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

«___»_____ 2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення розподілених систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Апаратно – програмний комплекс інтелектуального управління та моніторингу освітлення доріг Smart City»

Виконав:
студент IV курсу, групи ТВ-61
Скороход Орест Анатолійович _____

Керівник:
доцент, к.т.н.
Ковальчук Артем Михайлович _____

Рецензент:
доцент, к.т.н., викладач кафедри ТЕУ Т і АЕС
Сірий Олександр Анатолійович _____

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

Київ – 2020 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
Напрям підготовки: 121 Інженерія програмного забезпечення
Спеціалізація: Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Олександр Коваль
(підпис)
"___"_____ 2020р.

ЗАВДАННЯ

на дипломну роботу студенту

Скороход Орест Анатолійович
(прізвище, ім'я, по батькові)

1. Тема роботи Апаратно – програмний комплекс інтелектуального управління та моніторингу освітлення доріг Smart City
керівник роботи Ковальчук Артем Михайлович кандидат наук, доцент
(прізвище, ім'я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від "25" травня 2020р.
№ 1168-с

2. Строк подання студентом роботи "10" червня 2020р.

3. Вихідні дані до роботи мова прорамування Python, середовища розробки Arduino IDE, PyCharm, СУБД PostgreSQL

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Порівняльний аналіз існуючих рішень, постановка вимог задачі, розробка алгоритму роботи системи, вибір інструментів для розробки системи, розробка програмної частина, створення апаратної частини, розробка клієнтської частини, реалізація можливості змінювати налаштування роботи системи, реалізація можливості перегляду зібраних даних, створення можливостей масштабування системи, висновки за результатами роботи.

5. Перелік ілюстративного матеріалу 1.Тема роботи. 2.Актуальність теми. 3. Постановка задачі. 4. Алгоритм роботи системи. 5. Використані технології. 6. Засоби розробки серверної частини. 7. Засоби розробки апаратної частини. 8. Конструювання системи. 9. Опрацювання сервером запиту. 10. Використання можливостей Django. 11. Робота в Arduino IDE. 12. Демонстрація сторінки логіну. 13. Демонстрація адаптивності. 14. Список світильників. 15. Редагування налаштувань датчика. 16. Висновки.

6. Консультанти розділів роботи

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

7. Дата видачі завдання ”11” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	11.10.2019	
2.	Вивчення та аналіз задачі	11.10.2019-23.12.2019	
3.	Розробка архітектури та загальної структури системи	03.02.2020-04.03.2020	
4.	Розробка структур окремих підсистем	05.03.2020-12.04.2020	
5.	Програмна реалізація системи	13.04.2020-17.05.2020	
6.	Оформлення пояснювальної записки	18.05.2020-07.06.2020	
7.	Захист програмного продукту	27.05.2020	
8.	Передзахист	09.06.2020	
9.	Захист	15.06.2020	

Студент _____ Скороход О. А._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Ковальчук А. М._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Пояснювальна записка містить 30 сторінок, включає 15 рисунків, 3 додатки та 12 посилань.

Мета дипломної роботи заключається в створенні системи інтелектуального управління та моніторингу освітлення доріг та клієнтського додатку для взаємодії з системою.

Результатом роботи є апаратно-програмний комплекс для управління та моніторингу освітленням доріг, розроблений з допомогою Python-фреймворка Django та на платформі Arduino.

Ключові слова: розумне освітлення, апаратна платформа, система моніторингу та управління, економія електроенергії, Smart City.

ABSTRACT

The explanatory note contains 30 pages, includes 15 figures, 3 appendices and 12 references.

The purpose of the thesis is to create a system of intelligent management and monitoring of road lighting and a client application for interaction with the system.

The result is a hardware-software complex for control and monitoring of road lighting, developed using the Python framework Django and on the Arduino platform.

Keywords: smart lighting, hardware platform, monitoring and control system, energy saving, Smart City.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ **7**

ВСТУП **8**

1 ПОСТАНОВКА ЗАДАЧІ **10**

2 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ **11**

2.1 Визначення розумного освітлення 11

2.2 Огляд існуючих аналогів 12

2.2.1 “Смарт” освітлення від Benish GPS 12

2.2.2 Освітлення від компанії Signify 13

2.2.3 Розумне освітлення від “БРАМА - ИНЖЕНЕРНЫЕ СИСТЕМЫ” 13

3 ЗАСОБИ РОЗРОБКИ **15**

3.1 Мова програмування Python 15

3.2 Фреймворк Django 17

3.3 Обчислювальна платформа Arduino 19

3.4 Система керування базами даних Postgres 22

3.5 Інтернет модуль ENC28J60 Ethernet 22

3.6. Набір шаблонів HTML та CSS Bootstrap 24

3.7 Датчик руху HC-SR501 25

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ **27**

4.1. Веб-інтерфейс 28

4.2 Сервер 30

4.3 Апаратна частина 32

5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ 34

ВИСНОВКИ 37

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 38

ДОДАТОК А 39

ДОДАТОК Б 41

ДОДАТОК В 49

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

JSON (JavaScript Object Notation) Простий формат обміну даними, зручний для читання як людиною, так і машиною.

MVC (Model View Controller) архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

SQL (Structured query language) мова структурованих запитів до баз даних.

GND (Ground) мінус живлення.

ПЗ - Програмне забезпечення.

ВСТУП

В сучасному світі дуже гостро стоїть питання економії енергоресурсів та, зокрема, електроенергії. Більшість світової електроенергії виробляється на теплових електростанціях. Вони в свою чергу спалюють вугілля, що є невідновним ресурсом. Спалювання вугілля також призводить до забруднення атмосфери викидами CO2, що, в свою чергу, негативно впливає на екологію усієї планети. З кожним роком людство споживає все більше і більше електроенергії, тому для вирішення цієї проблеми шукаються різні рішення. Одним з цих рішень є перехід пішохідних вулиць та автомобільних доріг на системи розумного освітлення.

Розумне освітлення - технологія освітлення, спрямована на збільшення енергоефективності та комфорту використання штучних джерел світла, що досягається завдяки використанню автоматизованого управління, датчиків освітленості, а також можливостей планування, акцентування і сучасних способів взаємодії з людиною та іншою технікою.

Системи автоматизації освітлення з'явилися ще в кінці 20 століття, але появу «розумного освітлення» можна приурочити лише до 2010-2012 років, коли почалося активне використання LED-ламп. Технологія LED дозволила не менше ніж в 5 разів знизити енерговитратність освітлення, в порівнянні з лампами розжарювання, а так само значно (до 100 разів) збільшити довговічність одного використовуваного джерела.

Завдяки цим властивостям світлодіоди повсюдно витіснили навіть люмінесцентні лампи. У 2015-2017 роках почалося активне оснащення джерел світла модулями зв'язку. Це дає їм змогу тісно взаємодіяти з IoT технікою і користувачем, що розширює можливості останнього.

Використання датчиків освітленості в колі освітлення дозволяє реалізувати найпростішу функцію «розумного освітлення» - самостійно вмикати і вимикати освітлення тоді, коли це потрібно, що активно використовується в вуличному освітленні. Додавши датчики руху або інфрачервоні камери, світло розпочне включатися при появі або знаходженні людини в приміщенні.

Завдяки пульту або смартфону користувач може налаштовувати освітлення потрібних ділянок та задавати час освітлення цих же ділянок після активації датчика.

1 ПОСТАНОВКА ЗАДАЧІ

Метою даної роботи є дослідження та аналіз переваг впровадження системи розумного освітлення як компонента системи Smart City та розробка такої системи. Для досягнення цієї мети було сформульовані такі завдання:

- дослідити схожі системи та їх характеристики;
- сформувати список вимог до клієнтської (серверної) частини системи;
- сформувати список вимог до апаратної частини системи;
- вибрати інструменти та засоби розробки, які найкраще підійдуть до сформованих вимог;
- створити серверну частину та базу даних;
- реалізувати зручний клієнтський інтерфейс для керування системою;
- створити модульну апаратну частину;
- з’єднати серверну частину з апаратною, тим самим створивши апаратно-програмний комплекс управління та моніторингу доріг;

Також були складено вимоги до розробки серверної частини системи. Основні з них - це те що система повинна:

- легко масштабуватись;
- мати модулі, які легко можна замінити;
- бути легкою в розробці, покращенні та подальшій підтримці;
- надавати зручну та інтуїтивно зрозумілу панель керування для користувача;
- використовувати надійні технології;

Відповідно, головними вимогами до створення апаратної частини стали:

- можливість легкого з’єднання усіх частин;
- простий зв’язок з сервером;
- можливість заміни конкретного модуля на інший;

- легкість та надійність розробки;

2 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Визначення розумного освітлення

Розумне освітлення - це спеціальна система для інтерактивного ввімкнення освітлення на вулицях та дорогах, створена за допомогою датчиків руху та спільного комп'ютера для забезпечення освітлення ділянок доріг, де є рух, та для економії електроенергії.

Кількість нових доріг, які потребують освітлення, з кожним роком зростає. Освітлення потребують навіть дороги, якими за ніч користується невелика кількість людей. З цим самим зростає кількість електроенергії, яка витрачається на освітлення. Зменшення споживання електроенергії допоможе містам зекономити гроші та зробити внесок у зменшення шкідливих викидів у повітря теплоелектростанціями.

Поєднання датчиків з комп'ютером дає можливим реалізувати не просто ввімкнення світла біля конкретного датчика руху, а реалізувати ввімкнення інших світлових елементів та регулювати їх яскравість. Особливо корисним це є на перехрестях, де потрібно освітити усі напрямки руху.

Інша позитивна характеристика розумного освітлення це адаптивність. Залежно від погодних умов освітлення може вмикатись на максимальну яскравість. Наприклад, при погоді без опадів дорога не потребує максимального освітлення, а при тумані чи опадах варто вмикати максимальну яскравість.

Протягом року постійно змінюється довжина світлового дня. Традиційне освітлення вмикається відповідно до певної години, що є не завжди комфортним. Розумне освітленя може вмикатись автономно після заходу сонця.

2.2 Огляд існуючих аналогів

2.2.1 “Смарт” освітлення від Benish GPS

Впровадження розумного освітлення цією компанією відбувається в три етапи. Спочатку місто обирає скільки електроенергії воно хоче економити та як має відбуватись керування процесом освітлення. Першим етапом є заміна традиційних ламп розжарювання на LED-лампи. Це, в свою чергу, дає змогу економити до половини витрат на електроенергію, тому що LED-лампи перетворюють в світлову енергію майже всю отриману енергію. Наприклад, звичні нам лампи розжарювання перетворюють в світло малу частину отриманої енергії - близько 5%, а інші 95% перетворюється в тепло.

Наступним етапом переходу на розумне освітлення є створення програмного забезпечення для LED-ламп, яке дозволяє переводити їх в “режим затемнення”, тобто світіння не в повну силу. Це дозволяє налаштовувати освітлення на 4 пори року: зиму, весну, літо та осінь. Налаштування кожного з режимів залежить від середньої довжини світлового дня даного сезону. Компанія визначає час роботи ламп на повну потужність, світіння в “режимі затемнення” і час, в який освітлення буде повністю виключено. Цей етап дозволяє економити ще близько 35-40% від витрат електроенергії.

Третій і останній етап - адаптивне освітлення. Для кожного світильника встановлюється система, коли диспетчер може самостійно регулювати освітлення: вмикати, вимикати та затемнювати. Встановлення такого виду освітлення дуже дорога і зможе окупитись в Україні лише за 10-15 років.

Створення в містах системи розумного освітлення несе в собі низку переваг перед звичайним освітленням, а саме:

- Покращує якість освітлення вулиць
- Підвищує енергоефективність і економічність роботи елементів освітлення майже на половину
- Забезпечує місто інтелектуальною мережею управління, яка є початком для створення “smart city”
- Ефективно проводить збір інформації та управління базами даних
- Адаптує систему освітлення до погодних умов
- Знижує частоту аварій на дорогах

Представлена компанія працює в Україні уже протягом багатьох років і співпрацює з державними структурами. За час роботи вона впровадила свої розробки в Києві, Одесі, Миколаєві та Харкові.

2.2.2 Освітлення від компанії Signify

Компанія займається впровадження мережевих систем освітлення у містах та наданням відповідного програмного забезпечення для керування освітленням.

Interact – платформа Інтернету речей компанії, яка за допомогою мережевих систем освітлення та програмного забезпечення, надає клієнтам не лише світло, а й додаткові функції на базі IoT. Програмні інтерфейси Interact можуть поєднати мережеве освітлення з іншими системами керування та надавати послуги, пов’язані з аналізом даних.

2.2.3 Розумне освітлення від “БРАМА - ИНЖЕНЕРНЫЕ СИСТЕМЫ”

Принцип роботи «Розумного освітлення» полягає в тому, що освітлення адаптивно змінюється на основі даних інших систем: відключається або зводиться до мінімуму, якщо немає нікого в зоні освітлення; автоматично включається, коли машина (людина) перетинають зону освітлення.

Наприклад, раптове падіння температури і випадання снігу ускладнюють видимість (мерехтіння снігу в світлі ламп) і дорожню обстановку. Система розумного вуличного освітлення знижує інтенсивність освітлення, щоб запобігти осліплення водіїв або підвищує в разі ожеледі без снігу, щоб мінімізувати ризики дорожніх аварій

Інший приклад - робота в надзвичайних ситуаціях. Система управління дорожнім рухом виявляє інцидент або аварію, інформація надходить в єдиний ситуаційний центр. На місці аварії рівень освітлення автоматично збільшується до 100%, в той час як найближчі до місця аварії лампи починають блимати, попереджаючи водіїв про необхідність звернути увагу і знизити швидкість.

Система розумного освітлення інтегрована з інформаційними панелями і цифровими дорожніми знаками, які можуть, наприклад, заборонити парковку на певній стороні вулиці в години пік для поліпшення дорожньої обстановки.

А ось приклад того, як «Розумне освітлення» працює в комплексі «Розумного міста». Громадянин повідомляє про інцидент в ситуаційному центрі або найближчому опорному пункті поліції, спрацьовує тривога і включається система відеоспостереження, щоб ідентифікувати і виявити злочинця. Персонал ситуаційного центру підвищує потужність вуличного освітлення, щоб убезпечити зону, в якій знаходиться потерпілий і для камер, які відстежують злочинця.

3 ЗАСОБИ РОЗРОБКИ

Для розробки були обрані інструменти, які дозволили швидко та надійно створити необхідну систему. Популярність вибраних інструментів гарантувала великий обсяг спільноти, яка може відповісти на запитання, та велику кількість доповнень, які легко можна інтегрувати в систему. Простота інструментів уможливлює доробки та адаптування системи під конкретні вимоги споживача.

3.1 Мова програмування Python

У сервера даної системи є тільки один постійний клієнт - Arduino, яка відправляє сигнали датчиків. Тобто на сервер не буде великого навантаження, що дає змогу обрати будь-яку мову програмування по швидкодії. І враховуючи велику спільноту програмістів, безліч бібліотек та простоту коду для читання і написання такою мовою став Python

Python - це високорівнева інтерпретована мова програмування. Філософія дизайну Python підкреслює читабельність коду завдяки помітному використанню відступам. Його мовні конструкції та об'єктно-орієнтований підхід мають на меті допомогти програмістам написати чіткий логічний код для малих та масштабних проектів. [1]

Python є мовою з динамічною типізацією та автоматичним вивільненням пам’яті від об’єктів, які не будуть використовуватись в подальшому. Тут присутні декілька парадигм програмування, зокрема структурне, об’єктно-орієнтоване та функціональне. Python часто називають “швейцарським ножем” через наявність вбудованих та сторонніх бібліотек та фреймворків для будь-яких задач в будь-яких сферах розробки.

Основні переваги Python:

- Дана мова включає в себе всі сучасні напрямки програмування. Також Python постійно розвивається: добавляються нові конструкції, нові модулі, нові функції та постійно збільшується швидкість виконання коду. Цьому сприяє велика спільнота розробників, які пишуть власні бібліотеки з відкритим кодом, і які з часом включаються в базовий набір бібліотек мови.
- Мова є простою для вивчення програмістам з різними досвідами роботи. Новачки можуть спробувати написання програм в будь-яких сферах програмування не боячись помилок. Досвідчені програмісти уже після години вивчення мови зможуть створити корисні та функціональні для себе скрипти для роботи з вебом, системним адмініструванням та іншим необхідним.
- Мова має в наявності велику кількість вбудованих та сторонніх бібліотек. Програмістам, зазвичай, не доводиться “вигадувати велосипед” - бібліотеки для машинного навчання, системного адміністрування, роботи з файлами різних типів, роботи з соціальними мережами та веб-фреймворки уже існують. Потрібно тільки вибрати, що саме найкраще підійде під кожну конкретну задачу.
- Надзвичайно простий синтаксис та красота коду. Окремі блоки коду виділяються чотирма пробілами, тому зрозуміти структуру коду є набагато простіше. Те що у інших мовах (Java, C++) вважається хорошим тоном, в Python це частина синтаксису.
- Python також є мультиплатформенним. Це гарантує запуск програмного продукту з усіма залежностями на більшості операційних систем: Mac OS, Linux, Windows та навіть на мобільних платформах. Дану властивість забезпечує інтерпретатор, який і перекладає код написаний на Python в машинні команди.

3.2 Фреймворк Django

Сервер даної системи не потребує особливого алгоритму роботи, тому було вирішено обрати найпопулярніший та найповноцінніший за кількістю вбудованого функціоналу веб-фреймворк для мови програмування Python - Django.

Django - це високорівневий веб-фреймворк, який дозволяє створювати повноцінні веб-додатки настільки швидко, наскільки це можливо. Django включає в себе десятки додаткових модулів, таких як: авторизація, адміністрування контенту, керування правами користувачів та багато інших - “прямо з коробки”.

Django є надійним, він серйозно ставиться до безпеки та допомагає розробникам уникнути багатьох помилок. Наприклад, вбудована ORM не дає можливості для sql-ін'єкцій. Його система аутентифікації користувачів забезпечує безпечний спосіб управління обліковими записами та паролями.

Цей фреймворк дозволяє легко масштабувати проект, щоб забезпечити стабільну роботу при великому навантаженні.

Компанії, організації та уряди використовують Django для побудови всіляких речей - від систем управління контентом до соціальних мереж.

Django впевнено можна віднести до списку MVC-фреймворків, тому що він досить точно відповідає структурі MVC. Model, View і Controller на прикладі фреймворку розподілені наступним чином:

- Model - частина доступу до баз даних.
- View - частина, яка відповідає за вибірку що і як показувати користувачу. Що показувати визначає view, а як показувати описується в шаблонах - templates.
- Controller - частина, яка вибирає view, залежно від запиту користувача. Django самостійно займається цим, використовуючи заданий список залежностей функцій Python від конкретних посилань.

Зважаючи на те, що за “Controller” відповідає сам фреймворк, а сама логіка описується в моделях (Model), представлення (View) та шаблонах (Templates), то Django може називатись MTV фреймворком.

Згідно цієї концепції:

- Model - також рівень даних, та усього, що з ними пов’язано: отримання доступу, створення нових та редагування існуючих записів у базі даних, і видалення. На цьому рівні описуються відношення між моделями та перевірки з валідацією даних.
- Template - шаблони. На базі даного рівня описується усе, що пов’язане з рендером даних на веб-сторінках: як повинні відображатись дані, де саме та у якому форматі.
- View - рівень представлення інформації, що є проміжним рівнем між необхідними моделями і шаблонами. Саме тут описується уся логіка даних, які підуть до шаблону: звідки їх дістати, як опрацювати чи перетворити, які зробити висновки та що відправити як результат. [3]

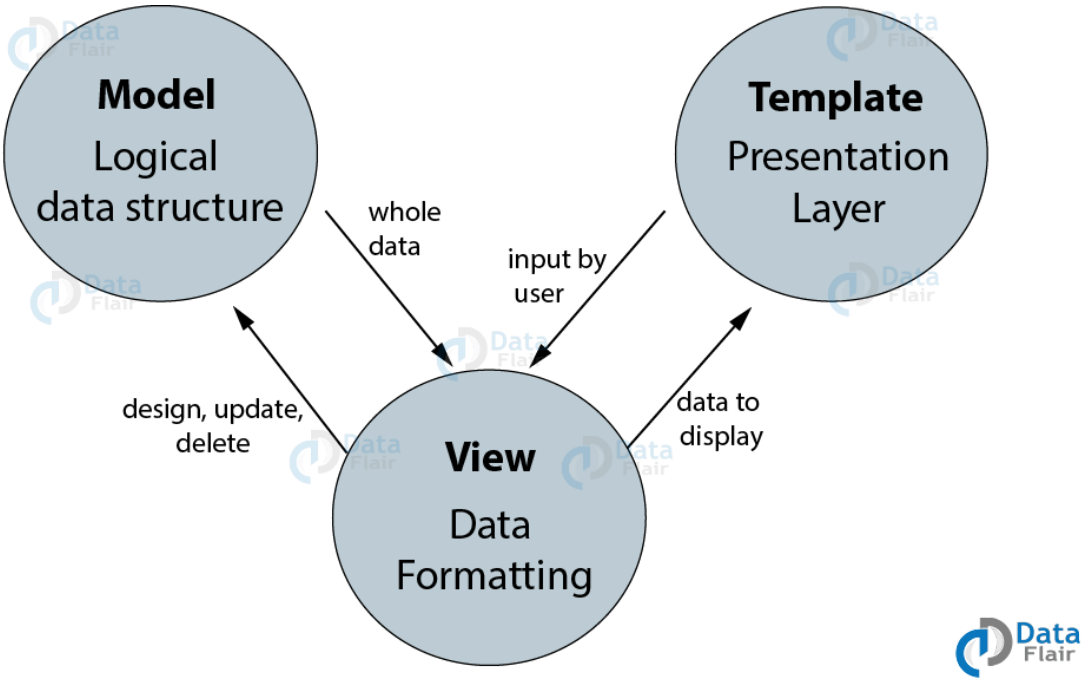

Рис 2.1 MTV модель

3.3 Обчислювальна платформа Arduino

Створення системи розумного освітлення включало в себе також створення апаратної частини - платформи, яка б розпізнавала сигнали з датчиків руху, відправляла їх на сервер та подавала б сигнали на реле для ввімкнення елементів освітлення на основі аналізу відповіді сервера.

Було розглянуто два популярних варіанта для даних потреб: Arduino та Raspberry Pi. Вибір був за Arduino, оскільки він має перевагу в простоті: вам не потрібно нічого налаштовувати, збирати повноцінну операційну систему Linux, просто пишіть код, компілюйте його і завантажуйте на пристрій.

Arduino (Ардуіно) — це спеціальна платформа для обчислень та конструювання різноманітних систем з використанням апаратної частини. Дана платформа складається з двох частин. Першою є фізична плата з мікроконтролером та елементами вводу/виводу. Друга частина це середовище розробки Processing/Wiring на мові програмування, яка схожа на C/C++, але спрощена. На даній платформі можна створювати як автономні системи, так і системи з використанням підключення до зовнішніх джерел інформації - комп’ютера чи інтернету. Уся інформація про платформу: програмне забезпечення, особливості елементів, рисунок фізичної плати - знаходяться у відкритому доступі для усіх охочих. [5]

На платі розміщено мікроконтролер Atmel AVR та інші необхідні елементи для підключення різних пристроїв та програмування. Тактова частота становить 16 або 8 МГц. Здійснює її кварцовий резонатор. На платах встановлені лінійні стабілізатори напруги 3,3 / 5 В. В мікроконтролері встановлено bootloader (завантажувач), тому зовнішній програматор не потрібен. Arduino дає змогу користуватись великою кількістю входів та виходів через свої виводи. Головна ідея проектування Ардуіно - це можливість під’єднання зовнішніх розширень та модулів. Такі елементи називаються “shields” та значно розширюють можливості самої

плати. Модулі розширення підключаються з допомогою роз'ємів, які встановлені на самому модулі. Доступні також спеціальні плати, які дозволяють жорстке з'єднання Arduino та модулів розширення в стопку через штирові лінійки. До того ж існують різні форм-фактори плат для спеціальних задач. Зменшені версії носять назву Nano, а збільшені носять назву Mega.

Середовище розробки для Arduino це кросплатформний додаток на Java. До складу цього додатку входить три елементи: компілятор, редактор коду та модуль для прошивки плати. Середовище розробки засноване на мові аналогічній мові Wiring - Processing. Сама мова програмування спроектована таким чином, щоб розробникам без досвіду було легше розпочати створення корисних продуктів. Також мова схожа на C++ з додаванням кількох бібліотек. Програми написані на цій мові спочатку обробляються препроцесором, а потім піддаються компіляції з допомогою AVR-GCC.

Середовище розробки поставляється з базовим набором бібліотек. Для початку роботи розробника з даною платформою потрібно описати лише дві функції. З їх допомогою програма буде працювати за циклічним принципом.

— setup(): функція виконується лише раз при старті програми і дозволяє задати початкові параметри

— loop(): функція виконується періодично, доки плата не буде вимкнена.

З усіх варіантів плат Arduino була обрана плата Arduino Mega 2560. Такий вибір зумовлений більшою кількістю, порівняно з іншими платами, входів та виходів. В даній роботі це дає змогу підключити більшу кількість датчиків та елементів освітлення для покриттям системою більшої території вулиці.

Arduino Mega 2560, зображений на рисунку 2.2, - це пристрій на основі мікроконтролера ATmega2560. Він має 54 порту введення виведення, 15 з яких можуть працювати, як джерело ШІМ сигналу, для плавного регулювання потужності, струму, швидкості, яскравості, в загальному, все, що можна регулювати

за допомогою широтно-імпульсної модуляції, плюс до цього 16 аналогових портів можуть обробляти сигнали з датчиків, використовуватися, як цифровий вихід.

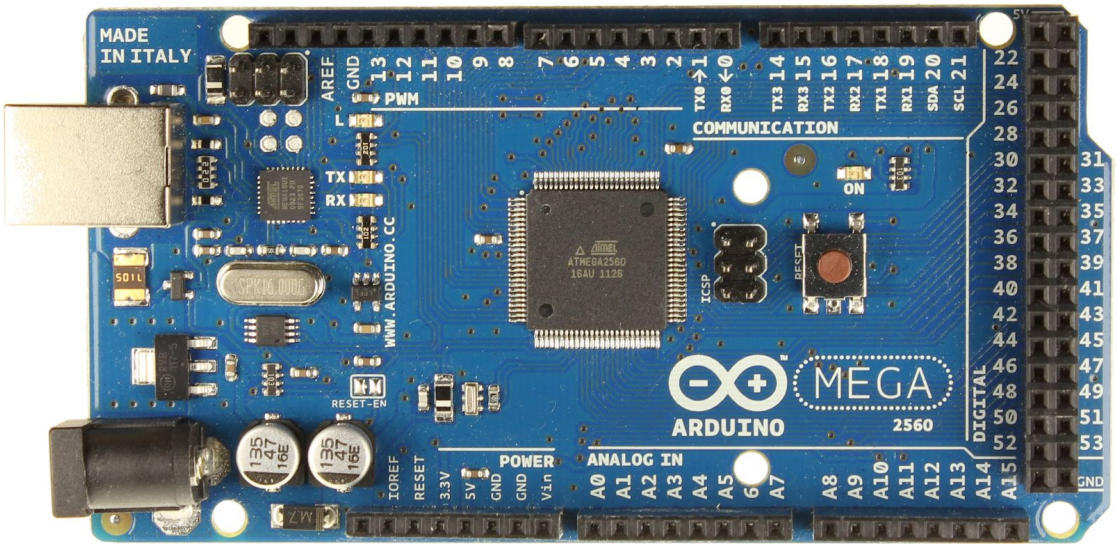

Рис. 2.2 - Arduino Mega 2560

Для зв'язку між різними пристроями передбачено цілих 4 UART інтерфейсу, в їх ролі виступають висновки 0, 1, 14-19. Один з портів спрямований на USB через мікроконтролер ATmega8U2 - він тут застосований замість звичного по молодшим платам USB-TTL контролера, а його прошивка доступна для вільного завантаження. Для зв'язку з різними дисплеями та іншими виконавчими пристроями передбачена SPI і I2C технології.

Atmega2560 - це дуже потужний чіп. У розпорядженні розробника цілих 256 кб Flash (в Ардуіно 8 кб займає завантажувач), 8 кб SRAM і 1 кб EEPROM. Працює Ардуіно з таким чіпом на частоті 16 мГц, втім, як і молодші плати - UNO і багато інших.

Живлення плати може здійснюватися як від круглого роз'єму живлення 2.1 мм з плюсом по центру, так і від USB порту, джерело вибирається автоматично.

Варто відзначити, що при напрузі живлення 7-20 вольт, плата працює відмінно, а при меншому, наприклад, 5 вольт, можуть виникнути ситуації з нестабільною роботою.

Програмний код для мікроконтролера завантажується з допомого студії Arduino IDE.

3.4 Система керування базами даних Postgres

PostgreSQL (Постгрес) — одна з об’єктно реляційних систем керування базами даних. Вона становить альтернативу багатьом іншим СКБД. Серед яких є і комерційні так і некомерційні. Проте порівнюючи навіть з базами даних з відкритим кодом, PostgreSQL не підконтрольна ніякій з компаній. А розробка та розвиток системи відбувається завдяки спільній роботі великою кількості людей та корпорацій, які користуються даною СКБД та прагнуть зробити її ще кращою. [7]

PostgreSQL забезпечує розробників багатьма функціональними перевагами над іншими СКБД. Основною характеристикою об'єктно-реляційної бази даних є підтримка визначених користувачем об'єктів та їх поведінки, включаючи типи даних, функції, оператори, домени та індекси. Це робить PostgreSQL надзвичайно гнучким та надійним. Крім усього іншого, складні структури даних можуть створюватися, зберігатися та витягуватися.

Існує великий список типів даних, які підтримує PostgreSQL. Крім типових чисел, з плаваючою комою, рядкових, булевих та датових типів, PostgreSQL може похвалитися uuid, грошовим, переліченим, геометричним, двійковим, мережевою адресою, бітовим рядком, пошуком тексту, xml, json , масиви, складові та типи діапазонів, а також деякі внутрішні типи для ідентифікації об'єкта та розташування журналу. Для справедливості, MySQL, MariaDB і Firebird мають деякі з них у різній мірі, але лише PostgreSQL підтримує їх. PostgreSQL має багато можливостей. Побудований за допомогою об'єктно-реляційної моделі, він підтримує складні структури та широту вбудованих та визначених користувачем типів даних. Він забезпечує велику ємність даних і довіряє своїй цілісності даних.

3.5 Інтернет модуль ENC28J60 Ethernet

Для забезпечення обміну даними між апаратною частиною (Arduino) та сервером потрібно було вибрати інтернет модуль. Головний вибір був між дротовим

зв’язком та бездротовим. Вибрано було модуль дротового інтернет з’єднання, так як для доступу в інтернет, цей модуль достатньо просто підключити до інтернет-роутера. Серед доступних дротових інтернет модулів було вибрано міні-версію модуля ENC28J60 Ethernet.

Характеристики:

- чіп ENC28J60 Ethernet, корпус SOP28
- робоча частота 25МГц
- SPI інтерфейс
- 2x5 роз'єм підключення до пристрою
- вбудований роз'єм RJ45
- харчування: +3.3 В
- розміри плати 50x15 мм

Для використання модуля з бібліотекою EtherCard, потрібно підключити його наступним чином:

- Вивід 10 Ардуіно до висновку CS модуля
- Вивід 11 Ардуіно до висновку SI модуля
- Вивід 12 Ардуіно до висновку SO модуля
- Вивід 13 Ардуіно до висновку SCK модуля
- Вивід 5V Ардуіно до висновку VCC модуля
- Вивід GND Ардуіно до висновку GND модуля

EtherCard - драйвер мікросхеми Microchip ENC28J60, сумісний з Arduino IDE. Він дозволяє на високому рівні працювати з передачею даних через HTTP.

3.6. Набір шаблонів HTML та CSS Bootstrap

Для створення графічного інтерфейс потрібно було вибрати інструмент, який би спрощував створення інтерфейсу та надавав змогу швидко змінювати інтерфейс відповідно до потреб. Іншим аспектом вибору була можливість створення адаптивного інтерфейсу під мобільні платформи. Для вирішення даного завдання було обрано Bootstrap.

Bootstrap - це велика колекція багаторазових та універсальних фрагментів коду, які написані у CSS, HTML та JavaScript. Оскільки це також фреймворк, всі основи вже закладені для адаптивної веб-розробки, і все, що потрібно зробити розробникам, - це вставити код у заздалегідь задану сітку. [11]

Це позбавляє розробників від необхідності писати довгі рядки особливо CSS-коду. Також це дає користувача фреймворку більше часу, а також можливостей для роботи над розробкою самих веб-сторінок.

Веб-розробники вже багато років віддають перевагу адаптивним дизайнам, оскільки адаптивні веб-сайти значно приваблюють статичні. Масивна зміна переваг прийшла пізніше, коли розробка мобільних веб-сайтів набрала темп і стала майже такою ж поширеною, як звичайний веб-дизайн. Фреймворк допоміг створювати адаптивні веб-сторінки та цілі веб-сайти без глибоких знань в сфері адаптивності.

Функції, що реагують на розмір екрану, дозволяють веб-сторінкам правильно відображатися на різних екранах (мобільний, планшетний, настільний тощо). Зображення також автоматично регулюються відповідно до екрана перегляду за заздалегідь визначеними інструкціями CSS з додаванням певних класів.

Також сам фреймворк, крім створення сіток та адаптивних дизайнів, пропонує розробникам цілий набір різноманітних готових компонентів для створення

приємного користувацького інтерфейсу. Серед цих компонентів є форми, навігаційні меню, кнопки, таблиці та багато інших.

3.7 Датчик руху HC-SR501

Для подання сигналів про рух об’єктів, машин чи людей, по вулиці чи дорозі потрібно було обрати датчик руху. Датчики руху бувають декількох типів, найпоширенішими є інфрачервоні. Для створення моделі системи було обрано саме такий найпростіший датчик руху, оскільки для відтворення системи в лабораторних умовах його достатньо.

Датчики руху працюють за принципом вимірювання інфрачервоного випромінювання людей та інших об'єктів. У датчиків апаратно можна налаштовувати відстань, на якій вони повинні реагувати на рух.

Рис. 2.2 Інфрачервоний датчик руху

На рисунку 2.2 зображено інфрачервоний датчик HC-SR501 призначений для вимірювання наявності руху на відстані від 4 до 7 м з ефективним кутом до 120^{o}.

Принцип дії інфрачервоного датчика заснований на аналізі теплового (інфрачервоного) випромінювання. Пасивний інфрачервоний датчик (PIR) при

цьому не випускає ніякого випромінювання, а лише аналізує приходять теплові промені.

Усередині датчика розташовуються зазвичай два чутливих елемента, що вимірюють потік інфрачервоного випромінювання. Перед кожним з чутливих елементів датчика встановлена лінза Френеля фокусуються на ньому падаючі на датчик інфрачервоні промені.

Найпростіший датчик сконструйований так, що зовнішній простір «розділене» між двома лінзами і чутливими елементами, кожна з лінз проектує теплове випромінювання зі своєї зони огляду на свій чутливий елемент. У звичайних умовах інтенсивність надходить на обидві частини датчика випромінювання приблизно однакова. Коли в поле зору з'являється випромінює інфрачервоні промені об'єкт (наприклад, людина), випромінювання спочатку потрапляє в поле зору тільки однієї частини датчика, при цьому свідчення двох чутливих елементів починають різнитися, і це є сигналом руху.

Особливістю саме цього датчика є можливість регулювати чутливість (4-7м) та час затримки сигналу (5-200с). Також є можливість отримання сигналів повторно, тобто, коли датчик ще подає сигнал і знову зреагував на рух, то таймер сигналу обнуляється і починає знову відлік.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Система розумного освітлення доріг складається з наступних головних частин:

1. Світлові елементи з датчиками
2. Мікрокомп'ютер
3. Сервер

Головним алгоритмом системи є те, що датчик подає сигнал на Arduino, той, в свою чергу, подає запит на сервер з даними щодо активних датчиків руху. Сервер приймає ці дані, дає запит у базу даних для отримання об’єктів датчиків. Далі відповідно кожного датчика, сервер дізнається, які лампочки необхідно увімкнути і відправляє результат в форматі JSON. Arduino обробляє отриману відповідь і подає сигнали на потрібні реле для ввімкнення елементів освітлення. На рисунку 3.1 схематично представлений алгоритм роботи системи.

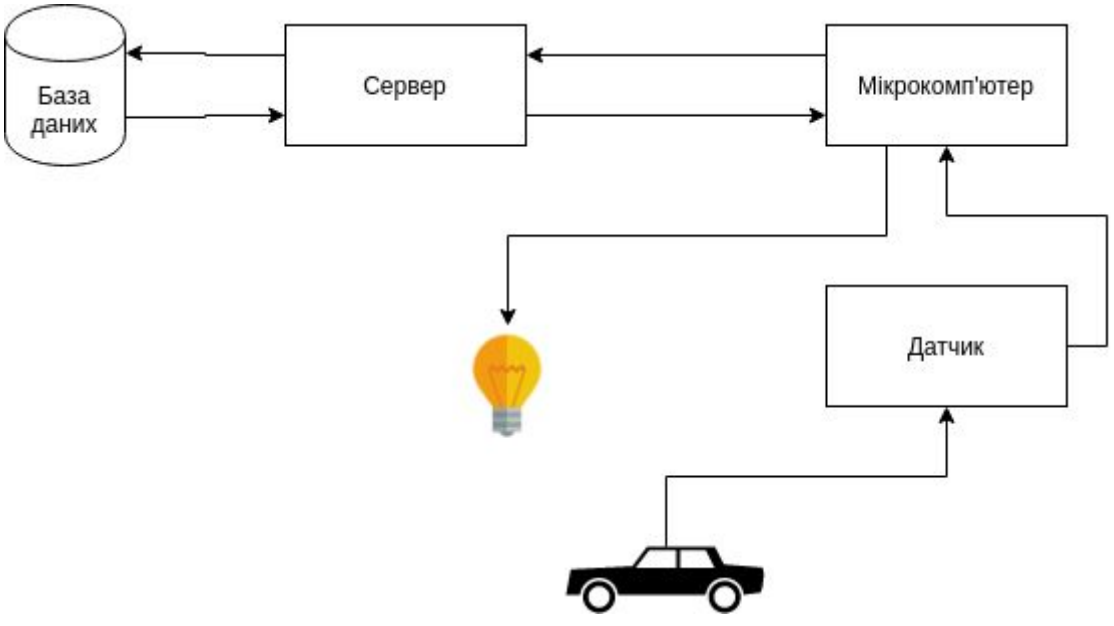

Рис. 4.1 - Алгоритм роботи системи

4.1. Веб-інтерфейс

Веб-додаток для керування та налаштування системи, виконаний на базі Python-фреймворку Django. Основним завданням веб-інтерфейсу є встановлення залежностей між датчиками та лампочками. Тобто вказати які конкретні лампочки повинні світитись при сигналі з певного датчика. Це дозволяє налаштувати розумне освітлення так, щоб воно освітлювало усі напрямки перехрестя або почало освітлювати пішохідний перехід за більшої відстані ніж зазвичай.

Також користувач веб-інтерфейса може побачити час роботи елементів освітлення та кількість сигналів від датчиків руху. За цими даними можна робити висновки про кількість нічних рухів машин та про зношення LED ламп.

Більшість даних рендериться на сервері за допомогою Django templates. Це дозволило створювати блоки коду та повторно їх використовувати. Як приклад був створений базовий шаблон з меню навігації. Інші шаблони уже розширяли базовий шаблон додаючи щось своє. Команда для розширення шаблону пишеться на початку документу і має вигляд : {% extends "light_control/admin_base.html" %}.

Шаблони Django також допомогли підключати статичні файли лише один раз, замість підключення їх у кожному файлі окремо. Була використана також спеціальна мова для створення шаблонів з динамічною інформацією. Наприклад, вивід лампочок, які будуть вмикатись від певного датчика виглядають наступним чином:

```
{% for bulb in sensor.bulbs %}
    {{ bulb }}
{% endfor %}
```

В даній системі існує розподілення дві ролі користувачі системи. Перша роль - це налаштовувач системи, людина яка займається монтажем системи та може підключати додаткові датчики і світильники. Друга роль - звичайний користувач, який може редагувати поведінку системи, але не може редагувати саму систему.

Налаштовувач має усі привілеї звичайного користувача. Можливості користувачів при взаємодії з веб-інтерфейсом системи зображені на рисунку 4.2.

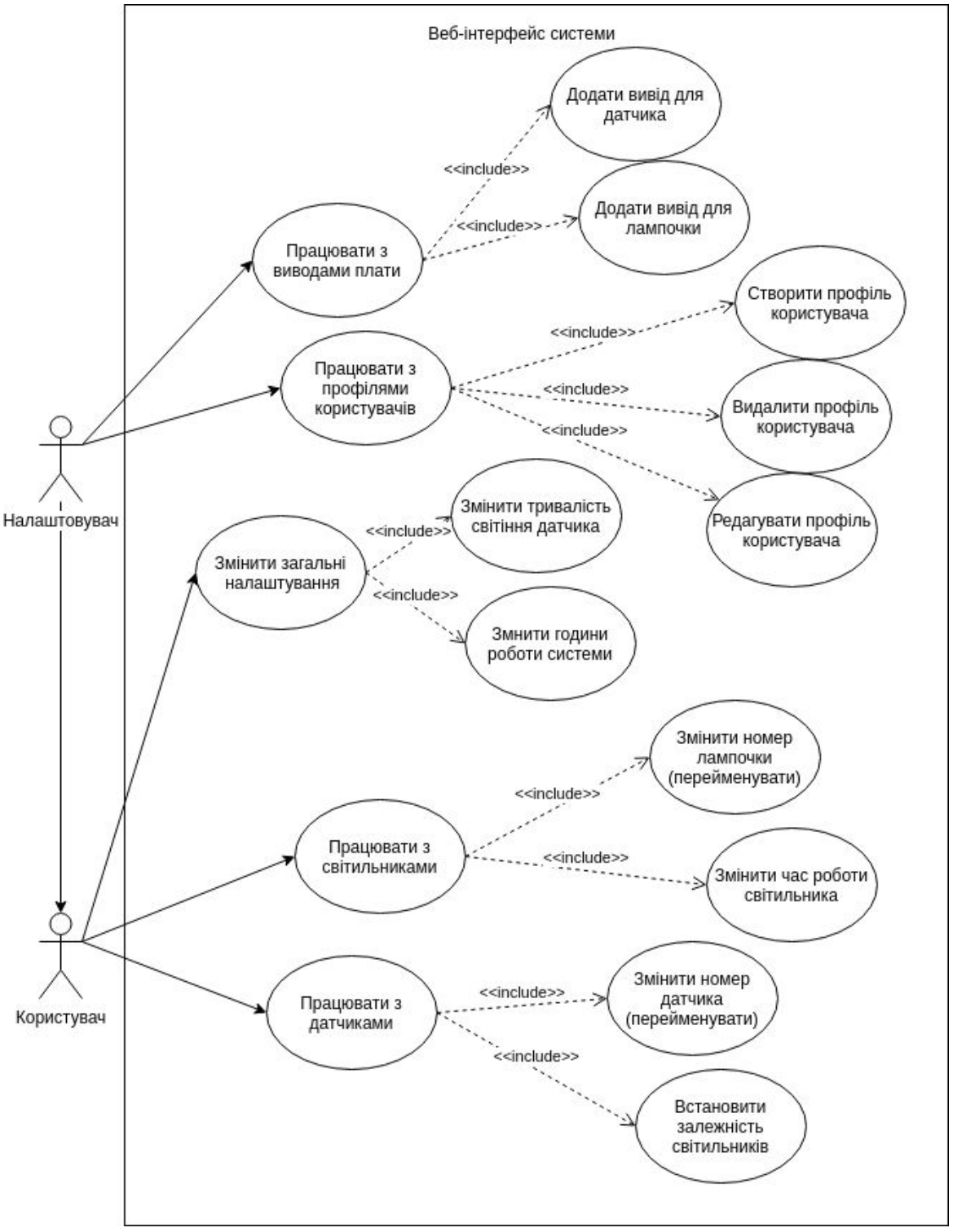

Рис. 4.2 - Діаграма прецедентів

4.2 Сервер

Архітектура додатку побудована на основі одного з найпопулярніших патернів проектування MVC (Model-View-Controller). Використовуючи цей патерн стає можливим чітке та логічне розподілення логіки роботи WEB-додатку.

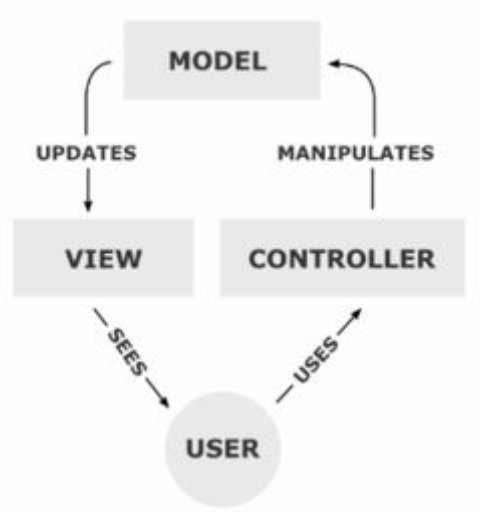

Рис. 4.3 Приклад роботи патерну MVC

Model View Controller - це модель архітектури програмного забезпечення, яка зазвичай використовується для реалізації інтерфейсів користувача. Тому це популярний вибір для архітектури веб-додатків. Загалом, вона розділяє логіку програми на три окремі частини, сприяючи модульності та простоті співпраці та повторного використання. Це також робить додатки більш гнучкими та дозволяє повторно використовувати компоненти.

Модель несе відповідальність за представлення ваших даних, відповідає за управління вашими даними.

Перегляд - це те, що бачить користувач, ця частина не повинна містити занадто багато логіки.

Контролер - це з'єднання між моделями та представленнями. Контролер зазвичай відомий як посередник між моделлю та поданням.

Дві найбільші цілі MVC - це одночасна розробка та повторне використання коду. MVC дозволяє одночасно розробляти, оскільки він роз'єднує різні компоненти програми, що дозволяє розробникам паралельно працювати над різними компонентами, не впливаючи і не заважаючи один одному. Наприклад, в інженерній команді може існувати розподіл розробників - одні розробляють клієнтську частину, інші - серверну. За допомогою MVC розробники серверної частини можуть спроектувати структуру даних та спосіб взаємодії користувача з ними без доповнення користувальницького інтерфейсу. Далі можливе повторне використання коду з MVC шляхом створення компонентів, незалежних один від одного. Це дозволяє розробникам можливість швидко та легко використовувати компоненти в інших додатках. Той самий або подібний компонент для однієї програми може бути зроблений для використання іншою програмою з іншими даними.

MVC вставляє посередника (контролера) між представленням та моделлю для усунення залежностей Model-View. В результаті Модель та Вид можуть бути використані повторно без змін. Це допомагає та спрощує простоту впровадження нових функцій та обслуговування.

Оскільки Django дає змогу працювати з MVC, то Controller для операцій з лампою виглядає наступним чином:

```
path('light_bulbs', login_required(LightBulbListView.as_view()), name='light_bulbs'),
path('create_light_bulb', login_required(LightBulbCreateView.as_view()), name='create_light_bulb'),
path('update_light_bulb/<int:pk>', login_required(LightBulbUpdateView.as_view()), name='update_light_bulb'),
path('remove_light_bulb/<int:pk>', remove_light_bulb, name='remove_light_bulb'),
```

Рис. 4.4 - Controller в Django

Перший аргумент функції path() фреймворк використовує для пошуку відповідності шляху посилання. Другий аргумент - це функція, яка буде обробляти даний запит. Аргумент name це назва функції, для використання її у шаблонах, наприклад: {% url 'light_bulbs' %} - посилання на функцію відображення списку лампочок.

4.3 Апаратна частина

Створення апаратної частини починається з подання живлення на Arduino. Далі він з’єднується з макетною платою таким чином, щоб вивести живлення VCC 5V і GND і на макетній платі в різні ряди розставляються дроти для подальшого підключення в них датчиків руху та реле. Усі датчики руху та реле мають 3 контакта, два для живлення від 5V та GND та один для виходу чи входу сигналу, відповідно.

Елементи освітлення, в даній системі це LED-лампи, підключаються тільки паралельно від елемента живлення, де один з контактів йде напряму від елемента живлення, а інший проходить через реле. В даній системі, елементом живлення є блок живлення на 12V, і саме ці 12V підключені через реле.

Для доступу в інтернет використовується спеціальний модуль , який підключається до Arduino 4 дротами та потребує живлення 3.3V. Даний модуль за допомогою мережевого дроту RJ45 підключається до роутера.

В програмному коді Arduino спочатку виконується ініціалізація виходів і входів командами pinMode(Номер піна, режим роботи). Для роботи з контактом як з отримувачем сигналу режим роботи встановлюється INPUT, а для роботи як з виходом - OUTPUT. В початкових налаштуваннях також відбувається ініціалізація бібліотеки для роботи з мережею EtherCard.

```
ether.browseUrl(
  PSTR("/command/?active="),
 activeSensorsCh,
website,
my_callback
);
```

Рис. 4.5 - Функція запиту на сервер

В функції loop() Arduino постійно зчитує значення входів з датчиками руху. Після отримання формується строка з номерів датчиків, які подали сигнал та відправляється на сервер, як на рисунку 3.6. Першим аргументом йде посилання на спеціальний шлях для обробки запиту, другим параметром йде змінна у якій наявні номери датчиків, з яких поступив сигнал, третій аргумент це посилання на сайт, четвертий аргумент це функція обробки відповіді сервера.

```
int start_json_index = answer_full.indexOf("{");
answer_full.remove(0, start_json_index);
Serial.println(answer_full);

 JsonObject& root = jsonBuffer.parseObject(answer_full);
```

Рис 4.6 - Функція парсингу відповіді сервера у хеш-таблицю

Як результат роботи коду, зображеного на рисунку 3.7, програма отримує змінну root, яка є хеш-таблицею, де ключі це номери лампочок (реле), які необхідно вмикнути чи вимкнути, а значення true чи false, відповідно до дії, яку необхідно виконати.

```
boolean sensor1 = root["1"];
boolean sensor2 = root["2"];
digitalWrite(11, sensor1);
digitalWrite(12, sensor2);
```

Рис. 4.7 - Передача даних на виходи Arduino

5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для користування даною системою знадобиться комп’ютер чи телефон з можливістю виходу в інтернет.

Рис. 5.1 - Форма логіну

Після успішного відкриття веб-додатку користувачу (оператору мережи освітлення) буде запропонована форма логіну (Рис. 5.1) для ідентифікації особи. Система не передбачає створення нових акаунтів користувачів, тому функція реєстрації в системі відсутня.

На кожній сторінці системи надане навігаційне меню зверху сторінки з можливістю вибору необхідного налаштування: лампи, датчики чи загальні налаштування системи. Також користувач може скористатись кнопкою виходу з системи та буде перенаправлений на сторінку входу. Меню, як і весь веб-сайт є адаптивним. Нижче представлені вигляди меню на комп’ютері (рис. 5.2) та вигляд меню на мобільному пристрої (рис. 5.3).

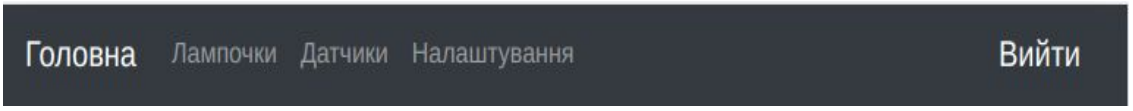

Рис. 5.2 Меню навігації на десктопі

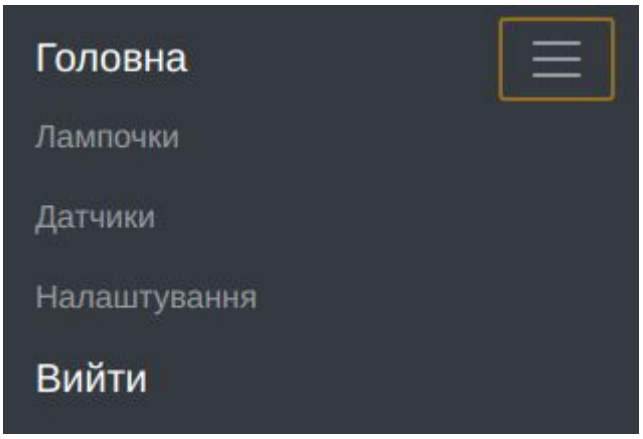

Рис. 5.3 Меню навігації на мобільному пристрої

Зайшовши в систему, користувачу буде відкрито список ламп таблицею з їх даними та можливістю додати, редагувати чи видалити певні лампи з бази даних.

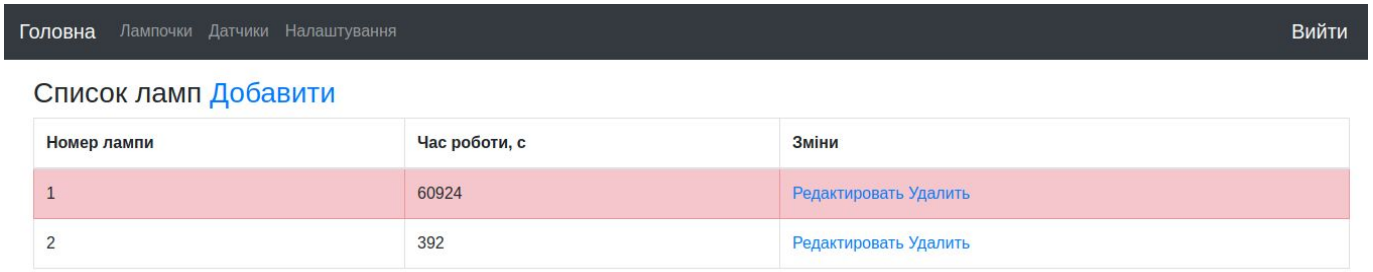

Рис. 5.4 - Список ламп

На рисунку 5.4 в таблиці відображені лампи з їх даними: номер та час роботи, та є кнопки, які надають функціонал редагування чи видалення конкретної лампи. Червоним кольором підсвічені лампи, які мають відпрацьований час більший, ніж зазначено у загальних налаштуваннях.

В меню сенсорів користувач системи може переглянути кількість спрацювань кожного датчика та визначити лампочки, які потрібні вмикатись при сигналі саме від цього датчика як на рисунку 5.5 (буде вмикатись 1, 2 і 5 лампочки)

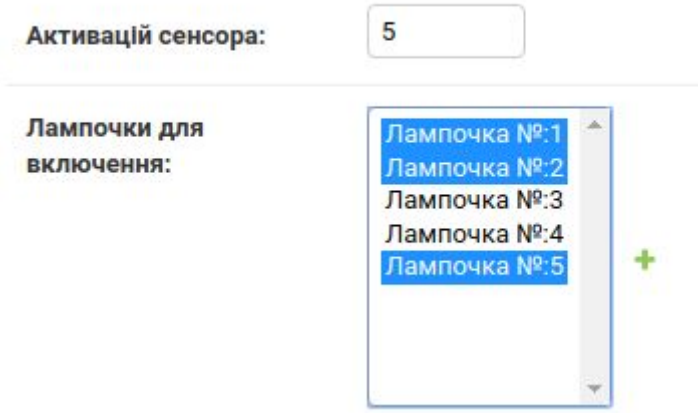

Рис. 5.5 - Меню датчика

В меню лампочки можна переглянути час її роботи та редагувати значення на потрібне. Наприклад, на 0, якщо встановлюється нова лампочка на дане місце, або на інше значення, якщо встановлюється б/в лампочка з відомим часом роботи.

В меню загальних налаштувань можна редагувати час світіння однієї лампочки після отримання сигналу з датчика та час роботи системи.

ВИСНОВКИ

Протягом виконання даної роботи були виконані поставлені завдання, а саме:

1. Проаналізовано існуючі системи керування і моніторингу освітлення.
2. Створені власні вимоги до системи розумного освітлення міста.
3. Були вибрані інструменти розробки, які задовільнили поставлені вимоги до системи.
4. Була розроблена серверна частина системи з допомогою Python-фреймворка Django та база даних з використанням PostgreSQL для керування залежностями між датчиками та світильниками.
5. Створено адаптивний клієнтський інтерфейс з використанням Bootstrap для взаємодії клієнта з системою
6. Було створено апаратну частину системи на базі платформи Arduino Mega 2560 з використанням інтернет модуля для підключення до сервера та інфрачервоних датчиків руху для визначення руху об’єктів.
7. Підключено апаратну частину до сервера, тим самим виконавши кінцеве завдання - створення апаратно-програмного комплексу управління та моніторингу доріг.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Python documentation [Електронний ресурс] — Режим доступу: https://docs.python.org/3/
2. David Beazley, Brian K. Jones. Python Cookbook: Recipes for Mastering Python 3
3. Nigel George. Mastering Django.
4. William S. Vincent. Django for APIs: Build web APIs with Python & Django.
5. Massimo Banzi. Getting Started with Arduino: The Open Source Electronics Prototyping Platform.
6. Michael Margolis. Arduino Cookbook.
7. PostgreSQL: Documentation [Електронний ресурс] — Режим доступу: https://www.postgresql.org/docs/
8. Luca Ferrari. PostgreSQL 11 Server Side Programming Quick Start Guide.
9. Cuno Pfister. Getting Started with the Internet of Things: Connecting Sensors And Microcontrollers To The Cloud.
10. Ethan Marcotte. Responsive Web Design.
11. Syed Fazle Rahman. Jump Start Bootstrap: Get Up to Speed With Bootstrap in a Weekend.
12. Martin Fowler. Patterns of Enterprise Application Architecture.

ДОДАТОК А

Апаратно – програмний комплекс інтелектуального управління та моніторингу освітлення доріг Smart City

Специфікація

УКР.НТУУ”КПІ ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ6143_20Б

Аркушів 1

Київ - 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ _АПЕПС_ТВ6143_20Б_81	Пояснювальна записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ _АПЕПС_ТВ6143_20Б 12-1	main.ino, views.py, models.py, forms.py	Основні компоненти
УКР.НТУУ”КПІ ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ _АПЕПС_ТВ6143_20Б 13-1	Додаток В.docx	Опис програмного модуля

ДОДАТОК Б

Апаратно – програмний комплекс інтелектуального управління та моніторингу освітлення доріг Smart City

Текст програми

УКР.НТУУ”КПІ ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ6143_20Б 12-1

Аркушів 7

Київ - 2020

```
#include <EtherCard.h>
#include <ArduinoJson.h>

// ethernet interface mac address, must be unique on the LAN
static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0x31 };

byte Ethernet::buffer[700];
static uint32_t timer;

const char website[] PROGMEM = "18bfa161f657.ngrok.io";

 StaticJsonBuffer<200> jsonBuffer;

// called when the client request is complete
static void my_callback (byte status, word off, word len) {
  Serial.println(">>>");

  Ethernet::buffer[off + 300] = 0;

  String answer_full = ((const char*) Ethernet::buffer + off);
  int start_json_index = answer_full.indexOf("{");
  answer_full.remove(0, start_json_index);
  Serial.println(answer_full);

   JsonObject& root = jsonBuffer.parseObject(answer_full);

//   Test if parsing succeeds.
  if (!root.success()) {
    Serial.println("parseObject() failed");
    return;
  }
  boolean sensor1 = root["1"];
  boolean sensor2 = root["2"];
  boolean sensor3 = root["3"];
  boolean sensor4 = root["4"];

  digitalWrite(11, sensor1);
  digitalWrite(12, sensor2);
  digitalWrite(14, sensor3);
  digitalWrite(15, sensor4);
  Serial.println("...");
}


void setup () {
   pinMode(2, INPUT);
   pinMode(3, INPUT);
   pinMode(4, INPUT);
   pinMode(5, INPUT);
   pinMode(6, INPUT);
   pinMode(20, INPUT);

   pinMode(11, OUTPUT);
   pinMode(12, OUTPUT);
   pinMode(14, OUTPUT);
   pinMode(15, OUTPUT);
```

```
  Serial.begin(57600);
  Serial.println(F("\n[webClient]"));

   if (ether.begin(sizeof Ethernet::buffer, mymac, SS) == 0)
    Serial.println(F("Failed to access Ethernet controller"));
  if (!ether.dhcpSetup())
    Serial.println(F("DHCP failed"));

  ether.printIp("IP:  ", ether.myip);
  ether.printIp("GW:  ", ether.gwip);
  ether.printIp("DNS: ", ether.dnsip);

#if 1
  // use DNS to resolve the website's IP address
  if (!ether.dnsLookup(website))
    Serial.println("DNS failed");
#elif 2
  // if website is a string containing an IP address instead of a domain name,
  // then use it directly. Note: the string can not be in PROGMEM.
  char websiteIP[] = "192.168.1.1";
  ether.parseIp(ether.hisip, websiteIP);
#else
  // or provide a numeric IP address instead of a string
  byte hisip[] = { 192,168,1,1 };
  ether.copyIp(ether.hisip, hisip);
#endif

  ether.printIp("SRV: ", ether.hisip);
}

void loop () {


  jsonBuffer.clear();
  ether.packetLoop(ether.packetReceive());

  if (millis() > timer) {
    timer = millis() + 1000;
    Serial.println();
    Serial.print("<<< REQ ");
      String sensors;
  boolean sensor1 = digitalRead(2);
  Serial.println(sensor1);
  if (sensor1) {
    sensors += "1+";
  }
  boolean sensor2 = digitalRead(3);
   if (sensor2) {
    sensors += "2+";
  }
  boolean sensor3 = digitalRead(4);
   if (sensor3) {
    sensors += "3+";
  }
  boolean sensor4 = digitalRead(5);
   if (sensor4) {
    sensors += "4+";
  }
```

```
  String myVarsStr;
  myVarsStr += sensors;
  char myVarsCh[40];
  myVarsStr.toCharArray(myVarsCh, 40);


  String url = "/command/?active=";
  url += sensors;
  Serial.println(sensors);
    ether.browseUrl(
      PSTR("/command/?active="),
     myVarsCh,
    website,
    my_callback
    );
  }
}
```

```
import datetime

import django
from django.contrib.auth.decorators import login_required
from django.contrib.auth.views import LoginView
from django.http import HttpResponse, JsonResponse
from django.shortcuts import render, redirect

# Create your views here.
from django.urls import reverse
from django.views.generic import CreateView, ListView, UpdateView

from .forms import LightBulbForm, IndexLoginForm
from .models import Sensor, LightBulb, Config


def command(request):
    active_from_request = request.GET.get('active')
    activated_sensors_ids = active_from_request.split()

    utc_now_datetime = datetime.datetime.now(datetime.timezone.utc)
    activated_sensors = Sensor.objects.filter(number__in=activated_sensors_ids).all()
    for sensor in activated_sensors:
        sensor.count_of_activations += 1
        sensor.save()
        config = Config.objects.first()
        sensor.bulbs.all().update(
            light_to=utc_now_datetime +
datetime.timedelta(seconds=config.time_of_shine),
            last_time_updated=utc_now_datetime
        )

    light_bulbs_to_activate = {}
    bulbs = LightBulb.objects.all()
    for bulb in bulbs:
        if bulb.light_to >= utc_now_datetime:
            light_bulbs_to_activate[bulb.pk] = True
            add_worked_time = utc_now_datetime - bulb.last_time_updated
            bulb.time_worked += add_worked_time.seconds
        else:
            light_bulbs_to_activate[bulb.pk] = False
        bulb.last_time_updated = utc_now_datetime
```

```
        bulb.save()
    return JsonResponse(light_bulbs_to_activate, status=200)


def hello(request):
    return redirect(reverse("light_bulbs"))
    return HttpResponse("Here's the text of the Web page.", status=200)


class LightBulbCreateView(CreateView):
    template_name = 'light_control/state_.html'
    form_class = LightBulbForm
    model = LightBulb

    def get_success_url(self):
        return reverse('state_links')


class LightBulbListView(ListView):
    template_name = 'light_bulbs.html'
    model = LightBulb
    paginate_by = 10
    context_object_name = 'light_bulbs'


class LightBulbUpdateView(UpdateView):
    template_name = 'panel/state_link.html'
    form_class = LightBulbForm
    model = LightBulb

    def get_success_url(self):
        return reverse('state_links')


@login_required
def remove_light_bulb(request, pk):
    state = LightBulb.objects.filter(pk=pk).first()
    state.delete()
    return redirect(reverse('state_links'))


class PanelLoginView(LoginView):
    form_class = IndexLoginForm
    template_name = 'light_control/login.html'


from django.urls import path

# from .views import PanelLoginView
from .views import *
from django.contrib.auth import views as auth_views

urlpatterns = [
    path('command/', command, name='command'),
    path('', hello, name='hello'),

    path('light_bulbs', login_required(LightBulbListView.as_view()),
name='light_bulbs'),
    path('create_light_bulb', login_required(LightBulbCreateView.as_view()),
name='create_light_bulb'),
```

```
    path('update_light_bulb/<int:pk>', login_required(LightBulbUpdateView.as_view()),
name='update_light_bulb'),
    path('remove_light_bulb/<int:pk>', remove_light_bulb, name='remove_light_bulb'),

    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
    path('login', PanelLoginView.as_view(), name='login'),
]


import datetime

from django.db import models

# Create your models here.


class LightBulb(models.Model):
    number = models.PositiveSmallIntegerField(unique=True, verbose_name='Номер
лампочки')
    time_worked = models.PositiveIntegerField(default=0, verbose_name='Час роботи')
    light_to = models.DateTimeField(default=datetime.datetime(year=2000, month=1,
day=1))
    last_time_updated = models.DateTimeField(default=datetime.datetime(year=2000,
month=1, day=1))

    def __str__(self):
        return f'Лампочка №:{self.number}'


class Sensor(models.Model):
    number = models.PositiveSmallIntegerField(unique=True, verbose_name='Номер
датчика')
    count_of_activations = models.PositiveIntegerField(default=0,
verbose_name='Активацій сенсора')
    bulbs = models.ManyToManyField(LightBulb, verbose_name='Лампочки для включення')

    def __str__(self):
        return f'Датчик номер №:{self.number}'


class Config(models.Model):
    brightness = models.PositiveIntegerField(default=100, verbose_name='Яскравість')
    time_of_shine = models.PositiveSmallIntegerField(default=10, verbose_name='Час
світіння лампочки')




<!DOCTYPE html>
{% load static %}
<html>

<head>
    {% block head %}
        <meta charset="UTF-8">
        <title>{% block title %}{% endblock %}</title>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
```

```
integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
crossorigin="anonymous"></script>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        {% block afterhead %}{% endblock %}
    {% endblock %}
</head>


<body>
<header>
    <nav class="navbar navbar-expand-md bg-dark navbar-dark fixed-top shadow_head">
        <a class="navbar-brand" href="{% url 'light_bulbs' %}">
          Головна
        </a>
        <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#collapsibleNavbar">
            <span class="navbar-toggler-icon"></span>
        </button>
        {% if user.is_authenticated %}
            <div class="collapse navbar-collapse" id="collapsibleNavbar">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'light_bulbs' %}">Лампочки</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'light_bulbs' %}">Датчики</a>
                    </li>
                         <li class="nav-item">
                        <a class="nav-link" href="{% url 'light_bulbs'
%}">Налаштування</a>
                    </li>

                </ul>
                <a class="navbar-brand" href="{% url 'logout' %}?next={% url 'login'
%}" style="font-family: Geneva, Arial, Helvetica, sans-serif">Вийти</a>
            </div>


        {% endif %}
    </nav>
</header>
<br>
<div class="spacer">
     
</div>
{% block content %}{% endblock %}
<script src="{% static 'js/styling.js' %}"></script>
<script src="{% static 'jquery-3.3.1.min.js' %}"></script>
<script src="{% static 'bootstrap/js/bootstrap.min.js' %}"></script>
<script src="{% static 'js/admin.js' %}"></script>
{% block afterscripts %}{% endblock %}
```

```
</body>

</html>

{% extends "light_control/admin_base.html" %}
{% load static %}
{% block title %}Настройка світильників{% endblock %}

{% block content %}
    <div class="container-fluid">
        <br>
        <div class="container-fluid">
            <h3>Список ламп <a href="{% url 'create_light_bulb' %}">Добавити</a></h3>
            <div class="table-scroll">

                <table class="table table-striped table-bordered" id="table_users"
style="width:100%">
                    <thead>
                    <tr>
                        <th>Номер лампи</th>
                        <th>Час роботи, с</th>
                        <th>Зміни</th>
                    </tr>
                    </thead>
                    <tbody id="table-data">
                    {% for bulb in light_bulbs %}
                        <tr class="table-danger">
                            <td>{{ bulb.number }}</td>
                            <td>{{ bulb.time_worked }}</td>
                            <td><a href="{% url 'update_light_bulb' bulb.pk
%}">Редактировать</a> <a href="{% url 'remove_light_bulb' bulb.pk %}">Удалить</a></td>
                        </tr>
                    {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>

    </div>
{% endblock %}
```

ДОДАТОК В

Апаратно – програмний комплекс інтелектуального управління та моніторингу освітлення доріг Smart City

Опис програми

УКР.НТУУ”КПІ ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ6143_20Б 13-1

Аркушів 9

Київ - 2020

АНОТАЦІЯ

Пояснювальна записка містить 30 сторінок, включає 15 рисунків, 3 додатки та 12 посилань.

Мета дипломної роботи заключається в створенні системи інтелектуального управління та моніторингу освітлення доріг та клієнтського додатку для взаємодії з системою.

Результатом роботи є апаратно-програмний комплекс для управління та моніторингу освітленням доріг, розроблений з допомогою Python-фреймворка Django та на платформі Arduino.

Ключові слова: розумне освітлення, апаратна платформа, система моніторингу та управління, економія електроенергії, Smart City.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ **52**

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ **53**

ОПИС ЛОГІЧНОЇ СТРУКТУРИ **54**

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ **55**

ВИКЛИК І ЗАВАНТАЖЕННЯ **56**

ВХІДНІ ДАНІ **57**

ВИХІДНІ ДАНІ **58**

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку опис основних компонентів апаратно-програмного комплексу інтелектуального управління та моніторингу освітлення доріг Smart City. У другому додатку міститься програмний код певних модулів.

Розроблена система працює як інтернет додаток і не залежна від платформи користувача. Для розгортання даної системи потрібно зробити наступне:

1. Для роботи сервера: встановити Python 3.6, фреймворк Django та відповідні залежності.
2. Для роботи баз даних необхідно створити базу даних з встановленими даними користувача та провести міграції.
3. Для роботи апаратної частини потрібно завантажити код на мікроконтролер Arduino.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Дана система керує включенням світильників, відповідно до сигналів отриманих з датчиків руху. Веб-інтерфейс системи надає можливість користувачеві:

1. Змінювати залежності між датчиками.
2. Налаштовувати режими освітлення.
3. Переглядати дані по світильникам.
4. Розширювати дану систему.

Розроблений апаратно-програмний комплекс може бути використаний на вулицях та дорогах країни, на пішохідних зонах, приватних територіях та навіть у будинках.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Для створення даної системи було використано мікроконтролер Arduino, серверний додаток на Python-фреймворку Django та клієнтський веб-інтерфейс. Для комунікації між сервером та мікроконтролером використовується спеціальний інтернет модуль. Він дає можливість робити HTTP запити через мережу на сервер. Arduino.

Клієнтська частина реалізована з шаблонізатором фреймворку та може динамічно змінювати дані.

Серверний код організований за архітектурним паттерном MVC і складається з обробників де знаходиться уся бізнес логіка системи.

Апаратно система з’єднана через макетну плату. Це робить систему гнучкою та дає переваги в подальшій розробці чи масштабуванні.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Програмний код для серверної та клієнтської частини розроблявся у інтегрованому середовищі розробки PyCharm. Код для мікроконтролера Arduino розроблявся в спеціальному додатку Arduino IDE. Уся розробка виконувалась на комп’ютері з операційною системою Linux Mint. Використовувана мова програмування - Python, фреймворк - Django, база даних - PostgreSQL. Веб-інтерфейс розроблено з використанням HTML5 і CSS3,

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для роботи даної системи потрібне встановлення датчиків руху та світильників у відповідних місцях. Мікроконтролер повинен бути підключений до інтернету. Для користувача-адміністратора, щоб керувати системою, потрібно лише перейти за посиланням додатку системи, де після авторизація з’явиться меню керування системою.

ВХІДНІ ДАНІ

Вхідними даними для користування системою є встановлення залежностей ввімкнення світильників від сигналів з певних датчиків. Іншими вхідними даними є встановлення режимів світіння та час світіння лампочок. Для ввімкнення світильника вхідними даними є рух об’єктів: людей чи машин.

ВИХІДНІ ДАНІ

Вихідними даними є інформація про систему та зв’язок світильників і датчиків та інформація про час роботи світильників.