Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»
Навчально-науковий Інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи бакалавра

студент Лісунов Ігор Євгенійович
(ПІБ)

академічної групи 123-19ск-1
(шифр)

спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)

освітній рівень бакалавр
(назва освітнього рівня)

на тему: «Комп'ютерна система протипожежного захисту розумного будинку з віддаленим керуванням через сервер MQTT»

Виконавець: студент 3 курсу, групи 123-19ск-1 _____ Лісунов І.Є.
(підпис) (прізвище та ініціали)

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинг.	інституційною	
Кваліфікаційної роботи	доц. Сергєєва К.Л.			
Розділів:				
Розробка апаратної частини ком'пютерної системи	доц. Бешта Д.О.			
Проектування корпоративної мережі	ас. Панферова Я.В.			

Рецензент				

Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2022

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
інформаційних технологій та комп'ютерної інженерії
проф. Гнатушенко В.В.

_____
"25" січня 2022 р.

ЗАВДАННЯ
на кваліфікаційну роботу
бакалавра
(назва освітньо-кваліфікаційного рівня)

студенту групи *123-19ск-1* *Лісунов Ігор Євгенійович*
(група) (прізвище, ім'я та по батькові)

Тема дипломної роботи «*Комп'ютерна система протипожежного захисту розумного будинку з віддаленим керуванням через сервер MQTT*»

затверджена наказом ректора НТУ "Дніпровська політехніка"
від « 18 » 06 2022 р. №268-с

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел обгрунтувати необхідність модернізації комп'ютерної системи з детальною розробкою комп'ютерної мережі.	10.05.2022
Технічні вимоги до системи керування	На основі аналізу характеристик «Протипожежна система розумного будинку» сформулювати технічні вимоги до розробки комп'ютерної мережі.	17.05.2022
Спеціальна частина	Розв'язати завдання з розробки комп'ютерної мережі «Протипожежна система розумного будинку» з опрацюванням побудови апаратних засобів та налаштування.	31.05.2022

Завдання видав, кер. роботи _____ Сергєєва К.Л.
(підпис)

Завдання прийняв до виконання _____ Лісунов І.Є.
(підпис)

Дата видачі завдання *«25» січня 2022 р.*

Термін подання дипломної роботи до ДЕК .06. *2022 р.*

РЕФЕРАТ

Пояснювальна записка: 73с., 28 рис., 3 табл., 3додатка, 20 джерел

Об'єкт розробки: комп'ютерна система протипожежного захисту розумного будинку з віддаленим керуванням через сервер MQTT.

Мета: створення комп'ютерної системи для захисту будівель від різних побутових загроз. Забезпечити систему, яка створюється різними новітніми засобами ІТ можливостями, сучасне керування та найкращої системи прийняття рішень.

Система повинна мати можливість змінювати число виконуваних функцій за допомогою перепрограмування системи. Система має забезпечити можливість на технічну модернізацію, також вона повинна збирати інформацію, автоматично оброблювати її, перевіряти на зміни в системі, при відхилені від норми повинна швидко реагувати.

Розробка комп’ютерної мережі виконана відповідно до завдання на кваліфікаційну роботу бакалавра.

Розроблена схема мережі реалізована у вигляді моделі на симуляторі Cisco Packet Tracer і перевірена її робота.

ЗМІСТ

ВСТУП

Незалежно від того, наскільки ми захоплені окремими функціями пристроїв IoT, їх взаємодія у вирішенні конкретної проблеми. Перш за все, системи на основі IoT повинні бути добре захищені та прості в проектуванні, установці, обслуговуванні та експлуатації, особливо для систем безпеки, які поступово виходять далеко за межі оригінальних камер відеоспостереження.

Насправді завдяки Інтернету речей розширюються традиційні обов'язки систем безпеки, використовується спільне керування та контролювання всього комплексу або об'єктів різного призначення. Яскравим представником IoT являється "Розумний будинок" (Smart House) – це сучасна інтелектуальна мережа пристроїв, датчиків і приладів, які працюють для забезпечення прогресивного і зручного проживання людини у власній оселі. Актуальність системи «смарт-будинку» в першу чергу полягає у створенні комфортного житла. З такою технологією в оселі гарантована безпека та затишок, сучасне оформлення та легкість у керуванні нею. Можливий контроль за використанням енергоресурсів і оптимізація їх витрат, як наслідок, зменшення негативного впливу на навколишнє середовище. «Розумна система» дозволяє забути про дрібні повсякденні справи, залишаючи більше часу для приємного спілкування з родиною або друзями.

Тому цільова система безпеки спочатку отримує найширшу можливість обміну важливими даними з іншими пристроями за допомогою єдиного дистанційного керування. Впровадження IoT відбувається не в усьому світі, а в межах міст.

Не один виробник не надає повного рішення такої проблеми, в цьому і полягає складність створення даної системи. Треба правильно підбирати всі компоненти від різних виробників.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

Технології не стоять на місці і вже сьогодні кожному відомо про реальні речі, які раніше могли бути лише вигадкою фантастів і описувалися тільки в книгах і демонструвалися в рамках кінематографа. Раніше до галузі фантастики відносилось те, що воду в чайнику можна нагріти без участі людини в цьому процесі безпосередньо. А сьогодні це цілком реально та дуже поширено. Все це стало можливо та доступно завдяки технології «розумний будинок».

Визначення «розумний будинок» (Smart Home) – це вираз, що прийшов до нас із заходу. Під цією термінологією розуміється таке поняття, як автоматизація систем побуту, створення систем для того, щоб спростити життя людям. Завдяки «розумному будинку» рутинні завдання більше не займають багато часу, не приносять роздратування та не викликають втоми у власників будинку[1].

Це поняття часто включає все, що може істотно спрощувати перебування і проживання в квартирі, будинку і т.д. Доволі часто мається на увазі, що «розумний будинок» – це керування на автоматичному рівні чайниками, кухонними машинами, освітленням, проекторами, телебаченням та іншими мультимедійними пристроями. Але важливо розуміти, що система розумного будинку – це набагато більше, ніж автоматика активації роботи чайника, вона включає керування системами опалення, водопостачання, охоронними системами і відеоспостереженням [1].

При облаштуванні таких будинків проекти передбачають встановлення спеціальних пристроїв та датчиків, які здатні вмикати та вимикати світло, регулювати температуру в кімнаті, стежити за опаленням, охоронними системами та загалом полегшити господарям керування будинком.

Система керування «Розумний будинок» дозволяє за допомогою передових технологій автоматизувати роботу практично всіх інженерних

комунікацій, щоб позбавити людину зайвої роботи. Розумним будинком можна керувати наступними системами[1]:

– повний контроль безпеки, куди належить відеоспостереження, а також датчики руху, які попереджають про несанкціонований доступ на територію будинку;

– система протипожежного захисту;

– автоматичне регулювання електропостачання, забезпечення безперебійної подачі електроенергії, включення резервного живлення за необхідності;

– керування системою освітлення, куди входить також аварійне та чергове;

– створення оптимальної температури в будинку для комфортного перебування в ньому, погодження функціонування систем опалення, кондиціювання та вентиляції;

– оптимізація системи водопостачання, економія витрати води, встановлення систем антизатоплення;

– автоматизація процесу роботи побутових приладів, необхідних для прибирання або приготування їжі, а також мультимедійного та телевізійного обладнання;

– контроль за якістю функціонування та безпекою всіх систем, встановлених у будинку, а також відстеження стану самої будівлі.

Охоронно-пожежна сигналізація в «розумному будинку» – це комплекс обладнання, яке забезпечує безпеку майна від різних непередбачених ситуацій, пов’язаних з протіканнями, займаннями тощо.

1.1 Галузь застосування комп’ютерної системи

Встановлення системи протипожежного захисту дозволяє захистити будівлю від наступних непередбачених ситуацій:

– протікання в системах забезпечення водою та газом;

– короткого замикання в електромережі;

– займання;

– наслідків аварій різних інженерних систем.

Розумні датчики протипожежної системи реагують на появу диму, перебої в роботі електрики тощо. Вони передають сигнал головному центру системи, який в автоматичному режимі вживає комплекс заходів, щоб попередити можливий розвиток нештатної ситуації.

Ключові переваги такої системи:

– при спрацьовуванні датчиків протипожежної системи спрацьовує система, яка блокує електрощит та перекриває газові вентилі;

– протипожежна система може сповіщати не лише людей, які перебувають у будинку, а й відповідні служби. Також на мобільний телефон власника будівлі приходить СМС, що дозволить швидко діяти у нештатній ситуації, керуючи системою зі смартфона.

– всі записи передаються на віддалений сервер, що здійснюється через «хмару».

1.2 Характеристика і структура об’єкта впровадження

Об’єкт впровадження – приватний будинок.

Характеристики будинку[2]:

– приватний будинок являє собою двоповерхову будівлю, в будівлі присутнє гаражне приміщення, загальна площа кімнат та приміщень – становить 174,6 м^2 .;

– стіни, та несучі конструкція виконані з цегли, товщина 400 мм.;

– перекриття між поверхами виконане з пустотних залізобетонних плит;

– внутрішні перестінки виконані також з цегли, товщиною 200 мм.;

– стеля – підвісна, загальна висота від підлоги 270 см., висота міжстелевого проміжку 20 см. (на першому поверсі), 50 см. (на другому поверсі);

– підлога в кімнатах покрита ламінатом, на кухні, у ванній кімнаті та у санвузлах – кахельна плитка;

– вікна – металопластикові пакети з подвійним склом;

– двері виконані з виконані з металу та дерева;

– встановлена автономна система опалювання;

– системи електропостачання, водопостачання та каналізації – централізовані.

Таблиця 1.1 – Специфікація приміщень будинку

Номер приміщення	Назва кімнати/приміщення
1	Кабінет на першому поверхі
2	Гостьова кімната
3	Кухня
4	Ванна кімната
5	Санвузол на першому поверсі
6	Комора на першому поверсі
7	Гараж
8	Коридор на першому поверсі
9	Міжповерховий перехід
10	Спальня
11	Дитяча кімната
12	Гостьова спальня
13	Санвузол на другому поверсі
14	Комора на другому поверсі
15	Кабінет на другому поверсі
16	Спальня для гостей
17	Коридор на другому поверсі
18	Балкон

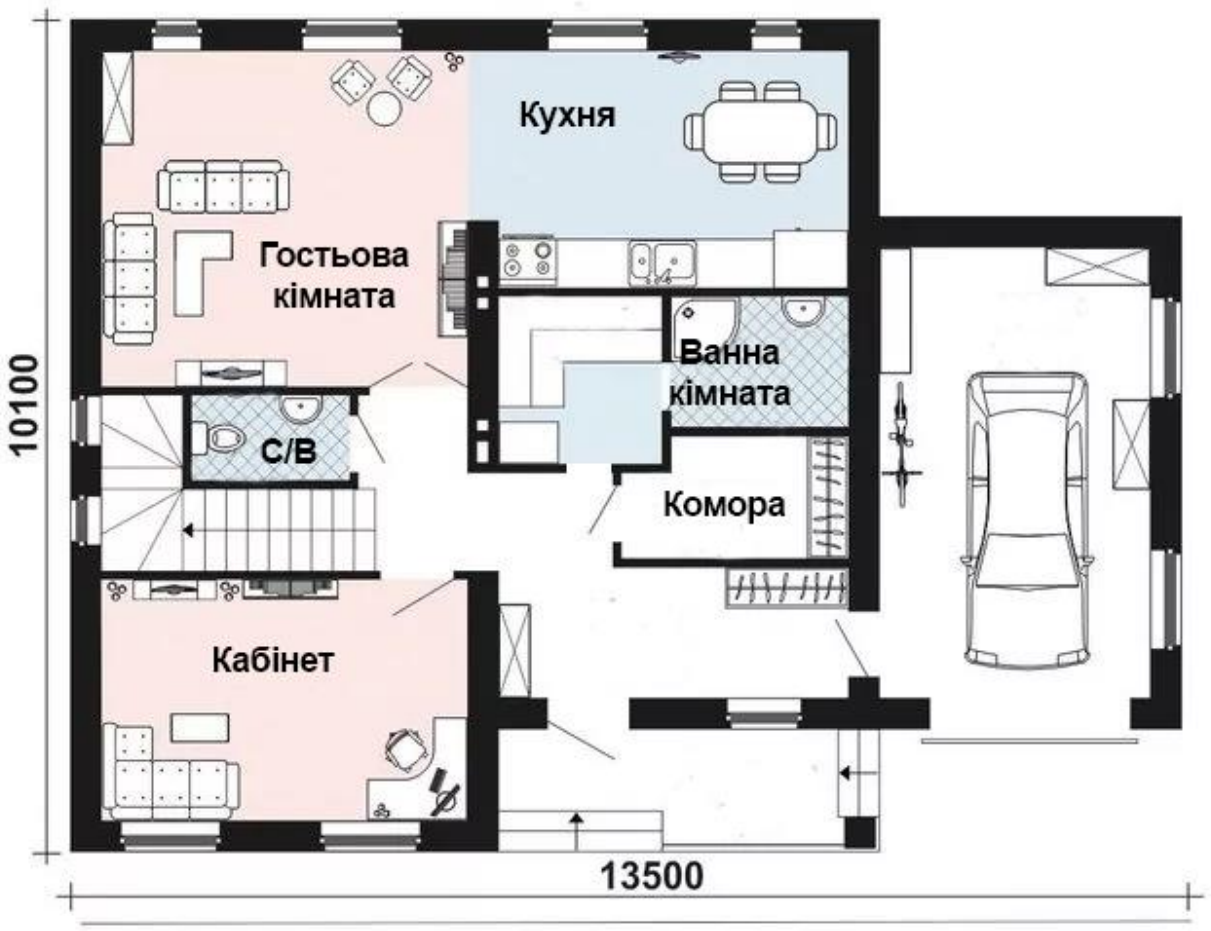

Рисунок 1.1 – Перший поверх будинку

Система розумного будинку включає мережу сенсорів, що контролюють різноманітні показники – температуру, вологість, рівень вуглекислого газу, чадного газу, наявність диму та полум’я.

Мережа датчиків першого поверху та другого поверху мають локальні сервери, кожен з яких забезпечує отримання та обробку даних.

Центральний сервер обробки даних забезпечує зберігання отриманої інформації за певний період, її опрацювання та прийняття рішень стосовно подальших дій в залежності від ситуацій. Також він забезпечує зв’язок з мережею Інтернет та сповіщення користувачів.

Рисунок 1.2 – Другий поверх будинку

1.2.2 Функціональні особливості комп'ютерної системи

Комп'ютерна система – будь-який пристрій чи група взаємопов'язаних чи суміжних пристроїв, один або кілька з яких, діючи відповідно до програми, здійснює автоматизовану обробку даних.

Інформаційна система – це методи, які використовують для передачі, обробки або зберігання даних для досягнення конкретної мети[3].

Умовно можна поділити процеси, що використовують інформаційні системи різного походження[3]:

– внесення нової інформації з різних джерел;

– обробка нових даних та перетворення їх до зрозумілого вигляду;

– оброблені дані передаються користувачам або ці дані використовують зовсім інші системи;

– зворотний зв’язок – це дані, які були відкориговані людьми для щоб вхідні дані відповідали необхідним результатам.

На сьогоднішній день властивості сучасної інформаційної системи можна розділити[4]:

– різну інформаційну систему можна аналізувати, створювати та керувати відповідно до загальних принципів створення систем;

– розвиток інформаційних систем є динамічним;

– в момент створення системи слід дотримуватися системного підходу;

– остаточне рішення будується на вихідних даних;

Використання комп’ютерної системи може впливати на швидке отримання найкращих можливих рішень будь-яких завдань, спрощення щоденної тяжкої праці, за допомогою автоматизації вдосконалення структури інформаційних потоків (включаючи систему документообігу); надання споживачам унікальних послуг; зменшення витрат на виробництво продуктів та послуг (включаючи інформаційні)[5].

Комп’ютерні системи поділяються на типи[6]:

– вільне використання даних між різними комп’ютерами;

– при прийомі і передачі даних або інформації повинен взаємодіяти користувач.

Інтерактивна інформаційна система – це можливість передавати та обмінюватися даними через режим діалогу, тобто це інформаційно-обчислювальна система.

Керований потік, наприклад електронна пошта, яка має файл з формою HTML, запуск цього файлу розпочинає процес обробки документа або оператор електронної пошти отримує електронний документ, вказує формат, а потім запускає програму обробки. У такому варіанті, весь процес повинен контролюватися оператором.

Щоб використовувати такий процес без участі оператора, потрібно автоматизувати можливість прийому та обробки документів у конкретному форматі.

1.3 Аналіз сучасних методик організації комп’ютерних систем

В основі будь-якої комп’ютерної інформаційної системи лежить передача даних, а отже і методи її організації.

Протокол передачі даних — це передача даних різних програм, через конкретні правила інтерфейсу логічного рівня. Такі правила визначають конкретні способи обробки різних помилок та передача самих повідомлень через різні інтерфейси[3].

Також можливо передавати дані і через стандартні протоколи, така передача буде виконуватися на фізичному рівні. У такому разі передачу даних модна виконувати через різні прилади, не звертаючи увагу на якогось конкретного виробника, чи на апаратну платформу.

Мережевий протокол — це можливість передавати або обмінюватись інформацією з багатьма пристроями, які мають доступ до мережі.

Різні протоколи, часто описують тільки різні аспекти одного і того ж типу зв’язку, але якщо об'єднати їх разом вони створять набір протоколів.

Найчастіше застосовують моделі OSI, які розділяються на 7 рівнів і кожен з цих рівнів відповідаю своєму призначенню, починаючи з фізичного (створення всіх сигналів) до прикладного (передачі даних) [7].

Прикладний рівень є останнім між прикладною програмою і іншими рівнями – забезпечує зручний інтерфейс зв’язку мережевих програм користувача.

MQTT (Message Queue Telemetry Transport) – це більш простий мережевий протокол, який працює у форматі стеку TCP/IP. Його головна роль, це передача даних між різними пристроями на основі видавець-підписник.

Протокол MQTT передбачає два типи мережевих об’єктів: брокер повідомлень і ряд клієнтів. Брокер MQTT – це сервер, який отримує всі повідомлення від клієнтів, а потім направляє повідомлення до відповідних цільових клієнтів. Клієнт MQTT – це будь-який пристрій (від

мікроконтролера до повноцінного сервера), який запускає бібліотеку MQTT і підключається до брокера MQTT через мережу[8].

В комбінації з протоколом DDS (Data Distribution Service) MQTT доволі часто використовується для Інтернету речей (IoT), для побудови систем «розумний будинок».

Переваги використання протоколу MQTT:

– простий у використанні. Протокол може без жодних проблем використовуватися у різних складних систем, бо у нього немає зайвих функцій.;

– найкраще підходить для роботи з різними датчиками. Дозволяє публікувати раніше невідомі або нерозпізнані повідомлення;

– просте керування;

– не навантажує канали зв’язку;

– працює в різних критичних умовах.

MQTT використовує різні методи для того, щоб отримати необхідну дію, яку необхідно виконати на ідентифікованому ресурсі. Цей ресурс може бути заповненим будь-якими даними, чи вже готові дані, чи ті які будуть тільки створюватися, точно відповісти не можливо, так як це залежить від самого сервера. Найчастіше структура ресурсу ідентична усім файлам, які знаходяться на сервері.

1.4 Можливості системи «Розумний дім»

«Розумний будинок» – це свого роду інтелектуальна електропроводка, яка працює на базі програми, що управляє. Програма, у свою чергу, керує будинком та контролює роботу різних домашніх приладів та інженерних систем. Вона може керувати системами безпеки, кондиціювання та вентиляції, а також шторами, жалюзі, брамою або різними інженерними елементами, електроприладами [20].

Основними завданнями системи «Розумного дому» є безпека, автоматизація приміщень, комфорт, безперебійність, економія електроенергії.

Система «Розумного дому» виключає необхідність використання:

– пульти управління системами відеоспостереження та охоронно-пожежної сигналізації;

– блоки управління системами опалення та кліматичної техніки;

– панелі управління воротами, жалюзі та іншим;

– десятки вимикачів світла для керування освітленням;

– кілька пультів під час перегляду телебачення.

Рисунок 1.3 – Можливості «Розумного будинку»

1.4.1 Призначення та характеристики підсистем «Розумний будинок»

Під терміном «Розумний дім» зазвичай розуміють інтеграцію наступних підсистем у єдину систему управління будівлею [21]:

– система безпеки та моніторингу;

– системи управління та зв'язку;

– система дистанційного керування електроприладами, приводами механізмів та всіма системами автоматизації;

– система автоматичної механізації будівлі (відкриття/закриття воріт, шлагбаумів, електропідігрів підлоги тощо);

– система керування з одного місця аудіо-, відеотехнікою, домашнім кінотеатром, підсистема «Мультирум»;

– система електроживлення будівлі;

– система освітлення.

Усі підсистеми, які контролюються «Розумним будинком», можна поділити на такі групи:

– Безпека (проводиться контроль роботи систем доступу, відеоспостереження, стежить за атмосферою в будинку – загазованість, задимленістю, сигналізує про спроби зловмисників проникнути в будинок, про пожежу, контролює доступ до інформації та технічних систем).

– Інженерна (управління вентиляційною системою, системами кондиціювання, електропостачання, телекомунікацій та освітлення).

1.4.2 Підсистема пожежної безпеки контроль загоряння

Система миттєво приведе в дію існуючу протипожежну систему, відключить вентиляцію, щоб потік повітря не сприяв загорянню, електриці та газу. А якщо вас немає вдома, зателефонує і відправить тривожне SMS-повідомлення. Система також включить сирену та зовнішній світловий сигнал, щоб попередити сусідів про те, що сталося тривога.

Послідовність процесів системи при виникненні займання [21]:

– інформація про загоряння та час його виникнення заноситься в протокол повідомлень;

– інформація про датчик, який зафіксував спалах, заноситься в протокол повідомлень;

– вимикається електроенергія та електророзетки;

– перекривається подача газу;

– вимикається вентиляція;

– вмикається система димовидалення;

– вмикається зовнішній сигнал «ПОЖЕЖА»;

– викликається пожежна служба;

– здійснюється дзвінок за певними користувачем номерами під час загоряння;

– вимикається режим контролю загоряння.

1.4.3 Стисла характеристика області та умов застосування системи протипожежного захисту розумного будинку.

Можливості системи пожежної безпеки «Розумного будинку»:

– захист від спалаху;

– захист від замикання електромережі;

– захист від протікання у системах газопостачання;

– захист від наслідків аварій інженерних систем.

Система пожежної безпеки в «Розумному будинку» вирішує такі завдання, слідуючи за певною схемою:

а) Здійснює постійний контроль стану електропроводки, повітря, наявності залишених включеними побутових електроприладів (чайник, праска, мікрохвильова піч і т.д.) – при виявленні система автоматично їх відключить.

б) Якщо сталося загоряння:

– забезпечує евакуацію мешканців;

– автоматично відключить систему вентиляції, щоб унеможливити надходження кисню всередину будинку, електроживлення, надходження газу;

– одночасно включає систему видалення диму;

– пожежна сигналізація у «Розумному будинку» передає сигнал про виникнення пожежі у спеціальні служби;

– повідомляє інформацію власників про виникнення спалаху. Якщо в цей момент їх немає вдома, то, отримавши сигнал, вони можуть дистанційно вжити заходів;

– розпочинає боротьбу з вогнем.

Автоматизована система протипожежної безпеки – це цілий комплекс сучасних технічних засобів, програмного забезпечення та програмних модулів, що керують роботою усієї системи.

Основними технічними засобами є надчутливі датчики. Вони поділяються на такі види:

– димові – оснащені фотоелементами та світлодіодами;

– газові – мають вбудований газоаналізатор, який вловлює будь-яку концентрацію CO;

– теплові – оснащені температурним реле;

– комбіновані – мають у своєму складі кілька пристроїв, запобігають хибним сигналам.

Система пожежної безпеки «Розумного будинку» забезпечує більш високий рівень надійності для людини та її оселі у порівнянні зі звичайною системою.

Основні переваги:

– запобігання ризикам спалаху або оперативне гасіння невеликого вогнища з метою недопущення великої пожежі;

– створення умов для безпечної евакуації людей та оперативного виклику пожежників;

– надсилання повідомлення про позаштатну ситуацію власникам будинку на мобільний телефон, вони можуть керувати системою через програми;

– часткове перекриття подачі електрики та газу. Якщо не зачеплені основні мережі, то відключаються лише деякі приміщення;

– обладнання системи пожежної безпеки має естетичний зовнішній вигляд та добре вписується у будь-який інтер'єр.

1.5 Завдання і мета роботи

Метою кваліфікаційної роботи є розробка комп'ютерної системи протипожежного захисту розумного будинку з віддаленим керуванням через сервер MQTT.

Відповідно до завдання комп'ютерна система повинна забезпечувати ефективну роботу в плані моніторингу та попередження потенційних пожежних ситуацій, а також можливих дій по їх запобіганню, в тому числі – повідомлення власників будівлі про можливу небезпеку.

Найкращим варіантом рішенням цієї задачі є використання протоколу MQTT для реалізації взаємодії систем розумного будинку.

В якості центру обробки даних є доцільним використання одноплатного комп'ютера Raspberry Pi, що може виступати також в ролі сервера для зв'язку з мережею Інтернет та забезпечувати роботу системи в цілому.

При розробці КС необхідно виконати такі задачі::

– створити в Packet Tracer модель «Розумного будинку» з пристроями та датчиками протипожежної безпеки;

– розробити правила взаємодії пристроїв протипожежної безпеки з іншими приладами системи та з навколишнім середовищем або їх поєднання;

– налаштувати роботу пристроїв пожежної безпеки;

– налаштувати систему моніторингу протипожежного стану по протоколу MQTT;

– налаштувати систему моніторингу протипожежного стану через веб-сервер;

– промоделювати керування пристроями протипожежної безпеки через веб-інтерфейс та MQTT-брокер та порівняти різницю у часі;

– протестувати протипожежну безпеку «Розумний будинку».

У результаті виконання поставленої задачі буде створена мережа «Розумного будинку» з комбінованим типом зв'язку, можливістю

налаштування конфігурацій через веб-інтерфейс, перебуваючи вдома або віддалено.

2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ КОМП'ЮТЕРНОЇ СИСТЕМИ

2.1 Розробка технічних вимог до комп'ютерної системи

2.1.1 Вимоги до системи в цілому

2.1.1.1 Структура і функціонування системи

Система, яка знаходиться в будинку повинна забезпечувати:

– збирання даних: постійне збирання даних з усіх датчиків будівлі, та передача цієї інформації для подальшої обробки та аналізу;

– обробка та аналіз отриманих даних: коли всі дані отримані система аналізує на скільки дані відрізняються від нормальних показників;

– збереження отриманих даних: зберігання тих самих отриманих даних для формування звітів процесу роботи. Завдяки цій функції буде відбуватися резервне збереження і копіювання даних на серверах;

– реакція на отримані результати: система розумного будинку повинна належним чином реагувати на зміну показників. Якщо вони виходять за вказані межі або спостерігається їх аномальна зміна – потрібно виконати ряд мір безпеки (що доступно для системи в автоматичному режимі) та повідомити користувача чи кількох користувачів (мешканців будинку) про ситуацію.

2.1.1.2 Вимоги до показники призначення

В системі треба забезпечити роботу всіх функцій відповідно до умов технологічного процесу:

– технології захисту;

– віддалений доступ;

– постійна робота;

– збереження даних на облаці.

2.1.1.3 Вимоги до експлуатації

При повній автоматизації та використані протипожежної системи у розумному будинку обслуговувати цю систему дозволяється тільки кваліфікованим інженерам. Обслуговувати систему необхідно кожний рік, щоб запобігти поганим наслідкам.

Системний ремонт або заміна елементів у системі, які стали не працездатні на нові дозволяється тільки спеціалістами. Заміна датчика вуглекислого газу MH-Z19B, диму MQ-2, температури та вологості DHT21/AM2301A та рівня чадного газу MQ-7 необхідно робити на проміжну восьми років. Заміна інші складових необхідно робити до десяти років.

2.1.1.4 Вимоги до патентної чистоти

У системі необхідно використовувати тільки пристрої або програми, які мають ліцензії та сертифікати які дозволені у використані на Українській території.

2.1.1.5 Додаткові вимоги

Тип кабелю повинен відповідати проектним розрахункам. Може бути вита–пара з екранованою оболонкою або оптоволокно. Використовується кабель типу UTP cat.5. Повинні встановлюватись інформаційні розетки.

При аварійних ситуація, при відмові роботи деяких з датчиків або сенсорів не повинна впливати на працездатність протипожежної системи.

Також у таких системах необхідно використовувати, резервні джерела живлення для того щоб, система працювала при відключенні або з перебоями електроенергії, до тих пір поки не налагодиться основне джерело енергії[9].

При проектуванні системи потрібно врахувати усі можливі варіанти загроз, а також потрібно визначити на які дані спрямовані ці загрози(керування системи, конфіденційності).

Класифікація загроз:

1) джерела загрози:

a. антропогенні:

– кримінальні структури, злочинці, хакери;

– обслуговуючий персонал.

b. техногенні:

– засоби зв’язку, комунікаційні мережі;

– неякісні технічні засоби обробки інформації, програмні засоби;

c. стихійні:

– пожежі, землетруси, повені, урагани;

2) вразливості безпеки:

a. об’єктивні;

– супутні технічним засобам випромінювання, що визначаються особливостями елементів, які визначаються особливостями об’єкта, що захищається.

b. суб’єктивні

– помилки та порушення;

c. випадкові:

– збої, відмови та пошкодження.

Для захисту від зовнішніх електромагнітних полів використовуйте екрановані кабелі витої пари. Щоб запобігти коливанням напруги в мережі, на вихідних кабелях необхідно встановити вимикач. Серверні приміщення повинні вентилюватися, щоб забезпечити нормальні кліматичні умови. Він також повинен бути заземлений.

2.1.2 Вимоги до функцій які виконує система

Вся система повинна працювати цілодобова, крім цього постійно працюють різні рятувальні служби. При відключення джерела, система повинна перейти на додаткові джерела. Всі датчики, сиренна та й вся система повинна мати зв'язок один з одним. Також в системі повинна бути можливість до розширення. Також система постійно збирає дані, аналізує, оброблює, зберігати на сервері.

2.1.3 Вимоги до видів забезпечення

2.1.3.1 Вимоги до інформаційного забезпечення системи

Система розумного дому – це інформаційний об’єкт, який вразливий до інформаційних загрозам. Не існує якогось одного варіанту захисту даних в системі, бо кожна загроза залежить від способу створення системи, які технології використані.

На сьогодні майже кожна система має в собі вбудовані компоненти інформаційної безпеки. Але деякі системи досі не використовують ці компоненти.

Дуже важливо враховувати основні фактори при створені комп’ютерній системі [10]:

– можливість до шифрування даних та зв’язку між різними датчиками системи;

– можливість контролювати та робити деякі зміни на відстані;

– повинна бути велика здатність до захисту віддаленого доступу.

На сьогодні немає якогось одного методу створення безпеки інформації в розумному будинку, бо при побудові системи використовують різні структури захисту [11].

Приклад найбільш частих загроз які виникають:

– напади на центральний сервер так званих «Хакерів»

– шкідливі вірусні програми, які руйнують працездатність системи;

– крадіжка даних з каналів зв’язку;

– доступ до мережі неавторизованих користувачів;

– наявність порушників у числі обслуговуючого персоналу;

– помилки користувача;

– крадіжка;

– перебої у мережі електропостачання;

– стихійні лиха;

– поломка апаратних складових системи;

– помилки програмного забезпечення.

2.1.3.2 Вимоги до лінгвістичного забезпечення

Для користувачів система повинна бути тільки на українській мові.

2.1.3.3 Вимоги до системи енергозабезпечення

Надійність, як одна з важливих вимог до систем енергопостачання, визначається числом незалежних джерел живлення та важливою схемою електропостачання. За надійністю електричне постачання відповідно до вимог правила влаштування електроустановок забезпечення безпеки робіт для електротехнічних пристроїв поділяють на[9]:

– надійність електричного постачання;

– якість електроенергії;

– модифікаційна можливість розвитку розумного дому;

– мало затратні ресурси;

Такі вимоги враховують на початку проектування та у процесі використання розумних джерел. Система частина енергетичних систем та в енергетичному завданні більш проста та більш ускладнена у плані використання та перетворення електричної енергії в технологічних цілях забезпечення електричної енергії. Електроприймачі як електричної частини технологічних пристроїв входять незамінними елементами в систему і багато в чому визначають якісну роботу такої системи та її параметрів.

При проектуванні споживачів електричної енергії в основному систематизують за надійністю електричного постачання, режимів роботи, потужності, напруги та роду струму.

2.1.4 Вимоги до схоронності інформації при аваріях

Причиною виникнення аварійних ситуації може бути будь-що. Щоб інформація не зникла при аварійній ситуації, систему налаштовують на збереження даних на хмарний диск.

2.2 Вимоги до функцій, виконуваних системою

2.2.1 Перелік функцій, задач комплексів

Структура системи розумного дому має 3 рівні(нижній рівень, середній рівень та верхній рівень).

До нижнього рівня відносяться різні датчики(температурні датчики, вологості та інші).

Середній рівень включає в себе контролер Arduino Mega, який підключає до себе всі датчики.

Верхній рівень, це SCADA, який виконує роль керування системою «розумний дім». На сьогодні вже створенні SCADA-системи розумного дому, але вони створені на основі промислових контролерах і мають дуже велику вартість. Система, яку ми створюємо розрахована на використані контролера Arduino Mega, яка коштує набагато менше і доступна до споживача середнього рівня, в чому і полягає актуальність даної роботи

2.2.2 Перелік функціональних підсистем

В комп'ютерній системі використовується декілька функціональних систем що використовує різне програмне забезпечення. Це – системне, захисне, прикладне та спеціалізоване.

До системного належить: MQTT 5.0 Broker, ОС Windows 10, ОС Android, iOS.

До захисного відноситься антивірус Avast Antivirus.

Прикладне ПО – це Google Chrome, Opera.

До спеціалізованого програмного забезпечення відноситься MQTT 5.0 Desktop Client.

2.2.3 Вимоги до регламенту і якості реалізації функцій

Якщо відокремлений об'єкт не зберігає інформацію, яку він містить, доки користувач або процес не буде надано і попередні права доступу до об'єкта не будуть відкликані, комп'ютерна система повинна надати послугу

повторного використання об’єкта. Реалізація даного сервісу забезпечує захист від атак типу «збір сміття».

Служба обміну конфіденційністю захищає об’єкти від несанкціонованого доступу до інформації, що міститься в них, під час експорту/імпорту через незахищене середовище.

Точки відновлення — це універсальна послуга, яка дозволяє відновлювати роботу після помилок користувача, збоїв програмного або апаратного забезпечення, а також підтримувати цілісність баз даних, програм на основі транзакцій тощо.

Реєстрація дозволяє вам контролювати небезпечну діяльність у вашій комп’ютерній системі. Рівень цієї послуги залежить від цілісності та складності методів аналізу даних журналу та можливостей виявлення загрози.

2.3 Вимоги до видів забезпечення

2.3.1 Інформаційне забезпечення системи

Користувач отримує лише готовий результат свого запиту, а головний процес обробки інформації виконується через сервер. Сервер даних використовує хмарний сервіс для обробки та зберігання даних. Найкраще є виборів використання рішення від компанії Microsoft – Azure IoT.

Передача даних між системними компонентами.

Передача даних від компонентів через мережу повинно використовуватися PHY - LAN PHY і WAN PHY, через стандарт IEEE 802.3a

2.3.2 Технічне забезпечення системи

Технічні характеристики точки доступу: забезпечення стабільної роботи мережі WiFi, рекомендована робоча частота 5 GHz дозволить позбавитися перешкод він іншого обладнання, а також завантаженості мережі він інших пристроїв та мереж.

2.3.3 Вимоги до організаційного забезпечення

Точний поділ прав доступу користувачів до програмного забезпечення та технічних даних, що має включати розподіл прав доступу до робочого місця, реєстрацію користувачів, що ввійшли та видалення вірусів.

Доступ до даних функціонального блоку має базуватися на матриці доступу та надавати можливість змінювати дані в момент роботи.

.

2.3.4 Вимоги до складу нормативно-технічної документації системи

Складові, які повинні входити:

– креслення, які відповідають нормативам;

– маркувальні позначки розеток, датчиків, кабелів;

– схема підключення кабельної проводки;

– перелік з’єднань кабелів;

– плани розміщення обладнання в шафах або стійках;

– програма і методика випробування.

2.4 Вибір і обґрунтування структурної схеми комплексу технічних засобів комп'ютерної системи

Структура системи розумного будинку визначається функціональними одиницями, які включає система і зв'язками між ними.

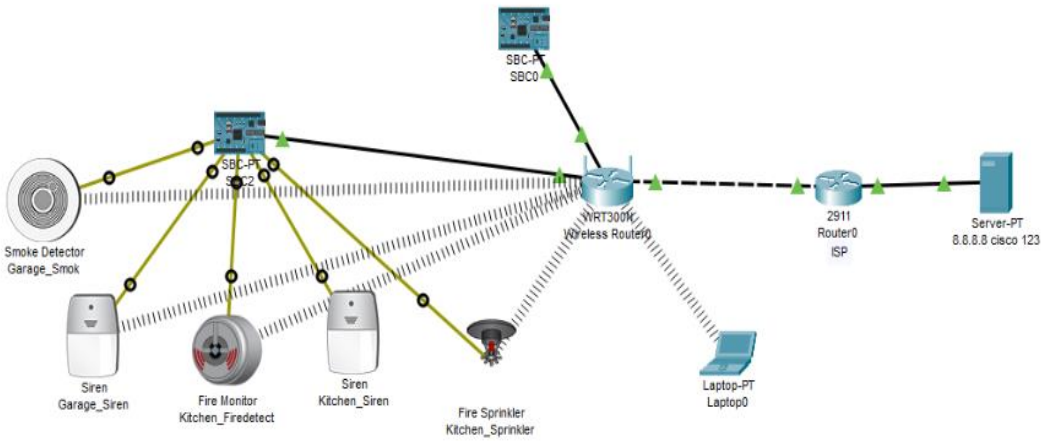

Рисунок 2.1 – Структура схеми системи розумного будинку

Для найкращої роботи лінійний блок повинен бути первинною потужністю, а функціональним блоком необхідно забезпечити правильну роботу першої конструкції.

Інформаційний доступ до різних ресурсів у домі повинен бути розмежованим.

Для того, щоб отримати повний доступ до деяких даних, необхідно ввести до системи класовий пароль.

Оскільки доступ до системи може здійснюватися за допомогою безпровідної мережі, у такому випадку необхідно звертати увагу на захист даних.

2.5 Розробка специфікації апаратних засобів комп'ютерної системи

2.5.1 Вибір та характеристики пристроїв керування

Першочергова проблема, яку необхідно вирішити – обробка отриманої інформації від датчиків. Рішенням цієї проблеми стала мікроконтролерна

платформа Arduino модель Mega 2560 R3. Який в якості USB-UART перехідника має додатковий мікроконтролера ATMega16u2, в якому не потрібно встановлювати додаткові драйвери і має найбільшу швидкість, ніж в попередніх моделях[12].

Однією з основних задач при проектуванні системи розумного будинку є забезпечення зв'язку системи з користувачем та доступу до Інтернету. Для реалізації даних цілей було обрано дводіапазонний маршрутизатор TP-Link Archer AX53. 4 фіксовані високопродуктивні антени з яких формується потужний сигнал, що збільшує радіус дії. Технологія Beamforming концентрує безпровідний сигнал у напрямку клієнтів для розширення радіусу дії Wi-Fi. Модуль FEM покращує потужність передачі для кращого покриття сигналу[13].

В якості сервера було взято одноплатний комп'ютер Raspberry Pi 4 Model B, який має 64-розрядний чотириядерним процесором з тактовою частотою 1,5 ГГц (ARM Cortex-A72), підтримкою двох дисплеїв з роздільною здатністю до 4K, оперативною пам'яттю до 8 ГБ, дводіапазонної безпровідної мережі 2,4/5,0 GHz, Bluetooth 5.0/BLE, True Gigabit Ethernet, USB 3.0 і можливість PoE. На ньому можна запустити повноцінний дистрибутив Linux або ж спеціалізовану систему розумного будинку Windows 10 IoT Core з доступом до технології Microsoft Azure[14].

Всі датчики, які представлені нижче мають велику чутливість, роздільну здатність, стабільну роботу. Також вони всі найкраще підходять в роботі з платформою Arduino, завдяки цьому вся робота буде проходити набагато швидше.

Таблиця 2.1 – Специфікація обладнення

№	Найменування і технічна характеристика	Тип, марка, позначення документа, опитувального листа	Одиниці виміру	Кількість	Примітки
1	Плата Arduino Mega 2560 R3 мікроконтролер: ATmega16u2; цифрові входи/виходи:54 14 PWM аналогові входи: 16. флеш-пам’ять: 256 KB ОЗП: 8 KB; тактова частота: 16 MHz.	SBC0 MQTT Client1 SBC2 MQTT Client2	од.	2	Основа всієї система.
2	Маршрутизатор TP-Link Archer AX53 Стандарт WI–FI 802.11: 2,4ГГц ax/n/b/g; 5ГГц x/ac/n/a. Порти LAN: Gigabit Ethernet (100/1000). Підтримка протоколів: PPPoE, PPTP, DHCP, DDNS, L2TP,VPN.	Router0 Router1	од.	2	Зв'язок з всією мережою
3	Сервер Raspberry Pi 4 Процесор: BCM 2711 Cortex-A72 Кількість ядер:4. Частота: 1.5 ГГц. Об'єм встановленої пам'яті: 8 ГБ	Server-PT	од.	1	Оброблює всю інформацію

Продовження таблиці 2.1

4	Датчик чадного газу MQ-7 навантажувальний опір: 10 К виявлення концентрації газу:10-1000 ppm; час розігріву: від 60 (напруга підігрівача 5В) до 90 секунд (для напруги підігрівача 1,4 В	MQ-7	од.	10	Вимірює шкидливий газ
5	Дитчик температури та вологості виробник: AOSONG; DHT21 AM2301A; точність: 0.1 ºC; діапазон вимірювання вологості: 0-100%; діапазон виміру температури: -40 ~ 80 ºC; точність вимірювання вологості: ± 2% RH; точність вимірювання температури:± 0.5%;	DHT21 AM2301A	од.	10	Слідкує за температурою, при перевищеню температури спрацювує.
6	Датчик диму MQ-2 діапазон: 300-10000 ppm; Rs опір елемента 20 кОм 50ppm толуол;	MQ-2	од.	10	Реагує на дим
7	Датчик полум'я KY-026 хвилі від 760 нм до 1100 нм дальність виявлення вогню1м.	KY-026	од.	10	Реагує на полум'я

3 МОДЕЛЮВАННЯ МЕРЕЖІ РОЗУМНОГО БУДИНКУ В CISCO PACKET TRACER.

Змодельована мережа «Розумний будинок» у середовищі Cisco Packet Tracer представлена на рисунку 3.1. Для представлення моделювання обрано 2 приміщення з контролю протипожежної безпеки: гараж та кухня.

Розумна домашня мережа складається з багатьох провідних та бездротових пристроїв IoT та пристроїв мережевої інфраструктури.

Бездротовий маршрутизатор Linksys WRT300N це концентратор і маршрутизатор для всіх внутрішніх пристроїв. Всі внутрішніх пристрої під'єднані до домашньої мережі через Wi-Fi та відслідковуються за допомогою Server IoT. Домашні пристрої можуть підключатися до домашнього шлюзу через бездротовий та дротовий зв'язок. Домашній шлюз має одну загальнодоступну IP-адресу (180.16.0.2/16), яку призначає його постачальник послуг Інтернету (ISP) (рис.3.2) , і поставляється з вбудованим серверним модулем протоколу динамічної конфігурації хосту (DHCP), який розподіляє приватні IP-адреси своїм локальним хостам та пристроям IoT у межах діапазон його пулу IP-адрес (192.168.0.0/24) (рис.3.3).

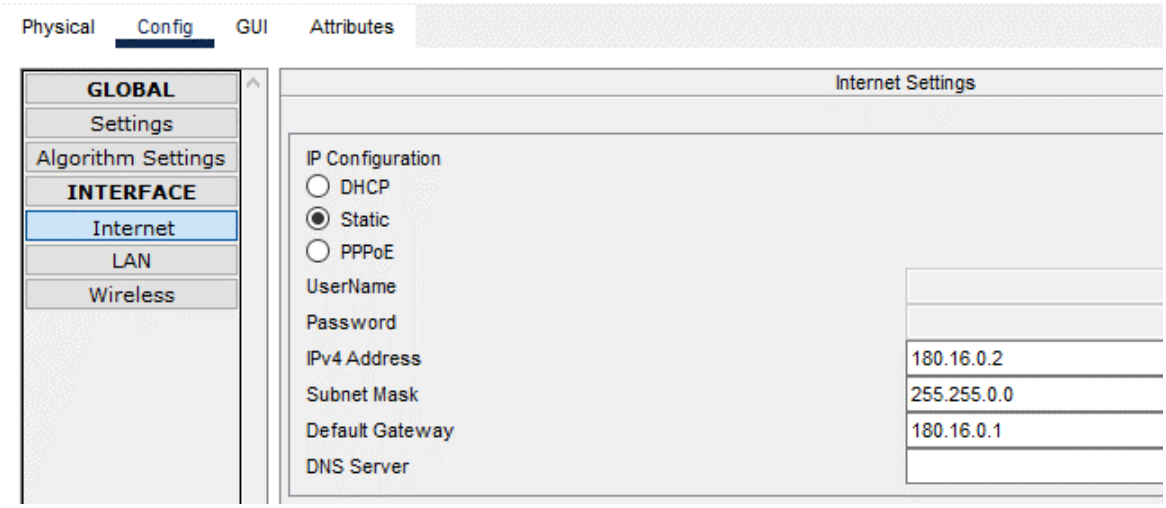

Рисунок 3.1 – Налаштування Internet на домашньому маршрутизаторі

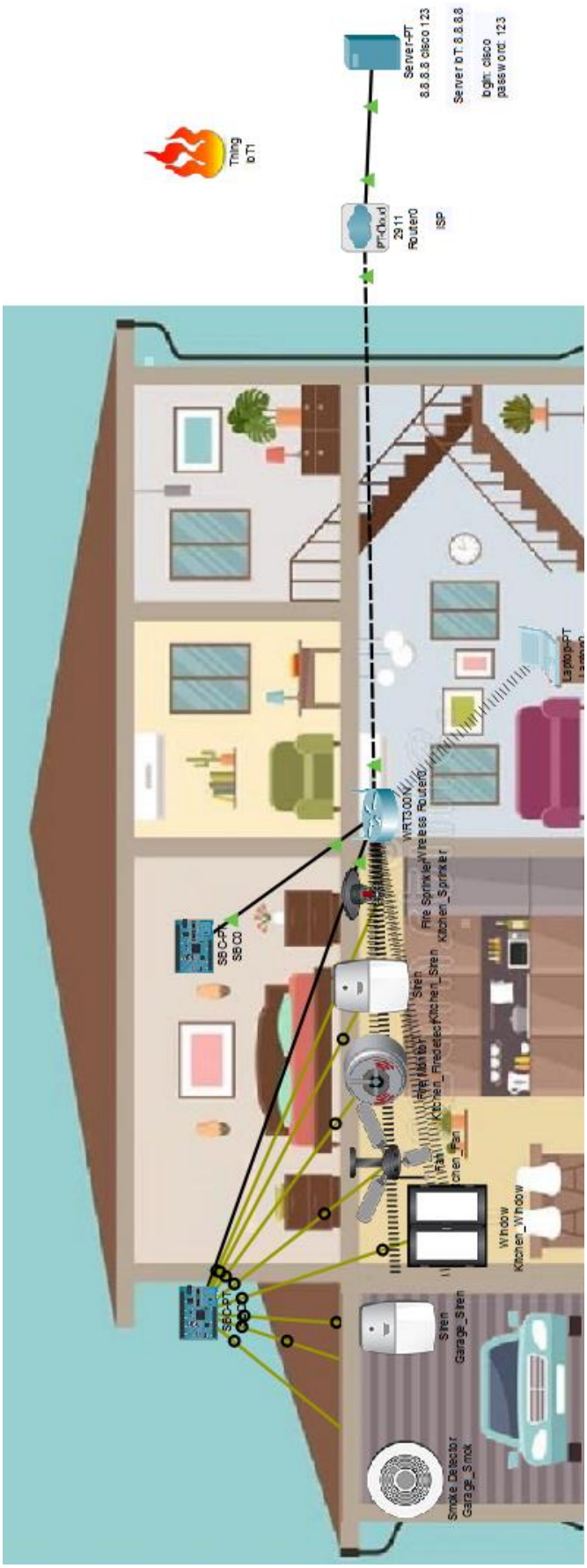

Рисунок 3.2 – Загальний вигляд системи «Розумний будинок»

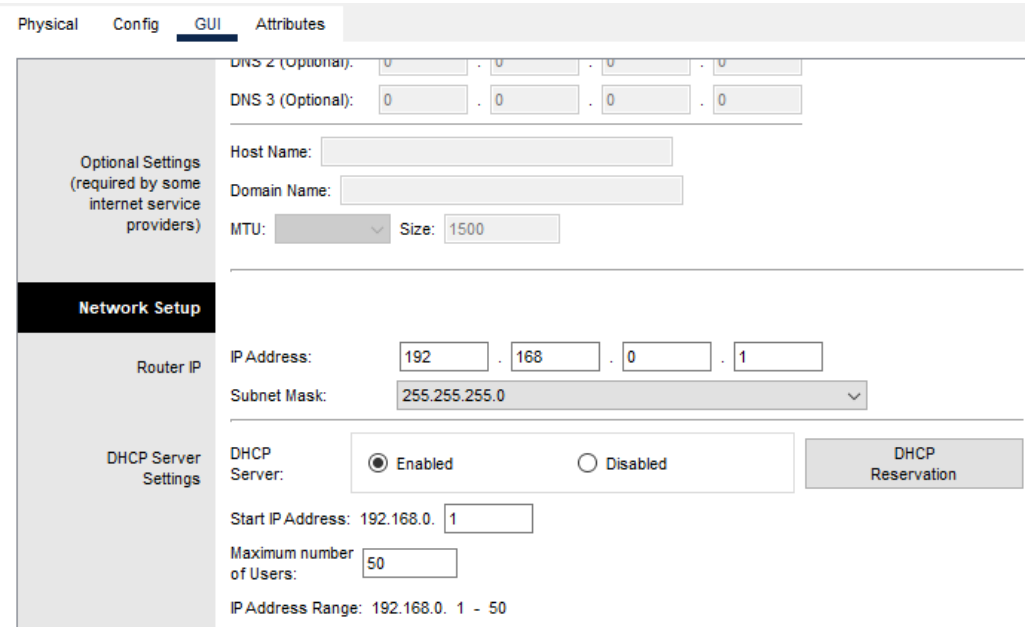

Рисунок 3.3 – Налаштування DHCP бездротового маршрутизатора

Маршрутизатор ISP виконує роль постачальник послуг Інтернету. Мережні налаштування представлено на рис. 3.4.

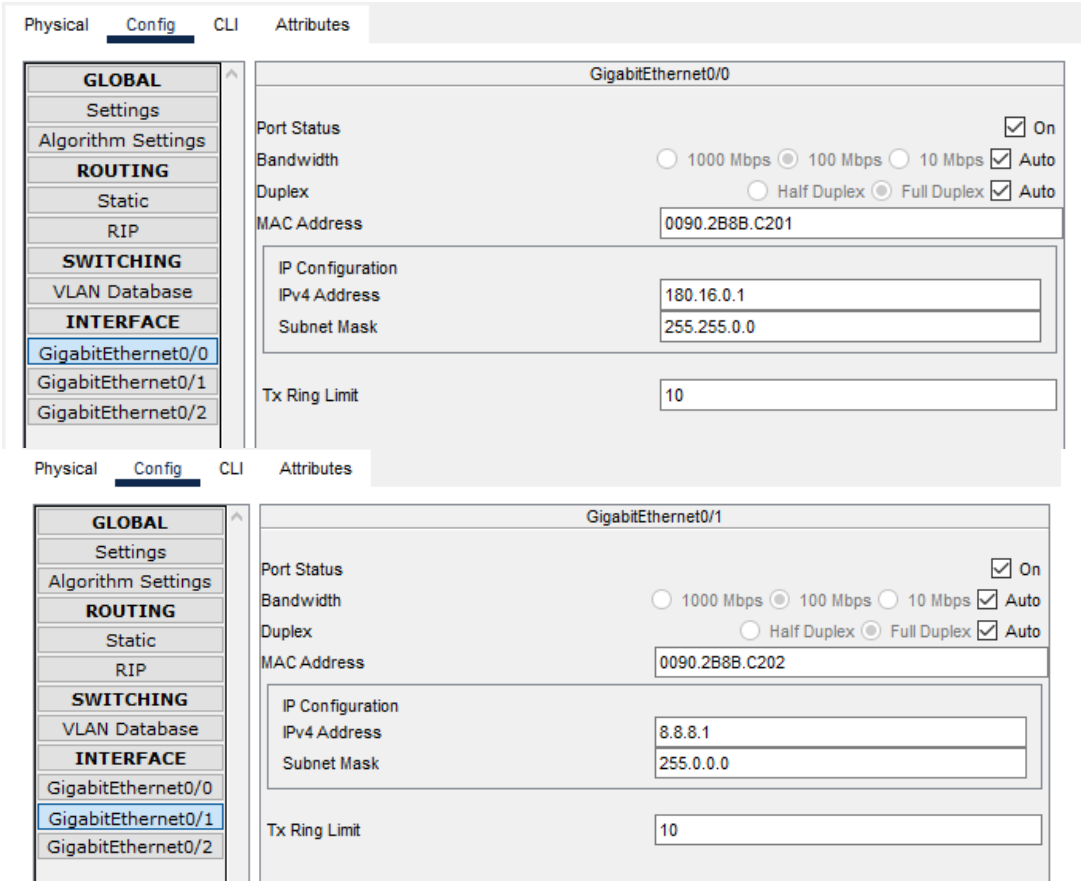

Рисунок 3.4 – Мережні налаштування ISP

Server IoT надає веб-інтерфейс, який дозволяє користувачам здійснювати моніторинг та керування різними розумними домашніми пристроями дистанційно через будь-який комп'ютер вдома. Він також виконує роль брокера MQTT. Йому надана публічна адреса 8.8.8.8/8. На вкладці Service включено сервіс IoT та створено користувача cisco (рис. 3.5).

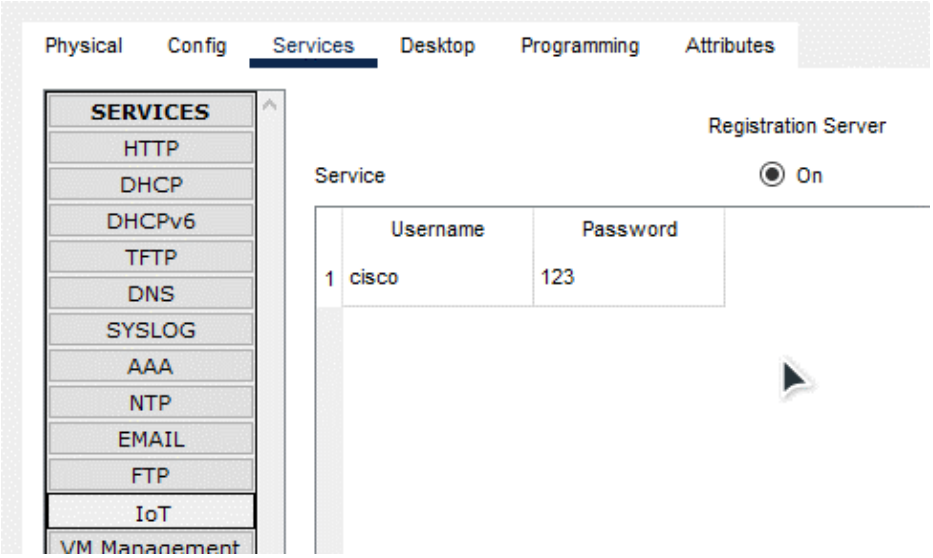

Рисунок 3.5 – Вікна властивостей сервера

Сирени та детектори диму передають тривожне повідомлення про пожежу та інформацію, місце розташування джерела пожежі (або загоряння) на контролери SBC. Кабель, що використовується для підключення пристроїв IoT до контролера SBC, називається IoT Custom Cable.

Контролери SBC всередині кожної кімнати відповідають за надання керування пристроями IoT в мережі. Вони мають зв'язок із кожним пристроєм IoT. SBC, додані в розумний дім, використовується для контролю за рівнем диму, зчитаним датчиком диму, і вирішує, чи потрібно ввімкнути сирену, відкривати вікна чи двері. Якщо рівень монооксиду вуглецю вищий за поріг, SBC запрограмований на автоматичне відкриття вікна, передніх дверей, гаражних дверей. Ця дія лише повертається (зачиняються двері та вікна), коли рівень окису вуглецю знизиться нижче порігу.

Контролери SBC також виконують роль клієнта MQTT. У цьому експерименті SBC діє як видавець MQTT, перевіряючи рівень диму, а клієнти IoT діють як абоненти MQTT для моніторингу стану пристроїв IoT.

У розумному будинку є ноутбук, за допомогою якого можна моніторити стан підключених пристроїв, а також керувати ними.

3.2 Моделювання в Packet Tracer системи протипожежної безпеки «Розумного будинку» з керуванням через web-сервер

Сервер IoT може бути web-сервером для IoT-пристроїв або брокером MQTT. Розглянемо випадок, коли IoT-сервер діє як HTTP-сервер. Server IoT надає web-інтерфейс, який дозволяє користувачам здійснювати моніторинг та керування різними розумними домашніми пристроями дистанційно через будь-який комп'ютер вдома.

Необхідно підключити пристрої до бездротової домашньої мережі. Для цього на кожному пристрою необхідно надати унікальну назву та в мережних налаштуваннях вказати отримувати мережну адресу по DHCP.

Щоб підключити пристрої до IoT web-серверу в розділі IoT Server необхідно вказати його дані (рис. 3.6).

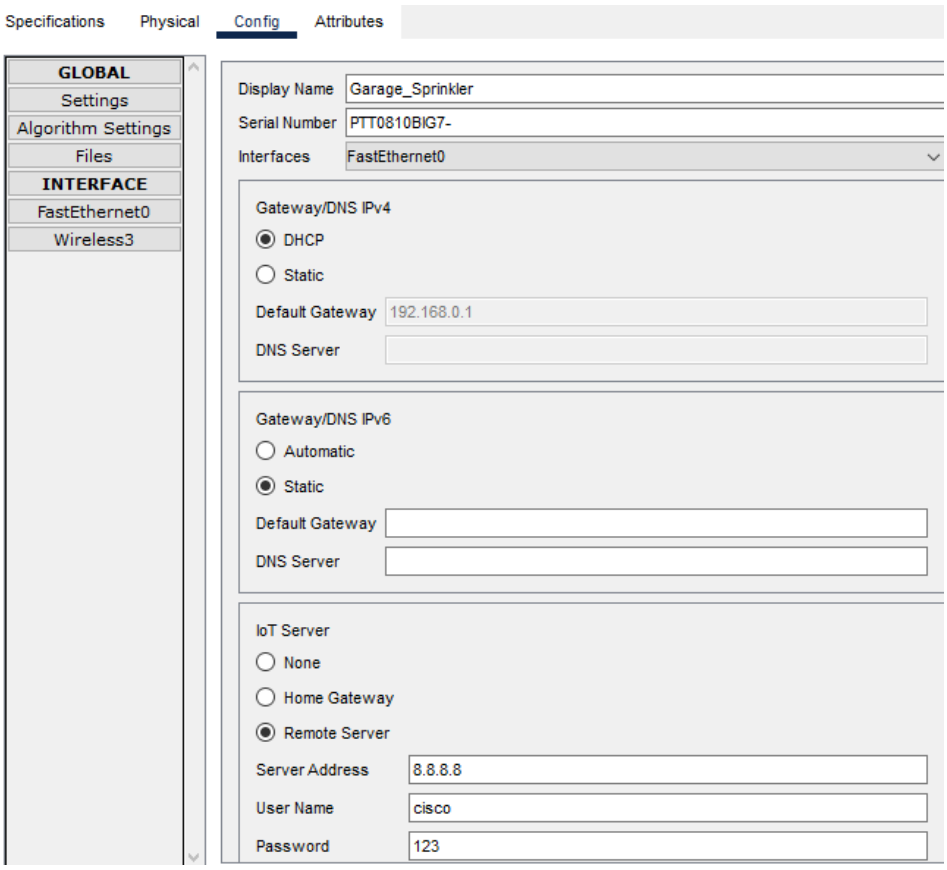

Рисунок 3.6 – Налаштування пристроїв IoT

Зробимо підключення до web-інтерфейсу IoT сервера, для цього зайдемо на ноутбуці у Web Browser і введемо в поле URL IP адресу IoT сервера 8.8.8.8.

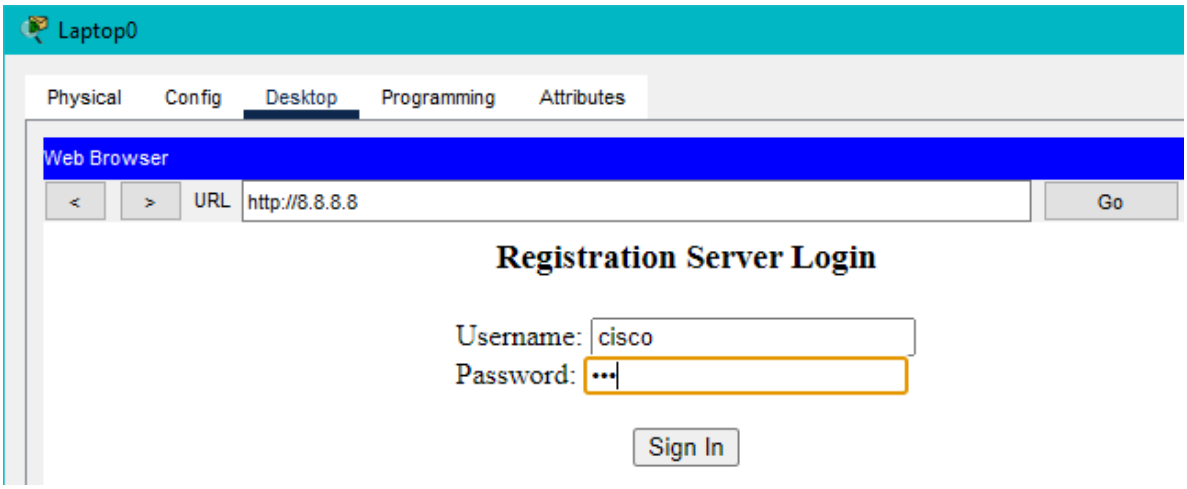

Рисунок 3.7 – Сторінка авторизації для входу на IoT Сервер

Вводимо логін: cisco та пароль: 123 і нам відкривається список усіх підключених IoT пристроїв (рис.3.7).

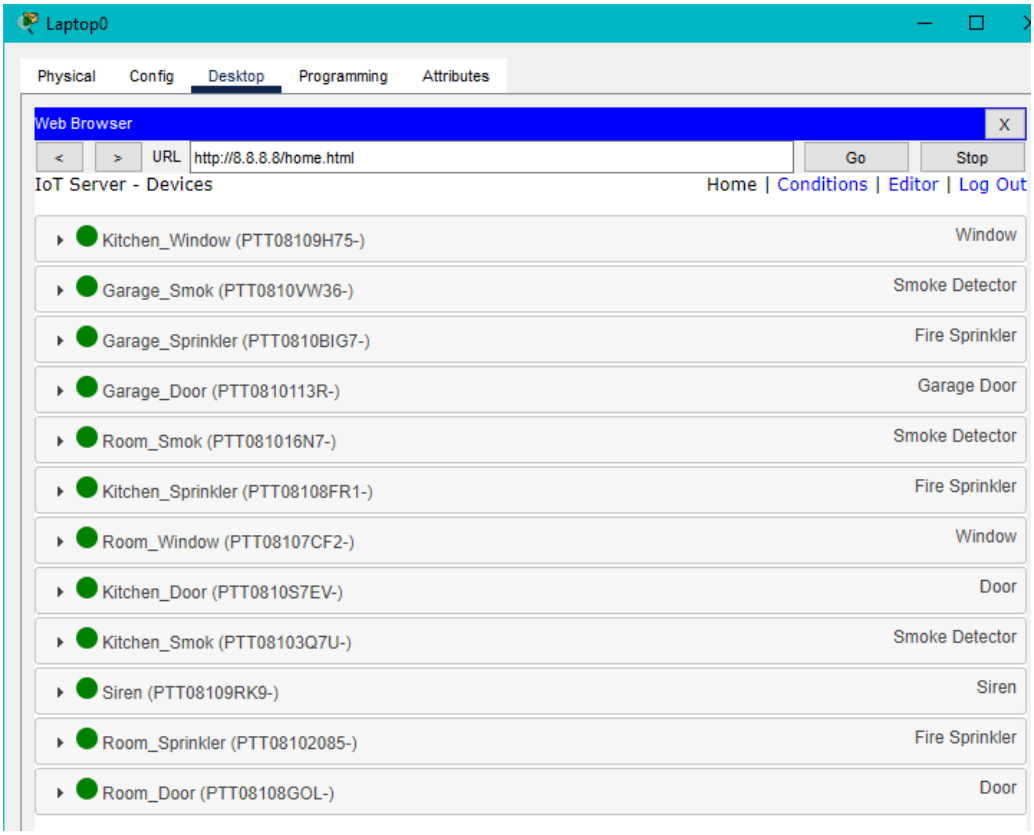

Рисунок 3.8 – Список підключених пристроїв

Як простий демонстраційний сценарій розглянемо автономну систему кухні в залежності від полум'я. Використовуємо полум'я, щоб змінити температуру в приміщені. Щоб реалізувати це, нам потрібно, щоб пристрій IoT, у даному випадку датчик полум'я, який надсилає інформацію на сервер IoT. Сервер IoT постійно контролює отримані дані про температуру. Коли значення перевищує 0,5, сервер надсилає команду On іншому IoT-пристрою, у цьому випадку вікну, яке його відкриває, вмикається стельовий вентилятор та вмикається розпилювач води. Відповідне налаштування показано на рисунку 3.9.

Створення правила увімкнення та вимкнення для спрацьовування пожежного захисту, для цього зайдемо на IoT сервер через web-браузер і виберемо вкладку Conditions, і в ній створимо правила за умовами.

Laptop0

Physical | Config | Desktop | Programming | Attributes

Web Browser

< > URL http://8.8.8.8/conditions.html Go Stop

IoT Server - Device Conditions

Home | Conditions | Editor | Log Out

Actions	Enabled	Name	Condition	Actions
Edit Remove	Yes	Garage_Alarm_On	Garage_Smok Level > 3	Set Garage_Sprinkler Status to true Set Garage_Door On to true
Edit Remove	Yes	Garage_Alarm_Off	Garage_Smok Level <= 3	Set Garage_Door On to false Set Garage_Sprinkler Status to false Set Siren On to false
Edit Remove	Yes	Kitchen_Alarm_On	Kitchen_Smok Level > 1	Set Kitchen_Window On to true Set Kitchen_Sprinkler Status to true Set Kitchen_Door Lock to Unlock Set Siren On to true
Edit Remove	Yes	Kitchen_Alarm_Off	Kitchen_Smok Level <= 0.5	Set Kitchen_Window On to false Set Kitchen_Sprinkler Status to false Set Siren On to false
Edit Remove	Yes	Room_Alarm_On	Room_Smok Level > 1	Set Room_Window On to true Set Room_Door Lock to Unlock Set Room_Sprinkler Status to true Set Siren On to true
Edit Remove	Yes	Room_Alarm_Off	Room_Smok Level <= 0.5	Set Room_Window On to false Set Room_Sprinkler Status to false Set Siren On to false

Рисунок 3.9 – Додавання правил у логіку роботи пристроїв

Тестування роботи системи протипожежного захисту розумного будинку (див. рисунок 3.10).

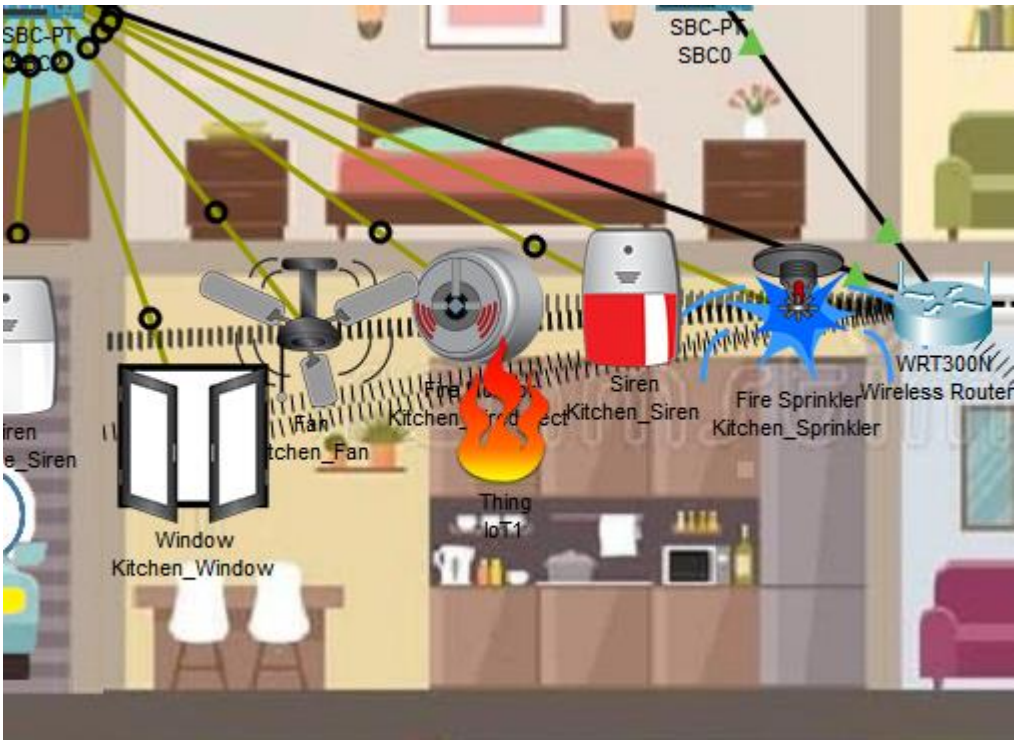

Рисунок 3.10 – Робота системи протипожежного захисту

В режимі симуляції час, що пройшов від отримання веб-сервером сигналу про пожежу та ввімкнення всіх пристроїв 0,845 с (рис. 3.11).

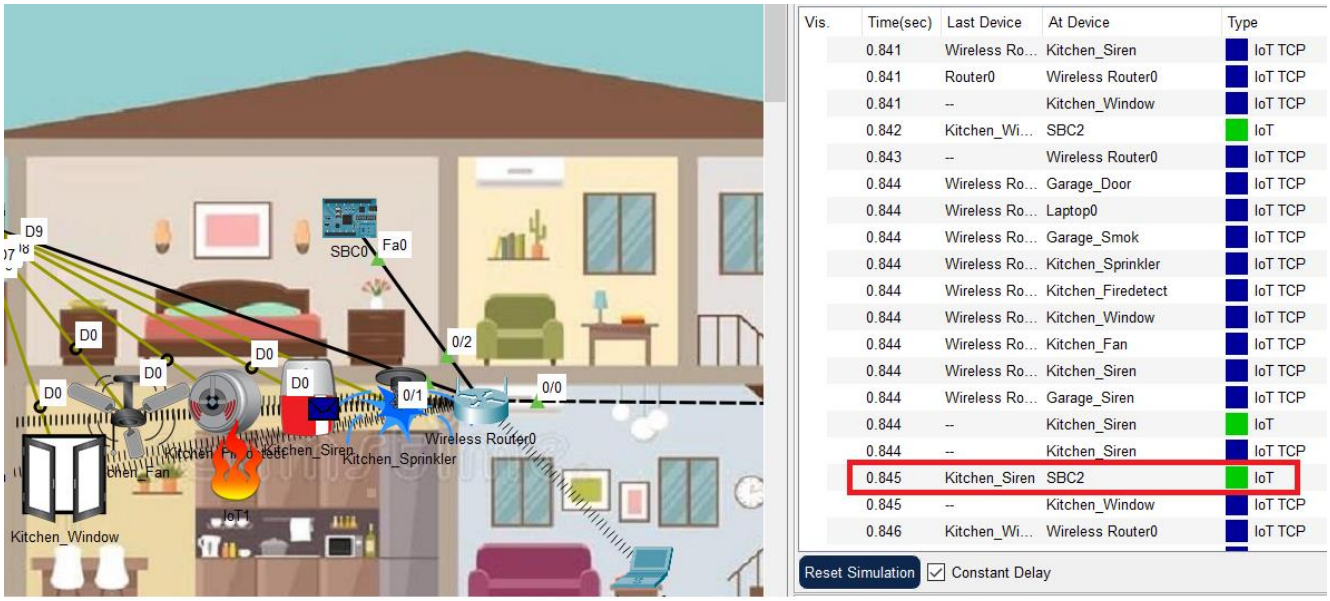

Рисунок 3.11 – Тестування в режимі симуляції

3.3 Моделювання в Packet Tracer системи протипожежної безпеки «Розумного будинку» з керуванням через MQTT-сервер

3.1 Огляд протоколу MQTT

MQTT (Message Queue Telemetry Transport) - невеликий, відкритий і маловаговий протокол обміну даними. Він використовується передачі інформації між віддаленими локаціями при обмеженої пропускної спроможності каналу і невеликого розміру коду. Ці особливості дозволяють застосовувати їх у Промисловому Інтернеті речей (IIoT), при Машинно-Машинному взаємодії (M2M). Для бездротових пристроїв, що вбудовуються, не підтримують TCP/IP-мережі спеціально розроблена окрема версія протоколу MQTT-SN (MQTT for Sensor Networks)[2].

3.1.1 Особливості протоколу MQTT

Розпочати опис протоколу MQTT варто з його основних моментів:

– функціонування за умов нестабільної роботи лінії передачі (у разі відпадання мережі). Забезпечується оперуванням невеликими повідомленнями – пересилаються компактно, що дозволяє заощаджувати буквально кожний біт. А це дуже актуально для бінарних протоколів, яким є MQTT;

– одночасна підтримка кількох рівнів якості обслуговування. MQTT працює на прикладному рівні, використовуючи для організації з'єднання та передачі TCP/IP інформації. За замовчуванням застосовується порт 1883. Якщо потрібно додатково забезпечити захист даних, використовується SSL. У цьому випадку для підключення застосовується порт 8883;

– швидкість та простота підключення нового апаратного забезпечення. MQTT адаптований до особливостей як устаткування, і каналів зв'язку.

У роботі протокол практично не навантажує обчислювальні потужності пристроїв, але при цьому коректно доставляє повідомлення до центрального блоку навіть при нестабільному інтернет-повідомленні.

У процесі взаємодії бере участь три категорії користувачів:

– видавці, це ті, хто надсилає повідомлення. Вони вказують на topic – тему. Як приклад – датчики, які знімають показання з термометрів або інших пристроїв, підключених до Інтернету;

– передплатники, кінцеві отримувачі інформації. Вони можуть працювати з різними видавцями, залежно від того, які топики вони підписані. Як приклад – аналітична хмарна система;

– брокер це основний вузол MQTT, який забезпечує стабільну передачу інформації між клієнтами: видавцями та передплатниками. Він отримує інформацію від брокера, обробляє її, передає передплатникам, контролює доставку. Роль брокера часто покладається на сервер або контролер[24].

Для взаємодії з брокером передбачено набір стандартизованих повідомлень:

– connect: встановлення доступу/з'єднання;

– disconnect: розрив з'єднання;

– publish: публікація інформації у topic;

– subscribe: підписка на topic;

– Unsubscribe: відписка від topic.

Всі ці дії виконуються із брокером.

Схема простої взаємодії між передплатником, видавцем та брокером (див.рис. 3.12).

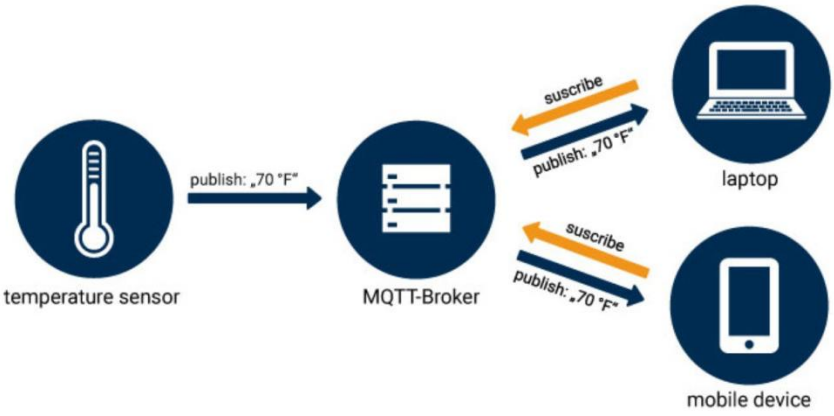

Рисунок 3.12 – Схема простої взаємодії протоколу MQTT

3.1.2 Структура повідомлень

MQTT повідомлення складається з кількох частин:

– фіксований заголовок (присутня за всіма повідомленнями);

– змінний заголовок (є лише у певних повідомленнях);

– дані, «навантаження» (є лише у певних повідомленнях).

Bit	7	6	5	4	3	2	1	0
Byte 1	Message Type				Flags specific to each MQTT packet			
Byte 2	Remaining Length							

Рисунок 3.13 – Структура повідомлень MQTT

Message Type – це тип повідомлення, наприклад: CONNECT, SUBSCRIBE, PUBLISH та інші.

Flags specific to each MQTT packet – ці 4 біти відведені під допоміжні прапори, наявність та стан яких залежить від типу повідомлення.

Remaining Length – представляє довжину поточного повідомлення (змінний заголовок + дані), може зайняти від 1 до 4 байти.

3.1.3 Основні переваги протоколу

Протокол MQTT має ряд вагомих переваг, серед яких варто виділити:

– нейтральність до вмісту повідомлень: може передаватись будь-яка інформація;

– підходить як для розподілених комунікацій, так і для роз'єднаних програм;

– наявність опції Last Will and Testament (LWT), що повідомляє про непередбачене відключення видавця або брокера;

– для базових завдань зв'язку цілком підходить TCP/IP;

– робота за стандартними шаблонами: «рівно один раз», «мінімум один раз» та «максимум один раз»;

– той самий користувач може взяти він роль і споживача, і видавця, зокрема одночасно.

Також до вагомих переваг MQTT відносять неперевершене розуміння протоколу каналів зв'язку. Вони представлені як шлях до файла. Таке рішення гарантує, що кожен споживач отримає повідомлення, які йому спрямовані. При цьому MQTT бере на себе фільтрацію повідомлень виходячи з того, на якій гілці та на якому рівні клієнти підписані на шлях до файла.

3.2 Моделювання брокера MQTT

Розглянемо випадок, коли IoT-сервер діє як MQTT-брокер. Щоб встановити MQTT Broker на сервері, на вкладці Desktop встановити додаток MQTT Broker (див. рисунок 3.14).

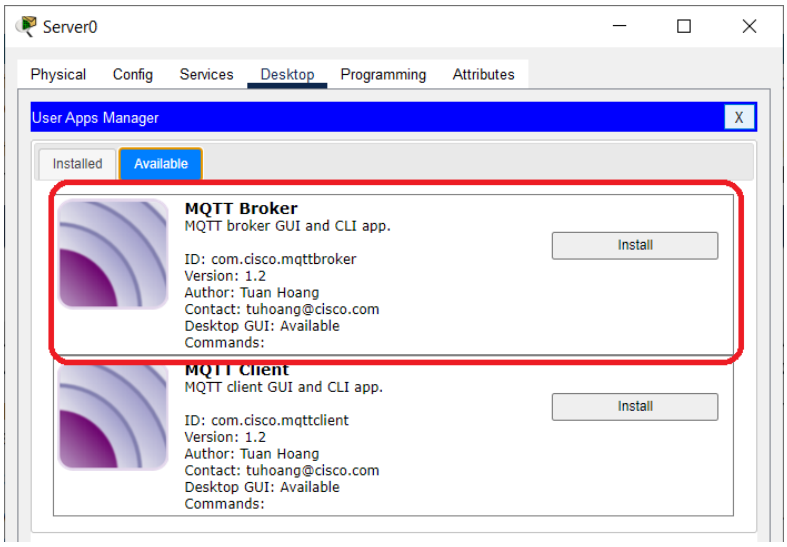

Рисунок 3.14 – Встановлення додатку MQTT Broker

Після встановлення на Desktop з’явиться ярлик брокера MQTT Broker. Після запуску додатку необхідно створити ім’я користувача та пароль (рис. 3.15).

IP-адреса брокера така ж, як і IP-адреса сервера.

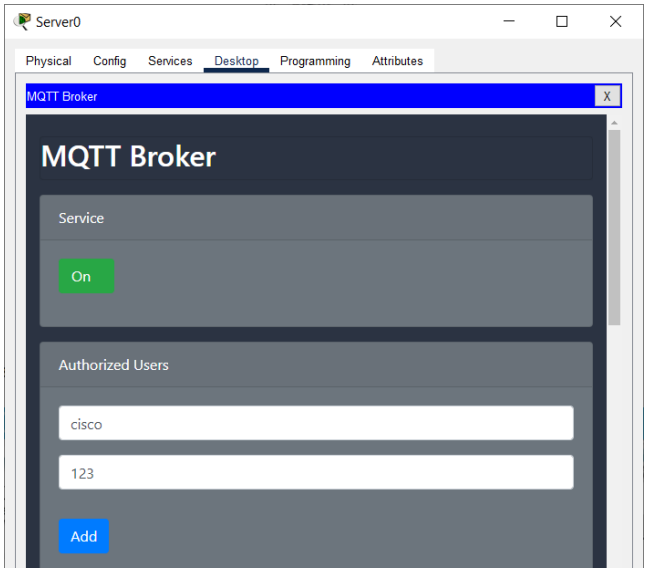

Рисунок 3.15 – Додавання користувача на MQTT Broker

3.3 Моделювання клієнта MQTT

Клієнтами MQTT можуть бути будь-які кінцеві пристрої, як от пристрої Інтернету речей, ноутбук або MCU тощо.

Контролери SBC0 виконують роль клієнта MQTT. У цьому експерименті SBC1 діє як видавець MQTT, перевіряючи рівень диму, а клієнти IoT діють як абоненти MQTT для моніторингу стану пристроїв IoT.

На вкладці Dekstop після встановлення MQTT Client необхідно вказати IP-адресу брокера, ім'я користувача та пароль (див. рисунок 3.16).

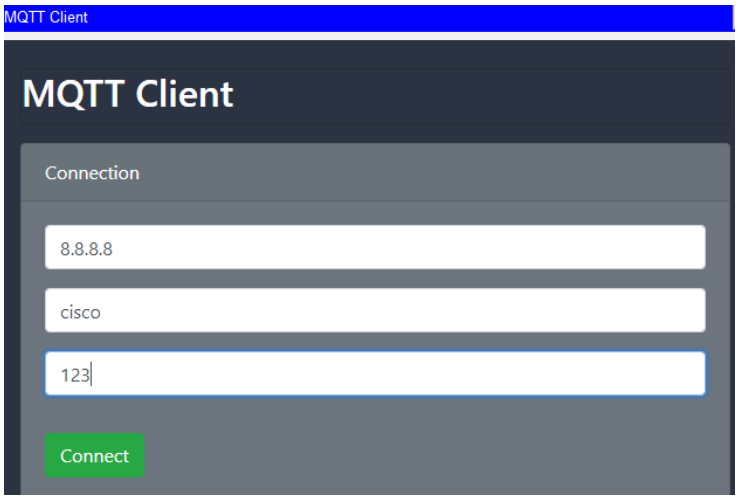

Рисунок 3.16 – Налаштування MQTT Client

Після успішного під'єднання клієнтів на MQTT-брокері відображаються відомості про них (див. рис. 3.17).

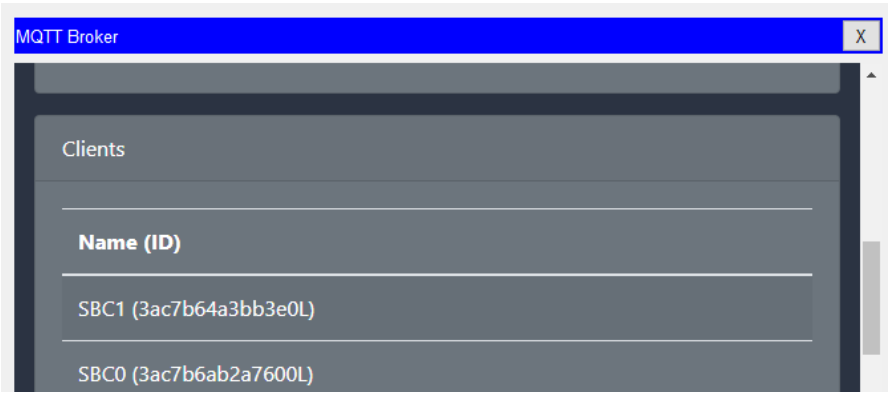

Рисунок 3.17 – Відомості про клієнтів

3.4 Моделювання роботи противопожежної системи по протоколу MQTT

У протоколі MQTT є 2 функції, підписатися і публікувати повідомлення. Підписка (Subscription) – це спосіб змусити програму «прослуховувати» повідомлення на певну тему в «Брокері» а публікація (Publish) – надсилати повідомлення на сервер, щоб інший пристрій міг «прослуховувати» ці повідомлення. Один пристрій завжди повинен бути підписаний на тему, а інший пристрій повинен публікувати повідомлення на цю тему.

В таблиці 3.1 представлені оголошені підписки.

Таблиця 3.1 – Реалізовані підписки

Topic	Payload	Дія
/GARAGE_SMOKE	open	Відкрити двері гаража
/GARAGE_SMOKE	close	Зачинии двері гаража
/GARAGE_DOOR	open	Відкрити двері гаража
/GARAGE_DOOR	close	Зачинии двері гаража
/GARAGE_SIREN	on	Включити розпилювач в гаражі
/GARAGE_SIREN	off	Виключити розпилювач в гаражі
/KITCHEN_WINDOW	open	Відкрити вікно на кухні
/KITCHEN_WINDOW	close	Зачинии вікно на кухні
/KITCHEN_SPRINKER	on	Включити розпилювач на кухні
/KITCHEN_SPRINKER	off	Виключити розпилювач на кухні
/KITCHEN_FAN	on	Включити вентилятор кухні
/KITCHEN_FAN	off	Виключити вентилятор кухні
/KITCHEN_SIREN	on	Включити сирену на кухні
/KITCHEN_SIREN	off	Виключити сирену на на кухні
/KITCHEN_FIRE	no	Включити розпилювач на кухні
/KITCHEN_FIRE	alarm	Виключити розпилювач на кухні
/KITCHEN_ALARM	off	Виключити розпилювач, вентилятор та сирену на кухні
/KITCHEN_ALARM	alarm	Включити розпилювач, вентилятор та сирену на кухні

У цьому експерименті SBC2 діє як видавець MQTT, перевіряючи стан повітря в гаражі та сигналізатор датчику вогню пристрою IoT. SBC0 діє як підписник MQTT за підписками «/GARAGE_SIREN» та «KITCHEN_SIREN», щоб отримувати повідомлення при зміни стану сирен. SBC2 підписан на «/KITCHEN_ALARM», щоб при отриманні команди alarm або off контролер ввімкнув/вимкнув сирену, розпилювач та вентилятор на кухні.

На рис. 3.18 на MQTT брокері відомості про підписки.

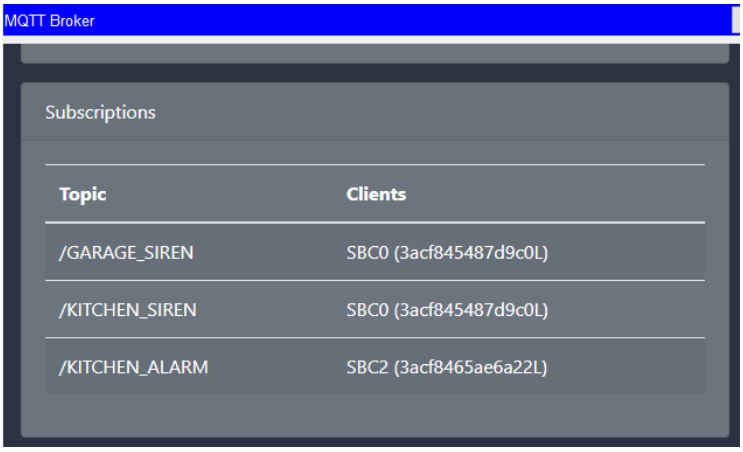

Рисунок 3.18 – Відопости про підписки на MQTT брокері

На рисунку 3.19 продемонстровано, як на SBC2 зроблена підписка на топік /KITCHEN_ALARM.

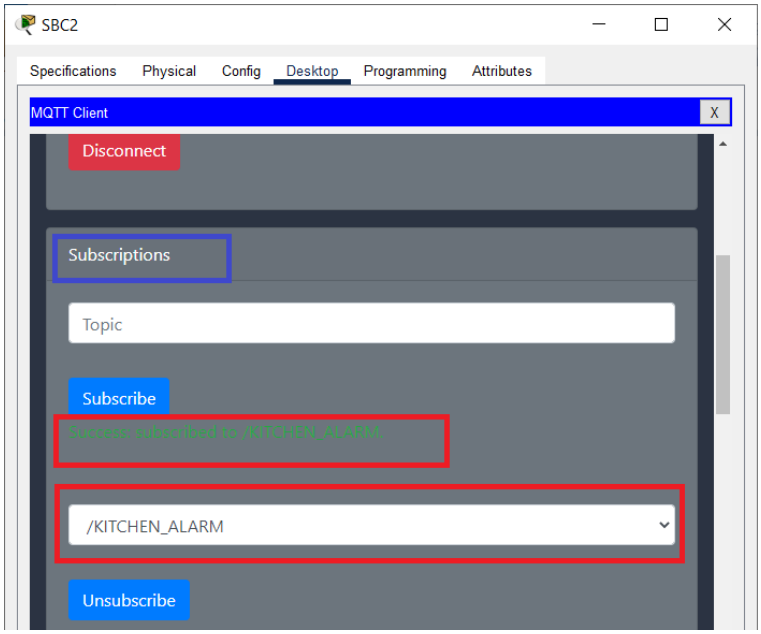

Рисунок 3.19 – Підписка на SBC2

На рисунку 3.20 показано, що на SBC0 створена публікація «/KITCHEN_ALARM» з payload alarm, щоб SBC2 контролер ввімкнув сирену, розпилювач та вентилятор на кухні. SBC2 успішно отримав це повідомлення і включає пристрої.

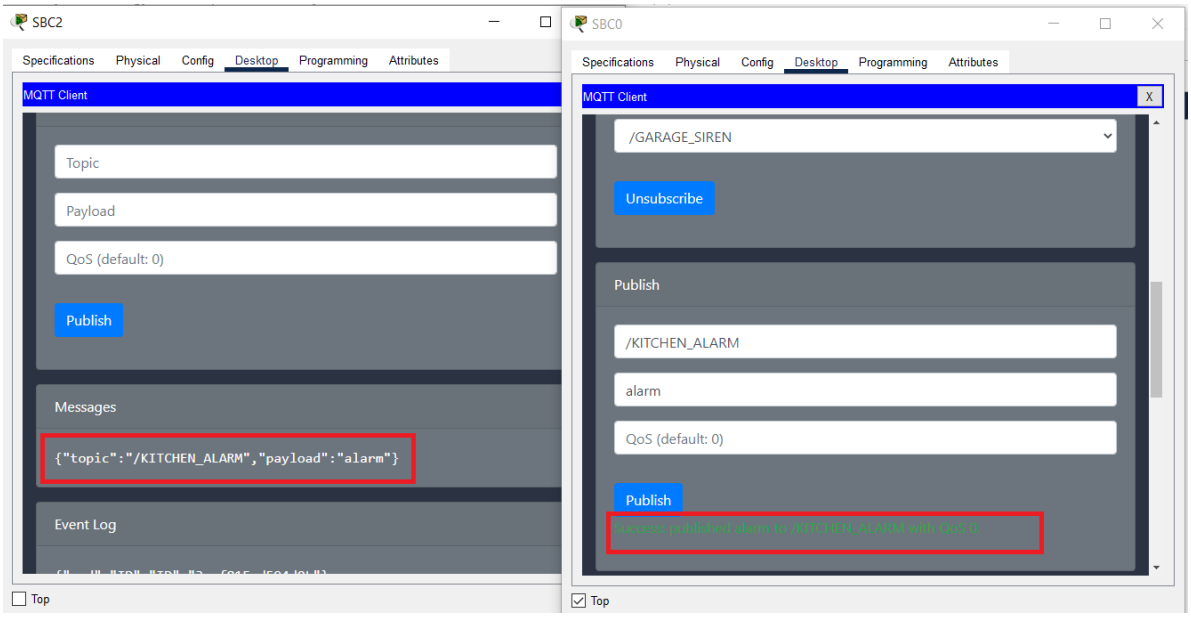

Рисунок 3.20 – Створення публикації /KITCHEN_ALARM/alarm

На рисунку 3.21 ми бачимо час (0,007) , який пройшов з моменту отримання повідомлення від брокера SBC0 до моменту включення пристроїв.

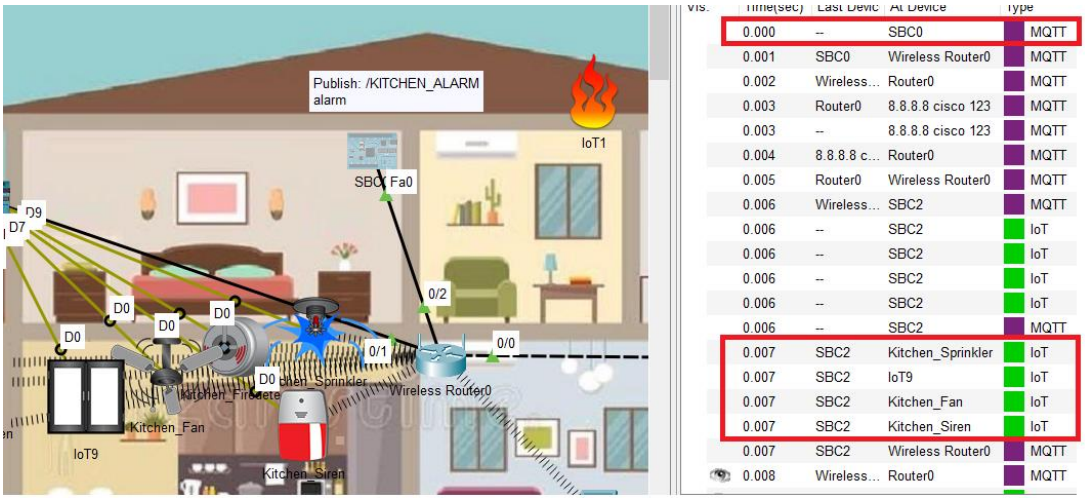

Рисунок 3.21 – Час моделювання експерименту

Після того, як включилась сирена на кухні, SBC0 отримав про це повідомлення (рис. 3.22).

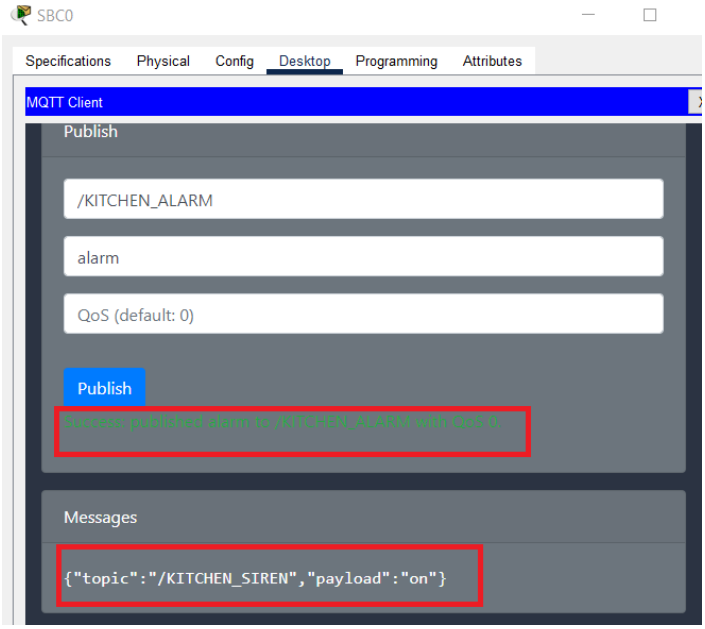

Рисунок 3.22 – Отпримання повідомлення про включення сирени на кухні

Проведенний експеримент демонструє, що час між публікацією на клієнті MQTT та передачею до контролера 0,007с, в порівнянні з часом 0,845 між web-сервером та контролером. Тож протокол MQTT краще використовувати для швидкої та своєчасної передачі даних між M2M.

4 ЗАХИСТ ІНФОРМАЦІЇ В КОМПЮТЕРНІЙ СИСТЕМІ

4.1 Загрози інформаційної безпеки

Для опису класифікації загроз інформаційної безпеки технології «розумний дім» використовуються найімовірніші загрози, вразливості, можливі наслідки та критерії аналізу загроз [10].

4.2 Проектування системи інформаційної безпеки

Проектування інформаційної системи полягає у визначенні функціональних вимог користувача системи, визначенні основних компонентів майбутньої системи та етапів її роботи, розробці алгоритмів [11]. Для розробки цих моделей використовується персональна мова моделювання UML.

Для опису функціональних вимог до системи була побудована діаграма варіантів використання.

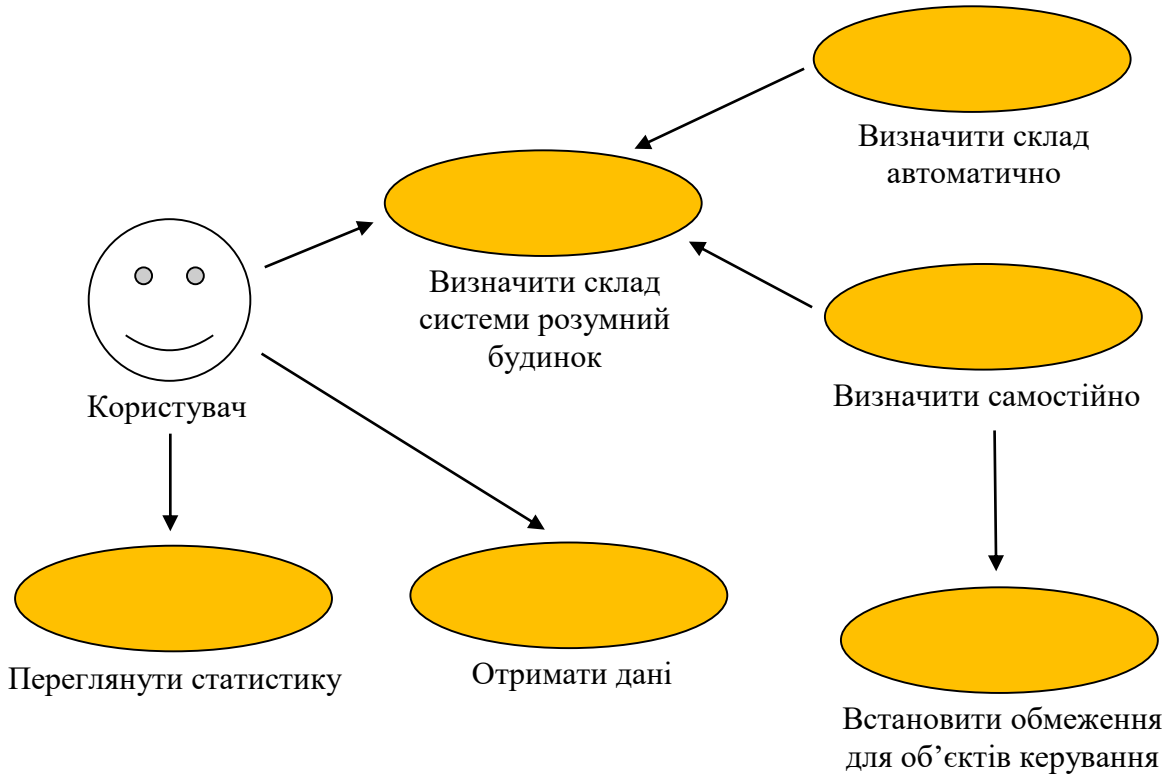

Рисунок 4.1 – Діаграма варіантів використання

Діаграма варіантів використання показує основні функціональні вимоги:

а) Визначити систему розумного будинку:

– автоматично шляхом визначення «ідеального» стану розумного будинку;

– визначити самостійно, встановлюючи при цьому обмеження об’єктів керування;

б) Отримати дані про склад розумного будинку;

в) Переглянути статистику про виявлені загрози.

На основі функціональних вимог було виділено основні етапи роботи проектованої системи:

а) Визначення складу системи розумного будинку та встановлення обмежень одним із способів:

– отримання «ідеального» стану системи розумного будинку;

– додавання користувачем вручну кожного елемента системи розумного будинку та обмежень для них.

б) Отримання даних із системи розумного будинку.

в) Перевірка всіх даних (моніторинг) в режимі реального часу.

г) Генерація сповіщень про стан системи розумного будинку.

На основі отриманих даних було визначено основні компоненти системи інформаційної безпеки та побудовано діаграму компонентів.

Рисунок 4.2 – Діаграма компонентів

Докладніше про вибрані компоненти системи:

– компонент визначення системи необхідний визначення складу системи розумного будинку з отриманих даних;

– компонент представлення даних служить відображення користувачеві отриманих системою інформаційної безпеки даних;

– компонент встановлення обмежень необхідний автоматичного визначення обмежень для показань елементів системи розумного будинку з даних;

– база даних служить для зберігання даних про склад системи розумного будинку, класифікації загроз інформаційної безпеки та даних, що використовуються системою інформаційної безпеки;

– компонент моніторингу здійснює моніторинг стану системи розумного будинку шляхом перевірки вхідних даних та визначення невідповідностей;

– компонент оповіщень служить для інформування користувача про виявлені погрози або підозрілі дії;

ВИСНОВОК

Відповідно до поставленого завдання комп’ютеризована система пожежної безпеки для розумного дому забезпечує ефективний захист і має можливість розширювати функціональні можливості. Система включає технічну, інформаційну, та інші види підтримки.

В зв’язку зі стрімким розвитком IT - індустрії, проектування мережі забезпечує максимальний захист будинку.

Можливим і найкращім рішенням є використання централізованої системи розумного будинку, яка включає сервер аналізу та обробки даних, проміжні контролери для отримання даних від сенсорів, маршрутизатор для забезпечення зв’язку із системою, в тому числі для віддаленого доступу через мережу Інтернет.

Розроблені технічні вимоги до комп’ютерної системи. Вони враховують особливості будинку та найкращій протипожежної безпеки.

Для забезпечння підвищеної надійності системи розумного дому було розроблено ряд методів безпеки, в тому числі реалізація віддаленого доступу через віртуально приватну мережу.

Результати розрахунку системи перевірені на базі розробленої моделі мережі розумного дому в Cisco Packet Tracer.

ПЕРЕЛІК ПОСИЛАНЬ

1. Kyas O. To Smart Home: A Step by Step Guide for Smart Homes & Building Automation. – New York: Key Concept Press, 2017. – 337 p.Оглад мережевого обладнання Cisco: [електронний ресурс]
2. https://dobrovdome.ru/catalog/as-2014
3. Глоба Л.С. Розподілені системи та мережі. Том 1. Підручник. – К.: Політехніка, 2013. – 378 с.
4. Лебедь О.О., Мислінчук В.О., Пастушенко В.Й. Фізичні основи комп'ютерно-інтегрованих інформаційних систем. Навчальний посібник. – Рівне : НУВГП, 2015. – 352 с.
5. Рамський Ю.С., Олексюк В.П., Балик А.В. Адміністрування комп'ютерних мереж і систем: Навч. пос. – Тернопіль: Навчальна книга – Богдан, 2015. – 196 с.
6. Розробка програмного забезпечення комп'ютерних систем. Програмування [Текст]: навч. посібник / Л.І. Цвіркун, А.А. Євстігнєєва, Я.В. Панферова. – 2-ге вид., випр. – Д.: Національний гірничий університет, 2011. – 222 с.
7. Жуков, І. А. Комп'ютерні мережі та технології : навч. посіб./І. А. Жуков, В. О. Гуменюк, І. Є. Альтман. – К. : НАУ, 2004. – 276 с.
8. Goodwin S. Smart Home Automation with Linux and Raspberry Pi. Second Edition. – Apress Media, 2013. – 328 p
9. Колонтаєвський, Ю. П. Електроніка і мікросхемотехніка [Текст]: підручник для студентів вузів, 2-е вид. / Ю. П. Колонтаєвський, А. Г. Сосков; за ред. докт. техн. наук, проф. А.Г. Соскова. – К.: Каравела, 2009. – 416 с.
10. Голев Д.В., Кононович В.Г., Хомич С.В. Методики оцінки інформаційної захищеності телекомунікацій. Навчальний посібник. – Одеса : ОНАЗ ім. О. С. Попова, 2013. – 217 с.

11. Гребенніков В.В. Комплексні системи захисту інформації: проектування, впровадження, супровід. Ужгород: Ужгородський національний університет, 2013. – 161 с.
12. https://doc.arduino.ua/ru/hardware/Mega2560
13. https://www.tp-link.com/ru/home-networking/wifi-router/archer-ax53/
14. https://york.rv.ua/2020/12/raspberry-pi-4-model-b/
15. Категорія «Розумний дім» [Електронний ресурс] / Що таке «Розумний дім»? http://www.dom-electro.ru/что-такоеумный-дом/
16. Розумний дім [Електронний ресурс] http://smarton.com.ua/.
17. http://nickshevtsov.blogspot.com/2017/10/cisco-packet-tracer.html
18. Що таке MQTT і для чого він потрібний у IIoT? [Електронний ресурс] https://ipc2u.ru/articles/prostye-resheniya/chto-takoe-mqtt/
19. MQTT: протокол передачі даних в інтернет речей [Електронний ресурс] https://www.xelent.ru/blog/mqtt-protokol-peredachi-dannykh-v-internete-veshchey
20. Цвіркун Л.І. Глобальні комп'ютерні мережі. Програмування мовою PHP: навч. посібник / Л.І. Цвіркун, Р.В. Липовий, під заг. ред. Л.І. Цвіркуна. – Д.: Національний гірничий університет, 2013. – 239 с.

ДОДАТОК А – ЛІСТИНГ SBC0

```
from cli import *
from gui import *
import json
import mqttclient
from time import *
from gpio import *
from time import *

def guiEvent(type, args):
     data = json.loads(args)

     mqttclient.init()

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

def cliEvent(type, args):
     if type == "invoked" and args[0] == "mqttclient":
          if len(args) < 2 or len(args) > 1 and args[1] == "-?"
or args[1] == "/?":
               print_cli_usage()
               CLI.exit()
          elif len(args) > 1 and args[1] != "-?" and args[1] !=
"/?":
               mqttclient.init()

               if len(args) > 2 and len(args) < 6 and args[1] ==
"connect":
                    username = ""
                    password = ""

                    if len(args) > 3:
                         username = args[3]

                         if len(args) == 5:
                              password = args[4]
```

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

                    if len(messages) > 0:
                         print messages[-1]

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

                    for message in messages:
                         print message

                    print ""
                    CLI.exit()
               elif len(args) == 2 and args[1] == "display-last-
event":
                    events = mqttclient.state()["events"]

                    if len(events) > 0:
                         print events[-1]

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

                    for event in events:
                         print event

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

def print_cli_usage():
```

```
     print "MQTT Client"
     print ""
     print "Usage:"
     print "mqttclient connect <broker address> [username]
[password]"
     print "mqttclient disconnect"
     print "mqttclient subscribe <topic>"
     print "mqttclient unsubscribe <topic>"
     print "mqttclient publish <topic> <payload> <qos>"
     print "mqttclient display-last-message"
     print "mqttclient display-all-messages"
     print "mqttclient display-last-event"
     print "mqttclient display-all-events"
     print ""

def on_connect(status, msg, packet):
     if status == "Success" or status == "Error":
          print status + ": " + msg
     elif status == "":
          print msg

     CLI.exit()

def on_disconnect(status, msg, packet):
     if status == "Success" or status == "Error":
          print status + ": " + msg
     elif status == "":
          print msg

     CLI.exit()

def on_subscribe(status, msg, packet):
     if status == "Success" or status == "Error":
          print status + ": " + msg
     elif status == "":
          print msg

     CLI.exit()

def on_unsubscribe(status, msg, packet):
     if status == "Success" or status == "Error":
          print status + ": " + msg
     elif status == "":
          print msg

     CLI.exit()

def on_publish(status, msg, packet):
     if status == "Success" or status == "Error":
          print status + ": " + msg
     elif status == "":
          print msg
```

```
     CLI.exit()

def on_message_received(status, msg, packet):
     if status == "Success" or status == "Error":
          print status + ": " + msg
     elif status == "":
          print msg

     CLI.exit()

def on_gui_update(msg, data):
     GUI.update(msg, json.dumps(data))

def autoConnect():
     brokerAdr="8.8.8.8"
     userName="cisco"
     userPass="123"
     mqttclient.init()
     mqttclient.connect(brokerAdr, userName, userPass)
     CLI.exit()


def main():
     GUI.setup()
     CLI.setup()
     mqttclient.init()
     mqttclient.onConnect(on_connect)
     mqttclient.onDisconnect(on_disconnect)
     mqttclient.onSubscribe(on_subscribe)
     mqttclient.onUnsubscribe(on_unsubscribe)
     mqttclient.onPublish(on_publish)
     mqttclient.onMessageReceived(on_message_received)
     mqttclient.onGUIUpdate(on_gui_update)
     autoConnect()
     delay(2000)
     mqttclient.subscribe("/GARAGE_SIREN");
     mqttclient.subscribe("/KITCHEN_SIREN");



     while True:
          delay(2000)

if __name__ == "__main__":
     main()
```

ДОДАТОК Б – ЛІСТИНГ SB2

```
from cli import *
from gui import *
import json
import math
import mqttclient
from time import *
from gpio import *

state_gsiren = False
state_ksiren = False

def guiEvent(type, args):
     data = json.loads(args)

     mqttclient.init()

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

def cliEvent(type, args):
     if type == "invoked" and args[0] == "mqttclient":
          if len(args) < 2 or len(args) > 1 and args[1] == "-?"
or args[1] == "/?":
               print_cli_usage()
               CLI.exit()
          elif len(args) > 1 and args[1] != "-?" and args[1] !=
"/?":
               mqttclient.init()

               if len(args) > 2 and len(args) < 6 and args[1] ==
"connect":
                    username = ""
                    password = ""

                    if len(args) > 3:
                         username = args[3]

                         if len(args) == 5:
```

```
                              password = args[4]

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

                    if len(messages) > 0:
                         print messages[-1]

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

                    for message in messages:
                         print message

                    print ""
                    CLI.exit()
               elif len(args) == 2 and args[1] == "display-last-
event":
                    events = mqttclient.state()["events"]

                    if len(events) > 0:
                         print events[-1]

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

                    for event in events:
                         print event

                    print ""
                    CLI.exit()
               else:
                    print_cli_usage()
                    CLI.exit()
          elif type == "interrupted":
               CLI.exit()
```

```
def print_cli_usage():
     print "MQTT Client"
     print ""
     print "Usage:"
     print "mqttclient connect <broker address> [username]
[password]"
     print "mqttclient disconnect"
     print "mqttclient subscribe <topic>"
     print "mqttclient unsubscribe <topic>"
     print "mqttclient publish <topic> <payload> <qos>"
     print "mqttclient display-last-message"
     print "mqttclient display-all-messages"
     print "mqttclient display-last-event"
     print "mqttclient display-all-events"
     print ""

def on_connect(status, msg, packet):
     if status == "Success" or status == "Error":
          print status + ": " + msg
     elif status == "":
          print msg

     CLI.exit()

def on_disconnect(status, msg, packet):
     if status == "Success" or status == "Error":
          print status + ": " + msg
     elif status == "":
          print msg

     CLI.exit()

def on_subscribe(status, msg, packet):
     if status == "Success" or status == "Error":
          print status + ": " + msg
     elif status == "":
          print msg

     CLI.exit()

def on_unsubscribe(status, msg, packet):
     if status == "Success" or status == "Error":
          print status + ": " + msg
     elif status == "":
          print msg

     CLI.exit()

def on_publish(status, msg, packet):
     if status == "Success" or status == "Error":
          print status + ": " + msg
     elif status == "":
```

```
          print msg

     CLI.exit()

def on_message_received(status, msg, packet):
     if status == "Success" or status == "Error":
          print status + ": " + msg
     elif status == "":
          print msg

     global state_gsiren
     global state_ksiren
     # It is verified if the program receives a message with the
topic " GARAGE_SMOKE".
     if msg.count("/GARAGE_SMOKE") == 1:
          # Check if the Payload is "on" or "off". If it is
neither of the 2 above, take no action.
          if msg.count("open") == 1:
               # If the payload is "on", the LED turns on, if it
is "off" it turns off.
               digitalWrite(0, HIGH)
          elif msg.count("close") == 1:
               digitalWrite(0, LOW)
     # The same is done with all the others. With doors and
windows, "open" and "close" are used instead of "on" and "off".
     elif msg.count("/GARAGE_DOOR") == 1:
          if msg.count("open") == 1:
               digitalWrite(1, HIGH)
          elif msg.count("close") == 1:
               digitalWrite(1, LOW)
     elif msg.count("/GARAGE_SIREN") == 1:
          if msg.count("on") == 1:
               digitalWrite(2, HIGH)
          elif msg.count("off") == 1:
               digitalWrite(2, LOW)
     elif msg.count("/KITCHEN_WINDOW") == 1:
          if msg.count("open") == 1:
               digitalWrite(7, HIGH)
          elif msg.count("close") == 1:
               digitalWrite(7, LOW)
     elif msg.count("/KITCHEN_SPRINKER") == 1:
          if msg.count("on") == 1:
               digitalWrite(6, HIGH)
          elif msg.count("off") == 1:
               digitalWrite(6, LOW)
     elif msg.count("/KITCHEN_FAN") == 1:
          if msg.count("on") == 1:
               digitalWrite(8, HIGH)
          elif msg.count("off") == 1:
               digitalWrite(8, LOW)
     elif msg.count("/KITCHEN_SIREN") == 1:
          if msg.count("on") == 1:
               digitalWrite(5, HIGH)
```

```
          elif msg.count("off") == 1:
               digitalWrite(5, LOW)
     elif msg.count("/KITCHEN_FIRE") == 1:
          if msg.count("no") == 1:
               digitalWrite(5, HIGH)
          elif msg.count("alarm") == 1:
               digitalWrite(5, LOW)
     elif msg.count("/KITCHEN_ALARM") == 1:
          if msg.count("alarm") == 1:
               customWrite(6, '1')
               digitalWrite(7, HIGH)
               customWrite(8, '2')
               customWrite(9, '1')
               if state_ksiren is False:
                    mqttclient.publish("/KITCHEN_SIREN", "on",
"0")
                    state_ksiren=True
          elif msg.count("off") == 1:
               print("!!!  reciver alarm")
               customWrite(6, '0')
               digitalWrite(7, LOW)
               customWrite(8, '0')
               customWrite(9, '0')
               if state_ksiren is True:
                    mqttclient.publish("/KITCHEN_SIREN", "off",
"0")
                    state_ksiren=False


     CLI.exit()

def on_gui_update(msg, data):
     GUI.update(msg, json.dumps(data))

def autoConnect():
     brokerAdr="8.8.8.8"
     userName="cisco"
     userPass="123"
     mqttclient.init()
     mqttclient.connect(brokerAdr, userName, userPass)
     CLI.exit()

def myPublish():
     a=read=math.floor(255. * (analogRead(0) / 1023.))
     topic="/GARAGE_DOOR"
     payload=str(a)
     qos="0"
     mqttclient.init()
     mqttclient.publish(topic, payload, qos)
     print(topic)

def loop():
     global state_gsiren
```

```
     global state_ksiren
     ## If it receives an electrical signal from the garage smoke
detector,
     #it sends a signal to the door and the siren.
     if digitalRead(0) == HIGH:
          digitalWrite(1, HIGH)
          digitalWrite(2, HIGH)
          if state_gsiren is False:
               mqttclient.publish("/GARAGE_SIREN", "on", "0")
               state_gsiren=True
     else:
          digitalWrite(1, LOW)
          digitalWrite(2, LOW)
          if state_gsiren is True:
               mqttclient.publish("/GARAGE_SIREN", "off", "0")
               state_gsiren=True

     ## If it receives an electrical signal from the kitchen fan
detector,
     #it sends a signal to the door and the siren.
     if digitalRead(5) == 1023:
          customWrite(6, '1')
          digitalWrite(7, HIGH)
          customWrite(8, '2')
          customWrite(9, '1')
          if state_ksiren is False:
               mqttclient.publish("/KITCHEN_SIREN", "on", "0")
               state_ksiren=True

     else:
          customWrite(6, '0')
          digitalWrite(7, LOW)
          customWrite(8, '0')
          customWrite(9, '0')
          if state_ksiren is True:
               mqttclient.publish("/KITCHEN_SIREN", "off", "0")
               state_ksiren=False


def main():
     GUI.setup()
     CLI.setup()
     mqttclient.init()
     mqttclient.onConnect(on_connect)
     mqttclient.onDisconnect(on_disconnect)
     mqttclient.onSubscribe(on_subscribe)
     mqttclient.onUnsubscribe(on_unsubscribe)
     mqttclient.onPublish(on_publish)
     mqttclient.onMessageReceived(on_message_received)
     mqttclient.onGUIUpdate(on_gui_update)
     delay(6000)
     autoConnect()
```

```
          # In each process 2 seconds of delay is added so that
the process is carried out well, because these methods work with
MQTT packets.
     delay(2000)
     # Subscribes to the indicated topic.
#    mqttclient.subscribe("/GARAGE_SMOKE")
#    delay(1000)
#    mqttclient.subscribe("/GARAGE_DOOR")
#    delay(1000)
#    mqttclient.subscribe("/GARAGE_SIREN")
#    delay(1000)
#    mqttclient.subscribe("/KITCHEN_WINDOW")
#    delay(1000)
#    mqttclient.subscribe("/KITCHEN_SPRINKER")
#    delay(1000)
#    mqttclient.subscribe("/KITCHEN_FAN")
#    delay(1000)
#    mqttclient.subscribe("/KITCHEN_SIREN")
#    delay(1000)
#    mqttclient.subscribe("/KITCHEN_FIRE")
#    delay(1000)
     mqttclient.subscribe("/KITCHEN_ALARM")


     while True:
     #    myPublish()
     #    loop()
          delay(2000)


if __name__ == "__main__":
     main()
```

ДОДАТОК В – ЛІСТИНГ MQTT-BROKER

```
from cli import *
from gui import *
import json
import mqttbroker
from time import *

def guiEvent(type, args):
     data = json.loads(args)

     if type == "state":
          GUI.update("state", json.dumps(mqttbroker.state()))
          mqttbroker.update_authorized_users_table()
          mqttbroker.update_clients_table()
          mqttbroker.update_subscriptions_table()
     elif type == "enable_service":
          mqttbroker.enable_service()
     elif type == "disable_service":
          mqttbroker.disable_service()
     elif type == "add_user":
          if mqttbroker.add_user(data["username"],
data["password"]):
               GUI.update("add_user_success",
json.dumps({"username": data["username"], "password":
data["password"]}))
          else:
               GUI.update("add_user_fail",
json.dumps({"username": data["username"], "password":
data["password"]}))
          mqttbroker.update_authorized_users_table()
     elif type == "remove_user":
          if mqttbroker.remove_user(data["username"]):
               GUI.update("remove_user_success",
json.dumps({"username": data["username"]}))
          else:
               GUI.update("remove_user_fail",
json.dumps({"username": data["username"]}))
          mqttbroker.update_authorized_users_table()

def cliEvent(type, args):
     if type == "invoked" and args[0] == "mqttbroker":
          if len(args) == 1 or len(args) == 2 and args[1] == "-?"
or args[1] == "/?":
               print_cli_usage()
               CLI.exit()
          elif len(args) > 1 and args[1] != "-?" and args[1] !=
"/?":
               if len(args) == 2 and args[1] == "enable-service":
                    mqttbroker.enable_service()
                    print "Success: MQTT broker service enabled."
                    print ""
                    CLI.exit()
```

```
                elif len(args) == 2 and args[1] == "disable-
service":
                    mqttbroker.disable_service()
                    print "Success: MQTT broker service
disabled."
                    print ""
                    CLI.exit()
                elif len(args) > 2 and len(args) < 5 and args[1]
== "add-user":
                    password = ""

                    if len(args) == 4:
                        password = args[3]

                    if mqttbroker.add_user(args[2], password):
                        GUI.update("add_user_success",
json.dumps({"username": args[2], "password": password}))
                        print "Success: added new user " +
args[2] + "."
                    else:
                        GUI.update("add_user_fail",
json.dumps({"username": args[2], "password": password}))
                        print "Error: could not add new user " +
args[2] + "."

                    print ""
                    CLI.exit()
                elif len(args) == 3 and args[1] == "remove-user":
                    if mqttbroker.remove_user(args[2]):
                        GUI.update("remove_user_success",
json.dumps({"username": args[2]}))
                        print "Success: removed user " + args[2]
+ "."
                    else:
                        GUI.update("remove_user_fail",
json.dumps({"username": args[2]}))
                        print "Error: could not remove user " +
args[2] + "."

                    print ""
                    CLI.exit()
                elif len(args) == 2 and args[1] == "display-last-
event":
                    events = mqttbroker.state()["events"]

                    if len(events) > 0:
                        print events[-1]

                    print ""
                    CLI.exit()
                elif len(args) == 2 and args[1] == "display-all-
events":
                    events = mqttbroker.state()["events"]
```

```
                    for event in events:
                         print event

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

def print_cli_usage():
     print "MQTT Broker"
     print ""
     print "Usage:"
     print "mqttbroker enable-service"
     print "mqttbroker disable-service"
     print "mqttbroker add-user <username> [password]"
     print "mqttbroker remove-user <username>"
     print "mqttbroker display-last-event"
     print "mqttbroker display-all-events"
     print ""

def on_gui_update(msg, data):
     GUI.update(msg, data)


def main():
     GUI.setup()
     CLI.setup()
     mqttbroker.init()
     mqttbroker.add_user("cisco", "123")
     mqttbroker.onGUIUpdate(on_gui_update)

     while True:
          delay(2000)

if __name__ == "__main__":
     main()
```