МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри,
д-р техн. наук, проф.

_____ І. М. Журавська

« __ » _____ 202__ р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

КЛАСТЕРНА СИСТЕМА ДОСТАВКИ КОНТЕНТУ (CDN) НА БАЗІ RASPBERRY PI

Спеціальність «Комп'ютерна інженерія»
123 – КМР.1 – 605.21710513

Студент _____К.В.Кравцов
підпис

«__» _____ 202__ р.

Керівник канд.фіз-мат.наук, доцент _____С.В.Пузирьов
підпис
«__» _____ 202__ р.

Миколаїв – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CDN – Content Delivery Network
CPU – Central Processing Unit
COM – Computer On Module
DCT – Digital Component Technology
DSI – Display Serial Interface
eDP – Embedded Display Port
FLOPS – Floating Point Operations Per Second
GPIO – General-purpose input/output
HATS – Hardware Attached on Top
IoT – Internet of things
MIMO – Multiple Input Multiple Output
OS – Operating system
PCIe – Peripheral Component Interconnect Express
PIN – Personal Identification Number
POS – Point of sale
QoS – Quality of Service
RFQT – Really Fixed Query Tree
SoC – System-on-a-Chip
SPI – Serial Peripheral Interface
SSH – Secure Shell
SSL – Secure Sockets Layer
TCP – Transmission Control Protocol
UART – Universal Asynchronous Receiver-Transmitter
USB – Universal Serial Bus

ВСТУП

Останнє десятиліття відзначилось появою та масовим використанням пристроїв, робота яких полягає у зчитування даних з датчиків та надання інформації користувачеві. Різноманітні моніторингові пристрої використовуються передусім для відстеження показників, які тим чи іншим чином можуть вплинути на контроль безпекової ситуації у приміщенні. Виміри, отримані з датчиків, дозволяють оцінити безпекову обстановку та спланувати заходи щодо її покращення. Але абсолютна більшість наявних пристроїв напряму залежать від енергопостачання, й у разі зникнення енергії пристрій просто перестає фукціонувати, в деяких випадках вимагаючи додаткового переналаштування.

Через це проблема дієздатності вимірювальних приладів та можливість моніторингу зафіксованих даними приладами значень набула актуальності. Найкращим рішенням даної проблеми є створення автономної системи доставки контенту, незалежної від наявності енергопостачання, зібраної з дешевих та доступних компонентів.

Актуальність теми полягає у потребі резервування мережевих вузлів доставки контенту в умовах обмеженного енергопостачання та перевантаження ліній зв'язку.

Метою кваліфікаційної роботи є створення кластерної системи доставки контенту на базі одноплатного комп’ютера Raspberry Pi

Об’єктом дослідження є процес доставки контенту у кластерних системах

Предмет дослідження: кластерна система доставки контенту (CDN) на базі Raspberry Pi.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

— Ознайомитися з теоретичною базою пов’язаною безпосередньо з темою використання одноплатних комп’ютерів..

— Дослідити основні методи створення систем доставки контенту.

— Проаналізувати шляхи створення пристрою врахувавши недоліки попередніх/наявних розробок.

Методи дослідження. У дипломній роботі для вирішення наукових завдань використані наступні методи досліджень:

дедукції – при виборі напрямків дипломного дослідження;

експертної оцінки – при виборі теми і постановці мети дослідження та аналізі результатів;

системного аналізу – при розробці технології наукових досліджень;

теорії нечіткої логіки – при розробці розпізнавання людини по «нечітким»;

- порівняння (емпіричний) – при виборі способів розпізнавання обличчя людини на фотографіях та відео фрагментах;
- пояснення(проміжний) – при формуванні і огляді концепції дипломної роботи;
- аналіз проблеми(проміжний) – при вирішенні проблем під час написання програмної частини системи;
- методи системного та порівняльного аналізу(теоретичний) – під час аналізу наукових джерел та статей для кращого розуміння методів розпізнавання образів.

Апробація результатів магістерської роботи відбулася під час:

- Всеукраїнська науково-практична конференція молодих вчених, аспірантів і студентів (7-10 лютого).

Публікації. За результатами магістерської роботи опубліковані тези доповідей [1].

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ СИСТЕМ ДОСТАВКИ КОНТЕНТУ

Системи доставки контенту являють собою розподілену мережу серверів, які займаються доставкою інформації до споживачів.

CDN дозволяють збалансувати тиск на мережевий трафік, гарантуючи відсутність перевантаження на окремі сервери. У випадку зупинки роботи принаймні одного сервера, CDN може ініціювати процес “failover”, який дозволяє залучити запасний сервер. Основною перевагою CDN є стійкість відносно потенційних атак. Зазвичай мережа доставки контенту складається з кластерів, завданням яких є передача, запис та зберігання окремих типів даних, та центрального хабу, який займається координацією та зв’язком між мережевими вузлами. У разі виходу зі строю окремого елементу мережа не передає працювати, а продовжує передавати дані у звичному режимі, доки вийшовший з ладу компонент не відновить роботу.

До таких систем доставки контенту можна віднести окремі різновиди сенсорних мереж. Сенсорна мережа – кластер, який складається з пристроїв, основним завданням яких є моніторинг даних.

1.1 Огляд сенсорних мереж

Прикладом сенсорних мереж є пристрої моніторингу показників чистоти повітря: різноманітні датчики температури, вологості, загазованості повітря, тощо. Ці системи зазвичай інтегруються в будинок та дозволяють користувачам отримувати своєчасний доступ до інформації, пов’язаною з безпековою ситуацією в приміщенні. Серед переваг таких систем потрібно виділити декілька:

— Зручність у налаштуванні за рахунок автоматизації процесів

— Наявність власного ПЗ , яке спрощує взаємодію з пристроєм

Пристрої відрізняються за ціною , функціоналом, та наявністю на українському ринку. Серед найбільш популярних на ринку варто виділити декілька:

— AC90

— AirVision-Z-Gas

— Tervix Pro Line ZigBee GAS Sensor

1.1.1 AC90

Датчик газу AC90 — це тип електрохімічного датчика газу, який використовується для виявлення та вимірювання концентрації різноманітних газів. Він широко використовується в промислових цілях, таких як хімічна та нафтохімічна промисловість, а також в автомобільній промисловості.

Датчик газу AC90 виробляється японською компанією Figaro Engineering, яка спеціалізується на розробці та виробництві датчиків газу. Датчик призначений для виявлення широкого діапазону газів, включаючи чадний газ (CO), діоксид азоту (NO2), сірководень (H2S) і різні вуглеводні.

Датчик працює за допомогою робочого електрода, протиелектрода та електрода порівняння для вимірювання концентрації виявленого газу. Коли газ контактує з робочим електродом, відбувається хімічна реакція, яка генерує електричний сигнал. Потім цей сигнал посилюється та обробляється для вимірювання концентрації газу.

Однією з ключових переваг датчика газу AC90 є його висока чутливість, яка дозволяє виявляти дуже низькі рівні газу. Він також є високоселективним, тобто може розрізняти різні гази, навіть якщо вони присутні в одному середовищі.

Окрім чутливості та селективності, датчик газу AC90 також є високонадійним і довговічним, що робить його добре придатним для використання в суворих промислових умовах. Однак, як і всі датчики газу, він потребує регулярного калібрування та обслуговування, щоб забезпечити точні показання з часом.Датчик зображено на рис.1.1.

Рисунок 1.1 – Датчик AC90

1.1.2 AirVision-Z-Gas

При виявленні у приміщенні підвищеного вмісту природного газу пристрій через вбудований динамік вмикає сирену (рівень звуку 85DB/3M), а також передає сигнал тривоги та виміряне значення концентрації природного газу в систему керування на мобільному телефоні для сповіщення користувача. Для зв'язку використовує модуль ZigBee. Для роботи пристрою обов'язково потрібний Контролер ZigBee Gateway. AirVision-Z-Gas підключається до платформи Maxus SMART, що об'єднує усі пристрої розумного будинку в єдину мережу з центром управління смартфоном. Джерело живлення пристрою – електромережа. Пристрій показано на рис. 1.2.

Рисунок 1.2 – Датчик загазованості повітря AirVision Z-Gas

1.1.3 Tervix Pro Line ZigBee GAS Sensor

Tervix Pro Line ZigBee GAS Sensor – це тип газового датчика, призначеного для виявлення та вимірювання концентрації різних газів у повітрі, включаючи природний газ, зріджений газ і чадний газ. Він призначений для використання в системах автоматизації розумного будинку або будівель, які використовують протокол бездротового зв’язку ZigBee.

Tervix Pro Line ZigBee GAS Sensor використовує вдосконалені електрохімічні датчики для виявлення присутності газу, і він оснащений вбудованою сигналізацією, яка звучить у разі виявлення газу. Він також має світлодіодний індикатор, який забезпечує візуальну індикацію рівнів концентрації газу.

Однією з ключових особливостей датчика газу Tervix Pro Line ZigBee є його сумісність із бездротовими мережами ZigBee. Це дозволяє легко інтегрувати датчик в існуючі системи домашньої автоматизації, які використовують ZigBee, надаючи користувачам простий і зручний спосіб контролювати якість повітря та безпеку своїх будинків.

Датчик зазвичай живиться від батарейок, які можуть працювати до 2 років залежно від використання. Він також призначений для легкого монтажу на стінах або стелі, і його можна запрограмувати та налаштувати за допомогою програми для смартфона або шлюзу ZigBee.

Загалом, датчик газу Tervix Pro Line ZigBee є надійним і простим у використанні датчиком газу, який може допомогти підвищити безпеку та комфорт у вашому домі чи будівлі, виявляючи наявність шкідливих газів у повітрі. Пристрій зображено на рис. 1.3.

Рисунок 1.3 – Tervix Pro Line Sensor

Причина популярності вищезгаданих пристроїв – відносно невелика ціна та простота налаштування. Пристрій просто вмикається у розетку та одразу починає свою роботу, в більшості випадків вимагаючи мінімальних налаштувань.

Основними недоліками доступних систем є неможливість об’єднання їх у кластери задля бачення більш конкретної та актуальної безпекової ситуації. Частіше всього пристрої просто не здатні взаємодіяти між собою через несумісність ПЗ, компонентів та різні методи доставки інформації до користувача (звукові сигнали у разі використання в приміщенні, сповіщення у мобільному застосунку, тощо). Також ці пристрої мають слабкий рівень кастомізації, що унеможливлює додавання додаткових параметрів для відстеження , не вдаючись до застосування принципово інших систем. Вагомим недоліком є і енергозалежність даних систем. У разі зникнення енергопостачання система припиняє свою роботу, втрачаючи дані, записані перед перервою в роботі.

Рішенням проблеми перебоїв в електропостачанні може стати придбання джерела бесперебійного живлення, проте дуже часто ця річ є не дуже

доступною через високу вартість як самого пристрою, так і підтримки його належного функціонування.

Останнім часом дуже широкої популярності набули пристрої “Інтернету Речей”. Ці пристрої відомі завдяки широким можливостям налаштування, гнучким функціоналам та можливістю створення кластерів. За допомогою компонентів IoT , доступним на українському ринку , можна створити власний кластер , який займається збором даних з двох та більше датчиків, та відправляє їх на сервер, відображаючи їх у зручному для користувача вигляді. При створенні таких системи основним питанням є спосіб передачі даних між пристроями.

1.2 Протоколи передачі даних

В основі взаємодії пристроїв Інтернету Речей лежать протоколи передачі даних. В загальному плані вони поділяються в залежності від ділянки мережі, на якій вони використовуються. В рамках концепції IoT існують наступні ділянки:

— Сенсорний вузол – сенсорний вузол (DDS),

— Сенсорний вузол – сервер (CoAP, MQTT, XMPP,STOMP),

— Сервер – сервер (AMQP)

1.2.1 XMPP

XMPP (Extensible Messaging and Presence Protocol) — розширюваний протокол обміну повідомленнями та інформацією про присутність, набір відкритих технологій для обміну миттєвими повідомленнями, багатостороннього чату, голосових і відеодзвінків, співпраці, полегшеного проміжного програмного забезпечення, синдикації вмісту та узагальненої маршрутизації даних.

У протоколі XMPP використовується текстовий формат XML. Як транспорт використовується протокол TCP. XMPP підтримує різні комунікаційні моделі (запит-відповідь, публікація-підписка та інші).

Адресація XMPP особливо зручна у випадках, коли дані передаються між віддаленими, найчастіше незалежними точками, наприклад, у разі взаємодії двох абонентів. За допомогою XMPP, наприклад, можливе підключення домашнього термостата до Web-сервера для отримання доступу до нього з телефону. Сильними сторонами цього протоколу є також безпека та масштабованість, що робить його ідеальним для додатків Інтернету речей із орієнтацією на споживача.

1.2.2 DDS

Протокол DDS (Data Disribution Service) розподіляє дані між пристроями. DDS реалізує прямий шинний зв'язок між пристроями з урахуванням реляційної моделі даних. Протокол DDS реалізує багатоадресну систему, використовуючи UDP. Цей протокол орієнтований на шаблон "видавець-передплатник", при цьому передача повідомлень здійснюється по шині з використанням методу "запит-відповідь". Операції, що виконуються протоколом, задаються тринадцятьма класами (Entity Class, WaitSet Class, Condition Class, Publisher Class, DataWriter Class, Subscriber Class, DataReader Class, ReadCondition Class, QueryCondition Class та іншими). Протокол DDS реалізує дві операції – читання та записи, використовуючи відповідні класи.

Операція читання (Read) здійснюється на всіх доступних пристроях. Дані не видаляються з локального кеша DDS в результаті цієї операції і можуть бути прочитані знову за вказівкою спеціальних параметрів. Отримання даних здійснюється трьома способами.

- **Опитування (Polling)** – програма (зазвичай періодично) запитує DDS для отримання нових даних або інформування про зміну стану. Інтервал опитування залежить від програми та даних.
- **Списки очікування (WaitSets)** – додаток реєструє в DDS списки очікування та чекає, поки одна з переданих подій не відбудеться.

— **Слухачі (Listeners)** – додаток реєструє в DDS (у класах, де події описані) спеціальні класи-слухачі, які будуть інформовані під час цих подій.

Основне призначення протоколу DDS – комунікація між сенсорними вузлами/датчиками, об'єднання їх прямим шинним зв'язком та забезпечення багатоадресної системи. Загалом DDS застосовується в автоматизованих системах управління рухом, а також в енергетиці, військовій та фінансовій сфері.

1.2.3 AMQP

AMQP (Advanced Message Queuing Protocol) — відкритий протокол передачі повідомлень між компонентами системи. Принцип роботи полягає в тому, що окремі підсистеми (або незалежні програми) можуть обмінюватися довільним чином повідомленнями через AMQP-брокер, який здійснює маршрутизацію, можливо, гарантує доставку, розподіл потоків даних, підписку на потрібні типи повідомлень.

AMQP протокол простий у реалізації та побудований на давно відомих технологіях текстового зв'язку між учасниками мережі. Тільки комунікація відбувається безпосередньо між пристроями, і з використанням третього вузла — брокера (чи AMQP-сервера). Така структура дає можливість уникнути втрати переданих даних, якщо вузол (AMQP-клієнт) вимкнено. Надіслані повідомлення все одно чекатимуть отримувача на вузловій станції.

Власне процедура виглядає так: видавець відправляє дані на вузол AMQP. Там повідомлення проходять сортування та розміщуються в чергу. Отримувач з'єднується з брокером і завантажує інформацію тільки з потоків, що запитуються, причому ключовий вузол віддає всі повідомлення лінії, а не конкретного творця.

Визначення, прийняті для мережі AMQP:

— **Consumer** - одержувач повідомлення;

— **Producer** - творець публікації;

— **Routing key** – унікальний ключ черги, в яку потрібно помістити повідомлення;

— **Queue** - черга, що зберігає саме повідомлення

— **Exchange** - так називається точка обміну, яка здійснює розподіл даних по чергах;

— **Message** - власне передані дані.

Відмінності протоколу AMQP від подібних мереж — використання синтаксису XML при формуванні пакетів повідомлень. Такий вид даних дозволяє відправляти і приймати в них не тільки текстову інформацію, а й бінарну (з кодуванням за алгоритмом base64). Крім того, сама структура мережі – асинхронна, немає необхідності бути онлайн постійно тому, хто надсилає дані, або тому, хто їх отримує. Головне, щоб вузол-брокер був у мережі та готовий обмінюватися інформацією.

Щодо точок обміну, то сортування в них використовуються чотирьох видів:

— **Topic** - використовує дані про ключ черги.

— **Fanout** — копіює повідомлення у всі створені черги незалежно від зазначеної інформації.

— **Direct** — циклічно розподіляє інформацію щодо розширених відомостей у ключах маршрутизації. Допомагає розвантажити черги.

— **Headers** - розширений варіант попереднього, для руху цифрових потоків по чергах використовує відомості про додаткові атрибути xml-структури пакета.

1.2.4 MQTT

Протокол MQTT (Message Queuing Telemetry Transport) був створений докторами Енді Стенфорд-Кларком і Арленом Ніппером у 1999 році. Причина створення нового методу зв'язку полягала у потребі якісного та своєчасного відстеження даних , зібраними пристроями моніторингу, задіяними у нафтогазовій промисловості, та їх відправлення на віддалений сервер. Зазвичай ці пристрої використовувалися у віддалених місцях, де будь-яке стаціонарне,

дротове з’єднання чи радіозв’язок було б важко або неможливо встановити. У той час єдиним варіантом для таких випадків був супутниковий зв’язок, який був дуже дорогим і оплачувався залежно від обсягу використаних даних. З тисячами датчиків у польових умовах промисловість потребувала форми зв’язку, яка могла б надавати достатньо надійні дані для використання, використовуючи при цьому мінімальну пропускну здатність.

Message Queuing Telemetry Transport (MQTT) входить до 10 найбільш використовуваних протоколів в галузі Інтернету речей. Оскільки протокол почав використовувватися більше двох десятиліть тому, він має за собою величезну базу активних користувачів, інструкцій та допоміжних посібників. Навіть сьогодні він отримує значну увагу завдяки своїй легкості, відкритості та доступності. Пристрої IoT можуть комунікувати за його допомогою, незважаючи на нестабільні умови мережі.

На прикладному рівні протокол працює поверх TCP/IP і може легко протоколювати віддалені об'єкти безпосередньо через Інтернет, без необхідності використання VPN-тоннелів. Достатньо, щоб брокер мав реальну IP-адресу і всі клієнти могли до нього підключитися. При цьому клієнти можуть знаходитись за NAT. Так як підключення до протоколу MQTT ініціюють клієнти, перевірка портів для встановлення з’єднань не потрібна, у цей час підключення до Modbus/TCP ініціює сервер (головний), який вимагає прямої мережевої доступності.

Стандартний порт MQTT-брокера для вхідних TCP-з'єднань — 1883. При використанні захищеного SSL-підключення використовується порт 8883.

MQTT включає в себе взаємодію наступних сутностей:

— **Publisher(Видавець):** пристрій в ролі видавця надсилає підписникам дані через канал.

— **Topic(топік\канал):** кожен ресурс має власний унікальний ідентифікатор. Видавець надсилає інформацію до каналу, потім ці дані отримує підписник.

— **Subscriber(Підписник):** підписником є пристрій , який отримує інформацію від видавця.

— **Брокер**: центральний хаб, відповідальний за комунікацію між видавцями да підписниками. У якості брокера може виступати серверне ПО або контролер.

Протокол дуже широко використовується при створенні IoT-пристроїв, які орієнтовані на помірне або ж низьке енергоспоживання. Протокол використовується для реалізації широкого спектра завдань, а саме:

— **Ефективна комунікація.** Низьке споживання ресурсів робить протокол ідеальним для пристроїі швидкої передачі повідомлень у реальному часі

— **Безпека.** При застосуванні в пристроях, інтегрованих у «Розумний дім», функція QoS дозволяє перевірити, чи вчасно важлива інформація дійшла до користувачів. У разі виявлення небезпеки , користувачі матимуть змогу своєчасно вжити заходів, спрямованих на забезпечення власної безпеки.

— **Збір інформації.** MQTT дозволяє організаціям та приватним особам вчасно збирати дані з великої кількості джерел, таких як сенсори та смартфони.

— **Відстеження стану здоров'я людини.** Пов'язані між собою сенсори здатні моніторити стан організму людини після її виписки з медичного закладу.

— **Зв'язок між транспортними засобами.** На відміну від HTTP, MQTT здатний на постійний зв`язок між кліентом та брокером. Особливо це корисно у разі відстеження транспорнтних засобів. Коли засіб рухається через зону з відсутнім стільниковим зв'язком, сессія продовжиться з поверненням з'єднання. Для складного HTTP-рукостискання немає потреби.

1.2.4.1 Quality of Service (QoS)

Рівень якості обслуговування (Quality of Service) є ключовою функцією протокола MQTT. Це так-звана угода між відправником та одержувачем

інформації. Вона визначає, наскільки впевнено система може доставити певне повідомлення.

QoS надає клієнтові можливість самостійно обирати рівень обслуговування, який відповідатиме надійності його мережі та логіці програми. Через те, що MQTT займається повторною передачею повідомлень та гарантує безпеку данного процесу (навіть якщо передача є ненадійною) , QoS значно збільшує якість безпеки комунікації у нестабільних мережах.

Quality of Service поділяється на три рівні:

QoS 0 – At most once. Мінімальний рівень. Він гарантує високу швидкість роботи, при цьому не даючи ніяких гарантій доставки. Отримувач не підтверджує доставку, саме повідомлення не пересилається та не зберігається.

Рисунок 1.4 – Рівень QoS 0.

QoS 1 – At least once. Даний рівень гарантує доставку повідомлення принаймні один раз. Відправник зберігає повідомлення, доки не отримує від отримувача пакет PUBACK, який підтверджує отримання повідомлення. Можливе також відправлення та отримання повідомлень декілька разів.

Відправник використовує ідентифікатор пакетів для співставлення пакету PUBLISH до PUBACK. Якщо відправник не отримує пакет PUBACK протягом заданого проміжку часу, він повторно надсилає пакет PUBLISH. Коли відправник отримує повідомлення з QoS 1, він має змогу негайно його обробити. У випадку, коли отримувач є брокером, брокер відправляє повідомлення всім клієнтам-підписникам, а потім відповідає відправленням пакету PUBACK.

Якщо клієнт-видавець пересилає повідомлення, воно відмічається як дуплікат (DUP). У QoS 1, відмітка використовується для внутрішніх процесів та не

оброблюється ні брокером, ні клієнтами. Отримувач повідомлення відправляє PUBACK незалежно від наявності DUP.

Рисунок 1.5 – Рівень QoS 1.

QoS 2 – Exactly once. Найвищий рівень сервісу в MQTT. Рівень гарантує, що кожне відправлене повідомлення доходить до всіх потенційних отримувачів один раз. QoS 2 є найповільнішим та водночас найбезпечнішим рівнем обслуговування. Гарантія заснована на принаймні двох потоках request/response між відправником та отримувачем. Отримувач та відправник використовують ідентиффікатор пакетів оригінального повідомлення PUBLISH задля координації доставки повідомлення.

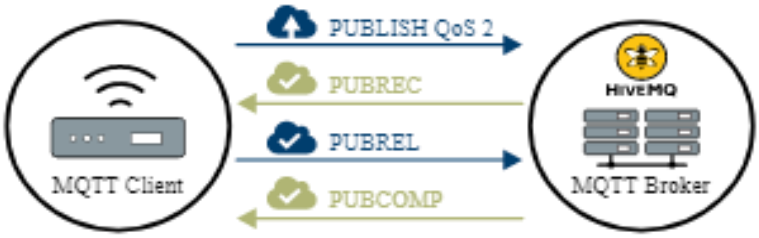


Рисунок 1.6 – Рівень QoS 2

Коли отримувач отримує пакет QoS 2 PUBLISH від відправника, він відповідним чином обробляє повідомлення публікації та відповідає відправникові пакетом PUBREC, який підтверджує пакет PUBLISH. Якщо відправник не отримує пакет PUBREC від одержувача, він знову надсилає пакет PUBLISH із позначкою дублікату (DUP), доки не отримає підтвердження.

Як тільки відправник отримає пакет PUBREC від одержувача, відправник може безпечно відкинути початковий пакет PUBLISH. Відправник зберігає пакет PUBREC від одержувача та відповідає пакетом PUBREL.

Після того, як отримувач отримає пакет PUBREL, він може відкинути всі збережені стани та відповісти пакетом PUBCOMP (те саме стосується випадків, у яких відправник отримує PUBCOMP). Поки отримувач не завершить обробку та не надішле пакет PUBCOMP назад відправнику, отримувач зберігає посилання на ідентифікатор вихідного пакета PUBLISH. Цей крок важливий, щоб уникнути повторної обробки повідомлення. Після того, як відправник отримує пакет PUBCOMP, ідентифікатор пакета опублікованого повідомлення стає доступним для повторного використання.

Коли потік QoS 2 завершено, обидві сторони впевнені, що повідомлення доставлено, і відправник має підтвердження доставки.

Якщо під час доставки пакет буде втрачено, відправник несе відповідальність за повторну передачу повідомлення протягом невеликого періоду часу. Це так само розповсюджується на випадки, у яких відправником є клієнт або брокер MQTT. Одержувач несе відповідальність відповідним чином відповісти на кожне командне повідомлення.

1.3 Огляд патентів

1.3.1. Патент №CN110807905B. Система моніторингу протипожежної безпеки на базі end-edge-cloud архітектури

Основа винаходу полягає у системі моніторингу протипожежної безпеки, в яку входить мережа 6LoWPAN, хмарне сховище та хмарна платформа. Сенсорна мережа 6LoWPAN має вбудовану систему моніторингу рівня протипожежної безпеки, мережеве сховище з’єднане з мережею та отримує доступ до платформи через мережу NB-IOT, дані передаються через протокол MQTT. Хмарна платформа використовує систему контролю пожежної небезпеки, побудовану на базі нечіткої нейронної мережі задля вчасного інформування одразу декількох груп користувачів.

Винахід є доступним, недорогим та передусім – енергонезалежним. Завдяки передачі даних за протоколом MQTT підвищена надійність передчачі даних між пристроями та сервером зберігання даних.

1.3.2 Патент № US10075353B2. Система управління сенсорною мережею

Винахід являє собою систему контролю звітів та систему їх обробки. Система обробки налаштована на отримання множини звітів від множини інших пристроїв, фільтрування одного або більше з множини звітів і пересилання решти множини звітів до іншого пристрою, при цьому кожен із множини звітів містить дані на основі на вимірювання датчика у відповідному кластері, який складається з деякої кількості інших пристроїв.

Спосіб обробки звітів полягає в прийомі множини звітів від множини інших пристроїв, фільтрацію одного або більше з множини звітів і пересилання решти множини звітів до іншого пристрою, причому кожен із множини звітів містить дані на основі даних, відстеженим окремим сенсорним кластером.

Не менш важливим функціоналом винаходу є й фільтрація отриманих звітів. Так, звіти фільтруються за наступними критеріями:

— Тип даних, що вимірюється

— Рівень пріорітету

— Географічне знаходження сенсорних кластерів

— Попередні виміри, які зберігаються в кешованому вигляді

1.3.3 Патент № US9800683B2. Керування пропускною здатністю в мережі доставки контенту з самостійним керуванням.

Суть винаходу в цілому спрямована на управління політикою пропускної здатності на основі схеми управління політикою та визначення популярності, у якій контент може бути попередньо визначений як популярний за місцем розташування та демографічними характеристиками з використанням кількох джерел. Принаймні деякі з варіантів використання винаходу можуть вигідно оптимізувати потоки трафіку CDN до вузлів, які страждають від перевантаження, шляхом попереднього надання вмісту на основі визначень популярності та розподілу. Завдяки попередньому забезпеченню адаптивних потокових пакетів або вмісту, особливо під час початкового кешування, оператор може заощадити значну кількість пропускної здатності розподілу

CDN, оскільки запити користувачів на вміст обслуговуються та/або перенаправляються CDN. У деяких альтернативних варіантах здійснення керування політикою для надсилання вмісту та отримання вмісту може бути застосовано в усій ієрархічній організації CDN, тим самим потенційно реалізуючи кращу якість обслуговування (QoS) і розподіл пропускної здатності мережі для отриманого вмісту. Крім того, один або більше додаткових варіантів здійснення цього розкриття можуть бути налаштовані для сприяння оптимізації каналів push і pull CDN на основі поточного розподілу та використання мережі, а також очікуваного майбутнього розподілу та/або використання вмісту.

Висновки до розділу 1

У першому розділі було проведено аналіз предметної області , пов’язаної з системами доставки контенту та передачею даних в цілому. Проаналізовано протоколи передачі даних XMPP, DDS, AMQP та MQTT. також рогзлянуто патенти, пов’язані з предметною областю.

Враховуючи проведені дослідження, можна зробити наступні висновки:

- Передачу даних доцільніше проводити за протоколом MQTT, оскільки він є найкращим для роботи з датчиками
- Апаратне забезпечення повинне бути недорогим та доступним на українському ринку
- Апаратний комплекс повинен бути зручним у використанні та незалежним від джерел живлень, що знаходяться в приміщенні

РОЗДІЛ 2
РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ КЛАСТЕРНОЇ СИСТЕМИ ДОСТАВКИ КОНТЕНТУ НА БАЗІ RASPBERRY PI

2.1 Вибір апаратного забезпечення та датчиків

Завдання створення кластерної системи доставки контенту вимагає збору точних даних, їх впорядковування та представлення користувачеві.
Передача даних між пристроями буде відбуватись за протоколом MQTT (рис.2.1).

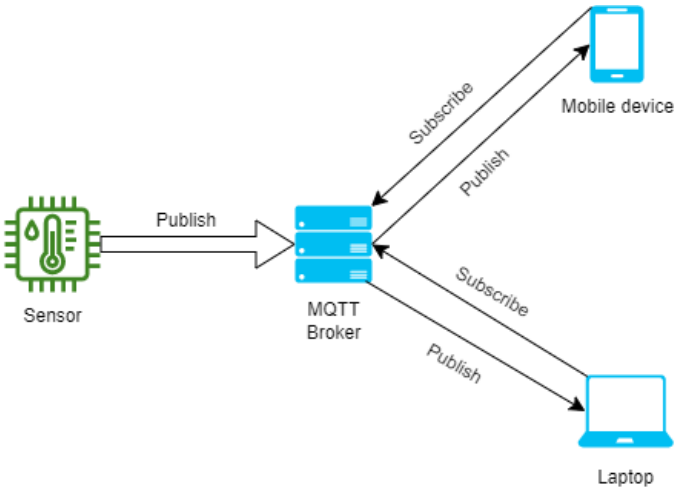


Рисунок 2.1 – Приблизна схема комунікації компонентів апаратно-програмного комплексу за допомогою MQTT

Найголовнішим елементом комунікації між сенсорами збору даних та користувачами є брокер MQTT. Для цієї ролі підійде модель одноплатного комп’ютера.

2.1.1 Raspberry Pi

Raspberry Pi — це недорогий одноплатний комп’ютер (SBC), створений благодійною організацією Raspberry Pi Foundation, що базується в Британії. З моменту першого випуску в 2012 році було випущено кілька поколінь мікрокомп’ютерів Raspberry Pi, які можна розділити на три окремі моделі:

Raspberry Pi A, B і Zero . Основи цих трьох моделей дуже схожі, кожна з яких містить систему на чіпі, яка складається з інтегрованого центрального процесора (CPU) і графічного процесора (on-a-chip). графічний процесор на мікросхемі), вбудована пам’ять і вхідна напруга 5 В постійного струму.

Усі моделі також мають порт для підключення спеціальної камери, а також набір контактів вводу/виводу загального призначення (GPIO), які можна використовувати для зв’язку з широким спектром електроніки, від світлодіодів і кнопок до сервоприводів і двигунів. , силові реле та величезний асортимент датчиків.

Спеціальні плати розширення HATS, які підключаються до контактів GPIO, можуть надавати додаткові функції, починаючи від керування живленням, виявлення RFID, контролерів двигунів і високоякісного аудіозапису. Більшість моделей також мають підключення Ethernet і бездротове підключення (Wi-Fi і Bluetooth), що в поєднанні з портами GPIO надає Raspberry Pi величезної універсальності.

Raspberry Pi має всі функції стандартного комп’ютера. Таким чином, ви можете підключити мишу, клавіатуру та екран комп’ютера без будь-якої конфігурації та мати контроль над простим у використанні середовищем робочого столу Linux. Крім Raspberry Pi OS, яка підтримується Raspberry Pi Foundation, можна встановити багато інших операційних систем, включаючи Windows 10 IoT і Android. Raspberry Pi також можна використовувати як блок без голови з дистанційним керуванням і запрограмувати на автономне виконання сценаріїв з використанням широкого діапазону комп’ютерних мов.

Raspberry Pi відрізняється від мікроконтролерів, таких як популярний Arduino або нещодавно випущений Raspberry Pi Pico, які є набагато більш простими комп’ютерами з дуже низькими потребами в енергії, які просто виконують одну програму, написану користувачем.

Raspberry Pi — чудовий дослідницький інструмент, який можна використовувати майже для чого завгодно, від простого моніторингу навколишнього середовища та запису лабораторних експериментів до

автономних польових вимірювальних станцій і вдосконалених пристроїв із замкнутим циклом, які можуть зчитувати різні модулі введення, запускати інші дії (наприклад, для вмикати та вимикати світло або сервоприводи), а також автоматично обробляти дані та надсилати попереджувальні повідомлення.

Серед переваг мікрокомп’ютера можна виділити наступні:

- Велика потужність обробки інформації в дуже компактній платі
- Велика кількість виділених інтерфейсів (UART, I2C, SPI, I2S, CSI, DSI) для підключення широкого діапазону датчиків і електричних компонентів
- Високі можливості підключення (HDMI, USB, Ethernet, Wi-Fi, Bluetooth)
- Доступність для широкого кола користувачів за рахунок невеликої вартості
- Простота у використанні, зумовлена великою користувацькою базою, кількістю посібників та гайдів для нових користувачів
- Початок роботи з платою не вимагає від користувача додаткових навичків програмування; простому процесу навчання сприяє достатня кількість спільнот розробників.
- Невеликий форм-фактор корпуса, який дозволяє створення компактних пристроїв на базі плати.
- Вбудована графіка з підтримкою HDMI (до 4K)
- Низьке енергоспоживання (але вище, ніж у мікроконтролерів) і може живитися від широкого спектру зовнішніх батарей і сонячних панелей
- Не містить рухомих частин, практично не генерує шум
- Висока можливість налаштування та гнучкість порівняно з комерційними рішеннями
- Висока ступінь зворотної сумісності

2.1.2 NodeMCU V3

WiFi-модуль ESP8266 - високоінтегрована мікросхема, випущена китайською компанією Espressif Systems в 2014 г. Дана плата пропонує повністю готове рішення для мереж WiFi, дозволяє забезпечувати доступ до мережі як в якості клієнта, так і точки доступа. Існує два сценарії використання ESP8266:

1.Модуль підключається до готового пристрою на базі мікроконтроллера і виконує функцію моста UART-WiFi. Такий варіант використання підійде для розширення функціоналу готових проектів і управління ними через Інтернет;

2.Реалізація нового пристрою, в якому в якості керуючого мікроконтролера використовується сам чіп ESP8266. Це дозволяє створювати не-великі пристрої, які здатні тривалий час працювати від акумуляторів-ром або батарей. Існує безліч модифікацій модуля ESP8266, основні із них представлені нижче на рисунку 1.2.

Рисунок 2.2 – Модифікації модуля ESP8266

Загальні технічні характеристики сімейства модулів ESP8266 наведено в таблиці 2.1:

Таблиця 2.1 – Технічні характеристики сімейства модулів ESP8266

Підтримувані режими Wi-Fi	Точка доступу, клієнт
Вхідна напруга	3,7В – 20 В
Робоча напруга	3В- 3,6В
Максимальний струм	220мА
Діапазон робочих температур	від -40℃ до 125℃

Продовження таблиці 2.1.

Час запуску та відправки пакетів	22 мС
Особливості	Тактова частота 80 МГц, 32-бітний процесор Наявні вбудовані перемикачі TR та PLL; Наявні підсилювачі потужності, регулятори, системи управління живленням; Вбудований стек TCP/IP; Підтримує Wi-Fi протокол 802.11 b / g / n;

Існує кілька поколінь плат NodeMcu - V1 (версія 0.9), V2 (версія 1.0) і V3 (версія 1.0). Позначення V1, V2, V3 використовуються при продажу в інтернет-магазинах. Нерідко відбувається плутанина в платах - наприклад, V3 зовні ідентична V2. Також всі плати працюють за принципом open-source, тому їх можуть виробляти будь-які фірми. Але в даний час виробництвом плат NodeMcu займаються Amica, DOIT і LoLin / Wemos.

Плати покоління V1 і V2 легко відрізнити - вони мають різний розмір. Також друге покоління оснащено поліпшеною модифікацією чіпа ESP-12 і 4 Мб флеш-пам'яті. Перша версія, застаріла, виконана у вигляді яскравої жовтої платформи. Використовувати її незручно, так як вона покриває собою 10

виходів макетної плати. Плата другого покоління зроблена з виправленням цього недоліку - вона стала більш вузькою, виходи добре підходять до контактів плати. Плати V3 зовні нічим не відрізняються від V2, вони мають більш надійним USB-виходом. Випускає плату V3 фірма LoLin, з відмінностей від попередньої плати можна відзначити те, що один з двох зарезервованих виходів використовується для додаткової землі, а другий - для подачі USB живлення . Також плата відрізняється більшим розміром, ніж попередні види.

2.1.3 Wemos D1 Mini

WeMos D1 mini побудована на базі 32-розрядного мікроконтролера ESP8266 (він входить до складу ESP12-E встановленої на платі) з інтегрованим WiFi модулем (802.11 b/g/n 2.4 ГГц). Також на платі присутні стабілізатор напруги на 3,3 В, роз'єм USB типу Micro-B і USB-UART перетворювач на базі чіпа CH340G. Мікроконтролер ESP8266 працює на тактовій частоті 80 МГц і має оперативну пам'ять RAM даних на 80 КБ (для зберігання значень змінних), і пам'ять RAM інструкцій на 32 КБ. Програми зберігаються у flash пам'яті об'ємом 4 МБ. Модуль зображено на рис. 2.3

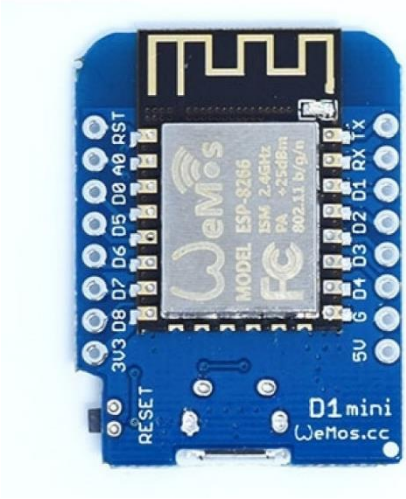


Рисунок 2.3. Wemos D1 Mini

Наступною версією плати WEMOS D1 mini є плата WEMOS D1 mini Pro, яка має 16 МБ flash пам'яті, встановлена керамічна SMD антена, є роз'єм IPX для підключення зовнішньої антени і використовується чіп USB-UART

перетворювача CP2104. Габарити цих плат, розташування виходів та їхнє призначення ідентичні.

Усі цифрові виходи крім D0 можна використовувати для роботи із зовнішніми перериваннями, ШІМ, шиною I2C або 1-wire (виходи шини I2C за замовчуванням D1 та D2, але їх можна перепризначити). Логічні рівні всіх цифрових виходів 3,3 У. На аналоговий вхід A0 можна подавати напругу до 3,2 У. Виходи D3, D4 і D8 підтягнуті до 3V3 через резистори 10 кОм (це пов'язано з особливістю завантаження скетчів у плату).

Характеристики:

— Мікроконтролер: ESP8266.

— Розрядність: 32 біти.

— Напруга живлення плати: 3,3/5,0 В.

— Бездротовий інтерфейс: Wi-Fi 802.11 b/g/n 2,4 ГГц (STA/AP/STA+AP, WEP/TKIP/AES, WPA/WPA2).

— Шини, що підтримуються: SPI, I2C, I2S, 1-wire, UART, UART1, IR Remote Control.

— Цифрові виходи I/O: 11 (RX, TX, D0...D8) ; всі виходи крім D0 підтримують INT (зовнішнє переривання), ШІМ, I2C, 1-wire.

— Аналогові входи: 1 (A0) 10-біт АЦП.

— Логічні рівні виходів: I/O: 3,3 В

— Максимальний струм на виході I/O: 12 мА (кожний вихід).

— Максимальна напруга на вході A0: 3,2 В (між виходами A0 та GND)

— Flash-пам'ять: 4 МБ.

— RAM-пам'ять даних: 80 КБ.

— RAM-пам'ять інструкцій: 32 КБ.

— Тактова частота мікроконтролера: 80 МГц.

— Чіп USB-UART перетворювача: CH340G.

— Діапазон робочих температур: -40...+85°C.

2.1.4 Датчик вологості , тиску , температури та загазованості повітря BME680

BME680 є першим датчиком газу, який об’єднує високолінійні та високоточні датчики газу, тиску, вологості та температури. Він спеціально розроблений для мобільних додатків і переносних пристроїв, де критичними вимогами є розмір і низьке енергоспоживання. BME680 гарантує - залежно від конкретного режиму роботи - оптимізоване споживання, тривалу стабільність і високу стійкість до EMC. Для вимірювання якості повітря для особистого благополуччя газовий датчик у BME680 може виявляти широкий діапазон газів, таких як леткі органічні сполуки (ЛОС). Датчик зображено на рис. 2.4

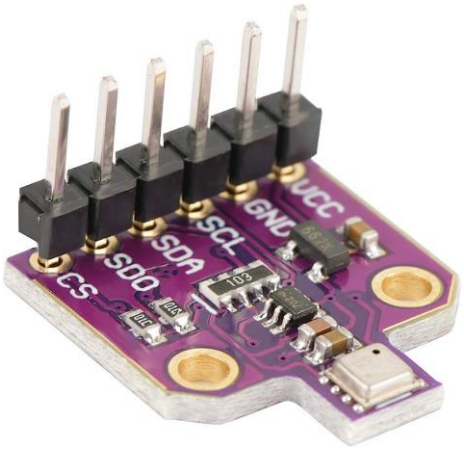


Рисунок 2.4 – BME680

Технічні характеристики наведені в таблиці 2.2

Таблиця 2.2 Технічні характеристики модуля

Діапазон вимірювання величин та точність	Тиск: 300… 1100 гПа Вологість: 0… 100% Температура: -40… 85 °C
Напруга живлення	1… 3.5 В
Інтерфейси	I2C та SPI

Продовження таблиці 2.2

Середнє споживання при частоті 1Гц **Середнє споживання в режимі сну**	2.1 мА при частоті 1Гц для вологості та температури; 3.1 мА при частоті 1 Гц для тиску та температури 3.7 мА при частоті 1 Гц для вологості, тиску та температури 0.09-12 мА для усіх величин, в залежності від режиму роботи
Датчик газу Час відгуку Споживання енергії Вивід даних	 <1 с; <0.1 мА (низьке споживання) Вивід індексу якості повітря
Датчик вологості Час відгуку Точність Гістерезиса	 8 с ± 3% відносної вологості ≤ 1.5 % відносної вологості
Датчик тиску Похибка вимірювання Здвиг температурного коефіцієнту	 ± 0.25 % ±1.3 Па/К

Модуль має 6 пінів, які наведені в таблиці 2.3

Таблиця 2.3 Піни BME680

VCC	Живлення (від 3.5 В до 5.5 В)
GND	Земля
SCL	SPI Clock Pin
SDA	Serial Data In
SDO	Serial Data Out
CS	Chip Select

2.2 Огляд альтернативних апаратних рішень

2.2.1 Одноплатний міні ПК Hackboard 2

Hackboard 2 — це одноплатний комп'ютер, який багато в чому з'явився завдяки краудфандингу. Розміри плати - 120x80 мм, її основа - 64-бітний Intel Celeron N4020. Плата має компактний, але функціональний дизайн із кнопкою живлення, світлодіодним індикатором стану живлення та невеликою батареєю RTC. Windows ставиться з коробки, також можна легко встановити будь-які x86-сумісні дистрибутиви Linux.

Для плати передбачено кілька додаткових пристроїв. Наприклад, у Hackboard 2 Starter Kit входить Hackboard 2 з Windows 10 Pro, блок живлення, тепловідведення, веб-камера, бездротова клавіатура із вбудованим тачпадом. У Hackboard 2 Complete Kit є ще й дисплей 13.3" IPS 1080p HD із вбудованими динаміками. Hackboard 2 зображено на рис. 2.5.

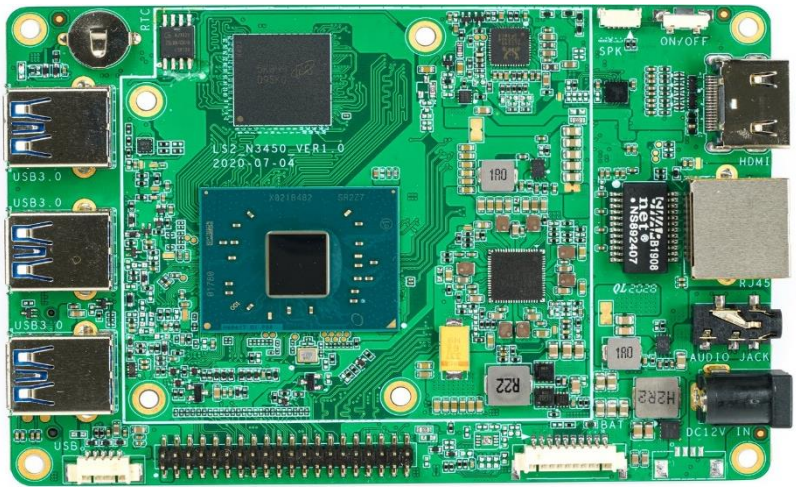


Рисунок 2.5 – Hackboard 2

Технічні характеристики Hackboard 2:

— Двоядерний процесор Intel Celeron N4020

— 4 ГБ оперативної пам'яті DDR4 64 ГБ EMMC

— слоти NVMe M.2, що дозволяють встановити SSD об'ємом до 4 ТБ

— Гігабітний Ethernet

— порти USB 3.0 типу B

— HDMI

— Аудіороз'єм для навушників

— Intel UHD Graphics 600 з виводом 4K HDMI 2.1 output

— 40-контактний Pi-сумісний GPIO

— Wi-Fi-модуль Intel dual-band AC95060

— Bluetooth 5.1

2.2.2 Одноплатний міні ПК Rock Pi 4

Rock Pi 4 - невеликий одноплатний комп'ютер на шестиядерному Hex Core Rockchip RK3399, досить потужний для свого форм-фактора та ціни. Комп'ютер підійде для використання як 4К мультимедіа приставки, ігрової ретро-приставки або простого домашнього сервера. Rock Pi 4 відповідає форм-фактору плати Raspberry Pi 3 та ASUS Tinker. Існує два варіанти Rock Pi 4 – модель A та B, дві ревізії v1.3 та v1.4. Єдина реальна різниця між моделлю A і моделлю B полягає в тому, що друга включає модуль бездротової мережі і додаткову підтримку для PoE. Підтримує Android TV 7, Android TV 9, Ubuntu Server 18.04, Debian 9 Desktop. Rock Pi 4 зображено на рис.2.6.

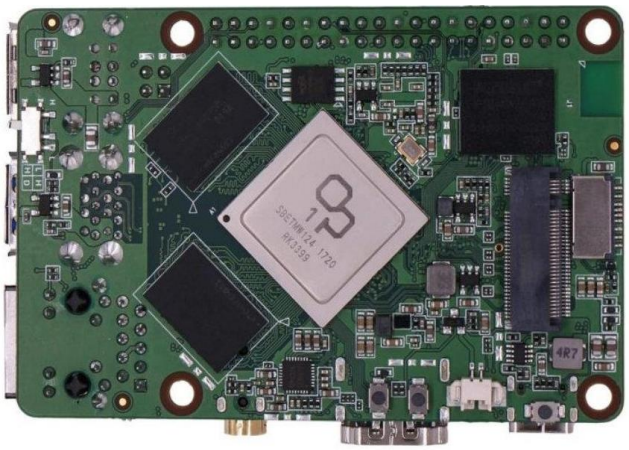


Рисунок 2.6 – Rock Pi 4

В основі системи Rock Pi - 64-бітовий процесор із шістьма ядрами. Вони розділені на два кластери: продуктивний, з двома ядрами ARM Cortrex A72, які працюють на частоті до 1,8 ГГц, і базовий з чотирма ядрами Cortex A53 з частотою до 1,4 ГГц. Коли потрібно виконувати завдання, які вимагають

значних обчислювальних ресурсів, використовується продуктивний кластер. В інших випадках система обходиться базовим кластером, споживає мінімум енергії.

Що стосується відеопідсистеми, це чотириядерна Mali T860 MP4, яка підтримує OpenGL ES версії від 1.1 до 3.2, Vulkan 1.0, а також Open CL 1.1 і 1.2 і DirectX 11.

Основні характеристики:

- Процесор RK3399 OP1 з 2 ядрами ARM Cortex A72 2 ГГц та 4 ядрами Cortex A53 1,5 ГГц.
- Графіка MP4 Mali T860
- ОЗУ LPDDR4-3200
- Модель B+ поставляється з модулем Wi-Fi 5 та Bluetooth 5.0. У A+ модуля немає.
- Порт Ethernet Gigabit
- Конектор GPIO
- MIPI DSI та MIPI CSI для підключення камери/екрана.
- HDMI
- Аудіопорт
- 2 роз’єма USB 2.0
- 3 роз’єма USN 3.0
- Конектор M.2 для SSD NVMe
- Слот для карт пам'яті формату MicroSD

2.2.3 Одноплатний міні ПК Asus Tinker Board

Плата ASUS Tinker Board - це перший комерційний SBC ASUS, розроблений як прямий конкурент Raspberry Pi Model B. Хоча в цій платі використовується застаріла технологія, вона, як і раніше, є життєздатною альтернативою до рівня документації та підтримки ASUS. Tinker Board нагадує RPi: тут чотири порти USB, HDMI, Ethernet. Asus Tinker Board зображено на рис. 2.7.

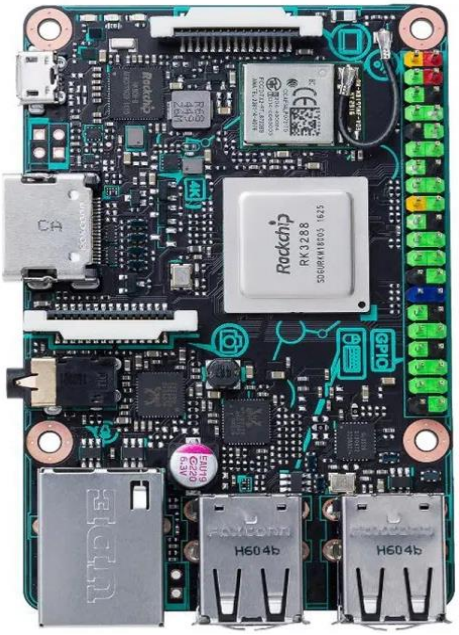


Рисунок 2.7 – Asus Tinker Board

Чіпсет Ethernet підтримує гігабітний інтерфейс, на відміну від 100 Мбіт/с на RPi. Плата позиціонується як навчальний матеріал щодо електроніки та програмування, як інструмент мейкера DIY, а також для виготовлення промислових додатків та пристроїв інтернету речей.

Технічні характеристики:

- Процесор: чотириядерний 1,8 ГГц ARM Cortex-A17 (Rockchip RK3288)
- GPU: ARM Mali-T764 GPU з підтримкою OpenGL ES1.1/2.0/3.0, OpenVG1.1, OpenCL, DirectX11
- Пам'ять: 2 ГБ Dual channel LPDDR3
- Зв'язок: Gigabit LAN та Bluetooth 4.0 + EDR
- Wi-Fi: 802.11 b/g/n із замінною антеною
- USB: 4 порти USB 2.0
- Інтерфейси розширення: 40 штиркових контактів, у тому числі 28 GPIO контактів для сигналів PWM та S/PDIF
- Звук: RTL HD кодек, аудіороз'єм 3,5 мм

— Камера: роз'єм Camera Serial Interface (CSI) для підключення камери
— Дисплей: 15-штирковий роз'єм Display Serial Interface (MIPI DSI) з підтримкою роздільної здатності HD
— Дисплей: один порт HDMI 2.0 з підтримкою роздільної здатності до 4K
— Зовнішня пам'ять: роз'єм для флеш-карт Micro SD із підтримкою UHS-I
— Операційна система: Debian OS / Kodi
— Живлення: 5V/2A Micro USB (кабель не входить у комплект).
— Максимальне енергоспоживання - 5 Вт. У неактивному режимі з HDMI 2,25 Вт, без HDMI – 2 Вт

2.2.4 NanoPi R4S

NanoPi R4S – це одноплатний headless-ПК з процесором RK3399 та двома гігабітними портами Ethernet. Він не призначений для використання як робочий стіл, т.к. не має відеовиходу. Зате цей крихітний (66-66 мм) SBC можна використовувати як основу для маршрутизаторів, компактних мережевих сховищ даних, шлюзів і т.д. Заявлений діапазон робочих температур тягнеться від мінус 20 до плюс 70 градусів Цельсія. Пристрій працює з OpenWrt, але можна використовувати інші зборки. Працює із Docker CE. NanoPi R4S зображено на рис.2.8.

Рисунок 2.8 – NanoPi R4S

Технічні характеристики:

— 64-бітний процесор Rockchip RK3399 з тактовою частотою 1,8 ГГц
— SoC – Rockchip RK3399 з двоядерним Core Cortex-A72, чотириядерним Cortex-A53 плюс відеочіп Mali-T86 з OpenGL ES1.1/2.0/1.0. з підтримкою AFBC, 4K VP9 та 4K 10-bit H265/H264 60fps.
— ОЗУ - 4 ГБ DDR3 або 1 ГБ DDR4.
— Слот для карток пам'яті microSD.
— Мережеві підключення - 2 гігабітні порти Ethernet
— USB – 2x USB 3.0 Type-A та USB 2.0.
— Розширення - 2*5 піновий конектор з SPI та l2C.
— Налагодження - 3-хпіновий UART.
— Декілька світлодіодів, включаючи power, SYS, LAN, WAN
— Конектор для підключення кулера.
— Живлення - USB-C

2.2.5 Zero Pi

ZeroPi - продукт від Friendly Arm, який позиціонується як пристрій для розробників, які створюють smart-гаджети та працюють з IoT-технологіями. Одноплатник, що не має відеовиходу, може служити як база для створення файлового сервера або роутера. Він оснащений чотириядерним процесором Allwinner H3, який працює на частоті 1,2 ГГц. Відсутність GPIO звужує сферу застосування пристрою, однак для тих, кому потрібен мало споживаючий і дуже дешевий Linux SBC, це гарне придбання. Розміри комп'ютера становлять 40x40 мм. Гаджет працює зі спеціалізованою ОС Friendly Core, база якої – Ubuntu Core. За бажанням можна встановити складання OpenWrt на ядрі Linux, яка призначена для маршрутизаторів. Zero Pi зображений на рис.2.9.

Рисунок 2.9 – Zero Pi

Технічні характеристики:

— SoC - чотириядерний процесор Allwinner H3 Cortex A7 @ 1.2 ГГц з графічним процесором Arm Mali-400MP2

— Оперативна пам'ять – 256 чи 512 Мб DDR3

— Сховище – слот для мікро SD-карти, опційно SPI флеш-пам'ять

— Мережа – гігабітний Ethernet через Realtek RTL8211E PHY

— USB – 1 порт USB 2.0, 1 micro USB порт

— Налагодження - 4-контактний роз'єм для послідовної консолі

— Різне – світлодіоди стану та живлення

— Живлення – 5 В / 2 А через мікро USB

2.2.5 NVIDIA Jetson Nano 2GB

NVIDIA Jetson Nano 2GB — остання розробка в лінійці комплектів для розробки NVIDIA SBC із чотириядерним процесором Arm A57 1,43 ГГц та 128-ядерним графічним процесором Maxwell. Компанія запевняє, що це чи не найкращий стартовий комплект штучного інтелекту та робототехніки для студентів, викладачів та любителів робототехніки. Новинка підтримується комплектом NVIDIA JetPack SDK, який постачається із середовищем виконання контейнера NVIDIA та повним середовищем розробки програмного забезпечення Linux. Заявлено підтримку інтерфейсів GPIO, I2C, I2S, SPI, PWM, UART. Рішення оснащене пасивною системою охолодження на базі радіатора, завдяки чому має нульовий рівень шуму. Швидкодія досягає 472 GFLOPS.

Пристрій зображено на рис.2.7.

Рисунок 2.10 – NVIDIA Jetson Nano

Технічні характеристики:

— CPU: 64-бітовий Quad-core ARM A57 (1.43 ГГц)

— GPU: 128-core NVIDIA Maxwell™

— GPU RAM: 2ГБ 64-bit LPDDR4

— Зв'язок: Ethernet 10/100/1000,

— Порти: 2x USB 2.0, 1x USB 3.0, HDMI, 1x CSI Camera Connector.

— Інтерфейс: 40-піновий GPIO (як на платформі RPi).

— USB 2.0 Micro-B (для роботи в режимі device mode)

— 4-хпіновий роз'єм для вентилятора

— Живлення через USB-C 5V 3A

Висновки до розділу 2

У другому розділі було проведено аналіз компонентів для створення апаратного комплексу. Було вивчено особливості одноплатного комп’ютера Raspberry Pi та його аналогів. Враховуючи результати проведених аналізів, зроблено наступні висновки:

— Raspberrry Pi буде використано в якості брокера згідно з протоколом MQTT

— Інформація буде надходити з кластера, зібраного з Wemos D1 Mini та модуля відстеження температури, вологості , тиску та загазованості повітря BME680.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ КЛАСТЕРНОЇ СИСТЕМИ ДОСТАВКИ КОНТЕНТУ НА БАЗІ СЕНСОРНОЇ МЕРЕЖІ

3.1 Апаратна частина

Апаратна частина кластерної мережі складається з 3-х елементів:

— Raspberry Pi

— Wemos D1 Mini

— BME680

Raspberry Pi виступає в ролі брокера, який отримує дані з кластерної мережі, створеної з використанням модулів ESP8266 та BME680.

Підключення модулів показано в таблиці 3.1

Таблиця 3.1 – Підключення модулів Wemos D1 Mini та BME680.

VCC	3.3V
GND	GND
SCL	GPIO 5
SDA	GPIO 4

Принципова схема підключення компонентів зображена на рис. 3.1

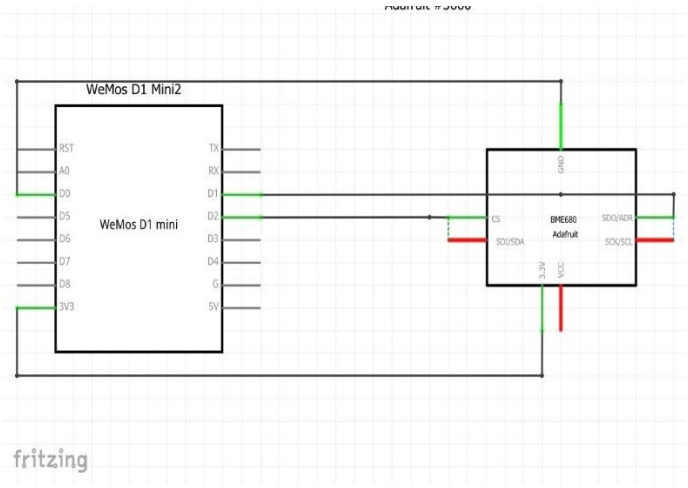


Рисунок 3.1 – Принципова схема підключення компонентів кластеру

Макетна схема підключення компонентів зображена на рис. 3.2

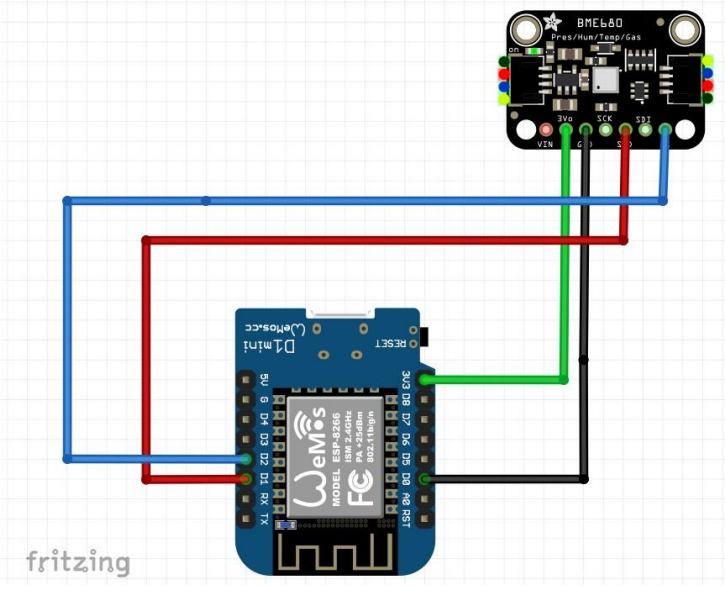


Рисунок 3.2 – Макетна схема з’єднання компонентів

3.2 Програмна частина мережі відстеження показників

Алгоритм роботи сенсорної мережі полягає у наступному:

— Wemos D1 Mini надсилає запит щодо даних, відстежуваних BME680

— Дані температури публікуються в топіку wemos/bme680/temp

— Дані вологості публікуються в топіку wemos/bme680/hum

— Показники тиску публікуються в топіку wemos/bme680/press

— Показники загазованості повітря публікуються в топіку wemos/bme680/gas

— Сервер «підписаний» на ці дані, та отримує їх через брокер

Блок-схема роботи апаратно-програмного комплексу зображена на рис. 3.3

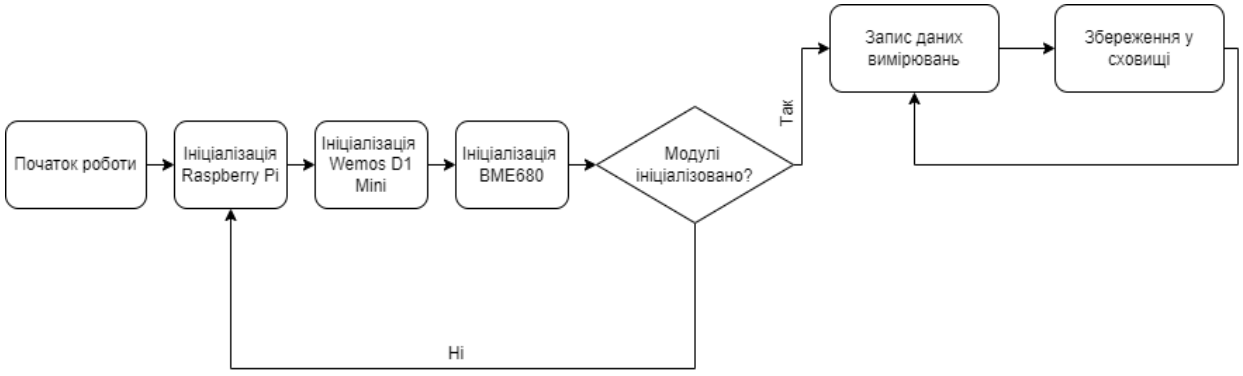


Рисунок 3.3 Блок-схема роботи комплексу

3.2.1 Створення та запуск веб-частини

Веб частина є однією з найголовніших частин комплексу, оскільки завдяки ній користувач має змогу взаємодіяти з показниками, аналізувати їх та приймати відповідні рішення.

Raspberry Pi буде взаємодіяти з Wemos D1 Mini за протоколом MQTT. Для надійного з'єднання потрібно встановити брокер Mosquitto.

Перед цим варто запустити термінал Raspberry Pi (рис.3.3) та ввести команди `sudo apt update && sudo apt upgrade` задля перевірки системи на предмет оновлень. Після цього варто ввести наступну команду:

```
sudo apt install -y mosquitto mosquitto-clients
```

Дана команда встановить брокер на систему. Орієнтовний час завантаження оновлень та брокера – 10 хвилин.

Оскільки комплекс повинен передавати дані безперервно, було б доцільно забезпечити автоматичний початок роботи Mosquitto відразу після запуску Raspberry Pi. Це можна реалізувати за допомогою наступної команди:

```
sudo systemctl enable mosquitto.service
```

```
pi@raspberrypi:~ $ sudo systemctl enable mosquitto.service
Synchronizing state of mosquitto.service with SysV service script with /lib/syst
emd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable mosquitto
pi@raspberrypi:~ $ mosquitto -v
1639226740: mosquitto version 2.0.11 starting
1639226740: Using default config.
1639226740: Starting in local only mode. Connections will only be possible from
clients running on this machine.
1639226740: Create a configuration file which defines a listener to allow remote
 access.
1639226740: For more details see https://mosquitto.org/documentation/authenticat
ion-methods/
1639226740: Opening ipv4 listen socket on port 1883.
1639226740: Error: Address already in use
1639226740: Opening ipv6 listen socket on port 1883.
```

Рисунок 3.4. Термінал Raspberry Pi.

Задля перетворення Raspberry Pi на веб-сервер необхідно встановити flask та socket.io. SocketIO дозволяє створити веб-сторінку , яка буде оновлюватись

разом з програмою. Тобто, користувач не буде змушений постійно перезавантажувати веб-сторінку, оскільки всі дані будуть оновлюватись одразу по мірі їх змінення. Встановлення виконується за допомогою наступної команди:

```
sudo pip install flask-socketio
```

Для створення веб-частини використано мову програмування Python.

Python — це високорівнева, інтерпретована, інтерактивна та об'єктно-орієнтована мова програмування. Python призначений для створення коду, який буде мати високу читабельність. В мові часто використовуються англійські ключові слова там, де інші мови використовують знаки пунктуації, і має менше синтаксичних конструкцій, ніж інші мови.

Окрім вищезгаданих можливостей, Python має великий список хороших функцій, деякі з яких перераховані нижче:

- Мова підтримує функціональні та структуровані методи програмування, а також ООП.
- Може бути використана як мова скриптів або скомпільована в байт-код для створення великих програм.
- Забезпечує динамічні типи даних дуже високого рівня та підтримує динамічну перевірку типів.
- Код, написаний на Python, легко може бути інтегровано з C, C++, COM, ActiveX, CORBA та Java.

Для початку, створюємо нову теку:

```
mkdir web-server
cd web-server
```

Створюємо у директорії новий файл app.py

```
nano app.py
```

Для належної роботи застосунку потрібно імпортувати компоненти з вищезгаданих Flask та SocketIO

```
from flask import Flask, render_template, request
from flask_socketio import SocketIO, emit
```

3.2.2 Збереження даних вимірювань

Дуже важливою з точки зору практичності є створення бази даних , яка допоможе зберігати дані вимірювань для удосконаленого моніторингу та їх подальшого аналізу. Для реалізації цієї можливості найкращим рішенням буде створення бази даних за допомогою SQLite.

SQLite — це вбудована бібліотека, яка реалізує самодостатній безсерверний механізм транзакційної бази даних SQL без конфігурації. Код для SQLite знаходиться у відкритому доступі, тому його можна безкоштовно використовувати для будь-яких цілей, комерційних чи приватних. SQLite — одна з найбільш використовуваних технологій створення баз даних у світі. SQLite — це вбудований механізм бази даних SQL. На відміну від більшості інших баз даних SQL, SQLite не має окремого серверного процесу. SQLite читає та записує безпосередньо у звичайні дискові файли. Повна база даних SQL із кількома таблицями, індексами, тригерами та представленнями даних міститься в одному дисковому файлі. Формат файлу бази даних є крос-платформним — ви можете вільно копіювати базу даних між 32-розрядними та 64-розрядними системами або між архітектурами big-endian та little-endian. Ці функції роблять SQLite популярним вибором як формат файлу програми.

SQLite — це компактна бібліотека. Якщо всі функції ввімкнено, розмір бібліотеки може бути менше 750 КБ, залежно від цільової платформи та налаштувань оптимізації компілятора. (64-розрядний код більший. І деякі оптимізації компілятора, такі як агресивне вбудовування функцій і розгортання циклу, можуть призвести до того, що об’єктний код буде набагато більшим.) Існує компроміс між використанням пам’яті та швидкістю. SQLite зазвичай працює швидше, чим більше пам’яті ви йому надаєте. Тим не менш, продуктивність зазвичай досить хороша навіть у середовищах з низьким обсягом пам'яті. Залежно від того, як він використовується, SQLite може бути швидшим, ніж прямий ввід-вивід файлової системи.

Для встановлення SQLite використовується наступна команда:

```
sudo apt-get install sqlite3
```

Після встановлення усіх потрібних компонентів, створюємо базу даних:

```
sqlite3 data.db
```

Таким чином, створено файл data.db.

Оскільки всі необхідні компоненти та база даних вже створені, можемо перейти до написання її структури. За допомогою командної стрічки SQLite створюємо структуру бази за допомогою команди `.fullschema.`

```
.fullschema
CREATE TABLE bmereadings(
id INTEGER PRIMARY KEY AUTOINCREMENT,
temperature NUMERIC,
humidity NUMERIC,

pressure NUMERIC,

gas NUMERIC,

currentdate DATE,

currentime TIME,

device TEXT);
```

Коректна адреса бази даних повинна бути прописана у вищезгаданому файлі App.py, інакше дані не будуть відображені.

3.2.3 Відображення даних

Задля кращої організації роботи апаратно-програмного комплексу, доцільно відокремити скрипти для створення та запуску веб-сервера та відображення даних для кінцевого користувача. Найкращим виходом з ситуації є написання сторінки на мові HTML. Flask використовує Jinja2, який дозволяє відправляти динамічні дані зі скрипта на Python до файлу HTML.

HTML (розшифровується як Hypertext Markup Language) - мова розмітки, яка використовується для створення та структурування вмісту у Мережі. Це стандартна мова, яка використовується для створення веб-сайтів і веб-додатків.

Використання HTML для створення веб-сторінок і веб-додатків має кілька переваг, зокрема:

- Простота: HTML — це проста мова розмітки, яку легко вивчити та використовувати, що робить її доступною як для початківців, так і для досвідчених розробників.
- Сумісність: HTML є широко поширеним стандартом, тому він сумісний із більшістю веб-браузерів і може використовуватися на різних пристроях, включаючи настільні комп’ютери, ноутбуки, планшети та смартфони.
- Доступність: HTML надає спосіб структурувати веб-вміст таким чином, щоб він був доступним для людей з обмеженими можливостями, наприклад тих, хто використовує програми зчитування з екрана або інші допоміжні технології.

- Оптимізація пошукових систем: HTML можна оптимізувати для пошукових систем за допомогою відповідних тегів заголовків, метаописів та інших методів, які можуть покращити видимість і рейтинг веб-сайту в результатах пошуку.
- Настроюваність: HTML можна поєднувати з іншими технологіями веб-розробки, такими як CSS і JavaScript, для створення власних і динамічних веб-сторінок і веб-додатків, які відповідають певним вимогам дизайну та функціональності.

Загалом HTML забезпечує простий і гнучкий спосіб створення веб-сторінок і веб-додатків, доступних, сумісних і оптимізованих для пошукових систем, що робить його важливою мовою для веб-розробки.

Хоча HTML має багато переваг, є також деякі обмеження та недоліки у його використанні для веб-розробки. Деякі з основних недоліків включають:

- Обмежена функціональність: HTML в основному використовується для створення структури та вмісту веб-сторінок, але він має обмежену функціональність, коли йдеться про динамічні функції,

інтерактивність і складну анімацію. Для досягнення цих функцій потрібні інші мови веб-розробки, такі як JavaScript і CSS.

— Питання безпеки: HTML надає обмежені функції безпеки, що робить його вразливим до таких атак, як міжсайтовий сценарій (XSS) і впровадження SQL.

— Проблеми сумісності браузера: різні веб-браузери можуть по-різному інтерпретувати HTML-код, що спричиняє проблеми сумісності та невідповідності у зовнішньому вигляді та функціональності веб-сторінок. Розробникам потрібно тестувати свій код у кількох браузерах, щоб забезпечити кросбраузерну сумісність.

— Відсутність варіантів дизайну: хоча HTML забезпечує структуру веб-сторінок, він не пропонує багато варіантів дизайну. CSS зазвичай використовується для стилізації, але навіть із CSS існують обмеження щодо доступних варіантів візуального дизайну.

— Крута крива навчання: хоча HTML порівняно простий порівняно з іншими мовами веб-розробки, він все одно вимагає певних знань і досвіду для ефективного використання. Ця крива навчання може стати перешкодою для новачків, які тільки починають веб-розробку.

Незважаючи на ці обмеження, HTML залишається основною мовою веб-розробки, забезпечуючи основу для веб-сторінок і веб-додатків.

3.3 Програмна частина сенсорної мережі

Функціонування апаратної частини неможливе без якісної програмної складової, яка створена для інтеграції компонентів між собою, забезпечуючи їх належне функціонування. Програмне забезпечення повинне бути простим у використанні та підтримувати високий рівень кастомізації, що водночас робитиме його гнучким та зручним для користування.

Перед розробкою програмної частини було досліджено наявні середовища розробки, які призначені для створення програмної частини апаратно-програмних комплексів, що містять в собі модулі серії ESP. Розглянуто два середовища розробки – ESPLorer та Arduino IDE. ESPLorer дозволяє створювати прошивку на Wemos D1 Mini за допомогою мови програмування Lua, у той час як Arduino IDE розраховане на написання коду мовами C/C++. ESPLorer відрізняється можливістю більш детальних налаштувань плати, що прошивається, включно з моніторингом вільного місця та ресурсами, що використовуються під час роботи пристрою. Водночас, користування середовищем ускладнюється відсутністю достатньої користувацької бази та активних розробок. Це створює незручності при виникненні проблем під час проектування апаратного комплексу, а інколи й унеможливлює їх рішення через недостатню кількість користувачів, задіяну в використанні ПЗ. Основними перевагами Arduino IDE є активне ком’юніті розробників, що дозволяє звернутися за допомогою та оперативно вирішити проблему у разі її виникнення, та велика база бібліотек, яка оновлюється майже щодня. Це дозволяє створювати комплекси з великою кількістю сторонніх пристроїв.

Arduino IDE (Integrated Development Environment) - відкрите середовище розробки, яке дозволяє користувачам писати та завантажувати код до робочого середовища в режимі реального часу. Оскільки написаний код можна переносити до хмарного сховища , середовище розробки зазвичай використовується для написання коду з надмірною наповненістю. Arduino IDE пропонує повну сумісність з будь-яким Arduino-сумісним пристроєм. Програмне забезпечення може з легкістю бути запущене на будь-якому пристрої, який підтримує ОС Windows, Mac або Linux. Більшість елементів IDE написано на JavaScript, що дає змогу плавно компілювати та редагувати код. Інтерфейс програми (рис.3.4) є зручним та зрозумілим для користувача.

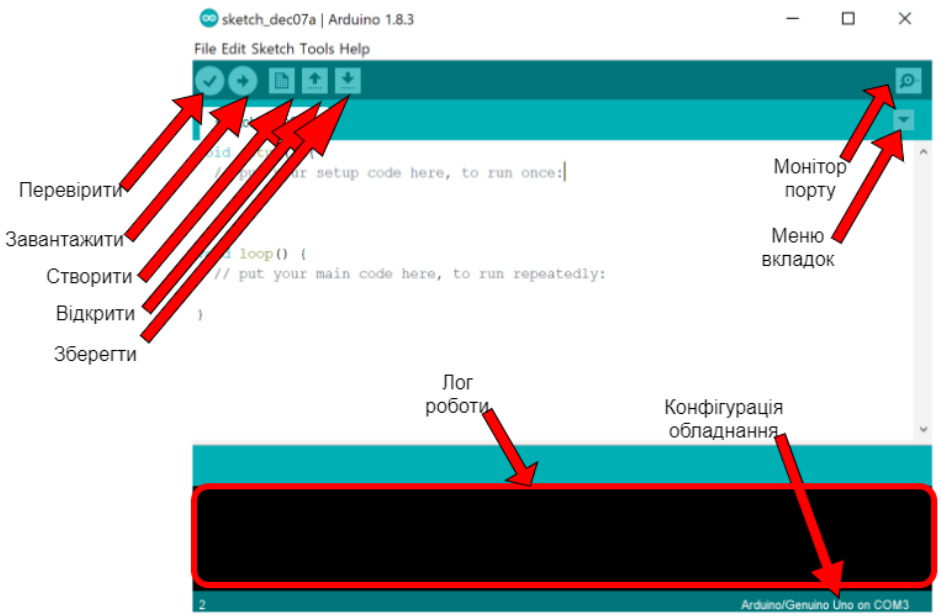


Рисунок 3.5.- Інтерфейс програмного середовища Arduino IDE

По центру інтерфейсу розташовано блокнот (sketchbook) - основне середовище для написання коду. На екрані розташовано кнопки виконання різних програмних функцій:

— **Перевірити** - використовується для компіляції та перевірки коду без його завантаження в плату. Тобто код можна написати та перевірити на помилки навіть не підключаючи плату до комп`ютера.

— **Завантажити** - компіляція та завантаження прошивки до плати.

— **Створити** - створення нового скетча

— **Відкрити** - відкриття готового скетчу

— **Монітор порту** - застосовується для відкриття монітору послідовного порту для взаємодії з платою

— **Меню вкладок** - робота з вкладками

— **Лог роботи** - у цьому полі виводиться лог компіляції та взагалі всі системні повідомлення

— **Конфігурація обладнання** - у цьому полі виводиться назва обраної плати , версії мікроконтролера та номер обраного COM порту.

— Меню представлено 4 головними вкладками :

— Файл (File) (рис 3.5)

Містить 12 вкладок:

— **Новий (new)** - створення нового файлу

— **Відкрити (open)** - відкриття існуючого файлу

— **Відкрити нещодавні (open recent)** - відкриття останнього закритого файлу

— **Тека зі скетчами (sketchbook)** - тека, в якій відбувається збереження скетчів за замовчуванням

— **Приклади (examples)** - список встановлених бібліотек з підсписком прикладів у кожній

— Закрити (close)

— Зберегти (save)

— Зберегти як (save as)

— **Налаштування сторінки (page setup)** - використовується у разі друкування вікна IDE

— Друк (print)

— Налаштування (preferences)

— Вихід (quit)

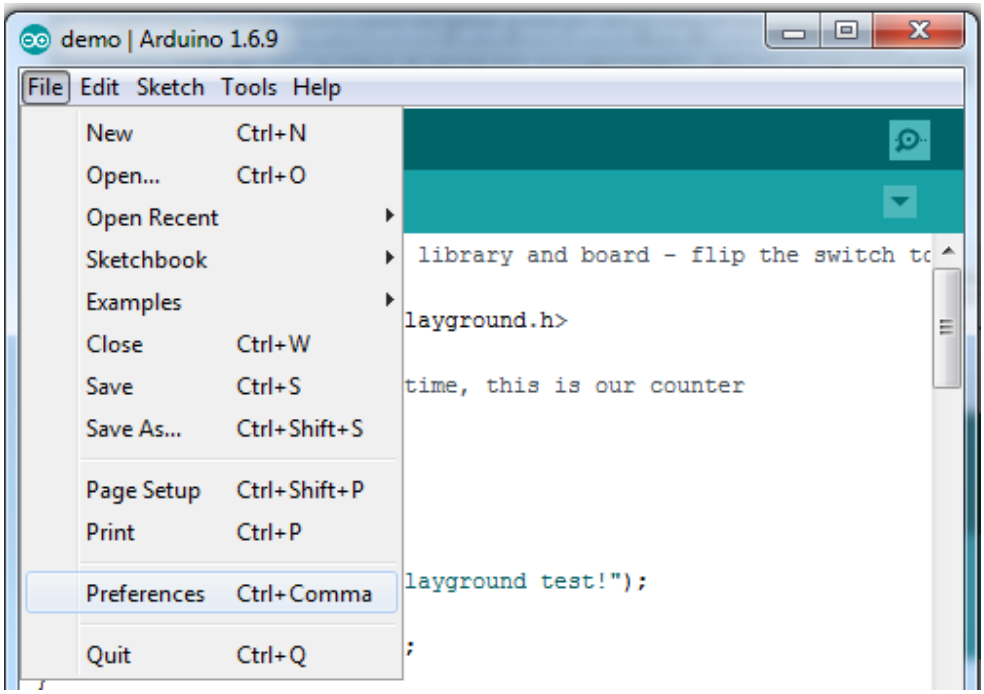


Рисунок 3.6 - Вкладка “File” середовища Arduino IDE

Редагувати (Edit) (рис. 3.6)

Містить 8 вкладок:

— Перехід назад/вперед (undo/redo)

— Вирізати/Копіювати/Вставити (Cut/Copy/Paste)

— **Копіювати для форуму (Copy for Forum)** - копіює код та зберігає його в буфері обміну у форматі, прийнятному для розміщення на форумі

— **Копіювати як HTML (Copy as HTML)** - копіює код та зберігає його в буфері обміну у форматі, прийнятному для веб-сторінок.

— Вибрати все (Select all)

— Коментувати/Розкоментувати (Comment/uncomment)

— Збільшити/зменшити відступ (Increase/decrease indent)

— **Знайти/Знайти наступне/Знайти попереднє (Find/Find next/Find previous)** - використовується для знаходження попереднього/наступного висвітлення заданого рядку

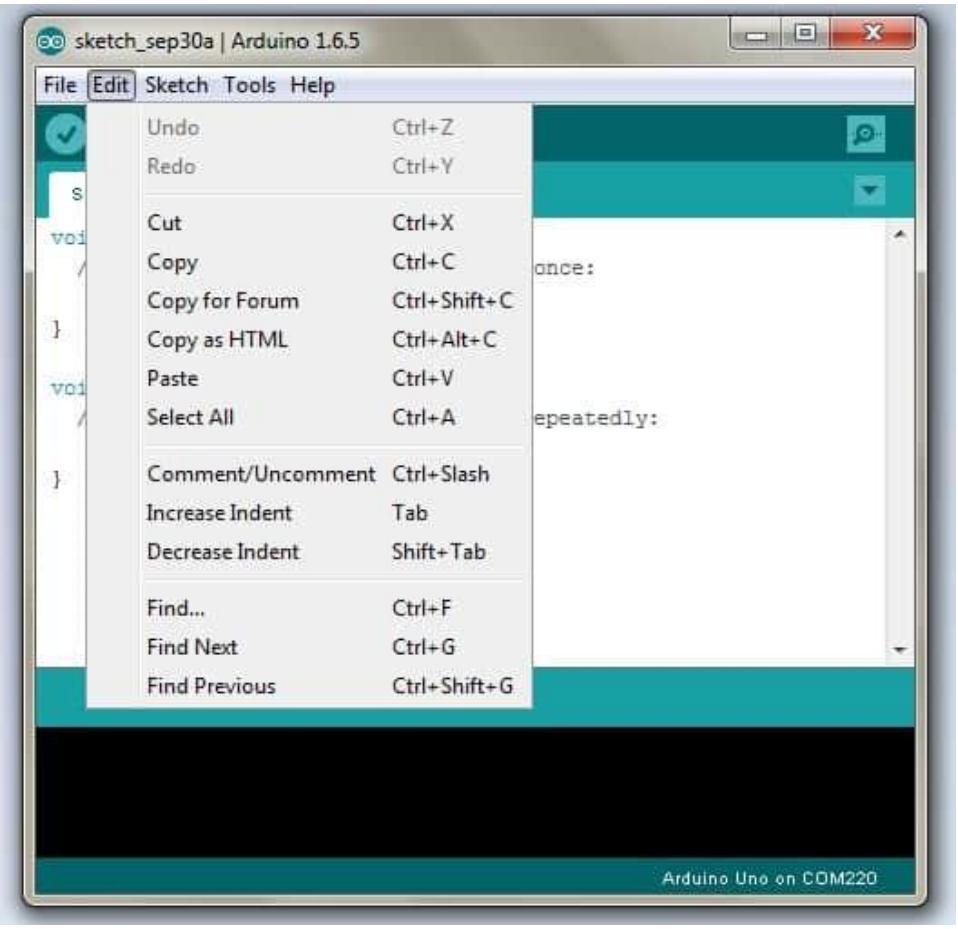


Рисунок 3.7. Вкладка "Edit" середовища Arduino IDE

Скетч (Sketch) (рис 3.7)

Містить 7 вкладок:

— Перевірити/компілювати (verify/compile)

— Завантаження (Upload)

— **Завантаження через програматор (upload using Programmer)** - завантаження скетча безпосередньо до мікроконтролера, оминаючи завантажувач

— **Експорт скомпільованого бінарного файлу (export compiled binary)** - зберігає скомпільований файл , який завантажується до мікроконтролера.

— Показати теку скетча (show sketch folder)

— **Під'єднати бібліотеку (include library)** - під'єднує до коду бібліотеку з використанням директиви include.

— **Додати файл (Add file)** - під'єднує до коду зовнішній файл

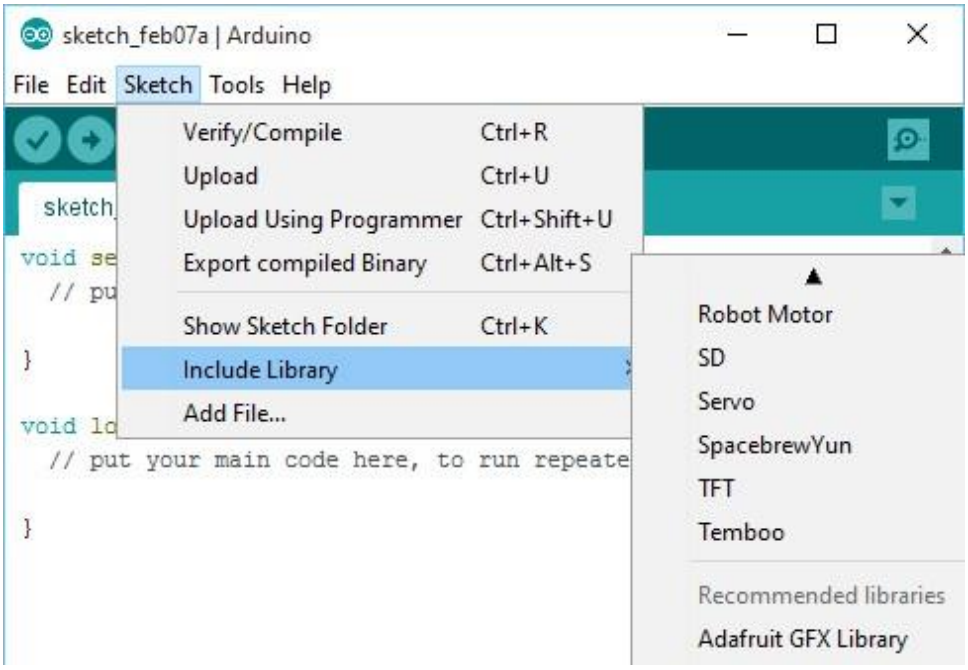


Рисунок 3.8. - Вкладка "Sketch" середовища Arduino IDE

Інструменти (Tools) (рис. 3.8)

Містить 12 вкладок:

— Автоформатування (Auto format) - вирівнює код за табуляцією.

— Архівувати скетч (Archive sketch) - зберігає скетч як .zip архів

— **Виправити кодування та перезавантажити (Fix encoding & reload)** - дозволяє виправляти завантажений зі сторонніх ресурсів код у разі збою кодування
— **Керувати бібліотеками (Manage libraries)** - дозволяє керувати бібліотеками з офіційного списку
— Монітор порту (Serial Monitor)
— **Плотер послідовного з’єднання (Serial Plotter)** - вбудований інструмент для створення графіків за прямуючими до порту даними.
— **Плата (Board:)** - вибір плати.
— **Порт (Port:)** - COM порт , до якого під’єднана плата.
— Отримати інформацію про плату (Get Board Info) - отримання інформації про обрану плату.
— **Програматор (Programmer:)** - вибір програматора для завантадження якоду через програматор
— **Записати завантажувач (Burn Bootloader)** - прошиває завантажувач, що відповідає обраній платі та прооцесору в мікроконтролер за допомогою програматора

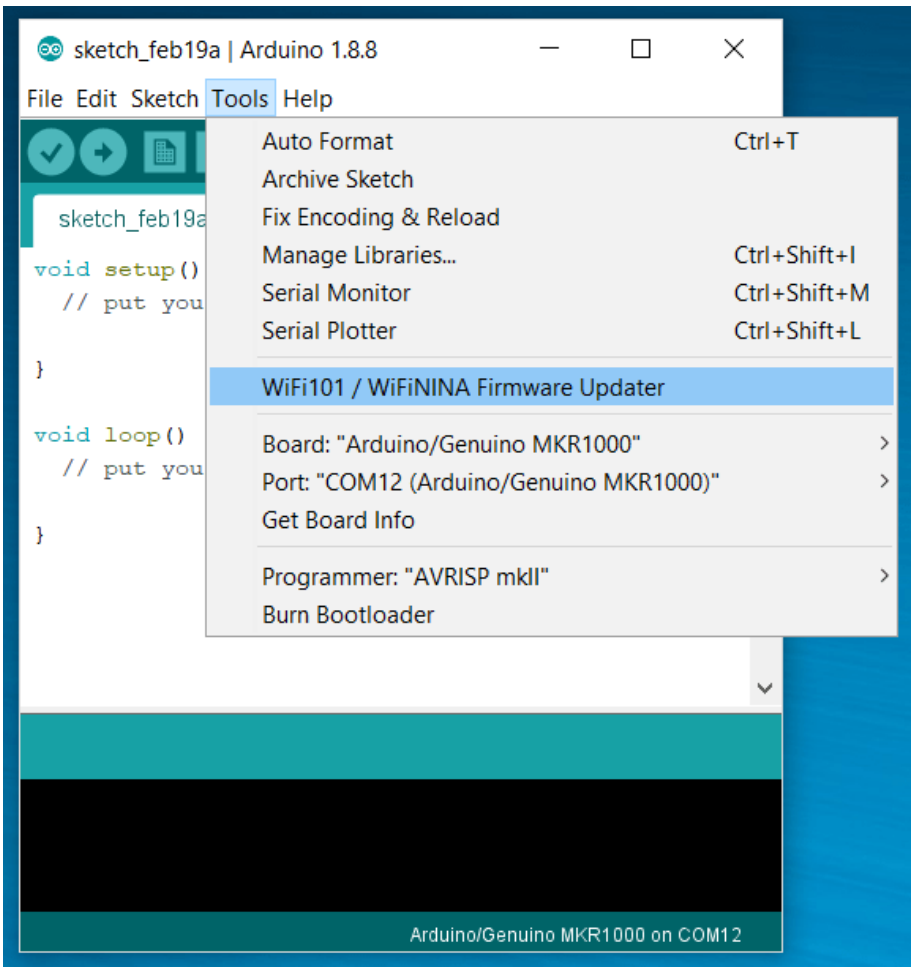


Рисунок 3.9 - Вкладка “Tools” середовища Arduino IDE

Меню вкладок (рис. 3.9)

Система вкладок в Arduino IDE працює вкрай незвично і дуже відрізняється від поняття вкладок в інших програмах:

— Вкладки відносяться до одного і того ж проекту, до файлів, що знаходяться з ним в одній папці

— Вкладки просто розділяють загальний код на частини, тобто в одній вкладці фігурна дужка { може відкритися, а в наступній - закритися }. При компіляції все вкладки просто об'єднуються в один текст по порядку зліва направо (з лівої вкладки до правої). Також це означає, що вкладки повинні містити код, що відноситься тільки до цього проекту, і реалізувати в одній вкладці void loop () та в іншій - не можна, оскільки loop () може бути тільки один

— Вкладки автоматично розташовуються в алфавітному порядку, тому створювана вкладка може з’явитися між іншими вже існуючими. Це означає, що розбивати блоки коду по різних вкладках не рекомендується.

— Усі змінні повинні бути оголошені до свого виклику, тобто вкладка з оголошенням змінної повинна бути лівіше вкладки, у якій змінна викликається. Створюючи нову вкладку потрібно відразу думати, де вона з'явиться з таким ім'ям і чи не створить це проблем. Також назва вкладок можна починатися з цифр і таким чином допоможе точно контролювати їх порядок. Щоб уникнути проблем зі змінними, все глобальні змінні краще оголошувати в найпершій вкладці.

— Вкладки зберігаються в папці з проектом і мають розширення .ino, при запуску будь-якої вкладки відкриється весь проект з усіма вкладками.

— Крім "рідних" .ino файлів Arduino IDE автоматично підчіплює файли з розширеннями .h (заголовки), .cpp (файл реалізації) і .pde (старий формат файлів Arduino IDE). Ці файли так само з'являються у вигляді вкладок, але, наприклад, заголовки .h не беруть участі в компіляції до тих пір, поки не будуть вручну підключені до проекту за допомогою команди include. Тобто він висить як вкладка, його можна редагувати, але без підключення він так і залишиться просто окремим текстом. У таких файлах зазвичай містяться класи або просто окремі масиви даних.

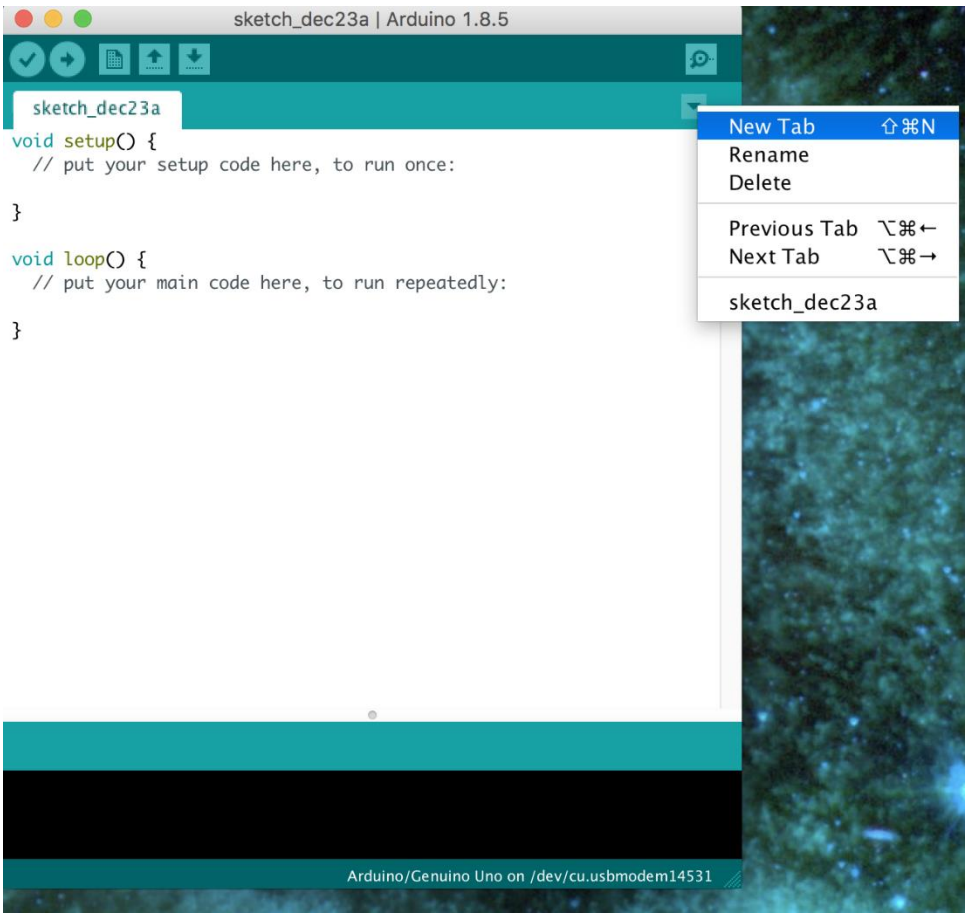


Рисунок 3.10 - Меню вкладок у середовищі Arduino IDE

Wemos D1 Mini є основою кластера, який відповідає за зчитування та передачу виміряних даних з датчиків. Для надійної роботи з протоколом MQTT обов’язковим є використання наступних бібліотек:

— PubSubClient

— AsyncTCP

Бібліотека **PubSubClient** забезпечує роботу сервера, який здійснює передачу повідомлень методом Publish/Subscribe. Для комунікації з мережевими пристроями використовується Arduino Ethernet Client API, що дозволяє використовувати її з низкою проектів, які містять в собі кластери з використанням WiFi модулів Sparkfun WiFly Shield, Intel Galileo/Edison, ESP32 та ESP8266.

Окрім перваг, бібліотека має ряд обмежень:

- Є змога публікувати лише повідомлення рівня QoS 0. Він може підписатися на QoS 0 або QoS 1.
- Максимальний розмір повідомлення, включаючи заголовок, за замовчуванням становить 256 байт. Це можна налаштувати через MQTT_MAX_PACKET_SIZE у PubSubClient.h або змінити за допомогою виклику PubSubClient::setBufferSize(size).
- За замовчуванням інтервал підтримки активності встановлено на 15 секунд. Це можна налаштувати за допомогою MQTT_KEEPALIVE у PubSubClient.h або змінити викликом PubSubClient::setKeepAlive(keepAlive).
- За замовчуванням клієнт використовує MQTT 3.1.1. Його можна змінити на використання MQTT 3.1, змінивши значення MQTT_VERSION у PubSubClient.h.

Async TCP - повністю асинхронна бібліотека TCP, спрямована на створення безпроблемного мережевого середовища з декількома підключеннями для мікроконтролерів ESP8266.

Для реалізації зчитування даних з модуля відстеження тиску , температури , вологості та загазованості повітря необхідне встановлення бібліотек BME680 Sensor Library та Adafruit Sensor Library.

Висновки до розділу 3

В третьому розділі проаналіщовано наявні середовища розробки, необхідні для створення програмного забезпечення системи доставки контенту на базі сенсорної мережі. Пректування та написання програмної частини комплексу зроблено за допомогою наступних програмних забезпечень:

- Середовище розробки Arduino IDE (створення прошивки для модулів)
- Середовище проектування та макетування Fritzing (створення макетних та принципових схем комплексу)

— Мова програмування Python , використана при створені скриптів для роботи серверної частини

Розроблена програмна частина комплексу забезпечує виконання наступних задач:

— Відображення записаних даних у браузері в режимі реального часу.

— Збереження відстежених даних тиску, температури, вологості та загазованості повітря, з можливість доступу до збережених показників

Безперебійну передачу даних на сервер

РОЗДІЛ 4
РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ КЛАСТЕРНОЇ СИСТЕМИ ДОСТАВКИ КОНТЕНТУ НА БАЗІ RASPBERRY PI

4.1 Результати роботи апаратно-програмного комплексу.

Після того, як програмне забезпечення готове до налаштування та запуску, можна переходити до запуску веб-сервера. Для цього запускаємо термінал, переходимо до директорію, в якому розташовано файли комплексу, та запускаємо сервер за допомогою наступної команди:

```
sudo python app.py
```

У випадку правильного налаштування, сервер запуститься на порті 8181.

Після запуска сервера відкривається головна сторінка (рис.4.1), яка відображає вимірювані дані.

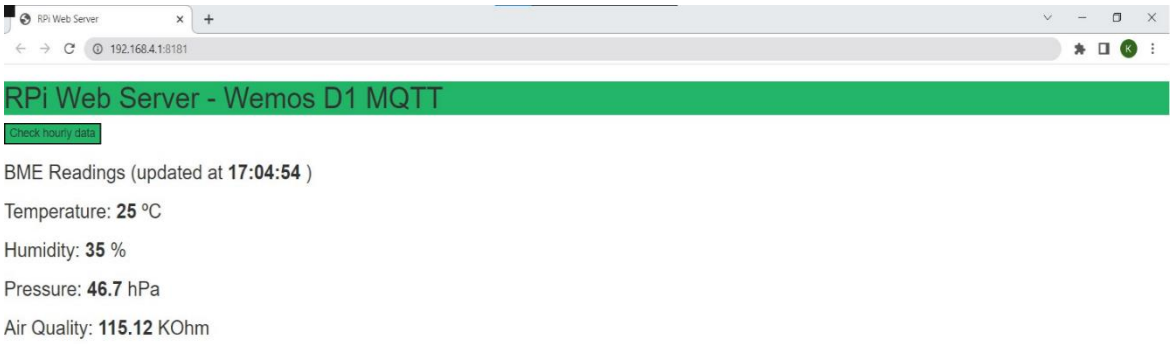
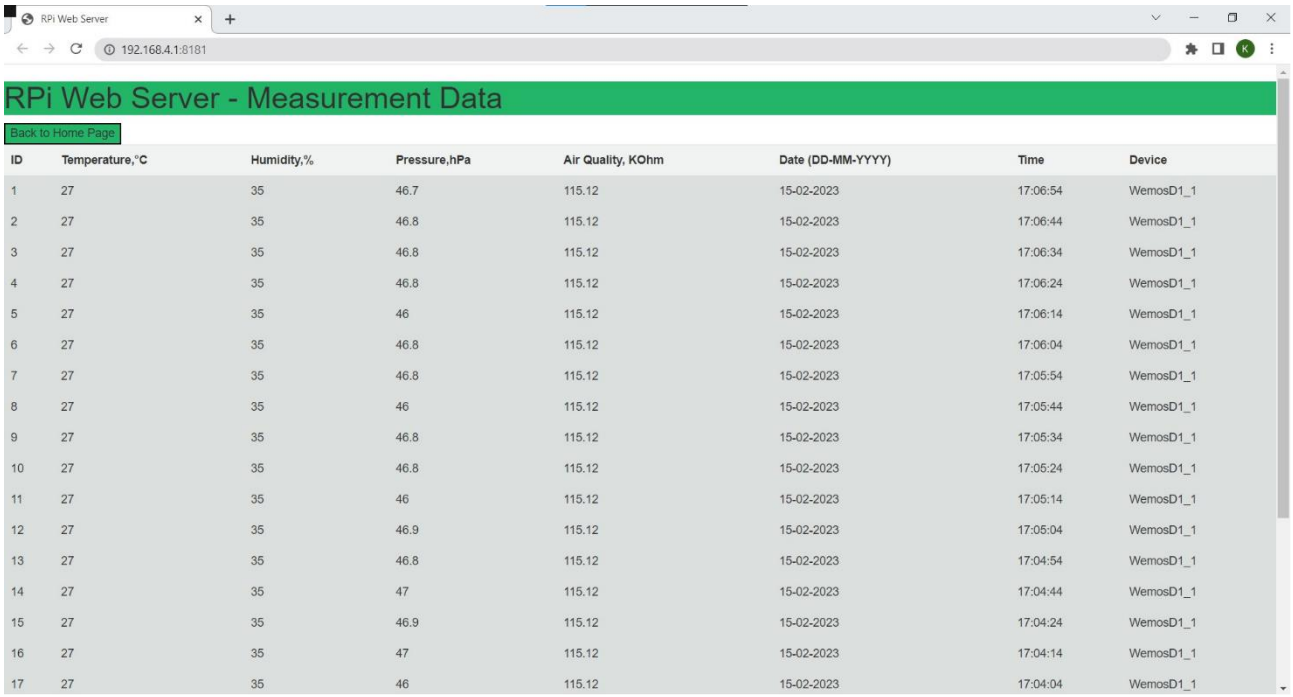


Рисунок 4.1. Головна сторінка з відображенням даних

Основна сторінка відображає дані, які вимірюються в режимі реального часу. Вказуються показники температури, вологості, тиску та рівня загазованості, також вказано час останнього оновлення показників.

Наверху сторінки розміщена кнопка «Check Hourly Data», за допомогою якої можна перейти на сторінку , де відображені дані, зібрані протягом деякого часу. Виміри згруповано та представлено у вигляді таблиці. Ця сторінка зображена на рис. 4.2.



ID	Temperature,°C	Humidity,%	Pressure,hPa	Air Quality, KOhm	Date (DD-MM-YYYY)	Time	Device
1	27	35	46.7	115.12	15-02-2023	17:06:54	WemosD1_1
2	27	35	46.8	115.12	15-02-2023	17:06:44	WemosD1_1
3	27	35	46.8	115.12	15-02-2023	17:06:34	WemosD1_1
4	27	35	46.8	115.12	15-02-2023	17:06:24	WemosD1_1
5	27	35	46	115.12	15-02-2023	17:06:14	WemosD1_1
6	27	35	46.8	115.12	15-02-2023	17:06:04	WemosD1_1
7	27	35	46.8	115.12	15-02-2023	17:05:54	WemosD1_1
8	27	35	46	115.12	15-02-2023	17:05:44	WemosD1_1
9	27	35	46.8	115.12	15-02-2023	17:05:34	WemosD1_1
10	27	35	46.8	115.12	15-02-2023	17:05:24	WemosD1_1
11	27	35	46	115.12	15-02-2023	17:05:14	WemosD1_1
12	27	35	46.9	115.12	15-02-2023	17:05:04	WemosD1_1
13	27	35	46.8	115.12	15-02-2023	17:04:54	WemosD1_1
14	27	35	47	115.12	15-02-2023	17:04:44	WemosD1_1
15	27	35	46.9	115.12	15-02-2023	17:04:24	WemosD1_1
16	27	35	47	115.12	15-02-2023	17:04:14	WemosD1_1
17	27	35	46	115.12	15-02-2023	17:04:04	WemosD1_1

Рисунок 4.2. Сторінка відображення даних, що були виміряні напередодні

Таблиця містить наступні ряди:

— ID – порядковий номер виміру, який відбувся

— Temperature - значення виміряної температури у приміщенні. Результати подаються у градусах Цельсію.

— Humidity – рівень вологості в приміщенні. Дані подаються у відсотках.

— Pressure – рівень атмосферного тиску.

— Air Quality – рівень якості повітря.

— Date – дата виміру.

— Time – час виміру.

— Device – назва апаратного комплексу, який здійснював вимір. В даному випадку, апаратний комплекс має назву «WemosD1_1».

Також на сторінці наявна кнопка «Back to home page». Кнопка реалізує функцію повернення на попередню сторінку, тобто на сторінку виміру показників в режимі реального часу.

Основний принцип роботи зображено на рис. 4.3.

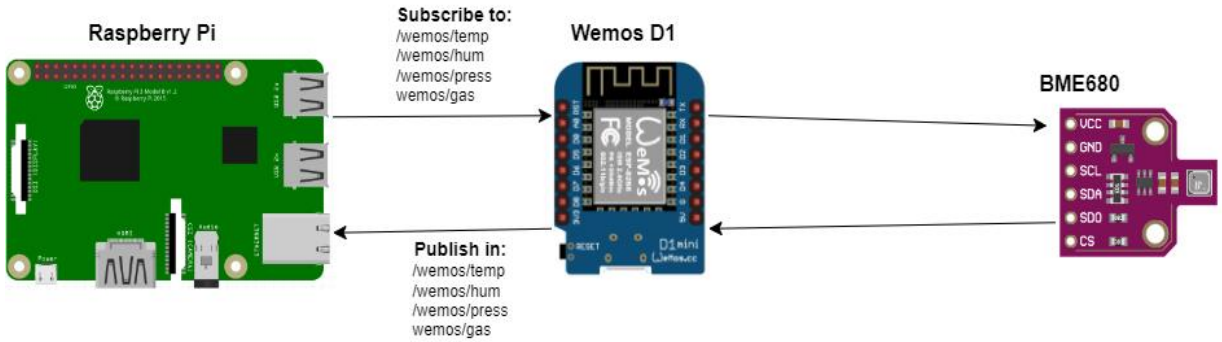


Рисунок 4.3. Основний принцип роботи апаратно-програмного комплексу

Спочатку відбувається ініціалізація компонентів апаратно-програмного комплексу. У випадку проблем з ініціалізацією цикл починається заново, аж поки всі компоненти не будуть запущені та передаватимуть дані. Після цього відбувається запуск веб-частини, яка отримує дані з сенсорної мережі та зберігає їх у новоствореній базі даних.

Оскільки комунікація пристроїв відбувається за протоколом MQTT , елементи апаратно-програмного комплексу комунікують між собою за принципом «Підписник - видавець» (PubSub, або «Publish - Subscribe»). Raspberry Pi «підписується» на дані з сенсорного комплексу, а саме на топіки, пов’язані з вимірюванням температури, вологості, тиску та якості повітря.

Дані вимірювань записуються кожні 10 секунд.

4.2 Методи тестування програмного забезпечення

Взаємодія компонентів є ключовою складовою , яка забезпечує якісну та безперебійну роботу пристрою. Для належної роботи апаратно-програмного комплексу необхідно переконатися, що усі компоненти здатні комунікувати між собою, передавати дані та представляти їх користувачеві без затримок.

Тестування програмного забезпечення — це різні стратегії або підходи, які використовуються для тестування програми, щоб переконатися, що воно працює і виглядає так, як очікується. Вони охоплюють все, від зовнішнього до внутрішнього тестування, включаючи модульне та системне тестування.

Мета використання численних методологій тестування в процесі розробки – переконатися, що ваше програмне забезпечення може успішно працювати у різних середовищах та на різних платформах. Зазвичай їх можна розділити на функціональне та нефункціональне тестування. Функціональне тестування включає тестування програми на відповідність бізнес-вимог. Він включає всі типи тестів, призначені для гарантії того, що кожна частина програмного забезпечення поводиться так, як очікується, з використанням варіантів використання, наданих групою розробників або бізнес-аналітиком. Ці методи тестування зазвичай проводяться по порядку і включають:

— Модульне тестування

— Інтеграційне тестування

— Тестування системи

— Приймальні випробування

Методи нефункціонального тестування включають всі типи тестування, орієнтовані на експлуатаційні аспекти програмного забезпечення. До них відносяться:

— Тестування продуктивності

— Тестування безпеки

— Юзабіліті-тестування

— Тестування сумісності

Модульне тестування – це перший рівень тестування, який часто виконують самі розробники. Це процес забезпечення того, щоб окремі компоненти програмного забезпечення на рівні коду функціонували та працювали так, як вони були розроблені. Розробники в тестовому середовищі зазвичай пишуть і запускають тести перед тим, як програмне забезпечення або

функція будуть передані групі тестування. Модульне тестування можна проводити вручну, але автоматизація процесу прискорить цикли доставки та розширить тестове покриття. Модульне тестування також спростить налагодження, оскільки виявлення проблем на більш ранньому етапі означає, що на їх виправлення потрібно менше часу, ніж якщо вони були виявлені пізніше в процесі тестування.

Інтеграційне тестування

Після ретельного тестування кожного модуля він інтегрується з іншими модулями до створення модулів чи компонентів, призначених до виконання певних завдань чи дій. Потім вони тестуються як група за допомогою інтеграційного тестування, щоб переконатися, що цілі сегменти програми поводяться так, як очікувалося (тобто взаємодія між модулями є безшовним). Ці тести часто засновані на сценаріях користувача, таких як вхід у додаток або відкриття файлів. Інтегровані тести можуть проводитися або розробниками, або незалежними тестувальниками і зазвичай складаються з комбінації автоматизованих функціональних та ручних тестів.

Тестування системи

Системне тестування – це метод тестування «чорної скриньки», який використовується для оцінки завершеної та інтегрованої системи в цілому, щоб переконатися, що вона відповідає заданим вимогам. p align="justify"> Функціональність програмного забезпечення тестується від початку до кінця і зазвичай проводиться окремою групою тестування, а не командою розробників, перш ніж продукт буде запущений у виробництво.

Приймальне тестування

Приймальний тест є останньою фазою функціонального тестування і використовується для оцінки того, чи готова остаточна частина програмного забезпечення до доставки. Це включає забезпечення того, щоб продукт відповідав усім початковим бізнес-критеріям і задовольняв потреби кінцевого користувача. Це вимагає, щоб продукт був протестований як усередині, так і зовні, а це означає, що вам потрібно буде передати його в руки вашим кінцевим

користувачам для тестування бета разом з вашою командою контролю якості. Бета тестування є ключем до отримання реального зворотного зв'язку від потенційних клієнтів і може вирішити будь-які остаточні проблеми зі зручністю використання.

Тестування продуктивності

Тестування продуктивності — це метод дисфункції тестування, який використовується для визначення того, як додаток поводитиметься в різних умовах. Мета полягає в тому, щоб перевірити його чуйність і стабільність у реальних користувачах ситуаціях. Тестування продуктивності можна розділити на чотири типи:

— Навантажувальне тестування — це процес збільшення кількості змодельованих вимог до вашого програмного забезпечення, додатку або веб-сайту, щоб перевірити, чи воно може впоратися з тим, для чого воно призначене.

— Стрес-тестування робить ще один крок вперед і використовується для оцінки того, як ваше програмне забезпечення буде реагувати на пікове навантаження або за його межами. Метою стрес-тестування є навмисне перевантаження програми до тих пір, поки воно не зламається, застосовуючи як реалістичні, так і нереалістичні сценарії навантаження. За допомогою стрес-тестування ви зможете знайти точку відмови вашого програмного забезпечення.

— Тестування на витривалість, також відоме як тестування на витримку, використовується для аналізу поведінки програми при певному навантаженні, що моделюється, протягом більш тривалого періоду часу. Мета полягає в тому, щоб зрозуміти, як ваша система поводитиметься при тривалому використанні, що робить цей процес більш тривалим, ніж навантаження або стрес-тестування (які розраховані на завершення через кілька годин). Найважливішою частиною тестування на витривалість і те, що допомагає виявити витоку пам'яті.

— Стрибкове тестування — це тип тесту навантаження, який використовується для визначення того, як ваше програмне забезпечення буде реагувати на значно більші сплески одночасних дій користувачів або системи протягом різних періодів часу. В ідеалі це допоможе зрозуміти, що станеться, коли навантаження різко і різко зросте.

Тестування безпеки

З появою хмарних платформ тестування та кібератак зростає занепокоєння та потреба у безпеці даних, що використовуються та зберігаються у програмному забезпеченні. Тестування безпеки — це нефункціональний метод тестування програмного забезпечення, який використовується для визначення того, чи захищена інформація та дані в системі. Мета полягає в тому, щоб цілеспрямовано знаходити лазівки та загрози безпеці в системі, які можуть призвести до несанкціонованого доступу або втрати інформації шляхом перевірки додатка на наявність слабких місць. Існує кілька типів цього методу тестування, кожен із яких спрямований на перевірку шести основних принципів безпеки:

— Цілісність
— Конфіденційність
— Аутентифікація
— Авторизація
— Доступність
— Безвідмовність

Юзабіліті-тестування

Юзабіліті-тестування - це метод тестування, який вимірює простоту використання програми з точки зору кінцевого користувача і часто виконується на етапах системного або приймального тестування. Ціль полягає в тому, щоб визначити, чи відповідають видимий дизайн і зовнішній вигляд програми передбачуваному робочому процесу для різних процесів, таких як вхід до

додатка. Юзабіліті-тестування — чудовий спосіб для команд перевірити окремі функції чи систему загалом, інтуїтивно зрозумілу у використанні.

Тестування сумісності

Тестування сумісності використовується для оцінки того, як програма або частина програмного забезпечення працюватиме в різних середовищах. Він використовується для перевірки сумісності вашого продукту з кількома операційними системами, платформами, браузерами або конфігураціями роздільної здатності. Ціль полягає в тому, щоб гарантувати, що функціональність вашого програмного забезпечення послідовно підтримується в будь-якому середовищі, яке, як ви очікуєте, будуть використовувати ваші кінцеві користувачі.

4.2.1 Тестування модуля BME680

Для тестування доцільно створити окремий проект у середовищі розробки Arduino IDE. При створенні скетчу достатньо додати наступні бібліотеки:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"
```

Далі пишемо функцію запуску датчика

```
void setup() {
  Serial.begin(9600);
  while (!Serial);
  Serial.println(F("BME680 test"));
```

При цьому створюємо вивід на випадок невдалої ініціалізації датчика

```
  if (!bme.begin()) {
    Serial.println("Could not find a valid BME680 sensor, check wiring!");
    while (1);
  }
```

Після цього ініціалізуємо передискретизацію та фільтри

```
bme.setTemperatureOversampling(BME680_OS_8X);
```

```
bme.setHumidityOversampling(BME680_OS_2X);
bme.setPressureOversampling(BME680_OS_4X);
bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
bme.setGasHeater(320, 150);
```

Після цього запускаємо процес перевірки справності функції зчитування показників `bme.performReading()`

У разі належної роботи датчика через монітор порту буде виведено дані температури, тиску, вологості та якості повітря (рис.4.3)

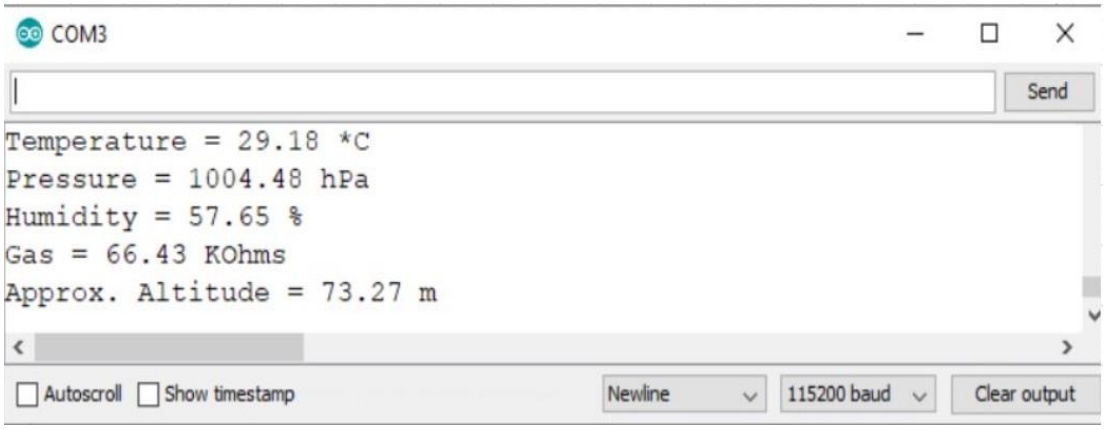


Рисунок 4.3. Вивід даних через монітор порта в середовищі Arduino IDE

4.2.2 Тестування роботи протоколу MQTT

Тестування реалізацій MQTT є важливим для забезпечення їх надійності, безпеки та функціональної сумісності. Ось кілька найпоширеніших методів тестування протоколів MQTT:

— **Модульне тестування**: це включає ізольоване тестування окремих компонентів реалізації MQTT, таких як брокер повідомлень, клієнтська бібліотека або протокол зв'язку. Модульне тестування зазвичай включає написання тестів на рівні коду, які можна автоматизувати і швидко виконати.

— **Інтеграційне тестування**: це включає тестування того, як різні компоненти реалізації MQTT працюють разом, наприклад, тестування того, як клієнт взаємодіє з брокером повідомлень. Інтеграційне тестування зазви-

чай включає використання тестового набору або інструменту моделювання, який емулює мережеве середовище MQTT.

- Тестування продуктивності: включає тестування реалізації MQTT на масштабованість, надійність та час відгуку за різних умов навантаження. Тестування продуктивності зазвичай включає використання інструментів тестування навантаження, які можуть імітувати велику кількість клієнтів і повідомлень.
- **Тестування безпеки**: це включає тестування реалізації MQTT на наявність уразливостей безпеки, таких як несанкціонований доступ або витік даних. Тестування безпеки зазвичай включає використання інструментів та методів тестування на проникнення для виявлення та використання слабких місць у мережі MQTT.
- **Тестування функціональної сумісності**: це включає в себе тестування реалізації MQTT на сумісність з іншими реалізаціями MQTT, такими як різні брокери повідомлень або клієнтські бібліотеки. Тестування функціональної сумісності зазвичай включає використання стандартизованих наборів тестів та інструментів, щоб гарантувати, що різні реалізації можуть взаємодіяти один з одним.

В цілому, тестування реалізацій MQTT є важливою частиною забезпечення їхньої надійності, безпеки та функціональної сумісності. Використовуючи комбінацію цих методів тестування, розробники можуть виявляти та усувати проблеми до того, як вони стануть критичними, та гарантувати, що їх реалізації MQTT відповідають необхідним стандартам продуктивності та безпеки.

Для тестування належної роботи протоколу потрібно створити новий, тестовий топік, та підписатись на нього:

```
mosquitto_sub -d -t testTopic
```

Результатом буде підписка на даний топік (рис.4.4)

```
pi@raspberrypi:~ $ mosquitto_sub -d -t testTopic
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending SUBSCRIBE (Mid: 1, Topic: testTopic, QoS: 0, Options: 0x00
)
Client (null) received SUBACK
Subscribed (mid: 1): 0
```

Рисунок 4.4. Підписка на тестовий топік

За замовчуванням, повідомлення надсилаються та отримуються за QoS 0.

Після підписки треба запустити друге вікно терміналу та ввести тестове повідомлення, яке повинне відображатися у інших користувачів.

```
mosquitto_pub -d -t testTopic -m "Hello world!"
```

У вікні терміналу №2 видно, що почався процес передачі повідомлення (рис. 4.5).

```
Client mosqpub/868-raspberrypi sending CONNECT
Client mosqpub/868-raspberrypi received CONNACK
Client mosqpub/868-raspberrypi sending PUBLISH (d0, q0, r0, m1, 'testTopic', ...
tes))
Client mosqpub/868-raspberrypi sending DISCONNECT
```

Рисунок 4.5. Вікно №2. Відправлення повідомлення

В той же час, через вікно №1 видно, що доставка повідомлення пройшла успішно (рис.4.6).

```
pi@raspberrypi:~ $ mosquitto_sub -d -t testTopic
Client mosqsub/867-raspberrypi sending CONNECT
Client mosqsub/867-raspberrypi received CONNACK
Client mosqsub/867-raspberrypi sending SUBSCRIBE (Mid: 1, Topic: testTopic, QoS: 0)
Client mosqsub/867-raspberrypi received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/867-raspberrypi received PUBLISH (d0, q0, r0, m0, 'testTopic', ... (
ytes))
Hello world!
```

Рисунок 4.6. Вікно №1. Отримання повідомлення

Для перевірки можливості доставки повідомлення декільком клієнтам запускаємо ще один термінал, в якому вводимо команду для підписки на топік.

У другому вікні вводимо команду на підписку до топіка та надсилання повідомлення (рис.4.7).

```
pi@raspberrypi:~ $ mosquitto_pub -d -t testTopic -m "Hello world!"
Client mosqpub/920-raspberrypi sending CONNECT
Client mosqpub/920-raspberrypi received CONNACK
Client mosqpub/920-raspberrypi sending PUBLISH (d0, q0, r0, m1, 'testTopic', ...
tes))
Client mosqpub/920-raspberrypi sending DISCONNECT
pi@raspberrypi:~ $ mosquitto_pub -d -t testTopic -m "Hello world!"
Client mosqpub/922-raspberrypi sending CONNECT
Client mosqpub/922-raspberrypi received CONNACK
Client mosqpub/922-raspberrypi sending PUBLISH (d0, q0, r0, m1, 'testTopic', ...
tes))
Client mosqpub/922-raspberrypi sending DISCONNECT
```

Рисунок 4.7. Вікно №2. Надсилання повідомлення

У першому (рис.4.8) та третьому (рис.4.9) терміналах відображаються отримані повідомлення. Отже, передача повідомлень за протоколом MQTT пройшла успішно.

```
Client mosqsub/919-raspberrypi sending CONNECT
Client mosqsub/919-raspberrypi received CONNACK
Client mosqsub/919-raspberrypi sending SUBSCRIBE (Mid: 1, Topic: testTopic, QoS: 0)
Client mosqsub/919-raspberrypi received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/919-raspberrypi received PUBLISH (d0, q0, r0, m0, 'testTopic', ... (
ytes))
Hello world!
Client mosqsub/919-raspberrypi received PUBLISH (d0, q0, r0, m0, 'testTopic', ... (
ytes))
Hello world!
```

Рисунок 4.8. Вікно №1. Лог отримання повідомлення

```
pi@raspberrypi:~ $ mosquitto_sub -d -t testTopic
Client mosqsub/921-raspberrypi sending CONNECT
Client mosqsub/921-raspberrypi received CONNACK
Client mosqsub/921-raspberrypi sending SUBSCRIBE (Mid: 1, Topic: testTopic, QoS: 0)
Client mosqsub/921-raspberrypi received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/921-raspberrypi received PUBLISH (d0, q0, r0, m0, 'testTopic', ... (
ytes))
Hello world!
```

Рисунок 4.9. Вікно №3. Лог отримання повідомлення

Висновки до розділу 4

У четвертому розділі було представлено результати роботи спроектованого апаратно-програмного комплексу. За результатами проведеної роботи, реалізовано:

— взаємодію компонентів за протоколом MQTT

— передачу даних на сервер

— відображення даних у зручному для користувача вигляді

— збереження даних на сервері

Також розглянуто методи та засоби тестування програмного забезпечення. Проведено тестування окремих компонентів комплексу на предмет їх працездатності. У ході тестування виявлено, що всі елементи мережі працюють у штатному режимі.

ВИСНОВКИ

Перед початком роботи над реалізацією апаратно-програмного комплексу системи доставки контенту на базі сенсорної IoT- мережі було здійснено аналіз інформації стосовно пристроїв передачі та збереження даних, а саме:

- Розглянуто та проаналізовано інформацію, пов’язану з теоретичною базою використання одноплатних комп’ютерів.
- Досліджено шляхи передачі даних у кластерних сенсорних мережах та пристроях IoT.
- Проаналізовано шляхи створення пристрою , врахувавши недоліки попередніх/наявних розробок.
- Спроектовано апаратно-програмний комплекс , який складається з серверної частини та сенсорної мережі, яка за протоколом MQTT передає дані на сервер
- Створено розподілену систему доставки контенту на основі доступних компонентів з автономним живленням з використанням сучасних IoT технологій.

Після завершення процесів розробки розглянуто методи тестування програмно-апаратного комплексу , проведено більш глибокий аналіз методів тестування наявних компонентів комплексу та проведено відповідні тестування окремих елементів на предмет їх належної роботи

Результати дослідження та розробки програмно-апаратного комплексу доставки контенту на базі сенсорної IoT - мережі дозволять використовувати дану систему в умовах обмеженого енергопостачання та за його відсутності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кравцов К. В., Пузиреьв С. В. Кластерна система доставки контенту (CDN) на базі Raspberry Pi. Інформаційні технології та інженерія : тези доп. Всеукр. наук.-практ. конф. Миколаїв, 07–10 лют. 2023 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2023. С. 74–76.
2. A.Shostak. Estimation of coverage factor of a wireless sensor network, *Advanced Information Systems*, p.74-77, 2018 (Дата звернення: 21.01.2023)
3. V.Abramov. PROTECTED CABLE INTERFACES OF THE SENSORY NETWORKS OF THINGS INTERNET, *Cybersecurity: Education, Science, Technique*, p.73-85, 2019 (Дата звернення: 21.01.2023)
4. Download Raspbian for Raspberry Pi – URL: https://www.raspberrypi.org/downloads/raspbian/ – Дата звернення: 22.01.2023.
5. A.Karpenko, T.Bondarenko, V.Ovsiannykov et al. ENSURING INFORMATION SECURITY IN WIRELESS SENSOR NETWORKS, *Cybersecurity: Education, Science, Technique*, p.54-66, 2020 (Дата звернення: 21.01.2023)
6. V.Terokhin, M.Stervoyedov, O.Ridozub. Application Of The IoT Technology and Cloud Services for Radiation Monitoring, *Control Systems and Computers*, p.60-68, 2021 (Дата звернення: 21.01.2023)
7. О.Лаврів, О.Шпур, Н.Пелех. Забезпечення доступу до статичного контенту із використанням CDN мереж як PaaS сервісу Azure Cloud, *Системи обробки інформації*, с.83-91, 2019 (Дата звернення: 26.01.2023)
8. R.Chernenko, O.Riabchun, M.Vorokhob et al. INCREASING THE LEVEL OF SECURITY OF INTERNET THINGS NETWORK SYSTEMS DUE TO ENCRYPTION OF DATA ON DEVICES WITH LIMITED COMPUTER SYSTEMS, *Cybersecurity: Education, Science, Technique*, p.124-135, 2021 (Дата звернення: 03.02.2023)

9. А.Петросян, Р.Петросян, К.Колос. Розробка платформи віддаленого управління інфраструктурою Інтернет речей, *Технічна інженерія*, с.73-80, 2021 (Дата звернення: 05.02.2023)
10. M.Widjaja, D.K.Halim, R.Andarini. The Development of an IoT-based Indoor Air Monitoring System Towards Smart Energy Efficient Classroom, *Ultima Computing*, p.28-35, 2022 (Дата звернення: 07.02.2023)
11. M.Caratu, V.Gallo, V.Paciello. IEEE 1451: Communication among smart sensors using MQTT protocol, *2022 IEEE International Symposium on Measurements & Networking (M&N)*, p.105-117, 2022 (Дата звернення: 07.02.2023)
12. L.Hao, X.Yu, T.Zhang. Distributed MQTT Brokers at Network Edges: A Study on Message Dissemination, 2022 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), p.17-24, 2022 (Дата звернення: 09.02.2023)
13. I.Sukma, W.Ardiatna, N.Novitasari, Real-time wireless temperature measurement system of an infant incubator. *International Journal of Electrical and Computer Engineering*,p.13-14, 2023 (Дата звернення: 11.02.2023)
14. А.Карпилов. Засіб автоматизації контролю параметрів електроенергетичних систем. *Automatization of technological and business processes,* с.54-58 (Дата звернення: 15.02.2023)
15. MQTT Essentials [Електронний ресурс].- Режим доступу: https://www.hivemq.com/tags/mqtt-essentials/
16. Quality of Service levels [Електронний ресурс].- Режим доступу: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/
17. Пат. на корисну модель CN. Pat. № CN110807905B Community fire monitoring system based on end-edge-cloud architecture / Chun Ho. Ching, Shao Jun-Tzu; заявл. 23.10.2019, опубл. 12.12.2019.

18. Пат. на корисну модель US. Pat. № US10075353B2 Sensor network management/ Gregory Bullard,Desiree Brake,Christopher Pruetting,Raymond Stits,Jason Tory Ryberg,Jason Thomas,Diane Winchell; заявл. 11.09.2018, опубл. 12.01.2019
19. Пат. на корисну модель US. Pat. № US9800683B2 Bandwidth policy management in a self-corrected content delivery network/ Chris Phillips, Jimmie Rodgers; заявл. 24.10.2019, опубл. 17.11.2019
20. ESPlorer – Integrated Development Environment for ESP8266 developers[Електронний ресурс]. – Режим доступу: https://esp8266.en/esplorer/
21. Arduino IDE [Електронний ресурс].- Режим доступу: https://www.arduino.cc/en/software
22. Wemos D1 Mini board[Електронний ресурс].- Режим доступу: https://www.openimpulse.com/blog/products-page/product-category/wemos-d1-mini-esp8266-development-board/
23. BME680 Gas sensor [Електронний ресурс].- Режим доступу: https://www.bosch-sensortec.com/products/environmental-sensors/gas-sensors/bme680/
24. MQTT vs AMQP in IoT [Електронний ресурс].- Режим доступу: https://www.hivemq.com/blog/mqtt-vs-amqp-for-iot/
25. Hackboard 2 [Електронний ресурс]. -Режим доступу: https://www.crowdsupply.com/hackboard/hb2
26. RockPI datasheet [Електронний ресурс].- Режим доступу: https://dl.radxa.com/rockpi/docs/hw/rockpi4/rockpi4_datasheet_v1.1.pdf
27. AsyncMQTT Library [Електронний ресурс].- Режим доступу: https://forum.arduino.cc/t/asyncmqtt-generic-library/

Додаток А

Код програми

App.py

```
import paho.mqtt.client as mqtt
from flask import Flask, render_template, request
import json
import sqlite3
app = Flask(__name__)
def dict_factory(cursor, row):
    d = {}
    for idx, col in enumerate(cursor.description):
        d[col[0]] = row[idx]
    return d
def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe("/wemos/sensor")
def on_message(client, userdata, message):
    if message.topic == "/wemos/sensor":
        print("Sensor reading update")
        #print(message.payload.json())
        #print(sensor_json['temp'])
        #print(sensor_json['hum'])
       #print(sensor_json['press'])
        #print(sensor_json['gas'])
        sensor_json = json.loads(message.payload)
        conn=sqlite3.connect('data.db')
        c=conn.cursor()
        c.execute("""INSERT INTO sensor (temperature,
            humidity, pressure, gas, currentdate, currentime,
device) VALUES((?), (?), date('now'),
            time('now'), (?))""", (sensor_json['temperature'],
    sensor_json['humidity'], sensor_json['pressure'],
```

```
     sensor_json['gas'],   'wemos1dmini') )
            conn.commit()
            conn.close()
     mqttc=mqtt.Client()
     mqttc.on_connect = on_connect
     mqttc.on_message = on_message
     mqttc.connect("localhost",1883,60)
     mqttc.loop_start()
     @app.route("/")
        conn=sqlite3.connect('data.db')
        conn.row_factory = dict_factory
        c=conn.cursor()
        c.execute("SELECT * FROM sensor ORDER BY id DESC LIMIT 20")
        readings = c.fetchall()
        #print(readings)
        return render_template('main.html', readings=readings)
     if __name__ == "__main__":
        app.run(host='0.0.0.0', port=8181, debug=True)
```

Файл тестування датчика BME680

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME680 bme; // I2C

void setup() {
  Serial.begin(9600);
  while (!Serial);
  Serial.println(F("BME680 test"));

  if (!bme.begin()) {
    Serial.println("Could not find a valid BME680 sensor, check
wiring!");
    while (1);
  }

  // Set up oversampling and filter initialization
  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
```

```
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
  bme.setGasHeater(320, 150); // 320*C for 150 ms
}

void loop() {
  if (! bme.performReading()) {
    Serial.println("Failed to perform reading :(");
    return;
  }
  Serial.print("Temperature = ");
  Serial.print(bme.temperature);
  Serial.println(" *C");

  Serial.print("Pressure = ");
  Serial.print(bme.pressure / 100.0);
  Serial.println(" hPa");

  Serial.print("Humidity = ");
  Serial.print(bme.humidity);
  Serial.println(" %");

  Serial.print("Gas = ");
  Serial.print(bme.gas_resistance / 1000.0);
  Serial.println(" KOhms");

  Serial.print("Approx. Altitude = ");
  Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  Serial.println(" m");

  Serial.println();
  delay(2000);
}
```

Файл прошивки для Wemos D1 Mini

```
#include <Ticker.h>

#include <AsyncMqttClient.h>

#include <Wire.h>

#include <SPI.h>

#include <Adafruit_Sensor.h>

#include "Adafruit_BME680.h"

#define WIFI_SSID "REPLACE_WITH_YOUR_SSID"

#define WIFI_PASSWORD "REPLACE_WITH_YOUR_PASSWORD"

#define MQTT_HOST IPAddress(192, 168, 1, XXX)
```

```
//#define MQTT_HOST "example.com"
#define MQTT_PORT 1883
#define MQTT_PUB_TEMP "esp/bme680/temperature"
#define MQTT_PUB_HUM  "esp/bme680/humidity"
#define MQTT_PUB_PRES "esp/bme680/pressure"
#define MQTT_PUB_GAS  "esp/bme680/gas"
/*#define BME_SCK 14
#define BME_MISO 12
#define BME_MOSI 13
#define BME_CS 15*/
Adafruit_BME680 bme; // I2C
//Adafruit_BME680 bme(BME_CS); // hardware SPI
//Adafruit_BME680 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK);
float temperature;
float humidity;
float pressure;
float gasResistance;
AsyncMqttClient mqttClient;
Ticker mqttReconnectTimer;
WiFiEventHandler wifiConnectHandler;
WiFiEventHandler wifiDisconnectHandler;
Ticker wifiReconnectTimer;
unsigned long previousMillis = 0;
const long interval = 10000;
void getBME680Readings(){
  unsigned long endTime = bme.beginReading();
  if (endTime == 0) {
    Serial.println(F("Failed to begin reading :("));
```

```
    return;
  }
  if (!bme.endReading()) {
    Serial.println(F("Failed to complete reading :("));
    return;
  }
  temperature = bme.temperature;
  pressure = bme.pressure / 100.0;
  humidity = bme.humidity;
  gasResistance = bme.gas_resistance / 1000.0;
}
void connectToWifi() {
  Serial.println("Connecting to Wi-Fi...");
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
}
void onWifiConnect(const WiFiEventStationModeGotIP& event) {
  Serial.println("Connected to Wi-Fi.");
  connectToMqtt();
}
void onWifiDisconnect(const WiFiEventStationModeDisconnected& event) {
  Serial.println("Disconnected from Wi-Fi.");
  mqttReconnectTimer.detach(); // ensure we don't reconnect to MQTT while
reconnecting to Wi-Fi
  wifiReconnectTimer.once(2, connectToWifi);
}


void connectToMqtt() {
  Serial.println("Connecting to MQTT...");
```

```
  mqttClient.connect();
}
void onMqttConnect(bool sessionPresent) {
  Serial.println("Connected to MQTT.");
  Serial.print("Session present: ");
  Serial.println(sessionPresent);
}
void onMqttDisconnect(AsyncMqttClientDisconnectReason reason) {
  Serial.println("Disconnected from MQTT.");
  if (WiFi.isConnected()) {
    mqttReconnectTimer.once(2, connectToMqtt);
  }
}
/*void onMqttSubscribe(uint16_t packetId, uint8_t qos) {
  Serial.println("Subscribe acknowledged.");
  Serial.print("  packetId: ");
  Serial.println(packetId);
  Serial.print("  qos: ");
  Serial.println(qos);
}
void onMqttUnsubscribe(uint16_t packetId) {
  Serial.println("Unsubscribe acknowledged.");
  Serial.print("  packetId: ");
  Serial.println(packetId);
}*/
void onMqttPublish(uint16_t packetId) {
  Serial.print("Publish acknowledged.");
  Serial.print("  packetId: ");
```

```
  Serial.println(packetId);

}

void setup() {

  Serial.begin(115200);

  Serial.println();



  if (!bme.begin()) {

    Serial.println(F("Could not find a valid BME680 sensor, check
wiring!"));

    while (1);

  }

  wifiConnectHandler = WiFi.onStationModeGotIP(onWifiConnect);

  wifiDisconnectHandler =
WiFi.onStationModeDisconnected(onWifiDisconnect);

  mqttClient.onConnect(onMqttConnect);

  mqttClient.onDisconnect(onMqttDisconnect);

  //mqttClient.onSubscribe(onMqttSubscribe);

  //mqttClient.onUnsubscribe(onMqttUnsubscribe);

  mqttClient.onPublish(onMqttPublish);

  mqttClient.setServer(MQTT_HOST, MQTT_PORT);

  // If your broker requires authentication (username and password), set
them below

  //mqttClient.setCredentials("REPlACE_WITH_YOUR_USER",
"REPLACE_WITH_YOUR_PASSWORD");

  connectToWifi();

  bme.setTemp
```