НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

"___"_____2015 р.

Дипломна робота

першого (бакалаврського)**рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності7.050102, 8.050102 Інформаційні технології проектування
7.050103, 8.050103 Системне проектування
(код та назва спеціальності)

на тему:«Системауправліннярозумнимдомом.
Програмнезабезпеченняцентральногоконтролера» _____

Виконав (-ла): студент (-ка) 4 курсу, групи ДА-12
(шифр групи)

Коренев Антон Анатолійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. викладач Британов О.А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Охорона праці доцент, к.б.н., Гусєв А.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензентдоцент, к.т.н., Тимошенко Ю.О. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль ст. викладач Британов О.А. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2015 року

Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут)ННК "Інститут прикладного системного аналізу"
(повна назва)

КафедраСистемного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський)
(перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 7.050102, 8.050102 Інформаційні технології проектування
7.050103, 8.050103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2015 р.

ЗАВДАННЯ
на дипломну роботу студенту
Кореневу Антону Анатолійовичу
(прізвище, ім'я, по батькові)

1. ТемароботиСистемауправліннярозумнимдомом. Програмнезабезпеченняцентральногоконтролера.
керівник роботиБритов Олексій Анатолійович., ст. викладач ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «02» квітня 2015 р. №30/1-ст

2. Строк подання студентом роботи08.06.2015

3. Вихідні дані до роботи _____

Операційна система Android OS
Тактова частота процесору 1.3 GHz
Ядра ЦПУ 2-8 ARMз типом взаємдії BIG.little
Форма реалізації – у вигляді двох Android додатків: серверу та клієнту.
Для отримання метеоданих використовується OpenWeatherMap.
Для зберігання резервних копій даних використовується сервіс Dropbox.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1. Провести аналіз реалізацій технології "Розумний будинок".
2. Дослідити особливості цільової операційної системи Android.

3. Розробити програмне забезпечення центрального контролера.
4. Розробити програмне забезпечення мобільного клієнту.
5. Виконати тестування роботи системи у комплексі.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

1. Блок-схема алгоритму роботи системи – плакат.
2. Алгоритм роботи DropboxAPI.
3. Знимки екранів додатку - плакат.
4. UML-діаграма пакету SHome.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Гусєв А.М.., доцент		

7. Дата видачі завдання 01.02.2015

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2015	
2	Збір інформації	15.02.2015	
3	Аналіз реалізацій технології "Розумний будинок".	28.02.2015	
4	Аналіз цільової операційної системи Android.	02.03.2015	
5	Розробка програмного забезпечення мобільного клієнта	10.03.2015	
6	Тестування роботи моделі	30.04.2015	
7	Оформлення дипломної роботи	31.05.2015	
8	Отримання допуску до захисту та подача роботи в ДЕК	08.06.2015	

Студент _____ А.А. Коренев
(підпис) (ініціали, прізвище)

Керівник роботи _____ О.А. Британов
(підпис) (ініціали, прізвище)

*Консультантом не може бути зазначено керівника дипломного проекту (роботи).

АНОТАЦІЯ

Бакалаврської дипломної роботи Коренев Антон Анатолійович
на тему: "Система управліннярозумним домом. Програмнезабезпечення центрального контролера."

Дипломна робота присвячена розробці програмного забезпечення центрального контролера для реалізації концепції "розумного" будинку. Тема роботи є актуально у зв’язку зі здороженням ціни на енергоносії та складною економічною ситуацією у нашій країні.

Цілю дипломної роботи є розробити програмне забезпечення для серверної частини системи, обрати необхідні програмні засоби для реалізиції концепції та створити додаток для управління центральним контролером через мобільний пристрій.

В роботі проведено аналіз існуючих реалізацій системи “Розумний будинок” та цільової операційної системи Android. Був розроблен власний варіант реалізації концепції.

В ході виконання дипломної роботи було побудовано робочу модель системи, а такожтестовий стенд з мікроконтролером, сервером та мобільним клієнтом.

В результаті роботи було побудовано робочу модель розумного дому з елементами штучного інтелекту.

Загальний обсяг роботи – 82сторінки, 33 рисунка,5 таблиць, 27бібліографічних найменувань.

Ключові слова: Arduino, Android, "розумний" будинок, Dropbox, нейронна мережа.

АННОТАЦИЯ

Бакалаврской дипломной работыКоренева Антона Анатольевича

на тему: "Система управления умным домом. Программное обеспечение центрального контроллера"

Дипломная работа посвящена разработке программного обеспечения центрального контроллера для реализации концепции "умного" дома. Тема работы является актуальной в связи с подорожанием цены на энергоносители и сложной экономической ситуацией в нашей стране.

Целью работы является разработать программное обеспечение для серверной части системы, выбрать необходимые программные средства для реализиции концепции и создать приложение для управления центральным контроллером через мобильное устройство.

В работе проведен анализ существующих реализаций системы "Умный дом" и целевой операционной системы Android. Был разработан собственный вариант реализации концепции.

В ходе выполнения дипломной работы была построена рабочая модель системы, а также тестовый стенд с микроконтроллером, сервером и мобильным клиентом.

В результате работы было построено рабочую модель умного дома с элементами искусственного интеллекта.

Общий объем работы - 82 страницы, 33 рисунка, 5 таблиц, 27 библиографических наименований.

Ключевые слова: Arduino, Android, "умный" дом, Dropbox, нейронная сеть.

ANNOTATION

For the bachelor's degree work of Korenev Anton Anatolievich
on "smart home control system. The software of central controller"

Thesis is devoted to the software development of the central controller to implement the concept of "smart" home. R & D is relevant in connection with rising energy prices and the difficult economic situation in our country.

The aim is to develop software for the server part of the system, select the required software for realizitsii concept and create an application to control the central controller via a mobile device.

The analysis of existing implementations of "smart house" and the target operating system Android. It was developed its own version of the concept.

In the course of the thesis was to build a working model of the system, as well as a test stand with a microcontroller, the server and the mobile client.

As a result, the work was to build a working model of a smart home with elements of artificial intelligence.

The total amount of work - 82 pages, 33 drawings, 5 tables, 27 bibliographical references.

Tags: Arduino, Android, "smart" house, Dropbox, neural network.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API	Application programming interface, прикладний програмний Інтерфейс
GPL	General public license, общая свободная лицензия
GUI	Graphical user interface, графічний інтерфейс користувача
IDE	Integrated development environement, вбудована система розробки
SDK	Software development kit, набір розробника програмного забезпечення
ІТ	Інформаційні технології
ОС	Операційна система
ПЗ	Програмне забезпечення
ПК	Персональний комп’ютер
БД	База даних
Віджет	Графічний засіб відображення інформації на екрані персонального комп’ютеру чи смартфону
Гаджет	Обчислювальний пристрій. Частіше за все смартфон чи планшет

ВСТУП

Світ технологій розвивається безперервно і динамічно. З кожним роком з'являються нові концепції та їх реалізації, а існуючі розширюють свої функціональні характеристики і можливості. Не нова і мета цієї спіралі прогресу - зробити умови життя людини якомога комфортнішими і продуктивними. Однією з причин такого бурхливого зростання є широке поширення Internet і значне збільшення швидкості передачі даних. Як зазначають в Google, до 2008 року був «Інтернет людей», тепер настав «Інтернет речей», так як пристроїв, підключених до світової мережі стало більше ніж жителів планети. Завдяки цьому взаємодію їх з людиною стало можливо винести на зовсім інший рівень.

Концепція «Розумний» будинок вперше з'явилася в 1961 році з появою пристрою для регулювання інтенсивності освітлення - діммера і з тих пір зазнала значних змін. Безумовно, на тому етапі називати систему «Розумним» будинком було занадто рано, проте думки більшості сходяться на тому, що саме це відкриття стало початком ери розумної домашньої електроніки.

Говорячи про технологію «Розумний» будинок та її новизну, варто зазначити скоріше не інноваційність, а перехід кількості в якість. На початку 2000х існувало більше десятка протоколів синхронізації з невеликою кількістю підтримуваних пристроїв і жорсткою прив'язкою до одного виробника, однак ситуація змінилася. У наш час практично будь-який пристрій може працювати з будь-якою платформою завдяки синхронізації через віддалені сервіси або другий варіант - він поєднує в собі функції всіх інших і просто підключається в домашню мережу.

Вже зараз багато відомих корпорацій, орієнтованих на споживчий ринок, розширюють можливості своїх систем за рахунок тісної інтеграції пристроїв і сервісів в екосистеми розумних будинків. Прикладом служать американські Apple і Google з фреймворком HomeKit, або китайська Meizu з додатком-

інтегратором Home. Зацікавленість великих корпорацій в системі «Розумний» будинок тільки підтверджує актуальність теми.

Суть концепції полягає у створенні автономної екосистеми з домашньої електроніки, що підвищує комфорт проживання і виводить його на якісно новий рівень. Вона не продиктована життєвою необхідністю, проте як і багато девайсів 21го століття економить час користувачів і підвищує продуктивність їх праці та відпочинку.

Не можна не визнати, що потенціал технології величезний і обмежується тільки відносно високою вартістю готової системи. Проте, в 2010 році була представлена перша плата для домашньої автоматизації - Arduino Mega, яка дозволяла налаштувати взаємозв'язок сенсорів з контролерами на С-образній мові скетчів без потреби глибоко вникати в саму суть процесів функціонування модулів. Саме наявність великої кількості готових бібліотек для найрізноманітнішої електроніки поряд з невисокою ціною дозволило зробити «Розумний» будинок загальнодоступним.

Однак для користувачів, які незнайомі з програмуванням і не мають хоча б базових знань по роботі з настроюваними мікроконтроллерами, розгортання такої системи так і залишилося нетривіальним завданням. Мета дипломного проекту - надати в широкий доступ власну реалізацію такої системи та інструменти графічного налаштування і керування, щоб знизити поріг входження для нових користувачів без технічної освіти. Друга мета - створити єдину функціонуючу екосистему, яку можна буде вільно видозмінювати і переробляти будь-кому при бажанні. Тому проект буде опублікований під вільною ліцензією GNU GPL.

1 ТЕХНОЛОГІЯ «РОЗУМНИЙ» БУДИНОК

Розумний будинок - житловий будинок сучасного типу, організований для проживання людей за допомогою автоматизації і високотехнологічних пристроїв [1]. Під «Розумним» будинком слід розуміти систему, яка забезпечує безпеку та ресурсозбереження (в тому числі і комфорт) для всіх користувачів. У найпростішому випадку вона повинна вміти розпізнавати конкретні ситуації, що відбуваються в будинку, і відповідним чином на них реагувати: одна з систем може управляти поведінкою інших по заздалегідь виробленим алгоритмам. Крім того, від автоматизації декількох підсистем забезпечується синергетичний ефект для всього комплексу.

Зі збільшенням обчислювальної здатності гаджетів концепція «Розумний» будинок отримала своє логічне продовження – систему «Інтернет речей», згідно з якою була проведена первинна стандартізація та визначені основні правила та рекомендації до побудови готового продукту на рівні як системи загалом, так і окремих компонентів. Незважаючи на відносну новизну, вже зараз існує декілька десятків різних рішень.

1.1 Історія технології «Розумний» будинок

Історія «Розумного» будинку почалася в 1961 році, коли Джоель і Рут Спіра винайшли і запатентували спеціальний пристрій для плавного регулювання світла – диммер[1]. Саме цей винахід стало приводом для створення всесвітньо відомої сьогодні компанії Lutron Electronics Company, Inc. Дана фірма продовжувала працювати над «розумними» технологіями, паралельно впроваджуючи в побут такі поняття, як світлова зона і сцена.

Значною подією у подальшому розвитку технології «розумного будинку» було створення шведською компанією Pico Electronics побутової автоматики в 1975 році, яку вперше почали використовувати для управління музичними

програвачами. Удосконалили домашню автоматику американці Скотт і Росслин Міллер.

Першим повноцінним проектом «Розумного» будинку став невеликий житловий будинок на південному березі Англії. В основу його автоматики лягло використання широкосмугового KNX - системи, що відповідає за управління освітленням, сигналізацією, жалюзі, опаленням та дверима гаража. Також в даному будинку був створений басейн, який згодом доповнили LED-системою з оригінальними кольоровими ефектами.

У 1987 році організація ASHRAE створила новий протокол домашньої автоматизації, який надалі був вдосконалений групою компаній Berker, Merten, Insta, Gira, Jung і Siemens і перетворений в абсолютно нову модель автоматики European Installation Bus. У 1999 році на її основі було розроблено нове покоління польових шин KNX, які досі вважаються кращим стандартом європейських систем домашньої автоматизації.

Сучасні системи пішли далеко вперед, істотно розширивши свої технічні можливості. Сьогодні в них використовуються вбудовані домашні кінотеатри, об'єднуються всі інженерні системи, застосовується інтелектуальне управління на основі спеціального ПЗ. Завдяки модульності системи у користувачів з'явилася можливість самостійно вибирати функціонал «Розумного» будинку [1].

У 2008-2009 роках була сформована концепція «Інтернет речей», яка стала логічним продовженням переходу до модульної архітектури та зробила тісну інтеграцію будь-яких підтрімуванних приладів простою як ніколи. Інтернет речей - це не просто безліч різних приладів і датчиків, об'єднаних між собою дротяними і бездротовими каналами зв'язку і підключених до мережі Інтернет, а це більш тісна інтеграція реального та віртуального світів, в якому спілкування здійснюється між людьми і пристроями.

У 2014 році компанія Apple зробила анонс нового framework під назвою HomeKit, який створений для інтеграції екосистеми «Розумного» будинку з пристроями на iOS та автомобілями з системою Apple CarPlay чи Google Car.

1.2 Принцип роботи системи

Термін “Розумний” будинок не має чіткого визначення, а тому під нього підпадає будь-яка система з автоматизованим керуванням приладами, яка спрощує життя людини та підвищує рівень його комфорту. Через нечіткі рамки, виникла багато реалізацій з різним рівнем інтеграції та принципом роботи. Їх можна умовно поділити на три группи:

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

Перша група є повністю налаштованою і встановленою виробником системою, яка управляється центральним обчислювальним пристроєм і не передбачає прямої взаємодії своїх компонентів між собою. Всі призначені для користувача налаштування зберігаються на центральному пристрої (сервері), а периферія лише виконує отримані від нього інструкції і часто не має вбудованої пам'яті і обчислювальних потужностей. Принцип роботи системи зображений на рис. 1.1.

Друга група є системою з напівавтономних пристроїв. Алгоритми взаємодії прописуються з програми контролера безпосередньо в пам'ять кожного пристрою і для їх зміни пристрій буде необхідно перепрограмувати. У зв'язку з відсутністю центрального компонента, зв'язки між приладами встановлюються безпосередньо і є можливість створення автономних груп, замкнутих один на одного. Принцип роботи наведено на рис. 1.2.

Рисунок 1.1 – Діаграма, що представляє принцип роботи вбудованної системи з центральним контроллером

Друга група є системою з напівавтономних пристроїв. Алгоритми взаємодії прописуються з програми контролера безпосередньо в пам'ять кожного пристрою і для їх зміни пристрій буде необхідно перепрограмувати. У зв'язку з відсутністю центрального компонента, зв'язки між приладами встановлюються безпосередньо і є можливість створення автономних груп, замкнутих один на одного. Принцип роботи наведено на рис. 1.2.

Рисунок 1.2 – Діаграма, що представляє принцип роботи вбудованної системи без центрального контроллера

Третя група - це зовнішні контролери, які приєднуються до звичайних приладів і залежно від показань своїх сенсорів і вбудованого алгоритму регулюють його роботу. Можуть мати центральний контролер, але часто

управляються і налаштовуються з інтернет- або хмарного серісу. Функціонують здебільшого як незалежні модулі і для налаштування прямого зв'язку можуть знадобитися додаткові датчики / сенсори. Принцип роботи наведено на рис. 1.3.

Рисунок 1.3 – Діаграма, що представляє принцип роботи системи з настроюваної інтеграцією

1.3 Основні характеристики системи

Основне призначення системи «Розумний» будинок - підвищення комфорту за рахунок збільшення рівня автоматизації рутинних процесів. Тобто, в першу чергу система повинна бути зручною у використанні і вимагати найменьшу кількість маніпуляцій з боку користувача. Для досягнення цієї мети «Розумний» будинок визначається такими основними параметрами [2]:

- Взаємодія. Особливість системи розумного будинку полягає в її здатності об'єднувати різні пристрої в єдину систему. Злагоджена робота девайсів може бути організована просто чи складно залежно від «відкритості» системи автоматизації. Найбільш відкритою системою вважається та, де взаємодія пристроїв проходить максимально легко. Для підтримки взаємодії між

декількома електронними пристроями виробники систем розумний будинок дуже часто укладають партнерські відносини між собою Це дозволяє більш серйозно підійти до питання інтеграції всіх систем будинку: від архітектурного освітлення та поливу газону до забезпечення роботи домашнього кінатеатра. Ще один спосіб взаємодії - це робота на основі технологічних стандартів. Багато виробників впроваджують у свої продукти бездротове управління на базі технології Z-Wave. Саме цей загальний елемент дозволяє пристроям злагоджено працювати один з одним. Чим більше у провайдера партнерів, тим ширше буде асортимент у клієнта;

- Віддалений доступ. Користувачам системи потрібна можливість швидко і легко змінювати налаштування, якщо це терміново необхідно. Дуже часто зробити це потрібно, коли клієнт знаходиться не вдома. Саме тому однією з найбільш затребуваних особливостей системи розумний будинок є можливість віддаленого управління і доступу до системи. Вона дозволяє контролювати що відбувається в будинку і навколо нього, змінювати налаштування освітлення, термостатів та іншого обладнання за допомогою ноутбука, смартфона або планшета. Віддалений доступ також дозволяє установникові налаштувати систему без необхідності його присутності в будинку, що підвищує зручність і рівень сервісу;

- Масштабованість. Необхідність цього параметра визначається тим, що технології постійно розвиваються, виплескуючи на ринок товари нового покоління. У майбутньому це дає можливість додати нові приміщення до системи без необхідності купувати нову. Також масштабованість дає можливість користувачеві самому визначати необхідні сенсори і функції системи при цьому не позбавляючи можливість додати їх у майбутньому. З цих та інших причин дуже важливо, щоб в «Розумному» будинку можна було додавати нові функції і пристрої (вертикальне розширення) або нові приміщення (горизонтальне розширення). Виробники часто поддержіваютоба типу розширення за допомогою розробки системи на одній мові мережі,

наприклад IP (Internet Protocol), а також можливістю бездротового дооснащення продуктами, які можуть взаємодіяти за допомогою існуючої домашньої мережі або провідних пристроїв.

Наведені параметри є основою концепції «Розумний» будинок, при цьому повна або часткова реалізація їх лежить на виробниках і може залежати від призначення системи та специфіки регіону установки. Обмежень реалізації жорсткими рамками немає, стандартизація стосується тільки протоколів взаємодії пристроїв між собою. Сама реалізація залишається на розсуд компаній і не стандартизується.

1.4 Існуючі концепції

В ідеальному варіанті «Розумний» будинок - це система, в якій кожен прилад інтегрований в загальну екосистему, проте зважаючи на відмінності різних протоколів зв'язку приладів, необхідності у здешевленні системи виробником та налаштування під конкретні завдання, відбувається поділ на три основні групи за призначенням [3]:

- Мультимедійний простір;
- Система розумного контролю мікрокліматичних параметрів приміщення;
- Змішана система.

Мультимедійний простір або мультимедійна система «Розумний будинок» - це цілісна екосистема з мультимедійних пристроїв із загальним хабом пам'яті для контенту і розширеними можливостями взаємодії.

Ця концепція заснована на переважаючій функції контенту і складається з таких компонентів як Smart TV з доступом до інтернет ресурсів і додатків, мультимедіа хаб, файловий хаб, система об'ємного звуку, функцій енергозбереження для техніки, віддалений контроль пристроїв. З появою готових пристроїв Smart TV і хмарних сервісів концепція може бути легко

реалізована будь-яким користувачем без необхідності додаткового придбання будь-чого окрім самих пристроїв. Є найбільш популярною в США і Канаді.

З появою доповненої реальності концепція мудьтімедійного розумного будинку знайшла нове життя у вигляді так званих шоломів віртуальної реальності. До основних функцій було додано модуляцію віртуальних об'єктів на реальність і можливість взаємодіяти з результатом як з цілісним середовищем. Варто відзначити, що тепер для побудови системи потрібно всього лише один пристрій без додаткових аксесуарів і функціональність системи залежить тільки від вбудованого в шолом ПЗ [4]. Приклад мультимедійної системи розумного будинку з доповненою реальністю на рис. 1.4.

Система розумного контролю мікрокліматичних параметрів приміщення - це система, заснована навколо ідеї створення найбільш оптимальних параметрів мікроклімату з найнижчими енерговитратами [5]. У даному випадку під контроль автоматики передаються кліматичні пристрої, а також джерела світла і, опціонально - інша побутова електроніка. Сама концепція системи вибудовується навколо об'єднання перерахованої техніки в єдину настроювану екосістему.

Рисунок 1.5 – приклад роботи Microsoft HoloLens [6]

По суті, система представляє з себе набір керованих алгоритмів, які запускаються при настанні стартової умови. Наприклад, при недостатній освітленості загоряються додаткові світильники, або система опалення працює на мінімумі поки користувач не вдома і тд.

Змішана система - це система, яка має функціонал обох попередніх. Він може бути реалізований як у повній мірі, так і частково. В основному будується навколо центрального контролюючого пристрою, щоб була можливість управління різними типами пристроїв за різними протоколами.

1.5 Переваги та недоліки існуючих рішень

«Розумний» будинок від Meizu [7] - це сукупність різних смарт-девайсів від різних компаній, які об'єднані єдиним софтом (LifeKit) та вимогами Meizu. На сьогодні в «розумну» екосистему потрапили такі смарт-девайси, як ваги RyFit (32 $), лампа X-Light Plus (19 $), розетки, очищувач повітря Air Cube та інше. Система відноситься до змішаного типу. Переваги: наявність готових пристроїв, простота установки і налаштування, функції керуючого пристрою бере на себе телефон. Недоліки: порівняно малий функціонал, обмежений вибір устаткування, прив'язка до пристроїв від Meizu, сумарна дорожнеча системи.

Рішення від Allone або Allone WiFi Smart Remote Control [8] представляє центральний контролер, керуючий підтримуваними приладами через wi-fi. Ціна 48 $. Переваги: велика кількість модулів, масштабованість. Недоліки: порівняно висока ціна модулів, не підходить для великих будинків з товстими стінами через використання wi-fi для управління пристроями.

Рішення від Clipsal. Являє собою невеликі модулі з сенсором, які підключаються до просто електроніці і управляють їй за рахунок вбудованого алгоритму. Переваги: низька вартість, масштабованість системи, немає потреби купувати техніку з вбудованими функціями взаємодії. Недоліки: відносно складне налаштування, низька взаємодія компонентів один з одним.

Порівняння рішень наведено у таблиці 1.1.

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

Параметр	Meizu	Allone	Clipsal
Вартість	Середня	Висока (разом з модулями)	Невелика за модуль, середня за систему в цілому
Установка	Проста	Проста	Потребує попереднього налаштування
Налаштування	Не потребує	Через веб-сервіс	Через прошивку
Готові модулі	Мало	Багато	Не потребує
Масштабованість	Масштабується	Масштабується, але залежить від розміру приміщення	Масштабується
Взаємодія компонентів	Через смартфон	Через сам пристрій	Через хмарний додаток
Функціонал	Базовий	Майже необмеженний	Майже необмеженний

1.6 Власний варіант реалізації

Варіант реалізації від SHome - це система розумного контролю мікрокліматичних параметрів з вбудованими функціями енергозбереження та прогнозу енергоспоживання. Система відноситься до класу з вбудованим центральним контролером. Архітектура модульного типу. Розширюваність

забезпечується за рахунок сумісних с Arduino модулів, однак для підхоплення нових сенсорів потрібно встановити новий скетч на Arduino через скріптогенератор. Система складається з трьох частин: модулі управління кліматотехніки на основі Arduino, центральний контролер і мобільний додаток для управління системою та відображення інформації у візуально-зрозумілому вигляді. Відмінність від інших рішень полягає у використанні тільки вільних компонентів під вільною ліцензією як у випадку з апаратним забезпеченням, так і з програмним. Абсолютно будь-який користувач може налаштувати систему під себе, використовучи недорогі компоненти для Arduino і малопотужний пристрій в якості центрального контролера, не написавши при цьому жодного рядка коду.

Основна мета проекту - надати користувачам недорогу систему на безоплатній основі з розширеними можливостями в налаштуванні енергетичних планів і кастомізації.Необхідне для розгортання реалізації від SHome на рис. 1.6.

Рисунок 1.6 – Базові складові для розгортання системи SHome

1.7 Висновки

Отже, «Розумний» будинок - житловий будинок сучасного типу, організований для проживання людей за допомогою автоматизації і високотехнологічних пристроїв.

За принципом побудови виділяють:

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

За концепцією:

- Мультимедійний простір;
- Система розумного контролю мікрокліматичних параметрів приміщення;
- Змішана система.

Основні характеристики:

- Взаємодія;
- Масштабованість;
- Віддалений доступ.

Основні риси власної системи SHome:

- Енергозбереження;
- Енергопрогнозування;
- Компоненти під вільною ліцензією;
- Кастомізація.

2 ОСОБЛИВОСТІ ЦІЛЬОВОЇ ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID

2.1 Загальна інформація про ОС Android

Android ОС - операційна система для смартфонів, планшетних комп'ютерів, електронних книг, цифрових програвачів, наручних годинників, ігрових приставок, нетбуків, смартбуков, окулярів Google, телевізорів та інших пристроїв [9]. В майбутньому планується підтримка автомобілів і побутових роботів. Заснована на ядрі Linux і власної реалізації віртуальної машини Java від Google. Спочатку розроблялася компанією Android Inc., яку потім купила Google. Згодом Google ініціювала створення альянсу Open Handset Alliance (OHA), який зараз займається підтримкою і подальшим розвитком платформи. Android дозволяє створювати Java-додатка, керують пристроєм через розроблені Google бібліотеки. Android Native Development Kit дозволяє портувати (але не налагоджувати) бібліотеки та компоненти додатків, написані на Сі та інших мовах.

У 86% смартфонів, проданих у другому кварталі 2014 року, була встановлена операційна система Android. При цьому за весь 2014 було продано більше 1 мільярда Android-пристроїв.

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

23 вересня 2008 офіційно вийшла перша версія операційної системи, а також перший повноцінний пакет розробника SDK 1.0, Release 1. З моменту виходу першої версії платформи сталося кілька оновлень системи. Ці

оновлення, як правило, стосуються виправлення виявлених помилок і додавання нової функціональності в систему.

У 2009 році було представлено цілих чотири оновлення платформи. Так, в лютому вийшла версія 1.1 з виправленням різних помилок. У квітні та вересні вийшли ще два оновлення - 1.5 «Cupcake» і 1.6 «Donut» відповідно. Оновлення «Cupcake» привнесло істотні зміни: віртуальна клавіатура, відтворення і запис відео, браузер і інші. У «Donut» вперше з'явилися підтримка різних дозволів і щільності екрану і мереж CDMA. У жовтні того ж року вийшла версія операційної системи Android 2.0 «Eclair» з підтримкою декількох акаунтів Google, підтримкою браузером мови HTML5 та інших нововведень, а також після невеликого оновлення в межах версії «Eclair» (2.1) з'явилися «живі шпалери» і був видозмінений екран блокування.

У середині 2010 Google представила Android версії 2.2 під найменуванням «Froyo», а в кінці 2010 року - Android 2.3 «Gingerbread». Після оновлення «Froyo» стало можливо використовувати смартфон в якості точки доступу, використовувати традиційну систему блокування смартфона цифровим або буквено-цифровим паролем та інші зміни, а оновлення «Gingerbread» привнесло більш повний контроль над функцією копіювання і вставки, поліпшення управління живленням і контролю над додатками, підтримку декількох камер на пристрої і т. д. 22 лютого 2011 року була офіційно представлена орієнтована на інтернет-планшети платформа Android 3.0 «Honeycomb».

Android 4.0 «Ice Cream Sandwich», що вийшла 19 жовтня 2011, - перша універсальна платформа, яка призначена як для планшетів, так і для смартфонів. Також оновлення привнесли новий інтерфейс «Holo», який використовувався до Android 4.4.4 Kitkat, на даний момент замінений на Material Design.

У червні 2012 вийшло оновлення під назвою «Jelly Bean» з порядковим номером 4.1, який змінився на 4.2 внаслідок невеликого оновлення в кінці

жовтня того ж року і на 4.3 після оновлення в липні 2013. 31 жовтня 2013 Google представила наступну версію операційної системи Android 4.4, яка отримала назву шоколадного батончика «KitKat» за угодою з компанією виробником Nestlé . Вперше KitKat з'явився на Nexus 5; ця версія Android оптимізована для роботи на більш широкому наборі пристроїв, що мають 512 МБ ОЗУ як рекомендований мінімум. 25 червня 2014 Google представили Android L, зараз доступний для розробників.

У 2014 році була анонсована операційна система для носимих пристроїв Android Wear. Також на Google I / O були представлені версії Android Auto (для автомобілів) і Android TV (для телевізорів), тим самим Android перестав бути операційною системою тільки для мобільних пристроїв. 15 жовтня 2014 була офіційно анонсована Android 5.0 Lollipop. Головне оновлення системи - новий дизайн Material design. Також, якщо на Android-пристрої встановлений пароль або графічний ключ, і якщо поблизу знаходяться годинник господаря пристрої з Android Wear, то пристрій автоматично розблокується. 9 грудня 2014 Google замінила офіційну середовище розробки, засновану на Eclipse (adt-bundle) на Android Studio.

Додатки під операційну систему Android є програмами в нестандартному байт-коді для віртуальної машини Dalvik, для них був розроблений формат настановних пакетів APK. Для роботи над додатками доступно безліч бібліотек: Bionic (бібліотека стандартних функцій, несумісна з glibc); мультимедійні бібліотеки на базі PacketVideo OpenCORE (підтримують такі формати, як MPEG-4, H.264, MP3, AAC, AMR, JPEG і PNG); SGL (движок двомірної графіки); OpenGL ES 1.0 ES 2.0 (движок тривимірної графіки); Surface Manager (забезпечує для додатків доступ до 2D / 3D); WebKit (готовий движок для веб-браузера; обробляє HTML, JavaScript); FreeType (движок обробки шрифтів); SQLite (легковага СУБД, доступна для всіх додатків); SSL (протокол, що забезпечує безпечну передачу даних по мережі). У порівнянні зі звичайними додатками Linux додатки Android підкоряються додатковим правилам: Content

Providers - обмін даними між додатками; Resource Manager - доступ до таких ресурсів, як файлиXML, PNG, JPEG; Notification Manager - доступ до рядку стану; Activity Manager - управління активними додатками.

Google пропонує для вільного скачування інструментарій для розробки (Software Development Kit), який призначений для x86-машин під операційними системами Linux, Mac OS X (10.4.8 або вище), Windows XP, Windows Vista і Windows 7. Для розробки потрібно JDK 5 або більш новий.

Розробку додатків для Android можна вести на мові Java (не нижче Java 1.5). Існує плагін для Eclipse - Android Development Tools (ADT), призначений для Eclipse версій 3.3-3.7. Також існує плагін для IntelliJ IDEA, який полегшує розробку Android-додатків, і для середовища розробки NetBeans IDE, який, починаючи з версії NetBeans 7.0, перестав бути експериментальним, хоч поки і не є офіційним. Крім того, існує Motodev Studio for Android - комплексна середовище розробки на базі Eclipse, що дозволяє працювати безпосередньо з Google SDK.

У 2009 році на додаток до ADT був опублікований Android Native Development Kit (NDK) - пакет інструментаріїв і бібліотек, що дозволяє реалізувати частину програми на мові C / C ++. NDK рекомендується використовувати для розробки ділянок коду, критичних до швидкості.

В Android 4.4 з'явилася можливість змінити віртуальну машину Dalvik на ART (Android Runtime). ART відрізняється підвищеною швидкістю завантаження програми. В Android 5.0 вибір машини пропав, тому що замість Dalvik стала використовуватися ART.

21 жовтня 2008 альянс OHA опублікував вихідний код платформи Android на відкритому вихідному коді Android: і операційна система, і проміжне ПЗ (middleware), і основні кінцеві додатки, написані на Java. Загальний обсяг вихідного коду Android склав 2,1 ГБ. «Переважною ліцензією» на вихідний код Android є ліцензія Apache 2.0. Після випуску Android 3.0 «Honeycomb» президент мобільного підрозділу Google Енді Рубін заявив про

те, що відкриття вихідного коду нової версії системи буде відкладено через те, що система була погано готова для запуску на комунікаторах і вимагає значних оптимізацій. Це рішення викликало критичні оцінки аналітиків: наприклад, оглядач ZDNet Крістофер Доусон назвав такий хід Google розчаруванням. Але, згідно з даними компанією обіцянкам, Google відкрила восени 2011 року вихідні коди наступної версії системи - Android 4.0 Ice Cream Sandwich.

Існує спільнота ентузіастів, що розробляє відкриті варіанти прошивок Android - CyanogenMod, MIUI, AOKP (Android Open Kang Project) та інші. Модифіковані версії Android створюються для доповнення операційної системи новими налаштуваннями, опціями, функціями; видалення з Android-пристрої сервісів Google для виключення можливості передачі ідентифікаційної інформації на сервери компанії, наприклад, інформацію про переміщення користувача в реальному часі, що призвело до судового розгляду; більш оперативного і частого (у порівнянні з виробниками самих апаратів) надання нових версій Android. Для перепрошивки Android-пристрої необхідний root-доступ (це називається рутінг, англ. Rooting). При технічній або програмної несправності пристрою, викликаної неправильним використанням root-доступу, в гарантійному ремонті пристрою сервісний центр може відмовити.

Переваги:

• Незважаючи на початковиу заборону на установку програм з «неперевірених джерел» (наприклад, з карти пам'яті), це обмеження відключається штатними засобами в налаштуваннях апарата, що дозволяє встановлювати програми на телефони та планшети без інтернет-підключення (наприклад, користувачам, які не мають Wi -Fi-точки доступу і не бажаючим витрачати гроші на мобільний інтернет, який зазвичай коштує дорого), а також дозволяє всім бажаючим безкоштовно писати програми для Android і тестувати на своєму апараті;

• Android доступний для різних апаратних платформ, таких як ARM, MIPS, x86;

• Існують альтернативні Google Play магазини додатків: Amazon Appstore, Opera Mobile Store, Yandex.Store, GetUpps !, F-Droid;

• У версії 4.3 введен режим декількох користувачів.

Недоліки:

• Наявність в деяких Android-пристроях сервісів Google, що забезпечують можливість передачі ідентифікаційної інформації на сервери компанії, наприклад, інформацію про переміщення користувача в реальному часі;

• У версії Android 1.6 розробники додали Native Development Kit, який дозволяє писати власні низькорівневі модулі для системи на мовах C / C ++, спираючись на стандартні Linux-бібліотеки. Хоча, наприклад, стандартна бібліотека мови Сі на платформі Android, відома як Bionic, якраз не є стандартною і повністю сумісною з libc;

• Для доступу до Google Play та інших сервісів від Google необхідно використовувати пропрієтарні додатки, які виробник телефону має право встановлювати на телефон тільки після укладення контракту з Google;

• Конкуренти Android виступили з критикою платформи, звинувачуючи її в надмірній фрагментації, що створює перешкоди розробникам. Google спростувала всі звинувачення, заявивши, що ніяких подібних проблем немає;

• Піддається критиці рішення Google не розміщувати в широкому доступі код Android 3.0 Honeycomb, доступний тільки для учасників Open Handset Allianceілі за індивідуальним запитом після підписання угоди. Google мотивує це неготовністю платформи і заходом попередження недбалої її реалізації. Річард Столлман заявляв, що «все просто і ясно: за винятком ядра Linux, Android 3 являє собою невільний софт» і «незважаючи на те, що телефони з Android на сьогоднішній день не такі погані, як смартфони Apple або Windows, не можна сказати, що вони поважають вашу свободу». На даний момент Google

відкрив вихідний код для всіх версій Android аж до 4.4, а також відправив у Linux всі зміни, відповідно до GPL.

• За даними Lookout Security Mobile, за 2011 рік у користувачів Android-смартфонів було вкрадено близько мільйона доларів США (напр., Відправлення СМС без відома власника телефону). Однак це часто виникає у користувачів, які неуважно читають списки дозволів.

2.2 Особливості Android M

В рамках конференції Google I / O була представлена нова версія операційної системи Android M, а відразу після презентації попередня версія програмного забезпечення стала доступна власникам смартфонів і планшетів Nexus [10].

Домашній екран. Компанія Google незначно змінила анімації, а також додала нові шпалери з географічними мотивами.

Екран блокування. Замість запуску Дайлер на екрані блокування в лівому нижньому кутку тепер знаходиться кнопка запуску голосового асистента.

Меню програм. В меню додатків з'явилася сортування додатків за алфавітом, окремий рядок з найпопулярнішими програмами, а також дизайнери інтерфейсів зробили вибір на користь єдиної простирадла з програмами та іграми.

Лаунчер Google Now тепер дозволяє видаляти встановлені програми з домашнього екрану. Раніше з домашнього екрану можна було прибирати ярлики додатків, а для їх повного видалення - відкривати меню зі встановленими програмами. Тепер обидві дії доступні з домашнього екрану лаунчера Google Now.

Меню шерінга. Компанія Google змінила зовнішній вигляд меню шерінга, зробивши його більш зручним. Крім того, для фото з'явилася опція «Отримати посилання», щоб напряму відправити контент потрібному користувачеві, минаючи сторонні сервіси.

Не турбувати. Покращений режим «Не турбувати» в Android M дозволяє автоматично приглушати звук, або вибирати потрібні параметри вручну.

Google Now On Tap. Віртуальний асистент Google Now став ще розумнішим. За аналогією з оновленням Google Play, для використання цієї функції не потрібно чекати виходу Android M. Втім, компанія поки не відкрила доступ до Now On Tap. Голосовий асистент отримає додаткову шторку в нижній частині екрана, на якій буде представлена інформація або запропоновані дії залежно від контексту, наприклад, запущеного в даний момент програми або відкритої сторінки в браузері.

App Links. Компанія Google вирішила змінити підхід до використання посилань в операційній системі. Довгий час було дуже незручно, що при появі будь-якого посилання користувачу доводилося відкривати браузер. Але тепер, якщо розробники будуть використовувати App Links, посилання на YouTube буде відкривати плеєр, а не сторінку в браузері, посилання на Twitter запустить ваш улюблений клієнт і так далі. Нарешті в залежності від посилань, на які переходить користувач, додатки зможуть взаємодіяти між собою.

Мультивіконний режим. Google експериментує з режимом поділу екрана для одночасної роботи з двома додатками в Android M. Зручність цього рішення покаже час тестування прошивки, а після в компанії будуть вирішувати, чи варто переносити цю експериментальну функцію в релізной версію або на невеликому екрані смартфона краще тримати відкритим тільки один додаток.

Chrome Custom Tabs. Ще одна новина для розробників - можливість використовувати кастомні сторінки Chrome в своїх додатках. Працює це аналогічно вбудованому браузеру в додатках для iOS і Android, але дозволяє користуватися звичними функціями. Наприклад, з Chrome Custom Tabs можна буде перейти по посиланню в Twitter-клієнті на сторінку в Facebook і ви будете там зареєстровані, за умови, що зробили це в Chrome. Імена користувачів,

паролі, автозаповнення форм та інші браузерні дані в Android M будуть доступні повсюдно.

Поліпшене керування гучністю. Android 5 викликав багато критики через незручне регулювання гучності системних звуків. Тепер користувачі можуть вибрати загальний рівень, або розкрити спливаюче меню регулювання для зміни параметрів окремих сигналів.

Покращені копіювання і вставка тексту. Компанія Google змінила функцію копіювання і вставки тексту, зробивши її більш схожою на спливаюче меню в iOS.

Підключення по USB. В Android M з'явилося більше варіантів дій при підключенні пристрою до комп'ютера, у тому числі режим «Тільки зарядка».

Колективна клавіатура. На пристроях з великими екранами тепер можна розділити віртуальну клавіатуру, для зручності набору тексту двома руками. Аналогічну функцію ми бачили на iPad.

Google Photos. Одночасно з презентацією Android M компанія представила новий додаток для зберігання, редагування і завантаження фотографій і відео в хмару. Google Photos вже доступний для пристроїв під управлінням Android, iOS, а також в інтернеті

Карти пам'яті тепер можна використовувати як внутрішню пам'ять. При використанні microSD тепер можна вибрати режим використання: для мультимедійних даних або в якості основної пам'яті. У першому випадку нічого не зміниться, а вибір другої опції дозволить встановлювати на картку програми та ігри, зберігати будь-які дані. Простіше кажучи, вбудована пам'ять і microSD функціонально стануть єдиним цілим.

Нове Android Recovery дозволяє встановлювати оновлення з карт памяті.Компанія Google розширила можливості використання карт microSD для установки оновлень системи.

Резервне копіювання даних додатків. Тепер додатки раз на добу будуть зберігати дані в хмарі, щоб користувач при покупці нового пристрою міг

безболісно почати ним користуватися без необхідності все налаштовувати заново, вводити логіни і паролі.

2.3 Фрагментація

Фрагментація - процес дроблення чого-небудь на безліч дрібних розрізнених фрагментів. У разі Android - це присутність на ринку одночасно декількох версій системи крім останньої. Фрагментація за версіями на рис. 2.1. Фрагментація по пристроях - не єдина проблема, з якою стикаються розробники. Система сама по собі фрагментована дуже сильно, і вона буде рухатися тільки далі в цьому напрямку. Ця графіка демонструє стадії фрагментації Android за версіями і стійке зниження популярності кожної з них (біла лінія вказує на сплески)[11].

Однією з сильних сторін фрагментації Android є свобода дій, яка надається виробникам пристроїв, щоб ті, у свою чергу, могли запропонувати споживачеві пристрій, що точно відповідає його потребам. Даний фактор став особливо важливий, коли Android зайняв місце пристроїв на Symbian від Nokia в менш економічно розвинених країнах

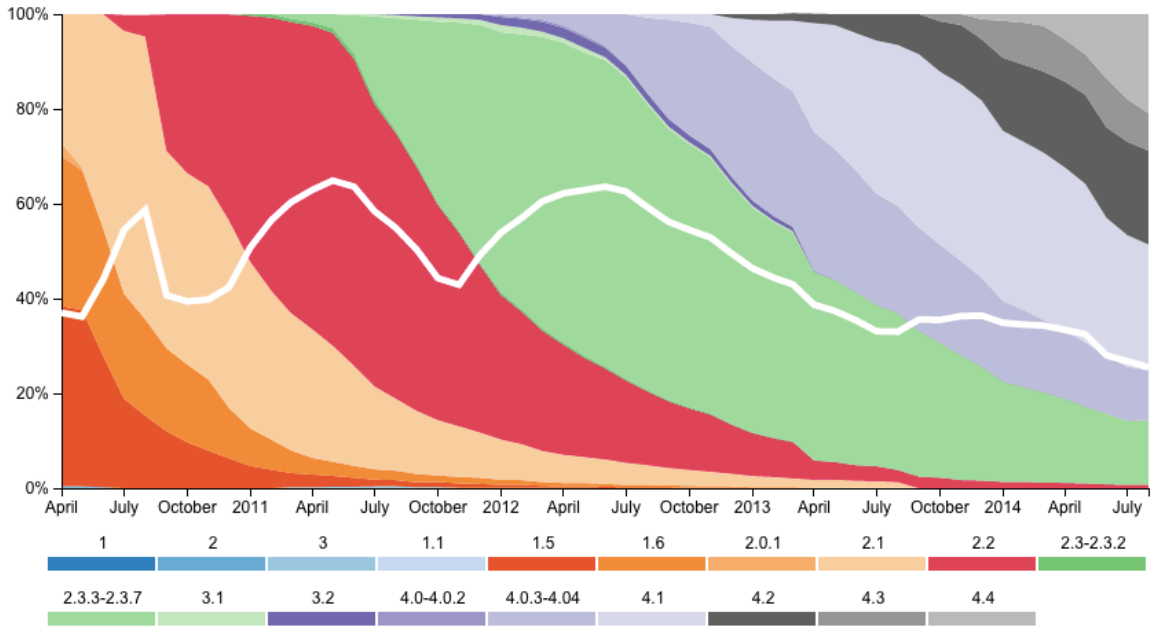

Рисунок 2.1 – Діаграмма поширення версій ОС Android [12]

Фрагментація по API репрезентативна для Android: старі пристрої використовують старі версії ОС, а з новими працюють гірше. Графік показує співвідношення ВВП на душу населення і 5 топових API для Android на ринку (4 версії KitKat і 4.3.1 Jelly Bean), більше значення по осі Y означає менший показник фрагментації. Кореляція між двома показниками очевидна, при цьому з графіка вибивається Катар с його високим ВВП, який далеко не завжди відображає реальне матеріальне становище населення [12].

Поширення системи залежно від ВВП на душу населення на рис. 2.2. Цей графік являє окремо фрагментацію для країн з ВВП на душу населення більше і менше $ 20 000, і ми бачимо, наскільки вона відрізняється. Створити додаток для економічно розвинутого ринку набагато простіше, оскільки у цільової аудиторії більше просунутих пристроїв на Android, що мають нові версії ОС. Близько 35% пристроїв в більш економічно розвинених країнах мають версію KitKat, в менш розвинених - близько 12%.

Порівняння фрагментації Android з фрашментацією iOS на рис. 2.3. Будь фрагментація Android часто показується в порівнянні з iOS. Ці дві кругові діаграми показують фрагментацію API у двох конкуруючих ОС.

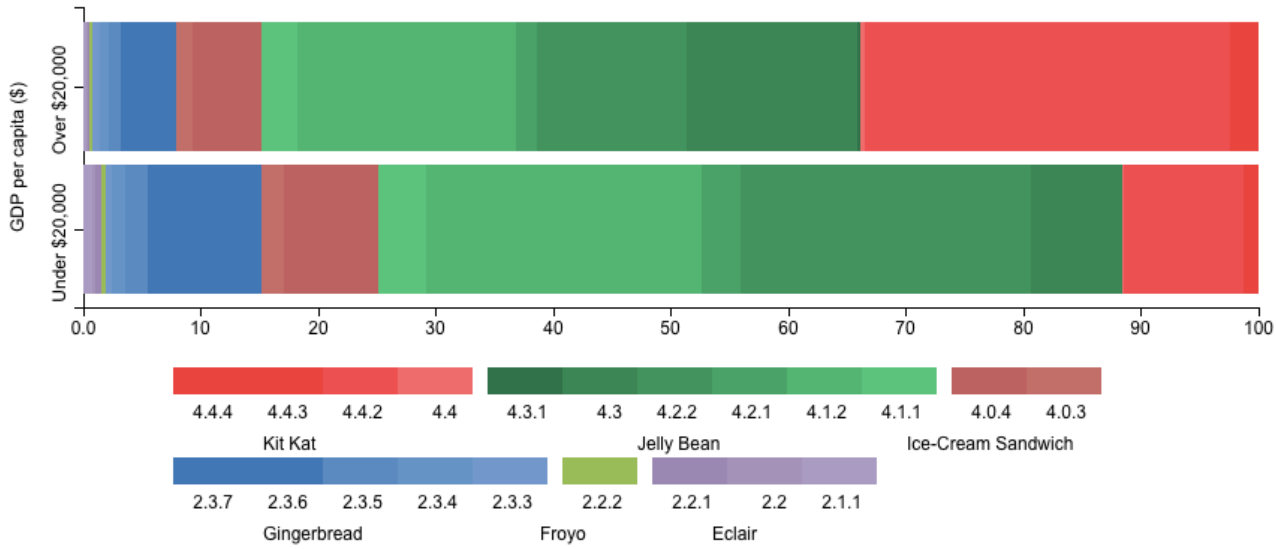

Рисунок 2.2 – Фрагментація Android в залежності від ВВП на душ населення [12]

Графік фрагментації по сенсорах на рис. 2.4. Цей графік показує розвиток пристроїв на Android з точки зору кількості використовуваних сенсорів на прикладі лінійки Galaxy S. Одне з найцікавіших напрямків розвитку - це пристрої, які здатні збирати інформацію про навколишній світ (як приклад можна привести погодну мережу Weather Signal). Серія смартфонів Galaxy демонструє, що виробники додають все більше складних сенсорів в свої пристрої. Графік демонструє також і зворотний бік явища - розробники створюють програми під певні сенсори (наприклад, згаданий погодний сервіс використовує датчики вологості і температури), а ті потім припиняють використовувати, і це додає фрагментації розробникам.

Важливий той факт, що ключ до успіху будь-якої програми - це правильне використання оболонки, і тут Android ставить перед розробниками дві принципові проблеми. Перша - це прагнення виробників кастомизировать стандартний інтерфейс (прикладами можуть служити TouchWiz у Samsung або HTC Sense). І друга - величезне різноманіття розмірів екрану у пристроїв на Android. Розробка схеми, яка адекватно працювала б на екранах різного розміру - дуже складне завдання.

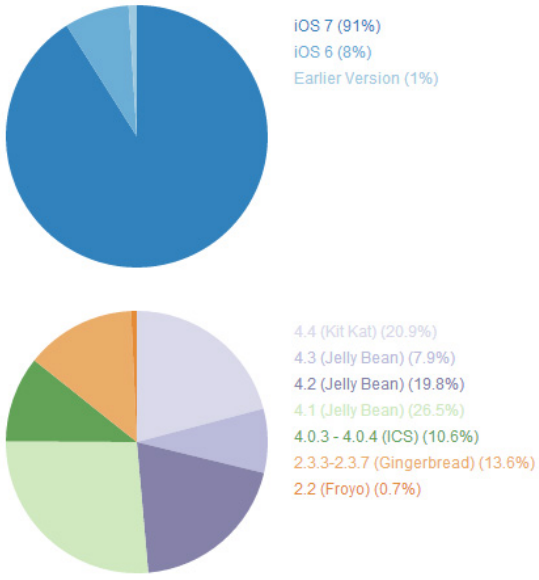

Рисунок 2.3 – Порівняння фрагментації iOS та Android – [12]

На противагу цьому, пристрої від Apple мають всього чотири розміри екрана по причині того, що в цій компанії подвоюють щільність пікселів і збільшують дозвіл в чотири рази, а екран залишають тим же. Графік нижче показує фрагментацію iOS за розміром екрану, яку легко порівняти з Android. Розміри екранах Android устройств на рис. 2.5, iOS - 2.6.

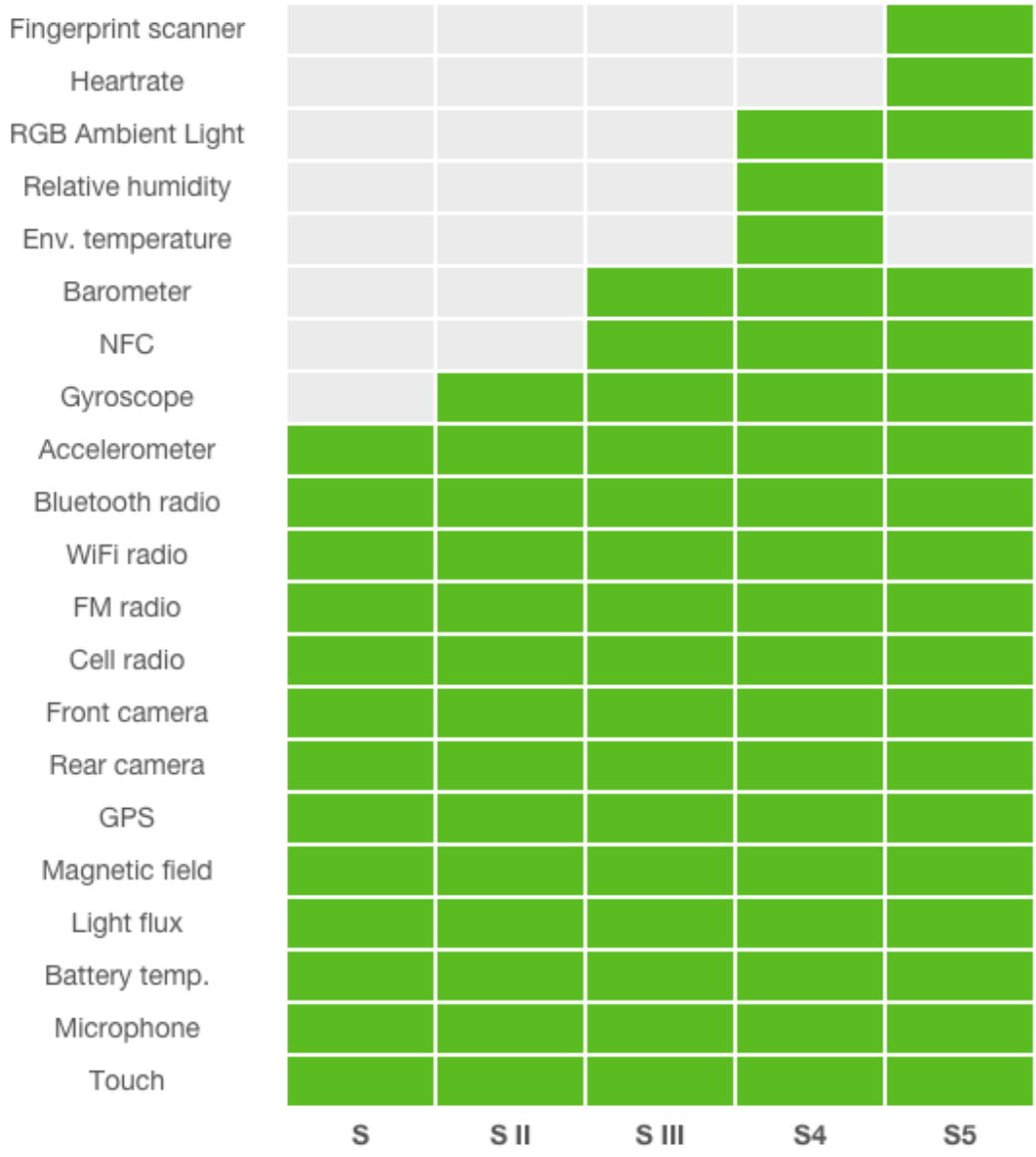

Рисунок 2.4 – Фрагментація сенсорів на прикладі лінійки смартфонів Galaxy S [12]

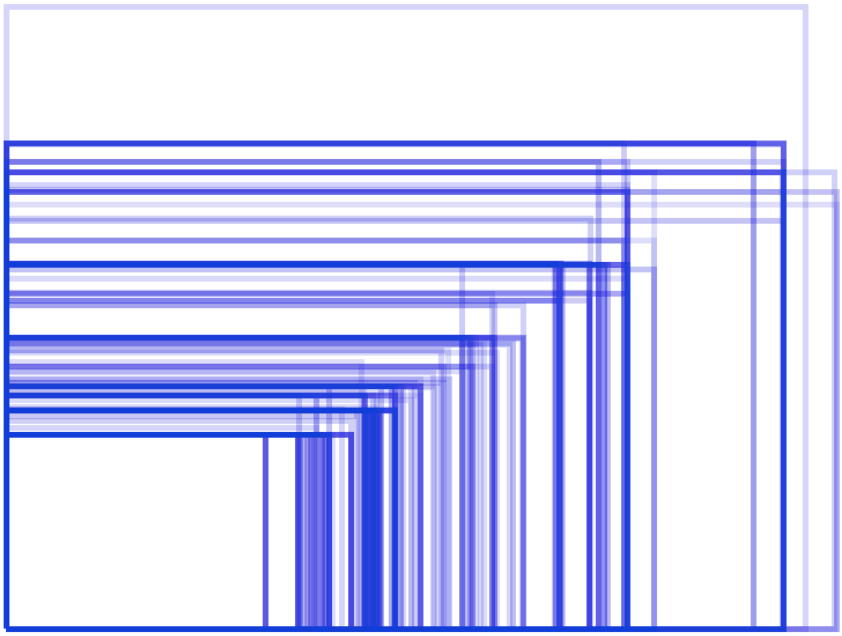

Рисунок 2.5 – Всі можливі розміри дисплеїв Android пристроїв у порівнянні

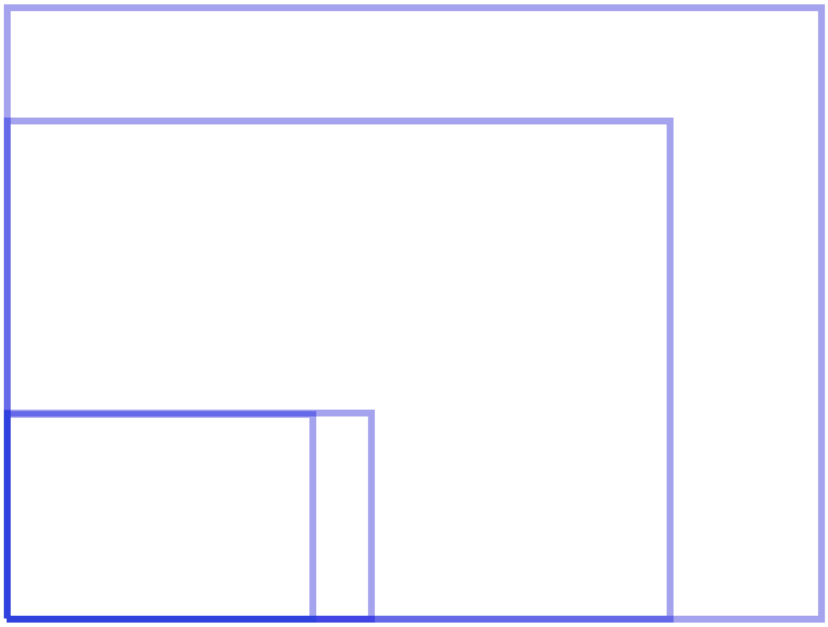

Рисунок 2.6 – Всі можливі розміри дисплеїв iOS пристроїв у порівнянні

2.4 SDK

Як і в будь-який інший мобільної ОС, в Android існує певний набір засобів розробки або далі SDK. Мета SDK - спростити написання програм під певну платформу шляхом надання готових модулів або бібліотек, для доступу

до системних функцій, середовища розробки або компілятора, і засобів тестування.

В Android на момент написання дипломної роботи був один з найбільш повних і функціональних SDK, який включає в себе просунуту середовище розробки Android Studio, набір пакетів Android Bundle, ПЗ для локалізації, ПЗ для редагування SDK (рис 2.7) і емулятор для тестування додатків.

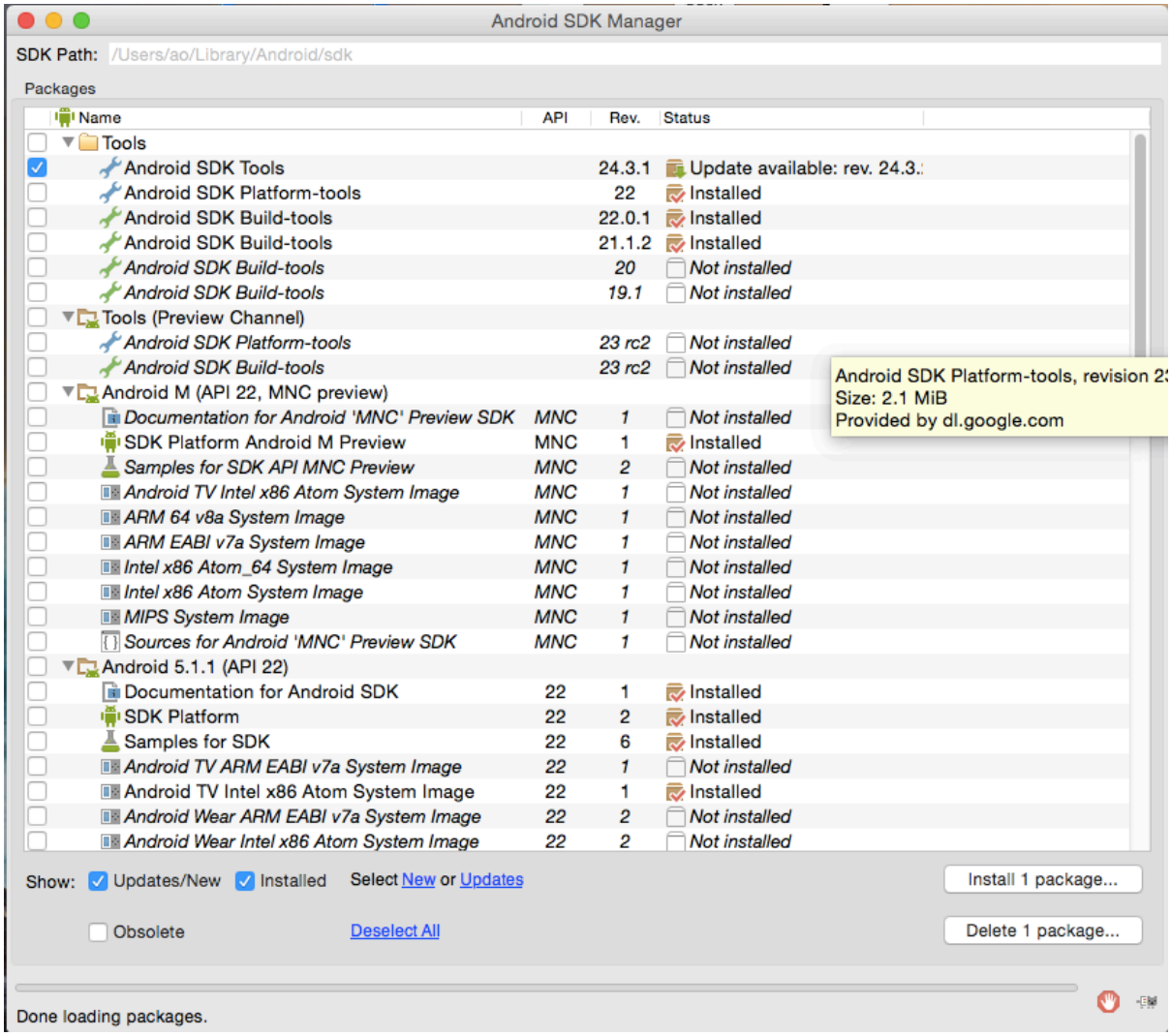

Рисунок 2.7 – Android SDK менеджер

Android Bundle складається з наборів пакетів, розбитих на «інструменти» і «екстра» [12]. В інструментах складаються пакети, необхідні для написання коду під певну версію системи з її особливостями, її arm і x86 збірки,

документація по ній і приклади коду. В екстра розташований інструментарій, для публікації додатки в Google play, драйвер для підключення телефону, драйвер платіжного сервісу та бібліотека підтримки. Як було сказано вище, фрагментація - одна з основних проблем системи андроїд і впровадження прошивки лежить на виробниках, тому щоб хоч якось згладити різницю в ПЗ Google поміщають частина нових функція в бібліотеку підтримки і за допомогою такого порту дають можливостям старих версій системи отримати більшу частину функціонала нових, а розробникам - писати більш універсальні програми.

Також важливою частиною SDK є емулятор Android. За допомогою нього можна тестувати свої програми на різних версіях системи без наявності фізичного пристрою. Правда, варто відзначити, що працюють програми в ньому в рази повільніше, відповідно для тестування високонагружаемих додатків емулятор не підходить. Також є можливість завантажити користувальницькі емулятори, які працюють трохи краще або встановити свій додаток в Chrome через спеціальний компонент. У кожному разі, Android SDK надає безліч можливостей для тестування розробниками своїх додатків. Приклад роботи емулятора з додатком SHome на рис. 2.8.

Рисунок 2.8 – Додаток SHome на емуляторі Android

2.5 Android Studio

IDE є необхідним компонентом для розробки програми під мобільні пристрої, так як зібрати і протестувати проект, використовуючи тільки редактор і компілятор стає неможливим. З'явилися системи автоматичного складання (Gradle, Maven), засоби тестування (JUnit, GreenHat), емулятори та багато іншого, що допомагає створити більш якісне ПЗ за коротший час.

Аж до 2012го року у Android не було свій IDE, тільки набір засобів розробки, вбудовується в рішення від Oracle або Eclipse. Однак на конференції Google IO 2012 була представлена перша бета-версія рідної IDE для Android - Android Studio.Android Studio створена на основі Intellij IDEA і Android Bundle, увібравши в себе переваги обох продуктів [13]. Підтримує всі засоби роботи з синтаксисом з IDEA і має вбудовані засоби створення і відображення інтерфейсу, емулятор і менеджер SDK з Android Bundle. Після релізу першої стабільної версії 1.0 в 2014 році, Android Studio стала офіційною підтримуваної Google IDE для Android. Інтерфейс на рис. 2.9.

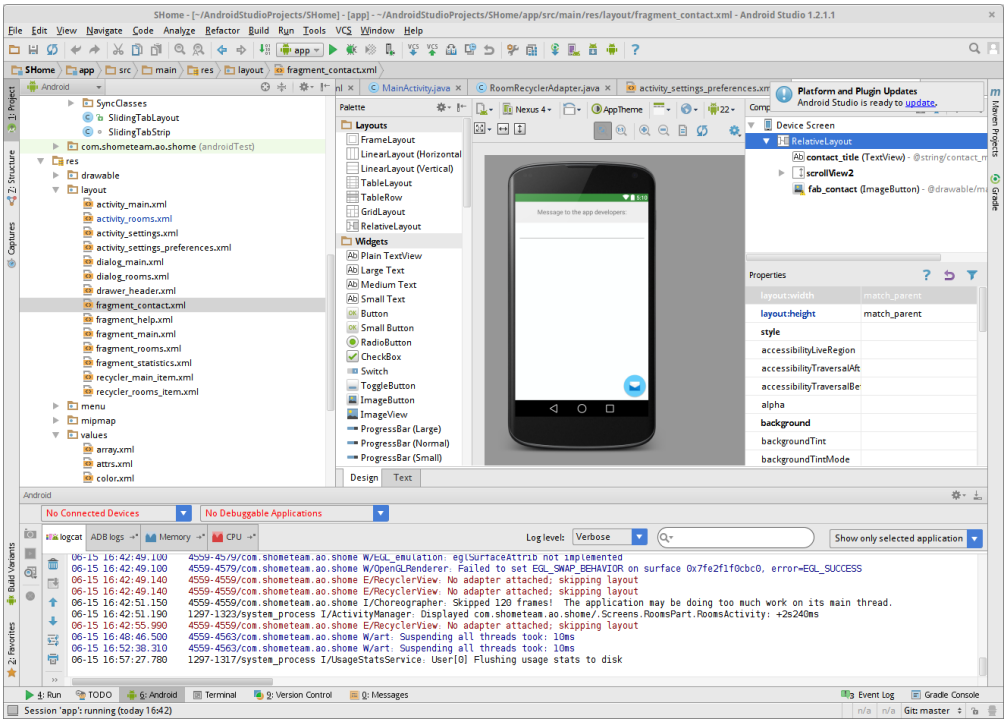

Рисунок 2.9 – Графічний інтерфейс IDE Android Studio

2.6 Висновки

Android OS – широко розповсюджена мобільна операційна система, яка охоплює 80% ринку смартфонів і планшетів. Якщо раніше основними наріканнями на систему були зовнішній вигляд і частково - недостатній функціонал, то з виходом Android M система зайняла лідуючі позиції за всіма показниками. Також, на ранніх етапах, за надмірне споживання ресурсів критикували віртуальну машину Dalvik, але з виходом Android 4.4 Kitkat і заміною Dalvik на ART це було вирішене.

З поширеності Android OS випливає і ряд недоліків, які необхідно враховувати при створенні додатків. Основним з них є висока фрагментація системи за версіями, параметрам дисплеїв і сенсорів. Присутність на ринку одночасно декількох ОС змушує розробників або створювати додатки з базовим набором функцій, або писати окремі для кожної версії системи, що не представляється доцільним. Зважаючи на ці проблеми, Google розробили

бібліотеку підтримки, що дозволяє поширити більшу частину нового функціоналу на більш ранні версії Android ОС.

Позитивним моментом є те, що завдяки популярності системи існує безліч готових пакетів і засобів для розробки додатків під неї. Google регулярно випускає нові вдосконалені версії емулятора, IDE і SDK.

Зважаючи на велику кількість інформаційних ресурсів, низьких вимог до апаратного забезпечення і широкої поширеності Android, ця операційна системи була обрана основною для написання під неї ПЗ центрального контролера і засобів візуального управління ним.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ЦЕНТРАЛЬНОГО КОНТРОЛЛЕРУ

3.1 Система взаємодії приладів

Система взаємодії між девайсами в системі SHome побудована на основі архітектури з центральним пристроєм-контролером / сервером. Виходячи з цього всі запити, які виходять з мобільного додатку надходять в чергу обробки на сервері і тільки після цього переформатуються в команди і розсилаються по мікроконтролерам (у разі реалізації з кількома кімнатами / приміщеннями). Також, в якості додаткового гаранта надійності, використовується синхронізація даних з хмарними сервісами, що у випадку технічного збою дозволить відновити систему. Схема взаємодії пристроїв зображена на рис. 3.1. Такий принцип організації має кілька значущих переваг:

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

• Зняття навантаження з мікроконтролера. Arduino mega містить обмежений набор пам'яті в 32мб і саме тому прямий доступ до неї, так само як і зберігання з обробкою великої кількості налаштувань, негативно позначаться на її функціонуванні. У випадку з сервером, логіка роботи зберігається і обробляється на ньому, а сам мікроконтролер тільки отримує прості команди виду «включити / виключити».

• Масштабованість. У випадку з використанням зв'язку Ардуіно-додаток існує певна проблема з перемиканням між мікроконтролерами для виконання різних скриптів по кімнатах. Крім того навіть якщо використовувати бездротовий зв'язок гарантувати доступ до будь-якого контролеру з будь-якої

точки будинку практично малоймовірно. У випадку з центральним сервером це навантаження він бере на себе і статично прописує всі контролери і їх положення в своїх налаштуваннях - користувачеві залишається тільки виставити бажані параметри через додаток, а за решту відповідає сервер.

Рисунок 3.1 – Діаграма, що представляє принцип взаємодії приладів у системі SHome

• Розширена система безпеки. Зважаючи на малу обчислювальну потужність мікроконтролерів, організувати більш-менш надійний криптографічний захист на ній неможливо. На відміну від мікроконтролерів, сервер має достатньо ресурсів для виконання шифрування/дешифрування інформації.

Не зважаючи на очевидні переваги, системи такого роду мають також певний перелік недоліків:

• Необхідна наявність додаткового пристрою для виконання функцій серверу. Через це збільшується складність системи та її обслуговування.

• Необхідно додаткове ПЗ для організації роботи сервера.

3.2 Синхронізація з хмарними сервісами

Хмарний сервіс або в нашому випадку хмарне сховище даних - модель онлайн-сховища, в якому дані зберігаються на численних розподілених в мережі серверах, що надаються в користування клієнтам, в основному, третьою стороною. На відміну від моделі зберігання даних на власних виділених серверах, придбаних або орендованих спеціально для подібних цілей, кількість або яка-небудь внутрішня структура серверів клієнту не видна. Дані зберігаються і обробляються в так званій хмарі, яка являє собою, з точки зору клієнта, один великий віртуальний сервер. Фізично ж такі сервери можуть розташовуватися віддалено один від одного географічно, аж до розташування на різних континентах.

У випадку системи SHome, хмарне сховище використовується для зберігання резервних копій системи і даних. Ця функція була введена для підвищення надійності системи, адже від технічних збоїв різного плану не огороджена ніяка користувацька електроніка. Зберігання ж даних на віддаленому ресурсі дозволяє відновити систему навіть при пошкодженні

інформації на носії. Також це дає можливість простого апгрейда комплектуючих без втрати інформації користувача і налаштувань. Для захисту інформації користувачів від злому використовується шифрування.

В якості провайдера було обрано рішення від Drobox Inc. Dropbox дозволяє користувачеві розміщувати файли на віддалених серверах за допомогою клієнта або з використанням веб-інтерфейсу через браузер. При установці клієнтського програмного забезпечення Dropbox на комп'ютері створюється папка, синхронізована з сервісом. Хоча головний акцент технології робиться на синхронізацію і обмін інформацією, Dropbox веде історію завантажень, щоб після видалення файлів з сервера була можливість відновити дані [14]. Також ведеться історія зміни файлів, яка доступна на період останніх 30 днів, крім цього доступна функція безстрокової історії зміни файлів «Pack-Rat». Історія зміни файлів ведеться за принципом diff-кодування, щоб заощадити місце, займане файлами. В історії зміни записується тільки відмінність однієї версії файлу від іншої. Файли, завантажені через клієнт, не мають обмеження на розмір, але файли, завантажені через веб-інтерфейс, обмежені 300 МБ. Є також можливість викладати файли для загального доступу через папку «Public», що дозволяє використовувати сервіс в якості файлообмінника. У версіях 0.8.x також з'явилася можливість надання в загальний доступ будь-якої папки в «My Dropbox» для подальшого доступу через так званий «shareable link», тобто через веб-інтерфейс. Для спільної роботи над проектами сервіс має можливість створення «Shared» папок для спільного доступу осіб, які мають різні облікові записи на сервісі. Доступна автоматична синхронізація файлів і папок і зберігання версій з можливістю відкату.

Для використання вищезгаданих технологій від Dropbox в Android додатку був використаний Dropbox Core API. Захищене з'єднання встановлюється за допомогою токена і вимагає підтвердження від користувача при використанні сервісу в додатку вперше.

Приклад екрану авторизації наведено на рис. 3.2. Щоб уникнути постійних підтверджень при передачі або завантаженні файлу, токен зберігається в SharedPreferences [15] класі програми. Для роботи синхронізації в тлі був використаний інтерфейс AssyncTask [16]. Він дає можливість асинхронно виконати код, не порушуючи при цьому роботу потоку графічного інтерфейсу і рекомендований виробником для використання в задачах, що потребують час виконання не більш 20-30 секунд для досягнення оптимальної продуктивності інтерфейсу. Цей же інтерфейс використовується також в методах завантаження і закачування програми. Приклад методу завантаження і його AssyncThread на рис 3.3.

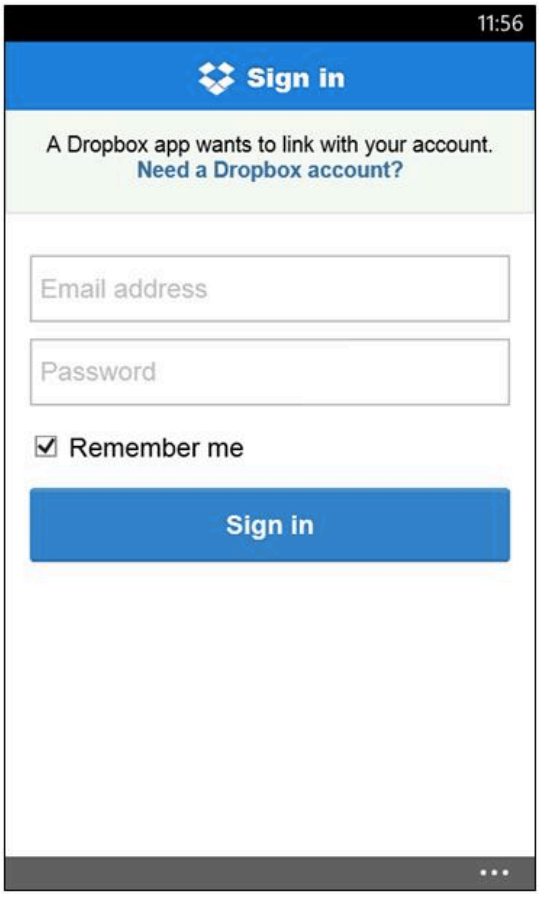

Рисунок 3.2 – Екран авторизації Dropbox при під’єднанні нового додатку до сервісу

```java
/**
 * The method for downloading files from Dropbox
 * @param fileNamesInDropbox The name of files, which you want to get
 * @param files The initialized files, where you want to write files from dropbox
 * @return ArrayList with files from Dropbox
 * @throws ExecutionException in case of wrong file name or some other clauses, which may result
 * in not finding file with your filename in Dropbox folder
 * @throws InterruptedException in case of interrupting this thread by another one
 */
public ArrayList<File> download(String[] fileNamesInDropbox, File... files) throws ExecutionException, InterruptedException {
    DropboxConnector.DownloadFileFromDropbox dwl = new DropboxConnector.DownloadFileFromDropbox(
            DropboxConnector.FILE_DIR, mDBApi, fileNamesInDropbox);
    return dwl.execute(files).get();
}

//Async classes

/**
 * The Async class for downloading file from Dropbox in new Thread. Have to be invoked in UI
 * thread
 */
private class DownloadFileFromDropbox extends AsyncTask<File, Void, ArrayList<File>> {
    /**
     * The path to app folder in Dropbox
     */
    private String path;
    /**
     * A Dropbox API class
     */
    private DropboxAPI<?> dropbox;
    /**
     * The name of files, which you want to get
     */
    private String[] fileNamesInDropbox;

    /**
     * The constructor of async download class
     * @param path The path to app folder in Dropbox
     * @param dropbox A Dropbox API class
     * @param fileNamesInDropbox The name of files, which you want to get
     */
    public DownloadFileFromDropbox(String path, DropboxAPI<?> dropbox, String[] fileNamesInDropbox) {
        this.path = path;
        this.dropbox = dropbox;
        this.fileNamesInDropbox = fileNamesInDropbox;
    }

    @Override
    protected ArrayList<File> doInBackground(File... params) {
        ArrayList<File> files = new ArrayList<>(params.length);

        int count = 0;
        for(File file: params){
            FileOutputStream fileOutputStream;
            try {
                fileOutputStream = new FileOutputStream(file);
                dropbox.getFile(path + fileNamesInDropbox[count], null, fileOutputStream, null);
                files.add(file);
                count++;
            } catch (DropboxException | IOException e) {
                e.printStackTrace();
            }

        }
        return files;
    }

    @Override
    protected void onPostExecute(ArrayList<File> resFiles) {

    }
}
```

Рисунок 3.3 – Код завантаження файлу в Dropbox та його AssyncThread класс

3.3 Система прогнозування енерговитрат

Прогнозування енерговитрат на опалення є нетривіальним й достатньо комплексним завданням з декількох причин:

• Різні властивості матеріалів, з яких побудовані будинки, а відповідно складність виведення формули для загального випадку;

• Різна товщина стін у різних будівлях;

• Використання різних склопакетів;

• Залежність енергоспоживання не тільки від зовнішньої температури, але і від вологості і сили вітру.

Враховуючи всі перераховані вище фактори, розрахувати енергоспоживання аналітичним чином досить складно, це вимагатиме від користувача ввести забагато специфічної інформації, яку він може не знати або помилитися, що в підсумку призведе до виведення некоректного результату. Інші параметри, що впливають на енергоспоживання, обчислюються з лінійної залежності від світлового дня.

Єдиним прийнятним виходом з даної ситуації є використання нейронної мережі прогнозу, яка через деякий час використання зможе давати точні свідчення, адже її результат буде залежати саме від дій і звичок користувача, а також статистичних даних за попередні періоди. Оптимальним вибором стала однослойна симетрична нейронна мережа без рекурентних зв'язків.

Дані про енергоспоживання беруться від сенсора з Ардуіно або задаються користувачем у разі відсутності такого обладнання. Точність залежить від того, як довго навчалася мережа перш ніж видати результат.

Погодні дані беруться з сервісу OpenWeatherMap за допомогою OWM API. OpenWeatherMap онлайн сервіс, який надає безкоштовний API для доступу до даних про поточну погоду, прогнозами, і архівними даними для web-сервісів і мобільних додатків. Як джерело даних використовуються офіційні метеорологічні служби, дані з метеостанцій аеропортів, і дані з приватних метеостанцій.

Інформація обробляються OpenWeatherMap, після чого, на основі даних будується прогноз погоди і погодні карти, наприклад карти хмарності та опадів. Основною ідеєю сервісу OWM є використання приватних погодних станцій, які допомагають підвищити точність вихідної погодної інформації і, як наслідок, точність прогнозів погоди [17].

OpenWeatherMap використовує безкоштовний API, щоб надати дані поточної погоди, прогнозу та карт з погодними явищами, такими як хмари, вітер, тиск і опади. Всі погодні дані можуть бути отримані в форматах JSON, XML або HTML. Максимально можливий прогноз можна отримати на місяць і щодо нього обчислити приблизний результат по енергоспоживанню. Метод отримання прогнозу на рис 3.4.

```
public float[] getWeatherForecastForDay(int dayNumber) throws IOException, JSONException { //in new Thread only 0-today
    float[] temperature = new float[4];
    OpenWeatherMap owm = new OpenWeatherMap("");
    DailyForecast today = owm.dailyForecastByCityName(mCityName, (byte)dayNumber);
    temperature[0]= today.getForecastInstance(dayNumber).getTemperatureInstance().getMorningTemperature();
    temperature[1]= today.getForecastInstance(dayNumber).getTemperatureInstance().getDayTemperature();
    temperature[2]= today.getForecastInstance(dayNumber).getTemperatureInstance().getEveningTemperature();
    temperature[3]= today.getForecastInstance(dayNumber).getTemperatureInstance().getNightTemperature();
    return temperature;
}
```

Рисунок 3.4 – Код методу отримання прогнозу погоди на один день

3.4 Збереження та редагування данних користувачів

В Android API існує кілька способів зберігати налаштування користувачів та інформацію залежно від їх призначення і використання.

SharedPreferences - клас для зберігання налаштувань і невеликих обсягів інформації іншого типу [15]. Отримати доступ до нього можна з будь-якого класу, пов'язаного з Activity або View. Мінус - зав'язаний на графічний інтерфейс і не підходить для зберігання великих обсягів інформації. Клас SharedPreferences по-замовчуванню використовується в класі PreferenceActivity для всіх його параметрів. У додатку від SHome SharedPreferences використовується для зберігання веб токена сервісу Dropbox, налаштувань і логу системи при збої. Приклад використання:

SharedPreferences prefs = context.getSharedPreferences (DROPBOX_NAME, 0);

SharedPreferences.Editor edit = prefs.edit ();

edit.putString ("KEY_TOKEN_PAIR", tokenPair);

edit.commit ();

SQL Connector - клас, для розгортання та доступу до вбудованої в систему Android бази даних SQL lite. Використовується для зберігання і подальшої обробки великих обсягів інформації в базі SQL типу. У додатку SHome використовується для зберігання історії прогноз погоди, показань з датчиків.

Тимчасові файли або TempFile - файли, які будуть видалені після закриття програми. Використовуються для зберігання тимчасової інформації. В системі SHome тимчасові файли використовуються для запису даних з БД для відправки в хмарне сховище. Також в бета-версії програми лог краш записується в тимчасовий файл і відправляється на пошту розробників.

3.5 Графічний інтерфейс користувача

Зважаючи на використання центрального контролера в ролі сервера, оптимальним є функціонування програми в фоні, а тому графічний інтерфейс був мінімізований до виводу і сортування логів, щоб визначати коректність роботи системи при тестуванні. Все інше управління відбувається через мобільний клієнт. Знімок екрану з графічним інтерфейсом сервера на рис. 3.5.

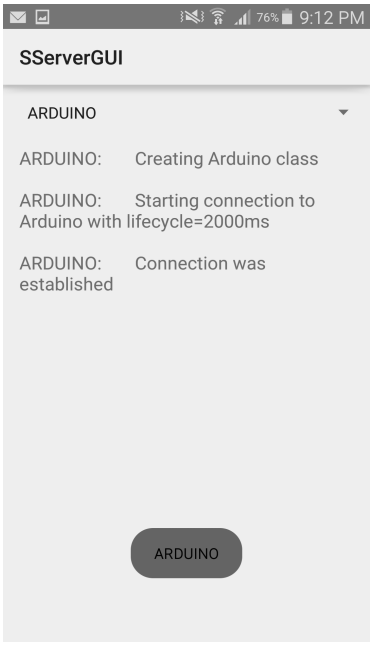

Рисунок 3.5 – Графічний інтерфейс додатку центрального контроллеру

Графічний інтерфейс представляє з себе Activity з Spinner і TextView. Spinner, він же список, зберігає в собі теги з класу логер по яких ведуться записи в текстове поле. TextView відображає текстову інформацію по логам в залежності від обраного елемента в Spinner.

3.6 Взаємодія користувача з системою

Найкраще продемонструє принципи взаємодії користувача з екосистемою SHome use-case діаграма на рис 3.6. Вона відображає у вигляді дій шлях від запиту користувача до отримання результату у вигляді відгуку системи, поетапно показуючи всі суміжні процеси. Наприклад, для зміни кліматичних параметрів користувач формує запит на мобільному телефоні, далі телефон робить з нього повідомлення-запит потрібного формату і відсилає його на сервер, де, після обробки вона надсилається на Arduino, яке дешифрує скрипт і виконує закодовані в ньому дії.

Програмно взаємодія налаштована через сокети по протоколу telnet. В ПЗ як сервера, так і клієнта є класи для передачі інформації та для отримання. Отримання йде в окремому фоновому потоці і сам процес починається, коли один з класів-передавачів висилає запит на відправку інформації по каналу. Завдяки використанню окремим потоків можлива множинна відправлення та отримання інформації, так як клас-одержувач може створити кілька десятків потоків отримання, пов'язаних із загальною чергою виконання, з якої завдання і підуть на обробку. Підхід був би невиправданий, якщо кількість запитів перевищувала б час обробки, проте система управління розумним будинком не є високонагруженою і максимум можливого - якщо всі користувачі захочуть змінити мікрокліматичні параметри одночасно, а з такого роду завданнями система впорається навіть на слабких комплектуючих. Втім, цей параметр залежить скоріше від апаратного забезпечення і тому SHome цілком може справлятися з більш складними завданнями.

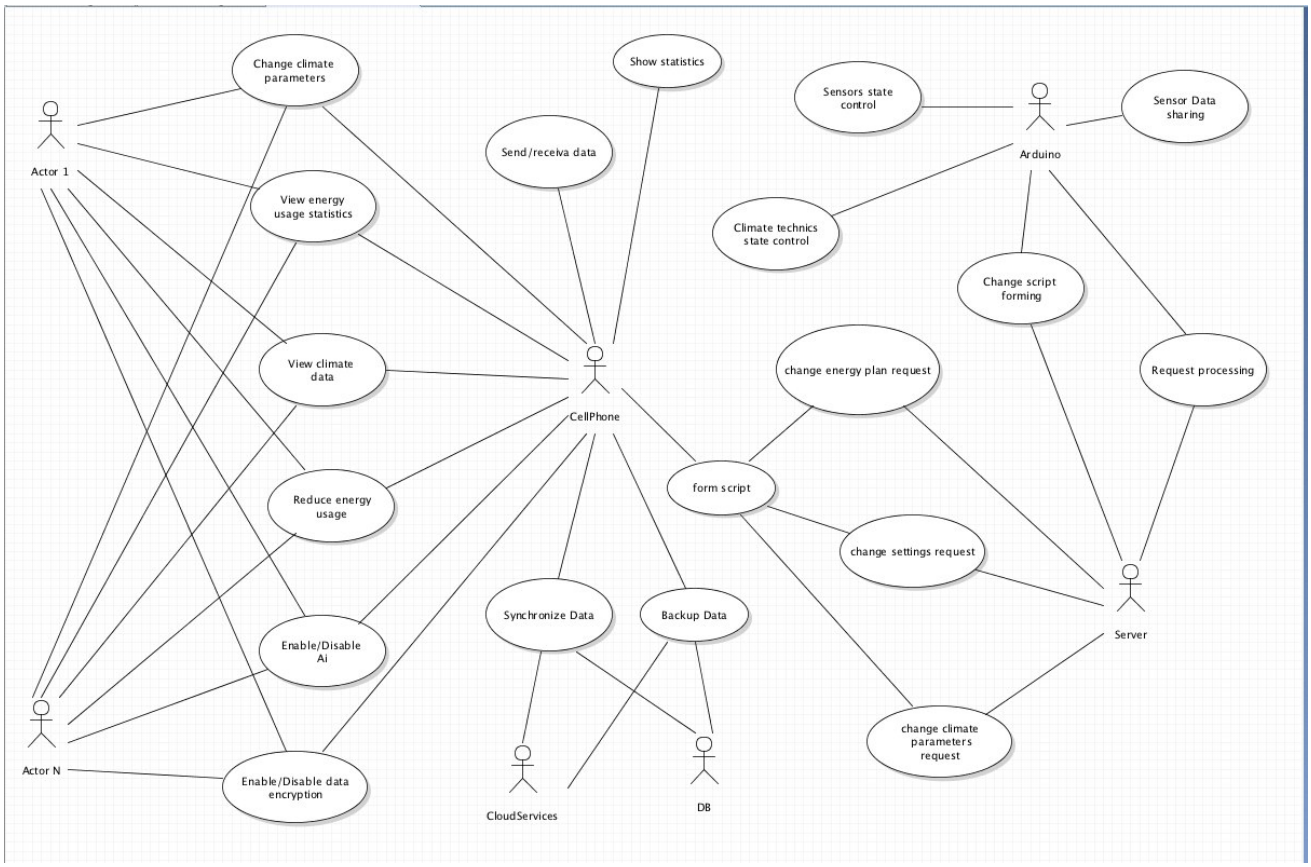

Рисунок 3.6 – Use-case діаграмма системи “Розумний” дом від SHome

3.7 Висновки

В архітектурі системи SHome 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.

Для огранізації цих функцій були використані засоби Android API, Dropbox Core API, OWM API, JAVA SE7. Код может бути запущений на будь-якому пристрої з версією Android не молодше 4.1.

Код серверу опублікованій під вільною ліцензією GNU GPL и є відкритим для внесення будь-якіх змін й поліпшень співтовариством. Також

весь код викладений у вільному доступі на хостингу коду Github и будь-хто може создати свій власний клон проекту або запропонувати зміни в існуючий. Для розгортання серверу на своєму обладнанні, за умови, що воно входить до списку підтримуваних або має необхідні засоби віртуалізації, не потрібні ніякі додаткові технічні навички - все робиться шляхом простої установки програми з Google Play або скачуванням apk безпосередньо з сайту проекту SHome.

4 ПРОГРАМНЕ ЗАБЕЗБЕЧЕННЯ МОБІЛЬНОГО КЛІЄНТУ

4.1 Графічний інтерфейс

Мобільний клієнт є основним засобом інтерракції з користувачем, тому дизайн його графічного інтерфейсу відіграє основну роль. Саме відносно нього формується думка користувача про систему. У такому випадку основні вимоги до дизайну мобільного додатку виглядають наступним чином:

• Рідний вид для операційної системи;

• Інтуїтивність;

• Дружелюбність до користувача;

• Простота налаштувань.

Інтуїтивність - це один з основних параметрів, що визначають вдалість інтерфейсу безпосередньо впливає на зручність його використання. Для визначення цього параметра проводять безліч соціальних досліджень і опитувань, на основі яких виробляють кінцеву концепцію для кожного стилю оформлення. Найпростішим способом зробити додаток інтуїтивним є слідування рекомендаціям до дизайну додатків, визначеними творцями операційної системи, в якій додаток буде працювати. У разі SHome рекомендації були взяті з сайту для розробників Android.

Під дружелюбністю до користувача мається на увазі зняття якомога більшого навантаження з самого користувача і розподіл його на внутрішню логіку програми, що дозволить знизити поріг входження. Інтерфейс не повинен бути перевантажений величезною кількістю різноманітних полів введення і view - їх повинно бути рівно стільки, скільки мінімально вимагає логіка програми.

Простота налаштувань - ще один ключовий фактор вдалого інтерфейсу. Налаштування повинні бути зібрані в одному місці і розбиті на секції за

призначенням. У випадку з Android рекомендується створити окрему Activity для цих цілей.

Рідний вигляд програми допомагає користувачеві швидко розібратися з інтерфейсом і крім того робить сам додаток гармонійною частиною тієї операційної системи, в якій він використовується. У випадку з SHome додаток функціонує в операційній системі Android і має відповідати концепціям дизайну, визначеним у Google. Збірна назва таких концепцій - дизайн матеріалів або Material Design цього типу дизайну: у його рамках об'єкти «відповідають» користувачеві тінями і реакцією на дотики. На перший погляд новий концепт занадто «примітивний». Він плоский і багатошаровий. Шари позначені тінями. Ідея така ж, як у фізичної книги або зошиту, і полягає в «гортанні сторінок». Розробники отримали можливість задавати визначення кута окремих елементів інтерфейсу і промальовувати тіні в режимі реального часу.

«Матеріальний» дизайн нового Android характеризують жирні яскраві шрифти і адаптивна «палітра», що дозволяє додаткам пристосовувати свої кольори до кольорової гами відображуваного їм контенту. Галерея знайде тони відображеної нею картинки, а музичний плеєр - обкладинки відтвореного альбому. Це найпростіші приклади того, на що здатна ця сама знаменита адаптивна «палітра» [18].

Іншою ключовою особливістю Material design є відгук на торкання користувача. Натискання кнопки анімовано. Тобто, користувач буде бачити, на які елементи інтерфейсу він «натискає». Ця можливість поширюється на всі програми, сторінки, меню та інші складові операційної системи [19].

Згідно перерахованим вище принципам і був сформований графічний інтерфейс мобільного клієнта SHome.

Домашній екран зображений на рис. 4.1, екран статистики на рис. 4.2, екран зворотного зв'язку на рис. 4.3, екран налаштувань на рис 4.4, бічне меню на рис. 4.5.

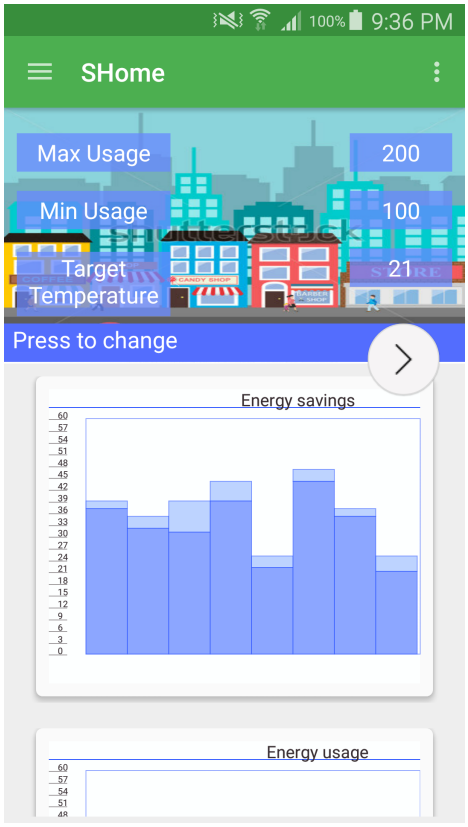

Рисунок 4.1 – Головний екран додатку SHome

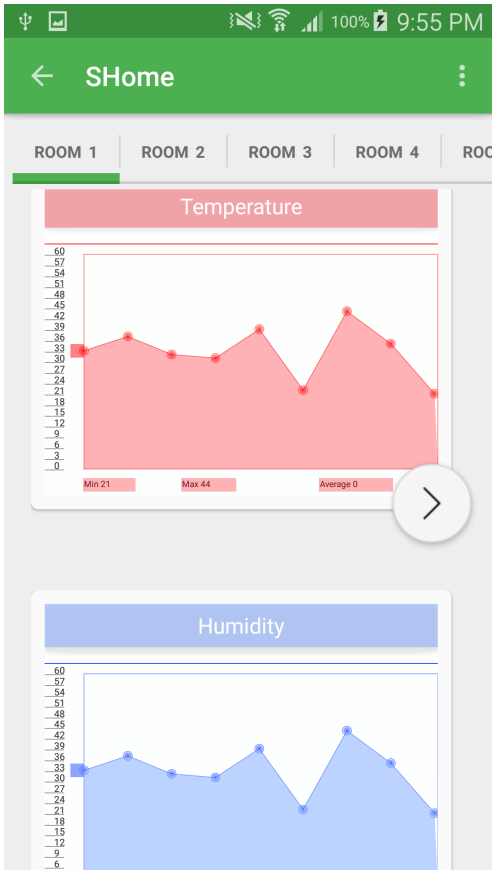

Рисунок 4.2 – Екран статистики додатку SHome

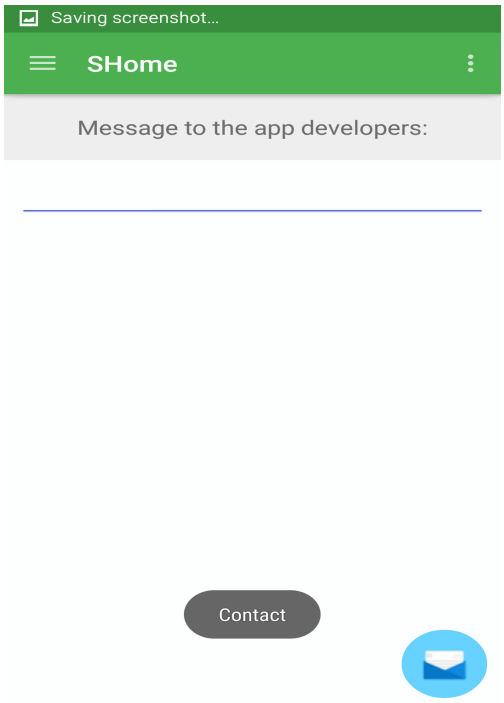

Рисунок 4.3 – Екран зворотнього зв’язку додатку SHome

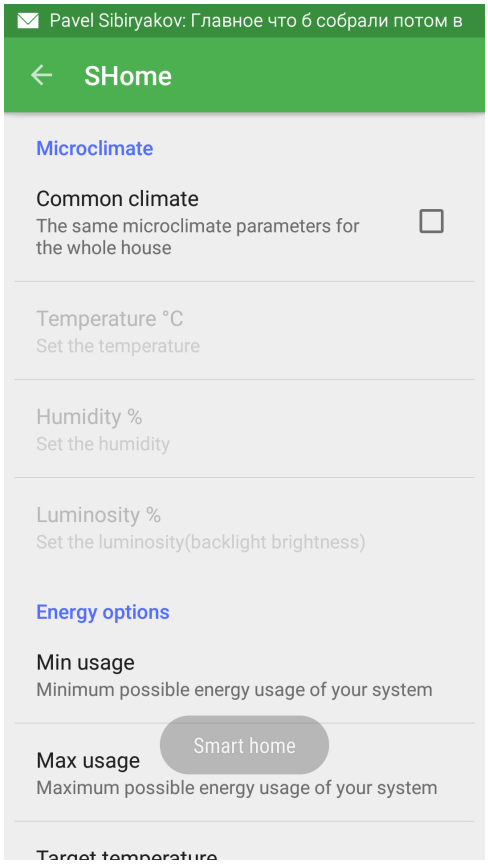

Рисунок 4.4 – Екран налаштувань додатку SHome

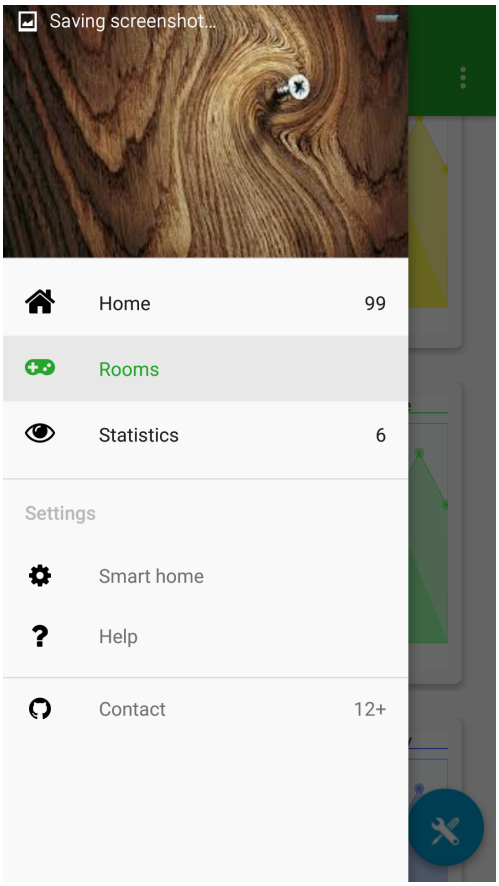

Рисунок 4.5 – Бокова панель навігації додатку Shome

4.2 Принципи відображення інформації

Від правильно вибудуваних шляхів інтеракції залежать як інформативність додатку, так і досвід користувачів в цілому. Від вибору найбільш оптимального способу виведення інформації в додатку залежить не тільки його рейтинг, але й найважливіший фактор для будь-якого розробника - ухвалення або не ухвалення додатка цільовою аудиторією.

Так як на серверну частину доводиться тільки обробка інформації та запитів, то весь графічний інтерфейс відображений в мобільному клієнта. За статистикою, візуальна інформація сприймається на 60% краще, ніж текст, тому було прийнято рішення максимально використовувати графіки і диаграма.

Перша проблема, що виникла при виборі даного підходу - відсутність в Android SDK відповідних віджетів. На жаль, спільнота також не публікувала вихідних кодів для даних типів view, тому єдиним виходом стало створити їх з нуля.

У мобільній операційній системі Android вся логіка додатків зав'язана на інтерфейс-орієнтований підхід, тому центральним класом є клас Activity і його атрибути. Activity - це і є відображення одного екрану додаток, що пов'язує xml-інтерфейс розмітки з програмною логікою і алгоритмами, зав'язаними на елементи управління. По суті, її можна розглядати як контейнер для виводу відповідних віджетів або View класів, які їх представляють. Є кілька способів розмістити View в Activity: статичні, він вимагає чітко прописати кожен view і його зв'язок з іншими компонентами у файлі xml-розмітки, і динамічний, що використовує класи Holderи для розміщення динамічно створених віджетів у вигляді списку.

За принципом побудови інтерфейсу Material Design будується навколо концепції карток і тіней, тому, для відповідності нормам дизайну, визначеним виробником, був обраний динамічний шлях розміщення контенту у вигляді карток. Програмно це реалізується за допомогою використання класів CardView і RecyclerView.Adapter.

RecyclerView, по суті, є еволюцією одного з найнеобхідніших в Android-розробці віджетів - ListView. Власне, призначення у нього рівно те ж саме - відображати список елементів, але є нюанси: обов'язкове використання патерну ViewHolder. Якщо при використанні ListView можна було через відсутність досвіду використовувати адаптер, що створює з нуля окреме view для кожного елементу списку, що при великому розмірі списку могло обернутися меншою чуйністю UI і використанням зайвої пам'яті, то при роботі з RecyclerView розробника насильно приводять до імплементації цього патерну [19].

LayoutManager. Для використання RecyclerView крім адаптера необхідно передати йому за допомогою методу setLayoutManager () об'єкт класу, що

реалізовує LayoutManager. Цей клас відповідає за роботу з адаптером, саме він вирішує, перевикористати View або створити новий, і відповідно, саме він смикає методи onCreateViewHolder (), onBindViewHolder () і getItemCount () адаптера. Поки доступна тільки одна реалізація цього класу - LinearLayoutManager, для створення кастомними LayoutManager, необхідно успадкувати від RecyclerView.LayoutManager.

CardView - це віджет, який імплементує такий елемент дизайну Material, як картка. По суті це контейнер, у якого можна задавати радіус заокруглення кутів, колір картки і висоту по осі z. Для його ипользование досить вставити його тег перед тегом відповідного View в Xml-файлі [19].

Однак, цих двох класів достатньо тільки для виведення стандартних View або їх комбінацій, у разі ж з одними віджетами потрібно попередньо зробити кілька кроків для імплементації необхідного функціоналу. Перший - створити стандартний View і перевизначити в ньому метод onDraw (), що відповідає за промальовування графічної складової. Незручність методу полягає в тому, що потрібно попіксельно задавати рамки і об'єкти, що вельми незручно при виведенні графіків. Також проблемою є отримання інформації з центральних ресурсів, так як для доступу до них потрібно мати контекст-клас певної Activity. Передавати Context безпосередньо вважається поганим тоном, тому був обраний спосіб, який реалізує його отримання через Bundle-клас. Також, був створений клас для колірної схеми, щоб було простіше використовувати групи кольорів при доступі з Holder- класу.

Використання внутрішніх ресурсів xml для зберігання інформації про колір і мірках має кілька переваг: централізація всіх ресурсів і легкість їх зміни, напівтоновість в кольорах (зчитування у форматі RGBA), більш проста адаптація під різні співвідношення сторін й роздільчи здатності екрану.

Також, були створені власні теги XML для графічного View, щоб мати доступ до зміни його зовнішнього вигляду або відображення статично, безпосередньо з файлу розмітки, без використання Java коду. Для цього були використані

можливості класу Bundle по роботі з XML-параметрами. Приклад кастомного View на рис. 4.6.

Вид підсумкового View, для відображення графіків на рис. 4.7, його в обгортці CardView на рис 4.8. Код в xml форматі на рис. 4.9.

```
<view
    xmlns:graphic_view="http://schemas.android.com/apk/res/com.shometeam.ao.shome"
    android:layout_width="match_parent"
    android:layout_height="320dp"
    class="com.shometeam.ao.shome.GraphicViews.GraphicView"
    android:id="@+id/gr_view"
    android:layout_gravity="center_horizontal"
    graphic_view:mtext = "asd"
    graphic_view:mtype = "linear"
    graphic_view:mcolor = "red"/>
```

Рисунок 4.6 – View з власними XML тегами

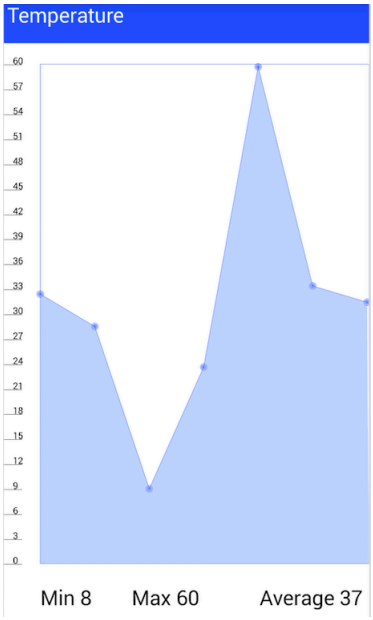

Рисунок 4.7 – View класс для відображення графіку

Рисунок 4.8 – View класс графіку в обертці CardView

```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/card_view"
    android:layout_gravity="center"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginBottom="5dp"
    card_view:cardCornerRadius="5dp"
    card_view:contentPadding="10dp"
    card_view:cardElevation="2dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="380dp"
        android:orientation="vertical">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="32dp"
            android:id="@+id/graphic_name"
            android:text="Humidity"
            android:layout_gravity="center"
            android:gravity="center"
            android:textAlignment="center"
            android:textSize="16sp"
            android:textColor="@color/text_color_light"
            android:elevation="4dp"
            android:background="@color/material_drawer_divider"
            />

        <view
            xmlns:graphic_view="http://schemas.android.com/apk/res/com.shometeam.ao.sh
            android:layout_width="match_parent"
            android:layout_height="320dp"
            class="com.shometeam.ao.shome.GraphicViews.GraphicView"
            android:id="@+id/gr_view"
            android:layout_gravity="center_horizontal"
            graphic_view:mtext = "asd"
            graphic_view:mtype = "linear"
            graphic_view:mcolor = "red"/>

    </LinearLayout>

</android.support.v7.widget.CardView>
```

Рисунок 4.9 – Код xml-розмітки віджету графіку у обертці CardView

4.3 Принципи взаємодії з сервером

Взаємодія клієнта з сервером базується на обміні повідомленнями певного формату за допомогою мережевого каналу. Для уникнення формування помилкових повідомлень, були виведені неколько конструкторів для різних типів повідомлення з обмеженими комбінаціями тегів з основних enum. Структура стандартного повідомлення виглядає наступним чином:

- Параметр;
- Дія;
- Значення.

Код класу повідомлення наведено на рис 4.9. Для обмеження можливих опцій створення повідомлення використовується зв'язка з private класів і статичних методів для створення їх об'єктів. Тобто, програміст не може створити екземпляр повідомлення безпосередньо, минаючи статичні методи, а значить і формування повідомлення з невірними параметрами виключено. Часто розробники виносять цей функціонал на графічний інтерфейс, проте підхід по відділенню логіки від GUI зарекомендував себе як більш надійний, тому реалізація була імплементована саме таким чином.

```
private Message(ACTIONS action, PARAMETER parameter){
    mAction = action;
    mParameter = parameter;
}

private Message(ACTIONS action, PARAMETER parameter, float setValue){
    mAction = action;
    mParameter = parameter;
    mSetValue = setValue;
}

private Message(ACTIONS action, PARAMETER parameter, boolean switchValue){
    mAction = action;
    mParameter = parameter;
    mSwitchValue = switchValue;
}

private Message(ACTIONS action, PARAMETER parameter, ArrayList<? extends Number> respondValues){
    mAction = action;
    mParameter = parameter;
    mRespondValues = respondValues;
}

public static Message createInfoMessage(PARAMETER getParameter){
    return new Message(ACTIONS.GET, getParameter);
}

public static Message createSetMessage(PARAMETER setParameter, float setValue){
    return new Message(ACTIONS.SET, setParameter, setValue);
}

public static Message createSwitchMessage(PARAMETER switchParameter, boolean switchValue){
    return new Message(ACTIONS.TURN, switchParameter, switchValue);
}

public static Message createRespondMessage(PARAMETER respondParameter, ArrayList<? extends Number> respondValues){
    return new Message(ACTIONS.GET, respondParameter, respondValues);
}
```

Рисунок 4.9 – Java код класса Message

Передача ведеться через інтерфейс соккет з пакету Java IO. Сервер «слухає» канал на виникнення нових підключень і після цього, у разі їх виникнення, створює окремий потік для кожного, щоб отримання кілька повідомлень водночас не перериваючи роботи програми і не викликаючи колізій або дедлоків.

Крім того, у додатку реалізован серйозній захист від доступу сторонніх, що представляє із себе фабрику сертифікатів і AES шифрування повідомлень.

4.4 Елементи навігації та взаємодії віджетів

Як писалося раніше, вся логіка андроїд додатків будується навколо класів, успадкованих від View або Activity, так як отримати доступ до ресурсів додатки можна тільки через них або їх Context класи. Однак динамічно можна міняти тільки віджети допомогою Holder-класів або заміною фрагмента на інший. Розглянемо ці продходи в порівнянні:

Holder-клас є раціональним рішенням для виведення різних списків віджетів заздалегідь визначеної структури за допомогою адаптера, однак для відображення одного віджета цей підхід не годиться. Основна проблема - на ініціалізацію трьох класів для настройки Holder витрачається певна кількість ресурсів. До них додаються також пам'ять на динамічне виділення під віджет і на його промальовування. Підхід буде виправданий у тому випадку, якщо потрібно виводити велику кількість однотипної інформації, однак для ініціалізації одного об'єкта він буде занадто ресурсоємним.

FragmentManager-клас для зміни фрагментів. Це рішення оптимально для заміни одного View на інше або для адаптації додатка і його компонентів під планшетний режим або зміну орієнтації екрану. Втім FragmentManager використовується для заміни фрагментів з коду програми з якоїсь дії. Щоб просто змінити положення фрагментів на екрані можна прописати відповідну поведінку у відповідних маніфест-файлі і xml-файлі.

Створити нове Activity. Раніше, до появи Holder і Fragment, це був єдиний спосіб для створення програми Мультиекрани. Інформація з одного Activity в інше можна було передати через Intent клас, необхідний для переходу. Мінус такого підходу в тому, що потрібно створювати три різних Activity: для двох варіацій орієнтації екрану і для планшета. Також явним недоліком є перерисовка всього екрану замість його частини. Google рекомендує використовувати Activity лише в тому випадку, коли неможливо зробити ті ж операції з фрагментами.

Всі три підходи використовуються в побудові комплексних програм та їх елементів навігації. У додатку SHome app для перемикання між екранами використовується Navigation Drawer шаблон або просто бічне меню. Як елемент навігації бічне меню найболее часто використовується в додатках з декількома різнотипними інформаційними екранами [20]. У форматі XML - це просто тип розмітки з вбудованим обліковим висновком скорочень-переходів. На рис. 4.10 зображений найпростіший варіант реалізації Navigation Drawer.

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <!– Основное содержимое -->
    <FrameLayout
        android:id="@+id/content_frame"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <!– Боковое меню -->
    <ListView android:id="@+id/left_drawer"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:choiceMode="singleChoice"
        android:divider="@android:color/transparent"
        android:dividerHeight="0dp"
        android:background="#111"/>
</android.support.v4.widget.DrawerLayout>
```

Рисунок 4.10 – XML розмітка базової реалізації Navigation Drawer

Крім XML файлу інтерфейсу також необхідний код для заміни фрагментів на екрані при виборі того чи іншого пункту бічного меню. Для цього і використовуються Inflater і FragmentManager класи. FragmentManager відповідає за зміну одного фрагмента на інший шляхом коміта транзакцій, а Inflater-клас - за отрисовку частин фрагмента після його заміни. Код заміни фрагмента на рис. 4.11.

```
FragmentManager fragmentManager = getFragmentManager();
fragmentManager.beginTransaction()
               .replace(R.id.content_frame, fragment)
               .commit();
```

Рисунок 4.11 – Java код для заміни поточного фрагмента на новий

4.5 Локалізація додатку

Локалізація додатку відіграє важливу роль у його подальшому поширенні, проте завжди доставляє ряд проблем розробнику:

• Зміна розмітки при зміні довжини полів з текстовими даними при перекладі;

• Неточність переведених значень щодо контексту;

• Механічна складність перекладу всіх значень, якщо вони зберігаються не централізованою.

Розглянемо шляхи вирішення цих проблем.

Зрушення розмітки неминучий при використанні XML - скриптів і статичної отрисовки компонентів. На жаль, але Android погано підходить для динамічно форматірумих додатків преймущественно через високу фрагментації системи. Різні розміри екранів, їх роздільна здатність і співвідношення сторін, а так само велика варіація різних версій системи на ринку просто фізично неможливо створити API, яке зможе підлаштувати розташування компонентів для них усіх. Єдиним вирішенням цієї проблеми при перекладі є тестування розмітки і створення розміру компонентів «із запасом».

Проблеми з неточністю перекладу з контексту виникають при використанні автоматичних засобів перекладу у зв'язку з різними синтаксичними особливостями мовних груп. Саме з цієї причини рекомендується проводити переклад вручну.

В операційній системі Android хорошим тоном вважається зберігати всі текстові значення в соответсвующей директорії xml, щоб уникнути проблем подібного роду. Для тих випадків, коли це неможливо, рішенням є використання програми аналізу apk, яка вмонтує переклад в залежності від локалі. Проблему цей метод вирішення усуває, але вимагає додаткового дискового простору для утиліти. Не рекомендується до використання в серйозних проектах.

Враховуючи вищеописані особливості, в додатку SHome всі текстові ресурси зберігаються в string.xml файлі, для простоти перекладу. Існують ручні локалізації для російської та англійської мов, а також автоматичні для інших. При виявленні користувачами помилок і неточностей перекладу, вони завжди зможуть звернутися до розробників шляхом зворотного зв'язку прямо з програми і запропонувати корективи, або внести свій комит в центральний репозиторій на хостингу коду GitHub.

4.6 Висновки

Враховуючи описані особливості цільової операційної системи, для створення ресурсоефективного додатку, що відповідає новим тенденціям в плані дизайну інтерфейсу, були виконані наступні кроки: комбіноване використання засобів статичної та динамічної розмітки, централизована локалізація, використані стандартні методи повідомлення клієнта з сервером і розвинута система захисту даних. Статично встановлюється розмітка компонентів, а динамічно - їх угруповання на екрані смартфона.

Способи інтерракції реалізовані з метою створення найкращого досвіду користувача. Навігація між частинами додатка виконується за допомогою використання шаблону Navigation Drawer від Google. Результат відповідає уявленням замовника про систему та узгоджений в технічній документації.

Взаємодія з сервером відбувається за допомогою протоколу telnet і реалізована програмно через клас Socket з пакету java.io. Підтримується одночасний прийом інформації від декількох передавачів.

Для локалізації були використані автоматичні засоби перекладу. Для спрощення доступу до текстових ресурсів і їх переведенню всі вони зберігаються у файлі string.xml. Щоб коригувати помилки автоматичного перекладу, є можливість використовувати систему зворотного зв'язку з розробниками прямо з програми.

Програмне забезпечення мобільного клієнта містить всі обумовлені в технічному завданні функції і знаходиться на ранній стадії відкритого тестування, щоб закрити ті помилки, які не виявили автоматичні тести. Весь код опублікований в публічному репозитарії в Github.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Вступ

У даному розділі проводиться аналіз середовища, в якому проводилося дослідження програмного продукту на основі санітарних норм України.

Проводилося дослідження роботи з розробки мобільного додатку керування системою «Розумний будинок», напряму пов'язані з роботою на комп'ютері.

Робота персонального комп’ютеру та тестувальних пристроїв (смартфон, планшет) впливають на фізичні і хімічні чинники середовища: електромагнітні випромінювання, статична електрика, температура і вологість повітря, вміст кисню і озону. Повітря забруднюється шкідливими хімічними речовинами антропогенного походження за рахунок деструкції полімерних матеріалів, що використовуються для обробки приміщень та обладнання.

Організація робочого місця з порушенням норм призводить до загальної та локальної напруги м'язів шиї, тулуба, верхніх кінцівок, викривлення хребта і розвитку остеохондрозу, зміні внутрішнього тиску у оці та інших проблем з органами зору.

5.1 Аналіз умов праці в приміщенні

Дослідження, що проводилося в даній дипломній роботі, проводилося в приміщенні, план якого приведено на рис. 5.1.

Приміщення має одностороннє природне освітлення (вікно: висота = 1,5 м, ширина = 2.5 м) і загальне штучне освітлення. Стіни і стеля обклеєні світлими шпалерами, підлога вкрита світлим ламінатом. У приміщенні відсутні сильні вібрації та шкідливі речовини. Склад повітря в нормі.

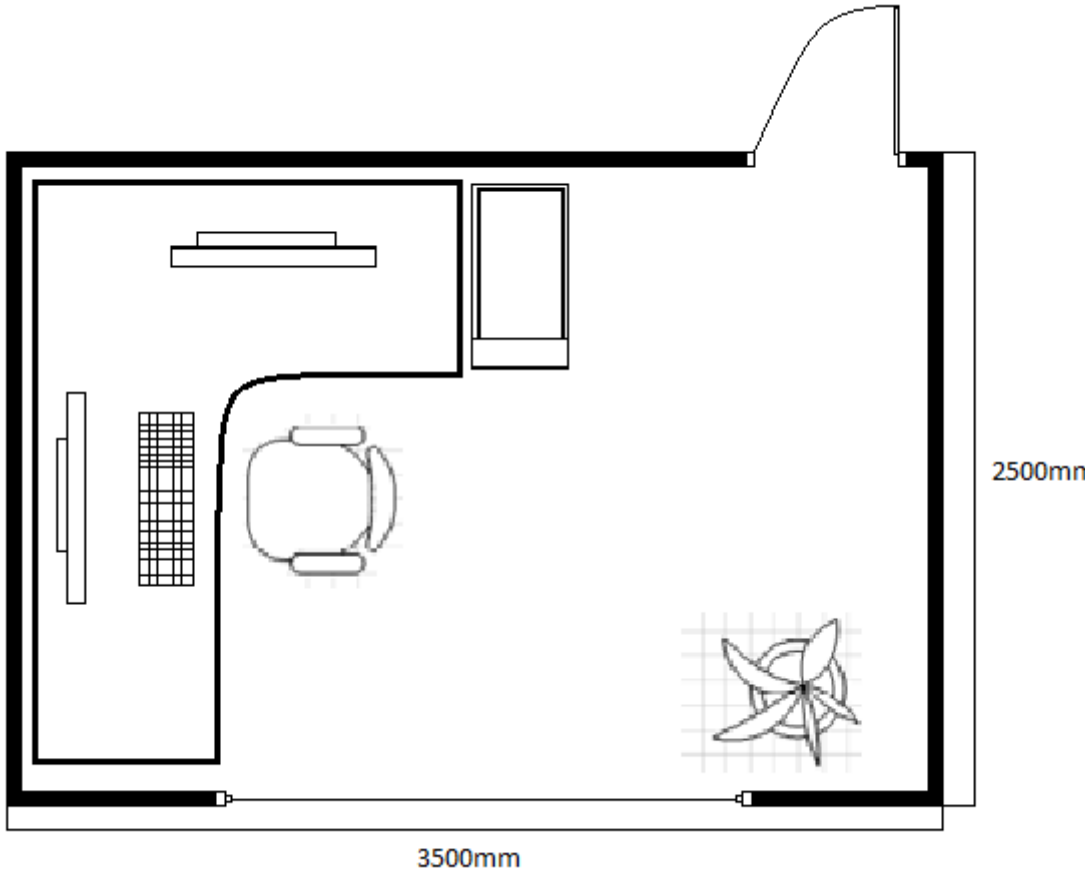

Рисунок 5.1 – План приміщення

Приміщення має довжину 3.5 м, ширину 2.5 м, висоту стелі 2,7 м. Кількість робочих місць - одне. Приміщення знаходиться на другому поверсі двохповерхової цегляної будівлі. Площа – 8.75 м^2, об'єм – 23.625 м^3. Виходячи з цього, отримаємо дані, наведені в таблиці 5.1. Нормативні значення згідно [21].

Таблиця 5.1 – Фактичні та нормативні значення параметрів приміщення

Параметр	Норма	Реальні параметри
Площа, S	не менше 6 м^2	8.75 м^2
Об’єм , V	не менше 15 м^3	23.625 м^3

Можна зробити висновок, що отримані показники відповідають існуючим нормам та вимогам.

Розглянемо тепер відповідність характеристик робочого місця нормативним. Для цього зведемо основні вимоги до організації робочого місця згідно [21] і відповідні фактичні значення для робочого місця, за яким виконується робота, у табл. 5.2:

Таблиця 5.2 - Характеристики робочого місця

Найменування параметра	Значення	
	Фактичне	Нормативне
Висота робочої поверхні, мм	700	680 - 800
Висота простору для ніг, мм	700	не менше 600
Ширина простору для ніг, мм	700	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	500	400 - 500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	500	не менше 300
Ширина опорної поверхні спинки, мм	400	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	700	700 - 800

Крісло є підйомно-поворотним, має підлокітники і можливість регулювання за висотою і кутом нахилу спинки. Екран монітору знаходиться на відстані 0.7 м. Таким чином, за всіма параметрами робоче місце відповідає нормативним вимогам.

З обладнання в приміщенні знаходиться 1 комп’ютер та 2 монітори ACER VH27IPS 27". На все обладнання є паспорт та інструкція з експлуатації

українською мовою. згідно з супроводжувальній документації обладнання відповідає стандартам України і його можна використовувати без загрози здоров'ю та життю працюючого.

5.2 Мікрокліматичні умови

Згідно [22] цю роботу можна віднести до категорії легка 1а. Джерелами тепла в цьому приміщенні є люди, електроустаткування, освітлювальні прилади в темний час доби і система опалювання зимою. Оператором виділяється до 120 Ккал теплової енергії за годину. Оптимальні та фактичні значення параметрів мікроклімату приведені в таблиці 5.3.

Таблиця 5.3 – Значення мікроклімату

Період року	Параметр	Оптимальний	Фактичний
Теплий	Температура	23 – 25 °С	23 °С
	Вологість	40 – 60 %	40 %
	Швидкість повітря	≤ 0.1 м/с	
Холодний	Температура	22 – 24 °С	22 °С
	Вологість	40 – 60 %	50 %
	Швидкість повітря	≤ 0.1 м/с	

Всі показники задовольняють вимогам зазначеним в [22] для робіт категорії легка 1а і є задовільними для здоров'я людини.

5.3 Освітлення

Природне освітлення здійснюється за допомогою вікна, площа якого складає S' = 2,5*1,5 = 3.75м^2 та являється боковим освітленням.

Штучне освітлення здійснюється за допомогою світлодіодних ламп білого світіння.

Робота за дисплеєм ПЕОМ за розрядом зорових робіт відноситься до III розряду. При загальному висвітленні освітленість робочого місця повинна становити від 200 до 400 лк.

При штучному освітленні нормуються:

- Е (лк) - найменша припустима освітленість;
- М - показник дискомфорту;
- Кп (%) - коефіцієнт пульсації освітленості;

Перевіримо відповідність нормам фактичних параметрів штучного освітлення в приміщенні. Номінальний світловий потік лампи білого світіння ЛБ-40 $Ф_л$ = 3120 лм.

У приміщенні застосовуються світильники, у яких встановлені 2 лампи.

Висоту підвісу світильника визначимо з формули :

$$h = H - h_c - h_p - h_n \quad (5.1)$$

Н - висота приміщення, м; h_c - висота світильника, м; h_n - відстань від стелі до підвісу, м; h_p - висота робочої поверхні, м.

Для розглянутого приміщення :

Н = 2,7 м; h_c = 0,2 м; h_n = 0,18 м; h_p = 0.7 м.

звідси :

$$h = 2,7 - 0,2 – 0,18 – 0,7 = 1,62 \text{ м.} \quad (5.2)$$

Світильники розташовані в 1 ряд. Відстань між світильниками 2 метра, відстань від ряду до стіни 0,8 метра. Приміщення має наступні габарити:

довжина А = 3.5 метрів,

ширина В = 2.5 метрів.

Визначимо освітленість у робочій точці. Для розрахунку загальної рівномірної освітленості при горизонтальній робочій поверхні використаємо метод коефіцієнта використання світлового потоку.

Розрахункова формула для світлового потоку світильника має вигляд:

$$Ф_л = \frac{E \cdot K_з \cdot S \cdot Z}{N \cdot n} \quad (5.3)$$

N - число світильників у приміщенні, N = 2*2=4;

n - коефіцієнт використання світлового потоку;

$Ф_л$ - світловий потік ламп;

$K_з$ - коефіцієнт запасу, $K_з$ = 1.5;

Z - коефіцієнт нерівномірності;

S - площа приміщення;

E - освітленість, створювана всіма світильниками.

Звідси одержуємо формулу для розрахунку освітленості на робочому місці :

$$E = \frac{Ф_л \cdot N \cdot n}{K_з \cdot S \cdot Z} \quad (5.4)$$

Коефіцієнт використання світлового потоку залежить від:

ККД, кривій розподілу сили світла світильника;

Коефіцієнта відбиття стелі Rc і стін Rc;

Висоти підвісу світильників $h_п$;

Показника приміщення і обчислимо за формулою:

$$i = \frac{A * B}{h * (A + B)} \quad (5.5)$$

i = (3.5 * 2.5) / (1,62 * (3.5 + 2.5)) = 0.9 (5.6)

Нам відомо, що стеля й стіни пофарбовані в світло-блакитний і світло-сірий кольори. Приймаємо:

Rп = 50%, Rс = 30%.

Звідси: n = 42%.

$$E = \frac{3120*4*0,42}{1,5*8.75*1,1} = 363.05 \quad (5.7)$$

Фактична освітленість на робочому місці становить 363.05 лк, що відповідає вимогам до III розряду зорової роботи. Можна зробити висновок, що для роботи з дисплеєм цілком достатньо існуючих джерел світла.

5.4 Шум і вібрація

Джерелом шуму в приміщенні є комп'ютер. Кулери комп'ютера є сучасними і мають низький рівень шуму за рахунок нового під шипника зі самостабілізуючимся тиском рідини (SSO2). Згідно технічній документації шум обумовлений кулером в блоці живлення складає 7-12 дБ, кулером процесора – 7-12 дБ, загальний, - 20 дБ. Враховуючи незначний рівень шуму від персонального комп'ютера та незначний рівень фонового шуму від іншого устаткування - сумарний рівень шумового забруднення приміщення не перевищує максимально допустимий рівень коригованої звукової потужності і складає не більше 50 дБ.

При роботі з персональним комп'ютером в робочому приміщенні значення характеристик вібрації на робочих місцях не повинна перевищувати допустимих значень.

5.5 Випромінювання

У приміщенні відсутні інфрачервоні, ультрафіолетові та електромагнітні випромінювання, бо усі монітори ПК вироблені на основі OLED матриці, підсвітка якої здійснюється за рахунок органічних світлодіодів, що не має сильного електромагнітного випромінювання і сертифіковані в Україні.

5.6 Оцінка умов безпеки праці

5.6.1 Вимоги електробезпеки

Технічні заходи із запобігання електротравм від контакту з нормально струмовідними елементами електроустаткування

- електромережа в приміщенні розведена в спеціальних каналах стін і підлоги.
- величина напруги мережі 380^{x}220В (міжфазна лінійна і фазна);
- всі нормально струмовідні елементів (в першу чергу електричні дроти) вкриті ізоляційними матеріалами;
- в джерелі безперебійного живлення персонального комп'ютера використовується механічне захисне блокування, що забезпечує вимикання напруги при його відкриванні;

Можна зробити висновок, що дане приміщення задовольняє нормам електробезпеки для встановлення ПК.

5.6.2 Оцінка пожежної безпеки приміщення

У приміщенні, що розглядається, можуть горіти: вироби з дерева, пластмас, тканини і паперу, ламінат. Горючі рідини, пил та волокна у приміщенні не використовуються і не виділяються, тому воно відноситься, відповідно до нормативної документації, до зони П-IIа [23] і до категорії пожежної небезпеки В.

Ймовірними причинами виникнення пожежі можуть бути несправна робота електрообладнання (кабелів, розеток), короткі замикання внаслідок виходу з ладу чи експлуатації несправного електроустаткування (ПЕОМ, периферійних пристроїв), порушення правил протипожежної безпеки тощо.

Експлуатація ліній електромережі практично повністю унеможливлює виникнення електричного джерела загоряння в наслідок короткого замикання

та перевантаження проводів, а автоматичні вимикачі миттєво відключать ділянку мережі у разі перевантаження чи замикання. Застосовуються дроти з важкогорючою і негорючою ізоляцією. Блок живлення комп'ютеру оснащений автоматичним вимикачем у разі перевантаження чи стрибка напруги у мережі.

Комплекс заходів для своєчасного попередження пожеж у приміщенні та оперативного реагування у разі їх виникнення:

- заборона використання відкритого вогню у приміщенні;

- автомати на кожній ланці мережі для запобігання перевантаження та замикання;

- обов'язковий інструктаж персоналу з питань охорони праці,

- зокрема, правила пожежної безпеки у приміщеннях з ЕОМ;

- наявність системи автоматичної пожежної сигналізації з димовими пожежними оповіщувачами;

- ступінь вогнестійкості будівлі, у якій розташовано приміщення – II;

- наявність шляхів евакуації при виникненні пожежі;

- призначення особи, відповідаючої за пожежну безпеку;

- розміщення схеми евакуації людей при пожежі і ознайомлення з нею персоналу.

Приміщення має один вихід, оскільки в ньому працює менше 25 чоловік. Будинок має два виходи – головний і запасних. Коридор між приміщеннями має два виходи на різні сходи, одні з яких ведуть до головного виходу, а другі - до спеціального евакуаційного виходу.

Для гасіння пожежі кожна кімната обладнана ручними вуглекислотними вогнегасниками ВВК-1,4 [24]. У загальному коридорі встановлені пінні вогнегасники ВВП. Призначена особа, що відповідає за дотримання персоналом вимог пожежної безпеки. Розроблено план евакуації персоналу і найбільш коштовного устаткування (майна). Співробітники ознайомлені з порядком і планом евакуації під розпис.

Можна зробити висновок, що шляхи евакуації з приміщення повністю відповідають нормам.

5.7 Висновки

Аналіз умов праці в розглянутому робочому приміщенні показав, що умови праці з ПЕОМ відповідають вимогам , оскільки площа та об’єм не менше нормативних значень, рівні шуму, вібрації і загазованості не перевищують нормативних обмежень.

Для підтримання параметрів мікроклімату в приміщенні встановлено радіатор центральної водяної системи опалення, що складається з 7 секцій.

Ергономіка робочого місця і режим зорової роботи задовольняють вимогам і сприяють зниженню втоми.

ВИСНОВКИ

На даний момент іде активна розробка і вдосконалення технології «Розумний» будинок. Незважаючи на це, процеси стандартизації та глобалізації почалися порівняно недавно й на даний час існує безліч однотипних рішень, але з різною проприетарной реалізацією, зав'язаною на виробників та їх технології. Завдяки зацікавленості основних гравців ринку в розвитку системи «Розумний» будинок та інтеграції в неї своїх пристроїв і сервісів, з'явилися зрушення у бік популяризації технології та спільна зацікавленість у ній як покупців, так і виробників.

Незважаючи на певний перелік недоліків, «Розумний» будинок з високою вірогідністю займе своє місце у сфері побутової електроніки і перейде з розряду рідкостей в повсякденність. Цьому сприяють і появи простих і функціональних контролерів сімейства Arduino та підвищення обчислювальної потужності мобільних пристроїв.

Отже, «Розумний» будинок надає своїм користувачам такі переваги як: підвищення комфорту проживання, єдиний інформаційний простір, автоматизація рутинних процесів. Для виробників, крім комерційної вигоди, це також спосіб залучити користувачів в свою екосистему. Однак поки «Розумний» будинок так і залишається системою для багатих або ж технічно підготовлених людей. Незважаючи на позитивні тенденції в сфері, залишилася потреба в недорогому і вільному варіанті реалізації, який кожен зможе налаштувати під свої потреби не володіючи технічними навичками і який зможе працювати з більшістю вже створених пристроїв і контролерів.

У даній дипломній роботі було спроектовано: додаток-сервер для роботи з Ардуіно та проведено ряд обчислень, спрямованих на підвищення стабільності і безпеки системи. Також було створено програму-клієнт під мобільну операційну систему Android, щоб надати користувачам простий і візуально зрозумілий інтерфейс управління сервером, з метою реалізації

режиму мультикористування. У ході роботи були проведені дії щодо інтеграції в додатки сервісу хмарного зберігання даних з метою забезпечення бесзбойної роботи системи. Крім того, для зменшення енерговитрат були створені розвинуті засоби прогнозу енергоспоживання на основі даних погодних сервісів і нейронної мережі. Також були проведені роботи з оптимізації інтерфейсу і локалізації додатків для більшості мов.

Для впровадження системи рекомендовано мати Arduino Mega, пристрій на Android 4.0+, набір сенсорів Sensor Kit. Для управління системою необхідно мати смартфон на Android 4.0+, рекомендовано – на Android 5.0+.

Результати даної роботи можуть бути використані для розгортання системи з контролем кліматичних параметрів і енергоспоживання як в звичайному будинку, так і в спеціалізованих приміщеннях з більш жорсткими вимогами до мікроклімату середовища (серверні, аранжереі, тощо).

У подальшому планується додати до системи сенсор електроспоживання та вдосконалити спільний режим користування. Також ПЗ серверу буде переписане на Java, щоб у якості серверу можна було використовувати не лише пристрої з Android ОС, а й персональні комп’ютери.

ПЕРЕЛІК ПОСИЛАНЬ

1. Умный дом– Режим доступу : http://www.directinfo.net/index.php?option=com_content&view=article&id=139%3A2010-07-06-13-57-09&catid=1%3A2008-11-27-09-05-45&Itemid=84&lang=ru - Дата доступу : 14.05.2013.

2. Система умный дом – технология экономии, удобства и комфорта высокого уровня. – Режим доступу : http://smarton.com.ua/smart_home/systema_umniy_dom_intro - Дата доступу : 14.05.2013.

3. Система умный дом. – Режим доступу : http://intelcity.com.ua/comfort_house - Дата доступу : 14.05.2013.

4. Виртуальная ревльность в шлеме Microsoft. – Режим доступу : http://www.bbc.com/russian/multimedia/2015/01/150122_microsoft_helmet - Дата доступу : 14.05.2013.

5. Кто отвечает за климат-контроль в доме?– Режим доступу : http://smarton.com.ua/smart_home/klimat_kontrol_v_umnom_dome - Дата доступу : 14.05.2013.

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

7. 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/ - Дата доступу : 14.05.2013.

8. Orvibo Allone Wi-fi – Режим доступу : http://www.geekbuying.com/item/Orvibo-Allone-WiFi---IR---RF-Remote-Control-Smart-Home-Automation-Rechargeable-Battery-for-IOS-Android-Mobile-338990.html - Дата доступу : 14.05.2013.

9. Android – Режим доступу : http://www.3dnews.ru/581873 - Дата доступу : 16.05.2013.

10. Полный обзор изменений в Android M. – Режим доступу : https://www.iguides.ru/main/gadgets/google/android_m_review/- Дата доступу : 16.05.2013.

11. Фрагментация Android – Режим доступу : http://habrahabr.ru/post/188738/ - Дата доступу : 16.05.2013.

12. Android Fragmentation Vizualized – Режим доступу : http://opensignal.com/reports/2014/android-fragmentation/#android_version_timeseries - Дата доступу : 24.05.2013.

13. Android Studio. – Режим доступу : https://developer.android.com/sdk/index.html - Дата доступу : 24.05.2013.

14. Хмарні сховища даних – Режим доступу : http://oksim.com.ua/index.php/167-khmarni-skhovishcha-danikh - Дата доступу : 24.05.2013.

15. SharedPreferences. – Режим доступу : http://developer.android.com/reference/android/content/SharedPreferences.html - Дата доступу : 24.05.2013.

16. AssyncTask. – Режим доступу : http://developer.android.com/reference/android/os/AsyncTask.html - Дата доступу : 28.05.2013.

17. OpenWeather Map – Режим доступу : http://openweathermap.org/ - Дата доступу : 28.05.2013.

18. Material Design Introduction. – Режим доступу : https://www.google.com/design/spec/material-design/introduction.html - Дата доступу : 28.05.2013.

19. Основние принципы Material Design. – Режим доступу : http://habrahabr.ru/post/248653/ - Дата доступу : 28.05.2013.

20. RecyclerView и CardView. Новые виджеты в Android L – Режим доступу : http://habrahabr.ru/post/237101/ - Дата доступу : 28.05.2013.

21. Creating a Navigation Drawer – Режим доступу : https://developer.android.com/training/implementing-navigation/nav-drawer.html - Дата доступу : 28.05.2013.

22. Ткачук К.Н., Зацарний В.В. та ін. Охорона праці та промислова безпека. Навчальний посібник. – К.: Лібра, 2010. – 559 с.

23. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98 (затверджено Постановою Головного державного санітарного лікаря України від 10.12.1998 р. № 7).

24. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень [Текст] / К., 2000.- 16 с.

25. Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою. НАПБ Б.03.002-2007. (затверджено наказом МНС України від 03.12.2007 № 833)

26. Типові норми належності вогнегасників (затверджено наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 2 квітня 2004 р. N 151)

27. Природнє і штучне освітлення : ДБН В.2.5-28:2006. – К. : Міністерство будівництва, архітектури та житлово-комунального господарства України, 2006. – 68 с. – (Національні стандарти України).