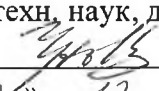


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ У МЕДИЦИНІ ТА ТЕЛЕКОМУНІКАЦІЯ

УДК 621.3: 455.63

«Допущено до захисту»

Завідувач кафедри біомедичної
інженерії та телекомунікацій
к-т техн. наук, доц. Чухов В. В.

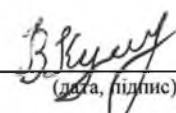

« 19 » « 12 » 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему Дослідження дистанційної системи моніторингу якості повітря
у приміщенні.

за спеціальністю 172 «Телекомунікації та радіотехніка»
освітній ступінь «магістр»

Виконавець роботи
студент 2 курсу, гр. ТРМ-22-1


(дата, підпис)

Кушнірук В.Л.

Керівник роботи
к.т.н., доц.


(дата, підпис)

Андреев О.В.

Житомир
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ У МЕДИЦИНІ ТА ТЕЛЕКОМУНІКАЦІЯ

Пояснювальна записка до кваліфікаційної магістерської роботи

на тему Дослідження дистанційної системи моніторингу якості повітря у приміщенні.

Виконав: студент 2 курсу,
групи ТРм-22-1 спеціальності
172 «Телекомунікації та радіотехніка»

Кушнірук В.Л.

Керівник Андреєв О.В.

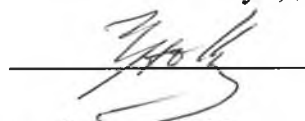
Рецензент Бенедицький В.Б.

Житомир
2023

«Затверджено»

Завідуючий кафедрою КТМ та Т

к-т техн. наук, доцент

 Чухов В. В.

« 27 » 10 2022 р.

Завдання на кваліфікаційну магістерську роботу

Кушнірук Віталій Леонідович

(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження дистанційної системи моніторингу якості повітря у приміщенні.

затверджено наказом по університету від «22» листопада 2023 р. № 679

2. Мета дослідження: визначення середньодобової точності вимірів якості повітря у приміщенні із застосуванням технології IoT при дистанційному добовому контролі.

3. Вихідні дані: Для контролю якості повітря використовуються датчики вологості, температури та концентрації вуглеводних газів.

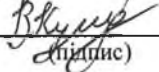
4. Основні задачі дослідження: Розробити проект IoT системи моніторингу якості повітря у приміщенні з відображенням даних вимірів у хмарному сервісі. Розробити методику розрахунку та дослідити зміну середньодобової точності вимірів вологості, температури та концентрації вуглеводних газів.

5. Термін подання роботи до захисту 2 грудня 2023р.

6. Дата видачі завдання 27 жовтня 2022 р.

Науковий керівник

Завдання прийняв
до виконання


(підпис)

(підпис)

Андрєєв О. В.
(прізвище, ініціали)

Кушнірук В. Л.
(прізвище, ініціали)

АНОТАЦІЯ

Кваліфікаційна робота спрямована на визначення середньодобової точності вимірювань якості повітря у приміщенні за допомогою технології Інтернету речей (IoT). Для цього використовуються датчики вологості, температури та концентрації вуглеводневих газів.

Основною метою є розробка методики розрахунку IoT системи моніторингу, яка забезпечить дистанційний добовий контроль та відображення даних в хмарному сервісі.

ANNOTATION

The qualification work is aimed at determining the average daily accuracy of indoor air quality measurements using the Internet of Things (IoT) technology. For this purpose, humidity, temperature, and hydrocarbon gas concentration sensors are used.

The main goal is to develop a methodology for calculating the IoT monitoring system, which will provide remote daily monitoring and display of data in the cloud service.

РЕФЕРАТ

Мета магістерської атестаційної роботи полягає у визначенні середньодобової точності вимірів якості повітря у приміщенні при застосуванні технології IoT для дистанційного добового контролю. Дослідження базується на використанні датчиків вологості, температури та концентрації вуглеводневих газів для ефективного контролю параметрів мікроклімату.

Обсяг атестаційної магістерської роботи - 81 сторінка. Робота складається з чотирьох розділів та містить 29 рисунків, 5 таблиць, 2 додатки та 13 джерел згідно з переліком посилань.

У першому розділі проведено огляд літературних джерел, де розглянуті параметри мікроклімату у приміщенні, такі як температура, вологість повітря та концентрація вуглекислого газу. Також вивчено сервіс "ThingSpeak", який використовується для відображення даних вимірів у хмарному сервісі.

У другому розділі проведено розробку моделі та вибір основних компонентів для визначення параметрів мікроклімату. Розроблена структурна схема та алгоритм роботи системи, вибрані компоненти для пристрою та розроблена електрична принципова схема. Також визначена методика розрахунку середньодобової точності вимірювань.

У третьому розділі описана розробка IoT системи моніторингу, включаючи методику визначення концентрації газу та налаштування пристрою для роботи з сервісом "ThingSpeak".

Четвертий розділ присвячено результатам дослідження. Проведено вимірювання якості повітря, аналізовано отримані дані з врахуванням концентрації CO, температури та вологості. Обчислено середньодобову точність вимірювань, що дозволяє зробити висновки щодо ефективності застосування дистанційної системи моніторингу.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ	9
1.1 Параметри мікроклімату у приміщенні.....	9
1.1.1 Вимірювання температури.....	9
1.1.2 Вимірювання вологості повітря.....	12
1.1.3 Датчики концентрації CO.....	13
1.2 Температура та вологість повітря.....	14
1.3 Концентрація вуглекислого газу.....	17
1.4 Сервіс «ThingSpeak».....	18
РОЗДІЛ 2 РОЗРОБКА МОДЕЛІ ТА ВИБІР ОСНОВНИХ КОМПОНЕНТІВ ВИЗНАЧЕННЯ ПАРАМЕТРІВ МІКРОКЛІМАТУ	20
2.1 Розроблення структурної схеми та алгоритму роботи.....	20
2.2 Вибір компонентів для роботи пристрою.....	23
2.3 Розробка електричної принципової схеми.....	31
2.4 Методика розрахунку середньодобової точності вимірювань.....	33
РОЗДІЛ 3 РОЗРОБКА ІОТ СИСТЕМИ МОНІТОРИНГУ	34
3.1 Методика визначення концентрації газу в «ррт».....	34
3.2 Створення веб сервісу.....	38
3.3 Налаштування приладу для роботи з сервісом «ThingsSpeak».....	46
РОЗДІЛ 4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ.....	51
4.1 Проведення вимірювання якості повітря	51
4.2 Обчислення середньодобової точності вимірювань	53
ВИСНОВКИ	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	57
ДОДАТОК А.....	59
ДОДАТОК Б.....	78

ВСТУП

Сучасний розвиток технологій не лише полегшує наше повсякденне життя, але й відкриває нові можливості для вирішення важливих проблем, пов'язаних із забезпеченням комфорту та безпеки в приміщеннях. Однією з таких актуальних проблем є контроль якості повітря, яке ми вдихаємо, особливо в умовах зростаючого рівня загроз від забруднень та інших агентів, що можуть впливати на наше здоров'я.

Ця робота спрямована на дослідження дистанційної системи моніторингу якості повітря у приміщенні за допомогою технології Інтернету речей (IoT). Метою дослідження є визначення середньодобової точності вимірів якості повітря, яка здійснюється за допомогою датчиків вологості, температури та концентрації вуглеводневих газів. Використання технології IoT в цьому контексті дозволяє здійснювати дистанційний добовий контроль, що важливо для постійного відстеження та аналізу стану повітря у реальному часі.

Для здійснення контролю якості повітря використовуються датчики, які фіксують параметри, такі як вологість, температура та концентрація вуглеводневих газів. Важливим етапом є розробка IoT системи моніторингу, яка дозволяє збирати, аналізувати та відображати дані вимірювань у хмарному сервісі, забезпечуючи доступність цієї інформації в будь-який час та з будь-якого місця.

Основними завданнями дослідження є розробка проекту IoT системи моніторингу якості повітря та реалізація методики розрахунку середньодобової точності вимірів параметрів повітря. В даній роботі також буде проведено дослідження зміни середньодобової точності вимірювань вологості, температури та концентрації вуглеводневих газів, що дозволить визначити ефективність запропонованої системи моніторингу.

Це дослідження є актуальним та важливим в контексті забезпечення здоров'я та комфорту в житлових та робочих приміщеннях, а також вносить

свій внесок у розвиток технологій для сталого та екологічно чистого середовища проживання.

Метою дослідження є визначення ефективності використання дистанційної системи моніторингу якості повітря у приміщенні з використанням технології IoT.

Об'єктом дослідження є процес вимірювання якості повітря за допомогою цифрових та аналогових датчиків.

Предметом дослідження є IoT-система моніторингу якості повітря.

Наукова новизна дослідження полягає в розробці методики розрахунку середньодобової точності вимірювань параметрів повітря без використання еталоннів, що дозволяє проводити оперативну діагностику системи моніторингу та забезпечити надійні дані для подальших досліджень у галузі екології та здоров'я.

Практичне значення одержаних результатів полягає в розробці та впровадженні IoT-системи моніторингу, яка забезпечить ефективний контроль за якістю повітря у приміщеннях з можливістю дистанційного статистичного аналізу даних. Показано, що середньодобова похибка вимірів контрольованих параметрів не перевищує значення інструментальної похибки вимірів датчиків з довірчою ймовірністю 0,997.

Результати магістерської роботи доповідались та обговорювалися на VI всеукраїнській науково-технічній конференції «Комп'ютерні технології: інновації, проблеми, рішення» у м. Житомир 29-30 листопада 2023 р.

Ключові слова: IoT, моніторинг якості повітря, добовий контроль.

РОЗДІЛ 1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1 Параметри мікроклімату у приміщенні

Сучасний ритм життя призводить до того, що значна частина населення проводить більшість свого часу у закритих приміщеннях, таких як офіси, школи, торгові центри та житлові приміщення. Важливість забезпечення високої якості повітря в цих приміщеннях стає критичною для забезпечення комфорту та підтримання здоров'я людей.

Згідно з даними Всесвітньої організації охорони здоров'я (ВООЗ), погіршення якості внутрішнього повітря може призвести до ряду проблем здоров'я, включаючи захворювання дихальних шляхів, алергії та інші хронічні захворювання. Однак багато приміщень не мають ефективних систем моніторингу якості повітря, що може призвести до недоліків у вчасному виявленні та вирішенні проблем [1].

Основними показниками клімату виробничого приміщення є контрольовані параметри: температура, рівень вологості повітря та вуглекислий газ. Ці показники можуть змінюватися протягом усього робочого дня або зміни.

Відповідно до прийнятих санітарно-гігієнічних норм параметри мікроклімату повинні дорівнювати наступним значенням:

1) температура:

- для житлових приміщень від +22 °C до +32 °C;
- для виробничих приміщень від +18 °C до +25 °C;
- для дитячих кімнат від +22 °C до +25 °C;

2) вологість повітря від 40% до 60%;

3) концентрація вуглекислого газу від 0,03% до 0,04% (від 400 до 800 ppm);

1.1.1 Вимірювання температури

Існують два основні методи вимірювання температури: контактні та безконтактні.

Контактні методи ґрунтуються на прямому контакті вимірювального перетворювача температури з об'єктом, що досліджується. Це призводить до досягнення стану теплової рівноваги між перетворювачем і об'єктом, але при цьому температурне поле об'єкта може спотворитися внаслідок введення термоприймача. Такий метод часто призводить до розбіжностей між температурою перетворювача і реальною температурою об'єкта. Верхня межа вимірювання температури обмежена властивостями матеріалів, включаючи виготовлені температурні датчики. Крім того, в разі недоступних об'єктів або тих, які обертаються з великою швидкістю, контактний метод не завжди ефективний.

Безконтактний метод ґрунтується на сприйнятті теплової енергії, яка передається через випромінювання і сприймається на певній відстані від об'єкта, що досліджується. Цей метод менше чутливий, але точність вимірювання температури значно залежить від умов градування під час експлуатації, інакше можуть виникати значні похибки.

Пристрій, який використовується для вимірювання температури, перетворюючи її значення у вигляді сигналу або показань, називається термометром. За принципом дії всі термометри можна розділити на різні групи, призначені для вимірювання температур у різних інтервалах:

- Термометри розширення охоплюють діапазон від -260 до $+700$ °C і ґрунтуються на зміні обсягів рідин або твердих тіл при зміні температури.
- Манометричні термометри, призначені для вимірювання температур у діапазоні від -200 до $+600$ °C, використовують тиск рідини, пари або газу в закритому об'ємі, який змінюється залежно від температури.
- Термометри електричного опору, стандартний діапазон яких від -270 до $+750$ °C, перетворюють зміну температури на зміну електричного опору провідників або напівпровідників.
- Термоелектричні термометри, також відомі як пірометри, мають стандартний діапазон від -50 до $+1800$ °C і базуються на залежності електрорушійної сили від температури спаю різних провідників.

Термоелектричний метод вимірювання температур базується на жорсткій залежності термоелектрорушійної сили термоелектричного термометра від температури. Цей метод широко використовується для вимірювання температур до 2500 °С у різних областях техніки та наукових досліджень. Хоча їх можна застосовувати для вимірювання температур від -200 °С, термоелектричні термометри менше популярні в області низьких температур порівняно з термометрами опору.

Термоелектричні термометри мають деякі переваги, такі як:

- Досить високий рівень точності.
- Можливість централізованого контролю температури шляхом підключення кількох термоелектричних термометрів через перемикач до одного вимірювального пристрою.
- Можливість автоматичного запису вимірюваної температури за допомогою самописного пристрою.
- Можливість роздільного градуювання вимірювального пристрою та термоелектричного термометра.

Термометри опору широко використовуються для вимірювання температур у діапазоні від -260 до 750 °С, а в окремих випадках їх можна використовувати для вимірювання температур до 1000 °С. Принцип роботи цих термометрів ґрунтується на властивості матеріалу змінювати свій електричний опір при зміні температури. Під час вимірювання температури термометр опору поглиблюють у середовище, температуру якого потрібно визначити. Знаючи залежність опору термометра від температури, можна визначити температуру навколишнього середовища.

Важливо враховувати, що чутливий елемент більшості термометрів опору має довжину кількох сантиметрів, тому при наявності температурних градієнтів у середовищі термометр опору вимірює середню температуру тих шарів, де знаходиться його чутливий елемент.

Останні дослідження вказують на те, що деякі напівпровідники можуть служити матеріалом для виготовлення термометрів опору. У порівнянні з

металевими термометрами опору, металеві мають позитивний температурний коефіцієнт електричного опору, тоді як у напівпровідників цей коефіцієнт різко збільшується зі збільшенням температури. Термометри опору з чистих металів виготовляють у вигляді обмоток з тонкого дроту на спеціальному каркасі з ізоляційного матеріалу, що називається чутливим елементом. Для запобігання механічним пошкодженням і впливу середовища, температура якого вимірюється, чутливий елемент поміщають у захисну гільзу.

Переваги металевих термометрів опору включають високий рівень точності вимірювання температури, можливість випуску вимірювальних приладів із стандартним градуюванням для різних температурних інтервалів, можливість централізованого контролю температури за допомогою кількох взаємозамінних термометрів опору та їх використання з інформаційно-обчислювальними машинами. Напівпровідникові термометри опору можуть бути використані для вимірювання температур від 1,3 до 400 К, але їх використання в порівнянні з металевими термометрами обмежене, оскільки вони вимагають індивідуального градуювання. У лабораторних умовах вони використовуються з потенціометрами, мостами та іншими засобами вимірювання для точних результатів. У промислових умовах термометри опору застосовуються з логометрами, врівноваженими мостами та компенсаційними приладами, при цьому слід пам'ятати, що шкала цих приладів дійсна лише для конкретного градуювання термометра опору та опору проводів, що його з'єднують.

1.1.2 Вимірювання вологості повітря

Розглянемо різновиди датчиків вимірювання вологості повітря та їх принцип дії більш детально. Ці датчики можна класифікувати за принципом дії на кілька типів:

- Волосяні датчики: Вони використовують властивість волосся змінювати довжину при зміні вологості.
- Ємнісні датчики: Зміна вологості призводить до зміни електричної ємності конденсатора з гігроскопічним діелектриком.

- Резистивні датчики: Опір провідника, на поверхню якого нанесено гігроскопічний матеріал, змінюється залежно від вологості.
- П'єзосорбційні датчики: Волога, поглиблена гігроскопічним покриттям, змінює частоту коливань п'єзокристалу.
- Оптичний абсорбційний датчик: Вимірюється частка поглиненої енергії світла у смугах поглинання парами води електромагнітного випромінювання.

У ємнісних датчиках вологості вимірюваною фізичною величиною є ємність конденсатора. Для їхньої конструкції на підкладку з кварцу наноситься алюмінієвий шар, на якому утворюється тонка плівка окису Al_2O_3 . Другий електрод з металу, що пропускає пари води, наноситься на окислену поверхню, утворюючи зовнішній пористий електрод конденсатора.

Резистивні датчики вологості мають конструкцію у вигляді меандру з двох електродів, на поверхню якого наноситься тонкий шар гігроскопічного діелектрика. Зміна вологості впливає на опір проміжків між електродами, що дозволяє вимірювати вологість.

Нещодавно з'явилися гігрометри, які використовують фізичний закон поглинання електромагнітного випромінювання - закон Ламберта-Бугера-Бера. Оптичні датчики мають певні переваги, такі як швидке встановлення аналітичного сигналу, але вони також піддаються впливу компонентів, які заважають, таких як молекули газів у різних областях спектра.

Зокрема, інфрачервоні та ультрафіолетові оптичні вологоміри виявляються ефективними, хоча вони можуть піддаватися впливу молекул різних газів. При цьому ультрафіолетові датчики вологості виявляються більш ефективними через відсутність впливу вакууму на їхню роботу.

1.1.3 Датчики концентрації CO

Датчики концентрації CO_2 використовуються для вимірювання вмісту діоксиду вуглецю в повітрі, що має важливе значення для забезпечення здоров'я та комфорту в приміщеннях. Ці датчики використовують

недисперсійний інфрачервоний аналізатор, який є технологічно високопродуктивним та довговічним пристроєм.

Принцип роботи цих датчиків полягає в вимірюванні ступеня ослаблення інфрачервоного випромінювання при збільшенні концентрації CO₂ у повітрі. За такого підходу вимірювальний пристрій може точно визначити вміст CO₂, що дозволяє контролювати рівні газу в приміщенні. Ці датчики зазвичай мають широкий інтервал вимірювання, наприклад, від 0 до 2000 часток на мільйон (ppm) CO₂, що дозволяє їм ефективно використовуватися в різних умовах.

Однією з ключових переваг таких датчиків є їхній вбудований мікропроцесор, який керує процесом вимірювань. Це забезпечує високу точність та надійність результатів. Крім того, датчики самокалібруються, періодично отримуючи доступ до свіжого повітря з низьким вмістом CO₂, що дозволяє підтримувати стабільність їхньої роботи.

Застосування таких датчиків важливо в приміщеннях, де контроль концентрації CO₂ може впливати на якість повітря та комфорт користувачів. Наприклад, це може бути використано в офісах, школах, лабораторіях або будь-яких інших закритих приміщеннях для забезпечення оптимальних умов для проживання та праці.

1.2 Температура та вологість повітря

Для забезпечення комфортних умов для людини важливо утримувати температуру в обмеженому діапазоні. Для житлових приміщень рекомендований температурний діапазон від +22 °C до +32 °C, а для виробничих приміщень - від +18 °C до +27 °C. Виходження за межі цих рамок може впливати на працездатність та загрожувати захворюваннями. Дитяча кімната повинна мати температуру не нижче +22 °C і не вище +24 °C.

Вологість в повітрі також важлива для здоров'я. Правильний рівень вологості може позитивно впливати на шкіру, дихальну та нервову систему. Оптимальний діапазон вологості повітря для здоров'я людини визначено на

рівні від 35% до 50%, як підвищено-оптимальні умови мікроклімату. Діапазон від 25% до 60% вважається оптимальним, а від 25% до 70% - допустимим.

Контроль параметрів повітря, таких як температура та вологість, є важливим для забезпечення комфорту та здоров'я. Для підтримання оптимальних умов можуть застосовуватися різні заходи, такі як використання зволожувачів, відповідне опалення приміщень, або регулярне вологе прибирання. Слід дотримуватися встановлених норм та вживати заходів для забезпечення необхідного рівня вологості в повітрі для забезпечення оптимальних умов для здоров'я.

Для досягнення оптимального рівня вологості в приміщеннях важливо враховувати різні джерела надходження та втрати вологи в різних зонах житлового простору. У зонах з високим вологовиділенням, таких як кухня та ванна, рекомендований рівень відносної вологості від 40% до 45%. Для зон із недостатнім рівнем вологовиділення, наприклад, кабінети, спальні та кімнати з електричною технікою, оптимальний діапазон відносної вологості становить від 40% до 60%. У дитячих кімнатах, де потрібно більше вологе повітря, допускається рівень відносної вологості до 55%. Ці рекомендації дозволяють забезпечити комфортні умови в кожній зоні житлового простору відповідно до їхніх специфічних потреб.

За для підтримки комфортного самопочуття треба підтримувати параметри повітря приміщення у діапазонах, що відповідають нормативам співвідношення температура та рівень вологості повітря у приміщенні. У таблиці 1.1 наведені відповідні температурні параметри клімату. Рівень вологості, перевищуючи 40%, розглядається як найбільш оптимальний та комфортний для загального самопочуття людини. У виробничому середовищі мікроклімат має значущий вплив на здоров'я працівників та їхню працездатність протягом робочого дня. Температура, відносна вологість повітря, швидкість руху повітря та теплове випромінювання від джерел тепла є ключовими показниками мікроклімату в житлових та виробничих приміщеннях.

Таблиця 1.1 – Температурні параметри для житолів приміщень

Тип зони	Умови мікроклімату	Результуюча температура	
		Діапазон (холодний період)	Діапазон (теплий період)
Спальна кімната	Підвищені	22,0±1,0	24,5±1,0
	Оптимальні	22,0±2,0	24,5±1,5
	Допустимі	22,0±3,0	24,5±2,5
Кухня	Підвищені	19,5±1,5	-
	Оптимальні	19,5±3,5	-
	Допустимі	19,5±4,0	-
Ванна кімната	Підвищені	25,0±0,5	-
	Оптимальні	25,0±1,5	-
	Допустимі	25,0±2,0	-

Під час трудової діяльності людина непрямо взаємодіє з виробничим середовищем, і оптимальні умови мікроклімату сприяють підтримці постійної температури тіла завдяки терморегуляції приблизно 36,6 °С. Вологість повітря впливає на самопочуття та працездатність, зменшуючи віддачу тепла при високій вологості та покращуючи тепловіддачу при низькій. У виробничому середовищі реалізуються заходи, такі як вентиляція та кондиціонування, для забезпечення допустимих параметрів мікроклімату відповідно до санітарних норм.

Санітарно-гігієнічні норми визначають оптимальний діапазон відносної вологості від 40% до 60%, з допустимим значенням не більше 75%. Верхню межу терморегуляції при спокої вважають 30–31 °С при відносній вологості 85% або 40 °С при відносній вологості 30%. При важкій фізичній праці ця межа нижча, і комфортні умови зберігаються при температурі повітря від +14 °С до +20 °С.

Комфортне самопочуття працюючого забезпечується відповідним співвідношенням температури, відносної вологості і швидкості руху повітря.

Параметри мікроклімату виробничих приміщень нормуються ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [2].

1.3 Концентрація вуглекислого газу

Рівень вуглекислого газу (CO_2) має важливий вплив на самопочуття та продуктивність людей під час виконання роботи. Для комфортного існування людина повинна дихати повітрям, що зазвичай містить 21,5% кисню і 0,03–0,04% вуглекислого газу. У закритих приміщеннях концентрація цих компонентів може змінюватися через метаболічні процеси в організмі людини, де концентрація кисню зменшується, а вуглекислого газу зростає.

В умовах порушення вентиляції та забезпечення свіжим повітрям спостерігається істотне збільшення концентрації вуглекислого газу. Отруєння вуглекислим газом може пройти непомітно, оскільки він не викликає відчуттів у дихальних органах чи зорі. Концентрація вуглекислого газу вище 1% є смертельно небезпечною для дихання. Індикатором якості повітря вважається рівень вуглекислого газу, і при його концентрації понад 0,1% (1000 ppm) може виникнути відчуття задухи, слабкість, головний біль, підвищення тиску та зниження концентрації уваги.

Тривале вплив на невеликі концентрації вуглекислого газу може призводити до хронічних отруєнь, які проявляються головними болями, легкою стомлюваністю, слабкістю та погіршенням пам'яті. Протягом тривалого перебування в приміщеннях з високим рівнем вуглекислого газу можливі зміни в системах кровоносної, центральної нервової та дихальної систем, а також ускладнення розумової діяльності та порушення сприйняття.

Кількість вуглекислого газу в приміщенні напряму залежить від життєважливих процесів людини. Надмірна концентрація вуглекислого газу в приміщенні, де працює людина, може негативно впливати на її самопочуття. Тому важливо вести моніторинг рівня вуглекислого газу (CO_2).

Таблиця 1.2 – Інтенсивність емісії CO₂

CO ₂ /год	Діяльність
18	Стан спокійного неспання
24	Робота за комп'ютером
30	Ходіння
36	Легке фізичне навантаження
32-34	Робота по дому

Для забезпечення комфортного стану і продуктивності людини важливо відслідковувати не лише концентрацію вуглекислого газу, але й інші параметри мікроклімату. Найкращим підходом є аналіз цих параметрів в комплексі, щоб приймати рішення на основі повного обсягу даних.

1.4 Сервіс «ThingSpeak»

ThingSpeak – це платформа Інтернету речей (IoT) в хмарі, яка надає можливість збирати, відображати та аналізувати поточні дані. Вона дозволяє передавати дані до ThingSpeak з різних пристроїв, конфігурувати їх відображення в реальному часі та відсилати повідомлення через Twitter та Twilio [3].

За допомогою ThingSpeak користувачі можуть відправляти дані з своїх IoT-пристроїв до хмарного сервісу, зберігати їх, візуалізувати та аналізувати, а також взаємодіяти з іншими веб-сервісами і соціальними мережами. Платформа також має вбудовані інструменти аналізу даних за допомогою мови програмування MATLAB.

ThingSpeak дозволяє інженерам та розробникам створювати та вдосконалювати IoT-системи, не заморочуючись на складних налаштуваннях серверів чи розробці власних веб-додатків.

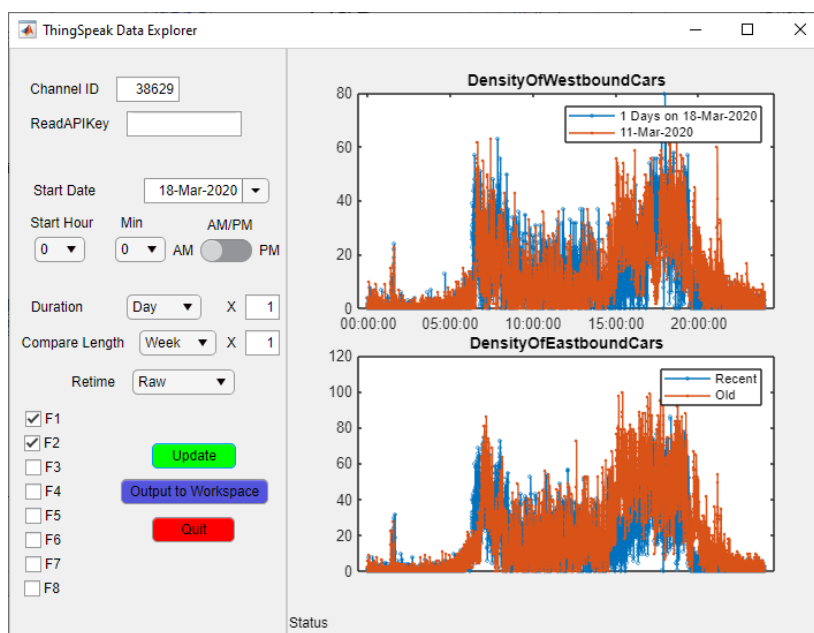


Рисунок 1.1 – Візуалізація даних на ThingSpeak

Дії, спрямовані на основі зібраних даних, можуть бути простими, такими як автоматичне створення твіту при перевищенні температури понад 30 градусів, або складними, наприклад, запуск насосу, коли рівень води в баці впаде нижче встановленого порогу. ThingSpeak надає можливість віддалено керувати виконавчими пристроями, такими як електронні дверні замки, за допомогою вбудованої програми «TalkBack».

РОЗДІЛ 2 РОЗРОБКА МОДЕЛІ ТА ВИБІР ОСНОВНИХ КОМПОНЕНТІВ ВИЗНАЧЕННЯ ПАРАМЕТРІВ МІКРОКЛІМАТУ

2.1 Розроблення структурної схеми та алгоритму роботи

Мікроконтролер виступає в ролі основного пристрою управління в системі. З метою спрощення конструкції, ми використовуємо один датчик, який може вимірювати два параметри навколишнього середовища одночасно: температуру та вологість повітря. Цей датчик працює через інтерфейс I2C. Для вимірювання концентрації вуглекислого газу ми використовуємо датчик CO₂, який взаємодіє через інтерфейс UART.

Пристрій розроблено з метою виконання двох основних завдань: контролю якості повітря та передачі вимірянних даних на платформу ThingSpeak. Функція контролю якості повітря включає в себе визначення концентрації конкретного газу, вимірювання температури та вологості.

На рисунку 2.1 представлена структурна схема бездротового пристрою для контролю якості повітря, що складається з декількох блоків.

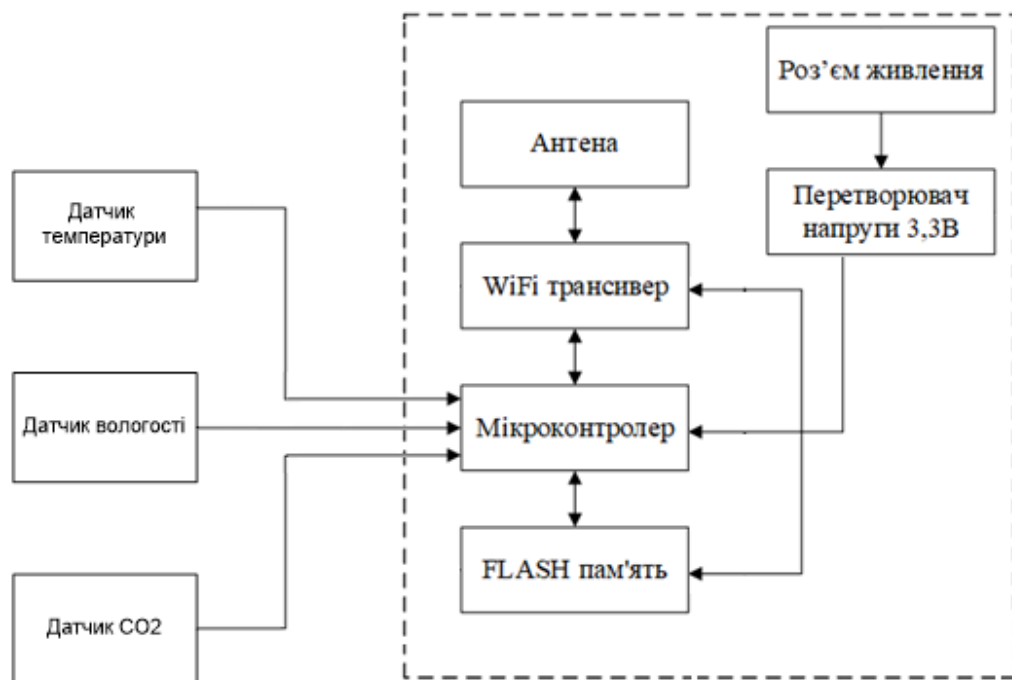


Рисунок 2.1 – Структурна схема пристрою контролю якості повітря

Згідно з рисунком 2.1, антена, трансивер та мікроконтролер розташовані в індивідуальному блоку. Це може бути пояснено тим, що вбудований модуль

вже включає в себе як мікроконтролер, так і WiFi-трансмітер на єдиній кристалічній платформі. На ринку представлені готові рішення у формі модулів, які враховують всі необхідні компоненти, включаючи антену. Саме такий модуль було використано при створенні пристрою. Незважаючи на використання готового рішення, цей модуль вимагає певної обов'язки радіоелектронних компонентів.

Для роботи пристрою який дозволяє забезпечити автономну роботу та віддалений моніторинг стану повітря в реальному часі, необхідно побудувати алгоритм, який буде містити такий розпорядок дій:

1. Перетворення температури в електричну величину:

- Зчитування значення температури за допомогою вбудованого датчика.
- Перетворення отриманого значення температури у величину електричного сигналу.

2. Перетворення вологості в електричну величину:

- Зчитування значення вологості за допомогою вбудованого датчика.
- Перетворення отриманого значення вологості у величину електричного сигналу.

3. Перетворення концентрації вуглекислого газу в електричну величину:

- Зчитування значення концентрації вуглекислого газу за допомогою вбудованого датчика.
- Перетворення отриманого значення концентрації вуглекислого газу у величину електричного сигналу.

4. Кодування отриманих результатів:

- Опрацювання та кодування вимірянних значень температури, вологості та концентрації вуглекислого газу для передачі по мережі.

5. Передача даних до мережі Інтернет:

- Відправлення закодованих даних на визначену Інтернет-платформу, таку як ThingSpeak.
- Забезпечення віддаленого моніторингу та доступу до виміряних параметрів через мережу Інтернет.

Таким чином спрощений алгоритм обробки сигналів матиме вигляд, наведений на рисунку 2.2

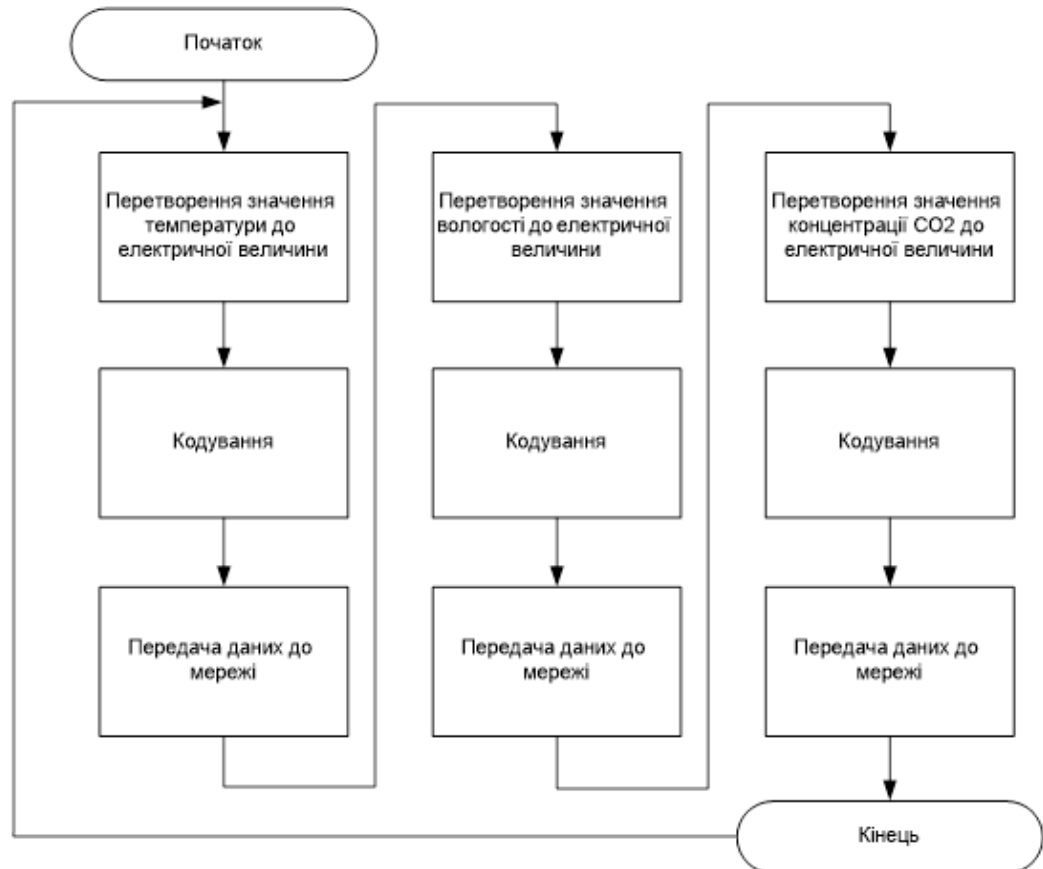


Рисунок 2.2 – Алгоритм обробки сигналів

Алгоритм нашого дослідження зображений на рисунку 2.3

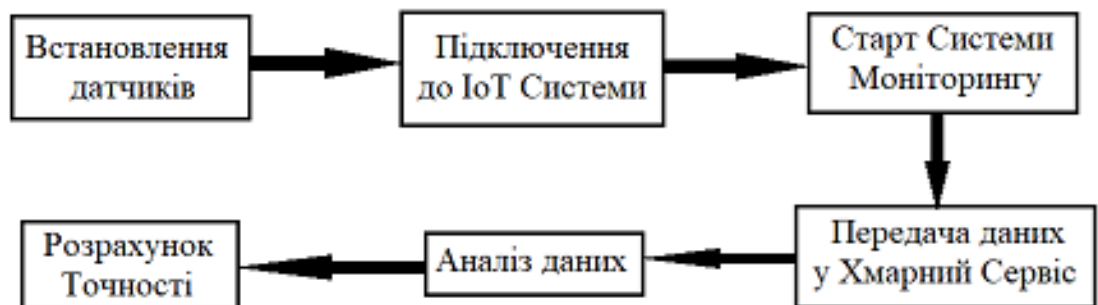


Рис. 2.3 – Алгоритм дослідження

Даний алгоритм включає основні кроки для проведення дослідження якості повітря з використанням дистанційної системи моніторингу Інтернету речей.

1. Встановлення датчиків вологості, температури та концентрації вуглеводневих газів у приміщенні.
2. Підключення до IoT системи датчиків пристрою для автоматичного передавання даних.
3. Старт Системи Моніторингу та здійснення вимірювання протягом визначеного періоду.
4. Передача Даних у Хмарний Сервіс забезпечить передачу даних у хмарний сервіс для зберігання та аналізу.
5. Аналіз Даних які отримали із замірів концентрації CO, температури та вологості.
6. Розрахунок Точності використовуючи розроблену методику для розрахунку середньодобової точності вимірювань.

2.2 Вибір компонентів для роботи пристрою

Для розробки бездротового пристрою був обраний модуль «ESP8266 ESP-12E». На рисунку 2.3 зображено його зовнішній вигляд. Модуль включає в себе 32-розрядний мікроконтролер Tensilica L106 з низьким рівнем енергоспоживання, який працює на частоті 80 МГц і підтримує режим 160 МГц. Важливо зазначити, що мікроконтролер не має вбудованої енергонезалежної пам'яті на кристалі.

Виконання програми здійснюється зовнішньою SPI Flash ПЗУ. Під час роботи програма динамічно завантажує необхідні частини коду в кеш інструкцій з цього зовнішнього джерела. Механізм завантаження виконується апаратно, і для програміста цей процес є прозорим.

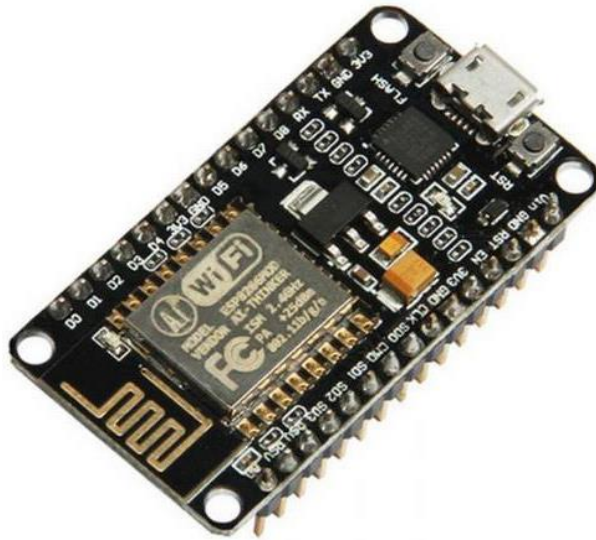


Рисунок 2.3 – Зовнішній вигляд «ESP8266 ESP-12E»

Підтримується до 16 МБ зовнішньої пам'яті програм. Залежно від версії модуля розмір ПЗУ може змінюватись. Виробник у документах не вказує точну кількість ОЗП на кристалі, а лише приблизну оцінку тієї кількості ОЗП, що залишиться користувачеві після лінкування API бібліотек – близько 50 кБ [4].

Технічні характеристики ESP8266:

- напруга живлення 3.0..3.6 В. Рекомендована 3.3 В.;
- струм: до 215мА в залежності від режиму роботи;
- вбудований 10-розрядний АЦП;
- вбудований стек протоколу TCP/IP;
- зібраний із синтезатором частоти, регулятором напруги та компонентами керування живленням, у режимі 802.11b. Вихідна потужність становить +25 дБм;
- внутрішня антена на друкованій платі;
- WiFi @ 2.4 GHz, підтримує режими захисту WPA/WPA2;
- підтримує віддалене оновлення та оновлення через хмару AT OTA;

- режими роботи: Клієнт (STA), Точка доступу (AP), Клієнт + Точка доступу (STA + AP);
- HSPI, UART, I2C, I2S, IR дистанційне керування, PWM, GPIO;
- у режимі глибокого сну споживання становить 10 мкА, у режимі вимкнення - менше 5 мкА;
- діапазон робочих температур: -40 °C - 125 °C.

Вибір датчику вологості та температури.

Для вимірювання температури та вологості можна використовувати датчики такого типу, як DHT11/DHT22. Щоб здійснити порівняння цих двох датчиків, надається відповідна інформація у таблиці 2.1.

Таблиця 2.1 – Характеристики датчиків температури та вологості

Параметри	DHT11	DHT22
Напруга живлення, В	3,...,5,5	
Споживаний струм, мА	0,1,...,2,5	0,05,...,2,5
Діапазон виміру вологості, %	20,...,95	0,...,100
Точність виміру вологості, %	5	2,..., 5
Діапазон виміру температури, °C	0,...,+50	- 40,...,+80
Точність виміру температури, °C	±2	±0.5
Частота вимірів	1Гц	0.5Гц
Розрядність	8 бит	16 бит
Вартість,грн	100	260

Порівнюючи обидва датчики, DHT-22 виявляється точнішим, має ширший діапазон вимірювання температури, але також є вищою за вартістю у порівнянні з DHT-11. Згідно з технічним завданням, для наших потреб достатньо використовувати датчик DHT-11. З метою вимірювання температури та вологості в повітрі було обрано модуль датчика «DHT11»,

який встановлений на платі та може бути підключений безпосередньо до Ардуіно, без необхідності використання підтягувального резистора, оскільки він вже вбудований у плату. Вигляд обраного датчика можна побачити на рисунку 2.4.

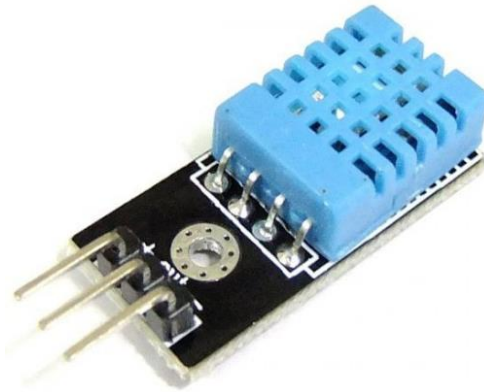


Рисунок 2.4 – Друкована плата з датчиком «DHT-11»

Характеристики датчику DHT-11 [5]:

Відносна вологість:

- Роздільна здатність: 16Bit;
- Періодичність: $\pm 1\%$ відносної вологості;
- Точність: На $25\text{ }^{\circ}\text{C} \pm 5\%$ відносної вологості;
- Взаємозамінність: повністю взаємозамінні;
- Час реакції: $1/e$ (63%) з $25\text{ }^{\circ}\text{C}$ 6с 1 м/с повітря 6с;
- Гістерезис: $<\pm 0,3\%$ Відносної вологості;
- Довгострокова стабільність: $<\pm 0,5\%$ Відносна вологість/в рік.

Температура:

- Роздільна здатність: 16Bit;
- Періодичність: $\pm 0,2\text{ }^{\circ}\text{C}$;
- Діапазон: На $25\text{ }^{\circ}\text{C} \pm 2\text{ }^{\circ}\text{C}$;
- Час реакції: $1/e$ (63%) 10с.

Електричні характеристики:

- Напруга живлення: 3-5 В
- Обумовлена вологість: 20-90% Відносна вологість $\pm 5\%$ (макс.)
- Обумовлена температура: $0-50\text{ }^{\circ}\text{C} \pm 2\%$ (макс.)

- Частота опитування: 1 Гц
- Розміри: 15.5 x 12 x 5.5 мм

Модуль DHT-11 призначений для вимірювання температури і вологості повітря та передачі даних по одному дроту з використанням власного протоколу. Його можна використовувати в пристроях, таких як Arduino, AVR, PIC, ESP і інші. Для роботи з Arduino існує готова бібліотека. Даний датчик температури і вологості DHT11 є найбільш оптимальним вибором для використання в системі, оскільки він має найбільш прийнятне співвідношення технічних характеристик, простоти конструкції і ціни порівняно з його аналогами. На рисунку 2.5 зображено контакти виводу для підключення.

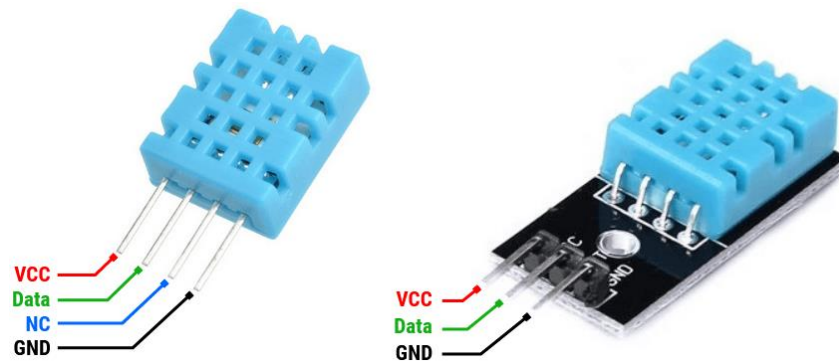


Рисунок 2.5 – Контакти підключення датчику

Вивід "VCC" та "GND" відповідає за живлення датчика. Навіть якщо напруга живлення може коливатися в діапазоні від 3,3 до 5,5 В, рекомендується використовувати напругу живлення 5 В. При використанні джерела живлення 5 В можна тримати датчик на відстані до 20 метрів від джерела живлення. Проте при напрузі живлення 3,3 В довжина кабелю не повинна перевищувати 1 метра, оскільки будь-яке падіння напруги в лінії може призвести до помилок вимірювань.

Вивід "Data" використовується для встановлення зв'язку між датчиком та мікроконтролером. Ніжка "NC" не використовується.

Вибір датчику газу.

Для визначення рівня якості повітря можна використовувати датчики, такі як CCS811, MQ-135 або універсальний датчик широкого спектру газів MQ-2.

Датчик якості повітря в приміщеннях на CCS811 модуль є цифровим газовим датчиком, який визначає широкий спектр загальних летючих органічних сполук (TVOC), включаючи еквівалентні рівні діоксиду вуглецю (CO₂) та оксиду металу (MOX). Перегляньте рисунок 2.6.

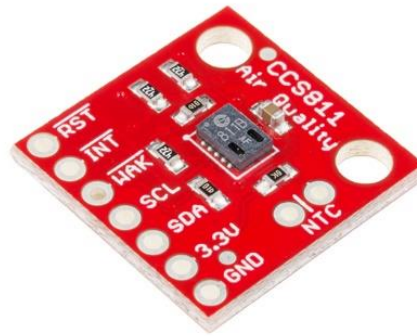


Рисунок 2.6 – Друкована плата CCS811

Характеристики:

- вимірювання загальних летючих органічних сполук (TVOC): від 0 до 1 187 частин на мільярд
- вимірювання CO₂: від 400 до 8 192 частин на мільйон
- кількість режимів роботи: 5
- внутрішня обробка даних: вбудований мікроконтролер
- інтерфейс: I2C
- вимірювання температури: зовнішній термістор NTC

Незважаючи на великий функціонал даного модуля, він виявляється надто високоцінним для наших потреб у проекті.

Модуль датчика якості повітря MQ135 розроблено для визначення концентрації та кількості шкідливих і небезпечних газів у повітрі, таких як NH₃, NO_x, пари алкоголю, бензину, диму, CO₂ і т.д. Він базується на датчику MQ135. На платі модуля передбачено компаратор для встановлення порогового рівня концентрації шкідливих речовин, а цей поріг регулюється за допомогою потенціометра. Зображено на рисунку 2.7.



Рисунок 2.7 – Друкована плата з датчиком «MQ-135»

Характеристики:

- Вихідний сигнал TTL рівня
- Аналоговий вихід датчика 0 - 5В. Чим вище концентрація газів тим вище напруга на виході
- Напруга живлення модуля: 5В постійного струму
- Розміри: 32мм X22мм X27мм

Обрано датчик газу із серії "MQ", а саме "MQ-2". Маючи діапазон чутливості до 10 000 ppm, датчик MQ-2 здатний виявляти різні типи газів, таких як зріджений вуглеводень (LPG), пропан, водень або інші горючі гази, дим, чадний газ. Його вигляд може бути представлений у вигляді рішення, показаного на рисунку 2.8.

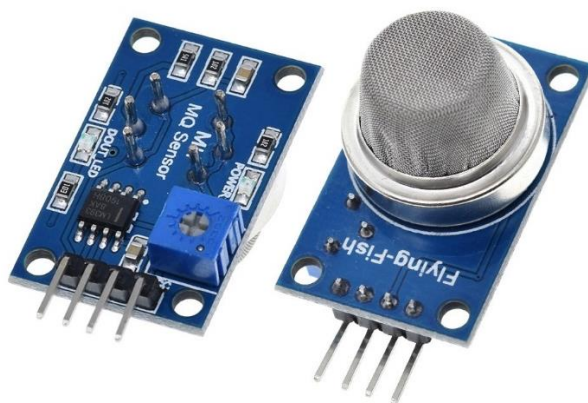


Рисунок 2.8 – Друкована плата з датчиком «MQ-2»

Характеристики датчику MQ-2 [6]:

- Газ, що детектується - Горючий газ, дим;
- діапазон чутливості: 300-10000 ppm;

- R_s (опір чутливого елемента): 1 ... 20 кОм 50ppm толуол;
- газ, до якого нормується датчик: Ізобутан, 1000ppm;
- час відгуку ≤ 10 с;
- чутливість (R в повітрі) / (R в присутності характерного газу) ≥ 5 с;
- R_h (опір нагрівача) $31\Omega \pm 3\Omega$;
- I_h (струм нагрівача) ≤ 180 мА;
- V_h (напруга нагрівача) $5V \pm 0,2$ В;
- P_h (потужність нагрівача) ≤ 900 мВт;
- V_c (напруга схеми) ≤ 24 В;
- стандартні робочі умови Температура: $-10 \sim + 50$ ° С, вологість: $\leq 95\%$, концентрація кисню: 21% (стандартні умови).

Цей датчик володіє високою чутливістю до зрідженого вуглеводневого газу, пропану, водню та може бути використаний для інших горючих газів. Основними перевагами цього датчика є довгий термін служби, низька вартість і проста схема управління. У рисунку 2.9 наведено призначення виходів модулю датчика.

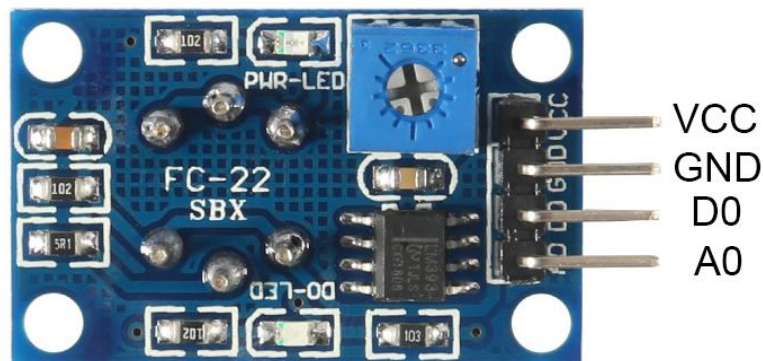


Рисунок 2.9 – Призначення виходів датчику

Виводи "VCC" та "GND" використовуються для подачі живлення на датчик; схема цього модулю передбачає напругу живлення 5В. Вивід "D0" генерує високий логічний рівень, якщо поріг спрацювання датчика перевищений, і цей поріг налаштовується змінним резистором на платі. Вивід "A0" видає постійну напругу різної амплітуди, яка залежить від концентрації газу.

2.3 Розробка електричної принципової схеми

Згідно інформації, наданої для ESP12F [4], напруга живлення мікроконтролера повинна знаходитися в межах 3 – 3,6 В. Згідно з технічним завданням, напруга живлення пристрою встановлена на рівні 5 В. Ця напруга придатна для живлення датчика газу, але виходить за рамки допустимих значень для мікроконтролера. З цією метою на платі ESP8266 встановлено стабілізатор напруги "AMS1117-3.3" у корпусі "SOT-223". Цей стабілізатор стабілізує напругу живлення в межах від 4,75 до 12 В та має максимальний вихідний струм до 1 А [7].

Для живлення датчиків використовується пін VIN, який надає напругу 5В безпосередньо з USB роз'єму.

Узгоджено вихідну напругу датчика з вхідною напругою, яка може бути подана на вхід АЦП мікроконтролера. Максимальна вхідна напруга відповідає напрузі живлення АЦП і становить 1 В [8], тоді як напруга на виході датчика в теорії може досягати 5 В.

Резистори R1 та R2 підібрані так, щоб вихідна напруга датчика була поділена на коефіцієнт "5". Значення цих резисторів мають допуск відхилення в 1%. У відповідності до електричної принципової схеми, використовуючи значення резисторів R1=1 кОм та R2=4 кОм, напруга, яка починається з 5 В, буде зменшена до 1 В.

$$U_{\text{вих}} = U_{\text{вх}} \cdot \frac{R1}{R2 + R1} = 5 \cdot \frac{1\,000}{4\,000 + 1\,000} = 1\text{В}, \quad (2.1)$$

де $U_{\text{вих}}$ – вихідна напруга, В;

$U_{\text{вх}}$ – вхідна напруга, В.

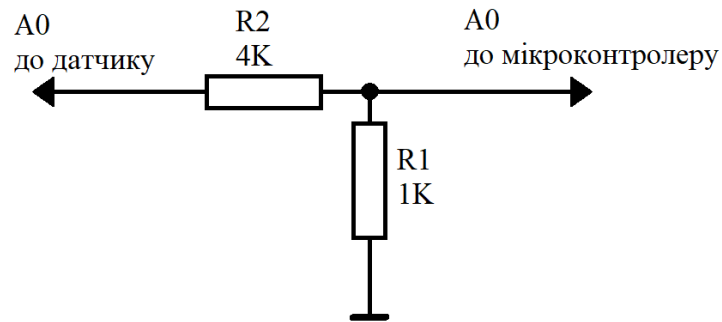


Рисунок 2.10 – Коло узгодження напруги для АЦП

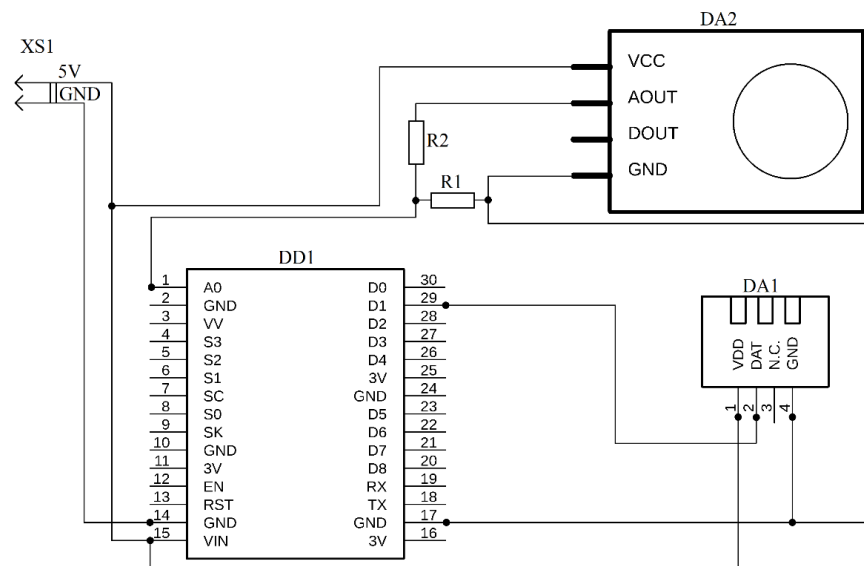


Рисунок 2.11 – Принципова схема приладу вимірювання мікроклімату

Проектована принципова схема розроблена за допомогою програмного забезпечення CadSoft Eagle Schematic. Увесь пристрій живиться від роз'єму Micro-USB. Також можна використовувати будь-який блок живлення, який надає постійну напругу 5 В та силу струму 500 мА.

Датчик MQ-2 підключений аналоговим виходом «AOUT» до входу «A0» на платі ESP. Живлення «VCC» підключене до «VIN» через те, що для коректної роботи датчика необхідно подавати живлення не менше 5 В. Оскільки на платі ESP8266 немає виводу 5 В, ми використовуємо вихід «VIN», який подається безпосередньо від USB-живлення.

Вивід «DAT», розташований на датчику DHT-11, підключається до цифрового входу «D1».

2.4 Методика розрахунку середньодобової точності вимірювань

Розробка методики розрахунку та дослідження зміни середньодобової точності вимірювань вологості, температури та концентрації вуглеводневих газів включає декілька етапів збір даних та вимірювань температури, вологості та концентрації вуглекислих газів,

1. Збір даних, таких як температури, вологості та концентрації вуглекислих газів. Ці заміри будуть відбуватися кожних 10 хв що у сумі за добу ми отримаємо 144 заміри за кожним типом даних.

2. Після отримання даних, проведемо розрахунки середньодобової точності для кожного параметру за допомогою формул:

$$T_{\text{СКВ}} = \sqrt{\frac{\sum_{i=1}^n |T_i - T_{\text{сер.}}|^2}{n}} \quad (2.2)$$

Де $T_{\text{СКВ}}$ – Середньодобова точність температури.

T_i – сума вимірювання значень.

$T_{\text{сер.}}$ – середнє значення вимірювання температури.

n – кількість вимірювань за добу.

$$H_{\text{СКВ}} = \sqrt{\frac{\sum_{i=1}^n |H_i - H_{\text{сер.}}|^2}{n}} \quad (2.3)$$

Де $H_{\text{СКВ}}$ – Середньодобова точність вологості.

H_i – сума вимірювання значень.

$H_{\text{сер.}}$ – середнє значення вимірювання вологості.

n – кількість вимірювань за добу.

$$G_{\text{СКВ}} = \sqrt{\frac{\sum_{i=1}^n |G_i - G_{\text{сер.}}|^2}{n}} \quad (2.4)$$

Де $G_{\text{СКВ}}$ – Середньодобова точність концентрації газів.

G_i – сума вимірювання значень.

$G_{\text{сер.}}$ – середнє значення вимірювання газів.

n – кількість вимірювань за добу.

РОЗДІЛ 3 РОЗРОБКА ІОТ СИСТЕМИ МОНІТОРИНГУ

3.1 Методика визначення концентрації газу в «ppm»

На основі інформації, отриманої з даташиту [7], були виконані розрахунки для визначення концентрації газів CO у частках на мільйон (ppm). Розрахунки проведено для чадного газу.

Датчик серії MQ-2 використовує невеликий нагрівач всередині із електрохімічним датчиком для вимірювання різних видів комбінацій газів. Зміна внутрішнього опору RS датчика газу MQ2 відбувається в залежності від концентрації газу. Схематичне зображення датчика подано на рисунку 4.1. Змінний резистор навантаження RL підключений послідовно з сенсором і забезпечує регулювання чутливості датчика. Опір RL може змінюватися від 2 кОм до 47 кОм [8].

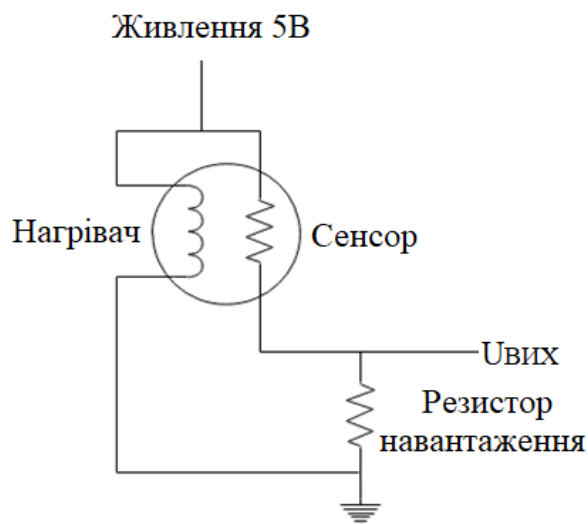


Рисунок 3.1 – Схема датчику MQ-2

Нагрівальний елемент, що використовується всередині датчика MQ-2, вимагає зовнішнього джерела живлення з напругою 5 В та приблизною потужністю 1 Вт [10]. Для визначення внутрішнього опору сенсора RS необхідно враховувати, що вихід модуля датчика фактично є дільником напруги джерела живлення V_{in} , що створюється опором датчика RS та послідовним включенням резистора RL. Оскільки значення резистора

навантаження R_L відоме і постійне, то на основі виміряного значення вихідної напруги датчика V_{out} , можна розрахувати опір датчика R_S :

$$R_S = \left(\frac{V_{in}}{V_{out}} - 1 \right) \cdot R_L. \quad (3.1)$$

На схемі, зображеній на рисунку 3.2, представлено графік, що відображає концентрацію газу в частках на мільйон (ppm) в залежності від відношення опорів R_S/R_0 , де R_0 визначає опір датчика в чистому повітрі. Таким чином, для оцінки концентрації чадного газу необхідно встановити значення опорів R_0 датчика в чистому повітрі.

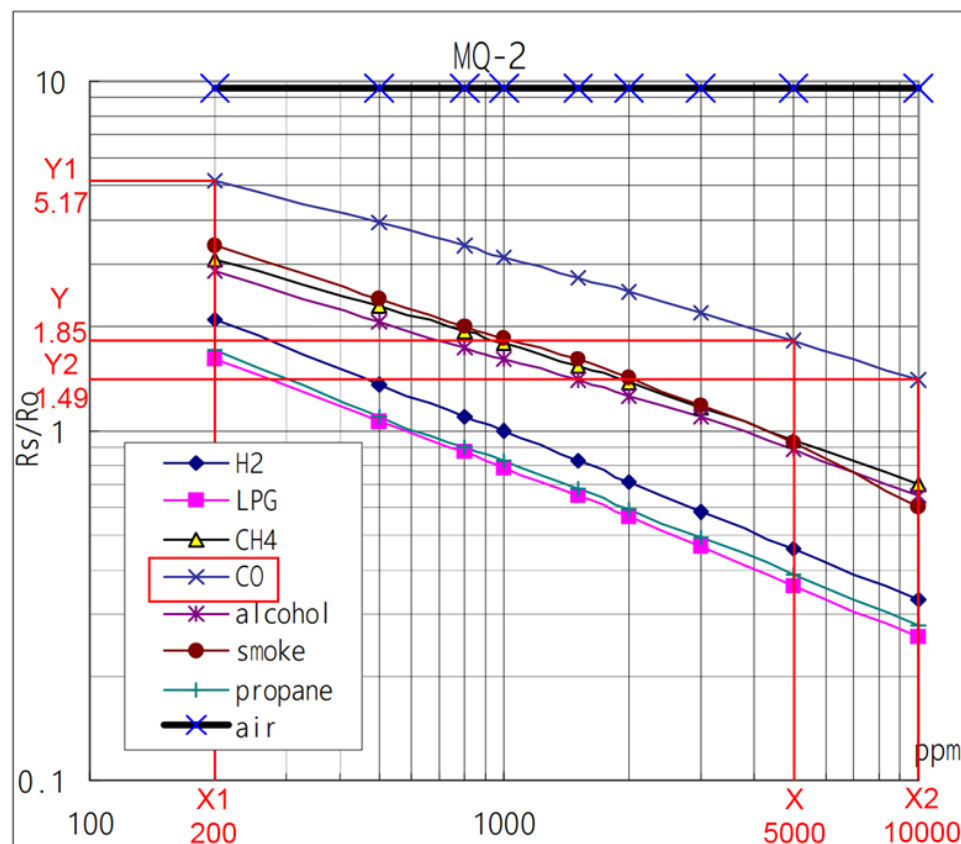


Рисунок 3.2 – Характеристики датчика MQ-2

Для цього, використовуючи мультиметр, проводили вимір опору резистора навантаження на платі датчика MQ-2, який складав $R_L = 2$ кОм. Потім датчик було підключено до джерела живлення $V_{in} = 5$ В і залишено на прогрівання в середовищі з чистим повітрям протягом однієї години. Після цього, через годину було виміряно напругу на виході датчика, яка склала $V_{out} = 0,138$ В. Далі проведено розрахунок опорів сенсора R_S за допомогою виразу (1):

$$RS = \left(\frac{5}{0,138} - 1 \right) \cdot 2 = 70,5 \text{ кОм.}$$

З рисунку 3.2 видно, що у чистому повітрі відношення $\frac{RS}{R0} = 9,8$. Тоді значення опору $R0$ у чистому повітрі:

$$R0 = \frac{RS}{9,8} = \frac{70,5}{9,8} = 7,2 \text{ кОм.}$$

При відомому значенні опору датчика $R0$ у чистому повітрі можна визначити концентрацію газу, яка відповідає розрахованому відношенню виміряного значення внутрішнього опору датчика RS до опору датчика у чистому повітрі $R0$, згідно рисунку 3.2.

Згідно з виразу (1), для розрахунку виміряного значення внутрішнього опору датчика RS , необхідно виміряти поточне значення напруги на виході датчика. Для вимірювання напруги ми підключимо аналоговий вихід датчика газу MQ-2 до входу A0 вбудованого 10-бітного аналого-цифрового перетворювача (АЦП) плати NodeMCUv3 – ESP-12E. Поточне значення вихідної напруги датчика можна розрахувати за допомогою формули:

$$V_{out} = \frac{V_{adc} \cdot V_{ref}}{2^n}, \quad (3.2)$$

де V_{adc} – цифровий код на виході АЦП;

V_{ref} – опорна напруга АЦП, яка дорівнює напрузі V_{in} ;

n – розрядність АЦП.

Графік залежності концентрації газу від відношення $\frac{RS}{R0}$ на рисунку 3.2 подано у логарифмічному масштабі, тобто :

$$\log(y) = m \cdot \log(x) + b, \quad (3.3)$$

де y – відношення опорів $RS/R0$;

x – значення концентрації газу в ppm;

m – нахил лінії;

b – точка, де графік функції перетинає вісь Y .

Нахил лінії розраховується за виразом:

$$m = \frac{\log(y_2) - \log(y_1)}{\log(x_2) - \log(x_1)}, \quad (3.4)$$

де x_1 – X координата точки початку лінії на графіку;

y_1 –Y координата точки початку лінії на графіку;

x_2 –X координата точки кінця лінії на графіку;

y_2 –Y координата точки кінця лінії на графіку.

Згідно рис. 2 початкова точка (x_1, y_1) має координати (200; 5,17), а координати кінцевої точки (x_2, y_2) становлять (10000;1,49). Тоді, згідно (3.4):

$$m = \frac{\log(1,49) - \log(5,17)}{\log(10\,000) - \log(200)} = -0,32.$$

Після розрахунку нахилу лінії m можна визначити з виразу (3.3) значення b :

$$b = \log(y_0) - m \cdot \log(x_0), \quad (3.5)$$

де (x_0, y_0) – координати точки, що відповідає середині графіку.

Координати точки, що відповідає середині графіку (рисунок 3.2) становлять (5000; 1,85). Тоді згідно (3.5):

$$b = \log(1,85) - m \cdot \log(5000) = 1,44.$$

Отже, як передус з виразу (3), $\log(x) = \frac{\log(y)-b}{m}$, а виміряне значення концентрації газу у ppm може бути визначено згідно виразу:

$$x = 10^{\frac{\log(y)-b}{m}} = 10^{\frac{\log(RS/R_0)-b}{m}}. \quad (3.6)$$

Розраховані значення m та b для визначення концентрації чадного газу, занесено до таблиці 3.1

Таблиця 3.1 – Дані для розрахунку концентрації чадного газу

Назва кривої газу	Початкова точка	Середня точка	Кінцева точка	Значення m	Значення b
CO	(200; 5,17)	(5 000; 1,85)	(10 000; 1,49)	-0,32	1,44

3.2 Створення веб сервісу

З метою забезпечення можливості користувача налаштовувати пристрій для контролю якості повітря у реальному часі та швидко змінювати його налаштування, був створений спеціальний веб-сервер. Загалом було розроблено три веб-сторінки.

Головна веб сторінка

Створено основну веб-сторінку з назвою «index.html», яка відображає інформацію про температуру та концентрацію газів. Зовнішній вигляд цієї сторінки представлений на рисунку 3.3.

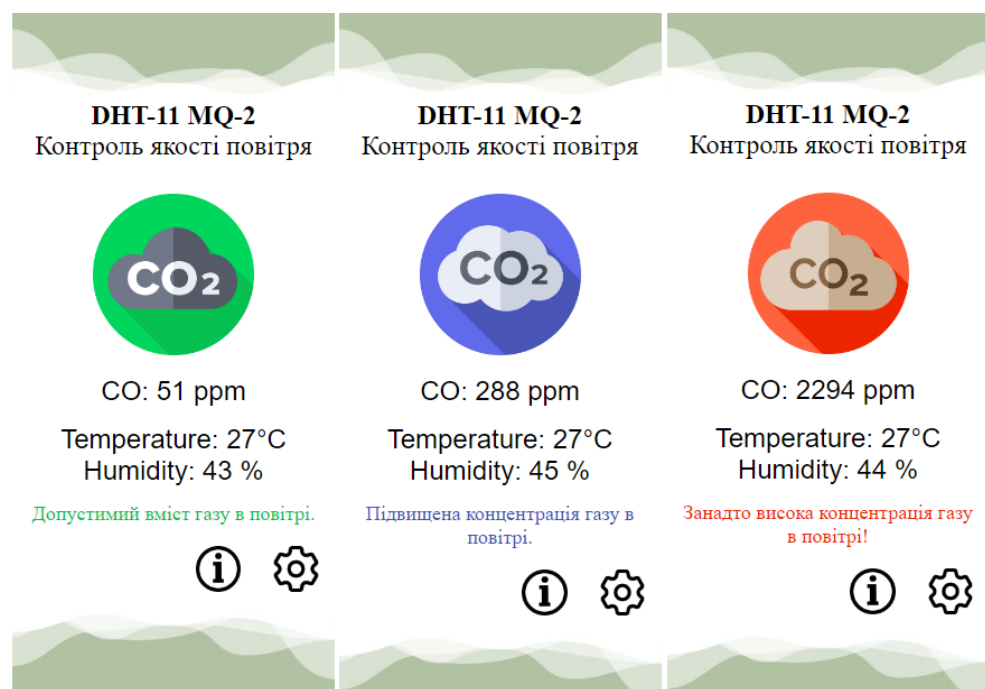


Рисунок 3.3 – Зовнішній вигляд головної сторінки

На головній сторінці відображається інформація щодо концентрації газу, температури та вологості в приміщенні. Вище блоку із даними про концентрацію розташований індикатор, який змінює свій колір в залежності від рівня концентрації газу. Паралельно зі зміною кольору відбувається модифікація повідомлення, що розташоване під блоком інформації про концентрацію. Детальний код програми можна знайти в додатку А.

Будемо докладно аналізувати блок скриптів, який відповідає за управління веб-сторінкою. Після завантаження сторінки викликається функція `init()`, обійти яку реалізовано таким чином:

```
<body onload="init();">
```

Далі були визначені змінні `Socket`, `field1`, `field2`, `field3`, `field4` та `field5`. Змінна `Socket` використовується для викликів вебсокет-серверу, який забезпечує оновлення інформації в режимі реального часу. Змінні `field1`, `field2`, `field3` відповідають за налаштування відображення інформації про концентрацію зрідженого вуглеводного газу, чадного газу та диму відповідно і мають значення «-1», що позначає відсутність участі у розрахунках, які використовуються для виведення повідомлень про концентрацію. Останні дві змінні відповідають за відображення температури та вологості.

На веб-сторінці спочатку вимкнено відображення інформації про концентрацію газів, температури та вологості:

```
#lpg, #co, #smoke, #temperature, #humidity {display: none;}
```

Для активації відображення даних виконується HTTP-запит до сервера, в якому немає передачі будь-яких параметрів:

```
var xhr = new XMLHttpRequest();
xhr.open("POST", '/loadFields');
xhr.send(null);
```

Після завантаження відповіді викликається функція `xhr.onload = function()`. У цій функції відбувається перевірка коду відповіді за допомогою оператора `if`:

```
if (xhr.status == 200) {}
```

Якщо код відповіді відповідає коду успішного відправлення, виконується обробка відповіді від серверу:

```
var fields = xhr.response.split(',', 5);
```

Так було створено змінну `fields`, що містить п'ять елементів `xhr.response`, розділених комою. Кожен елемент відповідає за відображення інформації про газ, температуру та вологість у тому порядку, в якому були ініціалізовані

змінні на початку скрипту. Функція `split()` відповідає за розділення елементів. Далі йде перевірка активності відображення інформації про газ:

```
if (fields[0] !== '-1') {}
```

Якщо отримане значення не дорівнює "-1", це вказує на те, що інформація про газ буде виведена на веб-сторінку, а змінна `field1`, яка відповідає за участь інформації про концентрацію даного газу у формуванні висновку повідомлення, буде врахована у розрахунку:

```
document.getElementById('lpg').style.display = 'block';  
field1 = parseInt(fields[0]);
```

Змінні `field1`, `field2`, `field3`, `field4` та `field5` можуть мати значення від "-1" до "8". Значення "-1" вказує на те, що інформація не відображається, в той час як всі інші значення дозволяють відображення та участь у розрахунку висновку повідомлення.

У функції `init()` реалізовано запит на відкриття вебсокету через порт 81, який передає інформацію про концентрацію:

```
Socket = new WebSocket('ws://' + window.location.hostname +  
' :81/');
```

У випадку отримання повідомлення реалізовано виклик функції `processReceivedCommand(event)`, який знаходиться у `init()`. В яку передається відповідь «event», яка приймається як змінна «value»:

```
function processReceivedCommand(value) {}
```

Всередині функції створено змінну `payload`, яка приймає три значення розділені комою:

```
var payload = value.data.split(',', 3);
```

Далі кожне значення записується у свою змінну:

```
var val1 = parseInt(payload[0]);  
var val2 = parseInt(payload[1]);  
var val3 = parseInt(payload[2]);  
var val4 = parseInt(payload[3]);  
var val5 = parseInt(payload[4]);
```

Змінна `val1` – концентрація зрідженого вуглеводного газу (ppm), `val2` – концентрація чадного газу, `val3` – концентрація диму, `val4` – температура, `val4` – вологість. Далі йде перевірка на дозвіл участі у розрахунку висновку повідомлення:

```
if (field1 == -1) {val1 = -1;}
```

Якщо участь не дозволена (`field1 == -1`), тоді значення концентрації замінюється на "-1". Потім отримана та оброблена інформація відображається на веб-сторінці у відповідному розділі, який має свій унікальний ідентифікатор (`id`):

```
document.getElementById('val1').innerHTML = val1;
```

Потім був впроваджений алгоритм розрахунку висновку щодо концентрації газу. Висновок формується на основі максимального значення концентрації, яке зберігається у змінній `"max"`.

```
var max = Math.max(val1, val2, val3);
```

Наступним кроком реалізовано перевірку за допомогою оператора `if`. Якщо максимальна концентрація газу має значення від 0 до 199 ppm, виводиться повідомлення, що це допустима концентрація:

```
if (max >= 0 && max < 200) {
  document.getElementById('pict').src='normal.png';
  document.getElementById('desc').style.color = '#06BF51';
  document.getElementById('desc').innerHTML = 'Допустимий вміст газу
в повітрі.'; }
```

Також зазначається зміна кольору та повідомлення на зелене. Підвищена концентрація газу визначається від 200 до 299 ppm. Висока небезпечна концентрація газу визначається від 300 до 10 000 ppm. Для налаштування параметрів публікації інформації на сервіс ThingsSpeak була створена веб-сторінка `"settings.html"`, на якій можна задати параметри публікації інформації.

Якщо у вас є конкретні фрази чи речення, які ви хочете перефразувати, або якщо у вас є конкретні аспекти, які вам хотілося б врахувати при перефразуванні, будь ласка, надайте більше контексту, і я буду раджений далі.

ThingSpeak налаштування

☐ Вмикнути надсилення даних

Write API Key:
Введіть ключ доступу для запису.
Максимум 20 символів.

WRITE API KEY

Channel ID:
Введіть номер свого каналу. Максимум 10 цифер.

channel id

Field number (LPG):
Введіть номер поля каналу для LPG.
Допустимі значення від 1 до 8.

field number for LPG

Field number (CO):
Введіть номер поля каналу для CO.
Допустимі значення від 1 до 8.

field number for CO

Field number (Smoke):
Введіть номер поля каналу для Smoke.
Допустимі значення від 1 до 8.

field number for Smoke

Field number (Temperature):
Введіть номер поля каналу для Temperature.
Допустимі значення від 1 до 8.

field number for Temperature

Field number (Humidity):
Введіть номер поля каналу для Humidity.
Допустимі значення від 1 до 8.

field number for Humidity

Інтервал оновлення:
Встановіть інтервал оновлення значень.
Рекомендовані значення від 16, в секундах.

інтервал оновлення

SUBMIT

Інструкцію про початок роботи з сервісом ThingSpeak можна знайти за [посиланням](#).
Для того щоб заборонити відправку даних конкретного поля, вкажіть номер даного поля -1.

Рисунок 3.4 – Веб сторінка налаштування параметрів TS

На цій сторінці можна вказати різні параметри, такі як власний ключ для запису в канал, ідентифікатор каналу, номери полів для інформації про газ, температуру та вологість, а також інтервал запису повідомлень. Код веб-сторінки розташований у додатку А.

Оновлення інформації на веб-сторінці відбувається завдяки HTTP-запиту, що виконується у функції `loadConfig()`. Ця функція викликається негайно після повного завантаження веб-сторінки. Після успішної відповіді отримується JSON-інформація, яку обробляється та відображається у відповідних полях введення. Також здійснюється перевірка дозволу надсилення інформації в канал. Якщо інформація про дозвіл містить значення "1", чекбокс на веб-сторінці встановлюється для позначення дозволу. Інші отримані дані також обробляються та відображаються у відповідних текстових полях сторінки.

```
if (json.state == "1") {
    document.getElementById('check').checked = true;}
```

Відправка дозволу для публікації повідомлень у каналі ThingSpeak відбувається під час натискання чекбоксу. При натисканні викликається функція `saveStatus()`:

```
<input onclick="saveStatus();" type="checkbox" id="check"><label  
for="check">Вмикнути надсилання даних</label>
```

В межах функції проводиться перевірка стану чекбоксу. Якщо він взятий, відправляється значення «1», що вказує на дозвіл, у випадку, якщо чекбокс не взятий – надсилається «0», означаючи заборону відправлення.

```
function saveStatus() {  
    var state;  
    var xhr = new XMLHttpRequest();  
    if (document.getElementById('check').checked) {  
        state = '1';  
    } else {  
        state = '0';  
    }  
    xhr.open("POST", '/saveStatus?state=' + state);  
    xhr.send();  
    xhr.onload = function() {  
        if (xhr.status == 200) {  
            alert('Зміни збережено.');        }  
    }  
}
```

Користувацькі дані, що введені у текстові поля, передаються при натисканні кнопки «Submit». Цей виклик виконується через виклик функції `saveConfig()`. У середині цієї функції створюється змінна `params`, яка містить параметри, отримані з полів для вводу. Після кожного збереження даних виводиться повідомлення «alert», яке повідомляє про успішне збереження налаштувань у випадку, якщо html-запит повернув код «200».

```
function saveConfig() {  
    var xhr = new XMLHttpRequest();  
    var params = 'key=' + key.value + '&id=' + id.value + '&field1='+  
field1.value + '&field2='+ field2.value + '&field3='+ field3.value +  
'&int=' + int.value;  
    xhr.open("POST", '/saveConfig?' + params);
```

```
xhr.send();
xhr.onload = function() {
  if (xhr.status == 200) {
    alert('Зміни збережено.');
```

Для отримання інформації про пристрій, таку як IP- та MAC-адреси, статус вдалості надсилання повідомлень у канал ThingsSpeak та можливість скидання на заводські налаштування або перезавантаження, була створена відповідна веб-сторінка «info.html». Зовнішній вигляд її представлений на рисунку 3.5.

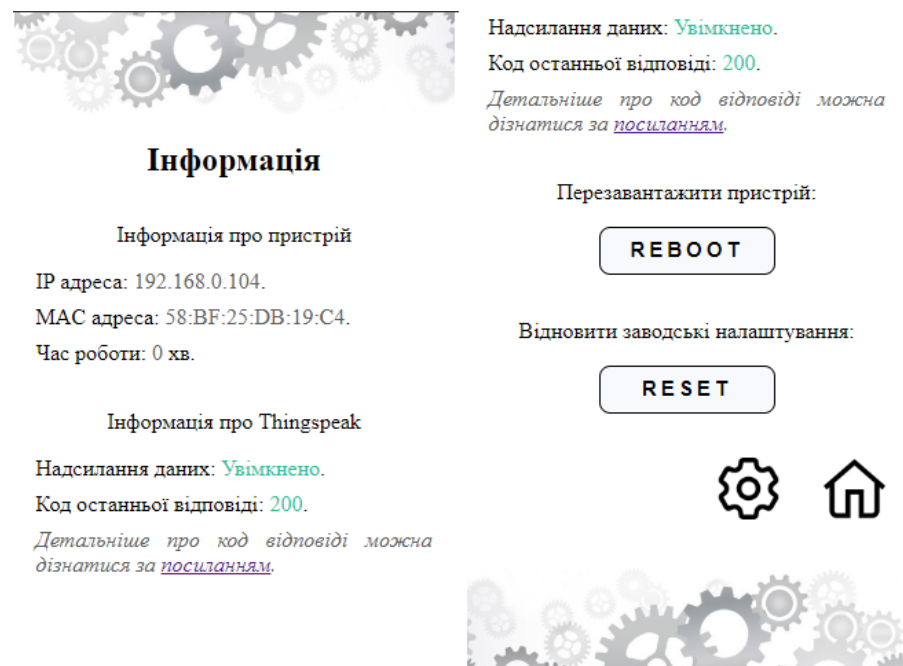


Рисунок 3.5 – Веб сторінка інформації про пристрій

Лістинг веб сторінки наведений в додатку А.

Після повного завантаження веб-сторінки викликається функція `loadDevice()`, яка виводить отриману від сервера інформацію на веб-сторінку. У середині цієї функції відбувається HTTP-запит, який отримує дані щодо IP- та MAC-адрес пристрою, статусу дозволу на публікацію повідомлень, успішності останньої публікації та загального часу роботи пристрою. Ця інформація отримується у формі JSON-масиву, після чого обробляється та виводиться на веб-сторінку.

```
var json = JSON.parse(xhr.response);
```

```

document.getElementById('ip').innerHTML = json.ip;
document.getElementById('mac').innerHTML = json.mac;
document.getElementById('uptime').innerHTML =
Math.floor(json.uptime / 60000);

```

Якщо значення елемента масиву `json.state`, що вказує на дозвіл на відправку, рівне "1", статус відправки змінюється на "Активно". Крім того, на сторінку виводиться код відповіді останнього відправленого повідомлення.

```

if (json.state == "1") {
document.getElementById('state').style.color = '#21C49C';
document.getElementById('state').innerHTML = 'Увімкнено';
document.getElementById('code').innerHTML = json.code;
if (json.code == "200") {
document.getElementById('code').style.color = '#21C49C'; }
else {
document.getElementById('code').style.color = '#DF3D2E';}}
else {
document.getElementById('state').style.color = '#606060';
document.getElementById('state').innerHTML = 'Вимкнено';
document.getElementById('code').innerHTML = 'Не активно';}}}}

```

На веб-сторінці розміщено дві кнопки. Кнопка "Reboot" викликає функцію `reboot()`, яка виводить вікно для підтвердження перезавантаження. Якщо користувач підтверджує перезавантаження, виконується відповідний http-запит на сервер, що призведе до виклику функції перезавантаження мікроконтролера.

```

function reboot() {
var result = window.confirm('Перезавантажити пристрій?');
if (result) {
var xhr = new XMLHttpRequest();
xhr.open("POST", '/reboot');
xhr.send();
alert('Пристрій перезавантажиться через декілька секунд.'));}}

```

Кнопка «Reset» викликає функцію `reset()` і у разі підтвердження скидання на заводські налаштування відправляє запит на виконання відповідної функції мікроконтролеру.

3.3 Налаштування приладу для роботи з сервісом «ThingsSpeak»

На певному веб-сервісі було зареєстровано відповідний обліковий запис. Після цього був створений канал з назвою "DHT11_MQ2HUB" та ідентифікатором "1764077". У цьому каналі були активовані три поля з номерами "2", "4" та "5". У другому полі записувалися дані про концентрацію чадного газу, у четвертому - про температуру, а у п'ятому - про вологість навколишнього середовища. Якщо потрібно записувати дані лише про один тип показників, то слід створити лише одне поле. Процес створення нового каналу показано на рисунку 3.6.

Channel Settings

Percentage complete 30%

Channel ID 1764077

Name DHT11_MQ2HUB

Description

Field 1 ☐

Field 2 CO ☒

Field 3 ☐

Field 4 Temperature ☒

Field 5 Humidity ☒

Рисунок 3.6 – Створення каналу

Далі на вкладці «API Keys» було отримано ключ для запису у створений канал «9YSOOOFTGGSNK4B2», рисунок 3.7.

Write API Key

Key

9YS000FTGGSNK4B2

Generate New Write API Key

Рисунок 3.7 – Отриманий ключ для запису у канал

Після створення каналу для запису даних, пристрій було налаштовано для підключення до WiFi мережі. Процес включав підключення пристрою до джерела живлення та пошук точки доступу під ім'ям "DHT11MQ2HUB". Якщо такої точки не існує або пристрій вже було підключено до іншої точки доступу, користувачеві слід натискати кнопку "WiFi", яка розташована на пристрої, та перезавантажувати його за допомогою кнопки "Reset". Кнопку WiFi можна відпустити протягом трьох секунд після натискання кнопки перезавантаження. Після цих дій з'явиться точка доступу з відповідним ім'ям і буде активною протягом 180 секунд.

Далі відбулося підключення до зазначеної мережі та перехід за IP-адресою "192.168.4.1". На цій адресі була відкрита веб-сторінка для налаштування підключення до мережі WiFi. Після натискання кнопки "Configure WiFi" користувач повинен ввести дані для підключення до домашньої WiFi мережі та натискати кнопку "Save", як показано на рисунку 3.8.

WiFiManager

DHT11MQ2HUB

Configure WiFi

Info

Exit

Update

TP-LINK

TP-Link_865E

SSID

TP-LINK

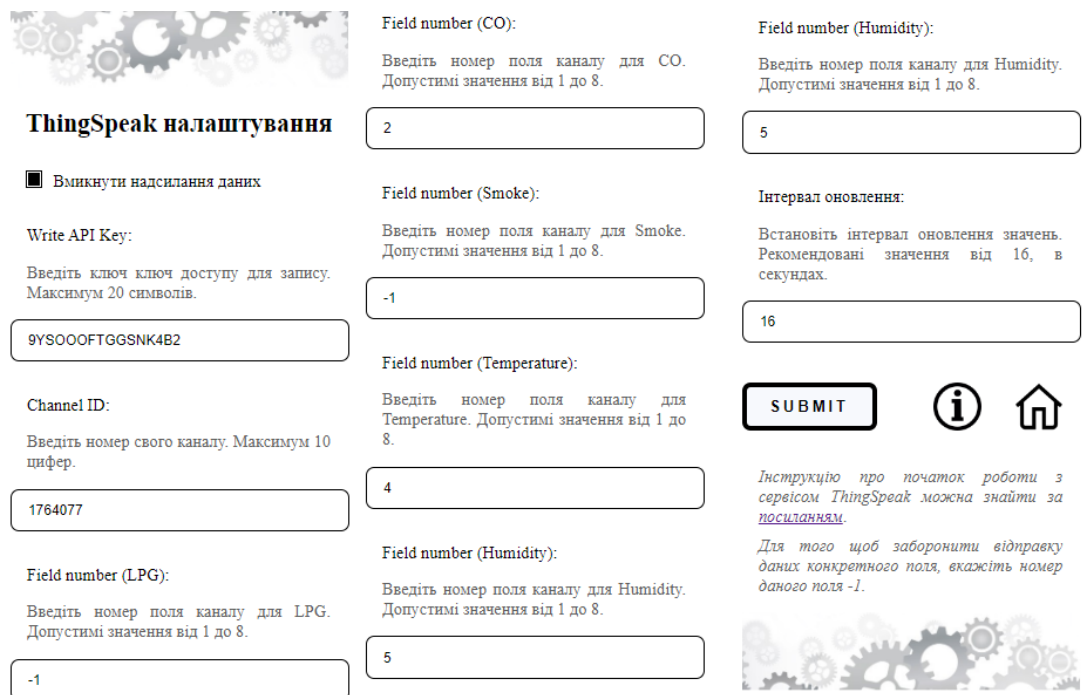
Password

Save

Refresh

Рисунок 3.8 – Підключення до домашньої WiFi мережі

Після підключення пристрою до мережі, шляхом знаходження IP-адреси у списку клієнтів роутера (192.168.1.101), виконано перехід за цією адресою. З метою налаштування сервісу ThingsSpeak, здійснено перехід на сторінку налаштувань, яка відкривається після натискання кнопки із зображенням шестерні. На цій сторінці введено ключ для запису в канал, вказано номер каналу та налаштовано три канали для відправлення повідомлень: "CO", "Temperature" та "Humidity", як зображено на рисунку 3.9.



ThingSpeak налаштування

☒ Вмикнути надсилання даних

Write API Key:
Введіть ключ доступу для запису. Максимум 20 символів.

9YSOOOFTGGSNK4B2

Channel ID:
Введіть номер свого каналу. Максимум 10 цифер.

1764077

Field number (LPG):
Введіть номер поля каналу для LPG. Допустимі значення від 1 до 8.

-1

Field number (CO):
Введіть номер поля каналу для CO. Допустимі значення від 1 до 8.

2

Field number (Smoke):
Введіть номер поля каналу для Smoke. Допустимі значення від 1 до 8.

-1

Field number (Temperature):
Введіть номер поля каналу для Temperature. Допустимі значення від 1 до 8.

4

Field number (Humidity):
Введіть номер поля каналу для Humidity. Допустимі значення від 1 до 8.

5

Field number (Humidity):
Введіть номер поля каналу для Humidity. Допустимі значення від 1 до 8.

5

Інтервал оновлення:
Встановіть інтервал оновлення значень. Рекомендовані значення від 16, в секундах.

16

SUBMIT ⓘ 🏠

Інструкцію про початок роботи з сервісом ThingSpeak можна знайти за [посиланням](#).

Для того щоб заборонити відправку даних конкретного поля, вкажіть номер даного поля -1.

Рисунок 3.9 – Налаштування підключення до TS

Для полів даних "LPG" та "Smoke" були встановлені значення "-1", що призвело до заборони відправлення інформації про концентрацію зрідженого вуглеводневого газу та диму на сервіс і її відображення на головній сторінці. Після цього був налаштований інтервал оновлення, який становив 16 секунд. Після введення даних вони були збережені після натискання кнопки "Submit". Активацію дозволу відправлення було здійснено натисканням відповідного чекбоксу на початку сторінки.

Статус роботи відправки повідомлень на сервіс було перевірено на сторінці інформації про пристрій, як показано на рисунку 3.10. Активація

статусу відправлення була включена, і код відповіді "200" підтверджує успішну публікацію даних у каналі.

Інформація про Thingspeak

Надсилання даних: **Увімкнено**.

Код останньої відповіді: **200**.

Детальніше про код відповіді можна дізнатися за [посиланням](#).

Рисунок 3.10 – Статус публікації даних у канал

Після цього було проведено перевірку надісланих пристроєм даних на сервісі ThingSpeak. Як видно з рисунку 3.11, інформація відображається на вкладці "Private View" на сторінці каналу.

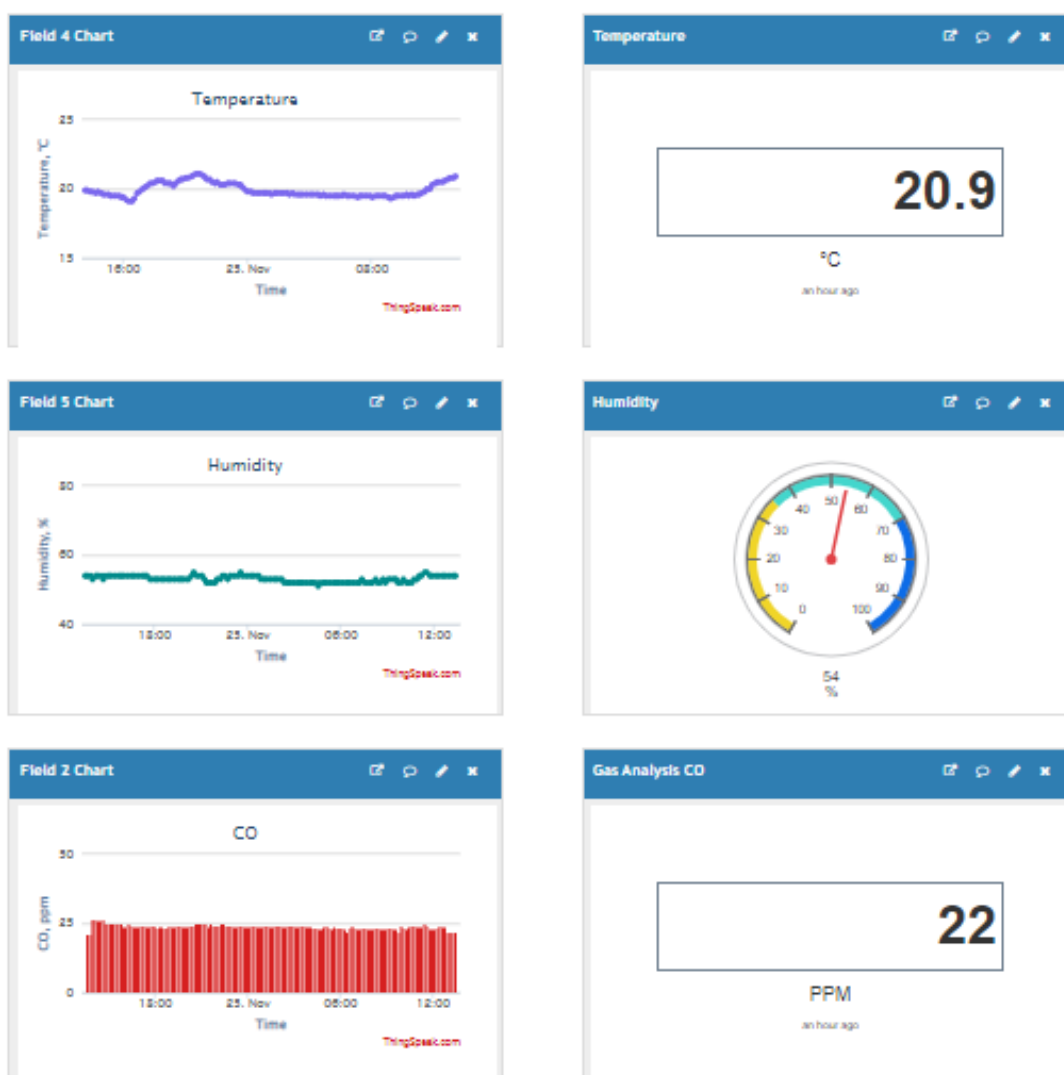


Рисунок 3.11 – Дані температури, вологості та концентрації CO, отримані сервісом TS

На платформі Google Play було знайдено та встановлено мобільний додаток для смартфонів під управлінням операційної системи Android, іменованій "ThingShow – ThingSpeak visualizer" [11]. У додатку був доданий новий канал, введено ідентифікатор створеного каналу та ключ для читання, який знаходиться на вкладці "API Keys". Записані в канал дані було успішно переглянуто на смартфоні, як зображено на рисунку 3.12.

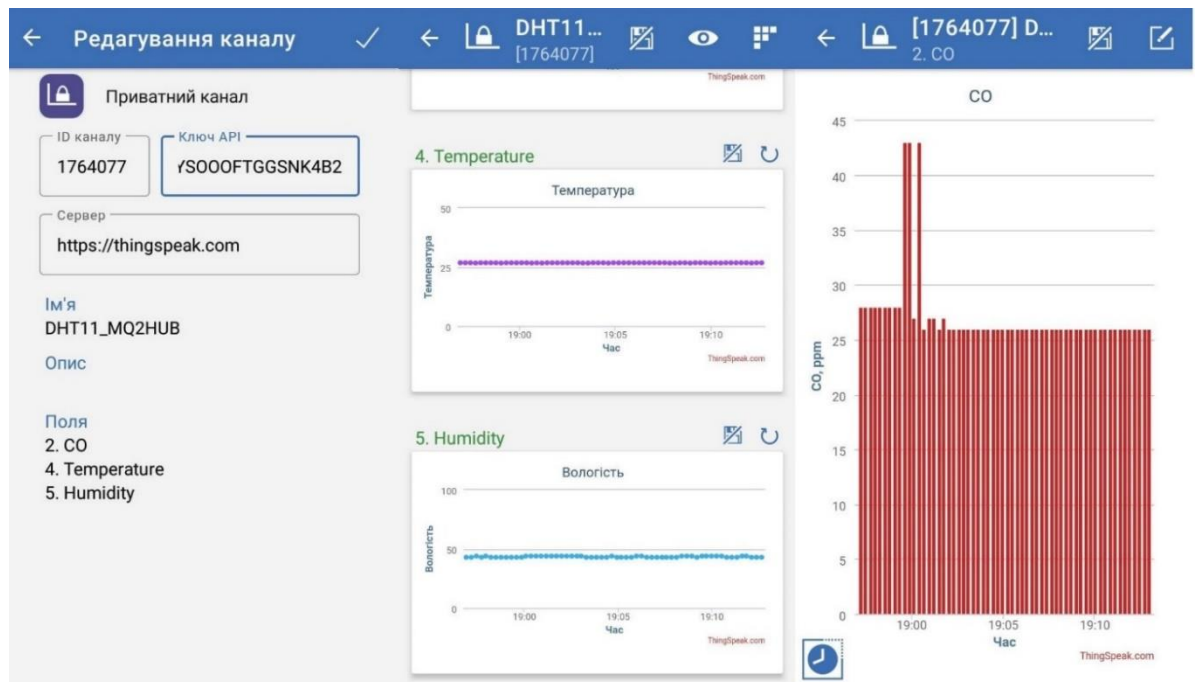


Рисунок 3.12 – Перегляд інформації каналу на пристрої Android

Таким чином, розробка проекту IoT системи моніторингу якості повітря у приміщенні з відображенням даних вимірів у хмарному сервісі ThingSpeak завершена. Використання цієї системи дозволить дослідити зміну середньодобової точності вимірів вологості, температури та концентрації вуглеводних газів у приміщенні за розробленою раніше методикою.

РОЗДІЛ 4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Проведення вимірювання якості повітря

Після отримання готового працюючого пристрою, проведемо заміри у кімнаті. Заміри будуть відбуватися 24 години, із інтервалом відправлення даних кожних 10 хвилин, у хмарний сервіс «ThingSpeak». Загалом ми отримаємо $n=144$ заміри для кожного з параметрів протягом доби. Графічне відображення результатів контролю якості повітря, що були отримані в хмарному сервісі ThingSpeak, наведені на рисунках 4.1,...,4.3.

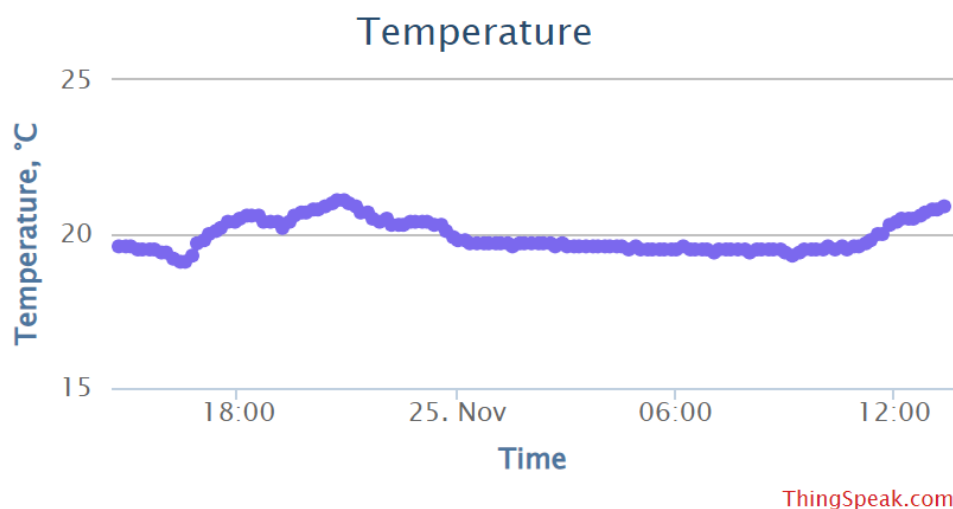


Рисунок 4.1 – Графік замірів температури

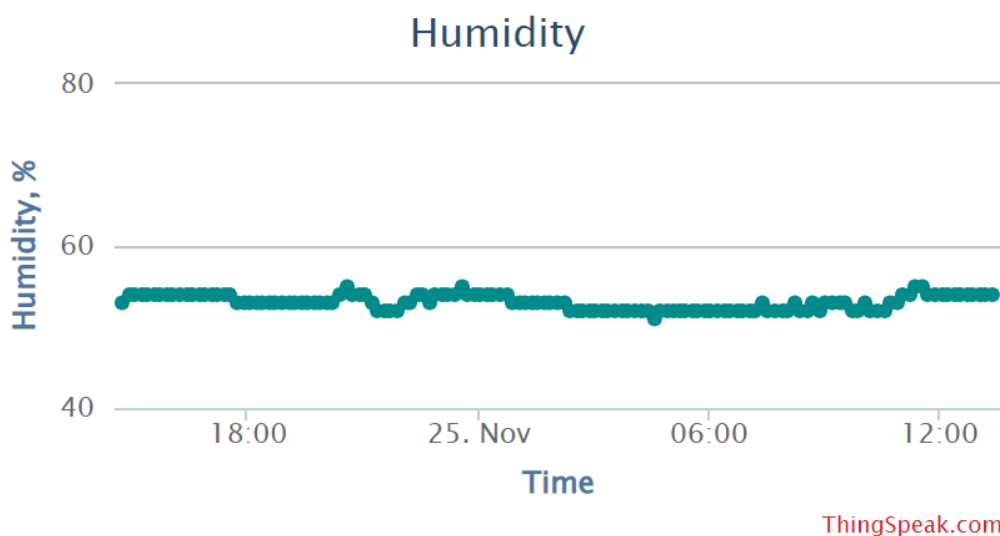


Рисунок 4.2 – Графік замірів вологості

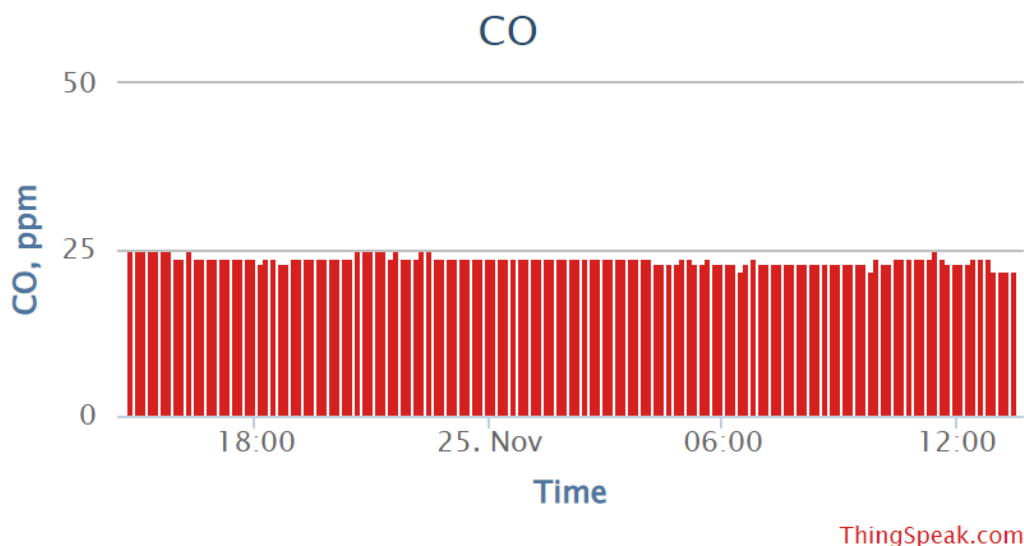


Рис. 4.3 – Графік замірів CO

Для того щоб вивантажити отримані дані, необхідно перейти на сайті «ThingSpeak» у вкладку «Data Import/Export», та обрати свій часовий пояс, та натиснути «Download» (рисунок 4.4).

Рисунок 4.4 – Ввивантаження отриманих даних

Дані вимірів, що були вивантажені, розміщені у таблиці, яка наведена у Додатку В.

4.2 Обчислення середньодобової точності вимірювань

Проведемо обрахунки згідно методики, котра була описана у розділі 2.4. Для обчислення середньодобової точності вимірювань будемо використовувати дані, що містяться у додатку В.

Для того щоб обрахувати середнє квадратичне відхилення вимірів температури, необхідно знайти середнє значення:

$$T_{\text{сер}} = \frac{\sum_{i=1}^n T_i}{144} = 19,91 \text{ }^{\circ}\text{C}.$$

Обрахуємо середнє квадратичне відхилення вимірів температури для обсягу вимірів $n = 144$:

$$T_{\text{СКВ}} = \sqrt{\frac{\sum_{i=1}^n |T_i - 19,91|^2}{144}} = 0,4973 \text{ }^{\circ}\text{C}.$$

Із розділу 2.2 передуює, що похибка вимірювання температури датчиком DHT11 складає $\pm 2^{\circ}\text{C}$, а при випадковому характері похибок вимірів, можна вважати, що відхилення вимірів температури від середнього значення не будуть перевищувати $\pm 3T_{\text{СКВ}}$ з довірчою ймовірністю 0,997. За результатами проведених розрахунків з'ясувалось, що відхилення вимірів температури від середнього значення знаходилось в межах $\pm 1,5^{\circ}\text{C}$. Отже середньодобова похибка вимірів температури не перевищує значення інструментальної похибки вимірів датчика.

Обрахуємо середнє значення вологості:

$$H_{\text{сер}} = \frac{\sum_{i=1}^n H_i}{144} = 53,1 \text{ } \%$$

Обрахуємо середнє квадратичне відхилення вимірів вологості:

$$H_{\text{СКВ}} = \sqrt{\frac{\sum_{i=1}^n |H_i - 53,1|^2}{144}} = 0,9031 \text{ } \%$$

Згідно [5], похибка виміру вологості датчика DHT11 складає $\pm 5\%$. Отже, при випадковому характері похибок вимірів, можна вважати, що відхилення вимірів вологості від середнього значення не будуть перевищувати $\pm 3H_{\text{СКВ}}$ з довірчою ймовірністю 0,997. За результатами проведених розрахунків з'ясувалось, що відхилення вимірів вологості від середнього значення

знаходилося в межах $\pm 2,7\%$. Отже середньодобова похибка вимірів вологості також не перевищує значення інструментальної похибки вимірів датчика.

Обрахуємо середнє значення концентрації чадного газу:

$$G_{\text{сер}} = \frac{\sum_{i=1}^n G_i}{144} = 23,81 \text{ ppm.}$$

Обрахуємо середнє квадратичне відхилення чадного газу:

$$G_{\text{скв}} = \sqrt{\frac{\sum_{i=1}^n |G_i - 23,81|^2}{144}} = 0,8793 \text{ ppm.}$$

Значення похибки вимірів концентрації газу CO датчиком MQ2 становить $\pm 5 \text{ ppm}$, а відхилення вимірів концентрації газу від середнього значення склало $\pm 2,64 \text{ ppm}$. А отже, як і для вимірів інших двох параметрів, можна стверджувати, що середньодобова похибка вимірів концентрації чадного газу також не перевищує значення інструментальної похибки вимірів датчика.

Зведені дані розрахунків середньодобової точності вимірювань параметрів, що характеризують якість повітря у приміщенні, містяться у таблиці 4.1.

Таблиця 4.1 – Обраховані значення дослідження

	Температура, °C	Вологість, %	Концентрація CO, ppm
Середнє значення	19,9146	53,0972	23,8056
Стандартне відхилення	0,4973	0,9031	0,8793
Точність виміру	$\pm 1,5$	$\pm 2,7$	$\pm 2,64$

З наведених даних передусь, що концентрація CO в приміщенні змінювалась не суттєво протягом доби. Середнє значення концентрації чадного газу склало $23,81 \text{ ppm}$, а максимальне досягало 26 ppm , що є цілком допустимим [11].

Середнє значення температури у приміщенні склало за результатами досліджень $19,91^\circ\text{C}$. Такий показник температури є допустимим у період опалювального сезону і забезпечує високий достатній тепловий комфорт і

підтримання відносної вологості на належному рівні. Більш висока температура сприяє висиханню слизових оболонок дихальних шляхів, що підвищує сприйнятливості до бактерій і вірусів в горлі і носі. Це послаблює природні механізми захисту від вірусів і бактерій [12].

Вологість під час досліджень у кімнаті практично не змінювалась. Максимальне значення вологості повітря склало 55%, а мінімальне - 51%. Згідно з санітарними правилами, в холодний період року у житлових приміщеннях вологість повітря не повинна перевищувати 60% [13].

ВИСНОВКИ

Дослідження дистанційної системи моніторингу якості повітря у приміщенні з використанням сучасних технологій IoT є актуальним науковим завданням через те, що несвоєчасне виявлення зростаючого рівня загроз від забруднень повітря може суттєво вплинути на здоров'я людини.

Метою дослідження було визначення середньодобової точності вимірів показників якості повітря при використанні датчиків вологості, температури та концентрації вуглеводневих газів. Для відстеження та аналізу стану повітря у приміщенні у реальному часі було розроблено IoT систему моніторингу з графічним відображенням зміни контрольованих параметрів протягом доби у хмарному сервісі «ThingSpeak».

Одною із задач дослідження було розроблення методики розрахунку середньодобової точності вимірювань вологості, температури та концентрації вуглеводнів у приміщенні. Як показники точності вимірів якості повітря було запропоновано використання середньоквадратичного відхилення вимірів від середнього значення.

За результатами проведених досліджень середнє значення температури у приміщенні склало $19,91^{\circ}\text{C}$, а відхилення вимірів температури від середнього значення знаходилось в межах $\pm 1,5^{\circ}\text{C}$. Середнє значення вологості повітря склало 53,1 %, а відхилення вимірів вологості від середнього значення склало $\pm 1,5^{\circ}\text{C}$. Концентрація чадного газу в приміщенні змінювалась не суттєво протягом доби. Середнє значення концентрації чадного газу склало 23,8 ppm, а відхилення вимірів концентрації газу від середнього склало $\pm 2,64$ ppm.

Отримані значення показників якості повітря відповідають санітарними нормам у холодний період року у житлових приміщеннях, а середньодобова похибка вимірів контрольованих параметрів не перевищує значення інструментальної похибки вимірів датчиків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Проблематика громадського здоров'я [Електронний ресурс] – Режим доступу до ресурсу: http://cgz.vn.ua/problematika-gromadskogo-zdorovya/problematika-gromadskogo-zdorovya_455.html.
2. Мікроклімат виробничих приміщень та вплив мікроклімату на працездатність людини [Електронний ресурс] – Режим доступу до ресурсу: <http://lutsk.dsp.gov.ua/мікроклімат-виробничих-приміщень-та/>.
3. ThingSpeak - The IoT Platform with MATLAB Analytics [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mathworks.com/help/thingspeak>.
4. Модуль WiFi ESP8266 ESP-12F [Електронний ресурс] – Режим доступу до ресурсу: https://voron.ua/catalog/028451--modul_wifi_esp8266_esp-12f.
5. Модуль датчика вологості та температури [Електронний ресурс] – Режим доступу до ресурсу: <https://arduino.ua/ru/prod185-datchik-vlajnosti-i-temperatyri-dht11>.
6. Модуль датчика диму MQ-2 [Електронний ресурс] – Режим доступу до ресурсу: <https://arduino.ua/prod298-modyl-datchika-dima-mq-2>.
7. Technical data MQ-2 gas sensor [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mouser.com/datasheet/2/321/605-00008-MQ-2-Datasheet-370464.pdf>.
8. ESP8266 ADC – Read Analog Values with Arduino IDE, MicroPython and Lua [Електронний ресурс] – Режим доступу до ресурсу: <https://randomnerdtutorials.com/esp8266-adc-reading-analog-values-with-nodemcu>.
9. Measuring PPM from MQ Gas Sensors using Arduino (MQ-137 Ammonia) [Електронний ресурс] – Режим доступу до ресурсу:

<https://circuitdigest.com/microcontroller-projects/arduino-mq137-ammonia-sensor>.

10. ThingShow - ThingSpeak visualizer [Електронний ресурс] – Режим доступу до ресурсу: <https://play.google.com/store/apps/details?id=com.devinterestdev.thingshow>
11. Де і як встановлювати датчик чадного газу (CO) [Електронний ресурс] – Режим доступу до ресурсу: <https://support.ajax.systems/uk/how-to-install-co-sensors/>
12. Кімнатна температура – якою вона повинна бути? [Електронний ресурс] – Режим доступу до ресурсу: <https://ht-heiztechnik.ua/bloh?layout=edit&id=112>
13. Норма вологості в квартирі. Як регулювати вологість у приміщенні [Електронний ресурс] – Режим доступу до ресурсу: <https://kantal.lviv.ua/норма-вологості-в-квартирі-як-регулюв/>

Додаток А

```

#include <ESP8266WiFi.h> // Бібліотека для роботи з wifi
#include <WiFiClient.h> // Бібліотека для роботи з wifi
#include <ESP8266WebServer.h> // Бібліотека для роботи з веб сервером
#include <WebSocketsServer.h> // Бібліотека для роботи з вебсокет сервером
#include <WiFiManager.h> // Бібліотека для зручного підключення до wifi мережі
#include <FS.h> // Бібліотека для роботи з файловою системою
#include <EEPROM.h> // Бібліотека для роботи з пам'яттю EEPROM
#include "ThingSpeak.h" // Бібліотека для роботи з сервісом TS
#include <DHT.h> // Бібліотека для роботи з DHT датчиком
ESP8266WebServer server(80); // Створення веб серверу на порту 80
WebSocketsServer webSocket = WebSocketsServer(81); // Створення вебсокет серверу
на порту 81
WiFiClient client; // Створення клієнту wifi
WiFiManager wifiManager; // Створення клієнту wifi manager
String getContentType(String filename); // Строка яка містить ім'я файлу веб серверу
bool handleFileRead(String path); // Строка яка містить шлях до файлу
unsigned long timing1, timing2 = 0; // Системні змінні для зберігання часових
інтервалів
String val, state, key, id, field1, field2, field3, field4, field5, period,
ip, mac, code; // Змінні для збереження зчитаних даних з EEPROM
const char * writeApiKey; // Змінна для збереження ключа запису в форматі char
unsigned long channelNumber, updatePeriod; // Змінні для збереження номеру каналу
та часу
оновлення каналу
int fieldNumber1, fieldNumber2, fieldNumber3, fieldNumber4, fieldNumber5, x; //
Змінні для збереження полів каналу та коду відповіді в форматі int
#define adcPin A0 // Пін входу АЦП
#define wifiPin 4 // Пін кнопки для підключення до wifi
#define DHTPIN 5 // Пін підключення датчика D1
DHT dht(DHTPIN, DHT11);
int readSample = 10; // Кількість виборок АЦП
float VCC = 5; // Напруга живлення датчику, В
float Vref = 1; // Напруга живлення АЦП, В
float Vdiv = 2.77; // Коефіцієнт зменшення напруги резистивним дільником
float RL = 2; // Номінал резистору RL, кОм
float R0 = 7.62; // Відкаліброване значення R0

```

```

float lpgCurve[2] = { -0.48, 1.33}; // Розраховані значення кривої для LPG газу
float coCurve[2] = { -0.32, 1.44}; // Розраховані значення кривої для CO газу
float smokeCurve[2] = { -0.45, 1.64}; // Розраховані значення кривої для диму
int lpgPPM, coPPM, smokePPM, temperature, humidity; // Змінні для зберігання значень ppm
void setup() {
    Serial.begin(115200); // Запуск послідовного порту на швидкості 115200 бод
    pinMode(wifiPin, INPUT_PULLUP); // Призначення піну кнопки як вхід підтягнутий
до напруги живлення (PULLUP)
    dht.begin();
    Serial.println(""); // Вивід пустого символу в монітор порту для відділення
повідомлення
при запуску
    delay(3000); // Очікування 3 секунди для натискання кнопки підключення wifi
    if (digitalRead(wifiPin) == LOW) { // Якщо натиснуто кнопку підключення до wifi
        wifiManager.resetSettings(); // Стерти налаштування wifi
        Serial.println("Reset wifi settings!");
    }
    Serial.println("Start DHT-11 and MQ-2 Hub");
    EEPROM.begin(44); // Ініціалізація EEPROM 44 біт даних
    if (!SPIFFS.begin()) { // Перевірка ініціалізації EEPROM
        Serial.println("An Error has occurred while mounting SPIFFS");
        return;
    }
    wifiManager.setTimeout(180); // Встановлення wifi точки доступу для налаштування
на 180 с.
    wifiManager.autoConnect("DHT11MQ2HUB"); // Задання ім'я точки доступу
    Serial.println("");
    eepromRead(); // Функція читання EEPROM
    server.onNotFound([]() { // У випадку, якщо веб шлях не знайдено
        if (!handleFileRead(server.uri()))
            server.send(404, "text/plain", "404: Not Found"); // Вивести на сторінку
інформацію
про відсутність
    });
    server.on("/reboot", reboot); // Якщо звернення за шляхом /reboot, викликати
функцію reboot()
    server.on("/reset", erase);
    server.on("/loadDevice", loadDevice);
    server.on("/loadConfig", loadConfig);
    server.on("/saveStatus", saveStatus);
    server.on("/saveConfig", saveConfig);

```

```

server.on("/loadFields", loadFields);
server.begin(); // Ініціалізація веб серверу
WebSocket.begin(); // Ініціалізація вебсокет серверу
}
void loop() {
    server.handleClient(); // Тримати веб сервер активним
    WebSocket.loop(); // Тримати вебсокет сервер активним
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();
    if (millis() - timing1 > 2000) { // Оновлення значень PPM раз у дві секунди
        timing1 = millis(); // Оновлення змінної зберігання часу оновлення даних
        float ratio = findRatio(adcPin); // Розрахунок відношення RS/R0
        lpgPPM = findPPM(ratio, lpgCurve); // Розрахунок ppm якщо газ - LPG
        coPPM = findPPM(ratio, coCurve); // Розрахунок ppm якщо газ - CO
        smokePPM = findPPM(ratio, smokeCurve); // Розрахунок ppm якщо газ - Smoke
        String payload = String(lpgPPM) + "," + String(coPPM) + "," + String(smokePPM)
+ "," + String(temperature) + "," + String(humidity); // Формування строки для надсилання
на веб сторінку
        WebSocket.broadcastTXT(payload); // Надсилання строки через вебсокет сервер
        Serial.print("Temperature: "); // Вивід інформації
        Serial.println(temperature);
        Serial.print("Humidity: ");
        Serial.println(humidity);
        Serial.print("Ratio: ");
        Serial.println(ratio);
        Serial.print("LPG: ");
        Serial.println(lpgPPM);
        Serial.print("CO: ");
        Serial.println(coPPM);
        Serial.print("Smoke: ");
        Serial.println(smokePPM);
        Serial.println("");
    }
    if (state == "1" && millis() - timing2 > updatePeriod) { // Перевірка дозволу на
відправку даних у TS та інтервалу оновлення
        timing2 = millis(); // Оновлення змінної зберігання часу відправки
        if (fieldNumber1 != -1) { // Перевірка дозволу в запис поля fieldNumber1
            ThingSpeak.setField(fieldNumber1, lpgPPM); // Якщо дозволено, записати в поле
дані про LPG
        }
        if (fieldNumber2 != -1) { // Перевірка дозволу в запис поля fieldNumber2

```

```

    ThingSpeak.setField(fieldNumber2, coPPM); // Якщо дозволено, записати в поле дані
про CO
}
if (fieldNumber3 != -1) { // Перевірка дозволу в запис поля fieldNumber3
    ThingSpeak.setField(fieldNumber3, smokePPM); // Якщо дозволено, записати в поле дані
про Smoke
}
if (fieldNumber4 != -1) { // Перевірка дозволу в запис поля fieldNumber4
    ThingSpeak.setField(fieldNumber4, temperature); // Якщо дозволено, записати в поле
дані про Temperature
}
if (fieldNumber5 != -1) { // Перевірка дозволу в запис поля fieldNumber5
    ThingSpeak.setField(fieldNumber5, humidity); // Якщо дозволено, записати в поле дані
про Humidity
}
x = ThingSpeak.writeFields(channelNumber, writeApiKey); // Оновити інформацію в полях
на TS
Serial.print("Update TS payload. Response: "); // Вивід в монітор порту коду
відповіді TS
Serial.println(x);
}
}
float findRatio(int pin) { // Розрахунок відношення RS/R0
    // Використано простий фільтр середнього арифметичного
    float sum = 0; // Змінна для зберігання значень

    for (int i = 0; i < readSample; i++) { // Зчитування значення АЦП
        sum += (float) analogRead(adcPin) * Vref * Vdiv / 1024; // Розрахунок напруги
на виході датчику
        // Через використання дільника напруги на 5 отримане значення було помножено на Vdiv!
    }
    float adc = (float) sum / readSample; // Розрахунок середнього арифметичного
значення напруги
    float RS = (float) ((VCC / adc) - 1) * RL; // Розрахунок опору датчику RS, кОм
    float ratio = RS / R0; // Розрахунок відношення RS/R0
    return (ratio); // Повернути розраховане значення RS/R0
}
int findPPM (float ratio, float *curve) { // Розрахунок ppm для різних газів
    int ppm = pow(10, (log10(ratio) - curve[1]) / curve[0]); // Розрахунок значення ppm
    if (ppm > 10000) { // Якщо розраховане ppm більше ніж 10000
        ppm = 10000; // Обмежити до 10000 згідно даташиту
    }
}

```

```

    }
    return (ppm); // Повернути значення ppm
}

String getContentType(String filename) { // Функція для відображення даних на веб сторінці
    if (filename.endsWith(".html")) return "text/html"; // Передача html файлу
    else if (filename.endsWith(".css")) return "text/css"; // Передача css файлу
    else if (filename.endsWith(".js")) return "application/javascript"; // Передача js файлу
    else if (filename.endsWith(".ico")) return "image/x-icon"; // Передача ico файлу
    return "text/plain"; // Передача тексту
}

bool handleFileRead(String path) { // Функція передачі файлу веб сторінки
    Serial.println("handleFileRead: " + path); // Вивід інформації про шлях відкритої
    сторінки
    if (path.endsWith("/")) path += "index.html"; // Якщо шлях складається з "/",
    відобразити головну сторінку
    String contentType = getContentType(path); // Отримання типу інформації для відображення
    if (SPIFFS.exists(path)) { // Якщо файл існує
        File file = SPIFFS.open(path, "r"); // Відкриття файл у режимі читання
        size_t sent = server.streamFile(file, contentType); // Відправка файл на веб сторінку
        file.close(); // Закриття файлу
        return true; // Підтвердження наявності файлу
    }
    Serial.println("\tFile Not Found");
    return false; // Не підтвердження наявності файлу
}

void loadConfig() { // Відправка даних на сторінку налаштування
    String json = "{"; // Формування json масиву
    json += "\"state\":\""; // Інформація про стан відправки дозволено/недозволено
    json += state;
    json += "\",\"key\":\""; // Ключ для запису в канал
    json += key;
    json += "\",\"id\":\""; // ID каналу
    json += id;
    json += "\",\"field1\":\""; // Поле для LPG
    json += field1;
    json += "\",\"field2\":\""; // Поле для CO
    json += field2;
    json += "\",\"field3\":\""; // Поле для Smoke
    json += field3;
    json += "\",\"field4\":\""; // Поле для Temperature

```

```

    json += field4;
    json += "\",\"field5\":\"; // Поле для Humidity
    json += field5;
    json += "\",\"int\":\"; // Інтервал запису в канал
    json += period;
    json += "\"}";
    server.send(200, "text/json", json); // Відправка масиву на веб сторінку
}

void loadDevice() { // Відправка даних на сторінку інформації
    String ip, mac; // Змінні для збереження IP та MAC адрес
    for (int i = 0; i < 4; i++) { // Формування IP адреси
        ip += WiFi.localIP()[i]; // Додавання в змінну кожного з чотирьох октетів
        ip += "."; // Додавання розділового знаку між октетами
    }
    ip = ip.substring(0, ip.length() - 1); // Видалення останнього розділового знаку
    for (int i = 0; i < 17; i++) { // Формування MAC адреси
        mac += WiFi.macAddress()[i]; // Зчитування MAC адреси по символам у змінну
    }
    String json = "{"; // Формування json масиву
    json += "\"ip\":\"; // Інформація про IP адресу
    json += ip;
    json += "\",\"mac\":\"; // Інформація про MAC адресу
    json += mac;
    json += "\",\"uptime\":\"; // Інформація про час роботи приладу
    json += millis();
    json += "\",\"state\":\"; // Інформація про стан відправки дозволено/недозволено
    json += state;
    json += "\",\"code\":\"; // Код останньої відповіді TS при відправці даних
    json += String(x);
    json += "\"}";
    server.send(200, "text/json", json); // Відправка масиву на веб сторінку
}

void loadFields() {
    //
    String fields = field1 + "," + field2 + "," + field3 + "," + field4 + "," + field5;
    server.send(200, "text/plain", fields); // Відправка масиву на веб сторінку
}

void saveStatus() { // Функція збереження дозволу відправки на TTN у EEPROM

```

```

server.send(200, "text/html"); // Відправка коду ОК на веб сторінку
// Status
EEPROM.write(0, server.arg("state")[0]); // Збереження дозволу у EEPROM
EEPROM.commit(); // Підтвердження запису в EEPROM
eepromRead(); // Виклик функції читання даних з EEPROM
}

void saveConfig() { // Збереження даних налаштування TTN у EEPROM
  server.send(200, "text/html"); // Надсилання коду ОК на веб сторінку
  for (int i = 1; i <= 43 ; i++) {
    EEPROM.write(i, 0); // clear 1 - 43 // Очищення старих даних EEPROM
  }
  // API Key [20] біт
  for (int i = 0; i < 20; i++) { // Запис даних про ключ запису TS
    EEPROM.write(1 + i, server.arg("key")[i]); // Запис у EEPROM по одному символу
  }
  // Channel ID [10] біт
  for (int i = 0; i < 10; i++) { // Запис даних про ID каналу
    EEPROM.write(21 + i, server.arg("id")[i]);
  }
  // Field number1 [2] біт
  for (int i = 0; i < 2; i++) { // Запис даних про поле LPG
    EEPROM.write(31 + i, server.arg("field1")[i]);
  }
  // Field number2 [2] біт
  for (int i = 0; i < 2; i++) { // Запис даних про поле CO
    EEPROM.write(33 + i, server.arg("field2")[i]);
  }
  // Field number3 [2] біт
  for (int i = 0; i < 2; i++) { // Запис даних про поле Smoke
    EEPROM.write(35 + i, server.arg("field3")[i]);
  }
  // Field number4 [2] біт
  for (int i = 0; i < 2; i++) { // Запис даних про поле Temperature
    EEPROM.write(37 + i, server.arg("field4")[i]);
  }
  // Field number5 [2] біт
  for (int i = 0; i < 2; i++) { // Запис даних про поле Humidity
    EEPROM.write(39 + i, server.arg("field5")[i]);
  }
  // Update interval [3] біт

```

```

for (int i = 0; i < 3; i++) { // Запис даних про інтервал оновлення каналу TS
    EEPROM.write(41 + i, server.arg("int")[i]);
}
EEPROM.commit(); // Підтвердження запису даних у EEPROM
eepromRead(); // Виклик функції читання EEPROM
}

void eepromRead() { // Функція читання EEPROM
    Serial.println("EEPROM storage: ");
    // Status [1] біт Зчитування дозволу відправки на TS.
    if (char(EEPROM.read(0)) == 0x00) { // Якщо інформація відсутня
        state = ""; // Стерти дані збережені у змінній
    }
    else {
        state = char(EEPROM.read(0)); // Якщо інформація наявна, зберегти дані у змінній
    }
    Serial.print("State: ");
    Serial.println(state);

    // API Key [20] біт
    key = ""; // Стерти дані у змінній перед читанням інформації з EEPROM
    for (int i = 1; i <= 20; i++) { // Читання інформації з EEPROM по одному символу
        if (char(EEPROM.read(i)) != 0x00) {
            key += char(EEPROM.read(i)); // Додавання зчитаного значення у змінну
        }
    }
    Serial.print("API Key: ");
    Serial.println(key);
    // Channel ID [10]
    id = "";
    for (int i = 21; i <= 30; i++) { // Читання інформації про ID каналу
        if (char(EEPROM.read(i)) != 0x00) {
            id += char(EEPROM.read(i));
        }
    }
    Serial.print("Channel ID: ");
    Serial.println(id);
    // Field number 1 [2] біт
    field1 = "";
    for (int i = 31; i <= 32; i++) { // Читання інформації про поле LPG
        if (char(EEPROM.read(i)) != 0x00) {

```

```

        field1 += char(EEPROM.read(i));
    }
}
Serial.print("Field number 1: ");
Serial.println(field1);
// Field number 2 [2] 6іт
field2 = "";
for (int i = 33; i <= 34; i++) { // Читання інформації про поле CO
    if (char(EEPROM.read(i)) != 0x00) {
        field2 += char(EEPROM.read(i));
    }
}
Serial.print("Field number 2: ");
Serial.println(field2);
// Field number 3 [2] 6іт
field3 = "";
for (int i = 35; i <= 36; i++) { // Читання інформації про поле Smoke
    if (char(EEPROM.read(i)) != 0x00) {
        field3 += char(EEPROM.read(i));
    }
}
Serial.print("Field number 3: ");
Serial.println(field3);
// Field number 4 [2] 6іт
field4 = "";
for (int i = 37; i <= 38; i++) { // Читання інформації про поле Temperature
    if (char(EEPROM.read(i)) != 0x00) {
        field4 += char(EEPROM.read(i));
    }
}
Serial.print("Field number 4: ");
Serial.println(field4);
// Field number 5 [2] 6іт
field5 = "";
for (int i = 39; i <= 40; i++) { // Читання інформації про поле Humidity
    if (char(EEPROM.read(i)) != 0x00) {
        field5 += char(EEPROM.read(i));
    }
}
Serial.print("Field number 5: ");
Serial.println(field5);

```

```

// Update interval
period = "";
for (int i = 41; i <= 43; i++) { // Читання інформації про період оновлення каналу TS
    if (char(EEPROM.read(i)) != 0x00) {
        period += char(EEPROM.read(i));
    }
}
Serial.print("Update interval: ");
Serial.println(period);
writeApiKey = key.c_str(); // Конвертування ключа запису з формату String в формат char
channelNumber = id.toInt(); // Конвертування ID каналу з формату String в формат int
fieldNumber1 = field1.toInt();
fieldNumber2 = field2.toInt();
fieldNumber3 = field3.toInt();
fieldNumber4 = field4.toInt();
fieldNumber5 = field5.toInt();
updatePeriod = period.toInt() * 1000; // Конвертування періоду запису в канал в
формат int в мс
if (state == "1") { // Якщо дозволено запис в канал TS
    ThingSpeak.begin(client); // Ініціалізувати клієнт для запису в канал
    Serial.println("ThingSpeak: Enable");
}
else {
    Serial.println("ThingSpeak: Disable");
}
}

void reboot() { // Функція перезавантаження приладу
    Serial.println("Restart");
    delay(100); // Очікування для коректного виводу інформації в монітор порта
перед перезавантаженням
    ESP.restart(); // Виклик перезавантаження приладу
}

void erase() { // Функція скидання на заводські налаштування
    for (int i = 0; i <= 43 ; i++) { // Очищення пам'яті EEPROM
        EEPROM.write(i, 0);
    };
    EEPROM.commit();
    wifiManager.resetSettings(); // Скидання інформації про збережену wifi мережу
    Serial.println("EEPROM erased");
    Serial.println("Reset WiFi settings!");
    eepromRead(); // Читання даних з EEPROM
}

```

```
}
```

File «index.html»

```
<!DOCTYPE html>
<html>
<head>
<title>DHT-11 та MQ-2 Якість повітря</title>
<meta charset="UTF-8">
<meta name="viewport" content="width = device-width, initial-scale = 1.0,
maximum-scale = 1.0">
</head>
<style>
body { margin-top: 0; margin-bottom: 0; margin-left: 0; margin-right: 0;
background-color: #FFFFFF; }
.container { margin-top: 0; margin-bottom: 0; margin-left: auto;
margin-right: auto; width: 20rem; }
.header { margin-top: 0; margin-bottom: 0; }
.title { margin-top: 0; margin-bottom: 0; text-align: center; font-size: 20pt;
font-weight: lighter; }
.indicator { margin-top: 2rem; margin-bottom: 1rem; width: 10rem; }
.value { margin-top: 0; margin-bottom: 0; text-align: center;
font-family: 'Montserrat', sans-serif; font-weight: lighter; font-size: 20pt; }
.hint { margin-top: 1rem; margin-left: 1rem; margin-right: 1rem;
text-align: center; font-size: 15pt; color: #606060; }
.icon { margin-bottom: 1rem; margin-left: 1rem; margin-right: 1rem;
float: right; outline: none; text-decoration: none; }
.icon img { border: none; }
.icon:hover { text-decoration: none; outline: none; }
#lpg, #co, #smoke, #temperature, #humidity { display: none; }
</style>
<body onload="init();">
<div class="container">

<p class="title"><b>MQ-2</b></p>
<p class="title">Контроль якості повітря</p>
<p style="text-align: center; margin-top: 0; margin-bottom: 0;"></p>
<p class="value" id="lpg">LPG: <span id="val1">*</span> ppm</p>
<p class="value" id="co">CO: <span id="val2">*</span> ppm</p>
```

```

<p class="value" id="smoke">Smoke: <span id="val3">*</span> ppm</p>
<p></p>
<p class="value" id="temperature">Temperature: <span id="val4">*</span>°C</p>
<p class="value" id="humidity">Humidity: <span id="val5">*</span> %</p>
<p class="hint" id="desc">Завантаження даних...</p>
<a class="icon" href="settings.html"></a>
<a class="icon" href="info.html"></a>

</div>
</body>
<script>
var Socket;
var field1 = -1;
var field2 = -1;
var field3 = -1;
var field4 = -1;
var field5 = -1;
function init() {
Socket = new WebSocket('ws://' + window.location.hostname + ':81/');
Socket.onmessage = function(event) { processReceivedCommand(event); };
var xhr = new XMLHttpRequest();
xhr.open("POST", '/loadFields');
xhr.send(null);
xhr.onload = function() {
if (xhr.status == 200) {
var fields = xhr.response.split(',', 5);
if (fields[0] !== '-1') {
document.getElementById('lpg').style.display = 'block';
field1 = parseInt(fields[0])}
if (fields[1] !== '-1') {
document.getElementById('co').style.display = 'block';
field2 = parseInt(fields[1])}
if (fields[2] !== '-1') {
document.getElementById('smoke').style.display = 'block';
field3 = parseInt(fields[2])}
if (fields[3] !== '-1') {
document.getElementById('temperature').style.display = 'block';
field4 = parseInt(fields[3])}

```

```

if (fields[4] !== '-1') {
document.getElementById('humidity').style.display = 'block';
field5 = parseInt(fields[4])}}}}
function processReceivedCommand(value) {
var payload = value.data.split(',', 5);
var val1 = parseInt(payload[0]);
var val2 = parseInt(payload[1]);
var val3 = parseInt(payload[2]);
var val4 = parseInt(payload[3]);
var val5 = parseInt(payload[4]);
if (field1 == -1) {val1 = -1;}
if (field2 == -1) {val2 = -1;}
if (field3 == -1) {val3 = -1;}
if (field4 == -1) {val4 = -1;}
if (field5 == -1) {val5 = -1;}
document.getElementById('val1').innerHTML = val1;
document.getElementById('val2').innerHTML = val2;
document.getElementById('val3').innerHTML = val3;
document.getElementById('val4').innerHTML = val4;
document.getElementById('val5').innerHTML = val5;
var max = Math.max(val1, val2, val3);
if (max >= 0 && max < 200) {
document.getElementById('pict').src='normal.png';
document.getElementById('desc').style.color = '#06BF51';
document.getElementById('desc').innerHTML = 'Допустимий вміст газу в повітрі.';}
if (max >= 200 && max < 300) {
document.getElementById('pict').src='middle.png';
document.getElementById('desc').style.color = '#4855B7';
document.getElementById('desc').innerHTML = 'Підвищена концентрація газу в повітрі.';}
if (max >= 300 && max <= 10000) {
document.getElementById('pict').src='high.png';
document.getElementById('desc').style.color = '#EE2600';
document.getElementById('desc').innerHTML = 'Занадто висока концентрація газу в
повітрі!';}}
</script>
</html>

```

File «settings.html»

```

<!DOCTYPE html>
<html>
<head>

```

```

<title>MQ-2 ThingSpeak налаштування</title>
<meta charset="UTF-8">
<meta name="viewport" content="width = device-width, initial-scale = 1.0,
maximum-scale = 1.0">
<style>
body { background-color: #FFFFFF; margin-top: 0; margin-bottom: 0;
margin-left: 0; margin-right: 0; }
.container { width: 20rem; margin-left: auto; margin-right: auto;
margin-top: 0; margin-bottom: 0; }
.header { margin-top: 0; margin-bottom: 0; }
.title { text-align: center; margin-top: 1rem; margin-bottom: 2rem; }
.tname { margin-top: 2rem; margin-left: 1rem; }
.checkbox { position: relative; padding-left: 25px; }
.checkbox input[type=checkbox] { display: none; }
.checkbox label:after { content: ''; display: block; height: 14px; width: 14px;
outline: 1px solid #000; position: absolute; top: 0; left: 0; }
.checkbox input[type=checkbox]:checked + label:after { outline: 1px solid #000;
border: 2px solid #fff; width: 10px; height: 10px; background-color: #000; }
.hint { margin-top: 0.5rem; margin-bottom: 0; margin-left: 1rem;
margin-right: 1rem; color: #606060; text-align: justify; }
.icon { margin-bottom: 1rem; margin-left: 1rem; margin-right: 1rem;
float: right; outline: none; text-decoration: none; }
.icon img { border: none; }
.icon:hover { text-decoration: none; outline: none; float: right; }
input[type=text] { margin-top: 1rem; width: 18rem; border: 0.05rem solid #000;
border-radius: 0.5rem; padding: 0.7rem 1rem; outline:none; }
input[type=text]:focus { border: 0.05rem dashed #000; }
input[type=button] { display: block; width: 40%; height: 2.8rem; padding: 0.5rem;
border: 0.3rem solid #000; border-radius: 0.5rem; background-color: #F7F9FD;
font-family: 'Montserrat', sans-serif; font-size: 11pt; font-weight: bold;
letter-spacing: 0.2rem; text-align: center; text-decoration: none; outline: none;
float: left; margin-bottom: 1rem; }
input[type=button]:active { border-color: #1A1A1A; background-color: #1A1A1A;
color: white; }
</style>
</head>
<body onload="loadConfig();">
<div class="container">

<h2 class="title"><b>ThingSpeak налаштування</b></h2>
<div class="checkbox tname">

```

```

<input onclick="saveStatus();" type="checkbox" id="check"><label for="check">Вмикнути
надсилання даних</label>
</div>
<p class="tname">Write API Key:</p>
<p class="hint">Введіть ключ доступу для запису. Максимум 20 символів.</p>
<input type="text" id="key" placeholder="write api key" style="text-transform:
uppercase;">
<p class="tname">Channel ID:</p>
<p class="hint">Введіть номер свого каналу. Максимум 10 цифер.</p>
<input type="text" id="id" placeholder="channel id">
<p class="tname">Field number (LPG):</p>
<p class="hint">Введіть номер поля каналу для LPG. Допустимі значення від 1 до 8.</p>
<input type="text" id="field1" placeholder="field number for LPG">
<p class="tname">Field number (CO):</p>
<p class="hint">Введіть номер поля каналу для CO. Допустимі значення від 1 до 8.</p>
<input type="text" id="field2" placeholder="field number for CO">
<p class="tname">Field number (Smoke):</p>
<p class="hint">Введіть номер поля каналу для Smoke. Допустимі значення від 1 до 8.</p>
<input type="text" id="field3" placeholder="field number for Smoke">
<p class="tname">Field number (Temperature):</p>
<p class="hint">Введіть номер поля каналу для Temperature. Допустимі значення від 1 до
8.</p>
<input type="text" id="field4" placeholder="field number for Temperature">
<p class="tname">Field number (Humidity):</p>
<p class="hint">Введіть номер поля каналу для Humidity. Допустимі значення від 1 до
8.</p>
<input type="text" id="field5" placeholder="field number for Humidity">
<p class="tname">Інтервал оновлення:</p>
<p class="hint">Встановіть інтервал оновлення значень. Рекомендовані значення від 16,
в секундах.</p>
<input type="text" id="int" placeholder="інтервал оновлення">
<br><br><br>
<input onclick="saveConfig();" type="button" id="sendbtn" value="SUBMIT">
<a class="icon" href="/"></a>
<a class="icon" href="info.html"></a>
<p class="hint" style="margin-top: 5rem;"><i>Інструкцію про початок роботи з сервісом
ThingSpeak можна знайти за <a href="https://www.mathworks.com/help/thingspeak/getting-
started-with-thingspeak.html">посиланням</i></a>.</p>
<p class="hint" style="margin-bottom: 1rem;"><i>Для того щоб заборонити відправку даних
конкретного поля, вкажіть номер даного поля -1.</i></p>

```

```


</div>
</body>
<script>
function loadConfig() {
var xhr = new XMLHttpRequest();
xhr.open("POST", '/loadConfig');
xhr.send(null);
xhr.onload = function() {
if (xhr.status == 200) {
var json = JSON.parse(xhr.response);
if (json.state == "1") {
document.getElementById('check').checked = true;
document.getElementById('key').value = json.key;
document.getElementById('id').value = json.id;
document.getElementById('field1').value = json.field1;
document.getElementById('field2').value = json.field2;
document.getElementById('field3').value = json.field3;
document.getElementById('field4').value = json.field4;
document.getElementById('field5').value = json.field5;
document.getElementById('int').value = json.int;}}}
function saveStatus() {
var state;
var xhr = new XMLHttpRequest();
if (document.getElementById('check').checked) {
state = '1';
} else {
state = '0';}
xhr.open("POST", '/saveStatus?state=' + state);
xhr.send();
xhr.onload = function() {
if (xhr.status == 200) {
alert('Зміни збережено.');}}}
function saveConfig() {
var xhr = new XMLHttpRequest();
var params = 'key=' + key.value + '&id=' + id.value + '&field1=' + field1.value
+ '&field2=' + field2.value + '&field3=' + field3.value + '&field4=' + field4.value
+ '&field5=' + field5.value + '&int=' + int.value;
xhr.open("POST", '/saveConfig?' + params);
xhr.send();

```

```
xhr.onload = function() {
  if (xhr.status == 200) {
    alert('Зміни збережено.');}}
</script>
</html>
```

File «info.html»

```
<!DOCTYPE html>
<html>
<head>
<title>DHT-11, MQ-2 Інформація</title>
<meta charset="UTF-8">
<meta name="viewport" content="width = device-width, initial-scale = 1.0,
maximum-scale = 1.0">
<style>
body { background-color: #FFFFFF; margin-top: 0; margin-bottom: 0;
margin-left: 0; margin-right: 0; }
.container { width: 20rem; margin-left: auto; margin-right: auto;
margin-top: 0; margin-bottom: 0; }
.header { margin-top: 0; margin-bottom: 0; }
.title { text-align: center; margin-top: 1rem; margin-bottom: 2rem; }
.tname { margin-top: 2rem; text-align: center; }
.hint { margin-top: 0.5rem; margin-bottom: 0;
margin-left: 1rem; margin-right: 1rem; }
.icon { margin-top: 2rem; margin-left: 1rem; margin-right: 1rem;
float: right; outline: none; text-decoration: none; }
.icon img { border: none; }
.icon:hover {text-decoration: none; outline: none; float: right; }
input[type=text] { width: 18rem; border: 0.05rem solid #000;
border-radius: 0.5rem; padding: 0.7rem 1rem; outline:none; }
input[type=text]:focus { border: 0.05rem dashed #000; }
input[type=button] { margin-left: auto; margin-right: auto; display: block;
width: 40%; padding: 0.5rem; border: 0.1rem solid #000; border-radius: 0.5rem;
background-color: #F7F9FD; font-family: 'Montserrat', sans-serif;
font-size: 11pt; font-weight: bold; letter-spacing: 0.2rem; text-align: center;
text-decoration: none; outline: none; }
input[type=button]:active { border-color: #1A1A1A; background-color: #1A1A1A;
color: white; }
</style>
</head>
<body onload="loadDevice();">
<div class="container">
```

```


<h2 class="title"><b>Інформація</b></h2>
<p class="tname">Інформація про пристрій</p>
<p class="hint">IP адреса: <span id='ip' style="color: #606060;">*</span>.</p>
<p class="hint">MAC адреса: <span id='mac' style="color: #606060;">*</span>.</p>
<p class="hint">Час роботи: <span id='uptime' style="color: #606060;">*</span> хв.</p>
<p class="tname">Інформація про Thingspeak</p>
<p class="hint">Надсилання даних: <span id='state' style="color:
#606060;">*</span>.</p>
<p class="hint">Код останньої відповіді: <span id='code' style="color:
#606060;">*</span>.</p>
<p class="hint" style="color: #606060; text-align: justify;"><i>Детальніше про код
відповіді можна дізнатися за <a href="https://github.com/mathworks/thingspeak-
arduino#return-codes">посиланням</a>.</i></p>
<p class="tname">Перезавантажити пристрій:</p>
<input onclick="reboot();" type="button" id="sendbtn" value="REBOOT">
<p class="tname">Відновити заводські налаштування:</p>
<input onclick="reset();" type="button" id="sendbtn" value="RESET">
<a class="icon" href="/"></a>
<a class="icon" href="settings.html"></a>

</div>
</body>
<script>
function loadDevice() {
var xhr = new XMLHttpRequest();
xhr.open("POST", '/loadDevice');
xhr.send(null);
xhr.onload = function() {
if (xhr.status == 200) {
var json = JSON.parse(xhr.response);
document.getElementById('ip').innerHTML = json.ip;
document.getElementById('mac').innerHTML = json.mac;
document.getElementById('uptime').innerHTML = Math.floor(json.uptime / 60000);
if (json.state == "1") {
document.getElementById('state').style.color = '#21C49C';
document.getElementById('state').innerHTML = 'Увімкнено';
document.getElementById('code').innerHTML = json.code;
if (json.code == "200") {

```

```

document.getElementById('code').style.color = '#21C49C'; }
else {
document.getElementById('code').style.color = '#DF3D2E';}}
else {
document.getElementById('state').style.color = '#606060';
document.getElementById('state').innerHTML = 'Вимкнено';
document.getElementById('code').innerHTML = 'Не активно';}}}}
function reboot() {
var result = window.confirm('Перезавантажити пристрій?');
if (result) {
var xhr = new XMLHttpRequest();
xhr.open("POST", '/reboot');
xhr.send();
alert('Пристрій перезавантажиться через декілька секунд.')}}
function reset() {
var result = window.confirm('Відновити заводські налаштування?');
if (result) {
var xhr = new XMLHttpRequest();
xhr.open("POST", '/reset');
xhr.send();
alert('Заводські налаштування відновлено. Здійсніть підключення пристрою заново.')}}
</script>
</html>

```

Додаток В

Отримані заміри дослідження				
Дата	Час	СО	Температура	Вологість
24.11.2023	13:34:47	21,0	19,9	54,0
24.11.2023	13:44:47	21,0	19,9	54,0
24.11.2023	13:54:47	26,0	19,8	54,0
24.11.2023	14:04:47	26,0	19,8	53,0
24.11.2023	14:14:47	26,0	19,7	54,0
24.11.2023	14:24:47	26,0	19,8	54,0
24.11.2023	14:34:48	26,0	19,7	54,0
24.11.2023	14:44:48	25,0	19,6	53,0
24.11.2023	14:54:47	25,0	19,6	54,0
24.11.2023	15:04:48	25,0	19,6	54,0
24.11.2023	15:14:48	25,0	19,5	54,0
24.11.2023	15:24:48	25,0	19,5	54,0
24.11.2023	15:34:48	25,0	19,5	54,0
24.11.2023	15:44:48	25,0	19,5	54,0
24.11.2023	15:54:48	24,0	19,4	54,0
24.11.2023	16:04:48	24,0	19,4	54,0
24.11.2023	16:14:48	25,0	19,2	54,0
24.11.2023	16:24:48	24,0	19,1	54,0
24.11.2023	16:34:48	24,0	19,1	54,0
24.11.2023	16:44:48	24,0	19,3	54,0
24.11.2023	16:54:48	24,0	19,7	54,0
24.11.2023	17:04:48	24,0	19,8	54,0
24.11.2023	17:14:48	24,0	20,0	54,0
24.11.2023	17:24:48	24,0	20,1	54,0
24.11.2023	17:34:48	24,0	20,2	54,0
24.11.2023	17:44:48	24,0	20,4	53,0
24.11.2023	17:54:48	24,0	20,4	53,0
24.11.2023	18:04:48	23,0	20,5	53,0
24.11.2023	18:14:48	24,0	20,6	53,0
24.11.2023	18:24:48	24,0	20,6	53,0
24.11.2023	18:34:48	23,0	20,6	53,0
24.11.2023	18:44:48	23,0	20,4	53,0
24.11.2023	18:54:48	24,0	20,4	53,0
24.11.2023	19:04:48	24,0	20,4	53,0

24.11.2023	19:14:48	24,0	20,2	53,0
24.11.2023	19:24:48	24,0	20,4	53,0
24.11.2023	19:34:48	24,0	20,6	53,0
24.11.2023	19:44:48	24,0	20,7	53,0
24.11.2023	19:54:48	24,0	20,7	53,0
24.11.2023	20:04:48	24,0	20,8	53,0
24.11.2023	20:14:48	24,0	20,8	53,0
24.11.2023	20:24:48	24,0	20,9	54,0
24.11.2023	20:34:48	25,0	21,0	55,0
24.11.2023	20:44:48	25,0	21,1	54,0
24.11.2023	20:54:48	25,0	21,1	54,0
24.11.2023	21:04:48	25,0	21,0	54,0
24.11.2023	21:14:48	25,0	20,9	53,0
24.11.2023	21:24:48	24,0	20,7	52,0
24.11.2023	21:34:49	25,0	20,7	52,0
24.11.2023	21:44:49	24,0	20,5	52,0
24.11.2023	21:54:49	24,0	20,4	52,0
24.11.2023	22:04:49	24,0	20,5	53,0
24.11.2023	22:14:49	25,0	20,3	53,0
24.11.2023	22:24:49	25,0	20,3	54,0
24.11.2023	22:34:49	24,0	20,3	54,0
24.11.2023	22:44:49	24,0	20,4	53,0
24.11.2023	22:54:49	24,0	20,4	54,0
24.11.2023	23:04:49	24,0	20,4	54,0
24.11.2023	23:14:49	24,0	20,4	54,0
24.11.2023	23:24:49	24,0	20,3	54,0
24.11.2023	23:34:49	24,0	20,3	55,0
24.11.2023	23:44:49	24,0	20,1	54,0
24.11.2023	23:54:49	24,0	19,9	54,0
25.11.2023	0:04:49	24,0	19,8	54,0
25.11.2023	0:14:49	24,0	19,8	54,0
25.11.2023	0:24:49	24,0	19,7	54,0
25.11.2023	0:34:49	24,0	19,7	54,0
25.11.2023	0:44:49	24,0	19,7	54,0
25.11.2023	0:54:49	24,0	19,7	53,0
25.11.2023	1:04:49	24,0	19,7	53,0
25.11.2023	1:14:49	24,0	19,7	53,0

25.11.2023	1:24:49	24,0	19,7	53,0
25.11.2023	1:34:49	24,0	19,6	53,0
25.11.2023	1:44:49	24,0	19,7	53,0
25.11.2023	1:54:49	24,0	19,7	53,0
25.11.2023	2:04:49	24,0	19,7	53,0
25.11.2023	2:14:49	24,0	19,7	53,0
25.11.2023	2:24:49	24,0	19,7	52,0
25.11.2023	2:34:49	24,0	19,7	52,0
25.11.2023	2:44:49	24,0	19,6	52,0
25.11.2023	2:54:49	24,0	19,7	52,0
25.11.2023	3:04:49	24,0	19,6	52,0
25.11.2023	3:14:49	24,0	19,6	52,0
25.11.2023	3:24:49	24,0	19,6	52,0
25.11.2023	3:34:49	24,0	19,6	52,0
25.11.2023	3:44:50	24,0	19,6	52,0
25.11.2023	3:54:49	24,0	19,6	52,0
25.11.2023	4:04:50	24,0	19,6	52,0
25.11.2023	4:14:50	23,0	19,6	52,0
25.11.2023	4:24:50	23,0	19,6	52,0
25.11.2023	4:34:50	23,0	19,6	51,0
25.11.2023	4:44:50	23,0	19,5	52,0
25.11.2023	4:54:50	24,0	19,6	52,0
25.11.2023	5:04:50	24,0	19,5	52,0
25.11.2023	5:14:50	23,0	19,5	52,0
25.11.2023	5:24:50	23,0	19,5	52,0
25.11.2023	5:34:50	24,0	19,5	52,0
25.11.2023	5:44:50	23,0	19,5	52,0
25.11.2023	5:54:50	23,0	19,5	52,0
25.11.2023	6:04:50	23,0	19,5	52,0
25.11.2023	6:14:50	23,0	19,6	52,0
25.11.2023	6:24:50	22,0	19,5	52,0
25.11.2023	6:34:50	23,0	19,5	52,0
25.11.2023	6:44:50	24,0	19,5	52,0
25.11.2023	6:54:50	23,0	19,5	52,0
25.11.2023	7:04:50	23,0	19,4	52,0
25.11.2023	7:14:50	23,0	19,5	52,0
25.11.2023	7:24:50	23,0	19,5	53,0

25.11.2023	7:34:50	23,0	19,5	52,0
25.11.2023	7:44:50	23,0	19,5	52,0
25.11.2023	7:54:50	23,0	19,5	52,0
25.11.2023	8:04:50	23,0	19,4	52,0
25.11.2023	8:14:50	23,0	19,5	53,0
25.11.2023	8:24:50	23,0	19,5	52,0
25.11.2023	8:34:50	23,0	19,5	52,0
25.11.2023	8:44:50	23,0	19,5	53,0
25.11.2023	8:54:50	23,0	19,5	52,0
25.11.2023	9:04:50	23,0	19,4	53,0
25.11.2023	9:14:50	23,0	19,3	53,0
25.11.2023	9:24:50	23,0	19,4	53,0
25.11.2023	9:34:50	23,0	19,5	53,0
25.11.2023	9:44:50	22,0	19,5	52,0
25.11.2023	9:54:50	24,0	19,5	52,0
25.11.2023	10:04:50	23,0	19,5	53,0
25.11.2023	10:14:50	23,0	19,6	52,0
25.11.2023	10:24:50	24,0	19,5	52,0
25.11.2023	10:34:51	24,0	19,6	52,0
25.11.2023	10:44:51	24,0	19,5	53,0
25.11.2023	10:54:51	24,0	19,6	53,0
25.11.2023	11:04:51	24,0	19,6	54,0
25.11.2023	11:14:53	24,0	19,7	54,0
25.11.2023	11:24:51	25,0	19,8	55,0
25.11.2023	11:34:51	24,0	20,0	55,0
25.11.2023	11:44:51	23,0	20,0	54,0
25.11.2023	11:54:51	23,0	20,3	54,0
25.11.2023	12:04:51	23,0	20,4	54,0
25.11.2023	12:14:51	23,0	20,5	54,0
25.11.2023	12:24:51	24,0	20,5	54,0
25.11.2023	12:34:51	24,0	20,5	54,0
25.11.2023	12:44:51	24,0	20,6	54,0
25.11.2023	12:54:51	22,0	20,7	54,0
25.11.2023	13:04:51	22,0	20,8	54,0
25.11.2023	13:14:51	22,0	20,8	54,0
25.11.2023	13:24:51	22,0	20,9	54,0