

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ
АВТОМАТИЗОВАНОГО ЗБИРАННЯ ДАНИХ З ІНТЕРНЕТ
SOFTWARE DEVELOPMENT FOR AUTOMATED DATA
COLLECTION IN THE INTERNET

Виконав: студент(ка) 4 курсу, групи КНТ-227

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

Осипович О.О.

(прізвище та ініціали)

Керівник Гладкова О.М.

(прізвище та ініціали)

Рецензент Терещенко Е.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Осипович Олександрі Олегівни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмного забезпечення для автоматизованого збирання даних з Інтернет. Software Development for Automated Data Collection in the Internet

керівник проєкту (роботи) Гладкова Ольга Миколаївна, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 5 травня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області та постановка завдання. 2. Вибір інструментарію розробки. 3. Проєктування системи. 4. Керівництво з використання програмного продукту.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГЛАДКОВА О.М., доцент		
Нормоконтролер	Дейнега Л.Ю., ст.викладач		

7. Дата видачі завдання “ 28 ” лютого 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Дослідження особливостей розробки системи. Вибір інструментарію розробки.	4 тиждень	Розділ 2
4	Проектування системи.	5–6 тиждень	Розділ 3
5	Оформлення пояснювальної записки та документів до неї.	7–8 тижні	Розділ 4, Додатки
6	Нормоконтроль та рецензування.	9 тиждень	
7	Захист роботи.	10 тиждень	
8			
9			
10			
11			
12			

Студент(ка)

(підпис) Осипович О.О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Гладкова О.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 58 с., 7 табл., 25 рис., 2 дод., 21 джерел.

ВБУДОВАНА СИСТЕМА, ДАНІ З ІНТЕРНЕТ, ПОГОДА, WI-FI, ESP32.

Об'єкт дослідження – програмні засоби для отримання даних з Інтернет.

Предмет дослідження – вбудована система для збирання та відображення даних з Інтернет.

Мета роботи – розробити програмно-апаратний комплекс, який дозволить збирати дані про погоду в Інтернет та відображати їх користувачу у реальному часі.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- провести дослідження існуючих інструментів для побудови засобу моніторингу погоди;
- розробити архітектуру системи;
- спроектувати апаратну частину системи;
- розробити програмне забезпечення.

Матеріали, методи та технічні засоби: об'єктно орієнтоване програмування, Arduino(c/c++) мова програмування, апаратно програмна платформа NodeMCU ESP32, технологія REST API.

Результати. Розроблено програмно-апаратний комплекс для збирання даних про погоду з Інтернет, що дозволяє відображати дані про погоду в заданій області не витрачаючи час на пошуки та ресурси для встановлення та налаштування персональної метеостанції.

Галузь використання – вбудовані системи, Інтернет речі.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 58 pages, 7 tables, 25 figures, 2 appendixes, 21 sources.

EMBEDDED SYSTEM, INTERNET DATA, WEATHER, WI-FI, ESP32.

The object of study is software for obtaining data from the Internet.

The subject of the research is the embedded system for collecting and displaying data from the Internet.

The purpose of the work is to develop a software and hardware complex that will allow to collect weather data on the Internet and display it to the user in real time.

To solve this goal it is necessary to solve the following tasks:

- conduct research on existing tools for building a weather monitoring tool;
- develop system architecture;
- design the hardware of the system;
- develop software.

Materials, methods and technical means: object-oriented programming, Arduino (c / c++) programming language, hardware and software platform NodeMCU ESP32, REST API technology.

Results. A software and hardware complex for collecting weather data from the Internet has been created, which allows you to display weather data in a particular area without spending time searching and resources to install and configure a personal weather station.

Field of use – embedded systems, Internet of things.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	8
ВСТУП	9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ	11
1.1 Огляд популярних інформаційних ресурсів для перегляду прогнозу погоди.....	11
1.2 Огляд програмно-апаратних пристроїв для відображення погоди....	12
1.3 Висновки за розділом та постановка задачі	15
2 ВИБІР ІНСТРУМЕНТАРІЮ РОЗРОБКИ СИСТЕМ	18
2.1 Вибір апаратно-програмної платформи.....	18
2.2 Вибір середовища розробки.....	21
2.3 Вибір API для отримання даних про погоду.....	22
2.4 Висновки за розділом 2	24
3 ПРОЄКТУВАННЯ СИСТЕМИ.....	25
3.1 Проєктування загальної архітектури системи	25
3.2 Проєктування прототипу графічного інтерфейсу	26
3.3 Проєктування схеми електричної принципової.....	27
3.4 Діаграма послідовності виконання програми	29
3.5 Висновки до розділу 3	30
4 КЕРІВНИЦТВО З ВИКОРИСТАННЯ ПРОГРАМНОГО ПРОДУКТУ	31
4.1 Призначення та умови використання програмного продукту	31
4.2 Функції системи	31
4.3 Вхідні та вихідні дані	31
4.4 Налаштування програмного продукту.....	32
4.5 Взаємодія із програмним продуктом	34
4.6 Висновки до розділу 4	36
ВИСНОВКИ.....	37
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	39

ДОДАТОК А Текст програми.....	41
ДОДАТОК Б Слайди презентації	51

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- API – Application Programming Interface
- IoT – Internet of Things (Інтернет речі)
- JSON – JavaScript Object Notation

ВСТУП

На сьогодні основною метою розвитку інформаційних технологій є створення природного інтерфейсу між людиною і машинами. Велика кількість населення починає втомлюватись від постійного відволікання на смартфони, адже у наш час навіть пошук крихітної порції даних здатний затягнути людину у нескінченний потік інформації соціальних мереж, новин та медіаплатформ. Через це особа може витратити набагато більше часу, ніж розраховувала.

Подібною крихтою важливої щоденної інформації можна вважати прогноз погоди – для отримання метеозведення людина звертається до браузера, який в процесі пошуку відволікає особу на перегляд інформаційних сторінок, новин та цікавих статей.

Розвиток Інтернет речей призвів до появи різних автоматизованих метеостанцій, які необхідно розміщувати безпосередньо у місці дослідження погоди. Подібні системи збирають різноманітні дані щодо стану навколишнього середовища та передають результати на смартфон користувача або спеціалізований пристрій, побудований на основі програмно-апаратних платформ, аналогічних Arduino.

Встановлення метеостанції передбачує розміщення конструкції з різноманітними сенсорами та датчиками, мета яких – проводити дослідження стану навколишнього середовища обраної місцевості в поточний момент часу. Система має свої переваги та недоліки, такі як: витрата грошей на матеріали для установки та налаштування станції спеціалістом, можливість виникнення похибок через низьку якість сенсорів або некоректне налаштування обладнання, обмежена область дослідження.

На сьогоднішній день завдяки поширенню Інтернет речей також існує можливість отримання точних даних про стан погоди і без наявності особистої метео-установки. Для цього існують різноманітні сервіси, які надають прогноз погоди на сайтах через спеціально розроблений інструмент API. Використання різних API методів для отримання даних зі спеціалізованих інтернет ресурсів

може допомогти скоротити витрати на розробку програмного-апаратного комплексу. При цьому дані відносно стану навколишнього середовища будуть точними та дозволять відслідковувати погоду в будь якій точці.

Таким чином розробка пристрою, який буде здатний виводити користувачеві інформацію щодо стану погоди в будь-який час є актуальним завданням. Подібна система прекрасно підійде для розміщення в будинку та буде корисною для користувача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Огляд популярних інформаційних ресурсів для перегляду прогнозу погоди

Крім того, що існує звичайний Google пошук, завдяки якому можна подивитись прогноз, також на мобільних девайсах є додаткові програми та віджети за допомогою яких можна переглянути погоду. Найбільш популярні з них перелічені нижче.

AccuWeather може відображати щотижневі, щоденні і щогодинні метеозведення. З них можна дізнатися про температуру повітря в градусах Цельсія або Фаренгейта, вологість, опади, швидкість і напрямок вітру, а також час сходу і заходу сонця. Також сервіс здатний повідомити користувачу про інтенсивність сонячного випромінювання, або насування бурі. Крім того, AccuWeather має вбудовану карту, що дозволяє в реальному часі стежити за погодою в вашому регіоні, країні або на цілому континенті. Є підтримка годин з Wear OS і watchOS, а також симпатичні віджети. Сервіс AccuWeather безкоштовний, але за відключення реклами доведеться заплатити [1].

1Weather – дуже популярний погодні додаток, яке надає всі необхідні інформацію про погоду в зручному і красивому виді. 1Weather показує тижневий прогноз, наведений в реальному часі, анімації сходів, заходів і фаз місяця. Ви можете вибрати до 12 міст, в яких хочете відстежувати температуру повітря, швидкість і напрям вітру, вологість і опади – це стане в нагоді мандрівникам. 1Weather включає в себе погодні карту. Є в ньому і віджети різних розмірів. Додаток підтримує годинник з Wear OS. Всі функції 1Weather доступні у безкоштовній версії, позбутися реклами можна також за гроші [2].

The Weather Channel один з найточніших погодніх додатків. Він неймовірно популярний за кордоном, але прекрасно підходить і користувачам з країн СНД. Надає прогнози погоди на кожну годину, на 15 днів і на вихідні. З відомостей можна дізнатися про реальну і відчутну температуру, швидкість

вітру, опади і інтенсивність ультрафіолетового випромінювання. На панелі сповіщень ви зможете розмістити інформацію про погоду в будь-яких точках світу. Додаток відмінно оптимізовано і для смартфонів, і для планшетів. В The Weather Channel є зручні віджети з різними співвідношеннями сторін під будь-який робочий стіл [3].

Weather Radar Forecast – приємний додаток з цікавою фішкою: він вміє підлаштовувати фон вашого екрану блокування до подій за вікном. Крім того, в ньому є різноманітні віджети з відомостями про погоду в реальному часі, зображеннями вашого міста, годинами і календарем. Weather Radar Forecast відображає дані про температуру і опади в рядку стану Android, так що вони будуть у вас перед очима, навіть коли відкриті інші додатки. Ще він вміє надсилати сповіщення про раптові урагани, бурі і лавини [2].

Недоліками такого рішення є витрати ресурсів пристрою, та витрати часу людини, яка разом з погодою переглядає ще багато чого взявши до рук смартфон.

1.2 Огляд програмно-апаратних пристроїв для відображення погоди

Метеостанція – це установка на суші, на морі та в повітрі, що має прилади та обладнання для вимірювання атмосферних умов для надання інформації для прогнозів погоди та вивчення погоди та клімату. Вимірювання включають температуру, атмосферний тиск, вологість, швидкість вітру, напрямок вітру та кількість опадів [4].

ІоТ (Інтернет речі) – це система взаємопов'язаних механічних та цифрових машин, об'єктів тощо, з можливістю передачі даних через мережу без взаємодії людина-людина або людина-комп'ютер [5].

Погода сильно варіюється залежно від координат місця – дані можуть відрізнитись навіть в декількох кілометрах один від одного. Супутникові спостереження за погодою є дорогими і неточними в прогнозуванні погоди для

цих місцевих районів. Попередні дослідження показали, що розподілена наземна метеостанція перевершує супутникові спостереження при вимірюванні швидкості / напрямку та температури, вологості, тиску опадів та вітру, а також вимірює дещо різні метеорологічні величини, такі як температура поверхні моря, висота хвилі та хвильовий період, температура води ... [5]. Тип метеостанції змінюється залежно від місця розташування та цілей її використання. Для їх порівняння згадаємо найвідоміші типи наземних метеорологічних станцій, які в свою чергу поділяються на два типи сухопутних метеорологічних станцій та водяних метеостанцій.

Наземні метеостанції:

- синоптична метеостанція;
- персональна метеостанція (включає домашню метеостанцію).

Персональна метеостанція (рис. 1.2) – це набір приладів для вимірювання погоди, що експлуатуються приватною особою [6]. Ці датчики можуть відрізнитися залежно від моделі, але більшість вимірюють швидкість та напрямок вітру, зовнішню та внутрішню температуру, вологість на вулиці та в приміщенні, барометричний тиск, кількість опадів та ультрафіолетове або сонячне випромінювання. Інші доступні датчики можуть вимірювати вологість ґрунту, температуру ґрунту та вологість [4].

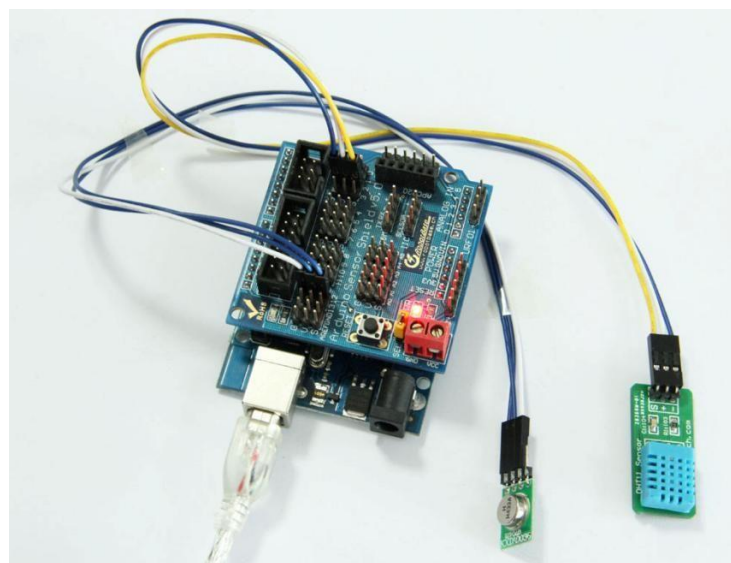


Рисунок 1.1 – Персональна метеостанція [6]

Персональні метеостанції, як правило, включають прилад (цифрову консоль), яка забезпечує зчитування даних, що збираються сенсорами – вони здатні вимірювати температуру, відносну вологість, тиск, падіння дощу, швидкість і напрямок вітру. Або спеціально розроблені додатки на смартфони.



Рисунок 1.2 – Прилад для виводу даних [7]

Недоліком такого рішення є необхідність встановлювати додаткову конструкцію з різними сенсорами та датчиками для зчитування кліматичних умов. Крім того, персональні метеостанції можуть давати похибки у випадку низької якості сенсорів та при їх неправильному налаштуванні.

Синоптичні метеорологічні станції (рис. 1.3) – це прилади, які збирають метеорологічну інформацію в синоптичний час 00:00, 06:00, 12:00, 18:00 (UTC) та в проміжні синоптичні години 03:00, 09:00, 15:00, 21:00 (UTC) [4].

Загальними інструментами вимірювання є анемометр, вітрозахист, датчик тиску, термометр, гігрометр та манометр [4]. Погодні заходи відформатовані у спеціальному форматі та передані до ВМО, щоб допомогти моделі прогнозу погоди [4]. Синоптична метеостанція зазвичай вимірює такі параметри [8]: швидкість та напрямок вітру на висоті 10 метрів; опади/інтенсивність опадів/накопичення опадів; глобальна радіація; тривалість сонячного проміння (зазвичай за розрахунком); температура повітря; точка роси (за розрахунком); транспірація з потенційним

випаровуванням; температура ґрунту; термічний індекс; відносна вологість повітря; атмосферний тиск; справжня погода/минула погода; хмара (тип, кількість, висота). Але такі станції коштовні та їх неможливо встановити без офіційного дозволу держави.



Рисунок 1.3 – Синоптична метеостанція [9]

Такі потужні метеостанції мають свої власні сайти та застосунки для відображення даних. І на сьогодні завдяки розвитку IoT звичайні користувачі можуть також користуватися цими даними використовуючи різні API методи.

1.3 Висновки за розділом та постановка задачі

В розділі були проведені дослідження існуючих інструментальних засобів для зчитування та відображення синоптичної картини в заданому регіоні. Проведені дослідження можна звести в порівняльній таблиці 1.1.

Виходячи з наведеного порівняння можна зробити висновки, що встановлення власної станції не є доцільним через коштовність або

можливості отримання неточних даних, у випадку з персональною метеостанцією, необхідність налаштування та підтримки в робочому стані всіх компонентів системи. Мобільні застосунки в свою чергу дуже відволікають користувача на інші ресурси.

Таблиця 1.1 – Порівняння існуючих засобів для моніторингу погоди

Засіб	Переваги	Недоліки
Мобільні застосунки та віджети	Безкоштовно. Можливість відстежувати погоду на декілька днів/місяць вперед.	Затрати ресурсів девайсу. Реклама, відключення якої коштує грошей. Затрати часу користувача, через додаткове відволікання на інші речі в смартфоні.
Персональні станції	Дешеве апаратне рішення порівняно з синоптичною станцією. Можливість повторного використання.	Необхідність встановлювати додаткову конструкцію з різними сенсорами та датчиками для зчитування кліматичних умов, крім того персональні метеостанції можуть давати похибки якщо сенсори неякісні або біли неправильно налаштовані
Синоптичні станції	Точність вимірювання.	Такі станції коштовні і їх неможливо встановити без офіційного дозволу держави. Необхідність налаштування.

Крім того, на сьогодні як показали дослідження, завдяки IoT існують способи зчитування точних даних з метеостанції за допомогою API методів. Це дозволяє уникнути недоліків перерахованих в таблиці для кожного з існуючих засобів.

Таким чином, завданням даної дипломної кваліфікаційної роботи бакалавра є розробка програмно апаратного комплексу для відображення даних отриманих з мережі за допомогою API методів.

2 ВИБІР ІНСТРУМЕНТАРІЮ РОЗРОБКИ СИСТЕМ

2.1 Вибір апаратно-програмної платформи

Одним з найважливіших кроків у реалізації проєкту є вибір платформи, найбільш підходящої для поставленої мети. Існує багато критеріїв відбору, які ми повинні враховувати, такі як:

- кількість аналогових та цифрових контактів;
- енергоспоживання;
- обчислювальна потужність;
- підключення до Інтернет;
- розмір запрограмованої пам'яті;
- розмір оперативної та пам'яті EPROM;
- розмір самої платформи;
- ціна на ринку.

Таблиця 2.1 представляє порівняльне дослідження деяких платформ, доступних на ринку [10,11].

Таблиця 2.1 – Порівняльне дослідження платформ

	NodeMCU ESP32	Arduino Nano	Arduino Uno Wi-Fi
Мікроконтролер	ESP32	ATmega328p	ATmega4809
Енергоспоживання	3.3V	5V	5V
Потужність	2.3-3.6V	7-12V	7-12V
Споживання струму	20-240mA	19-180mA	50-150mA
Оперативна пам'ять	520KB	2KB	6KB
Wi-Fi	так	ні	так
Ціна	130 грн	270 грн	1300 грн

NodeMCU – альтернативна прошивка-інтерпретатор мови Lua, написана в далекому 2014 для ESP8266. З релізом ESP32 команда розробників почала роботу над новою версією інтерпретатора, який підтримував би всі можливості оновленої платформи [12].

Реліз NodeMCU-32 Development board вже відбувся, тому його вже можна випробувати на модулях, в основі яких лежить ESP32. Разом з прошивкою паралельно розроблялася офіційна налагоджувальна плата для прошивки, яка має аналогічну назву - NodeMCU 32S [12].

На платі вже встановлено все необхідне для запуску нового модуля в проект, а саме: micro-USB роз'єм для живлення і зв'язку модуля з ПК перетворювач інтерфейсів USB-UART на основі CP2102, який створює на ПК віртуальний COM-порт з логічними рівнями TTL для полегшеної зв'язку з ESP32 (зв'язок, аналогічна зв'язку ПК з Arduino) лінійний стабілізатор напруги AMS1117-3.3V, який забезпечує ESP32 оптимальною напругою живлення, знижуючи вхідна напруга (з Піна Vin) або 5V з USB в прийнятні 3.3V пару кнопок для управління і два світлодіода - призначений для користувача і світлодіод індикації активності передачі по послідовному порту однорядні роз'єму типу «тато», на яких розведені всі піни модуля ESP-32S сам Wi-Fi модуль на основі ESP32 - ESP32S На модуль, що стоїть на платі, вже встановлена найсвіжіша на момент випуску прошивка NodeMCU, тому можна почати працювати, просто підключивши плату до комп'ютера, встановивши драйвера (які можна скачати на офіційному сайті Silabs) і відкривши ваш улюблений термінал послідовного порту [11].

Можна використовувати спеціально розроблені програми типу ESPLorer, що полегшують доступ до основних функцій прошивки і дозволяють зручно організовувати робочі файли. ESP32 має дійсно багато відмінностей від свого «молодшого брата», ESP32 -старший брат отримав ще одне ядро, і тепер тактова частота обох ядер може досягати 240МГц з продуктивністю до 600 мільйонів ітерацій в секунду (MIPS, millions iterations per second); також нова система-на-чипі отримала Bluetooth найновішою

специфікації 4.2 з можливістю переходу в режим наднизького споживання BLE, оновлений криптографічний движок, можливість шифрування вмісту зовнішньої Flash-пам'яті за допомогою п'яти різних алгоритмів, вбудовану ППЗУ і ОЗУ на майже полмегабайта [12].

З надбюджетного модуля для домашніх застосувань серія ESP починає перетворюватися в повноцінне пристрій з практично необмеженими можливостями для творчості! Development board передбачає можливість живлення від зовнішнього нестабілізованого джерела з напругою від 5 до 12В через вхід Vin. Але при перевищенні напруги живлення може призвести до пошкодження стабілізатора, робота плати не гарантується [12].

Цікава платка на якій інтегровано класичний контролер Arduino UNO та Wi-Fi модуль на ESP8266 з 8Mb flash пам'яті. Обидві частини можуть працювати як спільно так та абсолютно незалежно. Режим роботи обох контролерів задається перемикачем. Програмувати обидва контролери можна з середовища розробки Arduino IDE.

Для переведення ESP8266 в режим програмування потрібно натиснути кнопку "ESP Reboot". Режим роботи плати задається перемикачем [13].

Платформа Nano, побудована на мікроконтролері ATmega328 (Arduino Nano 3.0) або ATmega168 (Arduino Nano 2.x), має невеликі розміри і може використовуватися в лабораторних роботах. Вона має схожу з Arduino Duemilanove функціональність, проте відрізняється складанням. Відмінність полягає у відсутності силового роз'єму постійного струму і роботі через кабель Mini-B USB. Nano розроблена і продається компанією Gravitech.

Arduino Nano може отримувати живлення через підключення Mini-B USB, або від нерегульованого 6-20 В (висновок 30), або регульованого 5 В (висновок 27), зовнішнього джерела живлення. Автоматично вибирається джерело з найвищим напругою [14].

Мікросхема FTDI FT232RL отримує живлення, тільки якщо сама платформа запитана від USB. Таким чином при роботі від зовнішнього джерела (НЕ USB), буде відсутня напруга 3.3 В, що генерується мікросхемою

FTDI, при цьому світлодіоди RX і TX блимають тільки при наявності сигналу високого рівня на висновках 0 і 1 [14].

Мікроконтролер ATmega168 має 16 кБ флеш-пам'яті для зберігання коду програми, а мікроконтролер ATmega328, в свою чергу, має 32 кБ (в обох випадках 2 кБ використовується для зберігання завантажувача). ATmega168 має 1 кБ ОЗУ і 512 байт EEPROM (яка читається і записується за допомогою бібліотеки EEPROM), а ATmega328 - 2 кБ ОЗУ і 1 Кб EEPROM.

Кожен з 14 цифрових висновків Nano, використовуючи функції pinMode (), digitalWrite (), і digitalRead (), може налаштовуватися як вхід або вихід. Висновки працюють при напрузі 5 В. Кожен висновок має навантажувальний резистор (стандартно відключений) 20-50 кОм і може пропускати до 40 мА [14].

Недоліком для даного проєкту є відсутній вбудований модуль підключення до мережі.

Базуючись на аналізі таблиці найкращий мікроконтролер для розробки проєкту, у цьому порівнянні, є платформа на основі ESP. Платформа на основі ESP швидкі, мають низьке енергоспоживання, велику пам'ять та вбудований Wi-Fi. Також ціна дуже низька.

2.2 Вибір середовища розробки

Arduino представляє собою інструмент для проєктування електронних пристроїв (електронний конструктор) який більш щільно взаємодіє з навколишнім фізичним середовищем, що є стандартними персональними комп'ютерами, які фактично не виходять за рамки віртуальності. Це платформа, призначена для «фізичних обчислень» з відкритим програмним кодом, побудована на простому друкованому платіжі із сучасною середою для написання програмного забезпечення. Arduino застосовується для створення електронних пристроїв з можливістю прийому сигналів від різних цифрових та аналогових датчиків, які можуть бути підключені до нього та управління

різними виконавчими пристроями. Проєкти пристроїв, основані на Arduino, можуть працювати самостійно або взаємодіяти з програмним забезпеченням на комп'ютері (напр.: Flash, Processing, MaxMSP). Плати можуть бути зібрані користувачем самостійно або зібрані в сборі. Серед розробки програм з відкритим вихідним текстом доступна для безкоштовного скачування. Язык програмування Arduino - це реалізація проводки, схожа платформа для «фізичних обчислень», заснована на мультимедійному середовищі програмування обробки [15].

Переваги Arduino IDE є [16]:

- підсвічування коду;
- швидка заливка скетчу в платформу;
- можливо зробити простий проєкт за кілька хвилин;
- високорівнева мова програмування;
- безліч корисних вбудованих функцій, які прискорюють розробку;
- безліч бібліотек;
- велика інтернет спільнота.

Було обрано середовище розробки Arduino IDE, оскільки вона цілком відповідає вимогам і дозволяє виконати всі необхідні завдання для реалізації проєкту.

2.3 Вибір API для отримання даних про погоду

Наступним питанням, яке необхідно було вирішити, виявився пошук достовірної інформації про погоду. Існують API для Gismeteo [17], але для того щоб почати його використовувати існує ряд проблем, які необхідно вирішити [18], такі як реєстрація та отримання токена. Крім того для того, щоб використовувати сам API необхідно ознайомитись з документацією яка наводиться на сайті.

Як показало дослідження порівняно простим та зрозумілим сервісом є сервіс OpenWeather [19].

Він має безкоштовний та простий API, можливостей якого достатньо для реалізації проєкту розробки програмно апаратного комплексу для відображення поточного стану погоди у заданій місцевості. На рисунку 2.1 наводиться приклад JSON відповіді на запит до сервісу.

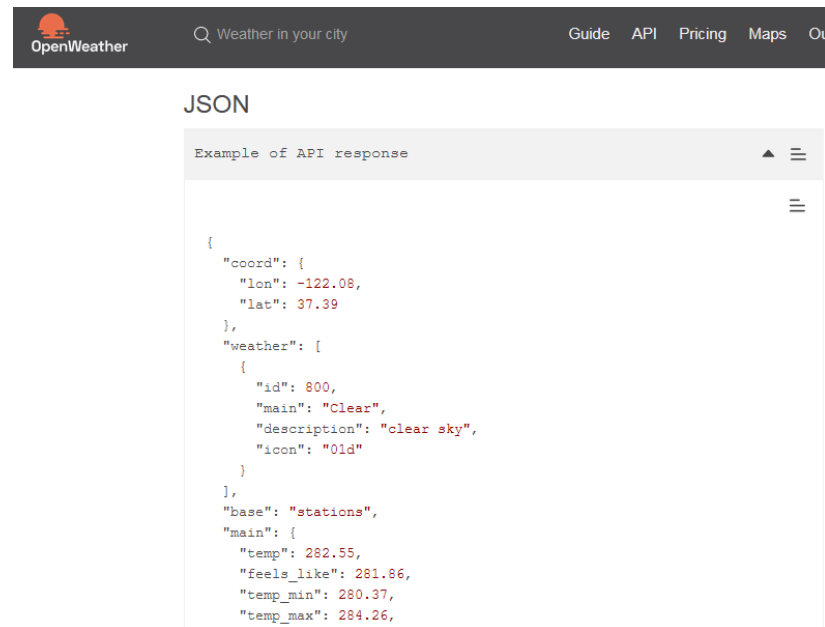


Рисунок 2.1 – Приклад відповіді від сервісу OpenWeather [20]

Структура передачі даних відповідає архітектурі, відомої як JSON, яка позначає нотацію об'єктів JavaScript. JSON – це так званий «обмін даними». Простіше кажучи, програмісти можуть запитувати інформацію в людському і машиночитному форматі. JSON – це стандартизований спосіб, завдяки якому програмісти, які створюють API, та його користувачі, знають, яким чином кожна з їхніх програм може «спілкуватися» одна з одною. Дані JSON зазвичай представляють собою набір пар ім'я і значення. Програмісти можуть відправляти їх у якості запитів до API, та API буде відповідати значеннями в об'єкті JSON відповіді [21].

З цього запиту нас цікавить два рядки “description” та “icon”. В таблиці 2.2 наводиться приклади всіх можливих значень цих рядків.

Таблиця 2.2 – Всі значення параметрів «description» та «icon»

«icon» (день/ніч)	«description»
01d/01n	Чисте небо
02d/02n	Малохмарна
03d/03n	Містами хмарно
04d/04n	Хмарно з проясненнями
09d/09n	Зливовий дощ
10d/10n	Дощ
11d/11n	Гроза
13d/13n	Сніг
50d/50n	Туман

Інформація може бути передана в або з API використовуючи різні методи, такі як GET, PUT, POST.

2.4 Висновки за розділом 2

В даному розділі проведено аналіз та обґрунтування вибору інструментальних засобів для розробки програмно-апаратного комплексу для відображення погоди в заданій місцевості. Проведено порівняльний аналіз апаратно-програмних платформ, виходячи з якого була обрана платформа на базі мікроконтролера ESP32, через свою низьку ціну та швидкодію.

Для налагодження та прошивки програми в мікроконтролер було обрано середовище розробки Arduino IDE через свою розповсюдженість, наявність великої кількості бібліотек та інтернет спільноти. Сервісом для отримання даних про погоду було обрано OpenWeather через свою простоту та безкоштовність. В розділі наводиться приклад отримання відповіді у JSON форматі від сервісу.

3 ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Проєктування загальної архітектури системи

Першим кроком проєктування системи є проєктування архітектури програмно-апаратного комплексу, в основі якої лежить REST технологія. Розроблена архітектура зображена на рисунку 3.1.

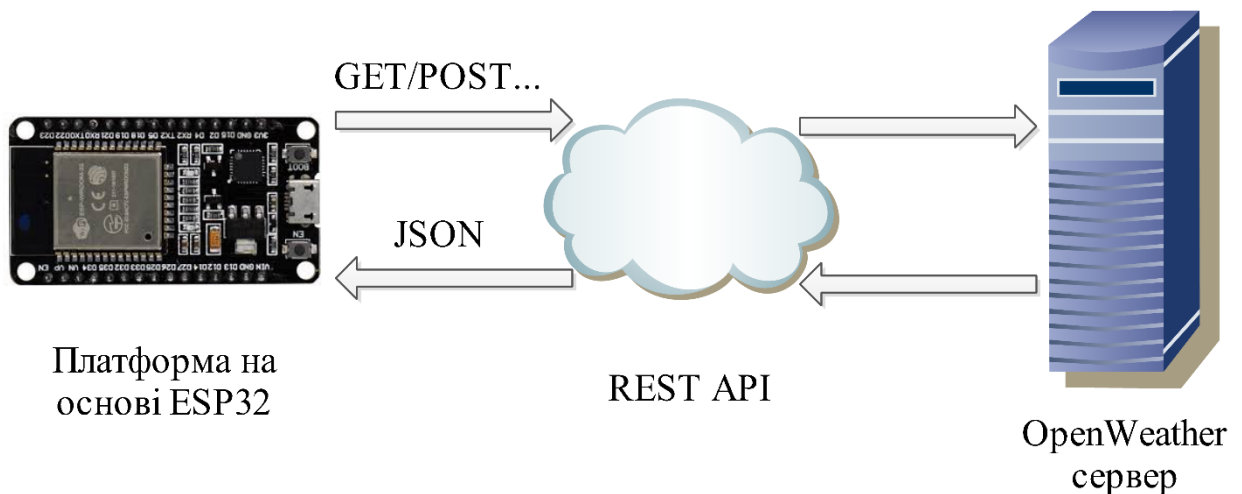


Рисунок 3.1 – Загальна архітектура системи

Сервер OpenWeather надає REST інтерфейс для отримання даних. Сервер отримує запит та надає відповідь на нього. Структура отриманих даних відповідає архітектурі JSON, яка означає нотацію проєкта JavaScript (рис. 2.1).

UML діаграма алгоритму роботи системи зображена на рисунку 3.2 та складається з кроків перерахованих нижче.

Крок 1. Відправлення запиту. Клієнтська програма відправляє структурований запит у вигляді HTTP методу (GET) для отримання даних через Інтернет в інтерфейс веб-API.

Крок 2. API отримує та обробляє запит. API отримує запит, обробляє його і отримує запитувану інформацію з внутрішньої бази даних, яка йому необхідна.

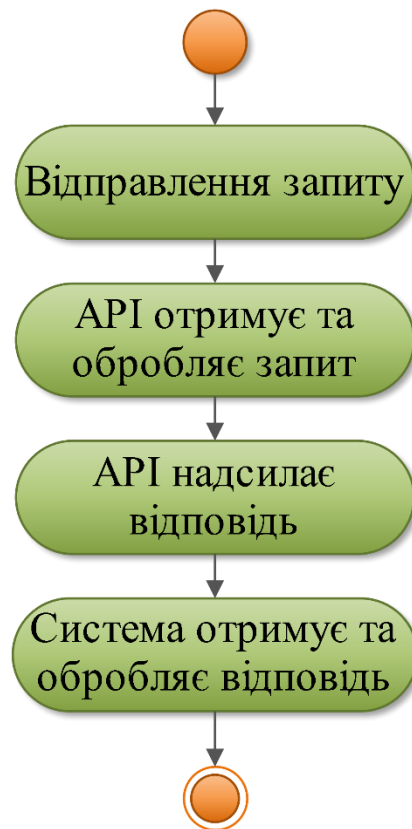


Рисунок 3.2 – UML-діаграма алгоритму роботи системи

Крок 3. API надсилає відповідь. API відповідає через Інтернет на запит, використовуючи той самий структурований формат JSON.

Крок 4. Система отримує та обробляє відповідь. Клієнтська програма отримує відповідь JSON і надає інформацію користувачеві через розроблений інтерфейс.

3.2 Проектування прототипу графічного інтерфейсу

Наступним кроком є розробка прототипу графічного інтерфейсу. Прототип інтерфейсу буде представлений у вигляді RGB світлодіоду, який відповідно до інформації про погоду, отриманої від сервісу OpenWeather буде загорятися тим чи іншим кольором. В таблиці 3.1 наведено відповідність кольорів світлодіоду та отриманих даних від сервісу погоди. Система побудована на основі таблиці 2.2, однак відображатись буде лише частина можливих значень.

Таблиця 3.1 – Відповідність кольорів світлодіоду та отриманих даних від сервісу погоди

“icon” (день/ніч)	“description”	Колір RGB світлодіоду
01d/01n	Чисте небо	Зелений
02d/02n	Малохмарно	Жовтий
04d/04n	Хмарно з проясненнями	Жовтий
09d/09n	Зливовий дощ	Синій
10d/10n	Дощ	Синій
13d/13n	Сніг	Білий

3.3 Проектування схеми електричної принципової

На рисунку 3.3 зображена схема електрична принципова розробленої системи відображення погоди. Компоненти необхідні для проектування апаратно програмного комплексу:

- NodeMCU на базі ESP32;
- RGB світлодіод 1 шт;
- резистори, 3 шт. (220 Ом);
- макетна плата;
- дратові перемички.

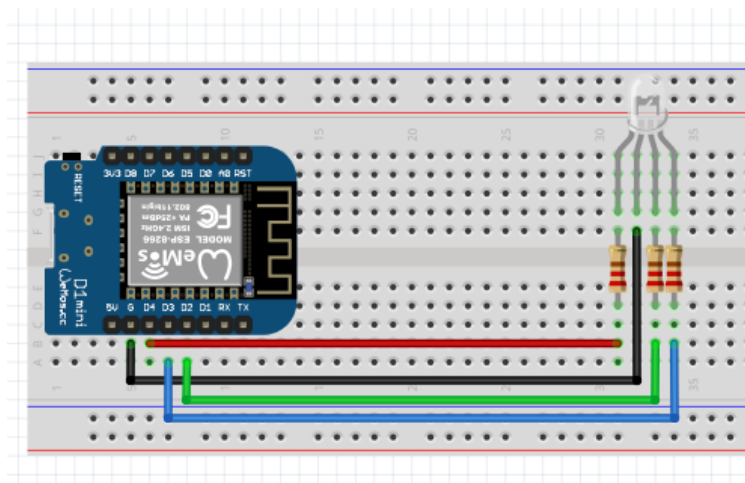


Рисунок 3.3 – Схема електрична принципова прототипу системи

В таблиці 3.2 наведено відповідність контактів платформи NodeMCU та кольорів RGB світлодіоду.

Таблиця 3.2 – Відповідність контактів платформи NodeMCU та кольорів RGB світлодіоду

Колір RGB світлодіоду	Контакт платформи NodeMCU
Червоний	03
Зелений	02
Синій	13

Таким чином, базуючись на розробленій схемі електричній принциповій, яка зображена на рисунку 3.3, був розроблений фізичний прототип системи відображення поточного стану погоди у заданій області (рис.3.4).

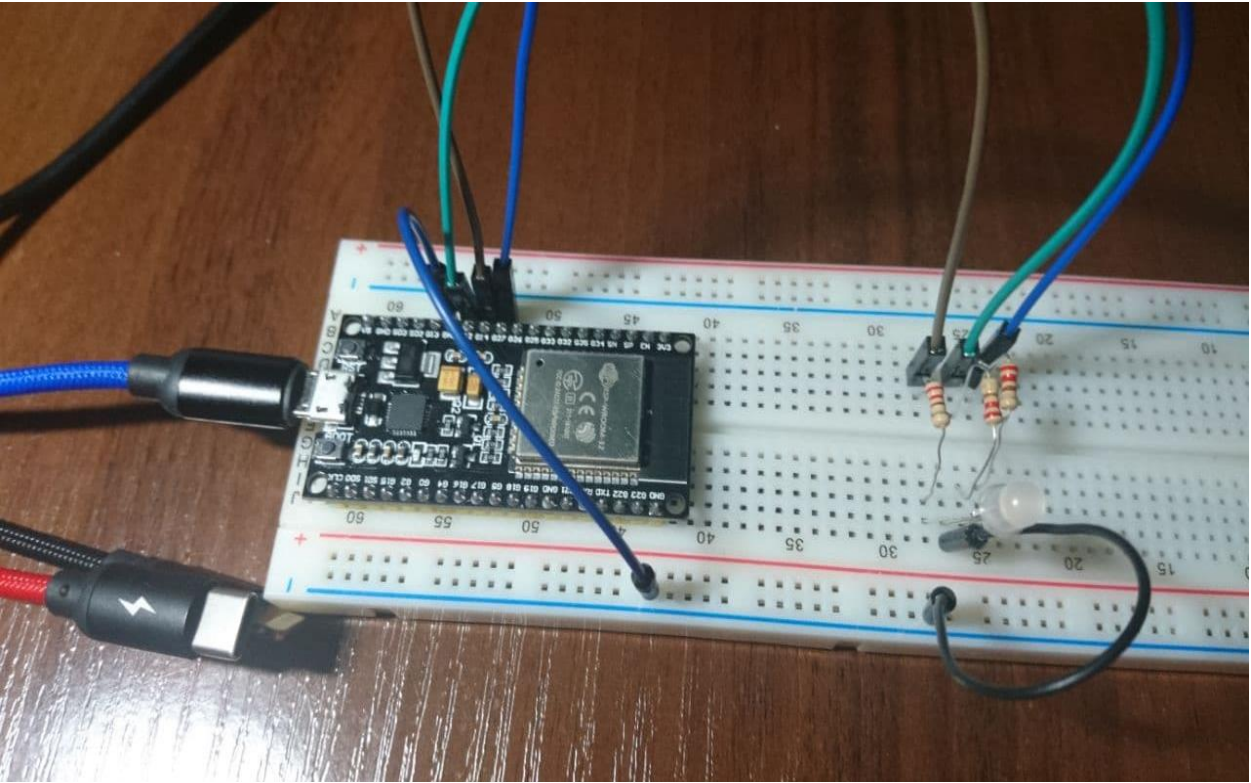


Рисунок 3.4 – Прототип системи

3.4 Діаграма послідовності виконання програми

На рисунку 3.5. представлена діаграма основної програми системи для відображення поточного стану погоди – вона складається з багатьох функцій, сукупність яких забезпечує коректну роботу системи.



Рисунок 3.5 – Діаграма послідовності виконання програми

Основними етапами роботи програми є:

- автоматичний запит погоди 1 раз в 15 хвилин;
- розбір прийнятої відповіді;
- зчитування потрібного параметра;
- виведення його значення відповідним кольором світлодіоду.

3.5 Висновки до розділу 3

Для розроблюваного програмного продукту в даному розділі було спроектовано загальну архітектуру системи та прототип графічного інтерфейсу, який представляє собою RGB світлодіод, який в залежності від отриманих результатів погоди загоряється відповідним кольором. Була розроблена схема електрична принципова, на основі якої спроектовано фізичний прототип системи. Наведено алгоритми роботи системи в цілому та самої програми.

4 КЕРІВНИЦТВО З ВИКОРИСТАННЯ ПРОГРАМНОГО ПРОДУКТУ

4.1 Призначення та умови використання програмного продукту

Програмний продукт представляє собою апаратно-програмний комплекс, який призначений для користувачів, яким необхідно отримувати дані про погоду використовуючи технологію REST API.

Програмний продукт може бути налаштований та використаний на будь-яких подібних пристроях, де встановлена платформа на базі мікроконтролера ESP32 з можливістю підключення до мережі через Wi-Fi.

4.2 Функції системи

Програмний продукт повинен виконувати такі функції:

- автоматично запитувати погоду 1 раз в 15 хвилин;
- розбирати прийняту відповідь;
- зчитувати потрібний параметр;
- виводити його значення відповідним кольором світлодіода

4.3 Вхідні та вихідні дані

Вхідні дані – JSON відповідь у вигляді структури написаною мовою JavaScript з різними параметрами відповідно до надісланого запиту.

У якості вихідних даних до серверу OpenWeather відправляється запит GET, а користувач, за допомогою прототипу графічного інтерфейсу у вигляді RGB світлодіоду отримує різні кольори відповідно до отриманої відповіді від серверу.

4.4 Налаштування програмного продукту

Для роботи з програмним продуктом необхідно виконати ряд інструкцій з його запуску та налаштувань.

У основі цього проєкту лежить інтерфейс прикладного програмування сервісу OpenWeather. Тому, спочатку, для можливості роботи з сервісом OpenWeather користувач повинен зареєструватися на цій платформі і отримати ключ для прикладного інтерфейсу перед тим, як піти далі і зайнятися апаратною і програмною частинами пристрою. Для доступу до інтерфейсу прикладного програмування вам необхідний ключ, так званий APIID. Щоб отримати ключ APIID: відкрийте веб-браузер і відкрийте сайт OpenWeather. Натисніть кнопку «Sign up» (Зареєструватися) і створіть безкоштовний обліковий запис. Після цього відобразиться інформаційна панель з декількома вкладками (див. рис. 4.1). Виберіть вкладку «API Keys» (Ключі для прикладного інтерфейсу) і скопіюйте свій унікальний ключ (з поля «Key»).

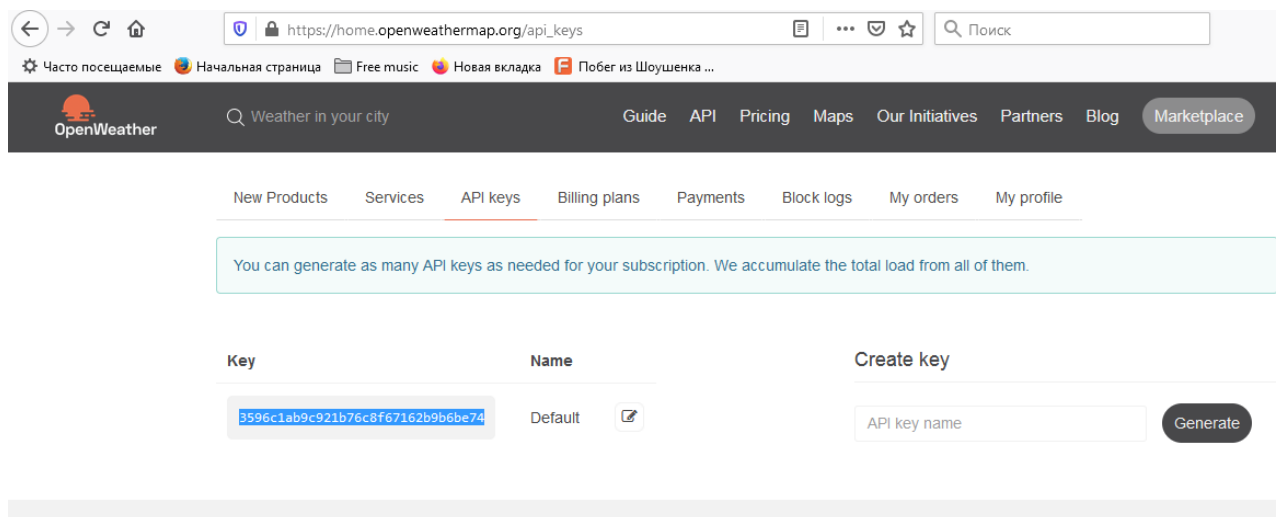


Рисунок 4.1 – Отримання ключа для роботи з сервісом OpenWeather

Щоб отримати інформацію про погоду в певній місцевості, необхідно ввести URL-адресу наведену нижче, замінивши текст з фігурними дужками на назву місцевості і ключ:

```
api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}.
```

Замініть текст {your city} (ваше місто) на назву міста, погоду в якому ви хочете дізнатися, {your country code} (код вашої країни) - на код країни, в якій знаходиться місто, а {your API key} (ключ для прикладного інтерфейсу) - на унікальний ключ, отриманий раніше. Наприклад, URL-адресу для міста Запоріжжя буде таким:

```
http://api.openweathermap.org/data/2.5/weather?q=Zaporizhia&appid=3596c1ab9c921b76c8f67162b9b6be74
```

Скопіювавши URL-адресу в браузер в результаті користувач отримає у відповідь прогноз погоди для заданої місцевості. На рисунку 4.2 зображена відповідь від сервісу щодо погоди у місті Запоріжжя (Україна).

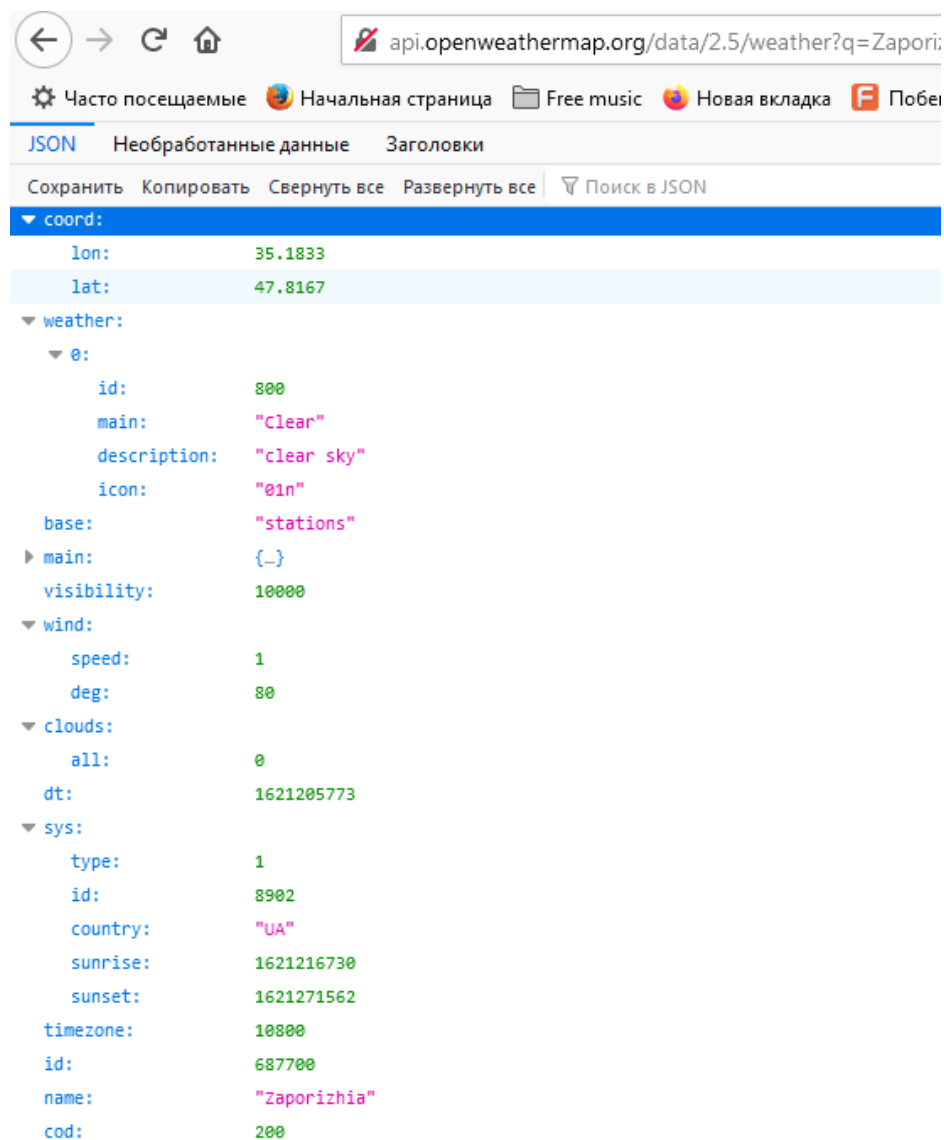


Рисунок 4.2 – Відповідь від сервісу щодо погоди у місті Запоріжжя

Після чого необхідно підключити апаратну частину до комп'ютера та завантажити код до плати. Крім того до середовища Arduino необхідно підключити бібліотеку Arduino JSON.

В таблиці 4.1 представлені основні функції програми та їх опис.

Таблиця 4.1 – Опис основних функцій та деяких змінних оточення

Назва	Опис
printWiFiStatus()	Визначення та вивід статусу з'єднання
makeRequest()	Секретний ключ, для налаштування безпеки системи.
City	Ім'я міста
apiKey	Секретний ключ, для використання в запити для доступу до сервісу
client.read()	Зчитування відповіді від сервера
parseJson()	Розбір тексту рядка у функції parseJson
void parseJson ()	Для синтаксичного аналізу отриманих даних JSON
diffDataAction()	Вивід отриманих даних від сервісу, шляхом загорання того чи іншого кольору світлодіода

Після виконання усіх кроків система налаштована та готова для виконання задач.

4.5 Взаємодія із програмним продуктом

В таблиці 4.2 наводиться всі можливі варіанти кольорів світлодіода, які загоряються на розробленому прототипі системи відображення поточного стану погоди, та представляють собою апаратно-програмний комплекс. В таблиці наводиться відношення погоди, яка на даний момент в заданому місті та відповідний колір світлодіода.

Таблиця 4.2 – Варіанти загоряння світлодіода і їх значення

“description”	Колір RGB світлодіода
Чисте небо	Зелений
Малохмарно	Жовтий
Хмарно з проясненнями	Жовтий
Зливовий дощ	Синій
Дощ	Синій
Сніг	Білий

Після підключення та налаштування платформи можна починати тестування системи. Приклад відображення поточної погоди зображений на рисунку 4.3.

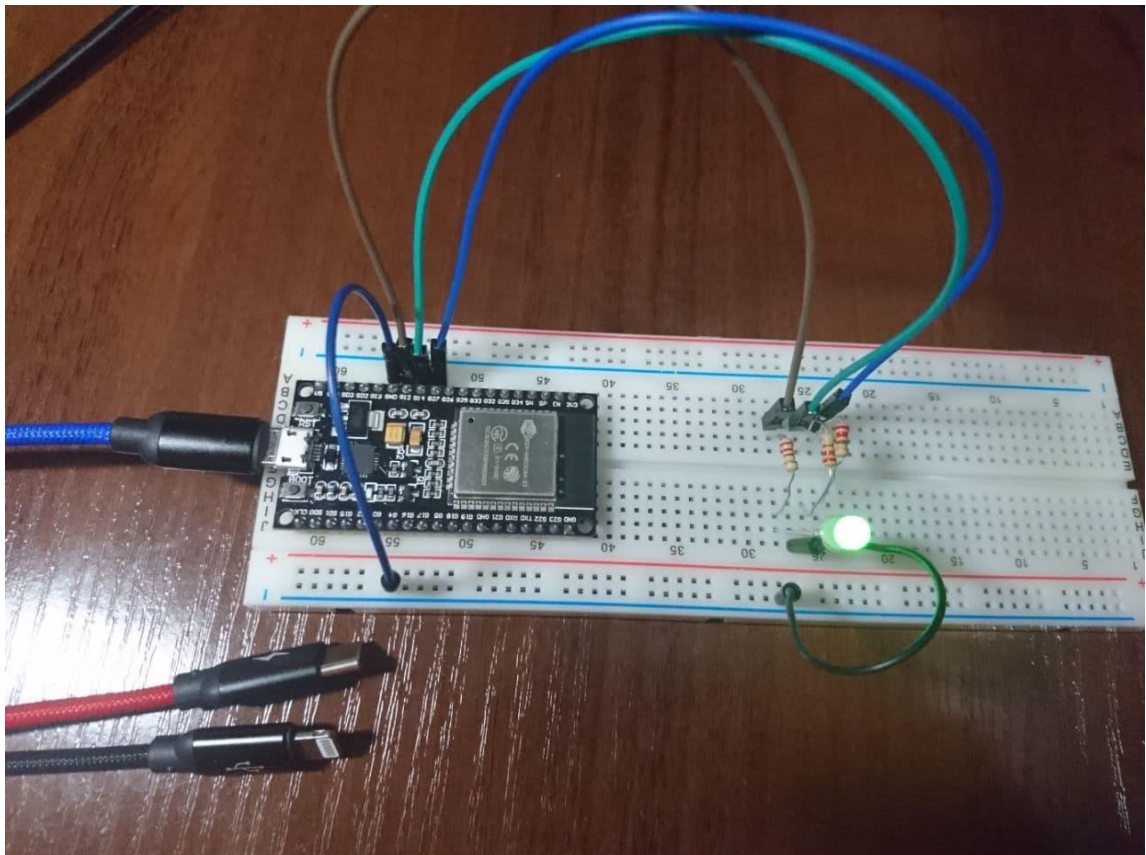


Рисунок 4.3 – Зібраний прототип системи в роботі

4.6 Висновки до розділу 4

У даному розділі наведено призначення системи, умови використання розробленого апаратно програмного комплексу, його вхідні та вихідні дані, а також наведено основні функції системи. Крім того, надається детальна інструкція для користувача щодо встановлення та використання апаратно-програмного комплексу.

ВИСНОВКИ

Погодні умови – один з найважливіших факторів, що впливають на життя людини в багатьох сферах життя. Знання точних погодних змін відіграє ефективну роль у покращенні та полегшенні людського життя, а також у його збереженні.

В ході розробки дипломної кваліфікаційної роботи бакалавра були проведені дослідження існуючих інструментальних засобів для зчитування та відображення синоптичної картини в заданому регіоні. Проведені дослідження були зведені в порівняльній таблиці.

Виходячи з наведеного порівняння зроблено висновки, що встановлення власної станції не є доцільним через кошовність та можливість отримання неточних даних, у випадку з персональною метеостанцією, необхідність налаштування та підтримки в робочому стані всіх компонентів системи. Мобільні застосунки в свою чергу дуже відволікають користувача на інші ресурси.

Крім того, на сьогодні, як показали дослідження, завдяки IoT існують способи зчитування точних даних з метеостанції за допомогою API методів. Це дозволяє уникнути можливих недоліків кожного з розглянутих аналогічних засобів.

Проведено аналіз та обґрунтування вибору інструментальних засобів для розробки програмно апаратного комплексу для відображення погоди в заданій місцевості. Проведено порівняльний аналіз апаратно-програмних платформ, виходячи з якого була обрана платформа на базі мікроконтролера ESP32, через свою низьку ціну та швидкодію.

Для налагодження та прошивки програми в мікроконтролер було обрано середовище розробки Arduino IDE через свою розповсюдженість, наявність великої кількості бібліотек та інтернет спільноти. Сервісом для отримання даних про погоду було обрано OpenWeather через свою простоту

та безкоштовність. Крім того, в розділі наводиться приклад отримання відповіді у JSON форматі від сервісу.

Для розроблюваного програмного продукту було спроектовано загальну архітектуру системи, прототип графічного інтерфейсу, який представляє собою RGB світлодіод, який в залежності від отриманих результатів погоди загоряється відповідним кольором. Була розроблена схема електрична принципова на основі якої спроектовано фізичний прототип системи. Наведено алгоритми роботи системи в цілому та самої програми.

Наведено призначення системи, умови використання розробленого апаратно-програмного комплексу, його вхідні та вихідні дані, а також основні функції системи. Крім того, надається детальна інструкція для користувача щодо встановлення та використання апаратно-програмного комплексу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- 1 12 приложений, которые помогут не попасть под дождь [Электрон. ресурс]. – Режим доступа: <https://daily.afisha.ru/brain/6104-12-prilozheniy-kotorye-pomogut-ne-popast-pod-dozhd>.
- 2 10 красивых приложений с точными прогнозами погоды [Электрон. ресурс]. – Режим доступа: <https://lifehacker.ru/tochnye-prognozy-pogody-android-ios>.
- 3 15 лучших погодных приложений для Android [Электрон. ресурс]. – Режим доступа: <https://androidelf.com/luchshie-pogodnye-prilozheniya-dlya-android-v-2019-godu>.
- 4 Weather station [Electronic source]. – Access mode: https://en.wikipedia.org/wiki/Weather_station.
- 5 Internet of Things (IoT)Automatic Weather Station, October 2020 [Electronic source]. – Access mode: https://www.researchgate.net/publication/345877902_Internet_of_Things_IoTAutomatic_Weather_Station.
- 6 IoT weather station sensor [Electronic source]. – Access mode: <https://nl.pinterest.com/edzob/iot-weather-station-sensor>.
- 7 Arduino Weather Station [Electronic source]. – Access mode: <https://www.instructables.com/Arduino-weather-station>.
- 8 Synoptic Automatic Weather Station [Electronic source]. – Access mode: <https://www.pulsonic.com/en/synoptic-automatic-weather-stations>.
- 9 Новые автоматические метеостанции улучшат климатический мониторинг Android [Электрон. ресурс]. – Режим доступа: <https://meteojournal.ru/novosti/novye-avtomaticheskie-meteostanczii-uluchshat-klimaticheskij-monitoring-v-uzbekistane>.
- 10 ESP32 и Ардуино [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/post/404685>.

- 11 Магаин Arduino [Электрон. ресурс]. – Режим доступа: <https://arduino.ua>.
- 12 NodeMCU — отладочная плата на ESP-32 + CP2102 [Электрон. ресурс]. – Режим доступа: <https://volti.ru/shop/nodemcu-32s>.
- 13 Плата розробника Arduino UNO WiFi R3 [Электрон. ресурс]. – Режим доступа: <https://arduino.ua/prod2014-plata-razrabotchika-arduino-unowifi-r3-ot-robodyn>.
- 14 Arduino Nano [Электрон. ресурс]. – Режим доступа: <http://arduino.ru/Hardware/ArduinoBoardNano>.
- 15 Arduino IDE [Электрон. ресурс]. – Режим доступа: <https://sites.google.com/site/100voltsamper/sredy-razrabotok/arduino-ide>.
- 16 Плюсы и минусы Arduino [Электрон. ресурс]. – Режим доступа: <http://micpic.ru/home/proekty-na-arduino/188-plyusy-i-minusy-arduino.html>.
- 17 Gismeteo Api документация [Электрон. ресурс]. – Режим доступа: <https://www.gismeteo.ru/api>.
- 18 Как получить доступ к API GISMETEO [Электрон. ресурс]. – Режим доступа: <https://qna.habr.com/q/558315>.
- 19 Вывод прогноза погоды из интернета на модуль со светодиодной матрицей [Электрон. ресурс]. – Режим доступа: <http://wiki.amperka.ru/дисплеи-и-индикаторы:8x8rgb>.
- 20 OpenWeather [Electronic source]. – Access mode: <https://openweathermap.org>.
- 21 Что такое API? [Электрон. ресурс]. – Режим доступа: <https://miiledi.ru/chto-takoe-api>.

ДОДАТОК А
Текст програми

```
#include "ESP8266WiFi.h"
#include "WiFiClient.h"
#include "ArduinoJson.h"

// деталі WiFi мережі
char ssid[] = "InetaNet"; //ім'я мережі
char pass[] = "*****"; // пароль

WiFiClient client;

// ім'я серверу Weather API до константи
const char server[] = "api.openweathermap.org";

String City = "Zaporizhia";

// API Key
String apiKey = "3596c1ab9c921b76c8f67162b9b6be74";

String text;

#define JSON_BUFF_DIMENSION 2500

unsigned long lasttimeConne = 15 * 60 * 1000; // час останнього з'єднання
const unsigned long Interval = 15 * 60 * 1000; // інтервал запитів 10 хвилин

int jsonend = 0;
boolean StartJ = false;
int status = WL_IDLE_STATUS;
```

```

int redLed = 3;
int greenLed = 2;
int blueLed = 13;

void setup() {
  pinMode(greenLed, OUTPUT);
  pinMode(redLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  Serial.begin(9600);

  text.reserve(JSON_BUFF_DIMENSION);

  WiFi.begin(ssid,pass);
  Serial.println("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected");
  printWiFiStatus();
}

void loop() {
  //запити з 15 хвилинним інтервалом

  if (millis() - lastimeConne > Interval) {
    // час за який відбулось з'єднання:
    lastimeConne = millis();
    makeRequest();
  }
}

```

```

    }
}

// вивід статусу Wifi
void printWiFiStatus() {
    // вивід SSID мережі до якої відувається підключення
    Serial.print("SSID: ");
    Serial.println(WiFi.SSID());

    // вивід WiFi IP адреси:
    IPAddress ip = WiFi.localIP();
    Serial.print("IP Address: ");
    Serial.println(ip);

    // вивід отриманого рівня сигналу
    long rssi = WiFi.RSSI();
    Serial.print("signal strongh (RSSI:");
    Serial.println("dBm");
}

void makeRequest() {
    // завершити будь-які з'єднання перед тим як з'єднатися з сервером та
    відправити новий запит
    client.stop();

    // якщо з'єднання успішне:
    if (client.connect(server, 80)) {

        // відправлення HTTP GET запиту:

```

```

client.println("GET /data/2.5/forecast?q=" + Zaporizhia +
"&3596c1ab9c921b76c8f67162b9b6be74=" + apiKey +
"&mode=json&units=metric&cnt=2 HTTP/1.1");
client.println("Host: api.openweathermap.org");
client.println("User-Agent: ArduinoWiFi/1.1");
client.println("Connection: close");
client.println();

```

```

unsigned long timeout = millis();
while (client.available() == 0) {
  if (millis() - timeout > 5000) {
    Serial.println(">>> Client Timeout !");
    client.stop();
    return;
  }
}

```

```

char c = 0;
while (client.available()) {
  c = client.read();

```

// оскільки json містить однакову кількість відкритих і закритих фігурних дужок, це означає, що ми можемо визначити, коли json повністю отриманий шляхом підрахунку

```

//Serial.print(c);
if (c == '{') {
  StartJ = true;    // встановлення StartJ=true для визначення, що json
повідомлення починається
  jsonend++;
}

```

```

    if (c == '}') {
        jsonend--;
    }
    if (StartJ == true) {
        text += c;
    }
    // якщо jsonend = 0 то ми отримали однакову кількість фігурних дужок
    if (jsonend == 0 && StartJ == true) {
        parseJson(text.c_str()); // розбір тексту рядка c у функції parseJson
        text = "";              // очистити текстовий рядок для наступного разу
        StartJ = false;         // встановіть StartJ на false, щоб вказати, що нове
        повідомлення ще не розпочато
    }
}
}
else {

    // якщо з'єднання не віддбулось

    Serial.println("connection failed");
    return;
}
}

// для синтаксичного аналізу даних JSON
void parseJson(const char * jsonString) {

    //StaticJsonBuffer<4000> jsonBuffer;

```

```

const size_t bufferSize = 2*JSON_ARRAY_SIZE(1) + JSON_ARRAY_SIZE(2)
+ 4*JSON_OBJECT_SIZE(1) + 3*JSON_OBJECT_SIZE(2) +
3*JSON_OBJECT_SIZE(4) + JSON_OBJECT_SIZE(5) +
2*JSON_OBJECT_SIZE(7) + 2*JSON_OBJECT_SIZE(8) + 720;
DynamicJsonBuffer jsonBuffer(bufferSize);

```

```

// знаходження необхідних даних у JSON дереві

```

```

JsonObject& root = jsonBuffer.parseObject(jsonString);
if (!root.success()) {
    Serial.println("parseObject() failed");
    return;
}

```

```

JsonArray& list = root["list"];
JsonObject& nowT = list[0];
JsonObject& later = list[1];

```

```

// включаючи температуру та вологість для розширення проєкту

```

```

String city = root["city"]["name"];

```

```

float tempNow = nowT["main"]["temp"];
float humidityNow = nowT["main"]["humidity"];
String weatherNow = nowT["weather"][0]["description"];

```

```

float tempLater = later["main"]["temp"];
float humidityLater = later["main"]["humidity"];
String weatherLater = later["weather"][0]["description"];

```

```
// перевірка шости основних погодніх можливостей
```

```
diffDataAction(weatherNow, weatherLater, "clear");
diffDataAction(weatherNow, weatherLater, "rain");
diffDataAction(weatherNow, weatherLater, "snow");
diffDataAction(weatherNow, weatherLater, "cloudy");
```

```
Serial.println();
```

```
}
```

```
//вивід отриманих даних
```

```
void diffDataAction(String nowT, String later, String weatherType) {
    int indexNow = nowT.indexOf(weatherType);
    int indexLater = later.indexOf(weatherType);
```

```
// якщо тип погоди = дощ, якщо поточна погода не містить типу погоди, а
пізніше повідомлення містить, надішліть повідомлення
```

```
if (weatherType == "rain") {
    if (indexNow == -1 && indexLater != -1)
    {
        digitalWrite(redLed, LOW);
        digitalWrite(greenLed, LOW);
        digitalWrite(blueLed, HIGH);
        Serial.println("It is going to " + weatherType + " later! Try " + later);
    }
}
```

```
// хмарно
```

```
else if (weatherType == "snow") {
    if (indexNow == -1 && indexLater != -1) {
        digitalWrite(blueLed,HIGH);
        digitalWrite(greenLed, HIGH);
        digitalWrite(redLed, HIGH);
        Serial.println("It is going to " + weatherType + " later! Try " + later);
    }

}
```

```
// злива
```

```
else if (weatherType == "cloudy") {
    if (indexNow == -1 && indexLater != -1) {
        digitalWrite(greenLed, HIGH);
        digitalWrite(redLed, HIGH);
        digitalWrite(blueLed,LOW);
        Serial.println("It is going to " + weatherType + " later! Try " + later);
    }

}
```

```
// чисте небо
```

```
else
{
    if (indexNow == -1 && indexLater != -1) {
        Serial.println("It is going to be sunny later! Try it after " + later);
        digitalWrite(greenLed,HIGH);
    }
}
```

```
    digitalWrite(redLed,LOW);  
    digitalWrite(blueLed,LOW);  
  }  
}  
}
```

ДОДАТОК Б
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Дипломна кваліфікаційна робота бакалавра на тему:

Розробка програмного забезпечення для
автоматизованого збирання даних з Інтернет. Software
Development for Automated Data Collection in the
Internet

Виконала студентка
групи КНТ-227
Керівник:
к.т.н., доц.

Осипович О.О.

Гладкова О.М.

Рисунок Б.1 – Титульний слайд

Актуальність

Засіб	Переваги	Недоліки
Мобільні застосунки та віджети	Безкоштовно. Існує можливість відстежувати погоду на декілька днів/місяць вперед.	Затрати ресурсів девайсу. Застосунки супроводжуються рекламою, відключення якої коштує грошей. Затрати часу користувача, через додаткове відволікання на інші речі в смартфоні.
Персональні станції	Дешеве апаратне рішення у порівнянні з синоптичною станцією. Є можливість повторного використання.	Необхідність встановлювати додаткову конструкцію з різними сенсорами та датчиками для зчитування кліматичних умов, крім того персональні метеостанції можуть давати похибки якщо сенсори неякісні або біли неправильно налаштовані.
Синоптичні станції	Точність вимірювання.	Такі станції коштовні та їх неможливо встановити без офіційного дозволу держави. Існує необхідність налаштування складної системи.

Рисунок Б.2 – Актуальність

— Мета та завдання —

Об’єкт дослідження - програмні засоби для отримання даних з Інтернет.

Предмет дослідження - вбудована система для збирання та відображення даних з Інтернет.

Мета роботи - розробити програмно-апаратний комплекс, який дозволить збирати дані про погоду в Інтернет та відображати їх користувачу у реальному часі.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- ▶ провести дослідження існуючих інструментів для побудови засобу моніторингу погоди;
- ▶ розробити архітектуру системи;
- ▶ спроектувати апаратну частину системи;
- ▶ розробити програмне забезпечення.

3

Рисунок Б.3 – Мета та завдання

— Порівняльне дослідження платформ —

	NodeMCU ESP32	Arduino Nano	Arduino Uno Wi-Fi
Мікроконтролер	ESP32	ATmega328p	ATmega4809
Енергоспоживання	3.3V	5V	5V
Потужність	2.3-3.6V	7-12V	7-12V
Споживання струму	20-240mA	19-180mA	50-150mA
Оперативна пам'ять	520KB	2KB	6KB
Wi-Fi	так	ні	так
Ціна	130 грн	270 грн	1300 грн

4

Рисунок Б.4 – Порівняльне дослідження платформ

— Обґрунтування вибору середовища розробки —

Перевагами середовища Arduino IDE є:

- ▶ підсвічування коду;
- ▶ швидка заливка скетчу в платформу;
- ▶ можливо зробити простий проєкт за кілька хвилин;
- ▶ високорівнева мова програмування;
- ▶ безліч корисних вбудованих функцій, які прискорюють розробку;
- ▶ безліч бібліотек;
- ▶ велика інтернет спільнота.

5

Рисунок Б.5 – Обґрунтування вибору середовища розробки

— Вибір API для отримання даних про погоду —

- ▶ Gismeteo API (недолік - дуже складна документація)
- ▶ OpenWeather API (простий у використанні)

6

Рисунок Б.6 – Вибір API

— Загальна архітектура системи —

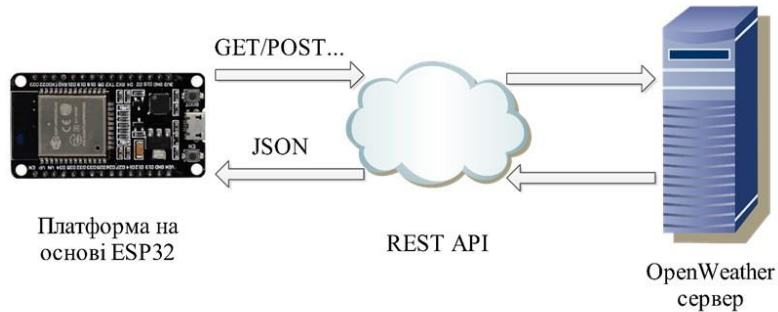


Рисунок Б.7 – Загальна архітектура системи

— Діаграма алгоритму роботи системи —

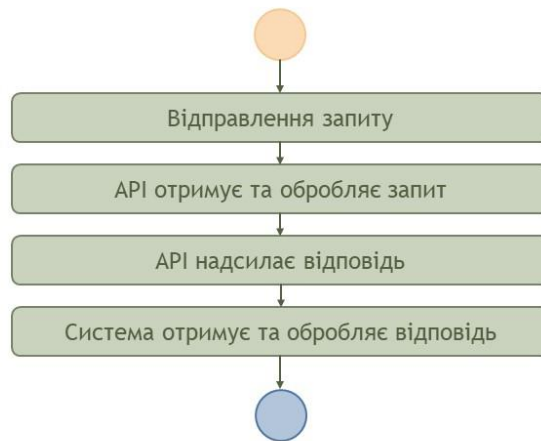


Рисунок Б.8 – Діаграма алгоритму роботи системи

— Схема електрична принципова прототипу системи —

"description"	Колір RGB світлодіоду
Чисте небо	Зелений
Малохмарно	Жовтий
Хмарно з проясненнями	Жовтий
Зливовий дощ	Синій
Дощ	Синій
Сніг	Білий

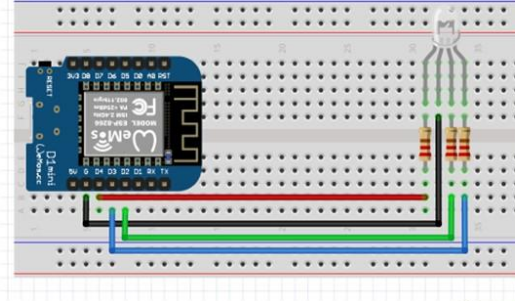


Рисунок Б.9 – Схема електрична принципова

— Діаграма послідовності виконання програми —



Рисунок Б.10 – Діаграма послідовності виконання програми

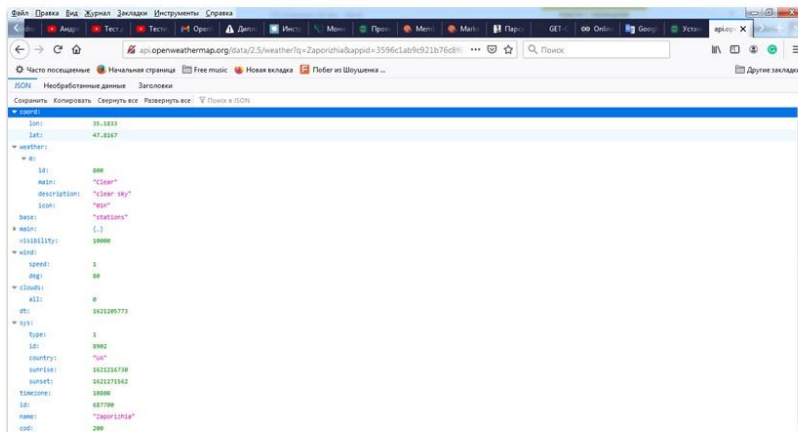
Опис основних функцій та деяких змінних

Назва	Опис
printWiFiStatus()	Визначення та вивід статусу з'єднання
makeHttpRequest()	Секретний ключ, для налаштування безпеки системи.
nameOfCity	Ім'я міста
apiKey	Секретний ключ, для використання в запити для доступу до сервісу
client.read()	Зчитування відповіді від сервера
parseJson()	Розбір тексту рядка у функції parseJson
void parseJson ()	Для синтаксичного аналізу отриманих даних JSON
diffDataAction()	Вивід отриманих даних від сервісу, шляхом загоряння того чи іншого кольору світлодіода

11

Рисунок Б.11 – Опис основних функцій та змінних

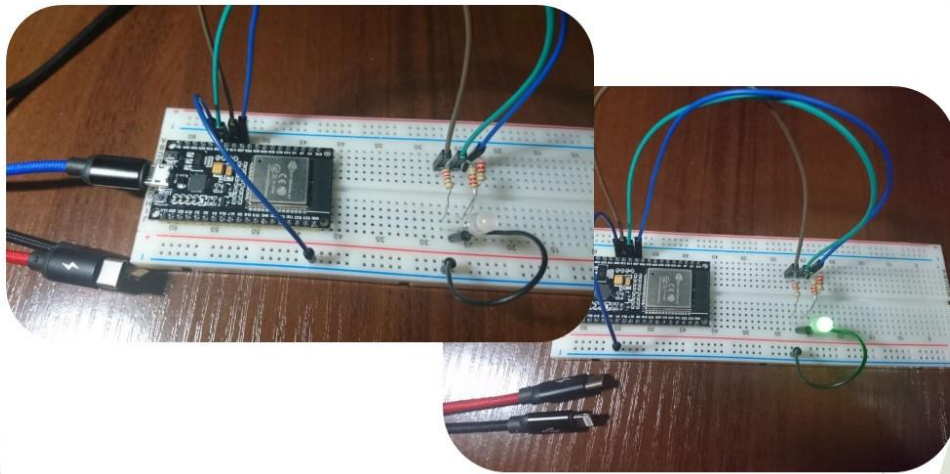
Відповідь від серверу OpenWeather стосовно погоди у місті Запоріжжя



12

Рисунок Б.12 – JSON відповідь від серверу OpenWeather

Прототип розробленої системи



13

Рисунок Б.13 – Прототип системи

Висновки

В ході розробки дипломної кваліфікаційної роботи бакалавра були:

- ▶ проведені дослідження існуючих інструментальних засобів для зчитування та відображення синоптичної картини в заданому регіоні;
- ▶ проведено аналіз та обґрунтування вибору інструментальних засобів для розробки програмно апаратного комплексу для відображення погоди в заданій місцевості;
- ▶ спроектовано загальну архітектуру системи;
- ▶ прототип графічного інтерфейсу який представляє собою RGB світлодіод, який відповідно до отриманих результатів погоди загоряється відповідним кольором;
- ▶ була розроблена схема електрична принципова на основі якої спроектовано фізичний прототип системи;
- ▶ наведено алгоритми роботи системи в цілому та розробленої програми.

14

Рисунок Б.14 – Висновки