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

« __ « _____ 2024 р.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Система обміну короткими повідомленнями на базі ESP8266 та MQTT

Спеціальність 123 Комп’ютерна інженерія

123 – КБР.01 – 405.22010610

Студент _____ І. М. Парфьонов
підпис
«__» _____ 202__ р.

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

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Зав. кафедри _____ І. М. Журавська

«_____» _____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної бакалаврської роботи

Видано студенту групи 405 факультету комп'ютерних наук

_____Парфьонов Ілля Миколайович_____
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Система обміну короткими повідомленнями на базі ESP8266 та MQTT

_____

Затверджена наказом по ЧНУ ім. Петра Могили від 30.01.2024 № 17.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Функціональна система обміну повідомленнями на базі ESP8266 з використанням протоколу MQTT, здатна передавати дані між пристроями через Інтернет. Система повинна бути налаштована таким чином, щоб забезпечити надійну комунікацію та можливість керування пристроями з використанням MQTT брокера.
_____

4. Перелік питань, що підлягають розробці
_Аналіз предметної області та постановка задачі. Огляд існуючих рішень системи обміну короткими повідомленнями. Проєктування системи обміну короткими повідомленнями на базі ESP8266 та MQTT. Реалізація програмного та апаратного забезпечення для системи обміну короткими повідомленнями на базі ESP8266 та MQTT. Розробка алгоритму для прошивки NodeMCU та реалізація коротких повідомлень з модулем BME680.____

5. Перелік графічних матеріалів
Діаграма структури системи в цілому
Блок-схема роботи на стороні ESP8266
Блок-схема роботи на стороні клієнтського додатку
Схема підключення
Блок-схема алгоритму роботи прошивки
Загальна схема пристрою

6. Завдання до спеціальної частини
_Проаналізувати основні ризики роботи з електронними компонентами. Описати основні заходи безпеки. Дослідити охорону праці при розробці програмного забезпечення для системи обміну повідомленнями. Надати рекомендації щодо безпеки експлуатації та технічного обслуговування системи на базі ESP8266.

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А. О. канд. техн. наук, доцент	кафедра екології Медичного інституту ЧНУ імені Петра Могили	Спеціальна частина з охорони праці

Керівник роботи
___доц., канд. фіз-мат. наук Пузирьов Сергій Володимирович
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

_____Парфьонов Ілля Миколайович
(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: _Система обміну короткими повідомленнями на базі ESP8266 та MQTT

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КР	11.12.23	12.12.23	Виконав
2	Огляд літератури за темою роботи	15.01.24	18.02.24	Виконав
3	Складання календарного плану БКР	19.02.24	06.03.24	Виконав
4	Аналіз предметної області	19.02.24	04.03.24	Виконав
5	Розробка проєктних рішень	23.02.24	09.03.24	Виконав
6	Моделювання та конструювання АПЗ	20.02.24	27.02.24	Виконав
7	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування	01.03.24	04.03.24	Виконав
8	Відгук керівника КР	09.03.24	07.04.24	Виконав
9	Оформлення БКР та презентації	15.03.24	21.04.24	Виконав
10	Попередній захист	13.06.24	13.06.24	Виконав
11	Рецензування	12.05.24	20.05.24	Виконав
12	Завершення оформлення КР та презентації	30.05.24	30.05.24	Виконав
13	Захист бакалаврської кваліфікаційної роботи			

Розробив здобувач ВО _Парфьонов Ілля Миколайович _____

(прізвище, ім'я, по батькові) *(підпис)*

«____» _____ 20__ р.

Керівник роботи _Пузирьов Сергій Володимирович _____

(посада, прізвище, ім'я, по батькові) *(підпис)*

«____» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«Система обміну короткими повідомленнями на базі ESP8266 та MQTT»
Студент 405 гр.: Парфьонов Ілля Миколайович
Керівник: доц., канд. фіз-мат. наук Пузирьов Сергій Володимирович

Ця робота присвячена розробці прототипу розподіленої системи обміну короткими повідомленнями з використанням мікроконтролера ESP8266 та протоколу MQTT. В контексті стрімкого розвитку Інтернету речей (IoT) та доступності бездротових технологій з'єднання, системи обміну повідомленнями займають ключову роль у забезпеченні ефективної взаємодії між пристроями.

Дослідження включає аналіз технічних аспектів ESP8266 та MQTT, розробку методів оптимізації системи, вивчення питань безпеки та інших ключових аспектів, що впливають на надійність та ефективність обміну даними в IoT.

Результати дослідження спрямовані на практичне застосування у розробці інноваційних IoT-проектів, надаючи фахівцям засоби для оптимального використання ESP8266 та MQTT з максимальною продуктивністю та забезпечуючи необхідний рівень безпеки та надійності систем обміну короткими повідомленнями.

Пояснювальна записка бакалаврської роботи складається зі вступу, трьох розділів, висновків та трьох додатків. У вступі визначається актуальність теми, сформульовані мета, об’єкт, предмет та завдання дослідження та розроблення бакалаврської роботи. У першому розділі проведено предметний огляд та існуючі рішення. У другому розділі описано детально архітектуру системи, протокол, обробка помилок, програмне та апаратне забезпеченню, з фокусом на протокол MQTT. У третьому розділі роботи розглянуті наступні етапи: реалізація на базі ESP8266 та MQTT, розробка програмного забезпечення, вивчення алгоритмів для прошивки NodeMCU та впровадження коротких повідомлень з використанням модуля BME680. У висновках наведено аналіз виконаної роботи та отриманих результатів дослідження та розроблення. У додатку А наведено Довідку з перевірки на унікальність. У додатку Б – код для підключення ESP8266 до MQTT-сервера і роботи з датчиком Adafruit BME680. У додатку В – Wi-Fi MQTT клієнт на ESP8266.

В цілому, бакалаврська робота без додатків містить 81 с., 46 рис., 2 табл., 23 джерела посилання.

Ключові слова: *ESP8266, MQTT, IoT, обмін повідомленнями, бездротове з'єднання, надійність, ефективність, безпека.*

ABSTRACT

of the Bachelor’s Thesis

«Short messaging system based on ESP8266 and MQTT»

Student: Parfenov Ilya Nikolaevich

Supervisor: Associate Professor, Candidate of Physical and Mathematical Sciences Puzyrev Sergey Vladimirovich

This work is devoted to the development of a prototype of a distributed short messaging system using the ESP8266 microcontroller and the MQTT protocol. In the context of the rapid development of the Internet of Things (IoT) and the availability of wireless communication technologies, messaging systems play a key role in ensuring effective interaction between devices.

The research includes analysis of the technical aspects of ESP8266 and MQTT, development of system optimization methods, study of security issues and other key aspects affecting the reliability and efficiency of data exchange in IoT.

The results of the research are aimed at practical application in the development of innovative IoT projects, providing specialists with the means to optimize the use of ESP8266 and MQTT with maximum performance and ensuring the required level of security and reliability of short message exchange systems.

The explanatory note of the bachelor's thesis consists of an introduction, three chapters, conclusions, and three appendices. The introduction defines the relevance of the topic, formulates the purpose, object, subject, and objectives of the research and development of the bachelor's thesis. The first chapter provides a substantive overview and existing solutions. The second chapter describes in detail the system architecture, protocol, error handling, software, and hardware, with a focus on the MQTT protocol. The third section of the paper discusses the following stages: implementation based on ESP8266 and MQTT, software development, study of algorithms for NodeMCU firmware, and implementation of short messages using the BME680 module. The conclusions provide an analysis of the work performed and the results of research and development. Appendix A contains the Uniqueness Checker Reference. Appendix B contains the code for connecting ESP8266 to the MQTT server and working with the Adafruit BME680 sensor. Appendix C - Wi-Fi MQTT client on ESP8266.

In total, the bachelor's thesis without appendices contains 81 pages, 46 figures, 2 tables, 23 references.

Keywords: ESP8266, MQTT, IoT, message exchange, wireless connection, reliability, efficiency, security.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

IoT – Інтернет речей
АПЗ – Архітектурно-планувальне завдання

ACL – Access Control List
ADC – Analog-to-Digital Converter
API – Application Programming Interface
AWS – Amazon Web Services
ESP – Embedded System Platform
HTTP – Hypertext Transfer Protocol
IDE – Integrated Development Environment
IP – Internet Protocol
LCD – Liquid Crystal Display
MAC – Media Access Control
MQTT – Message Queuing Telemetry Transport
OLED – Organic Light-Emitting Diode
OTA – Over-the-Air
PWM – Pulse Width Modulation
QoS – Quality of Service
RAM – Random Access Memory
SOMP - System-on-Module Platform
SSID – Service Set Identifier
SSL – Secure Sockets Layer
SSL – Secure Sockets Layer
TLS – Transport Layer Security
UART – Universal Asynchronous Receiver/Transmitter
URL – Uniform Resource Locator
WAF – Web application firewall
WPA – Wi-Fi Protected Access

ВСТУП

У сучасному світі, де Інтернет речей (IoT) стрімко набирає обертів, а бездротові технології з'єднання стають все більш доступними, системи обміну короткими повідомленнями відіграють ключову роль у забезпеченні ефективної взаємодії між різноманітними пристроями. Мікроконтролер ESP8266, завдяки своїй низькій вартості, широкому функціоналу та підтримці Wi-Fi, став одним з найпопулярніших рішень для розробки IoT-пристроїв. У поєднанні з протоколом MQTT (Message Queuing Telemetry Transport), який забезпечує легкий та надійний обмін даними, ESP8266 відкриває нові можливості для створення інноваційних систем обміну повідомленнями.

Метою даної роботи є розробка прототипу розподіленої системи обміну короткими повідомленнями на базі мікроконтролера ESP8266 та протоколу MQTT. Це дослідження спрямоване на глибоке вивчення технічних аспектів ESP8266 та MQTT, аналіз їх переваг та обмежень, а також розробку ефективних методів та рекомендацій для їх практичного застосування.

Об'єктом дослідження є система обміну короткими повідомленнями, що включає в себе як апаратну частину (мікроконтролер ESP8266), так і програмну (протокол MQTT). Дослідження охоплює вивчення архітектури системи, процесів обміну повідомленнями, питань безпеки та інших аспектів, що впливають на ефективність та надійність системи.

Предметом дослідження є процеси передачі та прийому повідомлень у розподілених системах зв'язку ESP8266, особливості протоколу MQTT, а також аналіз їх взаємодії та ефективності в системах обміну короткими повідомленнями. Особлива увага приділяється вивченню методів оптимізації та налаштування системи для досягнення максимальної продуктивності та надійності.

Результати даного дослідження мають важливе практичне значення для широкого кола фахівців, включаючи розробників IoT-систем, інженерів, дослідників та всіх, хто цікавиться сучасними технологіями обміну даними.

Розроблені методи та рекомендації дозволять оптимізувати використання ESP8266 та MQTT у практичних застосуваннях, забезпечуючи високу ефективність, надійність та безпеку систем обміну короткими повідомленнями в мережах IoT.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ СИСТЕМИ ОБМІНУ КОРОТКИМИ ПОВІДОМЛЕННЯМИ

1.1 Аналіз предметної області та постановка задач

У цьому розділі буде проведено детальний аналіз існуючих рішень для систем обміну короткими повідомленнями, побудованих на базі мікроконтролерів ESP8266 з використанням протоколу MQTT. Такий аналіз є фундаментальним етапом у розробці нової, ефективної системи, оскільки дозволяє зрозуміти, які технології, архітектури та підходи забезпечують оптимальну продуктивність, надійність та зручність використання. Аналіз проблем та недоліків існуючих рішень дозволяє уникнути повторення цих помилок у новій системі, підвищуючи її якість та стабільність. На основі аналізу можна сформулювати чіткі та обґрунтовані вимоги до апаратно-програмного забезпечення нової SOMP, що забезпечить її відповідність потребам користувачів та ефективну інтеграцію з існуючою інфраструктурою.

Детальне вивчення різноманітних SOMP на базі ESP8266 та MQTT, включаючи як комерційні, так і проєкти з відкритим вихідним кодом. Аналіз різних архітектур, протоколів, бібліотек та інтерфейсів користувача, що використовуються в існуючих системах. Оцінка сильних та слабких сторін кожного рішення з точки зору функціональності, продуктивності, безпеки, масштабованості та зручності використання. Визначення ключових вимог до апаратно-програмного забезпечення нової системи на основі результатів аналізу, включаючи вимоги до функціональності, продуктивності, безпеки, сумісності та зручності використання.

Цілі та завдання розділу включають проведення всебічного аналізу існуючих рішень для обміну короткими повідомленнями на базі ESP8266 та MQTT, визначити ключові технічні та функціональні характеристики цих рішень та сформулювати чіткі вимоги до апаратно-програмного забезпечення нової системи.

Ключові завдання:

1) дослідити та зібрати інформацію про різноманітні системи обміну короткими повідомленнями на базі ESP8266 та MQTT;

2) провести порівняльний аналіз функціональності, продуктивності, безпеки та зручності використання різних рішень;

3) виявити найкращі практики та типові помилки у розробці та реалізації системи обміну короткими повідомленнями;

4) сформулювати вичерпний перелік вимог до нової системи обміну короткими повідомленнями, враховуючи результати аналізу та потреби цільової аудиторії;

Цей розділ закладе міцний фундамент для подальшої розробки системи обміну короткими повідомленнями, забезпечивши її ефективність, надійність та відповідність сучасним вимогам.

1.2 Огляд протоколу короткими повідомленнями

MQTT – легковаговий протокол обміну повідомленнями, оптимізований для роботи з обмеженими ресурсами і ненадійними мережами. Він використовує архітектуру «видавець-підписник», де клієнти можуть видавати повідомлення на певні теми і підписуватись на отримання повідомлень з цих тем.

Основні функції та можливості:

1) публікація/підписка: Клієнти можуть публікувати повідомлення та підписуватися на теми;

2) друге,(Quality of Service): Три рівні якості обслуговування для забезпечення надійної доставки повідомлень;

3) збереження стану: Збереження останнього повідомлення для нових підписників (Last Will and Testament);

4) збереження сесій: Підтримка збереження інформації про підписки між відновленням з'єднань.

MQTT має ряд переваг і недоліків. Переваги включають легковаговість, що забезпечує низькі вимоги до пропускної здатності та ресурсів, простоту

реалізації та інтеграції в IoT проєкти, а також надійність завдяки підтримці QoS для забезпечення надійної доставки повідомлень.

Серед недоліків – відсутність вбудованих засобів безпеки, що вимагає додаткових заходів, таких як SSL/TLS, та обмежена масштабованість, яка підходить для малих та середніх систем, але потребує оптимізації для дуже великих мереж.

1.3 Огляд існуючих рішень для систем обміну короткими повідомленнями

Сьогодні ми розглянемо різні застосування MQTT одне з них буде в контексті підключення автомобіля до Інтернету за допомогою мікроконтролера Arduino MKR1000 і діагностичного адаптера ELM327. Це інноваційний підхід, який дозволяє збирати і передавати дані з блоку керування двигуном автомобіля (такі як оберти, температура двигуна, швидкість) до хмарного сервісу IBM Watson Cloud через протокол MQTT.

Рисунок 1.1 – MQTT OBD телеметрія автомобіля

В цю систему входить:

– плата Arduino MKR1000: Основний компонент проєкту, що має вбудований Wi-Fi shield для забезпечення з'єднання з IBM Watson Cloud через мережу Wi-Fi;

– мікроконтролер ELM327: який підключається до Arduino через UART і дозволяє читати дані з блоку управління двигуном автомобіля, використовуючи коди PID;

– модуль NEO-6M: Використовується для визначення місцезнаходження автомобіля, що додає контексту до зібраних даних;

– платформа для обробки та візуалізації даних. Вона виступає як MQTT брокер і забезпечує інтеграцію з Node-RED для обробки даних та створення веб-інтерфейсу.

Як це працює:

– збір даних: Arduino MKR1000 зчитує дані з ECU автомобіля за допомогою ELM327 через коди PID. Ці дані включають обороти, температуру двигуна, швидкість та інші параметри;

– передача до IBM Watson Cloud: Arduino використовує протокол MQTT для відправки зібраних даних на брокер в IBM Watson Cloud. MQTT забезпечує ефективну передачу даних з низькою латентністю та підтримкою QoS для надійної доставки;

– обробка та візуалізація: На платформі IBM Watson Cloud створюється ресурс Node-RED для обробки даних, що надходять від MQTT клієнтів. Node-RED використовується для створення веб-інтерфейсу, який може відображати та аналізувати дані з автомобіля;

– безпека та масштабованість: Платформа підтримує можливість налаштування TLS для безпечної передачі даних, а обмеження щодо використання ресурсів забезпечують ефективне управління обсягами даних.

Рисунок 1.2 – Додаток для iOS

Рисунок 1.3 – Інтерфейс вебпрограми

Проєкт із підключення автомобіля до Інтернету через Arduino MKR1000 та ELM327, з використанням MQTT для збору та передачі даних до IBM Watson Cloud, демонструє передові можливості IoT технологій у сфері автомобільного транспорту. Цей підхід дозволяє не тільки моніторити стан автомобіля в реальному часі, а й ефективно управляти та аналізувати зібрані дані для поліпшення експлуатаційних процесів і підтримки прийняття рішень.

Розглянемо наступну систему «Memo Minder» на базі ESP8266 та Arduino використовує MQTT протокол для ефективного надсилання текстових

повідомлень до пристрою, який привертає увагу шляхом звукових та світлових сигналів.

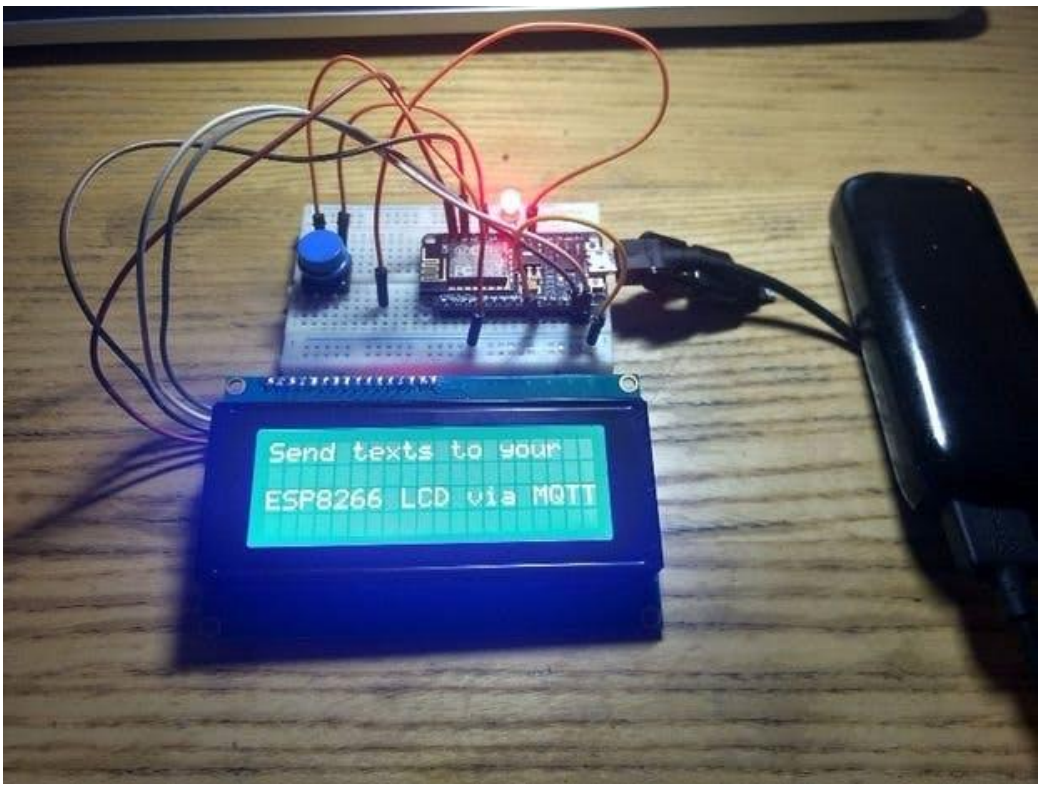


Рисунок 1.4 – ESP8266/Arduino MQTT Memo Minder W/LCD

Складається система з:

– мікроконтролер ESP8266-ESP12E (NodeMCU 1.0): який працює як приймач MQTT повідомлень і управляє всіма іншими компонентами проєкту;

– рк-дисплей: Використовується для відображення отриманих повідомлень;

– світлодіоди та зумер: Додаються для візуального та звукового сигналізування про отримання нових повідомлень;

– клієнт Android/iOS MQTT: Дозволяє надсилати текстові повідомлення на загальнодоступний MQTT брокер.

Як це працює:

– надсилання повідомлень: Користувач відправляє текстові повідомлення зі свого смартфону через безкоштовний MQTT клієнт на загальнодоступний MQTT брокер;

– прийом повідомлень на ESP8266: ESP8266 налаштований як MQTT клієнт, який підписується на тему брокера. Коли нове повідомлення надходить, ESP8266 отримує його і виконує дії згідно з програмним кодом;

– відображення на РК-дисплеї: Отримане повідомлення відображається на підключеному РК-дисплеї, щоб користувач міг його побачити;

– візуальне і звукове сповіщення: Після відображення повідомлення на дисплеї, виконуються додаткові дії для привертання уваги, такі як блимання світлодіодів і відтворення звукових сигналів з зумера.

Цей проєкт не лише дозволяє ефективно спілкуватися з геймерами, які занурені у свій світ онлайн-ігор, але й надає засіб привертати їхню увагу у відповідний момент. Він використовує доступні технології для створення корисного та захоплюючого пристрою, який може бути впроваджений в будь-якому домашньому середовищі з мінімальними витратами.

Цей проєкт є панеллю керування, побудованою на базі Raspberry Pi 3 з використанням сенсорного екрану і Android Things для зв’язку з платформою домашньої автоматизації через протокол MQTT.

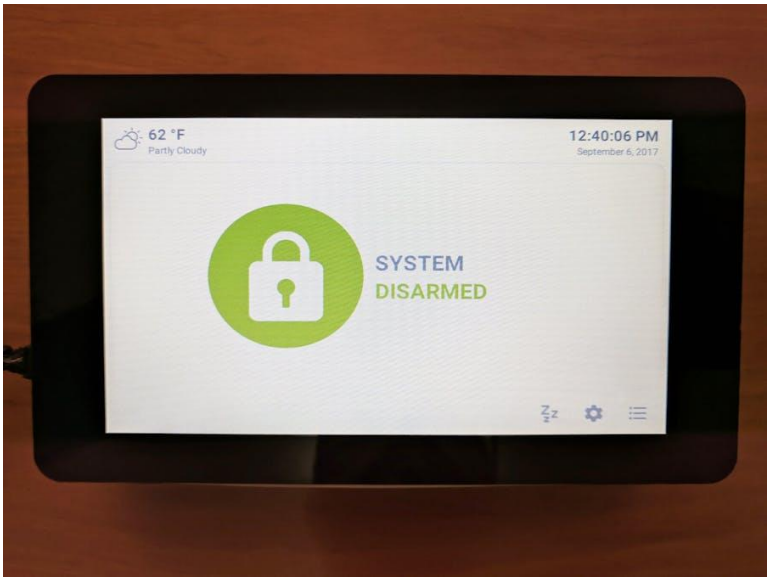


Рисунок 1.5 – Панель керування сигналізацією MQTT для домашнього помічника

Home Assistant, як платформа домашньої автоматизації з відкритим вихідним кодом, підтримує інтеграцію з MQTT для обміну даними з різними пристроями та системами. В даному проєкті Raspberry Pi використовується як клієнт MQTT, який взаємодіє з ручною панеллю керування сигналізацією Home Assistant. Це дозволяє контролювати статус системи сигналізації, отримувати сповіщення про події та змінювати налаштування зручним інтерфейсом.

Складається з:

– Raspberry Pi 3 Model B: Використовується як основний комп'ютер для обробки та відображення інформації на сенсорному дисплеї.

– 7» сенсорний дисплей для Raspberry Pi: Забезпечує інтерактивне керування та візуалізацію даних на панелі керування.

– Динамік з зовнішнім живленням: Використовується для аудіального сповіщення, замість п’єзо зумера, інтегрується зі смарт-дом системою через MQTT.

– Модуль камери Raspberry Pi: Додає можливість зйомки зображень для використання, коли система сигналізації вимкнена.

– PIR датчик руху: Забезпечує можливість виявлення руху для автоматизації дій на основі даних від нього.

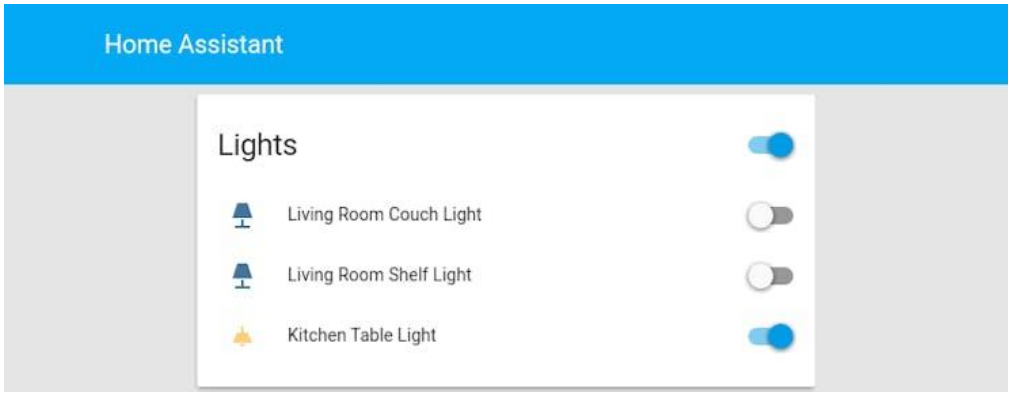


Рисунок 1.6 – Керування світлом в кімнатах за допомогою Home Assistant

Цей проєкт з використанням інтерфейсу панелі сигналізації з домашнім асистентом Home Assistant через MQTT працює так:

– інтеграція з Home Assistant: Установлено і налаштовано Home Assistant, який забезпечує керування домашніми пристроями і безпековими системами через різні датчики та сенсори;

– панель керування сигналізацією: Використовується Raspberry Pi з сенсорним екраном і Android Things. Ця панель отримує інформацію через протокол MQTT від Home Assistant про стан сигналізації і події безпеки;

– взаємодія через MQTT: Home Assistant надсилає повідомлення MQTT при спрацьовуванні сигналізації (наприклад, через відкриття дверей). Панель сигналізації отримує ці повідомлення і відображає відповідне сповіщення на сенсорному екрані;

– взаємодія з користувачем: Користувач може взаємодіяти з панеллю, вводячи код для вимкнення сигналізації. Після введення коду генерується повідомлення MQTT, яке надсилається до Home Assistant для вимкнення сигналізації;

– безпека і автоматизація: Якщо час для введення коду минув без відповіді, сигналізація активується, спрацьовуючи сирену або інші заходи безпеки. Home Assistant також може автоматично реагувати на стан датчиків і здійснювати відповідні дії, наприклад, надсилання повідомлень або активація інших пристроїв.

Цей проєкт забезпечує надійний механізм керування та моніторингу безпеки вдома за допомогою інтеграції різних датчиків, систем безпеки та цифрових інтерфейсів через MQTT та панель керування на базі Raspberry Pi з Android Things.

1.4 Інтеграція месенджерів через прошивку в Arduino

Інтеграція популярних месенджерів, таких як Telegram, WhatsApp чи Signal, з Arduino через MQTT відкриває широкі можливості для створення

систем сповіщень та керування розумним будинком. Хоча Arduino не має вбудованої підтримки цих месенджерів, інтеграція може бути реалізована за допомогою створення ботів та використання хмарних сервісів або локальних MQTT брокерів.

Для кожного месенджера необхідно створити бота, який буде отримувати та відправляти повідомлення. Telegram, WhatsApp та Signal надають API для створення ботів, які можна використовувати для взаємодії з Arduino. Бот повинен підключитися до MQTT брокера, який буде використовуватися для обміну повідомленнями між Arduino та месенджером. Бот може виступати як видавець (publisher) та/або підписник (subscriber) на певні топіки MQTT.

Arduino повинен бути запрограмований для підключення до того ж MQTT брокера та підписки на відповідні топіки. При отриманні повідомлення від бота Arduino може виконати певну дію (наприклад, увімкнути світло) та відправити відповідь на інший топік, який буде прослуховуватися ботом.

Бот повинен обробляти вхідні повідомлення від користувачів, визначати відповідні команди та відправляти їх на MQTT брокер. При отриманні відповідей від Arduino, бот повинен пересилати їх користувачам.

1.4.1 Приклади інтеграції

Telegram є одним із найпопулярніших месенджерів, що пропонує простий та зручний інтерфейс для створення ботів. Інтеграція з Arduino за допомогою MQTT може бути реалізована за допомогою бібліотеки UniversalTelegramBot. Ця бібліотека надає функції для обробки команд від користувачів, відправки повідомлень та файлів, а також для створення інтерактивних меню.

Наприклад, можна створити бота, який буде отримувати команди від користувачів (наприклад, «увімкнути світло») та публікувати їх на MQTT брокер. Arduino, підписаний на відповідний топік, отримає команду та виконає відповідну дію. Результат виконання команди (наприклад, «світло

увімкнено») може бути відправлений назад боту через MQTT та відображений користувачу в Telegram.

Переваги використання Telegram для інтеграції з IoT пристроями очевидні: простота у використанні, велика кількість користувачів, можливість створення багатофункціональних ботів, а також зручний інтерфейс для управління пристроями. Крім того, Telegram надає високу безпеку та швидкість передачі даних, що робить його ідеальним вибором для реалізації різноманітних IoT проєктів.

Інтеграція з WhatsApp може бути складнішою, оскільки WhatsApp не надає офіційного API для створення ботів. Проте, існують сторонні бібліотеки та сервіси, які дозволяють реалізувати таку інтеграцію. Наприклад, можна використовувати сервіс Twilio, який надає API для відправки та отримання повідомлень WhatsApp. За допомогою Twilio можна налаштувати бота, який буде отримувати команди від користувачів та взаємодіяти з MQTT брокером для управління IoT пристроями.

Signal також не має офіційного API для ботів, але існують проєкти з відкритим вихідним кодом, такі як Signal-cli, які можна використовувати для взаємодії з Signal через командний рядок. Для інтеграції з Arduino можна написати скрипт на Python або іншій мові програмування, який буде використовувати Signal-cli для відправки та отримання повідомлень, а також для взаємодії з MQTT брокером.

Таким чином, використання Telegram, WhatsApp та Signal для інтеграції з IoT пристроями через MQTT відкриває широкі можливості для автоматизації та управління різноманітними процесами. Кожен з цих месенджерів має свої переваги та обмеження, але завдяки наявним бібліотекам та сервісам можна знайти оптимальне рішення для конкретних завдань.

1.4.2 Переваги інтеграції месенджерів

Користувачі можуть взаємодіяти з системою розумного будинку або IoT-пристроями за допомогою звичного інтерфейсу месенджера, не

потребуючи встановлення додаткових програм. Мобільні додатки месенджерів, такі як Telegram, WhatsApp або Signal, відкривають можливості дистанційного управління пристроями з будь-якого місця, де є доступ до Інтернету.

За допомогою цих месенджерів користувачі можуть не лише відправляти текстові повідомлення, але й виконувати різноманітні додаткові функції. Наприклад, вони можуть отримувати сповіщення про стан пристроїв, спілкуватися в групових чатах для колективного керування, обмінюватися фотографіями та відео для візуалізації інформації, а також використовувати голосові повідомлення для зручного управління.

Ці месенджери стають центром взаємодії з IoT-пристроями, де кожен користувач може налаштовувати і персоналізувати свої уподобання щодо способу контролю пристроїв. Наприклад, вони можуть створювати спеціалізовані боти для автоматизації рутинних завдань або налаштовувати розклади включення та вимикання пристроїв за допомогою інтерактивних меню.

Такий підхід не тільки забезпечує зручність в управлінні, але й підвищує рівень інтерактивності і контролю за системою, що є важливим аспектом для сучасних користувачів, що цінують ефективність і зручність у використанні технологій.

1.4.3 Недоліки інтеграції месенджерів

Інтеграція месенджерів для управління IoT-пристроями через MQTT може виявитися складнішою порівняно з використанням простих MQTT клієнтів, особливо в разі WhatsApp та Signal, де відсутнє офіційне API для ботів. Для досягнення такої інтеграції може знадобитися використання сторонніх бібліотек або сервісів, що може вплинути на надійність і безпеку системи. Крім того, функціональність ботів у таких месенджерах часто обмежена можливостями їхніх API, що може потребувати додаткових зусиль для реалізації бажаних функцій.

З іншого боку, інтеграція популярних месенджерів з Arduino через MQTT відкриває нові можливості для створення зручних та функціональних систем управління розумним будинком та IoT-пристроями. Цей підхід дозволяє реалізувати індивідуальні та унікальні рішення, які відповідають конкретним потребам користувачів. Наприклад, користувачі можуть створювати ботів для виконання різних команд, від включення світла до налаштування температурного режиму, просто використовуючи звичайні текстові команди в месенджері.

При виборі месенджера для інтеграції важливо враховувати його популярність, наявність API для ботів, функціональні можливості та вимоги до безпеки. Наприклад, Telegram забезпечує широкі можливості для реалізації ботів через UniversalTelegramBot, що дозволяє взаємодіяти з Arduino і MQTT брокером з великою надійністю та зручністю. У той же час, для інтеграції з WhatsApp чи Signal можуть знадобитися альтернативні підходи, такі як використання сервісів Twilio чи Signal-cli, що можуть бути менш прямими, але все ще ефективними з точки зору функціональності.

1.5 Аналіз вимог до системи

Система обміну короткими повідомленнями призначена для забезпечення простого та зручного способу обміну текстовими повідомленнями між користувачами в локальній мережі.

Основні функції системи включають обробку та передачу повідомлень через MQTT протокол. Це означає, що система повинна бути здатна приймати, обробляти та відправляти короткі повідомлення за допомогою ESP8266. Система повинна працювати ефективно, забезпечуючи швидку передачу повідомлень. Має забезпечувати безпеку передачі даних через MQTT, включаючи застосування SSL/TLS. Система повинна забезпечувати стабільну та надійну передачу повідомлень навіть у випадку втрати зв'язку або виникнення помилок.

Функціональні вимоги:

– система повинна дозволяти користувачам відправляти короткі текстові повідомлення іншим користувачам в мережі. Користувачі повинні мати можливість переглядати отримані повідомлення. Повідомлення повинні доставлятися одержувачу з мінімальною затримкою;

– кожен користувач повинен мати унікальний ідентифікатор (наприклад, ім'я користувача або псевдонім). Система повинна забезпечувати можливість вибору одержувача повідомлення за його ідентифікатором;

– система повинна зберігати історію відправлених та отриманих повідомлень для кожного користувача. Користувачі повинні мати можливість переглядати історію своїх повідомлень;

– система повинна мати простий та інтуїтивно зрозумілий інтерфейс користувача. Інтерфейс повинен дозволяти користувачам легко вводити текст повідомлення, вибирати одержувача та переглядати історію повідомлень.

Нефункціональні вимоги:

– система повинна забезпечувати швидку відправку та отримання повідомлень. Затримка доставки повідомлень повинна бути мінімальною;

– система повинна бути надійною та стабільно працювати в умовах локальної мережі. Повідомлення не повинні губитися при передачі;

– система повинна забезпечувати конфіденційність повідомлень. Необхідно запобігти несанкціонованому доступу до повідомлень;

– система повинна мати можливість масштабування для підтримки великої кількості користувачів;

– система повинна бути сумісною з різними пристроями на базі ESP8266.

Обмеження:

– система обмежена обчислювальними ресурсами та пам'яттю мікроконтролера ESP8266;

– вибір прошивки (Tasmota, ESPEasy, NodeMCU) може вплинути на функціональність та складність реалізації системи.

Отже, створення системи обміну короткими повідомленнями на базі ESP8266 та MQTT є перспективним проєктом, який вимагає уважного підходу до вибору технологій та реалізації, з метою забезпечення надійності, безпеки та зручності використання для користувачів.

1.6 Огляд методів захисту таких систем

Безпека є критично важливим аспектом будь-якої системи обміну повідомленнями, особливо в контексті Інтернету речей (IoT), де пристрої часто підключаються до відкритих або громадських мереж і можуть стати об'єктом кібератак. Розглянемо основні методи захисту, які необхідно врахувати при розробці системи обміну короткими повідомленнями на базі ESP8266 та MQTT. Основні методи безпеки:

– шифрування зв'язку (SSL/TLS): Використання SSL/TLS для шифрування з'єднання між ESP8266 і MQTT брокером є критично важливим для забезпечення конфіденційності та цілісності передаваних даних. Це запобігає перехопленню та підслуховуванню інформації з боку зловмисників;

– аутентифікація і авторизація: Встановлення механізмів аутентифікації (перевірка ідентифікації користувача) і авторизації (контроль прав доступу) є необхідними для управління доступом до системи. Використання сильних паролів, токенів або сертифікатів для ідентифікації користувачів і забезпечення відповідних дозволів є ключовим аспектом забезпечення безпеки;

– захист Wi-Fi мережі: Оскільки ESP8266 підключається до Wi-Fi для зв'язку з MQTT брокером, важливо застосовувати належні заходи захисту для самої Wi-Fi мережі. Це включає використання сильних паролів, зміну стандартного SSID, використання WPA2/WPA3 шифрування, фільтрацію MAC-адрес та ізоляцію клієнтів для уникнення несанкціонованого доступу до мережі;

– оновлення прошивки та захист від вразливостей: Регулярні оновлення прошивки ESP8266 та MQTT брокера важливі для виправлення виявлених вразливостей і підвищення загального рівня безпеки системи. Використання оновленої версії програмного забезпечення допомагає запобігти експлуатації вразливостей, виявлених у попередніх версіях;

– моніторинг та журналювання подій: Важливим аспектом є належний моніторинг діяльності системи, виявлення незвичайних активностей і журналювання подій. Це дозволяє оперативно реагувати на потенційні загрози і вчасно вживати заходів для їх запобігання.

Застосування цих методів захисту дозволяє забезпечити високий рівень безпеки системи обміну короткими повідомленнями на базі ESP8266 та MQTT, зменшити ризики кібератак і зберігати конфіденційність даних користувачів.

1.6.1 Захист Wi-Fi мережі

Оскільки ESP8266 підключається до мережі Wi-Fi для взаємодії з MQTT брокером, важливо забезпечити безпеку цього з'єднання. Основні заходи захисту Wi-Fi мережі включають:

– використання сильних паролів: Пароль до Wi-Fi мережі повинен бути достатньо довгим та складним, щоб ускладнити його підбір зловмисниками. Рекомендується використовувати комбінацію літер (великих та маленьких), цифр та спеціальних символів;

– зміна стандартного SSID: Зміна стандартного імені мережі (SSID) ускладнює ідентифікацію мережі зловмисниками;

– використання WPA2/WPA3 шифрування: WPA2/WPA3 - це сучасні протоколи шифрування Wi-Fi, які забезпечують високий рівень захисту даних;

– фільтрація MAC-адрес: Дозволяє обмежити доступ до мережі лише для певних пристроїв, MAC-адреси яких внесені до білого списку;

– ізоляція клієнтів: Запобігає прямому зв'язку між клієнтами в мережі, що знижує ризик поширення шкідливого програмного забезпечення.

Застосування цих заходів є критичним для забезпечення безпеки зв'язку між ESP8266 та MQTT брокером через Wi-Fi мережу. Використання сильного пароля і шифрування даних дозволяє мінімізувати ризики перехоплення чутливої інформації, тоді як фільтрація MAC-адрес та ізоляція клієнтів допомагають контролювати доступ до мережі та зменшують загрози з боку зловмисників.

Загальний підхід до захисту Wi-Fi мережі сприяє підвищенню загальної безпеки систем IoT, зокрема забезпечує надійний захист ESP8266, який використовується для зв'язку з MQTT брокером.

1.6.2 Аутентифікація та авторизація користувачів

Для запобігання несанкціонованому доступу до системи необхідно реалізувати механізми аутентифікації та авторизації користувачів. Аутентифікація підтверджує особу користувача, а авторизація визначає, які дії він може виконувати в системі. Можливі методи аутентифікації:

– ім'я користувача та пароль: Найпоширеніший метод, але вимагає безпечного зберігання паролів;

– токени: Унікальні рядки символів, які видаються користувачам після успішної аутентифікації;

– сертифікати: Цифрові документи, які підтверджують особу користувача та можуть використовуватися для шифрування з'єднання;

Можливі методи авторизації:

– списки керування доступом (ACL): Визначають, які топіки доступні для читання та запису кожному користувачу;

– ролі: Групують користувачів за рівнями доступу та дозволяють призначати дозволи на основі ролей.

Застосування цих методів дозволяє забезпечити високий рівень безпеки та контролю над доступом до системи IoT через месенджери, зменшуючи ризик несанкціонованого доступу та зберігаючи конфіденційність даних користувачів.

Вибір оптимальних методів аутентифікації та авторизації є критичним для забезпечення безпеки системи IoT. Імплементація цих механізмів дозволяє ефективно захищати від несанкціонованого доступу та забезпечувати відповідність з вимогами до безпеки даних інтернет-речей.

1.6.3 Шифрування повідомлень

Для забезпечення конфіденційності повідомлень в системі обміну короткими повідомленнями через MQTT рекомендується активно використовувати шифрування, наприклад, за допомогою SSL/TLS. Це необхідно для захисту від можливих атак з перехопленням даних, які можуть відбуватися під час передачі інформації між пристроями та серверами.

Одним з важливих аспектів застосування шифрування є його вплив на продуктивність системи. Шифрування і розшифрування повідомлень вимагають обчислювальних ресурсів, особливо на пристроях з обмеженими характеристиками, таких як мікроконтролери типу ESP8266. Це може призвести до затримок у обробці повідомлень та збільшення споживання енергії, що потрібно враховувати при проєктуванні та налаштуванні системи IoT.

Крім того, для безпечного зберігання і розповсюдження сертифікатів SSL/TLS використовуються спеціальні практики із застосуванням безпечних засобів управління ключами. Це важливо для забезпечення цілісності і конфіденційності даних під час їх обміну.

Таким чином, вибір використання шифрування повідомлень у системі IoT через месенджери MQTT є ключовим аспектом забезпечення безпеки і може залежати від конкретних потреб проєкту, рівня конфіденційності і доступності ресурсів пристроїв.

Висновки до розділу 1

У результаті аналізу прошивок для ESP8266 та вимог до системи обміну повідомленнями можна зробити наступні висновки. Для реалізації системи обміну короткими повідомленнями на базі ESP8266 та MQTT оптимальним вибором буде Tasmota або NodeMCU. Tasmota пропонує готові рішення для роботи з MQTT та має зручний веб-інтерфейс, що спрощує налаштування та використання системи. З іншого боку, NodeMCU надає більшу гнучкість завдяки можливості програмування на Lua, що дозволяє реалізувати додатковий функціонал та індивідуальні рішення.

Система повинна забезпечувати основні функції обміну повідомленнями, включаючи відправку, отримання, перегляд історії та ідентифікацію користувачів. Крім того, важливо передбачити можливість розширення функціоналу в майбутньому, наприклад, додавання групових чатів і сповіщень. Надійність, продуктивність, безпека та масштабованість є ключовими аспектами проєкту.

Особливу увагу слід приділити забезпеченню конфіденційності повідомлень та захисту від несанкціонованого доступу. Враховуючи обмежені ресурси мікроконтролера ESP8266 та вибраної прошивки, важливо раціонально використовувати ресурси для оптимальної роботи системи.

На основі проведеного аналізу можна приступати до проєктування та реалізації системи обміну повідомленнями. Використання найкращих практик розробки вбудованих систем та уважне врахування всіх вимог і обмежень є запорукою успішної реалізації проєкту.

Таким чином, створення системи обміну короткими повідомленнями на базі ESP8266 та MQTT є перспективним проєктом, який відкриває широкі можливості для застосування в інтернеті речей. успішне завершення проєкту залежить від професіоналізму, глибини аналізу та вибору оптимальних рішень.

2 ПРОЄКТУВАННЯ СИСТЕМИ ОБМІНУ КОРОТКИМИ ПОВІДОМЛЕННЯМИ НА БАЗІ ESP8266 ТА MQTT

2.1 Архітектура системи

2.1.1 Загальний опис архітектури

Система обміну короткими повідомленнями на базі ESP8266 та MQTT буде побудована за клієнт-серверною архітектурою, де ESP8266 виступатиме в ролі клієнта, а MQTT брокер – в ролі сервера. Клієнтські пристрої (наприклад, смартфони, планшети або комп'ютери) будуть підключатися до MQTT брокера та обмінюватися повідомленнями через нього.

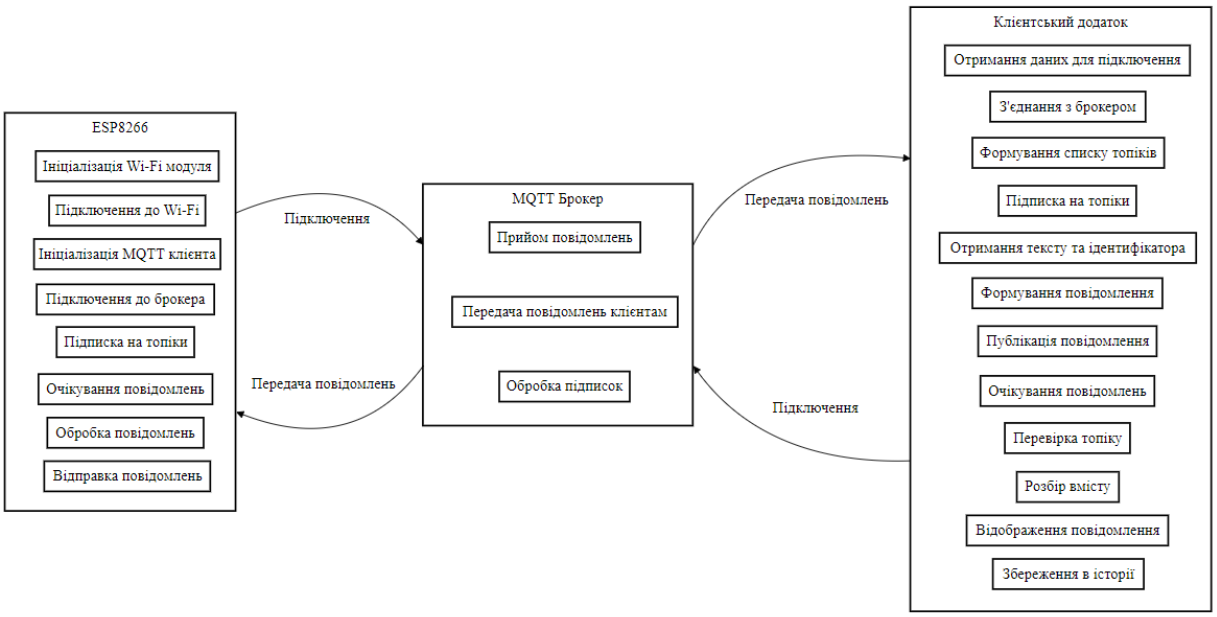


Рисунок 2.1 – Діаграма структури системи в цілому

Система складається з трьох основних компонентів:

1) перше, ESP8266: мікроконтролер з вбудованим Wi-Fi модулем, який виконує функції клієнта MQTT. Він відповідає за підключення до мережі Wi-Fi, встановлення зв'язку з MQTT брокером, публікацію та отримання повідомлень, а також взаємодію з користувачем через інтерфейс;

2) друге, MQTT брокер: центральний сервер, який відповідає за зберігання, маршрутизацію та доставку повідомлень між клієнтами MQTT. Він приймає повідомлення від клієнтів, що публікують, фільтрує їх за темами та доставляє їх клієнтам, що підписалися на відповідні теми;

3) клієнтські пристрої: пристрої, які використовуються користувачами для взаємодії з системою. Це можуть бути смартфони, планшети, комп'ютери або будь-які інші пристрої, здатні підключитися до мережі та використовувати MQTT клієнт.

Отже, ESP8266 відповідає за підключення до Wi-Fi мережі, зв'язок з MQTT брокером, публікацію та отримання повідомлень, а також взаємодію з користувачем через інтерфейс. MQTT брокер – це центральний сервер, який зберігає, маршрутизує та доставляє повідомлення між клієнтами MQTT, приймаючи повідомлення від публікуючих клієнтів і доставляючи їх підписникам. Клієнтські пристрої, такі як смартфони, планшети і комп'ютери, дозволяють користувачам взаємодіяти з системою через MQTT клієнт.

2.1.2 Взаємодія між компонентами

Взаємодія між компонентами системи обміну короткими повідомленнями відбувається наступним чином:

1) починаемо з, ESP8266 ініціалізується та підключається до локальної мережі Wi-Fi. Це дозволяє йому отримати доступ до MQTT брокера та інших пристроїв у мережі;

2) після успішного підключення до Wi-Fi, ESP8266 встановлює з'єднання з MQTT брокером. Для цього він використовує адресу брокера (URL або IP-адресу) та порт, на якому брокер очікує підключення. Під час встановлення з'єднання можуть використовуватися механізми аутентифікації та шифрування (SSL/TLS) для забезпечення безпеки;

3) потім, ESP8266 підписується на певні MQTT топіки. Топіки — це канали, за якими передаються повідомлення. Кожен користувач має свій унікальний топік, який використовується для адресації повідомлень.

Підписавшись на топік, ESP8266 отримуватиме всі повідомлення, опубліковані на цьому топіку;

4) користувач, використовуючи клієнтський пристрій (наприклад, смартфон), вводить текст повідомлення та ідентифікатор одержувача. Цей ідентифікатор використовується для визначення топіка, на який потрібно відправити повідомлення;

5) клієнтський пристрій відправляє повідомлення на MQTT брокер, вказуючи топік одержувача. Брокер приймає повідомлення та зберігає його. При необхідності, повідомлення може бути доставлено з певним рівнем якості обслуговування (QoS), що гарантує його доставку навіть при нестабільному з'єднанні;

6) також, MQTT брокер розповсюджує повідомлення всім клієнтам, які підписані на топік одержувача. Це означає, що ESP8266 одержувача отримає повідомлення;

7) у кінці, ESP8266 одержувача обробляє отримане повідомлення та відображає його користувачеві. Це може бути відображення на екрані пристрою, відтворення звукового сигналу або інший спосіб оповіщення, залежно від типу клієнтського пристрою та його можливостей.

Отже, взаємодія між компонентами системи обміну короткими повідомленнями відбувається наступним чином: спочатку ESP8266 ініціалізується та підключається до локальної мережі Wi-Fi, що дозволяє йому отримати доступ до MQTT брокера та інших пристроїв у мережі. Після успішного підключення до Wi-Fi, ESP8266 встановлює з'єднання з MQTT брокером, використовуючи адресу брокера та порт, з можливим застосуванням аутентифікації та шифрування для забезпечення безпеки. Далі ESP8266 підписується на певні MQTT топіки, отримуючи всі повідомлення, опубліковані на цих топіках. Користувач, використовуючи клієнтський пристрій, вводить текст повідомлення та ідентифікатор одержувача, визначаючи топік для відправки повідомлення. Клієнтський пристрій відправляє повідомлення на MQTT брокер, який приймає та зберігає його,

розповсюджуючи всім клієнтам, підписаним на топік одержувача, включаючи ESP8266 одержувача, що обробляє та відображає повідомлення користувачеві.

2.1.3 Обґрунтування вибору архітектури

Клієнт-серверна архітектура з використанням MQTT обрана з наступних причин:

1) протокол спеціально розроблений для IoT пристроїв та забезпечує легкий та ефективний обмін повідомленнями між великою кількістю клієнтів. Він використовує публікацію/підписку, що дозволяє ефективно масштабувати систему при збільшенні кількості користувачів;

2) брокер забезпечує механізми гарантованої доставки повідомлень (QoS), що дозволяє забезпечити надійність доставки навіть при нестабільному з'єднанні або тимчасовій недоступності клієнта;

3) підтримує асинхронний обмін повідомленнями, що дозволяє ESP8266 та клієнтським пристроям працювати незалежно один від одного та не блокувати роботу системи при очікуванні відповіді;

4) є легким протоколом, який споживає мінімум ресурсів та пропускної здатності мережі, що особливо важливо для пристроїв з обмеженими можливостями, таких як ESP8266;

5) дозволяє легко додавати нові клієнтські пристрої та розширювати функціональність системи без внесення суттєвих змін до архітектури.

Використання ESP8266 в якості клієнта MQTT обумовлено його низькою вартістю, доступністю, широкими можливостями (вбудований Wi-Fi модуль, підтримка різних прошивок) та великою спільнотою розробників.

Запропонована клієнт-серверна архітектура з використанням MQTT та ESP8266 забезпечує ефективне та масштабоване рішення для обміну короткими повідомленнями. Вибір MQTT гарантує надійну доставку повідомлень та легку інтеграцію з різними клієнтськими пристроями, тоді як ESP8266 надає необхідну гнучкість та економічність для реалізації системи.

2.2 Опис протоколів

Система обміну короткими повідомленнями використовує кілька протоколів для забезпечення взаємодії між компонентами та передачі даних:

1) основний протокол обміну повідомленнями в системі. Він забезпечує асинхронну комунікацію між ESP8266 (клієнтом) та MQTT брокером (сервером) за моделлю «видавець-підписник». ESP8266 підписується на певні топіки, які представляють собою канали для передачі повідомлень. Коли клієнтський пристрій відправляє повідомлення, воно публікується на відповідний топік на MQTT брокері, який потім доставляє його всім підписаним клієнтам. MQTT використовує TCP/IP як транспортний протокол, забезпечуючи надійну доставку повідомлень;

2) протокол бездротового зв'язку, який використовується ESP8266 для підключення до локальної мережі та MQTT брокера. ESP8266 підтримує стандарти Wi-Fi 802.11 b/g/n та різні режими безпеки, такі як WPA/WPA2;

3) взаємодії між веб-браузером користувача та ESP8266. ESP8266 в цьому випадку виступає в ролі веб-сервера, обробляючи HTTP запити та відправляючи відповіді у форматі HTML.

Отже, основним протоколом обміну повідомленнями в системі є MQTT, який забезпечує асинхронну комунікацію між ESP8266 (клієнтом) та MQTT брокером (сервером) за моделлю «видавець-підписник». ESP8266 підписується на певні топіки, що слугують каналами для передачі повідомлень. Коли клієнтський пристрій відправляє повідомлення, воно публікується на відповідний топік на MQTT брокері, який потім доставляє його всім підписаним клієнтам. MQTT використовує TCP/IP як транспортний протокол, забезпечуючи надійну доставку повідомлень. Протокол бездротового зв'язку, який використовується ESP8266 для підключення до локальної мережі та MQTT брокера, підтримує стандарти Wi-Fi 802.11 b/g/n та різні режими безпеки, такі як WPA/WPA2. Взаємодія між веб-браузером користувача та ESP8266 здійснюється через HTTP-запити, де ESP8266

виступає в ролі веб-сервера, обробляючи запити та відправляючи відповіді у форматі HTML.

2.3 Обробка помилок

Система обміну короткими повідомленнями повинна бути стійкою до різних помилок та збоїв, які можуть виникнути під час роботи.

Якщо ESP8266 втрачає зв'язок з Wi-Fi мережею, він повинен автоматично спробувати перепідключитися. Кількість спроб перепідключення та інтервал між ними можуть бути налаштовані. Якщо перепідключення не вдається протягом певного часу, система може перейти в режим очікування або вимкнутися.

Якщо ESP8266 втрачає зв'язок з MQTT брокером, він повинен спробувати відновити з'єднання. При цьому повідомлення, які не були доставлені через втрату зв'язку, можуть бути збережені в буфері та відправлені після відновлення з'єднання.

Якщо користувач вводить неправильні дані (наприклад, неіснуючий ідентифікатор одержувача), система повинна повідомити про це користувача та запропонувати виправити помилку.

Якщо буфер повідомлень на ESP8266 переповнюється, система повинна припинити прийом нових повідомлень та повідомити про це користувача. Можна реалізувати механізм скидання старих повідомлень при переповненні буфера.

У разі апаратних збоїв (наприклад, збій живлення), ESP8266 повинен автоматично перезавантажитися та відновити роботу після усунення збою.

Для забезпечення надійної обробки помилок важливо передбачити механізми логування та моніторингу стану системи. Це дозволить виявляти та аналізувати помилки, а також вживати заходів для їх усунення.

2.4 Програмне забезпечення

2.4.1 Вибір прошивки для ESP8266

Існує кілька відомих проєктів, які використовують ESP8266 та MQTT для управління IoT пристроями. До них належать Tasmota, ESPEasy та NodeMCU, кожен з яких має свої унікальні особливості та застосування.

Tasmota – це потужна та гнучка прошивка з відкритим вихідним кодом, призначена для мікроконтролерів ESP8266. Вона перетворює звичайні пристрої на «розумні», дозволяючи їм підключатися до мережі Wi-Fi, взаємодіяти з іншими пристроями та системами, а також управлятися віддалено.

Рисунок 2.2 – Tasmota для управління IoT пристроями

Серед її функціональних можливостей - підтримка широкого набору сенсорів та зручний веб-інтерфейс для налаштування. Однак Tasmota вимагає базових знань у налаштуванні мереж та роботи з MQTT.

Основні переваги Tasmota:

1) простота налаштування: Незважаючи на широкий функціонал, Tasmota має інтуїтивно зрозумілий веб-інтерфейс, що дозволяє легко налаштовувати пристрої навіть користувачам без глибоких технічних знань;

2) універсальність: Tasmota підтримує величезну кількість різноманітних пристроїв, включаючи розумні розетки, вимикачі, датчики, реле, світлодіодні стрічки та багато інших;

3) використання протоколу MQTT (Message Queuing Telemetry Transport) забезпечує надійну та ефективну комунікацію між пристроями, а також легку інтеграцію з популярними системами розумного дому, такими як Home Assistant;

4) спільнота та підтримка: Tasmota має велику та активну спільноту розробників та користувачів, які постійно вдосконалюють прошивку, створюють нові функції та допомагають один одному;

5) безпека: Tasmota надає можливість налаштування безпечного підключення до мережі Wi-Fi та захисту даних;

Функціональні можливості Tasmota:

1) підтримка різноманітних сенсорів: Tasmota дозволяє підключати та використовувати широкий спектр сенсорів, таких як датчики температури, вологості, руху, освітленості, рівня води та багато інших;

2) таймери та розклади: Можливість створення розкладів та таймерів для автоматичного ввімкнення/вимкнення пристроїв у певний час або за певних умов;

3) сценарії та правила: Створення складних сценаріїв автоматизації, які дозволяють пристроям взаємодіяти між собою та реагувати на зміни стану сенсорів;

4) веб-інтерфейс: Зручний веб-інтерфейс для налаштування пристроїв, перегляду їх стану та управління ними;

5) можливість управління пристроями через HTTP запити, що відкриває широкі можливості для інтеграції з іншими системами та сервісами;

6) оновлення прошивки «по повітрю» (OTA): Можливість оновлення прошивки пристроїв без необхідності підключати їх до комп'ютера.

Tasmota – це потужний інструмент для створення власного розумного дому та управління IoT пристроями. Завдяки своїй гнучкості, широкому

функціоналу та активній спільноті, Tasmota відкриває безмежні можливості для автоматизації та покращення комфорту вашого життя.

ESPEasy – це відкрита прошивка, спеціально розроблена для спрощення процесу створення власних розумних пристроїв на базі мікроконтролера ESP8266. Її головна мета – зробити налаштування та використання різноманітних сенсорів, актуаторів та інших компонентів максимально простим та доступним навіть для новачків.

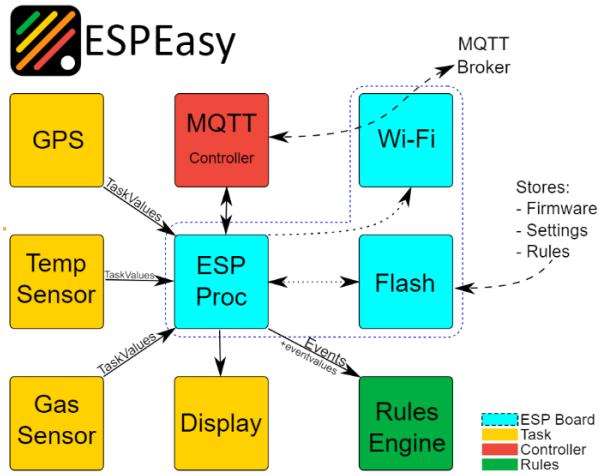


Рисунок 2.3 – Прошивка ESPEasy

Її основні переваги включають:

1) інтуїтивний інтерфейс: ESPEasy має зручний веб-інтерфейс, який дозволяє легко налаштовувати пристрої, додавати сенсори, створювати правила та керувати всіма функціями без необхідності писати код;

2) підтримка різноманітних компонентів: ESPEasy підтримує велику кількість сенсорів (температури, вологості, руху тощо), актуаторів (реле, сервоприводи, світлодіоди тощо) та інших компонентів, що дозволяє створювати різноманітні розумні пристрої;

3) підтримує протокол MQTT, що дозволяє легко інтегрувати створені пристрої з популярними системами розумного дому, такими як Home Assistant, та керувати ними віддалено;

4) правила: ESPEasy дозволяє створювати прості правила для автоматизації роботи пристроїв. Наприклад, можна налаштувати автоматичне

ввімкнення світла при спрацьовуванні датчика руху або відправлення повідомлення на смартфон при перевищенні певної температури;

5) спільнота: ESPEasy має активну спільноту користувачів та розробників, які допомагають один одному, діляться досвідом та створюють нові плагіни та функції.

Обмеження ESPEasy:

1) у порівнянні з Tasmota, ESPEasy має менше можливостей для налаштування та розширення функціоналу;

2) менша кількість підтримуваних компонентів;

3) правила в ESPEasy більш прості та менш гнучкі, ніж сценарії в Tasmota.

ESPEasy – це чудовий вибір для тих, хто хоче швидко та легко створити власні розумні пристрої на базі ESP8266. Завдяки простому інтерфейсу, підтримці MQTT та можливості створення правил, ESPEasy дозволяє реалізувати багато цікавих проєктів автоматизації. Однак, якщо вам потрібна більша гнучкість та ширший функціонал, Tasmota може бути кращим варіантом.

NodeMCU - це не лише прошивка, а ціла платформа розробки, яка поєднує апаратну частину (мікроконтролер ESP8266) та програмне забезпечення (інтерпретатор мови Lua). Це відкриває широкі можливості для створення різноманітних IoT пристроїв та проєктів, що вимагають гнучкості, високої продуктивності та індивідуального підходу.

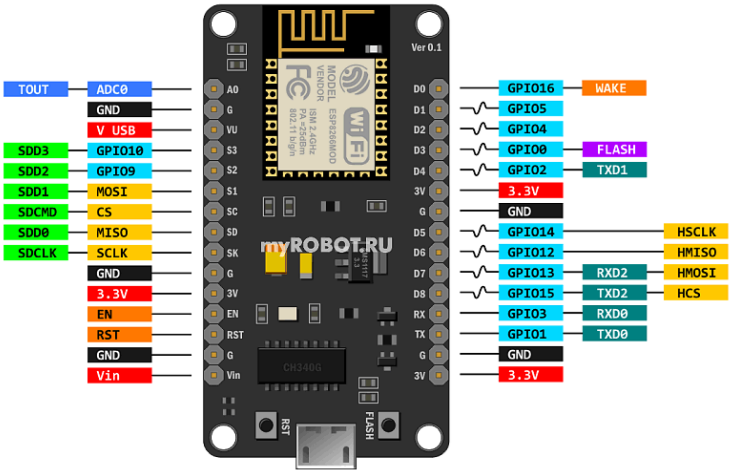


Рисунок 2.4 – Прошивка NodeMCU

Вона надає можливість писати власний код на Lua, що забезпечує високу гнучкість у розробці індивідуальних рішень. Проте, NodeMCU вимагає глибших знань у програмуванні та налаштуванні.

Основні переваги включають NodeMCU:

1) мова програмування Lua відрізняється простотою, елегантністю та високою швидкістю виконання. Вона ідеально підходить для вбудованих систем та IoT проєктів завдяки своїй легкості та мінімальному навантаженню на ресурси мікроконтролера;

2) дозволяє розробникам писати власний код на Lua, що надає повний контроль над функціональністю пристрою та можливість реалізації будь-яких ідей;

3) має вбудовану підтримку протоколу MQTT, що робить його ідеальним вибором для проєктів, пов'язаних з передачею даних між пристроями, хмарними сервісами та локальними серверами;

4) має велику кількість готових бібліотек та модулів, які спрощують роботу з різними сенсорами, актуаторами, мережевими протоколами та іншими компонентами;

5) має велику та активну спільноту розробників, які створюють нові бібліотеки, модулі та проєкти, діляться досвідом та допомагають один одному.

NodeMCU – це потужний інструмент для розробників, які хочуть створювати унікальні та індивідуальні IoT проєкти. Завдяки своїй гнучкості, підтримці MQTT та великій спільноті, NodeMCU відкриває безмежні можливості для реалізації найсміливіших ідей. Однак, якщо ви новачок у світі IoT та програмування, варто почати з більш простих прошивок, таких як ESPEasy, та поступово переходити до NodeMCU, коли ви набудете достатньо досвіду.

У світі Інтернету речей (IoT) та розумних пристроїв прошивки відіграють ключову роль, надаючи мікроконтролерам ESP8266 нові можливості та функціонал. Кожна з розглянутих прошивок – Tasmota, ESPEasy та NodeMCU – має свої унікальні переваги та особливості, що робить їх придатними для різних типів проєктів та рівня підготовки користувачів.

Tasmota вражає своєю універсальністю, широким набором функцій та підтримкою великої кількості компонентів. Вона ідеально підходить для створення складних систем розумного дому та управління різноманітними IoT пристроями.

ESPEasy приваблює своєю простотою та інтуїтивним інтерфейсом, що робить її чудовим вибором для новачків та тих, хто хоче швидко створити прості розумні пристрої без необхідності глибокого занурення у програмування.

NodeMCU надає максимальну гнучкість та контроль завдяки використанню мови Lua. Це ідеальний інструмент для досвідчених розробників, які хочуть створювати унікальні та індивідуальні проєкти, використовуючи весь потенціал мікроконтролера ESP8266.

Вибір прошивки залежить від ваших потреб, рівня знань та цілей проєкту. Якщо ви новачок, ESPEasy може бути найкращим варіантом для старту. Якщо вам потрібна універсальність та широкий функціонал, Tasmota стане чудовим вибором. А якщо ви досвідчений розробник, який прагне максимальної гнучкості та контролю, NodeMCU відкриє перед вами безмежні можливості.

Програмне забезпечення системи обміну короткими повідомленнями складається з двох основних частин: прошивки для ESP8266 та клієнтського додатку (якщо необхідно).

Для реалізації системи обміну повідомленнями на базі ESP8266 можна використовувати різні прошивки, кожна з яких має свої переваги та недоліки:

– Tasmota досить популярна прошивка з відкритим вихідним кодом, яка пропонує широкий набір функцій, включаючи підтримку MQTT, веб-інтерфейс для налаштування, сценарії автоматизації тощо. Вона добре підходить для створення складних систем, але може бути надмірною для простих проєктів.

– ESPEasy більш простіша прошивка, яка також підтримує MQTT та має зручний веб-інтерфейс. Вона легше в налаштуванні, але має менше можливостей, ніж Tasmota.

– NodeMCU прошивка на основі Lua, яка надає велику гнучкість та можливість програмування власної логіки роботи. Однак, вона вимагає знання мови Lua та може бути складнішою у використанні для початківців.

Для реалізації системи обміну повідомленнями на базі ESP8266 пропонується використовувати прошивку NodeMCU. Ця прошивка, заснована на мові програмування Lua, надає розробникам високу гнучкість та можливість створювати власні рішення, що ідеально підходить для реалізації індивідуальних вимог та функцій системи обміну повідомленнями.

2.4.2 Опис алгоритмів роботи

Алгоритми роботи системи обміну короткими повідомленнями на базі ESP8266 та MQTT можна розділити на дві основні частини: алгоритми роботи на стороні ESP8266 та алгоритми роботи клієнтського додатку.

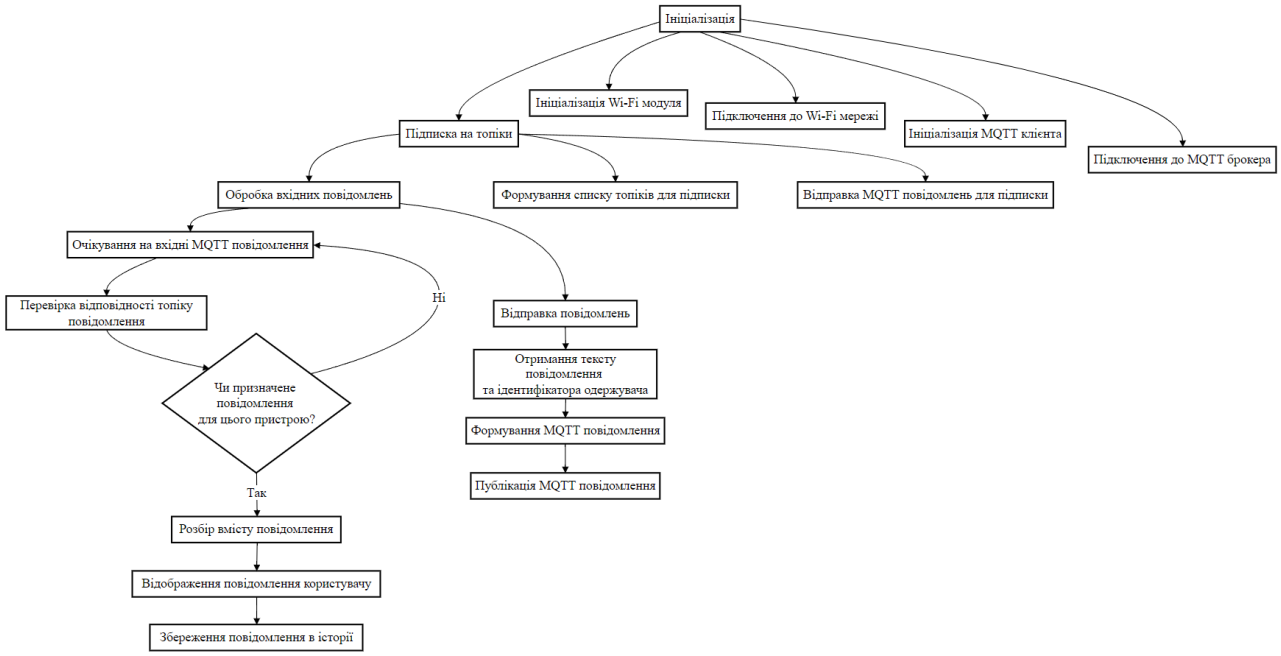


Рисунок 2.5 – Блок-схема роботи на стороні ESP8266

Нижче наданий детальний опис роботи алгоритму на стороні ESP8266:

1) ініціалізація: на цьому етапі відбувається ініціалізація Wi-Fi модуля ESP8266, встановлення підключення до локальної Wi-Fi мережі з використанням заданих параметрів (SSID та пароля). Також ініціалізується MQTT клієнт та встановлюється з'єднання з MQTT брокером, використовуючи його адресу, порт та ідентифікатор клієнта;

2) підписка на топіки: на основі ідентифікатора користувача та груп, до яких він належить, формується список MQTT топіків, на які ESP8266 повинен підписатися. Далі відправляються MQTT повідомлення на брокер, щоб підписатися на кожен з цих топіків;

3) обробка вхідних повідомлень: ESP8266 постійно очікує на вхідні MQTT повідомлення. При отриманні повідомлення відбувається перевірка топіка, на який воно було надіслане. Якщо топік відповідає ідентифікатору користувача або групі, до якої він належить, повідомлення вважається призначеним для цього пристрою. Далі відбувається розбір вмісту повідомлення (текст, відправник, час тощо) та його відображення користувачу

(наприклад, на дисплеї). Повідомлення також зберігається в історії, локально або віддалено, залежно від налаштувань системи;

4) відправка повідомлень: коли користувач вводить текст повідомлення та ідентифікатор одержувача, ESP8266 формує MQTT повідомлення, що містить текст повідомлення та топік одержувача. Потім це повідомлення публікується на MQTT брокері, який доставляє його одержувачу.

Далі, розглянемо роботу алгоритму на стороні клієнтського додатку:

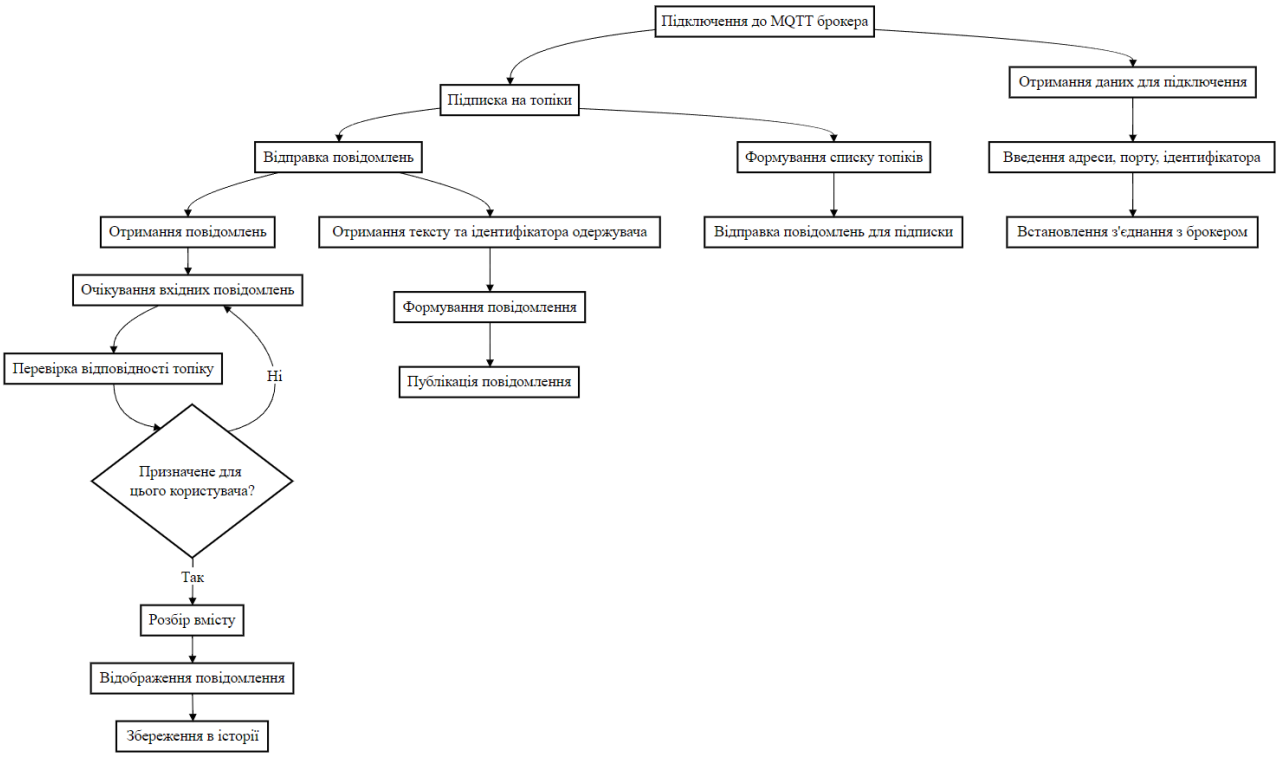


Рисунок 2.6 – Блок-схема роботи на стороні клієнтського додатку

1) підключення до MQTT брокера: клієнтський додаток отримує від користувача дані для підключення до MQTT брокера (адреса, порт, ідентифікатор клієнта) та встановлює з'єднання;

2) підписка на топіки: аналогічно до ESP8266, клієнтський додаток формує список топіків для підписки на основі ідентифікатора користувача та груп, до яких він належить, та відправляє MQTT повідомлення для підписки на ці топіки;

3) відправка повідомлень: при введенні користувачем тексту повідомлення та ідентифікатора одержувача, додаток формує MQTT повідомлення та публікує його на відповідний топік на MQTT брокері;

4) отримання повідомлень: клієнтський додаток очікує на вхідні MQTT повідомлення. При отриманні повідомлення він перевіряє топік, розбирає вміст повідомлення та відображає його користувачу. Повідомлення також може бути збережене в історії, локально або віддалено.

Отже, взаємодія клієнтського додатка з MQTT брокером відбувається наступним чином: спочатку клієнтський додаток отримує від користувача дані для підключення до MQTT брокера (адреса, порт, ідентифікатор клієнта) та встановлює з'єднання. Потім додаток формує список топіків для підписки на основі ідентифікатора користувача та груп, до яких він належить, і відправляє MQTT повідомлення для підписки на ці топіки. Коли користувач вводить текст повідомлення та ідентифікатор одержувача, додаток формує MQTT повідомлення та публікує його на відповідний топік на MQTT брокері. Після цього клієнтський додаток очікує на вхідні MQTT повідомлення, перевіряє топік, розбирає вміст повідомлення та відображає його користувачу, з можливістю збереження повідомлення в історії локально або віддалено.

2.4.3 Розробка програмного коду для ESP8266

Далі, перейдемо до розробки програмного коду для ESP8266. Першим етапом є підключення до Wi-Fi.

Для підключення ESP8266 до Wi-Fi мережі необхідно використати бібліотеку wifi. У коді Lua потрібно вказати SSID (назву мережі) та пароль (рис. 2.7).

```
1  wifi.setmode(wifi.STATION)
2  wifi.sta.config("SSID", "password")
3  wifi.sta.connect()
```

Рисунок 2.7 – Підключення до Wi-Fi

Після підключення до Wi-Fi, ESP8266 повинен встановити з'єднання з MQTT брокером. Для підключення до MQTT брокера використовується бібліотека mqtt. У коді необхідно вказати адресу брокера, порт, ідентифікатор клієнта та, за необхідності, дані для аутентифікації (рис. 2.8).

```
m = mqtt.Client("client_id", 120, "username", "password")
m:connect("broker_address", broker_port, 0, function(client)
  print("Connected to MQTT broker")
end)
```

Рисунок 2.8 – Підключення до MQTT брокера

Для обробки вхідних повідомлень необхідно налаштувати функцію зворотного виклику, яка буде викликатися при отриманні повідомлення на підписаний топік (рис. 2.9).

```
m:on("message", function(client, topic, data)
    print(topic .. ": " .. data)
    -- Обробка отриманого повідомлення
  end)
```

Рисунок 2.9 – Обробка вхідних повідомлень

Для відправки повідомлень використовується функція publish (рис. 2.10).

```
m:publish(topic, message, 0, 0, function(client)
  print("Message sent")
end)
```

Рисунок 2.10 – Обробка вихідних повідомлень

Історію повідомлень можна зберігати на ESP8266 за допомогою файлової системи або віддалено на сервері. Для зберігання на ESP8266 можна використовувати бібліотеку file. Для віддаленого зберігання можна використовувати HTTP запити або MQTT повідомлення для передачі даних на сервер.

Програмне забезпечення відіграє ключову роль у функціонуванні системи обміну короткими повідомленнями на базі ESP8266 та MQTT. Вибір

прошивки NodeMCU обумовлений її гнучкістю та широкими можливостями програмування на Lua, що дозволяє реалізувати специфічні вимоги проєкту. Розробка програмного коду для ESP8266 включає в себе кілька важливих етапів: підключення до Wi-Fi мережі, встановлення зв'язку з MQTT брокером, обробку вхідних та вихідних повідомлень, а також зберігання історії повідомлень.

Підключення до Wi-Fi та MQTT брокера здійснюється за допомогою відповідних бібліотек NodeMCU, що забезпечує стабільну та надійну комунікацію. Обробка повідомлень включає в себе підписку на певні топіки, отримання та парсинг вхідних повідомлень, формування та публікацію вихідних повідомлень. Зберігання історії повідомлень може бути реалізовано як на локальному файловому сховищі ESP8266, так і на віддаленому сервері, залежно від вимог до обсягу зберігання та доступності даних.

2.5 Апаратне забезпечення

2.5.1 Вибір моделі ESP8266

Враховуючи вимоги до системи обміну короткими повідомленнями, такі як низька вартість, компактність, наявність Wi-Fi та достатня продуктивність для обробки текстових повідомлень, оптимальним вибором є ESP8266. Хоча він має обмежений обсяг пам'яті порівняно з Raspberry Pi, цього достатньо для зберігання невеликої історії повідомлень та виконання необхідних операцій з MQTT. Крім того, ESP8266 підтримує різні прошивки, що дозволяє вибрати оптимальне рішення для конкретних потреб.

Для реалізації системи обміну короткими повідомленнями можна використовувати різні моделі ESP8266, такі як ESP-01, ESP-12E (NodeMCU), ESP-32 тощо. Вибір конкретної моделі залежить від вимог до системи, таких як обсяг пам'яті, кількість доступних контактів GPIO, необхідність підключення додаткових периферійних пристроїв тощо.

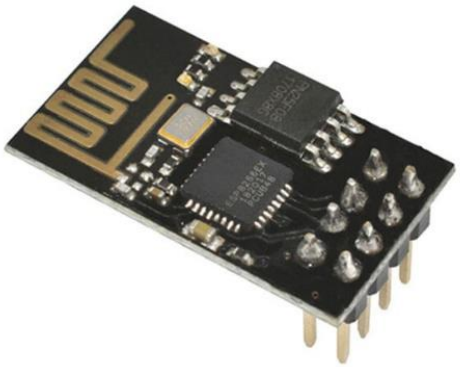


Рисунок 2.11 – Модуль ESP8266 ESP-01

Рисунок 2.12 – Модуль ESP8266 ESP-12E

Для розуміння різниці між модулями ESP8266 ESP-01 та ESP8266 ESP-12E було розроблено порівняльну таблицю (табл. 2.1), яка відображає основні технічні характеристики кожного з них. ESP8266 ESP-01, з одного боку, відзначається компактними розмірами та базовим набором функцій, що включає вбудовану PCB антену, UART і обмежену вбудовану пам'ять. У той же час, ESP8266 ESP-12E пропонує більш широкий функціональний набір, включаючи додаткові GPIO, SPI, I2C і можливість розширення функціоналу завдяки більшій кількості оперативної та вбудованої пам'яті. Це робить його більш універсальним рішенням для складніших проєктів IoT, де потрібна більша обробка даних чи підтримка різноманітних інтерфейсів.

Таблиця 2.1 – Порівняльна таблиця для модулів ESP8266 ESP-12E та ESP8266 ESP-01

Особливості	**ESP8266 ESP-01**	**ESP8266 ESP-12E**
Мікроконтролер	ESP8266EX	ESP8266EX
CPU	Tensilica L106 32-bit microcontroller	Tensilica L106 32-bit microcontroller
Частота CPU	80 MHz	80 MHz
Оперативна пам'ять (RAM)	32 КБ	128 КБ
Вбудована пам'ять (Flash)	512 КБ	4 МБ
Wi-Fi	802.11 b/g/n	802.11 b/g/n
Інтерфейси	UART	UART, GPIO, SPI, I2C
Антена	Вбудована PCB антена	Вбудована PCB антена
Розміри	14.2 x 24.8 мм	24 x 16 мм (модуль), 24 x 36 мм (PCB)
Підтримка прошивок	ESP8266 AT, NodeMCU, Tasmota, інші	ESP8266 AT, NodeMCU, Tasmota, інші

У даній роботі пропонується використовувати ESP8266-12E (NodeMCU), оскільки він має достатній обсяг пам'яті для зберігання прошивки та історії повідомлень, достатню кількість контактів GPIO для підключення

додаткових компонентів (якщо необхідно), а також вбудований USB-UART перетворювач, що спрощує процес програмування та налагодження.

2.5.2 Підключення додаткових компонентів

Для відображення повідомлень та іншої інформації користувачу можна підключити до ESP8266 дисплей. Найпоширенішими типами дисплеїв, що використовуються з ESP8266, є:

– перший, OLED дисплеї, які мають високу контрастність, низьке енергоспоживання та компактні розміри;

– другий, LCD дисплеї більш дешевші за OLED, але мають нижчу контрастність та вимагають підсвічування;

– третій, E-Paper дисплеї мають дуже низьке енергоспоживання та здатні зберігати зображення без підключення до живлення, але мають обмежену частоту оновлення.

Рисунок 2.13 – OLED дисплей

OLED (Organic Light-Emitting Diode) дисплеї відзначаються низьким енергоспоживанням, високою контрастністю та компактними розмірами, що робить їх особливо привабливими для використання в різних пристроях і системах. Основна перевага OLED полягає в тому, що кожен піксель самостійно світиться і не потребує підсвічування, що значно зменшує споживання електроенергії порівняно з традиційними LCD дисплеями. Це робить їх ідеальним вибором для пристроїв, які працюють в умовах

обмежених джерел живлення або потребують довготривалого автономного функціонування. Висока контрастність OLED дисплеїв забезпечує чітке і чітке відображення тексту і графіки при будь-яких умовах освітлення, що робить їх популярними вбудовуваним рішенням для різних промислових, наукових та споживчих застосувань.

Рисунок 2.14 – LCD дисплей

LCD (Liquid Crystal Display) дисплеї, незважаючи на свою більш низьку контрастність і вимогу до підсвічування, приваблюють користувачів завдяки більш доступній ціновій категорії. Вони залишаються популярними вибором для багатьох застосувань, де не потрібна висока контрастність, але важлива економія витрат на обладнання. LCD технологія використовує рідинні кристали для відображення зображень, які не самі по собі світяться, а тільки пропускають або блокують світло, що надходить від підсвітки ззаду. Це вимагає більшої кількості енергії для підсвічування і, відповідно, збільшує загальне споживання електроенергії. Незважаючи на це, LCD дисплеї залишаються важливими в більшості вбудованих систем, особливо там, де важлива економія і цінова доступність.

Рисунок 2.15 – E-paper дисплей

E-Paper дисплеї відомі своїм дуже низьким енергоспоживанням і унікальною здатністю зберігати зображення без подачі живлення, що робить їх ідеальними для застосувань, де важлива ефективність енергоспоживання. Ці дисплеї використовують технологію електрофорезу, що дозволяє змінювати кольори пікселів, не вимагаючи постійного живлення для підтримки зображення. Однак вони мають обмежену частоту оновлення, що робить їх менш практичними для динамічних відображень і вимагає вибору залежно від конкретного застосування.

Таблиця 2.2 – Порівняльна таблиця для дисплеїв

Особливості	**OLED дисплеї**	**LCD дисплеї**	**E-Paper дисплеї**
Енергоспоживання	Низьке	Високе	Дуже низьке
Підсвічування	Не потрібне	Потрібне	Не потрібне
Контрастність	Висока	Низька	Висока
Висока чіткість	Так	Так	Так
Частота оновлення	Висока	Висока	Низька
Зберігання зображення	Не зберігає	Не зберігає	Зберігає
Типи додатків	Динамічні	Всі	Статичні, динамічні
Споживання енергії	Високе	Високе	Низьке

Вибір типу дисплея залежить від вимог до якості зображення, енергоспоживання та вартості.

Кнопки можуть використовуватися для управління системою, наприклад, для перемикання між режимами роботи, відправки повідомлень, прокрутки історії повідомлень тощо. Кількість та тип кнопок залежать від конкретних вимог до інтерфейсу користувача.

2.5.3 Схема підключення

Схема підключення додаткових компонентів до ESP8266 залежить від конкретних моделей компонентів та їх інтерфейсів. Зазвичай, для підключення дисплеїв використовується інтерфейс I2C або SPI, для кнопок – GPIO, а для інших сенсорів та актуаторів – GPIO або інші інтерфейси, такі як UART або I2S.

Підключаємо BME680 модуль до ESP8266, як показано на наступній схематичній схемі, з контактом SDA, підключеним до GPIO 4 і контакт SCL, підключений до GPIO 5.

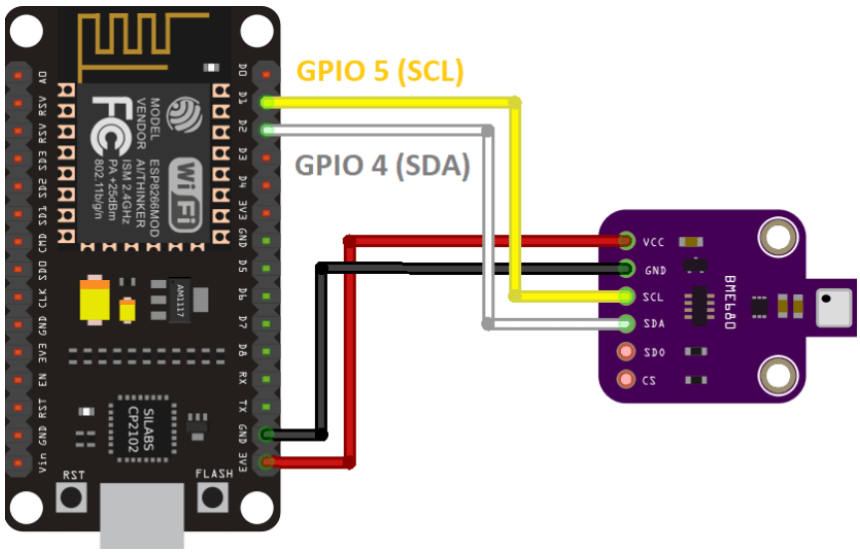


Рисунок 2.16 – Схема підключення

При розробці схеми підключення необхідно враховувати такі фактори:

1) наявність вільних контактів GPIO на ESP8266: Деякі моделі ESP8266 мають обмежену кількість контактів GPIO;

2) сумісність рівнів напруги: ESP8266 працює з рівнем напруги 3.3 В, тому необхідно забезпечити узгодження рівнів напруги при підключенні компонентів, які працюють з іншими рівнями (наприклад, 5 В);

3) споживання струму: Необхідно враховувати споживання струму всіх підключених компонентів, щоб не перевищити максимально допустимий струм для ESP8266.

Для спрощення підключення компонентів можна використовувати макетні плати або готові модулі розширення для ESP8266.

2.6 Протокол MQTT

MQTT (Message Queuing Telemetry Transport) - це легкий протокол обміну повідомленнями, спеціально розроблений для пристроїв з обмеженими ресурсами та ненадійними мережами, що робить його ідеальним для Інтернету речей (IoT). Він працює за моделлю «видавець-підписник», де відправники повідомлень («видавці») публікують дані на певні теми, а одержувачі («підписники») підписуються на ці теми, щоб отримувати відповідні повідомлення.

Основні характеристики MQTT:

1) легкість: MQTT використовує компактний двійковий формат повідомлень та мінімізує накладні витрати, що робить його ефективним для пристроїв з обмеженою пропускною здатністю та обчислювальними ресурсами;

2) асинхронність: MQTT підтримує асинхронний обмін повідомленнями, що означає, що відправник та одержувач не повинні бути одночасно підключені до мережі. Це забезпечує гнучкість та надійність у нестабільних мережевих умовах;

3) пропонує три рівні якості обслуговування (QoS) - 0 (максимальна швидкість), 1 (принаймні один раз) та 2 (рівно один раз), що дозволяє вибрати оптимальний баланс між надійністю доставки та ефективністю використання ресурсів;

4) структура топіків: MQTT використовує ієрархічну структуру топіків, що дозволяє організовувати та фільтрувати повідомлення за різними категоріями.

Отже, MQTT вирізняється своєю легкістю завдяки компактному двійковому формату повідомлень та мінімальним накладним витратам, що робить його ефективним для пристроїв з обмеженими ресурсами. Протокол підтримує асинхронний обмін повідомленнями, що забезпечує гнучкість та надійність у нестабільних мережевих умовах. MQTT пропонує три рівні якості обслуговування (QoS), що дозволяє вибрати оптимальний баланс між надійністю доставки та ефективністю використання ресурсів. Крім того, ієрархічна структура топіків дозволяє організовувати та фільтрувати повідомлення за різними категоріями.

2.6.1 Вибір MQTT брокера

MQTT брокер - це центральний компонент системи MQTT, який відповідає за отримання, зберігання та розповсюдження повідомлень між клієнтами. Існує безліч MQTT брокерів, як безкоштовних, так і комерційних, з різними функціями та можливостями.

При виборі MQTT брокера для системи обміну короткими повідомленнями на базі ESP8266 слід враховувати такі фактори:

1) легкість та ресурсоємність: Оскільки ESP8266 має обмежені ресурси, важливо вибрати брокер, який не вимагає багато пам'яті та обчислювальної потужності;

2) підтримка QoS: Брокер повинен підтримувати всі три рівні QoS, щоб забезпечити надійну доставку повідомлень;

3) безпека: Якщо система вимагає високого рівня безпеки, слід вибрати брокер, який підтримує аутентифікацію, авторизацію та шифрування SSL/TLS;

4) масштабованість: Якщо система планується масштабувати в майбутньому, важливо вибрати брокер, який може впоратися зі збільшенням кількості клієнтів та повідомлень.

Для даної системи пропонується використовувати Mosquitto - безкоштовний MQTT брокер з відкритим вихідним кодом, який відповідає всім перерахованим вище вимогам. Він легкий, підтримує всі рівні QoS, має засоби забезпечення безпеки та може бути масштабований при необхідності.

2.6.2 Налаштування MQTT топіків

MQTT топіки - це текстові рядки, які використовуються для ідентифікації повідомлень та визначення, які клієнти повинні їх отримувати. У системі обміну короткими повідомленнями топіки можуть використовуватися для ідентифікації користувачів або груп користувачів.

Наприклад, можна використовувати наступну структуру топіків:

1) /user/\<ідентифікатор користувача>: Для відправки повідомлень конкретному користувачу;

2) /group/\<назва групи>: Для відправки повідомлень групі користувачів.

ESP8266 повинен підписатися на топіки, які відповідають його ідентифікатору та групам, до яких він належить. При відправці повідомлення клієнтський пристрій повинен опублікувати його на відповідний топік.

2.6.3 Обробка MQTT повідомлень

Обробка MQTT повідомлень на ESP8266 включає наступні кроки:

1) отримання повідомлення: ESP8266 отримує повідомлення від MQTT брокера на підписаний топік;

2) перевірка топіка: ESP8266 визначає, чи є повідомлення призначеним для нього (тобто, чи відповідає топік повідомлення ідентифікатору користувача або групі, до якої він належить);

3) розбір повідомлення: Якщо повідомлення призначене для ESP8266, він розбирає його вміст (текст повідомлення, відправник тощо);

4) виконання дії: Залежно від типу повідомлення та його вмісту, ESP8266 може виконати певну дію, наприклад: відобразити повідомлення на дисплеї; відтворити звуковий сигнал; зберегти повідомлення в історії; відправити відповідь.

Обробка повідомлень може бути реалізована за допомогою функцій зворотного виклику, які викликаються при отриманні повідомлення на підписаний топік

Висновки до розділу 2

У цьому розділі було розглянуто основні аспекти проєктування системи обміну короткими повідомленнями на базі ESP8266 та MQTT. Було проаналізовано різні варіанти прошивок для ESP8266, включаючи Tasmota, ESPEasy та NodeMCU, та обрано NodeMCU як найбільш підходящу для даного проєкту завдяки її гнучкості та можливостям програмування на Lua.

Було описано процес розробки програмного коду для ESP8266, включаючи підключення до Wi-Fi мережі, підключення до MQTT брокера, обробку вхідних та вихідних повідомлень, а також зберігання історії повідомлень. Розглянуто різні варіанти зберігання історії повідомлень, такі як локальне зберігання на ESP8266 та віддалене зберігання на сервері.

Також було проаналізовано апаратне забезпечення, необхідне для реалізації системи, включаючи вибір моделі ESP8266 та підключення додаткових компонентів, таких як дисплеї, кнопки та інші сенсори або актуатори. Було розглянуто особливості використання протоколу MQTT, включаючи вибір MQTT брокера, налаштування MQTT топіків та обробку MQTT повідомлень.

Нарешті, було розглянуто важливі аспекти безпеки системи, такі як захист Wi-Fi мережі, аутентифікація та авторизація користувачів, а також шифрування повідомлень. Було запропоновано використовувати різні методи

захисту даних, такі як використання сильних паролів, WPA2/WPA3 шифрування, аутентифікація за допомогою логіна та пароля або токенів, та шифрування повідомлень за допомогою SSL/TLS.

Проведене дослідження та аналіз дозволяють зробити висновок про те, що ESP8266 та MQTT є потужними інструментами для створення системи обміну короткими повідомленнями. Завдяки своїй гнучкості, низькій вартості та широким можливостям, ці технології дозволяють створювати ефективні та надійні системи, які можуть бути використані в різних сферах застосування.

3 АПАРАТНО-ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Реалізації на базі ESP8266 та MQTT

Існують різні реалізації систем обміну короткими повідомленнями на базі ESP8266 та MQTT, кожна з яких має свої особливості та переваги. Деякі з них використовують готові прошивки, такі як Tasmota та ESPEasy, які спрощують процес налаштування та надають зручний веб-інтерфейс для управління системою. Інші реалізації можуть базуватися на написанні власного коду на мові Lua з використанням платформи NodeMCU, що надає більшу гнучкість, але вимагає певних навичок програмування.

Реалізація на базі ESP8266:

1) використовує популярну прошивку Tasmota, яка має вбудовану підтримку MQTT та широкий набір функцій. Вона дозволяє легко налаштувати ESP8266 для роботи як MQTT клієнт та публікувати повідомлення на певні топіки. Tasmota також надає зручний веб-інтерфейс для моніторингу та управління пристроєм;

2) також, ESPEasy - це ще одна популярна прошивка для ESP8266, яка також підтримує MQTT. Вона має більш простий інтерфейс, ніж Tasmota, та підходить для початківців. ESPEasy дозволяє легко підключати різні сенсори та актуатори до ESP8266 та управляти ними через MQTT;

3) прошивка на базі Lua, яка надає розробникам більше гнучкості у створенні власних рішень. За допомогою NodeMCU можна написати власний код для обробки MQTT повідомлень та реалізації специфічної логіки роботи системи;

4) можна програмувати за допомогою Arduino IDE, використовуючи бібліотеку PubSubClient для роботи з MQTT. Цей підхід надає розробникам знайоме середовище розробки та широкий вибір бібліотек для роботи з різними компонентами.

Окрім ESP8266, існує багато інших апаратних та програмних платформ, які можна використовувати для створення систем обміну короткими повідомленнями з підтримкою MQTT.

3.1.1 Реалізація на базі Arduino та Ethernet/Wi-Fi Shield та MQTT

Arduino, поряд з ESP8266, є ще однією популярною платформою для створення систем обміну короткими повідомленнями (SOMP) на базі MQTT. Нижче розглянемо детальніше компоненти та особливості цієї платформи.

Arduino - це апаратна обчислювальна платформа з відкритим вихідним кодом, заснована на мікроконтролерах сімейства AVR та інших. Вона широко використовується в освітніх цілях, хобі-проєктах та професійній розробці завдяки своїй простоті, доступності та широким можливостям. Вона пропонує широкий вибір мікроконтролерів з різними характеристиками, такими як Arduino Uno (ATmega328P), Nano (ATmega328P), Mega (ATmega2560) та інші. Кожен мікроконтролер має свої особливості щодо обчислювальної потужності, обсягу пам'яті та кількості входів/виходів.

Рисунок 3.1 – Плата Arduino

Arduino IDE надає велику кількість бібліотек, які спрощують роботу з різними компонентами (сенсорами, дисплеями, моторами тощо) та

протоколами (MQTT, HTTP, SPI, I2C тощо). Це значно прискорює та полегшує процес розробки.

Оскільки Arduino не має вбудованого модуля Wi-Fi, для підключення до мережі та взаємодії з MQTT брокером використовуються спеціальні модулі розширення - Ethernet або Wi-Fi Shield. Ethernet Shield підключається до Arduino через спеціальний роз'єм та дозволяє підключатися до мережі Ethernet за допомогою кабелю.

PubSubClient – це бібліотека для Arduino, яка спрощує роботу з MQTT протоколом. Вона надає зручний інтерфейс для підключення до MQTT брокера, публікації повідомлень на топіки та підписки на топіки для отримання повідомлень.

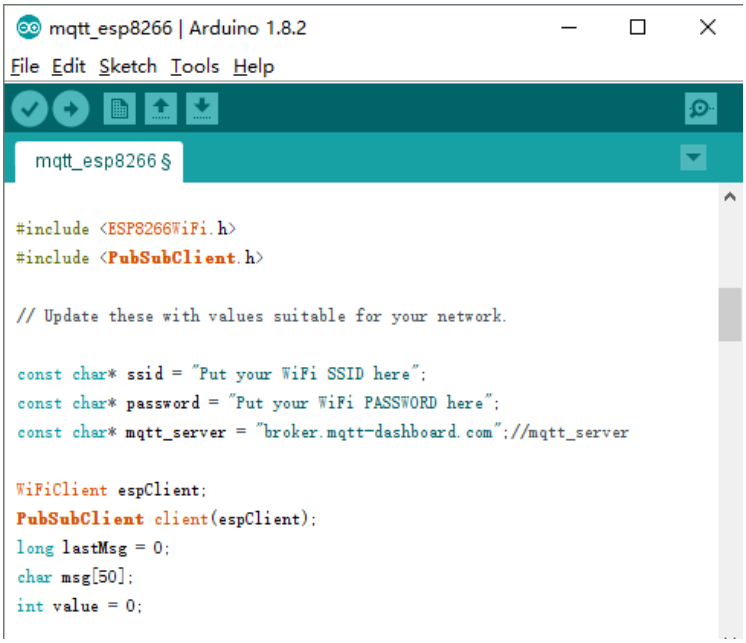


Рисунок 2.2 – Використання PubSubClient

PubSubClient дозволяє встановлювати з'єднання з MQTT брокером, відправляти та отримувати повідомлення, обробляти підтвердження доставки повідомлень та інші функції, необхідні для роботи з MQTT.

Переваги використання Arduino для системи коротких повідомлень:

1) наявність великої кількості бібліотек значно спрощують процес розробки;

2) має велику та активну спільноту користувачів та розробників, які готові допомогти та поділитися досвідом;

3) існує безліч готових бібліотек та прикладів коду для Arduino, які можна використовувати для створення SOMP.

Недоліки використання Arduino для SOMP:

1) мікроконтролери Arduino мають обмежений обсяг пам'яті та обчислювальну потужність порівняно з ESP8266, що може обмежувати функціональність та продуктивність системи;

2) для підключення до мережі Wi-Fi потрібно використовувати додатковий Wi-Fi Shield, що збільшує вартість та складність системи.

Загалом, Arduino є гарним вибором для створення простих системи обміну короткими повідомленнями з обмеженим функціоналом. Якщо ж потрібна більша продуктивність та гнучкість, варто розглянути інші платформи, такі як ESP8266 або Raspberry Pi.

3.1.2 Реалізації на базі Raspberry Pi та MQTT

Raspberry Pi, як потужний одноплатний комп'ютер, надає значні переваги для створення систем обміну короткими повідомленнями з використанням MQTT. Raspberry Pi – це серія одноплатних комп'ютерів, розроблених Raspberry Pi Foundation. Вони мають достатньо обчислювальної потужності та пам'яті для виконання різноманітних завдань, включаючи роботу в якості MQTT брокера або клієнта.

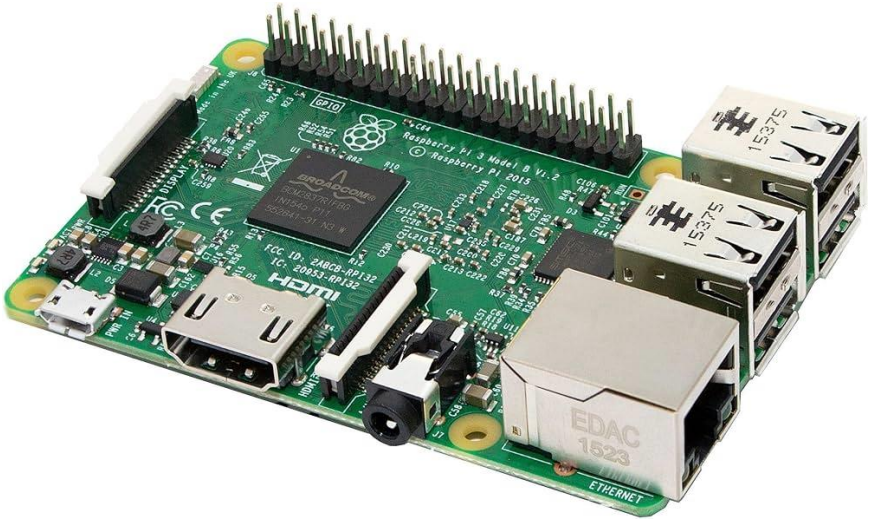


Рисунок 3.3 – Плати Raspberry Pi

Існує кілька моделей Raspberry Pi з різними характеристиками, такими як Raspberry Pi 4 Model B (з 1 Гбайт, 2 Гбайт, 4 Гбайт або 8 Гбайт оперативної пам'яті), Raspberry Pi 3 Model B+, Raspberry Pi Zero W тощо. Вибір моделі залежить від вимог до продуктивності та функціональності системи.

Raspberry Pi має широкий набір інтерфейсів, включаючи USB, Ethernet, Wi-Fi, Bluetooth (у деяких моделях), GPIO (General Purpose Input/Output) та інші. Це дозволяє підключати різноманітні периферійні пристрої, такі як дисплеї, клавіатури, сенсори, актуатори тощо.

Також Raspberry Pi підтримує різні операційні системи, засновані на Linux, такі як Raspbian (офіційна операційна система), Ubuntu, DietPi та інші. Це надає розробникам гнучкість у виборі середовища розробки та інструментів.

Mosquitto – це один з найпопулярніших MQTT брокерів з відкритим вихідним кодом. Він легкий, надійний та простий у налаштуванні, що робить його ідеальним вибором для використання на Raspberry Pi. Mosquitto підтримує всі основні функції MQTT, включаючи публікацію/підписку, QoS, зберігання повідомлень, аутентифікацію та авторизацію. Також Mosquitto підтримує шифрування SSL/TLS для забезпечення безпеки з'єднання між клієнтами та брокером.

Paho MQTT - це набір бібліотек MQTT з відкритим вихідним кодом, доступних для різних мов програмування, включаючи Python. Бібліотека Paho MQTT для Python дозволяє легко створювати MQTT клієнтів на Raspberry Pi. Paho MQTT надає функції для підключення до MQTT брокера, публікації та підписки на топіки, обробки повідомлень тощо.

Переваги використання Raspberry Pi:

– перша, Raspberry Pi має потужний процесор та достатній обсяг оперативної пам'яті для обробки великої кількості повідомлень та виконання складних завдан;

– друга, Raspberry Pi має різноманітні інтерфейси, що дозволяють підключати різні периферійні пристрої та розширювати функціональність системи;

– третя, можливість використання різних операційних систем та мов програмування.

Недоліки використання Raspberry Pi:

– дорожчий за ESP8266;

– споживає більше енергії, ніж ESP8266, що може бути важливим фактором для портативних або автономних пристроїв.

Загалом, Raspberry Pi є відмінним вибором для створення потужних та функціональних систем з підтримкою MQTT. Він надає розробникам широкі можливості для реалізації складних сценаріїв та інтеграції з іншими системами. Однак, якщо важливим фактором є вартість або енергоспоживання, то ESP8266 може бути більш підходящим варіантом.

3.2 Реалізація програмного забезпечення

У цьому розділі буде детально описано реалізацію програмного коду для ESP8266 з використанням прошивки NodeMCU та мови програмування Lua. Код буде відповідати за підключення до Wi-Fi мережі, взаємодію з MQTT

брокером, обробку вхідних та вихідних повідомлень, а також зберігання історії повідомлень.

3.2.1 Підключення до Wi-Fi

Для підключення ESP8266 до Wi-Fi мережі використовується бібліотека wifi. У коді Lua спочатку встановлюється режим роботи станції (клієнта), потім налаштовуються параметри підключення, такі як SSID (назва мережі) та пароль.

```
wifi.setmode(wifi.STATION) -- Встановлення режиму станції (клієнта)
wifi.sta.config("SSID", "home11", "password", "ieh5wia5") -- Підключення до Wi-Fi
wifi.sta.connect() -- Підключення до мережі

tmr.alarm(1, 1000, 1, function() -- Перевірка статусу підключення кожну секунду
  if wifi.sta.getip() == nil then
    print("Connecting to AP...")
  else
    tmr.stop(1) -- Зупинка таймера після успішного підключення
    print("IP address: ", wifi.sta.getip())
    -- Далі йде код для підключення до MQTT та інших дій
  end
end)
```

Рисунок 3.4 – Підключення до Wi-Fi мережі

Після цього відбувається спроба підключення до мережі. Для перевірки успішності підключення використовується таймер, який кожну секунду перевіряє, чи отримано IP-адресу. У разі успішного підключення таймер зупиняється та виводиться повідомлення з IP-адресою. Якщо підключення не вдалося, таймер продовжує перевірку до успішного результату або досягнення максимальної кількості спроб.

3.2.2 Підключення до MQTT брокера

Після встановлення Wi-Fi з'єднання ESP8266 підключається до MQTT брокера. Для цього використовується бібліотека mqtt. У коді створюється об'єкт MQTT клієнта з унікальним ідентифікатором (client_id), часом життя сесії (keepalive) та, за необхідності, ім'ям користувача (username) та паролем (password).

```
mqtt = require("mqtt") -- Підключення бібліотеки MQTT

-- Налаштування MQTT клієнта
client = mqtt.Client("ESP8266_Client", 120) -- client_id, keepalive
broker_ip = "YOUR_MQTT_BROKER_IP"
broker_port = 1883

-- Функція зворотного виклику при підключенні до MQTT брокера
client:on("connect", function(client)
  print("Connected to MQTT broker")
  -- Підписка на топіки
  client:subscribe("topic/in",0, function(client) print("Subscribe success") end)
end)

-- Функція зворотного виклику при отриманні повідомлення
client:on("message", function(client, topic, data)
  print(topic .. ": " .. data)
  -- Обробка отриманого повідомлення
end)

-- Підключення до MQTT брокера
client:connect(broker_ip, broker_port, 0, 1, function(client) print("Connection failed!") end)
```

Рисунок 3.5 – Підключення до MQTT брокера

Потім викликається метод connect для встановлення з'єднання з брокером за вказаною адресою (broker_address) та портом (broker_port). У разі успішного підключення викликається функція зворотного виклику, яка може виконати додаткові дії, такі як підписка на топіки.

3.2.3 Обробка вхідних та вихідних повідомлень

Для обробки вхідних повідомлень використовується механізм підписки на топіки. ESP8266 підписується на певні топіки, які відповідають ідентифікаторам користувачів або груп. При отриманні повідомлення на підписаний топік викликається функція зворотного виклику on(«message»), яка обробляє отримані дані.

```
function sendMessage(topic, message)
  client:publish(topic, message, 0, 0, function(client)
    print("Sent: " .. message .. " to topic: " .. topic)
  end)
end

-- Приклад відправки повідомлення
sendMessage("topic/out", "Hello from ESP8266!")
```

Рисунок 3.6 – Обробка повідомлень

Обробка повідомлень може включати аналіз вмісту повідомлення, його збереження, відображення на дисплеї або виконання інших дій, залежно від логіки програми.

Для відправки повідомлень використовується метод publish. У коді вказується топік, на який потрібно опублікувати повідомлення, та саме повідомлення. Опціонально можна вказати рівень якості обслуговування (QoS) та інші параметри. Після успішної публікації повідомлення може бути викликана функція зворотного виклику.

3.2.4 Зберігання історії повідомлень

Історію повідомлень можна зберігати локально на ESP8266 або віддалено на сервері. В даному випадку використовується локальне зберігання.

```
69  file.open("message_history.txt", "a+") -- Відкриття файлу в режимі додавання
70  file.writeline(message) -- Запис повідомлення у файл
71  file.close() -- Закриття файлу
```

Рисунок 3.7 – Збереження історії повідомлень

Отримавши нове повідомлення, можна використати бібліотеку file для відкриття файлу історії на файловій системі SPIFFS, додати нове повідомлення в кінець файлу за допомогою функції запису, а потім закрити файл. Такий підхід дозволяє зручно зберігати та оновлювати історію локально на ESP8266.

3.3 Алгоритми роботи прошивки NodeMCU

Блок-схема описує алгоритм роботи прошивки NodeMCU в системі обміну короткими повідомленнями на базі MQTT. Розглянемо детально кожен крок:

1) ініціалізація системи: на цьому етапі відбувається запуск NodeMCU та ініціалізація всіх необхідних компонентів. Встановлюються

початкові налаштування, такі як режим роботи Wi-Fi, параметри підключення до MQTT брокера тощо;

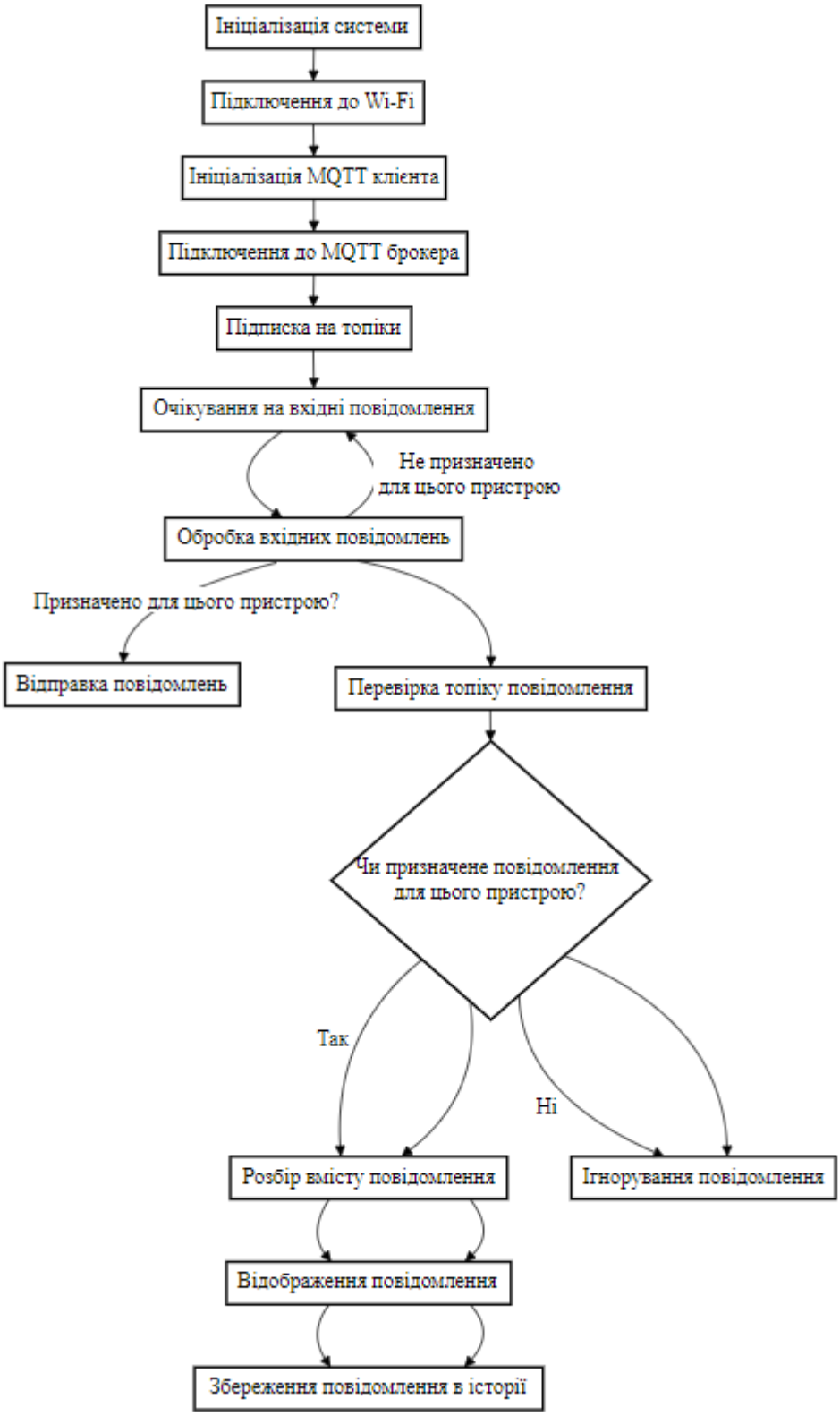


Рисунок 3.8 – Блок-схема алгоритму роботи прошивки

2) підключення до Wi-Fi: NodeMCU активує Wi-Fi модуль та намагається встановити з'єднання з заданою Wi-Fi мережею. Якщо підключення успішне, переходить до наступного кроку. Якщо ні, повторює спробу підключення або переходить у режим очікування/помилки;

3) ініціалізація MQTT клієнта: NodeMCU створює екземпляр MQTT клієнта, вказуючи унікальний ідентифікатор, параметри підключення до брокера (адреса, порт) та інші налаштування (наприклад, логін/пароль, якщо потрібні);

4) підключення до MQTT брокера: NodeMCU намагається встановити з'єднання з MQTT брокером, використовуючи налаштування, вказані на попередньому кроці. Якщо підключення успішне, переходить до наступного кроку. Якщо ні, повторює спробу підключення або переходить у режим очікування/помилки;

5) підписка на топіки: NodeMCU підписується на певні MQTT топіки, які відповідають ідентифікатору користувача або групам, до яких він належить. Це дозволить йому отримувати повідомлення, опубліковані на цих топіках;

6) очікування на вхідні повідомлення: NodeMCU переходить у режим очікування нових повідомлень від MQTT брокера;

7) обробка вхідних повідомлень: при отриманні повідомлення NodeMCU перевіряє топік, на який воно було надіслане;

8) перевірка топіку повідомлення: система перевіряє, чи призначене отримане повідомлення для цього пристрою, порівнюючи топік повідомлення з ідентифікатором користувача або групами, на які підписаний пристрій;

9) чи призначене повідомлення для цього пристрою?: якщо повідомлення призначене для цього пристрою, переходить до кроку 10. Якщо ні, ігнорує повідомлення та повертається до кроку 6 (очікування нових повідомлень);

10) розбір вмісту повідомлення: NodeMCU витягує з повідомлення текст, інформацію про відправника та інші дані;

11) відображення повідомлення: повідомлення виводиться користувачу на дисплей (якщо є) або іншим способом (наприклад, через послідовний порт);

12) збереження повідомлення в історії: повідомлення зберігається в історії, або локально на ESP8266, або віддалено на сервері;

13) відправка повідомлень: цей крок виконується незалежно від отримання повідомлень. Користувач вводить текст повідомлення та ідентифікатор одержувача. NodeMCU формує MQTT повідомлення та публікує його на відповідний топік.

Після відправки повідомлення або обробки вхідного повідомлення, NodeMCU повертається до кроку 6 (очікування нових повідомлень), і цикл повторюється.

Отже, взаємодія NodeMCU з MQTT брокером складається з наступних кроків: спочатку відбувається ініціалізація системи, де встановлюються початкові налаштування, такі як режим роботи Wi-Fi та параметри підключення до MQTT брокера. Далі NodeMCU підключається до Wi-Fi мережі, а після успішного підключення створює екземпляр MQTT клієнта з унікальним ідентифікатором та параметрами підключення. Потім NodeMCU намагається встановити з'єднання з MQTT брокером і, у разі успіху, підписується на певні топіки. Після цього пристрій переходить у режим очікування нових повідомлень, перевіряє топіки отриманих повідомлень і, якщо вони призначені для цього пристрою, розбирає їх вміст, відображає користувачу і зберігає в історії. Відправка повідомлень відбувається незалежно від отримання: користувач вводить текст ідентифікатор одержувача, після чого NodeMCU формує MQTT повідомлення та публікує його на відповідний топік.

3.4 Реалізація коротких повідомлень з модулем BME680

3.4.1 Загальна схема пристрою

На схемі зображено підключення модуля BME680 до мікроконтролера ESP8266 (модель NodeMCU). BME680 - це багатофункціональний датчик, який вимірює температуру, вологість, тиск та якість повітря (газовий опір).

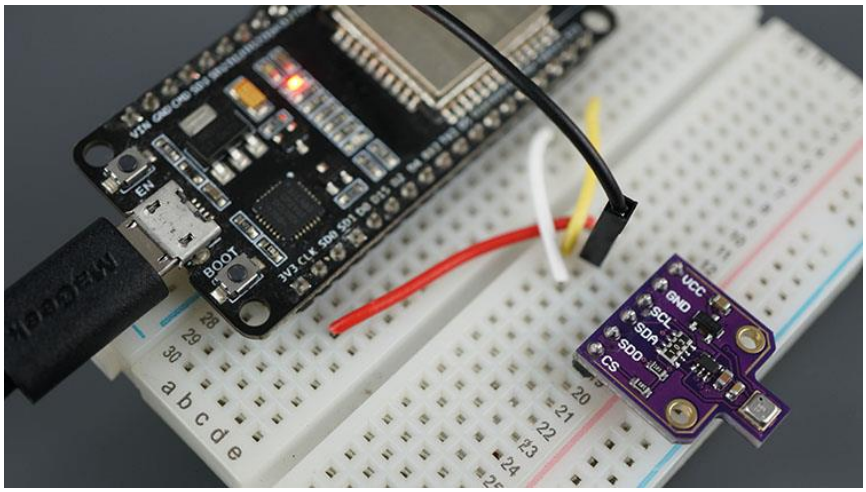


Рисунок 3.9 – Загальна схема пристрою

Підключаємо BME680 модуль до ESP8266, як показано на наступній схематичній схемі, з контактом SDA, підключеним до GPIO 4 і контакт SCL, підключений до GPIO 5.

3.4.2 Опис роботи основного коду

Спочатку додаємо облікові дані Wi-Fi, адресу MQTT брокера та ідентифікатор клієнта MQTT.

```
9   #define WIFI_SSID "ssd"
10  #define WIFI_PASSWORD "password"
```

Рисунок 3.10 – Додавання облікових даних

Далі, додаємо адресу для підключення до брокера та визначаємо порт MQTT.

```
12  #define MQTT_HOST IPAddress(192, 168, 1, 106)

16  #define MQTT_PORT 1883
```

Рисунок 3.11 – Додавання адреси для підключення до брокера та визначення порту MQTT

```
18  #define MQTT_PUB_TEMP "esp/bme680/temperature"
19  #define MQTT_PUB_HUM  "esp/bme680/humidity"
20  #define MQTT_PUB_PRES "esp/bme680/pressure"
21  #define MQTT_PUB_GAS  "esp/bme680/gas"
```

Рисунок 3.12 – Публікація температури, вологості та тиску в наступних топіках

```
28  AsyncMqttClient mqttClient;
29  Ticker mqttReconnectTimer;
30
31  WiFiEventHandler wifiConnectHandler;
32  WiFiEventHandler wifiDisconnectHandler;
33  Ticker wifiReconnectTimer;
```

Риснок 3.13 – Створення AsyncMqttClient для обробки MQTT та таймерів для повтороного підключення до брокера та маршрутизатора

AsyncMqttClient – це бібліотека, яка спрощує роботу з MQTT на ESP8266. Вона дозволяє асинхронно підключатися до MQTT брокера, публікувати та отримувати повідомлення, а також обробляти MQTT події.

Таймери використовуються для повторного підключення до брокера та маршрутизатора у разі втрати зв'язку. Якщо ESP8266 не може підключитися до брокера або маршрутизатора, таймер запускається і повторно викликає відповідні функції підключення через певний інтервал часу. Це забезпечує автоматичне відновлення зв'язку у разі тимчасових проблем з мережею.

3.4.3 Функції MQTT: підключення до Wi-Fi, підключення до MQTT та події Wi-Fi

Функція connectToWifi() відповідає за підключення ESP8266 до маршрутизатора. Якщо ESP8266 не може підключитися до маршрутизатора, він запускає таймер WiFiReconnectTimer і повторно викликає функцію connectToWifi() кожні 5 секунд.

```
40  void connectToMqtt() {
41    Serial.println("Connecting to MQTT…");
42    mqttClient.connect();
43  }
44
45  void onWifiConnect(const WiFiEventStationModeGotIP& event) {
46    Serial.println("Connected to Wi-Fi.");
47    connectToMqtt();
48  }
```

Рисунок 3.14 – Підключення ESP8266 до маршрутизатора та брокера MQTT

Після успішного підключення до маршрутизатора ESP8266 підключається до брокера MQTT, викликаючи connectToMqtt().

Функції onWifiConnect() і onWifiDisconnect() відповідають за обробку подій Wi-Fi.

```
45  void onWifiConnect(const WiFiEventStationModeGotIP& event) {
46    Serial.println("Connected to Wi-Fi.");
47    connectToMqtt();
48  }
49
50  void onWifiDisconnect(const WiFiEventStationModeDisconnected& event) {
51    Serial.println("Disconnected from Wi-Fi.");
52    mqttReconnectTimer.detach();
53    wifiReconnectTimer.once(2, connectToWifi);
54  }
```

Рисунок 3.15 – Обробка подій Wi-Fi

Наприклад, після успішного підключення до маршрутизатора та брокера MQTT він друкує IP-адресу ESP8266. З іншого боку, якщо з'єднання втрачено, він запускає таймер і намагається підключитися повторно.

```
56  void onMqttConnect(bool sessionPresent) {
57    Serial.println("Connected to MQTT.");
58    Serial.print("Session present: ");
59    Serial.println(sessionPresent);
60  }
```

Рисунок 3.16 – Початок сеансу з брокером

Таймер, який запускається кожні 10 секунд викликає функцію getBME680Readings(), яка отримує показання датчика BME680 і публікує їх у відповідних темах MQTT.

```
62  unsigned long currentMillis = millis();
63  // Every X number of seconds (interval = 10 seconds)
64  // it publishes a new MQTT message
65  if (currentMillis - previousMillis >= interval) {
66    // Save the last time a new reading was published
67    previousMillis = currentMillis;
68
69    getBME680Readings();
70    Serial.println();
71    Serial.printf("Temperature = %.2f °C \n", temperature);
72    Serial.printf("Humidity = %.2f % \n", humidity);
73    Serial.printf("Pressure = %.2f hPa \n", pressure);
74    Serial.printf("Gas Resistance = %.2f KOhm \n", gasResistance);
```

Рисунок 3.17 – Створення таймеру для отримання даних з датчику BME680

```
77  uint16_t packetIdPub1 = mqttClient.publish(MQTT_PUB_TEMP, 1, true, String(temperature).c_str());
78  uint16_t packetIdPub2 = mqttClient.publish(MQTT_PUB_HUM, 1, true, String(humidity).c_str());
79  uint16_t packetIdPub3 = mqttClient.publish(MQTT_PUB_PRES, 1, true, String(pressure).c_str());
80  uint16_t packetIdPub4 = mqttClient.publish(MQTT_PUB_GAS, 1, true, String(gasResistance).c_str());
```

Рисунок 3.18 – Публікація в топіках

В основному використовується publish()метод на mqttClient об'єкт для публікації даних по темі. Метод publish() приймає такі аргументи в порядку:

– topic (const char*);

– якість обслуговування – може бути 0, 1 або 2;

– retain flag (bool): зберегти прапор;

– payload (const char*) – у цьому випадку корисне навантаження відповідає показанню датчика.

QoS (якість обслуговування) – це спосіб гарантувати доставку повідомлення. Це може бути один із наступних рівнів:

– 0 : повідомлення буде доставлено один раз або не буде доставлено взагалі. Повідомлення не підтверджено.

– 1 : повідомлення буде доставлено принаймні один раз, але може бути доставлено більше одного разу;

– 2 : повідомлення завжди доставляється рівно один раз.

3.4.4 Завантаження коду в ESP8266 та перевірка брокеру MQTT

Запустивши брокер MQTT, завантажуємо код на мікроконтроллер ESP8266.

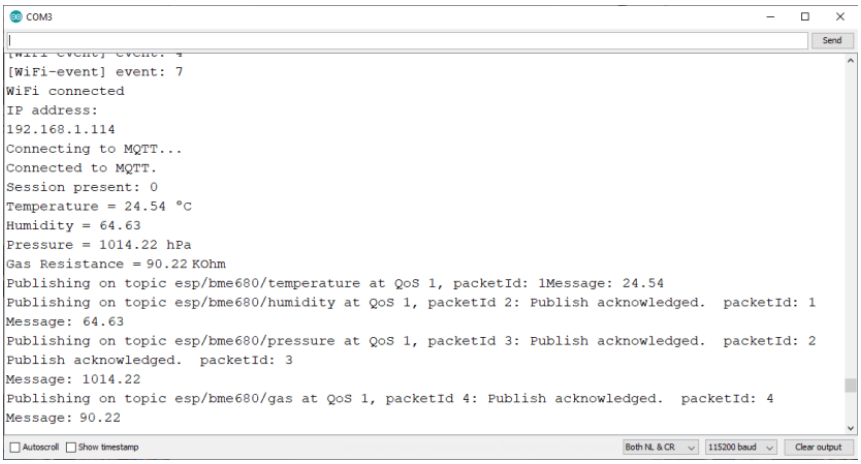


Рисунок 3.19 – Публікація ESP8266 повідомлень на топіки

Відкриваємо Serial Monitor та бачимо, що ESP8266 починає публікувати повідомлення на топіки, які було визначено раніше.

3.4.5 Підготовка інформаційної панелі Node-RED

ESP8266 публікує показання датчиків кожні 10 секунд за чотирма топіками MQTT. Тепер щоб побачити результат було обрано будь-яку інформаційну панель, яка підтримує MQTT, або будь-який інший пристрій, який підтримує MQTT, щоб підписатися на ці топіки та отримувати показання.

Було створено простий потік за допомогою Node-RED, щоб підписатися на ці теми та відобразити показання на датчиках.

Node-RED надає широкий спектр інструментів для візуалізації та обробки даних. Користувач може додавати різні вузли (nodes) на інформаційну панель для відображення даних у різних форматах, таких як графіки, таблиці, індикатори тощо.

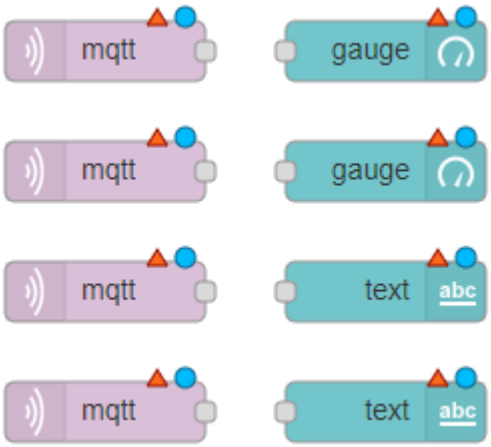


Рисунок 3.20 – Відкриття інтерфейсу Node-RED

Перетягуємо чотири вузли MQTT in, два вузли вимірювання та два вузли текстового поля до потоку.

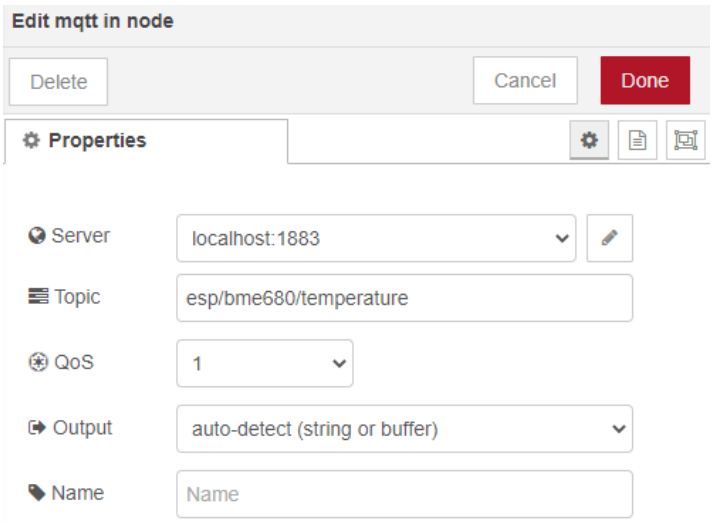


Рисунок 3.21 – Налаштування властивостей вузлу MQTT

Поле «Сервер» стосується брокера MQTT. У нашому випадку брокером MQTT є ESP8266, тому для нього встановлено localhost:1883.

Вставляємо топік, на який хочемо підписатися, і QoS. Цей попередній вузол MQTT підписаний на тему esp/bme680/temperature.

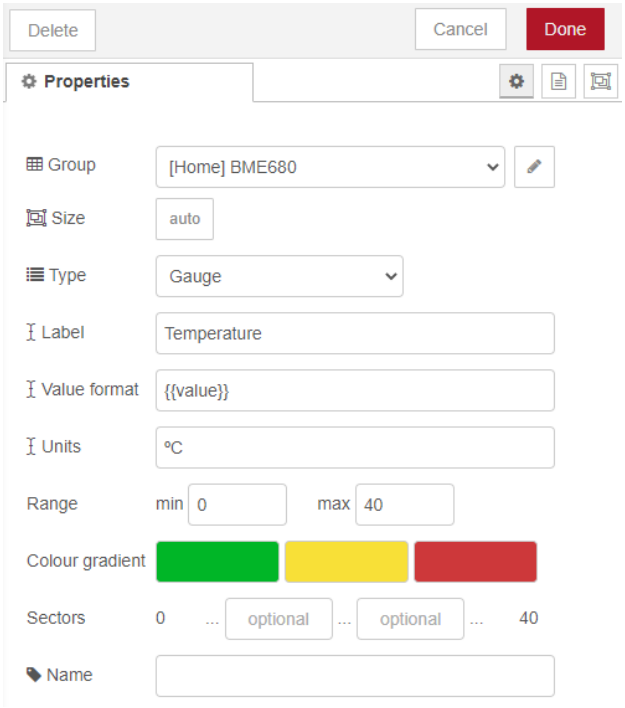


Рисунок 3.22 – Редагування властивостей для кожного показання BME680

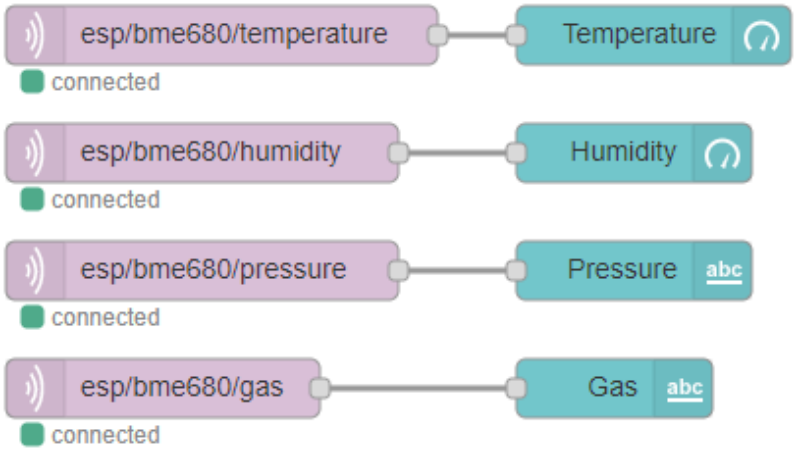


Рисунок 3.23 – Підключення вузлів

3.4.6 Демонстрація роботи

Далі, переходимо по локальній адресі нашого брокеру. Отримавши доступ до поточних показань датчика BME680 на приладовій панелі. Далі,

можемо використовувати інші вузли інформаційної панелі для відображення показань різними способами.

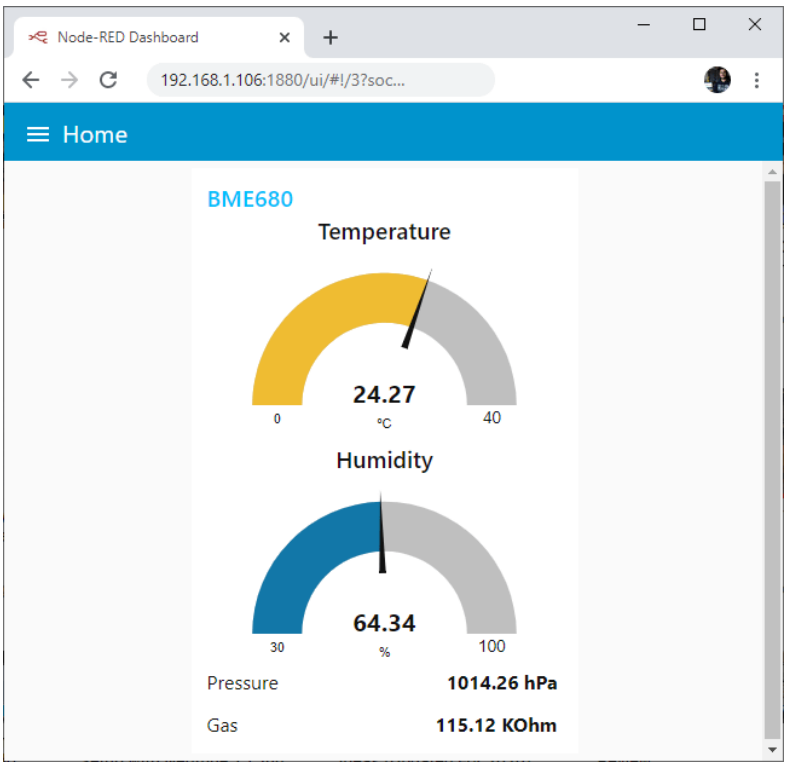


Рисунок 3.24 – Публікація показання температури, вологості, тиску, та газового опору BME680 на Node-RED через MQTT

Вся взаємодія між ESP8266 та Node-RED відбувається через протокол MQTT. ESP8266 публікує показання датчика на певні топіки MQTT, а Node-RED підписується на ці топіки та отримує дані для відображення на інформаційній панелі.

Таким чином, Node-RED надає зручний та гнучкий інструмент для візуалізації та аналізу даних, отриманих з датчиків через MQTT. Це дозволяє створювати інтерактивні інформаційні панелі, які можна використовувати для моніторингу та керування різними системами та пристроями.

Висновки до розділу 3

У розділі 3 детально описано систему обміну короткими повідомленнями, розроблену на базі ESP8266 та MQTT. Розглянуто архітектуру системи з усіма важливими компонентами, включаючи ESP8266

модуль, який відповідає за збір і передачу даних, та MQTT брокер, що координує комунікацію між пристроями. Кожен крок взаємодії, починаючи від підключення до Wi-Fi мережі до підписки на відповідні топіки для обміну повідомленнями, детально розглянуто для забезпечення стабільності та ефективності системи.

Програмне забезпечення системи реалізоване з використанням прошивки NodeMCU, що дозволяє гнучко програмувати та інтегрувати різноманітні функції. Висвітлені алгоритми роботи включають всі необхідні кроки взаємодії, від установки зв'язку до обміну та зберігання історії повідомлень. Крім того, надано приклад інтеграції датчика BME680, що дозволяє системі передавати дані про температуру, вологість, тиск та якість повітря через MQTT.

Детально описано процес підключення та налаштування датчика BME680 до ESP8266, зокрема встановлення MQTT топіків для обміну даними та їх візуалізацію через інструмент Node-RED. Це надає користувачам зручність у моніторингу та аналізі важливих параметрів довкілля в реальному часі.

Таким чином, забезпечує повний огляд та опис розробленої системи, включаючи її апаратну та програмну реалізацію, а також приклад застосування в конкретній ситуації з датчиком BME680. Ця інформація не лише допомагає у розумінні функціональних можливостей системи, а й служить важливим підґрунтям для подальшого розвитку та удосконалення IoT рішень на основі ESP8266 та MQTT.

ВИСНОВКИ

У цій кваліфікаційно бакалаврській роботі було проведено дослідження та розробку системи обміну короткими повідомленнями на базі мікроконтролера ESP8266 та протоколу MQTT. Було досягнуто поставленої мети – створено функціональну та ефективну систему, яка дозволяє обмінюватися повідомленнями між різними пристроями в локальній мережі.

У ході роботи було виконано такі завдання:

Проведено аналіз існуючих рішень: було досліджено різні прошивки для ESP8266, такі як Tasmota, ESPEasy та NodeMCU, а також розглянуто інші платформи, такі як Arduino та Raspberry Pi. На основі аналізу було обрано прошивку NodeMCU як найбільш підходящу для реалізації системи завдяки її гнучкості та можливостям програмування на Lua.

Спроєктовано архітектуру системи: було розроблено клієнт-серверну архітектуру з використанням MQTT, де ESP8266 виступає в ролі клієнта, а MQTT брокер – в ролі сервера. Така архітектура забезпечує надійну доставку повідомлень, масштабованість та гнучкість системи.

Розроблено програмне забезпечення: було написано програмний код для ESP8266 на мові Lua, який реалізує підключення до Wi-Fi мережі, підключення до MQTT брокера, обробку вхідних та вихідних повідомлень, а також зберігання історії повідомлень.

Розроблено апаратне забезпечення: було обрано модель ESP8266-12E (NodeMCU) та розглянуто можливість підключення додаткових компонентів, таких як дисплеї та кнопки, для розширення функціональності системи.

Проведено експериментальні дослідження: Було проведено тестування системи в різних умовах, включаючи різні рівні навантаження та якості з'єднання. Результати тестування підтвердили працездатність та ефективність розробленої системи.

Таким чином, у рамках даної роботи було успішно розроблено систему обміну короткими повідомленнями на базі ESP8266 та MQTT. Система

відповідає поставленим вимогам щодо функціональності, надійності, безпеки та масштабованості. Отримані результати можуть бути використані для подальшого розвитку та вдосконалення системи, а також для створення інших IoT-рішень на базі ESP8266 та MQTT.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kashyap M., Sharma V., Gupta N. Taking MQTT and NodeMcu to IOT: Communication in Internet of Things. Procedia *Computer Science*. 2018. Vol. 132. P. 1611–1618. URL: https://doi.org/10.1016/j.procs.2018.05.126 (date of access: 15.06.2024).

2. DoS/DDoS-MQTT-IoT: A Dataset for Evaluating Intrusions in IoT Networks Using the MQTT Protocol / A. Alatram et al. *Computer Networks*. 2023. P. 109809. URL: https://doi.org/10.1016/j.comnet.2023.109809 (date of access: 15.06.2024).

3. Development of WiFi Mesh Infrastructure for Internet of Things Applications / R. Muhendra та ін. Procedia Engineering. 2017. Т. 170. С. 332–337. URL: https://doi.org/10.1016/j.proeng.2017.03.045 (дата звернення: 12.05.2024).

4. Marino C. A., Chinelato F., Marufuzzaman M. AWS IoT analytics platform for microgrid operation management. Computers & Industrial Engineering. 2022. Т. 170. С. 108331. URL: https://doi.org/10.1016/j.cie.2022.108331 (дата звернення: 12.05.2024).

5. Praseed A., Thilagam P. S. HTTP request pattern based signatures for early application layer DDoS detection: A firewall agnostic approach. Journal of Information Security and Applications. 2022. Т. 65. С. 103090. URL: https://doi.org/10.1016/j.jisa.2021.103090 (дата звернення: 12.05.2024).

6. Klein D. Relying on firewalls? Here's why you'll be hacked. Network Security. 2021. Т. 2021, № 1. С. 9–12. URL: https://doi.org/10.1016/s1353-4858(21)00007-6 (дата звернення: 12.05.2024).

7. A systematic literature review: Messaging protocols and electronic platforms used in the internet of things for the purpose of building smart homes / A. Yudidharma та ін. Procedia Computer Science. 2023. Т. 216. С. 194–203. URL: https://doi.org/10.1016/j.procs.2022.12.127 (дата звернення: 12.05.2024).

8. Real-time reconfigurable web application firewall for a distributed platform : patent US9660960B2. United States. Applied on 16.08.2018 ; published

on 08.12.2018. 18 p. URL: https://patents.google.com/patent/US9660960B2/en?q=(Web+Application+Firewall)&oq=Web+Application+Firewall+ (дата звернення: 12.05.2024).

9. Methods and apparatus for packetized content delivery over a content delivery network : patent US11368498B2 United States. Applied on 15.04.2019 ; published on 10.10.2019. 103 p. URL: https://patents.google.com/patent/US11368498B2/en?q=(Content+Delivery+Network)&oq=Content+Delivery+Network (дата звернення: 12.05.2024).

10. Performing a specific action on a network packet identified as a message queuing telemetry transport (mqtt) packet : patent EP3367627B1 eu. Applied on 26.10.2017 ; published on 01.07.2020. 23 p. URL: https://patents.google.com/patent/EP3367627B1/en?q=(MQTT)&oq=MQTT (дата звернення: 12.05.2024).

11. Synthesis S. Review of current experiences & prospects for teleworking. Bristol : Systems Synthesis, 1990.

12. Cooper R., Ali S., Bi C. Extracting Information from Short Messages. *Natural Language Processing and Information Systems*. Berlin, Heidelberg, 2005. P. 388–391. URL: https://doi.org/10.1007/11428817_44 (date of access: 13.06.2024).

13. Authorship verification for short messages using stylometry / M. L. Brocardo et al. 2013 *International Conference on Computer, Information and Telecommunication Systems (CITS)*, Athens, Greece, 7–8 May 2013. 2013. URL: https://doi.org/10.1109/cits.2013.6705711 (date of access: 13.06.2024).

14. Boloña M. d. C., Allen C. Learners' perceptions of using Moodle Books in online ESP courses. *Intelligent CALL, granular systems and learner data: short papers from EUROCALL 2022*. 2022. P. 30–35. URL: https://doi.org/10.14705/rpnet.2022.61.1430 (date of access: 13.06.2024).

15. Zhao Z., Zhang T., Xie J. Distributed radio resource allocation for aeronautical short messages in LDACS1. *2014 IEEE/AIAA 33rd Digital Avionics*

Systems Conference (DASC), Colorado Springs, CO, USA, 5–9 October 2014. 2014. URL: https://doi.org/10.1109/dasc.2014.6979448 (date of access: 13.06.2024).

16. Classifying Short Messages on Social Networks using Vector Space Models. *9th International Conference on Web Information Systems and Technologies,* Aachen, Germany, 8–10 May 2013. 2013. URL: https://doi.org/10.5220/0004357304130422 (date of access: 13.06.2024).

17. Automatic Drying Roof with Short Messages (SMS) using a Microcontroller Module / M. A. Fhaizal et al. 2023 *17th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, Lombok, Indonesia, 12–13 October 2023. 2023. URL: https://doi.org/10.1109/tssa59948.2023.10366884 (date of access: 13.06.2024).

18. SSID bibliography of social science information & documentation / ed. by R. Kyllikki et al. Helsinki : Helsinki School of Economics, 1994. 88 p.

19. Gay W. UART. *Advanced Raspberry Pi*. Berkeley, CA, 2018. P. 167–187. URL: https://doi.org/10.1007/978-1-4842-3948-3_10 (date of access: 13.06.2024).

20. Spohn M. A., Genero W. B. Análise experimental dos protocolos MQTT e MQTT-SN. *Revista Brasileira de Computação Aplicada*. 2023. Vol. 15, no. 1. P. 22–33. URL: https://doi.org/10.5335/rbca.v15i1.13510 (date of access: 13.06.2024).

21. IoT Editorial Office I. Acknowledgment to the Reviewers of IoT in 2022. *IoT*. 2023. Vol. 4, no. 1. P. 56. URL: https://doi.org/10.3390/iot4010003 (date of access: 13.06.2024).

22. Turner S. Transport Layer Security. *IEEE Internet Computing*. 2014. Vol. 18, no. 6. P. 60–63. URL: https://doi.org/10.1109/mic.2014.126 (date of access: 13.06.2024).

23. Weaver A. C. Secure Sockets Layer. *Computer*. 2006. Vol. 39, no. 4. P. 88–90. URL: https://doi.org/10.1109/mc.2006.138 (date of access: 13.06.2024).

ДОДАТОК А

Довідка

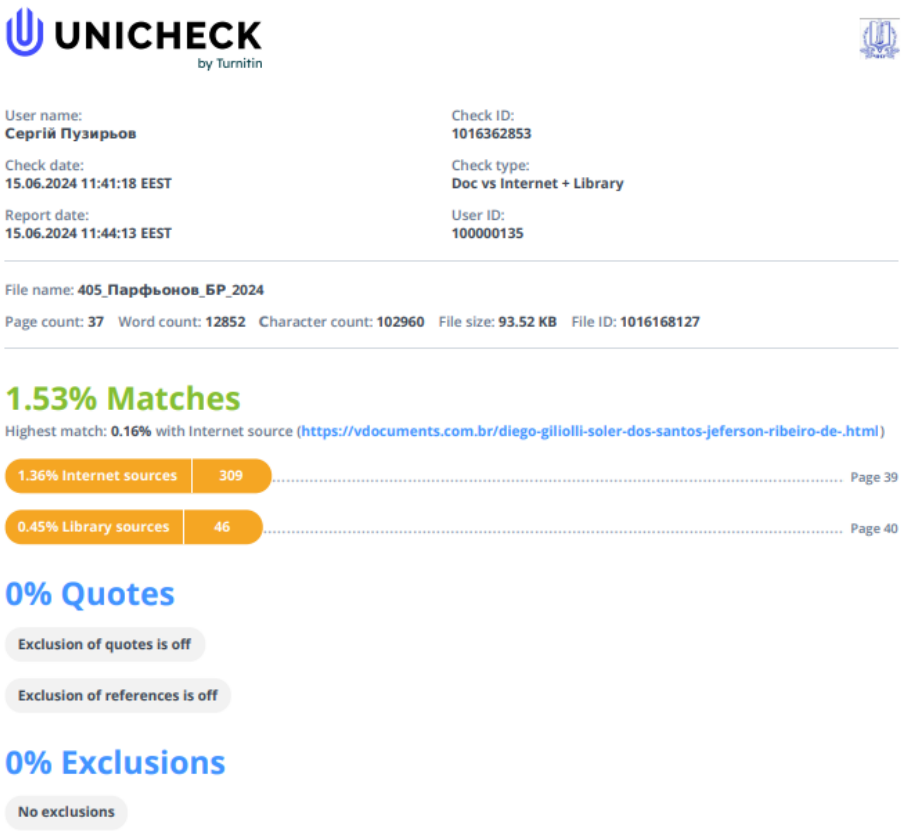
про перевірку на унікальність пояснювальної записки

бакалаврської кваліфікаційної роботи на тему:
«Система обміну короткими повідомленнями на базі ESP8266 та MQTT»

студента спеціальності 123 «Комп'ютерна інженерія», 405 групи
Парфьонов Ілля Миколайович
прізвище, ім'я, по-батькові

Перевірку тексту здійснено сервісом: онлайн-сервіс Unicheck

Результат перевірки тексту бакалаврської кваліфікаційної роботи: схожість складає 1,53%.


UNICHECK
by Turnitin

User name: Сергій Пузирьов
Check ID: 1016362853
Check date: 15.06.2024 11:41:18 EEST
Check type: Doc vs Internet + Library
Report date: 15.06.2024 11:44:13 EEST
User ID: 100000135

File name: 405_Парфьонов_БР_2024
Page count: 37 Word count: 12852 Character count: 102960 File size: 93.52 KB File ID: 1016168127

1.53% Matches
Highest match: 0.16% with Internet source (https://vdocuments.com.br/diego-giliolli-soler-dos-santos-jeferson-ribeiro-de-.html)
1.36% Internet sources 309 Page 39
0.45% Library sources 46 Page 40

0% Quotes
Exclusion of quotes is off
Exclusion of references is off

0% Exclusions
No exclusions

Здобувач:

_____ І. М. Парфьонов
підпис ініціали, прізвище

Керівник:

доц., канд. фіз-мат. наук

_____С. В. Пузирьов
підпис ініціали, прізвище

Дата: «___» _____ 2024 р.

ДОДАТОК Б

Код для підключення ESP8266 до MQTT-сервера і роботи з датчиком Adafruit BME680

```
#include <ESP8266WiFi.h>
#include <Ticker.h>
#include <AsyncMqttClient.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include «Adafruit_BME680.h»

#define WIFI_SSID «ssd»
#define WIFI_PASSWORD «password»

#define MQTT_HOST IPAddress(192, 168, 1, 106)

#define MQTT_HOST «google.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»

float temperature;
float humidity;
float pressure;
float gasResistance;

AsyncMqttClient mqttClient;
Ticker mqttReconnectTimer;

WiFiEventHandler wifiConnectHandler;
WiFiEventHandler wifiDisconnectHandler;
Ticker wifiReconnectTimer;
```

ДОДАТОК В

Wi-Fi MQTT клієнт на ESP8266

```
wifi.setmode(wifi.STATION)
wifi.sta.config(«SSID», «password»)
wifi.sta.connect()

m = mqtt.Client(«client_id», 120, «username», «password»)
m:connect(«broker_address», broker_port, 0, function(client)
  print(«Connected to MQTT broker»)
end)

m:on(«message», function(client, topic, data)
    print(topic .. «: « .. data)
    -- Обробка отриманого повідомлення
  end)

m:publish(topic, message, 0, 0, function(client)
  print(«Message sent»)
end)



wifi.setmode(wifi.STATION) -- Встановлення режиму станції (клієнта)
wifi.sta.config(«SSID», «home11», «password», «ieh5wia5») -- Підключення до Wi-Fi
wifi.sta.connect() -- Підключення до мережі

tmr.alarm(1, 1000, 1, function() -- Перевірка статусу підключення кожну секунду
  if wifi.sta.getip() == nil then
    print(«Connecting to AP...»)
  else
    tmr.stop(1) -- Зупинка таймера після успішного підключення
    print(«IP address: «, wifi.sta.getip())
    -- Далі йде код для підключення до MQTT та інших дій
  end
end)



mqtt = require(«mqtt») -- Підключення бібліотеки MQTT

-- Налаштування MQTT клієнта
```

```
client = mqtt.Client(«ESP8266_Client», 120) -- client_id, keepalive
broker_ip = «YOUR_MQTT_BROKER_IP»
broker_port = 1883

-- Функція зворотного виклику при підключенні до MQTT брокера
client:on(«connect», function(client)
  print(«Connected to MQTT broker»)
  -- Підписка на топіки
  client:subscribe(«topic/in»,0, function(client) print(«Subscribe success») end)
end)

-- Функція зворотного виклику при отриманні повідомлення
client:on(«message», function(client, topic, data)
  print(topic .. «: « .. data)
  -- Обробка отриманого повідомлення
end)

-- Підключення до MQTT брокера
client:connect(broker_ip, broker_port, 0, 1, function(client) print(«Connection
failed!») end)


function sendMessage(topic, message)
  client:publish(topic, message, 0, 0, function(client)
    print(«Sent: « .. message .. « to topic: « .. topic)
  end)
end

-- Приклад відправки повідомлення
sendMessage(«topic/out», «Hello from ESP8266!»)


file.open(«message_history.txt», «a+») -- Відкриття файлу в режимі додавання
file.writeline(message) -- Запис повідомлення у файл
file.close() -- Закриття файлу
```