Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: " Безпека в мережах IoT з використанням брокера MQTT з шифруванням "

Виконав: студент (ка)

Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)

Дмитрів Тарас Олегович
підпис (прізвище та ініціали)

Керівник Лечаченко Т. А.
підпис (прізвище та ініціали)

Нормоконтроль Тимощук Д. І.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Дмитріву Тарасу Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Безпека в мережах IoT з використанням брокера MQTT з шифруванням

Керівник роботи Лечаченко Тарас Анатолійович, доктор філософії, старший викладач кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 15 » 04 2024 року № 4/7-350

2. Термін подання студентом завершеної роботи 12.06.2024

3. Вихідні дані до роботи Вимоги до безпеки IoT пристроїв. Документація по операційній системі Linux. Документація по MQTT Mosquitto

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1. Аналіз предметної області
2. Елементи лабораторного тестового середовища
3. Налаштування та тестування безпечного з'єднання з MQTT
4. Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Тема. Актуальність дослідження. Мета, об'єкт, Предмет дослідження. Завдання дослідження.
Архітектура системи. Основні характеристики та можливості мікроконтролер ESP8266.
Вибір мови програмування та платформи. Налаштування MQTT broker Mosquitto.
Опис програмного коду для ESP8266. Перевірка вразливості мережевого зв'язку.
Налаштування TLS шифрування для MQTT. Інтеграція TLS в програмний код ESP8266.
Аналіз безпеки мережевого трафіку. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Мариненко С. Ю., к.т.н. доцент кафедри МТ		

7. Дата видачі завдання 29.01.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	29.01.2024	
2.	Опрацювання джерел в галузі дослідження	02.02 – 30.01	
3.	Оформлення розділу «Огляд предметної області»	21.02 – 10.03	
4.	Оформлення розділу «Елементи лабораторного тестового середовища»	11.03 – 25.03	
5.	Оформлення розділу «Налаштування та тестування безпечного з’єднання з MQTT»	10.04 – 05.05	
6.	Виконання завдання до підрозділу «Безпека життєдіяльності, основи охорони праці»	10.05 – 21.05	
7.	Оформлення кваліфікаційної роботи	23.05 – 06.06	
8.	Нормоконтроль	06.06 – 10.06	
9.	Перевірка на плагіат	11.06 – 12.06	
10.	Попередній захист кваліфікаційної роботи	14.06 – 15.06	
11.	Захист кваліфікаційної роботи	26.06.2024	

Студент _____ Дмитрів Т.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Лечаченко Т. А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Безпека в мережах IoT з використанням брокера MQTT з шифруванням. // Кваліфікаційна робота ОР «Бакалавр» // Дмитрів Тарас Олегович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБс-42 // Тернопіль, 2024 // С. 68 , рис. – 13, табл. – 0 , кресл. – 17, додат. – 5.

Ключові слова: IoT, безпека даних, TLS, ESP8266, Mosquitto, шифрування.

Кваліфікаційна робота присвячена розробці та валідації системи безпечної взаємодії компонентів IoT. У першому розділі розглянуто критичні аспекти безпеки IoT пристроїв, включаючи аналіз протоколів шифрування TLS, SSL та DTLS. Показано численні вразливості IoT пристроїв та механізми підсилення за допомогою впровадження багатофакторної автентифікації, складних паролів, регулярного оновлення програмного забезпечення та шифрування даних.

У другому розділі досліджено засоби забезпечення надійної взаємодії компонентів IoT системи з використанням мікроконтролерів ESP8266 для збору та передачі даних через MQTT-сервер. Охарактеризовано апаратні компоненти системи та описано програмну частину та характеристику брокера MQTT Mosquitto. Показано, що коректно налаштований MQTT брокер та оптимізований програмний код забезпечують безперебійну роботу системи.

У третьому розділі зосереджено увагу на тестуванні безпеки протоколу MQTT з використанням TLS шифрування. Проведено аналіз мережевого трафіку, виявлено потенційні вразливості та налаштовано TLS шифрування для захисту з'єднань та автентифікації клієнтів і сервера. Внесено зміни в програмний код ESP8266 для використання TLS, що забезпечило захищену передачу даних між клієнтом і сервером. Тестування підтвердило ефективність налаштувань безпеки.

ANNOTATION

Security in IoT networks using MQTT broker with encryption. // Thesis of educational level "Bachelor"// Taras Dmytriv // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, group СБс-42 // Ternopil, 2024 // P. 68, fig. - 13, tab. - 0, drawing - 17, added. - 5.

Keywords: IoT, data security, TLS, ESP8266, Mosquito, encryption.

The qualification work is devoted to the development and validation of a system for secure interaction of IoT components. The first chapter discusses critical aspects of IoT device security, including an analysis of TLS, SSL, and DTLS encryption protocols. Numerous vulnerabilities of IoT devices and mechanisms for strengthening them through the introduction of multi-factor authentication, complex passwords, regular software updates, and data encryption are shown.

The second section investigates the means of ensuring reliable interaction of IoT system components using ESP8266 microcontrollers for collecting and transmitting data via the MQTT server. The hardware components of the system are characterized and the software part and characteristics of the MQTT Mosquito broker are described. It is shown that a correctly configured MQTT broker and optimized program code ensure the smooth operation of the system.

The third section focuses on testing the security of the MQTT protocol using TLS encryption. The network traffic was analyzed, potential vulnerabilities were identified, and TLS encryption was configured to protect connections and authenticate clients and servers. Changes were made to the ESP8266 program code to use TLS, which ensured secure data transfer between the client and the server. Testing confirmed the effectiveness of the security settings.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

MQTT	—	Message Queuing Telemetry Transport
RSA	—	Rivest, Shamir, Adleman
TLS	—	Transport Layer Security
IoT	—	Internet Of Things
SHA	—	Secure Hash Algorithm
TCP	—	Transmission Control Protocol
SSL	—	Secure Sockets Layer
SHA-256	—	Secure Hash Algorithm 256-bit
ЦСК	—	Центр Сертифікації Ключів

ВСТУП

Актуальність теми. Безпека та ефективність роботи пристроїв Інтернету речей стає все більш актуальною, оскільки ці пристрої проникають у всі сфери нашого життя, від домашніх побутових приладів до складних систем промислового контролю. Важливість забезпечення належного рівня безпеки для захисту даних та збереження конфіденційності інформації є пріоритетом у розробці та впровадженні IoT технологій. Аналіз сучасних протоколів шифрування та методів забезпечення безпеки допомагає створювати більш надійні та захищені системи.

Мета і задачі дослідження. Метою даного дослідження є розробка та реалізація техніки забезпечення безпеки та ефективної взаємодії компонентів IoT системи.

Для досягнення цієї мети було визначено наступні задачі:

– Аналіз критичних аспектів безпеки пристроїв IoT та порівняння ключових протоколів шифрування.
– Проектування та реалізація схеми збору та передачі даних на базі мікроконтролерів ESP8266.
– Налаштування та оптимізація брокера MQTT для забезпечення надійної передачі даних.
– Впровадження TLS шифрування для протоколу MQTT.
– Тестування системи з використання шифрування даних.

Об’єкт дослідження. Об'єктом дослідження є процес забезпечення безпеки та ефективної взаємодії компонентів IoT системи.

Предмет дослідження. Предметом дослідження є методи та засоби забезпечення безпеки даних у системах IoT, зокрема використання мікроконтролерів ESP8266, протоколу MQTT та технологій шифрування TLS.

Практичне значення одержаних результатів. Результати дослідження мають практичне значення для розробки безпечних та надійних IoT систем, що дозволяє забезпечити ефективну передачу даних та захист від несанкціонованого доступу. Впроваджена шифрування даних може бути використані в різних галузях,

включаючи промисловість, домашню автоматизацію та медицину, сприяючи підвищенню рівня безпеки та надійності IoT пристроїв.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд безпеки пристроїв IoT

IoT пристрої є невід'ємною частиною сучасних технологій, забезпечуючи різноманітні функції в побутових, комерційних та промислових сферах. Їх інтеграція в повсякденне життя приносить безліч переваг, але також вимагає високого рівня уваги до безпеки. Захист IoT пристроїв має вирішальне значення для забезпечення безперервної і надійної роботи, а також для запобігання потенційним загрозам.

Використання слабких паролів є однією з найбільш поширених проблем у сфері IoT безпеки. Простий пароль може бути легко зламаний, що дозволяє зловмисникам отримати доступ до пристрою. Це, в свою чергу, призводить до серйозних проблем, таких як витік особистих даних, несанкціоноване керування пристроєм або навіть компрометація всієї мережі. Для забезпечення належного рівня безпеки необхідно використовувати складні паролі, що включають комбінації великих і малих літер, цифр і спеціальних символів. Регулярна зміна паролів також є важливим заходом для підтримання безпеки.

MFA є ще одним ефективним методом підвищення безпеки IoT пристроїв. MFA передбачає використання двох або більше незалежних факторів аутентифікації, таких як пароль, фізичний токен, біометричні дані (відбиток пальця, розпізнавання обличчя) або одноразовий код, надісланий на мобільний телефон. Впровадження багатофакторної аутентифікації значно ускладнює завдання зловмисникам, оскільки навіть якщо один з факторів буде скомпрометований, доступ до пристрою залишиться захищеним.

Шифрування даних є ще одним важливим аспектом безпеки IoT пристроїв. Дані, що передаються між пристроями або зберігаються на них, можуть містити конфіденційну інформацію, яка має бути захищена від несанкціонованого доступу. Використання сучасних протоколів шифрування допомагає забезпечити конфіденційність та цілісність даних, зменшуючи ризик їх перехоплення або маніпуляцій.

Протоколи шифрування, такі як TLS та SSL, забезпечують захист даних під час їх передачі через мережу. Вони створюють захищений канал зв'язку між пристроями, шифруючи дані, що передаються.

Шифрування даних на пристроях забезпечує їх захист навіть у випадку, якщо пристрій буде вкрадений або скомпрометований.

Окрім шифрування, важливо використовувати методи цілісності даних для виявлення будь-яких змін або маніпуляцій з даними. Цифрові підписи та хеш-функції є ефективними засобами для забезпечення цілісності даних.

Важливим аспектом є регулярне оновлення програмного забезпечення IoT пристроїв. Автоматичне оновлення програмного забезпечення допомагають підтримувати високий рівень безпеки.

Сегментація мережі є ще одним важливим заходом, який допомагає підвищити безпеку IoT пристроїв. Розміщення IoT пристроїв в окремій підмережі ізольовує їх від основної мережі, де можуть знаходитися більш критичні системи та дані. Це знижує ризик поширення загроз у випадку компрометації одного з IoT пристроїв. Наприклад, якщо один з пристроїв буде зламаний, зловмисники не зможуть отримати доступ до інших систем та даних у мережі, обмежуючи потенційні збитки.

Фізична безпека IoT пристроїв є ще одним важливим аспектом захисту. Забезпечення фізичного захисту пристроїв від крадіжок або несанкціонованого доступу допомагає зменшити ризик фізичного втручання та маніпуляцій. Це особливо важливо для пристроїв, що знаходяться в загальнодоступних або малозахищених місцях. Фізичні заходи безпеки можуть включати використання захисних кейсів, замків, охоронних систем або камер спостереження. Такі заходи допомагають запобігти фізичному доступу до пристроїв, знижуючи ризик крадіжок або несанкціонованого втручання.

Впровадження комплексних заходів безпеки для IoT пристроїв є важливою інвестицією для будь-якої організації або домогосподарства. Такі заходи забезпечують всебічний захист від різноманітних загроз, підвищуючи надійність та ефективність роботи IoT екосистеми. IoT пристрої стають все більш

поширеними і важливими, забезпечення їх безпеки має вирішальне значення для захисту даних, конфіденційності та стабільності технологічних процесів.

Системи управління безпекою мережі є ще одним важливим інструментом, що дозволяє централізовано контролювати та керувати всіма аспектами безпеки IoT пристроїв. Вони дозволяють адміністратору отримувати повну картину стану мережі, швидко виявляти та реагувати на потенційні загрози, а також проводити регулярні аудити безпеки. Використання таких систем допомагає забезпечити комплексний підхід до захисту IoT пристроїв.

Також важливим моментом є навчання та підвищення обізнаності користувачів щодо безпеки IoT пристроїв. Регулярні тренінги та інструктажі допоможуть користувачам зрозуміти важливість безпеки, дізнатися про сучасні загрози та навчитися правильним методам захисту. Це включає в себе навчання щодо створення складних паролів, розпізнавання фішингових атак, правильного налаштування пристроїв та використання засобів захисту.

Забезпечення безпеки IoT пристроїв вимагає комплексного підходу, що включає захист мережі, фізичну безпеку, шифрування даних, регулярне оновлення програмного забезпечення та навчання користувачів. Впровадження цих заходів допомагає забезпечити надійну роботу IoT пристроїв та захистити конфіденційні дані від потенційних загроз.

1.2 Огляд портоколу MQTT

Протокол MQTT є одним з найпопулярніших протоколів для обміну повідомленнями у системах Інтернету речей. Його основні переваги полягають у легкості, ефективності та здатності працювати в умовах обмеженого зв'язку або низької пропускної здатності мережі, що робить його ідеальним для використання в мобільних додатках та пристроях [1].

На рисунку 1.1 показано схему роботи MQTT. Протокол базується на моделі "підписка/публікація" (publish/subscribe), яка дозволяє розподіляти повідомлення через центральний брокер. Клієнти підписуються на теми (topics) і отримують повідомлення, коли інші клієнти публікують дані на ці теми. Ця

модель дозволяє високу масштабованість і знижує залежність між відправником і отримувачем даних.

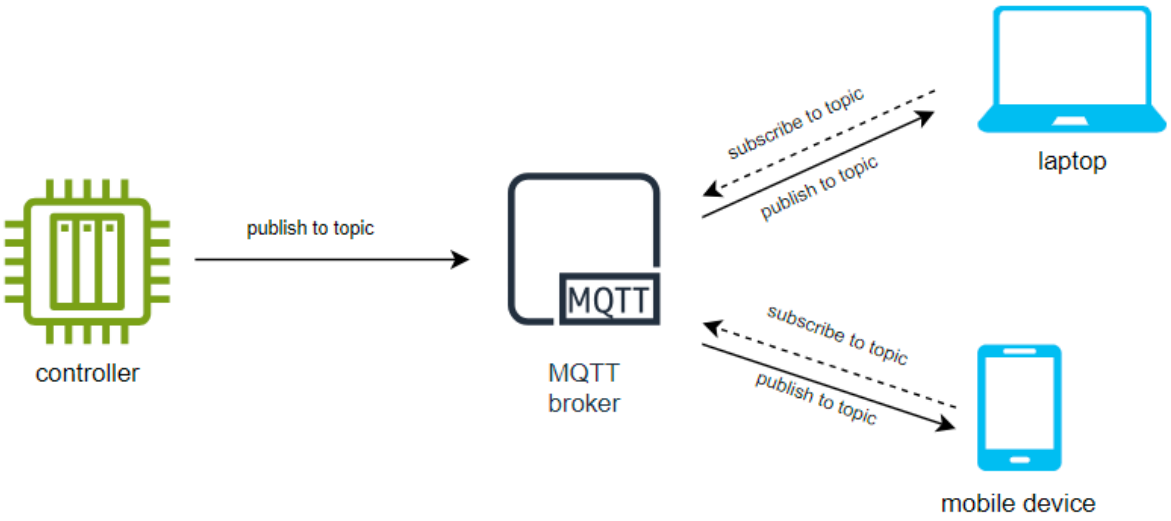

Рисунок 1.1 – Схема роботи протоколу MQTT

Одна з ключових особливостей MQTT є те, що він був спеціально розроблений, щоб мінімізувати обсяг мережевого трафіку, що необхідний для обміну повідомленнями. Це робить його особливо корисним у випадках, де мережеве з'єднання має високу вартість або обмежену пропускну здатність, як-от в мобільних застосунках [2].

MQTT також забезпечує надійність через різні рівні гарантії доставки повідомлень: від 0 (максимально швидка доставка без підтвердження прийому) до 2 (гарантія доставки повідомлення). Це дає можливість вибору між швидкістю доставки та надійністю залежно від потреб додатку.

На рисунку 1.2 показано схему роботи MQTT з використанням TLS. Протокол підтримує захищені з'єднання через TLS/SSL, що забезпечує безпеку даних, переданих через Інтернет. MQTT також пропонує механізми аутентифікації та авторизації, що дозволяють контролювати доступ до мережі та управління темами.

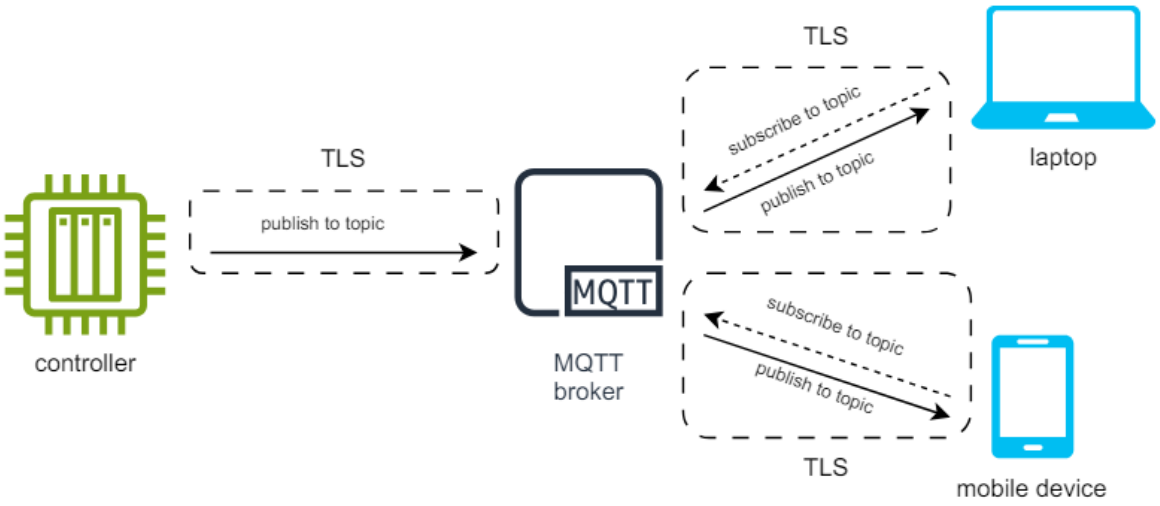

Рисунок 1.2– Схема роботи протоколу MQTT з використанням TLS

Його недоліки можуть включати залежність від центрального брокера, що може створити "bottleneck" та потенційні точки відмови в розподілених системах. Також, у той час як MQTT є ідеальним для простих відправлення повідомлень між пристроями, він може не мати необхідної гнучкості для деяких більш складних або спеціалізованих задач, де потрібно більше контролю над обміном даними.

1.3 Порівняння шифрування

Порівнюючи шифрування TLS, SSL та DTLS, маємо три важливі протоколи безпеки, кожен з яких має свої особливості та переваги, що робить їх придатними для різних застосувань.

SSL був першим широко використовуваним протоколом для забезпечення безпеки передачі даних через Інтернет. Розроблений компанією Netscape у 1995 році, SSL швидко став стандартом для захисту веб-трафіку, електронної пошти та інших форм передачі даних. Основні версії SSL включають SSL 2.0 і SSL 3.0, але обидві були визнані небезпечними через вразливості та атаки, такі як POODLE (Padding Oracle On Downgraded Legacy Encryption) [3].

Однією з ключових особливостей SSL є його використання як для аутентифікації сервера, так і для аутентифікації клієнта, що дозволяє

встановлювати захищені з'єднання між веб-серверами і браузерами. Проте, через численні вразливості та недостатню захищеність від сучасних методів атак, SSL більше не рекомендується для використання [4].

TLS був розроблений як безпечніший спадкоємець SSL. Починаючи з версії TLS 1.0, протокол зазнав значних поліпшень у порівнянні з SSL, включаючи посилення шифрування, підвищення ефективності захисту даних і виправлення вразливостей. Найновішою версією на момент написання є TLS 1.3, яка забезпечує ще більш високий рівень безпеки та продуктивності.

Основні переваги TLS над SSL:

– TLS 1.2 та TLS 1.3 виправили вразливості SSL, зокрема шляхом поліпшення алгоритмів шифрування і додавання нових криптографічних методів.
– TLS 1.3 значно зменшує час встановлення з'єднання (handshake), що підвищує швидкість та ефективність передачі даних.
– TLS підтримує широкий спектр криптографічних алгоритмів і протоколів, що дозволяє адаптуватися до різних потреб безпеки.

TLS забезпечує шифрування даних, аутентифікацію сторін і цілісність переданих даних, що робить його ідеальним для використання в веб-застосунках, електронній пошті, VPN та інших формах передачі даних.

DTLS є адаптацією TLS для використання з протокол передачі даних UDP. Це дозволяє забезпечити захищену передачу даних у середовищах, де важлива низька затримка і швидке встановлення з'єднань, наприклад, у VoIP або стрімінгових сервісах.

Основні особливості DTLS:

– Забезпечує швидке встановлення з'єднань, що робить його ідеальним для застосувань, де важлива мінімальна затримка.
– Пристосовується до середовищ з високими втратами пакетів, що дозволяє зберігати стабільність з'єднань навіть у несприятливих умовах.
– Зберігає більшість переваг TLS, включаючи сильне шифрування та аутентифікацію, що робить його надійним протоколом для захисту даних.

Порівняння SSL, TLS та DTLS:

– SSL, TLS і DTLS забезпечують шифрування даних, аутентифікацію сторін та цілісність даних. Проте кожен з цих протоколів має свої специфічні переваги та недоліки.
– TLS забезпечує найвищий рівень безпеки серед трьох протоколів. SSL вже застарів і має численні вразливості, тоді як TLS включає виправлення та поліпшення, що робить його стійким до сучасних атак. DTLS, будучи адаптацією TLS, також забезпечує високий рівень безпеки для застосувань на базі UDP.
– TLS 1.3 має значні поліпшення в продуктивності порівняно з попередніми версіями та SSL. Завдяки швидшому процесу встановлення з'єднань, TLS 1.3 мінімізує затримки і підвищує ефективність. DTLS зберігає більшість цих переваг, адаптуючись до середовищ з низькою затримкою.
– SSL, через свої обмеження та вразливості, більше не використовується в сучасних системах. TLS є універсальним протоколом, що підходить для більшості застосувань, включаючи веб-трафік, електронну пошту та VPN. DTLS, своєю чергою, ідеально підходить для застосувань, де важлива низька затримка, таких як VoIP та стрімінгові сервіси.

Хоча SSL став першим у сфері захисту онлайн-комунікацій, виникнення серйозних проблем та обмеженостей з часом зробило його використання неприйнятним для сучасних застосувань. Протокол TLS, який був розроблений як вдосконалення SSL, сьогодні визначає стандарти безпеки для захищених з'єднань через Інтернет.

Однією з ключових переваг TLS є його здатність виправляти недоліки, які були виявлені в SSL. Наприклад, TLS 1.3, найновіша версія протоколу, забезпечує вдосконалені заходи безпеки та ефективність, включаючи скорочення часу handshake та виключення застарілих алгоритмів шифрування, що робить його стійкішим до атак. Такі покращення не тільки сприяють забезпеченню конфіденційності даних, але й знижують загальне навантаження на систему, сприяючи більш швидкій і надійній обробці даних [5].

Також, TLS пропонує гнучкість у виборі криптографічних алгоритмів, що дозволяє організаціям та розробникам адаптувати рівень безпеки під специфічні

потреби та загрози. Це дуже важливо в умовах постійно змінюваного кіберпейзажу, де адаптація до нових викликів є ключовим фактором успішної кібербезпеки.

Для застосувань, які базуються на UDP, таких як потокове відео чи VoIP, DTLS пропонує адаптований до цих потреб захист, зберігаючи переваги TLS в умовах, де використовуються протоколи типу UDP. Втім, незважаючи на ефективність DTLS в специфічних сценаріях, TLS залишається більш універсальним рішенням, що підходить для ширшого спектру додатків [6].

1.4 Висновки до першого розділу

У першому розділі було детально розглянуто критичні аспекти безпеки пристроїв IoT та порівнює ключові протоколи шифрування, що забезпечують захист даних у цих системах.

Аналіз безпеки пристроїв IoT показує, що ці пристрої, незважаючи на свою корисність у різних сферах, мають численні вразливості. Використання слабких паролів та відсутність багатофакторної аутентифікації робить їх легкою мішенню для зловмисників. Тому застосування складних паролів, регулярна їх зміна, впровадження багатофакторної аутентифікації та шифрування даних є критично важливими заходами для забезпечення належного рівня безпеки. Регулярне оновлення програмного забезпечення та сегментація мережі також відіграють значну роль у захисті пристроїв IoT. Фізична безпека пристроїв є не менш важливою, особливо у випадках, коли пристрої знаходяться в загальнодоступних місцях.

Протокол MQTT був обраний для розгляду через його популярність та ефективність у середовищах з обмеженим зв'язком. Легкість протоколу та його здатність працювати у різних умовах робить його ідеальним для мобільних застосунків та пристроїв на акумуляторах. Однак, залежність від центрального брокера може стати вузьким місцем у системі. Впровадження шифрування через TLS/SSL у MQTT забезпечує надійний захист даних під час передачі, що є важливою перевагою цього протоколу.

Порівняння протоколів шифрування (TLS, SSL та DTLS) демонструє еволюцію технологій безпеки. SSL, будучи першим широко використовуваним протоколом, виявився вразливим до багатьох атак і вже не використовується у сучасних системах. TLS, як його наступник, включає значні поліпшення у шифруванні та продуктивності, забезпечуючи високий рівень безпеки. Остання версія TLS 1.3 пропонує скорочений час встановлення з'єднання та виключає застарілі алгоритми, що робить його ще більш надійним. DTLS, адаптація TLS для UDP, зберігає переваги TLS у середовищах з низькою затримкою, таких як VoIP та стрімінгові сервіси.

РОЗДІЛ 2 ЕЛЕМЕНТИ ЛАБОРАТОРНОГО ТЕСТОВОГО СЕРЕДОВИЩА

2.1 Загальна схема тестового середовища

На рисунку 2.1 показана схема, яка представляє комплексну систему збору та передачі даних, яка базується на мікроконтролерах ESP8266, підключених до маршрутизатора через Wi-Fi..

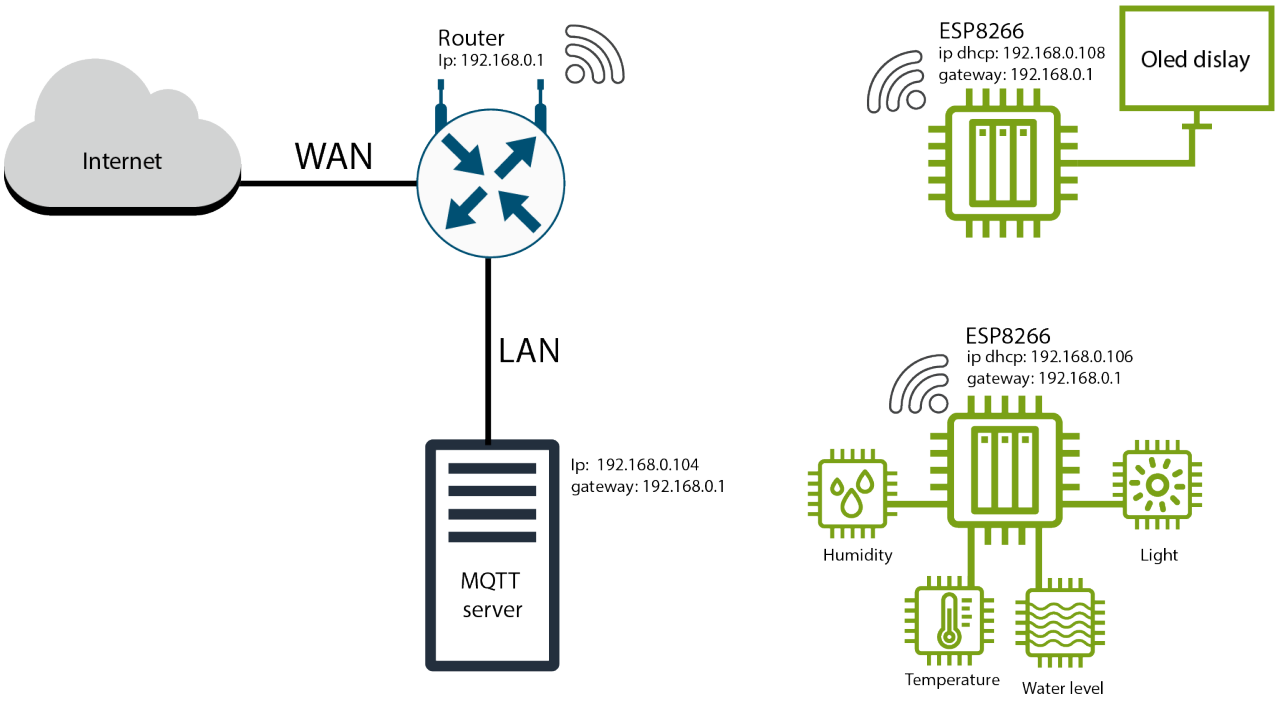

Рисунок 2.1 – Загальна схема тестового середовища

Маршрутизатор має IP-адресу 192.168.0.1 та забезпечує доступ до Інтернет через WAN-з’єднання. В локальній мережі (LAN) знаходиться MQTT-сервер з IP-адресою 192.168.0.104 та шлюзом 192.168.0.1, який слугує центральним вузлом для обміну даними між пристроями.

У системі використовуються два мікроконтролери ESP8266. Перший з них, з IP-адресою, отриманою по DHCP 192.168.0.108 та шлюзом 192.168.0.1, оснащений OLED-дисплеєм для візуалізації даних. Другий мікроконтролер, з IP-адресою 192.168.0.106 та тим же шлюзом, підключений до датчиків вологості, освітленості, температури та рівня води, що дозволяє здійснювати моніторинг різних параметрів навколишнього середовища.

Ця система забезпечує збір, обробку та передачу даних з різноманітних сенсорів до MQTT-сервера, звідки дані можуть бути доступні для подальшого аналізу та використання. Схема демонструє злагоджену взаємодію між усіма компонентами, що демонструє типову схему IoT мережі.

2.2 Опис апаратної частини системи

2.2.1 Мікроконтролер ESP8266

ESP8266 є високо інтегрованим мікроконтролером з підтримкою Wi-Fi, що був розроблений компанією Espressif Systems. Цей чип став популярним завдяки своїм потужним можливостям, низькому енергоспоживанню та доступності за ціною, що зробило його ідеальним вибором для проєктів Інтернету речей.

Основою ESP8266 є 32-бітний процесор Tensilica L106 Diamond, який працює на тактовій частоті до 160 МГц. Це забезпечує достатню продуктивність для виконання складних задач обробки даних. Окрім того, ESP8266 має 64 КБ інструкційного кешу і 96 КБ даних RAM, що дозволяє ефективно працювати з мережею та обробляти інформацію в реальному часі [7].

ESP8266 має вбудований модуль Wi-Fi, який підтримує стандарти 802.11 b/g/n. Він може працювати як в режимі точки доступу (AP), так і в режимі клієнта (STA), а також у комбінованому режимі AP+STA. Це дозволяє ESP8266 підключатися до існуючих Wi-Fi мереж або створювати власні бездротові мережі для зв'язку з іншими пристроями.

Однією з ключових особливостей ESP8266 є його здатність до програмування за допомогою різних мов, включаючи Lua (у середовищі NodeMCU), MicroPython, C та C++ (у середовищі Arduino IDE). Це надає розробникам велику гнучкість у виборі засобів та методів розробки програмного забезпечення.

На рисунку 2.2 показано інтерфейси вводу-виводу ESP8266 включають цифрові GPIO (до 17 пінів, в залежності від конкретної моделі), аналоговий вхід

(ADC), UART, SPI, I2C та PWM. Це робить його універсальним для підключення різноманітних сенсорів, актуаторів та інших периферійних пристроїв.

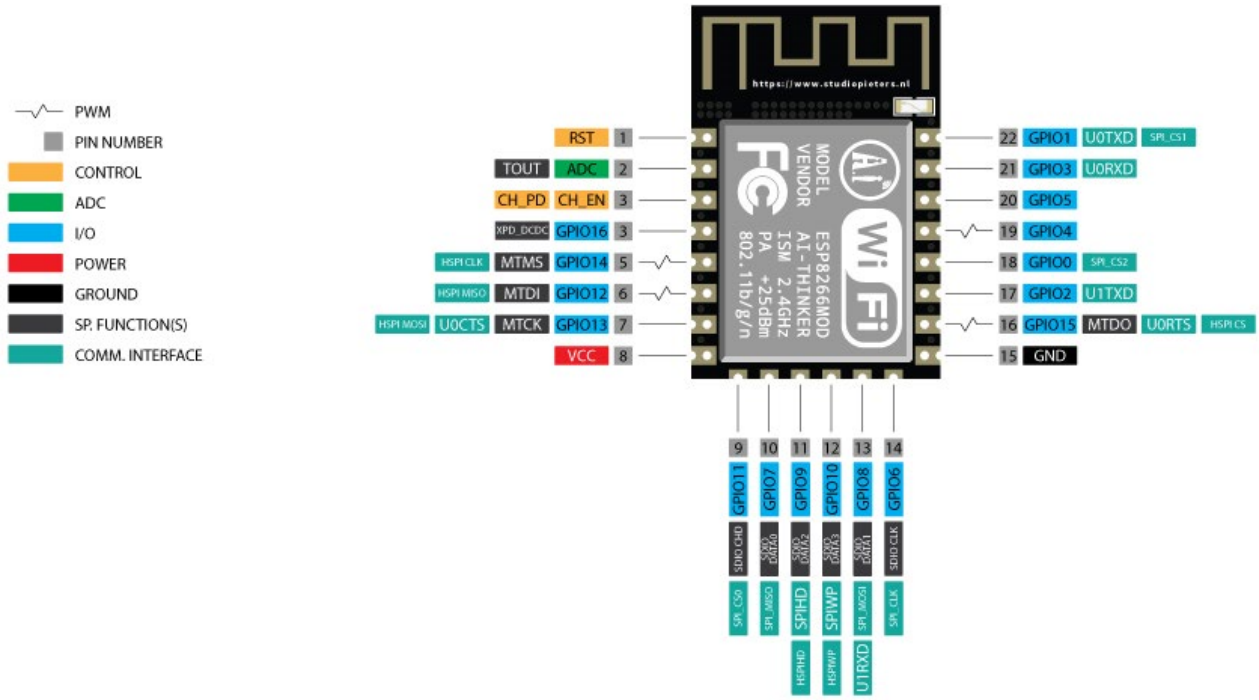

Рисунок 2.2 – Інтерфейси вводу-виводу ESP8266

З точки зору енергоспоживання, ESP8266 може працювати в різних режимах, включаючи режим глибокого сну, який значно зменшує споживання енергії, що є важливим для автономних систем на акумуляторах.

2.2.2 Давач температури та вологості DHT11.

На рисунку 2.3 показано DHT11, який є широко використовуваним давачем, який забезпечує вимірювання температури та вологості повітря. Цей давач відрізняється високою надійністю, довготривалою стабільністю та простотою інтеграції в різноманітні електронні системи, що робить його популярним вибором у проєктах з автоматизації, системах контролю клімату та розумних будинках.

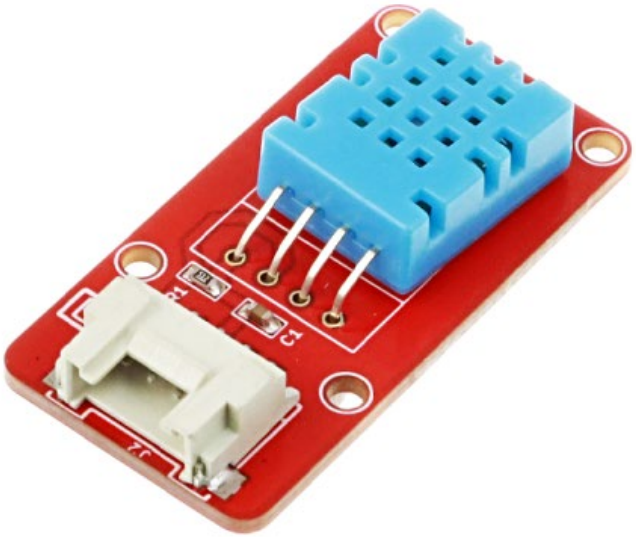

Рисунок 2.3 – Зовнішній вигляд давач DHT11

Давач DHT11 використовує термістор для вимірювання температури і ємнісний давач для визначення вологості. Це поєднання забезпечує точні та надійні показники з відносно швидким часом відгуку. Робочий діапазон температур становить від 0 до 50 °C з точністю ±2 °C, тоді як діапазон вологості – від 20% до 90% з точністю ±5% [8].

DHT11 передає дані за допомогою цифрового інтерфейсу, що значно спрощує його підключення до мікроконтролерів, таких як Arduino, Raspberry Pi та інших платформ розробки. Дані передаються по одному проводу, що зменшує кількість необхідних з'єднань і полегшує монтаж давача в системі.

Однією з ключових особливостей DHT11 є його низьке енергоспоживання, що робить його придатним для використання в автономних пристроях на батарейному живленні. Типове споживання струму в режимі очікування складає близько 50 мкА, а під час вимірювання – близько 2,5 мА.

Корпус DHT11 є компактним і міцним, що забезпечує зручність встановлення та довговічність у різних умовах експлуатації. Його розміри дозволяють легко інтегрувати давач у будь-які системи без значних змін конструкції.

2.2.3 Давач світла CT0013LS.

На рисунку 2.4 показано давач світла моделі CT0013LS, який є високочутливим пристроєм, призначеним для вимірювання рівня освітленості в оточенні. Цей сенсор використовує фотодіод як основний елемент, що дозволяє йому ефективно вловлювати та перетворювати світло на електричний сигнал. Перевага фотодіода полягає в його швидкій реакції на зміни освітленості та широкому спектрі чутливості.

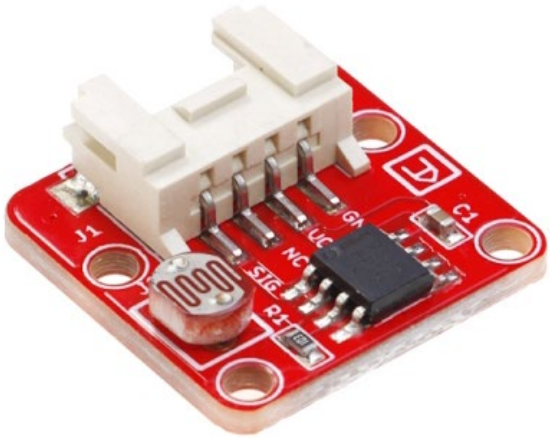

Рисунок 2.4 – Зовнішній вигляд давача CT0013LS

Завдяки низькому енергоспоживанню, цей сенсор є ідеальним для використання у батарейних системах та інших автономних пристроях. Робоча напруга давача світла складає 5 В, що дозволяє йому працювати з більшістю мікроконтролерів і платформ розробки, таких як ESP8266. Споживання струму варіюється від 0,5 до 3 мА, що робить його енергоефективним рішенням [9].

CT0013LS характеризується високою чутливістю, що дає змогу точно вимірювати рівень освітленості в різних умовах, від слабкого до яскравого світла. Спротив при освітленні складає 20 кОм, тоді як в темряві він досягає 1 МОм. Час відгуку сенсора становить 20-30 секунд, що дозволяє йому швидко реагувати на зміни в освітленості. Пікова довжина хвилі чутливості складає 540 нм, що відповідає максимальній чутливості до зеленого світла.

Температурний діапазон роботи сенсора становить від -30 °C до +70 °C, що робить його придатним для використання в різних кліматичних умовах. Компактні розміри пристрою (20 мм у довжину, 20 мм у ширину та 6,8 мм у висоту) дозволяють легко інтегрувати його у будь-які системи, не займаючи багато місця.

Crowtail-Light Sensor моделі CT0013LS забезпечує стабільну та надійну роботу, що є критично важливим для систем, де точність вимірювання освітленості є вирішальною. Завдяки своїй високій чутливості, низькому енергоспоживанню та сумісності з Crowtail інтерфейсами, цей сенсор є ідеальним вибором для широкого спектру застосувань у різних галузях.

2.2.4 Давач рівня води CT0042CWS.

На рисунку 2.5 показано давач рівня води моделі CT0042CWS, він є інноваційним пристроєм для вимірювання рівня води, який функціонує за допомогою серії відкритих провідних доріжок, що підключені до заземлення. Між цими заземленими доріжками розташовані сенсорні доріжки, які мають слабкий підтягуючий резистор на 1 МОм. Цей резистор підтягує значення сенсорних доріжок до високого рівня доти, доки крапля води не замкне сенсорну доріжку із заземленою доріжкою.

.

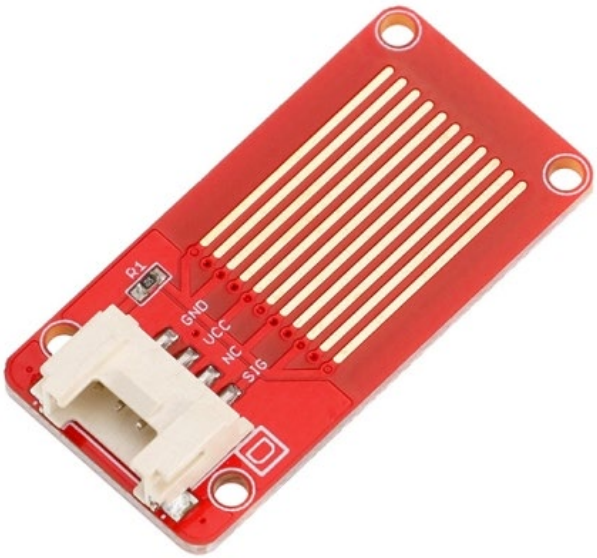

Рисунок 2.5 – Зовнішній вигляд давач CT0042CWS

Цей простий, але ефективний принцип роботи дозволяє давачеві взаємодіяти як з цифровими, так і з аналоговими входами мікроконтролерів, таких як Arduino. Використовуючи аналогові входи, можна визначити ступінь контакту між заземленими та сенсорними доріжками, що дозволяє оцінити кількість води, яка контактує з давачем.

Давач CT0042CWS має декілька важливих технічних характеристик. Він має сумісний з Crowtail інтерфейс, що забезпечує легке підключення до різних модулів. Пристрій відзначається низьким споживанням енергії та високою чутливістю. Він працює при напрузі в діапазоні від 4.75V до 5.25V та в температурному діапазоні від 10°C до 30°C. Розміри давача становлять 40 мм в довжину, 20 мм в ширину та 6.8 мм у висоту [10].

2.2.4 Дисплей SSD1306.

На рисунку 2.6 показано дисплей SSD1306 з діагоналлю 0,96 дюйма і роздільною здатністю 128x64 пікселів, це високоякісний OLED дисплей, який широко використовується в електроніці та різноманітних IoT-проєктах. Цей дисплей відрізняється компактними розмірами і чудовою якістю зображення, що

робить його ідеальним для використання в портативних пристроях, розумних гаджетах та інших інтерактивних системах.

Рисунок 2.6 – Зовнішній вигляд дисплея SSD1306

SSD1306 є монокристалічним OLED-драйвером, який забезпечує високу контрастність та яскравість зображення. Використання органічних світлодіодів (OLED) у цьому дисплеї дозволяє отримати глибокі чорні кольори та високу яскравість без необхідності у фоновому підсвічуванні, що значно знижує енергоспоживання. Технологія OLED також забезпечує широкі кути огляду, що робить зображення чітким та яскравим з будь-якого ракурсу.

Цей дисплей використовує інтерфейс I2C для передачі даних, що дозволяє зменшити кількість з'єднань і спростити процес інтеграції з мікроконтролерами, такими як Arduino, Raspberry Pi та іншими платформами розробки. Інтерфейс I2C підтримує чотири проводи, включаючи живлення (VCC), землю (GND), тактовий сигнал (SCL) та сигнал даних (SDA), що забезпечує надійну та швидку передачу даних [11].

Робоча напруга дисплея становить 3,3 В або 5 В, що забезпечує сумісність з широким спектром мікроконтролерів. Дисплей характеризується низьким енергоспоживанням, що є важливим для автономних пристроїв на батарейному

живленні. Типове споживання струму складає близько 20 мА, що сприяє тривалому часу роботи пристрою від батареї.

Дисплей SSD1306 I2C 4p 0,96" 128x64 має компактні розміри, що дозволяє легко інтегрувати його в різні проекти, не займаючи багато місця. Висока роздільна здатність дисплея (128x64 пікселів) забезпечує чітке та детальне відображення інформації, що є критично важливим для інтерфейсів користувача та відображення графіки.

2.3 Характеристика програмної частини

2.3.1 Вибір мови програмування

Arduino є потужним і широко використовуваним фреймворком для розробки електронних пристроїв і систем. Заснований на відкритій архітектурі, Arduino поєднує апаратні та програмні засоби, що дозволяє інженерам, розробникам і ентузіастам створювати різноманітні проекти, починаючи від простих давачів і закінчуючи складними роботизованими системами.

Програмне забезпечення Arduino представлено середовищем розробки (IDE), яке базується на мові програмування, похідній від C/C++. IDE забезпечує зручний інтерфейс для написання коду, компіляції та завантаження програм на мікроконтролери Arduino. Стандартна бібліотека Arduino включає безліч функцій для роботи з апаратними компонентами, що значно спрощує процес розробки.

Фреймворк Arduino також активно підтримується великою спільнотою розробників, що створюють і діляться численними бібліотеками, прикладами коду та документацією. Це робить Arduino доступним для новачків, надаючи їм необхідні інструменти та ресурси для навчання та розвитку своїх навичок у сфері електроніки та програмування [12].

2.3.2 Характеристика MQTT Mosquitto broker.

Mosquitto є популярним і широко використовуваним брокером протоколу MQTT, розробленим для забезпечення надійної і легкої у використанні платформи для обміну повідомленнями між пристроями в мережі IoT. MQTT є легким протоколом публікації/підписки, який спеціально розроблений для ефективної передачі даних в умовах обмежених ресурсів і нестабільних мережевих з'єднань.

Брокер Mosquitto є ключовим компонентом в архітектурі MQTT, оскільки він відповідає за прийом повідомлень від видавців, зберігання цих повідомлень та їх подальшу передачу підписникам. Mosquitto підтримує версії протоколу MQTT 3.1, 3.1.1 та 5.0, що забезпечує сумісність з широким спектром клієнтів і пристроїв.

Однією з головних переваг Mosquitto є його мала вага і висока продуктивність, що робить його ідеальним для використання на пристроях з обмеженими ресурсами, таких як Raspberry Pi та інші вбудовані системи. Mosquitto також підтримує SSL/TLS шифрування, що забезпечує безпеку передачі даних, захищаючи їх від несанкціонованого доступу та маніпуляцій.

Mosquitto легко інтегрується з різними мовами програмування і платформами, що забезпечується завдяки наявності бібліотек і клієнтських інтерфейсів для C, C++, Python, Java, JavaScript та інших мов. Це дозволяє розробникам швидко і ефективно створювати рішення на основі MQTT для широкого спектру застосувань.

Брокер Mosquitto може бути налаштований для роботи в різних режимах, включаючи автономний режим для невеликих мереж та кластерний режим для масштабованих рішень, що вимагають високої доступності та надійності. Конфігураційні файли Mosquitto дозволяють детально налаштовувати параметри з'єднання, безпеки та продуктивності, що забезпечує гнучкість та адаптивність до різних умов експлуатації [13].

Крім того, Mosquitto підтримує зберігання повідомлень на диск, що дозволяє зберігати дані навіть після перезапуску системи. Це забезпечує високу стійкість до збоїв і втрат даних. Підтримка ретенційних повідомлень та функцій

збереження сеансів підвищує зручність використання та надійність систем на основі Mosquitto.

2.3.3 Опис операційної системи Ubuntu.

Ubuntu є одним із найпопулярніших дистрибутивів операційної системи Linux, розробленим компанією Canonical Ltd. Він базується на Debian і має відкритий вихідний код, що забезпечує високий рівень гнучкості та адаптивності. Ubuntu відзначається своєю стабільністю, безпекою та простотою використання, що робить його підходящим як для новачків, так і для досвідчених користувачів.

Ubuntu підтримує широкий спектр апаратних платформ, включаючи настільні комп'ютери, сервери, вбудовані системи та хмарні інфраструктури. Це дозволяє використовувати Ubuntu у різних середовищах, від персональних комп'ютерів до великих дата-центрів. Крім того, доступні спеціалізовані версії, такі як Ubuntu Server для серверних рішень та Ubuntu Core для вбудованих систем і пристроїв IoT.

Однією з ключових особливостей Ubuntu є регулярні випуски оновлень і нових версій. Кожні шість місяців виходить нова версія, а кожні два роки випускається версія з довготривалою підтримкою (LTS), яка отримує оновлення безпеки та підтримку протягом п'яти років. Це забезпечує користувачам доступ до новітніх технологій та покращень, водночас гарантуючи стабільність і безпеку системи.

Система пакетного менеджменту в Ubuntu базується на APT (Advanced Package Tool), що спрощує встановлення, оновлення та видалення програмного забезпечення. Репозиторії Ubuntu містять тисячі програмних пакетів, що охоплюють різноманітні сфери, включаючи офісні програми, мультимедіа, інструменти розробки та системні утиліти. Крім того, підтримуються сторонні репозиторії та пакети у форматі Snap, що дозволяє швидко і безпечно встановлювати програмне забезпечення з різних джерел [14].

Безпека в Ubuntu забезпечується через регулярні оновлення системи, використання апаратної ізоляції, а також інструменти для шифрування даних і управління правами доступу. Canonical активно працює над виявленням і виправленням вразливостей, що забезпечує високий рівень захисту даних і приватності користувачів.

2.4 Реалізація системи

2.4.1 Налаштування MQTT broker.

Налаштуємо MQTT broker, для роботи системи. На рисунку 2.7 показано частину конфігураційного файлу MQTT-брокера Mosquitto.

```
/etc/mosquitto/conf.d/default.conf
allow_anonymous false
password_file /etc/mosquitto/passwd
listener 1883 0.0.0.0
```

Рисунок 2.7 – Вміст конфігураційного файлу MQTT

Цей конфігураційний файл містить наступні налаштування:

1) `allow_anonymous false` – ця опція забороняє анонімний доступ до MQTT-брокера. Це означає, що для підключення до брокера клієнтам необхідно буде вказати дійсні ім'я користувача та пароль;
2) `password_file /etc/mosquitto/passwd` – вказує на файл, де зберігаються імена користувачів та хешовані паролі для аутентифікації клієнтів. У цьому випадку файл розташований за шляхом /etc/mosquitto/passwd;
3) `listener 1883 0.0.0.0` – ця директива налаштовує брокер на прослуховування з'єднань на порту 1883 на всіх інтерфейсах мережі. Порт 1883 є стандартним портом для протоколу MQTT;

На рисунку 2.8 показано частину вмісту файлу паролів для MQTT-брокера Mosquitto. Файл знаходиться за шляхом /etc/mosquitto/passwd і містить записи

для аутентифікації користувачів. Ім'я користувача “rp5” та Зашифрований пароль.

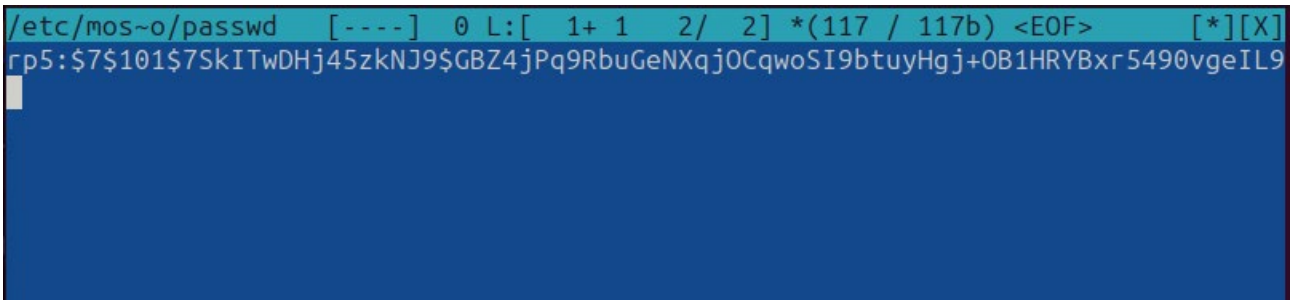

Рисунок 2.8 – Вміст файлу паролів для MQTT-брокера Mosquitto

Зашифрований пароль представлений у форматі bcrypt, що є стійким до атак методом підбору паролів. Використання bcrypt забезпечує додаткову безпеку, оскільки цей метод використовує сіль для ускладнення процесу взлому

2.4.2 Програмний код для надсилання даних з ESP8266.

Код надсилає кліматичні дані на MQTT сервер. Він реалізований на базі ESP8266 для збору та передачі даних про температуру, вологість, рівень води та рівень освітлення через протокол MQTT. Програма використовує сенсор DHT11 для вимірювання температури та вологості, аналоговий давач для вимірювання рівня води та цифровий давач для рівня освітлення.

У даному фрагменті коду реалізовано комплексну систему для збору та передачі даних про навколишнє середовище з використанням мікроконтролера ESP8266 (див. лістинг 2.1).

Лістинг 2.1 – Імпорт бібліотек та оголошення змінних

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
#define DHTPIN D1
#define DHTTYPE DHT11
#define WATER_SENSOR_PIN A0
```

```
#define LIGHT_SENSOR_PIN D3
const char* ssid = "";
const char* password = "";
const char* mqtt_server = "192.168.0.104";
const char* mqtt_user = "rp5";
const char* mqtt_password = "1234";
WiFiClient espClient;
PubSubClient client(espClient);
DHT dht(DHTPIN, DHTTYPE);
```

Використовуючи бібліотеки для Wi-Fi, MQTT та сенсора DHT11, код забезпечує безперебійний збір і відправлення даних на сервер через протокол MQTT. У початковій частині коду імпортуються бібліотеки ESP8266WiFi.h, PubSubClient.h та DHT.h, які забезпечують функціональність підключення до Wi-Fi, обміну повідомленнями через MQTT та роботу з сенсором DHT11 відповідно. Для роботи сенсорів визначаються пін-коди: пін D1 для сенсора DHT11, пін A0 для аналогового водного сенсора та пін D3 для цифрового сенсора освітлення. Константи для підключення до Wi-Fi та MQTT-сервера включають назву мережі (SSID), пароль до мережі, адресу MQTT-сервера та облікові дані користувача MQTT.

У функції setup(), яка виконується під час запуску програми, відбувається ініціалізація послідовного зв’язку для виведення даних на монітор послідовного порту (див. лістинг 2.2).

Лістинг 2.2 – Налаштування Wi-Fi та MQTT

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP8266Client", mqtt_user,
mqtt_password)) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
```

```
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

Викликається функція setup_wifi(), яка відповідає за встановлення з'єднання з Wi-Fi мережею. У функції setup_wifi(), після деякої затримки, виводиться повідомлення про початок підключення до мережі, і виконується команда WiFi.begin(ssid, password), яка ініціює процес підключення. Програма очікує, доки з'єднання не буде встановлено, після чого виводиться повідомлення про успішне підключення та IP-адресу, присвоєну мікроконтролеру. Далі, у функції setup(), встановлюється сервер MQTT за допомогою команди client.setServer(mqtt_server, 1883), ініціалізується сенсор DHT11 командою dht.begin(), а пін для цифрового сенсора освітлення налаштовується як вхідний за допомогою команди pinMode(LIGHT_SENSOR_PIN, INPUT).

У функції reconnect() відбувається відновлення з'єднання з MQTT-сервером у разі його втрати (див. лістинг 2.3).

Лістинг 2.2 – Налаштування Wi-Fi та MQTT

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP8266Client", mqtt_user, mqtt_password)) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
```

```
    }
  }
}
```

Під час виконання цієї функції виводиться повідомлення про спробу встановити з’єднання, після чого виконується команда client.connect("ESP8266Client", mqtt_user, mqtt_password), яка намагається підключитися до сервера з використанням облікових даних користувача. Якщо з’єднання успішне, виводиться повідомлення "connected". У випадку невдачі, програма виводить код помилки та чекає 5 секунд перед повторною спробою встановлення з’єднання.

У основному циклі програми, що реалізований у функції loop(), відбувається безперервний моніторинг стану підключення до MQTT-сервера та збирання даних з сенсорів (див. лістинг 2.4).

Лістинг 2.4 – Основний цикл програми

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP8266Client", mqtt_user, mqtt_password)) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

Спочатку перевіряється наявність підключення до сервера. Якщо з'єднання відсутнє, викликається функція reconnect(). Потім зчитуються значення вологості та температури з сенсора DHT11 за допомогою команд dht.readHumidity() та dht.readTemperature(). Значення рівня води зчитується з аналогового сенсора командою analogRead(WATER_SENSOR_PIN), а рівень освітлення визначається за допомогою команди digitalRead(LIGHT_SENSOR_PIN). Якщо сенсори DHT11 не повертають коректних значень, виводиться повідомлення про помилку зчитування, і виконання функції переривається. Отримані значення конвертуються у рядки за допомогою функції dtostrf(), після чого публікуються на MQTT-сервері за допомогою команд client.publish(). Основний цикл повторюється кожні 2 секунди, забезпечуючи регулярне оновлення даних на сервері.

2.4.3 Програмний код для отримання даних на ESP8266.

Даний код реалізує програму на мікроконтролері ESP8266 для отримання даних через протокол MQTT і відображення їх на дисплеї SSD1306. Програма складається з кількох основних компонентів, які забезпечують підключення до Wi-Fi, налаштування MQTT-клієнта, обробку отриманих повідомлень та оновлення дисплея.

Цей фрагмент коду імпортує необхідні бібліотеки для роботи з Wi-Fi, MQTT та дисплеєм SSD1306 (див. лістинг 2.5).

Лістинг 2.5 – Імпорт бібліотек та оголошення змінних

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Adafruit_ssd1306syp.h>
#define SDA_PIN 14
#define SCL_PIN 12
const char* ssid = "";
```

```
const char* password = "";
const char* mqtt_server = "192.168.0.104";
const char* mqtt_user = "rp5";
const char* mqtt_password = "1234";
Adafruit_ssd1306syp display(SDA_PIN, SCL_PIN);
WiFiClient espClient;
PubSubClient client(espClient);
float temperature = 0;
float humidity = 0;
int waterLevel = 0;
int lightLevel = 0;
```

Бібліотека ESP8266WiFi.h використовується для підключення до Wi-Fi мережі, PubSubClient.h забезпечує функціональність MQTT-клієнта, а Adafruit_ssd1306syp.h дозволяє взаємодіяти з дисплеєм. Також у коді визначаються пін-коди для підключення сенсорів та оголошуються константи для підключення до Wi-Fi та MQTT-сервера. Об'єкти WiFiClient, PubSubClient та Adafruit_ssd1306syp створюються для роботи з відповідними модулями.

У функції setup() виконується початкове налаштування ESP8266 (див. лістинг 2.6).

Лістинг 2.6 – Налаштування Wi-Fi та дисплея

```
void setup() {
  Serial.begin(115200);
  delay(10);
  Serial.println();
  Serial.println("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  display.initialize();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.print("MQTT Data Receiver");
  display.update();
  delay(2000);
  display.clear();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
```

Спершу ініціалізується послідовний зв'язок для виведення діагностичних повідомлень на монітор послідовного порту. Після цього відбувається підключення до Wi-Fi мережі за допомогою заданих SSID та пароля. Під час підключення виводяться відповідні повідомлення для користувача. Після успішного підключення до Wi-Fi мережі виводиться IP-адреса пристрою.

Далі відбувається ініціалізація дисплея, на якому відображається стартове повідомлення "MQTT Data Receiver". Це повідомлення допомагає переконатися

в успішному запуску програми та налаштуванні дисплея. Після короткої затримки дисплей очищається для подальшого використання.

Також у функції setup() налаштовується MQTT-клієнт, зокрема вказується сервер MQTT та функція зворотного виклику callback, яка буде викликатися при отриманні повідомлень від сервера.

У функції reconnect() відбувається підключення до MQTT-сервера (див. лістинг 2.7).

Лістинг 2.7 – Імпорт бібліотек та оголошення змінних

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP8266ClientReceiver", mqtt_user, mqtt_password)) {
      Serial.println("connected");
      client.subscribe("esp8266/temperature");
      client.subscribe("esp8266/humidity");
      client.subscribe("esp8266/water");
      client.subscribe("esp8266/light");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

Вона намагається встановити з'єднання з сервером, використовуючи задані ім'я користувача та пароль. У разі успішного підключення виводиться повідомлення "connected" та виконується підписка на відповідні топіки: esp8266/temperature, esp8266/humidity, esp8266/water та esp8266/light. Ці топіки будуть використовуватися для отримання даних про температуру, вологість, рівень води та освітлення відповідно. Якщо підключення не вдалося, виводиться код помилки та функція робить паузу на 5 секунд перед повторною спробою.

У функції callback() обробляються отримані повідомлення від MQTT-сервера (див. лістинг 2.8).

Лістинг 2.8 – Основний цикл програми

```
void callback(char* topic, uint8_t* payload, unsigned int length) {
  payload[length] = '\0';
  String message = String((char*)payload);
  if (String(topic) == "esp8266/temperature") {
    temperature = message.toFloat();
  } else if (String(topic) == "esp8266/humidity") {
    humidity = message.toFloat();
  } else if (String(topic) == "esp8266/water") {
    waterLevel = message.toInt();
  } else if (String(topic) == "esp8266/light") {
    lightLevel = message.toInt();
  }
  display.clear();
  display.setCursor(30, 0);
```

```
    display.print("Temp: ");
    display.print(temperature);
    display.println(" C");
    display.setCursor(20, 16);
    display.print("Humidity: ");
    display.print(humidity);
    display.println(" %");
    display.setCursor(40, 32);
    display.print("Water: ");
    display.println(waterLevel);

    display.setCursor(50, 48);
    display.print("Light: ");
    display.println(lightLevel);
    display.update();
}
```

При отриманні повідомлення функція спочатку визначає тему (топік) повідомлення, а потім обробляє його відповідно до змісту. Залежно від теми повідомлення (esp8266/temperature, esp8266/humidity, esp8266/water, esp8266/light), значення з payload перетворюється в потрібний формат (float або int) і зберігається у відповідних змінних: temperature, humidity, waterLevel та lightLevel.

Після оновлення змінних дисплей очищається і на ньому виводяться поточні значення температури, вологості, рівня води та освітлення. Оновлення дисплея здійснюється через відповідні методи, які встановлюють курсор та виводять текст і значення змінних.

Основний цикл програми, реалізований у функції loop(), забезпечує постійну роботу програми (див. лістинг 2.9).

Лістинг 2.9 – Основний цикл програми

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}
```

Спершу перевіряється наявність підключення до MQTT-сервера, якщо з'єднання розірване, викликається функція reconnect(), яка спробує відновити з'єднання. Після цього виконується метод client.loop(), який обробляє вхідні MQTT-повідомлення та викликає функцію зворотного виклику callback() у разі отримання нового повідомлення.

2.5 Висновки до другого розділу

У другому розділі було створено тестову схему для дослідження системи збору та передачі даних, яка базується на використанні мікроконтролерів ESP8266 для збору та передачі даних через MQTT-сервер. Встановлено, що мікроконтролери забезпечують надійне з'єднання та передачу даних, а також мають високу продуктивність і низьке енергоспоживання, що є критично важливим для автономних систем.

Детально охарактеризовано апаратні компоненти системи, включаючи мікроконтролер ESP8266, давачі температури та вологості DHT11, давач світла CT0013LS, давач рівня води CT0042CWS та дисплей SSD1306. Встановлено, що кожен з цих компонентів має свої унікальні характеристики, що дозволяють

ефективно виконувати поставлені задачі. Особливо було відзначено їх сумісність із мікроконтролерами та низьке енергоспоживання, що дозволяє використовувати їх у широкому спектрі застосувань.

Описано програмну частину системи, включаючи вибір мови програмування Arduino для розробки програмного забезпечення, характеристику брокера MQTT Mosquitto та операційної системи Ubuntu, яка використовується для розгортання серверної частини системи. Встановлено, що використання MQTT протоколу дозволяє забезпечити надійну та ефективну передачу даних між компонентами системи, а брокер Mosquitto забезпечує високу продуктивність і безпеку передачі даних.

Також було розглянуто реалізацію системи, включаючи налаштування MQTT брокера, написання програмного коду для надсилання та отримання даних з мікроконтролерів ESP8266. Встановлено, що правильно налаштований MQTT брокер та оптимізований програмний код дозволяють забезпечити безперебійну роботу системи з мінімальними затримками у передачі даних.

РОЗДІЛ 3 НАЛАШТУВАННЯ ТА ТЕСТУВАННЯ БЕЗПЕЧНОГО З'ЄДНАННЯ З MQTT

3.1 Перевірка захищеності мережевого зв'язку

Для перевірки захищеності мережевого зв'язку перехопимо трафік на Ip-адресі сервера, та порту на якому працює MQTT. Для цього виконаємо в консолі на Ubuntu server команду "`sudo tcpdump -i any host 192.168.0.104 and port 1883 -w capture.pcap`", яка виконує мережевий аналіз з використанням утиліти tcpdump. Ця команда запускається з правами root, щоб забезпечити доступ до мережевих інтерфейсів та пакетів на рівні ядра операційної системи.

Команда складається з кількох параметрів:

1) `sudo` – цей префікс використовується для виконання команди від імені суперкористувача. Це необхідно для отримання доступу до мережевих інтерфейсів та операцій з мережевими пакетами;
2) `tcpdump` – це утиліта для захоплення та аналізу мережевих пакетів. Вона дозволяє переглядати вміст мережевого трафіку в реальному часі або зберігати його для подальшого аналізу;
3) `-i any` – цей параметр вказує, що потрібно захоплювати трафік на всіх доступних мережевих інтерфейсах. Це особливо корисно в системах з кількома мережевими інтерфейсами або у випадках, коли невідомо, через який інтерфейс проходить необхідний трафік;
4) `host 192.168.0.104` – цей фільтр обмежує захоплення трафіку лише для хоста з IP-адресою 192.168.0.104. Таким чином, буде захоплюватися лише трафік, який відправляється або отримується цим хостом;
5) `and port 1883` – цей додатковий фільтр обмежує захоплення трафіку лише до пакетів, що використовують порт 1883. Порт 1883 зазвичай використовується для протоколу MQTT, який є легким протоколом обміну повідомленнями, часто використовуваним в інтернеті речей;

6) `-w capture.pcap` – цей параметр вказує на те, що захоплений трафік повинен бути збережений у файл з іменем capture.pcap. Формат файлу .pcap є стандартним форматом для збереження мережевих пакетів, який може бути відкритий і проаналізований за допомогою різних мережевих аналізаторів, таких як Wireshark;

Отримавши трафік, можна його проаналізувати за допомогою утиліти Wireshark, та перевірити порушення безпеки. На рисунку 2.9 можна побачити відкритий вміст одного з пакетів.

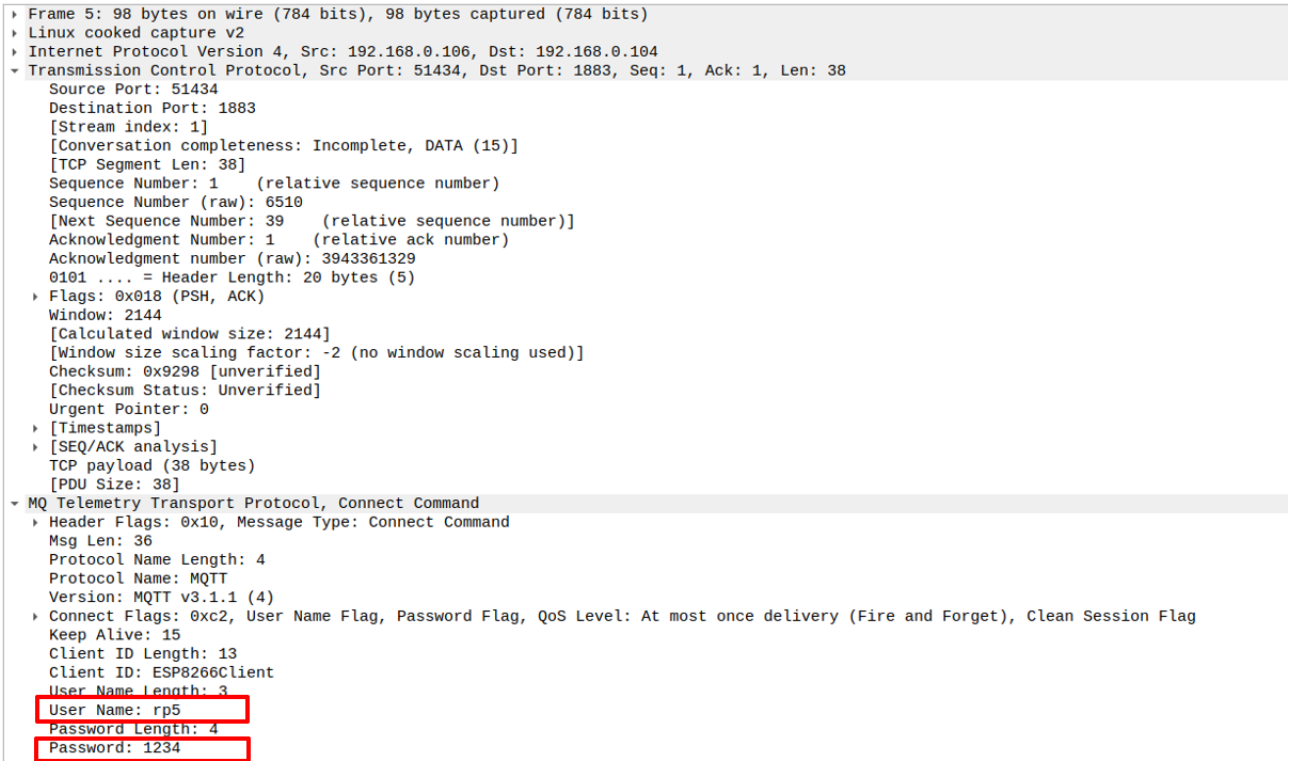

Рисунок 3.1 – Вміст перехоплених мережевих пакетів

На рисунку 3.1 продемонстровано аналіз мережевого пакету, захопленого в процесі комунікації пристроїв. У представленому прикладі зображено деталі одного з кадрів передачі даних.

Перший фрейм має загальну довжину 98 байтів (784 біта) включає заголовок протоколу Інтернету IPv4, де відображені IP-адреса джерела трафіку (192.168.0.106) і IP-адреса отримання трафіку (192.168.0.104). Наступний розділ

показує заголовок TCP, де порт джерела визначений як 51434, а призначення порт — 1883. TCP-сегмент має довжину 38 байтів і включає такі елементи:

– Порядковий номер (Sequence Number): 1
– Номер підтвердження (Acknowledgment Number): 1
– Значення вікна: 2144
– Контрольна сума: 0x9298

У фреймі також містяться прапорці TCP, такі як PSH (Push) та ACK (Acknowledgment), що вказують на різні стани зв'язку.

Також тут відображені параметри команди підключення (Connect Command), яка має наступні атрибути:

– Тип повідомлення: Connect Command
– Довжина повідомлення: 36 байтів
– Протокольне ім'я: MQTT
– Версія протоколу: 3.1.1

Прапорці підключення включають атрибути рівня обслуговування (QoS), використання імені користувача, пароля, а також підтримку чистої сесії (Clean Session Flag).

В MQTT команді підключення вказані такі дані:

– Ідентифікатор клієнта: ESP8266Client
– Ім'я користувача: rp5
– Пароль: 1234

Цей аналіз демонструє процес встановлення з'єднання за допомогою MQTT протоколу між клієнтом та сервером, включаючи всі ключові параметри, які забезпечують автентифікацію та конфігурацію сесії. Також аналіз показав, що можна легко побачити логін та пароль користувача, для підключення до MQTT сервера.

3.2 Налаштування TLS шифрування для MQTT

Напишемо bash скрипт, який є складовою системи безпеки, що забезпечує генерацію та адміністрування цифрових сертифікатів для сервера MQTT, що використовує захищені з'єднання SSL/TLS. Скрипт на мові bash автоматизує процес генерації сертифікатів для сертифікаційного центру (CA), сервера та клієнта, а також копіювання цих сертифікатів в конфігураційні каталоги брокера Mosquitto. Скрипт використовує утиліту OpenSSL для створення ключів і сертифікатів. Сервер MQTT, як центральний вузол для обміну повідомленнями між клієнтами в IoT або інших мережевих архітектурах, потребує високого рівня захисту для запобігання несанкціонованому доступу та атакам.

На лістингу 3.1 показано основні параметри для сертифікатів.

Лістинг 3.1 – Визначення змінних

```
    IP="192.168.0.104"
    SUBJECT_CA="/C=UA/ST=Ternopil/L=Ternopil/O=TNTU/OU=CA/CN=$IP"
    SUBJECT_SERVER="/C=UA/ST=Ternopil/L=Ternopil/O=TNTU/OU=Server/
CN=$IP"
    SUBJECT_CLIENT="/C=UA/ST=Ternopil/L=Ternopil/O=TNTU/OU=Client/
CN=$IP"
```

Змінна IP визначає IP-адресу, яка використовується як загальний ідентифікатор для всіх сертифікатів. Це дозволяє легко змінити IP-адресу в одному місці, якщо це необхідно для всіх сертифікатів. Кожна з наступних змінних (SUBJECT_CA, SUBJECT_SERVER, SUBJECT_CLIENT) містить рядок, що визначає ідентифікаційні дані для сертифікатів центру сертифікації, сервера та клієнта відповідно. Вони включають такі поля як країна (C), область (ST), місцезнаходження (L), організація (O), організаційна одиниця (OU) та загальне ім'я (CN), яке представлено IP-адресою.

У функції generate_CA() створюється самопідписаний сертифікат CA (див. лістинг 3.2).

Лістинг 3.2 – Функція generate_CA()

```
    function generate_server () {
       echo "$SUBJECT_SERVER"
       openssl req -nodes -sha256 -new -subj "$SUBJECT_SERVER" -
keyout server.key -out server.csr
       openssl x509 -req -sha256 -in server.csr -CA ca.crt -CAkey
ca.key -CAcreateserial -out server.crt -days 365
    }
```

Команда openssl req використовується для створення нового X.509 сертифіката з використанням SHA-256 як алгоритму хешування та RSA-2048 як алгоритму шифрування ключа.

SHA-256 належить до сімейства криптографічних хеш-функцій, розроблених Національним інститутом стандартів та технологій (NIST) США. Ця функція генерує унікальний, фіксований розмір 256-бітного (32 байта) хешу з вхідних даних. SHA-256 є частиною сімейства SHA-2, відомого своєю безпекою та використовується в різних безпекових застосуваннях та протоколах, включаючи TLS та SSL, PGP, SSH, IPsec та багато інших.

В контексті сертифікації, SHA-256 використовується для створення криптографічного хешу з усієї інформації сертифіката перед тим, як він буде підписаний цифровим підписом, що використовує приватний ключ CA. Хеш функціонує як одностороння функція, яка забезпечує детекцію будь-яких змін у даних, оскільки навіть мінімальні зміни в вхідних даних призводять до зовсім іншого виводу. Ця особливість робить SHA-256 ефективним інструментом проти спроб злому через зміни даних.

RSA, один з перших практичних методів асиметричного шифрування, де розділені ключі використовуються для шифрування та розшифрування, є фундаментальним для багатьох сучасних систем безпеки. RSA використовує два ключі: приватний ключ, який залишається таємним, та публічний ключ, який

може бути широко розповсюджений. Безпека системи RSA базується на математичній проблемі факторизації великих цілих чисел.

Розмір ключа 2048 біт в RSA означає, що приватний та публічний ключі, які використовуються для шифрування та розшифрування, мають довжину 2048 біт. Це забезпечує високий рівень безпеки, оскільки потрібна значна обчислювальна потужність для зламу ключа такої довжини. RSA-2048 рекомендується для сучасних застосувань, що вимагають високого рівня криптографічного захисту, і очікується, що він буде достатньо безпечним для комерційного використання протягом найближчих десятиліть.

Опція -x509 вказує, що це буде самопідписаний сертифікат, а -nodes вказує на створення приватного ключа без захисту паролем. Сертифікат має термін дії 365 днів, після чого його потрібно буде оновити або замінити.

У функції generate_server() генерується запит на підпис сертифіката (CSR) для сервера, а потім створює сертифікат, підписаний CA (див. лістинг 3.3).

Лістинг 3.3 – Функція generate_server()

```
function generate_server () {
   echo "$SUBJECT_SERVER"
   openssl req -nodes -sha256 -new -subj "$SUBJECT_SERVER" -
keyout server.key -out server.csr
   openssl x509 -req -sha256 -in server.csr -CA ca.crt -CAkey
ca.key -CAcreateserial -out server.crt -days 365
}
```

Спочатку генерується CSR з використанням команди openssl req з аналогічними параметрами безпеки, як у CA. Після цього CSR підписується кореневим сертифікатом з використанням команди openssl x509, що гарантує, що будь-який клієнт або сервер, який довіряє кореневому сертифікату CA, також довірятиме цьому сертифікату сервера.

Подібно до функції generate_server(), функція generate_client() створює CSR для клієнта та сертифікат, підписаний CA (див. лістинг 3.4).

Лістинг 3.4 – Функція generate_client()

```
function generate_server () {
   echo "$SUBJECT_SERVER"
   openssl req -nodes -sha256 -new -subj "$SUBJECT_SERVER" -
keyout server.key -out server.csr
   openssl x509 -req -sha256 -in server.csr -CA ca.crt -CAkey
ca.key -CAcreateserial -out server.crt -days 365
}
```

Це забезпечує, що клієнт може безпечно взаємодіяти з сервером, який також використовує сертифікати, підписані цим CA.

Функція copy_keys_to_broker(), використовується для копіювання ключів і сертифікатів, вона має важливе значення для налаштування безпеки з'єднань на сервері MQTT (див. лістинг 3.5).

Лістинг 3.5 – Функція copy_keys_to_broker()

```
function copy_keys_to_broker () {
   sudo cp ca.crt /etc/mosquitto/certs/
   sudo cp server.crt /etc/mosquitto/certs/
   sudo cp server.key /etc/mosquitto/certs/
}
```

Цей процес передбачає перенесення сертифікату центру сертифікації (CA), серверного сертифіката та відповідного приватного ключа у спеціалізовані директорії сервера. Розміщення цих файлів у визначених системою місцях забезпечує те, що сервер MQTT може використовувати їх для створення захищених SSL/TLS з'єднань. Таке налаштування дозволяє серверу перевіряти автентичність підключень та шифрувати всі передані дані, підвищуючи загальну безпеку системи.

Для роботи TLS потрібно зробити деякі зміни у конфігураційний файл Mosquitto. На рисунку 3.2 показано файл default.conf у який було внесені зміни для роботи TLS.

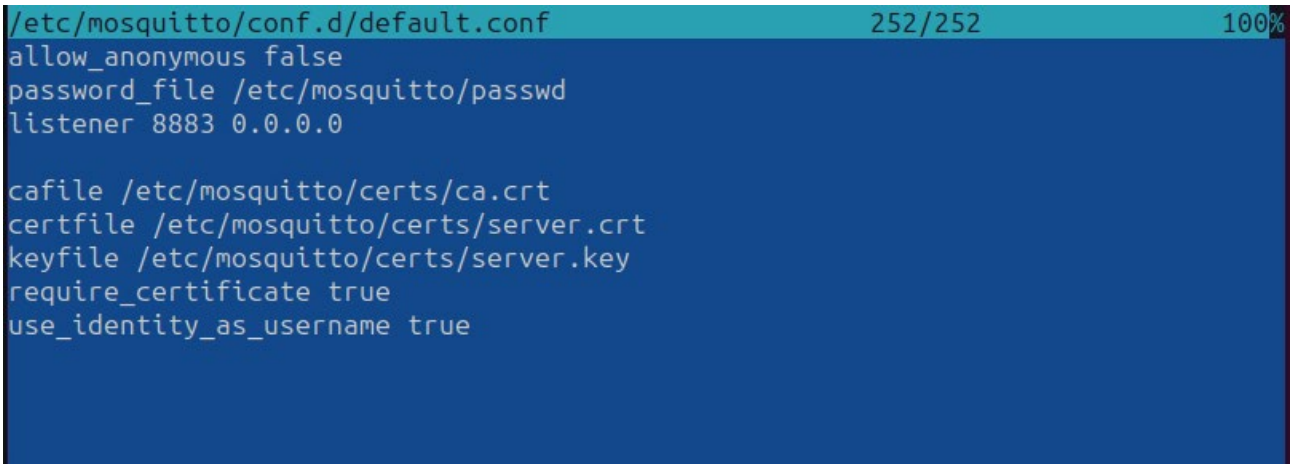

Рисунок 3.2 – Вміст конфігураційного файлу MQTT

Для роботи TLS було введено розширені параметри безпеки, що включають використання TLS/SSL для захисту даних, що передаються. Визначення cafile, certfile та keyfile демонструє використання сертифікатів для шифрування з'єднань, забезпечуючи конфіденційність та цілісність даних. Це також підвищує надійність системи шляхом верифікації з'єднань через визнані сертифікати. Крім того, параметр require_certificate встановлено в true, що зобов'язує всіх клієнтів надавати сертифікат для підключення, що значно підвищує рівень безпеки.

Також в налаштуваннях другої версії використовується порт 8883, який є стандартним для зашифрованих MQTT з'єднань. Це змінює модель взаємодії клієнта і сервера, де всі дані, що передаються, шифруються, мінімізуючи ризик перехоплення або модифікації даних третіми особами.

Введення use_identity_as_username встановлено в true дає можливість використовувати CN (Common Name) з сертифіката клієнта як ім'я користувача. Це спрощує управління ідентифікацією та аутентифікацією, дозволяючи легше інтегрувати системи автоматичного видачі та оновлення сертифікатів.

3.3 Програмний код для ESP8266 з використанням TLS

Для роботи з TLS винесемо зміни у попередні коди з додатку А та з додатку Б. Для забезпечення безпеки при передачі даних використовується TLS. В коді було додано ключові та сертифікати клієнта, а також сертифікат ЦСК (див. лістинг 2.5).

Лістинг 3.6 – Налаштування шифрування TLS при надсилання даних через MQTT

```
    void setup() {
      pinMode(BUILTIN_LED, OUTPUT);
      Serial.begin(115200);
      BearSSL::X509List        *serverTrustedCA        =        new
BearSSL::X509List(ca_cert);
      BearSSL::X509List        *serverCertList         =        new
BearSSL::X509List(client_cert);
      BearSSL::PrivateKey        *serverPrivKey        =        new
BearSSL::PrivateKey(client_private_key);
      espClient.setTrustAnchors(serverTrustedCA);
      espClient.setClientRSACert(serverCertList, serverPrivKey);
      setup_wifi();
      client.setServer(mqtt_server, 8883);
      client.setCallback(callback);
    }
```

Налаштування TLS відбувається в функції setup(), де відбувається ініціалізація сертифікатів та приватного ключа клієнта за допомогою класів BearSSL::X509List і BearSSL::PrivateKey. Об’єкт espClient налаштовується для використання цих сертифікатів і ключів.

Основні компоненти TLS:

1) Сертифікати та ключі – Код включає сертифікат та приватний ключ клієнта, а також сертифікат центру сертифікації (ЦСК), які використовуються для верифікації сторін в TLS-сесії. Сертифікати використовуються для підтвердження справжності сервера та клієнта, а

приватний ключ клієнта використовується для шифрування інформації, що передається.

2) Методи налаштування – Використання методів setTrustAnchors(), setClientRSACert(), і setPrivateKey() від класу WiFiClientSecure для налаштування сертифікатів і ключів забезпечує можливість встановлення захищеного з’єднання. Ці методи задають необхідні параметри для TLS-сесії, включаючи сертифікат ЦСК, що дозволяє клієнту перевіряти справжність сервера.
3) Порт і протокол. Зміна порту з 1883 на 8883 у вказівці client.setServer(mqtt_server, 8883) є стандартною практикою для MQTT з'єднань, що використовують SSL/TLS.

У цій функції espClient налаштовується з використанням методів setTrustAnchors() та setClientRSACert(), які приймають відповідно сертифікат ЦСК та пару сертифікат-ключ клієнта. Таке налаштування дозволяє створити захищене з’єднання з сервером MQTT на порту 8883.

3.4 Перевірка безпеки передачі даних

На рисунку 3.3 продемонстровано аналіз мережевого пакету, захопленого в процесі комунікації між двома пристроями за допомогою tcpdump. У представленому прикладі зображено деталі одного з кадрів передачі даних.

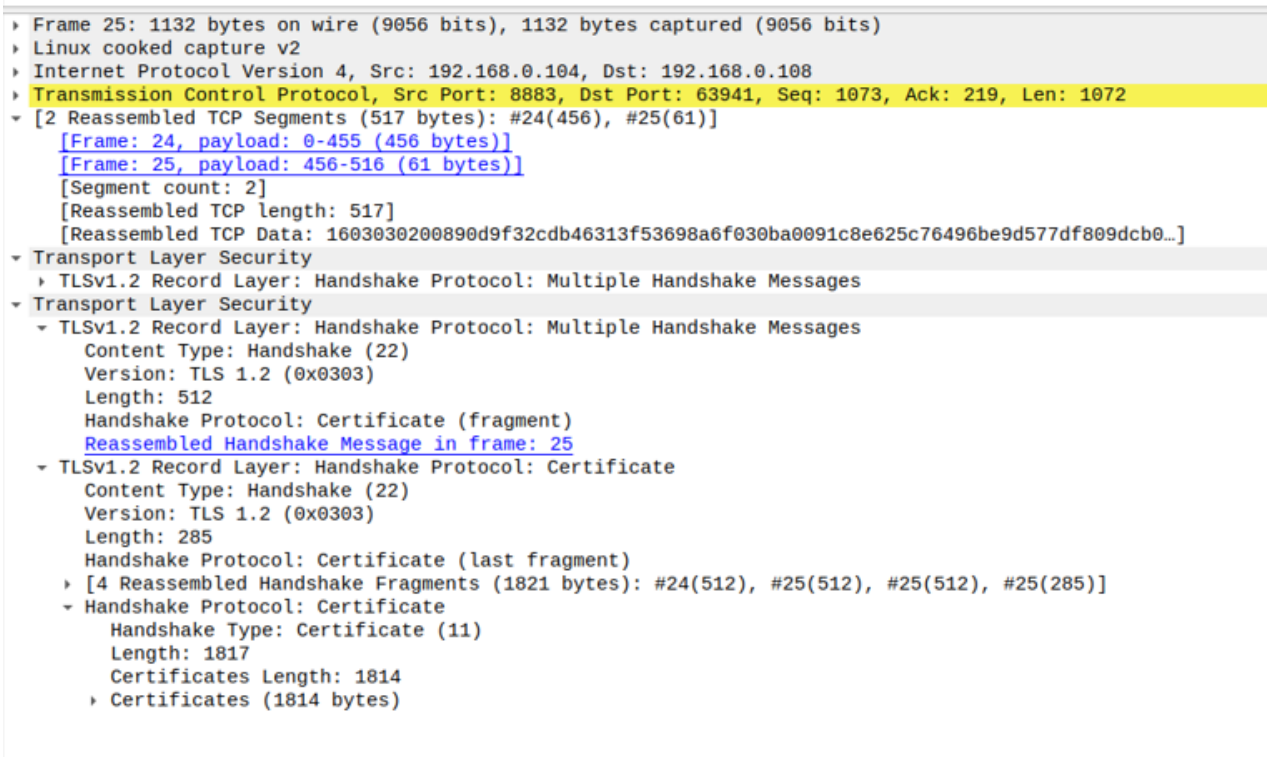

Рисунок 3.3 – Вміст перехоплених мережевих пакетів протоколу TLS

Фрейм має загальну довжину 1132 байти (9065 біт). Він включає декілька рівнів заголовків і даних, а саме:

Заголовок IP-протоколу версії 4 (IPv4):

– Вихідна IP-адреса 192.168.0.104.

– IP-адреса призначення 192.168.0.108.

Заголовок протоколу TCP:

– Вихідний порт 8883.

– Порт призначення59334.

– Порядковий номер (Seq) 1073.

– Номер підтвердження (Ack) 219.

– Довжина сегменту 1072 байти.

Зібрані TCP-сегменти:

– Перший сегмент (Frame: 24) має довжину 456 байтів (0-455 байти).

– Другий сегмент (Frame: 25) має довжину 61 байт (456-516 байти).

– Сегменти об'єднані, загальна довжина даних 517 байтів.

Рівень захисту транспортного шару (TLS):

– Рівень протоколу TLS 1.2 (0x0303)

– Перший фрагмент протоколу рукостискання (Handshake Protocol) Сертифікат (Certificate).
– Сертифікат (Certificate) довжина якого 1817 байтів.

Зібрані фрагменти рукостискання:

– Сегменти фрагментів рукостискання зібрані в один пакет загальна довжина якого становить, 1814 байтів

Представлений аналіз демонструє обробку та збирання декількох TCP-сегментів, які містять дані протоколу TLS 1.2 для встановлення захищеного з'єднання. Фрейм містить декілька рівнів заголовків, включаючи заголовки IPv4, TCP, а також дані протоколу TLS, що забезпечує безпеку зв'язку між клієнтом та сервером. Також це показує що ми не можемо перехопити логін та пароль користувача MQTT.

3.5 Висновки до третього розділу

У третьому розділі було зосереджено увагу на налаштуванні безпеки протоколу MQTT з використанням TLS шифрування.

Було проведено перевірку захищеності мережевого зв'язку шляхом перехоплення трафіку на IP-адресі сервера та порті, на якому працює MQTT. Використовуючи утиліту tcpdump, було захоплено мережеві пакети для подальшого аналізу. Це дозволило детально розглянути структуру трафіку, включаючи заголовки IP, TCP та вміст повідомлень протоколу MQTT. Аналіз за допомогою Wireshark виявив потенційні вразливості та можливості для підвищення безпеки комунікацій. Було встановлено, що є можливість перехопити логін та пароль користувача.

У другій частині розділу було розглянуто налаштування TLS шифрування для MQTT. Було написано bash скрипт, який забезпечує генерацію цифрових сертифікатів для сервера MQTT. Сценарій включав створення самопідписаного сертифіката CA, сертифікатів сервера та клієнта, а також налаштування ключів і сертифікатів на сервері MQTT. Це забезпечило захист з'єднань та автентифікацію клієнтів і сервера, що суттєво підвищило безпеку системи.

Було внесено зміни в програмний код для ESP8266, щоб використовувати TLS для захищеного з'єднання з сервером MQTT. Використовуючи бібліотеку BearSSL, було налаштовано сертифікати та приватні ключі для забезпечення шифрування та автентифікації. Зміни включали налаштування об'єкта espClient для роботи з TLS, що дозволило забезпечити захищену передачу даних між клієнтом і сервером на порту 8883.

На завершення було продемонстровано аналіз мережевого пакету, захопленого в процесі комунікації між двома пристроями за допомогою tcpdump. Аналіз показав, що з використанням TLS забезпечується високий рівень захисту даних, які передаються між клієнтом і сервером. Це підтвердило ефективність налаштувань безпеки, впроваджених у системі, та забезпечило захист від потенційних атак.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при термічних опіках

Долікарська допомога при термічних опіках є важливим кроком у зменшенні наслідків та запобіганню подальших ускладнень. Термічні опіки відбуваються, коли шкіра або м’які тканини пошкоджуються високими температурами, які можуть виникнути в результаті вогню, гарячих речовин або пари, сонячного випромінювання або інших джерел тепла. Опіки можуть бути серйозними і потребують негайного медичного втручання.

При наданні домедичної допомоги враховувати ступінь пошкодження шкіри та м’яких тканин:

1) I ступінь (еритема) – почервоніння шкіри, набряклість і біль.
2) II ступінь (утворення пухирів) – сильний біль із інтенсивним почервонінням, відшаруванням епідермісу з утворенням міхурів, наповнених рідиною.
3) III ступінь – пошкодження всієї товщі шкіри з утворенням щільного струпу, під яким перебувають ушкоджені тканини.
4) IV ступінь (обвуглення) – пошкодження всієї товщі шкіри з ушкодження м’язів, сухожиль, кісток.

Наказ Міністерства охорони здоров'я України від 09.03.2022 р. № 441 " Про затвердження порядків надання домедичної допомоги особам при невідкладних станах" встановлює порядки надання домедичної допомоги постраждалим при термічних опіках. У цьому порядку термін «термічний опік» вживається у такому значенні – це невідкладний стан, спричинений дією високих температур, в результаті чого виникає пошкодження шкіри та м’яких тканин. [15].

Надання домедичної допомоги постраждалим при термічних опіках передбачає такі кроки:

1) Переконайтеся у відсутності небезпеки для себе, оточуючих та постраждалого.

2) Припиніть дію високої температури на постраждалого та, при необхідності, зніміть тліючий одяг.
3) Зніміть прикраси, які можуть бути на ділянці опіку.
4) Заспокойте постраждалого та поясніть йому свої подальші дії.
5) Зателефонуйте на екстрений номер, щоб здійснити виклик екстреної медичної допомоги, та слухайте вказівки диспетчера.
6) Охолодіть місце опіку протягом щонайменше 20 хвилин, промиваючи його водою кімнатної температури. Це важливо, якщо площа опіку не перевищує 20% у дорослих або 10% у дітей.
7) Після охолодження накладіть на місце опіку чисту, стерильну суху марлеву пов'язку. Пов'язка не повинна надмірно тиснути на м'які тканини.
8) У разі наявності міхурів не пошкоджуйте їх. Якщо випадково міхур пошкоджено, накладіть пов'язки, як описано вище.
9) Якщо опіки охоплюють більше ніж 20% площі тіла у дорослих або 10% у дітей, накрийте постраждалого термопокривалом або покривалом;
10) Забезпечте постійний нагляд за постраждалим до прибуття бригади швидкої медичної допомоги.
11) У разі погіршення стану постраждалого до прибуття бригади швидкої медичної допомоги повторно зателефонуйте диспетчеру.
12) Якщо можливо, зберіть якомога більше інформації про обставини отримання травми від постраждалого або свідків. Передайте цю інформацію фахівцям бригади швидкої медичної допомоги або диспетчеру.

Це загальна послідовність дій, яку слід виконати, але завжди важливо дотримуватись інструкцій медичних фахівців та адаптувати допомогу до конкретної ситуації. Виконання цих кроків допоможе забезпечити постраждалому першу необхідну допомогу та зберегти його життя до прибуття медичних фахівців.

4.2 Техніка безпеки при роботі з ПК

До самостійної роботи на комп’ютерах допускаються особи, які пройшли медичний огляд, навчання по професії, вступний інструктаж з охорони праці та первинний інструктаж з охорони праці на робочому місці. В подальшому вони проходять повторні інструктажі з охорони праці на робочому місці один раз на півріччя, періодичні медичні огляди один раз на два роки [16].

Під час роботи на комп’ютерах можуть діяти такі небезпечні та шкідливі фактори, як:

– фізичні;

– психофізіологічні.

Основним обладнанням робочого місця користувача комп’ютера є монітор, системний блок та клавіатура, мишка.

Робочі місця мають бути розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін на відстані 1м, між собою на відстані не менше 1,5 м. Відносно вікон робоче місце доцільно розташовувати таким чином, щоб природне світло падало на нього збоку, переважно зліва.

Робочі місця слід розташовувати так, щоб уникнути попадання в очі прямого світла. Джерела освітлення рекомендується розташовувати з обох боків екрану паралельно напрямку погляду. Для уникнення світлових відблисків екрану, клавіатури в напрямку очей користувача, від світильників загального освітлення або сонячних променів, необхідно використовувати анти полискові сітки, спеціальні фільтри для екранів, захисні козирки, на вікнах – жалюзі.

Монітор повинен бути розташований на робочому місці так, щоб поверхня екрана знаходилася в центрі поля зору на відстані 400-700 мм від очей користувача. Рекомендується розміщувати елементи робочого місця так, щоб витримувалася однакова відстань очей від екрана, клавіатури, тексту.

Зручна робоча поза при роботі з комп’ютером забезпечується регулюванням висоти робочого столу, крісла та підставки для ніг. Раціональною робочою позою може вважатися таке положення, при якому ступні працівника розташовані горизонтально на підлозі або підставці для ніг, стегна зорієнтовані

у горизонтальній площині, верхні частини рук – вертикальні. Кут ліктєвого суглоба коливається в межах 70-90°, зап’ястя зігнуті під кутом не більше ніж 20°, нахил голови 15-20°.

Для нейтралізації зарядів статичної електрики в приміщенні, де виконується робота на комп’ютерах, в тому числі на лазерних та світлодіодних принтерах, рекомендується збільшувати вологість повітря за допомогою кімнатних зволожувачів. Не рекомендується носити одяг з синтетичних матеріалів.

Згідно статті 18 Закону України “Про охорону праці” працівник зобов’язаний:

– знати і виконувати вимоги нормативних актів про охорону праці, правила поводження з устаткуванням та іншими засобами виробництва, користуватися засобами колективного та індивідуального захисту;
– дотримуватись зобов’язань щодо охорони праці, передбачених колективним договором та правилами внутрішнього трудового розпорядку підприємства;
– співробітничати з власником у справі організації безпечних і нешкідливих умов праці, особисто вживати посильних заходів щодо усунення будь-якої виробничої ситуації, яка створює загрозу його життю чи здоров’ю, або людей, які його оточують, повідомляти про небезпеку свого безпосереднього керівника або іншу посадову особу.

Вимоги безпеки перед початком роботи:

– увімкнути систему кондиціювання в приміщенні;
– перевірити надійність встановлення апаратури на робочому столі. Повернути монітор так, щоб було зручно дивитися на екран – під прямим кутом (а не збоку) і трохи зверху вниз, при цьому екран має бути трохи нахиленим, нижній його край ближче до оператора;
– перевірити загальний стан апаратури, перевірити справність електропроводки, з’єднувальних шнурів, штепсельних вилок, розеток, заземлення захисного екрана;
– відрегулювати освітленість робочого місця;

– відрегулювати та зафіксувати висоту крісла, зручний для користувача нахил його спинки;
– приєднати до системного блоку необхідну апаратуру. Усі кабелі, що з'єднують системний блок з іншими пристроями, слід вставляти та виймати при вимкненому комп'ютері;
– ввімкнути апаратуру комп'ютера вимикачами на корпусах в послідовності: монітор, системний блок, принтер (якщо передбачається друкування);
– відрегулювати яскравість свічення монітора, мінімальний розмір світної точки, фокусування, контрастність. Не слід робити зображення надто яскравим, щоб не втомлювати очей.

Рекомендується:

– яскравість свічення екрана – не менше 100Kg/м2;
– відношення яскравості монітора до яскравості оточуючих його поверхонь в робочій зоні – не більше 3:1;
– мінімальний розмір точки свічення не більше 0,4 мм для монохромного монітора і не менше 0,6 мм для кольорового, контрастність зображення знаку – не менше 0,8.

При виявленні будь-яких несправностей роботу не розпочинати, повідомити про це керівника.

Вимоги безпеки під час виконання роботи:

– необхідно стійко розташовувати клавіатуру на робочому столі, не опускати її хитання. Під час роботи на клавіатурі сидіти прямо, не напружуватися;
– для забезпечення несприятливого впливу на користувача пристроїв типу "миша" належить забезпечувати вільну велику поверхню столу для переміщення "миші" і зручного упору ліктьового суглоба;
– не дозволяються сторонні розмови, подразнюючі шуми;
– періодично при вимкненому комп'ютері прибирати ледь змоченою мильним розчином бавовняною ганчіркою порох з поверхонь апаратури. Екран протирають ганчіркою, змоченою у спирті. Не дозволяється

використовувати рідинні або аерозольні засоби чищення поверхонь комп’ютера.

Психофізіологічне розвантаження є одним з варіантів зменшення стресу.

Дана практика включає в себе застосування методу аутогенного тренування, що передбачає свідоме використання комплексу прийомів психічної саморегуляції та виконання простих фізичних вправ зі словесним самонавіюванням. Основна увага приділяється розслабленню м'язів (релаксації).

Під час сеансів психофізіологічного розвантаження рекомендується використовувати три періоди, що відповідають фазам відновлення:

Перший період - абстрагування від виробничого середовища, що відповідає фазі залишкового збудження. В цей час відтворюється повільна мелодійна музика та звуки пташиного співу. Працівники знаходять зручну позу та психологічно готуються до наступних періодів.

Другий період – заспокоєння, що відповідає фазі відновлювального гальмування. Показуються фотослайди з зображеннями природи, таких як квітучі луки, березові гаї, ставки і т.д. Звуковий супровід включає спокійну музику та заспокійливі формули аутогенного тренування.

Третій період - активізація, що відповідає фазі підвищеної збудженості. Спочатку світло повністю вимикається, а потім на екрані появляється червона пляма, розмір і яскравість якої поступово збільшуються. В кінці періоду звучить бадьора музика, а працівники виконують мобілізуючі формули аутогенного тренування, попередньо зробивши глибоке вдихання та видихання.

Сеанси психофізіологічного розвантаження можуть проводитись за єдиною програмою через індивідуальні навушники і складатись із двох періодів по 5 хвилин кожний: повне розслаблення та активізація працездатності. При необхідності, на фоні музики можуть використовуватись фрази, що сприяють відпочинку, покращенню самопочуття та бадьорості на заключному етапі. Після сеансів психофізіологічного розвантаження працівники відчувають зменшення втоми, з'являється бадьорість та гарний настрій, а загальний стан помітно поліпшується.

Додатково до сеансів психофізіологічного розвантаження, працівники також можуть скористатись іншими методами для зниження стресу та покращення психологічного благополуччя.

Одним з ефективних підходів є впровадження регулярних перерв під час робочого дня. Це можуть бути короткі паузи, під час яких працівники займаються розслаблюючими вправами, дихальними техніками або просто відпочивають. Це допомагає знизити напругу і покращити фокусування під час робочого процесу.

Також важливо створити комфортне робоче середовище для працівників. Це може включати забезпечення комфортних стільців і столів, добре освітлення та достатню вентиляцію. Природне освітлення та наявність рослин у приміщенні також можуть позитивно вплинути на настрій та самопочуття працівників.

Підтримка від керівництва та колег також має важливе значення. Створення сприятливого та підтримуючого робочого середовища, де працівники можуть відчути підтримку та співпрацю, сприяє зниженню стресу та покращує загальний настрій в колективі.

Крім того, особисте самоуправління і здібність до саморегуляції є важливими навичками для працівників. Це включає уміння регулювати власні емоції, реагувати на стресові ситуації та знаходити способи їх подолання, наприклад, за допомогою медитації, йоги або інших релаксаційних технік.

Усі ці підходи сприяють створенню здорової та продуктивної робочої атмосфери, де працівники можуть ефективно керувати стресом, забезпечуючи своє фізичне та емоційне благополуччя.

4.3 Висновки до четвертого розділу

У четвертому розділі було розглянуто основні аспекти безпеки життєдіяльності та охорони праці, що мають вирішальне значення для забезпечення здорових та безпечних умов праці. Долікарська допомога при термічних опіках є важливою складовою надання першої допомоги, яка дозволяє мінімізувати наслідки ушкоджень та запобігти ускладненням. Було детально

описано кроки, які необхідно здійснити при наданні допомоги постраждалим, включаючи охолодження опіків, накладення стерильних пов'язок та забезпечення постійного нагляду до прибуття медичних фахівців. Виконання цих кроків є критично важливим для збереження життя та здоров'я постраждалих.

Окрім того, було детально розглянуто техніку безпеки при роботі з ПК, що включає в себе вимоги до розташування робочих місць, налаштування обладнання та дотримання відповідних інструкцій. Особлива увага була приділена забезпеченню комфортних умов роботи, правильного розташування монітора та інших компонентів, а також заходів для запобігання впливу шкідливих факторів. Раціональне облаштування робочого місця, регулярне проведення перерв та сеансів психофізіологічного розвантаження сприяють зниженню стресу та покращенню загального самопочуття працівників.

Загалом, результати четвертого розділу підтверджують, що дотримання правил охорони праці та надання своєчасної долікарської допомоги є необхідними умовами для забезпечення безпечних та здорових умов праці. Системний підхід до організації безпеки на робочому місці та підвищення обізнаності працівників щодо методів надання першої допомоги дозволяють знизити ризики травматизму та забезпечити високий рівень захисту здоров'я. Інтеграція цих аспектів у щоденну практику сприяє створенню безпечного та ефективного робочого середовища..

ВИСНОВКИ

У даній кваліфікаційній роботі було розроблено систему безпечної взаємодії компонентів IoT на базі мікроконтролерів ESP8266 та протоколу MQTT з шифруванням TLS. Під час дослідження детально розглянуто критичні аспекти безпеки пристроїв IoT, включаючи аналіз протоколів шифрування TLS, SSL та DTLS. Було показано, що, впровадження багатофакторної аутентифікації, складних паролів, регулярного оновлення програмного забезпечення та шифрування даних є критично важливими заходами для забезпечення належного рівня безпеки компонентів IoT.

У другому розділі досліджено засоби забезпечення надійної взаємодії між компонентами IoT системи, зосередивши увагу на використанні мікроконтролерів ESP8266 для збору та передачі даних через MQTT-сервер. Було детально охарактеризовано апаратні компоненти системи, включаючи мікроконтролери та давачі для вимірювання температури, вологості, освітленості та рівня води. Описано програмну частину системи та брокер MQTT Mosquitto. Показано, що коректне налаштований MQTT брокер та оптимізований програмний код забезпечують безперебійну роботу системи з мінімальними затримками у передачі даних.

Третій розділ присвячено тестуванню системи та налаштуванню безпеки для протоколу MQTT з використанням TLS шифрування. Проведено перевірку безпеки мережевого зв'язку та аналіз мережевого трафіку, що дозволило виявити потенційні вразливості. Налаштування TLS шифрування забезпечило захист з'єднань та автентифікацію клієнтів і сервера, підвищивши безпеку системи. Було написано програмний код для ESP8266 з використання TLS, що забезпечило захищену передачу даних між клієнтом і сервером. Аналіз мережевого трафіку підтвердив високий рівень захисту даних та ефективність налаштувань безпеки.

Результати дослідження підтверджують ефективність застосування засобів безпеки та протоколів шифрування для забезпечення надійної взаємодії та захисту даних у системах IoT. Це відкриває нові можливості для підвищення

безпеки та ефективності роботи IoT пристроїв у різних галузях, таких як промисловість, домашня автоматизація та медицина. Система демонструє високу продуктивність та надійність, що підтверджує її практичну цінність для використання в реальних умовах експлуатації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. MQTT Core Concepts | EMQX Docs. EMQX Documentation. URL: https://docs.emqx.com/en/emqx/latest/messaging/mqtt-concepts.html (date of access: 21.06.2024).

2. Тимощук, В., Карташов, В., Королюк, Р. І., & Рубен, Т. (2022). Огляд протоколів керування для побудови автоматизованих систем віддаленого управління. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “Тернопільського національного технічного університету імені Івана Пулюя, 143-144

3. Тимощук, В., Долінський, А., & Тимощук, Д. (2024). СИСТЕМА ЗМЕНШЕННЯ ВПЛИВУ DOS-АТАК НА ОСНОВІ MIKROTIK. Матеріали конференцій МЦНД, (17.05. 2024; Ужгород, Україна), 198-200. https://doi.org/10.62731/mcnd-17.05.2024.008

4. Тимощук, В., & Стебельський, М. (2023). Шифрування даних в операційних системах. Матеріали VI Міжнародної студентської науково-технічної конференції „Природничі та гуманітарні науки. Актуальні питання “, 183-184.

5. Skorenkyy, Y., Kozak, R., Zagorodna, N., Kramar, O., & Baran, I. (2021, March). Use of augmented reality-enabled prototyping of cyber-physical systems for improving cyber-security education. In Journal of Physics: Conference Series (Vol. 1840, No. 1, p. 012026). IOP Publishing.

6. Karnaukhov, A., Tymoshchuk, V., Orlovska, A., & Tymoshchuk, D. (2024). USE OF AUTHENTICATED AES-GCM ENCRYPTION IN VPN. Матеріали конференцій МЦНД, (14.06. 2024; Суми Україна), 191-193. https://doi.org/10.62731/mcnd-14.06.2024.004

7. ESP8266 IOT Board(Arduino IDE or NodeMCU Lua Programming) - Elecrow Wiki. Elecrow: PCB Prototype & Open Hardware For Makers. URL: https://www.elecrow.com/wiki/esp8266-iot-boardarduino-ide-or-nodemcu-lua-programming.html (date of access: 21.06.2024).

8. Crowtail- IR Reflective Sensor - Elecrow Wiki. Elecrow: PCB Prototype & Open Hardware For Makers. URL: https://www.elecrow.com/wiki/crowtail--temperature26-humidity-sensor.html (date of access: 21.06.2024).

9. Бекер, І., Тимощук, В., Маслянка, Т., & Тимощук, Д. (2023). МЕТОДИКА ЗАХИСТУ ВІД ПОВІЛЬНИХ ТА ШВИДКИХ BRUTE-FORCE АТАК НА IMAP СЕРВЕР. Матеріали конференцій МНЛ, (17 листопада 2023 р., м. Львів), 275-276.

10. Crowtail- Water Sensor - Elecrow Wiki. Elecrow: PCB Prototype & Open Hardware For Makers. URL: https://www.elecrow.com/wiki/crowtail--water-sensor.html (date of access: 21.06.2024).

11. Іваночко, Н., Тимощук, В., Букатка, С., & Тимощук, Д. (2023). РОЗРОБКА ТА ВПРОВАДЖЕННЯ ЗАХОДІВ ЗАХИСТУ ВІД UDP FLOOD АТАК НА DNS СЕРВЕР. Матеріали конференцій МНЛ, (3 листопада 2023 р., м. Вінниця), 177-178.

12. Arduino ide documentation. arduino.cc. URL: https://docs.arduino.cc/software/ide/ (date of access: 21.06.2024).

13. Тимощук, В., & Стебельський, М. (2023). Шифрування даних в операційних системах. Матеріали VI Міжнародної студентської науково-технічної конференції „Природничі та гуманітарні науки. Актуальні питання “, 183-184.

14. Ubuntu Server documentation. Enterprise Open Source and Linux | Ubuntu. URL: https://ubuntu.com/server/docs (date of access: 21.06.2024).

15. Djärv, T., Douma, M., Palmieri, T., Meyran, D., Berry, D., Kloeck, D., ... & Goolsby, C. (2022). Duration of cooling with water for thermal burns as a first aid intervention: a systematic review. Burns, 48(2), 251-262.

16. How can you protect yourself when working with computer components?. LinkedIn: Log In or Sign Up. URL: https://www.linkedin.com/advice/1/how-can-you-protect-yourself-when-working-computer (date of access: 21.06.2024).

ДОДАТКИ