Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Вебдодаток для моніторингу пристроїв розумного будинку

(тема)

Виконала:
здобувачка *4* року навчання,
групи *ТРІМІз-21-1*

Дар'я ГРІНЬ
(власне ім'я, прізвище)

Спеціальність *172 Телекомунікації та радіотехніка*
(код і повна назва спеціальності)

Тип програми *освітньо-професійна*

Освітня програма
«Інформаційно-мережна інженерія»
(повна назва освітньої програми)

Керівник *ст. викл. Ляшенко Галина Євгеніївна*
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри _____ *Валерій БЕЗРУК*
(підпис) (власне ім'я, прізвище)

2025 р.

Не містить відомостей заборонених до відкритого публікування.

Здобувачка / *Дар'я ГРІНЬ* /

Керівник / *Галина ЛЯШЕНКО* /

Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

Рівень вищої освіти *перший (бакалаврський)*

Спеціальність *172 Телекомунікації та радіотехніка*
(код і повна назва)

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *«Інформаційно-мережна інженерія»*
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«__»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачці *Грінь Дар'я Володимирівна*
(прізвище, ім'я, по батькові)

1. Тема роботи *«Вебдодаток для моніторингу пристроїв розумного будинку»*

затверджена наказом університету від «*02*» *травня* 2025 р. № 63 *Стз*

2. Термін подання студентом роботи до екзаменаційної комісії *20 червня 2025 р.*

3. Вихідні дані до роботи HTML, CSS, JS, Python, C++, ARDUINO Uno. Провести моделювання системи розумного будинку та створити вебдодаток для моніторингу пристроїв розумного будинку.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ
1. Аналіз стану проблеми та існуючих рішень.
2. Розробка користувацького інтерфейсу.
3. Розробка апаратної частини системи.
4. Опис програмного коду мікроконтролера Arduino.
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *Слайди у форматі Power Point*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	02.05 – 03.05.25	*виконано*
2	*Підбір літератури за темою роботи.*	04.05 – 10.05.25	*виконано*
3	*Виконання розділу 1*	11.05 – 20.05.25	*виконано*
4	*Виконання розділу 2*	21.05 – 29.05.25	*виконано*
5	*Виконання розділу 3*	30.05 – 11.06.25	*виконано*
6	*Виконання розділу 4*		*виконано*
7	*Оформлення пояснювальної записки*	12.06 – 20.06.25	*виконано*
8	*Оформлення презентаційного матеріалу,*		
	підготовка до захисту у ЕК, захист.	21.06 – 26.06.25	*виконано*

Дата видачі завдання *02 травня 2025 р.*

Здобувачка _____
(підпис)

Керівник роботи _____ *Ляшенко Галина Євгеніївна*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 72 с., 6 рис., 0 табл., 34 джерела, 2 додатки.

Об’єкт дослідження – програмне забезпечення для розумного будинку.

Мета роботи – розробка вебдодатку для моніторингу пристроїв розумного будинку, який забезпечуватиме зручний інтерфейс для взаємодії з системою в режимі реального часу.

У роботі проведено аналіз технологій для IoT та засобів комунікації між пристроями; реалізовано апаратну частину на основі Arduino Uno з використанням сенсорів PIR і LDR, серводвигунів та світлодіодів; створено вебдодаток із графічним інтерфейсом у Figma; протестовано функціональність системи в середовищі Tinkercad; сформульовано рекомендації щодо подальшого вдосконалення. Акцент зроблено на забезпеченні доступності, безпеки та зручності взаємодії користувача з системою.

РОЗУМНИЙ БУДИНОК, ІНТЕРНЕТ РЕЧЕЙ, IoT, MQTT, ARDUINO, PIR, LDR, ВЕБДОДАТОК, ГРАФІЧНИЙ ІНТЕРФЕЙС, TINKERCAD, FIGMA

THE ABSTRACT

Explanatory note: 72 p., 6 fig., 0 tabl., 34 sources, 2 app.

Object of research – smart home software.

The purpose of the work is to develop a web application for monitoring smart home devices, providing the user with a convenient and informative interface for real-time interaction with the system.

The thesis analyzes current IoT technologies and communication tools; implements the hardware part based on Arduino Uno using PIR and LDR sensors, servos, and LEDs; creates a user interface in Figma and tests the prototype in Tinkercad. The system is designed with an emphasis on usability, accessibility, and security, with future development recommendations included.

SMART HOME, INTERNET OF THINGS, IoT, MQTT, ARDUINO, PIR, LDR, WEB APPLICATION, GRAPHICAL INTERFACE, TINKERCAD, FIGMA

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

IoT – Internet of Things – інтернет речей;

MQTT – Message Queuing Telemetry Transport – телеметричний протокол публікації/підписки;

HTTP – HyperText Transfer Protocol – протокол передачі гіпертексту;

HTTPS – HyperText Transfer Protocol Secure – захищений протокол передачі гіпертексту;

CoAP – Constrained Application Protocol – протокол для обмежених пристроїв;

DDS – Data Distribution Service – сервіс розподілу даних у реальному часі;

REST API – Representational State Transfer Application Programming Interface – програмний інтерфейс на основі REST;

PWM – Pulse Width Modulation – широтно-імпульсна модуляція;

IDE – Integrated Development Environment – інтегроване середовище розробки;

UART – Universal Asynchronous Receiver-Transmitter – універсальний асинхронний приймач–передавач;

JSON – JavaScript Object Notation – формат обміну даними;

XML – Extensible Markup Language – розширювана мова розмітки;

WCAG – Web Content Accessibility Guidelines – настанови щодо доступності вебконтенту;

UI – User Interface – інтерфейс користувача;

UX – User Experience – досвід користувача;

SSL – Secure Sockets Layer – протокол захисту передачі даних;

TLS – Transport Layer Security – протокол безпечного передавання даних;

QoS – Quality of Service – якість обслуговування;

PIR – Passive Infrared Sensor – пасивний інфрачервоний сенсор (датчик руху);

LDR – Light Dependent Resistor – світлозалежний резистор (фоторезистор);

LED – Light Emitting Diode – світлодіод;

Vcc – Voltage Common Collector – позитивне живлення електронного пристрою;

GND – Ground – загальний провід («земля»);

SG90 – Standard Gear 90 – модель сервомотора 9г;

SG02R – Servo Gear 02R – модель мікросервомотора, аналог SG90.

ЗМІСТ

ВСТУП

Сучасний етап розвитку цифрових технологій супроводжується активним впровадженням концепції «розумного будинку», яка набуває дедалі більшої популярності як у побуті, так і в комерційній сфері. Така система дозволяє автоматизувати управління освітленням, кліматом, охороною, побутовими приладами та іншими елементами інфраструктури житла. Головною її перевагою є підвищення рівня комфорту, енергоефективності, безпеки та можливість дистанційного керування з будь–якої точки світу [1].

Основою для реалізації подібних рішень є технології Інтернету речей (IoT), які забезпечують взаємодію між пристроями, передавання даних у реальному часі та віддалене керування за допомогою спеціального програмного забезпечення. Архітектура таких систем передбачає наявність мікроконтролерів, датчиків, мережевих протоколів зв’язку та програмного забезпечення, зокрема веб– або мобільних додатків [2]. Завдяки цьому користувачі мають змогу не тільки контролювати пристрої, а й задавати автоматичні правила їх роботи відповідно до змін у середовищі або особистих налаштувань.

Під час розробки систем розумного будинку особливу увагу слід приділити створенню інтерфейсу, який буде водночас зручним, функціональним і безпечним. Для цього дедалі частіше використовуються сучасні веб–технології – HTML5, CSS3, JavaScript, React.js, Python, а також REST API або WebSocket для з’єднання з пристроями. Крім того, важливо враховувати вимоги до захисту персональних даних, оскільки подібні системи оперують конфіденційною інформацією про користувача [3].

Актуальність дослідження зумовлена стрімким розвитком ринку розумних технологій і зростаючим попитом на інтерактивні системи керування пристроями. Згідно з прогнозами, найближчими роками цей ринок демонструватиме стрімкий розвиток, що, у свою чергу, формує необхідність у висококваліфікованих розробниках відповідного програмного забезпечення [4].

Метою кваліфікаційної роботи є розробка вебдодатку для моніторингу пристроїв розумного будинку, який забезпечуватиме користувачеві зручний та інформативний інтерфейс для взаємодії з системою в режимі реального часу. Об'єктом дослідження є програмне забезпечення для розумного будинку, а предметом – вебдодаток як засіб візуалізації та керування станом пристроїв.

Для досягнення поставленої мети необхідно виконати такі завдання: дослідити існуючі технології, які використовуються для створення вебдодатків у сфері IoT; проаналізувати засоби забезпечення зв’язку між сервером і пристроями, створити модель з використанням Arduino; створити прототип вебдодатку; провести тестування й оцінити результати.

1 АНАЛІЗ СТАНУ ПРОБЛЕМИ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Розвиток технологій Інтернету речей (IoT)

Інтернет речей (англ. Internet of Things, IoT) – це концепція, згідно з якою фізичні об’єкти отримують змогу взаємодіяти між собою та з цифровими системами через інтернет. Цього досягають шляхом інтеграції в об'єкти спеціальних сенсорів, мікроконтролерів, мережевих інтерфейсів та програмного забезпечення. Це дозволяє їм автоматично, без прямої участі людини, збирати, обробляти та передавати дані [5, 6].

Система Інтернету речей (IoT) складається з низки ключових компонентів, кожен із яких виконує визначену функцію в процесі збирання, передавання, оброблення та відображення даних. Їхня узгоджена робота забезпечує безперервне функціонування IoT–середовища [7].

Сенсори та пристрої – це елементи, відповідальні за первинне збирання даних із навколишнього середовища. Вони фіксують параметри, як–от температура, вологість, рівень освітленості, рух та інші фізичні характеристики. До цієї групи належать як прості датчики, так і більш складні пристрої, включно з відеокамерами та смарт–гаджетами. Саме з цих компонентів починається процес взаємодії в IoT–системі.

Засоби зв’язку (Connectivity) забезпечують передавання інформації від сенсорів до центральних вузлів оброблення даних. Залежно від архітектури системи та умов її експлуатації, можуть використовуватись різні типи з’єднання: Wi–Fi, Bluetooth, стільникові мережі (3G/4G/5G), Ethernet, Zigbee, LoRaWAN та інші бездротові протоколи. Надійне з’єднання між пристроями є критично важливим для стабільної роботи всієї системи [7].

Оброблення даних (Data Processing) – наступний етап, на якому отримана інформація аналізується за допомогою програмного забезпечення. Залежно від логіки функціонування системи, це може бути як проста перевірка відповідності даних заданим умовам, так і складна аналітика, що ґрунтується на алгоритмах

машинного навчання. Оброблення даних дає змогу формувати керівні команди для виконавчих пристроїв або генерувати повідомлення для користувача.

Інтерфейс користувача (User Interface) слугує засобом візуалізації даних і забезпечує взаємодію з кінцевим користувачем. За допомогою вебінтерфейсів, мобільних додатків або спеціалізованих панелей керування користувач може контролювати стан системи, переглядати аналітичні дані, а також вручну керувати окремими пристроями за потреби.

Таким чином, архітектура IoT–системи включає як фізичні пристрої, так і програмно–комунікаційні компоненти, взаємодія яких забезпечує автономну й інтелектуальну роботу в найрізноманітніших сферах застосування.

1.2 Пристрої розумного будинку: класифікація і функціональність

Розумний будинок складається з цілої низки взаємопов’язаних систем, кожна з яких виконує свою функцію в загальній структурі автоматизації житла. Одним із ключових елементів є сенсори та датчики, що фіксують параметри середовища: температуру, вологість, освітленість, рух тощо. Вони є основним джерелом даних для прийняття автоматичних рішень – наприклад, система може самостійно відрегулювати мікроклімат або активувати вентиляцію за необхідності.

До інших важливих компонентів належать розумні освітлювальні системи, що дозволяють дистанційно налаштовувати інтенсивність та колір освітлення через додаток або голосові команди, що, своєю чергою, сприяє енергозбереженню й створює бажану атмосферу у приміщенні. Розумні розетки та електроприлади також дають змогу контролювати побутову техніку з будь–якого місця, наприклад, вмикати кондиціонер чи вимикати пральну машину на відстані.

Не менш важливу роль у системі відіграють засоби безпеки – це відеокамери, датчики відкриття дверей і вікон, а також детектори диму чи витоку газу, які миттєво повідомляють власника про загрозу. Системи управління енергоспоживанням дозволяють аналізувати витрати та оптимізувати роботу

електроприладів і опалення, знижуючи загальне споживання ресурсів. Усі ці елементи можуть бути об’єднані голосовими помічниками – такими як Google Assistant, Amazon Alexa чи Apple Siri – які надають зручний спосіб управління функціями будинку через голосові команди. Додатково, система автоматизації дає змогу налаштовувати сценарії дій – наприклад, автоматичне ввімкнення світла зранку, запуск кавоварки або нагрівання приміщення до заданої температури. Комплекс доповнюють мультимедійні системи, що дозволяють організувати відтворення звуку та відео в будь–якій частині будинку – від перегляду фільмів до прослуховування музики у фоновому режимі. Така взаємодія між усіма компонентами створює єдину розумну екосистему, яка адаптується до потреб користувача та підвищує загальний рівень комфорту [8].

1.3 Основні стандарти та протоколи IoT

У рамках реалізації системи розумного будинку, ключову роль відіграє ефективний обмін даними між пристроями, що забезпечується протоколами IoT. Зокрема, MQTT застосовується для організації стабільного зв’язку між пристроями та сервером, що особливо важливо при низькій пропускній здатності мережі. Для взаємодії з вебдодатком може бути використаний HTTP або HTTPS, які дозволяють зручно реалізувати REST API для обміну даними між клієнтською частиною, написаною на React.js, та серверною логікою з базою даних MySQL. У випадках, коли необхідно забезпечити мінімальні затрати ресурсів, можливе використання CoAP або DDS для більш спеціалізованих задач в реальному часі [9].

1.3.1 MQTT – легкий протокол публікації/підписки

MQTT (Message Queuing Telemetry Transport) – це протокол відкритого стандарту, розроблений для надійного обміну даними між пристроями в мережах Інтернету речей. Його основна перевага полягає в простоті та легкості, завдяки

чому MQTT можна використовувати навіть на пристроях з обмеженими обчислювальними ресурсами та низьким споживанням енергії.

Протокол працює за принципом публікації та підписки: кожен пристрій може бути видавцем (publisher), підписником (subscriber) або поєднувати ці функції. Комунікація між пристроями відбувається через центрального брокера, який керує маршрутизацією повідомлень відповідно до заданих тем. Така архітектура дозволяє створювати гнучкі, масштабовані та ефективні системи для розподілених мереж.

MQTT особливо корисний у випадках, коли мережеві ресурси обмежені або з'єднання нестабільне. Саме тому цей протокол знайшов широке застосування в розумних будинках, де необхідно забезпечити стабільний зв'язок між багатьма пристроями – датчиками, виконавчими елементами та серверами. Крім того, MQTT чудово інтегрується з хмарними платформами, що дає змогу здійснювати віддалене керування, збір та аналіз даних [10].

Отже, MQTT – це важливий елемент більшості сучасних IoT–систем, що поєднує простоту, гнучкість і високу надійність при мінімальних витратах ресурсів.

1.3.2 HTTP/HTTPS – класичні протоколи вебвзаємодії

У вебдодатках HTTP (HyperText Transfer Protocol) та його захищена версія HTTPS (HTTP Secure) виступають базовими механізмами для обміну даними між клієнтом і сервером. Саме через них відбувається більшість запитів до серверної частини вебдодатку, зокрема під час роботи REST API. HTTP–запити дозволяють клієнту ініціювати звернення до певного ресурсу, а серверу – надати відповідь у стандартизованому вигляді, зазвичай у форматі JSON або XML.

Основною перевагою HTTP/HTTPS є його універсальність і підтримка з боку всіх сучасних браузерів, клієнтських бібліотек і фреймворків, таких як React.js. Це робить його зрозумілим і зручним для розробників. Використання HTTPS, у свою чергу, забезпечує захищений обмін даними шляхом шифрування трафіку за

допомогою SSL/TLS–сертифікатів. Це особливо важливо в додатках, які працюють з особистими або конфіденційними даними користувачів.

У контексті сучасних вебсистем, включаючи розумні будинки, HTTP/HTTPS є надійною платформою для інтеграції компонентів через REST API. Наприклад, клієнтський інтерфейс може надсилати запит до сервера для перевірки стану пристроїв, а сервер – повертати актуальну інформацію або виконувати відповідні дії. Таким чином, протоколи HTTP/HTTPS виконують ключову роль у підтримці динамічного, гнучкого та безпечного зв’язку між фронтендом і бекендом системи [11].

1.3.3 CoAP – протокол для пристроїв з обмеженими ресурсами

CoAP (Constrained Application Protocol) – це мережевий протокол, створений спеціально для застосування у пристроях із обмеженими технічними можливостями, таких як мініатюрні сенсори, мікроконтролери або автономні пристрої з живленням від батарей. Його головна особливість полягає в тому, що він споживає мінімум енергії та вимагає дуже мало обчислювальних ресурсів, що робить його особливо придатним для вбудованих систем, які використовуються в Інтернеті речей (IoT) [12].

За своєю структурою CoAP багато в чому нагадує HTTP, однак пристосований до вимог IoT. Протокол використовує модель «запит–відповідь», проте передача даних відбувається через UDP, а не TCP, що значно знижує затримки та спрощує реалізацію в слабких мережах [13]. Крім того, CoAP підтримує механізми підтвердження доставки повідомлень, а також можливість кешування й роботи з ресурсами за URI, як у звичному HTTP.

Завдяки своїй компактності та енергоефективності CoAP є ідеальним рішенням для сенсорних мереж, систем моніторингу та розумних будинків. У таких системах пристрої повинні працювати тривалий час без підзарядки, перебуваючи у сплячому режимі більшу частину часу. CoAP дозволяє зменшити навантаження на мережу та забезпечити стабільний обмін інформацією навіть у складних умовах.

1.3.4 DDS – протокол для реального часу

DDS (Data Distribution Service) – це високопродуктивний протокол публікації–підписки, розроблений для обміну даними в реальному часі без участі центрального брокера. Його головна мета – забезпечити надійне, швидке й масштабоване передавання інформації в критично важливих або промислових IoT–системах, зокрема в авіації, автомобілебудуванні, системах автоматизованого виробництва та робототехніці [14].

На відміну від HTTP, який орієнтований на запит–відповідь у класичних вебдодатках (див. п.1.3.2), DDS не потребує централізованого сервера, адже пристрої обмінюються даними безпосередньо один з одним. Це значно зменшує затримки, підвищує надійність і дозволяє масштабувати систему без втрати продуктивності.

У порівнянні з MQTT, який також працює за принципом публікації–підписки, DDS забезпечує вищу швидкість обміну та розширені можливості керування якістю сервісу (QoS), наприклад, затримками доставки, пріоритетністю або гарантованістю доставки повідомлень. Крім того, DDS краще підходить для систем, де важливою є точна синхронізація даних у реальному часі, тоді як MQTT орієнтований на економне використання ресурсів у нестабільних мережах.

Таким чином, DDS займає особливе місце серед IoT–протоколів, виступаючи потужним рішенням для сценаріїв, де критично важливі час реакції, надійність і відсутність єдиного вузла відмови.

1.4 Апаратна реалізація системи розумного будинку

Апаратна реалізація є ключовим етапом створення системи «розумного будинку», оскільки саме фізичні компоненти забезпечують взаємодію із зовнішнім середовищем та виконання заданих функцій. Від правильного вибору елементної бази залежить стабільність, енергоефективність та масштабованість усієї системи.

У межах цього проєкту апаратна частина побудована на основі мікроконтролерної платформи Arduino Uno Rev3, до якої підключаються

різноманітні сенсори та виконавчі пристрої. Основні компоненти включають: датчик руху (PIR) для фіксації присутності, фоторезистор (LDR) для визначення рівня освітленості, світлодіоди для індикації стану системи, а також сервомотор, який моделює дію фізичного механізму (наприклад, відкриття жалюзі).

Кожен з обраних модулів був детально проаналізований за критеріями сумісності, простоти інтеграції, точності роботи та відповідності умовам побутового використання. Усі компоненти підключаються до Arduino через макетну плату, що дозволяє швидко змінювати конфігурацію та експериментувати зі схемою під час етапу прототипування.

У подальших підрозділах описуються апаратні складові системи, їхні технічні характеристики, принципи роботи, варіанти підключення та обґрунтування вибору.

1.4.1 Основні апаратні компоненти системи

Для розробки апаратної частини системи розумного будинку було обрано платформу Arduino Uno, що забезпечує ефективну інтеграцію датчиків, виконавчих елементів та елементів управління. Основною перевагою Arduino є її доступність, відкритість програмного забезпечення та простота апаратного підключення, що дозволяє реалізовувати інженерні рішення без значних фінансових затрат [16]. Arduino підтримує роботу з численними цифровими та аналоговими модулями, зокрема сенсорами освітленості, температури, руху, а також сервомоторами, реле та індикаторами.

Крім того, Arduino має значну перевагу у контексті STEM–освіти, оскільки поєднує теоретичні знання з практичною діяльністю. Це дозволяє студентам та дослідникам не лише засвоювати програмування і логіку роботи електронних систем, але й створювати реальні пристрої, втілюючи свої ідеї у функціональні прототипи [15]. Завдяки використанню в освітньому процесі середовищ моделювання, таких як Tinkercad, значно спрощується тестування схем і забезпечується безпечне середовище для початкового навчання [16].

Практичне застосування Arduino стимулює розвиток критичного мислення, навичок вирішення задач і творчого підходу до проєктування, що є ключовими вимогами сучасної IT–індустрії. У роботі платформа Arduino розглядається як базовий контролер, до якого підключено ключові апаратні компоненти: сенсори присутності (PIR), датчики освітленості (LDR), світлодіоди, а також серводвигуни для керування виконавчими механізмами. Завдяки цьому стає можливою реалізація базових сценаріїв роботи інтелектуального середовища на основі реальних подій.

У цілому, Arduino Uno обрана як оптимальна платформа для побудови моделі розумного будинку, оскільки поєднує функціональність, гнучкість у застосуванні та методичну цінність у сфері інженерної освіти [16; 15].

1.4.2 Вибір апаратної платформи: Arduino Uno Rev3

У межах даної роботи в якості базової платформи для побудови апаратної частини системи було обрано Arduino Uno Rev3 – надійне рішення, що поєднує гнучкість використання з простотою реалізації вбудованих проєктів. Завдяки відкритій архітектурі, широкій підтримці периферійних модулів та активній спільноті, ця плата стала стандартом для моделювання пристроїв у сфері розумної електроніки та IoT–рішень [17, 18].

Плата побудована на мікроконтролері ATmega328P і оснащена всіма необхідними елементами для автономної роботи: стабілізатором напруги, кварцовим резонатором, роз’ємами для підключення цифрових та аналогових сигналів, USB–інтерфейсом для живлення і прошивки, а також кнопкою Reset для перезавантаження. На відміну від попередніх версій, Rev3 використовує мікроконтролер ATmega16U2 замість FTDI–чіпа для реалізації USB–UART інтерфейсу, що забезпечує гнучкіші можливості обміну даними з комп’ютером [19, 20].

Завдяки підтримці середовища розробки Arduino IDE, користувач може швидко створювати та завантажувати програми на основі мови C/C++ [17, 20]. Підтримка моделювання у середовищах, таких як Tinkercad, дозволяє тестувати

електронні схеми без фізичного обладнання, що особливо корисно на етапі розробки [2].

Основні технічні характеристики Arduino Uno Rev3 [17, 19]:

- Мікроконтролер: ATmega328P
- Напруга живлення: 7–12 В (через роз'єм), 5 В (через USB)
- Цифрові входи/виходи: 14 (6 з підтримкою PWM)
- Аналогові входи: 6
- Flash–пам'ять: 32 КБ (0,5 КБ зарезервовано для завантажувача)
- SRAM: 2 КБ
- EEPROM: 1 КБ
- Тактова частота: 16 МГц
- USB–інтерфейс: ATmega16U2
- Сумісність: Arduino IDE, Tinkercad

Arduino Uno Rev3 була обрана для реалізації даного проєкту завдяки поєднанню простоти використання, стабільної роботи, великої кількості документації та повній сумісності з сенсорами (PIR, LDR) і виконавчими пристроями (сервоприводи, світлодіоди), що використовуються в системі «розумного будинку».

1.4.3 Датчик присутності: PIR–модуль HC–SR501

Для реалізації функції виявлення присутності в системі обрано інфрачервоний датчик руху HC–SR501, що поєднує компактність, енергоефективність та високу чутливість до змін у навколишньому середовищі.

Інфрачервоний пасивний датчик руху HC–SR501 являє собою компактний модуль для виявлення рухомих об'єктів на основі аналізу їх теплового (інфрачервоного) випромінювання. У його основі лежить пироелектричний сенсор, який фіксує зміну інтенсивності ІЧ–променів, що проходять крізь структуру лінз Френеля. Завдяки цьому датчик здатний розбивати простір перед собою на численні сектори–зони, у кожній з яких автоматично аналізується надходження

теплового сигналу: при різкому зміні співвідношення сигналів між двома пироелектричними елементами модуль генерує цифровий вихідний імпульс, сигналізуючи про рух у зоні спостереження [21].

Модуль інфрачервоного пасивного датчика руху HC–SR501 вирізняється широким робочим діапазоном і гнучкими можливостями налаштування. Напруга живлення сенсора становить від 4,5 до 20 В постійного струму, а струм спокою – приблизно 50 мкА, що забезпечує економне енергоспоживання. Вихідний цифровий сигнал модуля має два стани: HIGH з рівнем близько 3,3 В та LOW – приблизно 0 В, що дозволяє безпосередньо зчитувати сигнал мікроконтролером без потреби в додаткових перетворювачах.

Датчик здатен виявляти рух у межах 3–7 метрів, при цьому відстань регулюється потенціометром T2, а кут огляду досягає 100°, що є достатнім для моніторингу середніх приміщень або вузьких коридорів. Тривалість вихідного сигналу налаштовується потенціометром T1 і може варіюватися від 5 до 300 секунд, дозволяючи адаптувати поведінку модуля до конкретного сценарію роботи. Після кожного спрацювання передбачено блокування повторного вимірювання тривалістю 2,5 секунди, що мінімізує помилкові спрацьовування при нестабільних умовах.

У модулі реалізовано два режими роботи, які перемикаються джампером: H (retriggering) забезпечує безперервне реагування на будь–який новий рух у межах зони детекції, тоді як L (non–retriggering) – дозволяє лише одиничне спрацьовування з необхідністю паузи перед наступною активацією.

Пристрій працює в широкому температурному діапазоні – від –20 °C до +80 °C, що робить його придатним для як побутових, так і промислових умов. Габарити модуля становлять 32,5 × 24,5 × 30 мм, діаметр лінзи Френеля – 23 мм, а висота – 18 мм. Для зручності підключення передбачено роз’єм Goldpin із кроком 2,54 мм, що дозволяє з'єднувати модуль як за допомогою «шлейфів», так і стандартних джамперів.

PIR–датчики широко застосовуються в різних сферах завдяки своїй ефективності та простоті інтеграції. У системах охоронної сигналізації вони

забезпечують надійне виявлення присутності людей як у приміщеннях, так і на відкритих територіях, при цьому кут огляду може досягати 120°, а дальність – до 7 м. У сфері автоматизації «розумного будинку» такі сенсори використовуються для автоматичного увімкнення й вимкнення освітлення, вентиляційного обладнання або кліматичних систем при фіксації руху, що сприяє економії енергії та підвищенню комфорту мешканців [21]. Стійкість до підвищеної запиленості, а також широкий температурний діапазон роботи (від –20 до +80 °C) дозволяють ефективно застосовувати дані модулі в промислових і аграрних комплексних рішеннях, зокрема в системах моніторингу переміщення персоналу та техніки на складських, виробничих і сільськогосподарських об’єктах [22]. Крім того, PIR–сенсори знаходять своє застосування в інтерактивних інсталяціях та робототехніці – зокрема в проєктах, де необхідно забезпечити реакцію на наближення або проходження об’єктів, як–от у роботизованих платформах чи інтерактивних експонатах [23].

1.4.4 Датчик освітленості: фоторезистор (LDR)

У складі системи моніторингу освітленості ключовим елементом виступає фоторезистор, що забезпечує чутливу реакцію на зміни інтенсивності світла в навколишньому середовищі.

Фоторезистор, або LDR–сенсор (Light Dependent Resistor), – це напівпровідниковий елемент, опір якого змінюється в залежності від рівня освітлення. При зростанні інтенсивності світла опір фоторезистора зменшується, а при зниженні – збільшується. Це дозволяє використовувати його як датчик освітленості в різноманітних автоматизованих системах [24, 25].

Принцип дії LDR базується на фотопровідності – фізичному явищі, за якого під впливом світлового потоку у напівпровіднику зростає кількість носіїв заряду, що зменшує електричний опір. Як правило, матеріалом для виготовлення LDR є кадмій сульфід (CdS) [25].

У проєкті системи «розумного будинку» LDR використовується для вимірювання рівня зовнішнього освітлення з подальшим керуванням виконавчими пристроями: світильниками, шторами або системами безпеки [25].

Основні технічні характеристики LDR–сенсора (на прикладі фоторезистора діаметром 12 мм на основі CdS):

Фоторезистори такого типу зазвичай мають опір у добре освітленому середовищі в межах від приблизно 100 Ом до 1 кОм, тоді як у темряві опір перевищує 10 МОм. Час реакції на зміну освітлення (час підйому та спадання сигналу) становить десятки мілісекунд, що є достатнім для більшості побутових і промислових застосувань. Як чутливий матеріал використовується кадмій сульфід (CdS), що забезпечує стабільну роботу у видимому спектрі світла, проте має екологічні обмеження щодо утилізації. Робочий температурний діапазон фоторезистора, як правило, становить від –25 до +75 °C, а максимальна допустима напруга прикладеного сигналу залежить від конкретної моделі, але часто становить до 150 В [25].

Серед основних переваг використання LDR–сенсора варто відзначити його простоту інтеграції з мікроконтролерними платформами, зокрема Arduino. Завдяки аналоговому виходу, фоторезистор легко підключається до аналогових входів без потреби у складній схемотехніці або перетворювачах сигналу. Крім того, компонент відзначається низькою вартістю та широкою доступністю, що робить його особливо привабливим для навчальних проєктів, прототипування та систем початкового рівня. Ще однією важливою перевагою є мінімальне енергоспоживання, яке дозволяє використовувати фоторезистори у пристроях з автономним живленням, зокрема у портативних рішеннях та енергоефективних модулях «розумного будинку».

1.4.5 Актуатори: сервомотор Servo SG02R

В системі «розумного будинку» застосовується мікросервопривід Servo SG02R, сумісний з популярним Micro Servo 9g (SG90). Цей мінісерводвигун

керується за допомогою широтно–імпульсної модуляції (PWM), що дозволяє здійснювати точне позиціонування до 180°, що є достатнім для автоматичного управління шторами, камерами тощо.

Живлення подається через 5 В, а для керування використовується вихід D2 Arduino. Сервомотор має три виводи: Vcc, GND, PWM. Згідно з офіційною документацією, PWM–імпульс шириною 1–2 мс відповідає повороту 0–180°[26].

Технічні характеристики Servo SG02R:

- Живлення: 4,8–6 В [2]
- Крутний момент: 1,8–2,2 кг·см (залежно від напруги) [26, 27]
- Швидкість: ~0,1 с/60° [26, 27]
- Вага: ~9–11 г [26, 28]
- Розміри: ~22×12×29–31 мм [26, 28]
- Температурний діапазон: –10…+50 °C [28]
- Інтерфейс керування: PWM, що повністю сумісний з Arduino

Компактний розмір, еквівалентність за характеристиками з широко відомою моделлю SG90, низька енергоспоживаність і простота інтеграції роблять пристрій SG02R ідеальним для використання у системах «розумного будинку».

1.4.6 Індикатори стану: світлодіоди з обмежувальними резисторами

У складі реалізованої системи передбачено використання двох світлодіодів: зеленого та червоного – для забезпечення базової візуальної індикації.

Зелений світлодіод сигналізує про активний стан системи або очікування – наприклад, коли освітлення вмикається за допомогою фоторезистора (у разі низького рівня освітленості).

Червоний світлодіод вказує на події, наприклад, при спрацюванні PIR-датчика або фіксації руху.

Обидва світлодіоди підключено до цифрових виходів плати Arduino (D9 і D10) через токообмежувальні резистори номіналом 220 Ом. Ці резистори є необхідним елементом схеми, оскільки забезпечують захист світлодіода від

надмірного струму, а також оберігають цифрові виходи мікроконтролера від перевантаження. Без резисторів світлодіоди можуть споживати струм, що перевищує допустимий рівень для Arduino, що призведе до перегріву або навіть пошкодження елементів.

Вибір значення резистора ґрунтується на законі Ома, який дозволяє обчислити опір на основі напруги живлення, прямої напруги світлодіода (Vf) і бажаного струму (If). Наприклад, для стандартного червоного світлодіода з Vf ≈ 2 В і струмом 15–20 мА при живленні від 5 В, оптимальний опір становить приблизно:

Однак у практичних схемах часто використовують резистори з більшим номіналом – 220 або 330 Ом – що забезпечує менший струм (~9–14 мА) та збільшує термін служби компонентів без втрати яскравості для побутового застосування[29, 30].

Таким чином використання світлодіодів у поєднанні з правильно підібраними обмежувальними резисторами дозволяє змоделювати керування світлом у розумному будинку.

2 РОЗРОБКА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

2.1 Аналіз вимог до інтерфейсу користувача

Розробка інтерфейсу користувача для системи розумного будинку передбачає врахування як функціональних, так і ергономічних характеристик, що забезпечують ефективну взаємодію людини з цифровим середовищем. Інтерфейс у подібних системах виконує роль єдиного каналу управління середовищем, надаючи користувачеві доступ до контролю пристроїв, перегляду інформації та налаштувань.

З урахуванням завдань, які виконує система, до функціональних вимог інтерфейсу належать:

- підтримка механізмів автентифікації користувача, зокрема можливість входу до системи, реєстрації нового облікового запису та відновлення пароля;
- відображення актуальної інформації щодо стану приміщень та підключених пристроїв (температура, освітлення, наявність руху тощо);
- забезпечення дистанційного керування функціональними елементами будинку: освітленням, вентиляцією, живленням;
- реалізація налаштування таймерів або створення розкладів роботи пристроїв;
- адаптивність інтерфейсу до різних типів пристроїв (мобільні телефони, планшети, десктопи), що забезпечує зручність використання в будь–яких умовах.

Окрім функціональних аспектів, важливу роль відіграє відповідність сучасним принципам зручності користування (UX). У процесі проектування було визначено такі ключові вимоги до ергономічної складової:

- інтерфейс повинен бути інтуїтивно зрозумілим, щоб не викликати труднощів у користувачів без спеціальної технічної підготовки;
- структура елементів має бути логічною і послідовною, з чітко окресленими зонами навігації та управління;

– графічне наповнення повинно бути візуально стриманим, без перевантаження деталями, що може викликати когнітивне перевантаження;

Усі компоненти інтерфейсу мають відповідати стандартам доступності WCAG 2.1, що гарантує комфортну роботу з системою для людей із порушеннями зору, моторики або кольоросприйняття.

На основі попереднього аналізу було прийнято рішення реалізувати інтерфейс у вигляді мобільного застосунку з чотирма основними екранами, кожен із яких виконує визначену функціональну роль:

– Start (стартова сторінка входу до системи. Містить поля для логіна та пароля, а також можливість реєстрації).

– Home (головна сторінка. Відображає загальну інформацію про стан будинку, температуру та навігацію по кімнатах).

– Living Room (сторінка управління окремою кімнатою з відображенням поточного стану пристроїв).

– Lighting (екран для детального налаштування освітлення: режими, яскравість, температура кольору, таймери).

Таким чином, структура та функціональність інтерфейсу були сформовані з урахуванням потреб кінцевого користувача, вимог до доступності та принципів сучасного UI/UX–дизайну, що дозволяє забезпечити ефективне, зручне та безпечне управління елементами розумного будинку.

2.2 Розробка графічного дизайну у Figma

Для розробки інтерфейсу було обрано інструмент Figma, оскільки він поєднує простоту у використанні, підтримку спільної роботи онлайн та зручну систему компонентів. Завдяки цьому середовищу вдалося не лише створити цілісну структуру застосунку, а й швидко візуалізувати ключові елементи взаємодії користувача з системою розумного будинку.

У межах роботи було створено чотири основні макети, кожен з яких відповідає за окремий етап користувацької взаємодії:

– Екран входу (Start) – лаконічна сторінка авторизації з полями для введення електронної пошти та пароля. Також передбачено посилання для відновлення доступу та створення нового облікового запису.

– Головна сторінка (Home) – це перше, що бачить користувач після входу. Інтерфейс вітає по імені, відображає поточну температуру в будинку та дату. Нижче розміщено блоки всіх кімнат у вигляді зрозумілих карток, натиснувши на які можна перейти до керування відповідним приміщенням.

– Сторінка кімнати (Living Room) – демонструє детальний опис обраної кімнати: перелік підключених пристроїв, їх поточний стан (наприклад, Wi–Fi увімкнено, температура 17°C, світло активне), а також інформацію про користувачів, які мають доступ до цієї кімнати.

– Управління освітленням (Lighting) – окремий екран для точного налаштування параметрів освітлення. Користувач може обирати між нічним та яскравим режимом, регулювати яскравість і температуру світла за допомогою повзунків. Додатково передбачено функції встановлення таймеру або створення розкладу роботи світла.

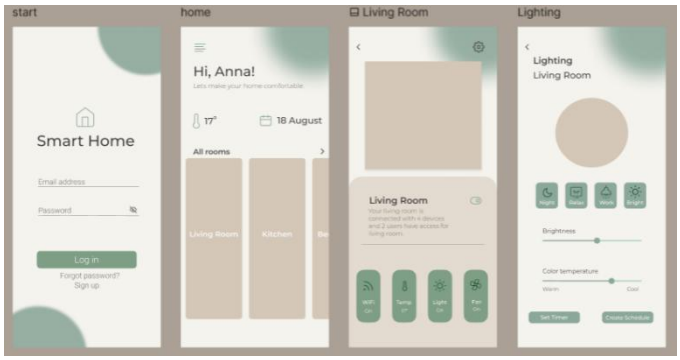

Рисунок 2.1 – Макети екранів мобільного застосунку Smart Home, створені у Figma

Завдяки гнучким можливостям Figma, усі ці макети вдалося реалізувати в єдиному стилі, забезпечивши логічну структуру та візуальну узгодженість між усіма екранами інтерфейсу.

2.3 Вибір кольорової палітри згідно з WCAG

Фінальний дизайн був створений з дотриманням принципів WCAG 2.1 (Web Content Accessibility Guidelines) для забезпечення доступності інтерфейсу для різних груп користувачів.

Враховано такі аспекти:

– Контрастність: текстові елементи та елементи керування мають достатній контраст із фоном у пастельних тонах, що відповідає вимогам стандарту WCAG 2.1. Як показано на рисунках 2.2 та 2.3, обрані кольори забезпечують доступність інтерфейсу для користувачів із порушеннями зору. Поєднання тексту кольору #F5F3EE з фоном #444444 має контрастність 8.78:1, що відповідає рівню WCAG AAA (рис. 2.1), а поєднання тексту кольору #D4C7B8 з тим самим фоном має контрастність 5.87:1, що відповідає рівню WCAG AA (рис. 2.2). Такі показники гарантують комфортну взаємодію з інтерфейсом незалежно від умов освітлення та типу пристрою.

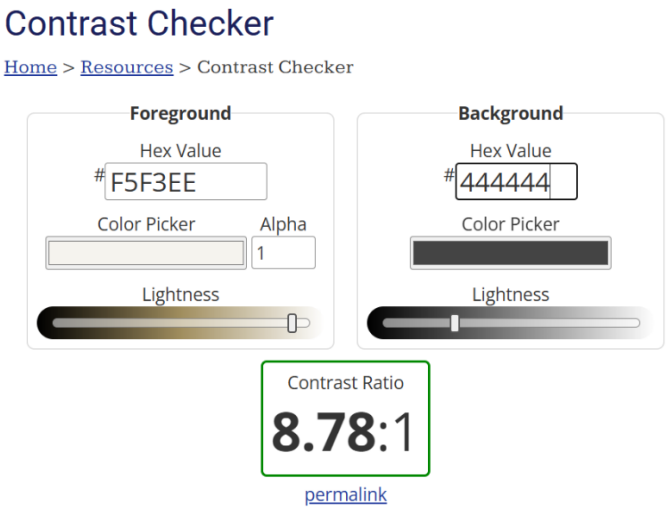

Рисунок 2.2 – Перевірка контрастності між текстом та фоном

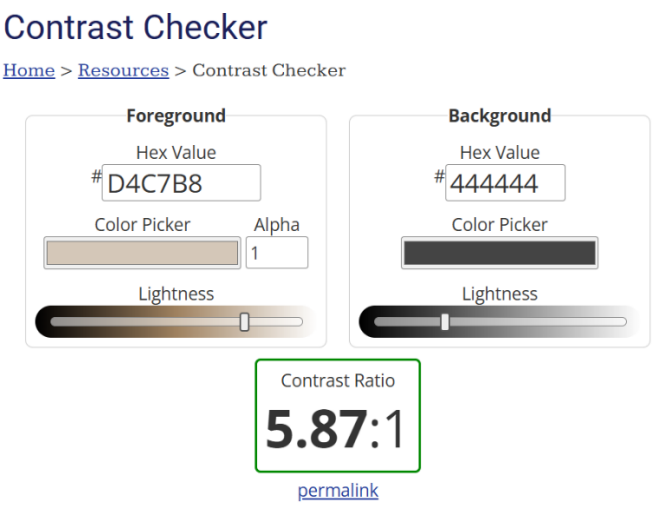

Рис. 2.3 – Перевірка контрастності між текстом і фоном

– Шрифти: використано прості, без засічок шрифти, які легко читаються на мобільних і десктопних пристроях.

– Колірна палітра: обрано м’які, ненасичені кольори – зелений, пісочний, світло–бежевий – які не викликають втоми очей і не є проблемними для людей з порушенням сприйняття кольору (рис. 2.4).

Рисунок 2.4 – Кольорова палітра застосунку Smart Home

Кольорова палітра на рисунку 2.2 була спеціально підібрана з урахуванням вимог доступності. Вона складається з м’яких пастельних відтінків, які мають

достатній контраст для тексту та інтерфейсних елементів і водночас є комфортними для тривалого використання.

- Розмір елементів: кнопки, повзунки та іконки мають достатній розмір для комфортної взаємодії на сенсорних екранах.
- Фокус–навігація: усі активні елементи мають візуальні стани фокусу для навігації з клавіатури.

3 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ СИСТЕМИ

3.1 Постановка задачі та цілі проєкту

У сучасних «розумних будинках» дедалі більш актуальним стає питання автоматизації систем освітлення з метою зниження енергоспоживання та підвищення комфорту користувачів. У рамках даного проєкту розроблено модель системи на базі контролера Arduino, що здатна в реальному часі визначати наявність людини в зоні спостереження та оцінювати рівень природного освітлення. За допомогою датчика руху PIR та фоторезистора LDR реалізовано комбіновану стратегію прийняття рішень: увімкнення світла або керування механічною заслінкою сервопривода тільки за необхідності, що суттєво знижує витрати електроенергії й продовжує ресурс світлових приладів [31, 32].

Основна ідея полягає в тому, щоб система самостійно вибирала оптимальний режим роботи: у разі виявлення руху за недостатнього рівня освітленості запускати сценарій «камертонного» підсвічування, а при відсутності руху або за високої інтенсивності природного світла – переключатися в енергозберігальний режим. Такий підхід не лише забезпечує оперативну реакцію (час спрацьовування системи не перевищує 300…400 мс), але й гарантує надійну роботу в широкому діапазоні температур та вологості [31].

Головною метою проєкту є створення прототипу автоматизованої системи інтелектуального керування освітленням, яка об’єднає апаратну частину (PIR–модуль, LDR–елемент, світлодіоди–індикатори, серво–привід) та алгоритм обробки сигналів на платформі Arduino.

3.2. Розробка принципової схеми у Tinkercad

Для реалізації апаратної частини системи управління «розумним будинком» було змодельовано принципову схему у середовищі Tinkercad. Схема представлена на рисунку 3.1 нижче та демонструє основні з’єднання компонентів з мікроконтролерною платою Arduino Uno Rev3.

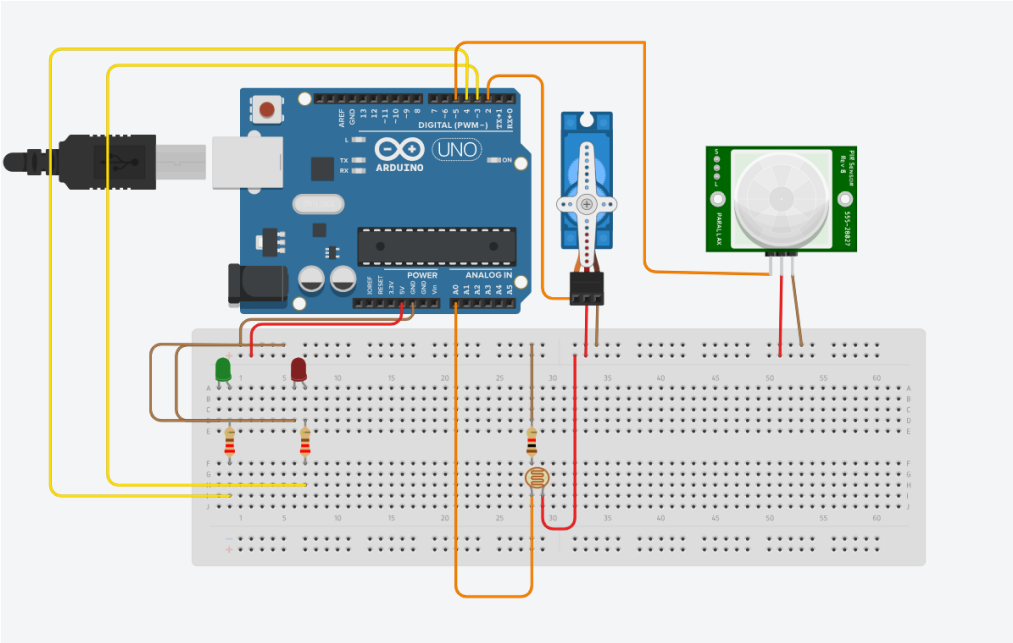

Рисунок 3.1 – Принципова схема у середовищі Tinkercad

У наведеній схемі використано наступні апаратні компоненти:

– мікроконтролер Arduino Uno Rev3;
– інфрачервоний датчик руху PIR HC–SR501;
– фоторезистор (LDR–сенсор);
– сервомотор SG02R;
– два світлодіоди (червоний та зелений);
– обмежувальні резистори;
– макетна плата (breadboard) та з’єднувальні дроти.

Принципова схема системи наведена на рисунку 3.1. Вона змодельована в середовищі Tinkercad і відображає реальні фізичні з'єднання компонентів з платою Arduino Uno Rev3 у складі автоматизованої системи для «розумного будинку».

З'єднання компонентів з піновими роз'ємами Arduino:

1. PIR–датчик HC–SR501:

– Вивід VCC підключено до виходу 5V плати Arduino.

– Вивід GND – до загального проводу (GND).

– Вивід OUT – до цифрового входу D7.

2. Фоторезистор (LDR):

– Один контакт LDR з'єднано з аналоговим входом A0.

– Інший контакт підключено до 5V через резистор приблизно 10 кОм, утворюючи дільник напруги.

– Інший кінець резистора – до GND, що забезпечує стабільне зчитування сигналу.

3. Сервомотор SG02R:

– Живлення (Vcc, помаранчевий провід) – до 5V.

– GND – до GND Arduino.

– Сигнальний пін (PWM) – до цифрового виходу D2, що підтримує керування ШІМ.

4. Світлодіоди з резисторами–обмежувачами:

– Зелений світлодіод: анод підключено до D9 через резистор, катод – до GND.

– Червоний світлодіод: анод підключено до D10 через резистор, катод – до GND.

– Таке підключення запобігає перевантаженню цифрових пінів і забезпечує безпечний рівень струму.

PIR-датчик відповідає за виявлення руху в зоні дії сенсора, а фоторезистор LDR визначає рівень зовнішнього освітлення. Сервомотор SG02R моделює фізичний процес підняття або опускання жалюзі, керується він вручну через

мобільний додаток за допомогою повзунка: переміщення повзунка в один бік відповідає за опускання жалюзі, в інший – за підняття. Світлодіоди виконують роль візуальних індикаторів стану системи: зелений сигналізує про нормальний режим, червоний – про виявлення руху.

У разі фіксації руху PIR-сенсором за недостатнього освітлення активується лише червоний світлодіод. Коли рух зникає або освітлення стає достатнім – система перемикається на зелений індикатор. При цьому сервомотор не залежить від показань сенсорів і приводиться в дію виключно через взаємодію користувача з додатком.

Таким чином, схема є енергоефективною та відповідає вимогам безпеки при використанні з мікроконтролером Arduino. Застосування макетної плати дало змогу вдало протестувати версію для керування пристроями розумного будинку.

3.2.1. Фізична реалізація моделі з використанням Arduino Uno

Модель, яка була попередньо змодельована у середовищі Tinkercad, була фізично реалізована на основі платформи Arduino UNO. Нижче представлено фото зібраної моделі на макетній платі (рис. 3.2).

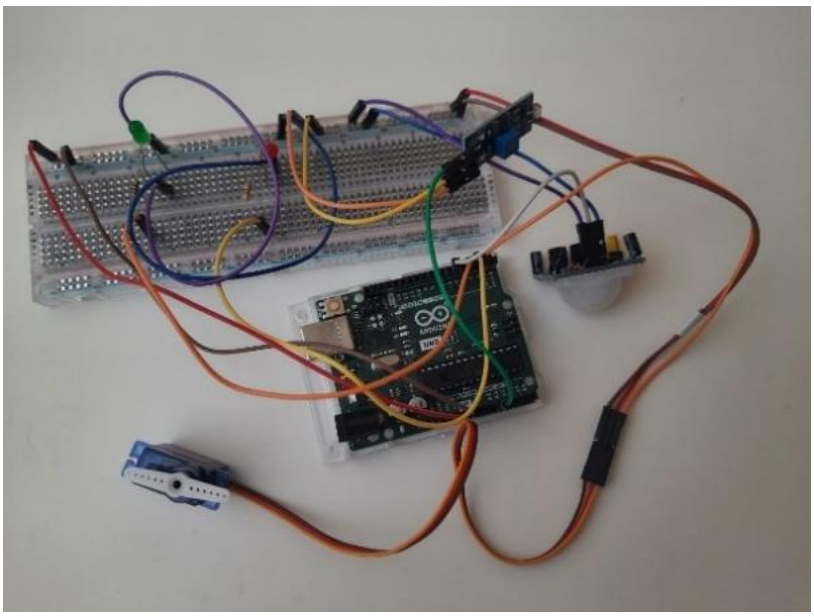

Рисунок 3.2 – Зібрана модель системи керування жалюзі на базі Arduino UNO

На макетній платі реалізовано підключення наступних компонентів:

– PIR-сенсор – виявляє рух у зоні дії;

– Фоторезистор (LDR) – визначає рівень освітлення;

– Світлодіоди (зелений та червоний) – слугують індикаторами режимів роботи;

– Сервомотор SG90 – імітує підняття та опускання жалюзі;

– Плата Arduino UNO – центральний елемент, що забезпечує обробку даних та взаємодію з додатком;

– Макетна плата (breadboard) – використана для з'єднання всіх елементів без пайки.

Для реалізації управління периферійними модулями в проєкті обрано плату Arduino Uno Rev3, до якої під'єднано датчик руху, фоторезистор, сервопривод та індикатори. Датчик руху HC–SR501 підключено до цифрового входу D2, оскільки саме цей пін підтримує зовнішні переривання (INT0). Такий підхід дозволяє отримувати миттєвий сигнал про зміну стану ("HIGH/LOW") без необхідності постійного опитування модуля, що значно зменшує затримки та навантаження на основний цикл програми [31]. Живлення датчика організовано від 5 V шини плати, а спільний провід підключено до GND згідно з рекомендаціями виробника.

Для вимірювання рівня природного освітлення фоторезистор уведено в класичну схему дільника напруги з виходом на аналоговий канал A0. Вбудований 10–бітний АЦП плати перетворює напругу в межах 0–5 V у цифрове значення від 0 до 1023, що забезпечує достатню точність оцінки освітленості для прийняття рішення про вмикання підсвітки [32, 34].

Сервопривод SG90 керується через ШІМ–вихід D9. Хоча Arduino за замовчуванням генерує на цьому піні сигнали з частотою близько 490 Гц, для коректної роботи серво налаштовано вихід на 50 Гц із тривалістю імпульсу 1–2 мс. Саме такий інтервал дозволяє точно позиціонувати заслінку, мінімізуючи

коливання та затримки [33]. Живлення серво також отримує від шини +5 V, а загальний провід з’єднано з GND макетної плати.

Індикатори стану – червоний і зелений світлодіоди – під’єднано до цифрових виходів D7 та D8 відповідно через послідовні резистори 220 Ω. Це рішення дозволяє швидко перемикати режими індикації «рух виявлено» і «режим очікування» без додаткових складних схем, зберігаючи при цьому мінімальне навантаження на виводи контролера [33].

4 РОЗРОБКА ВЕБДОДАТКУ ДЛЯ МОНІТОРИНГУ ПРИСТРОЇВ РОЗУМНОГО БУДИНКУ

4.1 Програмування Arduino UNO

У цьому розділі представлено програмний код, реалізований на мові програмування C++ з використанням платформи Arduino. Код призначений для керування мікроконтролером, що взаємодіє з датчиком руху, фоторезистором, сервоприводом та двома світлодіодами. Програма також забезпечує обмін даними з комп'ютером через серійний порт.

На початку програми підключається бібліотека Servo, яка забезпечує підтримку сервоприводу. Далі оголошуються константи для підключених елементів та змінні для обробки стану датчика руху (лістинг 4.1).

Лістинг 4.1 – Імпортування бібліотек та оголошення змінних

```
#include <Servo.h>

#define LED1_PIN 3
#define LED2_PIN 4
#define PIR_PIN 5
#define SERVO_PIN 2
#define LIGHT_SENSOR A0
Servo servo;
bool motionDetected = false;
unsigned long lastMotionTime = 0;
const unsigned long noMotionDelay = 5000;
```

У функції setup() виконується початкова конфігурація пінів: світлодіоди налаштовуються як виходи, датчик руху – як вхід. Сервопривод підключається до відповідного піну і встановлюється у нейтральну позицію (90 градусів).

Встановлюється з’єднання по серійному порту для подальшого обміну даними (лістинг 4.2).

Лістинг 4.2 – Ініціалізація в функції setup()

```
void setup() {
  pinMode(LED1_PIN, OUTPUT);
  pinMode(LED2_PIN, OUTPUT);
  pinMode(PIR_PIN, INPUT);
  servo.attach(SERVO_PIN);
  digitalWrite(LED1_PIN, LOW);
  digitalWrite(LED2_PIN, LOW);
  servo.write(90);
  Serial.begin(9600);
}
```

У функції loop() реалізовано зчитування сигналу з PIR–датчика. Якщо виявлено рух, активується світлодіод LED2, і в серійний порт надсилається повідомлення MOTION:1. Якщо протягом певного часу (5000 мс) нових рухів не зафіксовано, LED2 вимикається і надсилається повідомлення MOTION:0 (лістинг 4.3).

Лістинг 4.3 – Обробка сигналу з датчика руху

```
int motion = digitalRead(PIR_PIN);
unsigned long now = millis();
if (motion == HIGH) {
  motionDetected = true;
  lastMotionTime = now;
  digitalWrite(LED2_PIN, HIGH);
  Serial.println("MOTION:1");
}
```

```
if (motionDetected && (now - lastMotionTime >= noMotionDelay)) {
  digitalWrite(LED2_PIN, LOW);
  motionDetected = false;
  Serial.println("MOTION:0");
}
```

За допомогою фоторезистора, підключеного до аналогового входу, зчитується рівень освітлення в навколишньому середовищі. Значення передається у серійний порт у вигляді повідомлення LIGHT:<значення> (лістинг 4.4).

Лістинг 4.4 – Зчитування рівня освітлення

```
int light = analogRead(LIGHT_SENSOR);
Serial.print("LIGHT:");
Serial.println(light);
```

У цій частині програма обробляє текстові команди, що надходять із серійного порту. Залежно від отриманої команди, вмикається або вимикається один зі світлодіодів, або задається нова позиція сервопривода. Підтримуються команди типу LED1_ON, LED1_OFF, LED2_ON, LED2_OFF, а також SERVO:<кут> (лістинг 4.5).

Лістинг 4.5 – Обробка команд з комп’ютера

```
if (Serial.available()) {
  String command = Serial.readStringUntil('\n');
  command.trim();
  if (command == "LED1_ON") {
    digitalWrite(LED1_PIN, HIGH);
  } else if (command == "LED1_OFF") {
    digitalWrite(LED1_PIN, LOW);
  } else if (command == "LED2_ON") {
    digitalWrite(LED2_PIN, HIGH);
```

```
  } else if (command == "LED2_OFF") {
    digitalWrite(LED2_PIN, LOW);
  } else if (command.startsWith("SERVO:")) {
    int pos = command.substring(6).toInt();
    pos = constrain(pos, 0, 180);
    servo.write(pos);
  }
}
```

Наприкінці кожної ітерації головного циклу програма робить коротку паузу тривалістю 100 мс. Це запобігає надмірному навантаженню на контролер та уникненню зайвої частоти обробки (лістинг 4.6).

Лістинг 4.6 – Затримка між циклами

```
delay(100);
```

4.2 Опис коду HTML для вебдодатку «розумного будинку»

У цьому розділі представлено HTML-розмітку, яка визначає структуру та логіку відображення основних екранів і елементів веб-застосунку «Розумний будинок». Код включає послідовно організовані секції для авторизації, реєстрації, домашньої панелі, детального перегляду кімнати, керування освітленням та бокового меню.

Кожен екран ізольовано у власному <section>, що дозволяє легко керувати його показом через JavaScript. Всі інтерфейсні елементи (вхід, кнопки, поля, навігація) розміщено у логічних блоках для забезпечення зручної взаємодії.

Особливу увагу приділено повторно використовуваним компонентам (наприклад, групам пароля з іконкою, кнопкам переходу), що сприяє узгодженості дизайну.

Структура HTML створена з урахуванням адаптивності та подальшого розширення функціоналу, а також забезпечує підтримку сценаріїв керування розумними пристроями всередині додатку.

У цій частині коду налаштовується кодування сторінки (UTF-8), забезпечується адаптивність для мобільних пристроїв, задається заголовок сторінки "Smart Home" та підключається файл стилів style.css (лістинг 4.7).

Лістинг 4.7 – Підключення базових налаштувань (head)

```
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Smart Home</title>
  <link rel="stylesheet" href="css/style.css"/>
</head>
```

Цей блок відповідає за декоративний елемент (наприклад, фонову фігуру або кольорову пляму), який відображається в нижньому лівому куті інтерфейсу (лістинг 4.8).

Лістинг 4.8 – Декоративний елемент внизу зліва

```
<div class="decor bottom-left"></div>
```

Цей екран призначений для створення нового облікового запису користувача. У формі реєстрації передбачено поля для введення імені, адреси електронної пошти, а також пароля і його підтвердження – кожне з можливістю відображення введеного тексту за допомогою окремої кнопки. Форма завершується кнопкою "Sign up" для надсилання даних і містить посилання, яке дає змогу повернутися до екрана авторизації (лістинг 4.9).

Лістинг 4.9 – Екран авторизації

```
<section id="loginScreen" class="screen active">
  ...
</section>
```

Екран реєстрації дозволяє новому користувачеві створити обліковий запис у системі. У ньому передбачено введення імені, електронної пошти, а також пароля й підтвердження пароля – обидва поля мають кнопки для відображення введеного тексту. Після заповнення форми користувач може натиснути кнопку "Sign up" для завершення реєстрації. У нижній частині також розміщено посилання, яке дає змогу повернутися до сторінки входу (лістинг 4.10).

Лістинг 4.10 – Екран реєстрації користувача

```
<section id="signupScreen" class="screen">
  ...
</section>
```

Інтерфейс керування освітленням призначений для налаштування світла в обраній кімнаті. У верхній частині відображається назва кімнати, після чого розміщені кнопки для вибору одного з попередньо встановлених режимів освітлення, таких як нічний, релакс, робочий або яскравий. Нижче розташовані повзунки, що дозволяють регулювати рівень яскравості та колірну температуру освітлення. У кінці передбачені кнопки для встановлення таймера або створення розкладу увімкнення світла (лістинг 4.11).

Лістинг 4.11 – Керування освітленням

```
<section id="lightingScreen" class="screen">
  ...
</section>
```

4.3 Опис коду CSS для вебдодатку «розумного будинку»

У цьому розділі наведено CSS-стилі, що формують зовнішній вигляд інтерфейсу «розумного будинку». Код реалізує базове скидання стилів та оформлення фону сторінки, позиціонування декоративного елемента («плями»), керування показом різних «екранів» додатку (login, dashboard, налаштування), стилі універсальних кнопок та полів вводу, а також розмітку й оформлення горизонтальної каруселі карток кімнат із підписами і елементами управління освітленням. CSS забезпечує єдину кольорову гаму, відступи, скруглення та адаптивність інтерфейсу для зручної роботи на мобільних і десктопних пристроях.

Код (лістинг 4.12) забезпечує скидання стандартних відступів і полів для всіх елементів із встановленням моделі коробки «border-box», задає системний шрифт, світло-бежевий фон, фіксує висоту під розмір вікна та відключає прокрутку сторінки.

Лістинг 4.12 – Базове скидання стилів і оформлення фону сторінки

```
/* Reset & Base */
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
body {
  font-family: system-ui, sans-serif;
  background: #F5F2EB;
  color: #333;
  height: 100vh;
  overflow: hidden;
  position: relative;
}
```

Створює напівпрозорий круглий декоративний елемент зеленого кольору, який зміщений у нижній лівий кут поза областю контенту для візуального акценту (лістинг 4.13).

Лістинг 4.13 – Декоративний напівпрозорий зелений круг у нижньому лівому куті

```
/* Decorative Blob */
.decor {
  position: absolute;
  background: #82A98F;
  border-radius: 50%;
  width: 200px;
  height: 200px;
  opacity: 0.3;
}
.decor.bottom-left {
  bottom: -50px;
  left: -50px;
}
```

Фрагмент коду відповідає за відображення списку кімнат у вигляді горизонтального карусель-контейнера (лістинг 4.14):

- .rooms – створює гнучкий (flex) контейнер із горизонтальною прокруткою, до якого будуть додаватися картки кімнат.
- .room-card – задає стилі самої картки: фонове зображення кімнати (через background), фіксовані розміри та округлі краї, а також курсор-«рука» при наведенні.
- .room-label – розміщує напівпрозорий блок із назвою кімнати внизу картки, щоб текст був читабельним поверх зображення.

Лістинг 4.14 – Картки приміщень із фоновими зображеннями та підписами

```
/* Room Cards */
.rooms {
  display: flex;
  overflow-x: auto;
}
.room-card {
  min-width: 120px;
  height: 160px;
  margin-right: 0.75rem;
  border-radius: 12px;
  background-size: cover;
  background-position: center;
  cursor: pointer;
  position: relative;
}
.room-card .room-label {
  position: absolute;
  bottom: 0;
  width: 100%;
  background: rgba(0, 0, 0, 0.4);
  color: #fff;
  padding: 0.5rem 0;
  text-align: center;
}
```

4.4 Опис коду Python для вебдодатку «розумного будинку»

Нижче подано ключові фрагменти програми на Python із використанням Flask та pyserial, які реалізують REST–API для керування «розумним будинком», налаштування освітлення й взаємодію з апаратурою через Arduino.

У цьому блоці імпортуємо Flask для обробки HTTP-запитів і генерації JSON-відповідей, модуль pyserial (serial) для серійного зв’язку з Arduino, а також

бібліотеки threading і time, щоб у фоновому потоці періодично опитувати плату (лістинг 4.15).

Лістинг 4.15 – Імпорти та ініціалізація Flask

```
from flask import Flask, request, jsonify
import serial, threading, time
app = Flask(
    __name__,
    static_folder='static',
    static_url_path=''
)
```

Ініціалізація зв’язку з платою Arduino починається у функції init_arduino(): спочатку відкривається серійний порт /dev/ttyUSB0 зі швидкістю 9600 бод і встановлюється таймаут для читання, після чого одразу ж запускається окремий фоновий потік sensor_loop, який безперервно опитує плату, декодує отримані рядки та виводить їх у консоль.

Функція send_to_arduino(cmd) служить для формування і відправки готових команд у форматі ключ:значення–достатньо передати їй потрібний рядок, і він разом із символом нового рядка буде відправлений на Arduino через вже відкритий порт.

Таким чином, програма одночасно підтримує двосторонню комунікацію: приймає покази сенсорів і передає керуючі сигнали пристрою (лістинг 4.16).

Лістинг 4.16 – Модуль взаємодії з Arduino

```
arduino = None
def send_to_arduino(cmd):
    global arduino
    if arduino and arduino.is_open:
        arduino.write((cmd + "\n").encode())
```

```
    def sensor_loop():
        global arduino
        while True:
            try:
                line = arduino.readline().decode().strip()
                print("ARDUINO →", line)
            except:
                pass
            time.sleep(0.1)
    def init_arduino():
        global arduino
        try:
            arduino = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)
            threading.Thread(target=sensor_loop,
daemon=True).start()
            print("Arduino connected")
        except Exception as e:
            print("Could not open serial port:", e)
```

У цьому блоці програми всі дані зберігаються безпосередньо в оперативній пам’яті – у вигляді звичайних змінних Python, а не в базі даних. Словник users містить облікові записи для авторизації (email, пароль і ім’я), список rooms описує доступні кімнати з їхніми id та назвами, а room_states і lighting_settings імітують поточний стан пристроїв і параметри освітлення в кожній кімнаті. Такий підхід дозволяє швидко протестувати логіку REST–API без складного налаштування сховища даних (лістинг 4.17).

Лістинг 4.17 –Демонстраційні дані

```
    users = {
        "anna@example.com":  {"password":  "password123",  "name":
"Anna"}
    }
```

```
rooms = [
    {"id": 1, "name": "Living Room"},
    {"id": 2, "name": "Kitchen"},
    {"id": 3, "name": "Bedroom"}
]
room_states = {
    1: {"devices": {"wifi": True, "temp": 17, "light": True, "fan": True}, "users": ["Anna","Bob"]},
    2: {"devices": {"wifi": True, "temp": 20, "light": False, "fan": False}, "users": ["Anna"]},
    3: {"devices": {"wifi": False, "temp": 22, "light": True, "fan": False}, "users": ["Anna","Eve"]}
}
lighting_settings = {
    1: {"brightness": 50, "color_temp": 60},
    2: {"brightness": 30, "color_temp": 40},
    3: {"brightness": 80, "color_temp": 70}
}
```

Маршрут / обробляє GET–запит до кореневого URL і повертає клієнту файл index.html із папки static. Саме з цієї точки запускається фронтенд–інтерфейс, який відображається у браузері та забезпечує подальшу взаємодію користувача з додатком (лістинг 4.18).

Лістинг 4.18 – Головна сторінка (/)

```
@app.route('/')
def root():
    return app.send_static_file('index.html')
```

Маршрут /login приймає POST–запит із JSON–тілом, яке містить поля email та password. Сервер приводить email до нижнього регістру, перевіряє наявність такого запису в словнику users і порівнює переданий пароль із тим, що зберігається в пам’яті. Якщо аутентифікація пройшла успішно, клієнт отримує відповідь {"success": true, "name": "<Ваше ім’я>"} зі статусом 200; у протилежному випадку

повертається {"success": false, "message": "Invalid email or password"} з кодом 401 (лістинг 4.19).

Лістинг 4.19 – Логін користувача (/login)

```
@app.route('/login', methods=['POST'])
def login():
    data = request.json or {}
    email = data.get("email", "").lower()
    u = users.get(email)
    if u and u["password"] == data.get("password"):
        return jsonify(success=True, name=u["name"])
    return jsonify(success=False, message="Invalid email or password"), 401
```

Маршрут /signup приймає POST–запит з JSON–тілом, яке містить поля name, email та password. Спочатку сервер перевіряє, що всі три поля непусті, а потім упевнюється, що вказаний email ще не зареєстрований у словнику users. Якщо будь–яка з перевірок не проходить (наприклад, поле порожнє або email уже існує), клієнту повертається відповідь зі success: false і кодом HTTP 400. Якщо ж валідація пройшла успішно, у словник users додається новий запис, і сервер повертає {"success": true, "name": "<Ваше ім’я>"} з кодом HTTP 200 (лістинг 4.20).

Лістинг 4.20 – Реєстрація нового користувача (/signup)

```
@app.route('/signup', methods=['POST'])
def signup():
    data = request.json or {}
    name = data.get("name", "").strip()
    email = data.get("email", "").lower().strip()
    password = data.get("password", "")
    if not name or not email or not password:
        return jsonify(success=False, message="Name, email and password are required"), 400
```

```
        if email in users:
            return jsonify(success=False, message="Email already
registered"), 400
        users[email] = {"name": name, "password": password}
        return jsonify(success=True, name=name)
```

Маршрут /rooms обробляє GET–запит і повертає клієнту масив об’єктів, кожен із яких містить поля id та name наявних у системі кімнат. Це дозволяє фронтенду відобразити користувачу повний перелік доступних локацій (лістинг 4.21).

Лістинг 4.21 – Список кімнат (/rooms)

```
@app.route('/rooms', methods=['GET'])
def get_rooms():
    return jsonify(rooms)
```

При GET–запиті до маршруту /room/<int:room_id> сервер отримує числовий параметр room_id, шукає у списку rooms відповідний об’єкт і, якщо не знаходить, повертає пустий словник із статусом 404 Not Found. Якщо ж запис знайдений, він вибирає з room_states поля devices та users для цієї кімнати й повертає їх разом із id та назвою кімнати у вигляді JSON (лістинг 4.22).

Лістинг 4.22 – Деталі кімнати (/room/<int:room_id>)

```
@app.route('/room/<int:room_id>', methods=['GET'])
def get_room(room_id):
    room = next((r for r in rooms if r["id"] == room_id), None)
    if not room:
        return jsonify({}), 404
    st = room_states.get(room_id, {"devices": {}, "users": []})
    return jsonify(
```

```
            id=room_id,
            name=room["name"],
            devices=st["devices"],
            users=st["users"]
        )
```

У цьому маршруті /lighting/<int:room_id> обробка запиту залежить від його методу. Якщо приходить GET, сервер шукає в пам’яті налаштування освітлення для вказаної кімнати (brightness та color_temp) і повертає їх у вигляді JSON. У разі POST–прийому JSON із новими значеннями яскравості чи колірної температури, програма оновлює відповідні поля словника lighting_settings та викликає send_to_arduino, щоб надіслати команди у форматі light–brightness:<значення> або light–colortemp:<значення> на плату через серійний порт. Після обробки POST–запиту клієнт отримує відповідь {"success": true} (лістинг 4.23).

Лістинг 4.23 – Керування освітленням (/lighting/<int:room_id>)

```
@app.route('/lighting/<int:room_id>', methods=['GET','POST'])
def lighting(room_id):
    if request.method == "GET":
        return jsonify(lighting_settings.get(room_id, {}))
    data = request.json or {}
    ls = lighting_settings.setdefault(room_id, {})
    if "brightness" in data:
        ls["brightness"] = int(data["brightness"])
        send_to_arduino(f"light-brightness:{ls['brightness']}")
    if "color_temp" in data:
        ls["color_temp"] = int(data["color_temp"])
        send_to_arduino(f"light-colortemp:{ls['color_temp']}")
    return jsonify(success=True)
```

При безпосередньому запуску скрипта виконується:

1. init_arduino() – встановлення серійного зв’язку з Arduino й старт фонового потоку.

2. app.run(...) – підняття Flask–сервера на всіх інтерфейсах (0.0.0.0), порт 5000, з режимом налагодження (лістинг 4.24).

Лістинг 4.24 – Точка входу та запуск (__main__)

```
if __name__ == '__main__':
    init_arduino()
    app.run(host='0.0.0.0', port=5000, debug=True)
```

4.5 Опис коду JavaScript для вебдодатку «розумного будинку»

Функціональність вебінтерфейсу неможливо уявити без JavaScript. У цьому розділі наведено основні фрагменти коду, що відповідають за роботу інтерфейсу: навігацію між екранами, зміну станів елементів та обробку введених даних.

Основу інтерфейсу становить механізм перемикання між екранами: усі вони фізично присутні в DOM-структурі, однак видимим залишається лише той, що має клас active. Метод show('homeScreen') приховує поточний екран і показує новий без перезавантаження сторінки (лістинг 4.25).

Лістинг 4.25 – Навігація між екранами

```
function show(id) {
  document.querySelectorAll('.screen').forEach(s => s.classList.remove('active'));
  const tgt = document.getElementById(id);
  if (tgt) tgt.classList.add('active');
}
```

Керування меню. Кнопка «гамбургер» відкриває меню, а хрестик – повертає на головний екран. Функція show дозволяє легко керувати видимістю екранів (лістинг 4.26).

Лістинг 4.26 – Відкриття і закриття бокового меню

```
document.getElementById('menuBtn')
  .addEventListener('click', () => show('menuScreen'));

document.getElementById('closeMenu')
  .addEventListener('click', () => show('homeScreen'));
```

За допомогою виклику document.querySelectorAll(...) здійснюється вибірка всіх елементів меню, що знаходяться всередині екрана з ідентифікатором menuScreen і мають клас menu-item. Для кожного з вибраних елементів призначається обробник події click, який реагує на натискання користувача. Після кліку зчитується текст елемента за допомогою item.textContent.trim(), і далі, залежно від його значення, виконуються різні дії: якщо текст дорівнює 'Home', викликається функція show('homeScreen'), яка повертає користувача на головний екран; якщо 'Account', на екрані з'являється повідомлення "Сторінка 'Account' з’явиться пізніше!", що свідчить про тимчасову відсутність цієї сторінки; якщо ж текст відповідає 'Logout', користувача переадресовують на екран входу, викликаючи функцію show('loginScreen'). Таким чином, реалізується логіка обробки навігаційного меню в інтерфейсі (лістинг 4.27).

Лістинг 4.27 – Обробка пунктів меню: Home, Account, Logout

```
document.querySelectorAll('#menuScreen .menu-item').forEach(item
=> {
  item.addEventListener('click', () => {
    const txt = item.textContent.trim();
    if (txt === 'Home') {
      show('homeScreen');
    } else if (txt === 'Account') {
```

```
      alert('Сторінка "Account" з’явиться пізніше!');
    } else if (txt === 'Logout') {
      show('loginScreen');
    }
  });
});
```

Цей фрагмент коду відповідає за завантаження головного екрана (дашборду) та відображення списку кімнат, доступних у системі «розумного будинку».

На початку виконується створення поточної дати через об'єкт Date. Далі за допомогою методу toLocaleDateString дата форматується у вигляді «день місяць» відповідно до української локалі (наприклад, 28 червня) та вставляється в елемент із ідентифікатором currentDate.

Після цього за допомогою fetch('/rooms') виконується асинхронний запит до сервера для отримання списку кімнат. Якщо запит проходить успішно, відповідь у форматі JSON зберігається в масиві rooms. У разі помилки в консолі виводиться попередження, але виконання коду продовжується.

Далі очищується вміст елемента roomsList, у якому будуть відображені всі кімнати. Для кожної кімнати створюється картка: назва кімнати перетворюється на «слаг» (рядок нижнього регістру з підкресленням замість пробілів), використовується як частина URL для зображення фону, і додається текстова мітка з назвою кімнати. Крім того, кожна картка отримує обробник події click, який викликає функцію openRoom(r.id) для переходу до деталей відповідної кімнати.

Наприкінці викликається функція show('homeScreen'), яка відображає головний екран із усіма згенерованими елементами. Таким чином, функція loadHome() відповідає за ініціалізацію дашборду та динамічне відображення кімнат, доступних користувачеві (лістинг 4.28).

Лістинг 4.28 – Завантаження головного екрану

```
async function loadHome() {
```

```
    const now  = new Date();
    const opts = { month: 'long', day: 'numeric' };
    document.getElementById('currentDate')
      .textContent = now.toLocaleDateString('uk-UA', opts);

    let rooms = [];
    try {
      rooms = await (await fetch('/rooms')).json();
    } catch (err) {
      console.warn('Не вдалося завантажити кімнати:', err);
    }

    const list = document.getElementById('roomsList');
    list.innerHTML = '';
    rooms.forEach(r => {
      const slug = r.name.toLowerCase().replace(/\s+/g, '_');
      const card = document.createElement('div');
      card.className = 'room-card';
      card.style.backgroundImage = `url('img/room_${slug}.jpg')`;
      const label = document.createElement('span');
      label.className = 'room-label';
      label.textContent = r.name;
      card.appendChild(label);
      card.addEventListener('click', () => openRoom(r.id));
      list.appendChild(card);
    });

    show('homeScreen');
  }
```

Цей фрагмент коду відповідає за керування освітленням у «розумному будинку». За допомогою повзунків користувач може змінювати яскравість та кольорову температуру освітлення, а всі зміни надсилаються на сервер. Крім того, реалізовано кнопки з готовими режимами (ніч, відпочинок, робота тощо), які

автоматично встановлюють відповідні значення. Всі зміни одразу відображаються в інтерфейсі (лістинг 4.29).

Лістинг 4.29 – Керування освітленням: повзунки та режими

```
['brightnessSlider', 'colorTempSlider'].forEach(id => {
  document.getElementById(id)
    .addEventListener('input', async e => {
      const payload = {};
      payload[id === 'brightnessSlider' ? 'brightness' :
'color_temp'] =
        +e.target.value;
      try {
        await fetch(`/lighting/${window.currentRoom}`, {
          method: 'POST',
          headers: {'Content-Type':'application/json'},
          body: JSON.stringify(payload)
        });
      } catch (_) {}

updateLightingColor(+document.getElementById('brightnessSlider').val
ue);
    });
});

document.querySelectorAll('.preset-btn').forEach(b => {
  b.addEventListener('click', () => {
    const modes = {
      night:  { brightness: 20, color_temp: 30 },
      relax:  { brightness: 50, color_temp: 40 },
      work:   { brightness: 80, color_temp: 70 },
      bright: { brightness: 100, color_temp: 90 }
    };
    const p = modes[b.dataset.mode];
```

```
          document.getElementById('brightnessSlider').value          =
p.brightness;
          document.getElementById('colorTempSlider').value           =
p.color_temp;
          updateLightingColor(p.brightness);
        });
      });
```

У цьому розділі було реалізовано практичне поєднання апаратних компонентів та програмної логіки для створення базової системи розумного будинку. Було розроблено та протестовано кілька сценаріїв автоматизації із залученням сенсорів, виконавчих пристроїв і елементів індикації. Наведені фрагменти коду демонструють ефективне керування периферією за допомогою платформи Arduino. Отримані результати підтверджують працездатність системи та її готовність до подальшого розширення функціональності.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено прототип системи «розумного будинку» з інтеграцією апаратних засобів (Arduino Uno, сенсори, сервомотори) та вебінтерфейсу для дистанційного моніторингу й управління. Основною метою дослідження було створення функціонального вебдодатку, що дозволяє користувачеві зручно взаємодіяти з елементами системи в реальному часі.

На початковому етапі було проведено аналітичний огляд існуючих IoT-технологій, зокрема протоколів зв'язку (MQTT, HTTP/HTTPS, CoAP, DDS) і сенсорних пристроїв, придатних для автоматизації побутових процесів. Було обґрунтовано вибір мікроконтролера Arduino Uno Rev3, як оптимальної платформи для побудови апаратної частини.

Під час реалізації було створено інтерфейс користувача відповідно до сучасних вимог доступності (WCAG 2.1). Використано інструмент Figma для проєктування дизайну та технології HTML/CSS, JavaScript, Python (Flask) для створення вебдодатку з підтримкою REST API. Було також здійснено підключення й налаштування ключових сенсорів – PIR, LDR – а також виконавчих механізмів – серводвигуна й індикаторних світлодіодів.

Система успішно протестована на макетній платі, що підтвердило її працездатність, адаптивність до змін зовнішніх умов та ефективність у споживанні електроенергії. Програмне забезпечення забезпечує як автоматичну реакцію на зміни середовища, так і ручне керування пристроями з боку користувача.

Отримані результати можуть бути використані як основа для подальшої розробки повнофункціональної системи розумного дому, придатної до впровадження у реальних умовах.

ПЕРЕЛІК ПОСИЛАНЬ

1. Schulz, J. M., & Scilla, J. S. (2024). Broad Perspective of Smart Home Technology in 2024. International Journal of Smart Technologies, 1(1), 1–27.

2. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2021). IoT and Cloud Computing: Issues, Challenges and Opportunities – A Review. Retrieved from https://www.researchgate.net/publication/350065834_IoT_and_Cloud_Computing_Issues_Challenges_and_Opportunities_A_Review

3. Sicari S., Rizzardi A., Grieco L. A., Coen-Porisini A. Security, privacy and trust in Internet of Things: The road ahead // Computer Networks. – 2020. – Vol. 76. – P. 146–164. – Режим доступу: https://ieeexplore.ieee.org/document/8858493

4. MarketsandMarkets. (2024). Smart Home Market Size, Share & Trends by Product, Offering, Sales Channel, Installation Type and Region – Global Forecast to 2029.

5. Що таке IoT–технологія та як вона впливає на різні галузі [Електронний ресурс] // Hub.kyivstar.ua. – Режим доступу: https://hub.kyivstar.ua/articles/shho–take–iot–tehnologiya–ta–yak–vona–vplyvaye–na–rizni–galuzi

6. Словник термінів. Інтернет речей (IoT) [Електронний ресурс]. – Режим доступу: https://termin.in.ua/internet–rechey–iot

7. Rouse, Margaret (2019). «Internet of things (IoT)». IOT Agenda

8. Ok, E. (2025, February). A Detailed Review of the Progress in Home Automation Systems. [Рецензія, лютий 2025]. Emmanuel Ok.

9. Григоренко О. Г., Дідковська Н. А. Протоколи рівня застосунків в мережах IoT // Матеріали 2024 Міжнародної науково–практичної конференції студентів та молодих учених «Сучасні інформаційні системи та технології». – Київ: НН ІТС КПІ ім. Ігоря Сікорського, 2024. – С. 1–3. – Режим доступу: https://conferenc.its.kpi.ua/2024/paper/viewFile/29999/17460

10. Малохвій, Є., & Молчанов, Г. (2022). Дослідження протоколів передачі даних в умовах Інтернету речей [Електронний ресурс].– Режим доступу: https://journals.nupp.edu.ua/sunz/article/view/2446

11. Alashoor T., Baskerville R. Security and information sharing in the digital home: A new frontier // Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS 2019). – Association for Computing Machinery, 2019. – С. 6905–6914. – Режим доступу: https://doi.org/10.1145/3292006.3300025

12. Iera A., Atzori L., Morabito G. The Internet of Things: A survey // Computer Networks. – Режим доступу: https://ieeexplore.ieee.org/document/6076698

13. Kumar A., Sharma R. A review on the role of internet of things (IoT) in smart homes // Journal of The Institution of Engineers (India): Series B. – 2025. – Vol. 106. – P. 761–772. – Режим доступу: https://link.springer.com/article/10.1007/s41872-025-00324-7

14. What is DDS? [Електронний ресурс] // DDS Foundation. – Режим доступу: https://www.dds–foundation.org/what–is–dds–3/

15. Шевченко Л. С., Уманець В. О., Розпутня Б. М. Використання платформи Arduino у підготовці вчителів інформатики за принципами STEM навчання // Open educational e–environment of modern university. – 2023. – №15. – С. 130–134.

16. Вдовиченко О. С., Трофименко Т. В. Розробка та моделювання пристроїв на базі мікроконтролера Arduino в Tinkercad // Науковий вісник МДУ. – 2023. – №2(56). – С. 45–51.

17. Arduino Uno Rev3. Офіційна документація [Електронний ресурс]. – Режим доступу: https://docs.arduino.cc/hardware/uno–rev3

18. Arduino Uno Board Overview [Електронний ресурс]. – Режим доступу: https://www.arduino.cc/en/Main/ArduinoBoardUno

19. Al-Fuqaha A., Guizani M., Mohammadi M., Aledhari M., Ayyash M. Internet of Things: A survey on enabling technologies, protocols, and applications // IEEE Communications Surveys & Tutorials. – 2023. – Vol. 17, no. 4. – P. 2347–2376. – Режим доступу: https://ieeexplore.ieee.org/document/7848162

20. Banzi M., Shiloh M. Getting Started with Arduino. 3rd ed. – Maker Media, 2015. – 124 p.

21. Czujnik ruchu PIR HC–SR501 zielony JustPi [Електронний ресурс] // Botland. – Режим доступу: https://botland.com.pl/czujniki–ruchu/1655–czujnik–ruchu–pir–hc–sr501–zielony–justpi–5903351241359.html

22. Mini-Tech. Датчик руху інфрачервоний (PIR Sensor) HC-SR501 [Електронний ресурс]. – Режим доступу: https://www.mini-tech.com.ua/ua/datchik-dvizheniya-infrakrasniy-pir-sensor-hc-sr501

23. PIR–датчик руху HC–SR501 Arduino [Електронний ресурс] // Wiki TDTU. – Режим доступу: https://wiki.tntu.edu.ua/PIR_%D0%B4%D0%B0%D1%82%D1%87%D0%B8%D0%BA_%D1%80%D1%83%D1%85%D1%83_Arduino_HC–SR501

24. Czujnik światła LDR rezystancyjny dla Arduino Okystar [Електронний ресурс] // Botland. – Режим доступу: https://botland.com.pl/czujniki–swiatla–i–koloru/16560–czujnik–swiatla–ldr–rezystancyjny–dla–arduino–okystar–5904422378202.html

25. Light Dependent Resistor (LDR) [Електронний ресурс] // Electronics–Notes. – Режим доступу: https://www.electronics–notes.com/articles/electronic_components/resistors/light–dependent–resistor–ldr.php

26. SG90 Micro Servo – FriendlyWire Datasheet [Електронний ресурс]. – Режим доступу: https://www.friendlywire.com/projects/ne555–servo–safe/SG90–datasheet.pdf

27. SG90 Micro Servo – Waveshare Specs [Електронний ресурс]. – Режим доступу: https://www.waveshare.com/sg90–servo.htm

28. SG90 Micro Servo Datasheet – Kjell [Електронний ресурс]. – Режим доступу: https://www.kjell.com/globalassets/mediaassets/701916_87897_datasheet_en.pdf

29. Calculating red LED resistor Arduino [Електронний ресурс] // Electronics Stack Exchange. – Режим доступу:

https://electronics.stackexchange.com/questions/266328/calculating–red–led–resistor–arduino

30. Patel A. LED Resistor Values for Current Limiting Resistor [Електронний ресурс] // GS Network. – 2022. – Режим доступу: https://www.gsnetwork.com/led–resistor–values–for–current–limiting–resistor/

31. IJISRT. Automatic Security Light and Alarm Using Arduino and PIR Sensor [Електронний ресурс]. – Режим доступу: https://ijisrt.com/automatic–security–light–and–alarm–using–arduino–and–pir–sensor

32. RegentElectronics. Interfacing LDR Sensor with Arduino – Step–by–Step Guide with Code [Електронний ресурс]. – Режим доступу: https://regentelectronics.com/microcontrollers/interfacing–ldr–sensor–with–arduino–step–by–step–guide–with–code

33. OmArTronics. Arduino Servo Motor Control – Guide for SG90 & PCA9685 [Електронний ресурс]. – Режим доступу: https://omartronics.com/arduino–servo–motor–control–guide–for–sg90–pca9685/

34. Kazi S., Liyakat S., Kazi K., Kosgiker G. PIR Sensor–Based Arduino Home Security System [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/376722165_PIR_Sensor–Based_Arduino_Home_Security_System