МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Клієнт-серверна система керування мікрокліматом розумного будинку

Освітній ступінь "Магістр"
Спеціальність 122 "Комп'ютерні науки"

Науковий керівник роботи: _____ (підпис) О.І.Рязанцев (ініціали, прізвище)

Консультант з охорони праці: _____ (підпис) Я.О.Критська (ініціали, прізвище)

Студент: _____ (підпис) О.І.Черножуков (ініціали, прізвище)

Група: КН-17дм

Сєвєродонецьк 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітній ступінь магістр
Напрям підготовки _____
(шифр і назва)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Кашкарову Володимиру Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Клієнт-серверна система керування мікрокліматом розумного будинку

керівник проекту (роботи) Рязанцев Олександр Іванович, д.т.н., проф.
(прізвище, м.'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «18» 10 2018 р. № 221/48
2. Строк подання студентом роботи 10.01.2018
3. Вихідні дані до роботи Матеріали науково-дослідної практики, принципи роботи системи розумного дому, клієнт-серверна архітектура, Android додаток

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Огляд систем розумного будинку , дослідження автоматизації системи керування мікроклімату розумного дому, дослідження сучасних архітектур розробки мобільних додатків для ос android, розробка системи керування розумним будинком, охорона праці та безпека в надзвичайних ситуаціях, висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях	Критська Я.О. ст. викл. кафедри КНІ		

7. Дата видачі завдання 18.10.2018

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Розробка технічного завдання	10.09.2018-15.09.2018	
2	Огляд існуючих систем розумного будинку	16.09.2018-22.09.2018	
3	Аналіз архітектури розумного будинку	26.09.2018-06.10.2018	
4	Дослідження засобів зв'язку між елементами та методів розробки мобільного	07.10.2018-25.10.2018	
5	Проектування системи керування	25.10.2018-25.11.2018	
6	Розробка частини проекту "Охорона праці та безпеки в надзвичайних ситуаціях"	26.11.2018-1.12.2018	
7	Оформлення пояснювальної записки, автореферату та презентації	2.12.2018-09.01.2019	

Студент _____ Черножуков О.І.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Рязанцев О.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Черножуков О.І. Клієнт-серверна система керування мікрокліматом розумного будинку.

Метою магістерської атестаційної роботи є дослідження принципів побудови та автоматизації системи керування розумного будинку на базі системи з мобільним додатком в якості клієнта. У першому розділі проведено аналіз систем розумного дому та її складових. Другий розділ присвячений дослідженню алгоритмів автоматизації систем керування розумним будинком. У третьому відбувається дослідження сучасних архітектурних рішень для розробки Android додатків. У четвертому розділі була виконана реалізація клієнт-серверної системи розумного будинку.

Ключові слова: інтернет речей, розумний будинок, android, mvvm, mvp, беспровідні мережі, zigbee, wifi, bluetooth.

ABSTRACT

Chernozukov O.I. Client-server microclimate control system of a smart home.

The purpose of master's attestation work is to study the principles of construction and automation of a smart home-based management system based on a system with a mobile application as a client. In the first section, an analysis of the systems of the house and its components are carried out. The second section is devoted to research of algorithms of automation of control systems Smart home. One third is the study of modern architectural solutions for the development of Android Applications. In the fourth section the implementation of client-server system of a smart home was implemented.

Keywords: network internet, smart home, android, mvvm, mvp, wireless network, zigbee, wifi, bluetooth.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IoT — Internet of Things — інтернет речей

ZigBee — специфікація мережевих протоколів верхнього рівня

Wi-Fi — Wireless Fidelity

WPAN —Wireless Personal Area Networks

WLAN — Wireless Local Area Network

IEEE — Institute of Electrical and Electronics Engineers — Інститут інженерів електротехніки та електроніки

АСУБ — Автоматизовані системи управління будівлею

X10 — Міжнародний відкритий промисловий стандарт, застосовуваний для зв'язку електронних пристроїв в системах домашньої автоматизації

MVVM — Model-View-ViewModel

REST — Representational State Transfer — передача стану представлення

CRUD — Акронім, що позначає чотири базові функції, які використовуються при роботі з персистентного сховищами даних: створення, читання, редагування, видалення

IDE — Інтегроване середовище розробки

API — Прикладний програмний інтерфейс

ВСТУП

Наразі в світі набуває актуальності тенденція де більшість повсякденних завдань спрощені або автоматизовані і з кожним роком ця тенденція зростає. У побут сучасної людини щільно увійшли технології віддаленого управління. Ці технології допомагають не тільки економити час, а й дозволяють не залежати від місцезнаходження. Як приклади можна привести ситуацію — людина на роботі і не пам'ятає, чи вимкнено світло у ванній, або закриті вхідні двері. В цьому допоможуть автоматизовані системи.

Зростання популярності автоматизованих систем, таких як розумний будинок, обумовлено прагненням людини до комфорту і зручності. Додатковою привабливістю є безпека, будь то протипожежна система або сигналізація з дистанційним оповіщенням. Розумний будинок є сучасним інструментом підвищення рівня комфорту і життя, так як частина процесів відбувається автоматично, а решті можна керувати віддалено, що робить її актуальною для вивчення і вдосконалення.

У класичних автоматизованих системах управління будівлею (АСУБ), що є основою для систем розумного будинку, виділяються три рівні автоматизації:

– рівень диспетчеризації і адміністрування, на якому здійснюється взаємодія персоналу з системою, і збір статистичної інформації;

– рівень автоматичного управління, на якому реалізується автоматизація процесів в різних інженерних системах будівлі. Включає в себе контролери та комутаційне обладнання;

– рівень кінцевих пристроїв, який включає в себе датчики, виконавчі пристрої і безпосередньо фізичні з'єднання між компонентами.

В системі розумного будинку відбувається розширення рівня автоматизованого управління і зниження ролі рівня диспетчеризації за рахунок можливості автономного прийняття рішень.

Метою даної роботи є розробка системи управління "розумним будинком", що дозволяє переглядати показання з датчиків, управляти їх станом, а також автоматизувати питання освітлення та мікроклімату. Система передбачає управління з мобільного додатку на операційній системі Android.

Завдання, які були вирішені в рамках дипломної роботи:

– аналіз існуючих систем і готових рішень. Розробка базового прототипу;

– дослідження архітектури та базових алгоритмів систем управління розумними будинками;

– проектування і розробка системи.

1 ОГЛЯД СИСТЕМ РОЗУМНОГО БУДИНКУ

1.1 Історія розвитку

У всі часи люди прагнули зробити своє житло комфортним для проживання. З розвитком технологій з'являлися нові і все більш досконалі пристрої, які підвищують рівень зручності та безпеки проживання в будинках. З плином часу окремі пристрої стали об'єднуватися в системи, і на сьогоднішній день системи домашньої автоматизації представляють собою складну сукупність передових технологій і сучасних систем управління.

Перші «розумні будинки» з'явилися в США, ще в 50-ті роки минулого сторіччя. На той момент це були дійсно унікальні квартири, обладнані спеціальною електронікою, яка керувала та слідкувала за багатьма речами в будинку, наприклад за пральними машинами, телевізорами, мікрохвильовими печами і т.д. Всі ці побутові прилади були об'єднані в одне ціле, і управлялися з одного пульта, при цьому була можливість контролювати відключення, включення і деякі інші особливості роботи. Однак такі пульти назвати системами управління домашньої автоматизації ще не можна. З часом в країні стали з'являтися інтелектуальні будівлі, які вже були повністю обладнані різної автоматикою, об'єднаної в єдину мережу. У цей час розвиток стало бурхливим, дослідники та розробники стали приділяти особливу увагу не тільки комфортабельності, але і безпеці, а також економії ресурсів завдяки системі «Розумний будинок»

Визначення «розумний дім» (англ. Smart house) виникло на початку 70-х років XX століття в інституті «Інтелектуальних будівель» і означало будівля, яка забезпечує продуктивне й ефективне використання робочого простору. Повноцінним попередником сучасних розумних будинків є система, розроблена в 1978 році двома американськими компаніями X10 USA і Leviton. Цими компаніями була розроблена революційна технологія управління побутовими приладами через звичайну електричну мережу під назвою X10.

X10 використовує для передачі сигналів «пакети» коливань на частоті 120 кГц тривалістю 1 мс. Передача сигналу відбувається в момент, коли напруга в колі дорівнює нулю. При досягненні напруги в мережі нульового значення, приймач сигналу x10 (наприклад, вбудований в патрон лампочки) очікує сигналу в мережі протягом 6 мс. Якщо в цей час передавач сигналу x10 посилає пакет даних, приймач сприймає його як двійкову одиницю. Якщо пакетів не відправлялося, то приймач сприймає це як нуль. Кожен пристрій в даному протоколі має свою адресу, що складається з коду будинку (латинська

буква від A до P) і коду пристрою (числа від 1 до 16). Для управління одним пристроєм необхідно передати: стартовий код, адресу і команду, все це займає 11 циклів змінного струму. Тому протокол x10 має низку недоліків: низька швидкість передачі даних, дозволяє передавати тільки шість команд, що управляють можливість передачі тільки однієї команди в один проміжок часу і вплив пристроїв захисного відключення на якість сигналу, що передається.

Ще одним недоліком X10 було те, що він працював тільки в мережах з напругою в 110 В і частотою 60 Гц, тому поширення систем домашньої автоматизації на основі даного протоколу відбулося тільки на території Сполучених Штатів Америки.

Незважаючи на всі недоліки, даний комунікаційний протокол дозволив реалізувати такі функції як: автоматичне відкривання дверей, включення освітлення бавовною рук і інше, що спричинило подальший розвиток систем домашньої автоматизації.

З метою стандартизації протоколів зв'язку і управління розробниками систем домашньої автоматизації був створений Альянс Електронної Промисловості (Electronic Industries Alliance). Створення альянсу спричинило появу першої стандартизованої шини побутової електроніки (Consumer Electronic Bus, CEBus). CEBus — це стандарт двостороннього обміну даних між пристроями споживчої електроніки (телевізори, відеомагнітофони, світильники і т.д.). Даний протокол зв'язку для передачі керуючих сигналів дозволяє використовувати кручену пару (9600 Байт/с), силові лінії (1200/2400 Біт/с, спочатку тільки для 230 В, 50 Гц), радіочастоти (з 1998 року) і інфрачервоне випромінювання (з 1999 року). Створення універсального комунікаційного протоколу здатного передавати сигнали не тільки по дротах електромережі, а й через коаксіальний кабель, виту пару і радіохвиль, призвело до значного розвитку систем домашньої автоматизації.

В середині 90-х років XX століття в Європі з'являється ще одне об'єднання компаній під назвою EIBA (European Installation Bus Association), до якого увійшли такі великі компанії як Siemens, Gira, ABB, Berker, Jung і ін. EIBA створило свій універсальний протокол EIB (European Installation Bus). Даний протокол набув широкого поширення на пристроях автоматизації, що використовуються на території Європи, і вже до 2000 року більше 80% ринку інсталяційних пристроїв використовували протокол EIB. У травні 1999 відбулося об'єднання трьох європейських асоціацій автоматизації будівель в одну, яка з часом отримала остаточну назву "Асоціація KNX". Завдяки такому об'єднанню відбулося злиття трьох технологій передачі даних: EIB, EHS (European Home System) і Batibus. Об'єднана технологія в кінці 2003 року була затверджена як європейський стандарт

EN50090, а в 2006 році стала міжнародним стандартом ISO/IEC 14543, на якому базуються багато систем домашньої автоматизації.

Сучасні системи «Розумний будинок» будуються навколо мікроконтролера, який здійснює управління всіма системами в залежності від заданих користувачем налаштувань і зовнішніх чинників. «Розумний будинок» управляє кліматом, освітленням, безпекою, а також забезпечує бездротовий зв'язок з користувачем за допомогою Інтернету або мобільних мереж. Схематичне зображеня сучасного розумного будинку можна побачити на рисунку 1.1. Все це стало можливим завдяки масовому розвитку технологій інтелектуальних систем управління будинком, протоколів обміну даними, програмного забезпечення, мобільних пристроїв, швидкісного Інтернету.

Рисунок 1.1 — Схематичне зображения розмного будинку

На сьогоднішній день ринок систем «розумних будинків» є одним з найбільш швидкозростаючих по всьому світу. Головна мета сучасних виробників — створення універсальних систем домашньої автоматизації, доступних для споживача класу, які можна впровадити як в невелику квартиру, так і в заміський будинок. Далі розглянемо кілька прикладів діючих проектів, які пропонуються компаніями на сучасному ринку. Кожна з розглянутих компаній має в своєму наборі кілька варіантів системи в залежності від необхідної функціональності і вартості.

1.2 Концепції розумного будинку

«Розумний будинок» є сукупністю стандартів об'єднаних з різного роду приладами в систему і інтеграцію декількох систем в єдину систему управління будівлею. Системи бувають такі:

– система мікроклімату (опалення, вентиляція кондиціонування, зволоження). Системи опалення, кондиціонування, вентиляції та зволоження в розумному будинку працюють в єдиних режимах, що дозволяють встановлювати певну температуру в залежності від часу доби і погоди за вікном;

– система безпеки (охоронна, пожежна, система доступу, контроль витоків газу, відеоспостереження). «Розумний Дім» може мати повнофункціональну систему забезпечення безпеки, що включає в себе систему охоронної, пожежної безпеки, відеоспостереження, контролю доступу та інші системи, дозволяє здійснювати моніторинг всякого роду погроз для власника будинку;

– система електроживлення (резервні системи, контроль перевантаження електромережі, система освітлення). Система працює безперебійно. Відключення електроенергії не позначиться на функціонуванні розумного будинку і роботі техніки. При скороченні енергії в резервному джерелі живлення будуть сповіщені сервісні служби. Розподіл світла всередині приміщень буде визначатися встановленим режимом, часом доби або кількістю людей, що знаходяться в розумному будинку;

– система зв'язку (телефон, локальна мережа, SMS оповіщення). Системи «Розумний будинок» можуть надавати можливість отримувати сповіщення через мобільний зв'язок;

– система віддаленого управління. Система «Розумний будинок» надає можливість управління в режимі реального часу за допомогою будь-якого мобільного пристрою або ПК, розташованого в локальній мережі або має доступ в Інтернет.

1.3 Огляд стандартних реалізацій системи

Термін «Розумний будинок» не має чіткого визначення, а тому під нього підпадає будь-яка система з автоматизованим керуванням приладами, яка спрощує життя людини

та підвищує рівень його комфорту. Через нечіткі рамки, виникла багато реалізацій з різним рівнем інтеграції та принципом роботи. Їх можна умовно поділити на три групи:

- вбудовані системи з центральним контролером;
- вбудовані системи без центрального контролера;
- системи з настроюваної інтеграцією.

Перша група є повністю налаштованою і встановленою виробником системою, яка управляється центральним контролером і не передбачає прямої взаємодії своїх компонентів між собою. Всі призначені для користувача налаштування зберігаються на центральному пристрої (сервері), а периферія лише виконує отримані від нього інструкції і часто не має вбудованої пам'яті і обчислювальних потужностей. Принцип роботи наведено на рис. 1.2.

Рисунок 1.2 — Діаграма, що представляє принцип роботи вбудованної системи

Друга група є системою з напівавтономних пристроїв. Алгоритми взаємодії прописуються з програми контролера безпосередньо в пам'ять кожного пристрою і для їх зміни пристрій буде необхідно перепрограмувати. У зв'язку з відсутністю центрального компонента, зв'язки між приладами встановлюються безпосередньо і є можливість створення автономних груп, замкнутих один на одного. Принцип роботи наведено на рис. 1.3.

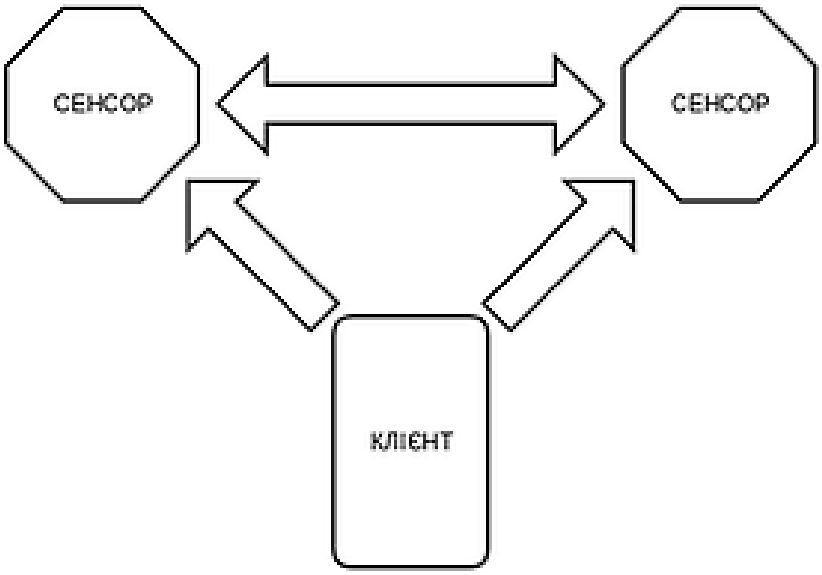


Рисунок 1.3 — Діаграма, що представляє принцип роботи вбудованної системи без центрального контроллера

Третя група — це зовнішні контролери, які приєднуються до звичайних приладів і залежно від показань своїх сенсорів і вбудованого алгоритму регулюють його роботу. Можуть мати центральний контролер, але часто 17 управляються і налаштовуються з інтернет- або хмарного серісу. Функціонують здебільшого як незалежні модулі і для налаштування прямого зв'язку можуть знадобитися додаткові датчики/сенсори. Принцип роботи наведено на рис. 1.4.

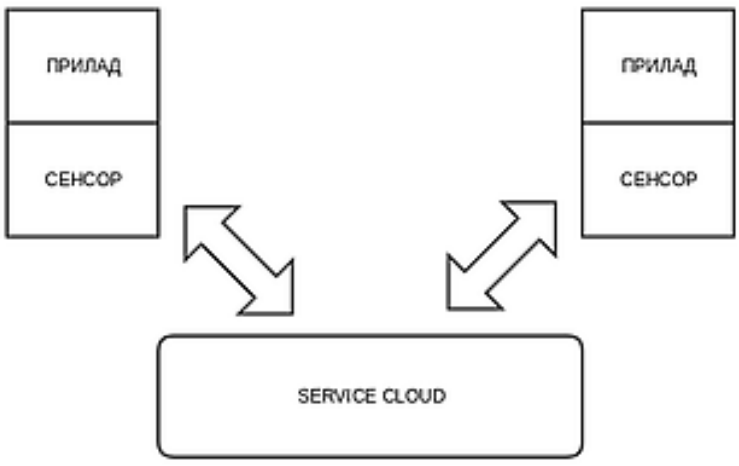


Рисунок 1.4 — Діаграма, що представляє принцип роботи системи з настроюваною інтеграцією

1.4 Аналіз обчислювальної частини системи розумного дому

Обчислювальна частина системи «Розумний Дім» представлена мікрокомп'ютером, що приймає сигнали датчиків і сенсорів, і реагує на них відповідним чином, шляхом передачі сигналів виконуючим модулів. Так, наприклад, при отриманні мікрокомп'ютером сигналу від датчика витоку води, він повинен передати команду микроконтроллеру-виконавцю системи водопостачання про перекриття надходження води.

Серед усіх мікрокомп’ютерів слід виділити два найбільш популярних варіанта — Raspberry Pi та Arduino. Обидва пристрої були винайдені в європейських країнах. Rasbperry Pi розроблений Ебен Аптоном в Великобританії, а Arduino Масімо Банзая в Італії. Обидва вони призначалися для навчання студентів. Raspberry вперше став доступний в 2012 році, тоді як Arduino в 2005. Щоб виконати порівняння Arduino та Rasbperry Pi, розглянемо переваги і недоліки обох платформ. Спочатку виконаємо аналіз переваг Arduino(рис 1.5).

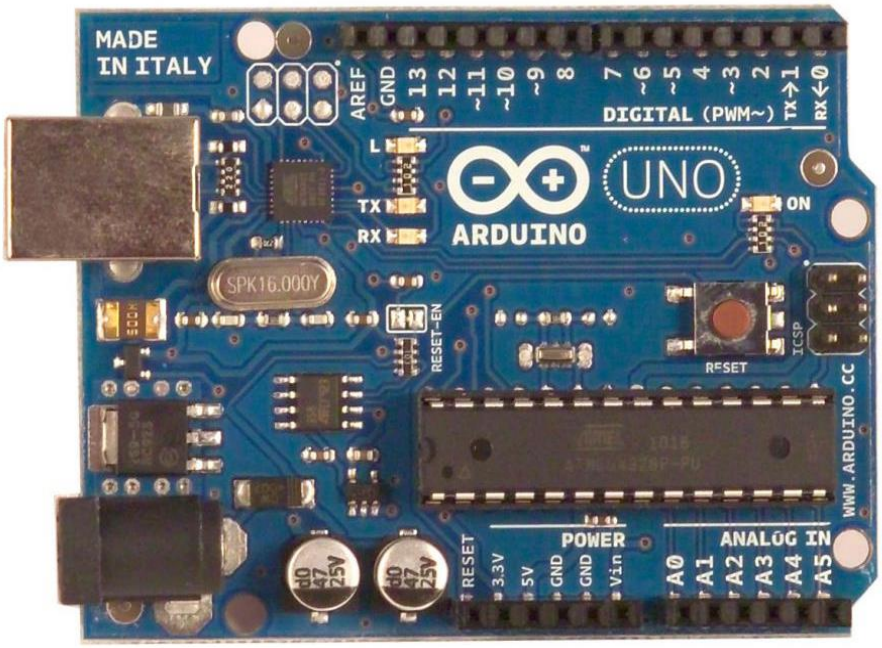


Рисунок 1.5 — Мікроконтролер Arduino UNO R3

За допомогою Arduino дуже просто взаємодіяти з аналоговими датчиками, двигунами і іншими електронними компонентами. Для управління ними досить кількох рядків коду. У той час як для Raspberry доведеться встановити безліч бібліотек і виконувати різні настройки для того, щоб управляти тими ж датчиками. Програмування Arduino простіше, а в Rasbperry потрібні знання Linux.

Rasbperry працює на операційній системі, тому його потрібно правильно вимикати. В іншому випадку додатки можуть бути пошкоджені. Arduino досить просто підключити до мережі. Його можна ввімкнути або вимкнути в будь-який момент.

Raspberry Pi — це потужне апаратне забезпечення, воно потребує постійного харчуванні від блоку з напругою 5 вольт. Комп'ютер важко змусити працювати від звичайної батареї. Arduino споживає набагато менше енергії і може харчуватися від блоку живлення.

Як можна помітити Arduino дешевше за Raspberry Pi. Arduino можна купити за $ 10-20 в залежності від версії. У той час ціна на Raspberry становить близько $ 40-50.

Далі розглянемо переваги Raspberry Pi над Arduino. З огляду на всі переваги Arduino, можна подумати, що це дійсно краще рішення. Але це повністю залежить від вашого проекту. Особливості Raspberry Pi(рис 1.6) — це його потужність і можливості.

Рисунок 1.6 — Мікрокомп’ютер Raspberry Pi 3 model B

Продуктивність — це головна перевага Raspberry Pi, він здатний виконувати кілька завдань одночасно, як звичайний комп'ютер. Якщо вам потрібно побудувати складний проект, наприклад, робот або проект, де ви хочете керувати всім через веб-інтерфейс, то кращим вибором для вас буде Raspberry. Ви можете зробити з нього веб-сервер, сервер VPN, сервер баз даних, сервер друку. Arduino хороший, якщо ви хочете блимати

світлодіодом, але якщо у вас сотні світлодіодів, і ними потрібно управляти через веб-інтерфейс, то краще використовувати Raspberry.

Raspberry Pi в 40 разів швидше ніж Arduino. За допомогою нього ви можете переглядати пошту, слухати музику, дивитися відео і навіть працювати в інтернет.

Raspberry Pi має вбудований порт Ethernet і бездротовий адаптер Wifi, через які ви можете підключити пристрій до інтернету або створити бездротову точку доступу. З огляду на, що мережеві можливості підтримуються операційною системою, то налаштувати мережу дуже просто. Підключити Arduino до мережі буде дуже складно. Всі апаратні засоби потрібно правильно підключити і написати спеціальний код для управління ними.

Для роботи з Arduino необхідно добре розбиратися в електроніці і знати вбудовані низькорівневі мови програмування. Але для роботи з Raspberry Pi необхідні тільки базові знання основних компонентів. Оскільки тут для роботи того чи іншого механізму досить підключити дроти, а для управління можна використовувати безліч вже готових програм.

Операційна система Raspberry Pi і всі файли знаходиться на SD карті, а тому ви можете її дуже просто витягти і перенести все на інший пристрій. Так що в цьому плані Raspberry буде краще.

В результаті проведених досліджень, можна прийти до висновку, що для проекту варто вибрати Arduino якщо:

– проект більш стосується електроніки або ви новачок і хочете в ній краще розібратися;

– проект дуже простий і йому не потрібна мережа;

– проекту не потрібно багато програмного коду.

Варто вибрати Raspberry Pi якщо:

– проект дуже складний і йому необхідна мережа;

– проект схожий на додаток, наприклад, веб-сервер або VPN сервер.

Для проекту розумного дому з центральним керуючим пристроєм варто обрати Raspberry Pi, за рахунок не великої складності використання, та наявності операційної системи розробка системи буде прискорена в декілька раз.

1.5 Аналіз протоколів бездротового зв’язку у системах розумного дому

Існує безліч протоколів, що використовуються в сфері автоматизації управління будівлями. На даний момент не існує загальноприйнятих стандартів для організації мережевої взаємодії пристроїв, що складають систему розумного будинку. Застосування з цією метою технологій побудови локальних обчислювальних мереж є малоперспективним в зв'язку з їх надмірністю.

Технології, що застосовуються в системах розумного будинку, повинні відповідати таким критеріям: низьке енергоспоживання; висока надійність і безпеку передачі; низька вартість; простота фізичного розміщення. Окремо можна відзначити відсутність необхідності в високих швидкостях передачі даних у багатьох сценаріях використання систем.

При порівнянні дротових і бездротових мереж вирішальним фактором є простота фізичного розміщення мережевих пристроїв. Існує сімейство конкурентоспроможних технологій провідних мереж, які називаються Power Line Communication (PLC). Дані технології покладаються на той факт, що приміщення, в яких розгортається мережа, як правило 12 електрифіковані.

Таким чином, можлива побудова як дротових, так і бездротових мереж із застосуванням ряду технологій таких, як X10, INSTEON, HomePlug, Lonworks для забезпечення зв'язку по проводах електричної мережі, і Bluetooth, Wi-Fi, ZigBee для забезпечення бездротового зв'язку.

Монтаж структурованих кабельних систем — досить складний і дорогий процес, тому передача даних від користувача і датчиків до системи здійснюється за допомогою бездротових технологій.

Останнім часом все більш інтенсивно в розвинених країнах світу ведеться робота зі створення нових високотехнологічних радіопристроїв малого радіусу дії SRD (Short Range Devices). Такі пристрої знайшли широке застосування в різних пристроях передачі даних, системах збору телеметричної інформації, в системах виявлення, охорони і безпеки, великому числі інших пристроїв різного призначення.

Використання такої мережі в промислових системах автоматизації дозволяє не тільки отримати інформацію про функціонування всіх її підсистем, а й дає можливість захищеного управління технічними процесами на виробництві.

Використання такої мережі в промислових системах автоматизації дозволяє не тільки отримати інформацію про функціонування всіх її підсистем, а й дає можливість захищеного управління технічними процесами на виробництві.

Все це призвело до необхідності використання персональних мереж бездротового зв'язку WPAN (Wireless Personal Area Networks) на нижньому рівні мережевий ієрархії автоматизації сучасних підприємств. Бездротові мережі відрізняються більш гнучкою архітектурою, вимагають менших витрат при їх установці і обслуговуванні.

Мережа WPAN являє собою систему обміну даними з невеликим радіусом охоплення і відносно низькими швидкостями передачі. Неліцензований діапазон частот 2,4 ГГц завдяки своїй доступності став популярним для промислової, наукової, медичної апаратури та підходить для недорогих бездротових рішень, які пропонуються для мереж WPAN.

На сьогоднішній день в діапазоні частот 2,4 ГГц найбільш широкого поширення набули три технології бездротової передачі даних: Bluetooth, Wi-Fi, і ZigBee. Порівняльні характеристики цих технологій наведені в табл. 1.1.

Таблиця 1.1 — Порівняння характеристик бездротових технологій передачі даних

Характеристика	Wi-Fi	Bluetooth	ZigBee
Диапазон (м)	20-300	10-100	10-100
Частота (ГГц)	2,4	2,4	2,4
Споживання потужності	Високий рівень	Низький рівень	Дуже низький рівень
Розмір стека протоколів кбайт	більш 1000	більш 250	32-64
Час безперервної роботи від батареї	0,5-5	1-10	100-1000
Максимальна кількість вузлів	10	7	65536
Передбачувана область застосування	Передача відеопотоку з камер	Передача аудіо сигналу (домашній кінотеатр, аудіосистема)	Бездротове з'єднання між датчиками і обчислювальною системою

Технологія Wi-Fi. Дана технологія заснована на стандарті IEEE 802.11. Цей стандарт визначає протоколи, необхідні для організації локальних бездротових мереж WLAN (Wireless Local Area Network). Основні з них — протокол управління доступом до середовища MAC (Medium Access Control) і протокол передачі сигналів у фізичному середовищі.

В якості основного методу доступу до середовища стандартом 802.11 визначено механізм множинного доступу з виявленням несучої і запобіганням колізій CSMA / CA (Carrier Sense Multiple Access with Collision Avoidance). Метод CSMA-CA полягає в наступному: для визначення стану каналу (зайнятий або вільний) використовується алгоритм, відповідно до якого виконується вимір потужності сигналів на вході приймача і оцінюється якість сигналу. Якщо потужність прийнятих сигналів на вході приймача нижче порогового значення, то канал вважається вільним, якщо ж їх потужність вище порогового значення, то канал вважається зайнятим.

В основу стандарту 802.11 покладена стільникова архітектура, причому мережа може складатися як з однієї, так і з декількох осередків. Кожна сота управляється базовою станцією, яка разом з розташованими в межах радіусу її дії робочими станціями користувачів утворює базову зону обслуговування. Точки доступу багатостільникової мережі взаємодіють між собою через розподільну систему, що є еквівалентом магістрального сегменту кабельних локальних обчислювальних мереж.

Оскільки обладнання, що працює на максимальній швидкості, має менший радіус дії, ніж працює на більш низьких швидкостях, то стандартом 802.11b передбачено автоматичне зниження швидкості при погіршенні якості сигналу, для чого використовується метод адаптивного вибору швидкості ARS (Adaptive Rate Selection). Зниження швидкості дозволяє застосовувати більш прості і менш надлишкові методи кодування, чому передаються сигнали стають менш схильними до загасання і спотворень внаслідок інтерференції. Завдяки методу адаптивного вибору швидкості обладнання стандарту IEEE 802.11b може здійснювати обмін даними в різній електромагнітній обстановці.

Технологія Bluetooth (стандарт IEEE 802.15) стала першою технологією, що дозволяє створити бездротову персональну мережу передачі дан них WPAN (Wireless Personal Network). Вона дозволяє здійснювати передачу даних і голосу по радіоканалу на невеликі відстані (10 — 100 м) в неліцензованому діапазоні частот 2,4 ГГц і з'єднувати ПК, мобільні телефони та інші пристрої при відсутності прямої видимості. Так званий нижній (2,45 ГГц) діапазон ISM (industrial, scientific, medical) призначений для роботи промислових, наукових і медичних приладів. Технологія Bluetooth підтримує як з'єднання типу «точка-точка», так і «точка-багато точок». Два або більше використовують один і той же канал пристрою утворюють пікомережа (piconet). Одне з пристроїв працює як основне (master), а решта — як підлеглі (slave). В одній пікомережі може бути до семи активних підлеглих пристроїв, при цьому інші підлеглі пристрої знаходяться в стані «паркування», залишаючись синхронізованими з основним пристроєм. Взаємодіючі пікомережі

утворюють «розподілену мережу» (scatternet). У кожній пікомережі діє тільки одне основне пристрій, проте підлеглі пристрої можуть входити в різні пікомережі. Крім того, основний пристрій однієї пікомережі може бути підлеглим в інший (рис. 1.7), де M — master, S — slave.

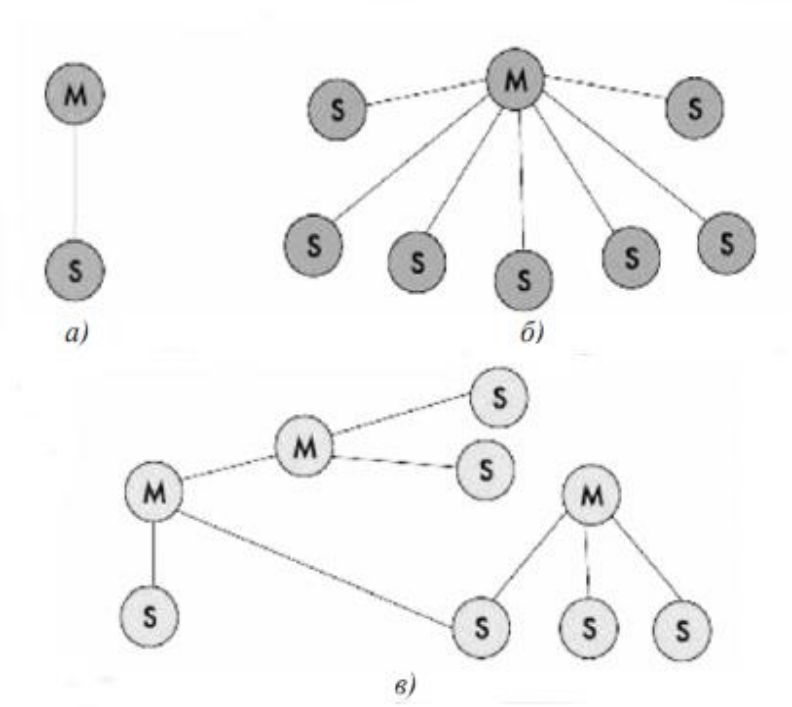


Рисунок 1.7 — Пікомережа з підлеглими пристроями: а) з одним підлеглим пристроєм; б) с кількома підлеглими пристроями; в) розподілена мережа

Основне завдання, яке вирішується за допомогою ZigBee, передача невеликих обсягів даних на середні відстані. Специфічність призначення ZigBee полягає в тому, що приймально-передавальні пристрої цього стандарту повинні мати мінімальне енергоспоживання. З IEEE 802.15.4 і ZigBee не можна передавати якісне потокове аудіо або відео високої чіткості, зате можна реалізувати складні схеми моніторингу та управління практично в будь-якій сфері.

Технологія ZigBee (IEEE, 802.15.4) призначена для низькошвидкісних персональних мереж бездротового зв'язку — LRWPAN (Low Rate Wireless Personal Area Network).

Всього за даними стандартом закріплено 27 каналів в трьох ефірних діапазонах: загальний для всього світу на частоті 2,4 ГГц (16 каналів), додатковий для США на частоті 915 МГц (10 каналів) і для Європи на частоті 868 МГц (один канал). Швидкість передачі даних між пристроями залежить від числа зайнятих каналів і коливається від 256 до 20 кбіт / с. Доступ до середовища здійснюється в частотних діапазонах, що не вимагають

ліцензування ISM (Industrial, Scientific and Medical). Фізичний рівень використовує двійкову фазову маніпуляцію (BPSK) на частотах 868/915 МГц і квадратичну фазову маніпуляцію зі зміщенням (QPSK) на частоті 2,4 ГГц. Для доступу до каналу використовується механізм множинного доступу до середовища з контролем несучої і запобіганням колізій (CSMA-CA).

Стандарт 802.15.4 грунтується на полудуплексной передачі даних, що дозволяє використовувати метод CSMA-CA тільки для запобігання колізій, а не для їх виявлення. На рис. 1.8 зображений варіант архітектури ZigBee-мережі.

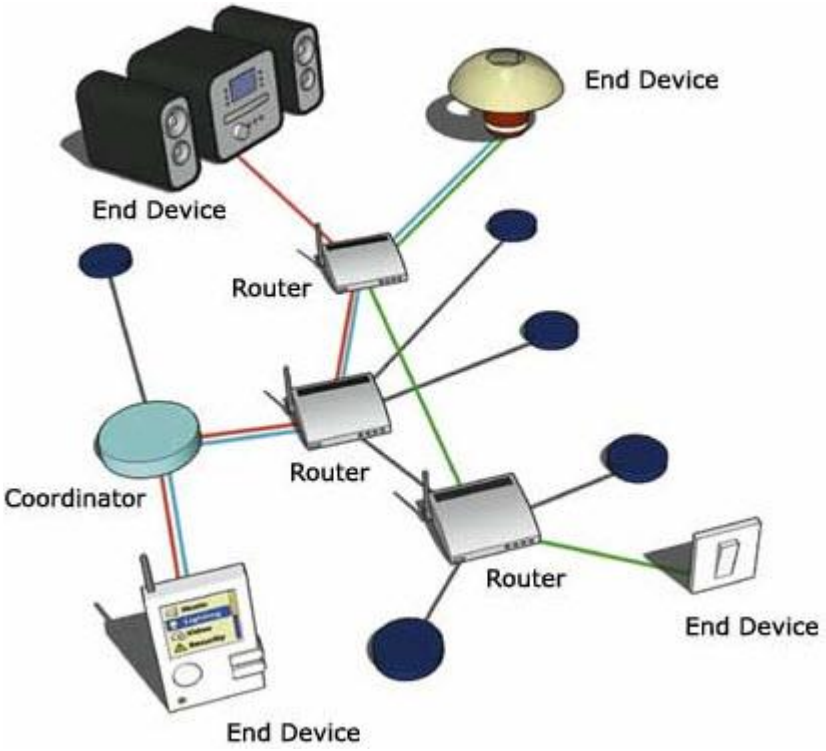


Рисунок 1.8 — Варіант архітектури ZigBee-мережі

Стандарт ZigBee визначає три типи пристроїв: координатори ZigBee, маршрутизатори ZigBee і кінцеві пристрої ZigBee. Кожна мережа повинна містити тільки один координатор ZigBee. Основне завдання координатора полягає в тому, щоб задати параметри для створення мережі і запустити процес налаштування, що передбачає вибір радіочастотного каналу, унікального мережевого ідентифікатора і набору робочих параметрів. Маршрутизатор ZigBee можуть використовуватися для розширення радіуса дії мережі, оскільки вони здатні виконувати функції і ретрансляторів між пристроями, розташованими занадто далеко один від одного, щоб взаємодіяти безпосередньо. Кінцеві пристрої ZigBee не беруть участі в маршрутизації.

Безліч ZigBee-пристроїв здатне працювати спільно, в загальній радіомережі, як в стандартній ієрархії типу «зірка» (коли один маршрутизатор керує всіма потоками даних),

так і в змішаній топології без єдиного координатора. Як тільки маршрутизатори та інші пристрої підключаються до мережі, вони отримують від координатора або від будь-якого маршрутизатора, вже задіяного в мережі, інформацію про неї, і на основі цієї інформації встановлюють свої робочі параметри відповідно до параметрів мережі.

Маршрутизатор ZigBee отримує блок мережевих адрес, які він розподіляє між підключилися до мережі бездротовими або іншими кінцевими пристроями. Щоб зробити маршрутизацію більш ефективної, алгоритм ZigBee також дозволяє маршрутизаторам використовувати скорочення шляхів. Кожен маршрутизатор, на якому передбачається використовувати скорочення, повинен підтримувати таблицю, яка містить пари виду D, N, де D — це адреса цілі, а N — адреса наступного пристрою на шляху до цієї мети.

Технологія ZigBee визначає характер роботи мереж датчиків. Пристрої утворюють ієрархічну мережу, коренем якої є координатор ZigBee. Маршрути можуть враховувати ієрархію, можлива також оптимізація інформаційних потоків.

ZigBee може бути інтегрований в наступні системи: автоматизації життєзабезпечення будинків і будов (віддалене управління мережевими розетками, вимикачами, і т.д.); управління побутовою електронікою; автоматичного зняття показань з різних лічильників (газу, води, електрики і т.д.); безпеки (датчики задимлення, датчики доступу і охорони, датчики витоку газу, води, датчики руху і т.д.); моніторингу навколишнього середовища (датчики температури, тиску, вологості, вібрації і т.д.); промислової автоматизації.

1.6 Огляд технологій об'єднання і існуючих систем

У цьому підрозділі будуть розглянуті існуючі аналоги, які виконують такі ж або схожі завдання. У проаналізованих системах виявимо достоїнства і недоліки.

Технології об'єднання і управління розумним будинком:

LanDriver — універсальна платформа побудови шинних систем управління використовувана в автоматизації будівель. Призначена для управління внутрішніми і зовнішніми системами. Система LanDriver складається з центрального контролера і модулів підключених між собою шиною (стандарт RS-485). До модулів підключається кероване обладнання;

EIB/KNX — Система EIB розподілена, управління здійснюється в межах пристроїв. Пристрої обмінюються інформацією по шині EIB відповідно до власного протоколом.

Система, побудована на EIB, автономна і не залежить від працездатності центрального контролера;

AMX розробляє програмно апаратні засоби віддаленого управління, медіа системою, системою відеоспостереження і широкого спектру датчиків. Протоколи передачі даних закриті. Спочатку застосовувалася власна шина передачі даних, в нової лінійки обладнання застосовуються стандартні протоколи Ethernet, Wi-Fi, а так само має шлюзи сполучень з системами EIB, LON і ін;

Z-wave — технологія бездротової передачі даних розроблена для домашньої автоматизації. В технології Z-wave застосовуються малопотужні і мініатюрні радіо модулі, що вбудовуються в побутову техніку. В основі технології лежить ячеистая технологія, в якій кожен вузол є приймачем і передавачем, тобто при виникненні перешкоди сигнал піде через сусідні вузли мережі, що знаходяться в радіусі дії. Ще однією перевагою є мале енергоспоживання, що разом з малими розмірами, дозволяє вбудовувати Z-wave в різні побутові прилади. Варто відзначити, що більшість систем і технологій автоматизації приміщень закриті.

Одним з найпоширеніших шляхів вирішення розробки системи домашньої автоматизації є використання готових систем управління «розумним будинком». Розглянемо деякі з них:

Node-Red — являє собою інструмент з відкритим вихідним кодом, розроблений компанією IBM, що дозволяє створювати при- розкладання, поєднуючи готові компоненти. Ці компоненти можуть бути пристроями, WEB API або онлайн-службами. Істотним недо- статки є відсутність візуалізації.

SmartVISU — оптимальне рішення для швидкої і недорогий анустановки системи KNX. Компонування / конфігурація віджетів происхо- дит тільки через конфігураційний текстовий файл, це є значним недоліком даної системи управління. Преімуще- ством є наявність інтуїтивно-зрозумілою візуалізації.

MajorDoMo — це відкрита програмна платформа, для автоматизації домашніх процесів. Дана система кроссплатформенная і не вимоглива до ресурсів комп'ютера. Може бути використана, без модулів (датчиків) в якості персонального органайзера. Завдання, які вирішуються за допомогою MajorDoMo:

- система безпеки;
- система мікроклімату;
- медіа система;
- організатор.

Скриншот інтерфейсу додатка зображено на рис 1.9.

Рисунок 1.9 — Інтерфейс головного екрану додатка MajorDoMo

NetPing — система моніторингу навколишнього середовища. Розробником і виробником пристроїв є російська компанія «Alentis Electronis». Основна сфера застосувня — віддалений контроль і моніторинг. Дозволяє здійснювати підключення вельми обмеженої кількості датчиків(до 16 датчиків на 1 пристрій). Не зручний і відносно інтуїтивно незрозумілий вбудований Web-інтерфейс. Завдання, які вирішуються за допомогою пристрою NetPing:

– віддалене управління електроживленням;

– управління безпекою та відстеження надзвичайних подій, використовуючи датчики диму, протікання води, витоку газу, антивандальні системи, управління камерами відео спостереження;

– управління мікрокліматом за допомогою датчиків температури, вологості і управління кондиціонером через інфрачервоний порт;

– управління АТС по порту RS-232 Дистанційне зміна налаштувань в залежності від ситуації;

– відправка повідомлень про неполадки або інші важливі події засобом SMS, електронна пошта;

– доступ до системи в реальному часі через HTTP або SNMP;

– управління освітленням та іншими побутовими приладами за розкладом.

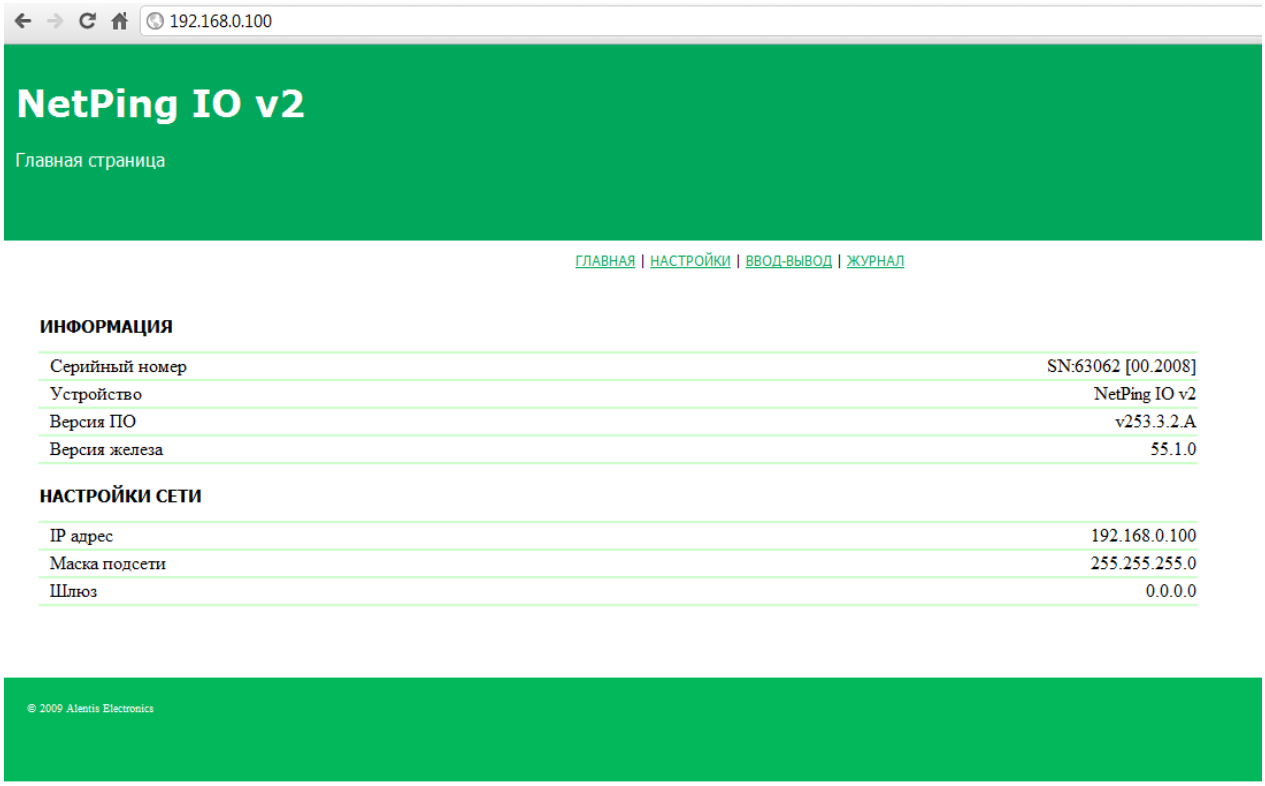


Рисунок 1.10 — Інтерфейс головного екрану додатку NetPing

Сам виробник рекомендує використання стороннього програмного забезпечення, наприклад Zabbix, Nagios, PRTG Network.

OpenRemote — програма забезпечує автоматизацію житлових і комерційних приміщень. OpenRemote дозволяє створити мобільний додаток для розумного будинку без програмування, при цьому можливо використовувати різні технології EIB / KNX, AMX, Z-wave. Простими словами це багатоплатформовий конструктор, в якому Ви створюєте інтерфейс майбутнього мобільного додатка, приклад на рис. 1.11. Контролери які можуть бути використані: AMX, KNX, Beckhoff, Lutron, Z-Wave, 1-Wire, MiCasaVerde Vera, EnOcean, xPL, Insteon, X10, Infrared, Russound, GlobalCache, IRTrans, XBMC, VLC, Samsung SmartTV, panStamps, Denon AVR, Marantz AVR, FreeBox, MythTV, RaZBerry і інші.

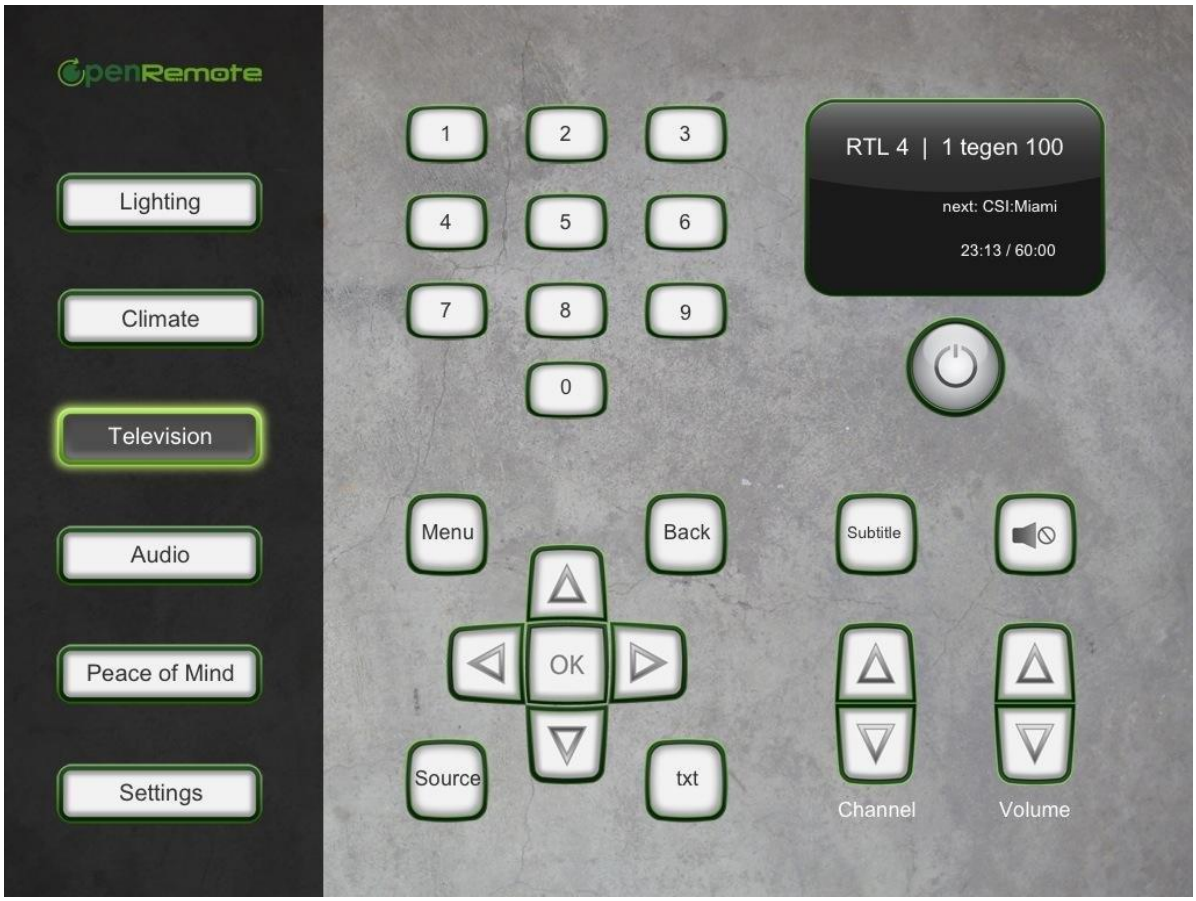


Рисунок 1.11 — Інтерфейс додатку OpenRemote

HomeSapiens — інтелектуальна система з голосовим управлінням. Також, як і OpenRemote, підтримує обмежену кількість технологій, такі як Z-wave, Gira, ZigBee, x10, C-Bus. Основний акцент розробників спрямований на рекламу голосового управління і зручного інтерфейсу.

Freedomotic — кросплатформне програмне забезпечення з відкритим вихідним кодом для процесів домашньої автоматизації. Основне завдання — надання ентузіастам можливості самостійного створення системи управління «розумним будинком», з використанням саморобного обладнання або готових рішень популярних архітектур.

AgoControl — небезплатна закінчена промислова система. Відмінно підійде для бюджетної системи автоматизації. Підтримує мінімальний набір протоколів і сценаріїв, має web-інтерфейс (дивитися рис 1.12).

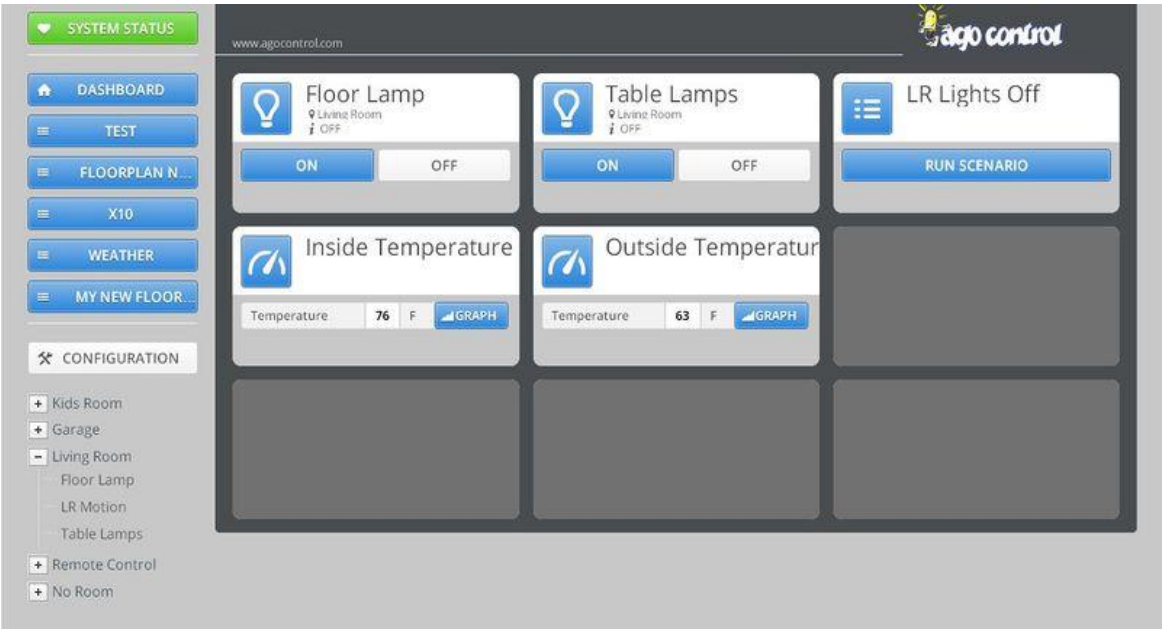


Рисунок 1.12 — Інтерфейс web-додатку AgoControl.

Fibaro — система автоматизації будівель заснована на бездротової технології передачі даних Z-wave. Простий метод монтажу, тому що не треба протягувати метри кабелю. Мініатюрні модулі можуть бути встановлені за будь-яким вимикачем світла або в побутовому приладі. Завдяки бездротовій технології передачі даних пристрою Fibaro можна демонтувати і переносити на нове місце. система Fibaro постійно сканує систему і при необхідності інформує Вас про подію. Висока інтеграція з іншими системами. Мозком системи Fibaro є Home Center. Інтерфейс надає простий контроль над групами пристроїв відповідають за функції — опалення кондиціонування, освітлення і т.д.

2 ДОСЛІДЖЕННЯ АВТОМАТИЗАЦІЇ СИСТЕМИ КЕРУВАННЯ МІКРОКЛІМАТУ РОЗУМНОГО ДОМУ

2.1 Алгоритм функціонування системи керування температурою

На рисунку 2.1 зображено алгоритм функціонування системи керування температурою, де КП — це керуючий пристрій, $t\,\text{зад}$ — це задана температура, яку обирає користувач, $t\,\text{вим}$ — це температура, яку отримує центральний контролер з датчику-термометру.

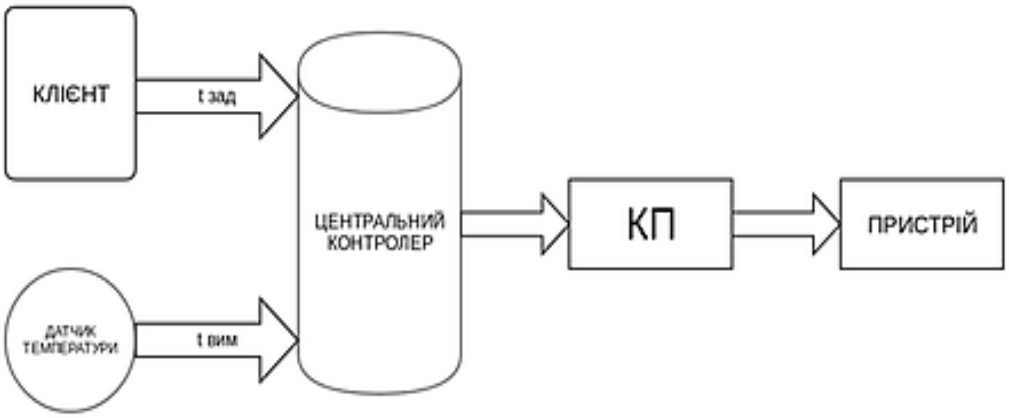


Рисунок 2.1 — Алгоритм функціонування системи керування температурою

Температура повітря підтримується побутовими приладами (нагрівач або пристрій охолодження) і вимірюється датчиком-термометром. Значення $t\,\text{вим}$ отримане з датчика, надходить на центральний контролер. Залежно від різниці між заданим ($t\,\text{зад}$) і виміряним значеннями температури, ЦК виробляє сигнал і передає його на пристрій виконання, яке, в свою чергу, впливає на прилад-термостат.

Прилад починає працювати до такого моменту, поки помилка $\varepsilon = t\,\text{вим} - t\,\text{зад}$не прагнутиме до нуля. Узагальнена структурна схема системи автоматичного регулювання представлена на рисунку2.2.

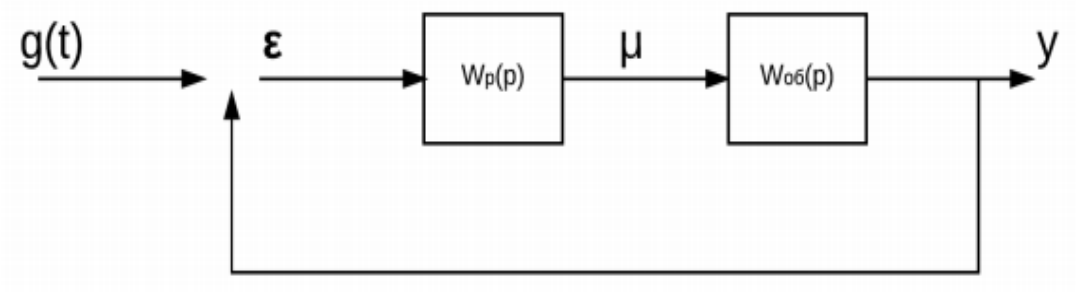


Рисунок 2.2 — Система автоматичного регулювання.

Комплексний коефіцієнт передачі (ККП) від неузгодженості ε до виходу y визначається виразом $W(jw) = Wp(jw)W$об(jw) , де Wр(jw) — ККП регулятора; Wоб(jw) — ККП об’єкту регулювання.

2.2 Вибір закону регулювання

Регулятори з лінійним законом регулювання по математичної залежності між вхідними і вихідними величинами поділяються на такі види:

- пропорційні (П-регулятори);
- пропорційно-інтегральні (ПІ-регулятори);
- пропорційно-інтегрально-диференціальні (ПІД-регулятори).

2.2.1 Пропорційні (П-регулятори)

Для настройки пропорційних регуляторів (П-регуляторів) необхідно задати коефіцієнт передачі kp. Комплексний коефіцієнт передачі регулятора набуде вигляду W_p $(jw) = kp$, в такому випадку система буде мати наступний вигляд(3.1):

$$W(jw) = kpW_{\text{об}}(jw). \quad (2.1)$$

Таким чином, при використанні пропорційного регулятора для управління об'єктом комплексний параметр передачі системи зростає на кожній частоті в kp раз.

Перехідні процеси задаються наступним чином:

$$\mu = kp\varepsilon, \tag{2.2}$$

де ε — вхідний вплив на регулятор, який визначається, як відхилення регульованого значення від поточного впливу; μ — керуючий вплив, який націлений на виключення відхилення регульованого значення від заданого.

Коефіцієнт $kp = \Delta U / \Delta\varepsilon$ — параметр регулювання. Якщо kp приймає великі значення, то існує ймовірність виникнення коливань в контурі регулювання (рис 2.3).

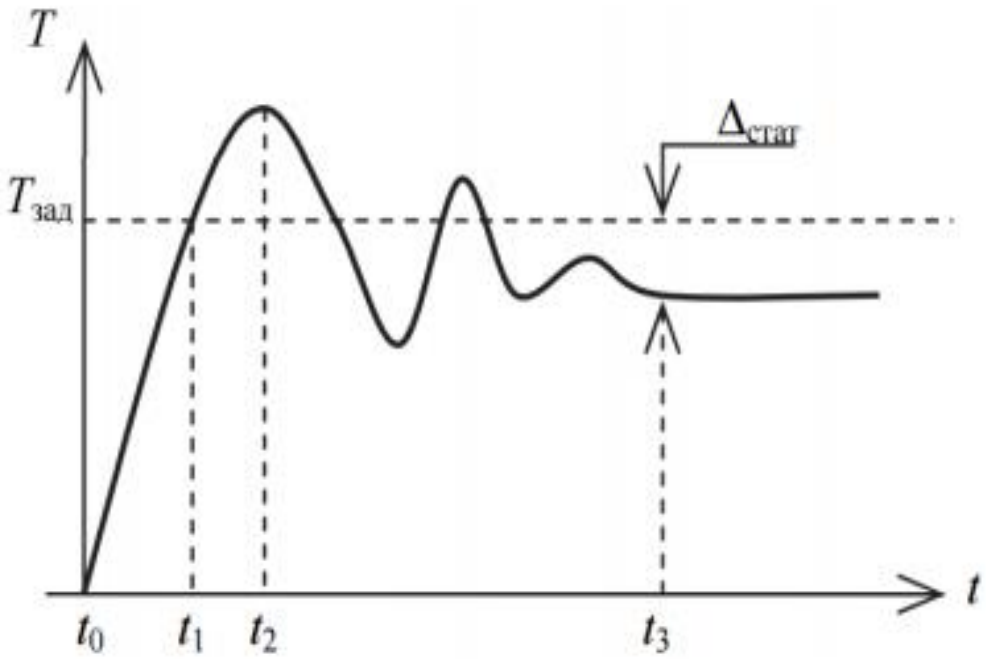


Рисунок 2.3 — Коливальний процес в контурі регулювання в разі пропорційного закону

Після запуску системи управління, дані з датчика температури t вим надходять на сервер. Температура може бути вище або нижче встановленої t зад. Залежно від цього, сервер відправляє максимально можливий сигнал интерфейсному модулю керуючому пристрою, після чого він адресує його відповідному порту виконавчого модуля, до якого підключені нагрівальне або охолоджувальний пристрій. Алгоритм роботи систем охолодження та опалення не відрізняється, різниця полягає лише в пристрої на яке подається сигнал, тому всі розглянуті закони регулювання будуть застосовні до обох

систем. У зв'язку з вищевикладеним, проведемо детальний розгляд тільки однієї із систем — системи опалення.

Після подачі максимально-можливого сигналу, починається процес підвищення температури. Коли вона буде дорівнює t зад, сервер відправить команду на відключення нагрівального елементу. Облік того, що процес охолодження нагрівального елементу відбувається протягом деякого часу, в ході якого, температура в приміщенні до моменту t_2 буде зростати, а потім буде знижуватися, відбувається за допомогою розгляду згасаючих коливань. Через деякий час t_3, відбудеться процес стабілізації температури в приміщенні. Після цього, в наслідок властивості інерційності даної системи, як правило, має місце статична помилка, обумовлена наступним видом: Δст = t зад — t вим.

Вихідний сигнал керуючого пристрою прийме наступний вигляд

$$y(t) = U_0 + kp\varepsilon, \tag{2.3}$$

де U_0 — вихідний сигнал керуючого пристрою, в разі $\varepsilon = 0$.

2.2.2 Пропорційно-інтегральний регулятор

Методом виключення статичної помилки, яка виникає в разі пропорційного регулювання, є введення інтегральною складовою.

Вплив на орган, який здійснює регулювання, з боку пропорційно-інтегрального регулятора відбувається пропорційно інтегралу від відхилення регульованої величини і відхилення:

$$\mu = kp\varepsilon + \frac{1}{T_i}\int_0^t \varepsilon dt, \tag{2.4}$$

де T_i — коефіцієнт, що відображає налаштування регулятора.

Передавальна функція інтегральної і пропорційною складової:

$$W_{\text{пі}}(p) = kp + \frac{1}{T_i}p. \tag{2.5}$$

Закон, що описує регулювання:

$$\mu = kp\left(\varepsilon + \frac{1}{T_{із}}\int_0^t \varepsilon dt\right), \quad (2.6)$$

де $T_{із}$ — постійна часу регулювання зі зворотним зв'язком.

У пропорційно-інтегральному регуляторі, в разі якщо t зад < t вим, то відбувається моментальне включення пропорційної (статичної) компоненти регулятора, а потім послідовно зростає інтегральна (астатична) компонента.

Під час налаштування ПІ-регулятора необхідно задати взаємнонезалежні параметри: коефіцієнт посилення kp і постійну часу інтегрування T_i.

Перехідний процес при ПІ-регулювання представлений на рисунку 2.4

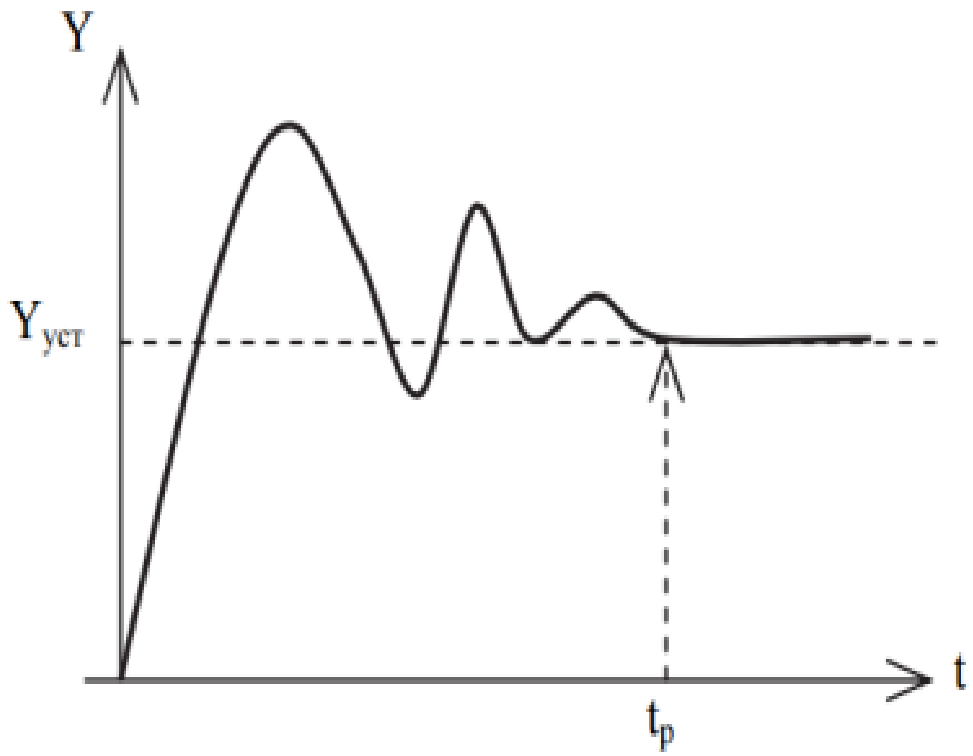


Рисунок 2.4 — Перехідний процес при ПІ-регулюванні

2.2.3 Пропорційно-інтегрально-диференціальний регулятор

Іноді необхідно надати допоміжний регулюючий вплив, з цією метою вводиться диференціальна складова, яка є пропорційною до швидкості відхилення регульованого значення величини від заданого:

$$\mu_д = T_д \, d\varepsilon/dt. \tag{2.7}$$

Регулятори даного виду впливають на керований об'єкт пропорційно відхиленню регульованого значення, інтегралу від наявного відхилення до темпу зміни цього значення:

$$\mu = k_p \varepsilon + \frac{1}{T_i} \int_0^t \varepsilon dt + T_д \frac{d\varepsilon}{dt} \tag{2.8}$$

У разі нерівномірного відхилення регульованого значення пропорційно-інтегрально-диференціальний регулятор в вихіднийперіод часу виробляє максимально-можливий вплив на об'єкт регулювання, після цього величина впливу зменшується до величини, задається пропорційною частиною, потім приступає до впливу інтегральна частина регулятора.

Проміжний етап в ПІД-регуляторах, графік якого зображено на рис 2.5, володіє мінімальним відхиленням по амплітуді до часу, щодо П- до ПІ-регуляторів.

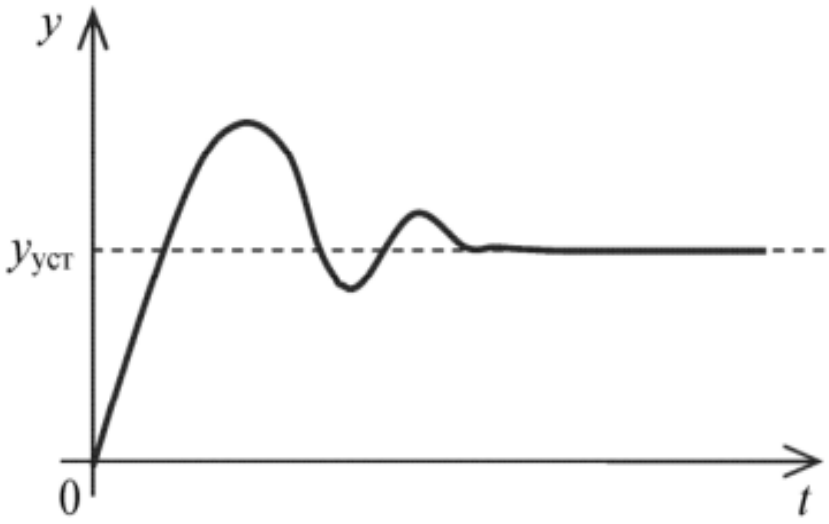


Рисунок 2.5 — Проміжний етап в ПІД-регуляторах

Налаштованим параметрами в ПІД-регуляторі є коефіцієнт пропорційності регулятора kp, постійна, що відображає час інтегрування T_i, і постійна, що характеризує час диференціювання $T_д$

В процесі зростання величини запізнювання в системі, відбувається досить різке збільшення негативних фазових зрушень, ця процедура призводить до зменшення впливу диференціальної частини регулятора.

Через присутність шумів в каналі вимірювання в системі з ПІД-регулятором призводить до суттєвих випадкових коливань сигналу регулятора, це призводить до зростання середньо-квадратичної помилки регулювання і зношування виконавчого пристрою.

Пропорційно-інтегрально-диференціальні регулятори забезпечують для систем управління температурним режимом високі показники регулювання, а саме: неузгодженість регулювання становить менше ніж один відсоток від заданого значення, досить невеликий час стабілізації і малу чутливість щодо збурень ззовні.

Крім того, необхідно враховувати, що в разі неточного визначення параметрів, призначених для настройки, розглянутий регулятор робить негативний вплив у вигляді погіршення роботи системи і переходу в автоколивання в порівнянні з іншими законами регулювання.

3 ДОСЛІДЖЕННЯ СУЧАСНИХ АРХІТЕКТУР РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ ДЛЯ ОС ANDROID

Концепція «Розумного будинку» підготувала особливе місце для мобільних технологій. До недавнього часу цей перспективний напрямок не знаходив широкого застосування, воно більше вважалося ексклюзивним. Але зараз ситуація змінилася докорінно, стало можливим управління «Розумним домом» за допомогою смартфонів або планшетів, в яких використана платформа Android. Це дає можливість постійно мати при собі мобільний пристрій управління, яке призначений для організації зв'язку з інтернетом, ділових записів, дзвінків та інших функцій. Відкритий бездротової протокол надав можливість управління будинком за допомогою планшета з операційною системою Android, так як має доступ до приладів і пристроїв, що знаходяться в будинку і підключеним до електромережі.

Окремі спроби обладнати побутові прилади автоматикою не змогли наділити будинок або квартиру інтелектом. Зараз звичайний смартфон або планшет, може стати універсальним пультом дистанційного керування, завдяки якому можна буде як вимикати в кімнатах світло, так і включати телевізор або кавоварку. Але, перераховані функції — це далеко не всі можливості системи. Наявність мобільної консолі вважається готової платформою для будь-яких додатків, які можуть не тільки вести правильний облік продуктів у вашому холодильнику, але, і ефективно управляти споживанням електричної енергії, опалення, що обов'язково позначиться на економії сімейного бюджету. Сам смартфон з платформою Android — це ланка комунікації між власником і «розумним» будинком. Завдяки рівню покриття мобільного зв'язку, залишатися на зв'язку зі своєю квартирою, навіть перебуваючи на великій відстані від неї, не складе ніяких труднощів.

З самого початку існування Android спостерігалися проблеми при розробці додатків. Які ж це проблеми? У плані дизайну зрозуміло, що абсолютно різні стилі додатків бентежать користувача системи Android, і йому буває важко орієнтуватися. Але що не так з відсутністю стандартів в самому коді? Адже користувач ніяк не може знати, наскільки хороший код програми, і це не впливає на його використання. Проблема полягає в тому, що не всі розробники добре володіють паттернами проектування і вміють розробляти гарну архітектуру додатків. Якщо ж не слідувати чітким принципам в архітектурі є великий ризик отримати код, який:

– неможливо підтримувати. У коді буде багато складної логіки, вона не буде розташована в суворо визначених класах, буде незрозуміло, як працює та чи інша частина вашої програми. З цього випливає, що при додаванні нового функціоналу доведеться або довго і посилено розбиратися в написаному коді, проводити рефакторінг і робити все правильно, або виконати розробку як вийде, не зважаючи на подальший результат;

– неможливо протестувати. Ця проблема плавно випливає з першої. У додаток не зможуть бути додані модульні тести, якщо весь додаток — це один великий модуль. Більш того, в силу особливостей написання тестів для Android-додатків на JVM, при великій кількості залежностей від класів Android в тестованих класах, написання тестів може стати невиконним завданням. А відсутність тестів дає набагато менше впевненості в тому, що код працює правильно.

Один з перших кроків до уніфікованної архітектури є паттерн MVC, діаграма якого представлена на рисунку 3.1. У мобільну розробку його привнесли web-розробники, які змінили профіль. Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

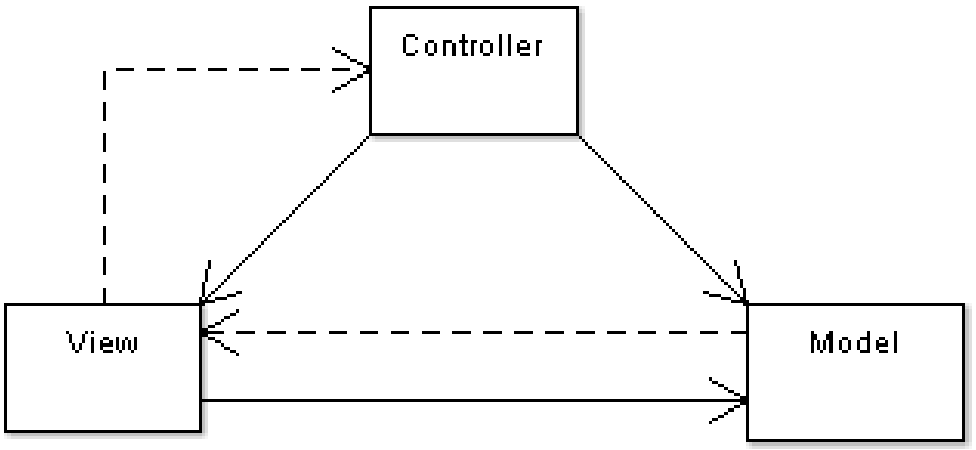


Рисунок 3.1 — Діаграма паттерну MVC

Проте цей підхід не набув значимісті, бо він не має можливості повністю функціонувати в архітектурному середовищі Android.

Наступним кроком стала Clean Architecture. Для Android ця концепція зазнала декількох змін, та декілька спростилася. Така адаптація була викладена в статті Фернандо Цехаса "Architecting Android ... The clean way?",

Найбільш критичною з точки зору тестування програми є бізнес-логіка або бізнес-правила, що визначають суть роботи програми. І вони повинні бути в першу чергу незалежні від інших елементів і протестовані.

Щоб досягти незалежності і можливості тестування, пропонується розбити додаток на 3 ключових шару:

– шар даних (Data Layer) — даний шар відповідає в першу чергу за отримання даних з різних джерел і їх кешування. Він реалізується за рахунок паттерна Repository;

– шар бізнес-логіки (Domain Layer) — на цьому шарі міститься вся бізнес-логіка програми. Цей шар є певним об'єднанням шарів сценаріїв взаємодії і бізнес-логіки в оригінальній «чистій архітектурі». Саме до цього шару звертається шари уявлення для виконання запитів і отримання даних;

– шар уявлення (Presentation Layer) відповідає за логіку відображення даних на екрані, за взаємодію з користувачем і за інші процеси, пов'язані з UI. Цей шар не повинен містити логіку додатка, не пов'язану з UI.

Саме цей шар прив'язується до екранів і допомагає організувати взаємодію з шаром бізнес-логіки і роботу з даними. Цей шар може бути реалізований з використанням будь-якого віддається перевага патерну, наприклад, MVC, MVP, MVVM та інших.

При цьому щоб забезпечити максимальну незалежність цих шарів, на кожному з них використовується своя модель даних, яка конвертується при взаємодії між шарами.

Схема цих шарів виглядає наступним чином(рис 3.2):

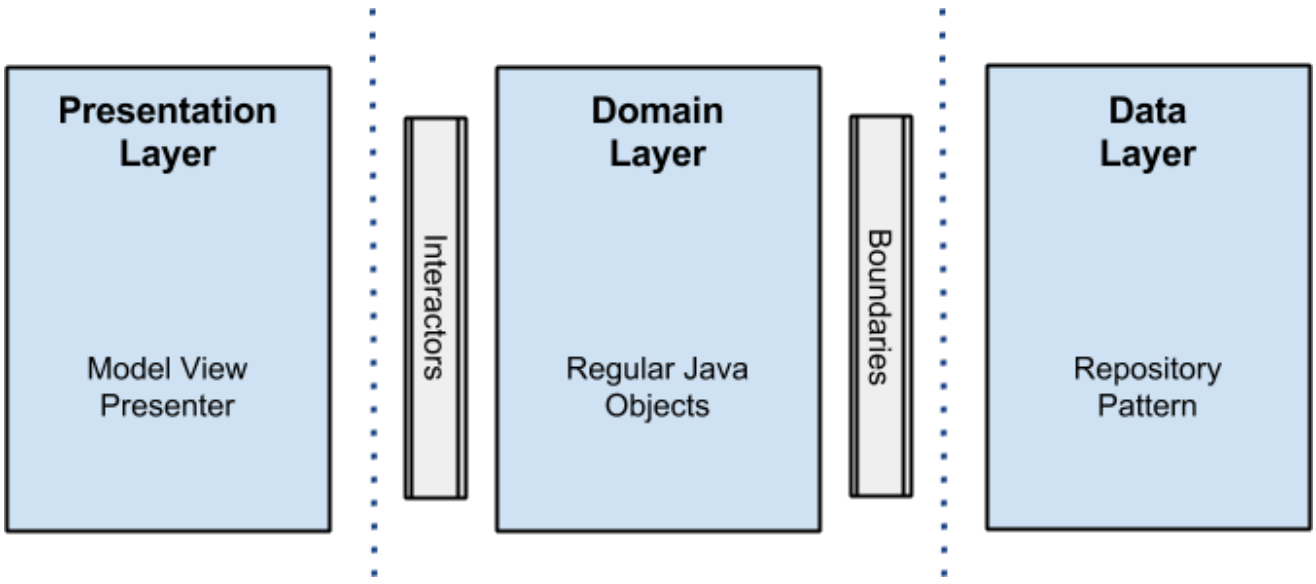


Рисунок 3.2 — Схема шарів додатку у рамках «чистої архітектури»

Метою дослідження є дізнатися про актуальні архітектури, та обрати оптимальну для реалізації мобільного додатку системи розумного дому. Тому особливої уваги заслуговують дві найбільш популярні концепції реалізації шару представлення MVP та MVVM.

3.1 MVP(Model-View-Presenter)

MVP(Model-View-Presenter) — це досить простий патерн, який дозволить нам розділити екран на UI-частина (View), на логіку роботи з UI (Presenter) і об'єкти для взаємодії з UI (Model). Presenter простий для написання тестів, а також може багаторазово використовуватися, тому що уявлення може реалізувати кілька інтерфейсів. У загальному вигляді цей патерн виглядає наступним чином(рис 3.3):

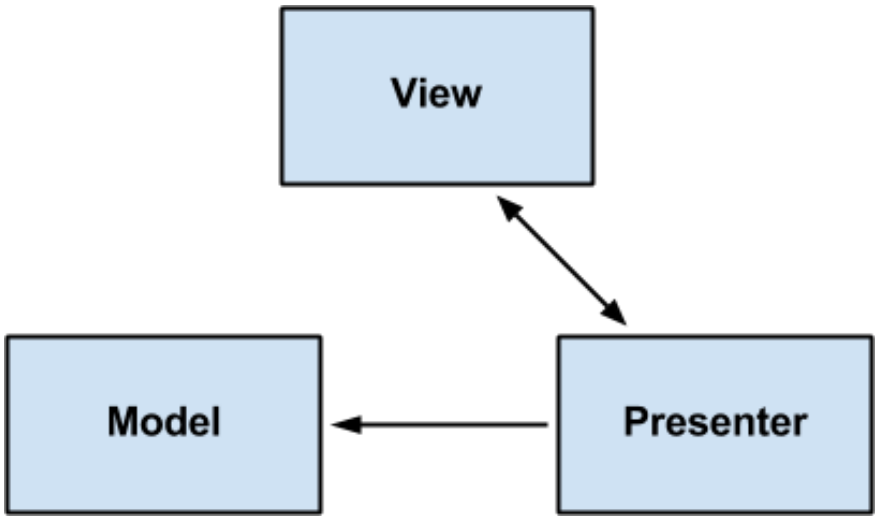


Рисунок 3.3 — Схема функціонування MVP

Склад даної архітектури має вигляд:

– Model — модель являє собою клас для визначення даних, які будуть відображатися або над якими будуть проводитися інші дії у інтерфейсі користувача;

– View — представлення управляє елементами на сторінці, та направляє події до класу пред'явника;

– Presenter — пред'явник містить логіку реагування на події, оновлює Модель (бізнес-логіки і даних з програми) і, в свою чергу, маніпулює станом Представлення. Для полегшення тестування пред'явника, він повинен мати посилання на інтерфейс

представлення замість посилання на конкретну реалізацію. Як наслідок, ви можете легко замінити діюче Представлення на макет для виконання тестів.

Особливості реалізації у системі Android:

– Presenter — це клас, що управляє відображенням даних через спеціальний інтерфейс View. Presenter також звертається до сховища за отриманням даних і займається обробкою життєвого циклу. З недавнього часа за обробку подій життєвого циклу відповідає LifecycleObserver з Android Architeture Components. LifecycleObserver — інтерфейс керування життєвим циклом Activity або Fragment, керування виконується за допомогою анотацій.

– У якості View виступає інтерфейс, який містить методи для роботи з UI, який реалізується в Activity або Fragment.

– Model — зв’язок з класом Repository, який віддає звичайні моделі сутностей.

Існує безліч фреймворків для роботи з MVP архітектурою у Android додадтках, наприклад Mosby, Moxy. Вони дозволяють акуратно структурувати код у відповідно до паттерну MVP.

3.2 MVVM(Model-View-ViewModel)

Інша популярна архітектура має назву MVVM(Model-View-ViewModel). Головною відміною від попереднього паттерну є наявність ViewModel замість Presenter. Вона з одного боку є абстракцією View, а з іншого надає обгортку даних з моделі, які мають зв'язуватись. Тобто вона містить модель, яка перетворена до вигляду, а також містить у собі команди, якими може скористатися ViewModel для впливу на модель. Фактично ViewModel призначена для того, щоб здійснювати зв'язок між моделлю та вікном, відслідковувати зміни в даних, що зроблені користувачем, відпрацьовувати логіку роботи View (механізм команд). Схема роботи паттерну представлено на рисунку 3.4.

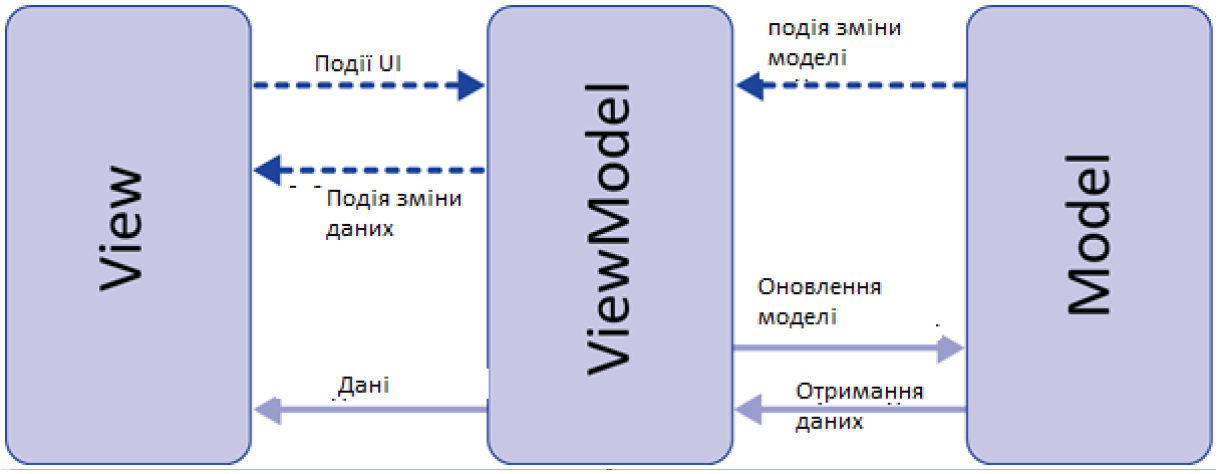


Рисунок 3.4 — Схема роботи паттерну MVVM

Ключовою особливістю паттерна MVVM є те, що жоден компонент (Model, View, ViewModel) не знає про інше явно. Ці компоненти взаємодіють між собою за рахунок механізму зв'язування даних (Bindings), який реалізується засобами тієї чи іншої системи. При цьому зміна даних у ViewModel автоматично змінює дані, які відображаються під View. Аналогічно, будь-яка подія або зміна даних у View (натискання на кнопку, введення тексту та інше) змінює дані в ViewModel. Це дозволяє не зберігати явні посилання на View у ViewModel і навпаки, а також тримати ці компоненти дуже слабо зв'язковими, що зручно при тестуванні.

Особливості реалізації у системі Android:

Основою реалізації VM є біндінг. Наразі є декілька досить якісних реалізацій. По-перше, це DataBinding від Google. Усі маніпуляції з View проходять на XML-розмітці. Элементи екрану зв’язуються з моделями даних, та при зміні стану даних відбувається зміна вигляду представлення.

Два інших варіанта: LiveData з Android Architeture Components та RxBindings від ReactiveX. В цих варіантах біндинг проходить безпосередньо у класі Activity або Fragment. Основою взаємодії є паттер Спостерігач(Observer). Елементи відображеня підписуються на зміну даних(Observable) у ViewModel, та при зміні виконують метод спостерігача, у якому відбувається оновлення відображення.

Окрім цього у Android Architeture Components було додано клас ViewModel, який виступає базовим класом для створення даної архітектури. Головною особливістю є те, що при використанні цього підходу при перевороті екрану більше не буде губитися стан активності, бо об’єкт ViewModel зберігається у спеціальному провайдері.

3.3 Виводи дослідження

Наразі патерн MVP можна вважати стандартним підходом. Його можна використовувати як в контексті Clean Architecture (в разі досить складного додатка), так і самостійно, без зайвих архітектурних шарів. MVVM виглядає свіжим віянням, проте таїть за собою деякі проблеми. Вони починаються при масштабуванні системи. І, на жаль, в разі використання MVVM проблеми будуть виникати куди частіше, ніж при використанні стандартного підходу з MVP. Існує багато задач, які складно вирішити в MVVM підході, починаючи з показу діалогу, закінчуючи відкриттям нового екрану з ViewModel. Тому що ViewModel не знає про існування View. З однієї сторони це навіть добре, бо ViewModel легше тестувати, та такий підхід захищає від витоків пам'яті. Звичайно, всі такі проблеми можна вирішити, наприклад підписати View на зміну стану VM, але використовувати запропонований підхід потрібно акуратно. І, зрозуміло, не можна забувати і про спільне використання MVP і Data Binding, це може вирішиті деякі проблеми та полегшити розробку.

У дипломному проекті буде застосовано Clean architecture з MVVM у якості шару представлення. Незважаючі на всі недоліки цього підходу, слід пам’ятати про його переваги. Також, як можно помітити Google робить великі кроки для популяризації підходу MVVM, як то створення фреймворку DataBinding та випуск Android Architecture Components.

4 РОЗРОБКА СИСТЕМИ КЕРУВАННЯ РОЗУМНИМ БУДИНКОМ

4.1 Проектування системи керування

За необхідний варіант прийнята система керування розумним домом з функціями автоматизації освітлення та мікроклімату. Система відноситься до класу з вбудованим центральним контролером. Архітектура модульного типу. В якості центрального керуючого пристрою обрана системна плата Raspberry Pi. Розширюваність забезпечується за рахунок сумісних с Raspberry модулів,. Система складається з трьох частин: набір сенсорів та актюаторів, центральний контролер і мобільний додаток для управління системою та відображення інформації у візуально-зрозумілому вигляді. Відмінність від інших рішень полягає у використанні тільки вільних компонентів під вільною ліцензією як у випадку з апаратним забезпеченням, так і з програмним. Абсолютно будь-який користувач може налаштувати систему під себе, використовучи недорогі компоненти для Raspberry не написавши при цьому жодного рядка коду. Основна мета розробки — створити недорогу та комфортну систему з можливостями в налаштуванні енергетичних планів і кастомізації. Серед планів на майбутнє необхідно виділити впровадження машинного навчання системи для підбору необідних комфортних режимів функціонування та впровадження рольової моделі доступу для безпеки доступу користувачів.

Схему функціонування власного варіанту системи представлено на рис. 4.1.

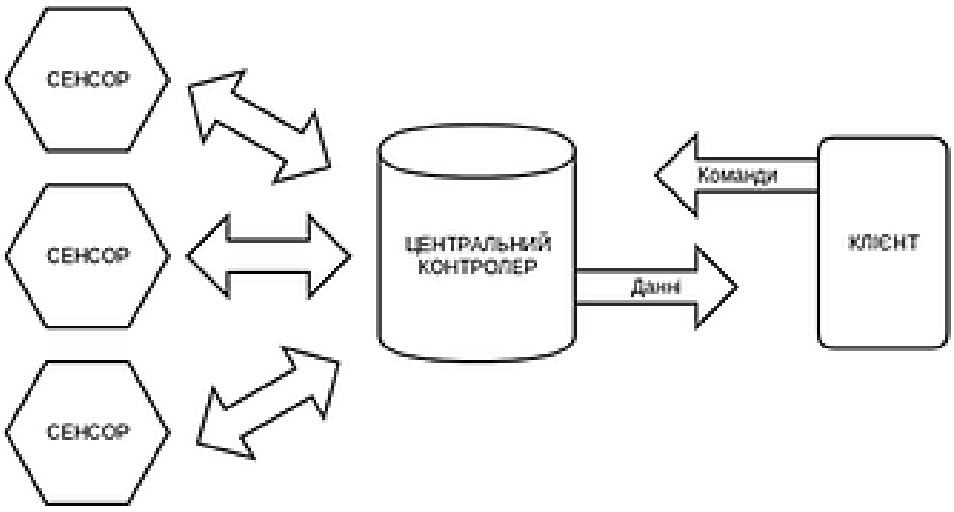


Рисунок 4.1 — Схема функціонування системи керування розумним будинком

Загальний алгоритм роботи системи «Розумний будинок»:

– програмне забезпечення серверу(центрального процесору) приймає керуючі команди з клієнтського додатку;

– програмне забезпечення центрального процесора обробляє отриману інформацію і генерує команди для керуючих пристроїв;

– власною мережі управління інформація від датчиків або інтерфейсів надходить до центрального процесора управління та в зворотньому напрямі.

Система взаємодії між девайсами в системі побудована на основі архітектури з центральним пристроєм-контролером/сервером. Виходячи з цього всі запити, які виходять з мобільного додатку надходять в чергу обробки на сервері і тільки після цього переформатуються в команди і розсилаються по мікроконтролерам (у разі реалізації з кількома кімнатами/приміщеннями).

Переваги даного типу системи:

– передобробка запитів. Запити користувачів потрапляють на мікроконтролери тільки після попереднього аналізу на сервері, що дозволяє уникнути конкуруючих команд, що призводять до Дедлок або одночасного запиту на доступ до контролера з декількох пристроїв. Передобробка гарантує коректність роботи в многопользовательском режимі;

– зняття навантаження з керуючих пристроїв. Вони містить обмежений набор пам'яті і саме тому прямий доступ до них, так само як і зберігання з обробкою великої кількості налаштувань, негативно позначаться на їх функціонуванні. У випадку з сервером, логіка роботи зберігається і обробляється на ньому, а самі пристрої тільки отримують прості сигнали з центрального контролеру;

– масштабованість. У випадку з використанням зв'язку центральний контролер-додаток існує певна проблема з перемиканням між КП для виконання різних скриптів по кімнатах. Крім того навіть якщо використовувати бездротовий зв'язок гарантувати доступ до будь-якого контролеру з будь-якої точки будинку практично малоймовірно. У випадку з центральним сервером це навантаження він бере на себе і статично прописує всі контролери і їх положення в своїх налаштуваннях — користувачеві залишається тільки виставити бажані параметри через додаток, а за решту відповідає сервер.

У якості реалізації буде представлена модель взаємодії клієнту та серверу. Взаємодія цих компонентів буде проходити у виді REST-архітектури. Простий приклад даної архітектури зображено на рисунку 4.2.

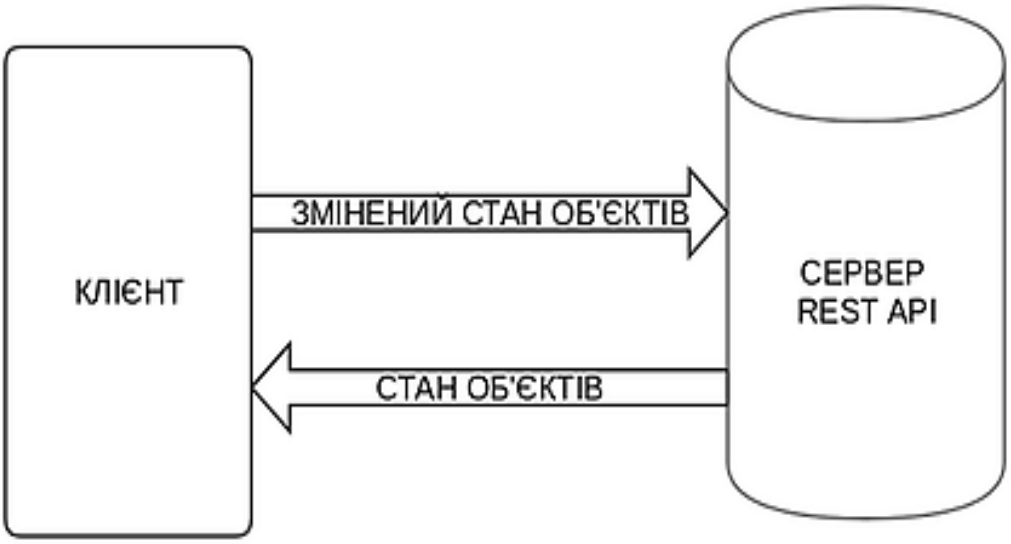


Рисунок 4.2 — Приклад REST архітектури.

Основною ідеєю системи є те, що користувачі можуть керувати своєю системою розумного дому. Окрім цього, серверна частина реалізує автоматизацію деяких сервісів покращення мікроклімату у приміщенні.

Особливості системи:

- керування системами освітлення, мікроклімату та інших;
- можливість зберігти поточний стан усіх сенсорів у якості режиму;
- статистичні дані про використання усіх систем;
- враховувати можливість керування однією системою з декількох акаунтів;
- можливість змінювати інформацію про сенсори;
- можливість додавати сенсори у мережу системи керування.

Програмне забезпечення серверу повинно бути реалізовано на базі REST-архітектури та з реалізацією стандартного CRUD інтерфейсу. У якості програмних засобів реалізації було обрано стек технологій по розробці web-сервісів Java + Spring Framework з базою даних MySQL.

У мобільному додатку повинно бути реалізовано усі функції для роботи з системою, а саме основні модулі керування системою. У якості ОС для мобільного додатку було обрано систему Android, як найпоширенішу у світі.

Інтерфейс користувача має бути ергономічним та інтуїтивно зрозумілим. Кольорова схема в програмному продукті буде відповідати основним принципам розробки інтерфейсів та передбачатиме наявність стандартних елементів, щоб користувачу не потрібно було розбиратися з новим інтерфейсом та його особливостями, що може зменшити коло активних користувачів.

4.2 Використані технології для розробки серверу

Даний додаток буде створений у IDE для розробки java web-додатків Intellij Idea 13.0, адже IntelliJ IDEA забезпечує повний набір функцій, включаючи інструменти та інтеграції з найбільш важливих сучасних технологій і рамок для підприємства і веб-розробки на мовах програмування Java, Scala, Groovy та інших. Було віддано перевагу цій IDE, бо вона на відміну від аналогів має всі необхідні компоненти, фреймворки та технологій вже встроєні та налаштовані, що не потребує їх додаткове скачування та встановлювання у IDE. Більш того вона має достатньо зрозумілий та зручний інтерфейс та багато корисного інструментарія, який полегшує розробку програмного продукту.

Для реалізації створеного алгоритму було обрано мову програмування Java, адже вона показує добрі результати на обробці великих об’ємів даних та достатньо швидко проводить обчислення.

Програми на Java транслюються в байт-код, що виконується віртуальною машиною Java (JVM) — програмою, обробній байтовий код і передавальної інструкції обладнанню як інтерпретатор.

Перевагою подібного способу виконання програм є повна незалежність байт-коду від операційної системи і устаткування, що дозволяє виконувати Java-додатки на будь-якому пристрої, для якого існує відповідна віртуальна машина. Іншою важливою особливістю технології Java є гнучка система безпеки, в рамках якої виконання програми повністю контролюється віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером), викликають негайне переривання.

Часто до недоліків концепції віртуальної машини відносять зниження продуктивності. Ряд удосконалень дещо збільшив швидкість виконання програм на Java:

– застосування технології трансляції байт-коду в машинний код безпосередньо під час роботи програми (JIT-технологія) з можливістю збереження версій класу в машинному коді;

– широке використання переносних орієнтованого коду (native-код) в стандартних бібліотеках;

– апаратні засоби, що забезпечують прискорену обробку байт-коду.

Мова Java активно використовується для створення мобільних додатків під операційну систему Android, тобто основні алгоритми залишаться однаковими, а візуальна реалізція буде відрізнятись в залежності від платформи.

У якості бази даних використовується MySQL. MySQL — вільна реляційна система управління базами даних. Розробка та підтримка сайта MySQL здійснює корпорація Oracle, яка отримала права на торговельну марку разом з поглиненої Sun Microsystems, яка раніше придбала шведську компанію MySQL AB. Продукт поширюється як під GNU General Public License, так і під власною комерційною ліцензією. Крім цього, розробники створюють функціональність за замовленням ліцензійних користувачів. Саме завдяки такому замовленню майже в найраніших версіях з'явився механізм реплікації. MySQL має API для мов Delphi, C, C ++, Ейфель, Java, Лисп, Perl, PHP, Python, Ruby, Smalltalk, Компонентний Паскаль і Tcl, бібліотеки для мов платформи .NET, а також забезпечує підтримку для ODBC за допомогою ODBC-драйвера MyODBC .

Для зручної роботи з базою даних використовуватиметься технологія Hibernate, яка не лише полегшує взаємодію з даними, а також здійснює самостійний захист даних, наприклад, від SQL ін`єкцій та інш. Hibernate ORM дозволяє розробникам легше писати додатки. Як ORM фреймворк, Hibernate пов'язана із збереженням даних , оскільки це належить до реляційних баз даних (через JDBC) .На додаток до своєї власної « рідний» інтерфейс API , Hibernate також реалізація Java API збереження (JPA) специфікації . Як така, вона може бути легко використана в будь-якому середовищі , що підтримує JPA включаючи додатки Java SE , серверів додатків Java EE , підприємства OSGi контейнерів і т.д. Hibernate дозволяє розробляти програми з уже звичними об'єктно- орієнтованими поняттями, включаючи успадкування , поліморфізм , асоціації в рамках колекцій Java. Hibernate не вимагає належності інтерфейсу або базових класів і дозволяє будь-якому класу або структурі даних буди об’єктом Hibernate.

Hibernate має високі експлуатаційні характеристики. Hibernate не вимагає спеціальних таблиць бази даних або поля і генерує більшу частину SQL під час ініціалізації системи замість під час виконання. Hibernate добре масштабується в будь-якому середовищі, надійний, стабільний та якісний, гнучкий, легко конфігурується і розширюється .Доступ до бази даних та параметри підключення знаходяться в окремому файлі properties і підключається програмно, що підвищують гнучкість програми, та надають можливість з легкістю змінити сховище даних без зміни роботи з нею.

При написанні програмного продукту використовуватиметься фреймворк Spring MVC, який дозволив створити проект за архітектурою MVC. У Spring Web MVC можна використовувати будь-який об’єкт в якості команди або форми-бек об'єкта; Не потрібно здійснювати налаштування фрейморвку. Зв'язування даних Spring є дуже гнучким. Spring MVC включає безліч унікальних особливостей веб -підтримки:

– чіткий поділ ролей. Кожна роль — контролер , валідатор , команда об'єкт , форма об'єкта , об'єктна модель , DispatcherServlet , відображення обробник , вигляд розпізнаватель, і так далі — можуть бути виконані спеціалізованим об'єкта;

– потужний і простий у конфігурації як основи фреймворку, так і класів JavaBeans;

– можливість адаптації, не нав'язливість і гнучкість;

– багаторазові бізнес код , немає необхідності в дублюванні;

– настроювані зв'язування і валідація;

– настроювані handler mapping і view resolution;

– гнучка модель передачі даних;

– підтримка простих , але потужних JSP бібліотек тегів.

Також використовувався модуль Maven, який дозволив динамічно підключати необхідні для розробки бібліотеки. Для обробки запитів від клієнта був обран контейнер сервлетів Tomcat 7.0. Apache Tomcat є програмною реалізацією з відкритими джерелами Java Servlet і JavaServer Pages. Apache Tomcat розробляється в рамках відкритого середовища і випущений під ліцензією Apache License версії 2. Apache Tomcat призначений для спільної роботи з кращими у своєму класі розробників з усього світу. За допомогою описаних програмних засобів було реалізовано запропонований алгоритм для рішення поставленої задачі.

4.3 Використані технології для розробки клієнтського додатку

Android ОС — операційна система для смартфонів, планшетних комп'ютерів, електронних книг, цифрових програвачів, наручних годинників, ігрових приставок, нетбуків, смартбуків, окулярів Google, телевізорів та інших пристроїв. В майбутньому планується підтримка автомобілів і побутових роботів. Заснована на ядрі Linux і власної реалізації віртуальної машини Java від Google. Спочатку розроблялася компанією Android Inc., яку потім купила Google.

Android дозволяє створювати Java-додатки, що керують пристроєм через розроблені Google бібліотеки. Android Native Development Kit дозволяє перенести (але не налагоджувати) бібліотеки і компоненти додатків користувача на Сі та інших мовах.

У 84% смартфонів, проданих у другому кварталі 2017 року, була встановлена операційна система Android. При цьому за весь 2017 було продано більше 1 мільярда Android-пристроїв.

У липні 2005 року корпорація Google купила компанію Android Inc. 5 листопада 2007 компанія офіційно оголосила про створення Open Handset Alliance (OHA) і анонсувала відкриту мобільну платформу Android, а 12 листопада 2007 альянс представив першу версію пакету для розробників Android «Early Look» SDK і емулятор Android.

Додатки під операційну систему Android є програмами в нестандартному байт-коді для віртуальної машини Dalvik, для них був розроблений формат настановних пакетів APK. У порівнянні зі звичайними додатками Linux додатки Android підкоряються додатковим правилам: Content Provider — обмін даними між додатками; Resource Manager — доступ до таких ресурсів, як файли XML, PNG, JPEG; Notification Manager — доступ до рядка стану; Activity Manager — управління активними екранами додатків.

Розробку додатків для Android можна вести на мові Java (не нижче Java 1.5). В даний момент офіційним та рекомендованим середовищем розробки є Android Studio.

Android Studio — це інтегроване середовище розробки (IDE) для роботи з платформою Android, анонсована 16 травня 2013 року на конференції Google I/O.

IDE перебувала у вільному доступі починаючи з версії 0.1, опублікованій в травні 2013, а потім перейшла в стадію бета-тестування, починаючи з версії 0.8, яка була випущена в червні 2014 року. Перша стабільна версія 1.0 була випущена в грудні 2014 року, тоді ж припинилася підтримка плагіна Android Development Tools (ADT) для Eclipse.

Android Studio(рис 4.3), заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains, офіційне засіб розробки Android додатків. Дане середовище розробки доступна для Windows, OS X і Linux.

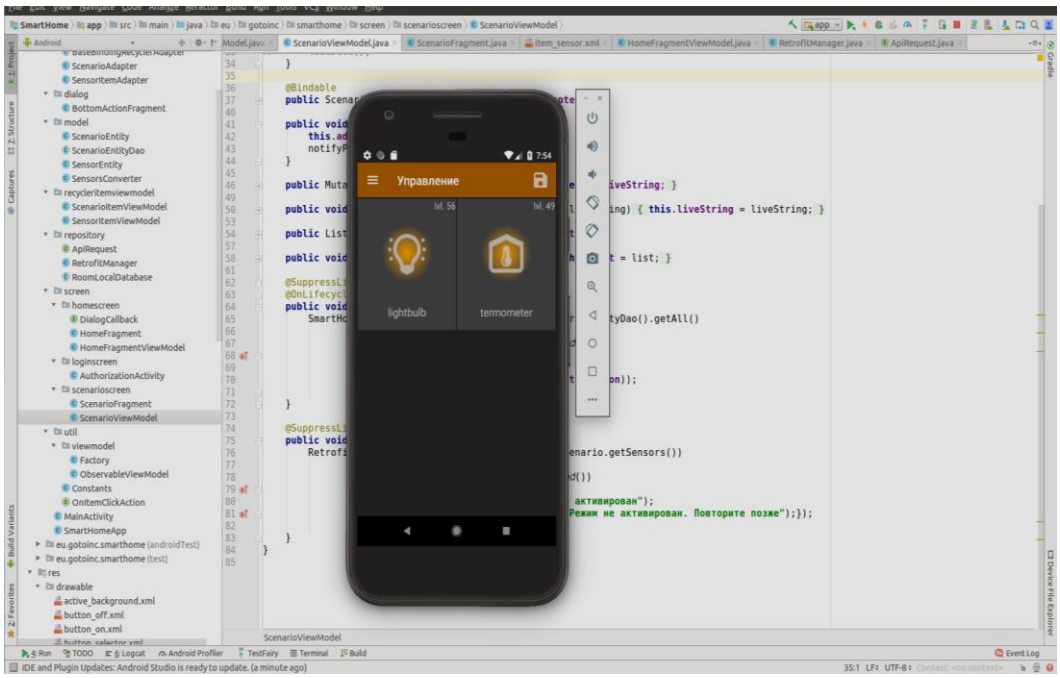


Рисунок 4.3 — Android Studio

17 травня 2017 на щорічній конференції Google I / O, Google анонсувала мову Kotlin Використовуваний в Android Studio Офіційною мовою програмування для платформи Android на додаток до Java і C ++.

У травні 2018 року було анонсовано дев'яту версію Андроіду, та зроблено великий крок до уніфікованої архітектури додатків за допомогою Android Jetpack («наступне покоління компонентів, інструментів і архітектурної допомоги»). Jetpack — набір з купи всього, як уже відомого, так і нового. Наприклад, в Android Architecture тепер додають WorkManager.

Як і в будь-який інший мобільної ОС, в Android існує певний набір засобів розробки або далі SDK. Мета SDK — спростити написання програм під певну платформу шляхом надання готових модулів або бібліотек, для доступу 39 до системних функцій, середовища розробки або компілятора, і засобів тестування. В Android на момент написання дипломної роботи був один з найбільш повних і функціональних

SDK, який включає в себе просунуту середовище розробки Android Studio, набір пакетів Android Bundle, ПЗ для локалізації, ПЗ для редагування SDK і емулятор для тестування додатків.

Окрім SDK будуть використовуватись зовнішні бібліотеки, такі як:

Retrofit 2 — типобезопасний HTTP-клієнт для Android і Java. Він є незамінним інструментом для роботи з API в клієнт-серверних додатках. Retrofit дозволяє зробити повноцінний REST-клієнт, який може виконувати POST, GET, PUT, DELETE. Для позначення типу та інших аспектів запиту використовуються анотації;

RxJava2 — це фреймворк для Java, немає нічого дивного в тому, що він повністю підтримується в Android. RxJava — це потужний інструмент для управління асинхронними потоками даних. Ймовірно, найголовніша причина широкої популярності RxJava в Android — це підтримка RxJava в Retrofit. У основі цього фреймворку стоїть паттерн Observer.

Architecture components від Google. Зокрема DataBinding, Room та ViewModel. DataBinding — бібліотека, за допомогою якої максимально мінімізується з'єднувальний код між логікою програми і її поданням. Room — ORM обгортка над локальною базою даних SQLite, Для позначення типу запиту до БД та інших аспектів використовуються анотації. ViewModel — елемент архітектури MVVM, головна особливість — зберігає стан даних при перевороті екрану, що було однією із головних проблем. згідно опитувань розробників.

4.4 Програмна реалізація системи

Робота з системою починається з входу. Даний додаток формально розподілен на такі логічні модулі :

– авторизація/регістрація;

– основний модуль.

Сторінка авторизації відображається як стандартна сторінка при першому вході на даний додаток(рис 4.4). Передбачається, що користувач вводить свої особисті дані, такі як логін та пароль, які після натискання кнопки “Login” відправляються на сервер. Перед відправленням на сервер дані перевіряються на наявність. Якщо дані були введені, то виповнюється запит на сервер на url /login. Вхідними даними для цього методу є дві строки: логін та пароль. У разі правильності відправлених даних у відповідь прийде true або false у разі неправильних даних.

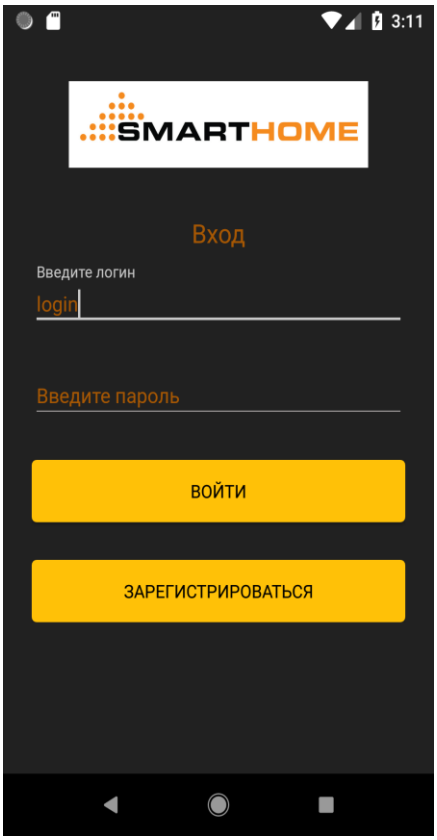


Рисунок 4.4 — Скриншот екрану входу у додаток

На стороні клієнту результат буде збережено за допомогою SharedPreferences. SharedPreferences — клас для зберігання налаштувань і невеликих обсягів інформації іншого типу. Отримати доступ до нього можна з будь-якого класу, пов'язаного з Activity

або View. Мінус — зав'язаний на графічний інтерфейс і не підходить для зберігання великих обсягів інформації. Клас SharedPreferences по-замовчуванню використовується в класі PreferenceActivity для всіх його параметрів. Приклад стандартних операцій наведено на рисунку 4.5

```
public static void saveToken(boolean token, SharedPreferences sharedPreferences) {
    SharedPreferences.Editor ed = sharedPreferences.edit();
    ed.putBoolean(TOKEN, token);
    ed.commit();
}

public static void removeToken(SharedPreferences sharedPreferences) {
    SharedPreferences.Editor ed = sharedPreferences.edit();
    ed.remove(TOKEN);
    ed.commit();
}

public static boolean loadToken(SharedPreferences sharedPreferences) {
    return sharedPreferences.getBoolean(TOKEN,  defValue: false);
}
```

Рисунок 4.5 — Лістинг операцій збереженя, видалення та завантаження даних з SharedPreferences

Друга частина даного модулю — регістрація. Якщо користувач не зареєстрований в системі, він може потрапити на сторінку регістрації перейшовши по посиланню на сторінці авторизації. Після цього відкриється сторінка регістрації на якій користувач повинен ввести свої регістраційні дані, які будуть перевірені на коректність та внесені у базу даних. Аналогічно випадку авторизації при некоректності даних користувач отримає вспливаючі підказки-помилки. Для завершення регістрації необхідно натиснути на кнопку “Зарегистрироваться”, після чого, якщо реєстрація пройшла успішно, відобразиться сторінка авторизації, на якій буде вспливаюче повідомлення про успішну реєстрацію.

Після цього користувач входить в основний модуль додатку, який поділяється на такі частини:

– управління — екран, де користувач може керувати усіма можливими підсистемами, які доступні;

– режими — користувач має можливість зберігти стан усіх сенсорів як режим. З цього екрану користувач може одразу змінити стан системи, а не змінювати кожну підсистему окремо;

– налаштування — на цьому екрані користувач має можливість редагувати інформацію для свого профілю, має можливість змінити пароль та відправити запрошення для користування системою, наприклад, родичам або співмешканцям.

Почати треба з інформації про об’єкт сенсору. В даній системі, програмна реалізація сенсорів представлена високорівневою абстракцією. Клас сенсору має такі поля:

– Id — цілочислений ідентифікатор підсистеми, потрібен більш для зручності збереження даних та більшої швидкості доступу у базі даних;

– Name — строковий ідентифікатор, використовується для зручності доступа у клієнтському шарі архітектури;

– State — властивість об’єкту класу, що вказує на стан сенсору. Може бути увімкненим(ON), вимкненим(OFF) та сенсором, що вийшов з робочого стану(OUT_OF_ORDER);

– Type — властивість об’єкту класу, що вказує на тип підсистеми. Може відповідати за освітлення(LIGHT), мікроклімат(TEMPERATURE), занавіски та вікна(CURTAIN), електрику(ELECTRICITY) та за систему безпеки(SECURITY);

– Value — типи, що відповідають за освітлення та температуру, можуть регулюватися. Тому ця властивість відповідає за ступінь регуляції сенсору.

На екрані регуляції представлено список усіх доступних сенсорів. Наприклад на скриншоті(рис. 4.6) можна побачити сенсори, які відповідають за температуру на кухні, світло у вітальній, охоронну систему та штори у спальні. Жовтим світлом позначені сенсори, які знаходяться у активному стані, без кольору — неактивні, червоним — ті, які вийшли із ладу та потребують обслуговування.

Екран управління був створений за допомогою архітектури MVVM(Model-View-ViewModel).

ViewModel. Виступає мостом між View і Model і обробляє логіку відображення. Запитує у Model дані і передає їх View у вигляді, який View може легко використовувати. Також містить обробку подій, скоєних користувачем додатки під View, таких, як натискання на кнопку. Крім того, ViewModel відповідає за визначення додаткових станів View, які треба відображати, наприклад, чи йде завантаження. У даній реалізації у ViewModel відбуваються запити до репозиторію для завантаження та відправки даних, та їх обробка.

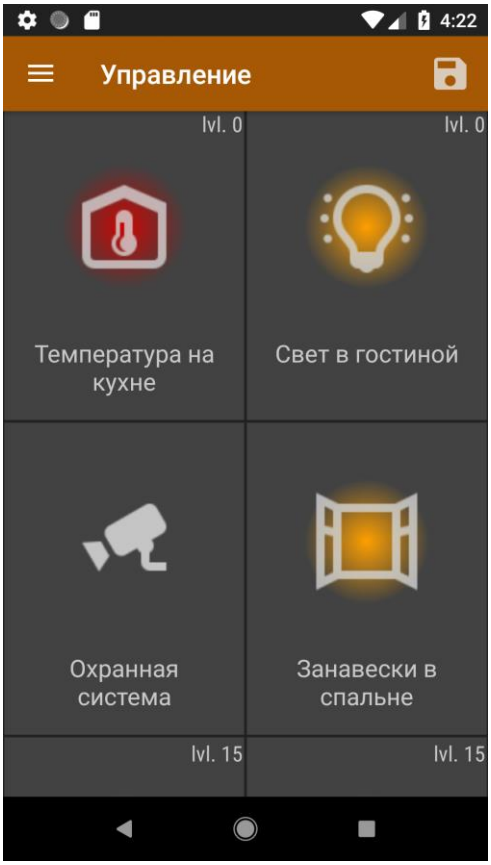


Рисунок 4.6 — Скриншот екрана управління

Зі сторони сервера цей екран використовує запити до бази даних, наприклад головний запит, який повертає список усіх доступних сенсорів, виглядає таким чином(рис 4.7):

```
@RequestMapping(value = "/{sensorId}/info", method = RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
public SensorEntity viewSensorInfo(@PathVariable Long sensorId){
    SensorEntity sensorById = sensorManipulationService.getSensorById(sensorId);
    return sensorById;
}
```

Рисунок 4.7 — Лістинг запиту на отримання усіх доступних сенсорів.

Обробка цього запросу на стороні клієнту проходить у ViewModel-класі екрану регулювання(рис. 4.8).

```
@SuppressLint("CheckResult")
@OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
public void loadData(){
    RetrofitManager.getInstance().getAllSensors()
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(sensorEntities -> {
                setList(sensorEntities);
                setAdapter(new SensorItemAdapter(list, R.layout.item_sensor, action));
            }, throwable -> liveString.setValue("Возникли проблемы с Интернет соединением"));

}
```

Рисунок 4.8 — Лістинг обробки запиту у ViewModel-класі клієнту

Обробка натискання проходить за допомогою callback-методів у RecyclerView. Результатом натискання на елемент списку є поява модального вікна BottomSheetDialogFragment(рис 4.9). Воно теж будується за допомогою архітектури MVVM, причому, для елементів списка та модального вікна використовується VM одного типу. Цей клас відповідає за відображення інформації, за обробку натискання на кнопку та зміну стану ProgressBar.

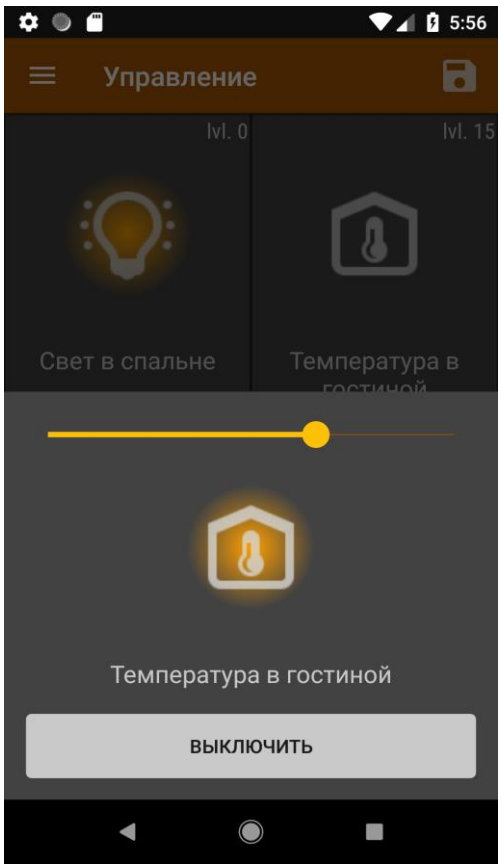


Рисунок 4.9 — Скриншот модального вікна керування сенсору

Окремо треба розповісти про обробку зміни стану сенсора з клієнтського додатку. Для цієї операції були використани деякі прийоми з реактивного програмування. В

момент натискання на кнопку або зміни стану Progress-bar, генерується еміт нових даних, через суб'єкт. Проте, щоб не відловлювати випадкове натискання або зміну стану, був встановлен оператор debounce, який відфільтровує елементи, сгенеровані джерелом(суб'єктом), які швидко послідують інший випущений елемент. За ним слідує оператор switchmap, який заміщує одне джерело іншим, ігноруючи результати виконання першого. Наступним джерелом є запит на сервер(рис 4.10 та 4.11), який міняє стан сенсору.

```
@RequestMapping(value = "/{sensorId}/edit", method = RequestMethod.PUT, produces = MediaType.APPLICATION_JSON_VALUE)
public SensorEntity editSensorInfo(@PathVariable Long sensorId,
                                   @RequestParam(value = "name", required = false) String name,
                                   @RequestParam(value = "desc", required = false) String description,
                                   @RequestParam(value = "type", required = false) String type,
                                   @RequestParam(value = "state", required = false) String state,
                                   @RequestParam(value = "value", required = false) Integer value){

    SensorEntity sensorToUpdate = new SensorEntity();
    sensorToUpdate.setName(name);
    sensorToUpdate.setDescription(description);
    if (type != null) {
        sensorToUpdate.setType(SensorType.valueOf(type));
    }
    if (state != null) {
        sensorToUpdate.setState(SensorState.valueOf(state));
    }
    sensorToUpdate.setValue(value);

    SensorEntity updatedSensor = sensorManipulationService.updateSensor(sensorId,sensorToUpdate);
    return updatedSensor;
}
```

Рисунок 4.10 — Лістинг виконання запиту на зміну стану сенсор на серверній частині

```
public SensorItemViewModel(SensorEntity sensor) {
    subject = PublishSubject.create();
    this.sensor = sensor;
    subject.debounce( timeout: 3, TimeUnit.SECONDS)
            .switchMap(sensorModel -> RetrofitManager.getInstance().changeSensor(sensorModel).toObservable())
    .subscribeOn(Schedulers.io())
    .observeOn(AndroidSchedulers.mainThread())
    .subscribe(sensorModel -> {
        this.sensor = sensor;
        notifyPropertyChanged(BR.sensor);
    });
```

Рисунок 4.11 — Лістинг виконання запиту на зміну стану сенсор на клієнтській частині

В цьому випадку. якщо відбувається зміна стану термосенсора, у свою роль вступає пропорційний регулятор, який вважаючи на передану інформацію запускає процес регулювання, по ходу якого відправляє сигнали на виконавчі елементи. Якщо термосистема була активована, то регулятор буде періодично перевіряти розбіжність бажаної температури та отриманої з термодатчика. При великій помилці він буде включатись автоматично. Лістинг регулятора наведено на рисунку 4.12.

```
public class TemperatureController extends DeviceController {
    private static final Logger LOGGER = LogManager.getLogger();
    private volatile boolean isRunning = false;
    private static final double kp = 4;
    private Microcontroller microcontroller = new Microcontroller(currentValue);
    private Integer wantedValue = currentValue;

    @Override
    public void run() {
        isRunning = true;
        while (isRunning) {
            try {
                wait(1000);
            } catch (InterruptedException e) {
                LOGGER.error(e.getMessage());
            }
        }
    }
    private void changeCurrentValueWithError() {
        double u = 0;
            while (currentValue != wantedValue){
                currentValue = microcontroller.getValue();
                Integer error = currentValue - wantedValue;
                if (error > 0) {
                    u = kp * error;
                }
                try {
                    wait(100);
                } catch (InterruptedException e) {
                    LOGGER.info(e.getMessage());
                }}
            microcontroller.getSignalToChangeValue(u);
    }
    @Override
    public void sendSignalToChangeValue(Integer value) {
        this.wantedValue = value;
        changeCurrentValueWithError();
    }
    public void terminate() { isRunning = false; }
```

Рисунок 4.12 — Лістинг пропорційного регулятора

В майбутньому по закінченню роботи регулятора, мобільний додаток буде отримувати повідомлення.

Одним із нововведень цього додатку є збереження поточного стану усіх сенсорів у якості режиму призначеному для користувача. Код збереження у локальну базу даних було реалізовано у ViewModel екрану регулювання(рис. 4.13).

```
public void saveScenario(String name){
    Completable.fromAction(() -> SmartHomeApp.getInstance().getDatabase()
            .scenarioEntityDao().insert(new ScenarioEntity(name, list)))
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(()-> liveString.setValue("Режим успешно сохранен."),
                    throwable -> liveString.setValue("Произошла ошибка, режим не сохранен"));
```

Рисунок 4.13 — Лістинг реалізації збереження режиму для користувача

Використати збережені сценарії можно на екрані режимів. Цей екран також розроблено за допомогою ViewModel. У шарі VM були виконані операції отримання сценаріїв з бази даних та впровадження їх у систему(рис 4.14)

```
@SuppressLint("CheckResult")
@OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
public void loadData() {
    SmartHomeApp.getInstance().getDatabase().scenarioEntityDao().getAll()
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(scenarioEntities -> {
                list = scenarioEntities;
                setAdapter(new ScenarioAdapter(list, action));
            });
}

@SuppressLint("CheckResult")
public void setScenario(ScenarioEntity scenario) {
    RetrofitManager.getInstance().updateSensors(scenario.getSensors())
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(scenarioList -> {
                liveString.setValue("Режим успешно активирован");
            }, throwable -> {liveString.setValue("Режим не активирован. Повторите позже");});
```

Рисунок 4.14 — Лістинг ViewModel екрану режимів

Окрім представлених та реалізованих функцій на серверній частині був реалізован CRUD-інтерфейс. У таблиці 4.1 є повний список запитів з повною інформацією про них.

Таблиця 4.1 – Запити для серверної частини.

URL запиту	Опис запиту	Параметри	Тип параметрів	Результат
/user/all	Повертає усіх користувачів			List<User>
/user/get/{id}	Повертає користувача за його ID	ID	User id (long)	User
/user/register	Регіструє нового користувача	Name Login Pass Phone Email	Усі поля (strings)	User
/user/{id}/info/edit/	Редагує інформацію про користувача	ID Name Login Pass Phone Email	User id (long) Інші (strings)	User
/user/{id}/delete	Видаля користувача за його ID	ID	User id (long)	User

/login	Виконує вхід в систему	Login Password	Усі поля (strings)	true/false
user/{id}/logout	Виконує вихід з системи	ID	User id (long)	true/false
/sensor/all	Повертає усі сенсори			List<Sensor>
/sensor/active	Повертає усі активні сенсори			List<Sensor>
/sensor/inactive	Повертає усі неактивні сенсори			List<Sensor>
/sensor/add	Реєструє новий сенсор	Name Descr Type	All fields (strings)	Sensor
/sensor/{id}/info	Повертає інформацію про сенсор по його ID	ID	Sensor ID (long)	Sensor
/sensor/edit	Редагує стан або інформацію про сенсор	Sensor	Sensor	Sensor
/sensor/edit	Редагує стан або інформацію про сенсори	List<Sensor>	Sensor	Sensor
/sensor/{id}/remove	Видаляє сенсор з системи	ID	Sensor ID (long)	Sensor

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Аналіз потенційних небезпечних і шкідливих виробничих чинників проектованого об’єкту, що мають вплив на персонал

У даному дипломному проекті розробляється програмне забезпечення. Розроблене програмне забезпечення орієнтоване на роботу з персональним комп'ютером. Експлуатовані для вирішення внутрішньовиробничих завдань ПЕОМ типу IBM PC мають наступні характеристики:

споживана потужність	220 Вт;
робоча напруга	220 В;
напруга джерел живлення	+12 В; - 12 В; +5 В;
робоча частота	50 Гц.

Виходячи з приведених характеристик, вочевидь, що для людини існує небезпека поразки електричним струмом, унаслідок недбалого поводження з комп'ютером і порушення правил експлуатації, залишення частин ПЕОМ, що знаходяться під напругою, відкритими або знятих для ремонту вузлів.

Відповідно до [15] до легкої фізичної роботи відносяться всі види діяльності, виконувані сидячи і ті, що не потребують фізичної напруги. Робота користувача ПК відноситься до категорії 1а.

При роботі на ПЕОМ користувач піддається ряду потенційних небезпек. Унаслідок недотримання правил техніки безпеки при роботі з машиною (невиконання огляду відкритих частин ПЕОМ, що знаходяться під напругою або знятих для ремонту вузлів) для користувача існує небезпека поразки електричним струмом.

Джерелами підвищеної небезпеки можуть служити наступні елементи:

- розподільний щит;
- джерела живлення;
- блоки ПЕОМ і друку, що знаходяться в ремонті.

Ще одна проблема полягає у тому, що спектр випромінювання комп'ютерного монітора включає рентгенівську, ультрафіолетову і інфрачервону області, а також широкий діапазон хвиль інших частот. Небезпека рентгенівського проміння мала, оскільки цей вид випромінювання поглинається речовиною екрану. Проте велику увагу слід приділяти біологічним ефектам низькочастотних електромагнітних полів (аж до порушення ДНК).

Відповідно до [16], при обслуговуванні ПЕОМ мають місце фізичні і психофізичні небезпечні, а також шкідливі виробничі чинники:

– підвищене значення напруги в електричному ланцюзі, замикання якої може відбутися через тіло людини;
– підвищений рівень статичної електрики;
– підвищений рівень електромагнітних випромінювань;
– підвищена або знижена температура повітря робочої зони;
– підвищений або знижений рух повітря;
– підвищена або знижена вологість повітря;
– відсутність або недостатність природного світла;
– підвищена пульсація світлового потоку;
– недостатня освітленість робочого місця;
– підвищений рівень шуму на робочому місці;
– розумове перенапруження;
– емоційні навантаження;
– монотонність праці.

5.2 Заходи щодо техніки безпеки

Основним небезпечним чинником при роботі з ЕОМ є небезпека поразки людини електричним струмом, яка посилюється тим, що органи чуття людини не можуть на відстані знайти наявності електричної напруги на устаткуванні.

Проходячи через тіло людини, електричний струм чинить на нього складну дію, що є сукупністю термічної (нагрів тканин і біологічних середовищ), електролітичної (розкладання крові і плазми) і біологічної (роздратування і збудження нервових волокон і інших органів тканин організму) дій.

Тяжкість поразки людини електричним струмом залежить від цілого ряду чинників:

– значення сили струму;
– електричного опору тіла людини і тривалості протікання через нього струму;
– роду і частоти струму;
– індивідуальних властивостей людини і навколишнього середовища.

Розроблений дипломний проект передбачає наступні технічні способи і засоби, що застерігають людину від ураження електричним струмом:

– заземлення електроустановок;

– занулення;

– захисне відключення;

– електричне розділення мережі;

– використовування малої напруги;

– ізоляція частин, що проводять струм;

– огорожа електроустановок.

Занулення зменшує напругу дотику і обмежує години, протягом яких людина, ткнувшись до корпусу, може потрапити під дію напруги.

Струм однофазного короткого замикання визначається по наближеній формулі:

$$I_K = \frac{\text{Uф}}{\text{Zп} + \frac{\text{Zт}}{3}}\ , \qquad (5.1)$$

де Uф - номінальна фазна напруга мережі, В;

Zп - повний опір петлі, створене фазними і нульовими дротами, Ом;

Zт - повний опір струму короткого замикання на корпус, Ом.

Згідно таблиці 4 [17]: Zт /3 = 0,1 Ом.

Для провідників і жил кабелю для розрахунку повного опору петлі використовуємо формулу(5.2.) :

$$Z\text{п} = \sqrt{\text{Rп}^2 + \text{Xп}^2}\ , \qquad (5.2)$$

де Rп = Rф + Ro - сумарний активний опір фазного Rф і нульового Ro дротів, Ом;

Xп - індуктивний опір паяння дротів, Ом.

Перетин 1 км мідного дроту S = 2.5 мм, тоді згідно таблицям 5 і 6 [17], має такий опір:

Xп = 0,11 Ом;

Rф = 7,55 Ом;

Ro = 7,55 Ом.

Отже, Rп = 7,55 + 7,55 = 15,1 Ом.

Тоді по формулі (5.2) знаходимо повний опір петлі :

$$Zп = \sqrt{15,1^2 + 0,11^2} \approx 15,1 \quad (Ом).$$

Струм однофазного короткого замикання рівний:

$$Iк = \frac{220}{15,1 + 0,1} = 14,47 \quad (А).$$

Дія плавкої вставки на ПЕОМ забезпечується, якщо виконується співвідношення:

$$Iк \geq k * Iн, \qquad (5.3)$$

де Iн - номінальний струм спрацьовування плавкої вставки, А;

k - коефіцієнт кратності нелінійного струму Iн, А.

Коефіцієнт кратності нелінійного струму Iн розраховується по формулі (5.4.) :

$$Iн = P / U, \qquad (5.4)$$

де P = 220 Вт - споживана потужність;

U = 220 В - робоча напруга;

k = 3 А - для плавких вставок.

Отже, Iн = 220 / 220 = 1 А.

Підставивши значення у вираз (5.3), одержимо:

$$14,47 > 3*1.$$

Таким чином, доведено, що апарат забезпечить спрацьовування(і захист) при підвищенні номінального струму.

5.3 Заходи, що забезпечують виробничу санітарію і гігієну праці

Вимоги до виробничих приміщень встановлюються [25], ДБН, відповідними ГОСТами і ОСТамі з урахуванням небезпечних і шкідливих чинників, що утворюються в процесі експлуатації електроустаткування.

Підвищення працездатності людини і збереження її здоров'я забезпечується стабільними метеорологічними умовами.

Мікроклімат виробничих приміщень визначається діючими на організм людини поєднаннями температури, вологості і швидкості руху повітря, а також температури навколишніх поверхонь. Значне коливання параметрів мікроклімату приводити до порушення систем кровообігу, нервової і потовидільної, що може викликати підвищення або пониження температури тіла, слабкість, запаморочення і навіть непритомність.

Відповідно до [15] встановлюють оптимальну і допустиму температуру, відносну вологість і швидкість руху повітря в робочій зоні . За відсутності надмірного тепла, вологи, шкідливих речовин в приміщенні досить природної вентиляції.

У приміщенні для виконання робіт операторського типу(категорія 1а), пов'язаних з нервово-емоційною напругою, проектом передбачається дотримання наступних нормованих величин параметрів мікроклімату (табл.5.1).

Таблиця 5.1 - Санітарні норми мікроклімату робочої зони приміщень для робіт категорії 1а.

Пора року	Температура, С	Відносна вологість, %	Швидкість руху повітря, м/с
Холодна	22…24	40…60	0,1
Тепло	23…25	40…60	0,1

У приміщенні, де знаходиться ПЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (з пристроєм вентиляційних каналів в перекриттях будівлі і вертикальних шахт) й установленого промислового кондиціонера фірми Mitsubishi, який дозволяє вирішити переважну більшість завдань по створінню та підтримці необхідних параметрів повітряного середовища. Цей метод забезпечує приток потрібної кількості свіжого повітря, визначеного в ДБН (30 $м^3$ в годину на одного працівника).

Шум на виробництві має шкідливу дію на організм людини. Стомлення операторів через шум збільшує число помилок при роботі, призводить до виникнення травм. Для

оператора ПЕОМ джерелом шуму є робота принтера. Щоб усунути це джерело шуму, використовують наступні методи. При покупці принтера слід вибирати найбільш шумозахисні матричні принтери або з великою швидкістю роботи(струменеві, лазерні). Рекомендується принтер поміщати в найбільш віддалене місце від персоналу, або застосувати звукоізоляцію та звукопоглинання(під принтер підкладають демпфуючі підкладки з пористих звукопоглинальних матеріалів з листів тонкої повсті, поролону, пеноплену).

При роботі на ПЕОМ, проектом передбачені наступні методи захисту від електромагнітного випромінювання : обмеження часом, відстанню, властивостями екрану.

Обмеження години роботи на ПЕОМ складає 3,5-4,5 години. Захист відстанню передбачає розміщення монітора на відстані 0,4-0,5 м від оператора. Передбачений монітор 20" TFT, Samsung 2043BW відповідає вимогам стандарту TCO'03.

TCO'03 пред’являє жорсткі вимоги в таких областях: ергономіка (фізична, візуальна і зручність користування), енергія, випромінювання (електричних і магнітних полів), навколишнє середовище і екологія, а також пожежна та електрична безпека, які відповідають всім вимогам [21].

Для зниження стомлюваності та підвищення продуктивності праці обслуговуючого персоналу в колірній композиції інтер'єру приміщень для ПЕОМ дипломним проектом пропонується використовувати спокійні колірні поєднання і покриття, що не дають відблисків.

У проекті передбачається використання сумісного освітлення. У світлий час доби приміщення освітлюватиметься через віконні отвори, в решту часу використовуватиметься штучне освітлення.

Як штучне освітлення необхідно використовувати штучне робоче загальне освітлення. Для загального освітлення необхідно використовувати люмінесцентні лампи. Вони володіють наступними перевагами: високою світловою віддачею, тривалим терміном служби, хоча мають і недоліки: високу пульсацію світлового потоку.

При експлуатації ПЕОМ виробляється зорова робота. Відповідно до [22] ця робота відноситься до розряду 5а. При цьому нормоване освітлення на робочому місці(Ен) при загальному освітленні рівна 200 лк.

Приміщення завдовжки 12 м, шириною 10 м, заввишки 4 м обладнується світильниками типу ЛП02П, оснащеними лампами типу ЛБ зі світловим потоком 3120 лм кожна.

Виконаємо розрахунок кількості світильників в робочому приміщенні завдовжки a=12 м, шириною b=10 м, заввишки з=4 м, використовуючи формулу (5.5) розрахунку штучного освітлення при горизонтальній робочій поверхні методом світлового потоку:

$$n = (E \cdot S \cdot Z \cdot k)/(F \cdot U \cdot M), \quad (5.5)$$

де F - світловий потік = 3120 лм;

E - максимально допустима освітленість робочих поверхонь = 200 лк;

S - площа підлоги = 120 м2;

Z - поправочний коефіцієнт світильника = 1,2;

k - коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації світильників = 1,5;

n - кількість світильників;

U - коефіцієнт використовування освітлювальної установки = 0,6;

M - кількість ламп у світильнику =2.

Отже, $n = (200 \cdot 120 \cdot 1{,}2 \cdot 1{,}5)/(3120 \cdot 0{,}6 \cdot 2) = 12$.

Виходячи з цього, рекомендується використовувати 12 світильників. Світильники слід розміщувати рядами, бажано паралельно стіні з вікнами. Схема розташування світильників зображена на рис. 5.1.

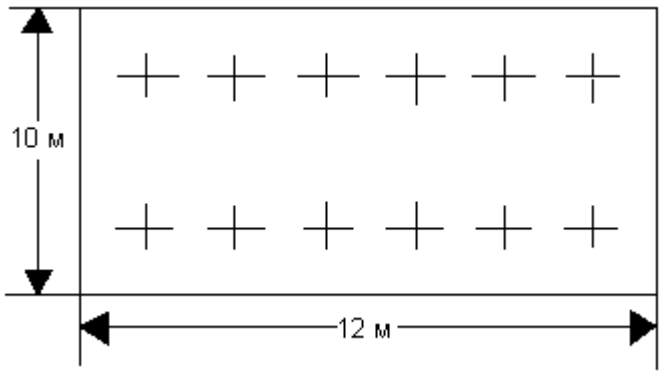


Рисунок 5.1 - Схема розташування світильників

5.4 Рекомендації по пожежній безпеці

Пожежі в приміщеннях, де встановлена обчислювальна техніка, представляють небезпеку для життя людини. Пожежі також пов’язані як з матеріальними втратами, так і з відмовою засобів обчислювальної техніки, що у свою чергу спричиняє за собою порушення ходу технологічного процесу.

Пожежа може виникнути при наявності горючої речовини та внесення джерела запалювання в горюче середовище. Пальними матеріалами в приміщеннях, де розташовані ПЕОМ, є:

– поліамід - матеріал корпусу мікросхеми, горюча речовина, температура самозаймання аерогелю 420 °С ;

– полівінілхлорид - ізоляційний матеріал, горюча речовина, температура запалювання 335 °С, температура самозаймання 530 °С, кількість енергії, що виділяється при згоранні - 18000 - 20700 кДж/кг;

– стеклотекстоліт ДЦ - матеріал друкарських плат, важкозаймистий матеріал, показник горючості 1.74, не схильний до температурного самозаймання;

– пластика кабельний №489 - матеріал ізоляції кабелю, горючий матеріал, показник горючості більш 2.1;

– деревина - будівельний і обробний матеріал, матеріал з якого виготовлені меблі, горючий матеріал, показник горючості більше 2.1, теплота згорання 18731 - 20853 кДж/кг, температура запалювання 399 °С, схильна до самозаймання.

Згідно [24] приміщення відносяться до категорії В (пожежовибухонебезпечним) і згідно правилам побудови електроустановок простір усередині приміщення відноситься до вогненебезпечної зони класу П - IIа (зони, розташовані в приміщеннях, в яких зберігаються тверді горючі речовини).

Потенційними джерелами запалення при роботі ПЕОМ є:

- іскри при замиканні і розмиканні ланцюгів;
- іскри і дуги коротких замикань;
- перегріви від тривалого перевантаження і наявності перехідного опору.

Продуктами згорання, що виділяються при пожежі, є : оксид вуглецю, сірчистий газ, оксид азоту, синильна кислота, акропеін, фосген, хлор та ін. При горінні пластмас, окрім звичайних продуктів згорання, виділяються різні продукти термічного розкладання: хлорангідрідні кислоти, формальдегіди, хлористий водень, фосген, синильна кислота, аміак, фенол, ацетон, стирол та ін., що шкідливо впливають на організм людини.

Для захисту персоналу від дії небезпечних і шкідливих чинників пожежі проектом передбачається застосування промислового протигаза з коробкою марки В(жовта).

Пожежна безпека об’єктів народного господарства регламентується [21] і забезпечується системами запобігання пожежам і протипожежному захисту. Для успішного гасіння пожеж вирішальне значення має швидке виявлення пожежі і своєчасний виклик пожежних підрозділів до місця пожежі.

Зменшити горюче навантаження не представляється можливим, тому проектом передбачається застосувати наступні способи і їх комбінації для запобігання утворенню(внесення) джерел запалення :

– застосування устаткування, що задовольняє вимогам електростатичної безпеки;

– застосування в конструкції швидкодіючих засобів захисного відключення можливих джерел запалення;

– виключення можливості появи іскрового заряду статичної електрики в горючому середовищі з енергією, рівної і вище мінімальної енергії запалення;

– підтримка температури нагріву поверхні машин, механізмів, устаткування, пристроїв, речовин і матеріалів, які можуть увійти до контакту з пальним середовищем, нижче гранично допустимої, становить 80% якнайменшої температури самозаймання пального.

– заміна небезпечних технологічних операцій більш безпечними;

– ізольоване розташування небезпечних технологічних установок і устаткування;

– зменшення кількості пальних і вибухонебезпечних речовин, що знаходяться у виробничих приміщеннях;

– запобігання можливості утворення пальних сумішей на лінії, вентиляційних системах і ін.;

– механізація, автоматизація та справність(потокова) виробництва;

– суворе дотримання стандартів і точне виконання встановленого технологічного режиму;

– запобігання можливості появи в небезпечних місцях джерел запалення;

– запобігання розповсюдженню пожеж і вибухів;

– використовування устаткування і пристроїв, при роботі яких не виникає джерел запалення;

– виконання вимог сумісного зберігання речовин і матеріалів;

– наявність громовідводу;

– ліквідація можливості самозаймання речовин і матеріалів .

Для запобігання пожежі в обчислювальних центрах проектом пропонується виконання наступних вимог :

– електроживлення ЕОМ повинно мати автоматичне блокування відключення електроенергії на випадок зупинки системи охолоджування і кондиціонування;

– система вентиляції обчислювальних центрів повинна бути обладнана блокуючими пристроями, що забезпечують її відключення на випадок пожежі;

– робочі місця повинні бути оснащені пожежними щитами, сигналізацією, засобами для сповіщення про пожежну небезпеку (телефонами), медичними аптечками для надання першої медичної допомоги, розробленим планом евакуації.

Для зниження пожежної небезпеки в приміщеннях використовуються первинні засоби гасіння пожеж, а також система автоматичної пожежної сигналізації, яка дозволяє знайти початкову стадію загоряння, швидко і точно оповістити службу пожежної охорони про час і місце виникнення пожежі.

Відповідно до правил пожежної безпеки для промислових підприємств приміщення категорії В підлягають устаткуванню системами автоматичної пожежної сигналізації. Проектом передбачається застосування датчика типу ІДФ - 1(димовий фотоелектричний датчик), оскільки специфікою пожеж обчислювальної техніки і радіоапаратури є, в першу чергу, виділення диму, а потім - підвищення температури.

При виникненні пожежі в робочому приміщенні обслуговуючий персонал зобов'язаний негайно вжити заходи по ліквідації пожежі. Для ліквідації пожежі використовують вогнегасники (хімічно-пінні, пінні для повітря ОП-5, ОП-6, ОП-9, вуглекислотні ОУ-5), пісок, пожежний інвентар (сокири, ломи, багри, шерстяну або азбестову ковдри). Як засіб індивідуального захисту проектом передбачається використовування промислового протигаза з маскою, фільтруючої коробки В.

В якості організаційно-технічних заходів рекомендується проводити навчання робочого персоналу правилам пожежної безпеки.

5.5 Вплив на навколишнє середовища

В даний час зростає кількість комп'ютерної техніки в усіх галузях діяльності людини. Багато користувачів і виробників помиляються, вважаючи, що зі зменшенням і удосконаленням комп'ютерів, зменшиться їх негативний вплив на навколишнє середовище.

На даний момент найбільш суворим з існуючих світових стандартів екологічності для комп'ютерної техніки є стандарт ТСО-99. У порівнянні з попередніми він містить додаткові обмеження по частині екології, ергономіки, енергоспоживання і емісії пристроїв.

Організація по захисту навколишнього середовища Greenpeace з 2006 року оцінює виробників електроніки за кількістю важких металів і отруйних речовин, наприклад інгібіторів горіння, використовуваних ними при виробництві (інгібітор - речовина, присутність якого в невеликих кількостях призводить до запобігання або уповільнення процесів горіння або корозії; інгібітори знижують швидкість хімічних реакцій або пригнічують їх). Однак навіть оцінки такої організації, як Greenpeace, не можуть претендувати на об'єктивність. Адже в одних випадках вона використовує перевірену інформацію, що стосується, наприклад, заходів щодо утилізації відходів, а в інших спирається тільки на дані виробника. А якщо компанія не повідомляє ніяких відомостей, то автоматично опиняється на нижніх рядках рейтингу. Крім того, енергетичні витрати на виробництво і перевезення продукції також необхідно враховувати при оцінці екологічної ефективності. Адже часи, коли техніка виготовлялася тільки на одному заводі, давно пройшли. Сьогодні окремі комплектуючі закуповуються на різних підприємствах по всьому світу, після чого здійснюється складання пристроїв. Тому найчастіше навіть самі компанії не можуть знати, які шкідливі речовини потрапляють в атмосферу при виготовленні їх продукції і які саме метали або токсини в ній містяться.

ЖК-екрани - один з джерел парникових газів, які набагато шкідливіше діоксиду вуглецю. Рідкокристалічні монітори швидко знайшли популярність, прийшовши на зміну громіздким ЕПТ-моделям. І це не дивно, адже вони мають тонкі корпуса і споживають значно менше електроенергії. За іншим аспектам екологічної безпеки дисплеї на основі рідких кристалів також вважалися проривом, тому що в них не використовувався газ, що містить свинець. Досить довго ніхто не звертав уваги на застосовуваний для чищення РК-панелей трехфторістий азот (NF3), і тільки в середині 2008 року вченими було доведено наявність даної хімічної речовини в атмосфері. Відкриття було вражаючим: порівняно з діоксидом вуглецю (CO2) NF3 має в 17 000 разів більше активного парникового газу, а його атмосферний час напіврозпаду може складати від 550 до 740 світлових років (у CO2 - від 30 до 40 років). Закону, який обмежував би рівень викиду NF3, поки не існує.

Виявлення енерговитрат є таким же проблематичним процесом, як і визначення кількості матеріалів, придатних для вторинної переробки, і важких металів, що містяться в пристроях. Таким чином, надійним показником екологічності залишається тільки рівень енергоспоживання.

Полівінілхлорид, що позначається зазвичай абревіатурою ПВХ, - це різновид пластику, що застосовується в самих різних цілях. З нього зроблена зовнішня оболонка кабелів, якими з'єднуються пристрої, він оточує електричний провід портативного комп'ютера. Це дешевий, міцний і вельми поширений матеріал. Разом з тим, за словами ІТ- аналітика «Грінпіс» Кейсі Харрелл, «ПВХ - найгірший з пластиків». Він є причиною виникнення гормонального дисбалансу, проблем в репродуктивної сфері та різних форм раку. Полівінілхлорид практично неможливо правильно утилізувати. Внаслідок старий матеріал виявляється зазвичай на звалищі з відходами або, того гірше, спалюється з метою вилучення мідних жил і інших цінних компонентів. При його згорянні утворюється вкрай шкідливий канцерогенний діоксин. Звалища і хімічні поховання забруднюють джерела води. Єдиний спосіб правильно утилізувати ПВХ полягає в тому, щоб відправити його в центр небезпечних відходів.

Залишається лише сподіватися, що настане час, коли технології будуть допомагати людині, не завдаючи незворотної шкоди здоров'ю навколишнього середовища.

ВИСНОВКИ

У результаті виконання магістерської атестаційної роботи було проведено аналіз систем розумного дому, а саме систем з центральним контроллером, засобів бездротового зв'язку між компонентами розумного будинку. Було проведене дослідження законів регулювання.

Було проведене дослідження багатьох існуючих систем розумного дому. Наразі ця галузь не стоїть на місці, тому був потрібен повний аналіз від найперших образців до майбутніх прототипів. Як результат досліджень стало ясно, що найоптимальнішим варіантом зараз є клієнт-серверна система з центральним контроллером. Хоча дуже привабливо виглядають приклади використання хмарних сервісів.

За підсумками досліджень у галузі автоматизації систем, з метою економії енергетичних ресурсів та підвищення комфорту було використано пропорційний закон керування системою охолодження і опалення. Управління всіма компонентами «розумного будинку» відбувається з розробленого мобільного додатку. Розроблено алгоритми функціонування системи управління і системи контролю, що дозволяють віддалено в реальному часі контролювати і управляти програмно-апаратним комплексом.

Було виконано дослідження новітніх способів архітектури створення додатків на базі ОС Android. Отримані знання та навички були використані на практиці в ході реалізації клієнтського додатку системи розумного дому.

В ході розробки були підтримані найновіші тенденції та проведена кропітлива робота в підвищенні якості програмного забезпечення.

В результаті проведеної роботи також були визначені напрямки подальших досліджень. Одним з напрямків є застосування методів штучного інтелекту для прийняття рішень зміну умов комфорту. Це дозволить реалізовувати більш складні сценарії автоматизації управління.

У розділі «Охорона праці» виконано аналіз потенційних небезпек при роботі із засобами обчислювальної техніки і механізмами, розроблені заходи щодо техніки безпеки, заходи, які забезпечують виробничу санітарію і гігієну праці, розраховане штучне освітлення, виконані рекомендації по пожежній безпеці, розглянутий можливий вплив на навколишнє середовище.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Табунщиков Ю. А. Интеллектуальные здания. Экологические системы: электронный журнал энергосервисной компании. [Текст] / Ю. А. Табунщиков — М. : АВОК, 2002 — 719 с.

2) Прохоров А. А. Цифровой дом завтрашнего дня [Текст] / А. А. Прохоров– Пб. : Питер, 2003. — 39-43с.

3) Кразит Т. Стандарт IEEE 802.15.4 как алтернатива [Текст] / Т. Кразит. — М. : ИНФРА-М, 2003. — 856 с.

4) Бараш Л. Многообразие стандартов беспроводных технологий [Текст] / Л. Бараш. — М. : Наука, 2003. — 436 с.

5) Шахнович И. Беспроводные локальные сети. Анатомия стандартов IEEE 802.11 [Текст] / И. Шахнович. — ЭЛЕКТРОНИКА: НТБ, 2003. — 354 с.

6) Тесля, Е.А. «Умный дом» своими руками. Строим интеллектуальную цифровую систему в своей квартире: учеб. пособие [Текст] / Е.А.Тесля — СПб.: Питер, 2008. — 224 с.

7) Дорф Р., Современные системы управления [Текст] / Р.Дорф, Р. Бишоп — М.: Лаборатория Базовых Знаний, 2002, — 832 с.

8) Лукас В.А. Теория автоматического управления: [Текст] / В.А. Лукас — М.: Недра, 1990. –416 с

9) Android-CleanArchitecture [Электронный ресурс] / Ф. Цехас. — Режим доступа : www/ URL: https://github.com/android10/Android-CleanArchitecture/ — 10.12.2016. — Загл. с экрана.

10) Doğan R.Temperature and humidity control of the tunnels in the dam using wireless sensor networks [Text] / R.Doğan, E.Erdem — IEEE, 2015. — 379 с.

11) Martin R. Clean Code: A Handbook of Agile Software Craftsmanship [Text] / R. Martin — Prentice Hall, 208. — 46 с.

12) Макаров И.М. Интеллектуальные системы автоматического управления [Text] И.М. Макаров, В.М. Лохин управления — М.: Физмалит, 2001. — 576 с

13) Бизнес журнал — Умный дом [Электронный ресурс] / С.В. Богданов.– Режим доступа : www/ URL: : http://b-mag.ru/themes/smart-. — СПб.: Наука и техника 2005. — 208 с.

14) Автоматизация зданий. Обзор и сравнение технологий. [Электронный ресурс] /] / С.-Петерб. гос. ун-т, фак. прикладной математики — процессов управления. — Режим

доступа : www/ URL: http://apcp.apmath.spbu.ru/ru/staff/tuzov/onapr.html/ — 10.12.2018 г. — Загл. с экрана.

15) ГОСТ 12.1.005-88. Міждержавний стандарт. Система стандартів безпеки праці. Загальні санітарно-гігієнічні вимоги до повітря робочої зони

16) ГОСТ 12.0.003-74 Небезпечні і шкідливі виробничі фактори. Класифікація

17) ДСТУ 7237:2011 Національний стандарт України. Система стандартів безпеки праці. Електробезпека. Загальні вимоги та номенклатура видів захисту

18) ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми. Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.

19) ГОСТ 12.1.004-91. Пожежна безпека. Загальні вимоги .

20) ДБН В.2.5-67. Опалення вентиляція та кондиціонування.

21) ГОСТ 12.1.006-84. Електромагнітні поля радіочастот. Допустимі рівні на робочих місцях і вимоги до проведення контролю

22) ДБН В.2.5-28-2006. Природне і штучне освітлення.

23) ГОСТ 12.4.009-83. Пожежна техніка для захисту об'єктів. Основні види. Розміщення і обслуговування.

24) ДСТУ Б В.1.1-36-2016. Визначення категорії приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною безпекою.

25) ДСП 173-96. Державні санітарні правила планування та забудови населених пунктів

26) Симметрон. Электронные компоненты. Каталог 2002, 2002г. – 192с.

ДОДАТОК А.
Лістинг клієнтського додатку

```
public class ScenarioViewModel extends ObservableViewModel implements LifecycleObserver {

 private List<ScenarioEntity> list;
 private ScenarioAdapter adapter;
 private OnItemClickAction action;

 private MutableLiveData<String> liveString;

 public ScenarioViewModel(OnItemClickAction action) {
   liveString = new MutableLiveData<>();
   this.action = action;
 }

 @Bindable
 public ScenarioAdapter getAdapter() {
   return adapter;
 }

 public void setAdapter(ScenarioAdapter adapter) {
   this.adapter = adapter;
   notifyPropertyChanged(BR.adapter);
 }

 public MutableLiveData<String> getLiveString() {
   return liveString;
 }

 public void setLiveString(MutableLiveData<String> liveString) {
   this.liveString = liveString;
 }

 public List<ScenarioEntity> getList() {
   return list;
 }

 public void setList(List<ScenarioEntity> list) {
   this.list = list;
 }
```

```
  @SuppressLint("CheckResult")
  @OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
  public void loadData() {
    SmartHomeApp.getInstance().getDatabase().scenarioEntityDao().getAll()
        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(scenarioEntities -> {
          list = scenarioEntities;
          setAdapter(new ScenarioAdapter(list, action));
        });
  }

  @SuppressLint("CheckResult")
  public void setScenario(ScenarioEntity scenario) {
    RetrofitManager.getInstance().updateSensors(scenario.getSensors())
        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(scenarioList -> {
          liveString.setValue("Режим успешно активирован");
        }, throwable -> {liveString.setValue("Режим не активирован. Повторите позже");});
  }
}

public class ScenarioFragment extends Fragment implements OnItemClickAction {
  private FragmentScenarioBinding binding;

  public ScenarioFragment() {
    // Required empty public constructor
  }

  public static ScenarioFragment newInstance() {
    ScenarioFragment fragment = new ScenarioFragment();

    return fragment;
  }

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

  }
```

```
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
          // Inflate the layout for this fragment

          binding = DataBindingUtil.inflate(inflater, R.layout.fragment_scenario, container, false);
          ScenarioViewModel viewModel = ViewModelProviders.of(this, new
Factory(this)).get(ScenarioViewModel.class);
          getLifecycle().addObserver(viewModel);
          binding.setVm(viewModel);
          binding.rvScenario.setLayoutManager(new LinearLayoutManager(getContext(),
LinearLayoutManager.VERTICAL, false));

          binding.getVm().getLiveString().observe(this, s -> {
            Toast.makeText(getContext(), s, Toast.LENGTH_SHORT).show();
          });
          return binding.getRoot();
        }

        @Override
        public void onClick(Object object) {
          AlertDialog alertDialog = new AlertDialog.Builder(getContext(),
THEME_DEVICE_DEFAULT_DARK)
              .setMessage("Вы уверены, что хотите включить режим \"" + ((ScenarioEntity) object).getName()
+ "\"?")
              .setNegativeButton("Отмена", (dialog, which) -> {
                dialog.dismiss();
              })
              .setPositiveButton("Включить", (dialog, which) -> {
                binding.getVm().setScenario(((ScenarioEntity) object));
                dialog.dismiss();
              })
              .create();


          alertDialog.show();
        }
      }
      public class MainActivity extends AppCompatActivity
          implements NavigationView.OnNavigationItemSelectedListener {
        HomeFragment homeFragment;
        ScenarioFragment scenarioFragment;
```

```
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    setTitle("Управление");
    homeFragment = new HomeFragment();
    scenarioFragment = ScenarioFragment.newInstance();

    FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
    ft.replace(R.id.main_frame, homeFragment);
    ft.commit();

    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
        this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
    drawer.addDrawerListener(toggle);
    toggle.syncState();

    NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
    navigationView.setNavigationItemSelectedListener(this);
    navigationView.setCheckedItem(R.id.nav_control);
//    navigationView.item
  }

  @Override
  public void onBackPressed() {
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
      drawer.closeDrawer(GravityCompat.START);
    } else {
      super.onBackPressed();
    }
  }

  @Override
  public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();
    if (id == R.id.nav_control) {
      FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
      ft.replace(R.id.main_frame, homeFragment);
```

```
        ft.commit();
        setTitle("Управление");
    } else if (id == R.id.nav_scenario) {
        FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
        ft.replace(R.id.main_frame, scenarioFragment);
        ft.commit();
        setTitle("Режимы");
    } else if (id == R.id.nav_manage) {
    }  else if (id == R.id.nav_send) {
    }

    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
  }
}

public class HomeFragment extends Fragment implements OnItemClickAction, DialogCallback {
  HomeFragmentViewModel viewModel;
  FragmentHomeBinding binding;


  public HomeFragment() {
    // Required empty public constructor
  }

  public static HomeFragment newInstance(String param1, String param2) {
    HomeFragment fragment = new HomeFragment();
    return fragment;
  }

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
    Log.d("pepsi", "onCreateView: " + isAdded());
    setHasOptionsMenu(true);
    binding = DataBindingUtil.inflate(inflater, R.layout.fragment_home, container, false);
    viewModel = ViewModelProviders.of(this, new Factory(this)).get(HomeFragmentViewModel.class);
    getLifecycle().addObserver(viewModel);
    binding.setVm(viewModel);
    binding.rvSensors.setLayoutManager(new GridLayoutManager(getContext(), 2));
    binding.getVm().getLiveString().observe(this, s -> {
      Toast.makeText(getContext(), s, Toast.LENGTH_SHORT).show();
```

```
        });
        return binding.getRoot();
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        super.onCreateOptionsMenu(menu, inflater);
        getActivity().getMenuInflater().inflate(R.menu.main, menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_save) {
            View customView = getLayoutInflater().inflate(R.layout.dialog_scenario, null);
            EditText etAmount = customView.findViewById(R.id.et_amount);
            AlertDialog alertDialog = new AlertDialog.Builder(getContext(),
THEME_DEVICE_DEFAULT_DARK)
                    .setView(customView)
                    .setNegativeButton("Отмена", (dialog, which) -> {
                        dialog.dismiss();
                    })
                    .setPositiveButton("Сохранить", null)
                    .create();

            alertDialog.setOnShowListener(dialog -> {
                Button button = (alertDialog).getButton(AlertDialog.BUTTON_POSITIVE);
                button.setOnClickListener(v -> {
                    if (!TextUtils.isEmpty(etAmount.getText().toString().trim())) {
                        viewModel.saveScenario(etAmount.getText().toString().trim());
                        dialog.dismiss();
                    }
                });
            });

alertDialog.getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_VISIBLE);
            alertDialog.show();
            return true;
        }

        return super.onOptionsItemSelected(item);
```

```
    }

    @Override
    public void onClick(Object object) {
        BottomActionFragment fragment = BottomActionFragment.newInstance((SensorEntity) object);
        fragment.show(getChildFragmentManager(), fragment.getTag());
    }


    @Override
    public void dialogCallback(SensorEntity sensorEntity) {
        Log.d("pepsi", "onCreateView: adaad");
        viewModel.updateSensor(sensorEntity);
    }
}

public class HomeFragmentViewModel extends ObservableViewModel implements LifecycleObserver{

    private ArrayList<SensorEntity> list;
    private SensorItemAdapter adapter;
    private OnItemClickAction action;

    private MutableLiveData<String> liveString;

    public HomeFragmentViewModel() {
        loadData();
        Log.d("pepsi", "HomeFragmentViewModel:");
    }

    public HomeFragmentViewModel(OnItemClickAction action) {
        this.action = action;
        liveString = new MutableLiveData<>();
        Log.d("pepsi", "HomeFragmentViewModel: action");
    }

    public List<SensorEntity> getList() {
        return list;
    }

    public void setList(ArrayList<SensorEntity> list) {
        this.list = list;
    }
```

```
public MutableLiveData<String> getLiveString() {
  return liveString;
}

@Bindable
public SensorItemAdapter getAdapter() {
  return adapter;
}

public void setAdapter(SensorItemAdapter adapter) {
  this.adapter = adapter;
  notifyPropertyChanged(BR.adapter);
}

public OnItemClickAction getAction() {
  return action;
}

public void setAction(OnItemClickAction action) {
  this.action = action;
}

@SuppressLint("CheckResult")
@OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
public void loadData(){
  RetrofitManager.getInstance().getAllSensors()
      .subscribeOn(Schedulers.io())
      .observeOn(AndroidSchedulers.mainThread())
      .subscribe(sensorEntities -> {
        setList(sensorEntities);
        setAdapter(new SensorItemAdapter(list, R.layout.item_sensor, action));
      }, throwable -> liveString.setValue("Возникли проблемы с Интернет соединением"));

}

public void updateSensor(SensorEntity sensorEntity){
  list.set(list.indexOf(sensorEntity), sensorEntity);
  adapter.notifyDataSetChanged();
}

@SuppressLint("CheckResult")
public void saveScenario(String name){
  Completable.fromAction(() -> SmartHomeApp.getInstance().getDatabase()
```

```
        .scenarioEntityDao().insert(new ScenarioEntity(name, list)))
        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(()-> liveString.setValue("Режим успешно сохранен."),
            throwable -> liveString.setValue("Произошла ошибка, режим не сохранен"));
  }
}


public class RetrofitManager {
  private static RetrofitManager instance;

  private static final String LIFE_URL = "https://smarthouse-khnure.herokuapp.com";

  private ApiRequest.Sensors sensors;


  private RetrofitManager() {
    init();
  }



  public static synchronized RetrofitManager getInstance() {
    if (instance == null) {
      instance = new RetrofitManager();
    }
    return instance;
  }


  private void init() {
    HttpLoggingInterceptor loggingInterceptor = new HttpLoggingInterceptor()
        .setLevel(HttpLoggingInterceptor.Level.BODY);
    OkHttpClient client = new OkHttpClient.Builder()
        .addInterceptor(loggingInterceptor)
        .connectTimeout(30, TimeUnit.SECONDS)
        .readTimeout(30, TimeUnit.SECONDS)
        .writeTimeout(30, TimeUnit.SECONDS)
        .build();
    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(LIFE_URL)
        .client(client)
```

```
                .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
                .addConverterFactory(GsonConverterFactory.create())
                .build();

        sensors = retrofit.create(ApiRequest.Sensors.class);
      }


      public Single<SensorEntity> changeSensor(SensorEntity sensorEntity){
        return sensors.changeSensor(sensorEntity);
      }

      public Single<ArrayList<SensorEntity>> updateSensors(ArrayList<SensorEntity> list){
        return sensors.updateSensors(list);
      }
      public Single<ArrayList<SensorEntity>> getAllSensors(){
        return sensors.getAllSensors();
      }
    //



    }
    public class SensorItemViewModel extends BaseObservable {
      private SensorEntity sensor;
      private OnItemClickAction action;
      private PublishSubject<SensorEntity> subject;

      @SuppressLint("CheckResult")
      public SensorItemViewModel(SensorEntity sensor) {
        subject = PublishSubject.create();
        this.sensor = sensor;
        subject.debounce(3, TimeUnit.SECONDS)
                .switchMap(sensorModel ->
RetrofitManager.getInstance().changeSensor(sensorModel).toObservable())
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .retry()
                .subscribe(sensorModel -> {
                  this.sensor = sensor;
                  notifyPropertyChanged(BR.sensor);
                });
      }
```

```
public SensorItemViewModel(SensorEntity sensor, OnItemClickAction action) {
  this(sensor);
  this.action = action;
}

@Bindable
public SensorEntity getSensor() {
  return sensor;
}

public void setSensor(SensorEntity sensor) {
  this.sensor = sensor;
  notifyPropertyChanged(BR.sensor);
  notifyPropertyChanged(BR.state);
  subject.onNext(sensor);
}

@Bindable
public boolean getState() {
  return sensor.getState().equals(STATUS_ON);
}

@Bindable
public boolean isRange() {
  return sensor.getType().equals(TYPE_LIGHT) || sensor.getType().equals(TYPE_TEMPERATURE);
}


public void doAction(View view) {
  action.onClick(sensor);
}


@BindingAdapter("factory")
public static void imageFactory(ImageView imageView, String type) {
  int id = 0;
  switch (type) {
    case TYPE_CURTAIN:
      id = R.drawable.ic_window;
      break;
    case TYPE_LIGHT:
      id = R.drawable.ic_light;
      break;
```

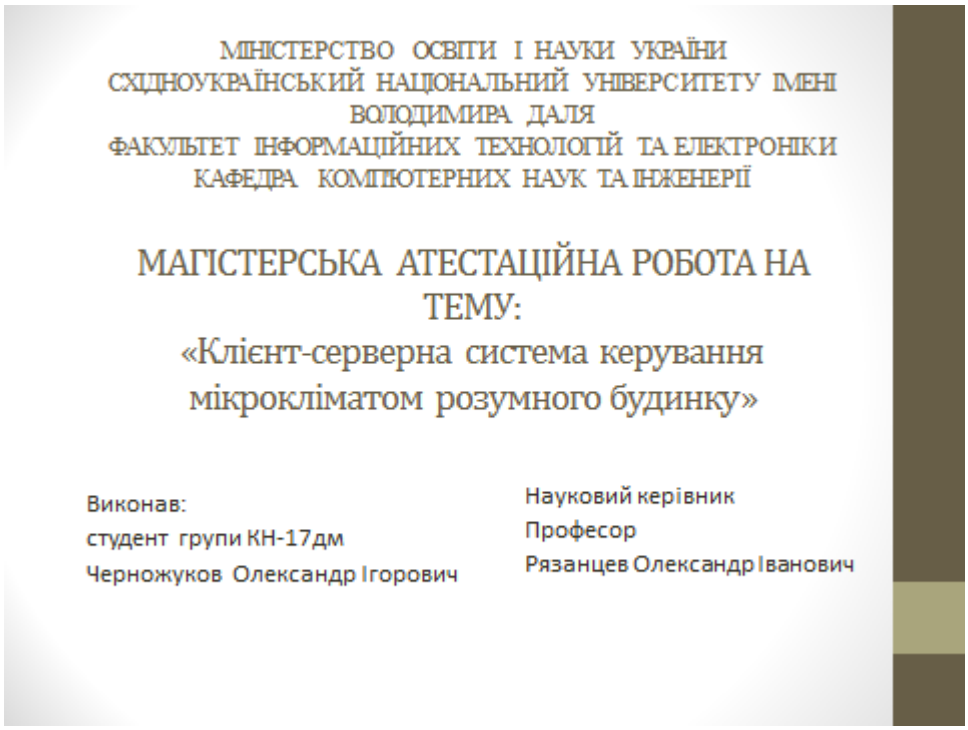
```
        case TYPE_TEMPERATURE:
          id = R.drawable.ic_temp;
          break;
        case TYPE_SECURITY:
          id = R.drawable.ic_security;
          break;
        case TYPE_ELECTRICITY:
          id = R.drawable.ic_electric;
          break;
      }
      imageView.setImageResource(id);
    }

    @BindingAdapter("background_factory")
    public static void backgroundFactory(RelativeLayout layout, String status) {
      int id = 0;
      switch (status) {
        case STATUS_ON:
          id = R.drawable.active_background;
          break;
        case STATUS_OFF:
          id = R.drawable.inactive_background;
          break;
        case STATUS_OUT:
          id = R.drawable.out_background;
          break;
      }
      layout.setBackgroundResource(id);
    }
  }


  public class BottomActionFragment extends BottomSheetDialogFragment {

    // TODO: Rename and change types of parameters
    private SensorEntity sensor;
    private DialogCallback callback;


    public BottomActionFragment() {
      // Required empty public constructor
    }
```

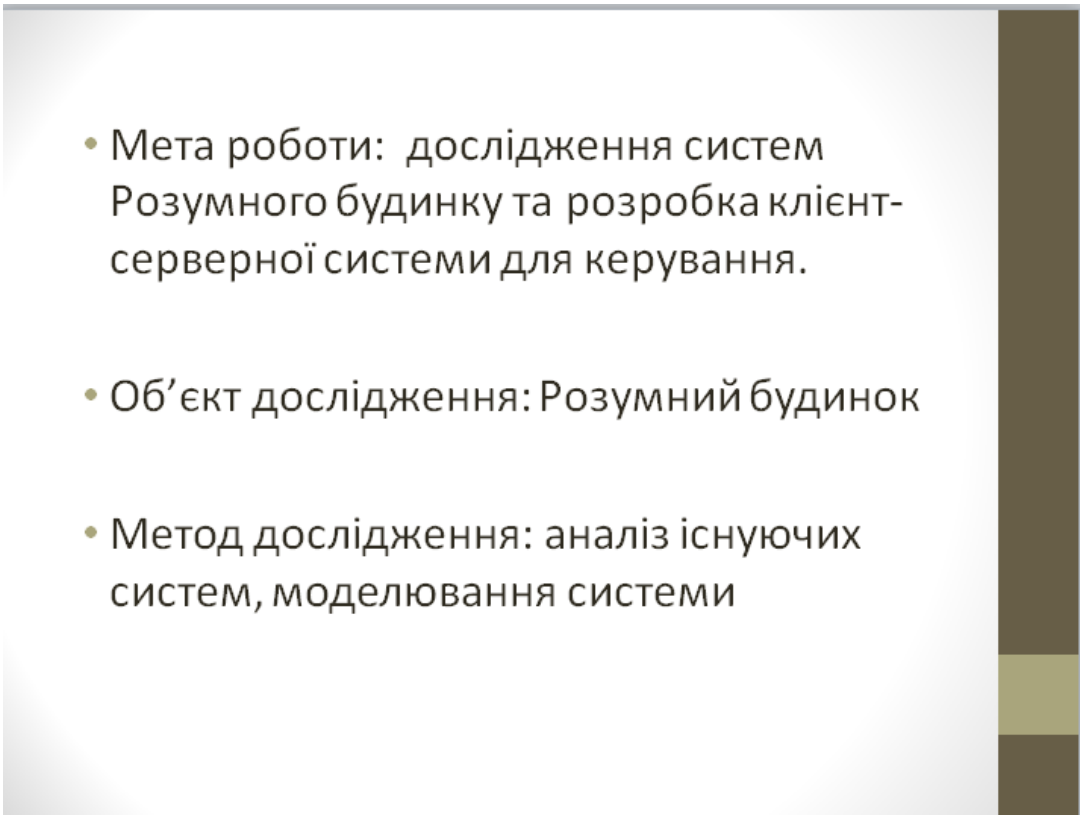
```
/**
 * Use this factory method to create a new instance of
 * this fragment using the provided parameters.
 *
 * @return A new instance of fragment BottomActionFragment.
 */
// TODO: Rename and change types and number of parameters
public static BottomActionFragment newInstance(SensorEntity sensor) {
  BottomActionFragment fragment = new BottomActionFragment();
  Bundle args = new Bundle();
  args.putSerializable("sensor", sensor);
  fragment.setArguments(args);
  return fragment;
}

@Override
public void onAttach(Context context) {
  super.onAttach(context);
  callback = (DialogCallback) getParentFragment();
}

@Override
public void onDetach() {
  super.onDetach();
  callback = null;
}

@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  if (getArguments() != null) {
    sensor = (SensorEntity) getArguments().getSerializable("sensor");
  }
}



@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
  // Inflate the layout for this fragment
  FragmentDialogBinding binding = DataBindingUtil.inflate(inflater, R.layout.fragment_dialog,
container, false);
```

```
    binding.setVm(new SensorItemViewModel(sensor));
    binding.executePendingBindings();

    ToggleButton toggleButton = binding.toggle;
    toggleButton.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
      @SuppressLint("ResourceAsColor")
      @Override
      public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

        if (isChecked) {
          sensor.setState(STATUS_ON);
        } else {
          sensor.setState(STATUS_OFF);

        }
        binding.getVm().setSensor(sensor);
        Log.d("pepsi", "onCheckedChanged: " + sensor);
      }
    });

    DiscreteSeekBar seekBar = binding.getRoot().findViewById(R.id.seek);
    seekBar.setMax(binding.getVm().getSensor().getType().equals(TYPE_TEMPERATURE) ? 50 : 100);
    seekBar.setProgress(binding.getVm().getSensor().getValue());
    seekBar.setOnProgressChangeListener(new DiscreteSeekBar.OnProgressChangeListener() {
      @Override
      public void onProgressChanged(DiscreteSeekBar seekBar, int value, boolean fromUser) {
        sensor.setValue(value);
        binding.getVm().setSensor(sensor);
        Log.d("pepsi", "onProgressChanged: " + binding.getVm().getSensor());
      }

      @Override
      public void onStartTrackingTouch(DiscreteSeekBar seekBar) {

      }

      @Override
      public void onStopTrackingTouch(DiscreteSeekBar seekBar) {

      }
    });
    return binding.getRoot();
  }
```

```
  @Override
  public void onDismiss(DialogInterface dialog) {
    super.onDismiss(dialog);

    callback.dialogCallback(sensor);
    dialog.dismiss();
  }

}
```

ДОДАТОК Б.
Електронні плакати





Актуальність дослідження

- Зараз ми живемо в світі, де більшість повсякденних завдань спрощені або автоматизовані і з кожним роком ця тенденція зростає. У побут сучасної людини щільно увійшли технології віддаленого управління. Ці технології допомагають не тільки економити час, а й дозволяють не залежати від місцезнаходження.
- Зростання популярності автоматизованих систем, таких як розумний будинок, обумовлено прагненням людини до комфорту і зручності. Додатковою привабливістю є безпека, будь то протипожежна система або сигналізація з дистанційним оповіщенням. Розумний будинок є сучасним інструментом підвищення рівня комфорту і життя, так як частина процесів відбувається автоматично, а решті можна керувати віддалено, що робить її актуальною для вивчення і вдосконалення.

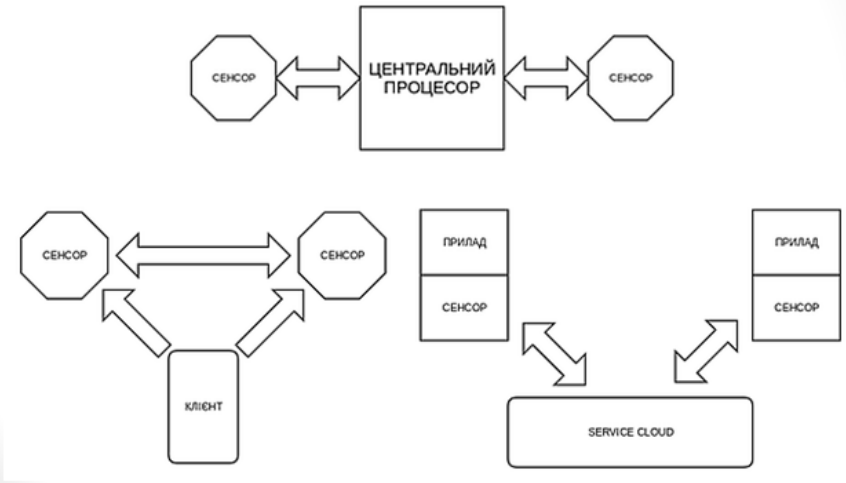
Огляд систем розумного будинку

«Розумний будинок» є сукупністю стандартів об'єднаних з різного роду приладами в систему і інтеграцію декількох систем в єдину систему управління будівлею. Системи бувають такі:

- система мікроклімату
- система безпеки
- система електроживлення
- система зв'язку
- система віддаленого управління.

Система «Розумний Дім» надає можливість управління в режимі реального часу за допомогою будь-якого мобільного пристрою або ПК, розташованого в локальній мережі або має доступ в Інтернет.

Варіанти реалізації розумного будинку



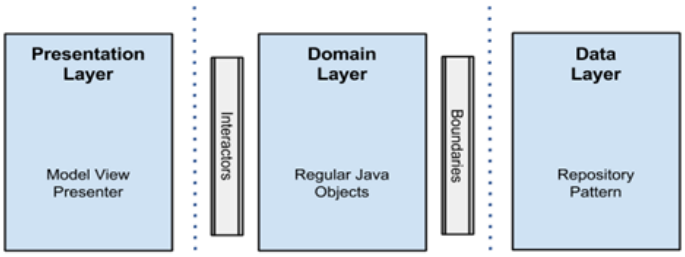
Способи зв'язку між компонентами

	Wi-Fi	Bluetooth	ZigBee
Диапазон (м)	20-300	10-100	10-100
Споживання потужності	Високий рівень	Низький рівень	Дуже низький рівень
Час безперервної роботи від батареї	0,5-5	1-10	100-1000
Максимальна кількість вузлів	10	7	65536
Передбачувана область застосування	Передача відеопотоку з камер	Передача аудіо сигналу (домашній кінотеатр, аудіосистема)	Бездротове з'єднання між датчиками і обчислювальною системою

Роль мобільного додатку

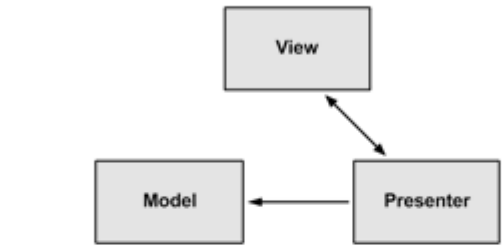
- Управління «Розумним домом» за допомогою смартфонів або планшетів, в яких використана платформа Android, дає можливість постійно мати при собі мобільний пристрій управління, яке призначений для організації зв'язку з інтернетом, ділових записів, дзвінків та інших функцій.

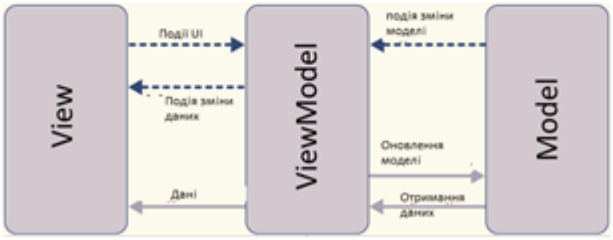
Дослідження сучасних архітектур для розробки мобільних додатків



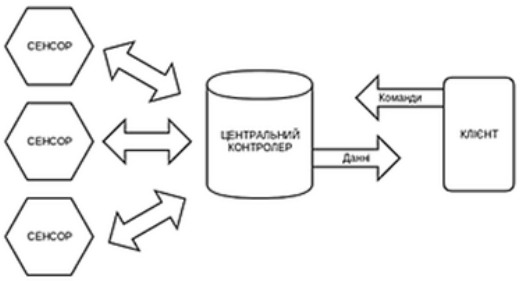
- шар даних (Data Layer) - даний шар відповідає в першу чергу за отримання даних з різних джерел і їх кешування. Він реалізується за рахунок паттерна Repository.
- шар бізнес-логіки (Domain Layer) - на цьому шарі міститься вся бізнес-логіка програми. Цей шар є певним об'єднанням шарів сценаріїв взаємодії і бізнес-логіки в оригінальній «чистій архітектурі».
- шар уявлення (Presentation Layer) відповідає за логіку відображення даних на екрані, за взаємодію з користувачем і за інші процеси, пов'язані з UI. Цей шар не повинен містити логіку додатка, не пов'язану з UI.

MVP vs MVVM





Проектування системи

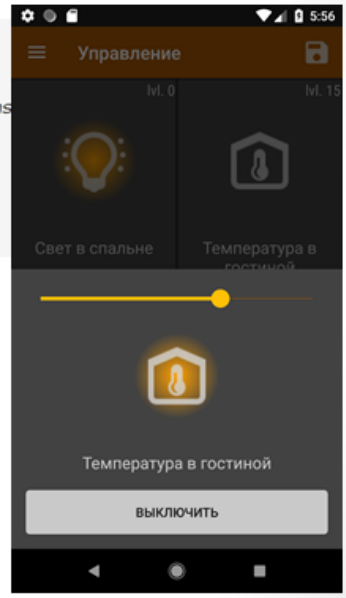


Реалізація


```
@SuppressLint("CheckResult")
@OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
public void loadData(){
    RetrofitManager.getInstance().getAllSensors()
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(sensorEntities -> {
                setList(sensorEntities);
                setAdapter(new SensorItemAdapter(list, R.layout.item_sensor, action));
            }, throwable -> liveString.setValue("Возникли проблемы с Интернет соединением"));
}
```

Реалізація

```
public SensorItemViewModel(SensorEntity sensor) {
    subject = PublishSubject.create();
    this.sensor = sensor;
    subject.debounce( timeout: 3, TimeUnit.SECONDS)
            .switchMap(sensorModel -> RetrofitManager.getIns
    .subscribeOn(Schedulers.io())
    .observeOn(AndroidSchedulers.mainThread())
    .subscribe(sensorModel -> {
        this.sensor = sensor;
        notifyPropertyChanged(BR.sensor);
    });
}
```



$$\mu = kp\varepsilon,$$

μ - керуючий вплив, який націлений на виключення відхилення регульованого значення від заданого.

ε - вхідний вплив на регулятор, який визначається, як відхилення регульованого значення від поточного впливу;

kp - коефіцієнт передачі

Після запуску системи управління, дані з датчика температури t вим надходять на сервер. Температура може бути вище або нижче встановленої t зад. Залежно від цього, сервер відправляє максимально можливий сигнал интерфейсному модулю керуючому пристрою, після чого він адресує його відповідному порту виконавчого модуля, до якого підключені нагрівальне або охолоджувальний пристрій.

Основні результати роботи

- проведено аналіз систем розумного дому, засобів бездротового зв'язку між компонентами розумного будинку.
- проведене дослідження законів регулювання.
- проведено дослідження сучасних рішень з розробки мобільних додатків
- змодельовано систему керування розумним домом
- на основі отриманих знань було реалізовано клієнт-серверну систему керування розумним домом