МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
Литвиненко О. Є.

«___» _____2021 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО–КВАЛІФІКАЦІЙНОГО РІВНЯ "БАКАЛАВР"

Тема: Мобільний додаток дистанційного керування електроприладом

Виконавець: Мартиненко О.В.

Керівник: Кучеров Д.П.

Нормоконтролер: Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
Освітньо-кваліфікаційний рівень бакалавр
Спеціалізація 6.050102 "Системне програмування"
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Литвиненко О. Є.
«_____»_____2021 р.

ЗАВДАННЯ
на виконання дипломної роботи (проекту)
Мартиненку Олександру Вікторовичу
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема проекту (роботи):Мобільний додаток дистанційного керування електроприладом
затверджена наказом ректора від "04" лютого 2021 року № 135 /ст.
2. Термін виконання проекту(роботи):з 17 травня 2021 р. по 20 червня 2021 р.
3. Вихідні дані до проекту (роботи):інформація про електроприлади , інформація про методи дистанційного керування, мова програмування високого рівня, *Microsoft Word*

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1) доцільність розробки програмного модуля
2) методи дистанційного керування технічними системами
3) проектування та розробка програмного модуля для дистанційного керування електроприладом
4) тестування програмного модуля для дистанційного керування електроприладом

5. Перелік обов'язкового графічного матеріалу:
1) структура програмного модуля;
2) блок-схема алгоритму керування технічним засобом;
3) діаграма компонентів додатку дистанційного керування електроприладом;
4) екранна форма інтерфейсу користувача;
5) приклад тест-кейсу;

6. Календарний план

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1	Ознайомлення з постановкою задачі на дипломний проект	17.05.2021-18.05.2021	
2	Вивчення спеціальної літератури і технічної документації	18.05.2021-19.05.2021	
3	Аналіз основних завдань та цілей проектування	19.05.2021-21.05.2021	
4	Підготування розділу 1.	24.05.2021-26.05.2021	
5	Проаналізувати сучасні методи і алгоритми реалізації програмних засобів	25.05.2021-26.05.2021	
6	Підготувати розділ 2.	27.05.2021-31.05.2021	
7	Підготувати розділ 3	31.05.2021-03.06.2021	
8	Провести моделювання програмного модулю, оцінити його ефективність	03.06.2021-05.06.2021	
9	Оформити пояснювальну записку та пройти нормоконтроль	06.06.2021-10.06.2021	
10	Підготовити графічний демонстраційний матеріал	10.06.2021-14.06.2021	

7. Дата видачі завдання: «_04_»___Лютого___ 2021 р.

Керівник дипломного проекту _____ Кучеров Д.П,
(підпис)

Завдання прийняв до виконання _____ Мартиненко О.В.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Мобільний додаток дистанційного керування електроприладом»: 58 сторінок, 15 рисунків , 1 таблиць, 18 використаних джерел.

ДИСТАНЦІЙНЕ КЕРУВАННЯ ЕЛЕКТРОПРИЛАДАМИ , МОБІЛЬНИЙ ДОДАТОК , ТЕХНОЛОГІЇ КЕРУВАННЯ ЕЛЕКТРОПРИЛАДАМИ, АЛГОРИТМ РОБОТИ ДОДАТКУ , ДІАГРАМИ ПРОЕКТУВАННЯ ДОДАТКУ.

Об’єкт дослідження дипломної роботи – дистанційні технології керування електроприладами.

Предмет дослідження дипломної роботи - мікроконтролерні системи та додатки для дистанційного керування електроприладами.

Мета дипломної роботи – створення сприятливого для кінцевого користувача мобільного додатку для дистанційного керування електроприладом на базі мікроконтролера для зручного дистанційного керування побутовими електроприладами.

Методи дослідження - опрацювання необхідного літературно-довідкового матеріалу , методи проектування , розробки та тестування програмного модуля для дистанційно керованих технічних систем , огляд сучасних технологій та методів дистанційного керування для створення дистанційно-керованих програмно-апаратних технічних засобів , проведення експерименту.

Проведено аналіз доцільності розробки програмного модуля; розглянуто методи дистанційного керування електроприладом; досліджено підходи до результативного проектування , розробки та тестування програмного модуля для дистанційного керування електроприладом; під час виконання експерименту було проаналізовано та протестовано результати отримані під час виконання роботи та зручність використання створеної програмно-апаратної системи.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ…… 7

ВСТУП……..8

РОЗДІЛ 1 АКТУАЛЬНІСТЬ ВИКОРИСТАННЯ ДИСТАНЦІЙНОГО КЕРУВАННЯ ЕЛЕКТРОПРИЛАДАМИ……10

1.1. Основні поняття у сфері систем дистанційного керування……10

1.2. Аналіз актуальності використання та розгляд сучасних засобів дистанційного керування електроприладом……14

1.3. Постановка завдання проєктування ……22

1.4. Висновки за розділом……24

РОЗДІЛ 2 ПРОГРАМНО-АПАРАТНА СИСТЕМА ТА ТЕХНОЛОГІЇ ДИСТАНЦІЙНОГО КЕРУВАННЯ……..25

2.1. Протоколи дистанційної взаємодії……..25

2.2. Структура програмного модуля……..29

2.3 Огляд інструментальних засобів розробника……34

2.4. Висновки до розділу……37

РОЗДІЛ 3 ПРОЕКТУВАННЯ ПРОГРАМНО-АПАРАТНОЇ СИСТЕМИ…….38

3.1. Системні вимоги……...38

3.2. Модель інтерфейсу……...40

3.3. Програмний код керування електроприладом……43

РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА……..44

4.1. Тестування програмної системи……..44

4.2. Висновки до розділу……47

Висновки……48

Список бібліографічних посилань використаних джерел……49

Додаток А Код модуля *Main Activity*……..52

Додаток Б Код модуля *Adapter List*……..58

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

МК – мікроконтролер

ЦП – центральний процесор

ШІМ – широтно-імпульсні модулятори

ОС – операційна система

ПЗ – програмне забезпечення

ООП – об’єктно-орієнтоване програмування

ОЗП – оперативний запам’ятовуючий пристрій

ПЗП – постійний запам’ятовуючий пристрій

ДК – дистанційне керування

IoT – internet of things (технологія інтернету речей)

ІК – інфрачервоне керування

ISM – industry, Science and Medicine (діапазон радіозв’язку промисловість , безпека та медицина)

FHSS – frequency Hopping Spread Spectrum (метод розширення спектру з стрибкоподібним переналаштуванням робочої частоти)

LMP – link management protocol (протокол керування посиланнями)

L2CAP – logical link control and adaptation protocol (протокол передачі по одному каналу різних потоків даних та керування логікою роботи)

SDP – service discovery protocol (протокол виявлення функцій)

HCI – host/controller interface (інтерфейс взаємодії пристрою керування з керованими пристроями)

RFCOMM – radio frequency communications (протокол радіозв’язкової взаємодії)

TCS – telephony control protocol (протокол контролю телефонії)

JVM – java virtual mashine (віртуальна машина платформи програмування java)

SDK – software development kit (набір засобів розробки програмного забезпечення)

ВСТУП

На сьогоднішньому етапі розвитку комп’ютерної техніки одним з найвизначніших досягнень інженерної думки є мікроконтролери.

Мікроконтролери використовуються у всіх сферах життєдіяльності людини, пристроях, технічних засобах та програмно-апаратних системах які оточують її. Простота підключення і великі функціональні можливості а також доступна ціна для придбання та відносна проста в програмуванні та налаштуванні при створенні певних технічних систем які створюються на базі мікроконтролерів дають можливості для вирішення великого списку практичних завдань апаратної техніки.

На основі практичного прикладу дуже легко показати переважні характеристики використання мікроконтролерів, необхідності їх впровадження в різні пристрої.

Сьогодні прийнято вважати що мікроконтролер (МК) - це комп'ютер, який топологічно розміщенно одній мікросхемі. Звідси і його основні привабливі якості: малі габарити високі продуктивність, надійність і здатність бути адаптованим для виконання самих різних завдань.

Мікроконтролер крім центрального процесора (ЦП) містить пам'ять і численні пристрої введення / виводу: аналого-цифрові перетворювачі, послідовні і паралельні канали передачі інформації, таймери реального часу, широтно-імпульсні модулятори (ШІМ), генератори програмованих імпульсів і т.д. Основне призначення мікроконтролеру - використання в системах автоматичного управління, вбудованих в самі різні пристрої: переважна частина електроприладів , що , існують в світі , кредитні картки, фотоапарати, мобільні телефони, музичні центри, телевізори, відеомагнітофони та відеокамери, пральні машини, мікрохвильові печі, системи охоронної сигналізації, системи запалювання бензинових двигунів, електроприводи локомотивів , ядерні реактори і багато, багато іншого. Системи керування, що стали настільки масовим явищем, що фактично сформувалася нова галузь

діяльності , що існує на стику економіки та інформаційних технологій , яка отримала назву *Embedded Systems* (вбудовані системи).

Зазвичай застосування МК при створенні програмно – апаратних технічних систем поділяють на два етапи: перший - програмування, коли користувач розробляє програму і виконує процес «прошивання» її безпосередньо в кристал, і другий - узгодження спроектованих виконавчих пристроїв з запрограмованим МК. Значно полегшують налагодження програми на першому етапі - симулятор, який наочно моделює роботу мікропроцесора. На другому етапі для налагодження використовується вбудований в схему емулятор, який є складним і дорогим пристроєм, часто недоступним пересічному користувачеві.

МК це комп'ютер на одній мікросхемі. Основним призначенням якого є управління різними електронними пристроями виконання алгоритму взаємодії між якими виконується відповідно з закладеною в мікроконтролер програмою. На відміну від мікропроцесорів, які використовуються в персональних комп'ютерах, мікроконтролери містять вбудовані додаткові пристрої. Ці пристрої виконують свої завдання під керуванням мікропроцесорного ядра мікроконтролера та певної архітектури взаємодії між собою в рамках одного технічного пристрою.

В наш час мікроконтролери можна зустріти у величезній кількості у сучасних промислових і побутових приладах: верстатах, складних засобах праці з цифровим програмним керуванням , автомобілях, телефонах, телевізорах, холодильниках, пральних машинах і навіть кавоварках.

Існує декілька видів управління такими пристроями , так як архітектурно мікроконтролери підтримують різні інтерфейси взаємодії з керуючими пристроями. Наприклад технології дистанційної передачі інформації такі як *Wi-fi* та технологія бездротової передачі інформації *Bluetooth* або бездротовий стандарт передачі даних *ZigBee.*

Сьогодні такі стандарти та технології дистанційного зв’язку *Bluetooth* та *Wi-fi* є базовими у сфері бездротової передачі інформації або дистанційного керування різного виду електроприладами , що розроблені та виконанні на базі МК. Але кожна з цих технологій має свої позитивні та негативні сторони , але це

не заважає їм бути дуже розповсюдженими та затребуваними у сучасному світі , та бути популярно використовуваними у сфері приладобудування , а саме мають величезну популярність при створенні систем управління цими пристроями , що підтримують дистанційний інтерфейс взаємодії через який і виконується керування електроприладами такого типу , а особливо електроприладами на базі МК , які мають вбудовані адаптери дистанційної взаємодії завдяки своїм архітектурним особливостям та корисному функціоналу , який був реалізований завдяки таким технічним пристроям.

В даній роботі реалізовано варіант додатку управління електроприладом який взаємодіє з МК засобами інтерфейсу взаємодії технології дистанційної передачі даних *Bluetooth* , який виступає каналом зв'язку керування між мобільним телефоном та пристроєм , що побудований на базі промислового МК ESP32 та має вбудований мікрочіп для роботи з *Bluetooth* технологією.

Саме бездротова технологія *Bluetooth* дає змогу зручно для кінцевого користувача використовувати в побуті МК та пристрої, що побудовані з їх використанням. *Bluetooth* - це сучасна технологія бездротової передачі даних, яка дозволяє поєднати один з одним практично будь-які пристрої: мобільні телефони, ноутбуки, принтери, цифрові фотоапарати і навіть холодильники, мікрохвильові печі, кондиціонери. З'єднати можна все, що з'єднується (тобто має вбудований мікрочіп *Bluetooth*). Технологія стандартизована, отже, проблеми несумісності пристроїв від різних виробників бути не повинно.

Bluetooth - це маленький чіп, що є високочастотним передавачем (2.4 - 2.48 ГГц) та має приймач, що працює в діапазоні *ISM* (*Industry, Science and Medicine;* промисловий, науковий і медичний). Використання цього частотного діапазону у побуті не потребує якихось спеціальних дозволів , або ліцензій , тому часто використовується в реальному житті.

Технологія *Bluetooth* використовує невеликі прийомо - передавачі малого радіусу дії, або безпосередньо вбудовані в пристрій, або , що підключаються через вільний порт або PC-карту. Адаптери працюють в радіусі 10 метрів і, на відміну від технології бездротової передачі інформації *IrDA*, не обов'язково повинні перебувати в зоні прямої видимості, тобто, між пристроями, що

сполучаються можуть бути різні перешкоди , це не буде впливати на повноту та якість інформації , що передається по радіоканалу , який використовує в роботі дана технологія.

Також в результаті виконання дипломної роботи (проєкту) , було отримано мобільний додаток для дистанційного керування електроприладом , який створено за допомогою мови програмування *Java* з використанням комплексу підходів до створення програмного забезпечення парадигми проектування ПЗ , який має назву ООП та його використання у створені застосунку для мобільної операційної системи *Android* за допомогою інтегрованого середовища розробки програмного забезпечення – *Android studio*, а саме мобільних застосунків для виконання на пристроях які підтримують мобільну ОС *Android* певної версії для коректного виконання та роботи застосунку керування мікроконтролером на базі якого створено пристрій , що є радіоприймачем на базі аналогового компоненту для прийому радіохвиль та передачі отриманого сигналу для обробки на сам МК та звуковідтворення за допомогою допоміжного пристрою – динаміка , що під’єднано до пристрою через дротовий канал передачі звуку а саме роз’єм *AUX*.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ АКТУАЛЬНОСТІ ПРОЄКТУ

1.1. Основні поняття у сфері систем дистанційного керування

Дистанційне керування (ДК) – це передача керуючого впливу або сигналу чи імпульсу від керуючої системи , чи конкретної особи – оператора до об’єкта керування , що знаходиться на певній відстані або якщо при керуванні певною системою чи технічним пристроєм немає можливості передавати сигнал напряму , якщо об’єкт управління змінює своє розташування у просторі.

Зазвичай ДК складається із передавача – пульту дистанційного керування та приймача який надалі може віддавати керуючі сигнали всій конкретній системі , та виконавчих механізмів , та інших складових систему керування якою відбувається.

Системи ДК мають можливість використовувати різні канали зв’язку.

Канал зв’язку для виконання механічного керування – це канал зв’язку який використовується у випадку коли об’єкти віддалені один від одного на невелику відстань або якщо треба забезпечити миттєву , не спотворену різними видами впливу реакцію систему над якою здійснюється процес керування.

Радіоканал – канал зв’язку при передачі сигналів по якому використовується принцип дистанційного керування при якому керуючий вплив та зворотній зв’язок керуємої системи відбувається через радіоканал за допомогою радіохвиль , тобто принципом вільного поширення у просторі електромагнітних хвиль під час якого передаючий пристрій та приймач можуть не знаходитись в зоні прямого контакту.

Ультразвуковий канал – це канал зв’язку що функціонує за принципом передавання звукових хвиль певного діапазону в системах принципу роботи передавач – приймач , та використовується для керування мобільними та стаціонарними об’єктами , але на суттєво обмеженій відстані від передавачем керуючого ультразвукового імпульсу та приймачем системи над якою здійснюється керування.

Канал випромінювання світла інфрачервоного спектру – канал зв’язку робота якого заснована на принципі фізичного явища випромінювання електромагнітного випромінювання , що займає спектральну область між червоним кінцем видимого людським оком світла та коротким електромагнітним, мікрохвильовим та радіовипромінюванням.

В сучасних побутових електроприладах логіка функціонування зав’язана на використанні керуючих пристроїв – мікроконтролерах які виконують роль інтерфейсу між кінцевим користувачем електроприладу та його керуючим пристроєм зазвичай мобільним телефоном який в сучасному світі є надійним помічником кожної людини та має в собі всі необхідні користувачеві функції для взаємодії з навколишнім світом за допомогою програмних додатків функціонал яких і дає змогу виконувати дистанційне керування побутовими електроприладами та здійснює цей процес по радіоканалу який використовують вбудовані в мобільний телефон чіпи через які виконується з’єднання між мобільним телефоном та іншими пристроями за допомогою бездротових дистанційних технологій передачі інформації *Wi-fi* або *Bluetooth.*

На сьогодні мікроконтролери являються однією з найважливіших складових великої кількості сучасної побутової та виробничо-промислової електронної техніки. Звичайний мікроконтролер представляє собою мікросхему

для керування електронними пристроями будь-якої архітектури та складності виконання. В своїй архітектурі мікроконтролер це однокристальний цифровий обчислювальний пристрій , що може мати на одному кристалі функціонал процесора – обчислювального модуля , різних допоміжних переферійних пристроїв наприклад таких як універсальні цифрові порти , вводу-виводу інформації , різні інтерфейси комунікації такі як *UART, USB, Ethernet*, тощо , різні запам'ятовуючі пристрої такі як ОЗП та ПЗП, але такого набору компонентів цілком достатньо для виконання задач простого – середнього рівня складності.

Завдяки використанню сучасних мікроконтролерів користувачеві доступні функції дистанційного керування різною виконавчою технікою через використання інтерфейсів керування таких як радіоканали , *Wi-fi , Bluetooth* та *ZigBee*. В цьому випадку виконавчими пристроями можуть виступати:

- освітлювальні пристрої;
- системи сигналізації та відеоконтролю;
- системи водопостачання та вентиляції;
- підйомні механізми або ворота;
- електронні комплексні системи безпеки;
- будь які пристрої що мають інтерфейс дистанційного зв'язку та знаходяться в важкодоступних місцях з відсутністю можливості організації дротового каналу зв'язку;

Управління даними пристроями здійснюється за допомогою мобільного телефону через встановлений програмний додаток що використовує вбудовані в мобільний телефон інтерфейси бездротової комунікації такі як *Wi-fi* або *Bluetooth.*

Рис.1.1. Дистанційна взаємодія мобільного пристрою з різними виконавчими кінцевими пристроями

Керування мікроконтролером може виконуватись у різних режимах роботи:

– імпульсному режимі роботи;
– дистанційно – релейному режимі роботи;

Імпульсний режим роботи – це коли управління виходами та вбудованим у мікроконтролер реле здійснюється імпульсним сигналом. Тобто сигнал від пристрою передавача поступає на приймач у мікроконтролері у вигляді короткого імпульсу. Режим роботи може бути одно або двонаправленим. При двунаправленому режимі стан вихідних релейних контактів можна контролювати за допомогою вбудованого світлового сигнального індикатора.

Дистанційно – релейний режим роботи – це коли керування релейними контактами здійснюється неперервним сигналом , тобто передача керуючого сигналу з пристрою керування відбувається постійно у часі під час чого реле замкнуто на протязі всього часу поки контакт знаходиться в стані замкнення відповідного каналу пристрою з якого відбувається передача сигналу керування , в процесі роботи відбувається передача сигналу кожні 80 секунд. Цей сигнал обновлює стан входів для синхронізації з виходами приймаючого сигнал пристрою вбудованого у мікроконтролер. Цей режим має більш високий захист від зовнішніх впливів або втрати живлення. Контроль за станом виходів відбувається в реальному часі. Управління відбувається в ручному або

автоматично запрограмованому режимі від зовнішніх сигналопередаючих пристроїв керування.

1.2. Аналіз актуальності використання та розгляд сучасних засобів дистанційного керування електроприладами

Сучасний світ , що оточує нас – це світ інтернету речей. Різного роду пристрої мають можливість комунікації між собою , вони зв’язуються , утворюючи мережі які об’єднуються один з одним та з глобальною комунікаційною мережею інтернет. Технологія інтернету речей та технології дистанційного керування електроприладами має доцільний напрямок використання у великому переліку сфер людської діяльності. Наприклад такі предметні області як домашній побут як засіб для того щоб зробити оселю сучасної людини більш зручнішою та комфортнішою , використання даних технологій також користується популярністю в промисловому виробництві та медицині. Але сфера впливу постійно та стрімко зростає. Великі світові технологічні компанії такі як *Google, Samsung, Apple, Intel, Qualcomm* займаються розробкою власних продуктів заснованих на використанні технології інтернету речей , створення сучасних та якісних електроприладів та спрощення виробництва та використання для кінцевого користувача систем керування даними електроприладами.

Згідно з дослідженням міжнародної консалтингової компанії *International Data Corporation* розподіл ринку інформаційних технологій збільшить рівень коштів що циркулюють на ринку інтернету речей с 1.9 трильйонів доларів США у 2014 році і за прогнозами *International Data Corporation* досягне 7 трильйонів доларів США у 2021 році.

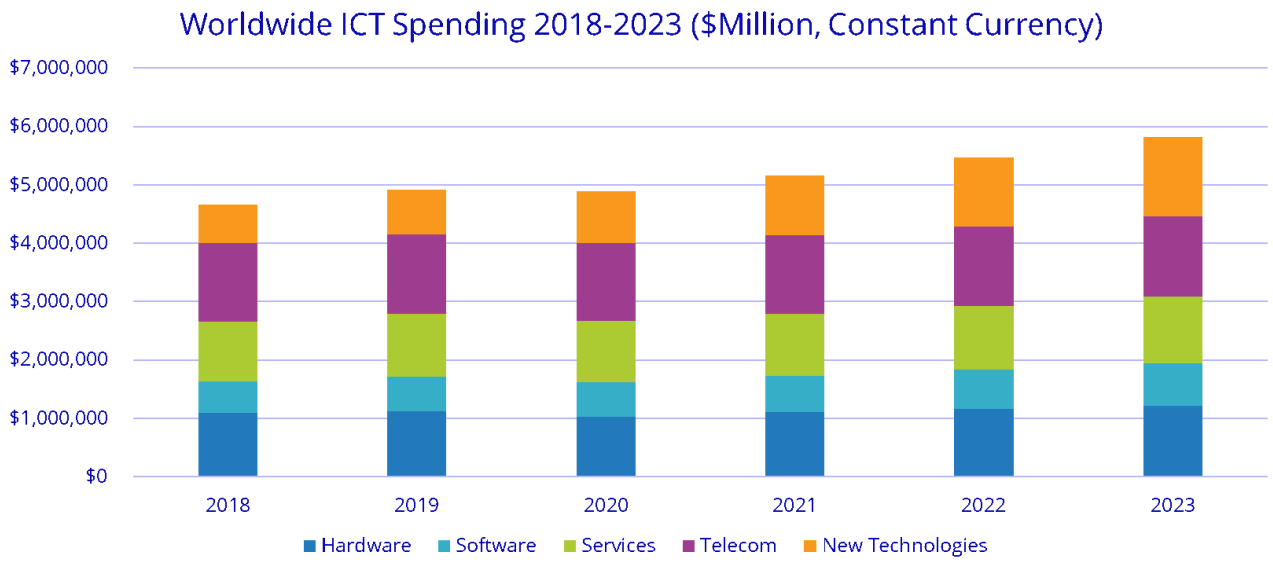

Рис.1.2. Загально-світовий оборотний об'єм ринку IoT з сегментами розподілу технологій до 2023 року

А за звітом аналітичного відділу компанії *Gartner* у 2020-му році кількість пристроїв , що використовуються у сфері технології інтернету речей та підключених до глобальної комунікаційної мережі інтернет налічує 10 мільярдів пристроїв.

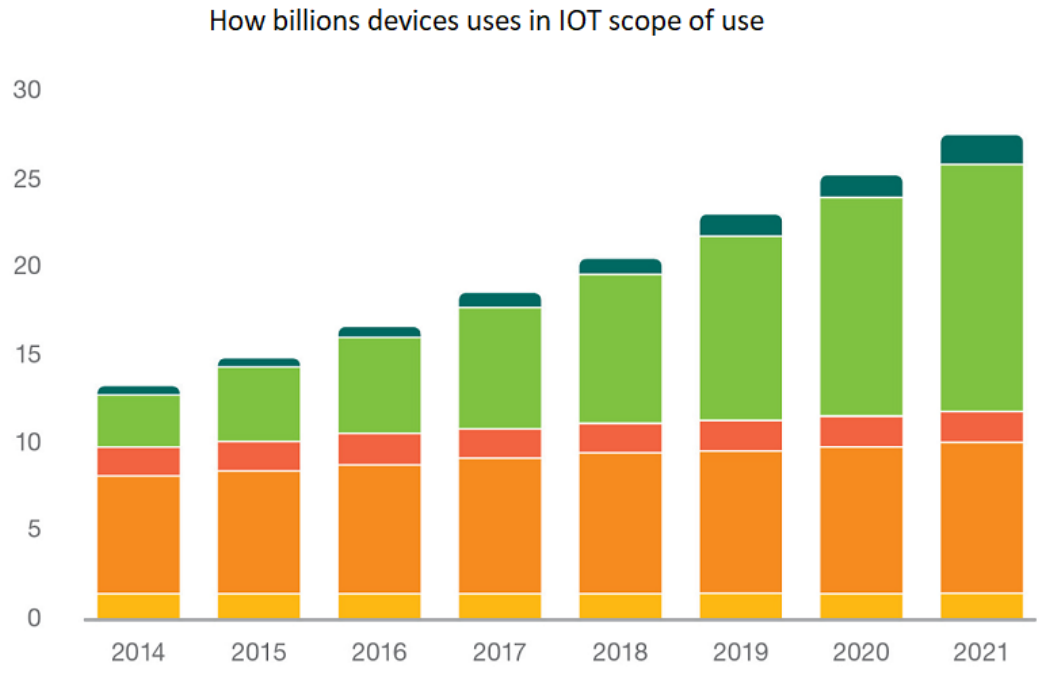

Рис.1.3. Графік загальної кількості пристроїв , що використовуються у сфері інтернету речей

Саме дякуючи великій розповсюдженості та доступності сьогодні кожна охоча людина має вільний доступ до використання технологій інтернету речей та зручного дистанційного керування побутовою електроапаратурою у повсякденному житті , тому ми маємо вільний доступ до «інтелектуальних» автомобілів , помешкань , побутової техніки та предметів щоденного використання.

Зацікавленість ринку мобільних додатків для смартфонів є еквівалентною кількості пристроїв що підключені до технології інтернету речей , тому як смартфони виступають надійною та зручною системою керування електроприладами для кінцевого користувача тому , що вони виступають зв'язуючою ланкою між кінцевим користувачем та пристроєм так як через них і виконується керування речами. Розвиток IoT та галузі розробки мобільних застосунків мають на меті створення доступу великому сегменту користувачів до широкої кількості можливостей та переваг над стандартними системами керування різного роду пристроїв. Саме тому галузь IoT стала однією з пріоритетних напрямків розробки мобільних додатків за останні роки.

Користувацькі мобільні застосунки в загальному випадку можна розділити на дві великі за призначенням групи:

– додатки для накопичування та аналізу даних;
– додатки для керування;

Основне завдання першої групи додатків це зняття показань з пристроїв та їх зберігання в пам'яті додатку до цієї групи можна віднести такі додатки як:

– додатки для фітнес-трекєрів;
– ваг;
– вимірювачів атмосферної вологості;
– камер та пристроїв що застосовуються у системах безпеки;
– пожежних сигналізаціях;
– різних датчиків та іншої вимірювальної техніки;

Друга група додатків спеціалізується на слідкуванні за пристроями , зніманню повної інформації з нього а також зміна його робочого стану тобто виконання процесу керування пристроєм.

До цієї групи належать додатки за допомогою яких здійснюється повне керування пристроями:

– додатки для керування кавоварками;
– електричними чайниками;
– телевізорами;
– кондиціонерами , обігрівачами , вентиляторами;
– пристрої системи керування «розумного будинку»;

Додатки як і самі пристрої інтернету речей можна віднести до будь-якої категорії – «фітнес та здоров'я» , «медицина» , «побутова техніка» , «розумний будинок».

Все залежить від того якого типу пристрій над яким здійснюється керування. У будь-якому випадку , якщо цей пристрій є у продажу , то додаток до нього можна скачати із каталогу додатків «*PlayMarket*» для мобільної операційної системи *Android.*

Функціональні можливості кожного додатку залежать від того якого типу є виконавчий пристрій яким виконується керування. Розпосюдженим є явище коли виробники електроприладів самі поставляють необхідні додатки для керування електроприладами їх виробничої лінійки при цьому додатки для керування приладами є сумісними та коректно працюють з усією виробничою лінійкою електроприладів даного виробника. В цьому випадку користувач лише виконує процес синхронізації додатків з своїми побутовими або промисловими електроприладами.

Розглянемо кілька прикладів користувацьких додатків для взаємодії з електроприладами , що дозволить зрозуміти які функціональні вимоги повинні виконуватись при розробці продукту пов'язаного з керуванням електроприладами.

Додаток *Lean remote* сам додаток являє собою реалізацію функціонального пульта дистанційного керування різного типу електроприладами , які , підтримують інтерфейси дистанційного керування та взаємодії. Додаток має два режими взаємодії з електроприладами.

Перший – це режим керування завдяки випромінюванню світла у інфрачервоному діапазоні але при цьому потрібно мати в мобільному телефоні встановлений порт випромінювання світла в інфрачервоному діапазоні для взаємодії з портом телевізора.

Другий режим керування це дистанційне керування за допомогою використання каналів зв'язку *Wi-fi* , *Bluetooth* дані інтерфейси взаємодії реалізовані у будь-яких сучасних смартфонах.

Сам додаток представляє собою функціональний пульт дистанційного керування побутовими електроприладами такими як *SMART*-телевізори , проектори , домашні кінотеатри , пристрої відтворення звуку , керування відтворенням потокового мультимедіа. Додаток має велику доступну базу даних пристроїв.

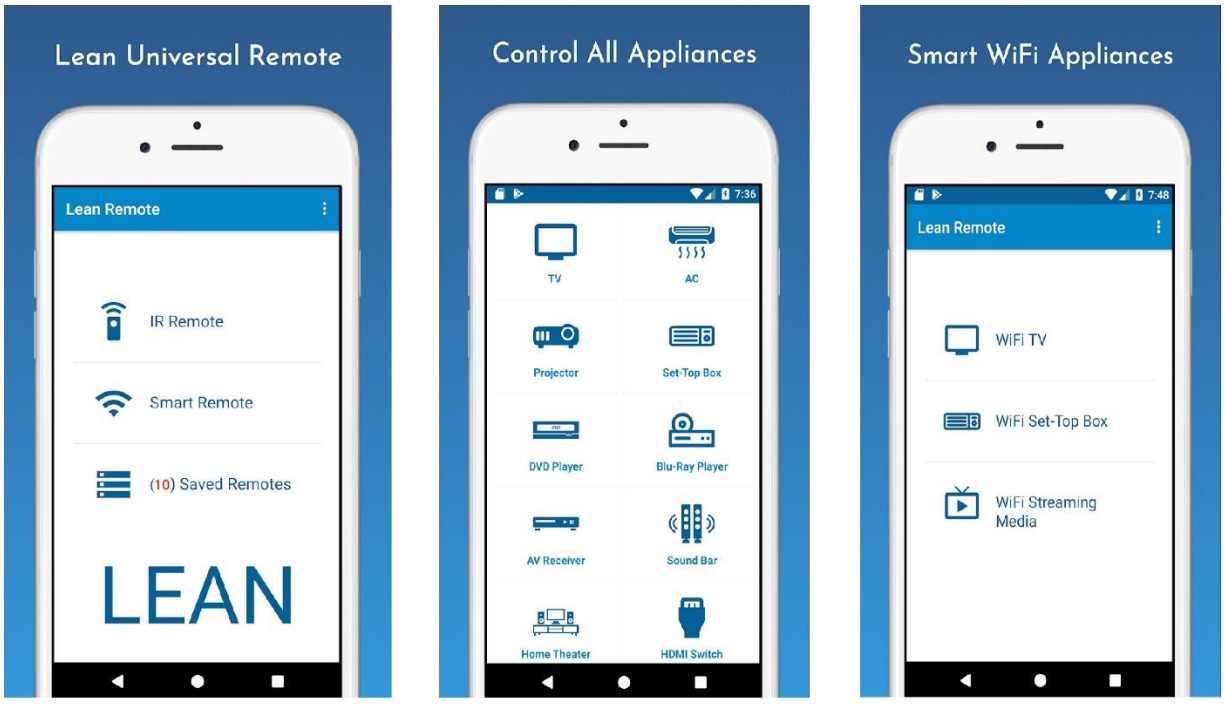

Рис.1.4. Головне вікно програми та вибір типу керування з вибором пристроїв керування

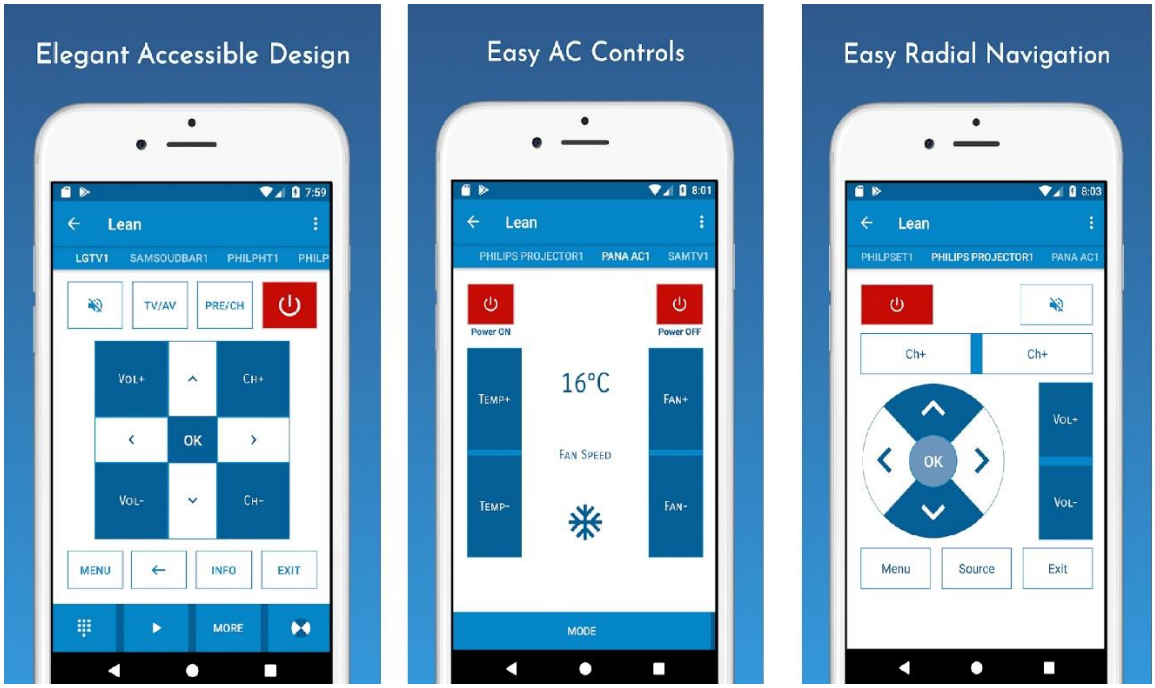

Рис.1.5. Функціонал реалізований в додатку

Функції дистанційного керування електроприладами реалізовані в додатку включають управління живленням , регулювання гучності звуку , навігація по перемиканню каналів , меню відтворення , пауза.

Кожен використаний пульт автоматично зберігається в меню збережені пульти в головному меню.

Додаток *Unified remote* – представляє користувачам функціонал дистанційного керування персональним комп'ютером за управлінням операційних систем *Windows* , *Mac* , *Linux*. Функціонал додатку дозволяє керувати різними функціями персонального комп'ютера з мобільного телефону через використання інтерфейсу взаємодії *Wi-fi* або *Bluetooth*.

В додатку реалізована підтримка великого пакету програм для роботи на персональному комп'ютері включаючи дистанційне керування переферійними пристроями такими як клавіатура та комп'ютерна миша за умови наявності в переферійних пристроях інтерфейсу дистанційного керування.

Основні функції додатку це допомога в організації зручності роботи технічному персоналу підприємств , установ , тощо для виконання своїх посадових обов'язків таких як наприклад налаштування серверів і серверних додатків зі смартфона , підтримка автоматичного пошуку серверів в локальній мережі до яких підключено пристрій керування (смартфон) , здійснення роботи

додатку по протоколу захищеного з’єднання , підтримка управління сенсорними переферійними пристроями за умови наявності інтерфейсу взаємодії , керування пристроями зі встановленими основними операційними системами *Windows* , *Mac* та *Linux*.

Також додаток підтримує керування пристроями на базі мікропроцесорної техніка *Arduino Yun* та має підтримку мікрокомп’ютерів лінійки *Raspberry Pi.*

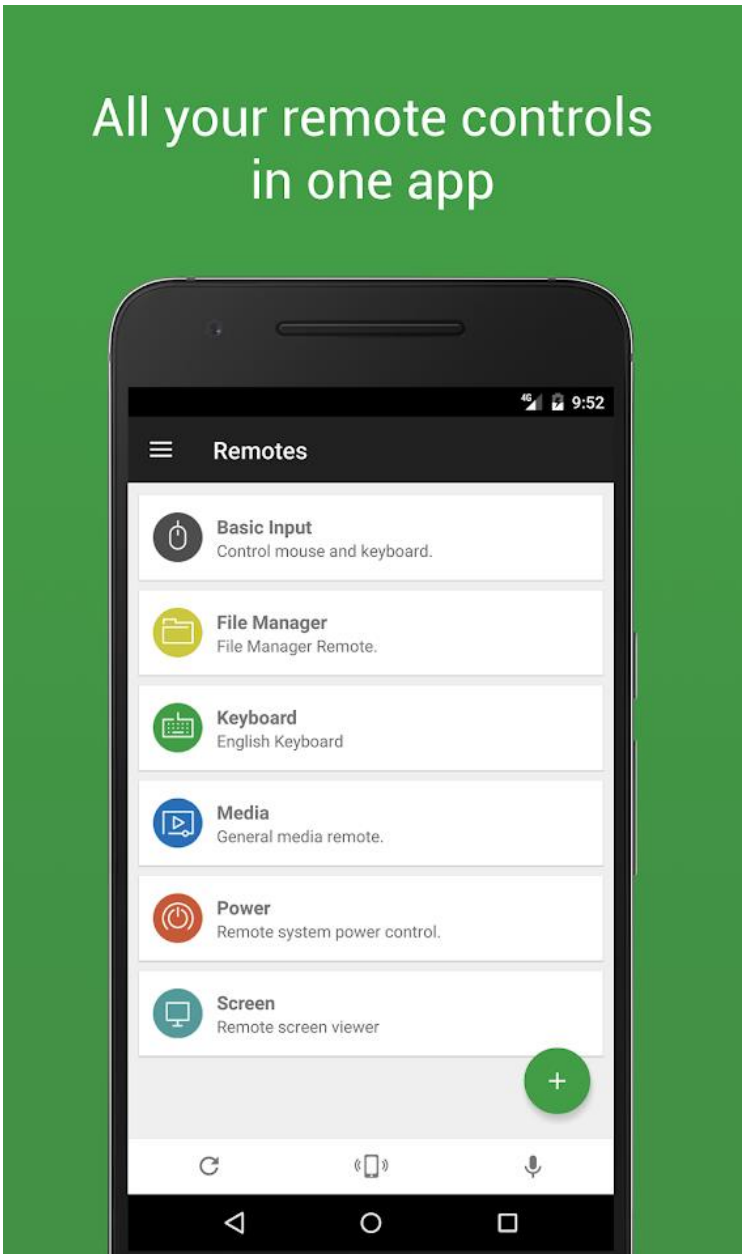

Рис.1.6. Головний екран додатку

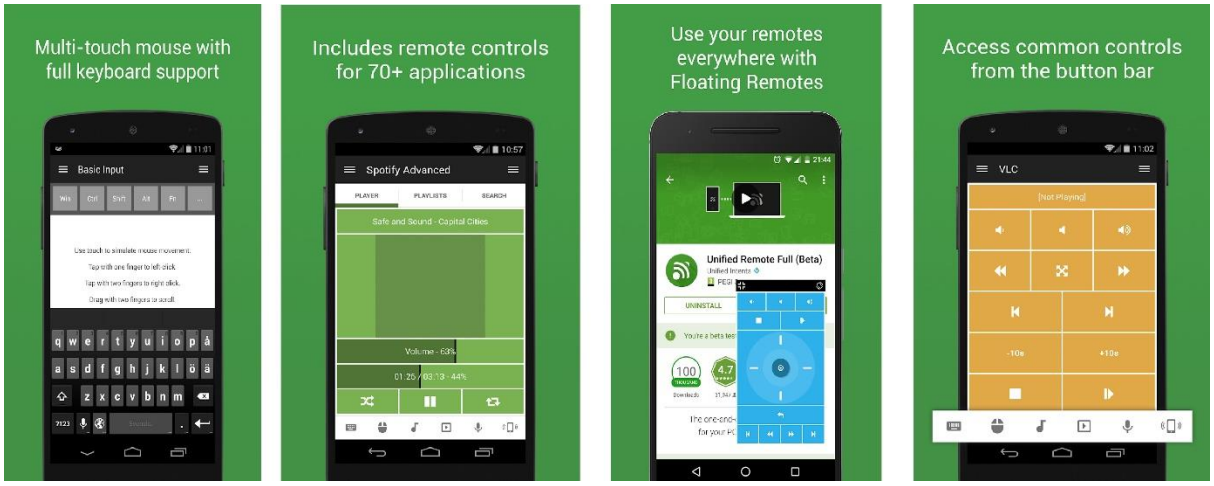

Рис.1.7. Функціонал реалізований в додатку

Додаток *Mi Remute* – Додаток для мобільних пристроїв , що працюють на операційній системі *Android* , використовується для дистанційного керування різними типами побутових пристроїв таких як:

– телевізори різних виробників;

– кондиціонери;

– DVD - плеєри;

– «розумні» розетки;

– проектори;

Додаток підтримує керування виробами найпопулярніших світових брендів виробників побутової техніки таких як *Samsung* , *LG* , *Sony* , *Panasonic* , *Sharp* , *Haier* , *Videocon* , *Micromax* , *Onida* та інші. Додаток працює з ІК – бластерами телефонів та різних видів побутової техніки яка підтримує даний інтерфейс керування. Всі мобільні пристрої можуть використовуватись для управління «розумними» телевізорами та іншими інтелектуальними пристроями з підтримкою стандартних протоколів керування *Wi-fi* або *Bluetooth.*

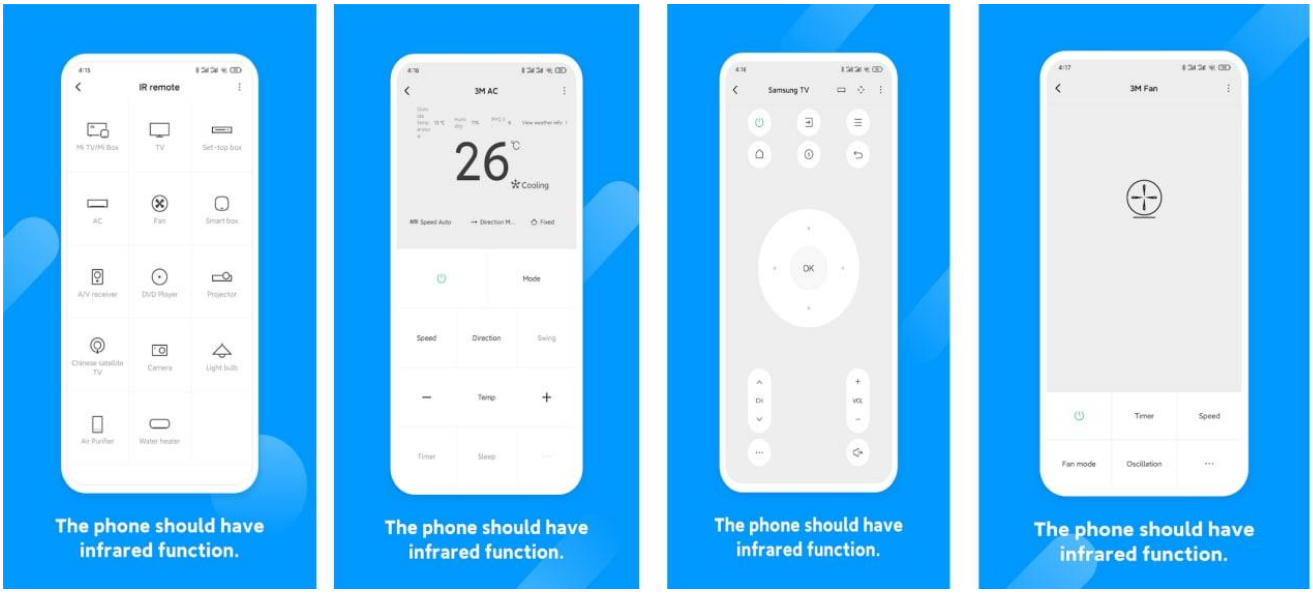

Рис.1.8. Екранні форми додатку

Після аналізу наявних у доступі мобільних додатків для дистанційного керування побутовими електроприладами можна дійти висновку , що питання дистанційного керування побутовими електроприаладми є актуальним для безлічі користувачів , але є проблеми у знаходженні зручних та зрозумілих додатків для певних груп користувачів та досі існують конфлікти сумісності додатків керування з деякими конкретними моделями побутової техніки.

1.3. Постановка завдання проектування

Проаналізовано найрозповсюдженіші мобільні додатки для дистанційного керування електроприладами на сьогоднішній день. Після аналізу наявних засобів дистанційного керування можна дійти висновку про те що різних додатків існує достатня кількість для керування найрозповсюдженішими моделями побутових пристроїв , але серед представлених додатків немає представників з відкритим вихідним кодом.

У підсумку можна сказати , що завданням для проектування є розробка програмного модуля для функціонування на мобільній операційній системі *Android* , що буде реалізовувати своїм функціоналом дистанційне керування побутовим електроприладом на мікропроцесорній базі мікроконтролеру *ESP32* який в свою чергу буде виконувати функції радіоприймача , а керування виконуєтсья через взаємодію мобільного пристрою (смартфону) та мікропроцесору шляхом реалізації бездротового інтерфейсу керування за використанням технології *Bluetooth.* Програмне забезпечення реалізовано у вигляді мобільного додатку створеного на мові програмування високого рівня *Java* та інструментальними засобами розробки *Android studio* , призначенням якого є керування даним електроприладом.

1.4. Висновки до розділу

1) Розглянуто основні поняття та терміни що використовуються в системах дистанційного керування електроприладами різного типу та призначення.

2) Проаналізовано дані досліджень використання мобільних додатків як систем керування електроприладами у побуті , промисловості , та комплексних програмно – апаратних системах «розумний будинок»

3) Відповідно до сучасних вимог з метою задоволення попиту на програмне забезпечення такого типу сформульоване завдання проектування метою якого є розробка програмного засобу – мобільного додатку для дистанційного керування електроприладом та додержання усіх технологічних етапів розробки а саме:

– аналіз існуючих програмних рішень , методів і технологій їх реалізації;

– розробити структуру системи керування електроприладом;

– розробити алгоритм роботи дистанційного керування;

– провести реалізацію розробленої системи дистанційного керування;

– розробити керівництво користувача;

– провести тестування системи дистанційного керування електроприладом.

РОЗДІЛ 2
ПОСТАНОВКА ЗАВДАННЯ ПРОЕКТУВАННЯ СИСТЕМИ ДИСТАНЦІЙНОГО КЕРУВАННЯ ЕЛЕКТРОПРИЛАДОМ

2.1. Протоколи дистанційної взаємодії

Протоколи дистанційної взаємодії – це певний набір промислових специфікацій бездротових персональних мереж реалізованих , як інтерфейс завдяки якому пристрої які підтримують технологію цю технологію можуть взаємодіяти між собою. Вони забезпечують обмін інформацією різні пристрої над якими можна здійснювати процес дистанційного керування , наприклад керування зі спеціального пульта або мобільного телефона засобом мобільного додатку.

До таких пристроїв можна віднести:

- персональні комп’ютери;
- мобільні телефони;
- принтери та різні офісні переферійні пристрої;
- акустичні системи;
- навушники та гарнітури;

Розглянемо основні технології дистанційного обміну даними які на сьогодні доступні для використання та не потребують спеціального дозволу є безкоштовними та доступними для розробки власних продуктів.

Технологія зв'язку	Wi-Fi	Bluetooth	ZigBee	Thread
Стандарт зв'язку	IEEE 802.11	IEEE 802.15.4	IEEE 802.15.4	IEEE 802.15.4
Швидкість передачі даних	300+ Мбит/с	до 3 Мбит/с	250 Кбит/с	250 Кбит/с
Енергоспоживання	Высокое	Низкое	Низкое	Низкое
Частотний діапазон	2,4 ГГц	2,4 ГГц	2,4 ГГц	2,4 ГГц
Підтримка IP-технологій	+	-	-	+
Топологія	«звезда»	«звезда»	«mesh»	«mesh»

Рис.2.1. Характеристики різних бездротових технологій

Технологія *Wi-fi* – ця технологія бездротової локальної мережі з приладами для передачі цифрових потоків даних по радіоканалам. Створювалась в якості заміни дротового інтерфейсу *Ethernet.* Саме тому ця технологія дозволяє використовувати великі швидкості передачі даних , але не дозволяє розробляти кінцеві вузли , які виконують свою роботу в довгому проміжку часу від джерела живлення малої ємності тому , що потребують великого енергоспоживання.

Технологія *Bluetooth* – технологія бездротової передачі інформації по радіочастотах ближнього радіозв'язку. Технологія *Bluetooth* дозволяє різним пристроям , у архітектурі яких передбачено встановлення *Bluetooth* – адаптерів дозволяє цим пристроям здійснювати обмін різною інформацією коли вони знаходяться на відстані один від одного , діапазон відстаней передачі інформації коливається від 100 м. у старих версіях протоколу зв'язку та до 1500 м. починаючи з версії *Bluetooth* 5 , відстань та якість переданої інформації залежить від фізичних перешкод та електромагнітних наведень навіть у одному приміщенні.

Принцип дії технології *Bluetooth* заснований на використанні радіоканалу зв'язку. Радіозв'язок *Bluetooth* виконується в ISM – діапазоні який використовується у різних побутових приладах та бездротових мережах зв'язку. Частоти роботи технології *Bluetooth* (2402 – 2480 МГц). Технологія *Bluetooth* використовує метод розширення спектру з стрибкоподібним переналаштуванням робочої частоти (FHSS) забезпечує стійкість до широкосмугових наведень.

Протокол *Bluetooth* підтримує з'єднання типу «*point-to-point*» але також і «point-to-multipoint».

Технологія *Bluetooth* у своїй роботі використовує певний стек протоколів (правил роботи) , який можна розділити на дві групи:

- протоколи універсального призначення;
- протоколи для використання у вбудованих системах;

Протоколи універсального призначення створенні для забезпечення функціональності та гнучкості використання на платформах різних пристроїв що підтримують використання цієї технології.

Протоколи для використання у вбудованих системах призначенні для використання у переферійних *Bluetooth* – пристроях де ресурси апаратної платформи обмежені а вимоги простіше. *Bluetooth* має багаторівневу архітектуру , що складається з основного протоколу , протоколів заміни кабелю , протоколів керування телефонією та запозичених протоколів радіозв'язку. Обов'язковими набором протоколів для обох стеків *Bluetooth* є протоколи *LMP* , *L2CAP* , *SDP*. Пристрої , які зв'язуються між собою для дистанційної передачі інформації зазвичай зв'язуються по *Bluetooth* використовуючи протоколи HCI та RFCOMM протоколи що під час своєї роботи виконують емуляццію послідовних портів поверх протоколу *L2CAP* , *RFCOMM* являє собою простий транспортний протокол з додатковими можливостями по емуляції 9 ланцюгів послідовних портів RS-232 та підтримує одночасно до 60 з'єднань між двома пристроями *Bluetooth*. У протоколі *RFCOMM* повний маршрут комунікації включає два додатки що являються кінцевими комунікаційними точками (кінцевими пристроями) між двома з'єднаними пристроями та комунікаційним сегментом між ними. Комунікаційний сегмент є *Bluetooth* – зв'язком від одного кінцевого пристрою до іншого та виконує топологію прямого з'єднання.

Загальний стек протоколів *Bluetooth* має наступний вигляд:

– *LMP* – використовується як інтерфейс комунікації між пристроями , та для встановлення та керування радіоз'єднанням між двома пристроями реалізован контроллером *Bluetooth*;

– *HCI* – ідентифікує зв'язок між стеком хост – пристрою та контроллером *Bluetooth*;

– *L2CAP* – використовується для передачі декількох потоків даних з меншою швидкістю по одному каналу зв'язку;

– *SDP* – дозволяє отримувати інформацію про послуги та функції , що виконують інші пристрої та ініціалізовувати їх параметри;

– *RFCOMM* – протокол який створює віртуальний послідовний потік даних та емулює сигнали керування RS -232;

– *TCS* – протокол , що визначає сигнали керування для передачі даних між пристроями з *Bluetooth* з'єднанням;

Технологія *Bluetooth* з появою стандарту 4.0 (*Bluetooth Smart* або *Bluetooth Low Energy*) стала більш привабливою для розробників мікроелектронних пристроїв , так як енергоспоживання в порівнянні з попередніми стандартами скоротилось.

Технологія *ZigBee* – це специфікація мережевих протоколів верхнього рівня додатків та мережевого рівня , що використовують сервіси нижніх рівнів для керування доступом по середовища та фізичного рівня інтегральних схем , які призначенні для виконання функцій фізичного рівня мережевої моделі *OSI*. Основним призначенням даного протоколу є надання бездротового інтерфейсу зв'язку між пристроями з низьким енергоспоживанням та можливостями побудови комірчастої топології мережі. Дана технологія розроблялась для створення надійних розподілених мереж певних датчиків для збору певних даних навколишнього середовища та керуючих пристроїв котрі використовують невисокі швидкості передачі даних , а підтримка IP-протоколу спрощує інтеграцію мереж з мережевими користувацькими додатками.

2.2. Структура програмного модуля

При дослідженні сучасних програмних додатків для дистанційного керування електроприладами можна дійти висновку , що програмний модуль повинен бути реалізований у вигляді мобільного додатку для смартфону під керуванням мобільної операційної системи *Android* з задіянням інтерфейсу дистанційної передачі файлів по каналу радіозв'язку *Bluetooth* та включати в себе наступний функціонал:

– набір функцій та методів для реалізації головного вікна програми та користувацького інтерфейсу та точки запуску додатку – *MainActivity.java* , та активності з реалізацією інтерфейсу керування *activity_main.xml*;

– файл реалізації інтерфейсу дистанційної взаємодії по радіоканалу *Bluetooth* мобільного додатку з самим керованим пристроем *AdapterList.java*

– файли реалізації користувацького меню пошуку інших пристроїв *device_item.xml* та *device_view_list.xml*;

– файл основних налаштувань додатку та установок взаємодії з мобільною операційною системою *Android;*

Структуру компонентів системи дистанційного керування електроприладом відображено на структурній схемі додатку системи дистанційного керування електроприладом (рис. 2..2.).

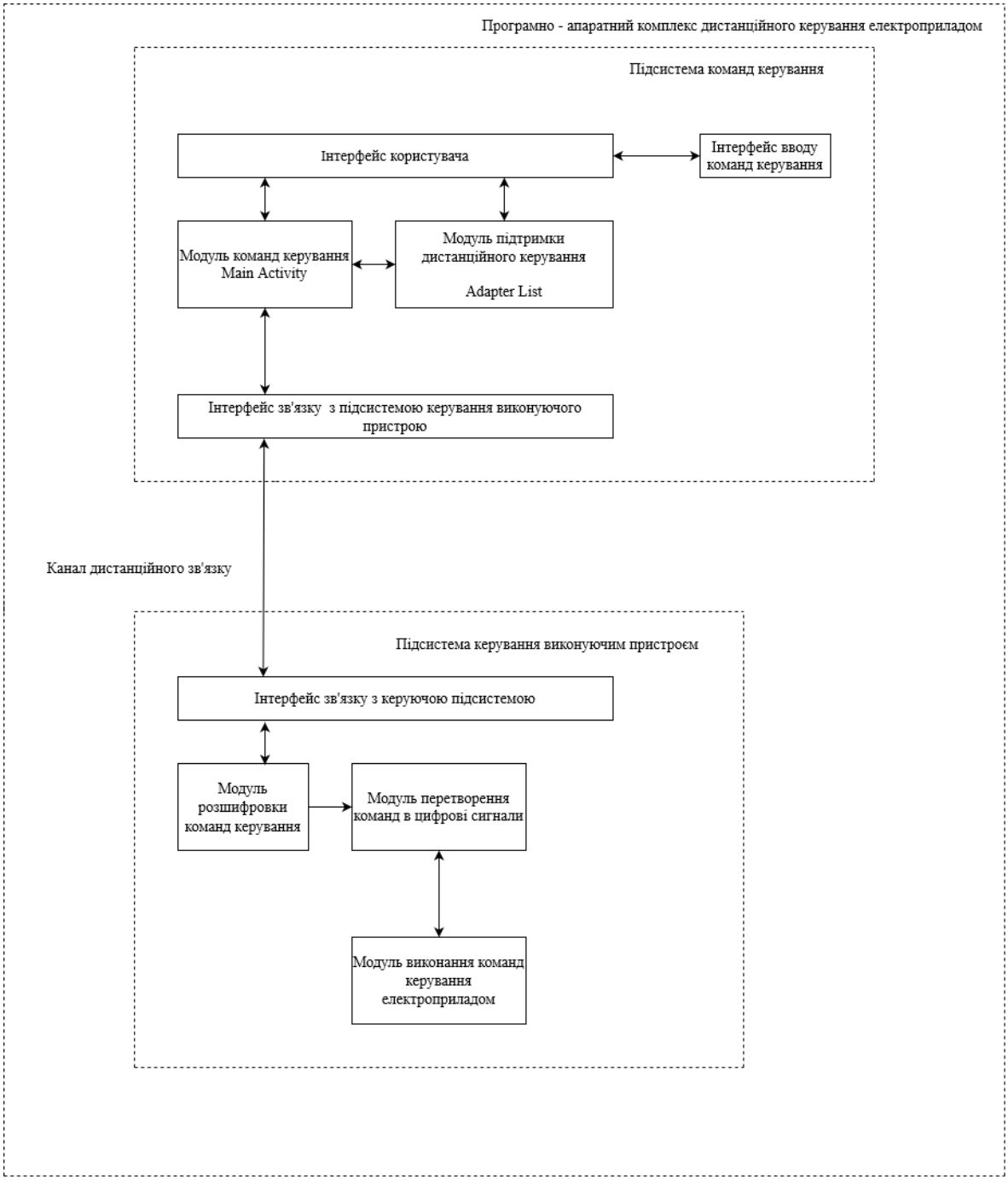

Рис.2.2. Структурна схема системи дистанційного керування електроприладом

Також під час проектування системи складено діаграму послідовностей (рис. 2.3.). На діаграмі послідовностей відображено послідовність дій , що відбуваються під час використання системи

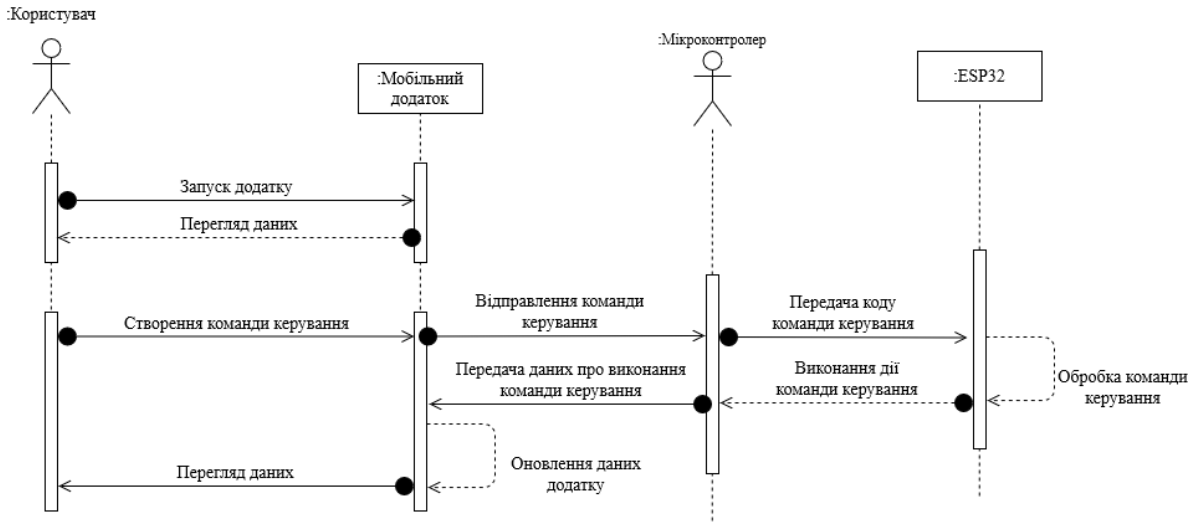

Рис.2.3. Діаграма послідовностей для демонстрації процесу роботи системи дистанційного керування електроприладом на базі мікроконтролера

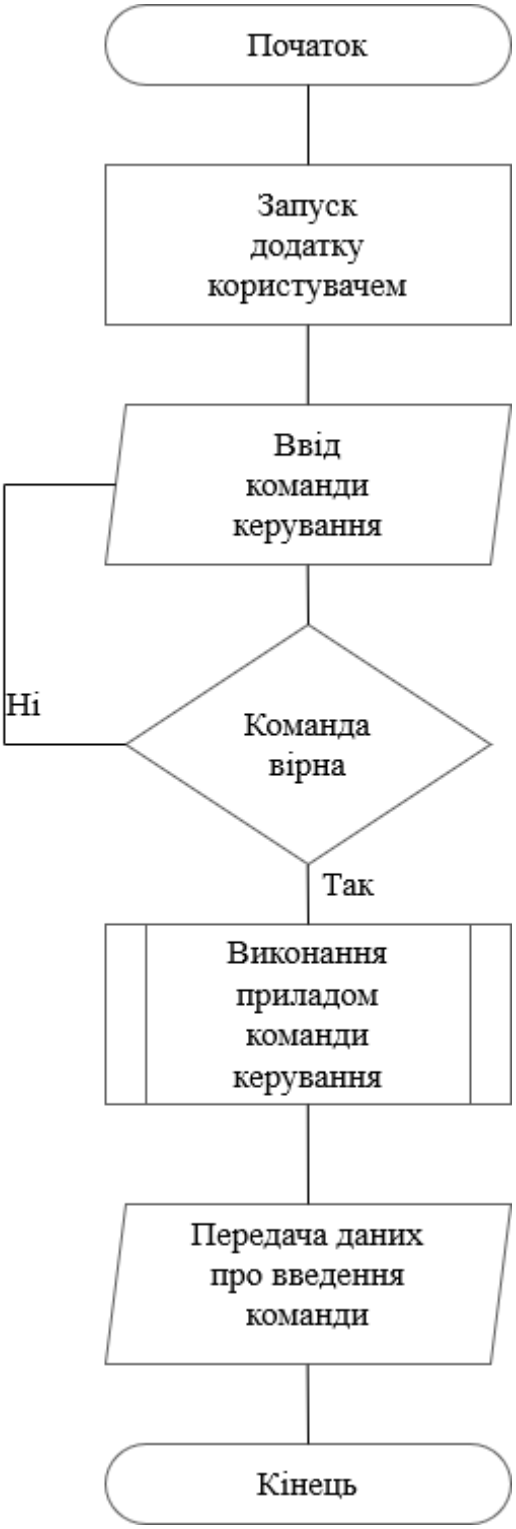

Рис.2.4. Загальна схема реалізації алгоритму дистанційного керування Електроприладом

На рис. 2.4. зображено загальну схему реалізації алгоритму роботи додатку яка містить всі етапи процесу дистанційного керування електроприладом при виконанні процесу керування.

Рис.2.5. Дає змогу зрозуміти вигляд класів розробленого додатку , який наочно демонструє всі необхідні елементи для створення мобільного додатку.

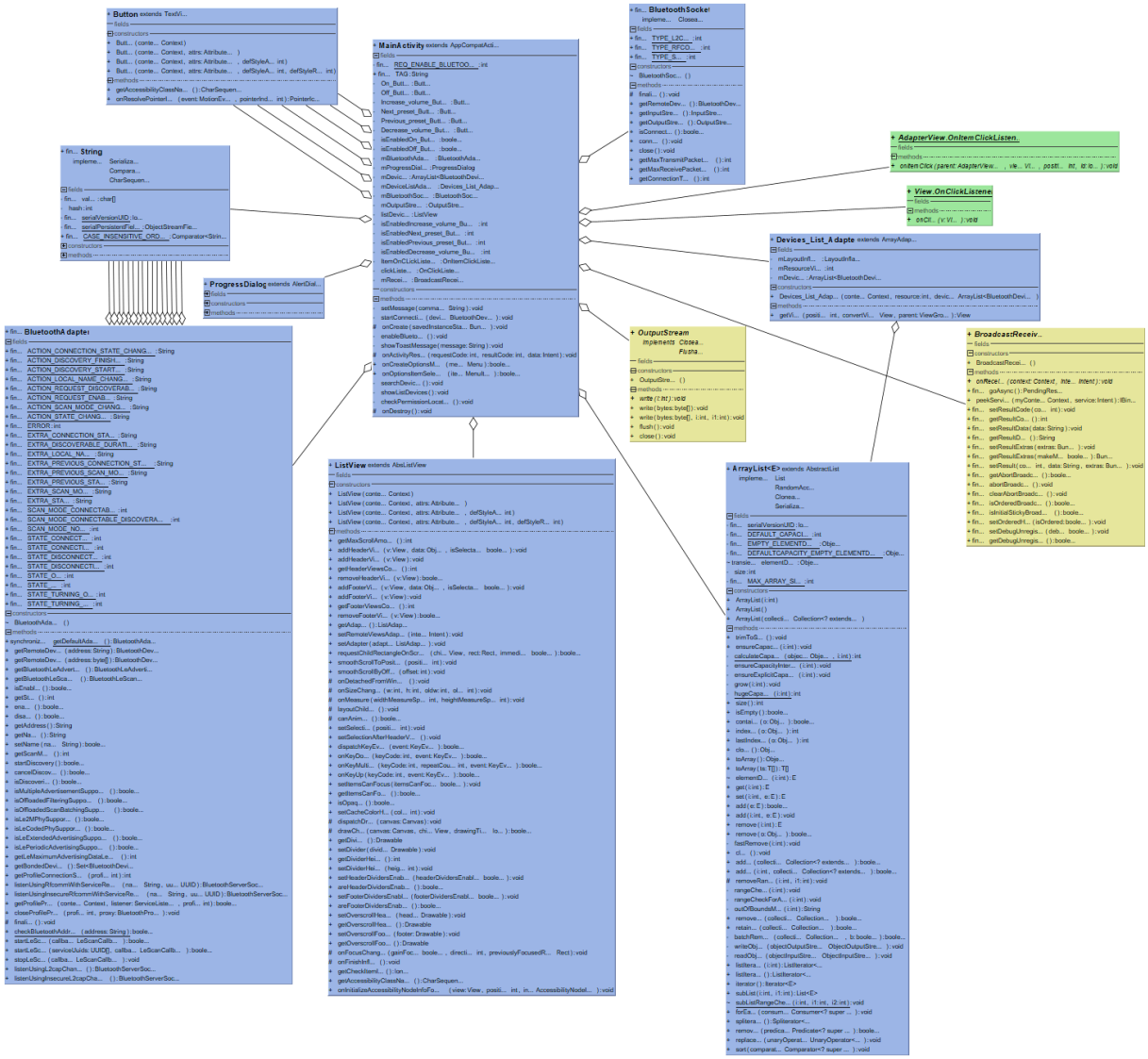

Рис.2.5. Діаграма класів мобільного додатку дистанційного керування електроприладом

2.3. Огляд інструментальних засобів розробника

Під час розробки даного програмного модулю основним інструментом розробника можна назвати інтегроване середовище розробки програмного забезпечення та мову програмування якою було написано сам програмний засіб. В цьому огляді розглянуто мови програмування та основні інструментальні засоби розробки завдяки яким було створено даний дипломний проєкт , далі приведено опис засобів реалізації.

Java – об'єктно – орієнтована мова програмування високого рівня підтримує реалізацію виконання кросплатформеного коду , офіційна реалізація мови програма написана мовою *Java* при початку процесу виконання програми компілює програму у байт-код , який при виконанні інтерпретується віртуальною машиною (*JVM*) для конкретної виконавчої платформи. Мова програмування *Java* розроблялася , як універсальна мова програмування , яку можна використовувати для різного спектру задач. На сьогодні *Java* це не лише мова програмування , а ціла платформа зі своєю унікальною єко-системою , яка об'єднує спектр технологій різного типу , що використовуються для різного ряду задач.

Основні направлення використання мови *Java*:

– створення додатків для виконання на персональних комп'ютерах;

– створення мережевих та клієнт-серверних додатків різної складності:

– створення мобільних додатків для виконання на пристроях які підтримують операційну систему *Android;*

Однією з ключових особливостей мови програмування *Java* є те , що при виконанні код створеного додатку спочатку транслюється у спеціальний байт-код , незалежний від конкретної платформи , потім цей код виконується спеціальною віртуальною машиною *JVM*. Таким чином реалізована архітектура забезпечує кросплатформеність та апаратну адаптацію під різні операційні системи та може виконуватись на таких платформах:

– *Windows*;

– *Linux*;

– *Mac OS*:

При цьому для кожної з платформ може бути своя унікальна реалізація віртуальної машини *JVM* , але при цьому кожна з них буде виконувати один і тот же програмний код. Мова програмування *Java* підтримує автоматичний збір непотрібних уривків даних у пам'яті ОЗУ пристрою на якому виконується додаток цей процес має назву «збір непотребу» точніше має назву *Garbage collection* у процесі виконання програми тому як у мові програмування *Java* пам'ять розподіляється на два сегменти – *heap* та *permanent generation*. Тому це означає , що розробнику не потрібно реалізовувати додаткові алгоритми по звільненню пам'яті виконуваного пристрою так як цей процес виконується автоматично. *Java* підтримує парадигму об'єктно-орієнтованого програмування

яка дозволяє використовувати такі конструкції мови як реалізація поліморфізму , наслідування , інкапсуляції та використання абстракцій у процесі написання коду при створенні програмних додатків. Саме така мова дозволяє вирішувати задачі по створенню великих за об'ємом але в той час «гнучких» програмних додатків.

Інтегроване середовище розробки *Android studio* – Інтегроване середовище розробки програмних додатків для операційної системи *Android* , яка підтримує розробку мобільних додатків на мовах програмування високого рівня *Java* та Kotlin. Розробником та компанією , що займається розвитком та підтримкою даного середовища програмування є компанія *Google*.

Середовище розробки програмного забезпечення *Android studio* заснована на програмному забезпеченні компанії *InteliJ IDEA* , яка є частиною компанії *JetBrains*. *Android studio* є офіційним засобом розробки мобільних додатків для операційної системи *Android.* Дане середовище розробки мобільних додатків доступне для операційних систем *Windows , Mac OS , Linux.*

Інтегроване середовище розробки програмного забезпечення *Android studio* створене з використанням мов програмування таких як *Java , Kotlin* та *C++*.

Android software development kit (*Android SDK*) – набір засобів розробки програмного забезпечення який дозволяє створювати додатки для певної апаратної платформи , комп'ютерної системи , мобільних та вбудованих операційних систем та середовищ виконання програмних додатків. В даному випадку технологія розробки програмного забезпечення для мобільних платформ *Android SDK* – це універсальний засіб розробки мобільних додатків та допоміжних компонентів для мобільної операційної системи *Android* , даний пакет надає розробникам програмного забезпечення функціональні можливості завдяки яким можна запускати тестування , налагодження отриманих в процесі розробки артефактів в режимі сумісності з різними видами операційної системи *Android* та спостерігати результат роботи в режимі реального часу. Даний набір програмних засобів підтримує більшу частину мобільних пристроїв серед яких є:

– мобільні телефони;

– портативні планшетні комп'ютери;

– автомобілі з вбудованими бортовими комп'ютерами , що працюють під керуванням операційної системи *Android*;

– телевізори з підтримкою «інтелектуального функціоналу роботи»;

– наручні годиники;

– та інші малогабаритні технічні пристрої;

2.4. Висновки до розділу

1) Проаналізовано сучасні інструментальні засоби розробки та мови програмування, технології та інтегровані середовища розробки. Зроблено вибір інструментальних засобів які задовольняють потребам розробки користувацького додатку для дистанційного керування електроприладами.

2) За допомогою функціональної схеми та діаграми послідовностей описано зв’язки модулів системи дистанційного керування електроприладами.

3) Розроблено діаграму компонентів додатку , що дає змогу детально зрозуміти функціональні залежності між компонентами додатку , які було використано в процесі розробки.

РОЗДІЛ 3
ПРОЄКТУВАННЯ СИСТЕМИ ДИСТАНЦІЙНОГО КЕРУВАННЯ ЕЛЕКТРОПРИЛАДОМ

3.1. Системні вимоги

Для опису характеристик, яким має відповідати виконуючий пристрій в даному випадку смартфон який виступає в ролі пульта керування електроприладом для коректної роботи розроблюваного додатку було визначено характеристики яким має відповідати керуючий пристрій.

1) смартфон під керуванням операційної системи *Android* версії 7.0 або 7.1 *Nougat*;

2) процесор з тактовою частотою 2.0 ГГц або більше;

3) Оперативна пам’ять –1024 мегабайт або більше;

4) Вільне місце у пам’яті внутрішнього накопичувача 10 мегабайт для встановлення додатку;

5) Вбудований інтерфейс комунікації *Bluetooth* 5.0;

3.2. Модель інтерфейсу

Під час проектування користувацької екранної форми додатку було використано засоби візуальної розробки програмного забезпечення створено екрану форму інтерфейсу користувача та реалізовано інтерфейс керування електроприладом , що відповідають за перемикання функцій електроприладу у випадку даної дипломної роботи це пристрій , що створений на базі мікроконтролеру *ESP32* за допомогою якого створено функціональний електроприлад , що виконує функції стаціонарного радіоприймача та реалізовано дистанційне керування за допомогою використання технології бездротової передачі даних *Bluetooth;*

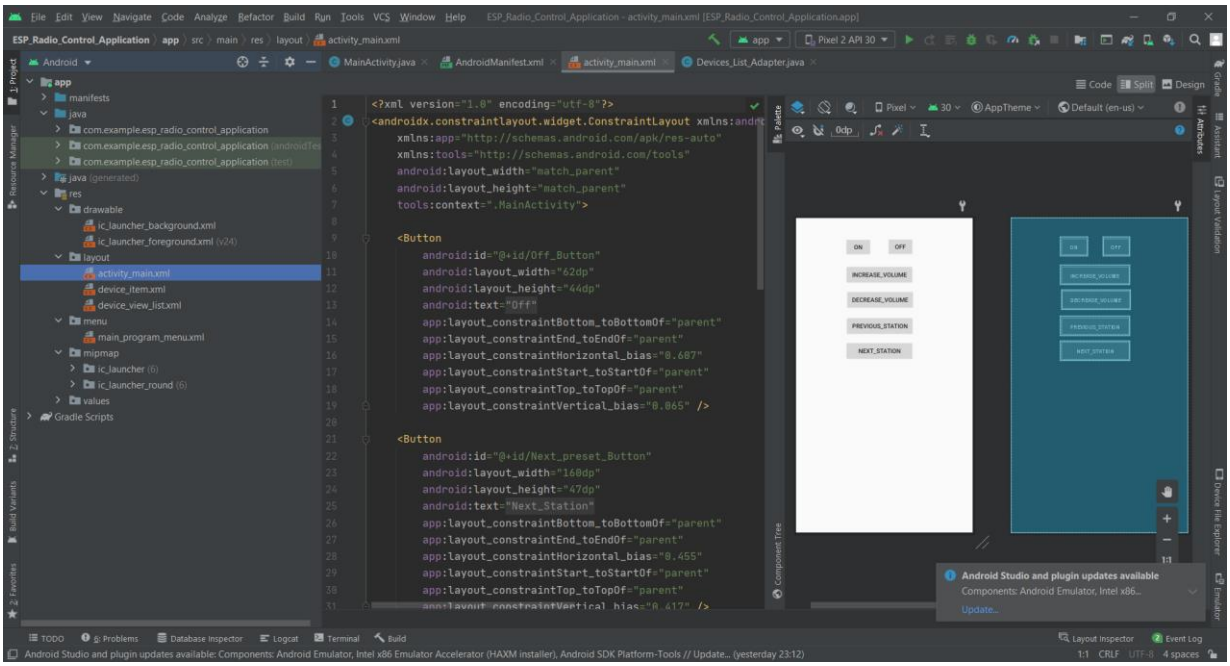

Рис.3.1. Проектування користувацького інтерфейсу додатку

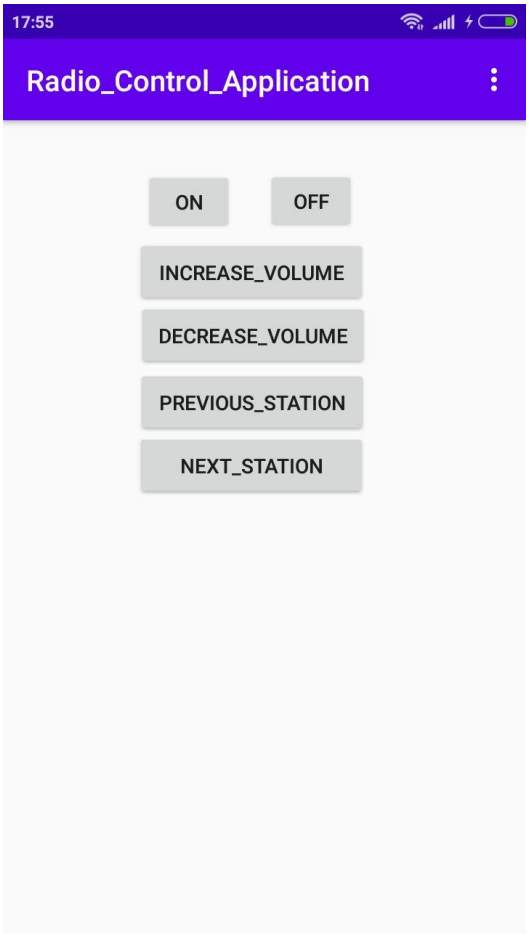

Рис.3.2. Екранна форма інтерфейсу користувача

3.3. Програмний код керування функціями електроприладу

В прикладі програмного коду керування електроприладом реалізовану функцію відправлення повідомлення через канал радіозв'язку *Bluetooth* під час виконання команди формується повідомлення в яке записується код керування пристроєм та посилається по каналу дистанційного зв'язку у вигляді набору байтів , що записуються до буферу обміну який формує повідомлення керування в якому за кожну функцію відповідає код керування мікроконтролером що записаний у скетч – прошивці мікроконтролеру.

```
private void setMessage (String command){
    byte[] buffer = command.getBytes();
    if (mOutputStream != null) {
        try {
            mOutputStream.write(buffer);
            mOutputStream.flush();

        } catch (IOException e) {
            showToastMessage("Command sending error!");
            e.printStackTrace();
        }
    }
}

private View.OnClickListener clickListener = new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String command = null;
        if (view.equals(On_Button)) {
            isEnabledOn_Button = !isEnabledOn_Button;
            command = "77";
            Log.d(TAG, "onClick: isEnabledOn_Button =  " + isEnabledOn_Button);

        }
        if (view.equals(Off_Button)) {
            isEnabledOff_Button = !isEnabledOff_Button;
            command = "77";
            Log.d(TAG, "onClick: isEnabledOff_Button =  " + isEnabledOff_Button);

        }
        if (view.equals(Increase_volume_Button)) {
            isEnabledIncrease_volume_Button = isEnabledIncrease_volume_Button + 1;
            command = "22";
            Log.d(TAG, "onClick: isEnabled_Increase_volume_Button =  " +
isEnabledIncrease_volume_Button);
        }
        if (view.equals(Next_preset_Button)) {
            isEnabledNext_preset_Button = isEnabledNext_preset_Button + 1;
            command = "44";
            Log.d(TAG, "onClick: isEnabledNext_preset_Button =  " +
isEnabledNext_preset_Button);

        }
        if (view.equals(Previous_preset_Button)) {
            isEnabledPrevious_preset_Button = isEnabledPrevious_preset_Button + 1;
            command = "55";
            Log.d(TAG, "onClick: isEnabledPrevious_preset_Button =  " +
```

```
isEnabledPrevious_preset_Button);

        }
        if (view.equals(Decrease_volume_Button)) {
            isEnabledDecrease_volume_Button = isEnabledDecrease_volume_Button +
1;
            command = "33";
            Log.d(TAG, "onClick: isEnabledDecrease_volume_Button =  " +
isEnabledDecrease_volume_Button);

        }
        setMessage(command);
    }

};
```

3.4. Висновки до розділу

1) У даному розділі було описано системі вимоги до пристрою керування для успішного використання мобільного додатку дистанційного керування електроприладом.

2) Було розроблено та продемонстровано графічний інтерфейс користувача.

3) Наведено програмний код системи керування електроприладом та описано , як саме відбувається процес дистанційного керування електроприладом шляхом відправки команд через використання інтерфейсу користувача мобільного додатку.

РОЗДІЛ 4
ТЕСТУВАННЯ СИСТЕМИ ДИСТАНЦІЙНОГО КЕРУВАННЯ ЕЛЕКТРОПРИЛАДОМ

4.1. Тестування програмної системи

Фінальним етапом розробки будь-якого програмного забезпечення є його тестування. Тестування являється одним із самих важливих та невиключних етапів у життєвому циклі розробки програмного забезпечення задля гарантування належної якості програмного продукту.

Тестування може бути ручне та автоматизоване.

Під час ручного тестування сам процес реалізовується у вигляді тест-кейсів. Тест – кейс це артефакт , сенс якого це виконання певного набору дій або умов необхідних для перевірки коректної функціональної роботи розроблюваної програмної системи.

Набір тест – кейсів є дуже поширеною практикою контролю забезпечення якості програмного забезпечення. У реалізації тест – кейсу це виглядає таким чином якщо отриманий результат відповідає очікуваному , то тест пройдено і до колонки результату записується стан тесту , як «пройдено» , а у інших випадках записується «не пройдено» Сама тестова ситуація зазвичай складається з трьох частин:

1) передумова – це послідовність кроків , що приводять систему до стану , придатного до тестування.
2) опис тестового випадку – це список дій завдяки яким є можливість перевірити необхідний функціонал програми.
3) постумова – це дії які повертають систему до початкового стану перед початком тестування.

Тестування даного програмного додатку проведено набором тест – кейсів , що перевіряють роботу основних функцій програми в різних умовах. Для прикладу в таблиці 1 розглянуто тестовий випадок відправок команд керування через використання інтерфейсу користувача до електроприладу шляхом дистанційної взаємодії передачі інформації з ним.

Таблиця 4.1

Тестовий випадок перевірки керування електроприладом шляхом посилання команд керування до електроприладу

Передумова		
Відкрити головне вікно програми. За допомогою клавіш *ON/OFF* перевірити вмикання та вимикання звуку під час роботи електроприладу , перевірити коректну роботу інших функціональних клавіш додатку		
Подія	Очікуваний результат	Фактичний результат
Тестовий випадок		
Обрати пункт *ON* у головному вікні користувацького інтерфейсу програми	Починається відтворення звуку на електроприладі	Пройдено.
Обрати пункт *OFF* у головному вікні користувацького інтерфейсу програми	Відтворення звуку на електроприладі припиняється	Пройдено.
Обрати пункт *INCREACE VOLUME* у головному вікні користувацького інтерфейсу програми	На електроприладі збільшується гучність відтворення звуку	Пройдено.

Продовження таблиці 4.1

Обрати пункт *DECREACE VOLUME* у головному вікні користувацького інтерфейсу програми	На електроприладі зменшується гучність відтворення звуку	Пройдено.
Обрати пункт *NEXT STATION* у головному вікні користувацького інтерфейсу програми	На електроприладі відбувається перемикання радіостанції «вперед» по частотному радіодіапазону	Пройдено.
Обрати пункт *PREVIOUS STATION* у головному вікні користувацького інтерфейсу програми	На електроприладі відбувається перемикання радіостанції «назад» по радіочастотному діапазону та повернення до попередньої радіостанції	Пройдено.
Постумова		
При успішному виконанні команд керування повертаємось до головного екрану програми	Відкриття головного екрану мобільного додатку	Пройдено.

4.2. Висновки до розділу

1) У даному розділі було описано процес та вимоги до тестування програмного забезпечення , задля забезпечення коректної роботи програмного додатку дистанційного керування електроприладом

2) Було проведено експеримент за планом тест – кейсу з тестування роботи мобільного додатку на предмет коректної взаємодії з електроприладом.

3) Протестовано мобільний додаток керування електроприладом за допомогою ручного тестування проведення якого продемонструвало наявну якість програмного продукту.

ВИСНОВКИ

Розглянуто та дано визначення основних понять та термінів, які використовуються у системах дистанційного керування, що дозволило визначити основні види дистанційного керування.

Проаналізовано статистику використання мобільних додатків для дистанційного керування різними промисловими та домашніми електроприладами.

Сформульовано завдання на проектування програмної системи, яке включає вхідні та вихідні дані та можливі обмеження.

Проаналізовано сучасний інструментарій розробника мобільних додатків для різних комп'ютерних систем та обрано необхідний інструментарій для виконання даної дипломної роботи.

Розроблено і описано структурну схему програмно – апаратного комплексу, що забезпечує дистанційну взаємодій мобільного пристрою керування та електроприладу шляхом взаємодії через радіоканал технології дистанційної передачі інформації *Bluetooth.*

Проектування проводилось засобами мови UML. В ході проектування складено діаграму класів мобільного додатку та діаграму послідовності, що дозволило встановити основні зв'язки між компонентами послідовність слідування повідомлень та команду управління.

Розроблено алгоритм дистанційного керування електроприладом, який є основою для розробленої програмної реалізації мобільного додатку дистанційного керування електроприладом.

В ході функціонального тестування складено набір тест-кейсів, за допомогою яких мобільний додаток було протестовано.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Barry, Burd Android Application Development All–in–One For Dummies® / Barry Burd. - Москва: Машиностроение 2011. - 816 с.

2. Голощапов, Алексей Google Android. Программирование для мобильных устройств , 2011. - 438 с.

3. Дэрси, Лорен Android за 24 часа. Программирование приложений под операционную систему Google / Лорен Дэрси , Шейн Кондер. - М.: Рид Групп, 2011. - 464 с.

4. Майер, Рето Android 2. Программирование приложений для планшетных компьютеров и смартфонов / Рето Майер. - М.: "Издательство "Эксмо", 2011. - 672 с.

5. Майер, Рето Android 4. Программирование приложений для планшетных компьютеров и смартфонов / Рето Майер. - М.: Эксмо, 2013. - 816 с.

6. Мартин, К. Соломон Oracle. Программирование на языке Java / Мартин К. Соломон, Нирва Мориссо-Леруа , Джули Басу. - М.: ЛОРИ, 2010. - 512 с.

7. Роджерс, Рик Android. Разработка приложений / Рик Роджерс и др. - М.: ЭКОМ Паблишерз, 2010. - 400 с.

8. Нотон Java. Справочное руководство. Все, что необходимо для программирования на Java / Нотон, Патрик. - М.: Бином, 2015. - 448 с.

9. Герберт, Шилдт Java 8. Руководство для начинающих / Шилдт Герберт. - М.: Диалектика / Вильямс, 2015. - 899 с.

10. Герберт, Шилдт Java. Руководство для начинающих / Шилдт Герберт. - М.: Диалектика / Вильямс, 2014. - 104 с.

11. Давыдов, Станислав IntelliJ IDEA. Профессиональное программирование на Java / Станислав Давыдов , Алексей Ефимов. - М.: БХВ-Петербург, 2015. - 800 с.

12. Джошуа, Блох Java. Эффективное программирование / Блох Джошуа. - М.: ЛОРИ, 2014. - 292 с.

13. Льюис, Дирк Самоучитель Java 7 / Дирк Льюис , Петер Мюллер. - М.: БХВ-Петербург, 2013. - 464 с.

14. Савитч, Уолтер Язык Java. Курс программирования / Уолтер Савитч. - М.: Вильямс, 2010. - 928 с.

15. Сеттер, Р. В. Изучаем Java на примерах и задачах / Р.В. Сеттер. - М.: Наука и техника, 2016. - 240 с.

16. Хорстманн, Кей С. Java SE 8. Вводный курс / Хорстманн Кей С.. - М.: Диалектика / Вильямс, 2014. - 898 с.

17. ДСТУ 3008-15 Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.

18. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.

ДОДАТОК А – Код модуля *MainActivity*

```
package com.example.esp_radio_control_application;

public class MainActivity extends AppCompatActivity {
  // protected void onCreate() {

      private static final int REQ_ENABLE_BLUETOOTH = 1001;
      public final String TAG = getClass().getSimpleName();
      private Button On_Button; // 1
      private Button Off_Button;// 2
      private Button Increase_volume_Button; // 3
      private Button Next_preset_Button; // 4
      private Button Previous_preset_Button; // 5
      private Button Decrease_volume_Button; // 6

      private boolean isEnabledOn_Button = false;
      private boolean isEnabledOff_Button = false;

      private BluetoothAdapter mBluetoothAdapter;
      private ProgressDialog mProgressDialog;
      private ArrayList<BluetoothDevice> mDevices = new
ArrayList<>();//Final_Searching Device
      private Devices_List_Adapter mDeviceListAdapter;

      private BluetoothSocket mBluetoothSocket;
      private OutputStream mOutputStream;

      private ListView listDevices;

      private int isEnabledIncrease_volume_Button = 0;
      private int isEnabledNext_preset_Button = 0;
      private int isEnabledPrevious_preset_Button = 0;
      private int isEnabledDecrease_volume_Button = 0;

      private AdapterView.OnItemClickListener ItemOnCLickListener = new
AdapterView.OnItemClickListener() {
          @Override
          public void onItemClick(AdapterView<?> adapterView, View view, int
position, long l) {
              BluetoothDevice device = mDevices.get(position);
              startConnection(device);

          }
      };
```

```
    private void setMessage (String command){
        byte[] buffer = command.getBytes();
        if (mOutputStream != null) {
            try {
                mOutputStream.write(buffer);
                mOutputStream.flush();

            } catch (IOException e) {
                showToastMessage("Command sending error!");
                e.printStackTrace();
            }
        }
    }

    private void startConnection (BluetoothDevice device){
        if (device != null) {
            try {
                Method method = device.getClass().getMethod("createRfcommSocket",
new Class[]{int.class});
                mBluetoothSocket = (BluetoothSocket) method.invoke(device, 1);
                mBluetoothSocket.connect();
                mOutputStream = mBluetoothSocket.getOutputStream();
                showToastMessage("Connection successful!");
            } catch (Exception e) {
                showToastMessage("Connection error!");
                e.printStackTrace();
            }

        }
    }

    @Override
    protected void onCreate (Bundle savedInstanceState){
        Log.d(TAG, "onCreate()");
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        On_Button = findViewById(R.id.On_Button);
        Off_Button = findViewById(R.id.Off_Button);
        Increase_volume_Button = findViewById(R.id.Increase_volume_Button);
        Next_preset_Button = findViewById(R.id.Next_preset_Button);
        Previous_preset_Button = findViewById(R.id.Previous_preset_Button);
        Decrease_volume_Button = findViewById(R.id.Decrease_volume_Button);

        On_Button.setOnClickListener(clickListener);
        Off_Button.setOnClickListener(clickListener);
        Increase_volume_Button.setOnClickListener(clickListener);
        Next_preset_Button.setOnClickListener(clickListener);
```

```
        Previous_preset_Button.setOnClickListener(clickListener);
        Decrease_volume_Button.setOnClickListener(clickListener);

        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if (mBluetoothAdapter == null) {
            Log.d(TAG, "onCreate: The device does not support Bluetooth
technology");
            finish();

        }

        mDeviceListAdapter = new Devices_List_Adapter(this, R.layout.device_item,
mDevices);
        enableBluetooth();
    }

    private void enableBluetooth () {
        Log.d(TAG, "enableBluetooth: EnableBluetooth()");
        if (!mBluetoothAdapter.isEnabled()) {
            Log.d(TAG, "enableBluetooth: Bluetooth off, turn on bluetooth?");
            Intent intent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(intent, REQ_ENABLE_BLUETOOTH);
        }
    }

    private void showToastMessage (String message){
        //Toast.makeText(this, "", Toast.LENGTH_SHORT).show();
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    }

    @Override
    protected void onActivityResult ( int requestCode, int resultCode, @Nullable
Intent data){
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQ_ENABLE_BLUETOOTH) {
            if (mBluetoothAdapter.isEnabled()) {
                Log.d(TAG, "onActivityResult: Trying to turn on Bluetooth again...");
                enableBluetooth();

            }
        }
    }

    @Override
    public boolean onCreateOptionsMenu (Menu menu){
        getMenuInflater().inflate(R.menu.main_program_menu, menu);
```

```
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected (@NonNull MenuItem item){
        switch (item.getItemId()) {
            case R.id.item_search:
                searchDevices(); //searchUsersDevice
                break;

            case R.id.Exit:
                finish();
                break;
        }

        return super.onOptionsItemSelected(item);
    }

    private void searchDevices () {
        Log.d(TAG, "searchDevices: ");
        enableBluetooth();
        checkPermissionLocation();
        if (!mBluetoothAdapter.isDiscovering()) {
            Log.d(TAG, "searchUsersDevice: Searching for devices...");
            mBluetoothAdapter.startDiscovery();
        }
        if (mBluetoothAdapter.isDiscovering()) {
            Log.d(TAG, "searchUsersDevice: Search has already been
started...restarting device search...");
            mBluetoothAdapter.cancelDiscovery();
            mBluetoothAdapter.startDiscovery();
        }
        IntentFilter SearchStatus = new
IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_STARTED);

SearchStatus.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        SearchStatus.addAction(BluetoothDevice.ACTION_FOUND);
        registerReceiver(mReceiver, SearchStatus);

    }

    private void showListDevices () {
        Log.d(TAG, "showDevicesList: ");
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Found devices:");

        View view = getLayoutInflater().inflate(R.layout.device_view_list, null);
```

```
        listDevices = view.findViewById(R.id.list_devices);
        listDevices.setAdapter(mDeviceListAdapter);
        listDevices.setOnItemClickListener(ItemOnCLickListener);

        builder.setView(view);
        builder.setNegativeButton("OK", null);
        builder.create();
        builder.show();
    }

    private void checkPermissionLocation () {
        if (Build.VERSION.SDK_INT > Build.VERSION_CODES.N) {
            int check =
checkSelfPermission("Manifest.permission.ACCESS_FINE_LOCATION");
            check +=
checkSelfPermission("Manifest.permission.ACCESS_COARSE_LOCATION");
            if (check != 0) {
                requestPermissions(new
String[]{Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION}, 1002);
            }
        }
    }

    private View.OnClickListener clickListener = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String command = null;
            if (view.equals(On_Button)) {
                isEnabledOn_Button = !isEnabledOn_Button;
                command = "77";
                Log.d(TAG, "onClick: isEnabledOn_Button =  " +
isEnabledOn_Button);

            }
            if (view.equals(Off_Button)) {
                isEnabledOff_Button = !isEnabledOff_Button;
                command = "77";
                Log.d(TAG, "onClick: isEnabledOff_Button =  " +
isEnabledOff_Button);

            }
            if (view.equals(Increase_volume_Button)) {
                isEnabledIncrease_volume_Button = isEnabledIncrease_volume_Button
+ 1;
                command = "22";
                Log.d(TAG, "onClick: isEnabled_Increase_volume_Button =  " +
```

```
isEnabledIncrease_volume_Button);
            }
            if (view.equals(Next_preset_Button)) {
               isEnabledNext_preset_Button = isEnabledNext_preset_Button + 1;
               command = "44";
               Log.d(TAG, "onClick: isEnabledNext_preset_Button =  " +
isEnabledNext_preset_Button);

            }
            if (view.equals(Previous_preset_Button)) {
               isEnabledPrevious_preset_Button = isEnabledPrevious_preset_Button
+ 1;
               command = "55";
               Log.d(TAG, "onClick: isEnabledPrevious_preset_Button =  " +
isEnabledPrevious_preset_Button);

            }
            if (view.equals(Decrease_volume_Button)) {
               isEnabledDecrease_volume_Button =
isEnabledDecrease_volume_Button + 1;
               command = "33";
               Log.d(TAG, "onClick: isEnabledDecrease_volume_Button =  " +
isEnabledDecrease_volume_Button);

            }
            setMessage(command);
         }

      };

      protected void onDestroy () {
         super.onDestroy();
         try {
            if (mBluetoothSocket != null) {
               mBluetoothSocket.close();
            }
            if (mOutputStream != null) {
               mOutputStream.close();
            }
         } catch (IOException e) {
            e.printStackTrace();
         }
      }

      private BroadcastReceiver mReceiver = new BroadcastReceiver() {
         @Override
         public void onReceive(Context context, Intent intent) {
```

```
            final String action = intent.getAction();
            if (action.equals(BluetoothAdapter.ACTION_DISCOVERY_STARTED)) {
               Log.d(TAG, "onReceive: ACTION_DISCOVERY_STARTED");
               showToastMessage("Start searching for devices...");
               mProgressDialog = ProgressDialog.show(MainActivity.this, "Search for
devices", "Wait please...");
            }
            if (action.equals(BluetoothAdapter.ACTION_DISCOVERY_FINISHED)) {
               Log.d(TAG, "onReceive: ACTION_DISCOVERY_FINISHED");
               showToastMessage("The search for devices is complete.");
               mProgressDialog.dismiss();
               showListDevices();
            }
            if (action.equals(BluetoothDevice.ACTION_FOUND)) {
               Log.d(TAG, "onReceive: ACTION_FOUND");
               //showToastMessage("Start searching for devices...");
               BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
               if (device != null) {
                  if (!mDevices.contains(device))
                     mDeviceListAdapter.add(device);

               }
            }
         }
      };
   }
```

ДОДАТОК Б – Код модуля *Adapter List*

```
package com.example.esp_radio_control_application;
public class Devices_List_Adapter extends ArrayAdapter<BluetoothDevice> {
   private LayoutInflater mLayoutInflater;
   private int mResourceView;
   private ArrayList<BluetoothDevice> mDevices = new ArrayList<>();
   public Devices_List_Adapter(@NonNull Context context,int
resource,ArrayList<BluetoothDevice> devices) {
      super(context, resource, devices);
      mLayoutInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
      mResourceView = resource;
      mDevices = devices;
   }

   @NonNull
   @Override
   public View getView(int position, View convertView,  ViewGroup parent) {
      convertView=mLayoutInflater.inflate(mResourceView,null);

      BluetoothDevice device = mDevices.get(position);
      TextView tvName = convertView.findViewById(R.id.tvNameDevice);
      TextView tvAddress = convertView.findViewById(R.id.tvAddressDevice);

      tvName.setText(device.getName());
      tvAddress.setText(device.getAddress());

      return convertView;
   }
}
```