НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Радіотехнічний факультет

Кафедра радіоконструювання та виробництва радіоапаратури

«На правах рукопису»
УДК 62-519_____

До захисту допущено:
В.о. зав. кафедри
_____ Євгеній НЕЛІН
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інтелектуальні технології мікросистемної радіоелектронної техніки»

за спеціальністю 172 «Телекомунікації та радіотехніка»

на тему: «Дистанційне керування силовим навантаженням»

Виконав (-ла):
студент (-ка) 2 курсу, групи РІ-91мп

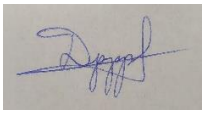
Драчук Олександр Сергійович_____

Керівник:

к.т.н., доцент Тарабаров Сергій Борисович

Рецензент:
Доцент, к.т.н. каф.РТПС Піддубний Володимир Олексійович_____ _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.



Студент (-ка)

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Радіотехнічний факультет

Кафедра радіоконструювання та виробництва радіоапаратури

Рівень вищої освіти – другий (магістерський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Інтелектуальні технології мікросистемної радіоелектронної техніки»

ЗАТВЕРДЖУЮ

В.о.завідувача кафедри

_____ Євгеній НЕЛІН

«2» вересня 2020 р.

ЗАВДАННЯ

на магістерську дисертацію студента

Драчука Олександра Сергійовича

1.Тема дисертації «Дистанційне керування силовим навантаженням»
науковий керівник дисертації Тарабаров Сергій Борисович, к.т.н., доцент
затверджені наказом по університету від «5» листопада 2020 р. №3223-с

2. Термін подання студентом дисертації 11 грудня 2020 року

3. Об'єкт дослідження сукупність засобів дистанційного керування силовим навантаженням._____

4. Вихідні дані протоколи комунікації програмних та прикладних засобів керування в межах локальної мережі WiFi._____

5. Перелік завдань, які потрібно розробити: провести аналіз аналогів на ринку, Аналіз можливих протоколів передачі даних, розробити необхідних моделей для проектування програмного рішення, алгоритм пеленгування вузла в локальній мережі, розробити програмне рішення для ПК та смартфону._____

6. Орієнтовний перелік графічного (ілюстративного) матеріалу _____ презентація_____

_____

7. Орієнтовний перелік публікацій _____

_____

9. Дата видачі завдання 02 вересня 2020 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Підбір джерел за темою МД	02.09.2020-05.09.2020	виконано
2	Розробка ТЗ	07.09.2020-15.09.2020	виконано
3	Дослідження моделей розробки ПЗ	20.09.2020-01.10.2020	виконано
4	Розроблення алгоритмів веб-додатків	02.10.2020-29.10.2020	виконано
5	Розроблення програмного забезпечення	03.11.2020-27.11.2020	виконано
6	Оформлення та підготовка до захисту МД	02.12.2020-05.12.2020	виконано

Студент Олександр ДРАЧУК

Науковий керівник Сергій ТАРАБАРОВ

РЕФЕРАТ

Мігстерська дисертація становить 103 сторінок, містить 5 розділів, 25 ілюстрацій, 21 таблицю, 6 додатків та 19 джерел за переліком посилань.

Ключові слова: DOM, AJAX, OSI, HTTP, JavaScript, HTML/CSS, XAML, C#.

Актуальність теми: розширення можливостей та якості функціонування електронних приладів з дистанційнним керуванням, які все ширше використовуються у виробництві та побуті, вимагає дослідження та модифікації програмного забезпечення взаємодії контролюючих та контрольованих пристроїв.

Мета дослідження: визначити необхідні технології, побудувати нові алгоритми та використати їх при розробці програмного забезпечення для керування пристроями силового навантаження.

Завдання дослідження: виявлення проблем сучасних пристроїв керування силовим навантаженням на прикладі розумного освітлення, аналіз можливих протоколів передачі даних, розгляд необхідних моделей для проєктування програмного рішення, розроблення драйверу для міктроконтролера, застосунків керування через веб-сторінку, окремий додаток для комп'ютера або смартфону, аналіз результатів.

Об'єкт дослідження — сукупність засобів дистанційного керування силовим навантаженням.

Предмет дослідження — протоколи комунікації програмних та прикладних засобів керування в межах локальної мережі WiFi.

Наукова новизна одержаних результатів. Запропоновано швидкий алгоритм пошуку адреси веб-серверу в межах локальної мережі WiFi за допомогою асинхронних функціїй в поєднанні із системними утилітами.

ABSTRACT

Migster's dissertation consists of 103 pages, contains 5 chapters, 25 illustrations, 21 tables, 6 appendices and 19 sources according to the list of references.

Key words: DOM, AJAX, OSI, HTTP, JavaScript, HTML/CSS, XAML, C#.

Significance of the study: due to the growing number of devices with the possibility of remote control, there is an urgent need to study algorithms for the interaction of software with control or management devices.

The purpose of the study is to define necessary technologies and constructions of new algorithms on the basis of the conducted research as well as the application of those technologies and algorithms at development of web applications for management of devices of power loading.

Objectives of the study imply the identification of the problems of modern power control devices on the example of intelligent lighting, the analysis of possible data transmission protocols, consideration of necessary models for software application design, development of microcontroller driver, browser control applications, separate application for computer and smartphone or the analysis of results.

The object of the study is a set of tools for remote control of power load.

The subject of the study is the communication protocols of software and application controls within the WiFi local network.

Scientific novelty of the obtained results. A fast algorithm for finding the address of a web server within a local WiFi network using asynchronous functions in combination with system utilities is proposed.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до магістерської дисертації

на тему: Дистанційне керування силовим навантаженням

Київ — 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ШІМ — широтно-імпульсна модуляція;

AJAX — асинхронний JavaScript та XML;

COM-port — послідовний інтерфейс передачі інформації;

CSS — мова каскадних таблиць стилів;

DHCP — мережевий сертифікат для автоматичного видання IP-адреси;

DOM — об'єктна модель документа;

HTML — мова гіпертекстової розмітки сторінки;

HTTP — протокол передачі гіпертексту, даних;

IDE — інтегроване середовище розробки програмного забещпечення;

IP-адреса — ідентифікатор пристрою мережевого рівня;

OSI — мережева модель для комунікації мережевих протоколів;

TCP/IP — набір протоколові мережі інтернет;

WEP — старий стандарт захисту бездротового трафіку;

WPA/WPA2 — новий стандарт захисту інформації у бездротових мережах;

XAML — декларативна мова розмітки інтерфейсу додатків;

XML — розширювана мова розмітки.

ВСТУП

Людство весь час прямує в напрямі економії свого часу на повсякденних задачах, а відповідно і зростає потреба в вирішенні будь-яких потреб шляхом автоматизації і використанні новітніх технологій.

Велике різноманіття давачів в поєднанні з досконалими програмними рішеннями на базі мікроконтролерів надає можливість сучасній електроніці дистанційно виконувати моніторинг, керування опаленням, освітленням, двигунами, відео наглядом, системами сигналізації тощо.

У зв'язку із зростаючою популярністю систем "розумний будинок" та відповідно більш глибоким дослідженням шляхів оптимізації розробки програмного забезпечення, сучасні методи дистанційної передачі даних дозволяють створювати пристрої не витрачаючи великих коштів.

Сьогодні розповсюдженні технології 4G, WiFi-мережі забезпечують достатні показники швидкості передачі даних та стабільності з'єднання між компонентами систем "розумного дому".

Мови програмування дозволяють загорнути використані при розробці пристроїв алгоритми в просту для користувача «обгортку» на комп'ютері або смартфоні. Метою даної магістерської дисертації є розробка пристрою дистанційного керування силовим навантаженням на прикладі керування освітленням побутового приміщення, що поєднує використання локльної мережі зв'язку та програмного рішення для мікроконтролеру. Розумна лампа буде забезпечувати можливість ввімкнення, вимкнення та плавне регулювання яскравості освітлення за допомогою веб-сторінки або програмного рішення через смартфон чи настільний комп'ютер.

Такі рішення надають змогу економити час та витрачати менше зусиль в побуті або в промисловості, що потенційно призводить до збільшення ефективності при певних показниках швидкодії та дотриманні сучасних моделей проєктування веб-застосунків.

1 АНАЛІЗ НАЯВНИХ РІШЕНЬ ТА АЛГОРИТМІВ

Наведено огляд конструкцій розуміної лампи, а також пирницип роботи плавного перемикання потужності(яскравості лампи). Розглянуті приклади реалізацій окремих функцій для програмкних додатків.

1.1 Огляд існуючих конструкцій розумної лампи

Розумна лампа — пристрій для освітлення приміщень. Розберемо існуючі реалізації, а також їх переваги та недоліки. Для початку розглянемо можливий вигляд такої лампи і з чого вона складаються. На рис. 1.1 наведено приклад конструкції розумної лампи на основі світлодіодів.

Рисунок 1.1 — Конструкція компактних розумних ламп

Така конструкція містить в собі:

1 — захисний ковпак; 2 — набір світлодіодів; 3 —теплопровідна платформа; 4 — алюмінієвий радіатор; 5 — драйвер керування живленням світлодіодів; 6 — основа корпуса; 7 — цоколь.

Конструкція звичайної світлодіодної лампи має розміри лампи розжарювання, але у випадку розумних ламп розмір дещо більший у зв'язку із потребою виділення місця для розміщення драйверу керування живленням. Зазвичай такий драйвер містить в собі схему живлення, мікроконтролер для подачі керуючих сигналів, приймач та передавач, а також, за потреби, давачі для контролю роботи системи.

В теперішній час широкого вжитку набули лампи Yeelight [1], WIZ [2], MiPow [3]. На рис. 1.2 відображена конструкція лампи Yeelight.

Рисунок 2.2 — Конструкція розумної лампи Yeelight

В характеристиках вказано, що дана розумна лампа розрахована на потужність 10 Вт, містить модулі для дистанційного керування за допомогою мережі WiFi та Bluetooth, а її габарити складають — 130 мм х 65 мм х 65 мм.

Основний недолік лампи полягає у відсутності спеціального програмного забезпечення. Користувачам пропонують використовувати уніфікований додаток від сторонньої компанії, що призводить до некоректної роботи в деяких режимах.

На рис. 1.3 зображена розумна лампа компанії WIZ.

Рисунок 3.3 — Конструкція розумної лампи WIZ

В цілому досить схожий виріб, але відмінність полягає у методі керування освітленням, використовується окремий пульт, що незручно і викликає проблеми з пошуком пульта.

Останній наведений екземпляр — це лампа MiPow, яка зображена на рис.1.4.

Рисунок 4.4 — Конструкція розумної лампи MiPow

Характеристики в цілому схожі до характеристик минулих ламп, різниця полягає тільки у формі корпуса. Суттєвий недолік лампи полягає у використанні Bluetooth, що не дає можливості підключення до лампи у разі наявного підключення іншого девайсу. Також діапазон нормальної роботи мережі Bluetooth значно менший у порівнянні з мережею WiFi, що викликає незручності при суттєвих розмірах приміщення.

Загалом можна виділити такі основні проблеми, як вибір мережі для передачі даних та програмне забезпечення для керування розумними лампами, оскільки відсутні відкриті вихідні коди для аналізу додатків, представлених на ринку.

1.2 Метод димерування

Задля спрощення макету для тестування дистанційного керування силовим навантаженням на прикладі розумної лампи можна використати звичайну лампу розжарювання і, відповідно, розглянуті можливі методи реалізації периферії.

Пересічні розумні лампи підключаються до цоколя, а, відповідно, до мережі 220 В, а в цьому випадку буде доцільно використати живлення від мережі.

При аналізі інформації в Інтернет мережі, виявлено пристрій, який зветься димером. Він доволі розповсюджений при плавному регулюванні навантаження. Особливо часто його застосовують при керуванні освітленням. Розповсюдженість надає переваги в кількості інформації щодо методів його використання, що, в свою чергу, допомагає при розробці.

Досліджено основний принцип роботи димеру [4]. Як приклад розглянемо наступну схему реалізації, яка зображена на рис. 1.5.

Принцип дії полягає в роботі таких елементів, як симистор та схема “детектора нуля”.

У випадку подачі напруги на керуючий електрод симістор відкривається і, навпаки, закривається при її відсутності. У зв’язку із можливістю пропуску струму в обох напрямках відпадає потреба називати виводи симістора “Катод” і “Анод”. Тому часто у datasheet зустрічаються позначення A1, A2, та G — це керуючий електрод, походить від “gate” — ворота. Такий принцип дозволяє застосувати алгоритм регулювання фази. Під час синусоїдальної хвилі змінного струму симістор повністю відкривається. За допомогою ввімкнення

або вимкнення симістора на задану кількість мікросекунд можна регулювати вихідний сигнал, що і призводитиме до зміни потужності на виході.

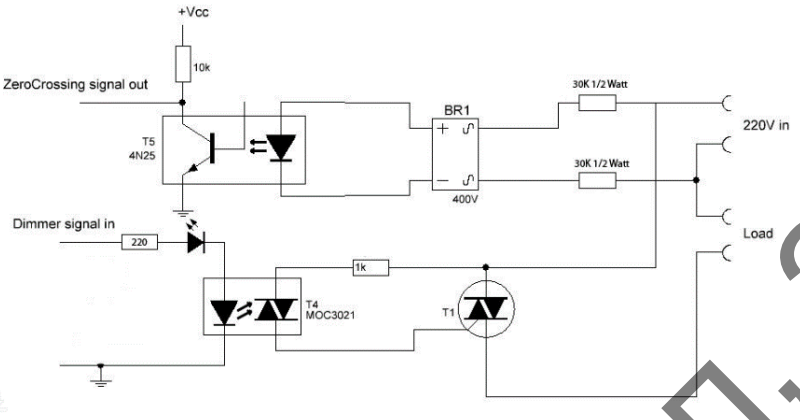


Рисунок 1.5 — Типова схема димеру

Основна проблема полягає в передбаченні точки на синусоїдальній хвилі, коли симістор буде відкриватись, що унеможливлює передбачення вихідної потужності, а значить і рівня освітлення. В зв’язку з цим виникає потреба у відслідковуванні, так званої “нульової” точки на синусоїдальній хвилі. Тому в схемі присутня частина, яка відповідає за детектування цієї точки на синусоїді. Ця схема зображена на рис. 1.6.

Відповідний сигнал передається на мікроконтролер в разі перетину синусоїдою осі абсцис. В свою чергу мікроконтролер подає відповідний керуючий сигнал на симістор. Таким чином, симістор відкривається на задану кількість мікросекунд.

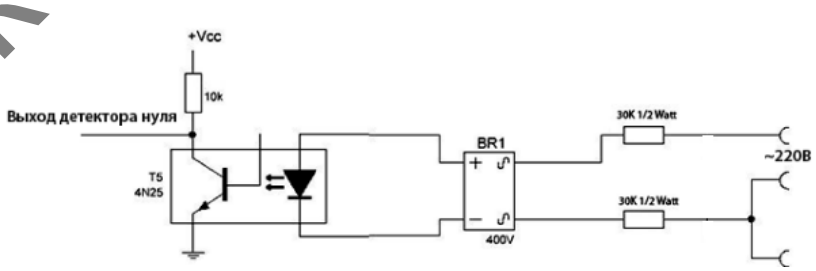


Рисунок 1.6 — Схема “детектора нуля”

В такому випадку, як зазначено на рис. 1.7, діодний міст забезпечує двофазне випрямлення. За допомогою гальванічної розв’язки сигнал перетину осі абсцис передається на певний контакт мікроконтролеру, що повідомляє про необхідність подачі сигналу відкриття симістора. Гальванічна

розв’язка реалізована за допомогою оптрона, що забезпечує захист мікроконтролера від перепадів напруги, зменшення рівня шумів тощо.

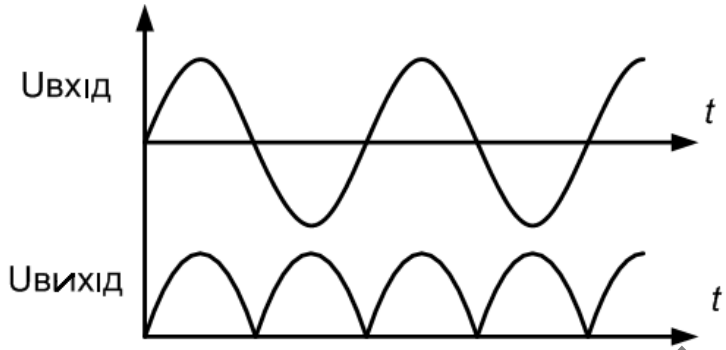


Рисунок 1.7 — Осцилограма сигналу випрямленого діодним містом

Результат роботи такої схеми можна побачити на рис. 1.8. Проводиться порівняння вхідного та вихідного сигналу при димеруванні в 50%.

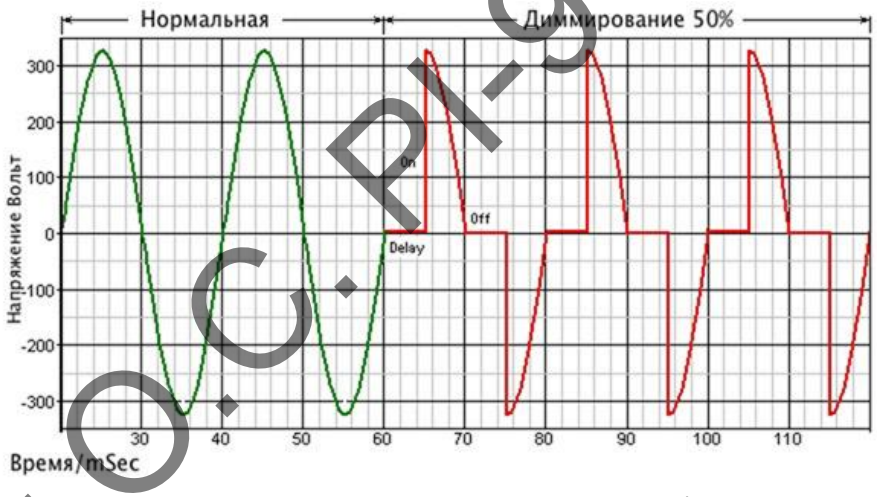


Рисунок 1.8 — Осцилограма синусоїди до та після димерування

За допомогою встановлення відповідної прошивки мікроконтролера можна легко регулювати час відкриття симістора, а значить і плавно регулювати вихідну потужність, тобто яскравість світіння лампи.

1.3 Твердотільне реле (SSR)

Перемикання навантаження, як і яскравості, можна реалізувати за допомогою однофазного твердотільного реле змінного живлення.

Твердотільне реле — це електричний комутаційний пристрій. Основна різниця з електромагнітним реле полягає в тому, що відсутні рухомі механічні частини, а також доступна більша швидкість перемикання і безшумність.

Принцип дії твердотільного реле полягає у взаємодії керуючого сигналу з керованим за допомогою гальванічної розв'язки – як правило, за допомогою оптрона. Керуюча напруга подає живлення на світлодіод, а він, в свою чергу, висвітлює фотодіод, і за допомогою струму останнього включається тиристор, що керує навантаженням.

Структура ТТР включає:

- вхід — первинний ланцюг, що складається з резистора на постійному ізоляторі, що має послідовне підключення; головною функцією вхідного ланцюга є прийняття сигналу і передача його пристрою реле, який комутує навантаження;
- оптична розв'язка — використовується для ізоляції вхідної та вихідної мережі змінного струму;
- тригерний ланцюг — окремий елемент, що обробляє вхідний сигнал і перемикає вихід;
- ланцюг перемикача — подає силу напруги; включає в себе транзистор, симистор і кремнієвий діод;
- ланцюг захисту — захищає пристрій від збоїв або появи помилок.

Для конструювання розумної лампи може підійти однофазне твердотільне реле [5] зображене на рис. 1.9.

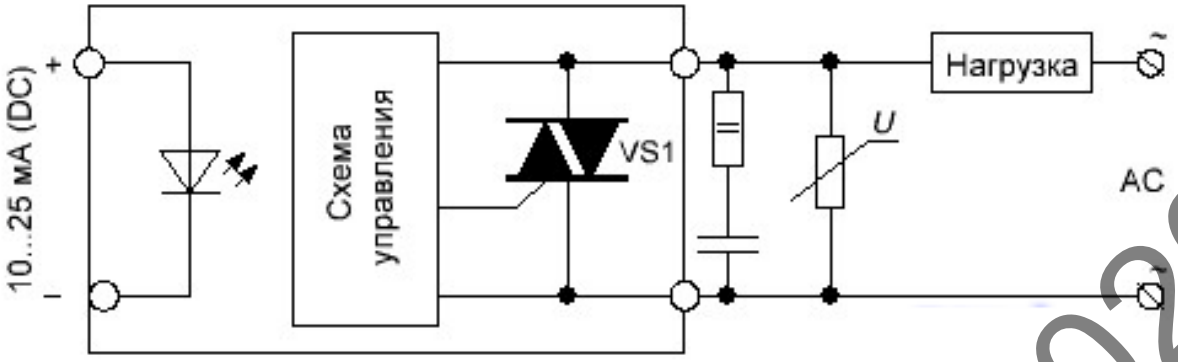


Рисунок 9.9 — Схема твердотільного реле

Власне підключити реле можна і до мікроконтролера, який буде забезпечувати зв’язок передавання керуючих сигналів через приймач і передавач від телефона або смартфона. Переваги таких реле відносно електромеханічних:

- висока швидкість перемикання;
- тривалий термін експлуатації через відсутність механічного або електричного зносу;
- безшумність.

В той же час до недоліків можна віднести:

- менш придатні до короткочасних скачків напруги, оскільки в основі лежать технології напівпровідникових пристроїв;
- сильне нагрівання при великих навантаженнях.

По суті, реалізації схожі між собою, але твердотільне реле найкраще підходить при керуванні інерційним навантаженням, таким як: мережевий обігрівач в кімнаті, паяльник або інфрачервона паяльна станція. Для таких потреб немає сенсу використовувати високочастотний ШІМ. Також, недоцільно розробляти димер в такому випадку, оскільки низькочастотний ШІМ та звичайне твердотільне реле вирішує цю проблему без створення відносно складної схеми із зайвим ускладненням коду для схеми “детектора нуля”.

Також використання залежить від габаритів. Власне схему димера можна реалізувати в меншому розмірі, що важливо при конструюванні розумної лампи, тому й доцільніше буде використати саме її.

1.4 Вибір мікроконтролера

Згідно із завданням магістерської дисертації пристрій повинен мати доступ до бездротових мереж зв'язку. На ринку присутні моделі мікроконтролерів в модифікаціях із вбудованими антенами, які забезпечують прийом і передачу даних бездротовим методом. Розповсюдженим і широко вживаним є мікроконтролер ESP8266 [6], що зображений на рис. 1.10.

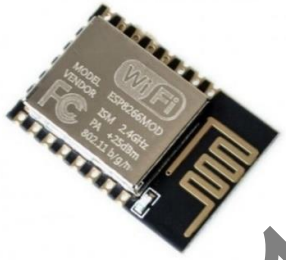


Рисунок 1.10 — Мікроконтролер ESP8266

Одна з основних характеристик даного мікроконтролеру полягає в габаритах – 15 мм х 25 мм. Це дозволить суттєво скоротити потрібне на "мозок" пристрою місце. ESP8266 підтримує роботу із WiFi IEEE 802.11 b/g/n. Ця технологія надає доступ до набору стандартів безпровідної передачі даних у локальних мережах частотного діапазону 2,4 ГГц при потужності сигналу не більше 100 мВт. Безпека та конфіденційність даних при передачі за допомогою локальних мереж забезпечується протоколами WEP (Wired Equivalent Privacy) та WPA/WPA2 (Wi-Fi Protected Access).

Даний мікроконтролер оснащений 80 МГц 32-bit процесором. За приблизними оцінками розмір оперативної пам'яті складає близько 80кБ (на жаль, виробник не надав точної інформації). Плати компанії Arduino, що використовують мікроконтролери Atmega328, забезпечені тактовою частотою процесора 16МГц та 32кБ оперативної пам'яті, тому ESP8266 має достатньо потужності, щоб підтримувати нормально оптимізований веб-сервер з можливість надання керуючих команд до периферії. При цьому, менші габарити забезпечують ергономічність конструкції корпусу.

Напруга живлення ESP8266 складає від 2,2 В до 3,5 В. Струм при передачі даних дорівнює близько 215 мА, а в режимі очікування — 70 мА.

Як вже зазначалось, живлення пристрою буде відбуватись від мережі 220 В, тому потрібно передбачити встановлення в конструкцію перетворювача із 220 В у потрібні нам межі, а саме від 3,3 В до 5 В.

Стоковий мікроконтролер ESP8266 не підтримується менеджером плат Arduino, тому необхідно буде використати сторонню модифікацію для підключення до даного IDE.

1.5 Аналіз програмних рішень

Задля виконання завдання магістерської дисертації потрібно розглянути можливі програмні реалізації методів димерування та методів передачі даних за допомогою мережі WiFi. Можливі декілька варіантів реалізації димерування. Вони полягають у використанні стандартних функцій програмного середовища для Arduino, а також використання сторонніх бібліотек.

1.5.1 Димерування на основі стандартних функцій

Як зазначалось раніше, димерування забезпечується сигналом із схеми “детектора нуля”, його подальшим обробленням мікроконтролером і віддачею сигналу для відкриття симістора на певну кількість мікросекунд.

Наступний лістинг демонструє можливу реалізацію стандартними методами:

```
int dimpin = 4;
char dim = 50;
void setup() {
pinMode(dimpin , OUTPUT);
attachInterrupt(0, light, FALLING);     }
void light() {
  if (dim > 0 && dim < 255) {
    delayMicroseconds(33*(255-dim));
    digitalWrite(dimpin , HIGH);
    delayMicroseconds(500);
```

```
    digitalWrite(dimpin , LOW);
  }
}
void loop() {
}
```

Даний лістинг доволі просто демонструє принцип дії димерування, але недолік такого методу полягає в неефективності на практиці. Справа в тому, що лампа при малих значеннях змінної dim починає неконтрольовано мигати, що не відповідає потребам користувача.

1.5.2 Димерування на основі основі бібліотеки "CyberLib"

У наступному лістингу розглянуто реалізацію на базі бібліотеки "CyberLib" [7]:

```
#define dimPin 4
#define zeroPin 2
#include <CyberLib.h>
volatile int tic, Dimmer;
   void setup() {
Serial.begin(9600);
pinMode(dimPin, OUTPUT);
pinMode(zeroPin, INPUT);
attachInterrupt(0, detect_up, FALLING);
StartTimer1(timer_interrupt, 40);
StopTimer1();
Serial.println("Start");
}
void loop() {
Dimmer = map(analogRead(0), 0, 1023, 240, 0);    }
void timer_interrupt() {          tic++;
if (tic > Dimmer)
digitalWrite(dimPin, 1);
}
void  detect_up() {
tic = 0;
```

```
ResumeTimer1();
attachInterrupt(0, detect_down, RISING);
}
void  detect_down() {  tic = 0;
StopTimer1();
digitalWrite(dimPin, 0);
attachInterrupt(0, detect_up, FALLING);  }
```

Це вже більш розвернута реалізація, однак такий лістинг забезпечує кращу швидкодію. Справа в тому, що алгоритм на основі бібліотеки “CyberLib” використовує апаратний лічильник мікроконтролера, забезпечуючи покращення показників швидкодії виконання команд майже в 2 рази. До недоліку можна віднести те, що не всі мікроконтролери можуть коректно працювати з апаратним лічильником з допомогою даного алгоритму, тому цей спосіб, у разі винекнення помилок, може призвести до виведення з ладу мікроконтролеру, а відповідно і подальшої перепрошивки або заміни мікроконтролеру.

1.5.3 Димерування на основі основі бібліотеки “AC_Dimmer”

Далі продемонстровано лістинг реалізації димерування на основі бібліотеки “AC_Dimmer” [8]:

```
#include <AC_Dimmer.h>
#define Dimmer_1 0
void setup()
{
Dimmer_init_begin();
Dimmer_pin_assign(Dimmer_1, 3);
Dimmer_init_end();
}
int adc_read;
void loop()
{
adc_read = analogRead(A4)/4;
```

```
Dimm_value(Dimmer_1,  adc_read);
delay(50);
}
```

Лістинг на основі цієї бібліотеки доволі спрощений відносно минулої ралізації. Зрозуміло, що основний алгоритм захований в коді функцій. Оскільки цей алгоритм розробляли сторонні програмісти спеціально під певні моделі димерів на ринку, на практиці це дає впевненість у відносно великих показниках надійності даної реалізації і забезпечення потрібної швидкодії, а також відсутність мерехкотінь.

1.5.4 Передача даних на ESP8266

Стандартні методи роботи програмного середовища не дозволяють використовувати можливості ESP8266, тому ці функції можуть бути забезпечені готовою бібліотекою “ESP8266WebServer” [9].

Дана бібліотека дає можливість обрати один із можливих режимів роботи мікроконтролеру: точка доступу, міст, користувач. Загалом, найзручніший метод роботи пристрою полягає в самостійному підключенні до WiFi мережі. Тому наступний лістинг демонструє реалізацію режиму підключення, як користувач:

```
void setup() {
Serial.begin(115200);
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.hostname("brightness_control");
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED)
delay(500);
Serial.print(".");
  }
Serial.println("");
Serial.println("WiFi connected.");
Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
```

```
 server.begin();
}
```

За нового підключення мікроконтролера до живлення, який виконує роль пристрою керування силовим навантаженням, в функції setup() відбуватиметься налаштування підключення пристрою до вказаної WiFi мережі. При цьому, варто розглянути рядки, що виводять до COM-port дані.

COM-port — це послідовний інтерфейс передачі (адже інформація передається біт за бітом) даних. При проектуванні веб-застосунків часто використовують логування взаємодії з веб-сервером. Такий механізм і відтворено тут, що дозволяє переглядати необхідні налаштування на сервері, як наприклад IP-адресу, яка присвоєна веб-серверу в межах локальної мережі.

IP-адреса — це ідентифікатор, необхідний для адресації host(пристроїв з мережевою картою) у мережах. В глобальній мережі Інтернет IP-адреса може бути повністю унікальною. Також пристрій може мати неунікальну адресу в глобальній мережі інтернет, проте зобов'язаний мати унікальну локальну адресу. Є певні стандарти для більшості локальних мереж різних операторів, хоча інколи бувають розходження. Як правило, у локальних мережах перший октет IP-адреси страндартизований, як наприклад 192, 10 та інші можливі варіанти. Зазвичай пристрої в локальних мережах працюють за системою DHCP (Dynamic Host Configuration Protocol) — мережевий протокол, який дозволяє host автоматично отримувати унікальну IP-адресу в межах локальної мережі. Проте деякі користувані використовують статичні конфігурації. В такому випадку IP-адреса буде зарезервована за певним host. В більшості випадків веб-сервер у зв’язку із DHCP буде отримувати іншу IP-адресу при кожному новому підключенні.

Таким чином, сформовано наступну проблематику:

- застарілі алгоритми комунікації із периферією;
- відсутність методу детектування веб-серверу в межах локальної мережі;
- відсутність прикладів вихідного коду застосунків у вільному доступі.

Це потребує проведення аналізу засобів розробки, а також розробки веб-додатків, що включатимуть алгоритм пеленгування веб-серверу.

2 ТЕХНОЛОГІЇ ДЛЯ ПРОЕКТУВАННЯ ПРОГРАМНОГО РІШЕННЯ

Насамперед потрібно визначитись із використанням потрібних технологій. Як вже зазначалось, є можливість розвернути веб-сервер на мікроконтролері.

2.1 Стек технологій та моделей для мікроконтролеру

Запрограмувати плату (на базі мікроконтролеру ESP8266), дозволяє платформа Arduino IDE (Integrated Development Environment) — інтегроване середовище розробки для Windows, MacOS і Linux, розроблена на C і C++ і призначена для створення і завантаження програм на Arduino-сумісні плати, а також на плати інших виробників.

Вихідний код для середовища випущений під загальнодоступною ліцензією GNU (вільна UNIX-подібна операційна система) версії 2. Підтримує мови C і C++ з використанням спеціальних правил структурування коду. Arduino IDE надає бібліотеку програмного забезпечення, яка пропонує безліч загальних процедур введення і виведення. Для написаного користувачем коду потрібні тільки дві базові функції для запуску ескізу і основного циклу програми, які скомпільовані і пов'язані з заглушкою програми main() у виконувану циклічну програму разом із ланцюжком інструментів GNU, також включених в дистрибутив IDE. Використовується програма avrdude для перетворення коду, в текстовий файл в шістнадцятковій системі кодування, який завантажується в плату Arduino програмою-завантажувачем у вбудованому програмному забезпеченні плати.

З ростом популярності Arduino, інші постачальники в якості програмної платформи почали впроваджувати призначені для користувача компілятори і інструменти з відкритим вихідним кодом (ядра), які можуть створювати і завантажувати ескізи в інші мікроконтролери, що не підтримуються офіційною лінійкою мікроконтролерів Arduino.

Власне для плат на базі мікроконтролерів ESP8266 розроблена бібліотека для менеджеру плат, яка дозволяє користуватись Arduino IDE за допомогою стандартних функцій.

Такі технології дозволяють застосувати ESP8266 тільки як звичаний мікроконтролер з набором керування пінами. В нашому випадку ще необхідно забезпечити передачу даних.

Для підключення мікроконтролеру до WiFi мережі, а також надання подальшої можливості розгортання веб-серверу із функціями обробниками надсилання і отримання запитів, доцільно використати сторонню білбліотеку “ESP8266WebServer”.

В оглядовому розділі зазначалось, що для реалізації потреби димерування можна використати готову сторонню бібліотеку. Оскільки для макету було обрано плату димерування RobotDyn, то для даної плати в комплекті розробник надає універсальну бібліотеку, яка працює з більшістю інших димерів, а саме “RBDdimmer” [10].

Тепер детальніше можна розібрати бібліотеку “ESP8266WebServer”. Як вже зазначалось, в ній містяться функції, які дозволяють приєднатись до WiFi мережі, а також розгорнути веб-сервер. Початкова гілка доступу для звичайного користувача буде зберігатись на звичайній веб-сторінці. Ця бібліотека знову стає у нагоді, адже дозволяє ініціалізувати вихідний код веб-сторінки, як окрему змінну в скетчі мікроконтролера. В цій змінній буде зберігатись як лістинг інтерфесу сторінки, так і частина, яка відповідає за логіку. За допомогою наданої функції буде можливість виклику надання доступу до цієї веб-сторінки за запитом в тілі основної функції скетчу мікроконтролера.

2.2 Ініціалізація веб-сторінки та дослідженян моделі AJAX

Для ініціалізації веб-сторінки слід розпочати із HTML/CSS.

HTML (HyperText Markup Language) [11] — стандартизована мова гіпертекстової розмітки веб-сторінок у мережі Інтернет. Вихідний код HTML інтерпретується браузерами і, як результат, виводиться у вікно браузера в гра-

фічному форматі. Специфікації HTML5 формулюються в термінах моделі DOM (Об'єктній моделі документа). Будь-який документ відомої структури за допомогою DOM може бути представлений у вигляді дерева вузлів, кожен вузол якого є елементом, атрибутом або текстовим, графічним чи будь-яким іншим об'єкт. Вузли пов’язані між собою відносинами «батьківський-дочірній». Приклад ієрархії об'єктів в DOM зображений на рис. 2.1.

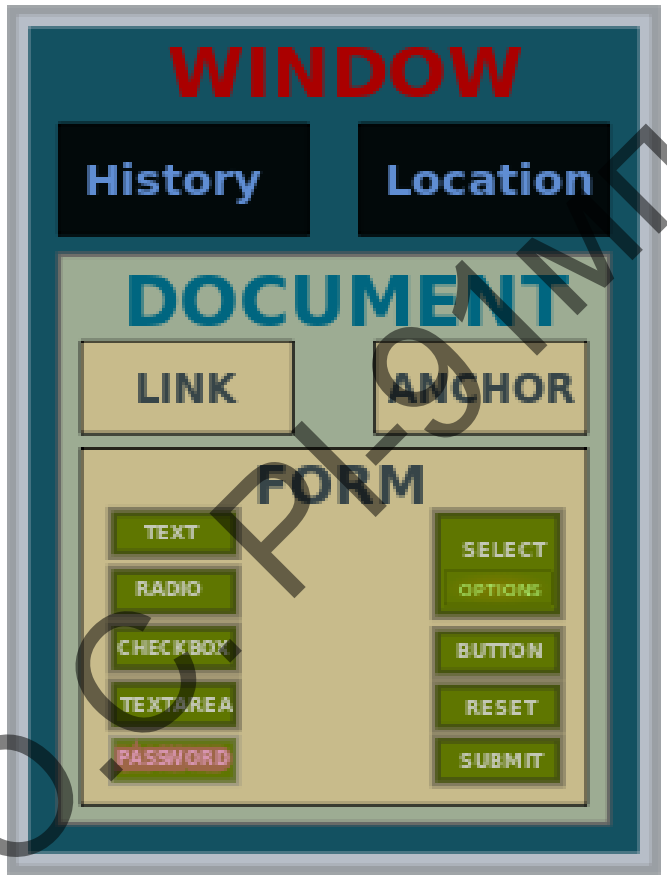


Рисунок 2.1 — Ієрархія об’єктів HTML в моделі DOM

Така модель чітко стандартизує шари коду сторінки, а також розподіляє вміст сторінки за допомогою симантично правильних тегів.

Структура HTML сторінки у вихідному коді ділиться на дві основні частини:

```
<! DOCTYPE html>
<html>
    <head>
        …
    </head>
```

```
<body>
  …
</body>
</html>.
```

Тег <head> повинен містити в собі інформацію заголовка веб-сторінки та іншої технічної інформації, в основнму “закритої” від звичайного користувача. В свою чергу тег <body> містить теги, які відповідають за наповнення сторінки.

Таким чином можна заповнювати сторінку текстом, таблицями, або картинками за допомогою синтаксису:

```
<p>ТЕКСТ</p> ,
```

де — початок тегу(в даному випадку означення параграфу);
 — закінчення тегу.

Все, що знаходиться між тегами відповідає змісту, який буде виведений на екран. Також в дані теги можна додати методи стилізації або присвоєння певних ідентифікаційних номерів:

```
<p><font id=”x” size="5" color="red" face="Arial">ТЕКСТ</font></p>,
```

де – тег налаштування шрифта; size=”5” – розмір шрифта; color=”red” – колір шрифта; face=”Arial” – вибір шрифта; id=”x” – присвоєння ідентифікаційного номеру даному тегу. В подальшому присвоєння парметру id надає можливість звернення до даного тегу з іншої частини лістингу. Такий функціонал надає гнучкість при розробці сторінки, дозволяє рознести по симантично правильним частинам різні функції, що сприяє зручності при розробці, а також відловлюванні помилок у вихідному коді при тестуванні.

Виявляється, HTML відповідає за наповнення сторінки вмістом. Для того щоб надати сторінці певного вигляду, використовують CSS.

CSS (Cascading Style Sheets) [12] — мова опису зовнішнього вигляду веб-сторінки, яка написана за допомогою мови розмітки, в нашому випадку HTML. Також її можна застосувати до будь-яких XML-документах, наприклад, до SVG або XUL.

CSS використовується розробниками веб-сторінок для задання кольорів, шрифтів, стилів, розташування окремих блоків та інших можливих реалізацій зовнішнього вигляду веб-сторінок. Основною метою розробки CSS було відділення опису структури сторінки від методів забезпечення зовнішнього вигляду. Такий поділ забезпечує простоту у читанні лістингу при розробці, тестуванню або доопрацюванні сторінки. Також це забезпечує зменшення повторювань одних і тих самих налаштувань для різних блоків сторінки.

Підключити CSS до вихідного коду сторінки можна в HTML документі наступними чином:

```
<head>
    <link rel = "stylesheet" type = "text/css" href = "style.css">
</head>
```

Такий приклад надає можливість прописати стилі в окремому документі і задає посилання на цей документ. Такий метод зручний, коли документ сторінки доволі об'ємний і містить багато різних тегів. За допомогою ідентифікаційного номеру з окремого CSS документа можна звернутись до потрібних блоків сторінки таким чином:

```
#x {
font-size: 5;
color: red;
font-family: Arial;
}
```

Наступного разу при використанні id “x”, блокові веб-сторінки автоматично буде присвоєні наступні конфігурації стилів. Схожим чином можна реалізувати звернення до повторюваних блоків, замінивши #x на p (відповідатиме всім тегам параграфу на сторінці).

У випадку, якщо структура веб-сторінки не розростається до великих масштабів або немає можливості підключити сторонній документ, можна прописати стилі CSS прямо в тезі <head>, наприклад:

```
<head>
    <style>
```

```
        #p {
          color: red;
        }
      </style>
    </head>
```

Оскільки вихідний код веб-серверу на мікроконтролері пишеться в межах одного документу без можливості підключення сторонніх, всі стилі будуть прописуватись безпосередньо в HTML документі в тезі <head>.

Для зв'язку інтерфесу, а саме функціональних клавіш або слайдеру із веб-сервером використаємо мову програмування JavaScript.

JavaScript [13] — мультипарадигмова мова програмування, яка підтримує об'єктно-орієнтований, імперативний і функціональний стиль. Програма є реалізацією стандарту ECMAScript. Як правило, вона застосовується у вигляді мови сценаріїв для забезпечення інтерактивності веб-сторінки.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу. На відміну від інших мов програмування, JavaScript має наступні переваги:

- об'єкти з можливістю інтроспекції;
- функції як об'єкти першого класу;
- динамічне приведення типів;
- автоматичне прибирання сміття;
- анонімні функції.

Однак, в межах цієї мови є також і нереалізовані корисні речі, як:

- стандартна бібліотека, зокрема відсутня можливість роботи з файловою системою, управління потоками введення-виведення;
- стандартні інтерфейси до веб-серверів і баз даних;
- система управління пакетами, яка відслідковує залежності та автоматично встановлює їх.

Проте в нашому випадку ці недоліки незначні, адже немає потреби працювати з файлами або потоками введення-виведення. Додати логіку веб-сторінки на JavaScript можна за допомогою посилання на окремий файл:

```
<head>
    <script type = "application / javascript" src="file.js">
    </script>
</head>
```

У випадку, якщо сторінка потребує об'ємного опису логіки або сценаріїв, а також якщо потрібно прописати прямо в коді сторінки:

```
<script type="application/javascript">
     alert('Hello, World!');
</script>
```

JavaScript в основному потрібен для відправлення запитів до веб-серверу. Для поєднання технологій, описані вище, використаємо термін AJAX.

AJAX (Asynchronous JavaScript and XML) [14] — так званий асинхронний JavaScript та XML, що, по суті, є не технологією, а моделлю, яка описує “новий” принцип використання технологій JavaScript, HTML, CSS, DOM, XML разом. Також в основі лежить робота з об'єктом XMLHttpRequest. Модель AJAX дозволяє веб-додаткам (додаткам на смарфон, ПК, веб-сторінкам в браузері) коригувати вміст (інтерфейс) веб-сторінки без потреби в постійному перезавантаженні. В такому випадку веб-застосунки стають швидшими та гнучкими.

На практиці за допомогою об'єкта XMLHttpRequest надається можливість спілкування з веб-сервером через HTTP протокол, а також зміни статусів функціональних клавіш, шматків тексту або блоків сторінки без перезавантажень веб-сторінки.

HTTP(HyperText Transfer Protocol) [15] — поширений протокол передачі даних, призначений для передачі гіпертекстових документів, та дозволяє організовувати переходи до інших документів. Він відповідає протоколу прик-

ладного, 7-го рівня, відповідно до мережевої моделі OSI [16], зображеної на рис. 2.2.

Рисунок 2.2 — Мережева модель OSI

Наразі HTTP протокол використовує клієнт-серверну структуру для передавання даних. Програма клієнта формує запит і надсилає на серевер, серверне обладнання приймає, опрацьовує та формує відповідь і надсилає назад до клієнта, після чого знову можливий обмін новими запитами та відповідями, які будуть оброблятись аналогічним чином.

При цьому дані, які будуть передаватись, можуть мати будь-який формат, наприклад JSON або XML.

В основному передача даних по протоколу HTTP здійснюється за допомогою TCP/IP з’єднання. За замовчуванням серверне програмне забезпечення використовує порт 80, хоча взагалі можна використати інший вільний порт.

За допомогою об'єкту XMLHttpRequest можна відправляти або отримувати дані, як синхронно так і асинхронно. Структура запитиів зображена на рис. 2.3.

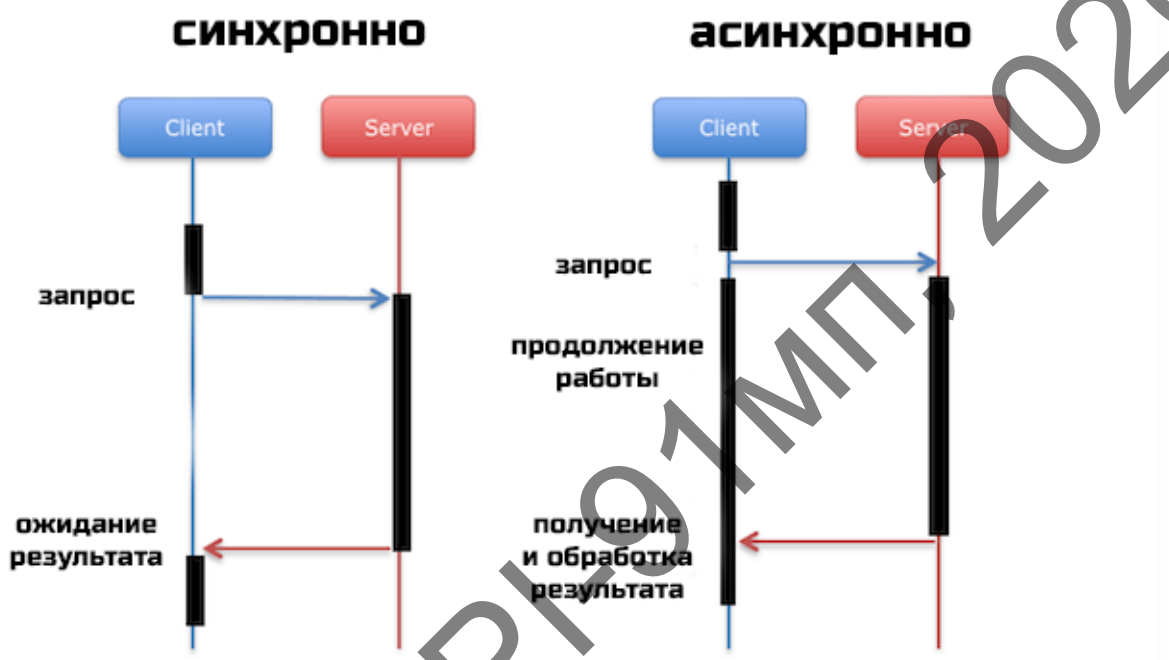


Рисунок 2.3 — Структура запитів клієнт-сервер

Власне принцип роботи синхронного запиту полягає в тому, що на момент відправлення запиту сценарій веб-сторінки зупиняє свою роботу, поки не отримає відповідь від серверного програмного забезпечення.

У випадку асинхронного запиту клієнт надсилає запит і надає доступ до вед-додатку іншим функціям або блокам сторінки. Коли надійде відповідь, функція «відстежував» надасть пріоритет обробці відповіді і знову дасть можливість спрацьовувати іншим функціям. Ця різниця дозволяє сформувати підхід, при якому під час комунікації з веб-сервером, користувач може виконувати інші справи на веб-сторінці без потреби очікувати закінчення очікування відповіді.

Наступний лістинг демонструє приклад відправлення асинхронного запиту:

```
function reqListener () {
```

```
  console.log(this.responseText);
}

var oReq = new XMLHttpRequest();
oReq.onload = reqListener;
oReq.open("get", "yourFile.txt", true);
oReq.send();
```

Спочатку за допомогою new XMLHttpRequest() створюється об'єкт. Методи виконують наступні функції:

- onload() — викликає функцію обробки відповіді;
- open() — відкриває запит, вказується тип (get/post), дані, асинхронний/синхронний;
- send() — відправлення запиту.

Зрозуміло, що синхронні запити використовують доволі рідко, в основному, коли запит не потребує значних ресурсів і завідомо буде виконаний швидко.

2.3 Стек технологій для ініціалізації застосунків на смартфон та ПК

Тепер перейдемо до додатку на комп’ютер. Для призначень на платформі Windows часто використовують IDE Visual Studio.

Дане IDE містить редактор вихідного коду з підтримкою технології IntelliSense і надає засоби найпростішого рефакторингу коду. Включений відладчик може працювати відладчиком рівня вихідного коду, а також відладжувати машинний код. Включені редактори форм для спрощення створення графічного інтерфейсу програмного забезпечення, веб-редактор, а також дизайнер класів і структури баз даних, що спрощує роботу програміста. Також наявна можливість створення і підключення сторонніх додатків, бібліотек, плагінів, тощо. Visual Studio надасть можливість проектування, як додатку на ПК, так і на смартфон.

Для додатку на ПК будуть використані наступні технології: XAML[17], C#[18]. Принцип схожий до розробки веб-сторінки, за інтерфейс застосунку буде відповідати XAML, а за логіку C#.

XAML (eXtensible Application Markup Language) — це мова розширеної розмітки додатків. Синтаксис схожий до HTML/CSS. Наступний лістинг демонструє приклад розмітки на XAML:

```
<WrapPanel Height="25" Width="25"> </WrapPanel>
```

Приклад показує, що синтаксис в цілому такий самий, але відрізняються назвами самих тегів і структурою документу. Відсутні, як такі, теги <head>, <body>. Є місце для вказування технічної інформації, а потім розділення іде в межах сітки додатку з додатковими панелями всередині для формування наповнення та інтерфейсу. В цій мові також розповсюджено використання ідентифікаційних номерів елементів, проте тут це зазначається наступним чином:

```
<Slider x:Name="x"/>
```

Оскільки для додатку на смартфон та ПК є можливість рознести вихідний код, то логіка на мові C# буде просто підкріплена до проекту окремим файлом.

C# — це об’єктно-орієнтована мова програмування, розроблена для забезпечення можливості проектування веб-додатків для Windows. Багато в чому синтаксис повторює мови програмування C++ та Java (не путати з JavaScript). Ця мова також підтримує методи роботи з синхронними і асинхронними запитами.

Для програмного забезпечення на смартфон буде використано технологію Xamarin Forms [19], яка є доступною в пакеті Visual Studio. Дана платформа дозволяє швидко розробляти cross-platform додатки на смартфони різних брендів, а відповідно різних операційних систем (Android, IOS), за допомогою одного коду на весь додаток. Хоча деякі додаткові функції доведеться прописувати окремо, задля коректної роботи додатку, Xamarin, як мову розмітки, використовує XAML. Зміни в синтаксисі обумовленні особливостями

позиціонування додатку на екрані смартфону. Відповідно, деякі теги мають інший механізм розташування на екрані та дуже рідко використовується абсолютне позиціонування в пікселях відносно певного блока інтерфейсу. Натомість більше в нагоді стають розширювані блоки, які автоматично підбираються під розміри екрану. Для логічної частини додатку на смартфон також використовують C# з відповідними змінами в алгоритмах.

Таким чином, в рамках моделей DOM, AJAX, OSI обрані необхідні мови розміток для забезпечення відклику інтерфейсу, а також мови програмування для реалізації необхідних алгоритмів передачі даних. Проаналізовані програмні середовища Arduino та Visual Studio, які пришвидшують цикл розробки програмного забезпечення.

3 РОЗРОБКА СТРУКТУРНОЇ СХЕМИ ТА МОДЕЛЮВАННЯ БЛОК-СХЕМ АЛГОРИТМІВ

Розробка веб-додатків потребує розробки структурної схеми проєкту для правильного симантичного розділення функцій. Для запобігання помилок в вихідному коді проведена розробка структурних схем алгоритмів додатку на ПК, смартфон та веб-сторінки.

3.1 Розроблення структурної схеми проекту

Структурна схема проекту, що розроблена згідно із завданням магістерської дисертації, наведена на рис. 3.1.

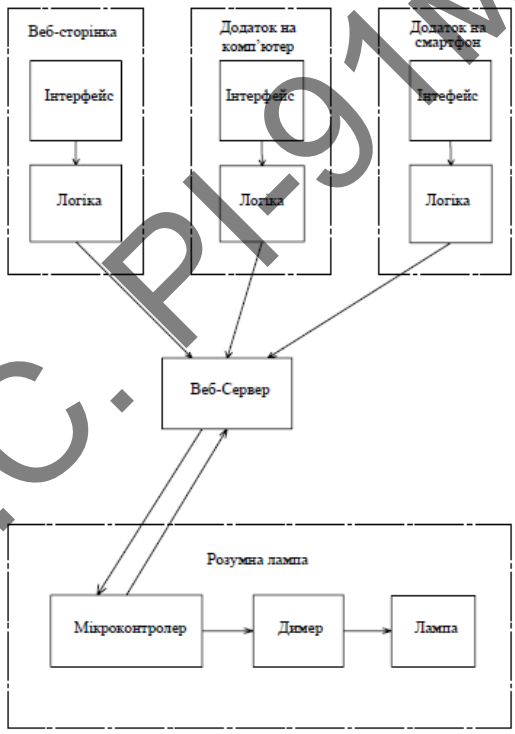


Рисунок 3.1 — Структурна схема проекту розумної лампи

Структурну схему можна умовно поділити на реальну та віртуальну частини. Реальна – це сам пристрій розумної лампи, а віртуальна – методи надання доступу користувачу для вказання бажаного режиму роботи пристрою, тобто веб-сторінка, додаток на комп'ютер, додаток на смартфон. Взаємодія між цими частинами відбувається за допомогою веб-серверу, розвернутому на базі мікроконтролера ESP8266.

Отже, веб-сервер має можливість отримання доступу до мережі WiFi і розроблені функції для основних керуючих сигналів. Також сформоване або стандартизоване отримання запитів на конкретні дії від додатків та веб-сторінки. Потім мікроконтролер передає відповідний сигнал на димер.

3.2 Моделювання схеми алгоритму веб-серверу

Відповідно до завдання, лістинг на базі мікроконтролера реалізує наступні функції та відповідає алгоритму, вказаному на схемі на рис. 3.2.

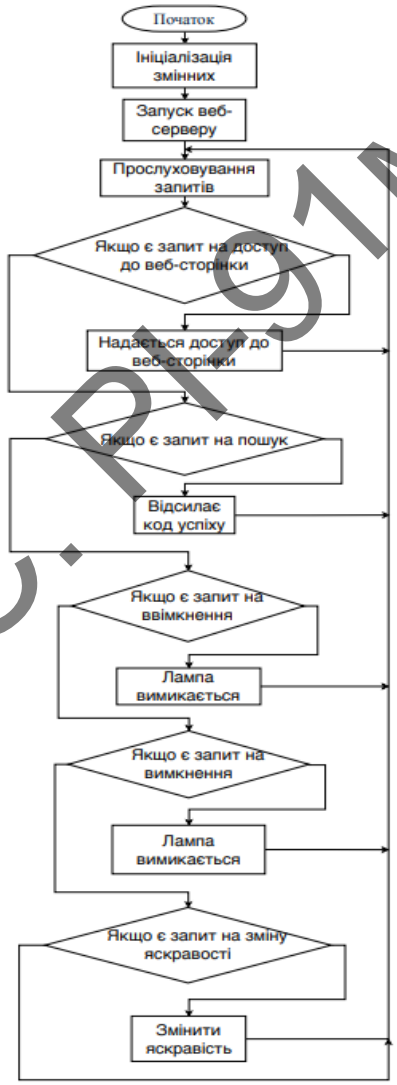


Рисунок 3.2 — Структурна схема алгоритму веб-серверу

В ініціалізації даних вказано підключення вхідних та вихідних контактів, змінні та функції потрібні для роботи алгоритму. Також в процесі ініціалізації даних як окремою змінною вказується код веб-сторінки, яка розгорту-

ється і надає змогу користувачу надсилати запити або моніторити ситуацію на пристрої.

Після ініціалізації відбувається підключення до мережі WiFi і подальше розвертання вже ініціалізованої веб-сторінки. За допомогою цикла відбувається прослуховування мережі на надходження запитів від веб-сторінки.

В алгортимі присутні 5 основних функції керування:

- запит на доступ до веб-сторінки;
- запит на пошук веб-серверу в мережі WiFi;
- ввімкнення лампи;
- вимкнення лампи;
- прийняття значення необхідної потужності(яскравості), яке подається користувачу у відсотках від 0% до 100%.

Запит на пошук веб-серверу в мережі WiFi необхідний для визначення IP адреси веб-сервера та подальшої комутації між ним та додатками на смартфоні та ПК. До наведених функцій можна додати і більше, однак, в цілому, на основі даних функцій можна відтворити більшість можливих потреб просто додавши декілька стрічок коду.

Як видно з схеми, в деяких функціях, окрім прийняття запитів, використовується ще й відсилання повідомлень про успіх або дозвіл на використання веб-сторінки.

3.3 Моделювання схеми алгоритму веб-сторінки

Як вже зазначалось, в скетчі мікроконтролера в розділі ініціалізації даних прописаний лістинг веб-сторінки. Структурна схема веб-сторінки зображена на рис. 3.3.

Код цієї сторінки проводить початкову ініціалізацію даних, які потрібні для роботи, і починає відслідковувати натиски на клавіші, які вказані в графічному інтерфейсі і викликає відповідні функції. Задача кожної з функцій полягає в надсиланні запиту з потрібними діями на веб-сервер. Ці функції

отримують відповіді від веб-серверу і змінюють відповідно до нього стан клавіш, тим самим повідомляють користувача про виконану дію.

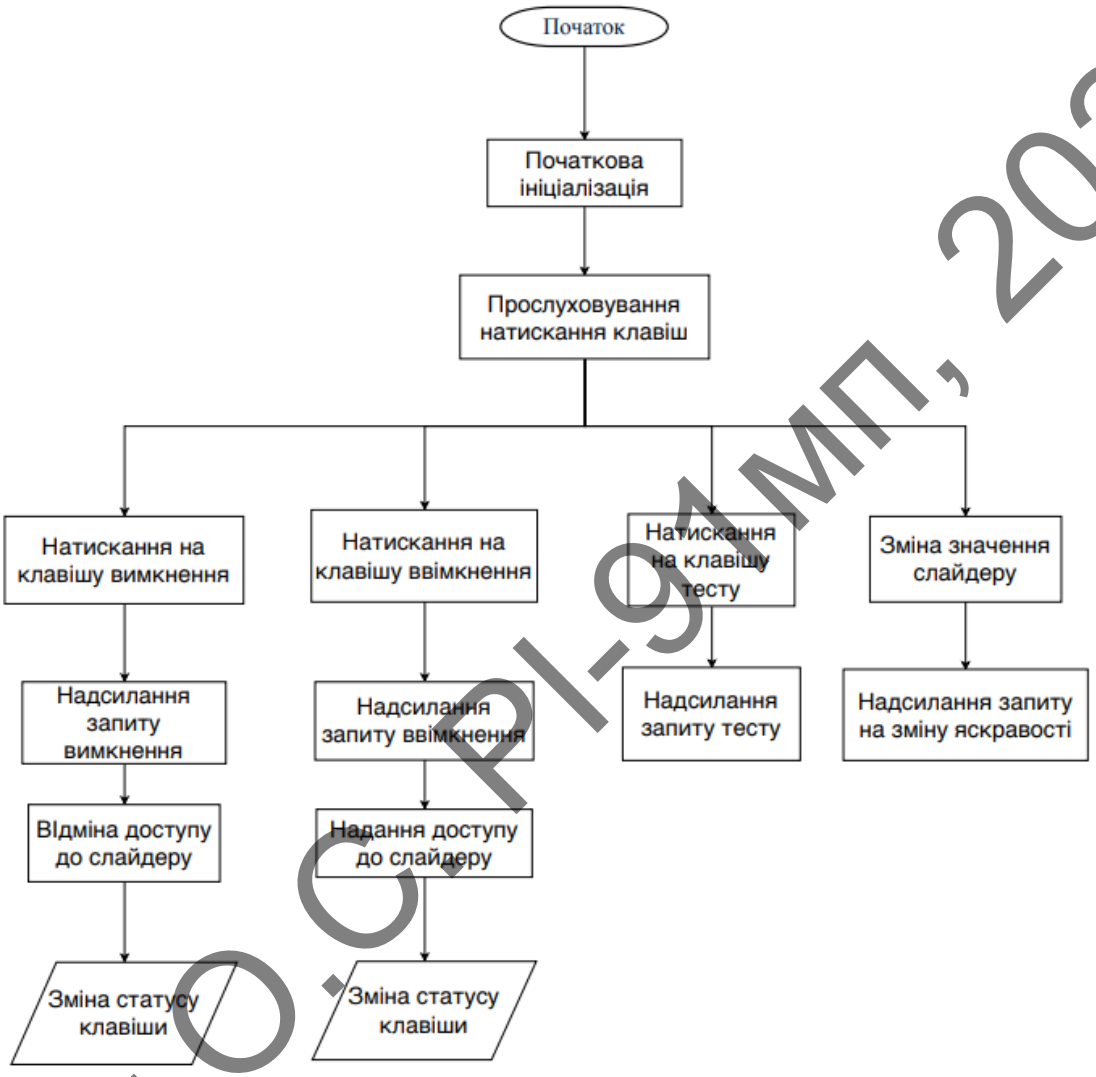


Рисунок 3.3 — Структурна схема алгоритму веб-сторінки

В цей раз алгоритм надає 4 основні можливості:

- відправлення запиту на ввімкнення;
- відправлення запиту на вимкнення;
- відправлення запиту із вказаним значенням яскравості (потужності);
- відправлення запиту на тестове ввімкнення (плавна зміна потужності від 0 до 100).

На веб-сторінці вибір потрібної яскравості лампи відбувається за допомогою “слайдера”. Значення передається відповідно до зміни положення повзунка.

3.4 Моделювання схеми алгоритму для смартфону та ПК

З огляду на функціонал, додаток на смартфон або ПК не може суттєво відрізнятись, тому доступні ті ж функції, що і на веб-сторінці. На рис. 3.4 зображена структурна схема алгоритму для ПК і, як бичимо, різниця незначна. Але є суттєва різниця між принципом передачі запиту від веб-сторінки, розгорнутої на базі веб-серверу та додатку. Справа в тому, що для надсилання запиту від додатку потрібно вказувати IP адресу отримувача, тобто веб-серверу. Стандартно користувач не знає цієї адреси, а отже додаток повинен мати алгоритм пошуку веб-серверу в мережі WiFi.

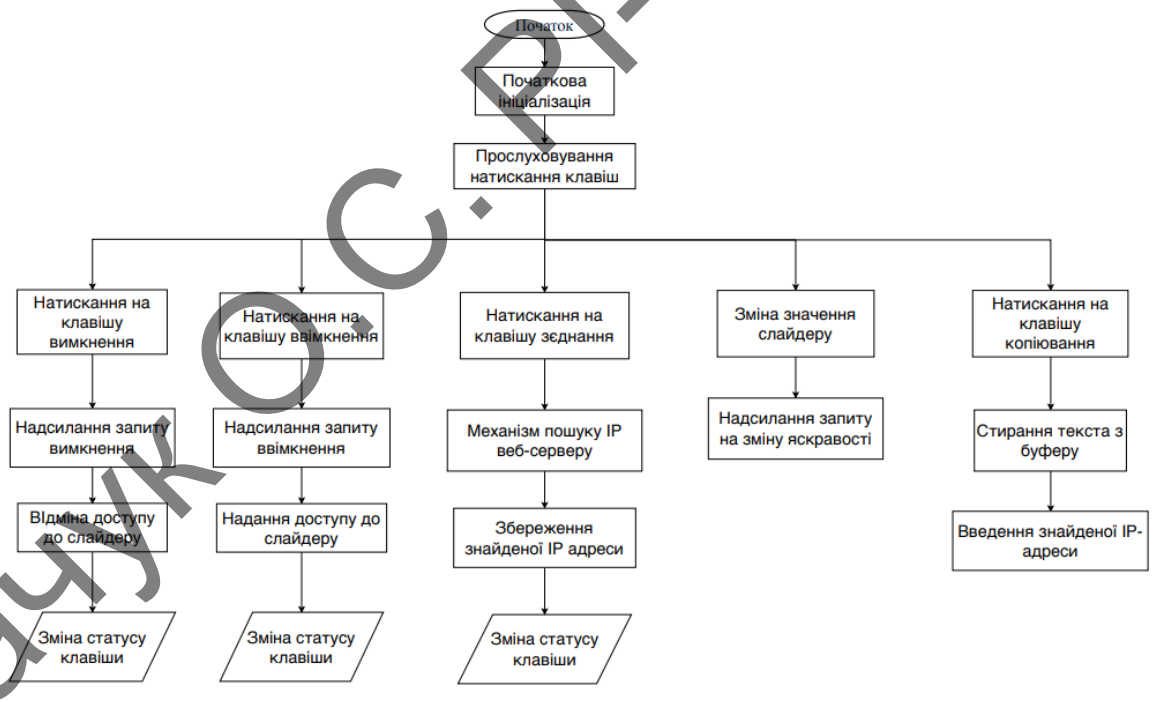


Рисунок 3.4 — Схема алгоритму додатку на ПК

Алгоритм додатку видає помилку в разі відсутності “з'єднання”. Таким чином першочергово необхідно активувати функцію пошуку, після того стає можливо надіслати інші запити. Також додано клавішу копіювання адреси з подальшим можливим збереженням і наданням доступу до веб-сторінки (як-

що так зручніше) за умови, якщо пристрій не був перепідключений до іншої мережі. При перепідключенні адреса змінюється.

Різниця в алгоритмі для додатку на ПК та смартфон проявляється в функції з'єднання, що зображено на рис. 3.5.

Рисунок 3.5 — Гілка структурної схеми алгоритму для смартфону

У зв'язку із менш ефективними можливостями смартфона швидко знайти IP адресу веб-сервера, було вирішено додати можливість пріоритетного підключення. Це означає, що додаток перезаписуватиме IP адресу останнього підключення і при кожному з'єднанні перевірить спочатку минулу адресу. Тільки після неможливості підключення до старої адреси буде задіяний алгоритм пошуку. В теорії, така реалізація додатку на смартфон дозволяє суттєво зекономити час.

Розроблена структура проекту надає чітке бачення взаємодії віртуального та фізичного обладнання. Проведено моделювання структурних схем алгоритмів для різних веб-застосунків. В його ході були виокремлені базові функції, які доступні користувачу, а також розглянута різниця алгоритму для різних платформ, як наприклад модифікація алгоритму пеленгування на ПК та смартфоні.

4 ПРОЕКТУВАННЯ ПРОГРАМНИХ ЗАСТОСУНКІВ

Згідно із розробленими структурними схемами алгоритмів для додатків відбулось проектування програмних рішень для цих цілей.

4.1 Проектування веб-серверу та драйверу мікроконтролера

У зв’язку із успішно проведеним аналізом можливих рішень, в наявності готові програмні рішення для підключення до WiFi та запуску веб-серверу. Відбувається ініціалізація потрібних даних у вигляді змінних та змінної веб-стоірнки.

В вихідному коді використано бібліотеку для димерування “RBDdimmer.h”, що надана виробником димера, який використовується в макеті. Також додано бібліотеку “ESP8266WebServer.h”. Застосовуються об’явлення пінів детекторного (zerocross), керуючого виводів (outputPin).

Перелік додаткових змінних ініціалізованих зовні основних функцій скетчу:

- dimmerLamp dimmer (outputPin, zerocross) — змінна відповідає за димерування (їй передається посилання на вхідний та вихідний піни);
- char* ssid = "name" — збереження назви WiFi мережі;
- char* password = "password" — збереження паролю до WiFi мережі;
- int brightness = 0 – змінна для збереження значення яскравості (навантаження);
- int pos1 = 0 — змінна для розпізнавання значення яскравості;
- int pos2 = 0 — допоміжна зміння значення яскаравості;
- String header — змінна зберігання заголовку запиту;
- char webpage [] — текстова змінна, що містить вихідний код веб-сторінки;
- WiFiServer server(80) — визначення порту розташування веб-серверу, необхідного при використанні протокола передачі TCP/IP.

При поєднанні програми для керування димером та веб-сервером змінилась і функція setup(), як продемонстровано у наступному коді.

Лістинг функції з основними налаштуваннями setup() виглядає наступним чином:

```
void setup() {
Serial.begin(115200);
dimmer.begin(NORMAL_MODE, ON);
dimmer.setPower(8);
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.hostname("brightness_control");
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected.");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
server.begin();
}
```

В даному лістингу відбувається задання швидкості передачі даних. Функції begin(), setPower() об’єкту dimmer забезпечують ввімкнення програмного димеру та встановлення потужність, яка згідно тестів рівна 0 (на лампу проходить потужність при менших значенням функції). В подальшому, вихідна потужність буде регулюватись за допомогою встановлення значення методу setPower() від 8 до 100. Також в монітор COM-port виводиться багато додаткової інформації. В подальшому ця інформація дозволятиме легко відловлювати помилки і допомагатиме при тестуваннях або ж при ремонті приладу, що знаходиться в експлуатації.

Основна циклічна функція loop() містить в собі всю логіку програми. наступний лістинг забезпечує комунікацію між димеруванням та запитами з серверу:

```
void loop(){
 WiFiClient client = server.available();
 if (client) {
  // If a new client connects,
  Serial.println("New Client.");
  String currentLine = "";
  while (client.connected()) {
   if (client.available()) {
    char c = client.read();
    Serial.write(c);
    header += c;
    if (c == '\n') {
     if (currentLine.length() == 0) {
      client.println("HTTP/1.1 200 OK");
      client.println("Content-type:text/html");
      client.println("Connection: close");
      client.println();
```

В даному випадку відбувається початок роботи із клієнтом, який підключається до веб-серверу. В якості клієнта виступає веб-сторінка в браузері, додаток на ПК або додаток на смартфон. Кожного підключеного клієнта можна побачити в моніторі COM-port. Як видно з розробленої структурної схеми алгоритму, для веб-серверу приведені основні функції оброблення запитів з виконанням відповідних керуючих сигналів на мікроконтролер. Це демонструється на наступному лістингу:

```
      if(header.indexOf("GET /find_esp")>=0) {
      client.stop();}
     client.println(webpage);
     if(header.indexOf("GET /LEDon")>=0) {
      for (int i = 10; i < 51; i++)
      {
```

```
          dimmer.setPower(i);
          delay(10);
          }            }
          if(header.indexOf("GET /LEDoff")>=0) {
          dimmer.setPower(10);
          }
         if(header.indexOf("GET /?Brightness=")>=0) {
          pos1 = header.indexOf('=');
          pos2 = header.indexOf('&');
          brightness = header.substring(pos1+1, pos2).toInt();
          dimmer.setPower(map(brightness,0, 100, 30, 100));
          }
         if(header.indexOf("GET /Test_start")>=0) {
          for (int i = 15; i < 36; i++)
          {
          dimmer.setPower(i);
          delay(100);
          }
          for (int i = 35; i>16 ; i--)
          {
            dimmer.setPower(i);
          delay(100);
          }          }
          client.println();
          break;
        } else {
          currentLine = "";
        }
      } else if (c != '\r') {
        currentLine += c;
      }
    }
}
header = "";
client.stop();
```

```
    Serial.println("Client disconnected.");
    Serial.println("");
  }}
```

Як видно, функцію прослуховування запитів виконує конструкція if..else. Кожну ітерацію основної виконавчої функції loop() за допомогою перевірки заголовку запиту відбувається визначення необхідної дії. Таким чином в лістингу приведині наступні можливі відповіді на заголовки:

- "/find_esp" — допоміжний запит для пошуку веб-серверу додатками на ПК та смартфон;
- "/LEDon" — запит на ввімкнення лампи;
- "/LEDoff" — запит на вимкнення лампи;
- "/Test_start" — запит на тестовий запуск;
- "/?Brightness=" — запит, в якому буде передаватись бажане значенння яскравості (потужності).

Запит на зміну яскравості відправляється у форматі "/?Brightness=x&". Програма перебирає рядок і дістає значення "x", переводячи його в чисельний формат з подальшим встановленням за допомгою функції setPower(). Тестовий запит виконує плавне вмикання лампи і проведення яскравості по шкалі від найменшої до найбільшої з подальшим поверненням до вимкненого стану. В кінці лістингу доданий блок коду, який відповідає за відключення клієнту від веб-серверу.

4.2 Розробка веб-сторінки

Розробка веб-сторінки полягає у написанні структури сторінки із використанням HTML, наступний лістинг демонструє чю частину:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8" name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
  <title>Brightness Control</title>
```

```
    </head>
    <body>
      <div class="main">
        <div class="slider_bright">
          <input type="range" min=0 max=100 value=50 class="slider" id="range" >
        </div>
        <p id="demo">Brightness: 50%</p>
        <button id="btn" onclick="turn_on_off(this)" value="1">Turn on led</button>
        <button id="btn_test_start" onclick="test_button_start()">Test</button>
      </div>
    </body>
    </html>
```

В даному випадку тег <head> містить в собі такі дочірні теги, як <meta>, <title>. <title> – назва веб-сторінки, яка буде відображатись в браузері. Тег <meta> зазвичай використовується для зберігання інформації, яка потрібна браузерам або пошуковим системам, наприклад пошукова система звертається до мета тегів, щоб отримати опис сайту або ключових слів. Але для визначеного завдання немає потреби в цьому, тому мета тег використаний лише для забезпечення зручності перегляду веб-сторінки з смартфона. Дані атрибути дозволяють забезпечити масштабування, щоб екран смартфона коректно відображав інтерфейс.

В середині блока <body> створено основний блок інтерфейсу( ), всередині умовно був поділений на наступні елементи:

- – блок для розташування слайдеру;
- <p> – параграф, який буде відповідати за текстову індицацію потужності у відсотках;
- <button> – клавіша перемикання (on/off);
- <button> – клавіша тестового старту;

Розбивка на блоки з подальшими присвоєннями id, class відбувалась для того, щоб було зручно корегувати зовнішній вигляд. Наразі інтерфейс сторінки зображений на рис. 4.1.

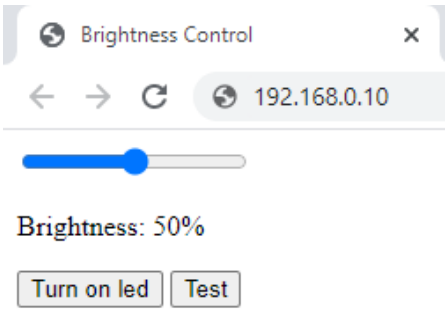


Рисунок 4.1 — Структура сторінки описана на HTML

Спочатку слід вказати стилі для основного тіла сторінки:

```
* {
    margin: 0px;
    padding: 0px;
  }
  body {
  background-color: rgb(27, 41, 48);
  }
```

Вказується, що тіло сторінки не повинно мати відступів, та колір фону. Лістинг для стилів основного блоку інтерфейсу:

```
.main {
    position: absolute;
    top:50%;
    left: 50%;
    transform: translate(-50%, -50%);
    width: 350px;
    height: 350px;
    border-radius: 10px;
    background: rgb(240, 240, 240);
  }
```

Вказано позиціонування відносно вікна браузера, що забезпечує збереження позиції по центру. Для блоку слайдера вказані окремі стилі:

```
.slider_bright {
    margin: 125px 25px 0px 25px;
    width: 300px;
    height: 10px;
    background: grey;
    border-radius: 20px;
  }
  #range {
    display: block;
    visibility: hidden;
    -webkit-appearance: none;
    appearance: none;
    width: 100%;
    height: 10px;
    background-color: rgb(27, 41, 48);
    border-radius: 20px;
    outline: none;
  }
#demo {
    visibility: hidden;
    margin: 20px 105px 0px 105px;
    width: 140px;
    height:  35px;
    text-align: center;
  }
```

Спочатку слайдер та текстова індикація не буде відображатись, адже за замовчування лампа виключена. Тому немає потреби надавати доступ користувачу до слайдеру. Клавіши, які відповідають за перемикання режиму та тестовий запуск, містять наступні налаштування:

```
#btn, #btn_test_start{
    -webkit-appearance: none;
    appearance: none;
    width: 100px;
    height: 40px;
```

```
  cursor: pointer;
  border: 1px solid rgb(27,41,48);
  background: rgb(27, 41, 48);
  color: rgb(240,240,240);
}
#btn {
  margin: 20px 125px 20px 125px;
}
#btn_test_start {
  margin: 20px 125px 20px 125px;
}
#btn:hover, #btn_test_start:hover {
  color: rgb(27, 41, 48);
  background: rgb(240,240,240);
  border: 1px solid rgb(27,41,48);
}
```

На рис. 4.2 зазначений зовнішній вигляд розмітки сторінки із застосованими стилями.

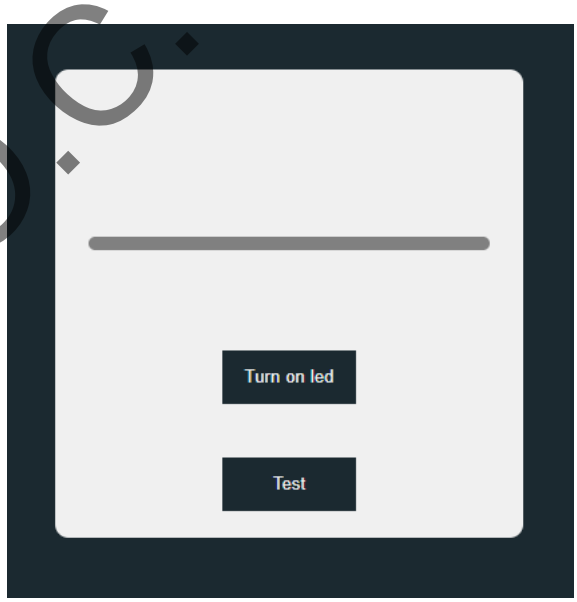


Рисунок 4.2 — Початковий інтерфейс сторінки із підключеними стилями

Для забезпечення функціональності з дотриманням правил моделі AJAX прописана логіка сторінки в окремому тезі <script>. Наступний лістинг демострує ініціалізацію необхідних змінних:

```
var xhr = new XMLHttpRequest();
```

```
var slider = document.getElementById("range");
var output = document.getElementById("demo");
var btn = document.getElementById("btn");
```

Ось демонстрація роботи моделі DOM, тут відбувається звернення і отримання доступу до окремих елементів сторінки, як: слайдеру, параграфу з текстовою індикацією та клавіші перемикання. Також відбулось створення об'єкту XMLHttpRequest, необхідного для надсилання асинхронних запитів до веб-серверу. Метод, що відстежує позицію слайдера oninput() забезпечує відправлення запиту типу "/?Brightness=x&":

```
slider.oninput = function() {
  xhr.open("GET", "/?Brightness=" + slider.value + "&" , true);
  xhr.timeout = 700;
  xhr.ontimeout = function(){
    xhr.abort();
  }
  xhr.send();
  output.innerHTML = "Brightness: " + slider.value + "%";
}
```

До того ж, цей метод забезпечує зміну текстового значення параграфу відповідно до значення, яке відправилось і обробилось на сервері. Також вказані функції захисники від зависань. У випадку якщо сервер не відповість, що приняв значення за вказаний час (можна регулювати в залежності від стабільності WiFi мережі), то запит буде скидуватись і надавати доступ до веб-сторінки. Функція перемикання режиму лампи та тестового ввімкнення, має наступний вигляд:

```
function turn_on_off(elem) {
  if ( elem.value == "1") {
    xhr.open("GET", "/LEDon", true);
      xhr.send();
    elem.value="2";
    elem.textContent = "Turn off led";
    slider.style.visibility = "visible";
```

```
    output.style.visibility = "visible";
  }
  else {
    xhr.open("GET", "/LEDoff", true);
       xhr.send();
    elem.value="1";
    elem.textContent = "Turn on led";
    slider.style.visibility = "hidden";
    output.style.visibility = "hidden";
    slider.value = "50";
    output.innerHTML = "Brightness: 50%";
  }   }
function test_button_start (){
    xhr.open("GET", "/Test_start", true);
       xhr.send();
 }
```

Клавіша має допоміжне значення, по суті це значення використано замість тригера і буквально означає положення “ON”, “OFF”. Демонстрація інтерфейсу при ввімкненому значенні ламп изображена на рис. 4.3.

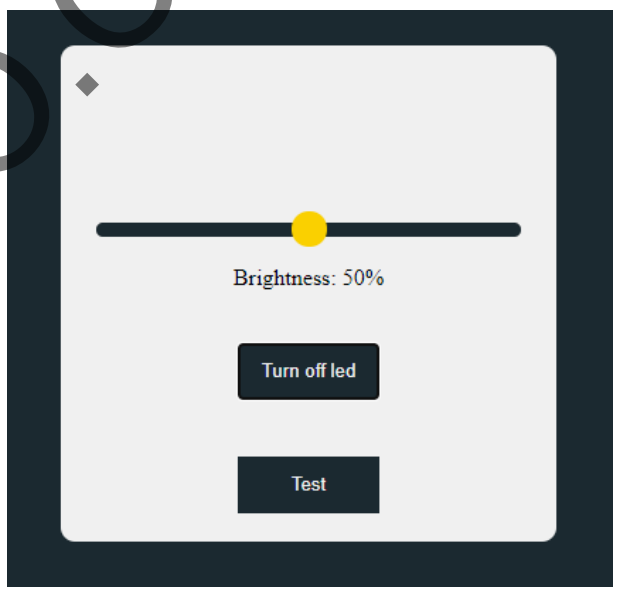


Рисунок 4.3 — Кінцевий інтерфейс сторінки

Таким чином при різних положеннях на сервер відправляються різні заголовки запитів, а після цього відбувається зміна статусу клавіши і надання

користувачу доступу до слайдеру та текстового індикатору при ввімкненому значені і навпаки обмеженням при вимкненому.

Як результат, даний додаток виконаний повністю згідно моделі AJAX, що досяглось за допомогою прописаних наперед стилів різних блоків сторінки, а потім регулювання стилями згідно до необхідного сцерію за допомогою JavaScript. Це дозволяє використовувати додаток без перезавантажень веб-сторінки і при цьому зберігає функціональність.

Окремим пунктом потрібно зазначити, що лістинг логіки сторінки потрібно вставляти тегом <script> вже вкінці перед закриттям тега <body>. Справа в тому, що браузер інтерпритує JavaScript, тобто вихідний код переводиться в машинний рядок за рядком. Відповідно, в разі об’явлення коду в тезі <head> JavaScript не зможе отримати доступ до необхідних елементів (клавіш, слайдеру, тексту).

Для доступу до веб-сторінки необхідно знати IP адресу веб-серверу, тому використання браузерного веб-застосунку можливе, якщо IP адреса веб-серверу не змінюється при певних конфігураціях DHCP або ж при статичних налаштуваннях.

4.3 Розробка додатку на комп’ютер

Оскільки всі додатки розробляються для одного пристрою, інтерфейси мають “фірмовий вигляд”, тобто відбулось майже повне повторення кольорової гами. Проте замінено вигляд клвіші ввімкнення/вимкнення на картинку лампочки, що горить або не горить. Також зміни притерпів слайдер. Для зручності користувача слайдер розташований вертикально. Замість клавіші тесту оформлена клавіша копіювання найденої IP-адреси. Виділено додаткове місце для клавіши підключення. Наступний лістинг відповідає за інтерфейс застосунку:

```
<Grid>
  <Canvas Background="#1B2930">
```

```
<WrapPanel Background="#F1F1F1" VerticalAlignment="Top" Height="25" Width="400">
<WrapPanel Height="25" Width="25">
<Image Source="105964493-light-lamp-sign-icon-bulb-with-gears-symbol.jpg" />
</WrapPanel>
<Label Height="25" Width="325" Content="Wireless Brightness Control"/>
<WrapPanel HorizontalAlignment="Left" Height="25" Width="50" WindowChrome.IsHitTestVisibleInChrome="True">
<Button Name="MinimizeBtn" Style="{StaticResource MinimizedBtn}" Height="25" Width="25" Background="#F1F1F1"/>
<Button Name="CloseBtn" Style="{StaticResource CloseBtn}" Height="25" Width="25" Background="#F1F1F1"/>
</WrapPanel>
</WrapPanel>
<StackPanel Height="400" Width="350" Margin="25,50,25,0">

<Border BorderThickness="0" CornerRadius="15" Background="#FFFFFF" Height="50">
<WrapPanel>
<WrapPanel Height="30" Width="206" Margin="16,10,0,0">
<Label Content="Lamp ip:" HorizontalContentAlignment="Center" Height="20" Width="56" Padding="0" FontSize="15"/>
<Label Content="Enter connect button" Height="30" Width="150" Name="Result" FontSize="15" />
</WrapPanel>
<Label Name="btn" Content="Connect" BorderThickness="1" BorderBrush="#1B2930" Foreground="#1B2930" Width="100" Height="30" Margin="14,10,14,0" FontSize="15" Padding="0" VerticalContentAlignment="Center" HorizontalContentAlignment="Center" PreviewMouseLeftButtonDown="Connect_Button" MouseEnter="btn_mouse_enter" MouseLeave="btn_mouse_leave"/>
```

```
                </WrapPanel>
            </Border>

            <Border BorderBrush="#1B2930" BorderThickness="0"
CornerRadius="15" Height="250" Background="#FFFFFF"
Margin="0,25,0,0">
                <WrapPanel>
                    <Slider Name="slider1" Visibility="Hidden"
Background="#FFFFFF" Style="{StaticResource AppSliderStyle}"
Orientation="Vertical" ValueChanged="Slider_ValueChanged" Value="50"
Minimum="0" Maximum="100" SmallChange="1" Width="20" Height="150"
Margin="100,50,0,50"/>
                    <StackPanel Name="on" Height="150" Width="100"
Margin="65,50,0,50" PreviewMouseLeftButtonDown="Led_on">
                        <Image Name="light" Source="light_off.jpg"
Height="150" Width="100"/>
                    </StackPanel>
                </WrapPanel>
            </Border>

            <Border Name="copy" BorderThickness="1"
CornerRadius="15" Background="#FFFFFF" Height="50" Margin="0,25,0,0">
                <Label Name="lb_copy" Content="Copy wifi lamp ip"
FontSize="15" Foreground="#1B2930" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center" PreviewMouseLeftButtonDown="Copy"
MouseEnter="btn_copy_enter" MouseLeave="btn_copy_leave"/>
            </Border>
        </StackPanel>
    </Canvas>
</Grid>
</Window>
```

З лістингу видно, що структура сторінки на XAML створюється схожим чином. Дочірними тегами <StackPanel> або <WrapPanel> сторінка поділяється на блоки:

- Блок, що відповідає за виведення IP адреси, та клавішу підключення;
- Блок, що відповідає за клавішу перемикання режиму, та слайдер;
- Блок, що відповідає за копіювання IP-адреси.

Різниця між стилями в CSS полягає в тому, що всі стилі в даному блоці прописані у вигляді атрибутів. Також позиціонування блоків та панелей відбулося за допомогою абсолютних величин, вказаних в пікселях. Функцію виведення тексту бере на себе тег <label>.

На рис. 4.4 зображений інтерфейс додатку для ПК.

Рисунок 4.4 — Стартовий інтерфейс додатку для ПК

Також цікавим рішенням було визначити окремий блок під картинку і за допомогою функцій обробників в лістингу вихідного коду на C# додано можливість змінювати стан картинки при виконанні натиску. Оскільки додаток не потребує багато місця на екрані було прийнято рішення обмежити можливість розтягування на весь екран.

Як і на веб-сторінці, доступ до слайдеру надаватиметься користувачеві після ввімкнення лампи.

Логіка веб-застосунку прописана в окремому файлі, який прикріплений до проекту при розробці. В минулому розділі на структурній схемі алгоритму

розглядалась різниця між додатком веб-сторінки та застосунком на ПК. Вона полягає в розробці алгоритму пошуку IP-адреси веб-серверу. Структурна схема алгоритму пошуку веб-сторінки зображена на рис. 4.5.

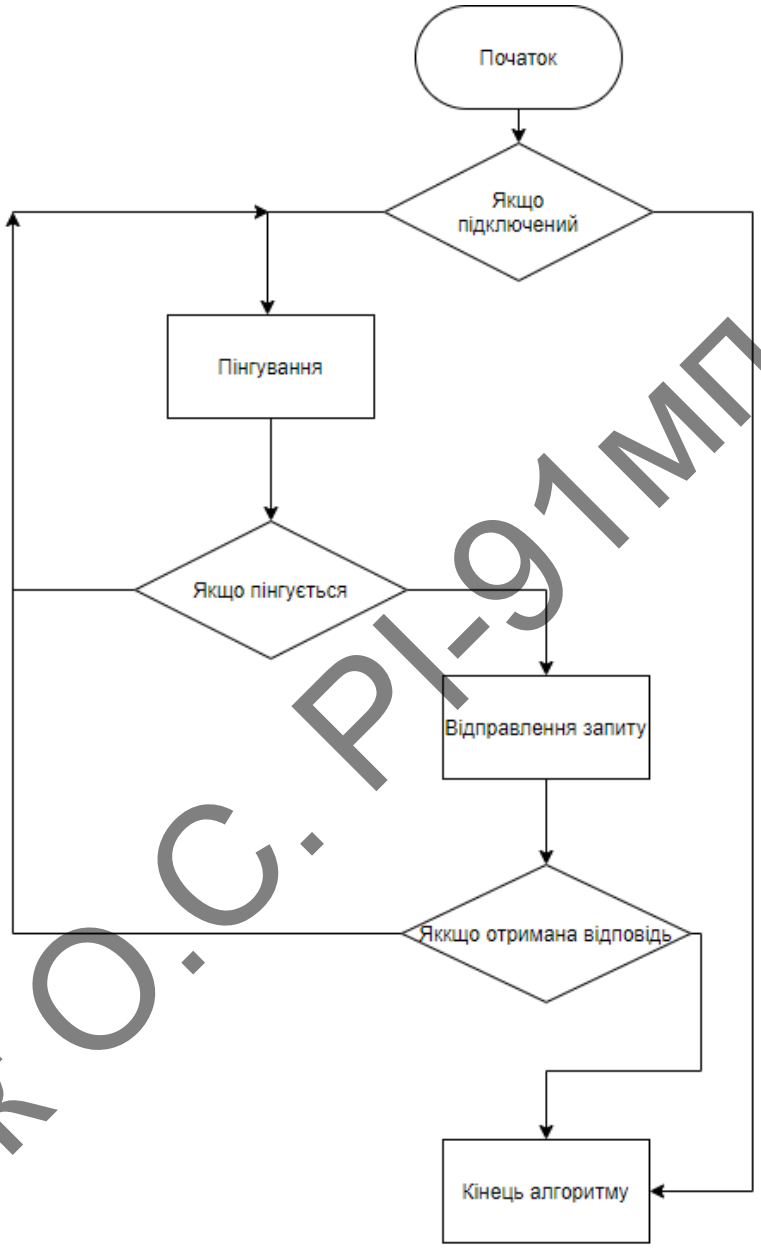


Рисунок 4.5 — Схема алгоритму пеленгування

Алгоритм забезпечує пінгування всіх IP-адрес локальної мережі і забезпечує надсилання запиту лише після успішного пінгування. Цей метод дозволяє зекономити ресурси і не навантажувати застосунком комп'ютер відправкою запитів. Ping — це службова утиліта необхідна для перевірки мережі.

Для перевірки застосовується надсилання пакету до зазначеного вузлу мережі та фіксується час отримання пакету у відповідь. Такий метод дозволяє визначити затримки, завантаження мережі або підтверджує сам факт існування пристрою за даною адресою. Пінгування проходить швидше і інформація про успішне пінгування займає менше пам'яті, що додає швидкості. Якщо в мережі відсутні користувачі, то алгоритм припинить свою дію і видасть статус помилки.

Для забезпечення коректної роботи застосунку спроектовані наступні функції:

- Функція старту пошуку;
- Допоміжна функція пінгування;
- Допоміжна функція, що надсилає та опрацює відповідь на запит про пошук;
- Функція клавіші ввімкнення та вимкнення;
- Функція клавіші зміни положення повзунка на слайдері;
- Функція копіювання IP адреси в буфер обміну.

Наступний лістинг реалізує функцію старту пошуку:

```
private async void Connect_Button(object sender, RoutedEventArgs e)
{
    if (!connected)
    {
        await btn.Dispatcher.BeginInvoke(new updateDelegate(updateButton_start), "Connecting");
        var host = Dns.GetHostEntry(Dns.GetHostName());
        foreach (var ip in host.AddressList)
        {
            if (ip.AddressFamily == AddressFamily.InterNetwork)
            {
                if (ip.ToString().Contains("192.168."))
                {
                    localip = true;
```

```
                    for (int i = 0; i <= 255; i++)
                    {
                        Ping p = new Ping();
                        var task = PingAndUpdateAsync(p,
ip.ToString().Substring(0, ip.ToString().LastIndexOf(".")) + "." + i.ToString());
                    }
                    await btn.Dispatcher.BeginInvoke(new
updateDelegate(updateButton_start), "Connect");
                }
            }
        }
        if (localip == false)
        {
            await btn.Dispatcher.BeginInvoke(new
updateDelegate(updateButton_start), "Connect");
            MessageBox.Show("Please, connect to wifi!");
        }    }    }
```

Функція містить в собі перевірки для відловлювання помилок, наприклад відразу йде перевірка, чи користувач не є вже підключеним задля обмеження наватаження програми на комп'ютер без необхідності. Також гостро стояла проблема виконання алгоритму “асинхронно”. Справа в тому, що даний алгоритм в теорії може завантажити застосунок на, відносно, велику кількість часу, що викличе ефект зависання, а далі і отримання помилки з вимогою завершити виконання програми. Для забезпечення асинхронності використано ключове слово await або async, яке запускає функцію прослуховувач відповіді або кінця виконання задачі. Асинхронність релізовується методом передачі завдання в інший потік, а основний залишається за користувачем. Відповідальність за правильність розподілення днаих у поток лежить на проектувальнику, тому необхідно правильно використовувати async, await. В окремих випадках в документації можна знайти функції, які працюють асинхронно за замовчуванням. Оскільки пінгування та запити можна виконувати паралельно, ще більше зменшується час виконання програми. Алгоритм про-

сто надсилає пінгування і в цей час може надсилатись запит із подальшим очікуванням на даресу, що була успішно пропінгована. Дана функція застосовує механізм для 254 можливих IP-адрес в даній підмережі та викликає функцію PingAndUpdateAsync() для кожної з них. Також кожна дія супроводжується зміною статусів клавіши, для того щоб користувач розумів чи відбувається з'єднання, чи воно вже завершилось, чи взагалі ПК не підключений до мережі.

Наступний лістинг демонструє реалізацію функції PingAndUpdateAsync():

```
private async Task PingAndUpdateAsync(Ping ping, string ip)
    {
        var reply = await ping.SendPingAsync(ip, timeout_p);

        if (reply.Status == IPStatus.Success)
        {
            await Task.Run(() => Request(reply));
        }
    }
```

Це допоміжна функція і вона забезпечує асинхронне пінгування вказаної в аргументі "ip" IP-адреси. У випадку невдачі — пінг скидується, успіх — викликатиме функцію, яка відповідає за синхронний HTTP запит із заголовком "/find_esp". Функція відправки запитів:

```
private void Request(PingReply reply)
        {
            try
            {
                var mYrequest = (HttpWebRequest)WebRequest.Create("http://" + reply.Address.ToString() + "/find_esp");
                mYrequest.Timeout = timeout_r;
                HttpWebResponse mYresponse = (HttpWebResponse)mYrequest.GetResponse();
```

```
                if ((mYresponse.StatusCode == HttpStatusCode.OK))
                {
                    esp_ip = reply.Address.ToString();
                    connected = true;
                    btn.Dispatcher.BeginInvoke(new
updateDelegate(updateButton_end), "Connected");
                }
                mYresponse.Close();
            }
            catch { }
        }
```

До функції передається успішно пропінгована IP-адреса, а потім виконується надсилання запиту, у випадку успіху — дія алгоритму зупиняється і статус підключення змінюється, у разі невдачі — помилка відловлюється конструкцією try..catch і відбувається запобігання зависання застосунку. В результаті виконання асинхронних функцій винає проблема, яка пов’язака із неможливістю надання доступу до блоків інтерфейсу із не основного потоку. Для цього в функціях було використано Dispathcer, це надало можливості передати звернення до блоків інтерфейсу в основний потік. Таким чином, в асинхронних функціях також забезпечується можливість зміни статусу клавіші підключення.

Функція перемикання станку лампи:

```
        private void Led_on(object sender, RoutedEventArgs e)
        {
            if (esp_ip.Contains("192.168."))
            {
                if (value == 1)
                {
                    try
                    {
                        var mYrequest =
(HttpWebRequest)WebRequest.Create("http://" + esp_ip + "/LEDon");
                        mYrequest.Timeout = timeout_r;
```

```
                    HttpWebResponse mYresponse =
(HttpWebResponse)mYrequest.GetResponse();
                    mYresponse.Close();
                    BitmapImage image = new BitmapImage(new
Uri("light_on.jpg", UriKind.Relative));
                    light.Source = image;
                    slider1.Value = 50;
                    slider1.Visibility = Visibility.Visible;
                    value = 2;
                }
                catch
                {           }
            }
            else
            {
                try
                {
                    var mYrequest =
(HttpWebRequest)WebRequest.Create("http://" + esp_ip + "/LEDoff");
                    mYrequest.Timeout = timeout_r;
                    HttpWebResponse mYresponse =
(HttpWebResponse)mYrequest.GetResponse();
                    mYresponse.Close();
                    BitmapImage image = new BitmapImage(new
Uri("light_off.jpg", UriKind.Relative));
                    light.Source = image;
                    slider1.Visibility = Visibility.Hidden;
                    value = 1;
                }
                catch { }
            }
        }
        else
        {
```

```
                MessageBox.Show("Please, connect to wifi and your smart
lamp!");
            }
        }
```

Знову особливості проектування веб-застосунків на ПК потребували використання великої кількості перевірок та конструкцій відловлювань помилок. Зміна статусу клавіші відбувається за допомогою зміни картинки. Картинка вимкненої лампи замінюється картинкою ввімкненої, при цьому відправляється відповідний запит на сервер, а потім надання доступу до слайдеру. Цього разу немає потреби в застосуванні асинхронних функцій, адже навантаження застосунку доволі незначне.

Функція для роботи слайдеру:

```
    private void Slider_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            if (esp_ip.Contains("192.168."))
            {
                try
                {
                    ((Slider)sender).SelectionEnd = e.NewValue;
                    var mYrequest =
(HttpWebRequest)WebRequest.Create("http://" + esp_ip + "/?Brightness=" +
e.NewValue + "&");
                    mYrequest.Timeout = timeout_r;
                    HttpWebResponse mYresponse =
(HttpWebResponse)mYrequest.GetResponse();
                    mYresponse.Close();
                }
                catch { }
            }
        }
```

Принцип полягає у відловлювані положення повзунка і відправці відповідного значення за вказаною IP-адресою.

Функція для копіювання в буфер обміну:

```
private void Copy(object sender, RoutedEventArgs e)
        {
            if (esp_ip.Contains("192.168."))
            {
                Clipboard.Clear();
                Clipboard.SetText(Result.Content.ToString());
            }
            else
            {
                MessageBox.Show("Please, connect to wifi and your smart lamp!");
            }
        }
```

Після виконання можливе подальше збереження адреси для повторного підключення, що надає доступ до веб-сторінки через браузер або ж просто економить час. На рис. 4.6 зображено інтерфейс програми для ПК при підключенні.

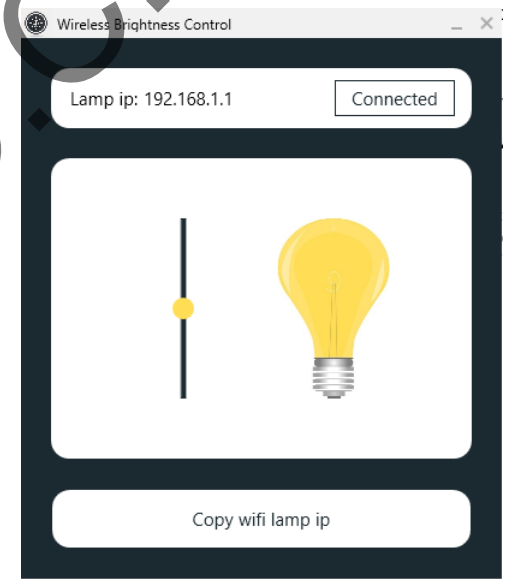


Рисунок 4.6 — Кінцевий інтерфейс додатку для ПК

Додатково в лістингу логіки були реалізовані допоміжні функції для обрання стилів (графічного зображення) слайдеру та клавіш. Нажаль наявний

функціонал не дав можливості реалізувати це за допомогою атрибутів в мові XAML.

4.4 Розробка додатку для смартфона

Інтерфейс додатку для смартфону розроблено за допомогою схожої технології, що забезпечує відносну простоту розробки. Наступний лістинг демонструє вихідний код структури застосунку:

```
<Grid BackgroundColor="#1B2930">
    <StackLayout Margin="0,25,0,25" HorizontalOptions="FillAndExpand" VerticalOptions="Center">
        <Frame CornerRadius="15" BackgroundColor="#FFFFFF">
            <StackLayout Orientation="Horizontal" HorizontalOptions="FillAndExpand">
                <StackLayout Orientation="Horizontal" HorizontalOptions="FillAndExpand">
                <Label Text="Lamp ip:" VerticalTextAlignment="Center" FontSize="14" TextColor="#1B2930"/>
                <Label Text="Enter connect button" VerticalTextAlignment="Center" FontSize="14" TextColor="#1B2930" x:Name="Result" HorizontalOptions="Start"/>
                </StackLayout>
                <Button Text="Connect" x:Name="btn_c" FontSize="12" WidthRequest="100" BackgroundColor="#FFDD55" TextColor="#1B2930" Clicked="Connect_Button" HorizontalOptions="End" />
            </StackLayout>
        </Frame>
        <Frame CornerRadius="15" BackgroundColor="#FFFFFF" Margin="0,25,0,0" HeightRequest="250">
            <AbsoluteLayout HorizontalOptions="FillAndExpand">
                <Slider x:Name="slider1" IsEnabled="False" MinimumTrackColor="#1B2930" MaximumTrackColor="#1B2930" ValueChanged="Slider_ValueChanged" Minimum="0" Value="50" Maximum="100" ThumbColor="#FFDD55" Rotation="-90"
```

```
AbsoluteLayout.LayoutBounds="-0.2,0.5,200,20"
AbsoluteLayout.LayoutFlags="PositionProportional"/>
                    <ImageButton x:Name="on" Clicked="Led_on"
BackgroundColor="#0000" Source="{local:ImageResource
lightapp.light_off.jpg}" AbsoluteLayout.LayoutBounds="0.5,0.5,115,275"
AbsoluteLayout.LayoutFlags="PositionProportional"/>
                  </AbsoluteLayout>
                </Frame>
                <Frame CornerRadius="15" BackgroundColor="#FFFFFF"
HeightRequest="50" Margin="0,25,0,25">
                  <StackLayout HorizontalOptions="FillAndExpand">
                    <Button Text="Copy wifi lamp ip" Clicked="Copy"
BackgroundColor="#FFDD55" TextColor="#1B2930"/>
                  </StackLayout>
                </Frame>
              </StackLayout>
            </Grid>
          </ContentPage>
```

В даному випадку для поділу на симантично різні блоки використано тег <frame>. Сильно змінилось вказування координат блоків. Додаток для ПК в силу можливостей моніторів не мав обмеженням по розміру і коретно відображається при абсолютному позиціонуванні блоків на більшості моніторів. Додаток на смартфон має довгий, але вузький екран, відповідно до такої форми блоки мають використовувати масштабування по горизонталі. Суттєва різниця ще полягає в наявності спеціального тегу для клавіш у вигляді картинок, а саме <imageButton>. Відпала потреба прописувати окремі допіміжні функції у файлі логіки додатку, цілком вистачає стандартних атрибутів.

Згідно із структурними схемами алгоритмів, вихідний код додатку на смартфон майже не змінився. Зміни відбулися в методі пошуку IP-адреси. Практичні тести показали, що навіть при використанні асинхронних методів роботи, час пошуку IP-адреси залишився доволі значним. Скоріше за все це пов'язано із урізаними можливостями паралелізації потоків на смартфонах.

Тому наступний лістинг демоснтрує зміни у принципі роботи функції пошуку відносно версії для ПК:

```
private async void  Connect_Button(object sender, EventArgs e)
    {
        if (!connected)
        {
Device.BeginInvokeOnMainThread(updateButton_connecting);
            var host = Dns.GetHostEntry(Dns.GetHostName());
            foreach (var ip in host.AddressList)
            {
                if (ip.AddressFamily == AddressFamily.InterNetwork)
                {
                    if (ip.ToString().Contains("192.168."))
                    {
                        localip = true;
                        my_ip = ip.ToString();
                     esp_ip1 = my_ip.Substring(0, my_ip.LastIndexOf(".")) + ".";
                    }
                }
            }
            if (localip == false)
            {
Device.BeginInvokeOnMainThread(updateButton_connect);
                await DisplayAlert("Error", "Please, connect to wifi!", "OK!");
            }
                if                                                    (await
DependencyService.Get<IFileWorker>().FileNotEmptyAsync(filename)     &&
(localip == true))
                {
                    try
                    {
                        string          check          =          await
DependencyService.Get<IFileWorker>().LoadTextAsync(filename);
```

```
                    var myrequest =
(HttpWebRequest)WebRequest.Create("http://" + check + "/find_esp");
                    myrequest.Timeout = 700;
                    HttpWebResponse myresponse =
(HttpWebResponse)myrequest.GetResponse();
                    if ((myresponse.StatusCode == HttpStatusCode.OK))
                    {
                        esp_ip = check;
                        connected = true;
                Device.BeginInvokeOnMainThread(updateButton_end);
                    }
                    myresponse.Close();
                }
                catch
                {
                    await
DependencyService.Get<IFileWorker>().SaveTextAsync(filename, "");
                }
            }
            if (!(await
DependencyService.Get<IFileWorker>().FileNotEmptyAsync(filename)) &&
(localip == true))
            {
                for (int i = StartIP; i <= StopIP; i++)
                {
                    Ping p = new Ping();
                    Task task = PingAndUpdateAsync(p, esp_ip1 +
i.ToString());
                }  }  }  }
```

Основа алгоритму майже не змінилась, принцип залишився тим же і полягає в пінгуванні пристроїв та подальшому відправленні запитів. Проте було додано додаткову перевірку на початку дії алгоритму. Ця перевірка забезпечує пошук “можливої” IP-адреси в збереженому застосунком файлі. Якщо файл містить не правильну адресу або взагалі пустий, алгоритм діє звичним

методом. В кінці успішного нового пошуку до файлу записується нова IP-адреса і в подальшому в пріорітетному порядку буде перевірятись саме вона.

Такий підхід суттєво економить час, коли веб-сервер не вимикався від WiFi мережі. Для даного алгоритму реалізовані наступні функції:

- Перевірка файлу на пустоту;
- Надання доступу до контенту програмі;
- Оновлення текстового значення файлу.

Реалізація перевірки файлу на пустоту:

```
public Task<bool> FileNotEmptyAsync(string filename)  {
bool filetext = false;
string filepath = GetFilePath(filename);
bool exists = File.Exists(filepath);
if (exists)
{                    using    (StreamReader    reader    =
File.OpenText(filepath))
{
if (reader.ReadToEnd() != "")
{
filetext = true;
} } }
Else
{
filetext = false;
}
return Task<bool>.FromResult(filetext);
}
```

Функція перевіряє спочатку наявність файла за вказаним стандартним для застосунків шляхом, за допомгою аналізу текстового потоку файлу надсилає текст назад в програму(в даному випадку IP-адресу) або ж видає негативний результат, що свідчить про пустоту файлу.

Допоміжна функція для доступу до контенту:

```
public async Task<string> LoadTextAsync(string filename) {
```

```
string filepath = GetFilePath(filename);
using (StreamReader reader = File.OpenText(filepath))
{
return await reader.ReadToEndAsync();
} }
```

Задача функції полягає у визначенні стандартного шляху файлу і виведенні текстового вмісту до потоку.

Оновлення текстового значення файлу:

```
public async Task SaveTextAsync(string filename, string text)
{
string filepath = GetFilePath(filename);
using (StreamWriter writer = File.CreateText(filepath))
{
await writer.WriteAsync(text);
} }
```

Функція створює writer, що за допомогою "письмовго" потоку забезпечує вставку або заміну старого тексту на новий. Вище описані функції використовують асинхронні методи роботи для запобігання зависань при довгих зчитуваннях сторонніх файлів.

Інтерфейс програми у підключеному стані зображений на рис. 4.7.

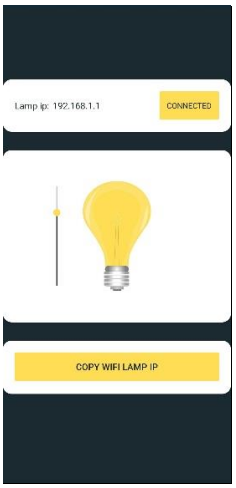


Рисунок 4.7 — Інтерфейс веб-додатку для смартфону

Додатки забезпечують необхідну гнучкість, швидкодію та надійність в рамках пристрою "розумної лампи", за допомогою розроблених алгоритмів.

5 РОЗРОБКА СТАРТАП-ПРОЕКТУ

Даний розділ має на меті проведення маркетингового аналізу стартап-проекту задля визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження.

5.1 Опис ідеї проекту (товару, послуги, технології)

В межах цього підпункту аналізується зміст ідеї, можливі напрямки застосування, основі вигоди які може отримати користувач товару та відмінності від існуючих аналогів та замінників табл. 5.1.

Таблиця 5.1 — Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка засобів дистанційного керування силовим навантаженням(освітленням)	1. Підвищення ефективності застосування апаратури	Виявлення нових механізмів програмної співпраці із периферією
	2. Спрощення процесу використання наявних засобів керування силовим навантаженням	Збільшення ефективності, швидкодії при роботі із пристроями керувань силовим навантаженням
	3. Оптимізація програмних рішень, забезпечення належної швидкодії продукта	Новий підхід в алгоритмах програмних продуктів для керування навантаженням

Конкурентими є аналогічні реалізації та механізми керування силовим навантаженням(розумні лампи). Основна відмінність полягає в використанні WiFi мережі для комунікації в поєднанні із використанням моделі AJAX в програмному рішенні, а також використанні власного швидкого алгоритму пошуку пристрою в локальній мережі.

Конкурентами є аналогічні методи та механізми застосування високочастотної електрохірургії. Основною відмінністю є те, що високочастотний електрохірургічний вплив на біотканину виведено на новий перспективний рівень зі зменшенням негативного впливу на людину табл. 5.2.

Довгостроковими перспективами є:

- збільшення кількості клієнтів, що будуть використовувати запропоно-

вані методи;

- додавання новітніх механізмів оптимізованого алгоритму взаємодії периферії та веб-застосунків.

Таблиця 5.2 — Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент 1	Конкурент 2	Конкурент 3			
1.	Дешевизна розробки	+	-	-	-	-	-	+
2.	Використання сучасних технологій розробки	+	-	+	-	-	+	+
3.	Швидкодія алгоритму	+	-	-	-	-	-	+
4.	Високий рівень розробки	+	+	-	+	-	+	-

5.2 Технологічний аудит ідеї проекту

Для реалізації даного проекту необхідно обрати набір технологій використаних при розробці:

1. C/C++ – мови програмування, що дають змогу працювати в межах скетчів на мікроконтролер за допомогою Arduino IDE.

2. HTML/CSS – мови для формування графічного інтерфейсу веб-сторінки для керування пристроєм розумної лампи.

3. JavaScript – мова програмування, що забезпечує логіку веб-сторінки, а також дотримання застосунку моделі розробки AJAX, опираючись на співпраці між різними елементами структури сторінки.

4. XAML – мова розмітки, яка застосовується для ПК та смартфону в межах одного IDE Visual Studio. Надає схожі можливості, як і мови HTML/CSS.

5. C# – мова програмування, нагадує синтаксисом мови C/C++, Java. В даному випадку забезпечує розробку логіки веб-застосунків для девайсів на системах Windows, Android, IOS.

Таблиця 5.3 — Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Розробка пристрою дистанційного керування навантаженням(розумна лампа)	C/C++	так	так
2		HTML/CSS	так	так
3		JavaScript	так	так
4		XAML	так	так
5		C#	так	так
Обрана технологія реалізації ідеї проекту: обрані всі технології, адже завдання полягає в розробці різних методів керування				

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Таблиця 5.4 — Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	2
2	Загальний обсяг продаж, ум.од	Невідомий
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	немає
5	Специфічні вимоги до стандартизації та сертифікації	Існують
6	Середня норма рентабельності в галузі (або по ринку), %	Невідома

На основі проведеного дослідження є можливість стверджувати про привабливість проекту для входження на ринок за попереднім оцінюванням.

Визначимо потенційні групи клієнтів.

Таблиця 5.5 — Характеристика потенційних клієнтів стартап-проекту

№ п/п	**Потреба, що формує ринок**	**Цільова аудиторія (цільові сегменти ринку)**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**
1	Вирішення основних недоліків інсуючих аналогів(розумних ламп)	Підприємста, побутові користувачі	Невідомі	Обов'язкова наявність сертифікатів, відповідність надами характеристикам та швидкодії

Проведемо аналіз ринкового середовища: складемо таблиці факторів, що перешкоджають ринковому впровадженню проекту, та факторів, що йому сприяють.

Таблиця 5.6 — Фактори загроз

№ п/п	**Фактор**	**Зміст загрози**	**Можлива реакція компанії**
1	Агресивність конкурентів	Розробка кращого алгоритму взаємодії	Додаткові витрати на розробку нового алгоритму
2	Новий функціонал	Впровадження можливості керування кольорами, поєднання декількох ламп	Падіння продаж

Таблиця 5.7 — Фактори можливостей

№ п/п	**Фактор**	**Зміст можливості**	**Можлива реакція компанії**
1	Вдосконалення існуючих технологій	Періодичне вдосконалення алгоритму	Може підняти продажі відносно конкурентів
2	Додавання нового функціоналу	Додавання різних можливостей в додаток за допомогою підключення нових технологій	Покращення показників продаж

3	Розширення сфери роботи пристрою	Проводити роботу дистанційним навантаженням на прикладі керувань тенами та моторами	Вихід компанії на нові ринки збуту

Проведемо аналіз пропозиції: визначимо загальні риси конкуренції на ринку.

Таблиця 5.8 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	**В чому проявляється дана характеристика**	**Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)**
1. Тип конкуренці — олігополія	Домінує невелика кількість конкруючих між собою фірм	В разі виявлення не достатньої активності на ринку зростають шанси втрати ринку збуту
2. За рівнем конкурентної боротьби національна	Товар виробляється, а також збувається в багатьох країнах світу	
3. За галузевою ознакою — міжгалузева	При певних умовах вдосконалення продукту, а також забезпечення належного функціоналу є можливість використати проект в інших галузях	Штовхає до постійного допрацювання продукту задля подальшого виходу на нові ринки
5. За характером конкурентних переваг — нецінова	При виборі даного товару основна увага повинанна звертатись на показники швидкодії та функціонал, а не на ціну	Можливе зменшення прибутку, в разі виставлення дорогої ціни, якщо не досягнути ефективності врази крощої за конкурентів

Таблиця 5.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	**Прямі конкуренти в галузі**	**Потенційні конкуренти**	**Постачальники**	**Клієнти**	**Товари-замінники**
	Навести перелік прямих конкурентів	**Визначити бар'єри входження в ринок**	**Визначити фактори сили постачальників**	**Визначити фактори сили споживачів**	**Фактори загроз з боку замінників**

Висновки:	Основними конкурентами є компанії, які вже займаються збутом розумних ламп на ринк	Бар'єри входу відносно невеликі, за допомогою сучасних маркетплейсів легко можна завоювати популярність виробу при дотриманні належного рівня якості та характеристик	Невідомо	Невідомо	Невідомо

Згідно з результатими аналізу можна зробити висновок, що працювати на ринку можна. Для достатнього поширення на ринку потрібно забезпечувати необхідний набір характеристик брати активну участь у вдосконаленні свого проекту.

Перелічимо фактори конкурентоспроможності.

Таблиця 5.10 — Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування вибору
1	Частка ринку	Дана область не досягла своїх піків в розробці, відповідно є місце для актуалізації та вдосконалення
2	Універсальність	При належних потребах та вдосконаленні, можна використовувати даний пристрій в інших галузях
3	Швидкодія	За використання сучасного алгоритму, та необхідного функціоналу досягається конкурентспроможність при однакових цінах на ринках

Проведемо аналіз сильних та слабких сторін стартап-проекту табл. 4.11.

Таблиця 5.11 — Порівняльний аналіз сильних та слабких сторін стартап-проекту

№	Фактор конкурентоспроможності	Вагові значення фактора (1–20)	Рейтинг конкурентів у порівнянні з проектом, що розробляється						
			-3	-2	-1	0	1	2	3
1	Частка ринку	20						1	
2	Універсальність	10							1

3	Швидкодія	18							1

Складемо матрицю SWOT-аналізу табл. 5.12.

Таблиця 5.12 — SWOT-аналіз стартап-проекту

Сильні сторони	**Слабкі сторони**
1. універсальний застосування; **2.** швидкодія алгоритму; **3.** актуальні технології;	**1.** Не відома компанія **2.** Всі розробки проводяться за рахунок розробника;
Можливості	**Загрози**
1. Розширення функціоналу **2.** Вдосконалення алгоритму **3.** Агресивна цінова політика	**1.** Загроза мати малі показники рентабельності; **2.** Вдосконалення своїх засобів конкурентами

З результатів SWOT-аналізу видно, що найбільш негативний вплив на створення даного стартап-проекту на ринку чинить вдосконалення технічних засобів конкурентів. В такому випадку за рахунок розробника буде провидитись вдосконалення, а відповідно немає потреби розробляти альтернативи впровадження.

5.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів табл. 5.13.

Таблиця 5.13 — Вибір цільових груп потенційних споживачів

№ п/п	**Опис профілю цільової групи потенційних клієнтів**	**Готовність споживачів сприйняти продукт**	**Орієнтовний попит в межах цільової групи (сегменту)**	**Інтенсивність конкуренції в сегменті**	**Простота входу у сегмент**
1	Підприємства	готовий	високий	мінімальна	простий
2	Звичайний користувач	готовий	високий	максимальна	простий
Які цільові групи обрано: підприємства, звичайни користувач побутовими товарами					

Для роботи в обраних сегментах ринку сформулюємо базову стратегію розвитку табл. 5.14.

Таблиця 5.14 — Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	**Стратегія диференціаційованого маркетингу**	Ефективність, універсальність, швидкодія	Стратегія спеціалізації

Таблиця 5.15 — Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Компанія буде забирати і нових користувачів так і відбивати існуючих	Частково	Наслідування лідеру

Розробимо стратегію позиціонування табл. 5.16, що полягає у формуванні ринкової позиції (комплекту асоціацій), за яким споживачі мають ідентифікувати торгівельну марку / проект.

Таблиця 5.16 — Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Відповідність харакатеристикам	Розвиток використаних алгоритмів та вдосконалення функціональнсоті	Показник швидкодії досягунтий за допомогою нового алгоритму та можливість використання універсально при дороблленні функціональності	Доступна ціна Реалізація нових методів Швидкодія

5.5 Розроблення маркетингової програми

Першим кроком є формування маркетингової концепції товару, яку отримає споживач. Для цього у табл. 5.17 підсумовуємо результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.17 — Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1.	Використання дистанційних засобів керування силовим навантаженням(розумна лампа)	1. Підвищення ефективності застосування апаратури	Виявлення нових механізмів програмної співпраці із периферією
2.		2. Спрощення процесу використання наявних засобів керування силовим навантаженням	Збільшення ефективності, швидкодії при роботі із пристроями керувань силовим навантаженням
3.		3. Оптимізація програмних рішень, забезпечення належної швидкодії продукта	Новий підхід в алгоритмах програмних продуктів для керування навантаженням

Розробимо трирівневу маркетингову модель товару з метою уточнення ідеї продукту, його фізичних складових, особливостей його надання табл. 5.18.

Таблиця 5.18 — Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
І. Товар за задумом	Дистанційне керування силовим навантаженням
ІІ. Товар у реальному виконанні	Ефективність, швидкодія, функціональність
	Якість: висока
	Марка: Підвищення ефективності застосування апаратури
ІІІ. Товар із підкріпленням	До продажу — невідомо
	Після продажу — невідомо

Варто відзначити, що товар буде захищено від копіювання за рахунок захисту ідеї товару (захист інтелектуальної власності) та маркою бренду.

Наступним кроком визначимо цінові межі, якими необхідно керуватись при встановленні ціни на потенційних товар.

Таблиця 5.19 — Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	1000грн.	800грн.	середній	600–900 грн.

Визначимо оптимальну систему збуту табл. 5.20.

Таблиця 5.20 — Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Мінімальна кількість посередників	Магазини,інтернет-маркетпелйси	невідома	Магазини, інтернет-маркетплейси

Розробимо концепцію маркетингових комунікацій.

Таблиця 5.21 — Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Невідома	Магазини, інтернет-маркетплейси	Можливості проекту	Донести про переваги до потенційних замовників	Акцентування на характеристиках та постійному вдосконаленні

За результатами проведеного аналізу можна зробити висновок, що пристрій дистанційного керування силовим навантаженням на прикладі розумної лампи має доволі суттєві шанси на комерціалізацію своєї ніші на ринку. Також є потенціал розвитку за допомогою збільшення функціональності пристрою та можливого вдосконалення для експансії нових ринків збуту.

ВИСНОВКИ

1. В рамках даної магістерської дисертації обрано WiFi мережу в якості засобу передачі даних, адже забезпечується необхідна швидкодія пристрою, надійність, можливість підключення декількох користувачі з будь-яких девайсів, наявність інтерфейсу комутації між пристроями в межах локальної мережі.
2. Симбіоз моделей DOM, AJAX, OSI в рамках однієї системи забезпечують високі показники інтерактивності додатку, гнучкості, швидкості підключення та передачі керуючих сигналів, а також обмеження користувача від не бажаних дій, що можуть призвести до помилки.
3. Проведено допрацювання функціоналу додатків, розроблено новий механізм швидкого пелентування вузла в локальній мережі за допомогою утиліти PING на ПК та проведено модифікацію із системою пріорітетного пошуку для додатку на смартфон, що забезпечує належну швидкодію та дозволяє зекономити до 30-40с часу при під’єднанні до серверу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Лампа Yeelight [Електронний ресурс]. — Режим доступу: https://rozetka.com.ua/ua/58659016/p58659016/ — Назва з екрану.

2. Лампа WIZ [Електронний ресурс]. — Режим доступу: https://rozetka.com.ua/ua/wiz_wz0126072/p55255656/ — Назва з екрану.

3. Лампа MiPow [Електронний ресурс]. — Режим доступу: https://bt.rozetka.com.ua/ua/54200568/p54200568/ — Назва з екрану.

4. Arduino Controlled Light Dimmer [Електронний ресурс]. — Режим доступу:https://www.instructables.com/id/Arduino-controlled-light-dimmer-The-circuit/ — Назва з екрану.

5. Твердотільне реле [Електронний ресурс]. — Режим доступу: https://go-radio.ru/tverdotelnoe-rele.html — Назва з екрану.

6. ESP8266 [Електронний ресурс]. — Режим доступу: https://ardushop.in.ua/arduino/wi-fi-module-esp8266-version-esp-12e — Назва з екрану.

7. Бібліотека "CyberLib" [Електронний ресурс]. — Режим доступу: https://github.com/pythonista/CyberLib — Назва з екрану.

8. Бібліотека "AC_Dimmer" [Електронний ресурс]. — Режим доступу: https://github.com/AlexGyver/AC_Dimmer — Назва з екрану.

9. Бібліотека "ESPWebServer" [Електронний ресурс]. — Режим доступу:https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WebServer — Назва з екрану.

10. Бібліотека "RBDDimmer" [Електронний ресурс]. — Режим доступу: https://github.com/RobotDynOfficial/RBDDimmer — Назва з екрану.

11. HTML [Електронний ресурс]. — Режим доступу: https://www.w3.org/TR/html52/ — Назва з екрану.

12. CSS [Електронний ресурс]. — Режим доступу:https://www.w3.org/Style/CSS/ — Назва з екрану.

13. JavaScript Електронний ресурс]. — Режим доступу:https://developer.mozilla.org/uk/docs/Web/JavaScript — Назва з екрану.

14. AJAX Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/ru/docs/Web/Guide/AJAX — Назва з екрану.

15. HTTP Електронний ресурс]. — Режим доступу:https://developer.mozilla.org/ru/docs/Web/HTTP/Overview — Назва з екрану.

16. Модель OSI Електронний ресурс]. — Режим доступу:https: https://community.fs.com/ru/blog/tcpip-vs-osi-whats-the-difference-between-the-two-models.html — Назва з екрану.

17. Документація по мові XAML [Електронний ресурс]. — Режим доступу:https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/xaml-in-wpf— Назва з екрану.

18. Документація по мові C# [Електронний ресурс]. — Режим доступу:https://docs.microsoft.com/en-us/dotnet/framework/wpf/getting-started/— Назва з екрану.

19. Документація по мові XAML для Xamarin Forms [Електронний ресурс]. — Режим доступу:https://docs.microsoft.com/en-us/xamarin/xamarin-forms/xaml/ — Назва з екрану.

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

(Копія)

ПОГОДЖЕНО
Керівник дипломного проекту
к.т.н., доц. Тарабаров С. Б.

02.09.2020
(дата)(підпис)

ЗАТВЕРДЖЕНО
Завідувач кафедри
радіоконструювання та
виробництва радіоапаратури
д.т.н., проф. Нелін Є. А.

_____
(дата)(підпис)

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської дисертації
на тему «Дистанційне керування силовим навантаженням»

Київ — 2020 року

1. ПІДСТАВА ДЛЯ ВИКОНАННЯ ДИСЕРТАЦІЇ

Підставою для виконання магістерської дисертації є завдання

2. МЕТА І ПРИЗНАЧЕННЯ

Мета: на основі проведеного дослідження визначити необхідні технології та побудови нових алгоритмів, а також застосування їх при розробці застосувань для керування пристроями силового навантаження.

Об’єкт дослідження: сукупність засобів дистанційного керування силовим навантаженням.

Предмет дослідження: протоколи комунікації програмних та прикладних засобів керування в межах локальної мережі WiFi.

3. ВИХІДНІ ДАНІ ДЛЯ ВИКОНАННЯ ДИСЕРТАЦІЇ

Основні джерела:

1. Arduino Controlled Light Dimmer [Електронний ресурс]. — Режим доступу:https://www.instructables.com/id/Arduino-controlled-light-dimmer-The-circuit/ — Назва з екрану.

2. Бібліотека "AC_Dimmer" [Електронний ресурс]. — Режим доступу: https://github.com/AlexGyver/AC_Dimmer — Назва з екрану.

3. Бібліотека "RBDDimmer" [Електронний ресурс]. — Режим доступу: https://github.com/RobotDynOfficial/RBDDimmer — Назва з екрану.

4. ВИМОГИ ДО ВИКОНАННЯ ДИСЕРТАЦІЇ

1. Виявлення проблем сучасних пристроїв керування силовим навантаженням на прикладі розумного освітлення.
2. Аналіз можливих протоколів передачі даних.
3. Розгляд необхідних моделей для проєктування програмного рішення.
4. Розроблення драйверу для міктроконтролера, застосунків керування через веб-сторінку, окремий додаток для комп’ютера або смартфону.
5. Аналіз результатів.

5. ЕТАПИ ТА ТЕРМІНИ ЇХ ВИКОНАННЯ

Таблиця 1 — Етапи та терміни їх виконання

Шифр етапів	Назва етапів виконуваного завдання	Терміни виконання
1	Підбір джерел за темою МД	09.2020
2	Розробка ТЗ	09.2020
3	Дослідження моделей розробки ПЗ	10.2020
4	Розроблення алгоритмів веб-додатків	10.2020
5	Розроблення програмного забезпечення	11.2020
6	Оформлення та підготовка до захисту МД	12.2020

6. ОЧІКУВАНІ РЕЗУЛЬТАТИ

6.1 Розробка лістингу логіки мікроконтролера, веб-сторінки, додатку на смартфон та ПК.

6.2 Висновки за результатами виконаної роботи.

7. МАТЕРІАЛИ, ЯКІ ПОДАЮТЬСЯ ПІСЛЯ ЗАКІНЧЕННЯ ДИСЕРТАЦІЇ

7.1 Завдання.

7.2 Технічне завдання.

7.3 Пояснювальна записка.

7.4 Презентація.

8. ПОРЯДОК ПРИЙМАННЯ МАГІСТЕРСЬКОЇ ДИСЕРТАЦІЇ ТА ЇЇ ЕТАПІВ

8.1 Поетапне узгодження з керівником.

8.2 Представлення кафедрі.

8.3 Попередній захист магістерської дисертації.

8.4 Захист дисертації перед екзаменаційною комісією.

9. ВИМОГИ ДО РОЗРОБЛЕННЯ ДОКУМЕНТАЦІЇ

1. ДСТУ 3008-2015. Інформація та документація. Звіт у сфері науки і техніки. Структура та правила оформлення.
2. ДСТУ 3973-2000. Система розроблення та поставлення продукції на виробництво. Правила виконання науково-дослідних робіт. Загальні положення.

10. ВИМОГИ ДО РОЗРОБЛЕННЯ ДОКУМЕНТАЦІЇ

10.1 Огляд аналогів та схожих технологій.

10.2 Вибір технологій розробки.

10.3 Розробка алгоритмів.

10.4 Розробка програмного забезпечення.

Виконавець роботи  Драчук О.С.

ДОДАТОК Б Лістинг логіки для мікроконтролера

```
#include <ESP8266WebServer.h>
#include <RBDdimmer.h>
#define USE_SERIAL  Serial
#define zerocross 4
#define outputPin 5
 dimmerLamp dimmer(outputPin, zerocross);
 char* ssid = "Redmi 5";
 char* password = "AaBbCc135";
 int brightness = 8;
 int pos1 = 0;
 int pos2 = 0;
 String header;
 char webpage[] = R"=(
<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8" name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
 <title>Brightness Control</title>
  <style>
   * {
    margin: 0px;
    padding: 0px;
   }
   body {
   background-color: rgb(27, 41, 48);
   }
  .main {
   position: absolute;
   top:50%;
   left: 50%;
   transform: translate(-50%, -50%);
   width: 350px;
   height: 350px;
   border-radius: 10px;
   background: rgb(240, 240, 240);
  }
  .slider_bright {
   margin: 125px 25px 0px 25px;
```

```
  width: 300px;
  height: 10px;
  background: grey;
  border-radius: 20px;
}
#range {
  display: block;
  visibility: hidden;
  -webkit-appearance: none;
  appearance: none;
  width: 100%;
  height: 10px;
  background-color: rgb(27, 41, 48);
  border-radius: 20px;
  outline: none;
}
#range::-webkit-slider-thumb {
    -webkit-appearance: none;
    appearance: none;
    width: 25px;
    height: 25px;
    border-radius: 13px;
    background: rgb(250, 208, 0);
    cursor: pointer;
    outline: none;
}
#range::-moz-range-thumb {
    width: 25px;
    height: 25px;
    border-radius: 13px;
    background: rgb(250, 208, 0);
    outline: none;
    cursor: pointer;
  }
#range::-webkit-slider-thumb:hover{
    width: 30px;
    height: 30px;
    border-radius:16px;
  }
#demo {
  visibility: hidden;
```

```
    margin: 20px 105px 0px 105px;
    width: 140px;
    height:  35px;
    text-align: center;
   }
   #btn, #btn_test_start{
    -webkit-appearance: none;
    appearance: none;
    width: 100px;
    height: 40px;
    cursor: pointer;
    border: 1px solid rgb(27,41,48);
    background: rgb(27, 41, 48);
    color: rgb(240,240,240);
   }
   #btn {
    margin: 20px 125px 20px 125px;
   }
   #btn_test_start {
    margin: 20px 125px 20px 125px;
   }
   #btn:hover, #btn_test_start:hover {
    color: rgb(27, 41, 48);
    background: rgb(240,240,240);
    border: 1px solid rgb(27,41,48);
   }
   </style>
</head>
<body>
  <div class="main">
   <div class="slider_bright">
    <input type="range" min=0 max=100 value=50 class="slider" id="range"
   </div>
   <p id="demo">Brightness: 50%</p>
   <button id="btn" onclick="turn_on_off(this)" value="1">Turn on led</button>
   <button id="btn_test_start" onclick="test_button_start()">Test</button>
  </div>
  <script>
   var xhr = new XMLHttpRequest();
    var slider = document.getElementById("range");
    var output = document.getElementById("demo");
```

```
    var btn = document.getElementById("btn");
    slider.oninput = function() {
      xhr.open("GET", "/?Brightness=" + slider.value + "&" , true);
      xhr.timeout = 700;
      xhr.ontimeout = function(){
        xhr.abort();
      }
      xhr.send();
      output.innerHTML = "Brightness: " + slider.value + "%";
    }
   function turn_on_off(elem) {
    var slider = document.getElementById("range");
    var output = document.getElementById("demo");
    if ( elem.value == "1") {
      xhr.open("GET", "/LEDon", true);
          xhr.send();
      elem.value="2";
      elem.textContent = "Turn off led";
      slider.style.visibility = "visible";
      output.style.visibility = "visible";
    }
    else {
      xhr.open("GET", "/LEDoff", true);
          xhr.send();
      elem.value="1";
      elem.textContent = "Turn on led";
      slider.style.visibility = "hidden";
      output.style.visibility = "hidden";
      slider.value = "50";
      output.innerHTML = "Brightness: 50%";
    }
   }
    function test_button_start (){
      xhr.open("GET", "/Test_start", true);
          xhr.send();
    }
  </script>
</body>
</html>
)=";
WiFiServer server(80);
```

```
void setup() {
Serial.begin(115200);
dimmer.begin(NORMAL_MODE, ON);
dimmer.setPower(10);
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.hostname("brightness_control");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}
void loop(){
  WiFiClient client = server.available();
  if (client) {
    // If a new client connects,
    Serial.println("New Client.");
    String currentLine = "";
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        header += c;
        if (c == '\n') {
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();
              if(header.indexOf("GET /find_esp")>=0) {
              client.stop();}
            client.println(webpage);
            if(header.indexOf("GET /LEDon")>=0) {
              for (int i = 10; i < 51; i++)
```

```
          {
          dimmer.setPower(i);
          delay(10);
          }
          }
          if(header.indexOf("GET /LEDoff")>=0) {
          dimmer.setPower(10);
          }
         if(header.indexOf("GET /?Brightness=")>=0) {
          pos1 = header.indexOf('=');
          pos2 = header.indexOf('&');
          brightness = header.substring(pos1+1, pos2).toInt();
          dimmer.setPower(map(brightness,0, 100, 30, 100));
          }
         if(header.indexOf("GET /Test_start")>=0) {
          for (int i = 15; i < 36; i++)
          {
          dimmer.setPower(i);
          delay(100);
          }
          for (int i = 35; i>16 ; i--)
          {
            dimmer.setPower(i);
          delay(100);
          }
          }
         client.println();
         break;
       } else {
         currentLine = "";
       }
     } else if (c != '\r') {
       currentLine += c;
     }
   }
  }
  header = "";
  client.stop();
  Serial.println("Client disconnected.");
  Serial.println("");
}}
```

ДОДАТОК В Лістинг інтерфейсу додатку для ПК

```
<Window x:Class="Brightness_Control.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Brightness_Control"
        mc:Ignorable="d"
        Title="Wireless Brigthness Control" WindowStartupLocation="CenterScreen"
WindowStyle="None" AllowsTransparency="True" MinHeight="475" MinWidth="400" MaxHeight="475"
MaxWidth="400">
    <Window.Resources>
        <Style x:Key="SliderThumbStyle" TargetType="Thumb">
            <Setter Property="SnapsToDevicePixels" Value="True"/>
            <Setter Property="OverridesDefaultStyle" Value="False"/>
            <Setter Property="Height" Value="18"/>
            <Setter Property="Width" Value="18"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Thumb">
                        <Ellipse
                         StrokeThickness="0"
                         Name="Ellipse"
                         Fill="#FFDD55"/>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>

        <Style TargetType="Slider" x:Key="AppSliderStyle">
            <Setter Property="OverridesDefaultStyle" Value="true"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Slider">
                        <Grid>
                         <Border Name="PART_Border"
                         BorderBrush="#1B2930" BorderThickness="1"
                         Padding="2"
                         Width="5"
                         Height="{TemplateBinding Height}"
                         Background="#1B2930"
                         HorizontalAlignment="Stretch"
                         VerticalAlignment="Center" />
                          <Track Name="PART_Track"
                            HorizontalAlignment="Stretch"
                            VerticalAlignment="Center"
                            Width="{TemplateBinding Width}"
                            Height="{TemplateBinding Height}">
                              <Track.Thumb>
                                 <Thumb Style="{StaticResource SliderThumbStyle}" />
                              </Track.Thumb>
                          </Track>
                        </Grid>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
        <Style TargetType="Button" x:Key="MinimizedBtn">
            <Setter Property="Template">
```

```
            <Setter.Value>
               <ControlTemplate TargetType="Button">
                  <Border x:Name="border" Background="#F1F1F1" BorderThickness="0">
                     <ContentControl>
                        <Path x:Name="pa" Stroke="#ABABAB" StrokeThickness="2">
                           <Path.Data>
                              <GeometryGroup>
                                 <LineGeometry StartPoint="8,17" EndPoint="17,17"/>
                              </GeometryGroup>
                           </Path.Data>
                        </Path>
                     </ContentControl>
                  </Border>
                  <ControlTemplate.Triggers>
                     <Trigger Property="IsMouseOver" Value="True">
                        <Setter TargetName="border" Property="Background" Value="#E5E5E5"/>
                     </Trigger>
                  </ControlTemplate.Triggers>
               </ControlTemplate>
            </Setter.Value>
         </Setter>
      </Style>
      <Style TargetType="Button" x:Key="CloseBtn">
         <Setter Property="Template">
            <Setter.Value>
               <ControlTemplate TargetType="Button">
                  <Border x:Name="border" Background="#F1F1F1" BorderThickness="0">
                     <ContentControl>
                        <Path x:Name="pa" Stroke="#ABABAB" StrokeThickness="2">
                           <Path.Data>
                              <GeometryGroup>
                                 <LineGeometry StartPoint="8,8" EndPoint="17,17"/>
                                 <LineGeometry StartPoint="8,17" EndPoint="17,8"/>
                              </GeometryGroup>
                           </Path.Data>
                        </Path>
                     </ContentControl>
                  </Border>
                  <ControlTemplate.Triggers>
                     <Trigger Property="IsMouseOver" Value="True">
                        <Setter TargetName="border" Property="Background" Value="#E81123"/>
                        <Setter TargetName="pa" Property="Stroke" Value="#FDE8E9"/>
                     </Trigger>
                  </ControlTemplate.Triggers>
               </ControlTemplate>
            </Setter.Value>
         </Setter>
      </Style>

   </Window.Resources>

   <WindowChrome.WindowChrome>
      <WindowChrome CaptionHeight="25"/>
   </WindowChrome.WindowChrome>

   <Grid>
      <Canvas Background="#1B2930">
         <WrapPanel  Background="#F1F1F1" VerticalAlignment="Top" Height="25" Width="400">
            <WrapPanel Height="25" Width="25">
               <Image Source="105964493-light-lamp-sign-icon-bulb-with-gears-symbol.jpg" />
            </WrapPanel>
            <Label Height="25" Width="325" Content="Wireless Brightness Control"/>
```

```
                <WrapPanel HorizontalAlignment="Left" Height="25" Width="50"
WindowChrome.IsHitTestVisibleInChrome="True">
                    <Button Name="MinimizeBtn" Style="{StaticResource MinimizedBtn}" Height="25"
Width="25" Background="#F1F1F1"/>
                    <Button Name="CloseBtn" Style="{StaticResource CloseBtn}" Height="25" Width="25"
Background="#F1F1F1"/>
                </WrapPanel>
            </WrapPanel>
            <StackPanel Height="400" Width="350" Margin="25,50,25,0">

                <Border BorderThickness="0" CornerRadius="15" Background="#FFFFFF" Height="50">
                    <WrapPanel>
                        <WrapPanel Height="30" Width="206" Margin="16,10,0,0">
                            <Label Content="Lamp ip:" HorizontalContentAlignment="Center" Height="20"
Width="56" Padding="0" FontSize="15"/>
                            <Label Content="Enter connect button" Height="30" Width="150" Name="Result"
FontSize="15" />
                        </WrapPanel>
                        <Label Name="btn" Content="Connect" BorderThickness="1"
BorderBrush="#1B2930" Foreground="#1B2930" Width="100" Height="30" Margin="14,10,14,0"
FontSize="15" Padding="0" VerticalContentAlignment="Center" HorizontalContentAlignment="Center"
PreviewMouseLeftButtonDown="Connect_Button" MouseEnter="btn_mouse_enter"
MouseLeave="btn_mouse_leave"/>
                    </WrapPanel>
                </Border>

                <Border BorderBrush="#1B2930" BorderThickness="0" CornerRadius="15" Height="250"
Background="#FFFFFF" Margin="0,25,0,0">
                    <WrapPanel>
                        <Slider Name="slider1" Visibility="Hidden" Background="#FFFFFF"
Style="{StaticResource AppSliderStyle}" Orientation="Vertical" ValueChanged="Slider_ValueChanged"
Value="50" Minimum="0" Maximum="100" SmallChange="1" Width="20" Height="150"
Margin="100,50,0,50"/>
                        <StackPanel Name="on" Height="150" Width="100" Margin="65,50,0,50"
PreviewMouseLeftButtonDown="Led_on">
                            <Image Name="light" Source="light_off.jpg" Height="150" Width="100"/>
                        </StackPanel>
                    </WrapPanel>
                </Border>

                <Border Name="copy" BorderThickness="1" CornerRadius="15"
Background="#FFFFFF" Height="50" Margin="0,25,0,0">
                    <Label Name="lb_copy" Content="Copy wifi lamp ip" FontSize="15"
Foreground="#1B2930" HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
PreviewMouseLeftButtonDown="Copy" MouseEnter="btn_copy_enter" MouseLeave="btn_copy_leave"/>
                </Border>
            </StackPanel>
        </Canvas>
    </Grid>
</Window>
```

ДОДАТОК Г Лістинг логіки для додатку на ПК

```
using System.Threading.Tasks;
using System.Windows;
using System.Net.NetworkInformation;
using System.Net.Sockets;
using System.Net;
using System.Windows.Controls;
using System.Windows.Media;
using System;
using System.Windows.Media.Imaging;

namespace Brightness_Control
{
   public partial class MainWindow : Window
   {
      private delegate void updateDelegate(string text);
      private int timeout_p = 100;
      private int timeout_r = 700;
      string esp_ip = "";
      int value = 1;
      bool localip = false;
      bool connected = false;

      public MainWindow()
      {
         InitializeComponent();
         MinimizeBtn.Click += (s, e) => WindowState = WindowState.Minimized;
         CloseBtn.Click += (s, e) => Close();
      }
      private void updateButton_start(string text)
      {
         btn.Content = text;
      }

      private void updateButton_end(string text)
      {
         btn.Content = text;
         Result.Content = esp_ip;
      }

      private async void Connect_Button(object sender, RoutedEventArgs e)
      {
         if (!connected)
         {
            await btn.Dispatcher.BeginInvoke(new updateDelegate(updateButton_start), "Connecting");
            var host = Dns.GetHostEntry(Dns.GetHostName());
            foreach (var ip in host.AddressList)
            {
               if (ip.AddressFamily == AddressFamily.InterNetwork)
               {
                  if (ip.ToString().Contains("192.168."))
                  {
                     localip = true;
                     for (int i = 0; i <= 255; i++)
                     {
                        Ping p = new Ping();
                        var task = PingAndUpdateAsync(p, ip.ToString().Substring(0,
ip.ToString().LastIndexOf(".")) + "." + i.ToString());
                     }
```

```
                    await btn.Dispatcher.BeginInvoke(new updateDelegate(updateButton_start),
"Connect");
                }
            }
        }
        if (localip == false)
        {
            await btn.Dispatcher.BeginInvoke(new updateDelegate(updateButton_start), "Connect");
            MessageBox.Show("Please, connect to wifi!");
        }
    }
}

private async Task PingAndUpdateAsync(Ping ping, string ip)
{
    var reply = await ping.SendPingAsync(ip, timeout_p);

    if (reply.Status == IPStatus.Success)
    {
        await Task.Run(() => Request(reply));
    }
}

private void Request(PingReply reply)
{
    try
    {
        var mYrequest = (HttpWebRequest)WebRequest.Create("http://" + reply.Address.ToString() +
"/find_esp");
        mYrequest.Timeout = timeout_r;
        HttpWebResponse mYresponse = (HttpWebResponse)mYrequest.GetResponse();
        if ((mYresponse.StatusCode == HttpStatusCode.OK))
        {
            esp_ip = reply.Address.ToString();
            connected = true;
            btn.Dispatcher.BeginInvoke(new updateDelegate(updateButton_end), "Connected");
        }
        mYresponse.Close();
    }
    catch { }
}

private void Led_on(object sender, RoutedEventArgs e)
{
    if (esp_ip.Contains("192.168."))
    {
        if (value == 1)
        {
            try
            {
                var mYrequest = (HttpWebRequest)WebRequest.Create("http://" + esp_ip + "/LEDon");
                mYrequest.Timeout = timeout_r;
                HttpWebResponse mYresponse = (HttpWebResponse)mYrequest.GetResponse();
                mYresponse.Close();
                BitmapImage image = new BitmapImage(new Uri("light_on.jpg", UriKind.Relative));
                light.Source = image;
                slider1.Value = 50;
                slider1.Visibility = Visibility.Visible;
                value = 2;
            }
            catch
            {
```

```
            }
        }
        else
        {
            try
            {
                var mYrequest = (HttpWebRequest)WebRequest.Create("http://" + esp_ip + "/LEDoff");
                mYrequest.Timeout = timeout_r;
                HttpWebResponse mYresponse = (HttpWebResponse)mYrequest.GetResponse();
                mYresponse.Close();
                BitmapImage image = new BitmapImage(new Uri("light_off.jpg", UriKind.Relative));
                light.Source = image;
                slider1.Visibility = Visibility.Hidden;
                value = 1;
            }
            catch { }
        }
    }
    else
    {
        MessageBox.Show("Please, connect to wifi and your smart lamp!");
    }
}

private void Slider_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
{
    if (esp_ip.Contains("192.168."))
    {
        try
        {
            ((Slider)sender).SelectionEnd = e.NewValue;
            var mYrequest = (HttpWebRequest)WebRequest.Create("http://" + esp_ip + "/?Brightness="
+ e.NewValue + "&");
            mYrequest.Timeout = timeout_r;
            HttpWebResponse mYresponse = (HttpWebResponse)mYrequest.GetResponse();
            mYresponse.Close();
        }
        catch { }
    }
}

private void Copy(object sender, RoutedEventArgs e)
{
    if (esp_ip.Contains("192.168."))
    {
        Clipboard.Clear();
        Clipboard.SetText(Result.Content.ToString());
    }
    else
    {
        MessageBox.Show("Please, connect to wifi and your smart lamp!");
    }
}

private void btn_mouse_enter(object sender, System.Windows.Input.MouseEventArgs e)
{
    btn.Background = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#1B2930"));
    btn.Foreground = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#FFFFFF"));
}

private void btn_mouse_leave(object sender, System.Windows.Input.MouseEventArgs e)
```

```
        {
            btn.Foreground = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#1B2930"));
            btn.Background = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#FFFFFF"));
        }

        private void btn_copy_enter(object sender, System.Windows.Input.MouseEventArgs e)
        {
            copy.Background = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#1B2930"));
            copy.BorderBrush = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FFFFFF"));
            lb_copy.Foreground = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#FFFFFF"));
        }

        private void btn_copy_leave(object sender, System.Windows.Input.MouseEventArgs e)
        {
            lb_copy.Foreground = new
SolidColorBrush((Color)ColorConverter.ConvertFromString("#1B2930"));
            copy.BorderBrush = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#1B2930"));
            copy.Background = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#FFFFFF"));
        }
    }
}
```

ДОДАТОК Д Лістинг інтерфесу для додатку на смартфон

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:lightapp"
             x:Class="lightapp.MainPage">

    <Grid BackgroundColor="#1B2930">
        <StackLayout Margin="0,25,0,25" HorizontalOptions="FillAndExpand" VerticalOptions="Center">
            <Frame CornerRadius="15" BackgroundColor="#FFFFFF">
                <StackLayout   Orientation="Horizontal" HorizontalOptions="FillAndExpand">
                    <StackLayout Orientation="Horizontal" HorizontalOptions="FillAndExpand">
                    <Label Text="Lamp ip:" VerticalTextAlignment="Center" FontSize="14"
TextColor="#1B2930"/>
                    <Label Text="Enter connect button" VerticalTextAlignment="Center" FontSize="14"
TextColor="#1B2930" x:Name="Result" HorizontalOptions="Start"/>
                    </StackLayout>
                    <Button Text="Connect" x:Name="btn_c" FontSize="12" WidthRequest="100"
BackgroundColor="#FFDD55" TextColor="#1B2930" Clicked="Connect_Button" HorizontalOptions="End"
/>
                </StackLayout>
            </Frame>
            <Frame CornerRadius="15" BackgroundColor="#FFFFFF" Margin="0,25,0,0"
HeightRequest="250">
                <AbsoluteLayout HorizontalOptions="FillAndExpand">
                    <Slider x:Name="slider1" IsEnabled="False" MinimumTrackColor="#1B2930"
MaximumTrackColor="#1B2930" ValueChanged="Slider_ValueChanged" Minimum="0" Value="50"
Maximum="100" ThumbColor="#FFDD55" Rotation="-90" AbsoluteLayout.LayoutBounds="-
0.2,0.5,200,20" AbsoluteLayout.LayoutFlags="PositionProportional"/>
                    <ImageButton x:Name="on" Clicked="Led_on" BackgroundColor="#0000"
Source="{local:ImageResource lightapp.light_off.jpg}" AbsoluteLayout.LayoutBounds="0.5,0.5,115,275"
AbsoluteLayout.LayoutFlags="PositionProportional"/>
                </AbsoluteLayout>
            </Frame>
            <Frame CornerRadius="15" BackgroundColor="#FFFFFF" HeightRequest="50"
Margin="0,25,0,25">
                <StackLayout HorizontalOptions="FillAndExpand">
                    <Button Text="Copy wifi lamp ip" Clicked="Copy" BackgroundColor="#FFDD55"
TextColor="#1B2930"/>
                </StackLayout>
            </Frame>
        </StackLayout>
    </Grid>
</ContentPage>
```

ДОДАТОК Е Лістинг логіки для додатку на смартфон

```
using System;
using System.ComponentModel;
using System.Threading.Tasks;
using Xamarin.Forms;
using System.Net;
using System.Net.NetworkInformation;
using System.Net.Sockets;
using Xamarin.Essentials;
using Xamarin.Forms.Xaml;

namespace lightapp
{
    [DesignTimeVisible(true)]
    [ContentProperty("Source")]
    public class ImageResourceExtension : IMarkupExtension
    {
        public string Source { get; set; }

        public object ProvideValue(IServiceProvider serviceProvider)
        {
            if (Source == null)
            {
                return null;
            }
            var imageSource = ImageSource.FromResource(Source);

            return imageSource;
        }
    }

    public partial class MainPage : ContentPage
    {
        private delegate void updateDelegate(string text);
        private string my_ip = "";
        private int StartIP = 0;
        private int StopIP = 255;
        private int timeout_p = 100;
        private int timeout_r = 700;
        string esp_ip = "";
        string esp_ip1 = "";
        int value = 1;
        bool localip = false;
        bool connected = false;
        string filename = "esp_ip.txt";
        public MainPage()
        {
            InitializeComponent();
        }
        private void updateButton_connecting()
        {
            btn_c.Text = "Connecting";
        }
        private void updateButton_connect()
        {
            btn_c.Text = "Connect";
        }
        private void updateButton_end()
        {
```

```
            btn_c.Text = "Connected";
            Result.Text = esp_ip;

        }
        private void updateButton_connected()
        {
            Result.Text = esp_ip;
            btn_c.Text = "Connected";
            DependencyService.Get<IFileWorker>().SaveTextAsync(filename, esp_ip);
        }
        private async void  Connect_Button(object sender, EventArgs e)
        {
            if (!connected)
            {
                Device.BeginInvokeOnMainThread(updateButton_connecting);
                var host = Dns.GetHostEntry(Dns.GetHostName());
                foreach (var ip in host.AddressList)
                {
                    if (ip.AddressFamily == AddressFamily.InterNetwork)
                    {
                        if (ip.ToString().Contains("192.168."))
                        {
                            localip = true;
                            my_ip = ip.ToString();
                            esp_ip1 = my_ip.Substring(0, my_ip.LastIndexOf(".")) + ".";
                        }
                    }

                }
                if (localip == false)
                {
                    Device.BeginInvokeOnMainThread(updateButton_connect);
                    await DisplayAlert("Error", "Please, connect to wifi!", "OK!");
                }

                    if (await DependencyService.Get<IFileWorker>().FileNotEmptyAsync(filename) && (localip
== true))
                    {
                        try
                        {
                            string check = await DependencyService.Get<IFileWorker>().LoadTextAsync(filename);
                            var myrequest = (HttpWebRequest)WebRequest.Create("http://" + check + "/find_esp");
                            myrequest.Timeout = 700;
                            HttpWebResponse myresponse = (HttpWebResponse)myrequest.GetResponse();
                            if ((myresponse.StatusCode == HttpStatusCode.OK))
                            {
                                esp_ip = check;
                                connected = true;
                                Device.BeginInvokeOnMainThread(updateButton_end);
                            }
                            myresponse.Close();
                        }
                        catch
                        {
                            await DependencyService.Get<IFileWorker>().SaveTextAsync(filename, "");
                        }
                    }
                    if (!(await DependencyService.Get<IFileWorker>().FileNotEmptyAsync(filename)) && (localip
== true))
                    {
                        for (int i = StartIP; i <= StopIP; i++)
                        {
```

```
                    Ping p = new Ping();
                    Task task = PingAndUpdateAsync(p, esp_ip1 + i.ToString());
                }
            }
        }
    }

    private async Task PingAndUpdateAsync(Ping ping, string ip)
    {
        var reply = await ping.SendPingAsync(ip, timeout_p);

        if (reply.Status == IPStatus.Success)
        {
            if (reply.Address.ToString() != my_ip)
            {
                await Task.Run(() => Request(reply));
            }
        }
        else if (reply.Address.ToString().Contains("255") && !connected)
        {
            Device.BeginInvokeOnMainThread(updateButton_connect);
        }
    }

    private void Request(PingReply reply)
    {
        try
        {
            var mYrequest = (HttpWebRequest)WebRequest.Create("http://" + reply.Address.ToString() +
"/find_esp");
            mYrequest.Timeout = timeout_r;
            HttpWebResponse mYresponse = (HttpWebResponse)mYrequest.GetResponse();
            if ((mYresponse.StatusCode == HttpStatusCode.OK))
            {
                esp_ip = reply.Address.ToString();
                connected = true;
                Device.BeginInvokeOnMainThread(updateButton_connected);
            }
            mYresponse.Close();
        }
        catch {}

    }
    private void Led_on(object sender, EventArgs e)
    {
        try
        {
            if (esp_ip.Contains("192.168."))
            {
                if (value == 1)
                {
                    try
                    {
                        var mYrequest = (HttpWebRequest)WebRequest.Create("http://" + esp_ip + "/LEDon");
                        mYrequest.Timeout = timeout_r;
                        HttpWebResponse mYresponse = (HttpWebResponse)mYrequest.GetResponse();
                        mYresponse.Close();
                        on.Source = ImageSource.FromResource("lightapp.light_on.jpg");
                        slider1.Value = 50;
                        slider1.IsEnabled = true;
                        value = 2;
                    }
```

```
                    catch { }
                }
                else
                {
                    try
                    {
                        var mYrequest = (HttpWebRequest)WebRequest.Create("http://" + esp_ip + "/LEDoff");
                        HttpWebResponse mYresponse = (HttpWebResponse)mYrequest.GetResponse();
                        mYresponse.Close();
                        on.Source = ImageSource.FromResource("lightapp.light_off.jpg");
                        slider1.IsEnabled = false;
                        value = 1;
                    }
                    catch { }
                }
            }
            else
            {
                DisplayAlert("Error", "Please, connect to wifi and your smart lamp!", "OK!");
            }
        }
        catch { }
    }
    private void Slider_ValueChanged(object sender, ValueChangedEventArgs e)
    {
        try
        {
            if (esp_ip.Contains("192.168."))
            {
                try
                {
                    var mYrequest = (HttpWebRequest)WebRequest.Create("http://" + esp_ip +
"/?Brightness=" + e.NewValue + "&");
                    mYrequest.Timeout = timeout_r;
                    HttpWebResponse mYresponse = (HttpWebResponse)mYrequest.GetResponse();
                    mYresponse.Close();
                }
                catch
                { }
            }
        }
        catch { }
    }
    private void Copy(object sender, EventArgs e)
    {
        try
        {
            if (esp_ip.Contains("192.168."))
            {
                Clipboard.SetTextAsync(Result.Text);
            }
            else
            {
                DisplayAlert("Error", "Please, connect to wifi and your smart lamp!", "OK!");
            }
        }
        catch { }
    }
  }
}
```