Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі»

Виконав: ст. 2-го курсу, групи КІТС-22 мз
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

_____ Гусєв О.О.
(прізвище та ініціали)

Керівник: к.т.н., доцент. каф. МБІС
_____ Карпінець В.В.
(прізвище та ініціали)
«____» _____ 2024 р.

Опонент: к.т.н., доцент, каф. ОТ
_____ Колесник І.С.
(прізвище та ініціали)
«____» _____ 2024 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

_____ Юрій ЯРЕМЧУК
“_____” _____ 2024 р.

Вінниця ВНТУ – 2024 рік
Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
_____ **Юрій ЯРЕМЧУК**
“_____” _____ 2024 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Гусєв Олексій Олексійович
(прізвище, ім`я, по-батькові)

1. Тема роботи:

«Підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі»
Керівник роботи_: Карпінець В.В. к.т.н., доцент каф. МБІС
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “ 11” березня 2024 року № 81
2. Строк подання студентом роботи_за тиждень до захисту.__
3. Вихідні дані до роботи:
Стандарти, електронні джерела, підручники та наукові статті по темі, які стосуються теми магістерської кваліфікаційної роботи.
4. Зміст текстової частини:
У вступі викладено актуальність теми та сформульовано мету роботи. У першому розділі розглянуто теоретичні засади створення вдосконаленої системи на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі. У другому розділі проведено проектування системи підвищення захищеності приміщень, визначено особливості захисту приміщень за допомогою IoT пристроїв, розроблено схему роботи пристроїв та обміну зашифрованими даними у реальному часі. У третьому розділі здійснено програмну реалізацію

системи, включаючи вибір програмної мови, реалізацію серверної та апаратної частини, систему обміну даними та тестування системи.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)
У дугому розділі магістерської кваліфікаційної роботи наведено 5 рисунків, у третьому розділі – 13 рисунків

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина			
I	Карпінець В.В.к.т.н., доцент каф. МБІС		
II	Карпінець В.В.к.т.н., доцент каф. МБІС		
III	Карпінець В.В.к.т.н., доцент каф. МБІС		
Економічна частина			
IV	Причепа І.В., к.е.н., доц. каф. ЕПВМ		

7.Дата видачі завдання _____2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1.	Визначення напрямку магістерської роботи, формулювання теми	1.03.2024	15.03.2024	
2.	Аналіз предметної області обраної теми	15.03.2024	1.04.2024	
3.	Розробка роботи	1.04.2024	10.04.2024	
4.	Написання магістерської роботи на основі розробленої теми	10.04.2024	20.04.2024	
5.	Передзахист магістерської кваліфікаційної роботи	20.04.2024	10.05.2024	
6.	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	10.05.2024	4.06.2024	
7.	Захист магістерської кваліфікаційної роботи	10.06.2024	10.06.2024	

Студент _____ Гусєв О.О.
(підпис)

Керівник роботи _____ Карпінець В.В.
(підпис)

АНОТАЦІЯ

УДК 004.56.5(043.2)

Гусєв О.О. Підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі

Магістерська кваліфікаційна робота зі спеціальності 125 – «Кібербезпека», освітня програма «Кібербезпека інформаційних технологій та систем». Вінниця: ВНТУ, 2024. 99с.

На укр.мові. Бібліогр.: 41 назв; рис.: 20; табл. 12

У магістерській кваліфікаційній роботі теоретичні засади функціонування IoT пристроїв, їх архітектурні особливості, а також методи збору, обробки та шифрування даних. Аналізуються основні протоколи обміну даними в IoT системах з точки зору забезпечення інформаційної безпеки.

У другому розділі проведено проектування системи захисту приміщень, визначено особливості взаємодії між IoT пристроями та розроблено схему обміну зашифрованими даними у реальному часі. У третьому розділі наведено програмну реалізацію системи, включаючи вибір програмної мови, реалізацію серверної та апаратної частини, а також тестування системи.

Економічна частина роботи містить оцінку комерційного потенціалу розробленого програмного забезпечення, прогнозування витрат на його впровадження та прогнозування комерційних ефектів. Проведено розрахунок ефективності інвестицій та періоду їх окупності.

Робота демонструє практичну цінність розробленої системи, її ефективність у підвищенні рівня захищеності приміщень та можливість комерційного впровадження.

Ключові слова: IoT, захищеност приміщень, шифрування, обмін даних в реальному часі, збір інформації

ABSTRACT

Gusev O.O. Increasing the security of premises with an improved system based on autonomous IoT devices for collecting information and exchanging encrypted data in real time

Master's qualification thesis on specialty 125 - "Cybersecurity", educational program "Cybersecurity of information technologies and systems". Vinnytsia: VNTU, 2024. 99p.

In the Ukrainian language. Bibliography: 41 titles; Fig.: 20; table 12

In the master's qualification work, the theoretical principles of the functioning of IoT devices, their architectural features, as well as the methods of data collection, processing and encryption. The main data exchange protocols in IoT systems are analyzed from the point of view of ensuring information security.

In the second section, the design of the premises protection system was carried out, the features of interaction between IoT devices were determined, and a scheme for exchanging encrypted data in real time was developed. The third chapter describes the software implementation of the system, including the selection of the programming language, implementation of the server and hardware parts, as well as system testing.

The economic part of the work includes an assessment of the commercial potential of the developed software, forecasting the costs of its implementation and forecasting commercial effects. The calculation of investment efficiency and their payback period was carried out.

The work demonstrates the practical value of the developed system, its effectiveness in increasing the level of security of premises and the possibility of commercial implementation.

Keywords: IoT, security of premises, encryption, real-time data exchange, information collection

ЗМІСТ

ВСТУП

Актуальність. З розвитком технологій та зростанням обсягу інформації, яку ми обробляємо і передаємо щодня, питання безпеки стають надзвичайно важливими. Інтернет речей (IoT) - це технологія, яка з'єднує різні пристрої у мережу для автоматичного збору, обробки та обміну даними. Ці пристрої активно використовуються в багатьох сферах, включаючи безпеку приміщень. Водночас зростає й кількість кіберзагроз, які можуть спричинити значні збитки, якщо інформація не буде належним чином захищена.

Питання захищеності IoT пристроїв є надзвичайно актуальним у сучасному світі. Згідно зі статистикою, кількість кібератак на IoT пристрої постійно зростає, що обумовлює необхідність впровадження нових методів захисту даних. Особливо це стосується систем, що використовуються для забезпечення безпеки приміщень, де компрометація даних може призвести до фізичної шкоди та великих фінансових втрат.

Сучасні IoT системи повинні не лише ефективно збирати та обробляти дані, але й забезпечувати їхню безпеку. Це вимагає впровадження вдосконалених методів шифрування та обміну даними у реальному часі, що гарантує захист інформації від несанкціонованого доступу. Враховуючи швидкий розвиток технологій і підвищення вимог до захищеності даних, дослідження у цій сфері є надзвичайно актуальними та своєчасними.

Таким чином, розробка вдосконаленої системи захисту приміщень, що базується на автономних IoT пристроях збору інформації та обміну зашифрованими даними у реальному часі, є актуальним завданням, яке має вагоме значення для забезпечення безпеки сучасного інформаційного суспільства.

Метою даної роботи є розробка вдосконаленої системи підвищення захищеності приміщень на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі.

Задачами дослідження є:

– аналіз основних принципів функціонування iot пристроїв та архітектурних особливостей iot систем.

– вивчення методів збору та обробки даних в реальному часі.

– огляд і аналіз методів шифрування даних в системах iot.

– дослідження основних протоколів обміну даними в iot системах та їх використання для захисту інформації.

– проектування схеми роботи системи захисту приміщень за допомогою iot пристроїв.

– розробка схеми обміну зашифрованими даними у реальному часі між iot пристроями.

– програмна реалізація системи, включаючи серверну та апаратну частини.

– тестування розробленої системи.

– економічне обґрунтування розробки та впровадження програмного забезпечення.

Об'єктом дослідження є системи захисту приміщень, засновані на використанні автономних IoT пристроїв.

Предметом дослідження є методи та технології збору, обробки та шифрування даних у реальному часі в автономних IoT системах для забезпечення підвищеної захищеності приміщень.

Новизна роботи полягає у розробці комплексної системи захисту приміщень, яка поєднує в собі передові методи шифрування даних та обмін ними у реальному часі між автономними IoT пристроями. Це дозволяє забезпечити більш високий рівень безпеки та оперативність реагування на загрози.

Практична цінність роботи полягає у створенні ефективної системи захисту приміщень, яка може бути впроваджена як у приватних домогосподарствах, так і в корпоративних об'єктах. Розроблена система дозволить знизити ризики несанкціонованого доступу до приміщень та забезпечити захист конфіденційної інформації, що передається між IoT пристроями.

1 ТЕОРЕТИЧНІ ЗАСАДИ СТВОРЕННЯ ВДОСКОНАЛЕНОЇ СИСТЕМИ НА ОСНОВІ АВТОНОМНИХ ІОТ ПРИСТРОЇВ ЗБОРУ ІНФОРМАЦІЇ ТА ОБМІНУ ЗАШИФРОВАНИМИ ДАНИМИ У РЕАЛЬНОМУ ЧАСІ

У першому розділі магістерської кваліфікаційної роботи будуть розглянуті теоретичні засади створення системи на основі автономних IoT пристроїв для збору інформації та обміну даними в реальному часі. Підрозділи розділу охоплюватимуть основні принципи функціонування IoT пристроїв, архітектурні особливості систем IoT, методи збору та обробки даних в реальному часі, методи шифрування даних у системах IoT, аналіз основних протоколів обміну даними в IoT системах та їх використання для захисту інформації, а також висновки та постановка задачі.

1.1 Основні принципи функціонування IoT пристроїв

У наш час інтернет речей (IoT) відіграє значущу роль у побуті та промисловості, забезпечуючи підключеність та автоматизацію великої кількості пристроїв і систем. Проте разом з цим зростають і виклики, пов'язані з безпекою та захистом інформації, яка передається та оброблюється IoT пристроями. Однією з ключових аспектів є захист приміщень та об'єктів за допомогою автономних IoT пристроїв, які забезпечують збір та обмін даними в реальному часі [1].

IoT-пристрої (Internet of Things) – це широкий спектр фізичних пристроїв, які оснащені датчиками, програмним забезпеченням та підключені до Інтернету. Ці пристрої можуть збирати дані про навколишнє середовище, взаємодіяти з іншими пристроями та управлятися віддалено.

Основні принципи функціонування IoT-пристроїв визначаються їх роллю у зборі, передачі, обробці, зберіганні даних, управлінні та взаємодії з іншими пристроями. Детальніше розглянемо кожен з цих принципів:

– збір даних;

Збір даних є одним із основних принципів функціонування IoT-пристроїв і відіграє ключову роль у їх роботі. Цей процес полягає в зібранні різноманітних

інформаційних параметрів з навколишнього середовища за допомогою вбудованих датчиків.

Датчики можуть бути різного типу і вимірювати різні параметри, такі як температура, вологість, тиск, рух, звук, світло, рівень CO2 та інші. Вони можуть бути встановлені на різних об'єктах або пристроях, наприклад, на датчиках рівня рідини, вбудованих у побутові пристрої або використовуватися в сільському господарстві для моніторингу ґрунту [2].

Після збору даних, вони можуть бути конвертовані у цифровий формат та передані для подальшої обробки. Збір даних дозволяє IoT-пристроям отримувати інформацію про навколишнє середовище, що є важливим для подальшого аналізу та прийняття рішень.

Важливим аспектом збору даних є їх якість і точність, оскільки від цього залежить коректність подальшого аналізу та прийняття рішень. Також важливо забезпечити безпеку зібраних даних, особливо у випадку, коли вони містять конфіденційну інформацію.

– передача даних;

Передача даних є ключовим етапом у функціонуванні IoT-пристроїв, оскільки вона забезпечує потік інформації між пристроями та іншими частинами системи. Для цього використовуються бездротові мережі, які забезпечують зв'язок між пристроями на великій відстані без необхідності фізичного підключення кабелями [3].

Wi-Fi (802.11): Це один з найпоширеніших протоколів бездротового зв'язку, який дозволяє передавати дані на відстань кількох десятків метрів. Wi-Fi забезпечує високу швидкість передачі даних та досить надійний зв'язок, що робить його популярним в застосуваннях IoT.

Bluetooth: Цей протокол використовується для підключення пристроїв на невеликій відстані, зазвичай до 10 метрів. Bluetooth широко використовується в побутових пристроях та дозволяє енергоефективну передачу даних.

Zigbee: Цей протокол призначений для мереж, в яких велика кількість пристроїв спільно використовує канал зв'язку. Він використовується для створення

мереж "розумного будинку" та "розумного міста", де велика кількість датчиків та пристроїв потребує зв'язку

LoRaWAN (Long Range Wide Area Network): Це протокол для створення мереж великого радіусу дії з низькою швидкістю передачі даних, але з великою зоною покриття. LoRaWAN часто використовується в системах моніторингу та вимірювання на великих відстанях.

Ці бездротові мережі дозволяють передавати дані в режимі реального часу, що дозволяє вчасно отримувати та обробляти інформацію з IoT-пристроїв. Важливим аспектом є вибір оптимального протоколу зв'язку, який враховує потрібну швидкість передачі даних, дальність зв'язку, витрати енергії та інші вимоги конкретного застосування [4].

– обробка даних;

Обробка даних в системах Інтернету речей (IoT) є ключовим етапом, оскільки вона дозволяє аналізувати та вживати дієві заходи на основі зібраних інформаційних потоків. Варіанти обробки даних включають:

Локальна обробка на пристрої: Деякі IoT-пристрої мають вбудовані процесори та програмне забезпечення, яке дозволяє їм обробляти дані навіть без зовнішнього зв'язку. Це може включати фільтрацію, агрегацію або зведення даних ще до їх передачі по мережі. Локальна обробка може бути корисною для зменшення обсягу передаваних даних та зменшення затримок у відповіді на події.

Обробка на локальному сервері: Деякі дані можуть вимагати більш складного аналізу або потребують збереження для подальшого використання. У таких випадках, дані можуть бути відправлені на локальний сервер для подальшої обробки. Це може включати використання алгоритмів машинного навчання для прогнозування подій або виявлення аномалій.

Хмарна обробка: У складних системах IoT, коли обсяг та складність даних великі, може використовуватися хмарний сервіс для обробки та аналізу інформації. Хмарні рішення можуть забезпечувати широкий спектр сервісів, включаючи потужність обчислень, бази даних, аналітику та інші.

Розподілена обробка даних: У деяких випадках, обробка даних може бути розподілена між декількома вузлами або пристроями. Це може бути корисно для великих обсягів даних або сценаріїв, де швидка обробка даних важлива.

Обробка даних в системах IoT може бути реалізована різними способами в залежності від вимог конкретного застосування, обсягу даних, доступності ресурсів та інших факторів. Однак вона завжди має на меті витягнути корисну інформацію з сирого потоку даних для подальшого використання [5].

– зберігання даних;

Зберігання даних є важливою складовою систем Інтернету речей (IoT), оскільки воно дозволяє зберегти та забезпечити доступ до оброблених інформаційних потоків. Основні аспекти зберігання даних в системах IoT включають:

Локальне зберігання на пристрої: Деякі IoT-пристрої можуть мати вбудовану пам'ять або накопичувачі для збереження оброблених даних. Це може бути корисним у випадках, коли з'єднання з сервером або хмарою недоступне або небажане. Наприклад, датчики, які збирають дані про температуру або вологість, можуть зберігати ці дані локально до тих пір, поки з'єднання не буде відновлене.

Локальне зберігання на сервері: Оброблені дані можуть бути збережені на локальному сервері, який знаходиться в межах тієї ж мережі, що й IoT-пристрій. Це дозволяє забезпечити доступ до даних з інших пристроїв у мережі або для аналізу за допомогою додаткових програм або сервісів.

Хмарне зберігання: Дані можуть бути відправлені до хмарного сервісу для зберігання на віддалених серверах. Це дозволяє отримати доступ до даних з будь-якої точки з'єднання з Інтернетом і забезпечує можливість резервного копіювання та відновлення даних [6].

Резервне копіювання і архівування: Для забезпечення надійності даних і захисту від втрати можуть застосовуватися стратегії резервного копіювання та архівування. Це може включати створення резервних копій на локальних пристроях або у хмарному сервісі, а також зберігання інформації в архівних сховищах для подальшого використання.

Заходи безпеки: При зберіганні даних важливо враховувати питання безпеки, такі як шифрування, контроль доступу та захист від несанкціонованого доступу. Це дозволяє зберегти конфіденційність та цілісність даних.

Зберігання даних в системах IoT може бути організоване різними способами в залежності від вимог застосування, рівня безпеки та доступності ресурсів. Однак воно завжди має на меті забезпечити надійність та доступність даних для подальшого використання або аналізу.

– управління пристроями;

Управління пристроями є важливою функцією в системах Інтернету речей (IoT), оскільки воно дозволяє користувачам керувати та моніторити пристрої з віддаленої точки з'єднання. Основні аспекти управління пристроями включають:

– віддалений доступ через веб-інтерфейс: Багато IoT-пристроїв мають веб-інтерфейс, до якого можна отримати доступ через Інтернет за допомогою веб-браузера. Цей інтерфейс дозволяє користувачам налаштовувати параметри пристрою, переглядати дані та виконувати дії з віддаленої точки.

– мобільні додатки: Деякі IoT-пристрої постачаються з мобільними додатками, які можна встановити на смартфон або планшет. Ці додатки надають зручний інтерфейс для взаємодії з пристроями, дозволяючи користувачам керувати ними та отримувати повідомлення про стан пристроїв.

– API (інтерфейс програмування додатків): Деякі IoT-пристрої надають API, яке дозволяє розробникам створювати власні програми для керування пристроями. Це може бути корисним для автоматизації операцій або інтеграції пристроїв з іншими системами.

– групове управління: Деякі платформи IoT надають можливість групового управління пристроями. Це дозволяє користувачам керувати кількома пристроями одночасно та виконувати масштабні операції, такі як оновлення програмного забезпечення або налаштування параметрів групи пристроїв.

– моніторинг та журналювання: Управління пристроями може включати моніторинг їхнього стану та журналювання подій. Це дозволяє виявляти проблеми та вживати відповідних заходів для їх вирішення.

– безпека доступу: Управління пристроями може також включати заходи безпеки, такі як аутентифікація та авторизація користувачів, щоб запобігти несанкціонованому доступу до пристроїв.

Усі ці аспекти управління пристроями спільно створюють зручну та ефективну інфраструктуру для керування пристроями IoT з будь-якого місця та в будь-який час [6].

– взаємодія з іншими пристроями.

Взаємодія з іншими пристроями є ключовим аспектом сучасних систем Інтернету речей (IoT), оскільки це відкриває широкі можливості для створення складних та інтелектуальних систем. Детальніше розглянемо основні аспекти взаємодії IoT-пристроїв з іншими пристроями:

Протоколи зв'язку: Для взаємодії між пристроями використовуються різні протоколи зв'язку, такі як MQTT, CoAP, HTTP, які дозволяють передавати дані між пристроями через різні канали зв'язку.

Взаємодія за допомогою хмарних платформ: Хмарні платформи IoT надають зручні інструменти для взаємодії між пристроями, дозволяючи передавати та обробляти дані в хмарному середовищі.

Системи керування: IoT-пристрої можуть взаємодіяти з системами керування, такими як системи автоматизації будівель, системи управління енергоефективністю або системи безпеки, для спільного вирішення завдань та оптимізації ресурсів.

Інтеграція зі смарт-дом системами: IoT-пристрої для дому можуть взаємодіяти з іншими смарт-пристроями, такими як розумні датчики, розумні світильники, розумні термостати тощо, для створення інтелектуальних систем управління домашньою автоматизацією.

Мережі сенсорів: IoT-пристрої можуть бути підключені до мереж сенсорів, де вони можуть обмінюватися даними та спільно вирішувати завдання зі збору та обробки інформації.

Інтеграція з великими системами: IoT-пристрої можуть взаємодіяти з великими системами, такими як системи моніторингу, системи управління виробництвом або логістичні системи, для забезпечення збору та обробки данних у реальному часі та автоматизації різних процесів.

Ці аспекти взаємодії дозволяють створювати потужні та ефективні системи, які можуть реалізувати різноманітні функції та задачі у різних галузях, починаючи від дому та закінчуючи виробництвом та логістикою [7].

Основні принципи функціонування IoT пристроїв включають збір, передачу, обробку, зберігання даних, управління пристроями та взаємодію з іншими пристроями. Ці принципи дозволяють IoT-пристроям збирати дані з датчиків, передавати їх через бездротові мережі, обробляти, зберігати, керувати віддалено та взаємодіяти з іншими пристроями для реалізації різноманітних функцій та завдань.

1.2 Архітектурні особливості систем IoT

У сучасному технологічному середовищі Інтернет речей (IoT) відіграє важливу роль, забезпечуючи підключеність та обмін даними між пристроями. Цей розділ присвячений розгляду архітектурних аспектів систем IoT, які визначають їх функціональність та ефективність. Детальне розглядання таких аспектів, як коннектівність, масштабованість, безпека та обробка даних, допоможе виявити ключові аспекти роботи та впровадження систем IoT в різних сферах діяльності [8].

Архітектурні особливості систем Інтернету речей (IoT) визначаються принципами, за якими організовані та взаємодіють між собою пристрої та компоненти цих систем. Нижче розглянемо детально кожен з аспектів архітектури IoT систем:

– коннектівність;

Коннектівність - це ключовий аспект архітектури Інтернету речей (IoT), що визначає можливість пристроїв підключатися до мережі Інтернет і обмінюватися даними. Цей принцип передбачає розгалужену мережу зв'язку, яка може включати бездротові технології, такі як Wi-Fi, Bluetooth, Zigbee, LoRaWAN, а також провідні мережі, наприклад, Ethernet.

Бездротові технології надають гнучкість і можливість підключення до мережі з будь-якого місця, що особливо важливо для мобільних пристроїв та розподілених систем. Дротові технології, з іншого боку, забезпечують стабільний та надійний зв'язок з високою швидкістю передачі даних.

Для забезпечення високої доступності та надійності зв'язку, системи IoT можуть використовувати резервні канали зв'язку та механізми автоматичного відновлення з'єднання. Це дозволяє підтримувати безперервну роботу системи навіть у випадку відключень чи відмов [9].

Узагальнюючи, коннектівність у системах IoT означає створення мережевої інфраструктури, яка забезпечує ефективний обмін даними між пристроями, що дозволяє їм працювати як єдиний розумний вузол у великих розподілених системах.

– масштабованість;

Масштабованість є ключовою характеристикою архітектури IoT, оскільки системи IoT зазвичай мають велику кількість підключених пристроїв та обробляють великі обсяги даних. Для успішного функціонування великих мереж IoT потрібна здатність масштабуватися зростанням обсягу даних та кількості підключених пристроїв.

Масштабованість в архітектурі IoT означає, що система може ефективно пристосовуватися до зростання обсягу роботи, не втрачаючи продуктивності. Це може бути досягнуто за допомогою ряду стратегій та підходів, таких як горизонтальне та вертикальне масштабування.

Горизонтальне масштабування полягає в додаванні нових пристроїв або вузлів до мережі для розподілу навантаження та підвищення продуктивності. Цей підхід дозволяє розширювати мережу без змін усередині окремих пристроїв.

Вертикальне масштабування означає покращення функціональних характеристик окремих пристроїв для забезпечення кращої продуктивності. Це може включати в себе збільшення обсягу пам'яті, потужності обробки або пропускної здатності мережі.

Масштабованість також передбачає ефективне керування мережею, оптимізацію пропускної здатності та обробку даних в реальному часі. Це дозволяє системі IoT ефективно працювати навіть при великому навантаженні та забезпечує надійну та швидку реакцію на події в мережі.

– розподіленість;

Розподіленість є важливою характеристикою архітектури IoT, оскільки системи IoT зазвичай мають велику кількість підключених пристроїв та обробляють великі обсяги даних. Це означає, що функції та завдання системи можуть бути розподілені між різними пристроями та серверами для забезпечення оптимальної продуктивності та надійності системи в цілому.

Розподіленість в архітектурі IoT передбачає, що різні завдання та функції можуть бути розподілені між пристроями та серверами, що складають систему. Наприклад, деякі пристрої можуть бути відповідальні за збір та передачу даних, в той час як інші можуть бути призначені для обробки цих даних та прийняття рішень на основі них. Це дозволяє ефективно розподіляти робоче навантаження та забезпечувати оптимальне використання ресурсів.

Окрім того, розподілена архітектура дозволяє забезпечити високу доступність та надійність системи. Навіть якщо один з пристроїв вийде з ладу, інші можуть продовжувати працювати, запобігаючи втраті функціональності системи в цілому. Це робить систему менш вразливою до відмов та забезпечує безперервну роботу [10].

Отже, розподіленість є ключовою характеристикою архітектури IoT, яка дозволяє забезпечити ефективність, доступність та надійність системи в умовах великої кількості підключених пристроїв та обробки великого обсягу даних.

– безпека;

Безпека є однією з найважливіших аспектів архітектури IoT, оскільки дані, які обробляються в таких системах, можуть бути конфіденційними та чутливими. Архітектура IoT повинна передбачати механізми для забезпечення безпеки, щоб запобігти несанкціонованому доступу, зламу та витоку даних.

Одним з основних аспектів безпеки є захист від несанкціонованого доступу до системи та її даних. Це може включати в себе різноманітні механізми аутентифікації, такі як паролі, біометричні дані або використання двофакторної аутентифікації, для перевірки ідентичності користувачів та пристроїв, які намагаються отримати доступ до системи.

Ще однією важливою складовою безпеки є шифрування даних. Всі дані, що передаються між пристроями та серверами, а також зберігаються на них, повинні бути зашифровані для захисту від несанкціонованого доступу. Застосування сучасних шифрувальних алгоритмів дозволяє забезпечити високий рівень конфіденційності даних [11].

Крім того, архітектура IoT повинна передбачати механізми для виявлення та запобігання кібератак. Це може включати в себе моніторинг мережі на наявність аномальної активності, виявлення та блокування шкідливого програмного забезпечення, а також регулярне оновлення програмного забезпечення та патчів для виправлення виявлених уразливостей.

Отже, безпека є невід'ємною частиною архітектури IoT і передбачає впровадження комплексу заходів, спрямованих на захист конфіденційності, цілісності та доступності даних та системи в цілому.

– інтеграція та інтероперабельність;

Інтеграція та інтероперабельність є ключовими аспектами архітектури систем IoT, оскільки вони вимагають взаємодії між різними типами пристроїв та компонентів. Ці пристрої можуть використовувати різні технології, стандарти та протоколи зв'язку, що може ускладнити їхню інтеграцію та взаємодію.

Архітектура систем IoT повинна передбачати механізми для ефективної інтеграції різних пристроїв та систем. Це може включати в себе використання

стандартів та протоколів, таких як MQTT, CoAP або HTTP, які дозволяють пристроям спілкуватися між собою та з центральними серверами.

При проектуванні архітектури IoT слід також враховувати питання інтероперабельності, тобто здатність різних пристроїв та систем працювати разом із збереженням сумісності та продуктивності. Для цього можуть використовуватися відкриті стандарти та протоколи, які дозволяють різним виробникам розробляти сумісні пристрої та програмне забезпечення.

Крім того, архітектура IoT повинна підтримувати гнучкість та масштабованість, щоб забезпечити можливість додавання нових пристроїв та компонентів до системи без перебудови всього існуючого інфраструктурного середовища. Це дозволить легко розширювати та модернізувати систему з плином часу.

Отже, інтеграція та інтероперабельність є важливими складовими архітектури IoT, оскільки вони забезпечують здатність різних пристроїв та систем працювати разом ефективно та безперервно.

– аналітика та обробка даних.

Аналітика та обробка даних в системах Інтернету речей (IoT) є однією з ключових складових їх архітектури. Ця компонента забезпечує здатність системи до ефективної обробки великого обсягу зібраних даних для отримання цінної інформації та прийняття відповідних рішень.

Збір даних - першим кроком у процесі аналітики та обробки даних є збір інформації від різних IoT-пристроїв. Ці дані можуть бути отримані від датчиків, сенсорів, різноманітних джерел або взаємодії з користувачами.

Трансформація даних - після збору дані можуть потребувати попередньої обробки для підготовки їх до аналізу. Ця трансформація може включати очищення, агрегацію, фільтрацію або інші операції з даними.

Аналіз даних - наступним етапом є аналіз отриманих даних з метою виявлення корисних зв'язків, закономірностей та відмінностей. Для цього використовуються різноманітні методи аналізу, такі як статистичний аналіз, машинне навчання, штучний інтелект та інші [12].

Виявлення аномалій - системи IoT можуть автоматично виявляти аномалії або відхилення від звичайних патернів, що може вказувати на потенційні проблеми або загрози.

Прийняття рішень - на основі результатів аналізу, система може приймати рішення про подальші дії. Це може включати автоматизовані дії, відправку сповіщень адміністраторам або користувачам, а також інші форми реакції на отриману інформацію.

Виконання заходів - останнім кроком є виконання відповідних дій або надсилання команд до інших пристроїв для реалізації прийнятих рішень та запобігання потенційним проблемам або загрозам [13].

В архітектурних особливостях систем IoT ключовою ролью відіграє коннектівність, розподіленість, безпека, інтеграція та інтероперабельність, аналітика та обробка даних. Ці характеристики забезпечують ефективне функціонування та взаємодію різноманітних пристроїв в мережі IoT, сприяючи збору, обробці та аналізу великого обсягу даних для прийняття важливих рішень та виконання різноманітних завдань.

1.3 Методи збору та обробки даних в реальному часі

В умовах стрімкого розвитку інтернету речей (IoT) методи збору та обробки даних в реальному часі набувають все більшого значення. Системи IoT забезпечують нам можливість спостереження, контролю та взаємодії з різноманітними пристроями та датчиками у реальному часі, що робить їх важливим інструментом в багатьох сферах життя, від медицини та промисловості до побуту та транспорту.

У цьому контексті дослідження методів збору та обробки даних в реальному часі в системах IoT важливе для розуміння та вдосконалення їхньої ефективності та надійності. В даному розділі ми розглянемо різноманітні підходи до збору та обробки даних, включаючи використання датчиків, хмарних технологій, комп'ютерного зору, методів машинного навчання та багато інших.

Методи збору та обробки даних в реальному часі є важливою складовою архітектури систем IoT, оскільки вони дозволяють забезпечити швидкий та ефективний обмін інформацією для прийняття рішень у реальному часі. Розглянемо детальніше ці методи:

– датчики та датчикові мережі;

Датчики та датчикові мережі є основним елементом інфраструктури в системах IoT. Вони забезпечують збір різноманітних даних про навколишнє середовище, що дозволяє системі реагувати на зміни та виконувати визначені функції. Датчики можуть бути встановлені у пристроях, об'єктах або навіть в самій інфраструктурі міста, що дозволяє створювати розгалужені мережі збору даних.

Основна функція датчиків полягає в тому, щоб перетворити фізичні або хімічні параметри навколишнього середовища на електричні сигнали, які можуть бути опрацьовані електронними пристроями. Ці дані можуть охоплювати широкий спектр параметрів, таких як температура, вологість, рівень освітленості, тиск, рух та багато інших.

Датчикові мережі включають в себе велику кількість датчиків, які можуть бути підключені до мікроконтролерів або вбудованих систем. Ці мережі забезпечують збір даних з різних джерел та їх передачу до центрального контролера або хмарного сервера для подальшої обробки. Така архітектура дозволяє створювати розгалужені та розширювані системи збору та обробки даних, які можуть використовуватися в різних сферах, від медицини та промисловості до міського управління та домашнього забезпечення [14].

– хмарні та клаудові рішення;

Хмарні та клаудові рішення є важливою складовою інфраструктури для багатьох систем IoT. Вони дозволяють збирати, зберігати та обробляти великі обсяги даних на віддалених серверах, що забезпечує більшу гнучкість та ефективність у роботі з інформацією.

Однією з переваг хмарних та клаудових рішень є легкий доступ до даних з будь-якого місця, де є Інтернет-з'єднання. Це дозволяє операторам систем IoT моніторити та управляти пристроями з будь-якого місця, що є важливим у випадку великих та розподілених систем.

Крім того, використання хмарних та клаудових рішень дозволяє оптимізувати витрати на обладнання та інфраструктуру, оскільки не потрібно будувати власні серверні мережі для зберігання даних. Також, завдяки масштабованості та гнучкості хмарних платформ, системи IoT можуть легко розширюватися зростанням обсягів даних та функціональних можливостей.

Однак, існують питання безпеки та конфіденційності даних, пов'язані з використанням хмарних та клаудових сервісів. Оскільки дані зберігаються на віддалених серверах, необхідно приділяти особливу увагу заходам безпеки та шифруванню, щоб запобігти несанкціонованому доступу до інформації [15].

– комп'ютерне зорове спостереження та обробка відеоданих;

Комп'ютерне зорове спостереження та обробка відеоданих є важливим аспектом у сучасних системах, які потребують аналізу великої кількості відеоматеріалу в реальному часі. Ці технології застосовуються у різних сферах, таких як системи відеоспостереження, моніторингу транспортного руху, медичні системи та інші.

Основною метою комп'ютерного зору є розпізнавання об'єктів, руху, людей або подій на відео та забезпечення аналізу в реальному часі. Для цього використовуються алгоритми та методи обробки зображень, які дозволяють виявляти та відстежувати об'єкти у відеопотоці.

Одним з прикладів використання комп'ютерного зору є системи відеоспостереження для забезпечення безпеки в громадських місцях, на транспорті або у великих промислових об'єктах. Ці системи можуть автоматично виявляти підозрілу активність, рух транспорту або навіть обличчя людей для подальшого аналізу або реакції.

Крім того, комп'ютерне зорове спостереження може бути використане у медичних системах для аналізу медичних зображень або в системах розпізнавання маркерів у виробництві та робототехніці.

Проте важливо враховувати питання приватності та безпеки даних при використанні таких систем, а також оптимізувати алгоритми обробки відеоданих для забезпечення ефективної роботи в реальному часі.

– технологія обробки стрічок подій (CEP);

Технологія обробки стрічок подій (CEP) - це потужний інструмент для аналізу та обробки великих потоків даних в реальному часі. Вона дозволяє виявляти певні події або шаблони в потоках даних, реагуючи на них негайно та ефективно.

У системах IoT CEP використовується для виявлення аномальної поведінки, відстеження ключових подій та вирішення важливих завдань в реальному часі. Наприклад, CEP може бути використана для моніторингу стану обладнання або інфраструктури, виявлення аварійних ситуацій та надання оперативних рекомендацій для їх вирішення.

Основні переваги технології CEP полягають у її швидкості, масштабованості та здатності адаптуватися до змін у вхідних даних та вимогах системи. Вона дозволяє аналізувати великі обсяги даних в реальному часі та реагувати на них без затримок, що робить її незамінним інструментом для розумних систем IoT.

– методи машинного навчання та штучного інтелекту;

Методи машинного навчання та штучного інтелекту відіграють ключову роль у сучасних системах Інтернету речей (IoT). Ці методи дозволяють системам автоматично виконувати аналіз великих обсягів даних в реальному часі та приймати інтелектуальні рішення без прямого втручання людини [16].

У контексті IoT методи машинного навчання можуть бути використані для різних цілей, таких як передбачення виникнення проблем або аномальної поведінки, виявлення закономірностей у великих потоках даних, а також автоматичний аналіз та інтерпретація отриманих результатів. Наприклад,

алгоритми машинного навчання можуть бути використані для прогнозування витрат енергії, виявлення ознаки небезпеки на виробничому обладнанні або оптимізації роботи системи управління ресурсами.

Штучний інтелект, у свою чергу, дозволяє системам IoT автоматично адаптуватися до змінних умов навколишнього середовища, вчиться на основі даних та досвіду та вдосконалює свої функції з часом. Наприклад, системи зі штучним інтелектом можуть самостійно визначати оптимальні стратегії управління енергоефективністю будівель або прогнозувати зміни погодних умов для оптимального управління сільськогосподарськими процесами [17].

Застосування методів машинного навчання та штучного інтелекту в системах IoT дозволяє підвищити їхню ефективність, точність та здатність адаптуватися до різних сценаріїв використання, що робить їх більш привабливими для широкого спектра застосувань.

– обробка на краю мережі (Edge Computing).

Обробка на краю мережі (Edge Computing) – це стратегія обробки даних, яка передбачає виконання обчислень на самому пристрої або на локальному сервері, що розташований ближче до джерела виникнення цих даних. Цей підхід став дуже популярним у системах IoT, де швидкість обробки та аналізу даних, а також мінімізація затримок є критичними факторами.

Завдяки обробці на краю мережі, дані можуть бути оброблені та аналізовані безпосередньо на пристрої або в локальній мережі, що дозволяє виконувати різноманітні завдання без необхідності передавати великі обсяги даних до центрального обчислювального центру або до хмарного сервісу. Це сприяє зменшенню навантаження на мережу та оптимізації використання ресурсів.

Прийом Edge Computing також забезпечує покращену конфіденційність даних, оскільки більша частина обробки відбувається на місці їхнього збору, що ускладнює несанкціонований доступ до них. Крім того, цей підхід дозволяє підвищити реактивність системи, оскільки рішення можуть бути прийняті на місці без затримок через мережу.

Узагальнюючи, обробка на краю мережі є ефективним та перспективним підходом у сфері IoT, який дозволяє забезпечити швидку та надійну обробку даних, зменшення затримок та покращення конфіденційності, що відповідає вимогам сучасних динамічних та розподілених систем [18].

Використання датчиків, хмарних рішень, комп'ютерного зору, технологій обробки стрічок подій, методів машинного навчання та обробки на краю мережі виявляється як ключові методи реалізації IoT-систем. Кожен з цих методів має свої переваги та обмеження, але разом вони дозволяють ефективно збирати, передавати, обробляти та аналізувати великі обсяги даних в реальному часі. Використання цих методів дозволяє побудувати потужні та надійні системи IoT, які забезпечують швидку реакцію на зміни у навколишньому середовищі та дозволяють виконувати різноманітні завдання в реальному часі.

1.4 Методи шифрування даних в системах IoT

У сучасному світі Інтернету речей (IoT) важливо забезпечити безпеку передачі та збереження даних, оскільки вони збираються, обробляються та обмінюються між різними пристроями та системами. Методи шифрування даних в системах IoT є ключовим елементом для захисту конфіденційності, цілісності та доступності цих даних. У цьому підрозділі розглянуться різні методи шифрування, їхнє застосування в контексті Інтернету речей та їхня важливість для забезпечення безпеки в системах IoT. Вивчення цих методів дозволить зрозуміти, як забезпечити захист важливих даних в сучасних мережах IoT.

Методи шифрування даних в системах Інтернету речей (IoT) є критично важливими для забезпечення конфіденційності, цілісності та доступності даних у вузьких мережах з великою кількістю пристроїв. Основні методи шифрування включають в себе:

– симетричне шифрування;

Симетричне шифрування є одним із найбільш поширених методів шифрування в інформаційній безпеці, включаючи системи Інтернету речей (IoT).

Цей метод використовує один і той же ключ для як шифрування, так і розшифрування даних.

Існують кілька важливих характеристик симетричного шифрування:

Ефективність та швидкість: Оскільки для шифрування та розшифрування використовується один ключ, симетричне шифрування зазвичай працює швидше, ніж асиметричне шифрування [19].

Вимога до безпеки ключа: Однак симетричне шифрування вимагає безпечного обміну ключів між сторонами, щоб забезпечити конфіденційність даних. Це може бути проблемою в системах IoT, де важко забезпечити безпеку обміну ключів.

Приклади алгоритмів: Один із найпоширеніших алгоритмів симетричного шифрування - Advanced Encryption Standard (AES). AES використовується для захисту конфіденційності даних у багатьох системах IoT через свою швидкість та надійність.

У контексті систем IoT, симетричне шифрування застосовується для захисту комунікації між пристроями, а також для збереження конфіденційної інформації на пристроях та в хмарних сервісах. Однак, важливо враховувати безпеку обміну ключів та забезпечити їх надійне зберігання, щоб уникнути можливих загроз безпеці в мережі IoT.

– асиметричне шифрування;

Асиметричне шифрування, або криптографія з відкритим ключем, є ще одним важливим методом шифрування в інформаційній безпеці, який широко використовується в системах IoT. У цьому методі використовуються два ключі: публічний та приватний.

Основні характеристики асиметричного шифрування:

– два ключі: У асиметричному шифруванні використовується пара ключів: публічний і приватний. Публічний ключ використовується для шифрування повідомлення, тоді як приватний ключ використовується для розшифрування;

– безпека обміну ключів: Один із головних переваг асиметричного шифрування - відсутність необхідності в безпечному обміні ключів між сторонами, оскільки публічний ключ може бути розповсюджений відкрито;

– приклади алгоритмів: RSA (Rivest–Shamir–Adleman) та ECC (еліптична крива) є одними з найбільш відомих та широко використовуваних алгоритмів асиметричного шифрування.

У системах IoT асиметричне шифрування може застосовуватися для забезпечення безпеки комунікації між пристроями та серверами, аутентифікації користувачів та захисту конфіденційної інформації. Використання асиметричного шифрування спрощує управління ключами та забезпечує високий рівень безпеки в системах IoT [20].

– хеш-функції;

Хеш-функції - це важливий інструмент для захисту даних в системах Інтернету речей (IoT). Вони використовуються для генерації унікального хешу з вихідних даних, який може бути використаний для перевірки цілісності та безпеки даних. Одна з ключових особливостей хеш-функцій - це їх невідбратність, що означає, що неможливо відновити вихідне повідомлення з хешу. Типові алгоритми хеш-функцій включають MD5 та SHA, які використовуються для генерації хешу фіксованої довжини з будь-якого вхідного повідомлення. У системах IoT хеш-функції застосовуються для забезпечення цілісності даних, аутентифікації користувачів та пристроїв, а також для захисту від несанкціонованого доступу та забезпечення безпеки даних [21].

– цифрові підписи;

Цифрові підписи - це ключовий елемент у забезпеченні безпеки даних в системах IoT. Вони використовуються для підтвердження автентичності повідомлення та забезпечення його цілісності під час передачі. Процес створення цифрового підпису включає підписання повідомлення за допомогою приватного ключа, який доступний тільки відправнику. При отриманні повідомлення його підпис перевіряється за допомогою відповідного публічного ключа, який

доступний для всіх. Якщо підпис вірний, це доводить автентичність джерела повідомлення та його недоторканість під час транспортування. Це є важливим механізмом для забезпечення безпеки даних у вимірах IoT, де велика кількість пристроїв збирає та обробляє чутливу інформацію.

– протоколи обміну ключами;

Протоколи обміну ключами відіграють важливу роль у забезпеченні безпеки даних у системах IoT. Ці протоколи дозволяють сторонам безпечно обмінюватися секретними ключами, які використовуються для шифрування та розшифрування даних. Наприклад, протокол TLS (Transport Layer Security) широко використовується для створення захищеного каналу зв'язку між клієнтом та сервером. Під час встановлення з'єднання TLS, клієнт та сервер взаємодіють для обміну та підтвердження секретних ключів, що дозволяє їм створити захищений тунель для передачі даних. Це є надійним механізмом забезпечення конфіденційності та цілісності даних у системах IoT, де забезпечення безпеки є важливою вимогою [22].

– квантова криптографія.

Квантова криптографія представляє собою перспективний напрямок в області шифрування даних, що використовує принципи квантової механіки для забезпечення безпеки комунікацій. Цей підхід вважається найбільш надійним, оскільки базується на фізичних законах, таких як непорушність квантових станів. Використання квантових властивостей, таких як квантова переплетеність та квантова неозначеність, дозволяє створювати непроникні для зламу системи шифрування. Одним із прикладів застосування квантової криптографії є квантова ключова дистрибуція, яка забезпечує безпечний обмін ключами, що неможливо зламати за допомогою класичних алгоритмів. Ця технологія відкриває нові можливості для створення надійних та стійких систем захисту інформації, що є особливо актуальним у зв'язку зі зростанням кількості кіберзагроз у сучасному світі.

Методи шифрування даних в системах IoT відіграють важливу роль у забезпеченні конфіденційності, цілісності та доступності інформації. Вони включають в себе симетричне та асиметричне шифрування, використання хеш-функцій, цифрові підписи, протоколи обміну ключами та навіть квантову криптографію. Кожен з цих методів має свої переваги та обмеження, але разом вони утворюють комплексну систему захисту, що дозволяє забезпечити безпеку передачі та збереження даних у мережах Інтернету речей.

1.5 Аналіз основних протоколів обміну даними в IoT системах та їх використання для захисту інформації

Протоколи обміну даними є ключовим компонентом IoT-систем, адже вони забезпечують зв'язок між різними компонентами та передачу даних. Існує багато різних протоколів, кожен з яких має свої особливості та переваги. В цьому розділі ми розглянемо основні протоколи, що використовуються в IoT-системах, та проаналізуємо їх можливості з точки зору захисту інформації.

Протоколи обміну даними в IoT системах можна порівнювати за різними критеріями, зокрема:

– пропускна здатність: Це кількість даних, яку протокол може передати за певний час. Деякі протоколи, такі як MQTT та CoAP, мають низьку пропускну здатність, але вони ефективні для пристроїв з обмеженими ресурсами. У той час як HTTP може мати вищу пропускну здатність, але вимагає більше ресурсів;

– легкість використання: Деякі протоколи можуть бути більш простими у використанні та розумінні, що сприяє їхньому широкому застосуванню. Наприклад, HTTP вже є загальноприйнятим протоколом, тому його використання може бути більш простим для розробників;

– безпека: Важливий аспект для IoT систем. Протоколи, які надають можливості шифрування даних, аутентифікації та захисту від кібератак, можуть бути більш привабливими для застосування в критичних системах;

– підтримка пристроїв з обмеженими ресурсами: Оскільки багато IoT пристроїв мають обмежені ресурси, такі як обсяг пам'яті та потужність обробки, важливо, щоб протоколи були оптимізовані для роботи з такими пристроями;

– надійність: Деякі протоколи можуть забезпечувати надійну передачу даних, яка гарантує доставку даних навіть у випадку втрати зв'язку;

– сумісність із стандартами: Протоколи, які добре інтегруються з існуючими стандартами та екосистемами, можуть бути більш привабливими для розробників та виробників.

Порівняння протоколів за цими критеріями допоможе визначити найбільш підходящий протокол для конкретного випадку застосування в IoT системі.

Аналіз основних протоколів обміну даними в системах Інтернету речей (IoT) є ключовим аспектом забезпечення безпеки та ефективності таких систем. Деякі з найпоширеніших протоколів включають MQTT (Message Queuing Telemetry Transport), CoAP (Constrained Application Protocol), HTTP (Hypertext Transfer Protocol), та AMQP (Advanced Message Queuing Protocol) [23].

Протокол MQTT (Message Queuing Telemetry Transport) є легковаговим, відкритим стандартом для обміну повідомленнями між пристроями в мережі IoT. Ось детальний опис його особливостей:

– легкість використання: MQTT був розроблений для того, щоб бути простим у використанні та інтеграції. Він використовує простий модель публікації/підписки (publish/subscribe), де пристрої публікують повідомлення на певні теми, а інші підписані пристрої отримують ці повідомлення;

– низька пропускна здатність: MQTT оптимізований для праці з мережами IoT, де пристрої можуть мати обмежену пропускну здатність та ресурси. Він використовує малу кількість протокольних повідомлень, що знижує навантаження на мережу;

– підтримка рівня QoS (Quality of Service): MQTT підтримує три рівні QoS, які дозволяють регулювати надійність доставки повідомлень. Це дозволяє вибирати рівень надійності передачі в залежності від потреб застосунку;

– шифрування та аутентифікація: MQTT може бути захищений за допомогою SSL/TLS для шифрування даних та забезпечення аутентифікації пристроїв. Це робить його безпечним для використання в критичних застосунках, де важлива конфіденційність та цілісність даних;

– підтримка різних платформ: MQTT має широку підтримку на різних платформах та мовах програмування, що дозволяє легко інтегрувати його в різноманітні системи;

– автономність: MQTT може працювати у режимі "останньє відоме значення", що дозволяє пристроям отримувати останні значення під час відновлення з'єднання після втрати зв'язку.

Ці характеристики роблять MQTT популярним протоколом для використання в IoT системах, особливо для пристроїв з обмеженими ресурсами та вимогами до безпеки.

CoAP (Constrained Application Protocol) є протоколом, призначеним для обміну даними між пристроями IoT з обмеженими ресурсами, такими як датчики та актуатори. Ось детальний опис його особливостей:

– оптимізація для обмежених пристроїв: CoAP розроблений з урахуванням обмеженого обсягу пам'яті та оброблювальної потужності, що характерно для багатьох пристроїв IoT. Він використовує мінімальну кількість ресурсів для обміну даними, що робить його ідеальним для використання на пристроях з обмеженими можливостями;

– підтримка DTLS: CoAP може бути захищений за допомогою DTLS (Datagram Transport Layer Security), який забезпечує шифрування даних та аутентифікацію пристроїв. Це забезпечує безпечний обмін даними навіть у вузьких мережних середовищах;

– простий використання: CoAP використовує архітектуру RESTful, що робить його схожим до HTTP, але оптимізованим для мережних пристроїв. Це дозволяє легко інтегрувати його в існуючі системи та використовувати звичайні HTTP-подібні запити та відповіді для обміну даними;

– підтримка методів обміну даними: CoAP підтримує методи обміну даними, такі як GET, POST, PUT та DELETE, що дозволяє виконувати різноманітні операції над ресурсами пристроїв;

– кешування та оптимізація трафіку: CoAP має можливість кешування, що дозволяє пристроям зберігати результати запитів та використовувати їх для майбутніх запитів. Це дозволяє зменшити навантаження на мережу та забезпечити більш ефективне використання ресурсів;

– підтримка UDP та SMS: CoAP може працювати як над транспортним протоколом UDP, так і над SMS, що робить його ідеальним для використання в різних мережних середовищах.

Ці характеристики роблять CoAP популярним протоколом для використання в обмежених пристроях IoT, де обсяг ресурсів обмежений, а потреба у безпеці та ефективності велика.

HTTP (Hypertext Transfer Protocol) є протоколом, який використовується для передачі даних в мережі Інтернет. Ось детальний опис його особливостей:

– клієнт-Серверна модель: HTTP базується на моделі клієнт-сервер, де клієнт (наприклад, веб-браузер або IoT-пристрій) відправляє запити на сервер для отримання веб-сторінок або інших ресурсів;

– запити та відповіді: Протокол використовує методи запитів, такі як GET, POST, PUT, DELETE, для взаємодії з сервером. Сервер обробляє ці запити та надсилає відповіді, які містять статус виконання запиту та потрібні дані;

– становість: HTTP є становим протоколом, що означає, що він не зберігає інформацію про попередні запити клієнта. Кожен запит обробляється незалежно, без попередніх даних про сеанс;

– протокол HTTPS: HTTPS (HTTP over SSL/TLS) є захищеною версією протоколу HTTP, яка використовує шифрування за допомогою SSL/TLS для забезпечення безпеки передачі даних. Це робить протокол HTTPS відповідним для використання в системах IoT, де важливо захистити конфіденційні дані від несанкціонованого доступу;

– завантаження сторінок та ресурсів: HTTP використовується для завантаження веб-сторінок, графіки, аудіо- та відеофайлів, а також для інших операцій, таких як надсилання форм та отримання даних;

– RESTful архітектура: HTTP є основою для архітектури RESTful, що дозволяє створювати зручні та ефективні веб-сервіси для обміну даними між клієнтом та сервером в системах IoT.

У контексті IoT, HTTP може бути використаний для взаємодії між пристроями IoT та хмаровими сервісами, а протокол HTTPS забезпечує захищену передачу даних, що є важливим у зв'язку з конфіденційністю та цілісністю інформації, переданої пристроями.

AMQP (Advanced Message Queuing Protocol) - це протокол для передачі повідомлень між програмними компонентами в мережах, що мають складну топологію. Ось детальний опис його особливостей:

– архітектура повідомлень: AMQP базується на асинхронному обміні повідомленнями між клієнтами та брокерами (міжкомпонентними посередниками). Клієнти надсилають повідомлення брокеру, який потім маршрутизує їх до призначених отримувачів;

– надійність: AMQP забезпечує надійну доставку повідомлень, використовуючи підтвердження (ACKs) та механізми перевірки доставки (delivery confirmations). Це дозволяє гарантувати, що повідомлення будуть доставлені та оброблені, навіть у випадку збоїв або відмов у системі;

– шифрування і аутентифікація: Використання TLS (Transport Layer Security) в AMQP забезпечує шифрування та аутентифікацію даних, що передаються по мережі. Це робить протокол AMQP безпечним для використання в системах IoT, де важливо захистити конфіденційні дані від несанкціонованого доступу;

– гнучкість і масштабованість: AMQP підтримує гнучку та масштабовану архітектуру, що дозволяє побудувати складні топології мереж з великою кількістю компонентів. Він також підтримує різні типи обміну повідомленнями, такі як прямий, тематичний та вентильний, для ефективної маршрутизації повідомлень;

– стандартизація: AMQP є відкритим стандартом, що розробляється й підтримується організацією OASIS (Organization for the Advancement of Structured Information Standards). Це забезпечує сумісність та інтероперабельність між різними реалізаціями протоколу.

У контексті IoT, AMQP може бути використаний для надійного обміну повідомленнями між пристроями IoT та хмаровими серверами, забезпечуючи шифрування та аутентифікацію даних для захисту конфіденційності та цілісності інформації.

Оскільки вибір протоколу обміну даними в системах IoT може бути критичним для успішності проекту, порівняння протоколів за різними критеріями може допомогти зробити більш обгрунтований вибір. В таблиці 1.1 наведено порівняльний аналіз протоколів MQTT, CoAP, HTTP та AMQP, що допомагає краще розуміти їхні особливості та відповідність конкретним потребам проекту [24].

Таблиця 1.1 – Порівняльний аналіз протоколів обміну даними в IoT системах

Критерій	MQTT	CoAP	HTTP	AMQP
Пропускна здатність	Низька	Низька	Висока	Середня
Легкість використання	Висока	Висока	Середня	Низька
Безпека	Середня	Середня	Низька	Висока
Підтримка пристроїв з обмеженими ресурсами	Висока	Висока	Низька	Середня
Надійність	Висока	Висока	Низька	Середня

Ця таблиця порівнює протоколи MQTT, CoAP, HTTP та AMQP за різними критеріями, такими як пропускна здатність, легкість використання, безпека,

підтримка пристроїв з обмеженими ресурсами, надійність та сумісність із стандартами.

Загальний аналіз таблиці показує, що кожен з протоколів - MQTT, CoAP, HTTP та AMQP - має свої унікальні характеристики, які можуть бути важливими для різних випадків застосування в системах IoT. Наприклад, якщо вам потрібен протокол з низькою пропускною здатністю, але відмінною ефективністю ресурсів, MQTT може бути вибором, оскільки він спроектований саме для цього типу пристроїв. З іншого боку, якщо важлива легкість використання та висока пропускна здатність, HTTP може виявитися кращим вибором.

Важливо також враховувати, що багато протоколів, зокрема MQTT, CoAP та AMQP, надають можливості шифрування даних та аутентифікації, що робить їх більш привабливими для використання в критичних системах, де важлива безпека. Крім того, протоколи HTTP та AMQP можуть бути більш надійними в передачі даних, що є важливим аспектом для деяких застосувань IoT [25].

У кінцевому підсумку, вибір конкретного протоколу обміну даними в IoT системах повинен враховувати комплексні потреби проекту, включаючи обсяг та характеристики даних, потужність пристроїв, рівень безпеки та надійності, а також сумісність із стандартами та легкість використання.

1.6 Висновки та постановка задачі

В рамках даної роботи проведено дослідження та аналіз теоретичних основ створення систем, спрямованих на підвищення захищеності приміщень за допомогою автономних IoT пристроїв збору інформації та обміну зашифрованими даними в реальному часі. Вивчення основних принципів функціонування IoT пристроїв, їхніх архітектурних особливостей, методів збору, обробки та шифрування даних було спрямоване на розробку ефективних та безпечних систем безпеки приміщень.

В результаті аналізу встановлено, що використання IoT пристроїв може значно підвищити рівень захисту приміщень, забезпечуючи постійний моніторинг

та обмін інформацією в реальному часі. Для забезпечення конфіденційності та цілісності даних були розглянуті та проаналізовані різноманітні методи шифрування, включаючи симетричне та асиметричне шифрування, хеш-функції, цифрові підписи, квантову криптографію та інші.

Також було проведено детальний аналіз основних протоколів обміну даними в IoT системах, таких як MQTT, CoAP, HTTP та AMQP, зокрема з точки зору їхнього впливу на захист інформації та ефективність взаємодії з пристроями.

Отже, на основі проведеного дослідження та аналізу було сформульовано ряд завдань для подальшої розробки:

– проаналізувати особливості захисту приміщень за допомогою IoT пристроїв

– здійснити проектування схеми роботи пристрою захисту приміщення

– здійснити проектування системи взаємодії між IoT пристроїями

– Здійснити розробку схеми обміну зашифрованими даними у реальному часі між IoT пристроїями.

– здійснити програмну реалізацію

2 ПРОЕКТУВАННЯ СИСТЕМИ ПІДВИЩЕННЯ ЗАХИЩЕНОСТІ ПРИМІЩЕНЬ ВДОСКОНАЛЕНОЮ СИСТЕМОЮ НА ОСНОВІ АВТОНОМНИХ ІОТ ПРИСТРОЇВ ЗБОРУ ІНФОРМАЦІЇ ТА ОБМІНУ ЗАШИФРОВАНИМИ ДАНИМИ У РЕАЛЬНОМУ ЧАСІ

В даному розділі розділі буде здійснено розгляд особливостей захисту приміщень за допомогою IoT пристроїв, буде здійснено проектування схеми роботи пристрою захисту приміщення, також буде здійснена розробка схеми обміну зашифрованими даними у реальному часі між IoT пристроїями.

2.1 Особливості захисту приміщень за допомогою IoT пристроїв.

Особливості захисту приміщень за допомогою IoT пристроїв базуються на інтеграції різноманітних технологій та методик, що спрямовані на підвищення безпеки та захищеності приміщень в реальному часі. Розробка системи на основі автономних IoT пристроїв для збору інформації та обміну зашифрованими даними вимагає глибокого розуміння технологічних, організаційних та безпекових аспектів.

Розподілена архітектура системи безпеки на основі IoT (Інтернет речей) є ключовою особливістю, що забезпечує високу гнучкість, масштабованість та надійність у захисті приміщень. Вона передбачає створення мережі з взаємопов’язаних, але автономно діючих пристроїв, які збирають та обмінюються даними. Така структура дозволяє системі захисту бути ефективною навіть у випадку збоїв частини компонентів, а також забезпечує гнучке масштабування. Ось декілька ключових аспектів розподіленої архітектури:

Система інтегрує різні типи IoT пристроїв, включаючи датчики руху, камери спостереження, датчики відкриття дверей, димові датчики, та інші сенсори. Ці пристрої розміщуються в стратегічних локаціях приміщення для моніторингу різних параметрів, що важливі для забезпечення безпеки [26].

Розподіленість архітектури дозволяє системі продовжувати працювати навіть при виході з ладу окремих компонентів. Дані з різних джерел можуть бути

скомбіновані для покращення точності ідентифікації загроз та підвищення надійності системи в цілому.

Систему можна легко адаптувати під змінні потреби, додавати нові IoT пристрої або оновлювати існуючі без необхідності повної перебудови системи. Це забезпечує ефективне масштабування системи відповідно до розширення або модифікації приміщень.

Кожен IoT пристрій в системі може автономно обробляти дані та реагувати на зміни у середовищі без постійного зв’язку з центральним сервером. Це знижує залежність від централізованої інфраструктури та покращує загальну відмовостійкість системи.

Обробка даних може здійснюватись як локально на окремих IoT пристроях, так і централізовано. Це дозволяє ефективно розподіляти обчислювальні ресурси, зменшуючи навантаження на мережу та підвищуючи швидкість реакції системи на загрози.

Розподілена архітектура вимагає впровадження комплексних заходів безпеки на кожному рівні: від захисту передачі даних між IoT пристроями до забезпечення безпеки зберігання даних на серверах. Це включає застосування шифрування, аутентифікації та інших технологій для забезпечення конфіденційності та цілісності даних.

Розподілена архітектура системи безпеки на основі IoT надає значні переваги у термінах гнучкості, надійності та ефективності. Її головна сила полягає у здатності адаптуватися до різноманітних умов та потреб безпеки, забезпечуючи високий рівень захисту приміщень [27].

Автономність і енергоефективність є ключовими характеристиками IoT (Інтернет речей) пристроїв, які використовуються в системах безпеки для приміщень. Ці особливості не тільки сприяють надійності та довготривалій роботі системи, але й забезпечують її економічну ефективність та широке впровадження. Ось детальніше про кожну з цих характеристик:

Незалежна робота: IoT пристрої здатні функціонувати незалежно, без необхідності постійного підключення до зовнішніх джерел енергії або

центрального сервера. Це досягається завдяки вбудованим акумуляторам або енергонезалежним джерелам живлення.

Мінімізація залежності від інфраструктури: Автономність дозволяє розміщувати IoT пристрої в труднодоступних місцях або в областях без стабільного доступу до електромережі, розширюючи можливості моніторингу безпеки.

Оптимізоване споживання енергії: Сучасні IoT пристрої для систем безпеки розроблені з урахуванням потреби в мінімізації споживання енергії. Використання енергоефективних компонентів та алгоритмів сплячого режиму забезпечує тривалу роботу від акумулятора.

Енергонезалежні джерела живлення: Деякі системи можуть інтегрувати використання альтернативних джерел енергії, таких як сонячні панелі або генератори кінетичної енергії, для подовження часу автономної роботи або навіть повної незалежності від традиційних джерел живлення.

Збільшення тривалості роботи: Автономні і енергоефективні IoT пристрої можуть працювати місяцями або навіть роками без необхідності заміни батарей або підзарядки, зменшуючи витрати на обслуговування та експлуатацію.

Гнучкість установки та розміщення: Висока автономність і енергоефективність дозволяють розміщувати пристрої в оптимальних точках для моніторингу, незалежно від доступності електромереж.

Надійність системи: Системи, що складаються з автономних і енергоефективних пристроїв, забезпечують неперервну роботу навіть у випадку відключення електроенергії або інших збоїв у інфраструктурі.

2.2 Проектування схеми роботи пристрою захисту приміщення.

Проектування схеми роботи пристрою захисту приміщення на основі IoT включає кілька ключових етапів та компонентів, які забезпечують його ефективність, надійність та автономність.

Визначення функціональних вимог для пристрою захисту приміщення, зосередженого на відслідковуванні вторгнень, вимагає глибокого аналізу

потенційних ризиків та способів їх виявлення. Така система має включати різні типи сенсорів та алгоритми обробки даних, щоб ефективно ідентифікувати спроби несанкціонованого доступу. Ось детальний опис вимог:

Використання інфрачервоних або мікрохвильових датчиків руху для виявлення несанкціонованої активності у приміщенні. Сенсори повинні бути налаштовані таким чином, щоб знизити ймовірність хибних спрацьовувань (наприклад, від домашніх тварин).

Можливість поділу приміщення на зони з окремим моніторингом, дозволяючи активувати систему частково або повністю залежно від потреби.

Датчики, здатні виявляти звуки розбитого скла або інші підозрілі звуки, які можуть вказувати на спробу вторгнення.

Можливість керувати всіма компонентами системи через єдиний інтерфейс, включаючи мобільні додатки або веб-платформи.

Сповіщення: Автоматичне сповіщення власника приміщення або служби охорони про будь-які інциденти або спроби вторгнення через SMS, електронну пошту або push-повідомлення [29].

Безперебійна робота: Система повинна мати вбудовані механізми для забезпечення роботи під час відключення електроенергії, включаючи акумуляторне живлення.

При виборі компонентів для системи захисту приміщення, де основним компонентом є NodeMCU, потрібно звернути увагу на сумісність, ефективність та надійність підключення до інших елементів системи. NodeMCU базується на ESP8266 (або ESP32 для більш потужної версії), який є високопродуктивним мікроконтролером з вбудованим Wi-Fi модулем, що ідеально підходить для IoT проектів. Ось основні категорії компонентів, які слід врахувати: Датчики руху (наприклад, PIR): для виявлення руху в певній зоні. Магнітні контакти: для моніторингу відкриття/закриття дверей та вікон. Акустичні датчики: для виявлення специфічних звуків, як-от розбиття скла. Камери: можуть використовуватись для відеоспостереження; потребують інтеграції через Wi-Fi з можливістю віддаленого доступу.

NodeMCU (ESP8266 або ESP32): в якості основного контролера з Wi-Fi модулем для зв'язку з інтернетом і віддаленим сервером або хмарним сервісом.

Ethernet модуль (додатково): для забезпечення більш стабільного мережевого з'єднання, якщо Wi-Fi з'єднання не є надійним у певному розташуванні.

Акумулятори та UPS: для забезпечення безперебійної роботи при відключенні електроенергії.

Під час вибору компонентів важливо звернути увагу на їх сумісність з NodeMCU, можливості інтеграції між різними елементами системи, а також на вимоги до живлення та зв'язку. Використання NodeMCU як основи для системи безпеки приміщення дозволяє створити гнучке та ефективне рішення з широким спектром можливостей для розширення та адаптації під конкретні потреби.

На основі, вище згаданих компонентів була розроблена схема роботи пристрою захисту приміщення (рис. 2.1)

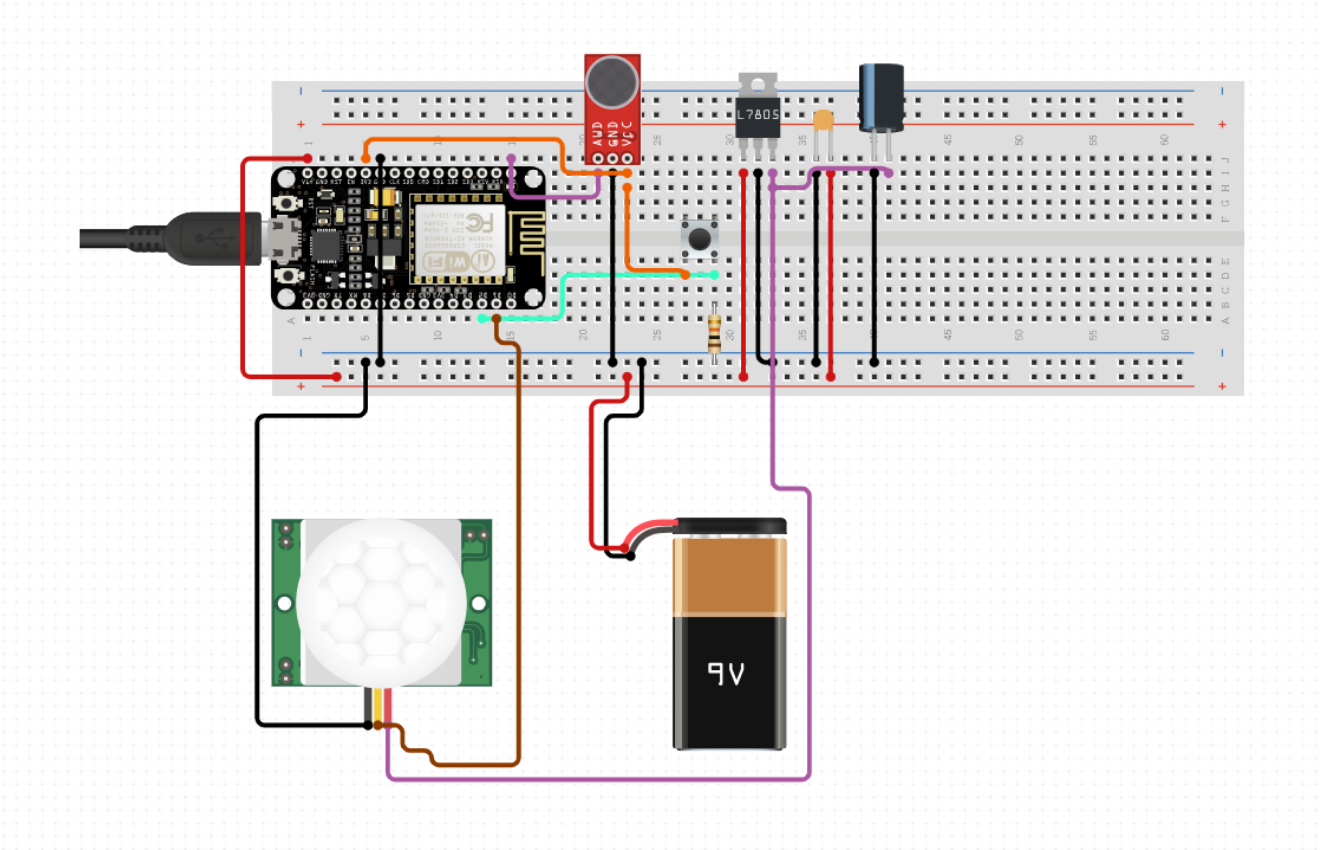


Рисунок 2.1 – Розроблена схема

Система захисту приміщення, заснована на використанні NodeMCU, інфрачервоного датчика руху, мікрофона та безперебійного живлення, має низку

переваг, що робить її ефективним рішенням для забезпечення безпеки. Ці компоненти разом створюють гнучку, інтелектуальну та надійну систему моніторингу. Ось детальніше про ключові переваги такого пристрою:

NodeMCU пропонує легке підключення додаткових сенсорів або виконавчих пристроїв, дозволяючи легко розширювати систему згідно зі специфічними потребами безпеки приміщення.

Програмування NodeMCU через Arduino IDE або інші середовища дозволяє налаштовувати логіку роботи системи, що забезпечує високий рівень адаптивності до різних умов експлуатації [30].

Інфрачервоний датчик руху забезпечує точне виявлення руху в приміщенні, що дозволяє своєчасно реагувати на несанкціоноване вторгнення.

Мікрофон може використовуватися для детекції звуків, які свідчать про присутність або діяльність людей в приміщенні, забезпечуючи додатковий рівень моніторингу.

Система може надсилати сповіщення власнику через інтернет (наприклад, через електронну пошту або мобільний додаток), забезпечуючи оперативне інформування про будь-які інциденти.

Використання безперебійного живлення гарантує неперервну роботу системи навіть при відключенні електроенергії, забезпечуючи постійний захист приміщення.

NodeMCU, завдяки своїй стабільності та надійності як платформи IoT, забезпечує високу продуктивність системи захисту.

Компоненти системи, включаючи NodeMCU та сенсори, є відносно недорогими, що робить це рішення економічно вигідним для широкого кола користувачів.

Самостійне налаштування та обслуговування системи на базі NodeMCU знижує витрати на професійні послуги з інсталяції та налаштування систем безпеки.

Можливість інтеграції з розумним домом та іншими IoT-пристроями відкриває широкі можливості для створення інтелектуальних сценаріїв реагування на різні події (наприклад, автоматичне включення освітлення при виявленні руху).

2.3 Проектування системи взаємодії між IoT пристроїями.

Проектування системи взаємодії між IoT пристроями є ключовим елементом для створення ефективної та надійної системи захисту приміщення. Така система повинна забезпечувати безперервний збір даних, їх аналіз та відповідну реакцію на виявлені події або загрози. Ось основні аспекти, які потрібно врахувати під час проектування.

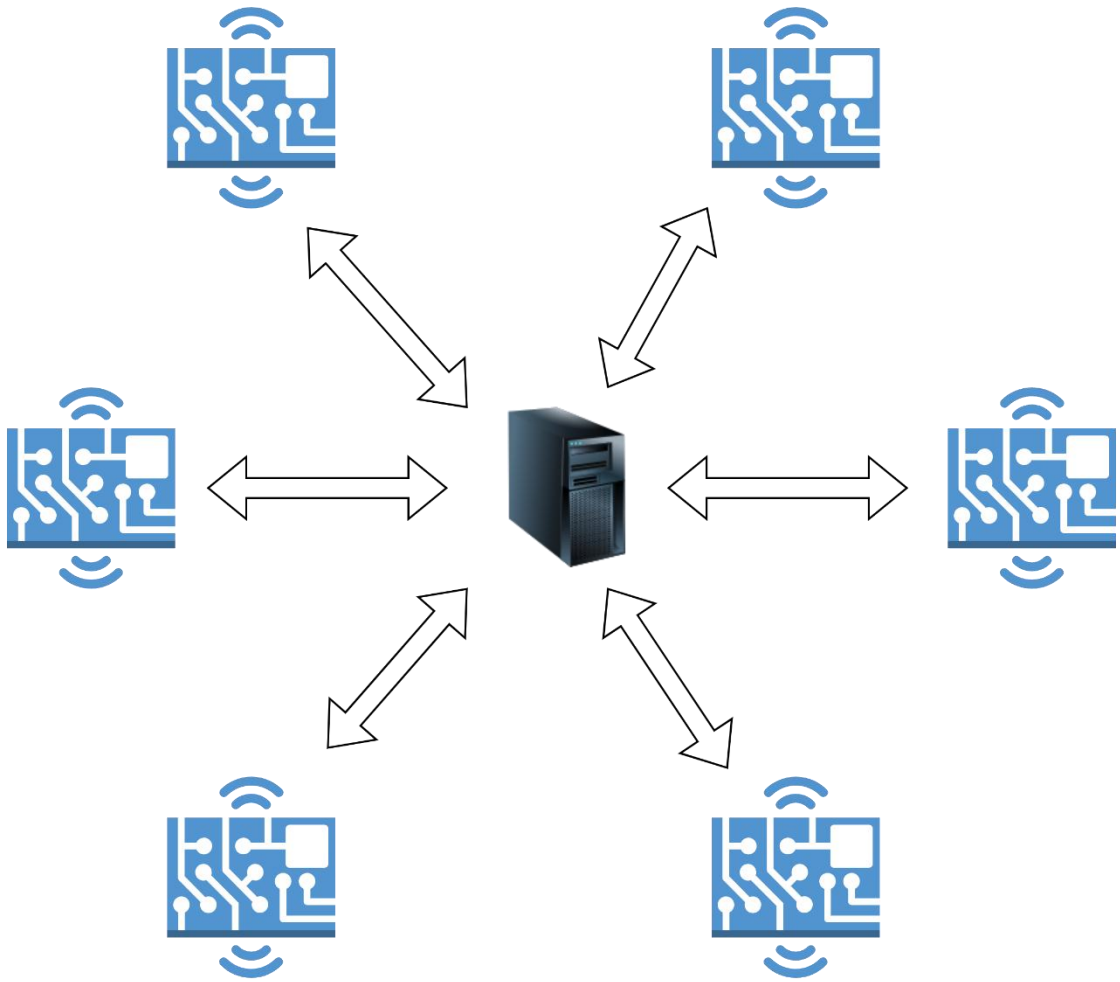


Рисунок 2.2 – Схема взаємодії між сервером та IoT пристроями

WebSocket було обрано як основу для комунікації між пристроями та сервером через його здатність забезпечити двосторонній зв'язок в реальному часі (рис. 2.2). Цей протокол ідеально підходить для задач, де важлива швидка реакція

системи на події від сенсорів або команди користувачів. Вибір мови програмування для сервера та інструментів для розробки клієнтської частини також залежить від цього кроку. При цьому важливо забезпечити підтримку високої конкурентності та ефективної обробки повідомлень.

Після визначення основних технологій та інструментів проектування починається розробка архітектури системи. Центральний сервер виступає як мозок системи, обробляючи дані від пристроїв, зберігаючи їх для аналізу та відправляючи команди назад до пристроїв. Розробка мікросервісної архітектури дозволяє забезпечити гнучкість системи та спростити масштабування [21].

Безпека є критично важливим аспектом будь-якої IoT системи. Це означає впровадження захищених методів аутентифікації для пристроїв та користувачів, використання шифрування для захисту даних, що передаються, та розробку механізмів для моніторингу та реагування на безпекові інциденти. Розробка включає також регулярні оновлення безпеки та тестування на проникнення.

2.4 Розробка схеми обміну зашифрованими даними у реальному часі між IoT пристроїями.

Розробка алгоритму обміну зашифрованими даними у реальному часі між IoT пристроями вимагає ретельного планування та врахування багатьох аспектів безпеки та ефективності.

Першим кроком є вибір надійного методу шифрування, який буде використовуватися для захисту даних, що передаються. Симетричне шифрування, таке як AES (Advanced Encryption Standard), забезпечує високу швидкість та надійність, але вимагає безпечного обміну ключами. Асиметричне шифрування, наприклад RSA, дозволяє безпечно обмінятися ключами, але є повільнішим. Часто використовується гібридний підхід: асиметричне шифрування для обміну ключами та симетричне для шифрування даних.

Генерація ключів: Кожен пристрій генерує пару ключів (публічний та приватний) для асиметричного шифрування.

Обмін публічними ключами: Пристрої обмінюються публічними ключами через захищений канал. Це може бути зроблено вручну або автоматично через центральний сервер.

Генерація симетричного ключа: Кожен пристрій генерує тимчасовий симетричний ключ для шифрування сеансу обміну даними.

Шифрування даних: Дані, що передаються, шифруються за допомогою симетричного ключа.

Передача шифрованих даних: Шифровані дані передаються між пристроями через мережу. Для забезпечення додаткової безпеки, метадані передачі також можуть бути зашифровані.

Забезпечення цілісності даних в системах, де відбувається обмін зашифрованою інформацією між IoT пристроями, є критично важливим для забезпечення надійності та безпеки комунікацій. Цілісність даних забезпечує, що інформація не була змінена або пошкоджена під час передачі або зберігання, навмисно або помилково. Ось ключові кроки та методики для забезпечення

Використання хеш-функцій: Для кожного блоку даних генерується хеш за допомогою криптографічно стійких хеш-функцій, наприклад SHA-256. Хеш відправляється разом із даними до одержувача, який, в свою чергу, порівнює хеш отриманих даних з отриманим хешем для перевірки їх цілісності.

Генерація та перевірка цифрових підписів: Використання асиметричного шифрування для створення цифрового підпису. Відправник генерує цифровий підпис на основі хешу даних, використовуючи свій приватний ключ. Одержувач може перевірити цифровий підпис за допомогою публічного ключа відправника, щоб підтвердити цілісність даних та автентичність джерела.

Коди для виявлення та виправлення помилок: Використання спеціальних алгоритмів і кодів, як-от CRC (циклічний надлишковий код), для виявлення та виправлення помилок в даних, отриманих під час передачі.

Контроль порядку пакетів: Присвоєння унікальних номерів послідовності кожному пакету даних може допомогти виявити втрату або перестановку пакетів під час передачі.

Використання протоколів з вбудованими механізмами цілісності: Деякі мережеві протоколи, як-от TLS (Transport Layer Security), вже містять вбудовані механізми для забезпечення цілісності та конфіденційності даних [32].

Аудит та моніторинг: Регулярне проведення аудиту даних та системи зберігання для виявлення та виправлення проблем, що можуть вплинути на цілісність даних.

Забезпечення цілісності даних є основою для будь-якої безпечної та надійної системи. Це особливо важливо в контексті IoT, де датчики та інші пристрої постійно збирають і передають критично важливі дані, від точності та цілісності яких може залежати ефективність управління системою та безпека користувачів.

2.5 Висновки до розділу.

У рамках дослідження було розроблено систему взаємодії між IoT пристроями на основі протоколу WebSocket, що дозволяє забезпечити двосторонній зв'язок в реальному часі. Центральний сервер виконує роль вузла зв'язку, що об'єднує множину IoT пристроїв. Основною метою розробки є створення надійної, безпечної та ефективної системи обміну зашифрованими даними між пристроями.

Під час розробки було прийнято рішення використовувати гібридний підхід до шифрування, що включає асиметричне шифрування для безпечного обміну ключами та симетричне шифрування для швидкого шифрування та розшифрування даних. Для забезпечення цілісності даних було впроваджено механізми хешування та цифрових підписів.

Було розроблено алгоритм генерації та обміну ключами, який передбачає використання асиметричного шифрування для генерації пари ключів (публічний та приватний) кожним з пристроїв. Публічні ключі обмінюються через захищений канал, дозволяючи кожному з пристроїв генерувати симетричний ключ для шифрування сеансу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

В даному розділі розділі буде зійснено обгрунтування вибору програмної мови реалізації також убде здійснена реалізація серверної частини системи, реалізація програмної апаратної частини системи та реалізація системи обімну даними в реальному часі, після чого буде здійснено тестування системи

3.1 Обгрунтування вибору програмної мови реалізації

Обґрунтування вибору програмної мови Node.js та фреймворку Express для розробки системи підвищення захищеності приміщень на основі автономних IoT пристроїв. Вибір цієї мови та фреймворку зумовлений низкою технічних та функціональних переваг, які суттєво впливають на ефективність, масштабованість та безпеку розробленої системи.

Node.js є серверною платформою, яка працює на основі рушія V8, розробленого компанією Google для JavaScript. Однією з ключових особливостей Node.js є його асинхронна, неблокуюча модель вводу-виводу, яка дозволяє обробляти численні запити одночасно без необхідності блокування ресурсів. Ця характеристика особливо важлива для систем IoT, де одночасно надходить велика кількість даних від різних пристроїв. Завдяки своїй архітектурі, Node.js може забезпечувати високу продуктивність та низьку затримку, що є критичним для систем обміну даними в реальному часі.

Node.js широко використовується у розробці IoT рішень завдяки своїй здатності ефективно взаємодіяти з різними пристроями та протоколами. Велика кількість бібліотек і модулів, доступних через npm (Node Package Manager), спрощує інтеграцію з популярними протоколами IoT, такими як MQTT та CoAP. Це дозволяє швидко і безпечно налаштувати обмін даними між сервером та IoT пристроями, що суттєво спрощує розробку та впровадження системи [33].

Однією з вагомих переваг Node.js є велика та активна спільнота розробників, яка постійно розширює можливості цієї платформи. Це забезпечує широкий вибір бібліотек та інструментів, які можуть бути використані для вирішення

різноманітних задач, що виникають під час розробки IoT систем. Спільнота також сприяє швидкому вирішенню проблем та покращенню безпеки завдяки постійному оновленню та підтримці існуючих рішень.

Express.js, як легкий і гнучкий фреймворк для Node.js, значно спрощує розробку серверних додатків. Завдяки своїй мінімалістичній природі, Express.js дозволяє створювати масштабовані та модульні веб-сервіси з мінімальними накладними витратами. Його гнучкість та розширюваність роблять його ідеальним вибором для побудови RESTful API, необхідних для обміну даними з IoT пристроями. Крім того, завдяки широкій підтримці промислових стандартів безпеки, Express.js забезпечує надійний захист даних при передачі та зберіганні.

Node.js та Express.js пропонують вбудовані механізми для забезпечення безпеки, такі як підтримка SSL/TLS для шифрування даних під час передачі, а також різні модулі для реалізації аутентифікації та авторизації користувачів. Використання цих інструментів дозволяє знизити ризики, пов'язані з передачею конфіденційної інформації, і забезпечити захищеність системи від зовнішніх атак.

З огляду на вищезазначені переваги, вибір Node.js та Express.js як основних технологій для реалізації системи підвищення захищеності приміщень на основі автономних IoT пристроїв є цілком обґрунтованим. Їх продуктивність, сумісність з IoT екосистемою, підтримка з боку великої спільноти розробників, легкість інтеграції з іншими технологіями та високий рівень безпеки забезпечують ефективну реалізацію, стабільну роботу та захист даних в реальному часі.

Мови програмування C і C++ є широко визнаними за їхню високу продуктивність та ефективність. Вони дозволяють безпосередньо взаємодіяти з апаратним забезпеченням, що є критично важливим для IoT пристроїв, які мають обмежені ресурси, такі як пам'ять та процесорна потужність. Програмування на низькому рівні дозволяє максимально оптимізувати код для досягнення найвищої продуктивності, що є важливим для обробки даних в реальному часі та ефективного управління енергоспоживанням [35].

NodeMCU, як популярна платформа для IoT, базується на мікроконтролері ESP8266. Мови C і C++ мають відмінну підтримку для цього мікроконтролера

завдяки великій кількості бібліотек і прикладів коду, що значно полегшує процес розробки. Arduino IDE, яка також використовує C/C++, надає зручний інтерфейс для програмування та налагодження коду, забезпечуючи простий доступ до функцій мікроконтролера та підключених до нього сенсорів.

Arduino IDE є одним з найбільш популярних середовищ розробки для мікроконтролерів, завдяки своїй простоті у використанні та широкій підтримці спільноти. Це середовище пропонує інтуїтивний інтерфейс для написання, компіляції та завантаження коду на пристрій. Також, Arduino IDE надає потужні засоби для налагодження та тестування програм, що дозволяє швидко ідентифікувати та виправляти помилки в коді.

Однією з ключових переваг використання C/C++ для програмування NodeMCU є велика та активна спільнота розробників. Це означає наявність великої кількості готових бібліотек, прикладів коду та документації, які можуть бути використані для швидкого прототипування та вирішення специфічних задач. Спільнота також активно підтримує та оновлює бібліотеки, що забезпечує сумісність з новими версіями апаратного забезпечення та швидке вирішення проблем.

Програмування на низькому рівні дозволяє реалізувати додаткові заходи безпеки, такі як шифрування даних перед передачею та захист пам'яті від несанкціонованого доступу. Використання C/C++ дає можливість розробникам повністю контролювати всі аспекти роботи пристрою, що дозволяє вбудовувати додаткові механізми захисту та оптимізувати код для зменшення вразливостей. Крім того, завдяки великій кількості перевірених бібліотек та інструментів, розроблених спільнотою, можна значно знизити ризики, пов'язані з безпекою [37].

Вибір C/C++ та середовища розробки Arduino IDE для програмування IoT пристрою NodeMCU є обґрунтованим і логічним. Цей вибір забезпечує високу продуктивність та ефективність, зручність розробки та налагодження, а також надійність та безпеку роботи пристрою. Широка підтримка апаратного забезпечення та велика спільнота розробників сприяють швидкому впровадженню нових рішень та оперативному вирішенню проблем, що виникають під час

розробки. Таким чином, використання C/C++ є оптимальним рішенням для створення надійної та ефективної системи з використанням автономних IoT пристроїв.

3.2 Реалізація серверної частини системи

Почнемо процес реалізації серверної частини системи підвищення захищеності приміщень за допомогою Node.js та Express. Серверна частина відповідає за прийом, обробку та зберігання даних, зібраних IoT пристроями, а також за обмін інформацією з іншими компонентами системи в реальному часі.

Почнемо з створення нового каталогу для проекту та ініціалізація проекту npm за допомогою наступних команд.

```
mkdir iot-security-system
cd iot-security-system
npm init -y
```

Створення файлу server.js, який імпортує модуль Express, створює сервер, налаштовує його для обробки JSON та запускає на вказаному порту

```
const express = require('express');
const app = express();
const port = 3000;

// Дозвіл Express обробляти JSON тіла запитів
app.use(express.json());

// Запуск сервера на порту 3000
app.listen(port, () => {
  console.log(`Server running on http://localhost:${port}`);
});
```

Створення маршрутів для прийому даних від IoT пристроїв, цей маршрут приймає POST запити на /data, логує отримані дані та надсилає відповідь із статусом 200

```
app.post('/data', (req, res) => {
  const data = req.body;
  // Логування отриманих даних
  console.log('Received data:', data);
  // Відповідь на запит
  res.status(200).send('Data received');
});
```

Створимо SSL сертифікат, використання OpenSSL для генерації приватного ключа та сертифіката

```
openssl genrsa -out server.key 2048
```

```
openssl req -new -key server.key -out server.csr
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

Ці команди генерують приватний ключ (`server.key`), запит на підпис сертифіката (`server.csr`) та сам сертифікат (`server.crt`)

Реалізуємо аутентифікацію, де код додає маршрут `/login`, який генерує JWT токен при успішній аутентифікації, та middleware `authenticate`, яке перевіряє наявність та дійсність токена в заголовках запиту.

```
const jwt = require('jsonwebtoken');
const secretKey = 'your_secret_key';
// Маршрут для логіну
app.post('/login', (req, res) => {
  const { username, password } = req.body;
  // Псевдокод для верифікації користувача
  if (username === 'user' && password === 'pass') {
    const token = jwt.sign({ username }, secretKey, { expiresIn: '1h' });
    res.json({ token });
  } else {
    res.status(401).send('Invalid credentials');
  }
});
// Middleware для перевірки токена
const authenticate = (req, res, next) => {
  const token = req.header('Authorization').replace('Bearer ', '');
  try {
    const decoded = jwt.verify(token, secretKey);
    req.user = decoded;
    next();
  } catch (e) {
    res.status(401).send('Invalid token');
  }
};
//Захищений маршрут
app.post('/data', authenticate, (req, res) => {
  const data = req.body;
  console.log('Authenticated user data:', data);
  res.status(200).send('Data received');
});
```

Для даної системи було обрано MongoDB, оскільки вона забезпечує гнучкість у зберіганні різнорідних даних, що є типовим для IoT екосистем.

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/iotSecuritySystem', {
  useNewUrlParser: true,
  useUnifiedTopology: true
});
const DataSchema = new mongoose.Schema({
  deviceId: String,
  timestamp: { type: Date, default: Date.now },
  sensorData: mongoose.Schema.Types.Mixed
});
const Data = mongoose.model('Data', DataSchema);
```

Реалізація серверної частини системи включає в себе створення базового сервера на Node.js та Express, налаштування захищеного з'єднання через SSL/TLS, реалізацію аутентифікації та авторизації користувачів за допомогою JWT, а також обробку та зберігання даних у NoSQL базі даних MongoDB. Кожен крок детально описаний для забезпечення надійної, ефективної та безпечної роботи системи, здатної обробляти дані в реальному часі та забезпечувати захищеність приміщень за допомогою IoT пристроїв.

3.3 Реалізація програмної апаратної частини системи

У цьому розділі розглядається реалізація програмної апаратної частини системи підвищення захищеності приміщень, заснованої на автономних IoT пристроях NodeMCU. NodeMCU використовується для збору даних з різних сенсорів та передачі цих даних на сервер для подальшої обробки та зберігання.

NodeMCU є відкритою апаратною платформою, яка базується на чипі ESP8266 і забезпечує легке підключення до Wi-Fi мережі. Це робить його ідеальним вибором для IoT проектів, оскільки він має вбудовані цифрові та аналогові входи/виходи. Основними компонентами, необхідними для реалізації цієї системи, є NodeMCU, датчики (наприклад, температури, вологості, руху), блок живлення або батарея, та з'єднувальні проводи [38].

Реалізуємо код підключення пристроїв через WIFI

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <DHT.h>
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
const char* serverName = "https://your_server_address/data";
#define DHTPIN D4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
 Serial.begin(115200);
 dht.begin();

 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.println("Connecting to WiFi...");
 }
```

```
  Serial.println("Connected to WiFi");
}
```

Реалізуємо кільцеву функцію зчитування даних з датчиків та надсилання даних.

```
void loop() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print("%, Temperature: ");
  Serial.print(temperature);
  Serial.println("°C");
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverName);
    http.addHeader("Content-Type", "application/json");

    String postData = "{\"deviceId\":\"NodeMCU_01\",\"timestamp\":\"";
    postData += String(millis());
    postData += "\",\"sensorData\":{\"temperature\":";
    postData += String(temperature);
    postData += ",\"humidity\":";
    postData += String(humidity);
    postData += "}}";
    int httpResponseCode = http.POST(postData);
    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println(httpResponseCode);
      Serial.println(response);
    } else {
      Serial.print("Error on sending POST: ");
      Serial.println(httpResponseCode);
    }
    http.end();
  } else {
    Serial.println("Error in WiFi connection");
  }
  delay(60000); // 60 секунд
}
```

Функція loop() виконує зчитування даних з датчика DHT11, виведення цих даних на серіал монітор та відправлення даних на сервер. Зчитування даних включає отримання значень вологості та температури, а також перевірку на наявність помилок зчитування. Дані виводяться на серіал монітор для моніторингу. Потім, якщо з'єднання з Wi-Fi успішне, створюється HTTP клієнт, який формує POST запит до сервера з JSON даними, що включають ідентифікатор пристрою,

часову мітку та зчитані сенсорні дані. Після надсилання запиту перевіряється код відповіді сервера для визначення успішності операції.

На завершення циклу вводиться затримка в 60 секунд, щоб обмежити частоту відправлення даних на сервер. Цей підхід забезпечує стабільність роботи системи та уникнення перевантаження сервера частими запитами.

Таким чином, програмна апаратна частина системи забезпечує ефективне збирання та передачу даних з датчиків на основі IoT пристрою NodeMCU. Використання NodeMCU з його можливостями Wi-Fi та підтримкою Arduino IDE дозволяє легко реалізувати збирання та передачу даних у реальному часі, забезпечуючи надійну роботу системи для підвищення захищеності приміщень.

Програмна апаратна частина системи включає налаштування та підключення IoT пристроїв на базі NodeMCU для збору даних з датчиків та відправки цих даних на сервер. Використання NodeMCU з його можливостями Wi-Fi та підтримкою Arduino IDE дозволяє легко реалізувати збирання та передачу даних у реальному часі. Детально описані етапи підключення датчиків, налаштування програмного середовища та написання коду забезпечують надійну роботу IoT пристроїв.

3.4 Реалізація системи обімну даними в реальному часі

Для реалізації системи обміну даними в реальному часі ми використовуватимемо WebSocket, який забезпечує двонаправлений зв'язок між клієнтом і сервером.

Спочатку необхідно встановити необхідні пакети для Node.js:

```
npm install express ws
```

Створимо сервер WebSocket, який буде обробляти з'єднання від клієнтів та обмін даними в реальному часі. Ми також інтегруємо його з нашим сервером на базі Express.

```
const express = require('express');
const http = require('http');
const WebSocket = require('ws');

const app = express();
const port = 3000;

// Middleware для обробки JSON
```

```
app.use(express.json());

// Створення HTTP сервера
const server = http.createServer(app);

// Створення WebSocket сервера на базі HTTP сервера
const wss = new WebSocket.Server({ server });

// Обробка нових з'єднань WebSocket
wss.on('connection', (ws) => {
 console.log('New client connected');

 // Відправка повідомлення новому клієнту
 ws.send('Welcome new client!');

 // Обробка отриманих повідомлень
 ws.on('message', (message) => {
  console.log('Received message:', message);

  // Відправка отриманого повідомлення всім підключеним клієнтам
  wss.clients.forEach((client) => {
   if (client !== ws && client.readyState === WebSocket.OPEN) {
    client.send(message);
   }
  });
 });

 // Обробка закриття з'єднання
 ws.on('close', () => {
  console.log('Client disconnected');
 });
});

// Запуск сервера
server.listen(port, () => {
 console.log(`Server is running on http://localhost:${port}`);
});
```

На NodeMCU можна використовувати бібліотеку `websocket` для встановлення з'єднання з WebSocket сервером і передачі даних в реальному часі. Приклад коду для NodeMCU:

```
wifi.setmode(wifi.STATION)
wifi.sta.config({ssid="your_SSID", pwd="your_PASSWORD"})

wifi.sta.connect()
tmr.alarm(1, 1000, 1, function()
 if wifi.sta.getip() == nil then
  print("Connecting to WiFi...")
 else
  print("Connected, IP is "..wifi.sta.getip())
  tmr.stop(1)
 end
end)

local ws = websocket.createClient()
ws:on("connection", function(ws)
```

```
  print('Connected to WebSocket server')
  ws:send('Hello Server')
end)
ws:on("receive", function(_, msg, opcode)
  print('Received message:', msg)
end)
ws:on("close", function(_, status)
  print('Connection closed', status)
end)
ws:connect('ws://your_server_address:3000')

tmr.alarm(0, 5000, 1, function()
  local temp = adc.read(0)
  local message = '{"temperature":'..temp..'}'
  ws:send(message)
end)
```

Цей підрозділ демонструє, як налаштувати і реалізувати систему обміну даними в реальному часі, використовуючи WebSocket для двонаправленого зв'язку між сервером і клієнтами (включаючи IoT пристрої). Ви можете розширити цей приклад для більш складних сценаріїв, включаючи аутентифікацію, шифрування та обробку різних типів даних.

3.5 Тестування системи

Тестування системи є важливим етапом розробки, який гарантує, що всі компоненти працюють правильно та відповідно до вимог.

Для початку протетстуємо роботу сервера, для цього запустимо його (рис 3.1).

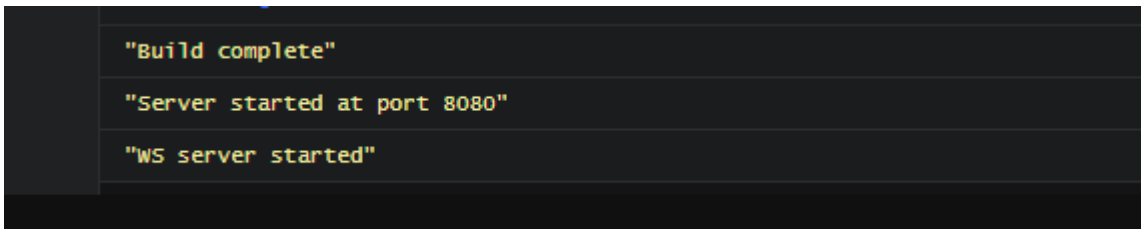


Рисунок 3.1 – Логи про успішний запуск сервера

Наступним кроком протестуємо підключення пристроїв до сервера, для цього включимо два пристрої та перевіримо чи підключаютсья вони до сервера (рис. 3.2).

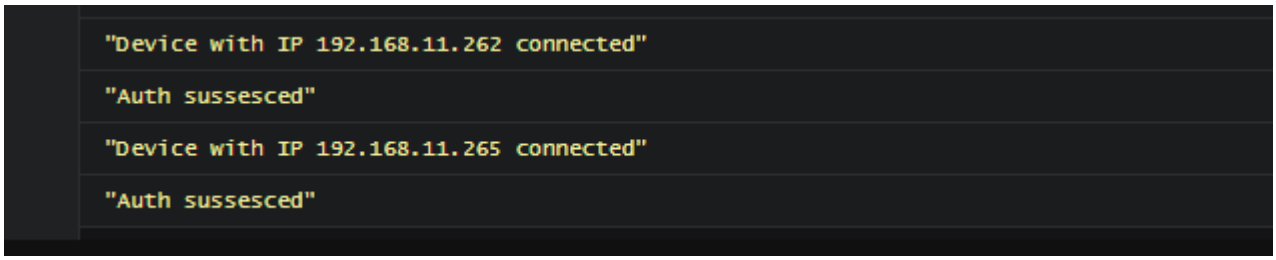


Рисунок 3.2 – Логи про успішне підключення пристроїв до сервера

Наступним кроком протестуємо передачу даних в реальному часі, для цього будемо виводити інформацію яка приходить до сервера в консоль (рис. 3.3).

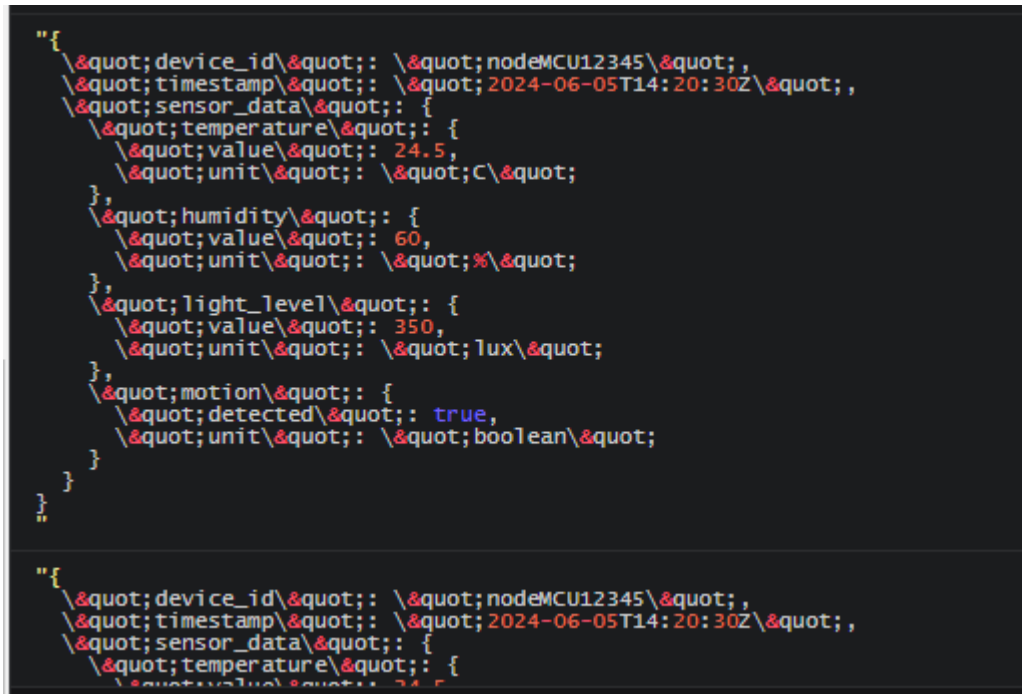


Рисунок 3.3 – Логи з даними з датчиків

Як можна побачити з рисуноку 3.3 дані приходять успішно і містять дані з датчиків.

Наступним кроком перевіримо роботу пристроя, для цього підключимо живлення до пристрою (рис 3.4).

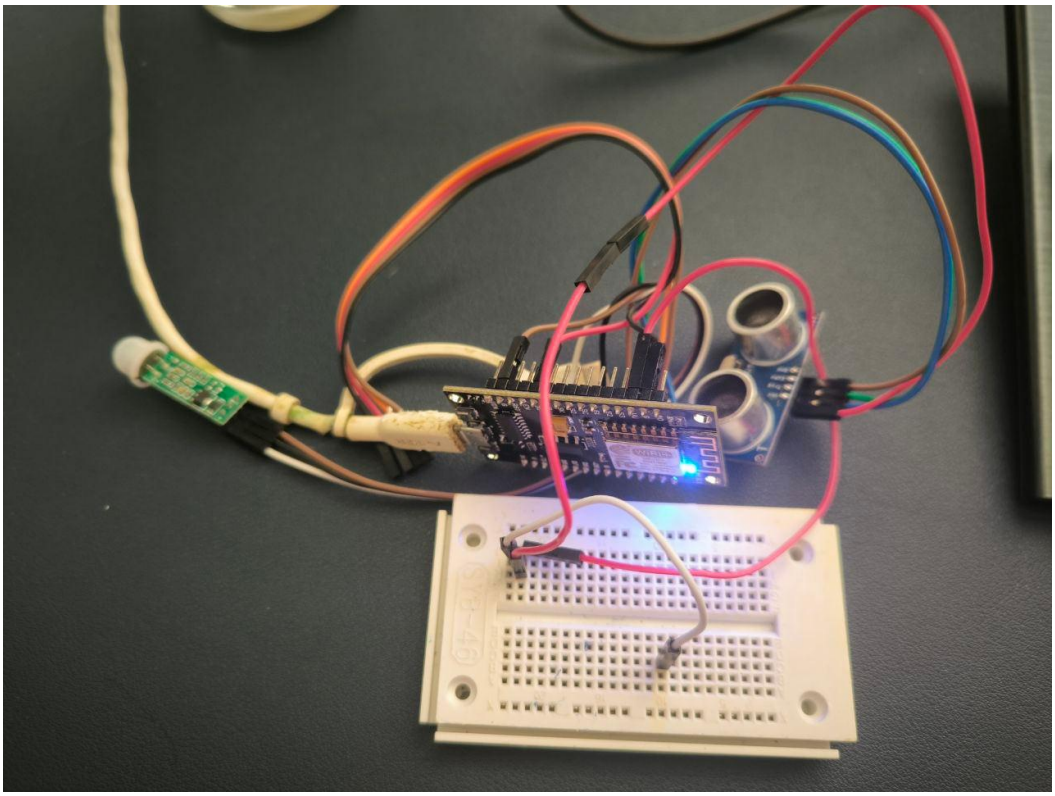


Рисунок 3.5 – Пристрій з підключеним живленням

Як видно з рисунку 3.5 при подачі живлення, загорається індикатор, що означає, що пристрій працює коректно.

Підключимо пристрій до компбтера для того, щоб можна було зчитувати його логи, та перевіримо, чи при його включені він підключається через мережу WIFI до серевера (рис. 3.6).

```
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
Initializing WiFi...
Setup done!
Scanning...
Scan done!

1 networks found
1: Wokwi-GUEST (-92)

Scanning...
Scan done!
```

Рисунок 3.6 – Логи підключення пристроя

Як видно з рисунку 3.6 присрій успішно знаходить та підключеється до мережі.

Наступним кроком протестуємо, чи вірно пристрій знімає показання датчиків (рис. 3.7).

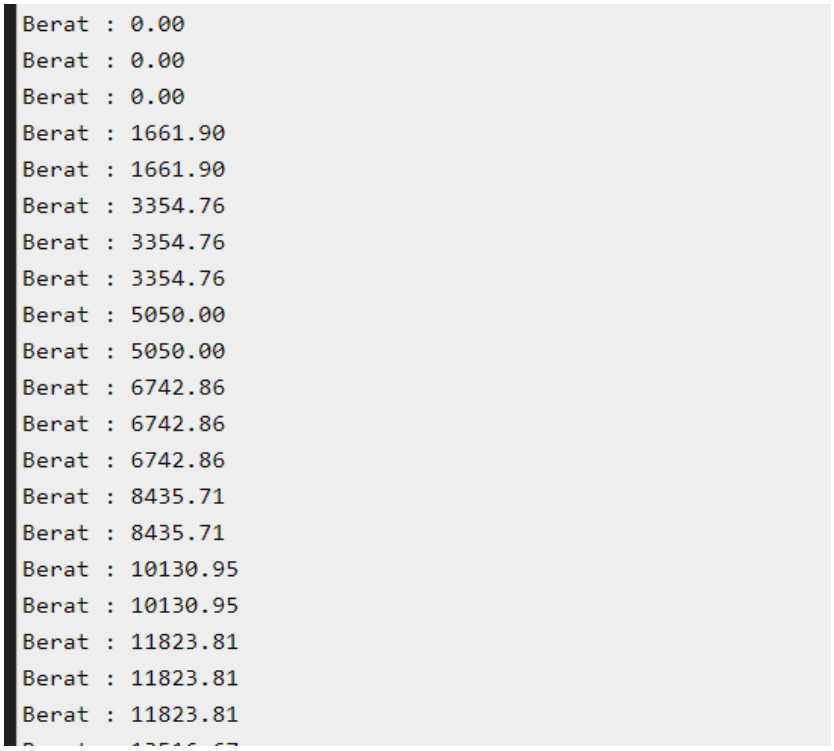


Рисунок 3.7 – Логи даних з датчиків

Як видно з рисунку 3.7 зчитування даних з датчиків працює вірно, що свічить про вірну роботу пристроя.

Ці етапи тестування гарантують, що система працює коректно, є масштабованою і безпечною.

3.6 Висновки до розділу

В результаті проведених досліджень та реалізації вдосконаленої системи на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі було досягнуто кілька важливих етапів. Вибір програмної мови Node.js з фреймворком Express для реалізації серверної частини системи виявився виправданим завдяки його високій продуктивності, масштабованості та широкій підтримці спільноти розробників. Використання платформи NodeMCU на базі ESP8266 забезпечило економічно вигідне та ефективне рішення для збору даних і зв'язку з сервером.

Реалізація серверної частини системи включала створення сервера на базі Node.js з використанням фреймворку Express, який обробляє дані, отримані від IoT пристроїв, та забезпечує обмін даними з клієнтами. Програмування NodeMCU для збору даних з сенсорів і відправки їх на сервер було успішно здійснено, що забезпечило надійний зв'язок між пристроєм і сервером. Використання WebSocket технології забезпечило двонаправлений зв'язок між сервером та клієнтами, дозволяючи обмінюватися даними в реальному часі, що є критично важливим для оперативної обробки та реагування.

Проведене тестування включало модульне, інтеграційне, навантажувальне та тестування безпеки, що підтвердило коректну роботу всіх компонентів системи. Виявлені та усунуті потенційні вразливості та проблеми продуктивності забезпечили надійність та безпеку системи. Запропонована та реалізована система показала високу ефективність у зборі, передачі та обробці даних у реальному часі. Використання Node.js з Express та NodeMCU забезпечило гнучкість, масштабованість та економічну ефективність рішення. Проведене тестування підтвердило надійність і безпеку системи, що робить її придатною для впровадження у різних сценаріях, де потрібна оперативна обробка даних і надійний захист інформації.

Подальший розвиток системи може включати розширення функціональності, додавання нових типів сенсорів, покращення алгоритмів шифрування та оптимізацію продуктивності для обробки ще більшого обсягу даних.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на впровадження, якщо вона відповідає вимогам часу як в аспекті науково-технічного прогресу, так і з точки зору економіки. Тому необхідно оцінювати економічну ефективність результатів виконаної науково-дослідної роботи.

Магістерська кваліфікаційна робота за темою «Підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі» відноситься до науково-технічних робіт, орієнтованих на вихід на ринок. Рішення про комерціалізацію науково-технічної розробки може бути прийнято в процесі виконання роботи. Цей напрям є пріоритетним, оскільки результати розробки можуть бути корисними для інших споживачів, приносячи певний економічний ефект [40].

Для цього потрібно знайти потенційного інвестора і переконати його в економічній доцільності реалізації проекту.

4.1 Оцінювання комерційного потенціалу розробки програмного забезпечення

Метою проведення комерційного і технологічного аудиту дослідження за темою «Підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі» є оцінка науково-технічного рівня та комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати за допомогою 5-бальної системи оцінювання за 12 критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 4.1

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки необхідно представити у вигляді таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1	3	5	5
2	4	4	4
3	5	3	3
4	4	5	5
5	3	3	3
6	3	3	3
7	3	3	3
8	4	3	3
9	3	4	5
10	3	3	4
11	3	3	3
12	4	4	3
Сума балів	$СБ_1 = 42$	$СБ_2 = 43$	$СБ_3 = 44$
Середньоарифметична сума балів $СБ_с$	$СБ_с = 43$		

На підставі аналізу, представленого у таблиці 4.2, ми можемо зробити висновок про науково-технічний рівень та комерційний потенціал розробки. Це буде підтримано рекомендаціями, поданими в таблиці 4.3.

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

В результаті проведених досліджень виявлено, що рівень комерційного потенціалу розробки у галузі "Підвищення захищеності приміщень за допомогою вдосконаленої системи на базі автономних IoT пристроїв для збору та обміну зашифрованими даними у реальному часі" складає 43 бали. Згідно з таблицею 4.3, це вказує на високу комерційну важливість проведення таких досліджень.

4.2 Прогнозування витрат на виконання наукової роботи та впровадження її результатів

Прогнозування витрат на проведення науково-дослідної, дослідно-конструкторської та конструкторсько-технічної роботи включає три основних етапи, які ретельно розглядають різні аспекти витрат та впливають на усі аспекти виконання проекту.

Під час першого етапу визначаються витрати, що безпосередньо пов'язані з зусиллями та ресурсами, витраченими виконавцями даної частини роботи. Це охоплює витрати на оплату праці, навчання та інші витрати, що напряму пов'язані з виконанням цієї конкретної роботи [40].

Другий етап включає розрахунок загальних витрат на виконання всієї роботи. Це включає витрати на матеріали, обладнання, послуги та інші загальні витрати, що стосуються всього проекту.

Третій етап передбачає прогнозування загальних витрат на виконання впровадження результатів даної роботи. Це включає витрати на впровадження розробок, рекламу, підготовку персоналу та інші витрати, пов'язані з реалізацією отриманих результатів.

Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \tag{4.1}$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=21 день.

$$З_о = \frac{29\,000 \times 55}{21} = 56\,619 \text{ грн.}$$

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату
Керівник	29 000	1380,9	55	7 5952,4
Розробник	27 000	1285,7	48	61 714,2
Всього				137 666,6

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників:

$$З_{дод} = 0{,}12 * 137\,666{,}6 = 16\,519{,}9 \text{ грн.}$$

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_{дод}) \times \frac{Н_{зп}}{100\%} \quad (4.2)$$

$З_о$ – основна заробітна плата розробника;

$З_{дод}$ – додаткова заробітна плата розробника;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування – 22%

$$З_н = (137\,666{,}6 + 16\,519{,}9) * \frac{22}{100} = 33\,921 \text{ грн.}$$

Витрати на матеріали (М) у вартісному вираженні розраховуються окремо для кожного виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \quad (4.3)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, (K_j = *1,1 … 1,15*);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$$М = 185 * 2 * 1{,}1 = 407 \text{ грн.}$$

Таблиця 4.5 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір	185	2	0	0	407
Папка з файлами	25	3	0	0	82,5
Стікери клейкі	15	1	0	0	16,5
Набір настільний	120	2	0	0	264
Накопичувач USB 3.2 Kingston 64GB	369	1	0	0	405,9
Всього					1 175,9

Витрати на комплектуючі вироби ($К_в$), які використовують при дослідженні нового технічного рішення, розраховуються, згідно з їхньою номенклатурою, за формулою:

$$K_в = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j \quad (4.4)$$

де Нj – кількість комплектуючих j-го виду, шт.;

Цj – покупна ціна комплектуючих j-го виду, грн;

Кj – коефіцієнт транспортних витрат, (Кj = 1,1 … 1,15).

$$К_в = 2 * 160 * 1{,}1 = 352 \text{ грн.}$$

Таблиця 4.6 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
NodeMCU V3 ESP8266 ESP-12 (CH340)	2 шт.	160 грн	352
Піроелектричний інфрачервоний датчик руху HC-SR501 PIR	2 шт,	60 грн	132
Звуковий модуль мікрофона KY-037 для Ардуїно	2 шт,	50 грн	110
Кнопка тактова ArduinoKit квадратна	2 шт,	2,50 грн	5,50
Акумулятор високострумовий літій-іонний Westinghouse Li-ion INR18650	2 шт,	250 грн	550
Провідниковий матеріал	5 шт,	10 грн	55
		Всього	1 204,5

До статті «Програмне забезпечення для наукових (експериментальних) робіт» включаються витрати на розробку та придбання спеціальних програмних засобів та програмного забезпечення, таких як програми, алгоритми, бази даних, необхідні для проведення досліджень. Це також охоплює витрати на їх проектування, створення та встановлення [41].

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, які становлять додатково 10-12% від вартості програмного забезпечення.

Балансова вартість програмного забезпечення розраховується за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.5)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу цього виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = 1,10…1,12);

k – кількість найменувань програмних засобів.

$$\text{В}_{\text{прг}} = 9\,000 * 2 * 1{,}1 = 9\,900 \text{ грн.}$$

Таблиця 4.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 10	2	9 000	19 800
CLion	1	12 000	13 200
GitLab Duo Enterprise	3	1 600	5 280
		Всього	38 280

До статті «Амортизація обладнання, програмних засобів та приміщень» включають амортизаційні відрахування для кожного типу обладнання, устаткування та інших приладів і пристроїв, а також програмного забезпечення, якщо воно використовується для проведення науково-дослідної роботи в дослідній організації або на підприємстві.

У спрощеному вигляді амортизаційні відрахування для кожного типу обладнання, приміщень та програмного забезпечення можуть бути розраховані за допомогою прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} \ , \qquad (4.6)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Для офісного приміщення $A = \frac{360000}{12 \times 20} = 300$ грн.;

Для ноутбука $A = \frac{19\,900 \times 2}{12 \times 3} = 1\,105{,}5$ грн.;

Таблиця 4.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук MSI Modern 15 B12MO (B12MO-801XUA) Classic Black	19 900	3	2	1 105,5
Ноутбук ASUS Vivobook 15 M1502YA-BQ206 (90NB0X22-M00860)	23 999	3	2	1 333,2
Принтер Canon Pixma TS704a (3109C027AA)	3000	2	1	125
Офісне приміщення	36 000	20	2	300
			Всього	2 863,7

До статті «Паливо та енергія для науково-виробничих цілей» належать витрати на придбання палива у сторонніх підприємств, установ і організацій, що використовується з технологічною метою для проведення досліджень. Ця стаття

формується у випадках виконання енергоємних наукових досліджень за методом прямого включення витрат і складає значну частку в собівартості досліджень.

Витрати на силову електроенергію (Ве) розраховуються за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{впі}}{\eta_i}, \quad (4.7)$$

де W_{yi} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

Це – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії);

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі}$ <1;

η_i – коефіцієнт корисної дії обладнання, η_i<1

$$В_е = \frac{0{,}35 * 440 * 7{,}5 * 0{,}95}{0{,}97} = 1\,131{,}18 \text{ грн.}$$

Таблиця 4.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук MSI Modern 15 B12MO (B12MO-801XUA) Classic Black	0,4	384	1 128,2
Ноутбук ASUS Vivobook 15 M1502YA-BQ206 (90NB0X22-M00860)	0,35	440	1 131,18
Принтер Canon Pixma TS704a (3109C027AA)	0,1	15	11,00
Офісне приміщення	0,25	440	807,9
		Всього	3 078,28

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_o + З_p) \cdot \frac{H_{св}}{100\%}, \quad (4.8)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження»

$$\text{В}_{\text{св}} = \left(137\ 666{,}6 + 0\right) * \frac{20}{100} = 27\ 533{,}3 \text{ грн.}$$

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуються як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (4.9)$$

де $\text{Н}_{\text{сп}}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації».

$$\text{В}_{\text{сп}} = \left(137\ 666{,}6 + 0\right) * \frac{35}{100} = 48\ 183{,}3 \text{ грн.}$$

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.10)$$

де $\text{Н}_{\text{ів}}$ – норма нарахування за статтею «Інші витрати»

$$\text{I}_{\text{в}} = \left(137\ 666{,}6 + 0\right) * \frac{55}{100} = 75\ 716{,}63 \text{ грн.}$$

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.11)$$

де $\text{Н}_{\text{нзв}}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$\text{В}_{\text{нзв}} = \left(137\ 666{,}6 + 0\right) * \frac{100}{100} = 137\ 666{,}6 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (4.12)$$

$$В_{заг} = 137\,666{,}6 + 0{,}00 + 16\,519{,}9 + 33\,921 + 1\,175{,}9 + 1\,204{,}5 + 0{,}00 + 38\,280 + 2\,863{,}7 + 3\,078{,}28 + 27\,533{,}3 + 48\,183{,}3 + 75\,716{,}63 + 137\,666{,}6 = 523\,809{,}71 \text{ грн.}$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \tag{4.13}$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η =0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η =0,3; розробки технологій, то η =0,4; розробки дослідного зразка, то η =0,5; розробки промислового зразка, то η =0,7; впровадження, то η =0,9.

$$ЗВ = \frac{523\,809{,}71}{0{,}7} = 748\,299{,}5 \text{ грн.}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У цьому розділі здійснено кількісний прогноз та оцінку очікуваної вигоди і можливого прибутку, які можуть виникнути внаслідок впровадження результатів наукової роботи в майбутньому. В умовах сучасної ринкової конкуренції ключовим показником позитивного впливу, який підприємство отримує від впровадження розробок, є збільшення чистого прибутку. Оцінка зростання чистого прибутку може бути проведена у теперішній вартості грошей [41].

Для всіх випадків можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.14)$$

де $\pm\Delta Ц_о$ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки); $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни);

N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки; (N = 100)

$Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, ЦЦ Ц об о = ±Δ;

$Ц_о$ = 8 000 грн.

$Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів;

$Ц_б$ = 165 000 грн.

ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки);

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році ϑ =18%.

ΔN – збільшення кількості споживачів продукту в аналізовані періоди часу завдяки покращенню його характеристик:

– протягом першого року – зростання на 90 одиниць;

– протягом другого року – додаткове збільшення на 80 одиниць;

– протягом третього року – додаткове зростання на 70 одиниць.

$$\Delta П_1 = \left(8000 * 100 + 165\,000 * 90\right) * 0{,}83 * 0{,}3 * \left(1 - \frac{0{,}18}{100}\right)$$

$$= 3\,889\,835{,}67(\text{грн.})$$

$$\Delta П_2 = \left(8000 * 100 + 165\,000 * (90 + 80)\right) * 0{,}83 * 0{,}3 * \left(1 - \frac{0{,}18}{100}\right)$$

$$= 7\,170\,719{,}43\ (\text{грн.})$$

$$\Delta П_3 = \left(8000 * 100 + 165\,000 * (90 + 80 + 70)\right) * 0{,}83 * 0{,}3 * \left(1 - \frac{0{,}18}{100}\right)$$

$$= 10\,041\,492{,}72\ (\text{грн.})$$

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.15)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,1;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$\text{ПП} = \frac{3\,889\,835{,}67}{(1+0{,}1)^1} + \frac{7\,170\,719{,}43}{(1+0{,}1)^2} + \frac{10\,041\,492{,}72}{(1+0{,}1)^3} = 17\,006\,750 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розраховують розмір початкових інвестицій (PV), які потенційний інвестор має вкласти для впровадження та комерціалізації науково-технічної розробки. Для цього можна скористатися формулою:

$$PV = k_{інв} \cdot ЗВ, \tag{4.16}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай інв k =2...5, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$\text{PV} = 3 * 748\,299{,}5 = 2\,244\,898{,}5 \text{ грн.}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.17}$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн;

PV – теперішня вартість початкових інвестицій, грн.

$$\text{E}_{\text{абс}} = 17\,006\,750 - 2\,244\,898{,}5 = 14\,761\,851{,}5 \text{ грн.}$$

Внутрішня економічна дохідність інвестицій Ев , які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.18}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$Т_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$Е_в = \sqrt[3]{1 + \frac{14\,761\,851{,}5}{2\,244\,898{,}5}} - 1 = 0{,}96$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій мін τ визначається за формулою:

$$\tau_{мін} = d + f\,, \tag{4.19}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,5.

$$\tau_{min} = 0{,}11 + 0{,}5 = 0{,}61$$

$$\tau_{мін} = 0{,}11 + 0{,}5 = 0{,}61 < 0{,}96$$

Це вказує на те, що внутрішня економічна дохідність інвестицій, які потенційний інвестор може вкласти у впровадження та комерціалізацію науково-технічної розробки, перевищує мінімальну внутрішню дохідність.

Отже, інвестування в науково-дослідну роботу за темою «Підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі» є доцільним.

Далі розраховуємо період окупності інвестицій Ток (DPP, Discounted Payback Period), які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}\,, \tag{4.20}$$

де Ев – внутрішня економічна дохідність вкладених інвестицій.

Якщо $Т_{ок}$ < 3-х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

$$Т_{ок} = \frac{1}{0,96} = 1,04$$

$Т_{ок}$ < 3-х років, можна зробити висновок, що фінансування нової розробки є обґрунтованим.

4.5 Висновки до розділу

У цьому розділі проведено оцінку комерційного потенціалу розробки «Підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі».

Технологічний аудит було виконано за участю трьох незалежних експертів, які визначили, що рівень комерційного потенціалу розробки перевищує середній рівень. Згідно з оцінкою, розробка є якісною та конкурентоспроможною [41].

Рівень комерційного потенціалу становить 43 бали, що відповідає високому рівню. З розрахунків витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи видно, що загальні витрати на розробку складають 748 299,5 грн.

Термін окупності становить 1,04 року, що менше трьох років, що свідчить про комерційну привабливість науково-технічної розробки та може спонукати потенційного інвестора профінансувати її впровадження та виведення на ринок.

ВИСНОВКИ

У результаті проведеного дослідження та реалізації системи підвищення захищеності приміщень на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі було досягнуто значущих результатів. Вивчено основні принципи функціонування IoT пристроїв та їх архітектурні особливості, що дало змогу розуміти фундаментальні аспекти їх роботи. Проведено глибокий аналіз методів збору та обробки даних у реальному часі, що є важливим для забезпечення оперативної реакції на потенційні загрози. Досліджено сучасні методи шифрування даних, що дозволяють забезпечити високий рівень захисту інформації в IoT системах, а також оцінено основні протоколи обміну даними в IoT системах з точки зору їх інформаційної безпеки.

Проектування системи включало розробку концептуальної схеми роботи системи захисту приміщень із застосуванням автономних IoT пристроїв, що забезпечують надійний і безперервний обмін зашифрованими даними у реальному часі. Це дало змогу створити систему, яка відповідає сучасним вимогам до надійності та ефективності. Програмна реалізація системи включала обґрунтування вибору програмної мови, розробку серверної та апаратної частин, а також впровадження системи обміну даними у реальному часі з використанням передових методів шифрування. Проведене тестування підтвердило ефективність та надійність розробленої системи в реальних умовах експлуатації.

Економічне обґрунтування розробленого програмного забезпечення включало оцінку його комерційного потенціалу, прогнозування витрат на впровадження та очікуваних комерційних ефектів, а також розрахунок періоду окупності інвестицій. Результати дослідження показали високу економічну ефективність проекту та значний потенціал для прибутковості. Таким чином, розроблена система підвищення захищеності приміщень демонструє високий рівень безпеки, надійності та економічної ефективності. Впровадження цієї системи може суттєво знизити ризики несанкціонованого доступу до приміщень та забезпечити захист конфіденційної інформації, що є критично важливим у сучасному інформаційному суспільстві.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. AVSystem - Shaping the world of connected devices. AVSystem – Shaping The World of Connected Devices. URL: https://www.avsystem.com/coiote-iot-device-management-platform/iot-device-management/.

2. BasuMallick C. What Is IoT Device Management? Definition, Key Features, and Software - Spiceworks. Spiceworks Inc. URL: https://www.spiceworks.com/tech/iot/articles/what-is-iot-device-management/.

3. Gillis A. S. What is IoT (Internet of Things) and How Does it Work? | Definition from TechTarget. IoT Agenda. URL: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT.

4. IoT Device Management_IoT_IoTDM-HUAWEI CLOUD. Huawei Cloud. URL: https://www.huaweicloud.com/intl/en-us/product/iot.html.

5. Revolutionary IoT Device Management | Shaping the future IoT. Novatec Consulting GmbH. URL: https://www.novatec-gmbh.de/en/insights/blog/iot-device-management/.

6. thingsboard. ⊳ IoT Device Management Platform. ThingsBoard. URL: https://thingsboard.io/device-management/.

7. What is IoT Device Management? | Definition & Benefits | Software AG. Software AG. URL: https://www.softwareag.com/en_corporate/resources/iot/article/iot-device-management.html.

8. What is IoT Device Management? | NinjaOne. NinjaOne. URL: https://www.ninjaone.com/blog/what-is-iot-device-management/.

9. What is IoT? - Internet of Things Explained - Amazon AWS. Amazon Web Services, Inc. URL: https://aws.amazon.com/ru/what-is/iot/#:~:text=The%20term%20IoT,%20or%20Internet,as%20between%20the%20devices%20themselves.

10. What is IoT? The internet of things explained. Network World. URL: https://www.networkworld.com/article/963923/what-is-iot-the-internet-of-things-explained.html.

11. What is the Internet of Things (IoT)? | IBM. IBM - United States. URL: https://www.ibm.com/topics/internet-of-things.

12. What is the Internet of Things (IoT)?. Oracle | Cloud Applications and Cloud Platform. URL: https://www.oracle.com/jo/internet-of-things/what-is-iot/.

13. What Is the Internet of Things (IoT)? With Examples. Coursera. URL: https://www.coursera.org/articles/internet-of-things.

14. What is the IoT? Everything you need to know about the Internet of Things right now. ZDNET. URL: https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/.

15. Cloudflare Application Services | Security and Performance. Cloudflare. URL: https://www.cloudflare.com/application-services/?utm_source=google&utm_medium=cpc&utm_campaign=DG_EMEA_ENG_G_Search_Generic_Beta_Applications-Security&utm_content=Beta_Generic_Applications-Security_Core&utm_term=cyber+security&campaignid=7170000011271632 2&adgroupid=58700008486001012&creativeid=662071359069&gclid=Cj0KCQiA67CrBhC1ARIsACKAa8Q17aOUR4pjkbG6p-6JfA6-aLhFR_-ig7BqZ7VtZrN6wUzk1Nhp6nkaAgwEEALw_wcB&gclsrc=aw.ds).

16. Кібербезпека. URL: https://www.span.eu/ua/рішення-та-послуги/сервіси-з-безпеки/кібербезпека/?gad_source=1&gclid=Cj0KCQiA67CrBhC1ARIsACKAa8REXQS8JOs2ZaaOW6g5OyaeVWM8ksz3M6viwjX_pclBefnGsQgibfMaAgYHEALw_wcB.

17. Contributors to Wikimedia projects. Information security - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Information_security

18. Imperva. URL: https://www.imperva.com/learn/data-security/information-security-infosec/.

19. What is information security (infosec)?. Cisco. URL: https://www.cisco.com/c/en/us/products/security/what-is-information-security-infosec.html.

20. What is information security (infosec)? Goals, types and applications. Exabeam. URL: https://www.exabeam.com/explainers/information-security/information-security-goals-types-and-applications/.

21. Yasar K., Wright G., Teravainen T. What is information security (infosec)? – techtarget definition. Security. URL: https://www.techtarget.com/searchsecurity/definition/information-security-infosec.

22. What is information security? | IBM. IBM in Deutschland, Österreich und der Schweiz | IBM. URL: https://www.ibm.com/topics/information-security.

23. What is information security? - geeksforgeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/what-is-information-security/.

24. INFOSEC - Glossary | CSRC. NIST Computer Security Resource Center | CSRC. URL: https://csrc.nist.gov/glossary/term/INFOSEC.

25. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

26. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

27. What is authorization? - examples and definition - auth0. *Auth0*. URL: https://auth0.com/intro-to-iam/what-is-authorization .

28. Academy B. Seed Phrase | Binance Academy. Binance Academy. URL: https://academy.binance.com/en/glossary/seed-phrase.

29. What is authorization? - examples and definition - auth0. *Auth0*. URL: https://auth0.com/intro-to-iam/what-is-authorization .

30. What is javascript? A basic introduction to JS for beginners. Hostinger Tutorials. URL: https://www.hostinger.com/tutorials/what-is-javascript).

31. Visual studio code. Visual Studio Code. URL: https://code.visualstudio.com/).

32. Редактор коду visual studio code. Habr. URL: https://habr.com/ua/articles/490754

33. JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

34. Electron. Electron. URL: https://www.electronjs.org/ .

35. Node.js. Node.js. URL: https://nodejs.org/uk .

36. The Modern JavaScript. URL: https://javascript.info/ .

37. Sufiyan T. What is node.js: a comprehensive guide. *Simplilearn.com*. URL: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs).

38. Megida D. What is javascript? A definition of the JS programming language. freeCodeCamp.org. URL: https://www.freecodecamp.org/news/what-is-javascript-definition-of-js/

39. What is javascript? A basic introduction to JS for beginners. Hostinger Tutorials. URL: https://www.hostinger.com/tutorials/what-is-javascript).

40. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

41. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
_____ Юрій ЯРЕМЧУК
"____" _____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:
Підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі
08-72.МКР.000.00.000.ТЗ

Керівник магістерської кваліфікаційної роботи
к.т.н., доцент каф. МБІС
_____ Карпінець В.В.

Вінниця – 2024 р.

1. Найменування та область застосування

Підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі**2.Підстава для розробки**

Розробка виконується на основі наказу ректора ВНТУ № 81 від 11. 03. 2024 р.

3. Мета та призначення розробки

3.1 Мета розробки: підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі

3.2 Призначення: підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв

4. Джерела розробки

4.1. Ахрамович В. М. Ідентифікація й аутентифікація, керування доступом // Сучасний захист інформації. – 2016. №4.– С. 47-51.

4.2. Бурячок В.Л. Політика інформаційної безпеки: підручник. / В.Л.Бурячок, Р.В.Грищук, В.О.Хорошко / За заг. ред. докт. техн. наук, проф. В.О. Хорошка. – К.: ПВП «Задруга», 2014. – 222 с.

4.3. Єсін В.І. Безпека інформаційних систем і технологій / В.І.Єсін, О.О. Кузнецов, Л.С. Сорока. – Харків: ХНУ імені В.Н. Каразіна, 2013. – 632 с.

4.4. ZakariaOmar, ZangooeiToomaj, MohdAfiziMohdShukran. Enhancing Mixing Recognition-Based and Recall-Based Approach in Graphical Password Scheme. IJACT, Vol. 4, No. 15, pp. 189-197, 2012.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов'язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1.	Визначення напрямку магістерської роботи, формулювання теми	1.03.2024	15.03.2024	
2.	Аналіз предметної області обраної теми	15.03.2024	1.04.2024	
3.	Розробка роботи	1.04.2024	10.04.2024	
4.	Написання магістерської роботи на основі розробленої теми	10.04.2024	20.04.2024	
5.	Передзахист магістерської кваліфікаційної роботи	20.04.2024	10.05.2024	
6.	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	10.05.2024	4.06.2024	
7.	Захист магістерської кваліфікаційної роботи	10.06.2024	10.06.2024	

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;
– програмний додаток;
– презентація;
– відзив керівника роботи;
– відзив опонента

Технічне завдання до виконання прийняв _____ Гусєв О.О.

Додаток Б. Лістинг програми

```
app.post('/data', (req, res) => {
  const data = req.body;
  // Логування отриманих даних
  console.log('Received data:', data);
  // Відповідь на запит
  res.status(200).send('Data received');
});
const jwt = require('jsonwebtoken');
const secretKey = 'your_secret_key';
// Маршрут для логіну
app.post('/login', (req, res) => {
  const { username, password } = req.body;
  // Псевдокод для верифікації користувача
  if (username === 'user' && password === 'pass') {
    const token = jwt.sign({ username }, secretKey, { expiresIn: '1h' });
    res.json({ token });
  } else {
    res.status(401).send('Invalid credentials');
  }
});
// Middleware для перевірки токена
const authenticate = (req, res, next) => {
  const token = req.header('Authorization').replace('Bearer ', '');
  try {
    const decoded = jwt.verify(token, secretKey);
    req.user = decoded;
    next();
  } catch (e) {
    res.status(401).send('Invalid token');
  }
};
//Захищений маршрут
app.post('/data', authenticate, (req, res) => {
  const data = req.body;
  console.log('Authenticated user data:', data);
  res.status(200).send('Data received');
});
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/iotSecuritySystem', {
  useNewUrlParser: true,
  useUnifiedTopology: true
});
const DataSchema = new mongoose.Schema({
  deviceId: String,
  timestamp: { type: Date, default: Date.now },
  sensorData: mongoose.Schema.Types.Mixed
});
const Data = mongoose.model('Data', DataSchema);
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <DHT.h>
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
const char* serverName = "https://your_server_address/data";
#define DHTPIN D4
#define DHTTYPE DHT11
```

```
DHT dht(DHTPIN, DHTTYPE);
void setup() {
 Serial.begin(115200);
 dht.begin();

 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.println("Connecting to WiFi...");
 }
void loop() {
 float humidity = dht.readHumidity();
 float temperature = dht.readTemperature();
 if (isnan(humidity) || isnan(temperature)) {
  Serial.println("Failed to read from DHT sensor!");
  return;
 }
 Serial.print("Humidity: ");
 Serial.print(humidity);
 Serial.print("%, Temperature: ");
 Serial.print(temperature);
 Serial.println("°C");
 if (WiFi.status() == WL_CONNECTED) {
  HTTPClient http;
  http.begin(serverName);
  http.addHeader("Content-Type", "application/json");

  String postData = "{\"deviceId\":\"NodeMCU_01\",\"timestamp\":\"";
  postData += String(millis());
  postData += "\",\"sensorData\":{\"temperature\":";
  postData += String(temperature);
  postData += ",\"humidity\":";
  postData += String(humidity);
  postData += "}}";
  int httpResponseCode = http.POST(postData);
  if (httpResponseCode > 0) {
   String response = http.getString();
   Serial.println(httpResponseCode);
   Serial.println(response);
  } else {
   Serial.print("Error on sending POST: ");
   Serial.println(httpResponseCode);
  }
  http.end();
 } else {
  Serial.println("Error in WiFi connection");
 }
 delay(60000); // 60 секунд
}
const express = require('express');
const http = require('http');
const WebSocket = require('ws');

const app = express();
const port = 3000;

// Middleware для обробки JSON
app.use(express.json());
```

```
// Створення HTTP сервера
const server = http.createServer(app);

// Створення WebSocket сервера на базі HTTP сервера
const wss = new WebSocket.Server({ server });

// Обробка нових з'єднань WebSocket
wss.on('connection', (ws) => {
  console.log('New client connected');

  // Відправка повідомлення новому клієнту
  ws.send('Welcome new client!');

  // Обробка отриманих повідомлень
  ws.on('message', (message) => {
    console.log('Received message:', message);

    // Відправка отриманого повідомлення всім підключеним клієнтам
    wss.clients.forEach((client) => {
      if (client !== ws && client.readyState === WebSocket.OPEN) {
        client.send(message);
      }
    });
  });

  // Обробка закриття з'єднання
  ws.on('close', () => {
    console.log('Client disconnected');
  });
});

// Запуск сервера
server.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
wifi.setmode(wifi.STATION)
wifi.sta.config({ssid="your_SSID", pwd="your_PASSWORD"})

wifi.sta.connect()
tmr.alarm(1, 1000, 1, function()
  if wifi.sta.getip() == nil then
    print("Connecting to WiFi...")
  else
    print("Connected, IP is "..wifi.sta.getip())
    tmr.stop(1)
  end
end)

local ws = websocket.createClient()
ws:on("connection", function(ws)
  print('Connected to WebSocket server')
  ws:send('Hello Server')
end)
ws:on("receive", function(_, msg, opcode)
  print('Received message:', msg)
end)
ws:on("close", function(_, status)
```

Додаток В. Ілюстративний матеріал

Підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі

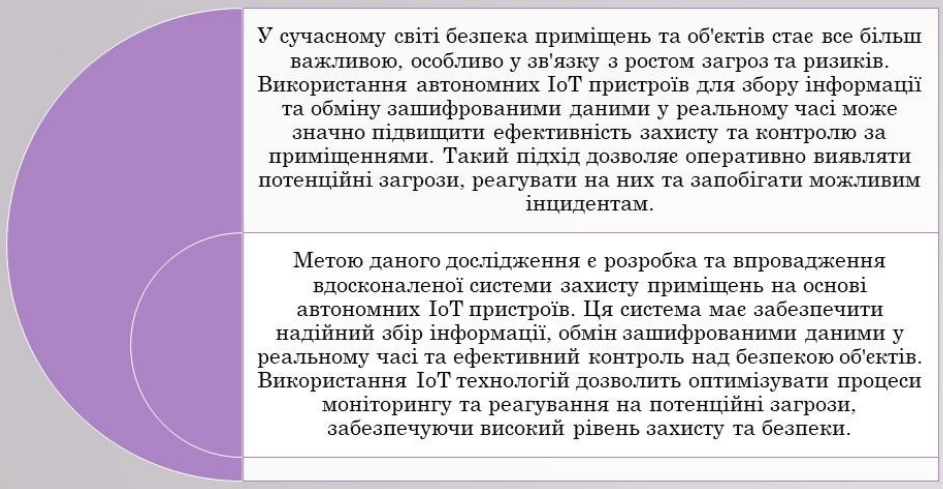
Виконав: ст. 2 курсу, групи КІТС-22 мз Гусєв О.О.

Керівник: к.т.н., доцент. каф. МБІС Карпінець В.В.

ВСТУП

У сучасному світі поняття безпеки стає все більш актуальним, особливо в контексті захисту приміщень та об'єктів від різних загроз. Від використання технологій Інтернету речей (IoT) очікується значний внесок у забезпечення безпеки, зручності та ефективності управління. У цьому контексті виникає необхідність вдосконалення захисту приміщень за допомогою автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі. Це дозволить ефективно контролювати та моніторити об'єкти, запобігати потенційним загрозам та швидко реагувати на події в реальному часі. У даному дослідженні ми дослідимо можливості та переваги використання автономних IoT пристроїв для підвищення безпеки приміщень та об'єктів у сучасному світі.

Актуальність та мета



Основні принципи функціонування IoT пристроїв

- IoT-пристрої (Internet of Things) – це широкий спектр фізичних пристроїв, які оснащені датчиками, програмним забезпеченням та підключені до Інтернету. Ці пристрої можуть збирати дані про навколишнє середовище, взаємодіяти з іншими пристроями та управлятися віддалено.
- Основні принципи функціонування IoT-пристроїв визначаються їх роллю у зборі, передачі, обробці, зберіганні даних, управлінні та взаємодії з іншими пристроями.

– збір даних;

– передача даних;

– обробка даних;

Архітектурні особливості систем IoT

- Архітектурні особливості систем Інтернету речей (IoT) визначаються принципами, за якими організовані та взаємодіють між собою пристрої та компоненти цих систем. Розглянемо детально кожен з аспектів архітектури IoT систем:

  - коннектівність;
  - масштабованість;
  - розподіленість;

Порівняльний аналіз протоколів обміну даними в IoT системах

Критерій	MQTT	CoAP	HTTP	AMQP
Пропускна здатність	Низька	Низька	Висока	Середня
Легкість використання	Висока	Висока	Середня	Низька
Безпека	Середня	Середня	Низька	Висока
Підтримка пристроїв з обмеженими ресурсами	Висока	Висока	Низька	Середня
Надійність	Висока	Висока	Низька	Середня

Схема роботи пристрою захисту приміщення

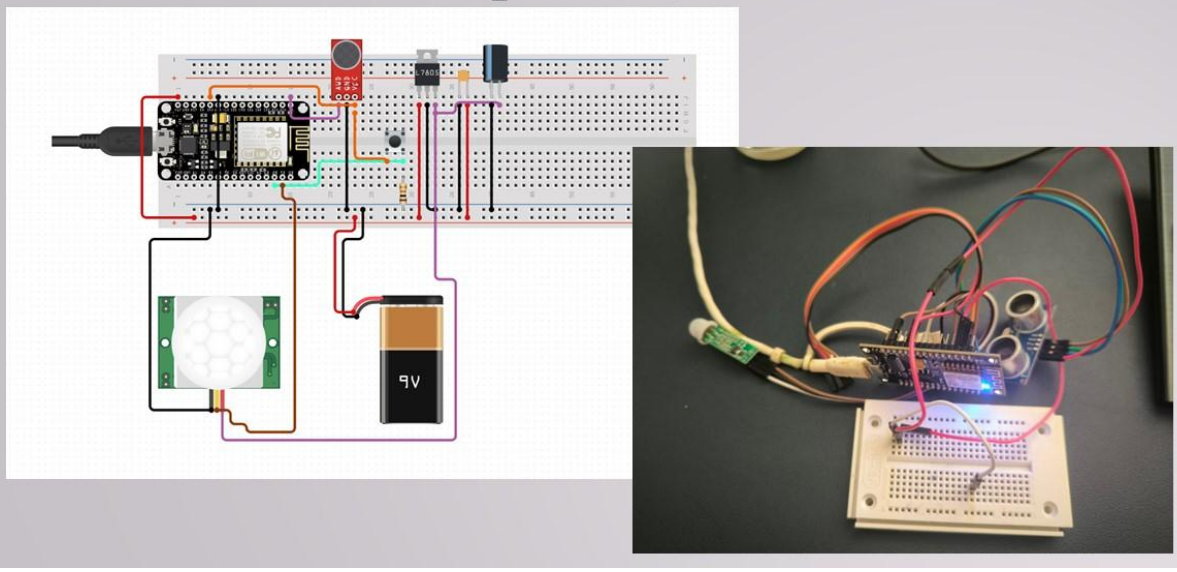
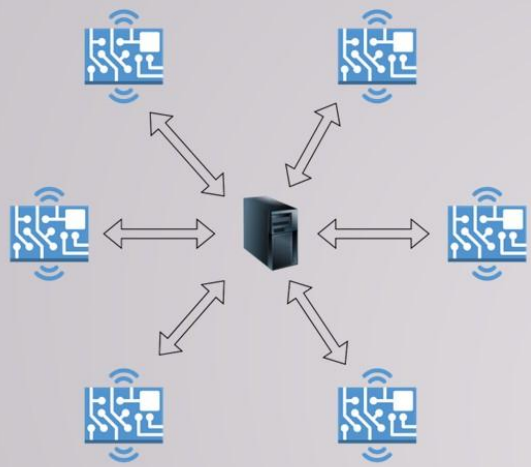
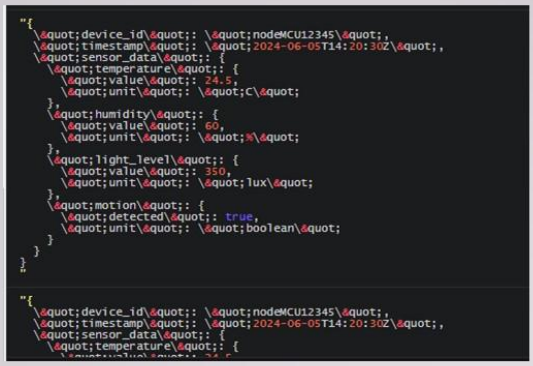


Схема взаємодії між сервером та IoT пристроями



Тестування серверної частини



Тестування апаратного частини

Використані технології

Висновок

- Вдосконалення захищеності приміщень за допомогою автономних IoT пристроїв для збору інформації та обміну зашифрованими даними у реальному часі є важливим кроком у напрямку підвищення безпеки та контролю над об'єктами. Розроблена система дозволяє оперативно виявляти потенційні загрози та ефективно реагувати на них, забезпечуючи надійний захист від різних небезпек.
- Застосування автономних IoT пристроїв дозволяє забезпечити швидкий та точний обмін інформацією між системними компонентами, зменшуючи час реакції на події та забезпечуючи оперативне прийняття відповідних заходів. Крім того, використання зашифрованих даних забезпечує конфіденційність інформації та запобігає несанкціонованому доступу до неї.
- Отже, впровадження вдосконаленої системи на основі автономних IoT пристроїв сприяє підвищенню рівня безпеки та захищеності приміщень, забезпечуючи спокій та безпеку для користувачів та власників об'єктів.

Дякую за увагу!

Додаток Г. Протокол перевірки на антиплагіат

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Підвищення захищеності приміщень вдосконаленою системою на основі автономних IoT пристроїв збору інформації та обміну зашифрованими даними у реальному часі

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 95 % Схожість 5 %

Аналіз звіту подібності (відмітити потрібне):

1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Коваль Н.П. (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Гусєв О.О. (прізвище, ініціали)

Керівник роботи _____ (підпис) Карпінець В.В. (прізвище, ініціали)