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

Радіотехнічний факультет
(повна назва інституту/факультету)

Кафедра радіоконструювання та виробництва радіоапаратури
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_______ Є.А. Нелін
(підпис) (ініціали, прізвище)

“13” 06 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

За напрямом підготовки 6.050902 Радіоелектронні апарати
(код та назва спеціальності)

на тему: Пристрій мережі інтернет речей для керування силовим навантаженням (на прикладі розумної лампи)

Виконав (-ла): студент (-ка) IV курсу, групи РК-51
(шифр групи)

Драчук Олександр Сергійович _______
(прізвище, ім'я, по батькові) (підпис)

Керівник старший викладач, к.т.н. Навроцький Д.О. _______
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з охорони праці доцент, к.т.н., Каштанов С.Ф. _______
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент асистент, Миронюк О.Ю. _______
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _______
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет (інститут) радіотехнічний
(повна назва)

Кафедра радіоконструювання та виробництва радіоапаратури
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

За напрямом підготовки 6.050902 Радіоелектронні апарати
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Є.А. Нелін
(підпис) (ініціали, прізвище)
«10» травня 2019 р.

ЗАВДАННЯ
на дипломний проект (роботу) студенту

Драчуку Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Пристрій передачі інтернет речей для керування силовим навантаженням (на прикладі розумної лампи)
керівник проекту (роботи) Нефедський Денис Олександрович ст. викл., к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «27» травня 2019 р. № 1399-с

2. Строк подання студентом проекту (роботи) 14.06.2019р

3. Вихідні дані до проекту (роботи) Напруга живлення 220 В, використання бездротового способу передачі даних, керування за допомогою програмного забезпечення.

4. Зміст (дипломної роботи) розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) Вступ; Огляд існуючих рішень. Розробка та аналіз технічного завдання; Обґрунтування та вибір схемо-технічного рішення; Розробка програмного забезпечення; Проектування

приводу та перевірка їхньої працездатності; Охорони праці.

5. Перелік (ілюстративного) графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо) Структурна схема алгоритму програми 1, Алгоритми програми 2

6. Консультанти розділів проекту (роботи)*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
З охорони праці	к.т.н., доцент Каштанов С.Ф.		

7. Дата видачі завдання 15 травня 2019 року

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд існуючих рішень	13.04. - 16.04.2019р	Виконав
2	Розробка та аналіз технічного завдання	18.04. - 21.04.2019р	Виконав
3	Обґрунтування та вибір схемотех. рішень	22.04. - 23.04.2019р	Виконав
4.	Вибір та обґрунтування елементної бази	25.04. - 29.04.2019р	Виконав
5.	Розробка програмного забезпечення	30.04. - 02.05.2019р	Виконав
6.	Проектування приводу	03.05. - 06.05.2019р	Виконав
7.	Аналіз працездатності	07.05 - 09.05.2019р	Виконав
8.	Охорона праці	10.05 - 14.05.2019р	Виконав
9.	Оформлення техн. та констр. документ.	14.05. - 16.05.2019р	Виконав

Студент _____ (підпис) О.С. Драчук (ініціали, прізвище)

Керівник проекту (роботи) _____ (підпис) Д.О. Навроцький (ініціали, прізвище)

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

АНОТАЦІЯ

У дипломному проекті розроблено пристрій мережі інтернет речей для керування електричним навантаженням (на прикладі електронної лампи).

Метою є розробка пристрою з можливістю бездротового керування за допомогою смартфона чи комп'ютера для зміни яскравості лампи.

Проводиться огляд існуючих рішень та пристроїв для виправлення їх недоліків, розроблено структурну схему, обрано потрібні електронні модулі. Розглянуто існуючі мови програмування та обґрунтовано вибір мов, як JS, C/C++, HTML/CSS, XAML, C# для написання забезпечення на смартфон та комп'ютер. Проведено тести на працездатність приладу та додатків.

Ключові слова: ESP8266, димер, WiFi, HTTP, мови програмування, веб-додатки.

ANNOTATION

The graduation project presents the elaboration of IoT device for the operation of electrical power load (for example electrical lamp).

The aim of the project is to develop a wireless remote control of a device with a smartphone or a computer for changing the lamp brightness. The project presents a block diagram, electronic module and an overview of existing solutions and devices for addressing the deficiencies. We considered available programming languages and substantiated the choice of such languages as JS, C/C++, HTML/CSS, XAML, C# for writing a smartphone or computer software. Also operability tests of the device and web applications were conducted.

Key words: ESP8266, dimmer, WiFi, HTTP, programming languages, web applications.

АНОТАЦІЯ

У дипломному проекті розроблено пристрій мережі інтернет речей для керування електричним навантаженням (на прикладі електронної лампи).

Метою є розробка пристрою з можливістю бездротового керування за допомогою смартфона чи комп’ютера для зміни яскравості лампи.

Проводиться огляд існуючих рішень та пристроїв для виправлення їх недоліків, розроблено структурну схему, обрано потрібні електронні модулі. Розглянуто існуючі мови програмування та обґрунтовано вибір мов, як JS, C/C++, HTML/CSS, XAML, C# для написання забезпечення на смартфон та комп’ютер. Проведено тести на працездатність приладу та додатків.

Ключові слова: ESP8266, димер, WiFi, HTTP, мови програмування, веб-додатки.

ANNOTATION

The graduation project presents the elaboration of IoT device for the operation of electrical power load (for example electrical lamp).

The aim of the project is to develop a wireless remote control of a device with a smartphone or a computer for changing the lamp brightness. The project presents a block diagram, electronic module and an overview of existing solutions and devices for addressing the deficiencies. We considered available programming languages and substantiated the choice of such languages as JS, C/C++, HTML/CSS, XAML, C# for writing a smartphone or computer software. Also operability tests of the device and web applications were conducted.

Key words: ESP8266, dimmer, WiFi, HTTP, programming languages, web applications.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту

на тему: Пристрій мережі інтернет речей для керування силовим навантаженням (на прикладі розумної лампи)

Київ — 2019 року

ЗМІСТ

					РК51.421211.001 ПЗ		
ЗМ.	Лис	№ докум.	Підпис	Дата			
Розробив		Драчук			Пристрій мережі інтернет речей для керування силовим навантаженням(на прикладі розумної лампи)	Літ.	Лист 1 / Листів
Переві-		Навроцький					
Н.						РК-51 РТФ	
Затвер-		Навроцький					

ПЕРЕЛІК СКОРОЧЕНЬ

ШІМ — широтно-імпульсна модуляція;

Arduino IDE — інтегроване середовище розробки під плати Arduino;

COM-port — послідовний інтерфейс передачі інформації;

CSS — мова каскадних таблиць стилів;

DHCP — мережевий сертифікат для автоматичного видання IP-адреси;

HTML — мова гіпертекстової розмітки сторінки;

HTTP — протокол передачі гіпертексту, даних;

IDE — інтегроване середовище розробки;

IP-адреса — ідентифікатор пристрою мережевого рівня;

WEP — старий стандарт захисту бездротового трафіку;

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

XAML — декларативна мова розмітки інтерфейсу додатків;

ВСТУП

Сьогодні люди шукають нові способи економії свого часу на побутові клопоти. З розвитком сучасних технологій людині стають доступні все кращі методи вирішення побутових проблем і людина може справлятись з ними, не докладаючи при цьому багато зусиль.

Існує велика кількість різних датчиків, які можна застосувати в побуті, поєднавши їх із зручним віддаленим керуванням. Сучасна електроніка дає змогу створити пристрої моніторингу, керування опаленням, освітленням, відео наглядом або системою сигналізації, використовуючи дистанційне керування.

Розвиток технологій передачі даних дає змогу використовувати безпровідні способи передачі, як, наприклад, канали радіозв'язку. Для втілення таких технологій не потрібно витрачати суттєві кошти у зв'язку з широкою популярністю таких методів. Досліджені переваги, недоліки та принципи роботи безпровідної передачі даних дозволяють швидко та менш затратно втілювати використання мереж такого типу у відомих нам пристроях.

Звичайному користувачу добре відомі такі технології, як WiFi-мережа або 3G. Нещодавно ввели 4G – покриття наступного покоління, що дає значний приріст швидкості в порівнянні з минулим поколінням.

Електричні пристрої доволі тісно поєднані з мовами програмування. Швидкий ріст потужності мікроконтролерів в свою чергу дозволяє використовувати мови програмування високого рівня. Такий підхід надає можливості для створення більш розвиненого та швидкодіючого програмного забезпечення на базі існуючих електричних приладів.

Можемо зробити висновок, що поєднання сучасних технологій дозволяє покращити рівень комфортного користування для сучасного користувача. Окрім безпровідної передачі даних на допомогу приходять сучасні мережі розробки програмного забезпечення для смартфонів, що дають можливість

створити зрозумілу, з точки зору користувача, програму для керування як інформацією, що надходить з пристроїв, так і відправленням певних команд на пристрої з будь-якої точки дому або міста, не витрачаючи час на пошук пульта керування кожним приладом.

Отже, мета дипломного проекту полягає у розробці програмного забезпечення з можливістю бездротової передачі даних для покращення звичної нам електроніки.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ. РОЗРОБКА ТА АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Огляд існуючих аналогів на ринку

Насамперед варто розглянути аналоги на ринку зі схожими характеристиками та потребами.

Після огляду стає зрозуміло, що значна кількість виробників розробляє лампи із дистанційним типом керування. Проте, більшість представників використовують керування пультом, що не є дуже зручним методом роботи з такими пристроями, адже такий пульт працює в обмеженому радіусі дії і не знаходиться під рукою. Розглянемо лампи керування у додатку на ноутбуці або смартфоні. Виявляється, що небагато виробників розробляють такі прилади. Виділити можна такі фірми, як Yeelight [1], WIZ [2] та MiPow [3]. Виробники працюють майже з однаковими характеристиками і єдине, що їх відрізняє – це програмне забезпечення і його можливості. Для початку розглянемо більш предметно прилад від компанії Yeelight. Конструкція такої лампи зображена на рис. 1.1.

Рисунок 1.1 — Розумна лампа Yeelight [1]

Виробник вказує, що ця лампа розрахована на потужність 10 Вт, має вбудовані модулі для керування WiFi та Bluetooth, а її габаритні розміри —

130 мм х 65 мм х 65 мм. Важливим мінусом є те, що не вказується наявність власного програмного забезпечення. Ця фірма посилається на існуючі програмні забезпечення інших розробників. У відгуках користувачі висловлюють незадоволення щодо відсутності рідних модулів некоректної роботи лампи.

Розглянемо наступну лампу від компанії WIZ, зображену на рис. 1.2.

Рисунок 1.2 — Розумна лампа WIZ [2]

Зовнішнім виглядом практично не відрізняється від попередньої. Виробник вказує, що лампа розрахована на потужність 11 Вт та має модуль управління WiFi. У цій моделі в комплекті надається пульт керування, хоча користувач також може скачати додаток. Проте, функціонал пульту управління є більш розвинутим, ніж в додатку. З боку користувачів цієї лампи є нарікання на програмне забезпечення із зауваженням, що пульт працює більш надійно.

Розглянемо ще один цікавий екземпляр – лампа MiPow, зображена на рис. 1.3.

Ця лампа працює дистанційно за допомогою додатку на смартфон, але використовує виключно Bluetooth.

Рисунок 1.3— Розумна лампа WIZ [3]

При детальному аналізі стає зрозуміло, що через особливості технології Bluetooth до лампи не може підключитись одночасно більше ніж один пристрій та діапазон управління лампою в порівнянні із технологією WiFi помітно зменшується, що є суттєвими недоліками. Також користувачі зазначають, що використання Bluetooth швидко розряджає батарею смартфона.

Можемо зробити висновок, що виробники ще не виробляють завершені продукти з точки зору синергії програмного забезпечення із самим приладом. Варто зазначити і те, що якість ламп такого типу не відповідає їх вартості.

Отже, виробники намагаються вводити все більше пристроїв для системи розумного дому, але проблема полягає в їх реалізації та співвідношенні ціни до якості. Тому завданнями нашого дипломного проекту є досягнення показників швидкодії відклику пристрою на зміну рівня освітленості користувачем, забезпечення стабільного з’єднання між гаджетом (програмним забезпечення) та лампою і відкидання використання пульта та технології Bluetooth.

1.2 Аналіз схемотехнічних рішень

Оскільки звичайні лампи живляться від мережі 220 В, то і в нашому випадку доцільно використати живлення такого ж типу. Також для забезпечення зміни яскравості потрібен механізм зміни вихідної потужності. Розглянемо можливі схемотехнічні рішення для функціональних частин.

Насамперед варто розглянути «мозок» цього пристрою — мікроконтролер, повинен містити в собі або бути з'єднаним із приймачем та передавачем. Відповідно до такого завдання можемо обрати мікроконтролер ESP8266 [4], що зображений на рис. 1.4.

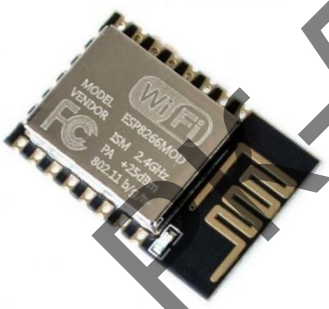

Рисунок 1.4 — Мікроконтролер ESP8266 [4]

Його важливою характеристикою є розміри (15 мм х 25 мм), адже вони дозволяють не сильно збільшувати габарити розроблюваного пристрою. ESP8266 підтримує роботу із WiFi IEEE 802.11 b/g/n, що відповідає за набір стандартів безпровідної передачі даних у локальній мережі частотного діапазону 2,4 ГГц при потужності сигналу не більше 100 мВт. Також доступна підтримка технологій WEP (Wired Equivalent Privacy) та WPA/WPA2 (Wi-Fi Protected Access), що відповідають за безпеку та конфіденційність даних при використанні безпровідних мереж зв'язку.

ESP8266 оснащений 80 МГц 32-bit процесором. Розробник не вказує в datasheet розмір оперативної пам'яті, але із приблизної оцінки ентузіастів можна дізнатись, що її розмір приблизно дорівнює 80 кБ. Для порівняння, плати Arduino на базі мікроконтролера Atmega328 мають тактову частоту процесора та розмір оперативної пам'ять 16 МГц та 32 кБ відповідно. Звідси

випливає, що характеристик ESP8266 достатньо, щоб запустити на ньому невеликий веб-сервер і керувати пристроєм, а маленькі розміри пригодяться у випадку обмежених габаритах 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.

Проаналізувавши інформацію в мережі інтернет більш детально, було розглянуто такий пристрій, як димер, що використовується для плавного регулювання потужністю. Його застосування доволі розповсюджено у приладах освітлення. Оскільки ним можна плавно регулювати вихідні характеристики, то димер можна використовувати для нагрівання, коли на виході замість лампи стоїть нагрівальний елемент, для живлення моторів, тощо. Через розповсюдження цього пристрою існує дуже багато інформації, яка допоможе при розробці.

Розглянемо основний принцип роботи димеру. Для прикладу візьмемо наступну реалізацію [5], схема якої вказана на рис. 1.5.

Весь принцип побудований навколо таких елементів, як симістор та схема “детектора нуля”. Розглянемо детальніше ці частини.

Симістор відкривається у випадку подачі невеликої напруги на керуючий електрод, а закривається при її відсутності. Оскільки симістор пропускає струм в обидва напрямки, то необхідність називати виводи симістора як “Катод” та “Анод” відпадає. Через це у datasheet частіше можна зустріти позначення A1, A2, та G, де G – керуючий електрод (англ. «gate» — ворота).

Звідси випливає, що ми можемо застосувати регулювання фази. Симістор повністю відкритий під час синусоїдальної хвилі змінного струму. Можемо вмикати та вимикати симістор на певну кількість мікросекунд, регулюючи таким чином вихідний сигнал.

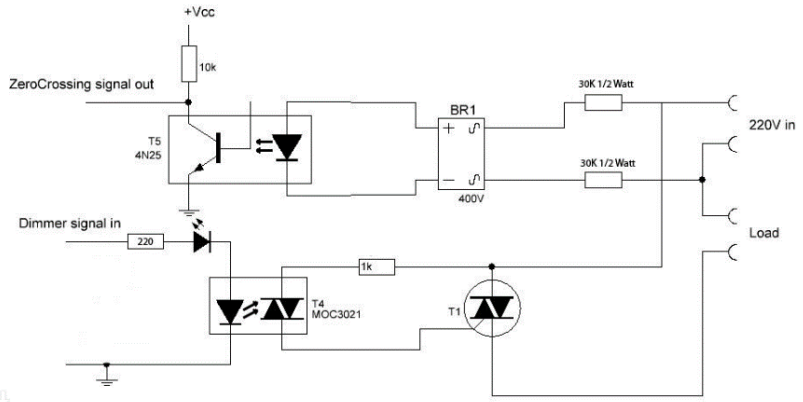

Рисунок 1.5 — Схема димеру [5]

Проблема полягає в тому, що ми не можемо передбачити в якій точці синусоїдальної хвилі симістор відкривається, а значить, ми не можемо передбачити рівень освітлення. Це означає, що є потреба у відслідковуванні “нульової” точки відліку на синусоїді. Такою задачею займається так званий “детектор нуля”, схема якого зображена на рис. 1.6. При перетині синусоїдою осі абсцис передається відповідний сигнал на мікроконтролер, який в свою чергу подає керуючий сигнал на симістор, відкриваючи його на певну кількість мікросекунд. В такому випадку ми можемо встановити бажаний рівень освітлення.

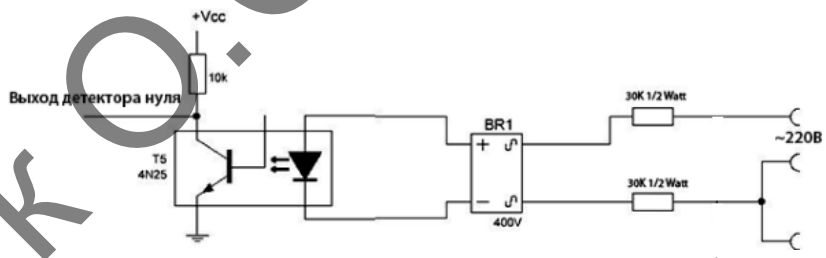

Рисунок 1.6 — Схема “детектора нуля” з виводом на мікроконтролер [5]

Діодний міст забезпечує двофазне виправлення, як зазначено на рис. 1.7. Далі сигнал перетину осі абсцис синусоїдою передається через оптрон на відповідний контакт мікроконтролеру, що повідомляє йому про потребу відкриття симістора на потрібну кількість мікросекунд. Варто зазначити, що оптрон в цій схемі потрібен для гальванічної розв’язки між мікроконтролером та мережею 220 В. Таким чином забезпечується захист мікроконтролера від небажаних перепадів напруги, зниження шумів, тощо.

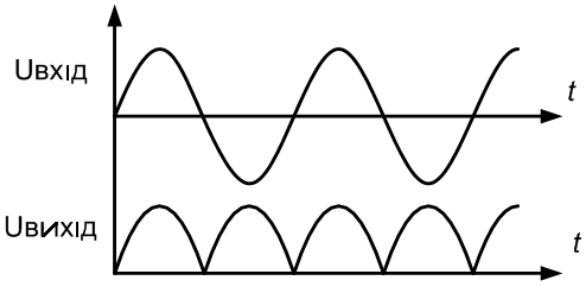

Рисунок 1.7 — Осцилограма сигналу випрямленого діодним містом

Порівнявши вхідні та вихідні сигнали при подачі на димер живлення із мережі 220 В, отримуємо результат, зображений на рис. 1.8.

Як розуміємо, за допомогою потрібної прошивки мікроконтролер може змінювати час відкриття симістора, регулюючи вихідну потужність, що забезпечує плавність зміни освітлення.

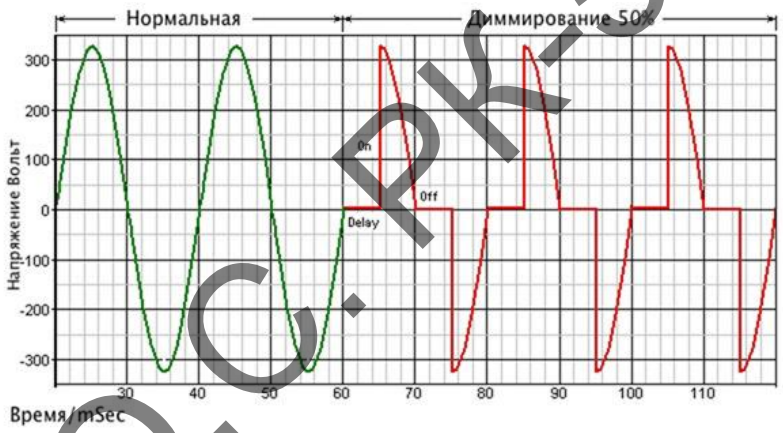

Рисунок 1.8 — Осцилограма синусоїди до та після димирування [6]

При цьому зміна часу відкриття симістору буде відповідати певному рівню освітлення, що і підтверджує розповсюдженість використання цього пристрою у системах освітлення.

1.3 Аналіз програмних рішень

Перейдемо до аналізу існуючих програмних рішень. Відповідно до завдань дипломного проекту існує потреба в розробці програмного забезпечення для роботи мікроконтролера, а саме для керування навантаженням при роботі із залізом, а також в розробці програмного забезпечення для смартфона

та комп'ютера, що надає звичайному користувачу інтуїтивно зрозумілий та зручний інтерфейс.

Насамперед переглянемо програмні рішення для роботи із периферією. Як виявилось, в мережі інтернет є доволі багато реалізацій димирування. Існує великий спектр бібліотек написаних для цих цілей, але в більшості вони немають хорошої оптимізації. Якісні бібліотеки, як правило, мають схожий, а інколи й однаковий функціонал.

Наприклад реалізація, наведена у наступному лістингу, написана стандартними методами [7]:

```
int dimpin = 4;                                    // вихід димеру на симістор
char dim = 50;            // початковий рівень димирування від 0 до 255

void setup() {              // функція ініціалізації основних налаштувань
pinMode(dimpin , OUTPUT);                  // ініціалізація виходу дімеру
attachInterrupt(0, light, FALLING);      // переривання для детектору
нуля
}
void light() {                          // функція керування яскравістю
  if (dim > 0 && dim < 255) {           // встановимо межі димирування
    delayMicroseconds(33*(255-dim));              // розрахунок кількості
мікросекунд для відкриття
    digitalWrite(dimpin , HIGH);                  // відкриття симістору
    delayMicroseconds(500);                                  // затримка
    digitalWrite(dimpin , LOW);                   // закриття симістору
  }
}
void loop() {                                     // тіло основного циклу
}
```

Код є досить простим і зрозумілим навіть для новачків у програмуванні, але проблема полягає в тому, що на практиці така реалізація є неефективною. Лампа при певних значеннях змінної dim починає неконтрольовано мигати, що унеможливлює використання цієї програми.

У наступному лістингу розглянемо реалізацію на базі бібліотеки “CyberLib” [8]:

```
#define dimPin 4        // об’ява змінної для керуючого виводу димеру
#define zeroPin 2       // об’ява змінної для декторного виводу димеру
#include <CyberLib.h>                          // підключення бібліотеки
volatile int tic, Dimmer;                      // об’ява змінних для роботи

void setup() {              // функція ініціалізації основних налаштувань
Serial.begin(9600);                  // задання швидкості передачі даних
pinMode(dimPin, OUTPUT);    // ініціалізація конфігурації керуючого виводу
pinMode(zeroPin, INPUT);    // ініціалізація конфігурації декторного виводу
attachInterrupt(0, detect_up, FALLING);    // налаштування спрацювання фукнції через низький рівень

StartTimer1(timer_interrupt, 40);              // час одного розряду ШІМ
StopTimer1();                                  // зупинка таймер
Serial.println("Start");                 //вивід додаткової інформації
}
void loop() {                                  // тіло основного циклу
Dimmer = map(analogRead(0), 0, 1023, 240, 0);      // зчитування з аналогово порту значення та перетворення у межі від 0 до 240
}
void timer_interrupt() {            // переривання таймеру спрацьовують кожні 40 секунд
tic++;                                         // інкремент лічильника
if (tic > Dimmer)                  // якщо настав час ввімкнути симістор
digitalWrite(dimPin, 1);                       // вмикаємо симістор
}
void  detect_up() {                // обробка переривання при переході через 0 “знизу”
tic = 0;                                       // обнуялємо лічильник
ResumeTimer1();                                // перезапускаємо таймер
```

```
attachInterrupt(0, detect_down, RISING);          // переналаштовуємо переривання
}
void  detect_down() {  // функція обробки переривання при переході синусоїди через 0 “зверху”
tic = 0;                                                  // обнуялємо лічильник
StopTimer1();                                             // зупиняємо таймер
digitalWrite(dimPin, 0);                                  // вимикаємо симістор
attachInterrupt(0, detect_up, FALLING);   // перемикаємо переривання
}
```

Хоча ця реалізація є важчою для розуміння за першу, її швидкодія набагато краща. Справа в тому, що бібліотека “CyberLib” використовує апаратний лічильник мікроконтролера, чим збільшує швидкість виконання команд майже в 2 рази. Мінусом цієї реалізації є те, що не всі мікроконтролери коректно працюють з таким лічильником, а користувачу не завжди наданий доступ до роботи з ним. У разі помилок, мікроконтролер потрібно перепрошивати, що може призвести до виведення із ладу самого мікроконтролера. Тому такий спосіб підходить, але він не є універсальним.

Далі продемонстровано лістинг на основі бібліотеки “AC_Dimmer” [9]:

```
#include <AC_Dimmer.h>                    // підключення бібліотеки
#define Dimmer_1 0              // обява змінної для виводу на димеру
void setup()                // функція ініціалізації основних налаштувань
{
Dimmer_init_begin();                              // ініціалізація димеру
Dimmer_pin_assign(Dimmer_1, 3);          // інціалізація портів димеру
Dimmer_init_end();                      // завершення ініціалізації димеру
}
int adc_read;       // змінна для збереження значень із потенціометру
void loop()                                       // тіло основного циклу
{
adc_read = analogRead(A4)/4;       // зчитуваня значення із виводу потенціометру та переведення у відповідний діапазон значень
```

```
    Dimm_value(Dimmer_1,  adc_read);          // встановлення значення
димирування відповідно значення потенціометру
    delay(50);                                                    // затримка
    }
```

Люди, які розробляли дану бібліотеку максимально спростили вигляд функцій. Зрозуміло, що ці функції місять в собі певний алгоритм, але кінцевий варіант інтуїтивно зрозумілий і не потребує особливих навичок у програмуванні. Проте програмісти відразу вказали, що вона не буде ідеально працювати із усіма димерами. Також вказано, що із платами їхнього виробництва ця бібліотека повністю сумісна і працює на всі 100%.

Отже, реалізацій керування периферією є дуже багато, а оптимальний вибір полягає лише в практичному тестуванні різних алгоритмів.

Перейдемо до існуючих реалізацій для розробки додатків на смартфон, комп’ютер та веб-сервер.

Якщо використовувати ESP8266, то є можливість запрограмувати його під різні режими роботи, як точка доступу та користувач. Існують і інші варіанти, проте в нашому випадку підійде варіант із підключенням у ролі користувача.

Наступний лістинг на основі бібліотеки “ESP8266WebServer” [10] демонструє можливість підключення ESP8266 до WiFi-мережі у режимі користувача:

```
    void setup() {              // функція ініціалізації основних налаштувань
    Serial.begin(115200);              // задання швидкості передачі даних
    Serial.print("Connecting to ");              // виводить у COM-port стан
підключення до мережі
    Serial.println(ssid);                    // виведення додаткових даних
    WiFi.hostname("brightness_control");              // задається hostname
для ESP8266
    WiFi.begin(ssid, password); // підключення до мережі під назвою із
змінної ssid, та паролем із змінної password
```

```
        while (WiFi.status() != WL_CONNECTED) {          // поки приєднується
до мережі
        delay(500);                                                  // затримка
        Serial.print(".");                              // виведення додаткових даних
         }
        Serial.println("");                             // виведення додаткових даних
        Serial.println("WiFi connected.");              // виведення додаткових даних
        Serial.println("IP address: ");                 // виведення додаткових даних
         Serial.println(WiFi.localIP());                // виведення додаткових даних
         server.begin();                                          // запуск веб-серверу
        }
```

Із наведеної частини коду зрозуміло, що кожного разу, коли живлення подається на мікроконтролер, він починає підвантажувати функцію setup(), у якій в свою чергу реалізується підключення до точки доступу WiFi і розгортання веб-серверу у локальній мережі.

Рядок: Serial.println(WiFi.localIP()); виводить локальну IP-адресу, яка була присвоєна ESP8266 у COM-port. COM-port – це послідовний інтерфейс передачі (адже інформація передається біт за бітом) даних. IP-адреса – це ідентифікатор, необхідний для адресації комп'ютерів чи пристроїв у мережах. IP-адреса повинна бути унікальною у випадку, якщо пристрій працює у глобальній мережі, або унікальною для локальної мережі. У локальних мережах перший октет IP-адреси – стандартизований, як 192 або 10. Зазвичай, WiFi роутери працюють із конфігурацією DHCP. Деякі користувачі можуть, при бажанні або у випадку потреб провайдера, працювати із статичною конфігурацією, але це не є зручно. DHCP (Dynamic Host Configuration Protocol) – мережевий протокол, який дозволяє пристроям автоматично отримувати IP-адресу.

При роботі із такою конфігурацією DHCP-сервер вирішуватиме які мережеві налаштування надати для клієнту. Отже, при перепідключенні пристрій часто буде отримувати інші налаштування, а відповідно і іншу IP-адресу. Зрозуміло, що у залежності від налаштувань на роутері, IP-адреса мі-

кроконтролера буде змінюватись. Отже, варто продумати механізм для знаходження мікроконтролера у локальній мережі для додатків на смартфон та комп'ютер.

На цьому пошук готових програмних рішень закінчується, адже залишається написати розмітку та логіку самої веб-сторінки, розмітки для додатку на смартфон та комп'ютер і відповідно логіки для цих додатків. Оскільки в цих випадках використовується авторський дизайн, то проблема у пошуку готових рішень у мережі інтернет відпадає. Для логіки готових рішень, як таких, немає, тому що розробники схожих додатків не викладають код у відкритий доступ, тому й інформації на таку тему дуже мало. У нагоді стає лише офіційна документація набору мов та вміння програмувати.

1.4 Аналіз технічного завдання

Згідно технічного завдання необхідно розробити пристрій та відповідне програмне забезпечення, що використовуватимуться для керування електричним навантаженням на прикладі розумної лампи.

Живлення димера – 220 В, а живлення мікроконтролера – 5 В. Для зменшення габаритів пристрою є сенс об'єднати живлення димера та мікроконтролера від мережі 220 В. Тоді потрібно передбачити перетворювач напруги із 220 В до приблизно 3,3 В. В такому випадку потрібно передбачити вивід проводу для живлення.

Корпус має вигляд прямокутного паралелепіпеда із виводом для проводів живлення та отвором для під'єднання лампи.

За показниками надійності пристрій повинен мати гарантійний термін не менше 2-ох років та середній час напрацювання має складати близько 15000 год. Ремонтом та технічним обслуговуванням займається виробник.

Корпус доцільно зробити розбірним, щоб у випадку поломок була забезпечена можливість проведення технічного обслуговування.

Пристрій призначений для використання у жилих приміщення без наявності агресивних середовищ та за стабільного температурного режиму.

В умовах експлуатації приладу С1 згідно з ГОСТ 16019-2001 [12] зазначено, що це стаціонарна апаратура, яка встановлюється в опалюваних наземних чи підземних спорудах. Умови експлуатації та їх основні характеристики подані в таблиці 1.1.

Таблиця 1.1 — Характеристики та значення механічних та кліматичних чинників

Чинник	Характеристика чинника	Значення чинника
Синусоїдальна вібрація	Діапазон частот, Гц	10-70
	Амплітуда прискорення, м/с^2 (g)	19,6 (2)
	Тривалість впливу, хв	90
Знижена температура	Робоча, °С	+5
	Гранична, °С	-40
Підвищена температура	Робоча, °С	+40
	Гранична, °С	+55
Знижений атмосферний тиск	Тиск, кПа	55

За такі вимоги відповідає УХЛ 4 за ГОСТ 15150-69 [11], тобто призначений для експлуатації у жилих приміщеннях з штучно регульованими кліматичними умовами. При такому виконанні робоча температура знаходиться в межах від +1°С до +35°С, а гранична — від +1°С до +45°С. Середнє значення відносної вологості повітря становить 65%, а граничне — близько 80%. Робоче значення атмосферного тиску складає 104 кПа, а мінімально допустиме – 84 кПа.

2 ОБГРУНТУВАННЯ ТА ВИБІР СХЕМОТЕХНІЧНОГО РІШЕННЯ

2.1 Розробка структурної схеми

Розглянемо структурну схему проекту, зображену на рис. 2.1, яка розроблена згідно із аналізом технічного завдання. Розглянувши схему, можна зробити висновок, що основним завданням є створення пристрою, який буде працювати із веб-сервером, який в свою чергу співпрацюватиме із програмним забезпечення на смартфонах та комп’ютерах.

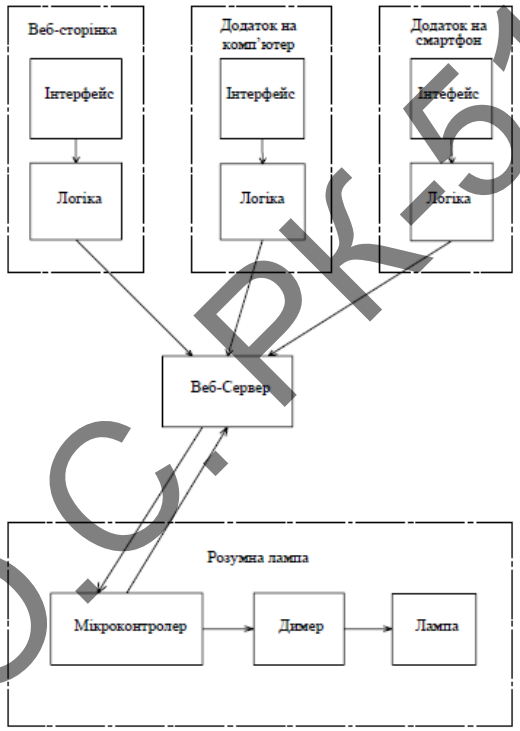

Рисунок 2.1 — Структурна схема проекту розумної лампи

Пристрій для керування електричним навантаженням повинен дистанційно сприймати керуючі сигнали і відповідно до них змінювати режим роботи навантаження. Як вже зазначалось у розділі про аналіз схемотехнічних рішень, ми можемо використати мікроконтролер ESP8266, який буде виконувати роль передавача та приймача.

Нам необхідний алгоритм зміни вихідного сигналу відповідно до прийнятих керуючих сигналів. Це можна зробити з допомогою ESP8266 при поєднанні безпровідної передачі керуючих сигналів та їх опрацювання з наступ-

ною зміною режиму роботи димера. Додатки на смартфон та комп'ютер повинні мати достатній функціонал для відправлення керуючих сигналів та забезпечувати простоту інтерфейсу керування для користувача.

2.2 Вибір плати з мікроконтролером

Як вже зазначалось в огляді існуючих схемотехнічних рішень, непоганим варіантом буде мікроконтролер ESP8266, адже в ньому вже вбудована антена. Отже, для проекту було обрано плату WeMos D1 mini [13], в основі якої – мікроконтролер ESP8266. Ця плата зображена на рис. 2.2.

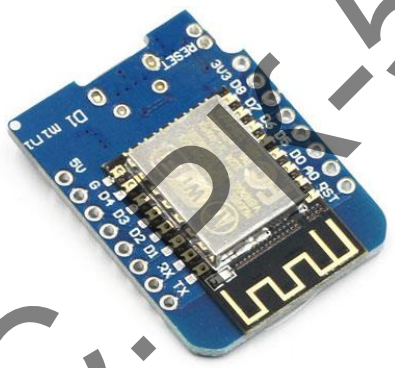

Рисунок 2.2 — Плата WeMos D1 mini [13]

Обрано саме плату, а не чистий мікроконтролер ESP8266, адже такий вибір полегшує роботу із платформою. У цю плату вбудовано конвертер CH340 USB–UART, що дозволяє передавати дані за допомогою COM-порту. Також наявний стабілізатор напруги, а контакти зручно розведені по роз'ємах.

Така плата підходить для використання у малогабаритних пристроях, на що вказує і назва версії – «mini», тобто плата має менші розміри. Справді, розміри плати становлять 39мм х 25мм х 5мм. Варто зазначити, що такі малі розміри впливають на кількість виводів. В цій версії аналоговий вивід лише

один і для нашого проекту це не настільки важливо, адже для керування димером потрібно лише 2 цифрових виводи.

Розглянемо основні характеристики, зазначені в таблиці 2.1.

Таблиця 2.1 — Параметри плати WeMos D1 mini

Параметр	**Значення**
Частота мікроконтролеру, МГц	80
Швидкість UART	115200
Робоча напруга, В	3,3
Напруга живлення, В	3,7 — 12
Максимальний струм споживання, А	240мА
Чіп флеш-пам’яті, МБ	4

Знову ж таки, конфігурація мікроконтролера залишається такою ж. А це означає, що ця плата підходить як і в якості мозку для керування периферією, так і для розгортування веб-сервера на її базі.

Є можливість використовувати програмне забезпечення Arduino IDE для прошивання цієї плати під наші потреби. Для цього потрібно встановити готові рішення для менеджеру плат, які надає виробник та вибрати потрібну нам плату у середовищі Arduino IDE.

2.3 Вибір димера

Як зрозуміло із аналізу технічного завдання та розгляду схемотехнічних рішень, для зміни вихідної потужності обрали димерування. Оскільки проект демонструє можливості пристрою дистанційного керування електричним навантаженням на прикладі розумної лампи, то немає потреби у виборі доволі потужного димера. Візьмемо до уваги той факт, що пристрій можна використовувати як для нагрівання води, так і для інших цілей. Таким чином, вибірка потужності зростає, адже для того ж комфортного нагрівання води потрібна потужність хоча б в 2 кВт. Отже, для нашого проекту було обрано одноканальний димер для Arduino компанії RobotDyn [14], зображений на рис. 2.3.

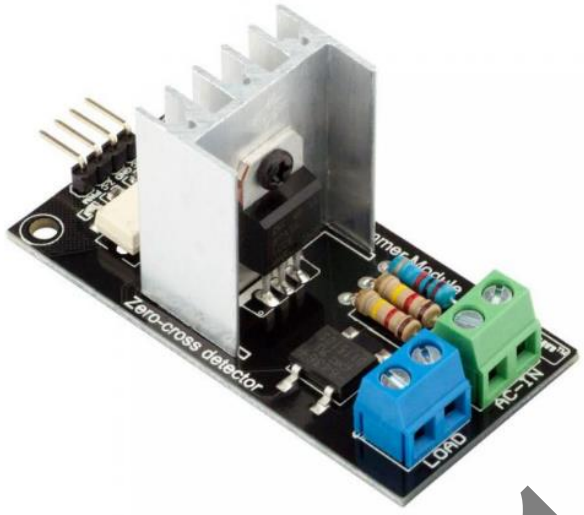

Рисунок 2.3 — Димер для Arduino RobotDyn [14]

У таблиці 2.2 наведено характеристики нашого вибору, що були вказані виробником.

Зрозуміло, що вихідної потужності такого димеру вистачить як для лампочки, так і для нагрівального елемента, адже при роботі із мережею 220 В та вихідним струмом до 16 А димер може працювати із потужністю близько 3 кВт на виході.

Таблиця 2.2 — Параметри димера для Arduino RobotDyn

Параметр	**Значення**
Частота змінного струму, Гц	50/60
Рівень логічних сигналів, В	3.3/5
Вихідний струм, А	16
Робоча температура, градуси	-20–80
Розміри, мм	63×30×30

Варто зазначити, що великим плюсом вибраної моделі є наявність вбудованого радіатору, що дозволяє не проводити розрахунки тепловіддачі для використання димера на великих потужностях.

Розглянутий димер сумісний із платами Arduino, адже в таблиці 2.2 вказуються значення рівня логічних сигналів 3,3 В або 5 В. Це означає, що окрім плат Arduino на базі мікроконтролерів Atmega328, димер сумісний із платами на базі мікроконтролеру ESP8266, а більшість мікроконтролерів і працюють на схожих рівнях логічних сигналів.

На сайті виробника вказана можлива схема підключення у випадку плати Arduino, яка зображена на рис. 2.4, та описано, які є логічні виводи і для чого вони потрібні.

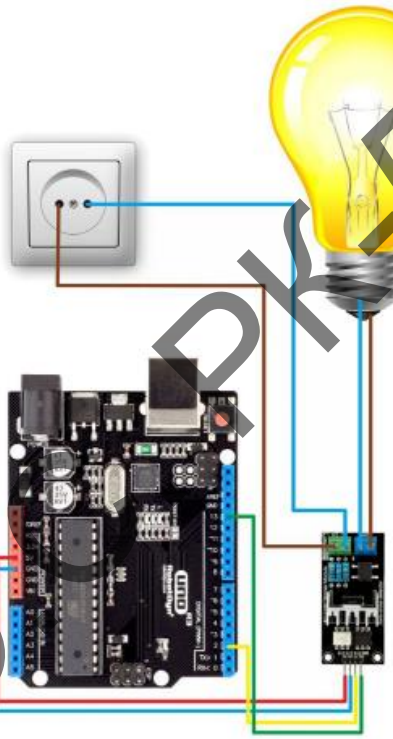

Рисунок 2.4 — Схема підключення димера [14]

Як видно із принципової схеми на рис. 2.5, димер має такі виводи керування, як VCC, GND, Z-C та PWM. Також варто зазначити, що схема дуже схожа на схеми, зо зображена на рис. 1.5.

Отже, живлення та земля відповідає контактам VCC та GND. Розглянемо виводи Z-C та PWM. Як зазначалось в аналізі схемотехнічних рішень, димер повинен містити схему “детектора нуля”. Вивід цієї схеми підписаний як Z-C (Z скорочення від ZERO).

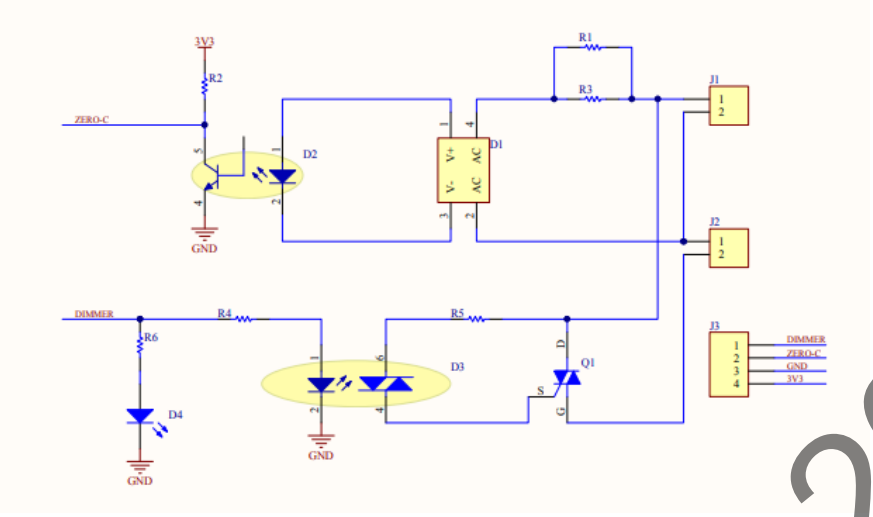

Рисунок 2.5 — Принципова схема [14]

Також повинен подаватись сигнал на керуючий контакт симістора, а для цього залишається вивід PWM.

2.4 Вибір перетворювача напруги

Оскільки наш пристрій повинен живитись тільки від мережі 220 В, то для того, щоб подати бажаний рівень напруги живлення на мікроконтролер потрібно використати перетворювач напруги, де вхідна напруга буде змінною, а вихідна – постійною. Отже є потреба в перетворювачі AC/DC. Ми обрали Hi-Link HLK-PM01 [15], зображений на рис. 2.6.

Рисунок 2.6 — Перетворювач напруги Hi-Link HLK-PM01 [15]

Його характеристики відображені у таблиці 2.3. Ця модель повністю сумісна по параметрам із платою WeMos D1 mini. Тобто напруга живлення на виході відповідає діапазону напруги живлення для нашої плати. Аналогічна ситуація із вихідним струмом.

Таблиця 2.3 — Параметри перетворювача напруги Hi-Link HLK-PM01

Параметр	**Значення**
Вхідна напруга, В	100-240(AC)
Вихідна напруга, В	5В(DC)
Вихідний струм, мА	600мА
Робоча температура, градуси	-20 – +60

Також Hi-Link HLK-PM01 має доволі компактні габарити, його розміри складають 30 мм х 15.4 мм х 20 мм.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір технологій та мов програмування для розробки

Насамперед потрібно визначитись із використанням потрібних нам технологій. Як вже зазначалось, є можливість розвернути веб-сервер на мікроконтролері. Ми можемо запрограмувати плату WeMos D1 mini, використовуючи платформу Arduino IDE (Integrated Development Environment). Це – інтегроване середовище розробки для плат Arduino та інших на мові C/C++. Мова програмування C — універсальна процедурна імперативна мова програмування, в основі якої лежить послідовна зміна стану програми за допомогою наказових функцій. В свою чергу C++ — це вже мова із підтримкою об’єктно-орієнтованої парадигми. В основі цієї мови лежить мова C. Такий підхід дозволяє полегшити роботу програміста. Отже, логіка та основа вебсерверу буде написана на мові C/C++ із використанням сторонніх бібліотек.

В свою чергу, потрібно відразу розвернути інтерфейс взаємодії із вебсервером – веб-сайт, який буде прописаний відразу в Arduino IDE. Для розмітки сторінки та підлаштування стилів використаємо HTML/CSS, а для логіки сторінки – JavaScript. Зазначені мови набули великої популярності і мають доволі хорошу підтримку зі сторони їх розробників.

HTML (Hyper Text Markup Language) — мова гіпертекстової розмітки сторінки, на якій пишуть «скелет» сторінки для більшості сайтів у мережі інтернет. Ця мова інтерпретується браузером, тобто переводиться у машинний код та кожен рядок коду виконується по черзі. Використана документація знаходиться на офіційному сайті розробника [16].

CSS (Cascading Style Sheets) — мова каскадних таблиць стилів. Вона поєднується із основною розміткою і вказує для неї стилі, як відступи, позиціонування відносно інших елементів, кольори, шрифти, тощо. Використана документація знаходиться на офіційному сайті розробника [17].

JavaScript — це мова, яка підтримує декілька парадигм програмування, як об’єктно-орієнтовну, функціональну та імперативну. Вона також інтерпретується браузером та часто використовується для логіки веб-сторінки. Це дозволяє виконувати певні функції та надає сайту інтерактивність. Використана документація знаходиться на офіційному сайті розробника [18].

Також варто визначитись із додатком на комп’ютер. Доволі зручним програмним забезпеченням для розробки на Windows є програма Visual Studio, тобто IDE, що містить багато технологій для написання програм на різні платформи на різних мовах програмування. Основні функції цієї програми є безкоштовними та оскільки вона розроблена компанією Microsoft, то ніяких проблем не виникне при розробці додатків на комп’ютер із системою Windows. В цьому напрямку можна використати технологію Windows Forms із підтримкою мови XAML для графічної розмітки сторінки та мови C# для логіки додатку.

XAML (eXtensible Application Markup Language) — це мова розширеної розмітки додатків. ЇЇ синтаксис віддалено нагадує HTML/CSS. Також ця мова є доволі інтуїтивно зрозумілою, що сильно спрощує роботу людям, які ніколи не писали на XAML. Використана документація знаходиться на офіційному сайті розробника [19].

C# — це об’єктно-орієнтована мова програмування, розроблена інженерами компанії Microsoft як мова для розробки веб-додатків. Синтаксис найбільше схожий на мову C++ та Java. Використана документація знаходиться на офіційному сайті розробника [20].

Для смартфону є сенс використати таке ж програмне забезпечення Visual Studio, адже у наявності є технологія Xamarin Forms, яка дозволяє доволі зручно розробляти додатки cross-platform (для декількох платформ одночасно), а саме для IOS та Android. При цьому Xamarin Forms також використовує XAML [21] та C# для розмітки та логіки, але уже із певними переробками з врахуванням особливостей смартфонів. Також варто зазначити, що

розробляти додаток відразу на дві платформи незручно і це потребує доволі хороших навичок програміста, адже деякі речі, як шляхи до створених файлів, працюють по-різному на різних платформах. Таким чином, для спрощення, додаток буде писатись тільки на платформу Android.

Для спілкування веб-серверу із веб-сайтом та веб-додатками доволі широко використовується стандарт HTTP (Hyper Text Transfer Protocol). HTTP відповідає за передачу даних у вигляді гіпертексту у мережі інтернет за допомогою спілкування “клієнт-сервер”. Така технологія надає можливість виконувати запити на сервер із додатків і отримувати відповіді від серверу із потрібними даними і навпаки. В нашому проекті ця технологія буде використовуватись для відправки керуючих команд на веб-сервер і на мікроконтролер.

3.2 Розробка веб-серверу та логіки WeMos D1 mini

Підключення до WiFi мережі та запуск веб-серверу можемо взяти із лістингу в пункті аналізу програмних рішень. Залишається ініціалізувати потрібні змінні для роботи програми та власне саму веб-сторінку.

Наступний лістинг демонструє ініціалізацію потрібних нам змінних та бібліотек:

```
#include <ESP8266WebServer.h>                 // підключення бібліотеки
#include <RBDdimmer.h>                        // підключення бібліотеки
#define USE_SERIAL  Serial                    // об’ява змінної для Serial
#define zerocross 4             // об’ява змінної для детекторного виводу
#define outputPin 5               // об’ява змінної для керуючого виводу

 dimmerLamp dimmer(outputPin, zerocross);   // об’ява змінної димеру
 char* ssid = "name";                       // об’ява змінної ssid WiFi мережі
 char* password = "password";    // об’ява змінної паролю WiFi мережі
 int brightness = 0;             // об’ява змінної для збереження значення
яскравості
```

```
int pos1 = 0;            // об’ява допоміжної змінної для розпізнавання значення яскравості
int pos2 = 0;            // об’ява допоміжної змінної для розпізнавання значення яскравості
String header;     // об’ява змінної для збереження заголовку запиту
WiFiServer server(80);                    // вказується порт для серверу
```

Тут використовується бібліотека для роботи димера “RBDdimmer.h” [22]. На сайті виробника димера для Arduino RobotDyn є посилання на їхню бібліотеку, яка сумісна з даним димером. Тому при поєднанні програми для керування димером та веб-сервером зміниться і функція setup(), як продемонстровано у наступному коді:

```
void setup() {            // функція ініціалізації основних налаштувань
Serial.begin(115200);              // задання швидкості передачі даних
dimmer.begin(NORMAL_MODE, ON);     //задання режиму димеру
dimmer.setPower(8);              //встановлення яскравості лампи на 0
Serial.print("Connecting to ");     // виведення додаткової інформації
Serial.println(ssid);              // виведення додаткової інформації
WiFi.hostname("brightness_control");              // задання hostname для ESP8266
WiFi.begin(ssid, password);              // підключення до WiFi мережі
while (WiFi.status() != WL_CONNECTED) {       // поки підключається до мережі
delay(500);                                                    // затримка
Serial.print(".");                    // виведення додаткової інформації
}
Serial.println("");                   // виведення додаткової інформації
Serial.println("WiFi connected.");   // виведення додаткової інформації
Serial.println("IP address: ");      // виведення додаткової інформації
Serial.println(WiFi.localIP());      // виведення додаткової інформації
server.begin();                           // розгортування веб-серверу
}
```

Відповідно до тестів, при встановленні димерування на 0, лампа починає мигати, а вже при значенні димирування 8, лампа не світиться взагалі. Отже це той рівень, який потрібний для вимкнення лампи.

Далі переходимо до основної функції loop(), у якій знаходиться основна логіка програми. Згідно із запитами, які надійшли на веб-сервер, потрібно змінювати режими роботи димера і яскравість лампи.

```
void loop(){                                        // тіло основного циклу
WiFiClient client = server.available();             // об’ява змінної клієнту
if (client) {                                       // при підключенні нового клієнту
Serial.println("New Client.");                      // відбувається виведення додаткової інформації
String currentLine = "";                            // об’ява допоміжної змінної, щоб не виникало колізій
while (client.connected()) {                        // поки клієнт підключений
if (client.available()) {                           // якщо клієнт доступний
char c = client.read();                             // прослуховувати запити від клієнту
Serial.write(c);                                    // виведення інформації про надіслані запити
header += c;                                        // записувати запит у заголовок
if (c == '\n') {                                    // якщо надіслано запит
if (currentLine.length() == 0) {                    // якщо в даній ітерації ще не поступав запит, починається відповідь на нього
client.println("HTTP/1.1 200 OK"); // виведення додаткової інформації
client.println("Content-type:text/html");           // виведення додаткової інформації
client.println("Connection: close"); // виведення додаткової інформації
client.println();                                   // виведення додаткової інформації

if(header.indexOf("GET /find_esp")>=0) {  // відповідь на запит пошуку
client.stop();}                                     // зупинка цієї ітерації
client.println(webpage);       // надання доступу до веб-сторінки(сайту)
if(header.indexOf("GET /LEDon")>=0) {   // відповідь на запит ввімк-нення лампи
for (int i = 8; i < 51; i++)          // цикл для плавного ввімкнення
```

```
{
dimmer.setPower(i);                          // задання сили димирування
delay(10);                                   // затримка
}}
if(header.indexOf("GET /LEDoff")>=0) {       // відповідь на запит вимкнення
dimmer.setPower(8);  // встановлення відповідної сили димирування
}
if(header.indexOf("GET /?Brightness=")>=0) { // відповідь на запит зміни яскравості
pos1 = header.indexOf('=');          // використання допоміжної змінної
pos2 = header.indexOf('&');          // використання допоміжної змінної
brightness = header.substring(pos1+1, pos2).toInt(); // пошук змінної яскравості у рядку запиту
dimmer.setPower(map(brightness,0, 100, 8, 100));     // встановлення відповідної сили димирування із підлаштуванням діапазону значень
}
if(header.indexOf("GET /Test")>=0) {         // відповідь на запит тесту
for (int i = 15; i < 36; i++)                // цикл для плавного ввімкнення
{
dimmer.setPower(i);     // встановлення відповідної сили димрування
delay(100); // затримка
}
for (int i = 35; i>16 ; i--)                 // цикл для плавного вимкнення
{
dimmer.setPower(i);     // встановлення відповідної сили димрування
delay(100);                                  // затримка
}
}
client.println();                            // виведення допоміжної інформації
break;                                       // переривання ітерації
} else {                                     // у разі виникнення колізії
currentLine = "";                            // обнулення допоміжної змінної
}
```

```
} else if (c != '\r') {        // якщо запит не містить спеціальних символів перед переносом строки
currentLine += c;        // записуємо запити клієнту в допоміжну змінну
}}}
header = "";                                    // обнулення заголовку запиту
client.stop();                                  // закінчення прослуховування
Serial.println("Client disconnected.");          // виведення додаткової інформації
Serial.println("");                             // виведення додаткової інформації
}}
```

Наведений лістинг демонструє основний алгоритм. Програма прослуховує запити, надіслані від користувачів. Після знаходження запиту вона виконує закладену функцію, а потім переходить на нову ітерацію прослуховування.

Запит на зміну яскравості відправляється у форматі /?Brightness=x&, а програма перебирає рядок і знаходить відповідне значення яскравості.

3.3 Розробка веб-сторінки

У лістингу веб-серверу використовується змінна webpage, яка надає доступ до веб-сторінки. Змінна webpage – це HTML/CSS, JavaScript, тобто код сайту. Розглянемо продемонстрований код:

```
char webpage[] = R"=(       // заносимо код сторінки у масив символів, розбиваючи строку на символи
<!DOCTYPE html>          // повідомляємо браузеру, що це код HTML
<html>                                          // відкриваємо тег для HTML
<head>                                          // відкриваємо головний тег
<meta charset="UTF-8" name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">     // в цьому тегові вказують розтягування по ширині і неможливість маштабування для коректного відображення на смартфонах
<title>Brightness Control</title>                // вказується назва вкладки із сайтом
```

```
<style>                                    // відкриття тегу зі стилями CSS
* {                                        // стилі для всієї сторінки
margin: 0px;                               // обнулення зовнішнього відступу
padding: 0px;                              // обнулення внутрішнього відступу
}
body {                                     // стилі для основного тіла сторінки
background-color: rgb(27, 41, 48);         // вказується фон сторінки
}
main {                                     // стилі для основного поля навігації
position: absolute;      // позиціонування відносно краю вікна браузеру
top:50%;                   // розташування на половині сторінки зверху
left: 50%;                 // розташування на половині сторінки знизу
transform: translate(-50%, -50%);          // позиціонування по середині
з урахуванням розмірів
width: 350px;                              // задання ширини
height: 350px;                             // задання висоти
border-radius: 10px;                       // округлення країв поля
background: rgb(240, 240, 240);            // встановлення фону
}
slider_bright {          // вказуємо стилі для основного поля слайдеру
margin: 125px 25px 0px 25px;     //  встановлення зовнішніх відступів
width: 300px;                              // задання ширини
height: 10px;                              // задання висоти
background: grey;     // встановлення фону основного поля слайдеру
border-radius: 20px;                       // округлення країв
}
#range {                  // вказумо стилі для робочого поля слайдеру
display: block;                            // відображення блочне
visibility: hidden;                        // не відображати на сторінці
-webkit-appearance: none;        // відключає відображення на основі
теми на комп'ютері для браузерів Chrome/Safari
appearance: none;                // відключає відображення на основі
теми на комп'ютері
width: 100%;                               // задання ширини
```

```
        height: 10px;                                   // задання висоти
        background-color: rgb(27, 41, 48);              // задання фону
        border-radius: 20px;                            // округлення країв
        outline: none;                                  // не відображає зовнішніх країв
        }
        #range::-webkit-slider-thumb {      // вказумо стилі повзунка слайдеру
для бразерів Chrome/Safari
        -webkit-appearance: none;           // відключає відображення на основі
теми на комп’ютері для браузерів Chrome/Safari
        appearance: none;                   // відключає відображення на основі
теми на комп’ютері для браузерів
        width: 25px;                                    // задання ширини
        height: 25px;                                   // задання висоти
        border-radius: 13px;                            // округлення країв
        background: rgb(250, 208, 0);                   // задання фону
        cursor: pointer;                                // вказується стиль курсору
        outline: none;                                  // вимикаємо зовнішні краї
        }
        #range::-moz-range-thumb {      // вказумо стилі повзунка слайдеру
для бразеру FireFox
        width: 25px;                                    // задаємо ширину
        height: 25px;                                   // задаємо висоту
        border-radius: 13px;                            // округлення країв
        background: rgb(250, 208, 0);                   // задання фону
        outline: none;                                  // вимикаємо зовнішні краї
        cursor: pointer;                                // вказується стиль курсору
        }
        #range::-webkit-slider-thumb:hover{         // стилі для повзунка при
наведенні на слайдер для Chrome/Safari
         width: 30px;                                   // задаємо ширину
         height: 30px;                                  // задаємо висоту
         border-radius:16px;                            // округлюємо краї
         }
         #demo {        // стиль для відображення поточного рівня яскравості
```

```
        visibility: hidden;                    // вимикаємо видимість елементу
        margin: 20px 105px 0px 105px;          // задаємо зовнішні відступи
        width: 140px;                          // задаємо ширину
        height:  35px;                         // задаємо висоту
        text-align: center;          // задаємо позиціонування тексту по центру
        }
        #btn, #btn_test_start{          // основні стилі для відображення клавіш
        -webkit-appearance: none;         // відключає відображення на основі
теми на комп’ютері для браузерів Chrome/Safari
        appearance: none;                 // відключає відображення на основі
теми на комп’ютері для браузерів
        width: 100px;                          // задаємо ширину
        height: 40px;                          // задаємо висоту
        cursor: pointer;                       // задаємо стиль курсору
        border: 1px solid rgb(27,41,48);       // задаємо параметри країв
        background: rgb(27, 41, 48);           // задаємо колір фону
        color: rgb(240,240,240);               // задаємо колір тексту
        }
        #btn {                          // відступи клавіши ввімкення/вимкненя
        margin: 20px 125px 20px 125px;         // задаємо зовнішні відступи
        }
        #btn_test_start {                      // відступ для клавіши тесту
        margin: 20px 125px 20px 125px;         // задаємо зовнішні відступи
        }
        #btn:hover, #btn_test_start:hover {  // стилі при наведення на клавіши
        color: rgb(27, 41, 48);                // задаємо колір тексту
        background: rgb(240,240,240);          // задаємо колір фону
        border: 1px solid rgb(27,41,48);       // задаємо параметри країв
        }
        </style>                               // закриття прописання стилів
       </head>                                 // закриття головного тегу
       <body>          // відкриття тегу, який відповідає за розмітку видимої
частини веб сторінки
```

```
<div class="main">              // основний контейнер, що містить в собі панель управління
<div class="slider_bright">             // контейнер, що містить слайдер
<input type="range" min=0 max=100 value=50 class="slider" id="range" >                                                  // слайдер
</div>
<p id="demo">Brightness: 50%</p>       // Поточний рівень яскравості
<button id="btn" onclick="turn_on_off(this)" value="1">Turn on led</button>                                     // клавіша, ввімкнення/вимкнення
<button id="btn_test_start" oclick="test_button_start()">Test</button>
                                                                    // клавіша тесту
</div>
<script>                        // відкриття тегу, який відповідає за логіку сторінки написину на JavaScript
var xhr = new XMLHttpRequest();                 // створення запиту, отримуємо доступ до потрібних нам елементів сторінки
var slider = document.getElementById("range");
var output = document.getElementById("demo");
var btn = document.getElementById("btn");
slider.oninput = function() {           // функція зміни значення слайдеру
xhr.open("GET", "/?Brightness=" + slider.value + "&" , true);
  // створення запиту для зміни яскравості із відповідним значенням
xhr.timeout = 700;              // максимальний час очікування відповіді
xhr.ontimeout = function(){                     // при вичерпанні часу
xhr.abort();                                    // запит скидується
}
xhr.send();                                     // запит надсилається
output.innerHTML = "Brightness: " + slider.value + "%";          // зміна текстового значення у відповідному полі
}
function turn_on_off(elem) {                    // сценарій при натисканні на клавішу ввімкнення/вимкнення
                        // отримується доступ до потрібних елементів сторінки
var slider = document.getElementById("range");
```

```
        var output = document.getElementById("demo");
        if ( elem.value == "1") {                    // перевірка значення клавіши,
якщо один то лампа зараз вимкнена
        xhr.open("GET", "/LEDon", true);//відкрити запит на ввімкнення лампи
        xhr.send();                                                  // надіслати запит
        elem.value="2";                                  //змінити статус клавіши
        elem.textContent = "Turn off led";               //змінити текст клавіши
        slider.style.visibility = "visible";             // зробити видимим слайдер
        output.style.visibility = "visible";                  // зробити видимим
поле із значенням
        }
        else {                                           // якщо лампа ввімкнена
        xhr.open("GET", "/LEDoff", true);        // відкрити запит на вимкнення
        xhr.send();                                          // надіслати запит
        elem.value="1";                                  // змінити статус клавіши
        elem.textContent = "Turn on led";                // змінити текст клавіши
        slider.style.visibility = "hidden";      // зробити невидимим слайдер
        output.style.visibility = "hidden";                   // зробити невидимим
поле із значенням
        slider.value = "50";                  // встановлення значення слайдеру
        output.innerHTML = "Brightness: 50%";                 // зміна тексту
поля із значенням
        }
        }
        function test_button_start (){                  // сценарій при натисканні
на клавішу тесту
        xhr.open("GET", "/Test_start", true);            // відкриття запиту тесту
        xhr.send();                                       // відправлення запиту
        }
        </script>                                        // закриття тегу з логікою
        </body>                          // закриття тегу видимої частини сторінки
        </html>                                      // закриття тегу із тілом HTML
        )=";
```

Цей код реалізує веб-сторінку, до якої сервер надає доступ для керування яскравістю із браузеру. Неважливо, чи користувач використовує смартфон чи комп'ютер. Тег <meta> із атрибутами, які прописані в цьому лістингу, буде розтягувати основне тіло веб-сторінки згідно розміру екрану смартфона, а на комп'ютері інтерфейс залишиться сталим. На рис. 3.1 зображено інтерфейс веб-сторінки.

Варто зазначити особливість вставки JavaScript у цей код. Як видно із коментарів у лістингу, посилання на окремий файл CSS або відразу на прописані CSS стилі вказувались в головному тезі сторінки. Очевидно, що JavaScript також мав би бути прописаним у головному тезі сторінки, але тут проявляється особливість JavaScript. Справа в тому, що для виконання сценарію JavaScript повинен бути інтерпретатор (програма, що переводить сценарій у машинний код і виконує скрипт). На відміну від компілятора, який аналізує та переводить у машинний код відразу всю програму, інтерпретатор виконує код по черзі, рядок за рядком.

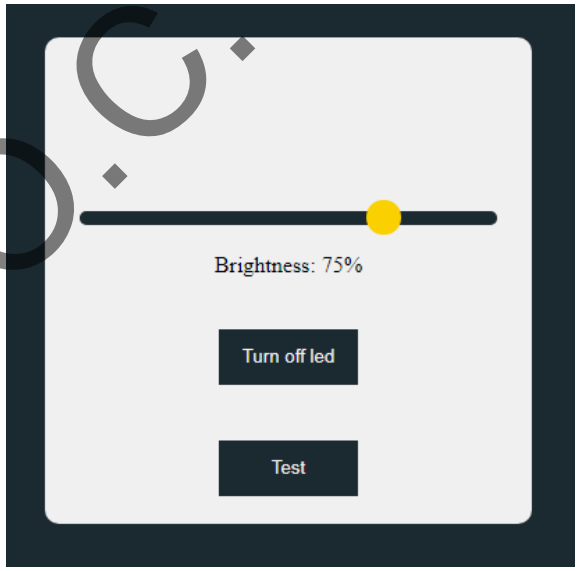

Рисунок 3.1 — Інтерфейс веб-сторінки

Якщо на момент виконання сценарію алгоритм не зможе отримати доступ до потрібних елементів на сторінці, логіка веб-сторінки працювати не буде. Отже, для коректної роботи завжди потребується прописувати сценарій JavaScript після розмітки сторінки.

Також варто зазначити, що для доступу до веб-сторінки потрібно знати IP-адресу веб-сервера. Таким чином зручне використання веб-сторінки можливе лише за умови, що користувач знає, що WiFi лампа не змінює свою IP-адресу. Таке може бути при певних налаштуваннях DHCP-сервера або ж при статичній конфігурації.

3.4 Розробка додатку на комп'ютер

Якщо розібрати алгоритм для програми на комп'ютер, то для початку потрібно, щоб додаток "знайшов" веб-сервер, тобто його IP-адресу. Для цього потрібно розробити механізм, який буде якомога швидше вести пошук по локальній мережі. Наступний лістинг демонструє цей алгоритм:

```
private async void Connect_Button(object sender, RoutedEventArgs e)
{                       // при натисканні виконується асинхронна функція
if (!connected)                         // якщо користувач ще не приєднаний
{
Await                                   btn.Dispatcher.BeginInvoke(new
updateDelegate(updateButton_start), "Connecting");// зміна статусу клавіши
var host = Dns.GetHostEntry(Dns.GetHostName());          // отримати
інформацію про хостів в мережі
foreach (var ip in host.AddressList)      // перелічення IP кожного хосту
{
if (ip.AddressFamily == AddressFamily.InterNetwork)          // якщо IP
відповідає формату IPv4(xxx.xxx.xxx.xxx)
{
if (ip.ToString().Contains("192.168."))         //якщо IP містить "192.168"
{
localip = true;                                  // допоміжне значення
for (int i = 0; i <= 255; i++)    // перебір останнього октету локальної IP
 {
Ping p = new Ping();                             // відкриття нового пінгу
```

```
        var task = PingAndUpdateAsync(p, ip.ToString().Substring(0, ip.ToString().LastIndexOf(".")) + "." + i.ToString());  //виклик функції пінгування з передачею IP
        }
        await btn.Dispatcher.BeginInvoke(new updateDelegate(updateButton_start), "Connect");  // зміна статусу клавіші
        } } }
        if (localip == false)  // якщо комп'ютер не підключений до WiFi
        {
        await btn.Dispatcher.BeginInvoke(new updateDelegate(updateButton_start), "Connect");  // зміна статусу клавіши
        MessageBox.Show("Please, connect to wifi!"); // повідомлення про помилку
        } } }
                    // функція асинхронного пінгування локальних ip-адрес
        private async Task PingAndUpdateAsync(Ping ping, string ip)
        {
       var reply = await ping.SendPingAsync(ip, timeout_p);  // асинхронне пінгування
        if (reply.Status == IPStatus.Success)  // якщо пінгування успішне
        {
        await Task.Run(() => Request(reply)); // викликається функція запиту
        } }
        private void Request(PingReply reply)  // функція запиту
        {
        Try  // механізм відлову помилок
        {
        var mYrequest = (HttpWebRequest)WebRequest.Create("http://" + reply.Address.ToString() + "/find_esp");  // надсилання пошукового запиту
        mYrequest.Timeout = timeout_r;  // час відклику
        HttpWebResponse mYresponse = (HttpWebResponse)mYrequest.GetResponse();  // прослуховування відповіді
```

```
        if ((mYresponse.StatusCode == HttpStatusCode.OK))            // якщо
відповідь код 200(успіх)
        {
        esp_ip = reply.Address.ToString();              // записуємо у змінну IP
        connected = true;              // вказуємо, що користувач приєднався
        btn.Dispatcher.BeginInvoke(new  updateDelegate(updateButton_end),
"Connected");                                   //оновлення статусу клавіши
        }
        mYresponse.Close();                  // закриваємо прослуховування
        }
        catch { }                          // у разі помилки, програма просто
переходить далі, а не вимикається
        }
```

У лістингу виникла потреба звернення до асинхронних функцій. Функціонал асинхронних функцій забезпечується ключовими словами – «async» та «await», або ж використанням уже готових асинхронних функцій.

Варто зауважити, що ефект зависання програми може створити ситуація, коли після натискання клавіша забирає керування на певний час у разі довгих операцій всередині програми. Може скластись враження, наче програма просто видасть помилку, що викликає недовіру до програми і вона не задовольнятиме наші потреби.

Власне асинхронні функції можуть виконувати потрібні задачі паралельно при цьому повертати контроль в основний потік. Тобто, якщо операція займає довгий час, то вона виконується паралельним потоком і дає можливість користуватись в цей час основним потоком. Такий підхід дозволяє залишати доступ до керування програмою користувачу, не завершуючи виконання необхідні операції. Відповідно, це не створює ілюзії зависання програми.

В цьому випадку асинхронні функції були використані також і для паралельного процесу пінгування та відправлення запитів, що значно скорочує час відклику.

Ping — службова утиліта призначена для перевірки з'єднань в мережах. При перевірці посилається запит зазначеному вузлу мережі та фіксується час від відправлення пакету до відповіді. Це дозволяє визначати двосторонні затримки, завантаженість мережі та чи існує пристрій за вказаною IP-адресою.

Наприклад, в нашій програмі останньому октету IP-адреси веб-сервера буде відповідати число 233, що означатиме майже повне проходження циклу пінгування, а відповідно і посилання запитів. За даними, час відклику пінгування однієї IP-адреси займає до 100 мс, а час відклику HTTP-запиту — 700 мс. Якщо підрахувати послідовне виконання такої роботи за умов, що до мережі більше ніхто не під'єднаний, то час відклику програми становитиме 25 с. Отже, на цей період робоча область просто зависне. Зрозуміло, що при наявності додаткових пристроїв у мережі час суттєво збільшиться. В такому випадку жоден користувач не буде використовувати програму із алгоритмом, на виконання якого підуть хвилини. При використанні асинхронних функцій час виконання пошуку може скоротитись до 2 с при наявності підключених девайсів до WiFi-мережі.

Якщо виникне помилка і цей алгоритм буде затрачати більше часу на виконання пошуку, то програма не зависне, а у користувача залишиться можливість натискати на інші клавіші та виконувати інші дії на робочій області.

Варто зазначити помилки, які виникають в результаті виконання асинхронних функцій. Вони виконуються в паралельному потоці, а відповідно не мають доступу до потоку керування графічним інтерфейсом. Наприклад, при закінченні сканування потрібно замінити статус клавіші. В такому випадку приходиться переходити до Dispathcer і вказувати виконання задачі в основному потоці.

Після, коли відома IP-адреса веб-серверу, можна спокійно надсилати запити ввімкнення та вимкнення лампи.

```
private void Led_on(object sender, RoutedEventArgs e)     //функція ввімкнення/вимкнення
{
```

```
        if (esp_ip.Contains("192.168."))      // якщо користувач не підключений
        {
        if (value == 1)                       // якщо клавіша у режимі вимкнення
        {
        Try                                   // механізм відлову помилок
        {
        var mYrequest = (HttpWebRequest)WebRequest.Create("http://" +
esp_ip + "/LEDon");        // надсилає на відому ip-адресу запит ввімкнення
        mYrequest.Timeout = timeout_r;                    // час відклику
        HttpWebResponse mYresponse =
(HttpWebResponse)mYrequest.GetResponse();          //прослуховування
відповіді
        mYresponse.Close();                   // закриття прослуховування
        BitmapImage image = new BitmapImage(new Uri("light_on.jpg",
UriKind.Relative));                                // пошук картинки
        light.Source = image;                 // зміна картинки клавіши
        slider1.Value = 50;                   // зміна значення слайдеру
        slider1.Visibility = Visibility.Visible;  // зміна видимості слайдеру
        value = 2;                            // зміна стану на вимкнення
        }
        Catch  {}                  // при помилці програма працюватиме далі
        }
        else                                  // якщо клавіша в режимі вимкнення
        {
        Try                                   // механізм відлову помилок
        {
        var mYrequest = (HttpWebRequest)WebRequest.Create("http://" +
esp_ip + "/LEDoff");                  // відправлення запиту вимкнення
        mYrequest.Timeout = timeout_r;                    // час відклику
        HttpWebResponse mYresponse =
(HttpWebResponse)mYrequest.GetResponse();  //початок прослуховуваня
        mYresponse.Close();                   // закриття прослуховування
        BitmapImage image = new BitmapImage(new Uri("light_off.jpg",
UriKind.Relative));                                // пошук картинки
```

```
light.Source = image;                          // зміна картинки клавіши
slider1.Visibility = Visibility.Hidden;        // зміна видимості слайдеру
value = 1;                                     // зміна стану на ввімкнення
}
catch { }                  // при помилці програма працюватиме далі
}}
Else                                           // якщо не підключено
{                                              // повідомлення про помилку
MessageBox.Show("Please, connect to wifi and your smart lamp!");
}}
private void Slider_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)      // функція зміни
значення слайдеру
{
if (esp_ip.Contains("192.168."))               // якщо підключено
{
Try                                            // механізм відлову помилок
{
 ((Slider)sender).SelectionEnd = e.NewValue;   //отримуємо нове
значення слайдеру
var mYrequest = (HttpWebRequest)WebRequest.Create("http://" +
esp_ip + "/?Brightness=" + e.NewValue + "&"); // надсилаємо запит із но-
вим значенням
mYrequest.Timeout = timeout_r;                 // час відклику
HttpWebResponse mYresponse =
(HttpWebResponse)mYrequest.GetResponse();// початок прослуховування
mYresponse.Close();                            // закриття прослуховування
}
catch { }                  // при помилці програма працюватиме далі
}}

private void Copy(object sender, RoutedEventArgs e) // функція, для
копіювання IP-адреси
{
```

```
if (esp_ip.Contains("192.168."))                    // якщо підключено
{
Clipboard.Clear();                                  // стираємо залишки в буфері
Clipboard.SetText(Result.Content.ToString());  // записуємо IP в буфер
}
Else                                                // якщо не підключено
{                                                   // повідомлення про помилку
MessageBox.Show("Please, connect to wifi and your smart lamp!");
}}
```

Розглянувши лістинг, варто зазначити, що функції виконуються звичайним методом, адже в асинхронності тут немає потреби. Такі операції не потребують значного часу для виконання, а отже не блокують інтерфейс і забезпечують нормальне функціонування програми. Зауважимо, що при виконанні запиту більше ніж за 700 мс, програма просто відкине такий запит і припинить його виконання, що покращує відклик програми.

На рис. 3.2 зображено інтерфейс програми для комп'ютера.

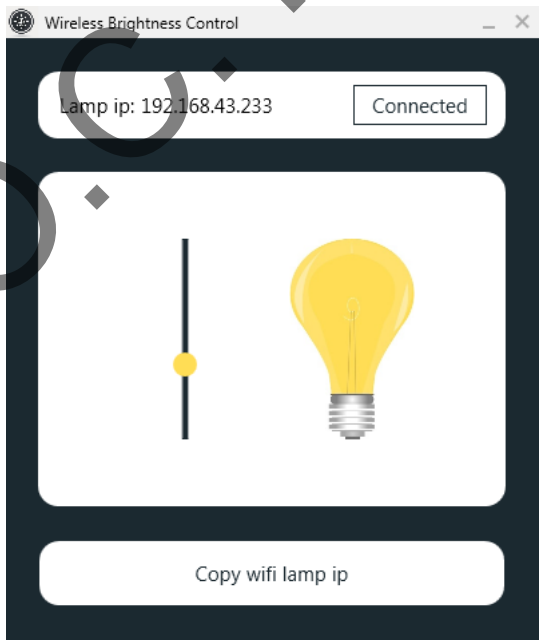

Рисунок 3.2 — Інтерфейс веб-додатку для комп'ютера

Наступний лістинг забезпечує відповідну розмітку:

```
<Grid>                                              // сітка для побудови
<Canvas Background="#1B2930">                       // основний контейнер
```

```
<WrapPanel Background="#F1F1F1" VerticalAlignment="Top" Height="25" Width="400">                    // контейнер для верхньої панелі
<WrapPanel Height="25" Width="25">    // контейнер для зображення
<Image Source="105964493-light-lamp-sign-icon-bulb-with-gears-symbol.jpg" />                    // зображення
</WrapPanel>
<Label Height="25" Width="325" Content="Wireless Brightness Control"/>                    // Мітка з назвою програми
<WrapPanel HorizontalAlignment="Left" Height="25" Width="50" WindowChrome.IsHitTestVisibleInChrome="True">    // панель для клавіш
<Button Name="MinimizeBtn" Style="{StaticResource MinimizedBtn}" Height="25" Width="25" Background="#F1F1F1"/>    // клавіша звертання
<Button Name="CloseBtn" Style="{StaticResource CloseBtn}" Height="25" Width="25" Background="#F1F1F1"/>    // клавіша закриття
</WrapPanel>
</WrapPanel>
        // панель, яка містить основні елементи керування програми
<StackPanel Height="400" Width="350" Margin="25,50,25,0">
<Border BorderThickness="0" CornerRadius="15" Background="#FFFFFF" Height="50">                    // округлення граней
<WrapPanel>                    // панель підключення та інформації
                    // панель, що містить інформацію про IP
<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>
                    // мітка, що імітує роботу клавіши підключення
<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">                                  // округлення граней
        <WrapPanel>              // панель для елементів керування лампою
        <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"/>                              // слайдер та його стилі
                    // панель, що імітує роботу клавіши ввімкнення/вимкненя
        <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"/>                                       // зображення лампи
        </StackPanel>
        </WrapPanel>
        </Border>

        <Border Name="copy" BorderThickness="1" CornerRadius="15"
Background="#FFFFFF" Height="50" Margin="0,25,0,0"> // округляємо грані
                                // мітка, що імітує роботу клавіши копіювання
        <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>
```

Також окрім цього коду були ще підключені певні стилі для слайдеру та клавіш. Зрозуміло, що вікно поділене на основні частини, які містять округлені межі та відповідні мітки для тексту та мітки, які імітують роботу клавіш.

3.5 Розробка додатку для смартфона

Оскільки для розробки додатку на смартфон було обрано Xamarin Forms, то код алгоритму не сильно змінився. В наступному лістингу можемо побачити та проаналізувати основні зміни в коді відносно версії для комп'ютера:

```
private async void  Connect_Button(object sender, EventArgs e)
{                                                    // функція приєднання
if (!connected)                                      // якщо вже приєднаний
{                                                    // зміна статусу клавіши
Device.BeginInvokeOnMainThread(updateButton_connecting);
var host = Dns.GetHostEntry(Dns.GetHostName());      // отримуємо
список хостів в мережі
foreach (var ip in host.AddressList)  // перебираємо IP із списку хостів
{
if (ip.AddressFamily == AddressFamily.InterNetwork)          // якщо IP
{                              // відповідає формату IPv4(xxx.xxx.xxx.xxx)
if (ip.ToString().Contains("192.168."))             // Якщо містить 192.168.
{
localip = true;                          // допоміжна змінна, для перевірки
}}}                                                  // підключення до WiFi
if (localip == false)                               // якщо не підключено WiFi
{       // зміна статусу клавіши, та вивід повідомлення про помилку
Device.BeginInvokeOnMainThread(updateButton_connect);
await DisplayAlert("Error", "Please, connect to wifi!", "OK!");
}
if(await
DependencyService.Get<IFileWorker>().FileNotEmptyAsync(filename)     &&
(localip == true)) // перевірка, чи існує і не пустий файл для запису IP
{
Try                                                 // механізм відлову помилок
```

```
        {                                                   // дістається IPз файлу
        string check = await
DependencyService.Get<IFileWorker>().LoadTextAsync(filename);
        var myrequest = (HttpWebRequest)WebRequest.Create("http://" +
check + "/find_esp");                                       // відсилається запит пошуку
        myrequest.Timeout = 700;                            // час відклику
        HttpWebResponse myresponse =
(HttpWebResponse)myrequest.GetResponse();// прослуховування відповіді
        if ((myresponse.StatusCode == HttpStatusCode.OK))   // якщо код
        {                                                   // відповіді 200(успіх)
        esp_ip = check;                                     // записуємо IP для роботи додатку
        connected = true;                                   // змінюємо стан на приєднаний
        Device.BeginInvokeOnMainThread(updateButton_end);   // зміна
        }                                                   // статусу клавіші
        myresponse.Close();                                 // закриття прослуховування
        }
        Catch {                                             // при помилці програма працюватиме далі
  await DependencyService.Get<IFileWorker>().SaveTextAsync(filename, "");
        } }                                                 // а також виконає стирання існуючої інформації в файлі
        if (!(await
DependencyService.Get<IFileWorker>().FileNotEmptyAsync(filename)) &&
(localip == true))                                          // якщо файл не існує, або пустий
        {
        for (int i = 0; i <= 255; i++)                      // перевірка IP локальної мережі
        {
        Ping p = new Ping();                                // відкриття пінгу
        Task task = PingAndUpdateAsync(p, ip.ToString().Substring(0,
  ip.ToString().LastIndexOf(".")) + "." + i.ToString());  // асинхронне пінгування
        } } } }
```

Як бачимо, головний алгоритм пошуку не дуже змінився, але додалась можливість роботи із файлами. Справа в тому, що Xamarin Forms виявився не настільки продуманим, як і Windows Forms. Звісно, можливий варіант нестачі потужності смартфону, але при тестах зрозуміло, що такий алгоритм не пот-

ребує великої кількості ресурсів. При детальному розгляді виконання асинхронних функцій виявилось, що при зміні платформи на Android, програма почала працювати набагато повільніше. Виявилось, що Xamarin Forms погано працює із асинхронними функціями. Тому було прийнято рішення вдосконалити цей алгоритм. З таким алгоритмом, при першому підключенні до веб-сервера час підключення складає близько 15 – 20 с. Звісно, це відносно довго, тому IP-адреса записується в файл, щоб надалі виконувати пошук по пріоритету. Спочатку перевіряється записана IP-адреса, а потім, у разі помилки, програма переходить до стандартного пошуку. В такому разі при кожному наступному підключенні додаток може підключитись до веб-сервера менш ніж за пару секунд. Цього більш ніж достатньо, адже користувач буде рідко вимикати лампу від живлення, зазвичай це робиться звичайним вимикачем. Так і в цьому випадку, живлення на мікроконтролері залишається, а відповідно IP-адреса у певній кількості випадків буде сталою.

Як пам’ятаємо, в програму додавався функціонал для роботи із файлами, а наступний лістинг демонструє його реалізацію:

```
public Task<bool> FileNotEmptyAsync(string filename)    // функція для
{                                                       // перевірки, що файл існує та не пустий
bool filetext = false;                                  // змінна для перевірки пустоти файлу
string filepath = GetFilePath(filename);                // змінна шляху до файлу
bool exists = File.Exists(filepath);                    // змінна існування файлу
if (exists)                                             // якщо файл існує
{                                                       // починає передавати у потік дані із файлу
using (StreamReader reader = File.OpenText(filepath))
{
if (reader.ReadToEnd() != "")                           // якщо потік пустий
{
filetext = true;                                        // вказує, що файл пустий
} } }
Else                                                    // якщо потік містить якісь данні
{
```

```
filetext = false;                                    // вказує, що файл не пустий
}
return Task<bool>.FromResult(filetext);          // повертаємо результат
}
public async Task<string> LoadTextAsync(string filename) // функція для надання доступу до контенту програмі
{
string filepath = GetFilePath(filename);     // дізнаємось шлях до файлу
using (StreamReader reader = File.OpenText(filepath))        // починає
{                                                    // передавати у потік дані із файлу
return await reader.ReadToEndAsync();          // повертає дані з файлу
} }
public async Task SaveTextAsync(string filename, string text)
{                                                    //функція для запису тексту в файл
string filepath = GetFilePath(filename);     // дізнаємось шлях до файлу
using (StreamWriter writer = File.CreateText(filepath))                  // потік
{                                                    // починає передавати дані у файл
await writer.WriteAsync(text);     // у потік передаються дані для запису
} }
```

Зверніть увагу, що ці функції використовують асинхронний підхід, тому що написані на готових допоміжних асинхронних рішеннях.

Перейдемо до інтерфейсу програми, що зображений на рис. 3.3.

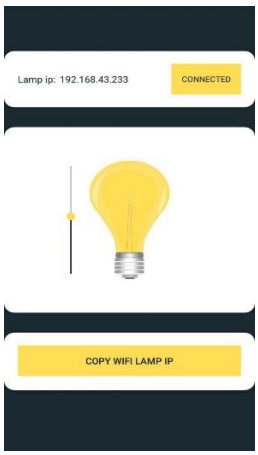

Рисунок 3.3 — Інтерфейс веб-додатку для смартфону

Знову ж таки, використовувалась мова XAML, але дещо перероблена під потреби смартфонів.

Власне, додаток на телефон більш залежний від ширини. Немає сенсу розтягувати все по довжині або робити квадратну форму основної панелі, адже тоді все буде сплюснутим, а текст буде погано видно.

```
<Grid BackgroundColor="#1B2930">                                    // основна сітка
<StackLayout Margin="0,25,0,25" HorizontalOptions="FillAndExpand"
VerticalOptions="Center">    // панель з основними елементами керування
        // кадр, що працює як і border, потрібний для округлення країв
<Frame CornerRadius="15" BackgroundColor="#FFFFFF">
<StackLayout                                        Orientation="Horizontal"
HorizontalOptions="FillAndExpand">// панель для під'єднання та виводу IP
<StackLayout                                        Orientation="Horizontal"
HorizontalOptions="FillAndExpand">                  // панель для виводу IP
<Label Text="Lamp ip:" VerticalTextAlignment="Center" FontSize="14"
TextColor="#1B2930"/>                               // мітка для тексту
<Label Text="Enter connect button" VerticalTextAlignment="Center"
FontSize="14"          TextColor="#1B2930"          x:Name="Result"
HorizontalOptions="Start"/>          // мітка, що містить інформацію про IP
</StackLayout>
<Button        Text="Connect"        x:Name="btn_c"        FontSize="12"
WidthRequest="100"  BackgroundColor="#FFDD55"  TextColor="#1B2930"
Clicked="Connect_Button" HorizontalOptions="End" />//клавіша приєднання
</StackLayout>
</Frame>
                                                    // округлюємо краї
<Frame        CornerRadius="15"        BackgroundColor="#FFFFFF"
Margin="0,25,0,0" HeightRequest="250">
        // панель із точним(x,y), або пропорційним позиціонуванням
 <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"/>       // стилі слайдеру
```

```
        <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"/>          // клавіша,
        </AbsoluteLayout>                                    // що містить зображення
        </Frame>
                                                             // округлюємо краї
       <Frame CornerRadius="15" BackgroundColor="#FFFFFF"
HeightRequest="50" Margin="0,25,0,25">
        <StackLayout HorizontalOptions="FillAndExpand">      // панель
        <Button Text="Copy wifi lamp ip" Clicked="Copy"
BackgroundColor="#FFDD55" TextColor="#1B2930"/>// клавіша копіювання
        </StackLayout>
        </Frame>
        </StackLayout>
        </Grid>
```

Тут кардинально змінюється спосіб написання стилів. Справа в тому, що при розробці додатку на телефон дуже складно проектувати щось під точні координати, адже у кожного телефону різна роздільна здатність. Таким чином, конкретні параметри вказуються доволі рідко. На заміну приходить розтягування та відносне позиціонування. Також можна відмітити, що змінюється синтаксис деяких атрибутів. Але в цілому це все ще схоже на комп'ютерну версію XAML та є доволі інтуїтивно зрозумілим.

4 ПРОЕКТУВАННЯ ПРИЛАДУ ТА ПЕРЕВІРКА ЙОГО ПРАЦЕЗДАТНОСТІ

4.1 Опис макету

Оскільки в дипломному проекті представлені перші версії такого приладу та програмного забезпечення, є можливість для вдосконалення. У розділах розробки наведені рішення, які працюють, але можуть містити певні недоліки. З цього випливає, що прилад можна продемонструвати, як макет. В такому разі з'являється можливість перевірки роботи периферії та віртуальної частини, а саме розгортування веб-сервера, взаємодія додатків із ним. Також з такої моделі перевірки стає ясно, що у подальшому можна змінити або що можна додати для збільшення функціоналу на базі прототипу.

Перейдемо до розробки макету. Потрібно визначитись із матеріалами та розмірами. Варто розуміти, що на пристрій подається напруга мережі 220 В. Отже, матеріал повинен бути діелектриком для запобігання небезпечних ситуацій. Тому в якості матеріалу макету використаємо звичайну дерев'яну пластину.

Оскільки це лише прототип і тут перевіряється робота пристрою, то поки немає потреби в обмеженні габаритних розмірів. Ми можемо розташувати елементи так, як буде зручно (в логічній послідовності згідно блок-схеми пристрою).

Для під'єднань модулів між собою використаємо 2 типи дротів. На вході пристрою, де подається напруга живлення, та на вихід з димера (з пристрою). Для цих цілей використаємо 2-ох жильний дріт із перерізом 0,5 мм^2 або ж 0,75 мм^2. При підключенні мікроконтролера до перетворювача AC/DC та підключенні до димера немає потреби у дроті із таким перерізом. В цьому випадку можна використати звичайні дроти для плат Arduino.

Для закріплення елементів на макеті використаємо звичайні стяжки. Просвердлимо відповідні отвори на платі, та закріпимо елементи.

Із міркувань безпеки всі силові дроти виведемо на низ макету, тому потрібно просвердлити ширші отворити для димера, лампи та перетворювача напруги. Щоб унеможливити ураження людини електричним струмом, силові дроти обіжмемо ізоляційними наконечниками. Плюс цього методу в тому, що при паралельному підключенні відпадає потреба в пайці контактів.

В результаті отримуємо макет, зображений на рис. 4.1.

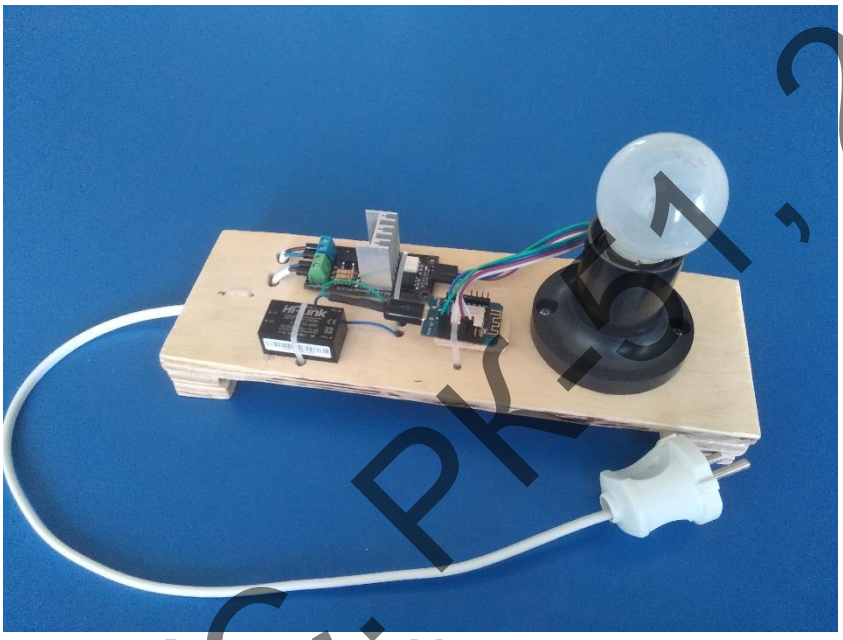

Рисунок 4.1 — Макет пристрою

Варто вказати, що для запобігання перегинання дротів макет трішки підняли використавши деревяні ніжки. Тепер можемо перейти до перевірки роботи макету.

4.2 Перевірка працездатності

Для перевірки працездатності пристрою в цілому варто перевірити основні аспекти взаємодії додатків із макетом, а саме механізми підключення та їх швидкодія, відповідність зміни режимів із показниками додатків.

Розпочнемо з тесту роботи макету з додатку на комп’ютер. Підключення виконується швидко, навіть при наявності 5 – 6 активних девайсів в мережі WiFi, як і було передбачено в розділі розробки програмного забезпечення.

При цьому клавіші керування починають вмикати та вимикати лампу, що підтверджує працездатність використаної моделі передачі керуючих сигналів. На лампі не спостерігається мерехкотіння світла, яке можна побачити при інших програмних реалізаціях або ж при використанні димерів інших виробників.

Сигнали із слайдера для зміни яскравості передаються доволі швидко, що не заглушує виконання програми та не вносить додаткові незручності (ефект пробуксовування повзунка слайдера).

Перевіримо оброблення помилок, які можуть виникнути при роботі програми. При відсутності підключення комп'ютера до WiFi, програма видає повідомлення про помилку, а не зупиняє своєї роботи. У разі натискання клавіш керування до того як програма підключилась до веб-серверу або WiFi, видається повідомлення про помилку. Отже з результатів видно, що поєднання асинхронних функцій із механізмом для відлову помилок try-catch дозволяє запобігти багатьом непередбачуваним помилкам та вильотам програм.

В основному додаток на смартфон повторює результати програми на комп'ютер, але як описано в розділі проектування програмного забезпечення він має певні відмінності у алгоритмі підключення, що потребує додаткових перевірок.

З приблизних теоретичних розрахунків швидкодії підключення при наявності 5 – 6 активних користувачів у мережі WiFi, в перший раз додаток повинен підключатись за час від 15 с до 20 с. На практиці ми отримуємо результати, які підтверджують зазначені оцінки швидкодії. Також було поєднання із механізмом, який є більш фундаментальним для роботи додатку на смартфон, а саме робота із файлами для пріоритетного підключення. В такому разі даний алгоритм забезпечує наступні підключення до веб-сервера за, приблизно, 2 секунди, що помітно покращує швидкодію та повністю задовольняє потреби користувача.

Перейдемо до так званих стрес тестів. Практика показала, що та ж конструкція асинхронних функцій та try-catch унеможливлює зупинку роботи програми внаслідок неправильних, з точки зору програміста, дій користувача. Програма також видає повідомлення про помилку у разі відсутності підключення до WiFi та застосування клавіш керування до підключення до веб-сервера.

Веб-сторінка призначена вже для людей, які знають IP-адресу веб-серверу. Отже потрібна тільки перевірка працездатності системи відправки керуючих сигналів, яка також має задовільні показники швидкодії.

Варто зазначити, що при одночасному підключенні додатків статус керуючих клавіш не змінюється відносно статусу самої лампи. Тобто відсутня система синхронізації між додатками.

Внаслідок перевірки працездатності отримано практичне підтвердження робити застосованих рішень, які в цілому задовольняють завдання цього дипломного проекту. Проте в подальшому краще удосконалити механізм підключення до веб-сервера для додатку на смартфоні, щоб швидкодія збільшилась навіть для першого підключення.

Також на базі такого рішення можна додати функціонал. В майбутньому потрібне додавання системи синхронізації додатків, керування кольором світла, тощо.

5 ОХОРОНА ПРАЦІ

Основним завданням цього розділу є визначення основних потенційно шкідливих та небезпечних виробничих факторів при розробці механізмів захисту інформації в безпровідній мережі із застосуванням ПК, а також розробка відповідних технічних рішень та організаційних заходів з безпеки, гігієни праці та виробничої санітарії на робочих місцях користувачів ПК та визначення основних заходів з пожежної безпеки та профілактики у робочих приміщеннях.

Оскільки даний пристрій призначений для використання на базі існуючих рішень, які мають сертифікати згідно з ДСНіП 239-96, то питання захисту від електромагнітного впливу не розглядається у даному розділі.

Основну увагу приділено питанням електробезпеки та питанням, які пов’язані із створенням безпечних та комфортних умов праці користувачів ПК.

У цьому розділі визначені основні потенційно шкідливі та небезпечні виробничі фактори, а також запропоновані необхідні рішення та організаційні заходи з безпеки гігієни та виробничої санітарії і визначено основні поняття пожежної безпеки та їх профілактика.

5.1 Визначення основних потенційно шкідливих та небезпечних виробничих факторів.

Основними потенційно небезпечними та шкідливими факторами, які мають місце при застосуванні ПК, є:

- можливістю поразкою електричним струмом;
- недостатнім рівнем освітленості або підвищеної яскравості освітленості робочої зони;
- пожежній безпеці;
- значне розумове навантаження тощо.

5.2 Технічні рішення та організаційні заходи з безпеки і гігієни та виробничої санітарії

5.2.1 Вимоги з охорони праці при роботі з ПК.

При роботі з комп'ютером працюючий піддається впливу наступних небезпечних та шкідливих факторів як:

- вплив електромагнітного випромінювання;
- наявність шуму вентиляторів;
- невідповідність освітлення;
- невідповідна організація робочого місця;
- можливість ураження електричним струмом;
- монотонність праці, тощо.

Важливу роль грає планування робочого місця, що повинна задовольняти вимогам зручності виконання робіт й економії енергії й часу працівника, зручності обслуговування пристроїв комп'ютера. Нераціональна конструкція й розташування робочих місць приводить до змушеної робочої пози й до напруги кістково-м'язової системи. При тривалій роботі за екраном дисплея в операторів спостерігається виражена напруга органів зору з появою скарг на незадоволеність роботою, дратівливість, порушення сну, хворобливі відчуття в очах, області шиї й у руках. У зв'язку з цим для працівників повинні забезпечуватися оптимальні умови праці й відпочинку. Праця операторів комп'ютера ставиться до I і II класу по гігієнічних умовах праці.

Вентилятори системних блоків при значному шумі слід замінювати.

Тривалість роботи оператора за кольоровим монітором не повинна перевищувати шести годин на добу.

Під час роботи електронно-променевої трубки екран опромінюється потоком заряджених часток, а це у свою чергу викликає виникнення електростатичного поля. Електростатичне поле притягає багато пилу й створює навколо таких приладів підвищену концентрацію пилу, що несприятливо впли-

ває на організм людини. Для боротьби із цим шкідливим фактором використається спеціальне екранування.

Основним джерелом електромагнітного випромінювання є котушка системи горизонтального та вертикального відхилення, що знаходиться біля цокольної частини електронно-променевої трубки. Дія електромагнітного випромінювання на організм людини проявляється у функціональному розладі ЦНС (підвищена стомлюваність, головні болі).

Умови при роботі із моніторами ПК повинні відповідати вимогам щодо безпеки захисту працівників під час роботи з екранними пристроями та ДСанНПіН_3.3.2.007-98.

5.2.2 Електробезпека.

Причинами поразок електричним струмом можуть бути:

- дотик до струмоведучих частин;
- дотик до відключених струмоведучих частин або до конденсаторів, на яких залишився залишковий заряд в разі помилкового включення установки;
- дотик до неструмоведучих металевих частин при аварійному режимі роботи електроустаткування.

Прилади й устаткування живляться від 3-х фазної електричної мережі напругою 220В и частотою 50Гц, тобто відносяться до електроустановок напругою до 1000В.

Згідно з ДСТУ IEC 61140:2015 електрообладнання, яке встановлене у робочому приміщенні має 0I клас електрозахисту - вимірювальні пристрої, I клас – системні блоки ПК, II клас – ПК, III – допоміжне обладнання та периферія ПК.

Виробниче приміщення згідно ПУЕ-2017 та ДБН В.2.5-27 – 2006 відноситься до приміщень без підвищеної небезпеки поразки персоналу електричним струмом, оскільки:

- відносна вологість повітря не перевищує 75%;

– температура не вище 35˚С;

– відсутні хімічно агресивні середовища;

– відсутня можливість одночасного дотику до металевих елементів електроустаткування та до металоконструкцій будинку, які з'єднані із землею;

– має місце надійне заземлення та занулення.

Проведемо перевірочний розрахунок вимикаючої здатності автоматів захисту й вирівнювання режиму роботи елемента заземлення корпусів електроустаткування. Струм короткого замикання визначаються по формулі:

$$I_{кз} = \frac{U_ф}{R_ф + R_н + \frac{Z_T}{3}}$$

де $U_ф$ _ фазова напруга, 220В;

де $R_ф$ — опір фазового проводу, 2Ом;

де $R_н$ — опір нульового проводу, 1,6Ом;

де $\frac{Z_T}{3}$ — еквівалентний опір трансформатора, 2Ом.

Підставивши всі значення отримаємо результат 57,89А. Для правильної роботи автоматів максимального струмового захисту повинна виконуватись нерівність:

$$I_{кз} \geq 1{,}4 \cdot I_{от}$$

де $I_{от}$ – номінальний струм спрацювання автомату захисту. Звідси визначаємо, що $I_{от}$ повинен бути не більше, ніж 41,35А. Автомати, що використовуються у робочих приміщеннях мають номінальний струм спрацювання 20А – 25А, що відповідає вимогам.

Розрахуємо напругу дотику до зануленого встаткування:

$$U_п = I_{кз} \cdot R_н$$

Розрахована величина напруги дотику $U_п$ при часі спрацьовування автоматів струмового захисту t < 0.5 с не перевищує припустимого значення

$U_{доп}$ = 100 В, що задовольняє вимогам ПУЕ-2017.

Додаткових засобів по забезпеченню електробезпечності не потрібно.

5.2.3 Вимоги до освітленості робочих місць користувачів ПК.

Одним із елементів, що впливають на комфортні умови праці працюючих, є виробниче освітлення. До систем виробничого висвітлення, що обслуговує робочі місця з комп'ютером, пред'являються наступні основні вимоги:

- відповідність рівня освітленості робочих місць характеру виконуваної зорової роботи;
- рівномірний розподіл яскравості на робочих поверхнях й у навколишньому просторі;
- відсутність різких тіней, прямих і відбитих відблисків;
- сталість освітленості в часі й просторі;
- оптимальна спрямованість випромінюваного освітлювальними приладами світлового потоку;
- довговічність, економічність, електро- і пожежобезпечніть, естетичність, зручність і простота експлуатації.

Для висвітлення робочих місць, обладнаних ПК, застосовується природне, штучне освітлення.

Згідно вимогам щодо безпеки захисту працівників під час роботи з екранними пристроями та ДСаНПіН 3.3.2-007-98, освітлення у виробничих приміщеннях з моніторами комп'ютерів повинна бути сумісне. Природне освітлення повинне здійснюватися через отвори, орієнтовані переважно на південь і північний схід. Згідно ДБНВ 2.5-28-2006 «Природне й штучне освітлення. Норми проектування» коефіцієнт природної освітленості (КПО) виробничих приміщень із моніторами комп'ютерів повинен бути не нижче 1,5 %.

Розташування будинків і планування виробничих приміщень повинне виключати надмірне надходження тепла від сонячної радіації через вікна і пряме попадання сонячних променів на пристрої ПК і носії інформації.

Штучне освітлення в приміщеннях з моніторами комп'ютерів необхідно здійснювати по загальній системі рівномірного освітлення. Світильники загального освітлення необхідно розташувати у виді ліній (суцільних або переривчастих) збоку від робочих місць паралельно лінії зору користувачів. Допускається використання світильників таких класів світлорозподілу:

- прямого світла (П);
- переважно прямого світла (Н);
- переважно відбитого світла (В).

Як джерела штучного освітлення використовуються люмінесцентні лампи типу ЛБ із розсіювачами й екранними сітками.

Коефіцієнт пульсації світлового потоку джерел світла не повинен перевищувати 5 %. Для зменшення коефіцієнта пульсації світлового потоку необхідно використати джерела світла з високочастотними пускорегулюючими апаратами. Яскравість світильників загального висвітлення в зоні кутів випромінювання від 50° до 90° щодо вертикалі в поздовжній і поперечній площинах повинна становити не більше 200 кд/м^2, а захисний кут світильників повинен становити не більше 40°.

Для обмеження прямої близькості від джерел природного й штучного освітлення необхідно, щоб яскравість їхніх поверхонь, які перебувають у полі зору користувачів, не перевищувала 100 кд/м^2, яскравість відблисків на екрані монітору комп’ютера 40 кд/м^2, а яскравість стелі 200 кд/м^2.

У поле зору користувача монітора комп'ютера повинне бути забезпечене відповідний розподіл яскравості. Відношення значень яскравості робочих поверхонь до загальної яскравості у приміщенні не повинне перевищувати 3:1, а робочих поверхонь і навколишніх предметів (стіни, устаткування, меблі) – 5:1.

У виробничих приміщеннях з моніторами комп'ютерів показник засліпленості повинен становити не більше 20 одиниць, а показник дискомфорту – не більше 40 одиниць.

Для забезпечення нормованих показників освітленості в приміщеннях з моніторами ПК необхідно не менш 2 разів у рік очищати від пилу й бруду скла вікон і світильники і вчасно замінювати несправні світильники.

5.3 Пожежна безпека та профілактика.

Причини виникнення пожежі в робочому приміщенні, де використовується комп’ютер, можуть носити електричний і неелектричний характер.

До причин електричного характеру ставляться: короткі замикання, перевантаження, іскріння від порушення ізоляції, електрична дуга, що виникає між контактами комутаційних апаратів, незадовільні контакти в місцях з'єднання проводів (скрутки) і їхнє сильне нагрівання внаслідок великого перехідного опору при протіканні електричного струму.

Причини неелектричного характеру: порушення режимних вимог, халатне й необережне поводження з вогнем, порушення правил пожежної безпеки.

Приміщення відповідно до ДБН В.2.5-56-2014 обладнане чотирма пожежними датчиками типу ДПС-038 (площа, що перебуває під захистом одного датчика становить до 25 м2, відстань між датчиками становить 4м). Відповідно до ДСТУ 3675-98 і ISO 3941-77 як первинний засіб гасіння пожежі використовується чотири вогнегасника ОУ-3 – вуглекислий (клас пожежі “Е”). Вибір речовини вогнегасника обґрунтовується тим, що у вогні можуть виявитися електричні пристрої, що перебувають під напругою. Кількість, розміщення і зміст первинних засобів гасіння пожежі повністю задовольняє всім вимогам ДСТУ 3675-98 й ISO 3941-77. Крім цього, у коридорі є 2 пожежних крани і ящик з піском. Витримано всі умови НАПБ А.01.001-2004 «Правила пожежної безпеки в Україні».

Витримано всі умови ДБН В.1.1-7-2016 по вогнестійкості будинків, часу евакуації у випадку пожежі, ширині евакуаційних проходів і виходів із приміщень. Двері приміщень відкриваються назовні, ширині дверей 1.3 м при нормі не менш 0.8 м, висота проходу 2.2 м при нормі не менш 2м, ширина коридору 3м при нормі не менш 2м.

ВИСНОВОК

1. Оглянувши існуючі аналоги на ринку, були виявлені такі основні недоліки, як слабке підключення, погана швидкодія та невідповідність якості до ціни даних виробів, а у деяких випадках – неможливість керування одночасно з декількох смартфонів. Проведено аналіз можливих схемотехнічних рішень, згідно з яким виявлені основні електронні модулі, необхідні для роботи пристрою, як мікроконтролер, димер, перетворювач AC/DC. Розглянуто приклади можливої програмної реалізації, які вказали на наявність готових рішень для підключення до WiFi-мережі та розгортання веб-сервера на базі мікронтролера з використанням бібліотеки ESP8266WebServer.
2. Відповідно до аналізу розроблено структурну схему та обрані потрібні електронні модулі: ESP8266, димер від RobotDyn, перетворювач HLK-PM01.
3. Обрано основні технології та мови для написання коду: C/C++, XAML, JavaScript, HTML/CSS. Розроблено логіку мікроконтролера, тобто поєднання розгортування веб-сервера та керування димером. Створено додатки для комп’ютера, смартфона та веб-сторінку для дистанційного керування режимами пристрою.
4. Створено макет для демонстрації роботи пристрою та тестування взаємодії додатків та веб-сторінки з пристроєм. Виявлено швидкодію, яка при подачі керуючих сигналів досягала 0,5 с, а при підключенні до веб-сервера становила 2 с для додатку на комп’ютер, 15 с для смартфона під час першого підключення та до 2-ох секунд при наступних підключеннях. Проведено стрес тести додатків, що виявили доволі низький відсоток ймовірності поломки програми.
5. Розроблений пристрій задовольняє вимоги технічного завдання.

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. ESP8266 [Електронний ресурс]. — Режим доступу: https://ardushop.in.ua/arduino/wi-fi-module-esp8266-version-esp-12e — Назва з екрану.
5. Arduino Controlled Light Dimmer [Електронний ресурс]. — Режим доступу:https://www.instructables.com/id/Arduino-controlled-light-dimmer-The-circuit/ — Назва з екрану.
6. Димери від компанії HDL [Електронний ресурс]. — Режим доступу:https://dardali.com/academiya-praktiki/teoriya/63-dimmer-standarta-knx-hdl — Назва з екрану.
7. Реалізація димирування стандартними методами [Електронний ресурс]. — Режим доступу: https://github.com/AlexGyver/AC_Dimmer — Назва з екрану.
8. Документація на бібліотеку CyberLib [Електронний ресурс]. — Режим доступу: https://github.com/pythonista/CyberLib — Назва з екрану.
9. Документація на бібліотеку AC_Dimmer [Електронний ресурс]. — Режим доступу: https://arduino.ua/docs/AOC117/AC_Dimmer-3.zip — Назва з екрану.
10. Документація на бібліотеку ESP8266WebServer [Електронний ресурс]. — Режим доступу: https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WebServer — Назва з екрану.

11. Машины, приборы и другие технические изделия. Термін та видання: ГОСТ 15150-69. — [Введ. в дію 01.01.1971]. — М. : Государственный стандарт союза ССР, 1969. — 81с.
12. Аппаратура сухопутной подвижной радиосвязи.. Термін та видання: ГОСТ 16019-2001. — [Введ. в дію 01.01.2002]. — М. : Межгосударственный стандарт, 2001. — 11с.
13. WeMos D1 mini [Електронний ресурс]. — Режим доступу: https://www.mini-tech.com.ua/wemos-d1-mini — Назва з екрану.
14. Одноканальний димер 220D для Arduino від RobotDyn [Електронний ресурс]. — Режим доступу: http://arduino.ua/prod3117-odnokanalnii-dimmer-220v-dlya-arduino-ot-robotdyn — Назва з екрану.
15. Hi-Link HLK-PM01 [Електронний ресурс]. — Режим доступу: https://www.mini-tech.com.ua/blok-pitaniya-5v-600ma-kompakt — Назва з екрану.
16. Документація по мові HTML [Електронний ресурс]. — Режим доступу: https://www.w3.org/TR/html52/ — Назва з екрану.
17. Документація по мові CSS [Електронний ресурс]. — Режим доступу:https://www.w3.org/Style/CSS/ — Назва з екрану.
18. Документація по мові JavaScript Електронний ресурс]. — Режим доступу:https://developer.mozilla.org/uk/docs/Web/JavaScript — Назва з екрану.
19. Документація по мові XAML [Електронний ресурс]. — Режим доступу:https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/xaml-in-wpf— Назва з екрану.
20. Документація по мові C# [Електронний ресурс]. — Режим доступу:https://docs.microsoft.com/en-us/dotnet/framework/wpf/getting-started/— Назва з екрану.

21. Документація по мові XAML для Xamarin Forms [Електронний ресурс]. — Режим доступу:https://docs.microsoft.com/en-us/xamarin/xamarin-forms/xaml/ — Назва з екрану.
22. Документація на бібліотеку RBDDimmer [Електронний ресурс]. — Режим доступу: https://github.com/RobotDynOfficial/RBDDimmer — Назва з екрану.

ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ

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 ІНТЕРФЕЙСА ДЛЯ ДОДАТКУ НА КОМП'ЮТЕР

```
<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 КОМП'ЮТЕР

```
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 ІНТЕРФЕЙСА 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
{
    // Learn more about making custom code visible in the Xamarin.Forms previewer
    // by visiting https://aka.ms/xamarinforms-previewer
    [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 { }
  }
 }
}
```