МІНІСТЕРСТВО ОСВІТИ НАУКИ УКРАЇНИ

ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО
«ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра комп'ютерної інженерії

ДО ЗАХИСТУ ДОПУЩЕНА

Завідувач кафедри,
д.е.н., проф.

_____ Левицький С.І.

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА
ВИКОРИСТАННЯ БАЗ ДАНИХ ЧАСОВИХ РЯДІВ У СИСТЕМАХ ВИМІРЮВАННЯ ПІД МІКРОКОНТРОЛЕРНИМ УПРАВЛІННЯМ

Виконав

ст. гр. КІ-119 _____ М.В. Краснокутський

Керівник

доц. _____ О.А. Жеребцов

Запоріжжя

2023

ПРАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра _____
(назва кафедри)

ЗАТВЕРДЖУЮ

Зав. кафедри _____
(підпис)

_____
(Науковий ступінь, вчене звання, прізвище та ініціали)

___.___._____ р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ (МАГІСТЕРСЬКУ) РОБОТУ

Студенту гр._

, спеціальності_____

Прізвище Ім'я По батькові

_____

1.Тема:

затверджена наказом по інституту № _____ від ____.___._____ р.

2. Термін здачі студентом закінченої роботи: ____.____. _____ р.

3. Перелік питань, що підлягають розробці

1. Провести огляд літератури, що присвячена тематиці досліджень

……..

8. Оформити звіт за результатами роботи

4. Календарний графік підготовки кваліфікаційної роботи

№ етапу	Зміст	Терміни виконання	Готовність по графіку %, підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Збір практичного матеріалу за темою кваліфікаційної бакалаврської (магістерської) роботи			
2	І атестація І розділ кваліфікаційної бакалаврської (магістерської) роботи			
3	ІІ атестація ІІ розділ кваліфікаційної бакалаврської (магістерської) роботи			
4	ІІІ атестація ІІІ розділ кваліфікаційної бакалаврської (магістерської) роботи, висновки та рекомендації, додатки, реферат)			
5	Перевірка кваліфікаційної бакалаврської (магістерської) роботи на оригінальність			
6	Доопрацювання кваліфікаційної бакалаврської (магістерської) роботи, підготовка презентації, отримання відгуку керівника і рецензії			
7	Попередній захист кваліфікаційної бакалаврської (магістерської) роботи			
8	Подача кваліфікаційної бакалаврської (магістерської) роботи на кафедру			
9	Захист кваліфікаційної бакалаврської (магістерської) роботи			

Дата видачі завдання: ___.___._____ р.

Керівник кваліфікаційної
бакалаврської
(магістерської) роботи _____ _____
(підпис) (ініціали та прізвище)

Завдання отримав до виконання _____ _____
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Бакалаврська дипломна робота містить: 64 сторінки, 23 рисунків, 3 таблиць, одного додатка, 19 використаних джерел.

Мета роботи – створення системи з мікроконтролера з датчиками та базами даних для збору даних часових рядів та порівняти реляційну та нереляційну базу даних для взаємодії з даними часових рядів.

Предмет дослідження – дослідження перспектив ефективності використання реляційних та нереляційних баз даних.

Об’єкт дослідження – данні часових рядів.

У першому розділі проведений розгляд предметної області, використання баз даних часових рядів у вимірювальних системах. Розглянуто аналоги та існуючий стан баз даних в системах загального та спеціального призначення, також була розроблена принципова модель системи проекту.

У другому розділі описано та обґрунтовано вибір програмних та апаратних інструментів розробки для реалізації створення умов необхідних для тестування ефективності взаємодії різних типів баз даних з даними часових рядів.

У третьому розділі описаний процес розробки і випробувань апаратно та програмної частини проекту взаємодії двух типів баз даних, а також, було проведено порівняльне тестування перспектив продуктивності двох СУБД з даними часових рядів, яке відобразило суттєву різницю в ефективності використання типу баз даних для даних часових рядів в загальній перспективі.

СУБД, ДАНІ ЧАСОВИХ РЯДІВ, СИСТЕМИ ВИМІРЮВАННЯ, МІКРОКОНТРОЛЕРИ, ОБМІН ДАНИМИ, MySQL, InfluxDB

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	Повна назва	Пояснення/переклад
БД	База даних	
БДЧС	Бази даних часових рядів	
БДЗП	Бази даних загального призначення	
СУБД	Система управління базами даних	
API	Application Programming Interface	Прикладний програмний інтерфейс
PC	Personal computer	Персональный компьютер
NTP	Network Time Protocol	Протокол мережного часу
I2C	Inter-Integrated Circuit	Послідовна асиметрична шина
JSON	JavaScript Object Notation	Текстовий формат обміну даними
XML	Extensible Markup Language	Стандарт побудови мов розмітки ієрархічно структурованих даних
NoSQL	Not only SQL	Позначення широкого класу різнорідних систем управління базами даних які відрізняються від традиційних реляційних СУБД з доступом до даних засобами мови SQL
IT	Information Technology	Інформаційні технології
SQL	Structured query language	Мова структурованих запитів
MQTT	Message queuing telemetry transport	Спрощений мережевий протокол
QoS	Quality of service	Якість послуг, які надає комунікаційна мережа
IDE	Integrated Development Environment	Інтегроване середовище розробки
IoT	Internet of things	Концепція мережі передачі даних між фізичними об'єктами

ВСТУП

Розвиток наукової сфери проектування та будування комп'ютерних систем, знайшов свій шлях у потребі розробки методів взаємодії різноманітних систем різного рівня інженерної складності. Так з'явились електронно-обчислювальні прилади, що побудовані на одному кристалі та, згідно завдання, які займають мало місця й потребують менше ресурсів та складності під час виготовлення- мікроконтролери. Вони зайняли своє почесне місце в сучасних інженерних розробках та в популярній на сьогоднішній день течії- вбудованих системах (англ. Embedded System).

Вбудована система- спеціалізована мікропроцесорна система управління, контролю та моніторингу, концепція розробки якої полягає в тому, що така система буде працювати, будучи вбудованою безпосередньо у пристрій, яким вона керує.

Вбудовані системи використовуються в якості як відокремлених електронних вузлів або систем, так і в умовах ланцюгового з'єднання, з використанням різних видів та методів зв'язку.

Основними напрямками використання структур вбудованого типу є: засоби автоматичного регулювання та управління технологічними процесами, наприклад авіоніка, контроль доступу, верстати з ЧПУ (Числовим Програмним Керуванням), банкомати, платіжні термінали, телекомунікаційне обладнання та величезна кількість інших реалізацій будь-якого типу та належності.

Завдяки великій кількості переважаючих властивостей мікроконтролерної невеликогабаритної техніки, вона зайняла суттєву нішу в програмі розробок електронних інженерів в усьому світі.

РОЗДІЛ 1

ОГЛЯД БАЗ ДАНИХ ЧАСОВИХ РЯДІВ У ВИМІРЮВАЛЬНИХ СИСТЕМАХ

1.1 Розгляд предметної області

Використання баз даних часових рядів у вимірювальних системах, керованих мікроконтролерами, є предметною областю, яка передбачає використання спеціалізованих баз даних для зберігання та аналізу даних, зібраних мікроконтролерами в режимі реального часу. Ця предметна область має велике значення в різних галузях промисловості, включаючи виробництво, аерокосмічну промисловість і охорону здоров'я, де точне вимірювання даних має вирішальне значення. [1]

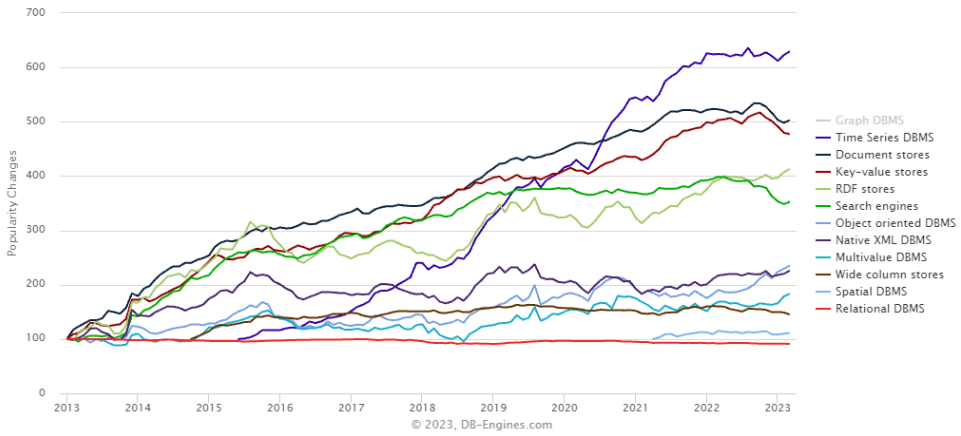

Рисунок 1.1 – Зростання категорії баз даних останні 10 років

Бази даних часових рядів — це бази даних, спеціально розроблені для зберігання й аналізу точок даних із мітками часу. У системах вимірювання, керованих мікроконтролером, ці бази даних використовуються для зберігання даних, зібраних з датчиків та інших вимірювальних пристроїв, підключених

до мікроконтролера. Дані, що зберігаються в цих базах даних, можна аналізувати та візуалізувати, щоб отримати уявлення про різні аспекти процесу вимірювання.

Вимірювальні системи, керовані мікроконтролерами — це системи, які використовують мікроконтролери для керування збором і обробкою даних від датчиків та інших вимірювальних пристроїв. Ці системи широко використовуються в різних галузях промисловості, включаючи автомобільну, аерокосмічну та охорону здоров'я, де точне й точне вимірювання даних має вирішальне значення. Зазвичай складаються з мікроконтролера, датчиків і комп'ютера або іншого пристрою обробки даних. Використання баз даних часових рядів у вимірювальних системах, керованих мікроконтролерами, може допомогти підвищити точність і надійність зібраних даних. Ці бази даних можуть зберігати великі обсяги даних і дозволяють ефективно шукати та аналізувати дані. Вони також дозволяють аналізувати дані в реальному часі, що може бути корисним для виявлення тенденцій і аномалій.

Однією з головних переваг використання баз даних часових рядів у вимірювальних системах, керованих мікроконтролерами, є те, що вони дозволяють виконувати аналіз даних у реальному часі. Це дозволяє негайно отримувати зворотний зв'язок щодо зібраних даних, дозволяючи операторам швидко приймати обґрунтовані рішення. Використання баз даних часових рядів також дозволяє зберігати великі обсяги даних, які можуть бути корисними для визначення закономірностей і тенденцій у часі.

Ще одна перевага використання баз даних часових рядів у вимірювальних системах, керованих мікроконтролерами, полягає в тому, що вони можуть допомогти зменшити обсяг пам'яті, необхідний для даних. Це пояснюється тим, що бази даних часових рядів зазвичай зберігають дані в стиснутому форматі, що може зменшити обсяг необхідного місця для зберігання. Це може бути особливо корисним у програмах, де збираються

великі обсяги даних, наприклад у виробництві чи аерокосмічній промисловості.

Підсумовуючи, використання баз даних часових рядів у вимірювальних системах, керованих мікроконтролерами, є критичною предметною областю в різних галузях промисловості, включаючи виробництво, аерокосмічну промисловість та охорону здоров'я. Ці бази даних дозволяють ефективно зберігати й аналізувати точки даних із мітками часу, надаючи розуміння різних аспектів процесу вимірювання. Використання баз даних часових рядів може допомогти підвищити точність і надійність зібраних даних, а також уможливити аналіз даних у реальному часі.

1.2 Розгляд аналогів та існуючого стану

Аналог використання баз даних часових рядів у вимірювальних системах, керованих мікроконтролерами, є використання баз даних в обчислювальних системах загального призначення. [2]

Бази даних часових рядів (БДЧС) — це спеціалізовані бази даних, розроблені спеціально для обробки великих обсягів даних часових рядів, створених системами, які збирають дані з часом. Вони оптимізовані для ефективного прийому та запиту даних на основі часових позначок та інших часових атрибутів. Бази даних загального призначення, з іншого боку, призначені для обробки широкого спектру типів даних і випадків використання. Хоча їх можна використовувати для зберігання даних часових рядів, вони не оптимізовані для цього конкретного випадку використання та можуть не забезпечувати такий самий рівень продуктивності чи функціональності, як БДЧС.

Однією з головних відмінностей між БДЧС і базами даних загального призначення є те, як вони зберігають та індексують дані. БДЧС

використовують індекс на основі часу, як правило, B-дерево або хеш-таблицю, для ефективного запиту даних на основі часових позначок. Навпаки, бази даних загального призначення зазвичай використовують реляційну або документну модель, яка зберігає дані в таблицях або документах і використовує індекси для запиту даних. Хоча ці індекси можна оптимізувати для запиту даних на основі часу, вони не такі ефективні, як спеціалізований індекс на основі часу, який використовується в БДЧС.

Рисунок 1.2 – Різниця реляційної БДЗП та БДЧС

Ще одна відмінність між БДЧС і базами даних загального призначення полягає в типі запитів, які вони підтримують. БДЧС розроблено для підтримки запитів, які аналізують і агрегують дані часових рядів за певні часові інтервали, наприклад, обчислення ковзних середніх або визначення тенденцій у даних. Бази даних загального призначення, з іншого боку, розроблені для підтримки широкого спектру запитів до різних типів даних, але можуть не забезпечувати той самий рівень функціональності чи продуктивності для запитів на основі часу.

Бази даних загального призначення можна використовувати для зберігання даних часових рядів, але вони не так добре підходять для цього

завдання, як спеціалізовані БДЧС. БДЧС забезпечують оптимізовану продуктивність і функціональність для обробки даних часових рядів і можуть бути кращим вибором для програм, які потребують моніторингу в реальному часі та аналізу великих обсягів даних часових рядів.

Існуючий стан використання баз даних часових рядів у вимірювальних системах, керованих мікроконтролерами, набуває популярності завдяки їх здатності ефективно зберігати та аналізувати великі обсяги даних часових рядів. Багато сучасних вимірювальних систем, особливо тих, що використовуються в промисловому секторі, покладаються на мікроконтролери для збору та обробки даних у режимі реального часу. Бази даних часових рядів все частіше використовуються для зберігання цих даних, що дозволяє ефективніше надсилати запити та аналізувати.

Деякі популярні бази даних часових рядів, що використовуються в системах вимірювання, керованих мікроконтролерами, включають InfluxDB, OpenTSDB і TimescaleDB. Ці бази даних пропонують такі функції, як стиснення та агрегація для оптимізації зберігання та запиту даних часових рядів. Крім того, вони пропонують API та бібліотеки, які полегшують інтеграцію з платформами мікроконтролерів, такими як Arduino та Raspberry Pi.

Також тривають дослідження та розробки в галузі баз даних часових рядів і вимірювальних систем, керованих мікроконтролерами. Наприклад, деякі дослідники вивчають використання алгоритмів машинного навчання для аналізу даних часових рядів у режимі реального часу, тоді як інші працюють над розробкою більш ефективних методів зберігання та індексування даних часових рядів.

Загалом, поточний стан використання баз даних часових рядів у вимірювальних системах, керованих мікроконтролерами, є ростом та інноваціями. Оскільки все більше галузей використовують системи

вимірювання на основі мікроконтролерів і генерують великі обсяги даних часових рядів, потреба в ефективному зберіганні та аналізі цих даних буде тільки зростати, стимулюючи подальший розвиток у сфері баз даних часових рядів.

1.3 Розробка принципової моделі системи

Розробка концептуальної моделі системи передбачає визначення ключових характеристик і функцій, необхідних для досягнення бажаного результату.

Ключові особливості системи включають датчики Arduino ESP8266, BME280 і DHT11, компьютер з базою даних InfluxDB та MySQL. Arduino ESP8266 буде використовуватися як мікроконтролер для збору даних з датчиків і запису в бази даних. Датчики BME280 і DHT11 використовуватимуться для вимірювання даних про температуру, вологість і тиск, а функція синхронізації часу забезпечить точну позначку часу в який дані були зібрані.

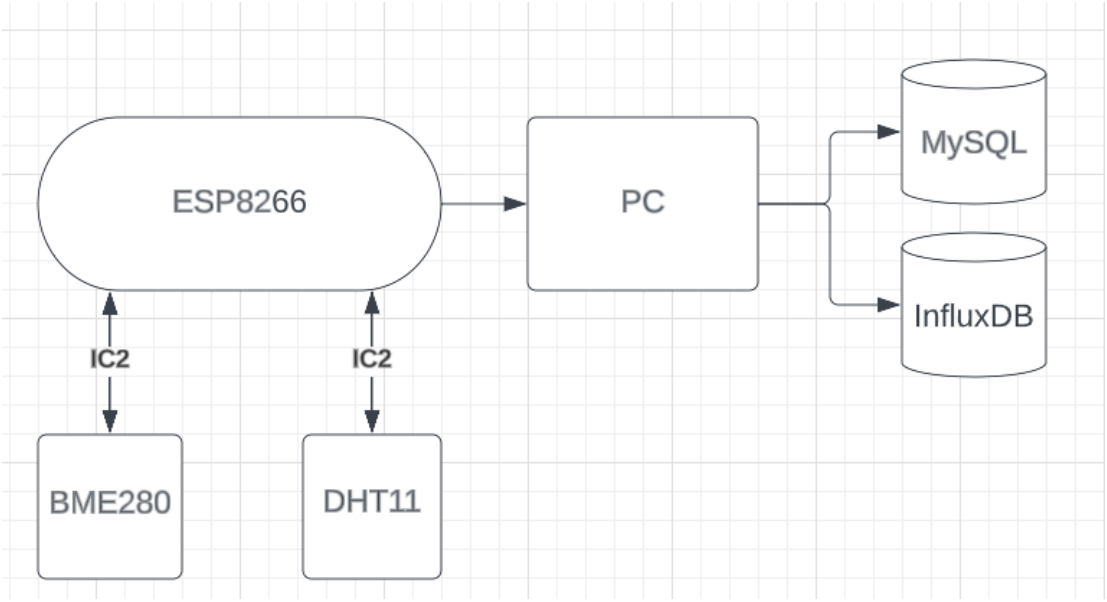

Рисунок 1.3 – Принципова схема системи

Функціональність системи передбачає розробку програмного та апаратного забезпечення, необхідного для ефективної роботи системи. Arduino ESP8266 буде запрограмований на збір даних із датчиків за протоколом I2C, синхронізації поточного часу у Києві за допомогою NTP протоколу, коли вони були зібрані. Пакет буде передано компьютер за допомогою Arduino ESP8266. Компьютер матиме базу даних InfluxDB та MySQL для зберігання зібраних даних разом із часом, коли вони були зібрані.

Архітектура системи передбачає фізичне складання та програмування системи. Датчики будуть підключатись до Arduino ESP8266 за допомогою I2C протоколу. Для програмування Arduino ESP8266 буде використовуватися Arduino IDE з необхідним програмним кодом для збору даних з датчиків. Базу даних InfluxDB та MySQL буде встановлено та налаштовано на компьютері для приймання та зберігання зібраних даних.

Валідація системи передбачає перевірку функціональності та продуктивності системи. Система тестуватиметься шляхом збору даних з датчиків та передачі даних на компьютер. Зібрані дані перевірятимуть, щоб переконатися, що вони точні та надійні. Продуктивність системи також перевірятиметься шляхом вимірювання часу, необхідного для збору та передачі даних.

РОЗДІЛ 2 ВИБІР ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ ПРОЕКТУ

2.1 Огляд та класифікація баз даних

База даних — це сукупність даних, організованих певним чином, що забезпечує ефективне зберігання та пошук інформації. Це важливий компонент сучасних комп’ютерних систем, оскільки він використовується для зберігання та керування різноманітною інформацією, включаючи фінансові дані, інформацію про клієнтів та інвентарні записи. Бази даних відіграють життєво важливу роль у функціонуванні бізнесу, державних організацій та інших установ. Вони використовуються для структурованого та організованого зберігання та керування даними, що полегшує доступ до інформації, її пошук і аналіз. [3]

Існує кілька різних типів баз даних, кожна з яких має власний набір функцій і можливостей. Найпоширенішим типом бази даних є реляційна база даних, яка використовує таблиці для організації даних і зв’язків між даними для забезпечення узгодженості даних. Реляційні бази даних широко використовуються в бізнесі та корпоративних програмах. Деякі популярні приклади реляційних баз даних включають MySQL, PostgreSQL і Oracle.

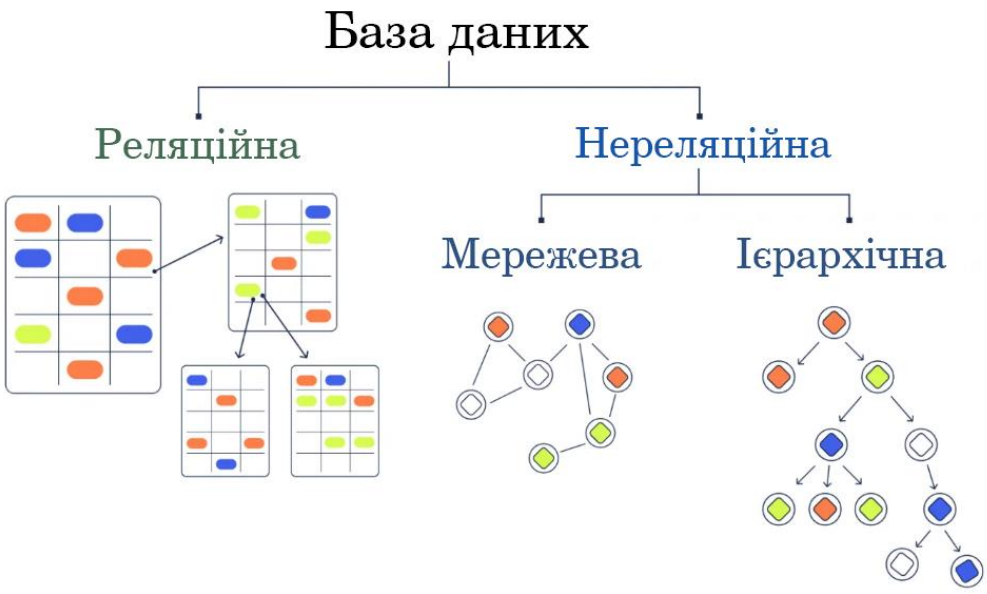

Рисунок 2.1 – Основні типи баз данних

Іншим типом бази даних є база даних документів, яка зберігає дані у формі документів, наприклад JSON або XML. Ці бази даних оптимізовано для зберігання та отримання напівструктурованих даних, що робить їх добре придатними для таких випадків використання, як системи керування вмістом і платформи електронної комерції. Деякі популярні приклади баз даних документів включають MongoDB і Couchbase.

Третій тип баз даних — бази даних «ключ-значення», які зберігають дані у формі пар «ключ-значення». Вони призначені для виконання вимог до високої продуктивності та зазвичай використовуються для кешування, керування сеансами та інших програм, чутливих до продуктивності. Прикладами баз даних ключ-значення є Redis і Riak.

Окрім цих традиційних баз даних, зростає також кількість баз даних NoSQL, які не дотримуються традиційної реляційної моделі та оптимізовані для конкретних випадків використання, таких як великі дані та обробка в реальному часі. Ці бази даних включають Apache Cassandra, Hbase і Neo4j.

Підсумовуючи, бази даних відіграють життєво важливу роль у функціонуванні сучасних комп’ютерних систем, зберігаючи та керуючи даними в структурований та організований спосіб. Різні типи баз даних підходять для різних випадків використання, і вибір правильної бази даних для конкретної програми є важливим для забезпечення продуктивності та масштабованості системи. Зі збільшенням обсягу даних, які генеруються, важливість баз даних у майбутньому лише зростатиме, що зробить їх важливою сферою вивчення для всіх, хто працює в галузі технологій та IT.

Щоб краще зрозуміти та використовувати бази даних, важливо класифікувати їх на основі їхніх характеристик та можливостей.

Одним із поширених способів класифікації баз даних є тип моделі даних, яку вони використовують. Існує три основних типи моделей даних: реляційна, документна та ключ-значення.

Реляційні бази даних використовують таблиці для зберігання даних і зв’язки між даними для забезпечення узгодженості даних. У реляційній базі даних дані зберігаються в таблицях, які є наборами рядків і стовпців. Кожен рядок представляє окремий запис, а кожен стовпець представляє певну частину інформації про цей запис. Зв’язки між таблицями визначаються за допомогою ключів, які пов’язують дані в кількох таблицях. Це дозволяє створювати складні структури даних, наприклад зв’язки «один до багатьох» і «багато до багатьох». Приклади реляційних баз даних включають MySQL, PostgreSQL і Oracle.

Бази даних документів зберігають дані у формі документів, наприклад JSON або XML. Ці бази даних оптимізовано для зберігання та отримання напівструктурованих даних, що робить їх добре придатними для таких випадків використання, як системи керування вмістом і платформи електронної комерції. Бази даних документів не мають фіксованої схеми, а це означає, що структура даних може змінюватися з часом. Це робить їх більш гнучкими, ніж реляційні бази даних, але це також означає, що вони менш підходять для програм, які вимагають високого ступеня узгодженості даних. Приклади баз даних документів включають MongoDB і Couchbase.

Бази даних ключ-значення зберігають дані у формі пар ключ-значення. Вони призначені для виконання вимог до високої продуктивності та зазвичай використовуються для кешування, керування сеансами та інших програм, чутливих до продуктивності. Бази даних «ключ-значення» прості та швидкі, але їм бракує багатьох розширених функцій реляційних баз даних і баз даних документів. Прикладами баз даних ключ-значення є Redis і Riak.

Інший спосіб класифікації баз даних — за типом даних, які вони зберігають. Існує три основні категорії зберігання даних: структуроване, напівструктуроване та неструктуроване. [4]

Структуровані дані — це дані, організовані чітко визначеним чином, наприклад дані, що зберігаються в реляційних базах даних. Структуровані дані можна легко шукати та аналізувати, що робить їх добре придатними для таких випадків використання, як фінансові дані, інформація про клієнтів та інвентарні записи.

Напівструктуровані дані – це дані, які не так чітко визначені, як структуровані дані, але все ж мають певну структуру. Напівструктуровані дані зберігаються в базах даних документів і включають такі дані, як системи керування вмістом і платформи електронної комерції.

Неструктуровані дані – це дані, які не організовані чітко визначеним чином, і їх важко знайти чи проаналізувати. Неструктуровані дані включають такі дані, як зображення, аудіо та відео.

Підсумовуючи, класифікація баз даних на основі їхніх характеристик і можливостей є важливою для розуміння та ефективного їх використання. Різні типи баз даних підходять для різних випадків використання, і вибір правильної бази даних для конкретної програми є важливим для забезпечення продуктивності та масштабованості системи. Незалежно від типу моделі даних або типу даних, які вони зберігають, класифікація баз даних дає змогу краще зрозуміти їхні можливості та обмеження, що робить її важливою сферою вивчення для тих, хто працює у сфері технологій та IT.

2.2 Характеристика баз даних часових рядів

Бази даних часових рядів — це спеціалізовані бази даних, призначені для зберігання та керування даними з мітками часу. Вони спеціально оптимізовані для обробки даних часових рядів, тобто серії точок даних, зібраних через регулярні проміжки часу. [5]

Їх загальні переваги це:

Дані з мітками часу: бази даних часових рядів призначені для зберігання та керування даними з мітками часу, тобто даними, пов’язаними з певним моментом часу. Це робить бази даних часових рядів ідеальними для зберігання таких даних, як дані датчиків, фінансові дані та системні журнали, які зазвичай збираються через регулярні проміжки часу та вимагають швидкої й ефективної обробки в залежності від часу.

Отримання на основі часу: бази даних часових рядів оптимізовано для пошуку на основі часу, що означає, що вони призначені для отримання даних на основі певного діапазону часу або часового вікна. Це робить бази даних часових рядів ідеальними для додатків, які вимагають швидкої та ефективної обробки даних на основі часу, таких як аналітика в реальному часі та системи моніторингу.

Висока продуктивність запису та запитів: бази даних часових рядів розроблено для роботи з високою продуктивністю запису та запитів, що означає, що вони оптимізовані для обробки великих обсягів даних у режимі реального часу. Це робить бази даних часових рядів ідеальними для додатків, які вимагають швидкої та ефективної обробки даних, таких як системи моніторингу та оповіщення в реальному часі.

Структури даних часових рядів: бази даних часових рядів використовують структури даних часових рядів, такі як таблиці часових рядів, для зберігання даних. Ці структури даних оптимізовані для обробки даних у часі та забезпечують швидкий і ефективний пошук і обробку даних.

Стиснення та агрегація даних: бази даних часових рядів зазвичай підтримують стиснення та агрегацію даних, що дозволяє ефективно зберігати великі обсяги даних. Стиснення даних зменшує розмір даних, що робить їх більш ефективним для зберігання, а агрегація даних дозволяє узагальнювати дані, що робить їх більш ефективними для обробки.

Масштабованість: Бази даних часових рядів розроблені таким чином, щоб бути масштабованими, що означає, що вони можуть обробляти великі обсяги даних і збільшувати кількість користувачів і пристроїв, не впливаючи на продуктивність. Це робить бази даних часових рядів ідеальними для додатків, які потребують швидкої й ефективної обробки даних і підтримки великої кількості користувачів і пристроїв.

Довговічність: Бази даних часових рядів розроблені для довговічності, що означає, що вони гарантують, що дані не будуть втрачені у разі збою. Це робить бази даних часових рядів ідеальними для програм, які вимагають надійного та узгодженого зберігання даних, наприклад фінансових даних і системних журналів.

Бази даних часових рядів мають унікальні характеристики, завдяки яким вони добре підходять для даних часових рядів і додатків, які потребують

швидкої та ефективної обробки даних на основі часу. Хоча вони пропонують багато переваг, але вони також мають деякі обмеження, які важливо враховувати, вирішуючи, чи використовувати базу даних часових рядів:

Складність: Бази даних часових рядів можуть бути складними для налаштування та використання, особливо для тих, хто не знайомий із розробкою та керуванням базами даних. Вони часто потребують спеціальних знань і навичок для ефективного керування, що робить їх менш доступними для нетехнічних користувачів.

Обмежені можливості надсилання запитів: бази даних часових рядів оптимізовано для пошуку та обробки даних за часом, але можуть мати обмежені можливості надсилання запитів порівняно з традиційними реляційними базами даних. Це може бути недоліком для додатків, які вимагають складних запитів або аналізу, що включає дані з кількох джерел.

Обмеження розміру даних: Бази даних часових рядів оптимізовано для високої продуктивності запису та запитів, але вони можуть мати обмеження, коли йдеться про зберігання великих обсягів даних. Як результат, вони можуть бути не найкращим вибором для програм, які вимагають зберігання надзвичайно великих наборів даних.

Узгодженість даних: бази даних часових рядів можуть бути вразливими до проблем узгодженості даних, особливо коли дані в базу даних записуються з кількох джерел одночасно. Це може призвести до проблем цілісності даних, які може бути важко вирішити без відповідних методів керування даними.

Вартість: Бази даних часових рядів можуть бути дорожчими, ніж традиційні реляційні бази даних, особливо якщо врахувати вартість обладнання, програмного забезпечення та підтримки. Це може зробити їх менш доступними для невеликих організацій або організацій з обмеженим бюджетом.

Інтеграція з іншими системами: Бази даних часових рядів може бути складно інтегрувати з іншими системами, такими як інструменти аналізу даних, що ускладнює вилучення та використання даних в інших програмах. Це може обмежити цінність даних часових рядів і зменшити потенційні переваги використання бази даних часових рядів.

Підсумовуючи, бази даних часових рядів мають унікальні характеристики, завдяки яким вони добре підходять для даних часових рядів і додатків, які вимагають швидкої та ефективної обробки даних на основі часу. Бази даних часових рядів забезпечують функціональність і продуктивність, необхідні для ефективного керування даними часових рядів, незалежно від того, чи це їх структури даних із мітками часу, висока продуктивність запису та запитів, чи масштабованість і довговічність. важливо враховувати їх обмеження, вирішуючи, чи використовувати базу даних часових рядів. Незалежно від того, чи це складність, обмежені можливості запитів, обмеження розміру даних, узгодженість даних, вартість або інтеграція з іншими системами, розуміння недоліків баз даних часових рядів має важливе значення для ефективного управління даними та прийняття рішень.

2.3 Огляд та вибір – СУБД

Система керування базами даних (СУБД) є важливим компонентом сучасних інформаційних систем. Він забезпечує зручний інтерфейс для керування та організації великих обсягів даних у базі даних. Зокрема, СУБД добре підходять для роботи з даними часових рядів, які є типом даних, які збираються через регулярні проміжки часу. Щоб ефективно керувати даними часових рядів, СУБД повинна вміти справлятися зі специфічними проблемами, пов’язаними з цим типом даних. Однією з цих проблем є велика кількість даних, які генеруються. Дані часових рядів збираються через регулярні проміжки часу, часто кілька разів на секунду, що може

призвести до створення дуже великих обсягів даних з часом. СУБД повинна мати можливість ефективно зберігати ці дані та керувати ними. [6]

Іншою проблемою є необхідність обробки даних часових рядів у режимі реального часу. У багатьох додатках важливо мати можливість обробляти та аналізувати дані часових рядів під час їх створення, щоб якнайшвидше визначати тенденції та закономірності. СУБД повинна мати можливість обробляти цю обробку даних у режимі реального часу, не спричиняючи проблем з продуктивністю.

Нарешті, СУБД повинна мати можливість обробляти унікальні характеристики даних часових рядів. Наприклад, дані часових рядів часто є нестаціонарними, тобто їх статистичні властивості можуть змінюватися з часом. СУБД повинна вміти справлятися з цією нестаціонарністю та надавати інструменти, необхідні для аналізу та візуалізації даних.

СУБД є важливим інструментом для керування даними часових рядів. Він надає зручний інтерфейс для керування та організації великих обсягів даних і добре підходить для вирішення конкретних проблем, пов’язаних із даними часових рядів, таких як велика кількість згенерованих даних, потреба в обробці в реальному часі, і унікальні характеристики даних.

2.3.1 Часові ряди в СУБД загального призначення

MySQL — це популярна система керування реляційними базами даних (СУБД) з відкритим кодом, яка широко використовується для керування даними часових рядів. Добре підходить для обробки даних часових рядів завдяки своїм потужним можливостям зберігання та пошуку даних. Це дозволяє користувачам легко зберігати великі обсяги даних часових рядів у централізованому місці, які можна ефективно отримати за допомогою SQL (мова структурованих запитів). Це спрощує для користувачів виконання комплексного аналізу даних і візуалізації даних часових рядів. [7]

MySQL також надає ряд функцій, які спеціально розроблені для керування даними часових рядів. Наприклад, він підтримує створення індексованих таблиць, які можна використовувати для ефективного отримання даних часових рядів на основі конкретних умов, таких як діапазони дат і часу. Це важливо для обробки даних часових рядів у режимі реального часу, що важливо в багатьох програмах. MySQL також надає інструменти для обробки нестаціонарності даних часових рядів, що є загальною характеристикою цього типу даних. Наприклад, це дозволяє користувачам виконувати комплексний аналіз даних і візуалізацію даних часових рядів, що може допомогти визначити тенденції та закономірності з часом.

Окрім потужних можливостей зберігання та пошуку даних, MySQL також забезпечує високий рівень безпеки та цілісності даних. Він включає такі функції, як автентифікація користувачів і шифрування даних, які гарантують, що дані часових рядів захищені від несанкціонованого доступу. Він також включає функції резервного копіювання та відновлення, які гарантують, що дані часових рядів можна відновити у разі збою.

Підсумовуючи, MySQL є потужною та широко використовуваною СУБД з відкритим кодом, яка добре підходить для керування даними часових рядів. Його можливості зберігання та пошуку даних, а також функції, спеціально розроблені для керування даними часових рядів, роблять його ідеальним рішенням для ефективного, точного та безпечного керування великими обсягами даних часових рядів.

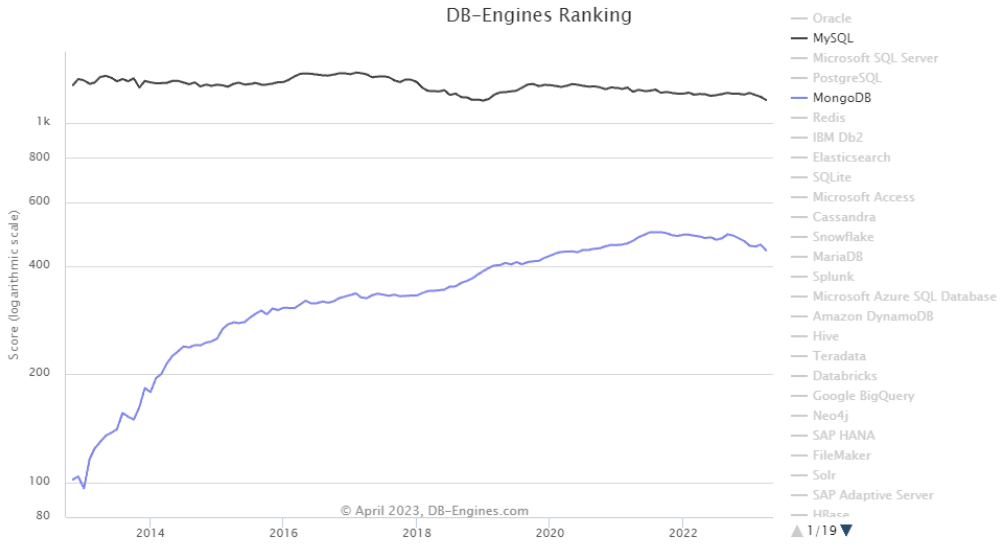

Рисунок 2.2 – Популярність MySQL та MongoDB за 10 років

MongoDB — популярна система керування базами даних NoSQL (СУБД) з відкритим кодом, яка добре підходить для керування даними часових рядів. Однією з ключових переваг MongoDB для даних часових рядів є її гнучкий дизайн схеми. На відміну від традиційних реляційних баз даних, MongoDB дозволяє створювати документи з різними структурами, що дозволяє легко зберігати дані часових рядів без необхідності заздалегідь визначати жорстку схему. Це може бути особливо корисним при роботі з даними часових рядів, які можуть мати різну структуру з часом. [8]

Ще однією перевагою MongoDB є його здатність обробляти великі обсяги даних. MongoDB може легко зберігати та отримувати великі обсяги даних часових рядів, що важливо для багатьох додатків часових рядів. Він також включає вбудований шардинг, який дозволяє розподіляти дані між декількома серверами, покращуючи продуктивність і масштабованість. Також включає потужні можливості індексування та запитів, що полегшує користувачам отримання даних часових рядів на основі конкретних умов, таких як діапазони дат і часу. Це важливо для обробки даних часових рядів у режимі реального часу, що важливо в багатьох програмах.

Крім того, MongoDB забезпечує високий рівень безпеки та цілісності даних. Він включає такі функції, як автентифікація користувачів і шифрування даних, які гарантують, що дані часових рядів захищені від несанкціонованого доступу. Він також включає функції резервного копіювання та відновлення, які гарантують, що дані часових рядів можна відновити у разі збою.

Підсумовуючи, MongoDB — це СУБД, яка добре підходить для керування даними часових рядів. Його гнучкий дизайн схеми, здатність обробляти великі обсяги даних, потужні можливості індексування та запитів, а також високий рівень безпеки та цілісності даних роблять його ідеальним рішенням для ефективного, точного та безпечного керування даними часових рядів.

2.3.2 Часові ряди в СУБД спеціального призначення

InfluxDB — це система керування базами даних часових рядів із відкритим вихідним кодом, розроблена для обробки високошвидкісних і великих обсягів даних часових рядів. Він забезпечує високу продуктивність запитів і гнучку модель даних, що робить його добре придатним для широкого спектру додатків даних часових рядів, включаючи моніторинг, IoT і DevOps. [9]

InfluxDB створено спеціально для даних часових рядів і надає низку функцій, які роблять його ідеальним рішенням для керування даними часових рядів. Він має схожу на SQL мову запитів InfluxQL, яка проста та інтуїтивно зрозуміла у використанні, і підтримує гнучку модель даних, яка дозволяє користувачам зберігати дані часових рядів у різних форматах.

Має високу масштабованість, що робить його добре придатним для великих програм даних часових рядів. Він може обробляти мільйони точок даних за секунду та забезпечує швидку роботу запитів для аналізу даних часових рядів. Крім того, InfluxDB надає низку інструментів для візуалізації та аналізу даних, що полегшує користувачам отримання інформації з даних часових рядів.

InfluxDB широко використовується в різних галузях, включаючи фінанси, охорону здоров’я, телекомунікації та роздрібну торгівлю. Це популярний вибір для додатків моніторингу та керування продуктивністю, а також для додатків IoT і DevOps, де він використовується для зберігання та аналізу даних часових рядів із широкого діапазону пристроїв і систем.

Підсумовуючи, InfluxDB — це потужна та гнучка система керування базою даних часових рядів, яка забезпечує швидку роботу запитів і просту та інтуїтивно зрозумілу мову запитів. Його масштабованість, можливості візуалізації та аналізу даних, а також широкий спектр застосувань роблять його ідеальним рішенням для широкого діапазону потреб управління даними часових рядів.

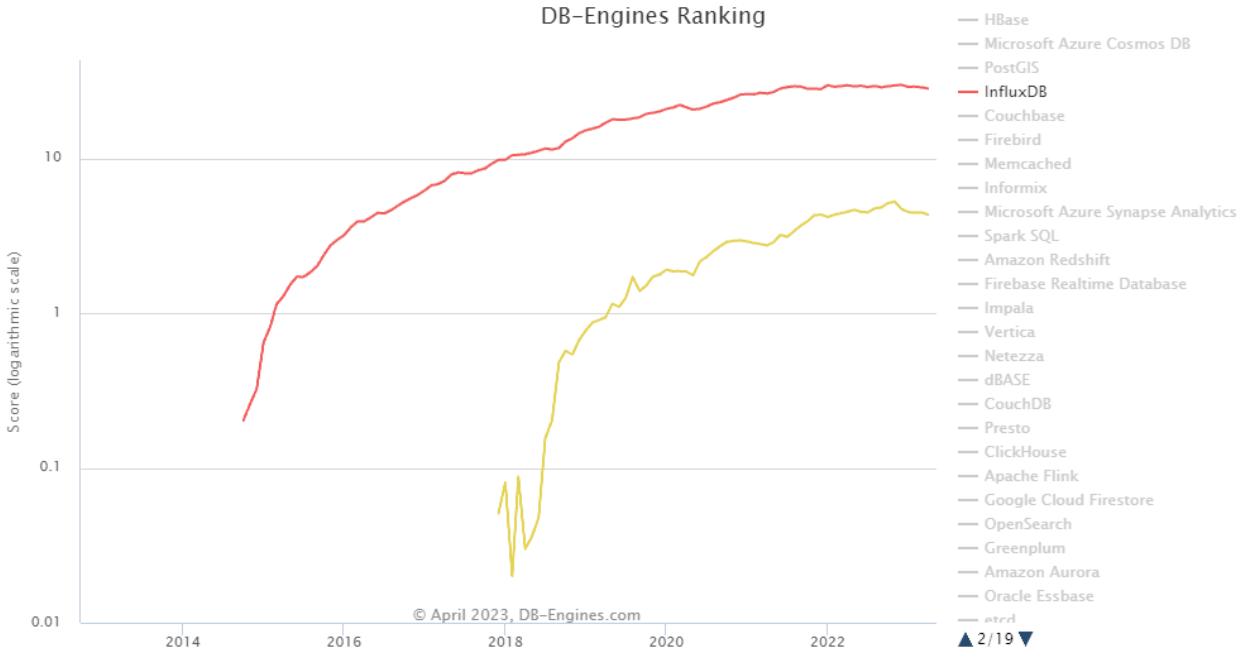

Рисунок 2.3 – Популярність InfluxDB та TimescaleDB за 10 років

TimescaleDB — реляційна база даних із відкритим вихідним кодом, яка спеціально розроблена для даних часових рядів. Він створений на основі PostgreSQL, широко використовуваної та добре налагодженої реляційної бази даних, і забезпечує інтерфейс SQL для запитів і керування даними часових рядів. Однією з ключових переваг TimescaleDB є його здатність легко обробляти дані часових рядів великого обсягу та високої швидкості. Він забезпечує швидку роботу запитів для великомасштабних даних часових рядів і підтримує широкий спектр типів даних, що робить його добре придатним для різноманітних програм даних часових рядів. [10]

TimescaleDB також надає низку інструментів для аналізу та візуалізації даних, що полегшує користувачам отримання інформації з даних часових рядів. Він підтримує низку клієнтських бібліотек та інтеграцію з популярними інструментами аналізу та візуалізації даних, такими як Grafana, що полегшує користувачам інтеграцію з наявними програмами та інструментами.

Ще однією перевагою TimescaleDB є його сумісність із SQL, яка є широко використовуваною та добре зрозумілою мовою баз даних. Це спрощує роботу з TimescaleDB для користувачів із попереднім досвідом роботи з SQL,

а також означає, що користувачі можуть використовувати широкий спектр інструментів і бібліотек на основі SQL для роботи з даними часових рядів.
Підсумовуючи, TimescaleDB — це потужна та гнучка база даних часових рядів, яка забезпечує швидку роботу запитів і знайомий інтерфейс SQL. Його масштабованість, можливості візуалізації та аналізу даних, а також сумісність із SQL роблять його ідеальним рішенням для широкого діапазону потреб керування даними часових рядів.

2.4 Вибір апаратно - програмного комплексу для розгортання проектуемої системи.

При виборі апаратно-програмного комплексу для розгортання проектованої системи необхідно враховувати кілька важливих факторів, таких як:

Продуктивність: для забезпечення швидкої та ефективної обробки даних необхідно вибирати апаратне забезпечення з високою продуктивністю. При цьому слід враховувати не тільки продуктивність процесора, а й обсяг оперативної пам'яті, швидкість роботи жорсткого диска та інші фактори, які можуть вплинути на загальну продуктивність системи.

Надійність: важливим фактором при виборі апаратно-програмного комплексу є надійність та стійкість до збоїв. Необхідно вибирати обладнання та програмне забезпечення, які забезпечують високий рівень доступності та захисту від збоїв.

Сумісність: при виборі апаратного та програмного забезпечення слід переконатися, що вони сумісні один з одним. Необхідно вибирати компоненти, які легко інтегруються один з одним та дозволяють створювати єдиний, сумісний комплекс.

Масштабованість: при проектуванні системи необхідно враховувати її потенційну масштабованість. Необхідно вибирати компоненти, які забезпечують можливість розширення системи у майбутньому, без значних витрат на оновлення обладнання та програмного забезпечення.

Безпека: під час роботи з конфіденційними даними необхідно забезпечити високий рівень безпеки системи. При виборі апаратно-програмного комплексу слід переконатися, що він забезпечує надійний захист від несанкціонованого доступу та передбачає заходи щодо забезпечення конфіденційності даних.

У результаті, вибираючи апаратно-програмний комплекс для розгортання проектованої системи, необхідно враховувати перераховані вище чинники. Тільки таким чином можна забезпечити високу продуктивність, надійність, сумісність, масштабованість та безпеку системи.

2.4.1 Вибір апаратної складової серверу

При виборі апаратно-складової серверу для бази даних InfluxDB слід враховувати кілька факторів, які можуть вплинути на продуктивність і ефективність роботи бази даних. [11]

Першим важливим чинником є обсяг даних, який зберігатиметься у базі даних InfluxDB. При великих обсягах даних необхідно вибирати сервер із достатньою пам'яттю для зберігання даних у оперативній пам'яті. Крім того, бажано використовувати жорсткий диск із високою швидкістю читання та запису даних, а також з великим обсягом зберігання.

Другим чинником є продуктивність процесора. InfluxDB є базою даних, яка працює в режимі реального часу та обробляє велику кількість даних. Тому

процесор має бути досить потужним, щоб забезпечити високу швидкість обробки даних та виконання запитів.

Третім фактором є мережний інтерфейс. InfluxDB є базою даних, яка часто використовується для збору даних із різних джерел. Тому важливо вибирати сервер із високошвидкісним мережевим інтерфейсом, який забезпечить швидке збирання даних та передачу їх у базу даних.

Четвертим чинником є вибір операційної системи. InfluxDB підтримує роботу на операційних системах Linux, MacOS та Windows. Однак для оптимальної продуктивності та безпеки бази даних рекомендується використовувати Linux.

П'ятий фактор, який також необхідно враховувати, – це ступінь надійності та доступності сервера. При роботі з InfluxDB важливо забезпечити надійність роботи сервера та доступність бази даних для користувачів. Для цього необхідно використовувати компоненти високої якості сервера, такі як блоки живлення, системи охолодження та інші елементи, які забезпечать безперебійну роботу сервера.

Отже, при виборі апаратно-складових серверів для бази даних InfluxDB необхідно враховувати обсяг даних, продуктивність процесора, мережевий інтерфейс, вибір операційної системи, а також надійність і доступність сервера. Крім того, важливо вибирати компоненти сервера високої якості, щоб забезпечити безперебійну роботу бази даних та гарантувати її високу продуктивність.

Крім того, важливо враховувати можливість розширення системи у майбутньому. При виборі сервера для бази даних InfluxDB необхідно враховувати можливість додавання нових серверів у кластер у майбутньому. Це допоможе забезпечити високу масштабованість бази даних та підтримувати її працездатність зі збільшенням обсягу даних.

Для максимальної продуктивності бази даних InfluxDB можна використовувати оптимізовані для цієї бази даних компоненти. Наприклад, можна використовувати SSD-диски для зберігання даних, оскільки вони забезпечують високу швидкість читання та запису даних. Також можна використовувати спеціальні процесори з максимальною підтримкою багатопоточності для обробки великої кількості запитів у реальному часі.

Вибір апаратно-складових серверів для бази даних InfluxDB повинен ґрунтуватися на збалансованому підході. Не варто вибирати найдорожчі та найпотужніші компоненти, якщо вони не відповідають вимогам бази даних. Натомість необхідно вибирати компоненти, які забезпечать оптимальну продуктивність бази даних та відповідають очікуваним навантаженням. Правильний вибір забезпечить стабільну роботу бази даних, високу продуктивність та ефективність.

2.4.2 Вибір операційної системи для серверу

Linux є чудовою операційною системою для запуску сервера, і в поєднанні з InfluxDB вона може забезпечити потужну платформу для зберігання та аналізу даних часових рядів. InfluxDB — це популярна база даних часових рядів із відкритим кодом, розроблена для обробки великих обсягів даних із високою продуктивністю запису та запитів.

Однією з найбільших переваг використання Linux для сервера є його стабільність і надійність. Linux відомий своєю надійністю, безпекою та здатністю справлятися з великими навантаженнями. Крім того, Linux можна налаштовувати та оптимізувати для конкретних випадків використання, що робить його чудовим вибором для запуску InfluxDB.

InfluxDB також добре підходить для роботи в Linux, оскільки він був розроблений, щоб бути легким і ефективним, споживаючи мінімальні системні ресурси. Це означає, що навіть на апаратному забезпеченні з нижчими специфікаціями InfluxDB може забезпечувати чудову продуктивність.

Ще однією перевагою використання Linux для сервера з InfluxDB є широкий спектр інструментів і бібліотек, доступних для розробників і системних адміністраторів. Linux має активну спільноту з відкритим вихідним кодом, і є багато доступних ресурсів для усунення несправностей, оптимізації продуктивності та інтеграції з іншими програмними інструментами.

Що стосується безпеки, Linux зазвичай вважається більш безпечною операційною системою, ніж багато інших її альтернатив. Частково це пов’язано з його природою з відкритим вихідним кодом, що дозволяє проводити більш ретельний і прозорий аудит коду. Linux також має надійні вбудовані функції безпеки, такі як дозволи та контроль доступу, які можуть допомогти захистити конфіденційні дані, що зберігаються в InfluxDB.

Загалом, використання Linux для сервера з InfluxDB є чудовим вибором для організацій, яким потрібна надійна, високопродуктивна платформа для зберігання та аналізу даних часових рядів. Стабільність, продуктивність і безпека Linux у поєднанні з ефективністю та масштабованістю InfluxDB створюють потужне та гнучке рішення, яке може задовольнити потреби в широкому діапазоні випадків використання.

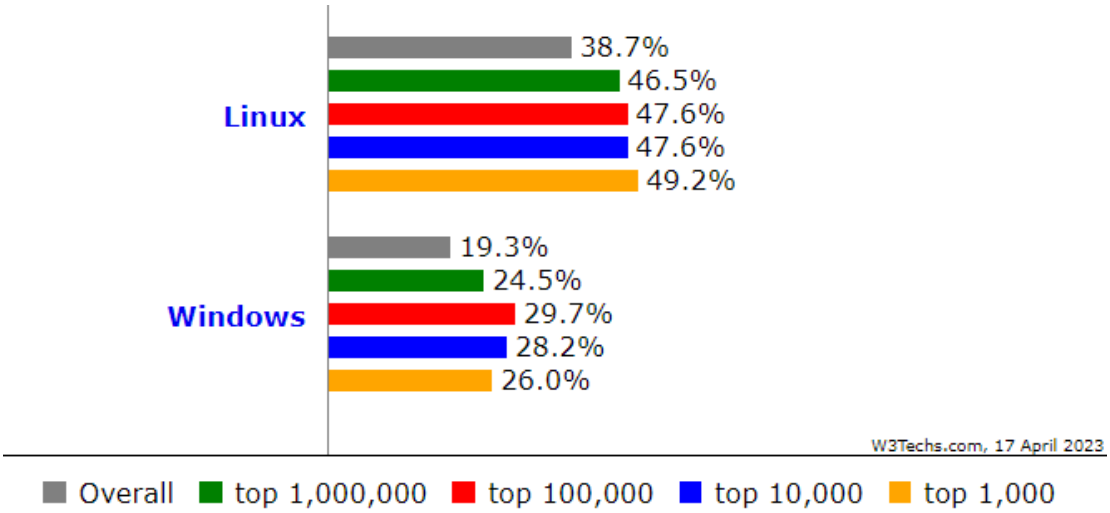

Рисунок 2.4 – Відсоток веб-сайтів, які використовують вибрані операційні системи, з розподілом за рейтингом

Використання Windows Server як операційної системи для сервера з InfluxDB може бути життєздатним варіантом для організацій, які мають існуючу інфраструктуру на базі Windows і знайомі з платформою. Однак слід пам’ятати про кілька міркувань.

Однією з головних переваг використання Windows Server є простота використання та звичність для тих, хто вже знайомий з операційною системою Windows. Крім того, багато організацій уже інвестували в технології Microsoft і, можливо, віддадуть перевагу використанню Windows Server як частини існуючої інфраструктури.

Windows Server також пропонує надійний набір функцій для керування серверними середовищами, включаючи Active Directory, яка може спростити керування користувачами та групами, і Remote Desktop Services, яка може забезпечити віддалений доступ до сервера. Ці функції можуть бути корисними для організацій, яким потрібен високий ступінь контролю та управління серверними середовищами.

Проте є також деякі потенційні недоліки використання Windows Server для сервера з InfluxDB. Одна з проблем полягає в тому, що Windows Server може бути більш ресурсномісткими, ніж Linux, що може вплинути на продуктивність і масштабованість, особливо для великих наборів даних.

Крім того, Windows Server історично був мішенню для кібератак, а це означає, що організації повинні переконатися, що їхні сервери належним чином захищені, щоб запобігти витоку даних та іншим інцидентам безпеки. Це може вимагати додаткових зусиль і ресурсів для підтримки та управління.

Загалом, хоча використання Windows Server для сервера з InfluxDB може бути життєздатним варіантом для деяких організацій, важливо ретельно розглянути можливі компроміси та переконатися, що платформа належним чином захищена та оптимізована для конкретного випадку використання.

2.4.3 Вибір іншого системного забезпечення

У сучасних проектах з мікроконтролерами та віддаленими базами даних широко використовуються різні протоколи для обміну даними між пристроями та серверами. Взаємозв'язок між протоколами відіграє у створенні ефективних і надійних систем.

Сучасні проекти з мікроконтролерами часто вимагають точного часу, збору та передачі даних з датчиків на віддалений сервер та збереження їх у базі даних для подальшої обробки та аналізу. Для вирішення цих завдань можна використовувати протоколи NTP, MQTT та Mosquitto.

Протокол NTP (Network Time Protocol) – це протокол, який використовується для синхронізації часу на пристроях в Інтернеті. Синхронізація часу здійснюється за допомогою точного часу передачі з одного джерела на інший. У проектах з мікроконтролерами NTP може бути корисним

у випадках, коли точність часу критична для роботи пристрою. Наприклад, у разі роботи з датчиками, які вимірюють значення в певний момент часу, необхідно, щоб час було точно синхронізовано з іншими пристроями в мережі. [12]

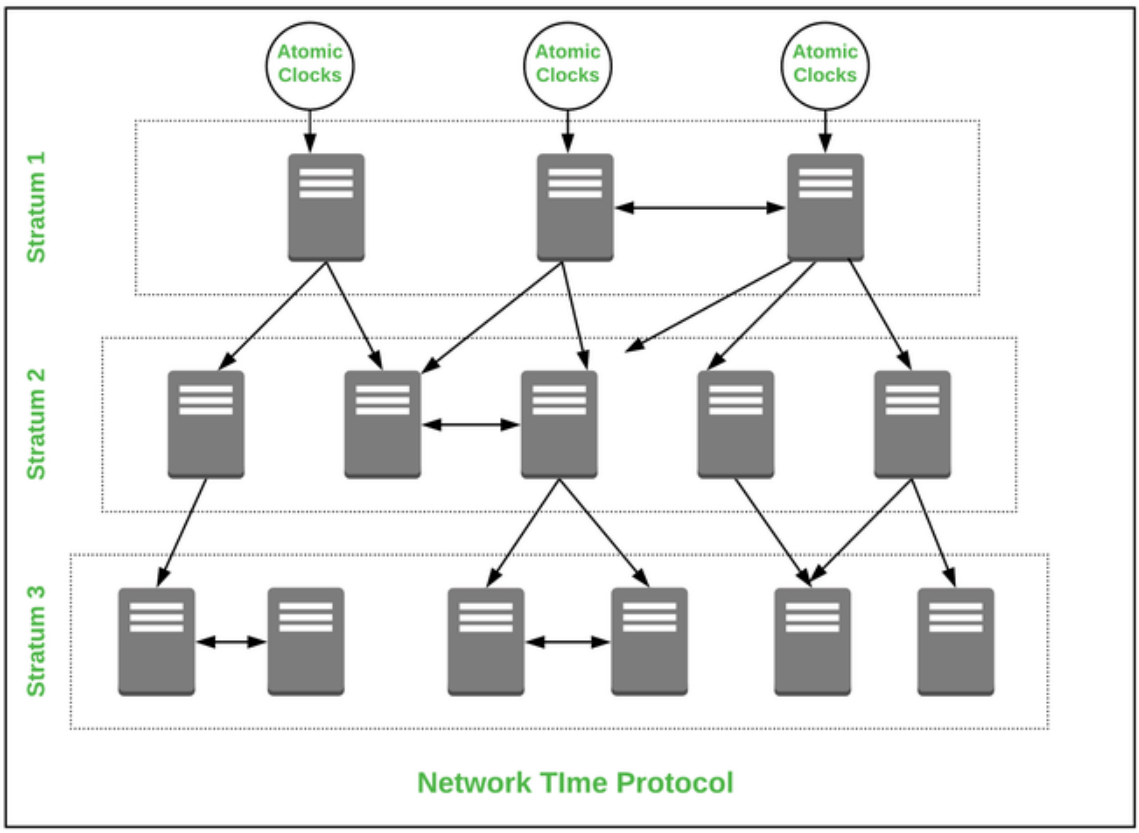

Рисунок 2.5 – Архітектура протоколу NTP

Для використання NTP на мікроконтролерах необхідно мати доступ до Інтернету та використовувати бібліотеки, які дозволяють синхронізувати час пристрою. Наприклад, в Arduino можна використовувати бібліотеку NTPClient, яка дозволяє синхронізувати час із серверами NTP.

Загалом протокол NTP є важливим інструментом для роботи з мікроконтролерами та іншими пристроями в мережі. Він дозволяє забезпечити точність часу на пристроях та синхронізувати час між пристроями, що особливо важливо у системах, де час відіграє критичну роль.

Протокол MQTT (Message Queuing Telemetry Transport) – це легкий протокол, який дозволяє обмінюватися повідомленнями між пристроями в режимі реального часу. Цей протокол був спеціально розроблений для використання в умовах низької пропускної спроможності та надійної передачі даних. [13]

Одним із найбільш затребуваних застосувань MQTT є його використання в проектах, пов'язаних із мікроконтролерами та віддаленим сервером. Протокол MQTT заснований на моделі видавця-передплатника. Він дозволяє пристроям підписуватись на певні теми та надсилати повідомлення на ці теми. Кожне повідомлення містить тему та дані, пов'язані з цією темою. Сервер, підключений до тієї ж теми, отримає повідомлення і може обробити його відповідним чином.

Для роботи з протоколом MQTT на мікроконтролері використовуються бібліотеки, які дозволяють встановлювати з'єднання з брокером MQTT та надсилати повідомлення. Одна з найпопулярніших бібліотек для роботи з MQTT на мікроконтролерах – це PubSubClient, яка надає простий інтерфейс для роботи з протоколом MQTT.

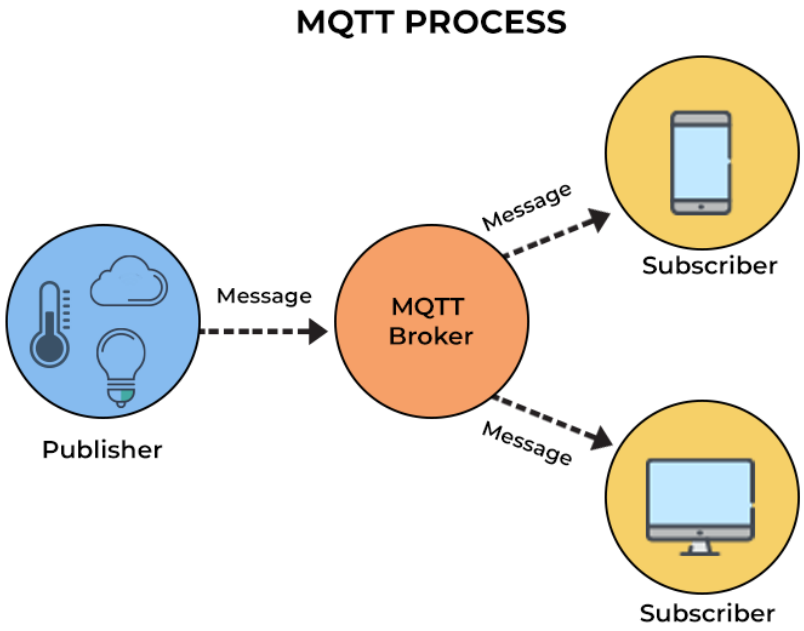

Рисунок 2.6 – Архітектура протоколу MQTT

Переваги протоколу MQTT у проектах з мікроконтролерами та віддаленим сервером включають:

- Економія ресурсів: MQTT – це протокол легкої ваги, який дозволяє заощаджувати ресурси мікроконтролерів та забезпечувати ефективну передачу даних у мережу.
- Зручність використання: Протокол MQTT дозволяє передавати повідомлення в режимі реального часу та використовувати його на мікроконтролерах для керування пристроями у віддаленій мережі.
- Надійність: MQTT забезпечує надійну передачу даних між пристроями та гарантує доставку повідомлень у разі обриву з'єднання.

Однак, необхідно враховувати деякі обмеження під час використання протоколу MQTT:

Одне з основних обмежень MQTT - обмежений розмір пакетів даних. За замовчуванням MQTT не дозволяє надсилати пакети даних розміром більше 256 МБ. Це означає, що якщо ви хочете передавати великі обсяги даних, вам може знадобитися розбити їх на дрібніші пакети і надсилати їх частинами.

Ще одне обмеження MQTT – це відсутність вбудованої підтримки безпеки. MQTT не надає механізмів для шифрування даних або автентифікації користувачів. Для забезпечення безпеки вам може знадобитися використовувати інші інструменти, такі як TLS/SSL або VPN.

Також слід враховувати, що MQTT - це протокол передачі повідомлень, а чи не передачі великих обсягів даних. Якщо ви плануєте надсилати багато даних, можливо, вам доведеться використовувати інший протокол, наприклад, HTTP.

Нарешті, при використанні MQTT ви повинні враховувати ліміти швидкості передачі даних та навантаження на мережу. Якщо надсилати занадто багато повідомлень занадто швидко, ви можете перевантажити свою мережу і створити затримки або втрату даних.

В цілому, MQTT - протокол, який дозволяє зручно та надійно передавати дані між пристроями у розподіленій системі. Завдяки своїй простоті та легкості у використанні, він широко застосовується в IoT-проектах, включаючи системи розумного будинку, моніторинг та управління виробничими процесами, автоматизацію будівель та інші.

Протокол Mosquitto – це брокер повідомлень, який працює за протоколом MQTT (Message Queuing Telemetry Transport). Mosquitto був створений для забезпечення швидкої та надійної передачі повідомлень між пристроями, які використовують протокол MQTT.

Mosquitto може бути використаний на різних платформах, включаючи Linux, Windows та MacOS. Він підтримує шифрування SSL / TLS для безпечної передачі даних, а також може бути налаштований для автентифікації користувачів та керування доступом. [14]

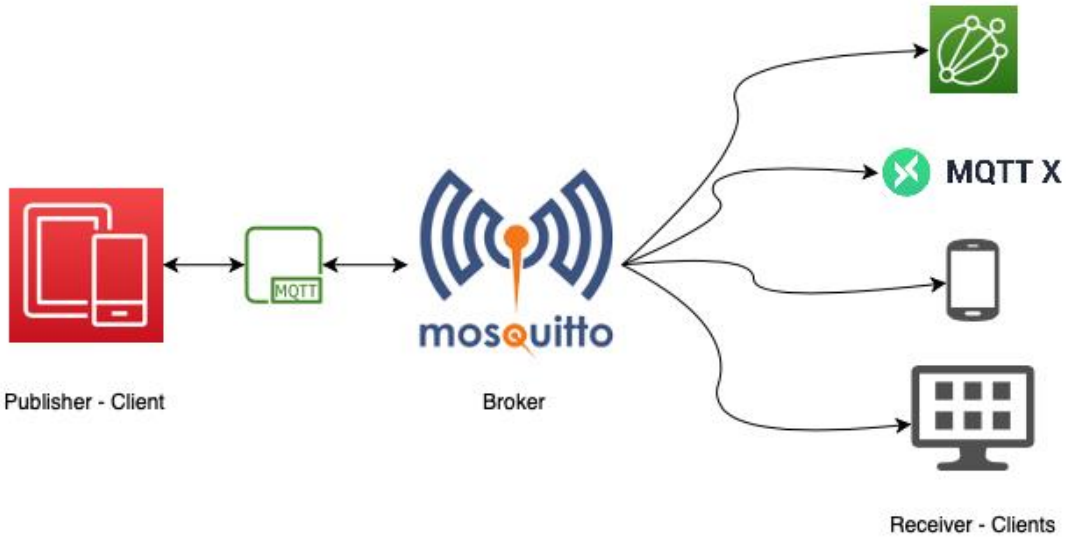

Рисунок 2.7 – Архітектура протоколу Mosquitto

Однією з основних переваг Mosquitto є його простота та легкість у використанні. Mosquitto має невеликий розмір і не вимагає великого обсягу пам'яті або процесорної потужності, що робить його ідеальним для використання на пристроях з обмеженими ресурсами, таких як мікроконтролери.

Mosquitto також має гнучкі налаштування, які дозволяють налаштувати брокер повідомлень під певні вимоги проекту. Наприклад, Mosquitto підтримує визначення максимального розміру повідомлення, час зберігання повідомлень у черзі та інші параметри.

Однак, Mosquitto також має деякі обмеження. Він не підтримує QoS (Quality of Service) рівня 2, який забезпечує доставку повідомлень без втрат і в порядку їх надсилання. Крім того, Mosquitto не забезпечує гарантовану доставку повідомлень, що означає, що повідомлення можуть бути втрачені при передачі або при неполадках в мережі.

Загалом, Mosquitto є надійним та зручним інструментом для передачі повідомлень за протоколом MQTT у проектах з мікроконтролерами та віддаленим сервером. Завдяки своїй простоті та гнучкості Mosquitto може бути використаний у різних додатках, від простих домашніх автоматизації до складних IoT-систем.

2.5 Огляд та вибір мікроконтролерного рішення для проекту системи вимірювання.

Вибираючи рішення мікроконтролера для проекту, який включає як платформу Arduino, так і системи вимірювання, слід враховувати кілька факторів, щоб переконатися, що вибране рішення найкраще відповідає вимогам проекту.

Одним із основних факторів, який слід враховувати, є тип вимірювальної системи, яка буде використовуватися. Різні мікроконтролери мають різні можливості аналого-цифрового перетворювача (АЦП), тому важливо вибрати мікроконтролер з АЦП, який здатний точно вимірювати фізичні параметри, які стосуються проекту. Наприклад, якщо проект вимагає вимірювання високоточних сигналів, може знадобитися мікроконтролер з АЦП високої роздільної здатності. [15]

Іншим фактором, який слід враховувати, є вимоги проекту до підключення. Якщо проект потребує підключення до Інтернету, найкращим вибором може бути мікроконтролер із вбудованим підключенням Wi-Fi, наприклад Arduino ESP8266 який і буде використаний надалі. Крім того, якщо проект вимагає зв'язку з іншими пристроями через дротове з'єднання, мікроконтролер із підключенням до Ethernet може бути більш придатним.

Обчислювальна потужність мікроконтролера також є важливим фактором. Якщо проект вимагає складних розрахунків або контролю в режимі реального часу, може знадобитися мікроконтролер із швидким процесором. Крім того, слід також враховувати пам'ять і можливості зберігання мікроконтролера, оскільки вони можуть вплинути на продуктивність проекту та здатність зберігати дані.

Нарешті, також слід враховувати сумісність з платформою Arduino. Мікроконтролер, сумісний з платформою Arduino, можна програмувати за допомогою Arduino Integrated Development Environment (IDE), що полегшує розробку та завантаження коду на пристрій. Крім того, мікроконтролер повинен бути сумісний з відповідними бібліотеками та інструментами, необхідними для проекту.

Підсумовуючи, під час вибору мікроконтролерного рішення для проекту, який включає як платформу Arduino, так і вимірювальні системи, важливо враховувати такі фактори, як тип вимірювальної системи, вимоги до підключення, обчислювальна потужність, пам'ять і можливості зберігання, а також сумісність з Платформа Arduino. Беручи до уваги ці фактори, можна

вибрати рішення, яке найкраще відповідає вимогам проекту та забезпечить бажаний рівень продуктивності та функціональності.

2.5.1 Вибір мікроконтролеру ESP8266

Arduino ESP8266 (рисунок 2.1) — це мікроконтролер, який став популярним завдяки простоті використання та універсальності. Він базується на модулі Wi-Fi ESP8266 і забезпечує просту та доступну платформу для окремих людей для розробки та створення прототипів Інтернету речей (Internet of Things, IoT) та інших підключених технологічних проектів [16].

Рисунок 2.8 – мікроконтролер ESP8266

Однією з ключових особливостей Arduino ESP8266 є підключення до Wi-Fi, що дозволяє мікроконтролеру підключатися до Інтернету та спілкуватися з іншими пристроями через мережу. Це відкриває широкий спектр можливостей для проектів, включаючи віддалений моніторинг і контроль, збір і аналіз даних, а також домашню автоматизацію. Модуль

ESP8266 має вбудовану антену Wi-Fi, що полегшує підключення до бездротової мережі, а також підтримує низку різних протоколів Wi-Fi, що робить його універсальним вибором для додатків IoT.

Іншим важливим аспектом Arduino ESP8266 є його сумісність з платформою Arduino. Це означає, що його можна програмувати за допомогою інтегрованого середовища розробки Arduino (IDE), яке забезпечує простий та інтуїтивно зрозумілий інтерфейс для розробки, тестування та завантаження коду в мікроконтролер. Arduino IDE також підтримує широкий спектр бібліотек, що полегшує додавання функцій і взаємодію з іншими компонентами, такими як датчики та виконавчі механізми. Це робить його ідеальною платформою для людей, які хочуть швидко створювати прототипи та створювати проекти без необхідності вивчати складні мови програмування.

Arduino ESP8266 також має широкі можливості налаштування та гнучкості. Він має велику кількість контактів введення/виведення (I/O), які можна використовувати для підключення до інших компонентів і периферійних пристроїв, і його можна легко інтегрувати з іншими технологіями, такими як датчики, приводи та дисплеї. Крім того, мікроконтролер має малий форм-фактор, що дозволяє легко інтегрувати його в компактні проекти, а також має низьку потужність, що робить його придатним для проектів з живленням від акумулятора.

Підсумовуючи, Arduino ESP8266 — це універсальний і доступний мікроконтролер, який добре підходить для IoT та інших підключених технологічних проектів. Його підключення до Wi-Fi і сумісність з платформою Arduino спрощують розробку та прототипування проектів, а його гнучкість і можливість налаштування роблять його ідеальним для широкого спектру застосувань.

2.5.2 Вибір датчиків та сенсорів BME280 та DHT11

Датчики є важливим компонентом у багатьох проектах на основі мікроконтролерів, що дозволяє збирати й аналізувати дані з фізичного світу. Платформа Arduino сумісна з широким спектром датчиків, включаючи BME280 і DHT11.

BME280 (рисунок 2.2) — це високоточний датчик навколишнього середовища, який вимірює температуру, вологість і тиск. Він використовує комбінацію датчиків температури та тиску для обчислення відносної вологості, що робить його універсальним вибором для широкого спектру застосувань. BME280 також є високоточним, з точністю вимірювання температури ±0,5°C і точністю вологості ±3% відносної вологості.

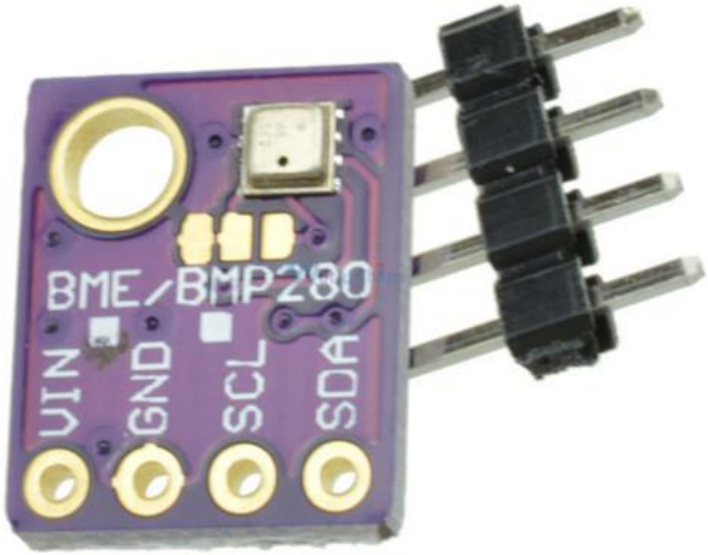

Рисунок 2.9 – датчик BME280

BME280 зазвичай використовується в проектах IoT, системах моніторингу погоди та програмах збору даних про навколишнє середовище. Його можна використовувати для вимірювання та відстеження змін температури та вологості в кімнаті, теплиці чи іншому середовищі або для моніторингу погодних умов у певному місці. BME280 також ідеально

підходить для використання в переносних пристроях, де його можна використовувати для моніторингу температури шкіри та відносної вологості користувача [17].

DHT11 (рисунок 2.3) — ще один популярний датчик навколишнього середовища, який вимірює температуру та вологість. На відміну від BME280, DHT11 є недорогим і низькоточним датчиком, який ідеально підходить для хобі-проектів і простих програм моніторингу. DHT11 має точність температури ±2°C і вологості ±5% RH, що робить його придатним для збору базових даних і моніторингу.

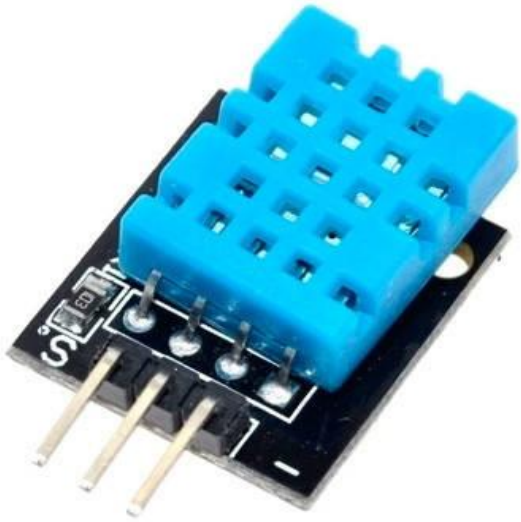

Рисунок 2.10 – датчик DHT11

DHT11 зазвичай використовується в проектах домашньої автоматизації, таких як розумні термостати та системи моніторингу температури, а також у проектах, де показання температури та вологості потрібні протягом обмеженого періоду часу. DHT11 також простий у використанні та не вимагає додаткових компонентів, що робить його популярним вибором для виробників і любителів. [18]

Підсумовуючи, і BME280, і DHT11 є універсальними та корисними датчиками для проектів Arduino. BME280 — це високоточний датчик, який ідеально підходить для додатків IoT і збору даних про навколишнє середовище, тоді як DHT11 — це недорогий і низькоточний датчик, який добре підходить для хобі-проектів і простих програм моніторингу. Вибираючи датчик для проекту Arduino, важливо враховувати точність, яка необхідна для вашої програми, а також вартість і простоту використання датчика.

2.5.3 Вибір програмних засобів реалізації

Вибір програмних засобів для реалізації проекту має вирішальне значення для успіху проекту. Правильні програмні засоби можуть допомогти оптимізувати процес розробки, зменшити ризик помилок і забезпечити надійність і легкість кінцевого продукту.

Перш за все, програмні засоби повинні бути сумісні з апаратними компонентами. Наприклад, програмні засоби, які використовуються для плати ESP8266, повинні бути сумісні з мікросхемою та її мікропрограмою. Подібним чином програмні засоби, що використовуються для датчиків BME280 і DHT11, повинні мати можливість обмінюватися даними з датчиками через інтерфейс I2C. Вибираючи програмні засоби, сумісні з апаратними компонентами, можна зменшити ризик помилок і забезпечити надійність кінцевого продукту.

Іншим важливим моментом є рівень технічних знань, необхідних для використання програмних засобів. Наприклад, програмні інструменти, які вимагають високого рівня технічної експертизи, можуть бути непридатними для використання любителями чи початківцями. З іншого боку, програмні інструменти, які прості у використанні та не вимагають технічних знань або не потребують жодних, можуть зробити процес розробки набагато простішим і швидшим.

Наявність бібліотек і зразків коду також є важливим моментом. Бібліотеки та зразки коду можуть допомогти спростити процес розробки, зменшити ризик помилок і заощадити час. Наприклад, наявність бібліотек для плати ESP8266 і датчиків BME280 і DHT11 може значно полегшити читання даних датчиків і обмінюватися даними з мережею.

Вартість програмних засобів також є важливим фактором. Хоча безкоштовні програмні засоби або інструменти з відкритим кодом можуть бути привабливими через їхню низьку вартість, вони можуть бути непридатними для використання в комерційних проектах через обмеження комерційного використання та відсутність підтримки. З іншого боку, комерційні програмні засоби можуть запропонувати вищий рівень підтримки та надійності, але вони можуть бути дорожчими.

Вибір програмних засобів для реалізації проекту Arduino має вирішальне значення для успіху проекту. Вибираючи програмні інструменти, які сумісні з апаратними компонентами, вимагають низького рівня технічної експертизи, мають велику кількість бібліотек і прикладів коду та є економічно ефективними, ви можете переконатися, що кінцевий продукт надійний, простий у використанні, і відповідає вашим потребам.

Найкращим програмним інструментом для реалізації мого проекту Arduino з платою ESP8266 і датчиками BME280 і DHT11 буде інтегроване середовище розробки Arduino (IDE). Arduino IDE сумісна з платою ESP8266 і надає велику кількість бібліотек і зразків коду для датчиків BME280 і DHT11. Крім того, Arduino IDE проста у використанні та вимагає лише базового рівня технічних знань. Крім того, це безкоштовний інструмент із відкритим кодом, що робить його привабливим вибором для любителів і новачків. Нарешті, велике та активне співтовариство користувачів і розробників Arduino IDE може надати підтримку та поради, забезпечуючи плавний та успішний процес розробки. Вибір Arduino IDE як програмного інструменту для цього проекту є розумним і враховує всі важливі фактори, включаючи сумісність, простоту використання, наявність бібліотек і прикладів коду, вартість і підтримку. [19]

РОЗДІЛ 3
ПРОГРАМНО-АПАРАТНА РОЗРОБКА ТА ВИПРОБУВАННЯ

3.1 Встановлення бази даних InfluxDB

InfluxDB — це популярна база даних часових рядів із відкритим кодом, розроблена для обробки великих обсягів даних із високою швидкістю. Вона забезпечує масштабоване та ефективне рішення для зберігання та аналізу даних із мітками часу. Для того щоб її встановити я виконав такі кроки встановлення InfluxDB в операційній системі Windows:

Крок 1. Завантаження InfluxDB. Щоб розпочати процес встановлення, я відвідав сторінку завантажень InfluxDB за адресою https://portal.influxdata.com/downloads . Вибрав відповідну версію InfluxDB для Windows відповідно моєї архітектури системи (64-розрядна). Натиснув посилання для завантаження, щоб почати завантаження виконуваного файлу інсталятора InfluxDB (.exe).

Крок 2. Запуск інсталятора: після завершення завантаження знайшов завантажений файл інсталятора та відкрив його. Інсталятор провів мене через процес встановлення.

Крок 3: Вибір каталогу встановлення: Вибрав папку призначення, у яку потрібно встановити InfluxDB. Можна вибрати каталог за замовчуванням або вказати інший відповідно до потреб користувача. Натиснув «Далі», щоб продовжити.

Крок 4: Вибір компонентів: інсталятор надав мені список компонентів, які можна встановити. За замовчуванням вибрано всі компоненти. Для базової інсталяції рекомендується зберегти вибір за замовчуванням. Однак можна налаштувати інсталяцію, встановивши або знявши прапорці відповідно до вимог користувача. Натиснув «Далі», щоб продовжити.

Крок 5: Перегляд відомостей про встановлення: буде відображено підсумок вибраних налаштувань. Я переглянув короткий опис встановлення, щоб переконатися, що все виглядає правильно. Якщо мені було б потрібно внести будь-які зміни, я міг би повернутися назад і змінити налаштування.

Крок 6: Перевірка інсталяції: щоб переконатися, що InfluxDB встановлено та працює правильно, я відкрив веб-браузер і перейшов до адреси http://localhost:8086. InfluxDB працює належним чином, я побачив домашню сторінку InfluxDB та авторизувався.

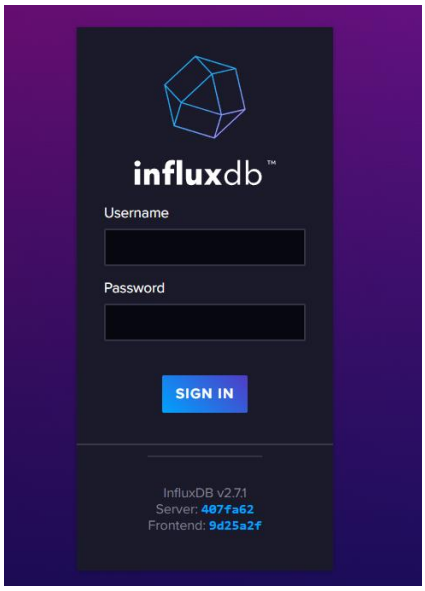

Рисунок 3.1 – Сторінка входу до бази даних

Крок 7: Створення бази даних: для того щоб створити саму базу даних я перейшов до розділу «Data» (Дані) після успішного входу в систему та вибрав розділ «Data» (Данні) у верхньому меню веб-інтерфейсу. Та нажав кнопку створити нову базу даних «bucket» якою я дав назву «Sensor_data».

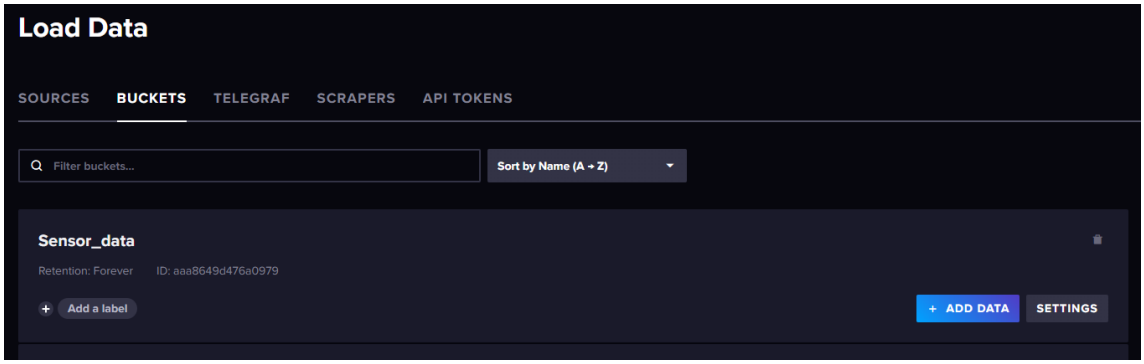

Рисунок 3.2 – Створена база даних InfluxDB

Дотримуючись наведених кроків, я успішно інсталював InfluxDB на моїй операційній системі Windows та створив базу даних «Sensor_data» в яку будуть зберігатися дані отриманні з мікроконтролерів.

3.2 Встановлення бази даних MySQL

MySQL — це популярна система керування реляційними базами даних (RDBMS) з відкритим вихідним кодом, яка дозволяє користувачам ефективно зберігати, керувати та отримувати дані. Для того щоб її встановити я виконав такі кроки встановлення MySQL в операційній системі Windows:

Крок 1: Отримав інсталятор MySQL. Відвідав офіційний веб-сайт MySQL (https://www.mysql.com/) і перейшов до розділу завантажень. Вибрав відповідну версію MySQL на основі моєї операційної системи. Переконався, що вибрав стабільну та сумісну версію MySQL.

Крок 2. Запустив інсталятор MySQL. Після завантаження пакета інсталятора MySQL знайшов завантажений файл та запустив його, щоб розпочати процес встановлення.

Крок 3: Вибрав продукти MySQL: Основним компонентом, який мені знадобиться, є «сервер MySQL», який є ядром RDBMS. Також я обрав додатковий інструмент, MySQL Workbench для графічного проектування та адміністрування баз даних.

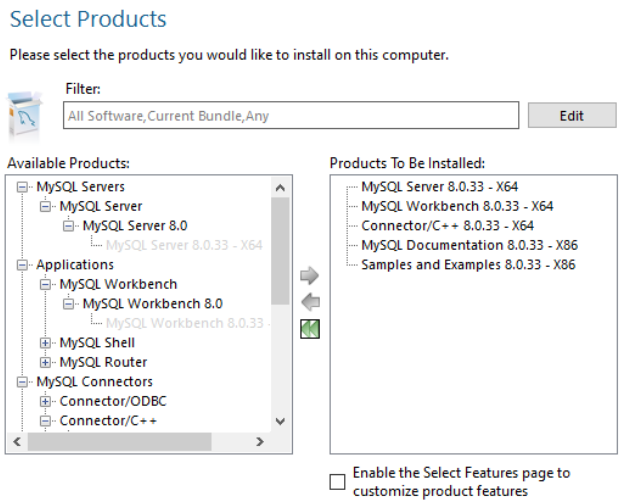

Рисунок 3.3 – Обрані компоненти при інсталяції

Крок 4: тип конфігурації MySQL Server: Конфігурації забезпечують попередньо визначені параметри, оптимізовані для конкретних цілей, таких як розробка, виробництво або використання виділеної пам’яті. Я виберу конфігурацію, яка найкраще відповідає моїм потребам для особистого користування та тестування бази даних.

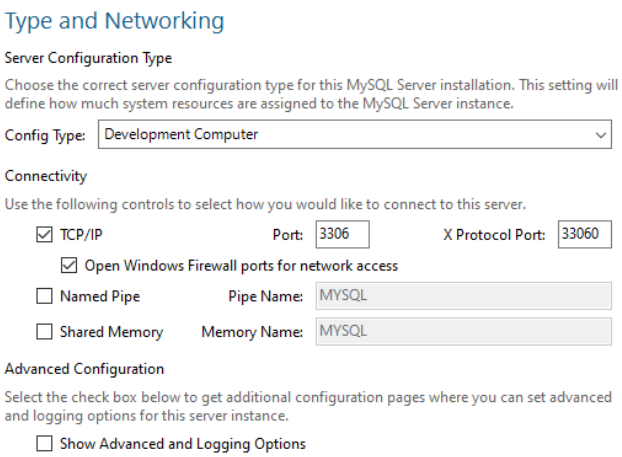

Рисунок 3.4 – Обрана конфігурація Development Computer

Крок 5: Створена база даних: Після автоматичної інсталяції я створив базу даних, обрав її необхідні параметри та успішно запустив.

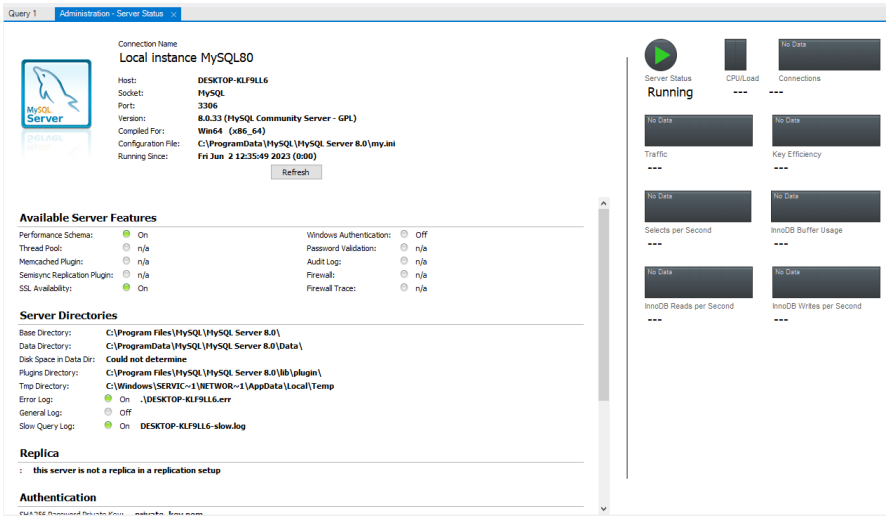

Рисунок 3.5 – Запущена база даних MySQL

Дотримуючись наведених кроків, я успішно інсталював другу необхідну базу даних MySQL на моїй операційній системі Windows, тож я можу перейти до підключення та програмування мікроконтролерів до моїх баз даних.

3.3 Підключення та програмування мікроконтролера з датчиками

Для того щоб заповнити бази даних та дослідити і порівняти їх ефективність при навантаженні, для початку я підключу мікроконтролер з датчиками та налаштую відправку та зберігання даних на базах даних. Щоб зібрати та передати дані, я використовуватиму мікроконтролер ESP8266 та датчики BME280 та DHT11.

ESP8266 потужний і компактний контролер підтримує Wi-Fi з'єднання, що робить його дуже зручним для передачі даних через мережу. Щоб підключити ESP8266, я дотримувався кількох кроків:

Крок 1: Підключив ESP8266 до комп'ютера за допомогою кабелю USB.

Крок 2: Встановив драйвери ESP8266, щоб мій комп'ютер міг розпізнати контролер.

Крок 3: Завантажив Arduino IDE та налаштував його для роботи з ESP8266.

Крок 4: В Arduino IDE я вибрав правильну плату (ESP8266) та порт, до якого було підключено мій мікроконтролер.

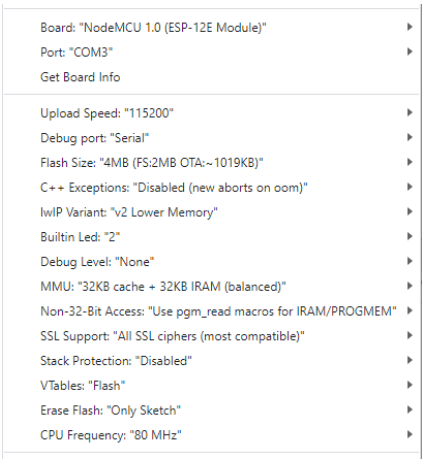

Рисунок 3.6 – Параметри підключення Arduino IDE до мікроконтролера

Для збору даних про температуру та вологість я використовуватиму датчики BME280 та DHT11. Ці датчики мають простий інтерфейс та дозволяють отримувати точні вимірювання. Ось як я підключив їх до мого проекту:

Крок 1: Підключив датчики BME280 та DHT11 до пін на ESP8266, дотримуючись схеми підключення.

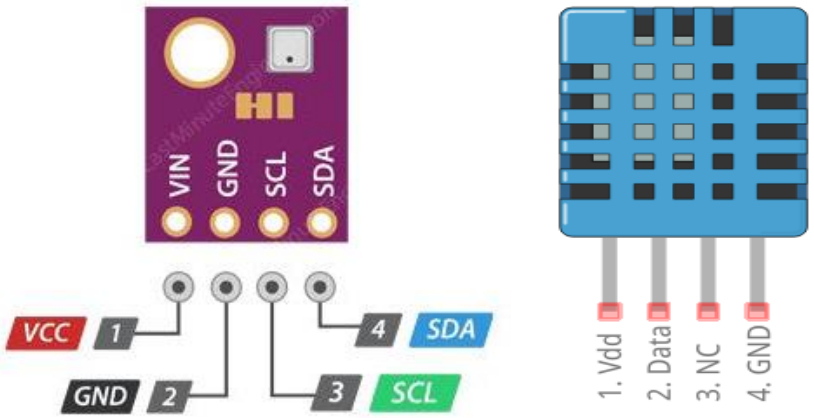

Рисунок 3.7 – Схема розташування виводів BME280 та DHT11

Для з'єднання Esp8266 з датчиками я використав такі піни:

- D1 (Esp8266) – SCL (bme280);
- D2 (Esp8266) – SDA (bme280);

- D3 (Esp8266) – Data (dht11);
- VCC (Esp8266) – VIN (bme280);
- VCC (Esp8266) – VDD (dht11);
- GND (Esp8266) – GND (bme280);
- GND (Esp8266) – GND (dht11);

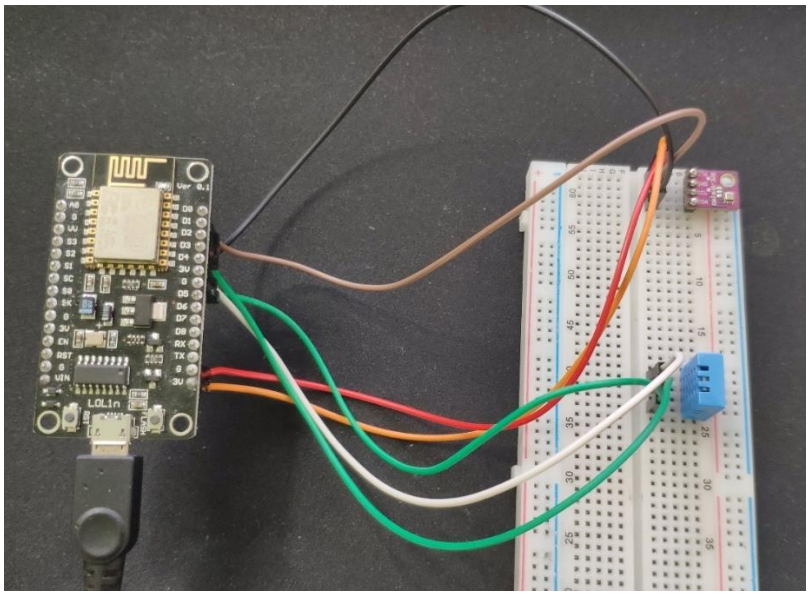

Рисунок 3.8 – З'єднаний мікроконтролер з датчиками

Крок 2: Встановив відповідні бібліотеки для роботи з датчиками Arduino IDE.

```
#if defined(ESP32)
#include <WiFiMulti.h>
WiFiMulti wifiMulti;
#define DEVICE "ESP32"
#elif defined(ESP8266)
#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti wifiMulti;
#define DEVICE "ESP8266"
#endif
#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>
#include <WiFiClient.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <DHT.h>
```

Рисунок 3.6 – Бібліотеки, що використовуються

Крок 3: Написав програму на мові Arduino для зчитування даних із датчиків та їх обробки.

setup(): Ця функція виконується один раз при запуску мікроконтролера і служить для налаштування необхідних компонентів та з'єднань.

- Ініціалізується зв'язок із послідовним портом для налагоджувальних повідомлень.
- Перевіряється наявність та ініціалізується датчик BME280.
- Ініціалізується датчик DHT11.
- Налаштовується режим Wi-Fi на режим станції (WIFI_STA) і додається точка доступу (AP) із вказаним ім'ям (WIFI_SSID) та паролем (WIFI_PASSWORD).
- Встановлюється з'єднання Wi-Fi та очікується підключення до мережі.
- Синхронізується час за допомогою сервера NTP.
- Перевіряється з'єднання з сервером InfluxDB і відображається інформація про підключення.
- Додаються теги (device, SSID) до об'єктів точок даних (sensor та sensor_data).

Лістинг 3.2 – Функція setup

```
void setup() {
  Serial.begin(115200);
  if (!bme.begin(0x76)) {  // Инициализируем BME280 с адресом 0x76
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }
  dht.begin();
  // Setup wifi
  WiFi.mode(WIFI_STA);
  wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to wifi");
  while (wifiMulti.run() != WL_CONNECTED) {
    Serial.print(".");
    delay(100);
  }
  Serial.println();
  // Accurate time is necessary for certificate validation and writing in
batches
```

```
  // We use the NTP servers in your area as provided by:
https://www.pool.ntp.org/zone/
  // Syncing progress and the time will be printed to Serial.
  timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");
  // Check server connection
  if (client.validateConnection()) {
    Serial.print("Connected to InfluxDB: ");
    Serial.println(client.getServerUrl());
  } else {
    Serial.print("InfluxDB connection failed: ");
    Serial.println(client.getLastErrorMessage());
  }
  sensor.addTag("device", DEVICE);
  sensor.addTag("SSID", WiFi.SSID());
  sensor_data.addTag("device", DEVICE);
  sensor_data.addTag("SSID", WiFi.SSID());
}
```

loop(): Ця функція виконується у циклі після завершення функції setup(). Вона служить для читання даних з датчиків та надсилання даних в СУБД.

- Зчитуються значення температури та вологості з датчиків BME280 та DHT11.
- Заповнюються поля даних точки sensor_data значеннями температури та вологості з датчиків.
- Заповнюється поле даних точки sensor значенням сигналу Wi-Fi (RSSI).
- Відображаються значення температури та вологості з датчика BME280 у послідовному порту.
- Відображається інформація про дані, які будуть записані в InfluxDB.
- Перевіряється підключення Wi-Fi та перепідключення, якщо необхідно.
- Записуються дані в InfluxDB за допомогою методу **writePoint()**.
- Записуються дані MySQL за допомогою SQL-запиту INSERT.
- Закривається з'єднання з базою MySQL.
- Очікування 1 секунду перед повторним виконанням циклу.

Лістинг 3.3 – Функція loop

```
void loop() {
  // sensor.clearFields();
  // sensor_data.clearFields();
```

```
float bmeTemperature = bme.readTemperature();
float bmeHumidity = bme.readHumidity();
float dhtTemperature = dht.readTemperature();
float dhtHumidity = dht.readHumidity();
sensor_data.addField("bme_temperature", bmeTemperature);
sensor_data.addField("bme_humidity", bmeHumidity);
sensor_data.addField("dht_temperature", dhtTemperature);
sensor_data.addField("dht_humidity", dhtHumidity);
sensor.addField("rssi", WiFi.RSSI());
Serial.print("Температура BME280: ");
Serial.print(bmeTemperature);
Serial.println("°C");
Serial.print("Влажность BME280: ");
Serial.print(bmeHumidity);
Serial.println("%");
// Print what are we exactly writing
Serial.print("Writing: ");
Serial.println(sensor.toLineProtocol());
// Check WiFi connection and reconnect if needed
if (wifiMulti.run() != WL_CONNECTED) {
  Serial.println("Wifi connection lost");
}
// Write point
if (!client.writePoint(sensor)) {
  Serial.print("InfluxDB write failed: ");
  Serial.println(client.getLastErrorMessage());
}
if (!client.writePoint(sensor_data)) {
  Serial.print("InfluxDB(DATA) write failed: ");
  Serial.println(client.getLastErrorMessage());
}
// Save data to MySQL
if (conn.connect(host, port, user, passwordMySQL)) {
  Serial.println("Connected to MySQL server");
  // Prepare INSERT statement
  char insertStmt[256];
```

```
    sprintf(insertStmt, "INSERT INTO %s.sensor_data (bme_temperature,
bme_humidity, dht_temperature, dht_humidity, rssi) VALUES (%f, %f, %f, %f,
%d)",
            database, bmeTemperature, bmeHumidity, dhtTemperature,
dhtHumidity, WiFi.RSSI());
    // Execute INSERT statement
    cur.execute(insertStmt);
    // Close the connection
    conn.close();
  } else {
    Serial.println("MySQL connection failed");
  }
  Serial.println("Waiting 1 second");
  delay(1000);
}
```

3.4 Порівняльне тестування продуктивності двох СУБД

В даний час вибір відповідної системи управління базами даних (СУБД) є критичним аспектом розробки додатків та забезпечення їх ефективної роботи. СУБД відіграють важливу роль у обробці, зберіганні та доступі до даних, тому порівняльне тестування продуктивності різних СУБД є невід'ємною частиною процесу вибору відповідної системи.

Метою даного порівняльного тестування буде оцінити та порівняти продуктивність роботи з даними часових рядів реляційної та не реляційної СУБД InfluxDB та MySQL на основі на основі ідентичних характеристик та навантаження. Зокрема, я зосередився на швидкості виконання операцій читання та запису, а також загальному часі відгуку кожної СУБД. Для порівняльного тестування, я використовуватиму однакове апаратне забезпечення, що складається з процесора Ryzen 9 5950x, 32-гігабайт оперативної пам'яті та SSD-M2 1 TB накопичувач на операційной системі Windows 10.

На тестовій системі було встановлено останні версії InfluxDB та MySQL, конфігурація та параметри були налаштовані відповідно до рекомендацій виробників та з урахуванням тестового сценарію. Я буду вимірювати час виконання типових операцій та читання даних з обох СУБД, під навантаженням даних в обсязі в 60 000 часових рядів зі значеннями з датчиків зібраних за останніх 7 днів.

Рисунок 3.8 – Спрощений графік зібраних даних за 7 днів

В тестуванні будуть проводиться типи операції вибірки, фільтрації та агрегації даних, до таблиці буде додано значення з 3х тестових повторень, заради точності отриманих даних відповідно до заздалегідь визначених критеріїв:

Таблиця 3.1 – Ефективність обох СУБД при операції вибірки даних

База даних	Вибірка всіх даних	Вибірка даних із фільтром за датою та часом (5 днів)	Вибірка даних із фільтром за температурою та сортуванням за вологістю
MySQL	0.17 секунд	0.14 секунд	0.04 секунд
InfluxDB	0.08 секунд	0.07 секунд	0.02 секунд

Таблиця 3.2 – Ефективність обох СУБД при операції фільтрації даних

База даних	Фільтрування за однією умовою	Фільтрування за кількома умовами з використанням логічного оператора AND	Фільтрування за кількома умовами з використанням логічного оператора OR
MySQL	0.08 секунд	0.04 секунд	0.12 секунд
InfluxDB	0.04 секунд	0.02 секунд	0.06 секунд

Таблиця 3.3 – Ефективність обох СУБД при операції агрегації даних

База даних	Агрегація суми для поля bme_temperature	Агрегація середнього значення для поля bme_humidity з угрупуванням за часовими інтервалами	Агрегація кількості записів за умови, що bme_temperature більше 25
MySQL	0.03 секунд	0.07 секунд	0.03 секунд
InfluxDB	0.01 секунд	0.03 секунд	0.01 секунд

На основі наданих даних, які відображають час виконання операцій вибірки, фільтрації та агрегації даних, можна зробити такі висновки про продуктивність двох СУБД - InfluxDB та MySQL:

InfluxDB показує нижчий час виконання операцій вибірки, фільтрації та агрегації даних порівняно з MySQL. В середньому, InfluxDB виконує ці операції за 0,04 секунди, в той час як MySQL вимагає 0,09 секунди. Це вказує

на те, що InfluxDB здатний обробляти запити на вибірку даних більш ефективно та швидко.

MySQL, у свою чергу, має трохи триваліший час виконання операцій вибірки, фільтрації та агрегації даних, що може вказувати на його менш оптимізовану роботу з данними часових рядів, тобто з часом ефективність реляційної бази даних падатиме залежно від обсягу даних.

3.5 Висновки за розділом

Результатом виконання розділу є створення двох баз даних MySQL та Influxdb, підключення та програмування мікроконтролеру esp8266 з датчиками bme280 та dht11. Після зібрано в бази даних необхідний обсяг даних, та проведено тестування в швидкості виконання в трьох видах запитів, доведена практична різниця в ефективності використання реляційної та нереляційної бази даних для даних часових рядів.

ВИСНОВКИ

В результаті виконання дипломного проекту були розглянуті реляційні та нереляційні системи управління базами даних (СУБД), а також застосування з ними даних часових рядів, мікроконтролера esp8266 та датчиків bme280 та dht11.

У практичній частині роботи було встановлено дві бази даних - MySQL та InfluxDB, та проведено програмування та підключення мікроконтролера до баз даних. Були зібрані дані та проведено дослідження ефективності використання реляційної та нереляційної баз даних з даними тимчасових рядів. Результати дослідження показали, що зі збільшенням обсягу даних часових рядів реляційна база даних стає менш ефективною порівняно з нереляційною базою даних.

Таким чином, дана робота дозволяє зробити висновок про те, що вибір між реляційними та нереляційними СУБД залежить від специфіки проекту та вимог до обробки даних. При роботі з даними тимчасових рядів, особливо при великому обсязі даних, нереляційні СУБД, такі як InfluxDB можуть забезпечити більш ефективне зберігання та обробку даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. Данні часових рядів [Електронний ресурс]. – Режим доступу: www. URL: http://surl.li/hseqt (дата звернення:15.02.23)

2. Relational Databases vs Time Series Databases [Електронний ресурс]. – Режим доступу: www. URL: https://www.influxdata.com/blog/relational-databases-vs-time-series-databases/ (дата звернення:22.02.23)

3. Бази даних [Електронний ресурс]. – Режим доступу: www. URL: http://surl.li/hnpko (дата звернення:26.02.23)

4. База даних wikipedia [Електронний ресурс]. – Режим доступу: www. URL: https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85 (дата звернення:29.02.23)

5. Вступний посібник із даних часових рядів [Електронний ресурс]. – Режим доступу: www. URL: http://surl.li/hsesn (дата звернення:02.03.23)

6. СУБД та часові ряди [Електронний ресурс]. – Режим доступу: www. URL: https://habr.com/ru/company/oleg-bunin/blog/329062/ (дата звернення:6.03.23)

7. https://freehost.com.ua/ukr/faq/wiki/chto-takoe-mysql/ [Електронний ресурс]. – Режим доступу: www. URL: Що таке MySQL (дата звернення:09.03.23)

8. Розгортання MongoDB [Електронний ресурс]. – Режим доступу: www. URL: https://devzone.org.ua/post/5-rechey-yaki-slid-znati-pered-rozgortannyam-mongodb (дата звернення:12.03.23)

9. Знайомство з InfluxDB [Електронний ресурс]. – Режим доступу: www. URL: https://tproger.ru/translations/influxdb-guide/ (дата звернення:14.03.23)

10. Зберігання та обробка часових рядів у TimescaleDB [Електронний ресурс]. – Режим доступу: www. URL: https://eax.me/timescaledb/ (дата звернення:19.03.23)

11. Вибір серверного обладнання [Електронний ресурс]. – Режим доступу: www. URL: https://docs.openstack.org/arch-design/design-compute/design-compute-hardware.html (дата звернення:21.03.23)

12. NTP [Електронний ресурс]. – Режим доступу: www. URL: https://www.ntp-servers.net/ntp.html (дата звернення:22.03.23)

13. What Is MQTT [Електронний ресурс]. – Режим доступу: www. URL: http://surl.li/hsetl (дата звернення:24.03.23)

14. Протокол MQTT: концептуальне занурення [Електронний ресурс]. – Режим доступу: www. URL: https://habr.com/ru/articles/463669/ (дата звернення:27.03.23)

15. Basic Microcontroller Use for Measurement and Control [Електронний ресурс]. – Режим доступу: www. URL: http://surl.li/hsetz (дата звернення:30.03.23)

16. INTRODUCTION TO ESP8266 module [Електронний ресурс]. – Режим доступу: www. URL: https://microcontrollerslab.com/esp8266-getting-started-tutorial/ (дата звернення:02.04.23)

17. Introduction to BME280 [Електронний ресурс]. – Режим доступу: www. URL: https://www.theengineeringprojects.com/2019/04/introduction-to-bme280.html (дата звернення:07.03.23)

18. Introduction to DHT11 [Електронний ресурс]. – Режим доступу: www. URL: https://www.theengineeringprojects.com/2019/03/introduction-to-dht11.html (дата звернення:11.03.23)

19. Introduction to Arduino IDE [Електронний ресурс]. – Режим доступу: www. URL: https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html (дата звернення:16.04.23)

Додаток А. Вихідний код програми мікроконтролера

```
#if defined(ESP32)
#include <WiFiMulti.h>
WiFiMulti wifiMulti;
#define DEVICE "ESP32"
#elif defined(ESP8266)
#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti wifiMulti;
#define DEVICE "ESP8266"
#endif

#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>

#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <DHT.h>

#define DHTPIN D3
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
Adafruit_BME280 bme;
// WiFi AP SSID
#define WIFI_SSID "Xiaomi_AUF"
// WiFi password
#define WIFI_PASSWORD "12345678"

#define INFLUXDB_URL "http://192.168.31.211:8086"
#define INFLUXDB_TOKEN "CBparQ-
CW03vqMGHMslzm3GYVWq9cA8wf2DyqcMwQ8nrCCr2o7vgyal8hP3snF0aB8TnGbvpFAwTlwu2iwx1FQ=="
#define INFLUXDB_ORG "25ae8e593c424bf5"
#define INFLUXDB_BUCKET "Sensor_data"
```

```
// Time zone info
#define TZ_INFO "UTC3"

// Declare InfluxDB client instance with preconfigured InfluxCloud certificate
InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG, INFLUXDB_BUCKET, INFLUXDB_TOKEN,
InfluxDbCloud2CACert);

// Declare Data point
Point sensor("wifi_status");
Point sensor_data("sensor_data");

// MySQL connection settings
byte serverIP[] = {127, 0, 0, 1};
char user[] = "<mykyta>";
char password[] = "<12345>";
char db[] = "<sensordataL>"; // Replace with your database name

// MySQL objects
WiFiClient wifiClient;
MySQL_Connection conn(&wifiClient);
MySQL_Cursor cur(&conn);

void setup() {
  Serial.begin(115200);

  if (!bme.begin(0x76)) {  // Инициализируем BME280 с адресом 0x76
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }

  dht.begin();

  // Setup wifi
  WiFi.mode(WIFI_STA);
  wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);
```

```
  Serial.print("Connecting to wifi");
  while (wifiMulti.run() != WL_CONNECTED) {
    Serial.print(".");
    delay(100);
  }
  Serial.println();

  // Accurate time is necessary for certificate validation and writing in batches
  // We use the NTP servers in your area as provided by:
https://www.pool.ntp.org/zone/
  // Syncing progress and the time will be printed to Serial.
  timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");

  // Check server connection
  if (client.validateConnection()) {
    Serial.print("Connected to InfluxDB: ");
    Serial.println(client.getServerUrl());
  } else {
    Serial.print("InfluxDB connection failed: ");
    Serial.println(client.getLastErrorMessage());
  }

  sensor.addTag("device", DEVICE);
  sensor.addTag("SSID", WiFi.SSID());
  sensor_data.addTag("device", DEVICE);
  sensor_data.addTag("SSID", WiFi.SSID());
}

void loop() {
  // sensor.clearFields();
  // sensor_data.clearFields();

  float bmeTemperature = bme.readTemperature();
  float bmeHumidity = bme.readHumidity();
  float dhtTemperature = dht.readTemperature();
  float dhtHumidity = dht.readHumidity();
```

```
sensor_data.addField("bme_temperature", bmeTemperature);
sensor_data.addField("bme_humidity", bmeHumidity);
sensor_data.addField("dht_temperature", dhtTemperature);
sensor_data.addField("dht_humidity", dhtHumidity);

sensor.addField("rssi", WiFi.RSSI());

Serial.print("Температура BME280: ");
Serial.print(bmeTemperature);
Serial.println("°C");

Serial.print("Влажность BME280: ");
Serial.print(bmeHumidity);
Serial.println("%");

// Print what are we exactly writing
Serial.print("Writing: ");
Serial.println(sensor.toLineProtocol());

// Check WiFi connection and reconnect if needed
if (wifiMulti.run() != WL_CONNECTED) {
  Serial.println("Wifi connection lost");
}

// Write point
if (!client.writePoint(sensor)) {
  Serial.print("InfluxDB write failed: ");
  Serial.println(client.getLastErrorMessage());
}

if (!client.writePoint(sensor_data)) {
  Serial.print("InfluxDB(DATA) write failed: ");
  Serial.println(client.getLastErrorMessage());
}

// Save data to MySQL
if (conn.connect(serverIP, 3306, user, password)) {
```

```
    Serial.println("Connected to MySQL server");


    // Prepare INSERT statement
    char insertStmt[256];
    sprintf(insertStmt, "INSERT INTO %s.sensor_data (bme_temperature, bme_humidity, dht_temperature, dht_humidity, rssi) VALUES (%f, %f, %f, %f, %d)",
            db, bmeTemperature, bmeHumidity, dhtTemperature, dhtHumidity, WiFi.RSSI());


    // Execute INSERT statement
    cur.execute(insertStmt);


    // Close the connection
    conn.close();
  } else {
    Serial.println("MySQL connection failed");
  }


  Serial.println("Waiting 1 second");
  delay(1000);
}
```