**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___» _____2021 р.

# Дипломна робота

**на здобуття ступеня бакалавра**

**спеціальності 121 «Інженерія програмного забезпечення»**
**освітня програма «Інженерія програмного забезпечення розподілених систем»**

**на тему: «Інтелектуальне освітлення приміщення з врахуванням індивідуальних потреб користувача»**

Виконав:
студент IV курсу, групи ТІ-72
Роєнко Олександр Олегович _____

Керівник:
Ст. викладач
Сарибога Ганна Володимирівна _____

Консультант:
_____

Рецензент:
Ст. викладач
Наливайчук Миколай Васильйович _____

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

Київ — 2021 року

**Національний технічний університет України**
**“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2021р.

# ЗАВДАННЯ
**на дипломну роботу студенту**

Роєнку Олександру Олеговичу
(прізвище, ім’я, по батькові)

1. Тема роботи Інтелектуальне освітлення приміщення з врахуванням індивідуальних потреб користувача
керівник роботи Сарибога Ганна Володимирівна, ст. викладач.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”24” травня 2021р. № **1267-с**

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи мова програмування Dart, Java та C\C++, фреймворк Flutter та Spring, модулі Arduino, середовище розробки Android Studio, Arduino Ide, IntelliJ Idea

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) дослідити літературу та існуючі рішення; розробити архітектуру застосунку; реалізувати програмне забезпечення інтелектуального освітлення з врахуванням індивідуальних потреб користувача згідно розробленої архітектури; здійснити тестування системи.

5. Перелік ілюстративного матеріалу:
Теоретичні відомості. Існуючі рішення. Діаграма розгортання. Діаграма прецидентів. Діаграма класів.. Апаратна частина програмного забезпечення. Інтерфейс мобільного застосунку. Висновки.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |

7. Дата видачі завдання "___"_____202___р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання дипломної роботи | Термін виконання етапів роботи | Примітки |
|---|---|---|---|
| 1. | Затвердження теми роботи | 16.02.2020 | |
| 2. | Вивчення та аналіз задачі | 09.10.2020 – 23.01.2021 | |
| 3. | Розробка архітектури та загальної структури системи | 09.03.2021 – 27.03.2021 | |
| 4. | Розробка структур окремих підсистем | 28.03.2021 – 01.04.2021 | |
| 5. | Програмна реалізація системи | 05.04.2021 – 28.04.2021 | |
| 6. | Оформлення пояснювальної записки | 20.05.2021 – 04.06.2021 | |
| 7. | Захист програмного продукту | 14.05.2021 | |
| 8. | Передзахист | 28.05.2021 | |
| 9. | Захист | 15.06.2021 | |

Студент _____ _____ Роєнко О.О
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Сарибога Г.В.
(підпис) (прізвище та ініціали,)

# АНОТАЦІЯ

Дипломна робота на здобуття першого (бакалаврського) рівня вищої освіти складається зі вступу, чотирьох розділів, висновків, списку використаних джерел із 30 найменувань, 4 додатків. Загальний обсяг 84 сторінки. Робота містить 4 таблиці та 24 рисунки.

При огляді літератури з'ясовано сутність основних понять дослідження. Визначено вплив освітлення на циркадні ритми та загалом на самопочуття людини. З'ясовано, що порушення добових ритмів може призвести до порушення психофізіологічного стану людини.

Окрема увага у дослідженні приділена висвітленню концепції людино-орієнтованого освітлення, що зосереджує увагу на поверненні динаміки природного денного світла у повсякденне життя людей за допомогою біологічно ефективного штучного освітлення. Реалізація такого підходу можлива через широке впровадження різного виду освітлення. Великі можливості створюють світлодіодні технології, які позитивно вирізняються серед інших видів освітлення енергоефективністю, рівномірністю освітлення, керованістю, довговічністю, впливом на навколишнє середовище тощо.

З метою збільшення позитивного впливу освітлення на людину було розроблено систему інтелектуального освітлення приміщення із врахуванням потреб користувача, що має такі функції: налаштування яскравості та колірної температури; налаштування кольору; автоматична адаптація яскравості; налаштування графіку увімкнення та вимкнення; налаштування будильника; містить режими за замовчунням освітлення для певної діяльності; відслідковує циркадні ритми; визначає користувача та застосування його налаштувань.

На практиці реалізовано серверну частину, мобільний додаток та апаратну частину.

Можливе подальше доопрацювання даної системи

**Ключові слова:** IoT, світлодіоди, циркадні ритми, апаратна частина, людинноорієнтоване освітлення

## ABSTRACT

Thesis for the first (bachelor's) level of higher education is created from the introduction, four sections, bran, a list of used sources of 30 items, 4 appendices. The total volume is 84 pages. The work contains 4 tables and 24 figures.

A review of the literature reflects the essence of the basic concepts of the study. The influence of lighting on circadian rhythms and in general on human well-being is determined. It has been found that the violation of circadian rhythms can lead to a violation of the psychophysiological state of man.

On the need to improve research on the coverage of the concepts of human-oriented lighting, which on average draws attention to the return of the dynamics of the natural daylight in the full life of people with biologically effective artificial lighting. The implementation of this approach may be due to the widespread introduction of different types of lighting. Great opportunities are created by LED technologies which are positively solved among other types of lighting of energy-efficient, uniform illumination, controllability, duration, influence on environment of another.

Due to the positive impact of lighting on people, a system of intelligent lighting of the room was developed with the help of user needs, which has the following functions: creation of features and color temperatures; color location; automatic brightness adaptation; location of on and off schedules; location of the alarm clock; contains default lighting modes for certain activities; monitors circadian rhythms; identifies users and applies their location.

In practice, the server part, mobile application and hardware part are implemented.

Further refinement of this system is possible



Keywords: IoT, LEDs, circadian rhythms, hardware, human-oriented lighting

**Зміст**

**ВСТУП ..... 7**

**1. ПОСТАНОВКА ЗАДАЧІ ..... 9**

**2. АНАЛІЗ ЛІТЕРАТУРИ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ..... 10**

2.1 Інтернет речей ..... 10

2.2 Вплив освітлення на циркадні ритми людини ..... 11

2.3 Світловий потік, освітленність та колірна температура ..... 15

2.4 Людино-орієнтоване освітлення ..... 18

2.5 Переваги та недоліки світлодіодів ..... 19

2.6 Огляд існуючих рішень ..... 25

**3. ЗАСОБИ РОЗРОБКИ ..... 27**

3.1 Опис системи та необхідних технологій ..... 27

3.2 Технології розробки серверної частини ..... 29

3.3 Технології розробки мобільного застосунку ..... 31

3.4 Технології розробки апаратно-програмного комплексу ..... 32

**4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ..... 36**

4.1 Опис програмної реалізації серверної частини ..... 36

4.2 Опис програмної реалізації апаратно-програмного комплексу ..... 40

4.3 Опис програмної реалізації мобільного застосунку ..... 43

4.4 Інструкція з використання ..... 45

**Висновки ..... 54**

**ДОДАТОК А ..... 59**

**ДОДАТОК Б ..... 61**

**ДОДАТОК В ..... 75**

**ДОДАТОК Г ..... 84**

# ВСТУП

Протягом тривалого часу індустрія освітлення вкладала практично всі свої ресурси в технології, рішення та продукти, які відповідають нашим візуальним потребам. Лише після відкриття третього фоторецептора (крім паличок і колбочок) в людському оці та доказів біологічного впливу світла, вчені почало пов'язувати здоров'я та добробут людини зі світлом. Висновки щодо невізуальних ефектів світла з того часу спонукають галузь застосовувати більш цілісний підхід, який одночасно враховує як візуальні, так і біологічні потреби людини.

Освітлення має досить значний вплив на самопочуття та емоційний стан людини. Наприклад, м'яке світло розслабляє і налаштовує організм на відпочинок, сон, яскраве денне світло навпаки підвищує продуктивність. Природне світло ідеально підходить для зору та нервової системи, тьмяне світло може провокувати тривогу.

Більше того, освітлення має прямий вплив на біологічний годинник та циркадні ритми людини, обмін речовин та вироблення гормонів. Вплив неправильного світла на людину виражається в сплесках активності та ентузіазмі, раптовій повній апатії, сонливості та неймовірній втомленості.

З розвитком технологій з'являється все більше нових видів освітлення, які позитивно впливають на людину та її самопочуття. Винахід світлодіодних джерел світла став величезним проривом в історії штучного освітлення. Значення світлодіодної технології полягає не лише у високоефективному перетворенні енергії. Вражаюча керованість світлодіода, порівнянно з іншими напівпровідниковими пристроями, пропонує величезні можливості для реалізації розширених функцій. Його програмована здатність динамічно визначати спектр світла для повного діапазону колірних температур від дуже теплого білого до білого денного світла дозволяє системам освітлення HCL повторювати зміни кольору. Наразі представлено достатньо видів ламп, якими можна керувати за допомогою телефону, проте не всі вони здатні автоматично підлаштовуватися під людину, позитивно впливаючи на неї.

Теоретичні положення дослідження можуть бути використані здобувачами вищої освіти, викладачами з метою ознайомлення з особливостями впливу освітлення на продуктивність та самопочуття людини.

Подані в роботі практичні розробки сприятимуть впровадженню основ людино-орієнтованого освітлення в навчальних закладах різного рівня, підприємствах, офісах, приватних будівлях.

# 1. ПОСТАНОВКА ЗАДАЧІ

**Мета:** з’ясувати сутність людиноорієнтованого освітлення визначити вплив освітлення на самопочуття людини освітлення та розробити технологію людинноорієнтованого освітлення.

**Завдання роботи** – розробити системи інтелектуального освітлення із можливостями керування через мобільний додаток за допомогою Wi-Fi та Bluetooth. Призначення даної системи є створення комфортного освітлення для певного виду діяльності та адаптація під циркадні ритми людини.

Дана система повинна містити наступний функціонал:

1. Налаштування яскравості світла
2. Налаштування кольору
3. Автоматична адаптація яскравості
4. Налаштування таймеру увімкнення та вимкнення
5. Налаштування будильника
6. Містити запрограмовані режими освітлення за замовчуванням для певного виду діяльності
7. Адаптація під циркадні ритми людини
8. Визначення користувача, врахування індивідуальних його потреб та застосування його налаштувань

# 2. АНАЛІЗ ЛІТЕРАТУРИ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

## 2.1 Інтернет речей

З розвитком технологій все більше і більше девайсів підключаються до інтернету, які до того з ним ніяк не були поєднані. Наприклад, лампа, якою ми можемо керувати за допомогою мобільного додатоку, перебуваючи навіть у іншому місці, розумне взуття, яке відслідковує інформацію та видає нам статистку, та багато інших пристроїв. Даний розвиток повпливав на введення такого поняття як «інтернет речей».

**Інтернет речей (Internet of things – IOT)** – об'єднання фізичних пристроїв, підключених до мережі, які збирають дані та обмінюються ними між собою за допомогою використання стандартних протоколів зв'язку [29]. Ідея об'єднання пристроїв у єдину мережу з'явилася ще в 70-х роках XX століття, але сам термін «Інтернет речей» був введений Кевіном Ештоном у 1999 р., проте особливої уваги дане поняття набуло тільки через 10 років [30].

Інтернет речей нині використовується у таких важливих галузях: енергетика, сільське господарство, логістика, медицина, тощо. Наприклад, в енергетиці інтернет речей дозволяє ефективніше керувати підстанціями та лініями електропередач. У сільському господарстві може керувати кількістю необхідних добрив які треба додати і додає їх автоматично. Також дає можливість слідкувати за самопочуттям тварин на фермах та керувати технікою автоматично. Прикладом використання інтернету речей у медицині є розумні пристрої, які слідкують за здоров'ям та надсилають інформацію, що стосується самопочуття, лікарю. Якщо брати до уваги не галузь, а використання у домашніх умовах, то яскравим прикладом є технологія розумний будинок, яка дає змогу користувачеві керувати всіма пристроями інтернету речей самостійно з одного місця або навіть дозволити автоматичне керування, яке братиме до уваги показники з датчиків, що розміщені в будинку.

Отже, галузей, у якій використовується інтернет речей, і прикладів є багато і кількість їх з кожним роком зростає.

Архітектура інтернету речей включає у себе:

– **пристрої** – об’єкти з яких і складається інтернет речей. Вони є інтерфейсами між реальним та цифровим світами;

– **програмне забезпечення** – відповідає за описану логіку роботи, інтеграцію з іншими сервісами та аналіз даних;

– **комунікація** – протоколи зв’язку між сервісами та користувачем;

– **платформа –** місце, де збираються та аналізуються всі дані;

– **безпека –** захист від різного роду кібер атак. Прикладом може слугувати використання безпечних протоколів, сервісів та регулярні оновлення пристрою.

Найсерйознішим питанням в інтернету речей є питання безпеки. Через те що багато пристроїв приєднані до мережі, то у випадку взлому зловмисник зможе отримати доступ до даних або керувати всією системою, що є великою проблемою, враховуючи, що інтернет речей все більше з’являється у багатьох описаних вище галузях. Даний чинник дещо сповільнює інтеграцію інтернету речей. Прикладом однієї з багатьох атак може слугувати використання ботнета Mirai, який перебором логінів та паролів, що були встановлені за замовчуванням, взламав велику кількість камер та роутерів, які потім використали для DDoS атак [2].

### 2.2 Вплив освітлення на циркадні ритми людини

Одним із важливих понять, яке необхідно дослідити для розробки дослідницької проблеми, є циркадні ритми людини та вплив освітлення на них.

Циркадні ритми – біологічний ритм з періодом 24 години, який координує багато процесів в організмі, включаючи сон [12]. Назва походить від латинського словосполучення „*circa diem*”, що означає протягом дня.

Дослідникам було відомо, що зорові відчуття у людини забезпечують два фоторецептори в сітківці: палички і колбочки. Стрижневий фоторецептор має низьку чутливість, забезпечує скотопічний зір (нічне бачення) і сприяє зору в

середньому діапазоні (мезопічний зір). Конусоподібний фоторецептор, який реагує на всі кольори зорового спектру, сприяє зору при високому рівні освітлення (фотопічний зір). На початку цього тисячоліття було виявлено третій тип фоторецепторів, власне світлочутливі клітини гангліїв сітківки (ipRGC) в оці. IpRGC, які також називаються світлочутливими гангліозними клітинами сітківки (pRGC), утворюють головний нервовий канал до супрахіазматичних ядер (SCN).

IpRGC беруть участь у циркадній фототрансдукції, біохімічному процесі, в якому фоторецепторні клітини генерують електрофізіологічні сигнали у відповідь на захоплені фотони.

SCN реагує на неврологічні сигнали, що передаються ipRGC через ретино-гіпоталамусовий тракт (RHT), і регулює секрецію або пригнічення таких гормонів, як дофамін, кортизол, серотонін та мелатонін. При слабкому освітленні та в умовах темряви епіфіз спрацьовує з утворенням циркадного нейрогормону мелатоніну (N-ацетил5-метокситриптаміну), що викликає сонливість та сприяє сну. Коли око відчує оптичне випромінювання з більшою інтенсивністю, вивільнення мелатоніну буде зменшено, але збільшено вироблення дофаміну, кортизолу та серотоніну. Допамін – це ендогенний катехоламін, який необхідний для забезбечення пильності та координації м'язів. Кортизол – гормон стресу, який сприяє неспанню та змушує людей відчувати себе пильними та активними. Серотонін – це нейромедіатор, який забезпечує контроль імпульсів, підвищує настрій та мотивацію.

Фоточутливість ipRGCs головним чином зумовлена фотопігментом, який називається меланопсин. Зорова система з дефіцитом меланопсину все ще може допомогти людському тілу адаптуватися до циклу світло-темрява, оскільки класичні фоторецептори (тобто палички та колбочки) також сприяють циркадній фототрансдукції. Фотобіологічна реакція ослаблюється приблизно на 40%. Меланопсин реагує лише на короткохвильове блакитне світло і має пікову світлочутливість на довжинах хвиль, переважно від 459 нм до 484 нм. Коли очне світло (світло, що сприймається очима) з високим відсотком синього досягає цих меланопсинвмісних ipRGC, меланопсинові рецептори активуються і сигналізують головному циркадному годиннику SCN припинити вивільнення гормону сну

мелатоніну при збільшенні секреції дофаміну кортизол та серотонін, щоб допомогти організму підготуватися до денної активності. Безперервний вплив світла збагачений синім кольором протягом дня імітує денну фізіологічну реакцію, підвищує температуру тіла та частоту серцевих скорочень, збільшує м'язову силу та координацію, максимізує пильність і продуктивність [23].

Циркадні ритми керуються супрахіазматичним ядром, яке являє собою скупчення близько 2000 нервових клітин. Дане ядро розташовується у гіпоталамусі і надсилає сигнали, спрямовуючи організм на активацію певних дій в залежності від часу доби. Багато клітин організму мають свої власні ритми і завданням супрахіазматичного ядра є синхронізація даних ритмів. Протягом дня коливання циркадних ритмів забезпечує ефективну роботу організму [22].

Циркадні ритми є складовою біологічного годинника, що допомагає регулювати час фізичних процесів. Добовий ритм – це вплив біологічних годин, але не всі біологічні годинники є циркадними. Наприклад, рослини пристосовуються до зміни сезону, використовуючи біологічний годинник із часом, який відрізняється від 24-годинного циклу [22].

Однією з найбільш важливих функцій, на яку впливають циркадні ритми, це регулювання сну. Ми проводимо третину життя у сні і не можемо без нього існувати. Коли ми спимо, наш мозок аналізує та запам'ятовує інформацію отриману протягом дня, наше тіло позбавляється токсинів і відновлюється, дозволяючи нам нормально функціонувати після пробудження [7].

Протягом дня під впливом випромінювання світла головний годинник сприяє генеруванню енергії, допомагає бути активними та пильними, в той час як з настанням темряви ініціює вироблення мелатоніну. Мелатонін – гормон який тісно пов'язаний зі світлом. У відповідь на темряву епіфіз головного мозку виробляє мелатонін, а світло впливає на його кількість на швидкість вироблення [22]. Наприклад, сонливість зростає із збільшенням мелатоніну. Крім того, щоденні цикли вироблення мелатоніну нормалізують циркадні ритми, створюючи стабільний графік сну та пробудження [21].

Кожен сон має свої цикли, які тривають від 70 до 120 хвилин. Зазвичай людина проходить 4-6 циклів сну. Кожен цикл складається із таких стадій сну:

дрімота;

поверхневий сон;

глибокий сон;

REM (кінцевий етам у циклі сну).

Вплив світла вночі може перешкоджати переходу між циклами сну, знижуючи якість сну. Занадто багато світла може спричинити пробудження, перериваючи цикл сну і скорочуючи час проведення на більш глибоких та відновлювальних стадіях сну.

Сучасна тенденція того, що люди проводять більшу частину часу в приміщенні, суттєво підвищує ризик циркадних збоїв як наслідок неправильного впливу штучного освітлення. Офіси, школи, комерційні та промислові об'єкти використовують інтенсивне, насичене синім світлом світло, щоб тримати людей більш пильними, зосередженими, чуйними та продуктивними. Вплив світла, насиченого синім кольором, в денний час є важливим для поліпшення життєвого тонусу та концентрації уваги, але це може нашкодити здоров'ю, якщо цей тип впливу світла подовжується на нічні години. Насправді велика частина населення часто працює або вчиться глибоко до ночі. Ці люди мають високий ризик циркадного порушення, оскільки нічний вплив біоактивного синього світла різко пригнічує секрецію мелатоніну і змушує організм порушити біологічний годинник. Наш біологічний годинник еволюціонував, щоб відстежувати схід та захід сонця. Відсутність або дуже низький рівень біоактивного синього світла після заходу сонця забезпечує нічний сигнал, який повідомляє мозку виробляти мелатонін. Секреція мелатоніну починає підніматися близько 21:00 і досягає піку близько 2:00 ночі. Мозок припинить вироблення мелатоніну до 7:00. Зміна циклів секреції мелатоніну через сезонні зміни, робочі нічні зміни або подорожі через різні часові пояси також може призвести до циркадних збоїв. SCN та периферійні годинники повинні будуть адаптуватися до нових циклів світло – темно.

Циркадні порушення в сучасному суспільстві, як правило, є результатом придушення секреції мелатоніну або зміни часу секреції мелатоніну. Протягом ночі вивільнення мелатоніну забезпечує регенеративний сон та освіжає наш організм. Вважається, що мелатонін внутрішньо синхронізує периферійні годинники з SCN і передає фотоперіодичну інформацію до розподілених систем по всьому тілу. Цей гормон безпосередньо і опосередковано бере участь у регуляції ендокринних, нейрональних, імунологічних та поведінкових процесів. Епіфізний ритм мелатоніну забезпечує фотоперіодичні корекції імунного захисту. Нічний викид мелатоніну сприяє життєво важливому захисному механізму, який пригнічує ріст ракових клітин в організмі.

Отже, порушення добових ритмів може призвести до ряду негативних наслідків. Це впливає не тільки на сон, що призводить до втоми, зниження денної працездатності та різних порушень сну, таких як безсоння та апное сну. Внутрішня десинхронізація між SCN та периферичними годинниками впливає на весь наш метаболізм та проліферацію клітин. Дослідженнями підтверджено, що порушення циркадних ритмів пов'язане з ожирінням, діабетом, депресією, біполярними розладами, серцево-судинними розладами, репродуктивними проблемами та сезонними афективними розладами. Незбалансоване вивільнення гормонів може погіршити функціонування імунної системи та перешкоджати експресії годинникових генів, пов'язаних з раком.

Беручи до уваги всі описані вище поняття та дослідження, можна зробити висновок, що світло відіграє важливу роль у нашому житті, впливаючи на циркадні ритми та біологічний годинник.

### 2.3 Світловий потік, освітленність та колірна температура

Одними з найважливішик параметрів у освітленні є світловий потік, освітленість та колірна температура, через це необхідно детальніше ознайомитися з даними поняттями.

**Світловий потік** – кількість світла, яке випромінюється за одиницю часу [9]. Одиницею виміру в міжнародній системі є Люмен (Лм). Величина світлового потоку

залежить від потужності джерела світла, але не визначається лише нею, адже на дану величину впливають ще й інші фактори. Наприклад, у лампах накалювання це може бути матеріал нитки накалювання та її температура.

Знаючи світловий потік та площу приміщення, що необхідно освітити, можна вирахувати інше важливе значення, таке як освітленість.

**Освітленість** – освітлення поверхні, що створюється світловим потоком, який падає на поверхню [3]. Одиницею виміру в міжнародній системі є Люкс (Лк), який рівняється 1 Люмену на квадратний метр. Приладом для виміру даної величини є люксметр. Для освітленості приміщень існують офіційні норми, які регламентують оптимальну кількість освітленості. У таблиці 2.1 наведені норми для деяких основних приміщень.

Таблиця 2.1. Норми освітленості

| **Тип приміщення** | **Норма освітленості** |
|---|---|
| Житлові кімнати | 150 |
| Кухні | 150 |
| Кабінети, бібліотеки | 300 |
| Коридори, ванни | 50 |
| Гардеробні | 75 |

Кожне світло має свою певну температуру, завдяки якій ми так чи інакше сприймаємо освітлення.

**Колірна температура** – характеристика розподілу інтенсивності випромінення джерела світла як функція довжини хвилі в оптичному діапазоні [8]. Температура світла вимірюється у Кельвінах(К), при чому чим вища температура, тим більш холодний відтінок має світло. На таблиці 2.2 та рисунку 1.1 представлено відповідність колірної температури до її відтінку.

Таблиця 2.2. Відповідність колірної температури її відтінку

| Колірна температура | Відтінок |
|---|---|
| 2500–3000 | Теплий оранжевий |
| 3000–4000 | Теплий жовтуватий |
| 4000–5000 | Нейтральний білий |
| 5000–6500 | Голубуватий |

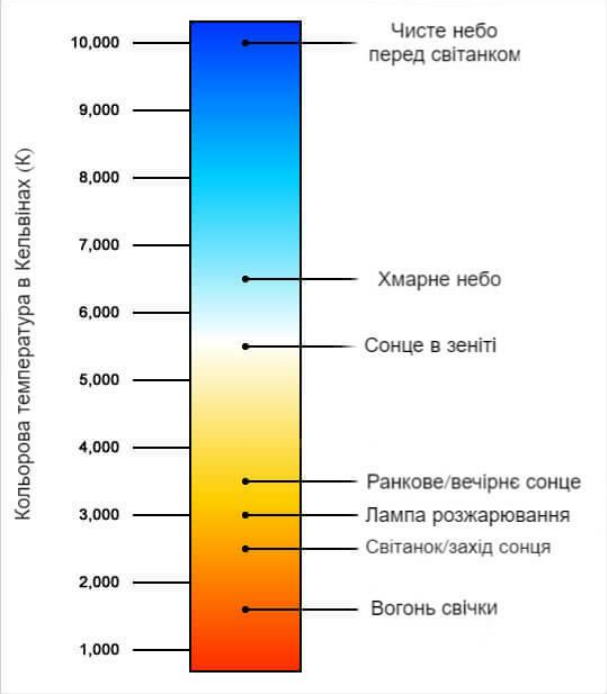


Рисунок 1.1 – Залежність відтінку світла від колірної температури

Колірна температура має сильний вплив на людину, зокрема на її психічний стан. Наприклад, теплі оранжеві та жовті відтінки краще всього сприймаються зранку, оскільки вони сприяють легкому пробудженню, налаштовують на позитивний лад і стимулюють діяльність. Також ці відтінки добре використовувати у вечірній час, через їх заспокійливий ефект [19].

Для приміщень, у який проводиться велика кількість часу, підходить денне світло. Такі відтінки найбільш відповідають денному сонячному світлі, тому організм сприймає таке освітлення як сигнал до активної діяльності.

Лампи з високою колірною температурою не бажано використовувати довгий час, так як вони мають сильний вплив на психіку людини. При короткочасному використанні таке світло стимулює організм. А при довгостроковому використання можливий зворотній ефект – погане самопочуття, зменшення активності, депресія [16].

При низькому рівні освітленості, тобто при тепловому світлі, людина краще почуває себе, це найбільш комфортна температура для людини. Якщо освітленість буде високою, то з'явиться дискомфорт і біль в очах. І навпаки, якщо освітленість низька то світло буде сприйматися тривожно.

### 2.4 Людино-орієнтоване освітлення

Орієнтоване на людину освітлення (Human Centric Lighting) – це концепція освітлення, яка зосереджує увагу на поверненні динаміки природного денного світла у повсякденне життя людей за допомогою біологічно ефективного штучного освітлення. HCL виходить за рамки основних візуальних потреб людей. Це допомагає людському тілу підтримувати природні циркадні ритми, якими людина обумовлювала свою історію. До 200 років тому наші предки проводили 90% свого часу на сонці і синхронізували свої біологічні годинники з 24-годинним ритмом змін навколишнього середовища Землі. У наш час люди вдень проводять обмежений час на відкритому повітрі. І таким чином вплив природного денного світла різко зменшується. Невідповідний вплив штучного світла може призвести до циркадних розладів.

Замість того, щоб забезпечити неадаптивне освітлення фіксованим оптичним випромінюванням, незалежно від активності та часу доби, освітлення, орієнтоване на людину, імітує динамічні зміни інтенсивності світла та колірної температури зовнішнього світла протягом дня. HCL забезпечує біологічно оптимізовані композиції світла для підвищення концентрації та продуктивності праці в денному робочому та навчальному середовищі, водночас стимулюючи розслаблення та регенерацію протягом ночі. Цілісна конструкція біологічно ефективних систем освітлення враховує вплив світлового впливу як на біологічні, так і на емоційні

аспекти людської фізіології та одночасно забезпечує візуально комфортне та підтримуюче освітлення для задоволення функціональних потреб.

Орієнтоване на людину освітлення спрямоване на пом'якшення фізіологічних та психологічних наслідків, спричинених розбіжностями між природним та штучним світлом, а також забезпечує покращене візуальне середовище та покращує працездатність людей. Концепція HCL вважається невід'ємною частиною дизайну освітлення для всіх середовищ, де люди живуть, працюють та навчаються.

Медичні установи мають чітку необхідність впроваджувати HCL для створення позитивного середовища пацієнта, сприяння одужанню пацієнтів, вирішення проблем зі сном пацієнтів, лікування та/або запобігання симптомам депресії.

Оскільки виявлено, що світло з різними спектрами та інтенсивністю підтримує концентрацію, зменшує втому та покращує когнітивні показники, навчальні заклади найкраще підходять для розгортання систем HCL, які сприяють навчальному середовищу, пристосованому до різних видів діяльності.

Також варто застосовувати циркадне освітлення на виробничих, офісних приміщеннях та конференц-залах, де широко використовується блакитне світло залиті багатим блакитним світлом для підвищення мотивації, прихильності та продуктивності праці персоналу чи робітників, гнучке регулювання спектрального складу та інтенсивності світла протягом робочого дня може покращити задоволеність та ефективність роботи працівників.

### 2.5 Переваги та недоліки світлодіодів

Світовий ринок освітлення зазнав радикальної трансформації, зумовленої все більшим впровадженням світлодіодних технологій (LED). Ця революція твердотільного освітлення (SSL) принципово змінила основну економіку ринку та динаміку галузі. Технологія забезпечила не тільки різні форми продуктивності, але перехід від звичайних технологій до світлодіодного освітлення глибоко змінює спосіб мислення людей про освітлення. Звичайні технології освітлення були розроблені головним чином для задоволення візуальних потреб. За допомогою

світлодіодного освітлення позитивна стимуляція біологічного впливу світла на здоров'я та добробут людей привертає все більшу увагу. Поява світлодіодних технологій також відкрила шлях для зближення між освітленням та Інтернетом речей, що відкриває цілий новий світ можливостей.

Світлодіод – це напівпровідниковий пристрій, що містить світлодіодну матрицю (мікросхему) та інші компоненти, що забезпечують механічну підтримку, електричне з'єднання, теплопровідність, оптичне регулювання та перетворення довжини хвилі. Взагалі, світлодіод містить світлодіодний чіп (або взаємозамінно відомий як світлодіодний штамп), розміщений в упаковці. Світлодіодна мікросхема або світлодіодна плашка в основному являє собою двотермінальний діод, що складається з анода (+) і катода (-). Він розроблений з цільного шматка напівпровідникової пластини, тонкого диска, як правило, виготовленого з сапфіру або кремнію. Напівпровідникова пластинка осідає позитивно зарядженим (P-тип) шаром і негативно зарядженим (N-типом) шаром за допомогою процесу, званого металорганічним хімічним осадженням парів (MOCVD), який також відомий як металоорганічна парофазна епітаксія (OMVPE). Позитивний та негативний шари, які діють як анод діода (+) та катод (-), як правило, складаються з напівпровідників III-V (елементи III та V групи Періодичної системи), таких як нітрид галію (GaN), галій фосфід (GaP), арсенід галію (GaAs) та фосфід індію (InP). Напівпровідниковий шар типу P легований акцепторними домішками, які мають менше електронів у валентній зоні, ніж напівпровідниковий матеріал, який вони замінюють у власній напівпровідниковій решітці. Напівпровідниковий шар N-типу легований донорними домішками, які мають надлишок електронів в зоні провідності.

Межа переходу, де стикаються позитивний і негативний шари, називається «областю виснаження», широко відомою як «p-n перехід». Коли утворюється p-n-перехід, частина вільних електронів із шару N-типу починає мігрувати по новоутвореному переходу, щоб заповнити отвори акцепторних домішок у шарі P-типу. Дифузія електронів і дірок утворює негативні іони і, таким чином, створює зворотне електричне поле між позитивною і негативною сторонами. Електричне поле створює область виснаження, яка обмежує подальшу дифузію носіїв заряду,

якщо по всій області виснаження не подається пряма напруга, яка є достатньо великою для подолання зворотного електричного поля. Коли діод зміщений вперед (увімкнено), електрони «стрибають» через p-n-перехід, щоб рекомбінувати з дірками і потрапляти в стан з меншою енергією. Надлишок енергії виділяється у вигляді фотона, який, по суті, є пакетом електромагнітного випромінювання у видимому діапазоні спектра (світла).

Довжина хвилі світла, що випромінюється під час електролюмінесценції, може залежати від енергії зазору, яка є різницею в енергії між зоною провідності та валентною зоною. Взагалі для світлодіодів потрібен напівпровідник із прямою зонною, що дозволяє ефективніше випромінювати рекомбінації електронно-дірочних пар, ніж напівпровідники із непрямою зоною. Епітаксійні шари світлодіодів зазвичай виготовляються з кристалів на основі галію, таких як GaN та GaP. Світлодіоди, виготовлені з напівпровідників із прямим зазором, згруповані в дві групи: сімейство нітридів та сімейство фосфідів. Світлодіоди сімейства нітридів, такі як композиції нітриду, галію, індію (InGaN), мають більший зазор між смугами між позитивним та негативним шарами і, отже, виробляють світло на коротших хвилях (синій та зелений) частин видимого спектра. Світлодіоди сімейства фосфідів, такі як композиції фосфіду, алюмінію, фосфіду, індію, галію (InGaAIP), мають невеликий зазор в енергетичній смузі між шарами P-типу та N-типу і, отже, виробляють випромінювання довжини хвилі, видиме як бурштинове або червоне світло [17].

**Енергоефективність**

Однією з головних переваг переходу на світлодіодне освітлення є енергоефективність. Світлодіодні лампи потребують у 10 разів менше енергії у порівнянні з лампами розжарення і в 2-3 рази менше, ніж люмінесцентні лампи. За 30 000 годин роботи світлодіодної лампи, еквівалентній лампі розжарення в 100 Вт, вона потребує 400 кВт електроенергії, в той час як лампа розжарення – 3000 кВт.

**Рівномірність освітлення**

Рівномірне освітлення є однією з головних переваг світлодіодного освітлення. Рівномірність – це міра взаємозв’язку освітленості над площею. Хороше освітлення

повинне забезпечувати рівномірний розподіл люменів, що падають на поверхню. Різниця яскравості, спричинена нерівномірним освітленням, може призвести до стомлення зору та інших проблем.

**Спектральна інженерія**

Технологія LED пропонує нову можливість контролю спектрального розподілу потужності джерела світла (SPD), що означає, що склад світла може бути адаптований до різних застосувань. Спектральна керованість дозволяє розробити спектр від освітлювальних приладів для залучення конкретних людських візуальних, фізіологічних, психологічних, рослинних фоторецепторів або навіть напівпровідникових детекторів (тобто HD-камери). Високої спектральної ефективності можна досягти за рахунок збільшення бажаної довжини хвилі і видалення або зменшення непотрібних частин спектра.

**Швидке вмикання/вимкнення**

Світлодіоди вмикаються майже миттєво (кілька десятків наносекунд) і мають час вимкнення в десятки наносекунд. Тоді як час розігріву або час, за який лампочка досягає повної світлової потужності, компактних люмінесцентних ламп може тривати до 3 хвилин. Світлодіоди вмикаються на 140–200 мілісекунд швидше, ніж лампи розжарення. Ще однією перевагою світлодіодів в режимі вмикання / вимикання є цикл перемикання. На тривалість життя світлодіодів не впливає часте вмикання / вимикання, з іншого боку, часте вмикання/вимикання може скоротити термін служби ламп розжарення та люмінесцентних ламп. Ці джерела світла, як правило, мають лише кілька тисяч циклів перемикання протягом їх номінального терміну служби.

**Можливість затемнення**

Здатність динамічно випромінювати світло дозволяє світлодіодам ідеально керувати затемненням, тоді як флуоресцентні лампи погано реагують на затемнення. Затемнення люмінесцентних ламп вимагає використання дорогих, великих та

складних схем для підтримки умов збудження та напруги газу. Галогенні та натрієві лампи високого тиску не можуть бути затемнені нижче 50% від номінальної потужності. Вони також реагують на сигнали затемнення значно повільніше, ніж світлодіоди. Затемнення світлодіодів може бути здійснено або за допомогою постійного зменшення струму (CCR), яке більш відоме як аналогове затемнення, або за допомогою широтно-імпульсної модуляції (ШІМ), відомого як цифрове затемнення. Аналогове затемнення контролює силу струму на світлодіодах. Це найбільш широко застосовуване рішення затемнення для загального освітлення, хоча світлодіоди можуть погано працювати при дуже низьких струмах (нижче 10%). ШІМ-затемнення варіює робочий цикл широтно-імпульсної модуляції, щоб створити середнє значення на його виході в повному діапазоні від 100% до 0%. Управління затемненням світлодіодів дозволяє узгодити освітлення з людськими потребами, максимізувати економію енергії, включити змішування кольорів та продовжити термін служби світлодіодів.

**Керованість**

Цифрова природа світлодіодів спрощує інтеграцію датчиків, процесорів, контролера та мережевих інтерфейсів в освітлювальній системі для реалізації різних інтелектуальних стратегій освітлення, такі як динамічне та адаптивне освітлення. Динамічний аспект світлодіодного освітлення варіюється від простих кольорових змін до складних світлових шоу на сотнях або тисячах індивідуально керованих освітлювальних вузлів. Керування освітленням у мережі на основі IP дозволяє системам освітлення взаємодіяти з іншими пристроями в мережах IoT. Управління світлодіодними системами освітлення може бути реалізоване за допомогою різноманітних протоколів дротового та бездротового зв’язку.

**Довговічність**

Світлодіод випромінює світло з напівпровідникового блоку, а не зі скляної колби або трубки, як це відбувається у застарілих лампах, де використовуються нитки розжарювання або гази для генерування світла. У світлодіодів відсутнє

крихке скло, рухомі частини та нитка розжарення, тому світлодіодні системи освітлення надзвичайно стійкі до ударів, вібрацій та зносу.

**Термін служби**

Тривалий термін служби виокремлюється як одна з головних переваг світлодіодного освітлення, але твердження про тривалий термін експлуатації, що ґрунтуються виключно на метриці ресурсу для світлодіодного пакета (джерела світла), можуть ввести в оману. Час корисного використання світлодіодного пристрою часто називають моментом часу, коли світловий потік зменшився до 70% від його початкового виходу, або L70. Як правило, світлодіоди мають термін служби L70 від 30000 до 100000 годин (при T = 85°C).

**Вплив на навколишнє середовище**

Світлодіодне освітлення має помітно менший вплив на довкілля, ніж традиційні джерела освітлення. Низьке споживання енергії означає низькі викиди вуглецю. Світлодіоди не містять ртуті і, таким чином, створюють менше екологічних ускладнень під час утилізації. Для порівняння, утилізація ртутних флуоресцентних ламп передбачає використання суворих протоколів щодо утилізації відходів.

Як ми бачимо, світлодіодне освітлення має багато переваг над застарілими видами освітлення, але воно також має певні недоліки.

Найбільш відомим недоліком світлодіодного освітлення є те, що світлодіоди виробляють багато тепла і не випромінюють тепло у вигляді інфрачервоної енергії. Приблизно половина електричної енергії, що подається на світлодіод, перетворюється в тепло, що скорочує час роботи світлодіода на 30-50% за кожні 10С підвищення температури.

Найбільша слабкість світлодіодного освітлення полягає в тому, що світлодіоди надзвичайно вимогливі до струму. Струм надає системам освітлення чудову керованість, але навіть незначна зміна струму призведе до коливання світлової потужності. Світлодіоди - це пристрої, що керуються постійним струмом,

однак їх часто доводиться живити джерелом змінного струму, що може призвести до залишкової пульсації струму, що надходить від драйвера до світлодіодів. Ця пульсація змушує світлодіоди мерехтіти з подвоєною частотою 100 Гц або 120 Гц. З іншого боку, чим більший струм, тим більше тепла утворюється в напівпровідниковій матриці.

І останнє, але не менш важливе, збільшення складності системи призводить до більших витрат на виробництво, в порівнянні зі старими освітлювальними приладами.

### 2.6 Огляд існуючих рішень

Перед початком розробки власного освітлення необхідно дослідити існуючі рішення розумного освітлення, які є популярними на даний момент.

Розумні лампи – це лампи, які забезпечуються додатковими сенсорами і модулями. Зазвичай до розумних освітлювальними приладами відносять LED-світильники і лампи. Загалом розумна лампа складається з пластикового або алюмінієвого корпусу з ребрами для охолодження, плати зі світлодіодами, драйвера для живлення світлодіодів, плати керування з мікро контролером та з приймача сигналів. У схему входить набір датчиків, серед яких може бути датчик освітленості, руху, присутності, таймера.

Під час дослідження розумних ламп з’ясували, що їх є багато варіантів із різним функціоналом, ціною та призначенням. Є лампи, що керуються лише пультом керування, та лампи, якими можна керувати за допомогою мобільного застосунку або голосового помічника.

Найпопулярнішими протоколами керування є WiFi та Bluetooth. Наприклад, за допомогою WiFi можна керувати з будь-якої точки світу, при умові доступності інтернету та інтегрувати їх в систему розумного будинку, в той час як Bluetooth працює швидше, коштує дешевше та дозволяє виводити звук із пристрою, якщо лампа має динамік, тому дорожчі пристрої працюють за допомогою двох протоколів. Також деякі лами містять різні датчики, такі як датчики руху,

освітленості та руху, які дозволяють лампі налаштовуватися під середовище самостійно.

Одним із найбільш важливих факторів є кут розсіювання. Так, кут розсіювання менше 30° забезпечує точкове освітлення, тому для використання лампи як основної, необхідно щоб кут був більше 90°.

Після проведення дослідження вирішено вибрати весь функціонал з переглянутих ламп і при розробці власної виділити ті, які будуть найбільш корисні у власному проекті. Список усього функціоналу, який може бути присутній у розумних лампах:

1. Налаштування яскравості, теплоти та інтенсивності світіння
2. Налаштування кольору
3. Наявність акумулятора
4. Голосове керування
5. Автоматична адаптація яскравості
6. Динаміки та світіння в такт музики
7. Будильник
8. Графік роботи
9. Об’єднання ламп в мережу

Опираючись на описані вище фактори та дослідивши представлені на нашому ринку варіанти, для дослідження обрано наступні моделі:

– Xiaomi Mi LED Smart Bulb
– Philips Hue
– WIZ Smart LED
– Philips Xiaomi Smart LED Zhirui

Порівняльні характеристики обраних моделей наведено в таблиці 2.3.

Таблиця 2.3. Порівняння існуючих рішень

| **Властивості** | **Xiaomi Mi LED** | **Philips Hue** | **WIZ Smart LED** | **Philips Xiaomi** |
|---|---|---|---|---|

| | **Smart Bulb** | | | **Smart LED Zhirui** |
|---|---|---|---|---|
| **Потужність, Вт** | 10 | 8.5 | 9 | 6.5 |
| **Колір** | Різнокольрова | Різнокольоровий | Білий | Білий |
| **Кут розсіювання°** | 360 | 160 | 200 | 200 |
| **Температура, К** | 1700-6500 | 2200-6500 | 2500-6500 | 3000-5700 |
| **Світловий потік, Лм** | 600 | 600 | 810 | 450 |
| **Протокол зв'язку** | WiFi | WiFi | WiFi | WiFi |

# 3. ЗАСОБИ РОЗРОБКИ

## 3.1 Опис системи та необхідних технологій

Завданням роботи є розроблення системи інтелектуального освітлення приміщення із врахуванням потреб користувача. Беручи до уваги проведені дослідження, система буде мати наступні функції:

1. Налаштування яскравості та колірної температури
2. Налаштування кольору
3. Автоматична адаптація яскравості
4. Налаштування графіку увімкнення та вимкнення
5. Налаштування будильника
6. Містити режими за замовчунням освітлення для певної діяльності
7. Відслідковування циркадних ритмів
8. Визначення користувача та застосування його налаштувань

Опираючись на даний функціонал, створено діаграму прецедентів, представлену на рисунку 3.1.

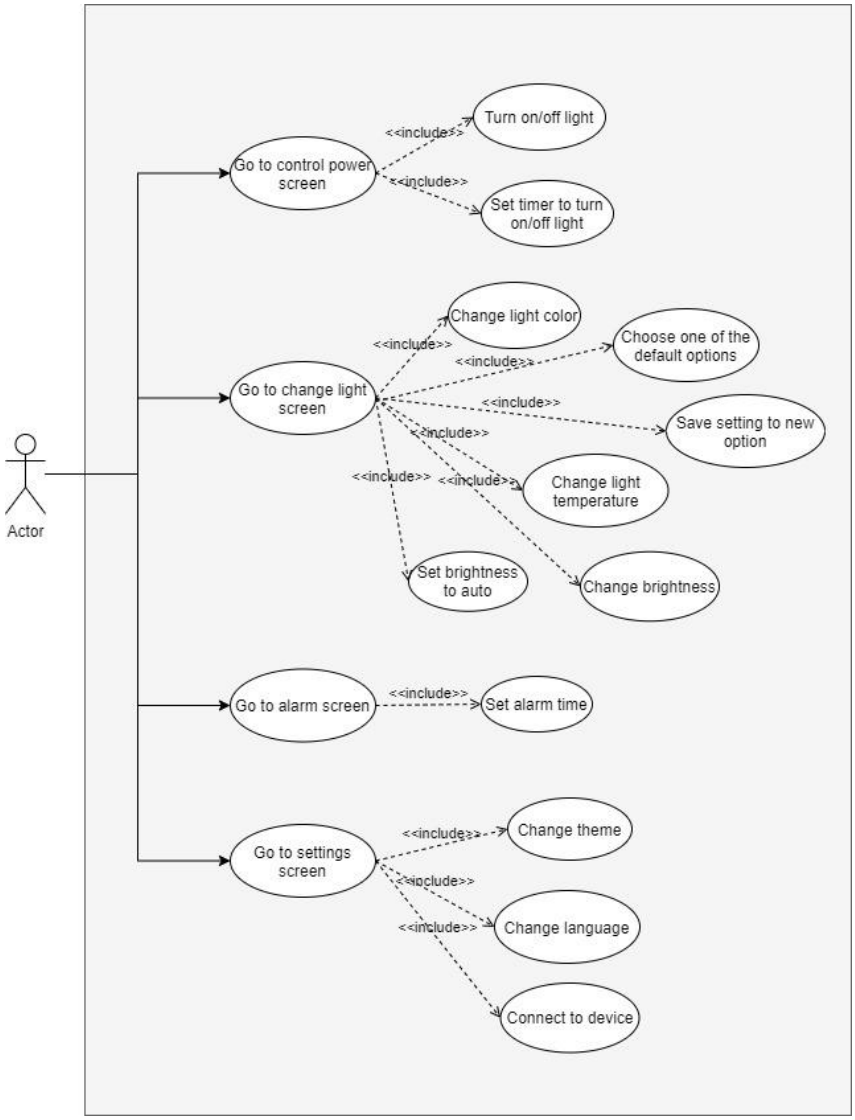


Рисунок 3.1 – Діаграма прецидентів

Після детального аналізу системи та діаграми прецидентів, зрозуміло, що система буде складатися з декількох модулей, які будуть відповідати за певний функціонал та взаємодіяти між собою. Даними модулями є:

1. Сервер бази даних
2. Власний web сервер
3. Мобільний додаток
4. Модуль розумного освітлення

Компоненти системи представлені на діаграмі розгортання (рисунок 3.2), а взаємозв’язки між ними представлено на діаграмі компонентів (рисунок 3.3)

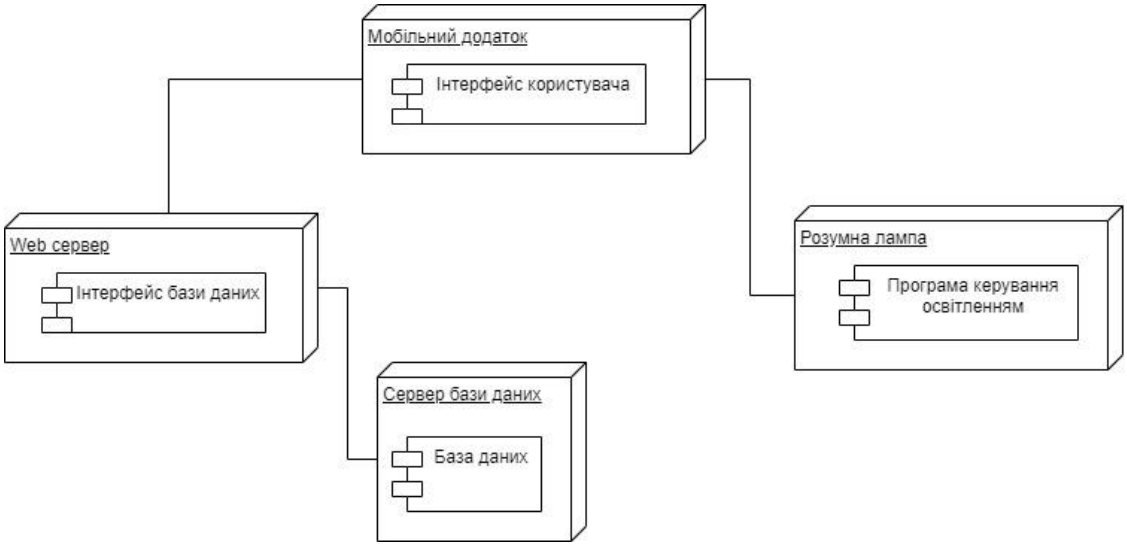


Рисунок 3.2 – Діаграма розгортання

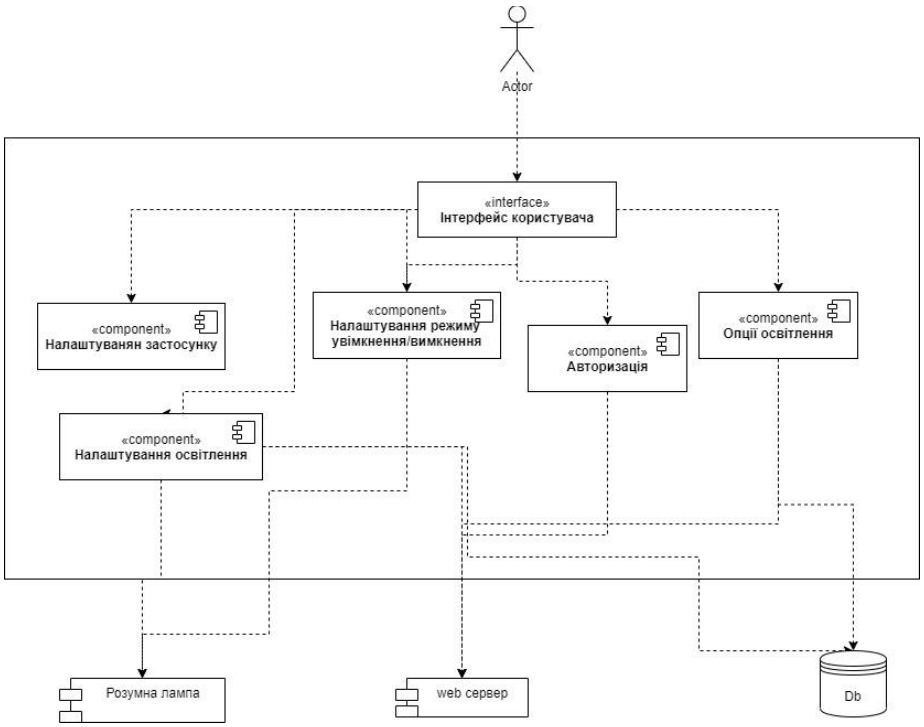


Рисунок 3.3 – Діаграма компонентів

### 3.2 Технології розробки серверної частини

Перед початком розробки серверної частини, необхідно визначитися, яку базу даних використовувати, через це необхідно дослідити оптимальні варіанти та обрати

найкращий. Обирати будемо поміж SQL баз даних, оскільки дизайн схеми передбачає багато зв'язків між таблицями.

База даних – організований набір структурованої інформації або даних які зазвичай зберігаються в електронному вигляді у комп'юній системі [25].

**MySQL** – одна з найпопулярніших баз даних, яку використовують як першу базу для вивчення SQL. Вона має безкоштовні версії, легко встановлюється, підходить для багатьох операційних систем, має багато налаштувань та має відкритий код. Також вона може працювати з іншими базами даних, такими як Oracle та DB2.

До мінусів даної бази даних можна віднести низьку продуктивність в порівнянні з іншими базами, можливість критичних поломок при зупинці сервера та важка зміна структури даних при великій кількості зв'язків.

**PostgreSQL –** одна із перших розроблених баз даних, через це вона сильно розвинена і продовжує розвиватися. До плюсів можна віднести те, що ця база є безкоштовною, підтримується багатьма операційними системами, дозволяє керувати структурованими та не структурованими даними та імпортувати інформацію із інших баз даних. Також дана база легко маштабується, може обробляти терабайти даних та підтримує формат json.

Мінусами бази, що розглядається, є не чітка документація, важка конфігурація, просадка швидкості при читанні.

База даних **Oracle** була розроблена в 70-х роках XX століття і має гарну репутацію. Вона має найновіший функціонал, високу стабільність роботи та може працювати з великою кількістю даних.

Недоліком є те, що база даних не має безкоштовних версій і для її інтеграції в систему необхідно виділити досить багато ресурсів.

**Microsoft SQL сервер** є базою даних розробленою компанією Microsoft. Перевагами є стабільність роботи, простота використання, моніторинг ресурсів, інтеграція з іншими сервісами Microsoft.

Мінусами, як і у випадку з Oracle, є висока ціна, що потребує виділення ресурсів.

Дослідивши всі переваги та недоліки зазначених вище баз даних, вважаємо, що MySQL є оптимальною через її ціну та простоту використання.

Визначившись із базою даних переходимо до визначення мови програмування та фреймворку для написання власного web сервера для взаємодії з іншими модулями системи. Вибір випав на мову Java та фреймворк Spring.

**Java –** об'єктно-орієнтована мова програмування, випущена у 1995 році. Програми написані даною мовою є платформонезалежними. Дана мова підтримує багатопоточність, автоматичне керування пам'яттю [1].

**Spring –** фреймворк, який надає комплексну модель програмування і конфігурації для сучасних програм написаних на основі мови Java. Даний фреймворк надає інверсію керування, підтримує аспектно-орієнтоване програмування, містить багато корисних модулів та може автоконфігурувати проект [5; 24].

### 3.3 Технології розробки мобільного застосунку

Мобільний застосунок вирішено розробляти під операційну систему андроїд.

Операційна система Андроїд – це мобільна операційна система на базі Linux, яка переважно працює на смартфонах і планшетах. Платформа Android включає операційну систему на основі ядра Linux, графічний інтерфейс, веб-браузер та програми для кінцевих користувачів, які можна завантажити [10].

Мовою розробки було обрано мову програмування Dart та фреймворк flutter.

**Dart** – це оптимізована клієнтом мова для розробки швидких додатків на будь-якій платформі. Його мета – запропонувати найпродуктивнішу мову програмування для розробки мультиплатформ у поєднанні з гнучкою платформою виконання для фреймворків програм.

Мови визначаються їх технічною оболонкою – рішеннями, зробленими під час розробки, що формують можливості та сильні сторони мови. Dart розроблений для технічного пакету, який якнайкраще підходить для розробки клієнтів, надаючи пріоритет як розробці (гаряче перезавантаження), так і високоякісному виробничому

досвіду для широкого спектру цілей компіляції (веб, мобільних та настільних програм) [27].

**Flutter** – це безкоштовний фреймворк для мобільного інтерфейсу з відкритим кодом, створений Google і випущений у травні 2017 року. Він дозволяє створити власний мобільний додаток лише з однією кодовою базою. Це означає, що можливо використовувати одну мову програмування та одну базу коду для створення двох різних програм (для iOS та Android і в майбутньому для web та desktop версій) [26]. Flutter має багато переваг, основними з яких є чудова продуктивність, швидка розробка завдяки гарячому перезавантаженню, регулярні оновлення та сумісність з багатьма платформами.

Отже, використання даних технологій дозволить у майбутньому легко розширити систему для використання на інших операційних системах та платформах. Також перед початком розробки створено User Flow – перехід користувачів від одного сценарію взаємодії з інтерфейсом до іншого, очікуваний алгоритм дій користувача, представлений на рисунку 3.4.

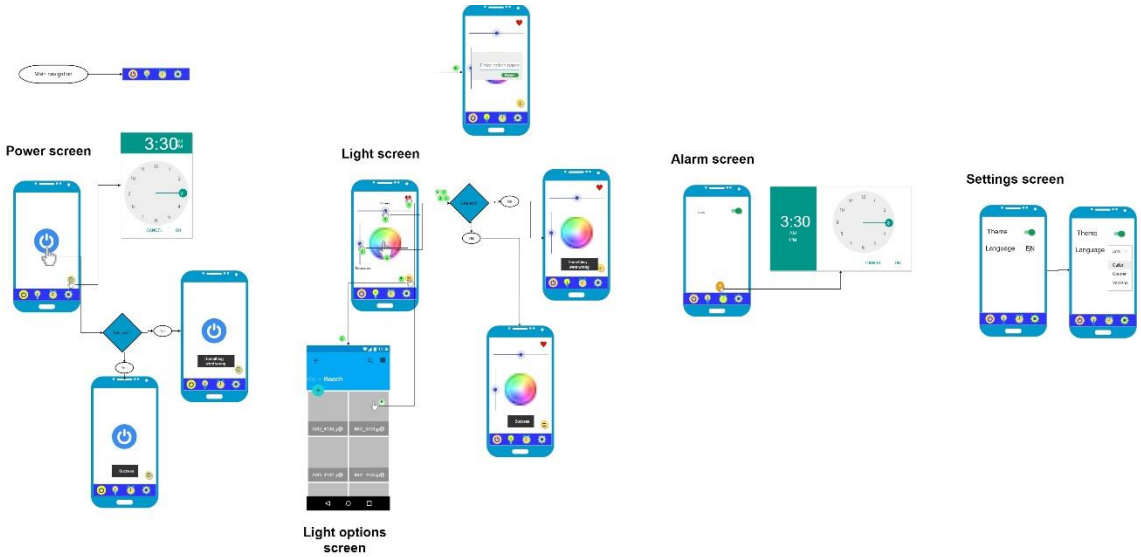


Рисунок 3. 4 – User Flow дизайн

### 3.4 Технології розробки апаратно-програмного комплексу

Розробку апаратно-програмного комплексу розумного освітлення вирішено розробляти на платформі ардуіно.

**Ардуіно** – електронна платформа з відкритим вихідним кодом, яка базується на простому використанні апаратної та програмної частин [15]. Дана платформа є кросплатформеною, не дорога, має просте і зрозуміле середовище програмування, може працювати з великою кількістю датчиків.

Перед розробкою апаратно-програмного комплексу необхідно визначитися із мікроконтролером на якому буде будуватися вся система. Мікроконтролер повинен швидко опрацьовувати команди та мати у собі WiFi та Bluetooth, даним мікроконтролером є ESP32.

ESP32 – це недорогий мікроконтролер на мікросхемі (SoC) від Espressif Systems, розробники знаменитого SoC ESP8266. Він є наступником SoC ESP8266 і представлений як в одноядерних, так і в двоядерних варіаціях 32-розрядного мікропроцесора Xtensa LX6 від Tensilica з вбудованим Wi-Fi та Bluetooth.

ESP32, як і ESP8266, містить вбудовані радіочастотні компоненти, такі як підсилювач потужності, підсилювач з низьким рівнем шуму, антенний комутатор, фільтри та радіочастотний сигнал. Це робить проектування апаратного забезпечення навколо ESP32 дуже простим, оскільки вам потрібно дуже мало зовнішніх компонентів.

Інша важлива річ, яку слід знати про ESP32, – це те, що він виготовлений за технологією TSMC ультра низької потужності 40 нм. Отже, розробляти додатки, що працюють від акумулятора, такі як носимі пристрої, аудіообладнання, розумні годинники тощо, за допомогою ESP32 має бути нескладно [14].

Характеристики мікроконтролера:

- Одно- або двоядерний 32-розрядний мікропроцесор LX6 з тактовою частотою до 240 МГц4;
- 520 КБ SRAM, 448 КБ ПЗУ і 16 КБ RTC SRAM.
- підтримка підключення Wi-Fi 802.11 зі швидкістю до 150 Мбіт / с.;
- підтримка класичних специфікацій Bluetooth v4.2 та BLE;
- 34 програмованих GPIO;
- до 18 каналів 12-бітного АЦП SAR та 2 канали 8-бітного ЦАП;
- послідовне підключення включає 4 x SPI, 2 x I2C, 2 x I2S, 3 x UART;

- Ethernet MAC для фізичного зв'язку в локальній мережі;
- 1 контролер хосту для SD / SDIO / MMC та 1 контролер веденого пристрою для SDIO / SPI;
- ШІМ двигуна та до 16 каналів світлодіодного ШІМ;
- безпечне завантаження та шифрування Flash;
- криптографічне апаратне прискорення для AES, Hash (SHA-2), RSA, ECC та RNG.

Блок діаграма та розпіновка представлені на рисунках 3.5 та 3.6.

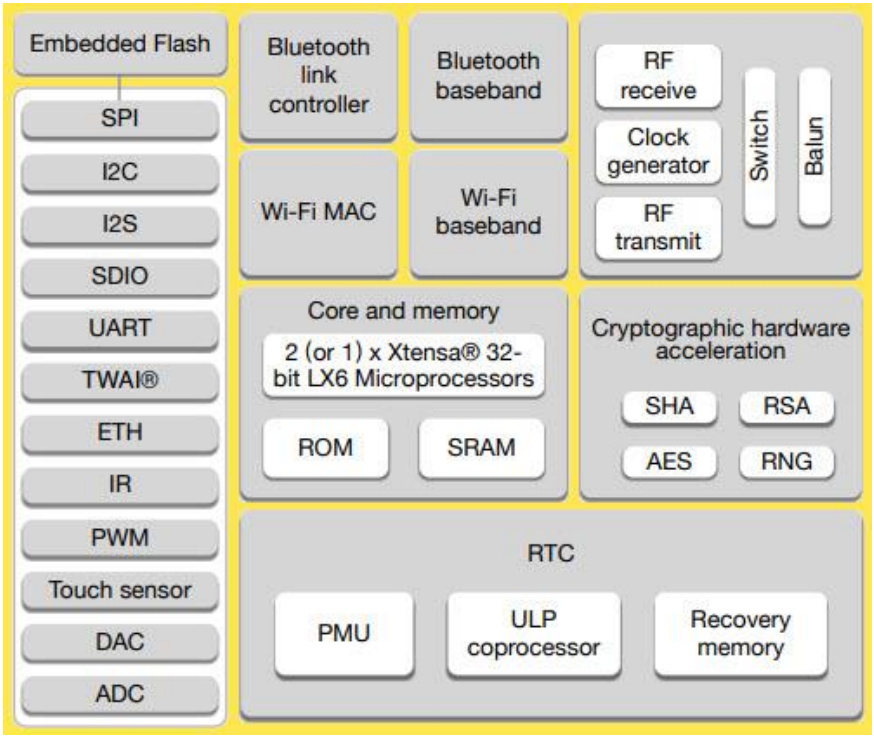


Рисунок 3.5 – Блок діаграма ESP32

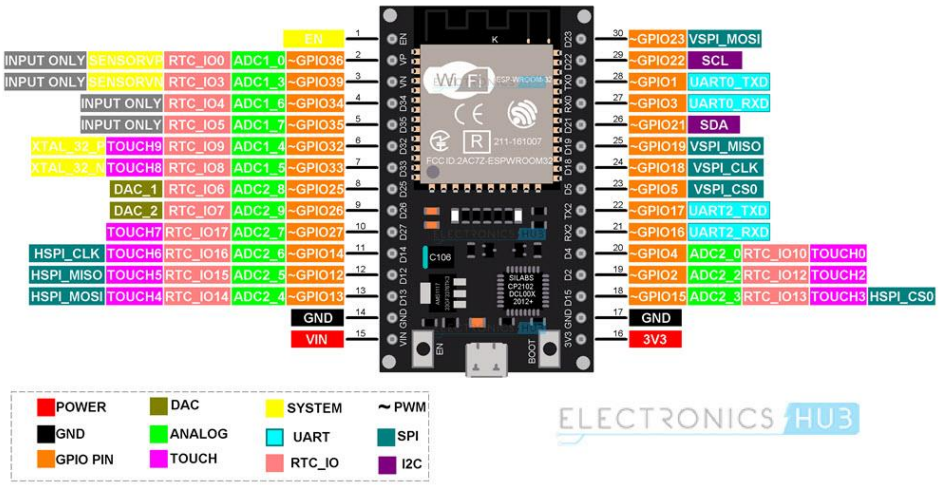

Рисунок 3.6 – Розпіновка ESP32

Отже, обраний мікроконтролер повністю відповідає поставленій задачі через це переходимо до вибору наступних модулів.

**Часи реального часу**

Оскільки наше інтелектуальне освітлення має функції будильника та таймера, то необхідно мати модуль часів реального часу. Було обрано модуль DS3231, який є недорогим, точним годинником реального часу із інтегрованим температурно-компенсованим кристалогенератором та кристалом. Пристрій має вхідний акумулятор і підтримує його точне вимірювання часу під час основного живлення пристрою переривається. Інтеграція кристалічного резонатора також підвищує довгострокову точність пристрою. DS3231 зберігає секунди, хвилини, години, день, дату, інформацію про місяць та рік. Дата в кінці місяця автоматично коригується на місяці з меншою кількістю більше 31 дня, включаючи виправлення для високосного року. Годинник працює у 24-годинному або 12-годинному форматі. Точна опорна напруга з компенсацією температури і схема порівняння контролює стан до виявлення збоїв у живленні, забезпечуючи скидання вихідних даних і автоматичне перемикання на резервне живлення, коли це необхідно [16].

Характеристики модуля:

– календар до 2100 року;
– похибка +- 4 ppm ;
– два програмованих будильника;
– робоча напруга живлення від 3.0в до 5.5в;
– точність внутрішнього цифрового датчика температури ± 3°с;
– дуже мале споживання від резервного джерела.

**Світлодіоди**

Джерелом освітлення обрано адресні світлодіоди ws2812b. Дані світлодіоди було обрано через наступні характеристики:

- розмір світлодіода – 5 х 5 мм4

- частота шім – 400 Гц;
- швидкість передачі даних – 800 кГц
- розмір даних – 24 біта на світлодіод;
- напруга харчування – 5 вольт;
- споживання при нульовій яскравості – 1 мА на світлодіод;
- споживання при максимальній яскравості – 60 мА на світлодіод;
- кольоровість: rgb, 256 відтінків на канал, 16 мільйонів кольорів

# 4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

## 4.1 Опис програмної реалізації серверної частини

Серверна частина, написана з використанням мови Java та фреймворку Spring boot відповідає за реєстрацію, авторизацію та збереження режимів користувача. Структура проекту оформлена згідно шаблону проектування MVC. У кожному класі, що містить більше 6 полів реалізовано породжувальний шаблон проектування будівник. Серверна частина побудована за архітектурою REST.

REST або REpresentational State Transfer – це архітектурний стиль забезпечення стандартів між комп'ютерними системами в Інтернеті, що полегшує взаємодію систем між собою. Системи, сумісні з REST, які часто називають системами RESTful, характеризуються тим, що вони не мають стану та відокремлюють проблеми клієнта та сервера.

В архітектурному стилі REST реалізація клієнта та реалізація сервера можуть здійснюватися незалежно, не знаючи один про одного. Це означає, що код на стороні клієнта можна змінити в будь-який час, не впливаючи на роботу сервера, а код на стороні сервера можна змінити, не впливаючи на роботу клієнта [28].

Поки кожна сторона знає, який формат повідомлень надсилати іншій, вони можуть бути модульними та окремими. Відокремлюючи проблеми користувальницького інтерфейсу від проблем зберігання даних, покращується гнучкість інтерфейсу між платформами та масштабованість, спрощуючи компоненти сервера. Крім того, розподіл дозволяє кожному компоненту розвиватися

незалежно. Використовуючи інтерфейс REST, різні клієнти потрапляють в однакові кінцеві точки REST, виконують однакові дії та отримують однакові відповіді.

Системи, які дотримуються парадигми REST, не мають стану, це означає, що серверу не потрібно нічого знати про те, в якому стані знаходиться клієнт, і навпаки. Таким чином, і сервер, і клієнт можуть зрозуміти будь-яке отримане повідомлення, навіть не бачачи попередніх повідомлень. Це обмеження забезпечується використанням ресурсів, а не команд. Ресурси описують будь-який об'єкт, документ чи речі, які вам можуть знадобитися для зберігання або надсилання іншим службам.

Оскільки системи REST взаємодіють за допомогою стандартних операцій над ресурсами, вони не покладаються на реалізацію інтерфейсів.

Ці обмеження допомагають додаткам RESTful досягти надійності, швидкої продуктивності та масштабованості як компонентів, якими можна керувати, оновлювати та використовувати повторно, не впливаючи на систему в цілому, навіть під час роботи системи.

REST вимагає від клієнта зробити запит на сервер для отримання або модифікації даних на сервері. Запит зазвичай складається з: дієслово HTTP, яке визначає, яку операцію виконувати, заголовок, що дозволяє клієнту передавати інформацію про запит, шлях до ресурсу, додаткове тіло повідомлення, що містить дані. Приклад REST запиту на веб сервер для отримання збережених режимів користувача представлено на рисунку 4.1.


Рисунок 4.1– Приклад REST запиту

На даному прикладі видно, що за допомогою команди curl надсилається GET запит на отримання даних і в параметрі header запиту передається токен і в результаті в якості відповіді отримуємо дані з сервера. Якщо даний параметр не

передавати, або він є неправильним, то отримаємо наступний результат – рисунок 4.1. На даному результаті видно код та повідомлення помилки.

```
C:\Users\sasha>curl -X GET "http://localhost:5624/smart-light/users/options"
{
  "timestamp" : "2021-06-06T14:40:03.027+00:00",
  "status" : 403,
  "error" : "Forbidden",
  "message" : "",
  "path" : "/smart-light/users/options"
}
```

Рисунок 4.2 – Приклад запиту з некоректним параметром

Для даного функціоналу було створено класи, представлені на UML діаграмі (рисунок 4.3).

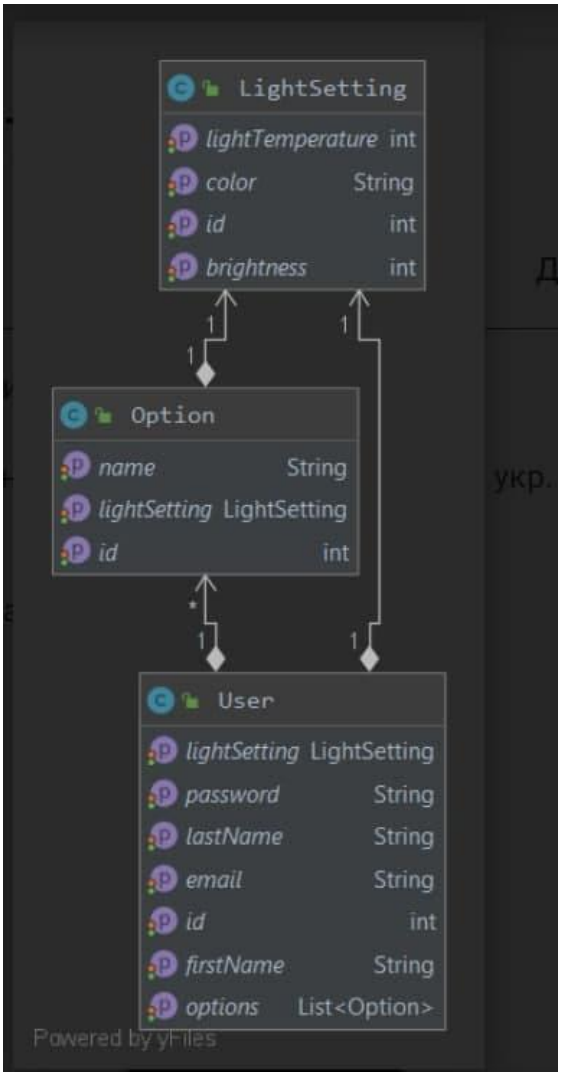


Рисунок 4.3 – UML діаграма

Для доступу застосунку до бази даних, використано бібліотеку Hibernate. Дана бібліотека призначена для вирішення завдань об’єктно-реляційного відображення (ORM). Після створення бази даних та необхідних таблиць, маємо структуру представлено на рисунку 4.4.

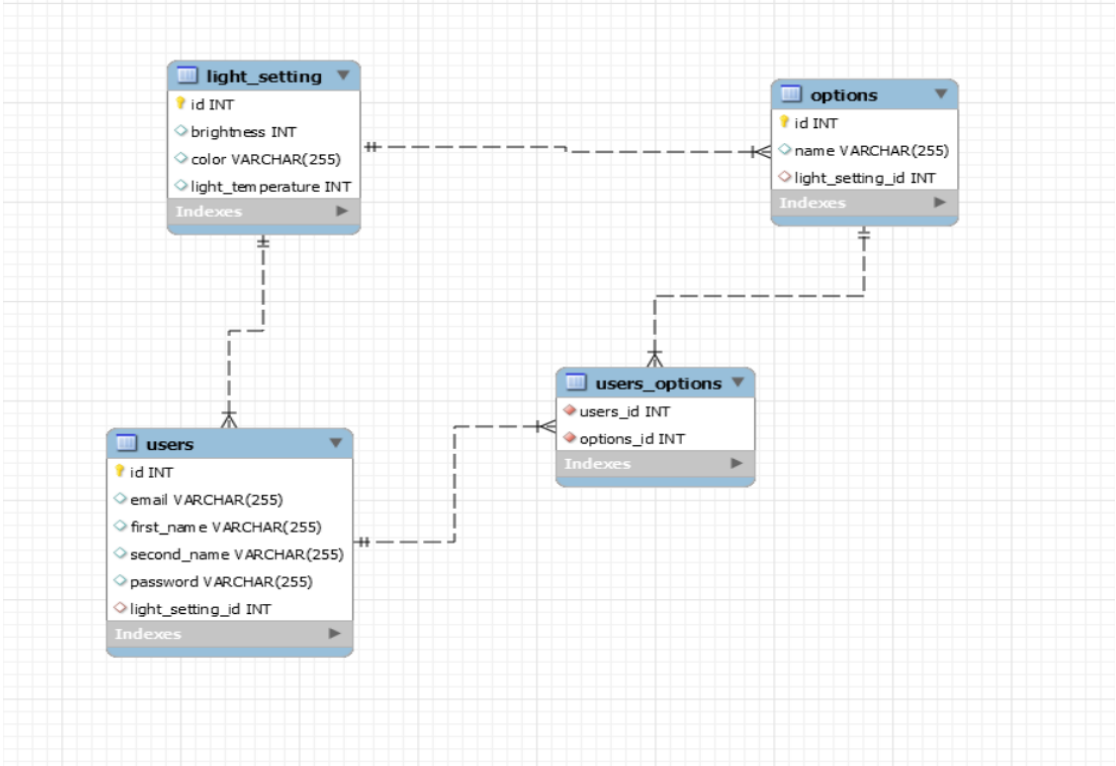


Рисунок 4.4 – Структура бази даних

Для авторизації та аутентифікації користувача використовується JWT токен. JSON Web Token (JWT) – це відкритий стандарт RFC 7519, який використовують для компактної, автономної і безпечної передачі даних у вигляді об’єкта JSON.

JWT складається з трьох блоків, між якими ставлять крапку: заголовок (header), поля (payload) і підпис (signature). Header і payload формуються окремо, потім на їх основі обчислюється signature [4]. Після того як відбулася аутентифікація, сервер відправляє JWT токен користувачеві і користувач надалі має передевати цей токен в header при кожному запиті, адже завдяки йому отримуються необхідні дані користувача відбувається авторизація, тобто перевірка доступу користувача до тих чи інших ресурсів. Якщо токен відсутній або його час валідності

закінчився, то користувач має пройти аутентифікацію. Даний підхід дає більшу захищеність застосунку, адже для генерації токену необхідно знати кодове слово, яке зберігається на сервері.

Також для специфікації RESTful API використано фреймворк Swagger. Swagger дає можливість не лише переглядати специфікацію, але і надіслати запити через інтерфейс. Ще однією перевагою даного фреймворку є те, що він генерує специфікацію автоматично, тобто при зміні одного чи іншого ендпоінта та його параметрів, специфікація оновиться. Специфікація розробленого застосунку представлена на рисунку 4.5.

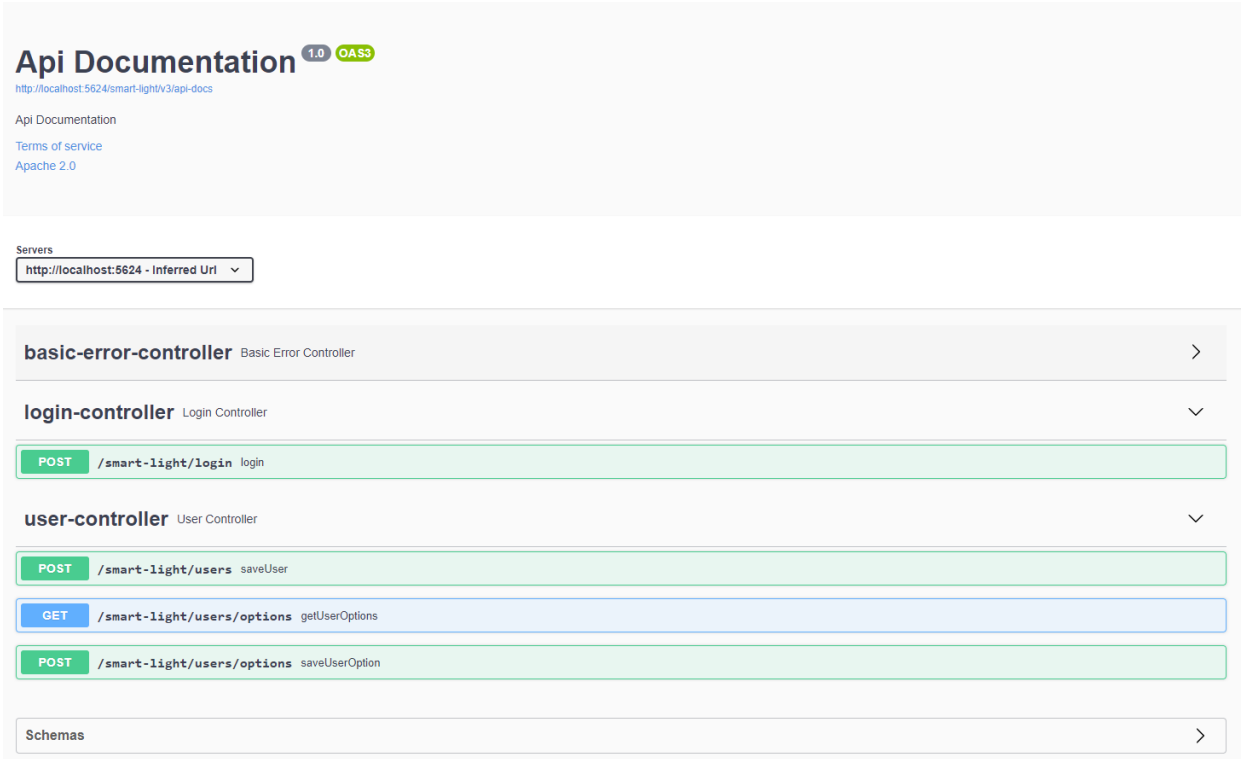


Рисунок 4.5 – Специфікація веб сервера

## 4.2 Опис програмної реалізації апаратно-програмного комплексу

Із обраних модулів створено наступну схему інтелектуального освітлення (рисунок 4.6). Для написання програми використовувалося середовище розробки Arduino Ide та мова програмування C/C++.

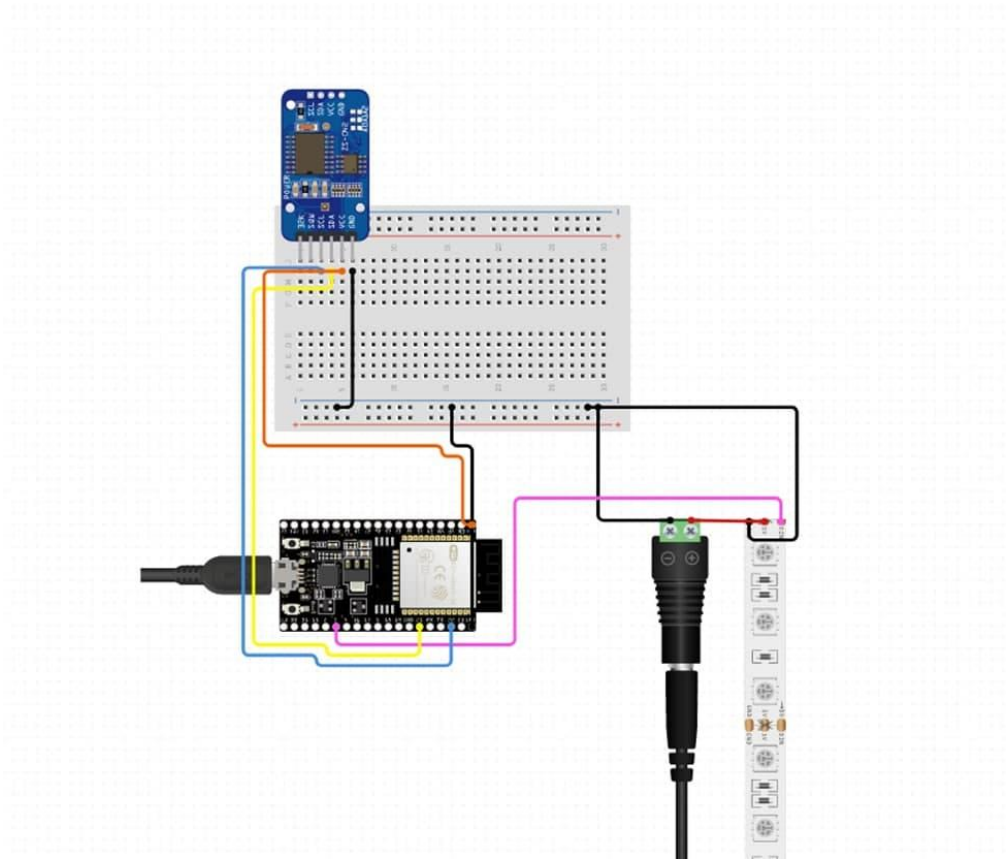

Рисунок 4.6 – Схема апаратно-програмного комплексу

Реалізація полягає в отримані даних у вигляді рядка, де першим словом є команда, а далі йдуть параметри у вигляді цілих чисел. Слова та числа розділені пробілом. Розглянемо команду **power 1**, де командою є слово **power**, а параметром є число **1**. Дана команда вмикає живлення, якби другим параметром було б число 0, то живлення було б вимкнено. Список команд із прикладом параметрів та їх призначення наведено в таблиці 4.1. У круглих дужках наведено діапазон допустимих значень.

Таблиця 4.1 – Команди та їх призначення

| **Команда** | **Призначення** |
|---|---|
| power | Зміна живлення |
| auto | Зміна автоматичного режиму роботи |
| light (0-255) (0-255) (0-255) | Зміна кольору |
| brightness (0-100) | Зміна яскравості |

| | |
|---|---|
| alarm_on (0-23) (0-59) | Встановлення часу будильника |
| alarm_off | Вимкнення будильника |
| timer (0-23) (0-59) (0-59) (0-1) | Встановлення таймера |
| time (0-23) (0-59) | Встановлення часу |

Управління апаратною частиною передбачає керування не тільки за допомогою Bluetooth, а й за допомогою WiFi. Для реалізації керування за допомогою WiFi було використано протокол MQTT. Використання даного протоколу дозволяє керувати нашим пристроєм не будучи підключеним до тієї ж мережі, що й сам пристрій. Це означає, що є можливість керування з будь-якого місця, головне, щоб було підключення до інтернету.

MQTT або Message Queue Telemetry Transport – це легкий, компактний і відкритий протокол обміну даними, створений для передачі даних на віддалених локаціях, де потрібен невеликий розмір коду та обмеження по пропускній здатності каналу. Його переваги дозволяють застосовувати його в системах M2M та IoT [8].

Протокол MQTT визначає два типи об’єктів у мережі: посередник повідомлень та кількість клієнтів. Брокер – це сервер, який отримує всі повідомлення від клієнтів, а потім направляє ці повідомлення до відповідних клієнтів. Клієнт – це все, що може взаємодіяти з брокером для надсилання та отримання повідомлень. Клієнтом може бути датчик IoT на місцях або додаток у центрі обробки даних, який обробляє дані IoT. Клієнт підключається до брокера. Він може підписатися на будь-яку тему повідомлення у брокера. Це з’єднання може бути звичайним з’єднанням TCP/IP або зашифрованим з’єднанням TLS для конфіденційних повідомлень. Клієнт публікує повідомлення під темою, надсилаючи повідомлення та тему брокеру. Потім брокер пересилає повідомлення всім клієнтам, які підписалися на цю тему. Оскільки повідомлення MQTT упорядковані, певні клієнти можуть взаємодіяти лише з певними повідомленнями [8].

MQTT має наступні переваги:

– ефективний обмін інформацією;
– збільшення масштабованості;

– зменшення споживання пропускної здатності мережі;
– дозволяє віддалене керування;
– максимізація доступної пропускної здатності;
– захищеність даних;
– економія часу на розробку.

В якості MQTT брокера обрано EMQ X Broker. EMQ X Broker – це масштабований широко розповсюджуваний розподільник повідомлень MQTT, написаний на мові Erlang / OTP. EMQ X Broker може бути розгорнутий де завгодно, від обмежених ресурсами пристроїв до приватних, гібридних та загальнодоступних хмар.

Даний брокер було обрано через його наступні переваги: повністю відкритий код, кластерна архітектура, підтримка MQTT V5.0, висока продуктивність, легке розгортання.

### 4.3 Опис програмної реалізації мобільного застосунку

Для керування інтелектуальним освітленням створено мобільний застосунок для операційної системи Android із використанням мови програмування Dart та фреймворку Flutter та середовища розробки Android Studio.

Оскільки даний застосунок має багато функцій, зокрема інтеграцію з апаратно-програмним комплексом та серверною частиною, було використано декілька підходів.

Для інтеграції з серверною частиною підключено бібліотеку http, яка дозволяє надсилати запити на сервер, та додано необхідний дозвіл в AndroidManifest. Також створено окремий клас під назвою SmartLightService, який відповідає за надсилання та обробку даних на сервер та з сервера. Даний клас відповідає за наступний функціонал: реєстрація, авторизація, отримання збережених режимів, збереження створеного режиму.

Через те, що створена система може керуватися як з використанням Bluetooth, так і з використанням WiFi, вирішено використати декілька принципів об’єктно-орієнтованого програмування під назвою абстракція та наслідування. Зокрема,

створено абстрактний клас CommandService, який має у собі опис необхідних методів для керування, та класи BluetoothCommandService і MQTTCommandService, які його наслідують. Даний підхід обрано через те, що мета у даних класів одна, але реалізація різна через різницю в протоколах зв'язку.

При реалізації BluetoothCommandService використано бібліотеку flutter_bluetooth_serial та додано необхідні дозволи в AndroidManifest. З метою зменшення запитів на під'єднання до апаратно-програмного комплексу створено допоміжний клас BluetoothConnectionService, який реалізує шаблон програмування під назвою Singletone. Це означає, що достатньо лише один раз підключитися і не буде необхідності перепідключатися кожен раз при відправці повідомлення. Перепідключення необхідне буде лише при втраті зв'язку.

Для спілкування через протокол WiFi з використанням протоколу MQTT підключено біблотеку mqtt_client. Реалізація подібна до тієї, що використовується у класі BluetoothCommandService.

При реалізації деяких функцій необхідно було реалізувати збереження даних у пам'яті пристрою із можливістю отримати їх після завершення програми. Для цього використано бібліотеку shared_preferences та створено клас SharedPreferencesService, який також реалізує шаблон проектування Singletone. Даний клас використовується для збереження токена авторизації, налаштувань, mac та ip адреси пристрою до якого під'єднувалися останнім.

Однією з задач при розробці мобільного застосунку було оновлення даних у додатку при зміні налаштувань інтелектуального освітлення іншим користувачем. Для реалізації поставленої задачі створено методи, які перевіряють зміну даних та оновлюють їх на екрані користувача. Оскільки сервіс перевірки даних знаходиться в іншому класі програми, а не в класі екрана, що відображається, виникла необхідність використання стейт-менеджеру із бібліотеки provider. Також для передачі налаштувань при зміні створено клас LightState, який наслідує клас ChangeNotifier та також реалізує шаблон проектування Singletone, оскільки об'єкт налаштувань має бути один на всю програму. При отримані даних про зміну з апаратно-програмного комплексу, в об'єкті змінюються поля, що відповідають за ті

чи інші налаштування, та викликається метод notifyListeners, який сповіщає інші класи про зміну та змінює величини на екрані.

У розробленому застосунку розроблено валідацію та сповіщання користувача про результат введення даних. На рисунку 4.7 наведено приклади спливаючих вікон про результат приєднання до апаратно-програмного комплексу та результат неправильного введення даних при авторизації. Для реалізації вспиваючих вікон використовується бібліотека fluttertoast. Її вирішено використовувати через те, що вона дозволяє викликати сповіщення у будь-якому місці програми, адже їй не потрібен контекст, який є лише у віджеті. Також, результатом про виконання операції слугує круглий індикатор виконання. Ще для вибору кольору та його відтінку було обрано віджет із бібліотеки flutter_circle_color_picker.

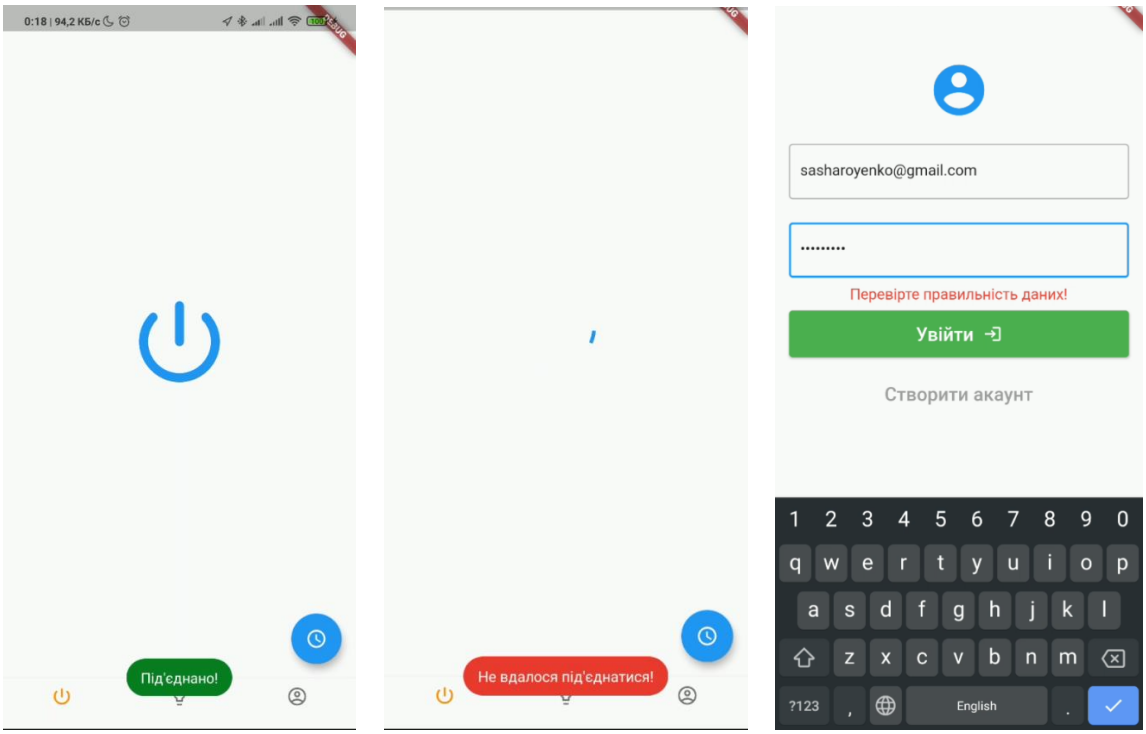


Рисунок 4.7 – Сповіщення користувача

### 4.4 Інструкція з використання

Після того, як користувач завантажить додаток, він повинен спочатку зареєструватися, якщо у нього не має акаунта, або авторизуватися (рисунок 4.8).

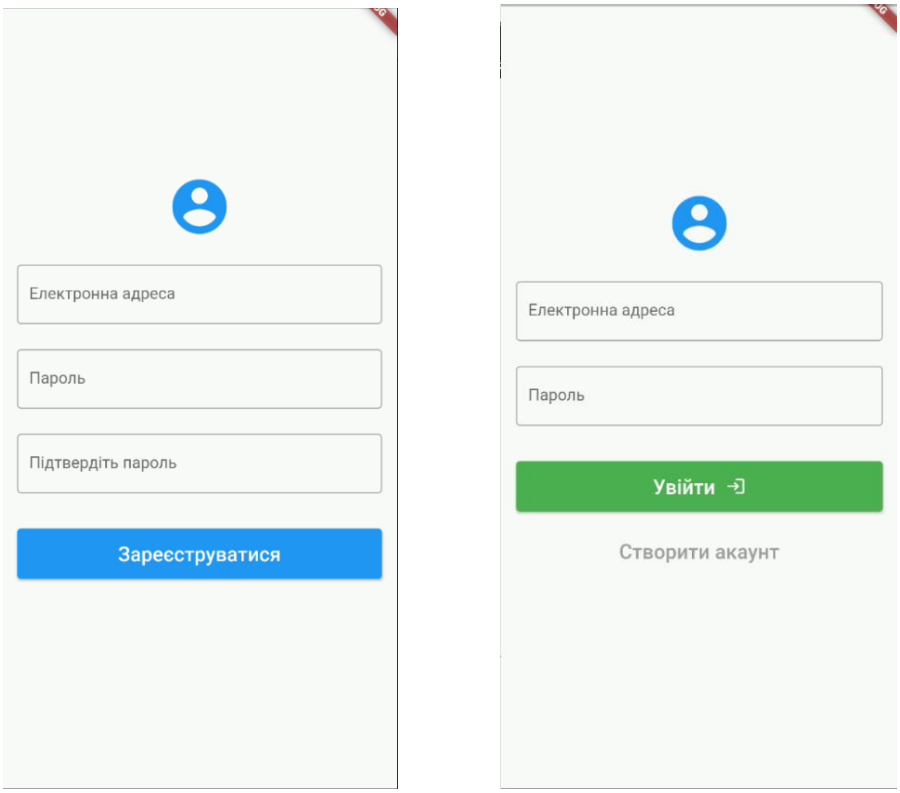


Рисунок 4.8 – Створення акаунта та авторизація

Після цього необхідно під’єднатися до лампи, попередньо включивши її. Для цього необхідно увімкнути Bluetooth або WiFi, в залежності від типу з’єднання, яким користувач хоче під’єднатися, і у додатку зайти у налаштування та натиснути кнопку «підключитися» і після цього обрати необхідний пристрій із списку (рисунок 4.9).

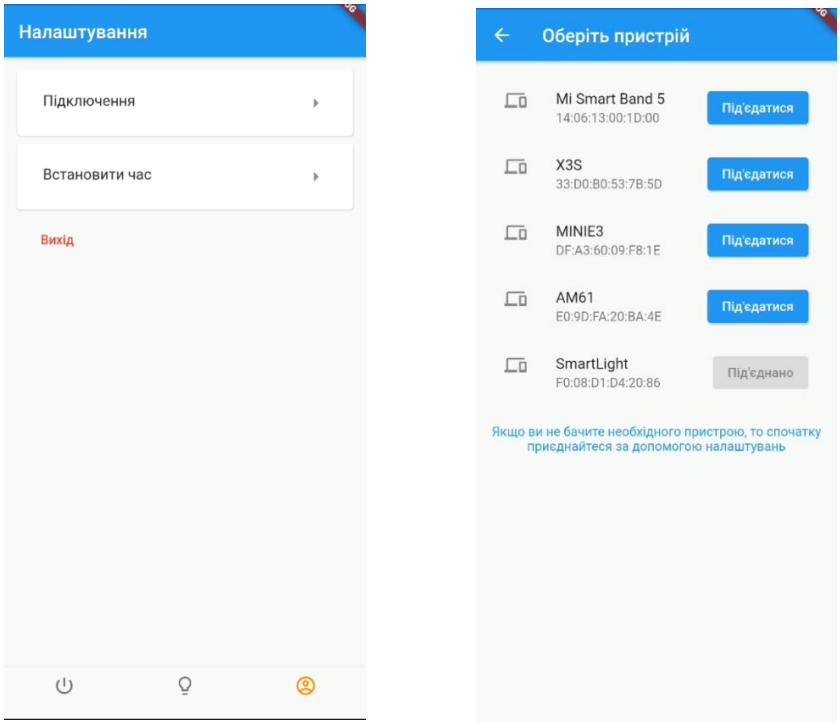


Рисунок 4.9 – Підключення лампи до додатку

Також у налаштуваннях можна встановити час за яким буде працювати лампа (Рисунок 4.10)

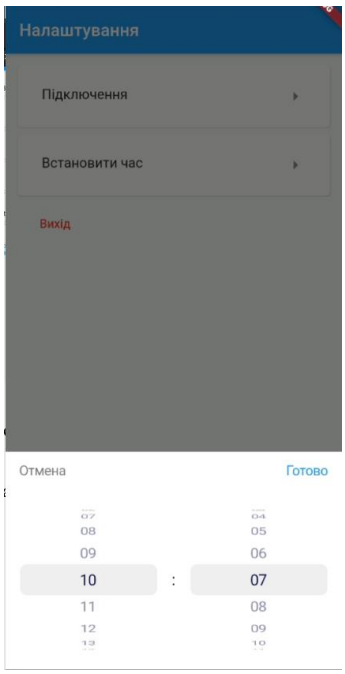


Рисунок 4.10 – Налаштування часу

Після цих дій користувач може керувати лампою. Для управління кольором, яскравістю необхідно перейти на необхідний екран, представлений на рисунку 4.11.

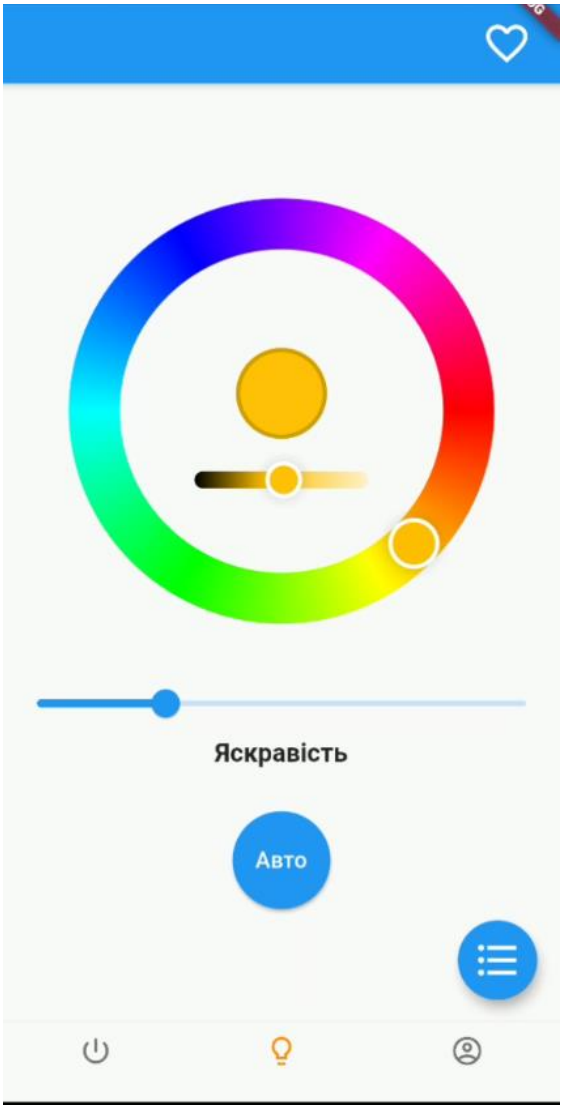


Рисунок 4.11 – Екран налаштування світла

Також на даному екрані можна увімкнути автоматичне освітлення, натиснувши на кнопку авто. Також є можливість зберігати обраний колір натиснувши на іконку у правому верхньому куті і увівши ім’я у вікні, що з’явилося, та натиснувши кнопку підтвердження (рисунок 4.12).

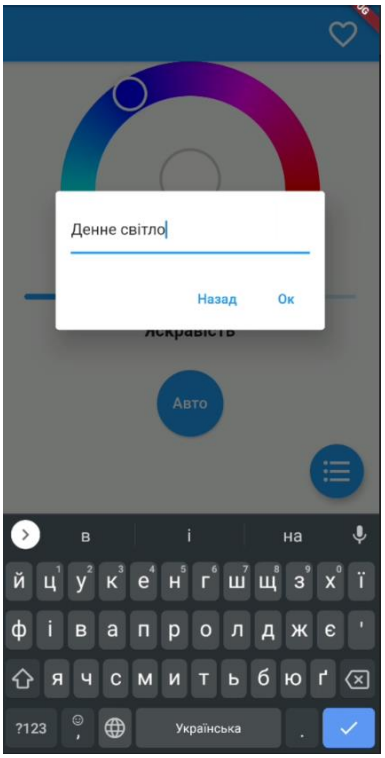


Рисунок 4.12 – Збереження обраного кольору

Переглядати збережені кольори можна натиснувши на кнопку у правому нижньому куті (рисунок 4.13).

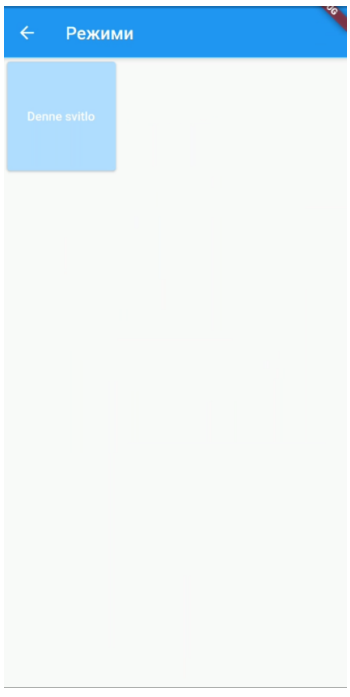

Рисунок 4.13 – Перегляд збережених кольорів

На головному екрані можна вмикати та вимикати світло, натиснувши на іконку живлення (рисунок 4.14) та можна перейти на екран налаштування таймера та будильника (рисунок 4.15). Для додавання таймера необхідно на його вкладці натиснути на іконку таймера, що знаходиться знизу по центру, після цього обрати час на дію(увімкнути/вимкнути) (рисунок 17). Перейшовши на вкладку будильника, ми може увімкнути/вимкнути уже налаштований будильник, або редагувати його час, натиснувши на іконку будильника, що знаходиться знизу по центру, після цього обрати час (рисунок 18).

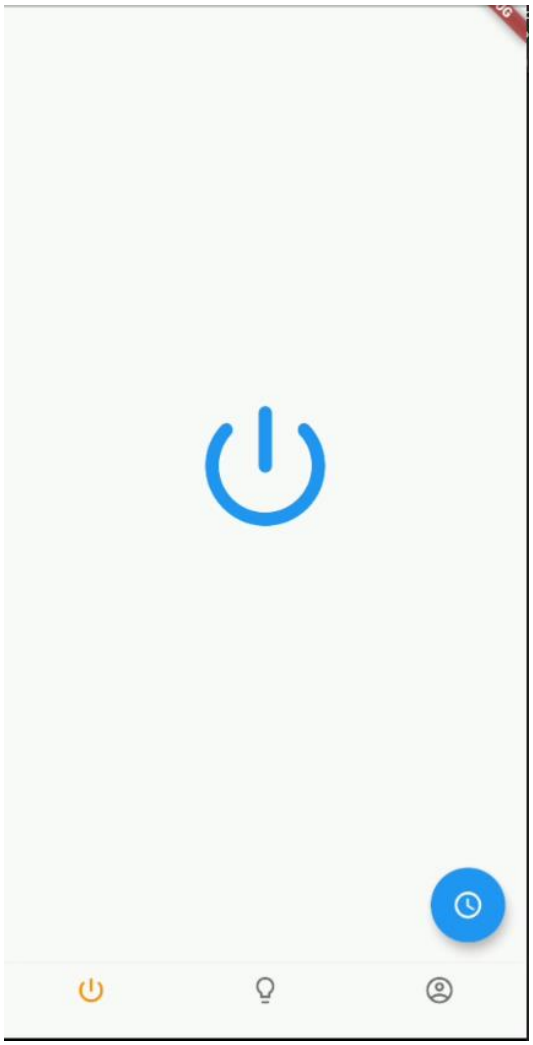

Рисунок 4.14 – Екран зміни режиму живлення

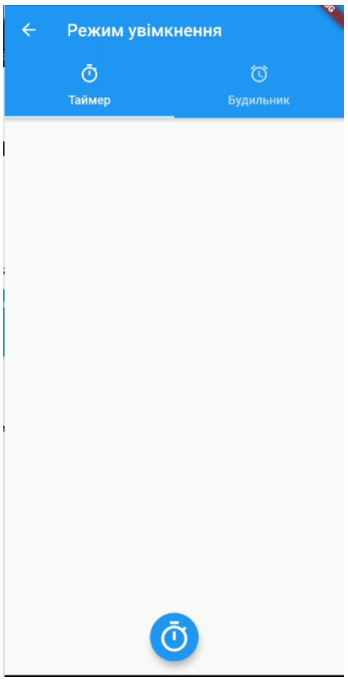


Рисунок 4.15 – Екран налаштування таймера та будильника

Для додавання таймера необхідно на його вкладці натиснути на іконку таймера, що знаходиться знизу по центру, після цього обрати час на дію (увімкнути/вимкнути) (рисунок 4.16).

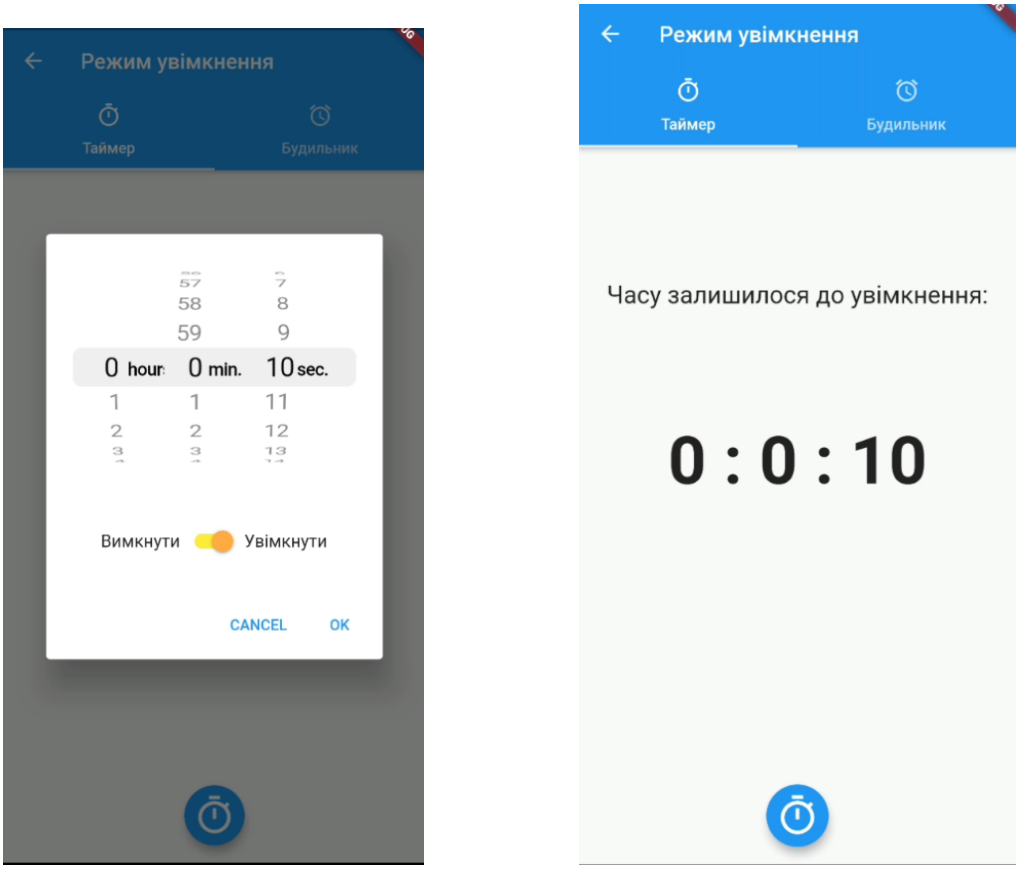


Рисунок 4.16 – Налаштування таймера


Рисунок 4.17 – Налаштування будильника

# ВИСНОВКИ

Результати проведеного дослідження, реалізовані мета та завдання надали підстави сформулювати такі висновки.

1. Проаналізувавши вітчизняну та зарубіжну літературу з проблеми дослідження з'ясовано, що освітлення має досить значний вплив на самопочуття та емоційний стан людини, зокрема воно має безпосередній вплив на біологічний годинник та циркадні ритми людини. Предметом дослідження науковців був вплив різних характеристик освітлення (колірна температура, яскравість, освітленість) на психофізіологічний стан людини.

2. Зменшити негативний вплив, зумовлений порушенням циркадних ритмів та біологічного годинника, можливо при використанні орієнтованого на людину освітлення (Human Centric Lighting), що зосереджує увагу на поверненні динаміки природного денного світла у повсякденне життя людей за допомогою біологічно ефективного штучного освітлення.

3. У межах реалізації завдань досліджено матеріали для розробки інтелектуального освітлення приміщення з врахуванням індивідуальних потреб користувача. Опираючись на дані дослідження, розроблено систему, яка відповідає поставленому завданню. Зокрема, програмний продукт має наступний функціонал:

- налаштування яскравості світла;
- налаштування кольору;
- автоматична адаптація яскравості;
- налаштування таймеру увімкнення та вимкнення;
- налаштування будильника;
- містить режими освітлення для певної діяльності;
- адаптація під циркадні ритми;
- визначення користувача та застосування його налаштувань.

4. Реалізовано на практиці серверну частину, мобільний додаток та апаратну частину. Для розробки було обрано якісні засоби та створено надійну архітектуру та інтуїтивно зрозумілий інтерфейс. Дана система показала високу

надійність під час тестування та може використовуватися в офісах, навчальних закладах, комерційних та промислових об’єктах для поліпшення продуктивності праці та психоемоційного стану людей.

Поставлена задача дипломної роботи реалізована в повній мірі.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Джошуа Блох Java ефективное програмирование / пер. с англ. И. В. Красикова. Киев : «Диалектика». 2020. 464 с.

2 IoT и проблемы безопасности. URL: https://habr.com/ru/company/unet/blog/410849/ (дата звернення 10.10.2020).

3 Колірна температура. URL: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D1%96%D1%80%D0%BD%D0%B0_%D1%82%D0%B5%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D1%83%D1%80%D0%B0 (дата звернення 12.10.2020).

4 Москаленко А. Авторизация и аутенфикация в ASP.NET Core с использованием ASP.NET Core Identity и JWT. URL: https://fuse8.ru/articles/secure-authorization-with-jwt. (дата звернення 12.04.2020).

5 Уоллс К. Spring в действии. Москва: ДМК Пресс. 2013. 752 с.

6 Фреймворк. URL: https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0

7 Циркадные ритмы: почему дневной свет так важен для хорошего сна? URL: https://www.bbc.com/russian/features-48326088 (дата звернення 20.10.2021).

8 Что такое MQTT и для чего он нужен в IIoT? Описание протокола MQTT. URL: https://ipc2u.ru/articles/prostye-resheniya/chto-takoe-mqtt/ дата звернення 07.10.2021).

9 Что такое люксы и люмены, и почему ватты – не главное в светодиодном освещении. URL: https://maxus.com.ua/ru/blog/chto-takoe-lyuksy-i-lyumeny-i-pochemu-vatty-ne-glavnoe-v-svetodiodnom-osveshchenii https://uk.wikipedia.org/wiki/%D0%9E%D1%81%D0%B2%D1%96%D1%82%D0%BB%D0%B5%D0%BD%D1%96%D1%81%D1%82%D1%8C (дата звернення 15.01.2021).

10 Android OS.URL: https://searchmobilecomputing.techtarget.com/definition/Android-OS(дата звернення 06.04.2021).

11 Chellappa S. L. Can light make us bright effects of light on cognition and sleep. *Progress in Brain Research.* 2011. vol. 190. P. 119-133.

12 Circadian Rhythms and YourBody Ckock. URL: https://www.sleep.org/circadian-rhythm-body-clock/ (дата звернення 21.10.2020).

13 Dart overview. URL: https://dart.dev/overview(дата звернення 30.03.2021).

14 Extremely Accurate I2C-Integrated RTC/TCXO/Crystal. URL: https://datasheets.maximintegrated.com/en/ds/DS3231.pdf (дата звернення 15.03.2021).

15 Getting Started with ESP32 | Introduction to ESP32. URL: https://www.electronicshub.org/getting-started-with-esp32/ (дата звернення 20.03.2021).

16 Grocoff P: Effects of correlated color temperature on perceived visual comfort. PhD dissertation, The University of Michigan. 1996.

17 How Do LED Lights Work? A Tricky Technology, A Messy Marketplace. URL: https://www.manufacturer.lighting/info/223/ (дата звернення 17.01.2021).

18 Jung H.-C., Kim J.-H, and Lee C.-W. The effect of the illuminance of light emitting diode (LED) lamps on long-term memory. *Displays* . 2017. Vol. 49, P. 1-5.

19 Kobayashi H., Sato M. Physiological responses to illuminance and color temperature of lighting. *Ann Physiol Anthropol.* 1992. № 11. P. 45-49.

20 Lee C. W. Kim J. H. The influence of LED lighting on attention and long-term memory. *International Journal of Optics.* 2020. P. 1-6.

21 Ruger M. Time-of-day-dependent effects of bright light exposure on human psychophysiology: comparison of daytime and nighttime exposure. *American Journal of Physiology.* 2006. Vol. 290, №. 5.P. 1413-1420

22 Sleep Foundation. URL: https://www.sleepfoundation.org/circadian-rhythm (дата звернення 10.11.2020).

23 Smolders K. C., de Kort Y. A. W., Cluitmans P. J. M. A higher illuminance induces alertness even during office hours: findings on subjective measures, task performance and heart rate measures. *Physiology & Behavior.* 2012. Vol. 107, P. 7-16

24 Spring Framework и Spring Boot: понимание различий. URL: https://topjava.ru/blog/spring-framework-vs-spring-boot-differences (дата звернення: 17.12.2020).

25 What Is a Database? URL: https://www.oracle.com/database/what-is-database/(дата звернення 01.04.2021).

26 What is Arduino? Getting Started with ESP32 | Introduction to ESP32. URL: https://www.arduino.cc/en/guide/introduction (дата звернення 28.03.2021).

27 What is Flutter and Why You Should Learn it in 2020. URL: https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/ (дата звернення 12.03.2021).

28 What is REST. URL: https://www.codecademy.com/articles/what-is-rest (дата звернення 20.03.2021).

29 What is the IoT? Everything you need to know about the Internet of Things right now. URL: https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/ (дата звернення 01.11.2020).

30 Why the Internet of Things is called Internet of Things: Definition, history, disambiguation. URL: https://iot-analytics.com/internet-of-things-definition/ (дата звернення 17.12.2020).

# ДОДАТОК А

Інтелектуальне освітлення приміщення з врахуванням індивідуальних потреб користувача

Специфікація

УКР.НТУУ “КПІ” _ТЕФ_АПЕПС_ТІ72135

Аркушів 2

Київ — 2021

| Позначення | Найменування | Примітки |
|---|---|---|
| Документація | | |
| УКР.НТУУ “КПІ”_ТЕФ_АПЕПС_ТІ72135 | Записка.docx | Текстова частина дипломної роботи |
| Компоненти | | |
| УКР.НТУУ “КПІ”_ТЕФ_АПЕПС_ТІ72135 | smart_light.ino | Компонент керування апаратною частиною |
| УКР.НТУУ “КПІ”_ТЕФ_АПЕПС_ТІ72135 | JwtTokenProvider.java | Компонент автоизації за допомогою токена |
| УКР.НТУУ “КПІ”_ТЕФ_АПЕПС_ТІ72135 | shared_preferences_service.dart | Компонент що відповідає за збереження даних у мобільному пристрої |
| УКР.НТУУ “КПІ”_ТЕФ_АПЕПС_ТІ72135 | smart_light_service.dart | Компонент що відповідає за надсилання запитів на веб сервер |
| УКР.НТУУ “КПІ”_ТЕФ_АПЕПС_ТІ72135 | bluetooth_command_service.dart | Компонент що відповідає за надсилання даних за допомогою Bluetooth |
| УКР.НТУУ “КПІ”_ТЕФ_АПЕПС_ТІ72135 | bluetooth_connection_service.dart | Компонент що відповідає за під’єднання за допомогою |

| | | |
|---|---|---|
| | | Bluetooth |

# ДОДАТОК Б

Інтелектуальне освітлення приміщення з врахуванням індивідуальних потреб користувача

Текст програми

УКР.НТУУ “КПІ” _ТЕФ_АПЕПС_ТІ72135

Аркушів 14

Київ — 2021

## smart_light.ino

```
// Подключаем библиотеку
#include "BluetoothSerial.h"
#include "esp32-hal-ledc.h"
#include "RTClib.h"
#include "GyverTimer.h"
#include <Adafruit_NeoPixel.h>

// Проверка, что Bluetooth на плате
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

//Define pins
#define LED_PIN 12
#define CLOCK_INTERRUPT_PIN 23
#define LED_COUNT 256

//Define constants
#define DAWN_TIMEOUT 15 + 1
#define DAWN_TIMEOUT_AFTERCLOCK 5

// Define variables
BluetoothSerial SerialBT;
RTC_DS3231 rtc;
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);

bool isOnAfterTimer = 0;
DateTime timerTime;
bool isTimer = true;
bool isPowerOn = false;
int red = 255;
int green = 255;
int blue = 255;
int brightness = 100;
// auto mode variables
bool isAuto = false;
const int dayColorsSize = 9;
int finishedCurrentTimeAutoMode = 0;
int dayColors[dayColorsSize][5] =
{
  {6, 8, 255, 115, 0}, //6-8
  {8, 10, 248, 211, 0}, //8-10
  {10, 12, 255, 245, 171}, // 10-12
  {12, 14, 255 , 255 , 255}, // 12-14
  {14, 16, 255, 252, 196}, //14-16
  {16, 18, 255, 241, 139}, //16-18
  {18, 20, 254, 255, 90}, //18-20
  {20, 22, 248, 211, 0}, //20-22
  {22, 24, 121, 90, 0} //22-24
};
// alarm and dawn variables
DateTime alarmTime;
int dawnColorTimeStepMs;
bool isAlarm = false;
int dawnTimeout = DAWN_TIMEOUT;
const int dawnColorsSize = 14;
```

```
int dawnColors[dawnColorsSize][3] =
{
  {170 , 0, 0},
  {232 , 29, 0},
  {185 , 62 , 0},
  {255 , 87 , 1},
  {231 , 101, 0},
  {231 , 135, 0},
  {238 , 217 , 0},
  {255 , 162, 32},
  {255 , 181 , 32},
  {255 , 252, 89},
  {255 , 208 , 32},
  {255 , 234, 15},
  {255 , 239, 66},
  {255, 255, 255}
};
int currentDawnColorIndex = 0;
int dawnBrightness = 20;
GTimer dawnTimer(MS);

void setup() {
  Serial.begin(115200); // включаем передачу данных у последовательного порта
  SerialBT.begin("SmartLight");  // Может придумать своё имя
  Serial.println("The  device started, now you can pair it with bluetooth!");

  pixels.setBrightness(100);
  setupRtc();
}

void loop() {
  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }
  if (SerialBT.available()) {
    String command = SerialBT.readStringUntil('\n');
    Serial.println(command);
    commandParser(command);
  }
  processCommands();
}

void processCommands() {
  DateTime now = rtc.now();

  if (rtc.alarmFired(1)) {
    Serial.println("Alarm  cleared!");
    rtc.clearAlarm(1);
  }

  if (isTimer
      && now.hour() == timerTime.hour()
      && now.minute() == timerTime.minute()
      && now.second() == timerTime.second()) {
    isPowerOn  = isOnAfterTimer;
    Serial.println("Inside  timer!");
    changePowerState(isOnAfterTimer);
    isTimer = false;
  }
  checkDawn();
  if (isAuto) {
    processAuto();
  }
```

```
}

void commandParser(String command) {
  if (getValue(command, ' ', 0) == "power") {
    //bool isOn = getValue(command, ' ', 1).toInt();
    Serial.println("Change power state");
    //Serial.println(isOn);
    isPowerOn = !isPowerOn;
    changePowerState(isPowerOn);
    if (isAlarm) {
      resetDawnParams();
    }
  } else if (getValue(command, ' ', 0) == "auto") {
    isAuto = !isAuto;
    finishedCurrentTimeAutoMode = 0;
    Serial.print("Auto mode: ");
    Serial.println(isAuto);
    if (!isAuto) {
      writeColor(red, green, blue);
    }
  } else if (getValue(command, ' ', 0) == "light") {
    Serial.println("Write color");
    red = getValue(command, '', 1).toInt();
    green = getValue(command, ' ', 2).toInt();
    blue = getValue(command, ' ', 3).toInt();
    isPowerOn = true;
    isAuto = false;
    writeColor(red, green, blue);
    if (isAlarm) {
      resetDawnParams();
    }
  } else if (getValue(command, ' ', 0) == "brightness") {
    brightness = getValue(command, ' ', 1).toInt();
    Serial.print("Set brightness ");
    setBrightness(brightness);
  } else if (getValue(command, ' ', 0) == "alarm_on") {
    Serial.println("Set alarm");
    printDate();
    int hour = getValue(command, ' ', 1).toInt();
    int minute = getValue(command, ' ', 2).toInt();
    processAlarm(hour, minute);
  } else if (getValue(command, ' ', 0) == "alarm_off") {
    Serial.println("Remove alarm");
    resetDawnParams();
    rtc.clearAlarm(1);
    rtc.disableAlarm(1);
    Serial.println("Alarm cleared!");

  } else if (getValue(command, ' ', 0) == "timer") {
    Serial.println("Set timer");
    int timerHour = getValue(command, '', 1).toInt();
    int timerMinute = getValue(command, '', 2).toInt();
    int timerSeconds = getValue(command, '', 3).toInt();
    isOnAfterTimer = getValue(command, ' ', 4).toInt();
    isTimer = true;
    timerTime = rtc.now() + TimeSpan(0, timerHour, timerMinute, timerSeconds);
  } else if (getValue(command, ' ', 0) == "time") {
    Serial.println("Set time");
    int hour = getValue(command, ' ', 1).toInt();
    int minute = getValue(command, ' ', 2).toInt();
    int second = getValue(command, ' ', 3).toInt();
    //rtc.setTime(hour, minutes, seconds);
    DateTime now = rtc.now();
```

```
    rtc.adjust(DateTime(now.year(), now.month(), now.day(), hour, minute, second));
    printDate();
    finishedCurrentTimeAutoMode = 0;
   }

  sendState();
}

void sendState() {
  DateTime now =  rtc.now();
  String settings = String("power  ") + isPowerOn  + "\n" +
                "light  " + red + " " + green + " " + blue + "\n" +
                "brightness " + brightness + "\n" +
                "isAuto " + isAuto + "\n" +
                "alarm " + alarmTime.hour() + " " + alarmTime.minute() + " " + isAlarm  + "\n" +
                "time " + now.hour() + " " + now.minute() + "\n";
  SerialBT.println(settings);
}

void writeColor(int  red, int green, int blue) {
  for (int i = 0; i < LED_COUNT;  i++)
  {
    pixels.setPixelColor(i,  pixels.Color(red,  green, blue));
  }
  pixels.show();

}

void setBrightness(int brightnessValue) {
  pixels.setBrightness(brightnessValue);
  pixels.show();
}

void changePowerState(bool isOn)  {
  if (isOn)  {
    writeColor(red,  green,  blue);
  } else  {
    writeColor(0,  0, 0);
  }
}

void processAlarm(int hour, int minute) {

  rtc.clearAlarm(1);
  DateTime now =  rtc.now();
  alarmTime = DateTime(now.year(), now.month(), now.day(), hour, minute, 0);

  Serial.print("Alarm  time: ");
  printDate(alarmTime);

  int difference = (alarmTime.hour() * 60 + alarmTime.minute()) - DAWN_TIMEOUT;
  DateTime dawnTime = DateTime(now.year(), now.month(), now.day(), difference / 60, (difference  - ((difference / 60) * 60)),
0);

  Serial.print("Dawn  time: ");
  printDate(dawnTime);

  //check if alarm time - dawn timeout < now
  DateTime timeAfterDawnTime = rtc.now() + TimeSpan(0, 0, DAWN_TIMEOUT,  0);

  Serial.print("TimeAfterDawnTime  time: ");
  printDate(timeAfterDawnTime);
```

```
  difference = (alarmTime.hour() * 60 + alarmTime.minute()) - (now.hour() * 60 + now.minute());
  /*if (dawnTime < now && abs(dawnTime.hour() - now.hour()) <= 1 && alarmTime > now) {
    Serial.println("Start  dawn time calculating");
    if (now.hour() - dawnTime.hour() == 0) {
      difference = abs(now.minute() - dawnTime.minute());
    } else {
      difference = abs(60 - (now.minute() + dawnTime.minute()));
    }
    }*/

  Serial.print("Difference:  ");
  Serial.println(difference);


  if (alarmTime > now && difference < DAWN_TIMEOUT  ) {
    dawnTimeout = abs(difference) - 1;
  } else {
    dawnTimeout = DAWN_TIMEOUT  - 1;
  }
  dawnTime = DateTime(alarmTime.unixtime() - dawnTimeout * 60);

  Serial.print("Dawn  timeout: ");
  Serial.println(dawnTimeout);

  dawnColorTimeStepMs = (double(dawnTimeout + 1) / double(dawnColorsSize)) * 60.0 * 1000.0;;

  Serial.print("Dawn  color time step ms time: ");
  Serial.println(dawnColorTimeStepMs);

  Serial.print("Dawn  time: ");
  printDate(dawnTime);


  if (!rtc.setAlarm1(
        dawnTime,
        DS3231_A1_Hour
      )) {
    Serial.println("Error,  alarm wasn't set!");
  } else {
    Serial.println("Alarm  set!");
    isAlarm  = true;
  }
}

void processAuto() {
  DateTime now = rtc.now();
  if (now.hour() >= 24 && now.hour() < 6) {
    writeColor(0  , 0, 0);
  } else {
    for (int i = 0; i < dayColorsSize; i++)  {
      if (now.hour() >= dayColors[i][0] && now.hour() < dayColors[i][1] && finishedCurrentTimeAutoMode !=
dayColors[i][1]) {
        Serial.print("Change auto mode to ");
        Serial.println(i  + 1);
        finishedCurrentTimeAutoMode = dayColors[i][1];
        writeColor(dayColors[i][2]  , dayColors[i][3], dayColors[i][4]);
        break;
      }
    }
  }

}
```

```
void checkDawn() {
  if (isAlarm) {
    DateTime now = rtc.now();
    DateTime finishDawnTime = alarmTime + TimeSpan(0, 0, DAWN_TIMEOUT_AFTERCLOCK,  0);
    if (isTime(now, finishDawnTime)) {
      resetDawnParams();
      writeColor(red,  green, blue);
      Serial.println("Dawn  finished!");

    } else if (currentDawnColorIndex == 0) {
      Serial.println("First  dawn color!");
      setBrightness(dawnBrightness);
      writeColor(dawnColors[currentDawnColorIndex][0], dawnColors[currentDawnColorIndex][1],
dawnColors[currentDawnColorIndex][2]);
      dawnTimer.setTimeout(dawnColorTimeStepMs);
      currentDawnColorIndex++;
    } else if (currentDawnColorIndex < dawnColorsSize && dawnTimer.isReady()) {

      Serial.print(currentDawnColorIndex + 1);
      Serial.print("  dawn color! ");
      Serial.print("  dawn color! Brightness=");
      Serial.println(dawnBrightness);
      setBrightness(dawnBrightness);
      writeColor(dawnColors[currentDawnColorIndex][0], dawnColors[currentDawnColorIndex][1],
dawnColors[currentDawnColorIndex][2]);
      currentDawnColorIndex++;
      dawnTimer.start();
    }
    if (dawnColorsSize - currentDawnColorIndex != 0) {
      dawnBrightness += (100 - dawnBrightness) / (dawnColorsSize - currentDawnColorIndex);
    } else {
      dawnBrightness = 100;
    }
  }
}

void resetDawnParams() {
  dawnBrightness = 20;
  isAlarm = false;
  currentDawnColorIndex = 0;
}

void resetAuto() {
  isAuto = false;
  finishedCurrentTimeAutoMode = 0;
}

bool isTime(DateTime firstTime, DateTime secondTime) {
  return firstTime.hour() == secondTime.hour()
        && firstTime.minute() == secondTime.minute();
}

void onAlarm() {
  Serial.println("Alarm  interrupt occured!");
  isAlarm = true;
  currentDawnColorIndex = 0;
  Serial.println(isAlarm);
}

void setupRtc() {
  if (!rtc.begin())  {
    Serial.println("Couldn't  find RTC!");
    Serial.flush();
```

```
    abort();
  }

  //if (rtc.lostPower()) {
  // this will adjust to the date and time at compilation
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  //}

  //we don't need the 32K Pin, so disable it
  rtc.disable32K();

  // Making it so, that the alarm will trigger an interrupt
  pinMode(CLOCK_INTERRUPT_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(CLOCK_INTERRUPT_PIN), onAlarm, FALLING);

  // set alarm 1, 2 flag to false (so alarm 1, 2 didn't happen so far)
  // if not done, this easily leads to problems, as both register aren't reset on reboot/recompile
  rtc.clearAlarm(1);
  rtc.clearAlarm(2);

  // stop oscillating signals at SQW Pin
  // otherwise setAlarm1 will fail
  rtc.writeSqwPinMode(DS3231_OFF);

  // turn off alarm 2 (in case it isn't off already)
  // again, this isn't done at reboot, so a previously set alarm could easily go overlooked
  rtc.disableAlarm(2);
  //rtc.disableAlarm(1);

}

void printDate() {
  DateTime now = rtc.now();
  Serial.print(now.year(), DEC);
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.print(now.day(), DEC);
  Serial.print(" (");
  Serial.print(now.dayOfTheWeek());
  Serial.print(") ");
  Serial.print(now.hour(), DEC);
  Serial.print(':');
  Serial.print(now.minute(), DEC);
  Serial.print(':');
  Serial.print(now.second(), DEC);
  Serial.println();
}

void printDate(DateTime date) {
  Serial.print(date.year(), DEC);
  Serial.print('/');
  Serial.print(date.month(), DEC);
  Serial.print('/');
  Serial.print(date.day(), DEC);
  Serial.print(" (");
  Serial.print(date.dayOfTheWeek());
  Serial.print(") ");
  Serial.print(date.hour(), DEC);
  Serial.print(':');
  Serial.print(date.minute(), DEC);
  Serial.print(':');
  Serial.print(date.second(), DEC);
```

```
  Serial.println();
}


String getValue(String data, char separator, int index)
{
  int maxIndex = data.length() - 1;
  int j = 0;
  String chunkVal = "";

  for (int i = 0; i <= maxIndex && j <= index; i++)
  {
    chunkVal.concat(data[i]);

    if (data[i] == separator)
    {
      j++;

      if (j > index)
      {
        chunkVal.trim();
        return chunkVal;
      }

      chunkVal = "";
    }
    else if ((i == maxIndex) && (j < index)) {
      chunkVal = "";
      return chunkVal;
    }
  }
}
```

## JwtTokenProvider.java

```
@Component
@RequiredArgsConstructor
public class JwtTokenProvider {

    @Value("${jwt.token.secret}")
    private String secret;

    //    @Value("${jwt.expiration.time.milliseconds}")
    private final long validityInMilliseconds = Integer.MAX_VALUE;

    private final UserDetailsService  userDetailsService;

    @PostConstruct
    protected void init() {
        secret = Base64.getEncoder().encodeToString(secret.getBytes());
    }


    public String createToken(String username) {
        Claims claims = Jwts.claims().setSubject(username);

        Date now = new Date();
        Date validity = new Date(now.getTime() + validityInMilliseconds);

        return Jwts.builder()
                .setClaims(claims)
```

```
            .setIssuedAt(now)
            .setExpiration(validity)
            .signWith(SignatureAlgorithm.HS256, secret)
            .compact();
    }

    public Authentication getAuthentication(String token) {
        UserDetails userDetails = this.userDetailsService.loadUserByUsername(getLogin(token));
        return new UsernamePasswordAuthenticationToken(userDetails, "", userDetails.getAuthorities());
    }

    public String getLogin(String token) {
        return Jwts.parser().setSigningKey(secret).parseClaimsJws(token).getBody().getSubject();
    }

    public boolean validateToken(String token) {
        try {
            Jws<Claims> claims = Jwts.parser().setSigningKey(secret).parseClaimsJws(token);

            return !claims.getBody().getExpiration().before(new Date());
        } catch (Exception e) {
            throw new JwtAuthenticationException(e);
        }
    }

    public String resolveToken(HttpServletRequest req) {
        String bearerToken = req.getHeader("Authorization");
        if (bearerToken != null && bearerToken.startsWith("Bearer ")) {
            return bearerToken.substring(7);
        }
        return null;
    }
}
```

## shared_preferences_service.dart

```
class SharedPreferencesService {
  static SharedPreferences _sharedPreferences;
  static SharedPreferencesService _sharedPreferencesService;

  SharedPreferencesService._() {}

  static SharedPreferencesService getInstance() {
    if (_sharedPreferencesService == null) {
      _sharedPreferencesService = SharedPreferencesService._();
      _initPreferences();
    }
    return _sharedPreferencesService;
  }

  static Future<SharedPreferencesService> getInstanceAsync() async {
    if (_sharedPreferencesService == null) {
      _sharedPreferencesService = SharedPreferencesService._();
      await _initPreferences();
    }
    return _sharedPreferencesService;
  }

  static _initPreferences() async {
    if (_sharedPreferences == null)
      _sharedPreferences = await SharedPreferences.getInstance();
```

```
  }
  void putObject(String key, var value) {
    _sharedPreferences.setString(key, jsonEncode(value));
  }

  Map<String, dynamic> getObject(String key) {
    var result = _sharedPreferences.get(key);
    return result != null ? jsonDecode(result) : null;
  }
  List<dynamic> getObjects(String key) {
    var result = _sharedPreferences.get(key);

    return result != null ? jsonDecode(result) : null;
  }
  void putString(String key, String value) {
    _sharedPreferences.setString(key, value);
  }
  String getString(String key) {
    return _sharedPreferences.getString(key);
  }

  void remove(String key) {
    _sharedPreferences.remove(key);
  }
  void clear() {
    _sharedPreferences.clear();
  }
}
```

## smart_light_service.dart

```
class SmartLightService {
  static const String MAIN_URL = "http://192.168.0.199:5624/smart-light";
  static const String LOGIN_URL = MAIN_URL + "/login";
  static const String USER_URL = MAIN_URL + "/users";
  static const String USER_OPTION_URL = USER_URL + "/options";
  static SharedPreferencesService _sharedPreferencesService =
  SharedPreferencesService.getInstance();
  Future<dynamic> login(Login login) async {
    var body = jsonEncode(
        <String, String>{"email": login.email, "password": login.password});
    Uri uri = Uri.parse(LOGIN_URL);
    var response = await http.post(
      uri,
      body: body,
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
      },
    );
    return response;
  }
  Future<dynamic> createAccount(CreateAccountDto createAccountDto) async {
    var body = jsonEncode(createAccountDto.toJson());
    Uri uri = Uri.parse(USER_URL);
    var response = await http.post(
      uri,
      body: body,
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
      },
    );
```

```
    return response;
  }
  Future<dynamic> saveOption(Option option) async {
    String token = _sharedPreferencesService.getString("token");
    var body = jsonEncode(option.toJson());
    Uri uri = Uri.parse(USER_OPTION_URL);
    var response = await http.post(
      uri,
      body: body,
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
        'Authorization': token,
      },
    );
    return response;
  }

  Future<List<Option>> getOptions() async {
    String token = _sharedPreferencesService.getString("token");

    Uri uri = Uri.parse(USER_OPTION_URL);

    var response = await http.get(
      uri,
      headers: <String, String>{
        'Authorization': token,
      },
    );
    if (response.statusCode == 200) {
      return jsonDecode(response.body)
          .map<Option>((e) => Option.fromJson(e))
          .toList();
    } else {
      return [];
    }
  }
}
```

## bluetooth_command_service.dart

```
class BluetoothCommandService implements CommandService {
  BluetoothConnection _connection;

  BluetoothCommandService(this._connection);

  @override
  void alarm(DateTime time) {
    String command =
        "alarm_on " + time.hour.toString() + " " + time.minute.toString();

    _executeCommand(command);
  }
  @override
  void stopAlarm() {
    _executeCommand("alarm_off ");
  }
  @override
  void color(Color color) {
    String command = "light " +
        color.red.toString() +
        " " +
        color.green.toString() +
```

```
      " " +
      color.blue.toString() +
      "\n";
  _executeCommand(command);
}
@override
void power() {
  _executeCommand("power ");
}
@override
void timer(DateTime time, int isOn) {
  String command = "timer " +
      time.hour.toString() +
      " " +
      time.minute.toString() +
      " " +
      time.second.toString() +
      " " +
      isOn.toString();

  _executeCommand(command);
}
Future<void> _executeCommand(String command) async {
  if (_connection != null) {
    if (!_connection.isConnected) {
      _connection = (await UtilService.connectToBluetooth())._connection;
    }
    _connection.output.add(utf8.encode(command + "\r\n"));
    _connection.output.allSent;
  } else {
    Fluttertoast.showToast(
        msg: "Не вдалося під'єднатися!",
        toastLength: Toast.LENGTH_SHORT,
        gravity: ToastGravity.BOTTOM,
        timeInSecForIosWeb: 2,
        backgroundColor: Colors.red,
        textColor: Colors.white,
        fontSize: 16.0);
  }
}
@override
void brightness(int brightness) {
  String command = "brightness $brightness";
  _executeCommand(command);
}
@override
void auto() {
  _executeCommand("auto ");
}
@override
void time(DateTime dateTime) {
  String command = "time " +
      dateTime.hour.toString() +
      " " +
      dateTime.minute.toString() +
      " " +
      dateTime.second.toString();

  _executeCommand(command);
}
@override
void read() {
  _executeCommand("read ");
```

```
  }
}
```

## bluetooth_connection_service.dart

```
class BluetoothConnectionService {
  BluetoothDevice _bluetoothDevice;
  BluetoothConnection _connection;
  static BluetoothConnectionService _bluetoothConnectionService;
  SharedPreferencesService _sharedPreferencesService =
      SharedPreferencesService.getInstance();
  BluetoothConnectionService._();
  Future<BluetoothConnection> connect(BluetoothDevice device) async {
    if (_connection == null ||
        _connection != null && _bluetoothDevice.address != device.address ||
        !_connection.isConnected) {
      print('Connecting to the device');
      if (_connection != null) {
        _connection.close();
      }
      _bluetoothDevice = device;
      _connection = await BluetoothConnection.toAddress(device.address).onError(
          (error, stackTrace) async =>
              await BluetoothConnection.toAddress(device.address));
      _sharedPreferencesService.putObject("bluetooth_device", device.toMap());

      connection.input.listen((Uint8List data) {
        String lightSettings = ascii.decode(data);
        print('Data incoming: $lightSettings');
        LightState lightState = LightState.getInstance();
        lightState.updateFromString(lightSettings);
        print(lightState);
      }).onDone(() {
        print('Disconnected by remote request');
      });

    }
    return _connection;
  }
  Future<BluetoothConnection> connectToSavedDevice() async {
    Map<String, dynamic> deviceParams =
        _sharedPreferencesService.getObject("bluetooth_device");
    if (deviceParams != null && deviceParams.isNotEmpty) {
      BluetoothDevice device = BluetoothDevice(
        name: deviceParams['name'].toString(),
        address: deviceParams['address'].toString(),
        type: BluetoothDeviceType.fromUnderlyingValue(
            int.parse(deviceParams['type'].toString())),
        isConnected: deviceParams['isConnected'].toString() == "true",
        bondState:
            BluetoothBondState.fromString(deviceParams['bondState'].toString()),
      );
      return await connect(device);
    }
    return _connection;
  }
  static BluetoothConnectionService instance() {
    if (_bluetoothConnectionService == null) {
      _bluetoothConnectionService = BluetoothConnectionService._();
    }
    return _bluetoothConnectionService;
```

```
  }
  BluetoothConnection get connection => _connection;
  BluetoothDevice get bluetoothDevice => _bluetoothDevice;
}
```

# ДОДАТОК В

Інтелектуальне освітлення приміщення з врахуванням індивідуальних потреб користувача

Опис програми

УКР.НТУУ “КПІ” _ТЕФ_АПЕПС_ТІ72135

Аркушів 9

Київ — 2021

# АНОТАЦІЯ

В даному додатку описано мобільний застосунок розроблений для системи інтелектуального освітлення з врахуванням індивідуальних потреб користувача.

Даний застосунок є інтерфейсом для взаємодії користувача з веб сервером та апаратно-програмним комплексом. Користувач може самостійно змінювати налаштування освітлення або увімкнути автоматичний режим за допомогою даного застосунку.

Мобільний застосунок розроблений під операційну систему Android з використанням мови програмування Dart з використанням фреймворку Flutter у середовищі Android Studio.

# ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ .....78
2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ.....79
3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ.....80
4. ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ .....81
5. ВИКЛИК І ЗАВАНТАЖЕННЯ.....82
6. ВХІДНІ ДАНІ ТА ВИХІДНІ ДАНІ.....83

# ЗАГАЛЬНІ ВІДОМОСТІ

У додатку міститься опис основних компонентів мобільного застосунку інтелектуального освітлення з врахуванням індивідуальних потреб користувача. Додаток Б містить програмний код цих компонентів.

Для роботи мобільного застосунку потрібно мати мобільний пристрій на платформі Android версії 9 і вище. Для керування через WiFi необхідно мати доступ до інтернету

Мобільний застосунок розроблений під операційну систему Android з використанням мови програмування Dart з використанням фреймворку Flutter у середовищі Android Studio.

# ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений мобільний застосунок має наступний функціонал:

1. Керування кольором, відтінком та яскравістю освітлення;
2. Керування автоматичним режимом;
3. Встановлення таймеру;
4. Встановлення розумного будильника;
5. Збереження режимів;
6. Налаштування часу
7. Наявність віддаленого керування

# ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Розроблена система складається з трьох основних модулів:

1. Веб сервер - відповідає за авторизацію та зберігання даних користувача в базу даних
2. Апаратно-програмний комплекс – відповідає за надсилання даних на датчики та зчитування з них інформації
3. Мобільний застосунок – відповідає за зв’язок користувача з веб сервером та апаратно-програмним комплексом.

# ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Під час розробки використано використано досить багато технічних засобів. Для реалізації веб серверу використано мову програмування Java, фреймворк Spring Boot та середовище розробки Intellij Idea. В якості бази даних обрано MySql. Апаратно-програмний комплекс реалізовано в середовищі Arduino Ide на мові програмування C/C++. Для реалізації мобільного застосунку використано мову програмування Dart та фреймворк Flutter.

# ВИКЛИК І ЗАВАНТАЖЕННЯ

Для завантаження мобільного застосунку на Android необхідно відкрити проект програмного забезпечення в Android Studio та під'єднати мобільний пристрій з використання шнура до комп'ютера та натиснути на кнопку "Run".

# ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними програми є дані, які користувач вводить за допомогою додатку: поточний час, таймер, будильник, дані освітлення та режим роботи.

Вихідними даними програмного модуля є отримані дані з сервера або апаратно-програмного комплексу.

# ДОДАТОК Г

Інтелектуальне освітлення приміщення з врахуванням індивідуальних потреб користувача

Довідки про впровадження результатів роботи

УКР.НТУУ “КПІ” _ТЕФ_АПЕПС_ТІ72135

Аркушів 9

Київ — 2021