МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інформаційна система домашньої метеорологічної станції»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Кузіков Б. О.**

Студента групи ІН – 61 **Чугая Н. В.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-61 спеціальності "Інформатика" денної форми навчання Чугая Нікіти Віталійовича.

Тема: "Інформаційна система домашньої метеорологічної станції"

Затверджена наказом по СумДУ

№_____ від _____2020 р.

Зміст пояснювальної записки: 1) огляд технологічних трендів таких як IoT та розумні будинки 2) постановка завдання 3) опис основних методів та інструментів розробки апаратної частини, та розробки мобільних додатків 4) розробка апаратної й програмної частини інформаційної системи; 5) аналіз результатів розробки та роботи системи.

Дата видачі завдання "_____" _____2020 р.

Керівник випускної роботи _____ Кузіков Б. О.

Завдання приняв до виконання _____ Чугай Н. В.

РЕФЕРАТ

Записка: 66 стор., 29 рис., 1 табл., 4 додатки, 22 джерела.

Об’єкт дослідження — платформа розробки Arduino

Мета роботи — розробка домашньої метеорологічної станці з можливістю передачі даних на мобільний додаток та контролем якості повітря.

Методи дослідження — метод розробки пристрою на платформі Arduino, та мобільного-додатку для Android.

Результати — розроблена інформаційна система домашньої метеорологічної станці на основі платформи Arduino. Даний пристрій має необхідні вбудовані датчики для прогнозування погоди на невеликі проміжки часу, а також контролю мікроклімату в приміщенні.

Для можливості віддаленого контролю показників з метеостанції, був розроблений веб-сервер та мобільний додаток на Android. Метеостанція може бути підключена до мережі Інтернет і надсилати поточні дані на веб-сервер. А користувач в свою чергу через мобільний додаток може отримати ці дані з будь-якої точки світу.

ARDUINO, ДОМАШНЯ МЕТЕОСТАНЦІЯ, ANDROID ДОДАТОК, КЛІМАТКОНТРОЛЬ.

ЗМІСТ

ВСТУП

Безпека і комфорт - це те чим люди завжди намагалися забезпечити своє життя і головним символом цього є дім, будівля, приміщення. Але чотирьох стін і стелі не достатньо для забезпечення належного рівня комфорту і безпеки, тому з розвитком технологій люди оснащували свої приміщення необхідними предметами, механізмами, пристроями.

Дослідження проведені компаніями Ribble і Statista кажуть про те, що ми в середньому проводимо 92% свого часу в замкнутому просторі. Таким чином, оскільки більшість свого життя ми проводимо в приміщенні, чи то в офісі або вдома, можна зробити висновки про важливість належного рівня комфорту, а безпосередній вплив на наше відчуття комфорту має мікроклімат [1].

Завдяки стрімкому розвитку і поширенню інформаційних технологій, нещодавно фантастичний термін «розумний будинок» став реальним. Розумний дім - це термін, який відноситься до сучасних будинків, що мають систему приладів, якими власник може керувати дистанційно, часто через мобільний додаток. Розумні пристрої з підтримкою будинку можуть також працювати разом з іншими пристроями вдома та передавати інформацію іншим розумним пристроям. Системи безпеки, управління мікрокліматом та освітленням все це є складовими розумного будинку[2]. Не дивлячись на його реальність, системи розумного дому ще є недоступними для більшості людей. Тому на ринку наявні окремі, доступніші пристрої які можуть зробити вашу оселю більш «розумною», комфортною і безпечною. А розвиток і доступність мікропроцесорів, датчиків та інших комплектуючих дають можливість створити власні пристрої і навіть систем розумного будинку, з можливостями які будуть обмежуватися тільки вашою фантазією.

Таким чином, метою данної робочи є розробка домашньої метеорологічної станці з можливістю передачі даних на мобільний додаток та контролем якості

повітря, опис принципів її роботи, практична реалізація апаратної та програмної частин.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Домашня метеостанція

Домашня метеостанція – це електронний прилад, який виміряє атмосферні показники (атмосферний тиск, температуру, вологість, тощо), а також прогнозує погоду на основі цих даних.

Вони можуть варіюватися від дуже простих моніторів мікроклімату в приміщенні з вбудованими датчиками, до набагато складніших станцій з зовнішніми датчиками необхідними для більш точного прогнозування погоди, а прогнозування погоди на атмосферних даних виміряних локально може бути більш точним ніж на глобальних даних для певного району. Тому домашня метеостанція має перевагу перед службою прогнозу погоди, тим паче для прогнозу погоди на невеликі терміни вистачить показників атмосферного тиску[4].

1.2 Огляд функціоналу домашніх метеостанцій

На ринку запропоновано великий асортимент домашніх метеостанцій з різним діапазоном цін. Давайте порівняємо метеостанції трьох цінових категорій і розглянемо їх основні функції (табл.1).

Таблиця 1 – Загальна порівняльна таблиця домашніх метеостанцій

Дешеві	Середні	Дорогі
Функціонал: - будильник; - годинник; - календар; - вимірювання внутрішніх показників (температура, тиск, вологість).	Функціонал: - будильник; - годинник; - календар; - вимірювання внутрішніх показників - вимірювання зовнішніх показників - фаза місяця	Функціонал: - будильник; - годинник; - календар; - вимірювання внутрішніх\внутрішніх показників - фаза місяця - прогноз погоди на 12 годин - графіки зміни показників - тощо...
Переваги: + ціна	Переваги: + точність вимірів + інформативний дисплей	Переваги: + точність вимірів + кольоровий, інформативний дисплей
Недоліки: - точність вимірів; - якість матеріалів; - відсутність функції прогрозування погоди;		Недоліки: - ціна

1.3 Системи контролю мікроклімату

Як вже зазначалося в останні роки більш реальним стає термін «розумний будинок» і однією із головних систем розумного будинку є система контролю мікроклімату.

Вказана система використовує показники з різноманітних датчиків (температури, вологості, якості повітря, свіжості повітря, тощо) і на їх основі вона може, як просто інформувати про стан мікроклімату в приміщенні, так і автоматизовано управляти пристроями які саме впливають на мікроклімат [5][6]. До них відносяться:

- кондеціонери;
- системи вентиляції;
- термостати;
- обігрівачі;
- зволожувачв повітря;
- бризери (очищувач повітря);
- тощо.

Давайте розглянемо, що таке мікроклімат приміщень і взагалі навіщо його контролювати. Існує міжнародний стандарт ГОСТ 30494-2011, який встановлює вимоги до мікроклімату громадських та житлових будинків. Цей ГОСТ визначає мікроклімат приміщень як "стан внутрішнього середовища приміщення, що впливає на людину". Внутрішнє середовище в більшості випадків - це повітря всередині приміщення. Основними складовими мікроклімату середовища є:

- якість повітря;
- температура повітря;
- відносна вологість.

Насправді мікроклімат має прямий вплив на людину. Якщо він нормальний (оптимальний), то людина відчуває почуття комфорту, а організм не витрачає сил

на адаптацію до зовнішніх умов. Наприклад, хороший мікроклімат усуває тепло, при якому людському організму доведеться активувати механізми терморегуляції.

Давайте розглянемо, що розуміється під «якістю повітря» в приміщенні. Повітря в квартирі містить забруднення з різних джерел. По-перше, це частинки, які потрапляють в приміщення зовні - через відкриті вікна або вентиляційну систему без очищення. Це може бути і пил, і вихлопні гази, і викиди рослин. По-друге, деякі шкідливі речовини можуть випаровуватися з меблів, оздоблювальних матеріалів та предметів. Часто в повітрі квартир можна зустріти формальдегід. По-третє, це біологічне забруднення людини - так звані антропотоксини. Організм людини виділяє ацетон, аміак, феноли, аміни, вуглекислий газ CO2.

Звичайно, дані категорії забруднюючих речовин відрізняються за ступенем небезпеки. Наприклад, концентровані викиди сірководню з сусідньої рослини спричинять більше шкоди, ніж будь-який з антропотоксинів. У будь-якому випадку хороший мікроклімат в квартирі передбачає мінімальний вміст забруднюючих речовин у повітрі.

Глибокий аналіз складу та чистоти повітря в квартирі неможливий без спеціального обладнання. Такий аналіз може проводити хімічна лабораторія. Непрямим показником чистоти повітря є концентрація CO2. Чим вона вища, тим гірша вентиляція. І чим гірша вентиляція, тим більше забруднюючих речовин накопичується у повітрі квартири.

Повітря можна очистити за допомогою витяжної вентиляції з фільтром, наприклад, компактним лезом. Його фільтри зберігаються у вигляді пилових частинок, пилку, мікроорганізмів, газів та запахів. Жирніше також може працювати очисник повітря - фільтрувати забруднення, джерела яких знаходяться не зовні, а всередині квартири. Крім того, ви можете

використовувати бризер разом із очищувачем повітря, який не лише зберігає інфекції та віруси, але й знищує їх, тим самим зменшуючи ризик захворіти[3].

1.4 Інтернет речей

Інтернет речей, або Internet of Things (далі IoT), являє собою систему взаємопов'язаних обчислювальних пристроїв, механічних і цифрових машин, об'єктів, тварин або людей, які забезпечені унікальними ідентифікаторами і здатністю передавати дані по мережі.

Як працює IoT? Екосистема IoT складається з розумних пристроїв з підтримкою Інтернету, які використовують вбудовані системи, такі як процесори, датчики та комунікаційне обладнання, щоб збирати, надсилати та діяти на даних, які вони отримують із свого середовища. Пристрої IoT діляться даними, які вони збирають, підключаючись до шлюзу IoT або іншого крайового пристрою, де дані надсилаються в хмару для аналізу або локального аналізу. Іноді ці пристрої спілкуються з іншими пов'язаними пристроями та діють на основі інформації, яку вони отримують один від одного. Пристрої виконують більшу частину роботи без втручання людини, хоча люди можуть взаємодіяти з пристроями - наприклад, налаштувати їх, дати їм інструкції або отримати доступ до даних.

Інтернет речей допомагає людям жити та працювати продуктивніше, а також отримати повний контроль над своїм життям[8]. Приклади використання IoT:

- розумний будинок;
- автомобілі;
- розумні міста;
- сільське господарство;
- тощо.

1.5 Розвиток і поширення мікроелектроніки

Як вже було зазначено вище, останній час набула популярності розробка і створення пристроїв на основі мікроконтролерів, мікрокомп’ютерів і сумісних з ними комплектуючими. Особливо це стосується готових платформ, які складаються з апаратної частини (плат, мікрокомп’ютерів) та середовища розробки. Простота, сумісність з більшістю операційних систем, низька вартістю як плат так і сумісних з ними комплектуючих - все це робить розробку на їх основі цікавою і доступною для кожного.

1.6 Постановка задачі

Розробити домашню метеостанцію на основі плати та датчиків, які буду отримувати інформацію про стан навколишнього середовища, а саме:

- температуру повітря;
- вологість;
- атмосферний тиск;
- рівень вуглекислого газу (CO2);
- рівень природного газу (метану).

На основі вказаних вимірів, даний пристрій повинен:

- відображення метеорологічних показників;
- прогнозування погоди;
- відображення показників необхідних для контролю мікроклімату в приміщенні;
- світлова індикація рівня вуглекислого газу;
- звукова попередженя про наявність природнього газу (метану) в повітрі;
- відображання графіків зміни кожної з величин за годину та за добу.
- передача даних на мобільний пристрій через інтернет;

Також має бути розроблений мобільний додаток який зможе отримувати поточні данні з метеостанції.

2 ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

2.1 Вибір засобів апаратної реалізації

Для створення апаратної частини домашньої метеостанції була вибрана електронна платформа з відкритим кодом Arduino, яка заснована на простому у використанні апаратному та програмному забезпеченні. Платформа Arduino стала досить популярною серед людей, що тільки починають з електроніки, і це є не дарма. Більшість плат потебують окремого під’єднання програматора для завантаження коду, плата Arduino має вбудований програматор, тобно ви можете просто використовувати USB-кабель. Також Arduino забезпечує стандартний форм-фактор, який розбиває функції мікроконтролера на більш доступний пакет[9]. Його активно використовую викладачі та студенти по всьому світу для реалізації проектів.

Завдяки своїй доступності і зрозумілості Arduino є ключовим інструментом для вивчення нового. Існує багато інших мікроконтролерів та платформ мікроконтролерів, доступних для фізичних обчислень. Parallax Basic Stamp, BX-24 Netmedia, Phidgets, MIT's Handyboard та багато інших пропонують подібну функціональність. Усі ці інструменти містять деталі програмування мікроконтролерів і загортають їх у простий у користуванні пакет[10][11]. Arduino також спрощує процес роботи з мікроконтролерами, але він має ряд переваг для користувачів перед іншими платформами:

- Доступність - плати Arduino відносно недорогі порівняно з іншими платформами мікроконтролерів. А через те що схеми плати Arduino з відкритим кодом, розширюваним обладнанням та публікуються за ліцензією Creative Commons, можна надрукувати свою версію плати з меншою собівартістю, що і зробили китайські виробники.

- Крос-платформленість - програмне забезпечення Arduino (IDE) працює на операційних системах Windows, Macintosh OSX та Linux. Більшість систем мікроконтролерів обмежені Windows.
- Просте, чітке середовище програмування - програмне забезпечення Arduino (IDE) (детальніше в пункті 2.3).
- Програмне забезпечення з відкритим кодом та розширюваним програмним забезпеченням - програмне забезпечення Arduino публікується як інструменти з відкритим кодом, доступні для розширення досвідченими програмістами. Мова може бути розширена за допомогою бібліотек C ++, і люди, які хочуть зрозуміти технічні деталі, можуть перейти з Arduino на мову програмування AVR C, на якій він заснований. Так само ви можете додати код AVR-C безпосередньо у свої програми Arduino.
- Великий вибір плат – завдяки різноманіттю плат Arduino можна підібрати саме ту, яка більше всього буде підходити для реалізації вашого проекту.

2.2 Вибір засобів програмної реалізації

Мова програмування пристроїв Arduino заснована на C / C ++, а компілюється і збирається з використання бібліотеки AVR Libc і дозволяє використовувати будь-які її функції. Разом з тим вона проста в освоєнні, і на даний момент мова програмування Arduino - це, мабуть, найзручніший спосіб програмування пристроїв на мікроконтролерах[11].

Для реалізації серверної частини та мобільного додатку на Android була вибрана мова програмування Java.

Java - потужна об’єктно-орієнтована мова програмування загального призначення. Вона використовується для розробки настільних і мобільних додатків, великої обробки даних, вбудованих систем тощо[12]. Її синтаксис здебільшого був запозичиний з мов програмування C і C++. В мові

програмування Java частина роботи яка зазвичай виконується програмістами, доручена віртуальній машині Java Virtual Machine. Вона має велику кількість вбудованих інструментів, фреймворків та сторонніх бібліотек, що робить розробку більш комфортною і продуктивною, а ще велику спільноту, яка завжди готова відповісти на питання і допомогти[13][14].

2.3 Вибір середовища розробки

Для програмування на Arduino використовується програмне забезпечення Arduino з відкритим кодом (IDE). Воно працює для Windows, Mac OS та Linux. Arduino IDE дуже просте і зручне у використанні, з підтримкою всіх наявних різновидів плат Arduino з різними мікропроцесорами [16]. Порівняно з більш серйозними середовищами розробки, в ній відсутнє дерева файлової структри проекту (складні проекти в ній не пришуться, тому це і не потрівно), відсутністю рефакторигу, автоматичного доповлення коду, а також швидкість компіляції коду не є її плюсом. Також в ній наявні базові бібліотеки для роботи з різними модулями, датчиками. Дуже зручним під час користування виявилася наявність готових програм-прикладів, для демонстрації роботи з певною бібліотекою [15].

Для розробки мобільного додатку була вибрана Android IDE, яка є офіційним середовищем для розробки додатків для Android. Вона була заснована на основі InteliJ IDEA, і крім потужного редактора коду та інструментів для розробниців, Android Studio пропонує ще більше функцій, що підвищують продуктивність при створенні програм для Android, таких як [17]:

- гнучка система побудови на основі Gradle;
- швидкий та багатофункціональний емулятор;
- єдине середовище, де можна розвиватись для всіх пристроїв Android;
- інструменти для вирішення проблем продуктивності, зручності використання, сумісності версій, тощо;

- підтримка C++ та NDK
- вбудована підтримка Google Cloud Platform , що спрощує інтеграцію Google Cloud Messaging та App Engine

Серверна частина була написана в InteliJ IDEA.

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Проектування інформаційної системи

Перед розробкою будь-якої інформаційної системи необхідно її спроектувати. Проектування (на відміну від моделювання) передбачає роботу з поки неіснуючим об'єктом і направлено на створення інформаційної системи в області: обробки об'єктів майбутньої бази даних, написання програм (в тому числі - звітних і екранних форм), що забезпечують виконання запитів до даних, виконання обліку функціонування конкретної середовища (технології).

Діаграма варіантів використання та її реалізація

Діаграма варіантів використання в UML відображає взаємодію користувача з системою, та показує зв'язок між користувачем і різними випадками використання , в яких бере участь користувач. Вона може ідентифікувати різні типи користувачів системи (актори) та різні випадки використання, а також часто супроводжуватися іншими типами діаграм[21]. Діаграми випадків використання UML ідеально підходять для [22]:

- представлення цілей взаємодії між системою і користувачем;
- визначення та організація функціональних вимог у системі;
- вказання контексту та вимог системи;
- моделювання базового потоку подій у випадку використання;

Розглянемо модель використання даної інформаційної системи з точки зору користувача домашньої метеостанції. І відповідно до постановки задачі (підрозділ 1.7) відобразимо усі вимоги до функціоналу на рис 1.1.

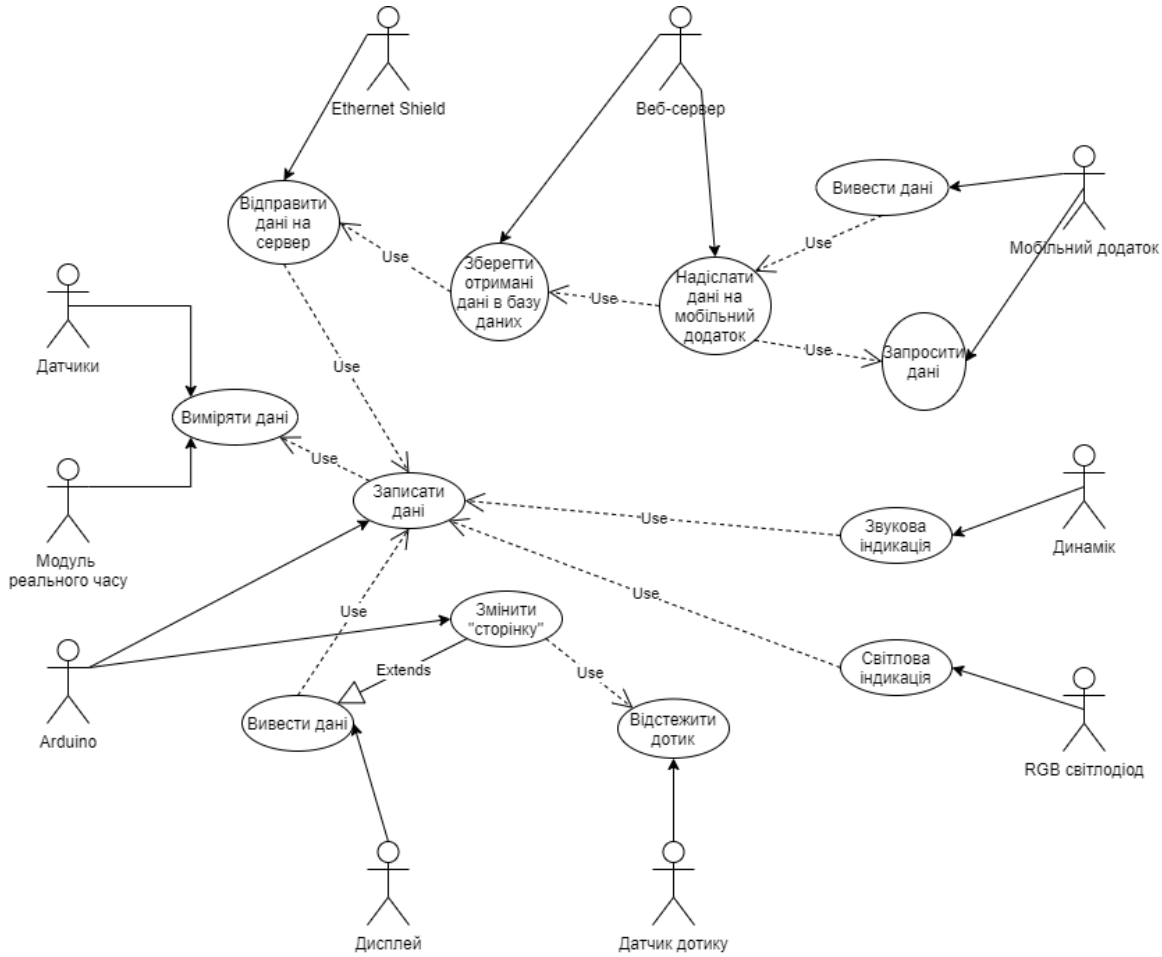

Рисунок 3.1 - Діаграма варіантів використання

Діаграма сутність – зв’язок (Entity-relationship diagram)

Діаграма «сутність – зв’язок» (далі ERD) – взаємозв'язок між сутностями, що зберігаються в базі даних. Суб'єкт у цьому контексті є об'єктом, складовою даних. Сутність являє собою представлення набору реальних або абстрактних об'єктів (людей, речей, місць, подій, ідей, комбінацій і т. д.), які можна виділити в одну групу, тому що вони мають однакові характеристики і можуть брати участь в схожих зв'язках.

Визначаючи сутності, їх атрибути та показуючи зв’язки між ними, ER-діаграма ілюструє логічну структуру баз даних.

ER-діаграми використовуються для побудови дизайну бази даних[23].

Оскільки в нашій системі використовується база даних з однією таблицею, то побудуємо діаграму з однією сутністю і переліком її атрибутів.

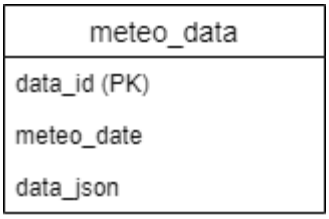

Рисунок 3.2 - ER-діаграма

Діаграма класів

В інженерії програмного забезпечення діаграма класів в уніфікованій мові моделювання (UML) є типом діаграми статичної структури, що описує структуру системи, показуючи класи системи, їх атрибути, методи та зв’язки між об'єктами.

Класи на діаграмі позначаються прямокутником з назвою класу вгорі, полями класу посередині та методами внизу.

Між собою класи поєднують різні типи зв’язків, а саме:

- асоціація (однонаправлений деякий з’вязок між класами);
- агрегація (один із об’єктів класу включає в себе деякі об’єкти іншого класу);
- композиція (більш строгий варіант агрегації, має жорстку залежність часу існування екземплярів класу контейнера та екземплятів класів що містяться в ньому).

Зобразимо діаграму класів мобільного-додатку для Android.

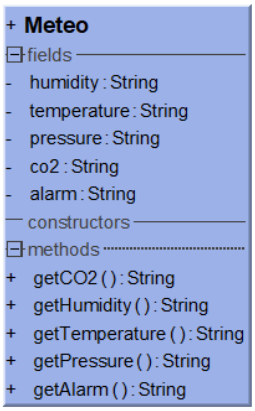

Рисунок 3.3 - Діаграма класів моделі

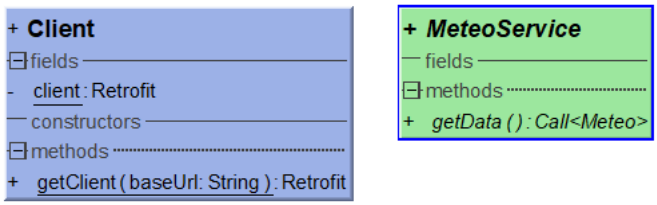

Рисунок 3.4 - Діаграма класів підключення до серверу

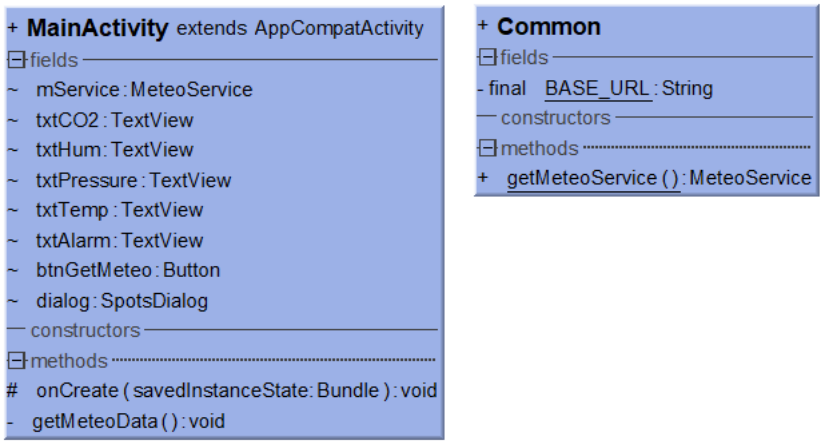

Рисунок 3.4 - Діаграма класів сервісів

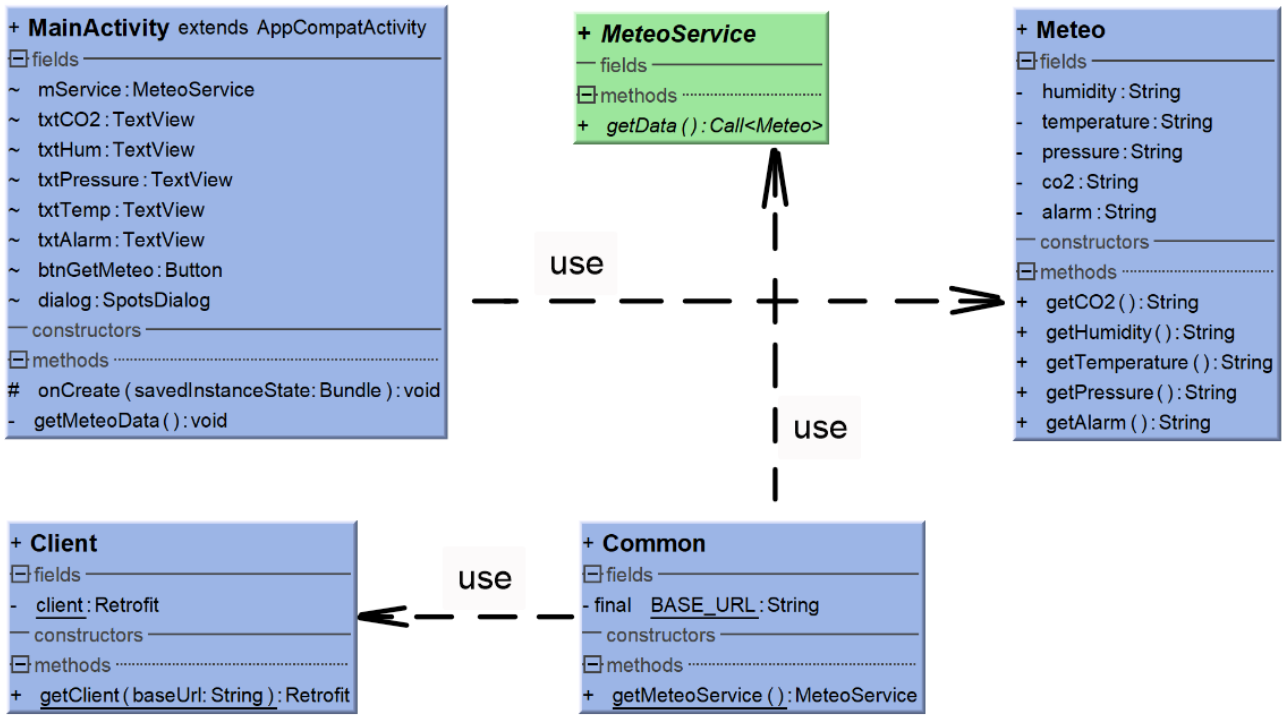

Рисунок 3.5 - Діаграма класів додатку із зв’язками

3.2 Опис та використання вхідних даних

Вхідними даними для нашої системи є показники (температура, вологість, атмосферний тиск, природний газ) отриманні з датчиків.

Поточні значення виводятся на дисплей та на їх основі будуються графіки зміни кожного з показників за годину та день. Також домашня метеостанція аналізує зміну атмосферного тиску і робить короткочасний прогноз погоди (ймовірність опадів). За наявності підключення до мережі Інтернет, пристрій раз у декілька хвилин надсилає поточні значення на сервер, де воні зберігаються в базі даних у json форматі.

3.3 Апаратна реалізація інформаційної системи

За основу була вибрана плата Arduino Uno. Вона базується на мікроконтролері ATmega328P. Arduino Uno має [19][20]:

- 32 Кб флеш-пам’яті;

- 2 Кб ОЗУ;
- EEPROM розміром 1 Кб (пам’ять в якій дані зберігаються після вимкнення пристрою);
- 14 цифрових входів\виходів;
- 6 аналогових входів;
- кварцевий генератор 16 МГц;
- порт USB;
- роз’єм живлення;
- ICSP (програматор);
- кнопку перезавантаження.

Саме з цією платою зручно використовувати мережевий модуль Ethernet Shield W5100, оскільки він має такий самий форм-фактор і розміщення виходів, тому просто встановлюється поверх Arduino Uno.

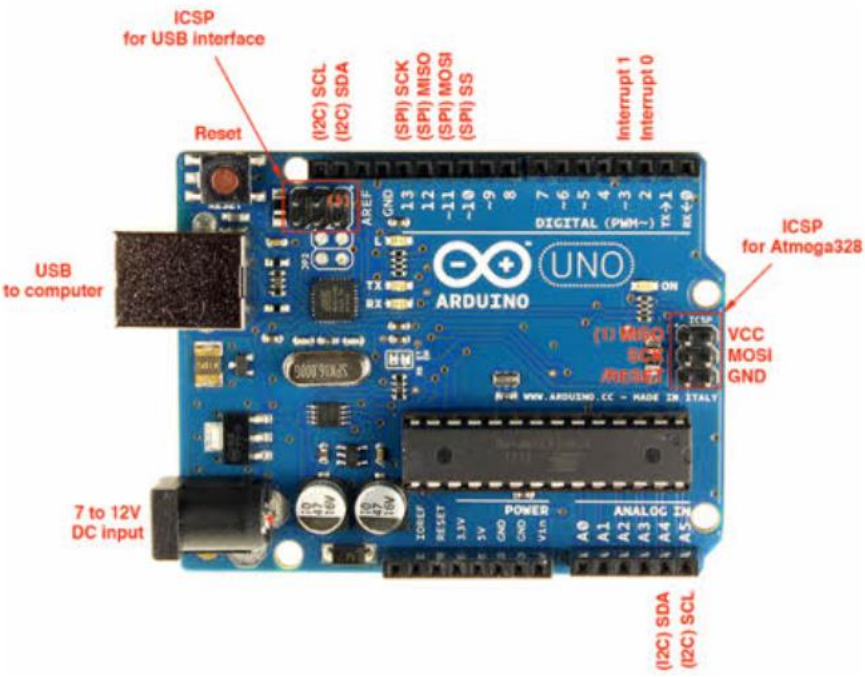

Рисунок 3.6 - Зовнішній вигляд та розміщення складових плати Arduino Uno

Давайте розглянемо модулі, датчики які були використані в даній роботі, а також їх з’єднання між собою.

1. LCD дисплей 20x4 з шиною I2C – призначений для виводу годинника, поточний даних з датчиків, графіків. Підключається через шину I2C тому має всього 4 виходи: GND (земля), VCC (живлення), SDA (лінія даних), SCL (лінія тактування). Живлення від 5V [18].

Рисунок 3.7 - LCD дисплей 20x4 з шиною I2C

2. BME280 – датчик для вимірювання вологості, температури та атмосферного тиску. Підключається через шину I2C, має 4 виходи: GND (земля), VCC (живлення), SDA (лінія даних), SCL (лінія тактування). Живлення від 5V.

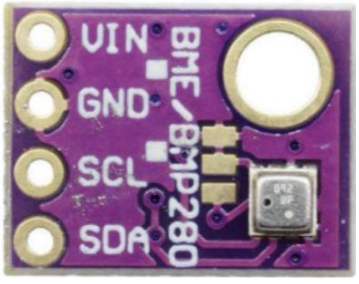

Рисунок 3.8 - Датчик вимірювання вологості, температури та тиску BME280

3. Real Time Clock модуль на DS3231SN – модуль реального часу з батарейкою (продовжує працювати після відключення пристрою від електромережі). Підключається через шину I2C, має 4 виходи: GND (земля), VCC (живлення), SDA (лінія даних), SCL (лінія тактування). Живлення від 5V.

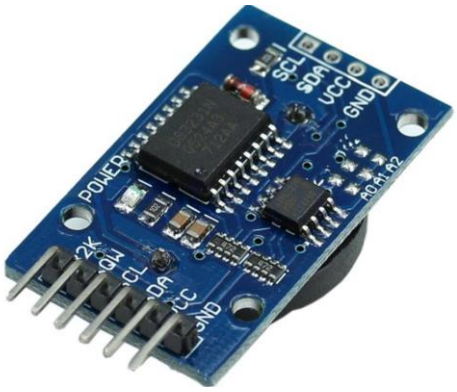

Рисунок 3.9 - Модуль реального часу DS3231SN з батарейкою

4. Модуль RGB світлодіода KY-016 – призначається для світлової індикації рівня вуглекислого газу (CO2) в повітрі. Має 4 виходи: GND (земля), R (червоний), G (зелений), B (синій).

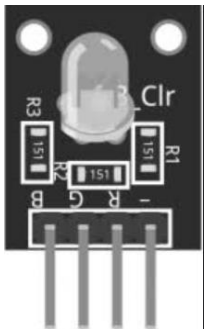

Рисунок 3.10 - RGB світлодіод

5. Плата перехідник microUSB-B – використовується для підключення зарядного пристрою, для живлення від електричної мережі. Підключається через 2 виходи: GND (земля) та VBUS (живлення).

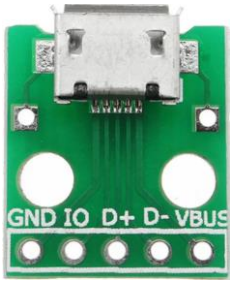

Рисунок 3.11 - Модуль USB-B для підключення живлення метеостанції

6. Модуль датчика якості повітря MQ-135 – призначений для вимірювання рівеня вуглекислого газу (CO2), NH3, NOx, диму в повітрі. Підключається через 3 виходи: GND (земля), VCC (живлення 5V), A0.

Рисунок 3.12 - Модуль датчика якості повітря MQ-135

7. Модуль датчика газу MQ-4 – призначений для вимірювання рівню природного газу (метану) в повітрі. Підключається через 3 виходи: GND (земля), VCC (живлення 5V), A0.

Рисунок 3.12 - Модуль датчика газу MQ-4

8. Модуль з пасивним динаміком KY-006 – призначений для звукової індикації рівня природного газу (метану) в повітрі. Підключається через 3 виходи: GND (земля), VCC (живлення 5V), цифровий.

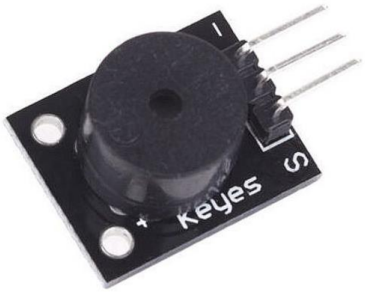

Рисунок 3.13 - пасивний динамік KY-006

9. Модуль датчика дотику TTP223B – використовується для переключення «сторінок» на дисплеї. Підключається через 3 виходи: GND (земля), VCC (живлення 5V), SIG (цифровий).

Рисунок 3.14 - Модуль датчика дотику

10. W5100 Ethernet Shield – «шилд» за допомогою якого можно під’єднати плату Arduino до локальної мережі або мережі Інтернет. Завдяки цьому модулю Arduino може виступати як веб-сервером так і веб-клієнтом. Деякі характеристики:
    - живлення від 5V, безпосередньо від плати Arduino;
    - Ethernet контролер W5100 з буфером 16 Кб;
    - швидкість підключення 10/100 Мб\с;
    - має слот micro-SD;

- одночасно підтримує до 4 з’єднань:
- роз’єм для підключення до мережі RJ-45.

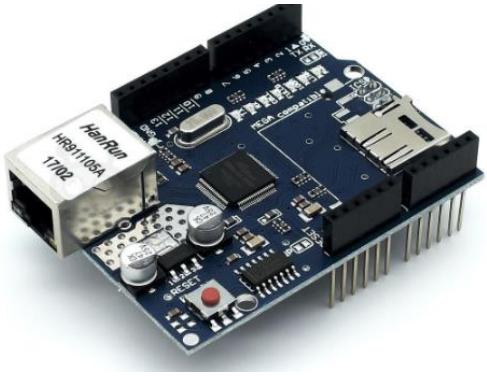

Рисунок 3.15 - W5100 Ethernet Shield

Нижче наведена схема підключення модулів до Ethernet Shield W5100, бо як було зазначено Ethernet Shield W5100 вставляється зверху Arduino Uno і займає всі роз’єми.

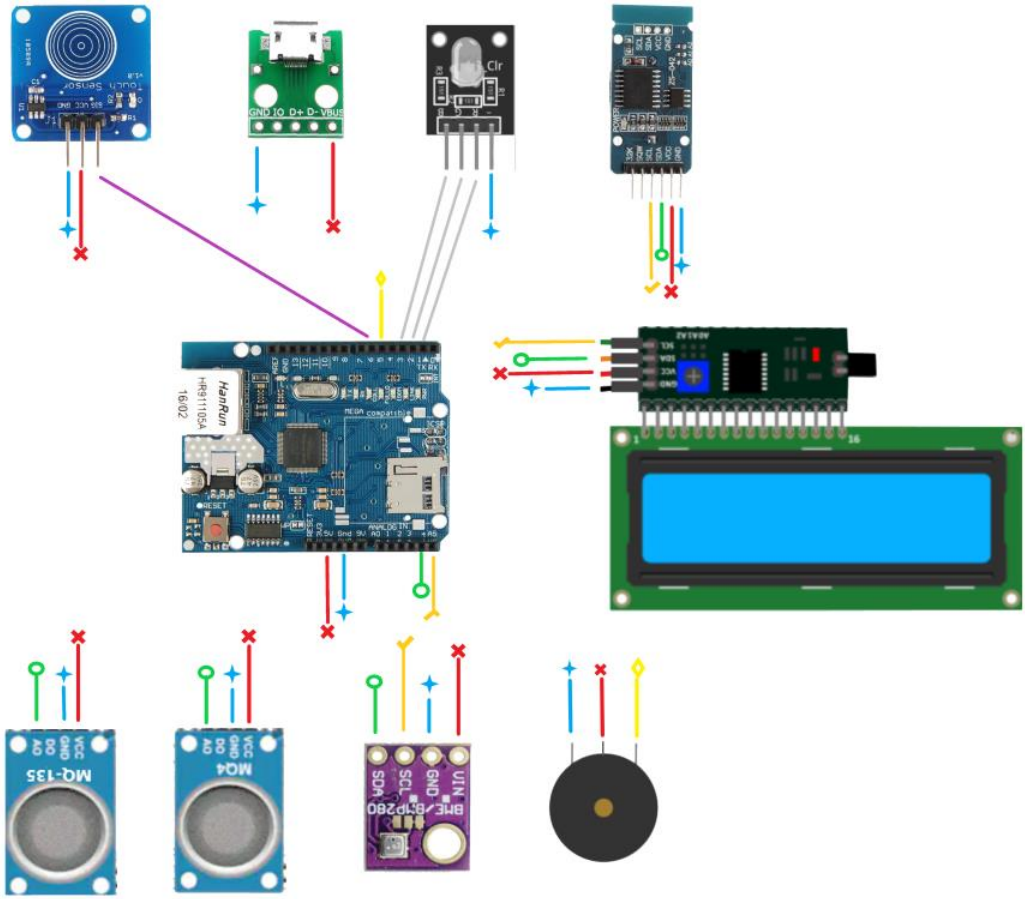

Рисунок 3.16 - Схема підключення модулів до плати

3.4 Програмна реалізація інформаційної системи

Для реалізації програмної частини домашньої метеостанції використовуються функції і методи з бібліотек, необхідних для роботи з датчиками і модулями.

Давайте коротко розглянемо дві основні функції, без яких не можливий запуск програми..

Функція setup() – код записаний в цьому блоці виконується лише один раз при запуску програми, в цій функції відбувається калібрування датчиків, ініціалізація та налаштування роботи модулів, а також початкове завантаження даних на екран.

```
void setup () {
  Serial.begin(9600);

  if (Ethernet.begin(mac) == 0) {
  Serial.println("Failed to configure Ethernet using DHCP");
  Ethernet.begin(mac, ip);
  }

  pinMode(LED_R, OUTPUT);
  pinMode(LED_G, OUTPUT);
  pinMode(LED_B, OUTPUT);
  analogWrite(LED_B, 0);

  Wire.begin();

  lcd.init(); // initialize the lcd
  lcd.backlight();
  lcd.home();
  dht.begin();
  lcd.clear(); // clear display before new values will be settled

  // following line sets the RTC to the date & time this sketch was compiled
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
```

```
        now = rtc.now();
        secs = now.second();
        mins = now.minute();
        hrs = now.hour();

        calibrate_modules();
        readSensors();
        digitalClockDisplay(hrs, mins);
        dispDate();
        dispSensors();
      }
```

calibrate_modules() - калібрування датчиків;

readSensors() - зчитування показників з датчиків;

digitalClockDisplay() - відображення годинника на екрані;

dispDate() - відображення дати;

dispSensors() - відображення поточних показників;

Функція loop() – все, що записано в цій функції виконується безперервно протягом всієї роботи мікроконтролера, одразу після закінчення виконання функції setup().

```
void loop () {
        readSensors();
        if (clockTimer.isReady()) clockTick(); // count the time twice a second
        // several timers for recalculating charts (per hour, per day and forecast)
        plotSensorsTick();
        modesTick();                    // catch button click and change modes
        if (sendTimer.isReady()) sendingToServer();
        if (mode == 0) {                                    // main page mode
          if (drawSensorsTimer.isReady()) dispSensors();  // refresh sensors value
with period SENS_TIME
        } else {                                            // chart mode
          if (plotTimer.isReady()) redrawPlot();           // refresh chart
        }
}
```

Дані на сервер надсилаються у функції sendingToServer() за допомогою POST http запиту раз у дві хвилини.

```
void sendingToServer()
 {
   if (client.connect(server, 50301)) {
    client.print(F("POST /postJsonData?humidity="));     //YOUR URL
    client.print(static_cast<String>(dispHum));
    client.print(F("&temperature="));
    client.print(static_cast<String>(dispTemp));
    client.print(F("&pressure="));
    client.print(static_cast<String>(dispPres));
    client.print(F("&co2="));
    client.print(static_cast<String>(dispCO2));
    client.print(F("&methane="));
    client.print(static_cast<String>(methane));
    client.print(F("&alarm="));
    client.print(static_cast<String>(alarm));
    client.print(" ");      //SPACE BEFORE HTTP/1.1
    client.print(F("HTTP/1.1"));
    client.println();
    client.println(F("Host: <Your Local IP>"));
    client.println("Connection: close");
    client.println();
  } else {
    // if you didn't get a connection to the server:
    Serial.println(F("connection failed"));
  }
}
```

Веб-сервер в свою чергу зберігає їх до бази даних яка має наступний вигляд.

```
create table meteo_data (
data_id serial NOT NULL PRIMARY KEY,
meteo_date timestamp,
data_json VARCHAR(1000)
);
```

І як видно всі показники зберігаються в один атрибут у json форматі, для того щоб мобільному додатку легше було зчитати ці данні. Оскільки дані на сервер надсилються дуже часто, була створена функція і тригер для запуску функції, яка видаляє записи старше одного дня.

Зі сторони Android додатку нижче наведений блок коду в якому відбувається зчитування та виведення даних.

```
private void getMeteoData() {
    mService.getData().enqueue(new Callback<Meteo>() {
        @Override
        public void onResponse(Call<Meteo> call, Response<Meteo> response) {
            dialog.dismiss();
            txtCO2.setText("CO2: " + response.body().getCO2() + " ppm");
            txtHum.setText("Humidity: " + response.body().getHumidity() + " %");
            txtTemp.setText("Temperature: " + response.body().getTemperature() +
" °C");
            txtPressure.setText("Pressure: " + response.body().getPressure() + "
mm");
            if(response.body().getAlarm().equals("0"))
                txtAlarm.setText("Warnings: Ok");
            else if(response.body().getAlarm().equals("1"))
                txtAlarm.setText("Warnings: Methane");
            else if(response.body().getAlarm().equals("2"))
                txtAlarm.setText("Warnings: Temperature");
            else if(response.body().getAlarm().equals("3"))
                txtAlarm.setText("Warnings: CO2");
            else if(response.body().getAlarm().equals("4"))
                txtAlarm.setText("Warnings: Smoke");
            else if(response.body().getAlarm().equals("5"))
                txtAlarm.setText("Warnings: Temperature and CO2");
            else if(response.body().getAlarm().equals("6"))
                txtAlarm.setText("Warnings: Temperature and Smoke");
            else if(response.body().getAlarm().equals("7"))
                txtAlarm.setText("Warnings: CO2 and Smoke");
            else
                txtAlarm.setText("Warnings: Temperature, Smoke and CO2");
        }

        @Override
        public void onFailure(Call<Meteo> call, Throwable t) {
            Log.e("ERROR",t.getMessage());
            dialog.dismiss();
        }
    });

    }
}
```

3.5 Аналіз роботи програми

На рис. 1.1 зображений головний екран домашньої метеостанції. На ньому відображається годинник, дата, температура, вологість, рівень вуглекислого газу, атмосферний тиск, вірогідність опадів та вівень природньоо газу (метану).

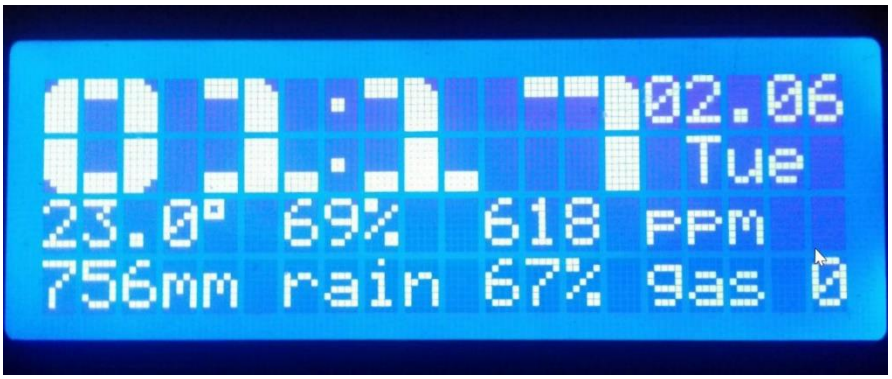

Рисунок 3.17 - Режим відобрження "головної сторінки"

Для зміни режимів відображення даних на екрані, необхідно натиснути на сенсорну кнопку, щоб швидко повернутися на головний екран треба затиснути кнопку. Всього передбачено 8 режимів. Верхнє значення це максимальне значення, посередині – середнє значення, і відповідно внизу – мінімальне.

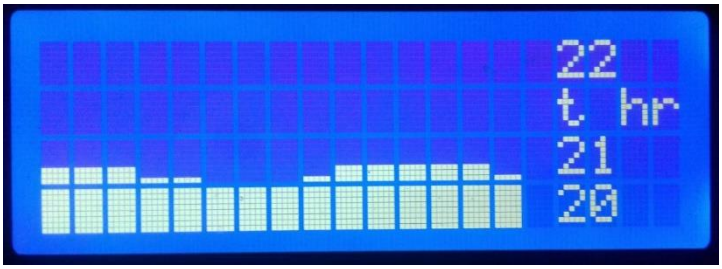

Рисунок 3.20 - Графік зміни температури за годину

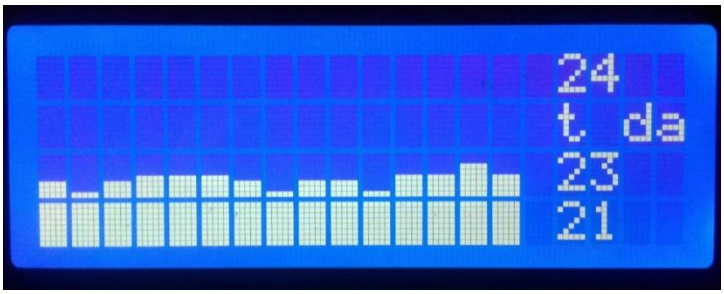

Рисунок 3.21 - Графік зміни температури за день

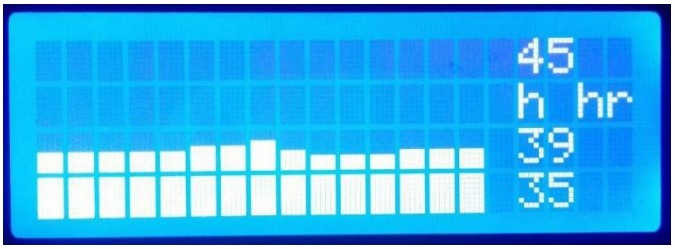

Рисунок 3.22 - Графік зміни вологості за годину

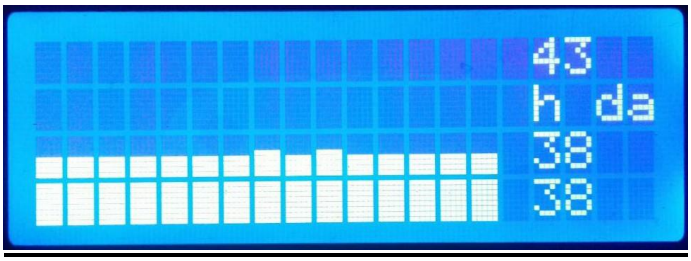

Рисунок 3.23 - Графік зміни вологості за день

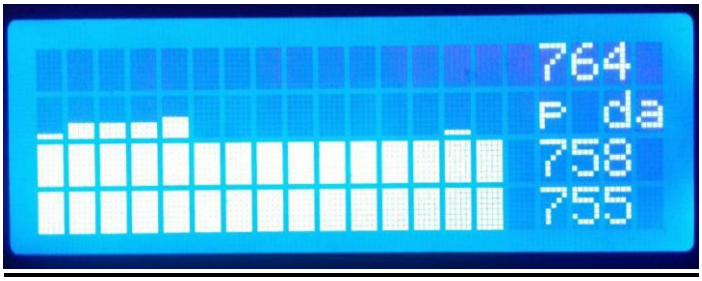

Рисунок 3.24 - Графік зміни атмосферного тиску за годину

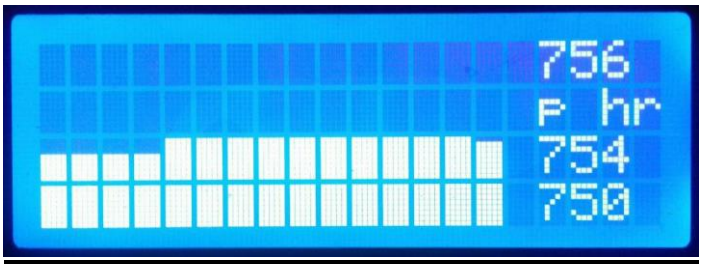

Рисунок 3.25 - Графік зміни атмосферного тиску за день

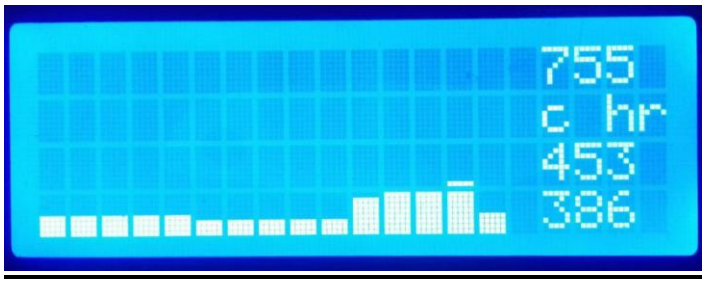

Рисунок 1.26 - Графік зміни рівня вуглекислого газу за годину

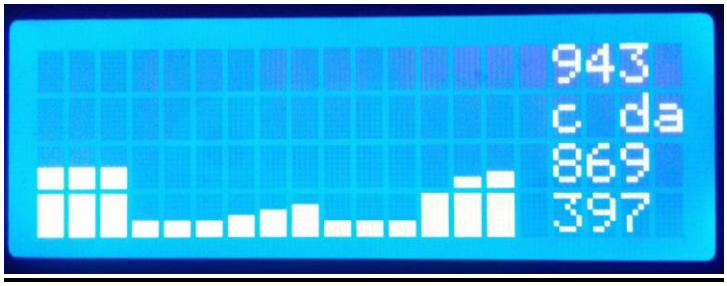

Рисунок 3.27 - Графік зміни рівня вуглекислого газу за день

Що до мобільного додатку, то все просто. Він складається з однієї кнопки і полів для вивелення даних. Метеостанція крім температури, вологості та атмосферного тиску передає ще одне значення Warning. Це значення використовується для оповіщення користувача про різкі зміни деяких показників. Тим самим несодівана зміна декількох показників, наприклад температури і кількості вуглекислого газу, може свідчити про пожежу, а високий рівень метану про витік газу в приміщенні.

Для отримання даних, необхідно ввімкнути додаток там під'єднатися до мережі інтернет, а потом натиснути на кнопку “Get Data” рис.

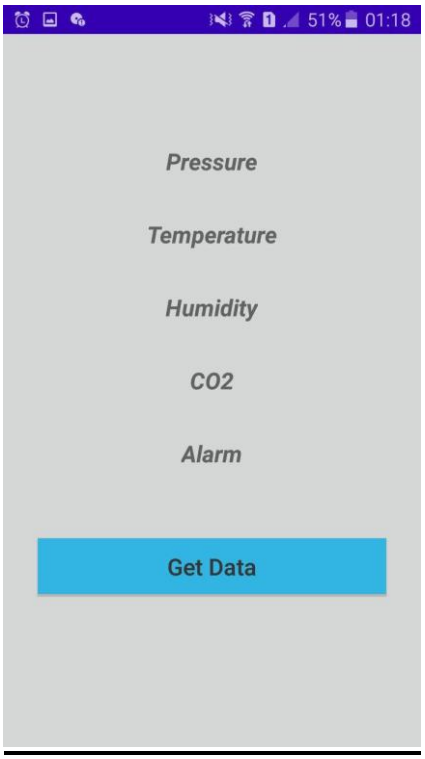

Рисунок 3.28 - Вигляд мобільного додатку на Andoid перед натиканням на кнопку

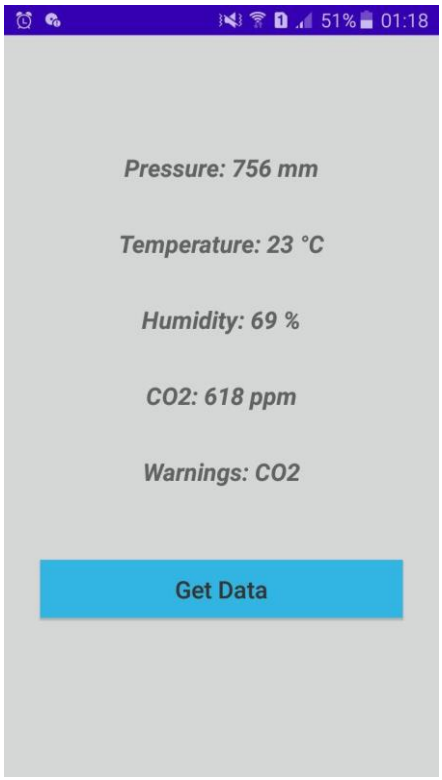

Рисунок 3.29 – Виведення даних отриманих з веб-сервера

ВИСНОВОК

Під час роботи була розроблена інформаційна система домашньої метеорологічної станці на основі платформи Arduino. Даний пристрій має необхідні вбудовані датчики для прогнозування погоди на невеликі проміжки часу, а також контролю мікроклімату в приміщенні.

Для можливості віддаленого контролю показників з метеостанції, був розроблений веб-сервер та мобільний додаток на Android. Метеостанція може бути підключена до мережі Інтернет і надсилати поточні дані на веб-сервер. А користувач в свою чергу через мобільний додаток може отримати ці дані з будь-якої точки світу.

Також при проектуванні інформаційної системи були розроблено:

- діаграму варіантів використання;
- ER-діаграму;
- діаграму класів.

В подальшому даний пристрій може бути розширений зовнішніми датчиками, для отримання більш точних атмосферних даних і відповідно вдосконалений алгоритм прогнозування погоди. Щодо покращень мобільного додатку – можна додати можливість роботи у фоновому режимі і постійного зчитування даних з серверу, це надасть можливість використовувати його як протипожежну сигналізацію.

СПИСОК ЛІТЕРАТУРИ

1. What is a room climate? [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.loxone.com/enen/what-is-room-climate/
2. The Microclimate In The House: Parameters, Requirements And Control [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://navyflex.com.ua/en/the-microclimate-in-the-house-parameters-requirements-and-control/
3. Що таке домашня метеостанція. Точність вимірювань. Радіус дії. [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://opticalmarket.com.ua/ua/scho-take-domashnya-meteostantsija.html
4. Benefits Of Home Automation In Climate Control [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://hdhtech.com/benefits-of-home-automation-in-climate-control/
5. Климат контроль для дома и квартиры: устройство и преимущества системы + тонкости выбора и монтажа [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://sovet-ingenera.com/umniy-dom/klimat-kontrol-dlya-doma.html
6. 10 Real World Applications of Internet of Things (IoT) [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.analyticsvidhya.com/blog/2016/08/10-youtube-videos-explaining-the-real-world-applications-of-internet-of-things-iot/
7. Internet of things (IoT) [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT
8. What is an Arduino? [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://learn.sparkfun.com/tutorials/what-is-an-arduino/all

9. Arduino – Introduction [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.arduino.cc/en/guide/introduction
10. Arduino Technology Architecture and Its Advantages. [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.edgefxkits.com/blog/arduino-technology-architecture-and-applications/
11. Learn Java Programming [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.programiz.com/java-programming
12. Java [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://uk.wikipedia.org/wiki/Java
13. Java vs Kotlin для Android [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://habr.com/ru/post/461877/
14. Работа с Arduino IDE [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://alexgyver.ru/lessons/arduino-ide/
15. Arduino – Software [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.arduino.cc/en/main/software
16. Meet Android Studio [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://developer.android.com/studio/intro
17. How to Use I2C Serial LCD 20X4 [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.instructables.com/id/How-to-Use-I2C-Serial-LCD-20X4-Yellow-Backlight/
18. Библиотека EEPROM [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://doc.arduino.ua/ru/prog/EEPROM
19. Arduino Uno [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://uk.wikipedia.org/wiki/Arduino_Uno
20. Use Case Diagram [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://en.wikipedia.org/wiki/Use_case_diagram

21. UML Use Case Diagram [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.lucidchart.com/pages/uml-use-case-diagram
22. Создание ER-Диаграмм [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: http://inf-teh-lotos.ru/sozdanie-er-diagramm

ДОДАТОК А. СКРИПТИ ЗАПОВНЕННЯ БАЗИ ДАНИХ

В даному додатку наведено скрипти створення таблиці для збереження даних, функції та тригера для очищення старих записів в таблиці.

```
drop table meteo_data;

create table meteo_data (
data_id serial NOT NULL PRIMARY KEY,
meteo_date timestamp,
data_json VARCHAR(1000)
);

CREATE FUNCTION delete_old_rows() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
BEGIN
  DELETE FROM meteo_data WHERE meteo_date < NOW() - INTERVAL '1 days';
  RETURN NULL;
END;
$$;

CREATE TRIGGER trigger_delete_old_rows
    AFTER INSERT ON meteo_data
EXECUTE PROCEDURE delete_old_rows();
```

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНОГО КОДУ МОБІЛЬНОГО ДОДАТКУ

У цьому додатку наведений код мобільного додатку.

com.example.meteoapp.Model.Meteo

```
package com.example.meteoapp.Model;

public class Meteo {
    private String humidity;
    private String temperature;
    private String pressure;
    private String co2;
    private String alarm;

    public String getCO2() {
```

```
        return co2;
    }

    public String getHumidity() {
        return humidity;
    }

    public String getTemperature() {
        return temperature;
    }

    public String getPressure() {
        return pressure;
    }

    public String getAlarm() {
        return alarm;
    }
}
```

com.example.meteoapp.Remote.Client

```
package com.example.meteoapp.Remote;

import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class Client {
    private static Retrofit client = null;

    public static Retrofit getClient(String baseUrl){
        if(client == null){
            client = new Retrofit.Builder()
                    .baseUrl(baseUrl)
                    .addConverterFactory(GsonConverterFactory.create())
                    .build();
        }
        return client;
    }

}
```

com.example.meteoapp.Remote.MeteoService

```
package com.example.meteoapp.Remote;

import com.example.meteoapp.Model.Meteo;
import retrofit2.http.GET;
import retrofit2.Call;

public interface MeteoService {
    @GET("/getJsonData")
    Call<Meteo> getData();
}
```

com.example.meteoapp.Common

```
package com.example.meteoapp;

import com.example.meteoapp.Remote.Client;
import com.example.meteoapp.Remote.MeteoService;

public class Common {
    private static final String BASE_URL = "http://95.181.179.92:50301";

    public static MeteoService getMeteoService()
    {
        return Client.getClient(BASE_URL).create(MeteoService.class);
    }

}
```

com.example.meteoapp.MainActivity

```
package com.example.meteoapp;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.example.meteoapp.Model.Meteo;
import com.example.meteoapp.Remote.MeteoService;

import dmax.dialog.SpotsDialog;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class MainActivity extends AppCompatActivity {

    MeteoService mService;
    TextView txtCO2;
    TextView txtHum;
    TextView txtPressure;
    TextView txtTemp;
    TextView txtAlarm;
    Button btnGetMeteo;
    SpotsDialog dialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mService = Common.getMeteoService();

        btnGetMeteo = (Button)findViewById((R.id.btnGetMeteo));
```

```
        txtAlarm = (TextView)findViewById(R.id.txtAlarm);
        txtCO2 = (TextView)findViewById(R.id.txtCO2);
        txtHum = (TextView)findViewById(R.id.txtHum);
        txtTemp = (TextView)findViewById(R.id.txtTemp);
        txtPressure = (TextView)findViewById(R.id.txtPressure);
        dialog = new SpotsDialog(MainActivity.this);

        btnGetMeteo.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view){
                dialog.show();
                getMeteoData();
            }
        });
    }

    private void getMeteoData() {
        mService.getData().enqueue(new Callback<Meteo>() {
            @Override
            public void onResponse(Call<Meteo> call, Response<Meteo> response) {
                dialog.dismiss();
                txtCO2.setText("CO2: " + response.body().getCO2() + " ppm");
                txtHum.setText("Humidity: " + response.body().getHumidity() + "
%");
                txtTemp.setText("Temperature: " +
response.body().getTemperature() + " °C");
                txtPressure.setText("Pressure: " + response.body().getPressure()
+ " mm");
                if(response.body().getAlarm().equals("0"))
                    txtAlarm.setText("Warnings: Ok");
                else if(response.body().getAlarm().equals("1"))
                    txtAlarm.setText("Warnings: Methane");
                else if(response.body().getAlarm().equals("2"))
                    txtAlarm.setText("Warnings: Temperature");
                else if(response.body().getAlarm().equals("3"))
                    txtAlarm.setText("Warnings: CO2");
                else if(response.body().getAlarm().equals("4"))
                    txtAlarm.setText("Warnings: Smoke");
                else if(response.body().getAlarm().equals("5"))
                    txtAlarm.setText("Warnings: Temperature and CO2");
                else if(response.body().getAlarm().equals("6"))
                    txtAlarm.setText("Warnings: Temperature and Smoke");
                else if(response.body().getAlarm().equals("7"))
                    txtAlarm.setText("Warnings: CO2 and Smoke");
                else
                    txtAlarm.setText("Warnings: Temperature, Smoke and CO2");
            }

            @Override
            public void onFailure(Call<Meteo> call, Throwable t) {
                Log.e("ERROR",t.getMessage());
                dialog.dismiss();
            }
        });

        }
    }
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="match_parent"
    android:background="?attr/colorButtonNormal"
    android:orientation="horizontal"
    tools:context=".MainActivity">
    android:statusBarColor=""

    <Button
        android:id="@+id/btnGetMeteo"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="@dimen/_110sdp"
        android:background="@android:color/holo_blue_light"
        android:text="Get Data"
        android:textAllCaps="false"
        android:textColor="#393636"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/txtAlarm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btnGetMeteo"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="@dimen/_15sdp"
        android:layout_marginBottom="@dimen/_50sdp"
        android:text="Alarm"
        android:textSize="20sp"
        android:textStyle="bold|italic" />

    <TextView
        android:id="@+id/txtCO2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/txtAlarm"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="@dimen/_15sdp"
        android:layout_marginBottom="@dimen/_15sdp"
        android:text="CO2"
        android:textSize="20sp"
        android:textStyle="bold|italic" />

    <TextView
        android:id="@+id/txtHum"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/txtCO2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="@dimen/_15sdp"
        android:layout_marginBottom="@dimen/_15sdp"
```

```
        android:text="Humidity"
        android:textSize="20sp"
        android:textStyle="bold|italic" />

    <TextView
        android:id="@+id/txtTemp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/txtHum"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="@dimen/_15sdp"
        android:layout_marginBottom="@dimen/_15sdp"
        android:text="Temperature"
        android:textSize="20sp"
        android:textStyle="bold|italic" />

    <TextView
        android:id="@+id/txtPressure"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/txtTemp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="@dimen/_15sdp"
        android:layout_marginBottom="@dimen/_15sdp"
        android:text="Pressure"
        android:textSize="20sp"
        android:textStyle="bold|italic" />

</RelativeLayout>
```

ДОДАТОК В. ЛІСТИНГ ПРОГРАМНОГО КОДУ ДЛЯ ARDUINO

```
// ---------- Settings ----------------
#define SENS_TIME 10000 //data refresh time on display


// pins
#define PIN_MQ135  A0
#define PIN_MQ4  A0
#define LED_MODE 0
#define DHTPIN 2
#define DHTTYPE DHT11
#define BLUE_YELLOW 1
#define LED_R 5
#define LED_G 6
#define LED_B 11
#define BTN_PIN 3


// chart displaying measures
#define TEMP_MIN 15
```

```
#define TEMP_MAX 35
#define HUM_MIN 0
#define HUM_MAX 100
#define PRESS_MIN -100
#define PRESS_MAX 100
#define CO2_MIN 200
#define CO2_MAX 2000

// libraries
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 20, 4);

//#include <TimeLib.h>
#include "RTClib.h"
RTC_DS1307 rtc;
DateTime now;

#include <TroykaMQ.h>
MQ135 mq135(PIN_MQ135);
MQ4 mq4(PIN_MQ4);

#include "DHT.h"
DHT dht(DHTPIN, DHTTYPE);

#include <GyverTimer.h>
GTimer_ms sensorsTimer(SENS_TIME);
GTimer_ms drawSensorsTimer(SENS_TIME);
GTimer_ms clockTimer(500);
GTimer_ms hourPlotTimer((long)4 * 60 * 1000);          // 4 минуты
GTimer_ms dayPlotTimer((long)1.6 * 60 * 60 * 1000);    // 1.6 часа
GTimer_ms plotTimer(240000);
GTimer_ms predictTimer((long)10 * 60 * 1000);          // 10 минут

#include "GyverButton.h"
GButton button(BTN_PIN, LOW_PULL, NORM_OPEN);

int8_t hrs, mins, secs;
byte mode = 0;
```

```
/*
  0 clock and data
  1 temperature chart per hour
  2 temperature chart per day
  3 humidity chart per hour
  4 humidity chart per day
  5 pressure chart per hour
  6 pressure chart per day
  7 co2 chart per hour
  8 co2 chart per day
*/

// variables for output
float dispTemp = 23;
int dispHum = 69;
int dispPres = 760;
int dispCO2;
int methane;
int dispRain = 23;

// graph arrays
int tempHour[15], tempDay[15];
int humHour[15], humDay[15];
int pressHour[15], pressDay[15];
int co2Hour[15], co2Day[15];
int delta;
uint32_t pressure_array[6];
uint32_t sumX, sumY, sumX2, sumXY;
float a, b;
byte time_array[6];

//RGB led pins
int redPin = 11;
int greenPin = 12;
int bluePin = 13;

static const char *daysOfTheWeek[] = {"Sun", "Mon", "Tue", "Wed", "Thu",
"Fri", "Sat"};
```

```
      // digits
      byte LT[8] = {B00111, B01111, B11111, B11111, B11111, B11111, B11111,
B11111};
      byte UB[8] = {B11111, B11111, B11111, B00000, B00000, B00000, B00000,
B00000};
      byte RT[8] = { B11100, B11110, B11111, B11111, B11111, B11111, B11111,
B11111};
      byte LL[8] = {B11111, B11111, B11111, B11111, B11111, B11111, B01111,
B00111};
      byte LB[8] = {B00000, B00000, B00000, B00000, B00000, B11111, B11111,
B11111};
      byte LR[8] = {B11111, B11111, B11111, B11111, B11111, B11111, B11110,
B11100};
      byte MB[8] = {B11111, B11111, B11111, B00000, B00000, B00000, B11111,
B11111};
      byte block[8] = {B11111, B11111, B11111, B11111, B11111, B11111, B11111,
B11111};

      //graph
      byte row8[8] = {0b11111,  0b11111,  0b11111,  0b11111,  0b11111,  0b11111,
0b11111,  0b11111};
      byte row7[8] = {0b00000,  0b11111,  0b11111,  0b11111,  0b11111,  0b11111,
0b11111,  0b11111};
      byte row6[8] = {0b00000,  0b00000,  0b11111,  0b11111,  0b11111,  0b11111,
0b11111,  0b11111};
      byte row5[8] = {0b00000,  0b00000,  0b00000,  0b11111,  0b11111,  0b11111,
0b11111,  0b11111};
      byte row4[8] = {0b00000,  0b00000,  0b00000,  0b00000,  0b11111,  0b11111,
0b11111,  0b11111};
      byte row3[8] = {0b00000,  0b00000,  0b00000,  0b00000,  0b00000,  0b11111,
0b11111,  0b11111};
      byte row2[8] = {0b00000,  0b00000,  0b00000,  0b00000,  0b00000,  0b00000,
0b11111,  0b11111};
      byte row1[8] = {0b00000,  0b00000,  0b00000,  0b00000,  0b00000,  0b00000,
0b00000,  0b11111};

      void setup () {
        Serial.begin(9600);
```

```
        pinMode(LED_R, OUTPUT);
        pinMode(LED_G, OUTPUT);
        pinMode(LED_B, OUTPUT);
        setLED(0);

        Wire.begin();

        lcd.init(); // initialize the lcd
        lcd.backlight();
        lcd.home();
        dht.begin();
        lcd.clear(); // clear display before new values will be settled

        // following line sets the RTC to the date & time this sketch was compiled
        rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

        now = rtc.now();
        secs = now.second();
        mins = now.minute();
        hrs = now.hour();

        calibrate_modules ();
        readSensors();
        digitalClockDisplay(hrs, mins);
        dateDisplay();
        dispSensors();
      }

      void loop () {
        readSensors();
        if (clockTimer.isReady()) clockTick();         // count the time twice a
second
        plotSensorsTick();                                  // several timers for
recalculating charts (per hour, per day and forecast)
        modesTick();                                      // catch button click and
change modes
        if (sendTimer.isReady()) sendingToServer();
        if (mode == 0) {                                    // main page mode
```

```
          if (drawSensorsTimer.isReady()) dispSensors();  // refresh sensors value
with period SENS_TIME
        } else {                                          // chart mode
          if (plotTimer.isReady()) redrawPlot();          // refresh chart
        }
      }

      void dateDisplay() {
        lcd.setCursor(15, 0);
        if (now.day() < 10) lcd.print(0);
        lcd.print(now.day());
        lcd.print(".");
        if (now.month() < 10) lcd.print(0);
        lcd.print(now.month());

        lcd.setCursor(16, 1);
        int dayofweek = now.dayOfTheWeek();
        lcd.print(daysOfTheWeek[dayofweek]);
        lcd.setCursor(19, 1);
        lcd.print(" ");
      }

      void digitalClockDisplay(byte hours, byte minutes) {
        // digital clock display of the time
        DateTime now = rtc.now();

        printDigits(now.hour() / 10, 0); //hours
        printDigits(now.hour() % 10, 4);

        lcd.setCursor(7, 0); //dots between minutes and hours
        lcd.write(165);
        lcd.setCursor(7, 1);
        lcd.write(165);

        printDigits(now.minute() / 10, 8); //minutes
        printDigits(now.minute() % 10, 12);
      }

      void loadClock() {
```

```
  lcd.createChar(0, LT);
  lcd.createChar(1, UB);
  lcd.createChar(2, RT);
  lcd.createChar(3, LL);
  lcd.createChar(4, LB);
  lcd.createChar(5, LR);
  lcd.createChar(6, MB);
  lcd.createChar(7, block);
}

void printDigits(byte digits, byte x) {
  // display digits for clock

  switch (digits) {
    case 0:
      lcd.setCursor(x, 0); // set cursor to column 0, line 0 (first row)
      lcd.write(0); // call each segment to create
      lcd.write(1); // top half of the number
      lcd.write(2);
      lcd.setCursor(x, 1); // set cursor to colum 0, line 1 (second row)
      lcd.write(3); // call each segment to create
      lcd.write(4); // bottom half of the number
      lcd.write(5);
      break;
    case 1:
      lcd.setCursor(x, 0);
      lcd.write(1);
      lcd.write(2);
      lcd.print(" ");
      lcd.setCursor(x, 1);
      lcd.write(4);
      lcd.write(7);
      lcd.write(4);
      break;
    case 2:
      lcd.setCursor(x, 0);
      lcd.write(6);
      lcd.write(6);
      lcd.write(2);
```

```
  lcd.setCursor(x, 1);
  lcd.write(3);
  lcd.write(4);
  lcd.write(4);
  break;
case 3:
  lcd.setCursor(x, 0);
  lcd.write(6);
  lcd.write(6);
  lcd.write(2);
  lcd.setCursor(x, 1);
  lcd.write(4);
  lcd.write(4);
  lcd.write(5);
  break;
case 4:
  lcd.setCursor(x, 0);
  lcd.write(3);
  lcd.write(4);
  lcd.write(7);
  lcd.setCursor(x, 1);
  lcd.print(" ");
  lcd.print(" ");
  lcd.write(7);
  break;
case 5:
  lcd.setCursor(x, 0);
  lcd.write(3);
  lcd.write(6);
  lcd.write(6);
  lcd.setCursor(x, 1);
  lcd.write(4);
  lcd.write(4);
  lcd.write(5);
  break;
case 6:
  lcd.setCursor(x, 0);
  lcd.write(0);
  lcd.write(6);
```

```
      lcd.write(6);
      lcd.setCursor(x, 1);
      lcd.write(3);
      lcd.write(4);
      lcd.write(5);
      break;
    case 7:
      lcd.setCursor(x, 0);
      lcd.write(1);
      lcd.write(1);
      lcd.write(2);
      lcd.setCursor(x, 1);
      lcd.print(" ");
      lcd.print(" ");
      lcd.write(7);
      break;
    case 8:
      lcd.setCursor(x, 0);
      lcd.write(0);
      lcd.write(6);
      lcd.write(2);
      lcd.setCursor(x, 1);
      lcd.write(3);
      lcd.write(4);
      lcd.write(5);
      break;
    case 9:
      lcd.setCursor(x, 0);
      lcd.write(1);
      lcd.write(2);
      lcd.print(" ");
      lcd.setCursor(x, 1);
      lcd.write(4);
      lcd.write(7);
      lcd.write(4);
      break;
  }
}
```

```
      void loadPlot() {
        lcd.createChar(0, row8);
        lcd.createChar(1, row1);
        lcd.createChar(2, row2);
        lcd.createChar(3, row3);
        lcd.createChar(4, row4);
        lcd.createChar(5, row5);
        lcd.createChar(6, row6);
        lcd.createChar(7, row7);
      }

      void drawPlot(byte pos, byte row, byte width, byte height, int min_val, int
max_val, int *plot_array, String label) {
        int max_value = -32000;
        int min_value = 32000;

        for (byte i = 0; i < 15; i++) {
          if (plot_array[i] > max_value) max_value = plot_array[i];
          if (plot_array[i] < min_value) min_value = plot_array[i];
        }
        lcd.setCursor(16, 0); lcd.print(max_value);
        lcd.setCursor(16, 1); lcd.print(label);
        lcd.setCursor(16, 2); lcd.print(plot_array[14]);
        lcd.setCursor(16, 3); lcd.print(min_value);

        for (byte i = 0; i < width; i++) {
          int fill_val = plot_array[i];
          fill_val = constrain(fill_val, min_val, max_val);
          byte infill, fract;
          if (plot_array[i] > min_val)
            infill = floor((float)(plot_array[i] - min_val) / (max_val - min_val)
* height * 10);
          else infill = 0;
          fract = (float)(infill % 10) * 8 / 10;
          infill = infill / 10;

          for (byte n = 0; n < height; n++) {
            if (n < infill && infill > 0) {
              lcd.setCursor(i, (row - n));
```

```
            lcd.write(0);
          }
          if (n >= infill) {
            lcd.setCursor(i, (row - n));
            if (fract > 0) lcd.write(fract);
            else lcd.write(16);
            for (byte k = n + 1; k < height; k++) {
              lcd.setCursor(i, (row - k));
              lcd.write(16);
            }
            break;
          }
        }
      }
}
      void calibrate_modules (){
        mq4.calibrate();
        mq135.calibrate();
      }

      void readSensors(){
        //bme.takeForcedMeasurement();
        dispTemp = dht.readTemperature();
        dispHum = dht.readHumidity();
        dispPres = (float)bme.readPressure() * 0.00750062;
        methane = mq4.readMethane();
        dispCO2 = mq135.readCO2();

        if (dispCO2 < 700) setLED(2);
        else if (dispCO2 < 1200) setLED(3);
        else if (dispCO2 >= 1200) setLED(1);

        if(methane >= 4)
        {
          tone(6, note, noteDuration*100);
        }
      }

      void dispSensors(){
```

```
  lcd.setCursor(0, 2);
  lcd.print(String(dispTemp)+".0");
  lcd.write(223);
  lcd.setCursor(5, 2);
  lcd.print(" " + String(dispHum) + "%  ");

  lcd.print(String(dispCO2) + " ppm");
  if (dispCO2 < 1000) lcd.print(" ");

  lcd.setCursor(0, 3);
  lcd.print(String(dispPres) + "mm rain ");
  //lcd.print(F("       "));
  lcd.setCursor(11, 3);
  lcd.print(String(dispRain) + "% gas ");
  lcd.setCursor(19, 3);
  if(methane < 10)lcd.print(String(methane));
  else lcd.print("X");
}

void setLED(byte color) {
  // сначала всё выключаем
  if (!LED_MODE) {
    analogWrite(LED_R, 0);
    analogWrite(LED_G, 0);
    analogWrite(LED_B, 0);
  } else {
    analogWrite(LED_R, 255);
    analogWrite(LED_G, 255);
    analogWrite(LED_B, 255);
  }
  switch (color) {
    case 0:
      break;
    case 1: analogWrite(LED_R, 255);
      break;
    case 2: analogWrite(LED_G, 255);
      break;
    case 3:
      analogWrite(LED_B, 255);
```

```
            break;
        }
      }

      void modesTick() {
        button.tick();
        boolean changeFlag = false;
        if (button.isClick()) {
          mode++;
         if (mode > 8) mode = 0;
          changeFlag = true;
        }
        if (button.isHolded()) {
          mode = 0;
          changeFlag = true;
        }

        if (changeFlag) {
          if (mode == 0) {
            lcd.clear();
            loadClock();
            digitalClockDisplay(hrs, mins);
            dispDate();
            readSensors();
            dispSensors();
          } else {
            lcd.clear();
            loadPlot();
            redrawPlot();
          }
        }
      }

      void redrawPlot() {
        lcd.clear();
        switch (mode) {
          case 1: drawPlot(0, 3, 15, 4, TEMP_MIN, TEMP_MAX, (int*)tempHour, "t
hr");
            break;
```

```
          case 2: drawPlot(0, 3, 15, 4, TEMP_MIN, TEMP_MAX, (int*)tempDay, "t
day");
            break;
          case 3: drawPlot(0, 3, 15, 4, HUM_MIN, HUM_MAX, (int*)humHour, "h hr");
            break;
          case 4: drawPlot(0, 3, 15, 4, HUM_MIN, HUM_MAX, (int*)humDay, "h day");
            break;
          case 5: drawPlot(0, 3, 15, 4, PRESS_MIN, PRESS_MAX, (int*)pressHour, "p
hr");
            break;
          case 6: drawPlot(0, 3, 15, 4, PRESS_MIN, PRESS_MAX, (int*)pressDay, "p
day");
            break;
          case 7: drawPlot(0, 3, 15, 4, CO2_MIN, CO2_MAX, (int*)co2Hour, "c hr");
            break;
          case 8: drawPlot(0, 3, 15, 4, CO2_MIN, CO2_MAX, (int*)co2Day, "c day");
            break;
        }
      }

      void plotSensorsTick() {
        // 4 минутный таймер
        if (hourPlotTimer.isReady()) {
          for (byte i = 0; i < 14; i++) {
            tempHour[i] = tempHour[i + 1];
            humHour[i] = humHour[i + 1];
            pressHour[i] = pressHour[i + 1];
            co2Hour[i] = co2Hour[i + 1];
          }

          tempHour[14] = dispTemp;
          humHour[14] = dispHum;
          co2Hour[14] = dispCO2;
          pressHour[14] = dispPres;

          if(methane >= 3) alarm = 1;
          else if(fabs(tempHour[14] - tempHour[13]) >= 4) alarm = 2;
          else if(fabs(co2Hour[14] - co2Hour[13]) >= 150) alarm = 3;
```

```
      else if(fabs(tempHour[14] - tempHour[13]) >= 4 && fabs(co2Hour[14] -
co2Hour[13]) >= 150) alarm = 4;
      else alarm = 0;
    }

    // 1.5 часовой таймер
    if (dayPlotTimer.isReady()) {
      long averTemp = 0, averHum = 0, averPress = 0, averCO2 = 0;

      for (byte i = 0; i < 15; i++) {
        averTemp += tempHour[i];
        averHum += humHour[i];
        averPress += pressHour[i];
        averCO2 += co2Hour[i];
      }
      averTemp /= 15;
      averHum /= 15;
      averPress /= 15;
      averCO2 /= 15;

      for (byte i = 0; i < 14; i++) {
        tempDay[i] = tempDay[i + 1];
        humDay[i] = humDay[i + 1];
        pressDay[i] = pressDay[i + 1];
        co2Day[i] = co2Day[i + 1];
      }
      tempDay[14] = averTemp;
      humDay[14] = averHum;
      pressDay[14] = averPress;
      co2Day[14] = averCO2;
    }

    // 10 минутный таймер
    if (predictTimer.isReady()) {
      long averPress = 0;
      long averPress = 7550;
      for (byte i = 0; i < 10; i++) {
        bme.takeForcedMeasurement();
        averPress += bme.readPressure();
```

```
      delay(1);
    }
    averPress /= 10;

    for (byte i = 0; i < 5; i++) {
      pressure_array[i] = pressure_array[i + 1];
    }
    pressure_array[5] = averPress;
    sumX = 0;
    sumY = 0;
    sumX2 = 0;
    sumXY = 0;
    for (int i = 0; i < 6; i++) {
      sumX += time_array[i];
      sumY += (long)pressure_array[i];
      sumX2 += time_array[i] * time_array[i];
      sumXY += (long)time_array[i] * pressure_array[i];
    }
    a = 0;
    a = (long)6 * sumXY;
    a = a - (long)sumX * sumY;
    a = (float)a / (6 * sumX2 - sumX * sumX);
    delta = a * 6;      // расчёт изменения давления
    dispRain = map(delta, -250, 250, 100, -100);
    }
}

boolean dotFlag;
void clockTick() {
  dotFlag = !dotFlag;
  if (dotFlag) {          // every seconr refresh time
    secs++;
    if (secs > 59) {      // every min
      secs = 0;
      mins++;
      if (mins <= 59 && mode == 0) digitalClockDisplay(hrs, mins);
    }
    if (mins > 59) {      // every hour
```

```
      now = rtc.now();
      secs = now.second();
      mins = now.minute();
      hrs = now.hour();
      if (mode == 0) digitalClockDisplay(hrs, mins);
      if (hrs > 23) {
        hrs = 0;
      }
      if (mode == 0) dispDate();
    }
  }
  if (dispCO2 >= 1200) {
    if (dotFlag) setLED(1);
    else setLED(3);
  }
}

void sendingToServer()
 {
   if (client.connect(server, 50301)) {
    client.print(F("POST /postJsonData?humidity="));     //YOUR URL
    client.print(static_cast<String>(dispHum));
    client.print(F("&temperature="));
    client.print(static_cast<String>(dispTemp));
    client.print(F("&pressure="));
    client.print(static_cast<String>(dispPres));
    client.print(F("&co2="));
    client.print(static_cast<String>(dispCO2));
    client.print(F("&methane="));
    client.print(static_cast<String>(methane));
    client.print(F("&alarm="));
    client.print(static_cast<String>(alarm));
    client.print(" ");      //SPACE BEFORE HTTP/1.1
    client.print(F("HTTP/1.1"));
    client.println();
    client.println(F("Host: <Your Local IP>"));
    client.println("Connection: close");
    client.println();
  } else {
```

```
          // if you didn't get a connection to the server:
          Serial.println(F("connection failed"));
        }
 }
```

ДОДАТОК Г. ЛІСТИНГ ПРОГРАМНОГО КОДУ ВЕБ-СЕРВЕРНОЇ ЧАСТИНИ

com.meteo.ConnctionManager

```
package com.meteo;

import org.postgresql.ds.PGPoolingDataSource;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.util.Properties;

/**
 * Database connection management.
 */
public class ConnectionManager {
    private static final Logger LOGGER =
LoggerFactory.getLogger(ConnectionManager.class);
    private static Properties properties = new Properties();
    public static final PGPoolingDataSource SIMPLE_DATA_SOURCE = new
PGPoolingDataSource();
    static int opened = 0;

    static {
        //read property from file
        //set properties to dataSource
        try {
            properties.load(new
FileInputStream("configurations/db.properties"));
            SIMPLE_DATA_SOURCE.setServerName(properties.getProperty("url"));

SIMPLE_DATA_SOURCE.setDatabaseName(properties.getProperty("database"));
            SIMPLE_DATA_SOURCE.setUser(properties.getProperty("username"));
            SIMPLE_DATA_SOURCE.setPassword(properties.getProperty("password"));
            SIMPLE_DATA_SOURCE.setMaxConnections(150);
            SIMPLE_DATA_SOURCE.setInitialConnections(10);

        } catch (IOException e) {
            LOGGER.error("Error in reading configuration file " + e);
        }
    }

    /**
     * Method for database connection
```

```
     *
     * @return connection
     */
    public static Connection getSimpleConnection() {
        Connection conn = null;
        try {
            conn = SIMPLE_DATA_SOURCE.getConnection();
            opened++;
        } catch (Exception e) {
            LOGGER.error("Error in getting connection for " +
properties.getProperty("url"), e);
        }
        return conn;
    }
}
```

com.meteo.Controller

```
package com.meteo;

import org.json.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import spark.ModelAndView;
import spark.Request;
import spark.Response;
import spark.Route;
import spark.template.thymeleaf.ThymeleafTemplateEngine;

import java.util.HashMap;

public class Controller {

    private final DataDAO dataDAO = new DataDAO();

    private static final Logger LOGGER =
LoggerFactory.getLogger(Controller.class);

    public final Route getIndex = ((request, response) -> {
        HashMap<String, Object> model = new HashMap<>();
        return new ThymeleafTemplateEngine().render(new ModelAndView(model,
"index"));
    });

    public final Route postJsonData = ((request, response) -> {

        String temperature = parseParameter(request, response, "temperature");
        String humidity = parseParameter(request, response, "humidity");
        String pressure = parseParameter(request, response, "pressure");
        String co2 = parseParameter(request, response, "co2");
        String methane = parseParameter(request, response, "methane");
        int alarm = parseAlarm(request, response);
        JSONObject object = new JSONObject();
```

```
        object.put("temperature", temperature);
        object.put("humidity", humidity);
        object.put("pressure", pressure);
        object.put("co2", co2);
        object.put("methane", methane);
        object.put("alarm", alarm);

        dataDAO.writeData(object.toString());
        return "ok";
    });

    public final Route getJsonData = ((request, response) -> {
        HashMap<String, Object> model = new HashMap<>();

        String meteoData = dataDAO.getData();
        if (meteoData == null || "".equals(meteoData)){
            meteoData = "{}";
        }
        model.put("meteoData", meteoData);
        return new ThymeleafTemplateEngine().render(new ModelAndView(model,
"data"));
    });

    private String parseParameter(Request request, Response response, String
parameterName) {
        String parameter = "";
        try {
            parameter = request.queryMap(parameterName).value();
        } catch (Exception e) {
            LOGGER.error("Error on parsing parameter", e);
        }
        if (parameter == null) {
            response.redirect("/index");
        }
        return parameter;
    }

    private int parseAlarm(Request request, Response response) {
        int alarm = 1;
        try {
            alarm = Integer.parseInt(request.queryMap("alarm").value());
        } catch (Exception e) {
            LOGGER.error("Error on parsing alarm", e);
        }
        return alarm;
    }
}
```

com.meteo.DataDAO

```
package com.meteo;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.sql.*;

public class DataDAO {
    private static final Logger LOGGER = LoggerFactory.getLogger(DataDAO.class);

    private static final String WRITE_DATA = "insert into meteo_data
(meteo_date, data_json) values (current_timestamp, ?)";
    private static final String GET_DATA = "select data_json from meteo_data
where data_id = (select max(data_id) from meteo_data)";

    public void writeData(String jsonString) {
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement statement =
connection.prepareStatement(WRITE_DATA)) {
            statement.setString(1, jsonString);
            statement.execute();
        } catch (SQLException e) {
            LOGGER.error("Error while WRITE_DATA ", e);
        }
    }

    public String getData() {
        String data = null;
        try (Connection connection = ConnectionManager.getSimpleConnection();
             PreparedStatement statement =
connection.prepareStatement(GET_DATA)) {
            ResultSet resultSet = statement.executeQuery();
            resultSet.next();
            data = resultSet.getString(1);
        } catch (SQLException e) {
            LOGGER.error("Error while GET_DATA ", e);
        }
        return data;
    }
}
```

com.meteo.Main

```
package com.meteo;

import org.eclipse.jetty.server.Server;
import org.eclipse.jetty.util.thread.ThreadPool;
import spark.Spark;
import spark.embeddedserver.EmbeddedServers;
```

```
import spark.embeddedserver.jetty.EmbeddedJettyFactory;
import spark.embeddedserver.jetty.JettyServerFactory;

import static spark.Spark.*;

public class Main {

    private static final Controller CONTROLLER = new Controller();
    public static void main(String[] args) {
        Spark.exception(Exception.class, (exception, request, response) -> {
            exception.printStackTrace();
        });
        System.out.println("started");
        port(50301);
        CustomJettyServerFactory customJettyServerFactory = new
CustomJettyServerFactory();
        EmbeddedServers.add(
                EmbeddedServers.Identifiers.JETTY,
                new EmbeddedJettyFactory(customJettyServerFactory));
        staticFileLocation("/public");

        get("/index", CONTROLLER.getIndex);
        get("/", CONTROLLER.getIndex);

        post("/postJsonData", CONTROLLER.postJsonData);
        get("/getJsonData", CONTROLLER.getJsonData);

        //if not found page
        get("/*", (((request, response) -> {
            if (response.status() != 200) response.redirect("/index");
            return null;
        })));
    }

    static class CustomJettyServerFactory implements JettyServerFactory {
        @Override
        public Server create(int maxThreads, int minThreads, int
threadTimeoutMillis) {
            Server server = new Server();

server.setAttribute("org.eclipse.jetty.server.Request.maxFormContentSize",
1500000);
            return server;
        }

        @Override
        public Server create(ThreadPool threadPool) {
            return null;
        }
    }
}
```