МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних наук та прикладної математики

"До захисту допущена"
Зав. кафедри комп'ютерних наук та прикладної математики
д.т н., проф. Ю.В. Турбал
«____» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

Мобільний додаток для збору даних з датчиків і передачі їх на хмарний сервіс

Виконав: Анішкевич Михайло Васильович _____
(прізвище, ім'я, по батькові) (підпис)

група ІПЗ-41

Керівник: к.т.н., доцент, доцент Жуковська Н. А. _____
(науковий ступінь, вчене звання, посада, прізвище, ініціали) (підпис)

Рівне – 2025

Національний університет водного господарства та природокористування

(повне найменування вищого навчального закладу)

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії

Кафедра **комп'ютерних наук та прикладної математики**
Освітньо-кваліфікаційний рівень **бакалавр**
Галузь знань **12 Інформаційні технології**
(шифр і назва)
Спеціальність **121 Інженерія програмного забезпечення**
(шифр і назва)
Спеціалізація –

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
д.т н., проф. Ю.В. Турбал
«____» _____ 2025 р.

З А В Д А Н Н Я
НА КВАЛФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Анішкевич Михайло Васильович

(прізвище, ім'я, по батькові)

1. Тема роботи *"Мобільний додаток для збору даних з датчиків і передачі їх на хмарний сервіс"*

керівник роботи *Жуковська Наталія Анатоліївна, к.т.н., доцент, доцент кафедри комп'ютерних наук та прикладної математики.*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада)

затверджені наказом по університету від _____.

2. Термін подання роботи студентом _____.

3. Вихідні дані до роботи: *У процесі розробки використовуються наступні технології: мова програмування Java, середовище Android Studio, бібліотеки*

MPAndroidChart, OkHttp, org.json, хмарна платформа для Інтернету речей ThingSpeak. Для збору даних використовується апаратне забезпечення на базі мікроконтролера ESP8266, підключеного до датчиків температури та вологості (DHT22) та вологості ґрунту (YL-69).

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *Розробити прототип мобільного рішення для збору, збереження та візуалізації сенсорних даних.*

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

Мультимедійна презентація

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1	*доцент Жуковська Н. А.*	*10.10.24*	*10.10.24*
Розділ 2	*доцент Жуковська Н. А..*	*15.01.25*	*15.01.25*
Розділ 3	*доцент Жуковська Н. А.*	*04.03.25*	*04.03.25*

7. Дата видачі завдання *« » 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Вивчення літератури за обраною тематикою*	*02.10.24 – 20.10.24*	*виконав*

2	*Розробка та підключення arduino*	*23.10.24 – 08.12.24*	*виконав*
3	*Створення сторінок та їх заповнення вибір налаштування початок вивчення android studio*	*11.04.25 – 15.05.25*	*виконав*
4	*Підключення додатки до хмари. Додаткові налаштування*	*15.05.25 – 01.06.25*	*виконав*
5	*Підготовка звіту магістерської роботи; Підготовка мультимедійної презентації*	*01.06.25 – 9.06.25*	*виконав*

Студент _____ **Анішкевич М.В.**

(підпис) (прізвище та ініціали)

Керівник роботи _____ **Жуковська Н. А.**

(підпис) (прізвище та ініціали)

ЗМІСТ

Реферат

Кваліфікаційна робота: 45 с., 22 рисунки, 12 джерел.

Мета роботи: розробка мобільного додатку для моніторингу даних, що передаються з сенсорів на платформу ThingSpeak. Метою також є забезпечення зручного інтерфейсу для користувача, можливості перегляду як поточних, так і історичних значень, адаптивності до різних типів сенсорів та використання сучасних підходів до візуалізації інформації.

Об'єкт дослідження: процес обробки, збереження, передачі та візуалізації сенсорних даних у мобільному середовищі з використанням хмарної платформи.

Предмет дослідження: програмна система у вигляді Android-додатку, що забезпечує зчитування, обробку, передачу, зберігання та представлення даних, отриманих від сенсорів, у вигляді таблиць та графіків, а також взаємодію з інтернет-сервісом ThingSpeak.

Методи та засоби: у роботі використано мову програмування Java для створення логіки додатку, бібліотеку MPAndroidChart для побудови графіків, Android SDK для реалізації користувацького інтерфейсу, OkHttp для мережевих запитів, а також REST API ThingSpeak для обміну даними. Для збереження налаштувань застосовано SharedPreferences.

Ключові слова: Інтернет речей , платформа ThingSpeak, Android, сенсорні пристрої, мобільний додаток, візуалізація даних, REST API.

ВСТУП

Дана теми полягає в створення android-додатка для здійснювати моніторингу середовища за допомогою датчиків. Інтеграція датчиків у глобальну мережу надає змогу контролювати та регулювати важливі процеси у режимі реального часу. Особливо важливо це у сферах екологічного моніторингу, розумних будинках, сільського господарства, промисловості також допомагає при енергозбереження . Широке впровадження мобільних технологій сприяє підвищенню зручності у користуванні та доступності таких сервісів для користувачів на телефонах.

Однією з платформ, що активно використовується для збору й аналізу сенсорних даних, є ThingSpeak. Це хмарне рішення, що дозволяє приймати дані з віддалених пристроїв через HTTP-запити та зберігати їх у спеціальних каналах. За допомогою цієї платформи можливо реалізувати повноцінну систему моніторингу, до якої можна підключати як мікроконтролери (ESP32, Arduino) які передають дані на ThingSpeak і для її перегляду.

Метою даної роботи є створення Android-додатку, який дозволяє зчитувати, зберігати, візуалізувати та аналізувати дані з сенсорів через платформу ThingSpeak. Такий додаток дає змогу в реальному часі переглядати показники сенсорів, аналізувати історичні дані, адаптувати інтерфейс відповідно до вподобань користувача (темна/світла тема), а також змінювати список типів сенсорів.

Для досягнення поставленої мети у роботі визначено наступні завдання:

• аналіз чи є доступні рішеня у сфері мобільного моніторингу IoT-даних;

• проектування архітектури додатку відповідно до вимог зручності, гнучкості та масштабованості всі сторінки створенні у лінії;

• реалізація програмної частини з використанням мови Java, C++ програми Android Studio, Arduino;

проведення тестування додатку та аналіз його функціональних можливостей, зокрема візуалізація даних у вигляді графіків та таблиць.

У ході дослідження використовувалися такі методи: структурний аналіз, об'єктно-орієнтоване програмування, принципи мережевої взаємодії, обробка JSON-даних, побудова діаграм та використання сторонніх бібліотек MPAndroidChart, OkHttp. Також реалізовано роботу зі бібліотекою SharedPreferences яка зберігає дані і не оновлюється при перезавантаження програми. Таким чином, розробка подібного додатку сприяє розуміння використання android studio та взаємодію між додатком і серверу за допомогою JSON, OkHttp . Отримані результати можуть бути застосовані як у навчальних цілях, так і при реалізації реальних проектів у сфері Інтернету речей.

Ще важливо для додатка є зручність використання та безпека. Надійна передача даних із сенсорів до хмарного сервісу, їхнє збереження та подальша обробка у додатку є ключовим завданням сучасних мобільних систем моніторингу. Застосування JSON допомагає та спрощує програмування HHTP дає змогу вдосконалювати платформу для користувачів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Сучасні хмарні сервіси забезпечують ефективне зберігання, обробку та візуалізацію даних із мікроконтролерів. У нашому проекті не є важливим вибір мікроконтролера. Найбільш поширеним серед мікроконтролерів є Arduino як: Arduino UNO, Nano, Mega. Вони популярні у простоті в написані за допомогою мови програмування C++. Також великою кількості бібліотеки. Також у нашому проекту можна було використати та інші мікроконтролери, вони мають мати можливість передавати HTTP як:

- ESP8266 – підтримує підключення до Wi-Fi, сумісний з Arduino, передачу даних через HTTP;
- ESP32 – потужніший варіант ESP8266, підтримує Wi-Fi та Bluetooth, дозволяє використовувати ще HTTPS;
- TM32 — високопродуктивні мікроконтролери для складних завдань, підтримують USB, UART, SPI та інші протоколи.
- ATmega328 — той самий чіп, що використовується в Arduino UNO. Доступний окремо для кастомних плат.
- Raspberry Pi Pico W – мікроконтролер від Raspberry з підтримкою Wi-Fi, програмується як у MicroPython, так і в C/C++.

У цьому проекту ми користувалися Arduino UNO та wifi адаптер esp8266 (не мікроконтролер).

У світі є багато хмарних сервісів. Розглянемо найпопулярніші з них.

1. Firebase - це платформа від Google, вона надає можливість зберігання даних та переглядати дані у реальному часі. Хмарних функцій, аналітики, а також використання Firestore та Cloud Functions для більш динамічної взаємодії між компонентами. Firebase має можливість інтеграцію з Android

Studio. Однак вона підтримує лише HTTPS-з’єднання, що ускладнює інтеграцію з ESP8266 без додаткових налаштувань чи проксі-сервера.

2. Blynk - це хмарна система для контролю за мікроконтролерів переглядати також дані за допомогою графіками, дисплеями. Платформа має можливість передавати дані за допомогою HTTP, так і MQTT протоколи. Це дозволяє прасувати з багатьма пристроями. Її перевага — простота і зручність у використання .
3. Adafruit IO - це хмарна система, що дозволяє переглядати дані з сенсорів, але й взаємодіяти з ними за допомогою тригерів. Надає можливість передачі MQTT і REST API. Має готові приклади для Arduino, Raspberry Pi та ESP-серії. Призначена для швидкого прототипування та освітніх проєктів. Має обмеження на кількість запитів у безкоштовному плані.
4. ThingSpeak — це відкрита платформа, що дозволяє збирати дані з сенсорів також візуалізувати у вигляді дані через HTTP-запити. Платформа підтримує роботу з одного до восьми полів на один канал. Переваги ThingSpeak — підтримка HTTP-запитів (підходить для wifi адаптер ESP8266), простий API, безкоштовний доступ до базового функціоналу та зручна інтеграція з мобільними клієнтами.

У розділі проведено аналіз IoT-платформ, які використовуються для зберігання та обробки даних з мікроконтролерів і датчиків навколишнього середовища. Зокрема, було відкинені платформи які розглядалися крім двох платформ для обробки даних на хмари: firebase та ThingSpeak. Firebase пропонує широкі можливості для зберігання та обробки даних у реальному часі, має більші можливості для налаштувань взаємодії додатка і хмарою. Проте ThingSpeak є простішим і підтримує HTTP-з'єднання. Firebase має недолік є вимога до використання захищеного протоколу HTTPS, що створює технічні складнощі при роботі з деякими мікроконтролерами, зокрема wifi адаптер ESP8266, який за замовчуванням підтримує лише HTTP-з'єднання. Існують способи додати підтримку HTTPS до ESP8266, але це вимагає додаткових налаштувань,

складніших бібліотек або використання проміжних серверів, що не відповідає критеріям простоти й автономності пристрою.

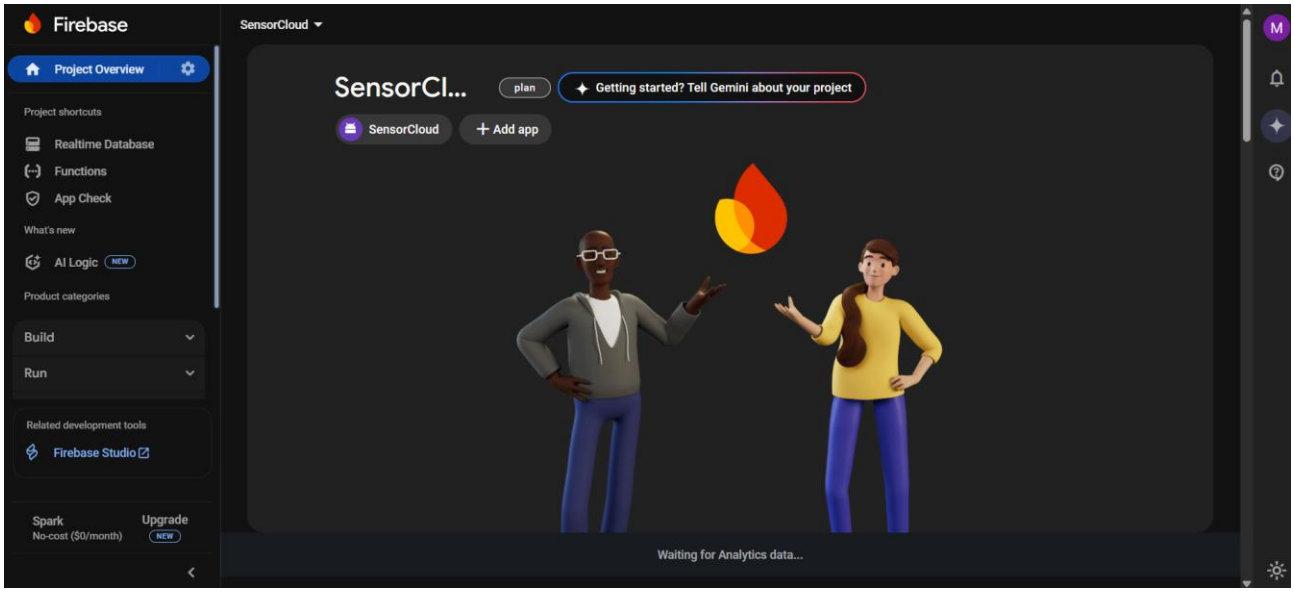


Рис. 1.1. Сторінка консоль firebase

У зв'язку з цим, для реалізації практичної частини проєкту було обрано платформу ThingSpeak. Вона дозволяє зберігати та обробляти дані сенсорів, приймаючи їх через звичайні HTTP-запити, що значно спрощує реалізацію з боку пристроїв. Крім того, ThingSpeak підтримує візуалізацію даних, що спрощує подальшу інтеграцію з мобільним додатком. ThingSpeak також забезпечує обробку до восьми полів у межах одного каналу, що зручно при підключенні кількох сенсорів одночасно.

Окрему увагу приділено Android-платформі як середовищу для розробки мобільних додатків, які дозволяють взаємодіяти з IoT-платформами. Розглянуто архітектури REST API, принципи побудови клієнт-серверної взаємодії, структуру HTTP-запитів, зокрема методи GET, POST, а також обробку відповідей у форматі JSON для оновлення даних для користувача.

.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі описано структуру мобільного додатку. Додаток має 4 сторінки які призначенні для того щоб відображати дані з сенсорів за допомогою HTTP протоколу. Також сторінки мають можливість для налаштування додатка. Інтерфейс в HTML розмітки скрізь використовується лінійне компонування для того щоб мати можливість переглядати на різних екранах різного розміру телефона.

1. **Головна сторінка (MainActivity):** відображає поточні значення з ThingSpeak у вигляді текстових полів для кожного сенсора. Дані надходять у форматі JSON, який за допомогою вбудованих методів Android відобажаються.

Основні функції:

1. Зчитування останнього запису з ThingSpeak API (feeds/last.json);
2. Відображення значень field1–field8 у вигляді назва теперішнє або останнє числове значення;
3. Динамічне формування інтерфейсу на основі локального списку сенсорів.

Рис. 2.1. Головна сторінка

2. **Таблиця і графіка сторінка (TableActivity):** візуалізація історичних даних, отриманих з ThingSpeak, у вигляді графіка за допомогою бібліотеки MPAndroidChart. Крім того, реалізовано таблицю з часовими мітками і відповідними значеннями сенсорів, що дозволяє здійснювати аналіз динаміки зміни даних;

Основні функції:

- Вибір сенсора через Spinner з локального списку;

- Відображення та побудови лінійних графіків;

- Виведення таблиці у вигляді дата та числове значення;
- Формування запитів з урахуванням прив’язки сенсора до поля (field).

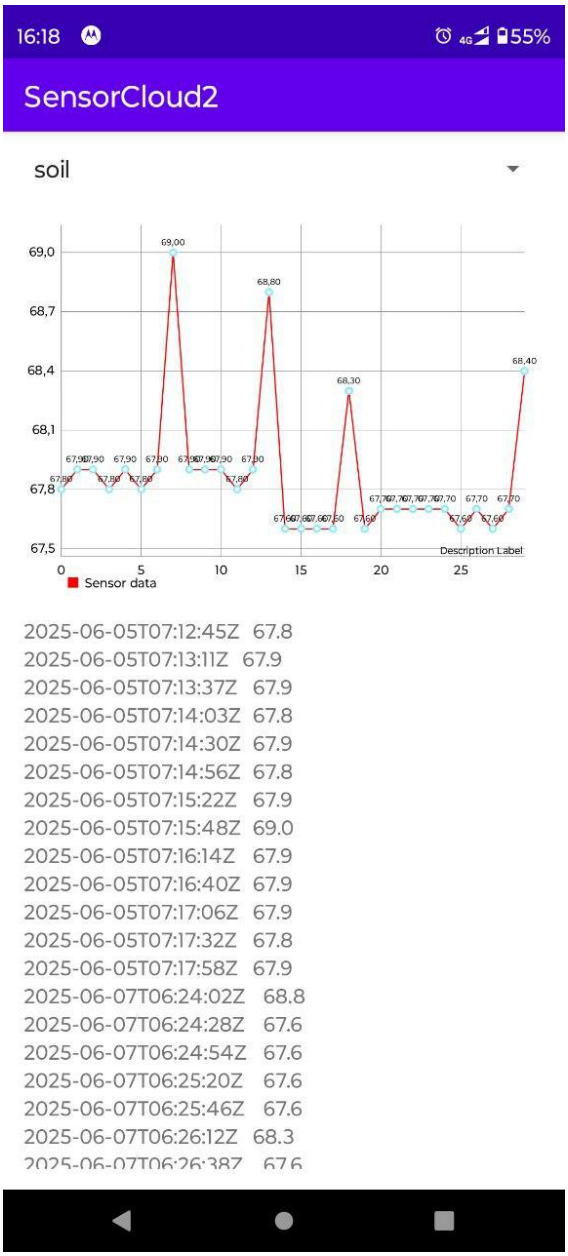


Рис. 2.2. Таблиця і графік

3. **Налаштування (SettingsActivity):** введення та збереження параметрів підключення до ThingSpeak, зокрема channel ID та API key. Також реалізовано перемикач темного/світлого режиму, який дозволяє користувачеві адаптувати інтерфейс відповідно до власних уподобань.

Тематичний режим зберігається у SharedPreferences і застосовується через AppCompatDelegate;

Основні функції:

- Збереження ключів для ThingSpeak через SharedPreferences;
- Вибір білої чи світлої теми AppCompatDelegate;
- Перезавантаження сторінок для миттєвого оновлення теми інтерфейсу.

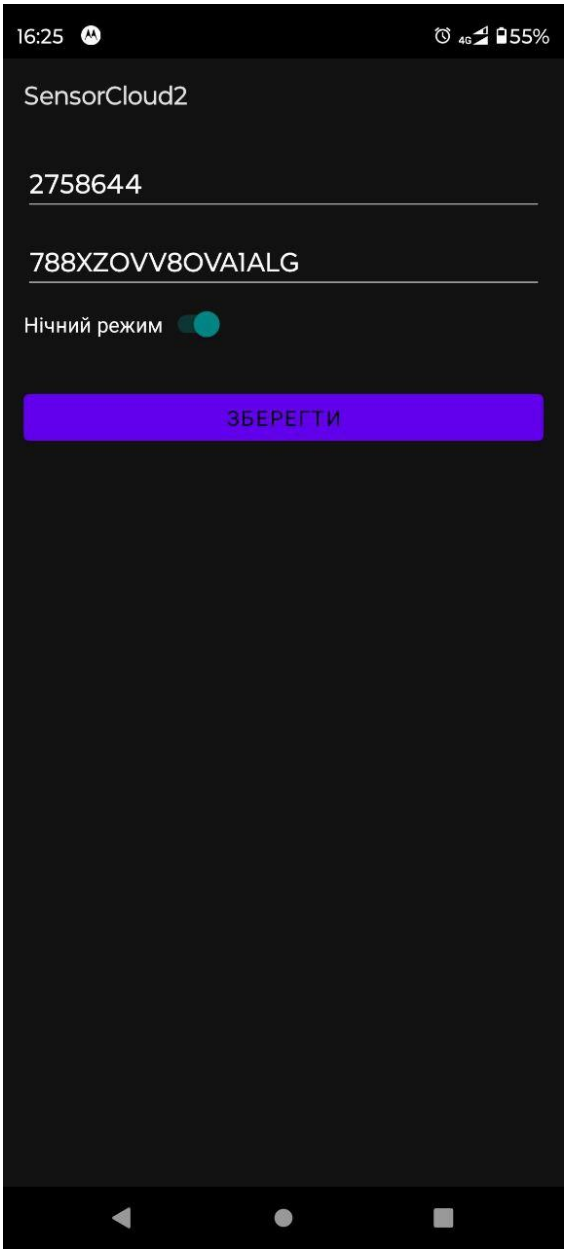


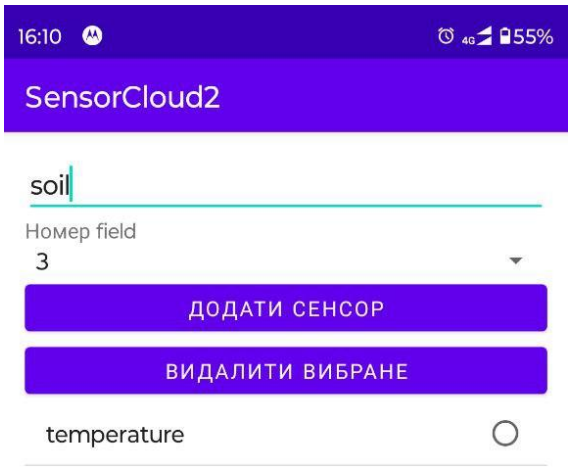
Рис. 2.3. Сторінка налаштування

4. **Додавання сенсорів (AddSensorActivity**): редагування списку типів сенсорів, що будуть відображатись у додатку. Реалізовано механізм

додавання нових назв сенсорів, їх видалення та збереження у локальному сховищі пристрою. Список використовується для формування вибору в спінері та в побудові запитів до ThingSpeak.

Основні функції:

- Додавання нових назв сенсорів та вибір поля(1–8);
- Видалення існуючих сенсорів;
- Збереження у локальне сховище (SensorPrefs) у форматі JSON через SensorMapping;
- Синхронізація між сторінками додатку за допомогою загального списку сенсорів.



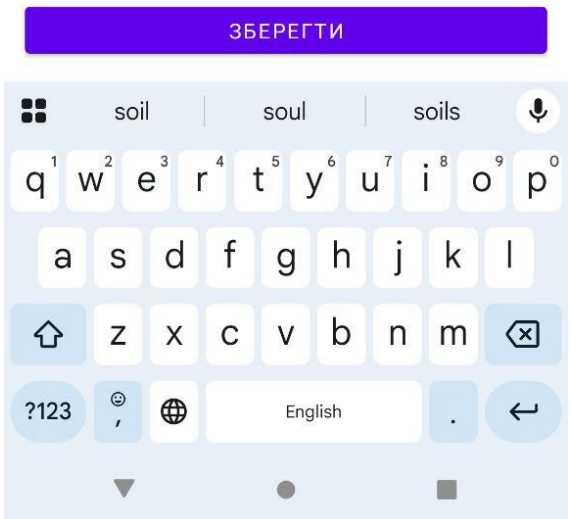


Рис. 2.4. Сторінка для додавання сенсорів

Реалізацію мобільного додатку виконано мовою Java з використанням Android SDK та низки додаткових бібліотек. У розробці використано об'єктно-орієнтований підхід із дотриманням принципів модульності та масштабованості.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

Реалізацію мобільного додаткувиконано виконано мовою Java з використанням Android studio та низки додаткових бібліотек .

1. **ThingSpeak** - це хмарна платформа для зберігання зберігання та візуалізації даних з датчиків. Вона дозволяє обробляти дані з датчиків через HTTp , зберігати їх у каналах та переглядати через веб- інтерфейс. У проєкті використовувалася для отримання отримання та обробки даних, надісланих з мікроконтролера ESP8266 .
2. **JSON (JavaScript Object Notation**) - це легкий текстовий формат обміну даними, який широко використовується в API. Він складається з пар ключ-значення і підтримується більшістю мов програмування. В Android обробка JSON здійснюється за допомогою класів JSONObject і JSONArray, які дозволяють зручно витягувати необхідні поля з відповіді сервера.
3. **OkHttp** - це потужна Java /Android бібліотека для виконання HTTP- запитів . допомагає і спрощує у взаємодії між додатком і хмарою. У проекті вона використовується для надсилання запитів доThingSpeak та обробки відповідей.
4. **SharedPreferences** - це збереження простих даних в локальному сховощі . Він є частиною Android SDK і використовується для зберігання налаштувань користувача , стану перемикачів , обраної теми, API- ключів та інших даних, які мають зберігатися між сесіями. Ця бібліотека забезпечує доступ до даних.

Основні функції реалізовані в рамках таких компонентів:

1. **Підключення до платформи ThingSpeak:** для взаємодії з API ThingSpeak використовувався протокол HTTPS і бібліотека **OkHttp**. Формування URL-запиту відбувається на основі параметрів, збережених у SharedPreferences (channel ID і API key), а виклик виглядає так:

String url = "https://api.thingspeak.com/channels/" + Settings.channelId + "/fields/" + getFieldNumber(sensorType) + ".json?api_key=" + Settings.apiKey + "&results=30";

Запит надсилається за допомогою клієнта:

OkHttpClient client = new OkHttpClient();

Request request = new Request.Builder().url(url).build();

2. **Обробка відповіді у форматі JSON**: дані з сервера ThingSpeak надходять у форматі **JSON**. Для прикладу, в MainActivity використано метод optString("fieldX"), де X — номер поля, прив'язаний до конкретного сенсора. Назви сенсорів і відповідність полів зберігаються в окремому класі SensorMapping, що дозволяє виводити значення у зручному для користувача вигляді.

```
String responseStr = response.body().string();
JSONObject json = new JSONObject(responseStr);
List<String> sensorNames = SensorMapping.getSensorList(MainActivity.this);
runOnUiThread(() -> {
        sensorsLayout.removeAllViews();
        for (String name : sensorNames) {
            int fieldNum = SensorMapping.getFieldNumber(MainActivity.this, name);
            String fieldKey = "field" + fieldNum;
            String value = json.optString(fieldKey);
            if (value != null && !value.equals("null")) {
                TextView sensorView = new TextView(MainActivity.this);
                sensorView.setText(name + ": " + value);
                sensorView.setTextSize(18);
                sensorsLayout.addView(sensorView);
            }
        }
```

```
        });
```

3. **Графіки:** для побудови графіків використовувалась бібліотека MPAndroidChart, яка є потужним засобом для візуалізації даних у вигляді лінійних, стовпчикових, кругових діаграм тощо. У проєкті реалізовано лінійний графік (LineChart), що дозволяє відображати зміну значень сенсора у часі. Дані з ThingSpeak парсяться у форматі List<Entry>, де кожна точка представляє порядковий індекс (X) і значення сенсора (Y). Побудова графіка здійснюється через створення LineDataSet, який потім передається до LineData, після чого графік оновлюється через lineChart.setData(...) і

```
private void updateChart(List<Float> values) {
    List<Entry> entries = new ArrayList<>();
    for (int i = 0; i < values.size(); i++) {
        entries.add(new Entry(i, values.get(i)));
    }
    LineDataSet dataSet = new LineDataSet(entries, "Sensor data");
    dataSet.setColor(getResources().getColor(R.color.red));
    LineData lineData = new LineData(dataSet);
    lineChart.setData(lineData);
    lineChart.getXAxis().setPosition(XAxis.XAxisPosition.BOTTOM);
    lineChart.getAxisRight().setEnabled(false);
    lineChart.invalidate();
}
```

У наведеному прикладі графік будується на основі масиву числових значень values, які попередньо були зчитані з API ThingSpeak. Кожне значення асоціюється з порядковим індексом, що виступає координатою по осі X, а саме “i” — по осі Y. Бібліотека MPAndroidChart автоматично створює точки на основі цих пар і з'єднує їх лініями для формування графіка.

4. **Таблиці:** таблиця значень реалізована динамічно через додавання елементів TextView до контейнера LinearLayout. Кожен рядок містить мітку часу

(created_at) та відповідне значення сенсора. Це дозволяє не лише бачити зміну параметрів на графіку, але й переглядати точні числові значення. Після вибору сенсора у Spinner, додаток автоматично будує таблицю з останніх 30 отриманих значень через ThingSpeak.

```
private void updateTable(List<String> times, List<Float> values) {
    tableLayout.removeAllViews();
    for (int i = 0; i < times.size(); i++) {
        TextView row = new TextView(this);
        row.setText(times.get(i) + "\t\t" + values.get(i));
        tableLayout.addView(row);
    }
}
```

У цьому прикладі таблиця формується шляхом додавання нових рядків даних до контейнера tableLayout. Для кожного запису створюється елемент TextView, який містить часову мітку та відповідне значення , отримане з JSON-відповіді з хмари. Всі елементи додаються до вертикального , що забезпечує перегляду таблиці на телефонах з малими екранами . Цей підхід дозволяє швидко сформувати таблицю без використання складних налаштувань хоча вона передбачена для малою кількості даних .

5. **Налаштування сенсорів:** реалізовано в окрему сторінки AddSensorActivity, де можна додавати або видаляти сенсори, вказуючи їхню назву та відповідне поле ThingSpeak (field1–field8). Для цього використано поле вводу (EditText), випадаючий список (Spinner) з номерами полів і список (ListView) для перегляду поточних сенсорів. При натисканні кнопки "Додати" виконується перевірка, чи не порожня назва сенсора та чи ще назва не співпадає з списку якого є вже. Якщо все коректно, викликається метод SensorMapping.saveMapping(...), який зберігає пару назва-поле у вигляді JSON у SharedPreferences.

```
addButton.setOnClickListener(v -> {
```

```
            String newSensor = sensorInput.getText().toString().trim();
            int selectedField = (int) fieldSpinner.getSelectedItem();
            if (!newSensor.isEmpty() && SensorMapping.getFieldNumber(this,
newSensor) == -1) {
                SensorMapping.saveMapping(this, newSensor, selectedField);
                sensorList.clear();
                sensorList.addAll(SensorMapping.getSensorList(this));
                adapter.notifyDataSetChanged();
                sensorInput.setText("");
            } else {
                Toast.makeText(this, "Сенсор вже існує або порожній",
Toast.LENGTH_SHORT).show();
            }
        });
```

Для видалення сенсора користувач обирає елемент у списку, після чого натискає кнопку "Видалити". Запис видаляється з sensorList та з інтерфейсу. Кнопка "Зберегти" записує оновлений список у SharedPreferences за допомогою методу saveSensorList().

6. **Темна та світла тема:** у SettingsActivity користувач має можливість змінювати тему оформлення додатку. Значення теми зберігається у PreferenceManager і застосовується через AppCompatDelegate.setDefaultNightMode(...). Зміна теми застосовується одразу після перемикання, завдяки методу recreate().

```
    themeSwitch = findViewById(R.id.themeSwitch);
        themeSwitch.setChecked(isDarkMode);

        themeSwitch.setOnCheckedChangeListener((buttonView, isChecked) -> {
            SharedPreferences.Editor editor = prefs1.edit();
            editor.putBoolean("dark_mode", isChecked);
```

```
            editor.apply();

            AppCompatDelegate.setDefaultNightMode(
                isChecked ? AppCompatDelegate.MODE_NIGHT_YES :
AppCompatDelegate.MODE_NIGHT_NO
            );
            recreate();
        });
```

7. **Збереження конфігурації:** усі налаштування, зокрема параметри підключення до ThingSpeak, тема оформлення, та сенсори, зберігаються у локальному сховищі SharedPreferences. Це дозволяє зберегти індивідуальні налаштування між сесіями користувача.

```
    public static void load(Context context) {
        SharedPreferences prefs =
context.getSharedPreferences("ThingSpeakSettings", Context.MODE_PRIVATE);
        channelId = prefs.getString("channelId", channelId);
        apiKey = prefs.getString("apiKey", apiKey);
    }

    public static void save(Context context) {
        SharedPreferences prefs =
context.getSharedPreferences("ThingSpeakSettings", Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = prefs.edit();
        editor.putString("channelId", channelId);
        editor.putString("apiKey", apiKey);
        editor.apply();
    }
```

Ці методи викликаються з будь-якої активності: наприклад, під час старту MainActivity для завантаження конфігурації, або в SettingsActivity після

натискання кнопки "Зберегти", щоб зберегти нові значення. Такий підхід гарантує, що навіть після перезапуску додатка користувацькі параметри буде збережено й використано автоматично.

Після повернення з інших активностей (AddSensorActivity, SettingsActivity) викликається метод onResume(), у якому оновлюються дані для ThingSpeak. Це забезпечує актуальні дані.

```
@Override
  protected void onResume() {
     super.onResume();
     Settings.load(this);
     loadCurrentData();
  }
```

Таким чином, мобільний додаток реалізовано, що інтегрується з платформою ThingSpeak, з налаштуваннями для виберу теми чи вибір кодів до хмари, підтримкою збережених параметрів і широкими візуалізації у вигляді графіка та таблиці. Код написано з урахуванням повторного використання, для можливого вдосконалення.

8. **Код Arduino:** Передача дані за допомогою коду на Arduino для підключення до ThingSpeak здійснюється за допомогою.

```
String postData = "api_key=" + apiKey;
 postData += "&field1=" + String(temp);
 postData += "&field2=" + String(humidity);
 postData += "&field3=" + String(h_gr);

 espSerial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80");
 delay(2000);

 String request = "POST /update HTTP/1.1\r\n";
 request += "Host: api.thingspeak.com\r\n";
```

request += "Connection: close\r\n";

request += "Content-Type: application/x-www-form-urlencoded\r\n";

request += "Content-Length: " + String(postData.length()) + "\r\n\r\n";

request += postData;

На початку водимо API коду для введення даних:

String apiKey = "DI2Q2DTAS3X8Q9W3";

Потім загружаємо дані на field окремо. За допомогою АТ комант налаштовуємо ESP8266 потім вводимо HTTP регістр для посилання на ThingSpeak. В postData числові параметри.

Схема взаємодії компонентів Android-додатку:

1. User Interface (UI):
   - MainActivity - показує поточні значення сенсорів;
   - TableActivity - візуалізує останні 30 значень (графік + таблиця);
   - AddSensorActivity - додає сенсори та прив'язує їх до field;
   - SettingsActivity - налаштування API та теми.
2. Локальне сховище:
   - SharedPreferences:
     - зберігає channelId і apiKey (через клас Settings);
     - зберігає відповідності сенсор-поле (через клас SensorMapping);
     - зберігає режим теми (через PreferenceManager).
3. Обробка даних:
   - Запити виконує OkHttpClient;
   - Відповіді у форматі JSON обробляються класами JSONObject, JSONArray.
4. Підключення до інтернет-сервісу:

- Дані отримуються з ThingSpeak API:
- /feeds/last.json для поточного значення;
- /fields/30.json для останніх 30 значень.

5. Візуалізація:
   - MPAndroidChart в TableActivity - побудова графіка;
   - LinearLayout з TextView - побудова таблиці значень.

6. Зовнішній пристрій:
   - ESP8266 надсилає дані до ThingSpeak через HTTP POST.

Схема коду Java та XML коду:

- Java-класи:

1 MainActivity;
2 TableActivity;
3 AddSensorActivity;
4 SettingsActivity;
5 SensorMapping;
6 Settings;

- XML-файли інтерфейсу:

1 activity_main.xml;
2 activity_table.xml;
3 activity_add_sensor.xml;
4 activity_settings.xml;

Компоненти:

- MainActivity - Основний екран: показ поточних значень сенсорів з ThingSpeak.

- TableActivity - Відображає останні 30 значень у вигляді графіка (MPAndroidChart) та таблиці.
- SettingsActivity - Дозволяє змінювати тему, API key, channelId.
- AddSensorActivity - Додає/видаляє сенсори. Зберігає відповідність назва до field.
- Settings - Клас для збереження та завантаження API-ключа і channelId через SharedPreferences.
- SharedPreferences – база даних де зберігають різні параметри назва сенсора і його відповідний field також ключи до ThingSpeak.
- SensorMapping - Клас для зберігання назв сенсорів та його field у SharedPreferences для його використання по всьому коду.
- ThingSpeak - Хмарний сервіс для прийому/віддачі даних.
- ESP8266 + сенсори + Arduino – комплекс для сбору даних та передачні на хмарний сервіс.
- MPAndroidChart – бібліотека для графіків.

РОЗДІЛ 4
ТЕСТУВАННЯ, РЕЗУЛЬТАТИ ТА МОЖЛИВСТЬ ВДОСКОНАЛЕННЯ

Можливості для майбутнього розширення:

- Підтримка HTTPS у ESP-модулях або заміна ESP8266 на ESP32 для сумісності з Firebase. ESP8266 не підтримує безпечні з'єднання (HTTPS) на базовому рівні. Перехід на ESP32 дозволить використовувати захищені з'єднання з Firebase, що відкриє можливість передачі даних у більш сучасні хмарні сервіси з автентифікацією. За допомогою того можна здійснювати інтеграцію з Firebase Realtime Database або Firestore для гнучкішого збереження даних. Це дозволить не лише зберігати історію значень, а й додавати ієрархічні структури (наприклад, користувачі, проєкти, унікальні ідентифікатори пристроїв) з можливістю багатокористувацького доступу.
- Можливість для додавання та виберу різних типів графіків:
  1. Стовпчикових
  2. Кругових
  3. Комбінованих

  За допомогою цього можна глянути дані у зручному для тебе вигляді. Як приклад вологість води у вигляді лінійної діаграми, а вологість у стовпчиках де максимальне значення рівне 100.
- Можливість отримувати повідомлень коли параметри досягають критичної точки. Користувач може отримувати сповіщення на телефон, якщо температура або вологість перевищує критичні значення. Це дозволить використовувати систему як елемент моніторингу або попередження в реальному часі.
- Введення профілів користувача з авторизацією. Це забезпечить збереження особистих налаштувань для кожного користувача, підтримку кількох акаунтів, обмеження доступу до конфіденційних даних та захист каналу API.

- Розширення додатку веб-інтерфейсом або панеллю адміністратора. Через браузер користувач зможе керувати каналами, переглядати графіки, редагувати дані, створювати звіти. Це розширить сферу застосування рішення.
- Реалізація можливості змінювати дані у ThingSpeak безпосередньо з мобільного додатку. Наприклад, можна надсилати тестові значення або керувати конфігураційними параметрами каналу (назви полів, статус приватності тощо) через HTTP POST-запити прямо з інтерфейсу Android-додатку.

Умови тестування

1. Мобільний додаток протестовано на реальному пристрої з Android 13.
2. ESP8266 підключено до Wi-Fi мережі й налаштовано на надсилання даних у ThingSpeak через HTTP POST-запити. Пристрій стабільно надсилав дані з DHT22 та датчика вологості ґрунту з інтервалом 20 секунд, що відповідало обраному режиму реального часу.

На рисунку відображено:

- ESP8266 - Wi-Fi адаптер
- DHT22 - сенсор вологості повітря та температури
- YL-69 - датчика вологості ґрунту
- Arduino UNO - плата мікроконтролерів

також схему їхнього з’єднання

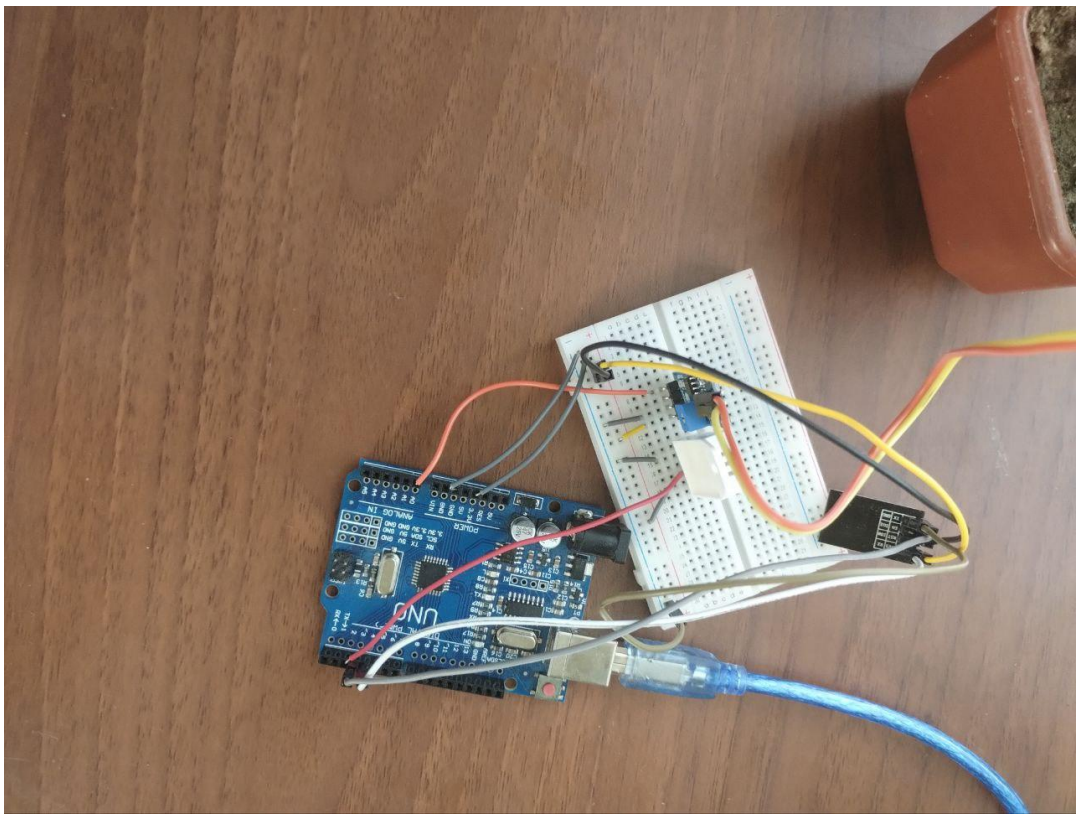


Рис. 4.1. приклад Arduino з ESP8266

3. Дані сенсорів надходили в режимі реального часу з інтервалом у 20 секунд. Інтервал налаштовано в коді прошивки ESP, і він витримувався стабільно. Дані з’являлися у ThingSpeak через 2–3 секунди після відправлення, що відповідає очікуваній затримці.

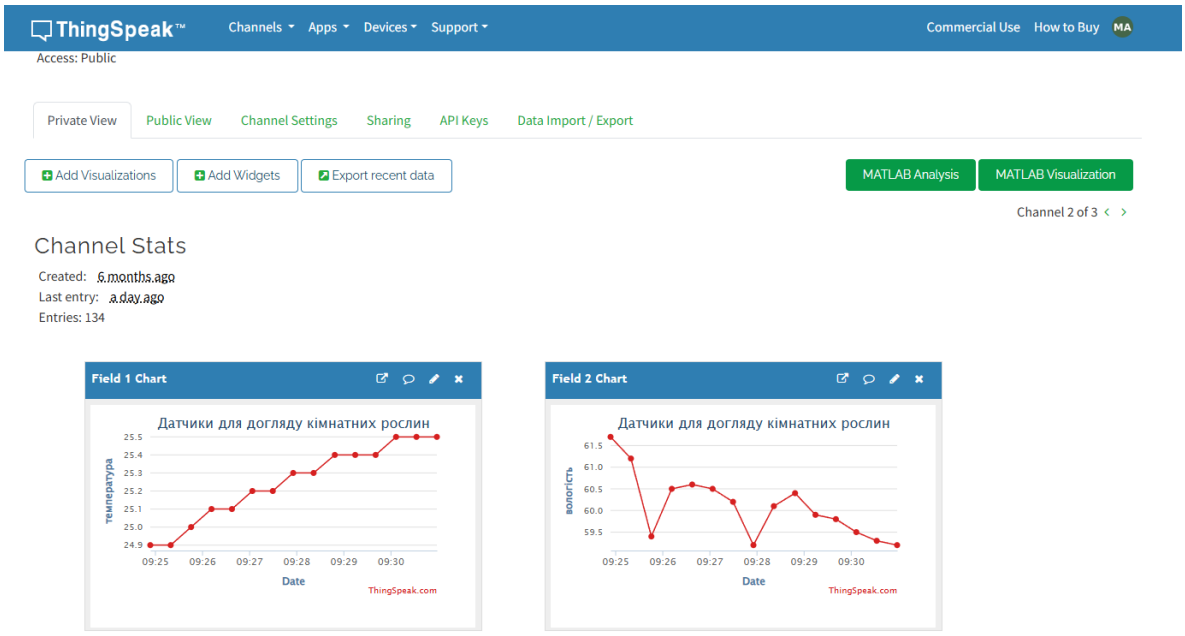


Рис. 4.2. ThingSpeak

4. Перевірка виконувалася за стабільного інтернет-з’єднання та у режимі офлайн. У разі втрати інтернету додаток адекватно повідомляє про помилку (Toast-повідомлення). У разі повторного з’єднання дані автоматично оновлюються через метод onResume().

Результати тестування функціональності

- Завантаження даних: MainActivity коректно відображає поточні значення сенсорів із ThingSpeak. Перевірено коректність обробки JSON, правильність співставлення полів та назв сенсорів, відсутність дублікатів і збоїв при порожньому або невалідному відповіді.

Рис. 4.3. головна сторінка

- Візуалізація: у TableActivity будуються графіки та таблиці для кожного сенсора. Графік оновлюється після вибору типу сенсора, таблиця коректно формує 30 останніх значень. Використання MPAndroidChart дозволило забезпечити високу швидкодію при великій кількості точок.

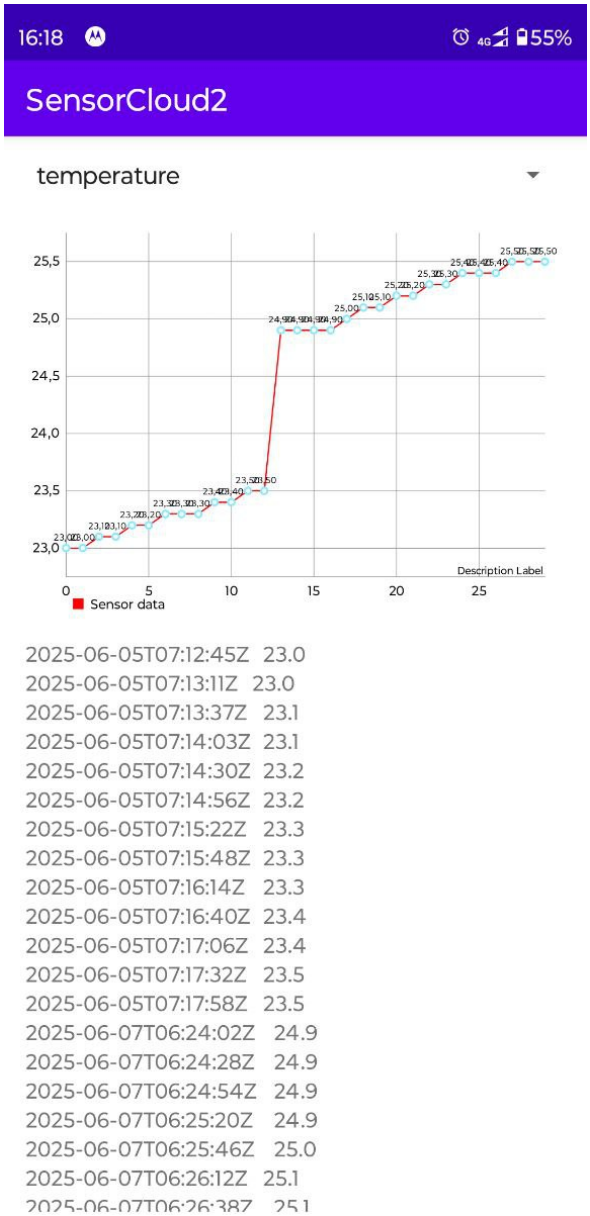


Рис. 4.4. Графік і таблиця

- Налаштування: параметри ThingSpeak (channel ID, API key) зберігаються та відновлюються після перезапуску програми. Налаштування перевірялися шляхом внесення неправильного ключа – програма реагувала повідомленням про помилку. Після виправлення даних – з’єднання успішне.

Ц поточні дані на головної сторінки до зміну коду як ми бачимо тут виводять температуру вологість ґрунту повітря

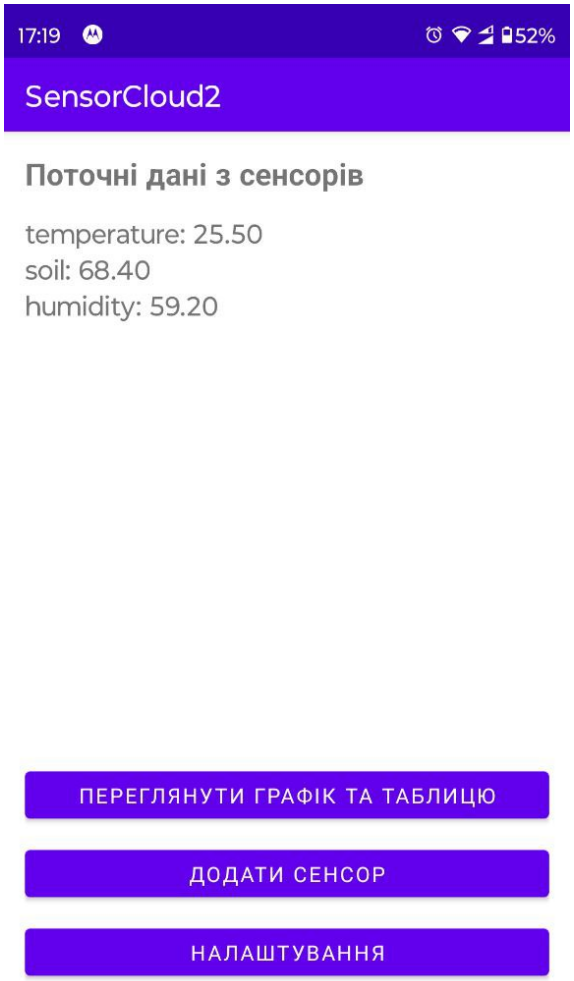


Рис. 4.5. Головна сторінка з даними

Тут показано налаштування ThingSpeak(в налаштування зразу відображаються дані які введено).

Теперішні дані:

channel ID: 2758644

API key: 788XZOVV8OVA1ALG

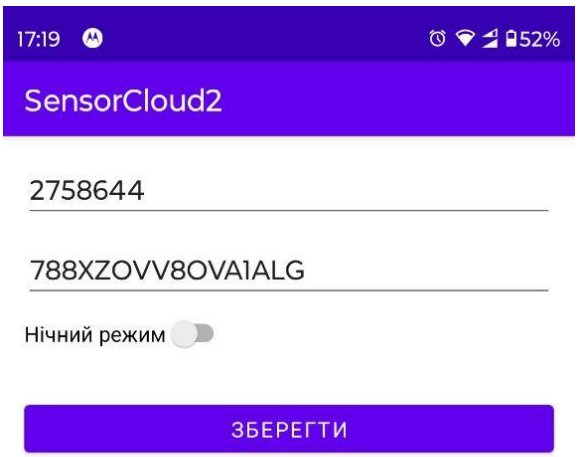


Рис. 4.6. Сторінка з налаштування

Змінимо дані на:

channel ID: 2758642

API key: WYSWY88T2U32OXS5

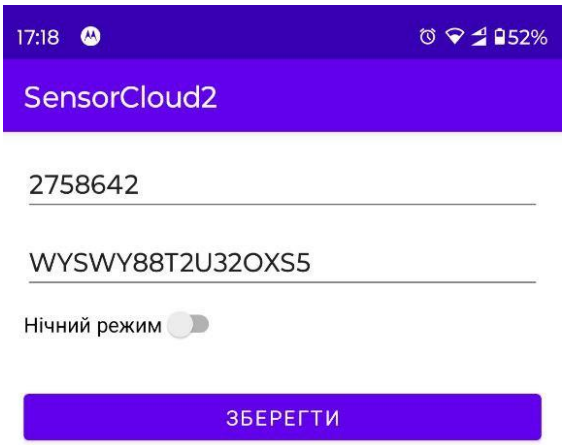


Рис. 4.7. Сторінка з видозмінним налаштування

Як бачимо дані на головні сторінки змінилися

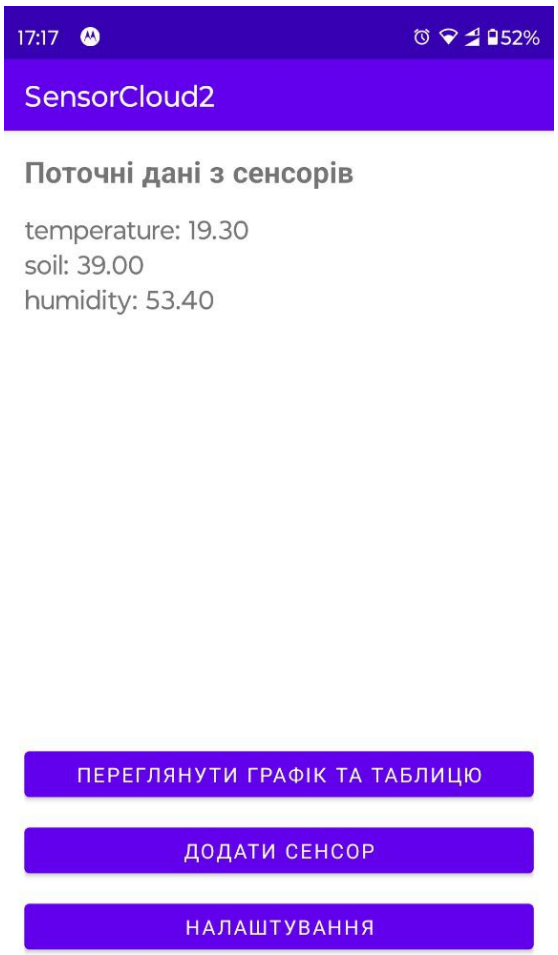


Рис. 4.8. Головна сторінка з іншими даними

- Тема оформлення: перемикач теми працює стабільно. Зміна теми застосовується без потреби в перезапуску програми. Усі активності оновлюють інтерфейс згідно з обраною темною або світлою темою.

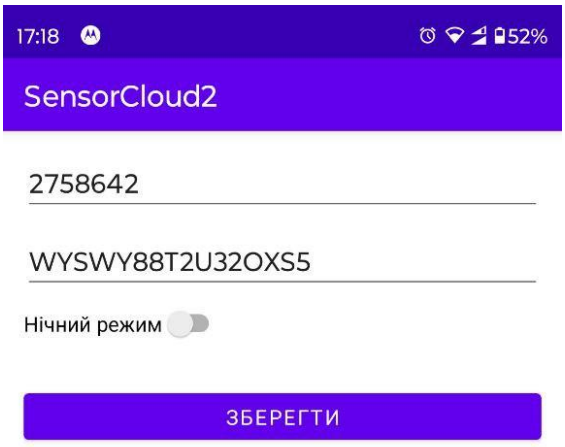


Рис. 4.9. Сторінка з білою темою

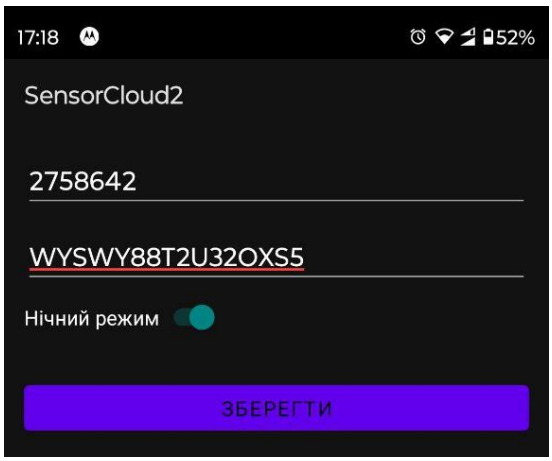


Рис. 4.10. Сторінка з чорною темою

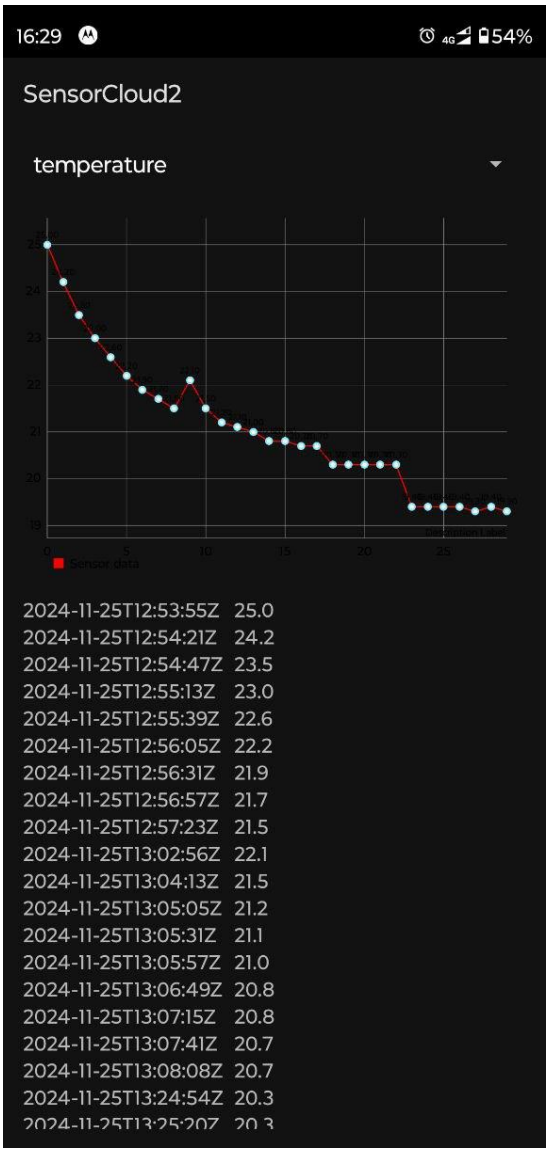


Рис. 4.11. Сторінка таблиці та графіка з чорною темою

- Додавання сенсорів: при додаванні нового сенсора перевіряється наявність дублікатів; поля заповнюються коректно; після збереження – відображаються в MainActivity та TableActivity. Видалення працює коректно: сенсор зникає зі списку, а також не відображається в інтерфейсі.

  Тут ми бачимо процес додавання сенсора SOIL(вологість ґрунту) на рисунку показано :

  - назва
  - номер field(цей номер треба брати на ThingSpeak для кожного датчика окремо)
  - кнопки додати та видалити
  - датчики які можна видалити
  - кнопку зберегти

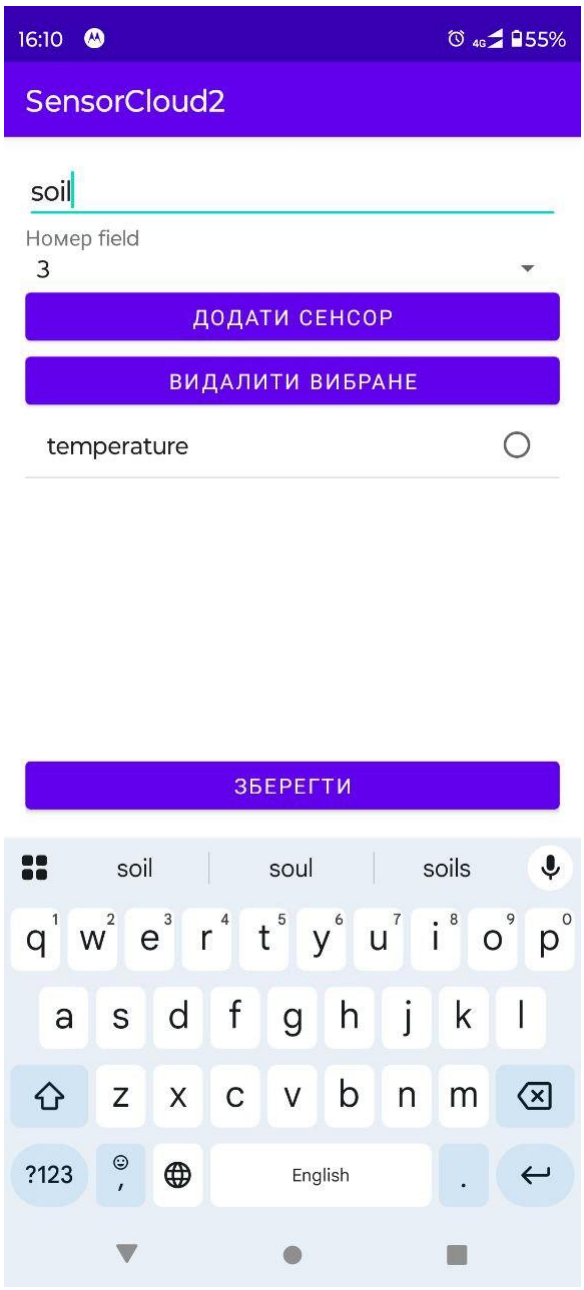


Рис. 4.12. Додавання сенсора

Після додавання сенсора SOIL відображено на головні сторінки

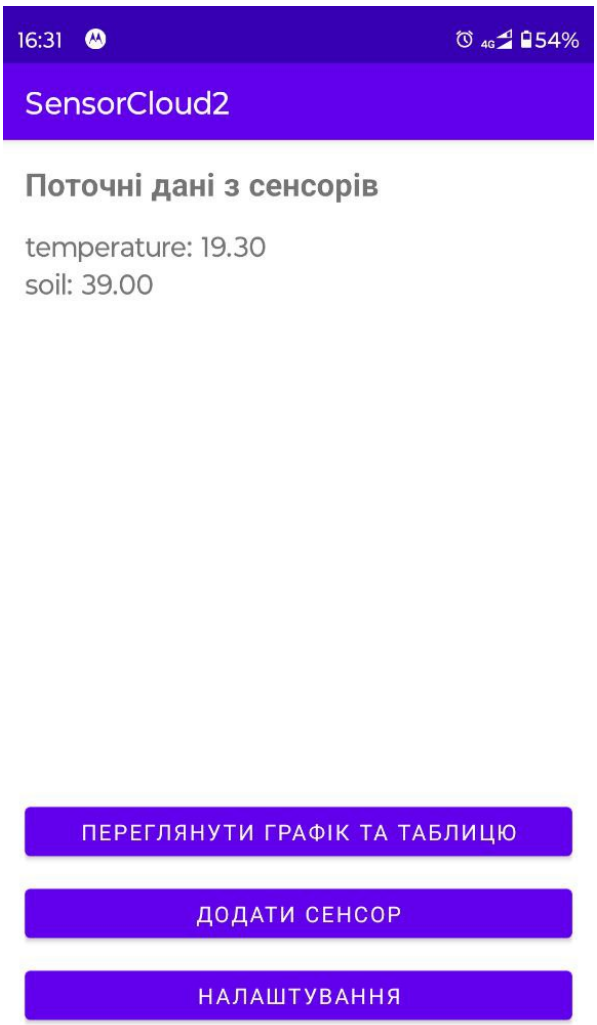


Рис. 4.13. Відображення доданого сенсора на головній сторінки

Також на сторінки з таблиці та графіки теж відображено дані з сенсора SOIL.

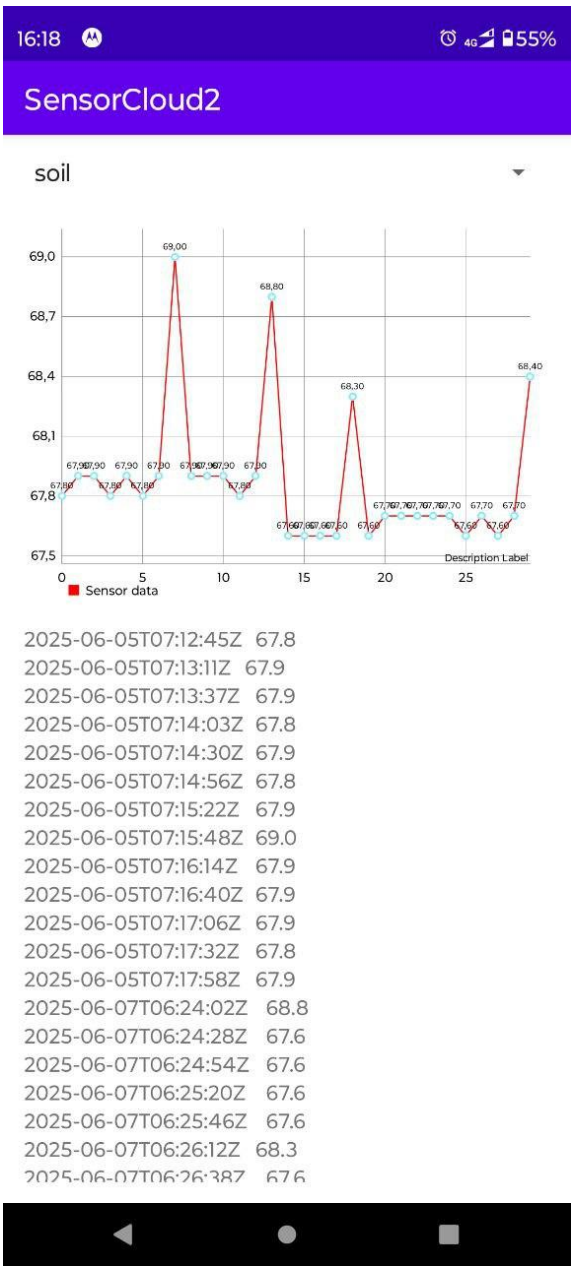


Рис. 4.14. Відображення доданого сенсора з графікою і таблицею

При додавання сенсора SOIL хоча вона вже присутня показують помилку(Toast) і сенсор не буде добавлений

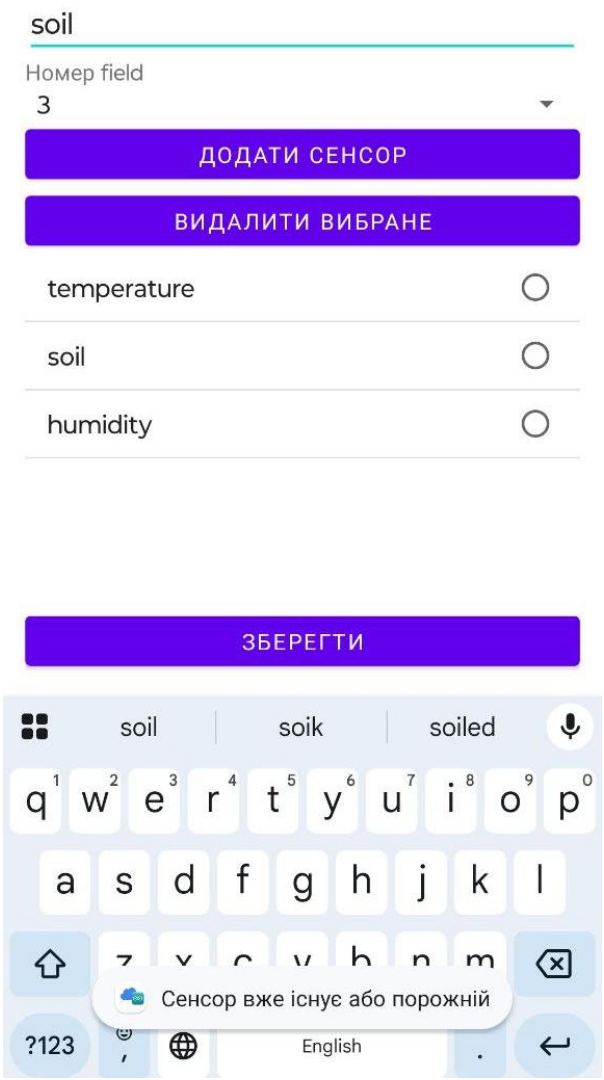


Рис. 4.15. З попередженням що сенсор вже є

Видалення існуючих записів для цього користувач повинен:

- вибрати зі списку сенсор (наприклад, "SOIL");
- після чого натиснути кнопку "Видалити вибране";
- Для завершення операції необхідно натиснути кнопку "Зберегти", яка оновлює локальний список і синхронізує зміни з іншими частинами додатку;

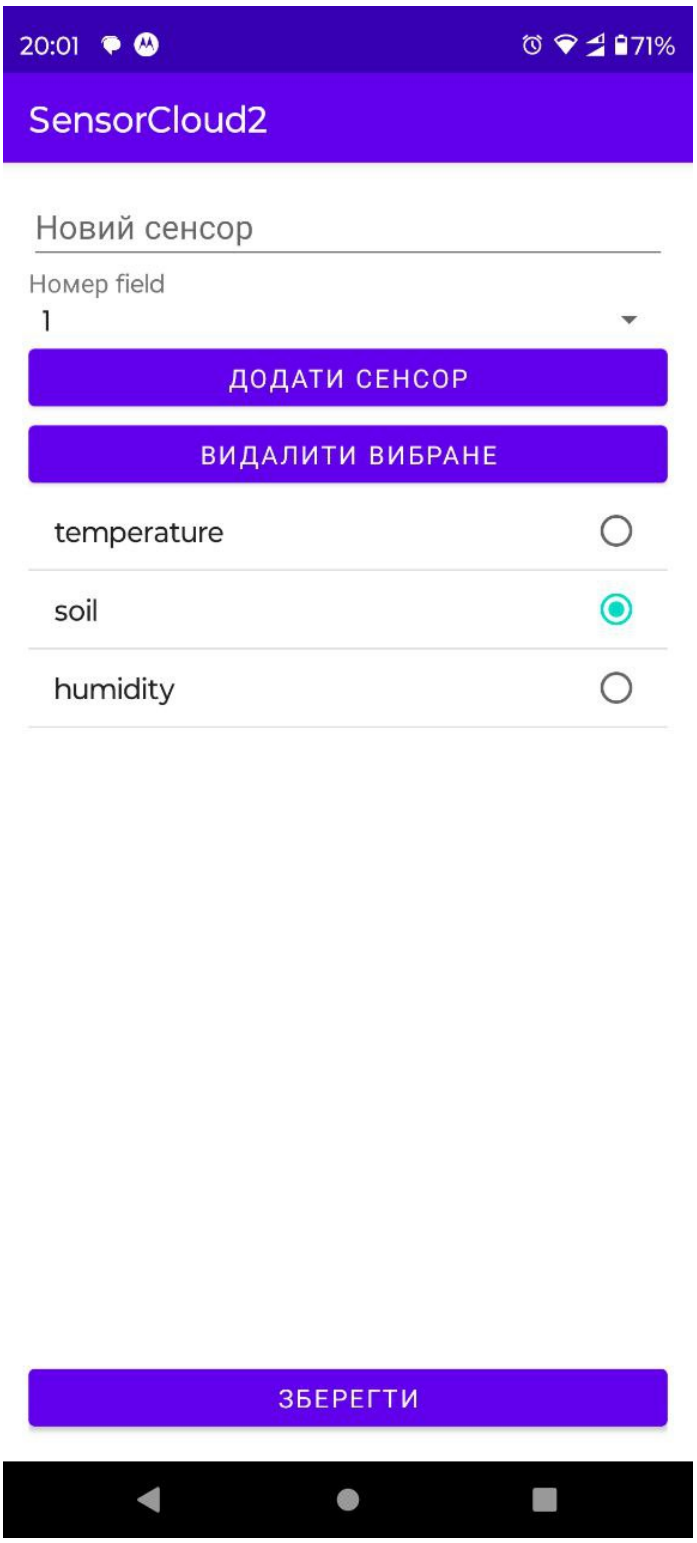


Рис. 4.16. Вибрано SOIL

Як бачимо на цій картинці сенсор SOIL відсутній.

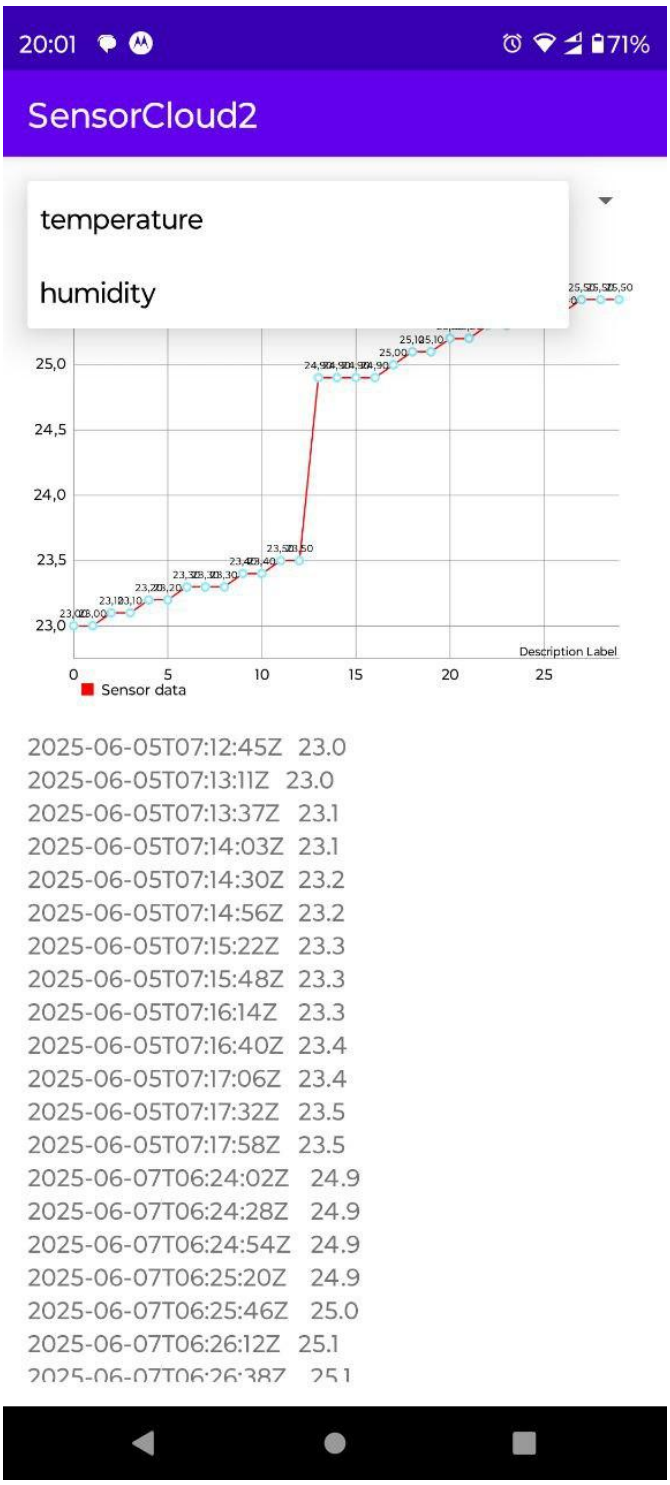


Рис. 4.17. Сторінка з графікою і таблицею на якому відсутній SOIL

Висновки тестування Усі функціональні модулі працюють згідно з очікуванням. Дані зчитуються з ThingSpeak стабільно, графіки та таблиці оновлюються коректна, зміни збережених параметрів набирають сили одразу. Обробка помилок реалізована адекватно. Результати тестування підтверджують, що застосунок готовий до практичного використання.

ВИСНОВКИ

У результаті виконаної кваліфікаційної роботи було реалізовано мобільний додаток на платформі Android, який дозволяє зчитувати, обробляти та візуалізувати дані з сенсорів, що підключені до мікроконтролера ESP8266. Дані передаються у хмарну службу ThingSpeak, звідки вони отримуються через API у форматі JSON.

У ході розробки були досягнуті наступні результати:

• реалізовано повноцінний клієнтський додаток із підтримкою роботи з віддаленим веб-сервісом;

• забезпечено зручний користувацький інтерфейс із підтримкою нічного режиму;

• додано можливість динамічного керування списком сенсорів і відображенням значень з відповідних полів ThingSpeak;

• реалізовано збереження параметрів додатку у бази даних за допомогою SharedPreferences;

• створено програму для відображення графіка за допомогою бібліотеки MPAndroidChart та таблиць у текстовому вигляді;

• проведено тестування, яке підтвердило стабільну роботу додатку, обробку помилок та готовність до практичного застосування.

Отримані результати показують хорошу взаємодію додатка з ThingSpeak. У подальшому додаток можна розширити шляхом firebase або покращити взаємодії з thingspeak.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бондаренко О. Г., Коваленко А. І. *Розробка програмного забезпечення в середовищі Android Studio: навчальний посібник.* — Київ: НАУ, 2020. URL: https://er.nau.edu.ua/handle/NAU/41471 (дата звернення: 13.07.2024)
2. YL-69 датчик вологості ґрунту з Arduino. Yasoob Khalid. URL: https://yasoob.me/posts/yl-69-soil-sensor-arduino/ (дата звернення: 23.08.2024).
3. DHT22 сенсор підключення до Arduino. URL: https://quartzcomponents.com/blogs/electronics-projects/diy-temperature-and-humidity-monitoring-system-with-using-arduino-and-dht22-sensor (дата звернення: 23.08.2024).
4. ESP8266 AT Commands using Arduino and Serial Monitor. YouTube – How To Mechatronics. URL: https://www.youtube.com/watch?v=igPqNlfLcs0 (дата звернення: 23.08.2024).
5. SharedPreferences. Android Developers. URL: https://developer.android.com/training/data-storage/shared-preferences#java (дата звернення: 03.05.2025).
6. Java + Android. Онлайн-курс. ITProger. URL: https://itproger.com/ua/course/java-android (дата звернення: 21.04.2025).
7. ThingSpeak Channel Settings. MathWorks Documentation. URL: https://www.mathworks.com/help/thingspeak/channel-settings.html (дата звернення: 24.04.2025).
8. Що таке Arduino?. Hardware libre. URL: https://www.hwlibre.com/uk/що-таке-ардуїно/ (дата звернення: 08.06.2024).
9. OkHttp – Square Open Source. URL: https://square.github.io/okhttp/ (дата звернення: 03.06.2025).

10. haredPreferences у Android Studio — приклад реалізації. Coding with Dev. YouTube. URL: https://www.youtube.com/watch?v=4WxKQTUweVg (дата звернення: 08.05.2025).
11. W3Schools. Java Tutorial. URL: https://www.w3schools.com/java/ (дата звернення: 12.06.2025).
12. Запити до ШІ GPT-чат (кітень-червень 2025 року):

Серед запитів є:

- Як викликати дані з ThingSpeak на android studio; (дата звернення: 25.04.2025).
- Як використовувати бібліотеки OkHttp + json; (дата звернення: 03.05.2025)
- Які зміни потрібні для коректного використання цих бібліотек в Android Studio; (дата звернення: 03.05.2025)