Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютерна система "Портативна карта України"*

Виконав: студент IV курсу, групи CI-42
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

		Сисак О.М.
	(підпис)	(прізвище та ініціали)
Керівник		*Осухівська Г.М.*
	(підпис)	(прізвище та ініціали)
Нормоконтроль		*Тиш Є.В.*
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		*Осухівська Г.М.*
	(підпис)	(прізвище та ініціали)
Рецензент		*Фриз М.Є.*
	(підпис)	(прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Осухівська Г.М.*
(підпис) (прізвище та ініціали)
«27» січня 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Сисаку Олегу Михайловичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютерна система "Портативна карта України"*

Керівник роботи *Осухівська Галина Михайлівна, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від *« 27 » січня 2025 року № 4/7-53*
2. Термін подання студентом завершеної роботи *20 червня 2025 року*
3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз технічного завдання
2. Проєктна частина
3. Практична частина
4. Безпека життєдіяльності, основи охорони праці. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Структурна схема системи
2. Схема електрична принципова
3. Блок-схема алгоритму роботи системи
4. Креслення кришки
5. Загальний вигляд комп'ютерної системи "Портативна карта України"

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	*д.т.н., професор Пилипець М. І.*		

7. Дата видачі завдання *27.01.2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Розробка технічного завдання*	*27.01 – 02.02*	*Викон.*
2.	*Аналіз технічного завдання*	*15.02 – 27.03*	*Викон.*
3.	*Проєктна частина*	*06.04 – 02.05*	*Викон.*
4.	*Практична частина*	*04.05 – 25.05*	*Викон.*
5.	*Безпека життєдіяльності, основи охорони праці*	*26.05 – 02.06*	*Викон.*
6.	*Оформлення пояснювальної записки і графічного матеріалу*	*03.06 – 10.06*	*Викон.*
7.	*Перевірка на академічний плагіат, перевірка керівником та консультантами*	*9.06 – 15.06*	*Викон.*
8.	*Попередній захист кваліфікаційної роботи бакалавра*	*16.06 – 22.06*	*Викон.*
9.	*Захист кваліфікаційної роботи бакалавра*	*27.06*	*Викон.*

Студент _____ *Сисак Олег Михайлович*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Осухівська Галина Михайлівна*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сисак О.М. Комп’ютерна система “Портативна карта України”: робота на здобуття кваліфікаційного ступеня бакалавра: спец. 123 – комп’ютерна інженерія. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2025.

Ключові слова: комп’ютерна система, карта України, мікроконтролер ESP-32, дисплей OLED I²C 128×64 1.3″, погодні умови, повітряні тривоги.

Кваліфікаційна робота присвячена розробці портативної карти України, мобільного електронного приладу, призначеного для візуалізації географічного простору нашої країни із різноманітним набором режимів відображення даних. Пристрій покликаний відображати територіальну карту в режимі реального часу, а також відображати актуальну інформацію про погодні умови, стан повітря, швидкість вітру, ультрафіолетове випромінювання та поточні загрози, а саме повітряні тривоги. Пояснювальна записка складається з чотирьох розділів.

У першому розділі виконано аналіз вимог до проектованої портативної карти України. Обґрунтовано актуальність теми та сфери застосування даного пристрою.

У другому розділі розроблено архітектуру портативної карти України. Пояснено вибір апаратної складової та каналів для обміну даними.

У третьому розділі описано реалізацію програмних функцій та апаратної будови портативної карти України. Також проведено тестування пристрою.

У четвертому розділі розглянуто питання безпеки життєдіяльності та охорони праці при роботі з портативною картою України.

ANOTATION

Sysak O. M. Computer System «Portable Map of Ukraine»: Bachelor's Graduation Thesis: speciality 123 – computer engineering. Ternopil: Ternopil Ivan Puluj National Technical University, 2025.

Keywords: computer system, portable map of Ukraine, ESP32 microcontroller, I²C OLED display 128×64 1.3″, weather conditions, air raid alerts.

The qualification thesis is devoted to the development of a Portable Map of Ukraine, a mobile electronic device designed to visualize the geographical space of our country with a variety of display modes. The device is intended to present the territorial map in real time and to show up-to-date information on weather conditions, air quality, wind speed, ultraviolet radiation, and current threats-specifically, air raid alarms. The explanatory note consists of four chapters.

Chapter I presents an analysis of the requirements for the designed Portable Map of Ukraine. The relevance of the topic and the areas of application for this device are substantiated.

Chapter II develops the architecture of the Portable Map of Ukraine. The choice of hardware components and communication channels is explained.

Chapter III describes the implementation of the software functions and the hardware design of the Portable Map of Ukraine. Device testing is also conducted and documented.

Chapter IV examines issues of occupational health and safety when working with the Portable Map of Ukraine.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

API – Application Programming Interface

AQI – Air Quality Index

BMS – Battery Management System

DIY – Do It Yourself

GPIO – General Purpose Input Output

GUI – Graphical User Interface

HTTP – Hypertext Transfer Protocol

I^2C – Inter-Integrated Circuit

IP – Internet Protocol

JSON – JavaScript Object Notation

SCL – Serial Clock

SDA – Serial Data

TCP – Transmission Control Protocol

UART – Universal Asynchronous Receiver-Transmitter

UV – Ultraviolet

ВСТУП

У сучасному інформаційному суспільстві швидкий доступ до актуальних даних стає критично важливим для прийняття обґрунтованих рішень та забезпечення безпеки громадян. В умовах зростання військових та техногенних загроз виникає потреба у мобільному пристрої, який дозволяє в режимі реального часу отримувати та візуалізувати інформацію про можливі надзвичайні події, екологічну ситуацію, а також стан погодних умов по всій території України.

У цій кваліфікаційній роботі запропоновано розробку портативної карти України – компактного електронного приладу з OLED-дисплеєм, адресною світлодіодною стрічкою та сенсорною клавішею, який відображає не лише контури регіонів нашої держави, а й поточні дані щодо погодних явищ, рівня забруднення повітря, швидкості вітру, ультрафіолетового індексу та оперативні сповіщення про повітряні тривоги. Завдяки підтримці роботи від літій-іонного акумулятора, а також вбудованому Wi-Fi на базі ESP32, цей пристрій забезпечує автономний режим роботи та миттєву комунікацію з відповідними API-сервісами [1].

Мета роботи – створити ефективний та інтуїтивно зрозумілий апаратно-програмний комплекс, що демонструє карту України з відображенням таких режимів:

– “Повітряна тривога” – індикація статусу загроз по областях (зелений/червоний), звуковий сигнал у разі початку або завершення тривоги;

– “Погода” – показ опису атмосферних умов (ясно, хмарно, дощ, сніг, туман) із відповідним кольором світлодіодів;

– “Температура” – відобрження градусів Цельсія та кольорова шкала для відмінності регіональних значень;

– “Якість повітря (AQI)” – індикатор екологічного стану (градація від сірого до оливкового);

– “Швидкість вітру” – візуалізація на шкалі від білого до темно-синього;

– “УФ-індекс” – градація від блакитного до фіолетового залежно від рівня ультрафіолету.

Для досягнення поставленої мети передбачено виконати такі завдання:

– аналіз існуючих рішень та API-сервісів, які забезпечують дані про погоду, якість повітря та загрози;

– розробка схемотехнічного рішення комп'ютерної системи “Портативна карта України”;

– створення алгоритму роботи та програмної реалізації системи;

– розробка прототипу комп'ютерної системи “Портативна карта України”

– налагодження та тестування системи.

Реалізація портативної карти України сприятиме підвищенню оперативності отримання життєво важливої інформації під час надзвичайних ситуацій та стане прикладом компактного пристрою, здатного працювати в автономному режимі. Використання такого рішення може бути корисним у мобільних пунктах моніторингу екологічного та безпекового стану, громадських місцях, освітніх закладах, у службовому автотранспорті рятувальних служб, тощо, а також серед звичайних громадян для швидкого отримання необхідної інформації в домашніх умовах.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Основні технічні вимоги до портативної карти України

Портативна карта України призначена для забезпечення швидкого та якісного доступу до актуальних даних про стан областей України у режимі реального часу, тому її технічні вимоги охоплюють низку обов’язкових характеристик. По-перше, пристрій має функціонувати у автономному режимі протягом кількох годин, отримуючи живлення від вбудованого акумулятора із можливістю заряджання і захистом від перевантаження. У вимкненому стані енергоспоживання повинно бути мінімальним, а перемикання живлення – реалізоване так, щоб запобігти витоку струму. Пристрій повинен автоматично коригувати вихідну напругу, зберігаючи стабільну роботу електронних компонентів незалежно від рівня заряду батареї [2].

По-друге, необхідно, щоб система мала вбудований механізм первинного налаштування бездротового зв’язку. При відсутності заданих параметрів Wi-Fi вона мусить запускатися у спеціальному конфігураційному режимі: створювати власну точку доступу, до якої потрібно буде під’єднатися для введення користувацького імені та пароля мережі Wi-Fi через вбудований веб-інтерфейс. Після цього у клієнтському режимі пристрій підключається до локальної мережі, синхронізує внутрішній годинник за протоколом NTP і лише потім продовжує роботу. Тривалість очікування підключення не повинна перевищувати заданий інтервал, після чого пристрій повторно переходить у конфігураційний режим, якщо підключення не було встановлено.

По-третє, інтерфейс відображення інформації повинен одночасно показувати три рядки тексту. Перший рядок відображає поточний час і назву

активного режиму, які збережені у внутрішній пам’яті та названі згідно імен оригінальних сервісів. Другий рядок призначений для відображення конкретного значення, вибраного для попередньо обраної області (наприклад, температура чи опис погодного явища), а третій – для циклічного виведення групових даних усіх областей. Оновлення екрану має відбуватися не рідше, ніж кожні 50 мілісекунд, щоб забезпечити плавність прокрутки та швидкий відгук на взаємодію. Шрифт і розташування тексту має бути читабельним у широкому діапазоні освітленості та кутів огляду.

По-четверте, світлодіодна індикація може використовуватися для одночасного кольорового позначення стану кожної області. Колірні шкали мають бути підібрані так, щоб проміжки умовних значень можна було чітко розрізняти при низькій яскравості та на відстані, зокрема:

– для температури передбачена градація від холодних відтінків (фіолетовий, синій, бірюзовий) до теплих (жовтий, помаранчевий, червоний);

– для швидкості вітру – від білого (штиль) до насиченого темно-синього (дуже сильний порив); для індексу якості повітря – від зеленого (хороше) через жовтий і помаранчевий до червоного та пурпурового (дуже погане);

– для рівня ультрафіолету – від світлих небесних відтінків до фіолетового (екстремальний рівень);

– для погодних явищ – умовні кольори «сонячно», «хмарно», «дощ», «сніг», «туман» тощо;

– для повітряних тривог – чергування зеленого (стан без загрози) та червоного (активний попереджувальний сигнал), а також обрана область повинна бути супроводжена звуковим сповіщенням про початок і завершення тривоги.

По-п’яте, час та інтервали оновлення даних повинні бути чітко регламентовані: повідомлення про повітряні тривоги оновлюються кожні 30 секунд, оскільки ситуація може змінюватись дуже швидко та ця інформація

може стати життєво важливою, а дані погодних умов, якості повітря, швидкості вітру та ультрафіолету – раз на 30 хвилин, аби уникнути перевантаження обчислювальних ресурсів і надмірного мережевого трафіку. Система має коректно обробляти помилки зв'язку. У разі відсутності відповіді від сервера інформація повинна залишатися попередньою, а повторна спроба зачекати визначений інтервал, не має призводити до зависання інтерфейсу. При цьому користувацький інтерфейс не повинен блокуватися під час очікування мережевих запитів – він має залишатися готовим до обробки команд [3-5].

По-шосте, логіка взаємодії потребує єдиного сенсорного елемента, який розпізнає короткий дотик для циклічної зміни вибору серед областей чи режимів і тривалий дотик для підтвердження остаточного вибору. Затримка для фільтрації випадкових спрацювань (не менше 300 мілісекунд між подіями) гарантує стабільність введення. При цьому взаємодія користувача з пристроєм повинна залишатися інтуїтивно зрозумілою, тобто: коротке натискання змінює індекс, довге – підтверджує, а в будь-який момент, перебуваючи в будь-якому режимі відображення, довгий дотик повертає назад до меню вибору режиму.

Програмне забезпечення портативної карти України має бути оптимізоване для роботи в обмежених умовах вбудованого контролера. Пам'ять необхідно використовувати економно, щоб вмістити механізми обробки JSON, реалізацію циклів опитування API, оновлення екранів і керування світлодіодами без затримок. Логіку слід організувати у вигляді станового автомата, де кожна фаза роботи – від початкового вибору яскравості, встановлення мережевого з'єднання, вибору області й режиму до постійного оновлення даних в активному режимі – має чітко визначені умови переходу. Вимоги до архітектури передбачають розділення функцій на обробку інтерфейсу, мережевий зв'язок, парсинг і візуалізацію, щоб помилки в одному підмодулі не блокували роботу інших [6-7].

Комп'ютерна система “Портативна карта України” повинна мати компактні розміри, низьку вагу для забезпечення його портативності, а ергономічний корпус із надійними кріпленнями гарантуватиме можливість експлуатації в будь-яких умовах. Дотримання цих технічних вимог забезпечить ефективну роботу портативної карти України у різних умовах, з можливістю оперативного моніторингу та візуалізації важливої інформації по всіх регіонах країни.

1.2 Аналіз існуючих рішень

1.2.1 Аналіз існуючих підходів до мобільної візуалізації просторової інформації

Наявні підходи до мобільної візуалізації просторової інформації зазвичай поєднують інтеграцію географічних даних зі з’єднанням до мережевих сервісів і динамічним відображенням різноманітних показників. У зв’язку з цим у можна виділити дві ключові групи підходів: класичні системи офлайн-відображення карт із попередньо завантаженою інформацією та гібридні рішення, що динамічно звертаються до API для одержання актуальних даних [8].

Аналіз програмних бібліотек і фреймворків для роботи з геоданими [9] демонструє, що найпростіші реалізації обмежуються виведенням растрових чи векторних плиток карти (наприклад, з використанням OpenStreetMap або Mapbox), однак у вимозі портативності, автономності та мінімальних обчислювальних ресурсів такі підходи не завжди прийнятні. З огляду на це вбудовані пристрої (мікроконтролери або міні-комп’ютери) застосовують стислий набір інструкцій для відображення спрощеної картографічної сітки у вигляді абревіатур регіонів або зафарбованих підсвічених ділянок з мінімальною роздільною здатністю. Прикладом таких легких рішень є кілька розробок на основі мікроконтролерів, які використовують OLED або e-ink

дисплеї для локального виведення спрощеної векторної графіки, при цьому попередньо згенеровані потрібна інформація зберігаються у флеш-пам’яті. Однак ці рішення зазвичай не підтримують динамічне оновлення інформації про поточні події чи параметри довкілля, а лише відображають статичні межі й стандартні підписані найменування.

Зі свого боку, одноплатні комп'ютери, такі як Raspberry Pi або BeagleBone, дозволяють запускати легкі веб-сервери та працювати з повноцінними бібліотеками Leaflet чи OpenLayers, отримуючи відображення карти із загального сховища в Інтернеті. Цей підхід забезпечує високий рівень візуалізації та інтеграції з хмарними базами даних, але водночас вимагає значних ресурсів (Linux-середовище, дискові накопичувачі, активне охолодження) і відповідного енергоспоживання, що ускладнює переносне автономне застосування.

Окрему групу становлять мобільні додатки для смартфонів, які, незважаючи на зручність, не дають змоги отримати повністю автономну систему без смартфона. Їхній функціонал зазвичай зосереджений на відображенні геомаркерів, а не на індикації конкретних регіонів із використанням підсилених кольорових схем для швидкого зчитування (наприклад, попередження про надзвичайні ситуації чи забруднення). Крім того, у таких додатках потрібен інстальований клієнт і постійна наявність екрану високого розділення, що не завжди доцільно, особливо, для візуалізації інформації для групи осіб.

Виділяють також спеціалізовані апаратно-програмні комплекси для відображення оперативних повідомлень на карті в режимі реального часу, які застосовуються, наприклад, у службах цивільного захисту та медичних диспетчерських. Вони, як правило, використовують дисплеї діагоналлю від 7 до 12 дюймів і працюють на базі вбудованих систем із підтримкою Ethernet або Wi-Fi [10]. Подібні рішення забезпечують детальну графіку та багатоканальне одночасне відображення декількох видів інформації, але їхній

розмір, роздільність екрана й споживана потужність роблять такі пристрої надто громіздкими для перенесення і потребують тривалого часу ініціалізації.

Важливою ознакою сучасних розробок є інтеграція кількох API: погодного, екологічного та попереджувального. Серед відкритих сервісів найчастіше використовують OpenWeather, який надає уніфікований кінцевий пункт для поточних метеоданих і якості повітря, а також регіональні сервери сейсмічних чи надзвичайних повідомлень [11]. Проте більшість існуючих прикладів націлені на пристрої з класичного сімейства Linux (single-board computers), де набагато простіше оперувати HTTP-запитами і JSON-парсингом. Водночас для малопотужних вбудованих контролерів більшістю розробників доводиться обмежуватися лише одним API, оскільки обсяг вмістимої пам’яті і пропускна здатність мережі обмежені.

Таким чином, аналіз свідчить про те, що попередні спроби створити компактну автономну карту-індикатор із динамічним підключенням до декількох сервісів або ж обмежувалися тільки одним видом даних, або використовували більш потужні обчислювальні платформи, які втрачають властивості портативності. Жодне типове готове рішення не поєднує простоти інтерфейсу, мінімального формфактора та одночасного відображення кількох режимів на одному дисплеї невеликого розміру з кольоровою LED-стрічкою.

1.2.2 Методи отримання та обробки динамічних даних

Розробникам пристроїв обмеженого ресурсу рекомендують використовувати REST-API із мінімальною кількістю полів у відповіді, що передбачає стислий обмін за протоколом HTTP. Багато сервісів пропонують «легкі» кінцеві точки, які повертають лише критично важливі показники (наприклад, температура, швидкість вітру, рівень AQI, UV-індекс) у формі лаконічного JSON, що легко розбирається навіть із невеликих буферів пам’яті [12]. Для повітряних тривог зазвичай застосовують простий пошук по UID регіонів у текстовому рядку відповіді – замість побудови глибоких дерев

JsonDocument. Таким чином, методика отримання динамічних даних полягає в періодичних запитах із чітким інтервалом і використанні побайтового порівняння підрядків замість повного JSON-парсера у великій структурі.

1.2.3 Існуючі апаратно-програмні реалізації

У світі вбудованих рішень можна знайти як промислові продукти, так і універсальні DIY-проєкти, проте більшість із них або працюють на одній з чотирьох груп платформ (кластерні контролери, планшети, одноплатники і мікроконтролери середнього класу) [13]. Пристрої на мікроконтролерах можуть обмежитися OLED-екраном і простим LED-індикатором, але тоді не забезпечується багатоканальність відображення. “Одноплатники” (наприклад, Raspberry Pi) здатні запускати GUI-фреймворки для карт, проте це значно підвищує енергоспоживання і габарити.

Зокрема, для мікроконтролера з обмеженим об’ємом флеш-пам’яті існують розробки, в яких карти областей України зберігаються у вигляді двомірних масивів або стислих bitmap, тобто кожна область відповідає певному фрагменту растрового зображення. Динамічні показники у таких реалізаціях зазвичай відображаються однотонним забарвленням регіону або за допомогою миготливих індикаторів. Найчастіше як основу беруть саме OpenWeather, оскільки він дає великий набір погодних і екологічних показників за єдиним ключем доступу [11].

В іншій категорії DIY-проектів розробники використовують LED-матриці низького розширення (8×8 або 16×16) для демонстрації контурів областей, проте через малу кількість пікселів не вдається передати реальні межі. LED-матриця часто служить єдиним індикатором – якщо в регіоні активована тривога, відповідний світлодіод змінює колір, але немає можливості відобразити одночасно й іншу інформацію (температуру чи AQI) [14].

Однією з найбільш функціональних реалізацій є мапа «JAAM» – апаратно-програмний комплекс на базі мікроконтролера ESP32, призначений для візуалізації стану повітряної тривоги, погоди та інших подій на мапі України за допомогою адресних світлодіодів (рис. 1.1) [15].

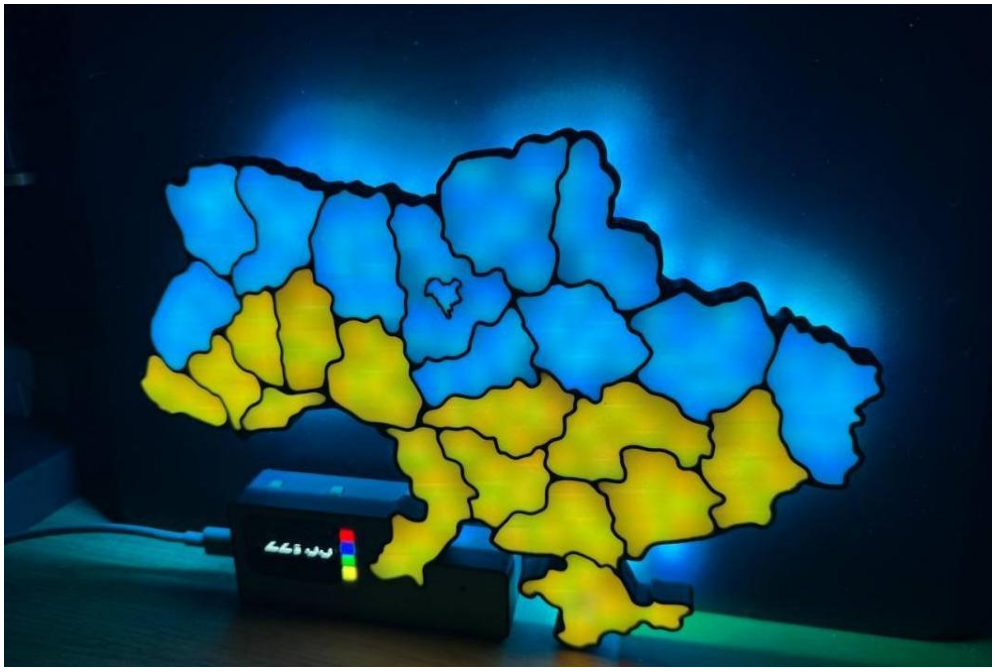

Рисунок 1.1 – Електронна мультимедійна DIY мапа України "JAAM" [16]

У конструкції передбачено 124 RGB-світлодіоди типу WS2812, які відповідають окремим регіонам України, OLED-дисплей для виводу поточної інформації, а також динамік, сенсори освітлення та клімату. Мапа має веб-інтерфейс для налаштувань, підтримку WebSocket з'єднання з власним сервером, а також можливість оновлення прошивки з браузера. Прошивка підтримує кілька режимів роботи: індикація повітряних тривог, погодні умови, візуальні ефекти та багато іншого. Вивід інформації на дисплей включає поточний час, температуру, вологість, технічний стан пристрою тощо [15].

JAAM також сумісний із системами розумного дому, зокрема, Home Assistant, що дозволяє інтегрувати карту у домашню інфраструктуру. Усі налаштування зберігаються у вбудованій пам'яті та автоматично

відновлюються після перезавантаження. Одним з недоліків цієї система є велика ціна, оскільки для виготовлення потрібно багато компонентів та ресурсів. Хоча проєкт більше не виробляється серійно, він є прикладом ефективного поєднання індикаторної системи з IoT-функціональністю та оперативним оновленням інформації в реальному часі.

У сучасних рішеннях широко застосовуються портативні карти, що виводять лише індикацію тривоги в регіонах без можливості використання інших режимів моніторингу. Такі пристрої характеризуються обмеженою функціональністю, зосереджуючись виключно на відображенні стану тривог, що не дозволяє користувачу отримати іншу інформацію [17]. Відсутність інших режимів відображення призводить до зниження корисності цих рішень.

Таким чином, на відміну від існуючих пристроїв, що обмежені відображенням лише сигналів тривоги, запропонована портативна карта України забезпечить мультифункціональний моніторинг із можливістю перемикання між шістьма інформаційними режимами. Окрім режиму повітряних тривог, користувач отримуватиме оперативні дані про погодні умови та її короткий опис, актуальні температурні показники, індекс якості повітря, швидкість вітру та рівень ультрафіолетового випромінювання. Кожен із цих режимів автономно оновлюватиметься через стандартизовані API з налаштуванням інтервалів запитів, що оптимізуватиме енергоспоживання та забезпечуватиме актуальність інформації. Такий комбінований підхід значно розширює аналітичні можливості портативного приладу, перетворюючи його на універсальний інструмент оперативного моніторингу екологічних, метеорологічних та безпекових параметрів.

1.3 Аналіз можливих рішень

Розробка комп'ютерної системи “Портативна карта України” вимагала поєднання компактної апаратної платформи з ефективною візуальною та

кольоровою індикацією. Для її реалізації пропонується інтегрувати декілька ключових апаратно-програмних складових, що разом забезпечать автономну роботу, достатню обчислювальну потужність, а також можливість динамічного відображення інформації. Основним обчислювальним ядром обрано мікроконтролер ESP32, адже ця платформа має вбудовані модулі Wi-Fi та Bluetooth, достатній для цієї системи обсяг оперативної пам’яті та енергонезалежної флеш-пам’яті, що дозволяє розгортати складні прошивки з підтримкою WebSocket-протоколу та локальних веб-серверів. Пристрій розміром 31×18 мм має мінімальні габарити та енергоспоживання, що є критично важливим для переносного рішення. Інтерфейс користувача реалізований на монохромному OLED 128×64 пікселів у трирядковому форматі. Кольорову індикацію забезпечує адресна WS2812-стрічка з 26 діодами, в якій кожен піксель відповідає за окремий регіон. Також запропоновано використання п’єзоелектричного пасивного зумера для відтворення найпростіших звукових сповіщень.

Взаємодія з користувачем здійснюватиметься через один сенсор дотику з можливістю розпізнавати короткий та довгий дотики. Апаратура живитиметься від одного 18650 літій-іонного акумулятора з зарядним модулем TP4056 і підвищувачем напруги MT3608, що гарантує стабільні 5 В для всіх компонентів незалежно від стану акумулятора. Автономність роботи в активному режимі сягатиме 5–6 годин.

Завдяки такій конфігурації система отримує необхідну гнучкість: компактний корпус розміром близько 150 × 200 × 50 мм та вагою до 500 г, мінімальні апаратні витрати та інтуїтивно зрозумілий інтерфейс.

РОЗДІЛ 2 ПРОЄКТНА ЧАСТИНА

2.1 Розробка архітектури портативної карти України

Відповідно до функціональних та експлуатаційних вимог “Портативна карта України” повинна забезпечувати одночасне відображення шести режимів, мати інтуїтивно зрозумілий інтерфейс, можливість циклічного перегляду даних для всіх 25 адміністративних одиниць та працювати автономно до 5-6 годин. Такі умови визначили загальну архітектуру пристрою, що складається з центрального мікроконтролера ESP-32 DevKit V1, OLED-дисплея 128×64, адресної LED-стрічки WS2812, єдиного сенсорного входу для навігації, пасивного зумера для звукових сповіщень та системи живлення на базі одного акумулятора 18650 із зарядним модулем TP4056 і підвищувальним перетворювачем MT3608 (рис. 2.1).

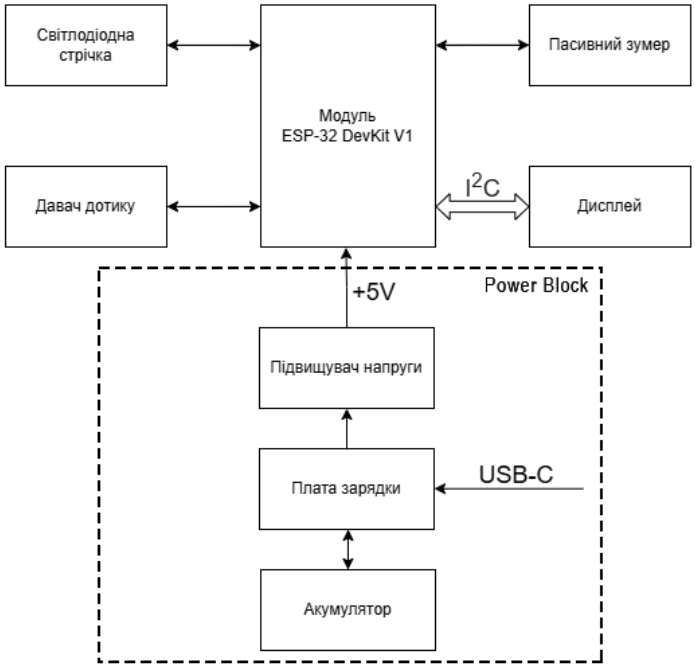

Рисунок 2.1 – Структурна схема портативної карти України

У цій схемі ESP32 виконує роль головної компоненти усієї системи, а саме, він ініціює запити до зовнішніх API, аналізує відповіді простим пошуком ключових підрядків у JSON-рядку, зберігає найновіші значення у внутрішніх буферах та керує виведенням інформації на дисплей та LED-стрічку. OLED-екран через I²C використовується для відображення часу, назви активного режиму та двох рядків тексту (обрані дані по регіону і бігучий рядок із груповими показниками всіх областей). Одночасно з цим світлодіодна стрічка WS2812 забезпечує індикацію кольорами та відтінками для кожного режиму й регіону, зумер повідомляє користувача про ввімкнення система, а також підсилює сповіщення про початок і завершення повітряної тривоги.

Взаємодія з користувачем відбувається через один сенсорний модуль TTP223, під'єднаний до окремого цифрового виводу ESP32. Короткий дотик послідовно переключає індекс області або режиму, довгий - підтверджує вибір і переводить систему у відповідний стан скінченного автомата. Така архітектура дозволяє обійтися без апаратних кнопок, зберігаючи мінімальний форм-фактор.

Система живлення реалізована за допомогою одного Li-Ion акумулятора формату 18650, заряджання якого здійснюється через порт USB-C і контролер TP4056 із вбудованим захистом від перенапруги та короткого замикання. Підвищувальний перетворювач MT3608 забезпечує стабільні +5 В для LED-стрічки, а вбудований стабілізатор на ESP32 формує +3.3 В для контролера, дисплея та інших компонентів. Система управління живленням (BMS) контролює рівень заряду й не допускає глибокого розряду, що гарантує заявлені 5–6 годин автономної роботи без зниження продуктивності.

У результаті запропонована структурна схема поєднує в собі гнучкість програмної архітектури, мінімальний набір апаратних компонентів та енергоефективність, що задовольняє всі технічні умови для портативного приладу з відображення потрібних даних.

2.2 Обґрунтування вибору апаратного забезпечення портативної карти України

2.2.1 Опис основних компонентів

При створенні комп'ютерної системи “Портативна карта України” пріоритетним завданням було знайти баланс між обчислювальною потужністю, енергоефективністю та мінімальними габаритами, щоб забезпечити надійну роботу в різноманітних умовах. “Серцем” системи обрано модуль ESP-32 DevKit V1 (рис. 2.2), оснащений двоядерним 32-бітним процесором XTensa LX6 із тактовою частотою до 240 МГц, вбудованими інтерфейсами Wi-Fi 802.11 b/g/n і Bluetooth LE, 30 GPIO, 12-розрядним АЦП . Завдяки двом ядрам одна лінія коду вирішує комунікацію з API, а інша – керує інтерфейсом користувача й світлодіодною індикацією. Мінімальне споживання у сплячому режимі у поєднанні з можливістю регулювання живлення до 3.0–3.6 В гарантує близько п’яти годин автономної роботи від одного акумулятора 18650. Зовнішній вигляд модуля ESP-32 DevKit V1 показано на рисунку 2.2 [18].

Рисунок 2.2 – Зовнішній вигляд модуля ESP-32 DevKit V1 [19]

Інформація про поточний час, назву режиму та дані про обрану область виводяться на компактний дисплей OLED I²C 128×64 1.3″ (рис. 2.3), підключений по I²C-шині. Така роздільність дозволяє чітко відображати трирядковий текст без зовнішнього підсвічування, при цьому споживання в активному режимі не перевищує 150 мА, а в режимі очікування – кілька сотень мікроампер. Бігучий рядок групових даних організовано програмно, без потреби в апаратних додаткових буферах [20].

Рисунок 2.3 – Зовнішній вигляд OLED дисплея I²C 128×64 1.3″ [20]

Кольорова індикація усіх режимів та регіонів реалізована за допомогою адресної LED-стрічки WS2812 із 26 діодами (рис. 2.4) [21]. Послідовний спосіб з’єднання дає змогу індивідуально налаштувати кожен світлодіод на будь-який колір RGB, а також керувати рівнем яскравості та оновленням до 400 Гц, що забезпечує плавну анімацію під час стартової індикації та миттєве відображення змін режиму.

Рисунок 2.4 – Зовнішній вигляд адресної LED-стрічки WS2812 [22]

Сенсорна взаємодія через єдиний модуль TTP223 (рис. 2.5) замінила класичні кнопки. Він використовується в проєкті як основний засіб керування. Цей мініатюрний модуль реагує на наближення пальця до своєї чутливої зони, генеруючи цифровий сигнал (LOW або HIGH), який зчитується ESP32. У даній системі він програмно розпізнає короткі та довгі дотики, дозволяючи користувачеві інтуїтивно перемикати режими без потреби в кнопках. TTP223 відзначається низьким енергоспоживанням, стабільною роботою в умовах портативного живлення та простотою інтеграції, що робить його ідеальним рішенням для компактних пристроїв [23].

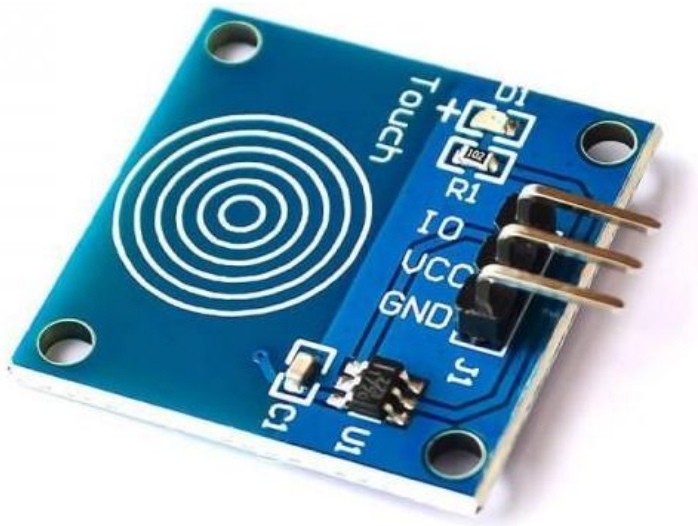

Рисунок 2.5 – Зовнішній вигляд давача дотику TTP223 [24]

Пасивний зумер у проєкті відіграє роль звукового індикатора, сигналізуючи про події, як-от початок повітряної тривоги та успішне увімкнення системи. Він керується за допомогою ШІМ сигналів через вбудований таймер ESP32. На відміну від активних зумерів, пасивний потребує зовнішнього генератора частоти, що дає змогу відтворювати різні тони. Зумер обраний за простоту, універсальність та здатність до гнучкого керування звуком без додаткових аудіо модулів.

Для автономності застосовано один акумулятор 18650 із зарядним контролером TP4056 і захистом BMS, а стабільні +5 В для LED-стрічки та +3.3 В для ESP32-модуля та дисплея формуються підвищувальним DC–DC MT3608 і вбудованими LDO-регуляторами модуля. Зарядка через USB-C одночасно живить пристрій і поповнює акумулятор, без потреби зовнішніх перемикань живлення.

2.2.2 Опис вузлів портативної карти України

Електрична принципова схема портативної карти України складається з кількох логічних вузлів, кожен із яких відповідає за окрему функціональну підсистему пристрою: джерела живлення та заряд, основний контролер і супутні периферійні інтерфейси, дисплейна та світлодіодна індикація, давач дотику, а також модулі зберігання й передавання даних.

Підсистема живлення портативної карти України побудована за модульним принципом і складається з трьох ключових елементів: літій-іонного акумулятора формату 18650, зарядного контролера TP4056 і підвищувального конвертера MT3608.

Акумулятор з’єднано через відсік із механічним фіксатором до входу TP4056, що контролює струм заряду до 1 А і напругу відключення при досягненні 4,2 В. Вузол зарядки має додатковий MOSFET-перемикач і захисний діод для автоматичного перемикання живлення між USB-C портом і батареєю (рис. 2.6).

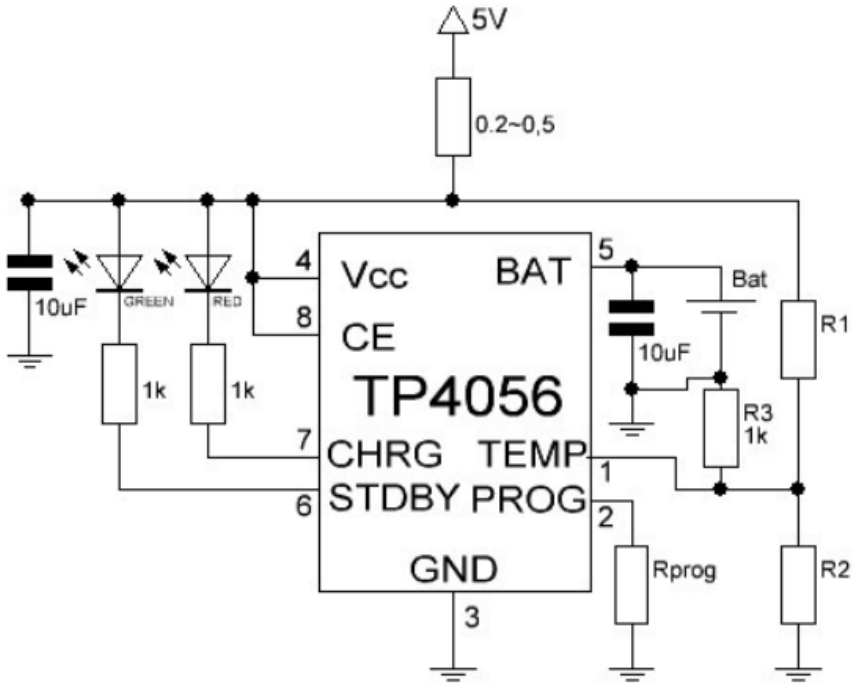

Рисунок 2.6 – Схема електрична принципова модуля зарядки TP4056

Після модуля TP4056 шина акумулятора подається на підвищуючий перетворювач DC-DC MT3608, який піднімає напругу до 5 В для підсистеми WS2812, і одночасно через лінійний стабілізатор на базі мікросхеми MIC5219 формується напруга 3,3 В для живлення ESP-32 DevKit V1 та OLED-дисплея (рис. 2.7). Між конвертером і навантаженнями розміщені фільтрувальні конденсатори різних ємностей, що забезпечують низький рівень пульсацій і стабільну роботу цифрових ланцюгів.

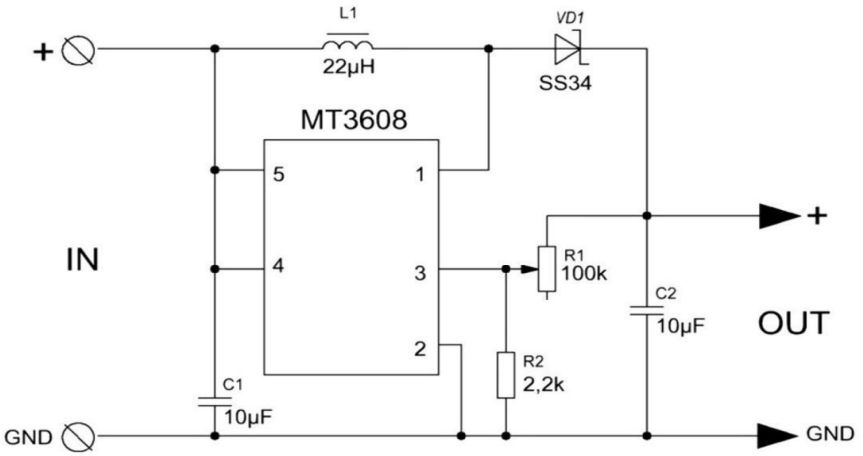

Рисунок 2.7 – Схема електрична принципова підвищувача DC-DC MT3608

Обчислювальний вузол реалізований навколо модуля ESP-32 DevKit V1, який відповідає за збір даних, обробку API-запитів та керування інтерфейсами (рис. 2.8).

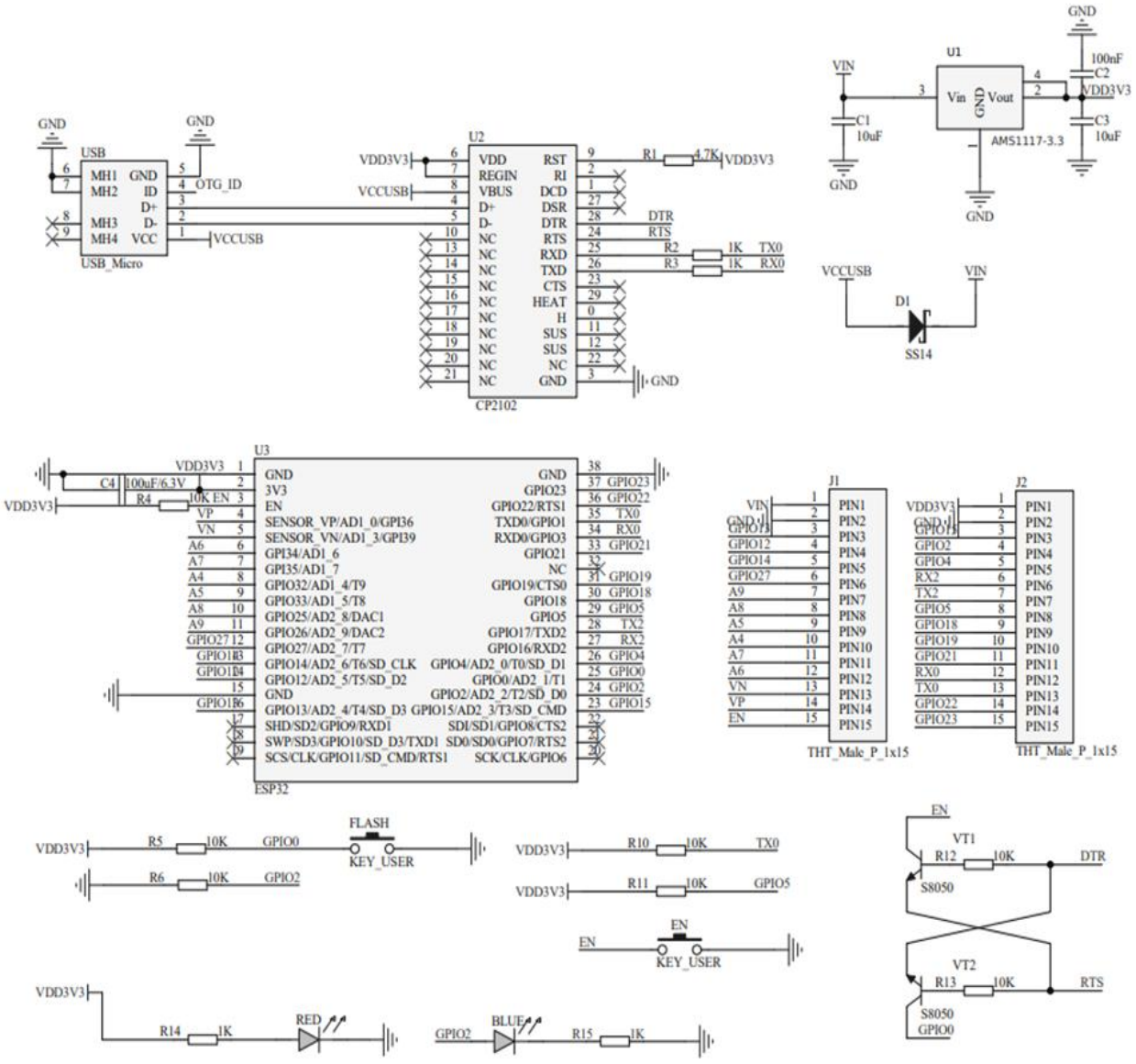

Рисунок 2.8 – Вузли схеми електричної принципової модуля ESP-32 DevKit V1

Всі ймовірні джерела шуму від розряду акумулятора та підвищувального конвертера гасаться багатоступеневим RC-фільтром, розташованим лінійно перед живленням мікроконтролера. Інтерфейс I²C використовується для керування OLED-дисплеєм 128 × 64 пікселів. Тактові резистори і буферні

конденсатори на шині даних захищають лінії від перехресних перешкод, а підтягуючі резистори 10 кОм на входах GPIO запобігають плаваючому стану під час перезавантаження.

Індикаційний блок поєднує в собі OLED дисплей I²C 128×64 1.3″ і WS2812 LED-стрічку: OLED показує поточний час, обраний режим і мінімальні текстові повідомлення (рис. 2.9).

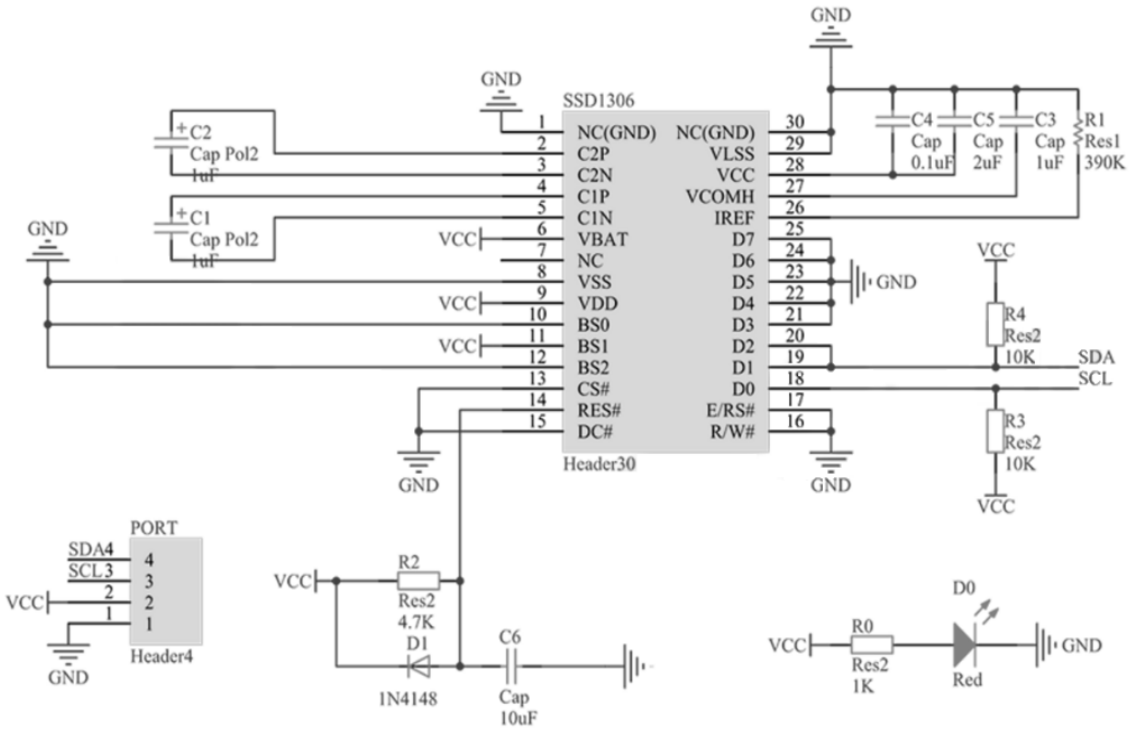

Рисунок 2.9 – Схема електрична принципова підключення OLED дисплея I²C 128×64 1.3″

Під час роботи дисплея світлодіодна стрічка WS2812 змінює колір і анімацію відповідно до режиму (рис. 2.10).

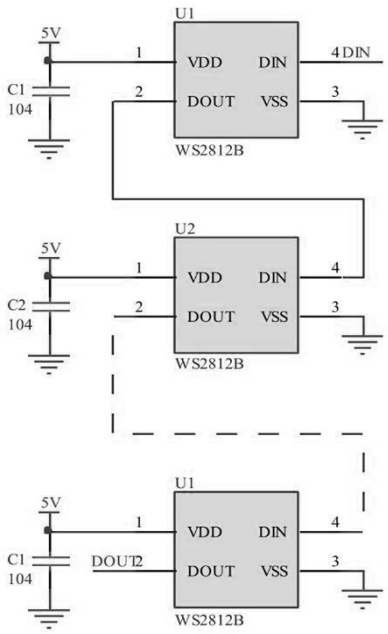

Рисунок 2.10 – Схема електрична принципова підключення світлодіодної стрічки WS2812

Сенсор TTP223 фіксує коротке та довге натискання для навігації меню (рис. 2.11). Пасивний зумер відтворює сигнали початку й завершення тривоги, а також повідомляє про увімкнення системи. Його схема виконана через GPIO-керований транзистор із обмежувальним резистором, що гарантує чіткі імпульси без додаткових мікросхем.

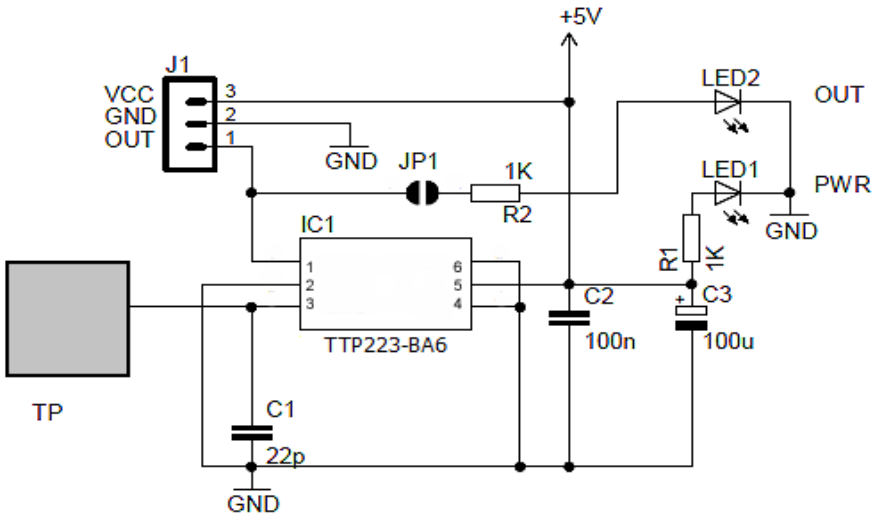

Рисунок 2.11 – Схема електрична принципова вузла сенсора дотику TTP223

Відсік для акумулятора 18650 і перемикач дають змогу повністю відключити систему від джерела живлення, забезпечуючи раціональне управління енергією та безпечне вимкнення пристрою.

Узгодження електричної структури та логіки з’єднань є критично важливим для гарантії коректної та безперебійної роботи портативної карти України. На рисунку 2.12 наведено принципову електричну схему, де наведено з’єднання усіх складових цієї системи: модуля зарядки TP4056, підвищувального конвертера MT3608, мікроконтролера ESP-32 DevKit V1, OLED дисплея I²C 128×64 1.3″, світлодіодної стрічки WS2812, давача дотику TTP223, пасивного зумера, відсік для акумулятора 18650 та перемикач.

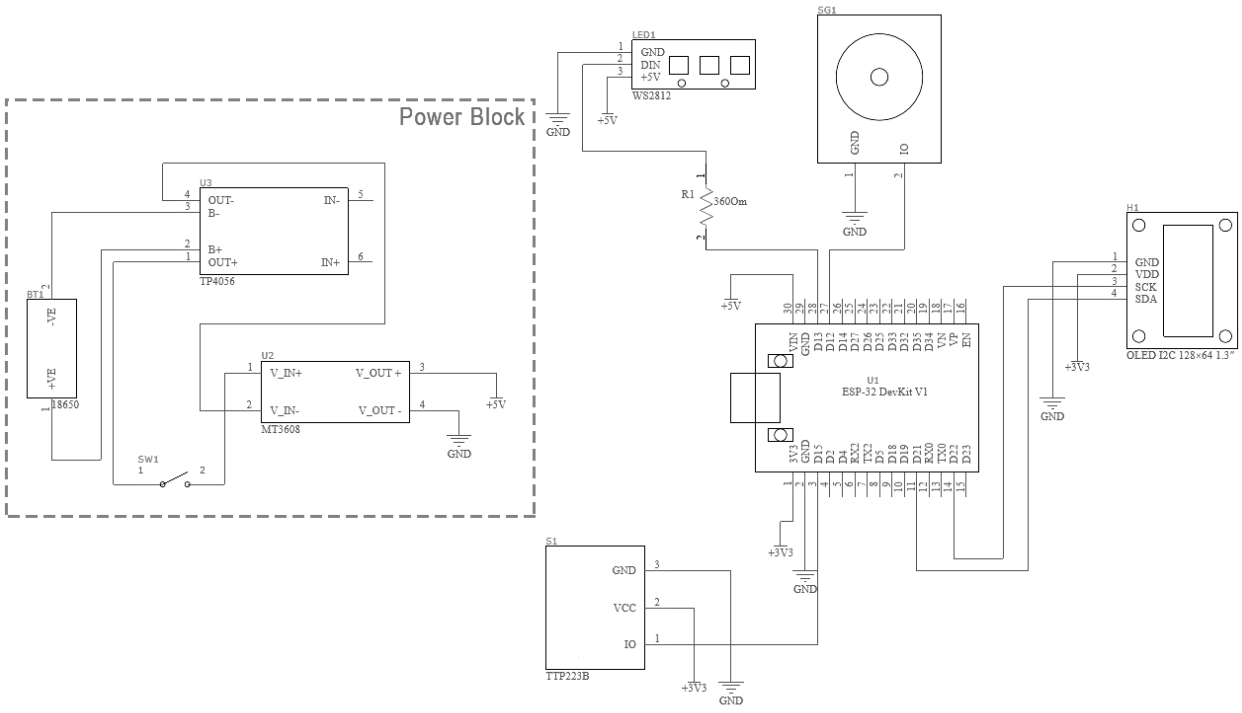

Рисунок 2.12 – Схема електрична принципова комп'ютерної системи “Портативна карта України”

Схема електричних з’єднань, яка зображена на рисунку 2.13 створена в середовищі Altium Designer для підключення компонентів на макетній платі. Вона відображає порядок підключення проводів, розташування компонентів і способи з’єднання усіх елементів, які потрібні для портативної карти України

відповідно до електричної принципової схеми. Такий підхід дозволяє легко збирати і тестувати систему, а також швидко усувати можливі недоліки у з’єднаннях.

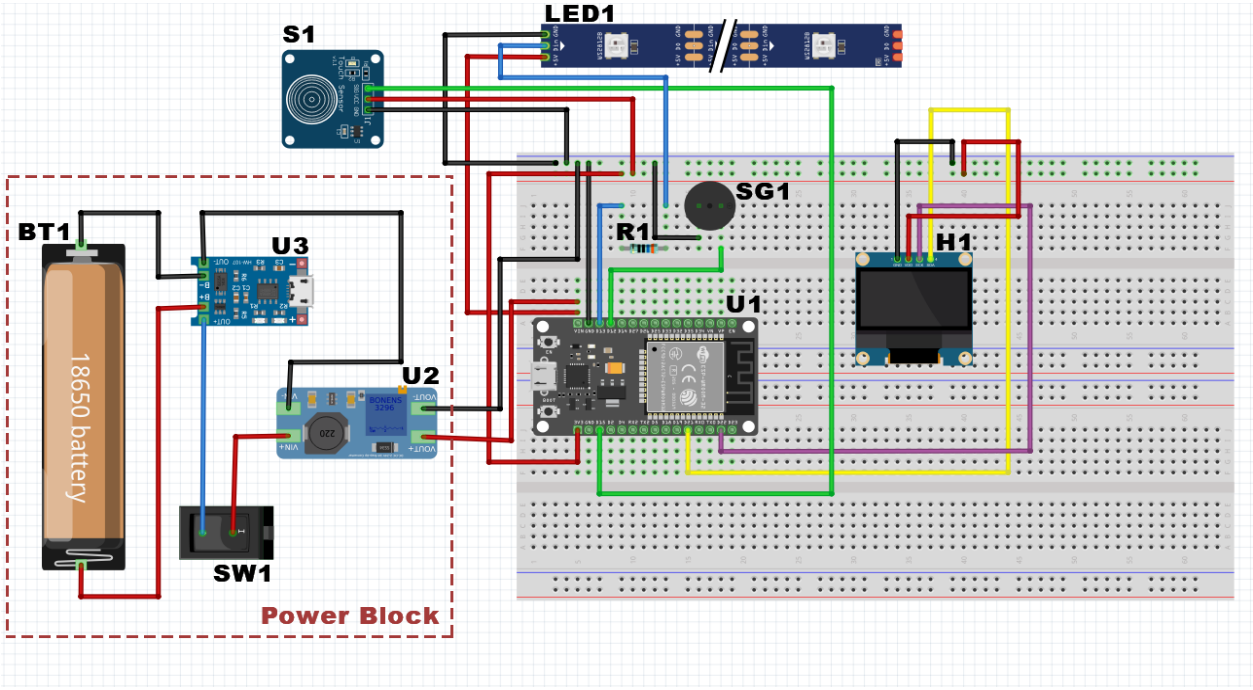

Рисунок 2.13 – Схема електрична з’єднань комп'ютерної системи “Портативна карта України”

Перевірка всіх компонентів згідно із схемою електричною принциповою та схемою електричних з’єднань гарантує коректну роботу схеми, зокрема, живлення, керування, індикації та взаємодії користувача в кожному режимі роботи портативної карти України.

2.3 Опис шин обміну даними

У комп'ютерних системах, що поєднують локальні апаратні компоненти з віддаленими сервісами, архітектура обміну даними будується на виборі протоколів з урахуванням вимог до швидкодії, енергоспоживання та надійності. Насамперед, для передачі команд та отримання інформації з низькошвидкісних периферійних модулів застосовується шина (Inter-

Integrated Circuit). Цей двопровідний протокол забезпечує архітектуру з 7-бітним або 10-бітним адресуванням пристроїв, що підключаються до ліній SDA (Serial Data) та SCL (Serial Clock). Завдяки вбудованим підтягуючим резисторам шина підтримує чіткі рівні логічних сигналів, водночас використовуючи відносно невисоку пропускну здатність (до 400 кГц у Fast Mode), яка вважається достатньою для оновлення графічного інтерфейсу OLED-дисплея зі швидкістю до 30 кадрів на секунду та своєчасного отримання синхронізованого часу по NTP. Також I²C-шина реалізує механізм Clock Stretching, що дозволяє периферійним пристроям уповільнювати передачу даних у разі необхідності обробки, і гарантує стійкість комунікації в умовах нестабільного живлення.

Керування адресованою світлодіодною стрічкою WS2812 здійснюється за допомогою однопровідного протоколу NeoPixel. Він формує одиночний цифровий канал із жорстким таймінгом імпульсів тривалістю приблизно 350 нс для логічного нуля та 900 нс для одиниці. Кожен світлодіод у ланцюжку приймає 24 біт даних (8 біт на кожен колір RGB) і ретранслює решту пакета наступному модулю, що дозволяє формувати складні візуальні ефекти з оновленням усіх 26 світлодіодів менше ніж за 33 мс.

Для виводу діагностичної інформації та налагодження мікроконтролера використовується UART – універсальний асинхронний інтерфейс з швидкістю 115200 бод, форматом кадру 8N1 та мінімальними апаратними вимогами. UART дозволяє передавати журнали стану з’єднання з Wi-Fi, HTTP-коди відповідей та повідомлення про помилки десеріалізації JSON у вигляді потокового тексту, що значно спрощує аналіз роботи системи під час розробки та експлуатації.

Зв’язок із віддаленими сервісами OpenWeather і UkraineAlarm реалізовано через мережевий стек TCP/IP поверх Wi-Fi IEEE 802.11 b/g/n. Після встановлення з’єднання за допомогою WiFi.begin() формується TLS-захищений канал на порті 443 для HTTPS-запитів. HTTP GET-запити до ресурсів /data/3.0/onecall та /v1/alerts/active.json передають параметри

місцезнаходження (latitude, longitude), апаратні ключі доступу та мовні установки, а відповіді у форматі JSON десеріалізуються бібліотекою ArduinoJson з виділенням полів поточної температури, погоди, швидкості вітру, UV-індексу, AQI-індексу та статусу тривог.

Комбіноване використання I²C, однопровідного протоколу WS2812, UART та Wi-Fi/HTTP забезпечує збалансовану архітектуру системи: низькошвидкісні шини реалізують локальний контроль і індикацію, а мережевий інтерфейс гарантує доступ до зовнішніх даних у реальному часі та захищений обмін інформацією.

2.4 Розробка блок-схеми алгоритму роботи та опис бібліотек портативної карти України

Розробка блок-схеми алгоритму забезпечує визначення послідовності логічних операцій, умовних переходів і циклічних процесів, що лежать в основі функціонування системи. У контексті портативної карти України це дозволяє чітко відобразити ініціалізацію апаратних модулів, процедури конфігурації мережі й вибору режимів, а також організацію періодичного опитування зовнішніх сервісів і відображення даних. Завдяки графічному представленню взаємозв’язків між підсистемами забезпечується простота аналізу, тестування й подальшого масштабування програмної архітектури. Алгоритм роботи комп'ютерної системи “Портативної карти України” по суті являє собою цикл (рис. 2.14).

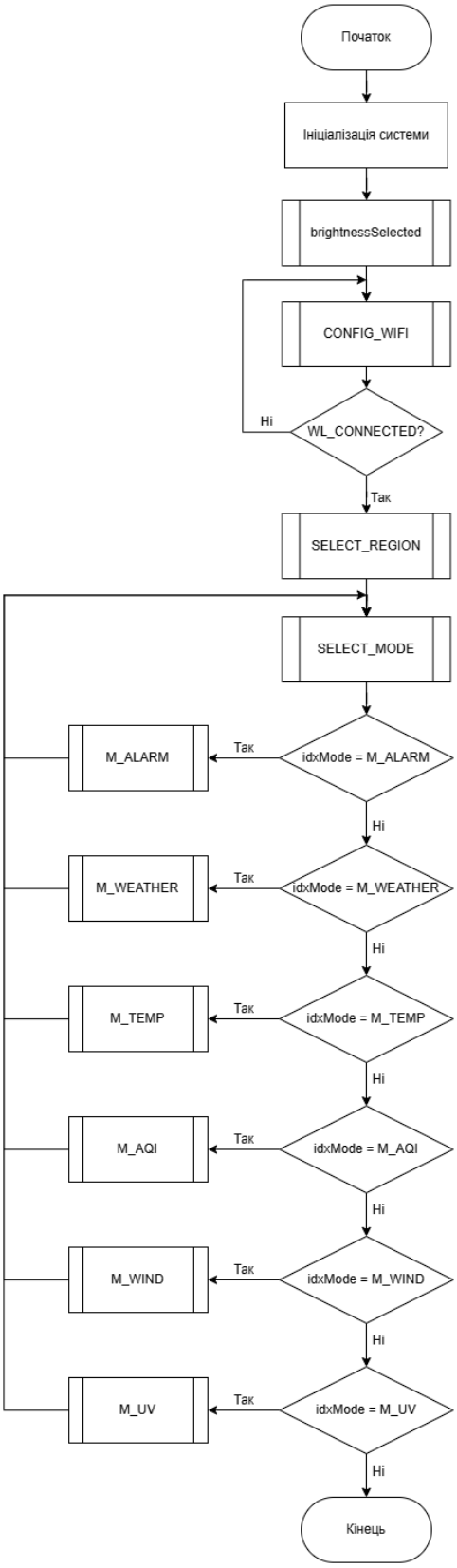

Рисунок 2.14 – Блок-схема алгоритму роботи комп'ютерної системи “Портативна карта України”

Після подачі живлення система переходить у стадію підготовки, яка включає налаштування апаратних інтерфейсів для відображення та зв’язку, а також синхронізацію внутрішнього годинника через мережевий часовий сервер. Далі відбувається ініціалізація світлової стрічки та дисплея з початковими параметрами яскравості, після чого користувачу пропонується інтерфейс вибору інтенсивності підсвічування. Кожен етап у цьому блоці взаємодіє із сенсорним модулем. Короткі дотики змінюють рівень яскравості, тоді як тривале натискання служить фіксацією остаточного значення.

Наступним етапом є налаштування бездротового з’єднання -запуск режиму створення власної точки доступу та надається можливість користувачеві ввести параметри Wi-Fi через веб-інтерфейс. Після отримання даних відбувається перевірка підключення до зовнішньої мережі. У разі успішного встановлення зв’язку здійснюється перехід до режиму вибору географічного регіону і типу відображуваних даних. Збій при підключенні призводить до повторного відкриття точки доступу, що гарантує надійність процедури конфігурації у різних умовах експлуатації.

При виконанні основного циклу відбувається періодичний збір інформації з зовнішніх сервісів щодо актуальних сигналів повітряної тривоги, метеоумов та показників якості повітря, а також індексу ультрафіолетового випромінювання. Інтервали запитів ідентифікуються залежно від природи даних, оскільки, інформація про повітряні тривоги перевіряються частіше, ніж погодна. Отримані значення обробляються з метою візуалізації на світлодіодній стрічці та виводу текстових повідомлень на екран, а також організується плавна прокрутка додаткових рядків. При цьому сенсор дотику залишає можливість повернутися до режиму налаштування відображення без перезавантаження пристрою та втрати накопичених даних

Для написання коду для портативної карти України використано такі бібліотеки: Arduino.h, WiFi.h, WebServer.h, HTTPClient.h, ArduinoJson.h Wire.h, U8g2lib.h, FastLED.h та time.h.

Arduino.h містить реалізації функцій для конфігурації та читання/запису цифрових і аналогових пінів, затримок, серіальної передачі даних тощо, що забезпечує базовий рівень взаємодії з апаратурою.

WiFi.h використовується для організації бездротової конфігурації та подальшого доступу до Інтернету, тобто надає інтерфейс для керування бездротовим модулем ESP32 у режимах точки доступу та клієнта і містить методи запуску та зупинки AP, встановлення параметрів з'єднання, перевірки стану мережі, відображення IP-адреси.

WebServer.h реалізує легкий HTTP-сервер на основі ESP-IDF/WiFi. Дозволяє реєструвати обробники GET і POST-запитів із заданим URI, формувати відповіді із зазначенням MIME-типу та статусу. Використовується для виведення веб-сторінки з формою введення SSID і пароля та для прийому цих параметрів користувачем.

HTTPClient.h забезпечує можливість формувати та відправляти HTTP(S)-запити до сторонніх REST-сервісів. Вміє ініціалізувати підключення, виконувати методи GET/POST, отримувати код відповіді та тіло повідомлення у вигляді рядка. Використовується для опитування API OpenWeather та UkraineAlarm.

ArduinoJson.h надає високопродуктивні засоби для розбору (парсингу) і формування JSON-документів у обмеженому вбудованому середовищі. Підтримує «документ», «масив» та доступ до елементів через ключі. Використовується для виділення температури, швидкості вітру, опису погоди, AQI та інших параметрів із JSON-відповідей серверів.

Wire.h – стандартна бібліотека I²C для Arduino-сумісних плат. Реалізує функції початку обміну як «хост» або «ведений» пристрій, передачі та прийому пакетів даних. Застосовується разом з U8g2 для обміну командами та даними з OLED-дисплеєм по шині I²C.

U8g2lib.h – універсальна бібліотека для малювання у буфер на монохромних дисплеях (OLED, LCD) з апаратним або програмним I²C/SPI. Підтримує векторні шрифти, основні графічні примітиви (лінії, прямокутники,

кола), текстовий вивід з різними стилями. Забезпечує плавний контроль екрану через командний буфер.

FastLED.h – широко використовувана бібліотека для керування адресними RGB-світлодіодами (WS2812, APA102 та ін.), яка дозволяє конфігурувати пін і порядок каналів, встановлювати глобальну яскравість, задавати кольори по індексу, виконувати анімації та ефекти, а також забезпечує швидку передачу даних завдяки оптимізованому часу затримок.

time.h – надає API для роботи з часовою підсистемою: конфігурацію NTP-клієнта, отримання локального часу у вигляді структури tm, перетворення в рядковий формат. Використовується для синхронізації годинника й виводу актуального часу на дисплей.

2.5 Обґрунтування вибору мови програмування

При розробці програмного забезпечення для комп'ютерної системи “Портативна карта України” вирішальним фактором стала необхідність поєднання високої продуктивності та ефективного використання обмежених апаратних ресурсів. Мова C++ у середовищі Arduino IDE була обрана з огляду на оптимальне співвідношення між низькорівневим доступом до периферії та зручністю розробки на високому рівні абстракції. Використання цієї комбінації забезпечує детермінованість операцій вводу-виводу, що критично для організації точного інтервалу опитування зовнішніх сервісів та швидкого реагування інтерфейсів відображення.

Модель пам’яті та управління стеком у C++ дозволяє мінімізувати витрати на динамічне виділення та автоматизоване керування життєвим циклом об’єктів, що особливо важливо в умовах жорстких обмежень оперативної та флеш-пам’яті мікроконтролера.

З огляду на апаратну архітектуру ESP32, мова C++ надає прямий доступ до низькорівневих API. Це дозволяє інтегрувати сторонні бібліотеки, що реалізують HTTP-клієнт, JSON-парсер, графічний інтерфейс OLED-дисплея та

драйвер адресної світлодіодної стрічки, без суттєвих модифікацій. Враховуючи сукупність вимог до своєчасності оновлення інформації, стабільності безперервної роботи та обмеженого енергоспоживання у портативному форматі, мова C++ представилася оптимальним вибором. Її можливості щодо управління апаратурою в режимі реального часу в поєднанні з розширюваною екосистемою бібліотек створюють фундамент для надійного функціонування та подальшого масштабування програмної архітектури [25].

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Реалізація програмної частини комп'ютерної системи “Портативна карта України”

Програмне забезпечення комп'ютерної системи “Портативна карта України” побудовано на основі процедурного підходу з розподілом відповідальності між окремими модулями. Код для цієї системи написаний у середовищі Arduino IDE. На блок-схемі алгоритму роботи комп’ютерної системи «Портативна карта України» показано послідовність виконання основних операцій. На етапі ініціалізації відбувається налаштування апаратних інтерфейсів і запуск службових процесів. Після старту застосунок входить у режим створення точки доступу для конфігурації Wi-Fi, що забезпечує просте введення SSID і пароля через веб-інтерфейс (рис. 3.1).

```
#include<Arduino.h> #include<WiFi.h> #include<WebServer.h>
#include<HTTPClient.h> #include<ArduinoJson.h> #include<Wire.h>
#include<U8g2lib.h> #include<FastLED.h> #include<time.h>
const char* OWM_API_KEY="5052a733bde7573036a91a81f40a5753";
const char*
UA_API_URL="https://api.alerts.in.ua/v1/alerts/active.json?token
=f74afc00edb18aa625aba79abc0471519ba54e5cab2203";
WebServer server(80); #define PIN_OLED_SDA 21 #define
PIN_OLED_SCL 22 #define PIN_LEDS 13 #define PIN_BUZZER 12
#define PIN_TOUCH 15
U8G2_SSD1306_128X64_NONAME_F_SW_I2C
u8g2(U8G2_R0,PIN_OLED_SCL,PIN_OLED_SDA,U8X8_PIN_NONE);
#define NUM_LEDS 26 CRGB leds[NUM_LEDS];enum
AppState{STARTUP,CONFIG_WIFI,CONNECTING_WIFI,SELECT_REGION,SELEC
T_MODE,RUN_MODE};enum
Mode{M_ALARM,M_WEATHER,M_TEMP,M_AQI,M_WIND,M_UV,M_COUNT};
const char* modeNames[M_COUNT]={"Air Raid
Alarm","Weather","Temperature","Air Quality","Wind Speed","UV
Index"};
```

Рисунок 3.1 – Лістинг підключення бібліотек, ініціалізації периферії та запуску WebServer

Після збереження налаштувань інтерфейс переводить пристрій у режим підключення до мережі (CONNECTING_WIFI). У разі успіху WebServer зупиняється, а система переходить до меню вибору регіону. У разі невдачі цикл запуску точки доступу відновлюється.

Меню вибору регіону (SELECT_REGION) реалізує функції showSelectRegion() та processSelectRegion(), забезпечуючи зручну навігацію через коротке та довге натискання (рис. 3.2).

```
void showSelectRegion(){
  u8g2.clearBuffer();
  u8g2.setCursor(2,12);
  u8g2.print("Select Region:");
  u8g2.setCursor(2,28);
  u8g2.print(regions[idxRegion].nameShort);
  u8g2.sendBuffer();}
void processSelectRegion(){
  uint8_t t=readTouchDigital();
  if(t==1) idxRegion=(idxRegion+1)%NUM_LEDS;
  else if(t==2) appState=SELECT_MODE;}
```

Рисунок 3.2 – Лістинг меню вибору регіону

Метод readTouchDigital() реалізує запобігання короткочасним перешкодам сигналу, вимірює час натиснення та повертає значення 1 для короткого і 2 для довгого натискання. Для запобігання дрібним завадам використовується затримка 300 мс між послідовними натисненнями (рис. 3.3).

```
uint32_t lastTouchMillis = 0;
uint8_t readTouchDigital(){
  static uint32_t t0 = 0;
  int raw = digitalRead(PIN_TOUCH);
  if(raw == HIGH){
    if(!t0) t0 = millis();
  } else if(t0){
    uint32_t dt = millis() - t0;
    t0 = 0;
    if(millis() - lastTouchMillis < 300) return 0;
    lastTouchMillis = millis();
    return (dt > 800) ? 2 : 1;
  }
  return 0;}
```

Рисунок 3.3 – Лістинг функції обробки коротких та довгих натискань

Для конфігурації яскравості світлодіодів (brightnessMenu) передбачено відображення варіантів та збереження вибору в FastLED.setBrightness(). Підтримка трьох рівнів забезпечує оптимізацію енергоспоживання та читабельності (рис. 3.4).

```
const uint8_t brightnessLevels[3]={20,128,255};
uint8_t idxBrightness=1;
bool brightnessSelected=false;
void showBrightnessMenu(){
  u8g2.clearBuffer();
  u8g2.setCursor(2,12);
  u8g2.print("Brightness:");
  u8g2.setCursor(2,28);
  u8g2.print(brightnessLevels[idxBrightness]);
  u8g2.sendBuffer();}
void processBrightnessMenu(){
  uint8_t t=readTouchDigital();
  if(t==1) idxBrightness=(idxBrightness+1)%3;
  else if(t==2){
    brightnessSelected=true;
    FastLED.setBrightness(brightnessLevels[idxBrightness]);}}
```

Рисунок 3.4 – Лістинг меню вибору яскравості

Після вибору регіону та яскравості користувач обирає режим відображення (SELECT_MODE) за допомогою функцій showSelectMode() та processSelectMode(). Вибір підтверджується довгим натисканням, що запускає цикл оновлення даних (рис. 3.5).

```
void showSelectMode(){
  u8g2.clearBuffer(); u8g2.setCursor(2,12);
  u8g2.print("Select Mode:");
  u8g2.setCursor(2,28);
  u8g2.print(modeNames[idxMode]);
  u8g2.sendBuffer();}
void processSelectMode(){
  uint8_t t=readTouchDigital();
  if(t==1) idxMode=(idxMode+1)%M_COUNT;
  else if(t==2){
    appState=RUN_MODE;
    tLastAlarm=0; tLastWeather=0; scrollBuf="";
    if(idxMode==M_ALARM) updateAlarm();
    else if(idxMode==M_UV){ updateUVAllRegions();
updateScrollUV(); } else updateWeatherAllRegions();}}
```

Рисунок 3.5 – Лістинг меню вибору режиму

Функція renderModeLED відповідає за зміни кольору світлодіодів залежно від вибраного режиму. У цьому лістингу наведені виклики функцій визначення кольору для кожного індикатора (рис. 3.6).

```
void renderModeLED(){
  for(int i=0;i<NUM_LEDS;i++){
  switch(idxMode){
  case M_ALARM: break;
  case M_WEATHER: leds[i]=colorForWeather(weatherDescs[i]);
  break;
  case M_TEMP:    leds[i]=colorForTemp(temps[i]);  break;
  case M_AQI:     leds[i]=colorForAQI(aqis[i]);    break;
  case M_WIND:    leds[i]=colorForWind(winds[i]); break;
  case M_UV:      leds[i]=colorForUV(uvs[i]); break;
  default:        leds[i]=CRGB::Black;    break;}}
  FastLED.show();}
```

Рисунок 3.6 – Лістинг встановлення кольорів LED за режимами

Функції colorFor... реалізовані відповідно до шкал вимірюваних величин (рис. 3.7).

```
CRGB colorForTemp(float t){
  if(t<=-20) return CRGB::Blue;
  else if(t<=0) return CRGB::Green;
  else if(t<=20) return CRGB::Orange;
  else return CRGB::Red;}
CRGB colorForWind(float w){
  if(w<=1) return CRGB::White;
  else if(w<=10) return CRGB::Blue;
  else return CRGB::Magenta;}
CRGB colorForAQI(uint8_t a){ switch(a){
    case 1: return CRGB::Green;
    case 2: return CRGB::Yellow;
    case 3: return CRGB::Orange;
    case 4: return CRGB::Red;
    default: return CRGB::Gray;}}
CRGB colorForUV(float uv){
  if(uv<=2) return CRGB::White;
  else if(uv<=5) return CRGB::Blue;
  else return CRGB::Red;}
CRGB colorForWeather(const String&s){
  String d=s; d.toLowerCase();
  if(d.indexOf("clear")>=0) return CRGB::Yellow;
  if(d.indexOf("cloud")>=0) return CRGB::White;
  if(d.indexOf("rain")>=0)  return CRGB::Blue;
  return CRGB::Green;}
```

Рисунок 3.7 – Лістинг функції визначення кольору

Основний цикл loop() реалізує керування станами та періодичне оновлення даних. Для режиму тривоги викликається updateAlarm(), який здійснює GET-запит до UkraineAlarm API та оновлює світлодіоди на основі UID. Для метеорологічних режимів дані збираються функціями updateWeatherAllRegions() і updateUVAllRegions(), які об'єднують HTTP-запити до OpenWeather та Open-Meteo (рис. 3.8)

```
void updateAlarm(){
  HTTPClient http;
  http.begin(UA_API_URL);
  if(http.GET()==200){
    String payload = http.getString();
    for(int i=0;i<NUM_LEDS;i++) leds[i] = CRGB::Green;
    for(int k=0;k<UID_TO_LED_COUNT;k++){
      int uid = uidToLed[k].uid;
      int li  = uidToLed[k].ledIndex;
      if(payload.indexOf("\"location_oblast_uid\":" +
String(uid)) != -1){
        leds[li] = CRGB::Red;
      }
    }
  }
  http.end();
  FastLED.show();}

void updateWeatherAllRegions(){
  for(uint8_t i=0;i<NUM_LEDS;i++){
    updateWeatherForOne(i);
    delay(100);}
  String buf;
  for(uint8_t i=0;i<NUM_LEDS;i++){
    buf += regions[i].nameShort + ":" + String((int)temps[i]) +
"C   ";}
  updateScroll(buf);}
```

Рисунок 3.8 – Лістинг функцій оновлення даних з тривог та погоди

Крім того, для коректного відображення текстової інформації на дисплеї реалізовано обробку прокрутки рядків у updateScroll() та getScrollOne(), що дозволяє відображати довгі рядки без зміни форматування (рис. 3.9).

```
String scrollBuf;
uint16_t scrollPos=0;
void updateScroll(const String& txt){
  scrollBuf=txt+"   ";
  scrollPos=0;}
String getScrollOne(){
  if(scrollBuf.length()==0) return "";
  String
s=scrollBuf.substring(scrollPos)+scrollBuf.substring(0,scrollPos
);
  scrollPos=(scrollPos+1)%scrollBuf.length();
  return s;}
```

Рисунок 3.9 – Лістинг обробки прокрутки буфера для OLED

Для отримання UV-індексу застосовується спеціалізована функція fetchUV_OpenMeteo(), яка робить запит до Open-Meteo API, десеріалізує тижневий масив показників та вибирає значення для поточного часу на основі локального RTC (рис. 3.10).

```
float fetchUV_OpenMeteo(double lat,double lon){
  char url[256];
  snprintf(url,sizeof(url),
    "https://air-quality-api.open-meteo.com/v1/air-quality?"
"latitude=%.4f&longitude=%.4f&hourly=uv_index&timezone=Europe%%2
FKyiv",
    lat,lon);
  HTTPClient http;http.begin(url);
  if(http.GET()==200){
    DynamicJsonDocument d(16384);
    deserializeJson(d,http.getString());
    JsonArray times=d["hourly"]["time"].as<JsonArray>();
    JsonArray vals=d["hourly"]["uv_index"].as<JsonArray>();
    struct tm tm; getLocalTime(&tm);
    char buf[20]; strftime(buf,sizeof(buf),"%Y-%m-
%dT%H:00",&tm);
    for(size_t i=0;i<times.size();i++) if(String(times[i])==buf)
return vals[i].as<float>();}
  http.end();
  return 0.0;}
```

Рисунок 3.10 – Лістинг отримання UV-індексу з Open-Meteo

Таким чином, система забезпечує повноцінну взаємодію з користувачем та серверами даних, реалізуючи всі етапи вибору та індикації за допомогою компактних лістингів програмного коду.

3.2 Реалізація апаратної частини комп'ютерної системи “Портативна карта України”

Апаратна реалізація портативної карти України спроектована з урахуванням вимог до компактності, надійності та простоти монтажу. Центральним елементом системи є плата ESP-32 DevKit V1, яка відповідає за обробку вхідних сигналів, управління LED-індикацією та комунікацію з віддаленими сервісами. Першою стадією збірки стало підготовлення індикаторної стрічки WS2812: стандартна гнучка LED-стрічка була розрізана на 26 окремих сегментів, кожен з яких призначений для відображення стану окремої області України. Кожен сегмент складається з трьох провідників: шина живлення +5 В, загальний провід (GND) і провід передачі даних (DIN). При пайці кожного сегмента використано твердотільний дріт із багатожильним жилим перерізом 0.07 мм², що гарантує мінімальний опір при передачі струму до кількох сотень міліампер на сегмент. Сегменти були з’єднані послідовно, забезпечуючи ланцюжковий потік даних від першого до останнього світлодіода (рис. 3.11).

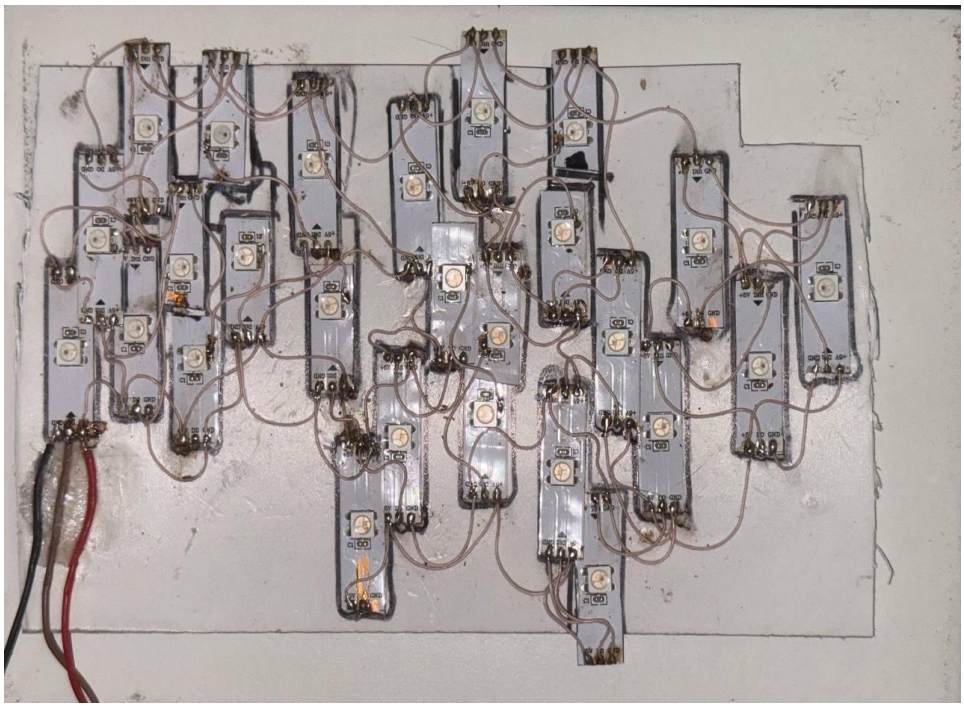

Рисунок 3.11 – Макет сегментів світлодіодної стрічки

Блок живлення реалізовано за модульною схемою, що включає літій-іонний акумулятор формату 18650, модуль зарядки TP4056 із USB Type-C та підвищувальний DC–DC конвертер MT3608, з’єднані послідовно. Акумулятор підключається до контактів B+ та B– плати TP4056, яка підтримує заряд струмом до 1 А і оснащена захистом від перезаряду та короткого замикання. Вихідні клеми OUT+ та OUT– TP4056 через важільний перемикач з’єднані з входом IN+ та IN– конвертера MT3608. Потенціометр на платі MT3608 дозволяє точно виставити вихідну напругу +5 В. Вимірювання напруги мультиметром є обов’язковим перед підключенням ESP32. Вихідні клеми OUT+ та OUT– MT3608 формують шину живлення +5 В та загальний провід GND для всієї системи (рис. 3.12).

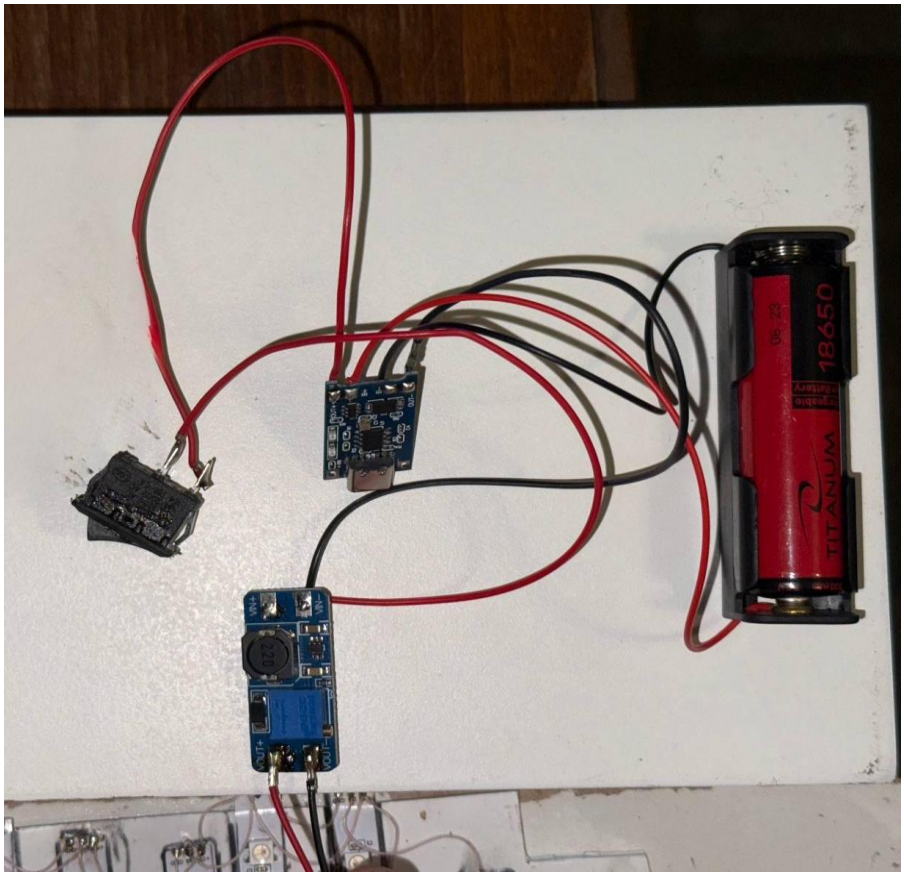

Рисунок 3.12 – Компоненти живлення

Усі модулі з’єднані за допомогою проводів довжиною 10–15 см. Для зручності обслуговування використовувалися макетні платформи, куди були

під’єднані провода до основних роз’ємів. Основні лінії: +5 В і GND прокладені окремими жилами для кожного сегмента, щоб запобігти просіданню напруги.

Плата ESP-32 DevKit V1 розташована у корпусі, що виготовлений із дерев’яної фанери товщиною 6 мм та покрашено в білий колір. До цифрових портів підключено: OLED-дисплей I²C 128×64, вхід LED-стрічки WS2812 через резистор 360 Ом, пасивний зумер, сенсор дотику TTP223. Лінія 5 В ESP32 живиться від виходу MT3608, а загальний провід GND спільний для усього кола.

Конструктивно корпус має верхню кришку, яка виконана з PLA-пластика методом 3D-друку на 3D-принтері Anycubic S кафедри комп’ютерних систем та мереж Тернопільського національного технічного університету імені Івана Пулюя. Модель була розроблена у програмі Blender, щоб задовільнити усі вимоги для реалізації портативної карти України (рис. 3.13) [26].

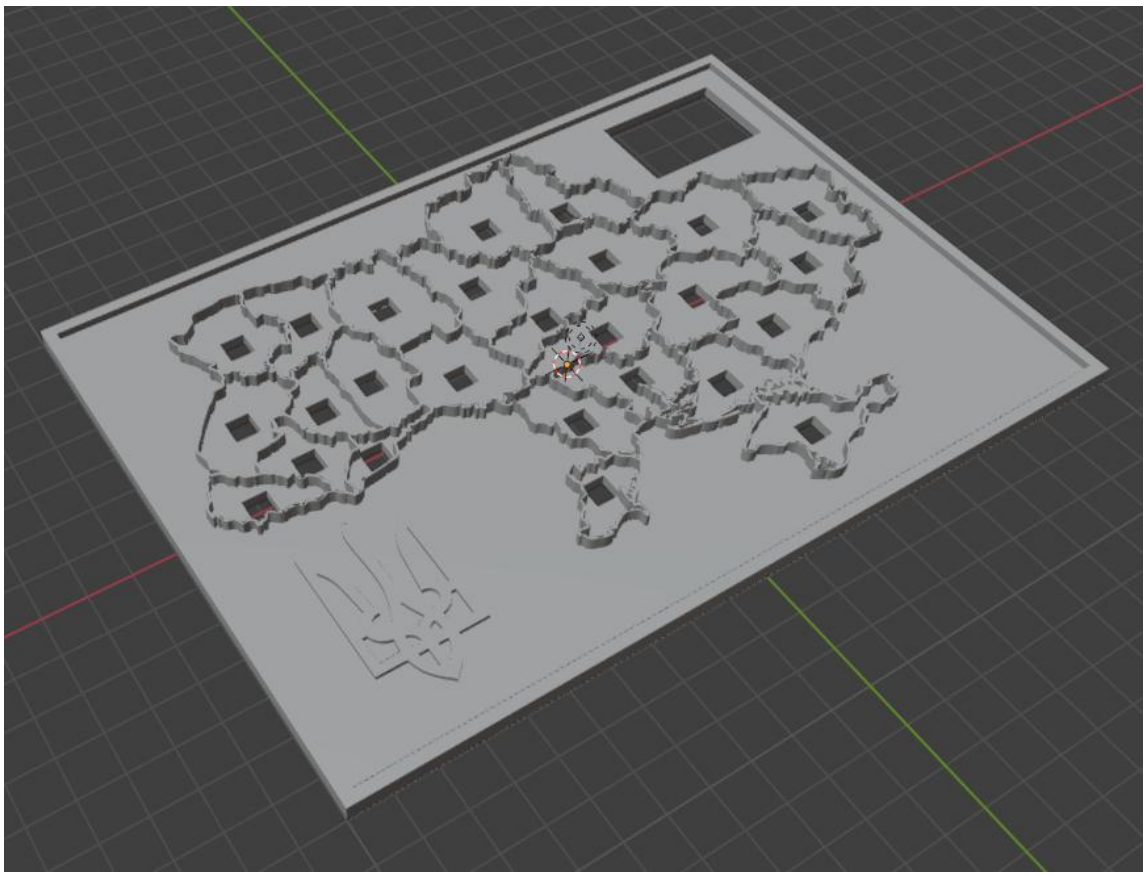

Рисунок 3.13 – Модель карти України

Кришка міцно закріплена до корпусу та містить рельєфне зображення карти України з виділенням усіх областей, а також з вирізами для кожного світлодіода та отвором для OLED-екрану (рис. 3.14).

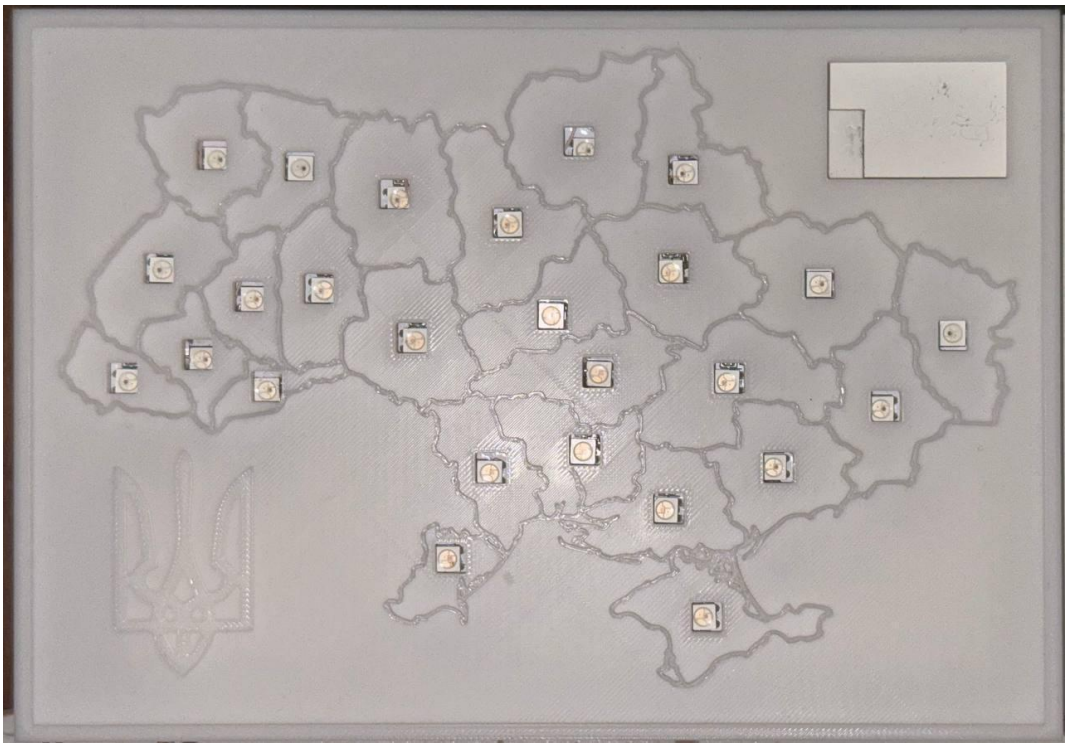

Рисунок 3.14 – Роздрукована на 3D-принтері модель карти України

По периметру дерев’яного корпусу вирізані отвори для встановлення перемикача живлення, сенсора дотику, micro-USB роз’єму для програмування ESP32 та USB Type-C для зарядки акумулятора (рис. 3.15).

Рисунок 3.15 – Прототип комп'ютерної системи “Портативна карта України” в корпусі

У цілому, конструкція та апаратна реалізація враховує баланс між зручністю монтажу, доступністю компонентів та механічною міцністю. Представлена архітектура сприяє подальшому масштабуванню та інтеграції нових датчиків або модулів без суттєвих змін базової схеми.

3.3 Налаштування та тестування системи

Для початку роботи з портативною картою України слід встановити середовище розробки Arduino IDE. Після завантаження останньої версії IDE з офіційного сайту необхідно додати підтримку плати ESP32, в меню “Менеджер плат” потрібно знайти пакет esp32 by Espressif Systems та натиснути Install (рис. 3.16) [27].

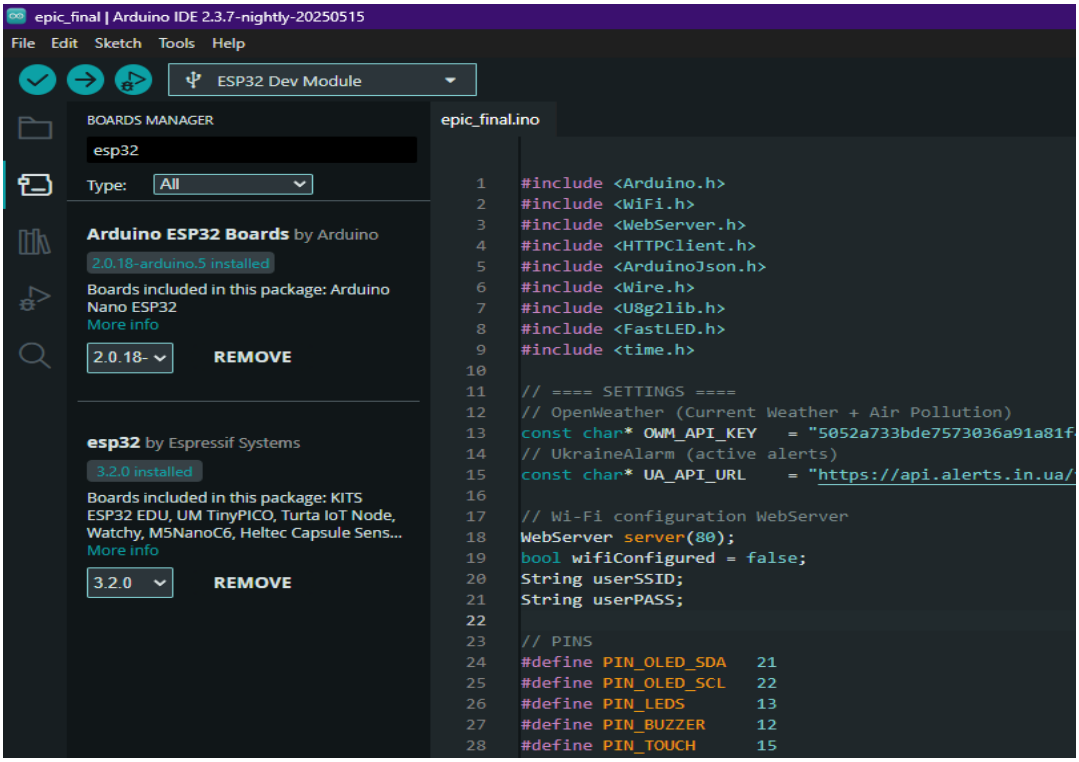

Рисунок 3.16 – Встановлення підтримки ESP32 в Arduino IDE

Після успішної інсталяції плат і драйверів слід обрати в панелі інструментів відповідну плату ESP32 Dev Module та COM-порт, що відповідає підключеній платі. Для запису програми використовується кнопка Upload (стрілка праворуч), яка запускає процес компіляції та передачі даних у флеш-пам’ять контролера. У разі успішного завершення компіляції та завантаження прошивки внизу IDE з’явиться відповідне повідомлення, що свідчить про готовність пристрою до використання (рис. 3.17).

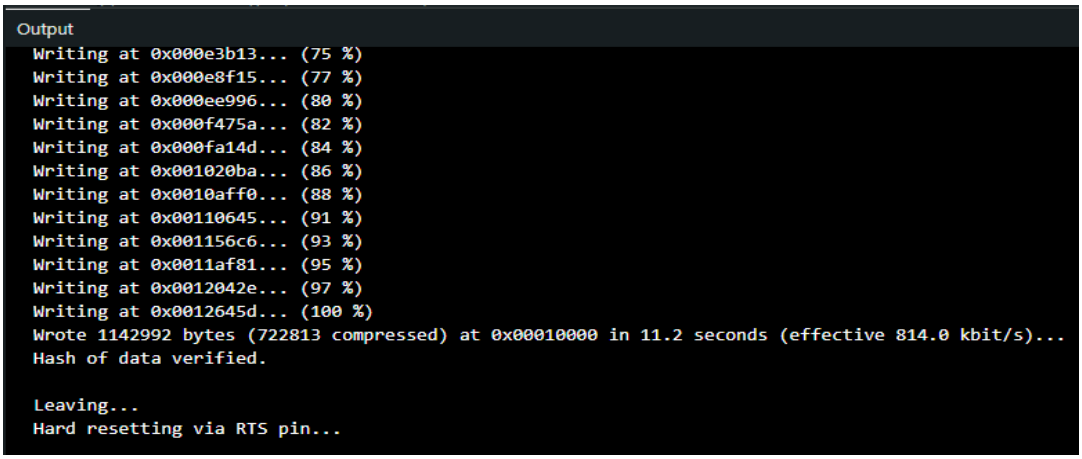

Рисунок 3.17 – Повідомлення про успішну компіляцію та завантаження прошивки

Після прошивки пристрій перезавантажується та переходить у режим налаштування яскравості LED-стрічки. На OLED-дисплеї з’являється меню, де коротким дотиком сенсорної кнопки можна послідовно переглянути три рівні яскравості, а довгим – вибрати поточний. Це дозволяє адаптувати інтенсивність світлових індикаторів до умов освітлення та зберегти ресурс акумулятора. (рис. 3.18).

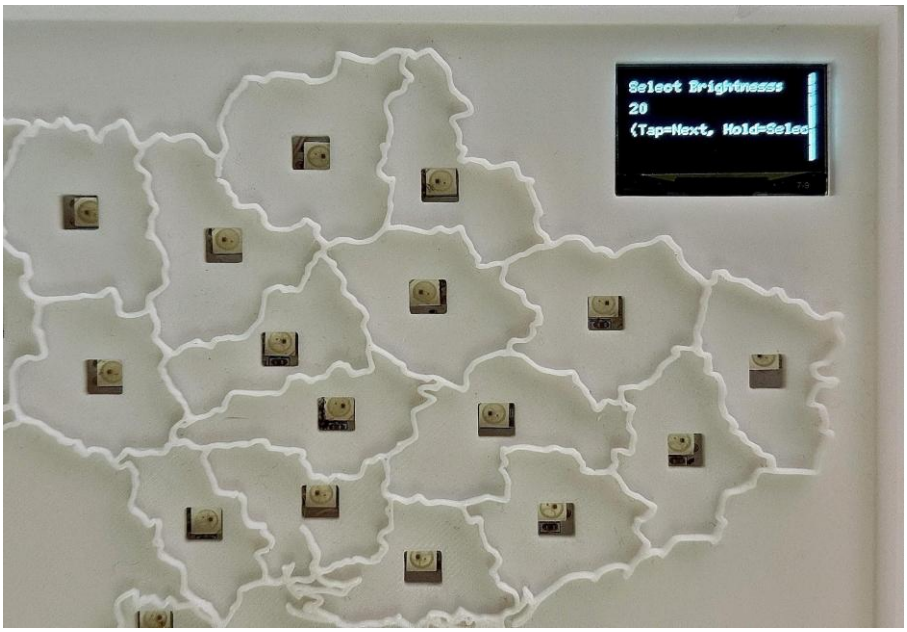

Рисунок 3.18 – Меню вибору рівня яскравості

Після вибору рівня яскравості пристрій створює власну Wi-Fi точку доступу з SSID ESP32_Config та відкриває вбудований веб-сервер на порті 80. Підключившись до цієї мережі з будь-якого браузера, користувач бачить форму введення SSID і пароля мережі. Після введення даних та натискання Save плата автоматично застосовує налаштування та спробує підключитися до зазначеної мережі (рис. 3.19).

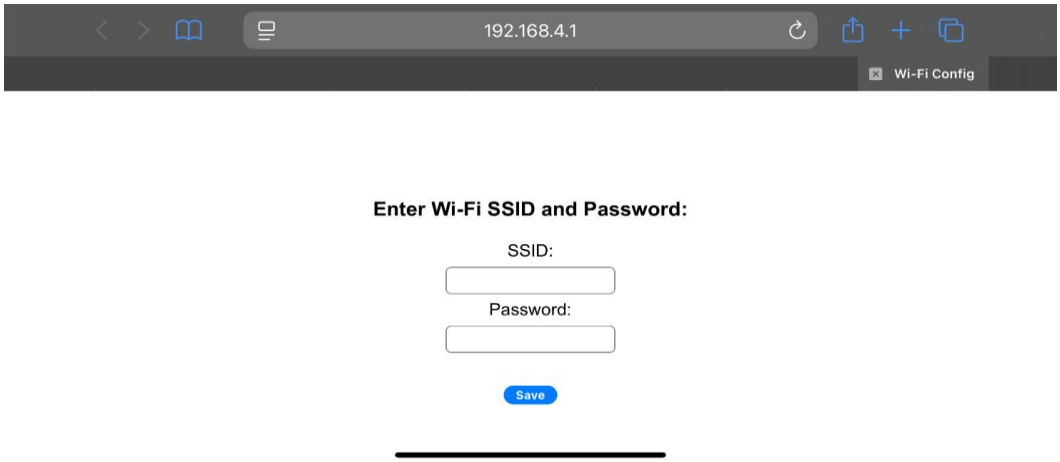

Рисунок 3.18 – Форма введення SSID і пароля мережі

При успішному підключенні ESP32 відображає на дисплеї повідомлення про встановлену мережу та вимикає режим точки доступу. Далі на екрані з’являється меню вибору області України, де короткий дотик рухає курсор по списку, а довгий підтверджує регіон (рис. 3.19).

Рисунок 3.19 – Меню вибору регіону

Після цього користувач переходить у меню вибору режиму: Air Raid Alarm, Weather, Temperature, Air Quality, Wind Speed та UV Index. Кожен режим супроводжується оновленням LED-стрічки та відповідним текстовим повідомленням на OLED (рис. 20).

Рисунок 3.19 – Меню вибору режиму

У режимі повітряних тривог система кожні 30 секунд надсилає запит до сервера UkraineAlarm і аналізує отриманий JSON-потік. Для кожної області за відсутності сигналу тривоги світлодіод світиться зеленим, а при фіксації активної тривоги змінює колір на червоний. Паралельно з оновленням індикації у вибраному регіоні відтворюється звуковий сигнал, що дозволяє одразу звернути увагу користувача на початок або завершення тривоги (рис. 3.20).

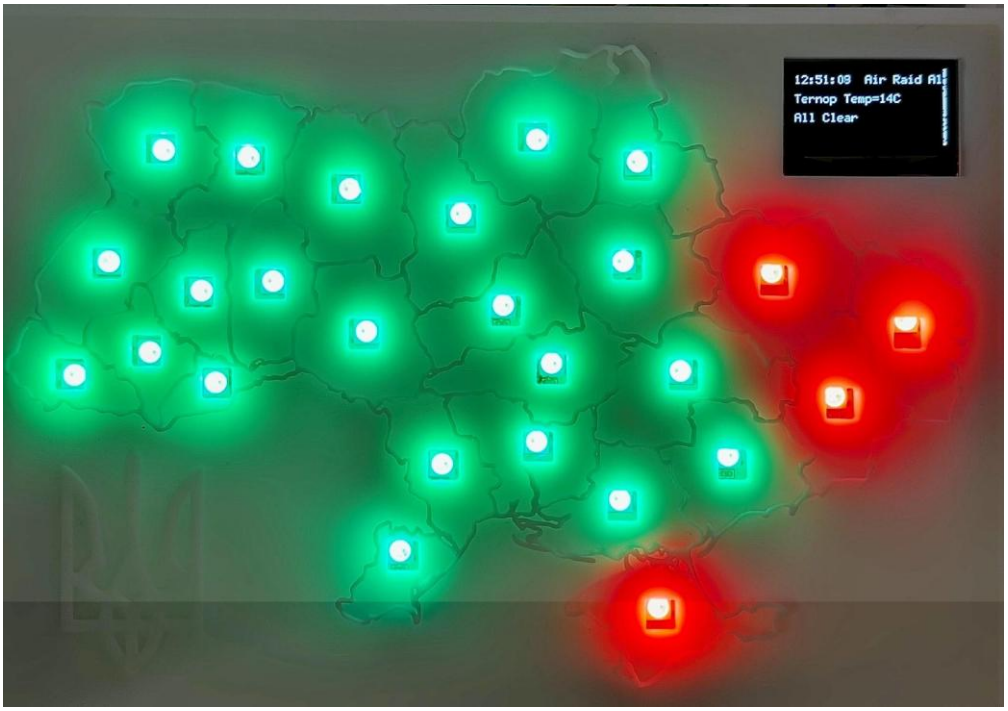

Рисунок 3.20 – Режим повітряних тривог

У режимі відображення погоди кожен світлодіод реагує на текстовий опис умов з OpenWeather API: ясне небо позначається жовтим кольором, хмарність – білим, дощ або сніг – синім, туман –пурпуровим, сильний вітер – бірюзовим, тоді як інші метеоумови отримують зелений колір, що дає візуальну картину погодних явищ по всіх регіонах (рис. 3.21).

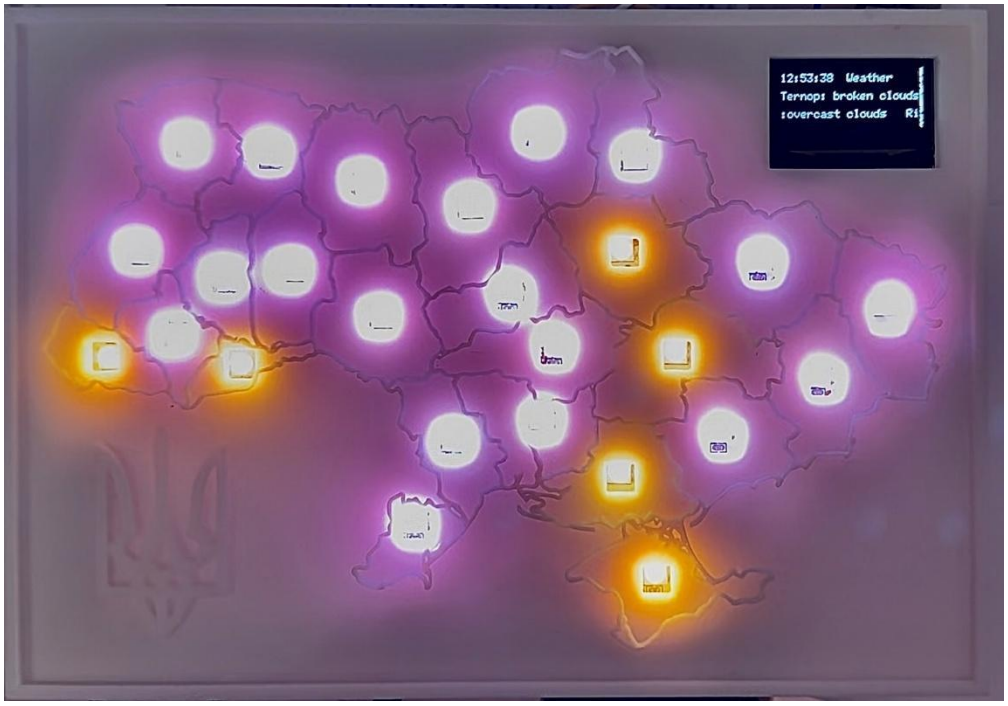

Рисунок 3.21 – Режим погоди

Під час відображення температури кожен світлодіод світиться кольорами відповідно такої шкали: значення до –20 °C відображаються синім, від –20 до –15 °C – індикацією синьо-фіолетового відтінку, від –15 до –10 °C – бірюзовим, від –10 до –5 °C – блакитним, від –5 до 0 °C – зеленим, від 0 до 5 °C – жовто-зеленим, від 5 до 10 °C – жовтим, від 10 до 15 °C – помаранчевим, від 15 до 20 °C – червоним, від 20 до 25 °C – пурпуровим, вище 25 °C – фіолетовим. Така детальна градація дозволяє не лише визначити характерні кліматичні умови, а й оцінити ступінь теплового комфорту по всіх регіонах (рис. 3.22).

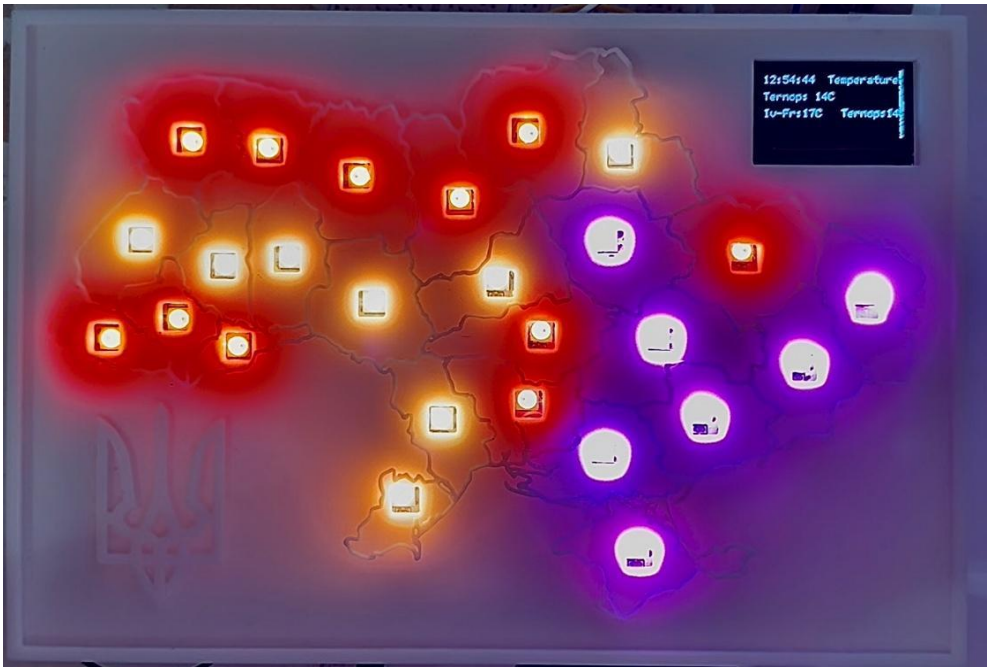

Рисунок 3.22 – Режим температури

У режимі якості повітря (AQI) застосовується п’ятибальна шкала: індекс 1 характеризується зеленим світлом, 2 – жовтим, 3 – помаранчевим, 4 – червоним, 5 – пурпуровим; при відсутності чи невизначеності даних світлодіод відображає сірий колір. Це дає швидке розуміння екологічного стану в режимі реального часу (рис. 3.23).

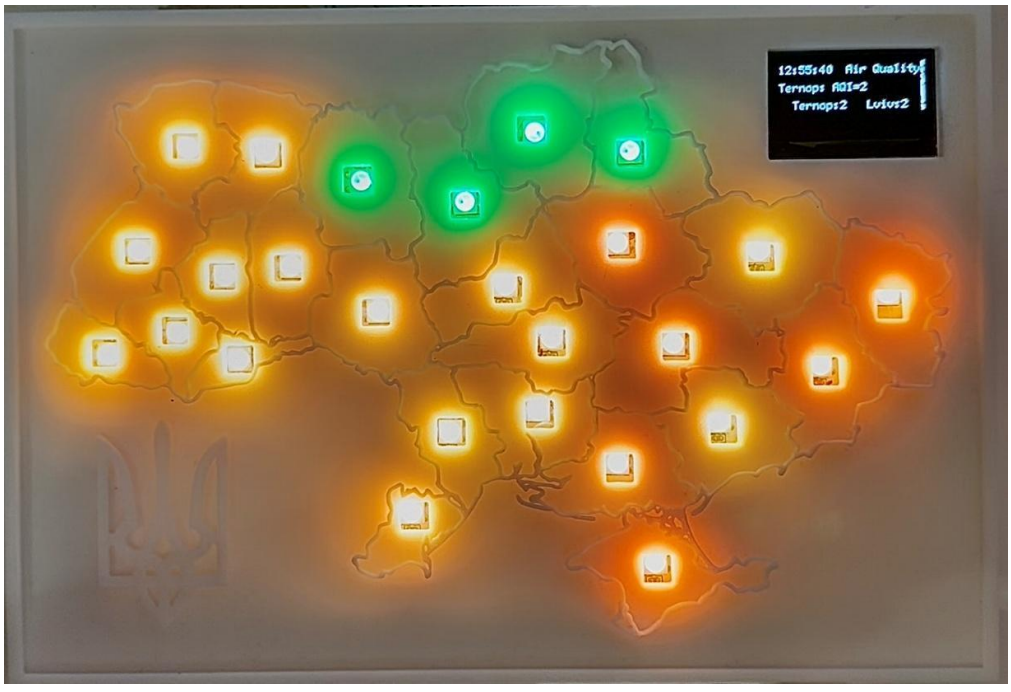

Рисунок 3.23 – Режим якості повітря

Режим швидкості вітру кодує інтенсивність таким чином: від 0 до 1 м/с – біле підсвічування, від 1 до 5 м/с – бірюзове, від 5 до 10 м/с – синє, від 10 до 15 м/с – пурпурове, більше 15 м/с – червоне підсвічування. Цей підхід дозволяє одразу візуалізувати силу вітру на карті (рис. 3.24).

Рисунок 3.24 – Режим швидкості вітру

У режимі індексу ультрафіолетового випромінювання (UV Index) використовується п’ятибальна шкала: від 0 до 2 – білий, від 3 до 5 – синій, від 6 до 7 – пурпуровий, від 8 до 10 – червоний, вище 10 – фіолетовий. Завдяки цьому користувач може оцінити рівень УФ-опромінення та вжити заходів захисту в разі необхідності (рис. 3.25).

Рисунок 3.25 – Режим індексу ультрафіолетового випромінювання

Проведене комплексне тестування підтвердило коректну роботу усіх функцій: меню інтерфейсу, мережевих запитів, обробки введення користувача та відображення результатів. Система успішно пройшла перевірку на стійкість до багаторазових перезапусків, автоматично відновлює Wi-Fi з’єднання при короткочасних перервах і коректно реагує на зміну рівня заряду акумулятора.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Загальний стан безпеки життєдіяльності в Україні

Система охорони здоров’я України формується на засадах єдиного державного медичного простору та включає первинну, вторинну та третинну ланки. У 2023 р. на первинну медичну допомогу припадало понад 70 % звернень пацієнтів із хронічними неінфекційними захворюваннями, головними серед яких є серцево-судинні та ендокринні патології. Після завершення окремих епідемічних спалахів COVID-19 організовано програми скринінгу раку шийки матки, грудної залози та колоректального раку. Проте нестача фахівців в сільській місцевості продовжує залишатися критичною проблемою: на 100 000 населення тут припадає лише 32 лікарі проти середньоєвропейських 40–45 [28].

Державна служба України з надзвичайних ситуацій (ДСНС) забезпечує запобігання, локалізацію та ліквідацію наслідків пожеж, техногенних аварій і природних катастроф. У 2023 р. зареєстровано понад 25 000 пожеж, 1 200 хімічних забруднень та 350 гідрометеорологічних інцидентів. Щоденно рятувальники здійснюють близько 70 оперативних виїздів, із них 15 % – до зон, наближених до лінії бойового зіткнення. Для персоналу та місцевого населення проводяться регулярні навчання з хімічного, радіаційного та біологічного захисту на засадах євростандартів ICSU 22300 [29].

Аварійність на дорогах України залишається високою: у 2023 р. зафіксовано 23 642 ДТП із понад 3 000 загиблих та 29 500 травмованих. Основні чинники – перевищення швидкості й порушення правил маневрування. З метою зниження показників запроваджено автоматизовані системи контролю швидкості, розширено мережу лежачих поліцейських та

активно впроваджуються освітні кампанії з безпеки руху в навчальних закладах [30].

Виробнича безпека регулюється кодексом законів про працю та спеціальними галузевими нормами, які зобов’язують роботодавців проводити атестацію робочих місць і навчати працівників з охорони праці. У 2023 році рівень нещасних випадків на підприємствах знизився на 8 % порівняно з попереднім роком завдяки впровадженню електронної системи управління ризиками і аудитам відповідності вимогам ДСТУ ISO 45001 [31].

Екологічна безпека включає моніторинг якості повітря, води та ґрунтів. У містах із населенням понад 100 000 осіб середньорічна концентрація $PM_{2.5}$ перевищує 40 мкг/м³, що вдвічі вище за рекомендовані ВООЗ значення. Для зменшення забруднення впроваджуються відновлювані джерела енергії, розширюються зелені зони та модернізуються очисні споруди, зокрема в депресивних промислових регіонах [32].

Радіаційна безпека залишається важливим завданням внаслідок Чорнобильської катастрофи 1986 р. Зони безумовного відселення охоплюють понад 2 800 км², де проводиться постійний контроль рівнів гамма-фону та радіонуклідів у продуктах харчування [32].

Хімічна безпека охоплює контроль шкідливих речовин на промислових об’єктах та транспортуванні. У 2023 р. зафіксовано 67 аварій із витоком аміаку та хлору; серед заходів – сертифікація ємностей за європейськими стандартами ADR та навчання аварійно-рятувальних бригад із застосуванням сучасних засобів індивідуального захисту [32].

Система цивільного захисту інтегрує нормативно-правові акти (Кодекс цивільного захисту України, ДБН В.1.1-7:2016) з організаційними заходами: розробка планів евакуації, встановлення систем оповіщення та регулярні тренування в усіх освітніх і виробничих закладах. У 2023 р. провели понад 15 000 навчань з евакуації та протипожежного захисту, що на 20 % більше за показник 2022 р. [31].

4.2 Електромагнітні випромінювання: спектр, біологічний вплив і методи захисту

Електромагнітні поля (ЕМП) та радіочастотне випромінювання охоплюють широкий діапазон частот – від ультранизьких (< 300 Гц) до надвисоких (до сотень ГГц). У традиційному поділі виокремлюють індустріальну НЧ-ділянку (50–400 МГц), комунікаційну СВЧ-смугу (0,4–2 ГГц) та мікрохвильовий діапазон (2–300 ГГц). Низькочастотні поля індукують у провідниках й біотканинах струми, що спричиняють електричну стимуляцію нервово-м’язових структур. У СВЧ-діапазоні основним механізмом є діелектричний нагрів: полярні молекули (насамперед вода) орієнтуються в змінному полі, перетворюючи енергію радіохвиль на тепло [33].

Клінічні та лабораторні дослідження демонструють, що експозиція інтенсивних СВЧ-полів (щільність потоку > 10 мВт/см²) призводить до локальних температурних підйомів на 1–2 °C протягом 15–30 хв, що може спричиняти коагуляцію білків і порушення провідності нервових волокон. На рівнях випромінювання нижче граничних (< 0,1 мВт/см²) фіксуються лише швидкі електродинамічні реакції без стійких патофізіологічних змін; проте довготривалі досліди вказують на можливість біологічного резонансу, коли окремі частотні компоненти модулюють активність іонних каналів мембран.

Оцінка ризиків базується на вимірюваннях електричної та магнітної складових полів за допомогою векторних антен і широкосмугових спектральних аналізаторів, а також на математичному моделюванні розподілу SAR (Specific Absorption Rate) в об’ємах організму. Міжнародні стандарти ICNIRP та IEEE встановлюють граничні значення: локальний SAR ≤ 2 Вт/кг і електрична напруженість поля < 61 В/м у діапазоні 900–1800 МГц [34].

Захист від ЕМП забезпечують як пасивні, так і активні методи. Пасивні включають екранування, фільтрацію провідників, заземлення та організацію відстаней від джерела до об’єкта захисту. Найпростіші екрани – суцільні металеві листи (мідь, алюміній, сталь) – забезпечують коефіцієнт загасання SE

до 80–100 дБ у СВЧ-діапазоні за відсутності щілин і дефектів. Для герметизації використовують бутилкаучукові стрічки, провідні фарби на основі срібла чи графіту, що додають цілісності провідного шару.

Поглиначі на основі феритів і композитів поєднують високу діелектричну проникність із магнітними втратами, забезпечуючи широкосмугове поглинання. Легкі магнітні пухкі матеріали знижують масу конструкції, а тонкоплівкові композити (MIM) товщиною < 1 мм використовуються для вузькосмугових застосувань (GPS, Wi-Fi), часто з нанодобавками (графен, нановолокна) для розширення смуги поглинання.

Перфоровані й тканинні екрани на основі мікросіток забезпечують баланс між гнучкістю та ефективністю: проникнення мінімізується при $d < \lambda/20$, а співвідношення площі отворів до загальної площі визначає кінцевий SE. Такі матеріали застосовують для захисту кабелів, вентиляційних люків, де потрібна повітропроникність [33].

Активні методи передбачають генерацію протифазних сигналів для деструктивної інтерференції завадового поля або використання систем моніторингу спектрального складу ЕМП із автоматичним підключенням блокаторів, фільтрів чи систем примусового заземлення у разі перевищення нормативних рівнів.

У комплексних системах поєднують зовнішні металеві корпуси, внутрішні шари феритового поглинача та тонкоплівкові резонансні елементи, що забезпечує широкосмугове загасання та точкове пригнічення перешкод у критичних діапазонах. Сучасні металополімерні композити з направленим орієнтуванням частинок додають можливість керувати анізотропією властивостей і оптимізувати масогабарити екрану [34].

Ефективна система захисту від електромагнітного випромінювання базується на принципах комплексності, багаторівневості та адаптивності, поєднуючи класифікацію впливів, методи оцінки ризиків та різнотипні захисні рішення.

ВИСНОВКИ

У ході виконання роботи було розроблено та створено прототип комп'ютерної системи “Портативна карта України”.

Система реалізована на базі мікроконтролера ESP32, яка забезпечує онлайн відображення актуальних погодних показників, якості повітря та ультрафіолетового індексу для обраного регіону, а також індикацію повітряних тривог.

Розроблена блок-схема алгоритму дозволила чітко виокремити стадії ініціалізації периферії, налаштування бездротового з’єднання, вибору режимів та циклічної обробки даних, що сприяло оптимальному розподілу обчислювальних ресурсів і гарантувало своєчасність оновлень. Застосування середовища розробки Arduino IDE та мови C++ забезпечило найкраще поєднання продуктивності з простотою кодування. Завдяки цьому вдалося реалізувати механізми динамічного парсингу JSON-повідомлень, обробки HTTP-клієнта та графічного виведення безпомилково із мінімальними затримками. Концепція енергоефективності була закладена у вибір режимів оновлення: критична інформація про поточний стан повітряних тривог опитується частіше, погода й AQI – рідше, UV-індекс – за окремим алгоритмом, що дозволяє знизити загальне енергоспоживання без втрати точності інформування.

Комплексне тестування підтвердило стабільність Wi-Fi-з’єднань, відсутність втрат пакетів під час обміну з API, коректність часового формату з NTP та точність відображення даних, надійну фіксацію вибору параметрів без випадкових спрацьовувань.

Таким чином, реалізована портативна карта України поєднує в собі гнучку архітектуру та модульність. Дана розробка може бути використана як ефективний інструмент для візуалізації та інформування про екологічну ситуацію та для оперативного реагування в разі надзвичайних подій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Жаровський Р.О., Луцик Н.С., Осухівська Г.М., Паламар А.М., Тиш Є.В. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти за спеціальністю 123 «Комп'ютерна інженерія» усіх форм навчання. Тернопіль: ТНТУ, 2024.39 с.

2. Паламар М.І., Стрембіцький М.О., Паламар А.М. Проектування комп'ютеризованих вимірювальних систем і комплексів. Навчальний посібник. Тернопіль: ТНТУ. 2019. 150 с.

3. Palamar A., Karpinski M., Palamar M., Osukhivska H., Mytnyk M. Remote Air Pollution Monitoring System Based on Internet of Things. CEUR Workshop Proceedings, 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2022), Ternopil, Ukraine, November 22–24, 2022. Vol. 3309. P. 194-204.

4. Ясінський Р.В., Осухівська Г.М., Паламар А.М., Величко Д.В. Комп'ютерна система для контролю параметрів мікроклімату теплиць на основі інтернету речей. Актуальні задачі сучасних технологій : збірник тез доповідей XI міжнародної науково-практичної конференції молодих учених та студентів (Тернопіль, 7-8 грудня 2022 року), Тернопіль: ФОП Паляниця В. А., 2022. С. 177.

5. Паламар А., Величко Д. Система моніторингу якості повітря в приміщеннях. Матеріали V Міжнародної студентської науково-технічної конференції "Природничі та гуманітарні науки. Актуальні питання" (Тернопіль, 28-29 квітня 2022 року), Тернопіль: ТНТУ. 2022. С. 138.

6. Слюз І., Жаровський Р. Принципи та основні етапи комплексного тестування комп'ютерної інформаційної системи. Матеріали X науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі системи та технології» (7-8 грудня 2022 року). Тернопіль: ТНТУ. 2022. С. 93.

7. Харченко О., Яцишин В. Розробка та керування вимогами до програмного забезпечення на основі моделі якості. Вісник ТДТУ. Тернопіль, 2009. Т. 14. №1. С. 201-207.

8. A Hybrid Architecture for Mobile Geographical Data Collection and Validation Systems. URL: https://www.researchgate.net/figure/A-Hybrid-Architecture-for-Mobile-Geographical-Data-Collection-and-Validation-Systems_fig1_272175386 (дата звернення: 10.03.2025).

9. Evaluating the Performance of Three Popular Web Mapping Libraries. URL: https://www.mdpi.com/2220-9964/9/10/563/xml (дата звернення: 12.03.2025).

10. Systems and methods for emergency communications. URL: https://patents.justia.com/patent/11425529 (дата звернення: 13.03.2025).

11. OpenWeather. URL: https://openweathermap.org/api (дата звернення: 15.03.2025).

12. Internet of Green Things with autonomous wireless wheel robots against green houses and farms. URL: https://www.academia.edu/98000289/Internet_of_Green_Things_with_autonomous_wireless_wheel_robots_against_green_houses_and_farms (дата звернення: 20.03.2025).

13. Choosing an Embedded Motherboard or Single Board Computer. URL: https://things-embedded.com/us/white-paper/choosing-an-industrial-motherboard-or-embedded-single-board-computer/ (дата звернення: 22.03.2025).

14. Led Matrix Metar Map. URL: https://www.instructables.com/LED-Matrix-Metar-Map/ (дата звернення: 23.03.2025).

15. Сервер даних JAMM. URL: https://jaam.net.ua (дата звернення: 23.03.2025).

16. SmartLight “Електронна мультимедійна DIY мапа України "JAAM". URL: https://smartlight.me/prystroyi-rozumnoho-budynku/wifi-prystroyi/mapa-jaam21 (дата звернення: 23.03.2025).

17. Prom “Карта повітряних тривог України Тривога Відбій Немає тривог”. URL: https://prom.ua/ua/p1900276227-karta-vozdushnyh-trevog.html (дата звернення: 24.03.2025).

18. DOIT Esp32 DevKit v1. URL: https://roboeq.ir/files/id/4034/name/ESP32%20MODULE.pdf/ (дата звернення: 09.04.2025).

19. RoboStore “Відладочна плата ESP-32 DevKit V1”. URL: https://robostore.com.ua/ua/otladochnye-platy/esp-moduli/esp-32/ (дата звернення: 09.04.2025).

20. OLED 4 PIN 128x64 Display Module 1.3” White Color. URL: https://www.rajguruelectronics.com/Product/4437/OLED%204%20PIN%20128x64%20display%20module%201.3%20inch%20white.pdf (дата звернення: 12.04.2025).

21. DIGITAL LED WS2812 Series Upgrade Instructions. URL: https://www.alldatasheet.com/datasheet-pdf/view/1134521/WORLDSEMI/WS2812.html (дата звернення: 12.04.2025).

22. Amazon “ws2812b LED Strip Waterproof White”. URL: https://www.amazon.se/LED-remsa-vattentät-LED-lampor-individuellt-adresserbar/dp/B0B2R17547 (дата звернення: 15.04.2025).

23. Datasheet TTP223 Capacitive Touch Sensor Module. URL: https://www.scribd.com/document/431499531/Datasheet-TTP223-Capacitive-Touch-Sensor-Module (дата звернення: 15.04.2025).

24. Ardushop “Сенсорна, ємнісна кнопка TTP223B”. URL: https://ardushop.in.ua/arduino/touch-capacitive-button-ttp223b (дата звернення: 20.04.2025).

25. Arduino/C++. URL: https://docs.arduino.cc/arduino-cloud/guides/arduino-c/ (дата звернення: 06.05.2025).

26. Blender. URL: https://www.blender.org (дата звернення: 20.05.2025).

27. Installing the ESP32 Board in Arduino IDE (Windows, Mac OS X, Linux). URL: https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/ (дата звернення: 21.05.2025).

28. МОЗ України. «Щорічний звіт про стан здоров'я населення України та епідемічну ситуацію за 2022 р.». URL: https://moz.gov.ua/storage/uploads/d8718889-1326-4060-a2fe-5f7e6359426f/Щорічн-звіт-про-стан-здоров%27я-та-епідемічну-ситуацію-за-2022-рік.pdf (дата звернення: 26.05.2025).

29. ДСНС України. «Звіт про діяльність ДСНС у 2023 р.». URL: https://dsns.gov.ua/upload/2/0/4/5/2/3/6/zvit-pro-osnovni-rezultati-diialnosti-dsns-u-2023-roci.pdf (дата звернення: 28.05.2025).

30. Національна поліція України. «Статистика дорожньо-транспортних пригод за 2023 р.». URL: https://patrolpolice.gov.ua/statystyka/ (дата звернення: 30.05.2025).

31. Держпраці України. «Річний звіт з охорони праці 2023 р.». URL: https://dsp.gov.ua/wp-content/uploads/2024/02/publichnyj-zvit-holovy-2023.pdf (дата звернення: 02.06.2025).

32. Держекоінспекція України. «Моніторинг якості атмосферного повітр'я в містах України». URL: https://openaccess.org.ua/data/blog_dwnl/Analitichna_zapiska_atmosferne_povitrya.pdf (дата звернення: 06.06.2025).

33. Державні санітарні правила та норми ДСП 3.3.6.096-2002. «Вимоги до електромагнітного випромінювання джерел побутового та виробничого призначення». URL: http://zakon.rada.gov.ua/laws/show/z0203-03 (дата звернення: 07.06.2025).

34. Український державний центр радіочастот. «Вивчення впливу електромагнітних полів на здоров'я людей.». URL: http://www.ucrf.gov.ua/storage/app/sites/1/uploaded-files/ч.2вер6.pdf (дата звернення: 07.06.2025).

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем і програмної інженерії

Кафедра комп’ютерних систем та мереж

“Затверджую”

Завідувач кафедри КС

_____ Осухівська Г.М.

“___” _____2025 р

Комп’ютерна система “Портативна карта України”

ТЕХНІЧНЕ ЗАВДАННЯ

на _9_ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп’ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студент групи СІ-42
_____ к.т.н., доц. Осухівська Г.М.	_____ Сисак О.М.
«____» _____ 2025 р.	«____» _____ 2025 р.

Тернопіль 2025

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: «Комп’ютерна система “Портативна карта України”».

Умовне позначення кваліфікаційної роботи: КС КРБ 123.234.00.00

1.2 Виконавець

Студент групи СІ-42, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Сисак О.М.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету (№4/7-53 від 27.01.2025 р.)

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 27.01.2025 р.

Плановий термін завершення виконання кваліфікаційної роботи – 23.06.2025 р.

1.5 Порядок оформлення та пред’явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ISO, ЄСКД, ЄСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи. Попередній захист кваліфікаційної роботи відбувається при готовності роботи – наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Комп’ютерна система «Портативна карта України» розробляється як компактний мобільний пристрій на базі мікроконтролера ESP32 з OLED-дисплеєм, адресною LED-стрічкою WS2812, пасивним зумером, сенсорною кнопкою та акумулятором з модулями TP4056 і MT3608 для заряджання й підвищення напруги. Система призначена для візуалізації та моніторингу актуальних даних про обстановку на території України в режимі реального часу. Пристрій забезпечує підключення до Wi-Fi та періодичне оновлення даних про повітряні тривоги через API Alerts.in.ua, а також погодні умови, температуру, швидкість вітру, рівень ультрафіолету та якість повітря через API OpenWeatherMap. Інтерфейс користувача реалізовано у вигляді трьох рядків тексту, індикацією статусів світлодіодною стрічкою та звуковим сигналом тривоги.

2.2 Мета створення системи

Мета створення системи полягає в розробці портативного інструменту для інформування населення про критичні події й метеорологічні умови в

Україні. Завданнями є забезпечення постійного з’єднання з мережею Wi-Fi з автоматичним оновленням даних, реалізація інтуїтивного меню вибору області та режиму роботи за допомогою сенсорної кнопки, а також забезпечення чіткого та швидкого відображення інформації на OLED та LED-стрічці.

2.3 Характеристика об'єкту

Розроблюваний пристрій призначений для використання в мобільних ситуаціях, де потрібна компактність, автономність і швидкий доступ до актуальної інформації про безпекову та метеорологічну обстановку на території України.

3 Вимоги до системи

3.1 Вимоги до системи в цілому

3.1.1 Вимоги до структури та функціонування системи

Система «Портативна карта України» повинна складатися з програмного забезпечення, що виконується на мікроконтролері ESP32 з бібліотеками FastLED, WiFiClient, HTTPClient та U8g2, а також з апаратних компонентів, до яких належать OLED I2C 128×64, адресна LED-стрічка WS2812 (26 світлодіодів), пасивний зумер, сенсорна кнопка TTP223, акумуляторна батарея 18650, модулі заряджання TP4056 та підвищення напруги MT3608.

3.1.2 Вимоги до способів та засобів зв’язку між компонентами системи

Для забезпечення узгодженої роботи всіх модулів система використовує безпровідне з’єднання ESP32 з мережею Wi-Fi за стандартом IEEE 802.11 b/g/n та інтерфейс I2C для обміну даними між мікроконтролером і OLED-дисплеєм.

3.1.3 Вимоги до режимів функціонування системи

Пристрій повинен підтримувати режим відображення даних у реальному часі з API, у якому він підключається до серверів Alerts.in.ua та OpenWeatherMap, отримує поточні значення і виводить їх на OLED та LED-стрічку. Періодичне оновлення інформації має виконуватися з інтервалом, при цьому при успішному отриманні нових даних від API всі попередні значення замінюються.

3.1.4 Вимоги по діагностуванню системи

Для перевірки працездатності пристрою необхідно при увімкненні виконати послідовність дій: обрати рівень яскравості LED-стрічки, встановити з’єднання з мережею Wi-Fi, обрати область України за допомогою сенсорної кнопки, а потім по черзі переключитися в кожен із режимів («Тривога», «Погода», «Температура», «Якість повітря», «Швидкість вітру», «УФ») і переконатися у відображенні відповідних даних на OLED-дисплеї та коректній індикації кольору на LED-стрічці. Успішне проходження кожного тесту підтверджується візуально. За наявності некоректного відображення або відмови апаратного компонента слід повторити вмикання пристрою та перевірити з’єднання і підключення відповідних модулів.

3.1.5 Перспективи розвитку, проектування системи

Перспективи розвитку та модернізація системи Перспективи розвитку та модернізація системи У подальшому передбачається розширення функціональності через додавання підтримки додаткових API, тобто нових режимів. Також можливо покращення апаратної частини шляхом використання більш енергоефективних модулів та підвищення ємності акумулятора.

3.2 Показники призначення

Система повинна передбачати можливість масштабування програмного забезпечення при зміні конфігурації API або розширенні числа режимів. Можливості масштабування забезпечуються належною архітектурою коду та конфігурованими параметрами завантаження.

3.2.1 Вимоги до надійності

Пристрій має забезпечувати безперебійну роботу протягом не менше ніж 4 години автономно від акумулятора та автоматично виконувати повторну спробу підключення до Wi-Fi у разі втрати зв’язку до трьох разів із затримкою 30 с між спробами. Помилки отримання даних від API не повинні призводити до перезавантаження пристрою, а до відображенням останніх коректних значень.

3.3 Вимоги до безпеки

Зовнішні компоненти, що перебувають під напругою, мають бути захищені від випадкового дотику шляхом використання ізоляції. Електроживлення пристрою має мати захисне автоматичне відключення при коротких замиканнях і перевантаженнях у колах навантаження. Загальні вимоги пожежної безпеки повинні відповідати нормам побутового електрообладнання, при цьому матеріали корпусу не повинні виділяти отруйні гази під час займання.

3.3.1 Вимоги до експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи

Для зберігання та експлуатації пристрою рекомендовано підтримувати температуру від $+0\,°C$ до $+35\,°C$, відносну вологість повітря $30 – 80\,\%$ та атмосферний тиск 760 ± 25 мм рт. ст. Періодичне технічне обслуговування

апаратних компонентів слід проводити не рідше ніж раз на рік із перевіркою акумулятора, сенсорної кнопки та цілісності пайок.

3.4 Вимоги до захисту інформації від несанкціонованого доступу

Пристрій не передбачає спеціальних мережевих інтерфейсів для віддаленого доступу до конфігураційних файлів або серійного монітора, тому ризики несанкціонованого доступу мінімізуються відсутністю відповідних каналів комунікації. Всі налаштування зберігаються у простому текстовому файлі на внутрішній пам’яті модуля ESP32, доступ до якого можливий лише через фізичне підключення до пристрою через USB та середовище Arduino IDE.

3.4.1 Вимоги по збереженню інформації при аваріях

При аварійних ситуаціях система повинна перезавантажитися та успішно увімкнутися. Це забезпечить відновлення після перезавантаження або виходу із-за втрати живлення.

3.4.2 Вимоги по стандартизації і уніфікації

Конструкція пристрою має відповідати ергономічним вимогам портативних пристроїв і базовим стандартам безпеки електронних пристроїв. Використані компоненти ESP32, OLED-дисплей та WS2812 повинні мати обов’язкові сертифікати.

3.4.3 Вимоги до функцій (завдань), що виконуються системою:

Система повинна забезпечувати зручний інтерфейс користувача для швидкого перегляду поточних даних, зберігати останні отримані значення у внутрішній пам’яті, виконувати оперативний пошук інформації за регіонами та режимами, а також гарантувати високу швидкодію обробки та відображення даних.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ

Комплект документації повинен складатись з:

– пояснювальної записки;

– графічного матеріалу:

а) Структурна схема системи.

б) Схема електрична принципова.

в) Блок-схема алгоритму роботи системи

г) Креслення кришки

ґ) Загальний вигляд комп’ютерної системи “Портативна карта України”

*Примітка: У комплект документації можуть вноситися зміни та доповнення в процесі розробки.

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ етапу	Назва етапу виконання кваліфікаційної роботи	Термін виконання
1	Розробка технічного завдання	27.01.2025р. – 02.02.2025р.
2	Аналіз технічного завдання	15.02.2025р. – 27.03.2025р.
3	Проєктна частина	06.04.2025р. – 02.05.2025р.
4	Практична частина	04.05.2025р. – 25.05.2025р.
5	Безпека життєдіяльності, основи охорони праці	26.05.2025р. – 02.06.2025р.
6	Оформлення пояснювальної записки і графічного матеріалу	03.06.2025р. – 10.06.2025р.
7	Перевірка на академічний плагіат, перевірка керівником та консультантами	9.06.2025р. – 15.06.2025р.
8	Попередній захист кваліфікаційної роботи бакалавра	16.06.2025р. – 22.06.2025р.
9	Захист кваліфікаційної роботи бакалавра	27.06.2025р

6 Додаткові умови виконання кваліфікаційної роботи

Під час виконання кваліфікаційної роботи у дане технічне завдання можуть вноситися зміни та доповнення.

Додаток Б

Перелік елементів

Поз. позначення	*Найменування*	*Кіл.*	*Примітка*
	Акумуляторні відсіки		
BT1	BHC-18650-1A	1	
	Дисплеї		
H1	OLED 128×64 I2C 1,3″	1	
	Світлодіодні стрічки		
LED1	WS2812B	1	
	Резистори		
R1	CFR0W4J0360A50	1	
	Модулі		
S1	Давач дотику TTP223	1	
SG1	Пасивний зумер TMB12A05	1	
SW1	Перемикач тумблер RA11131121	1	
U1	ESP-32 DevKit V1	1	
U2	Модуль MT3608	1	
U3	Модуль зарядки TP4056	1	

Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*	*КС КРБ 123.234.00.00 ПЕ*		
Розроб.		*Сисак О.М.*			*Комп'ютерна система "Портативна карта України" Перелік елементів*	*Літ.*	*Арк.* / *Аркушів*
Перевір.		*Осуховська Г.М.*					
Реценз.		*Фриз М.Є.*				*ТНТУ, каф. КС, гр. СІ-42*	
Н. Контр.		*Тиш Є.В.*					
Затверд.		*Осуховська Г.М.*					

Додаток В

Лістинг коду програми

```
#include <Arduino.h>
#include <WiFi.h>
#include <WebServer.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <Wire.h>
#include <U8g2lib.h>
#include <FastLED.h>
#include <time.h>

const char* OWM_API_KEY   = "5052a733bde7573036a91a81f40a5753";
const char* UA_API_URL    =
"https://api.alerts.in.ua/v1/alerts/active.json?token=f74afc00ed
b18aa625aba79abc0471519ba54e5cab2203";

WebServer server(80);
bool wifiConfigured = false;
String userSSID;
String userPASS;

#define PIN_OLED_SDA   21
#define PIN_OLED_SCL   22
#define PIN_LEDS       13
#define PIN_BUZZER     12
#define PIN_TOUCH      15

U8G2_SSD1306_128X64_NONAME_F_SW_I2C u8g2(
  U8G2_R0,
  PIN_OLED_SCL,
  PIN_OLED_SDA,
  U8X8_PIN_NONE
);

#define NUM_LEDS 26
CRGB leds[NUM_LEDS];

const uint32_t INTERVAL_ALARM   = 30UL * 1000;
const uint32_t INTERVAL_WEATHER = 30UL * 60 * 1000;
const uint32_t INTERVAL_DISPLAY = 50;

enum AppState { STARTUP, CONFIG_WIFI, CONNECTING_WIFI,
SELECT_REGION, SELECT_MODE, RUN_MODE };
AppState appState = STARTUP;

enum Mode { M_ALARM, M_WEATHER, M_TEMP, M_AQI, M_WIND, M_UV,
M_COUNT };
const char* modeNames[M_COUNT] = {
```

```
  "Air Raid Alarm", "Weather", "Temperature",
  "Air Quality",   "Wind Speed", "UV Index"
};

struct Region {
  const char* nameShort;
  const char* slug;
  double      lat, lon;
};
Region regions[NUM_LEDS] = {
  {"Zakarp", "zakarpatska",       48.6208, 22.2879},
  {"Iv-Fr",  "ivano-frankivska", 48.9215, 24.7097},
  {"Ternop", "ternopilska",       49.5535, 25.5948},
  {"Lviv",   "lvivska",           49.8397, 24.0297},
  {"Volyn",  "volynska",          50.7472, 25.3250},
  {"Rivn",   "rivnenska",         50.6199, 26.2516},
  {"Zhytom", "zhytomyrska",       50.2547, 28.6587},
  {"Kyiv",   "kyivska",           50.4501, 30.5234},
  {"Cherni", "chernihivska",      51.5055, 31.2849},
  {"Sumy",   "sumska",            50.9077, 34.7981},
  {"Khark",  "kharkivska",        49.9935, 36.2304},
  {"Luh",    "luhanska",          48.5740, 39.3078},
  {"Don",    "donetska",          48.0159, 37.8029},
  {"Zap",    "zaporizka",         47.8388, 35.1396},
  {"Khers",  "khersonska",        46.6550, 32.6178},
  {"Crimea", "crimea",            44.6167, 33.5253},
  {"Odesa",  "odeska",            46.4825, 30.7233},
  {"Odesa",  "odeska",            46.4825, 30.7233},
  {"Mykol",  "mykolaivska",       46.9750, 31.9946},
  {"Dnipro", "dnipropetrovska",   48.4647, 35.0462},
  {"Polt",   "poltavska",         49.5883, 34.5514},
  {"Cherk",  "cherkaska",         49.4444, 32.0584},
  {"Kirov",  "kirovohradska",     48.5078, 32.2623},
  {"Vinny",  "vinnytska",         49.2328, 28.4800},
  {"Khmel",  "khmelnytska",       49.4230, 26.9838},
  {"Chern",  "chernivetska",      48.2915, 25.9356}
};

uint8_t idxRegion = 0;
uint8_t idxMode   = 0;

uint32_t tLastAlarm   = 0;
uint32_t tLastWeather = 0;

float    temps[NUM_LEDS];
float    winds[NUM_LEDS];
float    uvs[NUM_LEDS];
uint8_t  aqis[NUM_LEDS];
String   weatherDescs[NUM_LEDS];

bool     alarmActive       = false;
time_t   alarmStart        = 0;
time_t   alarmEnd          = 0;
```

```
bool     alarmTempFetched  = false;

struct UidMap { int uid; int ledIndex; };
const UidMap uidToLed[] = {
  {11,  0},  {13,  1},  {21,  2},  {27,  3},
  {8,   4},  {5,   5},  {10,  6},  {14,  7},
  {25,  8},  {20,  9},  {22, 10},  {16, 11},
  {28, 12},  {12, 13},  {29, 14},  {23, 15},
  {9,   16}, {19,  17}, {15,  18}, {17,  19},
  {24,  20}, {18,  21}, {4,   22},  {3,   23},
  {26,  24}, {-1,  25}
};
const int UID_TO_LED_COUNT = sizeof(uidToLed) / sizeof(UidMap);

const uint8_t brightnessLevels[3] = { 20, 128, 255 };
uint8_t idxBrightness = 1;
bool brightnessSelected = false;

CRGB colorForTemp(float t) {
  if      (t <=  -20) return CRGB::Blue;
  else if (t <=  -15) return CRGB::BlueViolet;
  else if (t <=  -10) return CRGB::Cyan;
  else if (t <=   -5) return CRGB::Teal;
  else if (t <=    0) return CRGB::Green;
  else if (t <=    5) return CRGB::YellowGreen;
  else if (t <=   10) return CRGB::Yellow;
  else if (t <=   15) return CRGB::Orange;
  else if (t <=   20) return CRGB::Red;
  else if (t <=   25) return CRGB::Magenta;
  else                return CRGB::Purple;
}

CRGB colorForWind(float w) {
  if      (w <=   1) return CRGB::White;
  else if (w <=   5) return CRGB::Cyan;
  else if (w <=  10) return CRGB::Blue;
  else if (w <=  15) return CRGB::Magenta;
  else               return CRGB::Red;
}

CRGB colorForAQI(uint8_t aqi) {
  switch (aqi) {
    case 1: return CRGB::Green;
    case 2: return CRGB::Yellow;
    case 3: return CRGB::Orange;
    case 4: return CRGB::Red;
    case 5: return CRGB::Magenta;
    default: return CRGB::Gray;
  }
}

CRGB colorForUV(float uv) {
  if      (uv <=   2) return CRGB::White;
```

```
  else if (uv <=   5) return CRGB::Blue;
  else if (uv <=   7) return CRGB::Magenta;
  else if (uv <=  10) return CRGB::Red;
  else                return CRGB::Purple;
}

CRGB colorForWeather(const String &desc) {
  String d = desc;
  d.toLowerCase();
  if (d.indexOf("clear") >= 0)   return CRGB::Yellow;
  if (d.indexOf("cloud") >= 0)   return CRGB::White;
  if (d.indexOf("rain")  >= 0)   return CRGB::Blue;
  if (d.indexOf("snow")  >= 0)   return CRGB::White;
  if (d.indexOf("fog")   >= 0)   return CRGB::Magenta;
  if (d.indexOf("wind")  >= 0)   return CRGB::Cyan;
  return CRGB::Green;
}

String currentTime() {
  struct tm tm;
  if (!getLocalTime(&tm)) return "--:--:--";
  char buf[16];
  strftime(buf, sizeof(buf), "%H:%M:%S", &tm);
  return String(buf);
}

String scrollBuf;
uint16_t scrollPos = 0;

void updateScroll(const String &txt) {
  scrollBuf = txt + "   ";
  scrollPos = 0;
}

String getScrollOne() {
  if (scrollBuf.length() == 0) return "";
  String s = scrollBuf.substring(scrollPos) +
scrollBuf.substring(0, scrollPos);
  scrollPos = (scrollPos + 1) % scrollBuf.length();
  return s;
}

void drawOLED(const String& line2, const String& line3) {
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_6x12_tr);
  u8g2.setCursor(2, 12);
  u8g2.print(currentTime());
  u8g2.print("  ");
  u8g2.print(modeNames[idxMode]);
  u8g2.setCursor(2, 28);
  u8g2.print(line2);
  u8g2.setCursor(2, 44);
  u8g2.print(line3);
```

```
  u8g2.sendBuffer();
}

uint32_t lastTouchMillis = 0;
uint8_t readTouchDigital() {
  static uint32_t t0 = 0;
  int raw = digitalRead(PIN_TOUCH);
  if (raw == HIGH) {
    if (!t0) t0 = millis();
  } else {
    if (t0) {
      uint32_t dt = millis() - t0;
      t0 = 0;
      if (millis() - lastTouchMillis < 300) return 0;
      lastTouchMillis = millis();
      if (dt <= 800) {
        Serial.println("Touch: SHORT");
        return 1;
      } else {
        Serial.println("Touch: LONG");
        return 2;
      }
    }
  }
  return 0;
}

void playTone(uint8_t pin, unsigned int frequency, unsigned int
duration) {
  pinMode(pin, OUTPUT);
  unsigned long t0 = millis();
  unsigned int period = 1000000UL / frequency;
  while (millis() - t0 < duration) {
    digitalWrite(pin, HIGH);
    delayMicroseconds(period / 2);
    digitalWrite(pin, LOW);
    delayMicroseconds(period / 2);
  }
}
void stopTone(uint8_t pin) {
  digitalWrite(pin, LOW);
}

void handleRoot() {
  String page = "<!DOCTYPE html><html><head><meta charset='utf-
8'>";
  page += "<meta name='viewport' content='width=device-width,
initial-scale=1'>";
  page += "<title>Wi-Fi Config</title>";
  page += "<style>body { display: flex; justify-content: center;
align-items: center; height: 100vh; ";
  page += "margin: 0; font-family: Arial, sans-serif; } ";
  page += "form { text-align: center; } input { margin: 5px;
```

```
padding: 5px; }</style>";
  page += "</head><body><form action='/save' method='POST'>";
  page += "<h3>Enter Wi-Fi SSID and Password:</h3>";
  page += "SSID:<br><input type='text' name='ssid'><br>";
  page += "Password:<br><input type='password'
name='pass'><br><br>";
  page += "<input type='submit' value='Save'>";
  page += "</form></body></html>";
  server.send(200, "text/html", page);
}

void handleSave() {
  if (server.hasArg("ssid") && server.hasArg("pass")) {
    userSSID = server.arg("ssid");
    userPASS = server.arg("pass");
    server.send(200, "text/html", "<!DOCTYPE
html><html><body><h3>Saved. Connecting...</h3></body></html>");
    delay(1000);
    wifiConfigured = true;
  } else {
    server.send(400, "text/plain", "Bad Request");
  }
}

void startWiFiConfig() {
  WiFi.mode(WIFI_AP);
  WiFi.softAP("ESP32_Config");
  server.on("/", handleRoot);
  server.on("/save", HTTP_POST, handleSave);
  server.begin();
  u8g2.clearBuffer();
  u8g2.setCursor(2, 12);
  u8g2.print("AP Mode: ESP32_Config");
  u8g2.setCursor(2, 28);
  u8g2.print("Open browser to:");
  u8g2.setCursor(2, 44);
  u8g2.print("http://192.168.4.1");
  u8g2.sendBuffer();
}

void updateAlarm() {
  HTTPClient http;
  http.begin(UA_API_URL);
  int httpCode = http.GET();
  if (httpCode == 200) {
    String payload = http.getString();
    for (int i = 0; i < NUM_LEDS; i++) {
      leds[i] = CRGB::Green;
    }
    for (int k = 0; k < UID_TO_LED_COUNT; k++) {
      int uid = uidToLed[k].uid;
      int ledIndex = uidToLed[k].ledIndex;
      if (uid < 0 || ledIndex < 0 || ledIndex >= NUM_LEDS)
```

```
continue;
      String s1 = "\"location_oblast_uid\":" + String(uid) +
",";
      String s2 = "\"location_oblast_uid\":" + String(uid) +
"}";
      if (payload.indexOf(s1) != -1 || payload.indexOf(s2) != -
1) {
        leds[ledIndex] = CRGB::Red;
      }
    }
    int selUid = uidToLed[idxRegion].uid;
    String s1 = "\"location_oblast_uid\":" + String(selUid) +
",";
    String s2 = "\"location_oblast_uid\":" + String(selUid) +
"}";
    bool act = (payload.indexOf(s1) != -1 || payload.indexOf(s2)
!= -1);
    if (act && !alarmActive) {
      alarmActive = true;
      alarmStart  = time(nullptr);
      playTone(PIN_BUZZER, 1000, 500);
    } else if (!act && alarmActive) {
      alarmActive = false;
      alarmEnd    = time(nullptr);
      playTone(PIN_BUZZER, 600, 500);
    }
  }
  http.end();
  FastLED.show();
}

float fetchUV_OpenMeteo(double lat, double lon) {
  char url[256];
  snprintf(url, sizeof(url),
    "https://air-quality-api.open-meteo.com/v1/air-quality?"

"latitude=%.4f&longitude=%.4f&hourly=uv_index&timezone=Europe%%2
FKyiv",
    lat, lon
  );
  HTTPClient http;
  http.begin(url);
  int httpCode = http.GET();
  float uvValue = 0.0f;
  if (httpCode == 200) {
    String payload = http.getString();
    DynamicJsonDocument doc(16 * 1024);
    if (!deserializeJson(doc, payload)) {
      JsonArray times   = doc["hourly"]["time"].as<JsonArray>();
      JsonArray uvs_arr =
doc["hourly"]["uv_index"].as<JsonArray>();
      struct tm tmInfo;
      if (getLocalTime(&tmInfo)) {
```

```
        char buf[20];
        strftime(buf, sizeof(buf), "%Y-%m-%dT%H:00", &tmInfo);
        String nowStr = String(buf);
        for (size_t i = 0; i < times.size(); i++) {
          if (String(times[i].as<const char*>()) == nowStr) {
            uvValue = uvs_arr[i].as<float>();
            break;
          }
        }
      }
    }
  }
  http.end();
  return uvValue;
}

void updateWeatherForOne(uint8_t regionIndex) {
  char urlWeather[256];
  snprintf(urlWeather, sizeof(urlWeather),
    "http://api.openweathermap.org/data/2.5/weather?"
    "lat=%.4f&lon=%.4f&units=metric&appid=%s",
    regions[regionIndex].lat,
    regions[regionIndex].lon,
    OWM_API_KEY
  );
  HTTPClient http;
  http.begin(urlWeather);
  int httpCode = http.GET();
  if (httpCode == 200) {
    String payload = http.getString();
    DynamicJsonDocument d(16 * 1024);
    DeserializationError err = deserializeJson(d, payload);
    if (!err) {
      temps[regionIndex]        = d["main"]["temp"].as<float>();
      winds[regionIndex]        =
d["wind"]["speed"].as<float>();
      weatherDescs[regionIndex] =
d["weather"][0]["description"].as<const char*>();
    } else {
      temps[regionIndex]        = 0;
      winds[regionIndex]        = 0;
      weatherDescs[regionIndex] = "";
    }
  } else {
    temps[regionIndex]        = 0;
    winds[regionIndex]        = 0;
    weatherDescs[regionIndex] = "";
  }
  http.end();

  char urlAQI[256];
  snprintf(urlAQI, sizeof(urlAQI),
    "http://api.openweathermap.org/data/2.5/air_pollution?"
```

```
    "lat=%.4f&lon=%.4f&appid=%s",
    regions[regionIndex].lat,
    regions[regionIndex].lon,
    OWM_API_KEY
  );
  http.begin(urlAQI);
  httpCode = http.GET();
  if (httpCode == 200) {
    String payload2 = http.getString();
    DynamicJsonDocument d2(4 * 1024);
    DeserializationError err2 = deserializeJson(d2, payload2);
    if (!err2) {
      aqis[regionIndex] =
d2["list"][0]["main"]["aqi"].as<uint8_t>();
    } else {
      aqis[regionIndex] = 0;
    }
  } else {
    aqis[regionIndex] = 0;
  }
  http.end();

  Serial.print("Weather for ");
  Serial.print(regions[regionIndex].nameShort);
  Serial.print(": T=");
  Serial.print(temps[regionIndex], 1);
  Serial.print("C, W=");
  Serial.print(winds[regionIndex], 1);
  Serial.print("m/s, AQI=");
  Serial.println(aqis[regionIndex]);
}

void updateUVAllRegions() {
  for (uint8_t i = 0; i < NUM_LEDS; i++) {
    uvs[i] = fetchUV_OpenMeteo(regions[i].lat, regions[i].lon);
    Serial.print("UV for ");
    Serial.print(regions[i].nameShort);
    Serial.print(" = ");
    Serial.println(uvs[i], 1);
    delay(100);
  }
}

void updateWeatherAllRegions() {
  for (uint8_t i = 0; i < NUM_LEDS; i++) {
    updateWeatherForOne(i);
    delay(100);
  }
  String buf = "";
  for (uint8_t i = 0; i < NUM_LEDS; i++) {
    String val;
    switch (idxMode) {
      case M_WEATHER: {
```

```
        String desc = weatherDescs[i];
        if (desc.length() > 16) desc = desc.substring(0, 16) +
"..";
        val = desc;
        break;
      }
      case M_TEMP:
        val = String((int)temps[i]) + "C";
        break;
      case M_AQI:
        val = String(aqis[i]);
        break;
      case M_WIND:
        val = String((int)winds[i]) + "m/s";
        break;
      default:
        val = "";
        break;
    }
    buf += String(regions[i].nameShort) + ":" + val + "   ";
  }
  updateScroll(buf);
}

void updateScrollUV() {
  String buf = "";
  for (uint8_t i = 0; i < NUM_LEDS; i++) {
    String val = (uvs[i] > 0) ? String((int)round(uvs[i])) :
String("N/A");
    buf += String(regions[i].nameShort) + ":" + val + "   ";
  }
  updateScroll(buf);
}

void renderModeLED() {
  for (int i = 0; i < NUM_LEDS; i++) {
    switch (idxMode) {
      case M_ALARM:
        break;
      case M_WEATHER:
        leds[i] = colorForWeather(weatherDescs[i]);
        break;
      case M_TEMP:
        leds[i] = colorForTemp(temps[i]);
        break;
      case M_AQI:
        leds[i] = colorForAQI(aqis[i]);
        break;
      case M_WIND:
        leds[i] = colorForWind(winds[i]);
        break;
      case M_UV:
        leds[i] = colorForUV(uvs[i]);
```

```
        break;
      default:
        leds[i] = CRGB::Black;
        break;
    }
  }
  FastLED.show();
}

void setup() {
  Serial.begin(115200);
  Wire.begin(PIN_OLED_SDA, PIN_OLED_SCL);
  u8g2.begin();
  u8g2.setFont(u8g2_font_6x12_tr);
  FastLED.addLeds<WS2812, PIN_LEDS, GRB>(leds, NUM_LEDS);
  FastLED.clear();
  FastLED.show();
  pinMode(PIN_BUZZER, OUTPUT);
  pinMode(PIN_TOUCH, INPUT_PULLDOWN);
  configTime(3 * 3600, 0, "pool.ntp.org", "time.nist.gov");

  for (uint8_t i = 0; i < NUM_LEDS; i++) {
    temps[i]        = 0;
    winds[i]        = 0;
    uvs[i]          = 0;
    aqis[i]         = 0;
    weatherDescs[i] = "";
  }

  while (!brightnessSelected) {
    u8g2.clearBuffer();
    u8g2.setFont(u8g2_font_6x12_tr);
    u8g2.setCursor(2, 12);
    u8g2.print("Select Brightness:");
    u8g2.setCursor(2, 28);
    u8g2.print(String(brightnessLevels[idxBrightness]));
    u8g2.setCursor(2, 44);
    u8g2.print("(Tap=Next, Hold=Select)");
    u8g2.sendBuffer();

    uint8_t t = readTouchDigital();
    if (t == 1) {
      idxBrightness = (idxBrightness + 1) % 3;
      delay(200);
    } else if (t == 2) {
      brightnessSelected = true;
    }
    delay(50);
  }

  FastLED.setBrightness(brightnessLevels[idxBrightness]);
  FastLED.clear();
  FastLED.show();
```

```
  const uint8_t half = NUM_LEDS / 2;
  for (int i = 0; i < NUM_LEDS; i++) {
    leds[i] = (i < half) ? CRGB::Blue : CRGB::Yellow;
    FastLED.show();
    delay(50);
  }
  int melody[] = {262, 294, 330, 349, 392, 440, 494, 523, 587,
659};
  for (auto n : melody) {
    playTone(PIN_BUZZER, n, 200);
    delay(50);
  }
  stopTone(PIN_BUZZER);

  appState = CONFIG_WIFI;
  startWiFiConfig();
}

void loop() {
  switch (appState) {
    case CONFIG_WIFI:
      server.handleClient();
      if (wifiConfigured) {
        appState = CONNECTING_WIFI;
      }
      break;

    case CONNECTING_WIFI: {
      u8g2.clearBuffer();
      u8g2.setCursor(2, 12);
      u8g2.print("Connecting to Wi-Fi:");
      u8g2.setCursor(2, 28);
      u8g2.print(userSSID);
      u8g2.sendBuffer();

      WiFi.mode(WIFI_STA);
      WiFi.begin(userSSID.c_str(), userPASS.c_str());
      uint32_t start = millis();
      while (WiFi.status() != WL_CONNECTED && millis() - start <
15000) {
        delay(100);
      }
      if (WiFi.status() == WL_CONNECTED) {
        u8g2.clearBuffer();
        u8g2.setCursor(2, 12);
        u8g2.print("Connected to:");
        u8g2.setCursor(2, 28);
        u8g2.print(userSSID);
        u8g2.sendBuffer();
        delay(1000);
        WiFi.softAPdisconnect(true);
        server.stop();
```

```
      FastLED.clear();
      FastLED.show();
      appState = SELECT_REGION;
    } else {
      appState = CONFIG_WIFI;
      startWiFiConfig();
    }
    break;
  }

  case SELECT_REGION: {
    u8g2.clearBuffer();
    u8g2.setCursor(2, 12);
    u8g2.print("Select Region:");
    u8g2.setCursor(2, 28);
    u8g2.print(regions[idxRegion].nameShort);
    u8g2.sendBuffer();

    uint8_t t = readTouchDigital();
    if (t == 1) {
      idxRegion = (idxRegion + 1) % NUM_LEDS;
    } else if (t == 2) {
      appState = SELECT_MODE;
    }
    break;
  }

  case SELECT_MODE: {
    u8g2.clearBuffer();
    u8g2.setCursor(2, 12);
    u8g2.print("Select Mode:");
    u8g2.setCursor(2, 28);
    u8g2.print(modeNames[idxMode]);
    u8g2.sendBuffer();

    uint8_t t = readTouchDigital();
    if (t == 1) {
      idxMode = (idxMode + 1) % M_COUNT;
    } else if (t == 2) {
      appState     = RUN_MODE;
      tLastAlarm   = 0;
      tLastWeather = 0;
      scrollBuf    = "";
      alarmTempFetched = false;
      FastLED.clear();
      FastLED.show();
      if (idxMode == M_ALARM) {
        updateAlarm();
      } else if (idxMode == M_UV) {
        updateUVAllRegions();
        updateScrollUV();
      } else {
        updateWeatherAllRegions();
```

```
        }
      }
      break;
    }

    case RUN_MODE: {
      uint32_t now = millis();

      if (idxMode == M_ALARM && now - tLastAlarm >
INTERVAL_ALARM) {
        updateAlarm();
        tLastAlarm = now;
      }

      if (idxMode != M_ALARM && now - tLastWeather >
INTERVAL_WEATHER) {
        if (idxMode == M_UV) {
          updateUVAllRegions();
          updateScrollUV();
        } else {
          updateWeatherAllRegions();
        }
        tLastWeather = now;
      }

      renderModeLED();

      String line2, line3;
      if (idxMode == M_ALARM) {
        if (!alarmTempFetched) {
          updateWeatherForOne(idxRegion);
          alarmTempFetched = true;
        }
        line2 = String(regions[idxRegion].nameShort) +
                " Temp=" + String((int)temps[idxRegion]) + "C";
        line3 = "All Clear";
      } else {
        switch (idxMode) {
          case M_WEATHER: {
            String desc = weatherDescs[idxRegion];
            if (desc.length() > 16) desc = desc.substring(0, 16)
+ "..";
            line2 = String(regions[idxRegion].nameShort) +
                    ": " + desc;
            break;
          }
          case M_TEMP:
            line2 = String(regions[idxRegion].nameShort) +
                    ": " + String((int)temps[idxRegion]) + "C";
            break;
          case M_AQI:
            line2 = String(regions[idxRegion].nameShort) +
                    ": AQI=" + String(aqis[idxRegion]);
```

```
            break;
          case M_WIND:
            line2 = String(regions[idxRegion].nameShort) +
                    ": " + String((int)winds[idxRegion]) +
"m/s";
            break;
          case M_UV:
            line2 = String(regions[idxRegion].nameShort) +
                    ": UV=" + ((uvs[idxRegion] > 0)
                               ?
String((int)round(uvs[idxRegion]))
                               : String("N/A"));
            break;
          default:
            line2 = "";
            break;
        }
        line3 = getScrollOne();
      }
      drawOLED(line2, line3);

      if (readTouchDigital() == 2) {
        const uint8_t half = NUM_LEDS / 2;
        for (int i = 0; i < NUM_LEDS; i++) {
          leds[i] = (i < half) ? CRGB::Blue : CRGB::Yellow;
        }
        FastLED.show();
        appState = SELECT_MODE;
      }

      delay(INTERVAL_DISPLAY);
      break;
    }

    default:
      break;
  }
}
```