

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ  
ІМЕНІ СЕМЕНА КУЗНЕЦЯ**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

Рівень вищої освіти	Перший (бакалаврський)
Спеціальність	Кібербезпека
Освітня програма	Кібербезпека
Група	6.04.125.010.18.1

## **ДИПЛОМНА РОБОТА**

на тему: «Оцінка кібербезпеки в кіберфізичних системах та шляхи  
кіберзахисту»

Виконав: студент Євгеній ПРОСВЕТОВ

Керівник: старший викладач Олена МЕРЛАК

Рецензент: начальник обчислювального центру  
Харківського національного університету  
будівництва та архітектури,  
к.т.н., доцент  
Максим СВИНАРЕНКО

Харків – 2022 рік

## РЕФЕРАТ

Кваліфікаційна бакалаврська робота містить: 79 стор., 20 рис., 2 таблиці, 35 літературних джерел.

Метою бакалаврської роботи є оцінювання кібербезпеки у кіберфізичних системах та шляхи кіберзахисту.

Об'єкт дослідження – кіберфізична система.

Предмет дослідження – методи та процеси оцінювання кібербезпеки, технології розробки програмних засобів.

Методи дослідження – аналіз загроз, теорія захисту інформації, теорії ймовірностей і математичної статистики, експертного оцінювання.

У результаті дослідження були проаналізовані кіберфізичні системи на прикладі систем розумного будинку, зроблено оцінку інформаційної безпеки та було розроблено додаток оцінки загроз цієї системи.

Для розроблення веб-додатка були використані програмні продукти: Visual Studio 2017, Microsoft SQL server 2012, Internet Information Services 7, SQL Management Studio 17.

ІНФОРМАЦІЙНА БЕЗПЕКА, КІБЕРБЕЗПЕКА, РОЗУМНИЙ БУДИНОК,  
КІБЕРФІЗИЧНІ СИСТЕМИ, АНАЛІЗ, ОЦІНКА ЗАГРОЗ, КЛАСИФІКАЦІЯ.

## ABSTRACT

An explanatory message is to bakalavr's work: 79 pages, 20 figures, 2 tables, 35 references.

The aim of the bachelor's thesis is to assess cybersecurity in cyberphysical systems and ways of cybersecurity.

The object of research is the cyberphysical system.

Subject of research – methods and processes of cybersecurity assessment, software development technologies.

Research methods – threat analysis, information security theory, probability theory and mathematical statistics, expert evaluation.

As a result of the research, cyberphysical systems were analyzed on the example of smart home systems, information security was assessed and an application for threat assessment of this system was developed.

Software products were used to develop the web application: Visual Studio 2017, Microsoft SQL server 2012, Internet Information Services 7, SQL Management Studio 17.

INFORMATION SECURITY, CYBER SECURITY, SMART HOUSE, CYBERPHYSICAL SYSTEMS, ANALYSIS, THREAT ASSESSMENT, CLASSIFICATION.

## ЗМІСТ

Перелік умовних позначень та скорочень .....	8
Вступ.....	9
1. Призначення та область використання кіберфізичних систем.....	10
1.1 Призначення кіберфізичних систем .....	10
1.2 Область застосування кіберфізичних систем.....	11
2. Розгляд аналогічних існуючих систем.....	14
2.1 Загальний огляд на кіберфізичні системи .....	14
2.2 Аналіз та оцінка загроз .....	19
2.3 Система «Розумний будинок» .....	25
2.4 Розгорнута постановка задачі .....	31
3. Реалізація системи інформаційної безпеки .....	33
3.1. Опис функціонування системи .....	33
3.2 Алгоритмічне забезпечення .....	34
3.3 Розробка системи інформаційної безпеки .....	42
4. Основні висновки .....	50
Список джерел інформації .....	52
Додаток А Лістинг програми .....	56

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ**

ІБ – інформаційна безпека;

КФС – кіберфізичні системи;

РБ – розумний будинок;

Actuator – виконавчий механізм;

Average – обмеження;

Causes – можливі причини, Cause – можлива причина.

Component – компонент підсистеми;

Condition – умова;

Consequences – можливі наслідки; Consequence – можливий наслідок;

Min – мінімальне значення; Max – максимальне значення;

Name – назва;

Object – об'єкт керування;

Sensor – сенсор, датчик;

Sources – джерела, Source – джерело;

Subsystem – підсистема;

System – система РБ;

Threats – загрози, Threat – загроза;

Type – тип елемента;

## ВСТУП

Інтернет, соціальні мережі, хмарні служби та електронна комерція стали важливими складовими життя сучасної людини. Вартість тарифів інтернету стрімко знижується, технології щодня починають дедалі більше використовуватися в повсякденному житті, що призвело до виникнення складних систем, в яких поєднуються програмне забезпечення та фізична складова. Завдяки такому поєднанню, виникають важливі системи, які називають кіберфізичними. Із появою КФС розвиток зробив швидкий стрибок у майбутнє, бо саме завдяки цим системам можна забезпечити оптимізацію багатьох процесів, необхідних для стабільної економіки. Кіберфізичні системи можуть бути як автономними, так і не автономними, вони здатні приймати рішення та працювати самостійно. Однак на даний момент часу розвиток кіберфізичних систем відбувається переважно в напіваавтономних системах. Це системи, які працюють незалежно лише в заздалегідь визначених умовах, наприклад, напіваавтономні дрони. Користувачі можуть встановлювати траєкторію польоту, а потім машинне бачення в режимі реального часу дозволить дрону уникати перешкод, усуваючи потребу в ручному польоті. Важливою задачею є дослідження принципів роботи кіберфізичних систем, проведення аналізу загроз, які можуть виникнути при їх створенні, забезпечення безпеки цих систем, та дослідження класифікації КФС.

# 1. ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ КІБЕРФІЗИЧНИХ СИСТЕМ

## 1.1 Призначення кіберфізичних систем

Кіберфізична система (КФС) – це нове покоління цифрових систем, що складається з обчислювальних і фізичних можливостей, які взаємодіють з людьми, як ніколи раніше. Вона розроблена так, щоб діяти як мережа з кількох змінних як із фізичним введенням, так і з виводом, а не як самостійна технологія. Цей тип концепції тісно пов'язаний із сенсорними мережами, такими як робототехніка, які функціонують відповідно до обчислювального інтелекту. Ця здатність взаємодіяти та спілкуватися, розширюючи можливості фізичного світу за допомогою обчислень, є важливим кроком для майбутніх технологій [1].

Вбудовані комп'ютери та мережі контролюють і керують фізичними процесами з петлями зворотного зв'язку, де фізичні процеси впливають на обчислення, і навпаки. Економічний і соціальний потенціал таких систем набагато більший, ніж те, що було реалізовано, і великі інвестиції здійснюються в усьому світі для розвитку технології. Технологія базується на старій (але ще дуже молодій) дисципліні вбудованих систем, комп'ютерів і програмного забезпечення, вбудованого в пристрої, основна місія яких не є обчисленнями, наприклад, автомобілі, іграшки, медичні пристрої та наукові інструменти.

КФС мають надзвичайно важливе значення і використовуються фундаментально (урядами/підприємствами) для вирішення багатьох проблем тестування поточного повсякденного життя, оскільки усвідомлено, що швидка та правильна динаміка в системі великих даних є основним завданням. Незалежно від переваг (і вдосконалення) КФС є багато проблем, з якими стикаємося в нашому житті, наприклад, відсутність надточних

діагностичних апаратів, які впливають на системи охорони здоров'я. Крім того, вони не можуть мати справу з величезною кількістю інформації (яка створюється пристроєм, підключеним до Інтернету). У зв'язку з тим, що людей, які мають навички, недостатньо, щоб мати справу з цими гаджетами чи фреймворками, відстежувати чи досліджувати їх стає набагато важче. Аналогічно, багато вразливостей можуть виникнути на цих IoT/веб-асоційованих речах. Згодом узгодження машинного навчання в КФС є важливою необхідністю сьогодення. КФС в майбутньому може бути корисним у створенні кваліфікованих та дружніх організацій за допомогою оптимальних рішень.

Наведемо технічні передумови створення КФС.

1. Збільшення кількості пристроїв із вбудованими процесорами та запам'ятовуючих пристроїв: сенсорні мережі, медична техніка, розумні будинки тощо.

2. Інтеграція, яка досягає найбільшого ефекту шляхом з'єднання окремих компонентів у великі системи: Інтернет речей (IoT), World Wide Sensor Net, Smart Building Environments, оборонні системи майбутнього.

3. Обмеження когнітивних здібностей людини, які розвиваються повільніше, ніж машини. Люди не можуть впоратися з кількістю інформації, необхідної для прийняття рішень і делегування деяких функцій КФС шляхом обміну ними поза контуру управління.

Кіберфізичні системи можуть покращити аналітичні здібності людини. Існує потреба в нових інтерактивних системах, які тримають людину в контурі керування (human in the loop).

## 1.2 Область застосування кіберфізичних систем

Зростаюче використання цих розумних кіберфізичних систем має на меті посилити впровадження великомасштабних систем шляхом оптимізації



функціональності, автономності, надійності, безпеки та зручності використання цих мереж.

Наведемо декілька прикладів використання кіберфізичних систем.

Виробництво – ця галузь найчастіше асоціюється з кіберфізичними системами. Ці системи можна використовувати у виробничій промисловості для оптимізації процесів шляхом автоматизації всього виробничого процесу, створення єдиної децентралізованої платформи для цілих фабрик. Автоматизація виробництва заощаджує витрати на оплату праці та матеріалів і скорочує час виробництва.

Охорона здоров'я та медичні пристрої – у цьому секторі кіберфізичні системи можна використовувати для відстеження стану та фізичного стану пацієнтів дистанційно та в режимі реального часу. Крім того, і, що важливо, ненав'язливо. Крім того, кіберфізичні системи можна використовувати, щоб допомогти пацієнтам зі старінням на місці, тобто застосовувати розумні датчики в «розумних будинках» для виявлення нещасних випадків і негайного сповіщення системи.

Кіберфізичні системи готові трансформувати надання медичної допомоги, забезпечуючи розумне лікування та послуги. Датчики в домі виявлять зміну стану здоров'я; нові операційні системи зроблять персоналізовані медичні пристрої сумісними; а роботизована хірургія та біонічні кінцівки допоможуть вилікувати й відновити рух поранених та інвалідів і одного дня навіть підвищать людські здібності.

Сільське господарство – у сільськогосподарській галузі кіберфізичні системи можуть допомогти усунути використання пестицидів, вибираючи та використовуючи пестицид лише тоді, коли він дійсно необхідний. Цей метод не тільки ефективний, але і екологічно чистий. Крім того, ці системи можуть дати можливість управлінню сільським господарством точно вивчати, збирати та аналізувати різну інформацію про клімат, ґрунт, воду тощо.

Довкілля та його стійкість – кіберфізичні системи все частіше використовуються для забезпечення стійкості. Кіберфізичні системи допомагають пожежникам виявляти та стримувати пожежі, покращуючи сільськогосподарські методи та дають змогу вченим пом'якшувати підводні розливи нафти. Досягнення кібер-фізичних систем забезпечать можливість, адаптивність, масштабованість, стійкість, безпеку, безпеку та зручність використання, які значно перевищать прості вбудовані системи сучасності

Транспорт і енергетика – у майбутньому ми будемо подорожувати в автомобілях без водіїв, які безпечно спілкуватимуться один з одним на розумних дорогах і в літаках, які координують роботу, щоб зменшити затримки. Дрони перевіряють інфраструктуру на предмет пошкоджень і забезпечать Wi-Fi доступ до зон лиха. Будинки та офіси будуть живитися від інтелектуальної мережі, яка буде обізнана для користувачів і використовуватиме датчики для аналізу навколишнього середовища та оптимізації освітлення, опалення та охолодження [2].

## 2. РОЗГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

### 2.1 Загальний огляд на кіберфізичні системи

Не зважаючи на те, що кіберфізичні системи вже застосовуються в багатьох галузях, вони все ще знаходяться на дуже ранніх стадіях. Більшість цих кіберфізичних вимагають високої продуктивності, а також надійності, безпеки та захисту. Це причина, чому кіберфізичні системи зараз стикаються з великими проблемами, зокрема з конфіденційністю та сумісністю, оскільки системи піддаються ризику атак. Застосовувати кіберфізичні системи на даний момент необхідно з обережністю.

Для більшого розуміння роботи кіберфізичних систем розглянемо архітектурну схему 5C (Connection, Conversion, Cyber, Cognition, Configure) для побудови кіберфізичних систем. Кіберфізична система складається з 5 рівнів, а саме: комунікаційний рівень, конверсія, кібернетичний рівень, рівень пізнання та конфігурація. На рисунку 2.1 зображена архітектура 5C [3].



Рисунок 2.1 – Архітектурна схема кіберфізичних систем

Комунікаційна середа. Підключення машин та їх компонентів для отримання точних і надійних даних є першим кроком у розробці КФС для розумних підприємств. Різні пристрої або датчики використовуються для отримання різноманітних даних, включаючи напругу, струм, температуру, вібрацію, швидкість обертання, швидкість подачі та концентрацію масла машин та їх компонентів, а також зображень і відео заготовок. Деякі датчики навіть встановлені для визначення температури, вологості, освітленості, атмосферного тиску виробничого поля та складу. Для здійснення передачі даних використовуються специфічні протоколи, наприклад ті, що використовуються в технології Інтернету речей (IoT).

Конверсія. Дані на цьому рівні перетворюються в інформацію. Для реалізації перетворення даних в інформацію можна використовувати декілька механізмів. Деякі механізми розроблені для прогнозування та управління станом машини. Цей рівень привносить в машини властивість самосвідомості.

Кібернетичний рівень. Кібер-рівень діє як центральний інформаційний центр у цій архітектурі. Інформація надходить до нього з кожної підключеної машини для формування мережі машин. Зібравши велику кількість інформації, необхідно використовувати спеціальну аналітику, щоб отримати додаткову інформацію, яка дасть краще уявлення про стан окремих машин у парку. Ця аналітика надає машинам можливість самопорівняння, де продуктивність однієї машини можна порівняти та оцінити серед парку. З іншого боку, схожість між продуктивністю машини та попередніми активами можна виміряти, щоб передбачити майбутню поведінку механізму.

Пізнання. На цьому рівні користувачам для прийняття рішень надається належне представлення аналітичної інформації. Пріоритет завдань для процесу обслуговування можна легко визначити завдяки наявності порівняльної інформації та стану окремої машини.

Конфігурація. Рівень конфігурації дає зворотний зв'язок від кібер-частини назад до фізичної частини. Цей рівень виконує наглядний контроль для того,

щоб машини самоналагодилися та адаптувалися. Він діє як система керування стійкістю (RCS), щоб застосувати до машин засоби керування, що відповідають рішенням, прийнятим на рівні пізнання.

До недоліків цих систем можна віднести багато факторів. Особливо слід відзначити режим роботи без участі спеціалістів, який у поєднанні з гнучкістю системи, що забезпечується автоматичними оновленнями програмного забезпечення та розгортанням проміжного програмного забезпечення, може призвести до критичних ситуацій в ефективності та результативності систем у цілому. Наприклад, метрологічний збій, спричинений порушенням систем охорони здоров'я (автоматичне введення ліків із капсули, поміщеної під шкіру пацієнта), може призвести до необоротних наслідків.

Надійна система КФС повинна досягати високого рівня довіри. Надійність даних є основною проблемою в програмах КФС, частково через наступні ускладнення [4].

Величезний розмір. Типовий КФС має сотні датчиків, кожен з яких генерує дані кожні кілька хвилин. Система управління повинна обробляти величезні обсяги даних і високоефективно оцінювати сигнали тривоги.

Зашумлені дані. Багато реалізацій розгортань показали, що ненадійні та шумні дані є серйозною проблемою для програм КФС. Помилки можуть виникати за найнесподіваніших обставин, і менше 49% даних може бути використано для змістовної інтерпретації.

Відсутність навчальних комплектів. Багато традиційних методів виявлення хибних результатів покладаються на навчальні набори даних. Такі пристрої важко отримати в КФС, оскільки вони дорогі та схильні до помилок під час ручного маркування великого набору даних, створеного датчиками.

Конфлікти сенсорів. Добре реалізований КФС має зайву інформацію, наприклад, стандарт, який називається k-покриттям, вимагає, щоб кожна область контролювалася принаймні, k різноманітними сенсорами. Іноді один датчик не працює належним чином, а інші можуть надати точну інформацію. У

таких випадках виникають конфлікти між надійними і несправними датчиками. Оскільки користувач наперед не знає, якому датчику довіряють, система повинна отримати правдиву інформацію з суперечливих даних.

Невизначеність об'єктів. Система повинна інтегрувати дані з кількох сенсорів, щоб оцінити детальну інформацію про об'єкт. Через апаратні обмеження датчики не можуть надати детальну інформацію про зловмисників. Більшість з них можуть оцінити лише можливу площу об'єктів.

Розглянемо важливість використання програмного забезпечення у захисті та функціонуванні кіберфізичних систем.

Проміжне ПЗ є необхідною умовою ефективної роботи КФС [5]. Воно надає послуги для програм, відмінних від тих, що доступні в операційній системі, об'єднує програмні компоненти та корпоративні програми. Проміжне програмне забезпечення входять веб-сервери, сервери додатків, системи керування контентом та аналогічні інструменти, які підтримують розробку та доставку додатків, а також забезпечують зв'язок та управління даними в розподілених додатках.

Це стосується, зокрема, інформаційних технологій, заснованих на розширеній мові розмітки (RMP); на протоколі обміну структурованими повідомленнями у розподілених комп'ютерних системах при об'єктному доступі (SOAP); у інтернет-сервісах; на модульному підході (SOA) для розробки програмного забезпечення, заснованому на використанні розподілених слабозалежних заміщуючих компонентів зі стандартизованими інтерфейсами, що забезпечують взаємодію зі стандартизованими протоколами; в інфраструктурі Web 2.0 та Lightweight Directory Access Protocol (LDAP).

Послуги, які можна вважати проміжним програмним забезпеченням, включають інтеграцію корпоративних програм, інтеграцію керованими даними ППЗ (MOM), брокерів об'єктних запитів (ORB) та корпоративні сервісні шини (ESB). Наведемо інші приклади проміжного програмного забезпечення.

1. Програмне забезпечення Mer, яке не має ані ядра Linux, ані інтерфейсу. Mer більше стосується операційних систем, орієнтованих на мобільні пристрої, та на постачальників обладнання.

2. Операційна система Android, яка використовує ядро Linux, та надає прикладну систему додатків, яку розробники впроваджують у свої програми. Крім того, Android удосконалює рівень ППЗ, включаючи бібліотеки, що надають такі послуги, як зберігання, відображення на екрані, мультимедіа та веб-браузер. Оскільки бібліотеки розробляються машинною мовою, команди виконуються швидко. Бібліотеки проміжного програмного забезпечення також реалізують функції для конкретних пристроїв, тому програма не повинна залежати від різних пристроїв Android. Середній рівень Android також включає віртуальну машину Dalvik та її основні бібліотеки для програм Java.

3. У технології моделювання ППЗ використовується в контексті високорівневої архітектури, яка має застосування в багатьох розподілених моделюваннях. Це програмний рівень, вбудований між програмним кодом та інфраструктурою під час виконання. ППЗ складається з бібліотеки функцій і дозволяє запускати кілька програм моделювання (об'єднання), переміщуючи ці функції із загальної бібліотеки, замість того, щоб створювати їх заново для кожної програми.

4. Розробники бездротових пристроїв використовують ППЗ для усунення несправностей у сенсорних бездротових мережах. Реалізація ППЗ дозволяє розробникам цих мереж інтегрувати операційні системи та обладнання з різними програмами.

5. Universal Home API або UHAPI – це інтерфейс програмування побутової електроніки, розроблений форумом UHAPI. Мета UHAPI – забезпечити функціонування стандартних ППЗ на платформах потокової аудіо- та відео-передачі через незалежний від апаратного забезпечення стандартний API [6].

6. Програмні пакети RFID (програмне забезпечення для ідентифікації радіочастот) забезпечують ППЗ для фільтрації зашумлених та надмірно поступаючих даних.

7. ILAND – це програмне забезпечення для керування в режимі реального часу, яке забезпечує підтримку детермінованої реконфігурації протягом обмеженого часу.

Безперебійна та надійна робота кіберфізичних систем заснована на правильній організації вхідних інформаційних потоків завдяки надійній роботі підсистем, у тому числі контролю температури, а також надійному програмному забезпеченню. Наявність різнотипних сенсорів для кіберфізичних систем вимагає не стільки унікальних вихідних сигналів різних типів датчиків, скільки в обробці більшої кількості даних, які найчастіше виявляються надлишковими.

## 2.2 Аналіз та оцінка загроз

Аналіз і оцінка ризиків інформаційної безпеки – обов'язковий етап як в рамках побудови комплексних систем захисту інформації, так і в рамках побудови систем управління інформаційної безпеки для банківської сфери та для підприємства іншого виду діяльності.

Слід зазначити, що на сьогоднішній день існують декілька світових і вітчизняних, в тому числі галузевих, методик та методологій оцінки ризиків інформаційної безпеки. Можна виділити: FactorAnalysisofInformationRisk (FAIR), IS RISK ASSESSMENT, MEASUREMENT, NIST, Technology Systems, Operationally Critical Threat, Asset, and Vulnerability Evaluation, (OCTAVE).

Як загальний підхід для вирішення проблеми оцінки ризику визначимо наступний порядок [7].

1. Розробити загальний підхід, який дає змогу проводити практичну оцінку загрози інформаційній безпеці.



2. На основі підходу розробити методологію оцінки ризиків інформаційної безпеки (включаючи загальну термінологію, що використовується при оцінці ризику).

3. Складіть список загроз, вразливостей і список пар, які створювалися на їх основі.

4. Оцініть ризик/уразливість для всіх ідентифікованих пар, їх ймовірність та вплив на конфіденційність, цілісність, доступність та моніторинг систем.

5. Надайте інструкції щодо подальших дій для підтримки актуальності загроз, вразливостей їх пар і оцінок.

6. Автоматизувати розроблений підхід і методологію.

Безпека КФС піднімає низку нових проблем. Проблеми безпеки КФС включають [8]:

- моделювання загроз безпеки;
- розробка формального підходу до оцінки вразливостей КФС;
- проектування надійних та відмовостійких архітектур для обробки кіберфізичних загроз, що швидко розвиваються. Таким чином, нові методології та технології повинні бути розроблені для задоволення вимог.

Кіберзагрози в КФС призводять до проблем безпеки, викликаних інтеграцією обчислювальних процесів, комунікацій і мереж з управлінням фізичними системами [9]. Компоненти КФС, включаючи датчики, виконавчі механізми, розподілені центри управління, а також дротові та бездротові комунікаційні мережі, повинні бути ефективно інтегровані для забезпечення ефективного моніторингу та контролю фізичних систем [10].

Поточний стан КФС можна описати, фіксуючи значення важливих змінних процесу.

Два типи важливих процесів або змінних стану в КФС:

- 1) вимірювані змінні, що представляють дані, отримані від датчиків;
- 2) керуючі змінні, що представляють керуючі сигнали [11]. Нормальне значення параметра процесу називається заданим значенням. У КФС відстань

між значеннями змінної процесу та відповідними заданими значеннями розраховується контролером. Після обчислення цього зміщення контролери використовують складний набір рівнянь для створення стратегії активації та обчислення нових змінних активації або керування [12].

Кіберфізичні загрози – це загрози, які виникають у кібербезпеці, але впливають на фізичний простір системи. Однією з головних особливостей кіберзагроз є їх масштабованість, що означає, що їх можна легко автоматизувати та тиражувати [13]. Кіберзагрози впливають на:

- 1) конфіденційність, необхідну для збереження безпеки персональної інформації користувачів у Кодексі;
- 2) цілісність, коли дані або ресурси можуть бути змінені без дозволу;
- 3) доступність у разі відмови комп'ютерної техніки, адміністрування, зв'язку та обладнання;
- 4) надійність, коли необхідно підтвердити, що обидві сторони дійсно є тими, за кого себе видають.

Класифікація загроз КФС включає:

- 1) підробку ідентифікаційної інформації;
- 2) редагування даних (маніпулювання даними);
- 3) заперечення авторства (заперечення походження);
- 4) розкриття інформації;
- 5) збільшення повноважень;
- 6) відмова в обслуговуванні (DoS).

Атаки на кіберфізичні системи схематично представлено на рисунку 2.2.

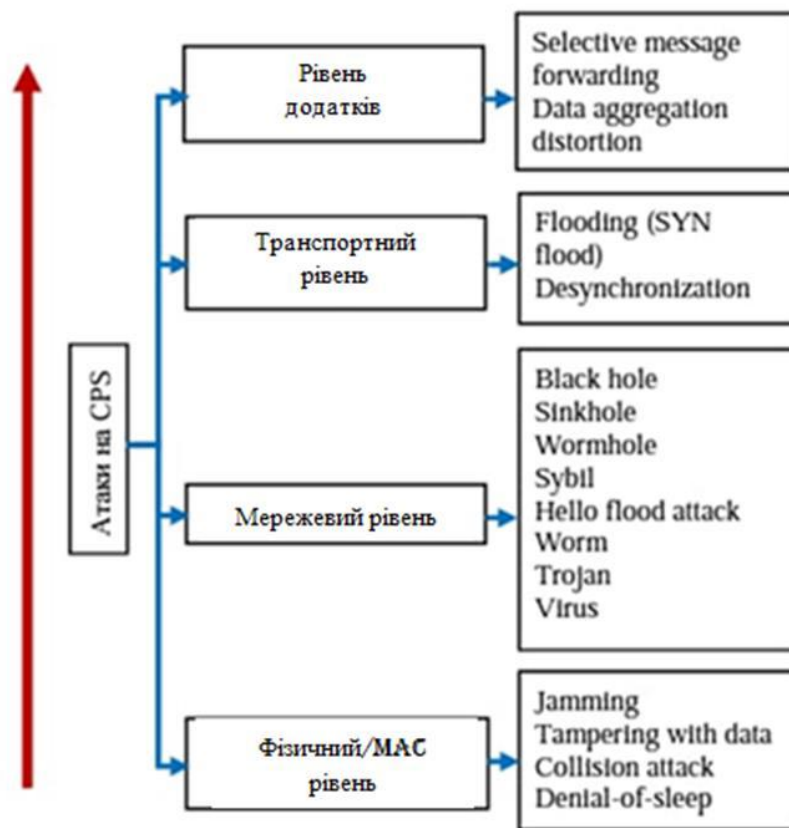


Рисунок 2.2 – Атаки на безпеку кіберфізичних систем

Алгоритми оцінки та контролю, що використовуються в КФС, розроблені для досягнення конкретних операційних цілей, таких як стабільність замкнутого циклу, безпека, стійкість або оптимізація продуктивності. Безпека – це захист цих цілей від зловмисників, які ставлять на мету нанести втрат кіберінфраструктурі. Наприклад, якщо вимірювання, зібрані мережею датчиків, містять конфіденційну інформацію, слід забезпечити доступ до цих даних лише уповноваженим особам. Підходи КФС до інформаційної безпеки можна розділити на проактивні та реактивні механізми [14].

Проактивні механізми. Аутентифікація є важливим інструментом для захисту розподілених систем. Контроль доступу запобігає несанкціонованому доступу до вашої системи: він забороняє доступ неавтентифікованим, одночасно накладаючи необхідні обмеження на діяльність автентифікованих користувачів. Безпечне спілкування між двома «справжніми» користувачами

забезпечується кодами аутентифікації повідомлень або цифровими підписами (вони можуть виявити, що повідомлення були підроблені третьою стороною). Час передачі повідомлень також може бути гарантовано за допомогою відміток часу (які вимагають безпечних протоколів синхронізації часу) або механізмів відповіді. Крім того, інструменти перевірки та програмного захисту можуть перевірити правильність архітектури та реалізації системи, зменшуючи вразливості. Безпека КФС також залежить від безпеки сенсорної мережі [15]. Більшість заходів із забезпечення безпеки сенсорної мережі включають наступні ефективні алгоритми:

- 1) отримання асоціації безпеки та керування ключем завантаження для створення надійної інфраструктури [16];
- 2) безпечний зв'язок [17, 18];
- 3) безпечні протоколи маршрутизації.

Існує кілька правил проектування безпечних систем управління [19]: надмірність, різноманітність і принцип поділу прав.

Реактивні механізми. Реактивні механізми – це заходи безпеки, які вживаються, коли атака вже сталася. Помилкових тривог і пропущених виявлень не уникнути, оскільки виявлення «згубної» логіки є непереборною проблемою. Доступні рішення безпеки не можуть впоратися з новими сценаріями атак, тому необхідні активні заходи для захисту вбудованих хостів [20].

Слід зазначити, що для чіткого спланування безпеки кіберфізичних систем, слід визначити бізнес-стратегію організації. Наступним кроком буде визначення відповідних технологічних факторів та екологічних тенденцій, пов'язаних безпосередньо з організацією. Такі кроки забезпечуть можливість порівняти всі оцінки з прогнозом кіберфізичних ризиків, що зробить ІТ-інфраструктуру більш безпечною. Не слід ігнорувати можливість різних кіберзагроз у середовищі, де немає таких і подібних спланованих рішень. Наприклад, за даними, зібраними Gartner за останні роки з Університету Темпл

у місті Філадельфія, штат Пенсільванія, США, спостерігалось значне збільшення атак програм-вимагачів на критичні інфраструктури [21]. Частота таких атак становила 2 у 2013 році і зросла до 297 у 2020 році (до вересня). У 2019 році цей показник становив 204. Тобто значне зростання можна спостерігати навіть при порівнянні 2020 року з 2019 роком.

Найважливішим фактором, що впливає на функціонування та контроль інформаційної безпеки, є IoT та системи кібербезпеки (43%). У зв'язку з цим, можна буде спостерігати збільшення цієї залежності, внаслідок поширення мереж 5G та IoT.

З іншого боку, це демонструє важливість включення кіберфізичних систем у мережу безпеки IT-інфраструктури за допомогою керування привілейованим доступом. Це пояснюється тим, що атаки програм-вимагачів можуть завдати серйозної шкоди бізнес-потоків, безпосередньо загрожуючи фізичним структурам організації. Наприклад, діяльність підприємства з постачання природного газу в Сполучених Штатах, була зупинена на два дні в результаті навмисного відключення управління атакою програм-вимагачів.

Аналогічно, логістична компанія Toll Group, що базується в Австралії, що працює в більш ніж 1200 місцях у 50 країнах з 40 000 співробітників, була змушена на деякий час припинити свою діяльність через атаку програм-вимагачів, яка спричинила несподівані затримки в доставці її клієнтам [22]. Крім того, 14 травня 2020 року австралійський виробник сталі BlueScope Steel Limited зазнав атаки програмного забезпечення-вимагача, і всі його операції були перервані по всій країні [23].

Крім того, у звіті, опублікованому компанією з кібербезпеки Draktrace, зазначено, що кіберзловмисники намагалися здійснити атаки програм-вимагачів за допомогою акваріума з підключенням до Інтернету [24]. У компанії заявили, що хакери намагалися вкрати дані з казино в Північній Америці за допомогою датчиків, підключених до комп'ютера, який регулює температуру, очищення та програму годування акваріума. Загальним аспектом

усіх цих випадків є конфіскація привілейованого облікового запису або інформація про привілейований доступ до облікового запису або те, що вони були частиною ланцюга постачання.

Переглянемо декілька засобів, які допоможуть у забезпеченні безпеки кіберфізичних систем.

Диспетчер сеансів: дозволяє контролювати всі сеанси ІТ-інфраструктури. Таким чином, стає легше запобігти потенційним плутанинам щодо керування доступом.

Центральне керування паролями: дозволяє перевіряти всі привілейовані сеанси у мережі, пропонує повністю зашифровану інфраструктуру та зберігає паролі привілейованих облікових записів у сховищах, ізольованих від системи [25].

Двофакторна автентифікація: запитує одночасну автентифікацію за місцем і часом від користувачів, які прагнуть увійти в систему, і запобігає неавторизованому та неавтентифікованому доступу.

Динамічне маскуванню даних: записує та маскує всі дії системних адміністраторів у мережі та запобігає будь-яким підозрам щодо дій.

Автоматизація привілейованих завдань: підвищує ефективність і усуває переривання обслуговування за рахунок автоматизації рутинних завдань.

Управління доступом TACACS+/RADIUS: пропонує комплексну автентифікацію та розширює можливості багаторазового входу та конфігурації політики кібербезпеки.

### 2.3 Система «Розумний будинок»

На прикладі розумного будинку розглянемо існуючі ризики безпеки у ньому, як у кіберфізичній системі.

Система «Розумний будинок» – це апаратно-програмний комплекс, який являє собою сукупність підключених в спільну мережу пристроїв, виконуючих

визначенні конкретні дії з мінімальною участю людини. Ця система оброблює великий потік інформації, і, таким чином, «Розумний будинок» піддається загрозам інформаційної безпеки, які залежать від структури самої системи розумного будинку, технологій та обробленої інформації [26, 27].

Технологія РБ використовується для різних цілей, можна виділити основні з них [28]:

- тепло- та енергозбереження;
- підвищення комфорту;
- забезпечення безпеки.

У зв'язку із складністю самої системи, її опис має різноманітні варіації, але найчастіше використовують розділ системи на підсистеми (рис. 2.3).



Рисунок 2.3 – Підсистеми «Розумного будинку»

Відсутність єдиної методології побудови захисту інформаційної безпеки технології розумного будинку пояснює різноманітність структур та методів роботи засобів захисту. Розробку даних засобів можна умовно поділити на такі етапи:

- опис моделі системи розумного будинку;
- опис моделі загроз інформаційної безпеки;
- розробка методів оцінки загроз;
- розробка автоматичного механізму моніторингу стану захисту системи.

Наведемо орієнтовний список найбільш імовірних загроз [29]:

- хакерські атаки на центральний сервер;
- вплив вірусів і троянських коней на функціонування системи;
- перехоплення інформації, що передається по дротових та бездротових каналах зв'язку;
- зловмисник з правами адміністратора отримує доступ до центрального сервера шляхом крадіжки паролів та інших даних контролю доступу;
- доступ до мережі неавторизованих користувачів;
- наявність порушників серед обслуговуючого персоналу;
- помилки користувача;
- крадіжка (зловмисне завантаження) апаратний збій;
- переривання мережі;
- природні катаклізми;
- апаратний збій системи;
- програмні помилки;
- витік інформації шляхом помилкового електромагнітного випромінювання та перешкод;
- витік інформації через акустико-електричний канал.

Існують багато засобів забезпечення інформаційного захисту. Один з найпоширеніших методів роботи на ринку пристроїв додаткового інформаційного захисту РБ – підключення до роутера та моніторинг потоку інформації між підключеними до Wi-Fi пристроями.

Три яскраві приклади подібних пристроїв:

- Dojo, розроблене невеликою ізраїльською компанією Dojo-Labs [30, 31];
- CUJO, розроблене групою каліфорнійських дослідників [32];
- Bitdefender Vox, яке виробляється великою компанією Bitdefender [33].

Існують також і більш функціональні засоби захисту інформаційної безпеки, які можуть контролювати всю мережу УД і не збирають додаткові дані. Приклад подібних пристроїв – серія пристроїв Cisco ASA 5500-X з функціями



FirePOWER [34], вироблена однією з найбільших ІТ-компаній Cisco.

Для опису класифікації загроз інформаційної безпеки в технології розумного будинку використано найбільш ймовірні загрози, вразливості, можливі наслідки та критерії аналізу загроз [35]. Для аналізу загроз було визначено три критерії: джерело загрози, вразливість інформаційної безпеки та підсистема розумного будинку, з якої походить загроза. Критерії джерела загроз інформаційної безпеки та вразливості поділяються на категорії та підкатегорії. Наведемо категорії, підкатегорії та приклади [36].

1. Джерело загрози:

а) антропогенні ресурси:

– зовнішні (І.А.1 – І.А.6): злочинні структури, потенційні злочинці, хакери та інші;

– внутрішні (І.В.1 – І.В.4): первинний персонал, допоміжний персонал та інші;

б) штучні ресурси:

– зовнішні (ІІ.А.1 – ІІ.А.3): засоби масової інформації, інженерні комунікаційні мережі та інші;

– внутрішні (ІІ.В.1 – ІІ.В.4): неякісні технічні засоби обробки інформації, неякісне програмне забезпечення обробки інформації та інші;

с) природні ресурси (ІІІ.А.1 – ІІІ.А.7): пожежі, землетруси, повені, урагани та інші.

2. Уразливість в інформаційній безпеці:

а) об'єктивні (А.І – А.ІV): викиди, пов'язані з технічними заходами, що визначаються характеристиками елементів об'єкта, що охороняється, та інші;

б) суб'єктивні (В.І, В.ІІ): помилки та порушення;

с) випадкові (С.І, С.ІІ): збої, пошкодження та відмови.

Згідно цим характеризуючим даним була складена таблиця класифікації загроз (табл. 2.1).

Таблиця 2.1 – Класифікація загроз кіберфізичних систем

Тип атаки	Джерело загрози	Вразливість	Уразливість інформаційної безпеки	Можливі наслідки	Підсистема
Хакерські атаки на центральний сервер	I.A.1, I.A.2, I.A.3, I.A.4, I.A.5, I.A.6, I.V.1, I.V.2, I.V.3, I.V.4	Підключення мережі до інтернету. Відсутність механізмів захисту периметру мережі.	A.I.a, A.I.b, A.II.a, A.II.b, A.III.b, A.IV.b, V.I.a, V.I.b, V.I.c, V.II.a, V.II.c, V.II.d, C.I.a, C.I.c	Порушення роботи, або вихід із ладу центрального сервера, і всієї системи. Порушення конфіденційності, цілісності та доступності інформації	Керування та зв'язки
Перехоплення інформації, що передається по дротових і бездротових каналах зв'язку	I.A.1, I.A.2, I.A.3, I.A.4, I.A.5, I.A.6, I.V.1, I.V.2, I.V.3, I.V.4, II.A.1, II.V.1, II.V.2	Можливість доступу зловмисникам до дротових каналів, або в зону стійкого перехоплення радіосигналів мережі. Відсутність або неефективність механізмів захисту трафіку	A.I.a, A.I.b, A.II.a, A.II.b, A.III.a, A.III.b, A.IV.a, A.IV.b, V.I.a, V.I.b, V.I.c, V.II.a, V.II.b, V.II.c, V.II.d, C.I.a, C.I.b, C.I.c, C.I.d, C.II.b	Порушення конфіденційності інформації, що передається по каналу. Можливий доступ до керування системою.	Управління та зв'язок; безпека та моніторинг; електроживлення; освітлення; опалення; вентиляції; кондиціонування; мультимедіа;
Доступ зловмисників з правами адміністратора на центральний сервер за допомогою крадіжки паролей та інших реквізитів розмежування доступу	I.A.1, I.A.2, I.A.3, I.A.4, I.A.5, I.A.6, I.V.1, I.V.2, I.V.3, I.V.4	Неефективність механізмів аутентифікації чи повна ідентифікація, або їх відсутність	A.I.a, A.I.b, A.II.a, A.II.b, A.III.a, A.III.b, A.IV.a, A.IV.b, V.I.a, V.I.b, V.I.c, V.II.a, V.II.b, V.II.c, V.II.d, C.I.a, C.I.b, C.I.c	Порушення конфіденційної інформації, що всередині мережі.	Керування та зв'язки

Описана у проєкті ІТ-система моніторингу та оцінки загроз інформаційної безпеки технології «Розумний будинок» базується на моделі засобів інформаційної безпеки, виявлених у процесі аналізу. Модель запропонованої системи:

- опис моделі системи розумного будинку;
- опис моделі ризику інформаційної безпеки;
- розробка методів оцінки ризиків;
- розробка автоматичного механізму контролю стану безпеки розумного будинку.

Для проєктування системи інформаційної безпеки використовуються основні методи системного аналізу та технології розробки програмного забезпечення.

На основі аналітичного вивчення теми було обрано метод опису системи «Розумний дім» шляхом поділу системи на підсистеми.

У системі «Розумний будинок» виділимо такі підсистеми:

- підсистема управління та зв'язку;
- підсистема безпеки та спостереження;
- підсистема живлення;
- підсистема освітлення;
- підсистема опалення;
- вентиляційна підсистема;
- підсистема кондиціонування повітря;
- мультимедійна підсистема.

Також пропонується поділ підсистем на компоненти, об'єкти управління та елементи: датчик і виконавчий механізм. Компоненти відповідають за роботу з об'єктами керування, а об'єкт керування повинен вказувати на тип підсистеми. Усі об'єкти управління в розумному будинку мають сенсор для зчитування необхідних даних і виконавчий механізм, який здійснює операцію.

Перелік загроз визначається для різних об'єктів управління залежно від підсистем, в яких вони знаходяться, і ступеня відхилення від даних датчика або

виконавчого механізму. Метод моніторингу стану розумного будинку базується на таких кроках:

- опис системи розумного будинку користувача;
- отримання даних від системи розумного будинку;
- аналіз даних, порівняння з визначеними граничними значеннями;
- визначення підсистеми та обладнання, де виникла небезпека;
- оцінка ризиків за встановленим переліком загроз.

Для опису моделі системи розумного будинку та загроз у запропонованій системі було обрано мову XML, а для опису структури цих файлів було обрано XMLSchema. Вибір мов опису даних обумовлений великими обсягами даних. Формат XML ефективний для роботи з саме з таким обсягом. XML – це розширювана мова розмітки, яка дозволяє створювати будь-яку розмітку, необхідну для конкретної програми. Структуру файлу XML можна описати за допомогою XML Schema, яка визначає правила документа. XML також підтримується на низькому рівні апаратного, мікропрограмного та програмного забезпечення в сучасному обладнанні.

#### 2.4 Розгорнута постановка задачі

З метою чіткої постановки задачі, було розроблено схему (рис. 2.4), в якій представлено основні функціональні вимоги.



Рисунок 2.4 – Основні функціональні вимоги системи «Розумний будинок»

До основних функцій проєктованої системи належить.

1. Визначення системи розумного будинку:

- а) автоматично визначити «ідеальний» стан розумного будинку;
- б) прийняти рішення про встановлення обмежень на елементи контролю.

2. Отримання даних про склад розумного будинку.

3. Перегляд статистики виявлених загроз.

Виходячи з функціональних вимог, були визначені основні етапи роботи запропонованої системи.

1. Визначення складу системи розумного будинку та встановлення обмеження одним із наступних способів:

- а) отримання «ідеального» стану системи РБ,
- б) ручне додавання кожного елемента системи РБ користувачем та їх обмеження.

2. Отримання даних із системи РБ.

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

4. Створення сповіщень про стан системи РБ.

### 3. РЕАЛІЗАЦІЯ СИСТЕМИ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

#### 3.1 Опис функціонування системи

Наведемо структурну схему системи інформаційної безпеки (рис. 3.1).

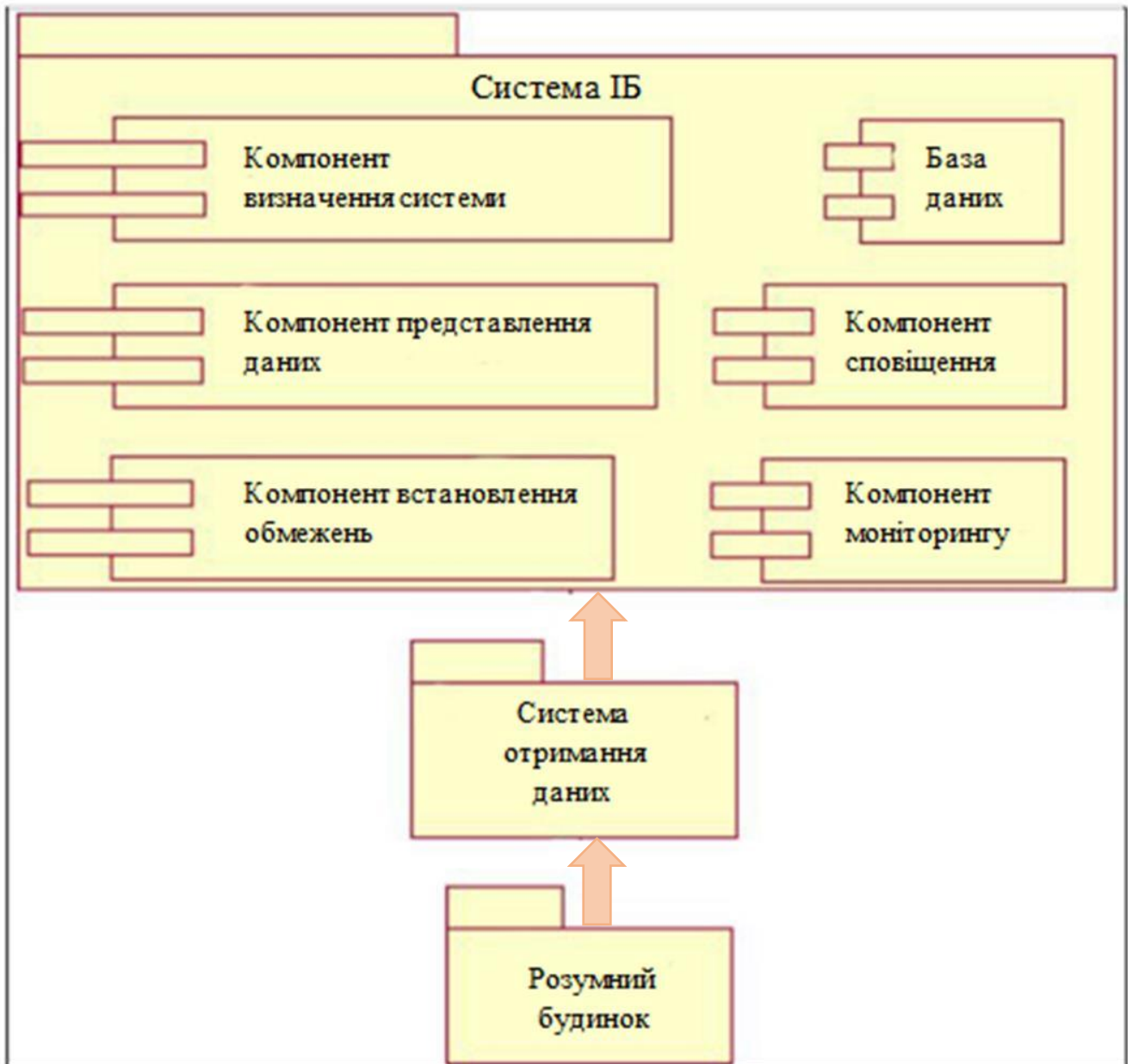


Рисунок 3.1 – Структурна схема інформаційної безпеки розумного будинку

Розглянемо детальніше вибрані елементи системи:

– компонент визначення системи необхідний для визначення складу системи DM на основі отриманих даних;

- компонент представлення даних використовується для відображення даних, отриманих системою інформаційної безпеки користувачу;
- компонент встановлення обмежень необхідний для автоматичного визначення обмежень для зчитування елементів системи РБ з отриманих даних;
- база даних використовується для зберігання даних про склад системи РБ, класифікації загроз інформаційної безпеки та даних, яку вона використовує;
- компонент моніторингу контролює стан системи РБ шляхом перевірки вхідних даних та виявлення розбіжностей;
- компонент сповіщення служить для інформування користувача про виявлені загрози або підозрілі дії.

Цей алгоритм виконується системою ІБ щодо складу системи РБ і встановлення обмежень шляхом додавання користувачем вручну кожного елемента системи РБ.

### 3.2 Алгоритмічне забезпечення

Для розроблення системи оцінки загроз у кіберфізичній системі було розроблено діаграми діяльності щодо визначення складу системи та алгоритму моніторингу даних.

Для відображення алгоритму визначення користувачем складу системи РБ було побудовано діаграму діяльності (рис. 3.2).

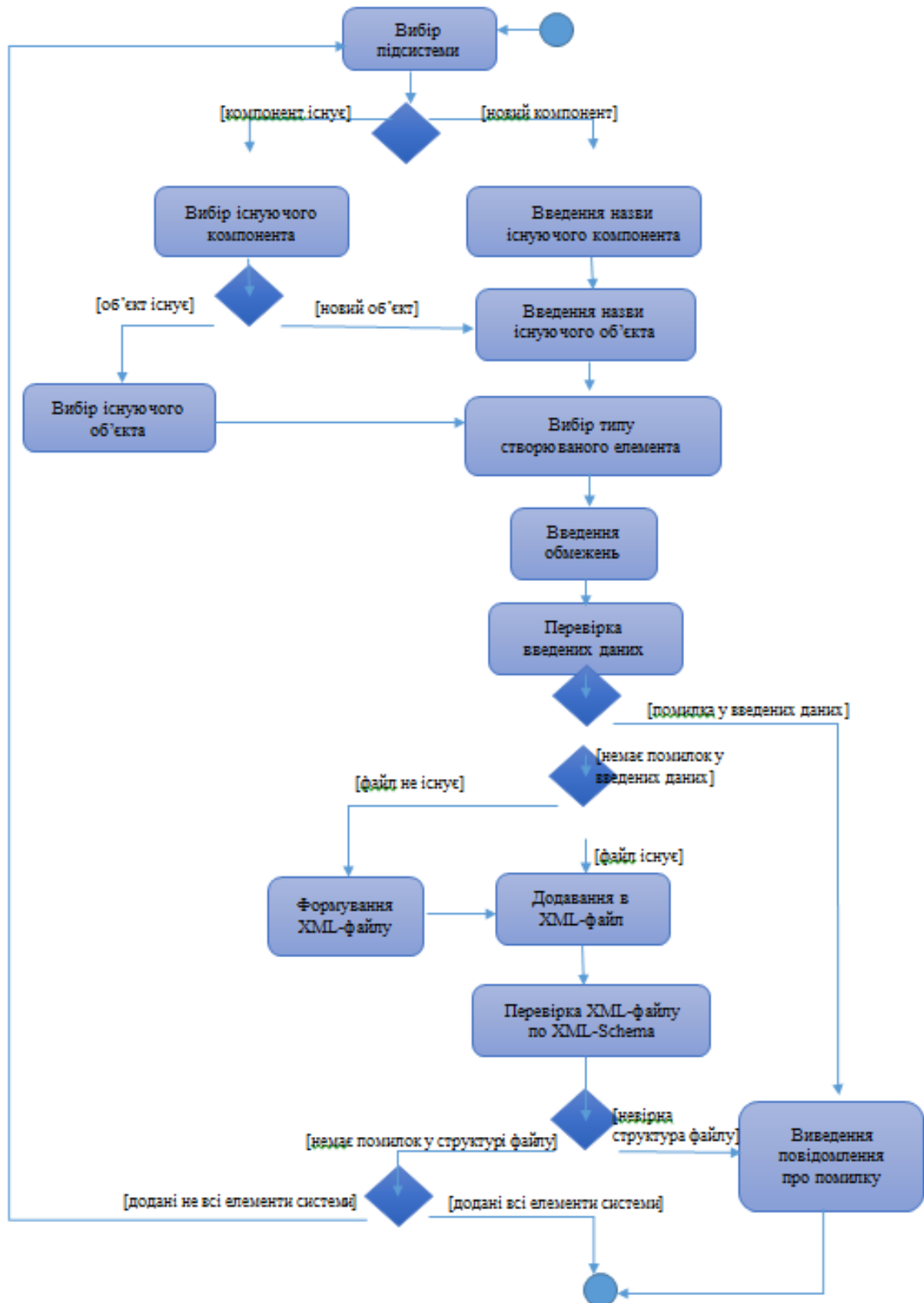


Рисунок 3.2 – Алгоритм визначення користувачем складу системи РБ



Для автоматичного визначення складу системи РБ було розроблено алгоритм, який виконується системою інформаційної безпеки, що входить до складу розумного будинку, і встановлює обмеження шляхом отримання «ідеального» стану системи. Для відображення алгоритму було побудовано діаграму діяльності (рис. 3.3).

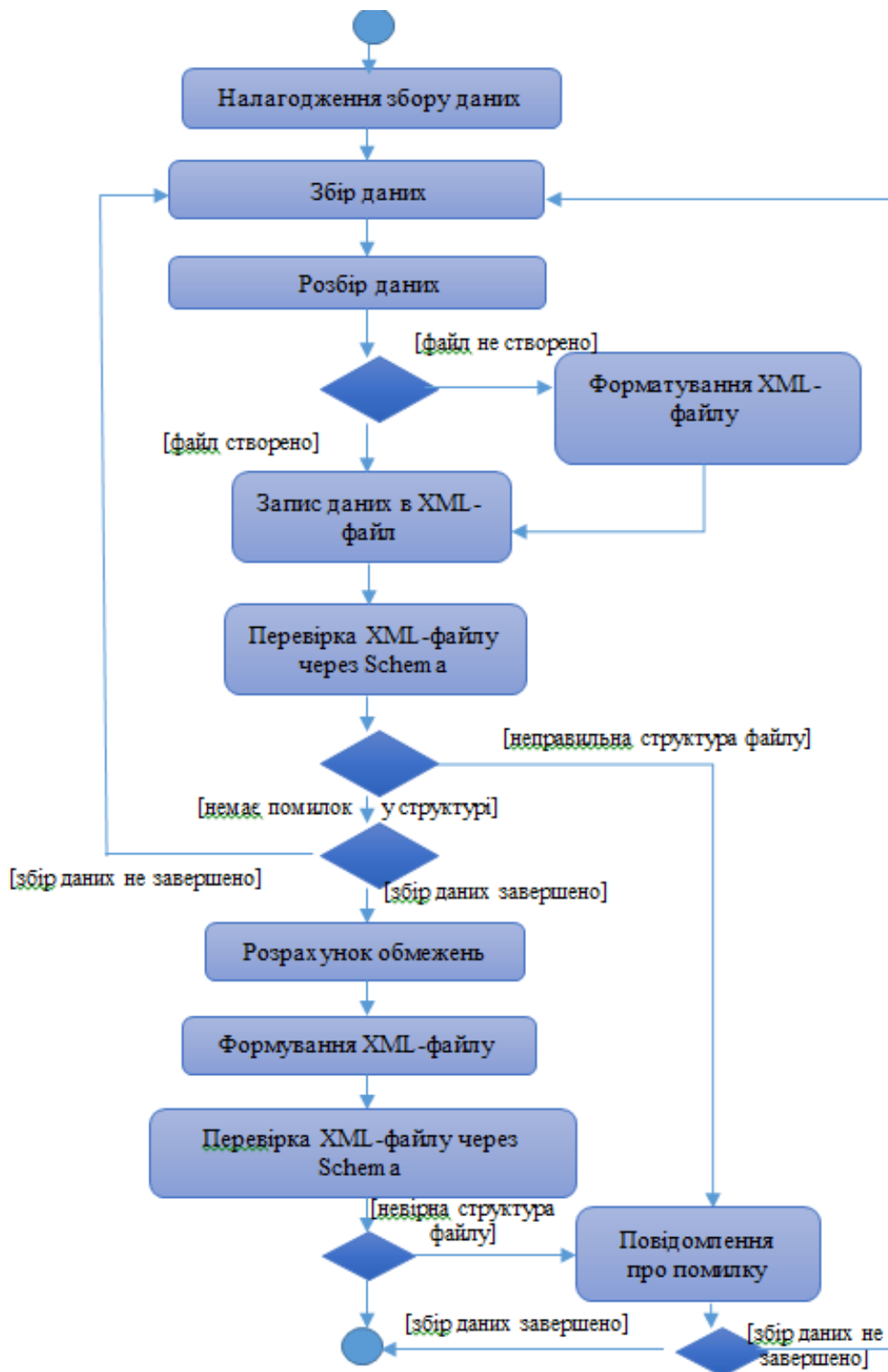


Рисунок 3.3 – Алгоритм автоматичного визначення складу системи РБ

Алгоритм моніторингу стану системи РБ полягає у перевірці всіх отриманих даних із системи РБ у режимі реального часу. Для відображення алгоритму було побудовано діаграму діяльності (рис. 3.4).

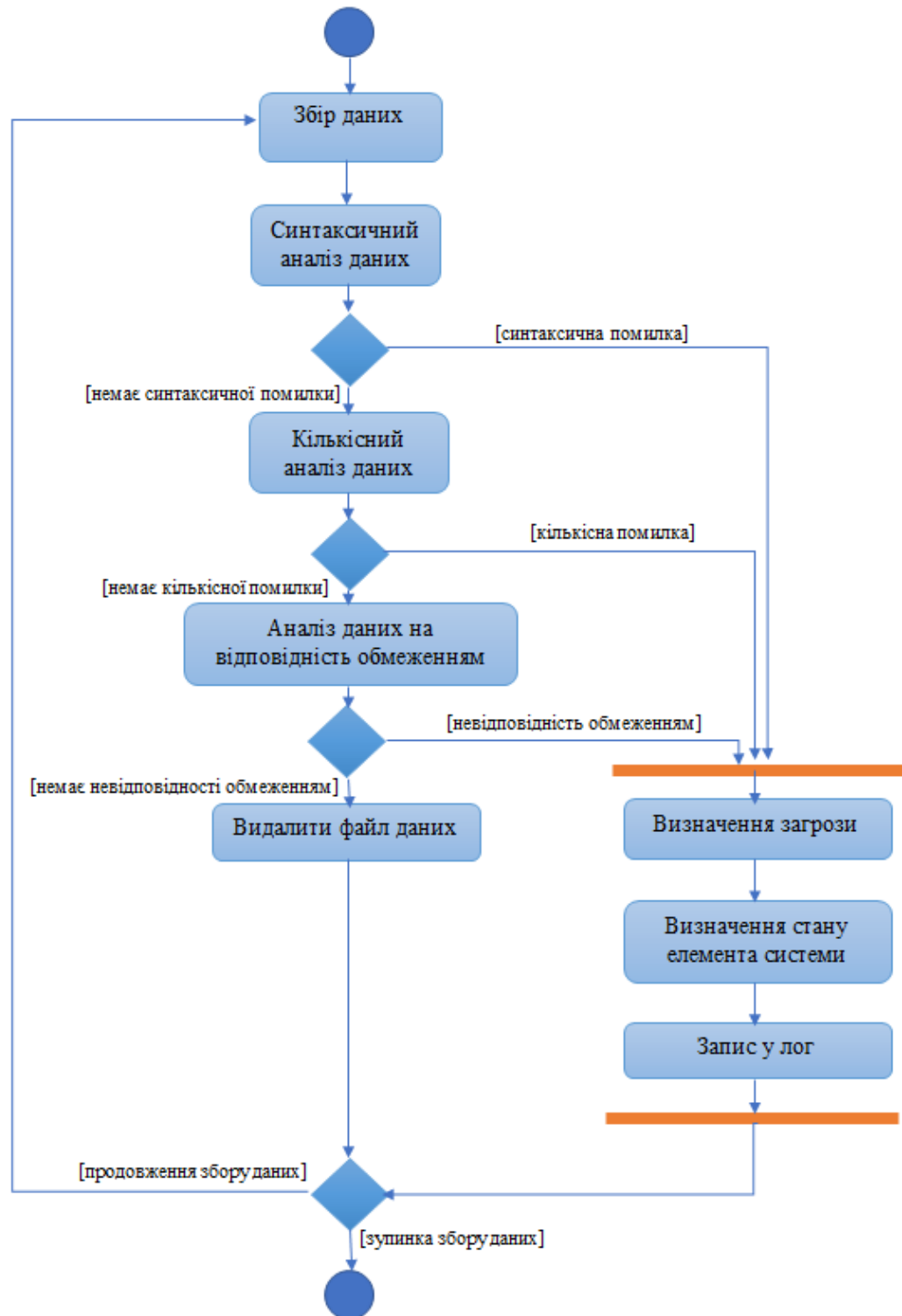


Рисунок 3.4 – Алгоритм моніторингу стану системи РБ

У проєктованій системі інформаційної безпеки для опису даних про склад системи «Розумний будинок» та для опису загроз інформаційної безпеки було обрано формат XML. Для опису структури файлів XML була використана мова XML Schema.

При визначенні складу системи РБ користувачем самостійно шляхом визначення кожного елемента системи формується XML-файл опису системи РБ. Графічне представлення схеми опису складу системи РБ зображено на рисунку 3.5.

При автоматичному способі визначення складу системи РБ спочатку формується файл з «ідеальним» станом системи, потім розраховуються обмеження елементів системи, і формується файл з описом складу системи РБ. Графічне представлення файлу, що описує структуру «ідеального» стану системи РБ зображено на рисунку 3.6.

Графічне зображення схеми опису загроз ІБ системи РБ показано на рисунку 3.7.

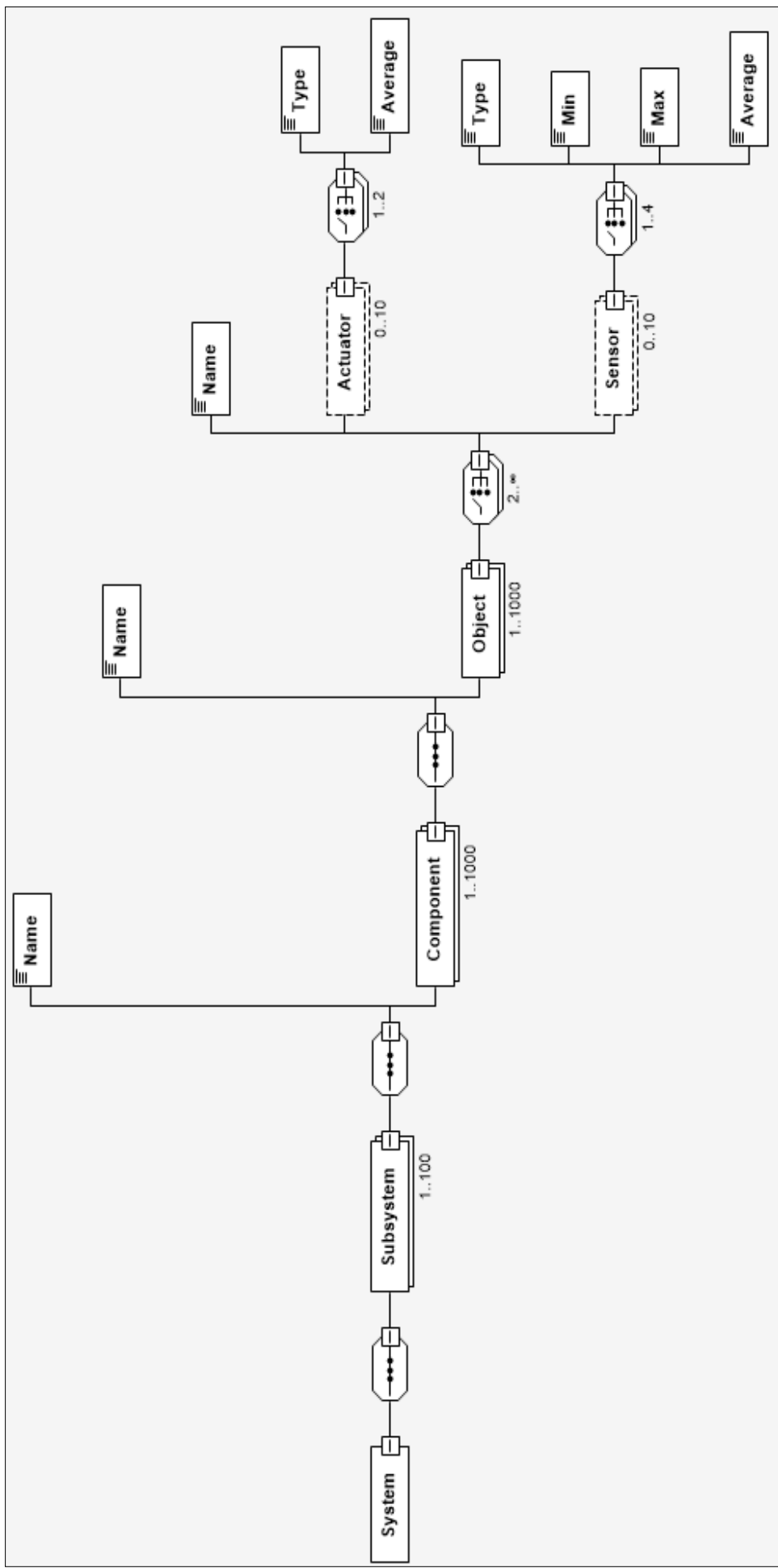


Рисунок 3.5 – Графічне зображення схеми опису складу системи «Розумний дім»

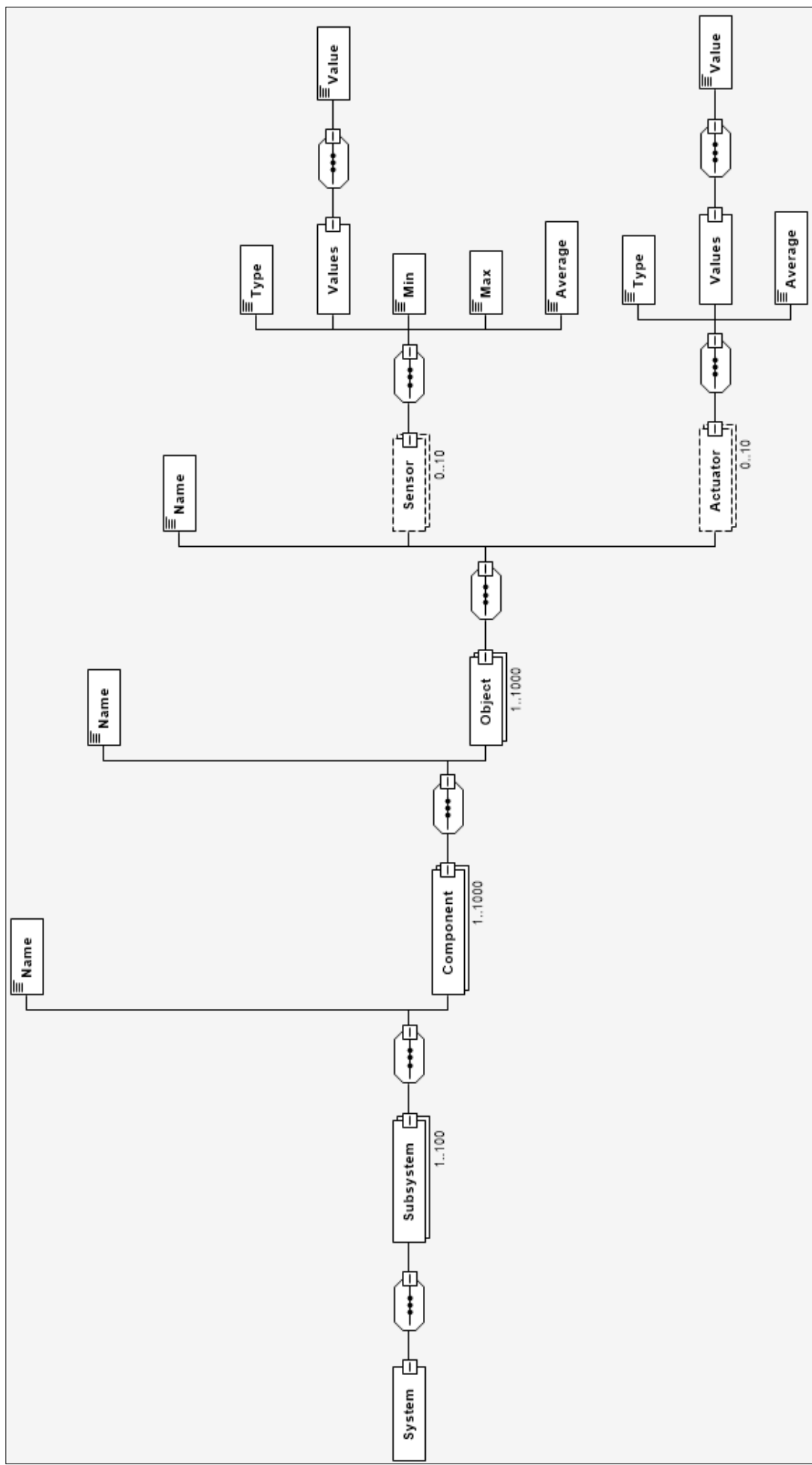


Рисунок 3.6 – Графічне зображення схеми опису «ідеального» стану систем

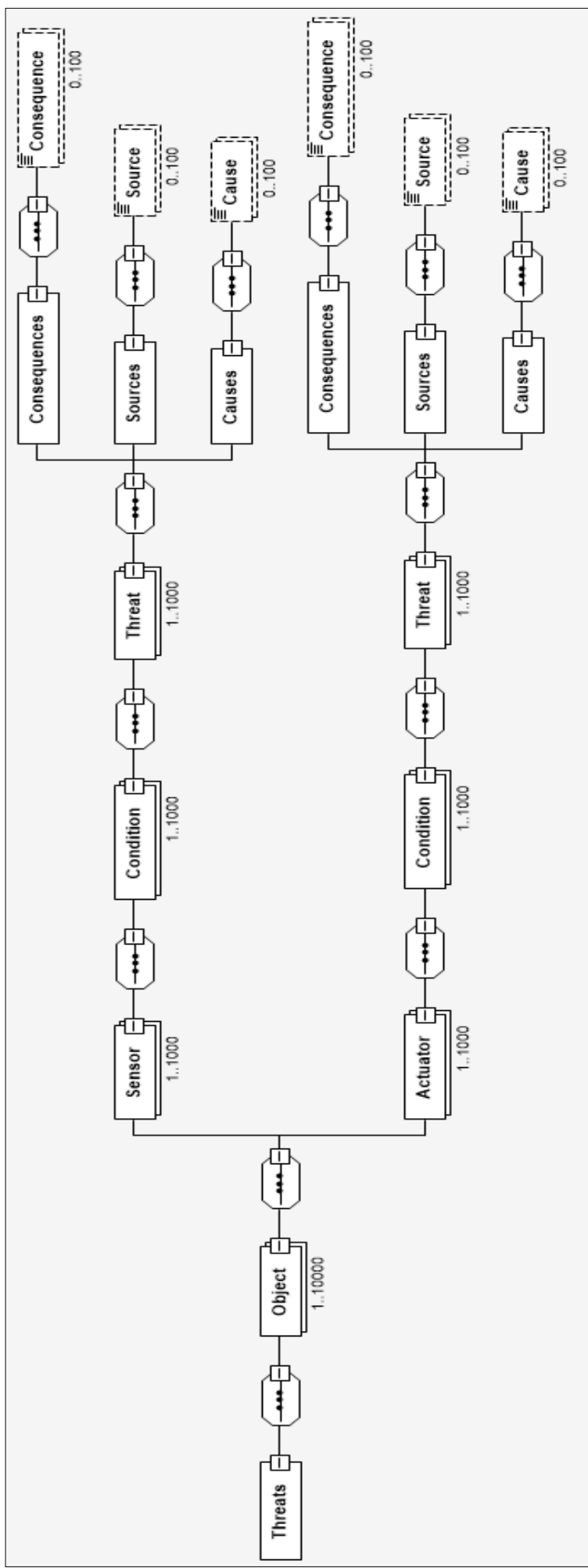
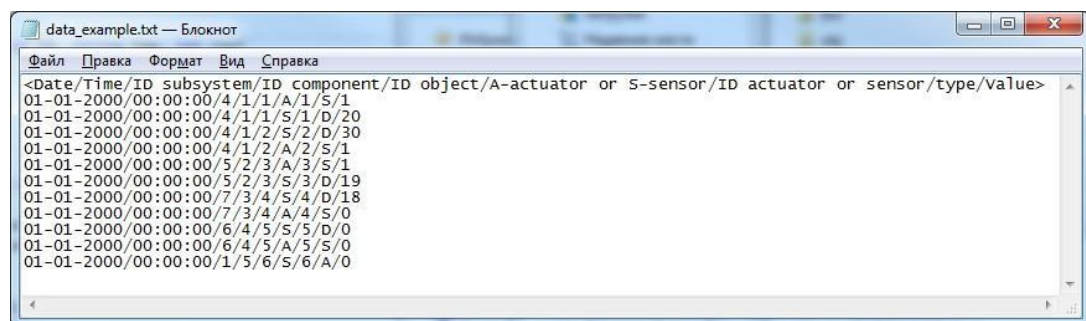


Рисунок 3.7 – Графічне представлення схеми загроз системи «Розумний будинок»

### 3.3 Розробка системи інформаційної безпеки

До компонентів системи інформаційної безпеки «Розумного будинку», що розробляється, доданий компонент генерування даних. Цей компонент використовує файл-зразок, що визначає склад системи РБ і значення показників об'єктів управління. Значення показників файлу-зразка використовуються при генерації випадкових значень у файлах, що генеруються. Файли генеруються у окремому потоці програми. Лістинг 1 демонструє потік генерування файлів (кожні дві секунди) з даними та запуску їх моніторингу (Додаток А). Приклад вмісту файлу-зразка зображено на рисунку 3.8.

The image shows a Notepad window titled "data\_example.txt - Блокнот". The window contains a list of data entries in a structured format. The first line is a header: "<Date/Time/ID subsystem/ID component/ID object/A-actuator or S-sensor/ID actuator or sensor/type/value>". The following lines are data entries, each representing a record with fields for date, time, subsystem ID, component ID, object ID, actuator/sensor type, and value. The entries are as follows:

```
<Date/Time/ID subsystem/ID component/ID object/A-actuator or S-sensor/ID actuator or sensor/type/value>
01-01-2000/00:00:00/4/1/1/A/1/S/1
01-01-2000/00:00:00/4/1/1/S/1/D/20
01-01-2000/00:00:00/4/1/2/S/2/D/30
01-01-2000/00:00:00/4/1/2/A/2/S/1
01-01-2000/00:00:00/5/2/3/A/3/S/1
01-01-2000/00:00:00/5/2/3/S/3/D/19
01-01-2000/00:00:00/7/3/4/S/4/D/18
01-01-2000/00:00:00/7/3/4/A/4/S/0
01-01-2000/00:00:00/6/4/5/S/5/D/0
01-01-2000/00:00:00/6/4/5/A/5/S/0
01-01-2000/00:00:00/1/5/6/S/6/A/0
```

Рисунок 3.8 – Приклад файлу зразка

При запуску програми необхідно визначити склад системи РБ, після чого будуть доступні кнопки управління моніторингом. Інтерфейс основного вікна для визначення складу системи представлений на рисунку 3.9.

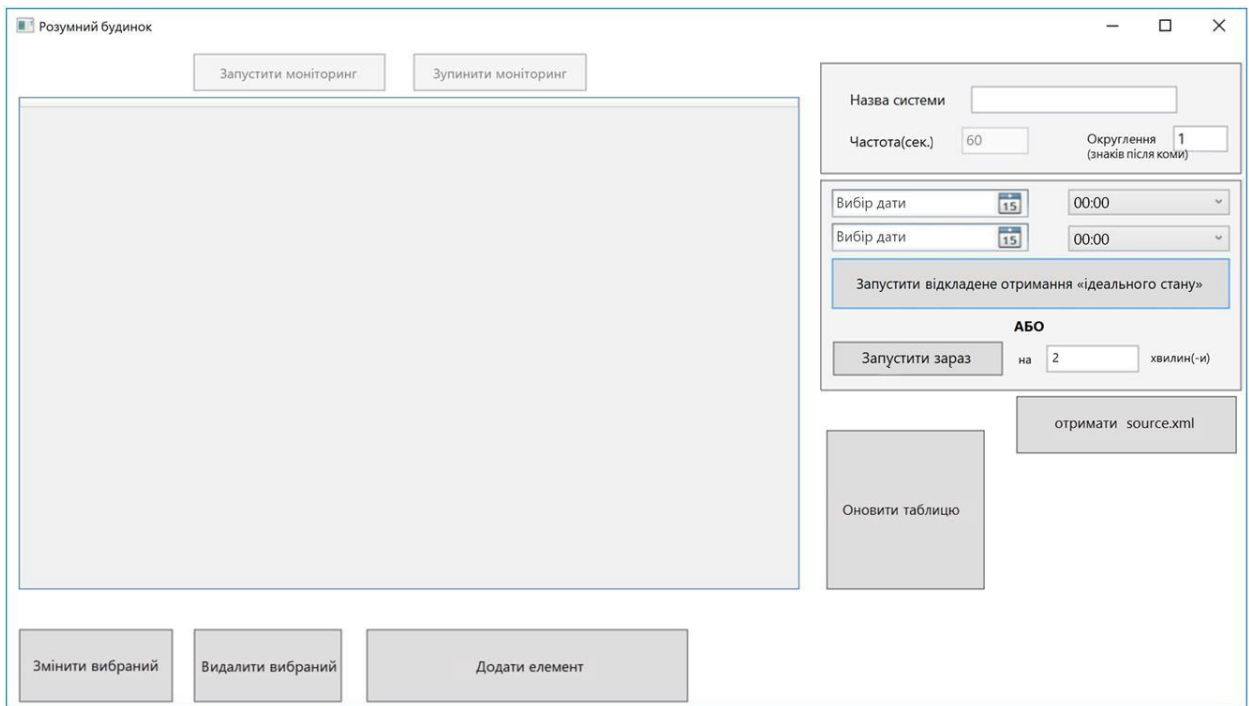


Рисунок 3.9 – Інтерфейс основного вікна визначення складу системи

Для автоматичного отримання складу системи РБ необхідно в правій частині основного вікна визначити налаштування отримання «ідеального стану»:

- вказати назву системи РБ;
- визначити частоту збору даних (частота генерування файлів з даними);
- вказати ступінь заокруглення значень показань (за замовчуванням встановлено заокруглення до одного знака після коми);
- позначити дату та час початку та закінчення відкладеного збору даних, або визначити час закінчення для збору даних у поточний час.

У додатку А наведено лістинг 2, який демонструє фрагмент коду, що додає елементи «датчик» або «виконавчий механізм», і дані елементів XML-файлу «ідеального стану системи».

Для самостійного визначення користувачем складу системи РБ необхідно на основному вікні вибрати кнопку «Додати елемент», після чого відкриється вікно для додавання/редагування елемента. Інтерфейс вікна додавання елемента представлений на рисунку 3.10.



The screenshot shows a window titled "Створення/редагування елемента" (Creation/editing of an element). It features a dropdown menu for selecting a subsystem ("--Обрати підсистему"), two text input fields for entering names ("Введіть назву"), a dropdown menu for selecting an element ("--Обрати елемент"), a button to view received data ("Переглянути отримані дані"), and a button to add an element ("Додати елемент").

Рисунок 3.10 – Інтерфейс вікна додавання елемента

Користувачеві необхідно вибрати зі списку підсистему, ввести назви для компонента підсистеми та об'єкта управління (або вибрати з існуючих, якщо вони були додані раніше), вибрати створюваний елемент. Після вибору елемента, стануть доступні список для визначення типу елемента, текстові поля для визначення обмежень. Приклад заповненої форми створення датчика елемента управління лампочки представлений на рисунку 3.11.

Рисунок 3.11 – Інтерфейс заповненого вікна додавання елемента

Після визначення складу системи у таблиці на основному вікні програми з'являться дані про елементи. Дані в таблиці можна сортувати за спаданням та зростанням будь-якого стовпця. Приклад заповненої таблиці з даними представлено на рисунку 3.12.

SubsystemName	SubsystemID	ComponentName	ComponentID	ObjectName	ObjectID	Average	Min	Max	ID	Type	Name	State
Система освітлення	4	Стельове освітлення	1	Лампочка	1	83	--	--	1	Switch	Actuator	green
Система освітлення	4	Стельове освітлення	1	Лампочка	1	24,5	11	38	1	Digital	Sensor	green
Система освітлення	4	Стельове освітлення	1	Лампочка	2	33,7	5	54	2	Digital	Sensor	green
Система освітлення	4	Стельове освітлення	1	Лампочка	2	83	--	--	2	Switch	Actuator	green
Система опалення	5	Опалення житлових кімнат	2	Батарея	3	16,2	7	28	3	Digital	Sensor	green
Система опалення	5	Опалення житлових кімнат	2	Батарея	3	83	--	--	3	Switch	Actuator	green
Система кондиціонування	7	Спільне кондиціонування	3	Кондиціонер	4	18,2	1	30	4	Digital	Sensor	green
Система кондиціонування	7	Спільне кондиціонування	3	Кондиціонер	4	16	--	--	4	Switch	Actuator	green
Система вентиляції	6	Вентиляція вікон	4	Вікно	5	0	0	30	5	Digital	Sensor	green
Система вентиляції	6	Вентиляція вікон	4	Вікно	5	16	--	--	5	Switch	Actuator	green
Система керування і зв'язку	1	Сервер	5	Системний блок	6	1	0	1	6	Analog	Sensor	green

Рисунок 3.12 – Екранна форма таблиці з даними

Лістинг 3 демонструє метод заповнення таблиці даними з XML-файлу зі складом системи (Додаток А).

Такі дані елементів як назва та обмеження можна змінювати, для цього необхідно виконати подвійне натискання на потрібній комірці або вибрати рядок у таблиці та натиснути кнопку «Змінити вибраний». У першому варіанті відкриється вікно для редагування конкретного значення даних елемента (рис. 3.13), у другому відкриється заповнене даними елемента вікно додавання/редагування елемента, описане раніше.

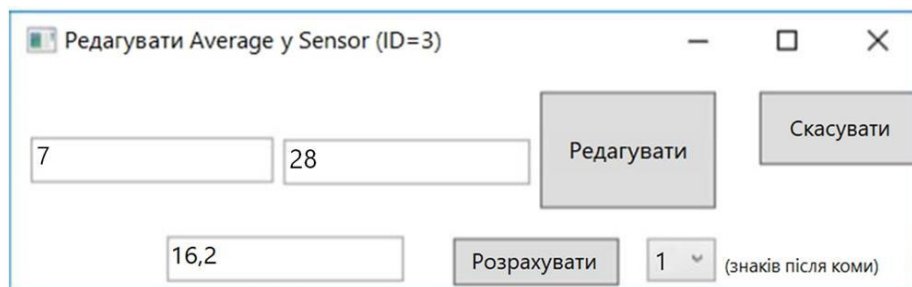


Рисунок 3.13 – Приклад редагування осередку таблиці

Для запуску моніторингу стану системи РБ на головному вікні потрібно вибрати кнопку «Запустити моніторинг». За потреби моніторинг стану РБ можна залишити, вибравши кнопку «Зупинити моніторинг». При виявленні погроз та невідповідності показань датчика або виконавчого механізму об'єктів керування, у таблиці з даними змінюється значення елемента в стовпці «State» (статус) і рядок елемента підсвічується кольором. Приклад зовнішнього вигляду таблиці з даними, у яких виявлено загрози, представлено на рисунку 3.14.

SubsystemName	SubsystemID	ComponentName	ComponentID	ObjectName	ObjectID	Average	Min	Max	ID	Type	Name	State
Система освітлення	4	Стельове освітлення	1	Лампочка	1	83	--	--	1	Switch	Actuator	green
Система освітлення	4	Стельове освітлення	1	Лампочка	1	24,5	11	38	1	Digital	Sensor	red
Система освітлення	4	Стельове освітлення	1	Лампочка	2	33,7	5	54	2	Digital	Sensor	red
Система освітлення	4	Стельове освітлення	1	Лампочка	2	83	--	--	2	Switch	Actuator	green
Система опалення	5	Опалення житлових кімнат	2	Батарея	3	16,2	7	28	3	Digital	Sensor	red
Система опалення	5	Опалення житлових кімнат	2	Батарея	3	83	--	--	3	Switch	Actuator	green
Система кондиціонування	7	Спільне кондиціонування	3	Кондиціонер	4	18,2	1	30	4	Digital	Sensor	red
Система кондиціонування	7	Спільне кондиціонування	3	Кондиціонер	4	16	--	--	4	Switch	Actuator	green
Система вентиляції	6	Вентиляція вікон	4	Вікно	5	0	0	30	5	Digital	Sensor	red
Система вентиляції	6	Вентиляція вікон	4	Вікно	5	16	--	--	5	Switch	Actuator	green
Система керування і зв'язю	1	Сервер	5	Системний блок	6	1	0	1	6	Analog	Sensor	red

Рисунок 3.14 – Приклад таблиці з виявленими загрозами

Метод визначення загрози представлений у лістингу 4 (Додаток А). У методі перевірки даних при виявленні помилок визначається їхній вигляд. Список описаних помилок показаний у таблиці 3.1.

Таблиця 3.1 – Список помилок

№ помилки	Вид помилки
1	Помилка у форматі дати
2	Помилка у форматі часу
3	Помилка у форматі ID підсистеми
4	Помилка у форматі ID компонента
5	Помилка у форматі ID об'єкта
6	Помилка у вигляді виду елемента
7	Помилка у форматі ID елемента
8	Помилка у форматі типу елемента
9	Помилка у форматі значення елемента
33	Не знайдено підсистему із заданим ID
44	Не знайдено компонент із заданим ID
55	Не знайдено об'єкт із заданим ID
771	Не знайдено датчик із заданим ID
772	Не знайдено виконавчий механізм із заданим ID
8810	У датчика із заданим ID не вказано тип
881	Невірний тип датчика
8820	У виконавчого механізму із заданим ID не вказано тип
882	Невірний тип виконавчого механізму
99	Дані сенсора не відповідають обмеженням
100	Неправильна кількість елементів даних
103	У вхідному масиві відсутні дані щодо підсистеми
104	У вхідному масиві відсутні дані щодо компоненту
105	У вхідному масиві відсутні дані щодо об'єкту
1071	У вхідному масиві відсутні дані щодо датчика
1072	У вхідному масиві відсутні дані про виконавчий механізм

Подвійне натискання на статус елемента відкриває вікно з детальною інформацією про можливу загрозу. Приклад відомостей про загрозу, виявлену в датчику розтину корпусу сервера системи управління, представлений на рисунку 3.15.

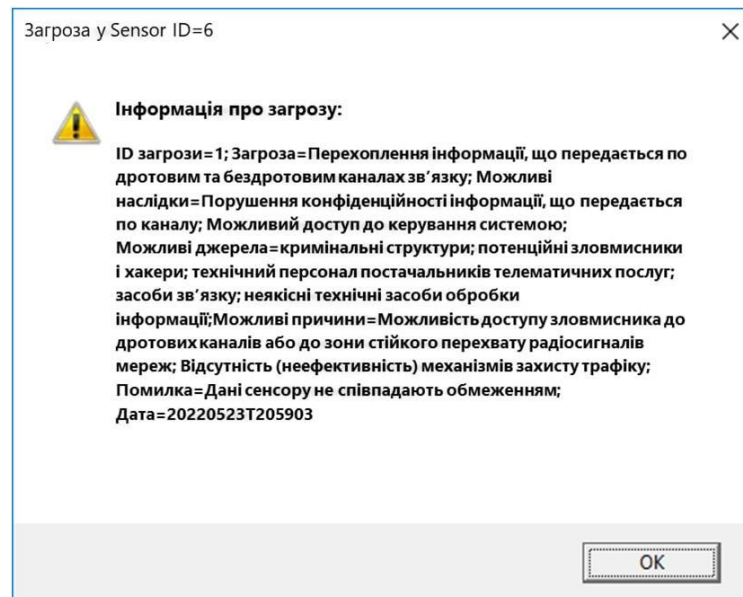


Рисунок 3.15 – Вікно відомостей про можливі загрози

За відсутності даних про загрозу система ІБ надає користувачеві дані про помилку та дату її виявлення. Приклад відомостей про невідому загрозу, виявлену в датчику відкриття вікна, представлений на рисунку 3.16.

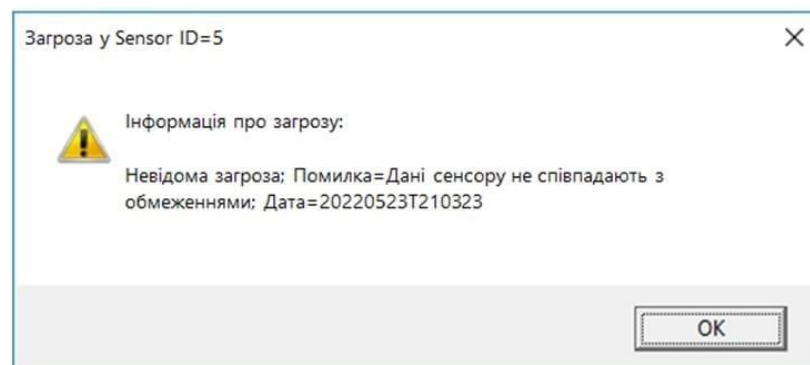


Рисунок 3.16 – Вікно відомостей про невідому загрозу

В результаті виконання кваліфікаційної бакалаврської роботи було розроблено застосунок, необхідний для оцінки загроз та аналізу безпеки системи «Розумного будинку». Застосунок визначає склад системи «Розумного будинку» та встановлює ліміти для відображення об'єктів керування користувачами, забезпечує моніторинг даних системи РБ та формує повідомлення про стан системи РБ, у випадку виявлення загроз інформує користувача повідомленням.

#### 4. ОСНОВНІ ВИСНОВКИ

Розглянуто проблеми створення та напрями розвитку кіберфізичних систем відповідно до досягнень та сучасних концепцій застосування комп'ютерних, інформаційних та телекомунікаційних технологій. Описані складові кіберфізичної системи та напрацювання в їх створенні. Сформовано проблеми побудови кіберфізичних систем та запропоновано принципи побудови апаратно-програмної платформи для створення прикладних кіберфізичних систем. Розглянуто напрями досліджень кіберфізичних систем та очікувані наукові результати. Було наведено приклади конкретних загроз безпеки кіберфізичних систем, та запропоновано шляхи їх запобігання.

Аналіз технології «Розумний будинок» та існуючих рішень захисту інформаційних систем вказав на відсутність єдиної методології опису систем РБ, а отже – єдиної методології виявлення та оцінки загроз для інформаційної безпеки РБ. На основі проведеного аналізу складено модель системи інформаційної безпеки систем РБ та складено перелік найбільш ймовірних загроз інформаційній безпеці цих систем.

У роботі представлено класифікацію ймовірних загроз системам інформаційної безпеки РБ, яка поєднує можливі загрози з об'єктами управління системи «Розумний дім». Ця класифікація дозволяє ідентифікувати та оцінити загрози ІБ у системі РБ.

Розроблено систему захисту інформації РБ, алгоритми роботи системи, структури опису даних та прототип ІТ-системи. Основні функції системи: визначення складу системи РБ та встановлення лімітів для відображення об'єктів керування користувачами, моніторинг даних системи РБ та формування повідомлень про стан системи РБ. Якщо в системі є невідома загроза, розроблений додаток попередить користувача про підозрілі дані.

Розроблена система була використана для імітаційних експериментів, під час яких у системі ІБ було створено вихідну систему для моделювання та аналізу системи РБ для виявлення можливих загроз ІС.

Результати роботи дозволять довести роботу комп'ютерної системи та зробити висновок про можливість розробки захисту для системи «Розумний будинок».

Розроблена система використана щодо імітаційних експериментів, під час яких у системі ІБ генерувалися вихідні дані для імітації дій системи РБ, та аналізу виявлення можливих загроз інформаційної безпеки.

Результати роботи доводять працездатність інформаційної системи та дозволяють зробити висновок про можливість розробки засобу захисту систем «Розумний будинок», який налаштовує користувач під його систему розумного будинку. Цей засіб захисту здійснює моніторинг стану всієї системи та оцінює знайдені загрози.



## СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Cyber-physical system. URL:  
[https://en.wikipedia.org/wiki/Cyber-physical\\_system#:~:text=A%20cyber-physical%20system%20\(CPS,monitored%20by%20computer-based%20algorithms.&text=Examples%20of%20CPS%20include%20smart,systems%2C%20and%20automatic%20pilot%20avionics](https://en.wikipedia.org/wiki/Cyber-physical_system#:~:text=A%20cyber-physical%20system%20(CPS,monitored%20by%20computer-based%20algorithms.&text=Examples%20of%20CPS%20include%20smart,systems%2C%20and%20automatic%20pilot%20avionics)
2. Аулін В. В., Гриньків А. В., Лисенко С. В., Лівіцький О. М., Головатий А. О., Дьяченко В. О. Принципи побудови та функціонування мобільних кіберфізичних систем // Принципи побудови та функціонування кіберфізичної системи технічного сервісу автотранспортної та мобільної сільськогосподарської техніки. Інженерія природокористування. 2020. №3 (17). С. 101-110. URL: <http://ena.lp.edu.ua/bitstream/ntb/32392/1/16-90-95.pdf>.
3. A Review of CPS 5 Components Architecture for Manufacturing Based on Standards Ahmadzai Ahmadi, Chantal Cherifi, Vincent Cheutet, Yacine Ouzrout. Colombo, Sri Lanka, 2017. 6 p.
4. E. A. Lee, "Cyber physical systems: design challenges," Proc. of IEEE ISORC, 2008. P. 363-369.
5. "What is Middleware?". Middleware.org. Defining Technology. (2008). Retrieved 2013-08-11.
6. The Universal Home API (UHAPI) Forum developed and maintained the A/V API for multiple types of home consumer electronics products. URL: <http://www.uhapi.org/home>
7. Єрмошин В.В, Невоїт Я.В. Аналіз і оцінка ризиків інформаційної безпеки для банківських та комерційних систем. URL: <http://journals.dut.edu.ua/index.php/dataprotect/article/view/182>
8. С. Н. Liu, Y. Zhang, "Cyber physical systems: architectures, protocols and applications," 2016. 249 p.
9. S. Zeadally, N. Jabeur, "Cyber-Physical System Design with Sensor Networking Technologies," 2015. 416 p.

10. A. A. Cardenas, S. Amin, S. Sastry, "Secure control: towards survivable cyber-physical systems," Proc. of WCPS, 2008. P. 495-500.
11. M. Krotofil, A. Cardenas, "Resilience of process control systems to cyberphysical attacks," Proc. of the 18th Nordic Conference on Secure IT Systems, 2013. P. 166-182.
12. H. Kopetz, "Real-Time Systems: Design Principles for Distributed Embedded Applications," 2011. 378 p.
13. D. Xu, M. Tu, M. Sanford, L. Thomas, D. Woodraska, W. Xu, "Automated security test generation with formal threat models," IEEE Trans. on Dependable and Secure Computing, 2012. Vol. 9. No. 4. P. 525-539.
14. "Program mühendisliyinin aktual elmi – praktiki problemləri" I respublika konfransı, Bakı, 17 may 2017-ci il.
15. A. Perrig, J. A. Stankovic, D. Wagner, "Security in wireless sensor networks," Commun. ACM, 2004. Vol. 47. No. 6. P. 53-57.
16. L. Eschenauer, V. Gligor, "A key-management scheme for distributed sensor networks," Proc. of ACM CCS'02, 2002. P. 41-47.
17. A. Perrig, R. Szewczyk, J. Tygar, V. Wen, D. E. Culler, "Spins: security protocols for sensor networks," Wireless Networks, 2002. Vol. 8. No. 5. P. 521-534
18. C. Karlof, N. Sastry, D. Wagner, "Tinysec: a link layer security architecture for wireless sensor networks," Proc. of SenSys, 2004. P. 162-175.
19. A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," IEEE Trans. on Dependable and Secure Computing, 2004. Vol. 1. No. 1. P. 11-32.
20. L. Adleman, "An abstract theory of computer viruses". Proc. of CRYPTO, 1990. P. 354-374.
21. Develop a Security Strategy for Cyber-Physical Systems 13 квітня 2021 року, автор: Сьюзен Мур. URL:  
<https://www.gartner.com/smarterwithgartner/develop-a-security-strategy-for-cyber-physical-systems>

22. Toll Group's Operations Shut Down by Yet Another Ransomware Attack  
Alicia Hope May 14 2020. URL: <https://www.cpomagazine.com/cyber-security/toll-groups-operations-shut-down-by-yet-another-ransomware-attack/>
23. BlueScope Steel hit by cyber attack causing worldwide system shutdown of operations, Jessica Clifford, Fri 15 May 2020. URL: <https://www.abc.net.au/news/2020-05-15/bluescope-steel-cyber-attack-shut-down-kembla-ransomware/12251316>
24. Importance of Security in Cyber-Physical Systems, Feb 01, 2022 / Krontech  
URL: <https://krontech.com/importance-of-security-in-cyber-physical-systems>
25. Darktrace AI Stops Sophisticated Ransomware Attack at South African Financial Services Provider, Cambridge, UK Tuesday April 5, 2022. URL: <https://www.darktrace.com/en/press/2022/413/>
26. Малюк В.М., Букреев Д.С. Аналіз загроз інформаційної безпеки системи «розумний дім» // Праці міжнародного симпозиуму «Надійність та якість». 2012. Т.1.
27. Push-Button Manor Popular Mechanics Magazine /NY, 1950. № 410426-298 с.
28. Гаврилов А.В. Штучний домовик // Штучний інтелект та прийняття рішень. 2012. - 02/2012. - С.77-89.
29. Перегудов Ф.І., Тарасенко Ф.П. Основи системного аналізу, 1997. 396 с.
30. Про захист інформації в інформаційно-комунікаційних системах: Закон України від 16.12.2020 № 1089-IX / ВР України. URL: <https://zakon.rada.gov.ua/laws/show/80/94-%D0%B2%D1%80#Text>
31. Дарія Хохлова. Dojo – пристрій для захисту "розумного" будинку від злому. URL: <https://vc.ru/p/dojo>
32. TechCrunch. Dojo Is Designed Protect Your Smart Home From Itself URL: <https://techcrunch.com/2015/11/19/dojo-labs>
33. Дарія Хохлова. CUJO – система захисту "розумного" будинку. URL: <https://vc.ru/p/cujo>

34. Нове рішення Cisco безпеки наступного покоління (NGFW+NGIPS+AMP). URL: <https://habrahabr.ru/company/cisco/blog/237759>
35. Снігуров А.В., Ткаченко О.О., Кравченко О.Д. Ризики інформаційної безпеки систем, побудованих за технологією «Розумний дім».
36. СанПіН 2.2.4.548-96. Гігієнічні вимоги до мікроклімату виробничих приміщень.

## Додаток А

### Лістинг програми

Лістинг 1 - Потік генерування та моніторингу файлів

```
private static void ThreadStartGenerateFiles() //потік для генерування
файлів даних та запуску їх моніторингу
{
while(do_generate)
{
mw.GenerateFile();//Метод генерування файлів-даних

if(do_monitoring)
{
while(generated_files.Count! = 0) //поки є файли для генерування
{
Monitoring();
}
}
Thread.Sleep(2000); //2sec
}
}
```

Лістинг 2 - Фрагмент додавання елементів

```
//додавання нового елемента з одним значенням
if(xdoc.Descendants(elem_tagname).Where(x => x.Attribute("ID").Value ==
elem_id).Count() == 0)
{
XmlElement elem = doc.CreateElement(elem_tagname);
elem.SetAttribute("ID", elem_id);
root_for_element.AppendChild(elem);
XmlElement elem1 =
doc.CreateElement("Type"); string type_ = "";
if(elem_tagname == "Actuator") //actuator
{
type_ = Actuator_types[type];
}
else if(elem_tagname == "Sensor") //sensor
{
type_ = Sensor_types[type];
}
elem1.InnerText = type_;
elem.AppendChild(elem1);
XmlElement elem2 = doc.CreateElement("Values");
elem.AppendChild(elem2);
element_for_value = elem2;
XmlElement elem3 = doc.CreateElement("Value");
elem3.SetAttribute("DateTime", date + "T" + time);
elem3.InnerText = value;
elem2.AppendChild(elem3);
}
else //Додавання наступних отриманих значень елемента
{
xpath = "System/Subsystem[@ID='" + subsystem_id + "']/Component[@ID='" +
component_id + "']/Object[@ID='" + object_id + "']/"+ elem_tagname + "[@ID='"
+ elem_id + "']/Values";
XmlElement elem4 = doc.CreateElement("Value");
elem4.SetAttribute("DateTime", date + "T" + time);
elem4.InnerText = value;
doc.SelectSingleNode(xpath).AppendChild(elem4);
}
```

Лістинг 3 - Метод для заповнення таблиці

```

private void fulltable() //заповнення таблиці даними з source.xml
{
    //дані з xml
    XmlDocument doc = XmlDocument.Load (path);
    var result = doc.Descendants("Average").Select(x =>new
    {
        SubsystemName =
            x.Parent.Parent.Parent.Element("Name").Value,
        SubsystemID =
            x.Parent.Parent.Parent.Parent.Attribute("ID").Value,
        ComponentName =
            x.Parent.Parent.Parent.Element("Name").Value, ComponentID =
            x.Parent.Parent.Parent.Attribute("ID").Value, ObjectName =
            x.Parent.Parent.Element("Name").Value,
        ObjectID =
            x.Parent.Parent.Attribute("ID").Value,
        Average = x.Value,
        Min = x.Parent.Element("Min") != null?
            x.Parent.Element("Min").Value : "--",
        Max = x.Parent.Element("Max") != null?
            x.Parent.Element("Max").Value : "--", ID =
            x.Parent.Attribute("ID").Value,
        Type =
            x.Parent.Element("Type").Value,
        Name = x.Parent.Name.ToString(),
        State = "green"
    });
    dgrid.ItemsSource = результат; //заповнюємо таблицю

    elements_count = dgrid.Items.Count; //записуємо кількість елементів
    системи УДФullLists(); //заповнюємо списки з наявними ID
    елементів підсистем, компонентів, об'єктів,
}

```

Лістинг 4 - Метод визначення загрози

```

private static
int GetThreatID(string tagname, string elemid, int dev) //Визначення ID загрози
{
    int threat_id = -1;
    XmlDocument xdoc = XmlDocument.Load
    (path); string name = "";

    switch (tagname)
    {
        case "S":
            tagname = "Sensor"; break;
        case "A":
            tagname = "Actuator"; break;
    }
    //отримуємо назву об'єкта, в якій виявлено загрозу
        var v = xdoc.Descendants("Object").Where(x =>
x.Element(tagname).Attribute("ID").Value == elemid).Elements("Name");
    foreach (XElement e in v)
    { name = e.Value; }
    XmlDocument doc = new XmlDocument();
    doc.Load("../..\\threat.xml");
        if (doc.SelectSingleNode("//Object[@Name='"+ name
+ "' ]//'+tagname+'//Condition[@Type='"+ GetConditionType(dev)
+ "' ]") != null) //якщо в threat.xml є об'єкт із заданою умовою
    {
        threat_id = Convert.ToInt32(doc.SelectSingleNode("//'+
tagname +'//Condition[@Type='"+ GetConditionType(dev)
+ "' ]//Threat/@ID").Value);
    }
        if (doc.SelectSingleNode("//Object[@Name='"+ name + "' ]//'+
tagname +'//Condition[@Type='"+ "00" + "' ]") != null) //Загроза для будь-
якого відхилення
    {
        threat_id = Convert.ToInt32(doc.SelectSingleNode("//'+
tagname +'//Condition[@Type='"+ "00" + "' ]//Threat/@ID").Value);
    }
    return threat_id;
}

```

XML Schema-файл для XML-файлу складу системи

```

<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="System" type="systemcomponents"/>
<xs:complexType name="systemcomponents">
<xs:sequence>
  <xs:element name="Subsystem" type="subsystemcomponents"
minOccurs="1" maxOccurs="100"/>
</xs:sequence>
<xs:attribute name="Name" type="xs:string" use="required"/>
</xs:complexType>
<xs:complexType name="subsystemcomponents">
<xs:sequence>
<xs:element name="Name" type="xs:string"/>
  <xs:element name="Component" type="componentcomponents"
minOccurs="1" maxOccurs="1000"/>
</xs:sequence>
<xs:attribute name="ID" use="required">
<xs:simpleType>
<xs:restriction base="xs:positiveInteger">
<xs:minInclusive value="1"/>
<xs:maxInclusive value="100"/>
</xs:restriction>
</xs:simpleType>
  </xs:attribute>
</xs:complexType>
<xs:complexType name="componentcomponents">
<xs:sequence>
<xs:element name="Name" type="xs:string"/>
<xs:element name="Object" type="objectcomponents" minOccurs="1"
maxOccurs="1000"/>
</xs:sequence>
<xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
</xs:complexType>
<xs:complexType name="objectcomponents">
<xs:choice minOccurs="2" maxOccurs="unbounded">
<xs:element name="Name" type="xs:string"/>
<xs:element name="Actuator" type="actuatorcomponents" minOccurs="0"
maxOccurs="10"/>
<xs:element name="Sensor" type="sensorcomponents" minOccurs="0" maxOccurs="10"/>
</xs:choice>
<xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
</xs:complexType>
<xs:complexType name="sensorcomponents">
<xs:choice minOccurs="1" maxOccurs="4">
<xs:element name="Type">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="Analog"/>
<xs:enumeration value="Digital"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="Min" type="xs:integer"/>
<xs:element name="Max" type="xs:integer"/>
<xs:element name="Average">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:integer">
<xs:attribute name="Same" use="required">

```



```

<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="Yes"/>
<xs:enumeration value="No"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
  </xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
</xs:choice>
<xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
</xs:complexType>
<xs:complexType name="actuatorcomponents">
<xs:choice minOccurs="1" maxOccurs="2">
<xs:element name="Type">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="Switch"/>
<xs:enumeration value="Action"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="Average">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="Same" use="required">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="Yes"/>
<xs:enumeration value="No"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
  </xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
</xs:choice>
<xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
</xs:complexType>
</xs:schema>

```

XML Schema-файл для XML-файлу ідеального складу системи

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="System" type="systemcomponents"/>
<xs:complexType name="systemcomponents">
<xs:sequence>
  <xs:element name="Subsystem" type="subsystemcomponents"
minOccurs="1" maxOccurs="100"/>
</xs:sequence>
<xs:attribute name="Name" type="xs:string" use="required"/>
<xs:attribute name="DateFrom" type="xs:string" use="required"/>
<xs:attribute name="DateTo" type="xs:string" use="required"/>
<xs:attribute name="EverySec" type="xs:positiveInteger" use="required"/>
<xs:attribute name="Round" type="xs:positiveInteger" use="required"/>
</xs:complexType>
<xs:complexType name="subsystemcomponents">
<xs:sequence>
<xs:element name="Name" type="xs:string"/>
  <xs:element name="Component" type="componentcomponents"
minOccurs="1" maxOccurs="1000"/>
</xs:sequence>
<xs:attribute name="ID" use="required">
<xs:simpleType>
<xs:restriction base="xs:positiveInteger">
<xs:minInclusive value="1"/>
<xs:maxInclusive value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
<xs:complexType name="componentcomponents">
<xs:sequence>
<xs:element name="Name" type="xs:string"/>
<xs:element name="Object" type="objectcomponents" minOccurs="1"
maxOccurs="1000"/>
</xs:sequence>
<xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
</xs:complexType>
<xs:complexType name="objectcomponents">
<xs:sequence>
<xs:element name="Name" type="xs:string"/>
<xs:element name="Sensor" type="sensorcomponents" minOccurs="0" maxOccurs="10"/>
<xs:element name="Actuator" type="actuatorcomponents" minOccurs="0"
maxOccurs="10"/>
</xs:sequence>
<xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
</xs:complexType>
<xs:complexType name="sensorcomponents">
<xs:sequence>
<xs:element name="Type">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="Analog"/>
<xs:enumeration value="Digital"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="Values" type="sensorvaluescomponents"/>
<xs:element name="Min" type="xs:positiveInteger"/>

```

```

<xs:element name="Max" type="xs:positiveInteger"/>
<xs:element name="Average">
<xs:complexType>
  <xs:simpleContent>
    <xs:extension base="xs:positiveInteger">
      <xs:attribute name="Same" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Yes"/>
            <xs:enumeration value="No"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
</xs:complexType>
<xs:complexType name="actuatorcomponents">
<xs:sequence>
<xs:element name="Type">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="Switch"/>
<xs:enumeration value="Action"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="Values" type="actuatorvaluescomponents"/>
<xs:element name="Average">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="Same" use="required">
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:enumeration value="Yes"/>
    <xs:enumeration value="No"/>
  </xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
</xs:complexType>
<xs:complexType name="actuatorvaluescomponents">
<xs:sequence>
<xs:element name="Value">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="actuatorvaluetype">
<xs:attribute name="DateTime" type="xs:dateTime" use="required"/>
  </xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>

```

```
</xs:sequence>
</xs:complexType>
<xs:simpleType name="actuatorvaluetype">
<xs:restriction base="xs:string">
<xs:enumeration value="ON"/>
<xs:enumeration value="OFF"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="sensorvaluetype">
<xs:restriction base="xs:integer"/>
</xs:simpleType>
<xs:complexType name="sensorvaluescomponents">
<xs:sequence>
<xs:element name="Value">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="sensorvaluetype">
<xs:attribute name="DateTime" type="xs:dateTime" use="required"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

XML Schema-файл для XML-файлу загроз «Розумного будинку»

```

<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Threats">
<xs:complexType>
<xs:sequence>
<xs:element name="Object" minOccurs="1" maxOccurs="10000">
<xs:complexType>
<xs:sequence>
<xs:element name="Sensor" maxOccurs="1000">
<xs:complexType>
<xs:sequence>
<xs:element name="Condition" minOccurs="1" maxOccurs="1000">
<xs:complexType>
<xs:sequence>
<xs:element name="Threat" type="threatcomponents" minOccurs="1"
maxOccurs="1000"/>
</xs:sequence>
<xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
<xs:attribute name="Type" type="xs:positiveInteger" use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Actuator" maxOccurs="1000">
<xs:complexType>
<xs:sequence>
<xs:element name="Condition" minOccurs="1" maxOccurs="1000">
<xs:complexType>
<xs:sequence>
<xs:element name="Threat" type="threatcomponents" minOccurs="1"
maxOccurs="1000"/>
</xs:sequence>
<xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
<xs:attribute name="Type" type="xs:positiveInteger" use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="Name" type="xs:string" use="required"/>
<xs:attribute name="SubsystemID" type="xs:positiveInteger" use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:complexType name="threatcomponents">
<xs:sequence>
<xs:element name="Consequences" type="consequencescomponents"/>
<xs:element name="Sources" type="sourcescomponents"/>
<xs:element name="Causes" type="causescomponents"/>
</xs:sequence>
<xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
<xs:attribute name="Name" type="xs:string" use="required"/>
</xs:complexType>
<xs:complexType name="consequencescomponents">
<xs:sequence>

```

```
<xs:element name="Consequence" minOccurs="0" maxOccurs="100">
  <xs:complexType mixed="true">
    <xs:attribute name="ID" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="sourcescomponents">
  <xs:sequence>
    <xs:element name="Source" minOccurs="0" maxOccurs="100">
      <xs:complexType mixed="true">
        <xs:attribute name="ID" type="xs:positiveInteger"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="causescomponents">
  <xs:sequence>
    <xs:element name="Cause" minOccurs="0" maxOccurs="100">
      <xs:complexType mixed="true">
        <xs:attribute name="ID" type="xs:positiveInteger"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:schema>
```

Метод отримання ідеального складу

```
private void Get_source_usual(bool now, string datefrom, string dateto)
{
    DateTime d1 = DateTime.Now; //дата та час при натисканні кнопки
    для отримання ідеального складу
    DateTime current = d1;
    DateTime next; DateTime d2 =
    d1;
    int tm = 0;
    string dates = ""; //ЛОГИ для методу
    string doc_path = "..\\..\\..\\source_usual.xml"; try
    {
        string system_name = textbox.Text; //дані з форми string
        everysec = textbox1.Text; //
        string round = textbox2.Text; //
        string datefrom_ = datefrom; //дата початку в форматі xml
        datefrom_ = datefrom_.Replace(".", ":");
        datefrom_ = datefrom_.Replace(" ", "T");
        string dateto_ = dateto; //дата завершення у форматі xml
        dateto_ = dateto_.Replace(".", ":");
        dateto_ = dateto_.Replace(" ", "T"); int
        current_it = 0; //номер ітерації

        int count = GetCount(datefrom, dateto, everysec); //розраховуємо
        кількість необхідних ітерацій
        dates += "count=" + count.ToString() + "\r\n";
        StreamWriter sw = File.CreateText(doc_path); //створюємо файл із коричневим тегом
        та атрибутами

        {
            sw.WriteLine("<?xml version=\"1.0\" encoding=\"utf-8\"
            standalone=\"no\"?>");
            sw.WriteLine("<System Name=\"" + system_name + "\""
            DateFrom=\"" + datefrom_ + "\" DateTo=\"" + dateto_ + "\" EverySec=\"" +
            everysec + "\" Round=\"" + round + "\"
            xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
            xsi:noNamespaceSchemaLocation=\"schema source usual.xsd\">");
            sw.WriteLine("</System>");
        }
        sw.Close();

        string xpath;
        XmlDocument doc = new XmlDocument(); String
        condition = null;
        String date = null; String
        time = null;
        String subsystem_id = null; String
        component_id = null; String object_id
        = null; String elem_tagname = null;
        String elem_id = null; String type =
        null;
        String value = null; String
        input_path = "";
        if (!now) //чекаємо початку збору даних
        {
            TimeSpan ts = Convert.ToDateTime(datefrom) - d1;
            tm = Convert.ToInt32(Math.Ceiling(ts.TotalSeconds)); tm = tm *
            1000;
        }
    }
}
```

```

dates = "wait=" + tm.ToString() + "msec" + "\r\n";
System.Threading.Thread.Sleep(tm);
dates += "after sleep time is " + DateTime.Now.ToString() + "\r\n";
}

        while (current_it <= count)//
{
current = DateTime.Now;//поточний час дата
next = current.AddSeconds(Convert.ToDouble(everysec));//час та дата для
наступної ітерації

dates += "current_it=" + current_it + "current="      + current.ToString()+ ",
next=" + next.ToString() + "\r\n";

        if (current_it != 0)
{
TimeSpan ts = next - current;
tm = (Convert.ToInt32(Math.Floor(ts.TotalSeconds)) - 2) * 1000;
System.Threading.Thread.Sleep(tm);
        dates += "after sleep " + tm.ToString() + "msecin
current_it=" + current_it + ",time                is " +
DateTime.Now.ToString() + "\r\n";
}

current_it++;
input_path = GenerateFile();//генеруємо файл

        foreach (string line in
File.ReadLines(input_path, Encoding.UTF8))//розбір файлу
{
if (line[0] != '<')
{
XmlNode root_for_element = null; XmlNode
root_for_object = null; XmlNode root_for_component =
null; XmlNode root = null;
XmlElement element_for_value = null; doc.Load(doc_path);
XDocument xdoc = XDocument.Load(doc_path); int s_id;
int c_id; int o_id; int ind;
string tmp = "";

ind = line.IndexOf("/");
date = line.Substring(0, ind); tmp = line.Remove(0,
ind + 1);

ind = tmp.IndexOf("/");
time = tmp.Substring(0, ind); tmp = tmp.Remove(0,
ind + 1);

ind = tmp.IndexOf("/");
subsystem_id = tmp.Substring(0, ind); tmp = tmp.Remove(0,
ind + 1);

ind = tmp.IndexOf("/");
component_id = tmp.Substring(0, ind); tmp = tmp.Remove(0,
ind + 1);

ind = tmp.IndexOf("/");
object_id = tmp.Substring(0, ind);
tmp = tmp.Remove(0, ind + 1);

ind = tmp.IndexOf("/");

```



```

elem_tagname = tmp.Substring(0, ind); if (elem_tagname ==
"A")

{ elem_tagname = "Actuator"; } else if
(elem_tagname == "S")
{ elem_tagname = "Sensor"; } tmp = tmp.Remove(0,
ind + 1);

ind = tmp.IndexOf("/");
elem_id = tmp.Substring(0, ind); tmp = tmp.Remove(0,
ind + 1);

ind = tmp.IndexOf("/");
type = tmp.Substring(0, ind); tmp = tmp.Remove(0,
ind + 1);

value = tmp;

        s_id =
xdoc.Descendants("Subsystem").Where(x => x.Attribute("ID").Value
== subsystem_id).Count();
        c_id =
xdoc.Descendants("Component").Where(x => x.Attribute("ID").Value
== component_id).Count();
        o_id =
xdoc.Descendants("Object").Where(x => x.Attribute("ID").Value
== object_id).Count();

        if (s_id != 0 && c_id != 0 && o_id != 0) // створення
нового елемента в існуючій підсистемі, об'єкті та компоненті
        {
condition = "all_old";
        }
else if (s_id == 0) // все нове
condition = "all_new";
        }

        else if (s_id != 0 && c_id == 0) // створення нового
компонента та об'єкта в існуючій підсистемі
        {
condition = "component_new";
        }

        else if (s_id != 0 && c_id != 0 && o_id ==
0) // створення нового об'єкта в існуючому компоненті та підсистемі
        {
condition = "object_new";
        }

switch (condition)
{
        case "1": // actuator or sensor
        {
// додаємо новий елемент з єдиним значенням if
(xdoc.Descendants(elem_tagname).Where(x =>
x.Attribute("ID").Value == elem_id).Count()
== 0)
                {

```

```

doc.CreateElement(elem_tagname);
XmlElement elem =

elem.SetAttribute("ID", elem_id); root_for_element.AppendChild(elem);

XmlElement elem1 = doc.CreateElement("Type"); string type_ = "";
if (elem_tagname == "Actuator">//actuator
{
    type_ = Actuator_types[type];
}
else if (elem_tagname == "Sensor">//sensor
{
    type_ = Sensor_types[type];
}
elem1.InnerText = type_; elem.AppendChild(elem1);

XmlElement elem2 = doc.CreateElement("Values"); elem.AppendChild(elem2);
element_for_value = elem2;
time);XmlElement elem3 = doc.CreateElement("Value");
elem3.SetAttribute("DateTime", date + "T" +

elem3.InnerText = value; elem2.AppendChild(elem3);
}

else //Додання наступних отриманих значень елемента

{
xpath = "System/Subsystem[@ID='" + subsystem_id+ "']/Component[@ID='" +
component_id + "']/Object[@ID='" + object_id + "']/" + elem_tagname
+ "[@ID='" + elem_id + "']/Values";

time);
XmlElement elem4 = doc.CreateElement("Value"); elem4.SetAttribute("DateTime",
date + "T" +

elem4.InnerText = value; doc.SelectSingleNode(xpath).AppendChild(elem4);
}
doc.Save(doc_path); break;
}

    case "all_old":///підсистема,компонент та об'єкт існують
    {
        xpath = "System/Subsystem[@ID='" +
subsystem_id + "']/Component[@ID='" + component_id + "']/Object[@ID='" +
object_id + "']";
root_for_element = doc.SelectSingleNode(xpath); goto case "1";
    }
    case "object_new":
    {
        if (root_for_object == null)//якщо прийшли
не з "component_new", то отримуємо компонент, в якому будемо створювати об'єкт
    {
xpath = "System/Subsystem[@ID='" + subsystem_id
+ "']/Component[@ID='" + component_id + "']";
root_for_object = doc.SelectSingleNode(xpath);
    }
XmlElement elem0 = doc.CreateElement("Object"); elem0.SetAttribute("ID",
object_id); root_for_object.AppendChild(elem0);

XmlElement elem01 = doc.CreateElement("Name"); elem0.AppendChild(elem01);

```

```

root_for_element = elem0; goto case "1";

    }
    case "component_new":
    {
        if (root_for_component == null)//якщо сбди
прийшли не з "all_new", то отримаємо підсистему для створення компоненту
    {
        xpath = "System/Subsystem[@ID='" +
+ "' ]";
        subsystem_id root_for_component =
    }

doc.SelectSingleNode(xpath
h);

XmlElement elem1 = doc.CreateElement("Component"); elem1.SetAttribute("ID",
component_id); root_for_component.AppendChild(elem1);
XmlElement elem11 = doc.CreateElement("Name"); elem1.AppendChild(elem11);
root_for_object = elem1; goto case "object_new";

    }
    case "all_new":
    {
xpath = "System";
root = doc.SelectSingleNode(xpath);
XmlElement elem00 = doc.CreateElement("Subsystem"); elem00.SetAttribute("ID",
subsystem_id); root.AppendChild(elem00);
XmlElement elem001 = doc.CreateElement("Name"); elem00.AppendChild(elem001);
root_for_component = elem00; goto case "component_new";
    }
}
}

}
}

d2 = DateTime.Now;
dates += "after get data time is " + d2.ToString() + "\r\n";
DeleteGeneratedFiles();//видаляємо згенеровані файли

////////////////////////////////////
////////////////////////////////////average

String average;
doc.Load(doc_path);
XDocument xdoc2 = XDocument.Load(doc_path); var
all_sen = xdoc2.Descendants("Sensor"); var all_act =
xdoc2.Descendants("Actuator");

foreach (XElement sen in all_sen)
{
    var obj = sen.Descendants("Value")
        .Select(y => Convert.ToInt32(y.Value)); //список всіх
значень
сенсора

average = Math.Round(obj.Average(), Convert.ToInt32(round)).ToString(); int
min = obj.Min();
int max = obj.Max();
sen.Add(new XElement("Min", min.ToString()));
sen.Add(new XElement("Max", max.ToString()));

```

```

XElement aver = new XElement("Average", average.ToString());
sen.Add(aver);

if (min == max)
{
    aver.SetAttributeValue("Same", "Yes");
}
else
{
    aver.SetAttributeValue("Same", "No");
}

foreach (XElement act in all_act)
{
    var obj = act.Descendants("Value")
        .Select(y => y.Value); // список всіх значень активатора

    int on = 0;
    bool same = false; string tmp =
        "";
    int obj_count = obj.Count();

    foreach (string val in obj)
    {
        if (val == "1")
        {
            on++;
        }
        tmp = val;
    }

    if (on == obj_count || on == 0)
    {
        same = true;
    }

    XElement aver = new XElement("Average");
    act.Add(aver);

    if (same)
    {
        aver.SetAttributeValue("Same", "Yes"); aver.Value = (tmp
            == "1") ? "100" : "0";
    }
    else
    {
        aver.SetAttributeValue("Same", "No");
        aver.Value = (Math.Round(Convert.ToDecimal(on * 100 /
            obj_count))).ToString
            (); //MessageBox.Show("obj.count=" + obj_count.ToString() +
            ", on=" +

```

```
on + ", aver value="+aver.Value.ToString());
}
}
xdoc2.Save(doc_path, SaveOptions.None);
}
catch (Exception ex)
{
    MessageBox.Show("Exception in get source_usual.xml=" +
ex.Message + "\r\n" + "Source=" + ex.Source + "\r\n" + "StackTrace=" +
ex.StackTrace);
}
finally
{
    MessageBox.Show("done everything");
    DateTime d3 = DateTime.Now;
    dates += "after all time is " + d3.ToString();
    //MessageBox.Show(dates);
    //MessageBox.Show("d1=" + d1.ToString() + "\r\n" +
"current=" + current.ToString() + "\r\n" + "d2=" + d2.ToString() + "\r\n"
+ "d3=" + d3.ToString());
}
}
```

Метод додання елемента

```
private void Add()////додання нових елементів
{
try
    {
        bool res_getid = getID();//false - помилка при отриманні ID для нових
елементи
в

if (res_getid == false || (is_new_component == null || is_new_object == null
|| /*is_sensor*/element_tagname == null ||
(is_new_component == true && is_new_object == false) ||
subsystem_id == null || subsystem_id == "" || component_id ==
null || component_id == "" ||
object_id == null || object_id == "" ||
element_id == null || element_id == "" || element_type ==
null || element_type == "" || element_average == null || element_average
== ""))
{
MessageBox.Show("Помилка при доданні елемента:" + "\r\n" + "тип
елемента = " + element_type + "\r\n" +
"average = " + element_average);
}

else
{
//вираз для коричневого елемента - нижній існуючий, в який
буде додано новий
string xpath;
string doc_path=file_path; XmlDocument doc = new
XmlDocument();

String condition=null;
XmlNode root_for_element=null; XmlNode
root_for_object=null; XmlNode
root_for_component = null; XmlNode root=null;

if (is_new_component == false && is_new_object ==
false)//створення елемента в існуючих об'єкті та компоненті
{
condition = "all_old";
doc.Load(doc_path);
}
else if (doc.SelectSingleNode("//Subsystem[@ID='" + subsystem_id + "']")
== null)//якщо такої підсистеми не існує
{
condition = "all_new";
if (!File.Exists(file_path))//створення нового файлу
{
StreamWriter sw = File.CreateText(doc_path);
{
sw.WriteLine("<?xml version=\"1.0\" encoding=\"utf-
8\"
standalone=\"no\"?>
");
sw.WriteLine("<System Name=\"system_name\"
```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="schema
source.xsd">");
sw.WriteLine("</System>");
}
sw.Close();
} doc.Load(doc_path);

}
else if (/*is_new_subsystem == false &&
*/is_new_component == true)//створення нового компоненту та об'єкта
{
condition = "component_new";
doc.Load(doc_path);
}
else if (/*is_new_component == false &&*/
is_new_object == true)//створення нового об'єкту в існуючому
компоненті
{
condition = "object_new";
doc.Load(doc_path);
}

switch (condition)
{
case "1":
{
//Create a new node - actuator or sensor
XmlElement elem = doc.CreateElement(element_tagname);
elem.SetAttribute("ID", element_id);
//Add the node to the document. root_for_element.AppendChild(elem);

XmlElement elem1 = doc.CreateElement("Type"); elem1.InnerText =
element_type; elem.AppendChild(elem1);

XmlElement elem2 = doc.CreateElement("Average");
elem2.SetAttribute("Same", "No"); elem2.InnerText = element_average;
elem.AppendChild(elem2);
if (element_tagname == "Sensor")
{
XmlElement elem3 = doc.CreateElement("Max"); elem3.InnerText =
element_max; elem.AppendChild(elem3);
XmlElement elem4 = doc.CreateElement("Min"); elem4.InnerText =
element_min; elem.AppendChild(elem4);
}
doc.Save(doc_path); MessageBox.Show("Готово"); break;
}
case "all_old":
{
xpath = "//Subsystem[@ID='" +
subsystem_id + "']/Component[@ID='" + component_id +
"']/Object[@ID='" + object_id + "']";
root_for_element = doc.SelectSingleNode(xpath); goto case "1";
}
case "object_new":
{

```

```

        if (root_for_object == null) //якщо прийшли
не з "component_new", то отримуємо компонент, в якому будемо створювати
об'єкт
    {
        xpath = "//Subsystem[@ID='" +
subsystem_id + "']/Component[@ID='" + component_id + "']";
root_for_object = doc.SelectSingleNode(xpath);
    }
XmlElement elem0 = doc.CreateElement("Object");
elem0.SetAttribute("ID", object_id);
root_for_object.AppendChild(elem0);

XmlElement elem01 = doc.CreateElement("Name"); elem01.InnerText =
object_name; elem0.AppendChild(elem01);

root_for_element = elem0; goto case "1";

    }
    case "component_new":
    {
        if (root_for_component == null) //якщо прийшли
не з "all_new", то отримуємо підсистему, в якій будемо створювати компонент
    {
        xpath = "//Subsystem[@ID='" + subsystem_id + "']"; root_for_component =
doc.SelectSingleNode(xpath);
        }
XmlElement elem000 = doc.CreateElement("Component");
elem000.SetAttribute("ID", component_id);
root_for_component.AppendChild(elem000);
XmlElement elem0001 = doc.CreateElement("Name"); elem0001.InnerText =
component_name; elem000.AppendChild(elem0001);

root_for_object = elem000; goto case "object_new";

    }
    case "all_new":
    {
xpath = "System";
root = doc.SelectSingleNode(xpath);

XmlElement elem00 = doc.CreateElement("Subsystem");
elem00.SetAttribute("ID", subsystem_id); root.AppendChild(elem00);
XmlElement elem001 = doc.CreateElement("Name"); elem001.InnerText =
subsystem_name; elem00.AppendChild(elem001);

        root_for_component = elem00;
        goto case "component_new";
    }
}

}
this.Close();

}
catch (Exception ee)
{
    MessageBox.Show("Ошибка добавления:
"+"\"r\n" +
ee.Message+"\"r\n"+ee.StackTrace+"\"r\n"+ee.Source);
}
}

```



## Метод моніторингу даних

```

private static void Monitoring()//поток для моніторингу даних в data.txt
{
string data = generated_files[0];
    int error = 0;//код помилки=номеру даних або =100, якщо не
співпадає кількість елементів у файлі
string tmp = "";//тимчасова змінна для збереження даних із рядка
int tmp_num = 1;//номер даниї: 1-date, 2-time, 3-id subsystem, 4-id
component, 5-id object, 6-element(S or A), 7-id element, 8-element type, 9-
element value int row = -1;//індекс рядка
int c_ind = -1;//індекс символу в рядку
string elemtagname="";//A-активатор або S-сенсор
string elemid = "";//id елемента для перевірки типу елемента
string xpath = "";//шлях для перевірки вкладеності
var file = File.ReadAllLines(data); List<string>
list1 = new List<string>(file);

if (list1.Count != elements_count)//якщо не співпадає кількість елементів
{
error = 100;
WriteLog(error, -1, data); int ind = 0;
string subsystem_id; string
component_id; string object_id;
string s_id;
string a_id; string tag;
//списки з елементами системи, визначеними у складі системи List<string>
act_source = Actuators_in_source;
List<string> sen_source = Sensors_in_source; List<string>
sub_source = Subsystems_in_source; List<string> com_source =
Components_in_source; List<string> obj_source =
Objects_in_source;
/////
//перевіряємо кількість підсистем) foreach (string str in list1)
{
tmp = "";
ind = str.IndexOf("/");
tmp = str.Remove(0, ind + 1);//delete date ind =
tmp.IndexOf("/");
tmp = tmp.Remove(0, ind + 1);//delete time ind =
tmp.IndexOf("/");
subsystem_id = tmp.Substring(0, ind);//subsystem id tmp =
tmp.Remove(0, ind + 1);//delete subsystem id ind =
tmp.IndexOf("/");
component_id = tmp.Substring(0, ind);//component_id tmp =
tmp.Remove(0, ind + 1);
ind = tmp.IndexOf("/");
object_id = tmp.Substring(0, ind);//object_id tmp =
tmp.Remove(0, ind + 1);
ind = tmp.IndexOf("/");
tag = tmp.Substring(0, ind); if (tag == "A")
{
tmp = tmp.Remove(0, ind + 1); ind =
tmp.IndexOf("/");
a_id = tmp.Substring(0, ind);
tmp = tmp.Remove(0, ind + 1);
act_source.Remove(a_id);
}
else
{

```

```

tmp = tmp.Remove(0, ind + 1); ind =
tmp.IndexOf("/");
s_id = tmp.Substring(0, ind); tmp =
tmp.Remove(0, ind + 1); sen_source.Remove(s_id);
}
sub_source.Remove(subsystem_id);
sub_source.Remove(component_id);
obj_source.Remove(object_id);
}
if (sub_source.Count > 0)
{
foreach (string s in sub_source)
{
ЛОГ
WriteLog(103, -1, data, Convert.ToInt32(s)); //запис помилки у
}
}
////////Перевіряємо кількість компонентів // else
{
if (com_source.Count > 0)
{
foreach (string s in com_source)
{
В ЛОГ
WriteLog(104, -1, data, Convert.ToInt32(s)); //запис помилки
}
}
else////////перевіряємо кількість objects//
{
if (obj_source.Count > 0)
{
foreach (string s in obj_source)
{
WriteLog(105, -1, data, Convert.ToInt32(s));
}
}
}
else////////Actuators and Sensors
{
if (act_source.Count > 0)
{
foreach (string s in act_source)
{
WriteLog(1072, -1, data, Convert.ToInt32(s)); int th_id =
GetThreatID("Actuator",s,100); Bad_actuators.Add(Convert.ToInt32(s),
Get_threat_info(th_id, 1072, data.Substring(10, 15))); //додаємо в список
активаторів із загрозами
}
}
}
if (sen_source.Count > 0)
{
foreach (string s in sen_source)
{
WriteLog(1071, -1, data, Convert.ToInt32(s)); int th_id =
GetThreatID("Sensor",s,100)Bad_sensors.Add(Convert.ToInt32(s),
Get_threat_info(th_id, 1071, data.Substring(10,15))); //додаємо у список
сенсорів із загрозами
}
}
}
}

```

```

}
}
}

else
{
foreach (string str in list1)
{
error = 0; tmp = ""; tmp_num =
1;
c_ind = -1; elemtagname = "";
elemid = ""; row++;
xpath = "";

foreach (char c in str)
{
c_ind++;

if (!c.Equals('/'))
{
tmp += c;
}
if (c.Equals('/') || c_ind == str.Length - 1)
{
switch (tmp_num)
{
case 6:
elemtagname = tmp; break;
case 3:
xpath += "//Subsystem[@ID='" + tmp + "']"; break;
case 4:
xpath += "//Component[@ID='" + tmp + "']"; break;
case 5:
xpath += "//Object[@ID='" + tmp + "']"; break;
case 7:
elemid = tmp; break;
}

error = CheckDataStructure(tmp_num,
tmp); //перевіряємо правильність структури кожного рядка в отриманих
даних
if (error != 0) //помилка в структурі отриманих даних
{
WriteLog(error, row, data);
}
else if (tmp_num == 3 || tmp_num == 4 || tmp_num ==
5 || tmp_num == 7 || tmp_num == 8 || tmp_num == 9) // немає помилки у структурі,
перевіряємо склад
{
if (tmp_num != 9) // якщо це не дані з елемента, а id
{
error = CheckSystemStructure(tmp_num,
tmp, elemtagname, elemid, xpath); //перевіряем совпадает ли состав системы
}
if (error != 0) //ошибка в составе
{
WriteLog(error, row, data);
}
else if (error == 0 && tmp_num == 9) //если это
элемент
данные с
а
{

```

```

error =
        CheckValue(tmp, elemtagname,
elemid); //перевіряємо дані елемента на відповідність обмеженням //метод повертає
%відхилення або 0
if (error != -1) //помилка в даних елемента
{
if (elemtagname == "S")
{
        WriteLog(99, row, data,
Convert.ToInt32(elemid) error = GetThreatID(elemtagname,
)); error); //threat id elemid, if
(Bad_sensors.ContainsKey(Convert.ToInt32(ele
mid)))
        {
Bad_sensors.Remove(Convert.ToInt32(ele
mid));
        }
        Bad_sensors.Add(Convert.ToInt32(lemi
d),
Get_threat_info(error, 99, data.Substring(10,15)); //додаємо до списку сенсорів з
загрозами
}
if (elemtagname == "A") //actuator
{
        error = GetThreatID(elemtagname, elemid,
error); //threat
id
        }
if (error != -1) //якщо знайдена загроза
{
        WriteLog(error, -1, "",
Convert.ToInt32(elemid)
});
        }
else //если нет ошибок в данных
{
        File.Delete(data); //видаляємо файл
        }
        }
tmp = ""; tmp_num++;
}
}
}
}
generated_files.RemoveAt(0);
if (Bad_actuators.Count != 0 || Bad_sensors.Count != 0)
{
//updatetable(); Write_dictionary_to_list();
Bad_sensors.Clear(); Bad_actuators.Clear();
}
}
}

```