НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)

"___" червня 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та компоненти»

зі спеціальності **123 «Комп'ютерна інженерія»**

на тему: «Мобільний додаток для моніторингу та прогнозування погодних умов у реальному часі»

Виконав: студент IV курсу, групи КВ-61
(шифр групи)

Штогрин Павло Петрович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент каф. СПіСКС, к. т. н., с.н.с. Боярінова Ю.Є. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»

Освітньо-професійна програма «Комп’ютерні системи та компоненти»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» червня 20___р.

ЗАВДАННЯ
на дипломний проєкт студента
Штогрина Павла Петровича

1. Тема проєкту «Мобільний додаток для моніторингу та прогнозування погодних умов у реальному часі», керівник проєкту Боярінова Юлія Євгенівна, доцент кафедри СПіСКС, кандидат технічних наук, старший науковий співробітник затверджені наказом по університету від «25» травня 2020 р. №1181-С

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту див. Технічне завдання.

4. Зміст пояснювальної записки: аналіз існуючих рішень та обґрунтування теми, опис розробки додатку для операційної системи Android, опис розробки серверної частини та апаратної частини, опис результатів роботи кінцевої системи та її тестування.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо): структурна схема зв’язку модулів пристрою, схема алгоритму прогнозування погоди, структурна схема всієї системи, схема алгоритму обробки повідомлення з пристрою.

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доц. каф. СПСКС, к.т.н.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Видача завдання на дипломне проєктування.	12.11.2019	
2	Вивчення літератури за тематикою роботи.	20.11.2019	
3	Розробка та узгодження технічного завдання.	25.11.2019	
4	Розробка структури додатку.	15.12.2019	
5	Проєктування апаратної частини.	22.12.2019	
6	Збірка та програмування пристрою.	03.01.2020	
7	Розробка дизайну та графічних елементів додатку.	18.01.2020	
8	Програмна реалізація додатку.	02.02.2020	
9	Розробка серверної частини.	05.03.2020	
10	Налагодження взаємодії між пристроєм, додатком та сервером.	15.03.2020	
11	Тестування додатку.	19.03.2020	
12	Підготовка матеріалів текстової для проєкту.	12.04.2020	
13	Оформлення технічної документації проєкту.	10.05.2020	
14	Попередній захист дипломного проєкту.	20.05.2020	

Студент _____ Павло ШТОГРИН
(підпис) (ініціали, прізвище)

Керівник проєкту _____ Юлія БОЯРІНОВА
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (74 стор., 47 рис.).

У бакалаврському проєкті реалізовано систему для моніторингу та прогнозування погодних умов у реальному часі. Система складається із пристрою для зчитування та передачі через bluetooth даних про погоду, серверної частини для обробки та передачі даних із погодних сервісів у мережі Internet, а також мобільного додатку для прийому, обробки та відображення інформації, отриманої із пристрою та сервера.

Метою проєкту є створення пристрою, який міг би передавати дані про погодні умови безпосередньо у смартфон, а також мобільного додатку із зручним інтерфейсом, який міг би приймати, обробляти та відображати ці дані.

У цьому проєкті було розроблено такі компоненти:

- апаратний засіб на основі платформи Arduino, датчика та bluetooth-передавача;
- сервер, створений мовою програмування Python на базі мікрофреймворку Flask та із використанням REST-архітектури;
- мобільний додаток, створений мовою програмування Java для пристроїв з операційною системою Android;

Результатом розробки є апаратний та програмний продукти, які дозволяють зручно відстежувати поточні погодні умови, а також формують прогноз для конкретної місцевості. Додаток має простий та зрозумілий інтерфейс, мінімальні системні вимоги (пристрій із операційною системою Android версії 4.4 чи вище, bluetooth-модуль та доступ до мережі Internet). Режим моніторингу може працювати без доступу до мережі Internet.

Ключові слова: мобільний додаток, Java, Android, Bluetooth, Python, Flask, REST-інтерфейс, клієнт-серверний додаток, Arduino nano, мікроконтролер, інтернет речей, прогнозування погоди.

ANNOTATION

Qualification work includes an explanatory note (74 p., 47 pic.).

The bachelor's project implements a system for monitoring and forecasting weather conditions in real time. The system consists of a device for reading and transmitting weather data via bluetooth, a server part for processing and transmitting data from weather services on the Internet, and a mobile application for receiving, processing and displaying information received from the device and server.

The aim of the project is to create a device that could transmit weather data directly to a smartphone and a mobile application with a user-friendly interface that could receive, process and display this data.

The following components were developed in this project:

- device based on Arduino platform, sensor and bluetooth transmitter;
- a server which was created in the Python programming language, based on the Flask microframework and using the REST architecture;
- mobile application created in the Java programming language for devices with the Android operating system;

The result of the development is hardware and software products that allow conveniently track current weather conditions and form a forecast for a specific area. The application has a simple and clear interface, minimum system requirements (device with Android operating system version 4.4 or higher, bluetooth module and Internet access). The monitoring mode can work without Internet access.

Keywords: mobile application, Java, Android, Bluetooth, Python, Flask, REST-interface, client-server application, Arduino nano, microcontroller, Internet of Things, weather forecasting.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
1	А4	ІАЛЦ.467200.002 ТЗ	Мобільний додаток для моніторингу та прогнозування погодних умов у реальному часі Технічне завдання	4		
2	А4	ІАЛЦ.467200.003 ТП	Мобільний додаток для моніторингу та прогнозування погодних умов у реальному часі Відомість технічного проєкту	1		
3	А4	ІАЛЦ.467200.004 ПЗ	Мобільний додаток для моніторингу та прогнозування погодних умов у реальному часі Пояснювальна записка	74		
4	А4	ІАЛЦ.467200.005 Д1	Зв'язок модулів пристрою Схема структурна	1		
5	А4	ІАЛЦ.467200.006 Д2	Алгоритм прогнозування погоди Схема алгоритму	1		

Зм	Лист	№ докум.	Підп	Дата	**ІАЛЦ.467200.001 ОА**		
Розроб.		Штогрин П.П.			*Мобільний додаток для моніторингу та прогнозування погодних умов у реальному часі.*	Літ.	Лист 1 / Листів 2
Перев.		Боярінова Ю.Є.					
Н. контр.		Клятченко Я.М.			**Опис альбому**	КПІ ім. Ігоря Сікорського, ФПМ, КВ-61	
Затв.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
6	А4	ІАЛЦ.467200.007 Д3	Загальна схема системи	1		
			Схема структурна			
7	А4	ІАЛЦ.467200.008 Д4	Обробка повідомлення з	1		
			пристрою			
			Схема алгоритму			
8		Диск CD-ROM	Текст ПЗ. Тексти програм	1		
			Графічний матеріал			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Мобільний додаток для моніторингу та прогнозування погодних умов у реальному часі».

Галузь застосування: використання в повсякденному житті для перегляду поточних погодних умов та прогнозу погоди.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є створення системи для моніторингу та прогнозування погодних умов за допомогою мобільного додатка та вимірювального пристрою. Призначення роботи полягає у розробці кінцевого продукту та створення комфортних умов для планування діяльності, яка залежить від погоди.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації в періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

1.1 Вимоги до продукту, що розробляється

- сумісність з операційною системою Android;
- наявність апаратної частини, яка збирає та передає через бездротовий канал зв'язку дані про погодні умови;
- автономне живлення апаратної частини;
- наявність серверної частини, що збирає дані із погодних сайтів та передає у додаток;
- можливість моніторингу погодних умов у реальному часі;
- можливість прогнозування погоди на основі зібраних даних;
- інтуїтивно зрозумілий інтерфейс;

5.2 Вимоги до програмного та апаратного забезпечення користувача

- операційна система Android не нижче 4.4 версії;
- наявність доступу до мережі Wi-Fi, або GPRS/EDGE/3G/4G;
- смартфон, що підтримує Bluetooth не нижче версії 2.0.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Видача завдання на дипломне проектування	12.11.2019
2.	Вивчення літератури за тематикою роботи	20.11.2019
3.	Розроблення та узгодження технічного завдання	25.11.2019
4.	Розроблення структури додатку	15.12.2019
5.	Проєктування апаратної частини	22.12.2019
6.	Збірка та програмування пристрою	03.01.2020
7.	Розроблення дизайну та графічних елементів додатку	18.01.2020
8.	Програмна реалізація додатку	02.02.2020
9.	Розробка серверної частини	05.03.2020
10.	Налагодження взаємодії між пристроєм, додатком та сервером	15.03.2020
11.	Тестування додатку	19.03.2020
12.	Виправлення помилок, виявлених під час тестів	29.03.2020
13.	Підготовка матеріалів текстової частини проєкту	12.04.2020
14.	Підготовка матеріалів графічної частини проєкту	05.05.2020
15.	Оформлення технічної документації проєкту	10.05.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.004 ПЗ	Мобільний додаток для моніторингу та прогнозування погодних умов у реальному часі. Пояснювальна записка	74		
	А4	ІАЛЦ.467200.005 Д1	Зв'язок модулів пристрою. Схема структурна	1		
	А4	ІАЛЦ.467200.006 Д2	Алгоритм прогнозування погоди. Схема алгоритму	1		
	А4	ІАЛЦ.467200.007 Д3	Загальна схема системи. Схема структурна	1		
	А4	ІАЛЦ.467200.008 Д4	Обробка повідомлення з пристрою. Схема алгоритму	1		
		Диск CD-ROM	Текст ПЗ. Тексти програм. Графічний матеріал	1		

Зм.	Лист	№ докум.	Підп.	Дата	**ІАЛЦ.467200.003 ТП**		
Розроб.		Штогрин П.П.			*Мобільний додаток для моніторингу та прогнозування погодних умов у реальному часі.* **Відомість технічного проєкту**	Літ.	Лист 1 / Листів 1
Перев.		Боярінова Ю.Є.					
Консульт						**КПІ ім. Ігоря Сікорського ФПМ, КВ-61**	
Н. контр.		Клятченко Я.М.					
Зав. каф.		Романкевич В.О					

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ОС – операційна система

SDK(software development kit) – набір із засобів розробки, утиліт і документації, який дозволяє програмістам створювати прикладні програми за визначеною технологією або для певної платформи.

XML(Extensible Markup Language) – запропонований консорціумом World Wide Web Consortium стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками.

APK(Android Package) – формат файлів, які містять скомпільований Android - додаток та всі необхідні ресурси для нього

API (Application Programming Interface) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

HTML (HyperText Markup Language) – мова тегів, якою пишуться гіпертекстові документи для мережі Інтернет.

GPS (Global Positioning System) – сукупність радіоелектронних засобів, що дозволяє визначати положення та швидкість руху об'єкта на поверхні Землі або в атмосфері.

UUID (Universally Unique Identifier) – стандарт ідентифікації, який використовується при створенні програмного забезпечення, для унікальної ідентифікації інформації без центру координації у розподілених системах.

JSON (JavaScript Object Notation)– текстовий формат обміну даними між комп'ютерами.

URL (Uniform Resource Locator) — система уніфікованих адрес електронних ресурсів, що використовується у мережі Internet.

HTTP (Hyper Text Transfer Protocol) – протокол передачі даних, що використовується в комп'ютерних мережах.

HTTPS (HTTP Secure) – протокол, зо базується на HTTP. Використання https: URL вказує, що протокол HTTP має використовуватися, але з іншим

портом за замовчуванням (443) і додатковим шаром шифрування/аутентифікації між HTTP і TCP.

DNS-сервер (Domain Name System)– ієрархічна розподілена система перетворення імені хоста.

Git – розподілена система керування версіями файлів та спільної роботи.

UART (Universal Asynchronous Receiver/Transmitter) – вузол обчислювальних пристроїв, призначений для організації зв'язку з іншими цифровими пристроями. Перетворює передані дані в послідовний вид так, щоб було можливо передати їх по одній фізичній цифровий лінії іншому аналогічному пристрою.

USB (Universal Serial Bus) – шина, призначена для з'єднання комп'ютерів і периферійних пристроїв.

FTDI (Future Technology Devices International) – шотландська компанія, яка розробляє напівпровідникові пристрої.

COM-порт(Communications port) – назва інтерфейсу RS-232 для зв’язку між пристоями. Інформація через порт передається по одному біту.

IDE (Integrated Development Environment) – комплексне програмне рішення для розробки програмного забезпечення. Зазвичай, складається з редактора початкового коду, інструментів для автоматизації складання та відлагодження програм.

ВСТУП

Погода – це те, з чим ми стикаємось кожного дня. Від неї залежить не лише те, як ми одягнемся, чи як заплануємо відпочинок, але і те як будемо почувати себе впродовж дня.

З давніх часів люди цікавились погодою, проводили спостереження, на основі яких потім складали прикмети. Ці прикмети можна назвати першими спробами прогнозування погоди. Надійність такого прогнозу не дуже висока, але все ж це давало хоча б уявлення про погодні умови у наступні кілька днів.

Із розвитком та поширенням писемних джерел інформації багато вчених стали записувати свої спостереження і на основі цих статистичних даних робити більш точні прогнози. З роками наука розвивалась, вимірювальні прилади вдосконалювались. Це дало змогу розуміти як працює погода і розробити алгоритми, на основі яких можна було робити прогнози із великою точністю. Також і звичайні люди після прочитання певної літератури та проведення спостережень за погодою отримали змогу передбачати погоду на невеликий проміжок часу.

Зараз, коли цифрові обчислювальні пристрої стали доступними кожному можна автоматизувати процес моніторингу та прогнозування погоди. Великі приватні чи державні підприємства мають багато ресурсів для збору необхідних даних – велику кількість точок спостереження, дані із супутників, літаків, радарів. Такі засоби можуть дати майже безпомилковий прогноз. Але звичайні люди теж мають можливість отримувати прогноз саме для їхньої місцевості за допомогою електронних датчиків та мікроконтролерів. Такі пристрої, звісно, не можуть забезпечити абсолютну точність, оскільки не мають усіх необхідних даних, проте зробити прогноз на добу вперед цілком здатні.

Оскільки зараз майже кожен має смартфон, то доцільно використовувати його обчислювальні потужності для формування прогнозу. Також у смартфоні зручно переглядати інформацію, оскільки він завжди під рукою. Таким чином

у цьому проєкті буде реалізовано саме мобільний додаток для перегляду та прогнозування погоди, а також апаратна частина на мікроконтролері для збору необхідних даних.

1. АНАЛІЗ ІСНУЮЧИХ ЗАСОБІВ МОНІТОРИНГУ ПОГОДНИХ УМОВ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1. Призначення систем передбачення та моніторингу погодних умов

Погодні умови впливають на велику кількість сфер сучасного життя. Наприклад при надто густому тумані, ожеледиці, чи бурі, що наближається небажаними є зліт та посадка повітряного транспорту. Також дуже важливим передбачення погоди є у сфері сільського господарства, оскільки у зв’язку із непередбачуваною грозою, заморозком, чи посухою можна отримати великі втрати як при посіві рослин, так і при зборі врожаю. Однак для потреб авіації чи сільського господарства існують спеціалізовані системи та служби.

Для звичайних людей також існують погодні сервіси, проте вони відображають усереднену інформацію для певного регіону. Реальні дані можуть дещо відрізнятись, особливо для населених пунктів, що віддалені від великих міст. Також дані можуть відрізнятися на різних сайтах, що спричиняє плутанину. Для усунення цих недоліків було б зручно отримувати дані про умови, які зараз є за вікном та на основі цих даних скласти прогноз на найближчі години для зручного планування дій як для роботи так і для відпочинку.

Зараз, як і в минулому багато людей використовують вимірювальні прилади, що не потребують електроніки – спиртові, ртутні термометри; ртутні барометри, чи анероїди. Проте набагато зручнішими є цифрові вимірювальні прилади, оскільки вони є компактнішими, часто точнішими, а також при використанні їх у парі із пристроями зв’язку(дротовими чи бездротовими) вони є набагато зручнішими, оскільки збирати та обробляти дані з них можна дистанційно та автоматизовано.

1.2 Огляд систем-аналогів

У зв’язку із стрімким розвитком електроніки останнім часом багато побутових пристроїв стають цифровими. Так сталось і з термометрами та барометрами, які стають все популярнішими, особливо у поєднанні із системами розумного дому.

На ринку існує декілька основних типів метеостанцій. Бюджетні рішення пропонує бренд EA2. Наприклад модель EA2 OT300 (рисунок 1.1) має монохромний дисплей, який розділений на дві секції – температура в приміщенні та на вулиці.

Рисунок 1.1 – Пристрій EA2 OT300

Датчик температури на вулиці – дротовий, кріпиться до самого пристрою. Ця модель може вимірювати лише температуру, також є функція фіксації мінімальної та максимальної температури. [1]

Також у цього виробника є пристрої із більшим функціоналом, наприклад EA BL503 SLIM (рисунок 1.2).

Рисунок 1.2 – EA BL503 SLIM

Ця модель може вимірювати температуру та вологість у приміщенні, а також температуру, вологість і тиск на вулиці. У ній є функція передбачення погоди на найближчі дні, а також передбачення динаміки зміни погоди (погіршення, покращення, без змін). Зовнішній датчик бездротовий, працює на частоті 433 МГц, що дозволяє отримати досить великий радіус дії, проте прийняти такий сигнал може лише пристрій, що працює на такій же частоті – головний модуль. Смартфон таку інформацію прийняти не може. [2]

Популярними також є пристрої німецької компанії TFA. Вона є одним з лідерів на європейському ринку метеоприладів. Модель TFA HORIZON (рисунок 1.3) обладнана бездротовим датчиком та головним модулем із екраном.

Рисунок 1.3 – TFA HORIZON

Датчик може вимірювати температуру, вологість та атмосферний тиск, сам пристрій робить передбачення погоди, показує гістограму зміни тиску, показувати час сходу та заходу місяця, а також фази місяця. Можливе одночасне підключення трьох датчиків. Пристрій має функцію будильника, може автоматично встановлювати час через радіосигнал DCF77 – інформаційний сигнал, призначений для передачі точного часу. Він транслюється кожну хвилину із 0 по 58 секунду із Майнфдінгена, що у Німеччині. Максимальна відстань розповсюдження сигналу – 1900 кілометрів вдень і 2100 вночі. Більша частина території України входить у цей радіус. [3]

TFA WeatherHub (рисунок 1.4) може бути двох комплектаціях: лише бездротовий датчик та хаб, що транслює інформацію на мобільний додаток, чи датчик, хаб та головний модуль з екраном.

Рисунок 1.4 – TFA WeatherHub

У додатку можна отримувати лише дані про температуру, а також налаштувати сповіщення, коли вона досягне певного значення. Одночасно можна підключити до 50 датчиків. Головний модуль отримує із датчиків інформацію не лише про температуру, а і про тиск та вологість. Решта функціоналу ідентична попередній моделі.

Наступна модель, яка варта розгляду – TFA Spring Breeze (рисунок 1.5). Вона обладнана бездротовими датчиками температури, вологості та тиску, може робити прогноз погоди, реєструє граничні показники. Присутня функція «Feels like» – як температура відчувається, також відображається точка роси та температура охолодження вітром.

Рисунок 1.5 – TFA Spring Breeze

Основна відмінність цієї моделі від усіх попередніх – наявність датчика вітру. Це дозволяє робити точніший прогноз погоди, а також отримувати більш інформативну картину погодних умов у реальному часі.

У цього приладу є і недолік – датчик вітру повинен бути встановлений на висоті 3 метри над будь-якими іншими об’єктами, що його оточують. Через це для моделі Spring Breeze в умовах великого міста функціонал зменшується лише до вимірювання температури, вологості та тиску. [3]

TFA WeatherHub Observer передає дані про температуру із датчика на хаб, а звідти – на веб-сервер, чи у мобільний додаток. Відповідно у веб-застосунку можна проводити моніторинг даних, формувати діаграми. Отримані дані у графічному чи текстовому поданні можуть бути записані у файл.

На ринку також присутні моделі, які майже збирають майже усі дані про погодні умови – не лише температуру, вологість та тиск, але і швидкість, напрям і температуру вітру, кількість опадів. До них відноситься модельTFA Sinus (рисунок 1.6).

Рисунок 1.6 – TFA Sinus

Цей прилад також може передавати дані про температуру, вологість та параметри вітру у мобільний додаток. Як і у попередній моделі, прилад вимірювання сили вітру має бути розміщеними на висоті 3 метри над іншими об’єктами.

Є також аналогічні модулі, які можна підключити до моделі WeatherHub. У цьому випадку модуль вимірювання сили та напряму вітру обладнаний сонячною батареєю. [3]

Усі вищезазначені станції, а також їхні бездротові модулі живляться від гальванічних батарей типу AAA чи AA. Деякі пристрої із великими дисплеями, які вимагають більше енергії мають можливість також підключатись до електромережі.

1.3 Обґрунтування теми дипломного проєкту

Вищевказані системи хоч і є досить функціональними, проте мають ряд значних недоліків:

- наявність дисплея, що не є необхідним у сучасних реаліях та здорожчує конструкцію (особливо у моделях із великими кольоровими дисплеями);
- елементи живлення, що не перезаряджаються, а отже є менш екологічними, зручними та більш дорогими у використанні;
- налаштування лише за допомогою кнопок, що зазвичай вимагає знімати пристрій із місця кріплення;
- функціонал обмежений лише заводськими характеристиками;
- відсутність мобільних додатків для моніторингу даних, чи дуже малий функціонал таких додатків;
- пристрої із широким функціоналом мають дуже високу вартість.

Пристрій та мобільний додаток, які були розроблені у моєму проєкті позбавлені цих недоліків. Відсутність дисплея можна пояснити тим, що майже кожна людина має смартфон, який має не гірший дисплей, а також завжди під рукою. Хоча більшість описаних вище пристроїв мають бездротові головні модулі, але зазвичай вони стоять на місці, бо носити їх із собою незручно та недоречно, на відміну від смартфона.

У пристрої встановлений літій-іонний акумулятор, який може заряджатися напряму у пристрої, чи бути замінений іншим акумулятором на час зарядки.

Усі необхідні налаштування (зміна одиниць вимірювання, прив’язка до іншого модуля вимірювання, тощо) можуть бути здійснені безпосередньо у додатку. Також є можливість додавання нового функціоналу без необхідності

купляти нову модель, оскільки модуль з датчиком лише надсилає інформацію, а її обробка відбувається у додатку, який легко оновити.

Висновки до розділу

У цьому розділі описано важливість систем моніторингу та передбачення погодних умов. Також були розглянуті системи-аналоги, що існують на ринку. Таким чином системи для моніторингу та передбачення погодних умов у реальному часі є дуже важливими як для підприємств так і для звичайних людей. На ринку існує велика кількість таких систем для побутового застосування. У залежності від ціни функціонал цих пристроїв може бути дуже широким. Вони можуть виводити дані про поточні погодні умови, час, дату на екран, фіксувати мінімальні та максимальні значення, повідомляти про негоду. Деякі системи мають мобільні додатки, але функціонал цих додатків дуже обмежений. Також існують системи для вимірювання не лише температури, тиску та вологості, а ще і сили, напрямку та температури вітру, кількості опадів, тощо. Попри усі свої переваги вищевказані системи мають свої недоліки – відсутність мобільного додатку, або додаток із малим функціоналом, використання елементів живлення, що не перезаряджаються, висока ціна для систем із функцією прогнозування погоди, неможливість додавання нових функцій у систему без покупки нового пристрою.

У пристрої, описаному у цьому проєкті ці та інші недоліки відсутні або мінімізовані.

2. ОПИС МОБІЛЬНОГО ДОДАТКУ

2.1 Аналіз технологій для розробки додатків

Спочатку був проведений аналіз існуючих мобільних ОС та їх популярності серед користувачів. Значну частину ринку (74.3% станом на січень 2020 року) становить ОС Android, 24.6% ринку становить iOS. Оскільки інші ОС складають менше 1% від усіх наявних, то надалі вони розглядатися не будуть.

Для розробки мобільних додатків існують кросплатформні рішення. Прикладом такого є Thunkable X – середовище програмування мобільних додатків, яке дозволяє будувати програму за допомогою блоків, які представляють спрощене візуальне зображення класів, функцій, змінних та основних конструкцій мов програмування (рисунок 2.1).

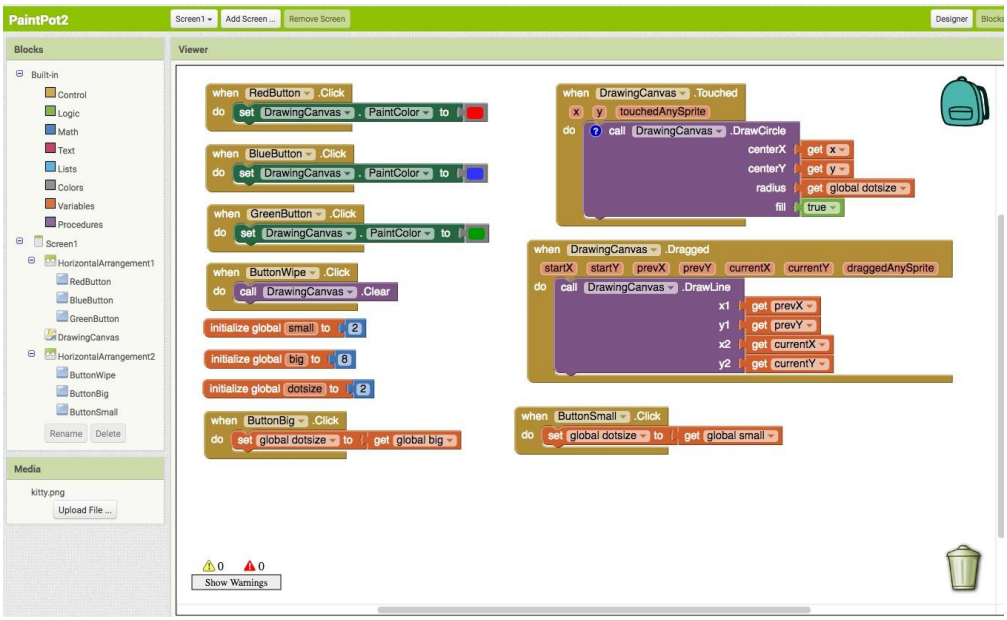


Рисунок 2.1 – Сервіс Thunkable

Платформа Thunkable розроблена на базі MIT App Inventor 2 – середовища для складання програм за допомогою візуальних блоків. Це середовище було розроблене працівниками компанії Google, але через певний

час корпорація передала його центру мобільної освіти Массачусетського технологічного інституту, де згодом за допомогою одного із розробників та кількох професорів інституту була реалізована версія App Inventor 2.

Хоча це середовище має перевагу у вигляді кросплатформності та простоти розуміння, але ця спрощеність створює безліч незручностей, яких можна уникнути, використовуючи мову програмування.

Набагато більш потужним інструментом кросплатформної розробки додатків є Xamarin – фреймворк, який дозволяє писати додатки мовою C#. Розробники цього продукту запевняють, що він має повний доступ до усіх можливостей "рідного" SDK, а також механізму створення користувацького інтерфейсу. Також розроблений додаток не матиме меншої продуктивності, ніж аналогічний написаний "рідною" мовою програмування.

Способи попередньої компіляції для iOS та Android відрізняються (рисунок 2.2).

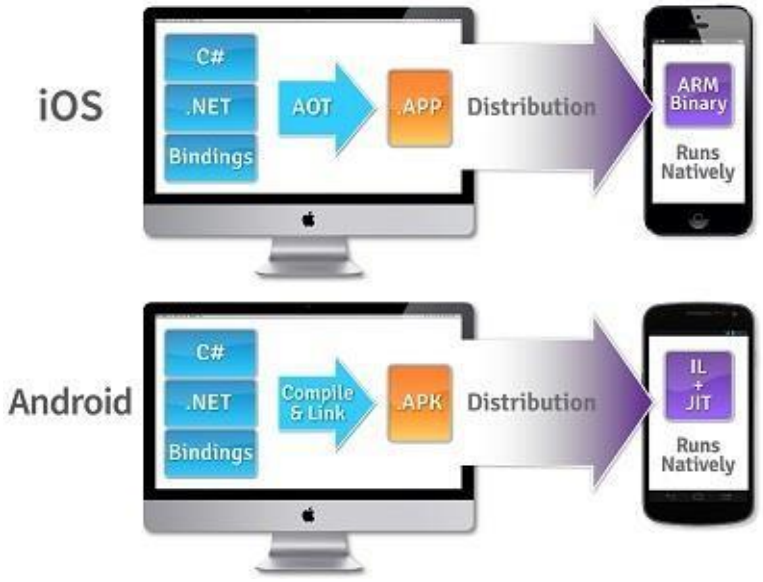


Рисунок 2.2 – Процес створення додатків для різних ОС

У Android використовується віртуальна Java – машина Dalvik. Додаток, який був написаний мовою Java компілюється у певний проміжний байт-код, який

потім інтерпретується у команди процесора в момент виконання програми. Це так звана компіляція на льоту (Just-in-time). У iOS же модель компіляції перед виконанням (Ahead-of-Time). Xamarin враховує ці відмінності і пропонує різні компілятори для різних платформ.

До недоліків Xamarin можна віднести те, що додаток все-таки буде не на 100% кросплатформним, оскільки графічний інтерфейс користувача потрібно розробляти для кожної платформи окремо. Також запропоновані середовища розробки – Xamarin Studio та Visual Studio мають низку проблем – відсутність емулятора у першій та дуже довгий процес відлагодження у другій. Мова C# ще не встигла стати популярною у мобільній розробці, тому при розробці саме нею можна зустрітися із труднощами, чи багами, рішення яких дуже важко знайти. Головним же недоліком Xamarin є те, що безкоштовна ліцензія має обмеження на розмір додатка, а за версії Indie (немає обмежень на розмір, але розробка лише у Xamarin Studio) та Business (немає обмежень, розробка у Xamarin Studio чи Visual Studio) доведеться заплатити 299$ та 999$ відповідно. Це значно підвищує поріг входу для молодих розробників та студентів.

Враховуючи недоліки усіх вищевказаних середовищ, було прийняте рішення розробити додаток лише на одну платформу. І, оскільки найпопулярнішою мобільною ОС зараз є Android, то додаток, описаний у проєкті, буде саме для неї. Мова для написання додатку – Java, оскільки це зараз найпопулярніша мова для Android - розробки, а також є дуже багато документації, курсів, та іншої інформації саме для цієї мови розробки додатків.

ОС Android була розроблена на базі ядра Linux, тому має з ним деякі спільні риси. Наприклад, кожен додаток має свій унікальний ідентифікатор. Після запуску додатків, кожен з них працює у своєму власному процесі, своїй власній віртуальній машині. По мірі необхідності ОС керує запуском чи зупинкою процесів додатків. Кожен додаток працює ізольовано від інших, але може запрошувати доступ до апаратних, чи інших ресурсів системи. Додаток повинен мати файл маніфесту – XML-файл, що містить усі компоненти додатку,

налаштування до них, а також дозволи (наприклад доступ до контактів, внутрішнього сховища, що Bluetooth).

Android - застосунок складається із чотирьох основних компонентів: активності, сервіси, постачальники контенту та приймачі широкомовних повідомлень (broadcast receivers). Основне використання активностей – окремі вікна додатку. Наприклад, у моєму додатку є дві активності – головний екран із інформацією з датчиків та екран даних із погодних сайтів. За організацію активностей відповідає клас System. Для запуску іншої активності достатньо викликати метод startActivity() з об'єктом Intent в якості параметра. У відповідь на це System створить новий об'єкт активності, або повторно використає старий. У Java є важлива функція, яка дозволяє повторно використовувати пам'ять – "збір сміття", її унаслідував і Android.

Сервіси – це компоненти додатку, які можуть виконувати тривалі операції у фоновому режимі і не мають графічного інтерфейсу. Сервіс може мати дві форми запущений та прив'язаний. Перша форма – це коли компонент додатку запускає його викликом startService() і сервіс працює на фоні необмежену кількість часу, навіть коли компонент, що запустив вже знищений. Такі фонові процеси можуть бути застосовані для завантаження чи вивантаження даних у мережу. Після завершення операції сервіс повинен самостійно зупинитись.

Друга форма – прив'язаний сервіс. Він виникає, коли компонент додатку прив'язується викликом bindService(). Цей сервіс передбачає інтерфейс клієнт - сервер, який дозволяє компонентам взаємодіяти з сервісом, відправляти запити, отримувати результати і навіть робити це між різними процесами. Прив'язаний сервіс працює до тих пір, поки до нього прив'язаний інший компонент додатку. Постачальники контенту керують доступом до структурованого набору даних.

Вони інкапсулюють дані і надають механізми забезпечення їх безпеки. Постачальники контенту представляють стандартний інтерфейс для об'єднання даних у одному процесі з кодом, який виконується у іншому процесі.

Приймач широкомовних повідомлень – це компонент, який реагує на повідомлення, що розповсюджуються по всій системі. Більшість із цих повідомлень надсилає сама система – наприклад про вимкнення екрану, низький заряд акумулятора, зроблене фото. Також вони можуть розсилатись і додатками – наприклад повідомлення іншим додаткам, що певний файл завантажений і готовий до роботи. Найчастіше широкомовні приймачі є просто “шлюзом” для інших компонентів і призначені для виконання мінімального об’єму рботи. Наприклад, вони можуть ініціювати виконання службою певних дій при виникненні певної події.[4]

Для розробки додатку було використане середовище Android Studio, яке, хоча з’явилось не так давно, та вже стало набагато зручнішим та функціональнішим для мобільної розробки, ніж середовище Eclipse, що застосовувалось раніше. Розглянемо основні можливості Android Studio:

- інтелектуальний редактор з розширеним автодоповненням, ре факторингом та аналізом коду;
- функція “миттєвий запуск”, яка дозволяє швидко перевіряти зміни, задавати параметри, запускати робочі цикли шляхом введення коду та зміни ресурсів, доступних додатку на пристрої чи в емуляторі;
- швидкий та багатофункціональний емулятор Android з віртуальним акселерометром, магнітометром, вимірювачем температури, проте з Bluetooth емулятор, на жаль, працювати поки не може;
- підтримка усіх платформ Android (смартфонів, планшетів, Android Wear, Android TV;
- гнучка система збірки на основі Gradle із автоматизацією процесу формування коду додатків, керуванням залежностями та конфігураціями APK, що налаштовуються;
- шаблони коду для реалізації стандартних функцій;
- зручний редактор макетів із можливістю перетягування елементів і режимом прототипування;

- менеджер обмеження макетів для розробки великих і складних макетів у однорівневій ієрархії;
- аналізатори коду для виявлення у ньому проблем, пов’язаних із продуктивністю, зручністю користування, можливими витоками пам’яті (memory leaks), чи несумісністю версій;
- підтримка C/C++ у режимі зміни коду і можливість від лагодження із використанням низькорівневого набору команд;
- вбудована підтримка Firebase SDK, Firebase Test Lab, Firebase App Indexing та Google Cloud Platform;
- аналізатор APK - файлів для перегляду долей певних компонентів у файлі;
- модуль запису тестів Espresso для створення тестів інтерфейсу користувача шляхом реєстрації взаємодії користувача з додатком, а також виводу програмного коду тестів;
- інспектор макету для перегляду ієрархії представлень додатку;
- налагоджував графічного процесора для захоплення потоку команд OpenGL ES на пристрої Android, його запуску в Android Studio та подальшого аналізу.[5]

2.2 Структура додатку

Розроблений мобільний додаток складається із трьох java - класів та одинадцяти файлів ресурсів, написаних мовою розмітки XML. Розглянемо структуру та призначення кожного файлу окремо. На файли, автоматично згенеровані середовищем особливої уваги звертати не будемо.

Почнемо з файлів ресурсів. Найважливішим серед них є файл маніфесту – AndroidManifest.xml. Він містить важливу інформацію про додаток, яка потрібна ОС Android. Без неї система не може виконувати жодного коду додатку. Основні функції цього файлу:

- задає ім'я пакету Java для додатку, це ім'я водночас є унікальним ідентифікатором додатку;
- описує компоненти додатку – операції, служби, приймачі широкомовних повідомлень, постачальники контенту;
- містить імена класів, які реалізують кожен компонент і публікує їх можливості (наприклад які повідомлення Intent вони можуть приймати;
- визначає у яких процесах будуть розміщені компоненти додатку;
- оголошує які дозволи мають бути видані додатку, щоб він отримав доступ до захищених частин API – інтерфейсу а також міг взаємодіяти з іншими додатками (наприклад дозволи на користування геолокацією, bluetooth, доступ до контактів, тощо);
- визначає дозволи, необхідні для комунікації із компонентами всередині додатку;
- містить список класів Instrumentation, які при виконанні додатку представляють відомості про профіль та іншу інформацію, вони необхідні для налагодження і видаляються перед публікацією додатку;

– визначає мінімальний рівень API - інтерфейсу Android, якого потребує додаток;
– містить список бібліотек, з якими повинен бути зв’язаний додаток.

Два файли макету – activity_child.xml та activity_main.xml відповідають за зовнішній вигляд двох активностей (вікон) додатку. Структура такого файлу схожа на структуру створення HTML - сторінки. У макеті повинен бути лише один кореневий елемент – об’єкт представлення View чи представлення групи – ViewGroup, всередині нього можна створювати інші об’єкти макету, або віджети, тобто елементи, які безпосередньо буде бачити користувач.

Файли ic_launcher_background.xml та ic_launcher_foreground.xml, а також ic_launcher.xml, ic_launcher_round.xml містять зображення іконки додатку, а також його мініатюри. Ці зображення залишились тими самими, що середовище згенерувало за замовчуванням.

Чотири файли параметрів – colors.xml, dimens.xml, strings.xml та styles.xml містять параметри різних компонентів додатку. Наприклад, перший файл описує кольори, якими будуть забарвлюватись елементи інтерфейсу. Файл dimens описує відступи які, які використовуються при побудові макету активностей. У самому макеті можна задати відступи числом, але зручніше прив’язати їх до певної константи у файлі dimens. Файл strings містить певні текстові константи, що можуть бути використані при побудові макету та у самій java - програмі. Цей файл потрібен щоб щоб уникнути так званого “хардкоду” – числових чи текстових значень безпосередньо у коді (замість винесення цих значень у константи). Кожна константа має параметр name, який є її унікальним ідентифікатором. Файл styles потрібен для того, щоб об’єднувати атрибути елементів, що повторяються у одному стилі. Стиль може бути застосованим як до одного об’єкту View, так і для всієї активності, чи всього додатку.

Файл network_security_config.xml містить у собі дозвіл з’єднуватись до сервісу api.openweathermap.org, який надає інформацію про поточні погодні

умови. Цей дозвіл необхідний, оскільки сервіс використовує http з'єднання (замість надійного https). [6]

Розглянемо java – файли, які відповідають за основний функціонал і логіку роботи додатку. Щоб краще зрозуміти необхідність тих чи інших компонентів розглянемо спочатку життєвий цикл android - додатку (рисунок2.3).

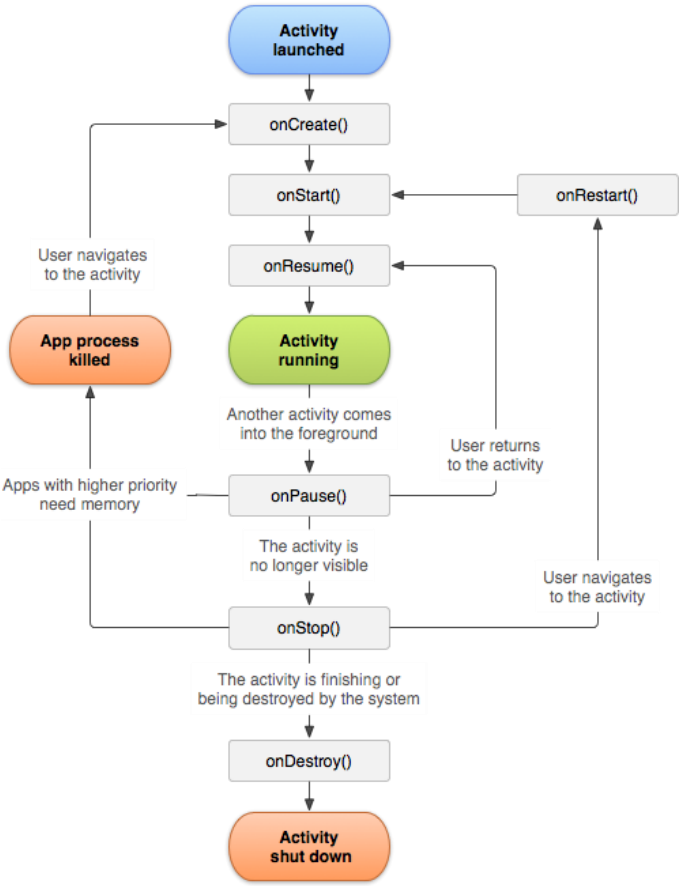


Рисунок 2.3 – Життєвий цикл Android – додатку

З рисунку видно, що життєвий цикл додатку супроводжують виклики певних функцій. Android викликає ці функції автоматично при виникненні певних умов. Деякі функції не обов'язково мають бути описані програмістом, система сама може сама виконати дії за замовчуванням. Обов'язковою є лише функція onCreate() – вона виконується, коли активність входить у створений

(Created) стан. Тут необхідно описати всю логіку роботи програми, чи викликати відповідні функції. Далі викликається onStart(), ця функція робить активність видимою для користувача, починається промальовка інтерфейсу. Цей метод виконується дуже швидко. Далі якщо активність увійшла у стан продовження (Resumed state) викликається метод onResunme(). У цьому стані додаток взаємодіє із користувачем і залишається у фокусі. Цей стан триває до тих пір, поки не станеться щось, що забере фокус із додатку. Це може бути телефонний дзвінок, перемикання користувача на інший додаток, чи вимкнення режиму розділення екрану. Користувач може перезаписати цей метод, щоб описати у ньому дії, які додаток має робити, поки компонент видимий і працює на передньому плані. Коли стається переривання додатку, активність переходить у призупинений (Paused) стан і система викликає метод onPause. Якщо активність повертається із призупиненого стану у стан продовження, onResunme() викликається знову. Саме тому у цьому методі варто ініціалізовувати компоненти, які припинили роботу у стані призупинення, а також провести ініціалізацію усіх компонентів, які можуть її потребувати на стадії продовження.

Метод onPause() викликається як перша індикація того, що користувач покидає активність (але це не завжди означає, що активність буде знищена). Це означає, що активність більше не на передньому плані. У цьому методі бажано припиняти роботу модулів програми, які потрібні лише для взаємодії з користувачем, або ті, які значно впливають на розряд батареї (наприклад GPS). Хоча призупинений додаток не у фокусі, проте він може залишатись видимим у багатовіконному режимі (у Android 7.0 та вище), тому повністю відключати ресурси, що відповідають за інтерфейс користувача варто лише у методі onStop(). Активність переходить у зупинений (Stopped) стан, коли вона більше не видима для користувача. Це спричиняє виклик методу onStop(). Це може відбуватись коли, наприклад нова активність повністю займає екран, або активність буде знищуватись. Окрім зазначеної вище рекомендації

використання onStop() також у ньому варто проводити роботу із відносно важкими завданнями для процесора (наприклад якщо немає підходящої можливості зберегти інформацію у базу даних, можна зробити це у onStop()). Метод onDestroy() викликається для знищення активності – звільнення усіх ресурсів. Не рекомендовано розміщувати у ньому якийсь код, оскільки немає гарантії, що він буде виконаний. [7]

Почнемо огляд програми із файлу MainActivity.java. Перший метод у цьому класі – onCreate(). У ньому ініціалізуються основні елементи інтерфейсу користувача для цієї активності – кнопки, текстові поля, а також контейнери, у яких вони розміщені (рисунок 2.4).

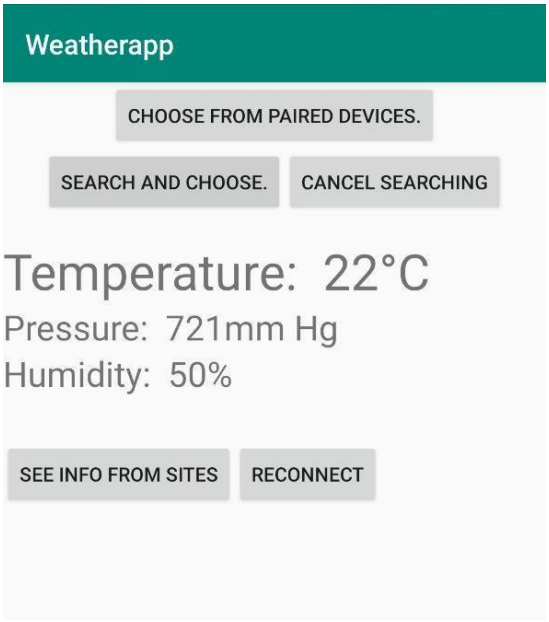


Рисунок 2.4 – Скріншот головного вікна додатку

Після усіх необхідних ініціалізацій відбувається перевірка пристрою на наявність bluetooth – модуля. Якщо він відсутній, з’явиться відповідне повідомлення (рисунок 2.5).

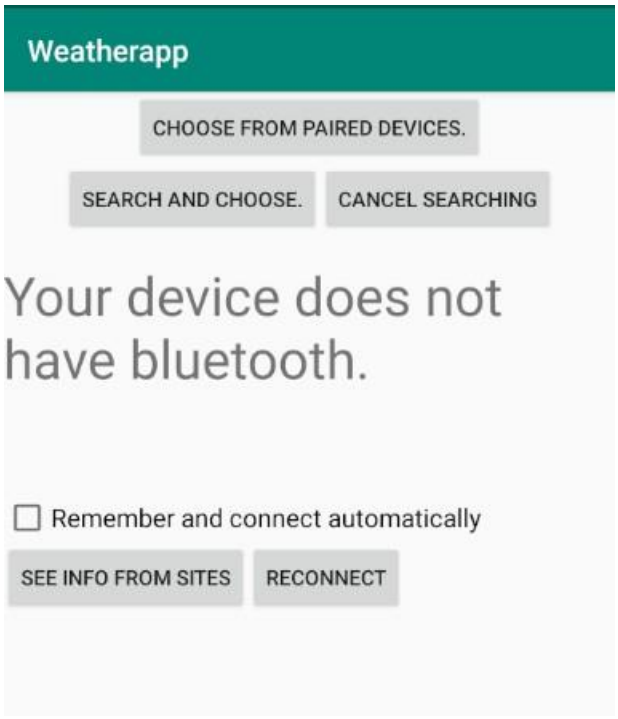


Рисунок 2.5 – Повідомлення про відсутність bluetooth – модуля

Якщо необхідний модуль є, але він вимкнений, з’явиться вікно про запит на увімкнення Bluetooth (рисунок 2.6).

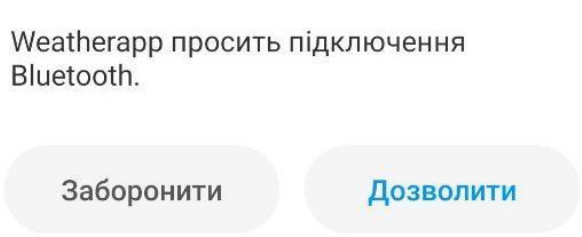


Рисунок 2.6 – Запит на увімкнення Bluetooth

Якщо користувач натисне “Заборонити”, додаток проінформує про це(рисунок 2.7).

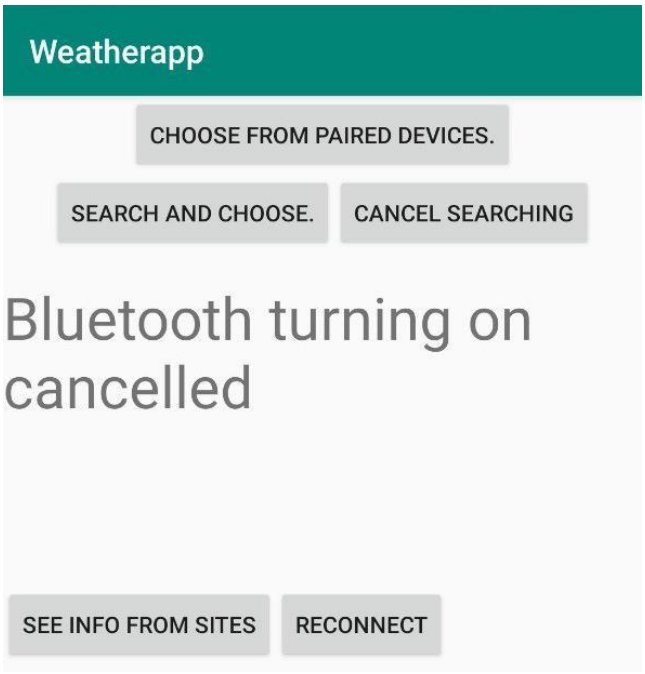


Рисунок 2.7 – Повідомлення про відхилення запиту увімкнення Bluetooth

Після цього відбувається запит дозволу на отримання геолокаціїї (якщо такий дозвіл ще не було надано). Цей дозвіл потрібен, оскільки сканування bluetooth може використовуватись для збору інформації про місцезнаходження користувача. Ця інформація може надходити як з пристроїв користувачів, так і з bluetooth - маяків. Тому для того, щоб пристрій був видимий дл сканування, або сам міг починати сканування користувачу необхідно надати дозвіл на доступ до геолокації. Сповіщення на екрані користувача будуть аналогічні тим, які з’являлись при запиті дозволу на увімкнення bluetooth.

Далі відбувається перевірка файлу Preferences. Це такий файл, у який програма може записувати певні дані формату “ключ - значення”. Він не видаляється після виклику onDestroy(). Якщо користувач заходить у програму вперше, або досі жодного разу не був відмічений чекбокс “зберегти пристрій та підключатись автоматично”, то файл порожній. Програма відображає лише стан Bluetooth – увімкнений (рисунок 2.8), чи вимкнений (рисунок 2.7). Якщо користувач раніше був підключений до пристрою, а також відмітив необхідний чекбокс, то у файлі збереглась мак-адреса пристрою. Отримавши цю адресу,

додаток ствоює новий екземпляр та запускає потік для встановлення зв’язку із пристроєм. Потоки встановлення зв’язку та обробки отриманих з пристрою даних будуть розглянуті пізніше.

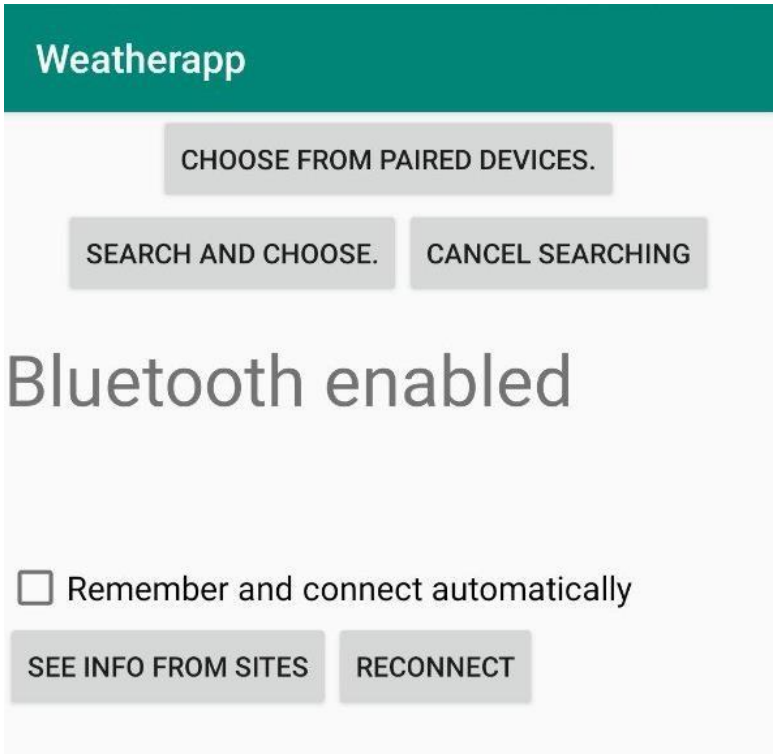


Рисунок 2.8 – Інформація про отримання дозволу на увімкнення bluetooth

Далі в коді додатку описуються обробники натискань для кнопок. Після натискання викликається відповідна функція, а також у деяких кнопках вмикається видимість елементу, в якому буде відображений список (наприклад список bluetooth-пристрїв, з якими створена пара, чи список знайдених під час сканування пристроїв). Останній важливий елемент у onCreate – хендлер, який отримує повідомлення із потоку, що отримує дані через bluetooth. Він необхідний тому, що деякі потоки, створені програмістом, не мають прямого зв’язку із інтерфейсом користувача Android, тому вони обмінюються повідомленнями із головними потоком за допомогою механізму хендлерів.

Функція BondedList() викликається при натисканні на кнопку відображення спарованих пристроїв. Ця функція відображає на екрані список

пристроїв, з якими раніше була створена пара, або виводить повідомлення, що таких пристроїв немає. Також на цей список встановлюється обробник натискань, тому при виборі одного з пристроїв ми отримаємо його мак-адресу і запуститься потік встановлення з'єднання із пристроєм.

Функція onActivityResult() відповідає за дії, що будуть зроблені після того, як система запросить дозвіл на увімкнення bluetooth і користувач надасть його. На екран буде виведено повідомлення про увімкнення, а також почеться процес встановлення з'єднання (якщо файл Preferences непорожній).

За пошук доступних bluetooth-пристроїв відповідає функція startDiscovery(). Вона викликається після натискання на кнопку пошуку нових пристроїв. Тут ми реєструємо приймач широкомовних повідомлень, який буде реєструвати зміни стану bluetooth-адаптера (пошук розпочато, пристрій знайдено, пошук закінчено), отримуємо об'єкт стандартного адаптера і за допомогою цього об'єкта починаємо пошук. Коли приймач широкомовних повідомлень отримав повідомлення про те, що знайдено новий пристрій, ім'я так мак-адреса цього пристрою записуються у масив та виводяться на екран. Як у випадку із списком спарованих пристроїв, користувач може натиснути на відповідний елемент списку і тоді почнеться процес підключення. Повідомлення про початок та закінчення сканування виводяться на екран у вигляді спливаючих сповіщень (Toasts).

Функція saveMac() зберігає мак-адресу пристрою, до якого підключений користувач у файл Preferences, якщо відповідний чекбокс був відмічений. Ця функція викликається у методі onPause(), про який згадувалось вище.

Клас AcceptThread описує потік для встановлення з'єднання по bluetooth. Тут створюється сокет для роботи із пристроєм. Варто зазначити, що у даному проєкті додаток виступає у ролі клієнта – він ініціює з'єднання, а пристрій є сервером – він прослуховує радіочастоти, призначені для bluetooth та очікує на запити про з'єднання. Щоб отримати сонет необхідно спочатку отримати об'єкт BluetoothDevice за допомогою мак-адреси необхідного віддаленого пристрою.

Потім, використовуючи цей об’єкт, а також UUID пристрою ми створюємо сокет. UUID – це стандарт ідентифікації, який використовується при створенні програмного забезпечення. Він представляє 16-байтний номер (у шістнадцятко вій системі має вигляд рядка цифр, розділених дефісами на п’ять груп). Кожен пристрій, який працює як сервер повинен мати цей код. Зазвичай він однаковий для пристроїв, що представляють однакові служби. Bluetooth-модуль розробленого у проєкті пристрою має UUID – "00001101-0000-1000-8000- 00805F9B34FB". Після отримання сокета скасовується пошук пристроїв на випадок якщо він був увімкнений (ця процедура займає дуже багато ресурсів) та викликається метод connect() для сокета. Потім викликається потік для отримання даних з сокета.

Клас AcceptedThread описує потік для отримання даних із вхідного потоку. При надходженні дані записують у буфер до тих пір, поки не зустрінеться знак переносу рядка "\n" при зчитуванні символа. Коли такий знак з’явився, дані з буфера через хендлер відправляються головному потоку, де відображаються у відповідному полі.

Розглянемо файл ChildActivity.java. У ньому міститься опис другої активності, призначеної для отримання даних про погоду з інтернету та їх відображення. У onCreate(), як і в попередній активності ініціалізуються елементи інтерфейсу – кнопки, поля для вводу, чекбокси. Тут також встановлюються обробники натискань для кнопок “пошук”, “змінити місто” та “назад”. Після натискання на “пошук” із поля вводу зчитується текст і відбувається пошук погодних даних для цього міста. Якщо назву було введено некоректно, з’явиться відповідне повідомлення (рисунок 2.9).

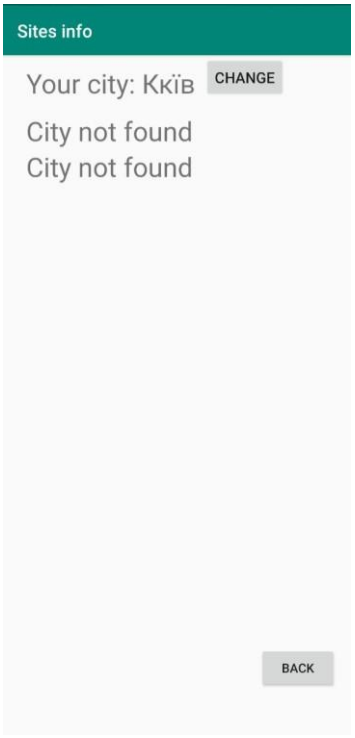


Рисунок 2.9 – Повідомлення про некоректно введене місто

Виведено два повідомлення, оскільки запит здійснювався до двох різних сервісів (детальніше це розглядатиметься у наступних розділах). Після натискання кнопки “змінити місто” у користувача з’явиться можливість заново ввести місто для перегляду погоди у ньому, а також зберегти його для автоматичного пошуку при наступному вході у активність. Кнопка “назад” повертає користувача до головної активності.

Якщо користувач заходить у активність вперше, або досі не відмічав чекбокс “зберегти місто”, додаток відкриє клавіатуру для вводу міста у полі. Якщо необхідний чекбокс був відмічений раніше, то у файлі Preferences збереглось введене місто і у методі onCreate() автоматично почнеться отримання інформації про поточні погодні умови у заданому місті. Збереження даних у файлі Preferences, як і у попередній активності відбувається у методі onPause().

Два класи: WQueryTask_SINOPTIK та WQueryTask_OWM відповідають за отримання відповіді із сервера за допомогою інструментів, описаних у файлі NetworkUtils.java (він буде розглядатись пізніше).

Розглянемо WQueryTask_OWM. Цей клас наслідує клас AsyncTask – інструмент для переміщення трудоємних операцій у фоновий потік. AsyncTask має багато методів для виконання тих чи інших завданнь, але у даному випадку використовувались лише три – onPreExecute(), doInBackground() та onPostExecute(). Метод doInBackground() повинен містити код усіх трудомістких операції, оскільки це і є основний меод класу. Він не має зв'язку із інтерфейсом користувача. Тут ми отримуємо відповідь із сервера у JSON-форматі. Метод onPreExecute() виконується до doInBackground() та має доступ до інтерфейсу користувача. У ньому вмикається видимість для анімації завантаження. Метод onPostExecute() виконується після doInBackground(), має доступ до інтерфейсу користувача. Тут ми аналізуємо відповідь, отриману із сервера. Якщо сервер надіслав "404", чи порожній рядок, встановиться відповідне повідомлення про помилку. Якщо від сервера надійшли коректні дані формату JSON, то програма отримає необхідні значення температури, вологості та тиску, здійснить необхідні перетворення (сервіс openweathermap вимірює повітряний тиск у гектопаскалях замість звичних нам міліметрів ртутного стовпчика), сформує повідомлення для виводу та надрукує його на екрані. Наостанок вимикається видимість анімації завантаження. Клас WQueryTask_SINOPTIK працює ідентичним чином, але отримує інформацію із сайту sinoptik.ua. Варто зауважити, що цей сайт не надає відкритиго API для роботи з ним, на відміну від сервісу openweathermap. Спосіб отримання даних із sinoptik.ua буде описаний далі.

Файл NetworkUtils.java містить одноіменний клас. Його призначення – з'єднання і з сервером по згенерованій URL-адресі, отримання та аналіз відповіді, наданої сервером. Спочатку у класі задаються рядкові константи, з яких потім функції generateURL_SINOPTIK() та generateURL_OWM()

ствоюють URL для HTTP-запиту. Функція на вхід отримує назву міста, введену користувачем у відповідному полі та додає цю назву до HTTP-запиту.

Функція getResponseFromUrl() спочатку аналізує код відповіді. Якщо цей код 404(Not Found) – сервер не може знайти інформацію, яку запитує клієнт, або 302(Found) – код перенаправлення, то ми поветраємо рядок “404”, який зчитується у WQueryTask. Код відповіді 302 означає, що ресурс, що запитується був тимчасово переміщений. Іноді такий код відповіді приходить при помилках якщо смартфон підключений до мережі через мобільні дані, а не через wi-fi. Якщо сервер повертає не один із вищезазначених кодів, то починається зчитування вхідного потоку. Отримані дані передаються у метод doInBackground класу WQueryTask, звідки функція getResponseFromUrl() і викликалась.

2.3 Опис використаних алгоритмів

Основні елементи програми уже були описані вище. У цьому підрозділі окремо розглянемо алгоритм передбачення погоди. У проєкті був використаний алгоритм Zambretti. Він розроблений на основі погодного калькулятора, що створила англійська фірма Negretti & Zambra. Для формування пронозу необхідні такі параметри: поточний атмосферний тиск, тенденція зміни тиску (росте, падає, не змінюється), поточний сезон – теплий (квітень-вересень) чи холодний (жовтень-березень) та напрям вітру. Для визначення тенденції зміни тиску пристрій розраховує середнє значення за останні 12 годин та передає його у додаток. Додаток порівнює поточне та середнє значення і визначає тенденцію зміни тиску. Поточний сезон береться із системного календаря, а напрям вітру отримується з мережі Інтернет.

Блок-схема алгоритму зображена у додатку 2. Розглянемо основні програмні модулі, призначені для створення прогнозу. У хендлері, що обробляє дані з потоку роботи із bluetooth перевіряються значення останніх трьох цифр повідомлення. Якщо ці цифри – “000”, то це значить що пристрій був увімкнений менше години тому і зібраних даних поки недостатньо для коректного середнього значення. У такому випадку на екрані користувача відобразиться відповідне повідомлення (рисунок 2.10). Якщо останні три цифри повідомлення не “000”, то викликається функція для підготовки даних для прогнозу а також встановлюється у стан true змінна forecast_flag. Вона необхідна для того, щоб прогноз формувався лише після першого успішного отримання хендлером середнього тиску. Прогноз складається лише один раз при запуску програми, оскільки він не змінюється різко, тому немає необхідності часто його обновляти та перевантажувати систему.

Функція prepare_forecast_data() отримує із хендлера середній та поточний тиск та записує їх у глобальні змінні для доступу інших функцій до цих даних.

Також у ній запускається потік AsyncTask для звернення до сервера та отримання даних про вітер.

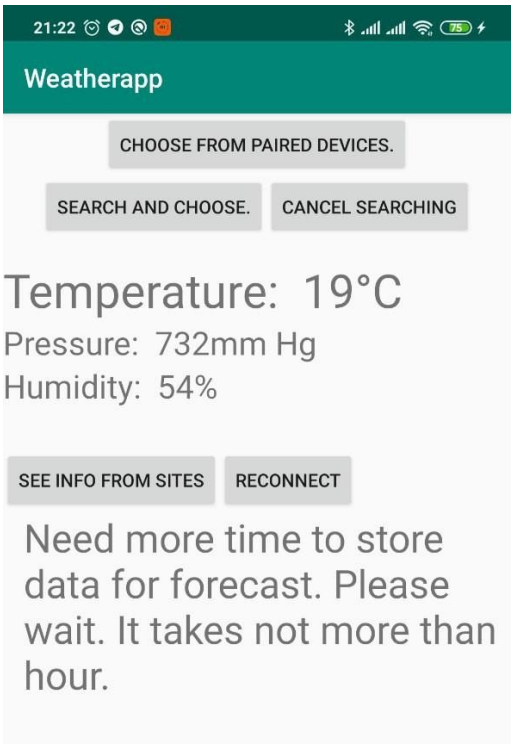


Рисунок 2.10 – Повідомлення про необхідність зібрати більше даних для формування прогнозу

У потоці AsyncTask після успішного отримання даних про поточний вітер викликається функція make_forecast(). У ній спочатку отримується поточна дата із системи Android і визначається який зараз сезон. Потім порівнюється поточний і середній тиск та визначається тенденція зміни. Після цього поточне значення тиску коригується залежно від напряму вітру. За це відповідає функція wind_dir_correction(). На вхід вона отримує поточний вітер та тиск. Залежно від напряму вітру до тиску додаються чи віднімаються певні значення, закладені у алгоритмі. Функція повертає скориговане значення тиску. Потім до цього тиску додається чи віднімається 3.2 залежно від поточного сезону. Далі залежно від тенденції зміни тиску та самого тиску обчислюється параметр z – цифрове представлення прогнозу погоди. Він отримується після обчислення по формулі,

що відрізняється для тиску що росте, падає чи залишається незмінним. Відповідно до цілої частини z та тенденції зміни тиску визначається буква, що характеризує поведінку погоди у найближчі години. За це відповідають функціїї grow_correction(), fall_correction() та stable_corecction(). Варто зазначити що для різних станів зміни (ріст, зменшення, стабільний) одній і ті ж значення z будуть відповідати різним буквам. Після цього функція forecastToString відповідно до отриманого буквеного ідентифікатора прогнозу формує прогноз словами та відображає на екрані користувача (рисунок 2.11).

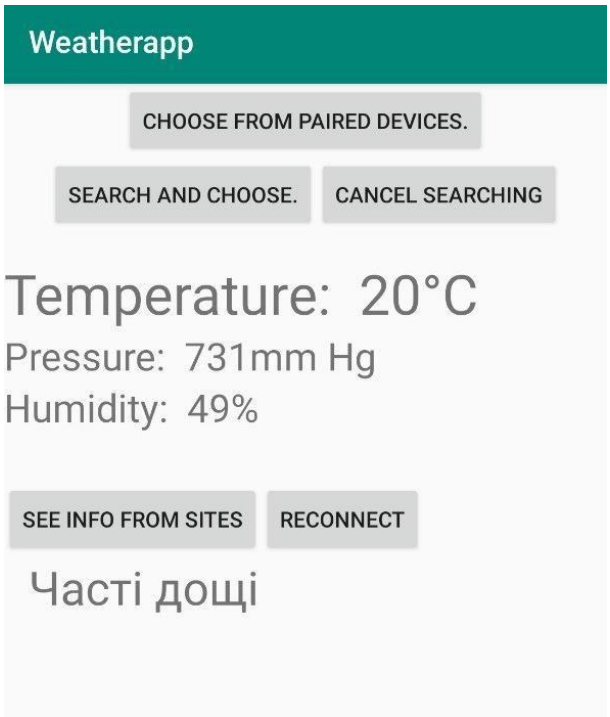


Рисунок 2.11 – Приклад прогнозу погоди

Висновки до розділу

У цьому розділі описано технології для розробки мобільних додатків та обґрунтовано вибір саме ОС Android, а також мови програмування Java та середовища розробки Android Studio. Також у розділі описується структура мобільного додатку, взаємодія його основних модулів. Окремо розглядається алгоритм пронозування погоди.

ОС Android була вибрана через те, що це найпопулярніша операційна система у сучасних смартфонах (близько 70% ринку). Мова програмування Java використовується для розробки додатків у середовищі Android Studio, яке прийшло на заміну старому та менш функціональному Eclipse. Додаток складається із двох активностей (вікон). Перша містить потоки для роботи із bluetooth, вона відображає дані про поточні погодні умови та прогноз погоди. Друга працює із HTTP-запитами для отримання та відображення даних про погоду із погодних сервісів. Алгоритм, що використаний для прогнозування погоди – Zambretti він сформований на основі багаторічних спостережень за погодою у країнах Європи.

3. ОПИС СЕРВЕРНОЇ ЧАСТИНИ

3.1 Загальна інформація

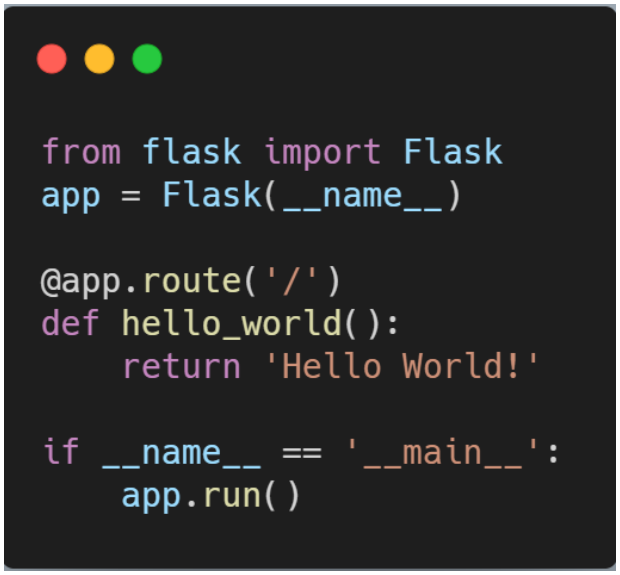
Як було зазначено вище, сайт sinoptik.ua не надає відкритого API для отримання даних про погодні умови у JSON-форматі по HTTP-запиту. Оскільки цей сайт є одним із найпопулярніших в Україні, а також він отримує дані від Українського Гідрометцентру, то виникла необхідність зібрати інформацію про погоду саме з нього. Для реалізації цього необхідно використати парсер – програму, яка отримує необхідні дані із HTML-сторінок.

Парсер можна реалізувати мовою програмування Java безпосередньо у додатку, проте існує набагато зручніший інструмент для таких завдань – бібліотека BeautifulSoup для мови Python. За її допомогою можна отримати необхідні дані із HTML-коду, написавши буквально кілька стрічок коду.

Щоб мобільний додаток міг отримувати дані, знайдені у HTML-сторінці за допомого парсера, необхідно щоб сама програма, яка містить парсер, мала доступ до мережі та могла відповідати на певні запити. Для цього потрібно написати код серверної частини. Вона була реалізована за архітектурою REST за допомогою Flask – мікрофреймворка для веб-додатків.

REST (REpresentational State Transfer) – стиль архітектури програмного забезпечення для взаємодії компонентів розподіленого додатку у мережі. Він є простим і зручним інтерфейсом керування інформацією без використання додаткових внутрішніх прошарків. Кожна одиниця інформації однозначно визначається глобальним ідентифікатором – URL. Дії з даними виконуються за допомогою методів GET (отримати), PUT (додати елементи), POST(додати, змінити, видалити), DELETE(видалити). Наприклад, частина URL “/sinoptik?q=київ” означатиме, що ми переходимо до розділу обробки даних з сервісу sinoptik.ua та формуємо GET-запит із параметром q та його значенням – назвою міста. [8]

Flask називається саме мікрофреймворком тому, що він намагається дотримуватись простого, але розширюваного ядра. Він не буде вирішувати замість програміста дуже багато речей, наприклад яку базу даних використовувати. А ті рішення, які Flask буде приймати сам дуже легко змінити, наприклад який рушій для роботи з шаблонами необхідно використовувати. На рисунку 3.1 можна побачити приклад найпростішого додатку, який виводить “Hello World!” на веб-сторінці, яка розміщена за заданим шляхом.



```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

Рисунок 3.1 – Простий додаток із використанням Flask

Спочатку ми імпортуємо клас Flask, потім створюємо екземпляр цього класу. Перший аргумент це ім’я модуля чи пакета додатку. Потім використовується декоратор route() для того щоб Flask знав який URL має запускати дану функцію. Функція повертає повідомлення, яке має бути відображене у браузері користувача. Функцію run() використовуємо для запуску локального сервера. Щоб запустити сервер лише коли програма запускається безпосередньо із інтерпретатора Python, а не коли її імпортують в якості модуля слід використати конструкцію “if_name_== '_main_'” . [9]

Щоб у мобільному додатку була можливість будь-коли і будь-де (за наявності інтернету) отримати дані, які зібрала програма, необхідно розмістити

її на сервері. Для цих цілей було використано Heroku – платформа, що дозволяє створювати, запускати та керувати застосунками, розміщеними у хмарі. Цей формат називається Platform as a Service (платформа як послуга). Програми, що працюють на Heroku використовують DNS-сервер (зазвичай додаток має адресу формату “ім’я-додатку.herokuapp.com”. Для кожного додатку виділяються кілька незалежних віртуальних процесів, які мають назву “dynos”. Heroku також має систему контролю версій Git. У базовій безкоштовній версії є обмеження на кількість процесів, а також процес може заснути, якщо не виконується ніяких дій впродовж певного часу. Наприклад, якщо надіслати відповідний запит, то сервісу необхідно приблизно 10 секунд що запустити процес, запустити додаток та виконати необхідну функцію. Якщо надсилати наступні запити одразу, сервер реагуватиме швидше. Heroku надає переваги, які зареєстровані у GitHub Students (необхідно надіслати фото студентського та коротко сказати для чого тобі потрібен цей сервіс). Таким користувачам надається Hobby Dyno, який дозволяє додатку не переходити у режим сну, здійснювати метрику додатку, призначати кілька робочих процесів, збільшити оперативну пам’ять до 512 мегабайт, дозволяє використовувати до 10 типів процесів.

Щоб вивантажити свою програму у хмару необхідно завантажити Heroku через термінал та залогінитись (у ОС Linux). Потім потрібно створити новий додаток Heroku, він зв’язується із локальним git-репозиторієм. Команда “git push heroku master” вивантажить додаток на сервер (рисунок 3.2), далі необхідно переконатись, що додаток працює на хоча б одному процесі – “ heroku ps:scale web=1”. Щоб перейти за URL-адресою додатку необхідно скопіювати її у терміналі, чи ввести команду heroku open.

У кореневій директорії додатку на сервері повинно бути ще кілька файлів, необхідних системі. Файл runtime.txt містить версію Python, якою написана програма. Файл requirements.txt містить перелік усіх необхідних бібліотек для роботи програми.

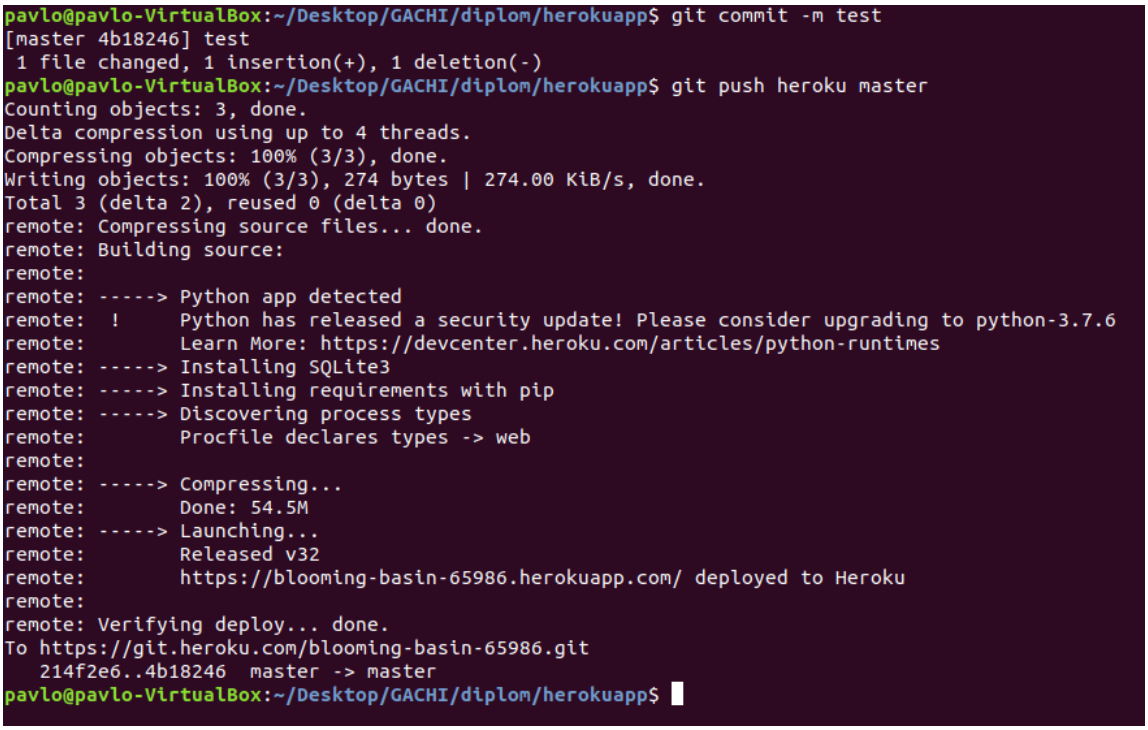


Рисунок 3.2 – Вивантаження додатку на сервер

У файлі Procfile визначається яка команда має бути використана для запуску програми. Наприклад для мого веб-додатку вміст цього файла буде таким: “web: gunicorn testflsk:app”. Це визначає один процес типу “web” та команду, необхідну для запуску. Ім’я “web” є важливим, оскільки це значить, що процес буде приєднаний до стеку маршрутизації HTTP Heroku та отримувати веб трафік після розгортання на сервері.

3.2 Програмування

Розглянемо розроблену мовою Python програму для сервера. Спочатку імпортуємо всі необхідні модулі, а також створюємо екземпляр класу Flask. Потім використовуємо “route(“/sinoptik”, methods=[‘GET’])” для того щоб вказати, що наступна функція, а саме parse() буде запускатись пісяля переходу по заданому URL, а також те, що буде використовуватись метод GET.

Для парсингу веб-сторінок використовується бібліотека BeautifulSoup. За допомогою її інструментів створюється дерево із документів типу HTML чи XML, які перед цим розпарсили. Бібліотека працює навіть для тих документів, які містять помилки (незакриті теги, тощо). У функції parse() спочатку ми отримуємо назву міста, отриману із HTTP-запиту. Потім отримуємо об’єкт BeautifulSoup, або дерево парсингу. Цей об’єкт приймає аргументом відповідь сервера на запит, у нашому випадку це “https://ua.sinoptik.ua/погода-<cityname>”, де <cityname> це назва міста, передана при запиті. Потім цей об’єкт парситься парсером lxml. Після цього можна збирати інформацію, яка нам потрібна. Для цього необхідно вручну переглянути HTML-код сторінки та знайти елементи, які нас цікавлять. Для цього натискаємо правою кнопкою миші на необхідний елемент, а потім вибираємо “переглянути код елемента”. У правій частині екрану (розташування може бути різним залежно від браузера) з’явиться HTML-код. Необхідно провести курсором по блоках коду, а редактор підсвічуватиме за який саме візуальний елемент цей код відповідає. Потім потрібно визначити class чи id необхідного елемента і за ним здійснювати пошук у програмі. Якщо на сторінці є кілька елементів із однаковим параметром class, то необхідно проводити додаткову ідентифікацію, наприклад по батьківському елементу. Розглянемо скріншот на рисунку 3.3. Нехай ми визначили, що елемент, який нас цікавить має тег <td> та клас “p8 cur”. Але є елемент із таким же тегом і класом, але вкладений у тег <tr> з класом “temperatureSens”. Хоча значення цього елемента і того що нас цікавить

однакові, але вони можуть змінитись, оскільки наш елемент – поточна реальна температура, а його “двійник” – як температура відчувається. Очевидно, ці значення можуть відрізнятись.

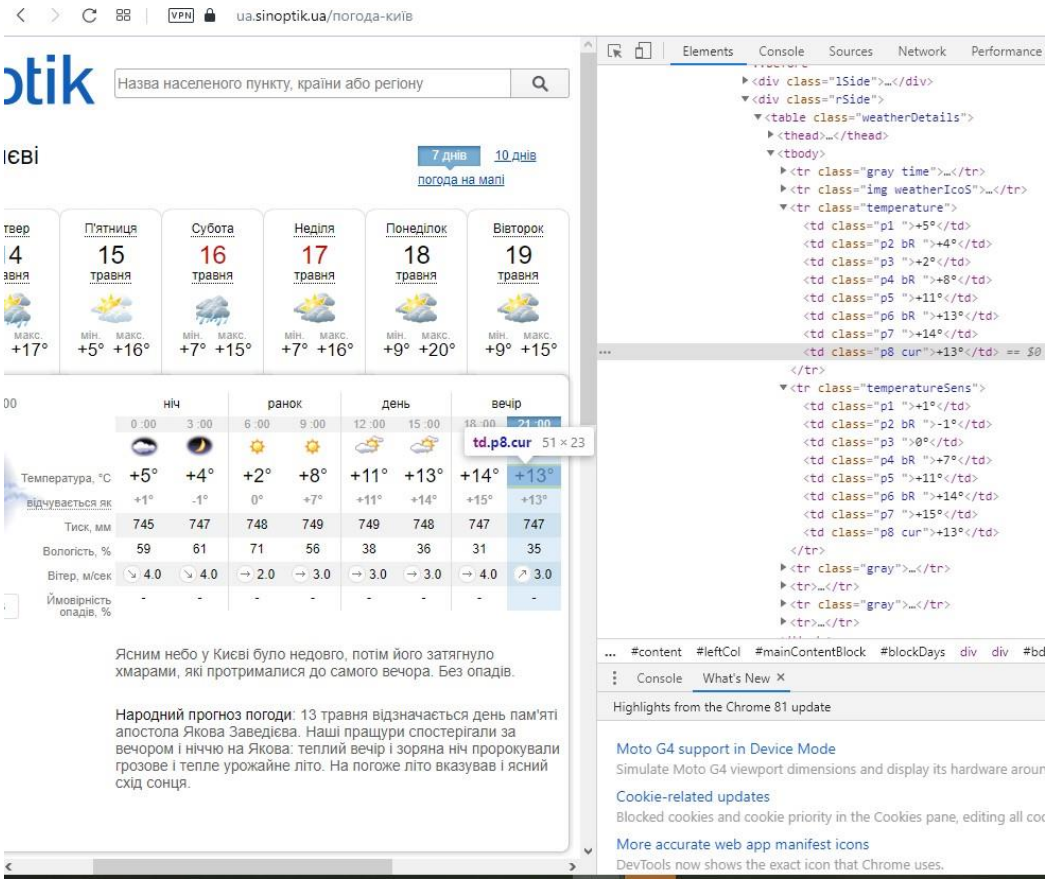


Рисунок 3.3 – Визначення необхідного елемента у HTML-коді

Тому для додаткової ідентифікації необхідно шукати тег <tr> із класом “temperature”, а вже в ньому тег <td> з класом “cur”. Проте не завжди є можливість так ідентифікувати елементи, оскільки і батьківські теги можуть мати своїх “двійників”. У такому випадку доводиться шукати теги, які ідентифікуються однозначно і знаходити бажаний елемент як той, що знаходиться на певній фіксованій позиції порівняно з ним. Саме таким способом було описано у коді як шукати дані вологості та тиску. Тиск – другий елемент серед знайдених із тегом <tr> та класом “grey” є батьком шуканого елемента – тегу <td> з класом “cur”. Вологість – другий елемент, знайдений

після тегу <tr> з класом “temperatureSens” є батьківським елементом для шуканого елемента – тегу <td> з класом “cur” (рисунок 3.4).

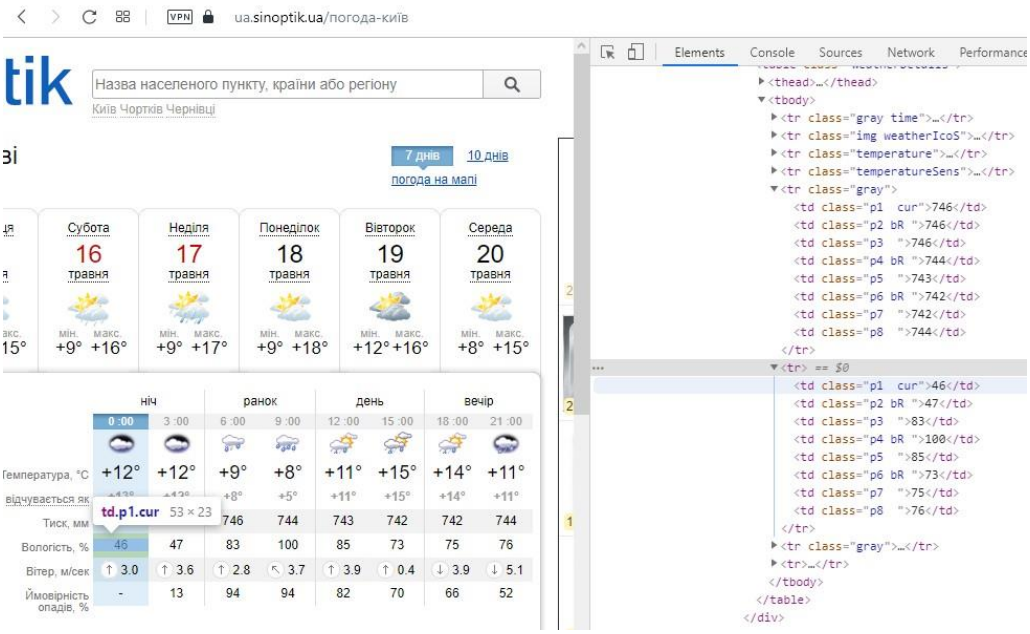


Рисунок 3.4 – Розташування елемента, шо містить значення поточної вологості

Пля того як всі потрібні значення знайдені, програма формує відповідь у JSON-форматі парами ключ-значення та відображає її на веб-сторінку. Якщо ж після запиту на sinoptik.ua прийшла порожня відповідь, то відображається код “404”, який мобільний додаток обробить та сформує необхідне повідомлення на екрані користувача. Приклад відповіді сервера зображено на рисунку 3.5.

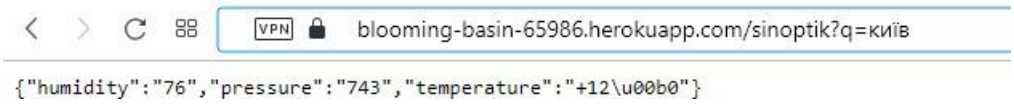


Рисунок 3.5 – Відповідь сервера на запит

Висновки до розділу

У цьому розділі описані технології для розробки серверних додатків а також створена серверна програма для парсингу сайту sinoptik.ua.

Для написання серверної частини використана мова програмування Python 3.7. У роботі використовувався мікрофреймворк Flask. Веб-додаток створений згідно із архітектурою REST. Парсинг сайту виконується за допомогою бібліотеки BeautifulSoup4. Цей процес запускається після HTTP-запиту від додатку, який містить назву міста, для якого необхідно отримати дані. Відповідь формується у форматі JSON.

4. ОПИС АПАРАТНОЇ ЧАСТИНИ

4.1 Використані апаратні засоби

За основу пристрою була взята платформа Arduino. Це проєкт з відкритим кодом, який базується на пристроях та програмному забезпеченні, які просто використовувати. Переваги цього мікроконтролера полягають у тому, що він недорогий, має сумісність з великою кількістю апаратних засобів (датчики, передавачі, контролери і т.д.), а також програмних засобів (Arduino можна програмувати не лише у IDE від розробника, є можливість розширити мову різними бібліотеками). Прошивка відбувається через USB-інтерфейс, а програмується він мовою, що базується на C/C++. Хоча ця простота використання зручна для малих та невимогливих проєктів (наприклад збір даних з датчиків та їх подальша передача), проте є можливість організувати набагато функціональніші пристрої на цій платформі (наприклад контролер польоту для квадрокоптера).

Для дипломного проєкту була вибрана платформа Arduino nano (рисунок 4.1) через її невеликі розміри – 43,18мм на 18,54мм.

Рисунок 4.1 – Arduino nano

Вона створена на базі восьмибітного мікроконтролера ATmega328. Він обладнаний 32 робочими регістрами, всі вони напряму під’єднані до арифметико-логічного пристрою, що дозволяє двом незалежним регістрам бути доступними для одної інструкції, що виконується за один такт.

Живлення Arduino може отримувати від Mini-B USB, від нерегульованого чи регульованого зовнішнього джерела живлення. Робоча напруга – 5В, рекомендована вхідна напруга від зовнішнього джерела – 7-12 В. Мікроконтролер має 32 Кб флеш-пам'яті для зберігання коду програми, 2 Кб використовується для зберігання завантажувача. Оперативний запам'ятовуючий пристрій має розмір 2 Кб, а EEPROM (Electrically Erasable Programmable Read-Only Memory) – енергонезалежна пам'ять для збереження даних має розмір 1 Кб. Тактова частота мікроконтролера – 16 МГц. Всього портів вводу-виводу є 20, при тому 8 з них підтримують аналогово-цифрове перетворення, а 6 з них – широтно-імпульсну модуляцію. Розрядність аналогово-цифрового перетворення – 10 біт, широтно-імпульсної модуляції – 8 біт. Максимальний вихідний струм піна 5V(5 вольт) – 800 мА, піна 3V3 (3,3 вольти) – 50 мА. Максимальний струм, який можна подавати на звичайний пін, чи отримати з нього – 40 мА.

На платформі присутні кілька інтерфейсів зв'язку із комп'ютером, іншими пристроями Arduino чи мікроконтролерами. Вона підтримує послідовний інтерфейс UART, який здійснюється через піни 0 (RX) та 1(TX). Установлена на платі мікросхема FTDI FT232RL направляє цей інтерфейс через USB, а драйвери FTDI надають віртуальний COM порт середовищу на комп'ютері. Моніторинг послідовної шини дозволяє надсилати і отримувати текстові повідомлення при підключенні до платформи. Світлодіоди RX та TX на пристрої будуть мигати при передачі даних через мікросхему FTDI та USB. Варто зауважити, що мікросхема FTDI FT232RL отримує живлення тільки якщо сама платформа живиться від USB. Таким чином при живленні від зовнішнього джерела інтерфейс UART через піни RXD та TXD може обмінюватись даними із зовнішнім пристроєм.[10] Уданому дипломному проєкті цей пристрій – bluetooth-модуль. Ще два інтерфейси зв'язку, які підтримуються платформою – I2C (Iter-Integrated Ciruit) та SPI (Serial Peripherial Interface). I2C – послідовна асиметрична шина для обміну даними між інтегральними схемами.

Використовує дві двонапрямлені лінії зв’язку, використовується для з’єдання низькошвидкісних периферійних пристроїв з процесорами та мікроконтролерами.[11] SPI – послідовний синхронний повнодуплексний стандарт передачі даних для забезпечення простого сполучення мікроконтролерів та периферії. На відміну від стандартного послідовного порту, SPI є синхронним інтерфейсом, в якому кожна передача синхронізована з тактовим сигналом, що генерується ведучим пристроєм (мікроконтролером).[12]

Розглянемо розпіновку Arduino nano (рисунок 4.2).

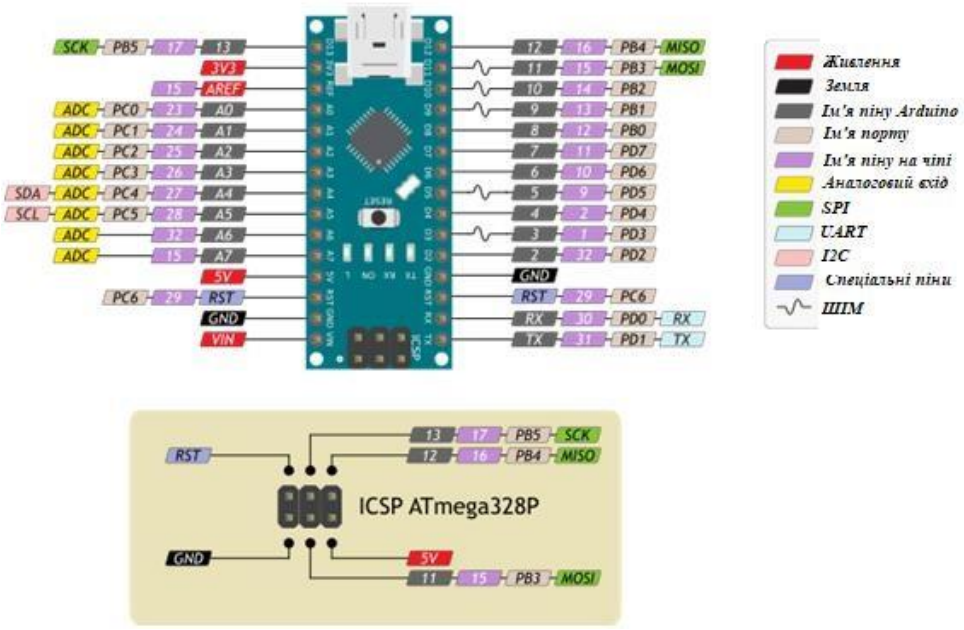


Рисунок 4.2 – Розпіновка Arduino nano

Піни живлення:

- VIN – вхідний пін для підключення зовнішнього джерела живлення х напругою 7-12 В;
- 5V – вихідний пін від регулятора напруги на павті із напругою 5 В і максимальним струмом 800 мА;

- 3V3 – вихідний пін від стабілізатора мікросхеми FT232R з напругою 3,3 В і максимальним струмом 50 мА;
- GND – земля;
- AREF – пін для підключення зовнішньої опорної напруги аналогово-цифрового перетворювача відносно якого виконуються аналогові виміри;

Порти вводу/виводу:

- цифрові входи/виходи – піни 0-13, рівень одиниці – 5 В, нуля – 0 В, до контактів піключені підтягуючі резистори, які виключені за замовчуванням, але можуть бути включені програмно;
- піни, що підтримують широтно-імпульсну модуляцію – 3, 5, 6, 9-11, дозволяють виводити аналогові значення у вигляді ШІМ-сигналу, розрядність ШІМ – 8 біт;
- піни, що підтримують аналогово-цифрове перетворення – A0-A7, дозволяють представляти аналогову напругу у цифровому вигляді, розрядність АЦП – 10 біт, вхідна напруга – 0-5 В;
- піни, що підтримують SPI – 11(MOSI), 12(MISO), 13(SCK), 10(SS);
- піни, що підтримують I2C – A4(SDA) і A5(SCL);
- піни, що підтримують UART – 0(RX) та 1(TX);
- пін RST перезавантажує контролер;

У платформі є можливість прошивки не через USB а через програматор через виходи блоку ICSP (In-System Programmer) – програматор всередині схеми. Arduino nano розроблена таким чином, що перед записом нового коду перезавантаження платформи здійснюється програмно. Одна із ліній FT232RL, які керують потоковими даними підключена до виводу перезавантаження мікроконтролера ATmega328 через конденсатор 100 нФ. Активація даної лінії, тобто подача сигналу низького рівня, перезавантажує мікроконтролер.

Ця функція має ще одне застосування. Перезавантаження платформи відбувається кожного разу при підключенні програми Arduino на комп'ютері. Наступні півсекунди після перезавантаження працює завантажувач. Під час програмування відбувається затримка декількох перших байтів коду щоб уникнути отримання платформою некоректних даних (усіх, крім коду нової програми). Якщо відбувається разове відлагодження скетчу, записаного в платформу, або ввід будь-яких інших даних при першому запуску необхідно впевнитись, що програма на комп’ютері очікує одну секунду перед передачею даних.

Розглянемо периферійні пристрої, використані у розробці. Для вимірювання температури, відносної вологості та тиску був використаний датчик Bosch BME280 (рисунок 4.3).

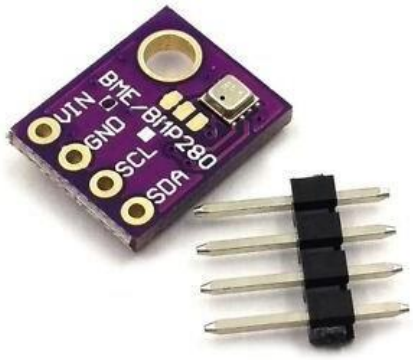


Рисунок 4.3 – Датчик BME280

Він має високу точність вимірювання, компактний розмір, високу швидкість передачі даних та низьке енергоспоживання. Розміри модуля – 15 х 12 х 3 мм. Він підключається до Arduino по інтерфейсу I2C. Максимальна швидкодія передачі даних – 3,4 МГц. Діапазон вимірювання температури – -40-85 °С, тиску – 300-1100 гПа (225-825 міліметрів ртутного стовпчика), відносної вологості – 0-100%. Можливі похибки вимірювання – до 1°С температури, до 3% вологості та до 1 гПа (0,75 міліметра ртутного стовпчика) тиску. Напруга

живлення – 1,8-5 В. Струм, що споживається в режимі виміру тиску: 714 мкА, в режимі виміру вологості: 340 мкА, в режимі виміру температури: 350 мкА, в режимі простою: 0.1-0.5 мкА.[13]

Для передачі даних через bluetooth був використаний модуль HC-05 (рисунок 4.4).

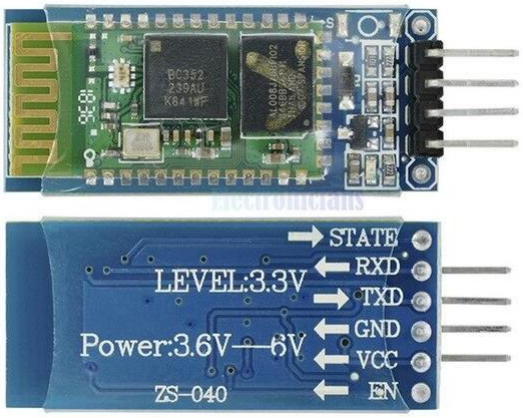


Рисунок 4.4 – Bluetooth-модуль HC-05

Він може працювати у режимах master(здійснює пошук пристроїв та ініціює підключення) та slave (прослуховує та знаходить запити на встановлення з'єднання). Уданому проєкті bluetooth-модуль працює у режимі slave(сервер). Він очікує на з'єднання та передає інформацію пристрою, який до нього підключився. Модуль має такі технічні характеристики: діапазон частот радіозв'язку – 2,4-2,8 ГГц, потужність передавання – 0,25-2,5 мВт, напруга живлення – 3,3-5 В, струм споживання – 50 мА, радіус дії – до 20 метрів, інтерфейс – послідовний порт, робочий діапазон температур – -25-75°С, розміри – 27 х 13 х 2,2 мм. Версія bluetooth, яка підтримується модулем – 2.0. Модуль має 6 виводів:

- VCC – живлення (3,6-6 В), підключається до виводу 5V Arduino;
- GND – земля, підключається до GND Arduino;
- TXD, RXD –UART-інтерфейс, підключається до RX та TX Arduino відповідно:

- STATE – індикатор стану (на ньому низький рівень якщо модуль не підключений по bluetooth, коли модуль підключається до якогось пристрою, на цьому виводі стає високий рівень), у проєкті цей вивід не використовується;
- EN – ввімкнення/вимкнення модуля, у проєкті не використовується.

Також цей модуль можна налаштовувати за допомогою АТ-команд. Щоб увійти у режим налаштування необхідно підключити модуль до програматора, або за допомогою USB-UART моста Arduino. На модулі є кнопка, після натискання якої та при правильному підключенні він переходить у режим налаштування.[14]

Для живлення пристрою використовується літій-іонний акумулятор ємністю 2500 мікро Ампер-годин. Для того щоб акумулятор можна було заряджати не виймаючи із пристрою, а також для контролю розряду (необхідно вимкнути живлення, якщо акумулятор видає менше 2,5 В, інакше він вийде з ладу) був використаний контролер заряду TP4056 (рисунок 4.5). Він має розміри 17 х 26 мм. Для заряджання використовує роз'єм micro USB, або входи IN+ та IN-. Модуль має захист від перевантаження. Він заряджає акумулятор до напруги 4,2 В, струм заряду – 1 А. До OUT+ та OUT- підключається пристрій, який необхідно живити, а до B+ та B- – акумулятор .

Оскільки робоча напруга літій-іонного акумулятора 3,6 В, а розроблений пристрій рекомендовано живити напругою 4,5 – 5 В, то у проєкті було використано підвищуючий конвертер напруги MT3608 (рисунок 4.6). На вхід він може приймати від 2 до 24 В, а на виході видавати від 5 до 28 В (вихідна напруга регулюється змінним резистором).

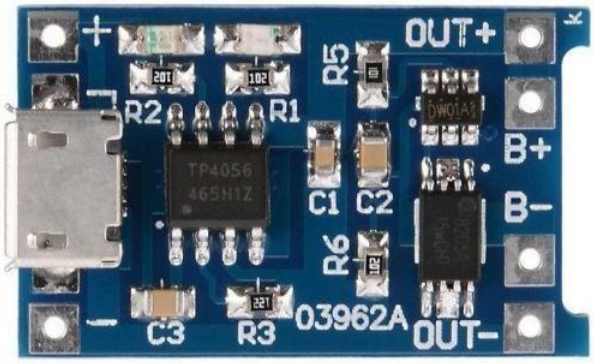


Рисунок 4.5 – Контролер заряду TP4056

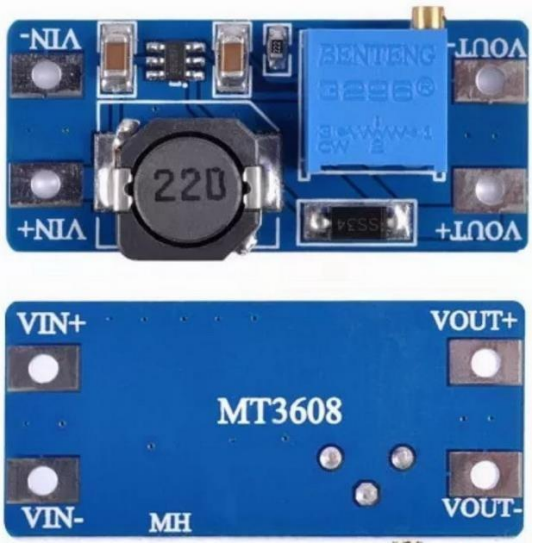


Рисунок 4.6 – Підвищуючий конвертер напргуи MT3608

Коефіцієнт корисної дії перетворювача близько 93%. Розміри модуля – 36 х 17 х 14мм. До VIN+ і VIN- підключається джерело напруги, яку слід підвищити, до VOUT+ та VOUT- – пристрій, який потребує підвищену напругу.[15]

4.2 Програмування

Як було сказано вище, мова програмування платформи Arduino базується на C/C++, тому візуально дуже її нагадує. Скріншот середовища Arduino та вигляд мови можна побачити на рисунку 4.7.

Рисунок 4.7 – Вигляд середовища та мови програмування Arduino

Середовище не надто функціональне, не має функції автодоповнення коду як більшість сучасних IDE, але для невеликих проєктів його цілком вистачає. Після натиснення кнопки вивантаження коду відбувається його компіляція та, якщо помилок не знайдено, вивантаження у контролер. Також можна перевірити код на помилки до вивантаження. У середовищі є емулятор послідовного порту та монітор для перегляду даних. У програмі вказується швидкість обміну даними через порт, у моніторі, який відкривається

відповідною кнопкою у середовищі, необхідно вибрати таку саму швидкість. Скріншот монітору послідовного порту зображено на рисунку 4.8.

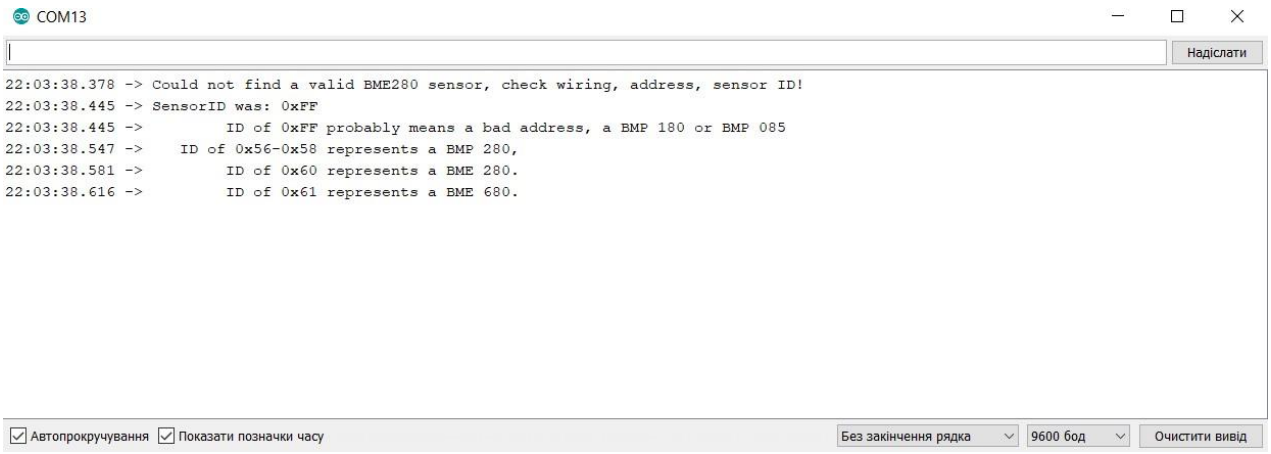


Рисунок 4.8 – Монітор послідовного порту Arduino

Через монітор можна надсилати та отримувати дані з контролера, а також задавати різні параметри відображення (позначки часу, автопрокручування, швидкість передачі даних, тощо).

Розглянемо розроблену програму. Стандартна структура програми на arduino передбачає наявність у програмі двох функцій – setup() та loop(). Код першої виконується лише один раз при запуску контролера, у ньому зазвичай ініціалізуються константи, а також відкривається монітор послідовного порту із певною швидкістю (якщо це потрібно). Код всередині функції loop() буде виконуватись безперервно – у вічному циклі. Тут описується логіка роботи контролера із урахуванням особливості цієї функції. Спочатку у програмі описується екземпляр bme класу Adafruit_BME280 із бібліотеки Adafruit_BME280.h для роботи із метеодатчиком. Потім із класу bme викликаємо метод begin() із параметром 0x76 – I2C адресою датчика. Якщо метод повернув код помилки – виводимо на екран відповідне повідомлення (його модна побачити на рисунку 4.8). Після цього задаємо константу затримки – 3 секунди. На цьому setup() завершується. У функції loop() викликається функція

printValues() та виконується засинання на заданий час – 3 секунди. Таким чином функція printValues(), яка збирає інформацію з датчика, робить перетворення гПа на міліметри ртутного стовпчика та виводить інформацію у UART-інтерфейс, до якого підключений bluetooth-модуль, виконуватиметься один раз у 3 секунди. Цієї частоти достатньо щоб отримувати актуальну інформацію та не перевантажувати систему (через надто часті опитування датчика він може нагріватись, що зменшить точність виміряних даних, а також це зменшує ресурс акумулятора).

Також пристрій обчислює середнє значення повітряного тиску за останні 12 годин. Кожні п’ять хвилин дані про тиск записуються у масив та відбувається обчислення середнього значення за алгоритмом рухомого середнього (moving average). Суть цього алгоритму полягає у тому, що середнє значення обчислюється заново при кожному отриманому новому значенні. Таким чином середнє значення рухається по часовому ряду (звідси і назва алгоритму). Це потрібно для того щоб відкинути старі неактуальні значення та бачити лише найсвіжішу тенденцію зміни тиску.

У послідовний порт відправляється така інформація: одна чи дві цифри температури, три цифри тиску, дві цифри вологості та три цифри середнього тиску. Після цього надсилається знак переносу рядка – “\n” (за нього відповідає функція println()). Саме по цьому знаку потік обробки даних з bluetooth у мобільному додатку ідентифікує завершення повідомлення із пристрою і надсилає рядок, отриманий до “\n” у головний потік. Для того щоб можна було зробити порівняння поточного та середнього тиску і побачити тенденцію зміни, необхідний певний час для накопичення даних. Тому першу годину після увімкнення пристрою він замість середнього тиску надсилатиме “000”. Це сигналізуватиме додаток про те, що дані ще збираються і їх кількість замала для аналізу.

Висновки до розділу

У цьому розділі описані апаратні засоби, використані для створення пристрою зчитування, обробки та передачі даних про погодні умови, а також програмне забезпечення для цього пристрою.

Для пристрою були використані такі апаратні засоби: мікроконтролер Arduino nano, датчик температури, вологості та тиску Bosch BME 280, bluetooth-модуль HC-05, модуль контролю живлення TP4056 та підвищуючий конвертер напруги MT3608. Програмне для пристрою написане мовою, що базується на мові С. Призначення апаратної частини – збір, обробка та надсилання даних про погодні умови на смартфон через бездротовий канал зв’язку.

5. РЕЗУЛЬТАТИ РОБОТИ ТА ТЕСТУВАННЯ

5.1 Результати роботи

Результатом роботи є пристрій для зчитування та передачі по bluetooth даних про поточні погодні умови, а також програмне забезпечення – парсер погодних сайтів на хмарному сервері та мобільний додаток, який отримує дані про поточну погоду через bluetooth, а також з хмари. Пристрій запускається автоматично при підключенні акумулятора. Хмарний сервер доступний завжди, але через обмеження безкоштовної версії він може засинати після певного часу бездіяльності, що впливає на швидкість реагування при першому підключенні після сну.

Розглянемо функціонал розробленого мобільного додатку. При першому вході у додаток запитуються дозволи на доступ до геолокації та увімкнення bluetooth, якщо він вимкнений (це описано у розділі 2.2). Якщо користувач надав необхідні дозволи, на екрані відображається головна активність із повідомленням, що bluetooth увімкнено (рисунок 5.1).

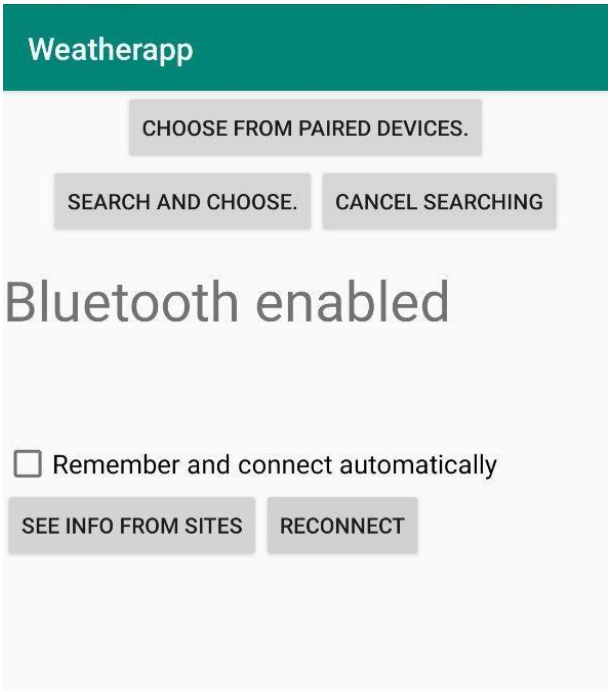


Рисунок 5.1 – Повідомлення про увімкнений bluetooth

Після цього можна почати пошук bluetooth-пристроїв навколо (кнопка “search and choose”), або підключитися до одного з пристроїв, пара з яким була створена раніше (кнопка “choose from paired devices”), або ж подивитись інформацію із погодних сайтів для свого міста (кнопка “see info from sites”). Після натискання на кнопку “search and choose” почнеться пошук пристроїв, про що користувача сповістить спливаюче сповіщення (рисунок 5.2).

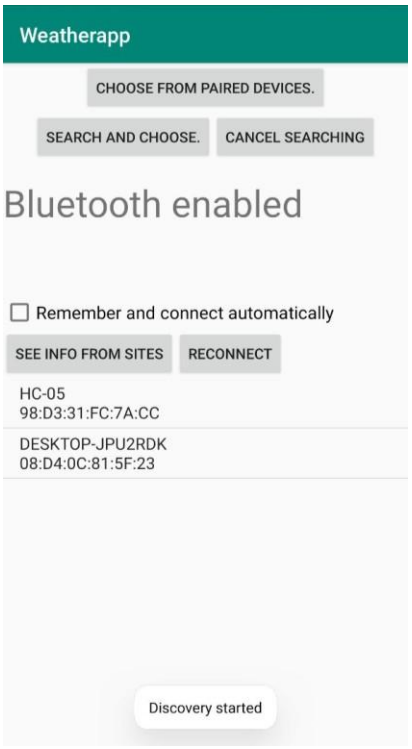


Рисунок 5.2 – Сповіщення про початок сканування

Після початку сканування пристрої по мірі їх виявлення будуть з’являтись на екрані (відображається назва так мак-адреса). Якщо користувач знайшов необхідний йому пристрій він може натиснути на нього – тоді сканування закінчиться і почнеться підключення, або якщо користувач передумав шукати пристрої, він може натиснути кнопку “cancel searching”, хоча пошук може завершитись і автоматично через певний час. Про закінчення сканування, а також якщо сканування закінчилось, але пристрої не знайдені сповіщає спливаюче повідомлення.

Якщо натиснути кнопку “choose from paired devices”, на екрані одразу відобразиться список усіх пристроїв, з якими створена пара. Спробувати підключитись до будь-якого з них можна через натискання на нього. Після натискання (як на знайдений пристрій під час сканування, так і на той, з яким створена пара) починається спроба підключення, про що відображається відповідне повідомлення на екрані. Якщо досі з пристроєм не була створена пара, система Android запропонує її створити (рисунок 5.3), інакше підключитись не вдасться.

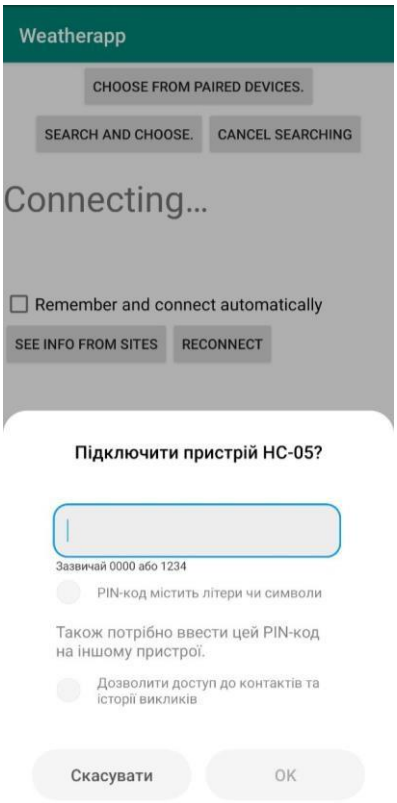


Рисунок 5.3 – Створення пари із bluetooth-модулем

Потрібно ввести стандартний PIN-код – “1234”, після цього почнеться підключення. Якщо було вибрано правильний пристрій, а також сам пристрій увімкнений і знаходиться у радіусі доступності (до 15 метрів), то підключення встановиться і на екрані відобразиться інформація про погодні умови (рисунок 5.4).

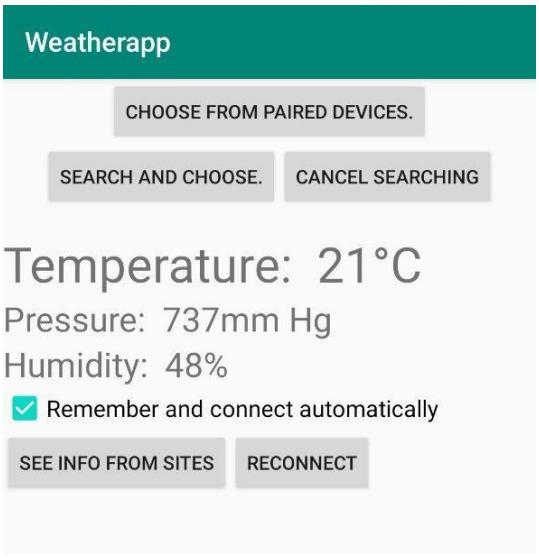


Рисунок 5.4 – Інформація про погодні умови

Якщо користувач відмітить чекбокс “Remember and connect automatically”, то мак-адреса запишеться у пам’ять пристрою і при наступному вході у програму підключення відбудеться автоматично. Кнопка “Reconnect” потрібна щоб перепідключитись до пристрою якщо зв’язок було втрачено, або система Android очистила пам’ять, у якій працював потік роботи із bluetooth (це може статись через політику енергозбереження пристрою чи якщо програма довго працювала у фоні і користувач не заходив у неї). У випадку втрати зв’язку на екрані з’явиться відповідне повідомлення (рисунок 5.5).

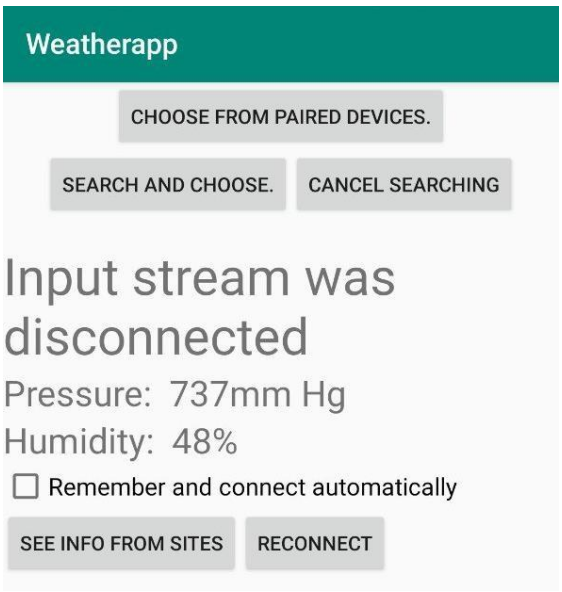


Рисунок 5.5 – Повідомлення про втрату зв’язку

Також повідомлення про помилку з’явиться якщо користувач намагається підключитись не до пристрою моніторингу погоди, пристрій вимкнено, чи він надто далеко (рисунок 5.6).

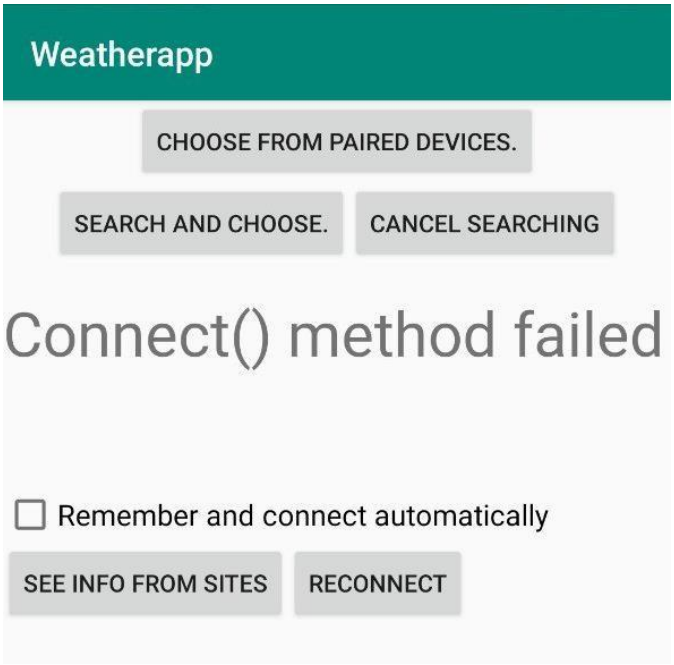


Рисунок 5.6 – Сповіщення про невдалу спробу підключитись до пристрою

Якщо натиснути кнопку “see info from sites”, відкриється друга активність, яка призначена для перегляду інформації із погодних сайтів – sinoptik.ua та openweathermap.com. При першому вході у програму активність виглядає як на рисунку 5.7. У текстове поле вгорі потрібно ввести назву міста, якщо погоду для цього міста надалі треба шукати автоматично, необхідно відмітити чекбокс “remember city” та натиснути кнопку “search”. Після цього екранна клавіатура зникне, а кнопка “search” зміниться на “change”. Якщо її натиснути, з’явиться можливість ввести інше місто і запам’ятати його замість попереднього. Після вводу треба натиснути кнопку “search”. Якщо місто було збережене, його назва буде відображатись на місці поля вводу (рисунок 5.8) поки користувач не натисне кнопку “change”. Кнопка back повертає користувача назад до головної активності – інформації з пристрою.

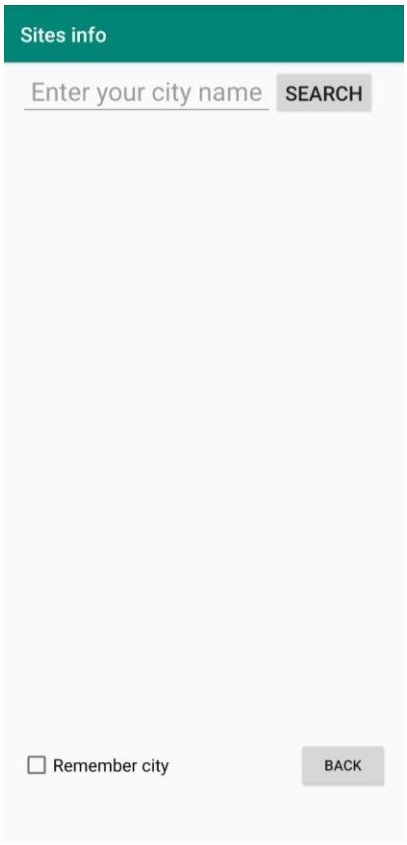


Рисунок 5.7 – Активність інформації із погодних сайтів

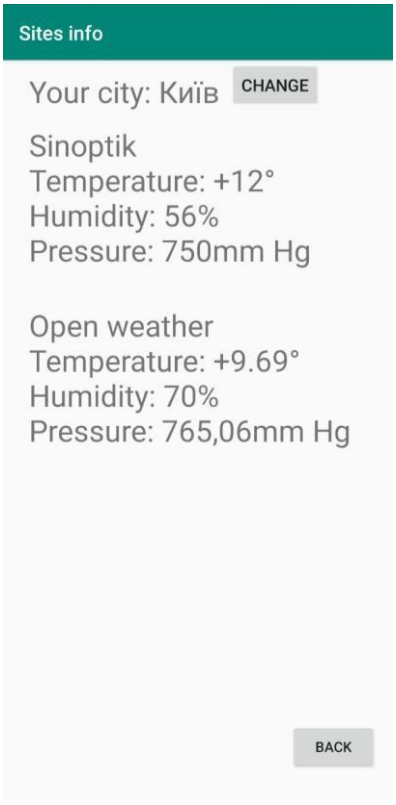


Рисунок 5.8 – Інформація із сайтів для міста Київ

Варто зауважити, що, хоча першопочатково додаток прєктувався для ОС Android 9 і нижче, проте із офіційним виходом ОС Android 10 його було модифіковано на сумісність і з цією версією. У новому Android розробники змінили політику щодо геолокації пристрою. Як було сказано вище, дозвіл до отримання геолокаціїї необхідний додатку, оскільки при пошуку нових пристроїв bluetooth місцезнаходження смарфону може бути визначено іншими пристроями, чи bluetooth-хабами. Проте у Android 10 для пошуку нових пристроїв користувач повинен не лише надати необхідний дозвіл, а і увімкнути геолокацію. Для цього у додаток було додано необхідний запит у інтерфесі користувача, який з'являється якщо версія Android більша чи рівна 10, а також геолокація вимкнена. Користувач може або відхилити запит, або прийняти його. У другому випадку він буде перенаправлений до налаштувань геолокації, де зможе її увімкнути (рисунок 5.9).

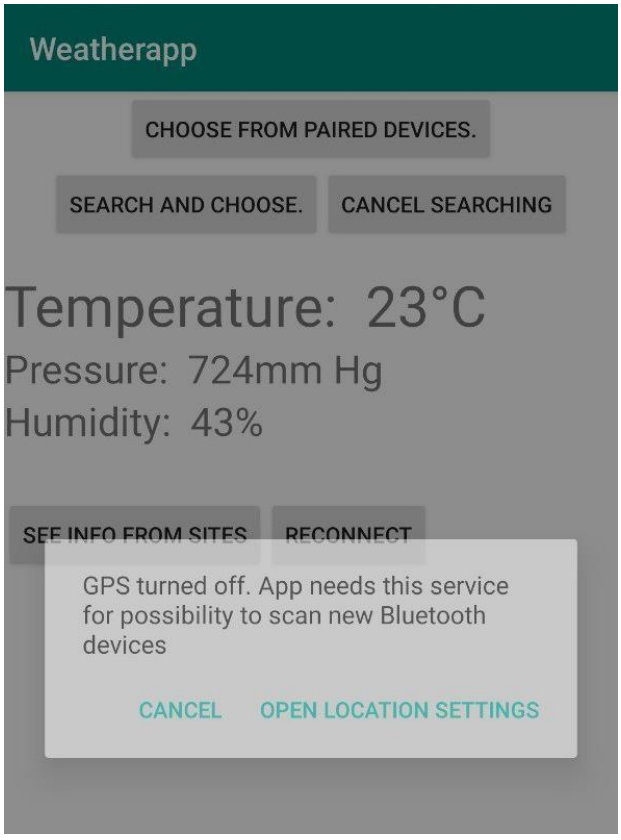


Рисунок 5.9 – Запит на увімкнення геолокації

Для формування прогнозу погоди необхідно отримати напрям вітру з мережі Інтернет. Для цього потрібно щоб користувач ввів та зберіг місто, у якому він зараз знаходиться. Також потрібне середнє значення атмосферного тиску мінімум за останню годину. Якщо користувач не ввів чи не зберіг місто, або пристрій був увімкнений менше години тому, то на екрані в полі прогнозу відобразяться відповідні повідомлення (рисунок 5.10 та рисунок 5.11).

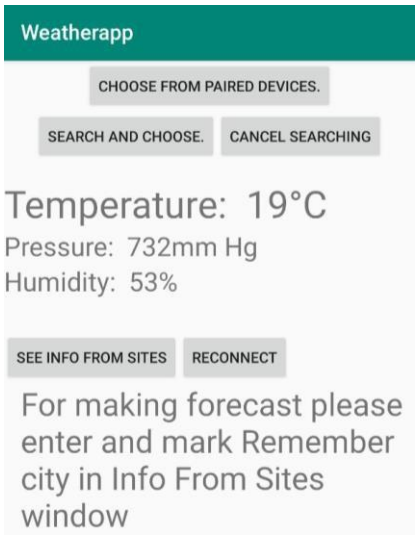


Рисунок 5.10 – Повідомлення про відсутність збереженого міста

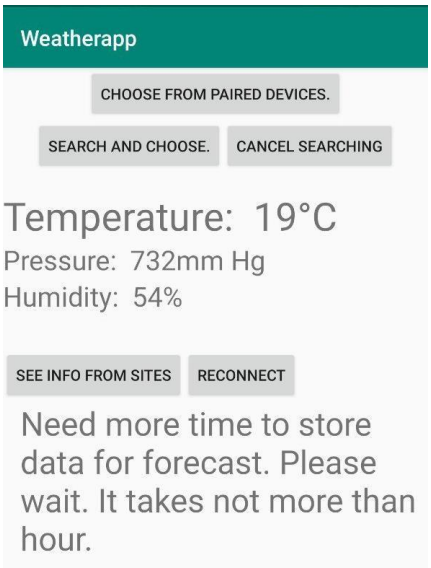


Рисунок 5.11 – Повідомлення про необхідність зібрати більше даних для формування прогнозу

5.2 Тестування програмного та апаратного забезпечення

Проведемо тести мобільного додатку щоб перевірити його поведінку у випадках некоректних вхідних даних та зовнішніх факторів. Якщо користувач відключив bluetooth або вимкнув живлення пристрою під час роботи додатку, на екрані з’явиться відповідне повідомлення (рисунок 5.12).

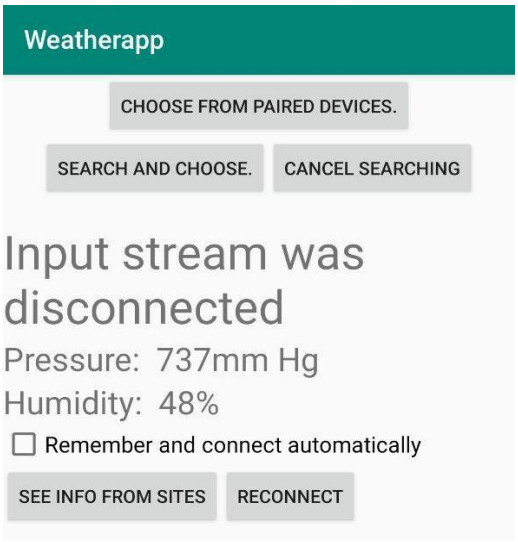


Рисунок 5.12 – Повідомлення про відключення пристрою

Також було протестовано дальність роботи пристрою. У межах приміщення зв’язок є у радіусі 12 метрів, навіть якщо між пристроєм і смартфоном є стіни. У мажах прямої видимості зв’язок із пристроєм є у радіусі до 50 метрів. Цих характеристик цілком достатньо для коректної роботи у більшості квартир чи навіть приватних будинків. Варто зауважити що якщо навколо є велика кількість bluetooth чи wi-fi передавачів, то можливе зменшення дальності роботи через перешкоди від інших пристроїв.

У випадку некоректно введеної назви міста чи взагалі спроби залишити поле вводу порожнім і натиснути кнопку “search” додаток відреагує як показано на рисунку 5.13 (у випадку некоректного міста будуть такі самі повідомлення, але без спливаючого повідомлення “Input field is empty”). У полі відображення

поточного міста виведене попереднє збережене місто, оскільки додаток не дозволить зберегти порожній рядок.

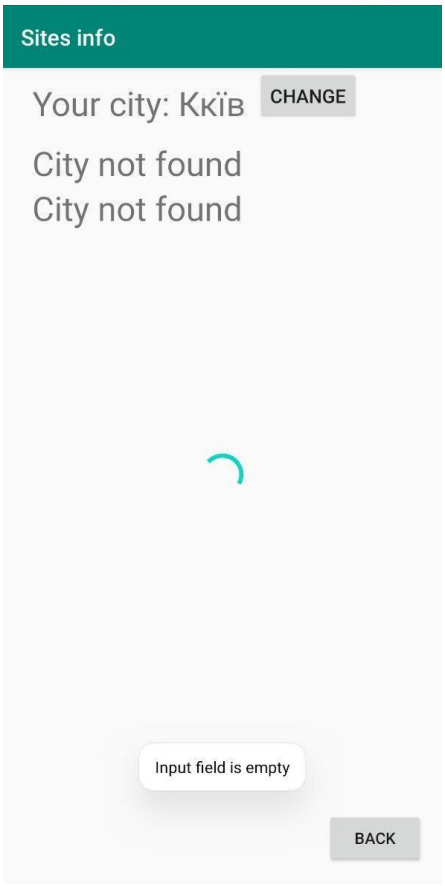


Рисунок 5.13 – Повідомлення про порожнє поле для вводу

5.3 Порівняння даних у реальному часі та прогнозів із даними погодних сервісів

Розглянемо порівняння прогнозу погоди у додатку та короткострокових прогнозів на одних із найпопулярніших в Україні погодних сайтах – sinoptik.ua та gismeteo.ua. Як видно із скріншотів на рисунку 5.14 о 21 годині 12 хвилин прогноз був “Похмуро, скоро проясниться”, але вже о 22.47 він змінився на “Дощі, дуже погана, похмуро”.

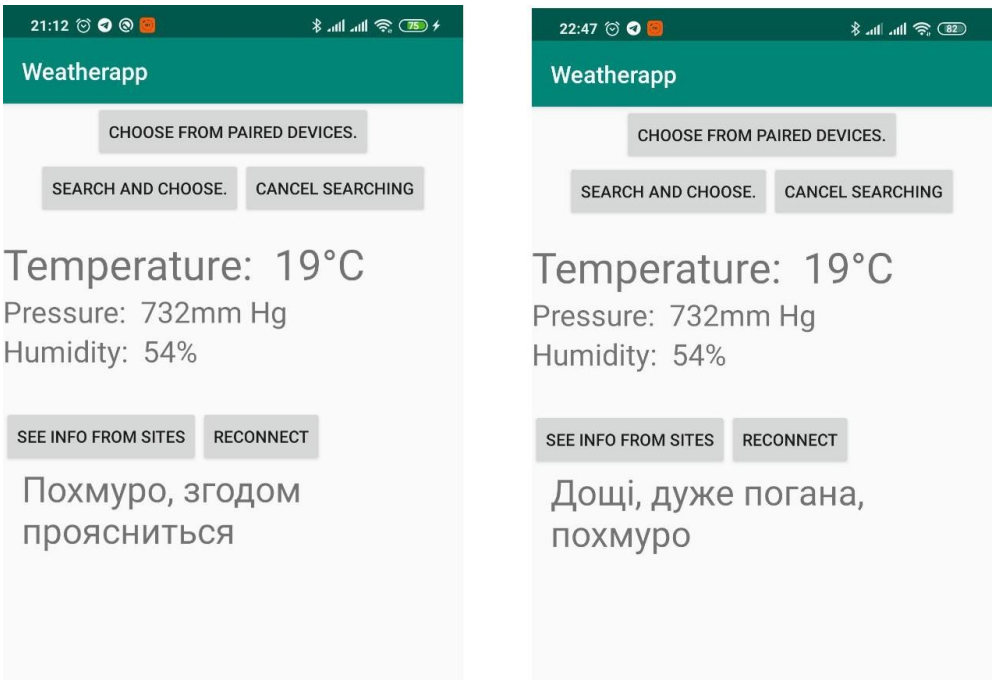


Рисунок 5.14 – Прогноз погоди у додатку

На рисунку 5.15 можна побачити що на сайтах теж очікується спочатку прояснення вночі, а потім погіршення погоди та дощі у другій половині наступного дня. Наступного дня додаток сформував прогноз “Штормова погода, дощі”. На погодних сайтах теж передбачається дощ та посилення вітру ввечері та дощ у першій половині наступного дня. Відповідні скріншоти показані на рисунках 5.16 та 5.17.

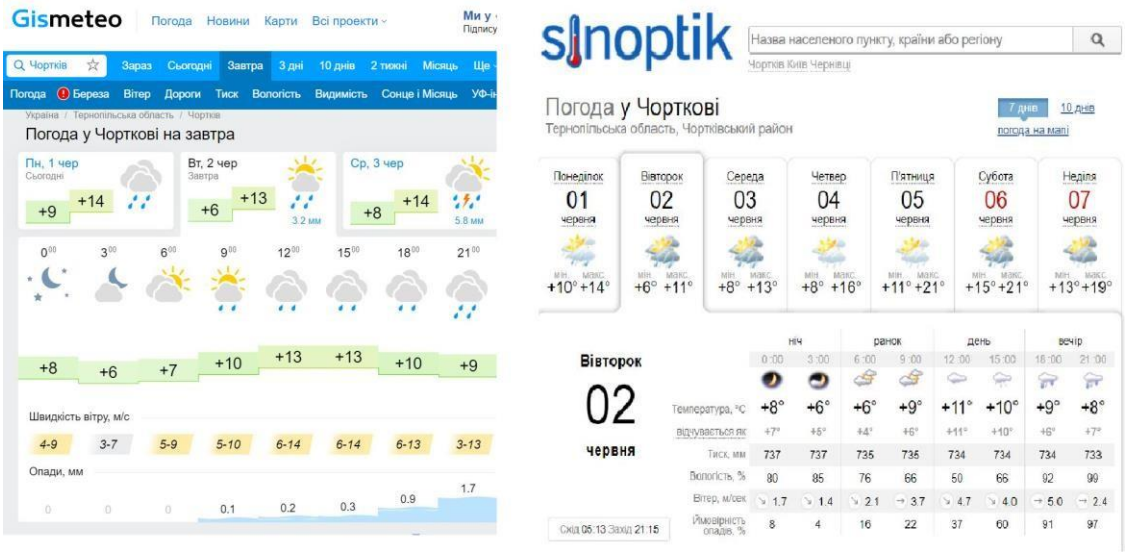


Рисунок 5.15 – Прогноз погоди у мережі Інтернет

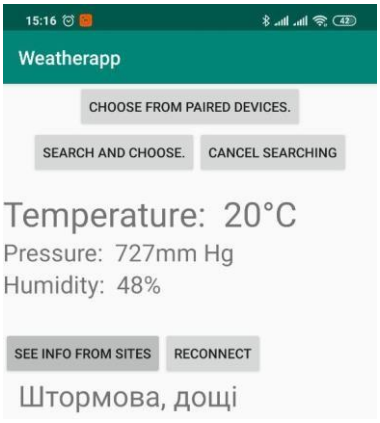


Рисунок 5.16 – Прогноз погоди у додатку

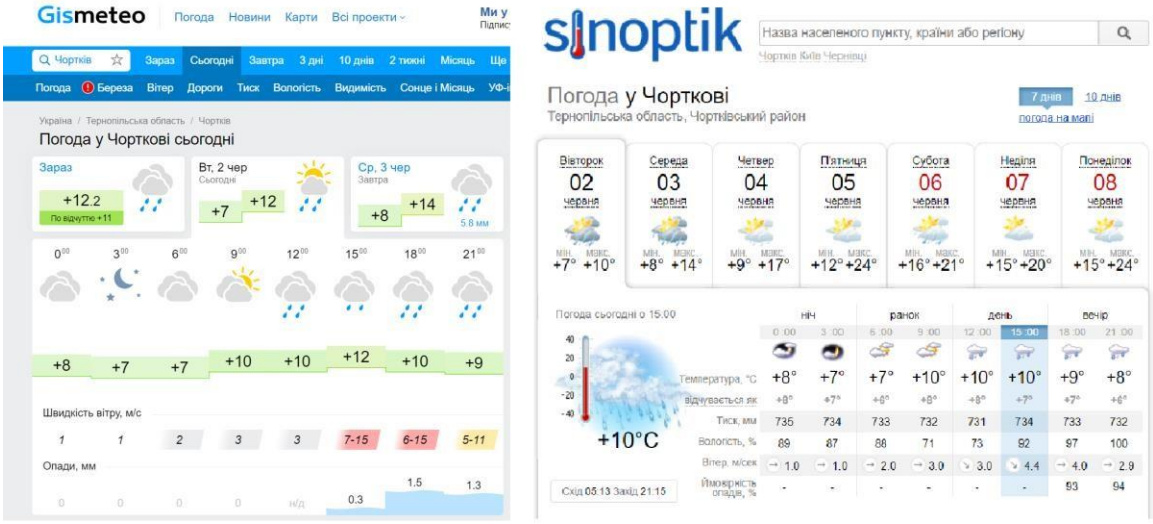


Рисунок 5.17 – Прогноз погоди у мережі Інтернет

Висновки до розділу

У цьому розділі описані результати роботи, тестування та порівняння прогнозу розробленої системи із прогнозом погодних сервісів.

Програмне забезпечення мобільного додатку та веб-сервера здатне обробляти будь-які вхідні дані (коректні та некоректні) і формувати відповідні повідомлення у інтерфейсі користувача. Інтерфейс є досить простим та інтуїтивно зрозумілим. Мобільний додаток не потребує великої обчислювальної здатності (ОС Android 4.4 чи вище) та багато місця у постійному запам’ятовуючому пристрої (до 4 Мб). Порівняння прогнозу, сформованого додатком та прогнозу із мережі Інтернет говорить про те, що додаток коректно визначає тенденцію зміни погоди. Прогноз у додатку є точнішим у деяких випадках, проте через відсутність комплексних даних про стан атмосфери чи через особливості місцевості іноді прогноз може не зовсім збігатись із реальністю. Однак такий недолік певною мірою присутній у всіх сучасних системах передбачення погоди.

ВИСНОВКИ

Метою цього проєкту було створення системи для зручного моніторингу та прогнозування погодних умов за допомогою смартфона. Система була розроблена із використанням мінімуму апаратних засобів та перенесенням усіх обчислень, налаштувань і т.д. у мобільний додаток. Завдяки цьому розроблений проєкт вийшов дешевшим та більш функціональним ніж його аналоги, які існують на ринку. Ще однією перевагою цього підходу є те, що у систему можна додавати нові можливості без необхідності змінювати апаратну частину, достатньо лише оновити додаток. Це можуть бути функції побудови графіків, інтеграція із системою розумного дому, тощо. Використання у проєкті сучасних технологій таких як зв'язок із хмарним сервером дозволить перенести ще більше обчислень у хмару, якщо у додатку з'явиться новий функціонал. Це дозволить зменшити системні вимоги додатку а також знизити споживання енергії.

Також у проєкті було реалізовано апаратну та програмну базу, яку можна розвинути у систему розумного дому, реалізувати інструменти керування цією системою. Такі системи зараз набувають популярності, тому розроблений продукт має потенціал.

При подальшому розвитку проєкту варто звернути особливу увагу на збільшення часу автономної роботи пристрою шляхом заміни bluetooth-передавача на модуль, який підтримує стандарт 4.0 LowEnergy, а також заміни модуля підвищення напруги на більш енергоефективний. Також можна переконфігурувати систему – передавати дані із датчика на стаціонарний хаб, а вже звідти на мобільні пристрої. Такий підхід допоможе збільшити радіус дії та кількість пристроїв, які можуть бути підключені одночасно. Також дані із датчика на хаб можна передавати радіомодулем, що працює на частоті 433 МГц – він споживає менше електроенергії. Можна реалізувати двосторонню передачу даних між пристроєм та додатком. Хоча у цьому на даний момент

немає необхідності, проте за бажання це можна втілити програмно. Якщо у систему додати карту пам’яті, то вона може збирати дані про погодні умови цілком автономно на час, поки вистачить заряду акумулятора.

Таким чином розроблена система продемонструвала свою високу функціональність, хорошу здатність до розширення та вдосконалення а також мінімальні вимоги до обчислювальної здатності як пристрою так і смартфону користувача.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Сторінка метеостанції EA2 OT300 у інтернет-магазині [Електронний ресурс]. Режим доступу : https://bt.rozetka.com.ua/ua/ea2_ot300/p17269150/ (дата звернення: квітень 2020).
2. Сторінка метеостанції EA2 BL503 SLIM у інтернет-магазині [Електронний ресурс]. Режим доступу : https://bt.rozetka.com.ua/ua/ea2-bl503/p272218/ (дата звернення: квітень 2020).
3. Веб-ресус компанії TFA [Електронний ресурс]. Режим доступу : http://tfa-dostmann.com.ua (дата звернення: квітень 2020).
4. Основы создания приложений [Електронний ресурс]. Режим доступу : https://developer.android.com/guide/components/fundamentals?hl=ru (дата звернення: квітень 2020).
5. Огляд Android Studio [Електронний ресурс]. Режим доступу : https://developer.android.com/distribute/best-practices/develop/build-with-android-studio?hl=ru (дата звернення: квітень 2020).
6. Network security configuration [Електронний ресурс]. Режим доступу : https://developer.android.com/training/articles/security-config (дата звернення: квітень 2020).
7. Understand the Activity Lifecycle [Електронний ресурс]. Режим доступу : https://developer.android.com/guide/components/activities/activity-lifecycle (дата звернення: квітень 2020).
8. Архитектура REST [Електронний ресурс]. Режим доступу : https://habr.com/ru/post/38730/ (дата звернення: квітень 2020).
9. Документація мікрофреймворка Flask [Електронний ресурс]. Режим доступу : https://flask.palletsprojects.com/en/1.1.x/ (дата звернення: квітень 2020).

10. Опис платформи Arduino nano [Електронний ресурс]. Режим доступу : http://arduino.ru/Hardware/ArduinoBoardNano (дата звернення: квітень 2020).
11. Веб-ресурс шини I2C [Електронний ресурс]. Режим доступу : https://www.i2c-bus.org (дата звернення: квітень 2020).
12. Последовательный интерфейс SPI [Електронний ресурс]. Режим доступу : http://www.gaw.ru/html.cgi/txt/interface/spi/index.htm (дата звернення: квітень 2020).
13. Докуметація модуля BME280 [Електронний ресурс]. Режим доступу : https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME280-DS002.pdf (дата звернення: квітень 2020).
14. Документація модуля HC-05 [Електронний ресурс]. Режим доступу : https://www.electronicaestudio.com/docs/istd016A.pdf (дата звернення: квітень 2020).
15. Документація модуля MT3608 [Електронний ресурс]. Режим доступу : https://www.mikrocontroller.net/attachment/212877/MT3608.pdf (дата звернення: квітень 2020).