Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему: Удосконалення методів та інструментів створення адаптивного інтерфейсу для управління пристроями в системі Smart Home

Виконав: студент II курсу
групи 1ПІ-18 м
спеціальності
121 – Інженерія програмного
забезпечення
(шифр і назва напряму підготовки, спеціальності)
Підгородецький Р. І.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ
Коваленко О.О.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. КН
Арсенюк І.Р.
(прізвище та ініціали)

ВНТУ – 2019

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Освітньо-кваліфікаційний рівень – магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
"____"_____2019 року

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

Підгородецькому Роману Ігоровичу

1. Тема роботи: «Удосконалення методів та інструментів створення адаптивного інтерфейсу для управління пристроями в системі Smart Home», керівник роботи: Коваленко Олена Олексіївна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від "__"_____2019 року №___

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи: міжнародні стандарти та протоколи управління пристроями; операційна система Android OS, Тактова частота процесору 1.3 GHz, Ядра ЦПУ 2-8 АКМз типом взаємдії BIG.little. Форма реалізації - у вигляді веб-додатку та Android додатку.

4. Зміст розрахунково-пояснювальної записки: вступ, аналіз предметної області, розробка моделей системи управління розумним домом; програмне забезпечення взаємодії приладів Smart Home, економічна частина, висновки. список використаної джерел, додатки.

5. Перелік графічного матеріалу: моделі управління системою розумного дому, модель адаптивного інтерфейсу управління пристроями та

програмними агентами; модель дизайну для веб і мобільних додатків, інтерфейс користувача, система клімат-контролю.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Коваленко О. О., д.т.н., завідувач кафедр ПЗ		
5	Бальзан М. В., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз теоретико-практичних підходів до побудови адаптивних інтерфейсів реалізації технології «Розумний будинок»	04.09.2019 – 14.09.2019	Вик.
2	Моделювання системи управління розумним будинком	15.09.2019 – 20.10.2019	Вик.
3	Програмне забезпечення взаємодії приладів розумного дому	21.10.2019 – 15.11.2019	Вик.
4	Програмне забезпечення мобільного додатку системи розумного дому	16.11.2019 – 21.11.2019	Вик.
5	Економічна частина	22.11.2019 – 01.12.2019	Вик.

Студент _____ **Підгородецький Р. І.**

(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Коваленко О.О.**

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі «Удосконалення методів та інструментів створення адаптивного інтерфейсу для управління пристроями в системі Smart Home» розроблено адаптивний інтерфейс для управління пристроями в системі розумного дому. В ході виконання роботи було проаналізовано існуючі методи та підходи до формування програмного додатку управління пристроями. Реалізовано тестовий приклад інтерфейсу управління системою клімат-контролю.

ABSTRACT

Master's qualification thesis "Improvement of methods and tools for creating an adaptive interface for managing devices in the Smart Home system" developed an adaptive interface for managing devices in the smart home system. The results of the study provide an analysis of existing methods and approaches to developing a device management software application. A test example of the climate control system management interface is implemented.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний етап розвитку програмного забезпечення розумного дому характеризується швидким прогресом та управлінням за допомогою смартфонів, спеціальних пультів, інтегрованих пристроїв управління тощо [1]. Під «розумним будинком» розуміють мережу пристроїв, які виконують рутинні завдання автономно, без взаємодії з людиною. Серед таких завдань – включення та виключення світла, зміна температури в кімнаті, включення та виключення електроприладів, нагадування про завдання тощо. Сьогодні вже можна говорити про використання голосових асистентів Apple HomePod, Google Home and Amazon Echo, які стають все більш масовими. Виробники різноманітних пристроїв вже передбачать сенсори та технології Wi-Fi, ZigBee для взаємодії різних пристроїв та контролю їх роботи. Але запровадження розумних пристроїв супроводжується такими проблемами як відсутність сумісного для всіх пристроїв та зручний інтерфейс, доступної ціни; забезпечення повноцінної безпеки даних [2].

Як правило, користувач очікує простого інтуїтивного інтерфейсу з простими налаштуваннями та подальшим безпроблемним функціонуванням. Розумні будинки поки що не мають напрацьованих вимог та стандартів щодо юзабіліті або якості інтерфейсу, інтеграції роботи різних пристроїв. Велика частина часу для користувача навіть при налаштуванні одного приладу припадає на установку мобільного додатка для розумного пристрою (наприклад, світла), створення свого аккаунту, підключення до мережі Wi-Fi, а потім – самого пристрою. Інтеграція різних пристроїв також вимагає спеціальної адаптації. Серед різноманітних нових підходів можна виділити такі як програмний додаток «proactive discovery», який автоматично перевіряє наявність розумних пристроїв в радіусі житла та підключає їх. Такий підхід дозволив знизити кількість кроків налаштування з 13 до 2, а час налаштування зменшити до 1 хвилини [1-5].

Питання формування дружнього інтерфейсу завжди були актуальними як для різноманітних систем автоматизації промислової діяльності, так і для інтернет речей, систем управління житлом, технікою, модулями управління оплюванням, світлом тощо. Але розрізнені модулі управління можуть бути несумісні між собою і потребують спеціальних підходів до формування інтерфейсу управління. Саме це обумовило необхідність розробки адаптивного інтерфейсу для управління пристроями в системі розумного дому, що передбачає удосконалення методів та інструментів для його створення.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Основною метою магістерської роботи є удосконалення методів формування адаптивного інтерфейсу для підвищення якості управління пристроями в системі розумного дому (Smart Home).

Основними задачами дослідження є:

- провести аналіз існуючих підходів, методів та інструментарію управління пристроями та програмними агентами розумного дому;
- запропонувати удосконалення для формування інтуїтивного та дружнього інтерфейсу;
- провести аналіз та удосконалення методів формування адаптивного інтерфейсу в системі розумного дому (Smart Home);
- розробити моделі та варіанти реалізації системи управління розумного дому та її інтерфейсу;
- розробити програмні компоненти імітаційної системи управління розумним домом;
- провести експериментальні дослідження та тестування програмних модулів на прикладі системи клімат-контролю.

Об’єкт дослідження – процес побудов інтерфейсу управління пристроями розумного дому (імітаційна програма на прикладі реалізації клімат-контролю) та її програмна реалізація.

Предмет дослідження – методи та інструментарій формування адаптивного інтерфейсу системи управління пристроями розумного дому.

Методи дослідження. У процесі досліджень використовувались: методи аналізу існуючих науково-практичних підходів щодо побудови інтерфейсів системи управління пристроями розумного дому, синтезу архітектури та моделей управління пристроями та програмним забезпеченням розумного дому; алгоритмізації основних процесів управління модулями розумного дому; комп’ютерного моделювання та програмної реалізації для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Одержала подальший розвиток модель адаптивного інтерфейсу користувача, яка, на відміну від існуючої, відповідає вимогам користувача, які записані в його профілі та змінюється відносно визначених модулів, підключених пристроїв та за вибором користувача.

2. Запропоновано метод аналізу дизайну інтерфейсу на основі аналізу респонсивного та адаптивного методів дизайну з можливістю підрахунку кількості приладів та подальших рекомендацій щодо варіантів формування дизайну інтерфейсу.

3. Набули подальшого розвитку методи формування адаптивного інтерфейсу управління пристроями в системі управління розумним будинком відповідно вимог до кожного окремого модулю та системи в цілому з врахуванням ризиків несумісності та колізій в роботі пристроїв та програмних агентів.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень та виконаного імітаційного моделювання роботи окремого модулю була виконана реалізація модулю

клімат-контролю та варіанти інтерфейсу системи управління. Одержані результати можуть бути використані в подальших розробках системи управління розумним будинком.

Впровадження. Впровадження результатів досліджень підтверджуються довідкою використання одержаних результатів у навчальному процесі при викладанні курсу «Основи програмної інженерії» для студентів спеціальності 121 – «Інженерія програмного забезпечення» як прикладу проєкту запровадження адаптивного інтерфейсу управління пристроями розумного дому.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: метод формування адаптивного інтерфейсу на основі існуючих модулів та пристроїв розумного дому; комп’ютерна програма для проведення імітаційного експерименту роботи розумного дому.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи були представлені на конференції Молодь в науці: дослідження, проблеми, перспективи.

Публікації. Основні результати досліджень опубліковані в матеріалах конференції ВНТУ Коваленко О.О., Підгородецький Р.І. Програмний інструментарій для управління пристроями розумного дому. Інформаційні технології та автоматизація, ВНТУ, 2020 р.

Структура та обсяг роботи. Магістерська кваліфікаційна роботи складається зі вступу, чотирьох розділів, висновків, списку літератури, що містить 45 найменувань, 3 додатків. Робота містить 15 ілюстрацій, 3 таблиці.

1 АНАЛІЗ ТЕОРЕТИКО-ПРАКТИЧНИХ ПІДХОДІВ ДО ПОБУДОВИ АДАПТИВНИХ ІНТЕРФЕЙСІВ РЕАЛІЗАЦІЇ ТЕХНОЛОГІЇ «РОЗУМНИЙ БУДИНОК»

1.1 Технологія розумний дім (Smart Home) та відомі методи формування інтерфейсу для управління системою

Технології розумного дому реалізують систему автоматизації процесів реалізації побутових задач, охорони, економічного використання ресурсів, нагадування, управління різноманітними пристроями. Загальна концепція технологій базується на сучасних інформаційних технологіях і потребах споживачів [2] (Див. Рис. 1).

Рисунок 1.1. – Загальна концепція управління розумним домом.

Корені розвитку технологій розумного дому відносяться до 1966 року, коли Джеймс Сазерленд створив програму включення приладів за розкладом, а також реалізував зв'язок з датчиками моніторингу та сигналізації,

використавши датчики братів Д. та Р. Спірам. Які були запатентовані в 1961 році.

В 1978 році було розроблено перший стандарт передачі даних для приладів домашньої автоматизації, який став основою автоматичних домашніх приладів.

В 1984 році термін «розумний будинок» став традиційним для Американської асоціації житлово-будівельних компаній та асоціації електронної промисловості, яка ініціювала протокол CEBus – шина для споживачів.

Починаючи з 2000 року системи домашньої автоматизації стали використовувати різноманітні мобільні пристрої. Крім систем домашньої автоматизації активно стали розвиватись системи розумних будівель, офісної автоматизації.

Серед різноманітних методів формування інтерфейсу управління розумним домом можна відокремити такі як: формування природного інтерфейсу; узгодженості в межах використовуваних приладів; дружнього інтерфейсу; метод гаманця Міллера [5; 6].

Розглянемо особливості кожного з наведених методів.

Природний інтерфейс передбачає інтуїтивно зрозумілу для користувача систему повідомлень та результатів, що не потребує спеціальних пояснень, а використовує зрозумілі поняття та образи (метафори). Але, такий інтерфейс ускладнюється разом з ускладненням системи, додаванням нових приладів, програмних модулів, а також не враховує досвід використання приладів до впровадження системи.

Метод узгодженості інтерфейсу може бути використаний в умовах запровдаження системи розумного дому та самонавчання системи на основі спакоємності знань та навичок на рівні операційної системи, відомих інтерфейсів, метафор (корзина, запуск, прилад).

Метод дружнього інтерфейсу передбачає визначення відповідного набору дій та формування повідомлень щодо ситуацій, які шкодять системі та

рекомендацій щодо подальших дій користувача. Причому такі рекомендації повинні бути зрозумілі та чіткі. Ефективний дружній інтерйфейс адаптується до помилок користувача та полегшує алгоритми роботи користувача з пристроями та системою в цілому. Такий інтерфейс повинен бути простим, гнучким, мати зворотній зв'язок та естетичну привабливість.

Метод гаманця Мілера дозволяє розрахувати кількість елементів в інтерфейсі (їх повинно бути від семи до дев'яти) та допомагає групувати елементи в програмі з урахуванням цього правила – тобто не більше семи-дев'яти закладок, опцій, кнопок в групі. Мінімілістичні дизайни зменшують цю кількість до п'яти.

Реальні приклади розумного дому можна розглядати на різних рівнях. Максимально автоматизовані дома Білла Гейтса та Марка Цукерберга, сформовані розумні дома та приклади розумних офісних будівель в науково-дослідних інститутах США та Європи – найвищий рівень домашньої та офісної автоматизації. Використання окремих модулів – розумного освітлення, опалення, управління кліматом, систем нагадування та безпеки – локальне використання систем розумного дому.

Розглянемо основні модульні системи. Одна з найпопулярніших – система управління освітленням. Такі системи дозволяють виконувати різноманітні функції та використовувати фірмові прилади. Так, наприклад, автоматизація роботи жалюзі, штор, віконниць і навісів здійснюється за допомогою спеціальних штор та карнизів Lutron; моніторинг появи людини в будинку або в кімнаті за допомогою розумних ламп Philips; програмні додатки здійснюють регулювання яскравості в залежності від індикаторів датчиків щодо кількості людей в приміщенні, часу доби, включення проектору або телевізору; імітація присутності господарів в будинку забезпечується лампою BeON; подієве оповіщення здійснює лампа Xiaomi Philips EyeCare 2; комплект ламп Nanoleaf Aurora Smarter Kit призначений для реалізації світлових сценаріїв; реалізовані також спеціальні алгоритми щодо голосового та віддаленого управління.

Вже існують технології запровадження кольорової гамми освітлення, роботи від акумулятора, також розроблений розумний патрон для лампи Vocca, який керується голосом.

Системи HVAC призначені для підтримки комфортних для життя температури, вологості та чистоти повітря. Такі системи клімат контролю дозволяють підтримувати комфортну температуру, регулювання інтенсивності обігріву, регулювання роботи зволожувачів, осушувачів та іонізаторів повітря, відповідають за автоматичну роботу кондиціонерів, фанкойлів, вентиляторів і систем подачі свіжого повітря (Keen Smart Vens); водопровідних систем [3]. Приклад роботи такої системи представлено на рис. 1.2.

Рисунок 1.2. – Загальна система управління мікрокліматом [2]

Головна проблема сучасних систем Smart Home полягає у відсутності єдиного стандарту та сумісності пристроїв розумного дому. Крім того, обладнання може супроводжуватись програмною системою з закритим кодом і це також не дозволяє використовувати різноманітні пристрої в єдиній системі. у

Найбільш універсальними протоколами є протоколи Z-Wave, і ZigBee, які є відкритими, працюють на основі коміркової мережі, зв'язку приладів між собою та на частоті до 1ГГц.

Навколо обох протоколів вже сформувались своєрідні альянси, в які входять виробники систем для розумних будинків. Стандартів для інтерфейсів розумного дому не існує. Але дизайнери та розробники використовують стандарти дружніх інтерфейсів, юзабіліті. Так, міжнародний стандарт ISO 9241-11 визначає юзабіліті як «ступінь, з якою продукт може бути використаний певними користувачами при певному контексті використання для досягнення певних цілей з належною ефективністю, продуктивністю і задоволеністю» [3].

Адаптивним інтерфейсом називають інтерфейс, що має можливість регулювання вигляду, властивостей відповідно до вимог користувача та визначених індикаторів. На жаль, часто поняття «адаптивний інтерфейс використовується дуже вузько, як адаптація до різних платформ та параметрів екранів, десктопних та мобільних пристроїв. Саме це обумовлює розвиток розуміння адаптивного інтерфейсу як інтерфейсу, що має гнучку архітектуру формування інформаційних панелей управління.

1.2. Принципи роботи та концепції управління системи Smart Home

Загальна робота розумного дому базується на принципі виконання команди. Команди формуються від людини з пульту управління, а також від датчиків та програмних агентів. Останнім часом активно розвиваються системи голосових команд, можливі команди зі смартфону [2].

До основних принципів роботи системи розумного дому відносять:

1. Принцип корисності;
2. Принцип сумісності;
3. Принцип простоти у використанні;
4. Принцип автоматизації визначених завдань та сценаріїв;

Принципи роботи системи розумного дому відповідають принципам побудови адаптивного інтерфейсу, мета якого підтримувати корисність автоматизації, сумісність всіх приладів, бути простим у використанні та надавати користувачу можливість вибору визначених сценаріїв роботи.

На рис. 1.3. представлено загальну систему датчиків, розумних виконавців, пристроїв, що входять в систему управління.

Рисунок 1.3. – Командне управління в системі розумного дому

Програмні агенти у взаємодії з датчиками та у відповідності з визначеними сценаріями та алгоритмами працюють відповідно до параметрів приладів, їх включення, регуляції тощо. На рис. 1.4. представлено центральний контролер для пристроїв розумного дому фірми Xiaomi [4].

Система розумного дому містить три головних модулі:

система датчиків зовнішнього середовища;

центральний контролер, який обробляє інформацію та приймає рішення;

прилади, які виконують вибрані сценарії та завдання.

Системи розумного дому можуть бути централізованими та нецентралізованими. Найкраще використовувати змішану систему, в якій рівень централізації вибирає сам користувач. Більшу безпеку надають нецентралізовані системи.

Рисунок 1.4. – Центральний контролер для пристроїв розумного дому Xiaomi

Всі компоненти системи розумного будинку можуть бути з’єднані між собою за допомогою провідного та безпровідного зв’язку (Див. Рис. 1.5.). Не дивлячись на те, що сьогодні більш популярним є використання протоколів Bluetooth, Wi-Fi, більш надійним є рішення на основі кабельного зв’язку, а також спеціальних протоколів безпровідних систем. Змішана система з’єднання ж найбільш популярною, але вона, як і безпровідна містить такі недоліки як низький рівень надійності та ризики колізій не тільки сумісності пристроїв розумного дому, а і пристроїв зв’язку.

Концепції управління можна поділити на три групи [5].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;
- інтегровані системи.

Перша група є, як правило, налаштованою виробником і працює від центрального процесора. Всі налаштування зберігаються на сервері, принцип

роботи представлений на рис. 1.5.

Рисунок 1.5 – Принцип роботи вбудованної системи з центральним контролером

Друга група є системою взаємодії напівавтономних пристроїв. Це дозволяє перепрограмувати систему і встановлювати зв’язки між пристроями безпосередньо. Принцип роботи наведено на рис. 1.6.

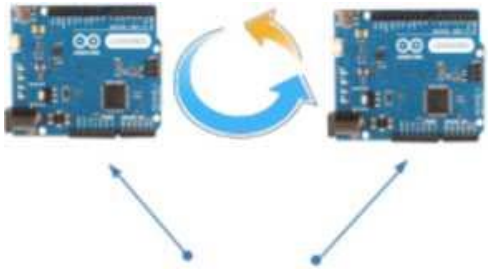


Рисунок 1.6 – Принцип роботи вбудованої системи без центрального контролера

Третя група – зовнішні контролери для інтеграції роботи приладів, що мають вбудовані алгоритми інтеграції та використовують спеціальні зовнішні датчики та сенсори. Принцип роботи такої системи представлено на рис. 1.7.

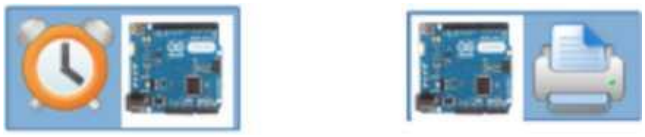


Рисунок 1.7 – Принцип роботи системи інтегрованого управління

Ідеальний варіант реалізації системи розумного дому – формування екосистеми взаємодії приладів, протоколів зв’язку за допомогою зручної системи управління [7-9]. Порівняння рішень наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння існуючих варіантів реалізації системи «Розумний дім»

Параметр	Meizu	Allone	Clipsal
Вартість	Середня	Висока (разом з модулями)	Середня
Установка	Проста	Проста	Попереднє налаштування
Налаштування	Не потребує	Через веб-сервіс	Програмування та перепрошивка
Готові модулі	Недостатньо	Достатньо	Не потребує
Масштабованість	Масштабується	Масштабується, залежить від розміру приміщення	Масштабується
Взаємодія компонентів	Через смартфон	Через сам пристрій	Через хмарний додаток
Функціонал	Базовий	Відкритий, майже необмежений	Відкритий, майже необмежений

Найбільш поширеними концепціями формування системи розумного дому є формування мультимедійного простору з загальним хабом пам'яті для контенту і розширеними можливостями взаємодії; концепція інтегрованої вбудованої техніки та взаємодії між приладами; змішана система управління зі спеціальними програмними агентами роботи пристроїв за різними протоколами. Прикладами реалізації змішаної системи є «розумний будинок»

від Meizu, Allone, Clipsal [7-9]. Аналіз існуючих концепцій та реалізацій дозволяє зробити висновок, щодо розвитку систем управління через хмарні та мобільні додатки з вирішенням питань узгодження протоколів взаємодії.

1.3. Підходи до дизайну панелей управління розумним домом

Принципи побудови моделей управління розумного дому базуються на таких видах дизайну як респонсивний та адаптивний. Такий дизайн враховує, як буде виглядати інтерфейс для користувача на різних пристроях і чи зручно буде ним користуватися. Обидва ці підходи передбачають адаптацію відображення контенту до різних розмірів екранів, але досягається ця мета різними засобами. Головна відмінність респонсивного і адаптивного дизайну полягає в наступному: Респонсивний дизайн передбачає створення додатків, які здатні "розтягуватися" під розмір екрану [10]. Тобто створюється єдина версія додатку, яка буде змінюватись в залежності від розміру екрану. Адаптивний дизайн передбачає створення структури для всіх версій додатку. Клієнтоорієнтований адаптивний дизайн передбачає формування панелей управління за запитами користувачів, надає вибір та можливості використання мінімалістських та розширених панелей управління. Технічні властивості обох видів дизайн передбачають збереження властивостей панелей управління: «іконки, значки, картинки і текст залишаються такими ж, і якщо переміщаються відносно один одного, то лише незначно». Але респонсивний дизайн передбачає використання водоспадної моделі в проектуванні, тобто врахування всіх підлаштувань в єдиній версії додатку. Адаптивний дизайн передбачає наявність певних наборів стилів, які відповідають за відображення контенту на різних пристроях в залежності від їх параметрів. Він забезпечує зручний збалансований інтерфейс. Кожен набір стилів, який використовується адаптивним додатком, розробляється таким чином, щоб користувачеві було максимально зручно і комфортно з ним працювати. Щоб домогтися цього, UX-експерти (від англ. "User experience" - призначений для користувача досвід)

досліджують найбільш популярні використовувані пристрої, поведінку користувачів, типові операції. Дизайнери враховують цю інформацію при роботі над кожною окремою версією адаптивного додатки. Потім розробники вибирають оптимальний набір стилів під певні діапазони параметрів пристроїв. Результатом цієї спільної роботи стає збалансований інтерфейс, точно підібраний функціонал і відчуття комфорту у користувача. Як показує практика, на оцінку сторінки користувачеві потрібно, в середньому, близько 2 секунд. Якщо протягом цього часу зображення на екрані не зможе привернути його увагу, він піде зі сторінки в пошуках більш цікавої інформації або дизайну. З цієї точки зору адаптивний дизайн є оптимальним рішенням і допоможе в залученні потенційних клієнтів. Для розумного дому важливо відчуття комфорту та наявність панелі управління на всіх пристроях користувачів. Є системи управління, що самонавчаються і змінюють дизайн. А також є можливість вибору панелей користувачем.

Рисунок 1.8. – Змішана система зв’язку

При всіх своїх перевагах респонсивний дизайн, який є ефективним для настільних комп'ютерів і ноутбуків, не в змозі забезпечити оптимальне відображення на мобільних пристроях з невеликими екранами і різними параметрами. Відповідно, адаптивний дизайн став своєрідною відповіддю на зміну потреб користувача. З адаптивним додатком користувач може

працювати, заходячи з будь-якого пристрою. Завдяки тому, що кожна версія візуально і функціонально адаптована під певні параметри пристрою, користувач не відчує різниці і не буде відчувати труднощів, використовуючи той або інший пристрій.

Адаптивний дизайн забезпечує більш високий ступінь комфорту для користувача завдяки оптимізації відображення і функцій програми під параметри вашого пристрою. І крім того, додаток зможе охопити більшу кількість пристроїв і, відповідно, більша кількість користувачів. [11-15]

Але, на жаль, жоден з описаних нами підходів не ідеальний. Як респонсивний, так і адаптивний дизайн мають свої недоліки, з якими доведеться зіткнутися при створенні додатків. Респонсивний дизайн дає вам можливість створити бюджетний додаток, який на будь-якому пристрої виглядає для користувача звичним і знайомим. Але це рішення передбачає і певні проблеми у використанні, такі як незбалансований дизайн, не завжди зручний інтерфейс, невідповідні інформаційні моделі, неврахування використання додаткових периферійних пристроїв (мишки, наприклад), невисокий відсоток охоплення різних пристроїв. Мобільна респонсивна версія буде виглядати незбалансованою і користувачеві буде незручно її використовувати. Відповідно, респонсивний дизайн підійде тільки для тих додатків, які не передбачають наявності мобільних версій, а для систем розумного дому це сьогодні не підходить. Але респонсивний дизайн має елементи, які можна запозичити – відомі та звичні елементи управління. А це означає, що ваш респонсивний додаток не зможе охопити мобільну частину користувачів, які потенційно могли б стати вашими клієнтами [16].

З урахуванням того, що частка користувачів, що віддають перевагу мобільні пристрої, продовжує зростати, цей недолік може стати вирішальним аргументом на користь адаптивного дизайну. Статистика свідчить про те, що на сьогоднішній день близько 50% відвідувачів використовують мобільні пристрої [18].

Прикладом використання адаптивного дизайну є розумний дисплей Home Hub компанії Google. Це спеціальний смарт-дисплей для управління "розумним будинком". З його допомогу можна передавати команди системі, в тому числі - за допомогою голосових команд. Дисплей взаємодіє з окремими компонентами "розумного будинку". Також розробку Google можна використовувати як домашній органайзер. У неї вбудовані функції перегляду календаря і створення нагадувань. Крім того, Home Hub легко взаємодіє зі Всесвітньою павутиною [19-20].

Вибираючи адаптивний дизайн для свого застосування необхідно врахувати властивості всіх приладів, сумісність, можливості адаптації під звички користувачів. Відомі підходи дизайну для інтерфейсів веб та мобільних додатків розумного дому повинні бути узгоджені с вибраними моделями та методами створення інтерфейсів.

1.3. Висновки

Таким чином, можна зробити висновок, що технології розумного дому передбачають використання різноманітних підходів до управління приладами – від спеціальних сенсорних панелей до пультів управління на спеціальних пристроях, смартфонах та комп’ютерах. Для формування клієнтоорієнтованого адаптивного дизайну доцільно виконати моделювання та удосконалити метод дизайну з можливістю вибору інтерфейсу користувача.

Адаптивний клієнтоорієнтований інтерфейс користувача може бути побудований на основі моделей та методів, що враховують методи дизайну, навички споживачів, групування функціональних панелей за напрямами.

РОЗДІЛ 2 МОДЕЛІ ІНТЕРФЕЙСІВ СИСТЕМИ УПРАВЛІННЯ РОЗУМНИМ ДОМОМ

2.1. Модель клієнтоорієнтованого адаптивного інтерфейсу користувача

«Під адаптивним інтерфейсом користувача розуміють такий інтерфейс, що не тільки забезпечує взаємодію між користувачем і додатком, а також реалізує модель уподобань користувача, на основі якої зовнішній вигляд інтерфейсу і спосіб роботи з додатком підлаштовується під потреби користувача» [15]. Такий підхід забезпечує адаптацію під навички та особливості поведінки користувача додатку. Деякі дослідники чітко розрізняють поняття адаптивного веб-дизайну та адаптивного інтерфейсу і уточнюють, що «адаптивні інтерфейси дозволяють персоналізувати контент на сторінці, відображати візуальні компоненти у зручному для користувача вигляді, виводити на перший план ті частини функціоналу, якими людина користується найчастіше, скривати непотрібні користувачу елементи на сторінці, надати зручний спосіб взаємодії для людей з обмеженими можливостями».

На нашу думку, основою моделі клієнтоорієнтованого адаптивного інтерфейсу користувача може бути модель IBM [3]. Ця модель має адаптивне подання і навігацію. Удосконалення запропонованої моделі буде формуватись відповідно до таких напрямів:

1. Врахування особливостей користувачів відносно їх комп’ютерних навичок, знань предметної області, персональних даних, попередньою історією роботи з пристроями, профілю користувача тощо [15].
2. Запровадження вибраної системи дизайну з врахуванням складності системи та алгоритмів формування інформаційних моделей.
3. Вибір моделей візуалізації та рекомендацій щодо роботи пристроїв.

Удосконалена модель для розумного дому представлена на рис. 2.1. її спрощену модель зображено на рис. 2.1.

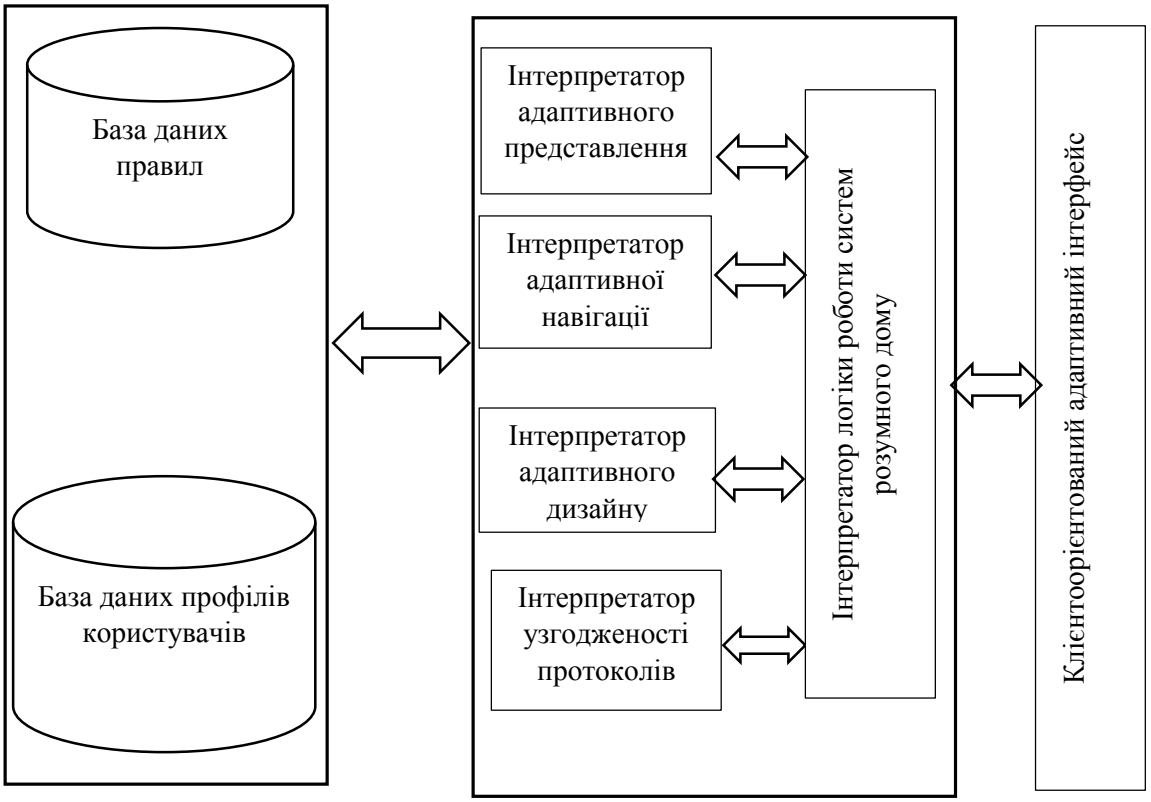


Рисунок 2.1. – Удосконалена модель клієнтоорієнтованого адаптивного інтерфейсу користувача системи розумного дому

Запропонована модель дозволить адаптувати інтерфейс за допомогою інформації даних профілю користувача, правил роботи пристроїв та побудови адаптивного дизайну. Врахування та інтерпретації адаптивного представлення, навігації. Дизайну, узгодженості протоколів дозволить врахувати поведінку користувача, поліпшити взаємодію між людиною і додатком, визначити закономірності між технічними даними дизайну, адаптації, навичками користувача та конфігурацією інтерфейсу.

2.2. Модель взаємодії приладів, програмних агентів та користувача

Оптимістичний сценарій моделі адаптивного інтерфейсу користувача дозволяє модифікувати представлену модель за модулями штучного інтелекту з подальшим машинним навчанням. Такий підхід дозволить знайти відповідності між окремими користувачами та найбільш вдалою конфігурацією інтерфейсу системи управління. Крім того, користувач має можливість налаштування конфігурації інтерфейсу з використанням типів компонентів та можливостей формування респонсивного або адаптивного дизайну. Інтерактивний профіль допомоги дозволяє зберігати в профілі користувача сценарії формування конфігурації інтерфейсу.

Така модель більш детально формується на основі модулів:

зберігання інформації про користувачів;

інтерпретатор адаптації;

рекомендаційна система;

вибору веб-дизайну;

набору компонентів;

CSS стилів;

агрегації;

узгодження протоколів;

допомоги роботи з системою.

Модель представлена на рис. 2.2. Розглянемо особливості кожного модулю. Модуль зберігання інформації про користувачів є базою даних, що містить інформацію профілю. Інформацію контексту, інформацію поведінки користувача, відповідну конфігурацію інтерфейсу відповідно до історію роботи користувача. Історія дій користувача включає в себе в себе інформацію про частоту використання, час перебування та різноманітні інтерактивні взаємодії користувача з приладами та програмними агентами системи розумного дому.

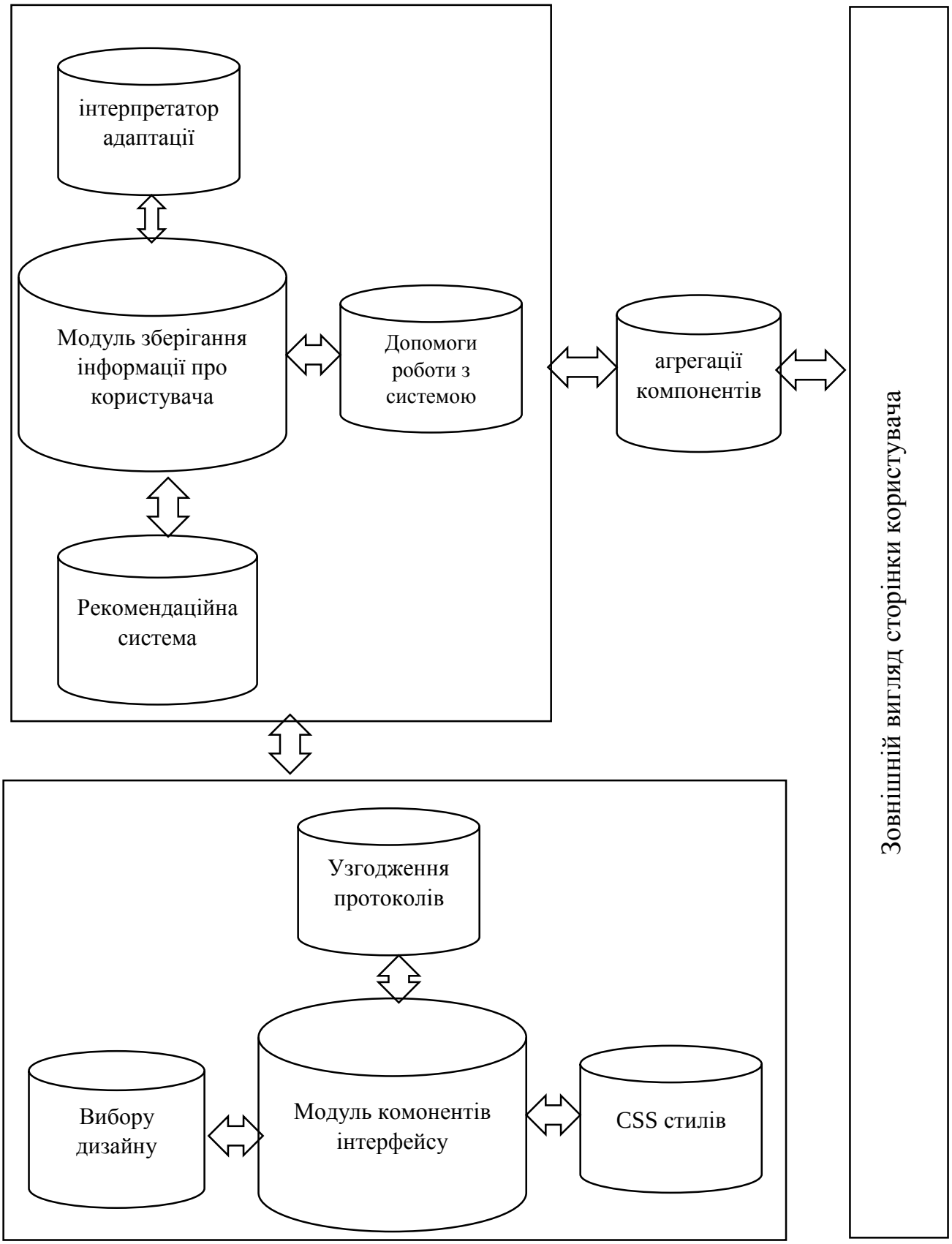


Рис. 2.2. Модельна деталізована модель адаптивного інтерфейсу користувача розумного дому

Дані про конфігурацію інтерфейсу включають в себе обраний безпосередньо користувачем список компонентів інтерфейсу, варіант їх стилізації та налаштувань у відповідності до системи дизайну. Запропонована модель враховує технічні та поведінкові дані для адаптації і дозволяє сформувати клієнторієнтований адаптивний інтерфейс як для веб-додатку, так і для мобільного додатку, або з використанням спеціалізованих пристроїв. Кожен модуль має свої функції і працює відповідно до визначених правил. У деяких випадках можна вибрати варіант, коли респонсивний дизайн розробляється тільки для десктопного зображення, а окрема мобільна версія створюється на піддомені сайту. Але це має на увазі окрему розробку двох сайтів, що теж створює певні незручності. У цього типу теж є свої певні недоліки. Якщо ви плануєте створити адаптивний додаток, враховуйте, що вам доведеться зіткнутися з високою вартістю розробки.

Розглянемо основні модулі моделі. Модуль інтерпретатора адаптації в системі розумного дому допомагає визначити набір компонентів інтерфейсу, їх налаштування, стилізацію, вибір дизайну. В цьому модулі доцільно використовувати машинне навчання, а також сформувати панелі налаштувань та вибору для досвідчених користувачів. . В режимі навчання повинно відбуватись знаходження закономірностей між інформацією про користувача та найбільш зручною для нього конфігурацією інтерфейсу. Система пропонує користувачу найбільш зручну конфігурацію. При недостатній базі кількості збережених конфігурацій, система пропонує стандартні конфігурації на вибір, що зменшить рівень адаптації.

Рекомендаційна система дозволяє визначити прилади, що найбільше використовуються, фільтрує найбільш пріоритетні програмні додатки та прилади використання, а також рекомендації щодо конфігурацій користувача.

Модулі з набором компонентів та узгодження протоколів дозволяють сформувати найбільш адаптивну інформаційну панель або/і сформувати блоки на цій панелі з параметрами узгодження взаємодії.

Модулі вибору дизайну та стилів дозволяють визначити загальну стилізацію проекту та адаптацію під пульти та гаджети.

Модуль агрегації дозволяє звести всю інформацію в цілісну релевантну сторінку, набір панелей, сформувати головну панель.

Модуль допомоги може бути сформований традиційно, а може бути інтерактивним помічника, поведінка якого формується за допомогою рекомендаційної системи на основі інформації щодо попередньої активності користувача, контексту та поведінкової інформації та реалізується через своєчасні персоналізовані підказки.

На рис. 2.3. представлено модель взаємодії приладів.

Рисунок 2.3 – Модель взаємодії приладів у системі

Дана модель може бути реалізована за допомогою різних технологічних рішень та архітектур програмного забезпечення з використанням мікросервісної архітектури.

Моделі UML представляють принципи взаємодії користувача з системою і представлені на рис 2.4.

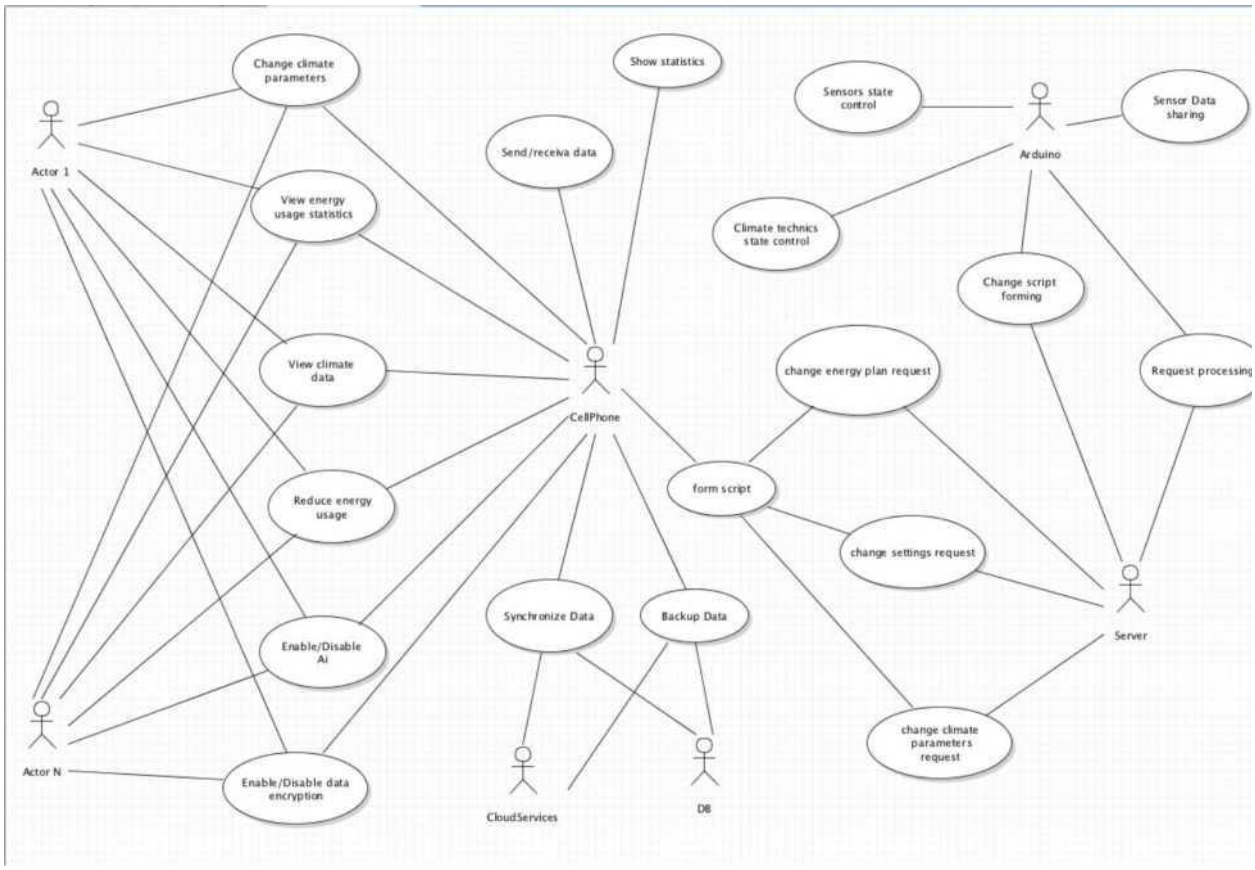


Рисунок 2.4 – Use-case діаграма системи Smart Home

Така діаграма відображає у шлях від запиту користувача до отримання результату у вигляді відгуку системи. Модель взаємодії приладів системи Smart Home може бути побудована на основі архітектури з центральним пристроєм-контролером / сервером. Запити з пульту, веб та мобільного додатку надходять на сервер і формують чергу обробки, після цього відправляються на мікроконтролери з можливістю синхронізації даних з

хмарними сервісам. Відновлення системи здійснюється за збереженими хмарними даними.

Онлайн сховище зберігає дані серверів і використовується для підвищення надійності і можливості відновлення системи після технічних збоїв. Таке сховище зберігає профілі роботи системи як файловий архів з можливістю подальшого відновлення.

Так, наприклад, для модулю регулювання мікроклімату Всі процеси взаємодії представлені на діаграмі як дії користувача, програмних агентів та приладів. Система розумного дому не повинна бути перезавантажено для того, щоб кількість запитів не була занадто великою. Наприклад, коли багато користувачів одночасно будуть змінювати кліматичні параметри та змінити конфігурацію інтерфейсів.

2.3. Методи аналізу дизайну інтерфейсу та формування клієнтоорієнтованого адаптивного інтерфейсу

Метод аналізу дизайну інтерфейсу базується на результатах дослідження та опрацювання методу адаптивного дизайну та методу оцінювання поведінки користувача.

Адаптивний дизайн базується на розмірах сітки для визначення концепцій дизайну відносно системи та приладів. Блоки управління містять каруселі повідомлень системи допомоги, кнопки управління. Саме тому, необхідно визначити розміри кнопок, повідомлень, параметри зменшення та збільшення варіантів інтерфейсу. Розміри для мобільних пристроїв та інших гаджетів враховуються відносно бібліотеки компонентів.

Розглянемо основні властивості адаптованості інтерфейсу. Адаптованість означає, що інтерфейс повинен бути:

сумісним з потребами та можливостями користувача;

забезпечувати простоту переходу від виконання однієї функції до іншої;

забезпечувати користувача на високому рівні методичними рекомендаціями стосовно його можливих дій, а також генерувати належний зворотний зв’язок на його запити;

надавати користувачу можливість відчувати себе повноправним керівником ситуації;

забезпечувати користувача різними, взаємно доповнюючими формами представлення результатів в залежності від типу запиту або від характеру отриманого рішення;

враховувати особливості користувачів різних рівнів знань та навичок.

Інтерфейс повинен бути достатнім з точки зору сценаріїв поведінки користувача, його однозначних дій та відклику на них зі сторони управління.

Дружність інтерфейсу полягає у максимальній простоті та готовності задовольнити реалізацію сценаріїв поведінки.

Гнучкість інтерфейсу полягає у конкретизації та адаптування до кожного завдання. Для розумного дому це здійснюється за допомогою різних інформаційних панелей.

Метод адаптивного дизайну для реалізації інтерфейсу управління розумним домом повинен передбачати забезпечення таких можливостей:

- повідомлення щодо ситуацій управління;
- зворотній зв'язок на дії користувача;
- зберігання бібліотеки профілів поведінки користувача;
- формування інформаційних панелей і повідомлення переходу від одної до іншої.;
- повідомлення щодо неузгодженості або несумісності протоколів та неможливості подальшої роботи.

Метод формування клієнтоорієнтованого адаптивного інтерфейсу формується на основі таких принципів.

- відповідності призначення і структури інтерфейсу реалізується у відповідності визначених задач модулів системи розумного дому.

- мінімізації витрат ресурсів користувача забезпечується формуванням спеціальних інформаційних панелей відповідно до підсистем розумного дому. Вибір дизайну дозволяє зменшити кількість інформації для запам’ятовування.
- незбитковості забезпечується збереженням інформації в профілі користувача та системою допомоги користувачу.
- безпосереднього доступу до системи підказок полягає у формуванні спеціальних повідомлень на дії користувача та повідомлення про стан системи.
- гнучкості забезпечується за двома напрямами: - 1 – адаптивність під різні протоколи, гаджети; 2 – адаптивність під поведінку користувача та вибрані модулі системи
- максимальної концентрації користувача на задачі розв’язується за допомогою локалізації дій користувача та надання повідомлень тільки за визначеним колом питань.
- врахування навичок конкретного користувача базується на моделі користувача, проектується «людський фактор», який тісно вплітається в особливості функціонування всієї системи за допомогою інформації, що зберігається в профілі користувача та еталонних профілів в системі.
- легкості користування і простоти навчання полягає в системі малих кроків та простоті виправлення помилок.
- надійності забезпечується модулями захисту, шифрування, рівнів доступу до інформації.

Врахування людського фактору базується на використання різноманітної інформації, внесеної користувачем, зібраної для еталонних сценаріїв, а також аналітики досвіду використання системи.

Клієнтоорієнтована адаптація – це метод проектування, дослідження та пояснювання інтерфейсу з врахуванням поведінки клієнта (користувача). Всіх користувачів комп’ютерних систем можна розділити на дві основні групи: фахівці (постійно користуються комп’ютером) та тимчасові користувачі (користуються комп’ютером час від часу - керівники різних рівнів,

бухгалтери, користувачі-початківці). Аналогічно користувачі розумного дому можуть бути поділені на новачків, досвідчених користувачів, адміністраторів.

Загальні принципи врахування людського фактору:

- користувач повинен завжди знати, що робити далі;
- система повинна давати йому інструкції стосовно того, як продовжити роботу, створити резервний файл результатів, вийти із системи тощо.
- користувач повинен знати, що система чекає від нього. І це може бути реалізовано у вигляді стислих повідомлень: ГОТОВО (READY), ВВЕДІТЬ КОМАНДУ (ENTER COMMAND), ВВЕДІТЬ ВИБІР (ENTER CHOICE) або ВВЕДІТЬ ДАНІ (ENTER DATA) .
- Користувач повинен знати, що дані введено коректно. ДАНІ КОРЕКТНІ або ВВЕДЕННЯ ЗАВЕРШЕНО (INPUT OK) .
- Користувач повинен знати, що дані не були введені коректно. ПОМИЛКА (ERROR) ; (ДОПОМОГА (HELP).

Важливим також э форматування екрану. Воно повинно містити титул, вікно з прапорцями і вказівниками, повідомлень, виходу., операційне вікно. Приклад розподілу на зони представлено на рис. 2.5.

-

- ЗАГОЛОВОК	
- КОМАНДИ ТА ПОВІДОМЛЕННЯ	
- ОСНОВНЕ ПОЛЕ ПРЕДСТАВЛЕННЯ РЕЗУЛЬТАТІВ	- П - Р - А - П - О - Р - Ц - І
- ВИХІД	

Рис. 2.5. Приклад розподілу екрану термінала на зони

Інформація та повідомлення щодо результатів повинні утримуватись на екрані такий час, щоб користувач міг прочитати сприйняти їх.

Для економії місця на екрані адаптивність інтерфейсу забезпечується не тільки зміною інформаційних панелей, а і тимчасового анулювання деяких зон. Система допомоги і рекомендацій повинна створювати спеціальні повідомлення для користувача, наприклад – ТЕМПЕРАТУРА? ВОЛОГІСТЬ?– для підказки показників клімат контролю. Або ви на панелі модулю клімат контролю.

Метод формування клієнтоорієнтованого адаптивного дизайну та відповідного інтерфейсу може бути реалізований за допомогою сукупності таких дій:

1. Визначення інформаційної панелі управління;
2. Визначення рівня знань користувача;
3. Пропозиція щодо використання виду дизайну еталонних профілів;
4. Пропозиція використання поведінкового профілю користувача;
5. Перевірка узгодженості протоколів пристроїв.
6. Формування сторінки інтерфейсу.

2.4. Висновки

Таким чином, були сформовані удосконалені моделі клієнтоорієнтованого адаптивного інтерфейсу для системи розумного дому, визначено метод формування клієнтоорієнтованого адаптивного дизайну та відповідного інтерфейсу. Одержала подальший розвиток модель адаптивного інтерфейсу користувача, яка, на відміну від існуючої, відповідає вимогам користувача, які записані в його профілі та змінюється відносно визначених модулів, підключених пристроїв та за вибором користувача.

Запропонований метод аналізу дизайну інтерфейсу базується на формуванні правил для створення інформаційної моделі, перевірки сумісності приладів та сценаріїв робот відповідно до вибраних профілів користувача.

РОЗДІЛ 3 СИСТЕМА ПРОЕКТУВАННЯ ВЗАЄМОДІЇ ПРИЛАДІВ SMART HOME

3.1. Підходи до проектування та програмне забезпечення взаємодії мікроконтролера та додатків управління

Розвиток та впровадження систем розумного дому [1-6] здійснюється за різними підходами. Один з них – використання готових рішень. Другий – блочно-ієрархічний підхід [21]. Він передбачає розбиття процесу проектування на ряд ієрархічних рівнів. Загальний випадок складається з трьох рівнів:

системного,

схемотехнічного;

компонентного.

Математична модель такого підходу для системи управління та інтерфейсу може бути представлена як [3-4]:

$$\text{Srd} = \bigcup_{i=1}^{n} P_{rd}^{2,i} \bigcup_{j=1}^{m} B_{rd}^{3,j} = \bigcup_{k=1}^{l} E_{rd}^{4,k} \quad (3.1).$$

Перший рівень система управління розумним домом, загальна панель.

Другий рівень – система управління модулями (клімат, освітлення, безпека тощо).

Третій – датчики, взаємодія, компоненти.

В процесі проектування використовують підходи зверху вниз і знизу до верху.

Підхід зверху вниз передбачає розробку проекту всієї системи розумного дому з використанням загального інтерфейсу системи та інформаційних панелей для кожного модулю.

Підхід знизу до верху передбачає розробку окремих модулів та їх інтеграцію в одну систему.

Варіанти проектування представлені на рис. 3.1.

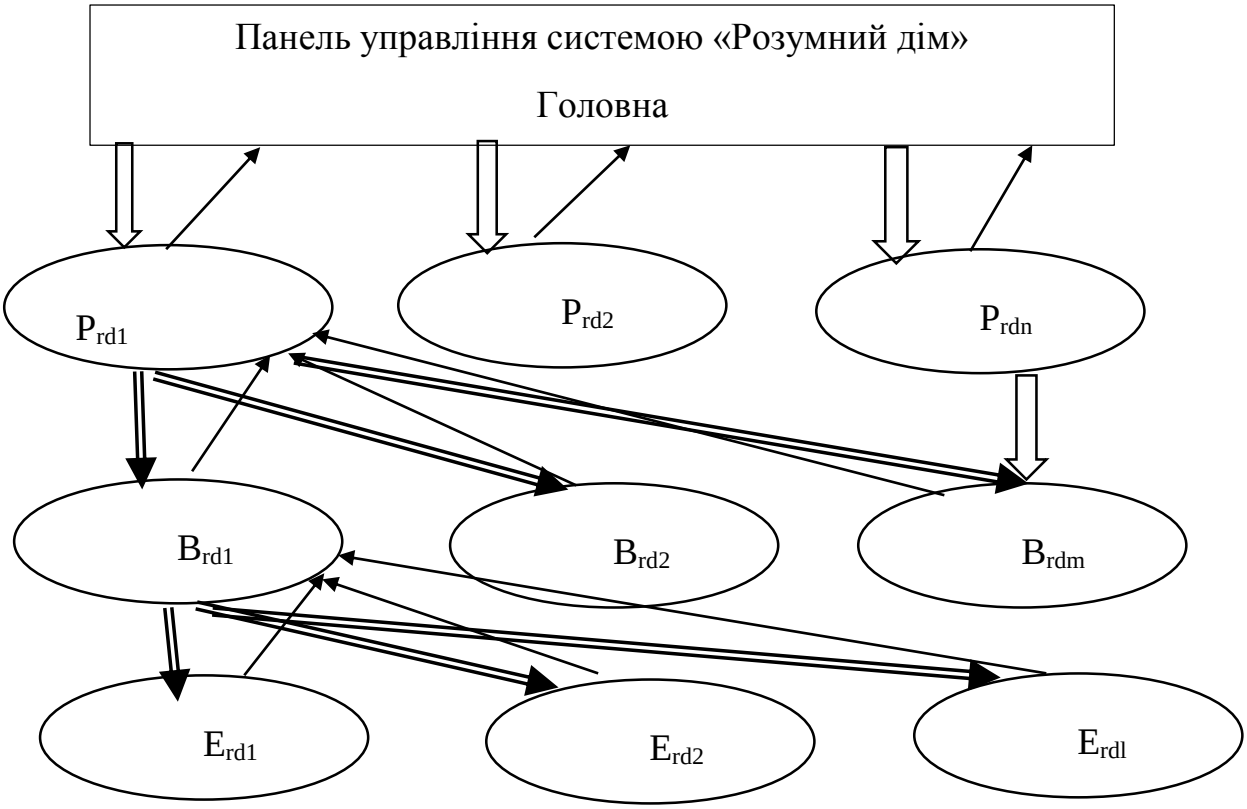


Рисунок 3.1. Змішана система проектування та формування даних для інформаційних панелей управління розумнім домом

В таблиці 3.1. представлено особливості кожного рівня.

Таблиця 3.1. - Основні задачі інтелектуального аспекту в процесі розроблення РД при використанні блочно-ієрархічного підходу

Назва рівня	Задачі
Системний рівень	Вибір рівня інтелектуалізації системи ІБ. Обґрунтування можливості реалізації інтелектуальних функцій. Вибір стратегії реалізації інтелектуальних функцій ІБ. Вибір методів та засобів реалізації інтелектуальних функцій ІБ. Побудова структурних схем реалізації інтелектуалізації функцій ІБ.
Схемотехнічний рівень	Задачі інтелектуального опрацювання даних в підсистемах ІБ. Вибір необхідних методів, засобів, моделей та методологій. Реалізація інтелектуальних функцій ІБ на рівні підсистем.
Компонентний рівень	Задачі інтелектуального аспекту в процесі проектування систем ІБ. Задачі інтелектуального опрацювання даних від давачів (відновлення втрачених даних тощо). Вибір методів, моделей та алгоритмів опрацювання даних від давачів.

На практиці використовують змішану систему нарощування модулів з готовою гнучкою панеллю управління з можливістю додавання функцій та інформаційних моделей. Така схема застосування сформована на основі двох підходів і може бути використана для формування системи розумного дому в середовищі симулятора Arduino та формування веб-додатку управління модулем клімат-контролю.

Системи розумного дому працюють за допомогою роботи мікроконтролерів, які виконують такі функції як слідкування за значеннями сенсорів; ініціація включення та виключення пристроїв, взаємодії серверу через протокол; зчитування файлів конфігурацій з карти пам’яті; формування алгоритмів за сценаріями дій тощо.

Програмне забезпечення Arduino IDE має такі переваги як невибагливість до ресурсів комп’ютера або іншого гаджета, незатребуваність великого обсягу пам’яті, можливості роботи з текстовим редактором та виведення тексту в контроль, роботи з панелями інструментів: займає мало пам’яті. Arduino IDE містить вбудовані приклади скетчів, що дають змогу працювати з різними модулями. Спеціальне середовище симулятора дозволяє моделювати роботу датчиків без підключення плати. Основний функціонал симулятора Tinkercad для розробника Arduino може бути представлений таким чином [24]:

онлайн платформа, для роботи не потрібно нічого крім браузера і стійкого інтернету.

зручний графічний редактор для візуального побудови електронних схем.

набір попередньо встановлених моделей більшості популярних електронних компонентів, відсортоване за типами компонентів.

симулятор електронних схем, за допомогою якого можна підключити створене віртуальне пристрій до віртуального джерела живлення і простежити, як воно буде працювати.

симулятор датчиків та інструментів зовнішнього впливу. Ви можете

змінювати показання датчиків, стежачи за тим, як на них реагує система.

вбудований редактор Arduino з монітором порту і можливістю покрокової налагодження.

готові для розгортання проекти Arduino зі схемами і кодом.

візуальний редактор коду Arduino.

можливість інтеграції з рештою функціональністю Tinkercad і швидкого створення для вашого пристрою корпусу та інших конструктивних елементів – сформовані модель може бути відразу ж скинута на 3D-принтер.

Середовище розробки від Arduino надає можливість користування такими бібліотеками як:

SPI.h

SD.h

Ethemet.h

DHT.h

stdio.h

string.h

SFE_BMP180.h

Wire.h

OneWire.h

DallasTemperature.h

Визначені бібліотеки використовуються у таких відповідностях:

Бібліотека SPI.h необхідна для роботи з SD Card Reader та W5100 Ethernet Shield.

Бібліотека SD.h відповідає за роботу з SD Card Reader [21].

Бібліотека Ethernet.h використовується для взаємодії з W5100 Ethernet Shield [24].

Бібліотека DHT.h необхідна для зв’язку з датчиками сімейства DHT.

Бібліотеки stdio.h та string.h використовуються для роботи зі строками та стандартними функціями.

Бібліотека SFE_BMP180.h відповідає за взаємодію з датчиком

барометричного тиску BMP180.

Бібліотеки OneWire.h та DallasTemperature.h необхідні для роботи з датчиками DS1820/DS18B20/DS18S20.

Передача даних забезпечується протоколом TELNET, який працює для контролю та дозволу термінальним пристроям і процесам працювати один з одним. Для формування роботи використовують чотири основні функції:що дозволить легко тестувати роботу системи без необхідності використовувати додаткове ПЗ.

run_config()

rewrite_etalon()

setup()

loop()

Функція run_config представлена на рис. 3.2. і дозволяє здійснювати виконує зчитування файлу конфігурацій з карти пам’яті та виконувати первинне налаштування мікроконтролера для роботи.

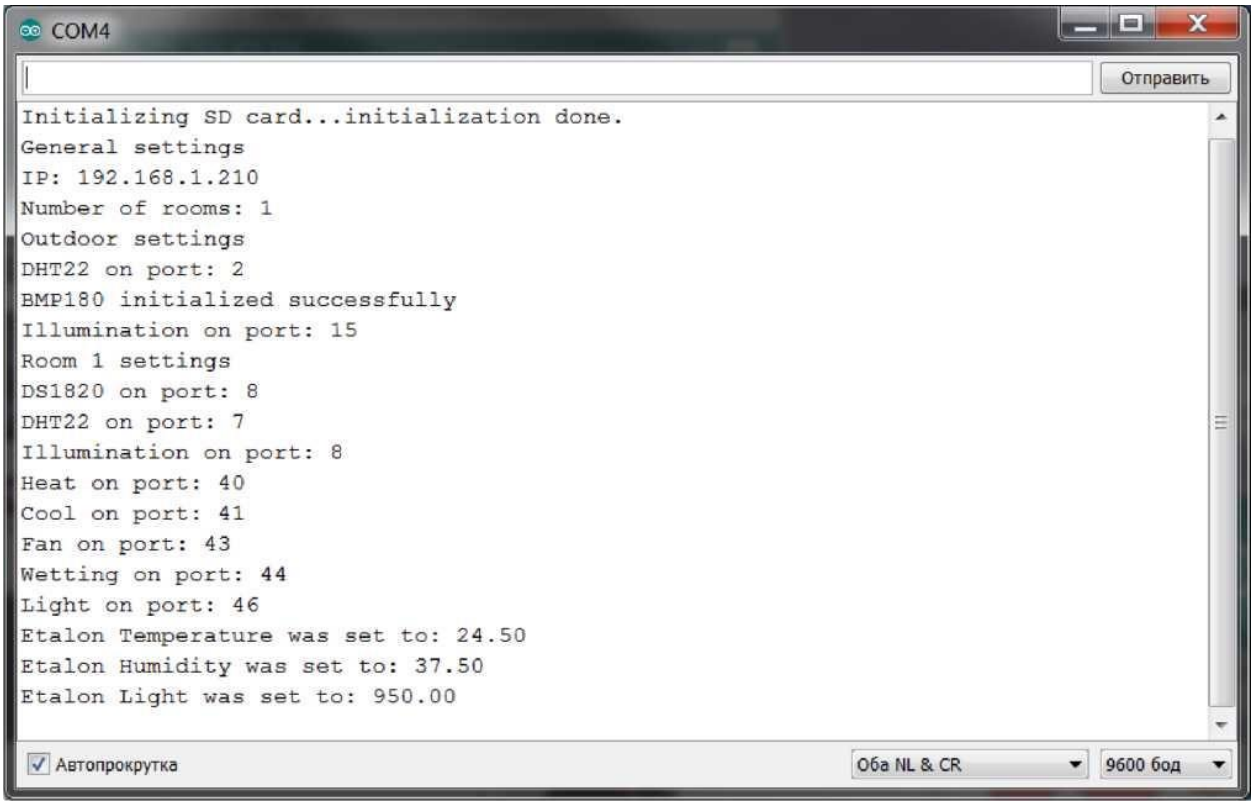


Рисунок 3.2. Ініціалізація роботи

При виконанні функції здійснюється лексичний аналіз файлу

конфігурацій, створюються та заповнюються відповідні масиви, які зберігають інформацію щодо взаємодії з сенсорами, роботи сценаріїв, керування навантаженням. Ця функція також відповідає за ініціалізації датчиків і зчитування еталонних параметрів. Виведення результатів може бути реалізовано на екран десктопного пристрою або гаджету. Приклад наведено на рис. 3.2.

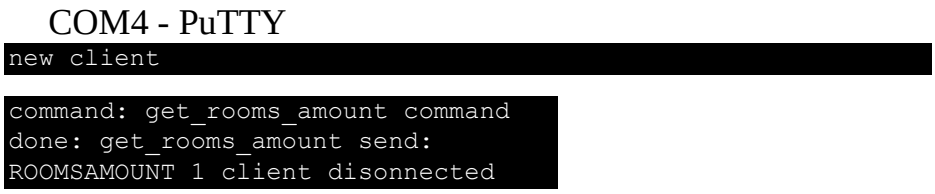


Рисунок 3.3. – Робота команд get_rooms_amount

Функція rewrite_etalon відповідає за видалення файлів та перезапис еталонних функцій.

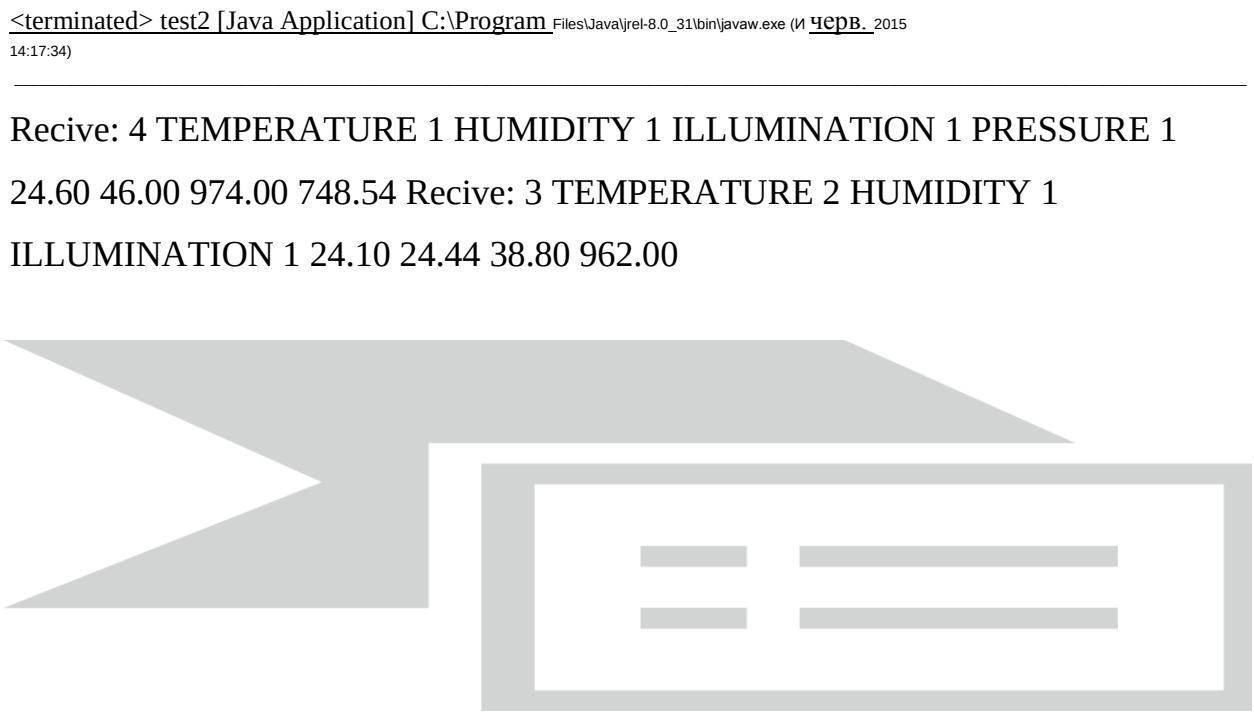


Рисунок 3.4 - Робота команд sensor_request PUTTY

Функція setup виконує старт мікроконтролера та первинні ініціалізації компонентів та запуск функції run_config.

Функція loop використовується для оновлення показників датчиків включення або виключення пристроїв, обробку запитів, формування повідомлень.

Розглянемо основні команди logout; get_rooms_amount; sensor_request; set.

Приклад роботи команд get_rooms_amount наведено на рис. 3.3

Лексичний аналіз усіх команд та файлу конфігурацій відбувається за допомогою використання команд розбиття строки на лексеми strtok(). Далі за допомогою порівнянь визначаються основні команди або ключові слова та параметри.

Усі команди, які отримує мікроконтролер, перевіряються на коректність, якщо були знайдені помилки, то мікроконтролер надішле відповідне повідомлення.

Змішана система проектування передбачає побудову загальної структури проектування та інформаційної моделі розумного будинку (Рис 3.1.), проектування та реалізацію одного модулю з подальшим збором інформації та додавання нових компонентів та модулів.

3.2. Система клімат-контролю

Система клімат-контролю може бути представлена як загальна структура, що описується графом (Див. Рис. 3.5.).

Структура є основою для описання роботи системи за допомогою графу. Якщо структура має датчик темпертури та датчик вологості, то контролер повинен спрацювати для регулювання параметрів таких приладів як кондиціонер, обігрівач, витяжка та зволожувач повітря. На рис. 3.6. представалено граф системи. Такий граф є орієнтованим і кожна вершина може містити потрібні маркування для подальшого використання Відповідно спираючись на структуру, яка описана об’єктами із посиланнями на вхідні та

вихідні дії/умови (переходи), необхідно використати швидке та ефективне перетворення даного запису в орієнтований граф [].

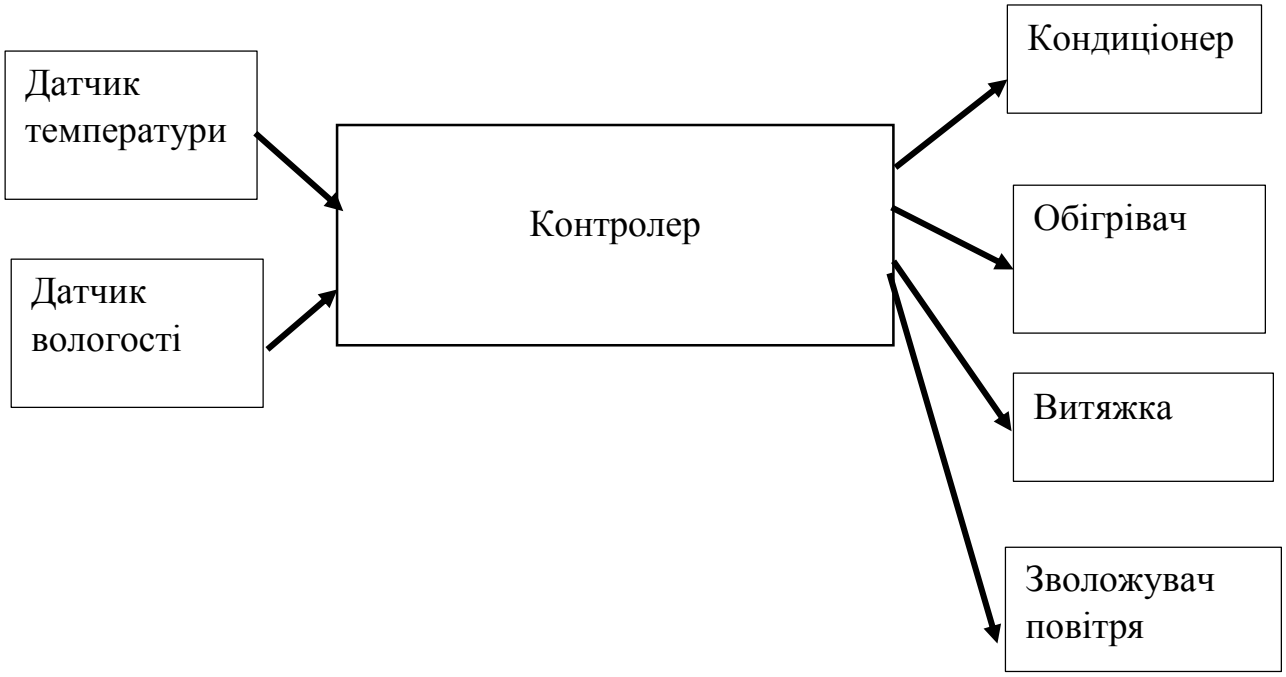


Рисунок 3. 5 Система регулювання мікроклімату

Система працює між двома (другим і третім рівнем) ієрархії за командами контролеру.

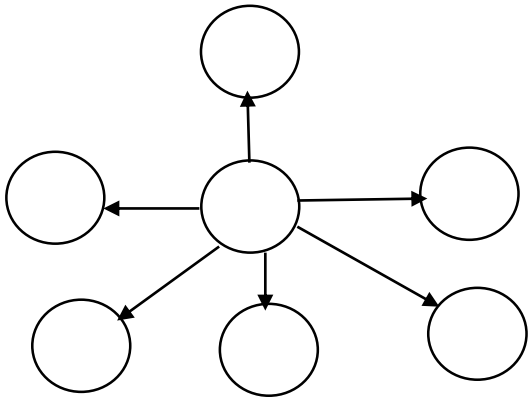


Рисунок 3.6. Абстрактний граф структури підсистеми клімат контролю однієї підкімнат.

Кожна вершина може мати необхідні маркування для розуміння процесів та ціни цих процесів. Для того, щоб таку структуру перетворити в граф необхідно послідовно проініціалізувати списки вершин та дуг; вибрати

перший об’єкт; визначити всі переходи; виконати потрібний перехід, додати дугу і маркування; виконати перевірку нових переходів, створити нові додаткові вершину-перехід, якщо перехід на перевірці відсутній. Повторити цей шлях для кожного об'єкту та сформувати структуру графу. Такий підхід дозволить перевірити всі потрібні зв'язки.

Формування інтерфейсу для веб-додатку здійснюється за сценарієм вибору від 4 до 6 параметрів та мінімалістичного дизайну. Програмна реалізація представлена на рис. 3.5.

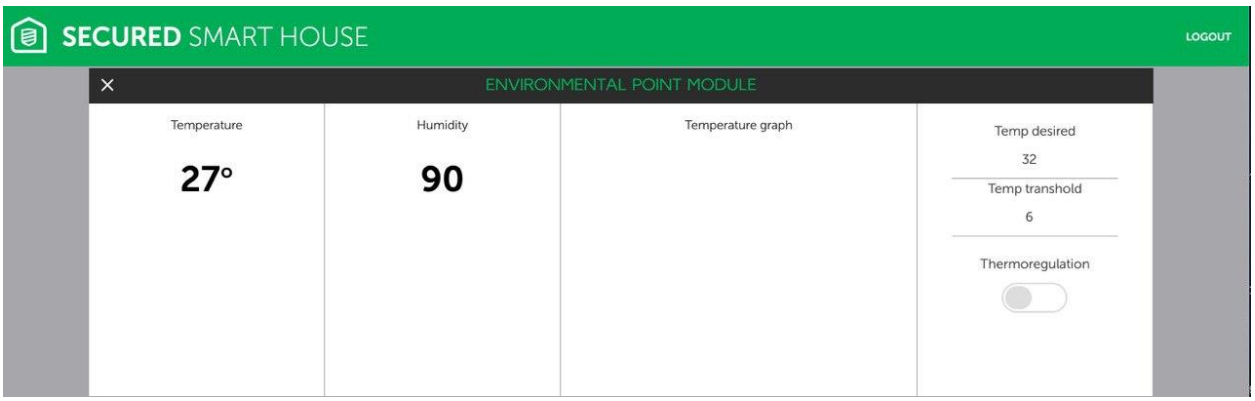


Рисунок 3.7. Головна сторінка підсистеми клімат-контролю системи «Розумний дім»

Лістінг коду представлено в додатку.

3.3 Висновки

Загальні підходи до побудови архітектури розумного дому можуть бути застосовані в процесах проектування та для побудови інформаційних панелей. В архітектурі системи Smart Home центральний контролер займає основне місце, так як є головним засобом комунікації між додатком й побутовою електронікою. Він виконує кілька основних функцій, таких як: зберігання налаштувань та інформації, зв'язок між приладами, система шифрування, зв'язок з хмарним сховищем, прогнозування енергоспоживання, тощо.

РОЗДІЛ 4 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ УПРАВЛІННЯ СИСТЕМОЮ КЛІМАТ-КОНТРОЛЮ

4.1. Особливості проектування мобільних додатків

Популярність використання мобільних пристроїв для безпосереднього та віддаленого управління зростає з кожним днем. Для систем розумного дому такі мобільні додатку можуть стати першою підсистемою з елементами штучного інтелекту, використанням асистентів, або роботи з інтерфейсом та пультом управління. Усі найновіші технології відразу ж стають доступні на мобільних пристроях. Багато нових приладів для розумного дому вже мають програмне забезпечення для запуску на мобільному додатку. Мобільні додатки тісно інтегруються з операційною системою – можуть передавати і отримувати через неї дані з інших програм, обробляти запити операційної системи і приймати сигнали про події, такі як зміна орієнтації пристрою, підключення або відключення бездротової мережі. Звернення до операційної системи виробляються через так званий програмний інтерфейс API (Application Programming Interface). Код, який реалізує API-функції, міститься у пам'яті тільки в одному екземплярі, що робить додатки значно компактнішими та зменшує кількість споживаних ресурсів персонального мобільного пристрою. Найпопулярнішою на сьогоднішній день платформою для якої розробляють мобільні додатки є операційна система Android, яка базується на ядрі Linux. На даний час її підтримкою та розробкою займається консорціум Open Handset Alliance, ініційований компанією Google після покупки компанії Android Inc. Операційна система Android забезпечує простоту та зручність використання і налаштування системи, захист даних від зараження вірусами завдяки - ізольованій роботі кожного додатку, високу функціональність в користуванні Інтернетом, зручну роботу з електронною поштою, підтримку додатків Adobe Flash, широкий спектр можливостей підключення - Wi-Fi, Bluetooth, GPRS, EDGE, 3G та багато іншого. Android

додатки пишуться на мові програмування Java та виконуються на віртуальній машині - Dalvik Virtual Machine [25-28].

Середовище Android Studio дозволяє здійснювати програмування та моделювання роботи мобільного додатку за зручними функціями. Це інтегроване середовище розробки, що запроваджено в 2013 році компанією

Середовище надає засоби для розробки застосунків не тільки для смартфонів і планшетів, але і для носимих пристроїв на базі Android Wear, телевізорів (Android TV), окулярів Google Glass і автомобільних інформаційно-розважальних систем (Android Auto). Для застосунків, спочатку розроблених з використанням Eclipse і ADT Plugin, підготовлений інструмент для автоматичного імпорту існуючого проекту в Android Studio.

Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android[4] . У тому числі у середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Крім можливостей, присутніх в IntelliJ IDEA, в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції.

Для прискорення розробки застосунків представлена колекція типових елементів інтерфейсу і візуальний редактор для їхнього компонування, що надає зручний попередній перегляд різних станів інтерфейсу застосунку (наприклад, можна подивитися як інтерфейс буде виглядати для різних версій Android і для різних розмірів екрану). Для створення нестандартних інтерфейсів присутній майстер створення власних елементів оформлення, що підтримує використання шаблонів. У середовище вбудовані функції завантаження типових прикладів коду з GitHub.

До складу також включені пристосовані під особливості платформи Android розширені інструменти рефакторингу, перевірки сумісності з минулими випусками, виявлення проблем з продуктивністю, моніторингу споживання пам'яті та оцінки зручності використання. У редактор доданий режим швидкого внесення правок. Система підсвічування, статичного аналізу та виявлення помилок розширена підтримкою Android API. Інтегрована підтримка оптимізатора коду ProGuard. Вбудовані засоби генерації цифрових підписів. Надано інтерфейс для управління перекладами на інші мови.

Деякі особливості будуть пізніше розгорнуті для користувачів так як програмне забезпечення розвивається; наразі, передбачені такі функції [24] :

- Живі макети (layout): редагувальник WYSIWYG — живе кодування — подання (rendering) програми в реальному часі.[25]
- Консоль розробника: підказки по оптимізації, допомога по перекладу, стеження за напрямком, агітації та акції — метрики Google аналітики.
- Резерви бета релізів та покрокові релізи.
- Базування на Gradle.
- Android-орієнтований рефакторинг та швидкі виправлення.
- Lint утиліти для охоплення продуктивності, юзабіліті, сумісності версій та інших проблем.
- Використання можливостей ProGuard та підписів до програм.
- Шаблони для створення поширених Android дизайнів та компонентів.
- Різноманітний редактор макетів (layouts) що дозволяє користувачам перетягнути і покласти (drag-and-drop) компоненти користувацького інтерфейсу, як варіант, переглянути одночасно макети (layouts) на різних конфігураціях екранів.

Згідно з індексом TIOBE, Java є найпопулярнішою мовою програмування. Але крім цієї мови, використовують також Kotlin, який був спроектований і розроблений компанією JetBrains, чеською компанією, яка

відома своєю популярною IDE - IntelliJ IDEA. Нещодавно команда Android від Google оголосила, що офіційно додає підтримку мови програмування Kotlin.

Kotlin був розроблений для розв'язання деяких проблем на Java. За словами фанатів мови, синтаксис Kotlin простіший, чистіший та призводить до меншого роздування коду. Це допоможе вам більше зосередитися на розв'язанні актуальної проблеми, а не на складному синтаксисі. Крім того, можна використовувати Kotlin і Java разом в одному проекті, і це робить його дійсно потужним.

C є другою за популярністю мовою в індексі TIOBE, як і Java, їх спільнота наповнена досвідченими розробниками, які можуть запропонувати вам цінні поради про те, як писати код без помилок. Якщо використовувати Android NDK (Native Development Kit), то необхідно працювати з мовою C.

Мова C++ - це розширення C, з більш високорівневими функціями та підтримкою об'єктно-орієнтованого програмування. C++ також є улюбленою мовою розробників Android NDK.

Python – ще одна популярна мова, яку легко засвоїти та легко читати. Творці мови доклали додаткові зусилля, щоб синтаксис був максимально простим і зрозумілим. Це дійсно допомагає початківцям розробникам підтримувати високий рівень продуктивності з першого дня.

Мобільні додатки можуть бути згруповані в три категорії: нативні, гібридні та кросплатформні. Нативні додатки можуть повністю використовувати всі можливості та функції ОС, і вони є найшвидшими, коли справа доходить до продуктивності. Однак вам необхідно підтримувати різні кодові бази для різних мобільних платформ, оскільки кожна платформа використовує різні мови програмування.

Наприклад, платформа Android використовує Java та C/C ++ для розробки власних додатків. Платформа iOS від Apple покладається на Objective-C і Swift як на рідні мови. C# використовується платформою Windows Mobile для кодування власних додатків. Всі ці рідні мови програмування додатків скомпільовані, а не інтерпретуються.

Гібридні мобільні додатки насправді є веб-сайтами, які також призначені для роботи з мобільними пристроями. Користувач може отримати до них доступ через мобільний браузер, як якщо б вони відвідували веб-сайт на настільному комп'ютері. Комбінація HTML5, CSS і JavaScript є очевидним вибором, якщо необхідно розробляти веб-додатки.

Останнім часом з'явилася нова серія мобільних кросплатформних фреймворків. Ці фреймворки поєднують в собі кращі функції рідних додатків і гібридних додатків – вони швидші та легкі та можуть отримати доступ до повної потужності власного пристрою, але вони також програмуються з використанням JavaScript та інших веб-мов, тому можна повторно використовувати багато коду і не залежно від платформ.

Kotlin повільно змінює Java в нативній розробці під Android

JavaScript (в поєднанні з іншими технологіями HTML5) є найбільш використовуваною кросплатформною мобільною мовою розробки. Існують також інші мови, такі як Python і Ruby, але JavaScript має найширший спектр підтримки.

В магістерській роботі була використана мова Java в звязку з досвідом роботи та відповідністю властивостей мови для створення додатку.

4.2. Програмна реалізація та тестування мобільного додатку управління системою клімат-контролю Smart Home

Для імітації роботи системи управління клімат контролем, нам необхідно розробити пакет, який буде виконувати наступні функції: відправляти та отримувати дані на мікроконтролер та від нього; обробляти результати запитів, зберігати результати вимірів. Розраховувати середнє значення вимів, зберігати параметри зєднання, опитувати датчики на вулиці та в кімнатах у фоновому потоці. Для цього було розроблено пакет ArduinoTelnet. Такий пакет містить класи з з’єднання, коннекції, управління кімнатами, сенсорами.

Клас Arduino є базовим, користувач з ним взаємодіє, використовуючи відповідні методи. Цей клас відповідає за ініціалізацію всіх компонентів, запуск та зупинку потоку перевірки вимірювань. Клас має 2 конструктори:

public Arduino (String hostname, int hostport)

public Arduino (String hostname, int hostport, SensorDataBase dataBase)

Різниця між конструкторами полягає в тому, що другий розрахований на роботу з базою даних. Перші 2 параметри відповідають за адресу підключення та порту на який потрібно надсилати запити.

Функціонал класу складається з таких методів:

void start(int delay)

void stop()

Sensors getLast()

void setTemperature(float value)

void setHumidity(float value)

void setIllumination(float value)

Функція start() використовується для запуску потоку у фоні, який буде опитувати мікроконтролер та конвертувати отриману інформацію у певний формат, а також додавати середні значення до бази даних (якщо вона була ініціалізована). В якості параметра delay виступає час у мс, який поток має чекати перед тим як надіслати новий запит.

Функція stop() зупиняє фоновий поток, використовуючи метод interrupt). Це дозволяє коректно завершити поток у відповідності з документацією java [24].

Функція getLast() повертає об’єкт типу Sensors з останніми показниками вимірювань.

Функція setTemperature() виконує встановлення температури на мікроконтролері у відповідності з параметром value. Для передачі повідомлення функція запускає окремий поток, який і надсилає команду.

Функція setHumidity() аналогічно до функції setTemperature() встановлює вологість.

Функція setIllumination() аналогічно до функції setTemperature() встановлює значення освітленості.

Клас Connector використовується для встановлення з'єднання з мікроконтролером та обміну повідомленнями. Клас має 2 конструктори:

Connector(String hostname)

Connector(String hostname, int hostport)

Різниця між конструкторами полягає в тому, що в першому випадку в якості параметра виступає лише адрес підключення, а в другому випадку користувач може вибрати ще й порт. В першому конструкторі за замовчуванням використовується порт 23, стандартний порт протоколу TELNET.

Функцію клас має одну String sendCommand(String command). Її код наведено нижче:

```
public synchronized String sendCommand(String command) {
    String result = "";
    String com = command +
    "\n"; boolean success = false;
    while (!success) { try {
        Socket s = new Socket(host, port);
        s.getOutputStream().write(com.getBytes());
        byte buf[] = new byte[64 * 1024]; int r =
        s.getInputStream().read(buf);
        String data = new String(buf, 0,
        r); result = result + data; s.close();
        success = true;
      } catch (Exception ex) {}
    }
    return result;
```

Функція намагається у циклі while() відправити повідомлення серверу, якщо їй це вдається, то отримані дані конвертуються у строку формату String

та повертаються назад, якщо ж виконати підключення не вийшло, то функція спробує зробити це знов.

Клас Room використовується для збереження показників вимірів з кімнат, для цього в нього є відповідні поля:

humidity

temperature

illumination

Вони відповідають за зберігання даних про вологість, температуру та освітленість у приміщенні.

З методів у класі реалізовані такі функції:

synchronized void setHumidity(Float humidity)

synchronized void setTemperature(Float temperature)

synchronized void setIllumination(Float illumination)

synchronized Float getHumidity ()

synchronized Float getTemperature ()

synchronized Float getIllumination ()

Функції setHumidity(Float humidity), setTemperature(Float temperature) та setIllumination(Float illumination) встановлюють значення вологості, температури та освітлення.

Функції getHumidity (), getTemperature () та getIllumination () повертають поточне значення вологості, температури та освітленості.

Клас Sensors зберігає показники значень з усіх кімнат, а також датчиків, розташованих на вулиці. Для цього в ньому є відповідні поля та масив:

roomsNumber

humidity

temperature

illumination

pressure

rooms[]

Для доступу та роботи з класом у ньому реалізовані наступні методи:

synchronized void addMeasurements (Float temperature, Float humidity, Float illumination, Float pressure)

synchronized String toString()

synchronized void addRoomByIndex (int index, Room room)

synchronized void setHumidity(Float humidity)

synchronized void setTemperature(Float temperature)

synchronized void setIllumination(Float illumination)

synchronized void setPressure(Float pressure)

synchronized Float getHumidity ()

synchronized Float getTemperature ()

synchronized Float getIllumination ()

synchronized Float getPressure ()

synchronized int getRoomNumber ()

synchronized Room getRoomByIndex (int index)

synchronized Sensors getAverage (List<Sensors> list)

Функція addMeasurements додає нові дані до поточного об’єкта. Параметрами виступають, температура, вологість, освітленість та барометричний тиск.

Функція toString використовується для конвертації даних, які зберігає клас, у зміну формату String.

Функція addRoomBylndex додає показники вимірювань для кімнати. В якості параметрів необхідно вказати порядковий номер кімнати та об’єкт типу Room.

Функції setHumidity, setTemperature, setlllumination та setPressure використовуються для встановлення нових значень вологості, температури, освітленості та барометричного тиску.

Функції getHumidity, getTemperature, getIllumination, getPressure повертають значення вимірів вологості, температури, освітлення та барометричного тиску.

Функція getRoomNumber повертає кількість кімнат.

Функція getRoomBylndex повертає кімнату у відповідності з вказаним її порядковим номером.

Функція getAverage знаходить середнє значення всіх параметрів з переданого їй списку та повертає новий об’єкт з цими значеннями.

Клас Checker є спадкоємцем класу Thread та виконується у фоновому потоці. Він надсилає запити мікроконтролеру, виконує лексичний розбір, отриманої інформації та додає нові вимірювання до списку. При умові наявності бази даних з кожних 30 вимірювань розраховує середнє і додає до бази даних. Клас складається лише з перезавантаженого метода run, який виконується при запуску об’єкта цього класу.

4.3. Програмне забезпечення бази даних

Для роботи систем прогнозування, а також для оптимізації збереження інформацій, спрощення доступу до неї необхідно реалізувати базу даних. Це дозволить легко виконувати збереження даних у хмарі, імпорт та експорт даних. Для реалізації цієї задачі доцільним буде скористатися можливостями програмування для Android, а саме базою даних на мові SQLite.

Використання SQLite дозволяє зберігати базу даних локально у директорії програми. Для експорту та імпорту достатньо буде просто отримати доступ до файлу и виконати переписування або зчитування файлу.

UML діаграма класів наведена на рис. 4.1. Для реалізації роботи з базою даних на Android було створено пакет SensorDB. Він складається з наступних класів:

SensorDataBaseHelper

SensorDataBase

Для зручності роботи з даними необхідно розробити власне саму базу даних. Створимо базу даних 3 нормальної форми. У відповідності до 3NF

сформуємо по таблиці на кожну кімнату, таблицю для вуличних вимірювань та базову таблицю.

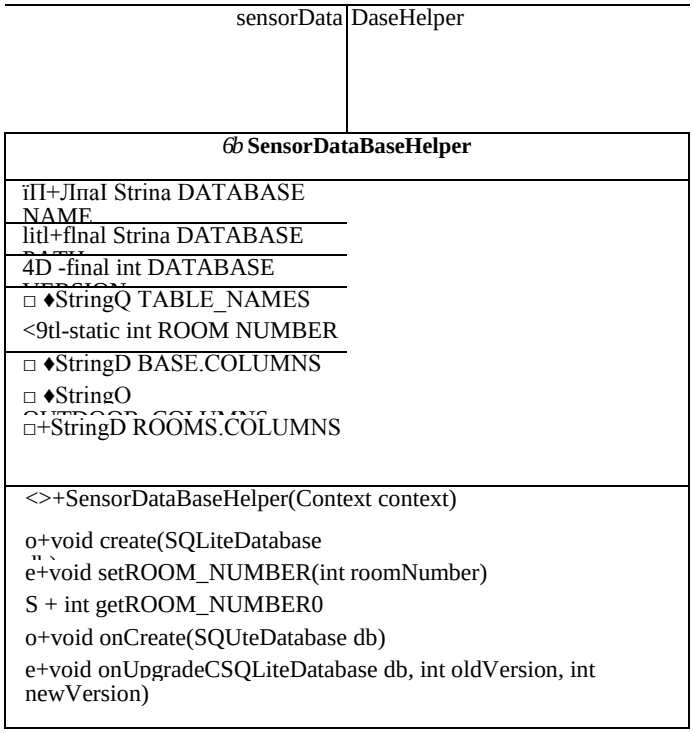


Рисунок 4.1 – UML діаграма класів для роботи з БД

При цьому у базовій таблиці будемо мати посилання на таблиці кімнат та зовнішніх вимірів. Діаграма бази даних для випадку 1 кімнати наведена на рис.4.2. Базова таблиця outdoor складається з таких стовбців:

_id

DATE

TIME

key_outdoor

key_room1

Стовпець _id є унікальним ідентифікатором запису у таблиці.

У полі DATE будемо зберігати дату, коли було зроблено запис.

У полі TIME зберігаємо час запису.

Поле key_outdoor є зовнішнім посиланням на запис у таблиці outdoor.

Поле key_room є зовнішнім посиланням на запис у таблиці room1. Таблиця outdoor складається з наступних стовпців:

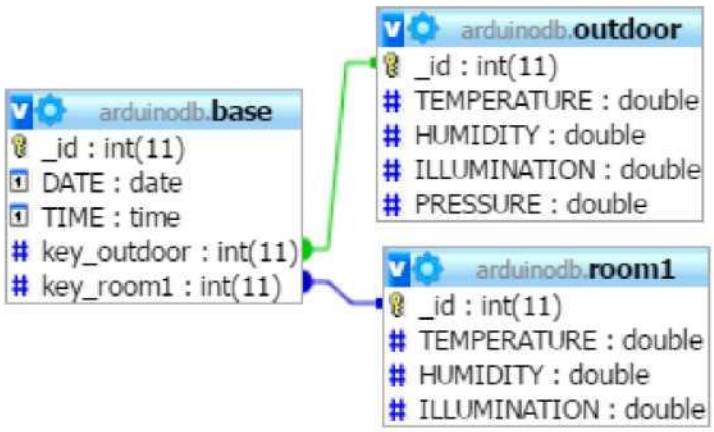


Рисунок 4.2 – Схема БД

id

TEMPERATURE

HUMIDITY

ILLUMINATION

PRESSURE

Стовпець _id є унікальним ідентифікатором запису у таблиці.

У полі TEMPERATURE зберігається значення температури на вулиці.

У полі HUMIDITY зберігається значення вологості на вулиці.

У полі ILLUMINATION зберігається значення освітленості на вулиці.

У полі PRESSURE зберігається значення атмосферного тиску на вулиці.

Таблиця room1 має такі стовпці:

_id

TEMPERATURE

HUMIDITY

ILLUMINATION

Стовпець _id є унікальним ідентифікатором запису у таблиці.

У полі TEMPERATURE зберігається значення температури.

У полі HUMIDITY зберігається значення вологості.

У полі ILLUMINATION зберігається значення освітленості.

У відповідності з правилами роботи з SQLite на Android ми маємо створити клас успадкований від класу SQLiteOpenHelper [26]. Далі необхідно перевизначити методи onCreate та onUpgrade. Вони відповідають за створення бази або її модифікацію, якщо версії бази даних відрізняються.

Базову таблицю створимо командою:

create table base (_id integer primary key autoincrement, DATE date NOT NULL,TIME time NOT NULL, key_outdoor integer, key_room1 integer);

Таблицю для зовнішніх вимірів створимо командою:

create table outdoor (_id integer primary key autoincrement, TEMPERATURE real default null, HUMIDITY real default null, ILLUMINATION real default null, PRESSURE real default null);

Таблицю для вимірів у кімнаті створимо командою:

create table room1 (_id integer primary key autoincrement, TEMPERATURE real default null, HUMIDITY real default null, ILLUMINATION real default null);

Для роботи користувача з базою даних було створено клас SensorDataBase. Він реалізує фунціонал, щоб забезпечити безпечну роботу з базою даних, та позбавити користувача писати SQL запити. У класі присутні наступні методи:

- List<Sensors> getData(String filter)
- void addData(Sensors sensor)
- void setRoomNumber(int number)
- void create ()
- List<Sensors> getData ()
- Sensors getLast()

- List<Sensors> getDataByDate (String date)
- List<Sensors> getDataByDateInRange (String dateBegin, String dateEnd)
- List<Sensors> getDataByTimeInRange (String timeBegin, String timeEnd)
- List<Sensors> getDataByDateAndTimeInRange (String date, String timeBegin, String timeEnd)
- List<Sensors> getDataByDateInRangeAndTimeInRange (String dateBegin, String dateEnd, String timeBegin, String timeEnd)
- File getBaseFile ()
- void drop()

Функція getData (з фільтром " String filter ") виконує вибір даних з бази даних відповідно до фільтру. Далі данні конвертуються у формат Sensors та додаються до списку, який і повертається.

Функція addData додає дані типу Sensors до бази.

Функція setRoomNumber встановлює кількість кімнат та перебудовує таблиці за необхідності.

Функція create створює таблиці.

Функція getData повертає усі значення з бази даних у форматі списку.

Функція getDataByDate виконує вибірку інформації з бази певну дату та повертає список.

Функція getDataByDateInRange виконує вибірку з бази даних за дату у вказаному проміжку.

Функція getDataByTimeInRange проводить вибірку інформації у вказаний період часу та повертає список

Функція getDataByDateAndTimeInRange виконує вибірку даних з бази за певну дату у певний проміжок часу.

Функція getDataByDateInRangeAndTimeInRange фільтрує дані з бази за певний період дати та часу.

Функція getBaseFile повертає посилання на файл, в якому зберігається БД.

Функція drop видаляє всі таблиці разом з даними.

Мобільний додаток управління системою клімат-контролю створюється на основі розроблених моделей і методів, структур підсистеми.

На рис. 4.1.представлено розгорнуте середовище Android Studio, яке дозволяє створити програмний код, та виконати моделювання роботи модулів системи.

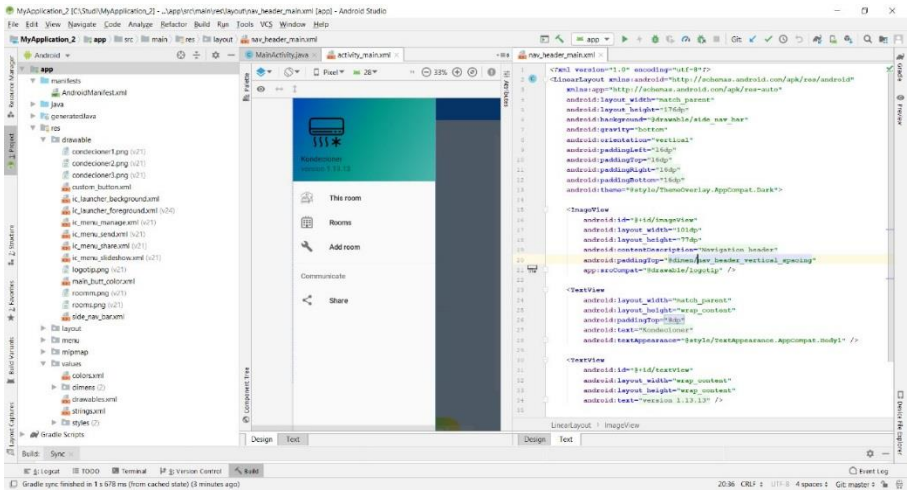


Рисунок 4.3. – Створення мобільного додатку в середовищі Android Studio

Середовище дозволяє сформувати ієрархічну систему інформаційних моделей та сформувати події зміни показників мікроклімату та введення різноманітних індикаторів користувачем або імітацією передачі інформації з датчиків на контролер. Так, наприклад, на рисунку 4.2. представлено формування інформаційної моделі з'єднання системи кондиціонеру з системою управління та датчиками показників, що впливають на зміни в роботі кондиціонеру.

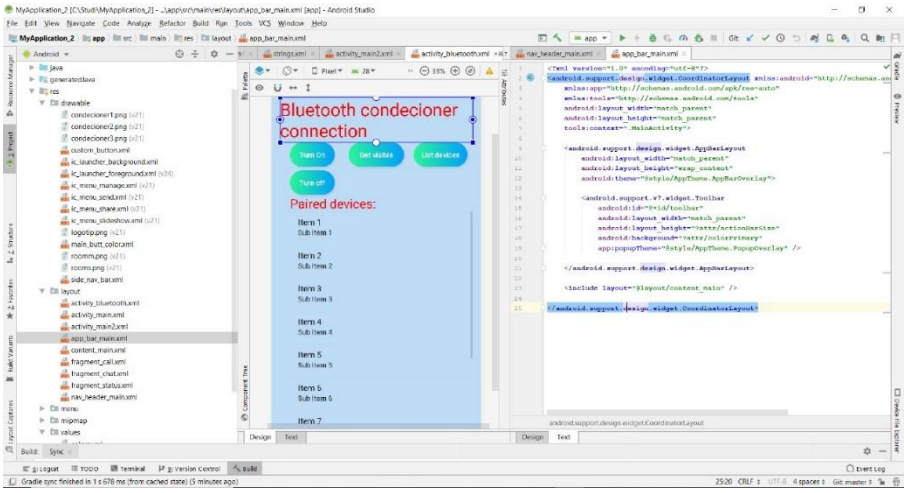


Рисунок 4.4. – Створення мобільного додатку в середовищі Android Studio

Для інформаційної моделі третього рівня – компоненти – на прикладі кондиціонера виконано формування інформаційної моделі з можливістю переходу на панель та одержання повідомлень управління.

Фрагменти коду створення bluetooth та інтернет зєднання представлені в роботі та в лістінгу коду, який здійснює пошук потрібного приладу та можливостей його з’єднання.

```
package com.example.myapplication_2;

import android.app.Activity;
public class Bluetooth extends Activity  {
    Button b1,b2,b3,b4;
    private BluetoothAdapter BA;
    private Set<BluetoothDevice>pairedDevices;
    ListView lv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_bluetooth);

        b1 = (Button) findViewById(R.id.button);
        b2=(Button)findViewById(R.id.button2);
        b3=(Button)findViewById(R.id.button3);
```

```
        b4=(Button)findViewById(R.id.button4);

        BA = BluetoothAdapter.getDefaultAdapter();
        lv = (ListView)findViewById(R.id.listView);
    }

    public void on(View v){
        if (!BA.isEnabled()) {
            Intent turnOn = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(turnOn, 0);
            Toast.makeText(getApplicationContext(), "Turned
on",Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText(getApplicationContext(), "Already on",
Toast.LENGTH_LONG).show();
        }
    }
    public void off(View v){
        BA.disable();
        Toast.makeText(getApplicationContext(), "Turned off"        ArrayList list =
new ArrayList();

        for(BluetoothDevice bt : pairedDevices) list.add(bt.getName());
        Toast.makeText(getApplicationContext(), "Showing Paired
Devices",Toast.LENGTH_SHORT).show();

        final ArrayAdapter adapter = new
ArrayAdapter(this,android.R.layout.simple_list_item_1, list);

        lv.setAdapter(adapter);
    }
```

За результатами виконання визначеного коду формується панель з’єднання, представлена на рис. 4.5.

Рисунок 4.5. – Створення панелі з’єднання кондиціонеру середовищі Android Studio

Визначення температури, зміни параметрів температури або вологості можливі після аналізу стану, введення та виведення спеціальних повідомлень здійснюється на основі авторежиму або заданих температурних (чи інших) параметрів. На рис. 4.6. представлено повідомлення щодо визначеного приладу (кондиціонеру), яке формується для користувача.

Рисунок 4.6. – Повідомлення стану системи в адаптивному інтерфейсі

На рис. 4.7., повідомлення для дій користувача як приклад адаптивного мінімалістського інтерйфейсу з позначенням рівня приладу та короткого зрозумілого повідомлення.

Рисунок 4.7. – Повідомлення для дій користувача в системі

Таким чином було проведено тестування інінціації системи управління – головна панель управління; пошуку з’єднаних пристроїв (другий рівень), стану системи (інформація датчиків – третій рівень системи) та формування інтерактивного діалогу для дій користувача.

4.3. Висновки

Таким чином була виконана програмна реалізація мобільного додатку управління за принципами клієнтоорієнтованого адаптивного інтерфейсу управління в системі розумного дому. Реалізовано на прикладі систем мікроконтролю. Імітована робота мікроконтролеру, передачі даних датчиків та надання інформації. Виконано тестування на мобільному додатку формування з’єднання з кондиціонером, формування повідомлення щодо стану температури та повідомлення щодо активних дій користувача.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	4	4
4	4	4
5	3	3
6	3	4
7	4	3
8	3	4
9	4	4
10	4	4
11	3	4
12	3	4
Сума балів	$СБ_1 = 43$	$СБ_2 = 45$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 44$	

Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Коваленко О.О. та Ракітянська Г.Б.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Отже, з отриманих даних таблиці 5.1 видно, що нова розробка має високий рівень комерційного потенціалу.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (5.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (5.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 22 дні;

t - число днів роботи розробника, t = 45 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 5.2.

Таблиця 5.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	6000	272,72	5	1363,6
Інженер-програміст	3500	159,09	45	7159,05
Всього:				8522,65

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 8522,65 = 852,26 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп}= (З_о + З_р)\cdot \frac{\beta}{100}, \quad (5.2)$$

$$Н_{зп}= (8522,65+852,26)\cdot \frac{36,3}{100} = 3403,09 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot\frac{Т}{12}, \quad (5.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 5.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп’ютер	10000	25	3	625
Всього:				625

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (5.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 5.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Пачка паперу	уп.	150	1	150
Ручка	шт.	10	1	10
Всього з урахуванням транспортних витрат				176

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п ; \quad (5.5)$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп’ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп’ютера (Ф=150 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0,9$).

$$В_е= 1,7 \cdot 0,6 \cdot 150 \cdot 0,9 = 137,7 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (5.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (8522,65 + 852,26) = 9374,91 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 8522,65+852,25+3403,09+625+176+137,7+9374,91= 23091,61 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (5.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{23091,61}{1} = 23091,61$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (5.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{23091,61}{0,9} = 25657,34 \text{ (грн.)}$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (5.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 25 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 25 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року

– на 200 користувачів, протягом другого року – на 175 користувачів, протягом третього року – 150 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 1000 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 200 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 25 \cdot 1000 + (200 + 25) \cdot 200 = 70000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 25 \cdot 1000 + (200 + 25) \cdot (200 + 175) = 109375 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 25 \cdot 1000 + (200 + 25) \cdot (200 + 175 + 150) = 143125 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \qquad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 5.1.

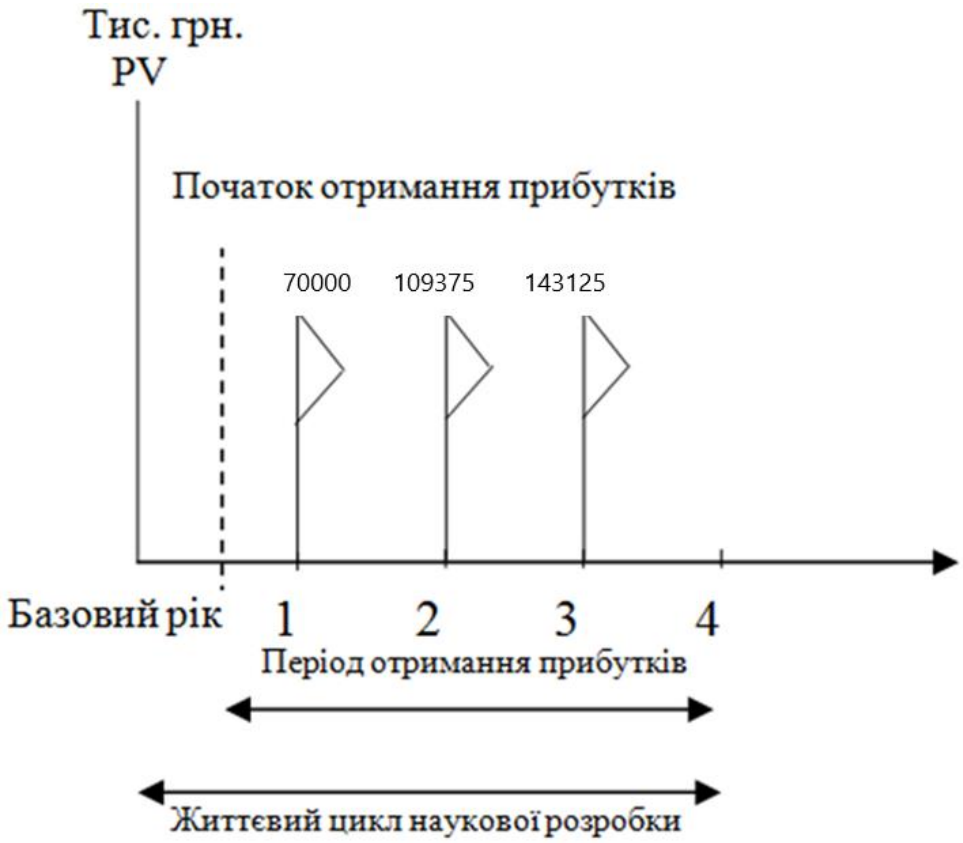


Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$\text{ПП} = \sum_1^m \frac{\Delta\Pi_i}{(1+\tau)^t} \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{25657,34}{(1+0,1)^0} + \frac{70000}{(1+0,1)^2} + \frac{109375}{(1+0,1)^3} + \frac{143125}{(1+0,1)^4} = 263439,92 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 263439,92 - 25657,34 = 237782,58 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (5.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{237782,58}{25657,34}} - 1 = 1,17 \text{ або } 117\ \%$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0,2 + 0,1 = 0,3$$

Оскільки $E_в = 117\% > \tau_{мін} = 0,3 = 30\%$, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

$$T_{ок} = \frac{1}{1,17} = 0,85 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи було обґрунтовано доведено актуальність теми дослідження в зв’язку зі змінами технологій виробництва та формування систем розумного дому, домашньої та офісної автоматизації.

В роботі розглянуті концепції побудови систем управління розумним домом, удосконалені моделі та методи формування адаптивного інтерфейсу системи управління розумним домом.

Технології управління потребують нових підходів до побудови адаптивних інтерфейсів з використанням мінімальних, простих інструментів для створення ефективних інформаційних моделей.

Незважаючи на позитивні тенденції в сфері продажі готових рішень управління та реалізації технологій розумного дому, залишилася потреба в недорогому та відкритому варіанті реалізації, який кожен споживач зможе налаштувати під свої потреби та розширювати в залежності від змін в списку приладів та технологій зв’язку.

Удосконалення запропонованої моделі сформовано відповідно до таких напрямів:

4. Врахування особливостей користувачів відносно їх комп’ютерних навичок, знань предметної області, персональних даних, попередньою історією роботи з пристроями, профілю користувача тощо.

5. Запровадження вибраної системи дизайну з врахуванням складності системи та алгоритмів формування інформаційних моделей.

6. Вибір моделей візуалізації та рекомендацій щодо роботи пристроїв.

В магістерській роботі було створено веб-додаток, мобільні додатки в середовищі Arduino та Android. Програмна реалізація модулю клімат контролю дозволила перевірити роботу ієрархічно блочної системи управління, яка використовує графову структуру.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Best Smart Home Devices for 2019 https://www.pcmag.com/article/303814/the-best-smart-home-devices-for-2019

2. Технология «Умный дом» https://www.inspectorgadgets.ru/post/smart-home-explained

3. Береговський В. В. Методи та моделі автоматизованого проектування системи “інтелектуального будинку” на базі нейроконтролерів / Береговський В. В., Теслюк В. М., Матвійчук К. В., Денисюк П. Ю. // Науковий вісник НЛТУ України : Збірник науково-технічних праць. - Вип. 26.7. - Львів : РВВ НЛТУ України, 2016. - С. 342-349.

4. Теслюк В. М. Автоматизація системного рівня проектування інтелектуального будинку / Теслюк В. М., Береговський В. В., Пукач А. І., Сидор А. Р. // Збірник наукових праць ІППМЕ ім. Г. Є. Пухова НАН України. - Вип. 67. - Київ, 2013. - С. 138-147.

5. Системы “Умный дом” [Електронний ресурс]. URL: http://www.vashdom.ru/articles/research_2.htm

6. Элсенпитер Р. Умный Дом строим сами / Роберт К. Элсенпитер, Тоби Дж. Велт. - М. : Кудиц - Образ, 2005. - 384 с.

7. Головач Влад В.. Дизайн пользовательского интерфейса. Искусство мыть слона — 2009. — 94 с.

8. Якоб Нильсен, Хоа Лоранжер. Web-дизайн: удобство использования Web-сайтов Prioritizing Web Usability — М.: «Вильямс», 2007. — 368 с.

9. Стив Круг. Как сделать сайт удобным. Юзабилити по методу Стива Круга = Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems — СПб.: Питер, 2010. — С. 208.

10. Джеф Раскин. Интерфейс: новые направления в проектировании компьютерных систем — Символ-Плюс. — 2004.

11. Гарретт Д. Веб-дизайн: книга Джесса Гарретта. Элементы опыта взаимодействия. The Elements of User Experience: User-Centered Design for the Web — Символ-Плюс, 2008. — С. 192.

12. Бугай А. А. Концептуальна модель адаптивного веб-інтерфейсу користувача з використанням інтелектуальних технологій / А. А. Бугай, В. В. Олійник // Адаптивні системи автоматичного управління. - 2018. - № 1. - С. 15-22. - Режим доступу: http://nbuv.gov.ua/UJRN/asau_2018_1_4

13. Адаптивний дизайн http://smarton.com.ua/smart home/systema umniy dom intro - Система умный дом. - Режим доступу :http://intelcity.com.ua/comfort house -

14. Кто отвечает за климат-контроль в доме?- Режим доступу:http://smarton.com.ua/smart home/klimat kontrol v umnom dome -

15. Microsoft's HoloLens Is "Something Different" Than Oculus or Morpheu Режим доступу : http://www.gamespot.com/articles/microsoft-s-hololens-is-something-different-than-o/1100-6424809/ - Дата доступу : 14.05.2013.

16. Meizu выходит на рынок «Умного дома» - Режим доступу: http://itc.ua/news/meizu-vyihodit-na-ryinok-umnogo-doma-s-pomoshhyu-platformyi- lifekit-lineyki-raznogo-roda-ustroystv/ -

17. Orvibo Allone Wi-fi - Режим доступу :http://www.geekbuying.com/item/Orvibo-Allone-WiF i—IR—RF -Remote-Control- Smart-Home-Automation-Rechargeable-Battery-for-IOS-Android-Mobile- 338990.html -

18. Gulla F., Ceccacci S. Design Adaptable and Adaptive User Interfaces: a Method to Manage the Information // Ambient Assisted Living: Italian Forum 2014. 2014. № 3, pp.2-4.

19. Makris N. Creating Adaptable and Adaptive User Interface Implementations in Model Driven Developed Software // Radboud University Nijmegen Press. 2014. № 3,pp.8-12.

20. Ramachandran K. Adaptive user interfaces for health care applications // IBM DeveloperWorks. 2009. № 2 (2009). С.2-5.

21. Fowler M. Microservices - a definition of this new architectural term [Електронний ресурс] // martinfowler.com. 2014. URL: https://martinfowler.com/articles/microservices.html. Дата звернення: 28.3.2018

22. Kim D. Reinforcement Learning-Based Dynamic Adaptation Planning Method for Architecture-based Self-Managed Software / Dongsun Kim, Sooyong Park // 2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, 2009, pp.76-85.

23. Романенко А.Ю. Узагальнена модель розпізнавання голосових команд / А. Ю. Романенко, В. В. Олійник // Міжвідомчий науково-технічний збірник «Адаптивні Системи Автоматичного Управління», К:Політехніка - 2017. - Т.1,№30 - С. 130-139.

24. Arduino Uno. - Режим доступу : http://arduino.ua/ru/hardware/Uno

25. Android - Режим доступу : http://www.3dnews.ru/581873 - Полный обзор изменений в Android M. - Режим доступу : https://www.iguides.ru/main/gadgets/google/android_m_review/-

26. Фрагментация Android - Режим доступу : http://habrahabr.ru/post/188738/ -

27. Android Fragmentation Vizualized - Режим доступу http://opensignal.com/reports/2014/androidfragmentation/#android_version_timeseries - -

28. Android Studio Режимдоступу https://developer.android.com/sdk/index.html

Д О Д А Т К И

Додаток А. Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"____ "_____2019 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Удосконалення методів та інструментів створення адаптивного інтерфейсу для управління пристроями в системі Smart Home**» за спеціальністю**

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____

"____"_____2019 р.

к. т. н., доц. О.О. Квасенко

Виконав

_____

студент гр. 1ПІ-18м Р.І. Підгородецький

"____"_____2019 р.

В

і

н

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «».

Галузь застосування –системи домашньої та офісної автоматизації.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є розробка адаптивного інтерфейсу для підвищення якості управління пристроями в системі розумного дому (Smart Home) та його програмна реалізація на прикладі системи мікроконтролю

Призначення роботи – підвищення якості управління модулями системи «Розумний дім».

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

29. The Best Smart Home Devices for 2019 https://www.pcmag.com/article/303814/the-best-smart-home-devices-for-2019

30. Технология «Умный дом» https://www.inspectorgadgets.ru/post/smart-home-explained

31. Джеф Раскин. Интерфейс: новые направления в проектировании компьютерных систем — Символ-Плюс. — 2004.

4. Технічні вимоги

Вихідні дані до роботи: міжнародні стандарти та протоколи управління пристроями; операційна система Android OS, Тактова частота процесору 1.3

GHz, Ядра ЦПУ 2-8 АКМз типом взаємдії BIG.little. Форма реалізації - у вигляді веб-додатку та Android додатку.

5. Конструктивні вимоги.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– ілюстративний матеріал;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз теоретико-практичних підходів до побудови адаптивних інтерфейсів реалізації технології «Розумний будинок»	04.09.2019 – 14.09.2019
2	Моделювання системи управління розумним будинком	15.09.2019 – 20.10.2019
3	Програмне забезпечення взаємодії приладів розумного дому	21.10.2019 – 15.11.2019
4	Програмне забезпечення мобільного додатку системи розумного дому	16.11.2019 – 21.11.2019
5	Економічна частина	22.11.2019 – 01.12.2019

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б. Лістинг вихідного коду

```
<?php

$uri = urldecode(
    parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH)
);

// This file allows us to emulate Apache's "mod_rewrite" functionality from the
// built-in PHP web server. This provides a convenient way to test a Laravel
// application without having installed a "real" web server software here.
if ($uri !== '/' && file_exists(__DIR__.'/public'.$uri)) {
    return false;
}

require_once __DIR__.'/public/index.php';
```

User.php

```
<?php

namespace App;

use Illuminate\Notifications\Notifiable;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;
```

```
class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
```

```
        'email_verified_at' => 'datetime',
    ];
}
```

IndexController.php

```
<?php

namespace App\Http\Controllers;

use App\Arduino;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Log;

class IndexController extends Controller
{
   public function index(){
       $ard = Arduino::where('user_id' ,Auth::id())->first();
       $uid = $ard->uid;
       return view('index',compact('uid'));
   }

   public function getData(Request $request){
       return Arduino::where('uid',$request->get('uid'))->first();
   }
    public function updateData(Request $request){
       $data = $request->get('data');
```

```
    $arduino = Arduino::where('uid',$request->get('uid'))->first();
    $arduino->update($data);
    if ($data['temperature']){
        if($arduino->temperature_graph)
         $arr = $arduino->temperature_graph;
        else
            $arr = [];
        if (count($arr)>10){
            array_splice($arr,0,1);
            $arr[]=$data['temperature'];
        }else{
            $arr[]=$data['temperature'];
        }
        $arduino->temperature_graph = $arr;
        $arduino->update();
    }
    return 'done';
 }

public function add(){
    $arduino = new Arduino();
    $arduino->uid= 'abc';
    $arduino->user_id=1;
    $arduino->temperature= '19';
    $arduino->temperature_graph=[17,18,19,25];
    $arduino->temp=[17,18];
    $arduino->thermoregulation=1;
    $arduino->color=['red'=>0,'green'=>0, 'blue'=>0];
```

```
        $arduino->colorAuto=0;
        $arduino->music=0;
        $arduino->save();
        dd($arduino);
    }
}
```

main.js

```
/**
 * Created by AndriiSydoruk on 5/11/16.
 */

var PATH = 'https://25.29.29.170:1243/';



$(document).ready(function(){
    $('#nav-icon1').on('click', function () {
        $('#content1').toggleClass('hidden');
        $('#nav-icon1').toggleClass('open');

    });
    $('#nav-icon2').on('click', function () {
        $('#content2').toggleClass('hidden');
        $('#nav-icon2').toggleClass('open');

    });
```

```
$('#nav-icon3').on('click', function () {
    $('#content3').toggleClass('hidden');
    $('#nav-icon3').toggleClass('open');
});
$('#nav-icon5').on('click', function () {
    $('#content5').toggleClass('hidden');
    $('#nav-icon5').toggleClass('open');

});

//startGraphik();
actionListeners();


//graphics

getElecricityArr();

getEnvTempArr();
getLastTempArr();



getTempRange();

getTempTh();

getTermoregulations();
```

```
});


        setLight(1);
    else
        setLight(0);
});

$("#stepper1").on("input change", function() {
    setTempRange($("#stepper1").val());
});
$("#stepper2").on("input change", function() {
    setTempTh($("#stepper2").val());
});




////33333333

//444


//   thermoregulation

$('#thermoregulation').click(function(){
    if($('#thermoregulation').prop('checked'))
```

```
            setTermoregulations(1);
        else
            setTermoregulations(0);
    });



}


function User (name){

    this.name  = name;
    var self = this;


    this.user = document.createElement('li');
    var text = document.createElement('p');
    text.innerText= this.name;
    var img  = document.createElement('img');
    img.setAttribute('src', 'img/profileIcom.png');
    img.setAttribute('height', '20px');

    $(text).prepend(img);

    this.user.appendChild(text);
    var closeBtn = document.createElement('a');
    closeBtn.innerText='x';
```

```
    this.user.appendChild(closeBtn);


    $(closeBtn).click(function(){
        self.removeUser();
    });
    return this.user
}

User.prototype.removeUser = function (){
    $(this.user).remove();

};


function CameraViewer (){
    var self = this;
    this.camera = document.createElement('div');
    this.camera.className = 'cameraBack';

    this.img = document.createElement('img');
    this.img.setAttribute('src', PATH+'cam.jpg');

    var closeBtn = document.createElement('a');
    closeBtn.innerText='x';

    this.camera.appendChild(this.img);
```

```
    this.camera.appendChild(closeBtn);

    $(closeBtn).click(function(){
        self.closeCamera();
    });
    self.updateCamera();



    //return this.camera;

    $('body').append(this.camera);

}
var stopCamera = false;



//requests

function checkUpdate(){
    var url = PATH+'param_access.php?act=read&param=wpn_alarm_mode_in';
    $.ajax({
        url: url,
        type: 'get',
        crossDomain: true,
        //dataType: 'jsonp',
        error: function (xhr, status, error) {
            console.log(xhr.responseText + '|\n' + status + '|\n' + error);
        },
```

```
        success: function (data) {
            console.log('some data'+ data);
            //getContent(data);
        }
    });
}


function getTempTh(){
    var url = PATH+'param_access.php?act=read&param=epm_temp_thresh_in';
    $.ajax({
        url: url,
        type: 'get',
        crossDomain: true,
        error: function (xhr, status, error) {
            console.log(xhr.responseText + '|\n' + status + '|\n' + error);
        },
        success: function (data) {
            console.log(data);
            $('#stepper2').val(parseInt(data));
        }
    });
}
function setTempTh(val){
    var url = PATH+'param_access.php?act=write&param=epm_temp_thresh_in';
    $.ajax({
        url: url,
        type: 'post',
```

```
        crossDomain: true,
        data: "+val,
        error: function (xhr, status, error) {
            console.log(xhr.responseText + '|\n' + status + '|\n' + error);
        },
        success: function (data) {
            //console.log(data);
        }
    });

}
function getTermoregulations(){
    var url = PATH+'param_access.php?act=read&param=epm_temp_mode_in';
    $.ajax({
        url: url,
        type: 'get',
        crossDomain: true,
        error: function (xhr, status, error) {
            console.log(xhr.responseText + '|\n' + status + '|\n' + error);
        },
        success: function (data) {
            console.log(data);
            if(parseInt(data))
                $('#thermoregulation').prop('checked', true);
            else
                $('#thermoregulation').prop('checked', false);
        }
    });
```

```
}
function setTermoregulations(val){
    var url = PATH+'param_access.php?act=write&param=epm_temp_mode_in';
    $.ajax({
        url: url,
        type: 'post',
        crossDomain: true,
        data: "+val,
        error: function (xhr, status, error) {
            console.log(xhr.responseText + '|\n' + status + '|\n' + error);
        },
        success: function (data) {
            //console.log(data);
        }
    });

}

function getRgb2(){
    var url = PATH+'param_access.php?act=read&param=epm_light_rgb_in';
    $.ajax({
        url: url,
        type: 'get',
        crossDomain: true,
        error: function (xhr, status, error) {
            console.log(xhr.responseText + '|\n' + status + '|\n' + error);
        },
        success: function (data) {
```

```
            console.log(data)
            var arr = data.split(', ')
            $('#redLight').val(parseInt(arr[0]));
            $('#greenLight').val(parseInt(arr[1]));
            $('#blueLight').val(parseInt(arr[2]));



        }
    });
}
function setRgb2(val){
    var url = PATH+'param_access.php?act=write&param=epm_light_rgb_in';
    $.ajax({
        url: url,
        type: 'post',
        crossDomain: true,
        data: "+val,
        error: function (xhr, status, error) {
            console.log(xhr.responseText + '|\n' + status + '|\n' + error);
        },
        success: function (data) {
            //console.log(data);
        }
    });

}
success: function (data) {
            //console.log(data);
```

```
        }
    });

}

}

function resetAll (){
    setFall(0);
    setCalp(0);
    setPanicButton(0);
}




//// end requests
function parseArray(data){
    var res = data.split("\n");
    //console.log(res);
    var tempArr  = [];
    for (var i=0;i<res.length;i++ ){
        var temp = res[i].split(' ');
        var obj = {date:  temp[0], value: temp[1]};
        tempArr.push(obj);
    }
    return tempArr;
```

```
}


var options = {
    // Don't draw the line chart points
    showPoint: true,
    showArea: true,
    width: '600px',
    height: '300px'
};
var options2 = {
    // Don't draw the line chart points
    showPoint: true,
    showArea: true,
    width: '400px',
    height: '300px'
};
var options3 = {
    // Don't draw the line chart points
    showPoint: true,
    showArea: true,
    width: '360px',
    height: '300px'
};



// GRAPHICS GRAPHICS GRAPHICS GRAPHICS GRAPHICS GRAPHICS GRAPHICS GRAPHICS GRAPHICS GRAPHICS GRAPHICS GRAPHICS


function setVoltageGraph(arr){
```

```
    var labels = [];
    var series = [];
    var last = arr.length-2;
    $('#currentVoltageValue').text(parseInt(arr[last].value/1024*400) +' V');
    for(var i=last-10;i<last;i++){
        labels.push(arr[i].date)
        series.push(arr[i].value)

    }
    var data = {
        labels: labels,
        series: [series]
    };
    new Chartist.Line('#voltageGraph', data, options);
};

    new Chartist.Line('#co2Graph', co2, options2);
}

function setEnviromentTemp(arr){
    var labels = [];
    var series = [];
    var last = arr.length-2;
    //$('#currentVoltageValue').text(arr[last].value +' V');
    for(var i=last-10;i<last;i++){
        labels.push(arr[i].date.substring(0,5));
        series.push(arr[i].value)

    }
    var envTemp = {
```

```
        labels: labels,
        series: [series]
    };

    //console.log(labels);
    //console.log(series);

    new Chartist.Line('#envTempGraph', envTemp, options3);

}

function setTempGraph(arr){
    var labels = [];
    var series = [];
    var last = arr.length-2;
    $('#currentTemp').text(arr[last].value +'°');
    for(var i=last-10;i<last;i++){
        labels.push(arr[i].date.substring(0,5));
        series.push(arr[i].value)
    }
    var envTemp = {
        labels: labels,
        series: [series]
    };

    new Chartist.Line('#lastTempGraph', envTemp, options2);

}
```

class Bluetooth

```
package com.example.myapplication_2;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;

import android.widget.Toast;

import java.util.ArrayList;
import java.util.Set;

public class Bluetooth extends Activity  {
    Button b1,b2,b3,b4;
    private BluetoothAdapter BA;
    private Set<BluetoothDevice>pairedDevices;
    ListView lv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_bluetooth);

        b1 = (Button) findViewById(R.id.button);
        b2=(Button)findViewById(R.id.button2);
        b3=(Button)findViewById(R.id.button3);
        b4=(Button)findViewById(R.id.button4);

        BA = BluetoothAdapter.getDefaultAdapter();
        lv = (ListView)findViewById(R.id.listView);
    }

    public void on(View v){
        if (!BA.isEnabled()) {
            Intent turnOn = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(turnOn, 0);
            Toast.makeText(getApplicationContext(), "Turned
on",Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText(getApplicationContext(), "Already on",
Toast.LENGTH_LONG).show();
        }
    }

    public void off(View v){
        BA.disable();
        Toast.makeText(getApplicationContext(), "Turned off"
,Toast.LENGTH_LONG).show();
    }


    public  void visible(View v){
        Intent getVisible = new
Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
```

```
        startActivityForResult(getVisible, 0);
    }


    public void list(View v){
        pairedDevices = BA.getBondedDevices();

        ArrayList list = new ArrayList();

        for(BluetoothDevice bt : pairedDevices) list.add(bt.getName());
        Toast.makeText(getApplicationContext(), "Showing Paired
Devices",Toast.LENGTH_SHORT).show();

        final ArrayAdapter adapter = new
ArrayAdapter(this,android.R.layout.simple_list_item_1, list);

        lv.setAdapter(adapter);
    }
}
```

class CallFragment

```
package com.example.myapplication_2;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import static com.example.myapplication_2.MainActivity.room1;
import static com.example.myapplication_2.MainActivity.room2;

public class CallFragment extends Fragment {

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

        setHasOptionsMenu(true);
        TextView textView = v.findViewById(R.id.text_call);

        textView.setText("Temperature in this room is "+room2.x+" degree");
        return v;
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        inflater.inflate(R.menu.menu_calls, menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.action_call) {
            Toast.makeText(getActivity(), "Condecioner in " +
item.getTitle()+" will be cleaned", Toast.LENGTH_SHORT)
                    .show();
        }
        return true;
    }
}
```

class ChatFragment

```
package com.example.myapplication_2;

import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import static com.example.myapplication_2.MainActivity.room1;


public class ChatFragment extends Fragment {



    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View v =inflater.inflate(R.layout.fragment_chat, container, false);
        setHasOptionsMenu(true);
        TextView textView = v.findViewById(R.id.text_chat);

        textView.setText("Temperature in this room is "+room1.x+" degree");

        return v;

    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        inflater.inflate(R.menu.menu_chats, menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.action_chat) {
            Toast.makeText(getActivity(), "Condecioner in " +
item.getTitle()+" will be cleaned", Toast.LENGTH_SHORT)
                    .show();

        }
        return true;
    }
}
```

class Main2Activity

```
package com.example.myapplication_2;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.support.design.widget.TabItem;
import android.support.design.widget.TabLayout;
import android.support.v4.content.ContextCompat;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

public class Main2Activity extends AppCompatActivity {

    Toolbar toolbar;
    TabLayout tabLayout;
    ViewPager viewPager;
    PagerAdapter pageAdapter;
    TabItem tabChats;
    TabItem tabStatus;
    TabItem tabCalls;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);


        tabLayout = findViewById(R.id.tablayout);
        tabChats = findViewById(R.id.tabChats);
        tabStatus = findViewById(R.id.tabStatus);
        tabCalls = findViewById(R.id.tabCalls);
        viewPager = findViewById(R.id.viewPager);

        toolbar = findViewById(R.id.toolbar);
        toolbar.setTitle(getResources().getString(R.string.app_name));
        setSupportActionBar(toolbar);

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);


        // getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        ////////////////not realased
/*
        if (ChatFragment.flg){
            ////////////////////////crash!! Maybe
            Intent i = new Intent(Main2Activity.this, MainActivity.class);
            startActivity(i);
        }*/


        pageAdapter = new PagerAdapter(getSupportFragmentManager(),
tabLayout.getTabCount());
        viewPager.setAdapter(pageAdapter);
        viewPager.addOnPageChangeListener(new
TabLayout.TabLayoutOnPageChangeListener(tabLayout));

        tabLayout.addOnTabSelectedListener(new
TabLayout.OnTabSelectedListener() {
```

```
            @Override
            public void onTabSelected(TabLayout.Tab tab) {
                viewPager.setCurrentItem(tab.getPosition());
                if (tab.getPosition() == 1) {

toolbar.setBackgroundColor(ContextCompat.getColor(Main2Activity.this,
                            R.color.colorAccent));

tabLayout.setBackgroundColor(ContextCompat.getColor(Main2Activity.this,
                            R.color.colorAccent));
                    if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.LOLLIPOP) {

getWindow().setStatusBarColor(ContextCompat.getColor(Main2Activity.this,
                                    R.color.colorAccent));
                    }
                } else if (tab.getPosition() == 2) {

toolbar.setBackgroundColor(ContextCompat.getColor(Main2Activity.this,
                            android.R.color.darker_gray));

tabLayout.setBackgroundColor(ContextCompat.getColor(Main2Activity.this,
                            android.R.color.darker_gray));
                    if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.LOLLIPOP) {

getWindow().setStatusBarColor(ContextCompat.getColor(Main2Activity.this,
                                    android.R.color.darker_gray));
                    }
                } else {

toolbar.setBackgroundColor(ContextCompat.getColor(Main2Activity.this,
                            R.color.colorPrimary));

tabLayout.setBackgroundColor(ContextCompat.getColor(Main2Activity.this,
                            R.color.colorPrimary));
                    if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.LOLLIPOP) {

getWindow().setStatusBarColor(ContextCompat.getColor(Main2Activity.this,
                                    R.color.colorPrimaryDark));
                    }

                }
            }

            @Override
            public void onTabUnselected(TabLayout.Tab tab) {

            }

            @Override
            public void onTabReselected(TabLayout.Tab tab) {

            }

        });


    }        /*Toolbar toolbar = findViewById(R.id.toolbar3);
        setSupportActionBar(toolbar);
```

```
        getSupportActionBar().setDisplayShowHomeEnabled(true);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setTitle("Rooms");


        SectionsPagerAdapter sectionsPagerAdapter = new
SectionsPagerAdapter(this, getSupportFragmentManager());
        ViewPager viewPager = findViewById(R.id.view_pager);
        viewPager.setAdapter(sectionsPagerAdapter);
        TabLayout tabs = findViewById(R.id.tabs);
        tabs.setupWithViewPager(viewPager);
        FloatingActionButton fab = findViewById(R.id.fab);


        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH LONG)
                        .setAction("Action", null).show();
            }
        });*/

////////////////function to come back when u are in
rooms//////////////////
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == android.R.id.home) {
            this.finish();
        }

        return super.onOptionsItemSelected(item);
    }
}
```

class MainActivity

```
package com.example.myapplication_2;

import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.view.Gravity;
import android.view.View;
import android.support.v4.view.GravityCompat;
import android.support.v7.app.ActionBarDrawerToggle;
import android.view.MenuItem;
import android.support.design.widget.NavigationView;
import android.support.v4.widget.DrawerLayout;

import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.widget.Button;
import android.widget.EditText;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;


public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener,
SeekBar.OnSeekBarChangeListener {

    public static Room room1;
    public static Room room2;
    public static Room room3;
    private TextView txtView;

    String PlayMarkLink;
    //private TextView textView11;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final SeekBar seekBar = (SeekBar) findViewById(R.id.seekBar1);
        seekBar.setOnSeekBarChangeListener(this);

        /*mTextView = (TextView)findViewById(R.id.textView);
        mTextView.setText("0");*/

        String def = "defoult, is 20 ";
        room1 = new Room(def);
        room2 = new Room(def);
        room3 = new Room(def);
        Button btnrm1 = findViewById(R.id.btnrm1);
        Button btnrm2 = findViewById(R.id.btnrm2);
        Button btnrm3 = findViewById(R.id.btnrm3);

        txtView = findViewById(R.id.text11);


        txtView = (TextView) findViewById(R.id.text11);

        btnrm1.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View view) {
                room1 = new Room((String.valueOf(seekBar.getProgress())));
            }
        });
        btnrm2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                room2 = new Room(txtView.getText().toString());
            }
        });
        btnrm3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                room3 = new Room(txtView.getText().toString());
            }
        });


        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

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

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
```

```
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();

        if (id == R.id.nav_home) {
            // Handle the camera action
        } else if (id == R.id.nav_rooms) {
            //Toast.makeText(getApplicationContext(), "LolText",
Toast.LENGTH_LONG).show();

            //////////////CHANGE SCENE
////////////////////////////////////////////////
            Intent i = new Intent(MainActivity.this, Main2Activity.class);
            startActivity(i);
        } else if (id == R.id.nav_slideshow) {

        } else if (id == R.id.nav_addroom) {
            //////////////CHANGE SCENE
////////////////////////////////////////////////
            Intent b = new Intent(MainActivity.this, Bluetooth.class);
            startActivity(b);

        } else if (id == R.id.nav_share) {
            Intent shareintent = new Intent();
            String shareBody = "Telegram";
            PlayMarkLink = "https://play.google.com/store";
            shareintent.setType("text/plain");
            shareintent.setAction(Intent.ACTION_SEND);
            shareintent.putExtra(Intent.EXTRA_SUBJECT,PlayMarkLink);

            ////////////////////////////// For send
button/////////////////////////////////////////
            //shareintent.putExtra(Intent.EXTRA CHOOSER TARGETS, shareBody);
            //shareintent.putExtra(Intent.EXTRA_INITIAL_INTENTS,
myInitialIntentArray);
           //shareintent.putExtra(Intent.EXTRA_TEXT,shareBody);

            startActivity(Intent.createChooser(shareintent, "share via"));
        } else if (id == R.id.nav_send) {

        }

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

    }

    @Override
    public void onProgressChanged(SeekBar seekBar, int i, boolean b) {

    }


    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        txtView.setText(String.valueOf(seekBar.getProgress()));
    }
```

class PagerAdapter

```
package com.example.myapplication_2;

import android.content.Context;
import android.support.annotation.Nullable;
import android.support.annotation.StringRes;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;

import com.example.myapplication_2.CallFragment;
import com.example.myapplication_2.ChatFragment;
import com.example.myapplication_2.R;
import com.example.myapplication_2.StatusFragment;

/**
 * A [FragmentPagerAdapter] that returns a fragment corresponding to
 * one of the sections/tabs/pages.
 */
public class PagerAdapter extends FragmentPagerAdapter {

    private int numOfTabs;

    public PagerAdapter(FragmentManager fm, int numOfTabs) {
        super(fm);
        this.numOfTabs = numOfTabs;
    }

    @Override
    public Fragment getItem(int position) {
        switch (position) {
            case 0:
                return new ChatFragment();
            case 1:
                return new StatusFragment();
            case 2:
                return new CallFragment();
            default:
                return null;
        }
    }

    @Override
    public int getCount() {
        return numOfTabs;
    }
}
```

class Room

```
package com.example.myapplication_2;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.NavigationView;
import android.support.v7.app.AppCompatActivity;
import android.view.MenuItem;
import android.widget.TextView;

public class Room {

    public String x = "All clean";

    public Room(String a) {
        x = String.valueOf(a);
    }
}
```

class StatusFragment

```
package com.example.myapplication_2;

import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import static com.example.myapplication_2.MainActivity.room2;
import static com.example.myapplication_2.MainActivity.room3;

public class StatusFragment extends Fragment {

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

        setHasOptionsMenu(true);
        TextView textView = v.findViewById(R.id.text_status);

        textView.setText("Temperature in this room is "+room3.x+" degree");
        return v;
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        inflater.inflate(R.menu.menu_status, menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        if (item.getItemId() == R.id.action_status) {
            Toast.makeText(getActivity(), "Condecioner in " +
item.getTitle()+" will be cleaned", Toast.LENGTH_SHORT)
                    .show();

        }
        return true;
    }
}
```

Додаток В. Ілюстративний матеріал

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

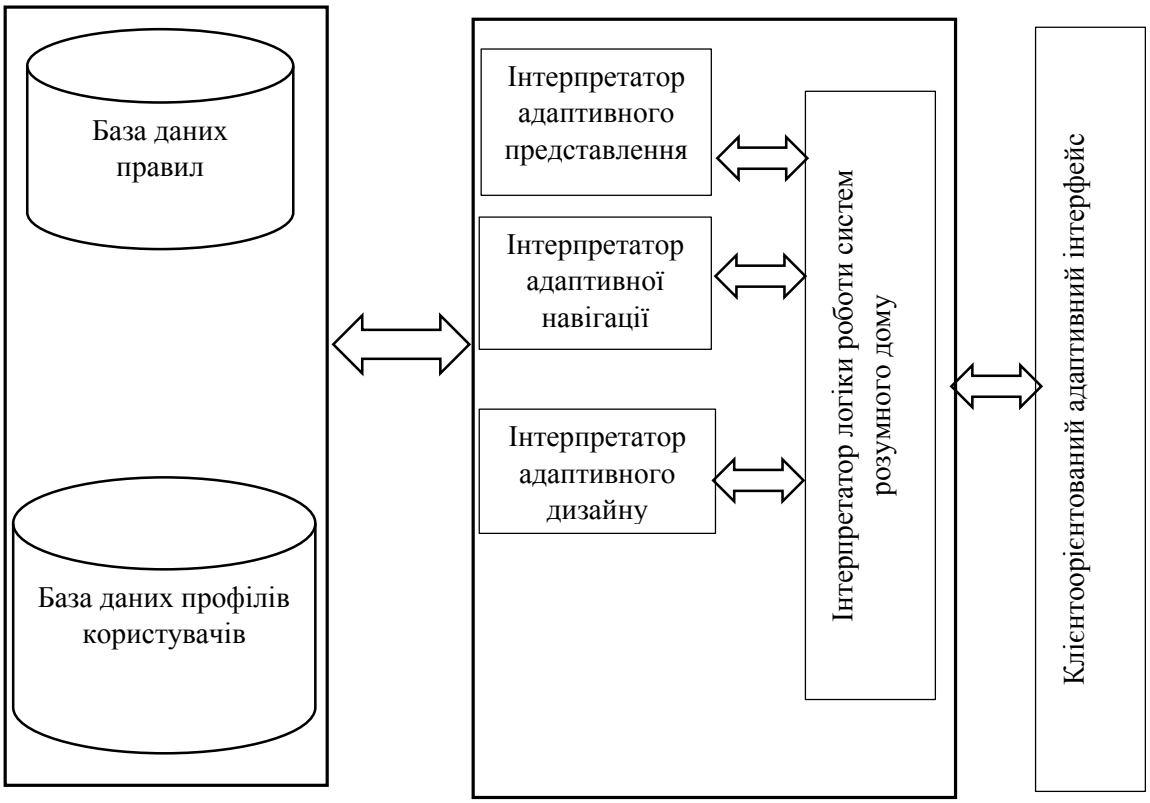
Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доцент кафедри ПЗ_____О. О. Коваленко

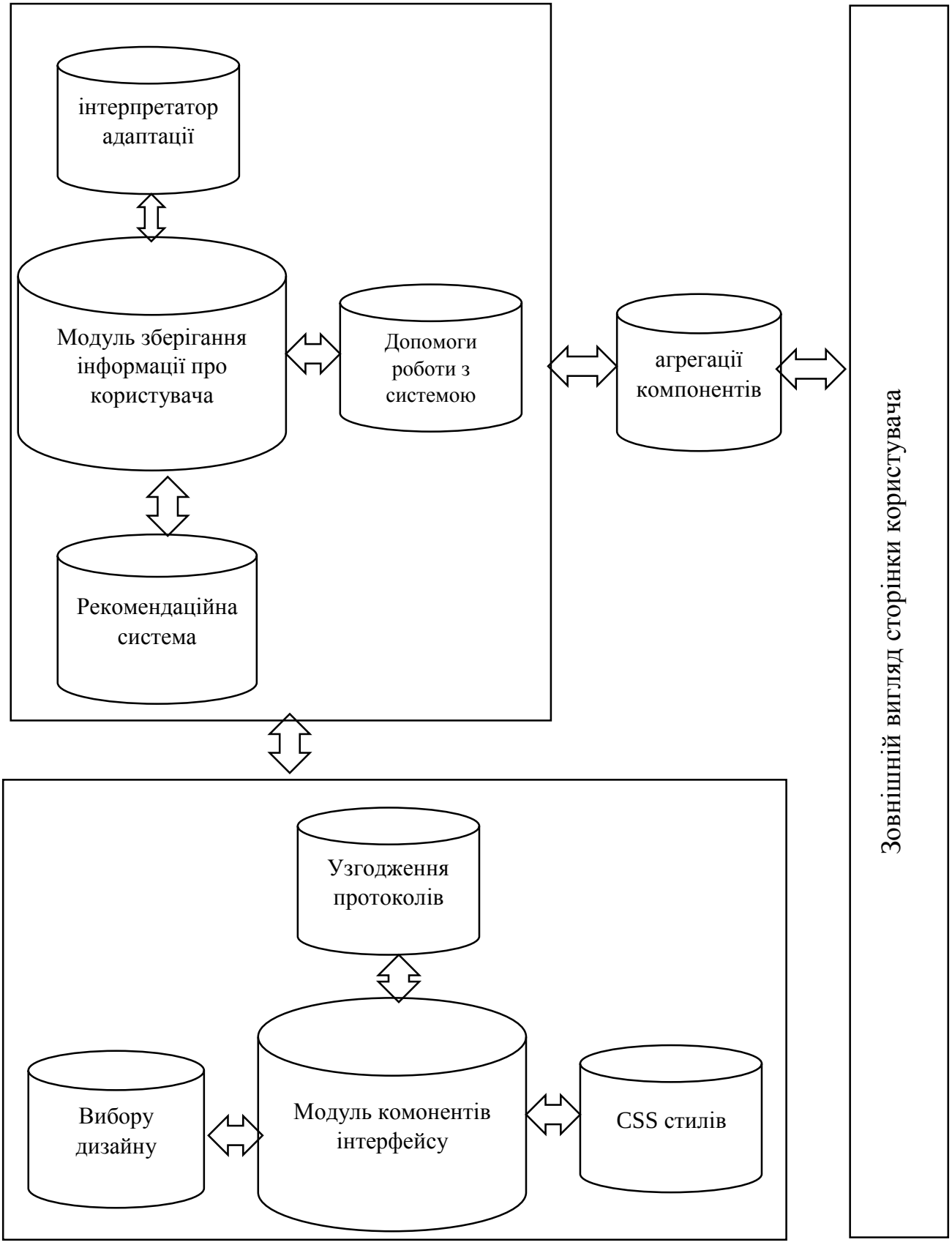
Рецензент, к. т. н., професор кафедри КН _____ І. Р. Арсенюк

Нормоконтроль, к. т. н., доцент кафедри ПЗ _____О. О. Коваленко

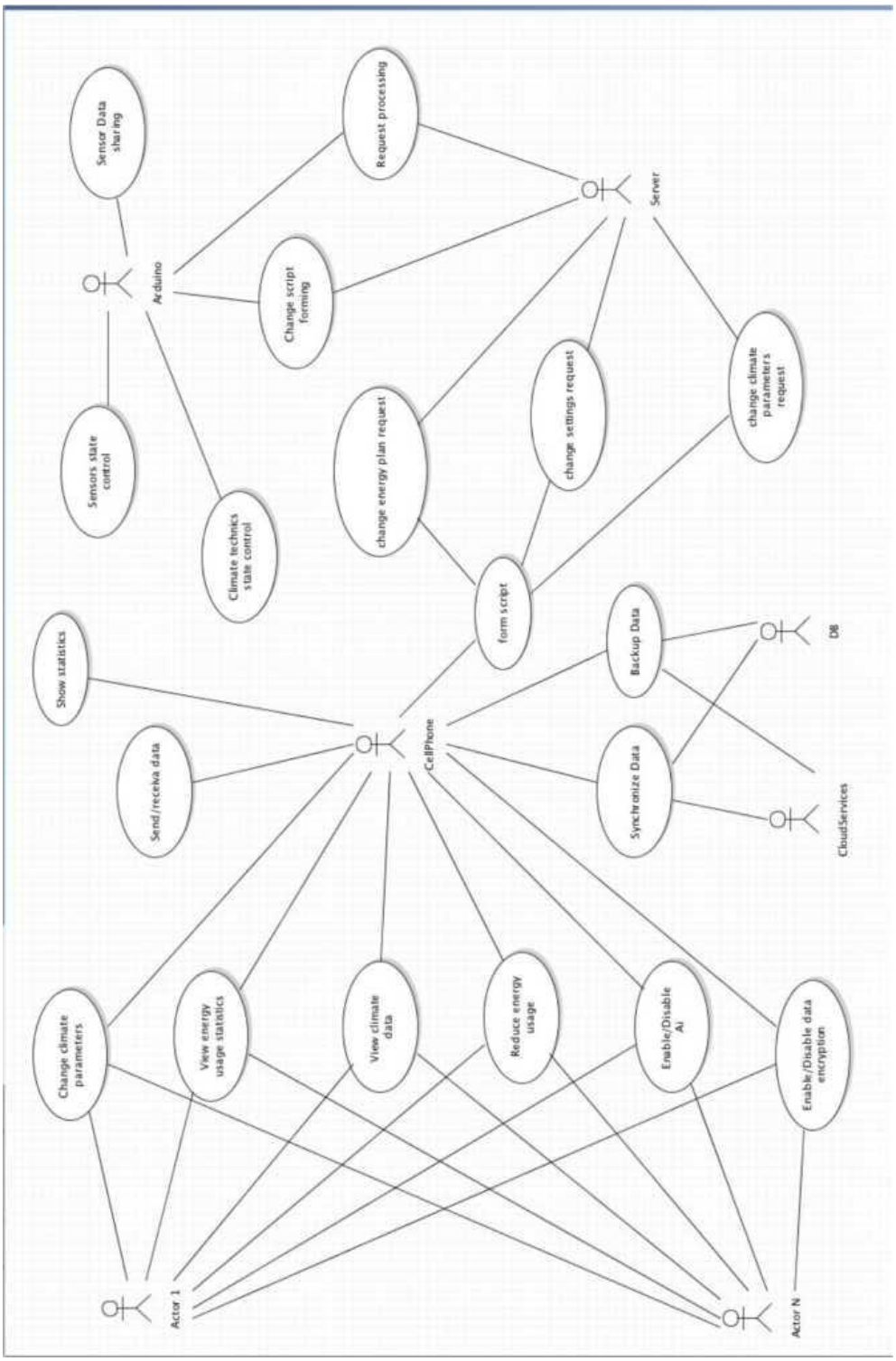
Виконавець, студент групи 1ПІ-18м_____Р.І. Підгородецький



Удосконалена модель клієнтоорієнтованого адаптивного інтерфейсу користувача системи розумного дому



Деталізована модель формування адаптивного інтерфейсу управління системою



Use-case діаграмма системи Smart Home

ЗАГОЛОВОК	
КОМАНДИ ТА ПОВІДОМЛЕННЯ	
ОСНОВНЕ ПОЛЕ ПРЕДСТАВЛЕННЯ РЕЗУЛЬТАТІВ	ПРАПОРЦІ
ВИХІД	

Приклад розподілу екрану термінала на зони

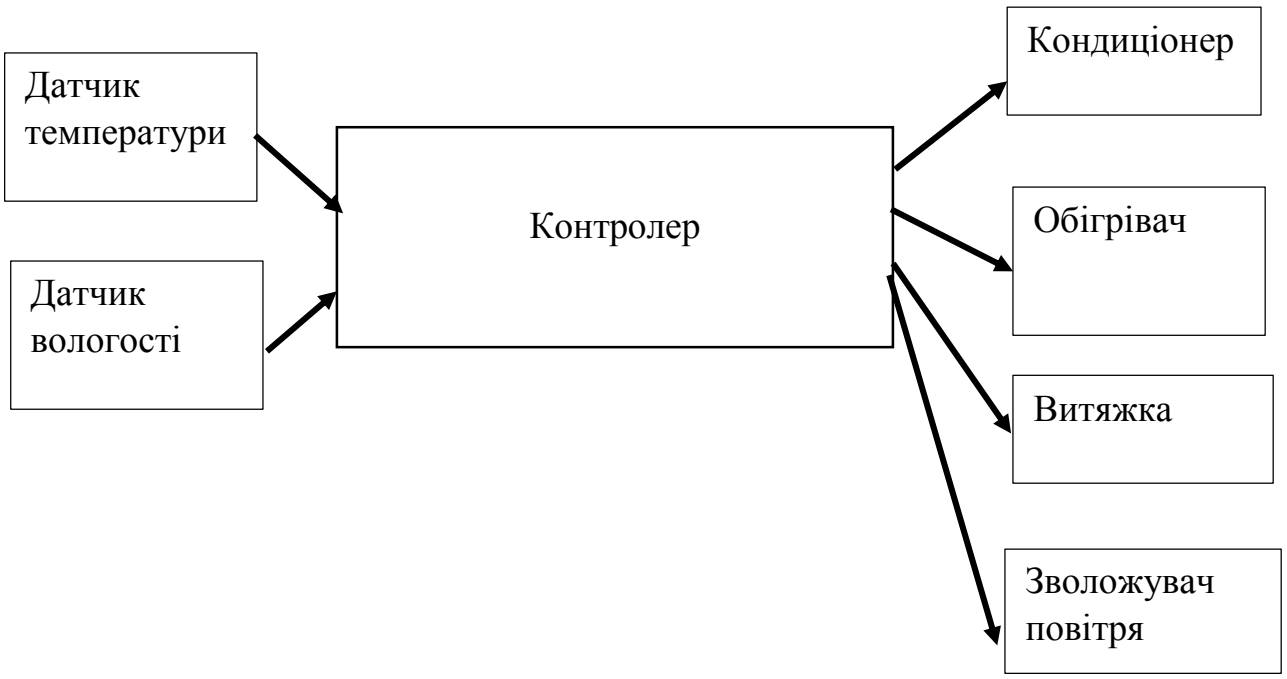
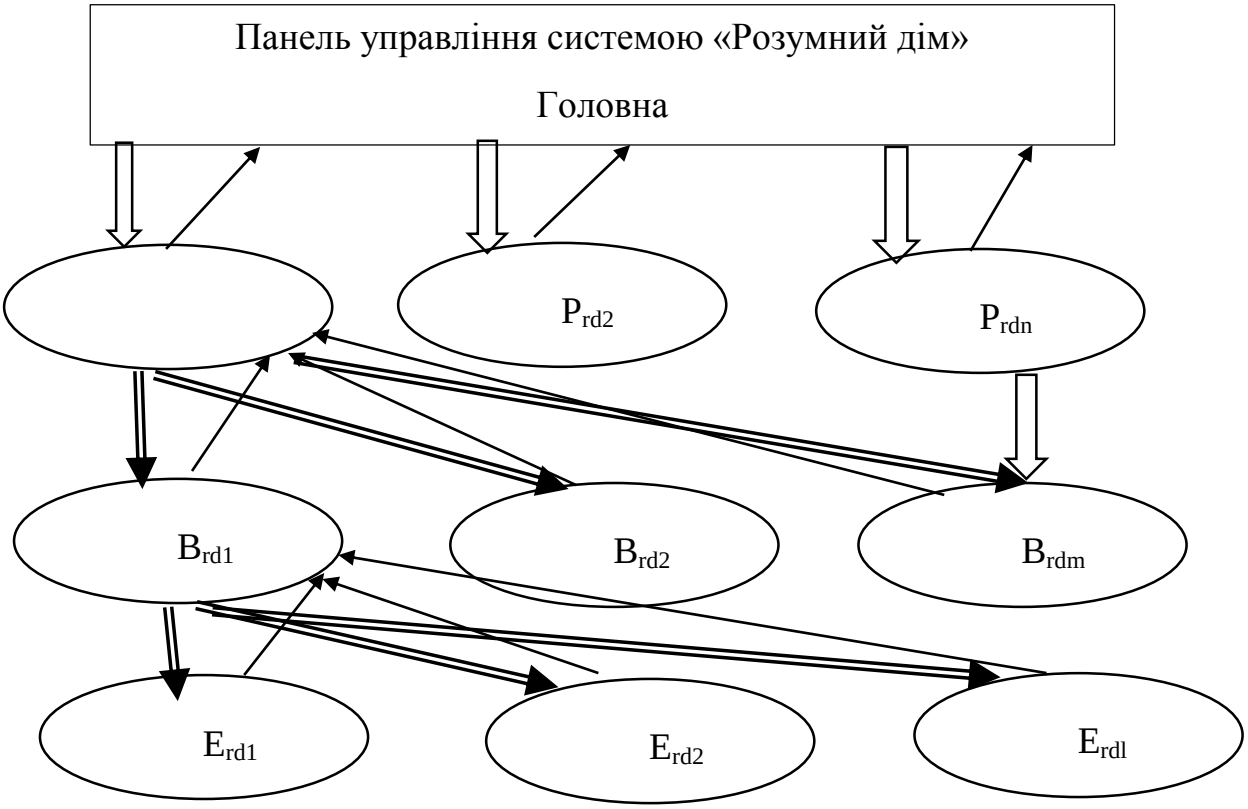


Схема системи регулювання клімату



Змішана система проектування та представлення інформаційних панелей системи розмуного дому