ВСЕУКРАЇНСЬКИЙ КОНКУРС СТУДЕНТСЬКИХ НАУКОВИХ РОБІТ

Назва спеціальності: інженерія вбудованих систем

СТУДЕНТСЬКА НАУКОВА РОБОТА

*«*Розумний будинок на Raspberry PI*»*

Шифр: *«*Зручність керування*»*

Харків 2021

АНОТАЦІЯ

Мета роботи – створення модульної системи розумного будинку з високим коефіцієнтом можливості/ціна, здібної працювати в умовах низько швидкісного Internet та об’єднуючої дротові та бездротові технології.

Актуальність роботи – створена система поєднує передові ідеї ведучих розробників світу та враховує особливості вітчизняних мереж і запитів користувачів.

Завдання – розробка структури сучасної, універсальної системи розумного будинку для вітчизняного користувача, алгоритмів функціонування, програмна реалізація частини системи.

Методика дослідження – системний підхід, порівняння, тестування

Загальна характеристика роботи – результатами роботи є: структура, алгоритми функціонування та тестове програмне забезпечення розумного дому, що розроблено мовами Python, html і Java script для апаратної платформи Raspberry PI. Використаний підхід дозволяє поєднати дротові та бездротові технології, а також легко додавати нові підсистеми та розширювати функціонал вже підключених. Наявність декількох інтерфейсів користувача та можливість функціонування в умовах низькошвидкісного Internet є важливими характеристиками на Українському ринку.

ЗМІСТ

ВСТУП

За останні роки у світі та в Україні зростає попит на автоматизацію виробництва та будівель. Впровадження автоматизованих систем дозволяє вирішувати багато актуальних задач: зберігання енергоресурсів, забезпечення комфорту людини, організація систем безпеки як від грабіжників, так і від вогню, протоку води та т.і. На світовому ринку автоматизованих систем існує багато готових рішень у галузі автоматизації та компаній, що пропонують проектування автоматизованих систем під ключ. Готові рішення дозволяють у короткий інтервал часу реалізувати керування виробництвом, будівлею або квартирою, але пропонують тільки обмежений спектр можливостей та не забезпечують адаптацію керуючої системи під конкретного споживача. Тому у останні роки при розробці автоматизованих систем все частіше користувачі звертаються до невеликих фірм, що за розумні гроші впроваджують рішення, адаптовані під конкретного споживача.

В останні роки при автоматизації будівель та житла з’явився новий термін «Розумний будинок». Під ним розуміють не тільки автоматизовану систему керування механізмами, приладами будинку, а і об’єднання з сучасними хмарними технологіями, голосовими асистентами та гелокацією.

Основними критеріями з розробки сучасної системи розумного будинку є: надійність, захищеність від несанкціонованого доступу, можливість автономного функціонування, модульність для обирання функцій, що потрібні конкретному споживачу та зручний інтерфейс користувача.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Основні функції автоматизації житла:

- клімат-контроль приміщень;
- керування світлом та електроспоживанням;
- моніторинг інженерних мереж;
- аварії (газ, вода, тощо...);
- охорона приміщення;
- віддалений доступ та моніторинг всіх інженерних систем.

1.2 Класифікація розумних будинків та приклади розробників

Перелік систем по основних ознаках[1,2,3]:

- Дротові.
- Бездротові.
- Централізовані.
- Децентралізовані.
- З відкритим протоколом.
- Із закритим протоколом.
- Дротові системи автоматизації.

Суть дротової системи "розумний будинок" полягає в тому, що всі керуючі пристрою - датчики, вимикачі, пристрої керування кліматом, різноманітні керуючі панелі зв'язуються єдиною дротовою інформаційною шиною, по якій ідуть сигнали- телеграми до виконавчих пристроїв, розташованих у щиті. У якості дротової інформаційної шини використовуються спеціальні кабелі, а в окремих випадках звичайна кручена пари. У дротовій системи є свої переваги й особливості.

Переваги:

- Надійність. Сигнал, що йде по спеціальних проводах - це надійно.

- Швидкість відгуку. Розумний будинок - це комфорт, тому якщо після натискання на клавішу запуску сценарію відбувається значна затримка, то це викликає дискомфорт і бажання нажати на кнопку ще й ще, тим самим інформаційна шина "забивається командами" і висне. Якщо сигнал іде по проводах, то швидкість відгуку висока, тому що ця система (правильно спроектована) є помехозахищеною і надійною.

- Перелік керуючих елементів. У таких систем у більшості пропонується великий вибір керуючих елементів (розумних вимикачів), у порівнянні з бездротовою системою. Вони постачені більшою кількістю функцій і можливостей.

- Різноманітність інтегрувальних систем. У дротових системах легше зробити інтеграцію із кліматом, аудио й відеомультирумом, чим в бездротових.

- Довгий термін служби. Система не має пристроїв на батарейках, які вимагали б регулярної заміни.

- Пожаробезпечність. Усі вимикачі є слабкострумовими і электро й пожежобезпечними

Недоліки:

- Місця розташування вимикачів (керуючих панелей необхідно вибирати заздалегідь), виводити туди кабель

- Якісний монтаж. Необхідно користуватися послугами кваліфікованих електромонтажників, та й будівельників взагалі. У випадку, якщо інформаційне проведення буде перебито, то система працювати не зможе і необхідно шукати та відновлювати з'єднання.

- У більшості випадків потрібен проект - на нього необхідно виділити час і ресурси

- У випадку з дерев'яними будинками, необхідно розробити й погодити проект заздалегідь, щоб пропили під проводку й під керуючі панелі були зроблені на виробництві заздалегідь

- Особлива топологія прокладки кабелів. Для реалізації проекту необхідно прокладати кабелі від усіх керованих приладів до щита. У підсумку в районі щита

утворюється досить значний пучок проводів який може здивувати, однак після того як щит змонтований - проведення стають не видні й щитова здобуває закінчений і охайний вид, природно, у випадку кваліфікованого монтажу.

- Установлюється така система може тільки на початку ремонту, поки не зроблена основна електропроводка за класичною схемою. У готовому ремонті, на жаль зробити провідний розумний будинок не вийде.

- Потрібен щит досить великих розмірів

Бездротові системи автоматизації:

У цих системах, на відміну від дротових, сигнал від керуючих пристроїв до виконавчих іде по радіоканалу, а не по проводах. Це дозволяє скоротити кількість проводів, а також час на інсталяцію системи. Ці системи можна монтувати на об'єкти з готовим ремонтом із класичною проводкою. Кожний бездротової "вимикач" є ще й радіопередавачем, який зв'язується з усіма іншими "вимикачами". Це дозволяє створювати різні світлові сценарії (нічний режим, виключити всі і т.д.), перепрограмувати функціонал клавіш.

Переваги

- Можна встановлювати у квартири й будинку із уже готовим ремонтом із класичною проводкою. Якщо використовувати повністю бездротової вимикач, який працює на батарейках і посилає сигнал виконавчому пристрою (наприклад радіореле, розташованому близько світильника або світлової групи), то такий вимикач можна розташувати скрізь, де тільки завгодно. Вони можуть бути як накладного так і вбудованого монтажу.

- Зменшення кількості проводів, у порівнянні із провідною системою. Цим викликана популярність подібних систем у дерев'яних будинках.

- Не потрібен проект. У більшості випадків проектування системи автоматизації не потрібно.

- Вартість. На ринку є багато систем з невисокою вартістю

Недоліки

- Радіоканал. Система, що працює по радіоканалу залежить від якості радіозв'язку. Перешкоди від СВЧ печей, будівельної техніки, DECT телефонів

можуть вплинути на проходження сигналу. Знову ж, матеріал стін, розтягнута по стіні електрогірлянда можуть виявити критичне значення на силу сигналу.

- Батарейки. Якщо система працює на батарейках, то їх необхідно міняти, причому регулярно. Якщо цього не зробити, то в самий відповідальний момент щось десь не спрацює.

- Необхідність нульового проведення. Є системи, у яких використовуються радіопередавачі, що харчуються від мережі змінного струму. Для них необхідне нульове проведення. У класичній проводці до вимикача підходить одна жила(фаза) і вона ж іде до групи світла. Тому краще відразу закласти додаткове нульове проведення в коробку під вимикач.

- Обмеженість функціонала. Дуже складно створити на радіоканалі стабільну повнофункціональну систему, яка управляла б усім, а не тільки світлом і теплими підлогами.

- Безпека. Якщо у випадку із провідною системою ми можемо обрубати всі зовнішні зв'язки - Wifi, інтернет, але система продовжить працювати, то у випадку відсутності провідної інформаційної шини ми не зможемо зробити цього відключення. Глушіння сигналу, переклад датчиків у режим підвищеного енергоспоживання і т.д. може швидко вивести систему з ладу.

- Частота роботи роботи систем 433 Мгц і 868Мгц. 433 Мгц використовують такі виробники як Jung, Gira. На цій же частоті працюють бездротові телефони, які можуть створювати перешкоди в роботі радіосистеми. Деякі виробники використовують більш перспективну частоту 868 МГц - Z-Wave, Vitrum, Zamel (Extra Free), inels і деякі інші. Однак є складності в реєстрації цієї частоти в Україні - це заважає її широкому застосуванню й просуванню.

Виробники бездротових систем: Z-Wave, Vitrum, Zamel, Delumo, Gira, Jung, HDL, Berker, Ectostroy, inels і інші

Централізовані системи автоматизації

Суть централізованого розумного будинку полягає в тому, що керування йде з одного центрального логічного модуля. Звичайно це вільно програмувальний контролер з більшою кількістю виходів. У контролер

заливається заздалегідь спеціально створена під об'єкт програма, на основі якої йде керування виконавчими пристроями та інженерними системами. Це дозволяє використовувати широкий вибір устаткування й складних сценаріїв. Централізовані системи можуть бути як провідними (Ctestron, AMX, Evika), так і бездротовими (Z-wave)

Переваги

- Можливість керування всіма інженерним системами в єдиному інтерфейсі
- Можливість створювати складні сценарії, прив'язані до часу доби, стану мешканця, температурі, місячному циклу.
- Можливість практично підключення будь-якого встаткування

Недоліки

- Людський фактор. Програміст, який написав програму є головною фігурою. У випадку, якщо із програмістом контакт загублений, то якщо буде потреба перепрограмувати центральний контролер необхідно буде заново писати всю програму. Програмування таких систем коштує досить відчутне.

- Надійність. Якщо контролер виходить із ладу, то перестає функціонувати вся система повністю. Звичайно контролери роблять дуже надійними, але прийнято вважати цю централізацію головним недоліком, хоча вихід з ладу блоку живлення розподіленої системи також виводить із ладу всю систему, хоча після заміни блоку живлення працездатність повністю відновлюється, програма не "злітає"

- Вартість. Більші можливості спричиняють і відносно значну вартість.

Виробники централізованих систем: CRESTRON, AMX, Bechoff, EVIKA, Z-WAVE, Ectostroy

Децентралізовані системи автоматизації

У розподілених системах "Розумного будинку" кожний виконавчий пристрій несе в собі мікропроцесор з енергонезалежною пам'яттю. Цим пояснюється надійність таких систем. При виході з ладу одного пристрою вся система працює справно, крім приладів підключених до цього пристрою.

Прикладом децентралізованої системи є "розумний удома" побудовані на основі протоколу KNX (самого популярного в Європі).

Переваги

- Надійність. Усі пристрої не залежать друг від друга й мають енергонезалежну пам'ять

- Популярність. Стандарт KNX, наприклад, дуже популярний і у вас не виникне складності з обслуговуванням

- Можливість використовувати додатковий блок логіки, який буде відповідати за специфічні сценарії

- Великий вибір керуючих панелей як по дизайну так і по функціоналу.

Недоліки:

Досить велике число пристроїв у щиті. Кількість пристроїв у щиті досить велике, тому при виборі сумнівного виробника (а такі є, тому що все покладають на ринок розумних будинків більші надії) замовник ризикує зіштовхнутися з виходом з ладу того або іншого пристрою, який буде потрібно замінити.

Виробники децентралізованих систем: ABB, Gira, Berker, Bticino, Vimar, Jung, HDL і т.д.

Системи автоматизації з відкритим протоколом.

Протокол - це мова на якому спілкуються всі пристрої в "розумному будинку". Якщо обрати протокол KNX, то він є відкритим. Багато виробників виготовляють пристрої, що працюють на цій мові. Асоціація KNX перевіряє їх на сумісність і тестує. Логотип KNX EIB на пристрої гарантує підвищена якість.

Переваги:

- Великий вибір виробників. Це значить, що є великий вибір пристроїв по дизайну, ціні, характеристиках

- Відновлення і конкуренція. Виробники конкурують в одному сегменті, що змушує їх розбудовуватися та придумувати нові пристрої

Переваги:

- Вартість ледве вище чому в систем із закритим протоколом за рахунок підвищеного контролю якості й просування єдиного стандарту

- Не висока гнучкість при створенні нових пристроїв. Необхідність проходження стандартам накладає свій відбиток

Системи автоматизації із закритим протоколом.

Для того, щоб спростити процес програмування, зменшити витрати на виробництво встаткування деякі виробники випускають устаткування, що працює на власному закритому протоколі. Крім них ніхто таке встаткування не випускає.

Переваги:

- Наявність цікавих розв'язків по більш низькій ціні

- Вартість у цілому нижче, чим у систем з відкритим протоколом (хоча й не завжди)

- Більш швидка реакція на вимоги ринку

Особливості:

- Залежність від одного виробника;

- Найчастіше усічені функції.

Приклади виробників систем із закритим протоколом: ABB free@home, Vimar By-Me, Bticino, MY HOME, HDL BUS PRO.

1.3 Приклади існуючих систем та їх характеристики

Majordomo - це безкоштовний (або open source) софт для створення розумного будинку. Або для “домашньої автоматизації”, своїми руками із пристроїв від різних виробників. Основна мета - дати можливість зробити розумний будинок з "збірної солянки" пристроїв, щоб вибирати під кожну функцію оптимальне за ціна/якість устаткування [4].

Majordomo - це фактично софт для створення сервера/мозку вашого Розумного будинку. Основа Majordomo - ядро.

Majordomo побудований на стэке веб-технологій (L/W)AMP -- Linux/Windows (ОС), Apache (веб-сервер), Mysql (база даних), PHP (мова програмування).

Схема загальної архітектури системи зображена на рис. 1.1.

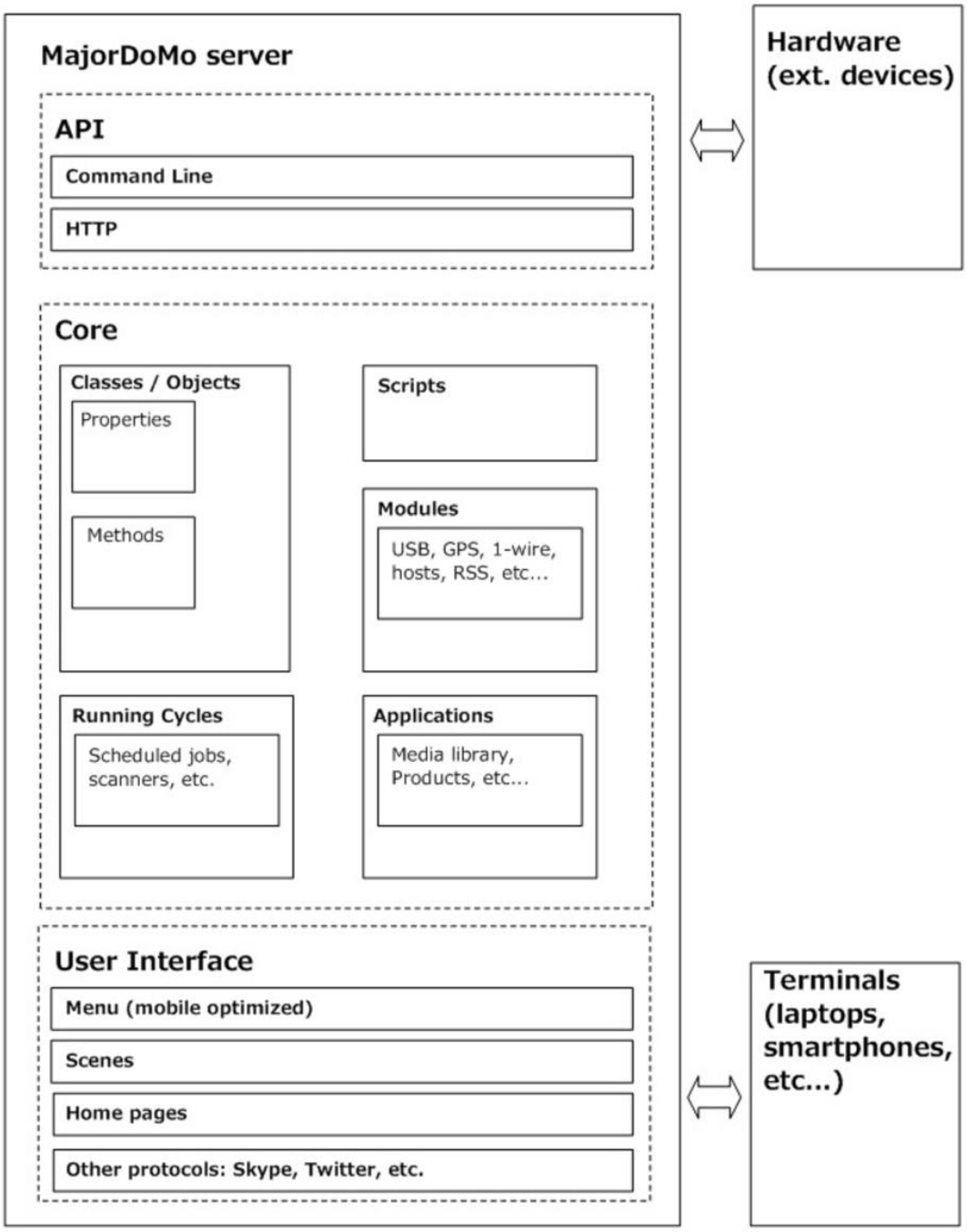

Рисунок 1.1 – Структура системи Majordomo

Основне поняття, що вводиться цифровим будинком «Majordomo» – об'єкт, безліч яких у свою чергу ділиться на різні класи. І кожний з них має свої властивості і методи взаємодії. Приклад, клас «лампочки», у який можуть входити об'єкти «люстра в залі», «підсвічування дзеркала», «світло на кухні». У кожного є властивість «статус» зі значенням «включене» і «виключене», а також метод «запалити» і «погасити».

Об’єкт Majordomo у апаратній формі – це контролер з можливістю зв'язку між керованим пристроєм або датчиком і центральною системою, на якій перебуває програмний комплекс Majordomo.

У якості апаратній складовій виступає як устаткування самостійного складання, що так і випускається деякими виробниками вже в комплекті, наприклад фірмою Xiaomi. «Розумний» будинок Mojordomo підтримує масу протоколів обміну – MQTT, Z-Wave, Broadlink (без вороття стану) і безліч інших.

Пристрою від Xiaomi славляться гарною якістю та оптимальною вартістю, що ставиться також і до тих гаджетів, які становлять екосистему розумного будинку від Xiaomi. Основою даної системи є Xiaomi Mi Smart Home Multifunctional Gateway, який поєднує в одну мережу всі дверні та віконні датчики, так само як різні пристрої, будучи головною сполучною ланкою між користувачем і його розумним будинком.

У систему Mi-Пристроїв для розумного будинку входять датчик руху Mi Smart Home Move Detector, датчики для вікон і дверей Mi Smart Home Window Detector, датчик температури та вологості Mi Smart Home Temerature and Humidifier Detector. Xiaomi не застосовує якусь одну бездротову технологію для своїх пристроїв, а вибирає оптимальну для кожного типу. Наприклад, для керування висвітленням, розетками й шторами використовується Zigbee, і для їх підключення обов'язково потрібний хаб від Xiaomi з підтримкою цього протоколу. Телевізори, пилососи і IP камери підключаються по Wi-Fi через роутер, адже не всім потрібний повноцінний розумний будинок, а Wi-Fi є майже в усіх.

Датчики температури, вологості, якості повітря та замки працюють по Bluetooth. Такі пристрої можна підключити прямо до телефону і тільки переглядати показання, а можна підключити до хабу Xiaomi з підтримкою Bluetooth, тоді з'являється можливість використовувати датчик у сценаріях керування кліматом.

2 ПРОЕКТУВАННЯ СТРУКТУРНОЇ СИСТЕМИ РОЗУМНОГО БУДИНКУ

Незважаючи на недоліки централізованої системи, вона має багато переваг при реалізації сучасного розумного будинку:

1) Уся інформація зберігається у одному місці та усіма пристроями можна керувати з єдиного вузлу. При цьому цей вузол може надавати користувачу загальний інтерфейс доступу до усіх підсистем розумного дому. Це дуже зручно для кінцевого користувача. Йому не потрібно встановлювати декілька програм на свій смартфон для доступу до різних підсистем або відкривати декілька незалежних веб сторінок з інтерфейсами керування та відображення стану цих підсистем.

2) При наявності бази даних на центральному вузлі користувач розумного будинку має доступ не тільки до поточного стану підсистем, але і може переглядати історію показань датчиків, подій, що відбулися у будинку. Це зручна і корисна на практиці можливість.

3) Захищеність усіх систем від несанкціонованого доступу обумовлюється захищеністю центрального вузла і налаштовується одноразово.

4) Налаштування усіх підсистем користувач виконує з використанням єдиного інтерфейса. Тому принципи налаштування, віджети та місце зберігання налаштувань є єдиними для усіх підсистем, що теж є зручним для кінцевого користувача.

Тому обираємо централізований варіант системи розумного будинку. При цьому до центрального вузла висувається низка вимог:

1) Показання усіх датчиків обробляються центральним вузлом.

2) Керування усіма виконавчими пристроями забезпечує центральний вузол.

3) Графічний та web інтерфейси створює центральний вузол.

4) База даних розміщується на центральному вузлі.

5) Голосовий асистент, при його потребі, також розміщується у центральному вузлі.

Для реалізації усіх наведених вище функцій центрального вузла необхідно використовувати пристрій комп’ютерного типу з операційною системою на сучасному процесорі.

Тепер розглянемо перелік підсистем, що повинні буди у складі автоматизованої системи керування будинком:

1) Підсистема керування опаленням приміщень;
2) Підсистема керування кондиціонуванням приміщень;
3) Підсистема керування вентиляцією приміщень;
4) Підсистема керування освітленням будинку та прибудівельної території;
5) Підсистема керування подачею води;
6) Підсистема безпеки;
7) Голосовий асистент.

Розуміючи, що розумний будинок найчастіше є штучним продуктом для конкретної будівлі або квартири та бажань замовника, замовник може забажати тільки одну з підсистем, декілька або усі з облежаним чи повним функціоналом. Тому система повинна бути розширюєма та мати можливість конфігурації та переконфігурації з часом. Цім вимогам відповідає модульний принцип створення програмного забезпечення, що дозволяю додати стільки модулів підсистем, скільки потрібно. Цей принцип дуже вдало поєднується з реалізацією web інтерфейса, кожна сторінка якого може відображати стан однієї з підсистем.

Аналіз сучасних прикладів реалізації розумних будинків дозволив зробити висновок, що найбільш успішними за продажами є недорогі рішення, що легко дозволяють об’єднати різноманітні датчики та виконавчі пристрої різних фірм виробників за різними стандартами передавання даних та різними протоколами. Тому центральний вузол розумного будинку повинен мати як найбільш поширені радіо інтерфейси (WiFi, Bluetooth), так і можливість дротового підключення датчиків чи виконавчих пристроїв.

Для України одним із дуже важливих питань при розробці розумного будинку є можливість відстежувати стан та мати доступ до керування будинку в умовах низької якості цифрового зв'язку, чого не дозволяє web інтерфейс. Тому необхідно реалізувати альтернативний інтерфейс керування, що дозволяє виконувати основні функції в умовах 2G. Аналіз існуючих рішень показує, що найбільш вдалим є поєднання завадостійких протоколів з хмарними технологіями.

За результатами аналізу переваг та недоліків існуючих систем була обрана наступна структура системи розумного будинку зображена на рис. 2.1.

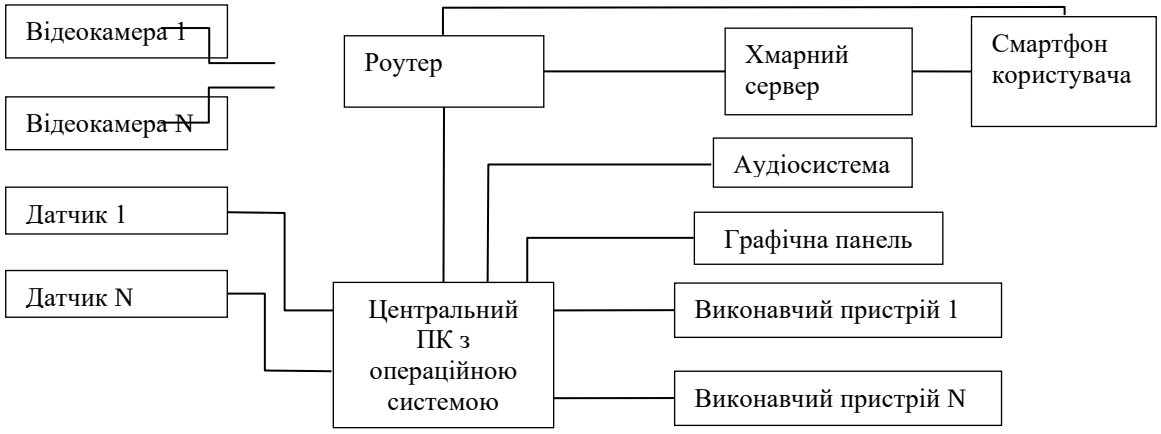

Рисунок 2.1 – Схема електрична структурна проектованої системи розумного будинку

До спроектованої системи підключаються N різноманітних датчиків. Вони необхідні для збирання інформації для усіх підсистем розумного будинку. Якщо ці датчики дротові, то їх кількість буде обмежена наявністю відповідних інтерфейсів та їх можливостями у центрального ПК. Якщо датчики бездротові, то практичних обмежень немає. Аналогічна ситуація з виконавчими пристроями. Відеокамери входять лише до системи безпеки будинку. Найчастіше зараз використовуються IP камери, відеопотік від котрих передається через Ethernet із швидкостями 100Мбіт/с, 1Гбіт/с чи через бездротовий інтерфейс WiFi. Обидва інтерфейса існують у сучасних роутерів, що забезпечують об'єднання потоків відеокамер у загальний потік та його передавання на центральний ПК, що може

обробляти зображення з використанням нейромереж, вбудовування зображення у web сторінку. Нейромережі дозволяють виконати виявлення зображення людини та його розпізнавання, що необхідно у підсистемі безпеки чи при взаємодії з голосовим асистентом.

3 РОЗРАХУНОК ОСНОВНИХ ПАРАМЕТРІВ СИСТЕМИ

Розрахунки почнемо з вибору центрального ПК. У якості центрального ПК будемо використовувати одноплатний ПК Raspberry PI 4. Він має багато переваг у порівнянні з стаціонарним ПК. По-перше для охолодження одноплатних ПК достатньо пасивних систем типа радіаторів, що є однією з головних вимог до автоматизованих систем. По-друге одноплаті ПК мають малі розміри, що дозволяє розмістити їх у невеликому об'ємі. По-третє ціни на одноплатні ПК в порівнянні із стаціонарними ПК невеликі.

На одноплатний ПК Raspberry PI 4 будемо встановлювати операційну систему Raspbian, що є різновидом Linux. Для функціонування операційної системи Raspbian достатньо 256Мбайт оперативної пам'яті та 2Гбайт флеш накопичувача. Програмне забезпечення буде розроблено мовою Python, для інтерпретатора якої потрібно ще понад 1Гбайт флеш пам'яті, для встановлення допоміжних бібліотек з нейронних мереж, розпізнавання та голосового асистенту 4Гбайт.

Розрахуємо об'єм флеш накопичувача для зберігання показань датчиків та подій системи. Показання кожного з датчиків – це 32 розрядне речовинне число. Якщо припустити, що у середньостатистичному житловому приміщенні 7 кімнат, включаючи кухню, ванну, туалет, прихожу і у кожній кімнаті встановлено 3 датчика, то інформація з усіх датчиків – це 84 байта. Також необхідно зберігати мітку дати та часу, що відповідає 8 байтам у базах даних типу MySQL. Усього одне вимірювання $N_{\text{в}} = 92$ байта.

При зберіганні показань датчиків 1 раз у 5 секунду (за такий час значних змін температури, вологості та інших фізичних параметрів не відбувається) та зберіганні показань за інтервал одного місяця буде потрібен об'єм накопичувача.

$$N = \frac{T_{\text{д}} \cdot 24 \cdot 60 \cdot 60}{\text{Т}_{\text{в}}} \cdot N_{\text{в}},$$

де $T_{\text{д}}$ – кількість діб;

$T_в$ – періодичність вимірювань, с.

$$N = \frac{31 \cdot 24 \cdot 60 \cdot 60}{5} \cdot 92 = 49{,}3 \text{ Мбайт}$$

Таким чином об’єму накопичувача в 1Гбайт достатньо для зберігання показань датчиків та подій впродовж року. При об’ємах сучасних накопичувачей 16-32Гбайт потреби розумного дому будуть задовільнені.

Виконувати обробку багатьох відопотоків з використанням одноплатного ПК неможливо. Тому у подібних системах використовуються датчики руху, що фіксують наявність руху, а подальшу обробку зображення вже реалізує однопалатний ПК. Від IP відеокамер відеопотік передається у форматі H.264 або H.265. Передавати потік з частотою кадрів 24кадри/с не має потреби, бо швидкість обробки на Raspberry PI не буде перевищувати 6 кадрів/с. При цьому швидкість потоку даних з відеокамери не перевищує 0,8Мбіт/с.

Raspberry PI має 28 цифрових ліній вводу-виводу, що можна програмувати як входи або виходи. До входів можна підключити датчики типа сухого контакту, а до виходів пристрої, що вмикаються або вимикаються. Також на апаратному рівні Raspberry PI підтримую багато цифрових інтерфейсів 1-Wire, USART, SPI, що дозволяє опитувати багато різноманітних датчиків для вимірювання температури, вологості і т.і. через дротовий інтерфейс (наприклад, датчик вимірювання температури DS18B20).

У цій науковій роботі для подальших досліджень розглянемо реалізацію лише трьох підсистем: безпеки, керування опаленням та освітленням приміщень будинку чи квартири. Також система розумного будинку буде мати 2 інтерфейса користувача: графічний, доступ до якого можливо здійснити з використанням графічної сенсорної панелі та Web-інтерфейс, котрий не потребує додаткових коштів.

Основним інтревейсом за замовчуванням є Web-інтерфейс, бо кожна сучасна людина має смартфон і з його допомогою може відкрити web-сторінку керування своїм розумним будинком. Для реалізації web-додатку на стороні центрального ПК розумного будинку використаємо web-сервер Flask [5]. Він має

достатні можливості для розумного будинку та не висуває значних вимог до ресурсів ПК. Крім смартфону можна використовувати більший за розмірами екрану планшет.

Допоміжним інтерфейсом є графічний. Цей інтерфейс зручний для літніх людей, які не дуже люблять користуватися смартфонами або їм зручніше користуватися більшими за розмірами графічними панелями. Наприклад, можна використати 7-дюймову панель Elecrow з роздільною здатністю 1024x600 пікселів, що легко вбудувати у сучасні корпуси (рис. 3.1).

Рисунок 3.1 – Панель Elecrow

Для реалізації графічних вікон керування розумним будинком використаємо кросплатформену безкоштовну бібліотеку Tkinter, що має повний перелік фіджетів для керування та відображення інформації та є однією з розповсюджених на практиці. Невід'ємною частиною графічних вікон автоматизованих систем керування є статичні та анімовани графіки. Для їх реалізації використаємо безкоштовну бібліотеку Matplotlib, що є частиною інженерного математичного пакету Matlab.

Для захвату відозображення з мережних IP відеокамер використаємо безкоштовну відкриту бібліотеку OpenCV [6]. Для інтеграції обробленого відео зображення у графічне вікно графічного інтерфейса використаємо бібліотеку Pillow.

Для виявлення обличчя людини у зображенні та його подальшої ідентифікації використаємо бібліотекb Dlib та Scipy.

Для реалізації голосового асистента можна використати бібліотеку PyTTSx3, що є зручною кроссплатформенною бібліотекою для реалізації TTS в додатках на Python 3. Вона дозволяє обирати 80 мов, різні голоси, налаштовувати швидкість відтворення повідомлення та гучність.

В підсистемі керування опалювальною системою будемо використовувати дротові датчики температури DS18B20, що дозволяють вимірювати температуру у діапазоні від -55°C до +125°C із похибкою не більше 0,5°C. Діапазон робочих напруг від 3 до 5В, час вимірювання 750мс. Датчики підключаються паралельно до однієї шини інтерфейсу 1-Wire відповідно до рисунку 3.2.

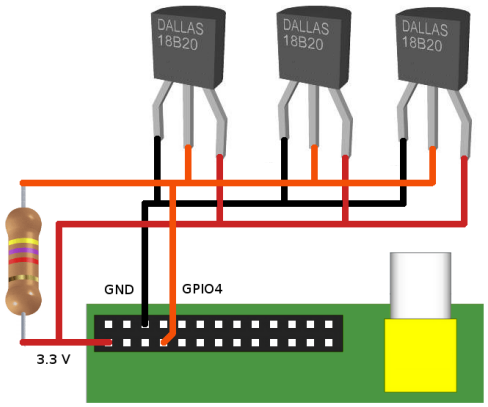

Рисунок 3.2 – Схема підключення датчиків DS18B20 до Raspberry PI

Така схема потребує прокладки мінімальної кількості кабелів у будинку. Для опитування датчиків використаємо безкоштовну бібліотеку w1thermsensor.

Для зберігання результатів вимірювань та подій, що трапляються у системі будемо використовувати базу даних MySQL [7]. В цій базі будуть знаходитися 4 типа таблиць: для зберігання налаштувань системи, Tag Logging, Alarm Logging, Event Logging. Tag Logging – це таблиця результатів вимірювань датчиків із часом, що підключені до системи. Alarm Logging – це таблиця аварійних подій чи подій системи безпеки із мітками часу та тлумаченням події. Event Logging – це таблиця подій у системі, що відбуваються автоматично при вмиканні чи вимиканні пристроїв, а також дії користувача

При реалізації швидкісного інтернету користувач розумного дому буде мати доступ до Web-інтерфейсу. При відсутності швидкісного інтернету користувач буде мати доступ через протокол MQTT. **MQTT** або Message Queue Telemetry Transport – це легкий, компактний і відкритий протокол обміну даними створений для передачі даних на вилучених локаціях, де потрібен невеликий розмір коду і є обмеження по пропускній здатності каналу. Перераховані вище гідності дозволяють застосовувати його в системах M2M (Машинно-Машинна взаємодія) і IoT (Промисловий Інтернет речей). Обмін повідомленнями в протоколі MQTT здійснюється між клієнтом (client), який може бути видавцем або передплатником (publisher/subscriber) повідомлень, і брокером (broker) повідомлень (наприклад, сервер Mosquitto MQTT).

Видавець відправляє дані на MQTT брокер, указуючи в повідомленні певну тему, топік (topic). Передплатники можуть одержувати різні дані від безлічі видавців залежно від передплати на відповідні топіки.

4 РОЗРОБКА АЛГОРИТМІВ ФУНКЦІОНУВАННЯ СИСТЕМИ ТА ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для реалізації організації програмного забезпечення був обраний модульний принцип, де є декілька обов'язкових модулів, а усі інші додаються, якщо вони необхідні. До обов'язкових входять:

1) Модуль web-інтерфейсу, як базовий інтерфейс користувача, котрий існує завжди у будь-якій конфігурації системи.

2) Ядро системи, у якому виконується періодичне опитування усіх дротових та бездротових датчиків, що підключені до системи та передавання команд керування. Опитування датчиків будемо виконувати з періодичністю 1с. Відображення результатів вимірювання та зберігання цих результатів у базу даних будемо здійснювати з періодичністю 5с.

До додаткових модулів відносяться:

1) Модуль реалізації графічного інтерфейса.

2) Модуль керування системою освітлення.

3) Модуль системи безпеки.

Кожний з модулів буде працювати у окремому потоці, що підвищую швидкість роботи на багатоядерному процесорі та дозволяє уникнути несумісності програмної реалізації різних бібліотек (наприклад, графічний та web-інтерфейси не можуть функціонувати у одному потоці).

Алгоритм функціонування web-серверу Flask зображено на рис. 4.1

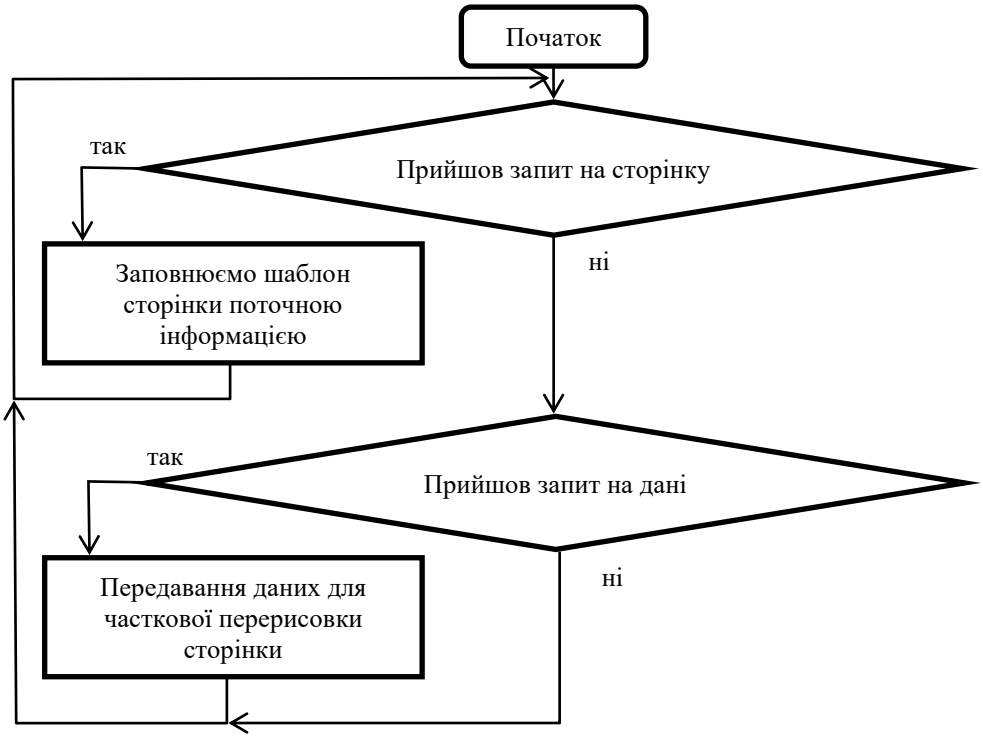

Рисунок 4.1 – Алгоритм функціонування web-серверу Flask

Web-сервер отримує запити від користувацьких браузерів. Запити бувають двох типів: на отримання сторінки та на отримання даних. У першому випадку сервер бере шаблон сторінки, заповнює у відповідних місцях цей шаблон інформацією від датчиків чи внутрішніх змінних та передає цю сторінку клієнту. У програмі, що написана мовою Python необхідно вказати декоратор url-запиту (наприклад, **@app**.route('/main/') та функцію, що буде обробляти цей запит (наприклад, **def** hello(name=**None**):). У цій функції для заповнення шаблону сторінки та її передавання користувачу використовується метод render_template **(**наприклад, **return** render_template('hello.html', name=name)). У цій сторінці помічається змінна, значення якої необхідно заповнити (наприклад, {{name}}).

Коли сторінка вже завантажена у браузері клієнта, у ній починає періодично виконуватися Java-скріпт, що дозволяє оновлювати стан освітлювальної системи, показання датчиків без перезавантаження усієї сторінки і т.і. Для цього формуються відповідні запити до сервера, а сервер формує відповіді, використовуючи клас jsonify [8]. З використанням відповідного методу jsonify

відправляються імена параметрів та їх значення у форматі jsonify({**'ім'я параметра'**: значення параметра}).

Опитування цифрових датчиків температури DS18B20 буде здійснюватись у окремому потоці. Для роботи з потоками використаємо бібліотеку Thread. Кожен з потоків буде містить нескінчений цикл, що забезпечує роботу розумного будинку 24 години на добу. Для автоматичного завершення потоків при закритті основного усі допоміжні потоки будуть демонічними. Роботу ядра ілюструє алгоритм на рисунку 4.2.

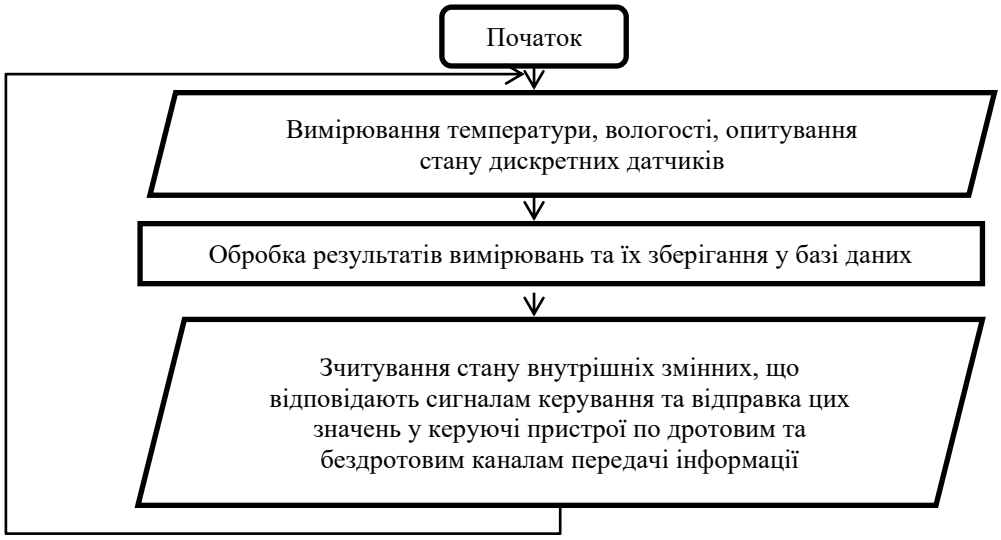

Рисунок 4.2 – Алгоритм роботи ядра

Різні програмні модулі можуть користатися результатами вимірювань чи відображати їх. Тому всі результати передаються у базу даних, а потім з неї зчитуються із інших потоків. Але основний алгоритм керування системою опалення реалізується у цьому потоці безпосередньо за результатами вимірювань.

Для візуалізації стану розумного будинку у окремому потоці на екрані дисплею створюється інтерфейс, що відповідає web-інтерфейсу.

Керування системою освітлення – це вмикання світла, вимикання та встановлення яскравості освітлення для освітлювальних пристроїв з функцією димирування. Керування освітлювальними пристроями виконує лише людина, за винятком ситуації коли світло вмикається за сигналом від датчику руху. Тому із

web чи графічного інтерфейсів керування виконується безпосередньо з фіксацією стану системи у базі даних.

Робота системи безпеки виконується у окремому потоці. У сучасних IP камерах на програмному рівні підтримується rtsp сервер, що видає потік відеокадрів. Захоплення цього потоку реалізується функцією VideoCapture бібліотеки OpenCV. Захоплене відео зображення вбудовується як в web-інтерфейс так і в графічний. Зображення може виводитись як необроблене, так і результат розпізнавання. Система безпеки доповнена дротовими датчиками руху. При появі на виході датчику руху логічної «1» захвачене зображення аналізується на наявність людини за допомогою бібліотеки tensorflow з метою виявлення класу людини на зображенні при використанні алгоритму YOLO та її подальшої ідентифікації з використанням бібліотек Dlib та Sciру.

На рисунку 4.3 наведено вигляд головної web сторінки розумного дому, вона дозволяє перейти до керування відповідними підсистемами.

На рисунку 4.4 наведено приклад web сторінки з відображенням підсистеми освітлення та віджетів керування освітленням. У цьому прикладу відображено освітлення трьох кімнат. У двох кімнатах світло можна тільки вмикати чи вимикати. У спальні можна контролювати інтенсивність освітлення.

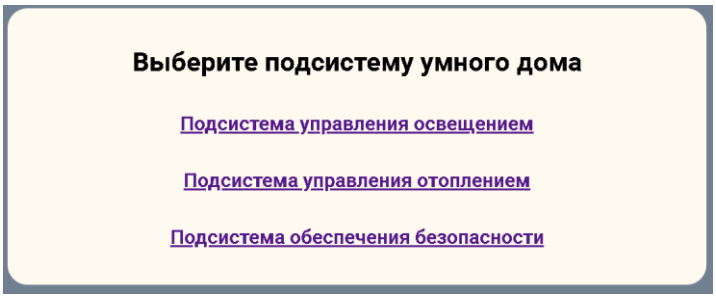

Рисунок 4.3 – Головна сторінка розумного дому

На рисунку 4.5 наведено приклад web сторінки з відображенням підсистеми опалення та віджетів керування порогами вмикання та вимикання опаленням.

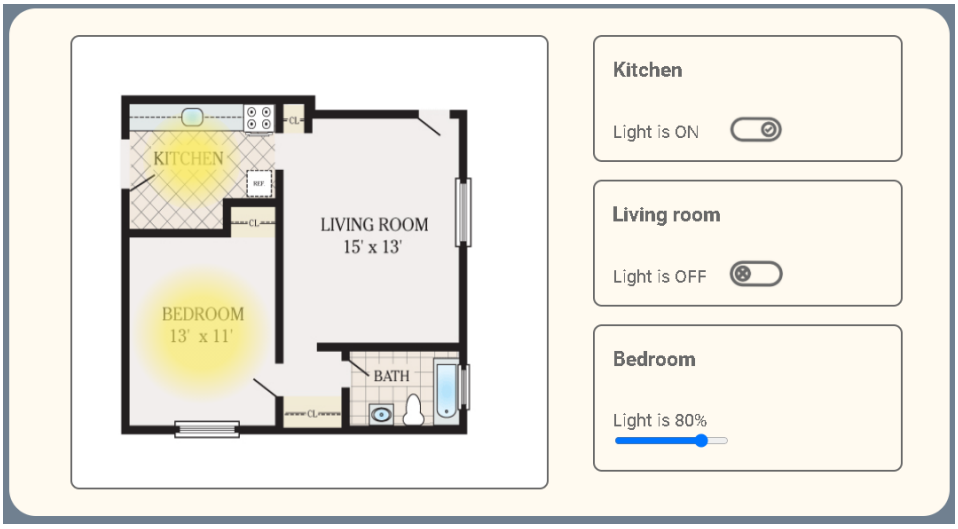

Рисунок 4.4 – Web-сторінка відображення стану підсистеми освітлювання.

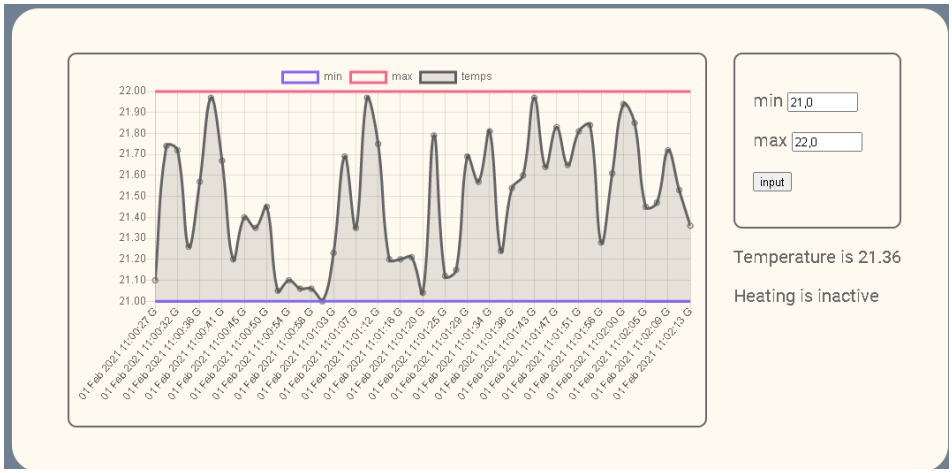

Рисунок 4.5 – Web-сторінка відображення стану підсистеми опалення.

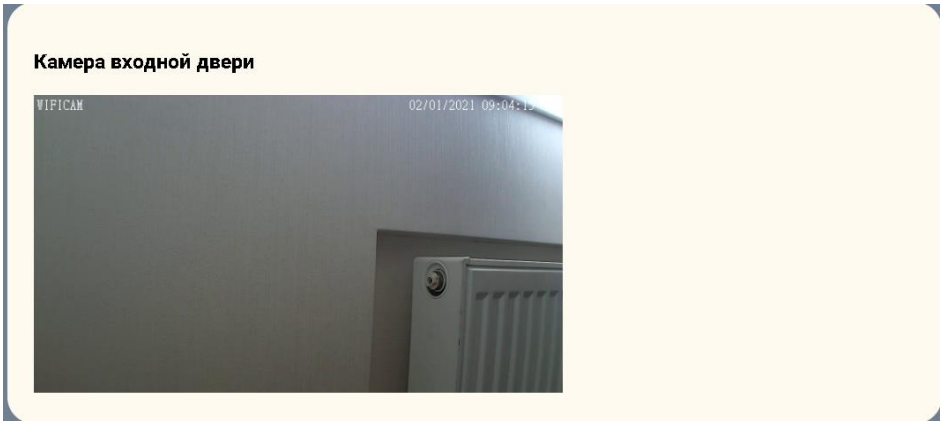

Рисунок 4.6 – Web-сторінка відображення стану підсистеми безпеки.

ВИСНОВКИ

Аналіз існуючих систем розумного дому дозволив зробити висновок, що ринок цих систем постійно розвивається і їх користувачі дуже часто надають перевагу системам індивідуально адаптованим до потреб кінцевого користувача з урахуванням його запитів та особливостей будівлі. Крім того, найбільш вдалими комерційними проектами є ті, що дозволяють об'єднати обладнання різних виробників та дротові і бездротові технології у єдину систему. Для вітчизняного ринку також дуже важливими параметрами є відношення ціна/можливості та робота в умовах низько швидкісних каналів зв'язку. Тому, задача створення сучасної системи керування розумним будинком для вітчизняного ринку, що може конкурувати із закордонними аналогами дуже актуальна. В запропонованій концепції розумного будинку обрано централізований варіант на апаратній платформі Raspberry PI. Обрано модульну концепцію створення програмного забезпечення мовою Python з реалізацією різних підсистем у окремих потоках. Користувачу пропонується два основні інтерфейса керування: web та графічний в умовах швидкісного Internet та спрощений варіант з використанням протоколу MQTT в умовах поганого зв'язку. У якості результату розробки створено алгоритми функціонування, інтерфейси та модулі підсистем керування опаленням, освітленням та підсистемою безпеки.

Таким чином, реалізована концепція є дуже вдалим варіантом для вітчизняного ринку та відповідає побажанням сучасних користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Петин В.В. Создание умного дома на базе Arduino. М: ДМК Пресс. – 2017. – 180 с.

2. Андрей Дементьев. «Умный» дом XXI века. М: Ridero. – 2016, – 142 с.

3. Трофимов В.Б., Кулаков С.М. Интеллектуальные автоматизированные системы управления технологическими объектами. М.: Инфра-Инженерия. — 2017. - 232 с

4. MajorDoMo [Електронний ресурс]: – Режим доступу: https://mjdm.ru/ - 25.01.2021 г.

5. Документация Flask [Електронний ресурс]: – Режим доступу: https://flask-russian-docs.readthedocs.io/ru/latest/quickstart.html - 25.01.2021 г.

6. Opencv [Електронний ресурс]: – Режим доступу: https://opencv.org/ - 25.01.2021 г.

7. MySQL Installer [Електронний ресурс]: – Режим доступу: https://dev.mysql.com/downloads/installer/ - 25.01.2021 г.

8. Chartjs [Електронний ресурс]: – Режим доступу: https://www.chartjs.org/ - 25.01.2021 г.

ДОДАТОК А

Лістинг програми мовою Python

```
# -*- coding: utf-8 -*-
from flask import Flask, render_template, request, jsonify, redirect, Response
import random
import time
from timeloop import Timeloop
from datetime import timedelta
import threading
import configparser
import os
import cv2

import tkinter as tk
from matplotlib import pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.animation import FuncAnimation
import matplotlib.dates as mdates
from datetime import datetime

conf_name = "settings.conf"
conf = configparser.RawConfigParser()

tl = Timeloop()

M1, M2 = 21, 22 # temp from to
N1, N2 = 22.6, 23.7 # temp min max
temp = 21.5
heating = False

GEN_INT = 2.2 # time in ms for temp regen
app = Flask(__name__)

rooms = [
        {
            'name': "Kitchen",
            'light': True
        },
        {
            'name': "Living room",
            'light': False
        },
        {
            'name': "Bedroom",
            'light': 80
        }
]

# GUI setup ########

X = 910
Y = 600

T1 = 500 # time in ms for tkinter animation
N = 50 # ammount of saved records


time_buffer = []
temp_buffer = []

heater_state = None
ax = None
min_spin = None
max_spin = None
```

```
xfmt = None

camera = cv2.VideoCapture('rtsp://admin:Ad303min@192.168.0.16:10554/tcp/av0_0')  # use 0 for web camera
#  for cctv camera use
rtsp://username:password@ip_address:554/user=username_password='password'_channel=channel_number_stream
=0.sdp' instead of camera
# for local webcam use cv2.VideoCapture(0)

def gen_frames():  # generate frame by frame from camera
    while True:
        # Capture frame-by-frame
        success, frame = camera.read()  # read the camera frame
        if not success:
            break
        else:
            ret, buffer = cv2.imencode('.jpg', frame)
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')  # concat frame one by one and show result


@app.route('/video_feed')
def video_feed():
    #Video streaming route. Put this in the src attribute of an img tag
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

def write_conf():
    conf.set('temp', 'min', N1)
    conf.set('temp', 'max', N2)
    with open(conf_name, "w") as config:
        conf.write(config)

def read_conf():
    set_min(conf.get('temp', 'min'))
    set_max(conf.get('temp', 'max'))

def upd_heat():
    global heating
    if temp < N1:
        heating = True
        if heater_state:
            heater_state.config(text="Система нагрева работает.")
    elif temp > N2:
        heating = False
        if heater_state:
            heater_state.config(text="Система нагрева отключена.")

def set_min(value):
    global N1
    if float(value) < N2:
        N1 = float(value)
        upd_heat()

def set_max(value):
    global N2
    if float(value) > N1:
        N2 = float(value)
        upd_heat()

def gen_temp():
    global temp
    temp = round(random.uniform(M1, M2), 2)
```

```
    time_buffer.append(datetime.today())
    temp_buffer.append(temp)
    upd_heat()
    while len(temp_buffer) >= N:
        temp_buffer.pop(0)
        time_buffer.pop(0)

# Web funcs ########

@tl.job(interval=timedelta(seconds=GEN_INT))
def upd_temp():
    gen_temp()

@app.route('/', methods=['GET', 'POST'])
def main_page():
    return render_template("index0.html")

@app.route('/Hating', methods=['GET', 'POST'])
def Hating():
    return render_template("index2.html", temp=temp, heating=heating, N1=N1, N2=N2)

@app.route('/Secur', methods=['GET', 'POST'])
def Secur():
    return render_template("index3.html")

@app.route('/Light', methods=['GET', 'POST'])
def hello_world():
    global volume
    global rooms
    if request.method == 'POST':
        if request.form.get('slide'):
            volume = request.form.get('slide')
            rooms[-1]['light'] = volume
            print("Light in " + rooms[-1]["name"] + " set to " + volume + "%")
            return jsonify({'volume': volume})
    return render_template("index1.html", rooms=enumerate(rooms))

@app.route('/<roomIndex>/<deviceName>/<action>', methods=['GET', 'POST'])
def do(roomIndex, deviceName, action):
    global rooms
    rooms[int(roomIndex)][deviceName] = (action == 'on')
    print("Light in " + rooms[int(roomIndex)]["name"] + " set to " + ("ON" if action == 'on' else "OFF"))
    return redirect("/Light")

@app.route('/getTemp')
def temp_info():
    x = [temp, heating]
    return jsonify(x)

@app.route('/syncLimits')
def sync_limits():
    x = [N1, N2]
    return jsonify(x)

@app.route('/getChartData')
def chart_info():
    x = [N1, N2, time_buffer, temp_buffer]
    return jsonify(x)

@app.route('/setLimits', methods=['GET', 'POST'])
def limits():
    global N1
```

```
    global N2
    if request.method == 'POST':
        set_min(request.form.get('min_lim'))
        set_max(request.form.get('max_lim'))
        write_conf()
    return redirect('/Hating')


# GUI funcs ########

def plt_create():
    plt.title('Зависимость …')
    plt.xlabel('Время')
    plt.ylabel('Показания датчика')
    plt.setp(ax.get_xticklabels(), rotation = 15)
    ax.xaxis_date()
    ax.xaxis.set_major_formatter(xfmt)

def fig_init():
    tb = temp_buffer
    x = mdates.date2num(time_buffer)
    plt.plot(x, [N1]*len(tb), 'r', x, tb, 'b', x, [N2]*len(tb), 'g')

def fig_update(i):
    ax.clear()
    plt_create()
    tb = temp_buffer
    x = mdates.date2num(time_buffer)
    plt.plot(x, [N1]*len(tb), 'r', x, tb, 'b', x, [N2]*len(tb), 'g')


conf.read(conf_name)
if not os.path.exists(conf_name):
    with open(conf_name, 'w'): pass
    conf.add_section('temp')
    write_conf()
else:
    read_conf()


def tk_apply():
    set_min(min_spin.get())
    set_max(max_spin.get())
    write_conf()


def run_tk():
    global heater_state
    global ax
    global min_spin
    global max_spin
    global xfmt

    global conn_tk
    global cursor_tk

    root = tk.Tk()
    root.title("Система управления отоплением")
    root.geometry(f"{X}x{Y}")

    fig = plt.figure(figsize=(7, 6), dpi=80)
    ax = fig.add_subplot(111)
```

```
    xfmt = mdates.DateFormatter('%Y.%m.%d %H:%M:%S')

    canvas = FigureCanvasTkAgg(fig, root)
    heater_state = tk.Label(root, text="Система нагрева работает." if heating else "Система нагрева отключена.")
    min_spin = tk.Spinbox(root, width=5, from_=0, to=40, format='%.1f', increment=0.1, textvariable=tk.StringVar(root, N1))
    max_spin = tk.Spinbox(root, width=5, from_=0, to=40, format='%.1f', increment=0.1, textvariable=tk.StringVar(root, N2))
    apply_btn = tk.Button(root, text="Apply")

    plt_create()

    apply_btn.config(command=tk_apply)

    canvas.get_tk_widget().place(x=0, y=0)
    max_spin.place(x=600, y=160)
    min_spin.place(x=600, y=300)
    heater_state.place(x=200, y=530)
    apply_btn.place(x=600, y=420)

    anim = FuncAnimation(fig, fig_update, init_func=fig_init, interval=T1)

    root.mainloop()


if __name__ == '__main__':
    th = threading.Thread(target=tl.start)
    th2 = threading.Thread(target=run_tk)
    th.start()
    th2.start()
    app.run(debug=True, use_reloader=False)
    th.join()
    th2.join()
```