ВІДЗИВ НАУКОВОГО КЕРІВНИКА НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ

Студента Ковальчука Іллі Олександровича
представлену на здобуття освітнього ступеня бакалавр
зі спеціальності 123 «Комп'ютерна інженерія»

1. Робота виконана на тему „Інформаційна система по вивченню протоколу MQTT для Інтернету речей".

2. Актуальність роботи зумовлена тим, що впровадження розумного будинку набуває все більшої популярності а також використання сучасних протоколів для їх проектування та реалізаці.

3. Основний зміст пояснювальної записки роботи викладено на 55 сторінках тексту, записка містить 21 ілюстрацій, 2 додатків. Список використаних літературних джерел налічує 19 найменування. Невід'ємною складовою роботи є реалізований у середовищі моделювання Cisco Packet Tracer проект розумного будинку.

4. Пояснювальна записка складається з анотації, вступу, трьох розділів основного тексту, висновків, списку використаної літератури, додатків.

Розділ 1 У першому розділі було описано та проаналізовано літературні джерела. Було розглянуто структуру організації розумних будинків з використанням протоколу MQTT, описано методи моделювання розумних будинків засобами Cisco Packet Tracer. Було спроектовано схему взаємодії об'єктів у протоколі MQTT.

Розділ 2 В другому розділі було розглянуто та проаналізовано сучасні світові стандарти опису та протоколи Інтернету речей, зокрема, протоколи MQTT та HTTP. Були проаналізовані типові архітектури рішень на базі протоколу MQTT.

Розділ 3 У третьому розділі було спроектовано імітаційну модель IoT-системи за допомогою Cisco Packet Tracer. Було досліджено працездатність та ефективність інформаційної системи по вивченню протоколу MQTT для Інтернету речей та проведено аналіз роботи. Результатом третього розділу є готове спроектоване рішення розумного будинку з використанням протоколу MQTT.

5. Висновки містять сумарні результати проведеної роботи.

6. Список використаних літературних джерел має достатній для даного виду робіт обсяг, є правильно оформленим та повністю опрацьованим у ході виконання роботи.

7. Додатки містять змодельовану схему мережі, лістинги конфігураційних файлів налагоджених пристроїв, результати перевірки працездатності мережі.

8. Таблиці та ілюстрації оформлені досить якісно, є інформативними та наведені у відповідних місцях роботи.

9. Якість оформлення роботи є високою.

10. Виявлені на етапі попереднього захисту недоліки оформлення пояснювальної записки та презентації усунені. Рекомендації стосовно покращення проекту враховані.

11. Ставлення студента до виконання випускної кваліфікаційної роботи – відповідальне.

12. У цілому при написанні випускної кваліфікаційної роботи студент Ковальчук І.О. виявив здібності до оволодіння знаннями, уміннями і навичками з таких дисциплін, як «Комп'ютерні мережі», «Архітектура та технології IoT», «Адміністрування комп'ютерних систем та мереж», «Хмарні технології». Також студент Ковальчук І.О. показав вміння підібрати необхідну літературу та інші інформаційні джерела, проаналізувати їх, виконати постановку задачі на розробку, розробити проект розумного будинку, обрати відповідний інструментарій, створити закінчений цілісний продукт (у вигляді змодельованого проекту) і оформити свої розробки та напрацювання як випускну кваліфікаційну роботу.

Випускна кваліфікаційна робота студента Ковальчук І.О. написана на високому рівні та може бути допущена до захисту в Екзаменаційній комісії.

Пропонована оцінка – «відмінно».

Науковий керівник
старший викладач кафедри КІ та КБ — І.А. Оринчак
19 червня 2020 року

РЕЦЕНЗІЯ
на кваліфікаційну роботу
студента 4 курсу групи КІ-2 спеціальності 123 «Комп'ютерна інженерія»
факультету інформаційно-комп'ютерних технологій
Державного університету «Житомирська політехніка»
Ковальчука Іллі Олександровича
на тему: „Інформаційна система по вивченню протоколу MQTT для Інтернету речей"

Кваліфікаційна робота студента Ковальчука І.О. „Інформаційна система по вивченню протоколу MQTT для Інтернету речей" присвячена актуальній тематиці – розробці та впровадженню розумного будинку з використанням сучасних протоколів для їх проектування та реалізаці.

Робота містить реалізований у середовищі моделювання Cisco Packet Tracer проект розумного будинку та пояснювальну записку наступного складу: вступ, три розділи основного тексту, висновки, список використаної літератури, два додатки.

У першому розділі пояснювальної записки проведено аналіз принципів, моделей, протоколів та технологій побудови захищених комп'ютерних мереж, зокрема, розглянуто: поняття топологій мереж, структура мережної моделі OSI, основні поняття технології Ethernet, принципи функціонування протоколу маршрутизації OSPF. У цьому ж розділі наведено огляд основних технологій захисту інформації в сучасних мережах.

У другому розділі пояснювальної записки здійснено проектування захищеної комп'ютерної мережі організації, зокрема, здійснено проектування топології мережі, розроблено схему адресації мережі, розроблено підсистему захисту мережі.

У третьому розділі пояснювальної записки здійснено вибір обладнання для розробленого проекту мережі, наведено опис процесу налагодження та тестування побудованої мережі. Також у цьому розділі здійснено перевірку функціоналу підсистеми захисту інформації мережі.

Зміст роботи відповідає поставленому завданню. У цілому кваліфікаційна робота виконана на високому рівні. Тема роботи розкрита повністю. Студентом продемонстровані ґрунтовні теоретичні знання та практичні фахові навички, що отримані як під час навчання, так і у процесі виконання роботи. Матеріал пояснювальної записки викладено логічно, науково і технічно грамотно. Ілюстративний матеріал (рисунки, таблиці), що наведений у роботі, оформлений якісно і наведений у відповідних місцях. Оформлення роботи у цілому відповідає вимогам стандартів. Проект розумного будинку реалізовано у середовищі моделювання Cisco Packet Tracer. Його функціонал відповідає заявленому.

Позитивною стороною є те, що отримані результати мають практичну цінність. Недоліками роботи є наступні: наявність деяких відхилень від вимого щодо оформлення пояснювальної записки, в пояснювальній записці недостатньо описані заходи реалізації певних частин програми. Зазначені недоліки у цілому не впливають на високу якість і актуальність роботи.

У цілому кваліфікаційна робота студента Ковальчука І.О. „Інформаційна система по вивченню протоколу MQTT для Інтернету речей" відповідає вимогам рівня вищої освіти «бакалавр» спеціальності 123 «Комп'ютерна інженерія» і рекомендується до захисту в ЕК.

Пропонована оцінка роботи – «відмінно».

Рецензент,
к.т.н., доцент, зав. кафедри КН І.І. Сугоняк

19 червня 2020 р.

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп’ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра

на тему:

«Інформаційна система по вивченню протоколу MQTT для Інтернету речей»

Виконав студент 4-го курсу, групи КІ-2
спеціальності 123 «Комп’ютерна інженерія»
_____ І.О. Ковальчук

Керівник старший викладач кафедри КІ та КБ
_____ І.А. Оринчак

доцент кафедри ІПЗ, к.т.н.
_____ І.І. Сугоняк

Житомир – 2020

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА КІБЕРБЕЗПЕКИ
СПЕЦІАЛЬНІСТЬ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерної інженерії
та кібербезпеки
Єфіменко А. А.
«06» квітня 2020 р.

ЗАВДАННЯ
на випускну кваліфікаційну роботу

Студента **Ковальчук Іллі Олександровича**

Тема роботи: **Інформаційна система по вивченні протоколу MQTT для Інтернету речей**
затверджена Наказом по університету від **«06» квітня 2020 р. № 98с**
Термін здачі студентом закінченої роботи – **1 червня 2020 р.**
Вихідні дані для роботи:

1. Фізична топологія мережі
2. Маршрутизатори Cisco 7200
3. Комутатори Cisco 2960
4. Операційні системи Windows 10
5. Графічний симулятор мереж Cisco Packet Tracer

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Зав. кафедри ІПЗ, к.т.н., доцет Пулеко І. В.	06.04.2020 р.	06.04.2020 р.
2	Зав. кафедри ІПЗ, к.т.н., доцет Пулеко І. В..	27.04.2020 р.	27.04.2020 р.
3	Зав. кафедри ІПЗ, к.т.н., доцет Пулеко І. В.	18.05.2020 р.	18.05.2020 р.

Керівник Оринчак І.А.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	6 квітня 2020 – 7 квітня 2020	Виконано
2	Пошук, огляд та аналіз літературних джерел щодо журналювання подій в гетерогенних комп'ютерних мережах	8 квітня 2020 – 11 квітня 2020	Виконано
3	Аналіз технологій журналювання подій	12 квітня 2020 – 16 квітня 2020	Виконано
4	Аналіз можливостей мережного обладнання Cisco для організації журналювання подій	17 квітня 2020 – 19 квітня 2020	Виконано
5	Розробка загального проекту мережі організації	20 квітня 2020 – 24 квітня 2020	Виконано
6	Проектування підсистеми журналювання подій та вибір обладнання для її реалізації	25 квітня 2020 – 30 квітня 2020	Виконано
7	Проектування та створення сценаріїв тестування функціоналу підсистеми	5 травня 2020 – 9 травня 2020	Виконано
8	Базове налагодження пристроїв мережі	10 травня 2020 – 13 травня 2020	Виконано
9	Налагодження підсистеми журналювання подій	14 травня 2020 – 18 травня 2020	Виконано
10	Тестування роботи підсистеми журналювання подій	19 травня 2020 – 22 травня 2020	Виконано
11	Аналіз результатів тестування	23 травня 2020 – 24 травня 2020	Виконано
12	Написання пояснювальної записки	25 травня 2020 – 29 травня 2020	Виконано
13	Підготовка презентації та виступу	30 травня 2020 – 31 травня 2020	Виконано
14	Отримання відгуку керівника та рецензії на кваліфікаційну роботу	4 червня 2020 – 8 червня 2020	Виконано
15	Захист	20 червня 2020	Виконано

Студент _____ Ковальчук І.О.

Керівник _____ Оринчак І.А

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра направлена на розробку інформаційної системи по вивченню протоколу MQTT для інтернету речей. Пояснювальна записка до роботи містить 55 сторінок та 21 ілюстрацій.

Метою роботи є розробка інформаційної системи по вивченню протоколу MQTT для інтернету речей. В роботі проаналізовано засоби для створення інформаційної системи а також описано існуючі аналогічні системи. Сьогодні, у час технологічного розвитку, технології стали частиною нашого повсякденного життя в сучасному суспільстві. Створення системи розумного будинку допоможе у керуванні побутовими пристроями, системою поливу, пожежною сигналізацією використовуючи смартфон. Схема розумного будинку виконується за допомогою моделювання та конфігурації системи в Cisco Packet Tracer за допомогою розумних пристроїв Інтернету Речей (IoT). Під час тестування системи бездротової мережі IoT можна керувати та контролювати розумними пристроями через смартфон на основі заздалегідь визначених умов конфігурації. Розумний дім може підвищити енергоефективність, зменшити витрати на споживання енергії, допоможе в керуванні електронними пристроями та змінить життя на краще.

КЛЮЧОВІ СЛОВА: ЗАХИСТ, МЕРЕЖА, MQTT, RFID, ПЕРИМЕТР МЕРЕЖІ, CISCO, МАРШРУТИЗАТОР, ПРОТОКОЛ, КЛІЄНТ, БРОКЕР.

					КІК.КР.Б-123-20-ПЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Ковальчук І.О.		19.06.20	Інформаційна система по вивченню протоколу MQTT для Інтернету речей	Літ.	Арк. 4 / Аркушів 83
Керівник		Оринчак І.А.		19.06.20			
Рецензент		Сугоняк І.І.		19.06.20		ФІКТ, група КІ-2	
Зав. каф.		Єфіменко А.А.		19.06.20			

ABSTRACT

The bachelor's thesis consists of an information system for studying the MQTT protocol for the Internet of Things and an explanatory note. The explanatory note to the work contains 83 pages and 21 illustrations.

The aim of the work is to develop an information system for studying the MQTT protocol for the Internet of Things. The paper analyzes the means for creating an information system and also describes the existing analog systems. Today, during technological development, technology has become part of our daily lives in modern society. Creating a smart home system will help in the management of household appliances, irrigation system, fire alarm using a smartphone. The smart home scheme is executed by modeling and configuring the system in Cisco Packet Tracer using smart IoT devices. When testing an IoT wireless system, you can control and monitor smart devices via a smartphone based on predefined configuration conditions. A smart home can increase energy efficiency, reduce energy consumption, help control electronic devices and change lives for the better.

KEY WORDS: PROTECTION, NETWORK, MQTT, RFID, NETWORK PERIMETER, CISCO, ROUTER, PROTOCOL, CLIENT, BROKER.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

EGP – ExteriorGatewayProtocol

RIP – Routing Information Protocol

OSI – Open Systems Interconnection

TCP – Transmission Control Protocol

WAN – Wide Area Network

IS-IS – Intermediate System to Intermediate System

OSPF – Open Shortest Path First

IGP – Interior Gateway Protocol

ДМЗ – демілітаризована зона

NAT – Network Address Translation

ASA – Adaptive Security Appliances

EIGRP – Enhanced Interior Gateway Routing Protocol

UDP – User Datagram Protocol

МЕ – міжмережний екран

IP – Internet Protocol

LAN – Local Area Network

MAC – Media Access Control

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 6

ВСТУП 9

РОЗДІЛ 1 11

ОБГРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ ПО РОЗРОБЦІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВИВЧЕННЯ ПРОТОКОЛУ MQTT ... 11

1.1 Аналіз еталонної архітектури Інтернет речей 16

1.2 Аналіз аналогів програмного продукту 22

1.3. Аналіз особливостей протоколу MQTT 26

1.4. Аналіз можливостей Cisco Packet Tracer щодо моделювання протоколів Інтернету речей 28

ВИСНОВОК ДО РОЗДІЛУ 1 31

РОЗДІЛ 2 32

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ПО ВИВЧЕННЮ ПРОТОКОЛУ MQTT ДЛЯ ІНТЕРНЕТУ РЕЧЕЙ 32

2.1. Розробка структури інформаційної системи по вивченню протоколу MQTT для Інтернету речей 32

2.2. Вибір інструментів Cisco Packet Tracer для реалізації інформаційної системи по вивченню протоколу MQTT для інтернет речей 34

ВИСНОВОК ДО РОЗДІЛУ 2 41

РОЗДІЛ 3 42

ОЦІНКА ПРАЦЕЗДАТНОСТІ ТА ЕФЕКТИВНОСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПО ВИВЧЕННІ ПРОТОКОЛУ MQTT ДЛЯ ІНТЕРНЕТУ РЕЧЕЙ 42

3.1. Імітаційне моделювання IoT-системи за допомогою Cisco Packet Tracer 42

3.2. Дослідження працездатності та ефективності інформаційної системи по вивченню протоколу MQTT для Інтернету речей 47

3.3. Інструкція користувачу інформаційної системи по вивченню протоколу MQTT для Інтернету речей 49

ВИСНОВОК ДО РОЗДІЛУ 3 54

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 56

ДОДАТКИ 58

ВСТУП

Актуальність. У сучасному світі будь-який будинок або квартира складається з певного набору підсистем, що відповідають за виконання різних функцій. Залежно від типу будівлі, кількість і складність підсистем змінюється. У міру їх ускладнення контроль над ними стає все складніше, витрати на обслуговування зростають. Щоб уникнути цього починають впроваджувати систему автоматизації будинку («Розумний будинок», «Smart Home»). До недавнього часу, автоматизований центральний контроль систем всієї будівлі застосовувався тільки в комерційних будівлях більшого розміру і дорогих будинках. Як правило, включаючи тільки освітлення, опалення та системи кондиціонування, а управління було доступно тільки в певних контрольних пунктах у межах самої будівлі. Зараз ситуація змінюється і системи автоматизації будинку виходять на загальнодоступний ринок. Система домашньої автоматизації дозволяє управляти багатьма інженерними вузлами і приладами з центрального пристрою, як локально, так і віддалено. В якості керуючого пристрою може бути комп'ютер, планшет або смартфон. Ця технологія може здаватися нової і в якійсь мірі унікальною, але насправді система автоматизації використовує і поєднує вже існуючі технології.

Актуальність обумовлена використанням протоколу MQTT, який є зручним для роботи з більшістю датчиків. Даний протокол дає можливість виходити на зв’язок пристроям а також робити публікації повідомлень, що не були визначені раніше.

Метою атестаційної бакалаврської роботи є створення інформаційної системи для вивчення протоколу MQTT для Інтернет речей.

Основні задачі дослідження:

1.Провести аналіз відомих ресурсів із даної предметної галузі.

2.Провести аналіз архітектури інтернет речей.

3. Здійснити вибір методів та засобів для створення моделі.

4. Розробити інформаційну систему на базі IoT з використанням MQTT протоколу

Об’єктом дослідження є – процес створення інформаційної системи для вивчення протоколу MQTT для Інтернет речей.

Предметом дослідження – є вивчення та застосування протоколу MQTT для Інтернету Речей

Результатом атестаційної бакалаврської роботи є завершена інформаційна система для вивчення протоколу MQTT для Інтернет речей, яка готова до впровадження.

РОЗДІЛ 1
ОБГРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ ПО РОЗРОБЦІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВИВЧЕННЯ ПРОТОКОЛУ MQTT

Розумний будинок (в перекладі з англійської мови «smart house» або «intelligent building») - це сучасне житловий будинок, створене з використанням високотехнічних механізмів і пристосувань, автоматизованих для комфортного проживання в ньому.

В даний час Інтернет речей є загальновизнаною концепцією розвитку мереж зв'язку в короткостроковій і довгостроковій перспективах, а також передовою платформою в рамках розвитку цифрового інтелекту в концепції «Розумна країна». На думку більшості консалтингових аналітичних компаній, протягом наступних п'яти років в кожній зі сфер життєдіяльності людини будуть присутні понад 25 мільярдів пристроїв. Таким чином, можна говорити про характер проникнення Інтернету речей в наше повсякденне життя. Як відомо, фраза «Інтернет речей» вперше прозвучала від Кевіна Ештона в 1999 році на презентації інноваційних рішень компанії «Проктер і Гембл». Ештон запропонував нанести RFID мітки на продукцію, що випускаються компанією, і таким чином забезпечити її взаємодію з радіоприймачем. Кевін Ештон припустив, що такий збір даних може бути використаний для вирішення багатьох проблем в реальному світі. В результаті в даний час багато пристроїв можуть обмінюватися даними через Інтернет, взаємодіючи з смартфонами, один з одним і з аналогічними схожими пристроями. У 2001 році дослідницький центр Auto-ID Массачусетського технологічного інституту, в якому працював Кевін Ештон, адаптував використання RFID міток для різноманітної продукції, місцезнаходження якої стало можливо відслідковувати через Інтернет.

У 2005 році термін Інтернет речей був офіційно використаний Міжнародним союзом електрозв'язку (МСЕ) в технічному звіті, присвяченому перспективним концепціям розвитку мереж зв'язку. В останнє десятиліття

Інтернет речей став однією з проривних технологій, загальновизнаних усіма країнами світу. Інтернет речей дозволяє людям і речам взаємодіяти де завгодно, коли завгодно, і в будь-яких поєднаннях при використанні інфраструктури Інтернету речей. Екосистема Інтернету речей передбачає збір даних з датчиків (або відправку команд на виконавчі пристрої), їх передачу через мережу зв'язку на хмарні платформи для подальшого аналізу з метою надання інтелектуальних послуг для людей.

У старі часи людина прагнула до форми розумного будинку ще з часів Єгипту і того раніше, вигляді захисних механізмів, схованок і інших прихованих механічних зручностей. Сама ідея автоматизації як комфорту і захищеності людини існувала завжди вигляді інженерної думки і ідеї її удосконалення. Але прорив на основі електроніки та її можливостей була створена американськими компаніями Leviton і X10-USAрозробили і запустили в масове виробництво оригінальну кабельну технологію X10, яка дозволила управляти домашніми електроприладами електричної мережі. Це був справжній прорив на території окремо взятої держави, оскільки технологія була призначена для роботи виключно в американських мережах. що стало офіційною датою народження системи розумний будинок вважається 1978 рік.

Передбачається, що дана система вміє розпізнавати конкретні ситуації, що відбуваються в будинку, і реагує на них. Система функціонує як комплекс або окрема структура, відповідно до заданих завданнями і умовами доступності, об'єднуючи важливі захисні і комфортні умови. Розумний будинок є напрямком майбутнього будь-якої високотехнологічної країни, де всі верстви суспільства будуть переходити на зручність житлових, офісних, виробничих і військових структур держави. Але головним завданням будить постійне і безперебійне енергопостачання всіх цих структур. У світі існує багато різних способів як підтримувати так і забезпечувати розумні будинки всім не обхідних. Головним завданням майбутнього і теперішнього часу будить повна автоматизація контрольована системою управління людиною, центрами логічного модуля, і

програмами штучного інтелекту або його прототипу версій нинішнього часу програми у відповідь питальній системи.

З розвитком прогресу з'явилася можливість управляти розумними будинками через всі доступні засоби зв'язку, це комп'ютери, мобільні телефони, панелі управління, планшети та інших видів гаджетів. Що означає повне впровадження в повсякденне життя пристроїв здатних працювати з розумним будинком не тільки на його території, а й за його межами.

Взаємодія людини з житловим простором дає якісно нових рівнів життя, на якому відповідно до внутрішніх і зовнішніх умов, за допомогою автоматизації відслідковуються і задаються режими роботи всіх приладів і систем. Комплекс робить життя людей безпечної, комфортної в рамках житлового простору та інших структур суспільства.

І все це необхідно забезпечувати енергією, що живить ці системи різні джерела живлення, в тому числі й альтернативні джерела живлення, які цілком стануть повсякденними життя. Такі, як сонячні батареї, електричні генератори, вітряна турбіна, відходи життєдіяльності людини, і будь-який інший спосіб, що здатний сприяти електропостачання розумних будинків.

В умовах обмеженого і непоновлюваного енергоресурсу на планеті або окремої країни проблема раціонального логічного використання вироблюваної електричної енергії набуває значної актуальності. Технології сьогодення і майбутнього часу так званого людством «розумний дім» дозволяють автоматизувати управління системами всіх електричних приладів будинку, здійснюючи комфортабельність, захист і значні економічні запаси електроенергії. Основними підсистемами розумного будинку, які надають на енергозбереження є:

- управління електричним освітленням;

- автоматизована системи прогріву (кондиціонери, автоматизовані газові котли індивідуального прогріву);

- контроль і регулювання іншими інженерними думками та комунікаціями. Основний перевагою економії електричної енергії є:

- збільшення ефективності використання електричної енергії;

- зменшення не раціонального споживання електроенергії;

- пошук і використання альтернативних джерел електроенергії. Збільшення ефективного споживання електричної енергії може бути отримано заміщенням електрообладнання на більш енергоефективні пристрої. Всі ці пристрої дозволяють відчутно знизити загальне енергоспоживання.

Сьогодні до 10 - 15% вироблюваної в світі електричної енергії споживається на освітлення. Скоротити розрив споживання електричної енергії на ці завдання можна шляхом більш раціонального її застосування.

Актуальність інтелектуальної інноваційної системи «Розумний Дім» для нашої країни на великому і малому ринку важко переоцінити значимо. Використовуючи культивацію інтелектуального розумного будинку, можна досягти 20 - 30% економії на непристойності комунальних послуг. Цей значимий факт готові сприйняти до уваги і творці будівель, які передбачають застосування технології розумного будинку в 30% проектованих будинків. Корисні можливості «розумного будинку» роблять набагато стійкішими і довговічними на століття всю систему життєзабезпечення людини, завдяки оптимізації якісних зв’язків між ними. Якщо ж згадувати про нові будівлі - то використання «розумного будинку» знімає можливі конфлікти між розумними системами, збільшує термін потенційної служби всієї будівлі, зменшуючи ризик для творців будівель і проживаючих осіб.

MQTT (відомий як Message Queueing Telemetry Transport) – це простий швидкодіючий протокол зв’язку, який працює на поверхні TCP/IP, розроблений для Інтернету Речей. Протокол створений бути простим протоколом обміну повідомленнями через брокера по схемі публікації/підписки, при розробці якого переслідується ціль забезпечити відкритість, простоту, мінімальні потреби до ресурсів та зручність умонтування. З моменту його появи ціль «зручність використання» була досягнута, так як були розроблені різні бібліотеки для умонтування клієнтів MQTT.

Найбільш відома реалізація брокера - Mosquitto від Eclipse. Платформа реалізує можливість обміну повідомленнями між компонентами системи на базі стандартів протоколу MQTT. Mosquitto написаний на C і поширюється під ліцензією EPL / EDL. Підтримуються операційні системи Windows, Mac і Linux.

Реалізацію за схемою публікації / підписки виконує бібліотека Paho також від Eclipse. Підтримуються всі основні популярні мови програмування: C / C ++, Java, JavaScript, Python, Go, C #.

HTTP найпопулярніший і широко використовуваний протокол для комунікації пристроїв. Перш ніж вирішувати свої завдання вибір був на користь HTTP, але з ростом популярності Інтернету речей і MQTT, змусило розібратися в цьому питанні. На сьогоднішній день продуктивність пристроїв для Інтернету речей досить невисока порівняно з високопродуктивними веб-серверами. Це стосується насамперед до Arduino платформ і популярному мікроконтролеру ESP8266. RaspberryPi набагато потужніший, однак якщо зв'язок підтримується тільки через мобільний інтернет, то і в цьому випадку можуть виникати проблеми.

Головними перевагами MQTT протоколу в даному випадку є його легковажність (MQTT передає дані як масив байтів) і використання схеми видавець-підписник при обміні повідомленнями. Тема повідомлень MQTT коротка, а найменший розмір пакету повідомлень становить 2 байта. Використання текстового формату повідомлень в HTTP протоколі дозволяє створювати довші заголовки і повідомлення, які зрозумілі людині. Це допомагає позбутися від помилок, але в той же час це зайве для пристроїв з обмеженими ресурсами.

1.1 Аналіз еталонної архітектури Інтернет речей

Незважаючи на те, що термін "Інтернет речей" виник нещодавно, вже існує безліч визначення цього терміну. «Інтернет речі» -це сімейство фізичних предметів обчислювальної мережі, оснащених технологіями, які сприяють взаємодією один з одним або навколишнім світом (зовнішнім середовищем). сама суть ідеї полягає в тому, що інтернет речі зможу стати активними учасниками (в деяких сферах навіть замінювати людей) в сфері бізнесу, передачі інформації, реагуючи на зовнішні процеси щосекунди (рис.1).

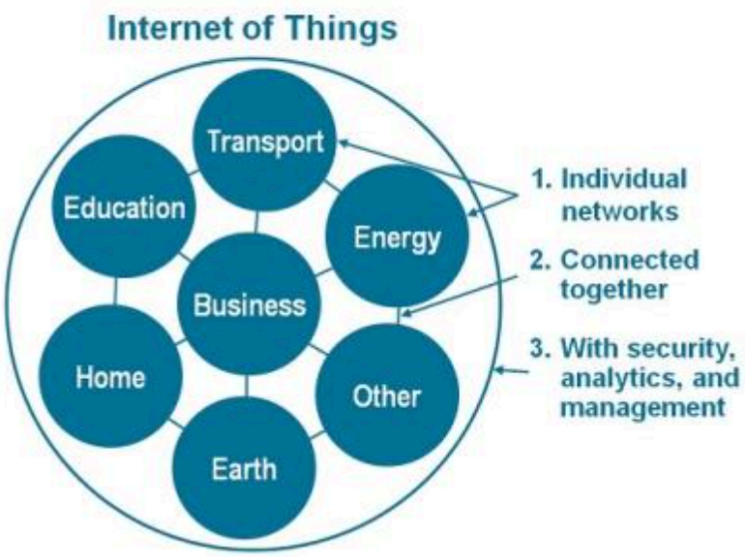


Рисунок 1.1 – Діаграма функціонального розподілення «Інтернет-речей»

Існує думка, що інтерне-речі представляють собою чотирьохшаровий пиріг:

1. Ідентифікація кожної інтернет-речі (об'єкта).
2. Сервіс з обслуговування споживчих потреб (приклад - «розумний дім»).
3. Урбанізація міського життя.

Всі Інтернет речі створюються для бізнесу, важливим є мінімізація витрат та максимальна ефективність.

Еталонна модель IoT не обмежує область дії або місцезнаходження її компонентів. Наприклад з фізичної точки зору, кожен елемент може знаходитись

в одному наборі обладнання або поширюючись по всьому світу. Еталонна модель IoT також дозволяє виконувати обробку, що відбувається на кожному рівні від тривіального до складного, залежно від ситуації. Модель описує як вирішувати завдання, щоб зберегти простоту, дозволити високу масштабованість та забезпечити підтримку. Також модель визначає функції, необхідні для завершення системи IoT.

Еталонна модель IoT починається з рівня 1: фізичні пристрої та контролери, які можуть керувати кількома пристроями. Це "речі" в IoT і вони включають широкий спектр пристроїв кінцевих точок, які надсилають та отримують інформацію. Сьогодні перелік пристроїв вже великий. Це стане майже необмеженим, оскільки з часом в IoT додається більше обладнання. Пристрої різноманітні, і немає правил щодо розміру, розташування, фактор форми чи походження. Деякі пристрої будуть розміром із силіконового чіпу. Деякі будуть такі великі, як транспортні засоби. IoT повинен підтримувати весь діапазон. Десятки чи сотні виробників обладнання вироблятимуть пристрої IoT. Для спрощення сумісності та підтримки технологічності, еталонна модель IoT зазвичай описує рівень обробки, необхідний для пристроїв рівня 1.

Комунікаційний зв’язок зосереджений на другому рівні. Найважливішою функцією другого рівня є надійна, своєчасна передача інформації між пристроями та мережею, по всій мережі (схід-захід), між мережею та обробкою інформації низького рівня, що відбувається на третьому рівні. Традиційні мережі передачі даних мають багато функцій, про що свідчить семи шарова еталонна модель міжнародної організації зі стандартизації (ISO).

Однак повна IoT система містить безліч рівнів, крім мережі зв'язку. Однією метою базової моделі IoT є здійснення зв'язку та обробки існуючими мережами. Еталонна модель IoT не вимагає і не вказує на створення іншої мережі - вона покладається на існуючі мережі. Однак деякі застарілі пристрої не підтримують IP-адресу, для чого знадобиться запровадити шлюзи зв'язку. Інші пристрої потребують власників контролерів для обслуговування функції зв'язку. Однак з часом стандартизація буде зростати. Оскільки пристрої першого рівня

поширюються, способи взаємодії з обладнанням другого рівня можуть змінюватися. Незалежно від деталей, пристрої першого рівня спілкуються через систему IoT, взаємодіючи з обладнанням другого рівня.

Функції третього рівня зумовлені необхідністю перетворення мережевих потоків даних у інформацію, що підходить для зберігання та обробки вищого рівня на четвертому рівні (накопичення даних). Це означає, що діяльність 3-го рівня зосереджена на високому обсязі аналізу та перетворення даних. Наприклад, датчик першого рівня може створювати зразки даних кілька разів в секунду, 24 години на день, 365 днів на рік. Основним принципом еталонної моделі IoT є те, що найрозумніша система ініціює обробку інформації якомога раніше та якомога ближче до краю мережі. Іноді, це називають обчисленням туману. Рівень 3 - це місце, де це відбувається. Зважаючи на те, що дані, як правило, подаються на мережеве обладнання рівня пристроями в невеликих одиницях, обробка третього рівня виконується на основі пакетів. Ця обробка обмежена, оскільки існує лише обізнаність про одиниці даних, а не про "сеанси" чи "транзакції". Обробка третього рівня може охоплювати безліч прикладів, таких як:

● Оцінка: Оцінка даних за критеріями щодо того, чи слід обробляти їх на більш високому рівні.

● Форматування: переформатування даних для послідовної обробки вищого рівня.

● Розширення / декодування: обробка крипто валютних даних з додатковим контекстом (наприклад, джерелом).

● Дистиляція / зменшення: зменшення та / або узагальнення даних для мінімізації впливу даних та трафіку на мережу та системи обробки вищого рівня.

● Оцінка: Визначення, чи є дані пороговими або попереджувальними; це може включати перенаправлення даних до додаткових напрямків.

Мережеві системи побудовані для надійного переміщення даних. Дані "у русі". До четвертого рівня дані переміщуються по мережі зі швидкістю та організацією, що визначаються пристроями та генерують дані. Модель керується подіями. Як було визначено раніше, пристрої першого рівня не містять самих обчислювальних можливостей. Однак, деяка обчислювальна діяльність може відбуватися на другому рівні, наприклад, переклад протоколу або застосування політики безпеки мережі. На третьому рівні можна виконати додаткові обчислювальні завдання, такі як перевірка пакетів. Приведення обчислювальних завдань якомога ближче до краю IoT, наскільки це можливо, з неоднорідними системами, розподіленими по декількох областях управління, є прикладом обчислення туману. Обчислення туману та туманні послуги будуть відмінною характеристикою IoT. Більшість програм не можуть або не потребують обробки даних зі швидкістю мережевого проводу. Програми, як правило, припускають, що дані знаходяться в стані спокою - або незмінними - в пам'яті або на диску. На четвертому рівні, накопичення даних, дані в русі перетворюються на дані в спокої. Рівень 4 визначає:

● Якщо дані цікавлять більш високі рівні: якщо так, обробка 4 рівня - це перший рівень, який налаштований на задоволення конкретних потреб більш високого рівня.

● Якщо дані потрібно зберігати: чи слід зберігати дані на диску в енергонезалежному стані або накопичувати в пам'яті для короткочасного використання?

● Тип необхідного зберігання: чи потрібна стійкість файлової системи, великої системи даних або реляційної бази даних?

● Якщо дані організовані належним чином: чи належним чином організовані дані для потрібної системи зберігання?

● Якщо дані повинні бути рекомбіновані або перераховані: дані можуть бути об'єднані, перераховані або агреговані з раніше збереженою інформацією, частина з якої може надходити з джерел, які не є IoT.

Оскільки четвертий рівень фіксує дані та ставить їх у спокій, тепер вони можуть використовуватися додатками в режимі реального часу. Програми отримують доступ до даних, коли це необхідно. Коротше кажучи, четвертий рівень перетворює дані на основі подій в обробку на основі запитів. Це найважливіший крок для подолання відмінностей між мережею реального часу та світом додатків у режимі реального часу.

Системам IoT потрібно буде наближатись до корпоративного (а то й глобального) рівня та потребуватиме декількох систем зберігання даних для розміщення даних та даних пристроїв IoT від традиційних корпоративних систем ERP, HRMS, CRM та інших систем. Функції абстрагування даних пятого рівня орієнтовані на рендеринг даних та їх зберігання способами, що дозволяють розробляти більш прості додатки, що підвищують продуктивність.

У багатьох пристроях, що генерують дані, є багато причин, через які ці дані можуть не знаходитися в одному сховищі даних:

● У одному місці може бути занадто багато даних.

● Переміщення даних у базу даних може споживати занадто багато потужностей для обробки, тому їх отримання має бути відокремлене від процесу генерації даних. Це робиться сьогодні за допомогою онлайнових баз даних обробки даних (OLTP) та сховищ даних.

● Пристрої можуть бути розділені географічно, а обробка оптимізована локально.

● Рівні 3 і 4 можуть відокремлювати "безперервні потоки необроблених даних" від "даних, що представляють подію". Зберігання даних для потокової передачі даних може бути великою системою даних, наприклад Hadoop. Зберігання даних про події може бути реляційною системою управління базами даних (RDBMS) з більш швидким часом запитів.

● Можливо, потрібні різні види обробки даних. Наприклад, обробка в магазині буде зосереджена на різних речах, ніж обробка підсумків у всіх магазинах. З цих причин рівень абстрагування даних повинен обробляти багато різних речей. До них належать:

● Узгодження декількох форматів даних з різних джерел

● Забезпечення послідовної семантики даних між джерелами

● Підтвердження того, що дані є повними для програми вищого рівня

● Консолідація даних в одному місці (за допомогою ETL, ELT або реплікації даних) або надання доступу до кількох сховищ даних за допомогою віртуалізації даних

● Захист даних відповідною автентифікацією та авторизацією

● Нормалізація або денормалізація та індексація даних для забезпечення швидкого доступу до додатків.

Рівень 6 – рівень програми, де відбувається інтерпретація інформації. Програмне забезпечення на цьому рівні взаємодіє з п’ятим рівнем та даними в спокої, тому йому не доведеться працювати на швидкості мережі. Довідкова модель IoT не чітко визначає додаток. Програми залежать від вертикальних ринків, характеру даних про пристрої та потреб бізнесу. Наприклад, деякі програми будуть зосереджені на даних пристроїв контролю. Деякі будуть зосереджені на керуванні пристроями. Деякі об’єднують дані пристрою та пристрої, які не є пристроєм. Моніторинг та контроль програми представляють багато різних прикладних моделей, моделей програмування та стеків програмного забезпечення, що веде до обговорення операційних систем, мобільності, серверів прикладних програм, гіпервізорів і т.д. Досить сказати, що складність програми буде дуже різнитися. Такими прикладами є:

● Найважливіші бізнес-програми, такі як узагальнені ERP або спеціалізовані галузеві рішення

● Мобільні додатки, які керують простими взаємодіями

● Звіти про бізнес-аналітику, де додатком є BI-сервер

● Аналітичні програми, які інтерпретують дані для бізнес-рішень

● Додатки центру управління / контролю системи, які керують самою системою IoT і не діють на дані отримані ними [1].

Якщо рівні 1-5 сконструйовані належним чином, обсяг робіт, необхідних рівню 6, зменшиться. Якщо рівень 6 розроблений належним чином, користувачі зможуть виконувати свою роботу краще.

Система IoT та створена ним інформація мають малу цінність, якщо вона не приносить дій, які вимагаються від людей та процесів.

Програми виконують бізнес-логіку для розширення можливостей людей. Люди використовують програми та пов'язані з ними дані для своїх конкретних потреб. Часто кілька людей використовують один і той самий додаток для різних цілей. Отже, мета не в застосуванні, а в тому, щоб дати можливість людям краще робити свою роботу. Програми (рівень 6) надають діловим людям потрібні дані в потрібний час, щоб вони могли робити правильно. Але часто потрібні дії потребують більше однієї людини. Люди повинні вміти спілкуватися та співпрацювати, іноді використовуючи традиційний Інтернет, щоб зробити IoT корисним. Для спілкування та співпраці часто потрібні кілька кроків. І це зазвичай перевершує кілька додатків [2].

1.2 Аналіз аналогів програмного продукту

Багатофункціональні системи «розумний будинок», які забезпечують комфорт та безпеку житла, з кожним роком набирають все більшу і більшу популярність.

Перш за все, це пов'язано з підвищенням технологічної грамотності простих людей на фоні розвитку цифрової індустрії.

По-друге, таке обладнання поступово дешевшає, що робить його ще більш доступним для широкої маси населення [3].

Сучасний ринок пропонує великий вибір технічних пристроїв для автоматизації житла, тому порівняння вже існуючих систем «розумний будинок» є необхідним для створення якісної системи.

Ajax

Дана система автоматизації будинку в повній мірі справляється відразу з двома важливими задачами: забезпечує комфорт та зручність в управлінні

будинком, гарантує безпеку житла в повній мірі, контролюючи об’єкт злому а також пожежну, газову, електричну загрози для будинку.

Перевагами даної системи є: простий монтаж, наявність безпровідних каналів зв’язку між елементами системи, обширна зона дії сигналу (до 2000м), наявність захисту від посторонніх на датчиках, автономна робота від акумулятора, різноманітність варіантів сповіщення користувача, керування за допомогою смартфону.



Рисунок 1.2 – Система «Ajax»

Недоліками системи є: функціонування тільки з роботою центрального контроллера, тобто відсутність автономності датчиків, немає особистої камери (є можливість підключення сторонньої камери), управління тільки через смартфон (хоча це знімає необхідність встановлення додаткового ПЗ на комп'ютер).

BroadLink

Дана система являє собою комплект цифрових пристроїв, створених для раціонального управління побутовою технікою а також освітленням, сигналізацією та іншими системами у будинку. Кожен елемент цієї системи може працювати як самостійно так і взаємодіяти один з одним.

Перевагами даної системи є: швидке встановлення, має великий вибір датчиків (вологості, температури, освітлення, шуму, забруднення повітря),

можна додавати та забирати датчики, функціонує без центрального контроллера, є своя камера відеонагляду, демократична ціна (~200$).



Рисунок 1.3 – система «BroadLink»

Недоліками є: невелика дальність дії (лише 50м), відсутність резервного живлення, пульт працює тільки на прийом сигналів.

Fibaro

Дана система відноситься до професійного обладнання по забезпеченню автоматизації та безпеки будинку з широким функціоналом. Перевагами даної системи є: величезна кількість різноманітних датчиків, наявність камери відеонагляду, розсилка повідомлень відразу на декілька мобільних пристроїв, розумна розетка, яка показує кількість спожитої електроенергії, голосове управління (англійською).



Рисунок 1.4 – система «Fibaro»

Недоліками даної системи є: вартість (~600$), тільки професійне встановлення та налаштування, відсутність резервного живлення, не функціонує без центрального контроллера, підключення центрального контроллера до мережі через LAN.

Orvibo

Не дорога система проста в експлуатації, головна мета якої забезпечення безпеки в будинку. Перевагами цієї системи є: просте встановлення та налаштування, керування смартфоном, широкий вибір додаткових датчиків (понад 100), вибір сценаріїв для роботи з технікою у будинку, доступна ціна (~150$).



Рисунок 1.5 – Система «Orvibo»

Недоліками даної системи є: маленький радіус дії (лише 30м), досить слабкий стартовий комплект, провідне підключення.

1.3. Аналіз особливостей протоколу MQTT

Telemetry Transport Queue Message (MQTT) – протокол, розроблений IBM, та призначений для забезпечення легких комунікацій M2M. Це асинхронний протокол публікації/підписки, який працює поверх стеку TCP. Спочатку він був розроблений Енді Стенфорд-Кларком (IBM) та Арленом Ніппером у 1999 році. Після внутрішнього використання у IBM він був оприлюднений у 2010 році. MQTT був розроблений простим у застосуванні на обмежених пристроях, що характеризується дефіцитом обчислювальної потужності та мережевими середовищами з обмеженою пропускною здатністю. Його характеристики роблять даний протокол гарним вибором для використання в обмежених середовищах та семантичних зв’язках типу "машина до машини" (M2M) чи "Інтернет-речей". Протоколи публікації/підписки краще відповідають вимогам IP, ніж запит/відповідь, оскільки клієнтам не потрібно вимагати оновлення. Таким чином, пропускна здатність мережі зменшується, а потреба у використанні обчислювальних ресурсів знижується [4].

У MQTT виробники даних можуть публікувати повідомлення на тему, а споживачі даних можуть підписатися на отримання повідомлень, опублікованих на цю тему. Ці взаємодії опосередковуються та відокремлюються через брокера, тому видавцям і абонентам потрібно лише знати про брокера, а не про інших виробників та споживачів. Теми визначаються за допомогою UTF-8 рядків, що мають структуру дерева, в якій рівні розділені за допомогою косою рисою ('/'). Наприклад, сенсор світла в вітальні будинку можна було б моделювати за темою "home/firstfloor/livingRoom/lightSensor". Температурному датчику у тій же вітальній кімнаті може бути присвоєно теми "home/firstfloor/livingRoom/tempSe nsor". Додаток, зацікавлений в цих даних, може підписатися на ці теми. MQTT

визначає підстановки для підписки на декілька тем у рівнях дерева тем. Багатоступенева підстановка "#" може бути використана для відповідності будь-якій кількості рівнів у межах теми, а шаблон "+" відповідає лише одному рівні теми. У нашому прикладі можливо підписатися на тему "house/firstfloor/#", щоб підписати всі теми, що включають піддерево "house/firstfloor". Це включатиме т акі теми як "house/firstfloor/kitchen/tempSensor", а також тему "house/firstfloor/liv ingRoom/wall/sensor". Для ілюстрації використання однорівневої підстановки ро зглянемо тему "house/firstfloor/+/tempSensor", яка відповідає темі "house/firstfloo r/kitchen/tempSensor", але не темою з додатковими рівнями, як наприклад "house /firstfloor/livingRoom/northwall/motionSensor".

MQTT визначає пакети керування, що складається з фіксованого заголовку, необов'язкового змінюваного за розміром заголовка та корисного навантаження. У контрольних пакетах дані можуть бути представлені з використанням бітів, 16- бітових цілих чисел у великому розмірі або UDF-8-закодованих рядків. Корисні завантаження надсилаються у форматі байтів, а кодування та декодування виконуються програмою.

MQTT забезпечує надійність, забезпечуючи можливість вибору трьох рівнів QoS:

1. *Відправити та забути:* повідомлення відправляється один раз, і підтвердження не потрібно.

2. *Поставляється щонайменше один раз:* повідомлення надсилається принаймні один раз, а підтвердження потрібно.

3. *Доставлено рівно один раз:* Механізм чотирьохстороннього рукостискання використовується для забезпечення доставки повідомлення рівно один раз.

Хоча MQTT працює на базі TCP, він розроблений для того, щоб мати низькі накладні витрати порівняно з іншими протоколами основаними на протоколі TCP. Крім того, архітектура опублікування/підписки, яку вона використовує, більше підходить для IP, ніж HTTP запит/відповідь, наприклад, тому що повідомлення мають отримати відповіді. Це призводить до нижчої пропускної

здатності мережі та меншу обробку повідомлень, що фактично збільшує термін служби пристроїв, що працюють на батареї.

Для забезпечення безпеки, брокери MQTT можуть вимагати перевірки автентичності на ім'я користувача та пароль, які обробляються TLS/SSL (Secure Sockets Layer), тобто ті самі протоколи захисту, які забезпечують конфіденційність HTTP-транзакцій у всьому Інтернеті.

1.4. Аналіз можливостей Cisco Packet Tracer щодо моделювання протоколів Інтернету речей

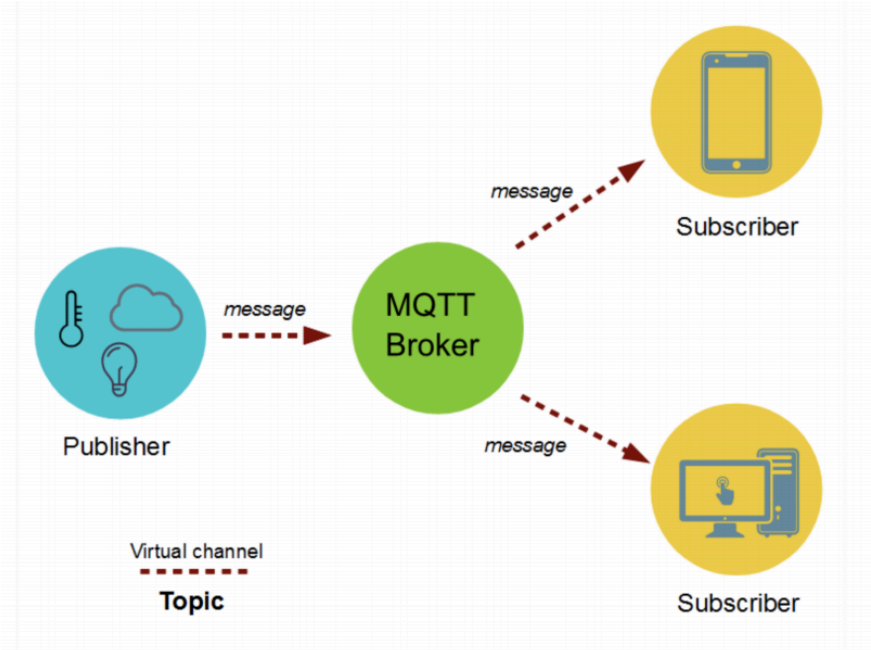


Рисунок 1.6 – Взаємодія MQTT клієнтів

Підхід до мереж розумних пристроїв було розглянуто ще в 1982 році, коли було створено машину «Coca-Cola» в університеті Карнегі Мелона. Дана машина була першою підключена до інтернету, та мала змогу повідомити про свій каталог а також, про те чи були завантажені напої. Однак, термін «Інтернет Речей» було винайдено британським підприємцем Кевіном Ештоном, виконавчим директором Auto-ID у 1999 році.

Cisco Packet Tracer – це інноваційний інструмент для моделювання та візуалізації мережі та пристроїв IoT. Як безкоштовне програмне забезпечення воно дозволяє моделювати, практикувати у підключенні пристроїв IoT, налаштовувати мережі та усувати неполадки.

Починаючи з 7 версії програми Cisco Packet Tracery програмі з'явилась підтримка Інтернет Речей. Інтернет речей – це концепція мережі, яка складається з пристроїв, які здійснюють обмін даними між фізичним світом і комп'ютерними системами, за допомогою використання стандартних протоколів зв'язку.

Пропонується така методика моделювання IoT проектів «розумного» будинку в Cisco Packet Tracer:

Етап 1. Підготовка проектного завдання. На цьому етапі отримуються креслення будинку та уточнюються вимоги замовника до технічних та вартісних характеристик IoT-пристроїв.

Етап 2. Налаштування середовища моделювання. Креслення будинку у вигляді графічного файлу завантажується в Cisco Packet Tracer. Налаштовуються моделі факторів навколишнього середовища.

Етап 3. Підбір із набору візуальних моделей IoT-пристроїв, що відповідають реальним вимогам замовника та розміщення їх на кресленні з урахуванням особливостей будинку. Програмне забезпечення має досить широкий набір моделей пристроїв, але у разі, якщо адекватної моделі не виявиться, можна створити і нову модель.

Етап 4. Формування та моделювання роботи підсистем мережі. На цьому етапі враховуючи особливості розміщення IoT-пристроїв відносно елементів конструкції будинку формуються кабельні чи бездротові з'єднання мережі. Підбираються і налаштовуються комутатори, маршрутизатори та домашні шлюзи. Зокрема налаштовується IP адресація для датчиків, та параметри безпеки маршрутизатора шлюзу. Весь проект доцільно розбити на типові підсистеми: пожежної безпеки, захисту від проникнення, освітлення, створення мікроклімату та інші.

Етап 5.Моделюються зовнішні підключення локальної мережі. Перевіряється, чи всі пристрої у домені мають IP-з'єднання.

Етап 6. Відпрацювання логіки та правил управління. Під час виконання цього етапу відпрацьовуються правила взаємодії для безпечної експлуатації підсистем та пристроїв.

Етап 7. Попередня оцінка вартості проекту. За встановленими елементами моделі підбираються технічні пристрої та системи і розраховується приблизна вартість системи.

Етап 8. Погодження моделі та вартості системи замовником. Після відпрацювання технічних питань на моделі, вона подається замовнику на погодження. У разі погодження моделі та вартості системи, розробляється технічна документація та здійснюється реалізація проекту. Якщо проект потребує доробок, то повертаємось до 3 етапу і проводимо моделювання знову. На рис. 1.7. подано приклад реалізації моделі «розумного» будинку. Така модель має велику функціональність та гарну візуалізацію .

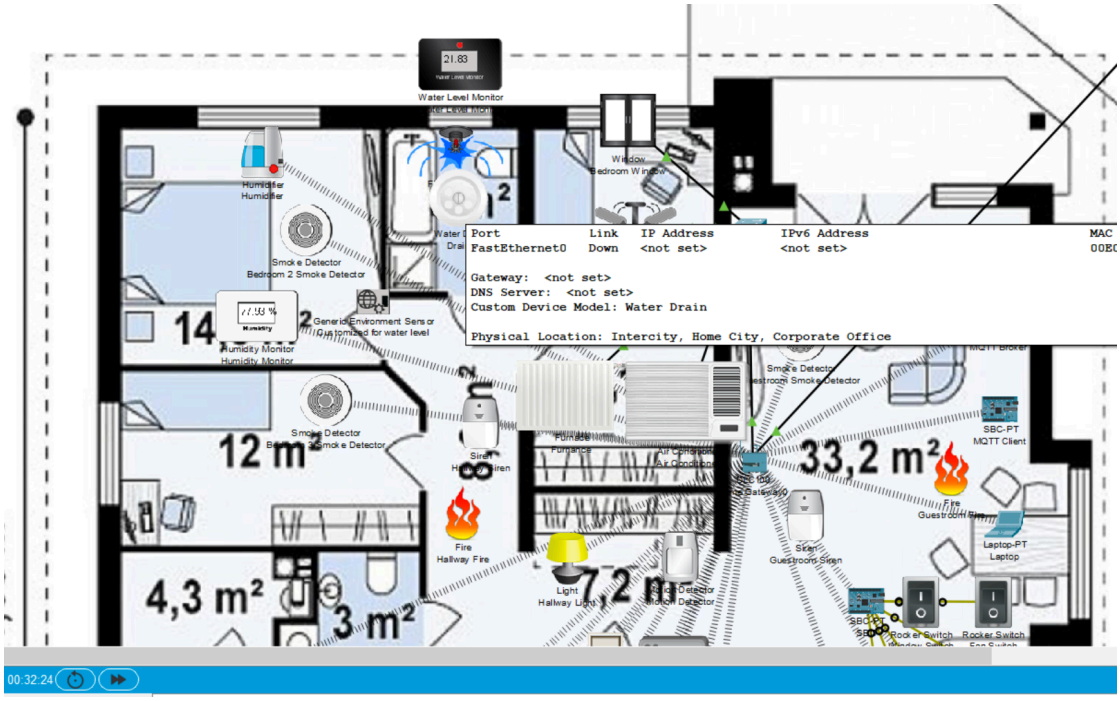


Рисунок. 1.7 – Приклад робочого вікна моделі проекту «розумного» будинку

Завдяки добре розвиненому функціоналу програмного забезпечення Cisco Packet Tracer моделювання проекту можна здійснити за невеликий час (декілька годин), що дозволить швидко погодити проект із замовником.

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі випускної кваліфікаційної роботи бакалавра було описано та проаналізовано літературні джерела. Було розглянуто структуру організації розумних будинків з використанням протоколу MQTT, описано методи моделювання розумних будинків засобами Cisco Packet Tracer. Було спроектовано схему взаємодії об’єктів у протоколі MQTT. Також, було проведено аналіз еталонної архітектури Інтернет речей. В результаті аналізу літературних джерел було встановлено, що впровадження розумного будинку набуває популярності а також використання сучасних протоколів для їх проектування та реалізації є дуже важливим. Отож, після огляду літературних джерел було вирішено обрати Cisco Packet Tracer як головний інструмент для вирішення поставленої задачі.

РОЗДІЛ 2
РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ПО ВИВЧЕННЮ ПРОТОКОЛУ MQTT ДЛЯ ІНТЕРНЕТУ РЕЧЕЙ

2.1. Розробка структури інформаційної системи по вивченню протоколу MQTT для Інтернету речей



Рисунок 2.1 – Схема взаємодії MQTT

Схема взаємодії між підписником, видавником та брокером зображена на рис. 2.1.

Як було зазначено до цього, протокол MQTT реалізовує парадигму публікація-підписка (publish-subscriber). Ця парадигма дозволяє відокремити клієнта, що публікує повідомлення (publisher) від інших клієнтів, що отримуються ці повідомлення (subscribers). Більше того, MQTT є асинхронним протоколом, а це означає, що при його використанні непотрібно блокувати клієнтів, які очікують на повідомлення. Іншою цікавою властивістю MQTT є те, що він не потребує, щоб клієнт, який публікує повідомлення та клієнт, який приймає повідомлення, були одночасно підключені один до одного [9].

Зазвичай, архітектура протоколу MQTT (рис. 2.2.) може бути розділена на два головних компонента:

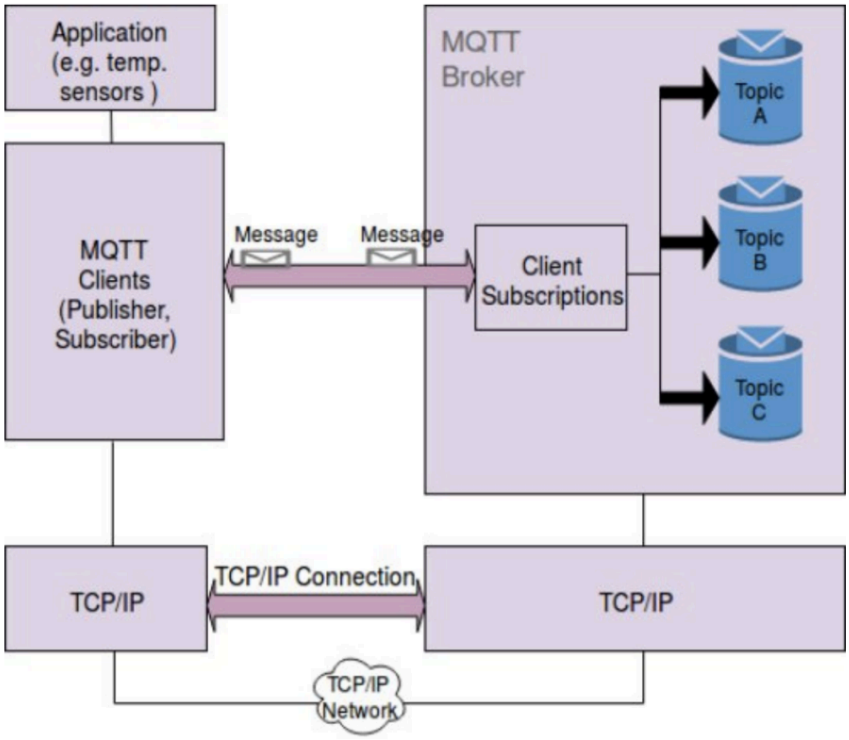


Рисунок 2.2 – Архітектура MQTT

1. Клієнт може бути у ролі відправника або отримувача повідомлень, при цьому він завжди встановлює з'єднання з сервером (брокером повідомлень). Клієнт може виконувати наступні функції:

- публікувати повідомлення для зацікавлених користувачів
- підписуватись на відповідні теми для отримання повідомлень
- відписуватись від тем
- від'єднатись від брокера повідомлень

2. Брокер повідомлень контролює поширення інформації і є відповідальним за прийом повідомлень від публікуючих клієнтів, фільтрування повідомлення та їх надсилання всім підписаним клієнтам [10]. Брокер може виконувати наступні функції:

- встановлювати з'єднання з клієнтами
- отримувати повідомлення від клієнтів
- правильно оброблювати різні види запитів, зокрема підписки та відписки від користувачів.

- після отримування повідомлень від видавника надсилати їх всім зацікавленим користувачам.

2.2. Вибір інструментів Cisco Packet Tracer для реалізації інформаційної системи по вивченню протоколу MQTT для інтернет речей

Поряд зі стандартними функціональностями мережі, у версії 7.0.0 Cisco Packet Tracer отримав важливе оновлення з компонентами IoT

Цей розділ в основному присвячений функціональності IoT інструментів, короткому огляду мережевих компонентів.

На малюнку нижче можна побачити, як приклад, різні маршрутизатори, пропоновані Cisco Packet Tracer. Основна відмінність, яку слід враховувати при розміщенні пристрою в моделюванні, це можливі обмеження обладнання, що поставляються з пристроями, з точки зору кількості доступних портів, варіанти зміни мережевих інтерфейсів, кількість слотів для розширення тощо. У цьому інструменті також доступний широкий список комутаторів, сервери, ПК та ноутбуку.

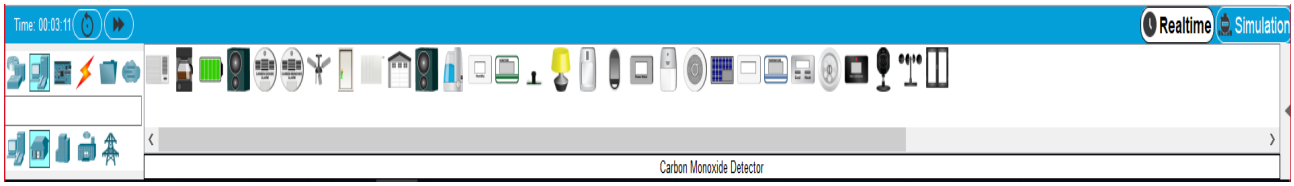


Рисунок 2.3 – Пристрої розумного дому в пакеті інструментів Cisco Packet Tracer

У випадках, таких як організація при «розумного будинку», також використовується моделювання підключення до Інтернету. Це з'єднання має на меті імітувати нормальне підключення до Інтернет-провайдера (ISP), надаючи можливість власнику будинку віддалено підключитися до власної домашньої мережі із зовнішньої мережі, наприклад, корпоративного офісу чи стільникової мережі.

Крім того, для створення моделей розумних будинків впроваджено 3G-мережі. Додатковий компонент, такий як стільникові вежі та сервери, були необхідні для забезпечення функціонування мережі. Використання стільникової мережі дало більше гнучкості під час підключення пристроїв IoT до мережі. Оскільки налаштування є дуже простим і без обмежень, такий діапазон та кількість підключених пристроїв такий же, яким би був у реальних WLAN.

Залежно від налаштувань, використовують різні типи з'єднань пристроїв у мережі такі як: мідні прямі кабелі, мідні перехресні кабелі, оптичні кабелі Fast Ethernet [11].

Як видно на малюнку нижче, Cisco Packet Tracer пропонує кілька варіантів кабельних підключень, однак важливою особливістю є варіант автоматичного кабелювання (значок блискавки на малюнку рис. 2.4). Вибираючи його, програма автоматично обере правильний кабель для підключення двох мережевих інтерфейсів.



Рисунок 2.4 – Види кабельних підключень у Cisco Packet Tracer

MQTT - це стандартизований Push протокол для публікації / підписки, який був випущений IBM у 1999 році.

У протоколі MQTT видавник повідомлень, що публікує повідомлення, та користувачі, які підписуються на теми розглядаються як модель публікації/підписки. Підписник підписується на певні теми, пов'язані з ними, і отримує всі повідомлення, які публікуються у цій теми. З іншого боку, клієнти можуть публікувати повідомлення на теми таким чином, щоб дозволяти всім підписникам отримувати доступ до повідомлень цих тем. Для реалізації роботи протоколу MQTT було використано контролер SBC-PT. SBC-PT – контроллер, який використовується для трансляції сигнальних протоколів та їх діалектів а

також для аналізу якості меді каналів, по яким здійснюється маршрутизація трафіку [12].

2.3 Розробка програмного забезпечення брокера MQTT

Брокер повідомлень є ключовою часиною MQTT-системи. Він забезпечує з'єднання між застосунками або фізичними пристроями та корпоративними системами. Брокери відповідають за підписки, створення сесій та керування ними, пропущені повідомлення та за загальну безпеку системи, включаючи автентифікацію та авторизацію.

Найбільш поширеними брокерами є наступні:

- Mosquitto
- Flepsi
- JoramMQ
- HiveMQ
- VerneMQ

Mosquitto надає бібліотеку на мові програмування C, що дозволяє реалізовувати MQTT-клієнти, а також доволі популярні консольні клієнти: mosquitto_pub та mosquitto_sub.

Flepsi – це безкоштовний публічний MQTT-брокер, що працює у хмарі та підтримує протоколи версій 3.1, 3.1.1, 5.0. Ця система реалізовує архітектуру для обробки великих об'ємів повідомлень, ізольовані простори імен, підтримку вебсокетів та SSL. Крім цього, вона надає можливість її використання як за комерційним договором, так і безкоштовно

JoramMQ – це програмне забезпечення від компанії Scalagent, яке, поміж інших функцій, реалізовує брокера повідомлень, що повністю підтримує протоколи MQTT 3.1, JMS 2.0 та AMQP 1.0. Іншими словами, цей брокер дозволяє одночасно працювати з багатьма протоколами та стандартами. MQTT може працювати з використанням різного транспорту, зокрема TCP/IP, TLS (SSL), веб-сокетів та захищених веб-сокетів.

HiveMQ – це MQTT-брокер, що був спроектований та побудований з нуля з орієнтацією на те, щоб забезпечувати максимальну масштабованість на безпеку, що задовольняє корпоративні стандарти. В ньому одразу наявна підтримка вебсокетів та SDK з відкритим сирцевим кодом, що дозволяє значно розширити функціональність цієї системи та інтегрувати з багатьма іншими компонентами. При цьому, для тестування цієї системи можна використати публічно доступний сервер.

VerneMQ – це розподілений брокер повідомлень, що є готовим до використання в корпоративних цілях та може працювати з високими навантаженнями. Він може маштабуватись як горизонтально, так і вертикально, щоб підтримувати велику кількість видавників та отримувачі повідомлень, які працюють одночасно. При цьому, дана система працює з мінімальними затримками та забезпечує стійкість до помилок. Доповнення до VerneMQ можуть бути розроблені багатьох мовах, наприклад, Erlang, Elixir, Lua та багатьох інших, що підтримують роботу з HTTP веб-хуками. VerneMQ використовує сучасні протоколи для розповсюдження повідомлень, а також систему LevelDB для реплікації стану у кластері. На додаток до зазначеної інформації, VerneMQ є відкритим програмним забезпеченням під ліцензією Apache 2 [13].

MQTT дозволяє розробнику гнучко контролювати надійність обміну повідомленнями, для цього в протоколі передбачено три рівня якості обслуговування QoS (Quality of Service):

QoS 0. Максимум одноразова доставка повідомлень. Приймач повідомлення не підтверджує їх отримання, відправник, відповідно, передає повідомлення лише раз, не роблячи спроб по їх повторної передачі. Це - метод «відправив і забув»

QoS 1. Мінімум одноразова доставка повідомлень. Гарантується, що приймач отримає повідомлення хоча б один раз. При цьому передплатник може отримати одне й те саме повідомлення кілька разів. Відправник робитиме

повторні спроби відправки до тих пір, поки не отримає підтвердження в успішну доставку повідомлення

QoS 2. Одноразова доставка повідомлень. Гарантується, що приймач отримає повідомлення один і тільки один раз. Найповільніша процедура доставки повідомлень, але при цьому і найнадійніша. Для доставки повідомлення використовується механізм чотирьохетапну рукостискання.

Все ж таки, із розвитком промисловості збільшилася кількість приладів, які потрібно контролювати та отримувати від них різного виду дані. Щоб вирішити проблеми взаємодії великої кількості приладів та проблем об'єднання їх в одну мережу, на допомогу ZigBee приходить протокол MQTT [14].

Обмін повідомлень в даному протоколі здійснюється між клієнтом (client), який може бути видавцем або підписником (publisher/subscriber) повідомлень та брокером (broker) повідомлень. Видавець відправляє дані на MQTT брокер, попередньо вказавши тему повідомлення. Підписники можуть отримувати різні дані від безлічі видавців залежно від підписки на відповідні теми.

Пристрої MQTT використовують певні типи повідомлень для взаємодії з брокером:

- Connect – з'єднання з брокером
- Disconnect – розірвання з'єднання з брокером
- Publish – опублікувати дані в тему на брокера
- Subscribe – підписатись на тему брокера
- Unsubscribe– відписатись від теми брокера

Протокол MQTT оточує дві теми: клієнта та брокера. Брокер MQTT - це сервер, а клієнтами є підключені пристрої. Коли пристрій або клієнт хоче надіслати дані серверу - або брокеру, це називається публікацією. Коли операція повернена, вона називається підпискою [14].

Якщо з'єднання від клієнта, що підписався, до брокера розірвано, то брокер буде буферувати повідомлення та виштовхувати їх перед передплатником, коли він знову в мережі. Якщо з'єднання від клієнта-видавця до брокера відключено

без повідомлення, то брокер може закрити з'єднання та надіслати абонентам кешоване повідомлення з інструкціями видавця [15].

Сеанс MQTT розділений на чотири етапи: з'єднання, аутентифікація, зв'язок та припинення. Клієнт починає із створення протоколу управління передачею / протоколу Інтернету (TCP / IP) до брокера, використовуючи стандартний порт або спеціальний порт, визначений операторами брокера. Під час створення з'єднання важливо визнати, що сервер може продовжувати старий сеанс, якщо він надає повторно використану ідентифікацію клієнта.

Стандартні порти 1883 для нешифрованого зв'язку та 8883 для шифрованого зв'язку - за допомогою шару захищених сокетів (SSL) / безпеки транспортного шару (TLS). Під час передачі SSL / TLS клієнт перевіряє сертифікат сервера та автентифікує сервер. Клієнт також може надати клієнтові сертифікат клієнта під час рукостискання. Брокер може використовувати це для автентифікації клієнта. Хоча конкретно не є частиною специфікації MQTT, брокери стали звичними підтримувати автентифікацію клієнтів за допомогою SSL / TLS-сертифікатів клієнта [15].

Оскільки протокол MQTT має на меті бути протоколом для обмежених ресурсів та IoT пристроїв, SSL / TLS не завжди може бути варіантом, а в деяких випадках може бути і не бажаним. У таких випадках автентифікація представляється як ім'я користувача та пароль із чітким текстом, які клієнт надсилає серверу - це як частина послідовності пакетів CONNECT / CONNACK. Крім того, деякі брокери, особливо відкриті брокери, опубліковані в Інтернеті, прийматимуть анонімних клієнтів. У таких випадках ім’я користувача та пароль просто залишаються порожніми.

MQTT називається полегшеним протоколом, оскільки всі його повідомлення мають невеликий слід коду. Кожне повідомлення складається з фіксованого заголовка - 2 байти - необов'язковий заголовок змінної, корисне навантаження повідомлення, яке обмежено 256 мегабайт (МБ) інформації та рівень якості обслуговування (QoS).

Під час фази спілкування клієнт може виконувати операції публікації, підписки, скасування підписки та пінг. Операція публікації надсилає двійковий блок даних - вмісту - до теми, визначеної видавцем.

MQTT підтримує бінарні великі об'єкти повідомлення (BLOB) розміром до 256 Мб. Формат вмісту буде залежним від програми. Підписки на теми проводяться за допомогою пари пакетів SUBSCRIBE / SUBACK, а скасування підписки аналогічно виконується за допомогою пакетної пари UNSUBSCRIBE / UNSUBACK [16].

Рядки теми утворюють природне дерево теми з використанням спеціального символу розмежувача, косої лінії (/). Клієнт може підписатися на цілі гілки в тематичному дереві та скасувати підписку на них із застосуванням спеціальних символів. Є два символи, що містять символи: однорівневий символ підстановки, символ плюс (+); і багаторівневий персонаж із символами wild-card, хеш-символом (#). Спеціальний символ теми, символ долара ($), виключає тему з будь-яких кореневих підписок на шаблонах. Зазвичай $ використовується для транспортування специфічних для сервера або системних повідомлень.

Оскільки протокол MQTT не був розроблений з урахуванням безпеки, протокол традиційно використовується в захищених бек-енд мережах для цілей, що стосуються додатків. Тема структури MQTT може легко сформувати величезне дерево, і немає чіткого способу поділити дерево на менші логічні домени, які можна об'єднати. Це ускладнює створення глобально масштабованої мережі MQTT, оскільки у міру збільшення розміру тематичного дерева складність збільшується.

Ще один негативний аспект MQTT - це його відсутність сумісності. Оскільки корисні навантаження повідомлень є бінарними, без інформації про те, як вони кодуються, можуть виникнути проблеми - особливо у відкритих архітектурах, де різні програми різних виробників повинні безперебійно працювати один з одним.

Як уже згадувалося раніше, MQTT має мінімальні функції автентифікації, вбудовані в протокол. Імена користувачів та паролі надсилаються у чіткому

тексті, і будь-яка форма безпечного використання MQTT повинна використовувати SSL / TLS, що, на жаль, не є легким протоколом.

Автентифікація клієнтів за допомогою клієнтських сертифікатів - це не простий процес, і в MQTT немає способу контролювати того, хто є власником теми та хто може публікувати інформацію про неї, за винятком використання власних, позадіапазонних засобів. Це дозволяє легко вводити шкідливі повідомлення в мережу, навмисно чи помилково [17].

Крім того, немає можливості, щоб приймач повідомлень знав, хто надіслав оригінальне повідомлення, якщо ця інформація не міститься у фактичному повідомленні. Функції безпеки, які доцільно реалізовувати поверх MQTT, збільшують кодовий слід і ускладнюють реалізацію.

ВИСНОВОК ДО РОЗДІЛУ 2

В даному розділі було розглянуто та проаналізовано сучасні світові стандарти опису та протоколи Інтернету речей, зокрема, протоколи MQTT та HTTP. Були проаналізовані типові архітектури рішень на базі протоколу MQTT. Також було зроблено порівняння існуючих MQTT-брікерів. Розглянуто аналоги програмних продуктів. Більше того, були проаналізовані переваги використання протоколу MQTT в контексті обраної предметної області мережевої взаємодії Інтернету речей.

РОЗДІЛ 3
ОЦІНКА ПРАЦЕЗДАТНОСТІ ТА ЕФЕКТИВНОСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПО ВИВЧЕННІ ПРОТОКОЛУ MQTT ДЛЯ ІНТЕРНЕТУ РЕЧЕЙ

Cisco Packet Tracer - це потужна програма моделювання мереж, яка дозволяє системним адміністраторам експериментувати з поведінкою мережі і оцінювати можливі сценарії розвитку подій. Цей інструмент доповнює фізичне обладнання, дозволяючи створювати мережі з практично необмеженою кількістю пристроїв, і допомагає отримати практичні навички конфігурації, пошуку та усунення проблем і виявлення пристроїв.

3.1. Імітаційне моделювання IoT-системи за допомогою Cisco Packet Tracer

Під час виконання атестаційної бакалаврської роботи було спроектовано інформаційно систему по вивченню протоколу MQTT для інтернету речей. Для створення моделі, було вирішено використовувати Cisco Packet Tracer, через широкий функціонал, який чудово підходить для проектування та реалізації моделі розумного будинку.

На рисунку 3.1 зображена система розумного будинку, яка включає в себе набір розумних пристроїв, що поєднані між собою за допомогою відповідних конфігурацій. Керування розумними пристроями та датчиками відбувається завдяки використанню мікрокомп’ютерів MCU-PT та SBC-PT.

Для моделювання приладів IoT можна широко використовувати багатомодульні пристрої (MCU-PT) або одноплатний комп’ютер (SBC-PT). Інтелект IoT серверу, який програмований через браузер, доступний лише для розумних пристроїв та дозволяє використовувати тільки простий набір правил таких як IF-THEN-ELSE. Натомість програмування мікроконтролера дає повну видимість параметрів датчиків, функції керування, реєстрацію даних а також дозволяє використовувати більш широкий спектр пристроїв IoT.

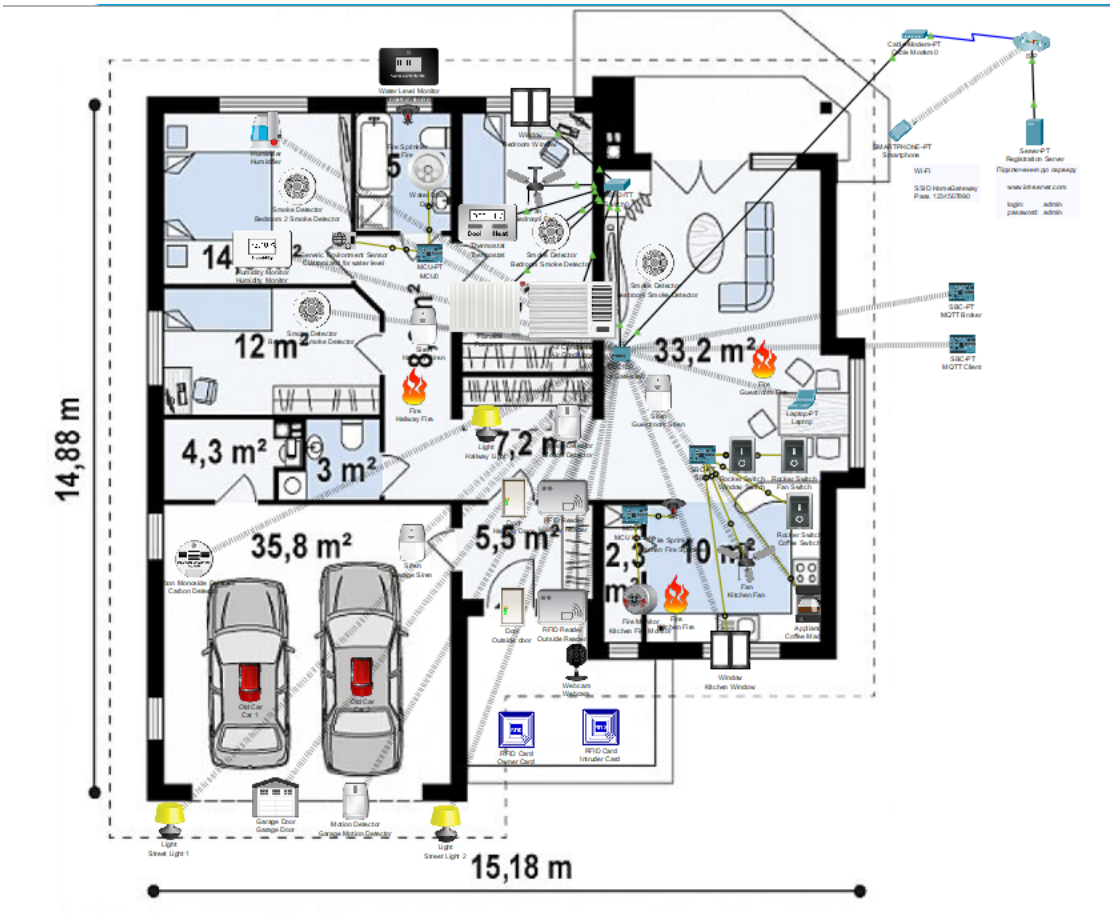


Рисунок 3.1 – Спроектована система розумного будинку

Як видно на рисунку 3.1, усі пристрої IoT, смартфон та сервер резервного копіювання IoT підключені до домашньої WLAN. Мережа імітує звичайну домашню бездротову мережу, де кожен пристрій підключений до бездротового маршрутизатора. Інтернет-параметри маршрутизатора були встановлені за замовчуванням DHCP ISP-значення, однак внутрішній DHCP WLAN був відключений через графічний інтерфейс користувача (GUI), оскільки локальний сервер, підключений до WLAN, функціонував як DHCP [18]. Лише інші параметри маршрутизатора WLAN були домашнім бездротовим SSID та паролем. Усі бездротові пристрої, необхідні для використання однакових налаштувань SSID, пароля та DHCP, за винятком локального сервера, який використовував статичний IP 10 класу.

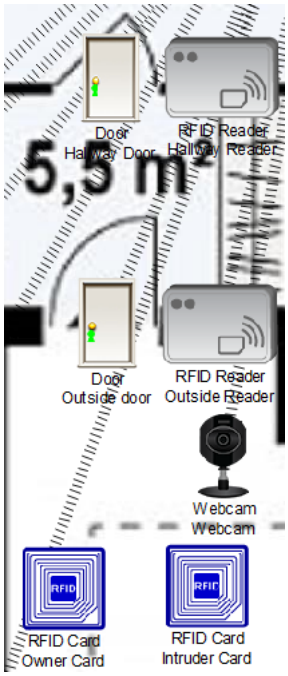


Рисунок 3.2 – Головний вхід у будинок

На рисунку 3.2 зображено головних вхід у будинок, де розташовано дві пари дверей, зчитувачі RFID та веб камера. Система розроблена так, щоб розрізняти вхід власника будинку та вхід зловмисника. Коли власник заходить у будинок система працює за стандартами інструкціями. Коли зловмисник спробує відкрити двері, зчитувач отримає сигнал про несанкціонований вхід, та розпізнає браковану RFID картку, після чого надсилає сигнал до центрального контроллера та на веб камеру, яка відразу починає працювати. Таким чином, розпізнання зловмисника буде легшим.

RFID – це скорочення від Radio Frequency Identification, що перекладається як радіочастотна ідентифікація. Повна назва – це ключ до розуміння того, як ця технологія працює. RFID використовує радіохвилі для відправки інформації на дуже маленькі і прості пристрої. У той час як статті і теорії про системи, подібних RFID, були вперше опубліковані в 1940-х роках, перші реальні практичні застосування і пристрої з функцією RFID з'явилися в середині 1970-х років. У певному сенсі, RFID є більш простою версією технології NFC (Near Field Communication), яку багато смартфони використовують для цифрових платіжних систем, таких як SamsungPay і AndroidPay.

Є два різних типи RFID-пристроїв або міток. Найбільш поширеним чіпом, використовуваним в кредитних і дебетових картах, є «Пасивний RFID». Тег, що використовує цю технологію, не має внутрішнього джерела живлення. Замість цього він отримує живлення, коли стикається з іншим пристроєм, таким як зчитувач RFID-чіпів. Зчитувач відправляє радіохвилі на пасивну мітку RFID, активуючи її, а також зчитує інформацію з мітки у вигляді ідентифікаційного номера. Інший тип мітки RFID використовує технологію «Активний RFID» [19]. Ці мітки мають маленьку вбудовану батарею, яка служить їх власним джерелом живлення. Тому вони можуть відправляти свої власні радіохвилі для передачі будь-якої інформації, закодованої в тезі, відповідного зчитувача. Знову ж таки, це дуже схоже на те, як працює NFC.

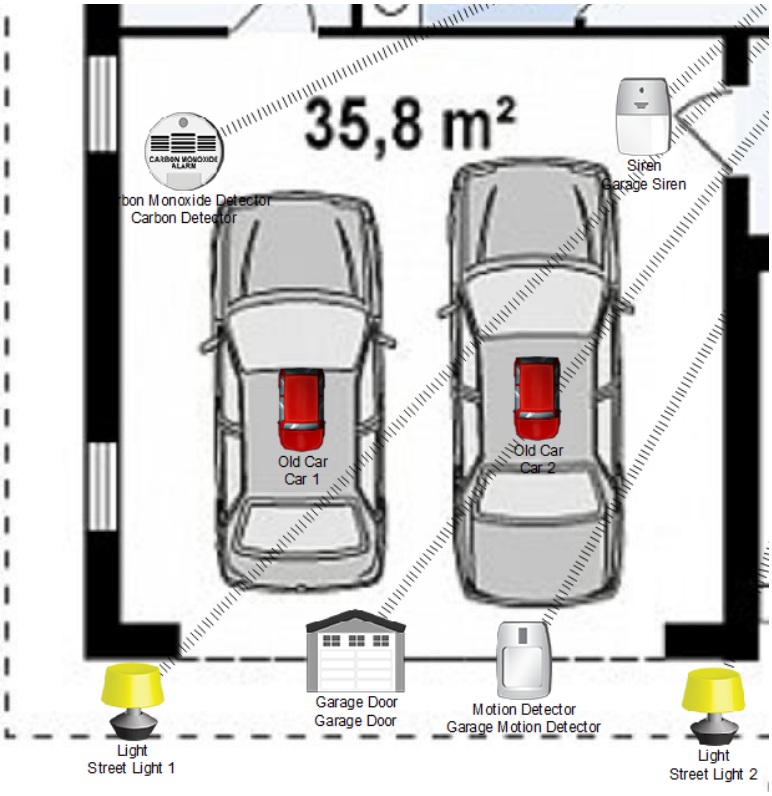


Рисунок 3.3 – Гараж

Важливим також є, забезпечення безпеки у гаражному відділі від несанкціонованого доступу. Датчик вуглекислого газу контролює кількість викидів та в разі перевищенні норми автоматично спрацьовують гаражні ворота та вмикається сигнал тривоги.

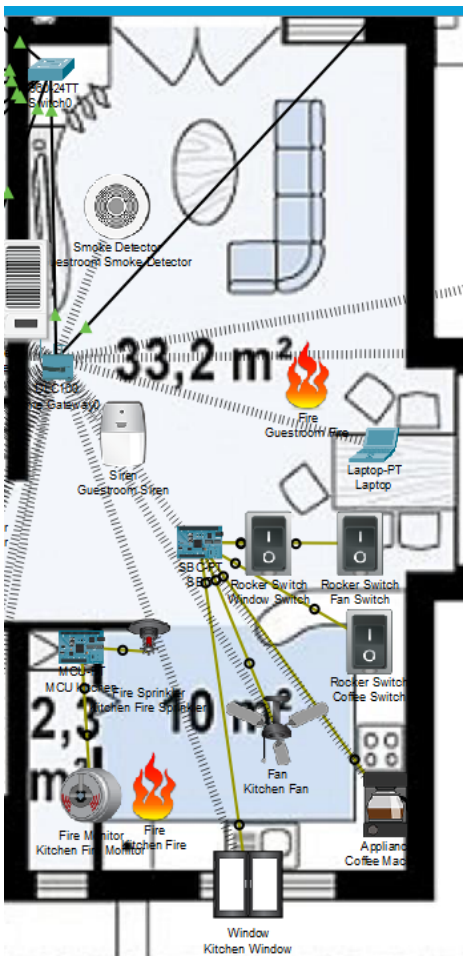


Рисунок 3.4 – Кухня та вітальня

Одна з найнасиченіших розумними пристроями частина розумного будинку. У кухні можна побачити наявність, розумного вікна, кавової машини, вентилятор, датчик вогню, спрікнлер вогню (пристрій гасіння вогню)та контроллер MCU-PT.

Завдяки, всім вище описаним датчикам можна керувати з легкістю кухнею. Не менш важливим фактором є гарантія безпеки. В разі появи вогню, спрацьовує датчик вогню, який надсилає сигнал контроллеру, після чого спрацьовує пристрій для гасіння пожежі. Вентилятор та кавову машину можна ввімкнути віддалено за допомогою безпровідного з’єднання.

У вітальні, яка як ми бачимо не відділена від кухні присутня сигналізація, HomeGateway (головний маршрутизатор), датчик диму та ноутбук.

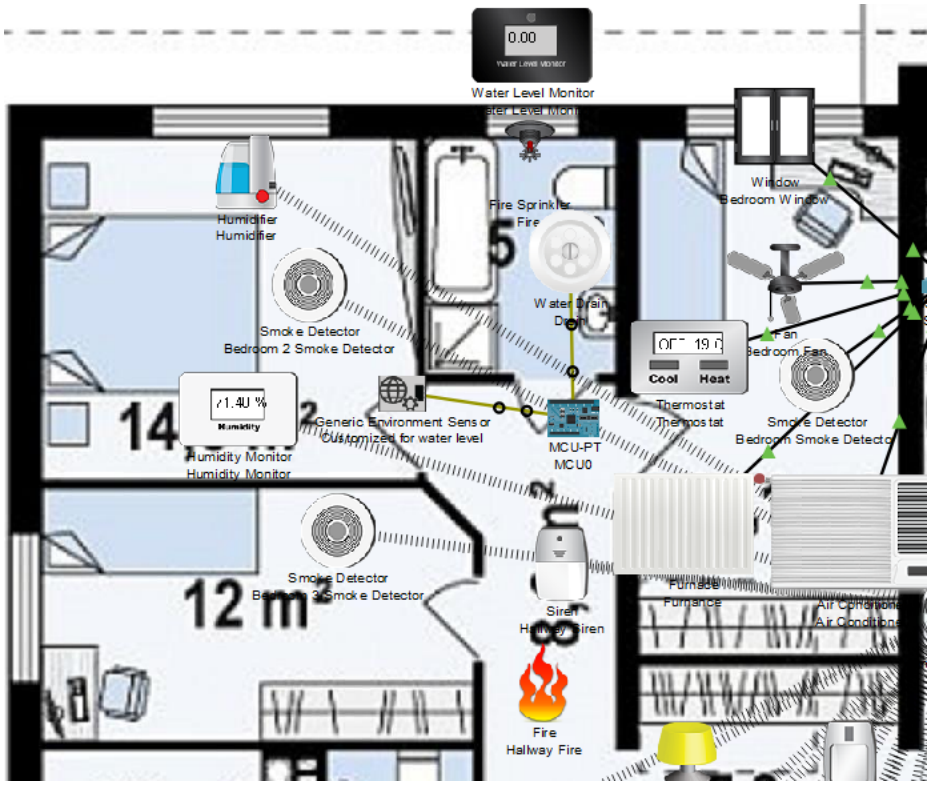


Рисунок. 3.5 – Схема головної житлової площі

На рисунку 3.5 зображено головну житлову площу розумного будинку, яка включає в себе: три спальні кімнати, санітарного вузла та коридор.

3.2. Дослідження працездатності та ефективності інформаційної системи по вивченню протоколу MQTT для Інтернету речей

Для того, щоб запровадити створену інформаційну систему по вивченні протоколу MQTT для Інтернету речей у дії, необхідно ретельно провести ряд тестувань, для уникнення проблем в подальшій роботі.

Для керування розумним будинком у створеній системі використовується телефон а також можливість контролю системи використовуючи ноутбук.

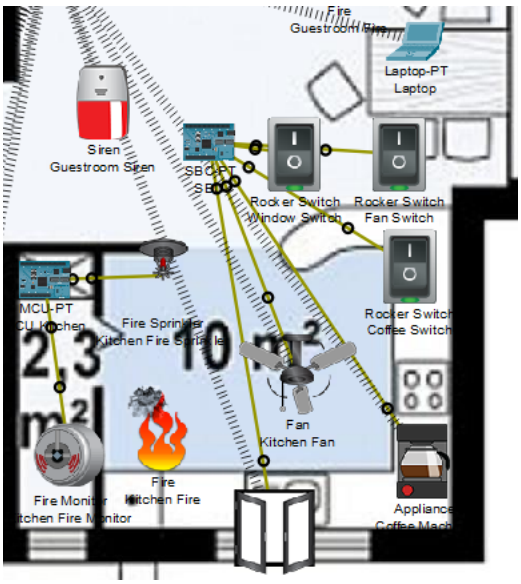


Рисунок 3.6 – Робота розумних пристроїв з імітацією пожежі.

На рисунку 3.6 зображено розумні пристрої під час їхньої дії. Для того щоб перевірити справність пристроїв, необхідно їх запустити. Запуск розумних пристроїв здійснюється за допомогою смартфону або ж ноутбуку (який зображено на рис 3.6). За допомогою смартфону або ж ноутбуку можна вмикати, налаштовувати, вимикати, контролювати систему розумного дому.

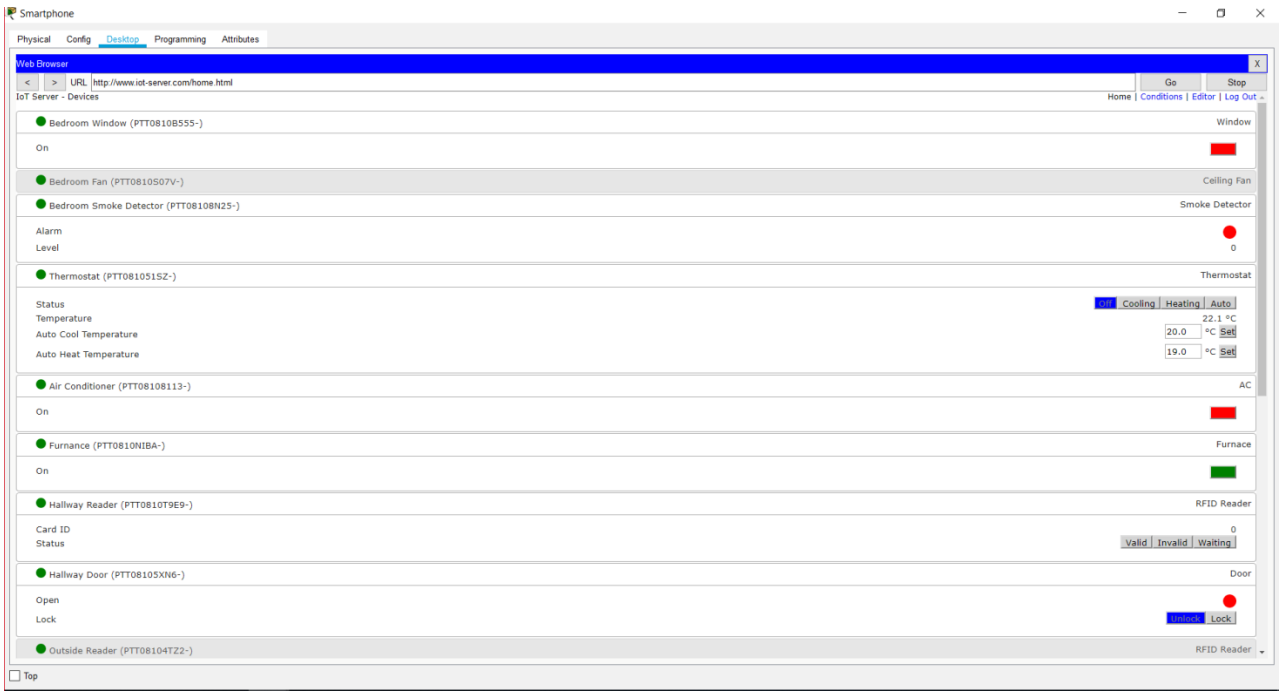


Рисунок 3.7 – Керування розумними пристроями за допомогою смартфону

На рисунку 3.7 продемонстровано інтерфейс смартфону для управління розумними пристроями та датчиками.

Для того, щоб здійснювати керування розумним будинком використовуючи смартфон, необхідно авторизуватись на сервері будинку за адресою: www.iot-server.com та пройти авторизацію ввівши логін та пароль користувача. Після успішної авторизації, надається можливість контролю над розумним будинком.

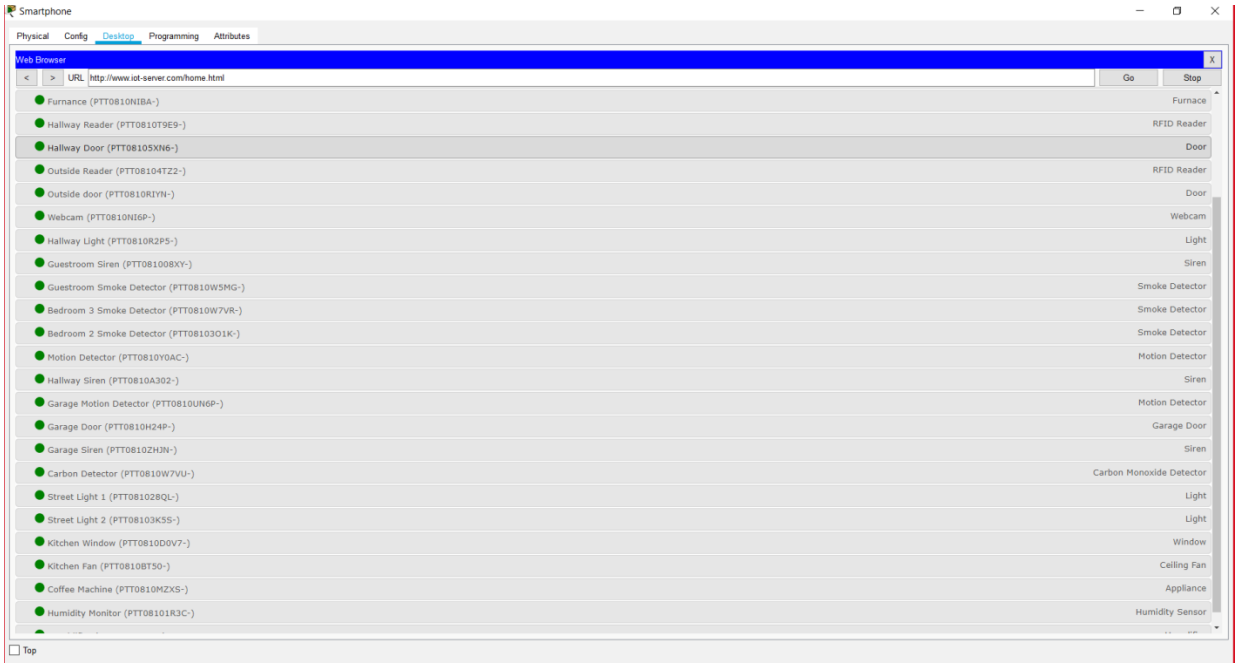


Рисунок 3.8 – Список розумних пристроїв та датчиків для керування будинком

3.3. Інструкція користувачу інформаційної системи по вивченню протоколу MQTT для Інтернету речей

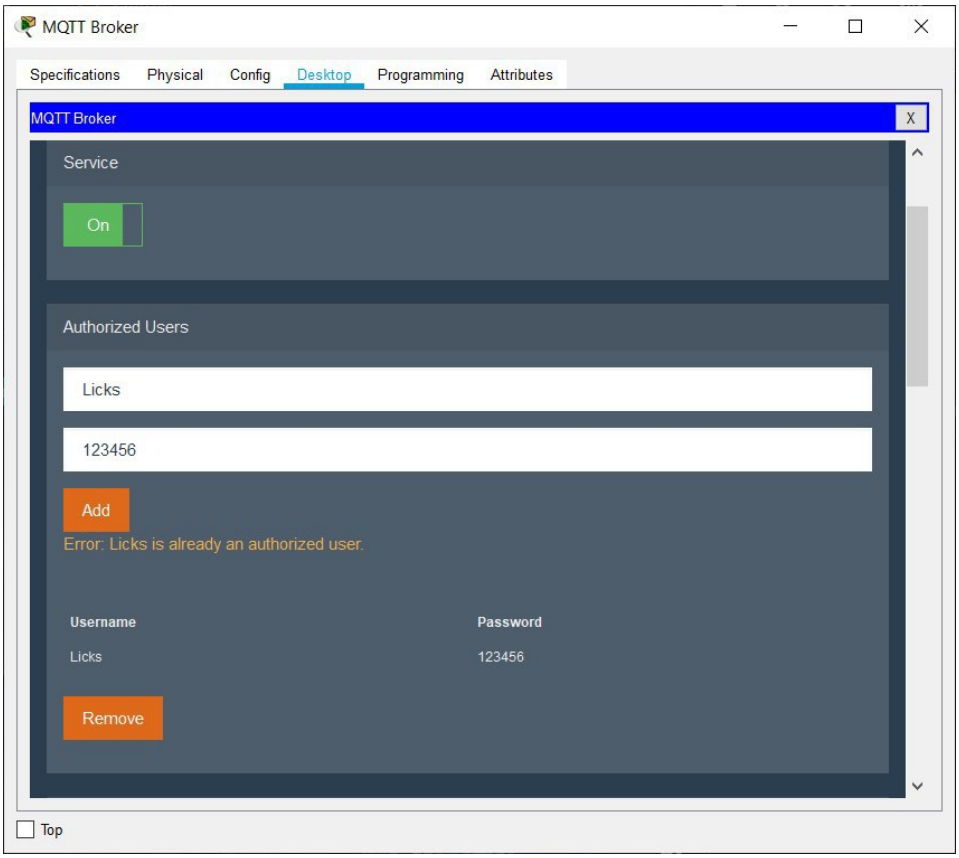
Для запуску додатку MQTT брокера потрібно виконати наступні дії:

1. Натиснути на SBC контроллер, який має назву MQTT Broker
2. Перейти у робоче меню
3. Натиснути на MQTT Broker

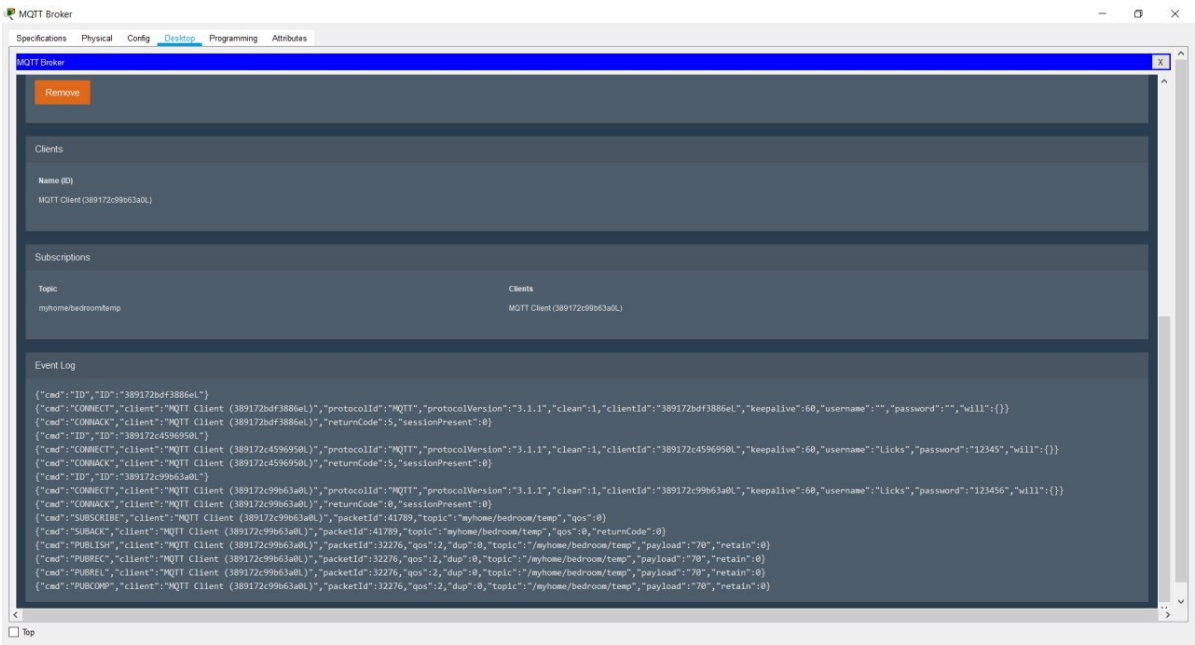
Для того, щоб додати користувачі необхідно:

1. Ввести ім’я користувача

2. Ввести пароль
3. Натиснути кнопку «додати»



Рисунку. 3.9 – Налаштування MQTT Broker



Рисунку 3.10 – Налаштування MQTT Broker

Алгоритм роботи MQTT брокеру було розроблено використовуючи мову програмування Python.

```
def disable_service():
    global authorized_users, clients, subscriptions, server_enabled
    authorized_users = {}
    setDeviceProperty(getName(), "MQTT_BROKER_AUTHORIZED_USERS", str(authorized_users))
    clients = {}
    setDeviceProperty(getName(), "MQTT_BROKER_CLIENTS", str(clients))
    subscriptions = {}
    unconverted_subscriptions = {}
    setDeviceProperty(getName(), "MQTT_BROKER_SUBSCRIPTIONS", str(unconverted_subscriptions))
    server_enabled = False
    setDeviceProperty(getName(), "MQTT_BROKER_ENABLED", str(server_enabled))
    return True
def add_user(username, password):
    global authorized_users
    if username == "":
        return False
    elif username == "" and password != "":
        return False
    elif username not in authorized_users:
        authorized_users[username] = password
        setDeviceProperty(getName(), "MQTT_BROKER_AUTHORIZED_USERS", str(authorized_users))
        return True
    else:
```

```
            return False
    def remove_user(username):
        global authorized_users
        if username == "":
            return False
        elif username in authorized_users:
            authorized_users.pop(username, None)
            setDeviceProperty(getName(),
"MQTT_BROKER_AUTHORIZED_USERS", str(authorized_users))
            return True
        else:
            return False
```

У коді, який описано вище, відбувається налаштування процесу роботи MQTT брокера а саме: авторизація користувачів та перевірка автентифікації (здійснення перевірки імені користувача та його паролю). Детальний код алгоритму описано у додатку А.

Для налаштування MQTT клієнта засобами Python та Cisco Packet Tracer було створено написано наступний алгоритм:

```
def subscribe(topic):
    if topic:
        if topic[0] == "$":
            if on_gui_update_callback is not None:
                on_gui_update_callback("subscribe_fail", {"topic": topic})
            if on_subscribe_callback is not None:
                on_subscribe_callback("Error", "Topic cannot start with $.", {})
        elif "#" in topic and topic[-1] is not "#":
            if on_gui_update_callback is not None:
                on_gui_update_callback("subscribe_fail", {"topic": topic})

            if on_subscribe_callback is not None:
                on_subscribe_callback("Error", "Multi-level wildcard must be at the end of the
topic.", {})
        elif topic in subscriptions:
            if on_gui_update_callback is not None:
                on_gui_update_callback("subscribe_fail", {"topic": topic})

            if on_subscribe_callback is not None:
                on_subscribe_callback("Error", "Already subscribed to " + topic + ".", {})
        else:
            out_packet = {}
            out_packet["cmd"] = "SUBSCRIBE"
            out_packet["client"] = client.ID
```

```
                    out_packet["packetId"] = random.randint(1, 65535)
                    out_packet["topic"] = topic
                    out_packet["qos"] = 0
                    pdu_info = PDUInfo(pdu_color)
                    pdu_info.addOutMessage("The MQTT client sends a SUBSCRIBE message to the MQTT broker.")
                    pdu_info.setOutFormat("MQTT", "MQTT SUBSCRIBE", {})
                    client.sendWithPDUInfo(json.dumps(out_packet), pdu_info)
                    print_event(out_packet)
        else:
            if on_gui_update_callback is not None:
                    on_gui_update_callback("subscribe_fail", {"topic": topic})

            if on_subscribe_callback is not None:
                    on_subscribe_callback("Error", "Topic cannot be empty.", {})
```

У коді програми, який описано вище, реалізовано алгоритм підписки на теми у MQTT протоколі а також створення тем та опису до них. Детальний код програми можна побачити у додатку В.

Для запуску MQTT client потрібно виконати наступне:

1. Натиснути на SBC контроллер, який має назву MQTT Client
2. Перейти у робоче меню
3. Натиснути на MQTT Client

Для з’єднання з MQTT Broker виконуємо наступні дії:

1. Ввести адресу IP адресу MQTT брокера
2. Ввести ім’я користувача та пароль (якщо авторизований хоча б один користувач)
3. Натиснути «Connect» для з’єднання або «Disconnect» для припинення

Для тяго, щоб підписатись на MQTT topic необхідно:

1. Ввести валідний шлях MQTT topic стрічкою вигляду - myhome/bedroom/temp
2. Натиснути «Subscribe»

Для того щоб відписатись від теми, необхідно обрати її з випадаючого списку та натиснути «Unsubscribe».

Для того щоб опублікувати повідомлення до MQTT topic необхідно:

1. Ввести існуючий MQTT topic у вигляді стрічки типу /myhome/bedroom/temp
2. Ввести повідомлення не перевищуючи 70 символів
3. Ввести дійсний QoS level (0-2) де 0 найнижчий QoS та 2 QoS найвищий відповідно.

ВИСНОВОК ДО РОЗДІЛУ 3

У третьому розділі випускної кваліфікаційної роботи бакалавра було спроектовано імітаційну модель IoT-системи за допомогою Cisco Packet Tracer. Було досліджено працездатність та ефективність інформаційної системи по вивченню протоколу MQTT для Інтернету речей та проведено аналіз роботи розумних пристроїв у системі «розумний будинок». Також було описано інструкцію користувача для роботи з системою.

Практично реалізована робота MQTT протоколу та застосована у проектуванні розумного будинку. Результатом третього розділу є готове спроектоване рішення розумного будинку з використанням протоколу MQTT, яке готове до використання.

ВИСНОВКИ

Під час випускної кваліфікаційної роботи бакалавра було виконано поставлені в роботі завдання та цілі. Зокрема, в роботі були проаналізовані технології Інтернет-речей. Більш детально було розглянута модель захисту банківської комірки і можливість застосування його у банках .

У вступі було розглянуто мету бакалаврської роботи та описано основні теоретичні відомості про інтернет речей. У першому розділі проаналізовано літературні та інші джерела. У другому розділі розглянута архітектура Інтернету-речей, яка складається з наступних рівнів: рівня сенсорів і сенсорних мереж, рівня шлюзів і мереж, сервісного рівня і рівня додатків. Також було розглянуто та проаналізовано сучасні світові стандарти опису та протоку домену Інтернету речей, зокрема, протоколи MQTT та HTTP. Проаналізовані типові архітектури рішень на базі протоколу MQTT. Більше того, були описані переваги використання протоколу MQTT в контексті обраної предметної області мережевої взаємодії Інтернету речей.

У третьому розділі проаналізовані можливості Cisco Packet Tracer щодо моделювання IoT-систем. Спроектовано систему розумного будинку з використанням протоколу MQTT. Розроблено алгоритми роботи MQTT брокера та MQTT клієнта. Результатом випускної кваліфікаційної роботи бакалавра, є створена інформаційна система по вивченню протоколу MQTT для інтернету речей та є готовою для використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Али С. Потоковая модель динамической балансировки очередей в MPLS-сети с поддержкой traffic engineering queues / С. Али, А.В. Симоненко // Проблемы телекоммуникаций. — 2010. — № 1. — с.59–67.

2. Бараш Л. Виртуальные частные сети на базе MPLS / Леонид Бараш //Компьютерное Обозрение. — 2004. — № 17. — с. 14–16.

4. Томас Т. Структура и реализация сетей на основе протокола OSPF.Руководство Cisco / Том Томас. — М. : Вильямс, 2004. — 816 с.

5. І. В. Пулеко, Єфіменко А.А. - Методика моделювання IoT проектів «Розумного» будинку в Cisco Packet Tracer . Режим доступу: https://conf.ztu.edu.ua/wp-content/uploads/2019/06/43-1.pdf

6. Олвейн В. Структура и реализация современной технологии MPLS.РуководствоCisco / Вивек Олвейн. — М. : Вильямс, 2004. — 480 с.

7. Олифер В. Искусство оптимизации трафика / В. Олифер, Н. Олифер// Журнал сетевых решений LAN. — 2001. — № 12. — с. 21–26.

8. Rosen, E. Multiprotocol Label Switching Architecture / E. Rosen, A.Viswanathan, R. Callon. — RFC 3031, 2001. — 61 с

9. Cisco Systems: документация [Электронный ресурс]. Режим доступа:http://www.cisco.com/en/US/docs .

10. Вивек Олвейн. Структура и реализация современной технологииMPLS. Руководство Cisco = Advanced MPLS Design andImplementation. — М.: Вильямс, 2004. — 480 с. — ISBN 1-58705

11. Иванов М. А. Криптографические методы защиты информации вкомпьютерных системах и сетях. — М.: КУДИЦ-ОБРАЗ, 2001. —368 с.

12. Столлингс В. Основы защиты сетей. Приложения и стандарты Network Security Essentials. Applications and Standards. —М.:«Вильямс», 2002. — С. 432. — ISBN 0-13-016093-8

13. Савельев А. Современные протоколы маршрутизации. [Электронный ресурс]. – Режим доступа к ресурсу: http://www.citforum.ru/nets/articles/ sovr_marsh.shtml.

14. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 4-е изд. / В.Г. Олифер , Н.А. Олифер. – СПб.: Питер, 2010. – 944 с.

15. Таненбаум Э. Компьютерные сети. 4-е изд. / Таненбаум Э. – СПб.: Питер, 2007. – 992 с.

16. Олифер В.Г. Протокол межсетевого взаимодействия IP [Электронный ресурс] / В.Г. Олифер, Н.А. Олифер. – Режим доступа к ресурсу: http://cylib.iit.nau. edu.ua/Books/NetWork/ ip/contents.htm.

17. Олифер Н.А. Использование моделирования для оптимизации производительности сети [Электронный ресурс] / В.Г. Олифер, Н.А. Олифер.. – Режим доступа к ресурсу: http://www. citforum.ru/ nets/optimize/locnop_09.shtml.

18. Плешаков В. Основы маршрутизации [Электронный ресурс]. – Режим доступа к ресурсу: http://www.citforum.ru/ nets/ito/2.shtml.

19. Остерлох Х. Маршрутизация в IP-сетях. Принципы, протоколы, настройка: Пер. с англ. / В. Плешаков. – СПб.: ООО «ДиаСофтЮП», 2002. – 512 с.

ДОДАТКИ

ДОДАТОК А

Код для налаштування MQTT брокера

```
from cli import *
fromguiimport *
importjson
fromphysicalimport *
importre
fromsimulationimport *
fromtcpimport *
fromtimeimport *
server = None
server_enabled = False
authorized_users = {}
clients = {}
subscriptions = {}
unconverted_subscriptions = {}
events = []
pdu_color = 0x79237c
on_gui_update_callback = None
setDeviceProperty(getName(), "MQTT_BROKER_ENABLED", str(server_enabled))
setDeviceProperty(getName(), "MQTT_BROKER_AUTHORIZED_USERS", str(authorized_users))
setDeviceProperty(getName(), "MQTT_BROKER_CLIENTS", str(clients))
setDeviceProperty(getName(), "MQTT_BROKER_SUBSCRIPTIONS", str(unconverted_subscriptions))
definit():
        globalserver

        ifserverisNone:
                server = TCPServer()
                server.listen(1883)
                server.onNewClient(on_new_client)
                enable_service()

        add_custom_mqtt_pdus()
```

```
defadd_custom_mqtt_pdus():
    ifnotSimulation.hasCustomPDU("MQTT", "MQTT CONNECT"):
        Simulation.addCustomPDU("MQTT", "MQTT CONNECT", {
            "title": "MQTT CONNECT",
            "units": "Bits",
            "unit_marks": [4],
            "width": 8,
            "fields": [
                {
                    "value": "RESERVED:0",
                    "size": 4
                },
                {
                    "value": "CONTROL PACKET TYPE:1",
                    "size": 4
                },
                {
                    "value": "REMAINING LENGTH (VARIABLE LENGTH)",
                    "size": 8
                }
            ]
        })

    ifnotSimulation.hasCustomPDU("MQTT", "MQTT CONNACK"):
        Simulation.addCustomPDU("MQTT", "MQTT CONNACK", {
            "title": "MQTT CONNACK",
            "units": "Bits",
            "unit_marks": [4],
            "width": 8,
            "fields": [
                {
                    "value": "RESERVED:0",
                    "size": 4
```

```
                },
                {
                        "value": "CONTROL PACKET TYPE:2",
                        "size": 4
                },
                {
                        "value": "REMAINING LENGTH:2",
                        "size": 8
                }
        ]
})

ifnotSimulation.hasCustomPDU("MQTT", "MQTT PUBLISH"):
        Simulation.addCustomPDU("MQTT", "MQTT PUBLISH", {
                "title": "MQTT PUBLISH",
                "units": "Bits",
                "unit_marks": [1, 3, 4],
                "width": 8,
                "fields": [
                        {
                                "value": "RETAIN:0",
                                "size": 1
                        },
                        {
                                "value": "QOS:{qos}",
                                "size": 2
                        },
                        {
                                "value": "DUP:0",
                                "size": 1
                        },
                        {
                                "value": "CONTROL PACKET TYPE:3",
                                "size": 4
```

```
                },
                {
                        "value": "REMAINING LENGTH (VARIABLE LENGTH)",
                        "size": 8
                }
        ]
})

ifnotSimulation.hasCustomPDU("MQTT", "MQTT PUBACK"):
        Simulation.addCustomPDU("MQTT", "MQTT PUBACK", {
                "title": "MQTT PUBACK",
                "units": "Bits",
                "unit_marks": [4],
                "width": 8,
                "fields": [
                        {
                                "value": "RESERVED:0",
                                "size": 4
                        },
                        {
                                "value": "CONTROL PACKET TYPE:4",
                                "size": 4
                        },
                        {
                                "value": "REMAINING LENGTH:2",
                                "size": 8
                        }
                ]
        })

ifnotSimulation.hasCustomPDU("MQTT", "MQTT PUBREC"):
        Simulation.addCustomPDU("MQTT", "MQTT PUBREC", {
                "title": "MQTT PUBREC",
                "units": "Bits",
```

```
                "unit_marks": [4],
                "width": 8,
                "fields": [
                        {
                                "value": "RESERVED:0",
                                "size": 4
                        },
                        {
                                "value": "CONTROL PACKET TYPE:5",
                                "size": 4
                        },
                        {
                                "value": "REMAINING LENGTH:2",
                                "size": 8
                        }
                ]
        })

ifnotSimulation.hasCustomPDU("MQTT", "MQTT PUBREL"):
        Simulation.addCustomPDU("MQTT", "MQTT PUBREL", {
                "title": "MQTT PUBREL",
                "units": "Bits",
                "unit_marks": [4],
                "width": 8,
                "fields": [
                        {
                                "value": "RESERVED:2",
                                "size": 4
                        },
                        {
                                "value": "CONTROL PACKET TYPE:6",
                                "size": 4
                        },
                        {
```

```
                "value": "REMAINING LENGTH:2",
                "size": 8
            }
        ]
    })

if not Simulation.hasCustomPDU("MQTT", "MQTT PUBCOMP"):
    Simulation.addCustomPDU("MQTT", "MQTT PUBCOMP", {
        "title": "MQTT PUBCOMP",
        "units": "Bits",
        "unit_marks": [4],
        "width": 8,
        "fields": [
            {
                "value": "RESERVED:0",
                "size": 4
            },
            {
                "value": "CONTROL PACKET TYPE:7",
                "size": 4
            },
            {
                "value": "REMAINING LENGTH:2",
                "size": 8
            }
        ]
    })

if not Simulation.hasCustomPDU("MQTT", "MQTT SUBSCRIBE"):
    Simulation.addCustomPDU("MQTT", "MQTT SUBSCRIBE", {
        "title": "MQTT SUBSCRIBE",
        "units": "Bits",
        "unit_marks": [4],
        "width": 8,
```

```
                "fields": [
                        {
                                "value": "RESERVED:2",
                                "size": 4
                        },
                        {
                                "value": "CONTROL PACKET TYPE:8",
                                "size": 4
                        },
                        {
                                "value": "REMAINING LENGTH (VARIABLE LENGTH)",
                                "size": 8
                        }
                ]
        })

ifnotSimulation.hasCustomPDU("MQTT", "MQTT SUBACK"):
        Simulation.addCustomPDU("MQTT", "MQTT SUBACK", {
                "title": "MQTT SUBACK",
                "units": "Bits",
                "unit_marks": [4],
                "width": 8,
                "fields": [
                        {
                                "value": "RESERVED:0",
                                "size": 4
                        },
                        {
                                "value": "CONTROL PACKET TYPE:9",
                                "size": 4
                        },
                        {
                                "value": "REMAINING LENGTH (VARIABLE LENGTH)",
                                "size": 8
```

```
                }
            ]
        })

ifnotSimulation.hasCustomPDU("MQTT", "MQTT UNSUBSCRIBE"):
        Simulation.addCustomPDU("MQTT", "MQTT UNSUBSCRIBE", {
            "title": "MQTT UNSUBSCRIBE",
            "units": "Bits",
            "unit_marks": [4],
            "width": 8,
            "fields": [
                {
                    "value": "RESERVED:2",
                    "size": 4
                },
                {
                    "value": "CONTROL PACKET TYPE:10",
                    "size": 4
                },
                {
                    "value": "REMAINING LENGTH (VARIABLE LENGTH)",
                    "size": 8
                }
            ]
        })

ifnotSimulation.hasCustomPDU("MQTT", "MQTT UNSUBACK"):
        Simulation.addCustomPDU("MQTT", "MQTT UNSUBACK", {
            "title": "MQTT UNSUBACK",
            "units": "Bits",
            "unit_marks": [4],
            "width": 8,
            "fields": [
                {
```

```
                    "value": "RESERVED:0",
                    "size": 4
                },
                {
                    "value": "CONTROL PACKET TYPE:11",
                    "size": 4
                },
                {
                    "value": "REMAINING LENGTH:2",
                    "size": 8
                }
            ]
        })

if not Simulation.hasCustomPDU("MQTT", "MQTT PINGREQ"):
    Simulation.addCustomPDU("MQTT", "MQTT PINGREQ", {
        "title": "MQTT PINGREQ",
        "units": "Bits",
        "unit_marks": [4],
        "width": 8,
        "fields": [
            {
                "value": "RESERVED:0",
                "size": 4
            },
            {
                "value": "CONTROL PACKET TYPE:12",
                "size": 4
            },
            {
                "value": "REMAINING LENGTH:0",
                "size": 8
            }
        ]
```

```
        })

ifnotSimulation.hasCustomPDU("MQTT", "MQTT PINGRESP"):
        Simulation.addCustomPDU("MQTT", "MQTT PINGRESP", {
                "title": "MQTT PINGRESP",
                "units": "Bits",
                "unit_marks": [4],
                "width": 8,
                "fields": [
                        {
                                "value": "RESERVED:0",
                                "size": 4
                        },
                        {
                                "value": "CONTROL PACKET TYPE:13",
                                "size": 4
                        },
                        {
                                "value": "REMAINING LENGTH:0",
                                "size": 8
                        }
                ]
        })

ifnotSimulation.hasCustomPDU("MQTT", "MQTT DISCONNECT"):
        Simulation.addCustomPDU("MQTT", "MQTT DISCONNECT", {
                "title": "MQTT DISCONNECT",
                "units": "Bits",
                "unit_marks": [4],
                "width": 8,
                "fields": [
                        {
                                "value": "RESERVED:0",
                                "size": 4
```

```
                },
                {
                        "value": "CONTROL PACKET TYPE:14",
                        "size": 4
                },
                {
                        "value": "REMAINING LENGTH:0",
                        "size": 8
                }
            ]
        })

defenable_service():
        globalserver_enabled

        server_enabled = True
        setDeviceProperty(getName(), "MQTT_BROKER_ENABLED", str(server_enabled))
        returnTrue

defdisable_service():
        globalauthorized_users, clients, subscriptions, server_enabled

        authorized_users = {}
        setDeviceProperty(getName(), "MQTT_BROKER_AUTHORIZED_USERS", str(authorized_users))
        clients = {}
        setDeviceProperty(getName(), "MQTT_BROKER_CLIENTS", str(clients))
        subscriptions = {}
        unconverted_subscriptions = {}
        setDeviceProperty(getName(), "MQTT_BROKER_SUBSCRIPTIONS", str(unconverted_subscriptions))
        server_enabled = False
        setDeviceProperty(getName(), "MQTT_BROKER_ENABLED", str(server_enabled))
        returnTrue

defadd_user(username, password):
```

```
        globalauthorized_users

        ifusername == "":
            returnFalse
        elifusername == "" andpassword != "":
            returnFalse
        elifusernamenotinauthorized_users:
            authorized_users[username] = password
            setDeviceProperty(getName(), "MQTT_BROKER_AUTHORIZED_USERS", str(authorized_users))
            returnTrue
        else:
            returnFalse

defremove_user(username):
        globalauthorized_users

        ifusername == "":
            returnFalse
        elifusernameinauthorized_users:
            authorized_users.pop(username, None)
            setDeviceProperty(getName(), "MQTT_BROKER_AUTHORIZED_USERS", str(authorized_users))
            returnTrue
        else:
            returnFalse

defon_new_client(client):
        importsimulation

        globalclients

        client_id = hex(int(time() * 10000000))[2:]
        clients[client_id] = client
        setDeviceProperty(getName(), "MQTT_BROKER_CLIENTS", str(clients))
        client.onConnectionChange(on_connection_change)
```

```
        client.onReceiveWithPDUInfo(process_mqtt_packet)


        fortopicinsubscriptions:
                ifclient_idinsubscriptions[topic]:
                        subscriptions[topic].remove(client_id)


        fortopicinunconverted_subscriptions:
                ifclient_idinunconverted_subscriptions[topic]:
                        unconverted_subscriptions[topic].remove(client_id)
                        setDeviceProperty(getName(), "MQTT_BROKER_SUBSCRIPTIONS",
str(unconverted_subscriptions))


        update_clients_table()
        update_subscriptions_table()


        out_packet = {}
        out_packet["cmd"] = "ID"
        out_packet["ID"] = client_id
        pdu_info = PDUInfo(pdu_color)
        pdu_info.addOutMessage("The MQTT brokersendsthe MQTT clientitsunique ID.
(CiscoPacketTracerpriorietary)")
        pdu_info.setOutFormat("MQTT", "MQTT CONNECT", {})
        client.sendWithPDUInfo(json.dumps(out_packet), pdu_info)
        print_event(out_packet)


defupdate_authorized_users_table():
        ifon_gui_update_callbackisnotNone:
                on_gui_update_callback("update_authorized_users_table", json.dumps(authorized_users))


defupdate_clients_table():
        ifon_gui_update_callbackisnotNone:
                on_gui_update_callback("update_clients_table", json.dumps([clientforclientinclients]))


defupdate_subscriptions_table():
```

```
        ifon_gui_update_callbackisnotNone:
                on_gui_update_callback("update_subscriptions_table", json.dumps(subscriptions))


defstate():
        state = {}
        state["server_enabled"] = server_enabled
        state["authorized_users"] = authorized_users
        state["clients"] = clients
        state["subscriptions"] = subscriptions
        state["events"] = events
        returnstate


defprint_event(packet):
        globalevents


        events.append(packet
        ifon_gui_update_callbackisnotNone:
                on_gui_update_callback("print_event", json.dumps(packet))
defon_connection_change(type):
        globalclients, subscriptions


        iftype == 1 ortype == 3 ortype == 4:
                forclientinclients:
                        ifnotclients[client].connected():
                                fortopicinsubscriptions:
                                        ifclientinsubscriptions[topic]:
                                                subscriptions[topic].remove(client)


                                fortopicinunconverted_subscriptions:
                                        ifclientinunconverted_subscriptions[topic]:
                                                unconverted_subscriptions[topic].remove(client[:client.rindex("(")
- 1])
                                                setDeviceProperty(getName(),
"MQTT_BROKER_SUBSCRIPTIONS", str(unconverted_subscriptions))
```

```
                clients.pop(client, None)
                temp_clients = dict(clients)

                fortemp_clientintemp_clients:
                    new_client = temp_client[:client.rindex("(") - 1]
                    temp_clients[new_client] = temp_clients.pop(temp_client)

                setDeviceProperty(getName(), "MQTT_BROKER_CLIENTS", str(temp_clients))
                update_clients_table()
                update_subscriptions_table()

defprocess_mqtt_packet(in_packet, pdu_info):
    globalclients, subscriptions

    in_packet = json.loads(in_packet)
    print_event(in_packet)

    ifin_packet["cmd"] == "CONNECT":
        clients[in_packet["client"]] = clients.pop(in_packet["clientId"])
        temp_clients = dict(clients)

        forclientintemp_clients:
            new_client = client[:client.rindex("(") - 1]
            temp_clients[new_client] = temp_clients.pop(client)

        update_clients_table()
        setDeviceProperty(getName(), "MQTT_BROKER_CLIENTS", str(temp_clients))
        out_packet = {}
        out_packet["cmd"] = "CONNACK"
        out_packet["client"] = in_packet["client"]
        out_packet["returnCode"] = 0
        out_packet["sessionPresent"] = 0
        pdu_info.addInMessage("The MQTT brokerreceives a CONNECT messagefromthe MQTT client.")
```

```
                pdu_info.addOutMessage("The MQTT brokersends a CONNACK messagetothe MQTT client.")
                pdu_info.setOutFormat("MQTT", "MQTT CONNACK", {})

                ifnotserver_enabled:
                    out_packet["returnCode"] = 3
                    clients[in_packet["client"]].sendWithPDUInfo(json.dumps(out_packet), pdu_info)
                elifauthorized_users == {} andin_packet["username"] == "" andin_packet["password"] == "":
                    clients[in_packet["client"]].sendWithPDUInfo(json.dumps(out_packet), pdu_info)
                elifin_packet["username"] inauthorized_usersandin_packet["password"] ==
authorized_users[in_packet["username"]]:
                    clients[in_packet["client"]].sendWithPDUInfo(json.dumps(out_packet), pdu_info)
                else:
                    out_packet["returnCode"] = 5
                    clients[in_packet["client"]].sendWithPDUInfo(json.dumps(out_packet), pdu_info)

                print_event(out_packet)
            elifin_packet["cmd"] == "SUBSCRIBE":
                converted_topic = "^" + in_packet["topic"].replace("+", "[^/]+").replace("#", ".+") + "$"

                ifconverted_topicnotinsubscriptions:
                    subscriptions[converted_topic] = []
                    unconverted_subscriptions[in_packet["topic"]] = []
                    setDeviceProperty(getName(), "MQTT_BROKER_SUBSCRIPTIONS",
str(unconverted_subscriptions))

                out_packet = {}
                out_packet["cmd"] = "SUBACK"
                out_packet["client"] = in_packet["client"]
                out_packet["packetId"] = in_packet["packetId"]
                out_packet["topic"] = in_packet["topic"]
                out_packet["qos"] = in_packet["qos"]
                out_packet["returnCode"] = in_packet["qos"]
                pdu_info.addInMessage("The MQTT brokerreceives a SUBSCRIBE messagefromthe MQTT client.")
                pdu_info.addOutMessage("The MQTT brokersends a SUBACK messagetothe MQTT client.")
```

```
                pdu_info.setOutFormat("MQTT", "MQTT SUBACK", {})


                ifin_packet["client"] notinsubscriptions[converted_topic]:
                        subscriptions[converted_topic].append(in_packet["client"])

        unconverted_subscriptions[in_packet["topic"]].append(in_packet["client"][:in_packet["client"].rindex("(") - 1])
                        setDeviceProperty(getName(), "MQTT_BROKER_SUBSCRIPTIONS",
str(unconverted_subscriptions))
                        clients[in_packet["client"]].sendWithPDUInfo(json.dumps(out_packet), pdu_info)
                else:
                        out_packet["returnCode"] = 128
                        clients[in_packet["client"]].sendWithPDUInfo(json.dumps(out_packet), pdu_info)


                print_event(out_packet)
                update_subscriptions_table()
        elifin_packet["cmd"] == "UNSUBSCRIBE":
                converted_topic = "^" + in_packet["topic"].replace("+", "[^/]+").replace("#", ".+") + "$"


                ifconverted_topicinsubscriptions:
                        ifin_packet["client"] insubscriptions[converted_topic]:
                                subscriptions[converted_topic].remove(in_packet["client"])

        unconverted_subscriptions[in_packet["topic"]].remove(in_packet["client"][:in_packet["client"].rindex("(") -
1])
                                setDeviceProperty(getName(), "MQTT_BROKER_SUBSCRIPTIONS",
str(unconverted_subscriptions))


                out_packet = {}
                out_packet["cmd"] = "UNSUBACK"
                out_packet["client"] = in_packet["client"]
                out_packet["packetId"] = in_packet["packetId"]
                out_packet["topic"] = in_packet["topic"]
                out_packet["qos"] = in_packet["qos"]
                pdu_info.addInMessage("The MQTT brokerreceivesan UNSUBSCRIBE messagefromthe MQTT
client.")
                pdu_info.addOutMessage("The MQTT brokersendsan UNSUBACK messagetothe MQTT client.")
```

```
                pdu_info.setOutFormat("MQTT", "MQTT UNSUBACK", {})
                clients[in_packet["client"]].sendWithPDUInfo(json.dumps(out_packet), pdu_info)
                print_event(out_packet)
                update_subscriptions_table()
        elifin_packet["cmd"] == "PUBLISH":
                out_packet = {}
                out_packet["cmd"] = "PUBLISH"
                out_packet["client"] = in_packet["client"]

                if "packetId" inin_packet:
                        out_packet["packetId"] = in_packet["packetId"]

                out_packet["qos"] = in_packet["qos"]
                out_packet["dup"] = in_packet["dup"]
                out_packet["topic"] = in_packet["topic"]
                out_packet["payload"] = in_packet["payload"]
                out_packet["retain"] = in_packet["retain"]
                pdu_info.addInMessage("The MQTT brokerreceives a PUBLISH messagefromthe MQTT client.")

                ifin_packet["qos"] != 2:
                        publish_pdu_info = PDUInfo(pdu_color)
                        publish_pdu_info.addOutMessage("The MQTT brokersends a PUBLISH messagetothe
MQTT client.")
                        publish_pdu_info.setOutFormat("MQTT", "MQTT PUBLISH", {"qos": in_packet["qos"]})

                        forsubscriptioninsubscriptions:
                                ifre.search(subscription, in_packet["topic"]):
                                        forsubscriberinsubscriptions[subscription]:
                                                clients[subscriber].sendWithPDUInfo(json.dumps(out_packet),
publish_pdu_info)
                                                print_event(out_packet)

                        ifin_packet["qos"] == 1:
                                out_packet["cmd"] = "PUBACK"
```

```
                pdu_info.addOutMessage("The MQTT brokersends a PUBACK messagetothe
MQTT client.")
                pdu_info.setOutFormat("MQTT", "MQTT PUBACK", {})
                clients[in_packet["client"]].sendWithPDUInfo(json.dumps(out_packet), pdu_info)
                print_event(out_packet)
        else:
            out_packet["cmd"] = "PUBREC"
            pdu_info.addOutMessage("MQTT brokersends a PUBREC tothe MQTT client.")
            pdu_info.setOutFormat("MQTT", "MQTT PUBREC", {})
            clients[in_packet["client"]].sendWithPDUInfo(json.dumps(out_packet), pdu_info)
            print_event(out_packet)
    elifin_packet["cmd"] == "PUBREL":
        out_packet = {}
        out_packet["cmd"] = "PUBLISH"
        out_packet["client"] = in_packet["client"]
        out_packet["packetId"] = in_packet["packetId"]
        out_packet["qos"] = in_packet["qos"]
        out_packet["dup"] = in_packet["dup"]
        out_packet["topic"] = in_packet["topic"]
        out_packet["payload"] = in_packet["payload"]
        out_packet["retain"] = in_packet["retain"]
        pdu_info.addInMessage("The MQTT brokerreceives a PUBREL messagefromthe MQTT client.")
        publish_pdu_info = PDUInfo(pdu_color)
        publish_pdu_info.addOutMessage("The MQTT brokersends a PUBLISH messagetothe MQTT
client.")
        publish_pdu_info.setOutFormat("MQTT", "MQTT PUBLISH", {"qos": in_packet["qos"]})

        forsubscriptioninsubscriptions:
            ifre.search(subscription, in_packet["topic"]):
                forsubscriberinsubscriptions[subscription]:
                    clients[subscriber].sendWithPDUInfo(json.dumps(out_packet),
publish_pdu_info)
                    print_event(out_packet)
        out_packet["cmd"] = "PUBCOMP"
        pdu_info.addOutMessage("The MQTT brokersends a PUBCOMP messagetothe MQTT client.")
```

```
                pdu_info.setOutFormat("MQTT", "MQTT PUBCOMP", {})
                clients[in_packet["client"]].sendWithPDUInfo(json.dumps(out_packet), pdu_info)
                print_event(out_packet)
        elifin_packet["cmd"] == "PINGREQ":
                out_packet = {}
                out_packet["cmd"] = "PINGRESP"
                out_packet["client"] = in_packet["client"]
                pdu_info.addInMessage("The MQTT brokerreceives a PINGREQ messagefromthe MQTT client.")
                pdu_info.addOutMessage("The MQTT brokersends a PINGRESP messagetothe MQTT client.")
                pdu_info.setOutFormat("MQTT", "MQTT PINGRESP", {})
                clients[in_packet["client"]].sendWithPDUInfo(json.dumps(out_packet), pdu_info)
                print_event(out_packet)
        elifin_packet["cmd"] == "DISCONNECT":
                fortopicinsubscriptions:
                        ifin_packet["client"] insubscriptions[topic]:
                                subscriptions[topic].remove(in_packet["client"])

                fortopicinunconverted_subscriptions:
                        ifin_packet["client"] inunconverted_subscriptions[topic]:

        unconverted_subscriptions[topic].remove(in_packet["client"][:in_packet["client"].rindex("(") - 1])
                                setDeviceProperty(getName(), "MQTT_BROKER_SUBSCRIPTIONS",
str(unconverted_subscriptions))
                pdu_info.addInMessage("The MQTT brokerreceives a DISCONNECT messagefromthe MQTT
client.")
                pdu_info.setAccepted()
                clients.pop(in_packet["client"], None)
                temp_clients = dict(clients)
                forclientintemp_clients:
                        new_client = client[:client.rindex("(") - 1]
                        temp_clients[new_client] = temp_clients.pop(client)
                setDeviceProperty(getName(), "MQTT_BROKER_CLIENTS", str(temp_clients))
                update_clients_table()
                update_subscriptions_table()
defonGUIUpdate(callback):
```

globalon_gui_update_callback

on_gui_update_callback = callback

ДОДАТОК Б

Код для налаштування MQTT клієнта

```
from cli import *
fromguiimport *
importjson
importmqttclient
fromtimeimport *

defguiEvent(type, args):
        data = json.loads(args)

        mqttclient.init()

        iftype == "state":
                GUI.update("state", json.dumps(mqttclient.state()))
        eliftype == "connect":
                mqttclient.connect(data["broker_address"], data["username"], data["password"])
        eliftype == "disconnect":
                mqttclient.disconnect()
        eliftype == "subscribe":
                mqttclient.subscribe(data["topic"])
        eliftype == "unsubscribe":
                mqttclient.unsubscribe(data["topic"])
        eliftype == "publish":
                mqttclient.publish(data["topic"], data["payload"], data["qos"])

defcliEvent(type, args):
        iftype == "invoked" andargs[0] == "mqttclient":
                iflen(args) < 2 orlen(args) > 1 andargs[1] == "-?" orargs[1] == "/?":
                        print_cli_usage()
                        CLI.exit()
                eliflen(args) > 1 andargs[1] != "-?" andargs[1] != "/?":
                        mqttclient.init()
```

```
iflen(args) > 2 andlen(args) < 6 andargs[1] == "connect":
        username = ""
        password = ""

        iflen(args) > 3:
                username = args[3]

                iflen(args) == 5:
                        password = args[4]

        mqttclient.connect(args[2], username, password)
eliflen(args) == 2 andargs[1] == "disconnect":
        mqttclient.disconnect()
eliflen(args) == 3 andargs[1] == "subscribe":
        mqttclient.subscribe(args[2])
eliflen(args) == 3 andargs[1] == "unsubscribe":
        mqttclient.unsubscribe(args[2])
eliflen(args) == 5 andargs[1] == "publish":
        mqttclient.publish(args[2], args[3], args[4])
eliflen(args) == 2 andargs[1] == "display-last-message":
        messages = mqttclient.state()["messages"]

        iflen(messages) > 0:
                printmessages[-1]

        print ""
        CLI.exit()
eliflen(args) == 2 andargs[1] == "display-all-messages":
        messages = mqttclient.state()["messages"]

        formessageinmessages:
                printmessage
```

```
                print ""
                CLI.exit()
            eliflen(args) == 2 andargs[1] == "display-last-event":
                events = mqttclient.state()["events"]

                iflen(events) > 0:
                    printevents[-1]

                print ""
                CLI.exit()
            eliflen(args) == 2 andargs[1] == "display-all-events":
                events = mqttclient.state()["events"]

                foreventinevents:
                    printevent

                print ""
                CLI.exit()
            else:
                print_cli_usage()
                CLI.exit()
        eliftype == "interrupted":
            CLI.exit()

defprint_cli_usage():
        print "MQTT Client"
        print ""
        print "Usage:"
        print "mqttclientconnect<brokeraddress> [username] [password]"
        print "mqttclientdisconnect"
        print "mqttclientsubscribe<topic>"
        print "mqttclientunsubscribe<topic>"
        print "mqttclientpublish<topic><payload><qos>"
        print "mqttclientdisplay-last-message"
```

```
        print "mqttclientdisplay-all-messages"
        print "mqttclientdisplay-last-event"
        print "mqttclientdisplay-all-events"
        print ""


defon_connect(status, msg, packet):
        ifstatus == "Success" orstatus == "Error":
                printstatus + ": " + msg
        elifstatus == "":
                printmsg


        CLI.exit()
defon_disconnect(status, msg, packet):
        ifstatus == "Success" orstatus == "Error":
                printstatus + ": " + msg
        elifstatus == "":
                printmsg
```