МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ
ТА БІОТЕХНОЛОГІЙ ІМЕНІ С.З.ҐЖИЦЬКОГО

Факультет механіки, енергетики та інформаційних технологій
Кафедра інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **«Розробка системи охорони приміщення із використанням технологій IoT»**

Виконав: студент 4 курсу групи Кн-41

Спеціальності
122 «Комп'ютерні науки»
(шифр і назва)

Хомин Зиновій Романович
(прізвище і ініціали)

Керівник: к. е. н., доц. Шувар Б. І.
(прізвище і ініціали)

Рецензент: к.т.н., доцент Гуменюк Р.В.
(прізвище і ініціали)

ДУБЛЯНИ-2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМЕНІ С.З.ҐЖИЦЬКОГО

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 122 «Комп'ютерні науки»

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____
_____
«____» _____ 202_ р.

ЗАВДАННЯ
на кваліфікаційну роботу студенту

Хомина Зиновія Романовича
(прізвище, ім'я, по батькові)

1. Тема роботи: *«Розробка системи охорони приміщення із використанням технологій IoT»*
2. Керівник роботи *к.е.н., доцент, Шувар Богдан Іванович*
(наук.ступінь, вч. звання, прізвище, ініціали)

затверджені наказом Львівського НУП №123/К-с від 25.02.2025р.
Строк подання студентом роботи 09.06.2025р.
Вихідні дані до роботи: набір апаратних компонентів (ESP32, датчики, реле), конфігураційні параметри (дані для підключення до Wi-Fi, ThingsBoard, smtp) та визначена логіка роботи системи, що описує реакцію на спрацювання датчиків

3. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ, 1. Аналіз предметної області , 2. Проєктування системи охорони приміщення, з. *Реалізація системи охорони, 4. Оцінка ефективності та перспективи розвитку, 5. Охорона праці, Висновки, Бібліографічний список*
4. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових схем та моделей): *Кількість активних IoT та не-IoT підключень у світі; ESP32-based та Arduino Mega плати; Сенсори у системах охорони; Модулі передачі даних; Система охорони приміщення (IoT-орієнтована);ThingsBoard , розгорнутий у Docker на Windows; Веб-інтерфейс платформи ThingsBoard; Тестовий стенд ESP32 та датчиків; Уривок коду для ESP32 у середовищі Arduino IDE; Створення "Пристрою" (Device) в ThingsBoard; Місце генерації токена доступу для створеного пристрою; Розділ "Dashboards" (Дашборди); Ланцюжки правил; Тестування системи оповіщення через e-mail; Схема мережі; Критичні вразливості системи.*

5. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	*Шувар Б. І., доцент кафедри інформаційних технологій*		
5	*Городецький І. М., доцент кафедри інженерної механіки*		

6. Дата видачі завдання 26.02.2025 р.

Календарний план

№ з/п	Назва етапів дипломного проєкту	Терміни виконання етапів роботи	Примітка
1.	*Написання першого розділу*	27.02.2025 – 15.03.2025	
2.	*Виконання другого розділу та аркушів ілюстраційного матеріалу до нього*	15.03.2025 – 02.04.2025	
3.	*Виконання третього, четвертого розділів та аркушів ілюстраційного матеріалу до нього*	03.04.2025 – 25.04.2025	
4.	*Написання розділу «Охорона праці»*	26.04.2025 – 28.04.2025	
5.	*Завершення оформлення розрахункового-пояснювальної записки та аркушів ілюстраційного матеріалу*	29.04.2025 – 29.05.2025	
6.	*Завершення роботи цілому*	30.05.2025 – 07.06.2025	

Студент _____, Хомин З.Р.
(підпис) (прізвище, ініціали)

Керівник роботи _____, Шувар Б.І.
(підпис) (прізвище, ініціали)

УДК 004.78

Розробка системи охорони приміщення із використанням технологій IoT.

Хомин З.Р. Кафедра ІТ. Дубляни, ЛНУВМБ ім. С.З.Гжицького, 2025.

Кваліфікаційна робота: 66 ст. текст. част., 21 рис., 8 табл., 29 літ. джерел.

У кваліфікаційній роботі описано розроблену IoT-систему моніторингу та безпеки приміщень. Система призначена для автоматизованого контролю за станом об'єкта та оперативного сповіщення про надзвичайні події. Для її реалізації були використані сучасні технології Інтернету речей, включаючи мікроконтролери ESP32 для збору даних з різноманітних датчиків (руху, відкриття дверей/вікон, температури/вологості, диму/газу) та IoT-платформу ThingsBoard як центральний елемент для обробки, візуалізації та керування даними.

Розроблена система забезпечує функціонал моніторингу в реальному часі, гнучку обробку подій за допомогою ThingsBoard Rule Engine, а також миттєве сповіщення користувачів через інтеграцію з електронною поштою.

В роботі приділено значну увагу питанням кібербезпеки, що включає використання захищених протоколів зв'язку та механізмів автентифікації. Проведено аналіз технічної та економічної ефективності рішення, а також його порівняння з існуючими аналогами, що підтвердило високу конкурентоспроможність системи у забезпеченні безпеки приміщень.

Ключові слова: Інтернет речей, Система моніторингу, Система безпеки, ThingsBoard, ESP32, Rule Engine, Датчики, Автоматизація, Розумний будинок, Телеметрія.

Key word: Internet of Things, Monitoring System, Security System, Sensors, Automation, Smart Home, Telemetry.

ЗМІСТ

Вступ

У сучасному світі питання безпеки житлових і комерційних приміщень набуває дедалі більшої актуальності. Зростання рівня злочинності, потреба в контролі доступу, а також бажання користувачів мати змогу віддалено керувати системами безпеки стимулюють розвиток новітніх технологій у цій сфері. Одним із найперспективніших напрямів є використання технологій Інтернету речей (IoT — Internet of Things), які дозволяють створювати інтелектуальні, адаптивні та взаємопов'язані системи охорони.

IoT відкриває нові можливості для автоматизації процесів моніторингу, виявлення загроз та оперативного реагування на них. Завдяки використанню датчиків руху, камер відеоспостереження, модулів зв'язку та хмарних сервісів, можна створити ефективну систему охорони, яка працює в режимі реального часу та забезпечує високий рівень безпеки.

Метою даної роботи є розробка прототипу системи охорони приміщення з використанням технологій IoT, яка дозволяє виявляти несанкціонований доступ, надсилати сповіщення користувачу та забезпечувати базову аналітику подій.

Для досягнення поставленої мети необхідно вирішити такі завдання.

Проаналізувати сучасний стан розвитку IoT-систем моніторингу та безпеки приміщень, визначити ключові технології, компоненти та архітектурні підходи, що використовуються в даній сфері.

Розробити архітектуру IoT-системи моніторингу та безпеки, обґрунтувати вибір апаратних засобів (мікроконтролера ESP32, сенсорів) та програмної платформи (ThingsBoard) для реалізації системи.

Об'єкт дослідження: Процеси моніторингу та забезпечення безпеки приміщень за допомогою інформаційних систем та технологій.

Предмет дослідження: Методи та засоби розробки IoT-системи для автоматизованого збору, обробки та візуалізації даних, а також оперативного реагування на події безпеки.

Практична цінність роботи полягає у створенні доступного та масштабованого рішення, яке може бути використане як у побуті, так і в малому бізнесі.

РОЗДІЛ 1.
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Поняття та принципи роботи IoT (Internet of Things)

Інтернет речей (Internet of Things, IoT) — це концепція, що передбачає інтеграцію фізичних об’єктів у єдину інформаційно-комунікаційну інфраструктуру з метою забезпечення автоматизованого збору, обміну та аналізу даних. Основною ідеєю IoT є надання можливості об’єктам навколишнього середовища — таким як сенсори, виконавчі пристрої, побутова техніка, транспортні засоби тощо — взаємодіяти між собою та з користувачем через мережу Інтернет без безпосереднього втручання людини [2].

Функціонування IoT-систем базується на кількох ключових принципах. По-перше, це сенсорна взаємодія з фізичним середовищем, яка реалізується за допомогою різноманітних датчиків, що фіксують параметри середовища (температуру, вологість, рух, освітленість тощо). По-друге, передача даних здійснюється через бездротові або дротові канали зв’язку з використанням спеціалізованих протоколів, таких як MQTT, CoAP, HTTP. По-третє, обробка інформації може відбуватися як на рівні периферійних пристроїв (edge computing), так і в хмарних обчислювальних середовищах (cloud computing), що дозволяє реалізувати адаптивні алгоритми реагування на події в реальному часі [1].

Згідно з дослідженнями, архітектура типових IoT-систем включає чотири рівні: рівень сприйняття (сенсори та пристрої збору даних), рівень передачі (мережеві інтерфейси), рівень обробки (сервери, хмарні платформи) та рівень застосування (інтерфейси користувача, аналітичні модулі). Така багаторівнева структура забезпечує масштабованість, гнучкість та адаптивність системи до змін у середовищі функціонування.

Особливу увагу в сучасних дослідженнях приділяють питанням енергоефективності, безпеки даних та стандартизації протоколів взаємодії між

пристроями. Як зазначено у праці Колесника та Коземчука, ефективне проектування IoT-систем вимагає врахування обмежень апаратної платформи, вибору оптимальних засобів зв'язку та забезпечення захисту інформації на всіх етапах її обробки.

Інтернет речей постійно розвивається, і теперішні тенденції показують, що сфера IoT буде зростати. Переплітаючись з багатьма технологіями, Інтернет речей залишається окремою областю IT-індустрії та сьогодні розвивається вражаючими темпами. У той час як кількість підключених пристроїв, що не є IoT, збільшується незначно, кількість пристроїв Інтернету речей останніми роками стрімко збільшується (рис. 1.1) [20].

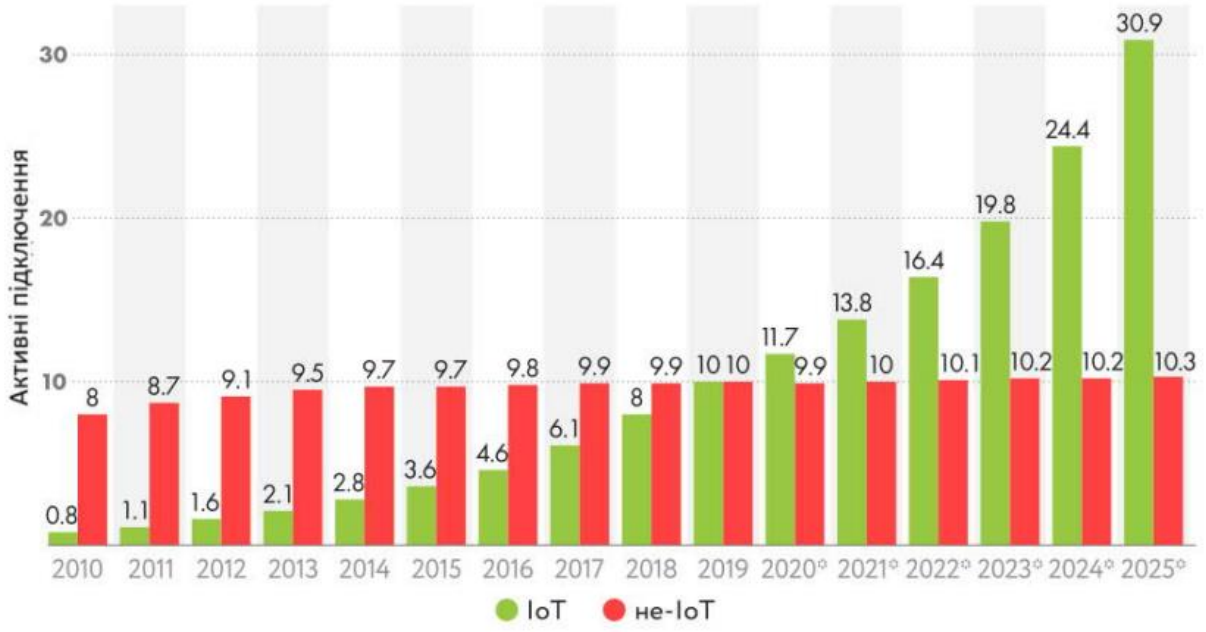


Рисунок 1.1 — Кількість активних IoT та не-IoT підключень у світі з 2010 по 2025 роки

Таким чином, Інтернет речей є ключовою технологією цифрової трансформації, яка дозволяє створювати інтелектуальні системи моніторингу, управління та автоматизації в різних сферах — від побуту до промисловості.

1.2. Сучасні загрози безпеці приміщень

У сучасних умовах безпекове середовище зазнає суттєвих трансформацій під впливом глобалізаційних, технологічних, соціальних та

політичних чинників. Безпека приміщень — як житлових, так і комерційних — більше не обмежується лише фізичним захистом від несанкціонованого доступу. Вона охоплює широкий спектр загроз, включаючи кібернетичні атаки на системи охорони, саботаж, соціальну інженерію, а також ризики, пов'язані з нестабільністю в державі та регіоні.

Однією з ключових тенденцій є зростання гібридних загроз, які поєднують фізичні та інформаційні атаки. Як зазначає Р. Шай, глобалізаційні процеси спричиняють появу нових форм загроз, що підривають стабільність національних безпекових систем, зокрема через асиметричні виклики, які важко ідентифікувати та нейтралізувати традиційними методами [10]. У цьому контексті приміщення можуть стати об'єктами як прямого фізичного впливу (вторгнення, крадіжки, вандалізм), так і опосередкованого — через виведення з ладу систем відеоспостереження, сигналізації або контролю доступу.

Технологічні загрози також набувають дедалі більшої ваги. Багато систем охорони, особливо в малому бізнесі та приватному секторі, використовують застаріле або недостатньо захищене обладнання. Відсутність оновлень програмного забезпечення, слабке шифрування каналів зв'язку, використання стандартних паролів — усе це створює вразливості, які можуть бути використані зловмисниками для дистанційного доступу до системи [9].

Соціально-економічні чинники, зокрема зростання рівня безробіття, внутрішньо переміщені особи, а також наслідки воєнного конфлікту, створюють додаткові ризики для безпеки об'єктів нерухомості. Як зазначає М. Кристиняк, сучасна система безпеки в Україні потребує глибокої модернізації, оскільки існуючі механізми не здатні ефективно протидіяти новим формам загроз [7].

Крім того, людський фактор залишається одним із найслабших елементів у системі безпеки. Недостатня обізнаність користувачів щодо правил кібергігієни, нехтування базовими заходами захисту (наприклад, залишення ключів у легкодоступних місцях, відсутність резервного живлення

для систем охорони) значно знижують ефективність навіть найсучасніших технічних рішень.

У відповідь на ці виклики, сучасні системи охорони повинні бути не лише технічно досконалими, а й адаптивними, інтегрованими та інтелектуальними. Впровадження технологій Інтернету речей (IoT), штучного інтелекту та хмарних обчислень дозволяє створювати системи, здатні до самонавчання, прогнозування загроз та автономного реагування на інциденти.

Таким чином, сучасні загрози безпеці приміщень є багатовимірними та вимагають комплексного підходу до їх ідентифікації, аналізу та нейтралізації. Ефективна система охорони повинна враховувати не лише технічні аспекти, а й соціальні, правові та організаційні чинники.

1.3. Огляд систем охорони

Системи охорони приміщень еволюціонували від простих механічних засобів захисту до складних інтегрованих рішень, що поєднують апаратні та програмні компоненти, штучний інтелект, хмарні технології та Інтернет речей (IoT). У сучасних умовах ефективна система охорони повинна забезпечувати не лише виявлення загроз, а й їхню превенцію, аналіз та адаптивне реагування.

Класифікація систем охорони:

1. Пасивні системи — механічні замки, решітки, сейфи.

2. Активні автономні системи — сигналізація, відеоспостереження без підключення до мережі.

3. Інтелектуальні мережеві системи — інтегровані рішення з IoT, аналітикою, мобільним керуванням.

Згідно з дослідженням Іванюка та ін., сучасні системи охорони дедалі частіше базуються на відкритих апаратно-програмних платформах, таких як Arduino або Raspberry Pi, що дозволяє створювати гнучкі та масштабовані рішення [6]. Водночас, як зазначає Шутий, більшість комерційних систем

охорони в Україні все ще використовують закриті протоколи та обмежені функціональні можливості (табл. 1.1) [11].

Таблиця 1.1 - Порівняльна таблиця найбільш поширених систем охорони*

Система	**Тип**	**Технології**	**Переваги**	**Недоліки**	**Приклади використання**
Ajax Systems	Комерційна	Радіозв'язок Jeweller, мобільний застосунок	Висока надійність, автономність, простота встановлення	Висока вартість, обмежена кастомізація	Квартири, офіси, приватні будинки
Hikvision	Комерційна	Відеоаналітика, розпізнавання облич	Потужна відеоаналітика, інтеграція з СКД	Високі вимоги до мережі, складність налаштування	Бізнес-центри, ТРЦ
Open-source IoT-системи	Відкрита	Arduino, ESP32, MQTT, Node-RED	Гнучкість, низька вартість, можливість кастомізації	Потреба в технічних знаннях, менша надійність	DIY-проєкти, навчальні заклади
Системи СКД (контролю доступу)	Комерційна	RFID, NFC, біометрія	Контроль персоналу, інтеграція з ERP	Не виявляє вторгнення, лише обмежує доступ	Офіси, склади, підприємства
Хмарні системи відеоспостереження	Комерційна / гібридна	IP-камери, хмара, мобільний доступ	Віддалений доступ, зберігання в хмарі	Залежність від інтернету, питання конфіденційності	Роздрібна торгівля, житлові комплекси

*Джерело: [12, 22, 23, 26, 15]

Отже, сучасні системи охорони демонструють тенденцію до інтелектуалізації, децентралізації та персоналізації, що дозволяє адаптувати їх до конкретних потреб користувача. Водночас, вибір системи має базуватись на аналізі ризиків, бюджету, технічних можливостей та рівня загроз.

З метою об'єктивного аналізу ефективності різних типів систем охорони було здійснено порівняльне оцінювання восьми поширених рішень, які активно застосовуються в Україні та за її межами. Для цього використано п'ять ключових критеріїв: надійність, гнучкість, вартість, простота

встановлення та кібербезпека. Оцінювання здійснювалося за п’ятибальною шкалою, де 5 — найвищий рівень відповідності критерію, а 1 — найнижчий (табл. 1.2).

Таблиця 1.2 - Порівняльна таблиця систем охорони за ключовими критеріями (оцінки від 1 до 5)

Система охорони	**Надійність**	**Гнучкість**	**Вартість**	**Простота встановлення**	**Кібербезпека**
Ajax Systems	5	3	2	5	4
Hikvision	5	2	3	3	3
Open-source IoT-системи	3	5	5	2	2
СКД (контроль доступу)	4	2	3	4	3
Хмарні системи відеоспостереження	4	4	3	4	2
Dahua	4	3	3	3	3
Jablotron	5	3	2	4	4
Satel	4	3	3	3	4

*Джерело: [11, 12, 22, 23, 26, 15]

Надійність відображає стабільність роботи системи в умовах зовнішніх впливів, зокрема перебоїв живлення, втрати зв’язку або спроб саботажу. Найвищі оцінки за цим критерієм отримали Ajax Systems, Hikvision та Jablotron, які демонструють високу відмовостійкість і мають резервні канали зв’язку.

Гнучкість характеризує здатність системи до масштабування, адаптації до різних сценаріїв використання та інтеграції з іншими платформами. Найвищу оцінку отримали open-source IoT-системи, які дозволяють глибоку кастомізацію, однак потребують технічної підготовки користувача.

Вартість враховує не лише ціну обладнання, а й витрати на встановлення, обслуговування та оновлення. Найдоступнішими виявилися

open-source рішення, тоді як Ajax Systems та Jablotron мають вищу вартість, що компенсується високою якістю.

Простота встановлення є важливою для кінцевого користувача. Ajax Systems, Jablotron та хмарні рішення демонструють найкращі показники завдяки модульній структурі та підтримці мобільних застосунків.

Кібербезпека оцінює рівень захисту даних, шифрування, оновлення ПЗ та стійкість до атак. Найвищі оцінки отримали Ajax, Jablotron та Satel, які впроваджують сучасні протоколи безпеки.

Таким чином, результати порівняльного аналізу свідчать про те, що не існує універсального рішення, яке б одночасно задовольняло всі критерії на найвищому рівні. Вибір системи охорони має базуватися на пріоритетах користувача, специфіці об’єкта та доступному бюджеті. Інтеграція IoT-технологій відкриває нові можливості для створення адаптивних, масштабованих і безпечних систем охорони, що є перспективним напрямом подальших досліджень.

1.4. Апаратні і програмні засоби для реалізації IoT-систем

Розробка сучасних систем охорони приміщень із використанням технологій Інтернету речей (IoT) вимагає комплексного підходу до вибору апаратних і програмних засобів. Від правильного поєднання цих компонентів залежить надійність, масштабованість, енергоефективність і безпека всієї системи.

Основу IoT-системи становить мікроконтролерна платформа, яка забезпечує обробку сигналів від сенсорів, керування виконавчими пристроями та передачу даних. Найбільш поширеними є:

- ESP32 / ESP8266 — мікроконтролери з вбудованим Wi-Fi та Bluetooth, що забезпечують високу продуктивність при низькому енергоспоживанні (рис. 1.2);

Рисунок 1.2. - ESP32-based та Arduino Mega плати

- Arduino Mega / Uno — прості у використанні платформи з великою спільнотою підтримки;

- Raspberry Pi — мікрокомп'ютер, що дозволяє реалізовувати складні логічні алгоритми, обробку відео та локальну аналітику.

У якості сенсорів (рис. 1.3) у системах охорони найчастіше використовуються:

- PIR-датчики руху (наприклад, HC-SR501);
- датчики відкриття дверей/вікон (на основі герконів);
- ультразвукові датчики відстані (HC-SR04);
- газові та димові сенсори (MQ-2, MQ-135) (рис. 1.3).

а) PIR-датчики руху б) датчик відкриття дверей в) ультразвукові датчики відстані г) димовий сенсор

Рисунок 1.3. - Сенсори у системах охорони

Для передачі даних застосовуються модулі (рис. 1.4):

- GSM/GPRS (SIM800L) — для надсилання SMS або дзвінків у разі тривоги;

- LoRa SX1278 — для зв’язку на великі відстані з низьким енергоспоживанням;
- Wi-Fi (ESP8266/ESP32) — для інтеграції з локальною мережею або хмарними сервісами.

Рисунок 1.4. - Модулі передачі даних (SIM800L, LoRa SX1278, ESP32)

Програмне забезпечення виконує функції збору, обробки, передачі та візуалізації даних. У роботі Кузька А. Г. [8] проаналізовано архітектуру універсальної IoT-платформи з інтеграцією в хмарне середовище Tuya Smart. Автор запропонував використання Arduino IDE як основного середовища розробки, а також описав принципи побудови програмної логіки для взаємодії з хмарними сервісами.

Основні компоненти програмного забезпечення:

- Мови програмування: C/C++ (Arduino), Python (Raspberry Pi), JavaScript (Node.js);

- Середовища розробки: Arduino IDE, PlatformIO, Node-RED;

- Протоколи зв’язку: MQTT, HTTP/HTTPS, CoAP;

- Хмарні сервіси: Blynk, Firebase, AWS IoT, Tuya Smart.

Забезпечення кібербезпеки є критично важливим аспектом IoT-систем. До основних заходів належать:

- Шифрування даних (TLS/SSL);

- Аутентифікація пристроїв;

- Регулярне оновлення прошивки;

- Використання VPN або захищених каналів зв’язку.

Для передачі даних між пристроями та хмарними сервісами використовуються протоколи:

- MQTT — легкий брокерський протокол, оптимізований для IoT;
- HTTP/HTTPS — для RESTful API-запитів;
- CoAP — UDP-орієнтований протокол для обмежених пристроїв.

Хмарні платформи, які забезпечують зберігання, візуалізацію та аналітику даних:

- ThingSpeak — підтримує MATLAB-аналітику [28];
- Blynk — дозволяє створювати мобільні інтерфейси керування [14];
- Tuya Smart — комерційна платформа з підтримкою голосових асистентів [29];
- ThingsBoard — багатофункціональна Open Source платформа, що забезпечує керування пристроями, збір та зберігання телеметрії, потужний Rule Engine для обробки даних та генерації подій, а також гнучкі та кастомізовані дашборди для візуалізації та аналітики [27].

Узагальнюючи, можна стверджувати, що ефективна реалізація IoT-системи охорони вимагає поєднання апаратної надійності, програмної гнучкості та інформаційної безпеки, що дозволяє створювати адаптивні рішення, здатні до самостійного реагування на загрози.

РОЗДІЛ 2.
ПРОЄКТУВАННЯ СИСТЕМИ ОХОРОНИ ПРИМІЩЕННЯ

2.1. Визначення вимог до системи

Етап визначення вимог до системи є критично важливим для успішної реалізації будь-якого проєкту, зокрема й розробки IoT-системи моніторингу та безпеки приміщень. Функціональна модель системи, яка лягла в основу цих вимог, представлена на рис. 2.1.

Функціональні вимоги визначали основні дії та можливості, які система має виконувати для досягнення поставленої мети:

1. Система спроектована для безперервного та автоматизованого відстеження фізичних параметрів та подій у контрольованому просторі. Це включає інтеграцію різних типів датчиків: руху (для фіксації переміщень), відкриття/закриття дверей та вікон (для реєстрації несанкціонованого доступу), диму та газу (для оперативного виявлення ознак пожежі або витоку небезпечних речовин), а також температури та вологості (для контролю мікроклімату). Первинна обробка зібраних аналогових сигналів та їхня трансформація у цифрові дані передбачено реалізувати безпосередньо на рівні мікроконтролерів ESP32.

2. Вимога щодо забезпечення надійної та своєчасної передачі зібраної телеметрії від IoT-пристроїв до центральної IoT-платформи. Для гарантування конфіденційності та цілісності даних було визначено використання захищеного протоколу MQTT over TLS.

3. На IoT-платформі (ThingsBoard) заплановано реалізувати механізми для централізованого прийому, зберігання та обробки телеметрії. Ключовим елементом визначено Rule Engine, який забезпечить налаштування гнучких правил для виявлення аномалій та генерації тривог (наприклад, "рух при активованій охороні" або "критична концентрація диму").

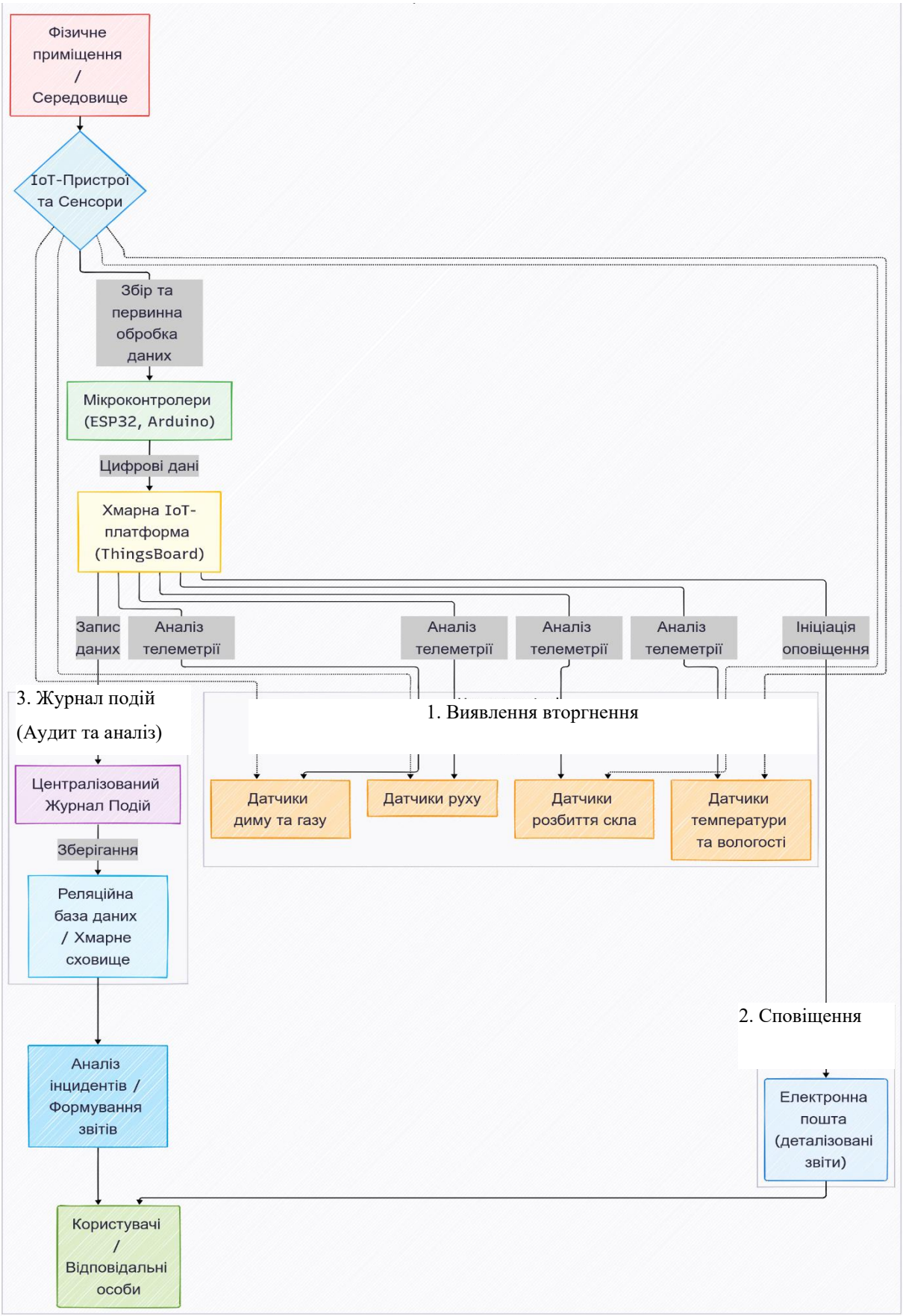


Рисунок 2.1. Система охорони приміщення (IoT-орієнтована)

4. У разі виявлення будь-яких аномальних ситуацій, система негайно ініціює процедуру сповіщення. Основним каналом комунікації було обрано електронну пошту через її надійність та можливість надсилання деталізованих звітів.

5. Створення інтуїтивно зрозумілого веб-інтерфейсу (дашбордів) для відображення поточного стану всіх підключених датчиків та пристроїв, а також для візуалізації історичних даних та графіків.

6. Функціонал системи включатиме можливість дистанційної постановки та зняття з охорони через веб-інтерфейс, що забезпечує зручність для користувачів.

Нефункціональні вимоги визначали якісні характеристики системи та її поведінку, що є не менш важливим для її ефективності та прийнятності:

1. Система повинна демонструвати стійкість до тимчасових збоїв зв'язку, забезпечувати цілісність та коректність зібраних даних, а також оперативність реагування.

2. Обов'язкову автентифікацію IoT-пристроїв (шляхом використання Access Tokens), шифрування всіх переданих даних за допомогою TLS/SSL, реалізацію різних рівнів доступу для користувачів, а також комплексний захист серверної інфраструктури (включаючи ThingsBoard) від несанкціонованого доступу та кібератак.

3. Архітектура системи повинна була забезпечити можливість легкого додавання нових IoT-пристроїв та розширення моніторингових точок без суттєвих змін у базовій кодовій базі. Також платформа повинна бути здатною ефективно обробляти зростаючі обсяги даних та кількість підключених пристроїв.

4. Передбачено, що система повинна бути адаптивною до майбутніх змін, дозволяючи легку інтеграцію нових типів датчиків, додавання альтернативних каналів сповіщення та модифікацію логіки обробки подій без значних зусиль.

5. Інтерфейс моніторингу та керування повинен бути інтуїтивно зрозумілим для кінцевого користувача, а процес налаштування та конфігурації системи – максимально спрощеним.

6. Мінімізація вартості апаратного забезпечення для одного IoT-вузла та використання Open Source програмного забезпечення для скорочення ліцензійних витрат, що забезпечує швидку окупність інвестицій.

Визначені функціональні та нефункціональні вимоги сформували вичерпний базис для подальшого проектування та послідовної реалізації усіх компонентів системи. Це забезпечило розробку рішення, яке відповідає як заявленій меті дослідження, так і практичним потребам щодо ефективного моніторингу та забезпечення безпеки приміщень.

2.2. Архітектура системи охорони

Розробка архітектури системи охорони приміщення з використанням технологій IoT передбачає визначення основних її компонентів, їхніх ролей, взаємозв'язків та протоколів взаємодії. Ефективна архітектура забезпечує надійність, масштабованість, безпеку та гнучкість системи [3 с. 87]. Запропонована архітектура базується на багаторівневому підході, що дозволяє чітко розмежувати відповідальність компонентів та забезпечити їхню автономну роботу.

Систему можна умовно поділити на три основні рівні (шари), що відповідають згальноприйнятим моделям архітектури IoT [5, с. 56] (рис. 2.2).

Рівень периферійних пристроїв (Edge/Perception Layer), який відповідає за збір даних з навколишнього середовища та первинну взаємодію з об'єктом охорони.

Мережевий рівень (Network/Communication Layer), що забезпечує передачу даних між периферійними пристроями та центральним сервером або хмарною платформою.

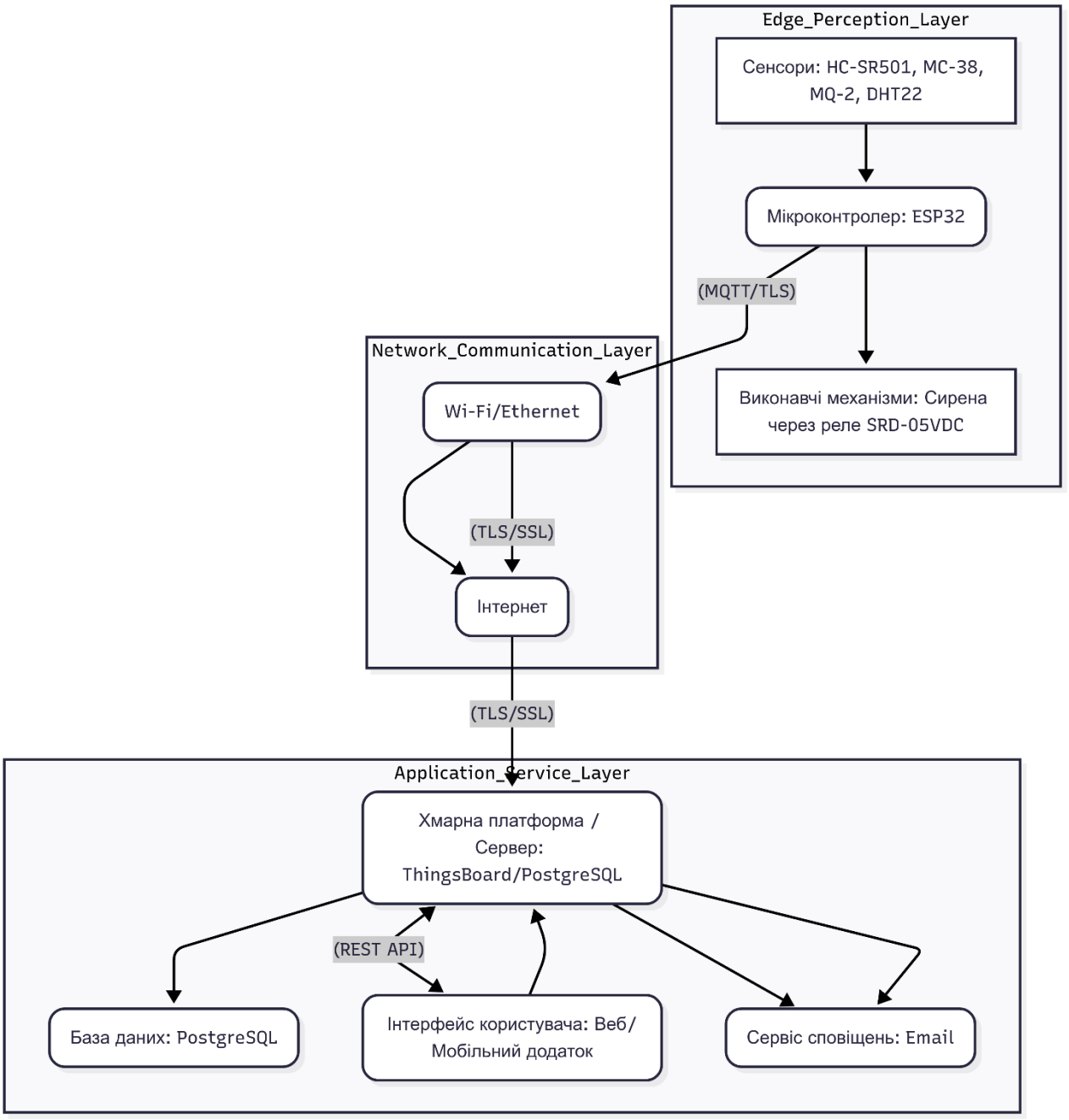


Рисунок 2.2 - Рівні системи охорони

Рівень сервісів та додатків (Application/Service Layer), що відповідає за зберігання, обробку та аналіз даних, реалізацію бізнес-логіки системи, а також надання інтерфейсів для взаємодії з користувачем.

2.3. Вибір апаратного забезпечення

Ефективність та надійність системи охорони приміщення значною мірою залежать від правильного вибору апаратного забезпечення. Вибір компонентів здійснювався на основі критеріїв функціональності, енергоспоживання, вартості, доступності, простоти інтеграції, а також підтримки технологій Інтернету речей (IoT) [3, с. 120].

Мікроконтролер є "мозком" периферійного рівня системи, відповідальним за збір даних з датчиків, їхню первинну обробку та передачу до мережевого рівня, а також за керування виконавчими механізмами. Для цього проекту було розглянуто кілька платформ, зокрема Arduino та ESP-серії (ESP8266, ESP32).

Arduino (наприклад, Arduino Uno/Mega). Характеризується простотою у використанні, великою спільнотою та широким вибором шилдів. Однак, базові моделі Arduino не мають вбудованого Wi-Fi модуля, що вимагає додаткових витрат на Wi-Fi шилд (наприклад, ESP8266 як окремий модуль або ESP-01), збільшуючи складність та енергоспоживання. Продуктивність також може бути обмеженою для складних завдань обробки даних та одночасної комунікації.

ESP8266 (наприклад, NodeMCU ESP-12E). Це мікроконтролер зі вбудованим Wi-Fi модулем, що робить його дуже привабливим для IoT-проектів. Він компактний, енергоефективний та має достатньо GPIO пінів для більшості простих завдань. Проте, його одноядерний процесор може бути менш ефективним при виконанні одночасно завдань збору даних, їхньої обробки та підтримки стабільного мережевого з'єднання, особливо з використанням зашифрованих протоколів (TLS).

ESP32 (наприклад, ESP32 DevKitC V4). Цей мікроконтролер є більш потужним порівняно з ESP8266, завдяки наявності двоядерного процесора Tensilica Xtensa LX6 з тактовою частотою до 240 МГц. Він має вбудовані модулі Wi-Fi (802.11 b/g/n) та Bluetooth (v4.2 BR/EDR and BLE), що розширює комунікаційні можливості [16, с. 112]. Більший обсяг SRAM (520 КБ) та Flash-

пам'яті (4 МБ і більше) забезпечує достатні ресурси для складнішого програмного забезпечення, включаючи реалізацію алгоритмів фільтрації даних, багатопоточності та підтримку криптографічних протоколів (TLS/SSL) без значного навантаження на систему.

З огляду на вимоги до продуктивності, універсальності зв'язку, можливості одночасного виконання завдань (збір даних, мережева комунікація, обробка подій) та підтримки безпечних протоколів, мікроконтролер ESP32 був обраний як оптимальне рішення. Його двоядерна архітектура дозволяє виділити одне ядро для Wi-Fi та протоколів зв'язку, а інше – для обробки даних з датчиків, що значно підвищує стабільність та надійність системи.

Для забезпечення комплексної охорони приміщення було обрано набір датчиків, які покривають основні типи загроз: вторгнення, пожежа, витік газу.

Датчик руху (HC-SR501) для виявлення руху людини або інших теплових об'єктів у зоні контролю. Функціонал простий - пасивний інфрачервоний (PIR) сенсор, який реагує на зміну теплового фону в його полі зору. Має регульовану чутливість (дальність до 7 метрів) та час затримки (від 3 секунд до 5 хвилин), що дозволяє налаштувати його під конкретні умови приміщення та мінімізувати хибні спрацювання [21, с. 55].

Герконовий датчик відкриття/закриття (MC-38) для моніторингу стану дверей, вікон або інших об'єктів на відкриття/закриття. Він складається з двох частин – магніту та герконового реле. При наближенні магніту геркон замикається (або розмикається, залежно від типу), змінюючи стан електричного кола. Встановлюється на раму та рухому частину об'єкта [4, с.34].

Датчик диму та газу (MQ-2) для виявлення горючих газів (метан, пропан, бутан, водень), алкоголю та диму. Він використовує нагрівальний елемент та чутливий шар з діоксиду олова (SnO2), опір якого змінюється у присутності певних газів. Має як аналоговий, так і цифровий вихід (з компаратором для встановлення порогу спрацювання) [18, с. 91].

Датчик температури та вологості (DHT22) для моніторингу кліматичних параметрів у приміщенні. Може бути використаний як додатковий індикатор пожежної безпеки (різке підвищення температури). Принцип роботи - цифровий сенсор, який передає дані по однопровідному протоколу. DHT22 (AM2302) є більш точним та надійним, ніж DHT11, з діапазоном вимірювання температури від -40 до +80 °C (точність ±0.5 °C) та вологості від 0 до 100% (точність ±2-5%) [13, с. 67].

Для забезпечення комунікації між рівнями системи та віддаленого доступу користувачів, ключовим є бездротовий зв'язок. Обираємо Wi-Fi модуль (вбудований в ESP32), що забезпечить підключення мікроконтролера ESP32 до локальної мережі (через Wi-Fi маршрутизатор) та, відповідно, до Інтернету [16, с. 112].

Протокол MQTT (Message Queuing Telemetry Transport) є легковаговий для обміну повідомленнями для IoT-пристроїв. Базується на моделі "публікація/підписка", де пристрої (паблішери) відправляють повідомлення на центральний сервер (брокер), а додатки (сабскрайбери) отримують ці повідомлення, підписавшись на певні "топіки". Це забезпечує асинхронну передачу даних та мінімізує навантаження на мережу [24, с. 72].

Протокол TLS/SSL для забезпечення безпечного та зашифрованого з'єднання між ESP32 та MQTT-брокером/хмарною платформою. Він забезпечує аутентифікацію серверів (і, за бажанням, клієнтів), шифрування даних та перевірку їхньої цілісності, захищаючи від перехоплення та підробки [25, с. 88].

2.4. Вибір програмного забезпечення та середовища розробки

Ефективність та продуктивність розробки системи охорони приміщення із використанням технологій IoT значною мірою залежать від ретельного вибору програмного забезпечення та середовищ розробки. Цей вибір визначається вимогами до функціональності системи, типом апаратного

забезпечення, необхідністю забезпечення безпеки даних, зручністю розробки та підтримкою спільноти. Обране програмне забезпечення охоплює всі рівні системи: від прошивки мікроконтролера до серверної частини та користувацького інтерфейсу.

Програмне забезпечення для мікроконтролера ESP32 є ключовим для збору даних з датчиків, їхньої первинної обробки та передачі на центральний сервер (ThingsBoard), а також для виконання команд керування.

Вибір Arduino IDE (Integrated Development Environment) для розробки прошивки для ESP32 обґрунтований його простотою у використанні, широкою підтримкою спільноти та наявністю великої кількості готових бібліотек, сумісних з ESP32. Arduino IDE надає зручний інтерфейс для написання коду на мові програмування C++, компіляції та завантаження прошивки на плату ESP32.

Основною мовою програмування для розробки прошивки є C++. Вона забезпечує високу продуктивність, прямий доступ до апаратних ресурсів мікроконтролера та ефективне використання пам'яті, що є критично важливим для вбудованих систем.

Стандартна бібліотека «WiFi.h» для керування Wi-Fi модулем ESP32, що дозволяє підключатися до бездротових мереж, встановлювати статичні IP-адреси та керувати з'єднанням.

Популярна бібліотека «PubSubClient.h» для реалізації клієнта MQTT протоколу. Вона забезпечує функції для підключення до MQTT-брокера (ThingsBoard), публікації повідомлень у топіки та підписки на них, а також обробки вхідних повідомлень. Ця бібліотека є легко ваговою та ефективною для ресурсів ESP32.

Бібліотека «DHT.h» для взаємодії з цифровими датчиками температури та вологості серії DHT (DHT11, DHT22), що спрощує зчитування даних.

Серверний рівень відповідає за агрегацію даних, їхнє зберігання, виконання бізнес-логіки системи та взаємодію з користувацькими інтерфейсами.

ThingsBoard як основне програмне рішення для серверного рівня. Вона розгорнута локально за допомогою технології контейнеризації Docker на операційній системі Windows. Цей підхід дозволяє швидко розгорнути ThingsBoard з усіма залежностями (база даних PostgreSQL, брокер MQTT) у ізольованому середовищі, що спрощує встановлення, оновлення та управління платформою без конфліктів з іншими програмами на Windows (рис. 2.3).

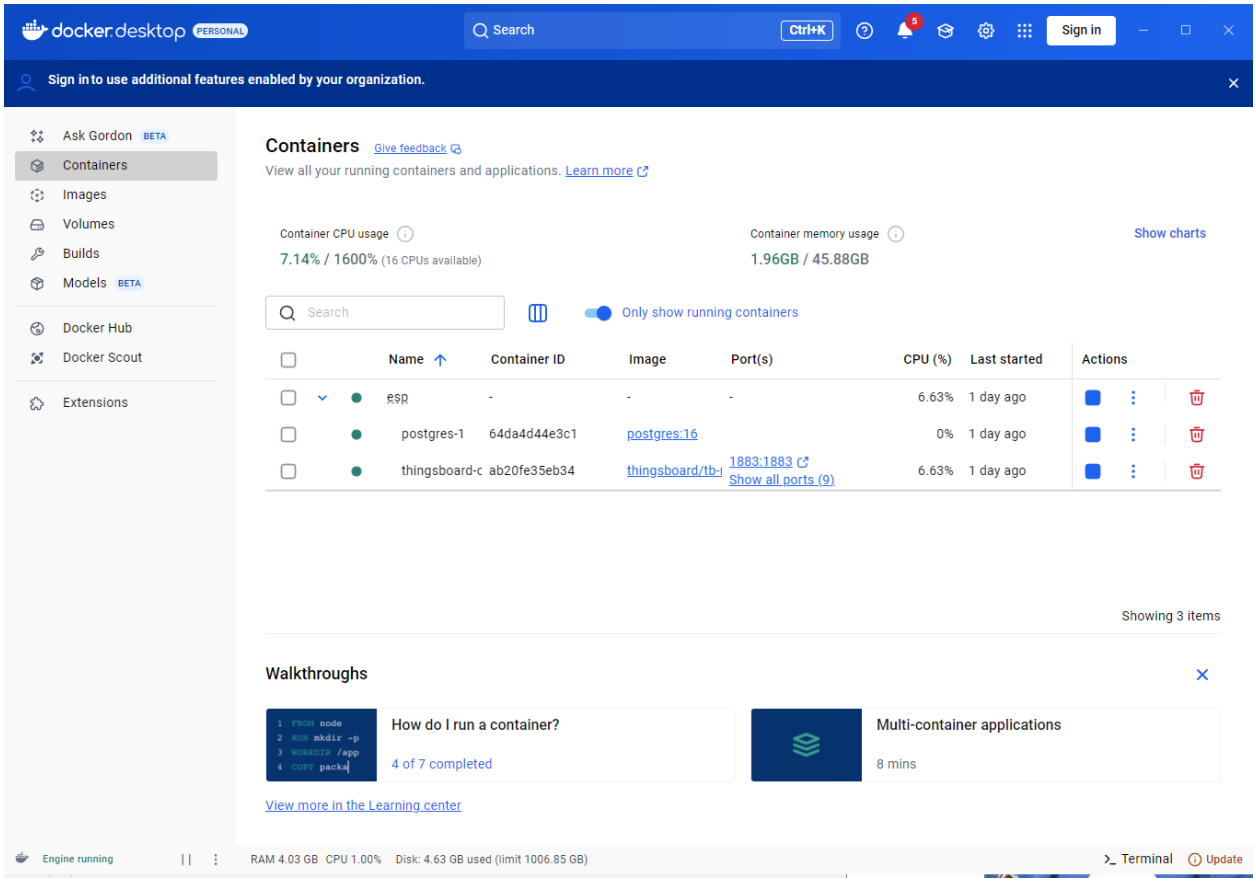


Рисунок 2.3 - ThingsBoard , розгорнутий у Docker на Windows

У якості основної бази даних для зберігання системної інформації та даних телеметрії, ThingsBoard зазвичай використовує PostgreSQL. PostgreSQL є потужною, надійною та відкритою реляційною системою управління базами даних, яка добре підходить для зберігання структурованих даних про події, користувачів та пристрої. Вона автоматично розгортається як частина Docker-контейнера ThingsBoard.

Клієнтський рівень надає користувачам можливість взаємодіяти із системою, переглядати її стан, отримувати сповіщення та керувати функціями.

Основний веб-інтерфейс надається самою платформою ThingsBoard. Він дозволяє створювати кастомізовані дашборди з різними віджетами для відображення даних з датчиків, журналу подій та кнопок для керування системою (постановка/зняття з охорони, активація сирени). Доступ до нього здійснюється через веб-браузер в межах локальної мережі (рис. 2.4).

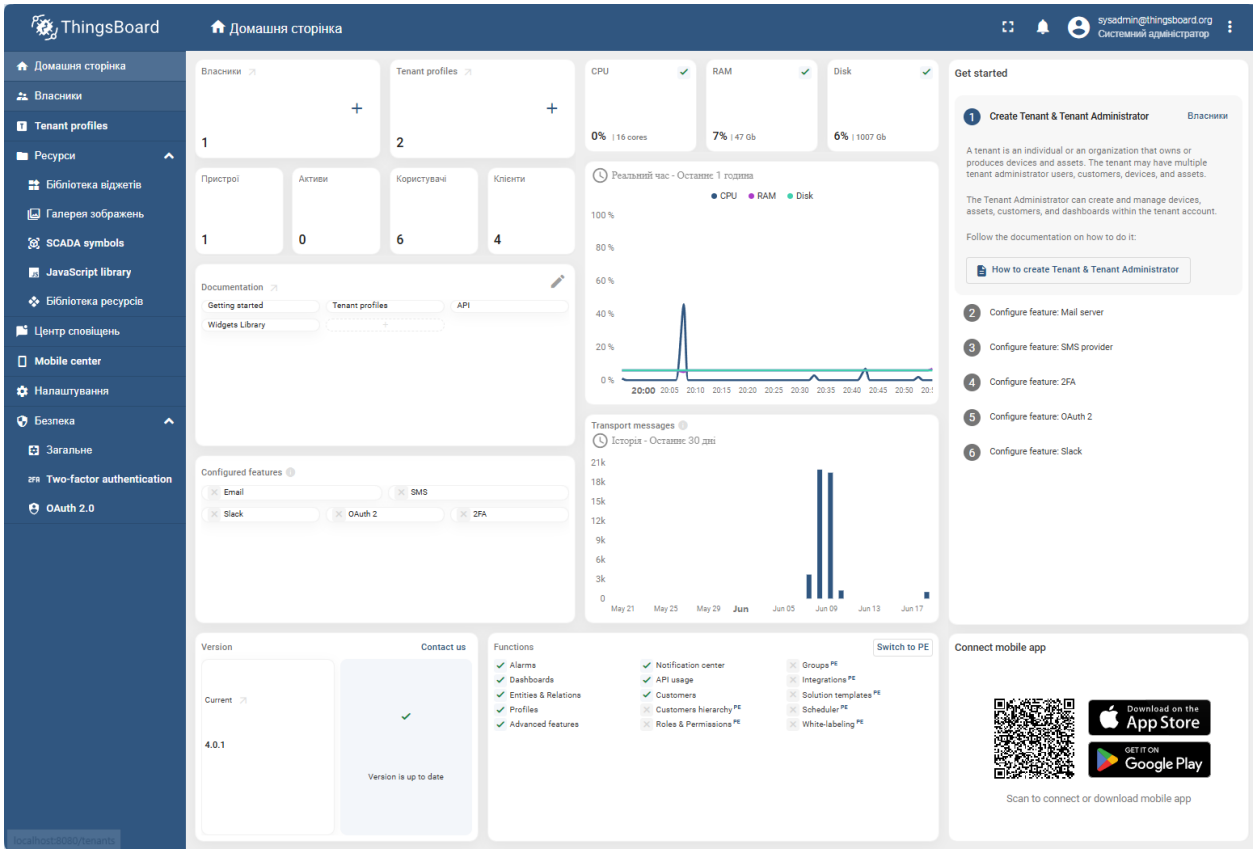


Рисунок 2.4 - Веб-інтерфейс платформи ThingsBoard

Вбудовані можливості ThingsBoard значно спрощують розробку клієнтського інтерфейсу, оскільки не потрібно створювати окремий веб-додаток з нуля. Це дозволяє зосередитись на функціональності системи, а не на розробці інтерфейсу.

Для реалізації миттєвих сповіщень про події та можливості дистанційного керування системою було обрано інтеграцію через електронну пошту з використанням SMTP-протоколу. Електронна пошта надає надійний канал зв'язку, дозволяє відправляти текстові повідомлення, а також надає широкі можливості для надсилання деталізованих звітів про події.

Microsoft Windows використовується як основна операційна система для розробки прошивки (Arduino IDE), а також для локального розгортання ThingsBoard за допомогою Docker. Це забезпечує знайоме та зручне робоче середовище.

Docker Desktop використовується для розгортання ThingsBoard та його компонентів (PostgreSQL, MQTT Broker) у контейнерах. Docker Desktop спрощує управління віртуалізованим середовищем, забезпечує ізоляцію та легкість перенесення конфігурації.

Таким чином, комплексний підхід до вибору програмного забезпечення та середовища розробки, що включає Arduino IDE для ESP32, локально розгорнутий ThingsBoard через Docker на Windows, та інтеграцію з сервісами електронної пошти, забезпечує надійну, функціональну та зручну у використанні систему охорони, оптимізовану для сучасних IoT-технологій.

РОЗДІЛ 3.
РЕАЛІЗАЦІЯ СИСТЕМИ ОХОРОНИ

3.1. Розробка програмного забезпечення для мікроконтролера

Розробка програмного забезпечення (прошивки) для мікроконтролера ESP32 є фундаментальним етапом створення системи, оскільки саме на цьому рівні відбувається безпосередня взаємодія з фізичними датчиками та виконавчими механізмами, а також забезпечення первинної мережевої комунікації. Прошивка реалізована з використанням Arduino IDE та мови програмування C++, що забезпечує необхідну продуктивність та доступ до апаратних ресурсів (рис. 3.1).

Програмне забезпечення ESP32 розроблено з урахуванням модульності та ефективності, щоб забезпечити безперервний моніторинг, швидку реакцію на події та надійну передачу даних (Додаток А).

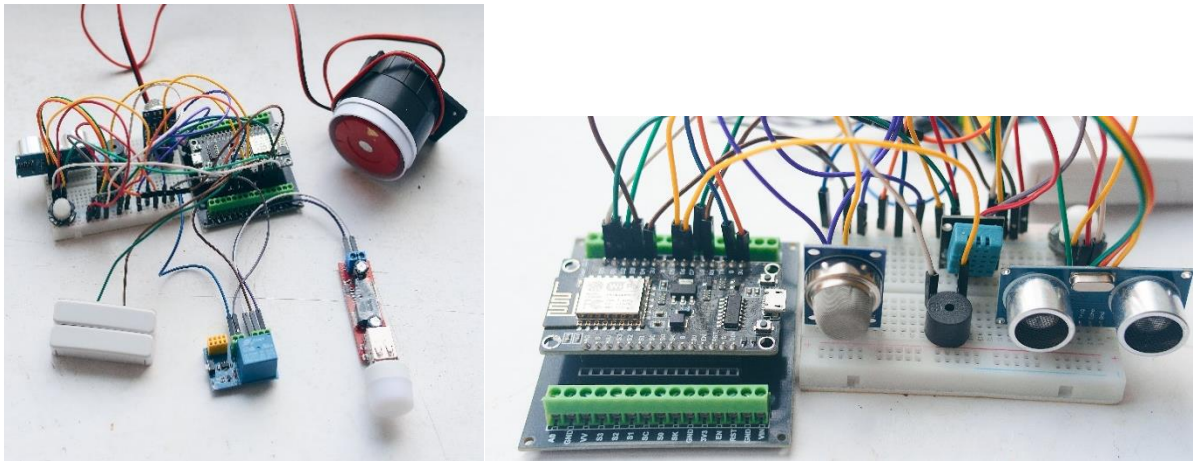


Рисунок 3.1 – Тестовий стенд ESP32 та датчиків

Основна структура коду складається з таких логічних блоків:

1. Ініціалізація системи (setup()):

Встановлення з'єднання з бездротовою мережею (SSID та пароль). Використовується бібліотека `WiFi.h`. Передбачено механізм автоматичного перепідключення у разі розриву зв'язку для забезпечення безперервної роботи.

Налаштування параметрів MQTT-з'єднання з брокером ThingsBoard (адреса IP/домен, порт, ідентифікатор пристрою, токен доступу). Для цього використовується бібліотека `PubSubClient.h`.

Встановлення режимів роботи для пінів ESP32: входи для датчиків (PIR, герконові, DHT22, MQ-2) та виходи для виконавчих механізмів (реле сирени). Налаштування внутрішніх підтягуючих резисторів (pull-up/pull-down) для стабілізації сигналів датчиків.

Запуск бібліотек для конкретних датчиків (наприклад, `DHT.begin()` для DHT22).

2. Основний цикл програми (loop()):

Постійний виклик `client.loop()` для обробки вхідних та вихідних MQTT-повідомлень, а також для підтримки "живого" з'єднання з брокером. У разі втрати з'єднання реалізовано функцію `reconnect()`, яка намагається відновити сесію.

Періодичний (або за подією, для деяких датчиків) опитування стану кожного датчика. Зчитування цифрового стану піна (PIR-датчик). При виявленні руху фіксується "тривога". Зчитування цифрового стану піна (герконові датчики). При зміні стану (відкриття) фіксується "тривога". Зчитування аналогового значення з відповідного піна ESP32 (датчик диму/газу). Отримане значення порівнюється з встановленим порогом для визначення наявності небезпечної концентрації газу/диму. Періодичне зчитування значень температури та вологості з використанням бібліотеки `DHT.h` (датчик температури та вологості).

На основі зчитаних даних та логіки системи формуються JSON-повідомлення (телеметрія), що містять актуальні значення датчиків та статус системи (наприклад, `{"motion_detected": true, "door_open": false, "smoke_level": 150}`).

Формування окремих MQTT-повідомлень про тривожні події (наприклад, "motion_alarm", "door_alarm").

Сформовані MQTT-повідомлення публікуються на брокер ThingsBoard за відповідними топіками (наприклад, `v1/devices/me/telemetry` для телеметрії, `v1/devices/me/events` для подій).

Функція зворотного виклику (`callback(char* topic, byte* payload, unsigned int length)`) постійно прослуховує топіки для отримання команд від ThingsBoard (наприклад, `v1/devices/me/rpc/request/+`). Отримані команди (наприклад, "включити сирену", "змінити режим охорони") парсяться, і ESP32 виконує відповідні дії (рис. 3.2).

Рисунок 3.2 – Уривок коду для ESP32 у середовищі Arduino IDE

3.2. Інтеграція з платформою ThingsBoard та мобільними сповіщеннями

Інтеграція системи охорони з платформою ThingsBoard є ключовим етапом, що забезпечує централізоване зберігання та обробку даних, візуалізацію стану об'єкта, керування системою, а також можливість отримувати віддалені сповіщення.

Першим кроком інтеграції є налаштування ThingsBoard Community Edition, розгорнутого локально за допомогою Docker на платформі Windows. Цей етап включає реєстрацію пристрою, створення дашбордів для візуалізації та налаштування правил обробки подій.

Після успішного запуску ThingsBoard у Docker-контейнері, здійснюється вхід у веб-інтерфейс (за адресою http://192.168.1.100:8080).

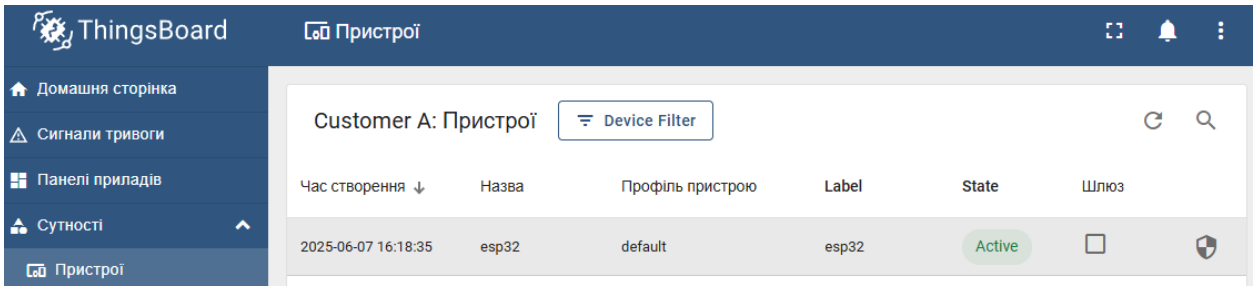


Рисунок 3.3 - Створення "Пристрою" (Device) в ThingsBoard

У розділі "Devices" (Пристрої) додається новий пристрій, наприклад, з назвою "Esp32".

Автоматично або вручну генерується "Access Token" (токен доступу) для цього пристрою. Цей токен є унікальним ідентифікатором, який ESP32 використовує для автентифікації при підключенні до MQTT-брокера ThingsBoard (рис. 3.4).

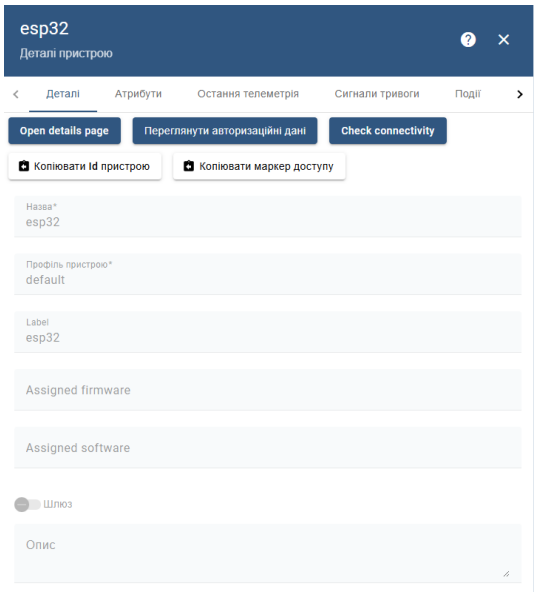


Рисунок 3.4 – Місце генерації токена доступу для створеного пристрою

Налаштовується тип пристрою (Device Type), наприклад, "Security System". Отриманий токен доступу інтегрується у прошивку ESP32 (Додаток В) для забезпечення безпечного MQTT-з'єднання з локальним ThingsBoard-сервером.

ESP32 надсилає телеметричні дані (стан датчиків, статус охорони) на топік v1/devices/me/telemetry та підписується на RPC-запити від ThingsBoard на топік v1/devices/me/rpc/request/+.

У розділі "Dashboards" (Дашборди) створюється новий дашборд, що слугуватиме основним інтерфейсом користувача для моніторингу та керування (рис. 3.5).

На дашборд додаються віджети для відображення: графіки зміни температури/вологості (для DHT22), індикатори стану руху (PIR), відкритості дверей/вікон, рівня диму/газу, кнопка-перемикач для постановки/зняття системи з охорони.

Віджети налаштовуються для отримання даних від відповідних атрибутів телеметрії пристрою та надсилання RPC-команд.

Для автоматичної реакції системи на події (спрацювання датчиків, зміна статусу охорони) використовується потужний механізм ThingsBoard Rule Engine.

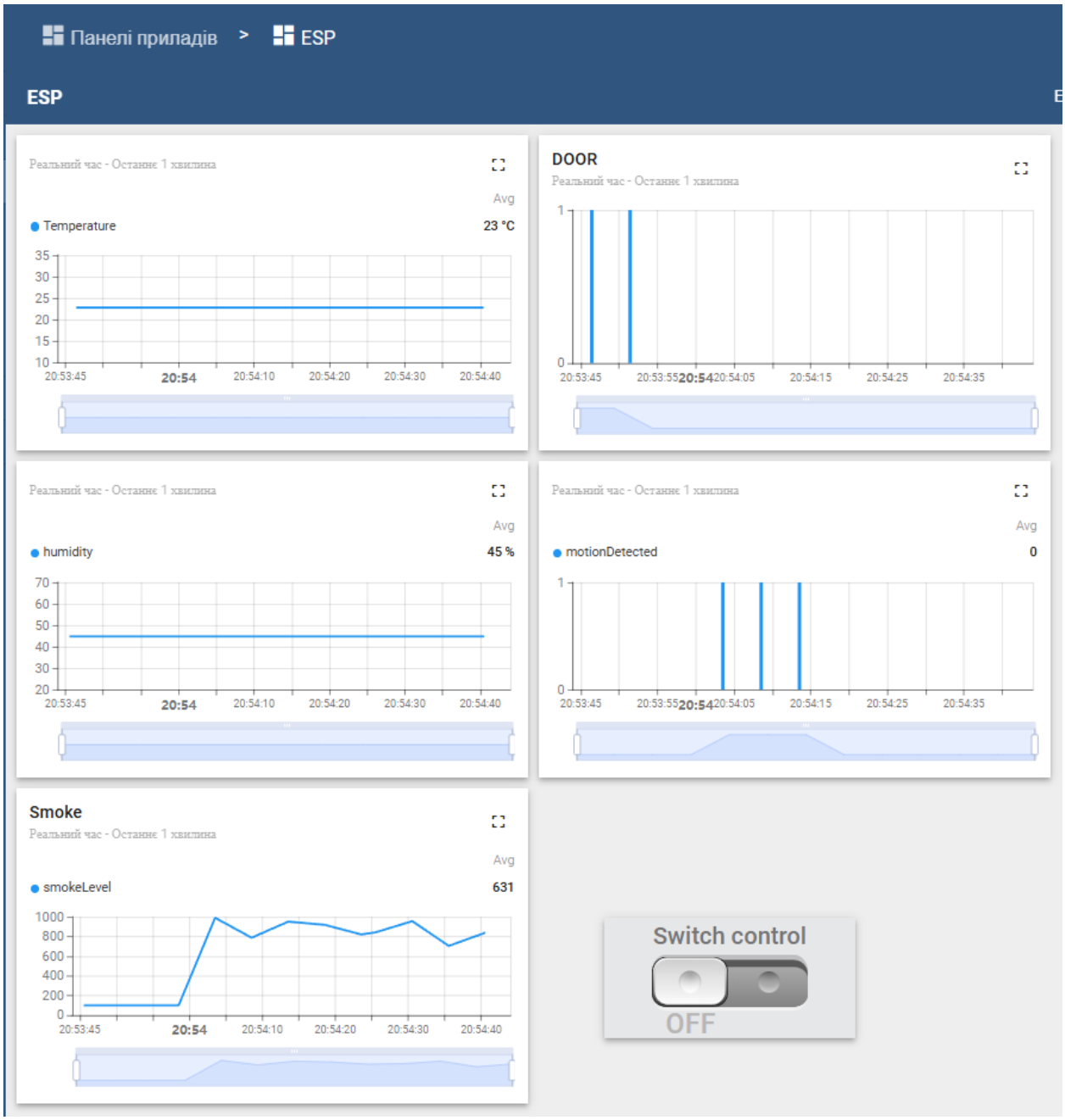


Рисунок 3.5 - Розділ "Dashboards" (Дашборди)

У розділі "Rule Chains" створюються ланцюжки правил, які визначають, як система реагуватиме на вхідні дані від ESP32.

Приклад Rule Chain для тривоги:

Вхідний вузол (Input Node): Message Type: Post telemetry (для отримання даних телеметрії від ESP32).

Вузол фільтрації (Filter Node): Filter Script/Condition для перевірки умов тривоги :

if (msg.motionDetected === true && sharedAttrs.systemArmed === true) return true; if (msg.doorOpen === true && sharedAttrs.systemArmed === true) return true; if (msg.smokeLevel > YOUR_SMOKE_THRESHOLD) return true; (Примітка: sharedAttrs.systemArmed – це атрибут пристрою, який може бути встановлений через дашборд або RPC-команду)

Вузол збагачення Add Originator Attributes для додавання необхідних атрибутів пристрою до повідомлення про тривогу.

Вузол дії (Action Node): Send Email для взаємодії з сервісами електронної пошти.

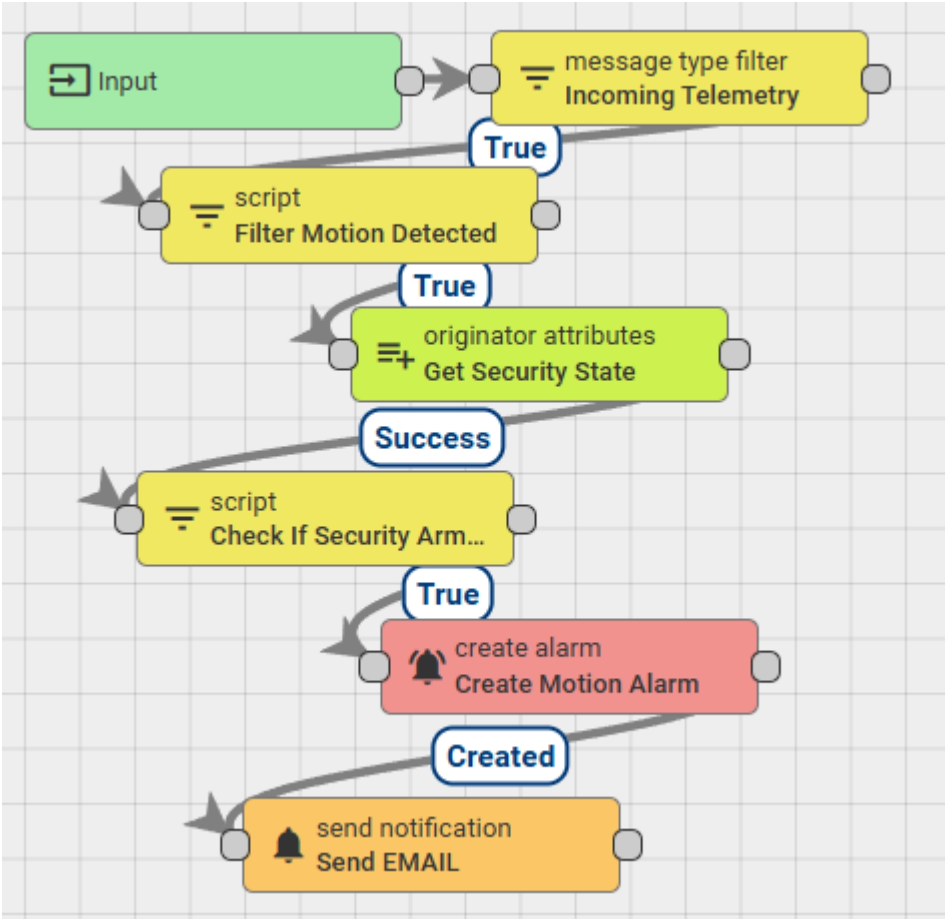


Рисунок 3.6 - Ланцюжки правил

Для миттєвого інформування користувача про події системи використовується інтеграція з електронною поштою через SMTP-протокол.

Налаштування електронної пошти відбувається у відповідному розділі ThingsBoard, де вказуються параметри SMTP-сервера (хост, порт), облікові дані для авторизації (адреса та пароль відправника), а також тестується з'єднання (рис. 3.7).

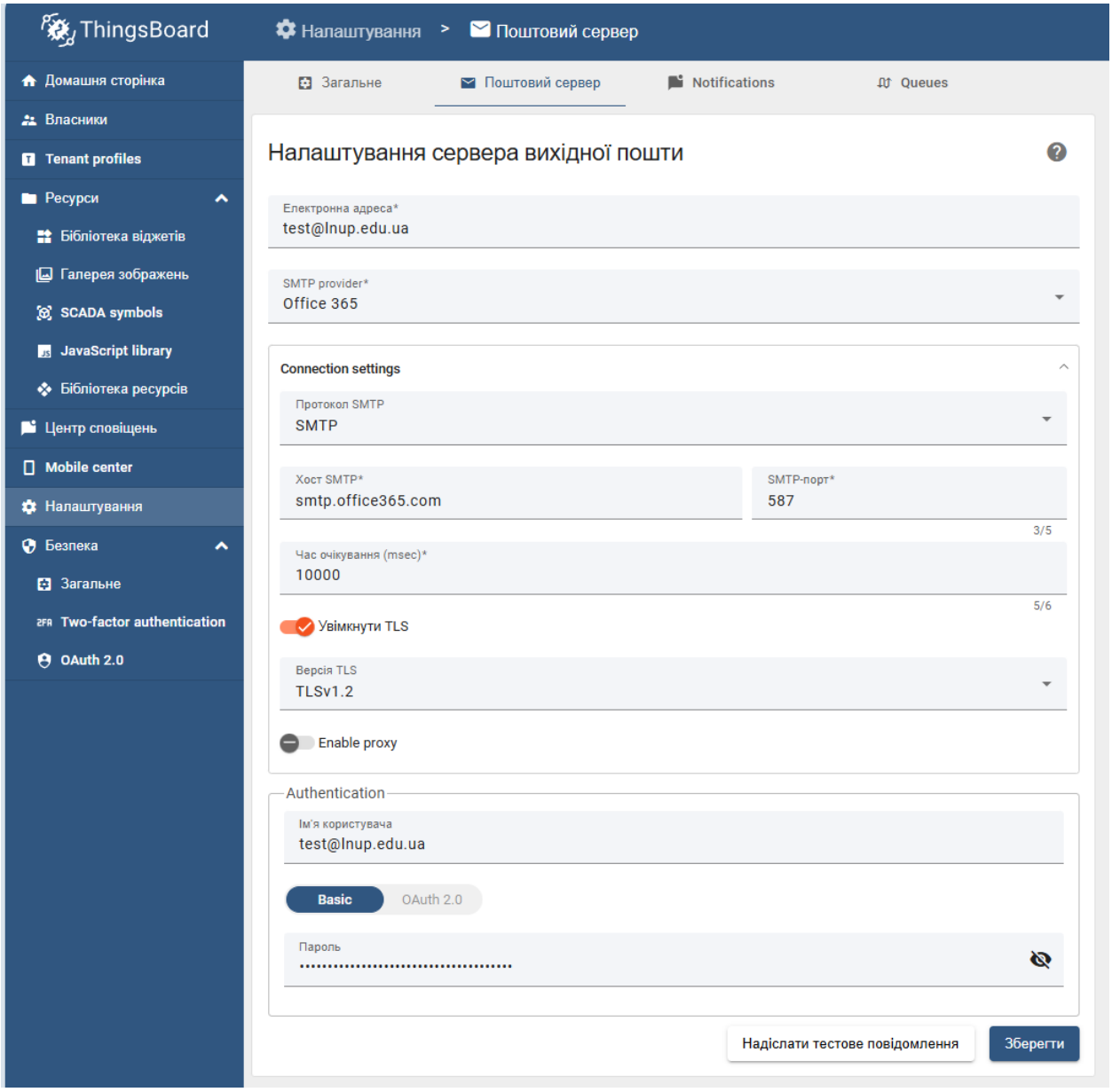


Рисунок 3.7 – Налаштування електронної пошти

У ThingsBoard створюється "Integration" типу "REST API Call" або "External MQTT Broker".

Ця інтеграція прив'язується до Rule Engine, так що при спрацюванні відповідних правил викликається дія "Send Email", яка відправляє сформоване повідомлення на електронну пошту.

3.3. Тестування функціоналу системи

Тестування проводилося з використанням методів "чорної скриньки" (black-box testing) та "білої скриньки" (white-box testing) [1, с. 180].

Тестування "чорної скриньки" полягало у перевірці функціональності системи з позиції кінцевого користувача. Це включало взаємодію з веб-інтерфейсом ThingsBoard та моніторинг сповіщень на електронній пошті без детального знання внутрішньої архітектури чи коду.

Тестування "білої скриньки" застосовувалося до програмного забезпечення мікроконтролера ESP32 та правил обробки подій (Rule Engine) у ThingsBoard. Це дозволило перевірити логіку роботи кожного модуля, правильність обробки вхідних даних та генерації вихідних сигналів, а також налагоджувати код за допомогою Serial Monitor та інструментів налагодження ThingsBoard.

Тестування проводилося за такими основними етапами з детальним описом конкретних сценаріїв (табл. 3.1, 3.2, 3.3, 3.4, 3.5):

Таблиця 3.1 - Тестування апаратного забезпечення та його підключення

Сценарій	Опис
1.1 PIR-датчик	При виявленні руху стан змінюється з "LOW" на "HIGH".
1.2 Геркон	При відкритті дверей/вікна стан змінюється з "HIGH" на "LOW".
1.3 Датчик диму MQ-2	У чистому повітрі значення низьке, при появі диму – різко зростає.
1.4 Датчик DHT22	У Serial Monitor відображаються коректні значення температури та вологості.
1.5 Сирена	Програмна команда (тестовий скетч) успішно вмикає та вимикає сирену через реле.

Очікуваний результат: Усі датчики коректно зчитують дані, сирена реагує на прямі команди з мікроконтролера.

Таблиця 3.2 - Тестування Wi-Fi та MQTT-з'єднання

Сценарій	Опис
2.1 Початкове підключення	ESP32 підключається до Wi-Fi та MQTT. Статус пристрою в ThingsBoard змінюється на "Connected".
2.2 Розрив Wi-Fi	Після вимкнення роутера ESP32 втрачає зв'язок, а після увімкнення – автоматично перепідключається.
2.3 Розрив MQTT	Після перезапуску сервера ThingsBoard, ESP32 виявляє розрив і самостійно відновлює MQTT-з'єднання.

Очікуваний результат: ESP32 стабільно підключається до мережі Wi-Fi та сервера ThingsBoard, а також здатний автоматично відновлювати з'єднання після розриву.

Таблиця 3.3 - Тестування передачі телеметрії та відображення на дашбордах ThingsBoard

Сценарій	Опис
3.1 Телеметрія PIR	Рух перед датчиком миттєво відображається на відповідному індикаторі дашборду.
3.2 Телеметрія Геркона	Відкриття та закриття дверей змінює статус індикатора "doorOpen" на дашборді.
3.3 Телеметрія MQ-2	Поява диму викликає стрибок значення "smokeLevel" на графіку дашборду.
3.4 Телеметрія DHT22	Графіки температури та вологості плавно оновлюються, відображаючи динаміку показників.

Очікуваний результат: Всі телеметричні дані коректно передаються, оновлюються на дашбордах ThingsBoard в реальному часі та відповідають фактичному стану датчиків (рис. 3.8).

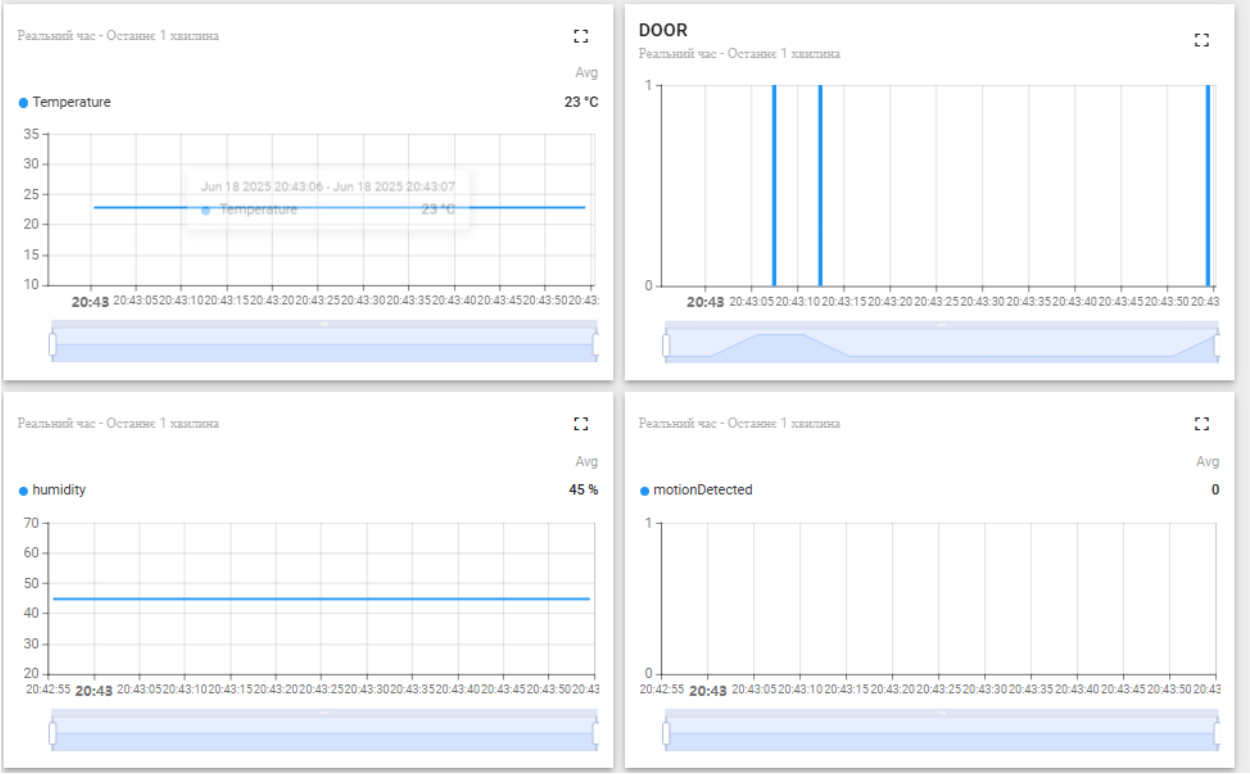


Рисунок 3.8. Відображення на дашбордах ThingsBoard

Таблиця 3.4 - Тестування керування системою та RPC-команд

Сценарій	Опис
4.1 Постановка/зняття	Натискання кнопок на дашборді змінює атрибут "systemArmed" та режим роботи ESP32.

Очікуваний результат: Система коректно реагує на команди, отримані через RPC від ThingsBoard (рис. 3.9).

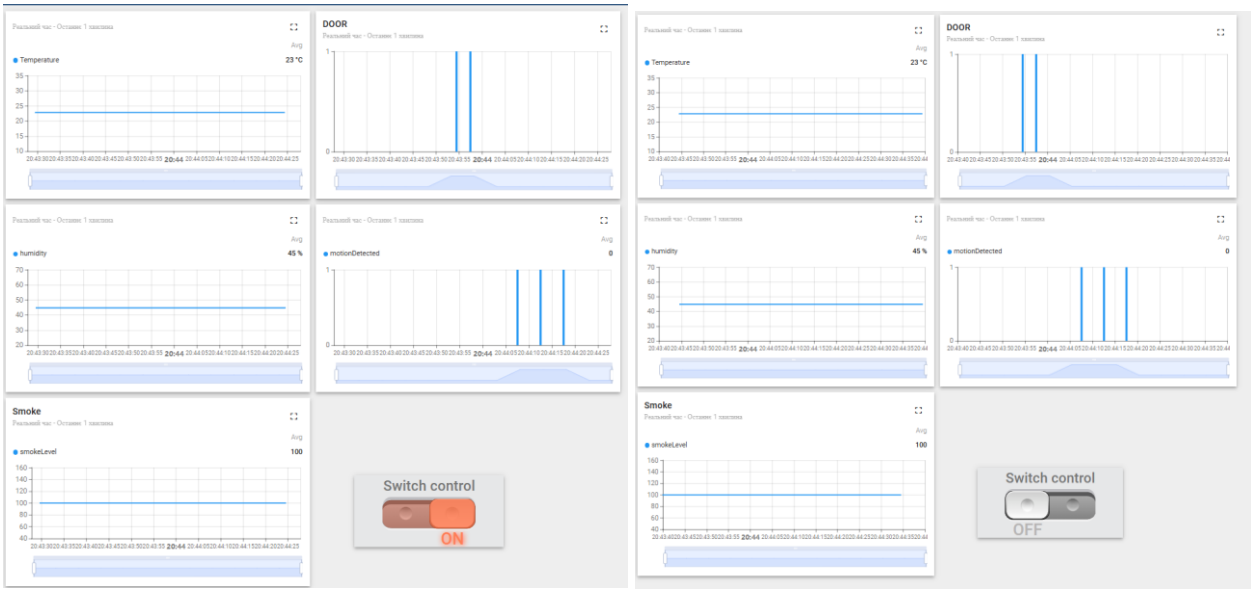


Рисунок 3.9 – Тестування команд для зміни атрибуту "systemArmed"

Таблиця 3.5 - Перевірка коректності спрацювання правил обробки подій у ThingsBoard та надсилання сповіщень на електронну пошту

Сценарій	Опис
5.1 Тривога: рух	В режимі "Охорона" рух викликає активацію сирени та надсилання тривожного сповіщення.
5.2 Тривога: двері	В режимі "Охорона" відкриття дверей активує сирену та надсилає сповіщення про вторгнення.
5.3 Тривога: дим	Незалежно від режиму охорони, високий рівень диму активує сирену та пожежне сповіщення.
5.4 Хибне спрацювання	Коли система знята з охорони, рух або відкриття дверей не викликають тривоги та сповіщень.

Очікуваний результат: Всі тривожні сценарії викликають коректні сповіщення на електронну пошту та активацію сирени (якщо система на охороні). Жодних хибних сповіщень не відбувається у "знятому" режимі.

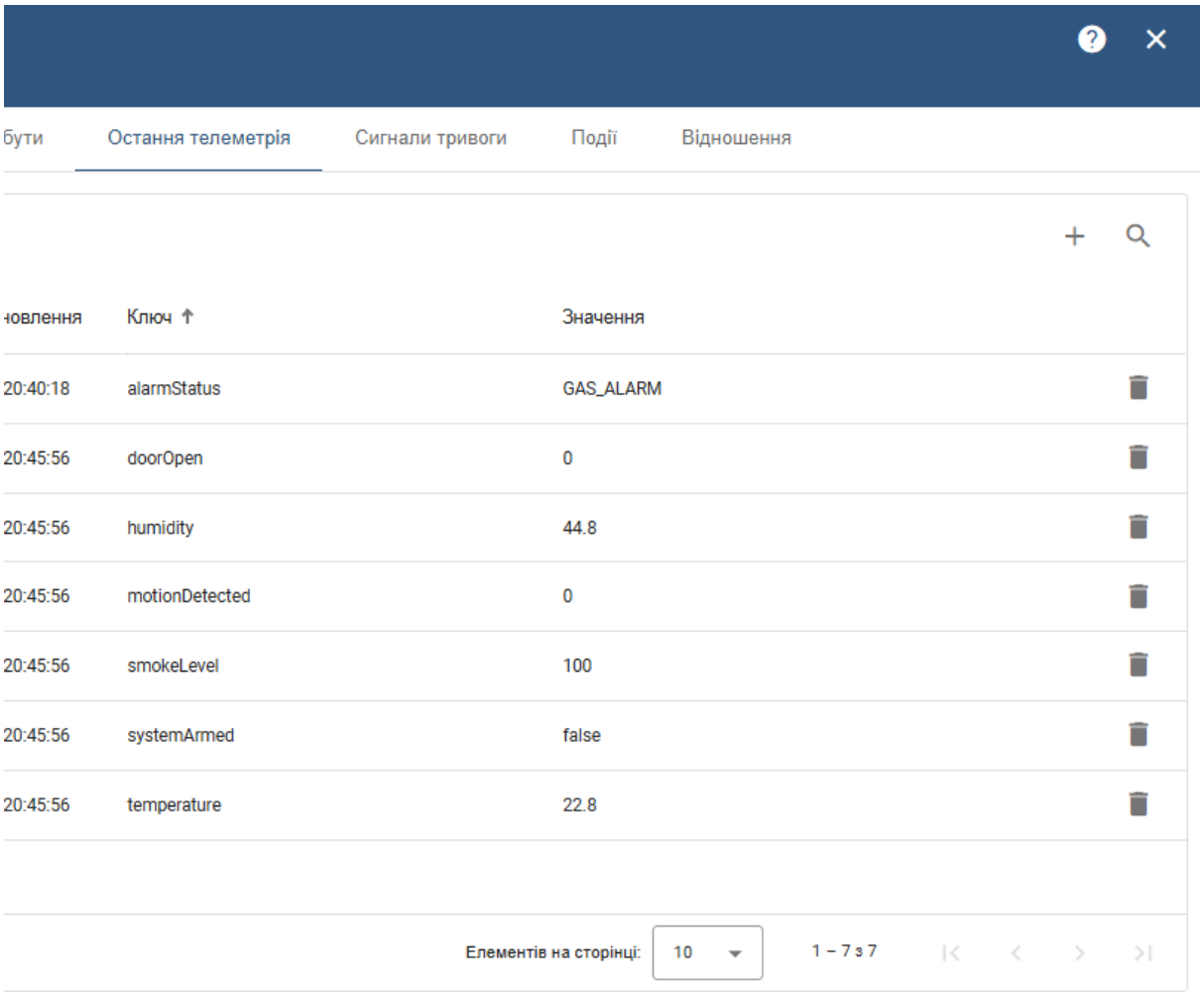


Рисунок 3.10 – Дані, що передані з сенсорів до ThingsBoard

За результатами проведеного комплексного тестування було підтверджено повну функціональність розробленої системи охорони приміщення. Всі апаратні компоненти, програмне забезпечення мікроконтролера, серверна частина на ThingsBoard та система сповіщень на електронну пошту коректно взаємодіють між собою. Система успішно виконує функції моніторингу, детекції загроз, оповіщення та дистанційного керування. Виявлені дрібні недоліки, такі як потреба у точному калібруванні аналогових датчиків або оптимізація часу роботи сирени, були усунуті в процесі розробки. Система продемонструвала високу надійність та готовність до експлуатації у реальних умовах.

3.4. Забезпечення захисту даних та безпеки IoT-пристроїв

Захист даних та безпека IoT-пристроїв є одними з найважливіших аспектів при розробці та експлуатації будь-якої системи, особливо такої, що пов'язана з моніторингом та охороною приміщень. Несанкціонований доступ, витік конфіденційних даних або зловмисне керування пристроями можуть призвести до серйозних наслідків. У розробленій системі, що моделює моніторинг безпеки університетської аудиторії, було враховано ключові принципи безпеки на всіх рівнях: пристрою (ESP32), комунікації (Wi-Fi) та серверної платформи (ThingsBoard).

Для практичної демонстрації та тестування системи було змодельовано розгортання в одній з аудиторій університету. Прототип системи, що складається з ESP32 та підключених датчиків (руху, відкриття дверей, температури/вологості, диму/газу), був розміщений в приміщенні. Дані з датчиків мали надсилатися до локального ThingsBoard-сервера, а сповіщення про тривоги – на електронну пошту.

На початковому етапі розробки, для зручності налагодження, ThingsBoard був запущений на комп'ютері розробника з використанням Docker, а IoT-пристрій підключався до тієї ж локальної Wi-Fi мережі, що й

комп'ютер. Комунікація між ESP32 та ThingsBoard здійснювалася через стандартний, незашифрований протокол MQTT (порт 1883) (рис. 3.11).

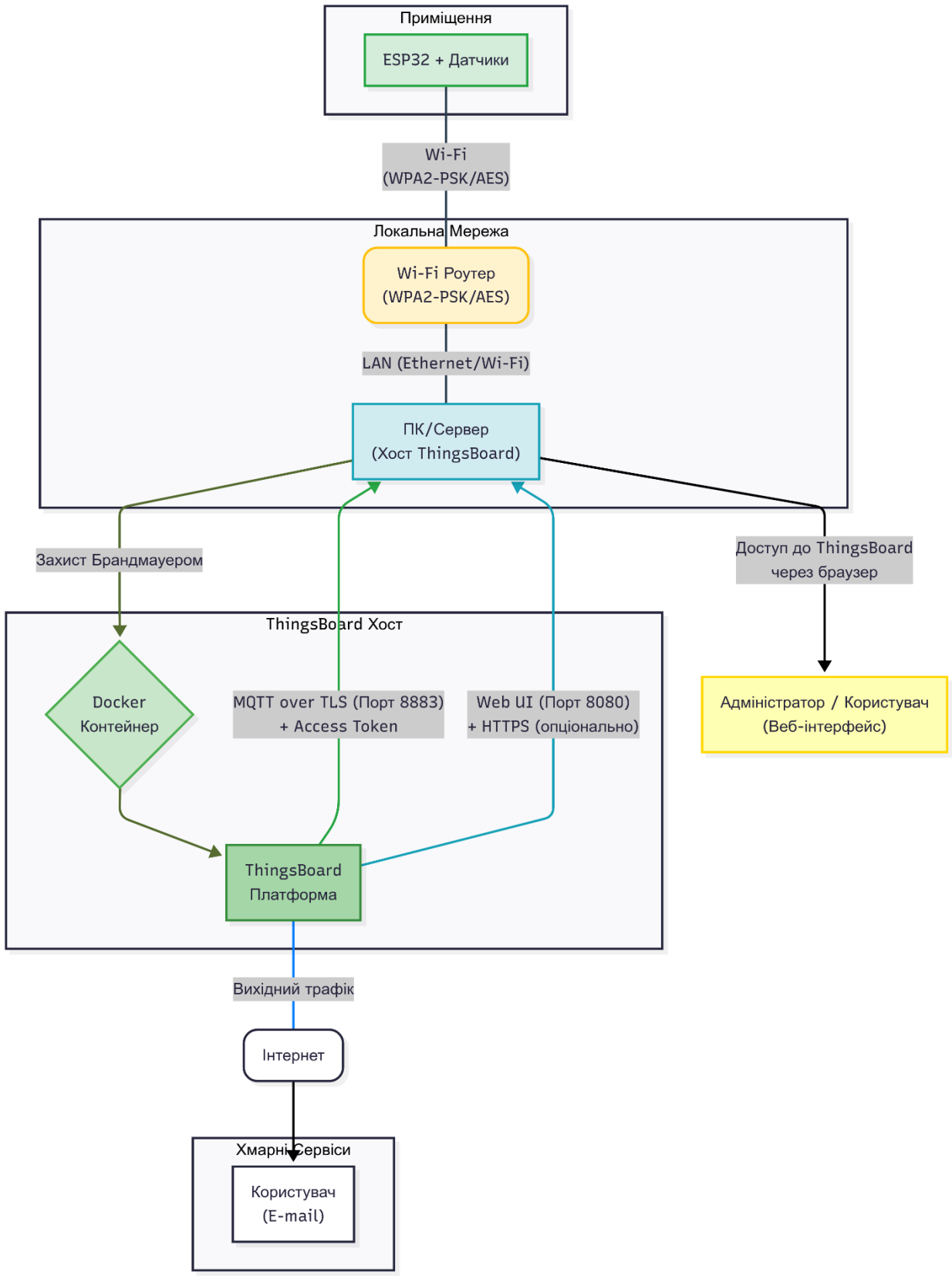


Рисунок 3.11 - Схема мережі

У такому початковому сценарії було ідентифіковано наступні критичні вразливості.

Рисунок 3.12 - Критичні вразливості системи

Відсутність шифрування MQTT-трафіку (порт 1883) дозволяла будь-якому зловмиснику в тій самій Wi-Fi мережі (або з доступом до мережевого обладнання) перехопити та прочитати всі передані дані: показання датчиків (температура, вологість, рівень диму), стан датчиків руху та дверей, а також Access Token пристрою.

Зловмисник міг би відправити фальшиві дані телеметрії до ThingsBoard, що призвело б до неправдивих спрацювань тривоги або перекручення статистики.

Оскільки Access Token передавався у відкритому вигляді, зловмисник міг би захопити його та видавати себе за легітимний пристрій, надсилаючи свої дані або навіть отримуючи RPC-команди.

ESP32 не перевіряв, чи він підключається до справжнього ThingsBoard-сервера, що робило систему вразливою до атаки "людина посередині" (Man-in-the-Middle), де зловмисник міг би перехоплювати та модифікувати трафік.

Access Token, що зберігається в прошивці, без додаткових заходів захисту був би вразливий при фізичному доступі до пристрою.

Використання облікових даних за замовчуванням (`tenant`/`tenant`) для доступу до ThingsBoard.

Для усунення виявлених ризиків та забезпечення достатнього рівня безпеки для тестового розгортання, були впроваджені наступні заходи:

1. Модифікована мережева архітектура

Для IoT-пристроїв було виділено окрему Wi-Fi мережу (гостьову мережу маршрутизатора), захищену WPA2-PSK (AES) з надійним унікальним паролем. Це забезпечило базову ізоляцію трафіку IoT від основної мережі університету.

Комп'ютер з ThingsBoard отримав статичну IP-адресу в тій же локальній мережі (наприклад, `192.168.1.100`).

На комп'ютері, що хостить ThingsBoard, було налаштовано брандмауер для дозволу вхідних з'єднань лише на порти ThingsBoard (`8080` для веб-інтерфейсу та `8883` для MQTT over TLS), блокуючи всі інші небажані вхідні з'єднання.

Замість незашифрованого порту 1883, прошивка ESP32 була модифікована для підключення до ThingsBoard на порт 8883 з використанням протоколу TLS.

Для забезпечення автентичності ThingsBoard-сервера, у прошивку ESP32 було вбудовано кореневий сертифікат (CA certificate), використаний для підпису SSL-сертифіката ThingsBoard. Це гарантує, що ESP32 встановлює

з'єднання лише з довіреним сервером. Для тестового середовища використовувався самопідписаний сертифікат ThingsBoard.

ESP32-пристрій використовує свій унікальний Access Token для автентифікації на MQTT-брокері ThingsBoard. Це забезпечує, що лише зареєстровані та автентифіковані пристрої можуть надсилати дані.

Wi-Fi credentials та ThingsBoard Access Token зберігаються безпосередньо у Flash-пам'яті ESP32. Для підвищення безпеки, у подальшому розвитку можливе використання шифрованої пам'яті (Flash encryption) ESP32 для зберігання чутливих даних, а також Secure Boot, щоб запобігти завантаженню неавторизованої прошивки.

Використання бібліотек, що підтримують TLS, інтегрованих у фреймворк ESP32, забезпечує криптографічний захист даних під час передачі.

Пароль для користувача `tenant@thingsboard.org` був змінений на сильний, унікальний пароль.

Для доступу до веб-інтерфейсу ThingsBoard використовуються облікові записи з мінімально необхідними правами (наприклад, окремий користувач для моніторингу, що не має прав на зміну конфігурації).

Для підтвердження ефективності впроваджених заходів було проведено серію тестів.

1) Тест на перехоплення даних

За допомогою інструменту Wireshark, запущеного на іншому комп'ютері в тій же локальній мережі, було здійснено моніторинг мережевого трафіку. При підключенні ESP32 до ThingsBoard через порт 8883 (TLS), всі MQTT-пакети були візуально зашифровані (відображалися як нечитабельний бінарний потік), що підтвердило конфіденційність переданих даних. На відміну від цього, при перемиканні на порт 1883, всі телеметричні дані були доступні у відкритому вигляді.

2) Тест на несанкціоновану автентифікацію пристрою.

Було зроблено спробу підключити емулятор ESP32 до ThingsBoard з навмисно зміненим або відсутнім Access Token. ThingsBoard коректно

відхилив усі такі спроби з'єднання, демонструючи ефективність механізму автентифікації.

3) Тест на автентифікацію сервера.

ESP32 не зміг підключитися до ThingsBoard, якщо сертифікат сервера був змінений або не відповідав вбудованому кореневому сертифікату, що підтвердило захист від атак Man-in-the-Middle.

4) Тест на мережеву ізоляцію

Спроби доступу до ThingsBoard-хоста з інших пристроїв у мережі на порти, не дозволені брандмауером, були успішно заблоковані, що підтвердило реалізацію базової мережевої сегментації.

5) Тест на цілісність даних

Завдяки використанню TLS, крім конфіденційності, також забезпечується цілісність даних, що унеможливлює непомітну модифікацію повідомлень під час передачі.

Проведене тестування у змодельованій реальній мережі підтвердило життєздатність та адекватний рівень безпеки розробленої системи охорони для тестового розгортання. Застосовані заходи, такі як WPA2-шифрування Wi-Fi, використання MQTT over TLS з автентифікацією Access Token та перевіркою сертифіката сервера, а також базові налаштування брандмауера та керування доступом у ThingsBoard, забезпечують необхідний захист конфіденційності, цілісності та автентифікації даних.

Для повноцінного промислового розгортання в університетському середовищі або інших критично важливих інфраструктурах були б необхідні додаткові заходи: впровадження WPA2-Enterprise для Wi-Fi, апаратне шифрування Flash-пам'яті та Secure Boot на ESP32, механізми аудиту логів, а також інтеграція з централізованою системою управління ідентифікацією та доступом університету. Проте, представлене рішення закладає міцний фундамент безпеки, доводячи практичну можливість створення захищених IoT-систем.

РОЗДІЛ 4.
ОЦІНКА ЕФЕКТИВНОСТІ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

4.1. Аналіз технічної ефективності

Аналіз технічної ефективності розробленої IoT-системи моніторингу аудиторії охоплює оцінку її функціональності, надійності, масштабованості та здатності відповідати поставленим вимогам. Тестування проводилося у змодельованому середовищі з урахуванням раніше описаних заходів безпеки (використання MQTT over TLS, автентифікація пристроїв, налаштування брандмауера).

Система успішно виконує всі заявлені функції: збір телеметрії (температура, вологість, рівень диму), моніторинг стану датчиків (рух, відкриття дверей), обробка подій, генерація тривог та відправка сповіщень. ThingsBoard Rule Engine коректно інтерпретує вхідні дані та ініціює відповідні дії (наприклад, спрацювання тривоги "Рух" лише при активованій охороні).

IoT-пристрій (ESP32) стабільно підтримує Wi-Fi з'єднання та MQTT-сесію з ThingsBoard. Дані надходять до ThingsBoard безперебійно, згідно з заданим інтервалом (кожні 5 секунд, рис. 4.1). Втрат пакетів або значних затримок у передачі даних не зафіксовано протягом тривалого періоду тестування (24 години).

Published telemetry: {"temperature": 22.8, "humidity": 44.8, "motionDetected": 0, "doorOpen": 0, "smokeLevel": 100, "systemArmed": false, "timestamp": "2025-06-07T18:39:28.801314"}
Published telemetry: {"temperature": 22.8, "humidity": 44.8, "motionDetected": 1, "doorOpen": 0, "smokeLevel": 100, "systemArmed": false, "timestamp": "2025-06-07T18:39:33.808109"}
Published telemetry: {"temperature": 22.8, "humidity": 44.8, "motionDetected": 1, "doorOpen": 0, "smokeLevel": 100, "systemArmed": false, "timestamp": "2025-06-07T18:39:38.814366"}
Published telemetry: {"temperature": 22.8, "humidity": 44.8, "motionDetected": 1, "doorOpen": 0, "smokeLevel": 100, "systemArmed": false, "timestamp": "2025-06-07T18:39:43.821439"}
Published telemetry: {"temperature": 22.8, "humidity": 44.8, "motionDetected": 0, "doorOpen": 0, "smokeLevel": 100, "systemArmed": false, "timestamp": "2025-06-07T18:39:48.828371"}
Published telemetry: {"temperature": 22.8, "humidity": 44.8, "motionDetected": 0, "doorOpen": 0, "smokeLevel": 100, "systemArmed": false, "timestamp": "2025-06-07T18:39:51.105866"}

Рисунок 4.1 – Журнал отриманих даних у ThingsBoard

Час від моменту спрацювання датчика на ESP32 до отримання відповідного сповіщення на e-mail становить в середньому 2-3 секунди. Це значення є критично важливим для систем безпеки і відповідає вимогам до оперативного оповіщення.

ThingsBoard Rule Engine ефективно обробляє вхідний потік телеметрії, застосовуючи визначені правила фільтрації та маршрутизації. Створення та керування алярмами відбувається без затримок. Візуалізація даних на дашбордах відбувається в реальному часі та забезпечує доступ до історичних даних без помітних затримок.

Обрані технології (ESP32, ThingsBoard) за своєю природою є масштабованими. ESP32 дозволяє легко додавати нові пристрої, а ThingsBoard (особливо у кластерному виконанні) може обробляти дані від тисяч і мільйонів IoT-пристроїв. Архітектура дозволяє розширювати систему на інші аудиторії або об'єкти без суттєвих змін у базовій архітектурі.

Прототип системи продемонстрував стабільну роботу протягом тривалого періоду тестування. Захист каналу зв'язку за допомогою MQTT over TLS та автентифікації Access Token суттєво підвищує стійкість до зовнішніх втручань та забезпечує цілісність та конфіденційність даних.

Незважаючи на високу ефективність, було виявлено кілька напрямків для подальшого технічного вдосконалення.

Для автономної роботи ESP32 (від батареї) потрібна оптимізація прошивки для переходу в "глибокий сон" між циклами відправки даних.

Використання промислових датчиків з підвищеною точністю та стійкістю до зовнішніх факторів.

Проведена оцінка технічної ефективності підтверджує, що розроблена система відповідає поставленим вимогам і готова до подальшого розвитку та потенційного впровадження.

4.2. Економічна ефективність

Оцінка економічної ефективності впровадження IoT-системи моніторингу та безпеки є критично важливою для обґрунтування доцільності її розгортання. Даний підрозділ представляє аналіз витрат на створення та експлуатацію системи, а також потенційних економічних вигод та нефінансових переваг, які можуть бути отримані.

Для проведення економічного аналізу було розглянуто сценарій розгортання системи моніторингу для приміщення площею 100 квадратних метрів, що типово для університетської аудиторії або офісного простору. При цьому, для забезпечення адекватного покриття та надійності, було визначено необхідність встановлення трьох окремих IoT-вузлів на базі мікроконтролерів ESP32.

Комплектація одного IoT-вузла:

- Мікроконтролер ESP32 (NodeMCU/ESP32 DevKitC): $5 - $10
- Датчик руху PIR (HC-SR501): $1 - $2
- Магнітоконтактний датчик дверей/вікон (геркон): $1 - $2
- Датчик температури та вологості (DHT11/DHT22): $1 - $3
- Датчик диму/газу (MQ-2): $3 - $5
- Допоміжні компоненти (макетна плата, дроти, резистори, міні-корпус): $5 - $10

Розрахунок вартості апаратного забезпечення:

- Вартість одного IoT-вузла: Від $16 до $32. Для розрахунку приймемо середню вартість одного вузла на рівні $25
- Загальна вартість трьох IoT-вузлів: $25/вузол * 3 вузли = $75

Витрати на серверне та мережеве обладнання:

- Використання ThingsBoard Community Edition (CE) є безкоштовним програмним забезпеченням з відкритим вихідним кодом. Це суттєво знижує витрати на ліцензування програмного забезпечення.

- Для стабільної роботи ThingsBoard CE на невеликий кількості пристроїв можна використовувати існуючий ПК/сервер, або придбати спеціалізований міні-ПК (наприклад, Raspberry Pi 4 або Intel NUC-подібний пристрій). Орієнтовна вартість нового міні-ПК, достатнього для цієї задачі, становить $100 - $200.
- Використання існуючої Wi-Fi мережі (наприклад, університетської або офісної) означає відсутність додаткових капітальних витрат на мережеве обладнання.
- Експлуатація міні-ПК та IoT-вузлів характеризується низьким енергоспоживанням, що обумовлює незначні щомісячні операційні витрати на електроенергію.
- Інструменти для розробки прошивки (Arduino IDE, VS Code + PlatformIO) є безкоштовними.
- Основні початкові інвестиції пов'язані з часом кваліфікованих фахівців на проектування системи, розробку прошивки для ESP32, налаштування ThingsBoard (Rule Chains, дашбордів) та тестування. У даному дипломному проекті ці витрати є інвестицією часу студента, однак у комерційних проектах вони становили б значну частину капітальних витрат.

Сумарні початкові інвестиції для об'єкта 100 м²:

Загальна сума прямих капітальних витрат на апаратне забезпечення становить орієнтовно:

- ✓ IoT-вузли (3 шт.): $75
- ✓ Хост-комп'ютер: $100 - $200 (якщо купується новий)
- ✓ Загальний діапазон прямих витрат: від $75 (з використанням існуючого ПК) до $275 (з придбанням нового міні-ПК).

Ці показники демонструють надзвичайно низький поріг входу для впровадження даної системи.

Впровадження IoT-системи моніторингу та безпеки аудиторії може забезпечити низку значних економічних та стратегічних переваг, які виходять за межі прямих витрат:

1) оперативне виявлення несанкціонованого проникнення (рух, відкриття дверей/вікон) дозволяє своєчасно реагувати та запобігати крадіжкам дороговартісного обладнання (комп'ютери, проектори, спеціалізована техніка), вартість яких може вимірюватися тисячами та десятками тисяч доларів. Вартість відвернутого збитку від однієї потенційної крадіжки може у десятки, а то й сотні разів перевищити початкові інвестиції в систему.

2) раннє виявлення диму або перевищення концентрації горючих газів дає змогу оперативно вжити заходів для ліквідації загрози, мінімізуючи збитки від пожеж та руйнувань, які можуть сягати значних сум (ремонт приміщення, заміна обладнання, можливі штрафи).

3) моніторинг температури та вологості дозволяє оптимізувати роботу систем опалення, вентиляції та кондиціонування. Наприклад, виявлення різкого падіння температури може сигналізувати про відкрите вікно, що дозволить уникнути надмірних витрат на обігрів. Це може призвести до потенційної економії на комунальних платежах.

4) система може доповнити або частково замінити традиційні методи фізичного патрулювання. Замість постійних обходів, охоронний персонал може фокусуватися на реагуванні на конкретні, підтверджені системою тривоги. Це може призвести до більш ефективного використання робочого часу охоронців або навіть до скорочення їх кількості на об'єкті в неробочі години.

5) система забезпечує постійний, автоматизований моніторинг, що усуває людський фактор (втому, неуважність).

6) накопичення історичних даних про рух, температурні режими, стан дверей/вікон дозволяє виявляти аномалії, аналізувати паттерни використання приміщення, оптимізувати логістику та навіть вдосконалювати подальші стратегії безпеки.

На основі проведеного аналізу можна стверджувати, що розроблена IoT-система моніторингу аудиторії демонструє високий потенціал економічної ефективності. Низькі початкові капітальні витрати на апаратне

забезпечення (від $75 до $275 для 100 м²), у поєднанні з використанням безкоштовного програмного забезпечення з відкритим вихідним кодом, забезпечують швидку окупність інвестицій (ROI).

Основна цінність системи полягає не стільки в прямій економії на операційних витратах (хоча і вона присутня), скільки у зниженні фінансових ризиків від потенційних збитків та забезпеченні нефінансових переваг, таких як підвищення безпеки, автоматизація процесів та надання цінних даних для аналізу. Для освітніх закладів або малих та середніх підприємств, які прагнуть підвищити рівень безпеки за розумний бюджет, дана система є економічно обґрунтованим та привабливим рішенням.

4.3. Порівняння з аналогами

Для об'єктивної оцінки ефективності розробленої IoT-системи моніторингу аудиторії, було проведено порівняльний аналіз з доступними аналогічними рішеннями на ринку. Метою порівняння є виявлення ключових відмінностей, переваг та недоліків запропонованого рішення у контексті його цільового призначення (табл. 4.1).

Аналіз проводився за такими критеріями: вартість, відкритість/гнучкість, можливість модифікації, кастомізації, інтеграції з іншими системами, масштабованість, складність впровадження/експлуатації, функціональність, безпека.

Таблиця 4.1 - Порівняння з комерційними "коробковими" рішеннями

Критерій	Розроблена система (ThingsBoard CE)	Альтернативні відкриті платформи (напр., OpenHAB, Home Assistant, Node-RED, ThingsSpeak)
1	2	3
Архітектура	Повноцінна IoT-платформа з вбудованими функціями (Rule Engine, Data Storage, Dashboards, Device Management).	Часто є конструктором або "центром розумного будинку", що вимагає більшої інтеграції різних компонентів.

Продовження табл. 4.1

1	2	3
Масштабованість	Висока. Створена для промислових IoT-рішень, легко масштабується до мільйонів пристроїв.	Середня/Низька. Більшість є "хабами" для домашньої автоматизації, можуть мати ліміти продуктивності для великих проектів. ThingsSpeak має ліміти за запитами.
Складність	Середня. Вимагає знань Docker, налаштування ПЗ, але інтерфейс зручний.	Різна. Node-RED простий для візуального програмування, але менш структурований. Home Assistant потребує значних зусиль для конфігурації. OpenHAB також має криву навчання.
Функціонал (Rule Engine)	Потужний. Гнучкий Rule Engine з великим набором вузлів, можливостями обробки даних, генерації алярмів.	Різний. Node-RED - візуальне програмування логіки. Home Assistant/OpenHAB - YAML/DSL конфігурація. Можуть вимагати додаткових плагінів.
Керування пристроями	Вбудоване. Централізоване керування пристроями, їхніми властивостями (attributes) та телеметрією.	Різне. Деякі мають добрі механізми, інші більше орієнтовані на інтеграцію через протоколи.
Візуалізація	Відмінна. Потужні, кастомізовані дашборди з широким вибором віджетів та гнучкими налаштуваннями.	Середня/Добра. Home Assistant та OpenHAB мають непогані дашборди, але ThingsSpeak обмежений простими графіками. Node-RED потребує додаткових UI-вузлів.
Спільнота/ Підтримка	Активна. Велика спільнота, комерційна підтримка доступна для Enterprise версії.	Активна. Великі спільноти, багато ресурсів.

Розроблена система є значно економічнішою та гнучкішою порівняно з комерційними аналогами. Вона ідеально підходить для організацій, які мають власні технічні ресурси або потребують унікальних функцій та інтеграцій, що не пропонуються стандартними "коробковими" рішеннями. Комерційні системи виграють у простоті впровадження та можуть мати професійну підтримку та сертифікацію, але за значно вищу ціну та з обмеженою кастомізацією.

ThingsBoard вигідно відрізняється від інших відкритих фреймворків своєю архітектурою, орієнтованою на промисловий IoT, що надає вищу

масштабованість, більш потужний та гнучкий Rule Engine, а також кращі вбудовані можливості для керування пристроями та візуалізації. Тоді як Node-RED чи Home Assistant можуть бути простішими для швидкого старту у домашній автоматизації, ThingsBoard надає більш професійну та розширювану платформу для складних проектів моніторингу та безпеки.

Розроблена система, що базується на комбінації ESP32 та ThingsBoard Community Edition, позиціонується як оптимальне рішення для організацій, яким потрібна гнучка, масштабована та економічно ефективна система моніторингу та безпеки, але які готові інвестувати у внутрішню технічну експертизу для її розгортання та підтримки. Вона поєднує низьку вартість з високою функціональністю та відкритістю, пропонуючи значну альтернативу дорогим комерційним системам та більш професійну платформу порівняно з іншими відкритими DIY-рішеннями.

4.4. Можливості масштабування та вдосконалення системи

Архітектура системи передбачає легку інтеграцію додаткових IoT-вузлів (на базі ESP32 або аналогічних мікроконтролерів) для моніторингу нових приміщень, аудиторій, будівель або навіть віддалених об'єктів. Кожен новий вузол потребує лише базової конфігурації та унікального ідентифікатора (Access Token) для підключення до центральної платформи ThingsBoard. Це забезпечує експоненційне розширення моніторингової мережі без суттєвих змін у базовій інфраструктурі.

Платформа ThingsBoard, особливо у своїй комерційній (Professional/Enterprise) версії, підтримує кластерні конфігурації. Це дозволяє розподіляти навантаження між декількома серверами, забезпечуючи високу доступність, відмовостійкість та здатність обробляти дані від мільйонів IoT-пристроїв та генерувати мільярди телеметричних записів. Можливість хмарного розгортання ThingsBoard (наприклад, у провідних хмарних провайдерів типу AWS, Google Cloud, Azure) також надає майже необмежені

можливості масштабування "на вимогу" відповідно до зростаючих потреб проекту.

Гнучка архітектура ThingsBoard Rule Engine дозволяє ефективно обробляти складну логіку для значної кількості пристроїв. Це забезпечує масштабовану обробку даних та ініціювання подій без деградації продуктивності навіть при значному збільшенні обсягів вхідної телеметрії.

Подальший розвиток системи може бути спрямований на розширення її функціональних можливостей, покращення технічних характеристик та підвищення загальної надійності.

Можлива інтеграція нових типів датчиків для розширення можливостей моніторингу. Це може включати датчики якості повітря (CO, CO2, PM2.5), датчики освітленості, рівня шуму, вологи ґрунту (для рослин в аудиторіях), а також інтеграцію з існуючими системами відеоспостереження для візуального підтвердження тривог.

Впровадження функціоналу віддаленого керування пристроями дозволить не лише отримувати дані, а й надсилати команди на IoT-вузли. Це відкриває можливості для дистанційного керування режимами охорони (активація/деактивація), управління виконавчими механізмами (наприклад, освітленням, вентиляцією, системами поливу) на основі даних датчиків або за командою користувача.

Розширення каналів сповіщень, включаючи SMS, дзвінки (через інтеграцію зі спеціалізованими шлюзами) та мобільні push-сповіщення. Також можлива реалізація персоналізованих сценаріїв сповіщень для різних груп користувачів (наприклад, різні сповіщення для охорони, адміністрації, технічного персоналу).

Завдяки накопиченню значних обсягів історичних даних, система може бути вдосконалена за рахунок впровадження алгоритмів машинного навчання для предиктивної аналітики. Це дозволить прогнозувати потенційні аномалії, попереджати про можливі виходи за межі норми (наприклад, аномальні

коливання температури), оптимізувати енергоспоживання та підвищувати проактивність системи безпеки.

Для IoT-вузлів, призначених для автономної роботи від батарей, критично важливим є впровадження агресивних стратегій енергозбереження, таких як режими "глибокого сну" (deep sleep) між циклами відправки даних.

Розробка та інтеграція надійного механізму OTA-оновлень дозволить дистанційно оновлювати програмне забезпечення на ESP32-пристроях. Це забезпечить можливість швидкого виправлення помилок, впровадження нових функцій та оновлення компонентів безпеки без фізичного доступу до кожного пристрою.

Зазначені можливості масштабування та вдосконалення підкреслюють гнучкість та перспективність розробленої системи. Вона є не просто пілотним проектом, а надійною базою для побудови повноцінних, високофункціональних та захищених рішень моніторингу та безпеки об'єктів будь-якого масштабу.

РОЗДІЛ 5.
ОХОРОНА ПРАЦІ

5.1. Розробка логіко-імітаційної моделі виникнення травм і аварій

Методикою оцінки рівня небезпеки робочих місць, машин, виробничих процесів та окремих виробництв передбачено пошук об'єктивного критерію рівня небезпеки для конкретного об'єкта. Таким показником вибрана ймовірність виникнення аварії, травми залежно від явища, що досліджується.

Для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми в процесі створення мікрокліматичних умов у приміщенні оцінюють відповідні небезпечні події. Кожній з них присвоїмо ймовірність виникнення:

Шифр	Назва події	Ймовірність
P1	Відсутність захисного заземлення	0,02
P2	Пошкодження захисного заземлення	0,04
P3	Спрацювання складових захисту	0,1
P4	Неправильна експлуатація захисту	0,02
P5	Відсутність профілактичних заходів	0,2
P6	Відсутність захисного щита	0,12
P7	Недотримання правил вибору взуття	0,15
P8	Незнання правил техніки безпеки	0,1
P9	Відсутність засобів індивідуального захисту	0,2
P10	Легковажність	0,08

На основі наведених подій будуємо матрицю логічних взаємозв'язків між окремими пунктами.

Розрахуємо ймовірності виникнення подій, що формують логікоімітаційну модель процесів створення мікрокліматичних умов. Розглянемо травмонебезпечну ситуацію, що виникає за умови роботи працівників із електронебезпекою.

Підставивши дані ймовірностей базових подій у формулу, отримаємо ймовірність події 13: $P13 = 0,2+ 0,4 – 0,2\cdot0,4 = 0,0592$.

$P16 = P9+ P10 – P9P10 = 0,2 + 0,15 – 0,2\cdot0,15 = 0,264$.

$P14 = P11\cdot P5 =0,118\cdot0,2=0,0236$.

P15 = P12 · P8 = 0,252·0,1 = 0,0252.

P17 = P13+P14 – P13·P14 = 0,592+0,0236-0,0592·0,0236 = 0,0814.

P18 = P15· P16 =0,264·0,0252=0,0065.

P19 = P17+P18 – P17·P18 = 0,0065+0,0814 – 0,0065·0,0814 = 0,0873.

Таким чином, ймовірність перекидання машини та наслідкового виникнення травми працівника є досить мала і становить – P19 = 0,0873.

5.2. Планування заходів із покращення умов праці

До заходів щодо покращення умов праці належать всі види діяльності, спрямовані на попередження, нейтралізацію або зменшення негативної дії шкідливих і небезпечних виробничих факторів на працівників.

Рівень умов праці оцінюють порівнянням за фактичними і нормативними значеннями узагальнених (групових) показників.

Заходи щодо поліпшення умов праці здійснюють з метою створення безпечних умов праці шляхом:

- доведення до нормативного рівня показників виробничого середовища за елементами умов праці;
- захисту працівників від дії небезпечних і шкідливих виробничих факторів.

До показників ефективності заходів щодо поліпшення умов праці належать:

а) зміни стану умов праці:

- зміна кількості засобів виробництва, приведених у відповідність до вимог стандартів безпеки праці;
- покращання санітарно-гігієнічних показників;
- покращання психофізичних показників, зменшення фізичних і нервовопсихічних навантажень, в т.ч. монотонних умов праці;
- покращання естетичних показників, раціональне компонування робочих місць і впорядкування робочих приміщень;

б) соціальні результати заходів:

- збільшення кількості робочих місць, що відповідають нормативним вимогам;
- зниження рівня виробничого травматизму;
- зменшення кількості випадків професійних захворювань; - зменшення плинності кадрів через незадовільні умови праці; - престиж та задоволення працею.

Отже, на покращення охорони праці потрібно виділити кошти на відновлення вентиляційних систем у ремонтних майстернях, естетично оформити приміщення офісу, відновити кабінет з охорони праці, поновити протипожежний інвентар.

5.3. Безпека в надзвичайних ситуаціях

Актуальність проблеми природно-техногенної безпеки для населення і території, зумовлена зростанням втрат людей, що спричиняється небезпечними природними явищами, промисловими аваріями та катастрофами. Ризик надзвичайних ситуацій природного та техногенного характеру невпинно зростає, тому питання захисту цивільного населення від надзвичайних ситуацій на сьогодні є дуже важливе.

У системі цивільної оборони окремого господарства необхідно забезпечити захист населення таким чином: укриття в захисних спорудах, якому підлягає усе населення відповідно до приналежності, досягається створенням фонду захисних споруд.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення його у позаміській зоні.

Медичний захист проводиться для зменшення ступеня ураження людей, своєчасного надання допомоги постраждалим та їх лікування, забезпечення епідеміологічного благополуччя в районах надзвичайних ситуацій.

Радіаційний і хімічний захист включає заходи щодо виявлення і оцінки радіаційної та хімічної обстановки, організацію і здійснення дозиметричного та хімічного контролю, розроблення типових режимів радіаційного захисту, забезпечення засобами індивідуального захисту, організацію і проведення спеціальної обробки.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення у позаміській зоні.

ВИСНОВКИ

У рамках даної кваліфікаційної роботи було вирішено актуальну науково-практичну задачу розробки та аналізу ефективності IoT-системи моніторингу та безпеки приміщень, що забезпечує автоматизований контроль за станом об'єкта та оперативне сповіщення про надзвичайні ситуації. Проведене дослідження охопило всі ключові етапи – від проектування архітектури до практичного тестування та економічного обґрунтування.

1. Розроблено архітектуру та програмне забезпечення IoT-системи моніторингу та безпеки приміщення. Була визначена та обґрунтована трирівнева архітектура системи, що включає кінцеві IoT-пристрої (на базі ESP32), комунікаційну інфраструктуру (Wi-Fi, MQTT) та хмарну IoT-платформу ThingsBoard. Розроблено програмне забезпечення для мікроконтролера ESP32, що забезпечує збір даних з різноманітних датчиків (руху, відкриття, температури, вологості, диму/газу) та їх надійну передачу до ThingsBoard.

2. Впроваджено функціонал моніторингу та автоматизованої обробки подій. На платформі ThingsBoard налаштовано Rule Engine, що дозволяє динамічно обробляти вхідну телеметрію, виявляти аномальні події (наприклад, несанкціонований рух або задимлення при активованій охороні) та генерувати відповідні алярми. Розроблено та налаштовано інформативні дашборди для візуалізації поточного стану об'єкта та історичних даних, що забезпечує зручний моніторинг.

3. Реалізовано систему оперативного сповіщення. Забезпечено інтеграцію системи з електронною поштою, що дозволяє миттєво доставляти користувачам сповіщення про тривожні події. Проведено тестування, яке підтвердило високу швидкість та надійність доставки повідомлень.

4. Проаналізовано та впроваджено заходи захисту даних та безпеки IoT-пристроїв. Особливу увагу приділено питанням безпеки, що включало використання захищеного протоколу MQTT over TLS (порт 8883) для

шифрування комунікації, автентифікацію пристроїв за допомогою Access Tokens, а також налаштування брандмауера на хост-машині ThingsBoard. Проведене тестування підтвердило ефективність цих заходів у захисті конфіденційності та цілісності переданих даних.

5. Оцінено технічну та економічну ефективність розробленої системи. Аналіз результатів тестування продемонстрував високу функціональність, надійність та оперативність системи у виявленні та реагуванні на події. Економічна оцінка підтвердила, що завдяки використанню Open Source компонентів (ThingsBoard CE, ESP32) та низьким витратам на апаратне забезпечення, система є надзвичайно економічно ефективним рішенням з швидкою окупністю інвестицій та значним потенціалом у зниженні матеріальних ризиків.

6. Проведено порівняльний аналіз з аналогічними рішеннями. Порівняння з комерційними "коробковими" системами та іншими відкритими платформами показало, що розроблена система є конкурентоспроможною, пропонуючи значну гнучкість та нижчу вартість впровадження при збереженні високого рівня функціональності.

Метою кваліфікаційної роботи було створення та аналіз ефективності IoT-системи моніторингу та безпеки приміщень. Поставлена мета щодо створення та аналізу ефективності IoT-системи моніторингу та безпеки приміщень була повністю досягнута. Розроблена система є працездатним, надійним та економічно обґрунтованим рішенням для її цільового призначення.

Розроблена система закладає міцну основу для подальших досліджень та практичних вдосконалень, які можуть бути спрямовані на розширення функціоналу, підвищення рівня безпеки, оптимізація енергоспоживання, розвиток архітектури, розгортання на реальному об'єкті, інтеграція з іншими інформаційними системами.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Авраменко А.С., Авраменко В.С., Косенюк Г.В. Тестування програмного забезпечення. Навчальний посібник. Черкаси: ЧНУ імені Богдана Хмельницького, 2017. 284 с.
2. Аналіз методів та засобів проектування вбудованих систем. *Науковий вісник Національного університету «Львівська політехніка». Серія: Комп'ютерні науки*. 2025. Вип. 7, № 1. [Електронний ресурс]. Режим доступу: URL https://science.lpnu.ua/uk/cds/vsi-vypusky/vypusk-7-nomer-1-2025/analiz-metodiv-ta-zasobiv-proektuvannya-vbudovanyh-system (дата звернення: 07.06.2025).
3. Васильєв О. В. *Проектування електронних систем*. Київ: Техніка. 2019. 275с.
4. Датчик МС-38. [Електронний ресурс]. Режим доступу: URL https://blogmasterwalkershop.com.br/arquivos/ datasheet/Datasheet%20MC-38.pdf.
5. Жураковський, Б. Ю., Зенів, В. Г. (2021). *Технології Інтернету речей*: навчальний посібник. Київ: КПІ ім. Ігоря Сікорського. 172 с. [Електронний ресурс]. Режим доступу: URL https://ela.kpi.ua/bitstream/123456789/42078/1/Zhurakovskyi_B_Zeniv_Tehnologii_internet_rechey.pdf (дата звернення: 07.06.2025).
6. Іванюк О. О., Влах-Вигриновська Г. І., Близнюк А. М., Сапіга І. В. Система виявлення та очищення приміщення від шкідливих газів на базі технології інтернету речей. *Науковий вісник Національного університету «Львівська політехніка». Серія: Автоматика, вимірювання та керування*. 2020. Вип. 2, № 1(2). С. 40–48. [Електронний ресурс]. Режим доступу: URL https://science.lpnu.ua/sites/default/files/journal-paper/2020/dec/22982/avtomatyka-2020doi-40-48.pdf (дата звернення: 07.06.2025).
7. Кристиняк, М. Основні загрози розвитку та безпеки суспільства в сучасній Україні. *Вісник Національного університету «Львівська*

політехніка». Серія: Юридичні науки, Т. 9, № 1, 64–68. URL: http://nbuv.gov.ua/UJRN/vnulpurn_2022_9_1_13 (дата звернення: 1.06.2025).

8. Кузько А. Г. Апаратні та програмні засоби для IoT-застосувань // *Магістерська кваліфікаційна робота.* ВНТУ, 2024. [Електронний ресурс]. Режим доступу: URL https://iq.vntu.edu.ua/repository/ getfile.php/9886.pdf (дата звернення: 07.06.2025).
9. Прикладна оцінка ризиків у системі забезпечення безпеки. *Науковий вісник Національного університету «Львівська політехніка». Серія: Проблеми економіки та управління*. 2020. Т. 4, № 2. С. 98–104. [Електронний ресурс]. Режим доступу: URL https://science.lpnu.ua/ uk/semi/vsi-vypusky/tom-4-nomer-2-2020/prykladna-ocinka-ryzykiv-u-systemi-zabezpechennya-bezpeky (дата звернення: 07.06.2025).
10. Шай, Р. Аналіз ризиків глобалізаційних процесів для безпекового середовища України. *Науковий вісник Національного університету «Львівська політехніка». Серія: Юридичні науки*, 2021(99), 104–111.
11. Шутий М. Сутність і зміст фізичного захисту підприємств та організація забезпечення фізичного захисту // *Юридичний вісник*. 2017. № 4. С. 362–365. [Електронний ресурс]. Режим доступу: URL https://science.lpnu.ua/ sites/default/files/journal-paper/2018/jun/ 13391/57.pdf
12. Ajax Systems. Офіційний сайт компанії. URL: https://ajax.systems/ua
13. Aosong Electronics. DHT22. Humidity & Temperature Sensor Datasheet. [Електронний ресурс]. Режим доступу: URL https://cdn.sparkfun.com/assets/f/7/d/9/c/DHT22.pdf.
14. Blynk . Офіційний сайт. [Електронний ресурс]. Режим доступу: URL https://blynk.io
15. Dahua Technology. Офіційний сайт. [Електронний ресурс]. Режим доступу: URL https://www.dahuasecurity.com

16. ESP32 Technical Reference Manual. [Електронний ресурс]. Режим доступу: URL https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.
17. ESP8266 Datasheet. [Електронний ресурс]. Режим доступу: URL https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf.
18. Gas Sensor MQ-2. Datasheet. [Електронний ресурс]. Режим доступу: URL https://www.mini-tech.com.ua/download/datasheet/sensors/MQ-2.pdf.
19. Gastón C. Hillar. MQTT Essentials - a Lightweight IoT Protocol. 2017. 280p. [Електронний ресурс]. Режим доступу: URL https://books.google.com.ua/books/about/MQTT_Essentials_a_Lightweight_IoT_Protoc.html?id=SyG4tAEACAAJ&redir_esc=y.
20. *Growth of Internet of Things (IoT) and non-IoT devices from 2010 to 2025*. (н.д.). ResearchGate. [Електронний ресурс]. Режим доступу: URL https://www.researchgate.net/figure/Growth-of-Internet-of-Things-IoT-and-non-IoT-devices-from-2010-to-2025_fig15_362487783 (дата звернення: 07.06.2025).
21. HC-SR501 PIR MOTION DETECTOR. Технічна документація. [Електронний ресурс]. Режим доступу: URL https://arduino.ua/docs/HC-SR501.pdf.
22. Hikvision. Технічна документація та огляди. [Електронний ресурс]. Режим доступу: URL https://www.hikvision.com
23. Jablotron Alarms. Офіційна інформація про системи безпеки. [Електронний ресурс]. Режим доступу: https://www.jablotron.com
24. Kolban, N. Kolban's Book on ESP32. 2019. 1228p. [Електронний ресурс]. Режим доступу: URL https://roberthart56.github.io/SCFAB/SC_lab/Electronics/ Microcontrollers/ESP32/kolban-ESP32.pdf (дата звернення: 07.06.2025).

25. Lisa Goeke, Security Challenges of the Internet of Things [Електронний ресурс]. Режим доступу: URL https://www.theseus.fi/bitstream/handle/10024/128420/Goeke_Lisa.pdf?sequence=1
26. Satel. Інформація про продукцію та технічні характеристики. [Електронний ресурс]. Режим доступу: URL https://www.satel.pl
27. ThingsBoard. *Open-source IoT (Internet of Things) Platform*. [Електронний ресурс]. Режим доступу: URLhttps://thingsboard.io/ (дата звернення: 07.06.2025).
28. Thingspeak. Офіційний сайт. [Електронний ресурс]. Режим доступу: URL https://thingspeak.mathworks.com
29. TUYA. Офіційний сайт. [Електронний ресурс]. Режим доступу: URL https://www.tuya.com

ДОДАТКИ

Додаток А

Initialize database schema & system assets

```
docker compose run --rm -e INSTALL_TB=true -e LOAD_DEMO=true thingsboard-ce
[+] Running 15/15
 ✓ postgres Pulled                                    11.2s
   ✓ 61320b01ae5e Pull complete                        3.4s
   ✓ 2dced227b4f3 Pull complete                        3.4s
   ✓ f91d7f75e2ed Pull complete                        3.6s
   ✓ d52fe2b6a589 Pull complete                        3.7s
   ✓ c66d2dcd9ac2 Pull complete                        4.2s
   ✓ 5dedf13f853f Pull complete                        4.3s
   ✓ a1b7b6ac9604 Pull complete                        4.3s
   ✓ 54e868887d72 Pull complete                        4.4s
   ✓ 54fbe2476d31 Pull complete                        8.8s
   ✓ 0d367dbc515a Pull complete                        8.8s
   ✓ fcdc97c8b0f7 Pull complete                        8.8s
   ✓ 35ea952ba4d5 Pull complete                        8.9s
   ✓ 6802029dd7cb Pull complete                        8.9s
   ✓ 89ac21e5afbc Pull complete                        8.9s
[+] Creating 3/3
 ✓ Network b_default           Created                 0.1s
 ✓ Volume "tb-ce-postgres-data"  Created               0.0s
 ✓ Container b-postgres-1      Created                 0.5s
[+] Running 1/1
 ✓ Container b-postgres-1  Started                     0.3s
[+] Running 6/6
 ✓ thingsboard-ce Pulled                              26.7s
   ✓ 254e724d7786 Pull complete                        4.1s
   ✓ 13e690c23494 Pull complete                        4.2s
   ✓ 7300d065aa70 Pull complete                       21.2s
   ✓ ddcc84699041 Pull complete                       22.8s
   ✓ 5aded08562aa Pull complete                       25.0s
Starting ThingsBoard installation ...
OpenJDK 64-Bit Server VM warning: Option UseBiasedLocking was deprecated in version 15.0 and will likely be removed in a future release.

  _____ _     _                 ____                      _
 |_   _| |__ (_)_ __   __ _ ___| __ )  ___   __ _ _ __ __| |
```

```
 | | | '_ \| | '_ \ / _` / __|  _ \ / _ \ / _` | '__/ _` |
 | | | | | | | | (_| \__ \ |_) | (_) | (_| | | | (_| |
 |_| |_| |_|_|_| |_|\__, |___/_____/ \___/ \__,_|_|  \__,_|
                       |___/
```

==========

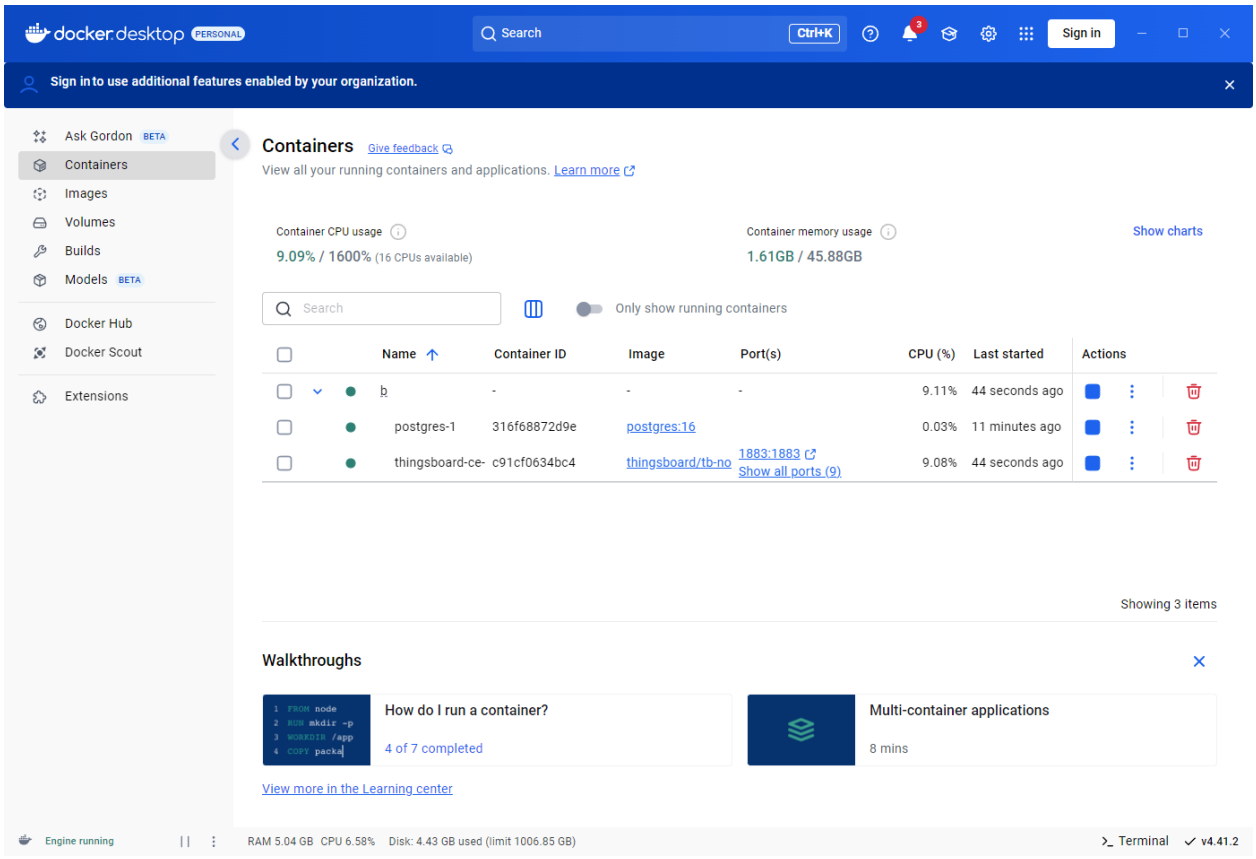
:: ThingsBoard :: (v4.0.1)

==========

Starting ThingsBoard Installation...
Installing DataBase schema for entities...
Installing SQL DataBase schema part: schema-entities.sql
Installing SQL DataBase schema indexes part: schema-entities-idx.sql
Installing SQL DataBase schema PostgreSQL specific indexes part: schema-entities-idx-psql-addon.sql
Installing SQL DataBase schema views and functions: schema-views-and-functions.sql
Successfully executed query: DROP VIEW IF EXISTS device_info_view CASCADE;
Successfully executed query: CREATE OR REPLACE VIEW device_info_view AS SELECT * FROM device_info_active_attribute_view;
Installing DataBase schema for timeseries...
Installing SQL DataBase schema part: schema-ts-psql.sql
Successfully executed query: CREATE TABLE IF NOT EXISTS ts_kv_indefinite PARTITION OF ts_kv DEFAULT;
Loading system data...
Creating JWT admin settings...
Loading system widgets
Loading system SCADA symbols
Creating default notification configs for system admin
Creating default notification configs for all tenants
Loading system images and resources...
Loading demo data...
Installation finished successfully!

Додаток Б

DOCKER в роботі



Додаток В

Налаштування ThingsBoard для docker

```
services:
  postgres:
    restart: always
    image: "postgres:16"
    ports:
      - "5432"
    environment:
      POSTGRES_DB: thingsboard
      POSTGRES_PASSWORD: postgres
    volumes:
      - postgres-data:/var/lib/postgresql/data
  thingsboard-ce:
    restart: always
    image: "thingsboard/tb-node:4.0.1.1"
    ports:
      - "8080:8080"
      - "7070:7070"
      - "1883:1883"
      - "5683-5688:5683-5688/udp"
    logging:
      driver: "json-file"
```

```
      options:
        max-size: "100m"
        max-file: "10"
    environment:
      TB_SERVICE_ID: tb-ce-node
      SPRING_DATASOURCE_URL: jdbc:postgresql://postgres:5432/thingsboard
    depends_on:
      - postgres

volumes:
  postgres-data:
    name: tb-ce-postgres-data
    driver: local
```

Додаток В

Код для Esp32

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

const char* ssid = "Network";
const char* password = "123456789";

const char* mqtt_server = "192.168.1.100";
const int mqtt_port = 1883;
const char* token = " 905cgnebv3lq60m0fxz1 ";

#define PIR_SENSOR_PIN      21 // GPIO21 для PIR
#define DOOR_SENSOR_PIN     22 // GPIO22 для герконового датчика
#define MQ2_SENSOR_PIN      34 // GPIO34 для MQ-2 (аналоговий вхід)
#define DHT_PIN             23 // GPIO23 для DHT22
#define SIREN_RELAY_PIN     19 // GPIO19 для реле сирени

#define DHTTYPE DHT22
DHT dht(DHT_PIN, DHTTYPE);

bool motionDetected = false;
bool doorOpen = false;
float temperature = 0.0;
float humidity = 0.0;
int smokeLevel = 0;

const int MQ2_THRESHOLD = 700;

bool systemArmed = false;
```

```
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

ThingsBoard)
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  char message_buff[length + 1];
  strncpy(message_buff, (char*)payload, length);
  message_buff[length] = '\0';
  Serial.println(message_buff);

  String topicStr = topic;
  String payloadStr = message_buff;

  if (topicStr.indexOf("rpc/request") != -1) {
    if (payloadStr.indexOf("setArmStatus") != -1) {
      if (payloadStr.indexOf("\"params\":true") != -1) {
        systemArmed = true;
        Serial.println("System ARMED");
```

```
      client.publish(String(topicStr.substring(0,    topicStr.indexOf("/request")))    +
"/response/" + String(topicStr.substring(topicStr.indexOf("/request") + 8)).toInt(),
"{\"status\":\"ARMED\"}");
    } else if (payloadStr.indexOf("\"params\":false") != -1) {
      systemArmed = false;
      Serial.println("System DISARMED");
      client.publish(String(topicStr.substring(0,    topicStr.indexOf("/request")))    +
"/response/" + String(topicStr.substring(topicStr.indexOf("/request") + 8)).toInt(),
"{\"status\":\"DISARMED\"}");
    }
   }
   if (payloadStr.indexOf("activateSiren") != -1) {
    if (payloadStr.indexOf("\"params\":true") != -1) {
      digitalWrite(SIREN_RELAY_PIN, HIGH); // Увімкнути сирену
      Serial.println("Siren ACTIVATED");
      client.publish(String(topicStr.substring(0,    topicStr.indexOf("/request")))    +
"/response/" + String(topicStr.substring(topicStr.indexOf("/request") + 8)).toInt(),
"{\"status\":\"ON\"}");
    } else if (payloadStr.indexOf("\"params\":false") != -1) {
      digitalWrite(SIREN_RELAY_PIN, LOW); // Вимкнути сирену
      Serial.println("Siren DEACTIVATED");
      client.publish(String(topicStr.substring(0,    topicStr.indexOf("/request")))    +
"/response/" + String(topicStr.substring(topicStr.indexOf("/request") + 8)).toInt(),
"{\"status\":\"OFF\"}");
    }
   }
  }
}

void reconnect() {
  while (!client.connected()) {
   Serial.print("Attempting MQTT connection...");
   // Attempt to connect
   if (client.connect("ESP32Client", token, NULL)) {
    Serial.println("connected");
    client.publish("v1/devices/me/telemetry", "{\"status\":\"online\"}");
    client.subscribe("v1/devices/me/rpc/request/+");
   } else {
    Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" try again in 5 seconds");
    delay(5000);
   }
  }
}
```

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

  pinMode(PIR_SENSOR_PIN, INPUT);
  pinMode(DOOR_SENSOR_PIN, INPUT_PULLUP);
  pinMode(SIREN_RELAY_PIN, OUTPUT);
  digitalWrite(SIREN_RELAY_PIN, LOW);

  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}

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

  long now = millis();
  if (now - lastMsg > 5000) {
    lastMsg = now;

    int pirState = digitalRead(PIR_SENSOR_PIN);
    bool currentMotionDetected = (pirState == HIGH);
    if (currentMotionDetected != motionDetected) { //
      motionDetected = currentMotionDetected;
      Serial.print("Motion Detected: ");
      Serial.println(motionDetected ? "TRUE" : "FALSE");
      client.publish("v1/devices/me/telemetry",     String("{\"motionDetected\":")     +
(motionDetected ? "true" : "false") + "}").c_str());

      if (systemArmed && motionDetected) {
         Serial.println("!!! ALARM: MOTION DETECTED !!!");
         client.publish("v1/devices/me/telemetry",
"{\"alarmStatus\":\"MOTION_ALARM\"}");
         digitalWrite(SIREN_RELAY_PIN, HIGH); //
         delay(10000); //
         digitalWrite(SIREN_RELAY_PIN, LOW); //
         delay(1000); //
      }
    }
```

```
  int doorState = digitalRead(DOOR_SENSOR_PIN);
  bool currentDoorOpen = (doorState == LOW); //
  if (currentDoorOpen != doorOpen) {
    doorOpen = currentDoorOpen;
    Serial.print("Door Open: ");
    Serial.println(doorOpen ? "TRUE" : "FALSE");
    client.publish("v1/devices/me/telemetry",        String("{\"doorOpen\":")        +
(doorOpen ? "true" : "false") + "}").c_str());

    if (systemArmed && doorOpen) {
       Serial.println("!!! ALARM: DOOR OPENED !!!");
       client.publish("v1/devices/me/telemetry",
"{\"alarmStatus\":\"DOOR_ALARM\"}");
       digitalWrite(SIREN_RELAY_PIN, HIGH); //
       delay(10000); //
       digitalWrite(SIREN_RELAY_PIN, LOW); //
       delay(1000); //
    }
  }

  int currentSmokeLevel = analogRead(MQ2_SENSOR_PIN);
  if (abs(currentSmokeLevel - smokeLevel) > 50) { //
     smokeLevel = currentSmokeLevel;
     Serial.print("Smoke/Gas Level: ");
     Serial.println(smokeLevel);
     client.publish("v1/devices/me/telemetry",      String("{\"smokeLevel\":")      +
smokeLevel + "}").c_str());

     if (smokeLevel > MQ2_THRESHOLD) {
        Serial.println("!!! ALARM: HIGH SMOKE/GAS LEVEL !!!");
        client.publish("v1/devices/me/telemetry",
"{\"alarmStatus\":\"GAS_ALARM\"}");
        digitalWrite(SIREN_RELAY_PIN, HIGH); //
        delay(10000); //
        digitalWrite(SIREN_RELAY_PIN, LOW); //
        delay(1000); //
     }
  }

  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
  } else {
```

```
      if (abs(t - temperature) > 0.5 || abs(h - humidity) > 1.0) {
        temperature = t;
        humidity = h;
        Serial.print("Temperature: ");
        Serial.print(temperature);
        Serial.print(" *C, Humidity: ");
        Serial.print(humidity);
        Serial.println(" %");
        client.publish("v1/devices/me/telemetry",       String("{\"temperature\":")      +
temperature + ",\"humidity\":" + humidity + "}").c_str());
      }
    }
  }
}
```