ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Системи моніторингу якості повітря»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 - Комп'ютерна інженерія
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Євгеній ПАВЛІЧЕНКО*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КІД-41
Євгеній ПАВЛІЧЕНКО
Ім'я, ПРІЗВИЩЕ

Керівник: Юлія БОРОВА
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 - Комп'ютерна інженерія

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерної інженерії
_____Наталія Лащевська
_____
«_____» _____ 2025р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Павліченко Євгеній Володимирович

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Системи моніторингу якості повітря»

керівник кваліфікаційної роботи: Борова Юлія Олександрівна, асистент кафедри

(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025р. №56

2. Строк подання кваліфікаційної роботи «30» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: Науково-технічна література, інтернет-ресурси щодо протоколів HTTP, програмне забезпечення: Arduino IDE, XAMPP, HTML/CSS/JS, середовище для зберігання даних: MySQL, веб-інтерфейс для візуалізації даних, модуль NodeMCU ESP8266

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. 1 Теоретичні основи моніторингу довкілля засобами IOT
2. Проектування та розробка системи моніторингу якості повітря
3. Експериментальні результати та практичне застосування системи

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	25.02 – 28.02.2025	
2	Обробка матеріалу	28.02 – 03.03.2025	
3	Розробка теоретичної частини	03.03 – 07.03.2025	
4	Розробка проекту	07.03 – 13.03.2025	
5	Висновки за результатами аналізу	20.05 – 23.05.2025	
6	Оформлення пояснювальної записки	23.05 – 25.05.2025	
8	Розробка презентації	26.05 – 27.05.2025	
9	Передзахист	27.05 – 29.05.2025	
10	Здача в деканат	20.06 – 20.06.2025	

Здобувач вищої освіти _____ Євгеній ПАВЛІЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Юлія БОРОВА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 55 стор., 39 рис., 25 джерел.

Мета роботи – розробка доступної, ефективної та адаптивної IoT-системи моніторингу якості повітря, яка дозволяє у режимі реального часу відстежувати параметри навколишнього середовища.

Об'єкт дослідження – процес моніторингу параметрів повітря.

Предмет дослідження – IoT-система збору, зберігання та візуалізації параметрів якості повітря на базі Arduino та ESP8266.

Короткий зміст роботи: У дипломній роботі описано процес розробки системи моніторингу якості повітря на основі IoT. Визначено апаратні компоненти (Arduino Mega, ESP8266, датчики MQ-2, MQ-135, DHT11 тощо), представлено схему підключення і програмний код. Створено веб-інтерфейс для візуалізації даних у реальному часі та історичному вигляді. Проведено тестування системи, розглянуто переваги, недоліки та можливості масштабування. Робота ілюструє практичну цінність недорогих рішень для екологічного моніторингу.

КЛЮЧОВІ СЛОВА: IOT, МОНІТОРИНГ, ЯКІСТЬ ПОВІТРЯ, ARDUINO, ДАТЧИКИ, ESP8266, ВІЗУАЛІЗАЦІЯ ДАНИХ, ЕКОЛОГІЯ.

ЗМІСТ

ВСТУП……. 9
1. ТЕОРЕТИЧНІ ОСНОВИ МОНІТОРИНГУ ДОВКІЛЛЯ ЗАСОБАМИ IOT ….. 11
1.1 Історія та еволюція Інтернету речей ……11
1.2 Основні концепції loT …….. 12
1.2.1 Багаторівнева модель loT…….....13
1.3 Інтероперабельність…….. 16
1.3.1 Типи інтероперабельності……..... 17
1.4 Веб-сервіси ……..... 18
1.5 Технології публікації-передплати та обміну повідомленнями…..……20
2. ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ЯКОСТІ ПОВІТРЯ…… 22
2.1 Апаратне та програмне забезпечення, що використовується в loT ……22
2.1.1 Датчики …… 22
2.1.2 Інтерфейси між датчиками та мікроконтролерами/пристроями……..25
2.1.3 Приводи…… 26
2.1.4 Протоколи зв'язку……..... 27
2.2 Розробка проекту……..…….28
2.2.1 Збір даних……..28
2.2.1.1 Конфігурація та вибір компонентів……..…….28
2.2.1.2 Електрична схема…….……..36
2.2.2 Зберігання даних……...…….37
2.2.2.1 Створення бази даних…….……38
2.2.3 Візуалізація даних…….…….39
3. ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ ТА ПРАКТИЧНЕ ЗАСТОСУВАННЯ СИСТЕМИ……..44
3.1 Результати інтерфейсу веб-сайту……...44
3.2 Збір, зберігання та візуалізація результатів……...50
3.3 Практичні застосування…….…..52

3.3.1 Практичні застосування в реальному світі …….. 52
3.3.2 Техніко-економічне обґрунтування …… 53
3.4 Безпека та конфіденційність IoT ……. 54
ВИСНОВКИ…….. 56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……...……...58
Додаток А. ……..….....61
Додаток Б. ……..….....65
Додаток В. ……..….....66
Додаток Г. ……..….....69
Додаток Ґ. ……..….....70
Додаток Д. ……..….....73
Додаток Е. ……..….....75
Додаток Є. ……..….....77
Додаток Ж. ……..……..….....78
Додаток З…….….....79
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ…….86

ВСТУП

У сучасному світі, де екологічна ситуація постійно змінюється, актуальним стає впровадження систем моніторингу, що дозволяють відстежувати параметри довкілля в режимі реального часу. Зокрема, якість повітря є критичним показником здоров'я людей і стану навколишнього середовища. Підвищення рівня урбанізації, зростання промислових викидів та чисельності транспорту вимагають запровадження доступних та ефективних інструментів для контролю атмосферного повітря.

Завдяки технологіям Інтернету речей (IoT) стало можливим створювати системи збору, передачі та обробки даних з навколишнього середовища з мінімальними витратами. Ці системи мають гнучку архітектуру, масштабованість і здатні працювати в автономному режимі. Розробка системи моніторингу на основі Arduino Mega та NodeMCU ESP8266 з використанням таких датчиків, як MQ-135 (якість повітря), DHT11 (температура і вологість), BMP180 (атмосферний тиск) та інших, дозволяє отримати комплексну картину екологічного стану.

Дана робота спрямована на дослідження, реалізацію та тестування прототипу IoT-системи, здатної вимірювати ключові параметри повітря і передавати їх на локальний веб-сервер для подальшої обробки і аналізу. Така система може бути корисною як для державних служб екологічного нагляду, так і для приватного користувача, який бажає контролювати якість повітря у своїй місцевості.

Описуються вибір апаратного і програмного забезпечення, методи зберігання даних у базі MySQL, а також створення інтуїтивного веб-інтерфейсу для візуалізації та доступу до історичних даних. Робота передбачає не лише технічну реалізацію, а й аналіз отриманих результатів, виявлення обмежень і формулювання пропозицій щодо вдосконалення.

Система використовує плату Arduino Mega для збору даних з декількох датчиків навколишнього середовища, включаючи MQ-2, MQ-135, DHT11, KY-037 і BMP180, для вимірювання якості повітря, температури, вологості, звуку та

атмосферного тиску. Зібрані дані обробляються і надсилаються до вузлового мікроконтролера ESP8266, який передає їх на веб-платформу для візуалізації. Для цього була реалізована мережева архітектура loT, де Arduino Mega збирає дані з датчиків, а Node MCU ESP8266 відповідає за зв'язок з сервером. Зв'язок між пристроями здійснюється за допомогою послідовного зв'язку. Розроблено веб-інтерфейс з використанням HTML, CSS та JavaScript у поєднанні з XAMPP для відображення зібраних даних у реальному часі.

Результати експериментів показують, що система здатна ефективно збирати і передавати дані, забезпечуючи екологічний моніторинг завдяки інтуїтивно зрозумілому інтерфейсу, який дозволяє користувачам отримувати доступ до даних відповідно до обраного дня. Ця можливість доступу до даних дає детальне уявлення про стан довкілля в минулому, полегшуючи порівняльний аналіз і відстеження закономірностей у часі.

Таким чином, метою цієї дипломної роботи є демонстрація практичного застосування IoT у сфері екологічного моніторингу, створення економічно ефективної системи, яка може бути адаптована до різних потреб і масштабів.

1 ТЕОРЕТИЧНІ ОСНОВИ МОНІТОРИНГУ ДОВКІЛЛЯ ЗАСОБАМИ IOT

1.1 Історія та еволюція Інтернету речей

Перші концепції об'єднання інтелектуальних пристроїв в мережу обговорювалися в 1970-х і 1980-х роках. У 1874 році на горі Монблан було встановлено кілька метеорологічних пристроїв для збору даних і передачі їх по радіоканалу. А ще раніше, у 1830 році, за допомогою зв'язку телеграфів між собою. Можливо, це був один з перших кроків на шляху до зародження Інтернету речей.

У 1990-х роках було опубліковано кілька наукових праць, які популяризували цей термін. Наприкінці 1990-х років термін "Інтернет речей" був запропонований Кевіном Ештоном під час роботи над проектом радіочастотної ідентифікації (RFID) в Массачусетському технологічному інституті (MIT). У цій ранній системі кожна мітка містила спрощений мікрочіп, який зберігав лише основний серійний номер (для зменшення витрат), які можна було передавати локально за допомогою бездротового зв'язку. Дані, пов'язані з серійним номером мітки, зберігалися окремо в базі даних, доступній онлайн. Ештон ввів цю фразу щоб проілюструвати, як об'єкти фізичного світу можуть підключатися до Інтернету і спілкуватися один з одним.

У 2000-х роках нові концепції та технології уможливили створення бездротових мереж датчиків/актуаторів (WSAN, або WSN): крихітні комп'ютери на батарейках спочатку співпрацюють для створення бездротових мереж, а потім використовують такі мережі для передачі даних своїх датчиків або для розподілу команд виконавчих механізмів. Узагальнюючи подібні концепції, було введено термін "повсюдні обчислення" (дещо схожий на ubiquitous computing), щоб відобразити тенденцію вбудовування певних обчислювальних і комунікаційних можливостей у повсякденні об'єкти.

З цього моменту концепція Інтернету речей та її розвиток були пов'язані з впровадженням датчиків і з'єднань у будь-який тип пристрою, який міг би його підтримувати. По суті, мова йшла про вбудовування "інтелекту" в різні електронні об'єкти, які могли його підтримувати.

1.2 Основні концепції IoT

Інтернет речей визначається як взаємозв'язок фізичних пристроїв, повсякденних об'єктів і систем через мережу, що дозволяє здійснювати віддалений і автоматизований збір, обмін і аналіз даних. Хоча IoT - це трендова технологія з необмеженим майбутнім потенціалом, поки що не існує єдиного або конкретного набору правил для компонентів IoT. Кілька організацій впровадили функції IoT на своїх об'єктах, оскільки їхня інтеграція дала їм різні переваги.

Якщо говорити про систему взаємопов'язаних об'єктів, підключених до Інтернету, які можуть збирати та передавати дані через бездротову мережу без участі людини, то особисті та комерційні можливості безмежні. Під "річчю" може матися на увазі підключений медичний пристрій, біочіп-транспондер, сонячна панель, підключений автомобіль з датчиками, які попереджають водія про безліч потенційних проблем (паливо, тиск у шинах, необхідність технічного обслуговування тощо) або будь-який об'єкт, оснащений датчиками, який має здатність збирати і передавати дані через мережу.

Існує багато аспектів, які складають ці типи систем, наприклад, рівні, додатки та компоненти мережевих технологій. Тому різноманітність таких систем дуже велика. Однак є певні основні елементи, які складають кожну систему і дозволяють їм працювати разом для забезпечення зв'язку, комунікації та збору даних. Нижче ми розглянемо деякі з найбільш важливі аспекти, з яких складаються ці проекти.

1.2.1 Багаторівнева модель IoT

Подібно до того, як в інших технологіях робиться архітектурний поділ для представлення моделі функціональних шарів, в IoT також можна диференціювати шари, в яких різні види діяльності згруповані разом, що представлено на рисунку 1.1. Далі описано кожен з шарів зображеної п'ятишарової моделі та основні функції, які вони виконують.



Рисунок 1.1 - Багаторівнева модель IoT

а) рівень сприйняття: На цьому рівні формуються фізичні компоненти розробки Інтернету IoT, які виступають в якості середовища між цифровим світом і реальним світом. Його основне призначення - перетворення аналогових сигналів у цифрову форму і навпаки. Функцію збору інформації з різних пристроїв та її обробки виконує безліч різних за формою і розміром пристроїв. Ось деякі з них і найбільш відомі:

1) датчики: це дуже маленькі пристрої або системи, створені для розуміння та виявлення змін у навколишньому середовищі та подальшої оптимізації

інформації для вашої системи. Як правило, ці датчики досить малі і потребують ще менше енергії для виконання свого завдання. Датчики мають унікальну здатність виявляти фізичні параметри, такі як вологість або температура, а потім перетворювати їх на електронні сигнали;

2) актуатори: Являють собою частину машини, яка дозволяє перетворювати електричний сигнал у фізичні дії. Ці приводи відіграють вирішальну роль як компоненти мереж loT;

3) машини та пристрої: це основні пристрої, які мають актуатори та датчики.

б) комунікаційний рівень: займається передачею даних від об'єкта до рівня підтримки. Цей рівень повинен реалізовувати функціональні можливості:

1) абстракція зв'язку, щоб інкапсулювати деталі базових комунікаційних технологій, використовуючи одну або декілька загальних точок доступу для надання набору послуг зв'язку;

2) функції зв'язку, що забезпечують незалежність від технології доступу, для чого повинні бути доступні спеціальні роз'єми з основними протоколами, що використовуються в світі loT;

3) функції кібербезпеки: Використовуються безпечні механізми зв'язку та управління.

Зв'язок відбувається двома методами:

- по-перше, безпосередньо через стек TCP або UDP/IP;

- по-друге, шлюзи діють як сполучна ланка між локальною мережею (LAN) і глобальною мережею (WAN), забезпечуючи таким чином шлях для проходження інформації через кілька протоколів.

Існує безліч мережевих технологій, включаючи Wi-Fi, Ethernet, Bluetooth, NFC, LPWAN, Zigssee і мобільні мережі, такі як LTE-M і NB-loT. Кожна технологія висвітлюється з точки зору її діапазону, енергоспоживання, пропускної здатності та основних застосувань.

Також доступні протоколи обміну повідомленнями, такі як DDS, AMQP, COAP і MQTT, які використовуються для полегшення зв'язку між пристроями в

системі loT. Кожен протокол має свій власний підхід і переваги, підходить для пристроїв з різними вимогами до потужності, пам'яті та радіусу дії мережі.

в) рівень підтримки: його основним завданням є управління послугами першого рівня. На ранніх етапах, коли loT-мережі набирали розмір і кількість, затримка ставала однією з головних перешкод. І коли кілька пристроїв намагалися підключитися до головного концентратора, це забивало систему і сповільнювало процедуру. Тут периферійні обчислення запропонували унікальне рішення, яке прискорило зростання систем loT в цілому. Тепер, з периферійними рівнями loT, системи може обробляти та аналізувати інформацію якомога ближче до джерела. Границя стала стандартом для п'ятого покоління мобільних мереж (5G), пропонуючи системам підключення до більшої кількості пристроїв з меншою затримкою, ніж у домінуючих стандартах 4G. Всі процедури для мереж loT виконуються на межі. Це економить час, ресурси, а також забезпечує зворотний зв'язок в режимі реального часу і підвищує продуктивність.

Цей рівень дозволяє розробникам рішень loT абстрагуватися від апаратної платформи. Функції включають в себе наступне:

1) управління пристроями, яке включає в себе повноваження зберігання необхідних характеристик, що визначають можливості пристрою, забезпечення і розгортання, управління активами і контроль, а також моніторинг і діагностику;

2) безпечне управління доступом на основі визначених ролей;

3) моніторинг системних подій (журналів);

г) рівень обробки: На рівні обробки виділяють два основних етапи;

1) накопичення даних: Кожен пристрій надсилає мільйони потоків даних через мережу loT. Тут дані мають різну форму, швидкість і розмір. Відокремлення важливих даних від цих великих потоків є першочерговим завданням, яке фахівці повинні вирішувати на цьому рівні;

2) абстрагування даних: Після завершення етапу накопичення даних, вибрані дані абстрагуються від великих обсягів даних для оптимізації бізнес-процедур програми. Тут абстрагування даних відбувається наступним чином:

3) збір всіх даних з усіх loT і не loT систем (CRM, ERP і ERM).

4) використання віртуалізації даних, щоб зробити їх доступними з одного місця;

5) керування необробленими даними в різних формах.

Після того, як накопичення та абстрагування даних завершено, аналітикам даних легко використовувати ділову хватку для виведення інтелектуальних факторів.

г) бізнес-рівень: Використовується для надання послуг, які запитує клієнт. Цей рівень містить функції управління додатками та специфічні сервіси, які взаємодіють з сервісами, представленими на попередньому рівні. Таким чином, створення різних бізнес-вертикалей припадає на цей рівень. Кожна система loT будується зі своїми конкретними цілями і завданнями, щоб відповідати бізнес-специфікаціям. Сьогодні більшість додатків loT працюють з різною складністю і використовують безліч технологічних стеків, які виконують конкретні завдання для бізнесу. Деякі приклади прикладного рівня включають:

1) програмне забезпечення для прийняття бізнес-рішень;

2) послуги моніторингу та управління пристроями;

3) аналітичні рішення, побудовані на основі машинного навчання та штучного інтелекту;

4) мобільний додаток для подальшої взаємодії.

1.3 Інтероперабельність

Комп'ютерні системи часто є розподіленими та динамічними за своєю природою. Крім того, ці системи базуються на гетерогенних технологіях і використовують різні способи представлення інформації. Ці фактори перешкоджають обміну та обробці даних між різними суб'єктами (пристроями, машинами, контролерами, датчиками...). Тому для досягнення повної інтероперабельності інформація, якою обмінюються, повинна мати не лише спільну синтаксичну основу, а й спільну структуру та семантику.

Через це термін "інтероперабельність" використовується для тих стандартів, протоколів, технологій і механізмів, які дозволяють передавати дані між різними системами з мінімальним втручанням людини. Це дозволяє різним системам спілкуватися одна з одною та обмінюватися інформацією в режимі реального часу.

1.3.1 Типи інтероперабельності

Інтероперабельність можна класифікувати з різних точок зору. Найбільш загальна класифікація описує чотири типи:

- технічна інтероперабельність: Цей тип відноситься до здатності систем, системних компонентів або додатки для встановлення зв'язку та обміну повідомленнями, не обов'язково розуміючи їхній зміст. Отже, він не передбачає знання формату та значення даних. Зазвичай вона вимагає наявності мережевого підключення. Технічна інтероперабельність тісно пов'язана з елементами, які уможливлюють міжмашинну комунікацію (наприклад, необхідні протоколи, апаратне та програмне забезпечення);

- синтаксична інтероперабельність: Це здатність систем правильно інтерпретувати структуру повідомлень в обмінюваній інформації і, таким чином, бути здатними читати її зміст, хоча вони можуть і не знати про значення цієї інформації. Прикладом синтаксичної інтероперабельності є система розумного міста, яка отримує інформацію з центру обробки даних і здатна правильно розпізнати її специфічний формат даних (наприклад, CSV) і, таким чином, правильно витягти дані з повідомлення (наприклад, набір значень). Тим не менш, вона може не знати, що ці дані представляють (наприклад значеннями можуть бути температури), таким чином, не маючи змоги використовувати дані в правильному контексті;

- семантична інтероперабельність: Ця інтероперабельність забезпечує спільне значення і розуміння даних, якими обмінюються. Зокрема, семантична інтероперабельність гарантує, що комп'ютерні системи можуть обмінюватися даними однозначно, поділяючи значення даних. Вона також є основою для

забезпечення ефективної обчислюваної машинної логіки (наприклад, алгоритмів машинного навчання), а також для вилучення та виявлення знань. Таким чином, система стає здатною використовувати цю інформацію у відповідному контексті.

1.4 Веб-сервіси

Веб-сервіси - це програми або системи, які забезпечують зв'язок між різними пристроями через Інтернет. Ці сервіси полегшують взаємодію між різнорідними програмними системами, дозволяючи обмінюватися даними і функціями між різними платформами і мовами програмування.

Ці сервіси використовують набір стандартних веб-протоколів для забезпечення взаємодії між додатками та системами. Веб-сервіси зазвичай використовують такі стандарти, як HTTP, XML, SOAP, REST і JSON, щоб полегшити комунікацію та обмін даними між різними додатками. Існує кілька різних форм веб-сервісів, описаних нижче:

- SOAP (Simple Object Access Protocol - простий протокол доступу до об'єктів): Використовує XML для обміну інформацією та визначає жорстку структуру для комунікації між додатками. SOAP дозволяє процесам, запущеним на клієнті з різних операційних систем (наприклад, Linux або Windows), отримувати доступ до сервера через виклики веб-сервісів і отримувати відповіді незалежно від мови і платформи;

- REST (Representational State Transfer - передача представницького стану): Використовує методи HTTP (GET, POST, PUT, DELETE) для доступу та маніпулювання ресурсами в системі. Він базується на таких принципах, як ідентифікація ресурсів через URL-адреси та маніпулювання цими ресурсами за допомогою HTTP-запитів. REST забезпечує незалежність між різними платформами (Unix, Windows, Mac, ...), а також незалежність від мови програмування (C++ може спілкуватися з Java і т.д.), базується на стандартах (оскільки працює через HTTP) і може легко використовуватися при наявності брандмауерів.

На рисунку 1.2 представлено взаємодію web-сервісів.

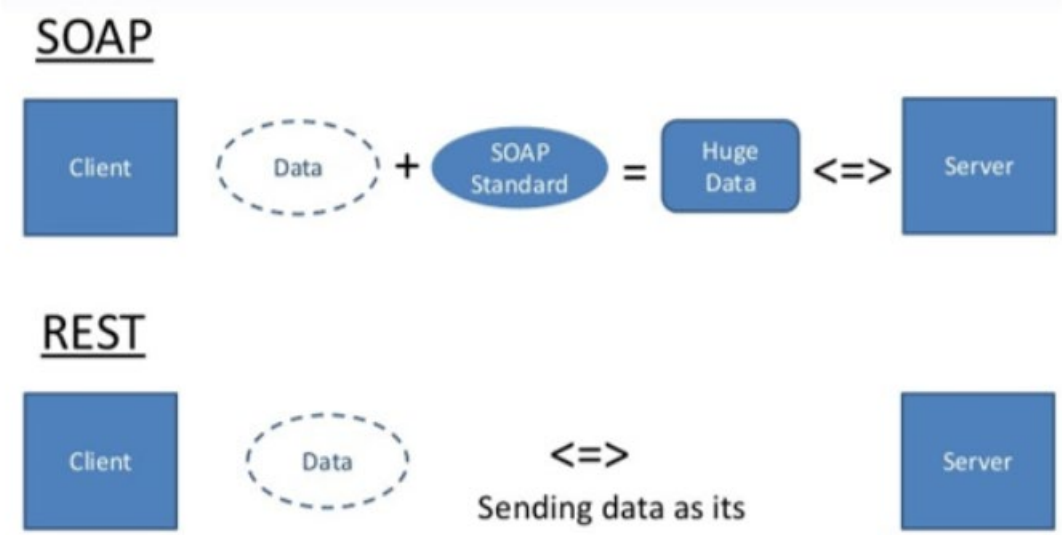


Рисунок 1.2 - Взаємодія web-сервісів

Розглянемо ключові відмінності REST API і SOAP:

- REST API взагалі не має офіційного стандарту, тому що це архітектурний стиль. SOAP API, з іншого боку, має офіційний стандарт, тому що це протокол;

- API REST використовує кілька стандартів, таких як HTTP, JSON, URL і XML, а API-інтерфейси SOAP значною мірою засновані на HTTP і XML;

- оскільки REST API розгортає кілька стандартів, він вимагає менше ресурсів і пропускної спроможності порівняно з SOAP, який використовує XML для створення корисного навантаження і призводить до файлу великого розміру;

- способи, якими обидва API розкривають бізнес-логіку, також різні. REST API використовує переваги експозиції URL, такі як @path (/ WeatherService), тоді як SOAP API використовує інтерфейси сервісів, такі як @WebService;

- SOAP API визначає занадто багато стандартів, і його розробник реалізує речі тільки стандартним способом. У разі непорозуміння від служби, результатом буде помилка. REST API, з іншого боку, не наголошує на занадто багатьох стандартах і в кінцевому підсумку призводить до пошкодження API;

- REST API використовує мову опису веб-додатків, а SOAP API використовує мову опису веб-сервісів для опису функцій, пропонованих веб-сервісами;

- API REST більш зручні з JavaScript і можуть бути легко реалізовані. API-інтерфейси SOAP також зручні при використанні JavaScript, але не підтримують велику реалізацію.

1.5 Технології публікації-передплати та обміну повідомленнями

Технології публікації-передплати та обміну повідомленнями - це комунікаційні парадигми, що використовуються в розподілених системах для полегшення передачі даних між різними компонентами або системами без їхнього безпосереднього з'єднання. Ці моделі широко використовуються в архітектурах розподіленого програмного забезпечення, а також в контексті веб-сервісів і хмарних додатків. Ось короткий опис обох моделей:

Публікація-підписка:

Концепція: У системі "публікація-передплата" компоненти або системи поділяються на дві основні ролі: видавці та передплатники. Видавці публікують події або повідомлення (публікують), пов'язані з певними темами. Підписники виявляють інтерес до певних тем і отримують сповіщення, коли відбуваються пов'язані з ними події.

Переваги:

- роз'єднання: компоненти не повинні знати один одного безпосередньо. Видавцеві не потрібно знати, хто є передплатниками, і навпаки.

- масштабованість: Нові компоненти можна додавати, не впливаючи на існуючі;

- гнучкість: Компоненти можуть взаємодіяти асинхронно.

Обмін повідомленнями:

Концепція: У системі обміну повідомленнями компоненти взаємодіють, надсилаючи та отримуючи повідомлення. Повідомлення можуть містити дані, інструкції або події і передаються між компонентами системи.

Моделі:

- синхронна: комунікація відбувається в режимі реального часу, і відправник чекає на підтвердження від одержувача;

- асинхронна: комунікація не блокує відправника. Він може продовжувати виконувати інші завдання, чекаючи, поки одержувач обробить повідомлення.

- точка-точка: Компонент надсилає повідомлення безпосередньо іншому компоненту;

- публікація-підписка: Схожий на концепцію публікації-підписки, але тут повідомлення транслюються всім зацікавленим підписникам.

Переваги:

- розділення: Компоненти можуть працювати незалежно;

- масштабованість: Нові компоненти можна додавати, не впливаючи на загальну систему;

- надійність: повідомлення можна зберігати та повторно надсилати у разі тимчасових збоїв.

Обидві парадигми забезпечують гнучкість і масштабованість у розподілених середовищах, дозволяючи системам розвиватися і розширюватися більш ефективно. Вибір між ними буде залежати від конкретних вимог програми та архітектури системи. Ось деякі з найбільш актуальних протоколів обміну повідомленнями:

- Message Queuing Telemetry Transport (MQTT): "полегшений" протокол обміну повідомленнями на основі підписки для використання поверх протоколу TCP/IP;

- Extensible Messaging and Presence Protocol (XMPP): протокол обміну повідомленнями, орієнтований на проміжне програмне забезпечення, заснований на XML;

- Advanced Message Queuing Protocol (AMQP): відкритий стандарт для обміну повідомленнями між програмами або організаціями.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ЯКОСТІ ПОВІТРЯ

Для реалізації ефективної IoT-системи екологічного моніторингу необхідно комплексно підійти до вибору апаратних та програмних засобів. У цьому розділі детально розглядаються етапи проєктування, компоненти системи, принципи збору, зберігання та візуалізації даних, що дозволяють забезпечити точний контроль параметрів повітря в реальному часі. Особлива увага приділяється вибору сенсорів, конфігурації мікроконтролерів та створенню веб-інтерфейсу для користувача.

2.1 Апаратне та програмне забезпечення, що використовується в IoT

У цьому розділі розглядаються основні елементи рішення IoT, включаючи протоколи зв'язку, що застосовуються в різних сферах, датчики і приводи, які взаємодіють з фізичним середовищем, а також платформи IoT, що надаються постачальниками для спрощення розробки цих рішень.

2.1.1 Датчики

Датчики є невід'ємними компонентами автоматизації різних додатків, оскільки вони вимірюють і аналізують зібрані дані для виявлення змін у фізичних характеристиках. Щоразу, коли відбувається зміна стану, який датчик призначений для моніторингу, він генерує вимірювану реакцію. Пристрій датчика IoT складається щонайменше з трьох елементів:

- датчики;
- мікроконтролери та пов'язані з ними елементи (пам'ять, периферія тощо);
- шлюз IoT для підключення для відправки відфільтрованих даних в інші системи.

На рисунку 2.1 показані компоненти інтелектуального датчика, позначені синім кольором.

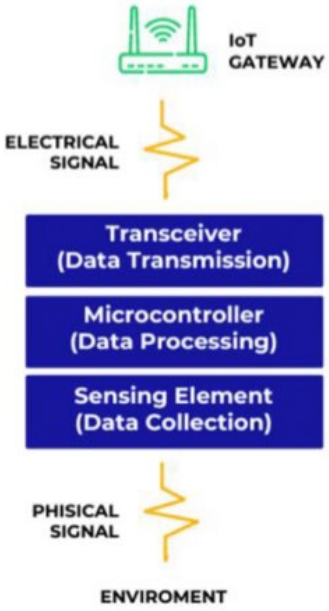


Рисунок 2.1 - Компоненти інтелектуального датчика

Існують різні типи датчиків, які можуть варіюватися від дуже простих до складних. Класифікація датчиків може ґрунтуватися на їхніх технічних характеристиках, методі перетворення, типі використовуваного матеріалу, фізичному явищі, яке вони вимірюють, властивостях того, що вони вимірюють, а також на сфері їхнього застосування.

На рисунку 2.2 зображені різні типи датчиків в IoT, які пояснюються нижче.

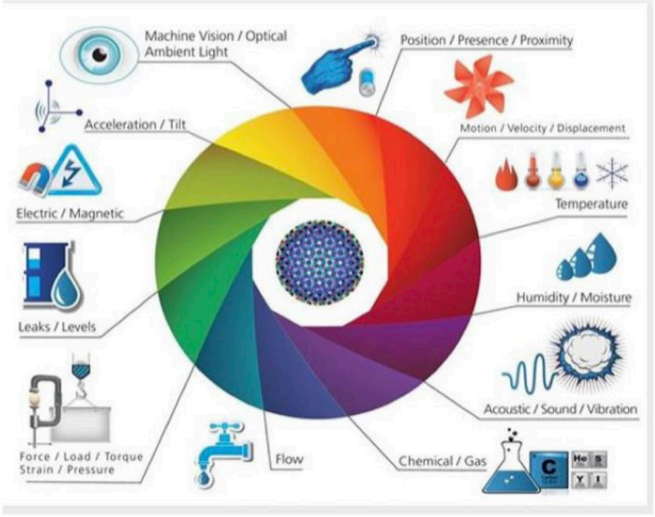


Рисунок 2.2 - Різні типи датчиків

Нижче наведено найпоширеніші технології зондування, класифіковані за типами фізичних величин і використовуваних технологій:

а) датчики температури: Найпоширенішими технологіями, що використовуються для вимірювання температури, є наступні:

1) Термопарні датчики: Складаються з двох різнорідних металів, з'єднаних на одному кінці. На нез'єднаних кінцях термопари генерується напруга, пропорційна температурі в точці з'єднання, і саме там вимірюється температура.

2) Термодатчик опору (RTD): RTD - це пристрої для вимірювання температури, опір яких змінюється зі зміною температури.

3) Термістори: Подібно до RTD, терморезистор є пристроєм для вимірювання температури, опір якого змінюється зі зміною температури. Основна відмінність полягає в технології їх виготовлення (кераміка і полімери), що робить варіацію опору набагато більшою, менш передбачуваною і менш стандартною в порівнянні з RTD. Діапазон температур, який вони охоплюють, явно нижчий, хоча їхня вартість нижча.

4) Напівпровідникові датчики (інтегральні схеми): Вони засновані на збільшенні провідності напівпровідників в залежності від температури.

б) датчики тиску: Це пристрої, призначені для вимірювання тиску рідини (рідини або газу) або атмосферного тиску в певному середовищі. Можна розділити ці датчики відповідно до їхньої топології:

1) Абсолютні: вони вимірюють тиск відносно абсолютного вакууму.

2) Відносні: Вимірюють тиск відносно еталонного тиску, як правило, атмосферного.

3) Диференціальний: Вимірюють різницю тиску між двома точками.

в) датчики наближення: За допомогою датчика наближення можна легко визначити положення будь-якого об'єкта, що знаходиться поблизу, без будь-якого фізичного контакту. Випромінюючи електромагнітне випромінювання, наприклад, інфрачервоне, він виявляє присутність об'єкта, просто шукаючи будь-які зміни у зворотному сигналі. Існують різні типи датчиків наближення: індуктивні, ємнісні, ультразвукові, фотоелектричні, магнітні та інші, орієнтовані на різні сфери

застосування. Цей тип датчиків здебільшого використовується в додатках, що вимагають безпеки та ефективності.

г) датчики швидкості: Це датчик, який обчислює швидкість зміни постійного вимірювання положення і значення положення через відомі проміжки часу. Датчик швидкості може бути лінійним або кутовим.

г) датчики вологості: Датчик вологості вимірює температуру повітря, а також вологість і сигналізує про вологість навколишнього середовища.

д) датчики газу: Їхнє призначення - визначати наявність і часто концентрацію певних газів у навколишньому середовищі. Для виявлення певних газів використовуються спеціальні технології, такі як напівпровідникові, електрохімічні або інфрачервоні датчики. Нижче наведено кілька найпоширеніших газових датчиків:

1) Датчик вуглекислого газу

2) Детектор чадного газу

3) Датчик водню

Це найпоширеніші датчики, які використовуються в проектах такого типу. Однак існує велика різноманітність датчиків, таких як датчики зображення, датчики рівня, датчики води, оптичні датчики та інші.

2.1.2 Інтерфейси між датчиками та мікроконтролерами/пристроями

Зазвичай, для того, щоб один з вищезгаданих датчиків міг інтегруватися в мережу (а згодом і в Інтернет), він повинен бути спочатку підключений до мікроконтролера. Цей мікроконтролер полегшить зв'язок зі шлюзом, локальною мережею або безпосередньо з Інтернетом. Зчитування даних датчика з мікроконтролера зазвичай відбувається з використанням одного з наступних форматів:

- аналогові лінії: Найпростіші датчики забезпечують вимірювання, які зчитуються мікроконтролером як аналогова напруга на вході ADC;

- RS232: Багато датчиків використовують послідовний зв'язок (послідовну лінію) для передачі своїх показань у цифровому вигляді до мікроконтролера;

- SPI (послідовний периферійний інтерфейс): послідовний протокол, який використовує 4 сигнали (MOSI, MISO, CLK, CS) для зв'язку периферійних пристроїв (наприклад, мікросхем датчиків) з мікроконтролерами;

- I2C (Inter IC Communication): для передачі/прийому даних використовується загальна шина, а для зчитування вихідних значень використовується тактовий генератор;

- RFID: RFID може варіюватися від зчитування ідентифікації пристрою (наприклад, мітки на шосе або в торговому центрі) до зчитування значень датчика без батареї або джерела живлення (пасивні бездротові датчики).

2.1.3 Приводи

Привід - це механічний, електричний або гідравлічний пристрій, який перетворює керуючий сигнал у фізичний рух або дію. Приводи є важливими компонентами в автоматизованих системах і відповідають за виконання рішень, прийнятих контролером або системою управління, перетворюючи електричну або іншу інформацію в механічний рух, зміну положення або іншу форму фізичної дії. Основні типи показані нижче:

- електричні приводи: це пристрої, які мають здатність приводити в дію невеликі двигуни, перетворюючи енергію в механічний крутний момент;

- гідравлічні приводи: пристрої, які генерують лінійний або обертальний рух за допомогою нестисливих рідин, як правило, гідравлічного масла. Ці приводи використовують тиск, що створюється гідравлічним насосом, для приведення в дію поршня або циліндра, створюючи таким чином механічний рух;

- пневматичні приводи:використовують стиснене повітря для створення руху. Коли стиснене повітря подається в циліндр, воно спричиняє лінійний або обертальний рух.

2.1.4 Протоколи зв'язку

У цьому підрозділі розглядаються різні способи зв'язку між пристроями loT, які обмінюються інформацією безпосередньо один з одним, без залучення зовнішнього посередника.

Zigbee - це протокол бездротового зв'язку, розроблений для малопотужних пристроїв з низьким енергоспоживанням в персональних мережах. Він широко використовується в додатках для локальних мереж і домашньої автоматизації. Zigbee працює в діапазоні частот 2,4 ГГц і дозволяє формувати mesh-мережі, що полегшує зв'язок між пристроями, навіть якщо вони не мають прямого підключення.

Bluetooth - це стандарт бездротового зв'язку малого радіусу дії, який дозволяє передавати дані між пристроями. Він широко використовується в мобільних пристроях, гарнітурах, колонках та інших периферійних пристроях. Bluetooth використовує радіочастоти малого радіусу дії для встановлення прямого з'єднання між пристроями, що робить його ідеальним для персональних і домашніх застосувань.

Ethernet - це стандарт дротового зв'язку, який використовується для локальних мереж (LAN). Він забезпечує високошвидкісне з'єднання і широко використовується в бізнесі та домашніх умовах для передачі даних. У локальних мережах Ethernet часто використовується для підключення пристроїв, які потребують більш міцного та надійного з'єднання.

Wi-Fi - це стандарт бездротового зв'язку, який забезпечує підключення до локальних мереж та Інтернету. Він широко використовується в домашньому та бізнес-середовищі для підключення таких пристроїв, як смартфони, комп'ютери та пристрої бездротової передачі даних. Wi-Fi забезпечує вищу швидкість передачі даних порівняно з іншими бездротовими протоколами і є важливим для пристроїв, які потребують постійного доступу до мережі.

2.2 Розробка проекту

У цьому підрозділі можна побачити, які існують об'єкти, що перетворюють фізичні величини, пов'язані з властивостями навколишнього середовища, на цифрові змінні.

2.2.1 Збір даних

Збір даних є фундаментальним етапом у побудові системи моніторингу якості повітря на основі IoT. На цьому етапі здійснюється реєстрація фізичних параметрів навколишнього середовища за допомогою відповідних сенсорів, що взаємодіють з мікроконтролерами. Надійність і точність вимірювань безпосередньо залежать від правильної конфігурації апаратного забезпечення та узгодженої роботи елементів системи.

2.2.1.1 Конфігурація та вибір компонентів

Arduino Mega

Arduino Mega 2560 (рисунок 2.3) - це плата мікроконтролера на базі ATmega2560. Вона має 54 цифрових виводи/виводи, 16 аналогових входів, кварцовий генератор на 16 МГц, USB-роз'єм, роз'єм живлення, заголовок ICSP і кнопку скидання. Він пропонує кілька варіантів зв'язку, включаючи UART, SPI та I2C. Це дозволяє підключатись до різноманітних зовнішніх пристроїв та датчиків, що є однією з головних причини її вибору.

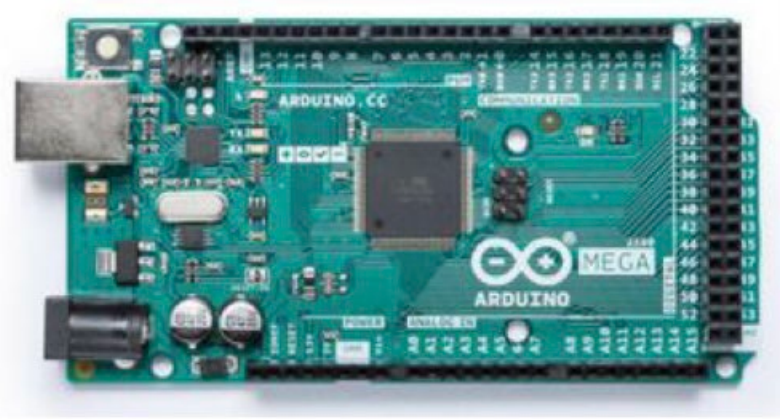


Рисунок 2.3 - Модель Arduino Mega 2560

На рисунку 2.4 показано частину коду Arduino Mega. Код цієї плати базується на отриманні даних з датчиків та їх роздруківці, а потім надсиланні до вузла MCU ESP8266. У додатку 8 цієї роботи представлено повний код.

```
void loop() {
  int s_gasmq135 = analogRead(mq135AnalogPin);
  int mq2AnalogValue = analogRead(mq2AnalogPin);
  int s_sound = analogRead(soundPin);
  float s_temperature = dht.readTemperature();
  float s_humidity = dht.readHumidity();
  float s_gasmq2 = map(mq2AnalogValue, 0, 1023, 0, 100);
  float bmpTemperature = bmp.readTemperature();
  float s_pressure = bmp.readPressure() / 100.0F;

  dataToSend = "temperature=" + String(s_temperature) + "&humidity=" + String(s_humidity) + "&gasmq135=" + String(s_gasmq135) + "&gasmq2=" + String(s_gasmq2) + "&sound="

  Serial.println(dataToSend);
  espSerial.println(dataToSend);

  delay(3000);
}
```

Рисунок 2.4 - Вивід коду датчиків даних

Node MCU ESP8266

NodeMCU Lolin V3 (рисунок 2.5) - це плата для розробки на базі модуля ESP8266 ESP-12F, який є мікроконтролером з інтегрованою функцією Wi-Fi. Це дозволяє пристрою підключатися до мереж Wi-Fi і спілкуватися через Інтернет, що робить його ідеальним для подібних проектів. Ще однією перевагою цього модуля є можливість його програмування за допомогою середовища розробки Arduino.



Рисунок 2.5 - Вузол MCU Lolin V3

Далі розглядається код ESP8266, який відповідає за отримання даних, надрукованих Arduino Mega, для відправки їх до бази даних. На рисунку 2.6 показано бібліотеки, які використовуються для цього коду. Також зберігається в константах назви мережі Wi-Fi, її пароль, хост і порт, що використовуються. Нарешті, для передачі і прийому даних між ESP8266 і Arduino Mega обрані виводи D7 і D8. Щоб побачити точне розташування, можна специфікацію продукту в Додатку.

```
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>

const char* ssid = "-";
const char* password = "-";

const char* host = "-";
const uint16_t port = 80;

const int RX_PIN = D7;
const int TX_PIN = D8;

SoftwareSerial arduinoSerial(RX_PIN, TX_PIN);
String receivedData;
```

Рисунок 2.6 - Бібліотеки та змінні, що використовуються у коді ESP8266

Для цієї частини коду, яка показана на рисунку 2.7, ми встановлюємо для послідовного зв'язку швидкість передачі даних. Це використовується для надсилання даних до та з плати Arduino через USB-порт, до якого вона підключена. У випадку arduinoSerial begin виконується та сама функція, але для пристрою, який підключено через виводи RX/TX плати Arduino. З іншого боку, рядки з 20 по 34 відображають початок Wi-Fi зв'язку ESP8266 з наведеними вище обліковими даними.

Нарешті, ми знаходимо функцію loop, яка, як випливає з її назви, працює в циклі нескінченно. Вона називається readArduino і sendDataToHost, що буде пояснено на наступних рисунках. Якщо коротко, то інформація зчитується з Arduino Mega, підключеного до послідовного порту Arduino, і відправляється на хост через Wi-Fi.

```
16  void setup() {
17    Serial.begin(9600);
18    arduinoSerial.begin(9600);
19
20    Serial.println();
21    Serial.println("Connecting to ");
22    Serial.println(ssid);
23
24    WiFi.mode(WIFI_STA);
25    WiFi.begin(ssid, password);
26
27    while (WiFi.status() != WL_CONNECTED) {
28      delay(1000);
29      Serial.println(".");
30    }
31    Serial.println("");
32    Serial.println("WiFi Connected");
33    Serial.println("IP address: ");
34    Serial.println(WiFi.localIP());
35  }
36
37  void loop() {
38    readFromArduino();
39    if (!receivedData.isEmpty()) {
40      sendDataToHost(receivedData);
41      receivedData = "";
42    }
43    delay(10000);
44  }
```

Рисунок 2.7 - Ініціалізація Wi-Fi та зациклення коду ESP8266

На рисунку 2.8 пояснюється функція readArduino, яка призначена для читання даних з пристрою, підключеного до послідовного порту. Рядок 47 коду перевіряє, чи доступні дані для читання. Потім вона зчитує символи з послідовного порту, доки не зустріне переведення рядка ("\n"). Отриманий рядок зберігається у

змінній receivedData. Нарешті, ці дані виводяться на друк, а буфер послідовного порту очищується, щоб переконатися, що залишкових даних немає.

```
46  void readFromArduino() {
47    if (arduinoSerial.available()) {
48      receivedData = arduinoSerial.readStringUntil('\n');
49      Serial.println("Received data from Arduino Mega: " + receivedData);
50
51      arduinoSerial.flush();
52      delay(10);
53
54    }
55  }
```

Рисунок 2.8 - Функція читання Arduino коду ESP8266

```
57  void sendDataToHost(String data) {
58    Serial.print(">>>>>> connecting to ");
59    Serial.print(host);
60    Serial.print(':');
61    Serial.println(port);
62
63    WiFiClient client;
64    if (!client.connect(host, port)) {
65      Serial.println("connection failed");
66      delay(5000);
67      return;
68    }
69
70    Serial.println("[Sending a request]");
71    client.print(String("GET /paginaweb/save.php?") + data + " HTTP/1.1\r\n" +
72                 "Host: " + host + "\r\n" +
73                 "Connection: close\r\n\r\n");
74
75    unsigned long timeout = millis();
76    while (client.available() == 0) {
77      if (millis() - timeout > 5000) {
78        Serial.println(">>> Client timeout");
79        client.stop();
80        delay(3000);
81        return;
82      }
83    }
84
85    Serial.println("receiving from the remote server");
86    String msg;
87    while (client.available()) {
88      char ch = static_cast<char>(client.read());
89      Serial.print(ch);
90      msg += ch;
91    }
92    Serial.println();
93    if (msg.indexOf("Guardado Correcto") != -1) {
94      Serial.println("DataSaved");
95    } else {
96      Serial.println("Could not save the data");
97    }
98
99    Serial.println();
100   Serial.println(">>>>>>closing connection");
101   Serial.println();
102   client.stop();
103 }
```

Рисунок 2.9 - Функція відправки даних на хост коду ESP8266

На рисунку 2.9 показано код, функція якого полягає у надсиланні даних на віддалений сервер через HTTP. Він з'єднується з віддаленим сервером за допомогою адреси (хост) і порту (порт). Потім він створює і надсилає HTTP GET-запит на віддалений сервер з наданими даними. Чекає до 5 секунд, щоб отримати відповідь від сервера. Нарешті, він зчитує і відображає на послідовному порту відповідь від сервера і закриває з'єднання з віддаленим сервером.

DHT11

DHT11 (рисунок 2.10) - це датчик температури і вологості. Цей датчик використовує терморезистор для вимірювання температури та резистивний датчик вологості для вимірювання вологості повітря. Ці аналогові значення внутрішньо перетворюються на цифрові сигнали, які надсилаються до мікроконтролера.

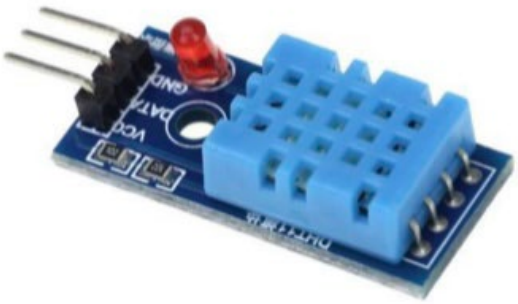


Рисунок 2.10 - Датчик DHT11

MQ-2

MQ-2 (рисунок 2.11) - це датчик газу та диму. Він визначає концентрацію таких газів, як метан, пропан, бутан, чадний газ, алкоголь, дим тощо. Цей датчик містить нагрівальний елемент і датчик газу. Коли газ або дим контактує з датчиком, електричний опір змінюється, що призводить до зміни напруги або струму. Ця зміна використовується для визначення концентрації газу.



Рисунок 2.11 - Датчик MQ-2

MQ-135

MQ-135 (рисунок 2.12) - це датчик якості повітря, який виявляє шкідливі гази, такі як аміак, сірководень, вуглекислий газ і леткі органічні сполуки. Як і MQ-2, MQ-135 має датчик, який виявляє різні гази та сполуки. Опір датчика змінюється залежно від концентрації газів, генеруючи аналоговий сигнал, що вказує на якість повітря.



Рисунок 2.12 - Датчик MQ-135

KY-037

KY-037 (рисунок 2.13) — це модуль виявлення звуку, який визначає інтенсивність звуку в навколишньому середовищі. Цей модуль використовує

мікрофон для виявлення інтенсивності звуку. Зміна інтенсивності звуку впливає на опір мікрофона, генеруючи аналоговий сигнал, пропорційний рівню звуку.

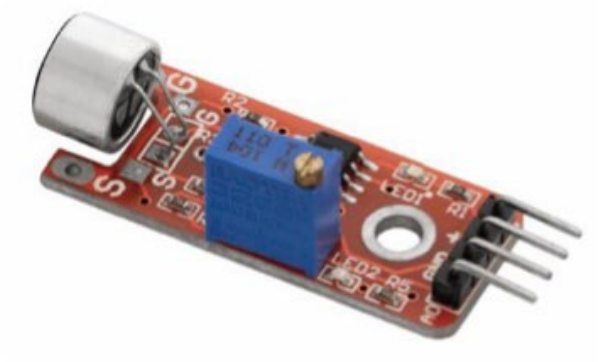


Рисунок 2.13 - Датчик KY-037

BMP180

BMP180 (рисунок 2.14) - датчик тиску і температури. Ця модель вимірює атмосферний тиск за допомогою п'єзоелектричного датчика тиску та температуру за допомогою термістора. Дані перетворюються в цифрові сигнали, які можна зчитувати через з'єднання I2C.

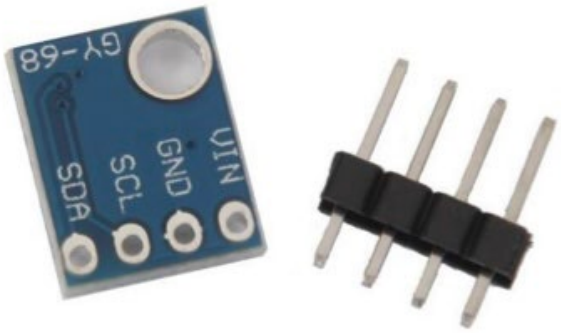


Рисунок 2.14 - Датчик BMP180

Програмування всіх цих датчиків було виконано в одному файлі Arduino, який можна знайти в Додатку 8.

2.2.1.2 Електрична схема

Після представлення вибраних компонентів ми перейдемо до представлення схеми підключення, яка дозволяє отримувати дані з системи. На рисунку 2.15 показана схема, що використовується.

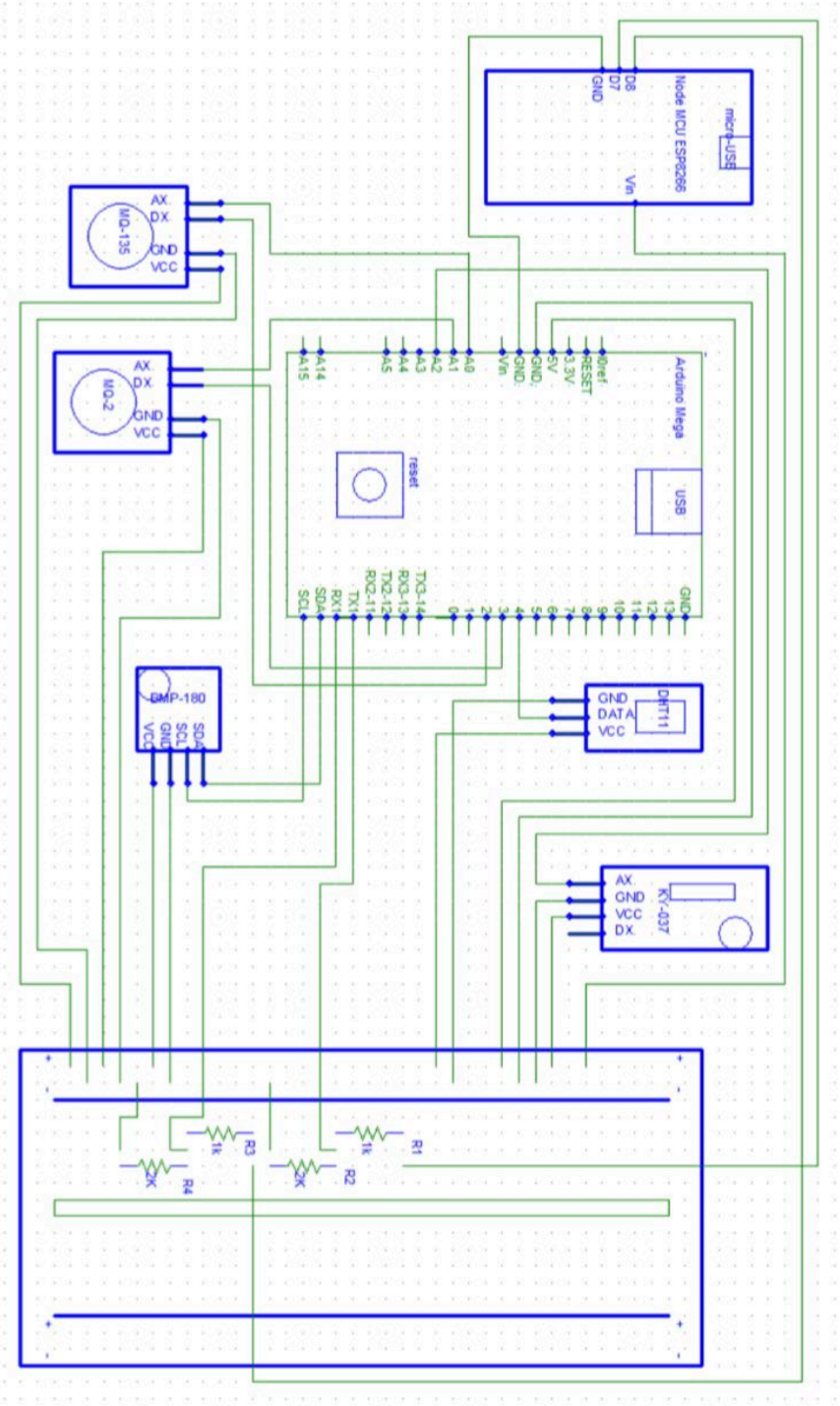


Рисунок 2.15 - Підключення схеми за допомогою PSpice

2.2.2 Зберігання даних

У цьому підрозділі зосереджено увагу на наступному етапі даного проекту: зберіганні даних. Якщо збір інформації - це перший крок, то цей процес просуває наші дані далі, надаючи їм безпечне та доступне місце. Тут пояснюється, як за допомогою XAMPP дані, отримані раніше за допомогою Arduino, будуть зберігатися в базі даних для подальшої обробки.

XAMPP

XAMPP — це безкоштовний пакет програмного забезпечення з відкритим кодом, що полегшує створення локальних веб-серверних середовищ. Його назва є акронімом, що позначає інструменти, які він інтегрує: X (для будь-якої операційної системи), Apache (веб-сервер), MySQL (система управління базами даних), PHP (мова програмування) та Perl (інша мова програмування). Цей набір інструментів забезпечує повне та функціональне середовище для веб-розробки, яке можна легко встановити та запустити на локальних комп'ютерах.

Як зазначалося вище, цей інструмент розробки дозволяє вам тестувати веб-розробку на PHP на власному комп'ютері без необхідності доступу до Інтернету. Ось деякі з включених функцій:

- Apache: це веб-сервер, який обробляє клієнтські запити та обслуговує веб-сторінки. XAMPP використовує Apache як основний сервер у своєму пакеті для управління доставкою веб-контенту;

- MySQL: це система управління базами даних (СУБД), яка організовує та витягує дані. XAMPP інтегрує MySQL для створення та адміністрування локальних баз даних;

- PHP: мова програмування, призначена для веб-розробки. PHP, працюючи разом з Apache, дозволяє створювати динамічні веб-додатки та маніпулювання даними в базі даних;

- Perl: універсальна мова програмування, особливо придатна для обробки тексту. Хоча вона не так широко використовується, як Apache, MySQL і PHP, Perl в XAMPP забезпечує додаткову гнучкість для певних випадків використання.

2.2.2.1 Створення бази даних

У цьому підрозділі розглядаються кроки зі створення бази даних для зберігання отриманих даних. Слід зазначити, що встановлення та налаштування цієї програми не вважається необхідним для цього посібника.

1. Необхідно увійти до PhpMyAdmin: запустити сервер XAMPP і відкріти веб-браузер. Відвідати http://localhost/phpmyadmin/, щоб отримати доступ до інтерфейсу адміністрування MySQL (PhpMyAdmin).

2. Створення бази даних: У розділі "Створити базу даних" ввести ім'я нової бази даних у відповідному полі та натиснути на вкладку "Створити".

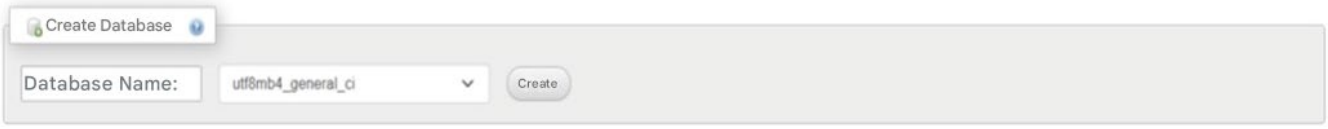


Рисунок 2.16 - Розгортання секції створення бази даних

3. Створення таблиці: У лівій колонці PhpMyAdmin трбеа обрати базу даних, яку було щойно створено. Натиснути кнопку "Створити", щоб почати створювати нову таблицю, і додати потрібну назву.

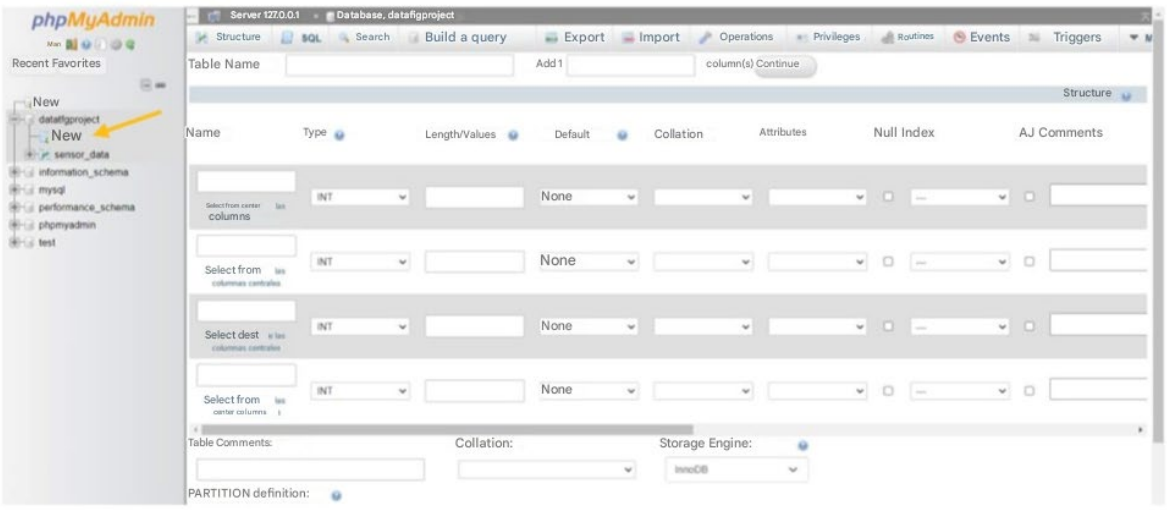
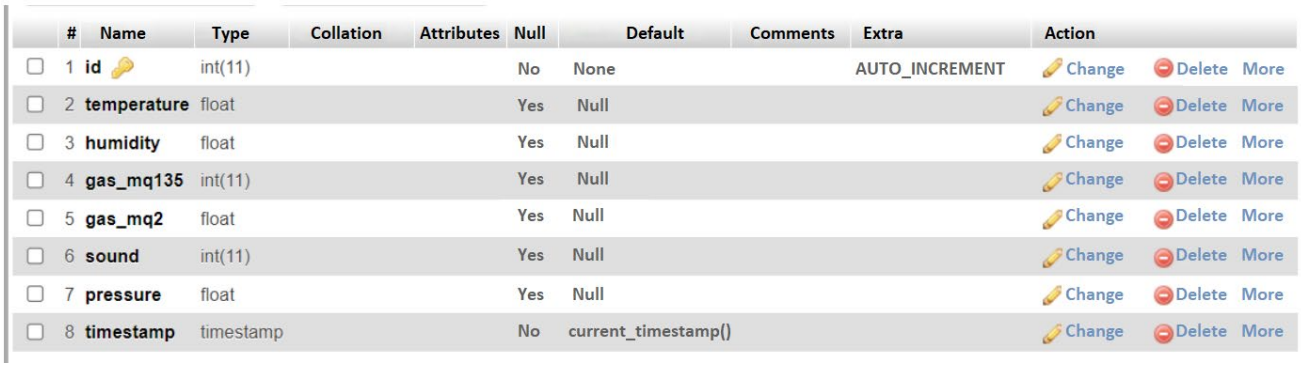


Рисунок 2.17 - Розгортання створення таблиці

4. Призначення змінних: для кожного стовпця таблиці вводимо назву в полі "Name", вибираємо відповідний тип даних (INT, VARCHAR і т.д.) і надаємо будь-які додаткові налаштування (довжина, автоінкремент і т.д.). Після завершення визначення бази даних і таблиці натискаємо кнопку "Зберегти".



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Delete More
2	temperature	float			Yes	Null			Change Delete More
3	humidity	float			Yes	Null			Change Delete More
4	gas_mq135	int(11)			Yes	Null			Change Delete More
5	gas_mq2	float			Yes	Null			Change Delete More
6	sound	int(11)			Yes	Null			Change Delete More
7	pressure	float			Yes	Null			Change Delete More
8	timestamp	timestamp			No	current_timestamp()			Change Delete More

Рисунок 2.18 - Розподіл створених змінних

Для даного проекту було додано змінні temperature, humidity, gasmq2, gasmq135, sound та pressure. Кожній з цих змінних призначено окремий тип змінної. На рисунку 2.18 показано, як кожна з них каталогізована.

2.2.3 Візуалізація даних

У цьому підрозділі розглядається реалізація та візуалізація даних, згенерованих нашою системою LoT. Візуалізація даних необхідна для розуміння та аналізу інформації, зібраної екологічними датчиками, підключеними до нашого середовища. Як невід'ємну частину цього проекту було розроблено локальний веб-сайт, використовуючи такі інструменти, як Visual Studio Code, HTML, CSS та PHP. Цей веб-сайт має подвійну мету: надати ґрунтовну теоретичну базу щодо компонентів реалізованої системи LoT та запропонувати інтерактивний простір для візуалізації даних у реальному часі.

Практичний розділ веб-сайту дозволить користувачам візуалізувати дані про стан навколишнього середовища в режимі реального часу. Було реалізовано інтуїтивно зрозумілий та зручний дизайн, який представляє актуальну інформацію

про температуру, вологість, концентрацію газів та інші параметри, виміряні нашими датчиками. Крім того, було додано можливість вибору дат (фільтрування даних за місяцями року) для вивчення історичних даних, що дає змогу отримати більш повне уявлення про тенденції та зміни в часі.

Вибраний редактор коду - Visual Studio Code. VS Code - це творча платформа, яку можна використовувати для редагування, налагодження та компіляції коду і, нарешті, публікації програми. Вона пропонує вбудовану підтримку HTML, CSS та JavaScript, які є важливими для функціонального та естетичного створення.

Нижче наведено деякі з найважливіших частин коду на розробленій веб-сторінці. На рисунку 2.19 показано код, який відповідає за встановлення з'єднання з базою даних MySQL.

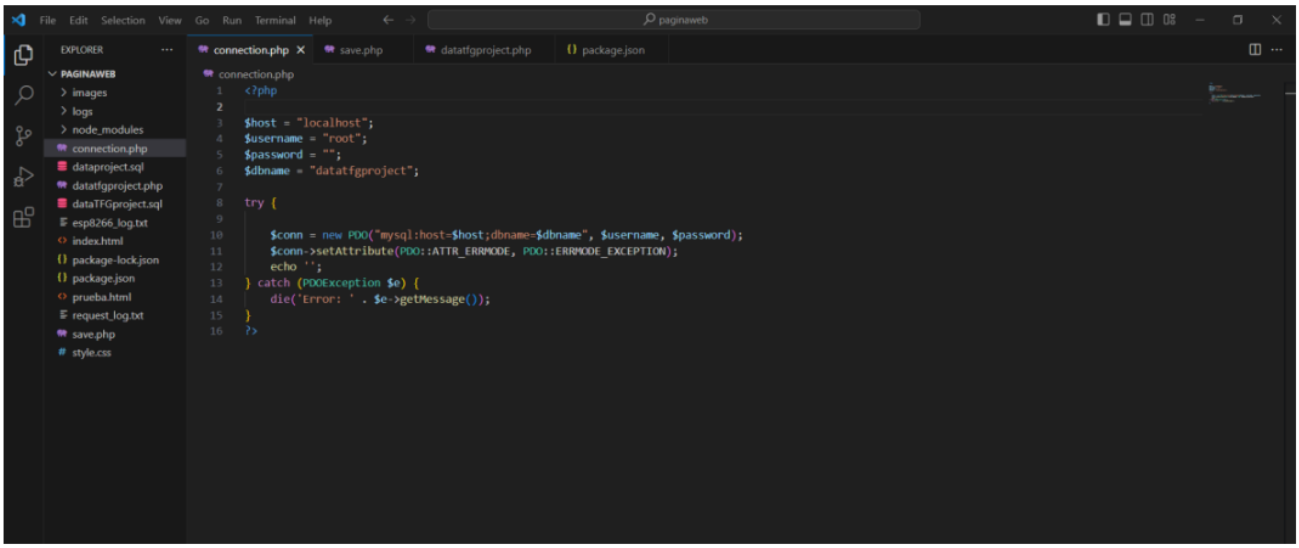


Рисунок 2.19 - Файл підключення до бази даних

Тут визначаються дані, необхідні для підключення до бази даних. Shost вказує місце розташування сервера бази даних, Susername і Spassword - облікові дані користувача, а Sabname - ім'я бази даних, до якої потрібно підключитися. У рядку 10 створюється новий об'єкт PDO, який представляє з'єднання з базою даних MySQL. Йому передаються дані з'єднання (хост, ім'я користувача, пароль) як параметри.

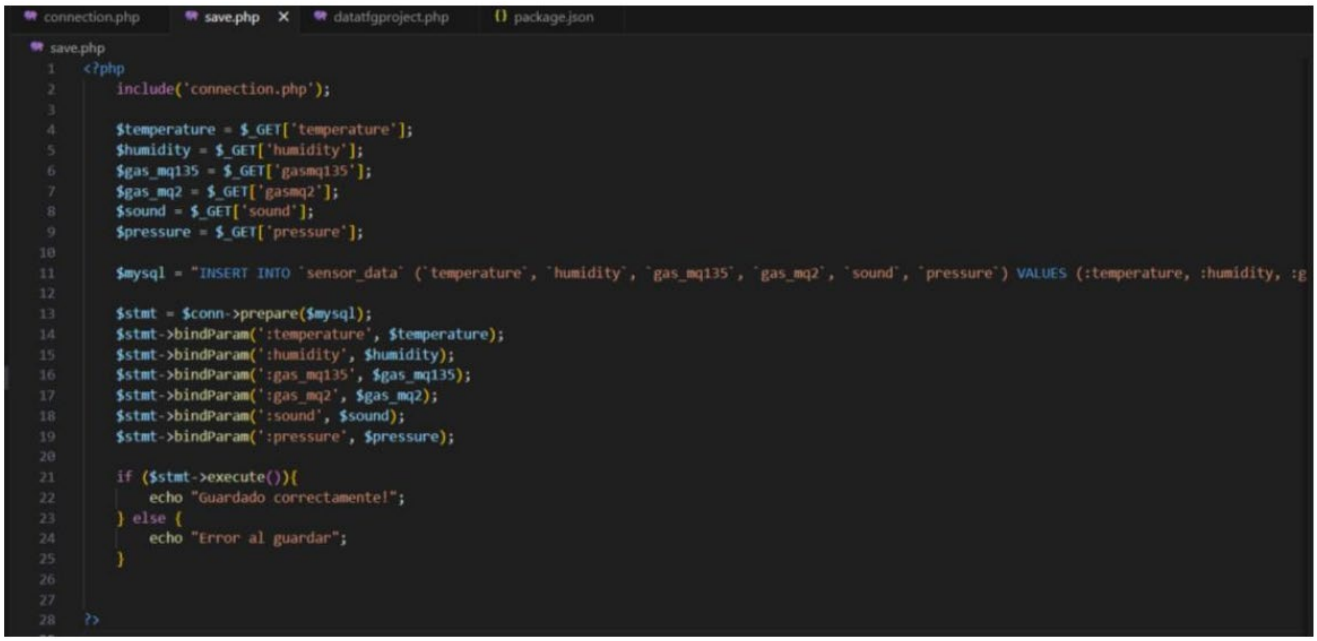


Рисунок 2.20 - Збереження файлу даних

На рисунку 2.20 показано, як дані зберігаються на рівні коду. Рядок 2 викликає наведений вище код для встановлення з'єднання з базою даних. З рядка 4 по рядок 9 дані отримуються з GET-запиту. Імена в квадратних дужках повинні збігатися з іменами, що використовуються для надсилання цих даних з програми. Рядок 11 готує SQL-запит для вставки даних у таблицю 'sensor_data' з відповідними значеннями. Нарешті, PDO використовується для підготовки SQL-запиту. Значення GET-запиту прив'язуються до заповнювачів у SQL-запиті. Таким чином, ми отримуємо дані, надіслані за допомогою GET-запиту, через URL-адресу, яка надсилається з коду ESP8266 Arduino. Потім ці дані додаються до бази даних MySQL.

Перейдемо до останнього файлу нашого сайту. Слід зазначити, що частини html-коду, в яких створюються теоретичні описи, таблиці та інші налаштування, будуть пропущені. Повний код можна знайти в додатку.

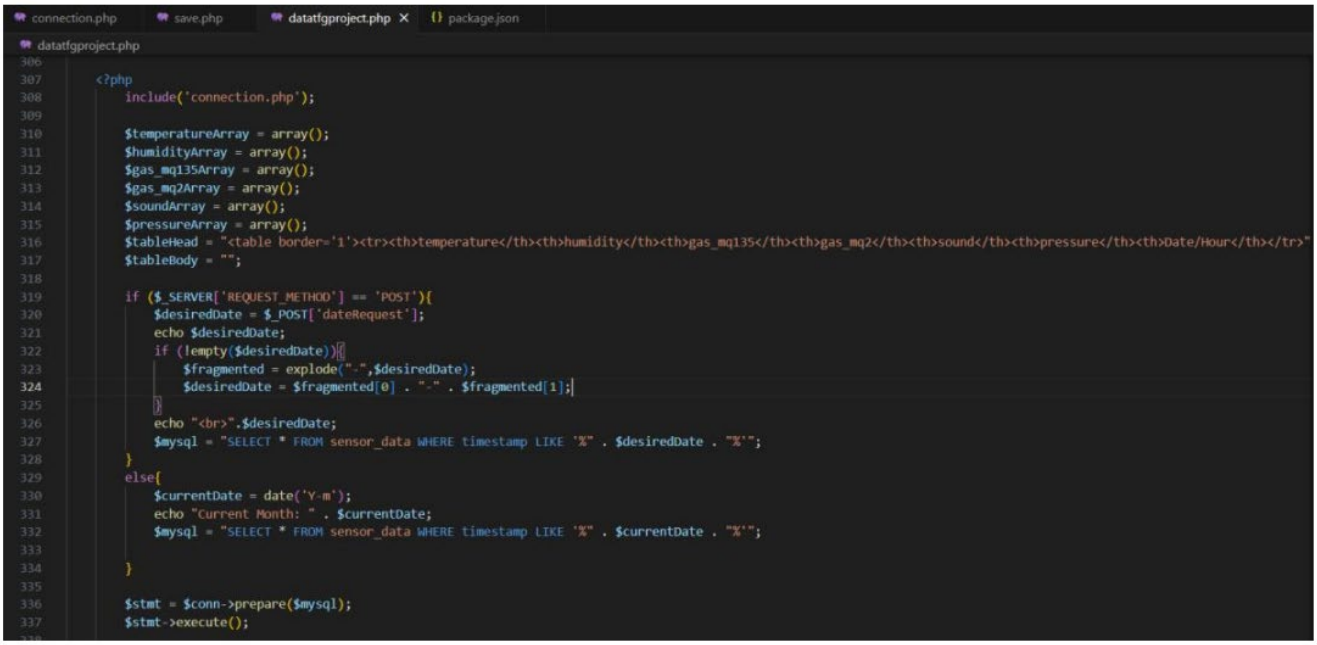


Рисунок 2.21 - Код вибірки дати

Ця перша частина файлу являє собою програмування таблиці, в якій відображаються значення, отримані в залежності від дати вибору. Перші рядки коду присвячені створенню масивів для зберігання змінних даних. Ці масиви будуть використовуватися для зберігання даних, отриманих з бази даних. Потім створюється таблиця, в яку будуть записані всі дані, отримані до цієї дати. Якщо HTTP-запит є POST-запитом, запитувана дата ($desiredDate) буде отримана з форми. Якщо запит не є POST-запитом, поточна дата завантажується з сервера (§currentDate). Нарешті, цикл while використовується для перебору результатів запиту. Кожен набір результатів зберігається у відповідних масивах і будується основна частина таблиці.

У цьому фрагменті рисунка 2.22 отримані дані обробляються і додаються до графіків, які знаходяться в розділі JavaScript в кінці файлу.

```
361         const temperatureData = <?php echo json_encode($temperatureArray); ?>;
362         const humidityData = <?php echo json_encode($humidityArray) ?>;
363         const gas_mq135Data = <?php echo json_encode($gas_mq135Array) ?>;
364         const gas_mq2Data = <?php echo json_encode($gas_mq2Array) ?>;
365         const soundData = <?php echo json_encode($soundArray) ?>;
366         const pressureData = <?php echo json_encode($pressureArray) ?>;
367         const table = "<?php echo $tableHead.$tableBody ?>";
368
369         let divTable = document.createElement("div");
370         divTable.innerHTML = table;
371         divTable.className = "text-center";
372         document.querySelector("#modalBody").appendChild(divTable);
373
374         let samples = [];
375         let dataLength = Object.keys(temperatureData).length;
376         for (let i = 0; i < dataLength; i++) {
377             samples.push(i+1);
```

Рисунок 2.22 - Вставка даних у код графіків

З одного боку, дані PHP (температура, вологість тощо) зчитуються і зберігаються у змінних JavaScript, таких як temperatureData, humidityData тощо. Потім ці дані використовуються для створення графіків за допомогою бібліотеки Chart.js. Код створення діаграми можна знайти нижче, де змінні "temperatureData", "humidityData" і т.д. використовуються для налаштування відповідних наборів даних для діаграми.

3 ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ ТА ПРАКТИЧНЕ ЗАСТОСУВАННЯ СИСТЕМИ

3.1 Результати інтерфейсу веб-сайту

Наступні рисунки показують кінцевий результат проекту. На рисунках 3.1-3.10 можна побачити остаточний дизайн веб-сайту. Тут ми бачимо теоретичну частину, в якій представлені використовувані датчики та їх функціональність. Нарешті, в останній частині графічно відображаються графіки, отримані за допомогою даних з датчиків.

На рисунку 3.1 показано зразок веб-сайту з описом мікроконтролера.



Рисунок 3.1 - Зразок веб-сайту

На рисунку 3.2 показано зразок веб-сайту з описом модуля ESP8266.



Рисунок 3.2 - Зразок веб-сайту

На рисунку 3.3 показано зразок веб-сайту з описом DHT11.



Рисунок 3.3 - Зразок веб-сайту

На рисунку 3.4 показано зразок веб-сайту з описом MQ-135.



Рисунок 3.4 - Зразок веб-сайту

На рисунку 3.5 показано зразок веб-сайту з описом MQ-2.



Рисунок 3.5 - Зразок веб-сайту

На рисунку 3.6 показано зразок веб-сайту з описом модуля KY-037.



Рисунок 3.6 - Зразок веб-сайту

На рисунку 3.7 показано зразок веб-сайту з описом BMP-180.



Рисунок 3.7 - Зразок веб-сайту

На рисунку 3.8 показана сторінка веб-сайту з інформацією про проект та екологічні сенсори.

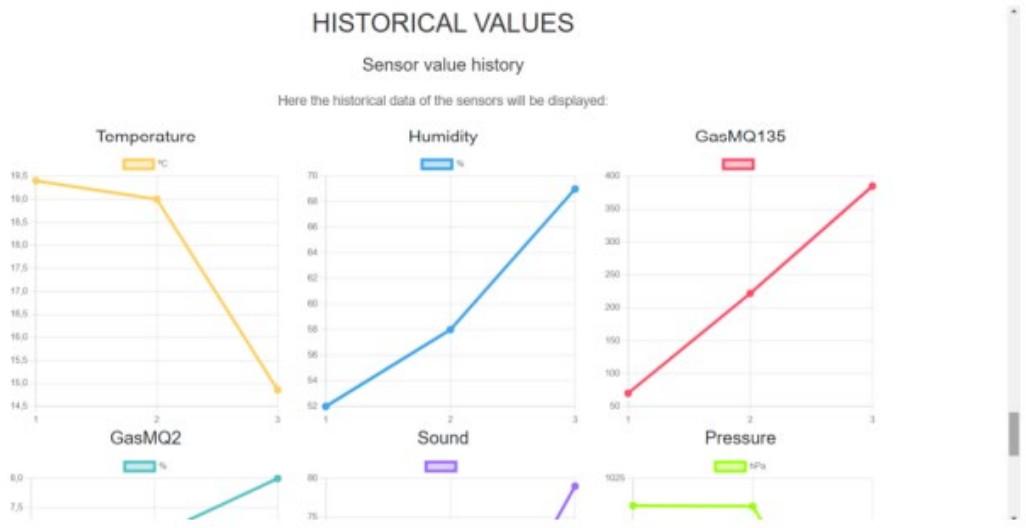


Рисунок 3.8 - Зразок веб-сайту

На рисунку 3.9 показана сторінка виводу результатів вимірювань у табличному форматі з фільтрацією за датою.

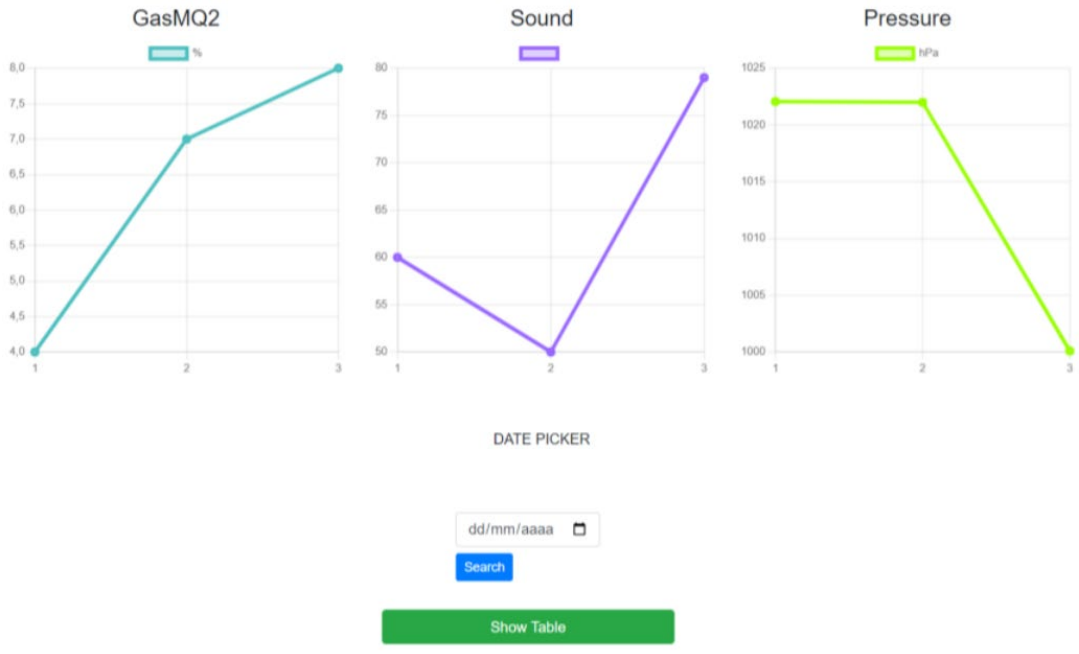


Рисунок 3.9 - Зразок веб-сайту

На рисунку 3.10 показаний інтерфейс графічної візуалізації даних про якість повітря за обраний період.

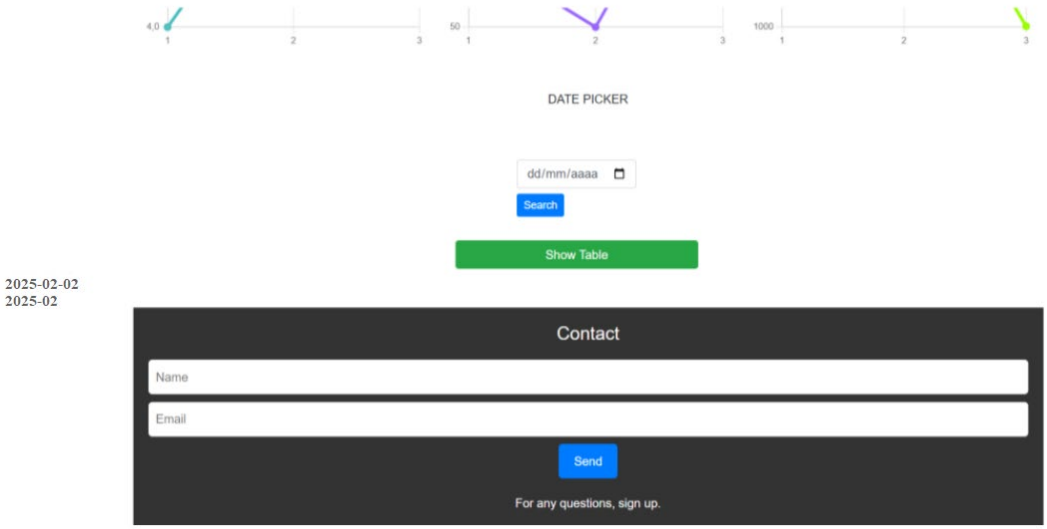


Рисунок 3.10 - Зразок веб-сайту

На рисунку 3.11 показано селектор дати, за допомогою якого можна вибрати місяць для вибірки даних. У свою чергу, опція "Показати таблицю" показує всі значення, зібрані в цьому місяці. Це можна побачити на рисунку 3.12.

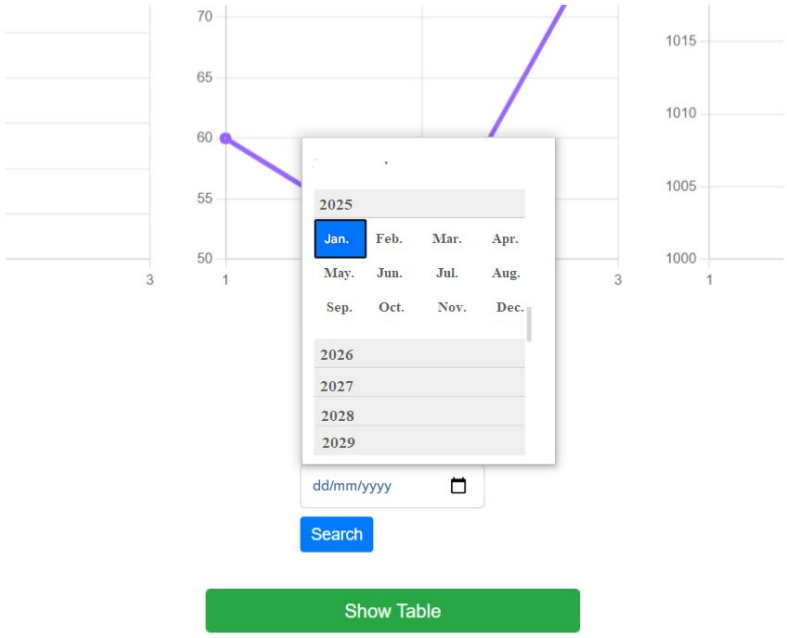


Рисунок 3.11 - Вибір дати

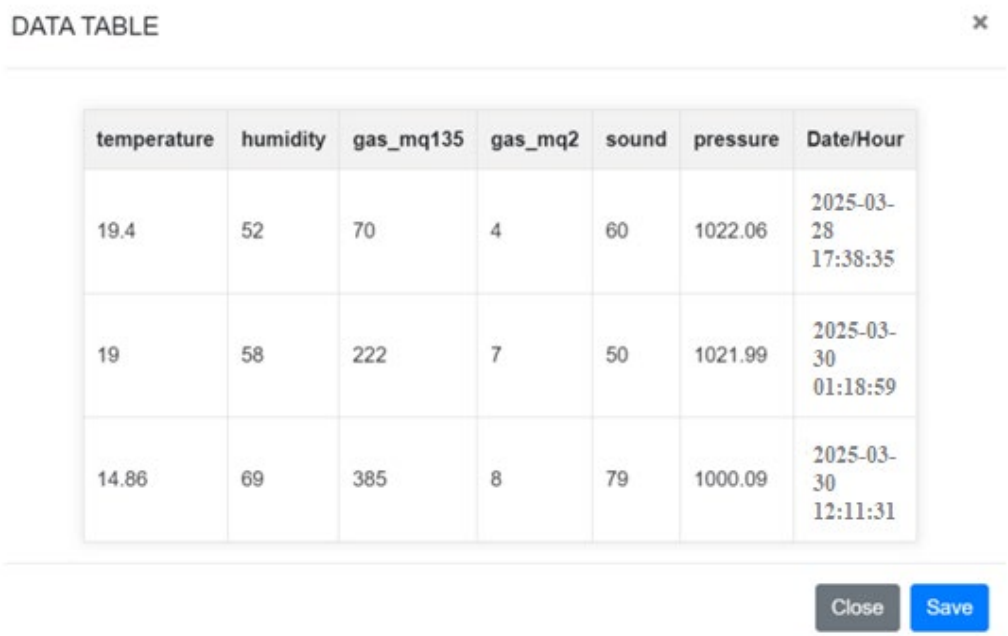


Рисунок 3.12 - Відображення даних у таблиці

3.2 Збір, зберігання та візуалізація результатів

Тепер ми подивимося, як відбувається вся процедура отримання, зберігання та візуалізації даних на нашій веб-сторінці. На рисунку 3.13 показано перший крок - друк даних, отриманих датчиками, на послідовний порт Arduino Mega. Потім ці рядки даних надсилаються до вузла ESP8266 MCU.



Рисунок 3.13 - Послідовний порт Arduino Mega

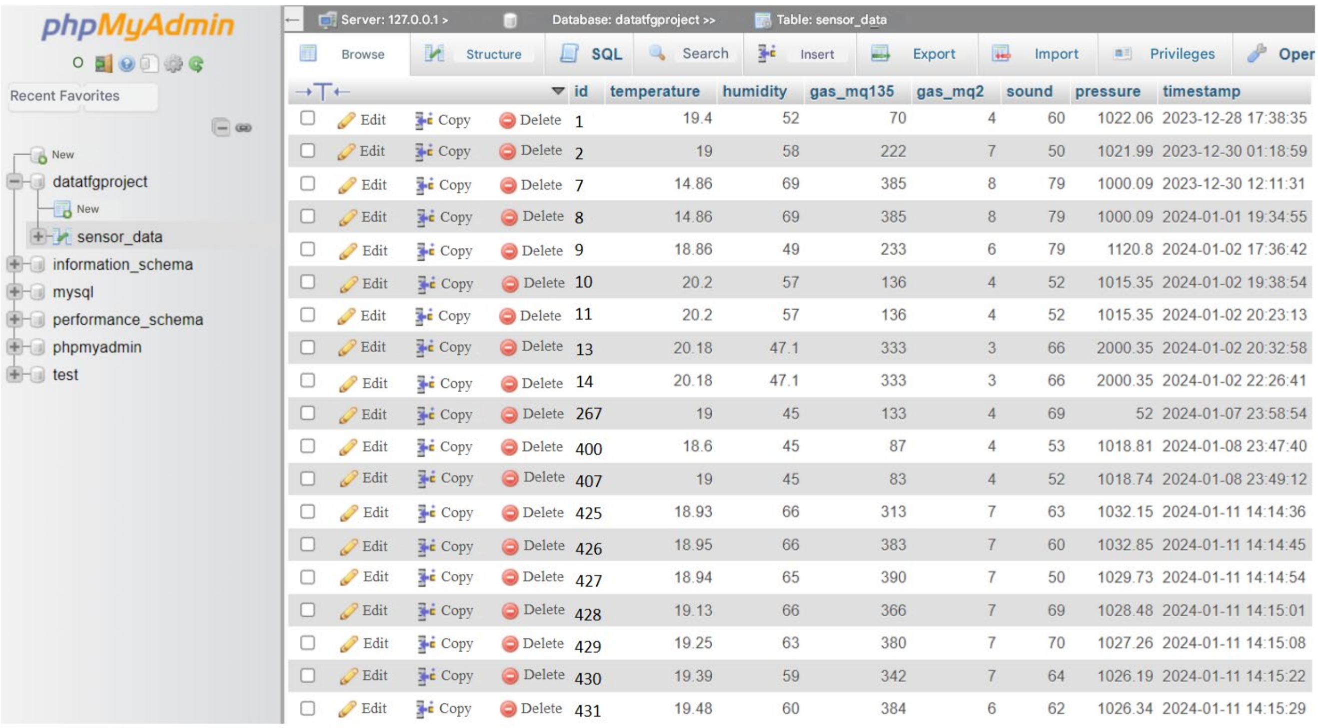
На рисунку 3.14 показано, як MCU вузла ESP8266 отримує рядок даних через послідовний зв'язок з Arduino Mega. Отримавши дані, ми розділяємо кожне

значення разом з даними, а потім додаємо їх до URL-адреси нашої бази даних, щоб зберігати та візуалізувати їх.



Рисунок 3.14 - Послідовний порт вузла MCU ESP8266

Нарешті, на цьому рисунку 3.15 можна побачити, як зберігаються отримані дані. Тут показано час отримання та його вартість.



id	temperature	humidity	gas_mq135	gas_mq2	sound	pressure	timestamp
1	19.4	52	70	4	60	1022.06	2023-12-28 17:38:35
2	19	58	222	7	50	1021.99	2023-12-30 01:18:59
7	14.86	69	385	8	79	1000.09	2023-12-30 12:11:31
8	14.86	69	385	8	79	1000.09	2024-01-01 19:34:55
9	18.86	49	233	6	79	1120.8	2024-01-02 17:36:42
10	20.2	57	136	4	52	1015.35	2024-01-02 19:38:54
11	20.2	57	136	4	52	1015.35	2024-01-02 20:23:13
13	20.18	47.1	333	3	66	2000.35	2024-01-02 20:32:58
14	20.18	47.1	333	3	66	2000.35	2024-01-02 22:26:41
267	19	45	133	4	69	52	2024-01-07 23:58:54
400	18.6	45	87	4	53	1018.81	2024-01-08 23:47:40
407	19	45	83	4	52	1018.74	2024-01-08 23:49:12
425	18.93	66	313	7	63	1032.15	2024-01-11 14:14:36
426	18.95	66	383	7	60	1032.85	2024-01-11 14:14:45
427	18.94	65	390	7	50	1029.73	2024-01-11 14:14:54
428	19.13	66	366	7	69	1028.48	2024-01-11 14:15:01
429	19.25	63	380	7	70	1027.26	2024-01-11 14:15:08
430	19.39	59	342	7	64	1026.19	2024-01-11 14:15:22
431	19.48	60	384	6	62	1026.34	2024-01-11 14:15:29

Рисунок 3.15 - База даних з отриманими значеннями

На рисунках вище вже показані результати графіків. Користувачеві потрібно вибрати потрібний місяць, щоб побачити результати.

3.3 Практичні застосування

Впровадження системи моніторингу навколишнього середовища IoT представляє собою універсальну платформу з численними практичними застосуваннями. Зібрані дані можуть бути застосовані в різних реальних сценаріях, надаючи цінну інформацію про стан навколишнього середовища.

3.3.1 Практичні застосування в реальному світі

Нижче наведені приклади практичного використання і техніко-економічне обґрунтування в реальних ситуаціях:

Сільськогосподарський моніторинг:

- застосування: Система може бути розгорнута в сільському господарстві для моніторингу температури, вологості та вологості ґрунту. Фермери можуть отримувати дані в режимі реального часу, що дозволяє їм приймати обґрунтовані рішення щодо зрошення, посадки та збору врожаю;

- економічна доцільність: Покращення врожайності та використання ресурсів, що веде до сталого ведення сільського господарства.

Управління якістю повітря в містах:

- застосування: Датчики Інтернету речей можуть вимірювати параметри якості повітря, такі як рівень загазованості та вміст твердих частинок у міських районах. Місцева влада може використовувати ці дані для своєчасного втручання для контролю забруднення.

- доцільність: Зменшення забруднення повітря, сприяння охороні здоров'я населення та сталому розвитку міст.

Розумні будинки та енергоефективність:

- застосування: Інтеграція екологічних датчиків у розумні будинки дозволяє в режимі реального часу контролювати якість повітря в приміщеннях. Автоматизовані системи можуть регулювати вентиляцію та опалення на основі зібраних даних.

- здійсненність: Підвищення енергоефективності та покращення комфорту в приміщенні для мешканців.

Збереження дикої природи:

- застосування: Розміщення датчиків у природних середовищах існування допомагає відстежувати параметри навколишнього середовища, важливі для збереження дикої природи, такі як температура і вологість. Ці дані допомагають дослідникам зрозуміти екосистеми та захистити види, що перебувають під загрозою зникнення.

- здійсненність: Сприяння науковим дослідженням та природоохоронним зусиллям для збереження біорізноманіття.

3.3.2 Техніко-економічне обґрунтування

Доцільність впровадження цієї системи моніторингу довкілля за допомогою Інтернету речей в реальних умовах очевидна з успіху подібних систем у різних сферах. Ключові фактори, що впливають на доцільність, включають:

- економічна ефективність: використання таких компонентів, як Arduino, забезпечує економічно ефективне рішення в порівнянні з традиційними системами моніторингу;

- масштабованість: система може бути легко масштабована для пристосування до широкого спектру середовищ і вимог до моніторингу.

- точність даних: калібрування і точна сенсорна технологія забезпечують точність даних, що робить систему надійною для критично важливих застосувань.

- можливість інтеграції:сумісність системи з існуючими технологіями сприяє безперешкодній інтеграції в різні галузі та сфери застосування.

3.4 Безпека та конфіденційність IoT

Безпека рішень IoT є ключовим питанням як для розробників, так і для користувачів. Додатки Інтернету речей повинні не тільки протистояти відомим атакам, але й забезпечувати безперервну безпеку передачі даних між пристроями. Відсутність безпеки збільшує ризик витоку вашої особистої інформації під час збору та передачі даних на IoT-пристрій. Пристрої Інтернету речей підключені до споживчої мережі. Ця мережа також пов'язана з іншими системами. Отже, якщо пристрій Інтернету речей містить вразливості в системі безпеки, він може завдати шкоди мережі споживача.

Інструменти пошуку та безпеки розробляються для вирішення цих проблем. Залежно від підходу до безпеки, який ми хочемо застосувати до нашого проекту, ми можемо додати різні рішення або запобіжні заходи:

- системи брандмауерів: брандмауер - це система захисту, яка безперервно контролює мережеву активність за допомогою набору заздалегідь визначених правил;

- системи виявлення та запобігання вторгнень: системи виявлення та запобігання вторгненням IDPS складаються з набору механізмів захисту, спрямованих на виявлення, реєстрацію та запобігання потенційним загрозам у режимі реального часу;

- шифрування даних: впровадження надійних алгоритмів шифрування, щоб забезпечити конфіденційність переданих і збережених даних;

- надійна автентифікація: використання надійних методів автентифікації, такі як надійні паролі або навіть біометричні механізми, щоб гарантувати, що тільки авторизовані користувачі мають доступ до пристроїв і даних;

- оновлення прошивки: створення безпечного та ефективного процесу віддаленого оновлення прошивки пристрою, виправлення потенційних вразливостей безпеки;

- брандмауери та фільтри трафіку: налаштування брандмауерів і фільтрів трафіку для контролю потоку даних між пристроями Інтернету речей та іншими системами, запобігаючи несанкціонованим вторгненням;

- безперервний моніторинг: впровадження систем безперервного моніторингу для виявлення та швидкого реагування на незвичну активність або спроби несанкціонованого доступу;

- захист від атак типу «відмова в обслуговуванні» (DoS): впровадження заходів для запобігання або пом'якшення наслідків DoS-атак, які можуть вплинути на доступність пристроїв і сервісів IoT;

- управління ідентифікацією та доступом (IAM): створення надійних політик IAM, щоб контролювати, хто має доступ до яких ресурсів і даних, тим самим зменшуючи поверхню атаки;

- фізична безпека: врахування фізичної безпеки пристроїв, наприклад, обмежуючи фізичний доступ до пристроїв або розміщуючи їх у безпечних місцях;

- освіта та обізнаність: навчання кінцевих користувачів і розробників найкращим практикам безпеки та підвищення обізнаність про потенційні загрози і способи їх зменшення.

ВИСНОВКИ

У процесі реалізації дипломного проєкту було досягнуто основної мети – створення дієздатної IoT-системи для моніторингу якості повітря, яка поєднує простоту апаратної частини, доступність для кінцевого користувача та ефективність передачі та візуалізації екологічних даних.

Було обґрунтовано вибір платформи Arduino Mega, яка забезпечила надійне зчитування інформації з кількох датчиків, та модуля NodeMCU ESP8266, що відповідає за передачу даних до серверу через Wi-Fi. Впроваджена система здатна вимірювати температуру, вологість, атмосферний тиск, рівень газів і шуму в режимі реального часу, зберігаючи дані для подальшого аналізу. Платформа XAMPP разом із PHP та MySQL дала змогу ефективно реалізувати серверну частину проєкту.

Розроблений веб-інтерфейс дозволяє не лише переглядати поточні показники, але й фільтрувати інформацію за історичними параметрами, що відкриває нові можливості для аналізу змін екологічної ситуації. Це важливо як для пересічного користувача, так і для фахівців, які займаються питаннями екології, міського планування, сільського господарства чи промисловості.

Проєкт також виявив низку обмежень, серед яких — обмежена точність недорогих датчиків, нестабільність передачі даних у разі відсутності інтернету та необхідність подальшого вдосконалення безпеки даних. Проте на фоні доступності компонентів, простоти масштабування та легкості адаптації до різних потреб, ці недоліки є мінімальними.

З точки зору практичного застосування, ця система демонструє свою корисність у різноманітних реальних ситуаціях. Від екологічного моніторингу в промисловому середовищі до оцінки якості повітря в міських районах - сфера застосування дуже широка. Доцільність системи була перевірена в різних реальних сценаріях, що підкреслило її надійність і адаптивність.

На закінчення, цей проект являє собою не тільки технологічний прорив у сфері моніторингу навколишнього середовища, а й свідчення здатності технології Інтернету речей вирішувати значні проблеми. Надаючи дані про стан довкілля в режимі реального часу, ця система екологічного моніторингу позиціонується як цінний інструмент для прийняття обґрунтованих рішень та збереження довкілля.

Таким чином, розроблена система демонструє перспективність використання IoT у сфері моніторингу довкілля. Подальші дослідження можуть бути спрямовані на інтеграцію більш точних сенсорів, автоматичну систему оповіщення про перевищення порогів шкідливих речовин та використання мобільних застосунків для доступу до даних. Такий підхід сприятиме формуванню свідомої екологічної культури та оперативному реагуванню на загрози для здоров’я і довкілля.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, 17(4), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095

2. Bandyopadhyay, D., & Sen, J. (2011). Internet of Things: Applications and challenges in technology and standardization. Wireless Personal Communications, 58, 49–69.

3. Buyya, R., Vecchiola, C., & Selvi, S. T. (2013). Mastering Cloud Computing: Foundations and Applications Programming. Morgan Kaufmann.

4. Cuno, M. (2018). Programming the Internet of Things: An Introduction to Building Integrated, Device-to-Cloud IoT Solutions. O'Reilly Media.

5. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645–1660.

6. Gupta, P., Agrawal, D., & Yamaguchi, S. (2016). Handbook of Research on Cloud Infrastructures for Big Data Analytics. IGI Global.

7. Hassan, Q. F. (2018). Internet of Things A to Z: Technologies and Applications. Wiley-IEEE Press.

8. Li, S., Xu, L. D., & Zhao, S. (2015). The internet of things: A survey. Information Systems Frontiers, 17(2), 243–259.

9. Mahmoud, R., Yousuf, T., Aloul, F., & Zualkernan, I. (2015). Internet of Things (IoT) security: Current status, challenges and prospective measures. International Conference on Internet Technology and Secured Transactions (ICITST), 336–341.

10. Minerva, R., Biru, A., & Rotondi, D. (2015). Towards a definition of the Internet of Things (IoT). IEEE Internet Initiative.

11. Mohammadi, M., Al-Fuqaha, A., Sorour, S., & Guizani, M. (2018). Deep learning for IoT big data and streaming analytics: A survey. IEEE Communications Surveys & Tutorials, 20(4), 2923–2960.

12. Ojha, T., Misra, S., & Raghuwanshi, N. S. (2015). Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges. Computers and Electronics in Agriculture, 118, 66–84.

13. Pahlavan, K., & Levesque, A. H. (2005). Wireless Information Networks (2nd ed.). Wiley.

14. Pottie, G. J., & Kaiser, W. J. (2000). Wireless integrated network sensors. Communications of the ACM, 43(5), 51–58.

15. Ray, P. P. (2018). A survey on Internet of Things architectures. Journal of King Saud University – Computer and Information Sciences, 30(3), 291–319.

16. Raza, S., Wallgren, L., & Voigt, T. (2013). Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues. IEEE Communications Surveys & Tutorials, 16(3), 1291–1307.

17. Shelby, Z., & Bormann, C. (2016). 6LoWPAN: The Wireless Embedded Internet. Wiley.

18. Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5), 637–646.

19. Singh, J., Pasquier, T., Bacon, J., Ko, H., & Eyers, D. (2015). Twenty security considerations for cloud-supported Internet of Things. IEEE Internet of Things Journal, 3(3), 269–284.

20. Smail, O., & Abid, M. (2017). Smart home system design based on internet of things. International Journal of Applied Engineering Research, 12(24), 14238–14242.

21. Stojkoska, B. L. R., & Trivodaliev, K. V. (2017). A review of Internet of Things for smart home: Challenges and solutions. Journal of Cleaner Production, 140, 1454–1464.

22. Suryadevara, N. K., & Mukhopadhyay, S. C. (2015). Smart Homes. Springer.

23. Tawalbeh, L. A., Muheidat, F., Tawalbeh, M., & Quwaider, M. (2020). IoT privacy and security: Challenges and solutions. Applied Sciences, 10(12), 4102.

24. Zhang, Y., Wang, L., & Sun, G. (2019). Distributed intrusion detection system in a multi-layer network architecture of smart homes. Future Generation Computer Systems, 92, 799–807.

25. ZigBee Alliance. (2017). ZigBee Specification: ZigBee Document 053474r20. Retrieved from https://zigbee.org/

Додаток А

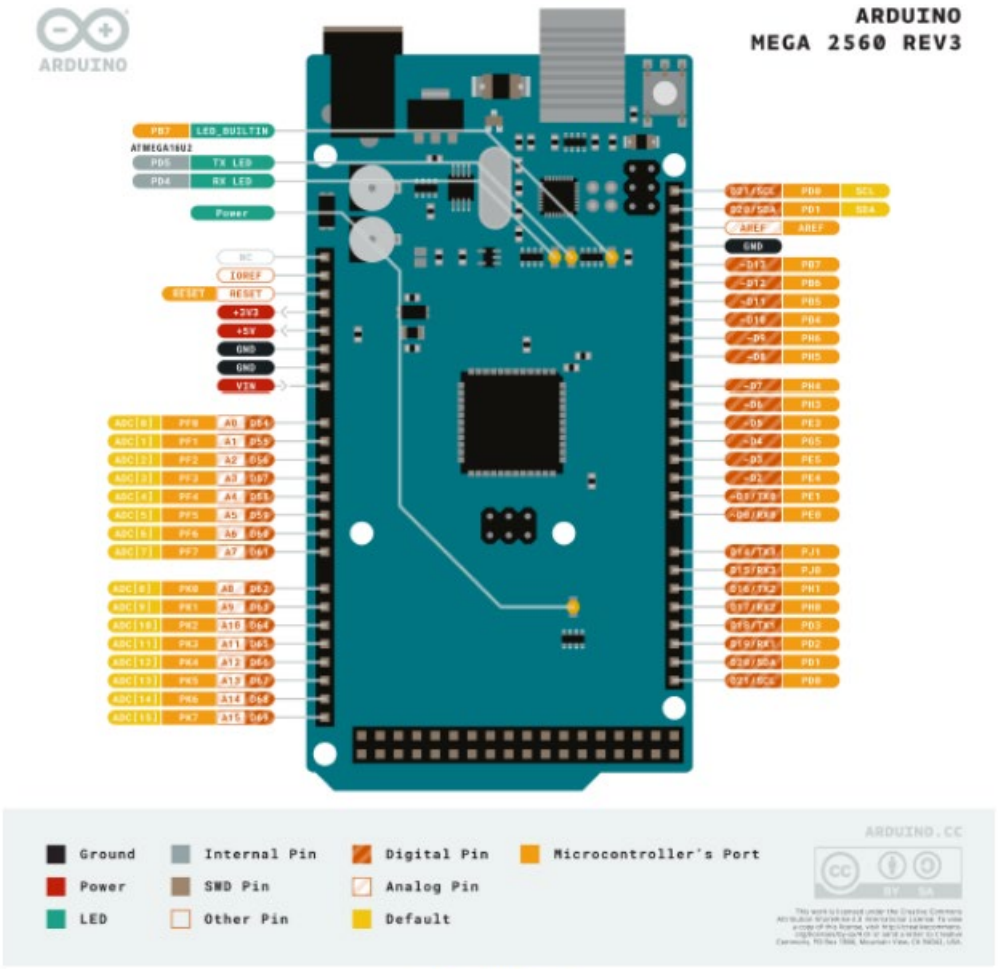
Специфікації Arduino Mega

Arduino® MEGA 2560 Rev3

Features

- **ATmega2560 Processor**
  - Up to 16 MIPS Throughput at 16MHz
  - 256k bytes (of which 8k is used for the bootloader)
  - 4k bytes EEPROM
  - 8k bytes Internal SRAM
  - 32 × 8 General Purpose Working Registers
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Four Programmable Serial USART
  - Controller/Peripheral SPI Serial Interface
- **ATmega16U2**
  - Up to 16 MIPS Throughput at 16 MHz
  - 16k bytes ISP Flash Memory
  - 512 bytes EEPROM
  - 512 bytes SRAM
  - USART with SPI master only mode and hardware flow control (RTS/CTS)
  - Master/Slave SPI Serial Interface
- **Sleep Modes**
  - Idle
  - ADC Noise Reduction
  - Power-save
  - Power-down
  - Standby
  - Extended Standby
- **Power**
  - USB Connection
  - External AC/DC Adapter
- **I/O**
  - 54 Digital
  - 16 Analog
  - 15 PWM Output

5 Connector Pinouts



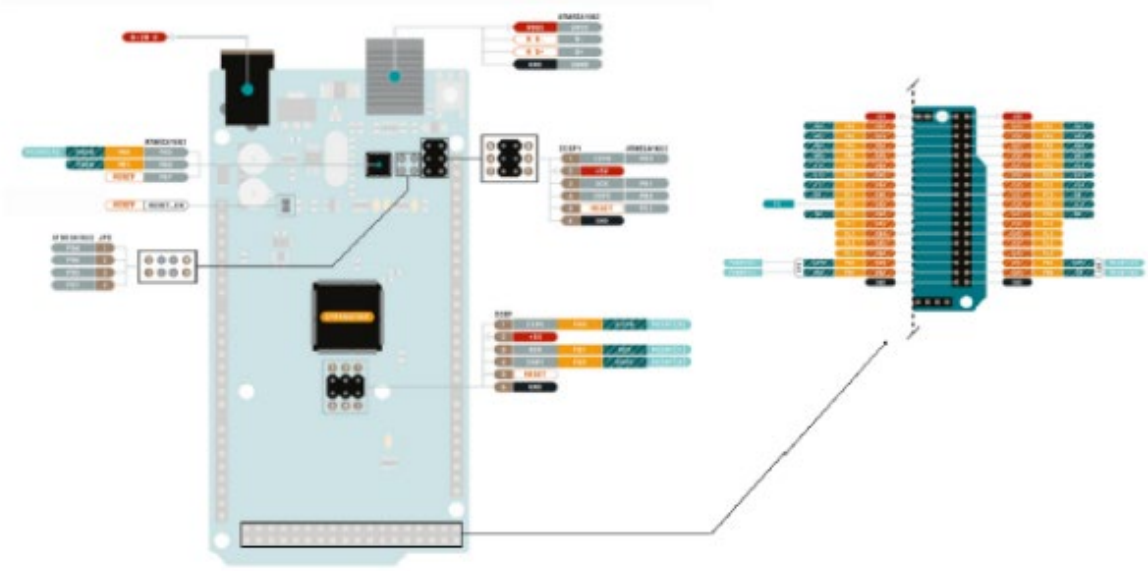
Arduino Mega Pinout

Pin	Function	Type	Description
1	NC	NC	Not Connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog	Analog input 0 /GPIO
10	A1	Analog	Analog input 1 /GPIO
11	A2	Analog	Analog input 2 /GPIO
12	A3	Analog	Analog input 3 /GPIO
13	A4	Analog	Analog input 4 /GPIO
14	A5	Analog	Analog input 5 /GPIO
15	A6	Analog	Analog input 6 /GPIO
16	A7	Analog	Analog input 7 /GPIO
17	A8	Analog	Analog input 8 /GPIO
18	A9	Analog	Analog input 9 /GPIO
19	A10	Analog	Analog input 10 /GPIO
20	A11	Analog	Analog input 11 /GPIO
21	A12	Analog	Analog input 12 /GPIO
22	A13	Analog	Analog input 13 /GPIO
23	A14	Analog	Analog input 14 /GPIO
24	A15	Analog	Analog input 15 /GPIO

5.2 Digital

Pin	Function	Type	Description
1	D21/SCL	Digital Input/I2C	Digital input 21/I2C Dataline
2	D20/SDA	Digital Input/I2C	Digital input 20/I2C Dataline
3	AREF	Digital	Analog Reference Voltage
4	GND	Power	Ground
5	D13	Digital/GPIO	Digital input 13/GPIO
6	D12	Digital/GPIO	Digital input 12/GPIO
7	D11	Digital/GPIO	Digital input 11/GPIO
8	D10	Digital/GPIO	Digital input 10/GPIO
9	D9	Digital/GPIO	Digital input 9/GPIO
10	D8	Digital/GPIO	Digital input 8/GPIO
11	D7	Digital/GPIO	Digital input 7/GPIO
12	D6	Digital/GPIO	Digital input 6/GPIO
13	D5	Digital/GPIO	Digital input 5/GPIO
14	D4	Digital/GPIO	Digital input 4/GPIO

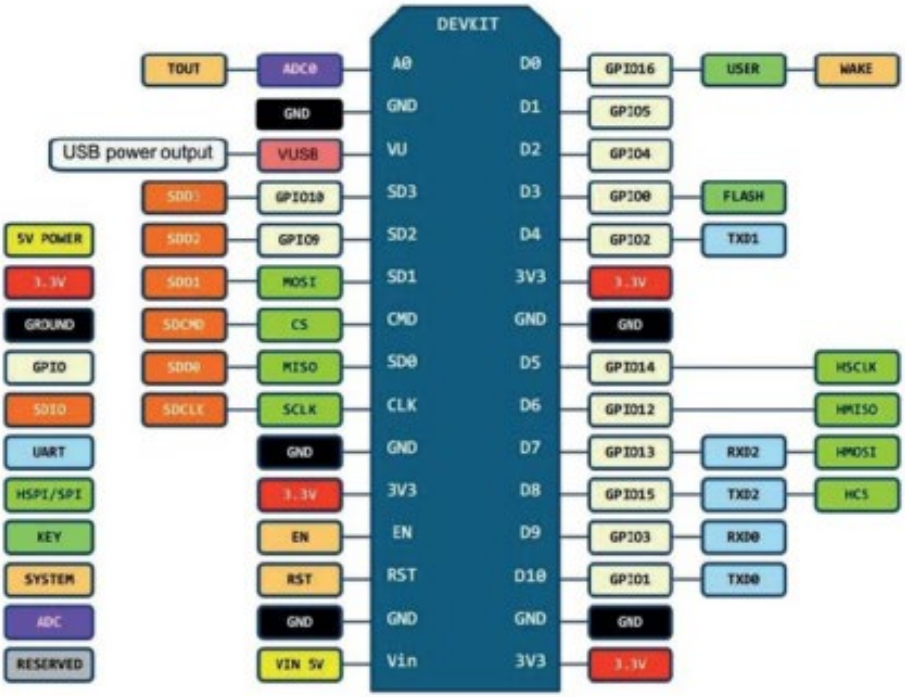
Pin	Function	Type	Description
15	D3	Digital/GPIO	Digital input 3/GPIO
16	D2	Digital/GPIO	Digital input 2/GPIO
17	D1/TX0	Digital/GPIO	Digital input 1 /GPIO
18	D0/Tx1	Digital/GPIO	Digital input 0 /GPIO
19	D14	Digital/GPIO	Digital input 14 /GPIO
20	D15	Digital/GPIO	Digital input 15 /GPIO
21	D16	Digital/GPIO	Digital input 16 /GPIO
22	D17	Digital/GPIO	Digital input 17 /GPIO
23	D18	Digital/GPIO	Digital input 18 /GPIO
24	D19	Digital/GPIO	Digital input 19 /GPIO
25	D20	Digital/GPIO	Digital input 20 /GPIO
26	D21	Digital/GPIO	Digital input 21 /GPIO



Arduino Mega Pinout

Додаток Б

Специфікації та розводка мікроконтролера ESP8266



Overview of the most important information

» Programming via micro USB-B-cable

» Power supply via:
  » Micro USB-B on the USB port of the computer
  » Micro USB-B on the 5V USB power adapter

» 11 digital I / O-Pins (3,3V!)
» 1 analog I / O-Pin
» ESP-12E processor with ESP8266 WLAN module
» CH340 USB interface
» Programmable via Code and Lua

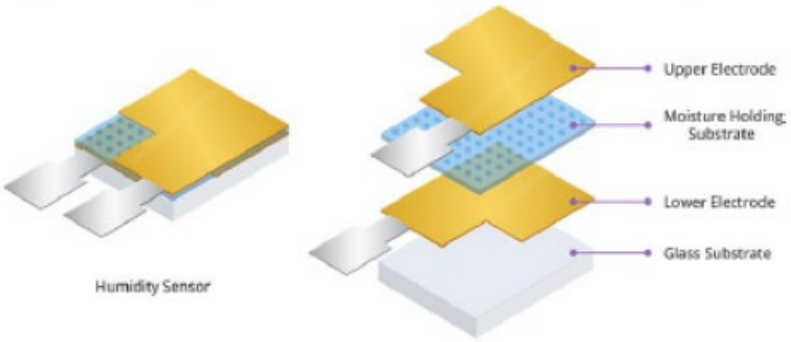
Додаток В

Специфікації та розводка DHT11

Specifications

Operating voltage range:	from 3.3V to 5V DC
Max operating current:	2.5mA max
Humidity range:	20% - 90% with accuracy of 5%
Temperature range:	0°C - 50°C with accuracy of ±2°C
Sampling rate:	1Hz (reading once per second)
Dimensions:	15 x 32 x 9mm [0.6 x 1.3 x 0.35in]

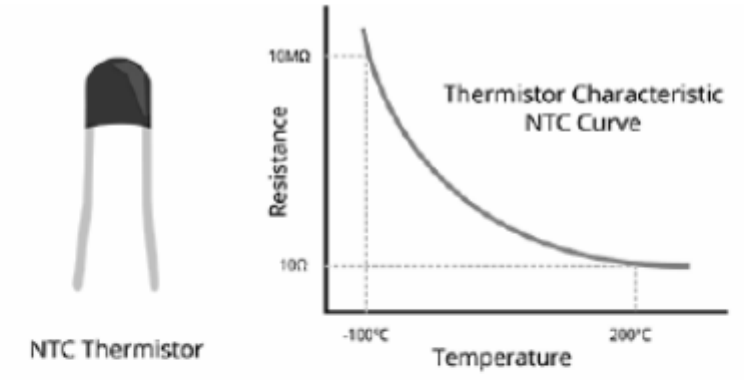
Inside the case, on the sensing side of the DHT11 sensor, there is a humidity sensing component along with an NTC temperature sensor (or thermistor).



The Humidity sensing component is used to measure relative humidity, it has two electrodes with moisture holding substrate (usually salt or conductive plastic polymer) sandwiched between electrodes. The ions are released by the substrate as water vapor is absorbed by it, which in turn increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity.

Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes.

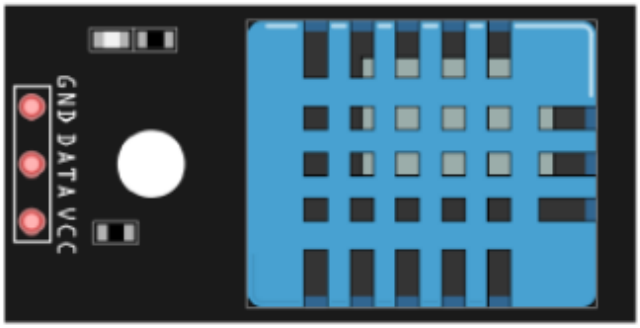
The temperature sensing part of the sensor consists of an NTC temperature sensor (thermistor). A thermistor is a thermal resistor, a resistor that changes its resistance with temperature. Technically, all resistors are thermistors, their resistance changes slightly with temperature, but the change is usually small and difficult to measure. Thermistors are made so that the resistance changes drastically with temperature. The resistance change can be up to *100*Ω per degree of temperature. The term *NTC* means *Negative Temperature Coefficient*, which means that the resistance decreases with the increase of temperature.



On the other side, there is a small *PCB* with an *8-bit SOIC-14* packaged integrated circuit (IC for short). The IC measures and processes the analog signal with stored calibration coefficients, does analog to digital conversion and outputs a digital signal with the data that contains information for temperature and humidity.

The pinout

GROUND - GND
DATA OUTPUT - DATA
POWER SUPPLY - VCC



***VCC* pin** - supplies power for the sensor. Although supply voltage can range between *3.3V* and *5V*, a *5V* supply is recommended. In the case of a *5V* power supply, a cable that connects sensor and microcontroller can be up to *20* meters long. However, with *3.3V* supply voltage, cable length shall not be longer than one meter, otherwise, the line voltage drop will lead to errors in measurement.

***DATA* pin** - is the data pin and it is used for communication between the sensor and the microcontroller.

***GND* pin** - is a ground pin and should be connected to the common ground or *0V* (on Uno or Raspberry Pi).

Додаток Г

Технічні характеристики та розводка MQ-2

Specifications

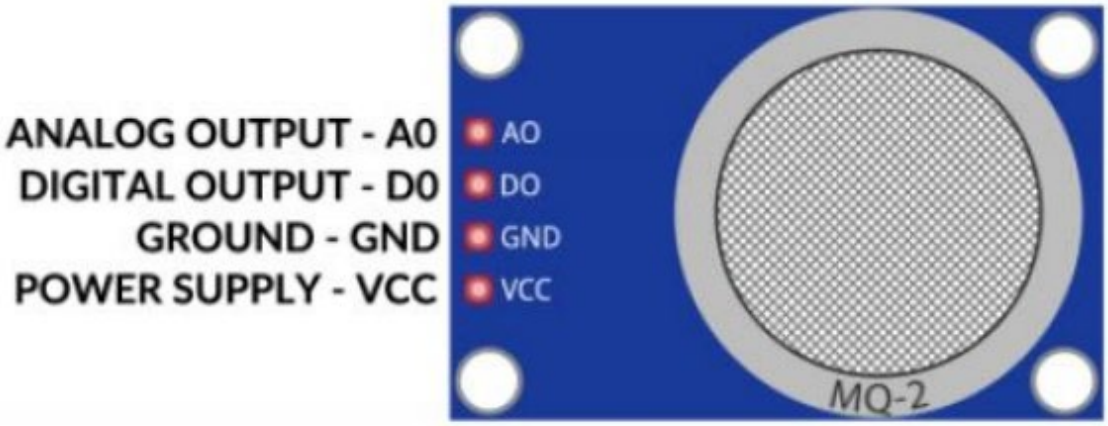
Operating voltage:	5V
Operating current:	150mA
Power consumption:	900mW
Load resistance:	20kΩ
Heater resistance:	33Ω+5%
Sensing resistance	10kΩ - 60kΩ
Preheat time:	24h
Concentration scope:	200 – 10000ppm (parts per million)
Output:	analog, digital
Dimensions:	33x21x22mm (1.3x0.8x0.9in)

For the best detecting results, gas sensor has to be preheated. The best preheat time for the sensor is above 24 hours. For the detailed information about the sensor specifications, refer to the datasheet.

The module sensitivity can be adjusted with an on-board potentiometer. Moving the potentiometer shaft into the clockwise direction increases the sensitivity. Moving the shaft of the potentiometer in the counterclockwise direction decreases the sensitivity of the module.

The pinout

The gas sensor module has four pins. The pinout is shown on the following image:



NOTE: The Raspberry Pi does not have a digital-analog converter and can not be used to read analog voltages.

Додаток Ґ

Технічні характеристики та розводка MQ-135

Specifications

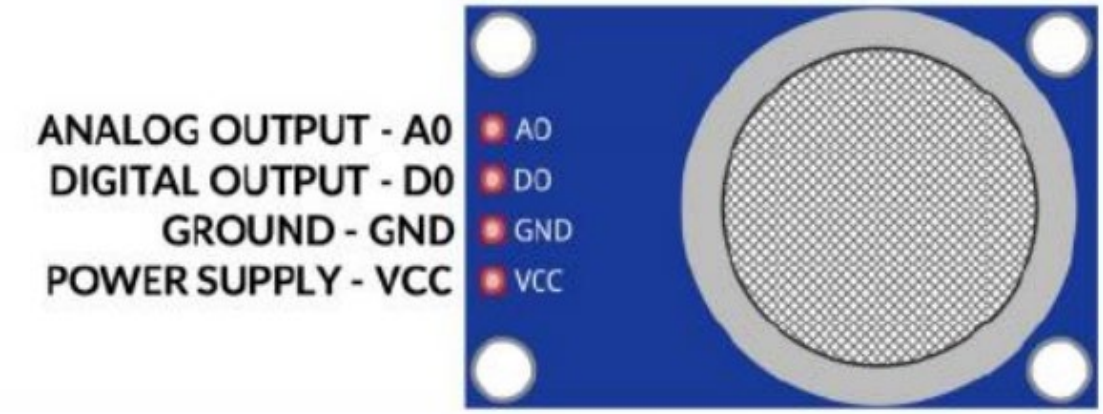
Operating voltage:	5V
Operating current:	150mA
Power consumption:	900mW
Load resistance:	20kΩ
Heater resistance:	31Ω+5%
Sensing resistance	2kΩ - 20kΩ
Preheat time:	24h
Concentration scope:	200 – 10000ppm (parts per million)
Output:	analog, digital
Dimensions:	33x21x22mm (1.3x0.8x0.9in)

For the best detecting results, gas sensor has to be preheated. The best preheat time for the sensor is above 48 hours. For the detailed information about the sensor specifications, refer to the datasheet.

The module sensitivity can be adjusted with an on-board potentiometer. Moving the potentiometer shaft into the clockwise direction increases the sensitivity. Moving the shaft of the potentiometer in the counterclockwise direction decreases the sensitivity of the module.

The pinout

The gas sensor module has four pins. The pinout is shown on the following image:



NOTE: The Raspberry Pi does not have a digital-analog converter and can not be used to read analog voltages.

Технічні характеристики та розводка KY-037

Specifications

- Operating voltage range: from 3.3V to 5V DC
- Output: digital and analog
- Frequency range: from 100Hz to 10.000Hz
- Sensitivity: -46 ± 2.0 (0 dB = 1V/Pa) at 1kHz
- Sensitivity to Noise Ratio: 58dB
- Dimensions: 15 x 36 x 13mm [0.6 x 1.4 x 0.51in]

There are two LEDs on-board the module, one for indicating power supply and one for indicating the intensity of the sound.

The analog output pin voltage sensitivity can be adjusted via an on-board blue potentiometer. With some adjustment of the potentiometer, when the analog voltage is read via analog to digital converter (ADC for short), the digital value of the voltage is around the value *595*, and that is when there is not sound near the KY-037 module.

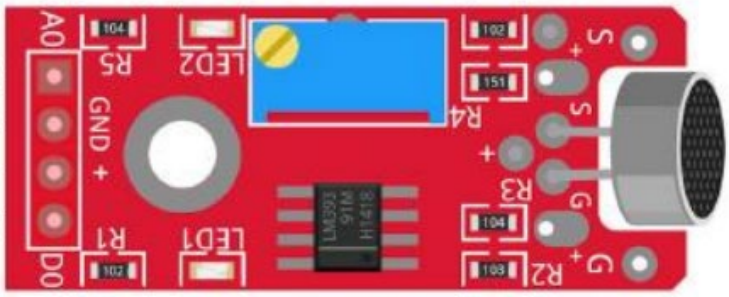
When there is a sound near the sensor, the ADC value changes dramatically around the value *595*, rise and drop a few times. This happens because of the nature of sound waves. This can be seen on the image of the Serial Plotter (later in the text).

The state when the digital output pin outputs the *HIGH* state (is turned *ON*) is the state when the ADC value is grater than the value of *700*.

The pinout

The KY-037 big microphone module has four pins. The pinout diagram is shown on the following image:

Analog output pin - A0
Ground - G
Power supply - +
Digital output pin - D0



Додаток Е

Технічні характеристики BMP-180 та розклад виводів

1. Electrical characteristics

If not stated otherwise, the given values are ±3-Sigma values over temperature/voltage range in the given operation mode. All values represent the new parts specification; additional solder drift is shown separately.

Table 1: Operating conditions, output signal and mechanical characteristics

Parameter	Symbol	Condition	Min	Typ	Max	Units
Operating temperature	T_A	operational	-40		+85	°C
		full accuracy	0		+65	
Supply voltage	V_{DD}	ripple max. 50mVpp	1.8	2.5	3.6	V
			1.62	2.5	3.6	
Supply current @ 1 sample / sec. 25°C	I_{DDLOW}	ultra low power mode		3		µA
	I_{DDSTD}	standard mode		5		µA
	I_{DDHR}	high resolution mode		7		µA
	I_{DDUHR}	Ultra high res. mode		12		µA
	I_{DDAR}	Advanced res. mode		32		µA
Peak current	I_{peak}	during conversion		650	1000	µA
Standby current	I_{DDSBM}	@ 25°C		0.1	4[1]	µA
Relative accuracy pressure V_{DD} = 3.3V		950 . . . 1050 hPa @ 25 °C		±0.12		hPa
				±1.0		m
		700 ... 900hPa 25 . . . 40 °C		±0.12		hPa
				±1.0		m
Absolute accuracy pressure V_{DD} = 3.3V		300 . . . 1100 hPa 0 . . . +65 °C	-4.0	-1.0*	+2.0	hPa
		300 . . . 1100 hPa -20 . . . 0 °C	-6.0	-1.0*	+4.5	hPa
Resolution of output data		pressure		0.01		hPa
		temperature		0.1		°C
Noise in pressure		see table on page 12-13				
Absolute accuracy temperature V_{DD} = 3.3V		@ 25 °C	-1.5	±0.5	+1.5	°C
		0 . . . +65 °C	-2.0	±1.0	+2.0	°C

[1] at 85°C

Conversion time pressure	$t_{c_p_low}$	ultra low power mode		3	4.5	ms
	$t_{c_p_std}$	standard mode		5	7.5	ms
	$t_{c_p_hr}$	high resolution mode		9	13.5	ms
	$t_{c_p_luhr}$	ultra high res. mode		17	25.5	ms
	$t_{c_p_ar}$	Advanced res. mode		51	76.5	ms
Conversion time temperature	t_{C_temp}	standard mode		3	4.5	ms
Serial data clock	f_{SCL}				3.4	MHz
Solder drifts		Minimum solder height 50µm	-0.5		+2	hPa
Long term stability**		12 months		±1.0		hPa

* The typical value is: -1±1
** Long term stability is specified in the full accuracy operating pressure range 0 ... 65°C

Додаток Є

Програмний код для датчиків Arduino Mega

```
#include <SoftwareSerial.h>
#include <DHT.h>
#include <Wire.h>
#include <Adafruit_BMP085.h>

const int mq135AnalogPin = A0;
const int mq135DigitalPin = 2;
const int mq2AnalogPin = A1;
const int mq2DigitalPin = 3;
const int soundPin = A2;
#define DHTPIN 4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
Adafruit_BMP085 bmp;

String dataToSend; // Declaración de la variable dataToSend
SoftwareSerial espSerial(19, 18);  // RX, TX

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

  pinMode(mq135DigitalPin, INPUT);
  pinMode(mq2DigitalPin, INPUT);

  dht.begin();
  if (!bmp.begin()) {
    Serial.println("No se pudo encontrar un sensor BMP085. Verifique las conexiones o la dirección I2C.");
    while (true);
  }
}
```

```
void loop() {
  int s_gasmq135 = analogRead(mq135AnalogPin);
  int mq2AnalogValue = analogRead(mq2AnalogPin);
  int s_sound = analogRead(soundPin);
  float s_temperature = dht.readTemperature();
  float s_humidity = dht.readHumidity();
  float s_gasmq2 = map(mq2AnalogValue, 0, 1023, 0, 100);
  float bmpTemperature = bmp.readTemperature();
  float s_pressure = bmp.readPressure() / 100.0F;

  dataToSend = "temperature=" + String(s_temperature) + "&humidity=" + String(s_humidity) + "&gasmq135=" +
  String(s_gasmq135) + "&gasmq2=" + String(s_gasmq2) + "&sound=" + String(s_sound) + "&pressure=" + String(s_pressure) + "\n";

  Serial.println(dataToSend);
  espSerial.println(dataToSend);

  delay(3000);
}
```

Додаток Ж

Програмний код ESP8266

```
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>

const char* ssid = "-";
const char* password = "-";

const char* host = "--";
const uint16_t port = 80;

const int RX_PIN = D7;
const int TX_PIN = D8;

SoftwareSerial arduinoSerial(RX_PIN, TX_PIN);
String receivedData;

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

  Serial.println();
  Serial.println("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println(".");
  }

  Serial.println("");
  Serial.println("WiFi Connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  readFromArduino();
  if (!receivedData.isEmpty()) {
    sendDataToHost(receivedData);
    receivedData = "";
  }
  delay(10000);
}

void readFromArduino() {
  if (arduinoSerial.available()) {
    receivedData = arduinoSerial.readStringUntil('\n');
    Serial.println("Received data from Arduino Mega: " + receivedData);

    arduinoSerial.flush();
    delay(10);

  }
}

void sendDataToHost(String data) {
  Serial.print(">>>>>> connecting to ");
  Serial.print(host);
  Serial.print(':');
  Serial.println(port);
```

```
62
63    WiFiClient client;
64    if (!client.connect(host, port)) {
65      Serial.println("connection failed");
66      delay(5000);
67      return;
68    }
69
70    Serial.println("[Sending a request]");
71    client.print(String("GET /paginaweb/save.php?") + data + " HTTP/1.1\r\n" +
72                 "Host: " + host + "\r\n" +
73                 "Connection: close\r\n\r\n");
74
75    unsigned long timeout = millis();
76    while (client.available() == 0) {
77      if (millis() - timeout > 5000) {
78        Serial.println(">>> Client timeout");
79        client.stop();
80        delay(3000);
81        return;
82      }
83    }
84
85    Serial.println("receiving from the remote server");
86    String msg;
87    while (client.available()) {
88      char ch = static_cast<char>(client.read());
89      Serial.print(ch);
90      msg += ch;
91    }
92    Serial.println();
93    if (msg.indexOf("Guardado Correcto") != -1) {
94      Serial.println("DataSaved");
95    } else {
96      Serial.println("Could not save the data");
97    }
98
99    Serial.println();
100   Serial.println(">>>>>>closing connection");
101   Serial.println();
102   client.stop();
103 }
104
```

Програмний код веб-сторінки

connection.php

```php
<?php

$host = "localhost";
$username = "root";
$password = "";
$dbname = "datatfgproject";

try {

    $conn = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo '';
} catch (PDOException $e) {
    die('Error: ' . $e->getMessage());
}
?>

```

save.php

```php
<?php
    include('connection.php');

    $temperature = $_GET['temperature'];
    $humidity = $_GET['humidity'];
    $gas_mq135 = $_GET['gasmq135'];
    $gas_mq2 = $_GET['gasmq2'];
    $sound = $_GET['sound'];
    $pressure = $_GET['pressure'];

    $mysql = "INSERT INTO `sensor_data` (`temperature`, `humidity`, `gas_mq135`, `gas_mq2`, `sound`, `pressure`)
    VALUES (:temperature, :humidity, :gas_mq135, :gas_mq2, :sound, :pressure)";

    $stmt = $conn->prepare($mysql);
    $stmt->bindParam(':temperature', $temperature);
    $stmt->bindParam(':humidity', $humidity);
    $stmt->bindParam(':gas_mq135', $gas_mq135);
    $stmt->bindParam(':gas_mq2', $gas_mq2);
    $stmt->bindParam(':sound', $sound);
    $stmt->bindParam(':pressure', $pressure);

    if ($stmt->execute()){
        echo "Guardado correctamente!";
    } else {
        echo "Error al guardar";
    }


?>
```

package.json > ...

```json
{
  "name": "paginaweb",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Alejandro",
  "license": "ISC",
  "dependencies": {
    "bootstrap": "^5.3.2",
    "chart.js": "^4.4.1",
    "express": "^4.18.2",
    "jquery": "^3.7.1",
    "popper.js": "^1.16.1",
    "serialport": "^12.0.0",
    "socket.io": "^4.7.2"
  }
}
```

```
deathproject.php
<!DOCTYPE html>
<html lang="en">
<head>

    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>TFG Alejandro Lescano</title>
    <link rel="stylesheet" href="style.css">


    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
    integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

</head>

<body>

    <header class="header">
        <div class="menu container">
            <link rel="stylesheet" type="text/css" href="style.css">
            <a href="#"class= "logo">Széchenyi István University</a>
            <input type="checkbox" id="menu">
            <label for="menu">
                <img src="images/menu.png" class="menu-icono" alt="">
            </label>
            <nav class="navbar">
                <ul>
                    <li><a href="#">Start</a></li>
                    <li><a href="#">Real time Values</a></li>
                    <li><a href="#">Hardware</a></li>
                    <li><a href="#">Historical Data</a></li>
                    <li><a href="#">Contact</a></li>
                </ul>
            </nav>
        </div>

        <div class="header-content container">
            <h1>University IoT sensors with Arduino</h1>
            <h2>Project Description</h2>
            <p>The integration of the IoT technology (Internet of Things) in the university environment has led to an improvement and advancement in current study methods.
                This trend takes advantage of the ability of IoT devices to collect data in real time and transmit it over the network.
                This project consists of the implementation of different sensors in different locations of the Széchenyi István University of Győr.
                With the use of Arduino microcontrollers, it will allow us to transmit this data to our website for subsequent processing.
            </p>
            <a href="#" class="btn-1">More Information</a>
        </div>

    </header>

    <section class= "general container">
        <h2>Hardware</h2>
        <div class="general-content-hardware">
            <div class="general- 1 text">
                <h3>Microcontroller</h3>
                <p>A microcontroller is a programmable integrated circuit, capable of executing commands recorded in its memory.
                    To carry out this project, Arduino Mega has been chosen as the microcontroller prototype.
                    Arduino Mega is a board based on the ATmega2560 microcontroller, which offers a wide variety of input/output pins,
                    making it ideal for projects that require a large number of connected devices or sensors.
                </p>
                <img src="images/Arduino_Mega.png" alt="Arduino Mega">
            </div>
            <div class="general- 2 text">
                <h3>Module ESP8266</h3>
                <p>
                    NodeMCU is a completely open development board, at the software and hardware level. As with Arduino, in NodeMCU
                    everything is arranged to facilitate the programming of a microcontroller. This model incorporates the ESP8266
                    module which is a microcontroller with integrated Wi-Fi connectivity. This will act as an intermediary between
                    our Arduino Mega (and sensors) with our data platform.
                </p>
                <img src="images/ESP8266.png" alt="NodeMCU ESP8266">

            </div>
        </div>
        <div>
            <h2>Used sensors</h2>
        </div>

        <div class="general-content-used-sensors"></div>

            <div class="general- 3 text">
                <h4>DHT11</h4>
                    <p>
                        The DHT11 sensor is used to measure the temperature and relative humidity of the air in the environment.
                        It is widely used in climate monitoring and HVAC control projects, as well as home automation systems.
                    </p>
                    <img src="images/dht11.png" alt="DHT11">
            </div>

            <div class="general- 4 text">
                <h4>MQ-135</h4>
                    <p>
                        The MQ-135 sensor is used to detect harmful gases such as carbon dioxide (CO2), ammonia (NH3),
                        carbon monoxide (CO) and other hazardous gases. It is used in air quality control systems, indoor air quality
                        monitoring and gas alarm systems.
                    </p>
                    <img src="images/Mq-135.png" alt="MQ-135">
            </div>

            <div class="general- 5 text">
                <h4>MQ-2</h4>
                    <p>
                        The MQ-2 sensor is used to detect flammable gases, smoke, carbon monoxide and flammable gases such as
                        methane and butane. It is commonly used in fire alarm systems, gas leak detectors and security systems.
                    </p>
                    <img src="images/MQ2-1.png" alt="MQ-2">
            </div>
```

```
            <div class="general- 6 text">
                <h3>Module KY-037</h3>
                    <p>
                        The KY-037 sound module is used to detect sounds or noises in the environment. It can be part
                        of noise alarm projects, sound activated audio recording systems or sound control projects.
                    </p>
                    <img src="images/KY-037.png" alt="KY-037">
            </div>

            <div class="general- 6 text">
                <h3>BMP-180</h3>
                    <p>
                        The BMP-180 sensor is used to measure atmospheric pressure and ambient temperature. It is common in
                        weather projects, personal weather stations, and navigation systems that require altitude data.
                    </p>
                    <img src="images/BMP-180.png" alt="BMP-180">
            </div>
        </div>
    </section >

    <body>
        <div id="datos-historicos">
            <h1>HISTORICAL ANALYSIS</h1>
            <h2>Sensor value history</h2>
            <p>Here the historical data of the sensors will be displayed</p>
            <ol id="datos-historicos">

            </ol>

        </div>

        <div class="container">
            <div class="row">
              <div class="col-12 col-lg-4">
                <div class="text-center">
                <h3>Temperature</h3>
```

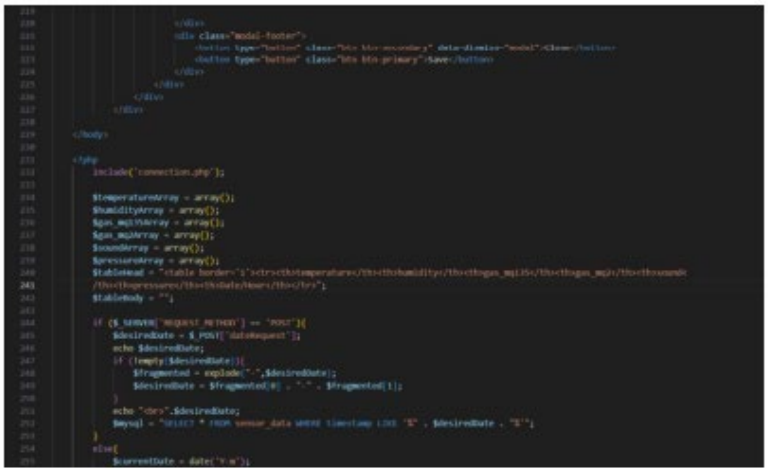
```
                </div>
                <canvas id="temperatureChart" width="400" height="400"></canvas>
              </div>
              <div class="col-12 col-lg-4">
                <div class="text-center">
                <h3>Humidity</h3>
                </div>
                <canvas id="humidityChart" width="400" height="400"></canvas>
              </div>
              <div class="col-12 col-lg-4">
                <div class="text-center">
                <h3>GasMQ135</h3>
                </div>
                <canvas id="gas_mq135Chart" width="400" height="400"></canvas>
              </div>
            </div>
            <div class="row">
              <div class="col-12 col-lg-4">
                <div class="text-center">
                <h3>GasMQ2</h3>
                </div>
                <canvas id="gas_mq2Chart" width="400" height="400"></canvas>
              </div>
              <div class="col-12 col-lg-4">
                <div class="text-center">
                <h3>Sound</h3>
                </div>
                <canvas id="soundchart" width="400" height="400"></canvas>
              </div>
              <div class="col-12 col-lg-4">
                <div class="text-center">
                <h3>Pressure</h3>
                </div>
                <canvas id="pressurechart" width="400" height="400"></canvas>
              </div>
```

```
        </div>
        <div class="container d-flex justify-content-center mt-5">
            <div class="col-12 col-lg-4">
              <div class="row">
                <h3 style="margin:10px 0 20px 0;" class="col-12 text-center text-uppercase">Date Picker</h3>
              </div>
              <div class="row col-12 d-flex justify-content-center">
                <form action="" method="POST" class="form-group">
                  <input type="date" class="form-control" name="dateRequest" id="">
                  <button style="margin-top:7px;" name="submit" class="btn btn-sm btn-primary" type="submit">Search</button>
                </form>
              </div>
              <div class="row col-12 d-flex justify-content-center">
                <button style="margin-top:15px;" name="button" class="form-control btn btn-success" data-toggle="modal" data-target="#tableModal">Show Table</button>
              </div>
            </div>
          </div>

        </div>

        <div class="modal fade" id="tableModal" tabindex="-1" role="dialog" aria-hidden="true">
            <div class="modal-dialog modal-lg" role="document">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title">DATA TABLE</h5>
                            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                                <span aria-hidden="true">&times;</span>
                            </button>
                    </div>
                    <div id="modalBody" class="modal-body">
                        <div class="table-responsive">
                            <table class="table table-bordered">

                            </table>
                        </div>
```



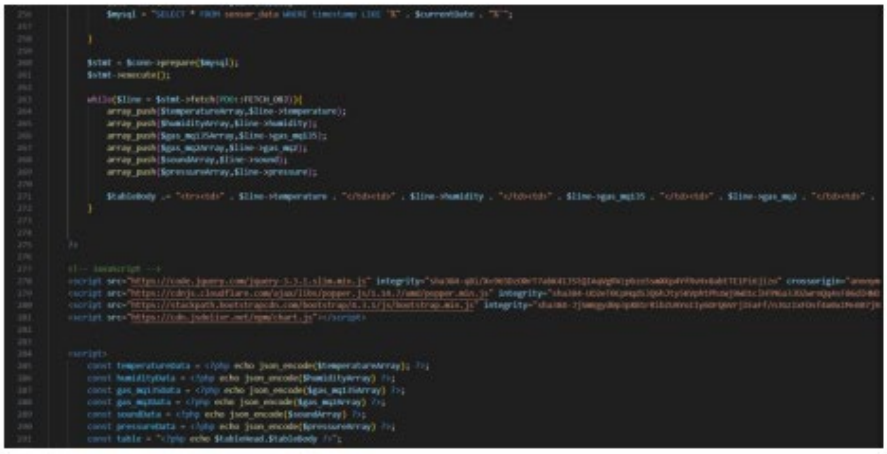
```
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                        <button type="button" class="btn btn-primary">Save</button>
                    </div>
                </div>
            </div>
        </div>

</body>

<?php
    include('connection.php');

    $temperatureArray = array();
    $humidityArray = array();
    $gas_mq135Array = array();
    $gas_mq2Array = array();
    $soundArray = array();
    $pressureArray = array();
    $tableHead = "<table border='1'><tr><th>temperature</th><th>humidity</th><th>gas_mq135</th><th>gas_mq2
    /th><th>pressure</th><th>Date/Hour</th></tr>";
    $tableBody = "";

    if ($_SERVER['REQUEST_METHOD'] == 'POST'){
        $desiredDate = $_POST['dateRequest'];
        echo $desiredDate;
        if (!empty($desiredDate)){
            $fragmented = explode("-",$desiredDate);
            $desiredDate = $fragmented[0] . "-" . $fragmented[1];
        }
        echo "<br>".$desiredDate;
        $mysql = "SELECT * FROM sensor_data WHERE timestamp LIKE '%" . $desiredDate . "%'";
    }
    else{
        $currentDate = date('Y-m');
```



```
        $mysql = "SELECT * FROM sensor_data WHERE timestamp LIKE '%" . $currentDate . "%'";

    }

    $stmt = $conn->prepare($mysql);
    $stmt->execute();

    while($line = $stmt->fetch(PDO::FETCH_OBJ)){
        array_push($temperatureArray,$line->temperature);
        array_push($humidityArray,$line->humidity);
        array_push($gas_mq135Array,$line->gas_mq135);
        array_push($gas_mq2Array,$line->gas_mq2);
        array_push($soundArray,$line->sound);
        array_push($pressureArray,$line->pressure);

        $tableBody .= "<tr><td>" . $line->temperature . "</td><td>" . $line->humidity . "</td><td>" . $line->gas_mq135 . "</td><td>" . $line->gas_mq2 . "</td><td>" .
    }


?>

<!-- JavaScript -->
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonym
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHN
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jR
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>


<script>
    const temperatureData = <?php echo json_encode($temperatureArray); ?>;
    const humidityData = <?php echo json_encode($humidityArray) ?>;
    const gas_mq135Data = <?php echo json_encode($gas_mq135Array) ?>;
    const gas_mq2Data = <?php echo json_encode($gas_mq2Array) ?>;
    const soundData = <?php echo json_encode($soundArray) ?>;
    const pressureData = <?php echo json_encode($pressureArray) ?>;
    const table = "<?php echo $tableHead.$tableBody ?>";
```



```
    let divTable = document.createElement("div");
    divTable.innerHTML = table;
    divTable.className = "text-center";
    document.querySelector("#modalBody").appendChild(divTable);

    let samples = [];
    let dataLength = Object.keys(temperatureData).length;
    for (let i = 0; i < dataLength; i++) {
        samples.push(i+1);
    }

    //var temperatureCtx = document.getElementById('temperatureChart');
    //var humidityCtx = document.getElementById('humidityChart');
    //var gas_mq135Ctx = document.getElementById('gas_mq135Chart');
    //var gas_mq2Ctx = document.getElementById('gas_mq2Chart');
    //var soundCtx = document.getElementById("soundChart");
    //var pressureCtx = document.getElementById("pressureChart");

    let ctxTemperature = document.getElementById('temperatureChart').getContext('2d');
    let chartTemperature = new Chart(ctxTemperature, {
        type: 'line',
        data: {
            labels: samples,
            datasets: [{
                label: "°C",
                backgroundColor: 'rgba(255, 206, 86, 0.2)',
                borderColor: 'rgba(255, 206, 86, 1)',
                borderWidth: 4,
                data: temperatureData
            }],
        },
        options: {
            responsive: true,
            title: {
```

```
                display: true,
                fontColor: "#fff",
                fontSize: 16,
                fontFamily: 'Trebuchet MS',
                text: 'TEMPERATURE'
            },
            legend: {
                labels: {
                    fontColor: "black",
                    fontSize: 15
                }
            },
            scales: {
                yAxes: [{
                    ticks: {
                        fontColor: "black",
                        fontSize: 12,
                    }
                }],
                xAxes: [{
                    ticks: {
                        fontColor: "black",
                        fontSize: 11,
                    }
                }]
            }
        }
    });

    let ctxHumidity = document.getElementById('humidityChart').getContext('2d');
    let chartHumidity = new Chart(ctxHumidity, {
        type: 'line',
        data: {
            labels: samples,
            datasets: [{
                label: 'H',
```

```
                backgroundColor: 'rgba(54, 162, 235, 0.3)',
                borderColor: 'rgba(54, 162, 235, 1)',
                borderWidth: 4,
                data: humidityData
            }],
        },
        options: {
            responsive: true,
            title: {
                display: true,
                fontColor: "#fff",
                fontSize: 16,
                fontFamily: 'Trebuchet MS',
                text: 'HUMIDITY'
            },
            legend: {
                labels: {
                    fontColor: "#eee",
                    fontSize: 15
                }
            },
            scales: {
                yAxes: [{
                    ticks: {
                        fontColor: "#eee",
                        fontSize: 12,
                    }
                }],
                xAxes: [{
                    ticks: {
                        fontColor: "#eee",
                        fontSize: 11,
                    }
                }]
            }
        }
```

```
    });


    let ctxgas_mq135 = document.getElementById('gas_mq135Chart').getContext('2d');
    let chartgas_mq135 = new Chart(ctxgas_mq135, {
        type: 'line',
        data: {
            labels: samples,
            datasets: [{
                label: '',
                backgroundColor: 'rgba(255, 70, 100, 0.3)',
                borderColor: 'rgba(255, 70, 100, 1)',
                borderWidth: 4,
                data: gas_mq135data
            }],
        },
        options: {
            responsive: true,
            title: {
                display: true,
                fontColor: "#fff",
                fontSize: 16,
                fontFamily: 'Trebuchet MS',
                text: 'GAS MQ135'
            },
            legend: {
                labels: {
                    fontColor: "#eee",
                    fontSize: 15
                }
            },
            scales: {
                yAxes: [{
                    ticks: {
                        fontColor: "#eee",
                        fontSize: 12,
```

```
            data: {
                labels: samples,
                datasets: [{
                    label: 'hPa',
                    backgroundColor: 'rgba(153, 255, 0, 0.3)',
                    borderColor: 'rgba(153, 255, 0, 1)',
                    borderWidth: 4,
                    data: pressureData
                }],
            },
            options: {
                responsive: true,
                title: {
                    display: true,
                    fontColor: "#fff",
                    fontSize: 16,
                    fontFamily: 'Trebuchet MS',
                    text: 'PRESSURE'
                },
                legend: {
                    labels: {
                        fontColor: "#000",
                        fontSize: 15
                    }
                },
                scales: {
                    yAxes: [{
                        ticks: {
                            fontColor: "#000",
                            fontSize: 12,
                        }
                    }],
                    xAxes: [{
                        ticks: {
                            fontColor: "#000",
                            fontSize: 11,
```

```
                            fontSize: 11,
                        }
                    }]
                }
            }
        });

    </script>

</body>

<footer class="footer container">
        <h1>Contact</h1>
        <form >
            <input class="campo" type="text" placeholder="Name">
            <input class="campo" type="email" placeholder="Email">
            <input type="submit" class="btn-2" value="Send">
        </form>
        <div class="footer-txt">
            <p>For any questions, sign up.</p>

        </div>

    </footer>

```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавра

НА ТЕМУ: «Системи моніторингу якості повітря»

Виконав студент групи КІД-41
Євгеній ПАВЛІЧЕНКО
Керівник дипломної роботи:
Юлія БОРОВА
асистент кафедри

Київ – 2025

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи - розробка доступної, ефективної та адаптивної IoT-системи моніторингу якості повітря, яка дозволяє у режимі реального часу відстежувати параметри навколишнього середовища.

Об'єкт дослідження - процес моніторингу параметрів повітря

Предмет дослідження - IoT-система збору, зберігання та візуалізації параметрів якості повітря на базі Arduino та ESP8266

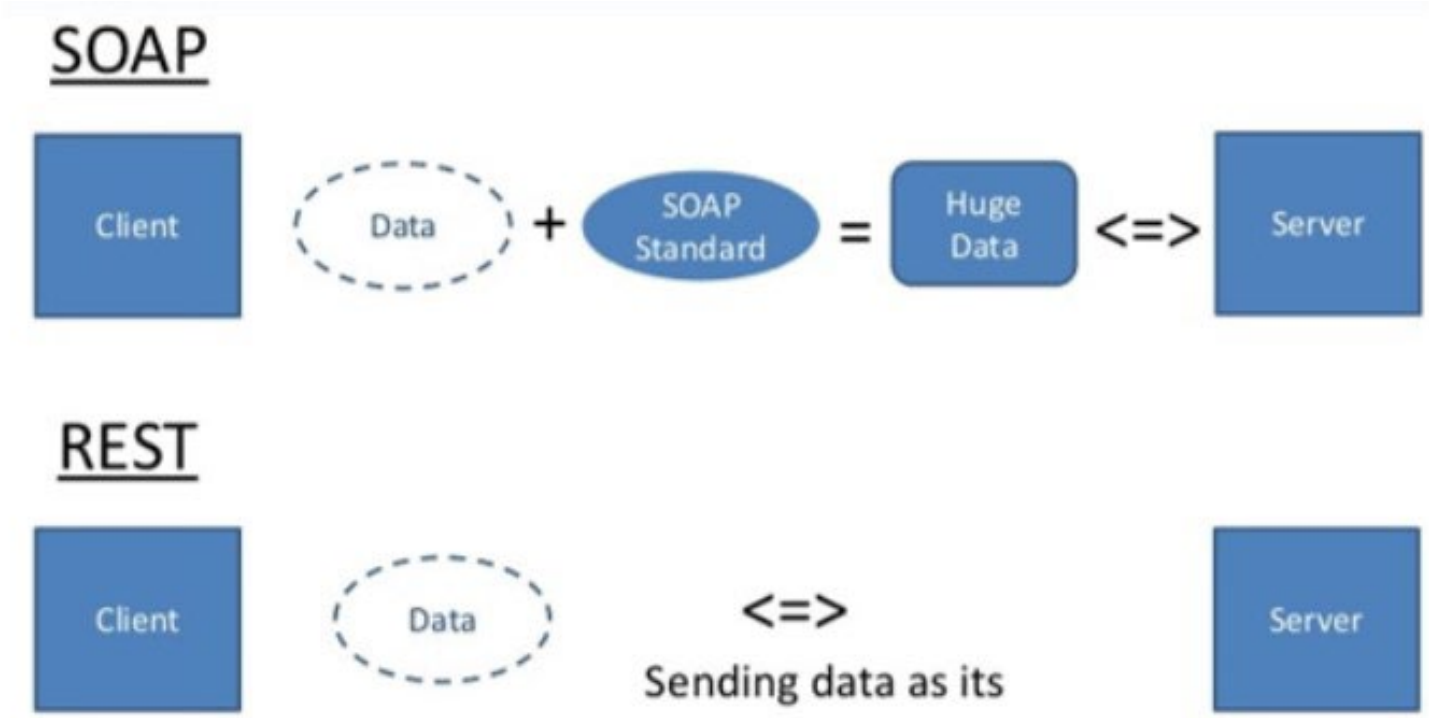
Актуальність роботи

Зміна клімату, зростання рівня забруднення повітря та необхідність екологічного моніторингу стали одними з головних викликів сучасності. Використання технологій Інтернету речей (IoT) дозволяє ефективно здійснювати моніторинг довкілля за допомогою доступних апаратних рішень. Системи збору і аналізу даних у реальному часі з подальшою візуалізацією створюють умови для обґрунтованого прийняття рішень у сфері охорони довкілля. Застосування недорогих пристроїв, таких як Arduino та ESP8266, забезпечує доступність таких систем широкому колу користувачів.

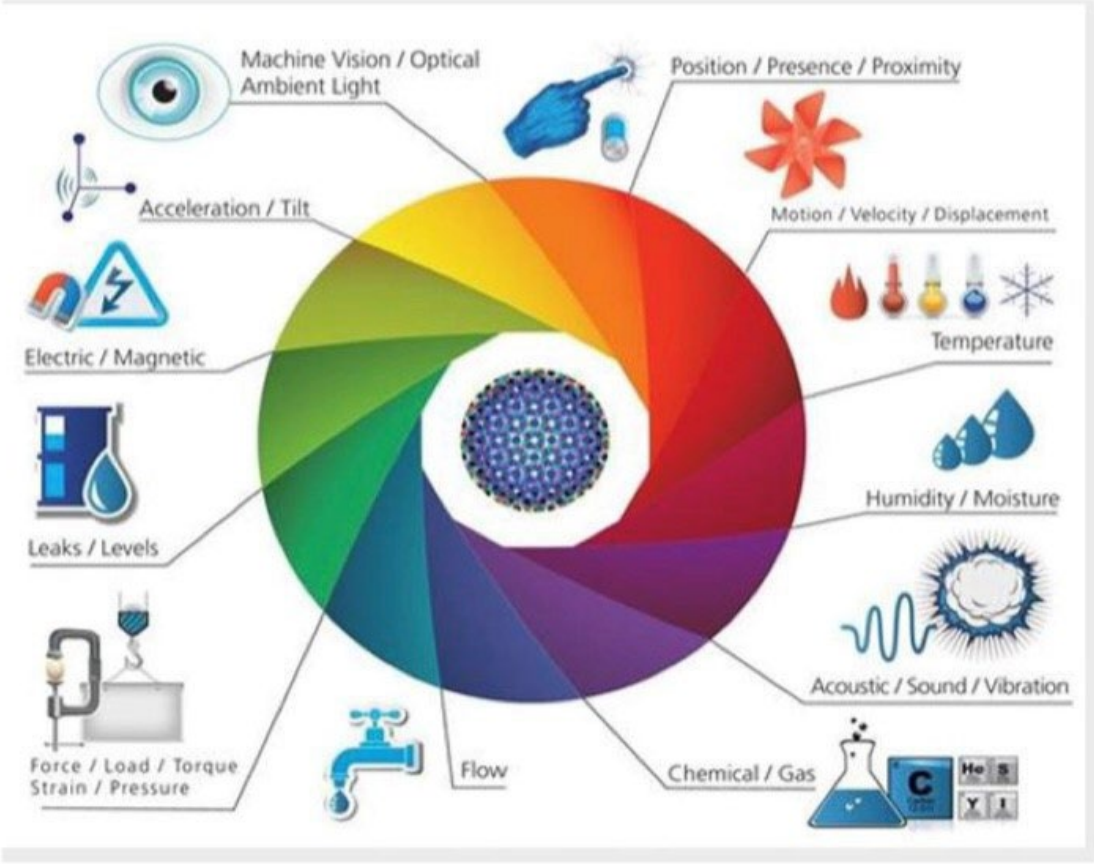
Багаторівнева модель IoT



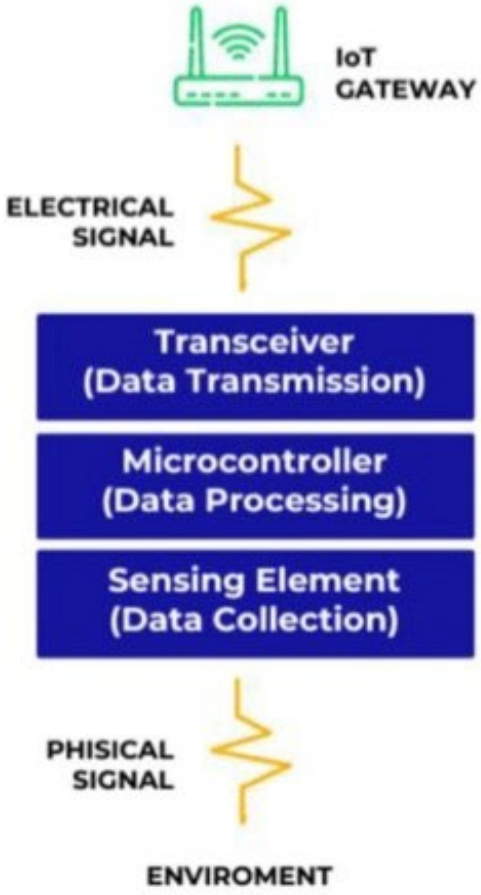
Взаємодія web-сервісів



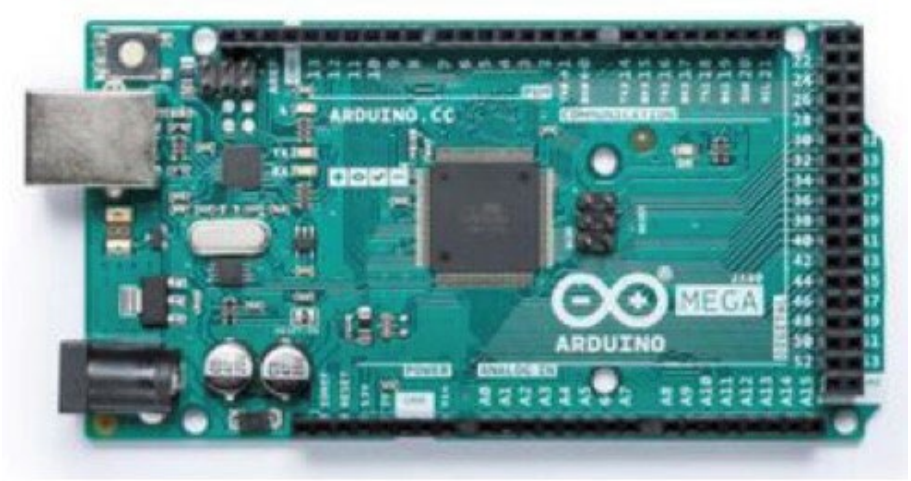
Різні типи датчиків



Компоненти інтелектуального датчика



Модель Arduino Mega 2560




```
void loop() {
  int s_gasmq135 = analogRead(mq135AnalogPin);
  int mq2AnalogValue = analogRead(mq2AnalogPin);
  int s_sound = analogRead(soundPin);
  float s_temperature = dht.readTemperature();
  float s_humidity = dht.readHumidity();
  float s_gasmq2 = map(mq2AnalogValue, 0, 1023, 0, 100);
  float bmpTemperature = bmp.readTemperature();
  float s_pressure = bmp.readPressure() / 100.0F;

  dataToSend = "temperature=" + String(s_temperature) + "&humidity=" + String(s_humidity) + "&gasmq135=" + String(s_gasmq135) + "&gasmq2=" + String(s_gasmq2) + "&sound="

  Serial.println(dataToSend);
  espSerial.println(dataToSend);

  delay(3000);
}
```

Вивід коду датчиків даних

Вузол MCU Lolin V3



```
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>

const char* ssid = "-";
const char* password = "-";

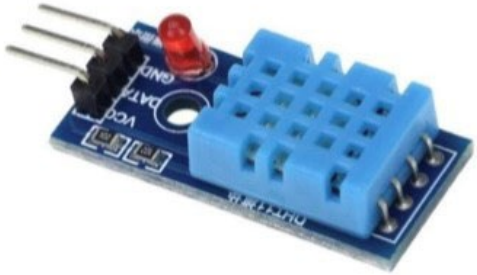
const char* host = "-";
const uint16_t port = 80;

const int RX_PIN = D7;
const int TX_PIN = D8;

SoftwareSerial arduinoSerial(RX_PIN, TX_PIN);
String receivedData;
```

Бібліотеки та змінні, що використовуються у коді ESP8266

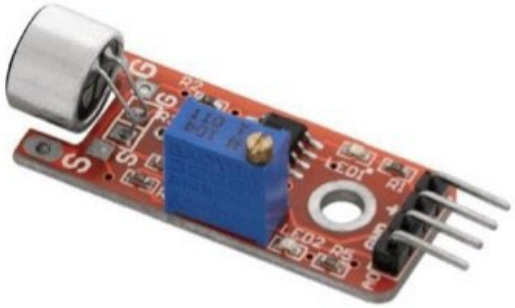
Датчики, що використовуються



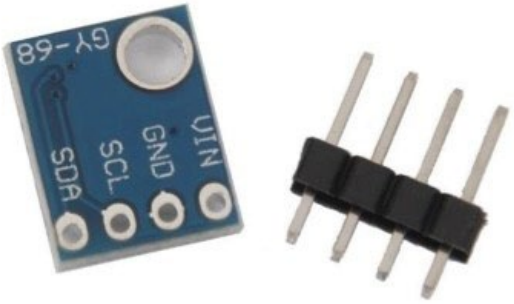
Датчик DHT11



Датчик MQ-2

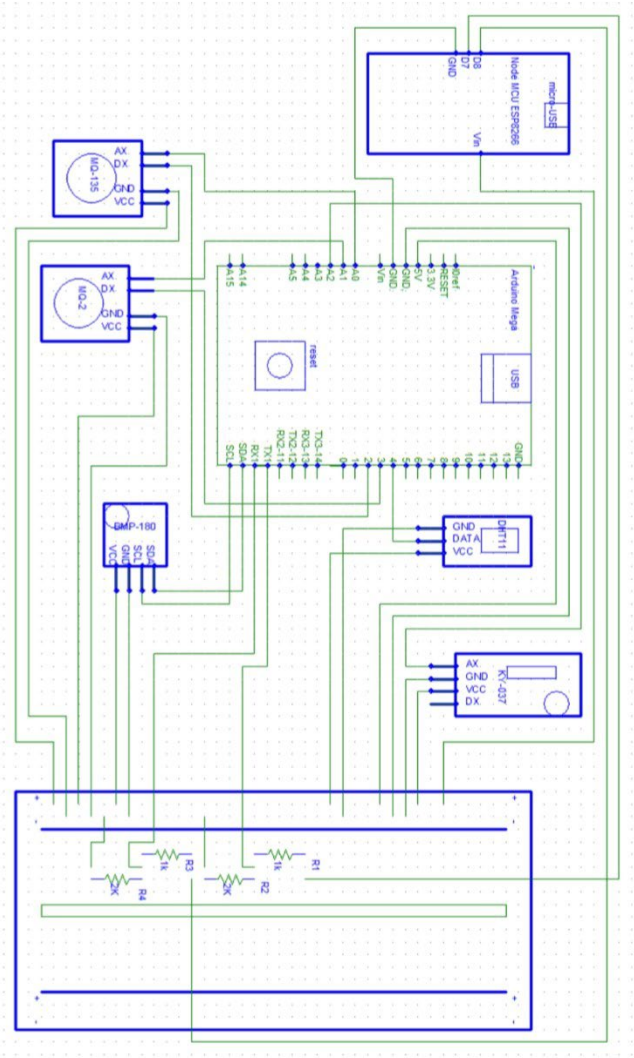


Датчик KY-037

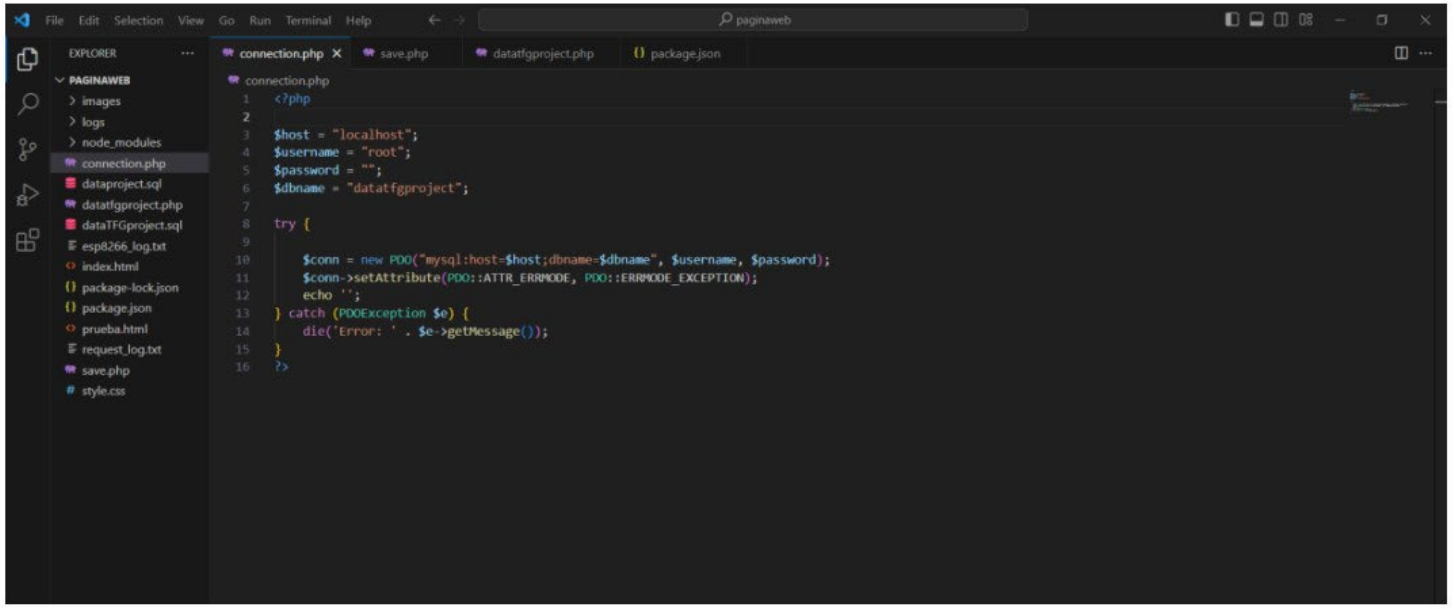


Датчик BMP180

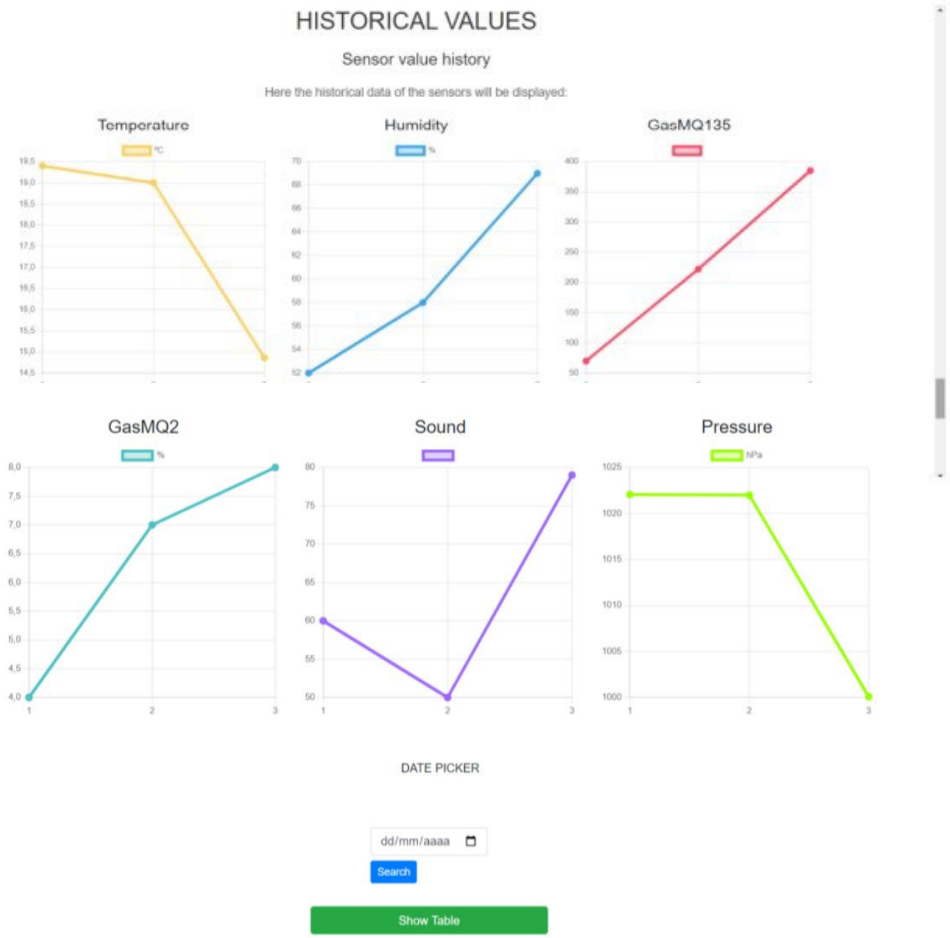
Підключення схеми за допомогою PSpice



Файл підключення до бази даних

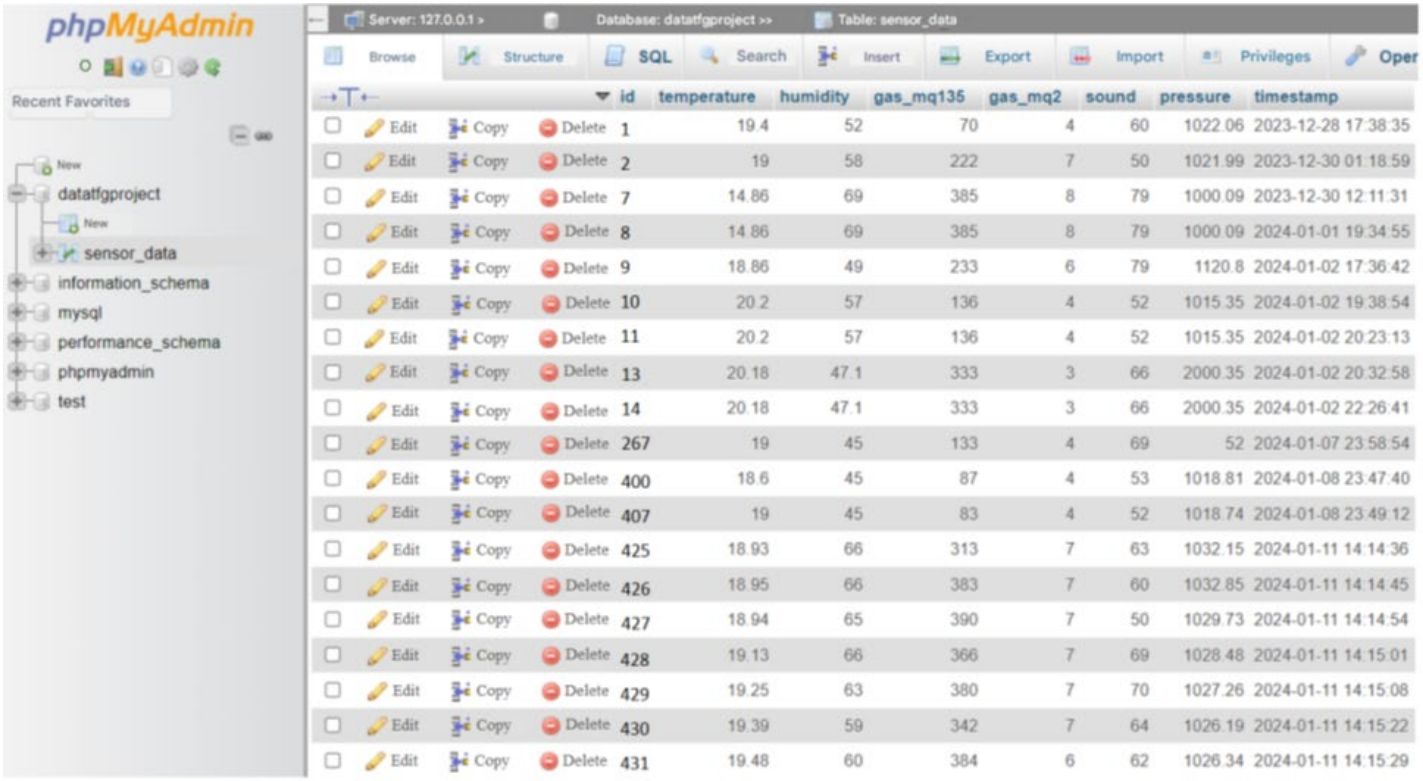


Зразок веб-сайту




Output
Serial Monitor
Message (Enter to send message to 'Arduino Mega or Mega 2560' on 'COM3')
20:19:30.772 -> temperature=21.40&humidity=56.00&gasmq135=257&gasmq2=12.00&sound=99&pressure=1006.47
20:19:30.902 ->
20:19:33.942 -> temperature=21.40&humidity=56.00&gasmq135=252&gasmq2=12.00&sound=98&pressure=1006.46
20:19:34.049 ->
20:19:37.132 -> temperature=21.40&humidity=55.00&gasmq135=241&gasmq2=11.00&sound=94&pressure=1006.45
20:19:37.375 ->
20:22:21.395 -> temperature=21.40&humidity=55.00&gasmq135=115&gasmq2=8.00&sound=95&pressure=1006.49
20:22:21.501 ->
20:22:24.570 -> temperature=21.40&humidity=55.00&gasmq135=166&gasmq2=9.00&sound=98&pressure=1006.45
20:22:24.678 ->
Послідовний порт Arduino Mega
Output
Serial Monitor
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM8')
20:22:40.036 -> >>>>>>closing connection
20:22:40.036 ->
20:22:50.163 -> Received data from Arduino Mega: temperature=21.50&humidity=55.00&gasmq135=232&gasmq2=11.00&soundt
20:22:50.347 -> >>>>>> connecting to 192.168.1.134:80
20:22:51.622 -> [Sending a request]
20:22:51.806 -> receiving from the remote server
20:22:51.849 -> HTTP/1.1 200 OK
20:22:51.849 -> Date: Thu, 18 Jan 2024 19:22:51 GMT
20:22:51.886 -> Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4
20:22:51.947 -> X-Powered-By: PHP/8.2.4
20:22:51.983 -> Content-Length: 23
20:22:51.983 -> Connection: close
20:22:52.015 -> Content-Type: text/html; charset=UTF-8
20:22:52.049 ->
20:22:52.049 -> Guardado correctamente!
20:22:52.082 -> Could not save the data
20:22:52.116 ->
20:22:52.116 -> >>>>>>closing connection
20:22:52.116 ->
Послідовний порт вузла MCU ESP8266

База даних з отриманими значеннями



	id	temperature	humidity	gas_mq135	gas_mq2	sound	pressure	timestamp
Edit Copy Delete	1	19.4	52	70	4	60	1022.06	2023-12-28 17:38:35
Edit Copy Delete	2	19	58	222	7	50	1021.99	2023-12-30 01:18:59
Edit Copy Delete	7	14.86	69	385	8	79	1000.09	2023-12-30 12:11:31
Edit Copy Delete	8	14.86	69	385	8	79	1000.09	2024-01-01 19:34:55
Edit Copy Delete	9	18.86	49	233	6	79	1120.8	2024-01-02 17:36:42
Edit Copy Delete	10	20.2	57	136	4	52	1015.35	2024-01-02 19:38:54
Edit Copy Delete	11	20.2	57	136	4	52	1015.35	2024-01-02 20:23:13
Edit Copy Delete	13	20.18	47.1	333	3	66	2000.35	2024-01-02 20:32:58
Edit Copy Delete	14	20.18	47.1	333	3	66	2000.35	2024-01-02 22:26:41
Edit Copy Delete	267	19	45	133	4	69	52	2024-01-07 23:58:54
Edit Copy Delete	400	18.6	45	87	4	53	1018.81	2024-01-08 23:47:40
Edit Copy Delete	407	19	45	83	4	52	1018.74	2024-01-08 23:49:12
Edit Copy Delete	425	18.93	66	313	7	63	1032.15	2024-01-11 14:14:36
Edit Copy Delete	426	18.95	66	383	7	60	1032.85	2024-01-11 14:14:45
Edit Copy Delete	427	18.94	65	390	7	50	1029.73	2024-01-11 14:14:54
Edit Copy Delete	428	19.13	66	366	7	69	1028.48	2024-01-11 14:15:01
Edit Copy Delete	429	19.25	63	380	7	70	1027.26	2024-01-11 14:15:08
Edit Copy Delete	430	19.39	59	342	7	64	1026.19	2024-01-11 14:15:22
Edit Copy Delete	431	19.48	60	384	6	62	1026.34	2024-01-11 14:15:29

Висновки

У ході реалізації проєкту було створено дієздатну IoT-систему для моніторингу якості повітря. Вона поєднує простоту апаратної реалізації з ефективною передачею та збереженням екологічних даних. Обрана архітектура виявилася надійною та зручною для користувача, а веб-інтерфейс забезпечив доступ до поточних і історичних показників. Система успішно працює в реальному часі та адаптується до різних умов використання. Вона може бути корисною для міст, промисловості, сільського господарства та екологічного нагляду. Проєкт продемонстрував потенціал технологій Інтернету речей у сфері довкілля. Подальший розвиток системи може включати інтеграцію точніших сенсорів, мобільного застосунку та автоматичних сповіщень.