматичного регулювання температури на основі розкладів та присутності людей в будинку.

Система Google Nest постійно розвивається та отримує нові функції завдяки оновленням програмного забезпечення. Google активно працює над розширенням екосистеми, додаючи підтримку нових пристроїв та інтеграцій. Завдяки своїй зручності, широкому спектру пристроїв та інтеграції з популярними сервісами, Google Nest є однією з провідних платформ на ринку розумних будинків, пропонуючи користувачам комплексне та інтуїтивно зрозуміле рішення для автоматизації їхнього житла.

Важливою особливістю Google Nest є підтримка стандарту Matter, який спрямований на забезпечення сумісності між пристроями розумного будинку від різних виробників. Це спрощує інтеграцію нових пристроїв в екосистему Nest та підвищує гнучкість при виборі обладнання для розумного будинку.

Таким чином, Google Nest являє собою потужну та зручну екосистему для створення розумного будинку, яка поєднує в собі якісні пристрої,

інтелектуального віртуального помічника та зручне програмне забезпечення для управління та автоматизації різних аспектів життя в сучасному домі.

1.5.2. Amazon Alexa + SmartThings

Поєднання Amazon Alexa та Samsung SmartThings являє собою потужну та універсальну екосистему для побудови всеосяжного розумного будинку. Amazon Alexa, хмарний голосовий сервіс, виступає в ролі інтелектуального голосового помічника, дозволяючи користувачам керувати широким спектром розумних пристроїв за допомогою голосових команд. SmartThings, зі свого боку, є платформою розумного будинку, яка дозволяє користувачам підключати та керувати різними сумісними пристроями від різних виробників, створюючи єдину систему розумного будинку.

Інтеграція між Alexa та SmartThings забезпечує безперешкодне голосове керування пристроями, підключеними до хабу SmartThings. Користувачі можуть попросити Alexa увімкнути світло, підключене до SmartThings, відрегулювати термостат, керований SmartThings, заблокувати розумні двері, якими керує SmartThings, та багато іншого. Ця інтеграція спрощує користувацький досвід, надаючи єдиний голосовий інтерфейс для керування різноманітними пристроями розумного будинку, усуваючи необхідність взаємодії з кількома додатками чи інтерфейсами.

SmartThings діє як центральний хаб, підтримуючи різні протоколи зв'язку, такі як Zigbee, Z-Wave та Wi-Fi, що дозволяє йому підключатися до широкого спектру розумних пристроїв, включаючи ті, які безпосередньо не сумісні з Alexa. Підключивши навичку SmartThings у додатку Alexa, користувачі надають Alexa дозвіл на виявлення та керування пристроями, підключеними до їхнього облікового запису SmartThings. Це створює надійну та розширювану екосистему розумного будинку, де користувачі можуть вибирати з величезного асортименту сумісних пристроїв та керувати ними зручно за допомогою голосу.

Окрім базового голосового керування, інтеграція Alexa та SmartThings дозволяє створювати складні автоматизації та сценарії розумного будинку. Користувачі можуть налаштовувати сценарії в додатку SmartThings, які запускають кілька дій на різних пристроях, а потім ініціювати ці сценарії однією голосовою командою до Alexa. Наприклад, сценарій "Доброго ранку" може увімкнути світло, відрегулювати термостат та почати відтворення музики, і все це запускається простою фразою: "Alexa, доброго ранку".

Крім того, Alexa може надавати голосовий зворотний зв'язок про стан пристроїв SmartThings, наприклад, чи зачинені двері, чи виявлено рух датчиком, підключеним до SmartThings. Це додає додатковий рівень зручності та обізнаності до досвіду розумного будинку. Об'єднана екосистема пропонує високий ступінь гнучкості та налаштування, дозволяючи користувачам адаптувати свій розумний будинок до своїх конкретних потреб та вподобань.

По суті, синергія між Amazon Alexa та Samsung SmartThings створює потужне рішення для розумного будинку, яке поєднує зручність голосового керування з широкою сумісністю пристроїв та можливостями автоматизації спеціалізованої платформи розумного будинку. Ця інтеграція дає користувачам змогу створити справді інтелектуальне та чутливе домашнє середовище, яким можна легко керувати за допомогою голосу.

1.5.3. Xiaomi Smart Home

Xiaomi Smart Home являє собою розгалужену та швидкозростаючу екосистему пристроїв для розумного будинку, розроблену китайською технологічною компанією Xiaomi. Ця екосистема охоплює широкий спектр продуктів, включаючи розумне освітлення (лампочки, світильники), датчики (руху, відкриття, температури, вологості), камери спостереження, розумні розетки, пристрої для керування кліматом (кондиціонери, обігрівачі), побутову техніку (пилососи, мультиварки) та багато інших пристроїв,

спрямованих на автоматизацію та інтелектуальне управління житловим простором.

Ключовим елементом екосистеми Xiaomi Smart Home є центральний хаб, такий як Xiaomi Mi Smart Hub або Aqara Hub (оскільки Xiaomi активно співпрацює з брендом Aqara, що спеціалізується на розумних датчиках та пристроях). Ці хаби служать для підключення та координації роботи різних розумних пристроїв за допомогою бездротових протоколів, таких як Wi-Fi, Zigbee та Bluetooth. Використання протоколу Zigbee особливо важливе для багатьох датчиків Xiaomi, оскільки він забезпечує низьке енергоспоживання та надійний зв'язок у великих мережах.

Управління всіма пристроями екосистеми Xiaomi Smart Home здійснюється через єдиний мобільний додаток Mi Home (також відомий як Xiaomi Home). Цей додаток надає користувачам інтуїтивно зрозумілий інтерфейс для додавання нових пристроїв, керування їхніми функціями в реальному часі, налаштування розкладів роботи та створення різноманітних сценаріїв автоматизації. Додаток також дозволяє отримувати сповіщення від датчиків та камер спостереження, забезпечуючи таким чином підвищений рівень безпеки та контролю над будинком.

Однією з головних переваг Xiaomi Smart Home є її доступність. Компанія пропонує широкий асортимент розумних пристроїв за конкурентними цінами, що робить технології розумного будинку більш досяжними для широкого кола користувачів. При цьому якість та функціональність багатьох пристроїв залишаються на високому рівні. Екосистема постійно розширюється новими продуктами, що дозволяє користувачам поступово автоматизувати різні аспекти свого житла.

Хоча основне управління екосистемою Xiaomi Smart Home здійснюється через власний додаток, багато пристроїв також підтримують інтеграцію з популярними голосовими помічниками, такими як Google Assistant та Amazon Alexa. Це дозволяє користувачам керувати своїми

розумними пристроями Xiaomi за допомогою голосових команд, що додає ще більшої зручності у повсякденному використанні.

Загалом, Xiaomi Smart Home є привабливою екосистемою для тих, хто прагне створити розумний будинок без значних фінансових витрат, отримуючи при цьому широкий вибір функціональних та надійних пристроїв з централізованим управлінням та можливостями автоматизації. Її постійний розвиток та розширення асортименту роблять її одним з ключових гравців на ринку розумних будинків.

1.5.4. Ajax Systems (Україна)

Система Ajax Systems, розроблена в Україні, є комплексною екосистемою бездротової та дротової безпеки, яка також включає елементи розумного будинку та автоматизації. Основна мета системи – забезпечення надійного захисту від вторгнень, пожеж та затоплень, а також надання можливостей для керування комфортом та автоматизації побутових процесів. Ajax відрізняється високим рівнем безпеки, надійністю зв'язку та зручним користувацьким інтерфейсом.

Центральним елементом системи є хаб (централь), який координує роботу всіх підключених пристроїв за допомогою власного захищеного радіопротоколу Jeweller. Цей протокол забезпечує двосторонній зв'язок на відстані до 2000 метрів на відкритому просторі, шифрування даних та захист від глушіння та підміни сигналу. Для передачі фотопідтверджень тривог використовується окремий високошвидкісний радіопротокол Wings.

Екосистема Ajax включає широкий спектр датчиків (руху, відчинення дверей/вікон, розбиття скла, диму, чадного газу, затоплення), сирен, клавіатур, брелоків, ретрансляторів сигналу, а також пристроїв автоматизації, таких як розумні розетки (Socket), реле (Relay) та настінні вимикачі (WallSwitch). Ці пристрої дозволяють реалізувати сценарії розумного

будинку, наприклад, автоматичне ввімкнення світла за розкладом або при тривозі, керування електроприладами дистанційно або за заданими умовами.

Управління системою Ajax здійснюється через зручний мобільний додаток для iOS та Android, який дозволяє користувачам контролювати стан системи, отримувати сповіщення про тривоги, переглядати фотопідтвердження, керувати пристроями автоматизації та налаштовувати сценарії. Додаток також надає можливість підключення до охоронних компаній для професійного моніторингу та реагування на тривоги.

Однією з ключових переваг Ajax є його надійність та професійний рівень безпеки, що підтверджується відповідними сертифікатами. Система розроблена з урахуванням захисту від кібератак та саботажу, має резервне живлення та кілька каналів зв'язку (Ethernet, Wi-Fi, GSM/LTE) для забезпечення безперебійної роботи навіть у разі відключення електроенергії або проблем зі зв'язком.

Хоча основний фокус Ajax спрямований на безпеку, система також активно розвиває функціональність розумного будинку. Завдяки сценаріям та пристроям автоматизації користувачі можуть створювати комфортне та інтелектуальне середовище проживання. Інтеграції зі сторонніми системами, такими як системи відеоспостереження та інші платформи розумного будинку, також розширюють можливості використання Ajax.

Таким чином, Ajax Systems є високотехнологічною українською розробкою, яка поєднує в собі професійну безпеку та зручні функції розумного будинку, пропонуючи користувачам надійне та багатофункціональне рішення для захисту та автоматизації їхнього житла. Її бездротова архітектура спрощує встановлення та розширення системи, роблячи її привабливим вибором для широкого кола користувачів.

1.6 Проблеми впровадження розумного будинку

Впровадження систем "Розумний будинок", незважаючи на їхні численні переваги, стикається з низкою значних проблем, які стримують їхнє масове поширення та повсюдне використання. Ці проблеми охоплюють технічні, економічні, соціальні та етичні аспекти, вимагаючи комплексного підходу для їхнього вирішення.

Однією з ключових проблем є фрагментація ринку та відсутність єдиних стандартів. На ринку представлено велику кількість пристроїв та платформ від різних виробників, які часто використовують власні пропрієтарні протоколи зв'язку та екосистеми. Це ускладнює інтеграцію пристроїв різних брендів в єдину систему, обмежуючи гнучкість користувачів при виборі обладнання та створюючи залежність від конкретного виробника. Відсутність універсальних стандартів також ускладнює розробку сумісного програмного забезпечення та ускладнює взаємодію між різними підсистемами розумного будинку.

Іншою серйозною проблемою є безпека та конфіденційність даних. Оскільки системи "Розумний будинок" збирають та обробляють велику кількість особистої інформації про звички, розпорядок дня та навіть фінансові транзакції користувачів, ризики несанкціонованого доступу, кібератак та витоку цих даних є досить високими. Недостатній рівень захисту може призвести до серйозних наслідків, включаючи крадіжку особистої інформації, несанкціоноване керування пристроями та навіть загрозу фізичній безпеці мешканців. Забезпечення надійного шифрування, аутентифікації та захисту від вразливостей є критично важливим для завоювання довіри користувачів.

Складність встановлення, налаштування та використання також є значною перешкодою для широкого впровадження розумних будинків. Багато існуючих систем вимагають спеціальних технічних знань та навичок для встановлення та налаштування, що відлякує багатьох потенційних

користувачів. Неінтуїтивні інтерфейси користувача та складна процедура інтеграції нових пристроїв можуть призвести до розчарування та відмови від використання розумних технологій. Спрощення процесу встановлення, розробка дружніх інтерфейсів та забезпечення якісної технічної підтримки є важливими кроками для подолання цієї проблеми.

Висока вартість впровадження комплексних рішень залишається суттєвим бар'єром для багатьох домогосподарств. Повноцінні системи "Розумний будинок", що включають широкий спектр функціональних можливостей, можуть бути досить дорогими як на етапі придбання обладнання, так і в процесі встановлення та обслуговування. Зниження вартості компонентів, розробка більш доступних рішень та демонстрація економічної вигоди від використання розумних технологій (наприклад, за рахунок економії енергії) можуть сприяти зростанню їхньої популярності.

Проблема сумісності та інтероперабельності не обмежується лише різними виробниками. Навіть в рамках екосистеми одного виробника можуть виникати проблеми з сумісністю між пристроями різних поколінь або лінійок продуктів. Забезпечення плавної та безперебійної взаємодії між усіма компонентами системи є важливим для створення цілісного та зручного користувацького досвіду.

Існує також залежність від стабільного інтернет-з'єднання. Багато функцій розумного будинку, особливо дистанційне керування та інтеграція з хмарними сервісами, вимагають постійного підключення до Інтернету. Перебої у роботі мережі можуть призвести до втрати контролю над пристроями та порушення функціональності системи. Розробка рішень, які можуть частково функціонувати в автономному режимі, є важливим напрямком розвитку.

Крім того, існують етичні та соціальні питання, пов'язані з впровадженням розумних будинків. Збір великої кількості даних про приватне життя користувачів викликає занепокоєння щодо їхнього використання та

потенційної можливості зловживань. Необхідно розробляти чіткі правові та етичні норми, що регулюють збір, зберігання та використання цих даних.

Нарешті, психологічний аспект та готовність користувачів до нових технологій також відіграють важливу роль. Деякі люди можуть відчувати дискомфорт або недовіру до систем, які автоматично керують їхнім житлом, побоюючись втрати контролю або складних технічних проблем. Проведення освітньої роботи, демонстрація переваг та простоти використання розумних технологій, а також забезпечення якісної підтримки користувачів можуть допомогти подолати ці психологічні бар'єри.

Вирішення цих проблем вимагає спільних зусиль виробників, розробників, регуляторних органів та наукової спільноти. Подолання технічних бар'єрів, забезпечення безпеки та конфіденційності, зниження вартості, спрощення використання та підвищення довіри користувачів є ключовими факторами для успішного та широкого впровадження систем "Розумний будинок" у майбутньому.

1.7 Перспективи розвитку технологій Smart Home

Технології Smart Home стрімко розвиваються, перетворюючи наші оселі на інтелектуальні простори, здатні адаптуватися до наших потреб та підвищувати рівень комфорту, безпеки та енергоефективності. Майбутнє цієї галузі обіцяє ще більш значні трансформації, зумовлені прогресом у сфері штучного інтелекту (ШІ), Інтернету речей (IoT), 5G, сенсорних технологій та енергоефективності. Розглянемо ключові перспективи розвитку технологій Smart Home на найближчі роки.

Однією з найважливіших тенденцій є поглиблена інтеграція штучного інтелекту та машинного навчання. Системи Smart Home ставатимуть дедалі розумнішими, здатними аналізувати поведінку користувачів, передбачати їхні потреби та автоматично адаптувати роботу пристроїв. ШІ дозволить реалізувати більш складні сценарії автоматизації, персоналізовані

рекомендації та проактивне управління будинком. Наприклад, система зможе самостійно оптимізувати енергоспоживання на основі погодних умов та звичок мешканців, регулювати освітлення та температуру в залежності від їхньої присутності та настрою, а також виявляти потенційні проблеми (наприклад, витік води або задимлення) на ранніх стадіях.

Подальший розвиток та розширення екосистем IoT відіграватиме ключову роль у майбутньому Smart Home. Кількість підключених пристроїв продовжуватиме зростати, охоплюючи практично всі аспекти домашнього життя – від побутової техніки та систем безпеки до меблів та навіть рослин. Важливим аспектом стане забезпечення безперебійної та безпечної взаємодії між цими різноманітними пристроями, незалежно від їхнього виробника та використовуваних протоколів зв'язку.

Очікується значний прогрес у сфері інтероперабельності та стандартизації протоколів зв'язку. Розробка та впровадження універсальних стандартів, таких як Matter, покликані вирішити проблему фрагментації ринку та забезпечити безпроблемну інтеграцію пристроїв різних брендів в єдину екосистему. Це зробить процес вибору та налаштування розумного будинку більш простим та зручним для користувачів.

Розвиток мереж 5G відкриє нові можливості для систем Smart Home, забезпечуючи більш високу швидкість передачі даних, низьку затримку та більшу пропускну здатність. Це сприятиме покращенню роботи пристроїв, що потребують швидкого та стабільного з'єднання, таких як камери відеоспостереження високої роздільної здатності та системи розширеної реальності (AR) для дому.

Удосконалення сенсорних технологій призведе до появи більш точних, компактних та енергоефективних датчиків, здатних збирати ширший спектр даних про стан будинку та його мешканців. Це дозволить системам Smart Home краще розуміти контекст та реагувати на події більш адекватно. З'являться нові типи сенсорів, наприклад, для моніторингу якості повітря, виявлення емоцій за виразом обличчя або аналізу сну.

Енергоефективність та сталий розвиток ставатимуть все більш важливими аспектами технологій Smart Home. Системи майбутнього будуть спрямовані на оптимізацію споживання енергії, управління відновлюваними джерелами енергії (наприклад, сонячними панелями) та сприяння більш екологічному способу життя. Розумні термостати, системи управління освітленням та пристрої моніторингу енергоспоживання допоможуть користувачам зменшити свій вуглецевий слід та заощадити кошти.

Зростання уваги до безпеки та конфіденційності даних стане ключовим фактором розвитку галузі. Виробники будуть впроваджувати більш надійні механізми захисту від кібератак, шифрування даних та надавати користувачам більший контроль над їхньою особистою інформацією. Прозорість у зборі та використанні даних стане обов'язковою умовою для завоювання довіри споживачів.

Персоналізація та адаптивність стануть визначальними характеристиками майбутніх систем Smart Home. Системи будуть навчатися на основі індивідуальних потреб та звичок кожного мешканця, пропонуючи персоналізовані сценарії автоматизації, рекомендації та налаштування. Інтерфейси користувача ставатимуть більш інтуїтивно зрозумілими та адаптованими до різних вікових груп та рівнів технічної підготовки.

Розширення можливостей голосового керування та інтеграція з іншими формами взаємодії (жести, погляд) зроблять керування розумним будинком ще більш природним та зручним. Голосові помічники стануть більш інтелектуальними та контекстуально обізнаними, здатними розуміти складні команди та вести діалог з користувачем.

Нарешті, зниження вартості компонентів та спрощення встановлення сприятимуть масовому впровадженню технологій Smart Home. З розвитком виробничих процесів та появою більш доступних рішень розумний будинок перестане бути розкішшю та стане невід'ємною частиною сучасного житла для більшості людей.

У підсумку, перспективи розвитку технологій Smart Home є надзвичайно багатообіцяючими. Майбутнє розумних будинків – це інтелектуальні, безпечні, енергоефективні та персоналізовані простори, які органічно інтегруються в наше повсякденне життя, роблячи його комфортнішим, зручнішим та безпечнішим.

2 ПРОЕКТУВАННЯ СИСТЕМИ РОЗУМНОГО БУДИНКУ

2.1 Опис обєкта, та загальні вимоги до системи

В якості обїєкта проектуваня системи розумного будинку будимо використовувати типовий проект одноповерхового будинку (рис. 2.1)

Рисунок 2.1 – План будинку для якого розробляється система

В будинку знаходяться 3 спальні, вітальня з кухнею, санітарно-битові та допоміжні приміщення, гараж.

Для забезпечення вхідних даних до системи розумного будинку необхідно визначити функції необхідні для реалізації. В цілому усі функції

можна виділіти відповідно до трьох груп задач – задачі запезпечення комфорту, задачі безпеки та задачі контролю за споживанням ресурсів.

Розглянемо сенсори, що забезпечують виконання цих функцій.

Сенсори комфорту:

- Датчики освітленості (фотоелементи). Вимірюють рівень природного освітлення. Можуть використовуватися для автоматичного регулювання яскравості штучного світла або для керування зовнішнім освітленням (ввімкнення вночі, вимкнення вдень).
- Датчики температури. Вимірюють температуру повітря в приміщенні або на вулиці. Використовуються для керування системами опалення, вентиляції та кондиціонування, а також для відображення поточної температури.
- Датчики вологості. Вимірюють відносну вологість повітря. Важливі для керування вентиляцією, запобігання утворенню конденсату та підтримання комфортного мікроклімату.
- Датчики якості повітря (VOC, CO2, PM2.5). Вимірюють рівень летких органічних сполук, вуглекислого газу та дрібних твердих частинок у повітрі. Можуть використовуватися для автоматичного ввімкнення вентиляції або очищувачів повітря при погіршенні якості повітря.

Сенсори безпеки:

- Датчики руху (інфрачервоні, ультразвукові). Виявляють рух у приміщенні або на прилеглій території. Можуть використовуватися для автоматичного ввімкнення/вимкнення світла в коридорах, ванних кімнатах, гардеробних, гаражі, а також для охоронної сигналізації.
- Датчики присутності. Більш чутливі до статичної присутності людини, ніж датчики руху. Можуть використовуватися в кімнатах для підтримки освітлення, поки там знаходяться люди .

- Датчики відчинення дверей та вікон (геркони). Реагують на відкриття або закриття дверей та вікон. Використовуються для охоронної сигналізації та контролю доступу.
- Датчики розбиття скла. Реагують на звук розбиття скла. Можуть встановлюватися на вікна та скляні двері.
- Датчики диму. Виявляють задимлення в приміщенні та подають сигнал тривоги. Обов'язкові для забезпечення пожежної безпеки.
- Датчики чадного газу (CO). Виявляють небезпечний рівень чадного газу. Особливо важливі, якщо в будинку є газові прилади або камін.
- Датчики протікання води. Виявляють витікання води в санвузлах, кухні, котельні. Допомагають запобігти затопленню та пошкодженню майна.
- Камери відеоспостереження. Передають відеозображення в режимі реального часу. Можуть використовуватися для моніторингу внутрішньої та зовнішньої території, запису подій та віддаленого перегляду.
- Датчики вібрації. Можуть використовуватися для виявлення спроб злому або незвичайних вібрацій.

Інші сенсори:

- Датчики енергоспоживання. Вимірюють споживання електроенергії окремими приладами або всім будинком. Допомагають контролювати та оптимізувати енергоспоживання.
- Датчики расходу газу та води. Вимірюють споживання газу та води окремими приладами або всім будинком. Допомагають контролювати та оптимізувати споживання газу та води.

Як правило, в залежності від призначення приміщень сенсори встановлюються відповідним чином:

- Спальні кімнати – датчики температури, вологості, якості повітря (за бажанням), датчики відчинення вікон (для безпеки), датчики руху (для нічного освітлення або безпеки).

- Кухня – датчики температури, вологості, якості повітря, датчик диму, датчик протікання води (біля мийки та посудомийної машини).
- Вітальня – датчики температури, вологості, якості повітря, датчики руху (для освітлення або безпеки).
- Ванні кімнати – датчики температури, вологості, датчик протікання води.
- Коридори та передпокій – датчики руху (для освітлення та безпеки).
- Гараж – датчик відкриття/закриття воріт, датчик руху, датчик CO (за необхідності).
- Зовнішня територія – датчики руху (для освітлення та безпеки), камери відеоспостереження, датчики освітленості (для керування зовнішнім освітленням).

Розглянувши вимоги до побудови сенсорної системи розумного будинку встановимо необхідні сенсори, як наведено на рис (2.2) .

Усього необхідно встановити 11 датчиків відкриття вікон та дверей, 7 датчиків руху, 5 датчиків диму, 4 датчика температури та вологості (DHT), 3 датчика протикання води, по 1 датчику споживання електричного струму, води та газу, забезпечити управління 7 освітлювальними, 4 кондиціонерами у спальнях та вітальні та 2 системи вентиляції.

Рисунок 2.2 – Разтошування сенсорів на плані будинку

2.2 Вибір системи управління

Розвиток технологій призвів до появи різноманітних систем управління, кожна з яких має свої особливості, переваги та недоліки. У контексті автоматизації та керування різними процесами, зокрема в розумному будинку, ключову роль відіграють системи управління. Розглянемо три основні типи таких систем: системи управління розумним будинком (спеціалізовані), комп'ютери (загального призначення) та мікроконтролери (вбудовані системи).

Спеціалізовані системи управління розумним будинком (СУРБ)

Це спеціалізовані апаратні та програмні комплекси, розроблені виключно для автоматизації та керування функціями житлового будинку. Вони інтегрують різноманітні датчики, виконавчі пристрої та комунікаційні протоколи для забезпечення комфорту, безпеки та енергоефективності.

До головних переваг СУРБ можна віднести простоту використання та налаштування. Інтерфейси СУРБ зазвичай інтуїтивно зрозумілі та орієнтовані на кінцевого користувача без глибоких технічних знань. Налаштування часто зводиться до підключення пристроїв та вибору готових сценаріїв. СУРБ мають вбудовану підтримку стандартних протоколів розумного будинку (наприклад, Zigbee, Z-Wave, KNX) та інтеграцію з широким спектром сумісних пристроїв (освітлення, термостати, замки, камери тощо). Оскільки СУРБ розробляються для конкретних завдань, вони часто демонструють високу стабільність роботи та меншу схильність до збоїв порівняно з системами загального призначення. Багато СУРБ оптимізовані для низького енергоспоживання, що є важливим для постійно працюючих систем. СУРБ забезпечують єдину точку керування всіма інтегрованими пристроями через зручний інтерфейс (сенсорні панелі, мобільні додатки). Спеціалізовані протоколи та вбудовані механізми безпеки часто забезпечують вищий рівень захисту даних та керування пристроями.

Однак **с**пеціалізовані рішення часто мають вищу початкову вартість порівняно з використанням існуючого комп'ютера або самостійно зібраної системи на мікроконтролері. Хоча екосистема розумного будинку постійно розширюється, інтеграція нестандартних або саморобних пристроїв може бути складною або неможливою. Функціональність та підтримка СУРБ часто залежать від виробника, що може призвести до проблем при бажанні змінити платформу або інтегрувати пристрої інших виробників. Порівняно з комп'ютерами, СУРБ можуть мати меншу обчислювальну потужність та обсяг пам'яті, що обмежує складність сценаріїв та аналіз даних.

Комп'ютер (ПК, одноплатні комп'ютери)

Використання комп'ютера як системи управління розумним будинком надає значну гнучкість та обчислювальну потужність, але вимагає відповідних знань та налаштувань.

До переваги комп'ютера, як системи управління можна віднести - високу обчислювальну потужність та обсяг пам'яті. Комп'ютери здатні обробляти складні алгоритми, аналізувати великі обсяги даних, запускати різноманітне програмне забезпечення. Можливість встановлення різноманітних операційних систем, програмного забезпечення та підключення широкого спектру периферійних пристроїв (USB-адаптери для різних протоколів, плати розширення). Завдяки гнучкості програмного забезпечення та наявності різних інтерфейсів, комп'ютер може бути інтегрований з саморобними або менш поширеними пристроями. Потужні графічні інтерфейси, можливість створення власних дашбордів та систем візуалізації. Якщо у користувача вже є комп'ютер, витрати можуть обмежитися придбанням необхідних адаптерів та програмного забезпечення. Особливо це стосується одноплатних комп'ютерів (наприклад, Raspberry Pi).

Однак ці системи мають певні недоліки. Налаштування операційної системи, встановлення необхідного програмного забезпечення, налагодження комунікації з пристроями розумного будинку вимагає певних технічних знань. Комп'ютери, як правило, споживають значно більше електроенергії порівняно з СУРБ та мікроконтролерами, що може призвести до вищих експлуатаційних витрат. Операційні системи загального призначення можуть бути менш стабільними та схильними до збоїв порівняно зі спеціалізованими системами. Для реалізації функцій розумного будинку необхідно встановлювати та налаштовувати відповідне програмне забезпечення (наприклад, Home Assistant, OpenHAB). Інтерфейс користувача може бути менш інтуїтивно зрозумілим для непідготовленого користувача. Комп'ютери, підключені до мережі, можуть бути більш вразливими до кібератак, якщо не вжити належних заходів безпеки.

Мікроконтролер (Arduino, ESP32 та ін.)

Мікроконтролери є невеликими, енергоефективними обчислювальними пристроями, призначеними для виконання конкретних завдань у вбудованих системах. Вони часто використовуються для створення кастомних рішень для розумного будинку.

Мікроконтролери мають ряд перваг перед іншими варіантами. Мікроконтролери є відносно недорогими, особливо у великих кількостях. Мікроконтролери споживають дуже мало електроенергії, що робить їх ідеальними для живлення від батарей або для постійно працюючих пристроїв. Мікроконтролери мають вбудовані цифрові та аналогові виводи, що спрощує підключення датчиків та виконавчих пристроїв. Можливість програмування мікроконтролерів на низькому рівні або з використанням спеціалізованих бібліотек для реалізації конкретних функцій. Компактні розміри дозволяють інтегрувати мікроконтролери в невеликі пристрої. Існує велика кількість документації, прикладів коду та готових бібліотек для різних завдань.

Однак, побудова системи управління на мікроконтролері має й певні недоліки. Мікроконтролери мають значно меншу обчислювальну потужність та обсяг пам'яті порівняно з комп'ютерами та СУРБ, що обмежує складність завдань, які вони можуть виконувати. Програмування мікроконтролерів може вимагати знання мов програмування (наприклад, C/C++, Python) та розуміння принципів роботи з апаратним забезпеченням. Для керування пристроями на базі мікроконтролерів часто необхідно розробляти власний інтерфейс (веб-сторінки, мобільні додатки) або використовувати зовнішні системи управління. Підтримка різних протоколів розумного будинку може вимагати використання додаткових апаратних модулів та написання відповідного програмного забезпечення. Керування великою кількістю складних пристроїв може бути складним завданням для одного мікроконтролера.

В цілому вибір системи управління розумним будинком залежить від конкретних потреб, бюджету, технічних знань користувача та бажаного рівня функціональності.

- СУРБ є найкращим варіантом для користувачів, які шукають просте у використанні, надійне та спеціалізоване рішення з готовою підтримкою широкого спектру пристроїв розумного будинку. Однак, вони можуть бути дорожчими та менш гнучкими.
- Комп'ютери надають високу гнучкість та обчислювальну потужність, що дозволяє реалізовувати складні сценарії та інтегрувати нестандартні пристрої. Однак, вони вимагають значних технічних знань для налаштування та підтримки, а також мають вище енергоспоживання.
- Мікроконтролери є ідеальним вибором для ентузіастів та розробників, які хочуть створювати кастомні та енергоефективні рішення для розумного будинку. Вони є недорогими та гнучкими, але вимагають значних зусиль у програмуванні та інтеграції.

У багатьох випадках оптимальним рішенням може бути комбінація різних підходів. Наприклад, централізована СУРБ може використовуватися для керування основними функціями, а окремі мікроконтролери можуть відповідати за специфічні кастомні пристрої або датчики. Розуміння переваг та недоліків кожної системи є ключем до вибору найбільш підходящого рішення для розумного будинку.

Тому, що в завданні на дипломне проектування вказано створити систему управління розумним будинком на мікроконтролері, то реалізуємо її наступним чином – для сбору інформації з сенсорів та керування виконавчіми пристроями будемо використовувати мікроконтролер на платі Arduino[7], для обміну інформацією з інтерфейсом користувача (він реалізується в іншій кваліфікаційній роботі) використовуємо ESP8266[8].

Комбінація Arduino та ESP8266 є надзвичайно популярним та ефективним рішенням для створення систем розумного будинку, особливо для ентузіастів та невеликих проектів. Цей вибір обґрунтовується синергією

переваг обох платформ, які взаємно доповнюють одна одну, забезпечуючи потужну та гнучку основу для автоматизації житла.

Arduino славиться своєю простотою підключення датчиків та виконавчих пристроїв завдяки зручним пінам та великій кількості доступних бібліотек. Arduino IDE з його спрощеною мовою програмування C/C++ робить процес розробки доступним навіть для початківців. Це дозволяє легко інтегрувати різноманітні сенсори та актуатори, необхідні для розумного будинку (датчики температури, вологості, руху, реле для керування освітленням тощо).

Велика та активна спільнота Arduino надає безліч готових рішень, прикладів коду та відповідей на поширені запитання. Це значно спрощує процес розробки та налагодження системи розумного будинку, особливо при виникненні проблем.

Різноманітність плат Arduino дозволяє вибрати оптимальний варіант для конкретних потреб проекту розумного будинку, враховуючи кількість необхідних пінів, розміри та інші характеристики. Багато плат Arduino, особливо призначені для роботи від батарей, мають низьке енергоспоживання, що є важливим для створення бездротових датчиків та пристроїв розумного будинку, які можуть працювати автономно.

Ключовою перевагою ESP8266 є наявність вбудованого Wi-Fi модуля. Це дозволяє пристроям розумного будинку легко підключатися до домашньої мережі та обмінюватися даними з іншими пристроями, серверами або хмарними платформами без необхідності використання додаткових дорогих Ethernet-шилдів або інших Wi-Fi модулів.

ESP8266 є надзвичайно економічним рішенням, особливо враховуючи наявність вбудованого Wi-Fi. Це робить його ідеальним для створення великої кількості підключених пристроїв розумного будинку без значних фінансових витрат.

ESP8266 має достатню обчислювальну потужність та обсяг пам'яті для виконання багатьох завдань розумного будинку, включаючи збір даних з

датчиків, керування виконавчими пристроями, обробку простих алгоритмів та обмін даними через Wi-Fi.

ESP8266 може бути легко запрограмований за допомогою знайомого середовища Arduino IDE після встановлення відповідних розширень. Це дозволяє використовувати існуючі знання та бібліотеки Arduino для розробки пристроїв на базі ESP8266.

ESP8266 підтримує різні мережеві протоколи, такі як TCP/IP, HTTP, MQTT, що робить його сумісним з багатьма платформами розумного будинку та хмарними сервісами.

Комбінуючи Arduino з ESP8266, ми отримуємо систему, яка використовує найкращі сторони обох платформ:

- Arduino чудово справляється з інтерфейсом з фізичним світом, забезпечуючи просте підключення та керування різноманітними датчиками та виконавчими пристроями.
- ESP8266 забезпечує надійне та недороге підключення до мережі Wi-Fi, дозволяючи пристроям розумного будинку обмінюватися даними та керуватися віддалено.

Вибір комбінації Arduino та ESP8266 є обґрунтованим рішенням для нашого проектурозумного будинку завдяки їхній низькій вартості, простоті використання, наявності вбудованого Wi-Fi, великій спільноті та гнучкості в інтеграції з різноманітними датчиками та хмарними платформами. Ця комбінація дозволяє швидко та ефективно створювати функціональні та підключені пристрої розумного будинку. Хоча для більш складних завдань може знадобитися потужніша обчислювальна платформа, для типових завдань нашого розумного будинку Arduino + ESP8266 є оптимальним вибором.

2.3 Побудова системи розумного будинку

Враховуючі обмеження кількості портів GPIO у контролера, мінімізації відстані від датчиків до контролера, доцільно розбити систему розумного будинку на 3 окремі зони, які будуту передавати дані та отримати сігнили управління з загального сервера (рис 2.3). Кожна зона обслуговується системою управління з мікоконтрлера Arduino та ESP8266.

Рисунок 2.3 – Розподіл зон управління розумного будинку

Перша зона включає в себе гараж, електрощітову та взагальний вхід. Тут разтошовані датчики відкривання вікон та дверей № 1, 2, 9, 10, датчик руху, датчик диму, датчик рівня CO_2, датчик енергоспоживання, система управління вентілятором та освітленням. Схема підключення елементів

наведено на рис 2.4 та табл. 2.1. Друга зона включає усі спальні. Тут разтошовані датчики відкривання вікон та дверей № 5, 6, 7, 3 датчики диму, 3 датчика температури та вологості, 3 датчика руху, датчик температури, 3 системи управління кондіціонерами та 3 системи освітлення. Схема підключення елементів наведено на рис 2.5 та табл. 2.2. Третья зона включає вітальну з кухнею, вбиральну та душеву кімнату. Тут разтошовані датчики відкривання вікон та дверей № 3, 4, 8, 11, 3 датчика руху, датчик диму, датчик рівня CO_2, датчик споживання води, 3 датчика протикання води, 3 системи освітлення, система управління вентілятором, система управління кондиціонером. Схема підключення елементів наведено на рис 2.6 та табл. 2.3.

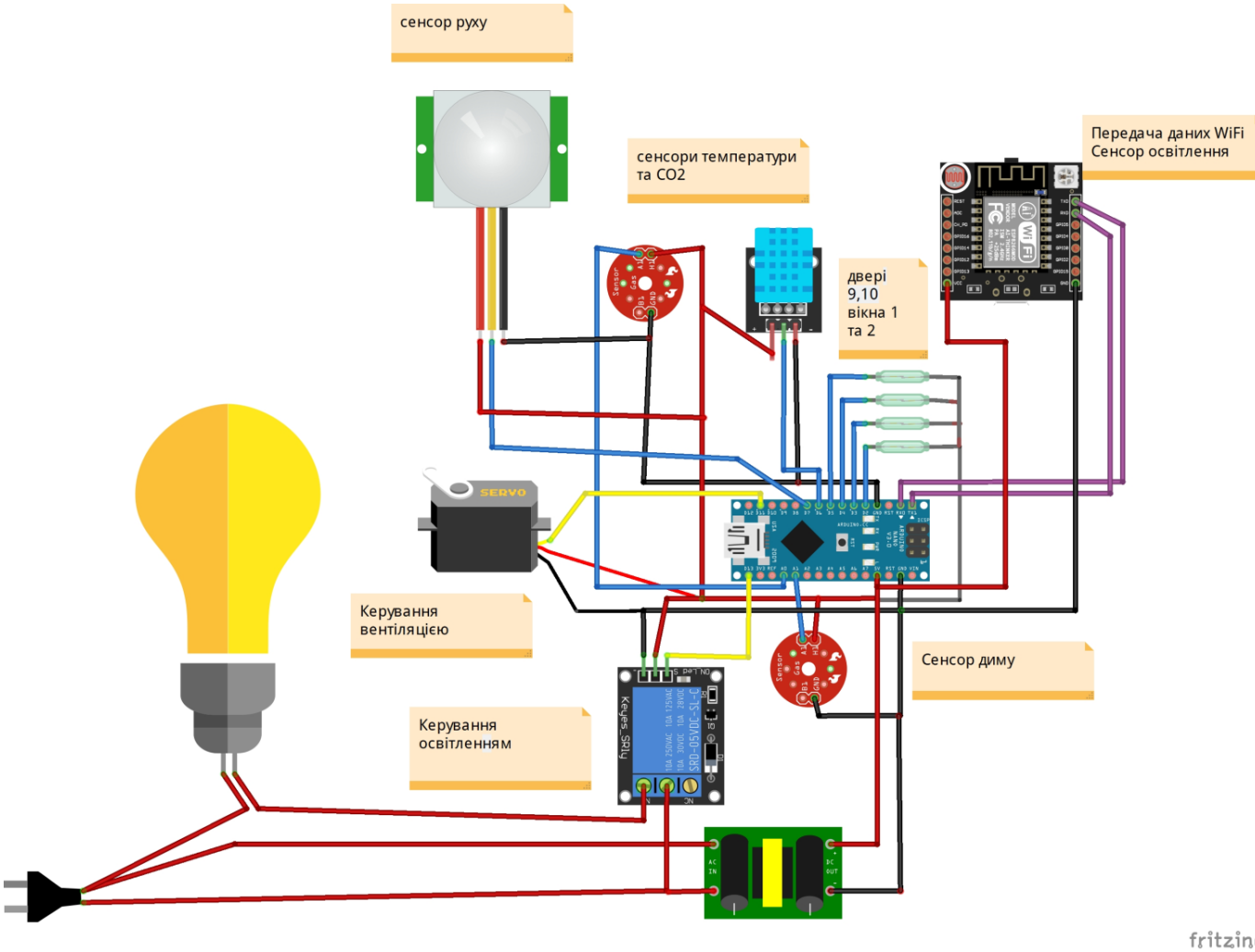

Рисунок 2.4 – Схема приладів 1 зони.

Таблиця 2.1 – Підключення пінів контролера Arduino Nano до елементів 1 зони

Номер пину Arduino	Що приєднано
TX	RX ESP8266
RX	TX ESP8266
D2	Двері 9
D3	Двері 10
D4	Вікно 1
D5	Вікно 2
D6	Out DHT11
D7	Out PIR sensor
D11	Signal Servo
D13	Signal Relay
A0	Out MQ-135
A1	Out MQ-2

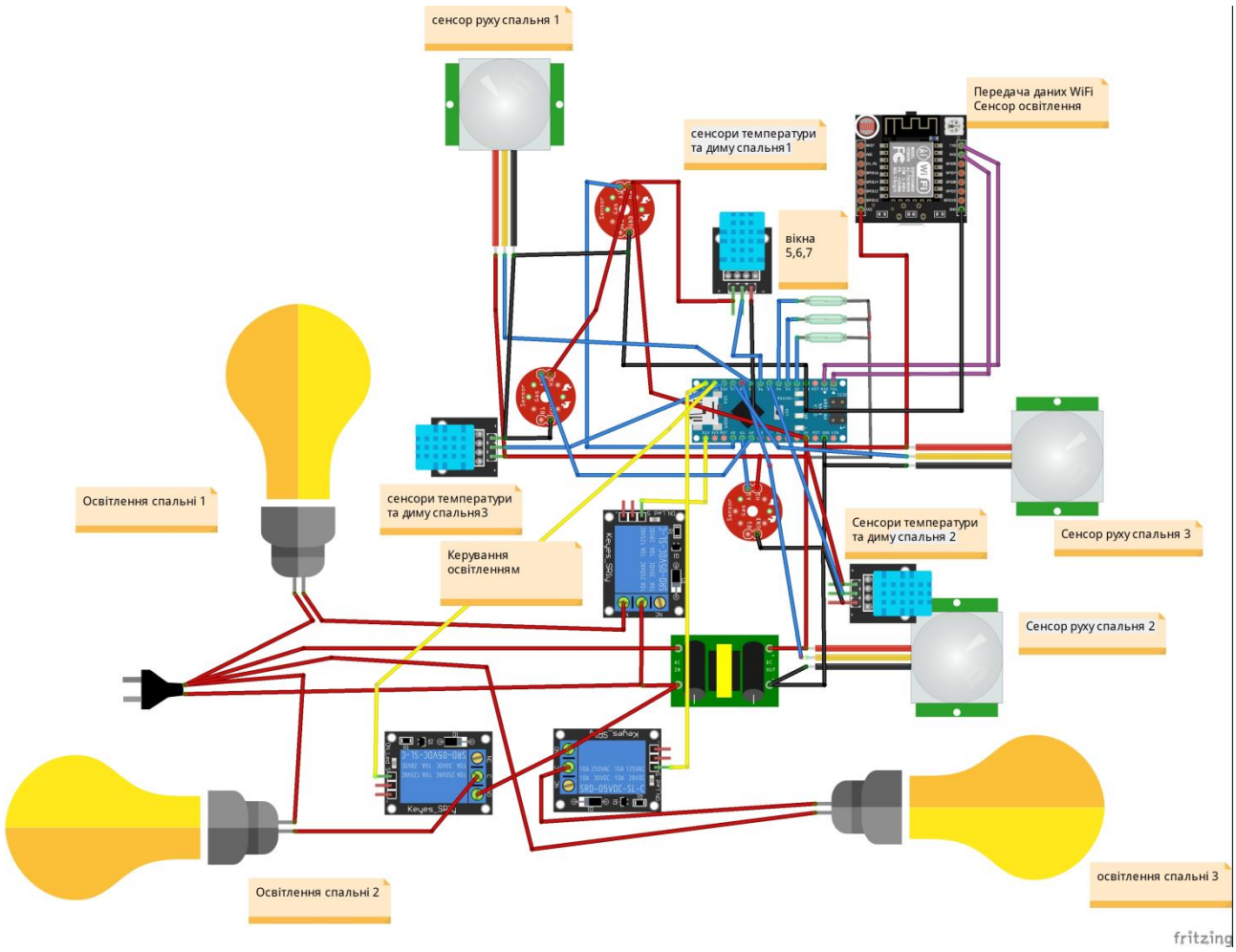

Рисунок 2.5 – Схема приладів 2 зони

Таблиця 2.2 - Підключення пінів контролера Arduino Nano до елементів 2 зони

Номер пину Arduino	Що приєднано
TX	RX ESP8266
RX	TX ESP8266
D2	Вікно 5
D3	Вікно 6
D4	Вікно 7
D5	Out DHT11 спальня 2
D6	Out DHT11 спальня 1
D7	Out PIR sensor спальня 1
D8	Out PIR sensor спальня 2
D9	Out PIR sensor спальня 3
D10	Out DHT11 спальня 3
D11	Signal Relay спальня 2
D12	Signal Relay спальня 3
D13	Signal Relay спальня 1
A0	Out MQ-2 спальня 1
A1	Out MQ-2 спальня 2
A2	Out MQ-2 спальня 3

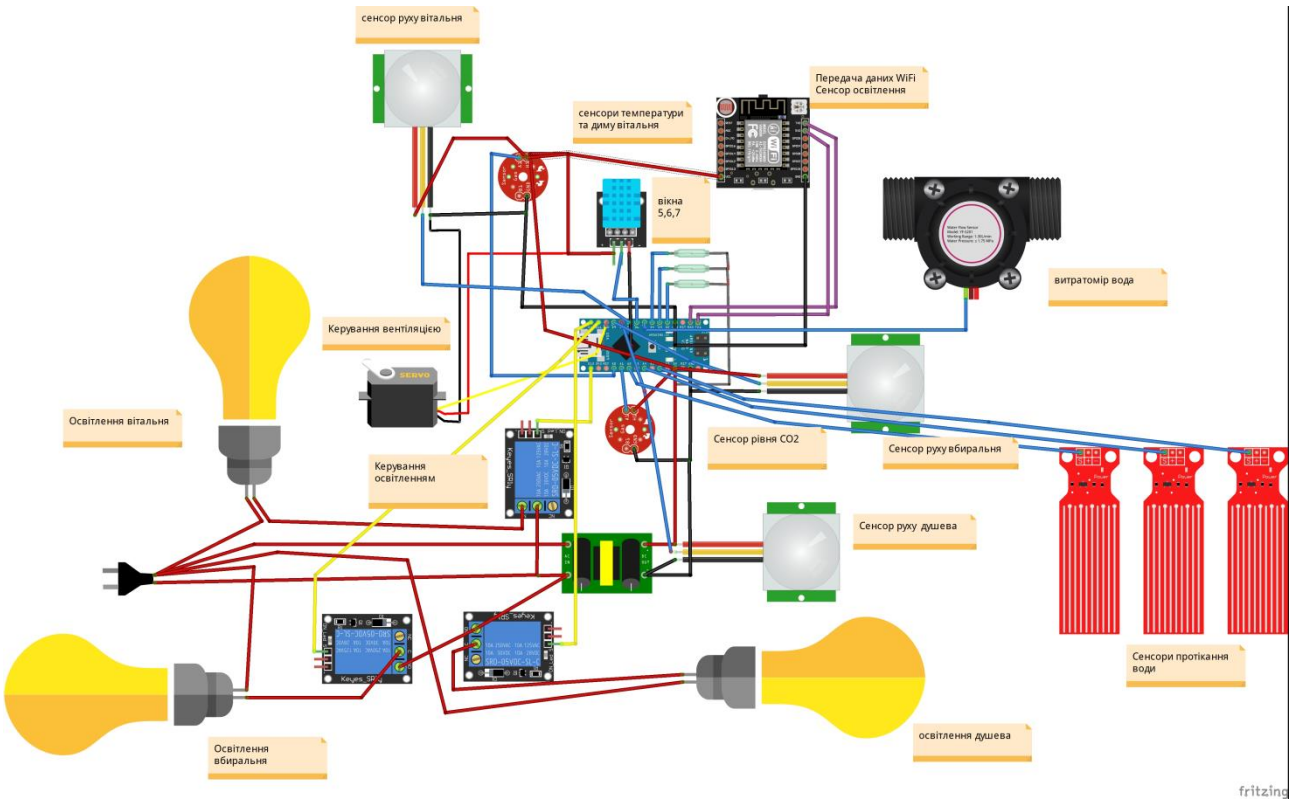

Рисунок 2.6 – Схема приладів 3 зони

Таблиця 2.3 - Підключення пінів контролера Arduino Nano до елементів 3 зони

Номер пину Arduino	Що приєднано
TX	RX ESP8266
RX	TX ESP8266
D2	Вікно 3
D3	Вікно 4
D4	Вікно 8
D5	Out витратомір води
D6	Out DHT11 вітальня
D7	Out PIR sensor вітальня
D8	Out PIR sensor душева
D9	Out PIR sensor вбиральня
D10	Двері 11
D11	Signal Relay вбиральня
D12	Signal Relay душева
D13	Signal Relay вітальня
A0	Out MQ-2 вітальня
A1	Out MQ-135 вітальня
A2	Сенсор протикання води вітальня
A4	Сенсор протикання води вбиральня
A6	Сенсор протикання води душева

3 ПРОЕКТУВАННЯ ПРОГРАММНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СИСТЕМИ РОЗУМНИЙ БУДИНОК

3.1 Загальні питання розробки програми для розумного будинку

Проектування програмного забезпечення для розумного будинку на базі Arduino та ESP8266 вимагає врахування обмежених ресурсів мікроконтролерів, необхідності реального часу обробки подій, забезпечення надійності та зручності керування. Загальні підходи до такого проектування включають модульність, подію-орієнтовану архітектуру, оптимізацію ресурсів, обробку помилок та забезпечення можливості оновлення. Розглянемо ці підходи детальніше.

Модульність

Розбиття програмного забезпечення на незалежні, функціонально завершені модулі є ключовим принципом для управління складністю та полегшення розробки, тестування та подальшої підтримки. У контексті розумного будинку кожен модуль може відповідати за керування окремим пристроєм (датчиком, реле, сервомотором), обробку певного протоколу (MQTT, HTTP) або реалізацію конкретної функції (керування освітленням, клімат-контроль, безпека).

Менші, логічно структуровані частини коду легше розуміти та підтримувати. Кожен модуль можна тестувати окремо, що спрощує виявлення та усунення помилок. Загальні функціональні блоки (наприклад, обробка даних датчика, керування реле) можуть бути використані в різних частинах проекту. Нові пристрої або функції можуть бути додані як окремі модулі без значного впливу на існуючий код. Кілька розробників можуть працювати над різними модулями паралельно.

При цьому можливо використання функцій та класів (у C++ для Arduino) для інкапсуляції логіки кожного модуля. Створення окремих файлів

(.h та .cpp) для кожного значного модуля. Визначення чітких інтерфейсів між модулями для забезпечення їхньої взаємодії. Використання патернів проектування, таких як "стратегія" або "команда", для реалізації гнучкої поведінки модулів.

Подія-орієнтована архітектура

У системах розумного будинку багато дій відбуваються асинхронно у відповідь на зовнішні події (наприклад, зміна стану датчика, натискання кнопки, отримання команди по мережі). Подія-орієнтована архітектура дозволяє реагувати на ці події ефективно та неблокуючим чином.

Система швидко реагує на зміни стану пристроїв та зовнішні команди. Процесор не витрачає час на постійне опитування станів, а виконує дії лише при виникненні подій. Завдяки неблокуючим операціям можна імітувати паралельне виконання кількох завдань. Легко додавати нові реакції на існуючі події.

Використання переривань (interrupts) для швидкої обробки критичних подій від датчиків. Реалізація механізмів підписки/сповіщення (publish/subscribe) для передачі інформації про події між модулями. Використання таймерів для періодичного виконання завдань або затримки реакцій. Створення центрального менеджера подій або використання бібліотек, що реалізують подійно-орієнтовану модель. Уникання блокуючих викликів функцій (наприклад, довгі delay()) та використання неблокуючих альтернатив (наприклад, відстеження часу за допомогою millis()).

Оптимізація ресурсів

Arduino та ESP8266 мають обмежені обсяги пам'яті (RAM та Flash) та відносно низьку обчислювальну потужність. Ефективне використання цих ресурсів є критично важливим для забезпечення стабільної роботи системи.

До головних методів оптимізації ресурсів можна віднести - використання локальних змінних замість глобальних, коли це можливо. Уникання створення великих масивів даних. Використання статичних змінних лише за необхідності. Звільнення пам'яті, яка більше не потрібна.

Використання рядкових літералів (прошитих у Flash) замість об'єктів String, де це можливо. Видалення невикористаного коду та бібліотек. Використання ефективних структур даних. Оптимізація алгоритмів для зменшення розміру коду. Використання PROGMEM для зберігання константних даних у Flash-пам'яті. Уникання довгих циклів delay(). Мінімізація кількості обчислень у критичних секціях коду. Використання переривань для швидкої обробки важливих подій.

Обробка помилок

Системи розумного будинку повинні бути надійними та стійкими до помилок, які можуть виникати через проблеми зі зв'язком, збої датчиків, некоректні вхідні дані тощо. Для цього необхідна перевірка даних, що надходять від датчиків або по мережі, на коректність та допустимість. Реалізація механізмів обробки непередбачених помилок (наприклад, використання try-catch у C++). У випадку тимчасових проблем зі зв'язком або зчитуванням даних можна реалізувати механізм повторних спроб. У випадку серйозних помилок система повинна мати можливість безпечно перейти в стабільний стан або перезавантажитися. Запис інформації про помилки для подальшого аналізу та діагностики (можна виводити в Serial Monitor або зберігати локально/віддалено). Використання світлодіодів або інших засобів індикації для відображення стану системи та наявності помилок.

Оновлення програмного забезпечення

Можливість оновлення програмного забезпечення "по повітрю" (OTA - Over-The-Air) є важливою для виправлення помилок, додавання нових функцій та покращення безпеки без необхідності фізичного підключення до пристрою. Реалізація цього можлива з використанням бібліотек, що підтримують OTA для ESP8266 (наприклад, ESP8266httpUpdate). Реалізація механізму завантаження нового образу прошивки з віддаленого сервера (HTTP, FTP). Забезпечення безпеки процесу оновлення (наприклад, перевірка контрольних сум). Реалізація механізму відкату до попередньої версії у

випадку невдалого оновлення. Розробка протоколу для ініціювання процесу оновлення (наприклад, через веб-інтерфейс або спеціальну команду).

Взагалі проектування програмного забезпечення для розумного будинку на Arduino та ESP8266 вимагає комплексного підходу, що враховує обмеження апаратних ресурсів та специфіку застосування. Застосування принципів модульності, подія-орієнтованої архітектури, оптимізації ресурсів, надійної обробки помилок та можливості оновлення дозволить створити стабільну, ефективну та легко підтримувану систему, яка відповідатиме потребам користувачів розумного будинку. Ретельне планування, розбиття на керовані модулі та увага до деталей є запорукою успішного проекту.

3.2 Розробка програмного коду для контролера Arduino Nano

В загальному випадку розробку програмного забезпечення для контролера ArduinoNano , можна розглянути на прикладі розробки програми для контролера, що обслуговує першу зону. Узагальнений алгоритм роботи наведено на рис. 3.1.

Як бачимо з таблиці 2.1 контролер Arduino Nano обслуговує 4 геркона, 2 газових сенсора, датчик руху, датчик темперетури та вологості, керує сервоприводом та релейним блоком. Для комунікацій з зовнішнім інтерфейсом зв’язок здійснюється з допомогою мікроконтролера ESP8266. Полний текст програми наведено у Додатку 1.

Для работи з герконам необхідно при визначенні пінів (функція pinMode(reedSwitch1Pin, INPUT_PULLUP); ...) замість INPUT використовувати саме INPUT_PULLUP для активації внутрішніх підтягуючих резисторів. Це повязано з необхідністю враховувати проблему брязкіту контактів (англ. contact bounce) [9] — небажане замикання й розмикання контактів в момент комутації, що виникає в електричних і електронних перемикачах і яке не передбачене заданою дією пристрою (процес триває приблизно упродовж від декількох до десятків мілісекунд).

З герконів ми отримаємо значення або 1 або 0, та відправляємо їх для подальшої обробки на сервер.

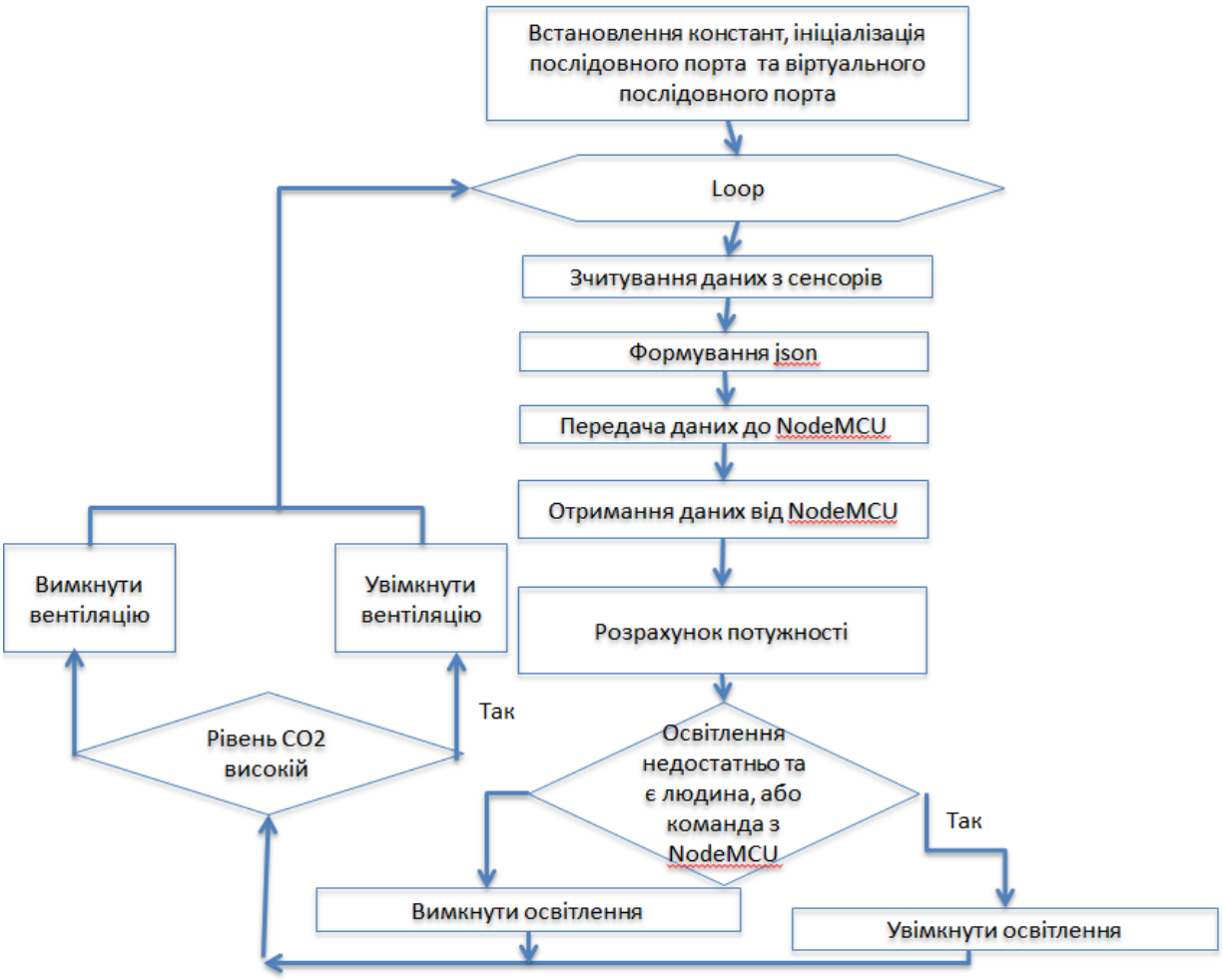

Рисунок 3.1 – Алгоритм роботи Arduino Nano (Зона 1)

Газові сенсори видають аналоговий сігнал, відповідно підключаємо їх до аналогових портів і отримуємо за допомогою АЦП сігнал в діапазоне (0,...,1023). Рівень сігналу відповідає концентрації газу/диму у повітрі.

Для роботи з сенсором DHT11 [10] та сервомотором нам необхідні бібліотеки DHT.h та Servo.h. Бібліотека DHT.h надає функції для зчитування даних про температуру та вологість, а бібліотека Servo.h дозволяє керувати сервомоторами.

Для визначення типу використовуваного DHT сенсора - define DHTTYPE DHT11. Та необхідно створити об'єкт dht класу DHT для роботи з сенсором DHT11, вказуючи пін підключення та тип сенсора - DHT dht(dhtPin,

DHTTYPE). Для керування сервомотором створюємо об'єкт myServo класу Servo.

Для встановлення поточних значено, отриманих від сенсорів оголошуються змінні та дорівнюємо їх до 0. Оголошуємо змінні для керування сервомотором (початкова позиція) та реле (початковий стан) - int pirState = 0; ... int servoPosition = 0; ...:

Встановлюємо змінні для реалізації інтервалу між зчитуваннями даних з DHT11, щоб не перевантажувати сенсор unsigned long lastDHTReadTime = 0;.

Функція setup() виконується один раз при запуску Arduino. Ініціалізує послідовний порт зі швидкістю 9600 біт/секунду для виведення налагоджувальної інформації на ESP8266 через UART - (Serial.begin(9600); ініціалізує сенсор DHT11. - dht.begin(); прив'язує об'єкт myServo до вказаного піна сервомотора - myServo.attach(servoPin); та налаштовує інши необхідні піни.

Функція loop() виконується безперервно після завершення setup().

Перевіряється, чи минув заданий інтервал (dhtReadInterval) з моменту останнього зчитування. Якщо так, зчитуються значення вологості та температури. Перевіряється наявність помилок при зчитуванні, і якщо дані успішно отримано, вони виводяться в послідовний порт.

Зчитується цифровий стан PIR сенсора. Якщо виявлено рух (HIGH), у послідовний порт виводиться повідомлення. Зчитуються аналогові значення з сенсорів MQ-2 [11] та MQ-135 [12]. Ці значення представляють рівень концентрації газів. Отримані аналогові значення виводяться в послідовний порт.

Зчитується цифровий стан кожного з чотирьох герконів. HIGH означає, що магніт далеко або відсутній (контакт розімкнений), а LOW означає, що магніт близько (контакт замкнений). Їхні стани виводяться в послідовний порт.

Відоповідно до команд отриманих з послідовного порта виконується керування сервоприводом та реле. Увімкнення реле (світла) відбувається при виявленні руху PIR сенсором та низбкого рівня освітленності на сенсорі ESP8266. Зміна положення сервомотора відбувається при високому рівні концентрації газа.

Використовуйте ESP8266 для підключення Arduino до локальної мережі або Інтернету дозволить нам віддалено контролювати та отримувати дані від нашої системи розумного будинку.

Ця функція відповідає Для зчитування даних, що надходять через послідовний порт, розроблена функція receiveAndProcessCommand().

Вона використовує буфер serialCommand для накопичення символів, доки не буде отримано символ нової строки (\n). Символ повернення каретки (\r) ігнорується. Коли команда повністю отримана (commandComplete стає true), вона виводить отриману команду в послідовний порт для підтвердження та викликає функцію processCommand() для її обробки. Після обробки команда serialCommand очищається, а commandComplete скидається в false для очікування наступної команди.

Функція processCommand(String command) приймає отриману команду у вигляді рядка (String command). Вона перевіряє, чи починається команда з певних префіксів, щоб визначити, до якого пристрою вона відноситься.

"RELAY=": Якщо команда починається з цього префікса, вона витягує значення, що йде після знака рівності. Якщо значення "ON", стан реле встановлюється в HIGH, і на відповідний пін виводиться високий рівень напруги для його ввімкнення. Якщо значення "OFF", стан реле встановлюється в LOW, і на відповідний пін виводиться низький рівень напруги для його вимкнення. Якщо отримано невідоме значення, виводиться повідомлення про помилку.

"SERVO=": Якщо команда починається з цього префікса, вона витягує значення, що йде після знака рівності, та намагається перетворити його на ціле число (toInt()). Якщо отримане число знаходиться в діапазоні від 0 до

180 (допустимий діапазон для більшості сервомоторів), воно використовується для встановлення нової позиції сервомотора за допомогою myServo.write(). Якщо отримане значення виходить за межі допустимого діапазону, виводиться повідомлення про помилку. Якщо команда не починається з жодного з відомих префіксів, виводиться повідомлення про невідому команду.

Функція readAndSendData() відповідає за зчитування даних з усіх сенсорів та їх передачу через послідовний порт . Дані про вологість та температуру передаються у форматі "DHT,humidity,temperature". У випадку помилки зчитування передається "DHT,ERR,ERR". Стан PIR сенсора (0 або 1) передається у форматі "PIR,state". Аналогове значення сенсора MQ-2 передається у форматі "MQ2,value". Аналогове значення сенсора MQ-135 передається у форматі "MQ135,value". Стани всіх чотирьох герконів (0 або 1) передаються у форматі "REED,state1,state2,state3,state4". Поточний стан реле (true або false) передається у форматі "RELAY,state". Поточна позиція сервомотора передається у форматі "SERVO,position". Кожне значення сенсора або пристрою передається на окремому рядку, починаючись з унікального префікса, що дозволяє легко розпарсити дані на приймаючій стороні.

На приймаючій стороні буде написаний код для зчитування даних з послідовного порту та розпарсування отриманих рядків на окремі значення, використовуючи префікси та роздільники (кома). Аналогічно, для керування реле та сервомотором, буде надсилани відповідні команди у форматі, описаному вище, через послідовний порт на Arduino.

3.3 Розробка програмного коду для контролера ESP 8266

Мікроконтролер ESP8266 призначен для здійснення комунікацій між головним контролером та зовнішнім інтерфейсом на веб-сайти. Він підключається до мережі Wi-Fi, надсилає дані з датчиків на сервер і керує

реле освітлення та сервоприводом на основі команд, отриманих із сервера. Код програми наведено у додатку 2. Узагальнений алгоритм роботи наведено на рис. 3.2.

Рисунок 3.2 – Алгоритм роботи ESP8266

Для забезпечення функціональності Wi-Fi на ESP8266 необхідно підключити бібліотек - ESP8266WiFi.h. Крім того, щоб забезпечити передачу HTTP-запитів необхідно підключити бібліотеку ESP8266HTTPClient.h. Для зручності відлагодження системи будемо створювати програмний UART за допомогою бібліотекі SoftwareSerial.h.

Крім того особливостю реалізації є перехід на HTTPS, це значний крок до підвищення безпеки, оскільки дані передаються в зашифрованому вигляді. Для цього використовується бібліотека WiFiClientSecure.h.

Константи ssid, password, server відповідно визначають SSID мережі Wi-Fi, пароль мережі Wi-Fi та URL-адресу сервера для надсилання даних.

В функції setup() встановлюємо відповідну швидкість передачі даних по UART, щоб забезпечити обмін даними між контролерами, підключаємось до мережі Wi-Fi.

В функції loop() виконуємо операції по отриманню даних з вбудованого сенсора освітлення та надіслані від контролера ArduinoNano. Формуємо json для відправки на сервер (рис 3.3) та відправляємо його за допомогою HTTP POST запиту та отриимуємо зворотню інформацію з вказівками для виконнанавчих механізмов яку распарсіваємо для передачи на ArduinoNano.

```
String createSensorDataJson(float temperature, float humidity,
int lightLevel, bool motion, String currentColor, bool ledState)
{
  const size_t capacity = JSON_OBJECT_SIZE(1) +
JSON_OBJECT_SIZE(6) + 100;
  StaticJsonDocument<capacity> doc;

  JsonObject data = doc.createNestedObject("data");

  data["temperature"] = temperature;
  data["humidity"] = humidity;
  data["lightLevel"] = lightLevel;
  data["motion"] = motion;
  data["state"] = ledState;

  String jsonString;
  serializeJson(doc, jsonString);

  return jsonString;
}
```

Рисунок 3.3 – Функція формування json для відправки на сервер

Замість використання високоуровневого об'єкта HTTPClient, запит тепер формується вручну. Це дає більше контролю над запитом. В цілому ця

раелізація забезпечує віддалене керування. ESP8266 виступає як "клієнт", який надсилає дані на "сервер" і отримує від нього команди . Це типова архітектура для IoT-пристроїв.

РЕЗУЛЬТАТИ РОБОТИ ТА ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було проведено аналіз існуючих вимог та принципов побудови систем «розумний будинок». Проаналізовани існуючі ситеми «розумного будинку» наявні в Україні.

Для створення проекту розроблено систему «розумний будинок» на базі мікроконтролерів Arduino Nano та ESP8266. Поєднання контролерів дає мождливість переразподілу функцій. Arduino Nano обробляє інформацію з датчиків та забезпечує керування в реальному часі, тоді як ESP8266 забезпечує підключення до Wi-Fi та зв'язок із центральним сервером. Це використовує сильні сторони обох плат. В ході роботи було розроблено макет для демонстрації окремих функцій розумного будинку. Отримані результати дозволяют з мінімальними фінансовими витратами будувати достатньо складі системи контролю і управління будинком.

Отримана система дозволяє легко масштабувати функції розумного будинку за рахунок модульності.

В якості подалшого розвитку проекту можна визначити напрямки підвищення рівня інформаційної безпеки за рахунок застосування шифрування у протоколах пердачи даних, впровадження технології Over-The-Air для дистанційного оновлення програмного забезпечення.

_____ Бригар К.О.

(Підпис автора роботи)

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dag Spicer. The echo iv home computer: 50 years later. – [Електронний ресурс]. – Режим доступу: https://computerhistory.org/blog/the-echo-iv-home-computer-50-years-later/
2. Protocol X10 and home automation. – [Електронний ресурс]. – Режим доступу: https://piko-domotique.com/en/x10.php
3. Z-Wave. – [Електронний ресурс]. – Режим доступу: https://www.home-assistant.io/integrations/zwave_js/
4. Linda Rosencrance. Zigbee – [Електронний ресурс]. – Режим доступу: https://www.techtarget.com/iotagenda/definition/ZigBee
5. Технології «розумного дому» в українських новобудовах: сучасні можливості та перспективи. – [Електронний ресурс]. – Режим доступу: https://riel.ua/blogs/tekhnologiyi-rozumnogo-domu-v-ukrayinskikh-novobudovakh-suchasni-mozhlivosti-ta-perspektivi
6. П. Кузнецов. Розробка та впровадження системи автоматизації розумного будинку в умовах українського житлового сектору: виклики та перспективи. – [Електронний ресурс]. – Режим доступу: https://bulletin-chstu.com.ua/uk/journals/tom-29-1-2024/rozrobka-ta-vprovadzhennya-sistemi-avtomatizatsiyi-rozumnogo-budinku-v-umovakh-ukrayinskogo-zhitlovogo-sektoru-vikliki-ta-perspektivi
7. Arduino Nano Pinout, Specs, and How to Use It. – [Електронний ресурс]. – Режим доступу: https://docs.arduino.cc/hardware/nano/
8. ESP8266 Welcome to ESP8266 Arduino Core’s documentation! – [Електронний ресурс]. – Режим доступу: https://arduino-esp8266.readthedocs.io/en/latest
9. Брязкіт контактів. – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%91%D1%80%D1%8F%D0%B7%D0%BA%D1%96%D1%82_%D0%BA%D0%BE%D0%BD%D1%82%D0%B0%D0%BA%D1%82%D1%96%D0%B2

10. – DHT11 Humidity & Temperature Sensor. – [Електронний ресурс]. – Режим доступу: https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf
11. Gas/Smoke Sensor Module (MQ2) – [Електронний ресурс]. – Режим доступу: https://docs.sunfounder.com/projects/ultimate-sensor-kit/en/latest/components_basic/02-component_gas.html
12. MQ135 Air Quality Sensor : Pin Configuration, Working & Its Applications – [Електронний ресурс]. – Режим доступу: https://www.elprocus.com/mq135-air-quality-sensor/

ДОДАТОК А

Програма для Arduino Nano

```
#include <DHT.h>
#include <Servo.h>
#include <ArduinoJson.h>

const int DHT_PIN = 6;
const int PIR_PIN = 7;
const int MQ2_PIN = A1;
const int MQ135_PIN = A0;
const int REED_SWITCH_1_PIN = 4;
const int REED_SWITCH_2_PIN = 5;
const int REED_SWITCH_3_PIN = 2;
const int REED_SWITCH_4_PIN = 3;

const int RELAY_PIN = 13;
const int SERVO_PIN = 11;

#define DHT_TYPE DHT11
DHT dht(DHT_PIN, DHT_TYPE);
Servo myServo;

float temperature = 0.0;
float humidity = 0.0;
int pirState = 0;
int mq2Value = 0;
int mq135Value = 0;
int reedSwitchStates[4] = {0, 0, 0, 0};

bool currentRelayState = LOW;
int currentServoPos = 90;
String serialCommand = "";
bool commandComplete = false;
unsigned long lastSensorReadTime = 0;
const long sensorReadInterval = 3000;


void setup() {

  Serial.begin(9600);
  Serial.println("Arduino Nano startue...");

  dht.begin();

  pinMode(PIR_PIN, INPUT);
  pinMode(REED_SWITCH_1_PIN, INPUT_PULLUP);
  pinMode(REED_SWITCH_2_PIN, INPUT_PULLUP);
  pinMode(REED_SWITCH_3_PIN, INPUT_PULLUP);
  pinMode(REED_SWITCH_4_PIN, INPUT_PULLUP);
```

```
  myServo.attach(SERVO_PIN);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, currentRelayState);
  myServo.write(currentServoPos);
  Serial.println("Arduino Nano inicializovano.");
}


void loop() {
  unsigned long currentTime = millis(); // Поточний час


  if (currentTime - lastSensorReadTime >= sensorReadInterval) {
    readSensors();
    sendSensorData();
    lastSensorReadTime = currentTime;  }

   receiveAndParseCommand();

   controlActuators();

   delay(50);
}


// Функція для зчитування всіх датчиків
void readSensors() {
   float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (isnan(h) || isnan(t)) {
    Serial.println("DHT sensor ne pracue");
  } else {
    temperature = t;
    humidity = h;
  }

    pirState = digitalRead(PIR_PIN);


  mq2Value = analogRead(MQ2_PIN);
  mq135Value = analogRead(MQ135_PIN);


  reedSwitchStates[0] = !digitalRead(REED_SWITCH_1_PIN);
  reedSwitchStates[1] = !digitalRead(REED_SWITCH_2_PIN);
  reedSwitchStates[2] = !digitalRead(REED_SWITCH_3_PIN);
  reedSwitchStates[3] = !digitalRead(REED_SWITCH_4_PIN);


}
```

```
// Функція для форматування та надсилання даних датчиків через
//Serial до ESP8266
// Формат:
//T:25.5,H:60.2,PIR:1,MQ2:700,MQ135:800,R1:0,R2:1,R3:0,R4:1\n
void sendSensorData() {
  Serial.print("T:"); Serial.print(temperature);
  Serial.print(",H:"); Serial.print(humidity);
  Serial.print(",PIR:"); Serial.print(pirState);
  Serial.print(",MQ2:"); Serial.print(mq2Value);
  Serial.print(",MQ135:"); Serial.print(mq135Value);
  Serial.print(",R1:"); Serial.print(reedSwitchStates[0]);
  Serial.print(",R2:"); Serial.print(reedSwitchStates[1]);
  Serial.print(",R3:"); Serial.print(reedSwitchStates[2]);
  Serial.print(",R4:"); Serial.print(reedSwitchStates[3]);
  Serial.println(); // Новий рядок як роздільник повідомлень
}

// Функція для отримання та розпакування JSON-команд від ESP8266

void receiveAndParseCommand() {
  while (Serial.available() > 0) {
    char inChar = (char)Serial.read();
    if (inChar == '\n') {
      commandComplete = true;
      break;
    }
    if (inChar != '\r') {
      serialCommand += inChar;
    }
  }

  if (commandComplete) {
    Serial.print("Received command JSON: ");
    Serial.println(serialCommand);


    const size_t capacity = JSON_OBJECT_SIZE(2) + 50;
    StaticJsonDocument<capacity> doc;

    DeserializationError error = deserializeJson(doc,
serialCommand);


    if (error) {
      Serial.print(F("JSON parsing error: "));
      Serial.println(error.f_str());
    } else {

      if (doc.containsKey("relayState")) {
        currentRelayState = doc["relayState"].as<bool>();
      } else {
        Serial.println("Vidpravka 'relayState' do JSON
comand.");
```

```
      }

      if (doc.containsKey("servoPos")) {
        currentServoPos = doc["servoPos"].as<int>();

        if (currentServoPos < 0) currentServoPos = 0;
        if (currentServoPos > 180) currentServoPos = 180;
      } else {
        Serial.println("Vidpravka 'servoPos' do JSON comand.");
      }

      Serial.print("Otrymannja statusu rele: ");
Serial.println(currentRelayState ? "ON" : "OFF");
      Serial.print("Otrymannja pozycii servo: ");
Serial.println(currentServoPos);
    }

    serialCommand = "";
    commandComplete = false;
  }
}

// Функція для керування реле та сервоприводом

void controlActuators() {

  digitalWrite(RELAY_PIN, currentRelayState ? LOW : HIGH);

  myServo.write(currentServoPos);
}
```

ДОДАТОК Б

Программа для ESP8266

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClientSecure.h>
#include <ArduinoJson.h>

const char* ssid = "********";
const char* password = "********";


const char* serverHost = "smart-home-6j54.onrender.com"; const
int httpsPort = 443;
const char* dataEndpoint = "/api/esp8266/data";

const int ESP_LIGHT_SENSOR_PIN = A0;

// --- Структура для зберігання розпарсених даних з Arduino Nano
struct NanoSensorData {
    float temperature = 0.0;
    float humidity = 0.0;
    int pirState = 0;
    int mq2Value = 0;
    int mq135Value = 0;
    int reedSwitchStates[4] = {0, 0, 0, 0};
    bool newDataAvailable = false; };
NanoSensorData nanoData;

//Структура для зберігання розпарсених даних з відповіді сервера
struct ServerResponseData {
    bool success = false;
    bool relayState = false;
    String timestamp = "";
    bool parseSuccess = false; };
ServerResponseData serverResponse;

String nanoSerialBuffer = ""; //Буфер для збору даних з Serial
bool nanoDataComplete = false;

unsigned long lastSendTime = 0;
const long sendInterval = 3000;


void setup() {
  Serial.begin(9600);
  delay(2000);
  Serial.println("\nESP8266 Startue...");
  Serial.print("Pryednannja do WiFi: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
  }

  Serial.println();
  Serial.println("WiFi pracue!");
  Serial.print("ESP8266 IP address: ");
  Serial.println(WiFi.localIP());
}


void loop() {
  unsigned long currentTime = millis();
  readSerialFromNano();

  if (WiFi.status() == WL_CONNECTED && (currentTime -
lastSendTime >= sendInterval || nanoData.newDataAvailable)) {
    sendDataToServer();
    lastSendTime = currentTime;
    nanoData.newDataAvailable = false;

  } else if (WiFi.status() != WL_CONNECTED) {
    Serial.println("WiFi ne pracue, povtorne zziednannja...");
    WiFi.begin(ssid, password);

  if (serverResponse.parseSuccess) {
      sendCommandsToNano();
      serverResponse.parseSuccess = false;  }

  delay(50);
}

// Функція для зчитування даних з послідовного порту від
// Arduino Nano
void readSerialFromNano() {
  while (Serial.available() > 0) {
    char inChar = (char)Serial.read();
    nanoSerialBuffer += inChar;
    if (inChar == '\n') {
      nanoDataComplete = true;
      break;
    }
  }

  if (nanoDataComplete) {
    Serial.print("Otrymano vid Nano: ");
    Serial.println(nanoSerialBuffer);

     parseNanoData(nanoSerialBuffer);
     nanoData.newDataAvailable = true;

    nanoSerialBuffer = "";
    nanoDataComplete = false;
  }
}
```

```
// --- Функція для розпакування даних від Arduino Nano
// Формат:
//T:temp,H:hum,PIR:pir,MQ2:mq2,MQ135:mq135,R1:r1,R2:r2,R3:r3,R4:
//r4\n
void parseNanoData(String dataString) {
  dataString.trim();

  int startIndex = 0;
  int endIndex = 0;


  startIndex = dataString.indexOf("T:") + 2;
  endIndex = dataString.indexOf(",H:", startIndex);
  if (startIndex > 1 && endIndex > startIndex) {
    nanoData.temperature = dataString.substring(startIndex,
endIndex).toFloat();
  }

  startIndex = dataString.indexOf("H:") + 2;
  endIndex = dataString.indexOf(",PIR:", startIndex);  if
(startIndex > 1 && endIndex > startIndex) {
    nanoData.humidity = dataString.substring(startIndex,
endIndex).toFloat();
  }

  startIndex = dataString.indexOf("PIR:") + 4;
  endIndex = dataString.indexOf(",MQ2:", startIndex);
  if (startIndex > 3 && endIndex > startIndex) {
    nanoData.pirState = dataString.substring(startIndex,
endIndex).toInt();
  }

    startIndex = dataString.indexOf("MQ2:") + 4;
  endIndex = dataString.indexOf(",MQ135:", startIndex);
  if (startIndex > 3 && endIndex > startIndex) {
    nanoData.mq2Value = dataString.substring(startIndex,
endIndex).toInt();
  }

    startIndex = dataString.indexOf("MQ135:") + 6;
  endIndex = dataString.indexOf(",R1:", startIndex);
  if (startIndex > 5 && endIndex > startIndex) {
    nanoData.mq135Value = dataString.substring(startIndex,
endIndex).toInt();
  }


  for (int i = 0; i < 4; i++) {
    String key = "R" + String(i + 1) + ":";
    startIndex = dataString.indexOf(key) + key.length();
    if (startIndex > key.length() -1) {
      if (i < 3) {
```

```
        endIndex = dataString.indexOf(",R" + String(i + 2) +
":", startIndex);
      } else {
        endIndex = dataString.length();
      }
      if (endIndex > startIndex) {
        nanoData.reedSwitchStates[i] =
dataString.substring(startIndex, endIndex).toInt();
      }
    }
  }

  Serial.println("Nano Dani proanalizovano:");
  Serial.print("  Temp: ");
Serial.println(nanoData.temperature);
  Serial.print("  Hum: "); Serial.println(nanoData.humidity);
  Serial.print("  PIR: "); Serial.println(nanoData.pirState);
  Serial.print("  MQ2: "); Serial.println(nanoData.mq2Value);
  Serial.print("  MQ135: ");
Serial.println(nanoData.mq135Value);
  Serial.print("  Reed 1: ");
Serial.println(nanoData.reedSwitchStates[0]);
  Serial.print("  Reed 2: ");
Serial.println(nanoData.reedSwitchStates[1]);
  Serial.print("  Reed 3: ");
Serial.println(nanoData.reedSwitchStates[2]);
  Serial.print("  Reed 4: ");
Serial.println(nanoData.reedSwitchStates[3]);
}


//  Функція для відправки даних на сервер через HTTPS POST
void sendDataToServer() {
  Serial.println("\nNadsylannya danykh na server...");
  WiFiClientSecure client;

  if (!client.connect(serverHost, httpsPort)) {
    Serial.println("Z_yednannya ne vdalosya!");
    return;
  }

  int espLightLevel = analogRead(ESP_LIGHT_SENSOR_PIN);
  Serial.print("ESP8266 Light level (A0): ");
  Serial.println(espLightLevel);

 const size_t capacity = JSON_OBJECT_SIZE(1)
+ JSON_OBJECT_SIZE(11) + 256;
  StaticJsonDocument<capacity> doc;
  JsonObject data = doc.createNestedObject("data");

  data["temperature"] = nanoData.temperature;
  data["humidity"] = nanoData.humidity;
  data["lightLevel"] = espLightLevel;
```

```
  data["motion"] = (bool)nanoData.pirState;
  data["mq2Value"] = nanoData.mq2Value;
  data["mq135Value"] = nanoData.mq135Value;
  data["reedSwitch1"] = (bool)nanoData.reedSwitchStates[0];
  data["reedSwitch2"] = (bool)nanoData.reedSwitchStates[1];
  data["reedSwitch3"] = (bool)nanoData.reedSwitchStates[2];
  data["reedSwitch4"] = (bool)nanoData.reedSwitchStates[3];

  String jsonPayload;
  serializeJson(doc, jsonPayload);

  String request = String("POST ") + dataEndpoint +
                   " HTTP/1.1\r\n" +
                   "Host: " + serverHost + "\r\n" +
                   "Content-Type: application/json\r\n" +
                   "Content-Length: " + jsonPayload.length() +
                   "\r\n" +
                   "Connection: close\r\n\r\n" + jsonPayload;

  Serial.println("Nadsylannya zapytu:");
  Serial.println(request);

  client.print(request);
  Serial.println("Ochikuvannya vidpovidi...");


  String httpResponse = "";
  while (client.connected()) {
    String line = client.readStringUntil('\n');
    httpResponse += line;
    if (line == "\r") {
      Serial.println("Otrymani zaholovky ");
      break;
    }
  }


  String responseBody = "";
  while (client.available()) {
    responseBody += (char)client.read();
  }

  Serial.println("Otrymana vidpovid':");
  Serial.println(responseBody);

    parseServerResponseJson(responseBody);

   client.stop();
}

// --- Функція для розпакування JSON-відповіді від сервера ---
// Очікуваний формат: {"success":true,"data":{"relayState":true,
//"timestamp":"..."
```

```
void parseServerResponseJson(String jsonString) {
  // Визначаємо необхідний розмір JsonDocument
  const size_t capacity = JSON_OBJECT_SIZE(2) +
JSON_OBJECT_SIZE(3) + 200;
  DynamicJsonDocument doc(capacity);

  DeserializationError error = deserializeJson(doc, jsonString);
  serverResponse.parseSuccess = false;
  if (error) {
    Serial.print(F("Server response JSON parsing error: "));
    Serial.println(error.f_str());
    return;
  }

    if (doc.containsKey("success")) {
    serverResponse.success = doc["success"].as<bool>();
  } else {
    Serial.println(F("Missing 'success' key in server
response."));
    return;
  }

    JsonObject dataObject = doc["data"];
  if (dataObject.isNull()) {
    Serial.println(F("Missing or invalid 'data' object in server
response."));
    return;
  }

    if (dataObject.containsKey("relayState")) {
    serverResponse.relayState =
dataObject["relayState"].as<bool>();
  } else {
    Serial.println(F("Missing 'relayState' key in server
response."));
    return;
  }


  if (dataObject.containsKey("timestamp")) {
    serverResponse.timestamp =
dataObject["timestamp"].as<String>();
  } else {
    Serial.println(F("Missing 'timestamp' key in server
response."));
    serverResponse.timestamp = "";
  }

  serverResponse.parseSuccess = true;
  Serial.println("Server Response Parsed: Success=" +
String(serverResponse.success) +
                 ", RelayState=" +
String(serverResponse.relayState));
```

```
}

// --- Функція для надсилання команд на Arduino Nano через
//Serial ---
void sendCommandsToNano() {
  Serial.println("\nSending commands to Arduino Nano...");

  // Формуємо JSON-команду для Nano
  // Очікуваний формат Nano: {"relayState":true, "servoPos":90}
  const size_t capacity = JSON_OBJECT_SIZE(2) + 50;
StaticJsonDocument<capacity> doc;

  doc["relayState"] = serverResponse.relayState;
  doc["servoPos"] = 90;
  String nanoCommandJson;
  serializeJson(doc, nanoCommandJson);

  Serial.print("Komanda dlya Nano: ");
  Serial.println(nanoCommandJson);

  Serial.println(nanoCommandJson);
}
```